The challenges and potential of GPIB

By Andre Hsieh
Senior Application Engineer
ADLINK Technology Inc.

The IEEE 488 standard, better known as the General Purpose Interface Bus (GPIB), is a popular interface that connects instruments to computers to form ATE. GPIB was developed initially by Hewlett-Packard and was recognized as an IEEE standard in 1978. Since then, the IEEE has released IEEE 488.1 (1978) to define the GPIB hardware specifications, including its electrical, mechanical and basic protocol parameters, and IEEE 488.2 (1987) to define related software specifications.

Revolutionary changes being applied to the PCI I/O bus, such as higher throughput and smaller mechanical footprint, have increased the popularity of legacy ISA bus and more mature PCI bus standards. These standards have significantly surpassed the RS-232 in speed. On top of these are the USB and LAN interfaces, which are proven to be faster, better performing, and more versatile. Due to cost-effectiveness and easy connectivity, current PCs are equipped with both USB and LAN interfaces. Meanwhile, after more than three decades of e nhancements and wide-ranging development, countless traditional GPIB devices now support hot-plug functionality and remote access to keep in pace with IP-based instruments that test engineers are more familiar with. Because of these parallel GPIB innovations, it will be difficult for newer and faster I/O interfaces—USB or LAN—to completely replace the standard in the ATE industry.

	PCIE-based GPIB	USB 2.0	Gigabit LAN
Theoretical bandwidth	1.8Mbytes	480Mbps	1,000Mbps
Maximum connection	20m	5m	100m
Transmission latency	Minimum	General	General
Real-time response	Best (with SRQ line)	Normal (N/A)	Better (depend on traffic)
Connector mechanism	Robust	Normal	Better (depend on traffic)
Expansion capability	Normal (needs additional slots)	Good (via hub)	Best (via Internet)
Portability	X	Best	Good
Installation cost	Good	Best	Good
Hot-swap operation	X	Good (via hub)	Good
Remote and shared access	X	X	Good

A comparison between the various standards based on key specifications.

Bridge to USB or LAN

Realizing the GPIB's strong position in this market, major instrument makers have implemented a bridge communication protocol from GPIB to USB or LAN. This move takes advantage of the flexibility and high data throughput of USB and LAN while maintaining GPIB-based instrument investments in full operation. The bridge communication protocol comes in the form of a GPIB-USB/LAN adapter, GPIB-LAN gateway and converters. This and similar technological trends indicate that GPIB is here to stay.

For hardware, GPIB's interconnectivity with USB and LAN delivers faster integration and more convenient maintenance, as it is no longer necessary to physically install an ISA/GPIB card into an available expansion slot. The collaboration thus lowers the cost and complexity of test systems.

For software, combining GPIB with USB/LAN is similarly advantageous as most mainstream operating systems are capable of monitoring the LAN or USB port status so that any newly-connected instruments are automatically detected and recognized. Highly integrated environments automatically recognize connected instruments and dispatch dedicated software resources for it, thus eliminating the need to manually search, identify and initialize connected devices.

Adopting GPIB

Before implementing ATE, the following questions need to be addressed:

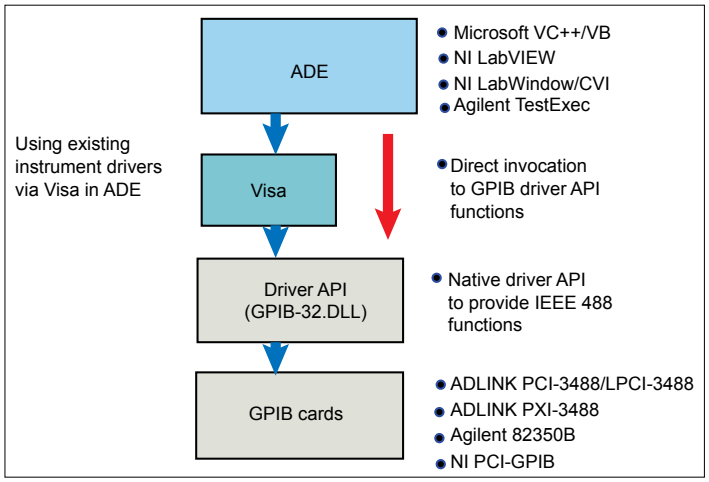
- What is your preferred physical interface connectivity solution within your instrument: GPIB, USB or LAN?
- What are the software requirements, specifications, capability and performance of your instrument?
- Based on the selected software application, what software development environment will you use to control and communicate with the instrument?

If you decide to adopt GPIB as the interface to control instruments, the next step is to determine the I/O software kits for instrument communication. These I/O software kits are regarded as a software layer positioned between the integrated application designs and the physical interface to instruments. There are two ways to create an automated test application: perform native driver API or via high-level instrument drivers.

The first method involves native driver API conventions, which are usually provided by most adapter vendors and come in the form of ANSI C functions. For users requiring a more detailed instrument control and maximum system throughput, using a driver API with SCPI string commands is highly-recommended.

For users who want to avoid complicated instrument commands, high-level instrument drivers such as VISA or IVI-COM are a suitable solution.

High-level instrument libraries provide transparent software compatibility with all types of connection interfaces and come with functions that are mostly independent from the device interface used. Whether you intend to access measurement instruments through RS-232, GPIB, USB or LAN, high-level instrument drivers do not give you the additional burden of modifying software codes when you change the communication bus type. Lastly, high-level instrument drivers give you more time to focus on



After deciding on the software I/O layer, the next consideration would be the most appropriate ADE.

software development and make user programs more scalable for later reintegration.

After deciding on the software I/O layer, the next consideration would be the most appropriate Application Development Environment (ADE). The ADE and software tool kit combination is crucial, and may directly impact the total cost of development and the application completion time. Thus, serious considerations must be made in software price and time to learn or training. You may also consider using available software development kits that accelerate test system development. Software developments may be divided into two groups: graphical and textual.

Many graphical programming environments currently cater to test and measurement engineering. An easy-to-learn graphical programming environment allows rapid creation of prototype test systems and enables the user to efficiently handle the data flow between several concurrent activities. The straightforward programming offered by graphical environments is easier compared with creating a program using textual programming. In addition, you do not need to learn complex syntax as it allows you to learn and share predefined codes easily.

Textual ADE programming is suitable for large-scale applications and for improving system throughput. However, this type requires an experienced programmer. The good news is that the runtime performance difference between graphical and textual programming has been shrinking.

FPGA alternative

Fast and reliable connectivity is the most important concern for instrument vendors and users. With the ever-increasing performance of commercial desktop and notebook computers comes an evolution of communication fundamentals between PCs and instruments. Despite the facts that GPIB is the de facto standard for connecting instruments and the PCI bus is the standard I/O interface for industrial control and measurement, new-generation USB and Ethernet are crossing the line and venturing into instrument control. It is therefore relevant to compare these standards.

Selecting which I/O interface to use is the first task in building your ATE. You may use a legacy instrument with a pure GPIB implementation or adopt a modern instrument with LAN or USB. A combination of GPIB and USB/LAN offers a comprehensive solution to meet all kinds of requirements.

The GPIB bus controller—a key GPIB component—is an ASIC. Since very few suppliers produce this component, the cost of ASIC-based GPIB is expensive. Even though there are assurances from component vendors that ASIC-based GPIB has better performance, adopting ASIC-based GPIB has not become a cost-effective solution vs. FPGA implementation, especially for prototype verification or small-scale production.

With advancements in EDA tools, the FPGA offers an alternative solution to expensive ASIC implementations. In the meantime, as more and more FPGA standards emerge and as verified IP cores become readily available from the Internet, building the GPIB protocol into FPGAs is already on the horizon and opens a promising opportunity for test and measurement applications.