

USB TECHNOLOGY: PAST, PRESENT AND FUTURE

WAQAR HUMAYAN, SENIOR TECHNICAL LEAD, MENTOR GRAPHICS



E M B E D D E D S O F T W A R E

W H I T E P A P E R

www.mentor.com

INTRODUCTION

USB is the most popular connectivity technology of our time. Although originally designed for the personal computer to get rid of heterogeneous connectivity options (serial, parallel, or PS2 ports), it has become tremendously popular in many of today's more popular gadgets. Nowadays, no matter what the target market when it comes to connectivity, USB is the obvious choice.

There are two main reasons for USB's widespread acceptance: 1) plug and play, and 2) it has the flexibility to fit any kind of device while maintaining the same user experience. This uniformity spreads across all aspects of the bus, like mechanical (standard cable and connectors), electrical (signaling and bus power), and software (drivers and applications). On top of these features, the USB Implementers Forum (USB-IF) has continuously been working to improve the USB standard in terms of speed, better power management, adoption of more device classes, and standardization.

In the following sections, this paper explores the technical details of wired USB standards, USB 2.0 (Hi-Speed) and USB 3.0 (SuperSpeed). At the conclusion, this paper identifies future applications of USB in the light of the new SuperSpeed USB standard, or USB 3.0.

HISTORY AND BACKGROUND

In 1996, USB 1.0 specifications were released. Considering the application requirements at that time, USB offered two levels of bandwidth over a single link. The two levels were 12 Mb/s (Full-Bandwidth) or 1.5 Mb/s (Low-Bandwidth). USB 1.0 was superseded by the USB 1.1 specification released in September, 1998.

As application complexity grew and new connectivity standards were being introduced (such as FireWire), USB-IF released the USB 2.0 specification in April, 2000. USB 2.0 is also called Hi-Speed USB, offering bus bandwidth of 480 Mb/s (around 40 times faster than its predecessor, Full-Speed USB).

USB 2.0 BUS ARCHITECTURE

USB 2.0 is a tiered star topology comprising the following three elements:

Host: The USB host occupies the central position in USB and is responsible for originating all transactions on the bus and configuring the attached devices. There can be only one host on a single USB tree.

Hub: Hubs are a special kind of USB device, used to extend the USB interconnect.

Device: A USB device provides one or more than one useful function to the USB host. Each USB device is addressable at a unique address, which is assigned by the USB host. There can be up to 127 devices connected to a single USB host. This count of 127 includes the hubs which are connected to expand the downstream connections.

OPERATIONAL MODE

From an operational point of view, each USB device appears to a host as a collection of endpoints. An endpoint is the source or sink of data, in other words, each transaction from the USB host is targeted to a particular endpoint of a device. Thus, a single USB device must have at least one endpoint so that communication with the host can take place (it can have a maximum of 16 endpoints). Endpoints can be categorized by their respective attributes, which include the following:

- **Address:** Each endpoint contains a unique address in the USB device.
- **Direction:** Direction of endpoint is always from the host perspective. IN endpoints carry data from a device to a host and OUT endpoints carry data from a host to a device.

- **Maximum Packet Size:** Maximum payload size an endpoint can carry in a packet.
- **Type:** Endpoint type defines the communication behavior of an endpoint.

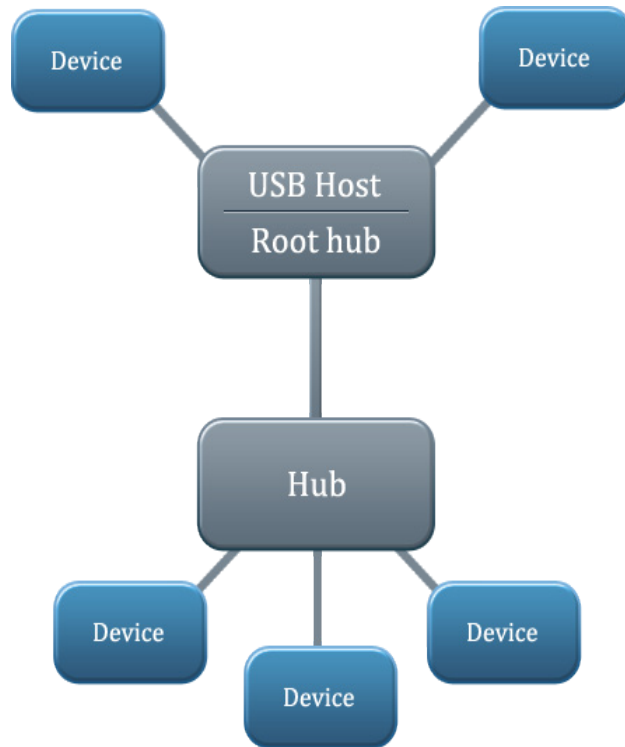


Figure 1: Typical USB topology.

USB supports four types of endpoints which are **Control**, **Bulk**, **Interrupt**, and **Isochronous**. Each of these endpoints differ in protocol constraints. Details on each of the individual endpoints include:

Control: Control endpoints are used to configure and enumerate a peripheral, allowing the host software to interrogate the peripheral and to find out what functionality it has and how it should be accessed. Control endpoints are bidirectional. It is mandatory for each USB device to implement a control endpoint. After connection, each device is initially addressable at default address zero and default control endpoint (i.e. endpoint zero).

Bulk: Reliable data transport that guarantees sequential reception of data. Retransmits are used to ensure the data is received correctly. Depending on the current bus utilization, the throughput with bulk transfers will vary. Typical uses of bulk transport are Disk-On-Keys and printers. Bulk endpoints are unidirectional, they are either IN or OUT. Using a mass storage device as an example, the device needs to implement one bulk IN endpoint (for reading data) and one bulk OUT endpoint (for writing data).

Interrupt: Meant for low bandwidth transfers with periodic transport needs. The host pre-schedules an interval at which time the peripheral is queried. Unlike bulk, this transport type has guaranteed allocation on the bus. The typical use includes keyboards, mouse, track-balls, and game controllers. Using a keyboard as an example, a keyboard device implements an interrupt IN endpoint. The host keeps on polling this endpoint. As soon as the user presses a key, the device prepares a notification, which is sent to the host on the next query, telling the ASCII code of the pressed key.

Isochronous: Guaranteed bandwidth, but not guaranteed data. This method is typically associated with data streaming applications like video cameras and audio speakers. It is used to insure that the data always has a path. USB has a maximum limit on data transfer rates so in order to support such devices, during enumeration and configuration, the host determines if a configuration setting has enough bandwidth to talk with a certain peripheral. If sufficient bandwidth is not available, it rejects the configuration selection in favor of less bandwidth consumption.

Endpoints in a USB device are further grouped to form interfaces. An interface defines a view to a particular function of a USB device to USB host. It's worth noting that although most of the USB devices like mouse, keyboard, or flash drive are single function devices, USB devices can also be multifunctional (single physical USB device with two or more logical functions). These multifunctional devices are called "composite devices" in USB terminology.

USB DEVICE FRAMEWORK

One of the key features of USB is its plug and play ability. USB devices implement a set of descriptors to exchange their capabilities with the USB host. These descriptors convey information such as: what is this USB device? How many possible configurations does it have? What USB functions are supported in each configuration? How many endpoints does it implement in each USB function? And what are the attributes of these endpoints? After receiving these descriptors, the USB host can select a set of capabilities that it supports.

Figure 2 below, offers a view of descriptor hierarchy in a USB device. On the top level, there is a device descriptor which tells the overall capability of the device, and then there can be one or more configuration descriptors. Each configuration can have one or more than one interface descriptors, and each interface descriptor can have one or more than one endpoint descriptors.

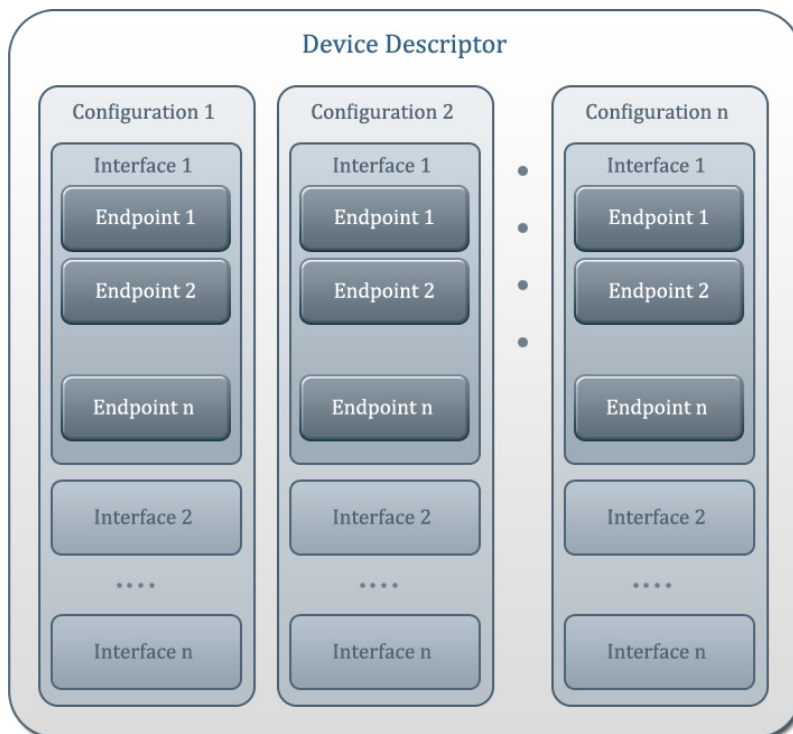


Figure 2: USB descriptor hierarchy.

The USB host fetches these descriptors from the device through standard requests defined by established USB specifications. It is mandatory for every USB device to implement these standard requests. Apart from standard requests, a USB device might also support certain requests that are class-specific and defined in respective class specifications. All standard and class-specific requests are exchanged on the control endpoint of the device.

After connection with host, a USB device transitions from a number of states before it is fully configured and ready to use. The process of configuring a USB device is called "Device Enumeration." The following sequence of operations is carried out by the USB host while enumerating a USB device.

1. The hub in which the device is connected, notifies the USB host about a new device connection.
2. Once the host identifies the port on which device is connected, it waits for some time so that device insertion process is completed and power at device becomes stable.
3. The host resets the device so that it transitions to *Default state*.
4. The host assigns a unique address to the device, this process takes the device to *Addressed state*.
5. The host reads device and configuration descriptors from the device to get an understanding of various attributes of the device and its capabilities.

After analyzing the configuration descriptor (which tells the host how this device will be used), the host software checks for the required software support i.e. the device or class drivers on the host machine. Once the driver support is successfully determined, the host selects a configuration on the USB device. This process takes the device to *Configured state*. Now the device is fully configured/enumerated and ready for use. Figure 3 describes the enumeration process in terms of state transition.

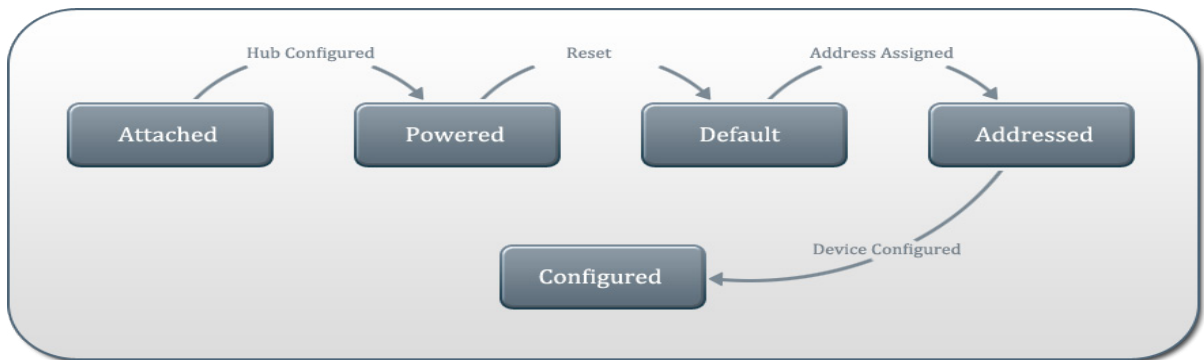


Figure 3: USB state transition during device enumeration.

POWER MANAGEMENT

USB 2.0 only supports device level power management by defining a special state called *Suspend state*. A USB device enters Suspend state either if the hub port it is attached to is suspended, or there is no bus activity for 3 milliseconds. The primary purpose of entering suspend state is to conserve power by shutting down inactive parts of the USB device. The USB device exits the suspend state upon sensing the bus activity (resume signaling). Once powered, the USB device can enter Suspend state from any of the above mentioned states.

FRAME AND MICROFRAME

When scheduling packets on a bus, USB host divides bus time in 1 millisecond intervals called “frame in full-speed mode” and in 125 microsecond intervals called “microframe in high-speed mode.” The start of a frame interval is indicated by a start of frame (SOF) packet sent by the host to device. Frames and microframes help in servicing periodic endpoints (interrupt and isochronous). For example, a full-speed host is sending audio data on an isochronous endpoint. Audio data is encoded as PCM 8KHz, 16-bit stereo samples. This makes around 32000 bytes/second. In order to maintain a constant rate, the USB host can service it as one isochronous packet carrying 32 bytes of data per frame (millisecond).

USB HOST CONTROLLER INTERFACES (OHCI AND EHCI)

USB-IF has specified two register level interfaces for USB host: 1) Open Host Controller Interface (OHCI), and 2) Enhanced Host Controller Interface (EHCI). EHCI can only handle high speed devices hence the need to have at least one OHCI controller to allow low and full speed device operations. These host controller interfaces have native support in all kinds of PCs and modern operating systems.

USB STANDARD DEVICE CLASSES

USB uses the device class paradigm. A USB class specifies the descriptor hierarchy, types of endpoints, and class-specific requests. All USB functions belonging to a particular class are required to adhere to mandatory and optional set of features of that class specification. On the other hand, a class driver running on a USB host knows what to expect from devices of this class and how to use them. This helps interoperability and standardization in a sense that a standard USB class device should work seamlessly with all USB hosts supporting that class.

Current USB-IF device classes include:

- Human Interface Device (HID)
- Mass Storage (MS)
- USB Communication Class (COMM)
- Still Image Class
- USB Audio Class
- USB Video Class

Most of the modern operating systems has native support of USB standard device classes, hence they do not require installation of additional drivers to handle USB standard class devices. It is still possible to design USB devices that are not compliant to any of the defined classes. These devices are called “vendor devices” in USB terminology. Additional drivers are required to be installed on USB host for handling a vendor device. It’s the responsibility of a device vendor to provide USB host drivers for a vendor device.

USB 3.0 - “SUPERSPEED” USB

USB 3.0, or “SuperSpeed” USB is the next evolutionary step towards wired USB. Released in November 2008, USB 3.0 takes theoretical bandwidth of USB to around 5 Gb/s, which is more than 10 times faster than its predecessor USB 2.0 (Hi-Speed USB). This significant increase in speed has made USB 3.0 an extremely lucrative option for bandwidth hungry applications like storage and multimedia.

Although, much like previous USB standards, USB 3.0 is backward compatible, it is not merely an extension of USB 2.0, rather, it’s a completely new bus specification from the hardware point of view. Still, it’s the beauty of design that old class drivers and applications will want to migrate over to USB 3.0 either with no modifications or with minor changes.

BUS ARCHITECTURE

USB 3.0 is dual bus architecture, in which both USB 3.0 and USB 2.0 signals run in parallel to each other. USB hosts enumerate devices on the highest supported signaling rate. Figure 4 gives a detailed overview of USB

3.0 bus architecture. Besides being a dual bus architecture, USB 3.0 shares the same architectural components with USB 2.0 i.e. host, hubs, and devices. USB 3.0 is also a tiered star topology and maintains the same four transfer types; control, bulk, interrupt, and isochronous. Operations of these transfer types in USB 3.0 are very similar from a software perspective, but are entirely different from the protocol point of view.

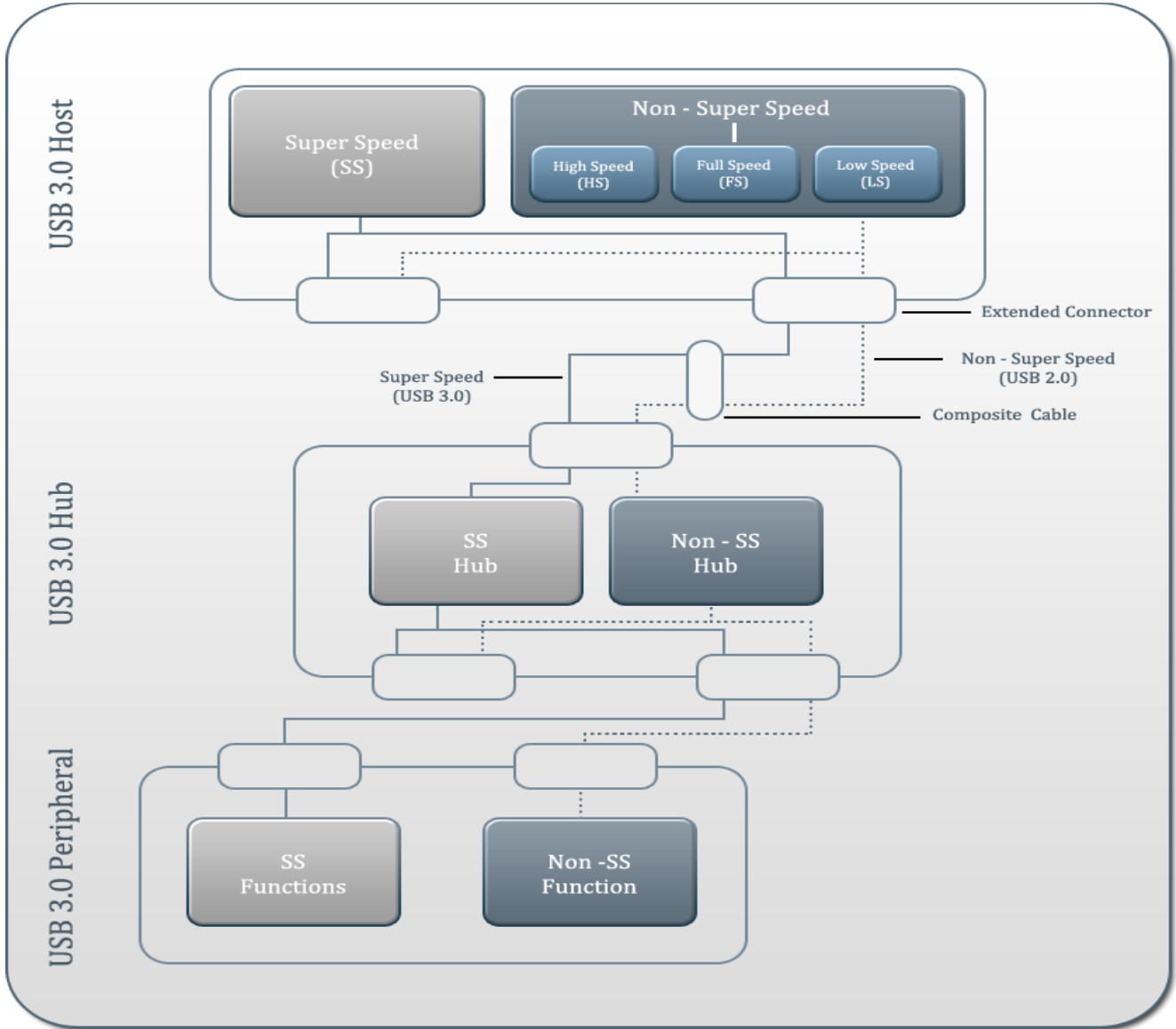


Figure 4: USB 3.0 system overview.

CABLE AND CONNECTORS

USB 3.0 cable carries both USB 2.0 and USB 3.0 signals (as seen in Figure 5). USB 3.0 is a dual, simplex bus which supports simultaneous bidirectional flows. USB 3.0 cable carries two differential pairs for USB 3.0 signaling (SSTX+, SSTX-, and SSRX+, SSRX-) and one differential pair for USB 2.0 signaling (DP, DM). The Figure on page 8 is a comparison between USB 2.0 and USB 3.0 cables. In order to house additional four USB 3.0 signals, USB 3.0 connectors and receptacles have also been modified.

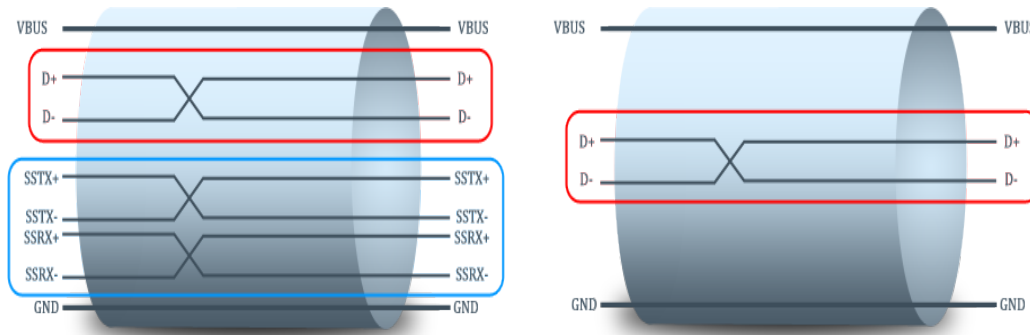


Figure 5: Comparison between USB 3.0 cable (left) and USB 2.0 cable (right).

PACKET ROUTING

Another major difference in USB 3.0 is packet routing. In USB 2.0 and previous standards, packets were broadcast on the bus, each device received all packets and rejected the packets that were not meant for it. In USB 3.0 packets are explicitly routed. This leads to the provision of suspending a link instead of a whole tree under a port. Figure 6 draws a comparison of packet routing between USB 3.0 and USB 2.0.

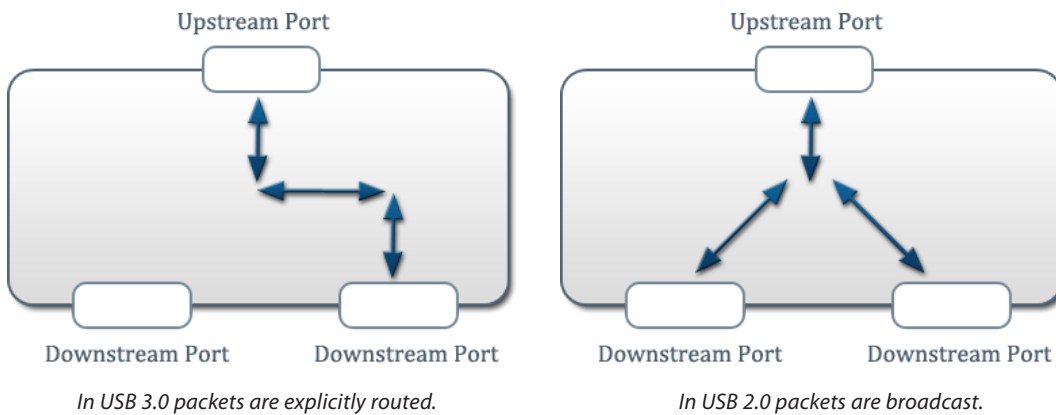


Figure 6: Comparison of USB traffic routing in USB 3.0 and USB 2.0.

BIDIRECTIONAL PROTOCOL FLOW

Up until USB 2.0, USB only supported unidirectional data flow and the direction of bus was swapped after negotiation between host and function. As there are separate Rx and Tx channels in USB 3.0, simultaneous bidirectional flows are now possible. Looking at a bulk IN endpoint in USB 2.0 as an example, the host has no way of determining if the device wants to send the data, therefore it keeps on polling the bulk IN endpoint. If a device has no data to send on this endpoint, then it simply sends a negative acknowledgement handshake (NAK) unless data is available on this endpoint. Benefiting from the bidirectional bus in USB 3.0, the host initially asks for data on a bulk IN endpoint, if the device has no data to send then it simply sends a not ready handshake (NRDY) and the host will no longer poll this endpoint. Once data is available on this endpoint, it will send an endpoint ready handshake (ERDY) to the host. This protocol sequence saves bus bandwidth and

power by allowing the host to stay idle if the device has no data to send. The following figure gives a comparison between bulk IN protocol flow between USB 2.0 and USB 3.0. In order to improve performance of the system, USB 3.0 also uses bursts. Bursts allow an exchange of multiple packets and the receiver can send the status for all packets out of order.

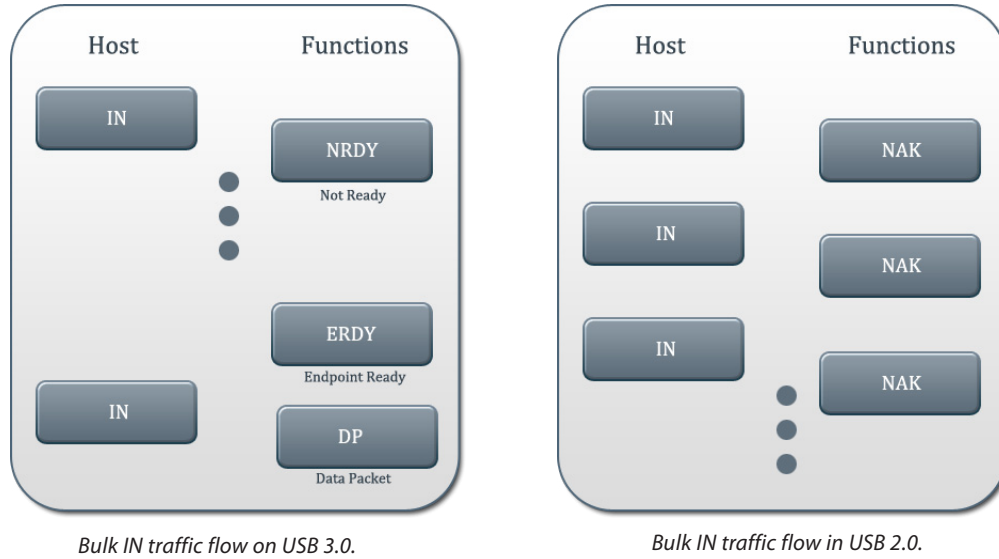


Figure 7: Comparison of USB traffic flow between USB 3.0 and USB 2.0.

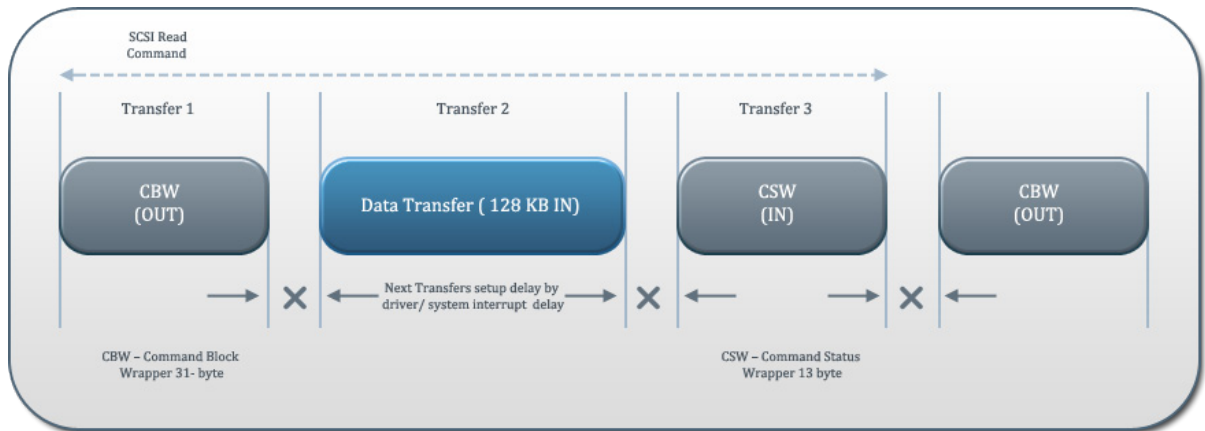
BULK STREAMING

Another significant change in USB 3.0 with respect to achieving better throughput, is “Bulk Streaming.” Bulk streaming is when a standard USB bulk endpoint supports a single stream of data between host and device via a host memory buffer and device endpoint. New USB SuperSpeed provides protocol level support for multi-stream model and uses stream pipe communications mode.

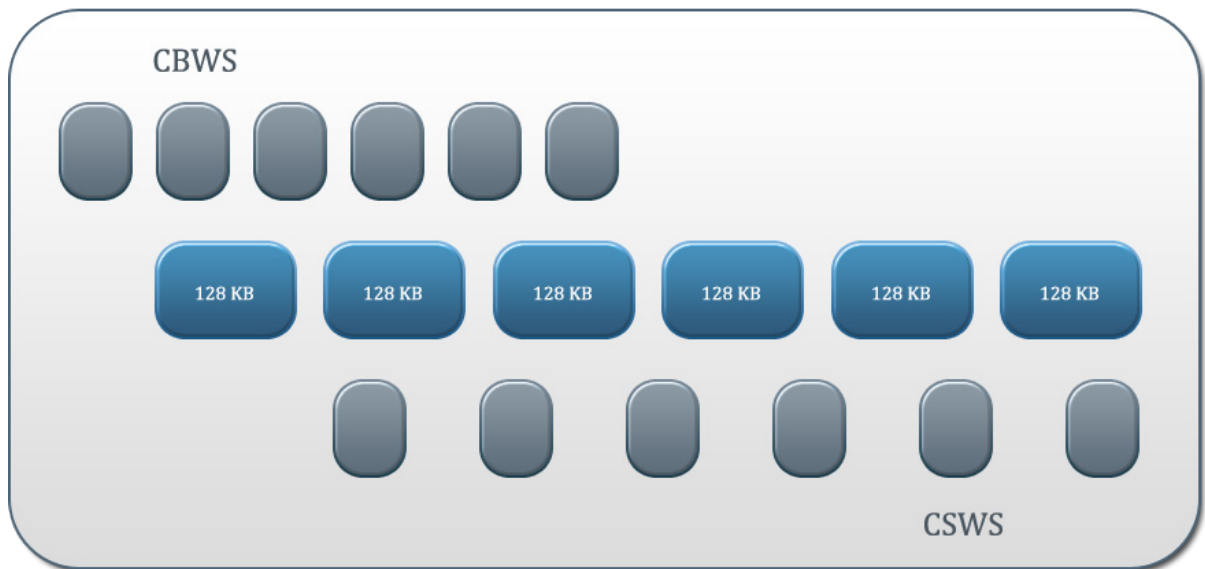
Bulk streaming allows command queuing and out of order completion of these commands. Typical application of this scenario is USB mass storage devices. Streaming allows the host to queue in multiple commands to the device and also allows a device to return data out of order to optimize performance.

In order to better understand this concept, let’s take a look at a USB mass storage device. USB mass storage Bulk Only Transport (BOT) protocol uses one bulk IN and one bulk OUT endpoint. It involves sending a 31 byte Command Block Wrapper (CBW) on bulk OUT endpoint. As a result, data will be exchanged between host and device. If the host wants to read data then the device will send data on bulk IN endpoint otherwise if the host wants to write data, it will send data on bulk OUT endpoint. After the data is exchanged, the device sends a 13 byte Command Status Wrapper (CSW) reporting the result of this command. Observing this protocol keenly reveals that the host has to wait for completion of CBW before moving to the data stage and has to wait for completion of the data stage before scheduling the CSW.

Now let’s look at the same BOT protocol using bulk streaming. The host can send multiple CBWs through one bulk OUT endpoint and data transfers are scheduled through other bulk IN and bulk OUT endpoints. CSWs are scheduled through one other bulk IN endpoint. Each command has stream ID associated with it to identify the transfer. Multiple commands and data transfers can be scheduled and the device and host hardware handle the stream switching on their own. As the host and device do not need to wait for the completion of the previous stage, the host can send back-to-back commands and the device completes these commands as they appear on bus and queue the status of each command. Figure 8 on the following page provides a comparison of mass storage BOT protocol with and without bulk streams.



Without Streams



With Streams

Figure 8: Comparison of mass storage BOT with and without bulk streams.

USB 3.0 - POWER MANAGEMENT

Another important design goal of USB 3.0 is aggressive power management. Despite port level and device level suspension in USB 2.0, USB 3.0 introduces the following multi-level power management attributes:

- **Link:** USB 3.0 allows suspension of a particular link contrary to only port level suspend in USB 2.0.
- **Device:** Device level suspends are same as in USB 2.0.

- Function:** If there is a composite device, then USB 3.0 allows suspending only one function of this device. An example would be a composite device having a mass storage and LAN functions. If the host system has nothing to do on the LAN, and it wants to turn it off while still being able to access the storage contents, it can thus suspend only the LAN function while storage remains active. Since most of the USB devices are bus powered, putting a portion of device to sleep would allow a considerable power savings in the device.

USB 3.0 HUBS

USB 3.0 hubs are special devices in a sense that they operate at both speeds while all other SuperSpeed devices can only operate in one mode at a given time, either SuperSpeed or non SuperSpeed. Once connected with a SuperSpeed host, USB 3.0 hubs enumerate as two distinct devices, one as SuperSpeed and the other as non-SuperSpeed. A SuperSpeed device connected on a downstream port of a USB 3.0 hub connects to a host through SuperSpeed part of hub, while a non-SuperSpeed device connects through the non-SuperSpeed part of the USB 3.0 hub.

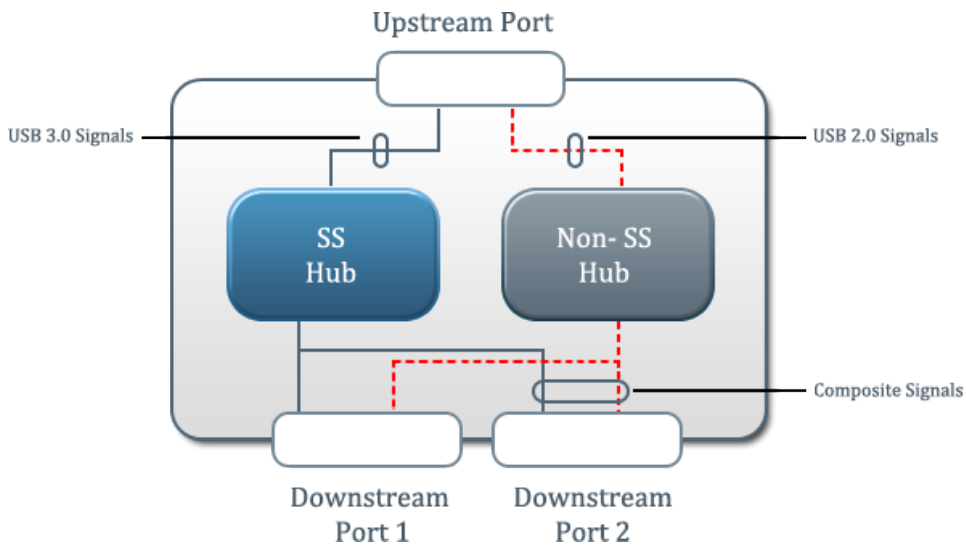


Figure 9: USB 3.0 hub.

XHCI - NEW HOST CONTROLLER INTERFACE

With the release of SuperSpeed USB specifications, USB-IF has also released a new host controller interface called *eXtensible Host Controller Interface* (xHCI). Apart from handling SuperSpeed devices, xHCI alone is capable of handling non-SuperSpeed devices. Today many PCs and laptops are entering the market with native support for xHCI controllers. In addition, xHCI PCIe adapter cards are also becoming available.

FUTURE APPLICATIONS FOR USB

Most PC and embedded vendors are providing USB as the only option for external device connectivity. Almost every kind of device we use in our daily lives, such as a mouse, keyboard, mobile phones, printers, etc. are using USB connectivity. It's anticipated that with SuperSpeed capability, USB will gain momentum in bandwidth hungry applications. The following areas are where USB will excel in future.

STORAGE

So far the storage community is the most excited about USB 3.0. Earlier USB mass storage transport protocol, the “bulk only transport” (BOT) protocol, was sequential in nature. A sequential protocol can’t take maximum benefit of a bidirectional bus and undermines SuperSpeed capability of USB 3.0. USB-IF’s device working group for USB mass storage devices has deprecated the BOT protocol and adopted a new transport protocol standard called **USB Attached SCSI Protocol** (UASP). UASP is fully compatible with the T10 committee’s SCSI architecture model standard and capable of utilizing maximum bandwidth of SuperSpeed USB. Figure 10 is a detailed illustration of standards defined by T10 committee and shows UASP as a transport layer protocol.

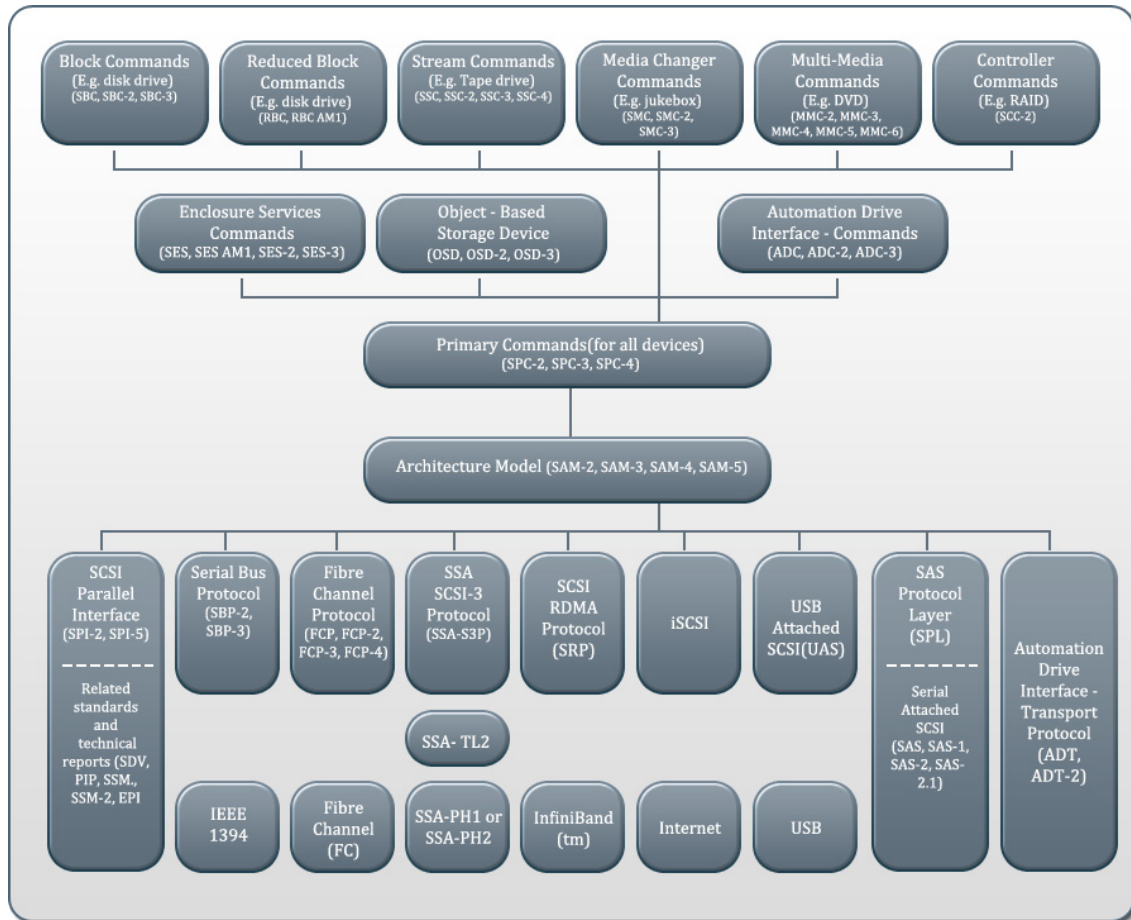


Figure 10: The T10 committee’s SCSI architecture model suite of standards.

Although external SATA (eSATA) matches the SuperSpeed USB bandwidth, the storage community seems more tilted towards USB. All of the leading vendors from the storage industry have worked aggressively in formalizing the UASP and related standards. There are two obvious reasons for this. First, all PCs and laptops have USB host ports, only a few support of eSATA. Second, adoption of UASP has eradicated the limitations introduced from the mass storage BOT standard.

At the time of this writing, a number of SuperSpeed USB storage devices have already hit the market and many more are expected. Most of these devices are still using the BOT protocol because operating systems lack native support of UASP. This trend will change once more operating systems become SuperSpeed and UASP aware.

MULTIMEDIA

Multimedia devices are the next obvious application area which can take benefit of SuperSpeed USB. Recently, USB-IF announced a new class specification for USB audio and video (USB AV). The use case of such a device can be found in a monitor with built-in speakers, microphone, and web camera.

USB DEVICE AUTHENTICATION

Current USB standards lack a device authentication mechanism. USB device authentication can add significant value for the end-user. This will help prevent unauthorized use of content on removable devices. Both the USB device and host will exchange public keys to authorize each other. The device will only be enumerated if the authorization was completed successfully.

CONCLUSION

Recent studies have shown that more than ten billion USB units are in use all over the world - with more than three billion new USB units added every year. USB is the connectivity standard that has shown tremendous growth in recent times and has surpassed competing technologies. With a wide adoption from key players and entire ecosystems, and continuous improvement in terms of speed, power management, and user experience USB has dominated in the wired connectivity space. There's no question that USB 3.0 will maintain this position into future as well.

For the latest product information, call us or visit: www.mentor.com

©2011 Mentor Graphics Corporation, all rights reserved. This document contains information that is proprietary to Mentor Graphics Corporation and may be duplicated in whole or in part by the original recipient for internal business purposes only, provided that this entire notice appears in all copies. In accepting this document, the recipient agrees to make every reasonable effort to prevent unauthorized use of this information. All trademarks mentioned in this document are the trademarks of their respective owners.

Corporate Headquarters Mentor Graphics Corporation 8005 SW Boeckman Road Wilsonville, OR 97070-7777 Phone: 503.685.7000 Fax: 503.685.1204	Silicon Valley Mentor Graphics Corporation 46871 Bayside Parkway Fremont, CA 94538 USA Phone: 510.354.7400 Fax: 510.354.7467	Europe Mentor Graphics Deutschland GmbH Arnulfstrasse 201 80634 Munich Germany Phone: +49.89.57096.0 Fax: +49.89.57096.400	Pacific Rim Mentor Graphics (Taiwan) Room 1001, 10F International Trade Building No. 333, Section 1, Keelung Road Taipei, Taiwan, ROC Phone: 886.2.87252000 Fax: 886.2.27576027	Japan Mentor Graphics Japan Co., Ltd. Gotenyama Garden 7-35, Kita-Shinagawa 4-chome Shinagawa-Ku, Tokyo 140-0001 Japan Phone: +81.3.5488.3033 Fax: +81.3.5488.3004
Sales and Product Information Phone: 800.547.3000 sales: info@mentor.com	North American Support Center Phone: 800.547.4303			

