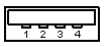
Connectors

All devices have an upstream connection to the host and all hosts have a downstream connection to the device. Upstream and downstream connectors are not mechanically interchangeable, thus eliminating illegal loopback connections at hubs such as a downstream port connected to a downstream port. There are commonly two types of connectors, called type A and type B which are shown below.



Type A USB Connector

Type B USB Connector

Type A plugs always face upstream. Type A sockets will typically find themselves on hosts and hubs. For example type A sockets are common on computer main boards and hubs. Type B plugs are always connected downstream and consequently type B sockets are found on devices.

It is interesting to find type A to type A cables wired straight through and an array of USB gender changers in some computer stores. This is in contradiction of the USB specification. The only type A plug to type A plug devices are bridges which are used to connect two computers together. Other prohibited cables are USB extensions which has a plug on one end (either type A or type B) and a socket on the other. These cables violate the cable length requirements of USB.

USB 2.0 included errata which introduces mini-usb B connectors. The details on these connectors can be found in Mini-B Connector Engineering Change Notice The reasoning behind the mini connectors came from the range of miniature electronic devices such as mobile phones and organisers. The current type B connector is too large to be easily integrated into these devices.

Just recently released has been the On-The-Go specification which adds peer-to-peer functionality to USB. This introduces USB hosts into mobile phone and electronic organisers, and thus has included a specification for mini-A plugs, mini-A receptacles, and mini-AB receptacles. I guess we should be inundated with mini USB cables soon and a range of mini to standard converter cables.

Pin Number	Cable Colour	Function
1	Red	V_{BUS} (5 volts)
2	White	D-
3	Green	D+
4	Black	Ground

Standard internal wire colours are used in USB cables, making it easier to identify wires from manufacturer to manufacturer. The standard specifies various electrical parameters for the cables. It is interesting to read the detail the original USB 1.0 spec included. You would understand it specifying electrical attributes, but paragraph 6.3.1.2 suggested the recommended colour for overmolds on USB cables should be frost white - how boring! USB 1.1 and USB 2.0 was relaxed to recommend Black, Grey or Natural.

PCB designers will want to reference chapter 6 for standard foot prints and pinouts.

Electrical

Unless you are designing the silicon for a USB device/transceiver or USB host/hub, there is not all that much you need to know about the electrical specifications in chapter 7. We briefly address the essential points here.

As we have discussed, USB uses a differential transmission pair for data. This is encoded using NRZI and is bit stuffed to ensure adequate transitions in the data stream. On low and full speed devices, a differential ‘1’ is transmitted by pulling D+ over 2.8V with a 15K ohm resistor pulled to ground and D- under 0.3V with a 1.5K ohm resistor pulled to 3.6V. A differential ‘0’ on the other hand is a D- greater than 2.8V and a D+ less than 0.3V with the same appropriate pull down/up resistors.

The receiver defines a differential ‘1’ as D+ 200mV greater than D- and a differential ‘0’ as D+ 200mV less than D-. The polarity of the signal is inverted depending on the speed of the bus. Therefore the terms ‘J’ and ‘K’ states are used in signifying the logic levels. In low speed a ‘J’ state is a differential 0. In high speed a ‘J’ state is a differential 1.

USB transceivers will have both differential and single ended outputs. Certain bus states are indicated by single ended signals on D+, D- or both. For example a single ended zero or SE0 can be used to signify a device reset if held for more than 10mS. A SE0 is generated by holding both D- and D+ low (< 0.3V). Single ended and differential outputs are important to note if you are using a transceiver and FPGA as your USB device. You cannot get away with sampling just the differential output.

The low speed/full speed bus has a characteristic impedance of 90 ohms +/- 15%. It is therefore important to observe the datasheet when selecting impedance matching series resistors for D+ and D-. Any good datasheet should specify these values and tolerances.

High Speed (480Mbits/s) mode uses a 17.78mA constant current for signalling to reduce noise.

Speed Identification

A USB device must indicate its speed by pulling either the D+ or D- line high to 3.3 volts. A full speed device, pictured below will use a pull up resistor attached to D+ to specify itself as a full speed device. These pull up resistors at the device end will also be used by the host or hub to detect the presence of a device connected to its port. Without a pull up resistor, USB assumes there is nothing connected to the bus. Some devices have this resistor built into its silicon, which can be turned on and off under firmware control, others require an external resistor.

For example Philips Semiconductor has a SoftConnect^{TM} technology. When first connected to the bus, this allows the microcontroller to initialise the USB function device before it enables the pull up speed identification resistor, indicating a device is attached to the bus. If the pull up resistor was connected to V_{bus}, then this would indicate a device has been connected to the bus as soon as the plug is inserted. The host may then attempt to reset the device and ask for a descriptor when the microprocessor hasn’t even started to initialise the usb function device.

Other vendors such as Cypress Semiconductor also use a programmable resistor for Re-Numeration^{TM} purposes in their EzUSB devices where the one device can be enumerated for one function such as In field programming then be disconnected from the bus under firmware control, and enumerate as another different device, all without the user lifting an eyelid. Many of the EzUSB devices do not have any Flash or OTP ROM to store code. They are bootstraped at connection.

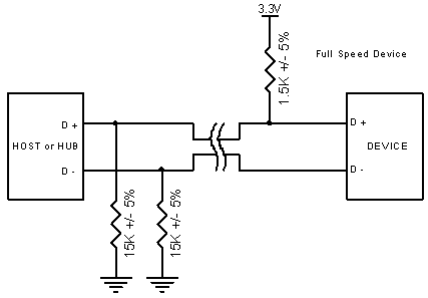


Figure 2 : Full Speed Device with pull up resistor connected to D+

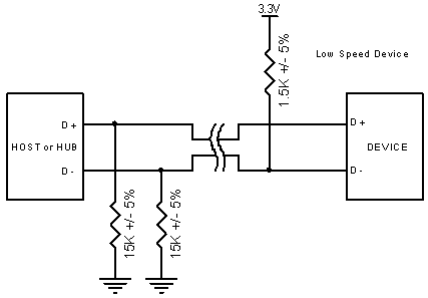


Figure 3 : Low Speed Device with pull up resistor connected to D-

You will notice we have not included speed identification for High Speed mode. High speed devices will start by connecting as a full speed device (1.5k to 3.3V). Once it has been attached, it will do a high speed chirp during reset and establish a high speed connection if the hub supports it. If the device operates in high speed mode, then the pull up resistor is removed to balance the line.

A USB 2.0 compliant device is not required to support high-speed mode. This allows cheaper devices to be produced if the speed isn't critical. This is also the case for a low speed USB 1.1 devices which is not required to support full speed.

However a high speed device must not support low speed mode. It should only support full speed mode needed to connect first, then high speed mode if successfully negotiated later. A USB 2.0 compliant downstream facing device (Hub or Host) must support all three modes, high speed, full speed and low speed.

Power (V_{BUS})

One of the benefits of USB is bus-powered devices - devices which obtain its power from the bus and requires no external plug packs or additional cables. However many leap at this option without first considering all the necessary criteria.

A USB device specifies its power consumption expressed in 2mA units in the configuration descriptor which we will examine in detail later. A device cannot increase its power consumption, greater than what it specifies during enumeration, even if it looses external power. There are three classes of USB functions,

- Low-power bus powered functions
- High-power bus powered functions
- Self-powered functions

Low power bus powered functions draw all its power from the V_{BUS} and cannot draw any more than one unit load. The USB specification defines a unit load as 100mA. Low power bus powered functions must also be designed to work down to a V_{BUS} voltage of 4.40V and up to a maximum voltage of 5.25V measured at the upsteam plug of the device. For many 3.3V devices, LDO regulators are mandatory.

High power bus powered functions will draw all its power from the bus and cannot draw more than one unit load until it has been configured, after which it can then drain 5 unit loads (500mA Max) provided it asked for this in its descriptor. High power bus functions must be able to be detected and enumerated at a minimum 4.40V. When operating at a full unit load, a minimum V_{BUS} of 4.75 V is specified with a maximum of 5.25V. Once again, these measurements are taken at the upstream plug.

Self power functions may draw up to 1 unit load from the bus and derive the rest of it's power from an external source. Should this external source fail, it must have provisions in place to draw no more than 1 unit load from the bus. Self powered functions are easier to design to specification as there is not so much of an issue with power consumption. The 1 unit bus powered load allows the detection and enumeration of devices without mains/secondary power applied.

No USB device, whether bus powered or self powered can drive the V_{BUS} on its upstream facing port. If V_{BUS} is lost, the device has a lengthy 10 seconds to remove power from the D+/D- pull-up resistors used for speed identification.

Other V_{BUS} considerations are the Inrush current which must be limited. This is outlined in the USB specification paragraph 7.2.4.1 and is commonly overlooked. Inrush current is contributed to the amount of capacitance on your device between V_{BUS} and ground. The spec therefore specifies that the maximum decoupling capacitance you can have on your device is 10uF. When you disconnect the device after current is flowing through the inductive USB cable, a large flyback voltage can occur on the open end of the cable. To prevent this, a 1uF minimum V_{BUS} decoupling capacitance is specified.

For the typical bus powered device, it can not drain any more than 500mA which is not unreasonable. So what is the complication you ask? Perhaps Suspend Mode?

Suspend Current

Suspend mode is mandatory on all devices. During suspend, additional constrains come into force. The maximum suspend current is proportional to the unit load. For a 1 unit load device (default) the maximum suspend current is 500uA. This includes current from the pull up resistors on the bus. At the hub, both D- and D+ have pull down resistors of 15K ohms. For the purposes of power consumption, the pull down resistor at the device is in series with the 1.5K ohms pull up, making a total load of 16.5K ohms on a V_{TERM} of typically 3.3v. Therefore this resistor sinks 200uA before we even start.

Another consideration for many devices is the 3.3V regulator. Many of the USB devices run on 3.3V. The PDIUSBD11 is one such example. Linear regulators are typically quite inefficient with average quiescent currents in the order of 600uA, therefore more efficient and thus expensive regulators are called for. In the majority of cases, you must also slow down or stop clocks on microcontrollers to fall within the 500uA limit.

Many developers ask in the USB Implementor's Forum, what are the complications of exceeding this limit? It is understood, that most hosts and hubs don't have the ability to detect such an overload of this magnitude and thus if you drain maybe 5mA or even 10mA you should still be fine, bearing in mind that at the end of the day, your device violates the USB specification. However in normal operation, if you try to exceed the 100mA or your designated permissible load, then expect the hub or host to detect this and disconnect your device, in the interest of the integrity of the bus.

Of course these design issues can be avoided if you choose to design a self powered device. Suspend currents may not be a great concern for desktop computers but with the introduction of the On-The-Go Specification we will start seeing USB hosts built into mobile phones and mobile organisers. The power consumption pulled from these devices will adversely effect the operating life of the battery.

Entering Suspend Mode

A USB device will enter suspend when there is no activity on the bus for greater than 3.0ms. It then has a further 7ms to shutdown the device and draw no more than the designated suspend current and thus must be only drawing the rated suspend current from the bus 10mS after bus activity stopped. In order to maintain connected to a suspended hub or host, the device must still provide power to its pull up speed selection resistors during suspend.

USB has a start of frame packet or keep alive sent periodically on the bus. This prevents an idle bus from entering suspend mode in the absence of data.

- A high speed bus will have micro-frames sent every 125.0 µs ±62.5 ns.

- A full speed bus will have a frame sent down each 1.000 ms ±500 ns.
- A low speed bus will have a keep alive which is a EOP (End of Packet) every 1ms only in the absence of any low speed data.

The term "Global Suspend" is used when the entire USB bus enters suspend mode collectively. However selected devices can be suspended by sending a command to the hub that the device is connected too. This is referred to as a "Selective Suspend."

The device will resume operation when it receives any non idle signalling. If a device has remote wakeup enabled then it may signal to the host to resume from suspend.

Data Signalling Rate

Another area which is often overlooked is the tolerance of the USB clocks. This is specified in the USB specification, section 7.1.11.

- High speed data is clocked at 480.00Mb/s with a data signalling tolerance of ± 500ppm.
- Full speed data is clocked at 12.000Mb/s with a data signalling tolerance of ±0.25% or 2,500ppm.
- Low speed data is clocked at 1.50Mb/s with a data signalling tolerance of ±1.5% or 15,000ppm.

This allows resonators to be used for low cost low speed devices, but rules them out for full or high speed devices.

 Chapter 1 : Introduction

Chapter 3 : USB Protocols