USB interface

Introduction:

Universal Serial Bus (USB) is a set of interface specifications for high speed wired communication between electronics systems peripherals and devices with or without PC/computer. The USB was originally developed in 1995 by many of the industry leading companies like Intel, Compaq, Microsoft, Digital, IBM, and Northern Telecom.

The major goal of USB was to define an external expansion bus to add peripherals to a PC in easy and simple manner. The new external expansion architecture, highlights,

1. PC host controller hardware and software
2. Robust connectors and cable assemblies
3. Peripheral friendly master-slave protocols
4. Expandable through multi-port hubs.

USB offers users simple connectivity. It eliminates the mix of different connectors for different devices like printers, keyboards, mice, and other peripherals. That means USB-bus allows many peripherals to be connected using a single standardized interface socket. Another main advantage is that, in USB environment, DIP-switches are not necessary for setting peripheral addresses and IRQs. It supports all kinds of data, from slow mouse inputs to digitized audio and compressed video.

USB also allows hot swapping. The "hot-swapping" means that the devices can be plugged and unplugged without rebooting the computer or turning off the device. That means, when plugged in, everything configures automatically. So the user needs not worry about terminations, terms such as IRQs and port addresses, or rebooting the computer. Once the user is finished, they can simply unplug the cable out, the host will detect its absence and automatically unload the driver. This makes the USB a plug-and-play interface between a computer and add-on devices.

The loading of the appropriate driver is done using a PID/VID (Product ID/Vendor ID) combination. The VID is supplied by the USB Implementer's forum

Fig 1: The USB "trident" logo

The USB has already replaced the RS232 and other old parallel communications in many applications. USB is now the most used interface to connect devices like mouse, keyboards, PDAs, game-pads and joysticks, scanners, digital cameras, printers, personal media players, and flash drives to personal computers. Generally speaking, USB is the most successful interconnect in the history of personal computing and has migrated into consumer electronics and mobile products.

USB sends data in serial mode i.e. the parallel data is serialized before sends and de-serialized after receiving.

The benefits of USB are low cost, expandability, auto-configuration, hot-plugging and outstanding performance. It also provides power to the bus, enabling many peripherals to operate without the added need for an AC power adapter.

Various versions USB:

As USB technology advanced the new version of USB are unveiled with time. Let us now try to understand more about the different versions of the USB.

USB1.0: Version 0.7 of the USB interface definition was released in November 1994. But USB 1.0 is the original release of USB having the capability of transferring 12 Mbps, supporting up to 127 devices. And as we know it was a combined effort of some large players on the market to define a new general device interface for computers. This USB 1.0 specification model was introduced in January1996. The data transfer rate of this version can accommodate a wide range of devices, including MPEG video devices, data gloves, and digitizers. This version of USB is known as full-speed USB.

Since October-1996, the Windows operating systems have been equipped with USB drivers or special software designed to work with specific I/O device types. USB got integrated into Windows 98 and later versions. Today, most new computers and peripheral devices are equipped with USB.

USB1.1: USB 1.1 came out in September 1998 to help rectify the adoption problems that occurred with earlier versions, mostly those relating to hubs.

USB 1.1 is also known as full-speed USB. This version is similar to the original release of USB; however, there are minor modifications for the hardware and the specifications. USB version 1.1 supported two speeds, a full speed mode of 12Mbits/s and a low speed mode of 1.5Mbits/s. The 1.5Mbits/s mode is slower and less susceptible to EMI, thus reducing the cost of ferrite beads and quality components.

USB2.0: Hewlett-Packard, Intel, LSI Corporation, Microsoft, NEC, and Philips jointly led the initiative to develop a higher data transfer rate than the 1.1 specifications. The USB 2.0 specification was released in April 2000 and was standardized at the end of 2001. This standardization of the new device-specification made backward compatibility possible, meaning it is also capable of supporting USB 1.0 and 1.1 devices and cables.

Supporting three speed modes (1.5, 12 and 480 megabits per second), USB 2.0 supports low-bandwidth devices such as keyboards and mice, as well as high-bandwidth ones like high-resolution Web-cams, scanners, printers and high-capacity storage systems.

USB 2.0, also known as hi-speed USB. This hi-speed USB is capable of supporting a transfer rate of up to 480 Mbps, compared to 12 Mbps of USB 1.1. That's about 40 times as fast! Wow!

USB3.0: USB 3.0 is the latest version of USB release. It is also called as Super-Speed USB having a data transfer rate of 4.8 Gbit/s (600 MB/s). That means it can deliver over 10x the speed of today's Hi-Speed USB connections.

The USB 3.0 specification was released by Intel and its partners in August 2008. Products using the 3.0 specifications are likely to arrive in 2009 or 2010. The technology targets fast PC sync-and-go transfer of applications, to meet the demands of Consumer Electronics and mobile segments focused on high-density digital content and media.

USB 3.0 is also a backward-compatible standard with the same plug and play and other capabilities of previous USB technologies. The technology draws from the same architecture of wired USB. In addition, the USB 3.0 specification will be optimized for low power and improved protocol efficiency.

USB system overview:

The USB system is made up of a host, multiple numbers of USB ports, and multiple peripheral devices connected in a tiered-star topology. To expand the number of USB ports, the USB hubs can be included in the tiers, allowing branching into a tree structure with up to five tier levels.

The tiered star topology has some benefits. Firstly power to each device can be monitored and even switched off if an overcurrent condition occurs without disrupting other USB devices. Both high, full and low speed devices can be supported, with the hub filtering out high speed and full speed transactions so lower speed devices do not receive them.

The USB is actually an addressable bus system, with a seven-bit address code. So it can support up to 127 different devices or nodes at once (the "all zeroes" code is not a valid address). However it can have only one host: the PC itself. So a PC with its peripherals connected via the USB forms a star local area network (LAN).

On the other hand any device connected to the USB can have a number of other nodes connected to it in daisy-chain fashion, so it can also form the hub for a mini-star sub-network. Similarly it is possible to have a device, which purely functions as a hub for other node devices, with no separate function of its own. This expansion via hubs is possible because the USB supports a tiered star topology. Each USB hub acts as a kind of traffic cop. for its part of the network, routing data from the host to its correct address and preventing bus contention clashes between devices trying to send data at the same time.

On a USB hub device, the single port used to connect to the host PC either directly or via another hub is known as the upstream port, while the ports used for connecting other devices to the USB are known as the downstream ports. USB hubs work transparently as far as the host PC and its operating system are concerned. Most hubs provide either four or seven downstream ports or less if they already include a USB device of their own.

The host is the USB system's master, and as such, controls and schedules all communications activities. Peripherals, the devices controlled by USB, are slaves responding to commands from the host. USB devices are linked in series through hubs. There always exists one hub known as the root hub, which is built in to the host controller.

A physical USB device may consist of several logical sub-devices that are referred to as device functions. A single device may provide several functions, for example, a web-cam (video device function) with a built-in microphone (audio device function). In short, the USB specification recognizes two kinds of peripherals: stand-alone (single function units, like a mouse) or compound devices like video camera with separate audio processor.
The logical channel connection host to peripheral-end is called pipes in USB. A USB device can have 16 pipes coming into the host controller and 16 going out of the controller.

The pipes are unidirectional. Each interface is associated with single device function and is formed by grouping endpoints.

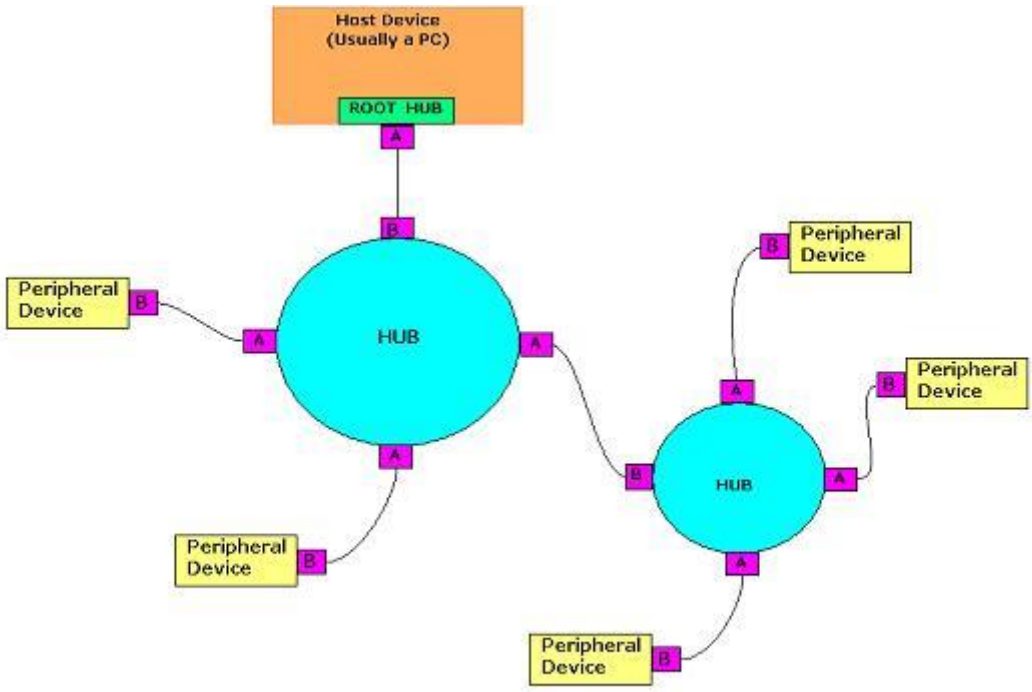


FIGURE Fig2: The USB "tiered star" topology

The hubs are bridges. They expand the logical and physical fan-out of the network. A hub has a single upstream connection (that going to the root hub, or the next hub closer to the root), and one to many downstream connections.

Hubs themselves are considered as USB devices, and may incorporate some amount of intelligence. We know that in USB users may connect and remove peripherals without powering the entire system down. Hubs detect these topology changes. They also source power to the USB network. The power can come from the hub itself (if it has a built-in power supply), or can be passed through from an upstream hub.

USB connectors & the power supply:
Connecting a USB device to a computer is very simple -- you find the USB connector on the back of your machine and plug the USB connector into it. If it is a new device, the operating system auto-detects it and asks for the driver disk. If the device has already been installed, the computer activates it and starts talking to it.

The USB standard specifies two kinds of cables and connectors. The USB cable will usually have an "A" connector on one end and a "B" on the other. That means the USB devices will have an "A" connection on it. If not, then the device has a socket on it that accepts a USB "B" connector.

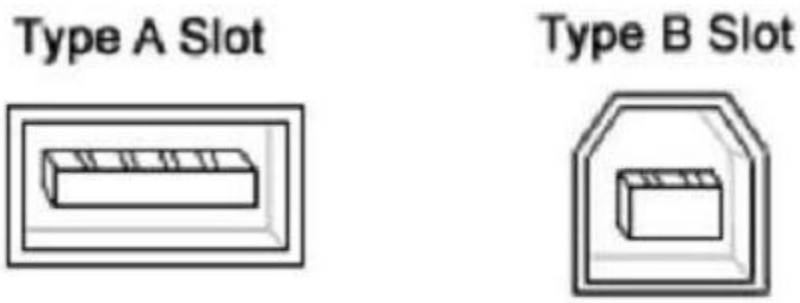


Fig 3: USB Type A & B Connectors

The USB standard uses "A" and "B" connectors mainly to avoid confusion:

1. "A" connectors head "upstream" toward the computer.
2. "B" connectors head "downstream" and connect to individual devices.

By using different connectors on the upstream and downstream end, it is impossible to install a cable incorrectly, because the two types are physically different.

Individual USB cables can run as long as 5 meters for 12Mbps connections and 3m for 1.5Mbps. With hubs, devices can be up to 30 meters (six cables' worth) away from the host. Here the high-speed cables for 12Mbps communication are better shielded than their less expensive 1.5Mbps counterparts. The USB 2.0 specification tells that the cable delay to be less than 5.2 ns per meter

Inside the USB cable there are two wires that supply the power to the peripherals-- +5 volts (red) and ground (brown)-- and a twisted pair (yellow and blue) of wires to carry the data. On the power wires, the computer can supply up to 500 milliamps of power at 5 volts. A peripheral that draws up to 100ma can extract all of its power from the bus wiring all of the time. If the device needs more than a half-amp, then it must have its own power supply. That means low-power devices such as mice can draw their power directly from the bus. High-power devices such as printers have their own power supplies and draw minimal power from the bus. Hubs can have their own power supplies to provide power to devices connected to the hub.

Pin No:	Signal	Color of the cable
1	+5V power	Red
2	- Data	White / Yellow
3	+Data	Green / Blue
4	Ground	Black/Brown

Table - 1: USB pin connections

USB hosts and hubs manage power by enabling and disabling power to individual devices to electrically remove ill-behaved peripherals from the system. Further, they can instruct devices to enter the suspend state, which reduces maximum power consumption to 500 microamps (for low-power, 1.5Mbps peripherals) or 2.5ma for 12Mbps devices.

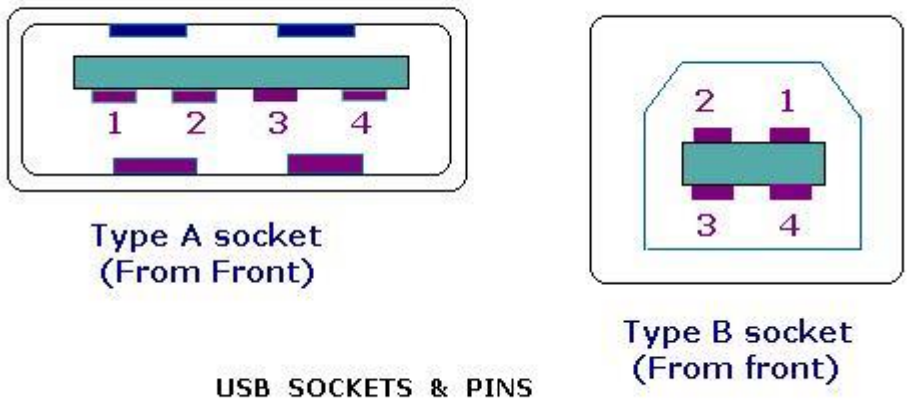


Fig 3: USB Type A & B Connectors

In short, the USB is a serial protocol and physical link, which transmits all data differentially on a single pair of wires. Another pair provides power to downstream peripherals.

Note that although USB cables having a Type A plug at each end are available, they should never be used to connect two PCs together, via their USB ports. This is because a USB network can only have one host, and both would try to claim that role. In any case, the cable would also short their 5V power rails together, which could cause a damaging current to flow. USB is not designed for direct data transfer between PCs. But the "sharing hubs" technique allows multiple computers to access the same peripheral device(s) and work by switching access between PCs, either automatically or manually.

USB Electrical signaling

The serial data is sent along the USB in differential or push-pull mode, with opposite polarities on the two signal lines. This improves the signal-to-noise ratio by doubling the effective signal amplitude and also allowing the cancellation of any common-mode noise induced into the cable. The data is sent in non-return-to-zero (NRTZ) format. To ensure a minimum density of signal transitions, USB uses bit stuffing. I.e.: an extra 0 bit is inserted into the data stream after any appearance of six consecutive 1 bits. Seven consecutive 1 bits is always considered as an error.

The low speed/full speed USB bus (twisted pair data cable) has characteristic impedance of 90 ohms +/- 15%. The data cable signal lines are labeled as D+ and D-. Transmitted signal levels are as follows.

1. 0.0V to 0.3V for low level and 2.8V to 3.6V for high level in Full Speed (FS) and Low Speed (LS) modes
2. -10mV to 10 mV for low level and 360mV to 440 mV for high level in High Speed (HS) mode.

In FS mode the cable wires are not terminated, but the HS mode has termination of 45O to ground, or 90O differential to match the data cable impedance.

As we already discussed, the USB connection is always between a host / hub at the "A" connector end, and a device or hub's upstream port at the other end. The host includes 15 kO pull-down resistors on each data line. When no device is connected, this pulls both data lines low into the so-called "single-ended zero" state (SE0), and indicates a reset or disconnected connection.

A USB device pulls one of the data lines high with a 1.5kO resistor. This overpowers one of the pull-down resistors in the host and leaves the data lines in an idle state called "J". The choice of data line indicates a device's speed support; full-speed devices pull D+ high, while low-speed devices pull D- high. In fact the data is transmitted by toggling the data lines between the J state and the opposite K state.

A USB bus is reset using a prolonged (10 to 20 milliseconds) SE0 signal. USB 2.0 devices use a special protocol during reset, called "chirping", to negotiate the High-Speed mode with the host/hub. A device that is HS capable first connects as an FS device (D+ pulled high), but upon receiving a USB RESET (both D+ and D- driven LOW by host for 10 to 20 mS) it pulls the D- line high. If the host/hub is also HS capable, it chirps (returns alternating J and K states on D- and D+ lines) letting the device know that the hub will operate at High Speed.

How do they communicate?

When a USB peripheral device is first attached to the network, a process called enumeration process gets started. This is the way by which the host communicates with the device to learn its identity and to discover which device driver is required. The enumeration starts by sending a reset signal to the newly connected USB device. The speed of the USB device is determined during the reset signaling. After reset, the host reads the USB device's information, and then the device is assigned a unique 7-bit address (will be discussed in next section). This avoids the DIP-switch and IRQ headaches of the past device communication methods. If the device is supported by the host, the device drivers needed for communicating with the device are loaded and the device is set to a configured state. Once a hub detects a new peripheral (or even the removal of one), it actually reports the new information about the peripheral to the host, and enables communications with it. If the USB host is restarted, the enumeration process is repeated for all connected devices.

In other words, the enumeration process is initiated both when the host is powered up and a device connected or removed from the network.

Technically speaking, the USB communications takes place between the host and endpoints located in the peripherals. An endpoint is a uniquely addressable portion of the peripheral that is the source or receiver of data. Four bits define the device's endpoint address; codes also indicate transfer direction and whether the transaction is a "control" transfer (will be discussed later in detail). Endpoint 0 is reserved for control transfers, leaving up to 15 bi-directional destinations or sources of data within each device. All devices must support endpoint zero. Because this is the endpoint, which receives all of the devices control, and status requests during enumeration and throughout the duration while the device is operational on the bus.

All the transfers in USB occur through virtual pipes that connect the peripheral's endpoints with the host. When establishing communications with the peripheral, each endpoint returns a descriptor, a data structure that tells the host about the endpoint's configuration and expectations. Descriptors include transfer type, max size of data packets, perhaps the interval for data transfers, and in some cases, the bandwidth needed. Given this data, the host establishes connections to the endpoints through virtual pipes.

Though physically configured as a tiered star, logically (to the application code) a direct connection exists between the host and each device.

The host controller polls the bus for traffic, usually in a round-robin fashion, so no USB device can transfer any data on the bus without an explicit request from the host controller.

USB can support four data transfer types or transfer mode, which are listed below.

1. Control
2. Isochronous
3. Bulk
4. Interrupt

Control transfers exchange configuration, setup and command information between the device and the host. The host can also send commands or query parameters with control packets.

Isochronous transfer is used by time critical, streaming device such as speakers and video cameras. It is time sensitive information so, within limitations, it has guaranteed access to the USB bus. Data streams between the device and the host in real-time, and so there will not be any error correction.

Bulk transfer is used by device like printers & scanners, which receives data in one big packet. Here the timely delivery is not critical. Bulk transfers are fillers, claiming unused USB bandwidth when nothing more important is going on. The error correction protects these packets.

Interrupt transfers is used by peripherals exchanging small amounts of data that need immediate attention. It is used by devices to request servicing from the PC/host. Devices like a mouse or a keyboard comes in this category. Error checking validates the data.

As devices are enumerated, the host is keeping track of the total bandwidth that all of the isochronous and interrupt devices are requesting. They can consume up to 90 percent of the 480 Mbps of bandwidth that is available. After 90 percent is used up, the host denies access to any other isochronous or interrupt devices. Control packets and packets for bulk transfers use any bandwidth left over (at least 10 percent).

The USB divides the available bandwidth into frames, and the host controls the frames. Frames contain 1,500 bytes, and a new frame starts every millisecond. During a frame, isochronous and interrupt devices get a slot so they are guaranteed the bandwidth they need. Bulk and control transfers use whatever space is left.

USB packets & formats

All USB data is sent serially, of course, and least significant bit (LSB) first. USB data transfer is essentially in the form of packets of data, sent back and forth between the host and peripheral devices. Initially, all packets are sent from the host, via the root hub and possibly more hubs, to devices. Some of those packets direct a device to send some packets in reply.

Each USB data transfer consists of a

1. Token Packet (Header defining what it expects to follow)
2. Optional Data Packet, (Containing the payload)
3. Status Packet (Used to acknowledge transactions and to provide a means of error correction)

As we have already discussed, the host initiates all transactions. The first packet, also called a token is generated by the host to describe what is to follow and whether the data transfer will be a read or write and what the device's address and designated endpoint is. The next packet is generally a data packet carrying the content information and is followed by a handshaking packet, reporting if the data or token was received successfully, or if the endpoint is stalled or not available to accept data.

USB packets may consist of the following fields:

1. Sync field: All the packets start with this sync field. The sync field is 8 bits long at low and full speed or 32 bits long for high speed and is used to synchronize the clock of the receiver with that of the transmitter. The last two bits indicate where the PID fields starts.

2. PID field: This field (Packet ID) is used to identify the type of packet that is being sent. The PID is actually 4 bits; the byte consists of the 4-bit PID followed by its bit-wise complement, making an 8-bit PID in total. This redundancy helps detect errors.

3. ADDR field: The address field specifies which device the packet is designated for. Being 7 bits in length allows for 127 devices to be supported.

4. ENDP field: This field is made up of 4 bits, allowing 16 possible endpoints. Low speed devices however can only have 2 additional endpoints on top of the default pipe.

5. CRC field: Cyclic Redundancy Checks are performed on the data within the packet payload. All token packets have a 5-bit CRC while data packets have a 16-bit CRC.

6. EOP field: This indicates End of packet. Signaled by a Single Ended Zero (SE0) for approximately 2 bit times followed by a J for 1 bit time.

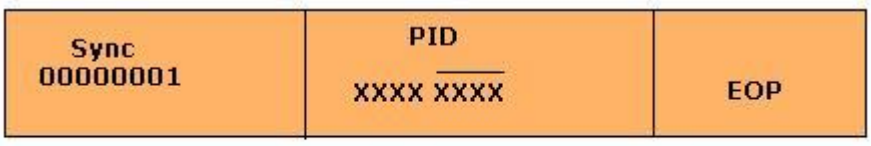
The USB packets come in four basic types, each with a different format and CRC field:

1. Handshake packets
2. Token packets
3. Data packets
4. PRE packet
5. Start of Frame Packets

Handshake packets:

Handshake packets consist of a PID byte, and are generally sent in response to data packets. The three basic types of handshake packets are

1. ACK, indicating that data was successfully received,
2. NAK, indicating that the data cannot be received at this time and should be retried,
3. STALL, indicating that the device has an error and will never be able to successfully transfer data until some corrective action is performed.



HANDSHAKE PACKET FORMAT

Fig 4: Handshake packet format

USB 2.0 added two additional handshake packets.

1. NYET which indicates that a split transaction is not yet complete,
2. ERR handshake to indicate that a split transaction failed.

The only handshake packet the USB host may generate is ACK; if it is not ready to receive data, it should not instruct a device to send any.

Token packets:

Token packets consist of a PID byte followed by 11 bits of address and a 5-bit CRC. Tokens are only sent by the host, not by a device.

There are three types of token packets.

1. In token - Informs the USB device that the host wishes to read information.
2. Out token- informs the USB device that the host wishes to send information.
3. Setup token - Used to begin control transfers.

IN and OUT tokens contain a 7-bit device number and 4-bit function number (for multifunction devices) and command the device to transmit DATA-packets, or receive the following DATA-packets, respectively.

An IN token expects a response from a device. The response may be a NAK or STALL response, or a DATA frame. In the latter case, the host issues an ACK handshake if appropriate. An OUT token is followed immediately by a DATA frame. The device responds with ACK, NAK, or STALL, as appropriate.

SETUP operates much like an OUT token, but is used for initial device setup.

Fig 5: Token packet format

USB 2.0 added a PING token, which asks a device if it is ready to receive an OUT/DATA packet pair. The device responds with ACK, NAK, or STALL, as appropriate. This avoids the need to send the DATA packet if the device knows that it will just respond with NAK.
USB 2.0 also added a larger SPLIT token with a 7-bit hub number, 12 bits of control flags, and a 5-bit CRC. This is used to perform split transactions. Rather than tie up the high-speed USB bus sending data to a slower USB device, the nearest high-speed capable hub receives a SPLIT token followed by one or two USB packets at high speed, performs the data transfer at full or low speed, and provides the response at high speed when prompted by a second SPLIT token.

Data packets:

There are two basic data packets, DATA0 and DATA1. Both consist of a DATA PID field, 0-1023 bytes of data payload and a 16-bit CRC. They must always be preceded by an address token, and are usually followed by a handshake token from the receiver back to the transmitter.

1. Maximum data payload size for low-speed devices is 8 bytes.
2. Maximum data payload size for full-speed devices is 1023 bytes.
3. Maximum data payload size for high-speed devices is 1024 bytes.
4. Data must be sent in multiples of bytes

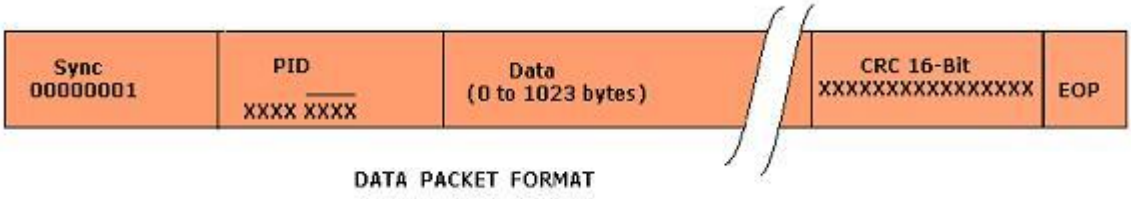


Fig6: Data packet format

USB 2.0 added DATA2 and MDATA packet types as well. They are used only by high-speed devices doing high-bandwidth isochronous transfers, which need to transfer more than 1024 bytes per 125 µs "micro-frame" (8192 kB/s).

PRE packet:

Low-speed devices are supported with a special PID value, PRE. This marks the beginning of a low-speed packet, and is used by hubs, which normally do not send full-speed packets to low-speed devices.
Since all PID bytes include four 0 bits, they leave the bus in the full-speed K state, which is the same as the low-speed J state. It is followed by a brief pause during which hubs enable their low-speed outputs, already idling in the J state, then a low-speed packet follows, beginning with a sync sequence and PID byte, and ending with a brief period of SE0. Full-speed devices other than hubs can simply ignore the PRE packet and its low-speed contents, until the final SE0 indicates that a new packet follows.

Start of Frame Packets:

Every 1ms (12000 full-speed bit times), the USB host transmits a special SOF (start of frame) token, containing an 11-bit incrementing frame number in place of a device address. This is used to synchronize isochronous data flows. High-speed USB 2.0 devices receive 7 additional duplicate SOF tokens per frame, each introducing a 125 µs "micro-frame".

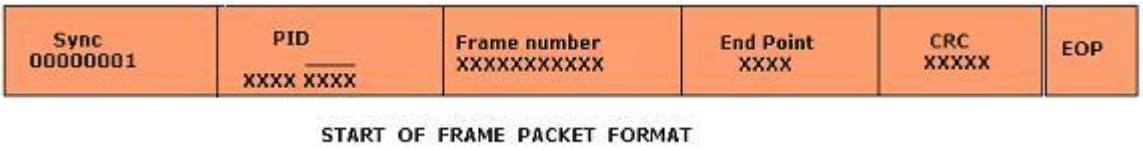


Fig7: Start of Frame packet format

The Host controllers

As we know, the host controller and the root hub are part of the computer hardware. The interfacing between the programmer and this host controller is done by a device called Host Controller Device (HCD), which is defined by the hardware implementer.
In the version 1.x age, there were two competing HCD implementations, Open Host Controller Interface (OHCI) and Universal Host Controller Interface (UHCI). OHCI was developed by Compaq, Microsoft and National Semiconductor. UHCI and its open software stack were developed by Intel. VIA Technologies licensed the UHCI standard from Intel; all other chipset implementers use OHCI. UHCI is more software-driven, making UHCI slightly more processor-intensive than OHCI but cheaper to implement.

With the introduction of USB 2.0 a new Host Controller Interface Specification was needed to describe the register level details specific to USB 2.0. The USB 2.0 HCD implementation is called the Enhanced Host Controller Interface (EHCI). Only EHCI can support hi-speed (480 Mbit/s) transfers. Most of PCI-based EHCI controllers contain other HCD implementations called 'companion host controller' to support Full Speed (12 Mbit/s) and may be used for any device that claims to be a member of a certain class. An operating system is supposed to implement all device classes so as to provide generic drivers for any USB device.
But remember, USB specification does not specify any HCD interfaces. The USB defines the format of data transfer through the port, but not the system by which the USB hardware communicates with the computer it sits in.

Device classes

USB defines class codes used to identify a device's functionality and to load a device driver based on that functionality. This enables a device driver writer to support devices from different manufacturers that comply with a given class code.
There are two places on a device where class code information can be placed. One place is in the Device Descriptor, and the other is in Interface Descriptors. Some defined class codes are allowed to be used only in a Device Descriptor, others can be used in both Device and Interface Descriptors, and some can only be used in Interface Descriptors.

Further developments in USB

USB OTG:

One of the biggest problems with USB is that its host controlled. If we switch off a USB host, nothing else works. Also USB does not support peer-to-peer communication. Let us take an example: Many USB digital cameras can download data to a PC, but it is unable to connect them directly to the USB printer or to a CD Burner, something which is possible with other communication mediums.
To combat these problems, a standard was created to USB 2.0. USB On-The-Go (OTG) was created in 2002. It is actually a supplement to the USB 2.0 specification. USB OTG defines a dual-role device, which can act as either a host or peripheral, and can connect to a PC or other portable devices through the same connector. The OTG specification details the "dual role device", in which a device can function as both a device controller (DC) and/or a host controller (HC).
The OTG host can have a targeted peripheral list. This means the embedded device does not need to have a list of every product and vendor ID or class driver. It can target only one type of peripheral if needed.

Mini, Micro USBs

The OTG specification introduced two additional connectors. One such connector is a mini A/B connector. A dual-role device is required to be able to detect whether a Mini-A or Mini-B plug is inserted by determining if the ID pin (an extra pin introduced by OTG) is connected to ground. The Standard-A plug is approximately 4 x 12 mm, the Standard-B approximately 7 x 8 mm, and the Mini-A and Mini-B plugs approximately 2 x 7 mm. These connectors are used for smaller devices such as PDAs, mobile phones or digital cameras.

The Micro-USB connector was introduced in Jan-2007. It was mainly intended to replace the Mini-USB plugs used in many new smart-phones and PDAs. This Micro-USB plug is rated for approximately 10,000 connect-disconnect cycles. As far as the dimensions are concerned, it is about half the height of the mini-USB connector, but features a similar width.

Pin No:	Name	Description	Color
1	VCC	+5V	Red
2	D-	Data-	White
3	D+	Data+	Green
4	ID	Type –A: Connected to GNDType – B: Not connected	None
5	GND	Ground	Black

Table-2: Mini/Micro plug connection