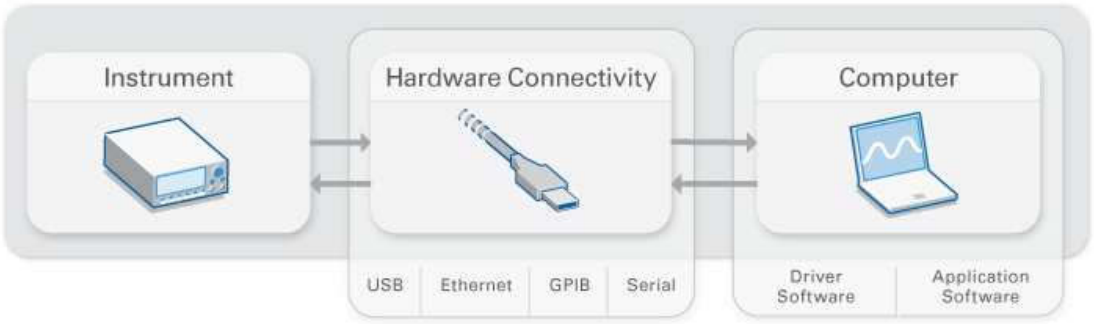
INSTRUMENT CONTROL

Instrument control is a PC-based approach that combines programmable software and hardware connectivity for automating measurement acquisition from third-party instrumentation. An instrument control system consists of instrumentation, connectivity hardware, and a computer with programmable software. Compared to traditional measurement systems, the combination of software, instrument drivers, and connectivity hardware is the most productive, flexible approach to automating third-party instrumentation



What Is an Instrument?

There are two types of instruments: stand-alone and modular. Traditional instrument control applications employ stand-alone (bench top) instruments to gather specific measurement data, but as hardware technologies advance, modular instruments are becoming increasingly popular for acquiring measurement data.

Types of Instrument Control Systems

Stand-Alone

Although primarily designed for manual use, stand-alone instruments can also be incorporated into an automated measurement system. Using instrument control software, one can control the instrument with a PC through a communication bus such as GPIB, USB, serial, or LAN/Ethernet.

Modular

Also now one has the option of creating more flexible and scalable measurement systems by taking advantage of virtual instrumentation. By combining general-purpose, modular hardware with flexible, high-performance software, one can design and develop custom system solutions for measurement needs.

What Is Instrument Control Hardware Connectivity?

Instrument control hardware connectivity is the interface that is used to connect your instrument to PC. One can rely on instrument control hardware solutions to communicate to instruments directly via stand-alone or modular buses such as GPIB, RS232, USB, Ethernet, PCI, and PXI.

Types of Instrument Control Buses

Overview

Instruments usually offer one or more bus options through which you can control them, and PCs usually offer multiple bus options for instrument control. If a PC does not natively have the bus that is on the instrument, one can usually add it as a plug-in board or an external converter. There are many buses for instrument control and they can be divided into these general categories.

Stand-Alone

Stand-Alone instruments are used to communicate with rack-and-stack instruments, these include test and measurement-specific buses such as GPIB and PC-standard buses such as serial (RS232), Ethernet, and USB.

Modular

Modular instruments incorporate the interface bus into the instrument itself. Modular buses include PCI, PCI Express, VXI, and PXI.

What Is a Computer's Role in an Instrument Control System?

A computer with programmable software controls the automation of instruments and processes, visualizes, and stores measurement data. Different types of computers are used in different types of applications. A desktop may be used in a lab for its processing power, a laptop may be used in the field for its portability, or an industrial computer may be used in a manufacturing plant for its ruggedness.

What Are the Different Software Components in an Instrument Control System?

Application Software

Application software facilitates the interaction between the computer and user for automating the process of acquiring, analyzing, and presenting measurement data from instruments. Instrument control software tools come in the form of either a prebuilt application with predefined functionality, or a programming environment for building applications with custom functionality.

Driver Software

Driver software provides application software the ability to interact with an instrument. One can control instruments through direct I/O commands or by using an instrument driver. An instrument driver is a set of software routines that control a programmable instrument. Each routine corresponds to a programmatic operation such as configuring, reading from, writing to, and triggering the instrument. Instrument drivers simplify instrument control and reduce test program development time by eliminating the need to learn the programming protocol for each instrument.

Instrument Control in LabVIEW

Instrument control is accomplished by sending commands and data between the instrument and the PC. With LabVIEW, one can use an instrument driver, or can develop own VIs using VISA. The LabVIEW Instrument Library contains instrument drivers for a variety of programmable instrumentation, including GPIB, VXI, and RS-232/422 instruments.

Because instrument driver VIs contain high-level functions with intuitive front panels, end users can quickly test and verify the remote capabilities of their instrument without the knowledge of device-specific syntax. The end user can easily create instrument control applications and systems by programmatically linking instrument driver VIs in the block diagram.

LabVIEW instrument drivers usually communicate with instruments using Virtual Instrument Software Architecture (VISA) functions. VISA is the underlying protocol used when talking to instruments. VISA is a standard I/O Application Programming Interface (API) for instrumentation programming. VISA can control VXI, GPIB, PXI, or serial instruments, making the appropriate driver calls depending on the type of instrument being used.

VISA Programming Terminology

The most important objects in the VISA language are known as *resources*. The functions one can use with an object are known as *operations*. In addition to the operations, the object has variables, known as *attributes*, associated with it that contain information related to the object.

- **Resource:** This is any instrument in system (includes serial and parallel ports).
- **Session:** A VISA session must be opened before, a resource can communicate with it, similar to a communication channel. When one opens a session to a resource, a VISA Session Number is returned, which is a unique handle to that instrument. The Session Number must be used in all subsequent VISA functions. In LabVIEW, this is a refnum.
- **Instrument Descriptor:** This is the exact name of a resource. The descriptor specifies the interface type (GPIB, VXI, ASRL), the address of the device (logical address or primary address), and the VISA Session type (INSTR or Event). The table below shows the proper syntax for the instrument descriptor.

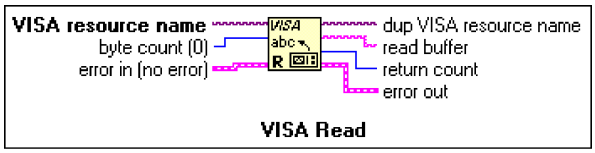
Interface	Grammar
SERIAL	ASRL[board][::INSTR]
GPIB	GPIB[*board*]::*primary address*[::*secondary address*][::INSTR]
VXI	VXI[*board*]::*VXI logical address*[::INSTR]
GPIB-VXI	GPIB-VXI[*board*][::*GPIB-VXI primary address*]::*VXI logical address* [::INSTR]

VISA Functions

VISA Write

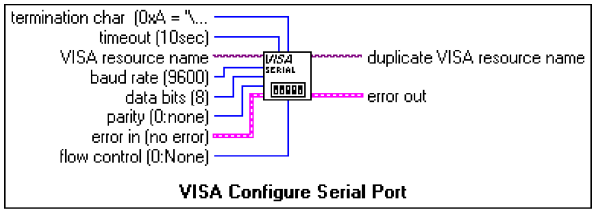
The VISA Write function writes the **write buffer** string to the device specified by the **VISA resource name**. **dup VISA resource name** returns the same handle to that session. On UNIX platforms, data is written synchronously; on all other platforms, it is written asynchronously. **Return count** contains the number of bytes actually transferred across the GPIB.

VISA Read



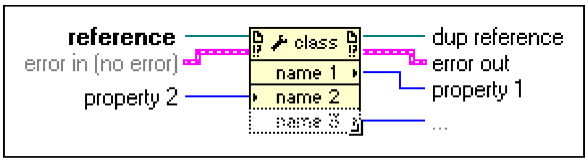
The VISA Read function reads data from the device specified by the **VISA resource name**. **Byte count** indicates the number of bytes to be read into the returned **read buffer** string. **Dup VISA resource name** returns the same handle to that session. On UNIX platforms, data is read synchronously; on all other platforms, it is read asynchronously. **Return count** contains the number of bytes actually transferred across the GPIB.

VISA Configure Serial Port



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.

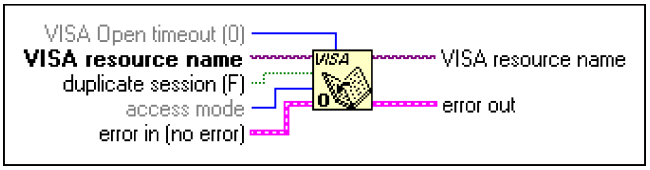
Property Node



Property nodes are used to read or set the values of VISA properties. After placing the property node on the block diagram, wire a VISA Session to the reference input terminal of the property

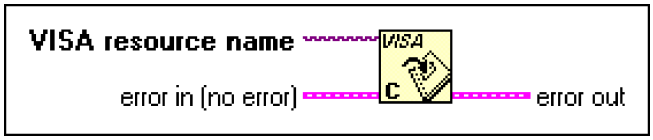
node. The property node contains a single property terminal when it is initially placed on the block diagram. However, it can be resized to contain as many terminals as necessary.

VISA Open



When VISA Read and/or VISA Write is used, LabVIEW checks to see if a reference has been opened for the instrument specified. If a reference is already open, the VISA call uses that reference. If there is not an open reference, VISA automatically opens one. One can choose to explicitly open references to your instruments using VISA Open.

VISA Close



An open session to a VISA resource uses system resources within the computer. To properly end a VISA program, all of the opened VISA sessions should be closed. To do this, VISA Close is to be used.