

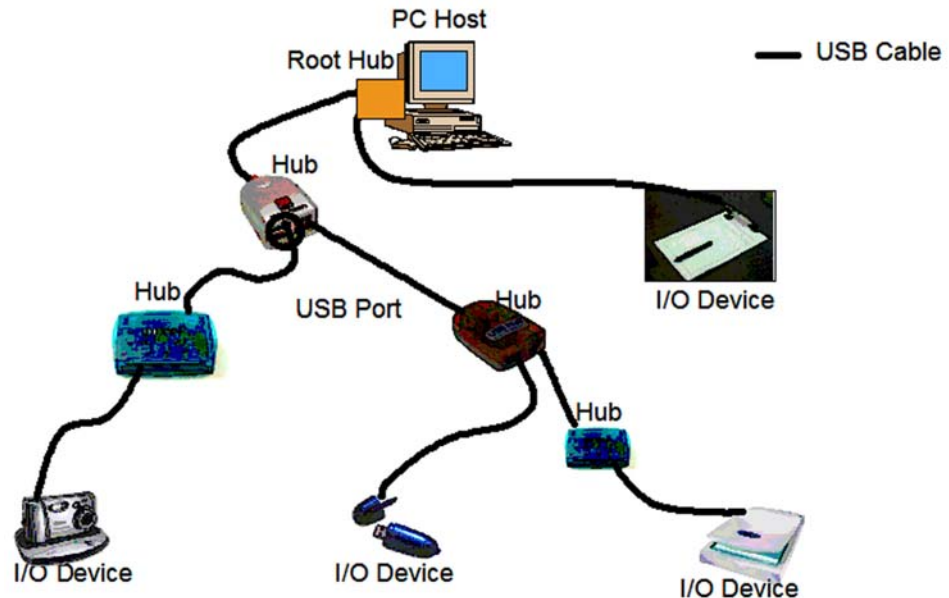
### 3 USB Overview

#### 3.1 USB Terminology

The USB specification introduced new terms that are used throughout the USB literature. This section introduces those terms and presents an overview.

A typical configuration has a single PC host with multiple devices interconnected by USB cables. The PC host has an embedded hub, also called the root hub, which typically contains two or more USB ports.

Host  
Root Hub



**Figure 1: Standard USB designation**

Device configuration range from simple to complex:

- Hub
  - **Hub:** If a device contains only additional downstream USB ports, then it is called simply a hub.
- I/O Device
  - **I/O device:** An I/O device adds capability to the host. It has a single upstream connection and interacts with the real world to create or consume data on behalf of the PC host.
- Compound Device
  - **Compound device:** If a device includes both I/O and hub functionality, it is called a compound device. A keyboard that includes additional USB downstream ports is such an example.
- Composite Device
  - **Composite device:** If a single device implements two or more sets of device functions, it is called a composite device. For example an eyecam camera with a camera and dual audio channels and a microphone is a composite device

As far the PC host is concerned, devices are the important feature, and as many as 126 devices can be interconnected using external hubs up to five levels deep (in Figure 1 the hub level is three levels deep).

Speed

USB 2.0 supports three device speeds. The USB specification version 1.1 defined only two device speeds, such as **low speed** at 1.5 Mbps and **full speed** at 12 Mbps. The **high speed** at 480 Mbps has been added in USB specification 2.0. Low speed devices are the cheapest to manufacture and are adequate for supporting low data rate devices such as mice, keyboards, and a wealth of other equipment designed to interact with people. The introduction of high speed data rate enables high bandwidth devices such as full colour page scanners, printers and mass storage devices.

### 3.2 PC Host

A typical configuration has a single PC host. The PC host runs an USB aware operating system software that supports two distinct functions: initialization and monitoring of all USB devices.

The USB initialization software is active all the time and not just during the PC host powered-on. Because the initialization software is always active, USB devices can be added and removed at any time, also known as Plug and Play. Once a device is added to the host, the device is enumerated by the USB initialization software and assigned a unique identifier that is used at run time.

Figure 2 shows how the USB host software is layered (layering supports many different software solutions).

Client Software

On the top **Client Software** is being executed to achieve the desired USB device functionality. The application software is written in user mode and therefore does not harm the operating system when errors occur. Class libraries gain access in user mode to class driver functions. A class is a grouping of devices with similar characteristics that can be controlled by a generic class device driver. Examples of classes include mass-storage devices, communication devices, audio devices, human-interface devices (HID) and some more that can be found at [www.usb.org](http://www.usb.org). If a device does not fit into one or more of these predefined classes, then a specific device driver has to be written for the device. Device drivers are executed in the kernel mode and must therefore be validated and tested to be error free. Next to the Client Software layer follows the **USB System Software** layer. Enumerating processes and USB monitoring is the major task of this layer. It is also responsible for recognising removal and attachments of devices. The deepest layer is the **USB Host Controller**. It is the hardware and software that allows USB devices to be attached to the host.

System Software

Host Controller

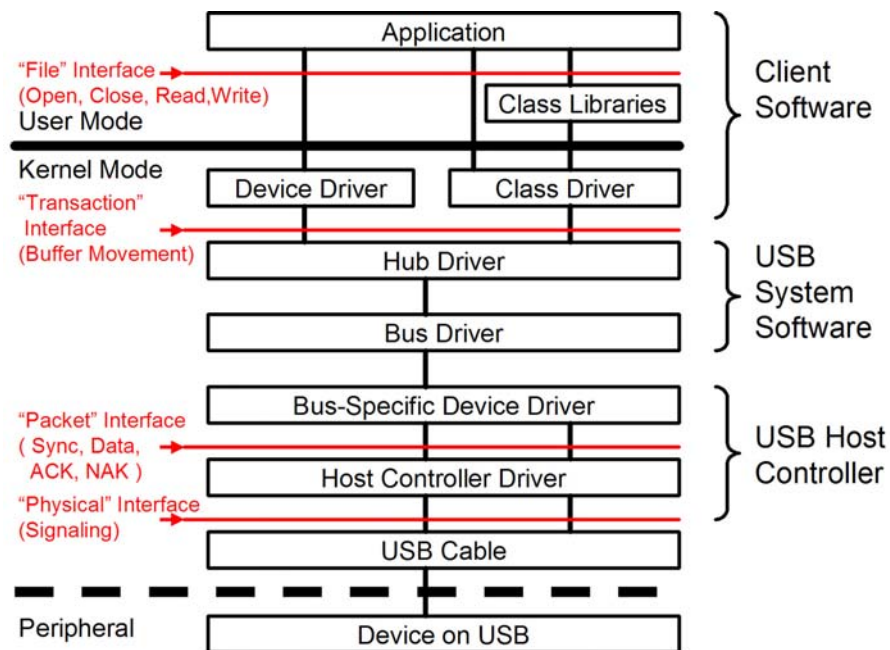


Figure 2: PC host software for USB is defined in layers

### 3.3 USB Cable

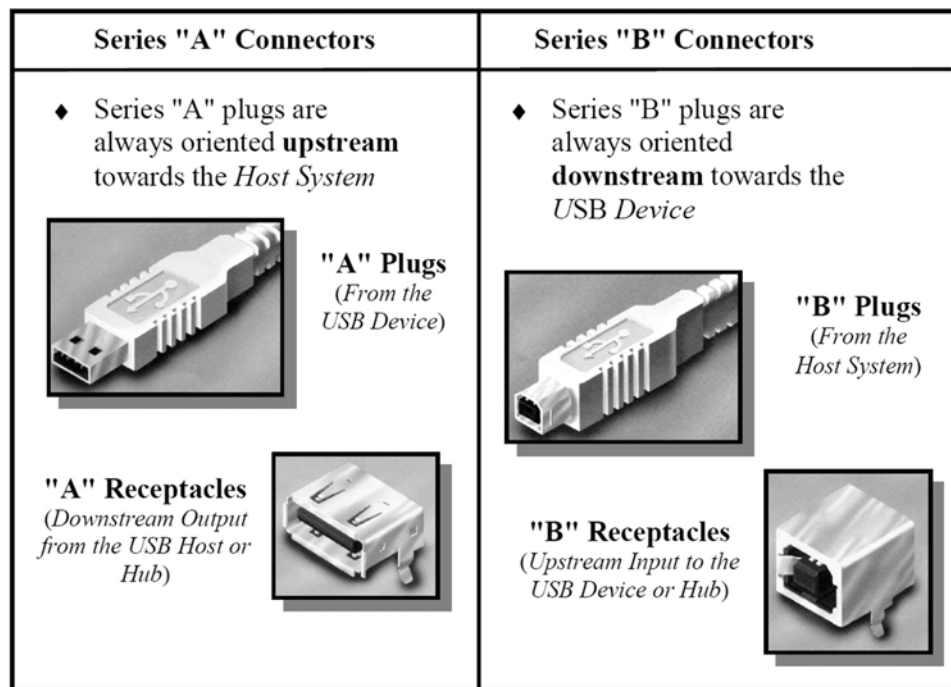
A USB Cable transports both power supply and data signals.

The power supplied by the USB cable is an important benefit of the USB specification. A simpler I/O device can rely on the USB cable for all its power needs and will not require the traditional “black brick” plugged into the wall. The power resource is carefully managed by the USB, with the hub device playing the major role. A hub or an I/O device can be self-powered or bus-powered.

- **Self-powered** is the traditional approach in which the hub or I/O device has an additional power cable attached to it.
- A **bus-powered** device relies solely on the USB cable for its power needs and is often less expensive.

Connectors

The USB cable connectors were specifically designed with the power pins longer than the signal pins so that power would always be applied before signals. Two different connector types were defined, to ensure that illegal configurations could not be made. An “**A**”-type connector defines the downstream end of the cable, and a “**B**”-type connector defines the upstream end (Figure 3).



**Figure 3: USB cable connector types [27]**

Maximum Length

The maximum cable length is 5 meters between a hub and a device. With up to five levels of hubs we reach a length of 30 meters from the PC host to the device.

### 3.4 Hub Device

The hub has two major roles: power management and signal distribution.

External Hub

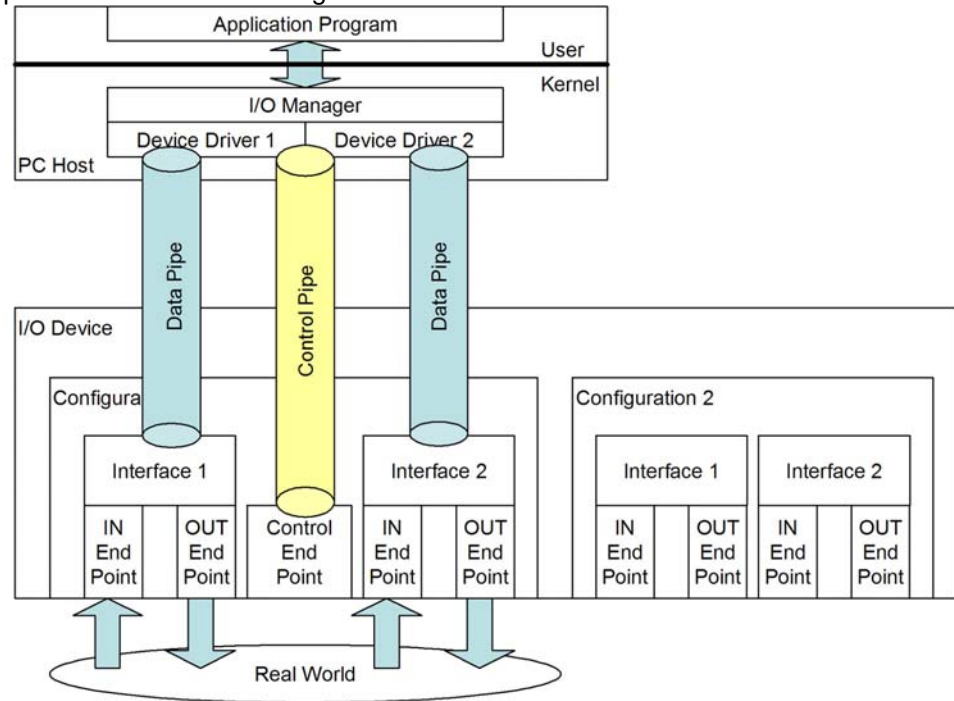
An external hub has one upstream connection and multiple downstream connections. The USB specification does not limit the number of downstream connections. The most popular hub size has four downstream ports.

A hub can be self-powered or bus-powered. The self-powered hub can provide up to 500mA to each of its downstream ports while a bus-powered has a maximum of 500mA to all the ports.

### 3.5 I/O Device

A PC host creates data for or consumes data from the real world as shown in Figure 1. A scanner is a good example of a data-creating I/O device and a printer of a data consuming I/O device.

Logical View



**Figure 4: Logical view of an I/O device**

The software (or logical) view of the USB connection is shown in Figure 4. This diagram is best explained bottom up.

Endpoint

The term **endpoint** is used to describe a point where data enters or leaves a USB system. An IN endpoint is a data creator, and an OUT endpoint is a data consumer. Note that the data direction is relative to the PC host – if we remember that the PC host is the “master” controlling all data movements, then the data direction is easy to understand.

Interface

A typical real-world connection may need multiple IN and/or OUT endpoints to implement a reliable data-delivery scheme. This collection of endpoints is called an **interface** and is directly related to a real-world connection. The operating system will have a software driver that corresponds to each interface. The operating system uses the term pipe to describe the logical connection between a software driver on the PC host and the interface on the I/O device. There is always a one-to-one mapping between software drivers and interfaces.

Pipes

Configuration

A collection of interfaces is called a **configuration**, and only one configuration can be active at a time. A configuration defines the attributes and features of a specific model. Using configuration allows a single USB connection to serve many different roles, and the modularity of this system solution saves in development time and support costs.

### 3.6 Information Flow

Transfer Types

USB defines four methods of transferring data, as summarized in Table 1. The methods differ in the amount of data that can be moved in a single transaction in whether any particular periodicity or latency can be guaranteed, and in whether errors will be automatically corrected. Each method corresponds to a particular type of endpoint.

Transfer Type	Description	Lossless?	Latency Guarantee?
Control Transfer	Used to send and receive structured information of control nature	Yes	Best effort
Bulk Transfer	Used to send or receive blocks of unstructured data	Yes	No
Interrupt Transfer	Like a bulk pipe but includes maximum latency	Yes	Polled at guaranteed minimum rate
Isochronous Transfer	Used to send or receive blocks of unstructured data with guaranteed periodicity	No	Read or written at regular intervals

**Table 1: USB data transfer types**

Endpoints have several attributes in addition to their type. One endpoint attribute is the maximum amount of data that the endpoint can provide or consume in a single transaction. Table 2 indicates the maximum values for each endpoint type for each speed of device. In general, any single transfer can involve less than the maximum amount of data that the endpoint is capable of handling.

Transfer Type	High Speed	Full Speed	Low Speed
Control	64	8,16,32 or 64	8
Bulk	< 512	8,16,32 or 64	not allowed
Interrupt	< 1024	< 64	< 8
Isochronous	< 3072	< 1023	not allowed

**Table 2: Allowable endpoint maximum packet sizes in bytes**

### 3.7 Descriptors

USB devices maintain on-board data structures known as descriptors to allow for self-identification to host software. Table 3 lists the different descriptor types.

Descriptor Type	Description
Device	Describes an entire device
Configuration	Describes one of the configurations of a device
Interface	Describes one of the interfaces that is part of configuration
Endpoint	Describes one of the endpoints belonging to an interface
String	Contains a human readable Unicode string describing the device, a configuration, an interface, or an endpoint.

**Table 3: Descriptor types**