Universal Serial Bus

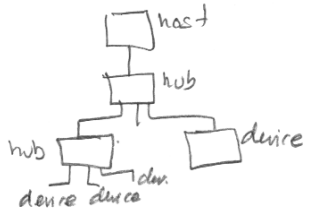
Introduction

Although the USB interface is a peripheral interface bus rather than a communication interface, it is often used to interface to communication devices such as modems. The USB bus is often used to replace peripheral interfaces that formerly used serial interfaces such as printers.

This lecture provides a brief overview of the USB interface and how it compares to the much simpler RS-232 serial interface.

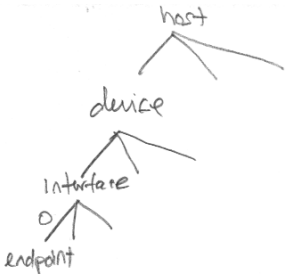
Architecture

USB is asymmetrical. One end of the link is the host and the other is the device. The host supplies power and controls the bus. Multiple USB devices can be connected to one host using a tree topology with a host at the root, devices as the leaves and USB hubs to connect them.



Each device is programmed with a device descriptor that identifies the type of device using two 16-bit values: a vendor ID and a device ID. The descriptor also specifies the type of device, for example an audio output device. This allows device drivers to support devices made by different vendors. Custom devices are also possible.

Each device can have multiple interfaces (e.g. audio and video). Each interface can have several endpoints, each either input or output. Endpoint 0 is used for controlling the interface. Endpoint 0 also contains a block of data that the host reads to discover the type of interface and its capabilities.



Each endpoint can be designed for different types of data:

- constant-rate ("isochronous") endpoints for time-sensitive data such as audio
- non-constant rate data ("bulk") for time-insensitive data such as data from a disk drive
- polled data (confusingly called "interrupt" endpoints) such as keystrokes from a keyboard
- configuration endpoints for controlling the interface

Hardware Interface

Four different data rates are supported:

USB 1 defined "Low Bandwidth" 1.5 Mb/s and "Full Speed" at 12 Mb/s

USB 2.0 defined "High Speed" at 480 Mb/s.

USB 3.0 defined "Super High Speed" at 5 Gb/s.

Throughputs are considerably lower, particularly at high data rates and short transfers.

The USB cable uses two pairs, one pair for 5V power supply for the peripheral and one pair for data. Signaling is differential, half-duplex, differential NRZ (NRZI). The cable is terminated in 90 ohms.

Bit stuffing is used to ensure transitions for timing: a 0 inserted after every six 1 bits.

Frame ends are indicate by violating the differential signaling by bringing both lines low. The start of a frame is indicated by a special sync sequence.

End of frame (and reset) by using same polarity on both wires of a pair.

The USB 3.0 connector has an additional two pairs to allow full-duplex operation. The 5 Gb/s signalling uses scrambling, 8B/10B line coding and differential signalling.

Comparison with RS-232

The USB interface was designed to address a number of shortcomings of previous peripheral buses such as RS-232 serial interfaces. Some of the advantages of USB include:

Speed even the slowest USB interfaces are 10 times faster than RS-232 serial interfaces.

Self-Configuration a USB peripheral identifies itself and its capabilities. The user does not have to configure the computer (other than making sure the appropriate driver software is installed). On the other hand, serial ports must be configured by the user for the correct baud rate, parity, etc.

Bus Power the host can supply power to peripherals (up to 2.5W)

Hot Plug the USB interface supports connecting and disconnecting peripherals without having to power off or reset the host. This is called "hot plugging" and requires both hardware and software support.

Standard Device Classes the USB standard defines the behaviour of specific types peripherals (keyboard, video camera, mass storage, etc). Manufacturers that build devices that conform to these standards do not need to supply driver software for their devices.

Implementing USB Interfaces

Simple USB interfaces can be implemented using microcontrollers with USB interface peripherals built-in. Such microcontrollers are currently available for about the same cost as an RS-232 UART (about $1).

For simple applications application-specific software can be added to the microcontroller. For more complex applications the microcontroller can be treated as an interface peripheral much as a UART would be used to implement a serial interface.

USB microcontrollers pre-programmed to act as USB-to-serial interface peripherals are available and can be used to implement simple interfaces without the need for special software.