Universal Serial Bus (USB)

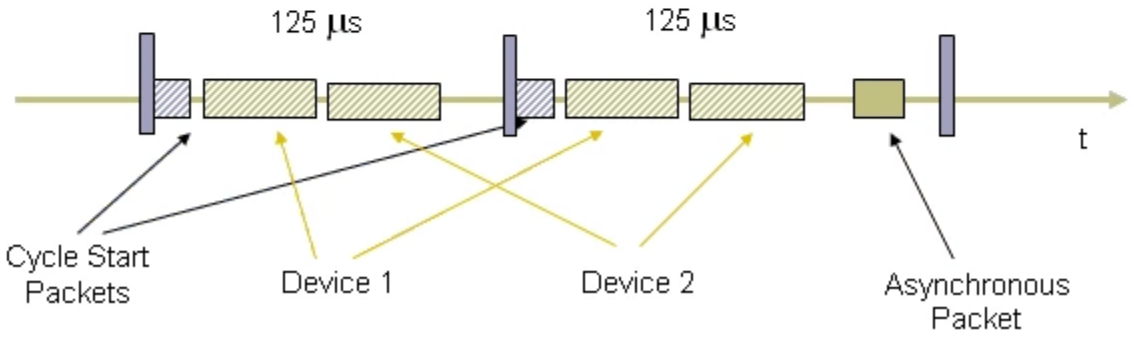
- USB is designed to allow many peripherals to be connected using a single standardized interface.
- provides an expandable, fast, bi-directional, low-cost, hot-pluggable Plug and Play serial hardware interface that makes the life of the computer users easier allowing them to plug different peripheral devices into a USB port and have them automatically configured and ready to use.
- Using a single connector type, USB allows the user to connect a wide range of peripheral devices, such as keyboards, mice, printers, scanners, mass storage devices, telephones, modems, digital still-image cameras, video cameras, audio devices to a computer.
- USB devices do not directly consume system resources.
- USB is an industry standard developed in the mid-1990s that defines the cables, connectors and protocols used for connection, communication and power supply between computers and electronic devices.
- USB has effectively replaced a variety of earlier interfaces, such as serial and parallel ports, as well as separate power chargers for portable devices.

Features of USB

- Single connector type
- Hot-swappable
- Plug and Play
- High performance
- Expandability
- Power supplied from the bus
- Easy to use for end user
- Low-cost implementation
- Wide range of workloads and applications
- Isochronous bandwidth
- Robustness

Isochronous Data Transfer

- Guaranteed bandwidth but not guaranteed data integrity
- Data sent in 125 microsecond slices
- Every device on the bus owns a portion of the slice
- 20% of bandwidth still reserved for asynchronous transfer



USB Standards

USB 1.0

- Released in January 15, 1996. Specified data rates of 1.5 Mbit/s (Low-Bandwidth) and 12 Mbit/s (Full-Bandwidth). Does not allow for extension cables or pass-through monitors (due to timing and power limitations). Few such devices actually made it to market.
- USB 1.1: Released in September 23, 1998. Introduced the improved specification and was the first widely used version of USB. Fixed problems identified in 1.0, mostly relating to hubs. Earliest revision to be widely adopted.

USB 2.0

- The USB 2.0 specification was released in April 27, 2000 and was ratified by the USB Implementers Forum (USB-IF) at the end of 2001.
- The major feature of revision 2.0 was the addition of a high-speed transfer rate of 480 Mbit/s. USB 2.0 supports three speeds namely High Speed - 480Mbits/s, Full Speed - 12Mbits/s and Low Speed - 1.5Mbits/s with one host per bus (at a time).

USB 3.0

- The USB 3.0 specification was published on 12 November 2008.
- Brings significant performance enhancements to the USB standard while offering backward compatibility with the peripheral devices currently in use. Legacy USB 1.1/2.0 devices continue to work while plugged into new USB 3.0 host and new USB 3.0 devices work at USB 2.0 speed while plugged into USB 2.0 host.
- Delivering data transfer rates up to ten times faster (the raw throughput is up to 5.0 Gbit/s) than Hi-Speed USB (USB 2.0), SuperSpeed USB is the next step in the continued evolution of USB technology.
- Its main goals were to increase the data transfer rate (up to 5 Gbit/s), to decrease power consumption, to increase power output, and to be backwards-compatible with USB 2.0. USB 3.0 includes a new, higher speed bus called SuperSpeed in parallel with the USB 2.0 bus. For The first USB 3.0 equipped devices were presented in January 2010
- Transfer of 25 GB file in approx 70 seconds
- Extensible – Designed to scale > 25Gbps
- Optimized power efficiency
  - No device polling (asynchronous notifications)
  - Lower active and idle power requirements
- Backward compatible with USB 2.0
  - USB 2.0 device will work with USB 3.0 host
  - USB 3.0 device will work with USB 2.0 host

Wireless USB

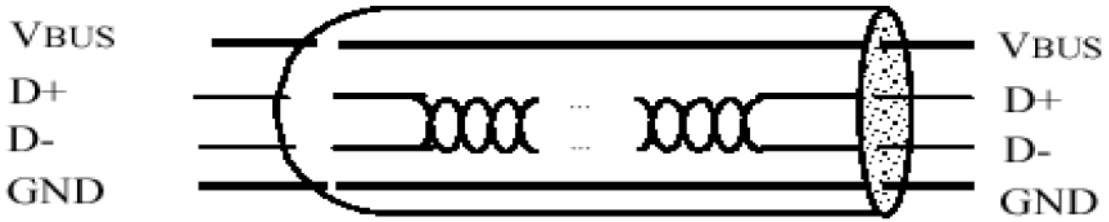
- Released in May 12, 2005 which uses UWB (Ultra Wide Band) as the radio technology.
- 480 M bits/sec up to 3m
- 110 m bits/sec up to 10m
- Signals, Throughput & Protocol

USB Interconnect

- Bus Topology: Connection model between USB devices and the host.
- Inter-layer Relationships: In terms of a capability stack, the USB tasks are performed at each layer in the system.
- Data Flow Models: The manner in which data moves in the system over the USB between producers and consumers.
- USB Schedule: The USB provides a shared interconnect. Access to the interconnect is scheduled in order to support isochronous data transfers and to eliminate arbitration overhead.

Fig: 'A' Plug, 'B' Plug and 'Mini-B' Plug

Signals



Pin	Color	Name	Description
1	Red	Vcc	+5V dc
2	White	D-	Data-
3	Green	D+	Data+
4	Black	GND	Ground

Fig: USB electrical signals

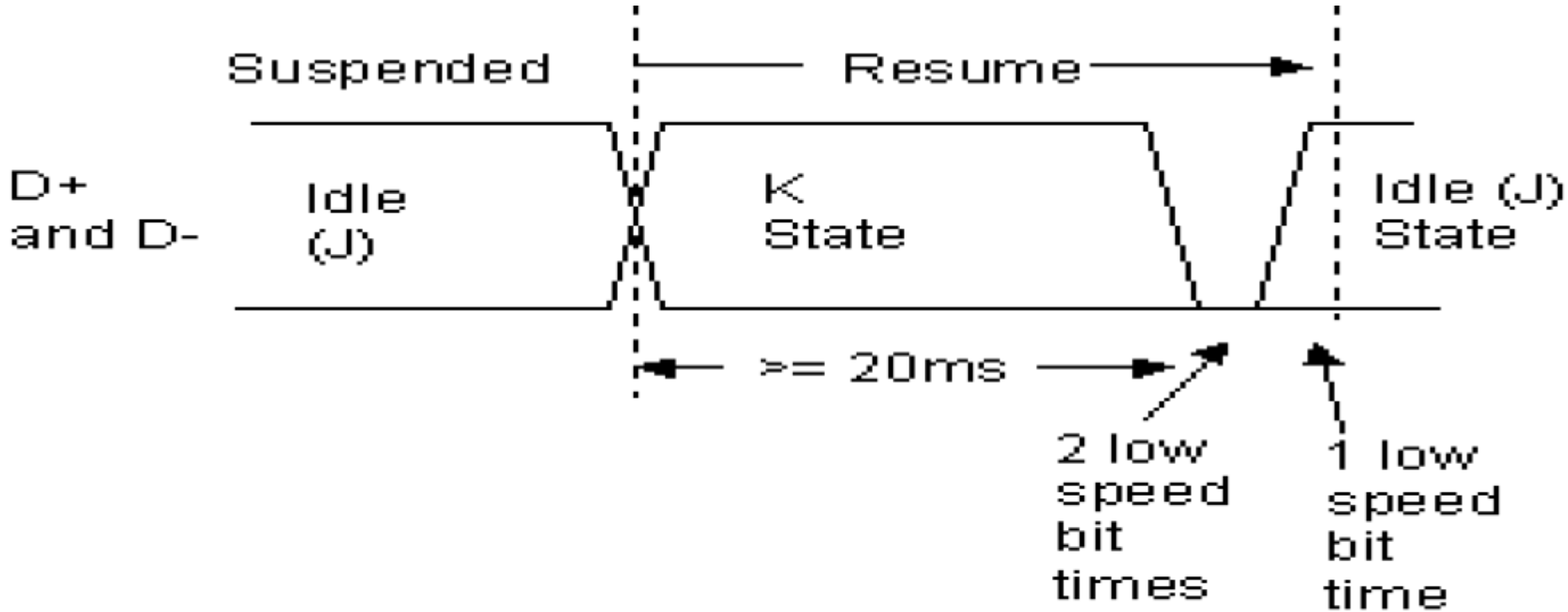


Fig: USB signals and states

Source: https://www.usbmadesimple.co.uk/ums_3.htm

Bus State	Levels
Differential '1'	D+ high, D- low
Differential '0'	D- high, D+ low
Single Ended Zero (SE0)	D+ and D- low
Single Ended One (SE1)	D+ and D- high
Data J State: Low-speed Full-speed	 Differential '0' Differential '1'
Data K State: Low-speed Full-speed	 Differential '1' Differential '0'
Idle State: Low-speed Full-speed	 D- high, D+- low D+ high, D- low
Resume State	Data K state
Start of Packet (SOP)	Data lines switch from idle to K state

Bus State	Levels
End of Packet (EOP)	SE0 for 2 bit times followed by J state for 1 bit time
Disconnect	SE0 for >= 2us
Connect	Idle for 2.5us
Reset	SE0 for >= 2.5 us

J, K and SE0 States

To make it easier to talk about the states of the data lines, some special terminology is used. The 'J State' is the same polarity as the idle state (the line with the pull-up resistor is high, and the other line is low), but is being driven to that state by either host or device.

The K state is just the opposite polarity to the J state.

The Single Ended Zero (SE0) is when both lines are being pulled low.

The J and K terms are used because for Full Speed and Low Speed links they are actually of opposite polarity.

Single Ended One (SE1)

This is the **illegal** condition where both lines are high. It should never occur on a properly functioning link.

Signals (Contd..)

Reset

- When the host wants to start communicating with a device it will start by applying a 'Reset' condition which sets the device to its default configured state.
- The Reset condition involves the host pulling down both data lines to low levels (SE0) for at least 10 ms. The device may recognize the reset condition after 2.5 us.
- This 'Reset' should not be confused with a micro-controller power-on type reset. It is a USB protocol reset to ensure that the device USB signalling starts from a known state.

EOP signal

- The End of Packet (EOP) is an SE0 state for 2 bit times, followed by a J state for 1 bit time.

Suspend

- One of the features of USB which is an essential part of today's emphasis of 'green' products is its ability to power down an unused device. It does this by suspending the device, which is achieved by not sending anything to the device for 3 ms.
- Normally a SOF packet (at full speed) or a Keep Alive signal (at low speed) is sent by the host every 1 ms, and this is what keeps the device awake.
- A suspended device may draw no more than 0.5 mA from V_{bus}.
- A suspended device must recognise the resume signal, and also the reset signal.

Signals (Contd..)

Resume

- When the host wants to wake the device up after a suspend, it does so by reversing the polarity of the signal on the data lines for at least 20ms. The signal is completed with a low speed end of packet signal.
- It is also possible for a device with its remote wakeup feature set, to initiate a resume itself. It must have been in the idle state for at least 5ms, and must apply the wakeup K condition for between 1 and 15 ms. The host takes over the driving of the resume signal within 1 ms.

Keep Alive Signal

- This is represented by a Low speed EOP. It is sent at least once every millisecond on a low speed link, in order to keep the device from suspending.

Throughput

- Throughput is the actual output of any device, USB's actual throughput is a function of many variables:
  - Target device's ability to source or sink data
  - Bandwidth consumption by other devices in the bus
  - Efficiency of host's USB ports
  - Types of data

Signals (Contd..)

Speed

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.

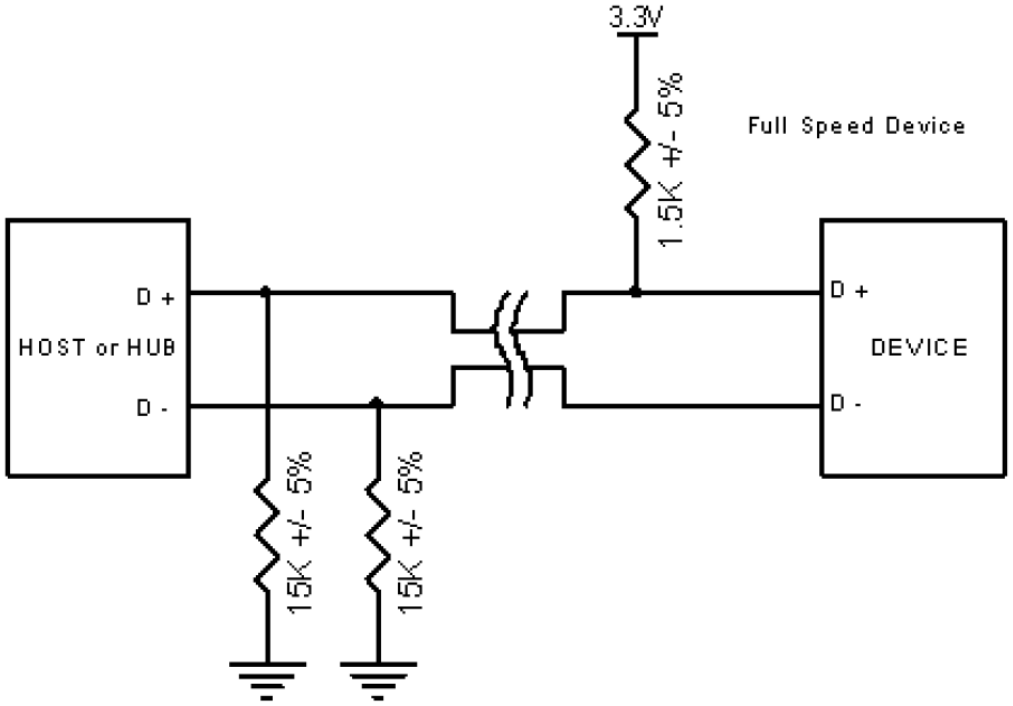


Figure : Full Speed Device with pull up resistor connected to D+

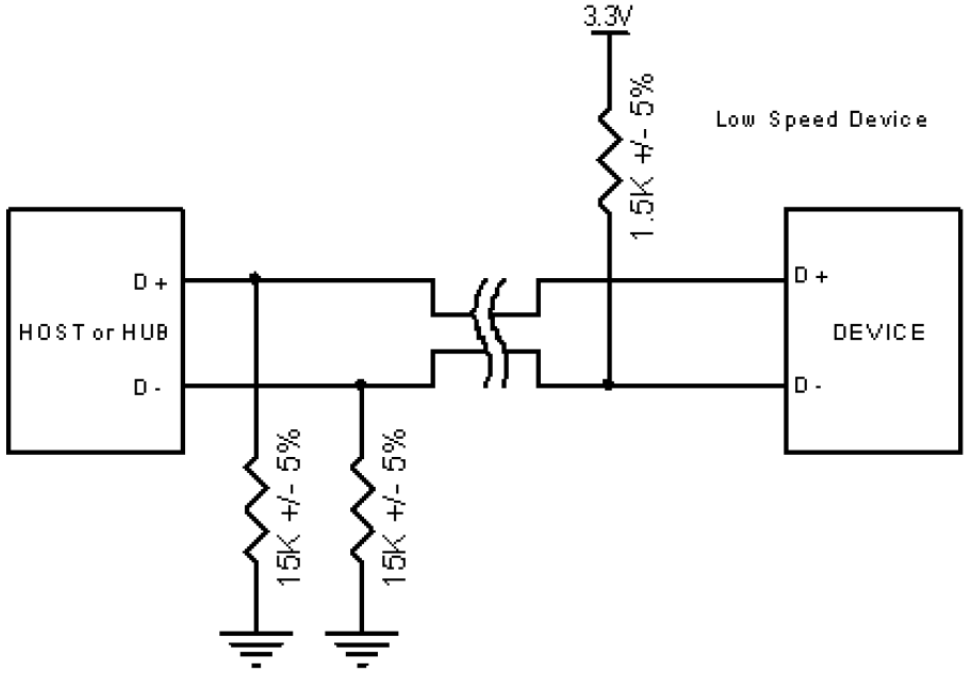


Figure : Low Speed Device with pull up resistor connected to D-

USB Protocols

Unlike RS-232 and similar serial interfaces where the format of data being sent is not defined, USB is made up of several layers of protocols. Most USB controller I.C.s will take care of the lower layer, thus making it almost invisible to the end designer.

Each USB transaction consists of a

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

As we have already discussed, USB is a host centric bus. 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 transaction 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 payload and is followed by an handshaking packet, reporting if the data or token was received successfully, or if the endpoint is stalled or not available to accept data.

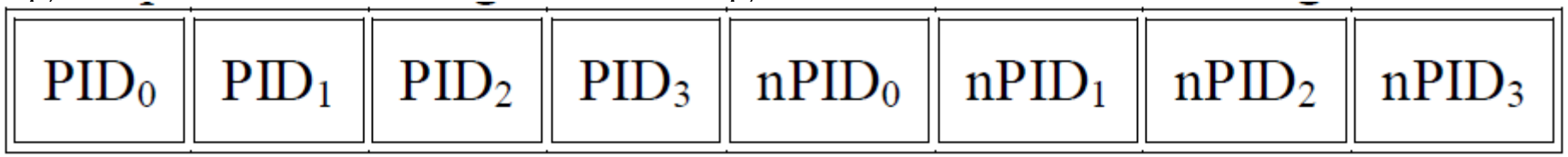
Common USB Packet Fields

Data on the USB bus is transmitted LSB first. USB packets consist of the following fields,

• **Sync**: All packets must start with a 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.

• **PID**: PID stands for Packet ID. This field is used to identify the type of packet that is being sent.

There are 4 bits to the PID, however to insure it is received correctly, the 4 bits are complemented and repeated, making an 8 bit PID in total. The resulting format is shown below.



• **ADDR**: The address field specifies which device the packet is designated for. Being 7 bits in length allows for 127 devices to be supported. Address 0 is not valid, as any device which is not yet assigned an address must respond to packets sent to address zero.

• **ENDP**: The endpoint 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. (4 endpoints max)

• **CRC**: 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.

• **EOP**: End of packet. Signalled by a Single Ended Zero (SE0) for approximately 2 bit times followed by a J for 1 bit time.

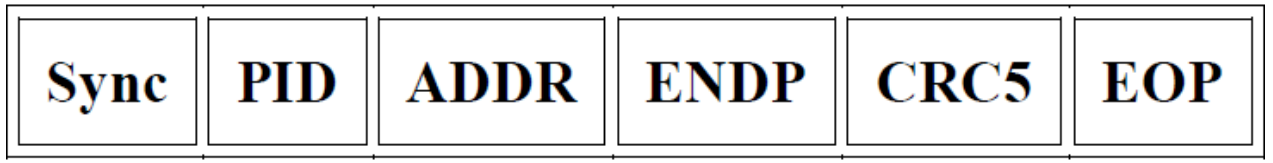
USB Packet Types

USB has four different packet types. Token packets indicate the type of transaction to follow, data packets contain the payload, handshake packets are used for acknowledging data or reporting errors and start of frame packets indicate the start of a new frame.

Token Packets: There are three types of token packets,

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

Token Packets must conform to the following format,

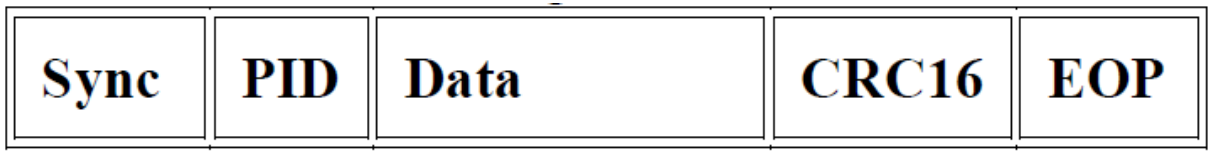


Data Packets: There are two types of data packets each capable of transmitting up to 1024 bytes of data.

- Data0
- Data1

High Speed mode defines another two data PIDs, DATA2 and MDATA.

Data packets have the following format,

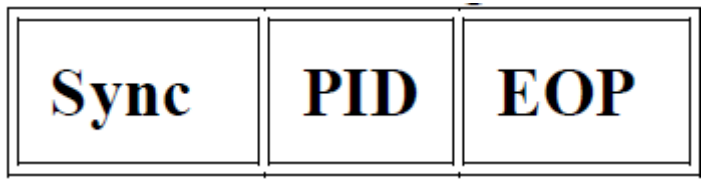


- Maximum data payload size for low-speed devices is 8 bytes.
- Maximum data payload size for full-speed devices is 1023 bytes.
- Maximum data payload size for high-speed devices is 1024 bytes.
- Data must be sent in multiples of bytes.

USB Packet Types

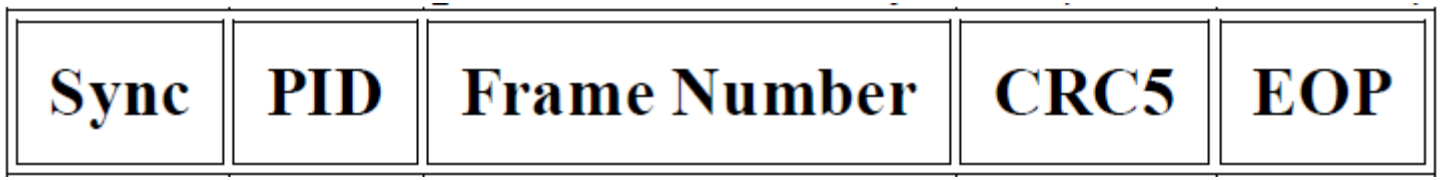
Status / Handshake Packets: There are three type of handshake packets which consist simply of the PID

- ACK - Acknowledgment that the packet has been successfully received.
- NAK - Reports that the device temporary cannot send or received data. Also used during interrupt transactions to inform the host there is no data to send.
- STALL - The device finds its in a state that it requires intervention from the host.
- Handshake Packets have the following format,



Start of Frame Packets

The SOF packet consisting of an 11-bit frame number is sent by the host every 1ms ± 500ns on a full speed bus or every 125 μs ± 0.0625 μs on a high speed bus.



Transfer Model

Endpoints

sources or sinks of data

It is host centric, endpoints occur at the end of the communications channel at the USB function.

At the software layer, your device driver may send a packet to your devices EP1 for example. As the data is flowing out from the host, it will end up in the EP1 OUT buffer. Your firmware will then at its leisure read this data. If it wants to return data, the function cannot simply write to the bus as the bus is controlled by the host. Therefore it writes data to EP1 IN which sits in the buffer until such time when the host sends a IN packet to that endpoint requesting the data. Endpoints can also be seen as the interface between the hardware of the function device and the firmware running on the function device.

All devices must support endpoint zero. 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.

Pipes

While the device sends and receives data on a series of endpoints, the client software transfers data through pipes. **A pipe is a logical connection between the host and endpoint(s)**. Pipes will also have a set of parameters associated with them such as how much bandwidth is allocated to it, what transfer type (Control, Bulk, Iso or Interrupt) it uses, a direction of data flow and maximum packet/buffer sizes. For example the default pipe is a bi-directional pipe made up of endpoint zero in and endpoint zero out with a control transfer type.

USB defines two types of pipes

- **Stream Pipes** have no defined USB format, that is you can send any type of data down a stream pipe and can retrieve the data out the other end. Data flows sequentially and has a pre-defined direction, either in or out. **Stream pipes will support bulk, isochronous and interrupt transfer types**. Stream pipes can either be controlled by the host or device.
- **Message Pipes** have a defined USB format. They are host controlled, which are initiated by a request sent from the host. Data is then transferred in the desired direction, dictated by the request. Therefore message pipes allow data to flow in both directions but will only support control transfers.

Data Flow Types

- Control Transfers:
  - typically used for short, simple commands to the device, and a status response, used e.g. by the bus control pipe number 0
- Bulk Data Transfers:
  - Large sporadic transfers using all remaining available bandwidth (but with no guarantees on bandwidth or latency). A device like a printer, which receives data in one big packet, uses the bulk transfer mode. A block of data is sent to the printer (in 64-byte chunks) and verified to make sure it is correct.
- Interrupt Data Transfers:
  - Devices that need guaranteed quick responses (bounded latency). A device like a mouse or a keyboard, which will be sending very little data, would choose the interrupt mode.
- Isochronous Data Transfers:
  - At some guaranteed speed (often but not necessarily as fast as possible) but with possible data loss A streaming device (such as speakers) uses the isochronous mode. Data streams between the device and the host in real-time, and there is no error correction.

3.7.3 Devices, Hosts and On-The-Go

- The USB is based on a so-called 'tiered star topology' in which there is a single host controller and up to 127 'slave' devices. The host controller is connected to a hub, integrated within the PC, which allows a number of attachment points (often loosely referred to as ports). A further hub may be plugged into each of these attachment points, and so on. However there are limitations on this expansion.
- A device can be plugged into a hub, and that hub can be plugged into another hub and so on. However the maximum number of tiers permitted is six.
- All devices have an upstream connection to the host and all hosts have a downstream connection to the device.
- The length of any cable is limited to 5 metres. This limitation is expressed in the specification in terms of cable delays etc, but 5 metres can be taken as the practical consequence of the specification. This means that a device cannot be further than 30 metres from the PC, and even to achieve that will involve 5 external hubs, of which at least 2 will need to be self-powered.
- So the USB is intended as a bus for devices near to the PC. For applications requiring distance from the PC, another form of connection is needed, such as Ethernet.

3.7.3 Devices, Hosts and On-The-Go

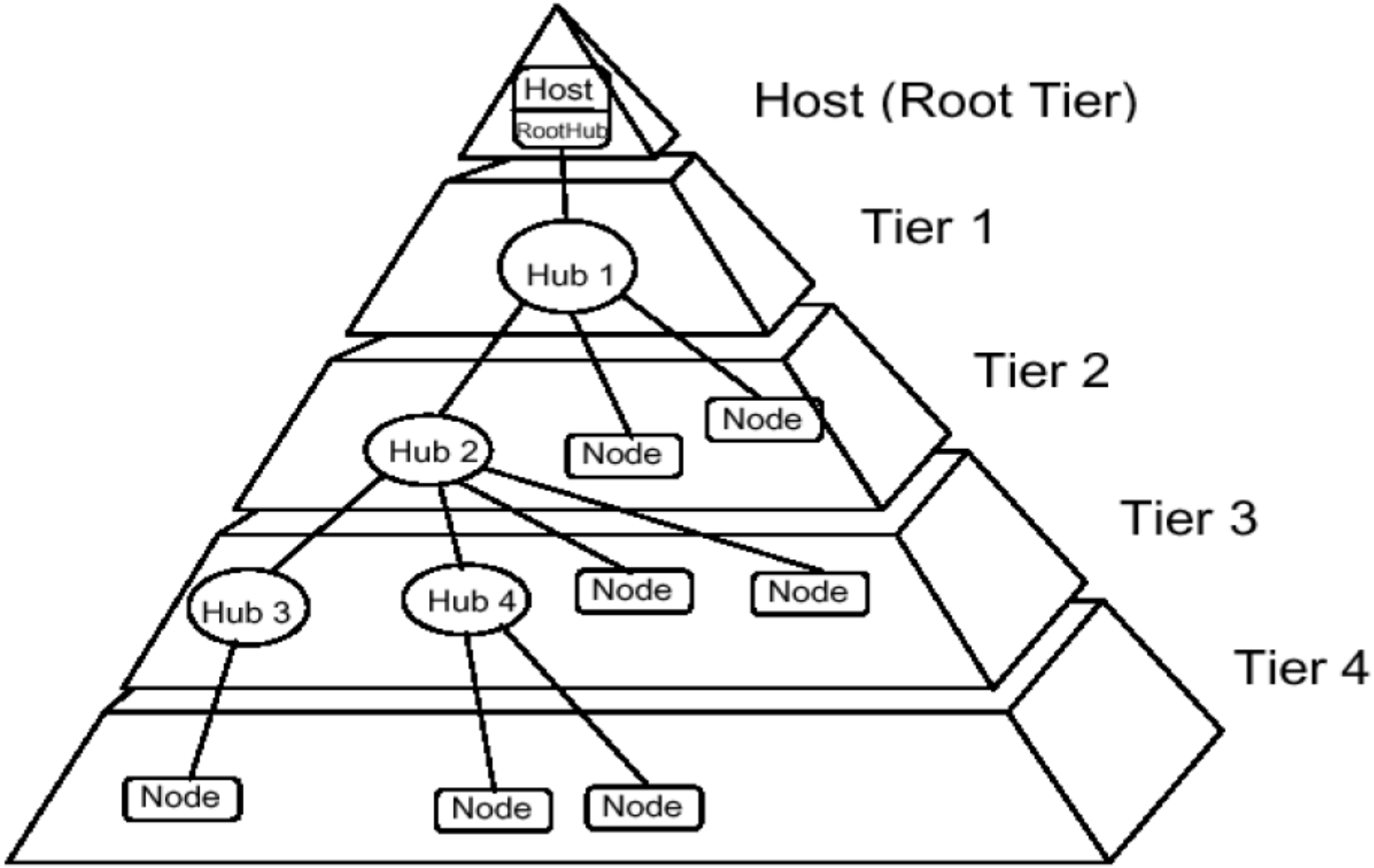


Fig: USB network protocol architecture

3.7.3 Devices, Hosts and On-The-Go

- **Hub**
  - Hub has two major roles: power management and signal distribution.
  - Hubs can be linked, potentially giving you unlimited USB ports to your computer.
  - The biggest difference between types of hubs that is important to know when dealing with USB devices is between un-powered and powered hubs.
- **Powered Hub**
  - Needed when connecting multiple unpowered devices such as mice or digital cameras.
  - These low-powered devices derive their power source from the bus.
  - If too many are connected through a hub, the computer may not be able to handle it.
- **Un-powered Hub**
  - Un-powered hubs can be used with any number of high-power devices such as printers and scanners that have their own power supply, thus not requiring power from the bus.
  - Safe to use with low-power devices (mice, cameras, joysticks, etc.) as long as too many aren't connected as once.

3.7.3 Devices, Hosts and On-The-Go

- **USB On The Go (OTG)**

USB OTG is a new supplement to the USB 2.0 specification that arguments the capability of existing mobile devices and USB peripherals by adding host functionality for connection to USB peripherals. Since USB has traditionally consisted of host-peripheral topology where the PC was the host and the peripheral was the relatively dump device, following new features were needed to upgrade USB technology:

- A new standard for small form factor USB connectors and cables
- The addition of host capability to products that have traditionally been peripherals only, to enable point-to-point connection
- The ability to be either host or peripheral (dual role devices) and to dynamically switch between the two.
- Lowest power requirements to facilitate USB on battery powered devices.

For USB ON-THE-GO (OTG): USES AND SUPPORT, https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1822.html

3.7.3 Devices, Hosts and On-The-Go

- **USB On The Go (OTG) (Contd…)**
- USB On-The-Go (OTG) allows two USB devices to talk to each other without requiring the services of a personal computer (PC). Although OTG appears to add peer-to-peer connections to the USB world, it does not. Instead, USB OTG retains the standard USB host/peripheral model, in which a single host talks to USB peripherals. OTG does introduce, however, the dual-role device, or simply stated a device capable of functioning as either host or peripheral. Part of the magic of OTG is that a host and peripheral can exchange roles if necessary.

For USB ON-THE-GO (OTG): USES AND SUPPORT, https://www.maximintegrated.com/en/design/technical-documents/tutorials/1/1822.html

3.7.4 Interface Chips: USB Device and USB Host

- **Endpoint** is where data enters or leaves the USB system. An IN endpoint is data creator and OUT endpoint is data consumer. For reliable data delivery scheme, need multiple IN and OUT endpoints.
- The collection of endpoints is called an **interface** and is directly related to the real world connection.
- An operating system will have a driver that corresponds to each interface.
- Some devices may have multiple interfaces such as a telephone has a keypad interface and audio interface. Operating system will manage two separate device drivers.
- A collection of interface is called a **configuration**, and only one configuration can be active at a time.
- A configuration defines the attribute and features of a specific model.

3.7.4 Interface Chips: USB Device and USB Host

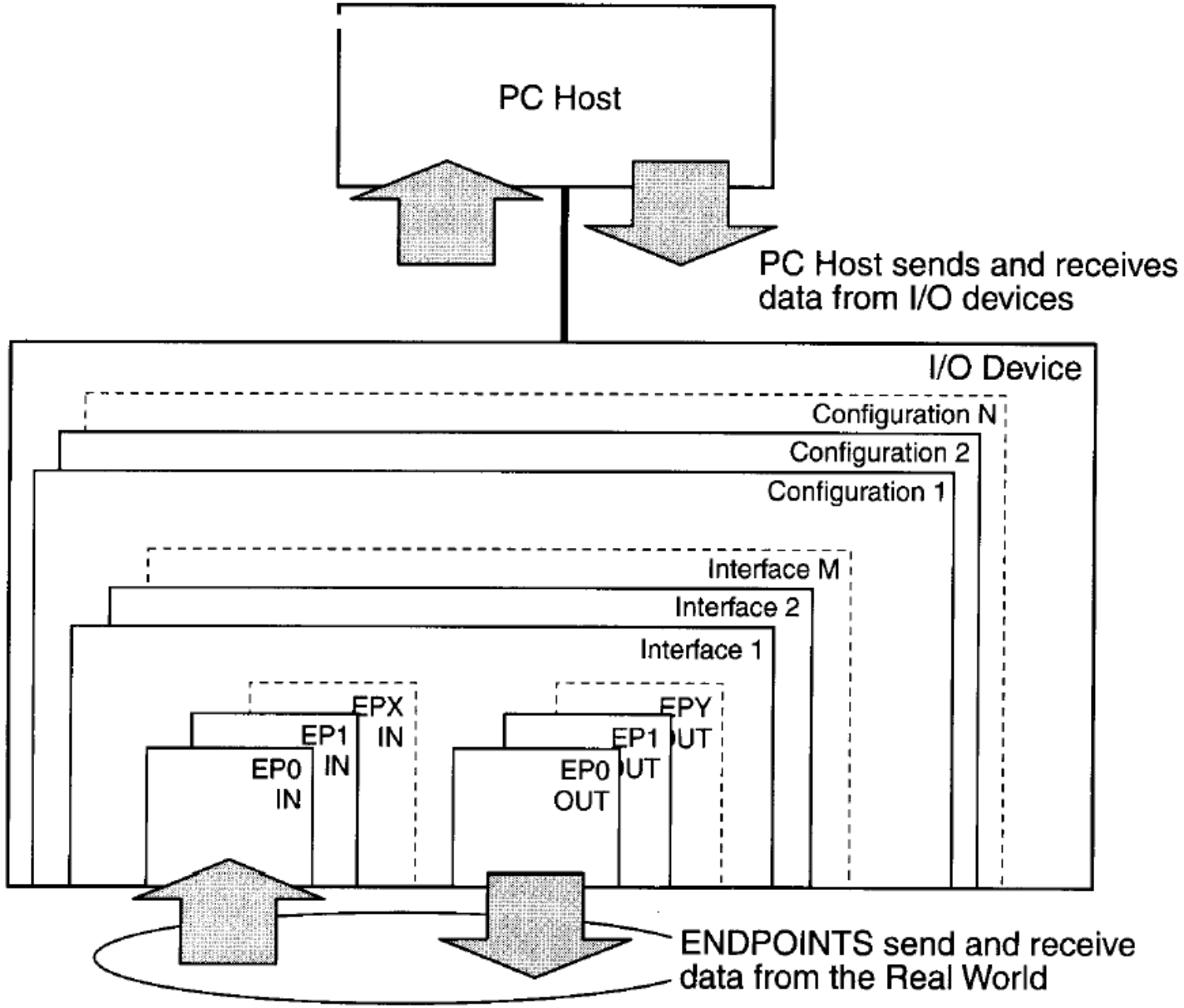


Fig: Logical view of device host interface

3.7.4 Interface Chips: USB Device and USB Host

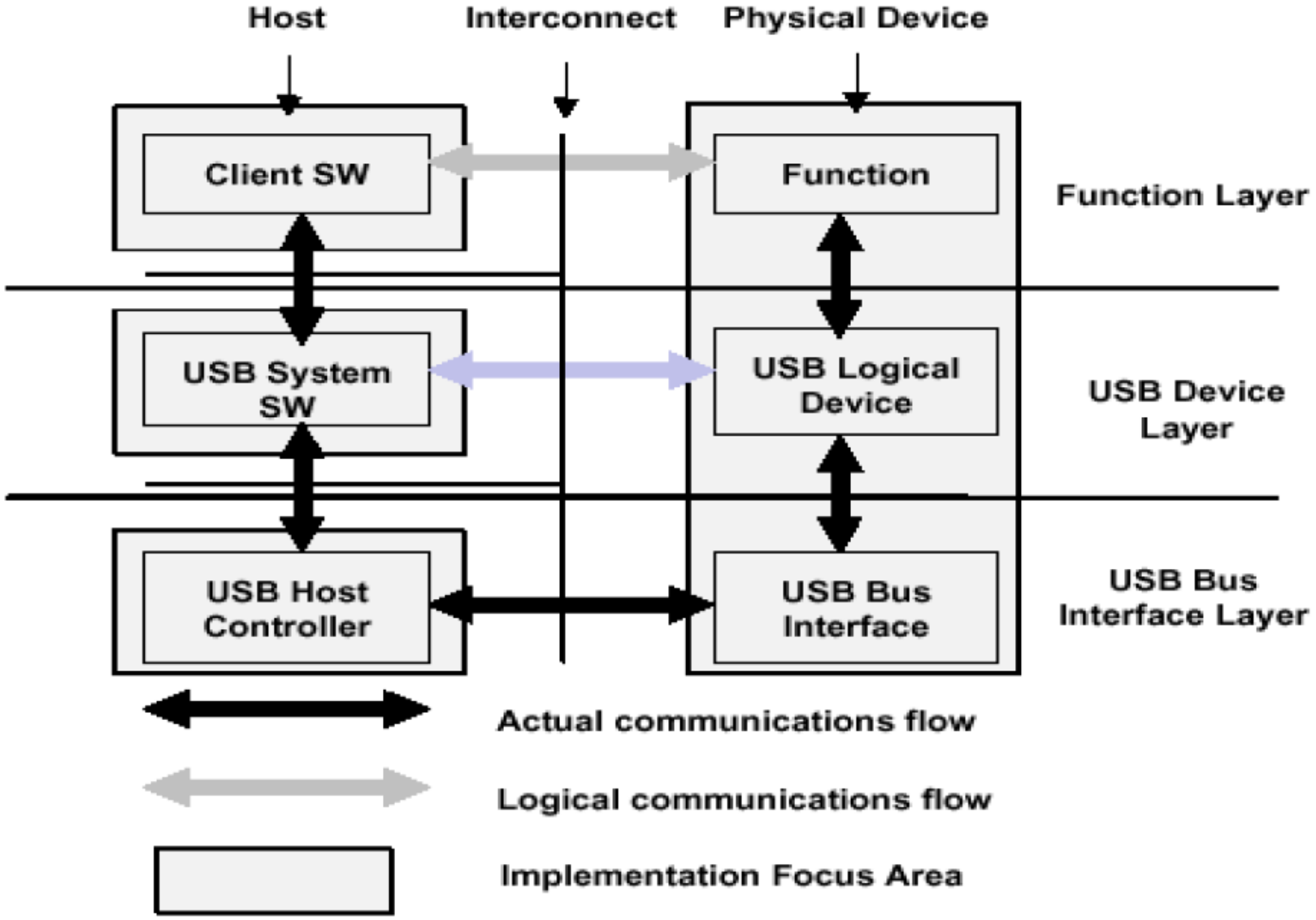


Fig: Interface between device and host

https://www.sciencedirect.com/topics/computer-science/host-controller
https://www.techdesignforums.com/practice/technique/usb-3-0-functional-layer/