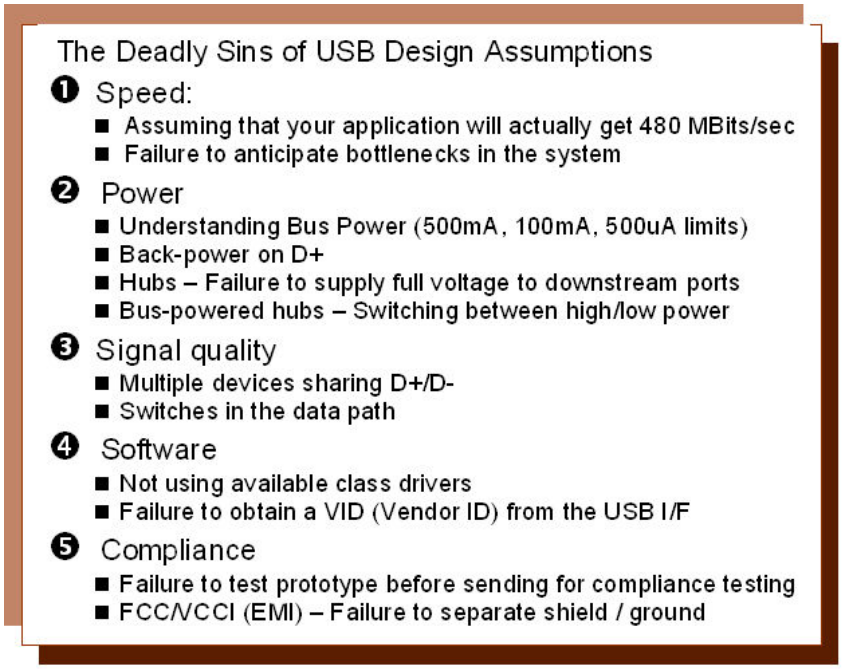
Common USB Development Mistakes – You Don't Have To Make Them All Yourself!

By (Steve Kolokowsky, Systems Products Engineer, Cypress Semiconductor Corp. and Trevor Davis)

Executive Summary

11 years have passed since USB was first introduced. Believe it or not, people are finding new and innovative ways to use this protocol every single day. They are also making the same mistakes every single day on thousands of designs worldwide! When will learn from one another? How about right now? Interestingly, USB has many similarities to the protocols that developers are already familiar with. And yet, it is far enough away from the familiar territory of PS/2 and RS-232 that engineers make the same mistakes over and over again. Additionally, with tough compliance standards in place, engineers make mistakes that could cost them USB compliance. These mistakes fall into five main categories: Speed, power, signal quality, software and compliance.

Figure 1. The Most Common "Deadly Sins" For USB Design



Speed

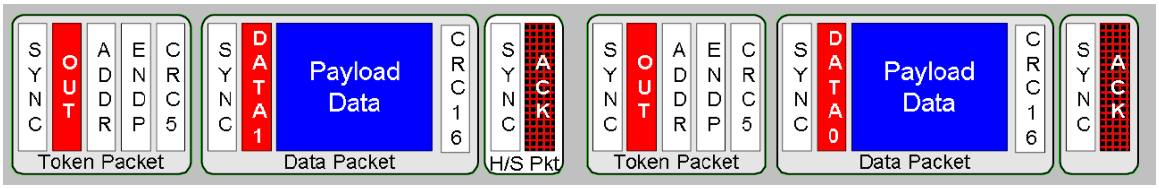
Without question, one of the most misunderstood USB issues between consumers and developers, and one of the most common questions on the floors of retail stores all over the world, is what speed a given USB device is. There are three current data transfer speeds for wired USB. USB Low Speed transfers data at 1.5 Mbits/sec, Full Speed USB transfers data at 12 Mbit/sec, and High Speed USB transfers data at 480 Mbit/sec. Don't get confused though: **USB 2.0 does not equal High Speed USB**. High Speed USB was first introduced in the USB Specification version 2.0 release that also addressed both Low Speed and Full Speed data transfers.

As with any electronic system, designers want the best performance possible. Unfortunately, with USB, many designers begin their design believing they are going to get the full 1.5, 12, or 480 Mbit/sec performance from their system – this is a bad assumption. There are several reasons why your device will never be able to use all of this bandwidth. First of all, the USB bus is shared among several users. Even if you are plugged into different ports on the motherboard, you are probably sharing

the same host controller as all of the other devices on the bus, so your device is sharing the USB bus bandwidth with all of the other devices.

Second, USB is a packetized protocol where longer blocks of data are divided into 512-byte packets. Each packet contains a header identifying the packet contents, and a CRC at the end of the packet for data integrity. Each packet also requires an ACK from the other side of the link. Start of Frame (SOF) packets are sent every 125 uSec (microframe) to maintain timing on the bus. The net effect of this is that the theoretical maximum bandwidth of USB is 13 bulk packets per microframe, or 53,248,000 bytes/second. Even this limit is not achievable with current host controllers, which can receive 10 bulk packets/microframe or send 8 bulk packets/microframe.

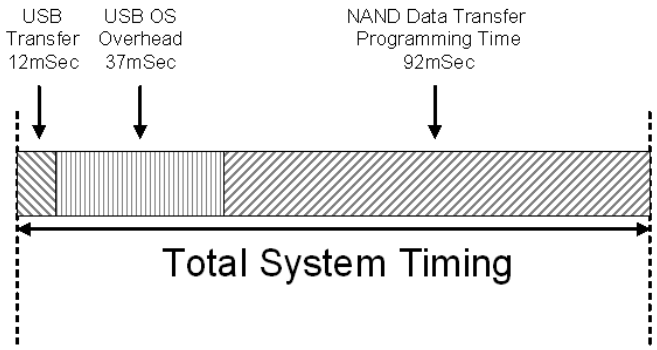
Figure 2. USB Bulk Transfer Frame



Failure to Anticipate Bottlenecks in the System

More than one high-speed system has fallen prey to this issue. Cypress Semiconductor recently analyzed a system that was writing data to a NAND flash. This system was sending data over USB, storing the data into a buffer and then writing the data to the flash. Every packet had three components of time: 1) the time to perform the USB transfer, 2) the primary Operating System overhead timing, and 3) the programming time of the NAND Flash firmware. A real-time analysis of the performance revealed the timing below for a 128Kbyte block:

Figure 3. System Bottleneck Example



Until this time breakdown was completed, the engineers were spending all of their time trying to reduce the USB transfer time by speeding up the waveforms used by the USB interface chip. Once they realized, however, that performance was dominated by their NAND Flash programming firmware, they were able to greatly improve performance by reducing their NAND overhead. In most systems, high-speed USB will not be the bottleneck. As a result, designers must look closely at their entire system to ensure they have the bandwidth headroom to reach the system speeds they desire.

Power

USB Bus Power

According to the USB specification, USB devices can either be "Bus-powered", powered through the USB cable, or "Self-powered", powered by a battery or plugged into the wall. One of the best ideas in USB was allowing bus-powered devices – no need for a power plug! However, using USB bus-power means that you have to live within the 500uA, 100mA and 500mA limits imposed by USB. Unfortunately, many designers do not closely monitor these limits and they create designs that do not properly comply with Bus Powered compliance rules.

- 500uA – When the host is supplying power, but there is no activity on USB, your device must be in USB suspend. In this state, you can only draw 500uA from VBUS. This state exists to minimize current draw when a PC is in suspend mode.
- 100mA – USB has high-power (500mA) and low-power (100mA) ports. Low-power ports are generally found on bus-powered hubs, which take in 500mA and distribute 100mA to each of their downstream ports. When your device is first plugged in, it doesn't know what kind of port it is plugged into, so it is limited to 100mA until it receives a SET_CONFIGURATION message from the host. This means that your device must be functional enough to enumerate on USB at a very low power setting until it receives the SET_CONFIGURATION message that allows it to switch to a high power setting. This was very difficult to do in high-speed USB until the Cypress Semiconductor's FX2LP chip was introduced in 2004.
- 500mA – This is the absolute maximum power allowed under the USB spec.

Real world design testing should be conducted by system engineers to ensure they are meeting all the different power levels necessary for Bus Powered operation – or else you'll be shipping an expensive wall plug with that new USB system of yours.

Back-power

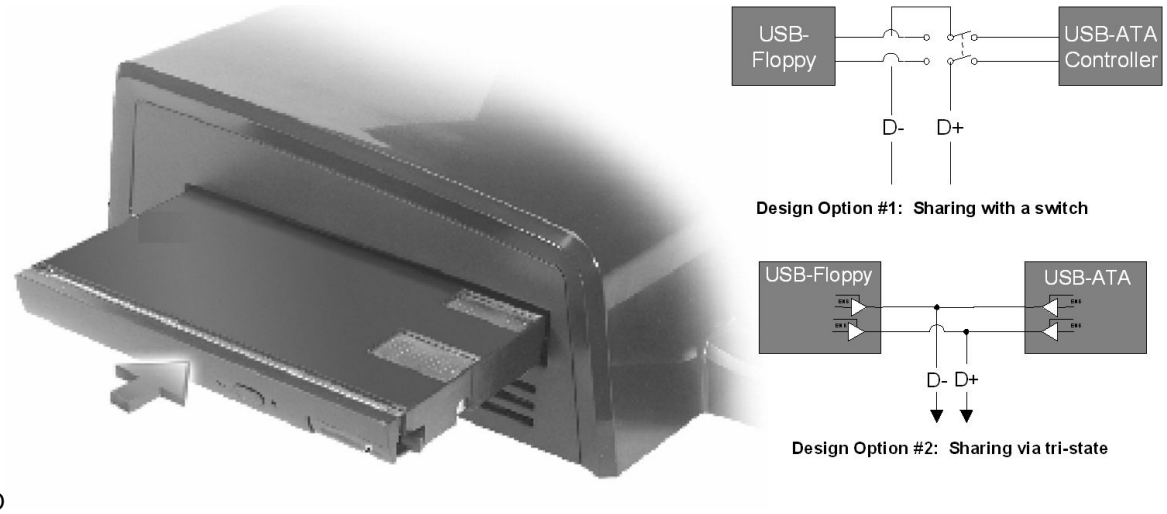
"Self-powered" USB devices can also have their own power problems. Since self-powered devices have their own independent power supplies, they can be ON while the host is turned OFF. This causes a potential problem where the small pull-up voltage applied to D+ to enable USB device detection slowly charges up the entire host system and interferes with startup. Self-powered USB devices (including battery-powered devices) must either drive this pull-up directly from VBUS or turn it off via software control using a VBUS sensor.

Signal Quality

Sharing D+/D-

In an effort to save time, effort, and money, some products attempt to share the USB signal lines between multiple devices. For example, a USB-based docking station may want to allow a floppy drive or a DVD player to be inserted in a storage slot. Sharing the USB lines between the two devices saves cost by reducing the number of required hub ports. However, it is difficult to implement either of these approaches without fully understanding the characteristics of all of the devices in the system. In the tristate arrangement (Figure 4 option #2), the other device on the bus will add capacitance to the USB lines. Also, the trace to this other device will cause reflections that may interfere with high-speed USB operation. In the switch arrangement (Figure 4 option #1), the switch will add both capacitance and resistance to the USB lines, slowing the rise/fall times of the USB lines and closing the USB eye.

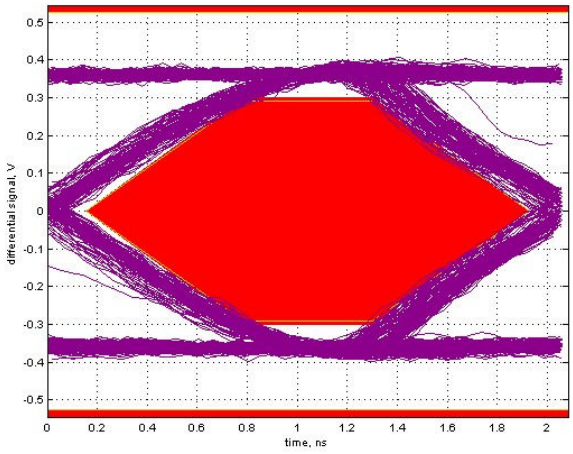
Figure 4: Docking Station D+/D- Sharing Options



O

Figure 5 shows a typical USB device transmit eye after it goes through a switch with a 10pF / 10 Ohm load; there should be no interference with the red area within the signal eye. The key to successful USB signal sharing is to keep this load low and use a chip with fast edge rates.

Figure 5. Impeding Signals with Extra Interference Because of Shared Signal Lines



Software

Not Using Available Class Drivers

Class drivers are an important part of the USB ecosystem. These drivers are provided by the major operating systems so that device driver development is not needed. USB classes are defined by the Device Working Groups, volunteers who meet under the auspices of the USB to create a standard language for talking to devices. The USB device classes that exist today include HID – Human Interface Device (mice, keyboards and other controls), Mass Storage (disk drives), Communication (modems, network adapters), Audio, Video, and Still Image – Photos and Scanners.

If your device exactly fits into the existing class structure, the decision is simple. Just go to www.usb.org, download the class definition and implement it. However, if your device doesn't exactly fit into the existing classes, they can still be useful. For

example, Microsoft used the Still Image Class to implement the new MTP (Media Transfer Protocol) class. The HID class doesn't have to connect to something that interacts with the user; it can be used to interface thermometers, pressure sensors, and pump controls to your program without creating a driver.

For some reason, many companies attempt to take on custom driver development or to hire outside driver design houses to perform expensive driver development when implementation of a Class driver will do. Class drivers eliminate design risk, cost and schedule issues, and debugging and complexity issues.

Failure to Obtain a VID from the USB I/F

Every USB device contains a unique identifier so that the operating system can find the correct driver for the device. The first part of the identifier is a 16-bit value assigned by the USB Implementor's Forum (www.usb.org) (VID). The second part of the identifier is a 16 bit value assigned by the vendor (the company that created the product), called the PID.

USB Vendor ID – Assigned by USB Implementers Forum (USB I/F)	Product ID – Assigned by Vendor
16 bits	16 bits

Obtaining the VID and then creating the PID is a simple process if the company and designer plans ahead and is in proper communication with the USB Implementers Forum. Unfortunately, every year thousands of devices are delayed as a result of not having proper identifiers assigned as firmware or software work is being done and the VID and PID numbers are needed for identification.

Compliance

Failure to Test Prototype Before Sending for Compliance Testing

To legally use the various USB logos, products must pass USB compliance testing. The USB IF performs compliance testing to insure that all end customers have a good experience with USB. This is important because all USB vendors are relying on each other to generate goodwill with the public. If a customer has a bad experience with one USB device, that customer will be much less willing to invest his time and effort in another one.

Figure 6: USB Certified Logos

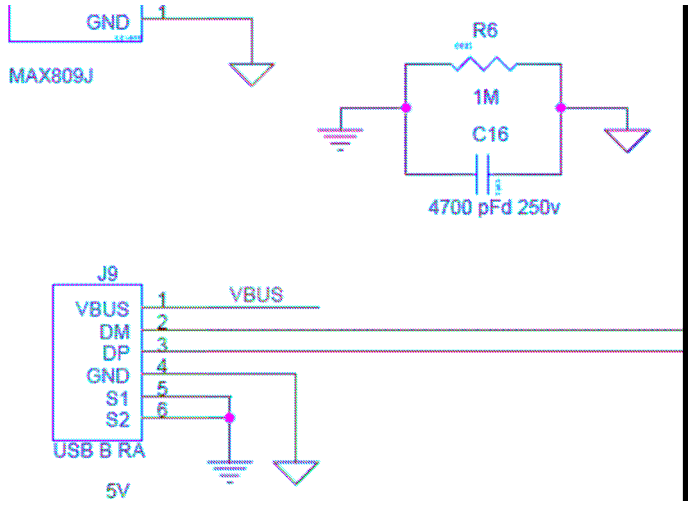
Some of the tests done during compliance testing require expensive, high-end testing equipment that may be difficult for developers to afford. However, many devices fail compliance testing due to simple items that any developer can check. Before heading out to a plugfest or sending your device for compliance testing, you should at least run these tests:

- Chapter 9 test – USB Command Verifier Tool. This program verifies that your device can handle the most important SETUP commands that may be sent from a host (check http://www.usb.org/developers/tools/ for tools)
- Power tests – Suspend current, inrush current, unconfigured current

EMI

Good EMI design techniques can fill an entire article on their own. The most common and simple-to-fix EMI error is mistakenly tying the shield in the USB cable directly to the ground plane of your system. This allows any noise injected into the ground plane to escape any shielding around your device. In addition, this noise is now being broadcast on a 2-meter long antenna!

Figure 7: Schematic of Cypress FX2LP development board with separate shield ground



Conclusion

So now you know the problem areas. Hundreds of designers every year lose more and more hair trying to solve issues resulting from the problems above. Save yourself the hair-loss and the agony of rework and schedule delays – learn from the mistakes of others. But most of all, don't stop being creative because the world will continue to demand more innovative, sophisticated, and enjoyable USB based devices. Happy designing!

References

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com