DEVELOPING INSTRUMENT DRIVERS with LabVIEW

Terry Stratoudakis, PE
Certified LabVIEW Developer

ALE System Integration
Melville, New York
June 12, 2008

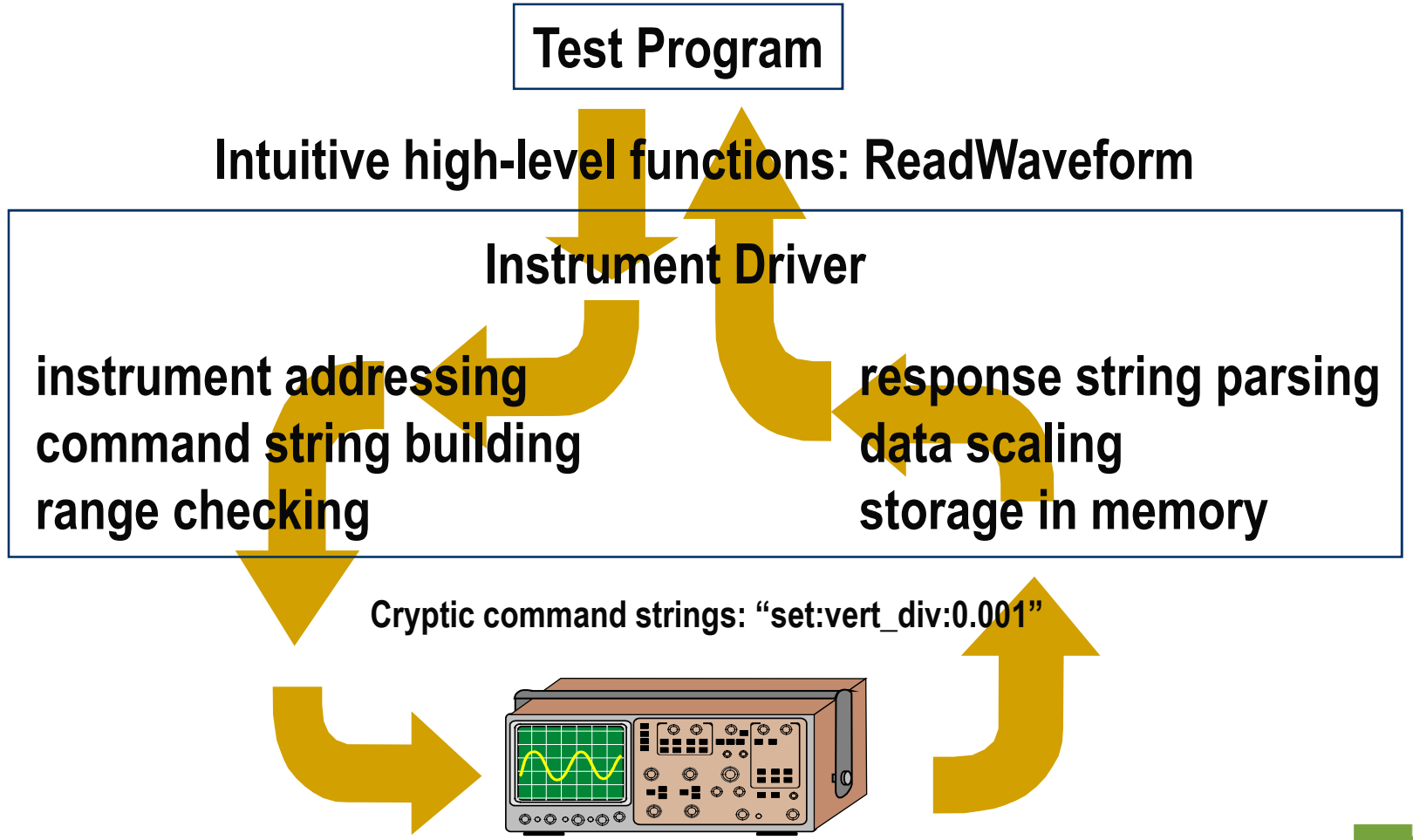
Overview

I. Why use Instrument Drivers?
II. Finding LabVIEW Instrument Drivers
III. Instrument Interfaces
IV. Instrument Driver Architecture
V. NI Instrument Driver Guidelines
VI. Calling Shared Libraries
VII. Example:
    I. Tektronix AFG 3252
    II. Tektronix DPO7104

I. Why use Instrument Drivers?

- Simplify Development
- Automate Testing
- Abstract Developer from Device Syntax
- Code Reuse
  - write once, use many times

Simplify Development with Instrument Drivers

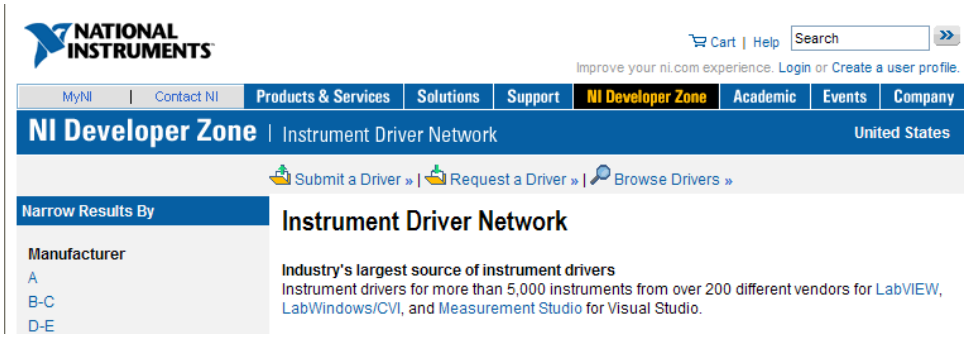


II. Finding Instrument Drivers

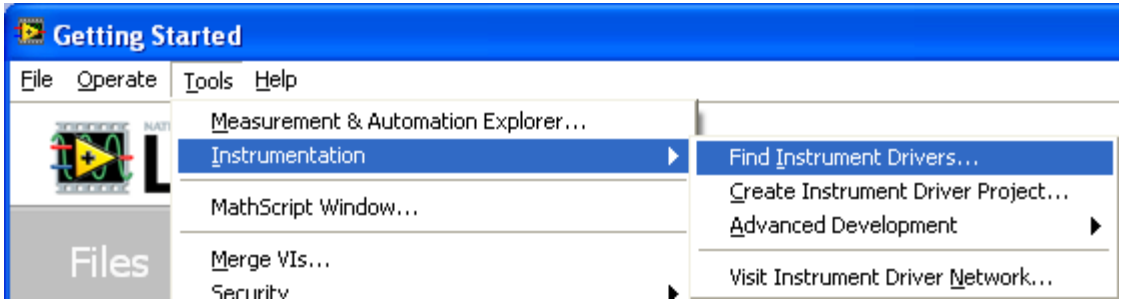
- Manufacturer's website
  - Not all provide drivers
  - Some offer only API (e.g. DLLs)
- NI Instrument Driver Network
  - Learn about drivers
  - Get help with developing drivers
  - Submit your driver to the network
- Request Instrument Driver

NI Instrument Driver Network

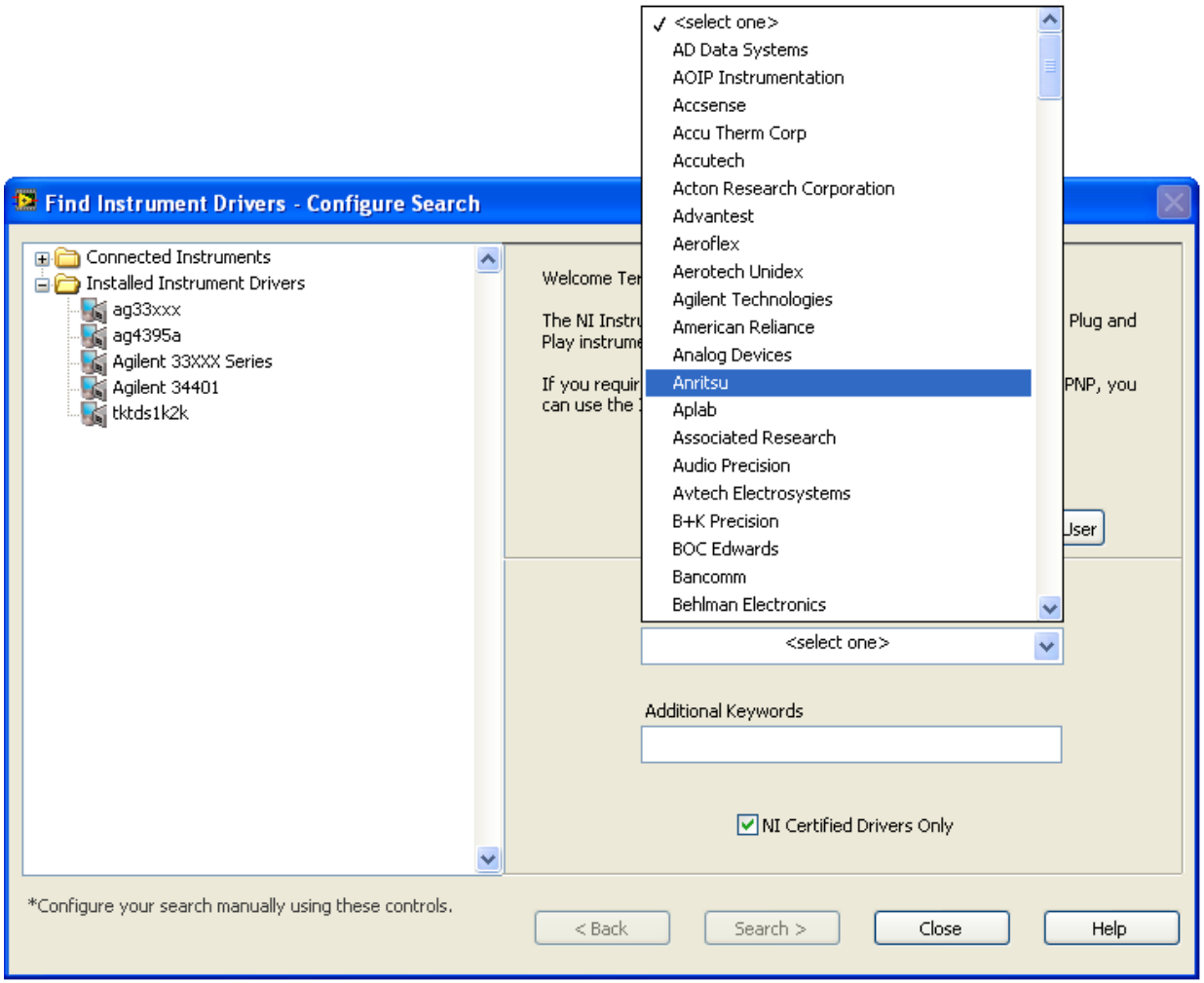
- Via web browser www.ni.com/idnet



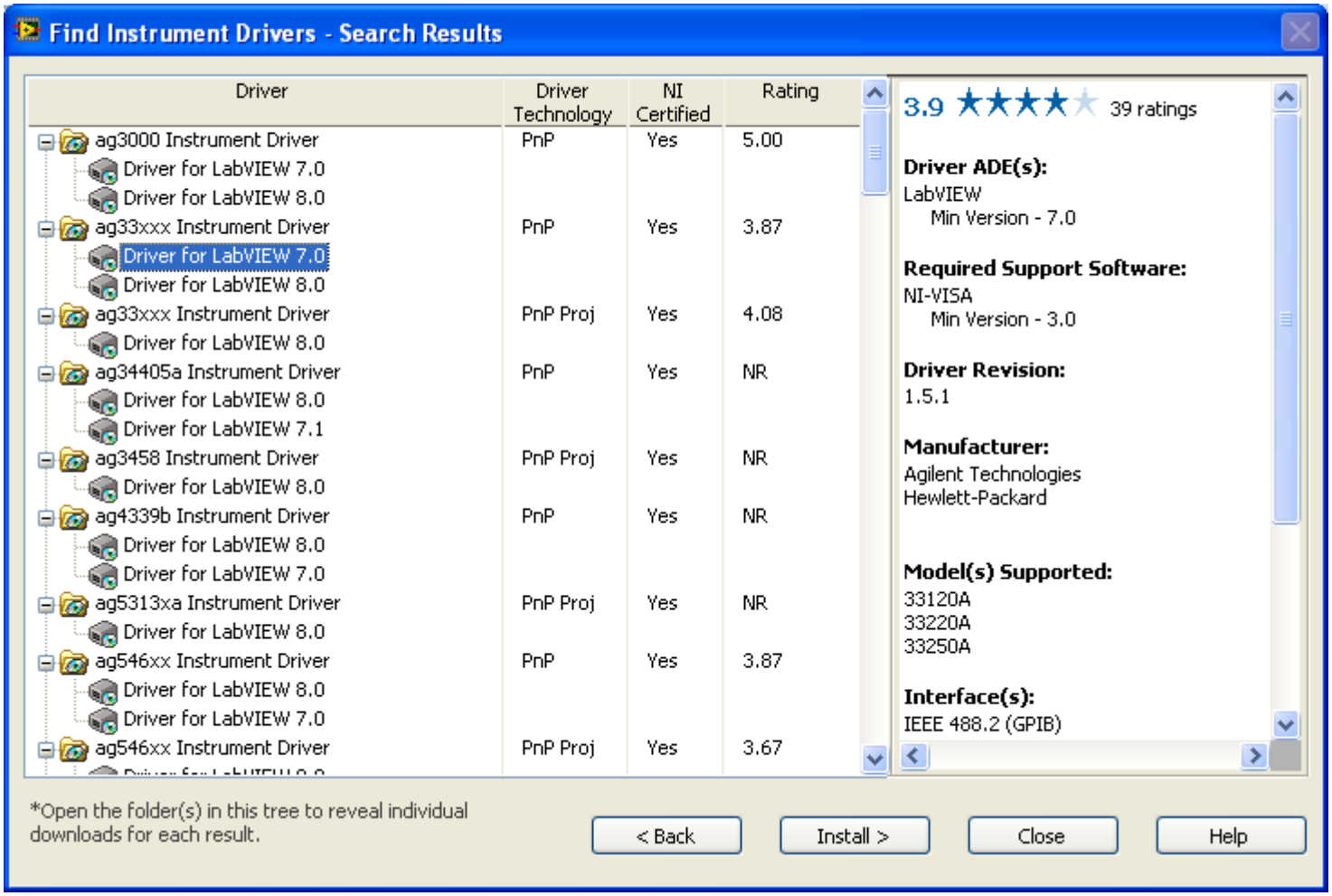
- Find drivers from within LabVIEW



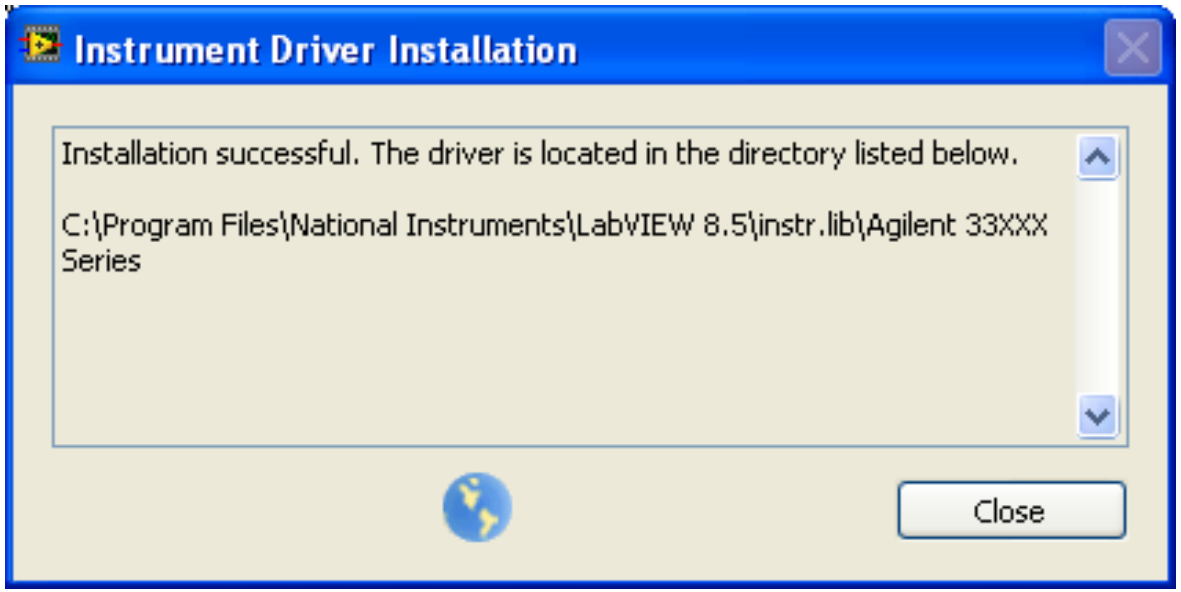
Search by Manufacturer



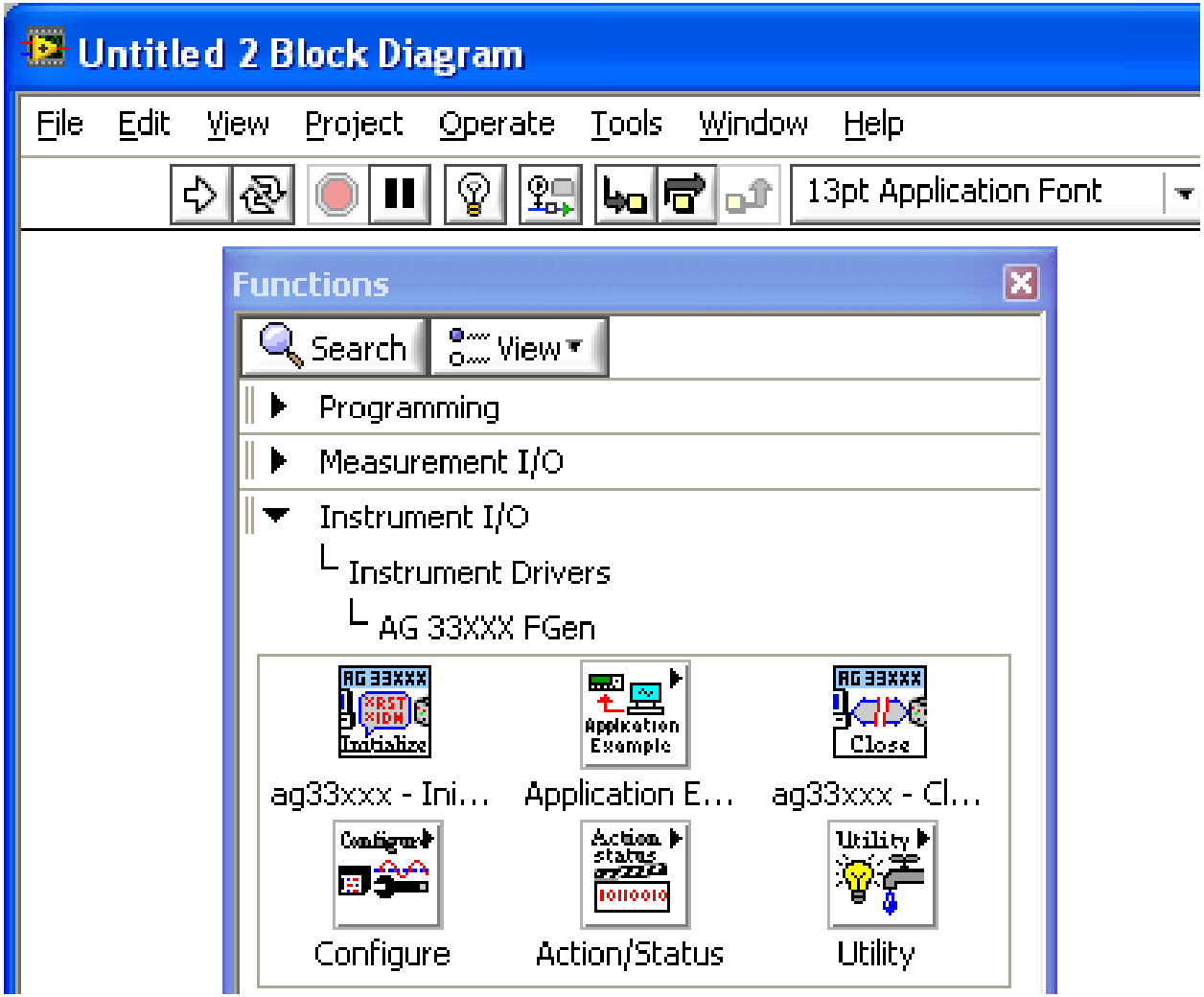
Select Drivers to Install



Install Drivers



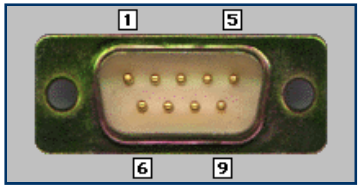
Use Installed Drivers



III. INSTRUMENT INTERFACES

Hardware Instrument Interfaces

- Serial (e.g. RS232)
- GPIB – IEEE 488
- USB 1.1/2.0
- FireWire – IEEE 1394
- Ethernet (e.g. IEEE 802.3)
- Internal Computer Buses
  - ISA/PCI/PCIe/PXI/PXIe
- Industrial Networks
  - DeviceNET, PROFIBUS, etc.



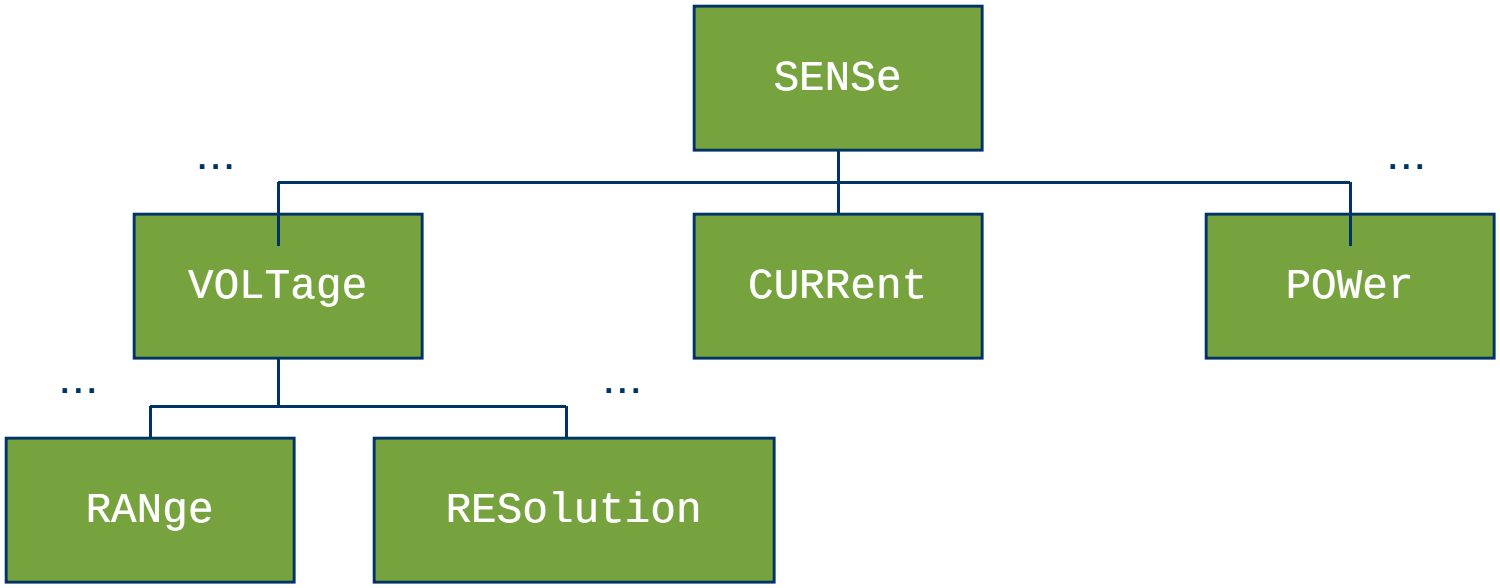
PCI EXPRESS®

FireWire

Software Instrument Interfaces

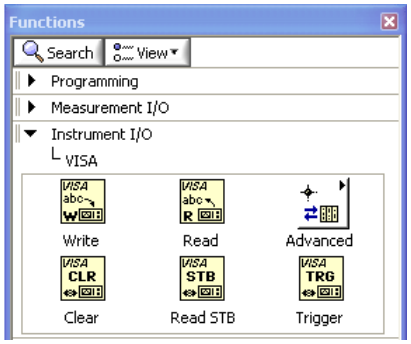
Programming With SCPI

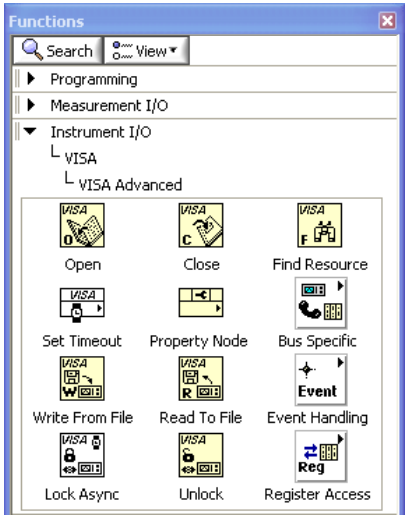
- Standard Commands for Programmable Instruments
- Set of required commands (*IDN?, *RST, *TST?, …)
- Hierarchical command structure
- Example: “set:vert_div:0.001”

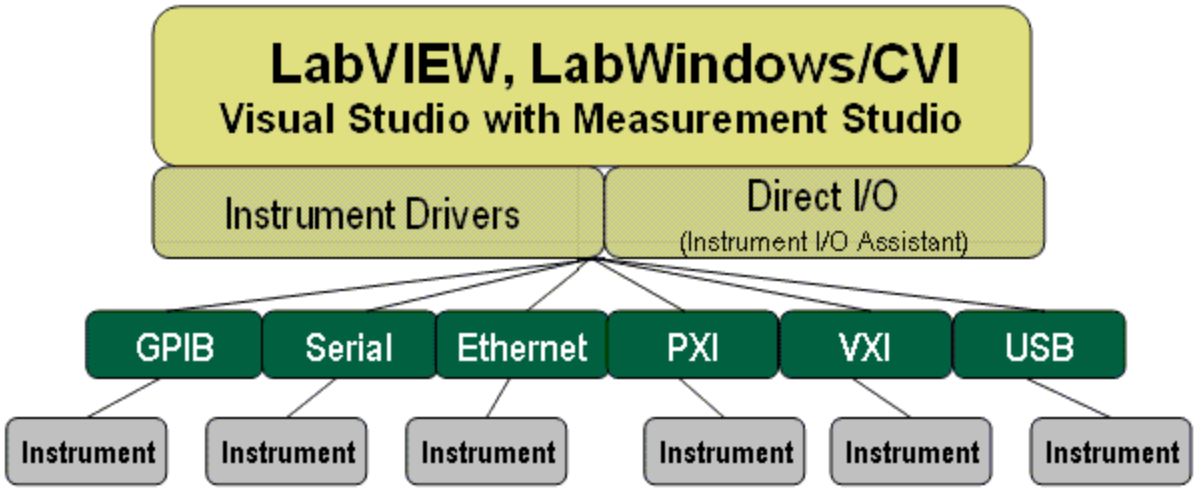


Virtual Instrument Software Architecture

- Platform and Interface independent
- Is the backbone of the IVI
- SCPI command set used

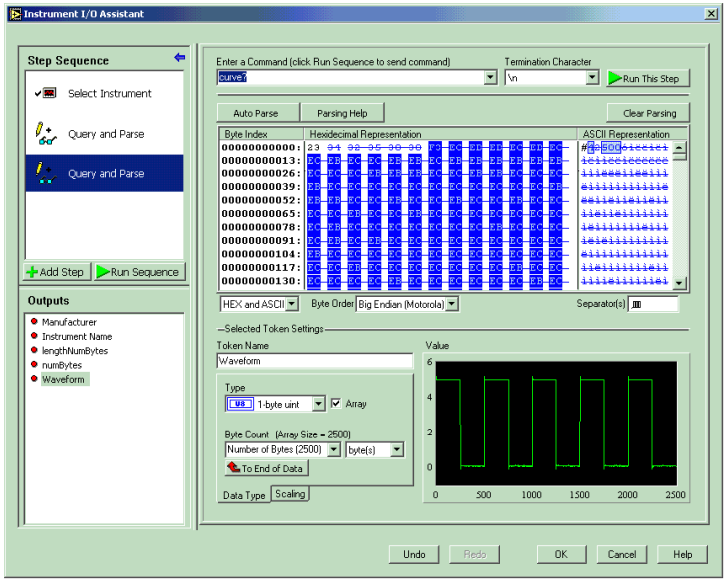




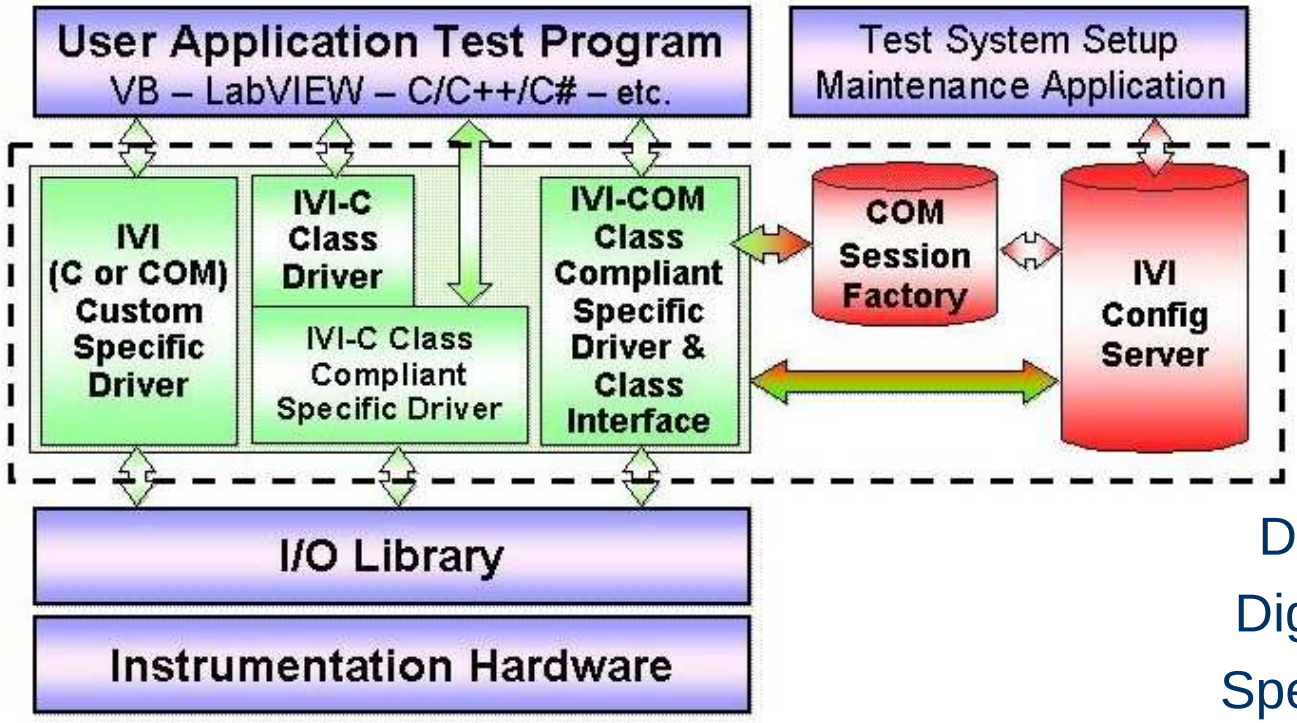


Instrument I/O Assistant – (demo)

- Interactive window uses VISA to help user communicate with SCPI commands
- Automatic and Manual parsing of received output
- Build I/O Steps, Test, and Verify
- Instr. I/O Assistant generates LabVIEW code.



Interchangeable Virtual Instrument (IVI)



IVI Classes

Switch

Power meter

Oscilloscope

DC power supply

Digital Multi-Meter

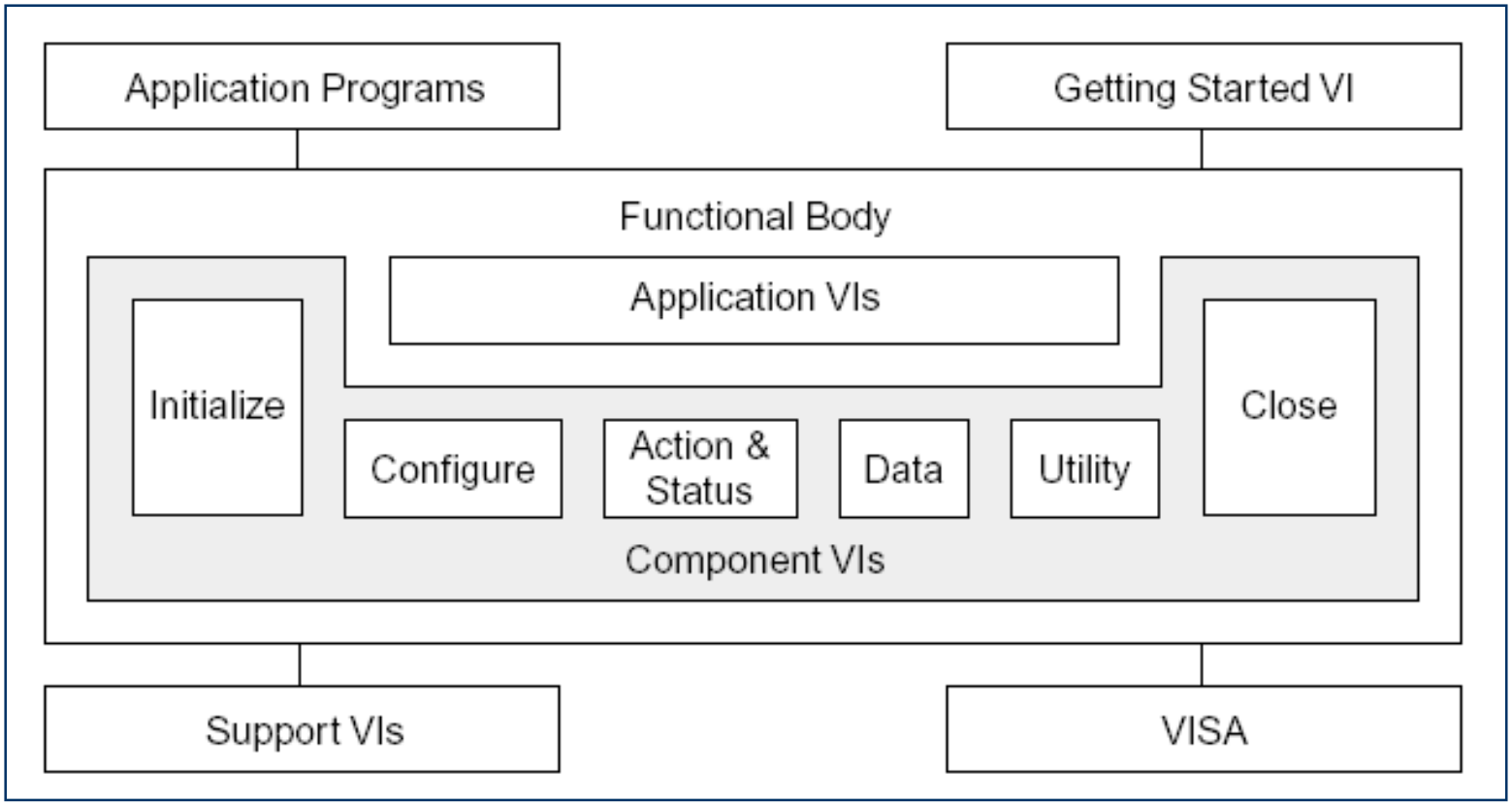
Spectrum analyzer

RF signal generator

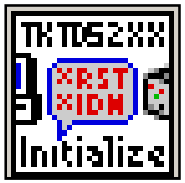
Arbitrary waveform/function generator

Courtesy of Interchangeable Virtual Instruments Foundation, Inc.

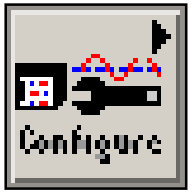
IV. Instrument Architecture



Instrument Driver VIs

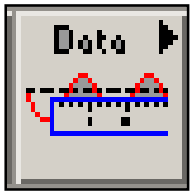


Initialize

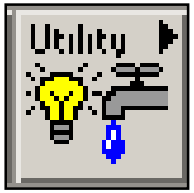


Configure

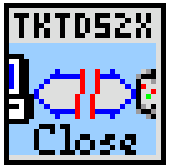
Action/Status



Data

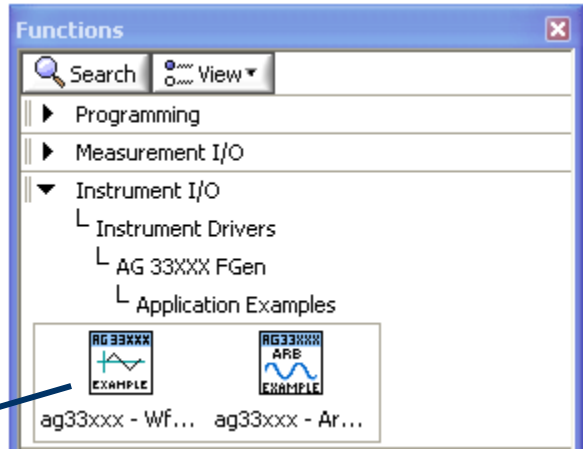


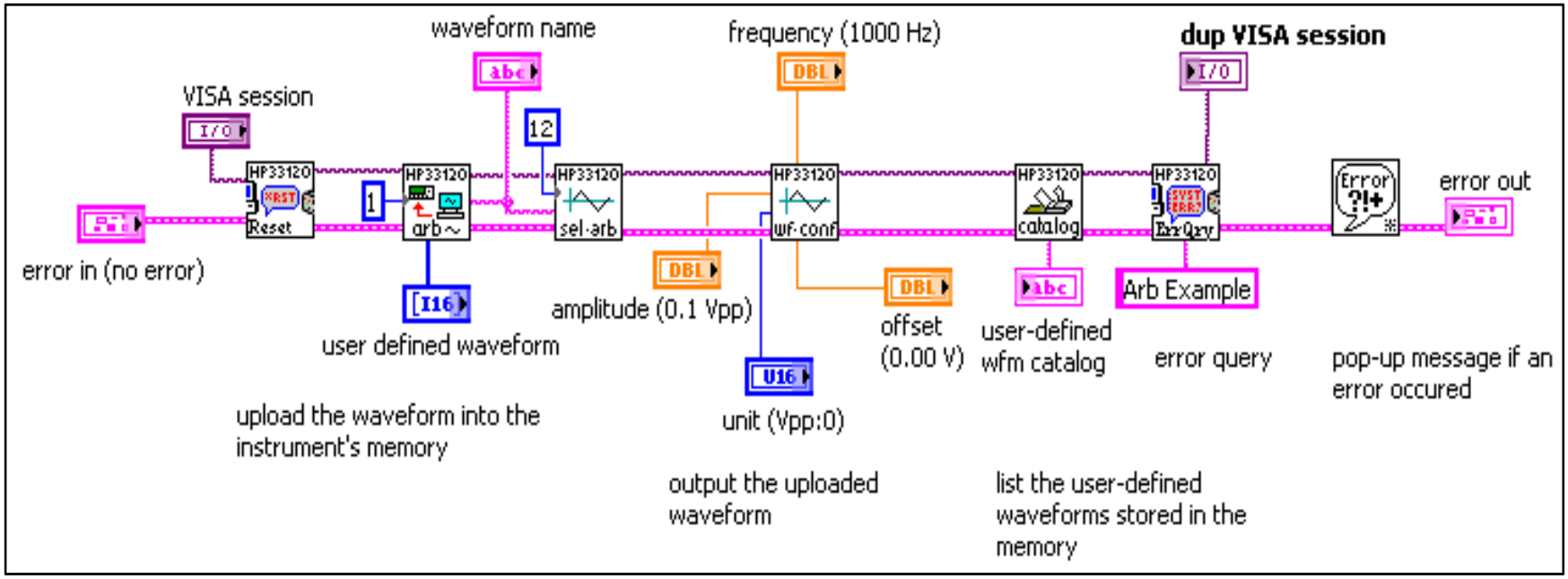
Utility



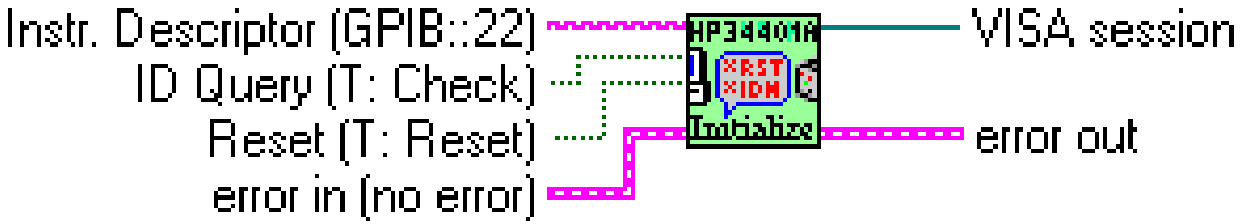
Close

Application VIs



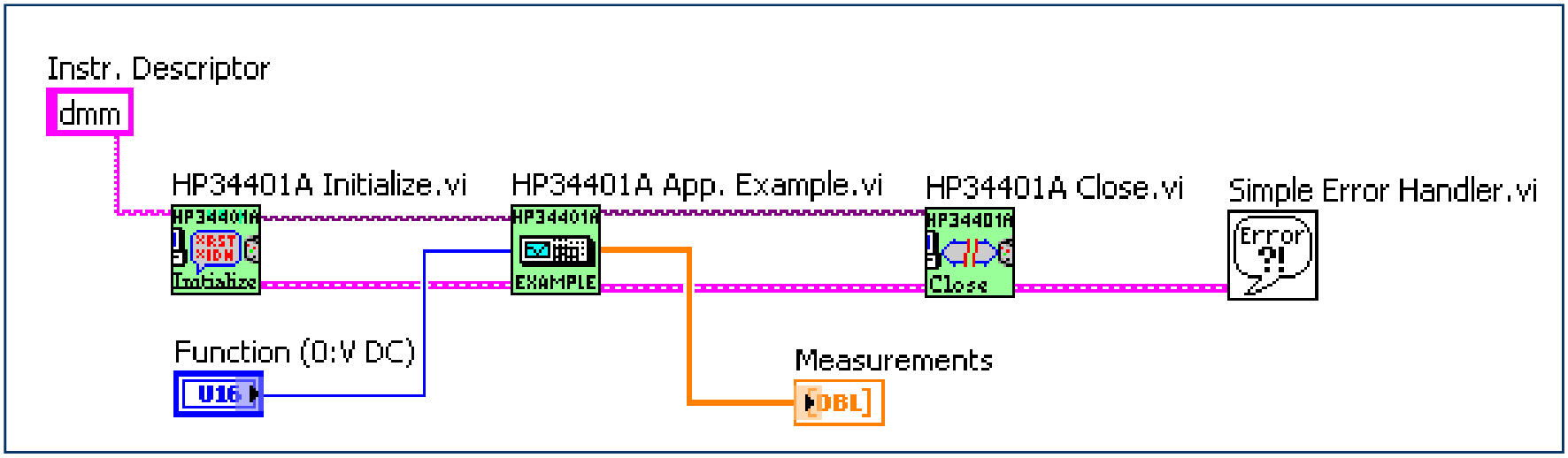


Instrument Driver Inputs & Outputs



- Instrument Descriptor
- 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 cluster

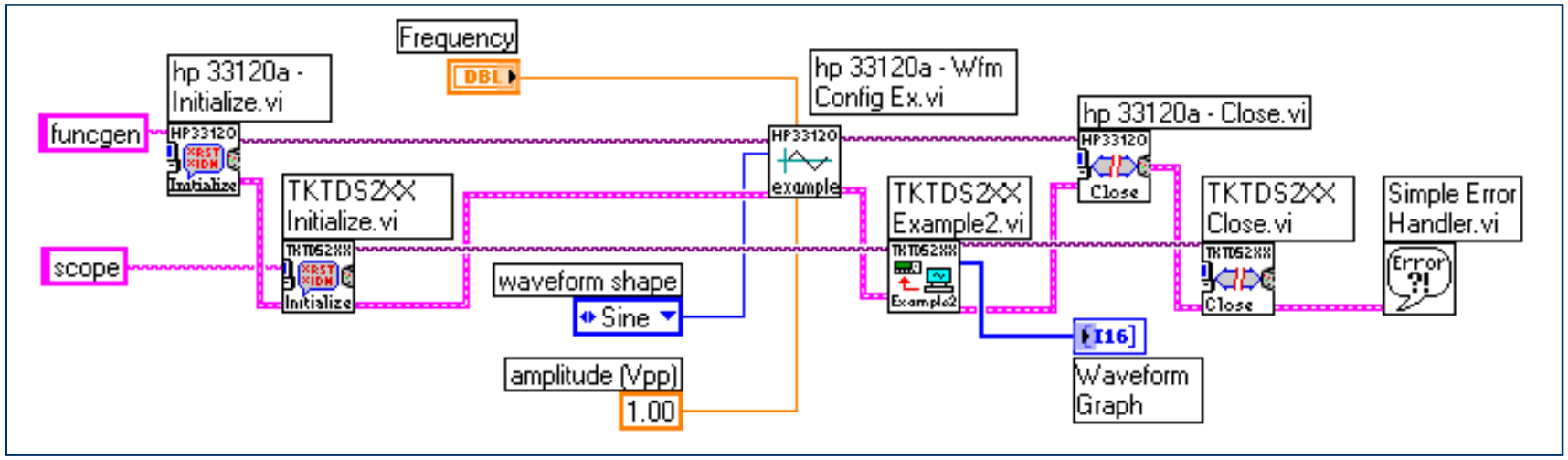
Putting It All Together



- Initialize instrument
- Perform operation(s)
- Close instrument
- Check error status

Controlling Multiple Instruments

- Similar to controlling one instrument
- Get details and make a flowchart
- Keep VISA sessions separate
- Use error clusters to define execution order



V. Driver Guidelines

1. Familiarize yourself with instrument
2. Driver Architecture and API Design
3. VIs: Names and Properties
4. Control/Indicators
5. VI Front Panels
6. Icon and Connector Panes

Driver Guidelines (continued)

7. Block Diagrams
8. Testing
9. Documentation
10. Example VIs
11. Palette Menu Files
12. Files and Documents to be Submitted

VI. Shared Library Overview

- A shared library is a software module containing executable code and data that can be call by applications or other shared libraries
- Functions and data in a shared library are loaded and linked at run time
- Shared libraries can be written in a variety of languages

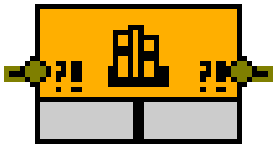
Shared Library Overview (continued)

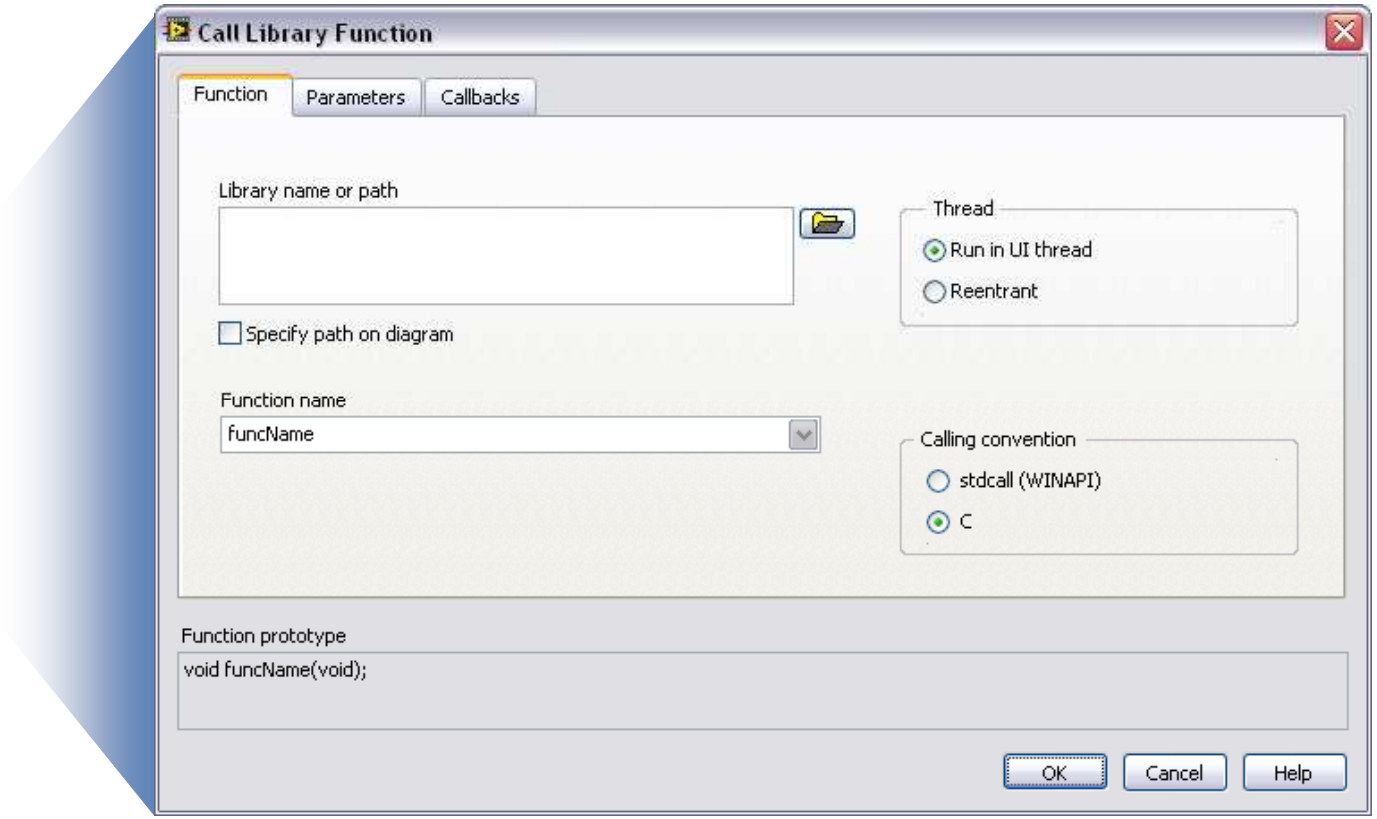
- Shared libraries expose functions and data through a standardized interface
- Most types of shared library definition are similar to function definitions in the C programming language
- Shared libraries are often called by different names depending upon the platform where they are used
  - Windows = DLLs
  - MacOS = Frameworks
  - Unix = Shared Libraries

Calling Shared Libraries

- Two methods for calling a Shared Library from LabVIEW
  - Configure functions manually using the Call Library Function node
  - Allow LabVIEW to generate code by using the Import Shared Library wizard

Call Library Function Node





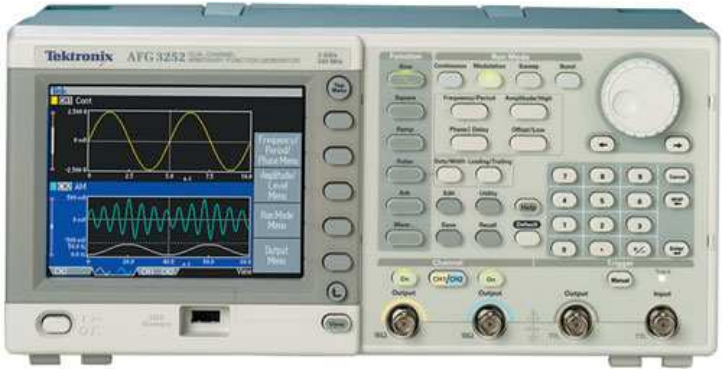
Shared Libraries Summary

- Call Library Function node to offer easy access to your shared libraries.
- To call a function in a shared library, you need to know the following:
  - The data type returned by the function
  - The calling convention used
  - The parameters to be sent to the function, their types and the order in which they must be passed
  - The location of the library on your computer
  - Whether the function can be called safely by multiple threads simultaneously
- The Shared Library Import Wizard allows you to automatically generate shared library calls if you have a header file for the library

VII. Sample Drivers – (demo)

- Tektronix AFG 3252
  - Dual Channel Arbitrary/Function Generator

- Tektronix DPO7104
  - Digital Phosphor Oscilloscope



References

- National Instruments Instrument Driver network
  http://www.ni.com/idnet
- NI LabVIEW Instrument Driver Guidelines
  http://www.ni.com/devzone/idnet/library/instrument_driver_guidelines.htm
- LabVIEW Instrument Guidelines and Information:
  LabVIEW -> Tools -> Instrumentation
- LabVIEW 8 Help > Controlling Instruments
  Available within LabVIEW and online at
  http://zone.ni.com/reference/en-XX/help/371361D-01/
- ALE System Integration website:
  http://www.aleconsultants.com
- Interchangeable Virtual Instruments Foundation, Inc.
  Your Guide to Getting Started with IVI Drivers
  http://www.ivifoundation.org/downloads/IVI_GSG_v_1.0.pdf

Terry Stratoudakis, P.E.

- B.S. and M.S. in Electrical Engineering, Polytechnic University
- NI Certified LabVIEW Developer and Certified Prof. Instructor
- New York State licensed Professional Engineer
- Former Assistant Adj. Prof. at NYC College of Technology
- Co-founder and President of ALE System Integration
- Worked for Underwriters Laboratories for six years
- Ten years LabVIEW and Test & Measurement experience
- Member of the IEEE, IEEE-LICN