Lesson 6
Instrument Control

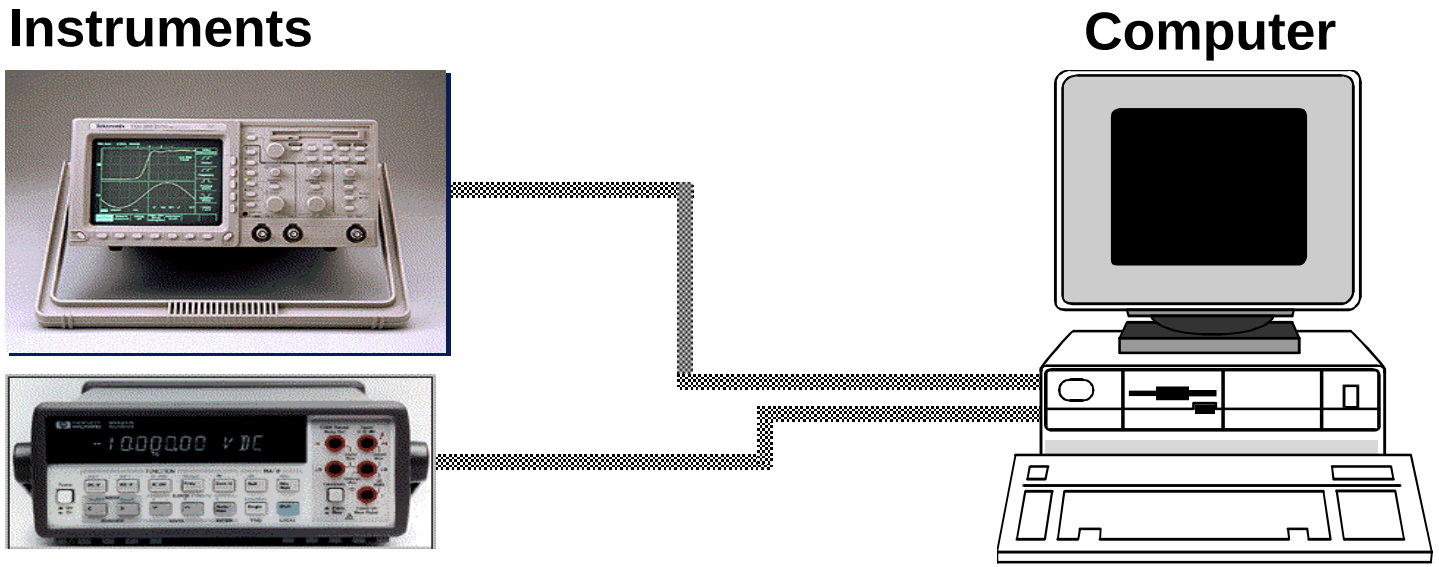
You Will Learn:

- **A. An overview of instrument control**
- **B. About GPIB communication and configuration**
- **C. About LabVIEW instrument drivers**
- **D. How to use instrument driver VIs**
- **E. About Virtual Instrument Software Architecture (VISA)**
- **F. How to use the VISA functions**
- **G. About serial port communication**
- **H. About waveform transfers**

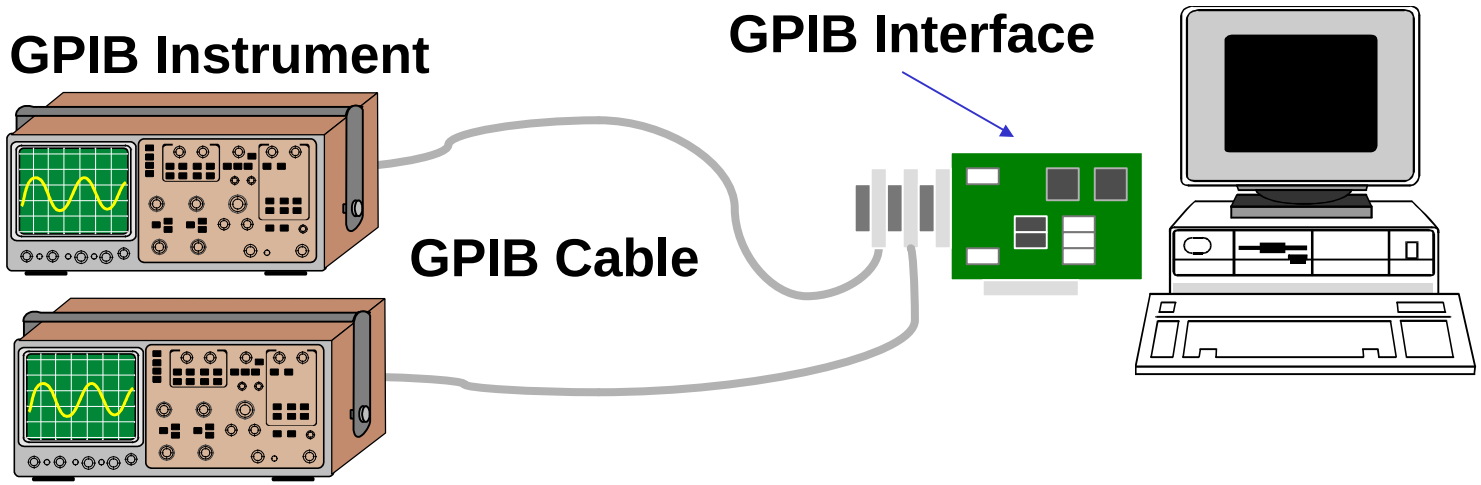
Instrument Control Overview

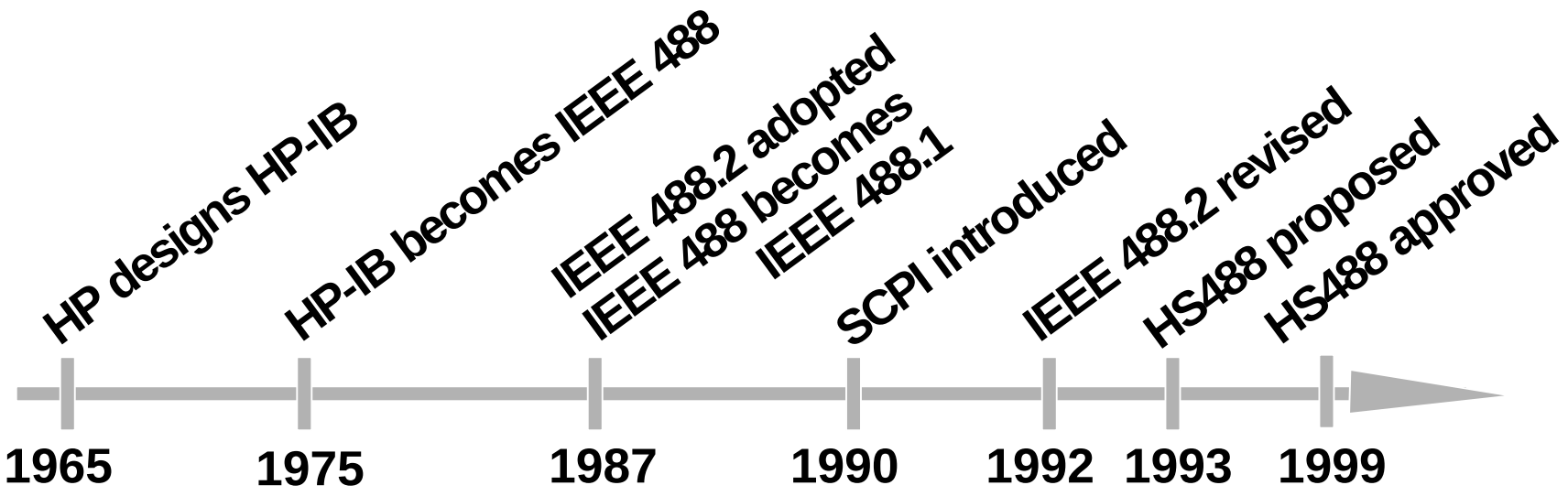
Control any instrument if you know the following:

- **Type of connector on the instrument**
- **Type of cables needed**
- **Electrical properties involved**
- **Communication protocols used**
- **Software drivers available**

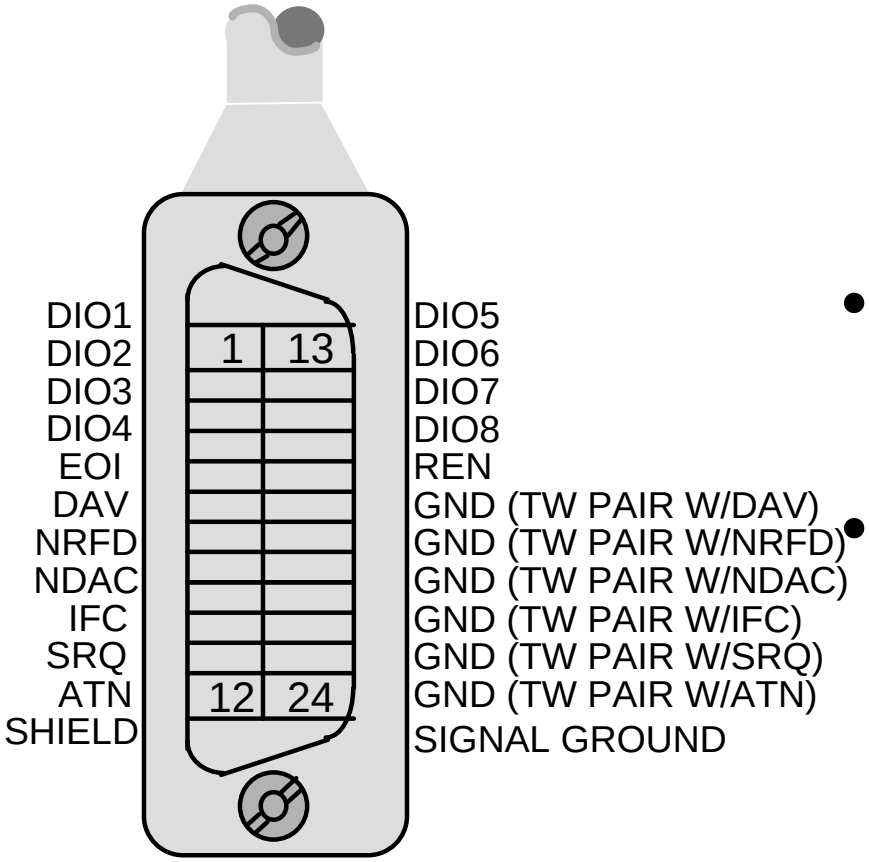


GPIB Communication



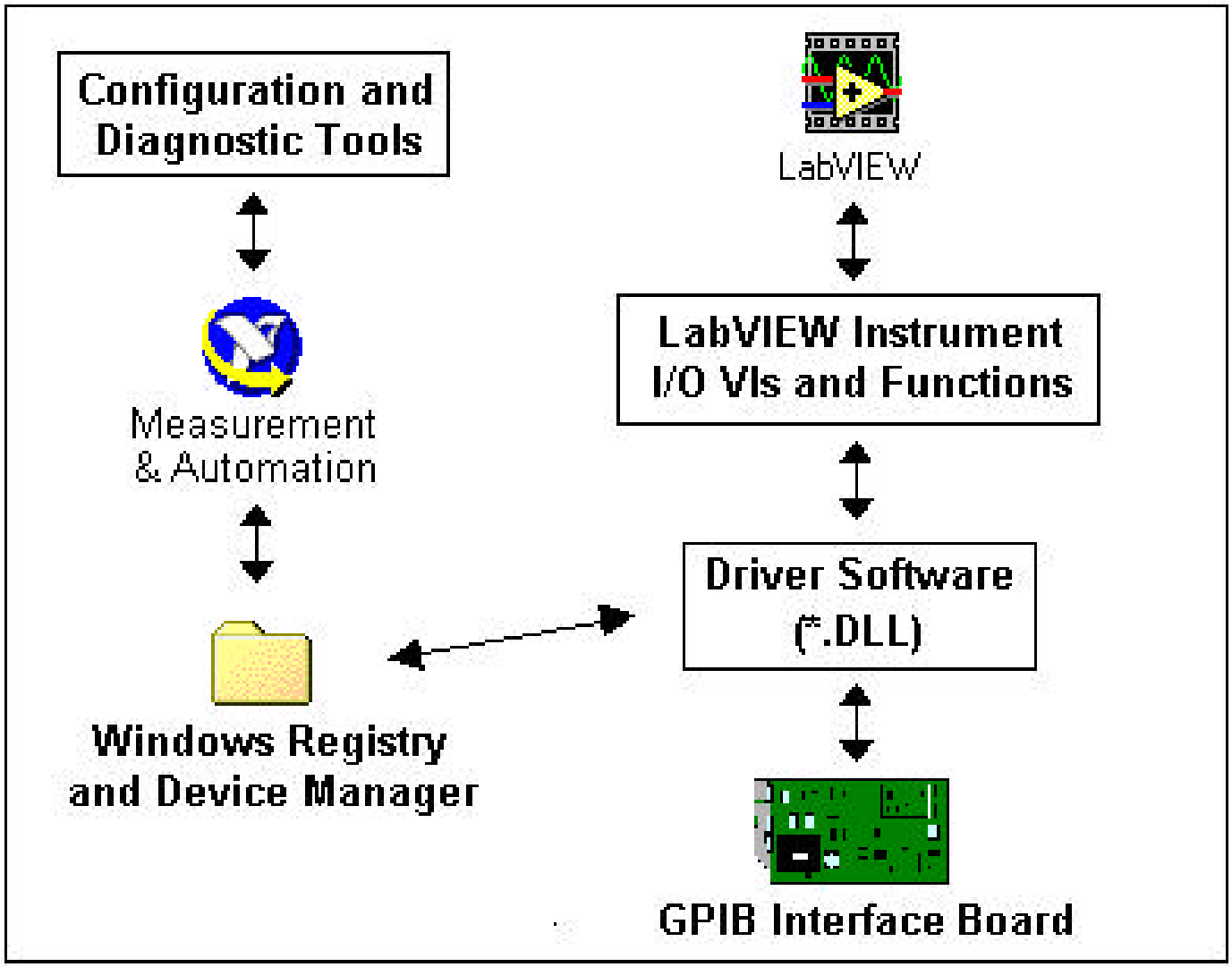


GPIB Hardware Specifications



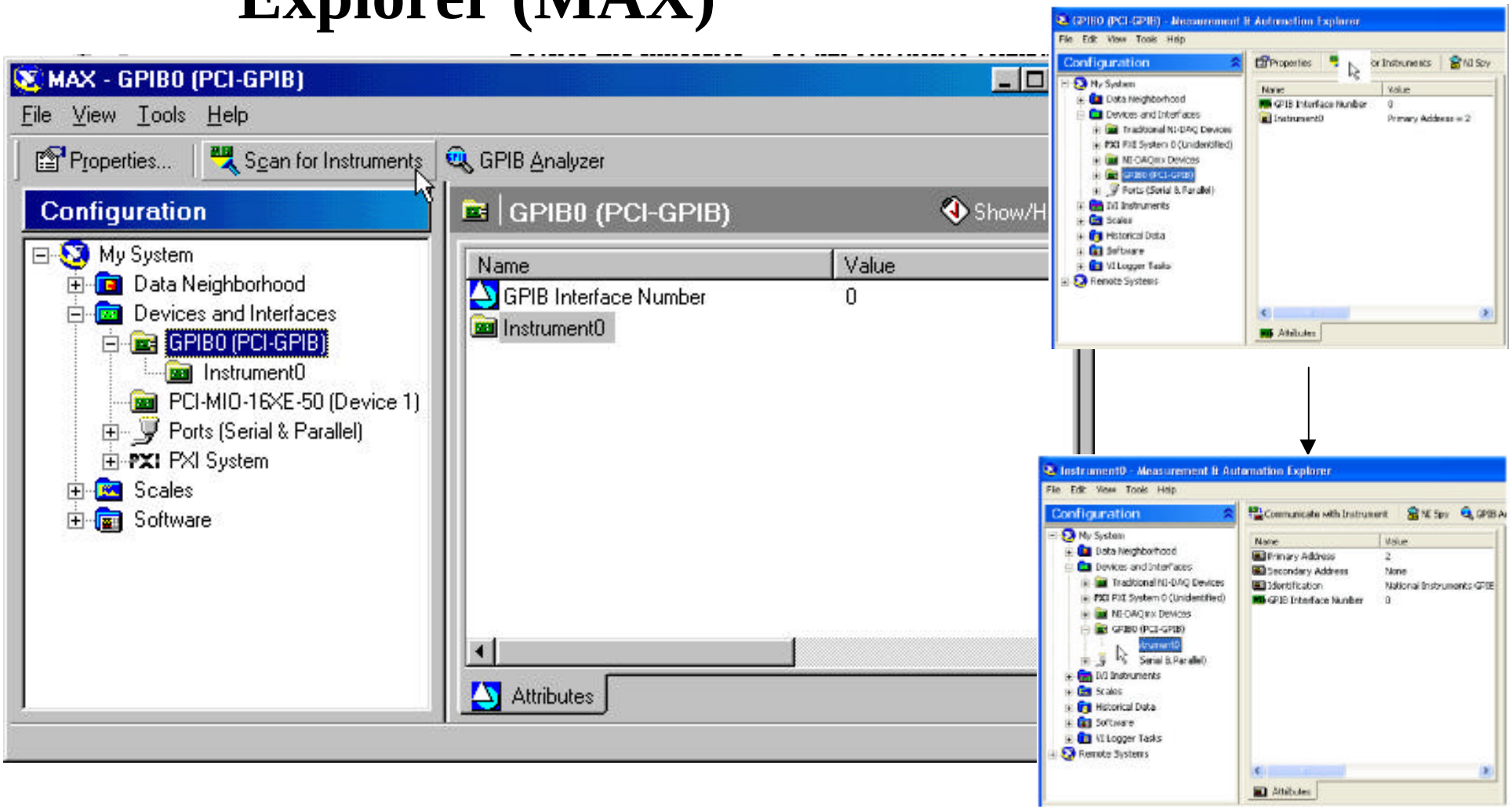
- **Max cable length between devices = 4 m (2 m average)**
- **Max cable length = 20 m**
- **Max number of devices = 15 (2/3 powered on)**

GPIB Software Architecture – Windows

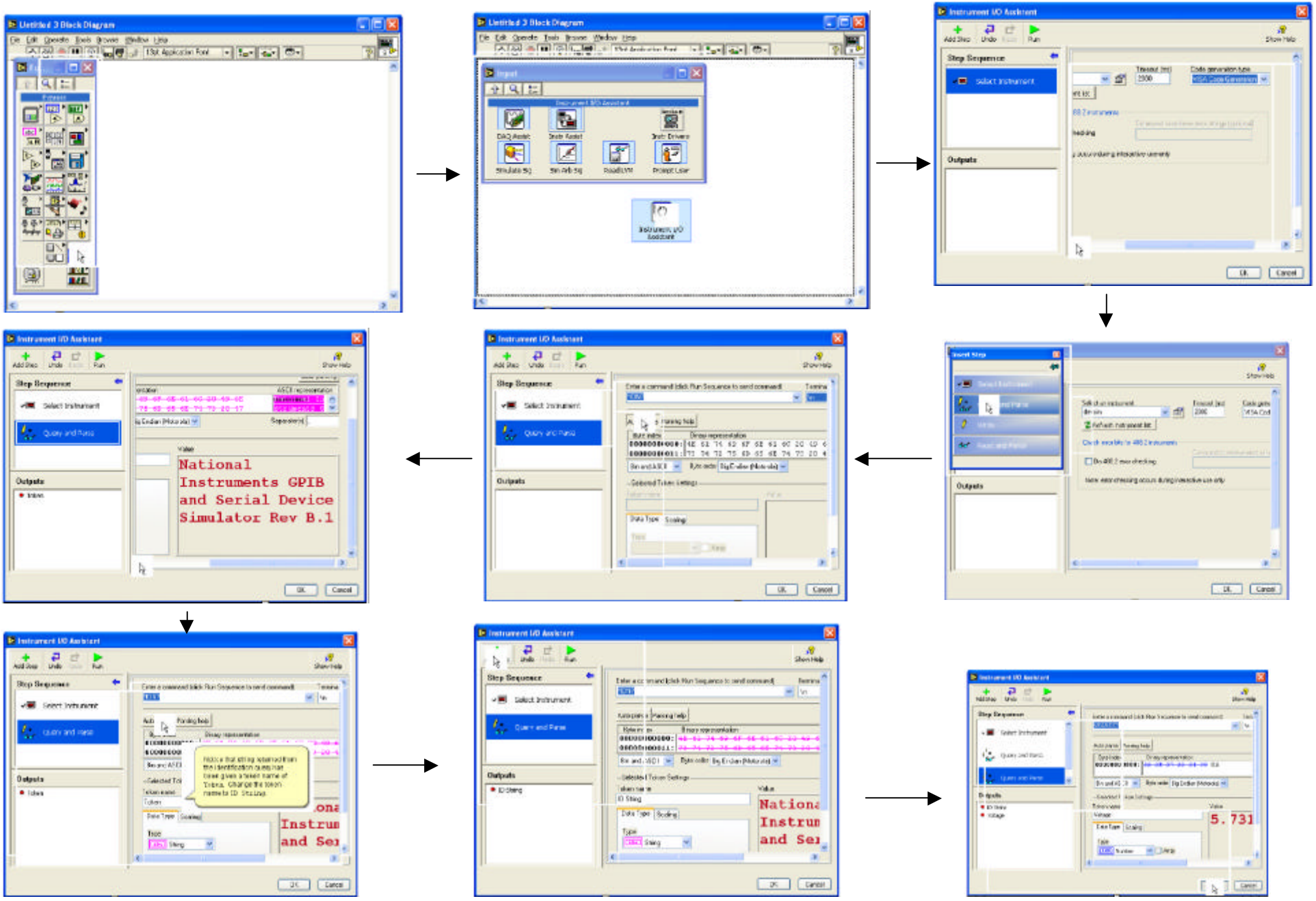


Configuring GPIB Board and Instruments

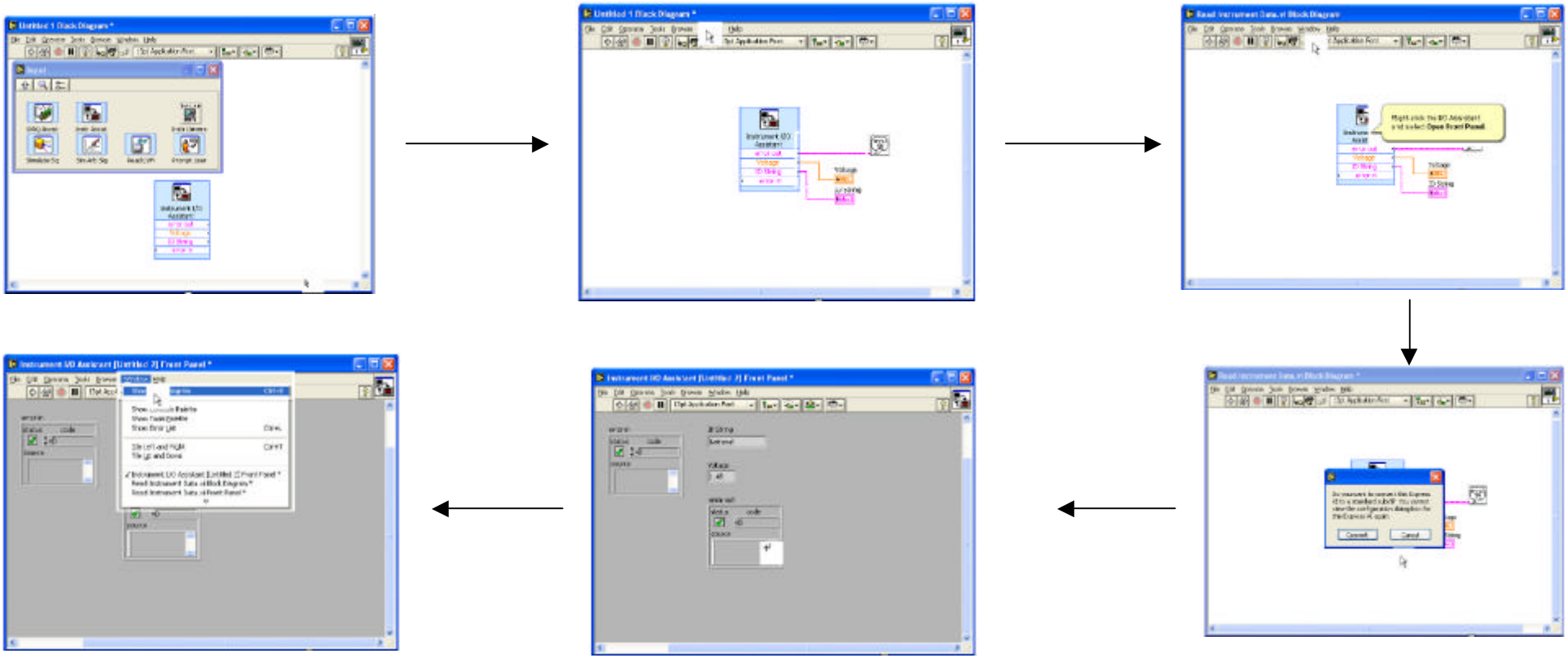
- **Measurement & Automation Explorer (MAX)**



Configuring GPIB Board and Instruments

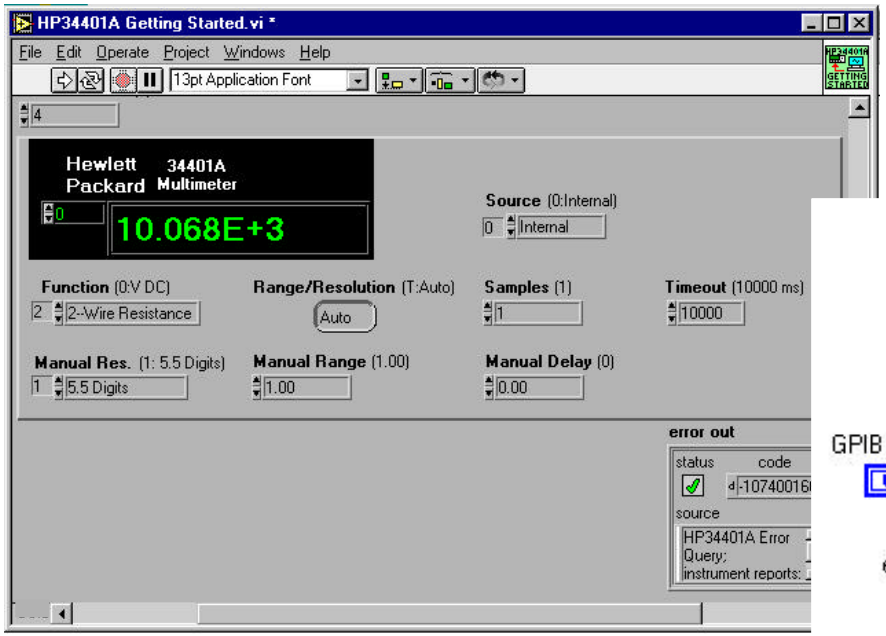


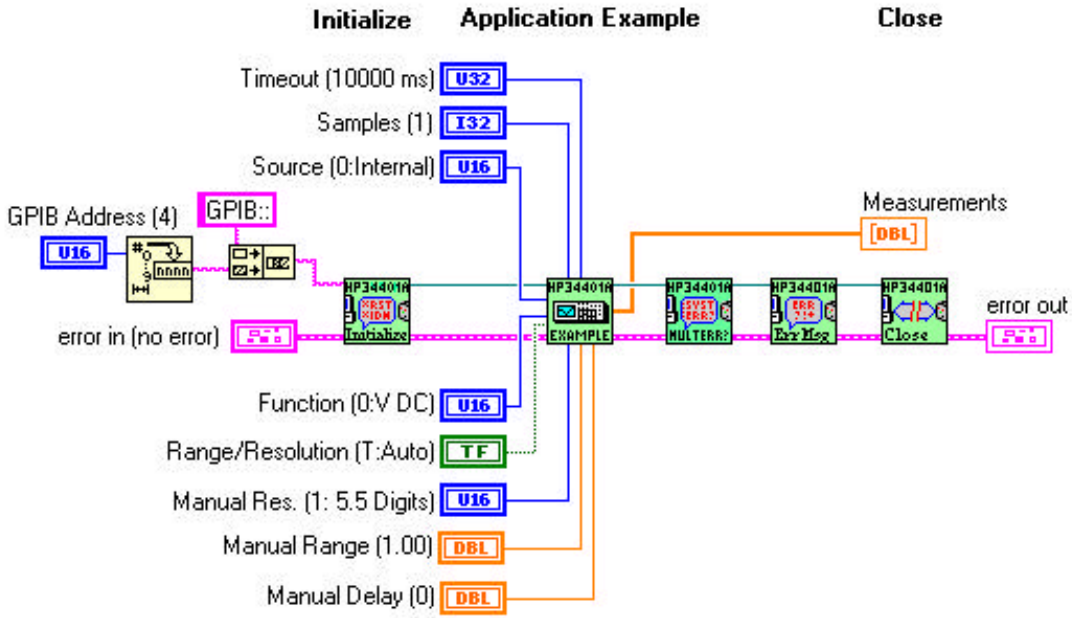
Configuring GPIB Board and Instruments



Instrument Drivers

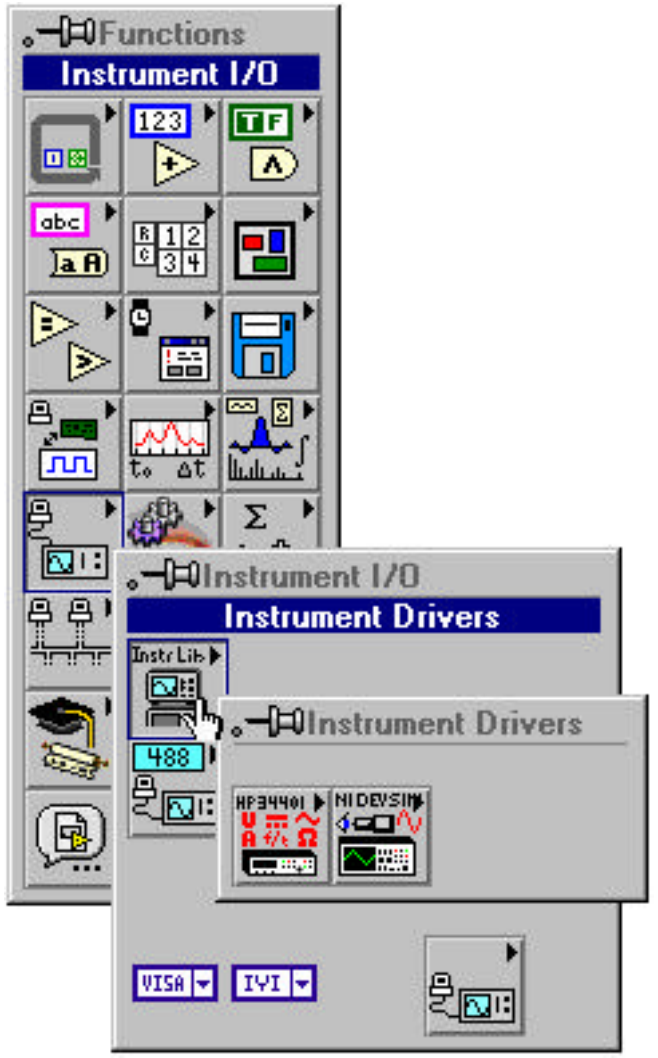
- **More than 2,200 LabVIEW instrument drivers from more than 150 vendors**
- **Programming simplified to high-level Application programming interface (API)**





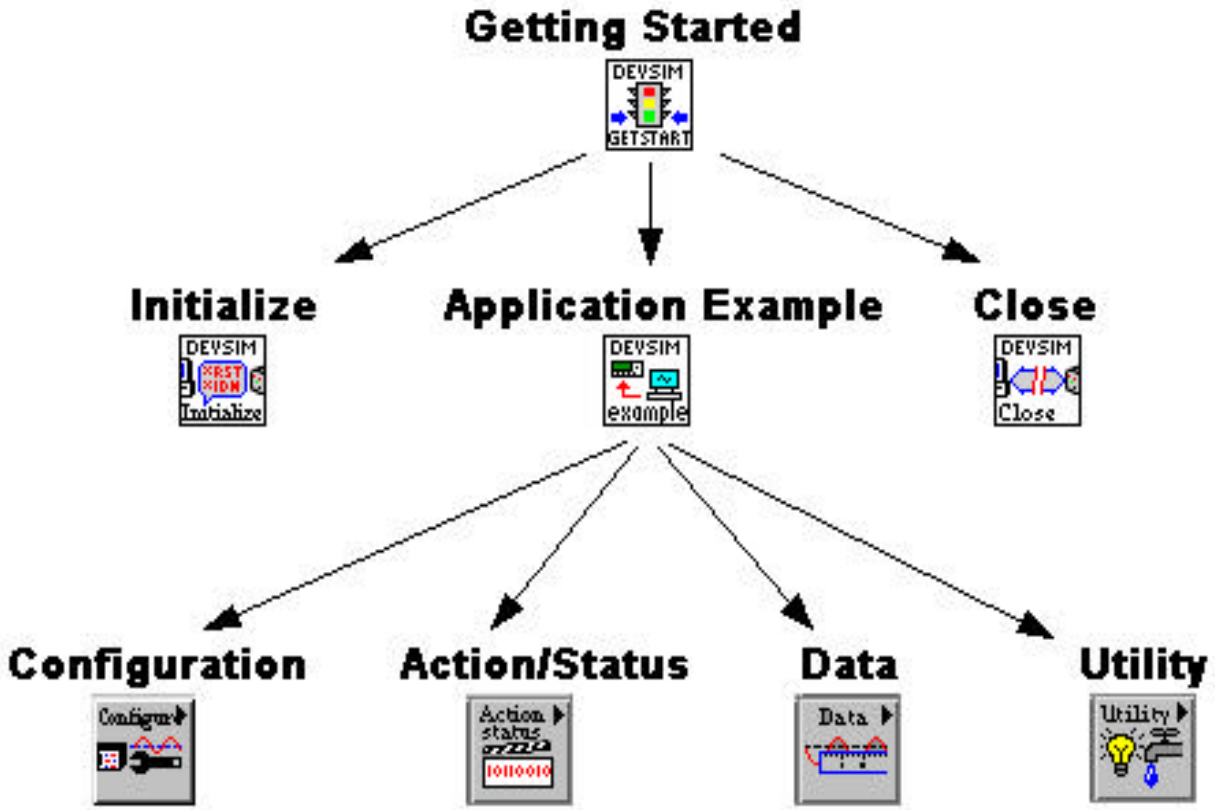
Installing and Finding Instrument Drivers

- **NI Web site or Instrument Driver CD**
- **Install the instrument driver VI Library into `LabVIEW\instr.lib` directory**
- **Access drivers from `Instrument I/O` » `Instrument Drivers` subpalette**



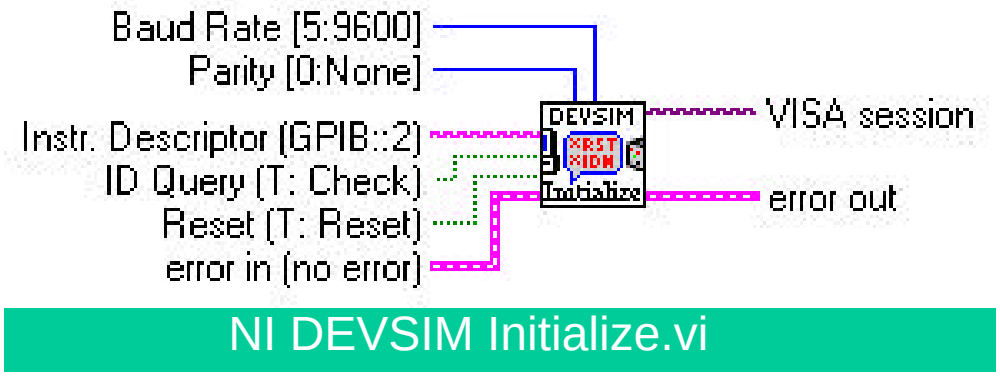
Instrument Drivers

- **Instrument drivers have a similar hierarchy**



Instrument Driver VI Tree

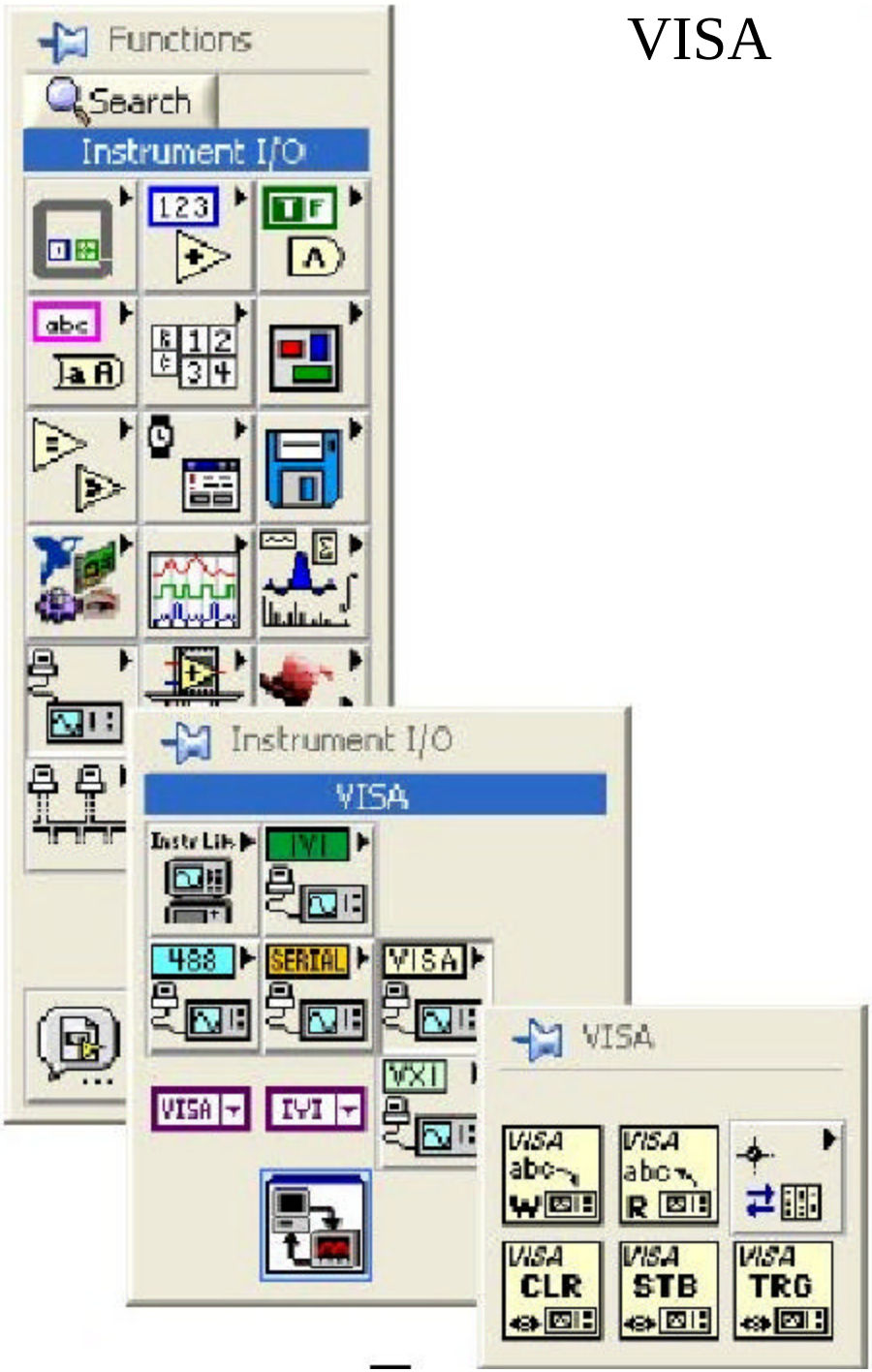
Instrument Driver Inputs and Outputs



NI DEVSIM Initialize.vi

- **VISA Sessions**
  - **A connection or link to a specific instrument**
  - **Created after instrument is initialized**
  - **Used throughout VI whenever you communicate with that specific instrument**
- **Error Clusters**

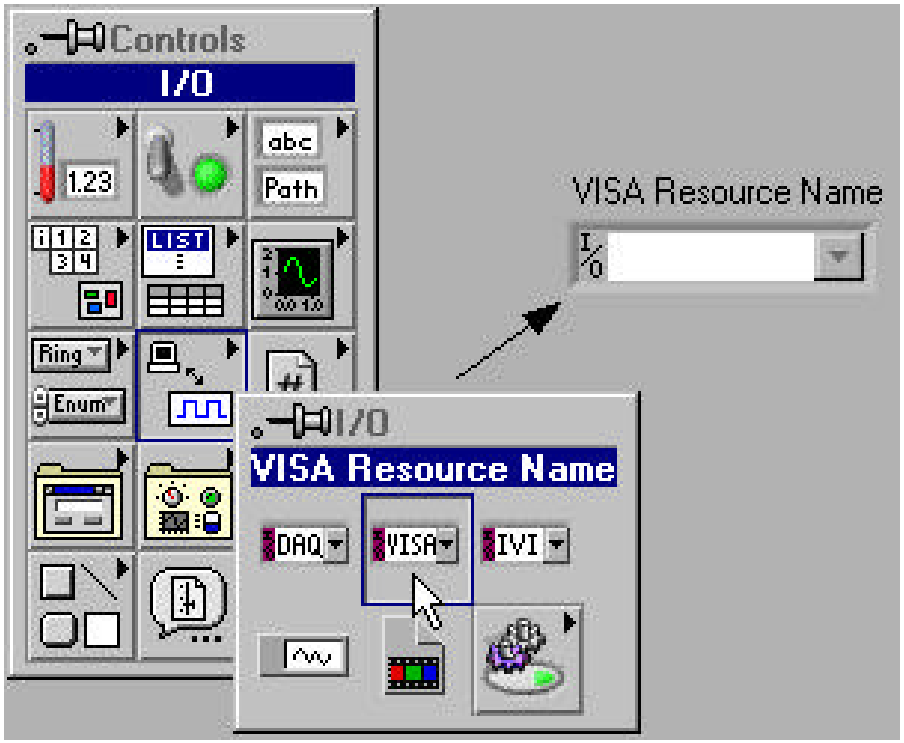
VISA

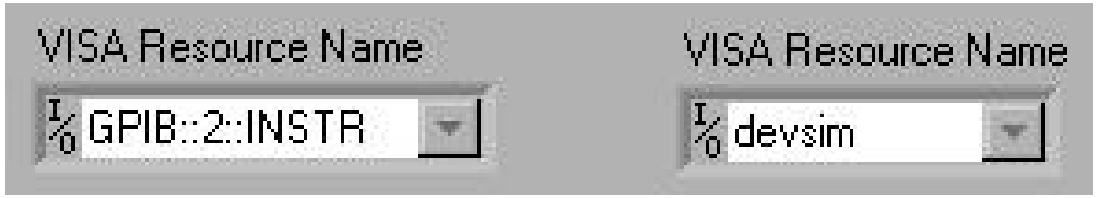


VISA VIs and Functions

The lower layer of functions in the LabVIEW instrument driver VIs that communicates with the driver software uses the Virtual Instrument Software Architecture (VISA). The VISA architecture enables instrument drivers to talk to different hardware interfaces using one standard application programming interface. You can use LabVIEW VISA functions and VIs to communicate with an instrument that does not have a driver or that has a driver that implements only a subset of the total capabilities of the instrument. This lesson shows how you can use the VISA Write and VISA Read functions to communicate with GPIB instruments.

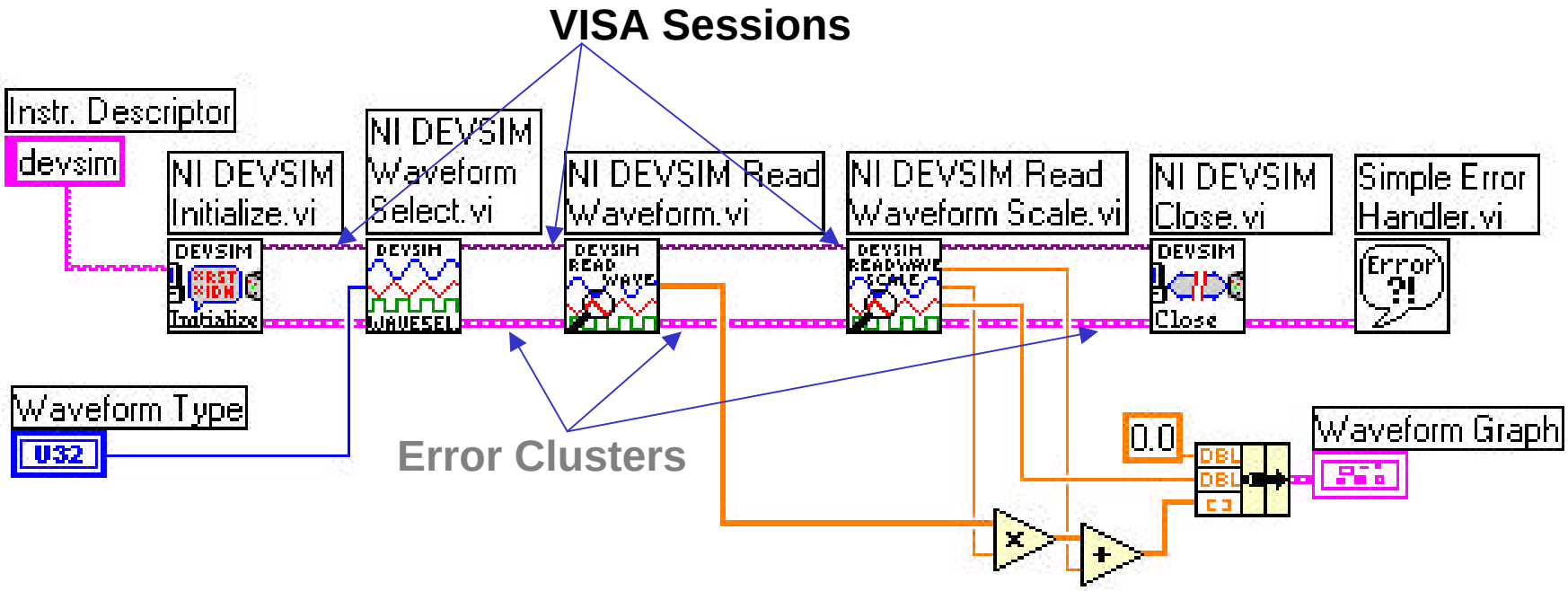
VISA Resource Name





- **Exact name and location of the instrument**
- **Use the VISA Resource Name control (like the DAQ Channel Name control)**
- **You can specify the full resource name or the VISA Alias**

Putting It All Together



- **Initialize instrument**
- **Do operation(s)**
- **Close instrument**
- **Check error status**

VISA

Interface	Syntax
Asynchronous	`ASRL[board][::INSTR]`
GPIB	`GPIB[board]::primary address[::secondary address][::INSTR]`
VXI instrument through embedded or MXIbus controller	`VXI[board]::VXI logical address[::INSTR]`
GPIB-VXI controller	`GPIB-VXI[board][::GPIB-VXI primary address]::VXI logical address[::INSTR]`

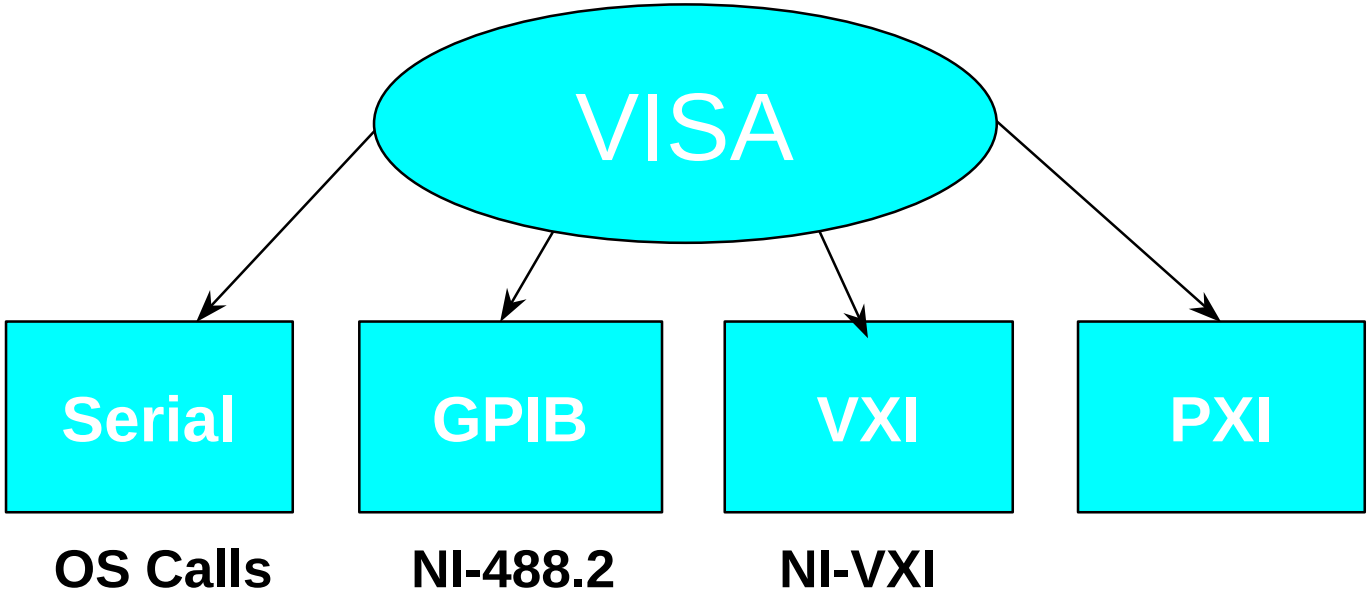
syntax for the VISA resource name varies depending
he interface to the instrument. The table on this slide
ides the syntax for the VISA resource name for the
chronous, GPIB, VXI, and GPIB-VXI interfaces. You can
Measurement & Automation Explorer (MAX) to
rmine what resources and instrument addresses are
lable. Refer to Exercise 9-1 in this program for
rmation about how to use MAX.

When you initialize the communication channel to an instrument, you must know the VISA resource name or instrument descriptor. A resource is an instrument or interface and the instrument descriptor is the exact name and location of a resource in the following format:

Interface Type[*board index*]::*Address*::INSTR

For example, `GPIB::2::INSTR` is the instrument descriptor for a GPIB instrument at address 2. Use the VISA Resource Name control (shown above) located on the **I/O** subpalette of the **Controls** palette to supply the VISA resource name from the front panel controls of a VI.

Virtual Instrument Software Architecture



- **Standard API for programming instruments**
  - **Platform independent**
  - **Interface independent**
  - **Interface to Serial, GPIB, and VXI instruments**
  - **Uses VISA.DLL**

VISA Terminology

- **Resource—Instrument, Serial Port, or Parallel Port**
- **Session—Connection to a Resource**
- **Instrument Descriptor—Resource location**
- **Format: Interface Type::Address::INSTR**
- **Examples:**

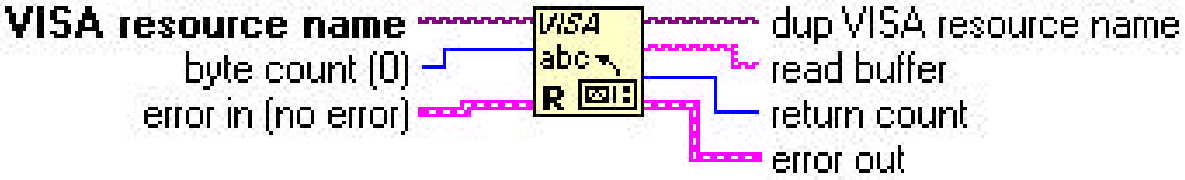
Instrument Descriptor Syntax

- **Resource Name contains interface info**
- **VISA Aliases also work**

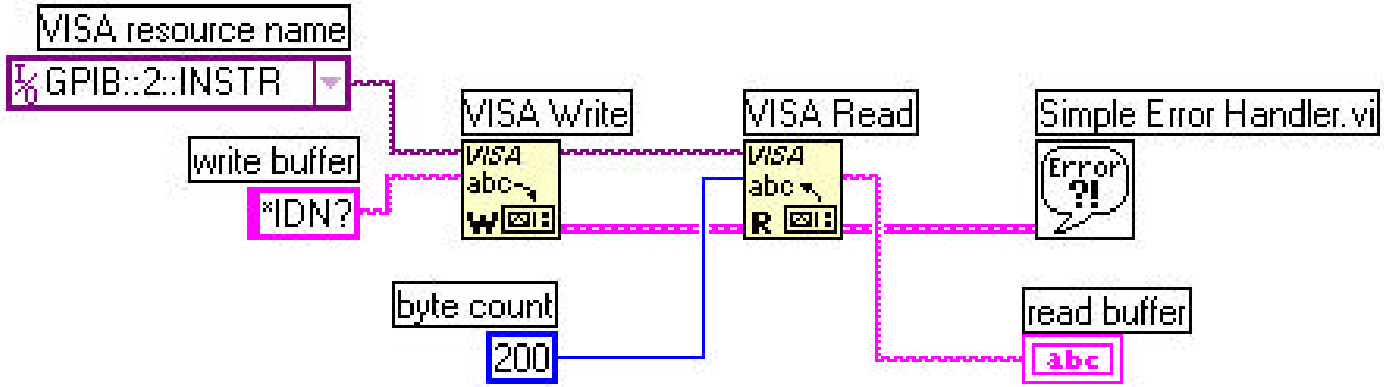
Interface	Resource Name Grammar
Serial	**ASRL[board][::INSTR]**
GPIB	**GPIB[board]::*primary address*[::INSTR]**
VXI	**VXI[board]::*VXI logical address*[::INSTR]**
GPIB-VXI	**GPIB-VXI[board][::*GPIB-VXI primary address*]::*VXI logical address*[::INSTR]**

VISA Functions

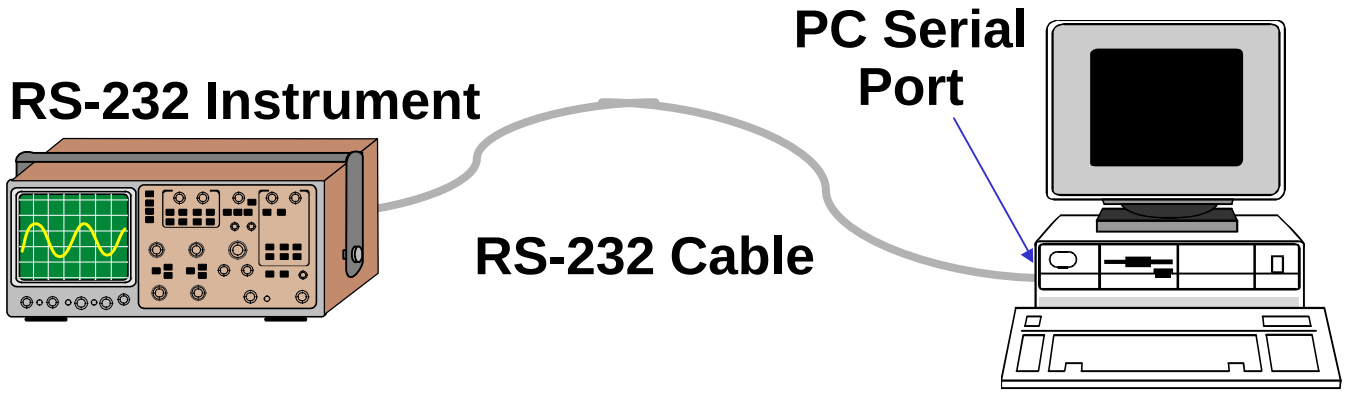
VISA Write



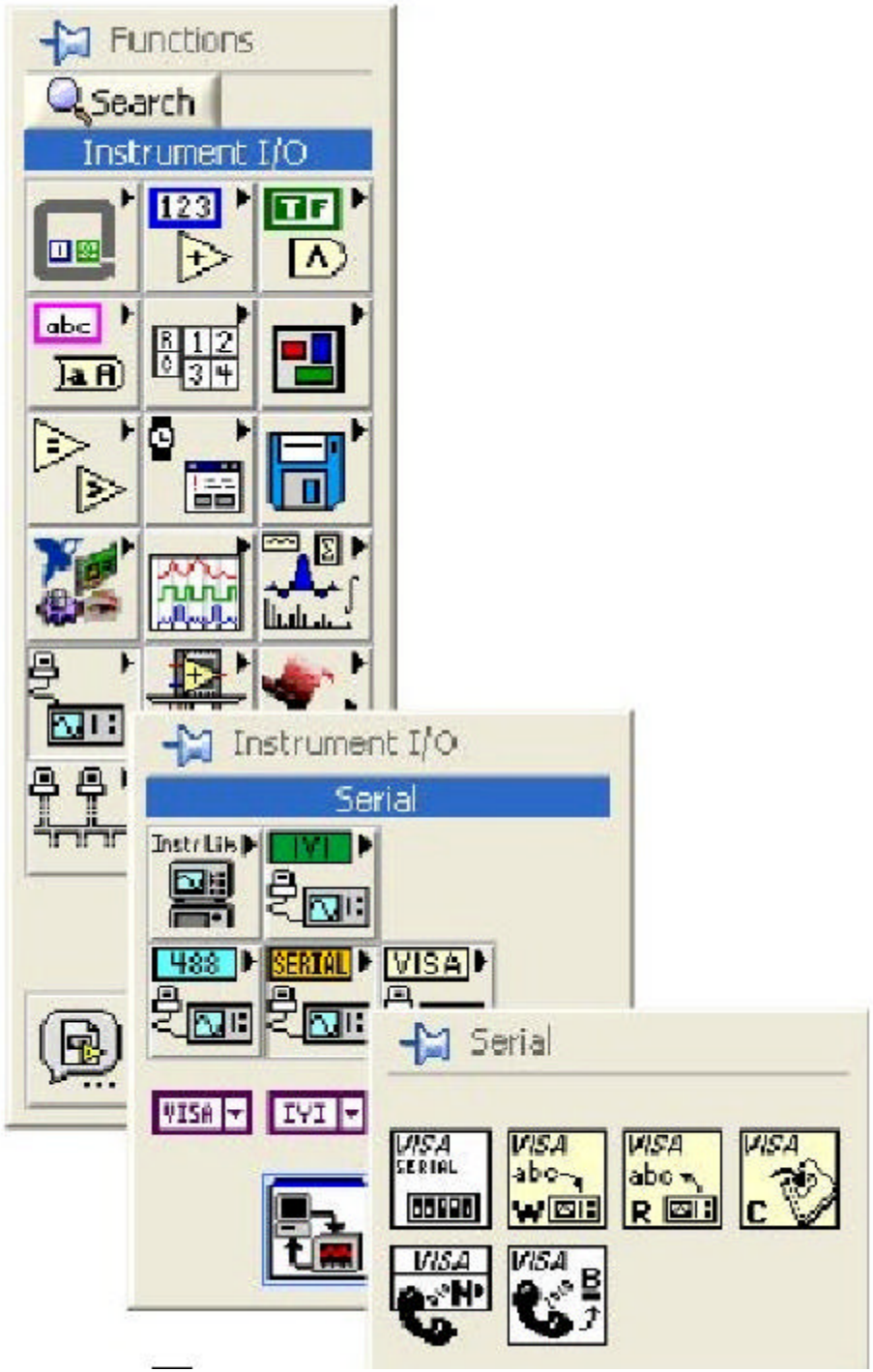
VISA Read



Serial Communication



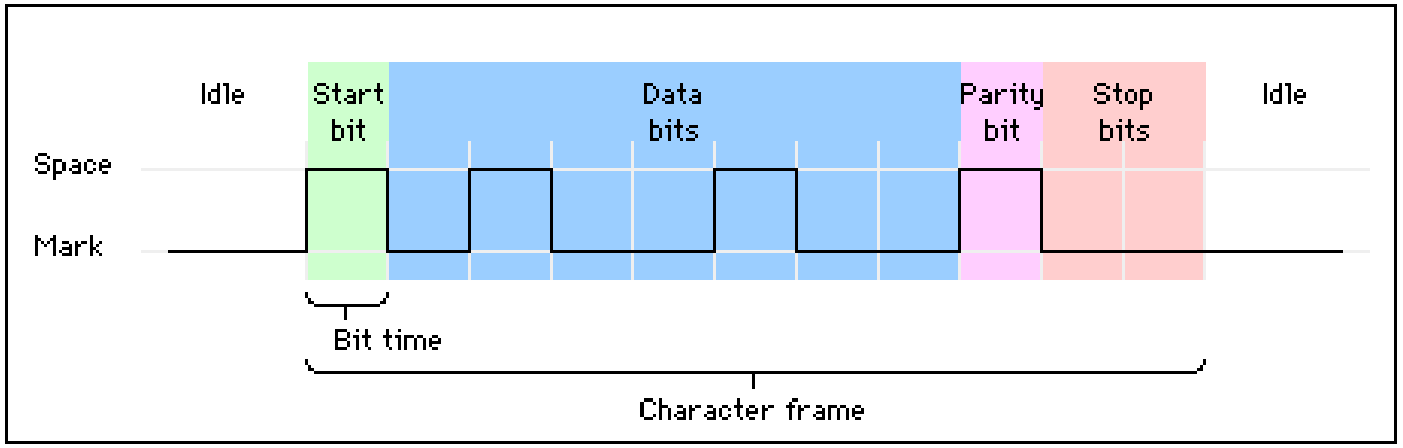
- **Popular means of communication between computer and peripheral device**
- **Data sent one bit at a time across the cable**
- **Used for low transfer rates or long distances**
- **Only a cable is needed since most computers have at least one available serial port**



Serial Port Communication

Use the VIs and functions located on the **Serial** subpalette for serial port communication. You used the VISA Write and VISA Read functions on this palette for GPIB communication in Exercise 9-4. The VISA Read and VISA Write functions work with any type of instrument communication, including serial. However, because serial communication requires you to configure extra parameters, you must start the serial port communication with the VISA Configure Serial Port VI. This lesson shows how you can use the VISA Configure Serial Port VI to communicate with an instrument through a serial interface.

Serial Communication

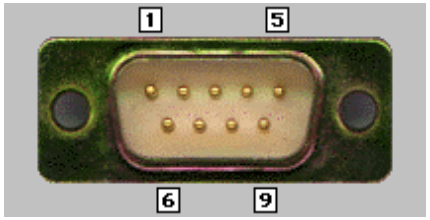


Terminology

- **Baud rate – bits per second**
- **Data bits – inverted logic and LSB first**
- **Parity – optional error-checking bit**
- **Stop bits – 1, 1.5, or 2 inverted bits at data end**
- **Flow control – hardware and software handshaking options**

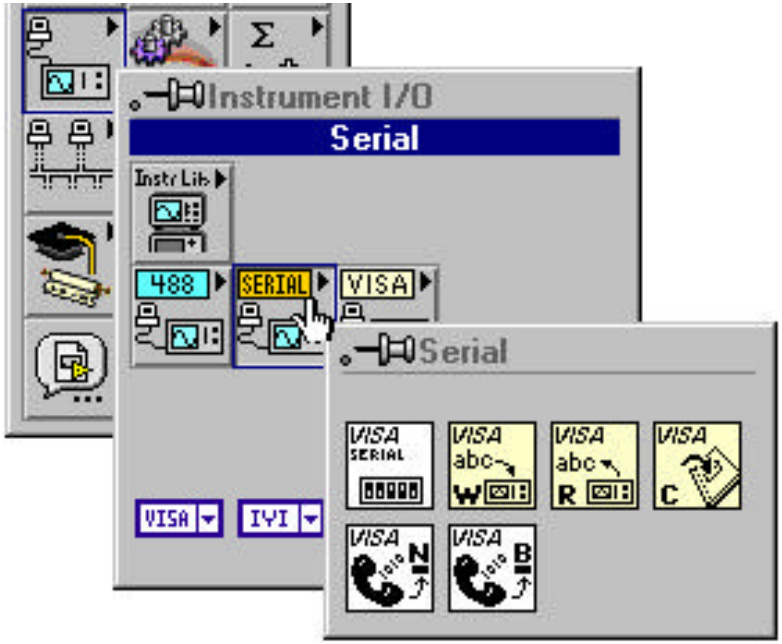
Serial Hardware Connection

- **RS-232**
  - **Most PCs**
  - **DCE or DTE configurations**
  - **8-pin or 25-pin**
  - **Single-ended**
- **RS-422**
  - **Macintosh**
  - **8-pin**
  - **Differential**
- **RS-485**
  - **Multidrop**
  - **Industrial Automation**

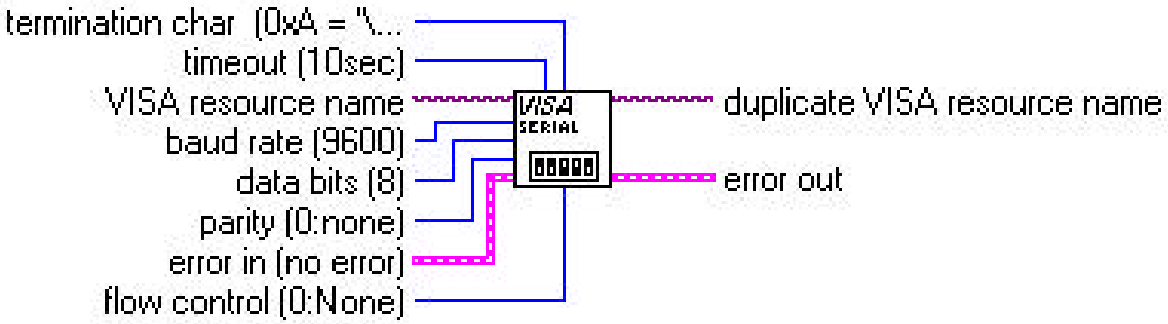


Pin		DTE	DCE
1	DCD	Input	Output
2	RxD	I	O
3	TxD	O	I
4	DTR	O	I
5	Com	-	-
6	DSR	I	O
7	RTS	O	I
8	CTS	I	O
9	RI	I	O

Serial VIs and Functions

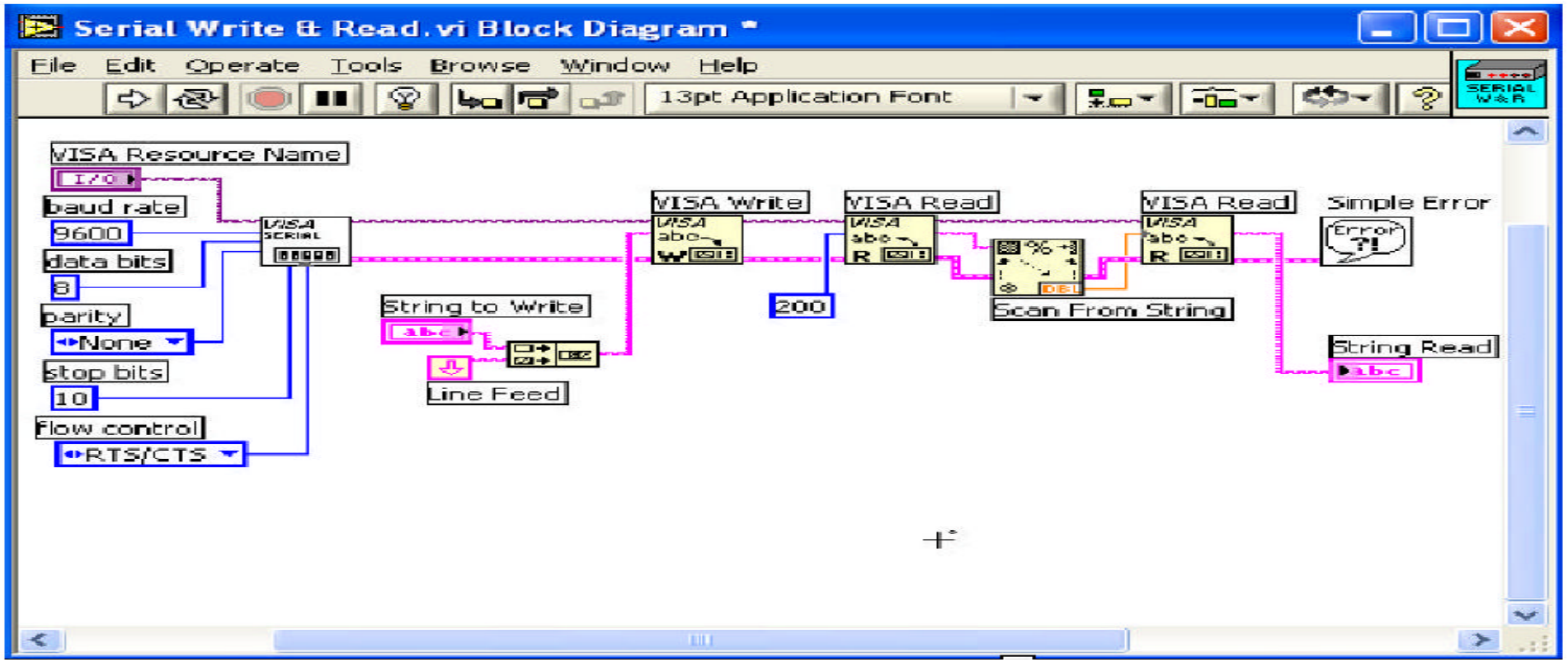


- **Found in Serial subpalette under Instrument I/O**
- **Based on VISA functions**
- **Serial VIs and functions also work with parallel port communication**



VISA Configure Serial Port

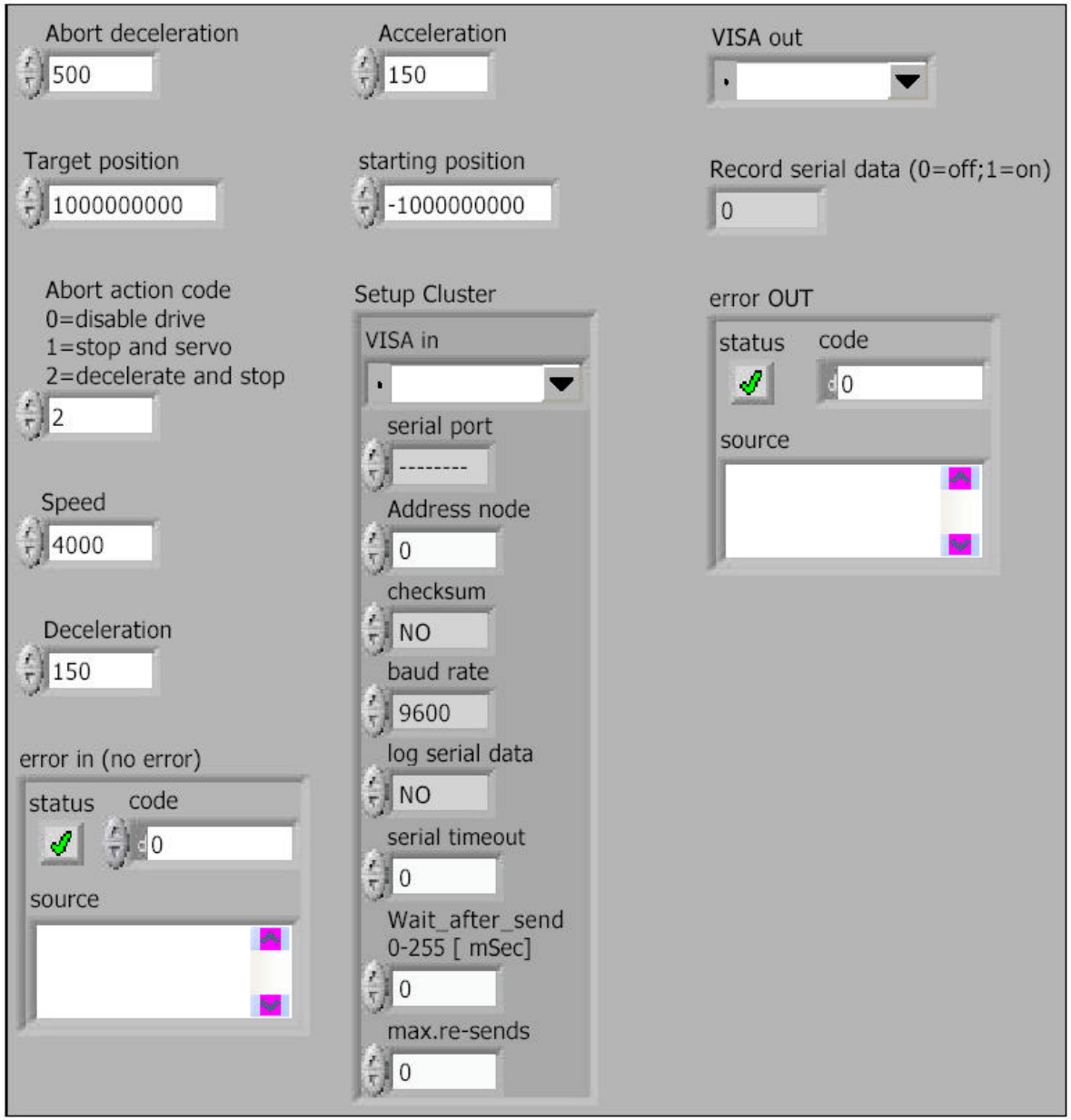
Serial I/O Example



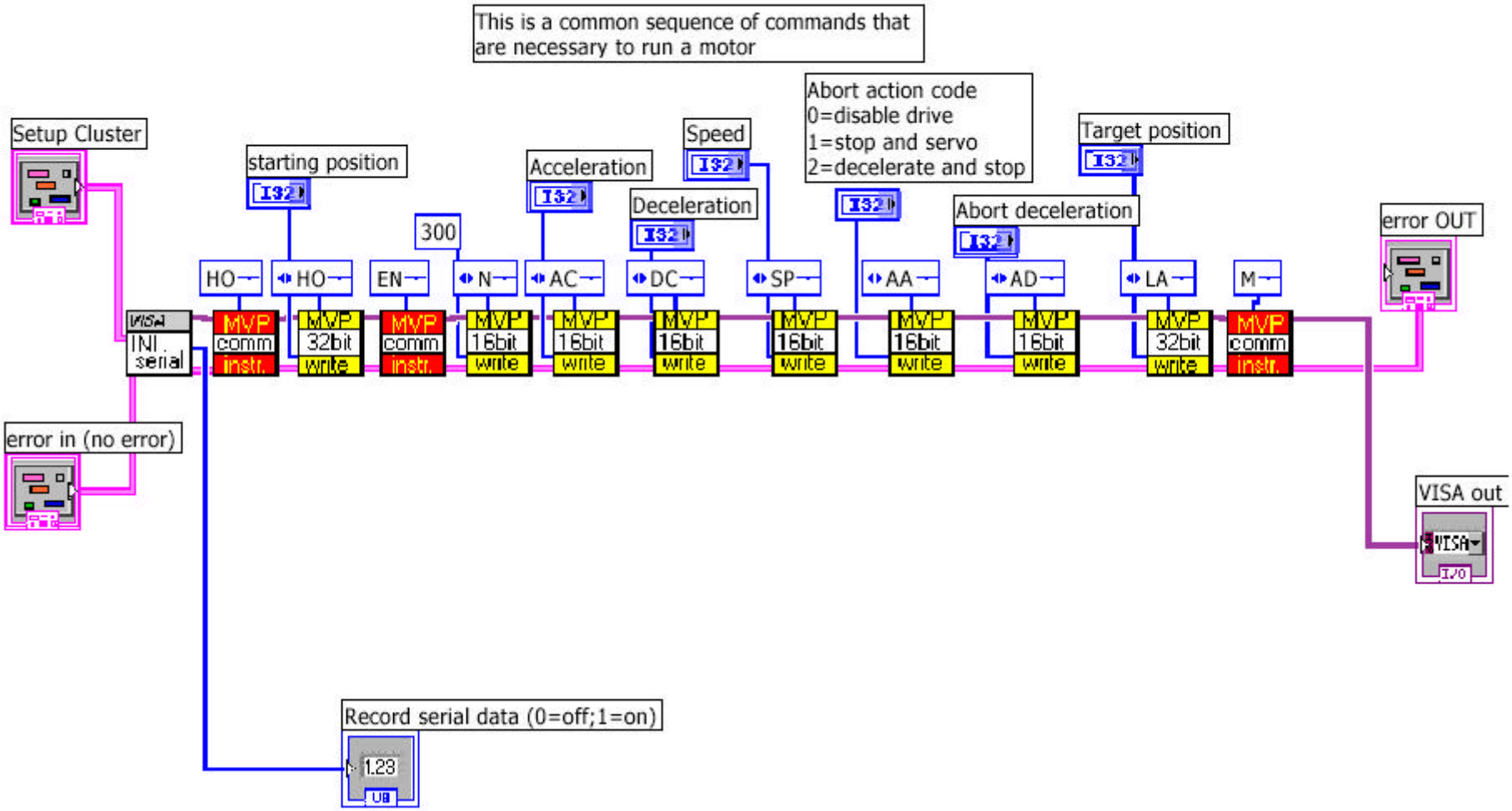
- **Initialize the serial port settings**
- **Write commands to the device**
- **Read device response**
- **Check for errors**

Serial I/O Example

National Aperature stepper motor



Block Diagram



Summary

- **LabVIEW can communicate with any instrument that connects to your computer if you know the interface type**
- **Use the Measurement & Automation Explorer (MAX) to detect, configure, and test your GPIB interface and instruments**
- **An instrument driver eliminates the need for your to have detailed knowledge of the specific strings used by an instrument**
- **Instrument Library – more than 650 instruments supported**
- **Instrument driver VIs share a common hierarchy and come with an example to help you get started**
- **VISA a standard protocol for using multiple types of I/O and instrument driver development**
- **Serial library contains functions for serial communication**
- **You need to know the format of the returned data string in order to convert it to the correct values**