# Chapter 4
# Architectural Overview

This chapter presents an overview of the Universal Serial Bus architecture and key concepts. USB is a cable bus that supports data exchange between a host computer and a wide range of simultaneously accessible peripherals. The attached peripherals share USB bandwidth through a host scheduled token based protocol. The bus allows peripherals to be attached, configured, used, and detached while the host and other peripherals are in operation. This is referred to as dynamic (or hot) attachment and removal.

Later chapters describe the various components of the USB in greater detail.

## 4.1 USB System Description

A USB system is described by three definitional areas:

- USB interconnect
- USB devices
- USB host

The USB interconnect is the manner in which USB devices are connected to and communicate with the host. This includes:

- Bus Topology: Connection model between USB devices and the host.
- Inter-layer Relationships: In terms of a capability stack, the USB tasks that 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.
- Scheduling the USB: USB provides a shared interconnect. Access to the interconnect is scheduled in order to support isochronous data transfers.

USB devices and the USB host are described in detail in subsequent sections.

### 4.1.1 Bus Topology

The Universal Serial Bus connects USB devices with the USB host. The USB physical interconnect is a tiered star topology. A hub is at the center of each star. Each wire segment is a point-to-point connection between the host and a hub or function, or a hub connected to another hub or function. Figure 4-1 illustrates the topology of the USB.

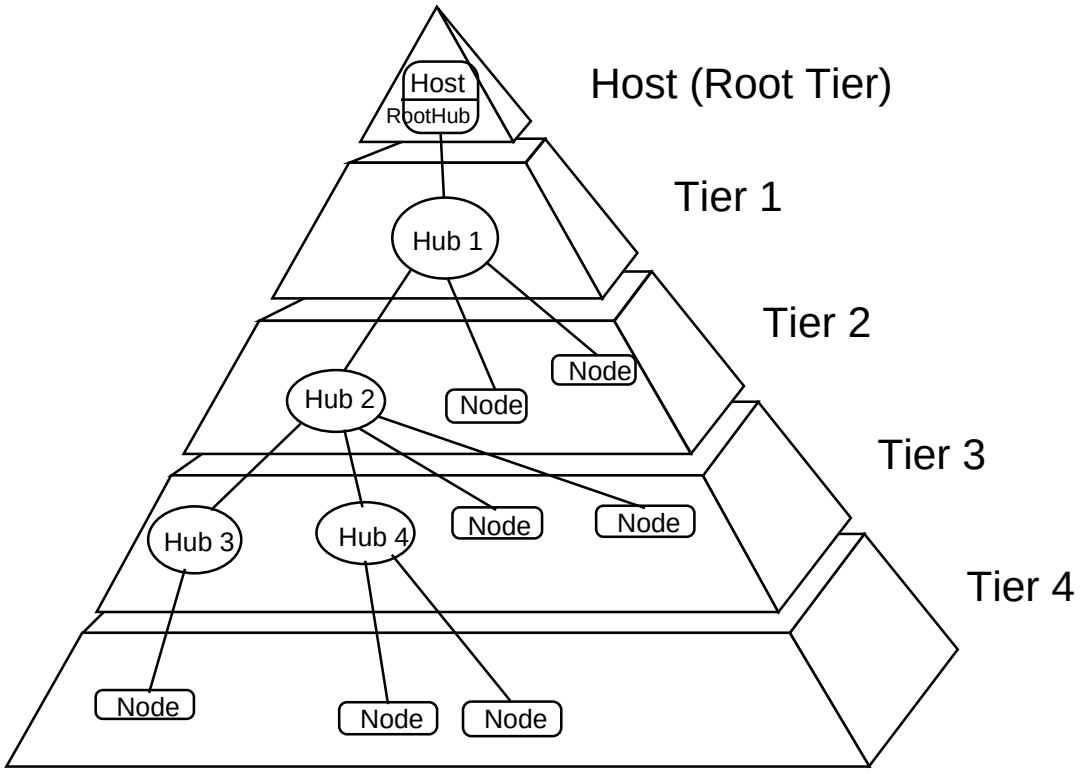


**Figure 4-1. Bus Topology**

#### 4.1.1.1 The USB Host

There is only one host on any USB system. The USB interface to the host computer system is referred to as the host controller. The host controller may be implemented in a combination of hardware, firmware, or software. A root hub is integrated within the host system to provide one or more attachment points.

Additional information concerning the host may be found in Section 4.9 and in Chapter 10, USB Host: Hardware and Software.

#### 4.1.1.2 USB Devices

USB devices are:

- Hubs, which provide additional attachment points to the USB
- Functions, which provide capabilities to the system; for example, an ISDN connection, a digital joystick, or speakers

USB devices present a standard USB interface in terms of their:

- Comprehension of the USB protocol
- Response to standard USB operations such as configuration and reset
- Standard capability descriptive information

Additional information concerning USB devices may be found in Section 4.8 and in Chapter 9, USB Devices.

## 4.2 Physical Interface

The physical interface of the USB is described in the electrical (Chapter 7) and mechanical (Chapter 6) specifications for the bus.

### 4.2.1 Electrical

USB transfers signal and power over a four wire cable, shown in Figure 4-2. The signaling occurs over two wires and point-to-point segments. The signals on each segment are differentially driven into a cable of 90 Ω intrinsic impedance. The differential receiver features input sensitivity of at least 200 mV and sufficient common mode rejection.

There are two modes of signaling. The USB full speed signaling bit rate is 12 Mbs. A limited capability low speed signaling mode is also defined at 1.5 Mbs. The low speed method relies on less EMI protection. Both modes can be simultaneously supported in the same USB system by mode switching between transfers in a device transparent manner. The low speed mode is defined to support a limited number of low bandwidth devices such as mice, since more general use would degrade the bus utilization.

The clock is transmitted encoded along with the differential data. The clock encoding scheme is NRZI with bit stuffing to ensure adequate transitions. A SYNC field precedes each packet to allow the receiver(s) to synchronize their bit recovery clocks.

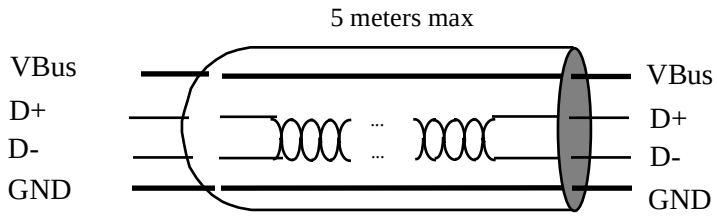


**Figure 4-2. USB Cable**

The cable also carries VBus and GND wires on each segment to deliver power to devices. VBus is nominally +5 V at the source. USB allows cable segments of variable lengths up to several meters by choosing the appropriate conductor gauge to match the specified IR drop and other attributes such as device power budget and cable flexibility. In order to provide guaranteed input voltage levels and proper termination impedance, biased terminations are used at each end of the cable. The terminations also permit the detection of attach and detach at each port and differentiate between full speed and low speed devices.

### 4.2.2 Mechanical

The mechanical specifications for cables and connectors are provided in Chapter 6. All devices have an upstream connection. Upstream and downstream connectors are not mechanically interchangeable, thus eliminating illegal loopback connections at hubs. The cable has four conductors: a twisted signal pair of standard gauge and a power pair in a range of permitted gauges. The connector is four position, with shielded housing, specified robustness, and ease of attach-detach characteristics.

## 4.3 Power

The specification covers two aspects of power:

- Power distribution over the USB deals with the issues of how USB devices consume power provided by the host over the USB.
- Power management deals with how USB software and devices fit into the host-based power management system.

### 4.3.1 Power Distribution

Each USB segment provides a limited amount of power over the cable. The host supplies power for use by USB devices that are directly connected. In addition, any USB device may have its own power supply. USB devices that rely totally on power from the cable are called bus-powered devices. In contrast, those that have an alternate source of power are called self-powered devices. A hub also supplies power for its connected USB devices. The architecture permits bus-powered hubs within certain constraints of topology that are discussed later in Chapter 11. Self-powered devices must implement prescribed power decoupling safety mechanisms. In Figure 4-4, the keyboard, pen, and mouse can all be bus-powered devices.

### 4.3.2 Power Management

A USB host has a power management system which is independent of the USB. USB system software interacts with the host's power management system to handle system power events such as SUSPEND or RESUME. Additionally, USB devices can carry USB-defined power management information which allow them to be power managed by system software or generic device drivers.

The power distribution and power management features of USB allow it to be designed into power sensitive systems such as battery based notebook computers.

## 4.4 Bus Protocol

All bus transactions involve the transmission of up to three packets. Each transaction begins when the host controller, on a scheduled basis, sends a USB packet describing the type and direction of transaction, the USB device address, and endpoint number. This packet is referred to as the Token Packet. The USB device that is addressed selects itself by decoding the appropriate address fields. In a given transaction, data is transferred either from the host to a device or from a device to the host. The direction of data transfer is specified in the token packet. The source of the transaction then sends a Data Packet or indicates it has no data to transfer. The destination in general responds with a Handshake Packet indicating whether the transfer was successful.

The USB data transfer model between a source or destination on the host and an endpoint on a device is referred to as a pipe. There are two types of pipes: stream and message. Stream data has no USB defined structure while message data does. Additionally, pipes have associations of data bandwidth, transfer service type, and endpoint characteristics like directionality and buffer sizes. Pipes come into existence when a USB device is configured. One message pipe, Control Pipe 0, always exists once a device is powered in order to provide access to the device's configuration, status, and control information.

The transaction schedule allows flow control for some stream mode pipes. At the hardware level, this prevents buffers from underrun or overrun situations by using a NACK handshake to throttle the data rate. The token for a NACK'ed transaction is reissued when bus time is available. The flow control mechanism permits the construction of flexible schedules that accommodate concurrent servicing of a heterogeneous mix of stream mode pipes. Thus, multiple stream mode pipes can be serviced at different intervals and with packets of different sizes.

## 4.5 Robustness

There are several attributes of the USB that contribute to its robustness:

- Signal integrity using differential drivers, receivers, and shielding
- CRC protection over control and data fields
- Detection of attach and detach and system-level configuration of resources
- Self-recovery in protocol, using time-outs for lost or broken packets
- Flow control for streaming data to ensure isochrony and hardware buffer management
- Data and control pipe constructs for ensuring independence from adverse interactions between functions

### 4.5.1 Error Detection

The core bit error rate of the USB medium is expected to be close to that of a backplane and any glitches will very likely be transient in nature. To provide protection against such transients, each of these packets includes error protection fields. When data integrity is required, such as with lossless data devices, an error recovery procedure may be invoked in hardware or software.

The protocol includes separate CRCs for control and data fields of each packet. A failed CRC is considered to indicate a corrupted packet. The CRC gives 100% coverage on single and double bit errors.

### 4.5.2 Error Handling

The protocol optionally allows for error handling in hardware or software. Hardware handling includes reporting and retry of failed transfers. The host controller will retry an error three times before informing the client software of the error. The client software can recover in an implementation specific way.

## 4.6 System Configuration

The USB supports USB devices attaching to and detaching from the USB at any point in time. Consequently, enumerating the USB is an on-going activity which must accommodate dynamic changes in the physical bus topology.

### 4.6.1 Attachment of USB Device

All USB devices attach to the USB via a port on specialized USB devices known as hubs. Hubs indicate the attachment or removal of a USB device in its per port status. The host queries the hub to determine the reason for the notification. The hub responds by identifying the port used to attach the USB device. The host enables the port and addresses the USB device with a control pipe using the USB Default Address. All USB devices are addressed using the USB Default Address when initially connected or after they have been reset.

The host determines if the newly attached USB device is a hub or a function and assigns a unique USB address to the USB device. The host establishes a control pipe for the USB device using the assigned USB address and endpoint number zero.

If the attached USB device is a hub and USB devices are attached to its ports, then the above procedure is followed for each of the attached USB devices.

If the attached USB device is a function, then attachment notifications will be dispatched by USB software to interested host software.

### 4.6.2 Removal of USB Device

When a USB device has been removed from one of its ports, the hub automatically disables the port and provides an indication of device removal to the host. Then the host removes knowledge of the USB device from any host data structures.

If the removed USB device is a hub, the removal process must be performed for all of the USB devices which were previously attached to the hub.

If the removed USB device is a function, removal notifications are sent to interested host software.

### 4.6.3 Bus Enumeration

Bus enumeration is the activity that identifies and addresses devices attached to a bus. For many buses, this is done at startup time and the information collected is static. Since the USB allows USB devices to attach to or detach from the USB at any time, bus enumeration for this bus is an on-going activity. Additionally, bus enumeration for the USB also includes the detection and processing of removals.

### 4.6.4 Inter-Layer Relationship

USB devices are logically divided into a USB device interface portion, a device portion, and a functional portion. The host is logically partitioned into the USB host interface portion, the aggregate system software portion (USB system software and host system software), and the device software portion.

Each of these portions is defined such that a particular USB task is the responsibility of only one portion. The USB host and USB device portions correspond as shown in Table 4-1.

**Table 4-1. Correlation Between Host and Device Layers**

| USB Host Portion | USB Device Portion |
|---|---|
| Device Software | Function |
| System Software | Device |
| USB Interface | USB Interface |

## 4.7 Data Flow Types

The USB supports functional data and control exchange between the USB host and a USB device as a set of either uni- or bi-directional fashions. USB data transfers take place between host software and a particular endpoint on a USB device. A given USB device may support multiple data transfer endpoints. The USB host treats communications with any endpoint of a USB device independently from any other endpoint. Such associations between the host software and a USB device endpoint are called pipes. As an example, a given USB device could have an endpoint which would support a pipe for transporting data *to* the USB device and another endpoint which would support a pipe for transporting data *from* the USB device.

The USB architecture comprehends four basic types of data transfers:

- Control transfers that are used to configure a device at attach time and can be used for other device specific purposes
- Bulk data transfers which are generated or consumed in relatively large and bursty quantities and has wide dynamic latitude in transmission constraints
- Interrupt data transfers such as characters or coordinates with human perceptible echo or feedback response characteristics
- Isochronous or streaming real time data transfers which occupy a prenegotiated amount of USB bandwidth with a prenegotiated delivery latency

Any given pipe supports exactly one of the types of transfers described above. The USB Data Flow model is described in more detail in Chapter 5.

### 4.7.1 Control Transfers

Control data is used by USB software to configure devices when they are first attached. Other driver software can choose to used control transfers in implementation specific ways. Data delivery is lossless.

### 4.7.2 Bulk Transfers

Bulk data typically consists of larger amounts of data such as that used for printers or scanners. Bulk data is sequential. Reliable exchange of data is ensured at the hardware level by using error detection in hardware and, optionally, invoking a limited hardware retry. Also, the bandwidth taken up by bulk data can be whatever is available and not being used for other transfer types.

### 4.7.3 Interrupt Transfers

A small, spontaneous data transfer from a device is referred to as interrupt data. Such data may be presented for transfer by a device at any time and is delivered by the USB at a rate no slower than as is specified by the device.

Interrupt data typically consists of event notification, characters, or coordinates that are organized as one or more bytes. An example of interrupt data is the coordinates from a pointing device. Although an explicit timing rate is not required, interactive data may have response time bounds which the USB must support.

### 4.7.4 Isochronous Transfers

Isochronous data is continuous and real-time in creation, delivery, and consumption. Timing related information is implied by the steady rate at which isochronous data is received and transferred. Isochronous data must be delivered at the rate received to maintain its timing. In addition to delivery rate, isochronous data may also be sensitive to delivery delays. For isochronous pipes, the bandwidth required is typically based upon the sampling characteristics of the associated function. The latency required is related to the buffering available at each endpoint.

A typical example of isochronous data is voice. If the delivery rate of these data streams is not maintained, glitches in the data stream will occur due to buffer or frame underruns or overruns. Even if data is delivered at the appropriate rate, delivery delays may degrade applications requiring real-time turn around, such as telephony based audio conferencing.

The timely delivery of isochronous data is ensured at the expense of potential transient losses in the data stream. In other words, any error in electrical transmission is not corrected by hardware mechanisms such as retries. In practice, the core bit error rate of the USB is expected to be small enough not to be an issue. USB isochronous data streams are allocated a dedicated portion of USB bandwidth to ensure that data can be delivered at the desired rate. The USB is also designed for minimal delay of isochronous data transfers.

### 4.7.5 Allocating USB Bandwidth

USB bandwidth is allocated among pipes. The USB allocates bandwidth for some pipes when a pipe is established. USB devices are required to provide some buffering of data. It is assumed that USB devices requiring more bandwidth are capable of providing larger sized buffers. The goal for the USB architecture is to ensure that buffering induced hardware delay is bounded to within a few milliseconds.

USB' bandwidth capacity can be allocated among many different data streams. This allows a wide range of devices to be attached to the USB. For example, telephony devices ranging from 1B+D all the way up to T1 capacity can be accommodated. Further, different device bit rates, with a wide dynamic range, can be concurrently supported.

USB bandwidth allocation is blocking; i.e., if allocating an additional pipe would disturb preexisting bandwidth or latency allocations, further pipe allocations are denied or blocked. When a pipe is closed, the allocated bandwidth is freed up and may be reallocated to another pipe.

The USB specification defines the rules for how each transfer type is allowed access to the bus.

## 4.8 USB Devices

USB devices are divided into device classes such as hub, locator, or text device. The hub device class indicates a specially designated USB device which provides additional USB attachment points (refer to Chapter 11). USB devices are required to carry information for self-identification and generic configuration. They are also required at all times to display behavior consistent with defined USB device states.

### 4.8.1 Device Characterizations

All USB devices are accessed by a unique USB address. Each USB device additionally supports one or more endpoints with which the host may communicate. All USB devices must support a specially designated Endpoint 0 to which the USB device's USB control pipe will be attached.

Associated with Endpoint 0 is the information required to completely describe the USB device. This information falls into the following categories:

- Standard. This is information whose definition is common to all USB devices and includes items such as vendor identification, device class, and power management. Device, configuration, interface, and endpoint descriptions carry configuration related information about the device. Detailed information about these descriptors can be found in Chapter 9.
- Class. The definition of this information varies depending on the device class of the USB device.
- USB Vendor. The vendor of the USB device is free to put any information desired here. The format, however, is not determined by this specification.

Additionally, each USB device carries USB control and status information. All USB devices support a common access method via their USB control pipe.

### 4.8.2 Device Descriptions

Two major divisions of device classes exist: hubs and functions. Only hubs have the ability to provide additional USB attachment points. Functions provide additional capabilities to the host.

#### 4.8.2.1 Hubs

Hubs are a key element in the plug-and-play architecture of USB. Figure 4-3 shows a typical hub. Hubs serve to simplify USB connectivity from the user's perspective and provide robustness at low cost and complexity.

Hubs are wiring concentrators and enable the multiple attachment characteristics of USB. Attachment points are referred to as ports. Each hub converts a single attachment point into multiple attachment points. The architecture supports concatenation of multiple hubs.

The upstream port of a hub connects the hub towards the host. Each of the other downstream ports of a hub allows connection to another hub or function. Hubs can detect attach and detach at each downstream port and enable the distribution of power to downstream devices. Each downstream port can be individually enabled and configured as either full or low speed. The hub isolates low speed ports from full speed signaling.

A hub consists of two portions: the Hub Controller and the Hub Repeater. The repeater is a protocol controlled switch between the upstream port and downstream ports. It also has hardware support for reset and suspend/resume signaling. The controller provides the interface registers to allow communication to/from the host. Hub specific status and control commands permit the host to configure a hub and to monitor and control its ports.

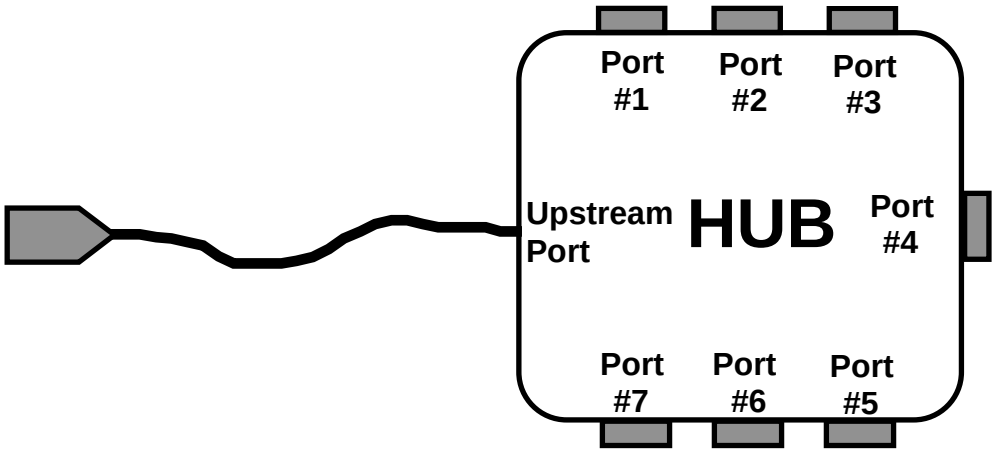


**Figure 4-3. A Typical Hub**

Figure 4-4 illustrates how hubs provide connectivity in a desktop computer environment.

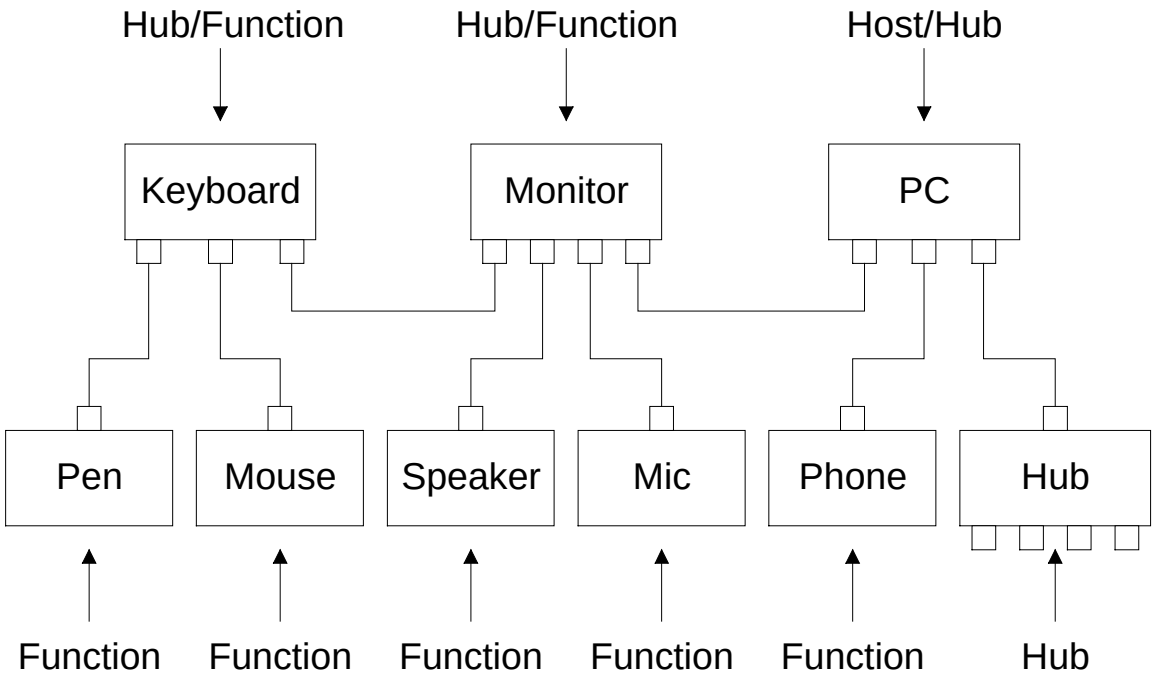


**Figure 4-4. Hubs in a Desktop Computer Environment**

### 4.8.2.2 Functions

A function is a USB device that is able to transmit or receive data or control information over the bus. A function is typically implemented as a separate peripheral device with a cable that plugs into a port on a hub. However, a physical package may implement multiple functions and an embedded hub with a single USB cable. This is known as a compound device. A compound device appears to the host as a hub with one or more permanently attached USB devices.

Each function contains configuration information that describes its capabilities and resource requirements. Before a function can be used, it must be configured by the host. This configuration includes allocating USB bandwidth and selecting function specific configuration options.

Examples of functions are:

- A locator device such as a mouse, tablet, or light pen
- An input device such as a keyboard
- An output device such as a printer
- A telephony adapter such as ISDN

## 4.9 USB Host: Hardware and Software

The USB Host interacts with USB devices through the host controller. The host is responsible for the following:

- Detecting the attachment and removal of USB devices
- Managing control flow between the host and USB devices
- Managing data flow between the host and USB devices
- Collecting status and activity statistics
- Providing a limited amount of power to attached USB devices

USB system software on the host manages interactions between USB devices and host-based device software. There are five areas of interactions between USB system software and device software, they are:

- Device enumeration and configuration
- Isochronous data transfers
- Asynchronous data transfers
- Power management
- Device and bus management information

Whenever possible, USB software uses existing host system interfaces to manage the above interactions. For example, if a host system uses Advanced Power Management (APM) for power management, USB system software connects to the APM message broadcast facility to intercept suspend and resume notifications.

## 4.10 Architectural Extensions

The USB architecture comprehends extensibility at the interface between the Host Controller Driver and USB Driver. Implementations with multiple host controllers, and associated Host Controller Drivers, are possible.