IVI drivers: Slower but simpler

Andrea Clary - February 01, 2003

listing 1

The road to interchangeable instruments
SCPI vs. IVI speed study

If you've ever replaced an instrument in a test rack with a new or competitive model, you've probably run into programming problems caused by differences in command sets. Similarly, if you've ever wanted to replace an IEEE 488 interface card with a competitor's IEEE 488 card, you found that you needed to rewrite your application code because each card uses different commands. Interchangeable Virtual Instrument (IVI) drivers promise to eliminate those frustrations through a consistent programming interface.

Now that IVI drivers are becoming available, they should make programming and instrument-swapping easier, as they provide a set of standard commands for a given class of instruments. Currently, the IVI Foundation (Ref. 1) has defined instrument classes for oscilloscopes, DMMs, function generators, DC power supplies, switches, power meters, RF signal generators, and spectrum analyzers. (See "The road to interchangeable instruments," below, for the history of instrument programming that led to IVI drivers.)

In most applications, test engineers can use IVI drivers to control IEEE 488-based instruments in automated test stands. You can get IVI drivers from software suppliers or from instrument makers.

IVI drivers, built on the VXIplug&play framework (Ref. 2), circumvent low-level programming details through intuitive function calls such as "Configure Measurement" or "Read Waveform." From those function calls, IVI drivers communicate with a Virtual Instrument Software Architecture (VISA) driver. The VISA driver ensures compatibility among I/O buses—IEEE 488, VXI, RS-232, and Ethernet.

IVI drivers communicate with the VISA layer just as VXIplug&play drivers do. IVI drivers add measurement functions that you won't find in VXIplug&play drivers. The functions provide features such as state management, simulation, range checking, status query, and real-number arrays:

- **State management** keeps track of an instrument's state, which reduces redundancy in sending setup commands. Suppose you use the "Configure Measurement" command to set a DMM for two-wire ohms measurements. A subsequent command that also indicates the use of the two-wire ohms function won't resend the configuration command. With VXIplug&play drivers or with direct calls to an IEEE 488 card, you'll unnecessarily resend configuration commands to the instrument. Sending configuration commands when your instrument is already properly configured wastes bus resources and slows tests.

- **Simulation** lets you develop application code without having an instrument. That cuts test-system development time because you can begin programming while waiting for a new instrument to
arrive from the manufacturer. When simulating a measurement, an IVI driver returns a random number as the data in the instrument's selected range and function.

- **Range checking** compares a requested value against a table of legal values for a specified range. You could use range checking to verify the instrument's measurement speed or digits of resolution. You'll get an error if you try to program a 6.5-digit DMM for 7.5 digits.

- **Status query** lets you check an instrument's status. You'll find status queries useful for uncovering coding problems during test-program development. Once you've debugged your code, you should turn off this feature to increase execution speed.

- **Real-number arrays** eliminate the need to parse comma-delimited ASCII strings. Without IVI drivers, your programming language doesn't provide high-level functions for string-to-numeric conversion, so you may need to build string-parsing routines.

### IVI mechanics

![Figure 1](image)

**Figure 1** IVI drivers sit atop the VISA I/O layer, providing bus-independent instrument programming.

IVI drivers consist of two parts—a class driver and an instrument-specific driver (Figure 1). Class drivers contain function calls consistent across all instruments in an instrument class. Thus, an IVI DMM class driver will use the same function calls whether you use a Keithley 2000, an Agilent 34401A, or a Fluke 8840. You can assign names to specific resources in a test system. In Figure 2, I've assigned the name "DMM1" to an HP (Agilent) 34401A DMM. In this application, I just had to assign the instrument, its bus, and its bus address to the name DMM1. If you use an IVI driver and you change your DMM to another one that has an IVI driver, you just reassign DMM1 and install the new driver. Your application code will then control the replacement meter.

You invoke an IVI driver by making function calls from the test-application program. With the code in Listing 1, you can use any DMM that has an IVI driver, supports 4.5 digits, and has an autoranging VDC feature. In the instrument configuration, you assign the logical name such as "DMM1" to a specific instrument.

### What's the catch?

You may find disadvantages to using IVI drivers that include a learning curve, possible bugs, and slower execution speed. IVI drivers cause programs to run slower than if you use standard commands for programmable instruments (SCPI) commands (Ref. 3) sent directly to an IEEE 488 card's I/O driver. "SCPI vs. IVI speed study," below, provides some benchmark test results.
Using a configuration panel, you can assign names to instruments. Here, the name DMM1 belongs to 34401A.

Function calls use a completely different programming syntax from SCPI, so learning how to program through IVI drivers will take some effort. Because many instrument manuals provide only the SCPI command syntax, if you want to use an IVI driver, you may have to get the driver's programming details from the driver supplier. See "For more information" at the end of this article for sources of information on how to use IVI drivers.

Watch out for bugs

If you don't understand all the subtleties of an instrument's functions and controls, you could easily introduce bugs to IVI drivers. The bugs might not appear while running simple tasks such as configuring measurements, setting triggers, and collecting a single measurement value, but more advanced functions such as multiple measurements could uncover driver bugs.

For example, National Instruments provides IVI drivers for both the Keithley 2000 and Agilent 34401A. Your application code should work equally well with both instruments, making them completely interchangeable. At Keithley, we tested the drivers by taking multiple measurements with programs written in Visual Basic 6 and in LabView 6. Both programs worked perfectly with the 34401A, but both produced an error with the 2000. The IVI driver caused the error because its developer didn't account for buffering differences between the DMMs.

In contrast to the 34401A, the 2000 uses two different buffers to store readings: scratch buffer and trace buffer. The trace buffer lets you perform complex handshaking that includes limit testing, asserting the IEEE 488 bus' SRQ line, and coordinating measurement timing with external trigger signals. When you don't need this complexity, you can use the scratch buffer.

We used the NI-Spy utility to reveal that the driver was attempting to use the 2000's trace buffer but didn't set it up properly. Hence, the multipoint measurements produced a bus error. Apparently, the driver developer was unaware of instrument-specific details.

No IVI driver?

Not every instrument in production today (and certainly very few older instruments) will have IVI drivers. If you're a test-application developer or system integrator, you should probably refrain from writing your own IVI drivers for instruments that don't already have one unless you see an opportunity to market your driver.

I recommend that if you have an IVI driver, use it. If not, then you're better off not trying to write one. You can, for instance, develop an application program that communicates directly with the VISA DLL or you can write code to control an IEEE 488 instrument by sending SCPI commands directly to
an interface card's driver.

**For more information**


"Using IVI Drivers to Simulate Your Instrumentation Hardware in LabView and LabWindows/CVI," Application note 120, National Instruments, Austin, TX. [www.ni.com](http://www.ni.com).


**References**