Digital microsystems design

2019

Scop

- Understanding the USB interface and communication protocol

Content

- USB interface
  - Introduction
  - Implementation
  - Protocol
  - Descriptors
  - Classes of devices

Introduction

- Problems of general I/O connected to CPU
  - The use processor resources
    - I/O addresses
    - Interrupts lines
    - DMA channels
  - I/O devices detection and configuration at runtime
    - Plug-and-Play
  - Cables and connectors

Introduction

- Challenges for USB standard
  - One type of connector
  - Possibility to attach multiple devices
  - Resources allocation and conflicts management
  - Plug-and-Play and automatic detection and configuration

Introduction

- Requirements for modern I/O interfaces
  - Low-cost
  - Speed (transfer rates)
  - Extensibility
  - Compatibility
  - Low power

Introduction

- USB interface characteristics
  - Low cost
  - Unique standard for cables and connectors
  - Resource allocation
  - Hot Plug-and-Play

Introduction

- USB interface characteristics
  - Allows to connect up to 127 devices
  - Multiple functional devices
  - Transfer rates
    - Low-speed/low-cost devices (1.5 Mbps) – USB 1
    - Full-speed devices (12 Mbps) – USB 1
    - High-speed devices (480 Mbps) – USB 2
    - Super-speed devices (5 Gbps/10 Gbps) – USB 3/3.1

Introduction

PERFORMANCE	APPLICATIONS	ATTRIBUTES
LOW-SPEED • Interactive Devices • 10 – 100 kb/s	Keyboard, Mouse Stylus Game Peripherals Virtual Reality Peripherals	Lowest Cost Ease-of-Use Dynamic Attach-Detach Multiple Peripherals
FULL-SPEED • Phone, Audio, Compressed Video • 500 kb/s – 10 Mb/s	POTS Broadband Audio Microphone	Lower Cost Ease-of-Use Dynamic Attach-Detach Multiple Peripherals Guaranteed Bandwidth Guaranteed Latency
HIGH-SPEED • Video, Storage • 25 – 400 Mb/s	Video Storage Imaging Broadband	Low Cost Ease-of-Use Dynamic Attach-Detach Multiple Peripherals Guaranteed Bandwidth Guaranteed Latency High Bandwidth

Introduction

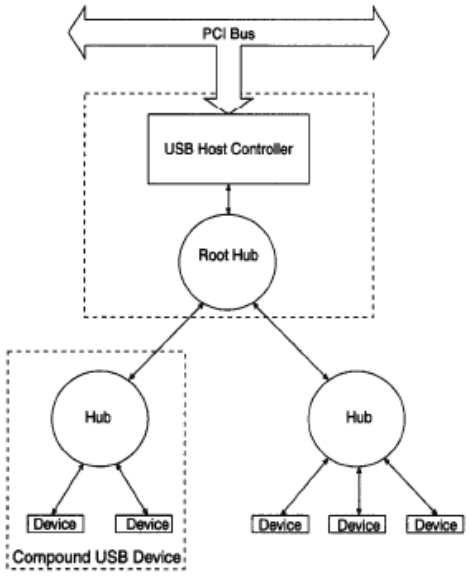
- USB interface characteristics
  - Flexible communication protocol
  - Simple interface for I/O peripherals
  - Error detection and correction
  - USB power supply (5V, 100-500 mA)
  - Power management

Implementation

- USB model
  - Two classes of USB devices:
    - USB hubs
      - Interconnection devices
      - Multiple port where multiple USB devices can be connected
      - Hubs can be interconnected
      - Maximum 5 levels
    - USB functions
      - USB peripheral devices
      - Perform specific operations called functions

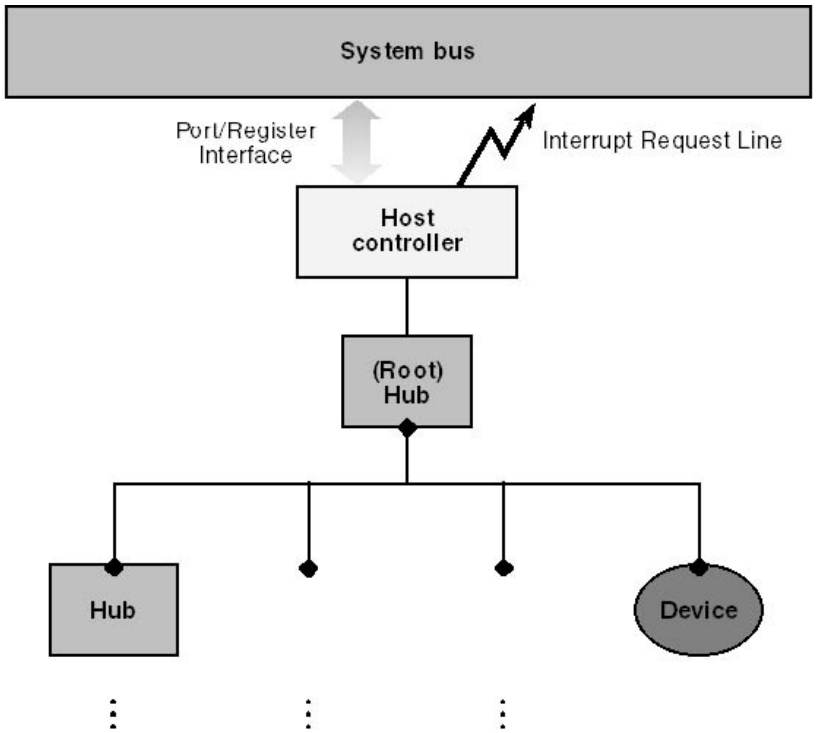
Implementation

- USB model
  - Tree based physical interconnecting topology
  - Root hub
    - All USB devices are connected
    - Implemented on the main board
  - Other hubs or functions can be connected at root hub



Implementation

- USB model

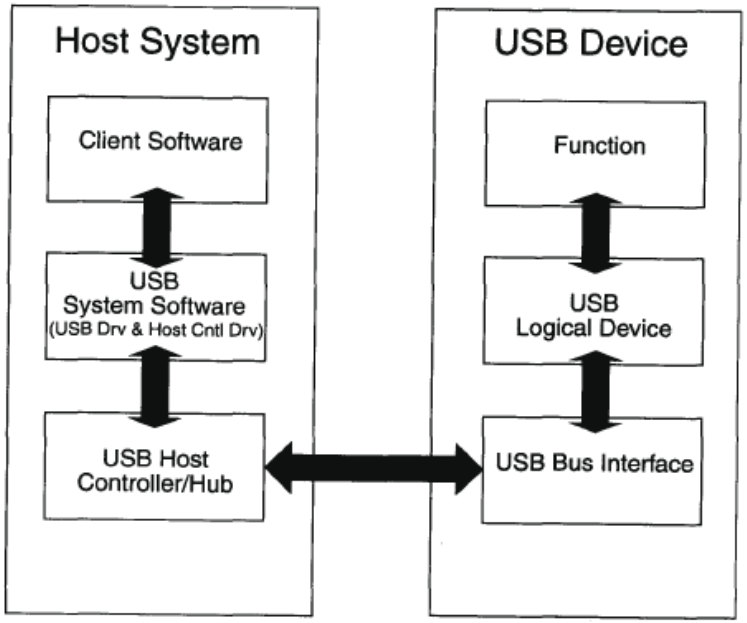


Implementation

- USB model
  - USB is logically implemented as a bus
  - All USB equipment share the same signals
  - Packet oriented
  - All devices will receive all packets even they are not the destination
  - Maximum distance between a hub and a function is 5 m

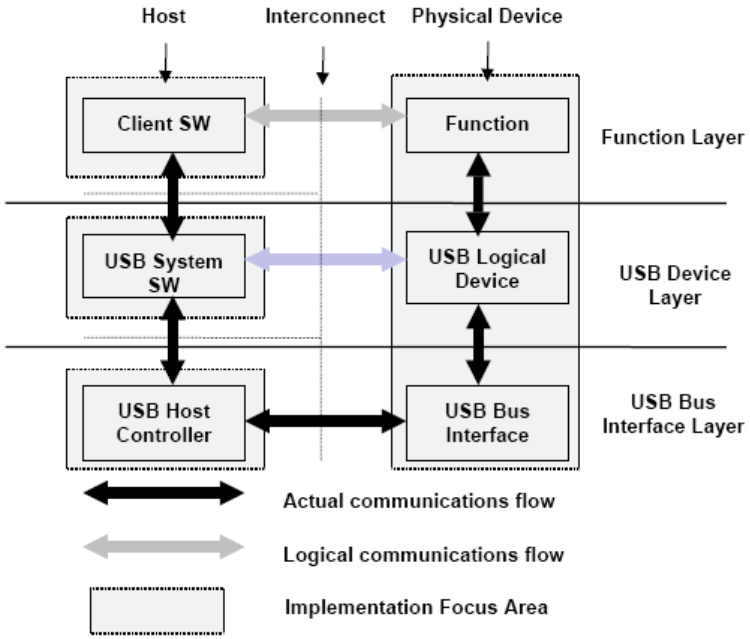
Implementation

- USB stack
  - System interface (host)
  - Device interface (device)



Implementation

- USB stack
  - Physical layer
  - USB layer
  - Function layer



Implementation

- USB model
  - HW/SW solution
    - Hardware: USB root Hub + host controller, USB hub and USB functions, USB physical interface
    - Software: USB device driver, USB protocol driver, USB controller driver
  - USB transactions are initiated by software
    - Device drivers
    - USB driver

Implementation

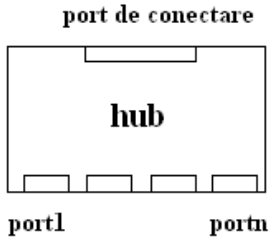
- Physical layer – function devices
  - Devices capable to receive and send USB packets
  - Store identification information
    - Identity
    - Configuration
    - Capabilities
    - Resources
  - Require configuration before usage

Implementation

- Physical layer – function devices
  - Simple – have one function
  - Multiple – have one hub and multiple functions

Implementation

- Physical layer – hubs
  - Interface for functions to connect to the host
  - Interface ports
    - Parent port (upstream)
    - Child port (downstream)

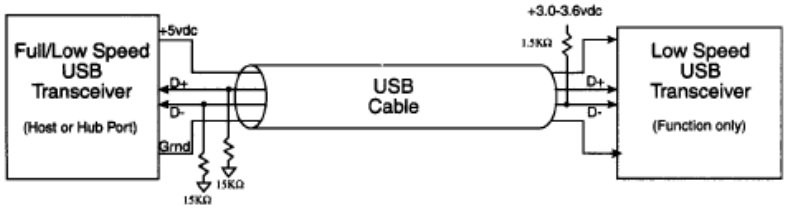


Implementation

- Physical layer – hubs
  - Monitor if new devices are connected to downstream ports
  - Provide power to attached devices
  - Configure ports according to device capabilities (enable/disable, transfer rates)
  - Forward packets to devices or host

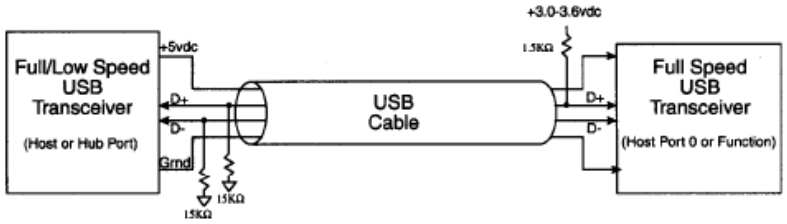
Implementation

- Low-speed USB devices
  - Have an internal resistor connecting V_{cc} to D_-:
    - Hubs are able to identify presence of the device
    - Hubs are able to identify type of the device



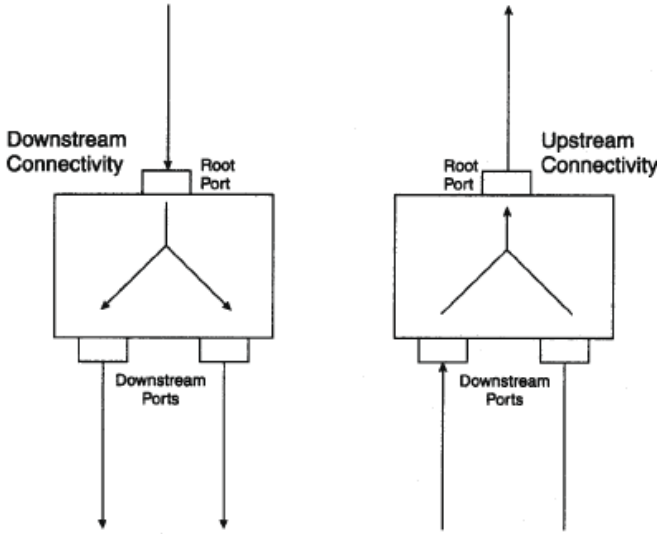
Implementation

- Full-speed USB devices
  - Have an internal resistor connecting V_{cc} to D_+:
    - Hubs are able to identify presence of the device
    - Hubs are able to identify type of the device



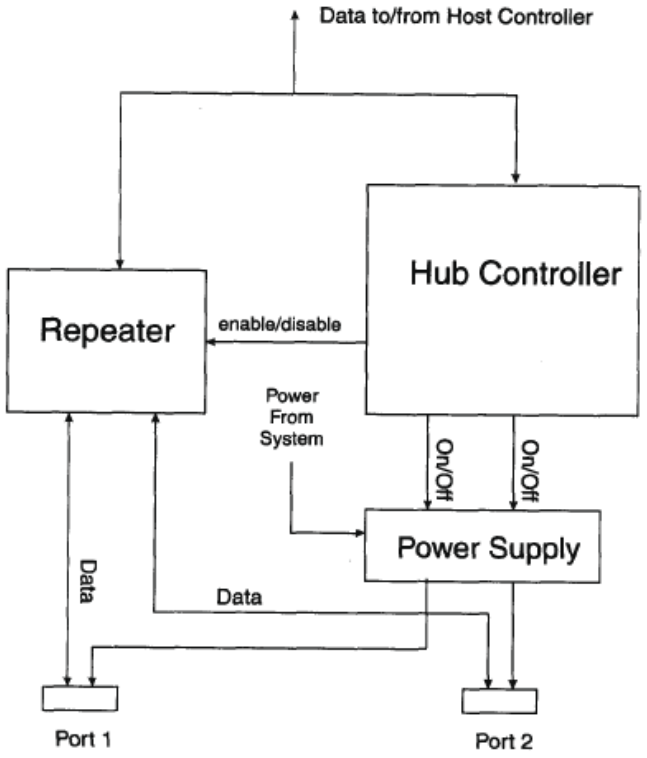
Implementation

- Physical layer – hubs
  - Packets dispatching
    - Upstream
    - Downstream



Implementation

- Physical layer – hubs
  - Hub controller
    - Hub setup
  - Hub repeater
    - Packets switch
    - Amplify signals



Implementation

- Physical layer – hubs
  - Root hub (host)
    - Detects the attach/detach of USB devices
    - Manages data transfers between host and USB devices
    - Monitors the status of USB devices
    - Provides power supply to USB devices

Implementation

- USB protocol layer
  - USB driver
    - Provides a communication interface for upper layers
    - Data transfers are based on transactions
    - Transactions are carried by 1 ms frames
    - Frames will carry packets from multiple applications

Implementation

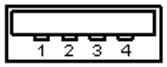
- USB protocol layer
  - USB controller driver
    - Dispatch and schedule packets to frames on USB bus
    - Make use of lists of transactions

Implementation

- USB application layer
  - USB device drivers (USB clients)
    - Initiates transactions to the device
    - It is specific to each type of USB device
      - Specific functions
    - Uses the USB protocol/driver to transfer data
    - USB protocol is transparent at this level

Implementation

- Connectors
  - Type A (upstream)



  - Type B (downstream)

    - Mini-B

    - Micro-B

  - Type C (both upstream/downstream)

USB protocol

- USB communication has two phases
  - Enumeration – communication between host and device needed to identify and configure the device
  - Transfer – communication between host and device carrying application specific data

USB protocol

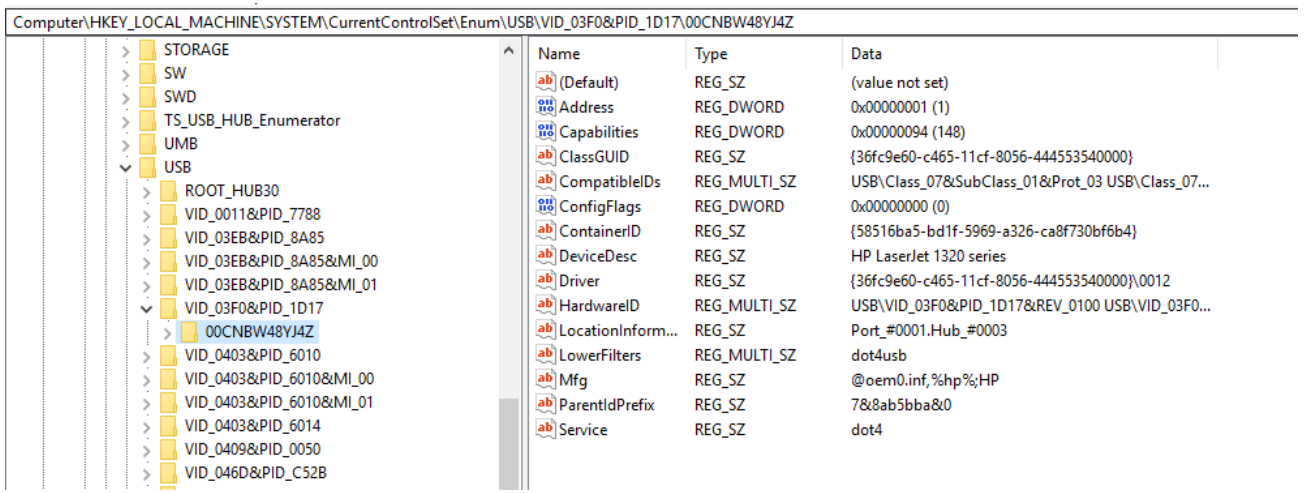
- Enumeration
  - The host can detect the presence of a USB device
    - When attach USB device – the device will receive power
    - The hub detects the presence and the type of device (low/full speed)
    - High-speed operation is selected by host after detection as full-speed if hub and device support it
    - The hub has an interrupt port where from the host will poll for new devices
    - The host polls the attached hubs for new events (attach/detach)

USB protocol

- Enumeration
  - Read device descriptor and allocate an address to the detected device
    - The host has to be able transfer control information to the device even the device has no allocated resources (before having an address)
    - The host will select a hub with a new attached device that will respond to address 0
    - Using endpoint (port) 0 of the device, the host can read the device descriptor
    - The host will assign a new unique address to the device
    - All communications from this point on will use the new address
    - The address is valid until the device is detached, a hub resets the port, or the system reboots.

USB protocol

- Enumeration
  - The host is able to identify the connected USB devices
    - The host reads entire device and configuration descriptors
    - Unique USB identifiers (VID/PID)
    - OS searches for and loads a driver for the detected device

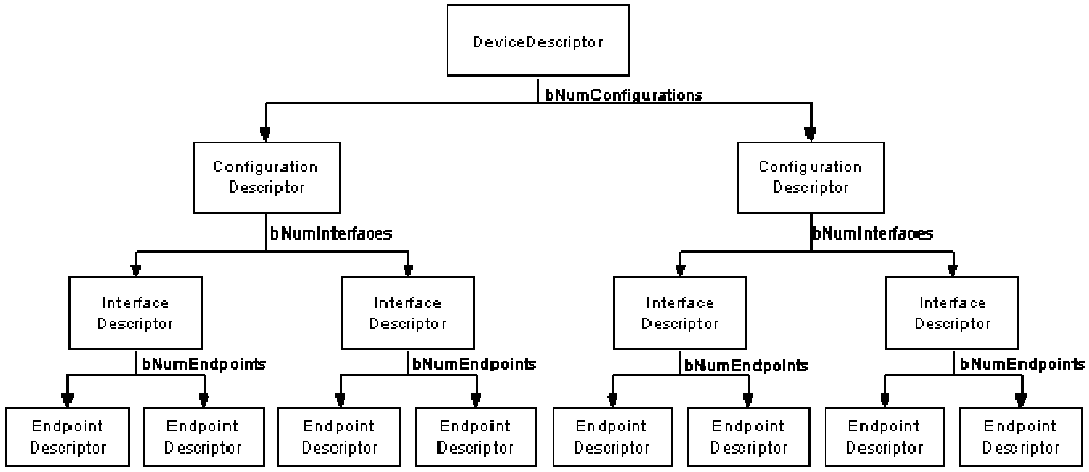


USB protocol

- Enumeration
  - The device driver selects a configuration
  - The driver configure the device (speed, endpoints, power)
  - The host is able to configure a device for data transfers
  - The host will initiate communication and the devices will answer to these requests
  - The driver will continue to initiate specific transactions with the device

USB protocol

- Enumeration
  - Device descriptors



USB protocol

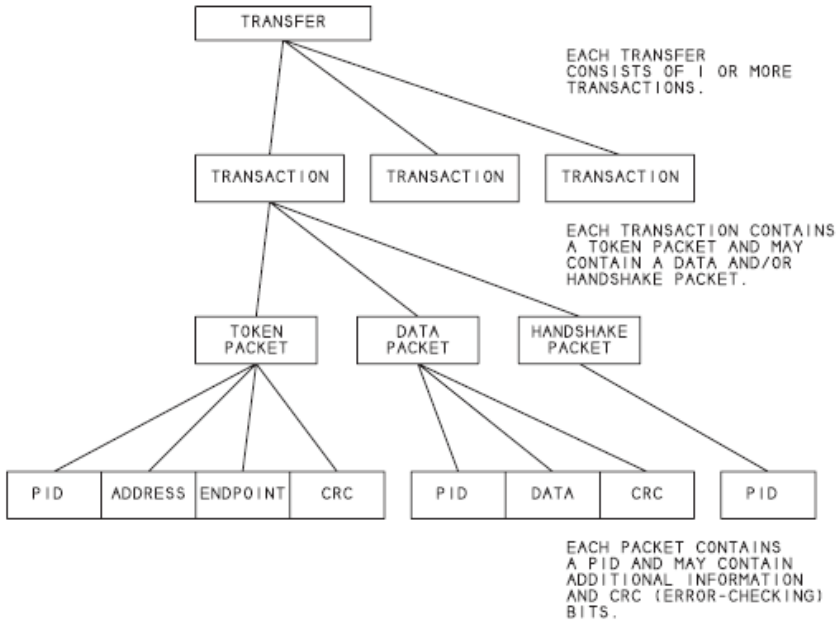
- Enumeration
  - The host/hub will continue with a new undetected device until no such devices are left

USB protocol

- Data transfer
  - Once configured, USB devices can be used in application specific transactions
  - USB specification defines four transfer/endpoint types:
    - Control
    - Interrupts
    - Bulk
    - Isochronous

USB protocol

- USB transfer

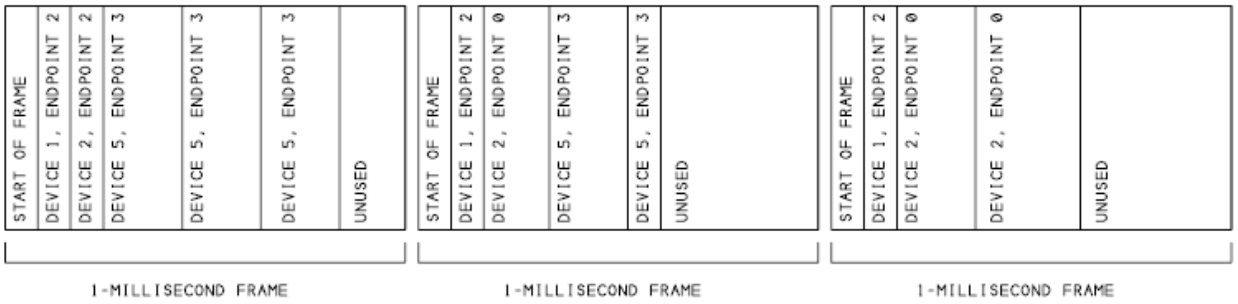


USB protocol

- USB frames
  - The host initiates all transfers with the device
  - The host divides transactions in 1 ms time intervals, names frames
  - The host reserves slots for current transactions on several frames
  - Each frame begins with a special packet called Start-of-Frame (SOF)

USB protocol

- USB frames
  - Transactions are split over several frames
  - All transfers are taking place between the host and one device

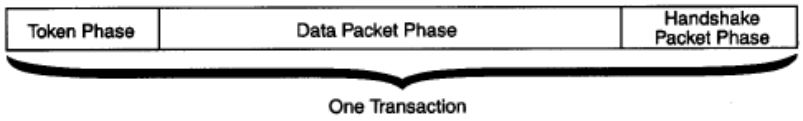


USB protocol

- USB transactions
  - Frame slots are allocated to transactions function of:
    - Type of the transaction
    - Transfer size
    - Requires speed and latency of the transactions
    - Other existing transactions

USB

- USB transactions
  - Have 1 to 3 phases
  - Each phase is encapsulated within one packet



USB protocol

- USB transactions
  - Each transaction starts with a packet, called token
  - The token, sent by the host, carries the address of the destination device and the type of transaction
  - Data packets hold transaction specific data
  - Handshake packets provide the feedback for the current transaction

USB protocol

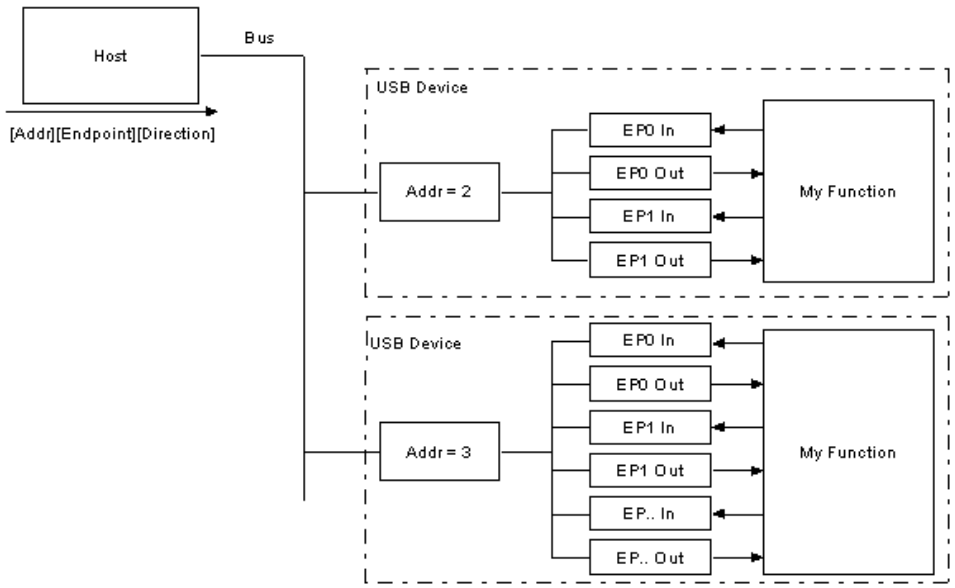
- USB addressing
  - Each transaction has to specify the address of the target device
  - Each device has a unique address given by the host once detected in the enumeration phase
  - USB devices have one or more interfaces under the same address, the host can communicate with
  - An endpoint is an interface buffer or register internal to the device

USB protocol

- USB addressing
  - The host does not transfer data directly to a device
  - The host transfers data to device endpoints
  - Each endpoint has a 4 bits address and one direction (IN or OUT)
  - All devices require endpoint 0 for control

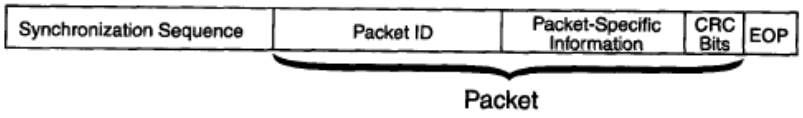
USB protocol

- USB addressing



USB protocol

- USB packets
  - Represents the basic communication unit of USB protocol
  - They have:
    - Synchronization sequence
    - Packet type
    - Payload (data, address)
    - CRC
    - End of packet (status on physical lines)

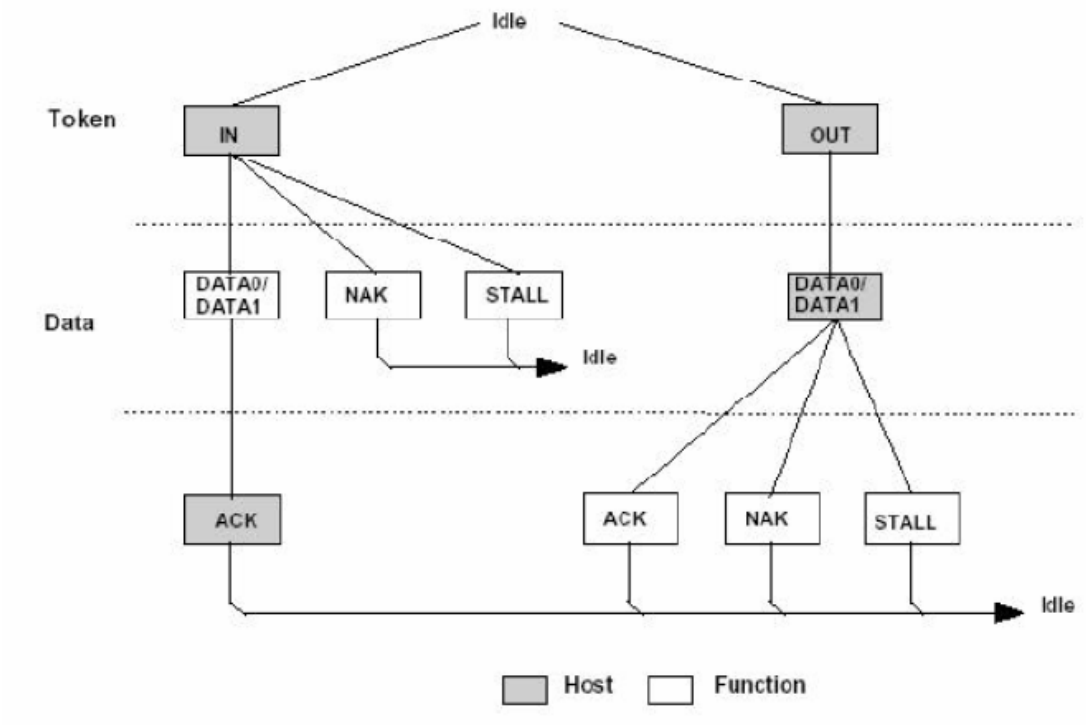


USB protocol

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
	DATA2	0111B	Data packet PID high-speed, high bandwidth isochronous transaction in a microframe (see Section 5.9.2 for more information)
	MDATA	1111B	Data packet PID high-speed for split and high bandwidth isochronous transactions (see Sections 5.9.2, 11.20, and 11.21 for more information)
Handshake	ACK	0010B	Receiver accepts error-free data packet
	NAK	1010B	Receiving device cannot accept data or transmitting device cannot send data
	STALL	1110B	Endpoint is halted or a control pipe request is not supported
	NYET	0110B	No response yet from receiver (see Sections 8.5.1 and 11.17-11.21)
Special	PRE	1100B	(Token) Host-issued preamble. Enables downstream bus traffic to low-speed devices.
	ERR	1100B	(Handshake) Split Transaction Error Handshake (reuses PRE value)
	SPLIT	1000B	(Token) High-speed Split Transaction Token (see Section 8.4.2)
	PING	0100B	(Token) High-speed flow control probe for a bulk/control endpoint (see Section 8.5.1)
	Reserved	0000B	Reserved PID

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

USB protocol



USB protocol

- USB transfers
  - Control
    - Identification and configuration of devices
    - Channel control
  - Bulk
    - Transfer of large amount of data with no special requirements (bandwidth, latency)
  - Interrupts
    - Transfer of small data blocks with low latency
  - Isochronous
    - Real-time transfer of large amount of data (high bandwidth)

USB protocol

- USB transfers

Transfer Type	*Description*	*Lossless?*	*Size(s)*	*Latency Guarantee?*
Control	Used to send and receive structured information of a control nature	Yes	≤ 8, 16, 32, or 64 bytes	Best effort
Bulk	Used to send or receive small blocks of unstructured data	Yes	≤ 8, 16, 32, or 64 bytes	No
Interrupt	Like a bulk pipe, butincludes a maximum latency	Yes	≤ 64 bytes	Polled at guaranteed minimum rate
Isochronous	Used to send or receive large blocks of unstructured data with guaranteed periodicity	No	≤ 1023 bytes	Fixed portion of every 1-ms frame

USB protocol

- USB transfers

Transfer Type	Control	Bulk	Interrupt	Isochronous
Typical Use	Identification and configuration	Printer, scanner, drive	Mouse, keyboard	Streaming audio, video
Required?	yes	no	no	no
Low speed allowed?	yes	no	yes	no
Data bytes/millisecond per transfer, maximum possible per pipe (high speed).*	15,872 (thirty-one 64-byte transactions/ microframe)	53,248 (thirteen 512-byte transactions/ microframe)	24,576 (three 1024-byte transactions/ microframe)	24,576 (three 1024-byte transactions/ microframe)
Data bytes/millisecond per transfer, maximum possible per pipe (full speed).*	832 (thirteen 64-byte transactions/ frame)	1216 (nineteen 64-byte transactions/ frame)	64 (one 64-byte transaction/ frame)	1023 (one 1023-byte transaction/ frame)
Data bytes/millisecond per transfer, maximum possible per pipe (low speed).*	24 (three 8-byte transactions)	not allowed	0.8 (8 bytes per 10 milliseconds)	not allowed
Direction of data flow	IN and OUT	IN or OUT	IN or OUT (USB 1.0 sup-ports IN only)	IN or OUT
Reserved bandwidth for all transfers of the type (percent)	10 at low/full speed, 20 at high speed (minimum)	none	90 at low/full speed, 80 at high speed (isochronous & interrupt combined, maximum)	
Error correction?	yes	yes	yes	no
Message or Stream data?	message	stream	stream	stream
Guaranteed delivery rate?	no	no	no	yes
Guaranteed latency (maximum time between transfers)?	no	no	yes	yes
*Assumes transfers use maximum packet size.				

USB protocol

- USB transfers

Transfer Type		Transactions	Phases (packets). Each downstream, low-speed packet is also preceded by a PRE packet.
Control	Setup Stage	One transaction	Token
			Data
			Handshake
	Data Stage	Zero or more transactions (IN or OUT)	Token
			Data
			Handshake
	Status Stage	One transaction (opposite direction of transaction(s) in the Data stage or IN if there is no Data stage)	Token
			Data
			Handshake
Bulk		One or more transactions (IN or OUT)	Token
			Data
			Handshake
Interrupt		One or more transactions (IN or OUT)	Token
			Data
			Handshake
Isochronous		One or more transactions (IN or OUT)	Token
			Data

USB descriptors

- Device descriptor
  - General information about the device

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of the Descriptor in Bytes (18 bytes)
1	bDescriptorType	1	Constant	Device Descriptor (0x01)
2	bcdUSB	2	BCD	USB Specification Number which device complies too.
4	bDeviceClass	1	Class	Class Code (Assigned by USB Org) If equal to Zero, each interface specifies it's own class code If equal to 0xFF, the class code is vendor specified. Otherwise field is valid Class Code.
5	bDeviceSubClass	1	SubClass	Subclass Code (Assigned by USB Org)
6	bDeviceProtocol	1	Protocol	Protocol Code (Assigned by USB Org)
7	bMaxPacketSize	1	Number	Maximum Packet Size for Zero Endpoint. Valid Sizes are 8, 16, 32, 64
8	idVendor	2	ID	Vendor ID (Assigned by USB Org)
10	idProduct	2	ID	Product ID (Assigned by Manufacturer)
12	bcdDevice	2	BCD	Device Release Number
14	iManufacturer	1	Index	Index of Manufacturer String Descriptor
15	iProduct	1	Index	Index of Product String Descriptor
16	iSerialNumber	1	Index	Index of Serial Number String Descriptor
17	bNumConfigurations	1	Integer	Number of Possible Configurations

USB descriptors

- Configuration descriptors
  - Device characteristics and capabilities

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of Descriptor in Bytes
1	bDescriptorType	1	Constant	Configuration Descriptor (0x02)
2	wTotalLength	2	Number	Total length in bytes of data returned
4	bNumInterfaces	1	Number	Number of Interfaces
5	bConfigurationValue	1	Number	Value to use as an argument to select this configuration
6	iConfiguration	1	Index	Index of String Descriptor describing this configuration
7	bmAttributes	1	Bitmap	D7 Reserved, set to 1. (USB 1.0 Bus Powered) D6 Self Powered D5 Remote Wakeup D4..0 Reserved, set to 0.
8	bMaxPower	1	mA	Maximum Power Consumption in 2mA units

USB descriptors

- Interface descriptors
  - Interfaces are groups of endpoints used to access a USB function

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of Descriptor in Bytes (9 Bytes)
1	bDescriptorType	1	Constant	Interface Descriptor (0x04)
2	bInterfaceNumber	1	Number	Number of Interface
3	bAlternateSetting	1	Number	Value used to select alternative setting
4	bNumEndpoints	1	Number	Number of Endpoints used for this interface
5	bInterfaceClass	1	Class	Class Code (Assigned by USB Org)
6	bInterfaceSubClass	1	SubClass	Subclass Code (Assigned by USB Org)
7	bInterfaceProtocol	1	Protocol	Protocol Code (Assigned by USB Org)
8	iInterface	1	Index	Index of String Descriptor Describing this interface

USB descriptors

- Endpoints descriptors

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of Descriptor in Bytes (7 bytes)
1	bDescriptorType	1	Constant	Endpoint Descriptor (0x05)
2	bEndpointAddress	1	Endpoint	Endpoint Address Bits 0..3b Endpoint Number. Bits 4..6b Reserved. Set to Zero Bits 7 Direction 0 = Out, 1 = In (Ignored for Control Endpoints)
3	bmAttributes	1	Bitmap	Bits 0..1 Transfer Type 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt Bits 2..7 are reserved. If Isochronous endpoint, Bits 3..2 = Synchronisation Type (Iso Mode) 00 = No Synchonisation 01 = Asynchronous 10 = Adaptive 11 = Synchronous Bits 5..4 = Usage Type (Iso Mode) 00 = Data Endpoint 01 = Feedback Endpoint 10 = Explicit Feedback Data Endpoint 11 = Reserved
4	wMaxPacketSize	2	Number	Maximum Packet Size this endpoint is capable of sending or receiving
6	bInterval	1	Number	Interval for polling endpoint data transfers. Value in frame counts. Ignored for Bulk & Control Endpoints. Isochronous must equal 1 and field may range from 1 to 255 for interrupt endpoints.

USB classes

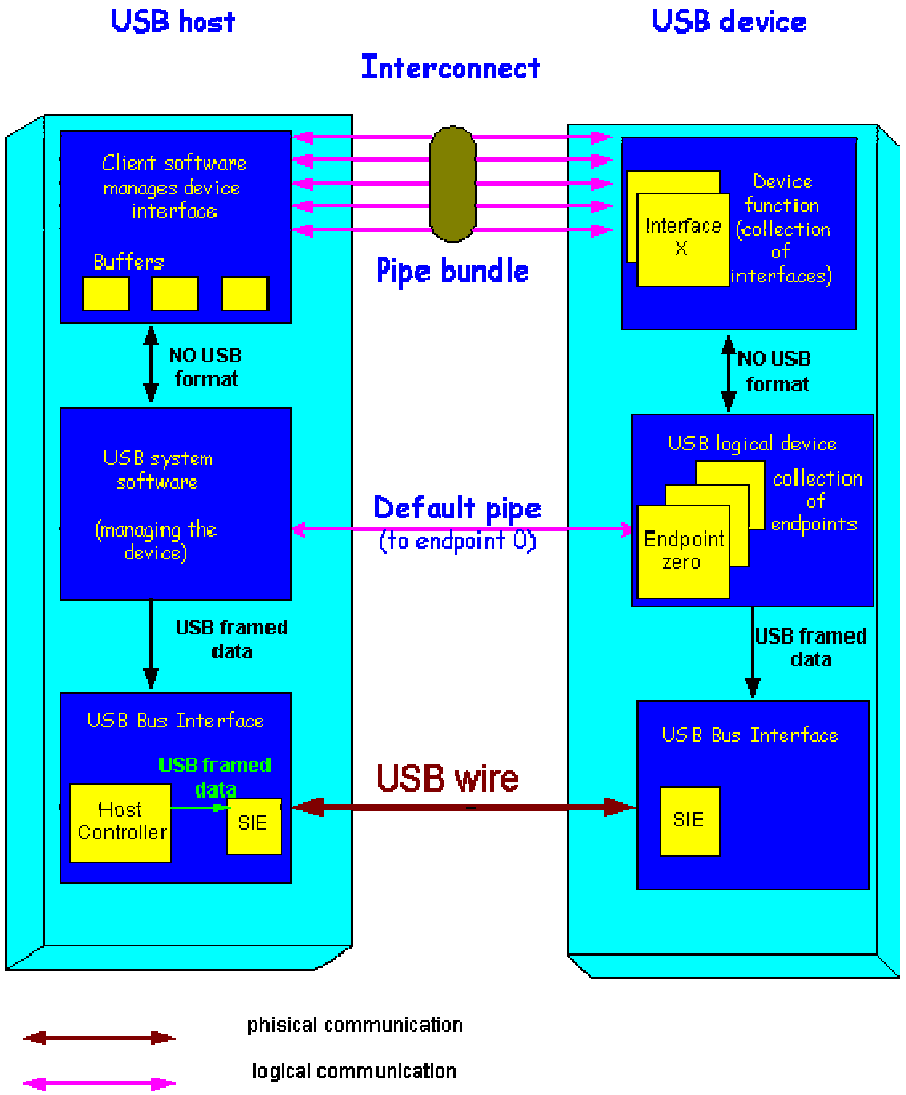
- Similar USB devices are grouped in classes
- Each class has class specific attributes and provides standardized services
- Class specifications are used by developers to develop the HW and device drivers

USB classes

Class	Descriptor Where Class Is Declared
Audio	Interface
Chip/Smart Card Interface	Interface
Communication	Device or Interface
Content Security	Interface
Device Firmware Upgrade	Interface (subclass of Application Specific Interface)
Human Interface (HID)	Interface
IrDA Bridge	Interface (subclass of Application Specific Interface)
Mass Storage	Interface
Printer	Interface
Still Image Capture	Interface
Test and Measurement	Interface (subclass of Application Specific Interface)
Video	Interface

USB programming

- USB device drivers do not access USB function directly
- USB device drivers will use USB driver to transfer data with USB function
  - Driver will call another driver
- Function data are encapsulated into an URB - USB request block


USB host
USB device
Interconnect
Client software manages device interface
Buffers
NO USB format
USB system software
(managing the device)
USB framed data
USB Bus Interface
USB framed data
Host Controller
SIE
Pipe bundle
Default pipe
(to endpoint 0)
USB wire
Interface X
Device function (collection of interfaces)
NO USB format
USB logical device
collection of endpoints
Endpoint zero
USB framed data
USB Bus Interface
SIE
phisical communication
logical communication

USB Driver

The USB driver knows the characteristics of the USB target device and how to communicate with the device via the USB. The USB characteristics are detected by the USB driver when it parses the device descriptors during device configuration. For example, some devices require a specific amount of throughput during each frame, while others may only require periodic access every nth frame.

When an IRP is received from a USB client driver, the USB driver organizes the request into individual transactions that will be executed during a series of 1ms frames. The USB driver sets up the transactions based on its knowledge of the USB device requirements, the needs of the client driver, and the limitations/capabilities of the USB.

Depending on the operating environment, the USB driver may be shipped along with the operating system or added as an extension via a loadable device driver.

USB Host Controller Driver

The USB host controller driver (HCD) schedules transactions to be broadcast over the USB. Transactions are scheduled by the host controller driver by building a series of transaction lists. Each list consists of pending transactions targeted for one or more of the USB devices attached to the bus. A transaction list, or frame list, defines the sequence of transactions to be performed during each 1ms frame. The USB host controller executes these transaction lists at 1ms intervals. Note that a single block transfer requested by a USB client may be performed as a series of transactions that are scheduled and executed during consecutive 1ms frames. The actual scheduling depends on a variety of factors including; the type of transaction, transfer requirements specified by the device, and the transaction traffic of other USB devices.

The USB host controller initiates transactions via its root hub or hubs. Each 1ms frame begins with a start of frame (SOF) transaction and is followed by the serial broadcast of all transactions contained within the current list. For example, if one of the requested transactions is a request to transfer data to a USB printer, the host controller would obtain the data to be sent from a memory buffer supplied by the client software and transmit the data over the USB. The hub portion of the controller converts the requested transactions into the low level protocols required by the USB.