NI-VISA™ FOR WINDOWS NT/95/98: WINNT, GWINNT, WIN95, AND GWIN95 FRAMEWORKS

NI-VISA is a standard I/O Application Programming Interface (API) for instrumentation programming.

NI-VISA can control VXI/VME, PXI, GPIB, 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 NT/95/98, Solaris 2, HP-UX, VxWorks, and Macintosh, as long as the appropriate *system*-level drivers are available for that platform.

321217E-01 *November 1998*

Contents

What Do You Have?

Your kit contains the following software and documentation.

Software

The VXI*plug&play* disks are labeled **NI-VISA for Windows NT/95/98 Version 2.0** and contain the following software components.

- VISA Dynamic Link Library
- VISA Microsoft C and Borland C Import Libraries
- VISA Microsoft Visual Basic Support Files
- VISA GPIB-VXI Component
- VPP-6 Compliant Windows NT/95/98 Installer

The NI-VISA software in this kit is compatible with the WIN95/GWIN95 and WINNT/GWINNT frameworks (the WIN95/GWIN95 frameworks include support for Windows 98). With NI-VISA installed on your computer, you can run any VXI*plug&play* software compatible with these

frameworks. This includes instrument drivers and executable soft front panel software included with VXI*plug&play*-compatible instruments from a variety of vendors.

Acrobat (Online) Documentation

Your kit includes online manuals in the form of Adobe Acrobat version 3.0 portable document format (PDF) files. The Acrobat manuals and their installed locations are as follows.

- The *NI-VISA User Manual* describes how to program using NI-VISA: **Start»Programs»VXIpnp»NI-VISA User Manual**
- The *NI-VISA Programmer Reference Manual* describes in detail the attributes, events, and operations you use in NI-VISA: **Start»Programs»VXIpnp»NI-VISA Programmer Reference Manual**

If you do not have Adobe Acrobat Reader 3.0, you can download a copy from the Adobe Web site at `http://www.adobe.com/`.

Utilities

This release of NI-VISA for Windows NT/95/98 includes 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, PXI, GPIB, and Serial devices appears on your screen. To view the properties of each device (such as logical address, address space used, and 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.

For PXI instruments, you must register device information with NI-VISA. See the section *Using NI-VISA with PXI* for information on how to do this, along with a description of the PXI functionality NI-VISA supports.

Supported Application Development Environments

This release of NI-VISA for Windows NT/95/98 supports the following Application Development Environments (ADEs):

- LabVIEW version 4.*x*, 5.*x*
- LabWindows™/CVI version 4.*x*, 5.*x*
- Borland C/C++ version 4.5.*x*
- Microsoft Visual C/C++ version 4.*x*, 5.*x*, 6.*x*
- Microsoft Visual Basic version 4.*x*, 5.*x*

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

Where Do You Start?

Step 1. Install LabVIEW and/or LabWindows/CVI

If you want to use LabVIEW and/or LabWindows/CVI as VISA application development environments, install them first. The NI-VISA installer will detect them and install the appropriate support files automatically.

If you install LabVIEW and/or LabWindows CVI after you install VISA, rerun the NI-VISA installer to get the latest VISA support files installed correctly.

Step 2. Install Your NI-VXI/NI-488.2 Driver Software

The installation method varies depending on the type of controller you purchased, as shown in the following table. Refer to the documentation that came with your controller for detailed installation instructions.

Controller	Installation
VXI-PCI8000 MXI Controller VXIpc 800/700/600 Series VXI-1394 Controller	The NI-VXI/VISA kit installs both the NI-VXI and NI-VISA driver software. You do not need to install this package separately, unless this is an upgrade. Refer to the *NI-VXI/VISA Read Me First* document for more information.
VXIpc-486 embedded controller	NI-VXI and NI-488.2 are already installed on your hard drive.

Controller	Installation
VXI-AT2010 MXI Controller	Install the NI-VXI for Windows 3.*x*/NT software. See Note.
VXI-AT4000 MXI Controller	Install the NI-VXI for Windows 3.*x* software. See Note.
GPIB-VXI/C controller or any National Instruments GPIB board that has a Windows NT/95/98 driver	Install the NI-488.2 for Windows NT/95/98 software.

You can also use a National Instruments serial board that works along with the serial ports on your computer. If you are using only serial I/O, you do not need to install NI-VXI or NI-488.2.

☞ **Note** ***Because this implementation of VISA is a 32-bit driver, it looks for 32-bit versions of NI-VXI and/or NI-488.2. If you are installing NI-VISA on Windows 95 and have only 16-bit drivers, please contact National Instruments to request the NI-VXI Upgrade for Windows 95 or the Compatibility Release for NI-488.2.***

Step 3. Install Your NI-VISA Software

Controller	Installation
VXIpc embedded controller	**New VXIpc controllers**—NI-VISA is already installed on your hard drive. The remainder of this document applies only if you want to use the Setup program to make changes to your software. **Upgrades**—If you received this kit to upgrade your VXIpc, continue with the remainder of this document to install the NI-VISA software.
All other cases	Install the NI-VISA software. The remainder of this document describes how to use the Setup program to install your software.

Start the Setup Program

Insert the NI-VISA diskette into your disk drive. You can run `SETUP.EXE` either from a DOS prompt or from the **Start»Run** menu.

The installer prompts you for the desired installation type. If you choose an option other than **Express**, the installer guides you with a series of prompts for additional information, including the components you want to install and the target directory.

After the installation completes successfully, if any system file has been modified, the following screen appears.

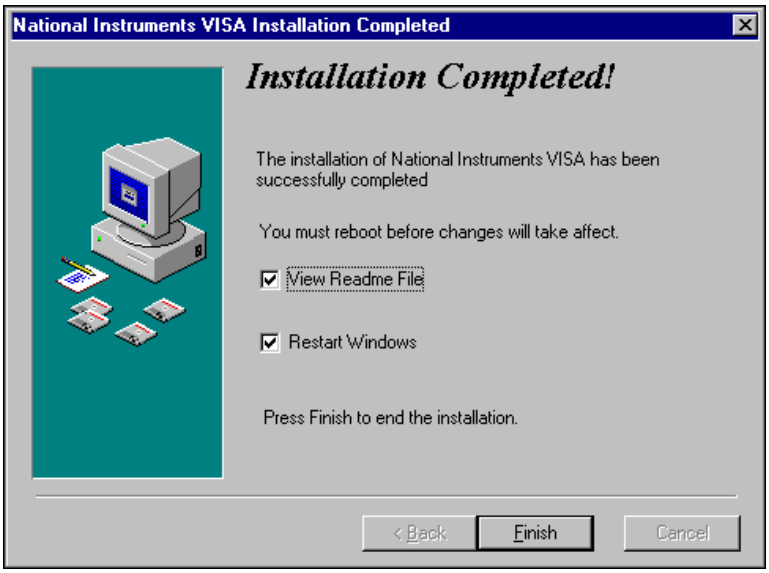

Figure 1. Setup Completion Dialog Box

You can either reboot the computer now so that the changes take effect immediately, or you can reboot the computer at a later time. However, do not use the NI-VISA software until you reboot your computer.

Updating the NI-VISA Setup

You can run the Setup program again to make changes to your software. The same screens and messages are displayed as when you installed the software. Select the **Custom** option to configure any changes.

Step 4. Check the Read-Me File

Refer to the `readme.txt` file included with your software for the most up-to-date information about this release of NI-VISA.

Using NI-VISA with PXI

☞ **Note** ***Read this section only if you intend to use NI-VISA with PXI. Skip this section if you are not running a PXI system.***

This section explains how to begin developing PXI-based applications with NI-VISA. Versions of NI-VISA prior to NI-VISA 2.0 detected PXI devices automatically, and gave users access to all PCI-based devices in the system. This created a potential stability risk by exposing core system components to unintentional access. With NI-VISA 2.0, you must register PXI device information using the PXI Setup Wizard.

Start the PXI Setup Wizard

Run *VXIpnp**os*\NIVisa\PXISWiz.EXE to start the PXI Setup Wizard. (*VXIpnp* refers to the directory in which you installed NI-VISA, and *os* refers to the operating system directory, either WinNT or Win95.) The wizard first prompts you for basic information NI-VISA needs in order to properly locate and create sessions for your PXI instrument.

Required setup information for PXI instruments includes:

- **Instrument Prefix**—The VXI*plug&play* compliant instrument prefix for the device.
- **PXI Manufacturer ID**—This 16-bit value is vendor-specific, and is unique among PCI-based device providers. The vendor ID number for National Instruments Corporation, for example, is 0x1093.
- **PXI Model Code**—The 16-bit device ID value is device-specific, defined by the instrument provider, and required for PCI-based devices.
- **Generates interrupts**—Checking this box informs NI-VISA that you wish to use the VISA event-handling model in response to hardware interrupts your PXI instrument generates. If you check this box, the wizard will prompt you for more information in later steps.

☞ **Note** ***In text boxes where numerical information is required, preceding the number with "0x" informs the wizard that the number entered is a hexadecimal value. The wizard assumes all other numeric entries are decimal values.***

When you have finished entering basic device information, choose **Next** to continue.

☞ **Note** ***If you did not check the* Generates interrupts *box, skip ahead to the* Device Manager Settings *section.***

Configure NI-VISA for PXI Interrupts

Enabling PXI interrupt handling within NI-VISA is a two step process. First, you must specify how your device detects a pending interrupt, and second, you must specify how to acknowledge an interrupt if one was found to be pending. The PXI Setup Wizard guides you through each of these steps.

Step 1: Detecting an Interrupt

Because PCI-based devices share one of four physical interrupt lines, VISA needs a way to query your device to find out if it requires service when the software detects a hardware interrupt. The wizard prompts you for the sequence of register accesses with which VISA makes this determination.

Types of Register Accesses

Register accesses assume the form of Reads, Writes, or Compares:

- Read—Performs a register read (of specified width) from a given offset relative to a given address space.
- Write—Performs a register write of a given value (of specified width) to a given offset relative to a given address space.
- Compare—Applies a user-supplied mask to the result of a Read operation, and compares the masked result with another user-supplied value.

For the purposes of determining whether or not your device is generating an interrupt, the Compare operation has an associated return value of True or False, dependent on the result of the compare. NI-VISA makes the assumption that the device is interrupting if and only if the result of all Compare operations is True. Because NI-VISA relies on the result of the Compare operation in making this assumption, at least one Compare operation must be present in an interrupt detection sequence for the sequence to be valid.

Create Register Operations

The **Interrupt Detection Info** dialog box prompts you to add, edit, or remove register operations.

When you click on the **Add a step before**, **Add a step after**, or **Edit a step** button, the PXI-VISA Interrupt Information (Register Operation) dialog box appears, as shown in Figure 2. Use the dropdown menus to specify the type of operation, its width, and the address space the operation applies to. Enter all other values directly into the text boxes. The relevance of each value is dependent on the operation as described in the previous section, *Types of Register Accesses*.

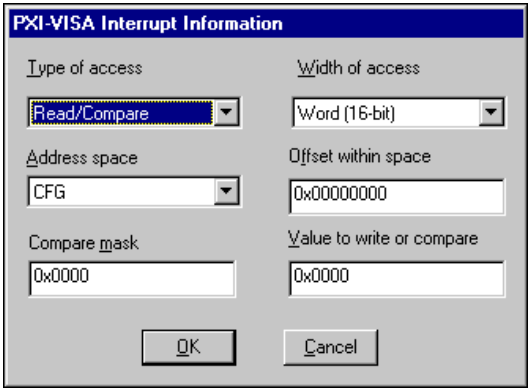

Figure 2. Register Operation Dialog Box

Adding a Register Operation

To add a register operation, choose either **Add a step before** or **Add a step after**, depending on where you want the new step to be placed relative to the currently selected step.

Editing a Register Operation

To edit an existing register operation, double-click on the entry (or highlight the entry and choose **Edit a step**).

Removing a Register Operation

To remove a register operation, highlight the entry and choose **Remove a step**.

When you finish entering the sequence of register operations, choose **Next** to continue.

Step 2: Acknowledging an Interrupt from Your PXI Device

In addition to the interrupt detection sequence, NI-VISA also needs the sequence of register operations required to acknowledge an interrupt condition for your device. At interrupt time, if NI-VISA determines that your device is interrupting (via the sequence of accesses specified in Step 1), this sequence should do whatever is necessary to quell the interrupt condition. This sequence is constructed using the same Read, Write, and Compare operations discussed in Step 1, and individual operations are entered in an identical manner.

Because this sequence should consist of the minimum operations necessary to turn off an interrupt condition for your device, the result of any Compare operations, while still valid, are irrelevant to interrupt acknowledgment.

Note ***If your device uses ROAK (Release on Interrupt Acknowledge) interrupts, and the ROAK register was accessed in the sequence specified by Step 1, this sequence can be left blank.***

Example

The hypothetical xyzScope is programmed to generate an interrupt when it finishes acquiring a waveform. The xyzScope defines an interrupt-enable register (INTEN), a generic status register (STATUS), and an interrupt-acknowledge register (INTACK). The xyzScope maps its register set to BAR0. The registers are defined as follows:

- INTEN (8 bit, offset 0x00)—Reading the register results in 0x1 if interrupts are enabled for the device, 0x0 otherwise.
- STATUS (32 bit, offset 0x04)—Writing the register with 0xABCD causes the device to update this register with the current device status. Reading the register returns this status. Suppose that reading a value of 0x0080 indicates a pending interrupt, while 0x0000 means no interrupt is pending.
- INTACK (16-bit, offset 0x08)—Writing the register with a non-zero value acknowledges a pending interrupt condition.

To configure NI-VISA to detect interrupts from the xyzScope, you would enter information into the PXI Setup Wizard as shown in Figures 3 and 4.

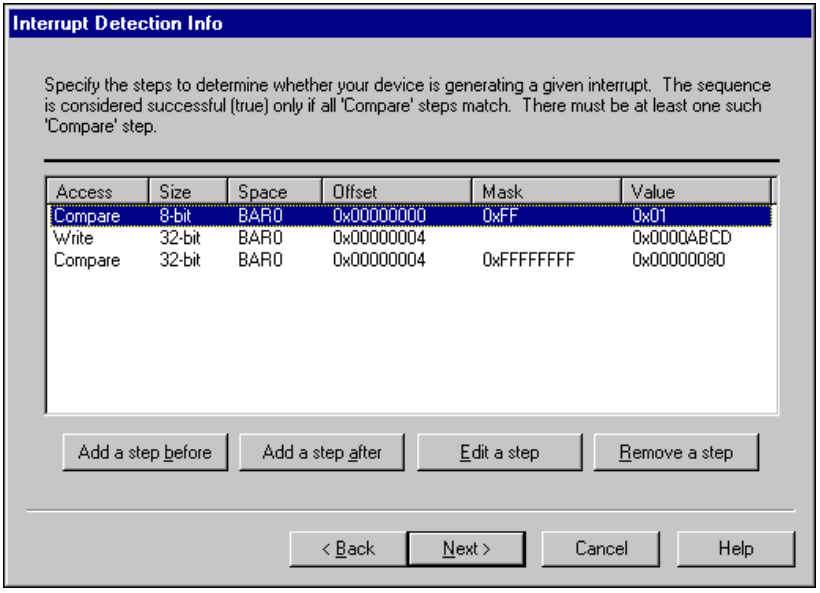

Figure 3. Interrupt Detection Information for xyzScope

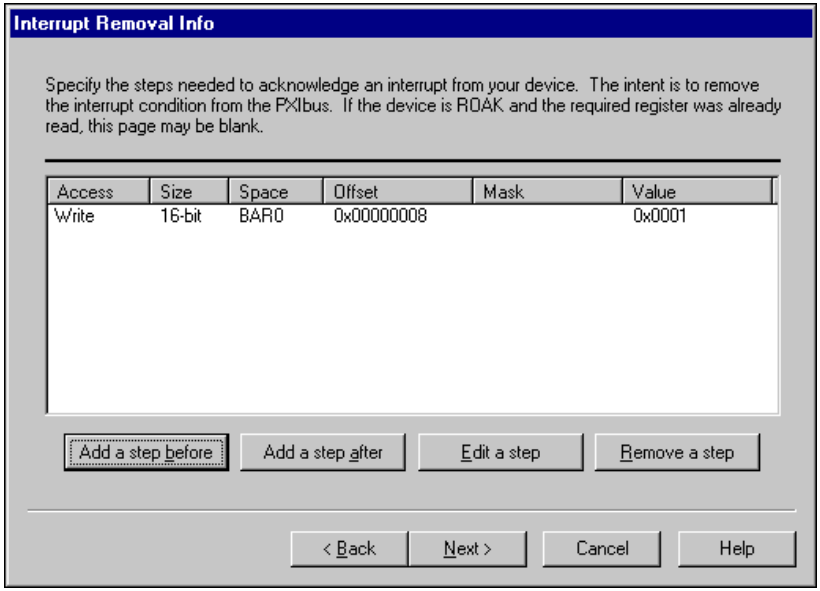

Figure 4. Interrupt Removal Information for xyzScope

In this example, you would choose **Next** to continue when you finished entering the sequence of register operations.

Device Manager Settings

The PXI Setup Wizard allows you to customize how your device should appear under Windows Device Manager on Windows 95/98 systems. This information includes the following:

- Model Name—The name of the device.
- Manufacturer Name—The manufacturer of the device.
- Device Classification—The category of the device in the Windows Device Manager tree view.

These settings are cosmetic and do not affect NI-VISA's ability to recognize and control your PXI instrument. They are provided as a convenience, allowing you to more fully customize your instrument driver package.

Refer to Figures 5 and 6 for an example of how these Device Manager settings would appear on your system.

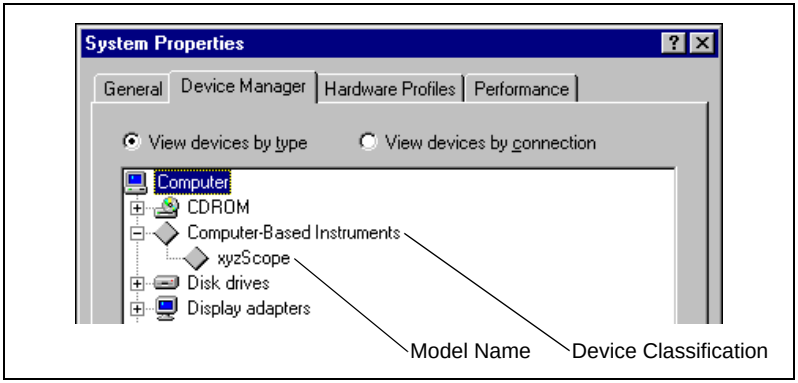

Figure 5. Model Name and Device Classification

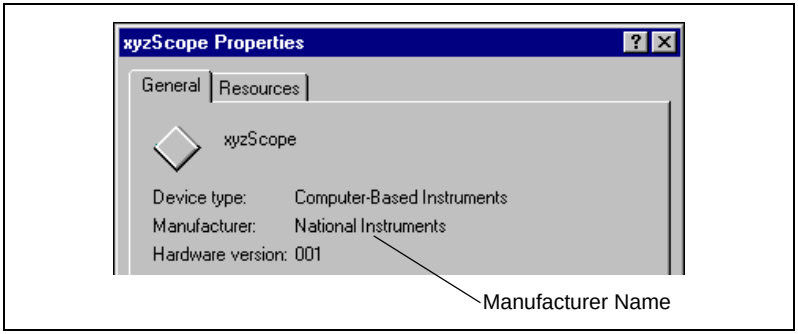

Figure 6. Manufacturer Name

When you finish entering the Device Manager settings, choose **Next** to continue.

Complete the PXI Device Registration

To make your PXI device available for use, NI-VISA must store the previously collected information in the system registry. You use the PXI Setup Wizard to specify a Windows Setup Information (.inf) file the operating system can use to configure NI-VISA for use with your device. You can distribute this file with any PXI application developed with NI-VISA. Refer to the `README.TXT` file for information on how to incorporate the .inf files into your installation package.

After choosing the format and storage location appropriate for your application, choose **Finish**.

☞ **Note** ***NI-VISA cannot locate your PXI device until you have updated your system registry using the .inf file.***

Using PXI Resources in your VISA Application

When you finish configuring NI-VISA for use with your PXI device, you are ready to begin application development. This section describes PXI-specific additions to NI-VISA.

☞ **Note** ***To make use of PXI functionality in your application, you must define the macro NIVISA_PXI before including the*** `visa.h` ***header file.***

Resource Manager Operations

PXI supports a single resource type: INSTR. The following VISA operations apply when using PXI:

- `viFindRsrc()`

  Basic Operation: Use either of the following strings to locate your PXI devices:

  - `?*INSTR`
  - `PXI?*INSTR`

  Advanced Operation: Beginning with VISA 2.0, you can use complex queries to find instruments matching a more precise specification. For example, you can use the following description string to find all National Instruments PXI devices:

  ```
  PXI?*INSTR{VI_ATTR_MANF_ID == 0x1093}
  ```

  Refer to the viFindRsrc topic in the NI-VISA help file for more information.

- `viOpen()`

  Resource strings assume the form:

  ```
  PXI[bus]::deviceNumber[::functionNumber][::INSTR]
  ```

- `viGetAttribute()`, `viSetAttribute()`

The following attributes are specific to PXI:

Attribute	Description
VI_ATTR_PXI_DEV_NUM	Device number (same as in viOpen)
VI_ATTR_PXI_FUNC_NUM	Function number (same as in viOpen)
VI_ATTR_PXI_SUB_MANF_ID	Subsystem manufacturer ID
VI_ATTR_PXI_SUB_MODEL_CODE	Subsystem model code
VI_ATTR_PXI_MEM_TYPE_BAR0 through VI_ATTR_PXI_MEM_TYPE_BAR5	Memory type used in the specified BAR. Valid values include: VI_PXI_ADDR_NONE VI_PXI_ADDR_MEM VI_PXI_ADDR_IO
VI_ATTR_PXI_MEM_BASE_BAR0 through VI_ATTR_PXI_MEM_BASE_BAR5	Memory base address assigned to the specified BAR (if applicable).
VI_ATTR_PXI_MEM_SIZE_BAR0 through VI_ATTR_PXI_MEM_SIZE_BAR5	Memory size used by the device in the specified BAR (if applicable).

The following attributes are common among all device classes but have special meaning in PXI:

Attribute	Description
VI_ATTR_MANF_ID	The PCI vendor ID for this device.
VI_ATTR_MODEL_CODE	The PCI device ID for this device.
VI_ATTR_INTF_TYPE	Always has a value of VI_INTF_PXI for PXI instruments.
VI_ATTR_INTF_NUM	The bus number for the device, as specified in viOpen().

The following attributes are common to all device classes. Refer to the *NI-VISA Programmer Reference Manual* for more information on these attributes:

- VI_ATTR_TMO_VALUE
- VI_ATTR_RSRC_NAME
- VI_ATTR_RSRC_SPEC_VERSION
- VI_ATTR_RSRC_IMPL_VERSION
- VI_ATTR_RSRC_MANF_NAME
- VI_ATTR_INTF_INST_NAME
- VI_ATTR_INTF_NUM
- VI_ATTR_WIN_BASE_ADDR
- VI_ATTR_WIN_SIZE
- VI_ATTR_WIN_ACCESS

Event Operations

The standard VISA event operations support the following event types for PXI:

- VI_EVENT_IO_COMPLETION—for asynchronous block moves
- VI_EVENT_PXI_INTR—for PXI hardware interrupt handling

Low-Level and High-Level Register Operations

The standard VISA register operations use the following address spaces for PXI:

- VI_PXI_BAR0_SPACE through VI_PXI_BAR5_SPACE—space parameter for memory or I/O cycles
- VI_PXI_CFG_SPACE—for configuration cycles

321217E-01 Nov98