Numerous standards for instrument hardware and software must work together to form a complete automated test system.

BY LARRY DESJARDIN, CONTRIBUTING TECHNICAL EDITOR

Test engineers have used open standard interfaces since 1975, when the IEEE approved IEEE 488, commonly known as GPIB. Since then, a spectrum of test-automation standards has appeared. These standards specify interface buses, modular formats, command languages, and software components. The breadth of choices makes the task of integrating instruments that use these different formats appear daunting. Fortunately, you can work through the challenges to develop automated test systems with improved throughput, measurement quality, and data analysis.

Hardware specifications include VXI, PXI, and AXIe modular formats. The LXI standard specifies an Ethernet communication standard widely adopted by traditional box instruments, and the IVI standards specify instrument software. (The table lists the full names of the communication standards used in this article.) In this article, I will explain how to use these standards to integrate modular and traditional instruments into a single test system, and I will focus on the controller interfaces and software required.

While this plethora of standards may seem to add complexity to a system, in some ways the standards are actually converging on two major interfaces: LAN and PCIe. Although GPIB is still widely used and will remain in use for some time, traditional “box” instruments have begun migrating to LAN, using the LXI protocols that transfer commands and data. Meanwhile, the three modular standards of VXI, PXI, and AXIe are converging on PCIe as the primary interface between the controller and the instrument. The VXIbus Consortium has recently adopted the VXI 4.0 specification that allows PCIe-based communication, PXI has been enhanced by PCIe for years, and AXIe uses PCIe as its principal interface. From a control and software point of view, communicating with any of these three modular instrument standards using PCIe is very similar and is nearly transparent to the system controller.

A third class of products—proprietary modular systems from a single vendor—doesn’t conform to any of the three modular specifications. While modular in form, such systems typically behave more like traditional instruments that use LXI as an interface. For the purposes of this article, you can include them with the other traditional instruments.

Engineers must consider several factors when deciding whether to deploy traditional instrumentation, modular instrumentation, or a combination of the two. The key advantages of traditional instrumentation are the availability of nearly any type of product, the ease of use of the programming interface, and the ability to use the product as a manual bench instrument. Open modular systems typically offer smaller size, increased flexibility, and higher speed. The open-system aspect of the VXI, PXI, and AXIe standards lets you integrate products from different manufacturers within a single chassis.
For example, assume that a user has chosen to integrate two traditional LXI instruments with a PCIe-based modular chassis, either PXI or AXIe. Figure 1 shows a simplified model of integrating traditional and modular instrumentation, using LAN and PCIe interfaces. It shows a controller with LAN interfaces for the traditional instruments, and a PCIe interface for the modular chassis. This is a logical model, not necessarily a physical model. With modular systems, the controller can be installed in the chassis. For the purpose of software integration, I’ll consider the external interface model first and will return to the special case of the embedded controller later.

**Review of the two interfaces**

LXI is a set of protocols built on standard Ethernet. LXI instruments have a standard RJ-45 connector and offer 10-Mbps, 100-Mbps, and 1-Gbps Ethernet speeds. With 8 bits/byte, and some loss of bandwidth due to the Ethernet protocol, the equivalent speeds are approximately 1 Mbyte/s, 10 Mbytes/s, and 100 Mbytes/s, respectively. Though the interface on an instrument uses the same Ethernet specification as the LAN port on a computer, instruments are rarely connected directly to one another without an intervening router or switch.

PCIe is a computer expansion bus, largely designed for embedded computer peripherals and expansion slots inside a laptop or desktop computer or a rack-mounted controller. PCIe is essentially a high-speed serial replacement for the older PCI and PCI-X buses and uses shared address and data lines to all peripherals. A PCIe link can range from one to 32 lanes. Four-lane links are common in modular instrument systems, though some instrument slots may support more. A typical four-lane link offers 1-Gbyte/s transfer rates at Gen-1 (Generation 1) speeds, and twice that for Gen-2 speeds. To
connect externally, the PCIe must be buffered through a dedicated interface card.

Early PXI used the parallel PCI bus for data communication. With the advent of PCIe, the PXI standard was updated to take advantage of the higher speed and independent paths that PCIe offers. The PXI specification uses a clever architecture that lets the older PXI modules and the newer PXI Express modules (often denoted as PXIe) be integrated in a system. All AXIe systems offer PCIe as their data fabric. For the remainder of this article, I will refer to PXIe, AXIe, and VXI 4.0 products as PCIe-based products, since they are logically similar.

Message-based and register-based instruments

While PCIe delivers higher bandwidths than LXI, it is important to understand the true speed differences that result from the protocols commonly used with each interface. The differences between message-based instruments and register-based instruments create the fundamental tradeoffs between ease-of-use and speed.

Message-based instruments communicate through high-level, English-like commands sent by the controller to the instrument in a serial fashion. The most widely deployed command language is SCPI, which uses plain text for sending commands to instruments and for receiving responses. The IVI Foundation manages SCPI, which is designed so similar functions executed by different products and different vendors use the same command syntax.

Register-based instruments are completely different. A software driver is called that executes the register manipulations directly. The result is captured as binary data and directly used by the controller without conversion. Speed improvements of a factor of 50 have been demonstrated by using register-based instruments.

With few exceptions, LXI and GPIB instruments are message-based devices that use SCPI commands, and PXI and AXIe instruments are register-based devices that require software drivers. VXI instruments may be message-based or register-based. Indeed, the definition of message-based and register-based instruments first appeared in the VXI specification. For the purposes of this article, you should assume that all PCIe instruments are register based.

This difference between message-based and register-based devices is what creates the speed advantage in modular instruments. In a traditional message-based instrument, the SCPI command is sent as a set of ASCII characters, such as:

```
SCPI:.Measure:VOLTage:DC? 10.0,0.001
```

This command requests that a DC voltage measurement be made with a maximum range of 10 V with 1 mV of resolution. The instrument, however, won’t understand this English-like command. The instrument’s internal processor must parse (interpret) a command to understand what is being requested. SCPI parsing typically requires milliseconds to execute.

The measurement is then executed by an internal register that accesses the instrument’s hardware. Depending on the measurement, this may only require microseconds. The voltage measurement above can be performed in less than 100 µs, with the result in binary format. The instrument then builds the result as an ASCII number, and sends it to the controller. The controller converts the ASCII number back to binary to store or compare the result to limits. The entire process takes several milliseconds, even though the measurement only required microseconds.

For a register-based device, the process is completely different. A software driver is called that executes the register manipulations directly. The result is captured as binary data and directly used by the controller without conversion. Speed improvements of a factor of 50 have been demonstrated by using register-based instruments.

FIGURE 1. Two major industry standards now dominate instrument interfaces: LAN and PCIe. Traditional bench instruments, based on the LXI standard, are converging on LAN, while modular products are converging on PCIe.

FIGURE 2. Two LAN connections are typically needed in a hybrid test system. One controls the test instruments, while another communicates with the enterprise network. Interface cards create a Cabled PCIe connection to the modular chassis.
There is, however, an ease-of-use tradeoff. SCPI commands are easy to use and to read. They are also portable; they work in any operating system or automation environment. They avoid the need to deploy drivers, though drivers may be useful for other reasons. If measurement speed is not critical, or the measurement time is much longer than the command-interpretation latency, message-based instruments may be a good choice. For all of these reasons, many systems include a combination of traditional message-based instruments and high-speed modular instruments.

Integrating a hybrid system
The first part of integrating traditional and modular instruments in a hybrid system is the choice of interconnects and their configuration. LXI instruments will require the use of Ethernet. Note, however, that LAN networks are not all the same. It is rare to connect instruments directly onto an enterprise LAN. The data on an enterprise LAN would interfere with and slow down the real-time execution of the instrument control, while the instrument LAN would add significant traffic at the enterprise level. Typically, the LXI instruments will be placed on a completely different network than that used for the enterprise.

Thus, the controller will require two LAN interfaces—one connected to the enterprise LAN where the test plans and results are managed, and another dedicated to instrument control. An Ethernet switch is typically deployed from the controller to each instrument. This requires that a second NIC (network-interface card) be installed in the controller.

The situation is similar for PCIe-based modular systems. There is LAN communication from the controller to the enterprise, but the instrument communication occurs over PCIe. Most computers have PCIe internally. Unlike the LXI case, the instruments will be part of the same PCIe network and memory map as all the other computer peripherals, as this is where they get their speed.

The internal PCIe bus must, however, be buffered to avoid loading of these external instruments. Thus, you must install a PCIe interface card, which allows PCIe to be extended as a cable to a PXI, an AXIe, or a VXI 4.0 chassis. This electrically buffers the PCIe bus but keeps the same memory map. If a system uses multiple PCIe chassis, they will each require an additional interface card, or chassis extender. Figure 2 shows a system configured with LAN and PCIe as the communication channels. The next step involves adding the required software.

Instrumentation software layers
Instruments, whether message-based or register-based, will require I/O commands. Even a simple SCPI-based instrument will need some way for the controller to send the command to the instrument. This is where VISA comes in. VISA is an industry-standard API (application-programming interface) for communicating with instruments from a PC. Versions of VISA are available from manufacturers of interface cards. VISA also allows communication to LXI instruments and PCIe memory-mapped instruments such as PXI, AXIe, and VXI 4.0.

VISA, or an equivalent I/O library, is the minimum software needed to control message-based instruments. With VISA installed, you can program SCPI instruments using documented commands.

VISA by itself, however, is rarely enough to control PCIe instruments. You need the driver that creates the instrument functions. IVI drivers, based on standard APIs managed by the IVI Foundation, are designed to enable instrument control from PC environments.

The IVI drivers are designed to address two issues: They deliver the actual register operation that is the heart of the instrument function, and they ensure compatibility at the driver level between different types of instruments and between instruments from different manufacturers. Here, IVI leverages from SCPI by defining classes based on products types, which use APIs that look remarkably similar to SCPI commands. For instance, the equivalent IVI function call for a DMM (digital multimeter) may be:

dmm.DCVoltage.Measure (10.0, 0.001)

Because this call is done at compile time, little time is lost in the execution of the command. Command parsing isn’t required at run time.

IVI drivers come in a number of variants, such as IVI-C and IVI-COM, optimized for different environments. PXI and AXIe products are nearly ubiquitously supported using IVI-C, which can be used in nearly any Microsoft automation environment, including the Microsoft Visual Studio environments, LabView, Matlab, and Agilent Vee. There are a number of shared components that must be installed at

Table. Instrument interface glossary

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AXIe</td>
<td>AdvancedTCA eXtensions for Instrumentation</td>
</tr>
<tr>
<td>GPIB</td>
<td>General Purpose Interface Bus, IEEE 488</td>
</tr>
<tr>
<td>IVI</td>
<td>Interchangeable Virtual Instruments</td>
</tr>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>LXI</td>
<td>LAN eXtensions for Instrumentation</td>
</tr>
<tr>
<td>PCI</td>
<td>Peripheral Component Interface</td>
</tr>
<tr>
<td>PCIe</td>
<td>PCI Express</td>
</tr>
<tr>
<td>PXI</td>
<td>PCI eXtensions for Instrumentation extension</td>
</tr>
<tr>
<td>SCPI</td>
<td>Standard Commands for Programmable Instruments</td>
</tr>
<tr>
<td>VISA</td>
<td>Virtual Instrument Software Architecture</td>
</tr>
<tr>
<td>VXI</td>
<td>VME eXtensions for Instrumentation</td>
</tr>
</tbody>
</table>
SYSTEM INTEGRATION

Figure 3

FIGURE 3. A test-programming language defines and sequences the tests, making high-level calls to the instrument driver. The instrument driver calls an I/O layer that sends the commands over a LAN or accesses the instrument registers over PCIe.

Embedded controllers

Up to this point, I have described the use of an external controller interfacing with LXI instruments and a PCIe-based modular chassis. But what if the controller is embedded in the modular system? What are the advantages and disadvantages of this approach?

The major change is that the PCIe interface that was needed to communicate to the modular chassis is essentially embedded in the modular controller. Other than that, the system architecture is the same. If the controller will control LXI instruments, there will need to be two LAN ports, one for the LXI control and one for the enterprise LAN, just as in the external controller case. Embedded controllers often have a spare expansion port that can accept a NIC or Cabled PCIe card.

One advantage to the embedded controller approach is that the system can be integrated into a mechanically smaller, lighter, and more transportable system. Another advantage is that the embedded controller will have been validated to work well as an instrument controller with robust PCIe enumeration.

Enumeration is the process by which a computer locates and identifies all PCIe devices. Many computers have not been validated for enumerating the deep PCIe tree structures that often are created with modular instruments. So, some instruments, though installed in the chassis, are not identified and cannot be controlled. This situation often needs to be resolved by the computer manufacturer with a BIOS update. Embedded controllers, through their design and extensive testing, are essentially guaranteed to perform the enumeration correctly.

A third advantage is enterprise governance. Even though a distinction between form factors does not seem valid to many engineers, embedded controllers are often accepted by an enterprise's IT department as deployable assets in instrument systems, while external computers fall under the regulatory auspices of the IT department. To prevent approval delays for purchasing external controllers, some engineers simply purchase embedded controllers.

At one time, there was a speed advantage with embedded controllers because their shorter bus lengths allow faster communication to the backplane. High-speed serial standards have all but removed this advantage, as Cabled PCIe to an external controller can deliver the same total bandwidth.

Embedded controllers do have some disadvantages. The first is cost. Instrumentation control remains a niche market compared to the consumer or industrial automation marketplace. Because of this, embedded controllers cost significantly more than their more common commercial counterparts.

The second disadvantage is performance. External commercial PCs often have the highest-performing processors and architectures, rapidly obsoleting their predecessors every few months. The embedded instrument controller market does not justify that level of investment, has fewer design cycles, and typically lags commercial PCs. Because of this, the highest-performance controllers will exist as stand-alone units and offer considerable cost savings. One common compromise is to deploy an industrial rack-mounted controller, which can have very high performance at competitive prices while requiring only 1U of rack space.

Another advantage of external controllers is expandability. External controllers offer any number of expansion slots, which can be used to control other peripherals or additional LAN or PCIe interfaces.

In summary, good instrument system design can allow a user to deploy the combination of traditional and modular instruments that best suits an application. In this article, I have focused on the computer system and software challenges and have presented options for creating a hybrid system. In a future article, I will explore the challenges and options in the electrical and mechanical aspects of integrating hybrid systems.