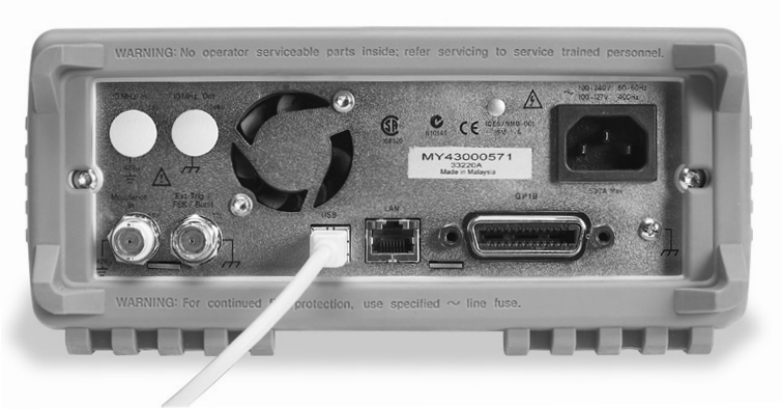
Keysight Technologies
System Developer Guide
Using USB in the Test and Measurement Environment

Application Note



KEYSIGHT TECHNOLOGIES

Introduction

This set of application notes shows you how to simplify test-system integration by utilizing open connectivity standards. The goal of these notes is to help you produce reliable results, meet your throughput requirements and stay within your budget.

Using USB in the Test and Measurement Environment, the fourth note in the series, offers a closer look at the universal serial bus (USB) as a test system connectivity option. The first note, Application Note 1465-9, Using LAN in Test Systems: The Basics, provides an introduction to the essential elements of local-area networking and the use of LAN in test systems, and the other notes in the series address important topics such as network and PC configuration. Please see page 10 for a list of the other titles in this series.

Reviewing your Connectivity Options

Whether you're setting up an ad hoc system on a lab bench or designing a permanent solution for a manufacturing line, the three best choices today for connecting modern instrumentation to computers are GPIB, LAN, and USB.

GPIB

The most commonly available programming interface on test and measurement instruments is the general-purpose interface bus (GPIB), formerly called HP-IB and also known as the IEEE-488 bus. GPIB is easy to connect, provides excellent electrical and mechanical reliability, and it offers enough throughput for a wide range of automation scenarios, making it the clear choice for most test automation needs for more than two decades. The biggest drawbacks to GPIB are the size and cost of the connectors and cables, as well as the need to install an interface card in your PC, since GPIB is not a standard PC interface.

LAN

The widespread availability of industrystandard LAN connections on PCs has recently made LAN an attractive alternative to GPIB. LAN technology combines openness, low cost, high speed, and dependable mechanical characteristics. Moreover, many offices and factories now have LANs installed, making it easy for dispersed teams to send, store, and retrieve test data. LAN is also available on many new Keysight Technologies, Inc. instruments, providing a direct connection to your PC. Given these advantages, LAN is the primary focus of this series of application notes. The biggest potential downside of using LAN for test automation is the network components such as routers. For more on these topics, please consult *Using LAN in Test Systems: Network Configuration* (AN 1465-10) and *Using LAN in Test Systems: PC Configuration* (AN 1465-11).

USB

USB is also widely available on new PCs, and its low cost and simplicity make it a great choice when you need to create a test solution quickly. This short "time to measurement" aspectof USB is particularly attractive in engineering labs and other environments where instruments are frequently moved, shared, and reconfigured. To facilitate these applications, nearly all new Keysight test and measurement instruments now offer USB connectivity. Most labs and production facilities have a mix of older and newer instruments, of course. To include legacy GPIB instruments in a USB system, you simply use a conversion device such as the Keysight 82357A USB/GPIB interface. Similarly, you can use the Keysight E5805A USB/4- port RS232 interface to connect legacy RS-232 instruments or other RS-232 devices (such as a barcode reader or a diagnostic port on the device under test) to your computer via USB. (Refer to page 5 for more information on Keysight's support for GPIB instruments in USB systems.

RS232

The oldest of the communication protocols still in common use in test and measurement applications is RS232. This simple serial interface is supported by many instruments, and it is often used for diagnostic or control ports on DUTs (device under test.) RS232's primary benefits as an instrument communication interface are its low cost and ubiquity among PCs and older instruments. Its slow speeds, inconsistent implementations, and primitive communication and discovery capabilities are its primary drawbacks. USB will eventually replace RS232 for bench users and R&D engineers as older instruments and devices are replaced. Until that time engineers will struggle with the quirks and limitations of this interface.

USB in the PC universe

Chances are you're already familiar with USB, thanks to its wide use in PC printers, scanners, cameras, and other digital devices. However, some background on USB's place in the PC universe might help you decide how and when to use USB for test and measurement.

The USB story

On the timescale of computing technology, USB has been around for quite some time: the original version of USB was introduced concurrently with Microsoft Windows 95. USB's original goals included replacing the multiple types of interfaces then in use in PCs and eliminating the complex configuration steps they sometimes required. Computers with USB 1.0 first appeared in 1996, and Windows has supported USB ever since.

The USB standard has gone through two major revisions since version 1.0. USB 1.1, introduced alongside Windows 98, took advantage of the new Plug and Play connectivity in the operating system. All you need to do in most cases is attach the connector and you're ready to go. (Nearly all PCs today, both desktop and laptop, come with built-in USB ports. You can also add USB cards to older PCs.)

This simplicity spurred rapid growth in the number of PCs and peripheral devices that offer built-in USB. However, as digital gadgets began to demand more bandwidth, the 12 Mbits/second top speed of USB 1.1 became a growing concern in some applications. USB 2.0, introduced in 2001, dramatically expanded bandwidth with speeds up to 480 Mbits/ second. USB 2.0 is backwards compatible with USB 1.1, although this can lead to some confusion about data rates, as you'll see below.

USB connections

With its intended use in consumer applications, USB is not only inexpensive but also easy to use. Connections are hot-pluggable (sometimes called hot swappable), so there's no need to power down before you add or change connections, and the PC auto discovers new devices as soon as you plug them in. And unlike GPIB, where you must assign a unique address identifier to every device in the system (and keep track of which device is where if you reconfigure the system), every USB device has an embedded serial number that the PC reads as soon as you connect it.

From a mechanical standpoint, USB 1.1 and 2.0 are identical; both use the same four-wire scheme (two power wires and twisted pair for data), and any fully compliant USB cable will work in any USB system, regardless of speed.

The theoretical maximum number of devices in a single USB system is 128 (the PC plus 127 other devices). However, you can't daisy-chain devices together as you can with GPIB. Rather, you can expand by using a hub; typical hubs provide ports for four to eight devices. To add more devices, you can daisy-chain additional hubs. Hubs can be either self-powered or bus-powered; devices that require a significant amount of power often require a self-powered hub to ensure adequate power levels.

The Benefits of USB

- Near-universal availability in today's PCs
- Hot-plugging with auto discovery for true plug-and-play
- Low cost
- Simple connection with no configuration required
- Flexible speed levels to accommodate a variety of devices
- Simultaneous connection of up to 128 devices

USB speeds

The USB 2.0 Specification encompasses all USB data transfer speeds: Hi-Speed (480 Mb/s), full-speed (12 Mb/s) and low-speed (a 1.5 Mb/s alternative designed for keyboards, mice, and other low data-rate devices). Just because a device is USB 2.0 compatible doesn't automatically mean it can operate at 480 Mb/s. The best way to verify the speed of USB devices is to look for the official USB logo. Devices that are certified to run at one of the two original USB rates, 1.5 Mb/s or 12 Mb/s, should carry a white and blue Certified USB logo (Figure 1). Devices certified to run at 480 Mb/s rates carry the red, white, and blue Certified Hi-Speed USB logo.[1]

Figre 1. USB

The speed rating of hubs in a USB system determines the operating speed of the system. For instance, if you connect Hi-Speed USB devices through a full-speed hub, the maximum speed you can expect from any of the devices is 12 Mb/s, not 480 Mb/s. To take advantage of Hi-Speed data rates, you must connect these devices through a Hi-Speed hub.

Keysight support for USB instrument connectivity

To provide users with the utmost in convenience, Keysight has committed to providing USB connectivity as a standard feature in nearly every new test and measurement instrument. In most cases, new instruments support 480 Mb/s Hi-Speed USB, which delivers greater bandwidth and lower latency (the time required to respond to programming commands) than GPIB. Those few instruments that support full-speed USB (12 Mb/s) offer bandwidth similar to GPIB with slightly higher latency.

You can also take advantage of USB with your existing GPIB instruments. The Keysight 82357A USB/GPIB Interface (Figure 2) connects GPIB instruments to a USB port on your computer, giving you a way to control up to 14 GPIB instruments from a laptop or other PC for each 82357A.

The 82357A is a fully compliant, hot pluggable USB device, so you can connect it whenever you need it, without rebooting your PC. Instruments connected via the 82357A have GPIB-style VISA and SICL addresses, just like Keysight's PCI- or older ISA-based GPIB cards, so legacy programs in systems that use these cards require no reconfiguration or code changes.

You're not limited to locally available instruments, either. With the Keysight E5813A networked 5-port USB hub, you can access remote USB devices or instruments through your standard LAN. With the E5813A connected to your PC and properly configured, those remote instruments and devices will function as though they were locally attached. Keysight also provides a USB solution for RS-232 instruments. The Keysight E5805A USB/4-port RS232 interface provides a direct connection from the USB port on your notebook or desktop PC to up to four RS232 instruments or devices.

Figure 2. Keysight 82357A USB/GPIB Interface

To simplify programming of instruments over USB connections, Keysight and other test equipment vendors co-developed the industry-standard USB Test and Measurement Class (USBTMC) and USB488 I/O protocols. These protocols, along with Keysight's E2094N IO Libraries Suite, allow you to easily switch from GPIB to USB connections without making big investments in new PC software or rewriting existing programs. Aside from address conventions, your USB instruments will look and act just as they would under GPIB control.

1. Source: USB Implementer Forum Web site, **www.usb.org**

Setting up USB instruments with the Keysight IO Libraries Suite

The Keysight E2094N IO Libraries Suite (which is now included with most Keysight instruments, T&M software products such as W1140A-VEE Keysight VEE Pro 7.0, and connectivity products such as the 82357A, E5805A and E5813A) makes USB measurement setups even simpler by automating the connection and configuration process for you. The IO Libraries Suite includes three separate direct I/O Application Programmer Interface (API) libraries so you can choose the library that works best with your development environment:

- VISA (Virtual Instrument Software Architecture, an industry standard application programming interface)
- VISA COM (a version of VISA that conforms to Microsoft's Common Object Model and IVI Foundation standards)
- Keysight SICL (included primarily to support existing test systems; VISA or VISA COM is the recommended direct I/O API for new system development)

Connecting instruments via USB

The IO Libraries include the drivers for USBTMC/USB488 devices as well as the 82357A USB/GPIB Interface, so you're ready to go as soon as you install the software. As you start plugging in instruments (or the 82357A interface), you'll be presented with a dialog box that lets you name each device with a human-readable USB alias (Figure 4). The alias capability is a helpful way to manage device names, since the standard VISA resource naming convention for USB devices can be rather cumbersome (USB0::2391::1031::MY4 3000786::0::INSTR, for example). The alias capability also enables the same test system software to work on multiple automated test systems, provided the same alias names are used, such as the alias "DMM" for a voltmeter. And if you have an existing program that communicates with an instrument over a GPIB or other non-USB interface, you can create a VISA alias that looks like a GPIB address, such as "GPIB1::23::INSTR" and the program will function as though it were still communicating over a GPIB interface.

Additional software included with the Keysight E5805 and E5813A works with the Keysight IO Libraries to provide the same flexibility for RS-232 instruments and remote USB instruments. These instruments and devices appear to be local to the PC and can be aliased as well.

To verify the connection with each instrument, simply launch the Keysight Connection Expert, the configuration utility in the new Keysight IO Libraries Suite 14 release. Refresh the list of instruments if your instrument is not already displayed, then choose "Verify This Instrument". You can also launch an interactive I/O session with the instrument and send the *IDN? command, the standard instrument identification query in the Standard Commands for Programmable Instruments (SCPI) command set. The instrument will respond by identifying its manufacturer, model number, serial number, and firmware revision.

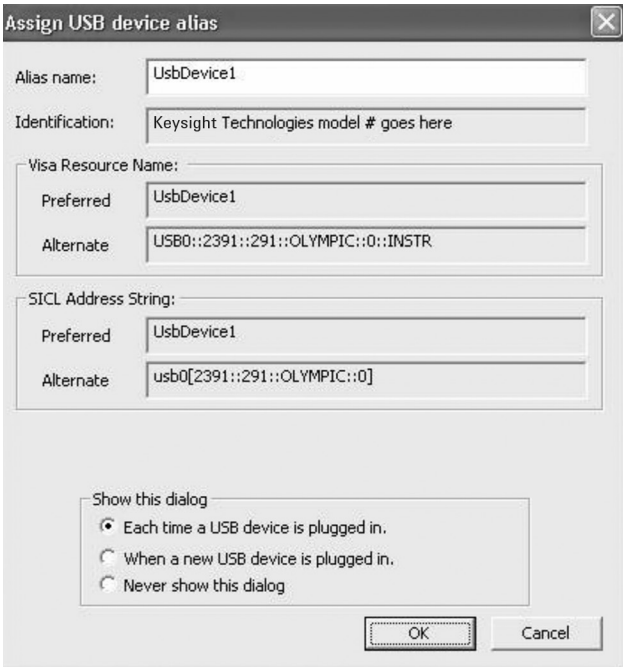


Figure 4. Example of the connection dialog in the Keysight IO Libraries.

Communicating with USB-connected devices

You don't need to worry about the details of a USB connection, so most programs written to talk to GPIB devices work with USB-connected devices without modification. However, if your program uses low-level commands that affect the entire GPIB bus (such as through a VISA session like GPIB::INTFC), you may encounter some unexpected results. USB devices are optimized for modern instrument communication, which discourages lower-level, error-prone interface manipulation operations. Consult the documentation for the instrument or I/O adapter for any limitations.

As mentioned earlier, an instrument connected via a USB cable acts like an instrument connected over a GPIB bus, aside from a different I/O address. Here's some example C code that communicates with a USB-connected instrument, either natively or with the E5813A networked 5-port USB hub:

```
#include <iostream>
#include <tchar.h>
#include <stdio.h>
#include "visa.h"

#pragma comment(lib, "visa32.lib") /* include the visa32.lib import library */

/* Error-checking routine */
void CHECKERROR(ViSession vi, ViStatus status)
{
      char desc[256];
      ViStatus err = 0;
      if (status < 0)
      {
            err = viStatusDesc(vi, status, desc);
            fprintf(stderr, desc);
            viClose(vi);
            _exit(status);
      }
}

int _tmain(int argc, _TCHAR* argv[])
{
      char idnResult[256];
      ViSession rm = 0, funcGen = 0;
      ViStatus err = 0;
      viOpenDefaultRM(&rm);
      err = viOpen(rm, "FuncGen", VI_NO_LOCK, 0, &funcGen);
      CHECKERROR(rm, err);
      err = viPrintf(funcGen, "*IDN?\n");
      CHECKERROR(funcGen, err);
      err = viScanf(funcGen, "%t", idnResult);
      CHECKERROR(funcGen, err);
      printf("The *IDN? string is %s", idnResult);
      viClose(funcGen);
      viClose(rm);
      return 0;
}
```

Similarly, the 82357A USB/GPIB interface looks and acts like a PCI/ GPIB adapter for typical instrument communication, so instruments connected to it have GPIB-style address names and act like any other GPIB-connected instruments. The source code is exactly the same as above, with the exception that the instrument address would be something like “GPIB0::23::INSTR” instead of “FuncGen”.In an RS-232 scenario, the E5805A USB/4-port RS-232 interface will look like a standard RS-232 port on your PC, and instruments connected to it will have RS-232-style address names and act like other RS-232-connected instruments:

```
/* Same header and error-handling code as above... */

/* Do a simple *IDN? Instrument Identification Query */
int _tmain(int argc, _TCHAR* argv[])
{
      char idnResult[256];
      ViSession rm = 0, dmm = 0;
      ViStatus err = 0;
      viOpenDefaultRM(&rm);
      err = viOpen(rm, “ASRL1::INSTR”, VI_NO_LOCK, 0, &dmm);
      CHECKERROR(rm, err);
      /* don’t bother checking errors for these, nothing will happen until communication is attempted
*/
      err = viSetAttribute(dmm, VI_ATTR_ASRL_PARITY, VI_ASRL_PAR_NONE);
      err = viSetAttribute(dmm, VI_ATTR_ASRL_BAUD, 9600);
      err = viSetAttribute(dmm, VI_ATTR_ASRL_DATA_BITS, 8);
      err = viSetAttribute(dmm, VI_ATTR_ASRL_STOP_BITS, VI_ASRL_STOP_ONE);
      err = viSetAttribute(dmm, VI_ATTR_ASRL_FLOW_CNTRL, VI_ASRL_FLOW_DTR_DSR);
      /* clear out any old data and prepare the instrument */
      err = viFlush(dmm, VI_IO_IN_BUF_DISCARD | VI_IO_OUT_BUF_DISCARD);
      CHECKERROR(dmm, err);
      err = viPrintf(dmm, “*CLS\n”);
      CHECKERROR(dmm, err);
      /* do the identification query */
      err = viPrintf(dmm, “*IDN?\n”);
      CHECKERROR(dmm, err);
      err = viScanf(dmm, “%T”, idnResult);
      CHECKERROR(dmm, err);
      printf(“The *IDN? string is %s”, idnResult);

      viClose(rm);
      return 0;
}
```

Glossary

Adapter – the LAN card and connector that provides an electrical interface to the network

Bridge – a LAN device that connects segments of a network

DHCP – dynamic host configuration protocol; a method of automatically obtaining an IP address for a LAN-connected device (e.g., PC, router, instrument, etc.)

DMZ – De-militarized zone; a firewall configuration that helps secure the private LAN

DNS – domain name server; maps specific names to IP addresses, enabling use of names in place of IP addresses in test programs

DUT – device under test; the component, subassembly or product to be measured by the test system

Ethernet – a specific LAN technology that is the dominant implementation of the physical and data link layers; also known as IEEE 802.3

Firewall – a hardware device or software program (or combination) that protects a computer network from unauthorized access

Gateway – a hardware device that connects devices that use different standards and protocols (e.g., LAN to GPIB)

GPIB – General Purpose Interface Bus; the dominant 8-bit parallel I/O connection for test equipment and test systems

HP-IB – Hewlett-Packard Interface Bus; another name for GPIB

Hub – a multi-port LAN device that connects multiple devices together, usually in a star topology

IP – Internet protocol; requires an address to communicate

LAN – local area network

NAT – network address translation; maps private addresses to one or more public addresses to enable access to an intranet or the Internet

RS232 – a legacy low-speed serial interface that is slowly being replaced by USB

Router – a LAN device that joins multiple networks and enables creation of small, private networks

Subnet – a group of connected network devices; used to partition networks into segments for easier administration

Subnet mask – a setting that accompanies an IP address and defines the boundaries of a subnet

Switch – a LAN device that connects multiple devices to a single LAN line; however, unlike a hub, it preserves full network bandwidth to each device

TCP/IP – Transfer Control Protocol and Internet Protocol; the two standards that provide the data communication foundation of the Internet

USB – Universal Serial Bus; designed to replace the RS-232 and RS-422 serial buses used in PCs

Related literature

The other notes in this series provide additional information about the successful use of LAN in test systems:

- *Using LAN in Test Systems: The Basics*, Application Note (pub no. 5989-1412EN) http://literature.cdn.keysight.com/5989-1412EN.pdf
- *Using LAN in Test Systems: Network Configuration*, Application Note (pub no. 5989-1413EN) http://literature.cdn.keysight.com/5989-1413EN.pdf
- *Using LAN in Test Systems: PC Configuration*, Application Note (pub no. 5989-1415EN) http://literature.cdn.keysight.com/5989-1415EN.pdf
- U*sing SCPI and Direct IO vs. Drivers*, Application Note
- *Using LAN in Test Systems: Applications*, Application Note

Other Keysight application notes provide additional hints that can help you develop effective test systems:

- *Creating a Wireless LAN Connection to a Measurement System,* Application Note (pub no. 5988-7688EN) http://literature.cdn.keysight.com/5988-7688EN.pdf
- *Introduction to Test-System Design,* Application Note (pub. no. 5988-9747EN) http://literature.cdn.keysight.com/5988-9747EN.pdf
- *Computer I/O Considerations,* Application Note (pub. no. 5988-9818EN) http://literature.cdn.keysight.com/5988-9818EN.pdf
- *Understanding Drivers and Direct I/O,* Application Note (pub. no. 5989-0110EN) http://literature.cdn.keysight.com/5989-0110EN.pdf
- *Choosing Your Test System Software Architecture*, Application Note (pub no. 5988-9819EN) http://literature.cdn.keysight.com/5988-9819EN.pdf
- *Choosing Your Test-System Hardware Architecture and Instrumentation,* Application Note (pub. no. 5988-9820EN) http://literature.cdn.keysight.com/5988-9820EN.pdf
- *Understanding the Effects of Racking and System Interconnections,* Application Note (pub. no. 5988-9821EN) http://literature.cdn.keysight.com/5988-9821EN.pdf
- *Maximizing System Throughput and Optimizing Deployment,* Application Note (pub. no. 5988-9822EN) http://literature.cdn.keysight.com/5988-9822EN.pdf
- *Operational Maintenance,* Application Note (pub. no. 5988-9823EN) http://literature.cdn.keysight.com/5988-9823EN.pdf

This document was formerly known as Application Note 1465-12

Published in USA, December 5, 2017
5989-1417EN
www.keysight.com