“The Spirit of USB”

Part 1, introduction
by Jerker Björkqvist, Åbo Akademi
Part 2, technical
by Robert Gyllenberg, Åbo Akademi

jerker.bjorkqvist@abo.fi
robert.gyllenberg@abo.fi

Contents

Design goals for USB

- Easy to use, no configuration and setup details
- Fast, interface should not be a bottleneck
- Reliable, errors are rare, automatic retries
- Versatile, many type of peripherals can use the interface
- Inexpensive
- Power conserving, for portable computers and devices
- Supported by Windows and other OS

USB Basic information

- 1.5 Mbit/s, 12 Mbit/s or 480 Mbit/s
- At maximum 127 devices
- Cable length max 5 m
- Up to 5 hubs on a line
- Hot-pluggable
- 5V Inline power (max 500 mA)

Benefits for the user

- Easy to use
  - One interface for many devices
  - Automatic configuration
  - Easy to connect
  - Easy cables
  - Hot-pluggable
  - No user settings
  - Frees hardware resource for other devices
  - No power supply required (depending on the device)

Benefits for the user II

- Speed – bus speeds
  - USB 2.0
    - High speed 480Mbits/s
  - USB 1.1
    - Full speed 12 Mbits/s
    - Low speed 1.5 Mbits/s
- Reliability
  - Hardware specifications
    - Quiet interface
    - Automatic retransmission

Benefits for the user III

- Low cost
  - Components and cables are inexpensive
  - Expected cost is the same or less as with corresponding older interface
- Low power consumption
  - Power down USB devices when not in use

Benefits for developers

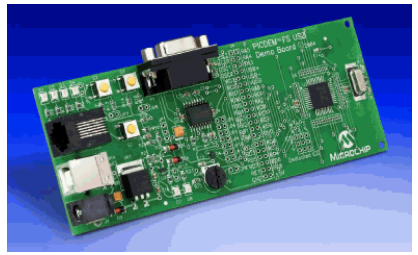
- Developers – types of
  - Device developers: Hardware, selecting of components and circuits in devices
  - Host software: Software for PC
- Standardized interface
  - No need for specifying cable characteristics or doing error checking in software
- Versatility
  - Transfer types and speed make the interface feasible for many types of peripherals
    - Large and small data amount
    - Guaranteed bandwidth, maximum time between transfers

Benefits for developers II

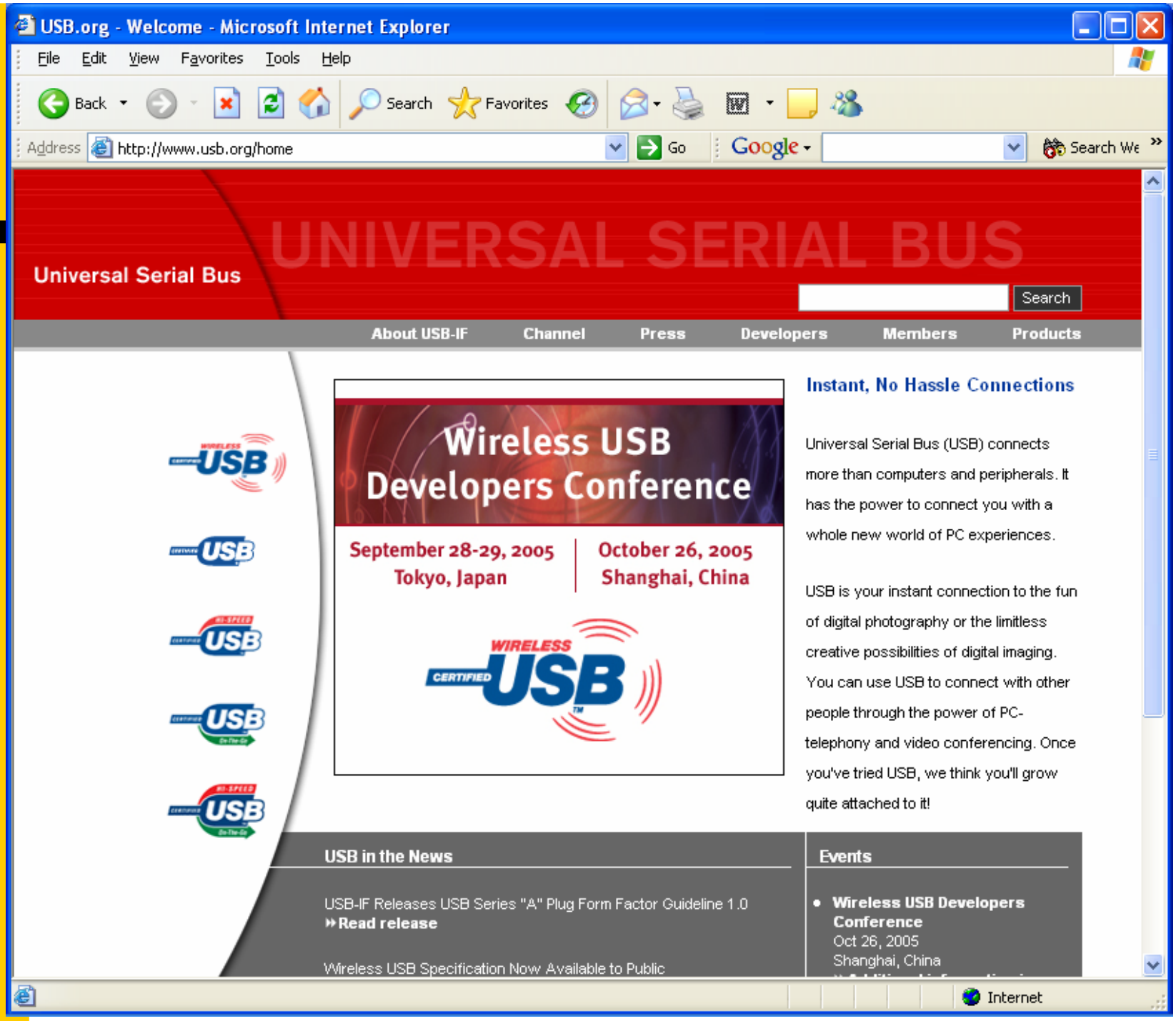
- Operating systems support
  - Win98, Win Me, Win 2000, Win XP
  - Linux
  - Macintosh
  - Some real-time kernels
- By support we here mean
  - Detect devices
  - Communicate in order to see how to exchange data
  - Provide mechanism so that software drivers can communicate with the device
  - At higher level: Provide class drivers for certain types of devices

Benefits for developers III

- Peripheral support
  - USB device hardware include a controller chip that manages details of USB communications
  - USB controllers based on popular architectures
    - Microchip PICMicro / M16C / Philips etc.etc.
      - Example Microchip USB Picdem



- USB Implementers Forum (USB-IF)
  - www.usb.org
  - Specifications, white papers, FAQ


USB.org - Welcome - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.usb.org/home
Go
Google
Search We
Universal Serial Bus
UNIVERSAL SERIAL BUS
Search
About USB-IF
Channel
Press
Developers
Members
Products
Wireless USB Developers Conference
September 28-29, 2005
Tokyo, Japan
October 26, 2005
Shanghai, China
WIRELESS CERTIFIED USB
Instant, No Hassle Connections
Universal Serial Bus (USB) connects more than computers and peripherals. It has the power to connect you with a whole new world of PC experiences.
USB is your instant connection to the fun of digital photography or the limitless creative possibilities of digital imaging. You can use USB to connect with other people through the power of PC-telephony and video conferencing. Once you've tried USB, we think you'll grow quite attached to it!
USB in the News
USB-IF Releases USB Series "A" Plug Form Factor Guideline 1.0
Read release
Wireless USB Specification Now Available to Public
Events
Wireless USB Developers Conference
Oct 26, 2005
Shanghai, China
Internet

USB Limitations

- Speed
  - USB 2.0 480 Mbit/s -- *Firewire IEEE-1394a 400 Mbit/s, IEEE-1394b 3.2 Gbit/s*
- Distance
  - Cable segment max 5 m (*Ethernet 100 m*)
- Peer-to-peer communication
  - USB: Only host to peripheral
  - (e.g. *IEEE-1394 offers direct peer-to-peer*)
  - USB On-The-Go: Partial solution
    - Peripheral can work as both peripheral and limited capability host

USB Limitations II

- Broadcasting
  - No way to send to multiple devices
- Legacy hardware
  - Old hardware must have converter (e.g. RS-232)
- Developer challenges
  - Protocol complexity
    - Developer must know USB protocols (both on peripheral and on PC end)
- Fees
  - Administrative fee for Vendor ID is $1500
    - Problem for developers of small quantities of inexpensive devices

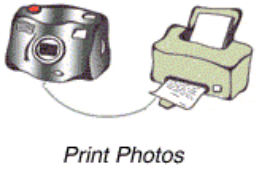
USB Evolution

- USB 1.0 January 1996
- USB 1.1 September 1998
  - New transfer method interrupt OUT
- USB 2.0 April 2000
  - New bus speed: High speed
- Windows 95 OEM Service Release 2
  - First PC OS with USB support, however limited and buggy
- Windows 98
  - Things are getting standard and working
- Windows NT (4.0) – No USB support, Windows 2000 → USB Support

USB 2.0

- New bus speed 480 Mbit/s
  - Scanners, printers, disk drives, video
- Completely hardware compatible
- External USB 2.0 hub
  - Must support all speeds: low, full, high
- USB 2.0 device
  - Works when connected to either 1.x or 2.0 PC
  - However:
    - ”Hi-Speed USB” – Support high speed
    - Otherwise:
      - ”Compatible with the USB 2.0 Specification”
      - ”Works with USB and Hi-Speed USB Systems”

USB and more

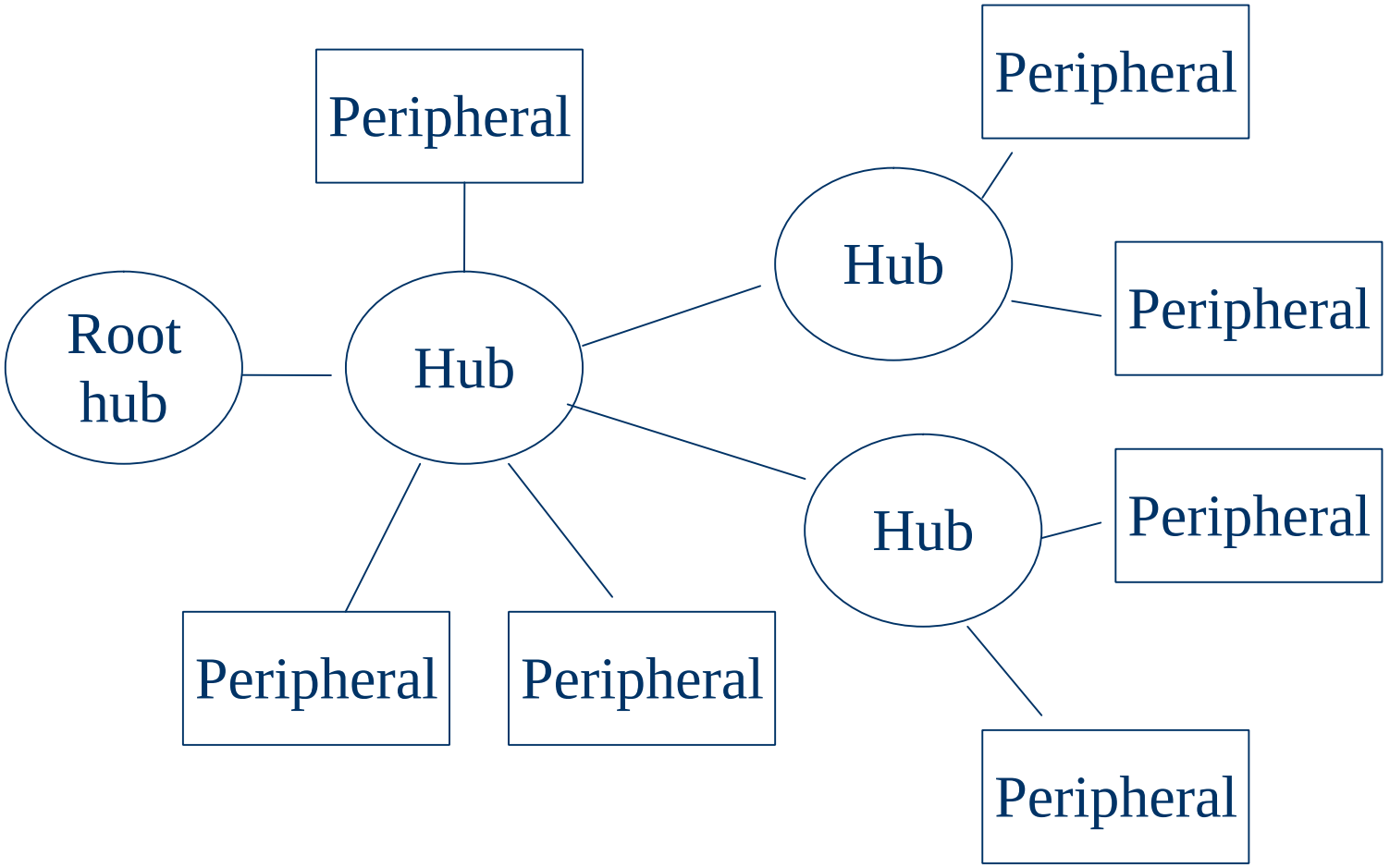


- USB On-The-Go
  - USB 2.0 supplement
    - Defines limited-capability host functions
- Wireless USB
  - Wireless communications up to 480 Mbit/s
  - Version 1.0, May 2005
  - Ultra Wide Band (UWB), up to ~3 m

USB and others

- USB vs. IEEE-1394
  - IEEE-1394 more flexible
  - IEEE-1394 devices can communicate with each others
  - Communication to multiple receivers
  - More expensive
- USB vs. Ethernet
  - Long cables, broadcasts, Internet protocols
  - Hardware is more complex on Ethernet

USB topology



Terminology

- Function
  - Devices provides capabilities to the host = functions
- Hub
  - Upstream connector for host communication
  - Downstream connectors to devices
- Device
  - “Function of a hub” – definition by USB specification
  - Composite devices: Many functions
- Port
  - Physical connector on a bus is a port
  - All ports on the same bus share the bus’ bandwidth

Division of labor

- Host duties
  - Detecting devices
  - Managing data flow
  - Error checking
  - Providing power
  - Exchanging data with peripherals

Division of labor II

- Peripheral duties
  - Detecting communication directed to the chip
  - Responding to standard requests
  - Error checking
  - Managing power
  - Exchanging data with the host

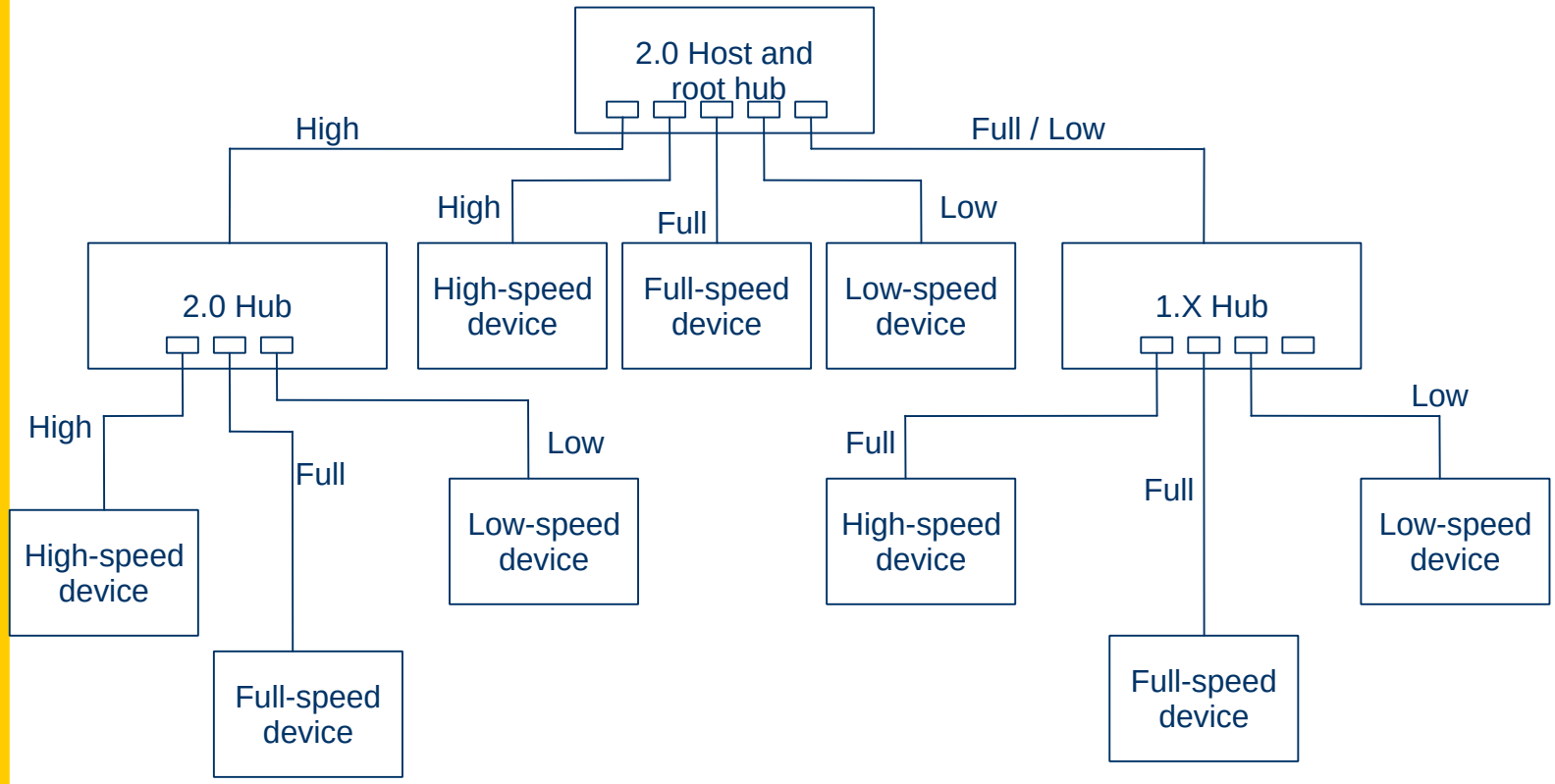
USB Transfers

- USB communication
  - Communication used for enumeration of the devices
    - A series of standard requests from the host
    - On a Windows PC, no user programming involved
  - Communication used to carry out the device´s purpose
    - Applications can use standard Windows API functions to read and write the device
    - Device: Placing data in USB controller´s transmit buffer

USB Transfers

- Data on the bus
  - Two signal lines carry data in USB
  - Host manages traffic by dividing time into Frames
  - Each frame is 1 ms
    - In high speed traffic, each frame is divided into 8 microframes,125 μs each
- Host speed vs. bus speed
  - USB 2.0 host: low, full and high speed
  - USB 1.x host: low and full speed
  - USB 1.x hub: Do not convert between speeds, only passes data up or down
  - USB 2.0 hub: Converts if necessary between speeds
  - The traffic on a bus segment is high only if
    - device, host and all hubs between are 2.0 compliant

Bus speeds


2.0 Host and root hub
High
High
Full
Low
Full / Low
2.0 Hub
High-speed device
Full-speed device
Low-speed device
1.X Hub
High
Full
Low
High-speed device
Full-speed device
Low-speed device
Full
Low
High-speed device
Full-speed device
Low-speed device

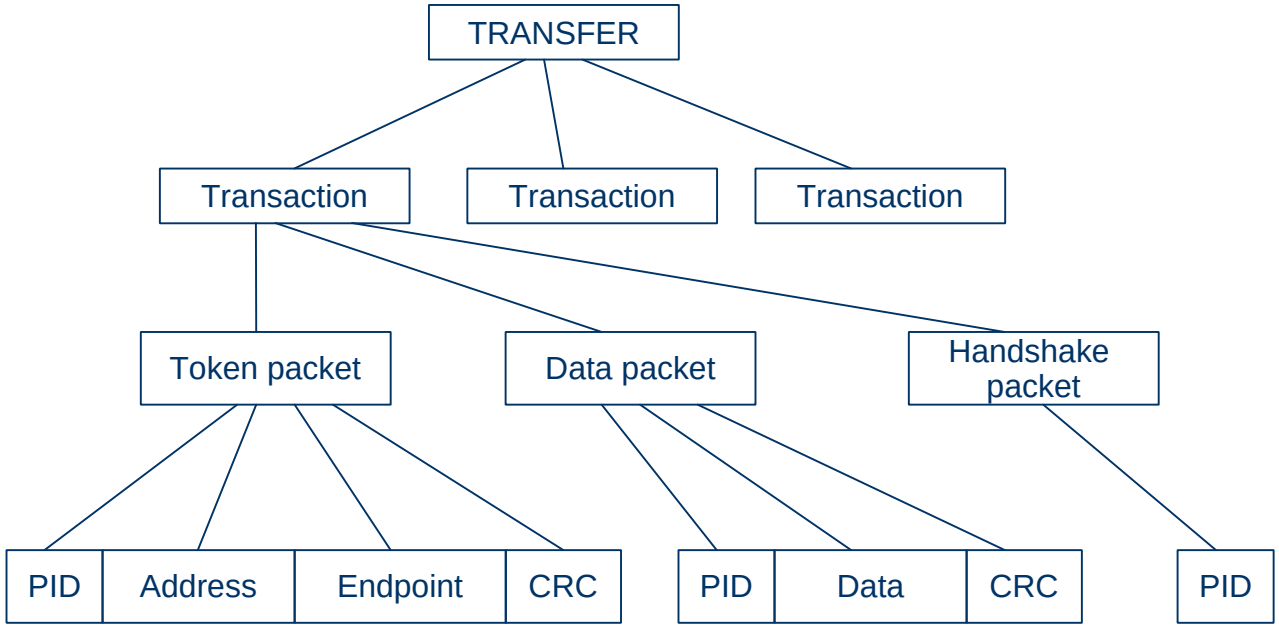
Device endpoints

- All bus traffic travels to or from a device endpoint
  - Block of data memory or register in the controller chip
- Uniquely addressable portion of a USB device that is the source or sink
- Address: Endpoint number (0-15) and direction
  - Direction defined from the host´s perspective
    - IN: Data to the host, OUT: Data to the device
  - Endpoint 0: Control endpoint (both IN and OUT)
  - Full, High-speed: Up to 30 endpoints (15 addressable endpoints, each can be both IN and OUT)
  - Low speed: Two additional endpoints (e.g. 1 IN+1OUT, 2 OUT, or 2 IN)

USB Transaction

- USB transfers consists of one or more transactions
- Transactions begins with a packet containing endpoint number and direction of data flow (IN, OUT, Setup)
- Setup is a special type of transaction that initializes a control transfer

USB Transaction II



USB Pipes

- Host and device establish a pipe before a transfer
- A Pipe is an association between a device's endpoint and the host controller software
- Pipes are established during the enumeration process
- Every device has a default control pipe using endpoint 0

Type of transfers

- Control
  - Identification and configuration
- Bulk
  - Printer, scanner, drive
- Interrupt
  - Mouse, keyboard
- Isochronous
  - Streaming audio, video

Control transfers

- Using endpoint 0 only
- Recognition and configuration of devices
- 10 % (High-Speed 20%) of bandwidth allocated
- Types
  - GET_STATUS
  - CLEAR_FEATURE
  - SET_FEATURE
  - SET_ADDRESS
  - GET_DESCRIPTOR
  - SET_DESCRIPTOR
  - GET_CONFIGURATION
  - SET_CONFIGURATION
  - GET_INTERFACE
  - SET_INTERFACE
  - SYNCH_FRAME

Bulk transfers

- Transferring large amount of data that not is time critical
- Only full and high speed devices
- Using only available bandwidth
- Scanners, printers, hard disks

Interrupt transfers

- Small amount of data
- Known latency
- Interval (max interval) 1-255 ms

Isochronous transfers

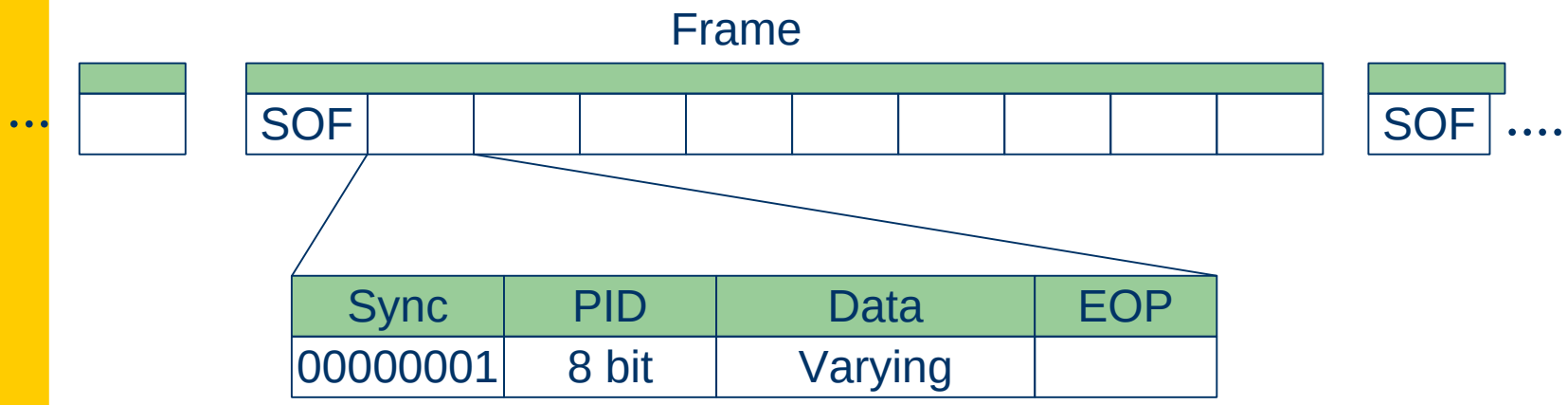
- Time-critical data with constant bandwidth
- Typically video, audio content
- No error correction/detection

Time critical transfers

- Interrupt or isochronous transfers
- If host has promised a requested USB bandwidth, there is no guarantee that host and device are ready to communicate when the bus is available
- High-speed devices must provide a default interface
  - Interrupt transfers: limited to packet size of 64 B
  - Isochronous transfers: not allowed
  - After enumeration, the device driver can request larger bandwidth

USB Frames

- Time is divided into frames of 1 ms (high speed 0,125 μs)



- PID: 4 bit PID + 4 bit PID-check

USB Frames II

- Frames are used for dividing the bus bandwidth between devices
  - Interrupt transfers (max)
    - Low: 8 bytes every 10 milliseconds → 800 B/s
    - Full: 64 bytes every millisecond (frame) → 64 kB/s
    - High:3 1024 byte packets each microframe → 24 MB/s
  - Bulk transfers (max,only if bus is free!!)
    - Full: nineteen 64 B packets / frame → 1260 B/frame → 1.26 MB/s
    - High : thirteen 512 B packets / microframe → 6656 B/microframe → 53.3 MB/s
  - Isochronous (max)
    - Full: 1023 B / frame → 1.0 MB/s
    - High: three 1024 B packets / microframe → 24.5 MB/s

Enumeration

- The enumeration process is when the host finds out which devices are attached and assign device drivers
  - Assigning a address to the device
  - Reading descriptors from the device
  - Assigning, loading device drivers
  - Selecting a configuration
    - Power requirements
    - Endpoints
    - Other features

The process

- Boot-up: Hosts polls root hub to learn if devices attached; After boot-up, periodical poll
- When new device found
  - Asks the hub to establish a communication path between host and device
  - Sends standard control transfers to endpoint 0
  - The device responds to each request by returning requested information
- After enumeration, the devices is added to the list of devices

Enumeration steps - Windows

1. The device is attached to a USB port → *Powered state*
2. The hub detects the device
3. The host learns of a new device
    1. The hub reports an event on the interrupt port
4. The hub detects if the device is low or full speed
5. The hub resets the device
6. The host learns if a full-speed device supports high speed
7. The hub establishes a signal path between the device and the host → *Default state*

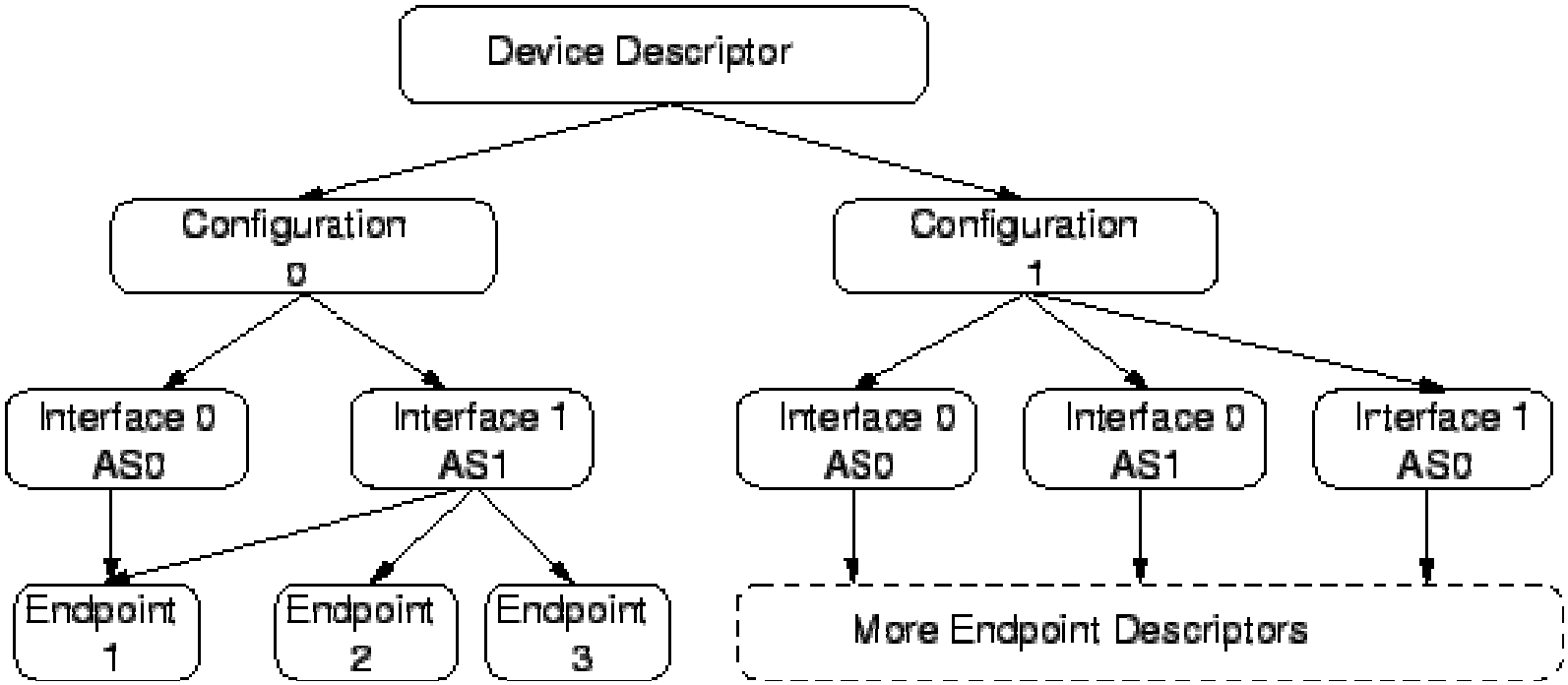
Enumeration steps II

1. The host requests a descriptor to learn the maximum packet size of the default pipe, address 0, endpoint 0
2. The hosts assigns an address → *Address state*
3. The host learns about the device's capabilities
4. The host assigns and loads a device driver
    1. VendorID, ProductID, release number of device
5. The host´s device driver selects a configuration → *Configured state*

Descriptors

- Device descriptor
  - General information, applies globally to the device
- Configuration descriptor
  - One or more configuration descriptors
    - One or more interfaces, with zero or more endpoints
- Interface descriptor
  - One or more settings per interface
- Endpoint descriptor
  - e.g. Bandwidth requirements
- String descriptors
  - Optional strings for e.g. vendor names

Device descriptors



Device classes

- Standard device classes
  - Specifications found at *www.usb.org/developers/devclass_docs*

Device class	**Example device**
Display	Monitor
Communication	Modem
Audio	Speakers
Mass storage	Hard drive
Human interface	Data glove

Ex1 - Human Interface Devices-HID

- Devices used by humans to interact with computer systems
  - Keyboards and pointing devices for example, standard mouse devices, trackballs, and joysticks
- Communicates using control and interrupt transfers
  - *bInterfaceClass = 03h*
  - In Windows, applications can communicate using API functions
    - HidD_SetFeature, HidD_GetFeature
    - However, Windows request exclusive access to input reports

Ex2 - Mass storage

- For file transfers in one or both directions
  - Flash, floppy, hard, CD, DVD –drives
- Bulk-only or Control/Bulk/Interrupt (CBI)
  - Bulk-only recommended for all new devices
  - Actual data is transfered using bulk
- Uses different industry-standard command-block sets for controlling devices
  - ATA/ATAPI, SFF-8070i, SCSI Block Command SCSI/ATAPI, Universal Floppoy Inteface

Ex3 - Test and Measurement class

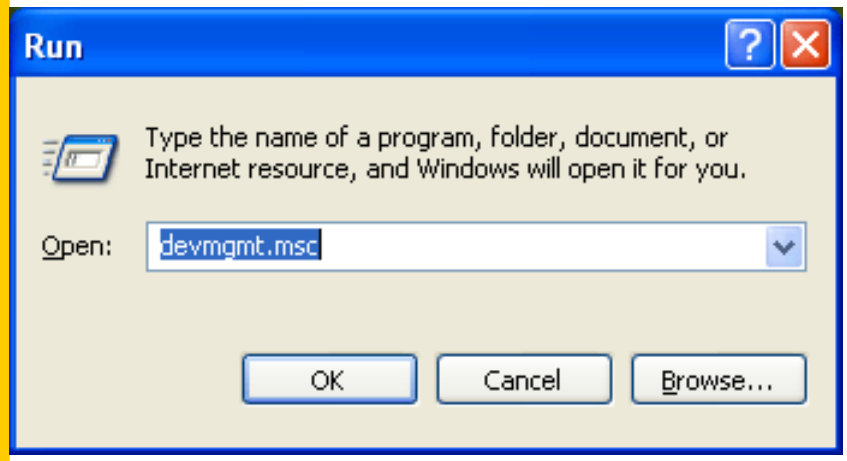
- Test and Measurement (USBTMC)
  - Instrumentation class where data does not need guaranteed timing
  - ADCs, DACs, sensors, transducers
  - USB488 subclass defines protocols for communication using IEEE-488 data format and commands
  - Bulk IN and bulk OUT endpoint
  - Interrupt IN for USB488 subclass devices
  - Windows does not include drivers for this class
    - HID class / mass-storage can

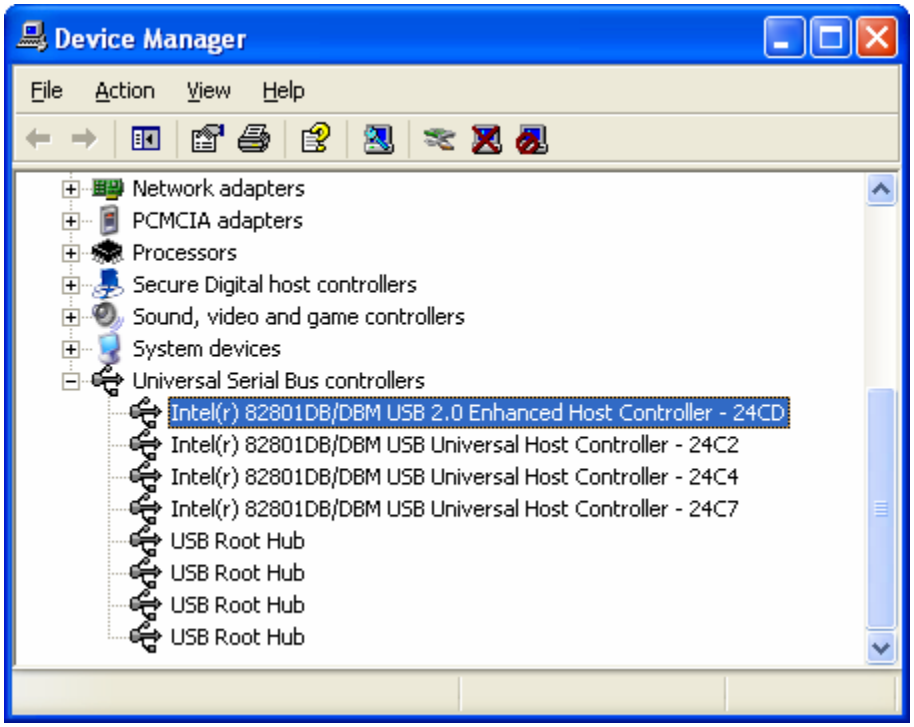
USB host controller types

- USB Enhanced Host Controller (EHCI)
  - High-Speed
- USB Universal Host Controller (UHCI)
  - Full and low speed
- USB Open Host Controller (OHCI)
  - Full and low speed
- Motherboards that support USB 2.0 must have at least one Enhanced Host Controller and usually one or more UHCI or OHCI controllers

USB and Windows

- Does my computer have any USB devices ?

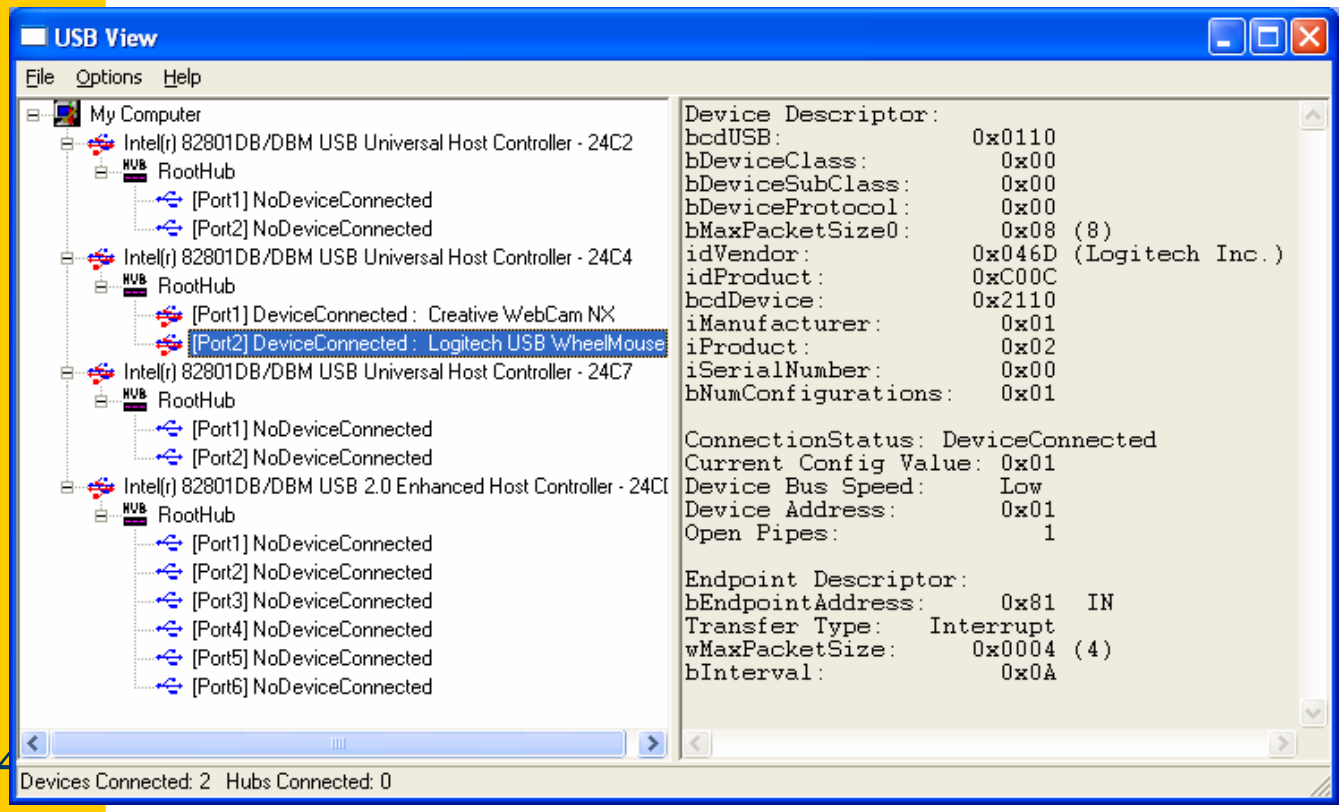




USB and windows

- WinDDK example: USB View
  - enumeration of USB hosts, hubs and devices

\WINDDK\\src\wdm\usb\usbview



USB View II - Isochronous

USB View

File Options Help

- My Computer
  - Intel(r) 82801DB/DBM USB Universal Host Controller - 24C2
    - RootHub
      - [Port1] NoDeviceConnected
      - [Port2] NoDeviceConnected
  - Intel(r) 82801DB/DBM USB Universal Host Controller - 24C4
    - RootHub
      - [Port1] DeviceConnected : Creative WebCam NX
      - [Port2] NoDeviceConnected
  - Intel(r) 82801DB/DBM USB Universal Host Controller - 24C7
    - RootHub
      - [Port1] NoDeviceConnected
      - [Port2] NoDeviceConnected
  - Intel(r) 82801DB/DBM USB 2.0 Enhanced Host Controller - 24CI
    - RootHub
      - [Port1] NoDeviceConnected
      - [Port2] NoDeviceConnected
      - [Port3] NoDeviceConnected
      - [Port4] NoDeviceConnected
      - [Port5] NoDeviceConnected
      - [Port6] NoDeviceConnected

```
Device Descriptor:
bcdUSB:              0x0110
bDeviceClass:          0xFF
bDeviceSubClass:       0x00
bDeviceProtocol:       0x00
bMaxPacketSize0:       0x08 (8)
idVendor:            0x041E (Creative Labs
idProduct:           0x401C
bcdDevice:           0x0100
iManufacturer:         0x00
iProduct:              0x00
iSerialNumber:         0x00
bNumConfigurations:    0x01

ConnectionStatus: DeviceConnected
Current Config Value: 0x01
Device Bus Speed:      Full
Device Address:        0x01
Open Pipes:               2

Endpoint Descriptor:
bEndpointAddress:      0x81  IN
Transfer Type: Isochronous
wMaxPacketSize:      0x0000 (0)
bInterval:             0x01

Endpoint Descriptor:
bEndpointAddress:      0x82  IN
Transfer Type:    Interrupt
wMaxPacketSize:      0x0008 (8)
bInterval:             0x0A
```

Devices Connected: 1 Hubs Connected: 0

USB View III - Bulk

USB View

File Options Help

- My Computer
  - Intel(r) 82801DB/DBM USB Universal Host Controller - 24C2
    - RootHub
      - [Port1] NoDeviceConnected
      - [Port2] NoDeviceConnected
  - Intel(r) 82801DB/DBM USB Universal Host Controller - 24C4
    - RootHub
      - [Port1] NoDeviceConnected
      - [Port2] DeviceConnected : Logitech USB WheelMouse
  - Intel(r) 82801DB/DBM USB Universal Host Controller - 24C7
    - RootHub
      - [Port1] NoDeviceConnected
      - [Port2] NoDeviceConnected
  - Intel(r) 82801DB/DBM USB 2.0 Enhanced Host Controller - 24C[
    - RootHub
      - [Port1] NoDeviceConnected
      - [Port2] NoDeviceConnected
      - [Port3] DeviceConnected : USB Mass Storage Device
      - [Port4] NoDeviceConnected
      - [Port5] NoDeviceConnected
      - [Port6] NoDeviceConnected

```
Device Descriptor:
bcdUSB:             0x0200
bDeviceClass:         0x00
bDeviceSubClass:      0x00
bDeviceProtocol:      0x00
bMaxPacketSize0:      0x40 (64)
idVendor:           0x0D7D
idProduct:          0x1600
bcdDevice:          0x0100
iManufacturer:        0x01
iProduct:             0x02
iSerialNumber:        0x03
bNumConfigurations:   0x01

ConnectionStatus: DeviceConnected
Current Config Value: 0x01
Device Bus Speed:     High
Device Address:       0x01
Open Pipes:              3

Endpoint Descriptor:
bEndpointAddress:     0x81  IN
Transfer Type:        Bulk
wMaxPacketSize:     0x0200 (512)
bInterval:            0x00

Endpoint Descriptor:
bEndpointAddress:     0x02  OUT
Transfer Type:        Bulk
wMaxPacketSize:     0x0200 (512)
bInterval:            0x00

Endpoint Descriptor:
bEndpointAddress:     0x83  IN
Transfer Type:    Interrupt
wMaxPacketSize:     0x0040 (64)
bInterval:            0x04
```

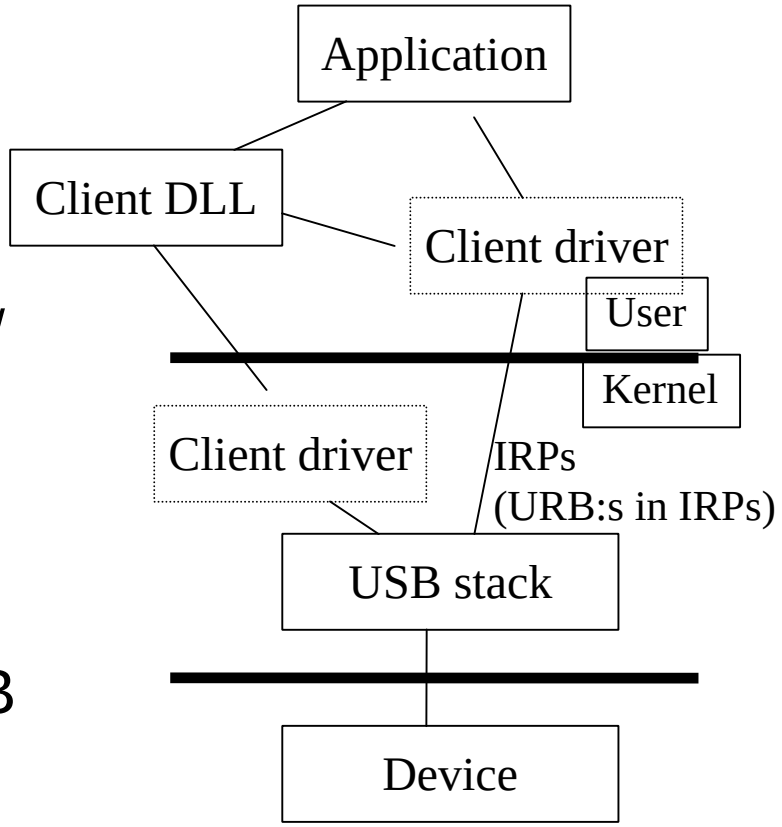
Devices Connected: 2 Hubs Connected: 0

Windows USB class support

- Bluetooth (Bthusb.sys)
- Chip/smart card [CCID](Usbccid.sys)
- Hub (Usbhub.sys)
- Human interfaces [HID](Usbhid.sys)
- Mass storage [MSC](Usbstor.sys)
- Printing [PTP](Usbprint.sys)
- Scanning/imaging (Usbscan.sys)
- USB Audio (Usbaudio.sys)
- Modem [CDC] Usbser.sys)
- Video [UVC](Usbvideo.sys)

Windows drivers hierarchy

- Kernel mode
  - Complete access to all data structures / memory / I/O
- User mode
  - Restricted access to memory / I/O
- Communication between layers using I/O Request Packets (IRPs)
- USB communications i USB Request Blocks (URBs)



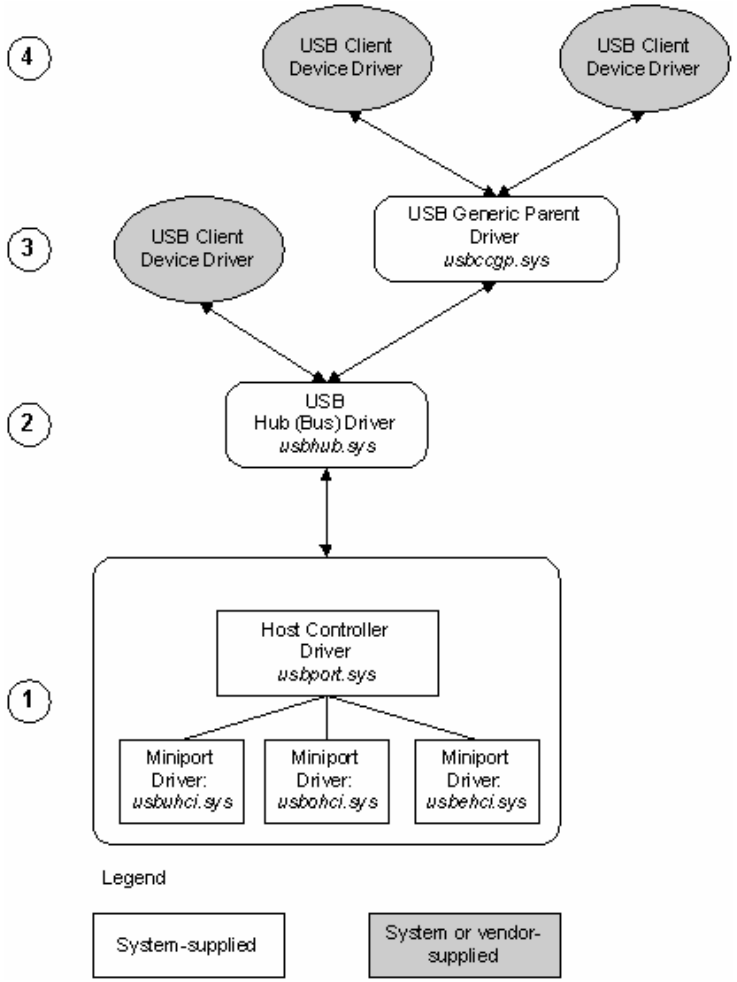
Client application – driver comm.

- Communcation with drivers often available using API functions
  - CreateFile()
  - ReadFile(), WriteFile(), DeviceIoControl()
  - CloseHandle()

```
// .. from WinDDK: bulkusb.c
   DriverObject->MajorFunction[IRP_MJ_DEVICE_CONTROL] = BulkUsb_DispatchDevCtrl;
   DriverObject->MajorFunction[IRP_MJ_POWER]          = BulkUsb_DispatchPower;
   DriverObject->MajorFunction[IRP_MJ_PNP]            = BulkUsb_DispatchPnP;
   DriverObject->MajorFunction[IRP_MJ_CREATE]         = BulkUsb_DispatchCreate;
   DriverObject->MajorFunction[IRP_MJ_CLOSE]          = BulkUsb_DispatchClose;
   DriverObject->MajorFunction[IRP_MJ_CLEANUP]        = BulkUsb_DispatchClean;
   DriverObject->MajorFunction[IRP_MJ_READ]           =
   DriverObject->MajorFunction[IRP_MJ_WRITE]          = BulkUsb_DispatchReadWrite;
   DriverObject->MajorFunction[IRP_MJ_SYSTEM_CONTROL] = BulkUsb_DispatchSysCtrl;
   DriverObject->DriverUnload                         = BulkUsb_DriverUnload;
   DriverObject->DriverExtension->AddDevice           = (PDRIVER_ADD_DEVICE)
```

Windows (XP) core USB drivers

- Usbport.sys
- Usbhub.sys
- Usbehci.sys
- Usbuhci.sys
- Usbohci.sys
- Usbccgp.sys



Drivers basics

- Drivers run in OS kernel mode
  - Drivers must be rigorously tested before deployment
  - Time consuming
- Alternatives to providing own driver
  - Use a MS supplied driver
  - Move all or part of driver to user mode
  - Software only driver (can be moved to user mode)
  - Is a user mode DLL an alternative to a driver?

Types of Windows drivers

- Legacy drivers
  - From NT, today only for pure software kernel functionality
- Windows drivers
  - Current drivers with Plug and Play and Power management
    - Windows Driver Model (WDM) -- USB drivers normally here
- Bus drivers
  - Special drives for bus enumeration arbitrates bus access
- File system drivers
- Printer drivers
- Graphics drivers
- Storage miniport drivers
- Filter drivers

Resource for driver development

- System for development
- System for debugging
  - Serial, IEEE1394 port for debugging
- Windows Drivers Development Kit (DDK)
  - Use latest version (currently Windows server 2003)
    - Can be used for developing drivers for older versions of windows
- WinDbg debugger

USB driver choices

- Windows Driver Model (WDM) driver
  - Rules that every WDM driver must implement
  - Behavior to get right
  - Details to effectively test
- No driver model – User mode “driver”
  - Use one of the base USB class drivers and access the device using corresponding class API

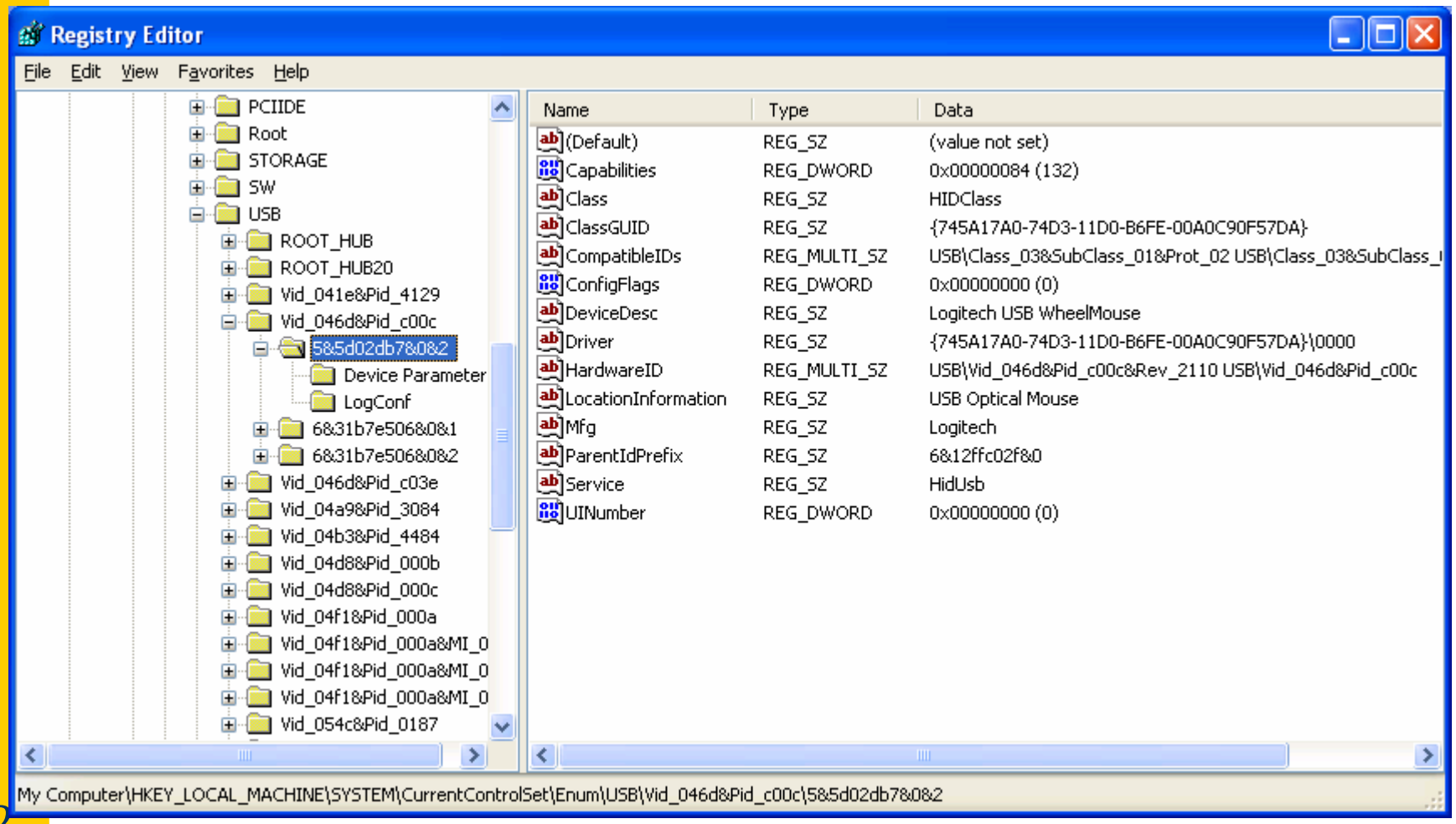
Automated driver generation

- USB communications follow protocols defined in USB specification. Why not a generic USB driver??
  - Jungo’s WinDriver USB Device toolkit
    - Applications communicates with generic USB driver
  - Jungo’s Kernel-Driver USB
    - Custom kernel mode driver

Locating Windows drivers

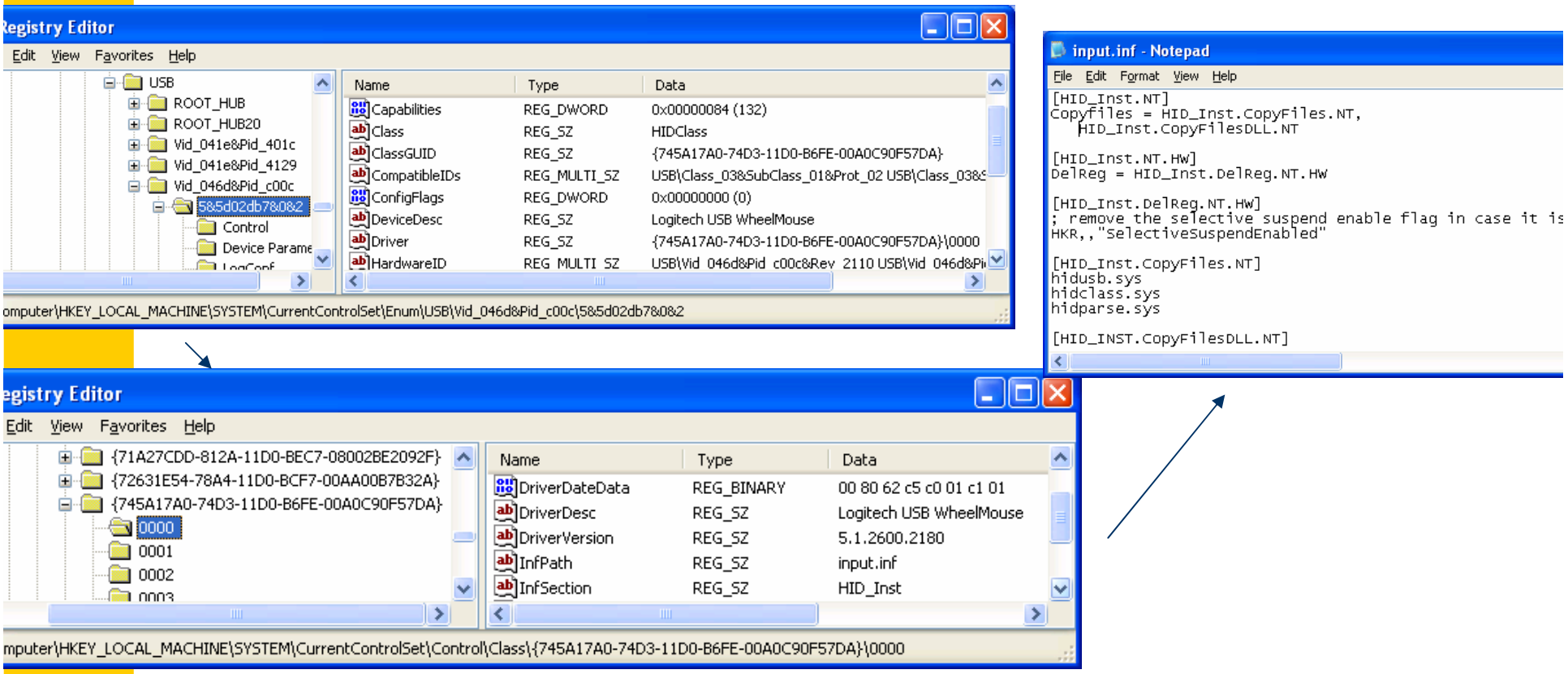
- Registry hardware key
  - HKEY_LOCAL_MACHINE\System\CurrentControlSet\Enum\USB**Vid_XXXX&Pid_XXXX**
    - Under this key, instances of the device
      - Class GUID
      - Driver GUID
- Registry class key
  - HEKY_LOCAL_MACHINE\System\CurrentControlSet\Control\Class
    - Gives inf-file → gives driver filename
- Globally Unique Identifier (GUID)
  - 128 bit value identifying a class or other entity

Registry: USB entries



Example – locating drivers

- Logitech wheel mouse VID = 046d, Pid = 000c

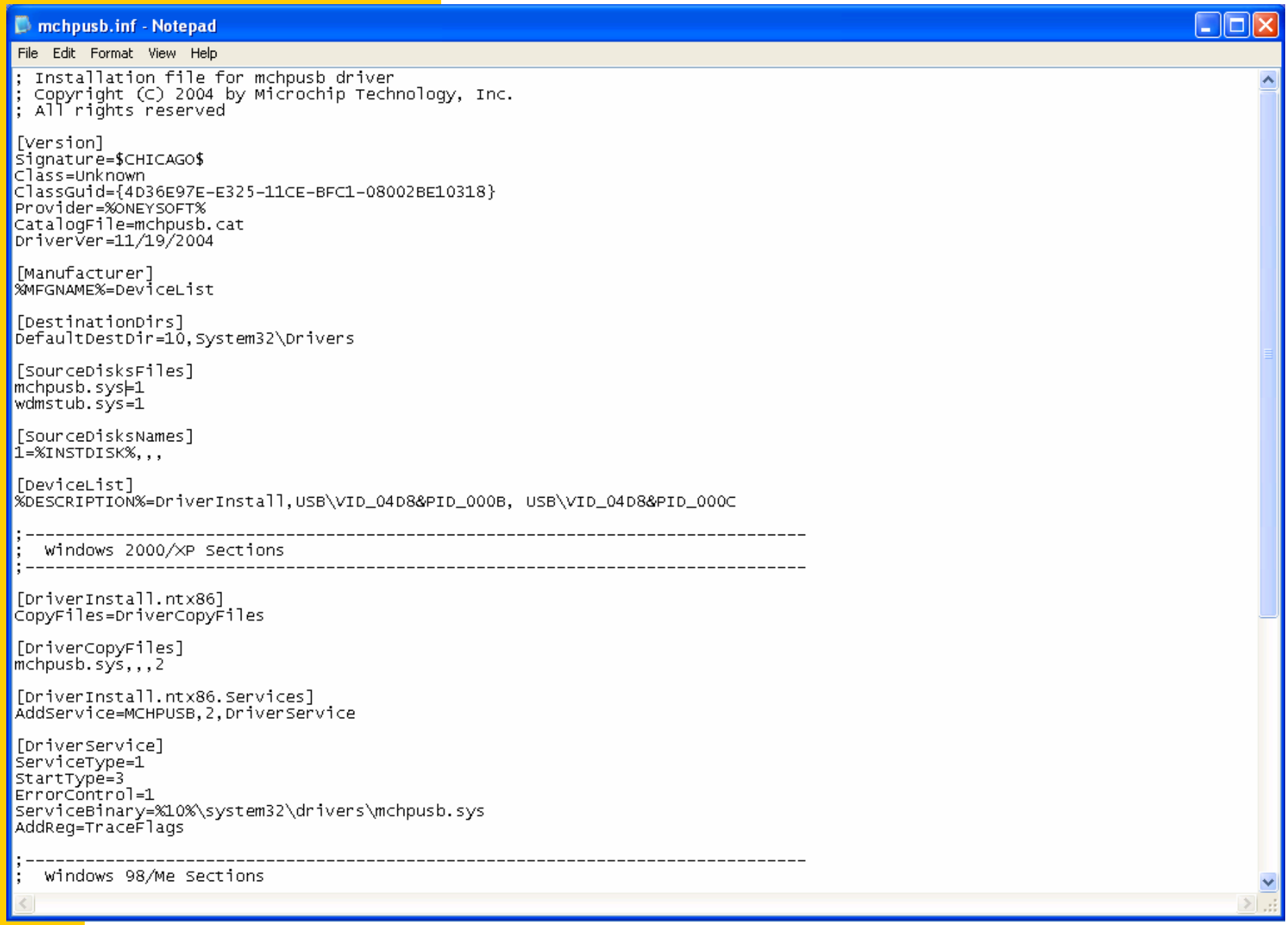


Windows INF-files

- A device setup information file:
  - text file, in directory *%SystemRoot%\inf*
  - information about one or more devices in a device setup class
  - tells windows what drivers to use
  - information on what to store in registry
- When a device is attached to the system
  - When descriptors read from device, try to find matching inf-file in system

Matching devices to inf-files

- [DeviceList]
  - Main ID, additional (compatible) ID:s (VID+PID)
- For all matching inf-files, rank the available files
  - Trusted (signed) drivers get higher rank
  - Hardware ID matching gives higher rank than compatible devices
  - If no match can be found, windows starts the ”new hardware found” wizard


mchpusb.inf - Notepad
File Edit Format View Help
; Installation file for mchpusb driver
; Copyright (C) 2004 by Microchip Technology, Inc.
; All rights reserved
[Version]
Signature=$CHICAGO$
Class=Unknown
ClassGuid={4D36E97E-E325-11CE-BFC1-08002BE10318}
Provider=%ONEYSOFT%
CatalogFile=mchpusb.cat
DriverVer=11/19/2004
[Manufacturer]
%MFGNAME%=DeviceList
[DestinationDirs]
DefaultDestDir=10,System32\Drivers
[SourceDisksFiles]
mchpusb.sys=1
wdmstub.sys=1
[SourceDisksNames]
1=%INSTDISK%,,,
[DeviceList]
%DESCRIPTION%=DriverInstall,USB\VID_04D8&PID_000B, USB\VID_04D8&PID_000C
;--
; Windows 2000/XP Sections
;--
[DriverInstall.ntx86]
CopyFiles=DriverCopyFiles
[DriverCopyFiles]
mchpusb.sys,,,2
[DriverInstall.ntx86.Services]
AddService=MCHPUSB,2,DriverService
[DriverService]
ServiceType=1
StartType=3
ErrorControl=1
ServiceBinary=%10%\system32\drivers\mchpusb.sys
AddReg=TraceFlags
;--
; Windows 98/Me Sections

Upcoming driver models

- Windows Driver Foundation (WDF)
  - KMDF (Kernel Mode Driver Foundation)
    - Drivers that need access to hardware resources like I/O ports, Interrupts, DMA
    - Need for communication with other kernel drivers
    - Need for tight protocol timing etc.
  - UMDF (User Mode Driver Foundation)
    - Drivers in user mode run in a separate process, driver functions passed on by a kernel reflector

WinUSB generic driver

- Kernel mode driver handles
  - Complex logging and I/O
  - Power management
  - PnP events etc.
- User mode DLL exposes
  - simpler user mode API
  - Incorrect implementation leads to app hang/crash (not PC crash)
  - Safer upgrades to new OS

WinUSB generic driver

- Potential candidates
  - Test and measurement devices
  - Sync/update utilities
  - University/independent projects
- When NOT using WinUSB
  - Bus drivers that need additional stacks in kernel
- Code development time
  - Beginners: 1-2 days for initial device communication
  - Advanced: Faster...

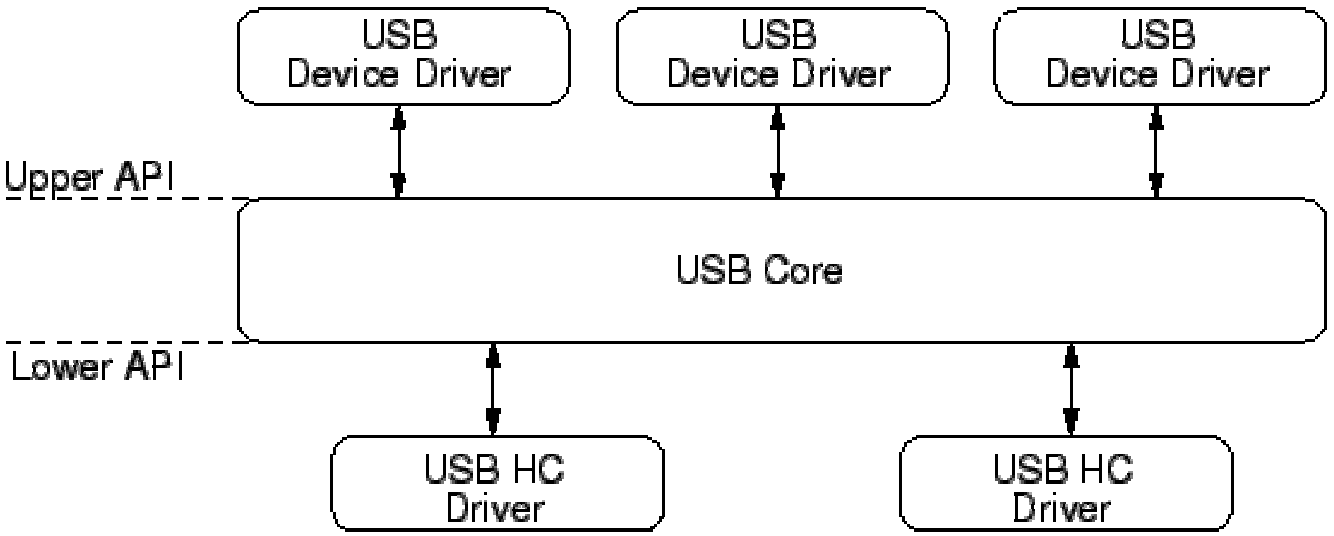
USB access and WinAPI

- Finding the device
  - Finding a pointer to device in a class
    - *SetupDiGetClassDevs(...)*
  - To enumerate device interfaces
    - SetupDiEnumDeviceInterfaces(...)
  - To get the device path name
    - SetupDiGetDeviceInterfaceDetail(...)
  - Use the path to get a communcations handle
    - handle = CreateFile(...)

USB access and WinAPI

- To get device descriptors
  - DeviceIoControl()
- To read from device
  - ReadFile(handle,...)
- To write to device
  - WriteFile(handle,...)

Linux USB Subsystem



Linux USB

- USB core
  - routines common to all USB devices
  - upper and lower API
- Device driver framework
  - Devices are registered and deregistered at the subsystem
    - A driver register 2 entry points and its name
      - probe(), disconnect()
    - Possible also to register file operations and a minor number
    - USB data structures start with usb_, to register a driver the structure usb_driver is needed

Linux USB framework functions

- int usb_register(struct usb_driver *drv);
- void usb_deregister(struct usb_driver *drv);
- void usb_driver_claim_interface(struct usb_driver *driver, struct usb_interface *iface, void *drv_context);
- int usb_interface_claimed(struct usb_interface *iface);
- void usb_driver_release_interface

Examples

- UsbView (ddk\src\wdm\usb\usbview)
- SnoopyPro (sourceforge.net/projects/usbsnoop)
- MCHPFSUSB (Microchip USB Toolkit)

Microchip demo

- Driver (binary only, debug / release version)
- MPUSBAPI (DLL)
  - MPUSBGetDeviceCount
  - MPUSBOpen
  - MPUSBClose
  - MPUSBRead
  - MPUSBReadInt
  - MPUSBWrite
  - MPUSBGetDLLVersion

Summary

- USB is a generic bus, development starting 1994, current version 2.0 April 2000
- Following transfer types
  - Control, Bulk, Interrupt, Isochronous
- Enumeration
  - Finding devices on the bus, allocating devices
- Host software basics for USB
  - Windows, Linux