

LabVIEW & Instrumentation Training

By
Uvais A. Qidwai, Ph.D.
Computer Science & Engineering Dept.
Qatar University, Doha, Qatar.

July 24th, 2006
@ NED University of Engineering & Technology,
Karachi, Pakistan.

NI-VISA Overview

- NI-VISA = National Instruments - Virtual Instrument Software Architecture API.
- You can use NI-VISA to communicate with most instrumentation buses including GPIB, USB, Serial, and Ethernet.
- It provides a consistent and easy to use command set to communicate with a variety of instruments.

History

- For years industry has moved towards purchasing instrumentation from a variety of vendors.
- This allows engineers to choose the best instruments for their application without being constrained to a specific vendor.
- This caused the invention of hardware standards like [SCPI](#) (Standard Commands for Programmable Instrumentation) that were designed to allow interoperability between instruments of different vendors.

- Even with these standards in place, it was still a difficult task to build a test system with instruments from different vendors.
- In 1993 National Instruments, along with several companies including GenRad, Racal Instruments, and Tektronix, formed the VXIplug&play Systems Alliance.
- VXI (VME {Versa Module Europa} Extension for Instrumentation) was the standard at the time for modular instrumentation.
- The goal of the alliance was to ensure multi-vendor interoperability for VXI systems and to reduce the development time for a fully working system including multi-vendor instrumentation.
- VISA was developed through this alliance with hopes of increasing productivity through a decrease in system setup time.

Advantages of VISA

- One of VISA's advantages is that it uses many of 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 USB.
- Thus, VISA provides interface independence.
- This can make it easy to switch interfaces and also gives the users who must program instruments for different interfaces a single language they can learn.

- VISA is also designed so that programs written using VISA function calls are easily portable from one platform to another.
- VISA does this by defining its own data types.
- This prevents problems like, for example, possible problems caused by moving from one platform to another where the size of an integer may be different.
- In other words, a LabVIEW application written with VISA calls can be easily ported to another platform that supports LabVIEW.
- Several operating systems are supported, including Windows XP/64, Windows 2000/ME/98, MAC OS X/9/8, some Linux distributions, as well as some Solaris distributions.

- VISA's greatest advantage is that it is an extremely easy language to learn.
- VISA provides a very simple-to-use API that has bus independent functions for most if its I/O functionality.
- VISA provides the most commonly used functionality for instrumentation in a very compact command set, eliminating the need to learn low level communication protocols for multiple bus types.

VISA Terminology

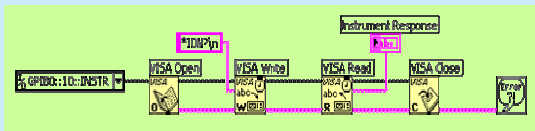
- Before you start using NI-VISA it helps to learn a little terminology commonly used throughout the development process.
- The most important objects in the VISA language are known as **resources**.
- A VISA Resource is any instrument in your system (this includes serial and parallel ports).
- If you have multiple resources connected to one GPIB controller, each of those instruments is considered a VISA Resource.

- An Instrument **Descriptor** is the exact name of a resource.
- It specifies the interface type (GPIB, serial, USB), the address of the device (logical address or primary address) and the VISA Session type (INSTR, Event, or INTFC).
- A VISA **Session** is a path of communication to a VISA Resource, so you must open a VISA Session any time you want to do VISA communication to an instrument.
- A VISA **Alias** is basically a nickname for a VISA Resource. You could give an instrument located on [GPIB0::3::INSTR] a visa alias of "Function Generator". Then, in your application you could make calls to "Function Generator" instead of having to use the instrument descriptor.

A Typical VISA Application

- A typical VISA application would go through the following steps:
 - Open a Session to a given Resource.
 - Do any configuration on the given resource (setting baud rates, termination character, etc...).
 - Perform writes and reads to the device.
 - Close the Session to the Resource.
 - Handle any errors that may have occurred.

- The following is a LabVIEW application that opens a session to a GPIB Instrument, performs a write of `"*IDN?\n"` and then queries the device for its response.



- This exact same format would be used in a text based language like C++ or Basic.
- You would also follow this exact same format if the instrument was Serial, USB, Ethernet, IEEE-1394, or any of the other buses VISA supports.
- All you would have to change is the Instrument Descriptor connected to the VISA Open.
- This code would run on any operating system that supports LabVIEW and NI-VISA.

Using NI-VISA to Control Your USB Device

- Universal Serial Bus (USB) is a message-based communication bus.
- This means a PC and a USB device communicate by sending commands and data over the bus as text or binary data.
- Each USB device has its own command set.
- You can use NI-VISA Read and Write functions to send these commands to an instrument and read the response from an instrument.
- Check with your instrument manufacturer for a list of valid commands for your instrument.
- Starting with version 3.0, NI-VISA supports USB communication.
- Two classes of VISA resources are supported: USB INSTR and USB RAW.

- The USB INSTR resource class is used by USB devices that conform to the USB Test and Measurement Class (USBTMC) protocol.
- USBTMC devices conform to a protocol that the VISA USB INSTR resource class can understand. No configuration is necessary to communicate with a USBTMC device.
- USB RAW instruments are any USB instrument other than those instruments that specifically conform to the USBTMC specification.
- Contact your instrument manufacturer for details about the communication protocol and the command set your instrument uses.

Configuring NI-VISA to Control Your USB Device

- This section walks through the steps for configuring a USB RAW device to be controlled by NI-VISA 3.0 on a Windows-based computer.
- At this point, NI-VISA already should be installed on your computer, and your USB device should not be connected.
- Furthermore, you should have a driver for your USB device installed.
- There are three steps to configure your USB device to use NI-VISA:
 - Create the INF file using the Driver Development Wizard.
 - Install the INF file and the USB device using the INF file.
 - Test the device with NI-VISA Interactive Control.

Create the INF File Using the Driver Development Wizard

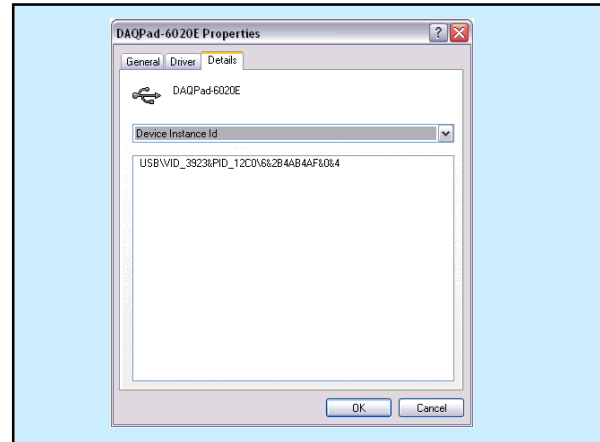
- To use NI-VISA, you must first tell Windows to use NI-VISA as default driver for the device.
- In the Windows environment, you can do this with an INF file. NI-VISA 3.0 and higher includes the VISA Driver Development Wizard (DDW) to create an INF file for your USB device.

To open the DDW, select **Start»Programs»National Instruments»VISA»VISA Driver Developer Wizard**.

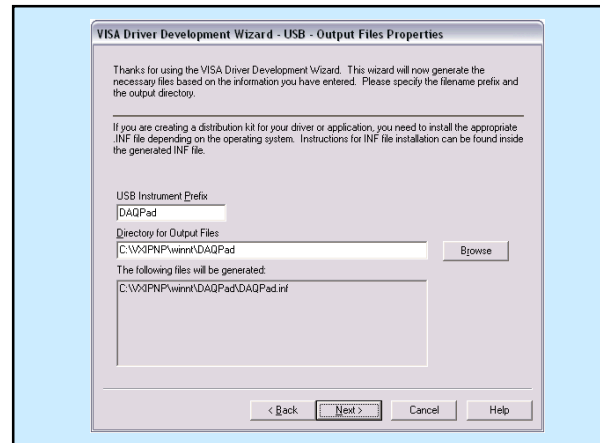


- You must know the USB vendor ID and product ID for your USB instrument. These numbers identify your USB device when you install it and address your device when you want to communicate with it.
- According to the USB specification, both numbers are 16-bit hexadecimal numbers and should be provided by the device manufacturer.
- If you do not know the USB vendor ID and product ID, to get them, plug the device into the computer and allow the computer to recognize the new device.
- Cancel out of the Found New Hardware Wizard if it starts.
- Open the Device Manager from the Control Panel and find your device on the list, usually under "Other Devices".

- It may show a yellow exclamation mark indicating it is an unknown device.
- Double click the device to open the properties.
- Select the Details tab and ensure that "Device Instance Id" shows in the attribute dropdown box.
- A string of characters will be displayed.
- The four characters to the right of "VID_" and "PID_" are your vendor ID and product ID, respectively.
- Write down the characters for your device, close the Device Manager, and unplug the device from the computer.



- Alternatively, you can contact your device vendor to obtain this information
- Before proceeding with the Driver Development Wizard, make sure the device has been removed from the computer.
- Enter the vendor ID, product ID, manufacturer name, and model name for your device in their respective fields.
- Click Next. The Output Files Properties window is displayed



- The USB Instrument Prefix is simply a descriptor you use to identify the files used for this device.
- Enter a USB instrument prefix, select the desired directory in which to place these files, and click **Next**.
- The next window will provide you installation options.
- The default selection is to install the setup information for the operating system and is usually the best option.
- Once you select an option, click **Finish** to exit the wizard.
- The INF file is created in the directory you specified in the output file directory field in the previous window.

Install the INF files and the USB device

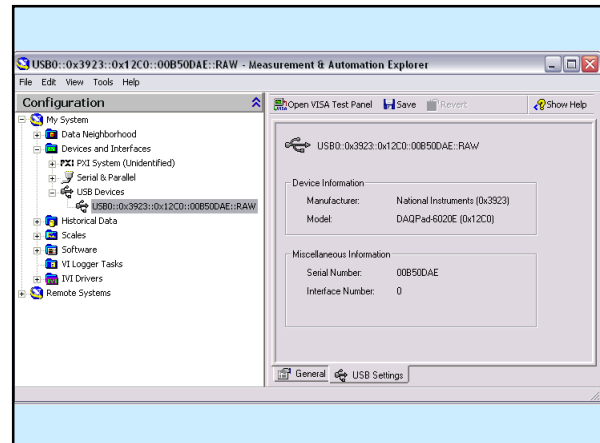
- The installation of the INF files is different for each version of Windows.
- When the DDW creates an INF file, installation instructions are included in a header at the top of the INF file.
- Because INF files are ASCII text files, they can be read in any text editor such as Notepad.
- For detailed information about installing your INF file, open your INF file in a text editor and follow the instructions at the top of the file.

- Copy the INF file to the INF folder.
 - On Windows XP, this folder is usually at **C:\WINDOWS\INF**.
 - This folder may be hidden, so you may need to change your folder options to view hidden files.
- Right-click on the INF file in **C:\WINDOWS\INF** and click **Install**.
 - This process creates a PNF file for your device.
 - You are now ready to install your USB device.

- Connect your USB device.
 - Because USB is hot pluggable, Windows should be able to detect your USB device, and the Add New Hardware Wizard should open automatically as soon as you connect it to the USB port.
 - Follow the onscreen instructions for the wizard.
 - When you are prompted to select a driver for this device, browse to the INF folder and select the INF file you created using the DDW.

Test Communication with VISA Interactive Control

- Open Measurement & Automation Explorer.
- Select **Tools»Refresh** to refresh the view.
- Your USB device should be listed as a USB Device under **Devices and Interfaces**.
- Your USB device is now installed and configured to use NI-VISA.
- If you select your USB device, the device information is displayed in the USB Settings window.
- Using this window, you can access information such as the manufacturer ID, model code, and serial number for your device.



VISA for Serial Port

- Use the VIs and functions located on the Functions>>All Functions>>Instrument I/O>>Serial palette for serial port communication.
- The VISA Write and VISA Read functions work with any type of instrument communication and are the same whether you are doing GPIB or serial communication.
- However, because serial communication requires you to configure extra parameters, you must start the serial port communication with the VISA Configure Serial Port VI.

- The VISA Configure Serial Port VI initializes the port identified by VISA resource name to the specified settings. timeout sets the timeout value for the serial communication. baud rate, data bits, parity, and flow control specify those specific serial port parameters.
- The error in and error out clusters maintain the error conditions for this VI.

VI Example

- Following example shows how to send the identification query command *IDN? to the instrument connected to the COM2 serial port.
- The VISA Configure Serial Port VI opens communication with COM2 and sets it to 9,600 baud, eight data bits, odd parity, one stop bit, and XON/XOFF software handshaking.
- Then the VISA Write function sends the command. The VISA Read function reads back up to 200 bytes into the read buffer, and the Simple Error Handler VI checks the error condition.

