The USB Bus

usb: general ideas

USB is a high speed serial bus

- Four wires:: D+, D-, 5V, GND
  - Differential signalling, NRZI, with bit stuffing
  - Current limit: 500mA
  - OTG connectors have a fifth pin: "ID"
- Packet-based communication
- Point-to-point physical connection
  - One of the parties is the controlling one

    Called "usb host" or "usb master"
  - The other party is only replying to queries

    Called "usb device" or "usb slave"
- You can use hubs to extend the bus
  - A hub device can be externally powered or not
  - Each bus can enumerate no more than 127 devices
- The specs include a special protocol for current management

USB-1, USB-2

USB-1 (v1.0: 1996, v1.1: 1998) supports the following speeds:

- Slow speed
  - 1.5Mbit/s, max 1023 bytes per packet
- Full speed
  - 12Mbit/s, max 1023 bytes per packet

Every millisecond, the host must send a SOF packet

USB-2 (v2.0: 2000) supports the following speeds:

- Slow speed
- Full speed
- High speed
  - 480Mbit/s, 8kB per packet

The host sends a SOF packet every 125 usecs (8kHz)

Beware: there are USB-2.0 devices that only support full-speed

USB3

USB-3 (v3.0: 2010) adds "Super Speed" (5Gbit/s)

- It uses additional data pairs to achieve higher speed
- 900mA current limit (was 500mA)
- Backward compatible with USB2 and USB1
- Not really relevant to the microcontroller world

USB-3.1 (2013), USB-3.2 (2017)

- ....

The USB hardware (ignoring v3 and later)

Lines should be equal length with 90 Ohm diff. impedance

- Each wire should be terminated to a 45 Ohm transceiver

The 5V supply is just convenience

- It is not related to signalling (which works at 3.3V or 0.8V)
- It is specified with 5% tolerance (4.75..5.25)
- Most chargers run a 5.3V or even a little more

Devices report presence with a 1.5k pull-up resistor (to 3.3V)

- The host side pulls down with 15k.

Cable length is limited to 5 meters

The chain is limited to 7 levels all-included (i.e. 5 hubs)

The USB protocol basics

The protocol is completely master-driven

- The slave must feature a good-enough clock
- "Internal RC" oscillators are not enough

Both master and slave devices are implemented in hardware

- Software can't deal with speed and determinism of the bus

The master port is depicted as a "Root Hub"

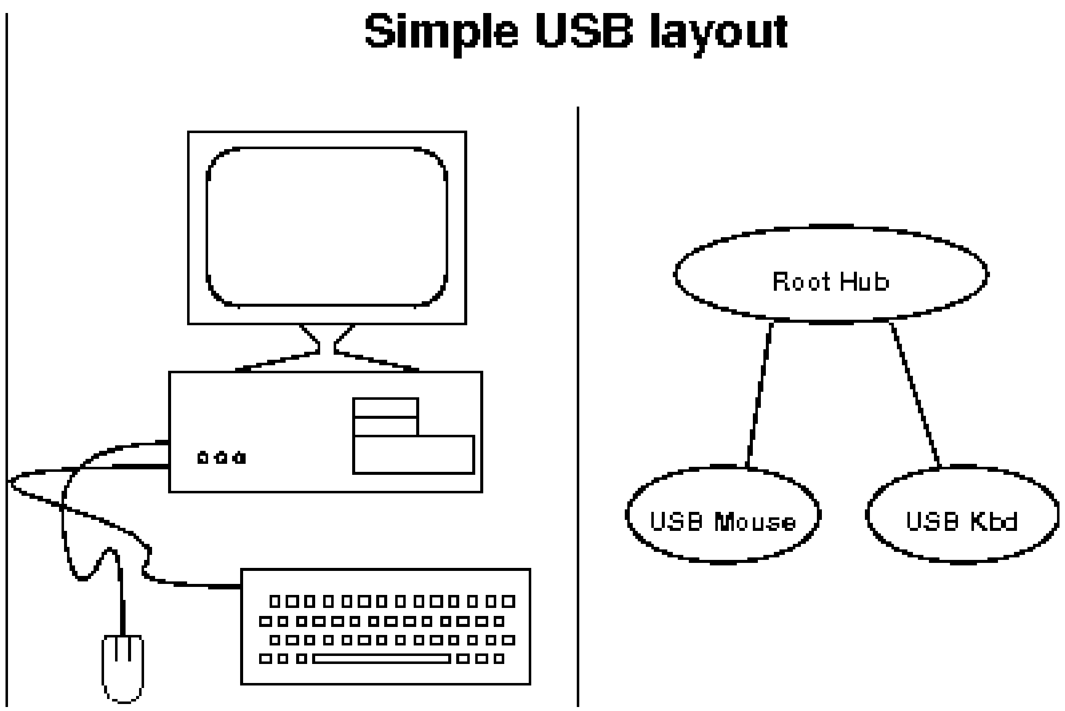
- A PC usually has several root hubs
- Root hubs can be single-port or multi-port

Bandwidth is effectively shared between devices

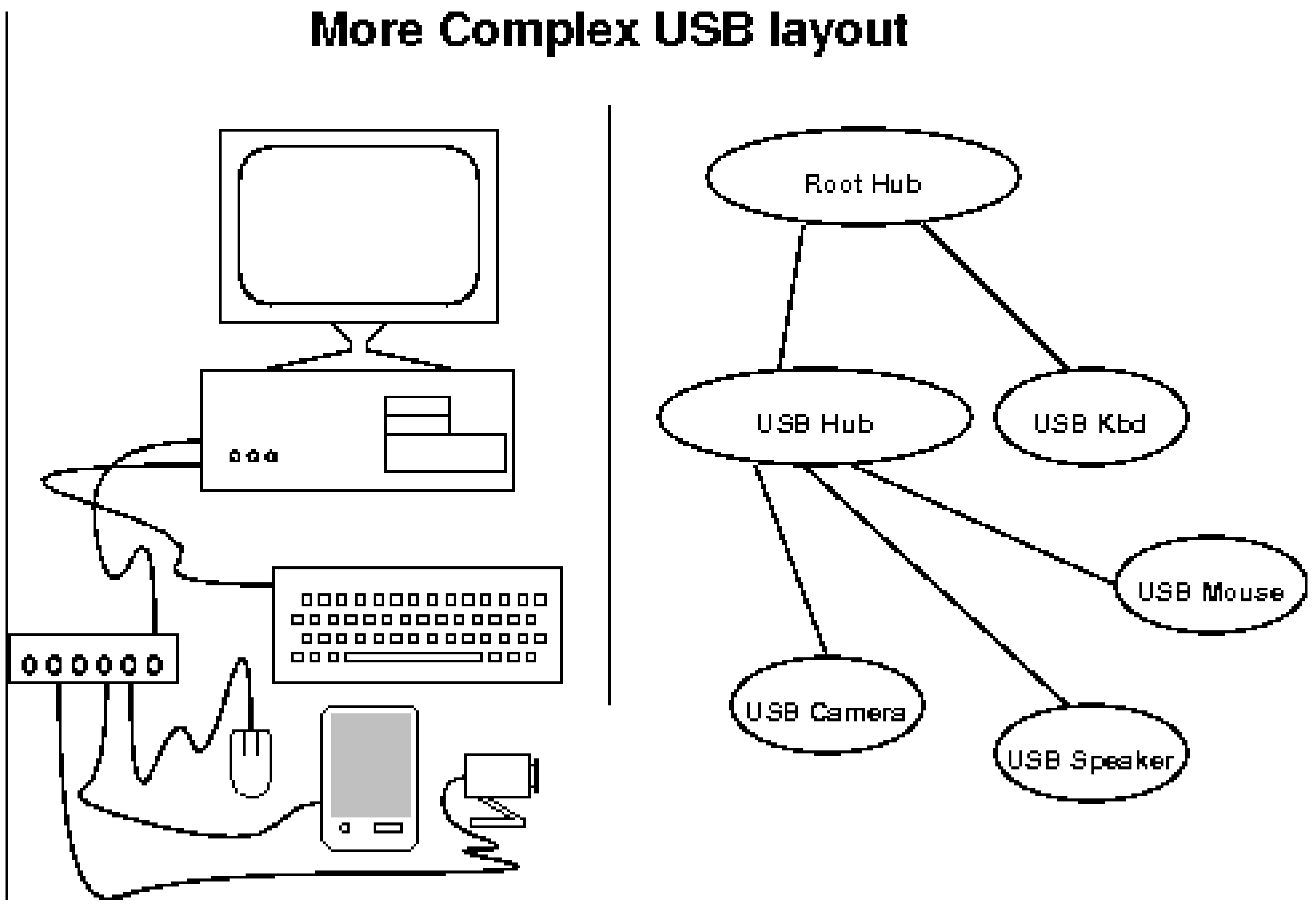
When a device appears, it must be enumerated

- Hubs are enumerated too (including the root hub)

Example of USB buses



Simple USB layout



More Complex USB layout

The USB protocol: endpoints and more

Each device features a vendor and device ID, plus class

- Vendor and device are 16 bits wide
  - This is artificial scarcity, designed by a greedy consortium
  - USB identifiers are very expensive, and must be renewed

Devices can feature more than one "Interface"

- Packets are addressed to a specific interface
- Each interface is like a separate device

Each interface features several "endpoints"

- Packets are addressed to a specific endpoint
- USB-1 was talking about "pipes", a now-forgotten word
  - Endpoints were the end points of each pipe
- An endpoint can be either input or output (as seen from the host)
- The standard defines 3 types of endpoints are defined (and "control")

Endpoint types

Control

- EP0 always exists, and it is the Control Endpoint
  - It is bidirectional, with a request/response protocol

Bulk

- Data channels without timing constraints
- They are a one-way data stream
- Usually, you run them in pairs (input and output)

Interrupt

- Input channel, much alike an interrupt event channel
- Actually, it's always the host who polls the device

Isochronous

- Sustained data flow, with guaranteed bandwidth
- Typically used to deliver audio or video streams

For all types, transmission is packet- and frame-oriented

- The receiver is aware of the size of each frame
- A packet can span multiple frames (e.g.: more than 64 bytes).

Enumeration

When a device appears, it must be enumerated.

- It initially responds to address 0
- The host queries device information
- It then assigns an address to the device
  - All of this happens on endpoint 0

With this new address, everything starts over

- The host queries device information, again
- It collects identifiers and "strings"
- Eventually, the driver may take over and use the other endpoints

Enumeration Example

Data collected with "usbmon" Linux, target is "hello.bin"

```
S Ci:000:00 s 80 06 0100 0000 0040 64 <
C Ci:000:00 0 64 = 12010001 00000040 c41060ea [...]
[...]
S Co:000:00 s 00 05 0024 0000 0000 0
C Co:000:00 0 0
S Ci:036:00 s 80 06 0100 0000 0012 18 <
C Ci:036:00 0 18 = 12010001 00000040 c41060ea 00010302 0101
[...]
S Ci:036:00 s 80 06 0300 0000 00ff 255 <
C Ci:036:00 0 4 = 04030904
S Ci:036:00 s 80 06 0302 0409 00ff 255 <
C Ci:036:00 0 12 = 0c036600 73006d00 6f007300
S Ci:036:00 s 80 06 0303 0409 00ff 255 <
C Ci:036:00 0 10 = 0a037200 75006200 6900
[...]
```

Explanation of the above example

USBMON is not a sniffer

- It cannot look at the wire, only at higher levels
- USB frames and timing require specific hardware
- And USB is not a shared channel like Ethernet

"S" means "Submit" and "C" means "Complete

- The software stack is a state machine
- Every submit must be followed by a complete, possible delayed

"Co" is "control out", we also have "Bo", "Ii" etc

- "000:00" is endpoint 0 of device 0 (not enumerated yet)
- "036:00" is endpoint 0 of device 36 (after enumeration)

"64 <" at the end of a submit line is the input buffer size

- The reply can be no bigger than that

Writing a USB stack

The USB protocol stack can be laid out in 2 or three levels

- Hardware management (which may include low-level protocol)
- Optionally, common protocol procedures
- The actual device code (usb-serial, network, storage)

Then, higher levels (UDP/IP, FATFS, whatever) will be generic

- The USB device driver will offer a non-usb API

Most USB implementations are state machines with callbacks

- This can happen based on interrupts
- Or you can just poll the status bits