Document Type: Tutorial
NI Supported: Yes
Publish Date: Apr 11, 2012

Long-Term Instrument Control and Connectivity Solutions

Table of Contents

Overview

The key to developing long-term instrument control applications is flexible and scalable software and hardware that enable seamless connectivity to any instrument in your application and offer built-in capabilities for advanced data analysis and presentation. Your choice of instrument should not limit or control your software choice, and your software preference should not limit the instrument that you pick for your application. The two decisions, the instrument and the software, should be independent of each other, empowering you to select the most optimal instrument and advantageous software environment for your application.

To attain flexible instrument control and connectivity, you need:

- Seamless industry-standard software integration
- Breadth of hardware connectivity options
- Comprehensive instrument driver coverage

This white paper details industry-standard options available to deliver these benefits, helping you choose the right software and hardware solutions for your application.

Key Characteristics of Instrument Control Software

There are several critical characteristics to look for as you evaluate a software tool to use for application development. Making the right decision about your application development environment (ADE) helps ensure your project's success by delivering the tools needed to develop higher quality products faster.

Key characteristics of instrument control software:
1. Complete ADE versus a limited-functionality executable
2. Built-in instrument connectivity functions
3. Analysis and presentation capabilities

1. Complete ADE versus a limited functionality executable
A complete application development environment provides considerable advantages over a preprogrammed software tool. Complete ADEs deliver flexibility and power for customization, application integration, system connectivity, and more. That flexibility combined with measurement- and automation-specific tools delivers the fundamentals of virtual instrumentation. Virtual instrumentation represents a fundamental shift from traditional hardware-centered instrumentation systems and defined software packages to flexible, measurement- and automation-software-centered systems that exploit the computing power, productivity, display, and connectivity capabilities of popular desktop computers and workstations. Although the PC and integrated circuit technology have experienced significant advances, it is software that truly provides the leverage to build on this powerful hardware foundation to create virtual instruments, providing better ways to innovate and significantly reduce cost. empowering engineers and scientists to build measurement and automation systems that suit their needs exactly (user-defined) instead of being limited by traditional fixed-function instruments and a software executable (vendor-defined) limited in functionality.

2. Built-in instrument connectivity functions
Another key to ensure project success is the out-of-the-box capabilities of the software package. To meet demanding project deadlines, you need software that does some of the work for you. Built-in instrument connectivity functions save you significant development time. There is a considerable difference between calling one native query command to communicate with your instrument and declaring the function through a DLLImport command, instructing the compiler to marshal the parameters, transferring control, throwing an exception to the managed caller, and then calling the query command.

3. Analysis and presentation capabilities
Finally, you should ensure that your software includes strong analysis and presentation capabilities. Once you have acquired the data from your instrument, you need built-in powerful algorithms and functions designed specifically for measurement analysis and signal processing. If the software package you choose does not contain measurement analysis functions, you are forced to write your own algorithms to turn raw data into critical information. With proper software capabilities, you can extract information from acquired data and unique measurements; generate, modify, process, and analyze signals; add intelligence and decision-making capabilities to your applications; perform inline and offline analysis; and then present information in a professional user interface designed for measurement data or to your data management systems.

Let's look at how these three features are presented in the industry standard development environments: NI LabVIEW, LabWindows™/CVI, and Microsoft Visual Studio.

LabVIEW

LabVIEW is a complete graphical development environment and the industry leader in instrument control software. It is a highly interactive and open environment for the rapid prototyping and incremental development of applications, from measurement and automation to real-time embedded to hardware design. LabVIEW delivers all of the flexible benefits of traditional development environments in addition to revolutionary, rapid development capabilities through graphical programming. With LabVIEW, you can design custom virtual instruments by creating a graphical user interface through which you:

- Operate the instrumentation program
- Control selected hardware
- Analyze acquired data

▪ Display results

You can customize front panels with knobs, buttons, dials, and graphs to emulate control panels of traditional instruments, create custom test panels, or visually represent the control and operation of processes. The similarity between standard flow charts and LabVIEW graphical programming shortens the learning curve associated with traditional, text-based languages.

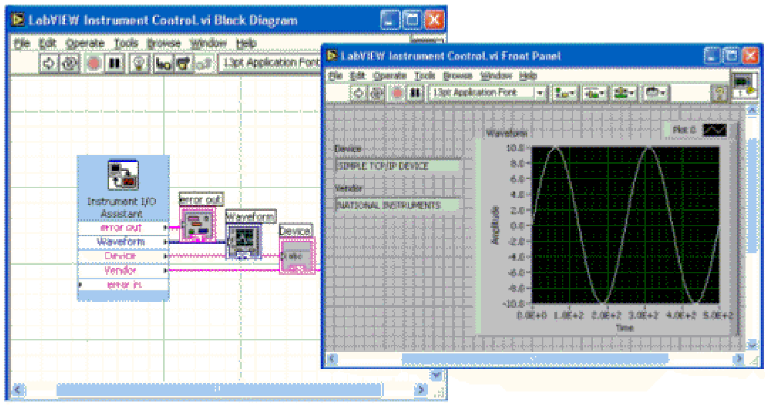

Figure 1. LabVIEW Graphical Programming Block Diagram and User Interface

You determine the behavior of the virtual instruments by connecting icons together to create block diagrams, which are natural design notations for scientists and engineers. With graphical programming, you can develop systems more rapidly than with conventional programming languages, while retaining the power and flexibility needed to create a variety of applications.

Software	Complete ADE	Built-in instrument connectivity functions	Built-in analysis and presentation capabilities
LabVIEW	Yes	Yes	Yes

LabWindows/CVI

LabWindows/CVI is also a complete development environment, but one based on ANSI C as opposed to graphical development. In the same way that LabVIEW offers the flexibility delivered by a measurement and automation-specific environment, so does LabWindows/CVI. LabWindows/CVI streamlines ANSI C development with hardware configuration assistants, comprehensive debugging tools, and interactive execution to run functions at design time. Built-in measurement libraries enable rapid development of complex instrument applications including direct bus support, analysis functions, and user interface controls.

Software	Complete ADE	Built-in instrument connectivity functions	Built-in analysis and presentation capabilities
LabWindows/CVI	Yes	Yes	Yes

Microsoft Visual Studio

Microsoft Visual Studio, including Visual C# .NET, Visual Basic .NET, Visual Basic 6.0, and Visual C++ is also a complete development environment. However, unlike LabVIEW and LabWindows/CVI, the Microsoft Visual Studio environment is a general-purpose tool not designed or optimized for measurement and automation application development. Therefore, it does not contain built-in tools for measurement and automation-specific tasks including instrument control, measurement data analysis, or measurement data presentation. This is where NI Measurement Studio add-in tools help you. Measurement Studio takes the general-purpose Visual Studio environment and integrates native instrument control interfaces, advanced analysis routines, and native user interface controls specifically designed to display measurement data.

Software	Complete ADE	Built-in instrument connectivity functions	Built-in analysis and presentation capabilities
Visual Studio	Yes	No	No
Visual Studio with Measurement Studio	Yes	Yes	Yes

Once you have chosen the appropriate software environment, you then need to determine the I/O bus and I/O connectivity that works best for your measurement.

Hardware Breadth

There are several hardware options for communicating with and controlling your instruments. Historically, most users have controlled their instruments through mature buses, namely GPIB and Serial (RS232). However some emerging commercial computer buses such as Ethernet and USB are appearing in the test and measurement arena. Combining existing instruments based on original communication buses such as GPIB and Serial with the latest instruments that rely on Ethernet and USB interfaces results in systems that are commonly referred to as mixed I/O test systems. To effectively develop and maintain mixed I/O systems, the software plays an important role. You need software tools that can support a wide variety of bus protocols in addition to scalable hardware solutions such as the options listed below.

GPIB

GPIB has been one of the most common I/O interfaces present in instruments for many years. GPIB was designed specifically for instrument control applications. The IEEE 488 specifications, which standardized this bus, define the electrical, mechanical, and functional specifications of the bus while also defining its basic software communication rules. GPIB is a digital 8-bit parallel communications interface capable of achieving data transfers of up to 8 MB/s, while supporting the connection of 14 instruments in addition to a system controller. GPIB provides robust shielded connectors that are designed for rugged industrial environments.

NI GPIB hardware comes complete with the NI-488.2 driver, which includes native interfaces for LabVIEW, LabWindows/CVI, Visual Basic 6.0, Visual Basic .NET, Visual C# .NET, and Visual C++. In addition, you can also communicate with GPIB hardware through VISA. VISA is a standard architecture originally defined by the VXI*plug&play* Systems Alliance, founded in 1993, to provide a common foundation for the development, delivery, and interoperability of high-level, multivendor software system components including support for GPIB, VXI, PXI, VME, Serial, and/or USB interface. VISA offers native interfaces for LabVIEW, LabWindows/CVI, Visual Basic 6.0, Visual Basic .NET, Visual C# .NET, and Visual C++.

Serial (RS232)

RS232 is a specification for serial communication and is popular in analytical and scientific instruments. It is also commonly used to control modems and printers. Unlike GPIB, with the RS232 interface, you can connect and control only one device at a time. RS232 is also a relatively slow interface, with typical data rates of less than 20 KB/s.
NI serial hardware is packaged with the NI-Serial driver. You access the NI-Serial driver through the Windows Serial API or the NI-VISA API with native interfaces for LabVIEW, LabWindows/CVI, Visual Basic 6.0, Visual Basic .NET, Visual C# .NET, and Visual C++.

Ethernet

Recently, instrument vendors have begun to include Ethernet/LXI as an alternative communication interface on stand-alone instruments. Although new to instrument control applications, Ethernet is a mature technology that is widely used for measurement systems – albeit in other capacities. Advantages of Ethernet for instrument control applications include remote control of instruments, sharing among different users at different locations, and easier data integration and publication. Most common Ethernet networks today can transfer data at 10 Mb/s, 100 Mb/s, or even 1 Gb/s with the new Gigabit Ethernet standard. However, these are theoretical maximum transfer rates and applications most likely will not achieve the same performance. In addition, data transfers across Ethernet are not deterministic and extra security measures, such as firewalls, should be taken to ensure data integrity.

You can control and communicate with Ethernet instruments through the VISA interface natively in LabVIEW, LabWindows/CVI, Visual Basic 6.0, Visual Basic .NET, Visual C# .NET, and Visual C++.

USB

USB was designed primarily to connect peripheral devices, such as keyboards and mice, to PCs. The USB is a plug-and-play bus, supports up to 127 devices on one port, and has a theoretical maximum throughput of 480 Mb/s (high-speed USB defined by the USB 2.0 specification). Since USB ports come standard on today's PCs, USB is viewed as a natural evolution of conventional serial port technology. Some of the bus' features may prevent it from achieving widespread appeal. These include the fact that cables are not industrially graded, are sensitive to noise, and can accidentally become detached, and the maximum distance between the controller and the device is 30 m.

You can control and communicate with Ethernet instruments through the VISA interface natively in LabVIEW, LabWindows/CVI, Visual Basic 6.0, Visual Basic .NET, Visual C# .NET, and Visual C++.

IEEE 1394

IEEE 1394, is a high-performance serial bus originally developed by Apple computer in the early 1990s. The baseline standard handles throughput rates of up to 400 Mb/s. The IEEE 1394 Trade Association is revising the specification to increase this transfer rate to 3.2 Gb/s. While devices must be within 4.5 m of the bus socket to conform to the specification, up to 16 devices can be daisy chained for a maximum run of 72 m. IEEE 1394, however, is less than ideal for test and measurement applications. For example, the 1394 cables are not industrial graded, the isochronous mode allows this bus to guarantee bandwidth but does not guarantee that no data will be lost, and application testing shows 1394 data transfers to be much lower than their theoretical values.

You can control and communicate with 1394-based instruments through the drivers that ship with the device as a DLL interface to your software development environment including LabVIEW and LabWindows/CVI.

Wireless

Wireless Ethernet (IEEE 802.11) is another emerging standard for I/O buses that can be used in instrument control applications. While it is capable of slower transfer rates than standard Ethernet and suffers from some of the same security issues, it offers the convenience of remote control without the necessity for cabling. Another potential wireless bus alternative that is not discussed here is Bluetooth.

You can control and communicate with IEEE 802.11 instruments through the drivers that ship with the device as a DLL interface to your software development environment including LabVIEW and LabWindows/CVI.

Bus Bridges

Although these buses may offer some advantages for instrument communication, they do have some of the drawbacks mentioned above and there is no clear choice of which bus, if any, will gain the widespread acceptance of GPIB. As a result, instrument manufacturers have been slow to make these buses available on their products. A viable alternative to native bus communication is through bus bridge products available on the market. With these bridges, you can convert from one bus type to another (for example, convert from USB or Ethernet to GPIB or Serial).

Bus converters offer some great advantages for instrument control. They allow you to take advantage of some of the advanced capabilities of these new buses on your computer, while maintaining your investment in your programs. Good bridges should be software transparent allowing you to use unchanged applications written for plug-in boards of the same bus (for example, a USB to GPIB converter and plug-in GPIB board).

You can control and communicate with bus bridges either through the NI-VISA API or the NI-488.2 interface natively in LabVIEW, LabWindows/CVI, Visual Basic 6.0, Visual Basic .NET, Visual C# .NET, and Visual C++.

Bus I/O Software

Bus bridges help you ease your way into using new bus technology. However, the future holds the promise of test systems that integrate mature buses, some of the newer buses discussed earlier, as well as others that may come into play in the future. The key to integrating all of these buses seamlessly into one system is through the use of good software architectures and powerful software environments.

Instrument Driver Applicability

There are several different ways to control your instruments – you can either use an instrument driver or control the instrument through direct I/O commands (see Figure 2).

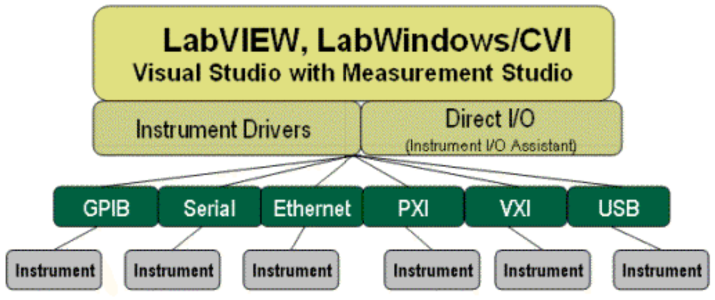

Figure 2. Instrument Drivers and Interactive, Direct I/O Tools for Instrument Control

In general, NI recommends the use of an instrument driver if one is available because of the clear time-saving benefits. NI recognized early the importance of instrument drivers, and in 1990 created the now industry-leading instrument driver repository: The Instrument Driver Network (*ni.com/idnet*). This online resource currently provides more than 2,200 instrument drivers from more than 150 different instrument vendors.

An instrument driver is a VI, library, or class that contains high-level functions. With it, you can control a specific instrument or family of instruments. If you cannot find an instrument driver for your particular instrument or if you do not need to use an instrument driver, NI software offers several interactive tools to facilitate direct I/O control and communication as well.

Some reasons why you would not use an instrument driver include the following:

- You only need to send a few commands to your instrument
- An instrument driver does not exist
- You do not need to distribute a set of instrument commands to other developers

Instrument Drivers

Instrument drivers are the key to rapidly developing test and measurement applications. By providing high-level and modular VIs, libraries, and classes for easy programming, they eliminate the need for you to learn complex programming protocols. An instrument driver is a set of software routines that correspond to a programmatic operation such as configuring, reading from, writing to, and triggering the instrument.

National Instruments provides instrument drivers for a wide variety of instruments; these instrument drivers are written in LabVIEW and/or LabWindows/CVI and use the VISA API for instrument control.

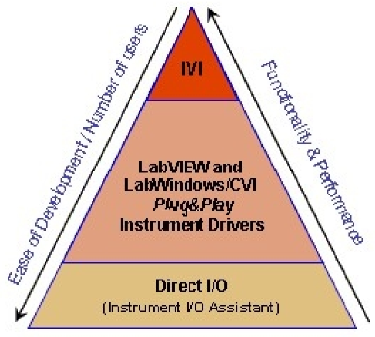
IVI
LabVIEW and LabWindows/CVI *Plug&Play* Instrument Drivers
Direct I/O
(Instrument I/O Assistant)
Ease of Development / Number of users
Functionality & Performance

Figure 3. A variety of instrument drivers provide different levels of functionality and complexity.

Because you have different needs for different test applications, there are two different types of instrument drivers to meet your requirements: Plug and Play drivers and IVI drivers.

Additional Resources
Instrument Driver Network

Plug and Play

Plug and Play instrument drivers simplify controlling and communicating with your instrument through a standard and simple programming model for all drivers. Plug and Play instrument drivers provide native source code for LabVIEW and LabWindows/CVI.

LabVIEW Plug and Play Drivers

A LabVIEW Plug and Play instrument driver is a set of VIs used to control and communicate with a programmable instrument. Each VI corresponds to a programmatic operation such as configuring, reading from, writing to, and triggering the instrument. LabVIEW Plug and Play instrument drivers comply with requirements and recommendations for programming style, error handling, front panels, block diagrams, icons, and online help. Because LabVIEW Plug and Play drivers maintain a common architecture and interface, you quickly and easily connect to and communicate with your instruments with very little or no code development. Moreover, with the standard programming model of LabVIEW Plug and Play instrument drivers, you can easily add instruments to your test system without worrying about learning new communication protocols or spending time understanding new programming paradigms.

In addition, LabVIEW instrument drivers that meet National Instruments requirements and recommendations do not rely on constantly changing technology such as Microsoft technologies. Instead, these instrument drivers are long-term solutions delivering more than a decade of compatibility. With LabVIEW instrument drivers, you can use an instrument driver written in 1992 in your test application today with little or no code modification.

LabWindows/CVI Plug and Play Drivers

A LabWindows/CVI Plug and Play instrument driver is a set of ANSI C software routines used to control and communicate with a programmable instrument. Each routine corresponds to a programmatic operation such as configuring, reading from, writing to, and triggering the instrument. LabWindows/CVI Plug and Play instrument drivers comply with requirements and recommendations for programming style, error handling, function panels, function trees, and help files. Because LabWindows/CVI Plug and Play drivers maintain a common architecture and interface, you quickly and easily connect to and communicate with your instruments with very little code development. Moreover, with the standard programming model of LabWindows/CVI Plug and Play instrument drivers, you can easily add instruments to your test system without worrying about learning new communication protocols or spending time understanding new programming paradigms. As with LabVIEW, LabWindows/CVI Plug and Play instrument drivers deliver long-term compatibility.

Interchangeable Virtual Instrument (IVI)

Interchangeable virtual instruments (IVI) drivers are more sophisticated instrument drivers that feature increased performance and flexibility for more complex test applications that require interchangeability, state-caching, or simulation of instruments. IVI drivers are based on industry standards developed by the IVI Foundation. The IVI Foundation was founded to create an instrument driver standard that builds on the VXI*plug&play* standards but also provides features such as instrument interchangeability, simulation, state-caching, and multithread safety. The IVI Foundation comprises user companies such as Boeing, integrators such as BAE Systems, and instrument and software vendors such as National Instruments, Agilent Technologies, and Tektronix. The IVI Foundation was founded in 1998, incorporated in March of 2001, and has an active membership of 27 companies. IVI driver capabilities include state-caching and multithreading to deliver higher performance simulation to increase productivity, and instrument interchangeability to maximize software reuse across similar instrument types.

National Instruments IVI drivers deliver a dramatic performance improvement by integrating a state-caching engine that only sends commands to the instrument necessary to incrementally change its state.

The IVI Foundation defines two architectures for IVI drivers, one based on ANSI-C and the other on Microsoft Component Object Model (COM) technology. Both architectures are designed to coexist and are not mutually exclusive. IVI-C drivers and IVI-COM drivers can be used in the same application.

Although the technologies underlying IVI-C and IVI-COM are different, the implementation technology by itself should not be your primary concern. Instead, focus on two key issues: (1) the *longevity* of the architecture on which the instrument driver is based, and (2) the *usability* of instrument drivers in your ADE.

Architectural Longevity

The issue of architectural longevity is particularly important to users of IVI drivers. Interchangeability is one of the most beneficial features of IVI. A primary reason for achieving interchangeability is to make it easier to replace instruments in systems that must last 10 to 20 years. Having a common API for instrument drivers is not sufficient if the architecture on which those instrument drivers are based changes every few years.

For this reason, National Instrument prefers to adhere to the IVI-C architecture. ANSI-C is a long established standard that is available on all platforms, and is expected to remain so. Conversely, COM is not available on all platforms and has already been superseded by .NET, which was introduced by Microsoft in 2002.

Usability

A key motivation behind IVI-COM drivers is the desire to develop one driver that works automatically in all development environments. Unfortunately, this comes at the expense of usability. IVI-COM drivers present an interface style that is optimal only in Microsoft Visual Basic 6.0, which was superseded by Microsoft Visual Basic .NET.

Ideally, instrument drivers should present an optimal interface for each development environment in which they are used. For example, instrument drivers should be presented as a set of LabVIEW VIs for use in LabVIEW, a set of C++ classes for use in Microsoft Visual C++, and a set of .NET classes for use in Microsoft Visual Basic.NET and Visual C#.

National Instruments follows this model in implementing IVI drivers. NI focuses on providing native support for NI development environments, LabVIEW and LabWindows/CVI. NI also provides support for other environments as they become popular. NI ships its IVI-C drivers with ANSI-C interfaces, LabWindows/CVI function panels, and LabVIEW VI wrappers. For message-based devices, NI includes the ANSI-C source code with the driver. NI also provides C++ wrappers for IVI drivers for Measurement Studio users. In the future, NI also intends to provide .NET assembly wrappers for IVI drivers. As Visual Basic 6.0 becomes obsolete through the growing Visual Basic .NET adoption, National Instruments does not plan to develop COM wrappers for IVI-C drivers.

If you need interchangeability and simulation in your test applications, IVI drivers provide a robust tool to save you significant development time and maintenance costs.

To build an interchangeable test system that can stand the test of time, IVI-C drivers are the best choice. However, if all that is available for your instrument is an IVI-COM or VXI *plug&play* instrument driver, then you should use it. Your goal is to be able to easily communicate with your instruments and not to be tied to one driver technology or another.

Additional Resources
How IVI-C Instrument Driver Technology Enables System Longevity and Platform Portability

Direct I/O Connectivity

If there is not an instrument driver available, you should take advantage of the interactive, direct I/O capabilities built into the software development environments. NI software offers the Instrument I/O Assistant, built-in VISA and bus-specific interfaces, and several debugging tools within Measurement & Automation Explorer (not covered in this document) including NI I/O Trace, Interface Bus Interactive Control (IBIC – for GPIB), and VISA Interactive Control (VISAIC).

Instrument I/O Assistant

The Instrument I/O Assistant provides a user interface to interactively write commands to a device, read data that the device returns, and specify how to parse the response into a format relevant to your application. It is fully integrated into LabVIEW, LabWindows/CVI, Visual Studio .NET through Measurement Studio, and Visual C++ through Measurement Studio. The Instrument I/O Assistant simplifies the challenge of writing instrumentation applications by automatically generating code from your configurations in your environment.

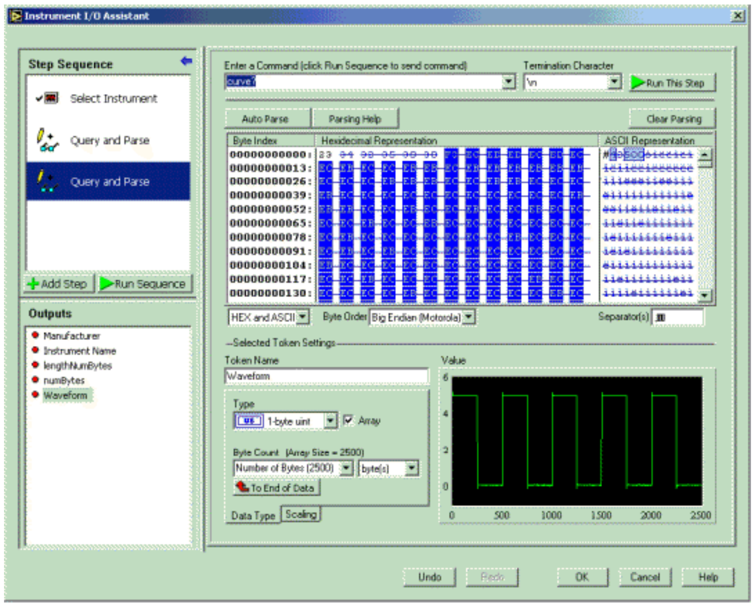

Figure 4. Interactive Instrument I/O Assistant Dialog

The Instrument I/O Assistant also helps you quickly parse complex responses from an instrument for use in native-code instrument drivers. Because string parsing can be complex, using the assistant's interactive window to parse complicated responses can save you time in creating your instrument application.

Built-In Instrument Control and Connectivity Functions

Having built-in support for direct instrument control and connectivity will save you countless hours of low-level development time. Built-in LabVIEW, LabWindows/CVI, and Measurement Studio for Visual Basic 6.0, Visual Basic .NET, Visual C# .NET, and Visual C++ support is provided through native libraries or NI-VISA for GPIB, PXI, Serial, Ethernet, USB, and more.

Additional Resources
LabVIEW Instrument Control Demo
LabWindows/CVI Features

Conclusion

The key to developing long-lasting instrument control applications is flexible and scalable software and hardware that enable seamless connectivity to any instrument in your application as well as built-in capabilities for advanced data analysis and presentation.

NI delivers the most flexible instrument control and connectivity tools by providing the following:

- Optimized industry-standard software development environments and add-ins
- Breadth of hardware connectivity options
- Long-term compatibility
- Comprehensive instrument driver coverage

This empowers you to choose the most optimal instrument and advantageous software for your instrument control applications.

Additional Resources
Instrument Driver Network

The mark LabWindows is used under a license from Microsoft Corporation. Windows is a registered trademark of Microsoft Corporation in the United States and other countries.