

17

Testing and Debugging

In addition to the chip-specific development boards and debugging software described in Chapter 6, a variety of other hardware and software tools can help in testing and debugging USB devices and their host software. This chapter introduces tools available from the USB-IF and other sources. I also explain what's involved in passing the tests that required for devices and drivers to earn the Certified USB logo and the Windows logo.

Tools

Without a doubt the most useful tool for USB device developers is a protocol analyzer, which enables you to monitor USB traffic and other bus events. The analyzer collects data on the bus and decodes and displays the data you request in user-friendly formats. You can watch what happened during enumeration, detect and examine protocol and signaling errors, view data transferred during control, interrupt, bulk, and isochronous transfers, and focus on any aspect of a communication you specify.

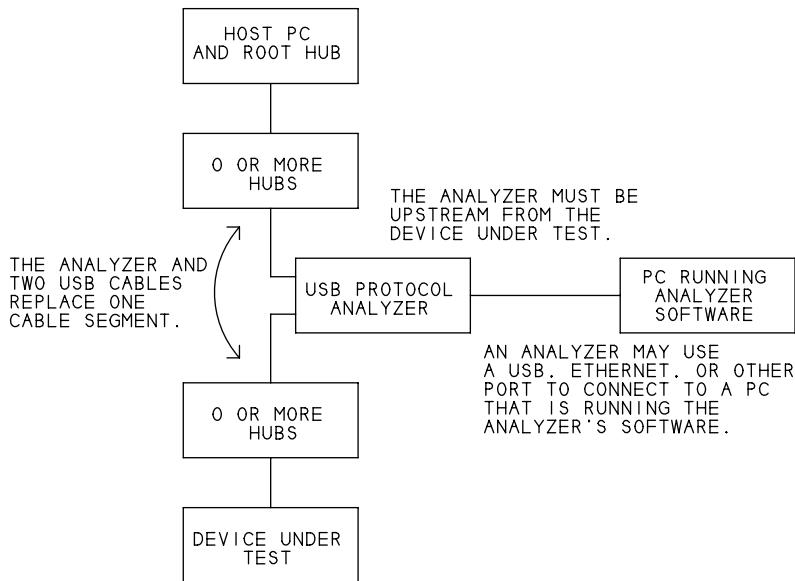


Figure 17-1: A hardware protocol analyzer monitors traffic between a device under test and the device’s host. An interface to a PC (or logic analyzer) enables viewing the captured data.

A hardware analyzer is a combination of hardware and software, while a software analyzer consists only of software that runs on the device’s host computer. The capabilities of the two types have much overlap, but each can also record and display information that isn’t available to the other type.

Another useful tool is a traffic generator, which emulates a host or device and offers precise control over what the emulated host or device places on the bus.

Hardware Protocol Analyzers

A hardware protocol analyzer includes a piece of equipment that captures the signals in a cable segment without affecting the traffic in the segment. The analyzer connects in a cable segment upstream from the device under test (Figure 17-1). To enable viewing the captured traffic, the analyzer has another connection to a PC or logic analyzer. A connection to a PC can be

via USB or another port type such as Ethernet. A few analyzers instead connect to logic analyzers from Agilent or Tektronix.

With a hardware analyzer, you can see the data in the cable down to the individual bytes that make up each packet. There's no question about what the host or device did or didn't send. For example, if the host sends an IN token packet, you can quickly see whether the device returned data or a NAK. You can view the packets in every stage of a control request. Time stamps enable you to see how often the host polls an endpoint.

Analyzers are available from a variety of vendors and with a range of prices. Ellisys' USB Explorer 200 is a relatively inexpensive yet very serviceable analyzer that supports all three bus speeds. In this chapter, I use the Explorer to illustrate the kinds of things you can do with an analyzer. Vendors are always updating and improving their products, so check for the latest information when you're ready to buy.

The Hardware

To use the Explorer, you must have two USB host controllers available. One communicates with the Explorer, and the other controls the bus being monitored. Both host controllers can be in the same PC, but for best performance, Ellisys recommends using two PCs.

The Explorer's back panel has a USB receptacle that connects to the PC that is running the Explorer's software. The PC detects the Explorer as a USB device that uses a vendor-specific driver provided by Ellisys.

Two USB receptacles on the front panel connect the analyzer in a cable segment upstream from the device being tested. One cable connects to the device being tested or a hub upstream from the device. The other cable connects to the host's root hub or another hub upstream from the analyzer.

The analyzer's circuits must capture the traffic as unobtrusively as possible. The host and device should detect no difference in the bus traffic when the analyzer is present. The two cables on the front panel and the analyzer's electronics must emulate an ordinary cable segment of 5 meters or less (3 meters or less for a low-speed segment). For these cables, Ellisys recommends using cables whose lengths together total 3 meters or less.

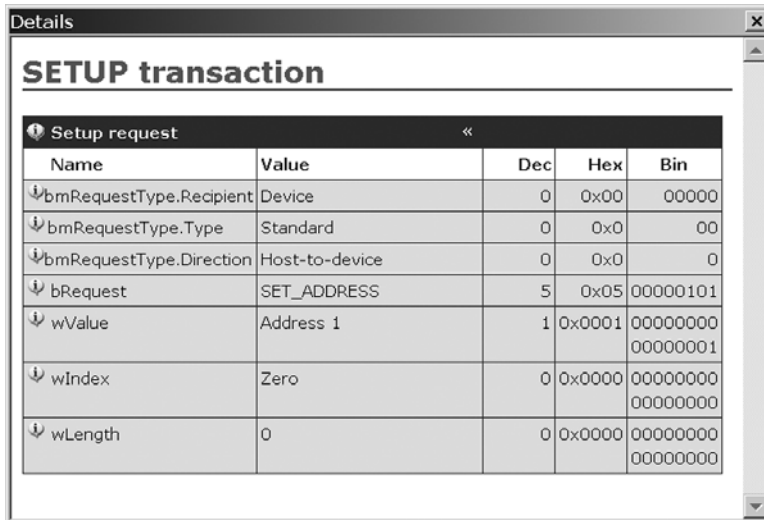
Item	Device	Endpoint	Status	Speed	Comment	Time
Enter text here	Ent...	Enter ...	Ent...	Ente...	Enter text here	Enter text here
■ ■ ■ Reset (4.0 s)						0.000 000 000
⏸ Suspended (159.2 ms)						3.954 854 458
⏸ Reset (11.0 ms)						4.114 037 395
⊞ GetDescriptor (Device)	0 (1)	0	OK	FS	8 bytes (12 01 00 02 00 00 00 08)	4.189 013 687
⊞ → SETUP transaction	0 (1)	0	ACK	FS	8 bytes (80 06 00 01 00 00 40 00)	4.189 013 687
→ SETUP packet	0	0		FS		4.189 013 687
→ DATA0 packet				FS	8 bytes (80 06 00 01 00 00 40 00)	4.189 016 750
← ACK packet			ACK	FS		4.189 025 333
⊞ ← IN transaction	0 (1)	0	ACK	FS	8 bytes (12 01 00 02 00 00 00 08)	4.190 013 520
→ IN packet	0	0		FS		4.190 013 520
← DATA1 packet				FS	8 bytes (12 01 00 02 00 00 00 08)	4.190 016 625
→ ACK packet			ACK	FS		4.190 025 479
⊞ → OUT transaction	0 (1)	0	ACK	FS	No data	4.192 013 250
→ OUT packet	0	0		FS		4.192 013 250
→ DATA1 packet				FS	No data	4.192 016 312
← ACK packet			ACK	FS		4.192 019 583
⏸ Reset (11.0 ms)						4.198 025 750
⊞ SetAddress (1)	0 (1)	0	OK	FS	No data	4.264 005 604
⊞ → SETUP transaction	0 (1)	0	ACK	FS	8 bytes (00 05 01 00 00 00 00 00)	4.264 005 604
→ SETUP packet	0	0		FS		4.264 005 604
→ DATA0 packet				FS	8 bytes (00 05 01 00 00 00 00 00)	4.264 008 666
← ACK packet			ACK	FS		4.264 017 270
⊞ ← IN transaction	0 (1)	0	ACK	FS	No data	4.265 005 375
→ IN packet	0	0		FS		4.265 005 375
← DATA1 packet				FS	No data	4.265 008 458

Figure 17-2: Ellisys’ USB Explorer 200 protocol analyzer includes Visual USB application software for viewing captured data. This example shows transactions and other events that occurred when a device was attached downstream from the analyzer

The Software

Ellisys’ Visual USB Analysis Software enables you to start and stop data logging and to save, view, and print the results. Figure 17-2 shows data captured by an analyzer. You can specify the amount, type, and format of data the displayed. For less detail, you can elect to hide the individual packets, repeated NAKs, and other information. Filters enable you to select the precise data to display. You can specify criteria such as a device or devices, endpoints, speeds, status codes, and control requests. The software displays only the traffic that meets the criteria you specify.

A Details pane provides more information about a request, transaction, packet, or other item in a row in the application’s main window (Figure 17-3). A Data pane displays the individual bytes in hexadecimal and ASCII.



The screenshot shows a window titled 'Details' with a sub-header 'SETUP transaction'. Below this is a table with the following data:

Setup request				
Name	Value	Dec	Hex	Bin
bmRequestType.Recipient	Device	0	0x00	00000
bmRequestType.Type	Standard	0	0x0	00
bmRequestType.Direction	Host-to-device	0	0x0	0
bRequest	SET_ADDRESS	5	0x05	00000101
wValue	Address 1	1	0x0001	00000000 00000001
wIndex	Zero	0	0x0000	00000000 00000000
wLength	0	0	0x0000	00000000 00000000

Figure 17-3: The Details pane in Ellipsis' Visual USB software has more information about a request, transaction, packet, or other event.

You can also search for specific items, including events, token-packet types, traffic to and from a specific device or endpoint, and data.

Additional software modules add support for triggering on events, decoding class-specific information, and exporting captured data in text, XML, and other formats.

Software Protocol Analyzers

A software-only protocol analyzer runs on the host computer of the device being tested. You can view traffic to and from any device that connects to any of the computer's host controllers.

A software analyzer can display driver information that a hardware analyzer can't access. As Chapter 8 explained, Windows drivers communicate with USB devices using I/O Request Packets (IRPs) that contain USB Request Blocks (URBs). A software analyzer can show the IRPs and URBs that a driver has submitted and the responses received from a device.

The screenshot shows the SourceUSB application in Capture Mode. The main window displays a list of USB I/O requests. The details pane shows information for record 113, which is a BULK_OR_INTERRUPT_TRANSFER request.

Type	#	Request	I/O	Elapsed Sec.	DO	Cfg:lfc:Epa	Irp Request	Irp Status
URB	113	BULK_OR_INTERRUPT_TRANSFER	IN	270.10560309	'USBPDO-15'	01:00:81	0x82ac8008	SUCCESS
URB	114	BULK_OR_INTERRUPT_TRANSFER	IN	270.10566455	'USBPDO-15'	01:00:81	0x82ac8008	NOT_SUPPORTED
URB	115	BULK_OR_INTERRUPT_TRANSFER	IN	270.11360829	'USBPDO-15'	01:00:81	0xff95f008	SUCCESS
URB	116	BULK_OR_INTERRUPT_TRANSFER	IN	270.11367534	'USBPDO-15'	01:00:81	0xff95f008	NOT_SUPPORTED
URB	117	BULK_OR_INTERRUPT_TRANSFER	IN	270.12160121	'USBPDO-15'	01:00:81	0x82ac8008	SUCCESS
URB	118	BULK_OR_INTERRUPT_TRANSFER	IN	270.12166490	'USBPDO-15'	01:00:81	0x82ac8008	NOT_SUPPORTED
URB	119	BULK_OR_INTERRUPT_TRANSFER	IN	270.12962458	'USBPDO-15'	01:00:81	0xff95f008	SUCCESS
URB	120	BULK_OR_INTERRUPT_TRANSFER	IN	270.12968687	'USBPDO-15'	01:00:81	0xff95f008	NOT_SUPPORTED
URB	121	BULK_OR_INTERRUPT_TRANSFER	IN	270.13759710	'USBPDO-15'	01:00:81	0x82ac8008	SUCCESS
URB	122	BULK_OR_INTERRUPT_TRANSFER	IN	270.13766163	'USBPDO-15'	01:00:81	0x82ac8008	NOT_SUPPORTED
URB	123	BULK_OR_INTERRUPT_TRANSFER	IN	270.14560314	'USBPDO-15'	01:00:81	0xff95f008	SUCCESS
URB	124	BULK_OR_INTERRUPT_TRANSFER	IN	270.14566935	'USBPDO-15'	01:00:81	0xff95f008	NOT_SUPPORTED
URB	125	BULK_OR_INTERRUPT_TRANSFER	IN	270.15360137	'USBPDO-15'	01:00:81	0x82ac8008	SUCCESS
URB	126	BULK_OR_INTERRUPT_TRANSFER	IN	270.15366506	'USBPDO-15'	01:00:81	0x82ac8008	NOT_SUPPORTED
URB	127	BULK_OR_INTERRUPT_TRANSFER	IN	270.16159456	'USBPDO-15'	01:00:81	0xff95f008	SUCCESS
URB	128	BULK_OR_INTERRUPT_TRANSFER	IN	270.16165630	'USBPDO-15'	01:00:81	0xff95f008	NOT_SUPPORTED

URB Field	Value
URB Length	72 bytes
URB Status	USB_STATUS_SUCCESS
TransferFlags	IN, SHORT_TRANSFER_OK
TransferBufferLength	3 bytes
TransferBuffer	Show/Hide Databar

Field	Value
InterfaceNumber	0
EndpointAddress	0x81 (IN,1)
PipeType	Interrupt
Interval	10
PipeHandle	82c54cdh

Figure 17-4: SourceUSB's application shows USB I/O requests at a host computer. These requests are for mouse communications.

But a software analyzer can't show anything that the host-controller or hub hardware handles on its own. For example, the analyzer won't show how many times an endpoint NAKed a transaction before returning an ACK or the precise time a transaction occurred on the bus.

Some software analyzers use a filter driver that loads when the operating system loads the driver for the device being monitored. Because the filter driver isn't loaded until the host has enumerated the device, the analyzer can't show the enumeration requests and other events that occur at device attachment.

Sourcequest, Inc.'s SourceUSB is a software analyzer that records USB I/O requests and other events, including enumeration requests. You can view the

requests along with additional information about the system's host controllers, the devices on the host controllers' buses, and the drivers assigned to each host controller and device. Figure 17-4 shows logged requests and additional information about the request in the selected row.

The SourceUSB application can also display a tree of all of the system's host controllers and their attached devices and provide information about the drivers assigned to each host controller and device. As with a hardware analyzer, you have great flexibility in selecting what information you want to log and view.

Another software-only analyzer is the SnoopyPro project, free with source code from *www.sourceforge.net*.

Traffic Generators

Sometimes it's useful to be able to control bus traffic and signaling beyond what you can do from host software and device firmware. Some higher-end protocol analyzers can also function as traffic generators that emulate a host or device and give you precise control over the traffic that the emulated host or device places on the bus. In addition to generating valid traffic, a traffic generator can introduce errors such as bit-stuff and CRC errors. Two protocol analyzers with these abilities are Catalyst Enterprises, Inc.'s SBAE-30 Analyzer/Exerciser and LeCroy Corporation's CATC USBTracer/Trainer. Another option is RPM Systems' Root 2 USB Test Host, which emulates a USB host and enables you to specify traffic to generate on the bus, control the bus voltage, and measure bus current.

Testing

The USB-IF and Microsoft offer testing opportunities for developers of USB devices and host software. Passing the tests can earn a product the right to display the Certified USB logo and/or the Microsoft Windows logo. A logo can give users confidence that a device is thoroughly tested and reliable. A driver that passes Microsoft's tests can be digitally signed, which gives users confidence that the driver will work without problems.

Compliance Testing

One advantage USB has over other interfaces is that the developers of the specification didn't stop with the release of the specification. The USB-IF remains involved in helping developers design and test USB products. The USB-IF's Web site has many useful documents and tools.

The USB-IF has also developed a compliance program that specifies and sponsors tests for peripherals, hubs, host systems, On-The-Go devices, silicon building blocks, cable assemblies, and connectors. When a product passes the tests, The USB-IF deems it to have “reasonable measures of acceptability” and adds the product to its Integrators List of compliant devices. On receiving a signed license agreement and payment, the USB-IF authorizes the product to display the Certified USB logo. Even if you don't plan to submit your device to formal compliance testing, you can use the tests to verify your device's performance.

To pass compliance testing, a device must meet the requirements specified in the appropriate checklists and pass tests of the device's responses to standard control requests, the ability to operate under all host types and with other devices on the bus, and electrical performance. All of the tests except the high-speed electrical tests are described in the document *Universal Serial Bus Implementers Forum Full and Low Speed Electrical and Interoperability Compliance Test Procedure*. The specifications, procedures, and tools for high-speed electrical tests are in additional documents and files on the USB-IF's Web site.

You can submit a device for compliance testing at a compliance workshop sponsored by the USB-IF or at one of the independent labs that the USB-IF authorizes to perform the tests. To save time and expense, you should perform the tests as fully as possible on your own before submitting a product for compliance testing

Checklists

The compliance checklists contain a series of questions about a product's specifications and behavior. There are checklists for peripherals, hubs, hub and peripheral silicon, and host systems. The Peripheral checklist covers

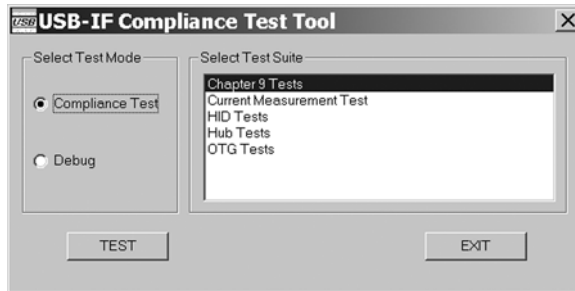


Figure 17-5: The USB Command Verifier utility includes several test suites.

mechanical design, device states and signals, operating voltages, and power consumption. You need to be able to answer yes to every question in the checklist. Accompanying each question is a reference to a section in the USB specification with more information.

Device Framework

The Device Framework tests verify that a device responds correctly to standard control requests. The USB Command Verifier (USBCV) software utility performs the tests. The document *USB Command Verifier Compliance Test Specification* describes the tests. The USBCV software and test-specification document are available from the USB-IF's Web site.

The USBCV software requires a PC with a USB 2.0 host controller. In addition, any low- or full-speed devices being tested must connect to the host via an external USB 2.0 hub. When you run USBCV, the software replaces the host-controller's driver with its own test-stack driver. On exiting USBCV, the software restores the original driver. The stack switching was tested using Microsoft's host-controller driver, and the USB-IF recommends running the software only on hosts that are using Microsoft's driver.

The software has several test suites: Chapter 9, Current Measurement, HID, Hub, and OTG (Figure 17-5).

In the Chapter 9 tests, the host issues the standard control requests defined in Chapter 9 of the USB specification and performs additional checks on the information returned by a device (Figure 17-6). For example, on retriev-

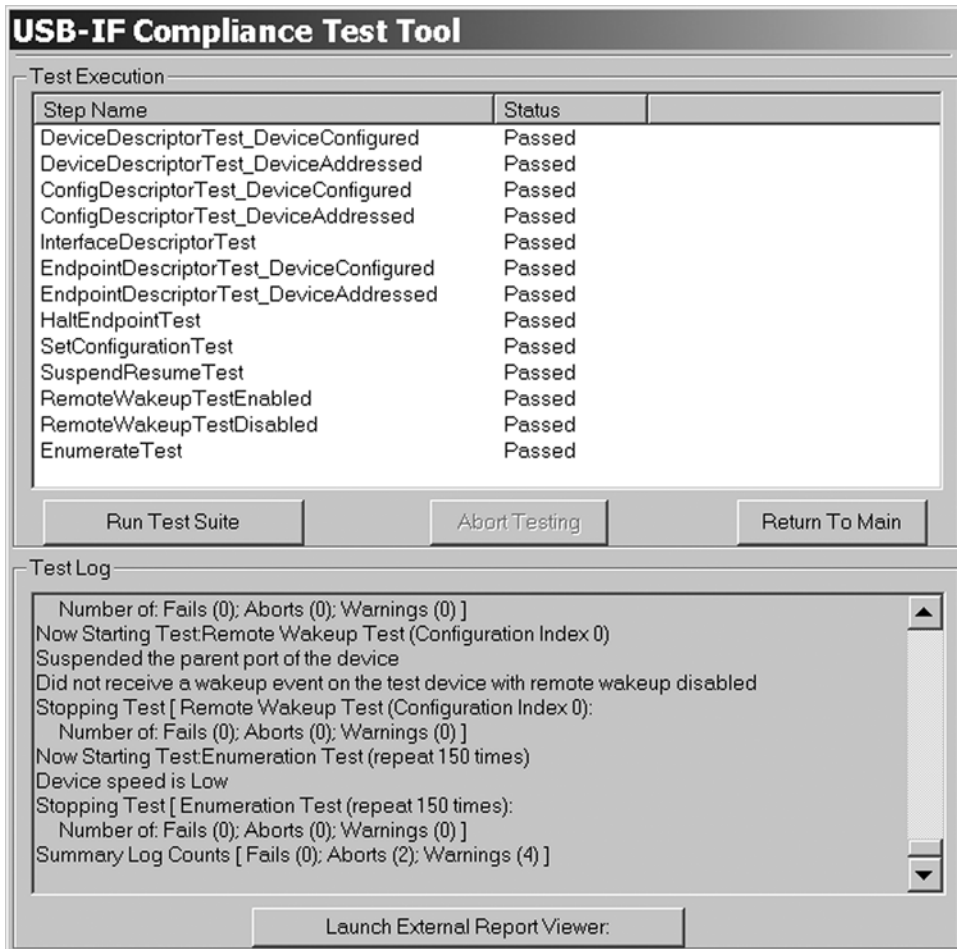


Figure 17-6: USBCV's Chapter 9 tests check the device's responses to the control requests defined in Chapter 9 of the USB specification.

ing a device descriptor, the software checks to see that the `bMaxPacketSize0` value is valid for the device's speed and that the `bDeviceClass` value is either a value for a standard class or `0FFh` (vendor-defined). The software requests the device descriptor when the device is in the default, address, and configured states, at both full and high speeds if the device supports both, and in every supported configuration.

The Chapter 9 tests also include these tests:

- Enumerate the device multiple times with different addresses.
- Verify that all bulk and interrupt endpoints can be halted with a Clear_Feature request.
- Ensure that the device returns STALL in response to receiving a request for an unsupported descriptor type.
- Ensure that the device returns STALL in response to receiving a Set_Feature request for an unsupported feature.
- Suspend and resume the device.
- If the device supports remote wakeup, suspend the device and request the user to perform an action to wake the device.

Every device must pass all of the Chapter 9 tests.

The Current Measurement test suite pauses with the device in the unconfigured and configured states to enable you to measure the bus current the device is drawing in each state. In the unconfigured state, the device should draw no more than 100 milliamperes. When configured, the device should draw no more than the amount specified in the bMaxPower field of the configuration descriptor for the currently active configuration.

Additional test suites provide tests for hubs, HID-class devices, and devices that return On-The-Go (OTG) descriptors.

The software has two modes. Compliance Test mode runs an entire test suite. Debug mode enables selecting and running a single test within a suite and offers more control, such as selecting a configuration to use when running a test.

Interoperability Tests

The interoperability tests emulate a user's experience by testing a product with different host controllers and with a variety of other USB devices in use. The device must be tested under both EHCI/UHCI and EHCI/OHCI hosts and under hubs that do and don't support high speed. To enable testing both implementations of the S3 Sleep state, the device must be tested both under a host that maintains VBUS on entering the S3 state and under a

host that removes VBUS on entering the S3 state. Devices are tested under all of these conditions:

- The bus is carrying control, bulk, interrupt, and isochronous transfers.
- There are five external hubs between the device and host.
- The device is 30 meters from the host (28 meters for low-speed devices).
- The bus is carrying full- and high-speed traffic.

For performing the tests, the test specification defines a Gold Tree that contains a variety of hubs and other devices on the bus with the device under test. As of revision 1.3 of the test specification, the Gold Tree contains these devices:

- Video camera: high speed, uses isochronous transfers, high power, bus powered.
- Mass storage device: high speed, uses bulk transfers, self powered.
- Flash media drive: high speed, uses bulk transfers, bus powered.
- Keyboard: low speed HID.
- Mouse: low speed HID.
- Seven hubs: five hubs that support all three bus speeds including one hub with multiple transaction translators; two hubs that support low and full speeds only.

The devices attach to the host in the configuration shown in Figure 17-7. The test specification names products that have been shown to have no interoperability problems of their own. You can use these or equivalent devices.

On attachment, the host must enumerate and install the driver for the device (with user assistance to identify the driver's location if appropriate). The device must operate properly while the other devices in the Gold Tree are also operating. In addition, the device must continue to operate properly after each of these actions:

- Detach the device and reattach it to the same port.
- Detach the device and attach it to a different port.

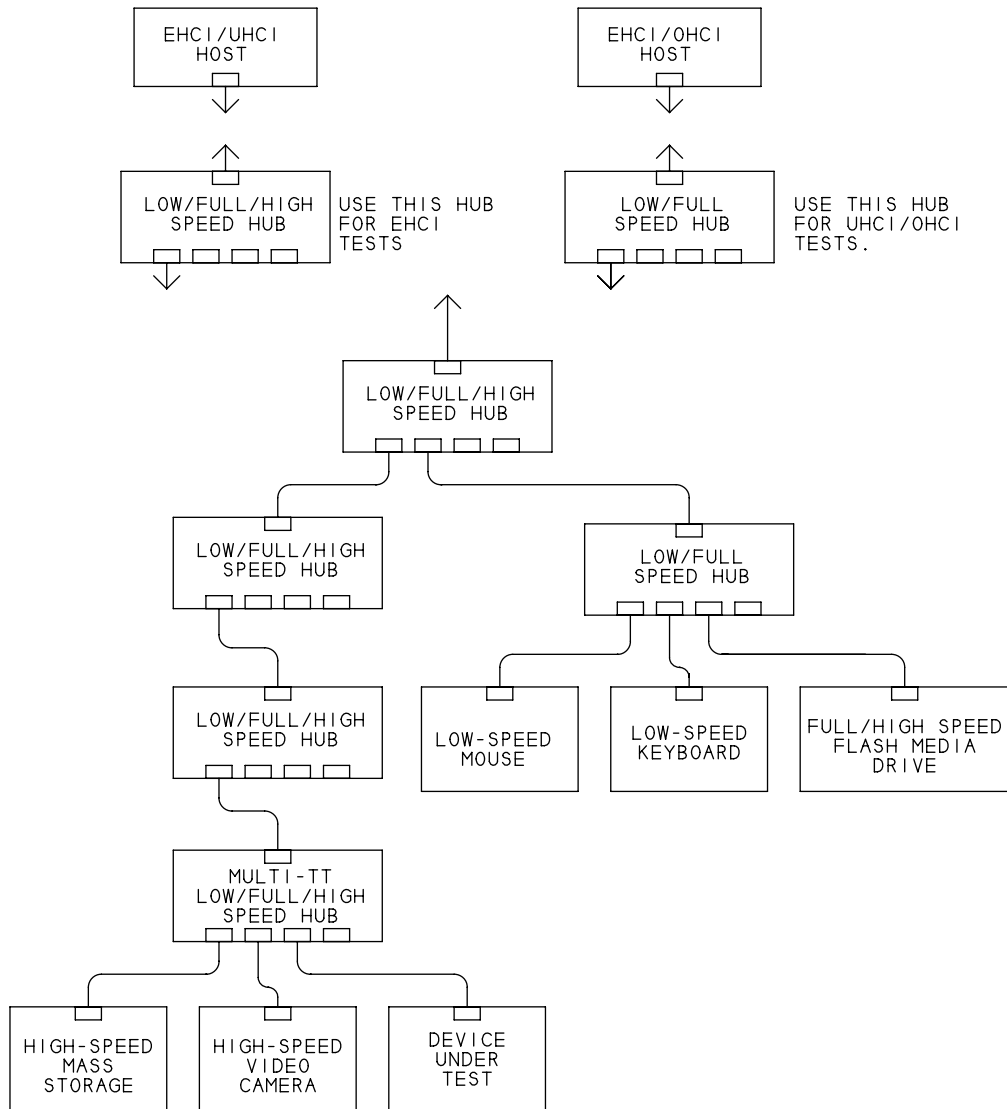


Figure 17-7: Compliance testing uses this Gold Tree configuration for testing how a device behaves in a system where other USB devices are in use.

- Do a warm boot. (Start > Shutdown > Restart.)
- Do a cold boot. (Start > Shutdown > Shutdown. Turn on the PC.)
- When the device is active, place the system in the S1 Sleep state and resume.
- When the device is idle, place the system in the S1 Sleep state and resume.
- When the device is active, place the system in the S3 Sleep state and resume.

A high-speed device must also be fully functional at full speed. The test specification has more details about the tests.

Waivers

A device can earn a USB Logo without passing every test. At its discretion, the USB-IF may grant a waiver of a requirement. For example, waivers have been granted for devices that should consume only 500 microamperes in the Suspend state but actually consume up to 2.5 milliamperes.

The Certified USB Logo

A device that passes compliance testing is eligible to display the official Certified USB logo. The logo indicates if a device supports high speed and/or USB On-The-Go (Figure 17-8). To use the logo, you must sign the USB-IF Trademark License Agreement. If you're not a member of the USB-IF, you also must pay a logo administration fee (\$1500 at this writing). The logo is different from the USB icon described in Chapter 19.

WHQL Testing

For devices and drivers that will be used on Windows PCs, Microsoft offers Windows Hardware Quality Labs (WHQL) testing. The tests identify devices and drivers that “meet a baseline definition of platform features and quality goals that ensure a good Windows experience for the end user.” When a device has passed WHQL tests, the device's packaging and marketing materials can display a *Designed for Microsoft Windows* logo. In



Figure 17-8: Devices that pass compliance testing can display one of the Certified USB logos. The logo indicates if the device supports high speed and/or USB On-The-Go.

Microsoft's online Windows Catalog of compatible devices, qualified devices show the logo in their listings. As Chapter 9 explained, a driver that passes WHQL testing has a digital signature that identifies the driver as a trusted driver.

The Windows Logo Program

To earn the Windows logo, a device must install and uninstall properly without interfering with other system components, and the device must interoperate well with other system components. Windows XP and Windows Server 2003 have different tests and logos. A device can qualify for multiple logos. Earning the Windows logo for a device requires performing the following steps:

- Pass the appropriate compatibility tests in the Windows Hardware Compatibility Test (HCT) kit provided by Microsoft.
- Use Microsoft's HCT Submission Review Utility to create a report that contains test logs of the compatibility tests.
- Submit the hardware, test logs, drivers (if any), user documentation, other configuration utilities or accessories as needed, and fee. The hardware is submitted to a Windows Quality Online Services test location. At this writing, the fee for most devices is \$250 per operating system.

Microsoft's Web site has the latest information and downloads relating to WHQL testing.

Digital Signatures

To earn the Windows logo, a device must use a digitally signed driver. The driver may be one of the drivers included with the operating system or the vendor may supply the driver. To obtain a digital signature, a driver must pass WHQL testing and the vendor must provide a VeriSign Digital Code Signing ID obtained from *www.verisign.com*. At this writing, a VeriSign ID costs \$400. Microsoft uses cryptographic technology to digitally sign the driver's catalog (.cat) file and returns the signed file to the vendor

The device's INF file references the catalog file. The signature enables Windows to detect if the driver has been modified since it passed WHQL testing. Each INF file has its own catalog file. A single INF file can support multiple devices. Any change in an INF file, including adding a new Product ID or device release number, requires obtaining a new digital signature.

For most USB devices, the INF file of a signed driver must include a device identification string that contains the device's VID and PID. An INF file that uses a compatible ID to identify the device only by class (and optional subclass and protocol) won't pass the WHQL tests, except for printers.

Your driver must be signed if you want it included in Microsoft's Windows Update. This feature of Windows makes it easy for users to update drivers installed on their systems. In some cases Windows Update can also find a driver for a newly installed device. A driver available via Windows Update

must meet additional requirements to ensure that Windows can easily identify, download, and install the driver.

Under Windows Server 2003 and later, some devices can use an alternate way to obtain a digital signature. If WHQL doesn't have a test program for the driver's setup class, a vendor can use Microsoft's code-signing tools to generate an Authenticode signature for the driver.

