USB

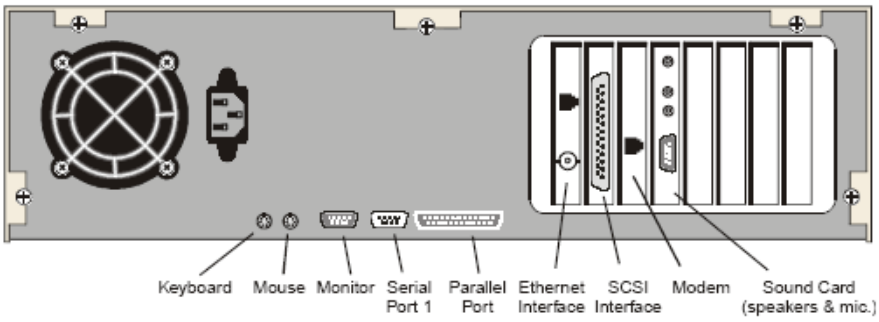
Universal Serial Bus

An example of a serial, half dupplex, master-slave, assynchronous, differential bus.

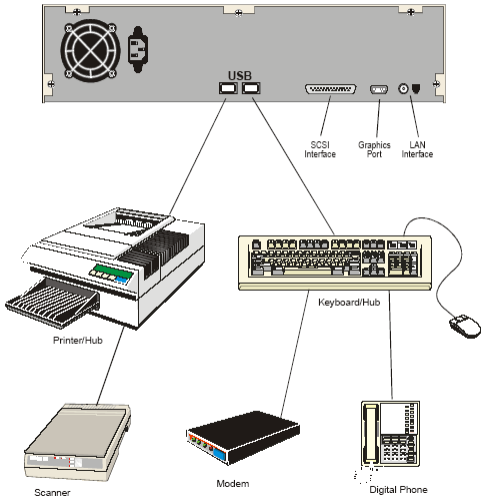
USB Universal Serial Bus

1. INTRO

Pre-USB Personal Computer connectivity structure



USB is intended to simplify PC peripherals connectivity



USB benefits

Feature	Description
Low Cost	The USB provides a low-cost solution for attaching peripheral devices to PCs.
Hot Pluggable	Device attachment is automatically detected by the USB and software automatically configures the device for immediate use, without user intervention.
Single Connector Type	The USB defines a single connector used to attach any USB device. Additional connectors can be added with USB hubs.
127 Devices	Supports the attachment of 127 devices per USB.
Low Speed, Full Speed, and High Speed Devices	The USB 2.0 supports three device speeds: 1.5Mb/s, 12Mb/s, and 480Mb/s.
Cable Power	Peripherals can be powered directly from the cable. 5.0vdc power is available from the cable. The current available can vary from 100ma - 500ma depending on the hub port.
System Resource Requirement Eliminated	USB devices, unlike their ISA, EISA, and PCI cousins, require no memory or I/O address space and need no IRQ lines.
Error Detection and Recovery	USB transactions include error detection mechanisms that are used to ensure that data is delivered without error. In the event of errors, transactions can be retried.
Power Conservation	USB devices automatically enter a suspend state after 3ms of no bus activity. During suspend devices can consume no more than 500µa of current.
Support for Four Types of Transfers	The USB defines four different transfer types to support different transfer characteristics required by devices. Transfer types include: bulk, isochronous, interrupt, and control transfers.
Ability to Extend Bus	USB hubs can be installed to add additional ports that permit additional devices to be attached.

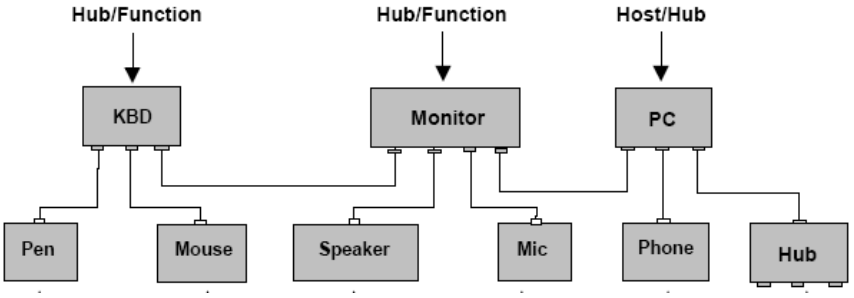
Specifications

	Performance	Applications	Attributes
LS	Low Speed: Interactive Devices 10-100 Kb/s	Keyboard, Mouse Stylus Game peripherals Virtual Reality peripherals	Lower cost Hot plug Ease of use Multiple peripherals
FS	Medium Speed: Phone, Audio 500-10,000 Kb/s	ISDN PBX POTS Digital audio Scanner Printer	Lower cost Ease of use Guaranteed latency Guaranteed bandwidth Hot plug Multiple devices
(USB 2.0) HS	High Speed: Video, Disk, LAN 25-500 Mb/s	Mass storage Video conferencing Imaging Broadband	Low cost Hot plug High bandwidth Guaranteed bandwidth Guaranteed latency Multiple devices Ease of use

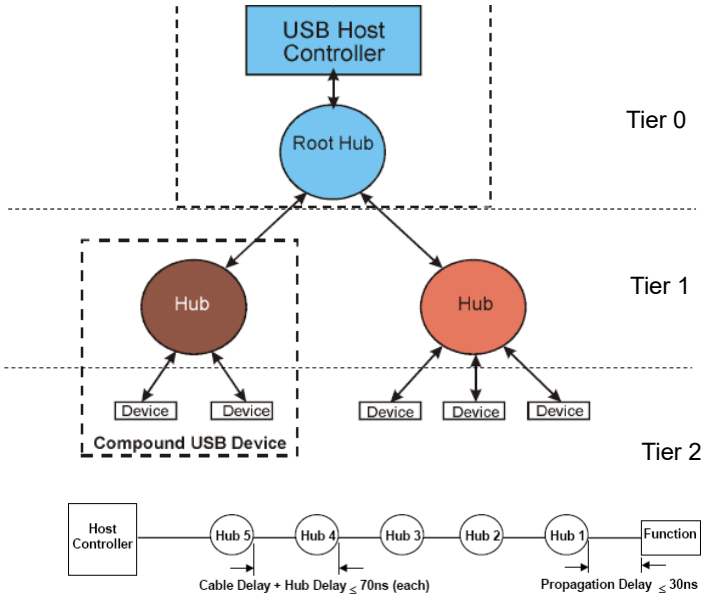
Low Speed = 1.5 Mb/s (USB 1.1)
Full Speed = 12Mb/s (USB 1.1)
High Speed = 480Mb/s (USB 2.0)
Super Speed= 5Gb/s (USB 3.0)

Connectivity examples

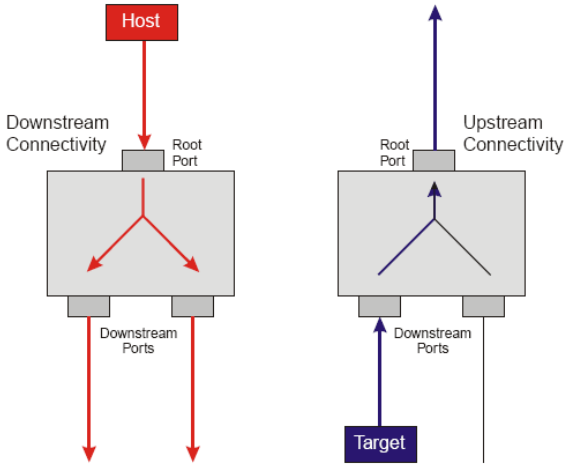
1 Host (master) controller
Up to 127 devices (slaves)
Some devices can perform as USB hubs
Up to 6 levels



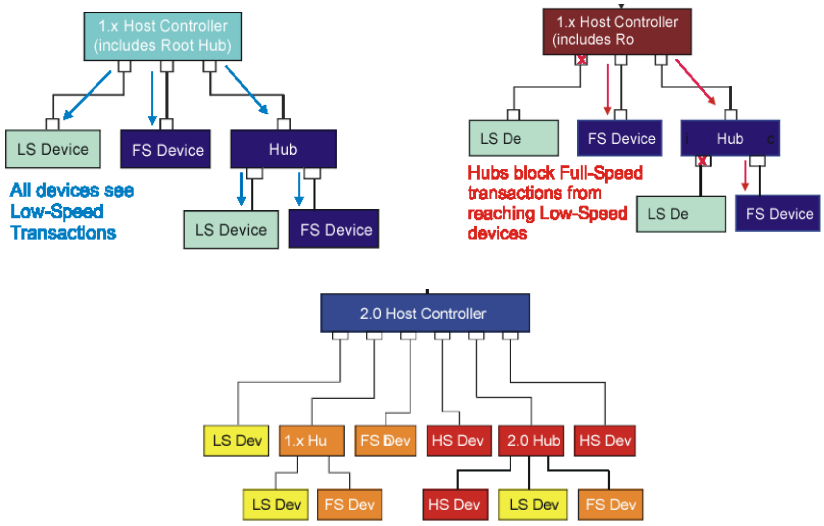
Hubs are used to increase the number of available ports



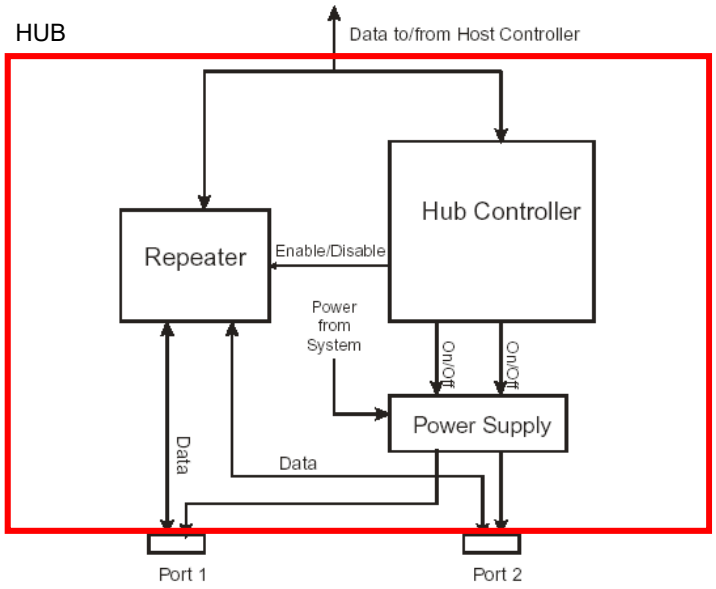
HUB Functions (I): repiter



HUB Functions (II): data throughput control



HUB Functions (III): Power control and regulation

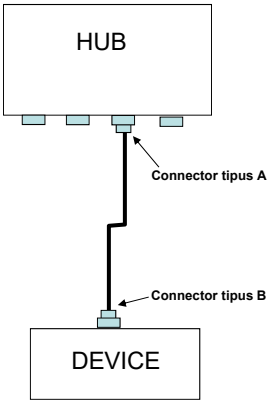


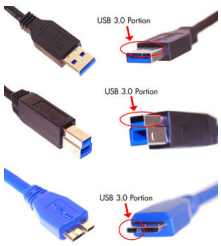
USB Connectors

Series "A" Connectors	Series "B" Connectors
♦ Series "A" plugs are always oriented **upstream** towards the *Host System*	♦ Series "B" plugs are always oriented **downstream** towards the *USB Device*
"A" Plugs (*From the USB Device*)	**"B" Plugs** (*From the Host System*)
"A" Receptacles (*Downstream Output from the USB Host or Hub*)	**"B" Receptacles** (*Upstream Input to the USB Device or Hub*)

Connector B Mini

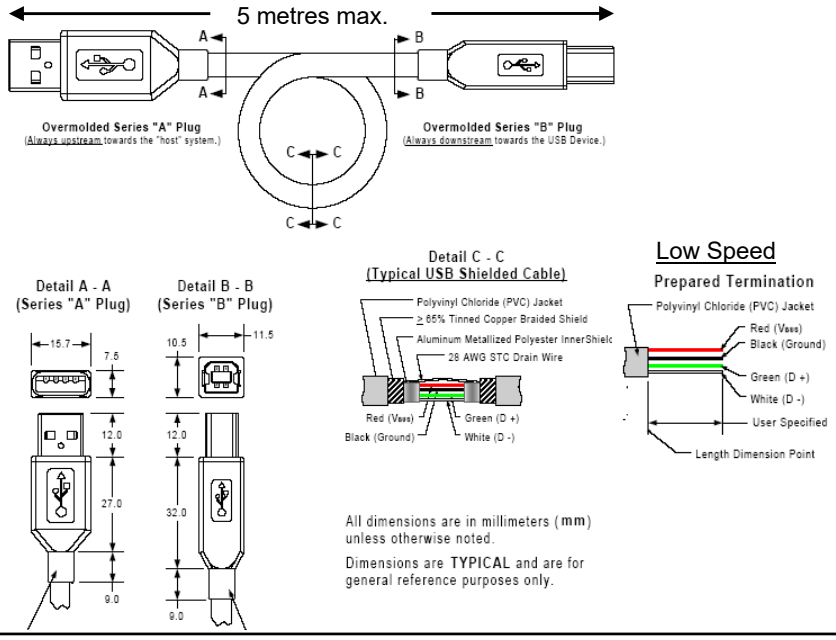
Connector B Maxi





USB 3.0

USB WIRE

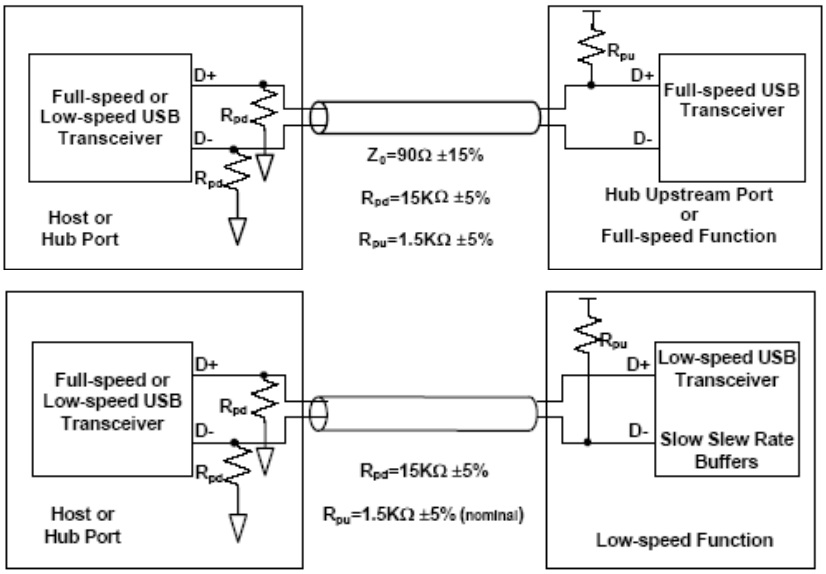


USB Universal Serial Bus

2. HOW IT WORKS?

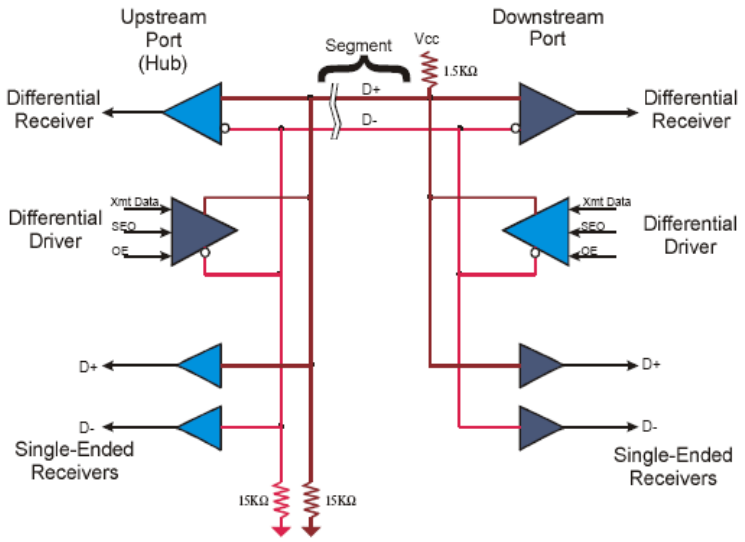
Signaling

The host can detect the connection of a new device and requested speed



Host interface

Differential and single-ended receivers

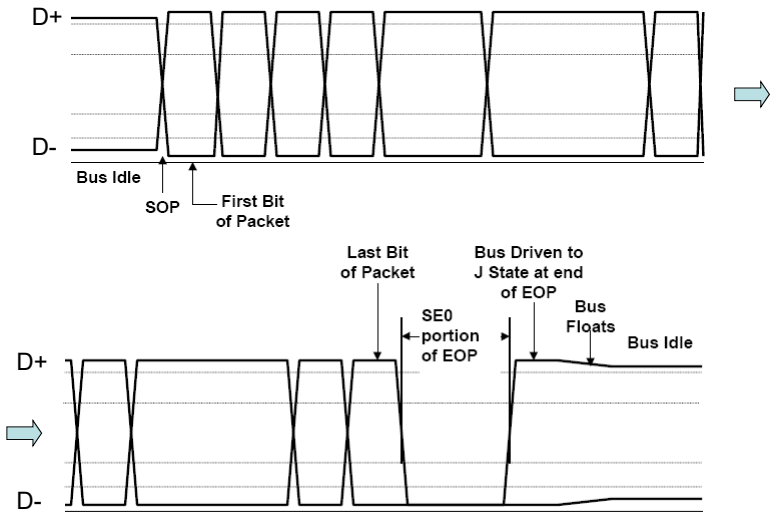


Electrical parameters of signaling

Bus State	Signaling Levels		
	At originating source connector (at end of bit time)	At final target connector	
		Required	Acceptable
Differential "1"	D+ > V_{OH} (min) and D- < V_{OL} (max)	(D+) - (D-) > 200mV and D+ > V_{IH} (min)	(D+) - (D-) > 200mV
Differential "0"	D- > V_{OH} (min) and D+ < V_{OL} (max)	(D-) - (D+) > 200mV and D- > V_{IH} (min)	(D-) - (D+) > 200mV
Single-ended 0 (SE0)	D+ and D- < V_{OL} (max)	D+ and D- < V_{IL} (max)	D+ and D- < V_{IH} (min)
Data J state:			
Low-speed	Differential "0"	Differential "0"	
Full-speed	Differential "1"	Differential "1"	
Data K state:			
Low-speed	Differential "1"	Differential "1"	
Full-speed	Differential "0"	Differential "0"	
Idle state:	N.A.		
Low-speed		D- > V_{IHZ} (min) and D+ < V_{IL} (max)	D- > V_{IHZ} (min) and D+ < V_{IH} (min)
Full-speed		D+ > V_{IHZ} (min) and D- < V_{IL} (max)	D+ > V_{IHZ} (min) and D- < V_{IH} (min)
Resume state	Data K state	Data K state	
Start-of-Packet (SOP)	Data lines switch from Idle to K state		
End-of-Packet (EOP)[4]	SE0 for approximately 2 bit times[1] followed by a J for 1 bit time[2]	SE0 for ≥ 1 bit time[3] followed by a J state for 1 bit time	SE0 for ≥ 1 bit time[3] followed by a J state
Disconnect (at downstream port)	N.A.	SE0 for ≥2.5μs	
Connect (at downstream port)	N.A.	Idle for ≥2ms	Idle for ≥2.5 μs
Reset	D+ and D- < V_{OL} (max) for ≥10ms	D+ and D- < V_{IL} (max) for ≥10ms	D+ and D- < V_{IL} (max) for ≥2.5 μs

Timing diagram of a USB transmission

(consider D+)

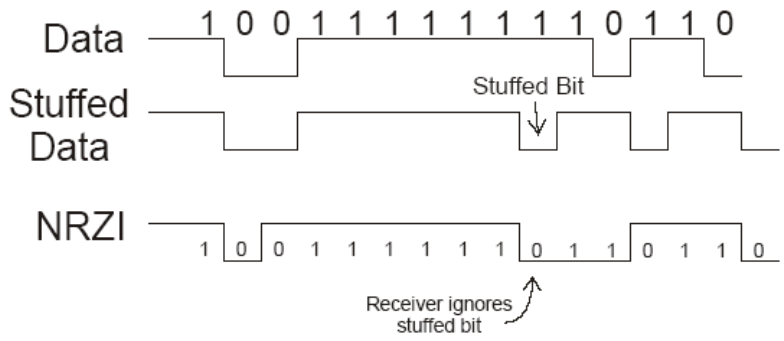


Data Codification

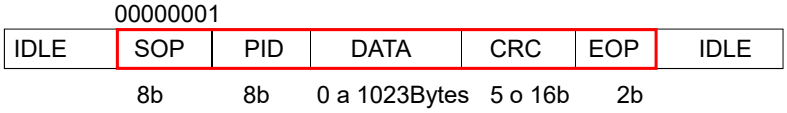
NRZi (non return to zero inverted) + Stuffed bit (extra 0 after the 6th 1 bit)

1 - hold level. 0 – switch level.

Changes are used to syncronize host and devices.



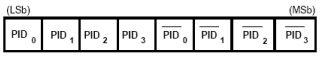
Packet Formats



PID Type	PID Name	PID[3:0]*	Description
Token	OUT	0001B	Address + endpoint number in host-to-function transaction
	IN	1001B	Address + endpoint number in function-to-host transaction
	SOF	0101B	Start-of-Frame marker and frame number
	SETUP	1101B	Address + endpoint number in host-to-function transaction for SETUP to a control pipe
Data	DATA0	0011B	Data packet PID even
	DATA1	1011B	Data packet PID odd
Handshake	ACK	0010B	Receiver accepts error-free data packet
	NAK	1010B	Rx device cannot accept data or Tx device cannot send data
	STALL	1110B	Endpoint is halted or a control pipe request is not supported.
Special	PRE	1100B	Host-issued preamble. Enables downstream bus traffic to low-speed devices.

*Note: PID bits are shown in MSb order. When sent on the USB, the rightmost bit (bit 0) will be sent first.

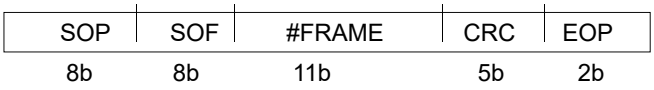
PID

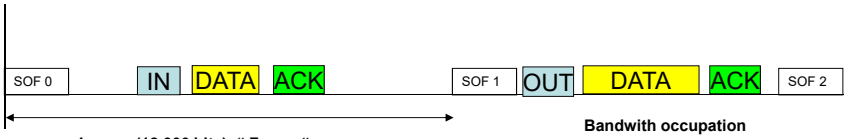


Examples of USB Packets (I)

Token Packets

SOF (Start Of a Frame)





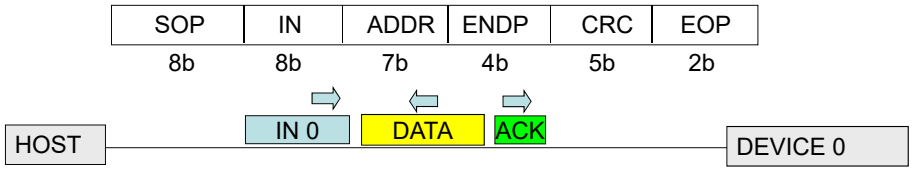
1 mseg (12.000 bits) " Frame "

Bandwith occupation

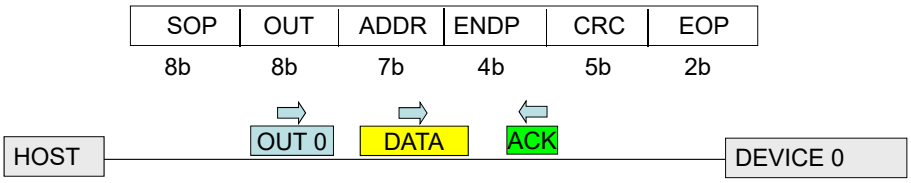
Data Payload (Bytes)	Percentage of Frame Bandwidth/ Transfer	Max Xfers/ Frame	Maximum Bandwidth
1	1%	150	150KB/s
2	1%	136	272KB/s
4	1%	115	460KB/s
8	1%	88	704KB/s
16	2%	60	960KB/s
32	3%	36	1.152MB/s
64	5%	20	1.280MB/s
128	9%	10	1.280MB/s
256	18%	5	1.280MB/s
512	36%	2	1.024MB/s
1023	69%	1	1.023MB/s

Examples of USB Packets (II)

IN (transmission allowed)



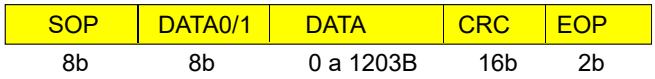
OUT (warns to be ready to accept data)



Examples of USB Packets (III)

Data Packets

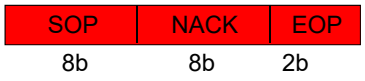
DATA (Data to be sent/received)



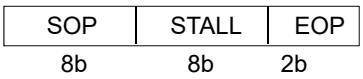
Handshake Packets

ACK (acknowledge)

NACK (Not acknowledge)

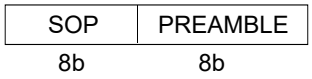


STALL



Special Packets

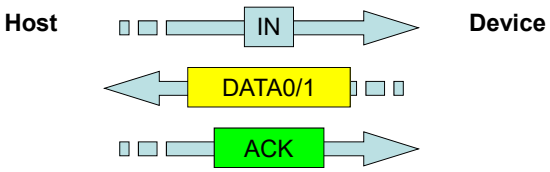
PREAMBLE (Warns LS op.)



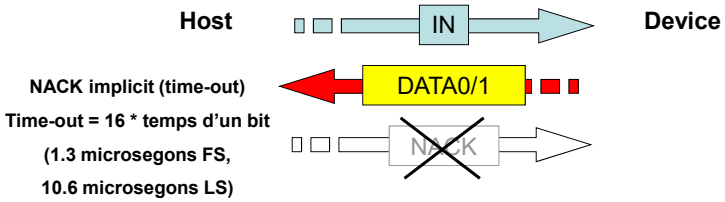
USB Transactions Examples (I)

IN

Without errors



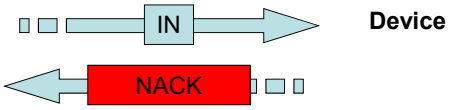
With errors



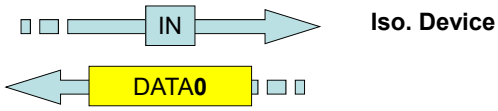
USB Transactions Examples (II)

IN

Device busy or without data



When addressing an isochronous endpoint

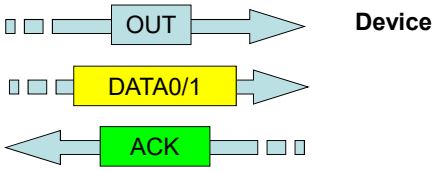


With or without error

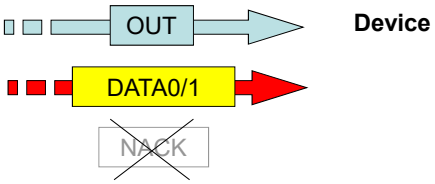
USB Transactions Examples (III)

OUT

Without errors



With errors

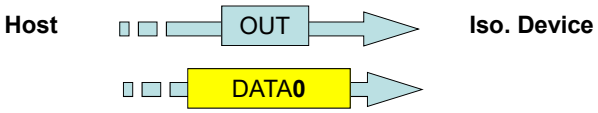


NACK implicit (time-out)

USB Transactions Examples (IV)

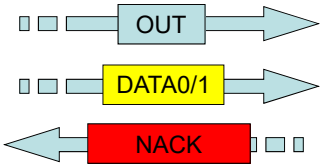
OUT

To an isochronous endpoint



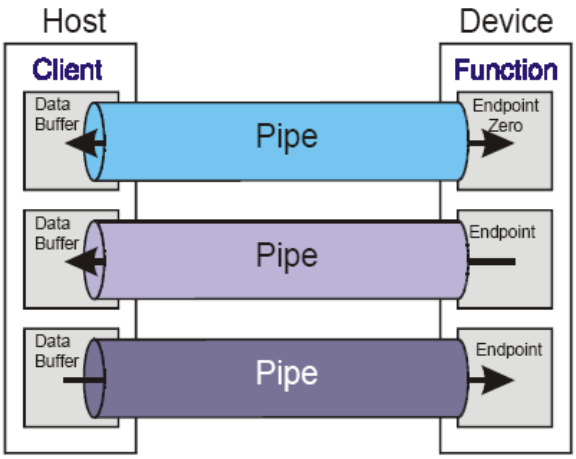
With or without errors

Device occupied or full buffer



Endpoints: concept

(logical channels for communications)



Endpoints = *Pipes* = *buffers* FIFO

Endpoints: Types of endpoints

Control: To send commands to the devices (small packets). Errors are controlled and retransmissions performed. Minimum bandwith guaranteed.

Interrupt: These endpoints are periodically polled (is a Master-Slave structure). Small packets (mouse, keyboard). Errors are controlled and retransmissions performed. Bandwith guaranteed.

Bulk: Used to transmit big amounts of data, with error control, retransmissions, but without time requirements (printer, scanner). Bandwith not guaranteed.

Isochronous: periodical transmission of data. Bandwith guaranteed. Errors are'nt controlled.

- Telephones
- High-Speed Modems
- Microphones/Headsets
- High-End Digital Speakers
- CD Audio Player
- DVD Players
- Video Conferencing Camera
- Digital Satellite Receivers

- A device can have 16 independent endpoints.
- Endpoint #0 is always a control endpoint.
- Endpoint #0 is bidirectional, endpoints # 1-15 can be configured as input or output.

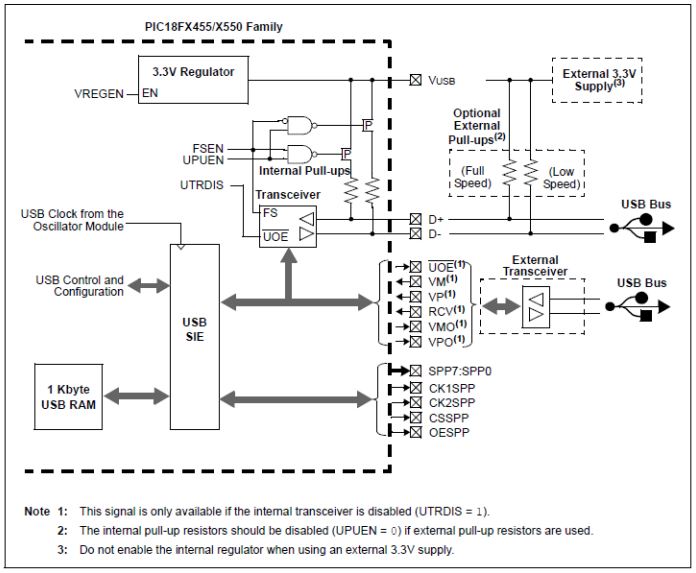
Endpoints: Bus occupancy

Transfer type	Max packet size per frame	Protocol overhead	Guaranteed bandwidth	Error recovery	Max speed
Control	64	45	10% BW is reserved	Yes	-
Bulk	64	13	No, but can use up to 95% if available	Yes	1.216MB/s
Interrupt	1-64	13	Yes, but cannot use more than 90% of BW combined.	Yes	**1.216MB/s**
Isochronous	1023	9		No	**1.280MB/s**

Table 2.1 Packet size and max speed for USB protocols

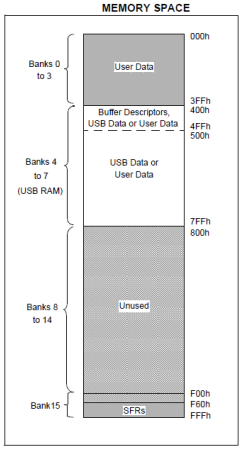
USB Module in PIC18F4550

FIGURE 17-1: USB PERIPHERAL AND OPTIONS



Note 1: This signal is only available if the internal transceiver is disabled (UTRDIS = 1).
2: The internal pull-up resistors should be disabled (UPUEN = 0) if external pull-up resistors are used.
3: Do not enable the internal regulator when using an external 3.3V supply.

FIGURE 17-5: IMPLEMENTATION OF USB RAM IN DATA MEMORY SPACE

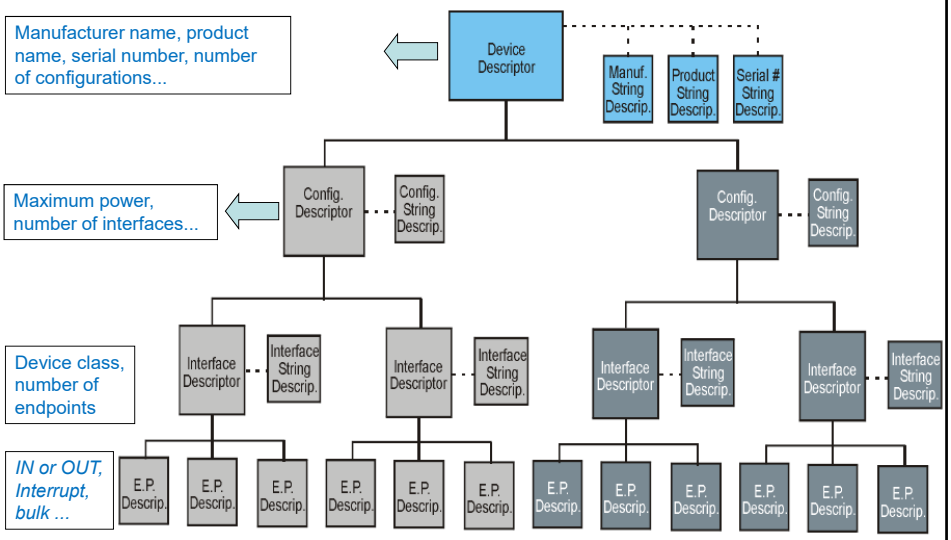


USB Setup Process

Device is identified for the first time in the bus

Request types	
GET_STATUS	0
CLEAR_FEATURE	1
Reserved for future use	2
SET_FEATURE	3
Reserved for future use	4
SET_ADDRESS	5
GET_DESCRIPTOR	6
SET_DESCRIPTOR	7
GET_CONFIGURATION	8
SET_CONFIGURATION	9
GET_INTERFACE	10
SET_INTERFACE	11
SYNCH_FRAME	12

USB device description structures



USB Description tables

8	*idVendor*	2	ID	Vendor ID (assigned by the USB)
10	*idProduct*	2	ID	Product ID (assigned by the manufacturer)
12	*bcdDevice*	2	BCD	Device release number in binary-coded decimal
14	*iManufacturer*	1	Index	Index of string descriptor describing manufacturer
15	*iProduct*	1	Index	Index of string descriptor describing product
16	*iSerialNumber*	1	Index	Index of string descriptor describing the device's serial number
17	*bNumConfigurations*	1	Number	Number of possible configurations

Device configuration description

Field	Size	Description
bLength	1	Descriptor size (bytes)
bDescriptorType	1	Type of descriptor (config descriptor = 02h)
wTotalLength	2	Size of all data returned, including interface and endpoint descriptors
bNumInterfaces	1	Number of interfaces supported
bConfigurationValue	1	Identifier for Get and Set configuration requests
iConfiguration	1	Index of string descriptor (optional)
bmAttributes	1	Self/bus power and remote wakeup settings
MaxPower	1	Bus power required (maximum mA/2)

Device interface descriptor

Offset	Field	Size	Description
0	bLength	1	Descriptor size (bytes)
1	bDescriptorType	1	Type of descriptor (interface descriptor = 04h)
2	bInterfaceNumber	1	Number identifying this interface (start at 0)
3	bAlternateSetting	1	Value used to select an alternate setting
4	bNumEndpoints	1	Number of endpoints supported, excluding Endpoint 0
5	bInterfaceClass	1	Class code
6	bInterfaceSubClass	1	Subclass code
7	bInterfaceProtocol	1	Protocol code
8	iInterface	1	Index of string descriptor for the interface

Endpoint descriptor

Offset	Field	Size	Description
0	bLength	1	Descriptor size (bytes)
1	bDescriptorType	1	Type of descriptor (endpoint descriptor = 05h)
2	bEndpointAddress	1	Endpoint number and direction
3	bmAttributes	1	Transfer type supported
4	wMaxPacketSize	2	Maximum packet size supported
6	bInterval	1	Polling interval, in milliseconds

Standard classes for USB devices

- Audio Class — Devices that are the source or sink of real-time audio information. This class is defined in four separate documents.
  - Audio Device Document 1.0
  - Audio Data Formats 1.0
  - Audio Terminal Types 1.0
  - USB MIDI (music instrument device interface) Devices 1.0
- Communications Device Class — Devices that attach to a telephone line (not local area networks). This class is defined in two documents:
  - Class Definitions for Communication Devices 1.1
  - Communications Device Class 1.0
- Content Security — This class defines transport mechanisms, descriptors, and USB requests to support a method for protecting the distribution of digital content via USB. The digital content being protected is usually copyrighted information. This class is defined in three documents:
  - Device Class Definition for Content Security Devices 1.0
  - Content Security Method 1 - Basic Authentication Protocol 1.0
  - Content Security Method 2 - USB Digital Transmission Content Protection Implementation 1.0
- Human Interface Device Class (HID) — Devices manipulated by end-users, and is defined in three documents:
  - Human Interface Devices 1.1
  - HID Usage Tables 1.1
  - HID Point of Sale Usage Tables 1.01

- Image Device Class — Devices that deal with still image capture.
  - Still Image Capture Device Definition 1.0 document.
- IrDA Class — This class defines an interface for infrared transceivers and is defined by the:
  - IrDA Bridge Device Definition 1.0 Document.
- Mass Storage Device Class — Devices used to store large amounts of information (for example, floppy drives, hard drives, and tape drives). This class is defined by four documents:
  - Mass Storage Overview 1.1
  - Mass Storage Bulk Only 1.0
  - Mass Storage Control/Bulk/Interrupt (CBI) Specification 1.0
  - Mass Storage UFI Command Specification 1.0
- Monitor Class — Defined to control monitor configuration and is specified in a single document.
  - Monitor Device Document 1.0.
- Physical Interface Device Class (PID) — Devices that provide tactile feedback to operator. Examples include: Joystick with variable resistance for simulating increased stick forces and turbulence. Split off from HID class. This class is specified in the:
  - Device Class Definition for PID 1.0 document.
- Power Device Class — Devices that provide power to system or to peripherals. Example devices include: Uninterruptable power supplies and smart batteries. Can be either stand alone device or integrated into the interface. The related document is the:
  - Power Device Class Document 1.0.
- Printer Device Class — Defines the descriptors, endpoints, and requests for printers. This class is specified by a single document.
  - Printer Device Class Document 1.1

Table 21-1: Audio Subclasses and Protocols

Subclass Code	Subclass Name	Protocol Code	Protocol Name
01h	8-bit Pulse Code Modulated (PCM) Audio Data	01h 02h	Mono Stereo
02h	16-bit PCM Audio Data	01h 02h 03h 04h	Mono Stereo Quadro Stereo & Stereo
03h	16-bit Dolby Surround Data	02h	Stereo
04h	IEC958 Audio Encoded Data	05h 06h	IEC958 Consumer IEC958 Professional
05h	MPEG1 Audio Encoded Data	07h 08h	Layer 1 Layer 2
06h	AC3 Audio Encoded Data	TBD	TBD

Table 21-2: Telephony Protocol Types and Codes Used by Telephony Devices

Protocol Code	Description	Related Reference Document
00h	Not defined	NA
01h	Common AT commands (Hayes compatible)	V.225ter
02h	Alternative PSTN modem command set	V.25bis
04h	Serial ISDN Terminal Adapter Control	V.120
08h	In-Band DCE control	V.ib
10h	ISDN TA control	Q.931
20h	Reserved	NA
40h	Other standard DCE control protocol not defined by the audio class specification. The control protocol used for this device is defined in a string descriptor.	NA
80h	Manufacturer-proprietary DCE control protocol is used. The protocol used is described in a string descriptor.	NA

Table 21-3: Display Class Standard Device Descriptor Definition

Offset	Field	Size (bytes)	Value	Description
4	DeviceClass	1	Class	Class code = 04h for display class
5	DeviceSubClass	1	SubClass	Subclass code: • 01h = CRT • 02h = Flat Panel Display • 03h = 3-D Display

Table 21-4: Mass Storage Class Code and Subclass Code

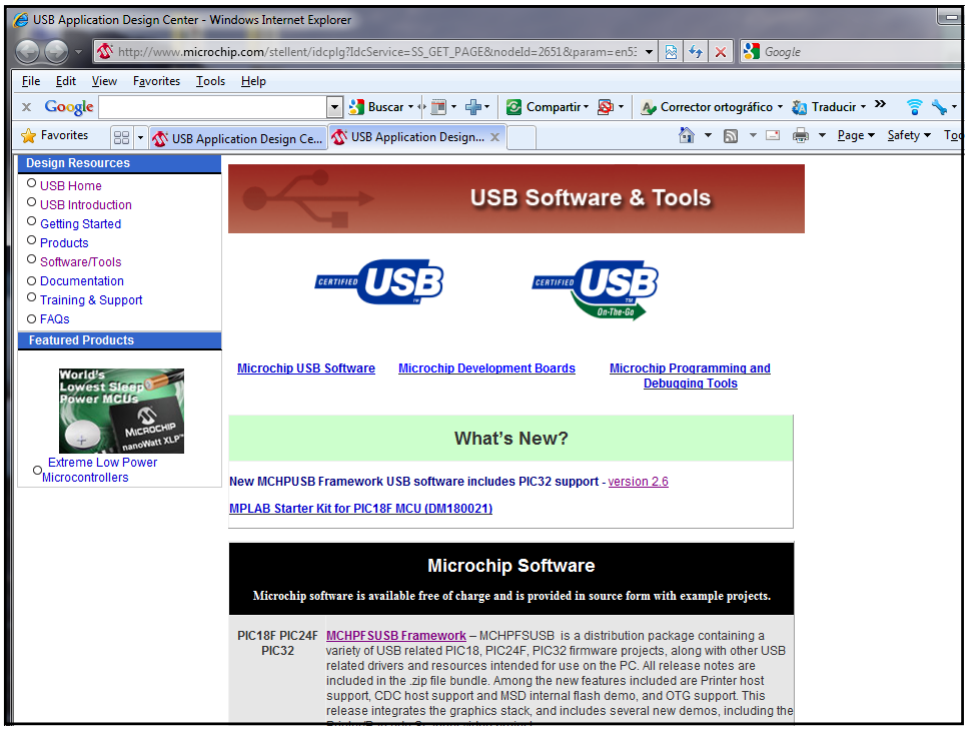
Offset	Field	Size (bytes)	Value	Description
5	InterfaceClass	1	Class	Class code for mass storage device = 01h.
6	InterfaceSubClass	1	SubClass	SubClass code for mass storage devices defined as: 01h = General Mass Storage 02h = CD-ROM 03h = Tape 04h = Solid State 05 - FEh Reserved

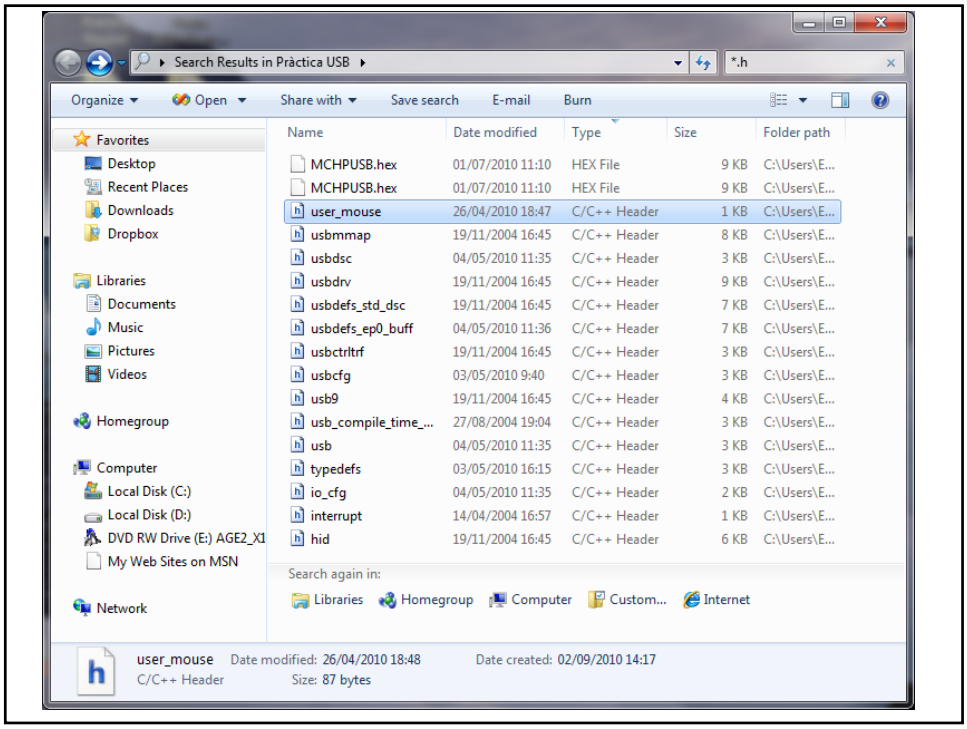
- General Mass Storage — ANSI X3.131, *Small Computer Systems Interface-2*
- CD-ROM — SFF-8020i, *ATA Packet Interface for CD-ROMs*
- Tape — QIC-157, *ATA Packet Interface for Tape*
- Solid State — *QIC-157, ATA Packet Interface for Tape* and SCSI command set (with modifications)

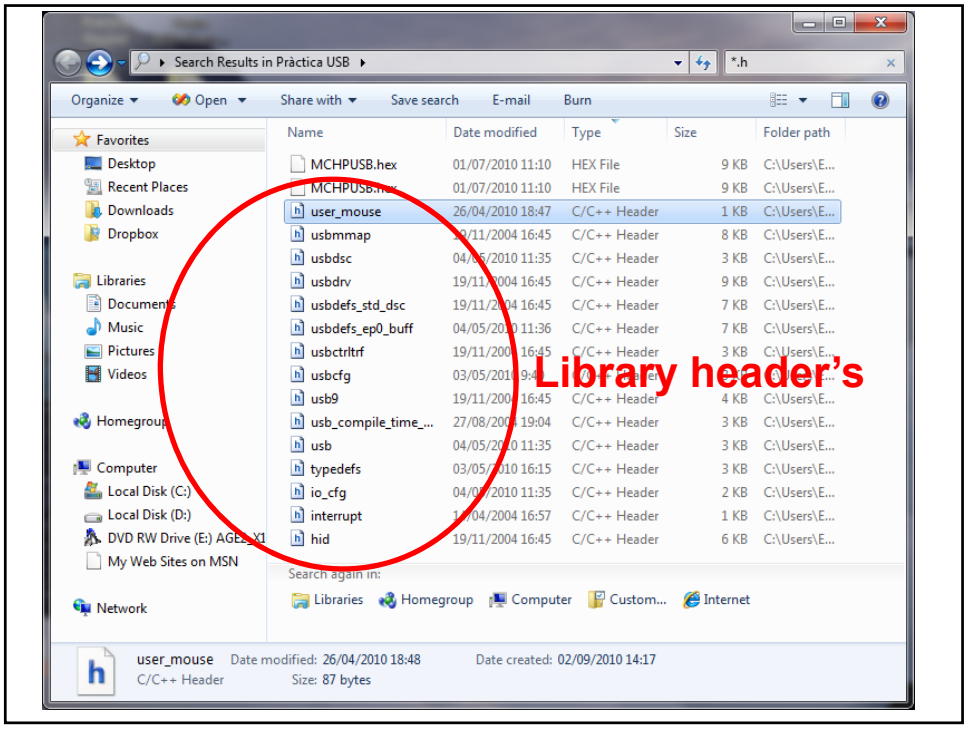
USB Universal Serial Bus

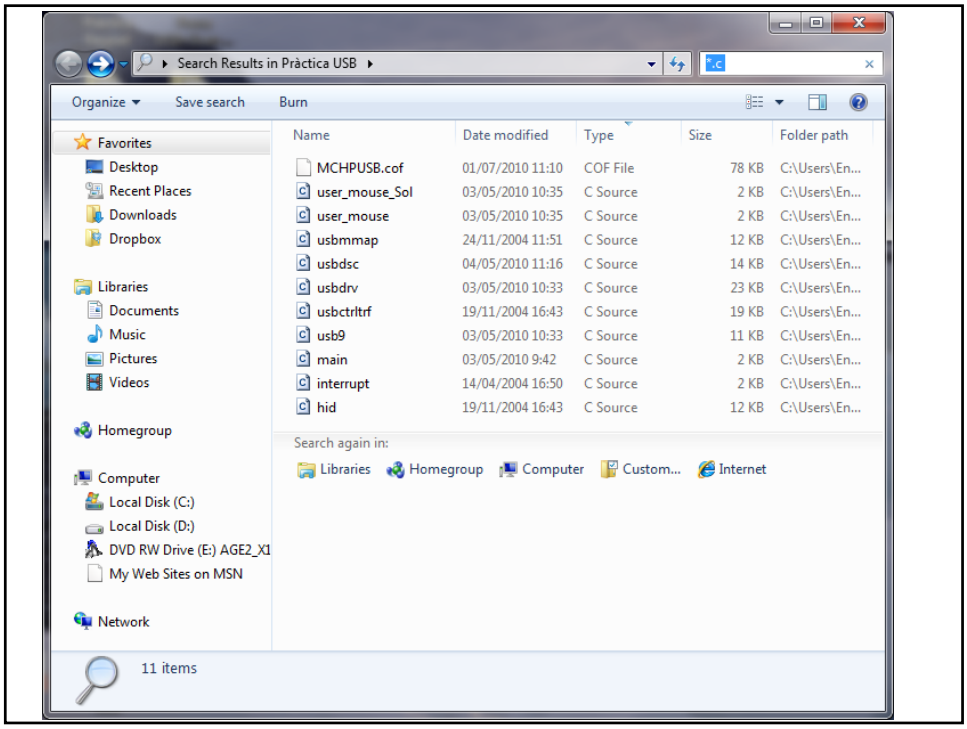
3. IMPLEMENTATION ON A MICROCOMPUTER

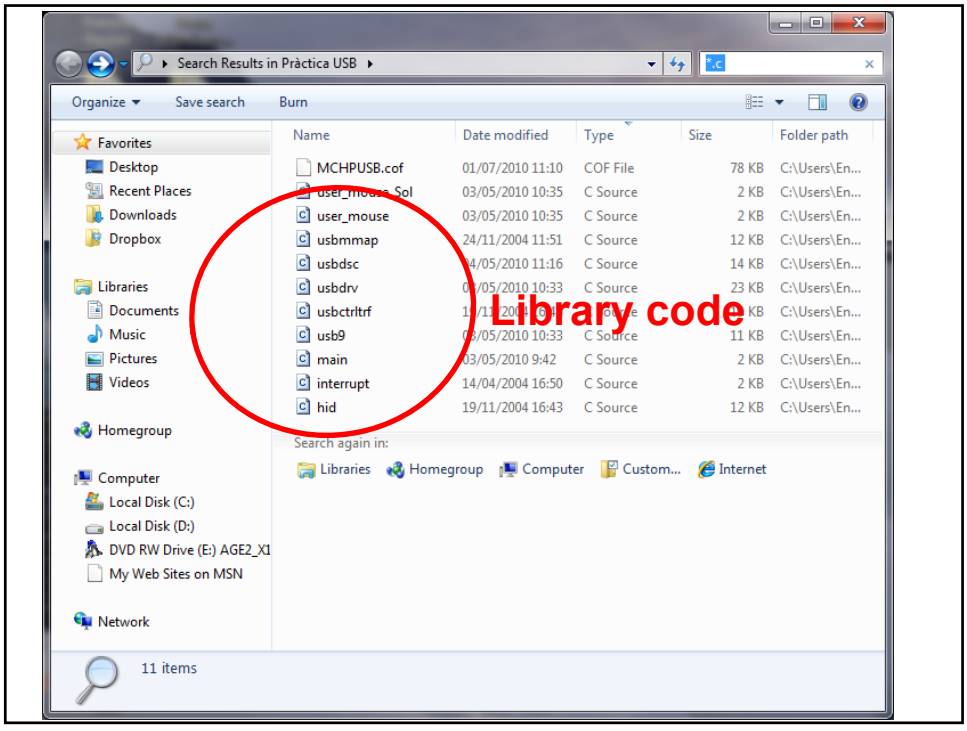

USB Application Design Center - Windows Internet Explorer
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1486
File Edit View Favorites Tools Help
pic usb driver windows7
Favorites
USB Application Design...
USB Application Design Ce...
MICROCHIP
myMicrochip Login | English | Chinese
Search
Home Products Design Support Applications Buy/Sample Corporate What's New
Home ▸ Design Resources Main Page ▸ Design Centers Home Page ▸
Design Resources
USB Home
USB Introduction
Getting Started
Products
Software/Tools
Documentation
Training & Support
FAQs
Featured Products
Extreme Low Power Microcontrollers
USB
Integrated
32-bit
16-bit
8-bit
Low-cost
Fast-development
Easy-to-use
Scaleable
Microchip Technology announces new families of USB capable microcontrollers covering 8-, 16-, and 32-bit architectures.
New to USB or curious how USB can help you? Start Here
USB Introduction
Products
Software & Tools


USB Application Design Center - Windows Internet Explorer
http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=2651&param=en5
File Edit View Favorites Tools Help
Favorites
USB Application Design Ce...
USB Application Design...
Design Resources
USB Home
USB Introduction
Getting Started
Products
Software/Tools
Documentation
Training & Support
FAQs
Featured Products
Extreme Low Power Microcontrollers
USB Software & Tools
Microchip USB Software
Microchip Development Boards
Microchip Programming and Debugging Tools
What's New?
New MCHPUSB Framework USB software includes PIC32 support - version 2.6
MPLAB Starter Kit for PIC18F MCU (DM180021)
Microchip Software
Microchip software is available free of charge and is provided in source form with example projects.
PIC18F PIC24F PIC32
MCHPFSUSB Framework – MCHPFSUSB is a distribution package containing a variety of USB related PIC18, PIC24F, PIC32 firmware projects, along with other USB related drivers and resources intended for use on the PC. All release notes are included in the .zip file bundle. Among the new features included are Printer host support, CDC host support and MSD internal flash demo, and OTG support. This release integrates the graphics stack, and includes several new demos, including the

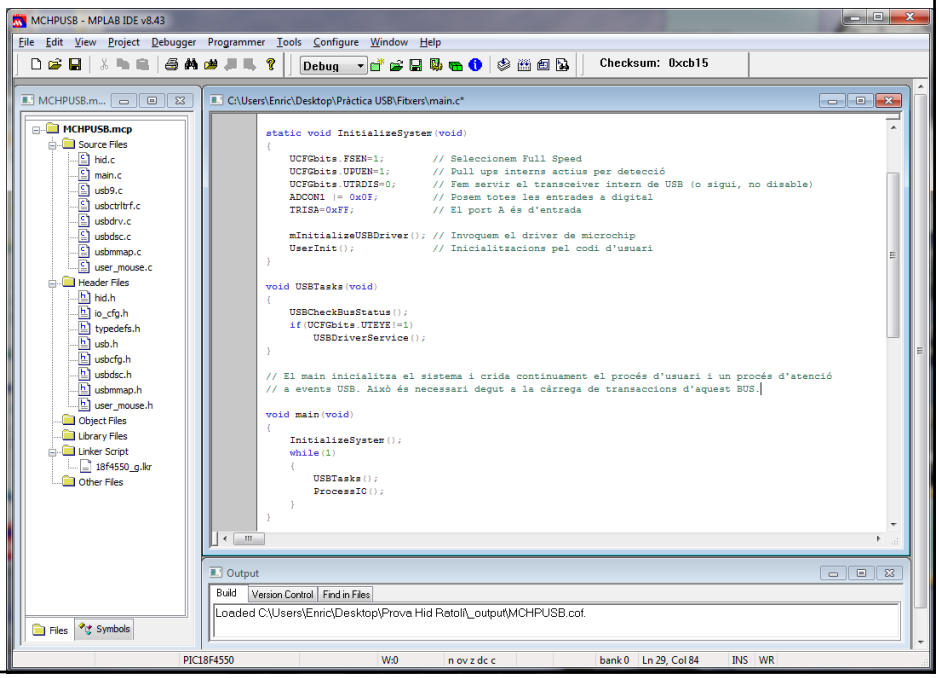

Search Results in Pràctica USB
*.h
Organize
Open
Share with
Save search
E-mail
Burn
Favorites
Desktop
Recent Places
Downloads
Dropbox
Libraries
Documents
Music
Pictures
Videos
Homegroup
Computer
Local Disk (C:)
Local Disk (D:)
DVD RW Drive (E:) AGE2_X1
My Web Sites on MSN
Network
Name
Date modified
Type
Size
Folder path
MCHPUSB.hex 01/07/2010 11:10 HEX File 9 KB
MCHPUSB.hex 01/07/2010 11:10 HEX File 9 KB
user_mouse 26/04/2010 18:47 C/C++ Header 1 KB
usbmmap 19/11/2004 16:45 C/C++ Header 8 KB
usbdsc 04/05/2010 11:35 C/C++ Header 3 KB
usbdrv 19/11/2004 16:45 C/C++ Header 9 KB
usbdefs_std_dsc 19/11/2004 16:45 C/C++ Header 7 KB
usbdefs_ep0_buff 04/05/2010 11:36 C/C++ Header 7 KB
usbctrltrf 19/11/2004 16:45 C/C++ Header 3 KB
usbcfg 03/05/2010 9:40 C/C++ Header 3 KB
usb9 19/11/2004 16:45 C/C++ Header 4 KB
usb_compile_time_... 27/08/2004 19:04 C/C++ Header 3 KB
usb 04/05/2010 11:35 C/C++ Header 3 KB
typedefs 03/05/2010 16:15 C/C++ Header 3 KB
io_cfg 04/05/2010 11:35 C/C++ Header 2 KB
interrupt 14/04/2004 16:57 C/C++ Header 1 KB
hid 19/11/2004 16:45 C/C++ Header 6 KB
Search again in:
Libraries
Homegroup
Computer
Custom...
Internet
user_mouse Date modified: 26/04/2010 18:48 Date created: 02/09/2010 14:17
C/C++ Header Size: 87 bytes


Search Results in Pràctica USB
*.h
Organize
Open
Share with
Save search
E-mail
Burn
Favorites
Desktop
Recent Places
Downloads
Dropbox
Libraries
Documents
Music
Pictures
Videos
Homegroup
Computer
Local Disk (C:)
Local Disk (D:)
My Web Sites on MSN
Network
Name
Date modified
Type
Size
Folder path
MCHPUSB.hex 01/07/2010 11:10 HEX File 9 KB
user_mouse 26/04/2010 18:47 C/C++ Header 1 KB
usbmmap C/C++ Header 8 KB
usbdsc C/C++ Header 3 KB
usbdrv C/C++ Header 9 KB
usbdefs_std_dsc C/C++ Header 7 KB
usbdefs_ep0_buff C/C++ Header 7 KB
usbctrltrf C/C++ Header 3 KB
usbcfg C/C++ Header
usb9 C/C++ Header 4 KB
usb_compile_time_... C/C++ Header 3 KB
usb C/C++ Header 3 KB
typedefs C/C++ Header 3 KB
io_cfg C/C++ Header 2 KB
interrupt C/C++ Header 1 KB
hid 19/11/2004 16:45 C/C++ Header 6 KB
Search again in:
Libraries
Homegroup
Computer
Custom...
Internet
user_mouse Date modified: 26/04/2010 18:48 Date created: 02/09/2010 14:17
C/C++ Header Size: 87 bytes
Library header's


Search Results in Pràctica USB
Organize
Save search
Burn
Favorites
Desktop
Recent Places
Downloads
Dropbox
Libraries
Documents
Music
Pictures
Videos
Homegroup
Computer
Local Disk (C:)
Local Disk (D:)
DVD RW Drive (E:) AGE2_X1
My Web Sites on MSN
Network
Name
Date modified
Type
Size
Folder path
MCHPUSB.cof 01/07/2010 11:10 COF File 78 KB C:\Users\En...
user_mouse_Sol 03/05/2010 10:35 C Source 2 KB C:\Users\En...
user_mouse 03/05/2010 10:35 C Source 2 KB C:\Users\En...
usbmmap 24/11/2004 11:51 C Source 12 KB C:\Users\En...
usbdsc 04/05/2010 11:16 C Source 14 KB C:\Users\En...
usbdrv 03/05/2010 10:33 C Source 23 KB C:\Users\En...
usbctrltrf 19/11/2004 16:43 C Source 19 KB C:\Users\En...
usb9 03/05/2010 10:33 C Source 11 KB C:\Users\En...
main 03/05/2010 9:42 C Source 2 KB C:\Users\En...
interrupt 14/04/2004 16:50 C Source 2 KB C:\Users\En...
hid 19/11/2004 16:43 C Source 12 KB C:\Users\En...
Search again in:
Libraries
Homegroup
Computer
Custom...
Internet
11 items


Search Results in Pràctica USB
Organize
Save search
Burn
Favorites
Desktop
Recent Places
Downloads
Dropbox
Libraries
Documents
Music
Pictures
Videos
Homegroup
Computer
Local Disk (C:)
Local Disk (D:)
DVD RW Drive (E:) AGE2_X1
My Web Sites on MSN
Network
Name
Date modified
Type
Size
Folder path
MCHPUSB.cof 01/07/2010 11:10 COF File 78 KB C:\Users\En...
user_mouse_Sol 03/05/2010 10:35 C Source 2 KB C:\Users\En...
user_mouse 03/05/2010 10:35 C Source 2 KB C:\Users\En...
usbmmap 24/11/2004 11:51 C Source 12 KB C:\Users\En...
usbdsc 04/05/2010 11:16 C Source 14 KB C:\Users\En...
usbdrv 03/05/2010 10:33 C Source 23 KB C:\Users\En...
usbctrltrf 19/11/2004 16:43 C Source 19 KB C:\Users\En...
usb9 03/05/2010 10:33 C Source 11 KB C:\Users\En...
main 03/05/2010 9:42 C Source 2 KB C:\Users\En...
interrupt 14/04/2004 16:50 C Source 2 KB C:\Users\En...
hid 19/11/2004 16:43 C Source 12 KB C:\Users\En...
Search again in:
Libraries
Homegroup
Computer
Custom...
Internet
11 items
Library code

Project structure



Special program structure

```
static void InitializeSystem(void)
{
    UCFGbits.FSEN=1;        // Seleccionem Full Speed
    UCFGbits.UPUEN=1;       // Pull ups interns actius per detecció
    UCFGbits.UTRDIS=0;      // Fem servir el transceiver intern de USB (o sigui, no disable)
    ADCON1 |= 0x0F;         // Posem totes les entrades a digital
    TRISA=0xFF;             // El port A és d'entrada
    mInitializeUSBDriver(); // Invoquem el driver de microchip
    UserInit();             // Inicialitzacions pel codi d'usuari
}
void USBTasks(void)
{
    USBCheckBusStatus();
    if(UCFGbits.UTEYE!=1)
        USBDriverService();
}
// El main inicialitza el sistema i crida continuament el procés d'usuari i un procés d'atenció
// a events USB. Això és necessari degut a la càrrega de transaccions d'aquest BUS.

void main(void)
{
    InitializeSystem();
    while(1)
    {
        USBTasks();
        ProcessIO();
    }
}
```

Special program structure

```
static void InitializeSystem(void)
{
    UCFGbits.FSEN=1;        // Seleccionem Full Speed
    UCFGbits.UPUEN=1;       // Pull ups interns actius per detecció
    UCFGbits.UTRDIS=0;      // Fem servir el transceiver intern de USB (o sigui, no disable)
    ADCON1 |= 0x0F;         // Posem totes les entrades a digital
    TRISA=0xFF;             // El port A és d'entrada
    mInitializeUSBDriver(); // Invoquem el driver de microchip
    UserInit();             // Inicialitzacions pel codi d'usuari
}
void USBTasks(void)
{
    USBCheckBusStatus();
    if(UCFGbits.UTEYE!=1)
        USBDriverService();
}
// El main inicialitza el sistema i crida continuament el procés d'usuari i un procés d'atenció
// a events USB. Això és necessari degut a la càrrega de transaccions d'aquest BUS.

void main(void)
{
    InitializeSystem();
    while(1)
    {
        USBTasks();
        ProcessIO();
    }
}
```

CPU is shared between the USB Tasks and user tasks

Special program structure

```
static void InitializeSystem(void)
{
    UCFGbits.FSEN=1;        // Seleccionem Full Speed
    UCFGbits.UPUEN=1;       // Pull ups interns actius per detecció
    UCFGbits.UTRDIS=0;      // Fem servir el transceiver intern de USB (o sigui, no disable)
    ADCON1 |= 0x0F;         // Posem totes les entrades a digital
    TRISA=0xFF;             // El port A és d'entrada
    mInitializeUSBDriver(); // Invoquem el driver de microchip
    UserInit();             // Inicialitzacions pel codi d'usuari
}
void USBTasks(void)
{
    USBCheckBusStatus();
    if(UCFGbits.UTEYE!=1)
        USBDriverService();
}
// El main inicialitza el sistema i crida continuament el procés d'usuari i un procés d'atenció
// a events USB. Això és necessari degut a la càrrega de transaccions d'aquest BUS.

void main(void)
{
    InitializeSystem();
    while(1)
    {
        USBTasks();
        ProcessIO();
    }
}
```

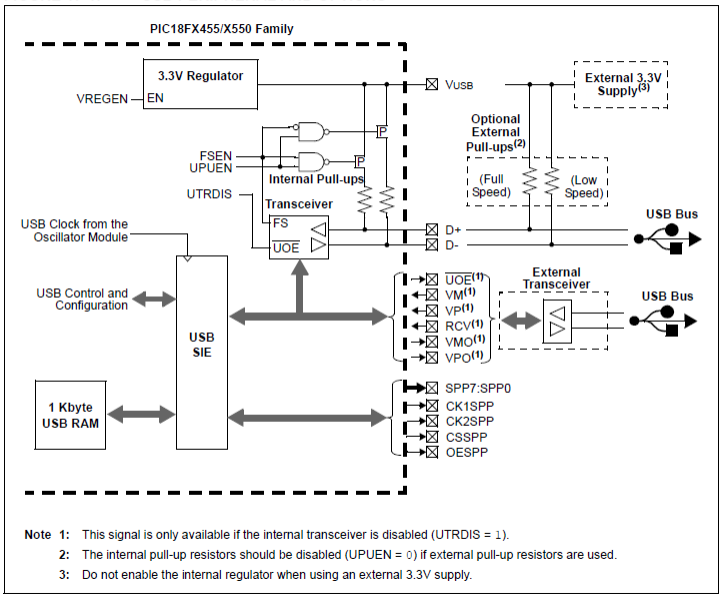
User can program an inicialization function and a non-blocking I/O process function

Special program structure

```
void ProcessIO (void) {

If ((usb_device_state < CONFIGURED_STATE)    ||
(UCONbits.SUSPND==1)) return;

buffer[0] = buffer[1] = buffer[2] = 0;

If (PORTAbits.RA0==1)    {
        buffer[0] = 0;   // Botons
        buffer[1] = 0;   // X-Vector
        buffer[2] = -2;  // Y-Vector };

If (PORTAbits.RA1==1)    {
        buffer[0] = 0;   // Botons
        buffer[1] = -2;  // X-Vector
        buffer[2] = 0;   // Y-Vector };
….

If (!mHIDTxIsBusy())    {
      HIDTxReport(buffer,3); };   // Send Buffer if Tx ready

}
```

USB Configuration

FIGURE 17-1: USB PERIPHERAL AND OPTIONS



Note 1: This signal is only available if the internal transceiver is disabled (UTRDIS = 1).
2: The internal pull-up resistors should be disabled (UPUEN = 0) if external pull-up resistors are used.
3: Do not enable the internal regulator when using an external 3.3V supply.

USB Configuration

```
static void InitializeSystem(void)
{
    UCFGbits.FSEN=1;            // Seleccionem Full Speed
    UCFGbits.UPUEN=1;           // Pull ups interns actius per detecció
    UCFGbits.UTRDIS=0;          // Fem servir el transceiver intern de USB (o sigui, no disable)
    ADCON1 |= 0x0F;             // Posem totes les entrades a digital
    TRISA=0xFF;                 // El port A és d'entrada
    mInitializeUSBDriver();     // Invoquem el driver de microchip
    UserInit();                 // Inicialitzacions pel codi d'usuari
}
void USBTasks(void)
{
    USBCheckBusStatus();
    if(UCFGbits.UTEYE!=1)
        USBDriverService();
}
// El main inicialitza el sistema i crida continuament el procés d'usuari i un procés d'atenció
// a events USB. Això és necessari degut a la càrrega de transaccions d'aquest BUS.

void main(void)
{
    InitializeSystem();
    while(1)
    {
        USBTasks();
        ProcessIO();
    }
}
```

Hardware interface configuration

USB Device Configuration

usbdsc.c

```
/** I N C L U D E S **********************************************/
#include "system\typedefs.h"
#include "system\usb\usb.h"

/** C O N S T A N T S ********************************************/
#pragma romdata

/* Device Descriptor */
rom USB_DEV_DSC device_dsc=
{
    sizeof(USB_DEV_DSC),    // Size of this descriptor in bytes
    DSC_DEV,                // DEVICE descriptor type
    0x0200,                 // USB Spec Release Number in BCD format
    0x00,                   // Class Code
    0x00,                   // Subclass code
    0x00,                   // Protocol code
    EP0_BUFF_SIZE,          // Max packet size for EP0, see usbcfg.h
    0x04D8,                 // Vendor ID
    0x0000,                 // Product ID
    0x0001,                 // Device release number in BCD format
    0x01,                   // Manufacturer string index
    0x02,                   // Product string index
    0x00,                   // Device serial number string index
    0x01                    // Number of possible configurations
};

/* Configuration 1 Descriptor */
CFG01={
    /* Configuration Descriptor */
    sizeof(USB_CFG_DSC),    // Size of this descriptor in bytes
    DSC_CFG,                // CONFIGURATION descriptor type
    sizeof(cfg01),          // Total length of data for this cfg
    1,                      // Number of interfaces in this cfg
    1,                      // Index value of this configuration
    0,                      // Configuration string index
    _DEFAULT|_RWU,          // Attributes, see usbdefs_std_dsc.h
```

Device descriptor

USB Device Configuration

usbdsc.c

Configuration

```
/* Configuration 1 Descriptor */
CFG01={
    /* Configuration Descriptor */
    sizeof(USB_CFG_DSC),    // Size of this descriptor in bytes
    DSC_CFG,                // CONFIGURATION descriptor type
    sizeof(cfg01),          // Total length of data for this cfg
    1,                      // Number of interfaces in this cfg
    1,                      // Index value of this configuration
    0,                      // Configuration string index
    _DEFAULT|_RWU,          // Attributes, see usbdefs_std_dsc.h
    50,                     // Max power consumption (2X mA)

    /* Interface Descriptor */
    sizeof(USB_INTF_DSC),   // Size of this descriptor in bytes
    DSC_INTF,               // INTERFACE descriptor type
    0,                      // Interface Number
    0,                      // Alternate Setting Number
    1,                      // Number of endpoints in this intf
    HID_INTF,               // Class code
    BOOT_INTF_SUBCLASS,     // Subclass code
    HID_PROTOCOL_MOUSE,     // Protocol code
    0,                      // Interface string index

    /* HID Class-Specific Descriptor */
    sizeof(USB_HID_DSC),    // Size of this descriptor in bytes
    DSC_HID,                // HID descriptor type
    0x0101,                 // HID Spec Release Number in BCD format
    0x00,                   // Country Code (0x00 for Not supported)
    HID_NUM_OF_DSC,         // Number of class descriptors, see usbcfg.h
    DSC_RPT,                // Report descriptor type
    sizeof(hid_rpt01),      // Size of the report descriptor

    /* Endpoint Descriptor */
    sizeof(USB_EP_DSC),DSC_EP, EP01_IN, INT,HID_INT_IN_EP_SIZE,0x0A
```

USB Device Configuration

usbdsc.c

```
rom struct{byte bLength;byte bDscType;word string[25];}sd001={
sizeof(sd001),DSC_STR,
'M','i','c','r','o','c','h','i','p',' ',
'T','e','c','h','n','o','l','o','g','y',' ','I','n','c','.'};

rom struct{byte bLength;byte bDscType;word string[22];}sd002={
sizeof(sd002),DSC_STR,
'M','o','u','s','e',' ','D','e','m','o',' ',
'P','r','a','c','t','i','c','a','l',' ','C','I'};

rom struct{byte report[HID_RPT01_SIZE];}hid_rpt01={
    0x05, 0x01, /* Usage Page (Generic Desktop)          */
    0x09, 0x02, /* Usage (Mouse)                         */
    0xA1, 0x01, /* Collection (Application)              */
    0x09, 0x01, /*  Usage (Pointer)                      */
    0xA1, 0x00, /*  Collection (Physical)                */
    0x05, 0x09, /*      Usage Page (Buttons)             */
    0x19, 0x01, /*      Usage Minimum (01)               */
    0x29, 0x03, /*      Usage Maximum (03)               */
    0x15, 0x00, /*      Logical Minimum (0)              */
    0x25, 0x01, /*      Logical Maximum (0)              */
    0x95, 0x03, /*      Report Count (3)                 */
    0x75, 0x01, /*      Report Size (1)                  */
    0x81, 0x02, /*      Input (Data, Variable, Absolute) */
    0x95, 0x01, /*      Report Count (1)                 */
    0x75, 0x05, /*      Report Size (5)                  */
    0x81, 0x01, /*      Input (Constant)    ;5 bit padding */
    0x05, 0x01, /*      Usage Page (Generic Desktop)     */
    0x09, 0x30, /*      Usage (X)                        */
    0x09, 0x31, /*      Usage (Y)                        */
    0x15, 0x81, /*      Logical Minimum (-127)           */
    0x25, 0x7F, /*      Logical Maximum (127)            */
    0x75, 0x08, /*      Report Size (8)                  */
    0x95, 0x02, /*      Report Count (2)                 */
    0x81, 0x06, /*      Input (Data, Variable, Relative) */
    0xC0, 0xC0};/* End Collection,End Collection        */
```

ID's for the operating system

Microcomputer USB Connection to the PC

Bus power: The USB powers the microcomputer (5V, 0.5A) so the micro's software assumes that it is connected to the PC.

Self power: The microcomputer is self powered and the bus power is used to detect the connection between the two devices (attach sense)

Dual power: the microcomputer is powered either by the USB or by its own source of power.

FIGURE 17-10: BUS POWER ONLY

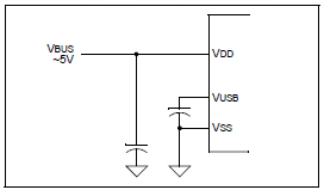


FIGURE 17-11: SELF-POWER ONLY

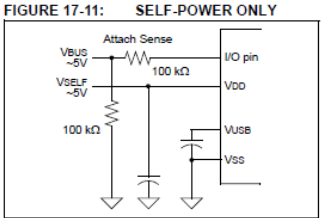
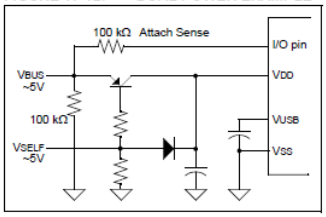
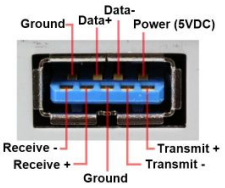


FIGURE 17-12: DUAL POWER EXAMPLE



USB 3.0



- **Full- duplex communication:** Transfer rates of 4.8 Gbps — 5 extra wires: SuperSpeed Transmit (SSTX+, SSTX–), SuperSpeed Receive (SSRX+, SSRX–), and an additional ground (GND).
- **Power consumption:** USB 2.0 provides up to 500 mA whereas USB 3.0 provides up to 900 mA.
- **Point-to-Point Routing:** Packets originated in the host contain a extra 20-bit "route string" field. Allows better power efficiency with less power for idle states.
- **Backward compatibility:** USB 2.0 devices can be plugged in a USB 3.0 hub.
- **Improved bus utilization:** A new feature was added (using packets NRDY and ERDY) to let a device asynchronously notify the host of its readiness.