

# VXI/VME-PCI8000 SERIES FOR WINDOWS 95/NT

This document contains information to help you understand the components of your kit, determine where to start setting up your kit, and learn about the NI-VXI/VISA features.

## Contents

---

What Do You Have? .....	2
Hardware .....	2
Software .....	2
Printed Documentation .....	2
Acrobat (Online) Documentation .....	3
Available Options .....	4
Where Do You Start? .....	4
What Is NI-VXI? .....	5
What Is VISA? .....	6
NI-VXI/VISA Release Notes .....	6
New Utilities .....	6
Supported Application Development Environments .....	7
New Features and Terminology .....	7
Window Mapping .....	8
MITE DMA .....	9
Shared Memory .....	10
Remote Controllers .....	10
Enhancements to the NI-VXI Software .....	11
Compatibility .....	11
System Configuration Functions .....	11
Low-Level VXIbus Access Functions .....	11
High-Level VXIbus Access Functions .....	12
Local Resource Access Functions .....	12

# What Do You Have?

---

Your VXI/VME-PCI8000 kit contains hardware, software, and documentation. You may also have ordered some optional equipment or software.

## Hardware

Your hardware includes the PCI-MXI-2, which you install in your PCI-based computer. You also get either the VXI-MXI-2 or VME-MXI-2 mainframe extender, which you install in your VXI or VME mainframe.



**Note:** *You do not receive any hardware if you ordered the NI-VXI/VISA software-only kit.*

## Software

Your software disks are specific to either Windows 95 or Windows NT and are labeled as follows:

- *NI-VXI/VISA for PCI-MXI-2 for Windows 95* or
- *NI-VXI/VISA for PCI-MXI-2 for Windows NT*

## Printed Documentation

Aside from this document, your kit contains the following printed manuals:

- *Getting Started with Your PCI-Based MXI-2 Interface for Windows 95/NT* contains an overview of the MXI-2 hardware and the NI-VXI/VISA software, guides you through setting up your kit, and helps you get started with application development. You can also use this manual as a reference for the hardware and software default settings and to find the answers for commonly asked questions.
- *MXI-2 Configuration Reference Manual* contains information on configuring, installing, and cabling your MXI-2 hardware. You may not need to use this manual if you are performing a simple hardware installation that uses the standard default settings. However, you should keep this manual handy in case you decide to try a different switch or jumper setting at a later time.



**Note:** You do not receive the *MXI-2 Configuration Reference Manual* if you ordered the NI-VXI/VISA software-only kit.

# Acrobat (Online) Documentation

Your kit includes manuals that are not in printed form. These other manuals are available as Adobe Acrobat version 3.0 portable document format (PDF) files. You can view these manuals online, navigate through them, and print them from your computer. You must have at least Acrobat Reader 3.0 to view these documents. You can get the latest version of the reader directly from Adobe.

Adobe maintains a Bulletin Board Service (BBS), which you can use to upload files to Adobe Technical Support and download certain Adobe files. The number for the Adobe BBS is 206-623-6984. It is located in Seattle, Washington, which is in the Pacific time zone. If you live outside the United States or Canada, you are welcome to connect to the BBS, but may first want to try Adobe's Web site (<http://www.adobe.com/>) or FTP site (<ftp://ftp.adobe.com/>), which may be accessible at a significantly lower connection cost.

The following list shows the path and filenames of the Acrobat manuals that are available after you install your software (where Winxx is either Win95 or WinNT):

- `/NIVXI/Manuals/NI-VXIUsersMan.pdf`  
This is the *NI-VXI User Manual*, which describes the features of the NI-VXI software.
- `/NIVXI/Manuals/NI-VXIProgrammerMan.pdf`  
This is the *NI-VXI Programmer Reference Manual*, which describes in detail the VXI/VME function calls in the C/C++ and BASIC languages.
- `/VXIpnP/Winxx/Nivisa/Manuals/NI-VISAUsersMan.pdf`  
This is the *NI-VISA User Manual*, which describes how to program using VISA.
- `/VXIpnP/Winxx/Nivisa/Manuals/NI-VISAProgrammersMan.pdf`  
This is the *NI-VISA Programmer Reference Manual*, which describes in detail the attributes, events, and operations you use in NI-VISA.

Here are some tips on using PDF files efficiently:

- To view and print the online documentation, launch the Acrobat Reader and open the PDF documents. You can assemble your own printed documentation by printing out either the entire document or only the sections relevant to your application.
- To use hypertext links, click on any entry in the table of contents to jump directly to that topic's page. You can jump to major sections within each manual quickly and easily by clicking on the bookmarks. When you open your PDF documents, the bookmarks show.

A bookmark with a triangle in front of it has bookmarks within it. To reveal the next level, click on the triangle. To collapse to a higher level, click on the triangle.

- You can change the magnification of the documents either by selecting the magnifying glass tool in the toolbar and clicking anywhere on the manual page or by using the various options in the **View** menu.
- You can quickly find and go to every occurrence of any word or phrase in the entire document. Select the **Tools»Find...** option and type in the key word or phrase you want to find.
- The PDF file assigns consecutive numbers to each page throughout the document and displays this page number in the lower-left corner of the Acrobat window. When you choose the **View»Go To Page...** option or are printing pages, you must use the page numbers in the lower-left corner of the window, not the number displayed at the bottom center of each manual page or listed in the contents and index. The number shown in the bottom center of the page is for printed documents only.

## Available Options

You may have ordered the following optional software or accessories:

- LabVIEW
- LabWindows™/CVI
- MXI-2 cable



### Note:

These options are not available if you ordered the NI-VXI/VISA software-only kit.

Both LabVIEW and LabWindows/CVI integrate the VXI and VISA library interfaces that are required to support your PCI-MXI-2 products. You also get hundreds of complete instrument drivers, which are modular, source-code programs that handle the communication with your instrument to speed your application development.

## Where Do You Start?

---

1. Compare your kit contents with the description in the preceding [What Do You Have?](#) section. Contact National Instruments regarding any discrepancies.
2. Install LabVIEW or LabWindows/CVI before you install NI-VXI/VISA if you intend to use either of these application development environments as your programming choice.

The NI-VXI/VISA installer will update necessary LabVIEW and LabWindows/CVI files required to interface to your VXI/VME system.

3. Install the NI-VXI/VISA software using the Setup program. Refer to Chapter 2, *Setup*, in *Getting Started with Your PCI-Based MXI-2 Interface for Windows 95/NT* for details. You must install this software prior to installing the MXI-2 hardware. Doing so causes the system to properly recognize your PCI-MXI-2 at the next startup. See the *What is NI-VXI?* and *What is VISA?* sections for an explanation of the NI-VXI/VISA software.
4. Install your MXI-2 hardware into your computer and mainframe. Chapter 2, *Setup*, in your getting started manual summarizes the installation procedures. You can also find full details at the end of each configuration chapter in the *MXI-2 Configuration Reference Manual*.
5. After you finish installing the software and hardware, refer to Chapter 3, *Developing Your Application*, in your getting started manual to learn how you can use your VXI/VME system and to ensure it is operating properly.
6. Please refer to the following files for important information that may affect your application program, including known issues and software corrections with this release, and additional information relevant for NI-VXI and NI-VISA API development:
  - `c:\NIVXI\README.TXT` for NI-VXI information
  - `c:\VXI\np\Winxx\README.TXT` for NI-VISA information

You can also reference the National Instruments [www.natinst.com](http://www.natinst.com) or [ftp.natinst.com](http://ftp.natinst.com) sites for driver updates, examples, and product news.

## What Is NI-VXI?

---

The NI-VXI system-level software is the driver that controls your PCI-MXI-2 interface and VXI/VME system. NI-VXI includes a Resource Manager, an interactive configuration and troubleshooting program, libraries of software routines for test and measurement (T&M) programming, interactive control programs for both NI-VXI and NI-VISA, and a logging utility you can use for debugging your applications. You can use this software to seamlessly program multiple-mainframe configurations and have software compatibility across a variety of controller platforms.

# What Is VISA?

---

VISA is a standard I/O Application Programming Interface (API) for instrumentation programming. VISA by itself does not provide instrumentation programming functionality for interfaces other than Serial. VISA is a high-level API that calls into system-level drivers. As an example, the NI-VISA implementation of VISA uses the NI-VXI system-level driver for National Instruments VXI controllers. If you are a GPIB or GPIB-VXI user, you must also install the NI-488.2 system-level driver for National Instruments GPIB controllers.

VISA can control VXI, PXI, GPIB, or Serial instruments, making the appropriate driver calls depending on the type of instrument being used. VISA uses the same operations to communicate with instruments regardless of the interface type. For example, the 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, 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 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.

Because VISA is the industry standard for developing instrument drivers, most instrument drivers currently written by National Instruments use VISA and therefore support Macintosh, Windows 3.x, Windows 95, Windows NT, Solaris 1, Solaris 2, and HP-UX, if the system-level drivers are available for that platform.

## NI-VXI/VISA Release Notes

---

This section describes the new utilities and features in this release of NI-VXI/VISA for Windows 95/NT.

### New Utilities

This release of NI-VXI/VISA for Windows 95/NT includes two new utilities to help you configure, develop, and debug your system quickly: T&M Explorer and NI Spy.

You can use T&M Explorer to view your entire T&M system and configure various components. When you launch T&M Explorer, a list of your VXI, GPIB, and Serial devices appears on your screen. To view the properties of

each device (such as logical address, address space used, primary address), right-click on the device in the list. When you view the properties of a National Instruments device, you can configure the hardware settings directly from the properties list.

T&M Explorer replaces many earlier utilities, such as VXIedit and VISAconf, and integrates with the NI-DAQ Configuration Utility (for VXI-DAQ instruments). T&M Explorer also has new features, such as an option to run Resource Manager at startup, and troubleshooting to guide you through configuration conflicts and errors.

NI Spy tracks the calls your application makes to National Instruments T&M drivers, including NI-VXI, NI-VISA, and NI-488.2. It highlights functions that return errors, so you can quickly determine which functions failed during your development. NI Spy can also log your program's calls to these drivers, so you can check them for errors at your convenience.

## Supported Application Development Environments

This release of NI-VXI/VISA for Windows 95/NT supports the following Application Development Environments (ADEs):

- LabVIEW version 4.x
- LabWindows/CVI version 4.x
- Borland C/C++ version 4.5.x
- Microsoft Visual C/C++ version 4.x, 5.x
- Microsoft Visual Basic version 4.x, 5.x



**Note:** *Although NI-VXI and NI-VISA have been tested and found to work with these ADEs, other ADEs or higher versions of the ADEs listed above may also work.*

## New Features and Terminology

New functionality has been added to NI-VXI/VISA in four major areas to exploit features in the MITE ASIC. These features are as follows:

- Window mapping
- MITE DMA
- Shared memory
- Remote controllers

## Window Mapping

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

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

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

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

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

## Address Space

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

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

The *NI-VXI Programmer Reference Manual* implies that the error code `MAP_TIMEOUT (-8)` is returned when the window is in use. Because the MITE-based products have multiple hardware windows of variable size,



the meaning of this error has been modified. `MapVXIAddress()` now returns the error code `MAP_TIMEOUT (-8)` whenever there are not enough resources to map the window.

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

## Number of MITE Windows

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

## Memory for Allocating Data Structures

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

## MITE DMA

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

- MXI block
- MXI synchronous
- Burst mode (on the PCI bus)
- VME64 (on the VXI bus)

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

The `VXImove()` and `viMoveXX()` functions automatically use appropriate bus protocols and transfer types to efficiently perform the data transfer specified in the function. You can also use configuration options in T&M Explorer to instruct the NI-VXI/VISA software to use DMA channels for particular types of operations and to designate what protocols the channel should use. In addition, you can programmatically control which protocols to use in NI-VXI. See the NI-VXI online help or the *NI-VXI Programmer Reference Manual* for complete descriptions of `VXImove()` and other

high-level functions. Notice that previously written NI-VXI and NI-VISA code uses the DMA capabilities of the MITE without modification.

To take full advantage of the throughput of the DMA channels, you should perform 32-bit transfers where both the source and the destination are longword aligned. If you need to transfer character data between devices of different byte orders—for example, between a big-endian device and an Intel 80x86-based Windows NT PC—transfer the data as longwords but adjust the byte-ordering parameters in `VXImove()` to get the correct data in the most efficient manner.

NI-VXI Examples:

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

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

## Shared Memory

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

## Remote Controllers

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

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

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

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

## Enhancements to the NI-VXI Software

The following sections describe the additional options beyond what is documented in the *NI-VXI User Manual* and the *NI-VXI Programmer Reference Manual*.

### Compatibility

NI-VXI applications that follow the guidelines documented in the *NI-VXI User Manual* will work with NI-VXI for the PCI-MXI-2.

### System Configuration Functions

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

### Low-Level VXIbus Access Functions

Do not make any assumptions about the size and features of a window returned from `MapVXIAddress()`. You should use `GetWindowRange()` to determine the size of a window.

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

## High-Level VXIbus Access Functions

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

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

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

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

## Local Resource Access Functions

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

