TEST & MEASUREMENT WORLD
THE MAGAZINE FOR QUALITY IN ELECTRONICS

VXIBUS-BASED TEST

VXIplug&play Cuts System Integration Time

Jim Ware
MTS-PowerTek, Farmington Hills, MI

Soft front panels and drivers let you use the same software on many VXIbus systems.

As a system integrator, MTS-PowerTek develops custom data-acquisition and control systems, many of which use VXIbus instruments. For the past three years, we've built systems that use a Unix-based, embedded VXIbus controller to control data-acquisition instruments. Our systems also use Windows NT-based PCs to provide the user interface. These PCs communicate to the Unix controller through an Ethernet port. This year, we investigated how we could build systems in which the Windows NT-based computer controls the VXIbus instruments *and* operates as the user interface.

We were intrigued by the idea of using VXIplug&play-compliant software to build these NT-based systems. We found that VXIplug&play's soft front panels, instrument drivers, driver source code, and help files can save us hundreds of hours over developing systems using Unix. Using VXIplug&play-compliant instruments reduces the time, and hence the cost, of integrating new systems.

We also wanted to move to a single operating system because we deliver systems that use any of several communications buses between the user-interface PC and a VXIbus Slot-0 controller. Those buses include MXIbus, IEEE 488, and Grand Interconnect, a fiber-optic link from KineticSystems (Lockport, IL). VXIplug&play instrument drivers let us develop systems with any system controller while using the same applications software. We can't do that with Unix.

Get It Working

When integrating a system, we start by verifying that each instrument works on its own. Before the advent of VXIplug&play software, I'd write a sample program to verify that each module worked. I had to write a sample program for each new module type that I received.

To write the program, I'd find some sample code in the manufacturer's manual, type it into my program, compile it, and run it. Often, the code didn't work correctly the first time, and I would have to eliminate potential problems such as a bad VXIbus module, a typing error in the code, an incorrect sample program, a wrong address, or the wrong driver software.

Using a VXIplug&play soft front panel eliminates the need to write test code. Soft front panels let users test and exercise a module right after plugging it into the VXIbus mainframe. Even if VXIplug&play standards specified soft-front panels and nothing else, I'd still consider moving from Unix to Windows.

Figure 1 shows a soft front panel for a Hewlett-Packard E1413 scanning 64-channel analog-to-digital converter (ADC). This soft front panel lets a user select the signal-conditioning plug-on (SCP) and configure each of the SCP's eight channels. Here, the soft front panel lets the user select the type of input (resistance is selected in the figure) and set parameters such as excitation current and voltage range.

Soft front panels offer a less obvious benefit, too. We perform a signal checkout after assembling each system. Electricians from our electrical shop perform this check to verify that they've correctly wired the system. I've found that electrical folks often don't trust software; they usually blame the software when a system doesn't work. Because soft front panels work, we can eliminate the software as an error source. If an external

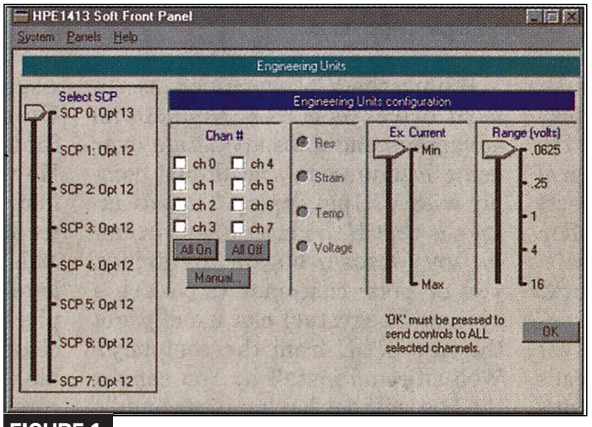

FIGURE 1. Soft front panels let you operate a VXIbus instrument without writing any code. This panel controls the engineering units for an HP E1413 ADC module.

signal doesn't check out with the soft front panel, we usually find a wiring problem.

I recently spent about 160 hours writing application code that lets customers perform basic maintenance and troubleshooting on our Unix-based VXIbus system. The maintenance included off-line functions that exercise a module to check external wiring, run module diagnostics, run self-tests, and perform calibration. Run-time functions let users view log files, check disk space, and check memory and process status.

If we were running a system with VXIplug&play-compliant products, we would not have had to write the diagnostic software. Instead, we could have used the soft front panels supplied with the VXIbus modules. Soft front panels are diagnostic tools only, though. They can't share data with other applications, and they often don't provide access to all of a module's features.

Driver Specifications

In addition to requiring that manufacturers supply soft front panels, the VXIplug&play Systems Alliance has specifications for instrument drivers. The VXIplug&play instrument drivers and a DLL called virtual instrument software architecture (VISA) let you use any programming language, regardless of which lower-level I/O driver you use—National Instruments' NI-VXI or Hewlett-Packard's standard instrument control language (SICL).

VISA is part of a software hierarchy that resides between instrument drivers and I/O driver libraries (**Fig. 2**). In effect, VISA translates commands from instrument drivers into the function calls that the I/O driver libraries require. VISA comes in the form of a Windows DLL and is supplied with any VXIplug&play instrument module. When working with VXIplug&play drivers and VISA, you can use the same applications code regardless of which communications bus connects the host PC and the Slot-0 controller.

Sitting above VISA in the software hierarchy is the VXIplug&play instrument-driver for a particular module. The instrument driver contains a library of function calls that control all of an instrument's features. VXIbus module manufacturers deliver their instrument drivers in DLL form.

Some instrument drivers also have programming shells that reside around the VXIplug&play driver DLLs. These shells let you program an instrument driver using a programming language such as LabView, LabWindows/CVI, or VEE. At MTS-PowerTek, we often write our own driver shells using C, C++, or Basic. Application programs, which are at the top of the hierarchy, make calls to the shells, which call the VXIplug&play instrument drivers, which make calls to VISA, and so on.

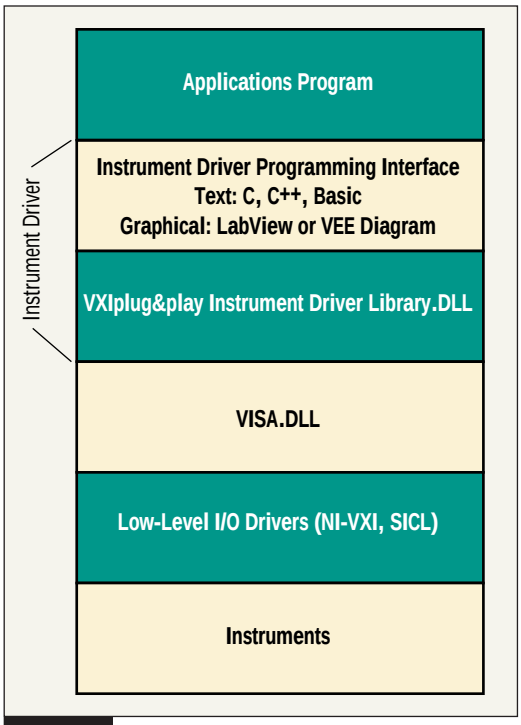

FIGURE 2. Instrument drivers take commands from applications programs, and VISA converts function calls from instrument drivers into commands that low-level I/O drivers can understand.

Having the VXIplug&play instrument driver as a DLL has both an advantage and a disadvantage over using instruments that do not comply with VXIplug&play. The advantage is that if the instrument manufacturer fixes a bug in the driver, you or your customer (if you're a system integrator) can usually get the new DLL from the company's Web site and install it. You can fix the bug without having to recompile any code.

The disadvantage is that it's possible for the system you use for development and testing to have a version of the DLL that differs from the version on your customer's machine, and this can complicate troubleshooting. You can resolve the problem by statically linking to the library, which will combine the applications program and the driver into one program.

Static linking ensures that a system integrator and user have the same instrument driver and VISA DLL. If you need to upgrade the user's driver, however, you'll have to recompile the entire application.

The VXIplug&play Systems Alliance may revise VISA from time to time. So if you use dynamic linking, you can just install a new version of an instrument driver or VISA. You don't have to recompile your applications software to upgrade to a new DLL.

Use the Latest DLL

Whenever you develop a new system, be sure to use the latest version of driver DLLs and VISA DLLs.[1] When I started exploring the capabilities of VXIplug&play by getting an IEEE 488 Slot-0 controller and VXIplug&play software for several of my modules, I was unable to get one manufacturer's soft front panel to control its module. I downloaded the new version of the VISA DLL from the company's Web site, and then the soft front panel recognized all of my modules.

In contrast to using VXIplug&play-compliant software, we often develop our own software shells when using Unix. For Unix systems, we use a RadiSys Slot-0 controller supplied by Hewlett-Packard. HP ships C-SCPI instrument drivers with each controller. Our driver shells can make calls to

Programming Comparison

Programming a Unix-based system can be more complex than programming a VXIplug&play-compliant system. When the engineers at my company build a Unix-based system, we use C-SCPI drivers and we can write function calls using high-level commands. The following code line shows a C-SCPI code sequence for reading a value from an HP E1332 counter/timer module:

```
INST_QUERY(e1332device, 'READ1?', '%lf', &value);
```

A problem occurs when we need a module that requires register-based programming. When that happens, we must program at the lower SICL level. The following code segment shows the more complex syntax that performs the same read operation with register-based code:

```
while(!(iwpeek(&statuscontrol_register)&1));
while(!(iwpeek(&statuscontrol_register)&0x80));
iwpoke(&cmdresponse_register,command);
while(!(iwpeek(&statuscontrol_register)&1));
iwpoke(&parameter_register,parameter);
```

When developing a VXIplug&play-compliant system, we never have to program at the SICL level. VXIplug&play-compliant instruments come with drivers that let us program using high-level commands similar to the C-SCPI commands that we use with Unix. We avoid the need to write register-based code, which greatly simplifies our system development.—*Jim Ware*

the drivers using the high-level C-SCPI language. For example, to set up a type J thermocouple channel on an HP E1413 ADC module, we send the following C-SCPI command to the instrument driver:

```
sense:function:temperature
 TC,J,0.0625,(@156)
```

Unfortunately, we run into problems when we need to develop an interface for VXIbus modules from manufacturers other than HP. Unless the module uses message-based programming commands, we need to develop a shell that communicates directly to HP's SICL drivers. Working at this lower level (see Fig. 2), we can no longer write instrument commands using the C-SCPI language, so writing to the SICL driver is more complex than writing a message-based command.

With Windows NT and VXIplug&play-compliant instruments, we can always write code at a level similar to C-SCPI. This standard way of making calls to instruments is what makes VXIplug&play so appealing. All card interfaces that we develop can use high-level function calls. An example VXIplug&play call to a KineticSystems V207 ADC module for reading a buffer of data looks like this:

```
retcode = ksv207_read
 (vi_handle, &dataArray);
```

where *vi_handle* identifies the VXIbus instrument and *dataArray* returns an array of data from the instrument to the program.

VXIplug&play standards define programming frameworks for Windows 3.1, Windows 95, Windows NT, HP-UX, and SunOS. Although some companies offer VXIplug&play drivers for SunOS and HP-UX frameworks, most don't. Therefore, if you want to take advantage of VXIplug&play, you must use one of the Windows frameworks.

We wanted to use the same framework regardless of which communications bus we used. We didn't want to recompile the applications code for each interface. Using Unix, we'd have to do that. If I wanted to use a VXIbus module that didn't have an HP-UX or SunOS VXIplug&play driver, I'd have to develop a register-based driver and rewrite it for each communications bus.

Using VXIplug&play instrument drivers and VISA, we now write applications in many languages and operate them through any communications bus. For example, if we develop a Windows NT application for the KineticSystems V207 module using an MXIbus connection from the PC to VXI, we can use that same program through an IEEE 488 bus connected to a Slot-0 controller, on a Grand Interconnect connection, or directly on an embedded Slot-0 controller. We don't have to recompile the applications program if we build another system using a different communications bus. Therefore, we can build multiple hardware configurations without having to maintain multiple software programs. We should be able to reuse the code we write today on any new VXIbus-to-host-PC communications bus that comes on the market.

In addition to getting drivers that work with any communications bus, we like that VXIplug&play instruments come with DLL driver source code. We also like the help files, which document all the functions in the plug&play library and include sample applications that we can paste into any application. I can't overstate the value of having both the on-line help file and the sample code when I'm developing an interface for a new module type. *T&MW*

FOOTNOTE

1. You can download VISA for Windows from several Internet sites, including *ftp.natinst.com/support/vxipnp/drivers/win32/1.2* and *ftp://fcext3.external.hp.com/dist/mxd/pc/binfiles/iol/index.html.*

Jim Ware *is the software product manager for MTS-PowerTek. He has been developing software for data-acquisition and control systems for the past 12 years. He holds a B.S. degree in computer science from the University of Dayton and is a Microsoft Certified Professional.*

Reprinted from **TEST & MEASUREMENT WORLD** June 1998