System Architecture

Universal Serial Bus

Ashish Kuvelkar

HPC Technologies Group

C-DAC, Pune

What is USB?

- Universal Serial Bus
- Industry Standard
- Defines cables, connectors and protocol between PC and devices used for
  - Connection
  - Communication
  - Power Supply

Need for USB

- USB emerged as a result of difficulties associated with peripheral devices in PC environment
- Concerns of designers
  - Legacy IO paradigm
  - IRQs cannot be shared
  - Only 10 lines used for IO decoding puts severe limitation of spare addresses
  - Interfaces allow single device to be connected
- User concerns
  - Dedicated cables for each peripheral
  - Configuration of expansion cards
  - No hot attachment or plug & play

Development of USB over years

- The first widely used USB standard, USB 1.1, was created by Compaq, DEC, IBM, Intel, Microsoft, NEC & Northern Telecom
- First USB ready systems were available in 1995
- OS support was extended from Windows 98 and MacOS 8.6 onwards
- USB's success lies in its support for "hot-swappable" operation and the "Plug 'n' Play" feature

Key USB Features

- Low-cost solution for attaching peripheral devices to PC
- Hot pluggable; attached device automatically detected and configured
- Single connector to attach any device

- Supports attachment of 127 devices
- Two device speeds 1.5Mb/s and 12Mb/s

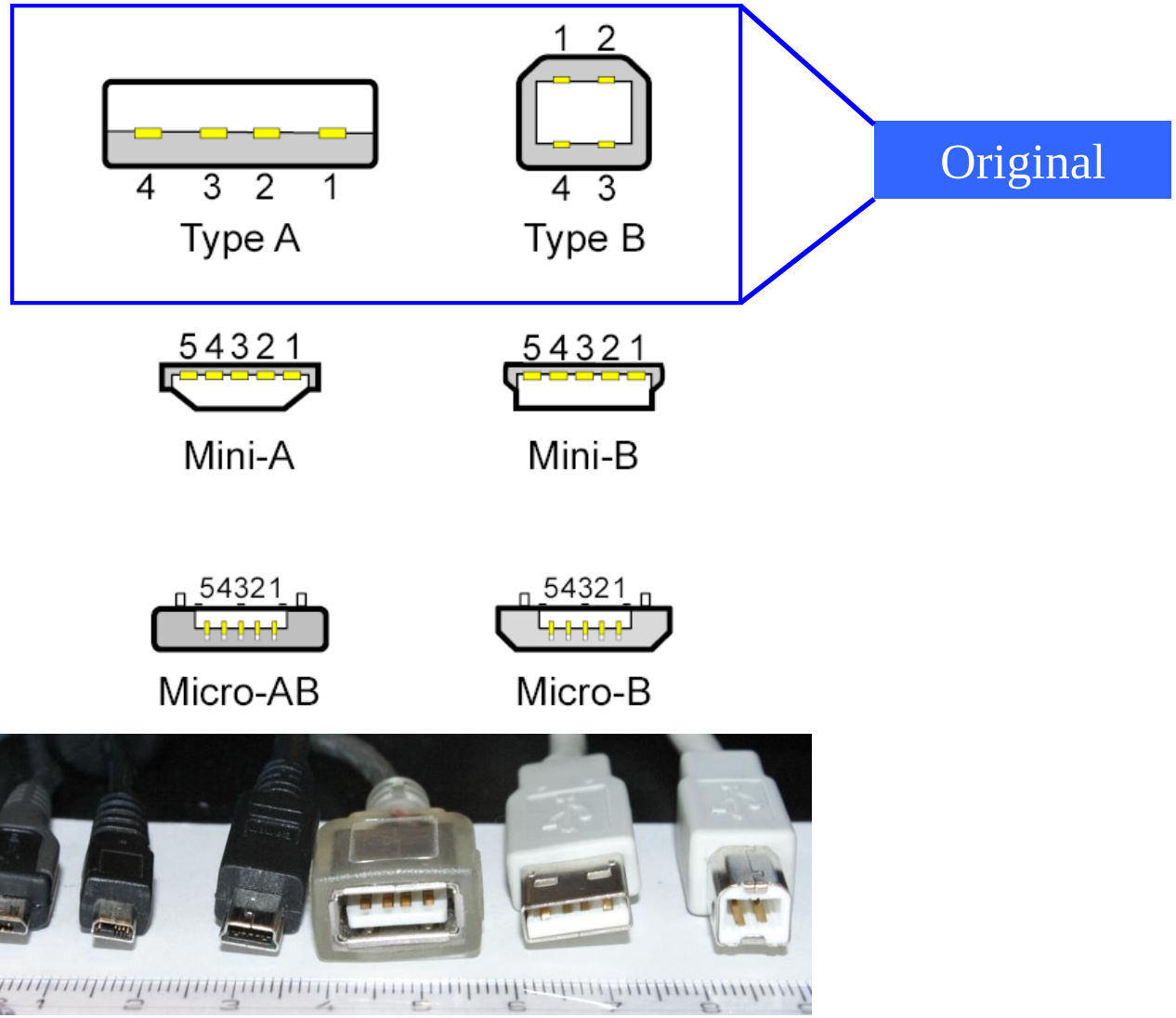
Key USB Features(contd.)

- Peripherals can be powered from cable
- USB devices do not use any system resources like IRQ, IO address space etc.
- USB transactions include error detection mechanisms
- USB devices conserve power by entering a suspend state (>500uA current)after 3ms of no bus activity.
- Support for four transfer types to suit different transfer characteristics of peripherals

USB 2.0

- Provision for higher speeds. 40x improvement !!
- Downward compatible. Same high speed cables can be used.
- A microframe of 1/8 of existing frame is introduced.
- Low cost and higher performance.

Types of USB Connectors



USB 3.0

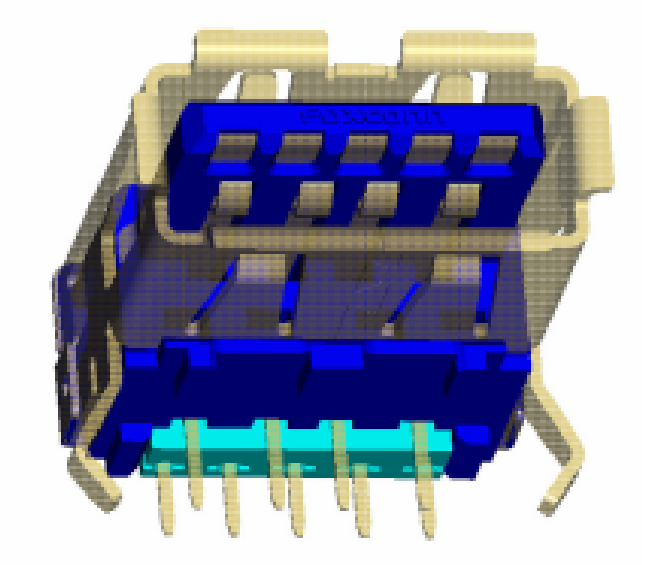
- 10x performance over USB 2.0
  - "SuperSpeed" provides transfer rate at 5.0 Gbit/s. The raw throughput is 4 Gbit/s, achieve 3.2 Gbit/s (400 MByte/s), or more, after protocol overhead
- Backward Compatible
  - Legacy devices work when plugged in new host connector
  - New devices work with plugged in legacy system, but at 2.0 speeds
  - Existing class drivers continue to work
- Same USB device model
  - Pipe model, USB framework, transfer types

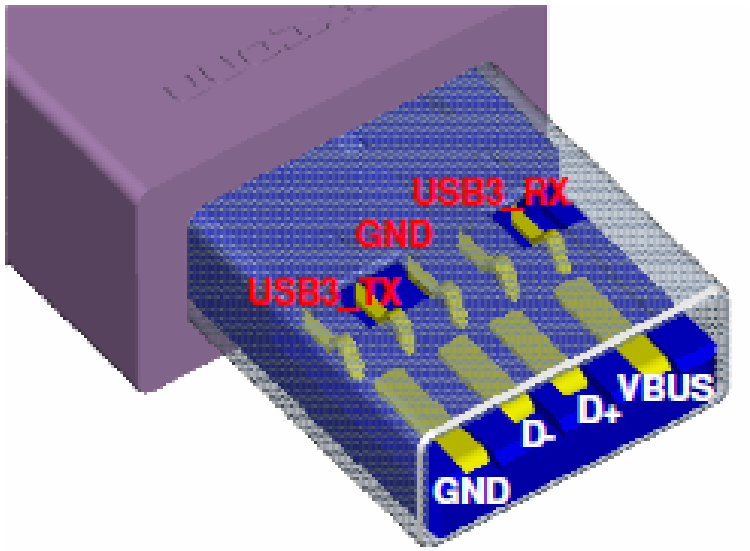
Type A Connector

Same interface as the USB 2.0 Standard-A connector, but with added pins for SuperSpeed USB signals

Complete compatibility with USB 2.0 Standard-A connector

Double-stacked connectors supported





Achieving higher performance

- Additional physical bus that is added in parallel with the existing USB 2.0 bus.
- This means that where USB 2.0 previously had 4 wires (power, ground, and a pair for differential data)
  - USB 3.0 adds 4 more for two pairs of differential signals (receive and transmit) for a combined total of 8 connections in the connectors and cabling.
- USB 3.0 utilizes a bi-directional data interface rather than USB 2.0's half-duplex arrangement

USB Devices

Smart Card Reader

Flash Drive

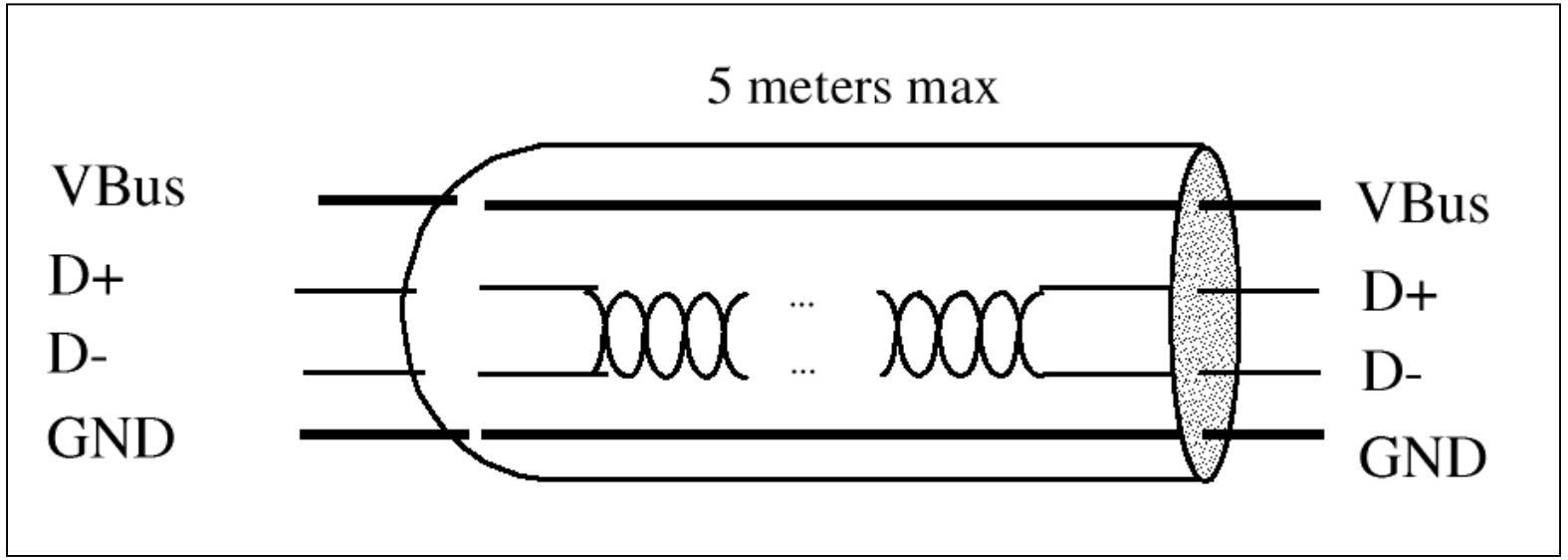
MP3 Player

Pedals

20GB Hard Disk

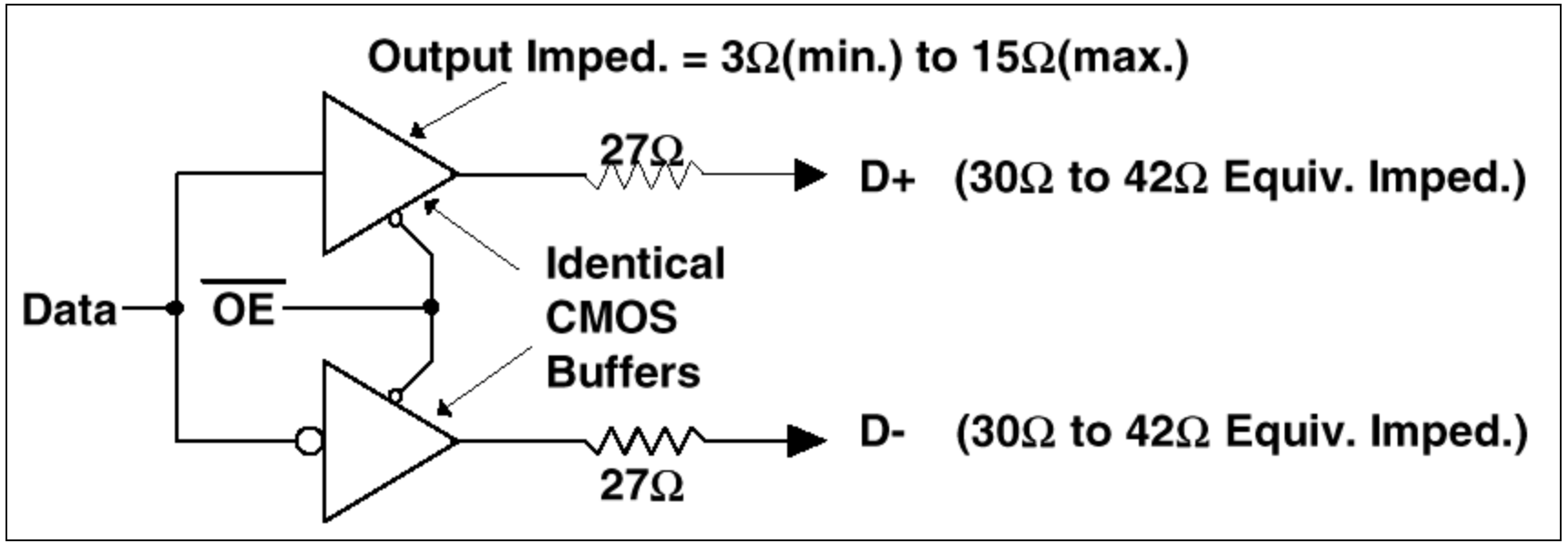
GPS Receiver

The Electrical Interface



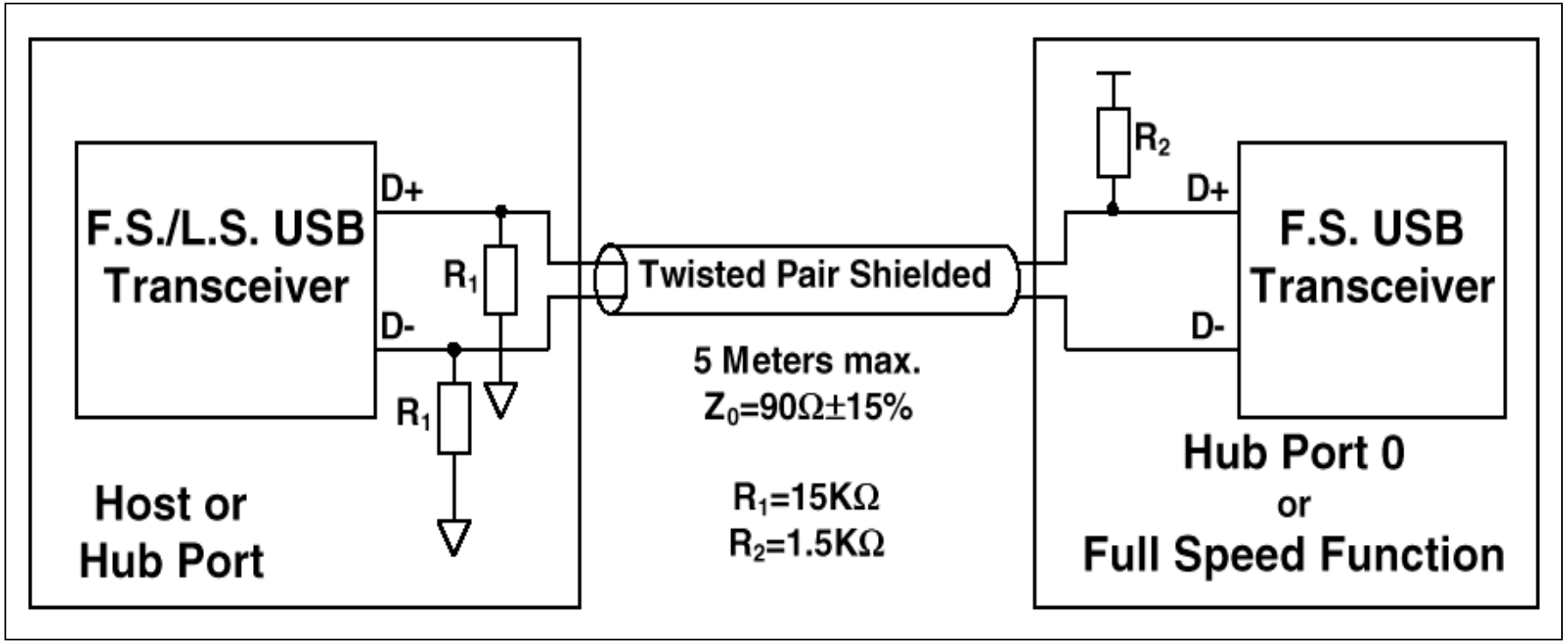
- Differential data; 90 ohm impedance; 5 meters max.
- VBus is power for devices. 5V at source.
- Two signaling speeds
  - low speed 1.5Mbs unshielded cable
  - full speed 12Mbs shielded cable

CMOS Driver



- VOH Max - 2.8 to 3.6V
- Rise and fall times between 4 to 20 ns. (full speed)

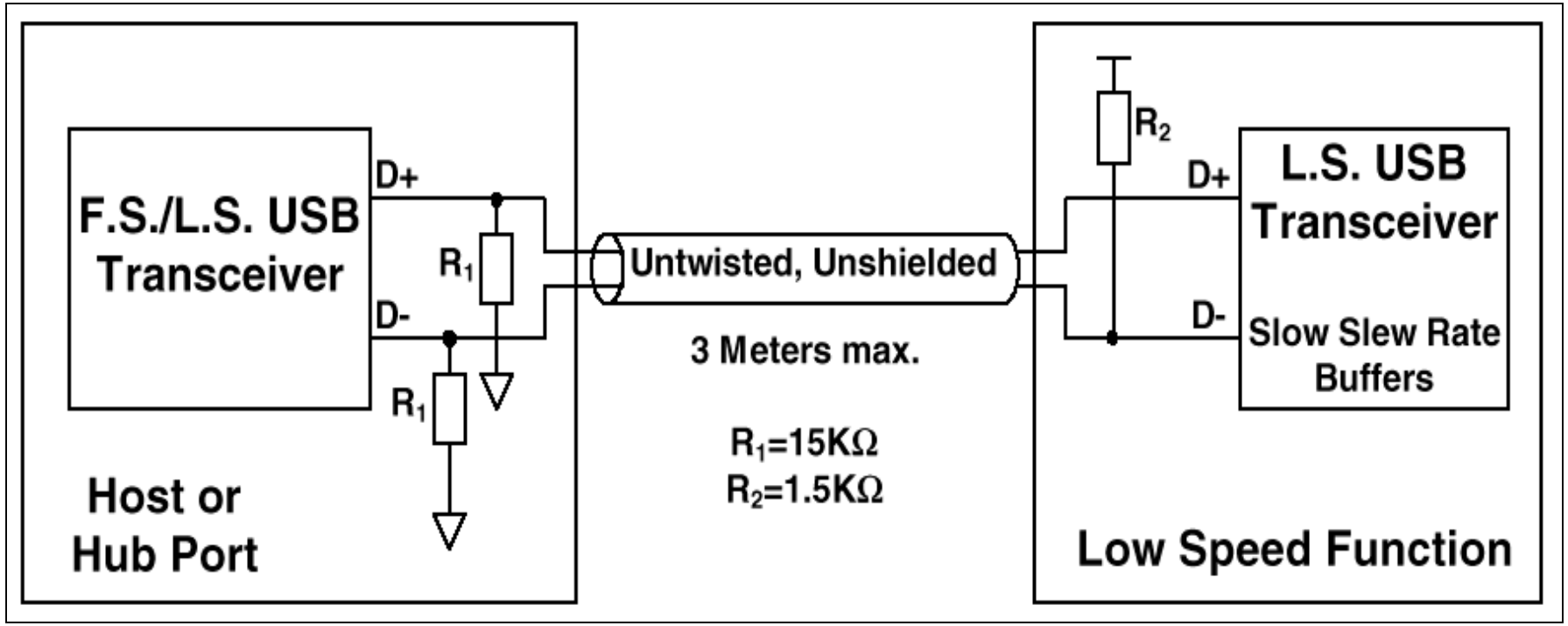
Full speed - cable and termination



D+ is pulled up.

Receiver sensitivity - 200 mV min.

Low speed - cable and termination



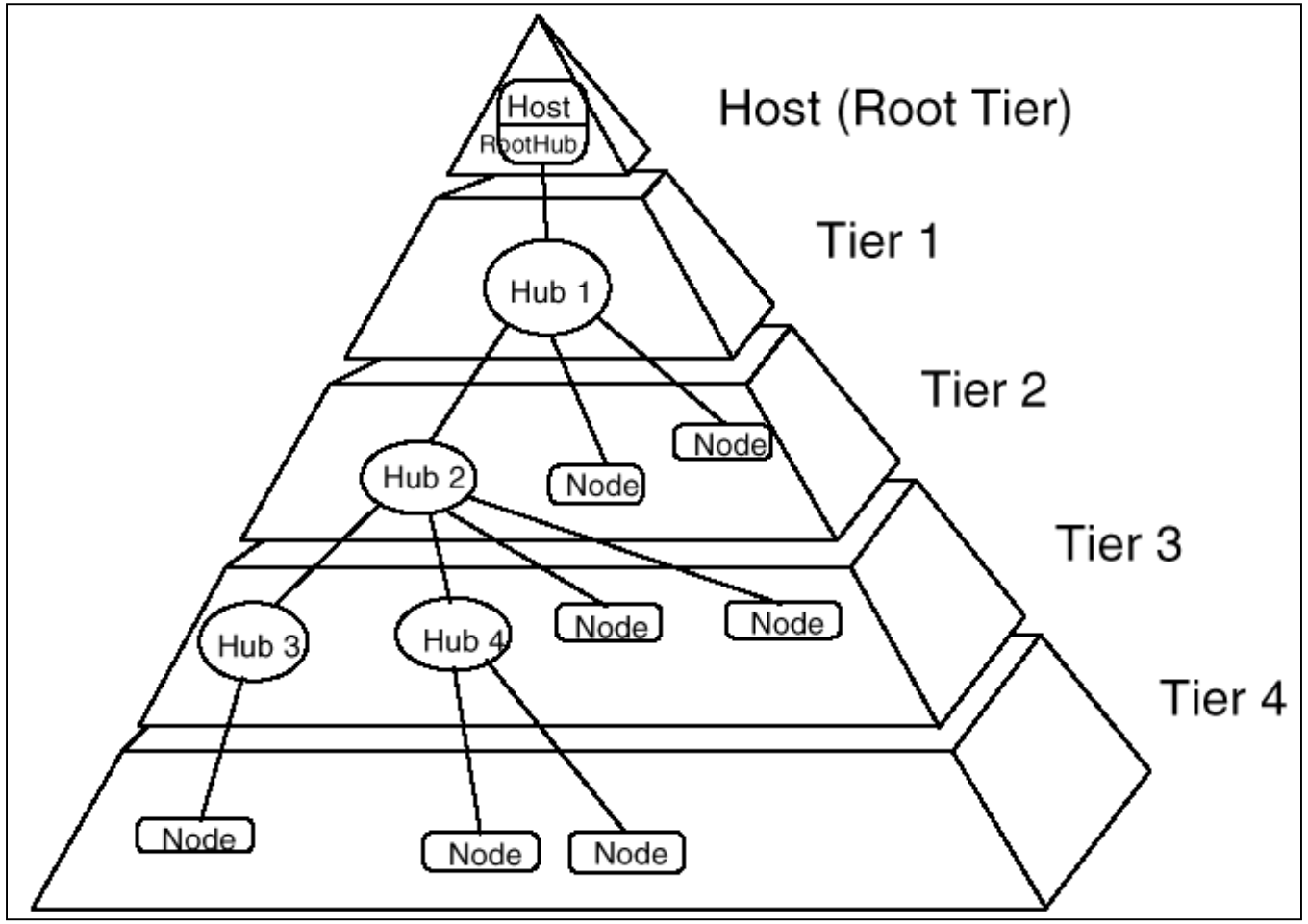
D- is pulled up.

Receiver sensitivity - 200 mV min.

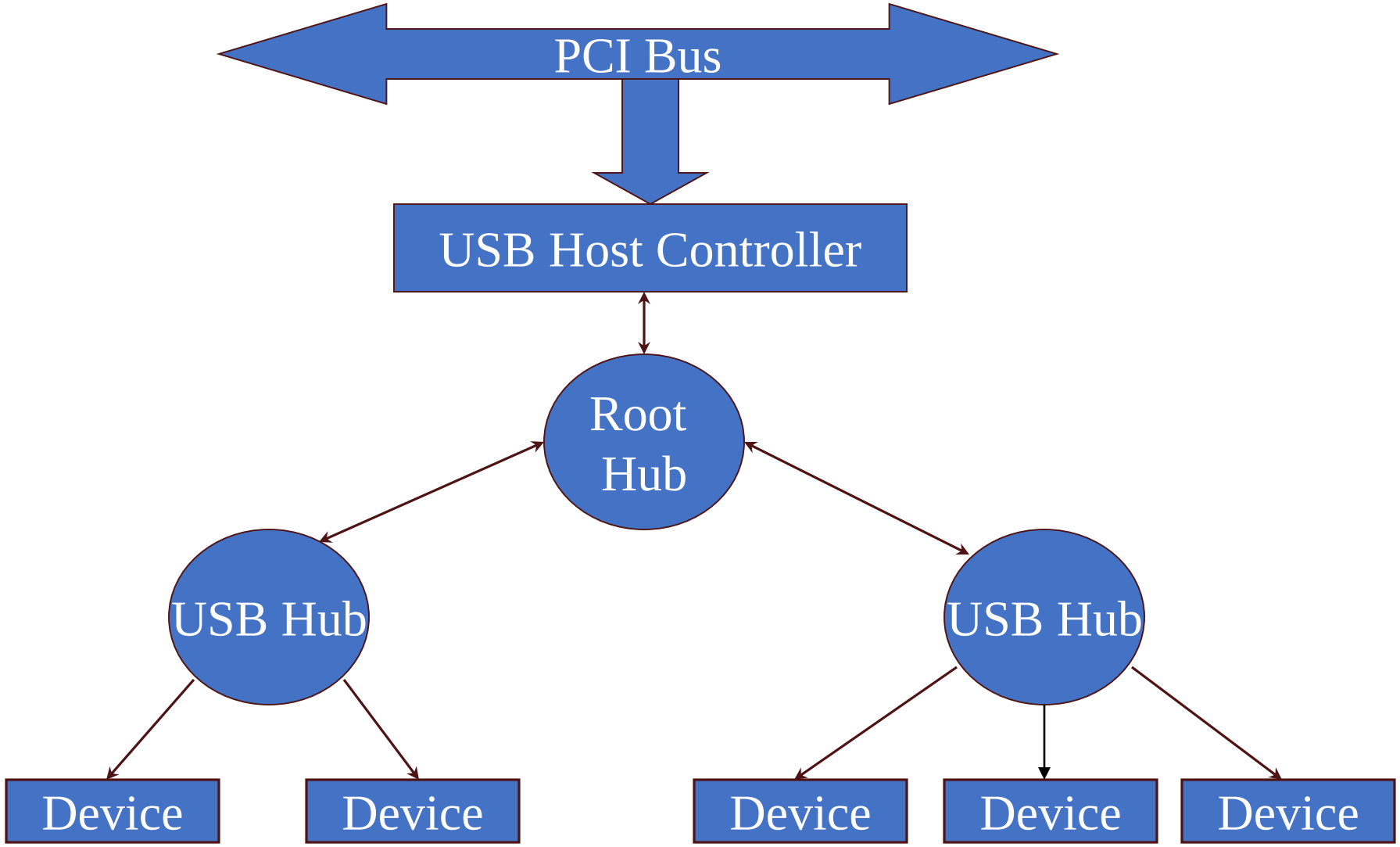
USB Power

- USB 1.X and 2.0 use 5V
- Unit load is 100mA in USB 2.0 and 150mA in USB 3.0
- Device may draw maximum 5 unit loads
- Two types of USB devices
  - Low power (1 unit load)
  - High powered (max defined by standard)

The Tiered Star Topology



Max 5 hops


PCI Bus
USB Host Controller
Root Hub
USB Hub
USB Hub
Device
Device
Device
Device
Device

USB components

- The USB host
  - There can be only one host in the USB system
  - The USB controller may be implemented in a combination of hardware, firmware and software.
- USB devices
  - Hubs, which provide additional attachment points to the USB.
  - Functions, which provide capabilities to the system; eg. Mouse, speakers, digital camera, tablet, etc.

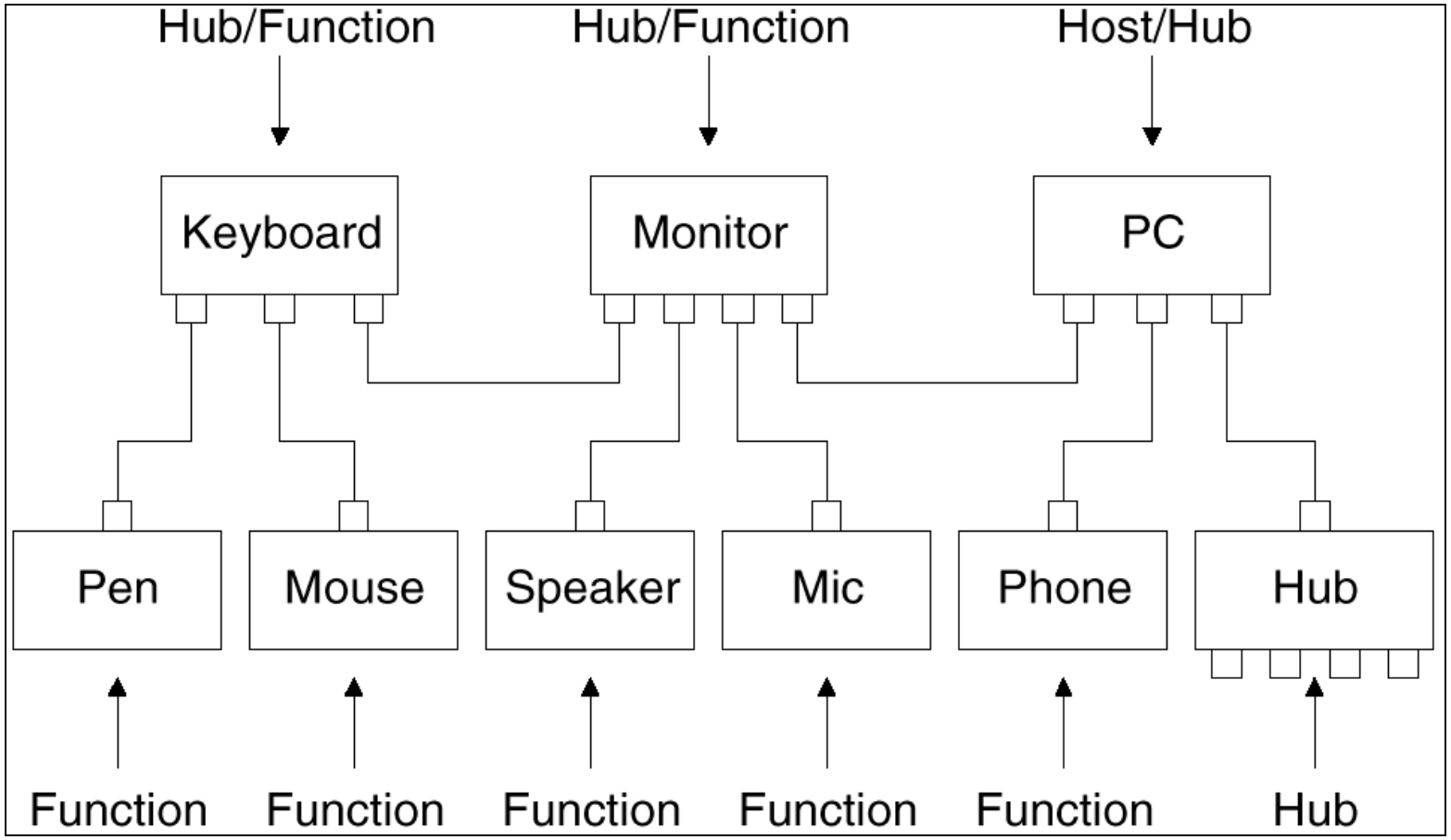
Functions of the Host

- Detect devices
  - During power-on, on attachment
- Manage data flow
  - Arbitration, priority, maintain data rates
- Error checking
  - Add bits during send, check during receive
- Provide power
  - For devices that draw power from host
- Exchange data with peripherals
  - Primary task

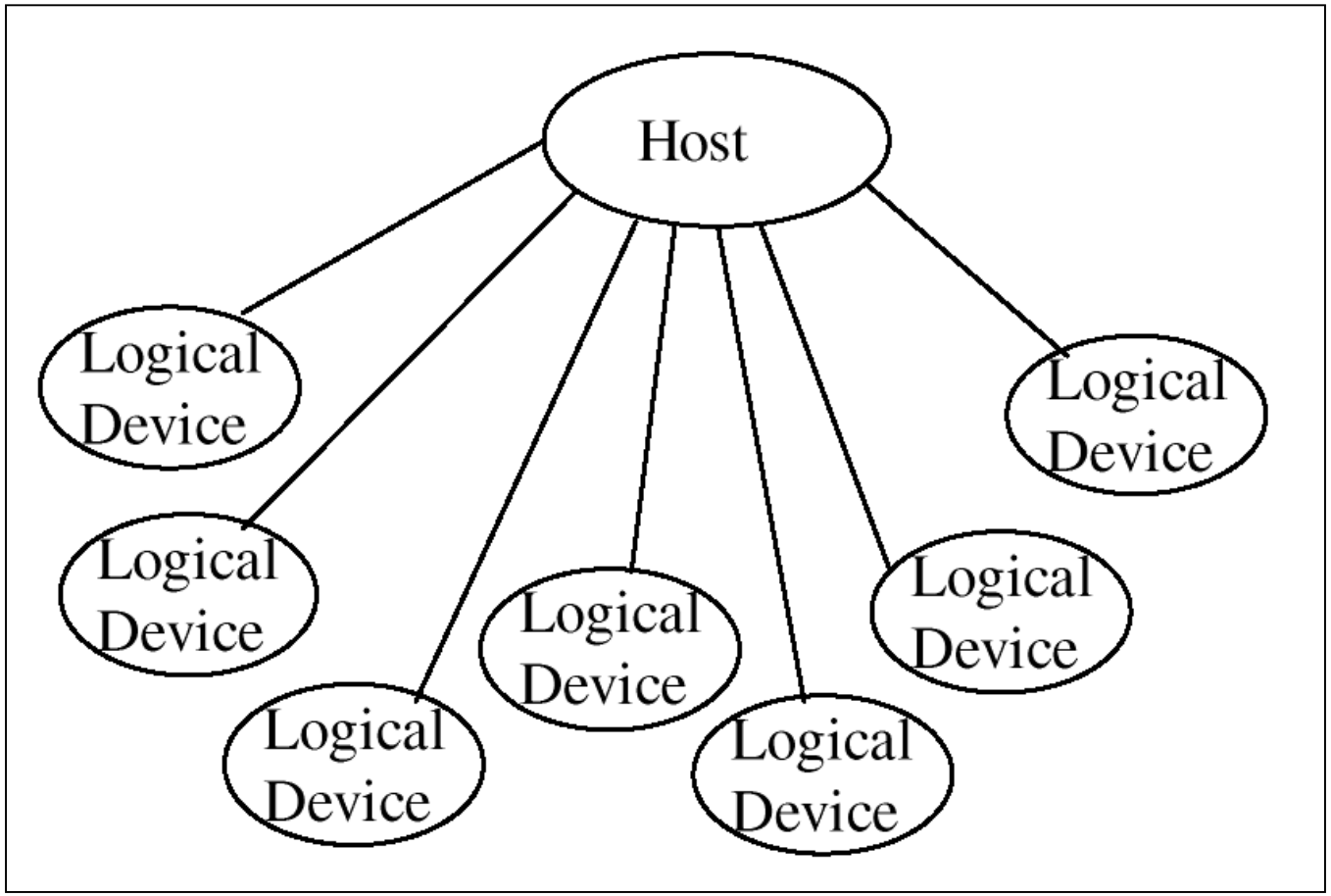
Functions of the Peripherals

- Detect communication for the device
  - Filter communication meant for device
- Respond to standard request
  - During attachment, describe themselves
- Error checking
  - Add bits during send, check during receive
- Manage power
  - For devices that draw power from host
- Exchange data with host
  - Primary task

Desktop Computing Environment



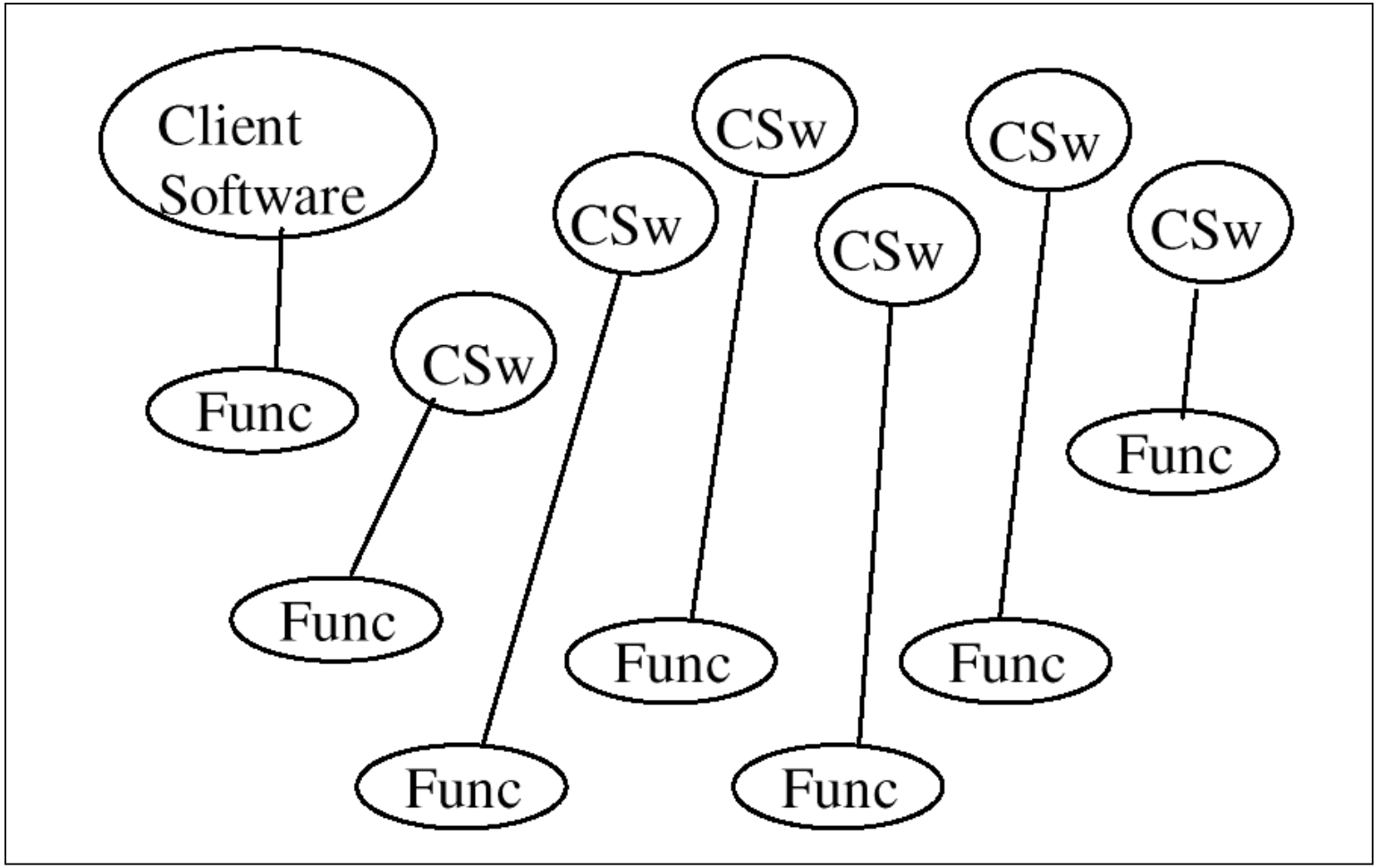
The Logical Topology



USB End-points

- An endpoint is a uniquely identifiable portion of a USB device that is the terminus of a communication flow between the host and device.
- Each USB logical device is composed of a collection of independently operating endpoints.
- Software may only communicate with a USB device via one or more endpoints.
- All USB devices are required to have an endpoint with endpoint number 0. This is used to initialize and generically manipulate the logical device.
- Endpoint 0 provides access to the device's configuration information and allows generic USB status and control access.

Client Software To Function Relation



End-point addressing

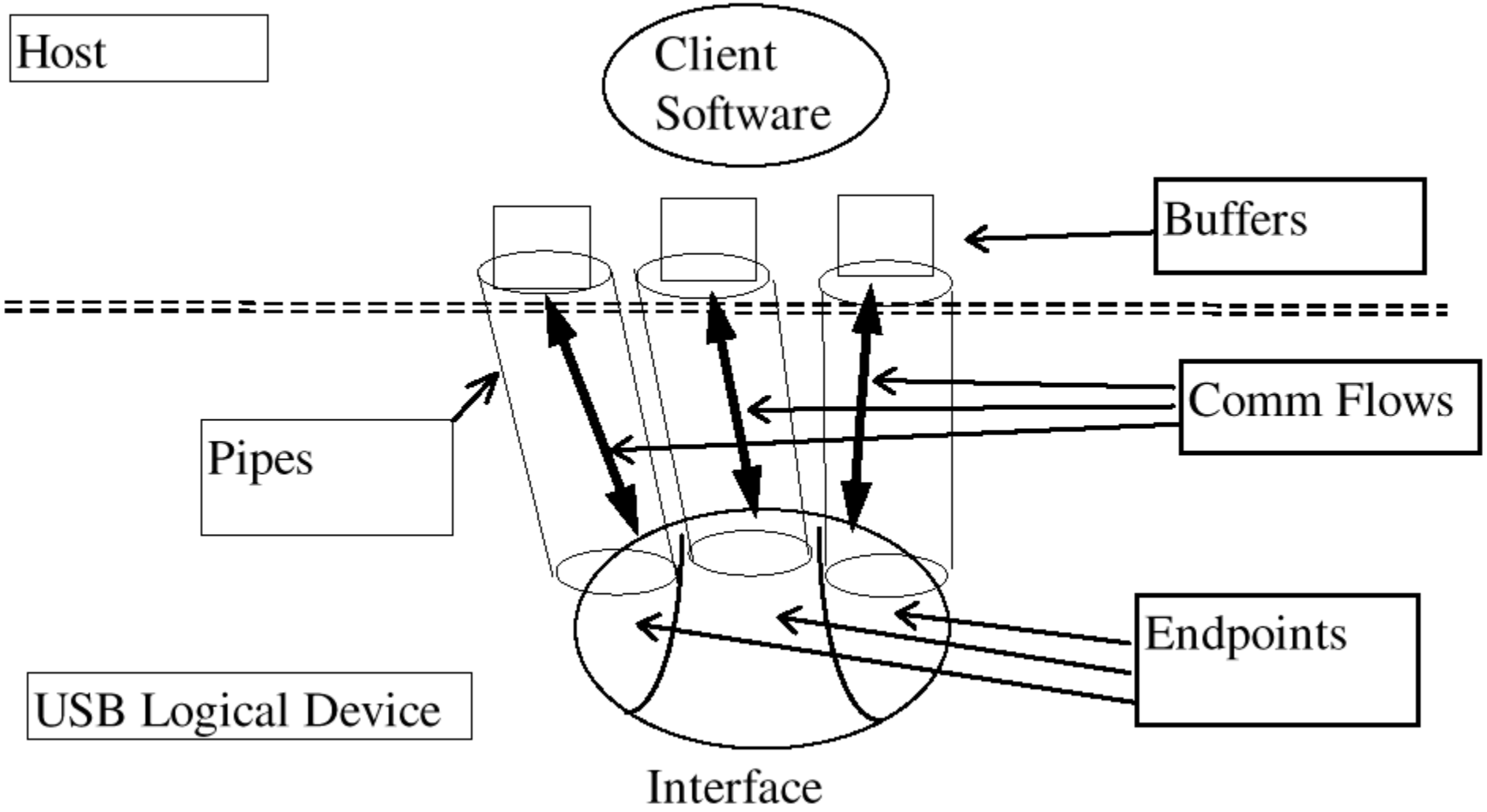
- Each logical device has a unique address assigned by the system at device attachment time.
- Each endpoint on a device has a device (design time) determined unique identifier, the endpoint number.
- A combination of the device address and the endpoint number allows each endpoint to be uniquely referenced.
- Direction is with reference to host
  - IN: from endpoint to host
  - OUT: from host to endpoint

USB End-point Properties

Endpoints describe themselves by:

- Their bus access frequency/latency requirements
- Their bandwidth requirements
- Their endpoint number
- The error handling behavior requirements
- Maximum packet size that the endpoint is capable of sending or receiving
- The transfer type for the endpoint (details later)
- For bulk and isochronous transfer types, the direction of data is transferred between the endpoint and the host.
- Endpoints must not be accessed by the host before being configured.

USB Pipes


Host
Client Software
Buffers
Comm Flows
Pipes
Endpoints
USB Logical Device
Interface

USB Pipes

- A USB pipe is an association between an endpoint on a device and software on the host.
- Pipes represent the ability to move data between software on the host via a memory buffer and an endpoint on a device.
- There are two different, mutually exclusive, pipe communication modes:
  - Stream. Data moving through a pipe has no USB defined structure.
  - Message. Data moving through a pipe has some USB defined structure.

The default pipe

- Pipes come into existence when a USB device is configured.
- Since Endpoint 0 is always configured once a device is powered, there is always a pipe for Endpoint 0. This pipe is called the Default Pipe.
- This pipe is used by system software to determine device identification and configuration requirements, and to configure the device.
- USB system software retains “ownership” of the Default Pipe and mediates use of the pipe by other client software.

Transfer types

USB defines four transfer types:

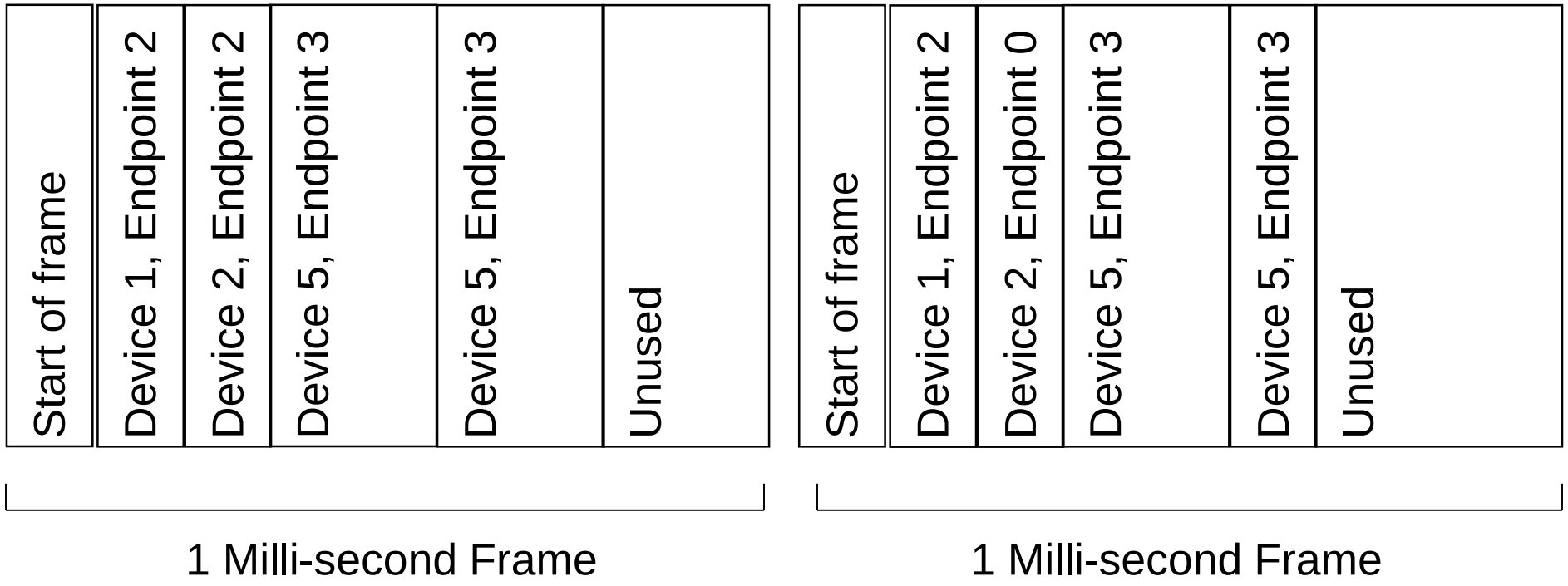
- Control Transfers - Bursty, non-periodic, host software initiated request/response communication typically used for command/status operations.
- Isochronous Transfers - Periodic, continuous communication between host and device typically used for time relevant information, e.g. Audio/video. Delivery on time is more important than integrity of data.

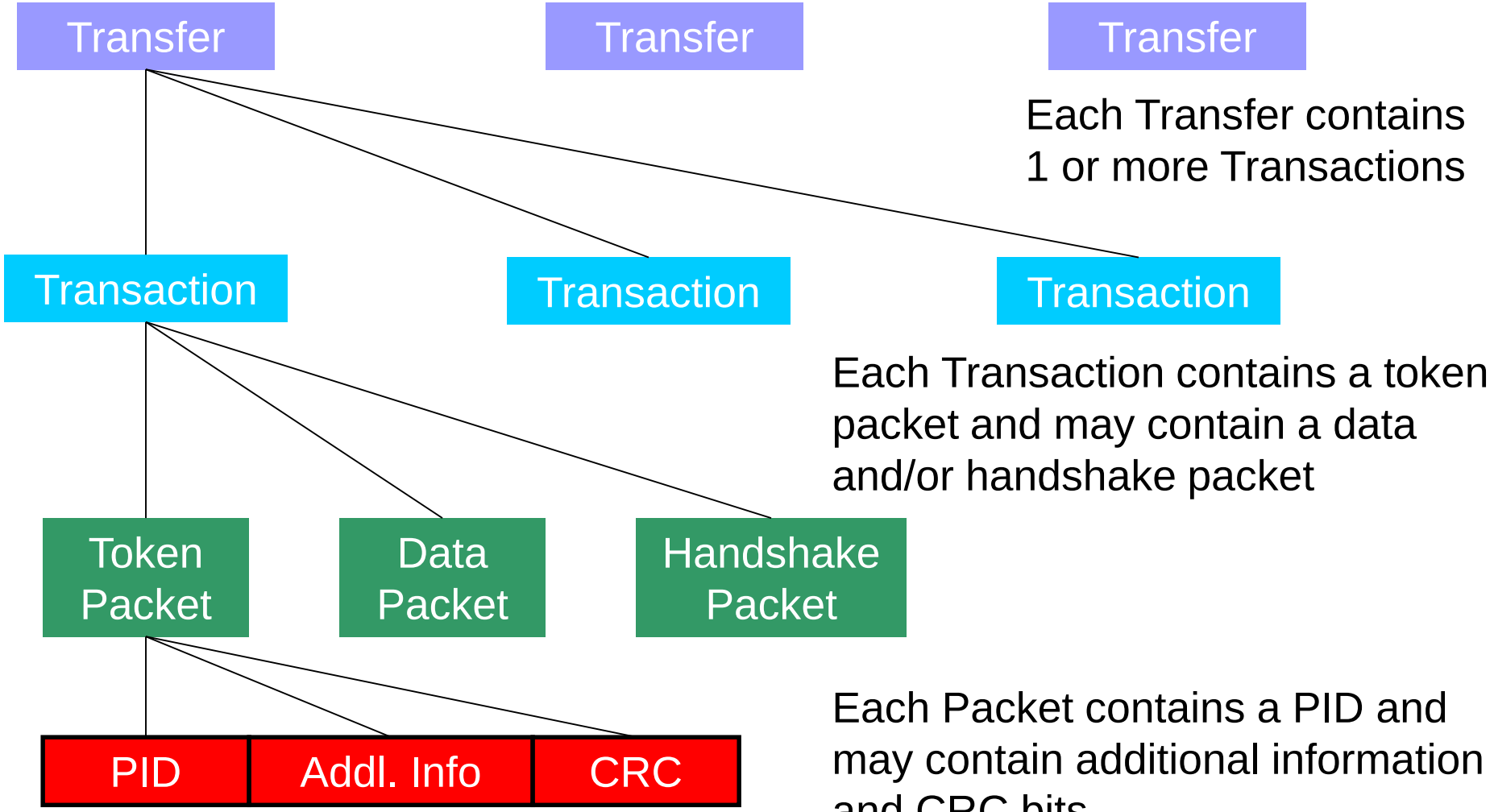
Transfer types (contd.)

- Interrupt Transfers - Small data, non-periodic, low frequency, bounded latency, device initiated communication typically used to notify the host of device service needs.
- Bulk Transfers - Non-periodic, large bursty communication typically used for data that can use any available bandwidth and also be delayed until bandwidth is available.

Frames

USB traffic is divided by host into 1 millisecond frames

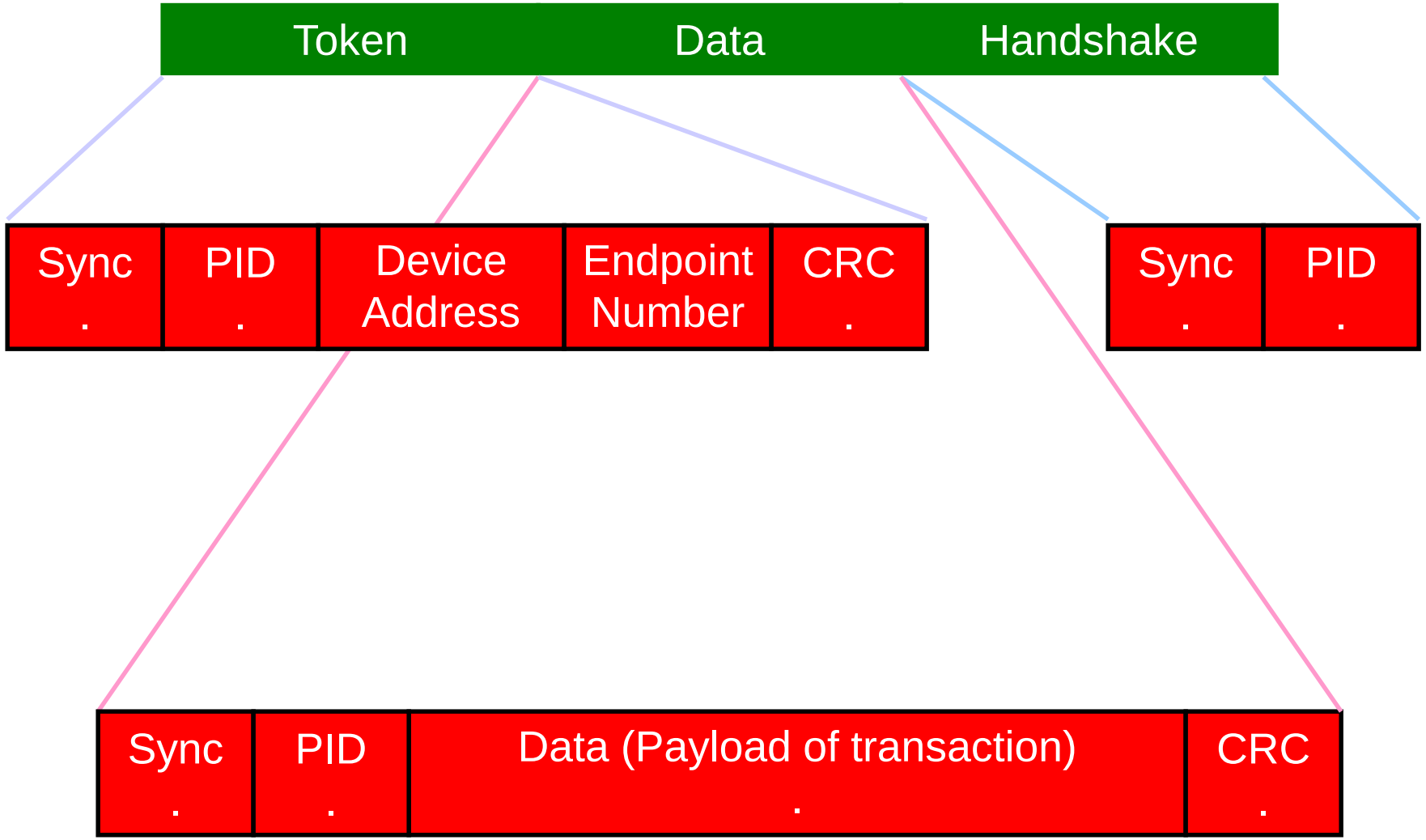




Each Transfer contains 1 or more Transactions

Each Transaction contains a token packet and may contain a data and/or handshake packet

Each Packet contains a PID and may contain additional information and CRC bits

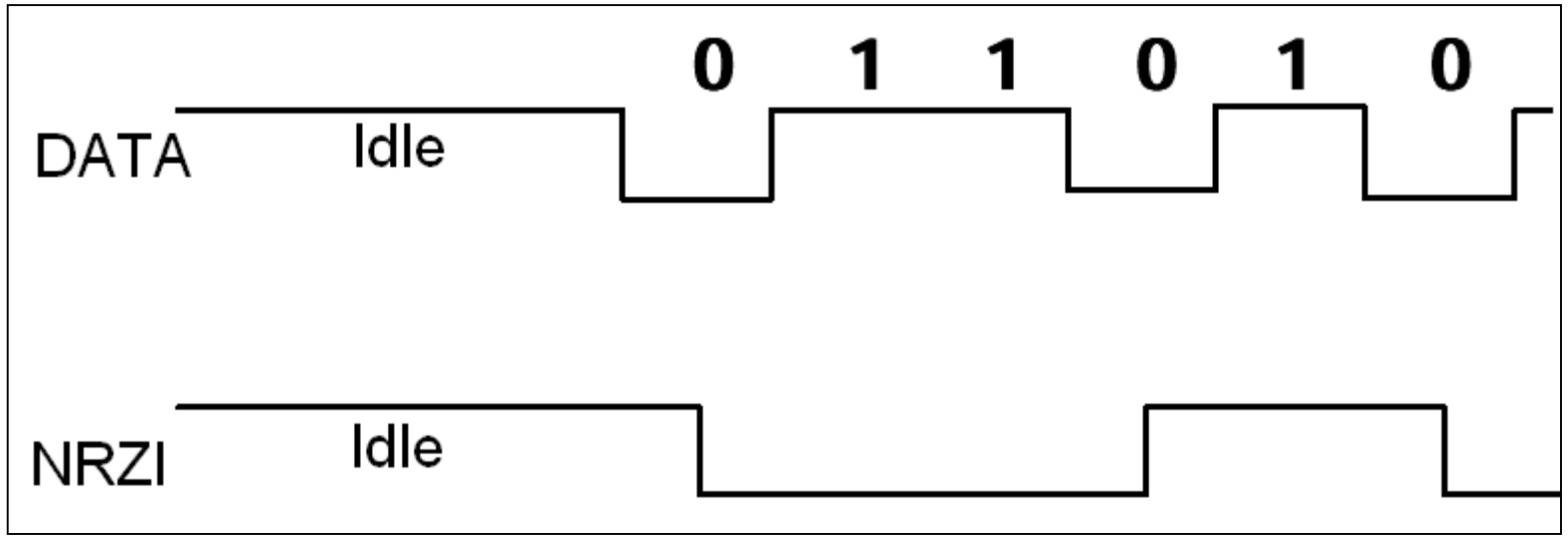

Token
Data
Handshake
Sync
.
PID
.
Device Address
Endpoint Number
CRC
.
Sync
.
PID
.
Sync
.
PID
.
Data (Payload of transaction)
.
CRC
.

Data Movement

- Data can be communicated over the USB using pipes.
- Pipes support different types of transfers
  - Message Pipes - Control Transfers
  - Stream Pipes - Bulk, Interrupt, isochronous transfers
- Each transfer may be made up of one or more transactions. (Depends on the amount of data)
- Each transaction is made up of multiple packets. (Token, data, Handshake)
- Each packet is made up of multiple fields. (PID, Address, Data, CRC)
- Bit stuffing is performed.
- NRZI encoding of the bit stuffed data is performed.
- The data is sent over the link differentially.

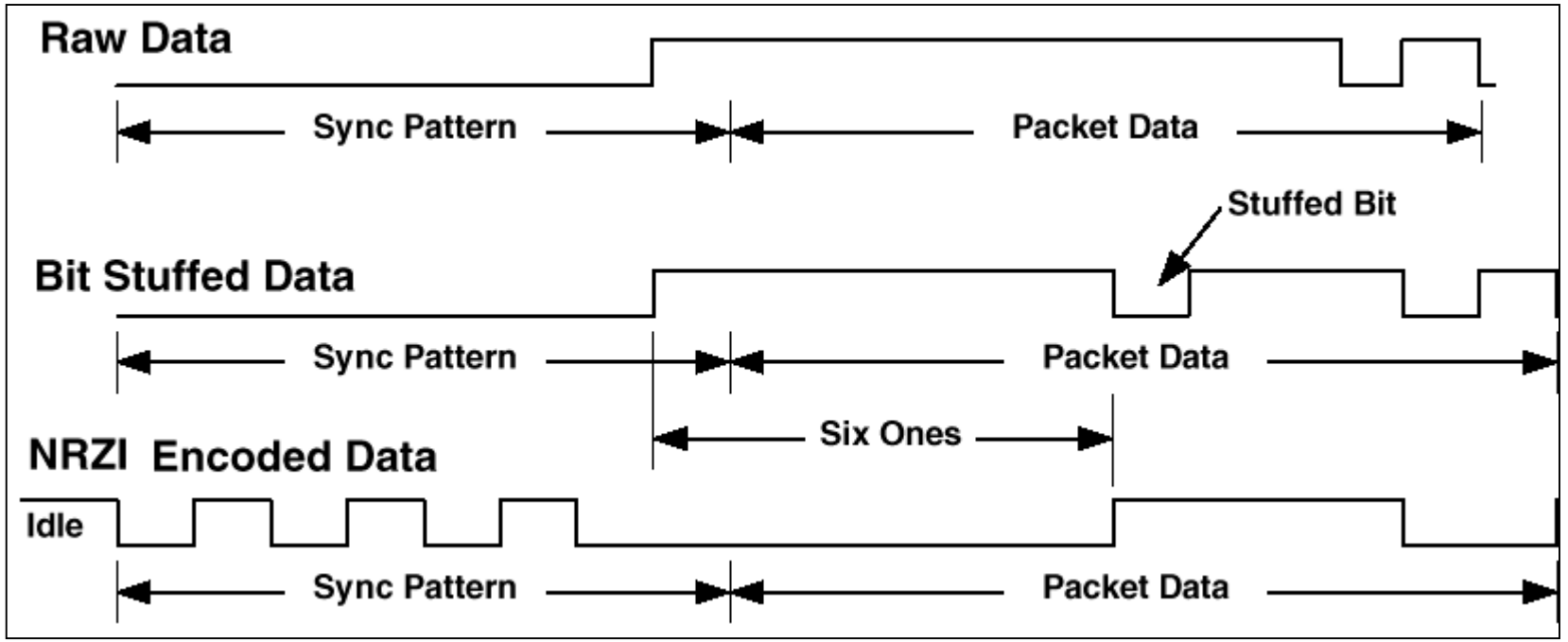
Data Encoding

NRZI encoding is used in USB



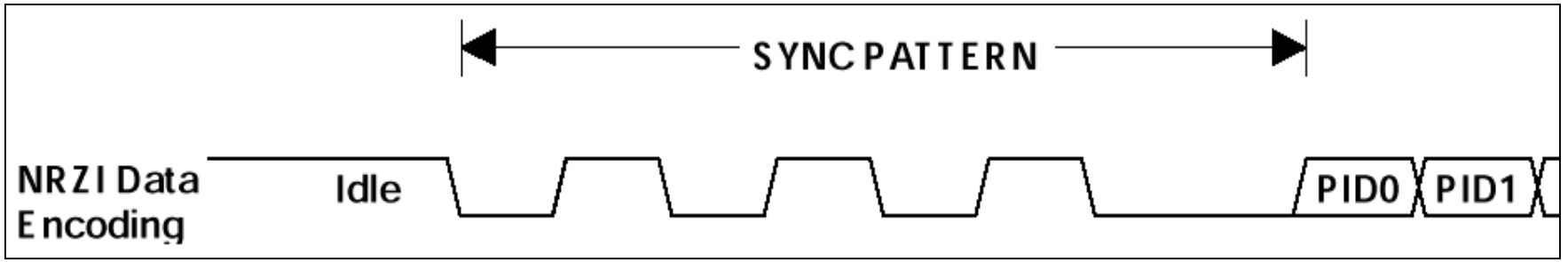
Bit Stuffing

- A 0 is inserted after every six consecutive 1's in the data stream before the data is NRZI encoded
- This forces a transition in the NRZI data stream.
- This gives the receiver logic a data transition at least once every seven bit times.



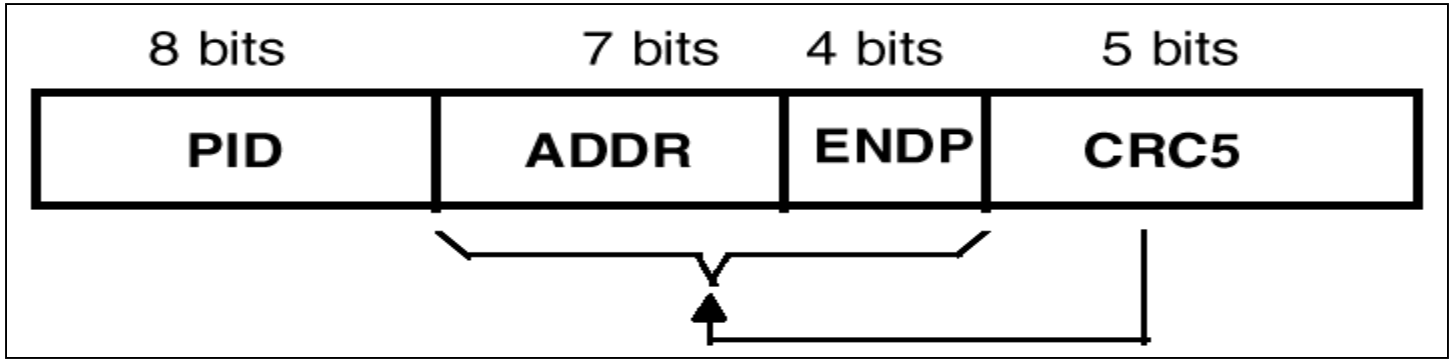
Sync Pattern

- Every packet starts with a synchronizing pattern. The pattern is equivalent to 7 zero's followed by a one. (0x80).

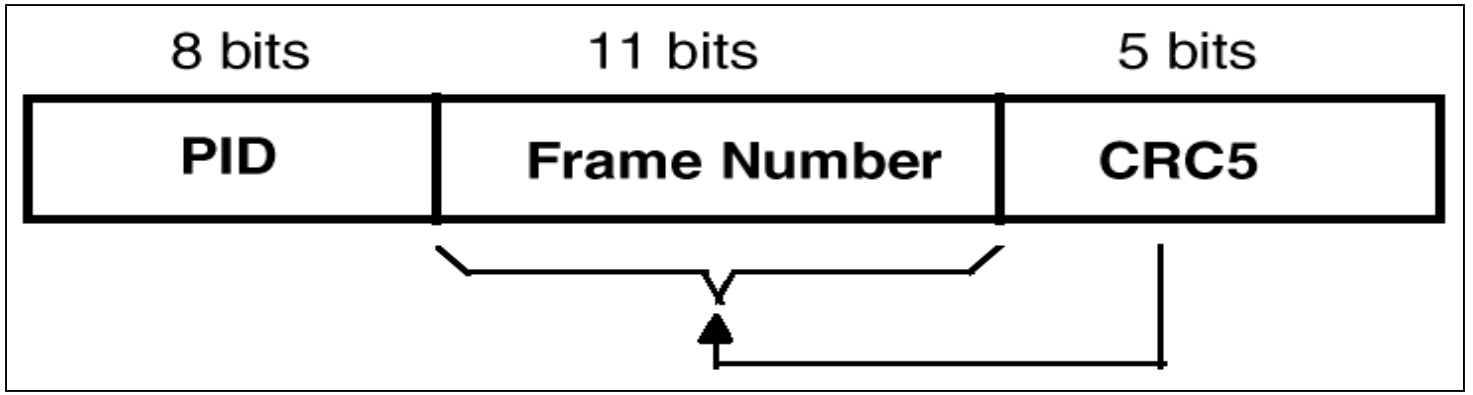


Different types of packets

Token Packet

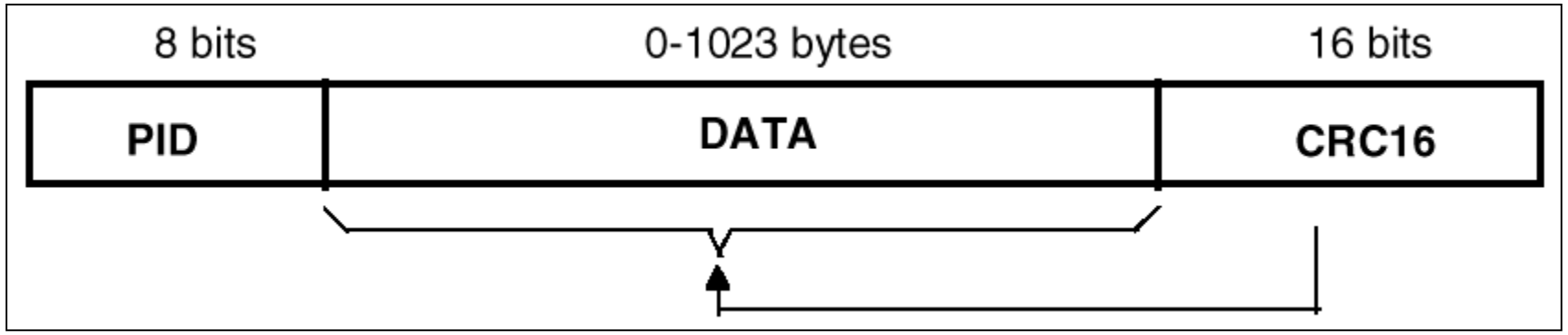


Start of Frame Packet

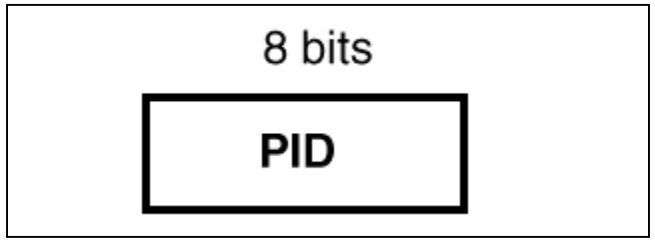


Different types of packets contd.

Data Packet



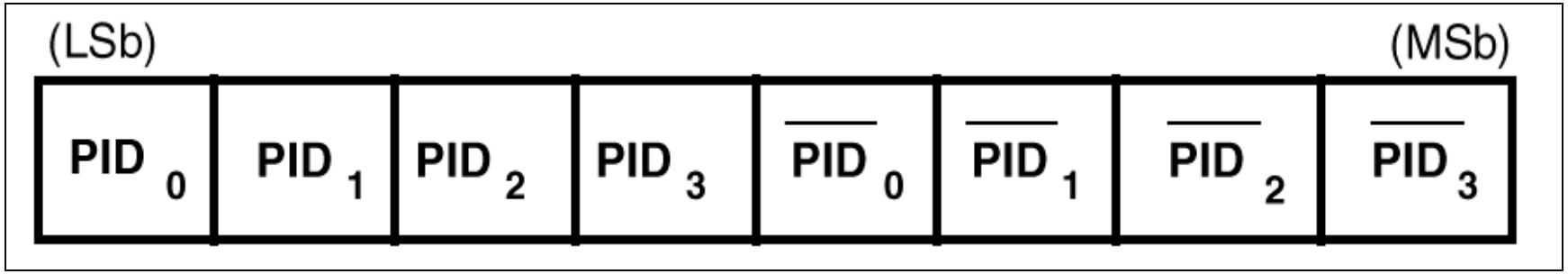
Handshake Packet



- Each packet is preceded by SYNC on the line.
- The different transfer types use one or more of the above packet type for communication.
- A sequence of packets will form one transaction.

The Packet Identifier - PID

- A PID consists of a four bit packet type field followed by a four-bit check field.
- The PID indicates the type of packet and, by inference, the format of the packet and the type of error detection applied to the packet.
- The check field of the PID insures reliable decoding of the PID. The PID check field is generated by performing a ones complement of the packet type field.

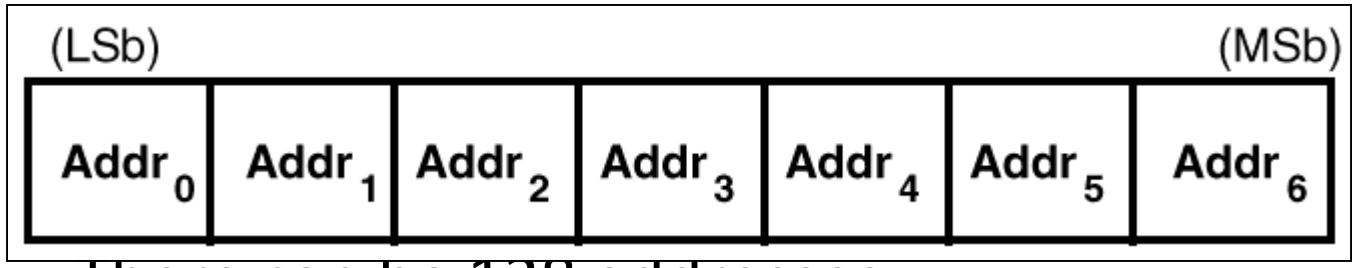


PID bits

Type	Name	Value	Description
Token	OUT	0001	Endpoint address for host-to-device transaction
	IN	1001	Endpoint address for host-to-device transaction
	SOF	0101	Start-of-Frame marker and frame number
	SETUP	1101	EP address for host-to-device setup transaction
Data	DATA0	0011	Data packet with even synchronization bit
	DATA1	1011	Data packet with odd synchronization bit
Handshake	ACK	0010	Receiver accepts error-free data packet
	NACK	1010	Receiver can’t accept data or sender can’t send data or has no data to transmit
	STALL	1110	A control request is not supported or EP is halted
Special	PRE	1100	Preamble issued by host. Enables downstream traffic to low-speed device

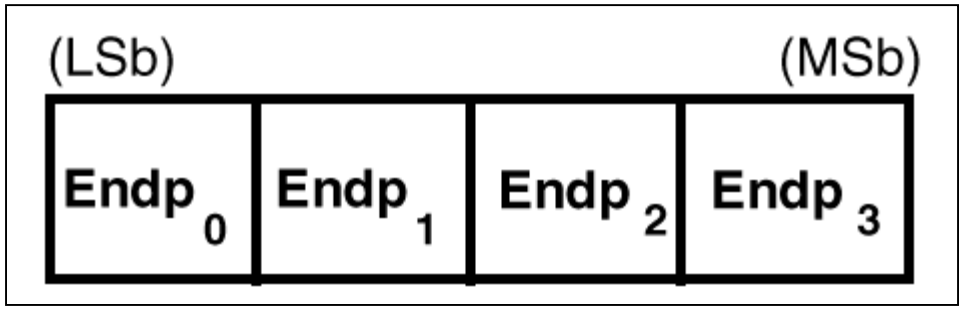
The Address field

- Function endpoints are addressed using two fields:
  - the function address field



There can be 128 addresses

  - the endpoint field

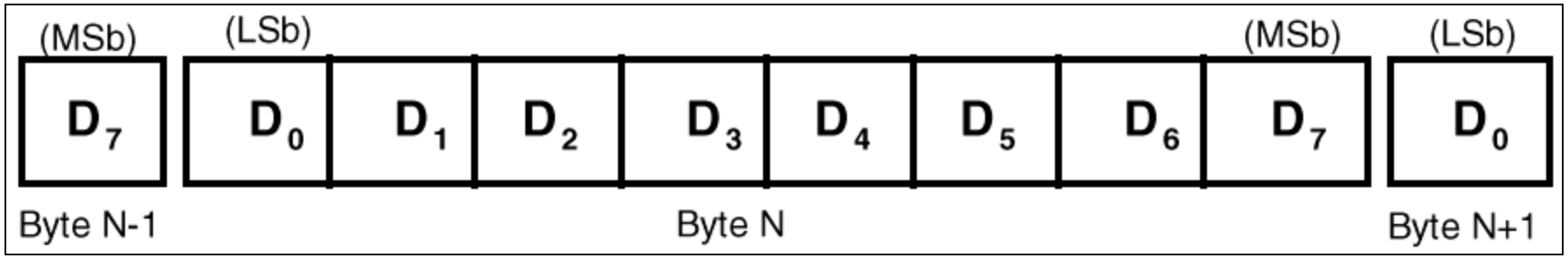


Low speed: max 2 endpoints
High speed: max 16 endpoints

- Minimum one endpoint (end point 0) is mandatory.

The data field

- The data field may range from 0 to 1023 bytes and must be an integral numbers of bytes. Data bits within each byte are shifted out LSB first.
- Data packet size varies with the transfer type as described later.



Cyclic Redundancy Checks

- Cyclic redundancy checks (CRCs) are used to protect the all non-PID fields in token and data packets.
- The PID is not included in the CRC check of a packet containing a CRC.
- All CRCs are generated over their respective fields in the transmitter before bit stuffing is performed.

CRCs

Token CRCs

A five-bit CRC field is provided for tokens and covers the ADDR and ENDP fields of IN, SETUP, and OUT tokens or the time stamp field of an SOF token. The generator polynomial is:

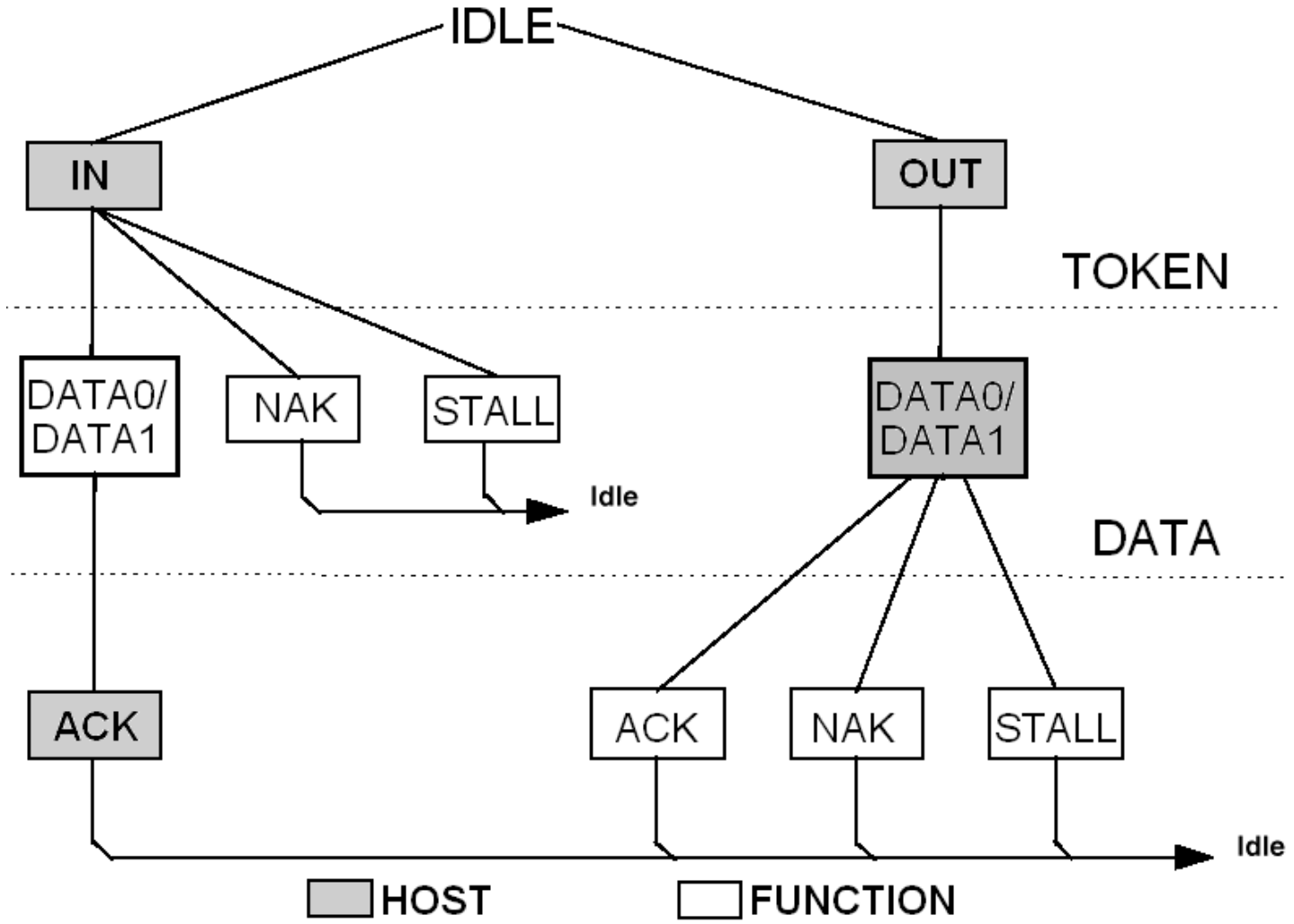
$$\mathbf{G(X) = X^5 + X^2 + 1}$$

DATA CRCs

The data CRC is a 16-bit polynomial applied over the data field of a data packet. The generating polynomial is:

$$\mathbf{G(X) = X^{16} + X^{15} + X^2 + 1}$$

Transactions for Bulk transfers



Bulk IN Transactions

- When the host wishes to receive bulk data, it issues an IN token.
- The function endpoint responds by returning either a DATA packet or, should it be unable to return data, a NAK or STALL handshake.
- A NAK indicates that the function is temporarily unable to return data, while a STALL indicates that the endpoint is permanently stalled and requires host software intervention.
- If the host receives a valid data packet, it responds with an ACK handshake. If the host detects an error while receiving data, it returns no handshake packet to the function.

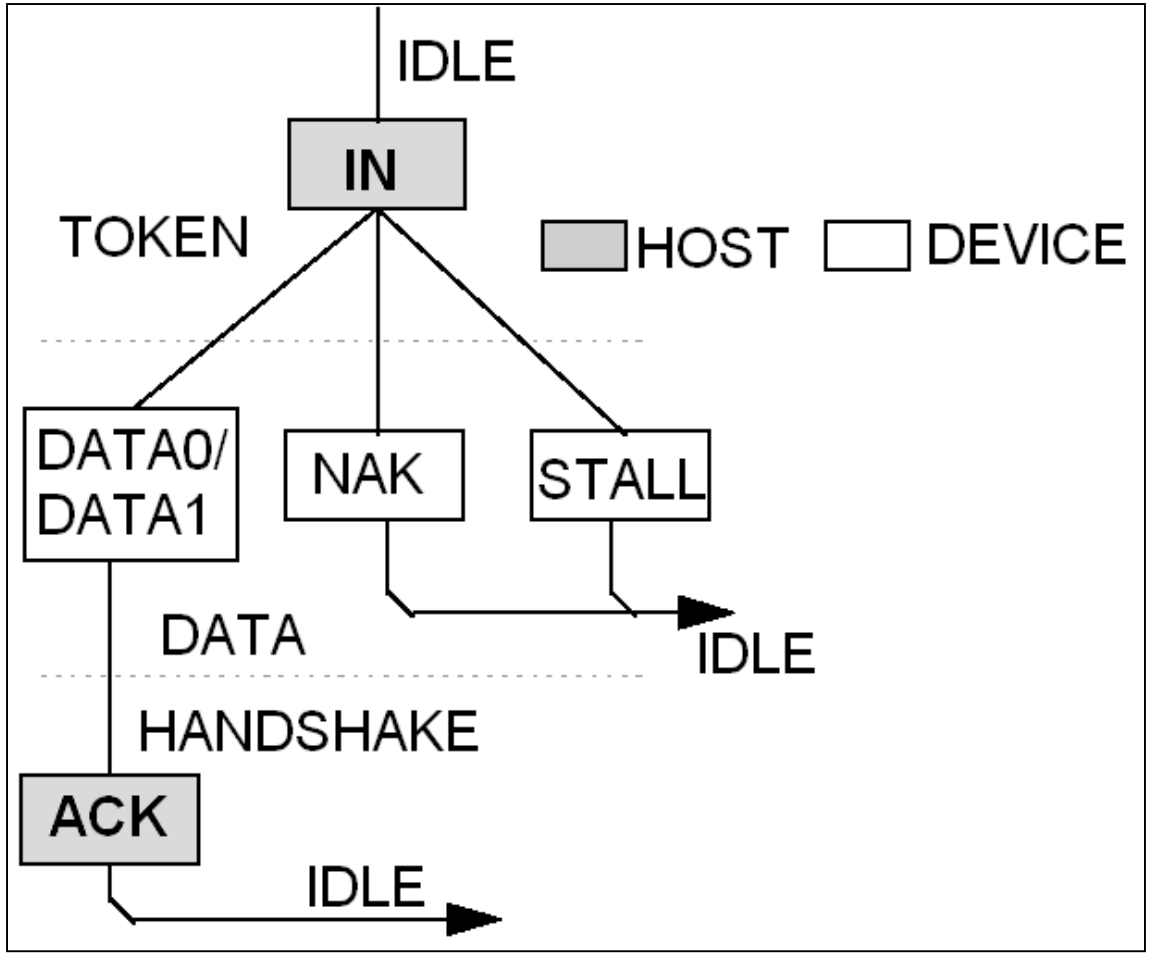
Bulk OUT Transactions

- The host first issues an OUT token packet followed by a data packet.
- The function then returns one of three handshakes. ACK indicates that the data packet was received without errors and informs the host that that it may send the next packet in the sequence.
- NAK indicates that the data was received without error but that the host should resend the data because the function cannot accept the data at this time. (e.g., buffer full).
- If the endpoint was stalled, STALL is returned.
- If the data packet was received with a CRC or bit stuff error, no handshake is returned.

Bulk Transfers

- Bulk transfers occur only on a bandwidth available basis.
- Requires a stream pipe and, therefore, always has communication flowing either into or out of the host for a given pipe. If a device requires bi-directional bulk communication flow, two bulk pipes must be used, one in each direction.
- An endpoint for bulk transfers specifies the maximum data payload size that the endpoint can accept from or transmit to the bus.
- USB defines the allowable maximum bulk data payload sizes to be only 8, 16, 32, or 64 bytes.

The Interrupt Transfers



Interrupt Transfers Contd.

- Interrupt transactions consist solely of IN.
- Upon receipt of an IN token, a function may return data, NAK, or STALL. If the endpoint has no new interrupt pending, the function returns a NAK handshake during the data phase.
- If an interrupt is pending, the function returns the interrupt information as a data packet.
- The host, in response to receipt of the data packet, issues either an ACK handshake if data was received error free or returns no handshake if the data packet was received corrupted.

Interrupt Transfers Contd.

- An interrupt pipe is a stream pipe and is therefore always unidirectional.
- The maximum allowable interrupt data payload size is 64 bytes or less for full speed. Low speed devices are limited to 8 bytes or less maximum data payload size.

Interrupt Transfers Contd.

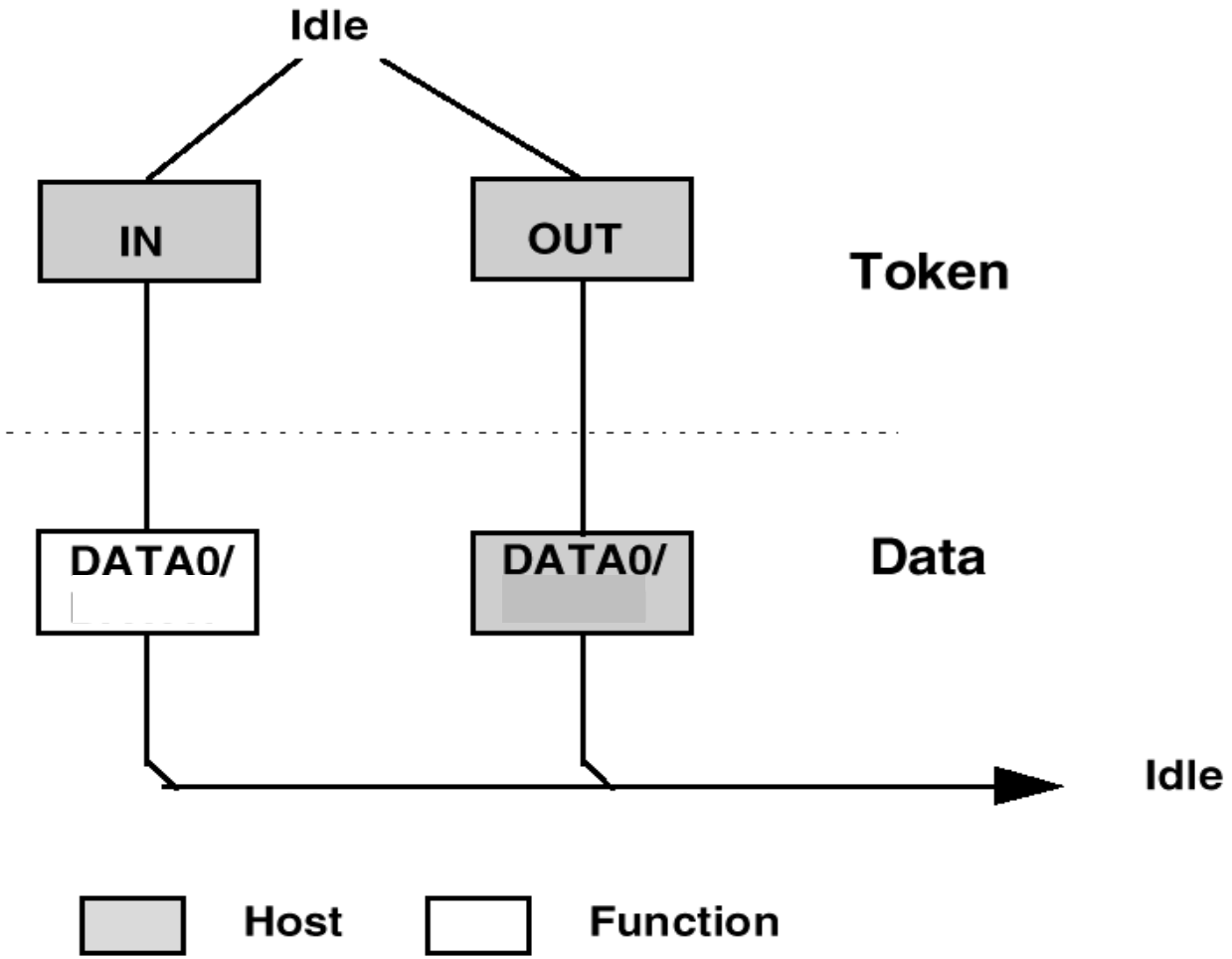
- The interrupt transfer has a guaranteed latency (like isochronous transfers).
- USB requires that no more than 90% of any frame be allocated for periodic (isochronous and interrupt) transfers.
- An endpoint for an interrupt pipe specifies its desired bus access period. A full speed endpoint can specify a desired period from 1 ms to 255 ms.
- Protocol overhead - 13 bytes

  (3-syncs, 3-pids, 2-EP+CRC, 2-CRC, 3 byte interpacket delay)

Isochronous Transfers

- In non-USB environments, isochronous transfers have the general implication of constant-rate, error-tolerant transfers.
- In the USB environment, requesting an isochronous transfer type provides the requester with the following:
  - Guaranteed access to USB bandwidth with bounded latency
  - As long as data is provided to the pipe, a constant data rate through the pipe is guaranteed
  - In the case of a delivery failure due to error, no retrying of the attempt to deliver the data

Isochronous Transfers



Isochronous Transfers

- ISO transactions have a token and data phase, but no handshake phase.
- The host issues either an IN or an OUT token followed by the data phase in which the endpoint (for INs) or the host (for OUTs) transmits data.
- ISO transactions do not support a handshake phase or retry capability.
- ISO transactions do not support toggle sequencing, and the data PID is always DATA0. The packet receiver does not examine the data PID.
- USB requires that no more than 90% of any frame be allocated for periodic (isochronous and interrupt) transfers.

Transfers Summary

Type	Stages (transactions)	Phases (packets)
Bulk	Data (IN or OUT)	Token
		Data
		Handshake
Interrupt	Data (IN only)	Token
		Data
		Handshake
Isochronous	Data (OUT or IN)	Token
		Data

Transfers Summary

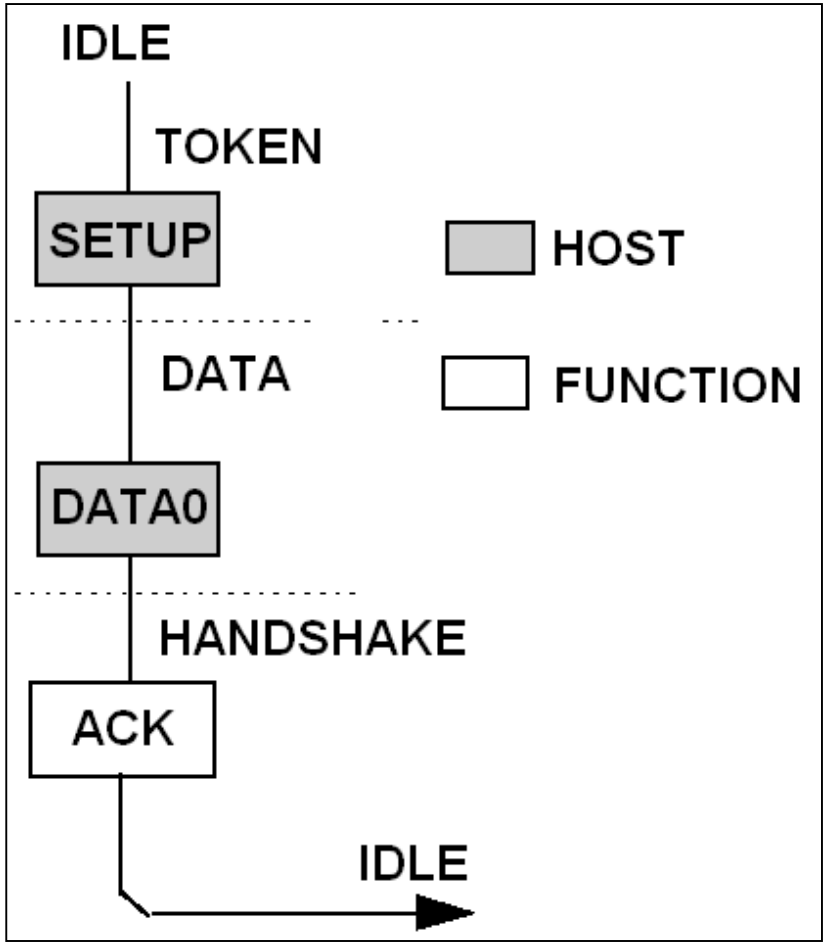
Type	Stages (transactions)	Phases (packets)
Control	Setup	Token
		Data
		Handshake
	Data (IN or OUT) (optional)	Token
		Data
		Handshake
	Status (IN or OUT)	Token
		Data
		Handshake

Control Transfers

The control transfers may have three phases:

- Setup
- Data (optional)
- Status

The diagram shows a setup transaction.

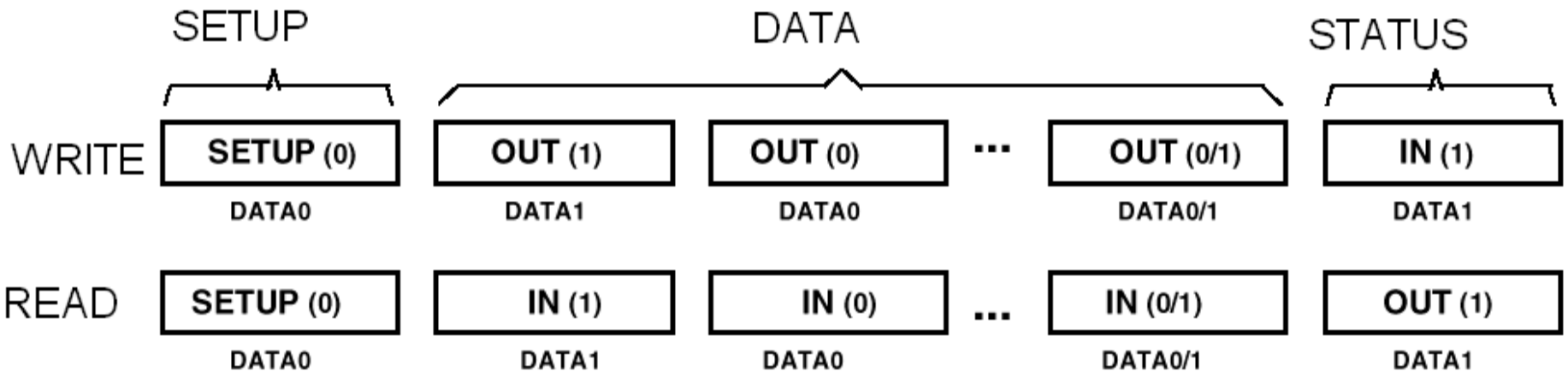


Control transfer contd.

- The Data stage, if present, of a control transfer consists of one or more IN or OUT transactions and follows the same protocol rules as bulk transfers.
- The amount of data to be sent during the data phase and its direction are specified during the Setup stage.
- The Status stage of a control transfer is the last operation in the sequence. A Status stage is delineated by a change in direction of data flow from the previous stage and always uses a DATA1 PID.
- The control transfers use the message pipes.
- The format of the data in the transactions is defined by USB standard.

Control transfer contd.

Control Write and Control Read transfer



Control transfer contd.

- Each USB device is required to implement Endpoint 0 with a control transfer type.
- Control transfers are only carried through message pipes.
- USB defines the allowable maximum control data payload sizes for full speed devices to be only 8, 16, 32, or 64 bytes. Low speed devices are limited to only an 8 byte maximum data payload size.
- A setup packet is always 8 bytes.

Bus Enumeration

- Bus enumeration is performed at startup. Since devices can be dynamically attached and removed, bus enumeration for USB is an on-going activity.
- A USB device when it is first powered or reset has a default address 00h.
- The host will poll every downstream port to find un-configured devices. It will use 0 as the destination address and configure devices one at a time. The address is assigned to the device during configuration.
- The devices on the other side of hubs are configured similarly.
- When ever a device is attached or removed the bus topology is updated by the host.

USB Device states

- Attached
- Powered
- Default
- Address
- Configured
- Suspended

USB Device states contd.

- Attached

  Device is attached to USB, but is not powered. Other attributes are not significant.
- Powered

  Device is attached to USB and powered, but has not been reset.
- Default

  Device is attached to USB and powered and has been reset, but has not been assigned a unique address. Device responds at the default address.

USB Device states contd.

- Address

  Device is attached to USB, powered, has been reset, and a unique device address has been assigned. Device is not configured.

- Configured

  Device is attached to USB, powered, has been reset, has unique address, is configured, and is not suspended. Host may now use the function provided by the device.

- Suspended

  Device is, at minimum, attached to USB, has been reset, and is powered at the minimum suspend level. It may also have a unique address and be configured for use. However, since the device is suspended, the host may not use the device's function.

USB configuration

- Each USB device provides information about itself to the host. It includes things like number of endpoints, max packet sizes, power requirements, etc.
- The USB configuration takes place through control transfers and setup packets.
- The default endpoint 0 is used for configuring USB devices.

Standard device requests

Some of the standard device requests are:

- GET_STATUS
- CLEAR_FEATURE
- SET_FEATURE
- SET_ADDRESS
- GET_DESCRIPTOR
- SET_DESCRIPTOR
- GET_CONFIGURATION
- SET_CONFIGURATION
- GET_INTERFACE
- SET_INTERFACE
- Further details available in the USB specification.

Error detection and recovery

- USB employs three error detection mechanisms: bit stuff violations, PID check bits, and CRCs.

FIELD	ERROR	ACTION
PID	PID check	Ignore packet
Address	Address CRC	Ignore Packet
Frame Number	Frame number CRC	Ignore frame number field
DATA	Data CRC	Discard data

Bit stuffing error is common to all.

18 bit times have to elapse before any timeout.

Things not covered

- Power management
- Detailed framework for devices, hubs and the host
- Details of configuration of devices.
- Detailed framework for isochronous transfers.
- Error recovery
- Others

Reference

- The USB specifications
  - http://www.usb.org/developers/docs/usb_20.zip
- USB System Architecture
  Don Anderson
  MindShare, Inc.
  ISBN 0-201-46137-4
- USB Complete
  Jan Axelson
  LakeView Research
  ISBN 0-965-08193-1

Thank You

ashishk @ cdac.in