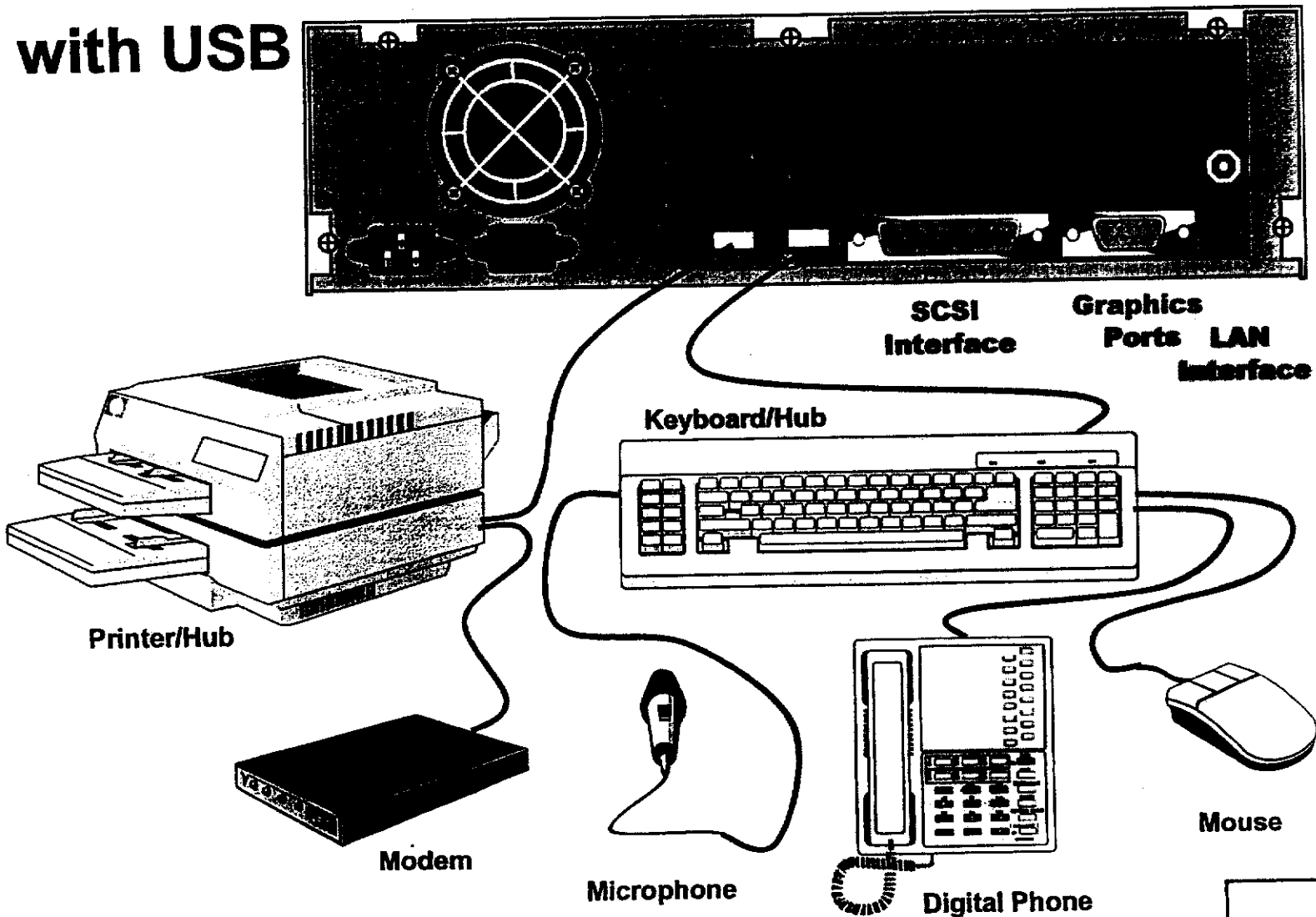
USB Introduction

What is USB?

- USB is a standard peripheral bus designed for desktop peripherals with a performance/cost point for todays periherals

- USB is developed by PC and telecom industry leaders:
  Compaq, DEC, IBM, Intel, Microsoft, NEC and Northern Telecom

PC with USB



Features of USB

- USB features one universal plug type for all USB peripheral to PC connection
- Eliminates install add-in cards, set DIPswitches, configure IRQ and I/O address
- Up to 127 Peripherals can run simultaneously on a computer

  Printer, Scanner, Keybord, Monitor, Speakers, Camera, PC telephone, Joysticks, ...
- Peripherals are automatically configured as soon as they are physically attached
- USB distributes power to many peripherals, no external power supply boxes needed
- Suspend and resume signaling are used to rduce power consumption of buspowerd devices

USB simplifies the attachment and configuration of peripheral devices

Plug and Play

Just plug them in and turn them on

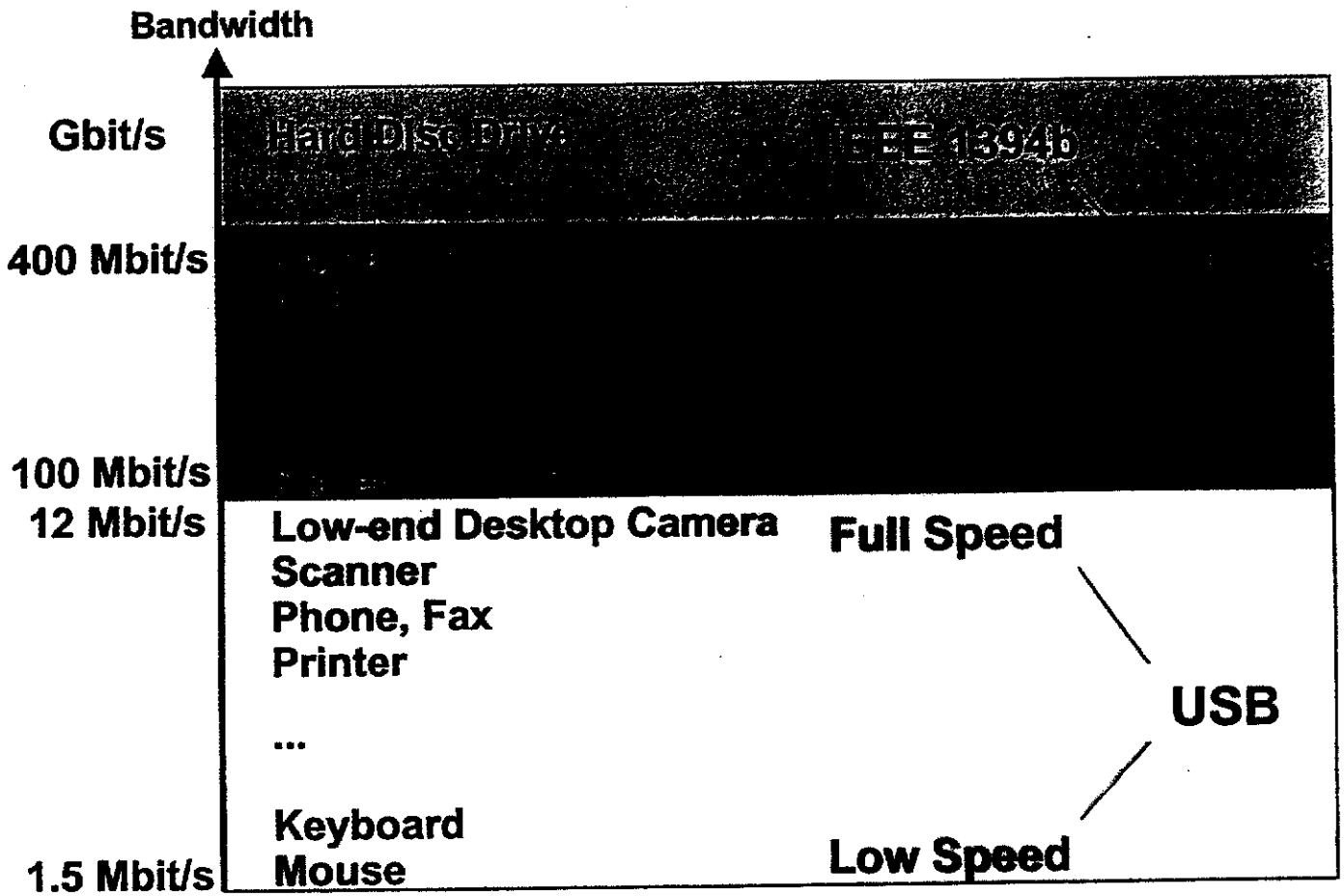
Hot Swapping

No shut down and restart when attach new peripherals

Target Applications for USB

- USB is targeted for low and medium speed applications, like mice, keyboard, modems, low speed cameras,...
- Devices that require up to 4 Mbps of bandwith constantly
- voice data can tolerate bit error rate but cannot tolerat delay
- data for filetransfer require high accuracy in delivery

Next Step



USB Implementers Forum

➪ the USB Implementers Forum was founded in 1995 by:

Compaq, Digital Equipment Corporation, IBM PC Company, Intel, Microsoft, NEC, Northern Telecom

➪ Tasks of the USB Implementers Forum:

- definition of USB Specifications
- distributing of Vendor-ID's to USB IF member
- conferring the USB trade-mark
- administration of USB activities (conferences, exhibitions, etc.)
- support of USB relevant information via WWW (www.usb.org/developers)

1998: 500 members

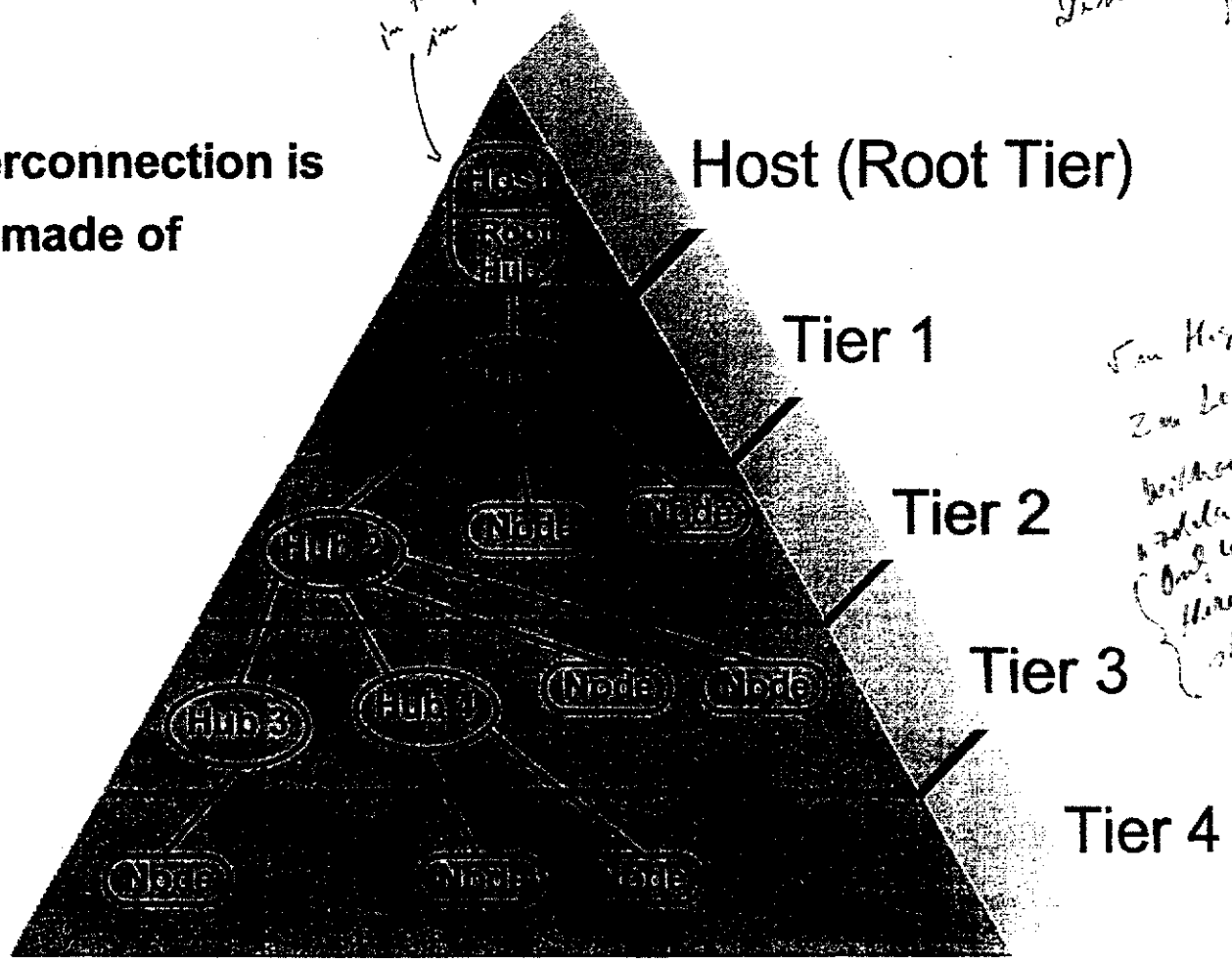
2,000 $ annual subscription for members

MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

USB Topology

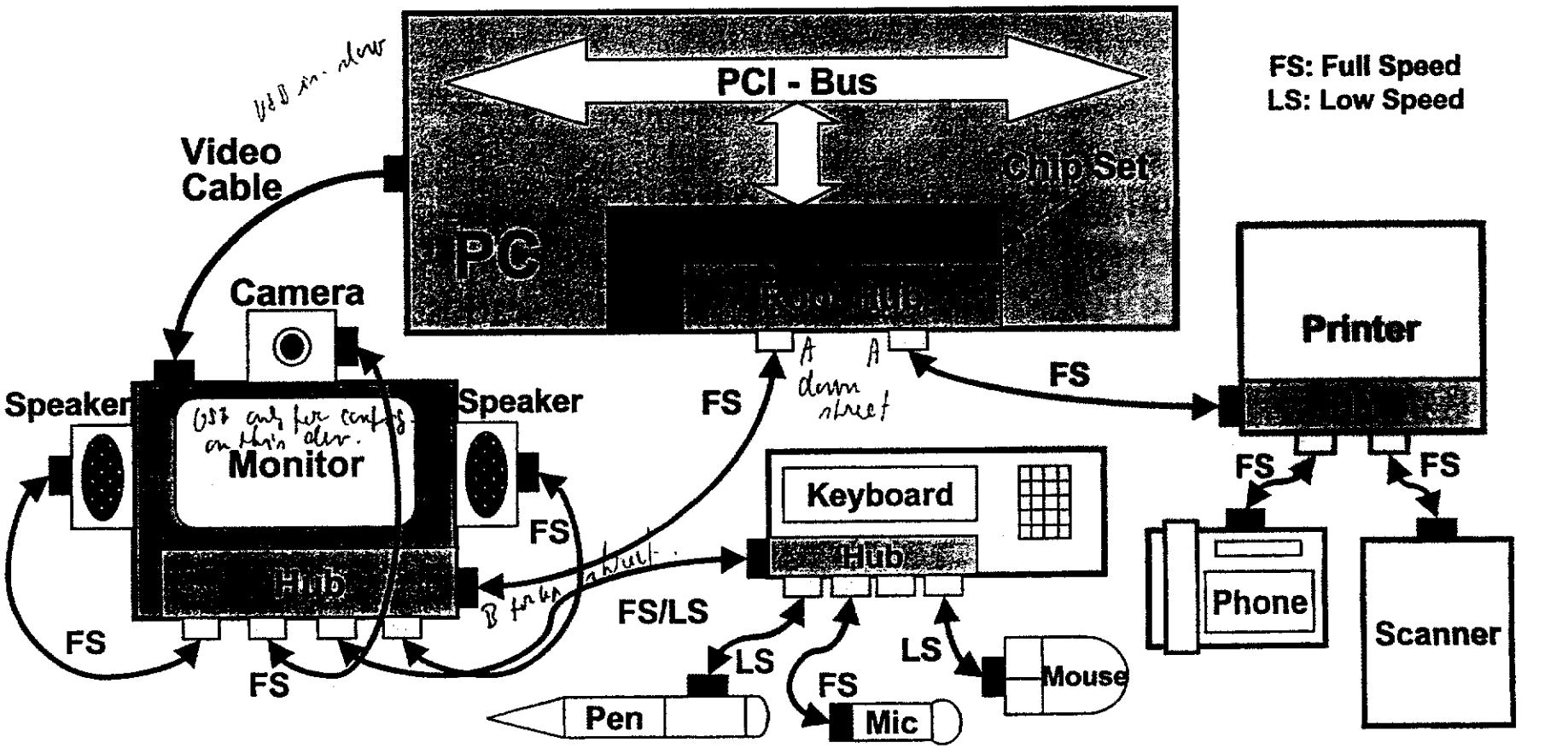
The USB physical interconnection is a tiered star topology made of

- ➪ **one Host,**
- ➪ **Hubs and**
- ➪ **Nodes (Functions)**



MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

The USB Bus - a Communication Interface for Desktop Applications

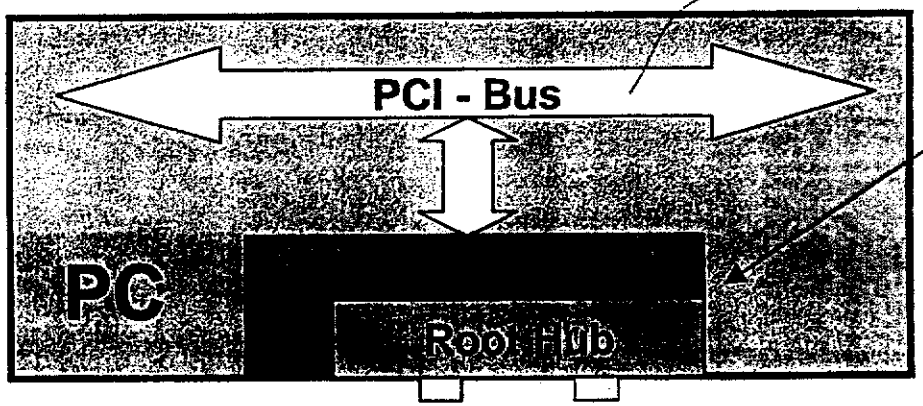


Functions of a USB Host

All Communication on the USB bus is processed under the host software control (hard- and software located inside of the PC).
There is only one host in the system.

The host hardware consists of:

- **USB host controller** (controls and schedules all activities **in the system**) and
- **USB root hub** (provides attachment to ports of the USB **bus)**



Functions of a USB Host

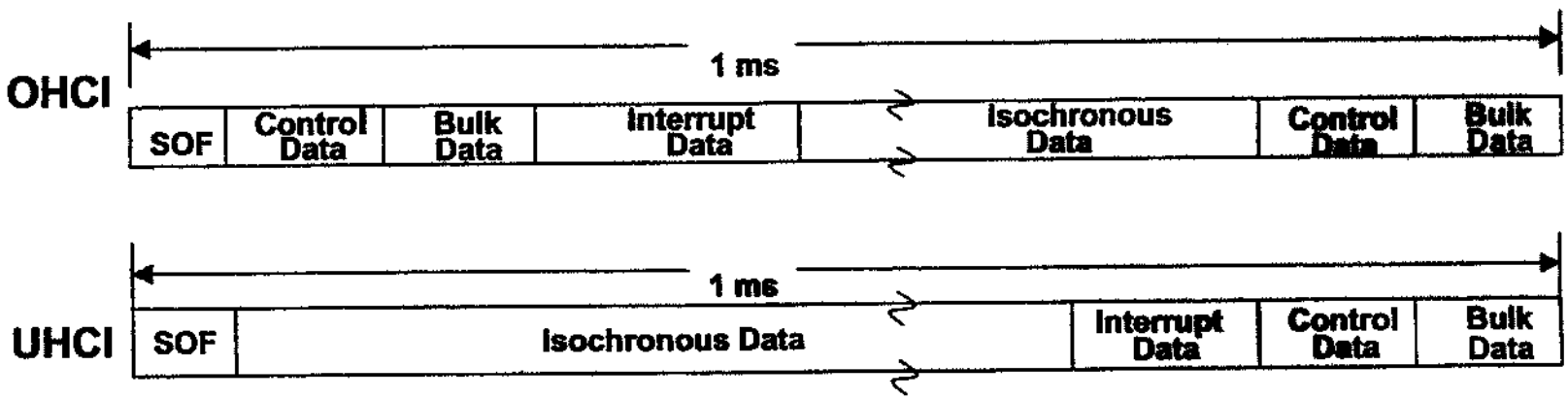
The functions of a **USB Host Controller** are:

- the control of **all data flow**:
  transmit of data to a Hub / Function (USB Device)
  request of data from a Function (USB Device)
- **conversion** of parallel data to serial and vice versa
- control of **enumeration process** of hubs / functions
- control of **power management:**
  supply of bus powered Hub
  control suspend status of USB Device
  control of power consumption of
  connected Hubs / Devices

USB Host Implementations

There are two different implementations for a USB Host possible:

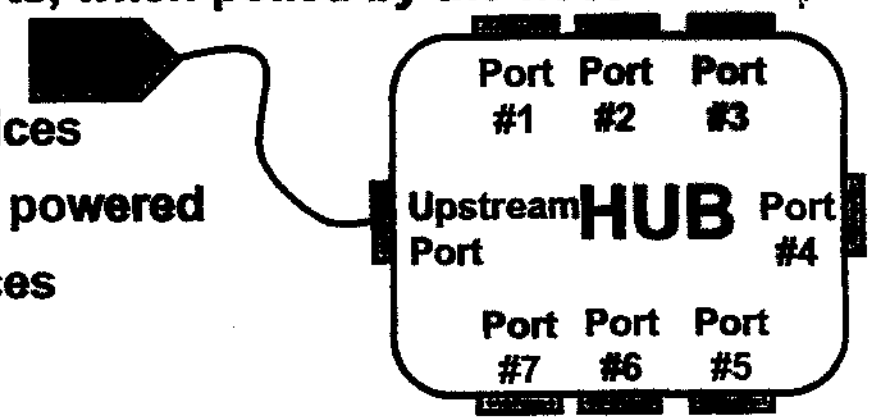
- **OHCI** Open Host Controller Interface
  specified by Compaq, Microsoft, Northern Telecom
- **UHCI** Universal Host Controller Interface
  specified by Intel
- They differ in the format of the 1 ms frame on the USB bus



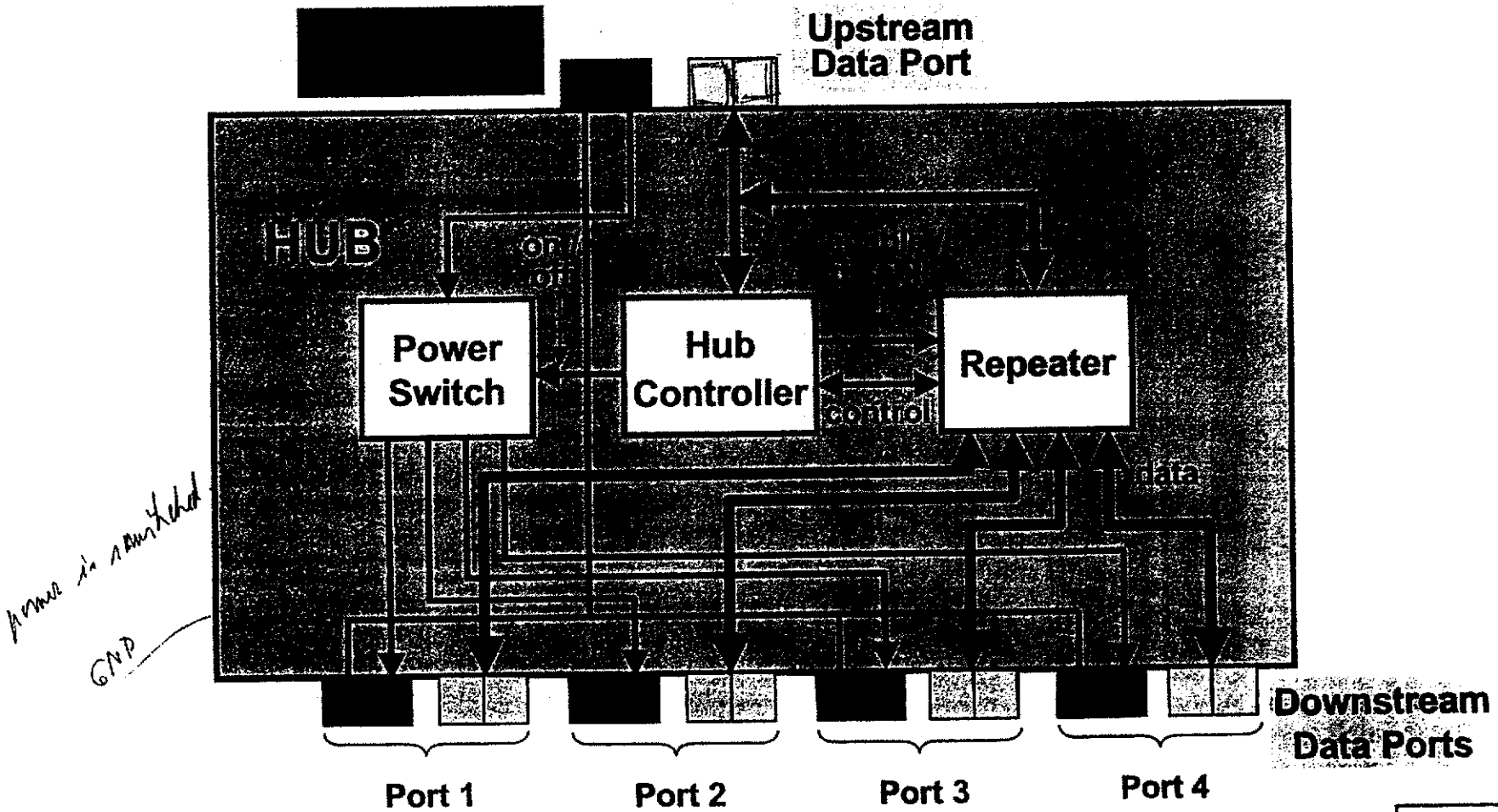
MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

Functions of a USB Hub

- recognition of attached / detached USB devices on the USB bus (down stream)
- enable disable of downstream ports
- distributing data downstream (forwarding of data to enabled ports)
- concentrating of data upstream
- control of power management of it's USB ports
- reporting status events on it's ports, when polled by the host software
- detection of full or low speed devices
- Hubs may be self powered or bus powered
- Hubs are always Full Speed Devices



Block Diagram of a USB Hub



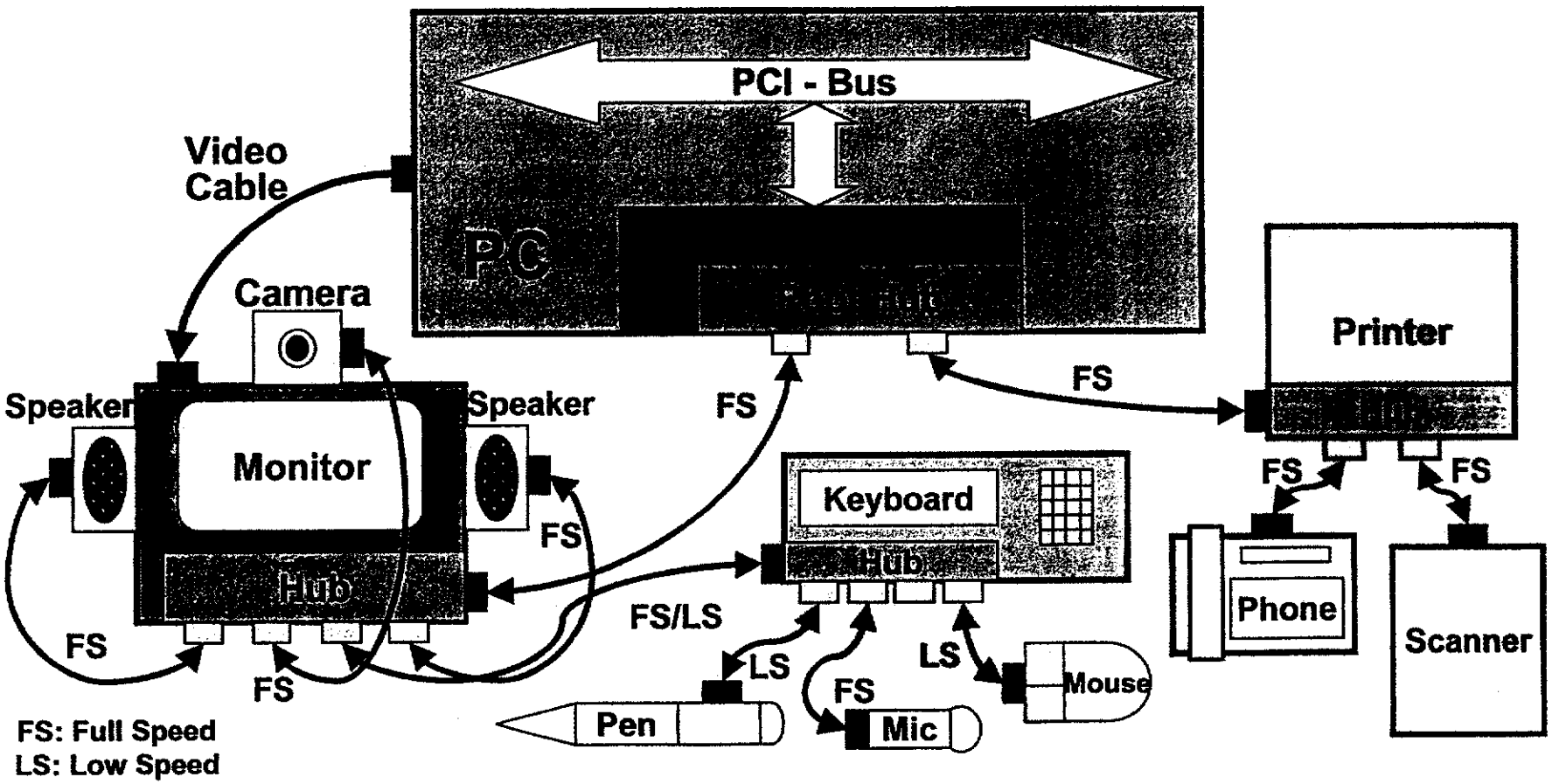
Functions of a USB Function

The functions of **USB Functions** are:

- terminal devices to a user function, human interface
- linked to hubs or directly to the root hub
- running with **low speed:** 1.5 Mbit/s or
  high speed: 12 Mbit/s
- USB Functions are: **self powered** or
  bus powered

  bus powered functions may be **low powered**: I_{max} = 100 mA
  high powered: I_{max} = 500 mA
- supports **Plug & Play** feature (hot plug in / hot plug out)
- cable for low speed may be **shielded / unshielded**, **twisted** pair or parallel, maximum length 3 m
- cable for high speed: must be **shielded** and **twisted** pair, maximum length 5 m

The USB Bus - a Communication Interface for Desktop Applications



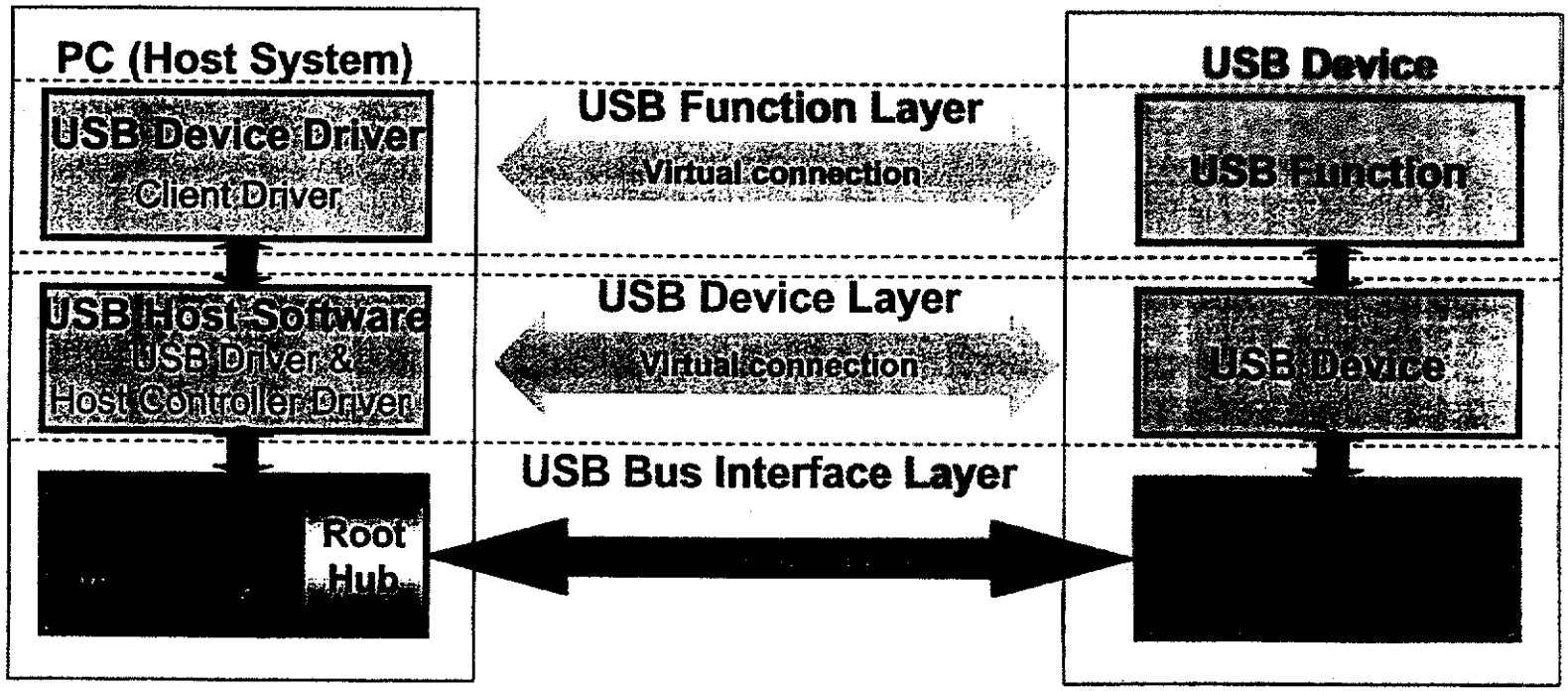
USB Device Classes

- Devices with similar attributes and services are combined to a Device Class
- Device Classes permit a device driver design that manipulates a set of devices
- USB Devices are described by a set of Standard Descriptors and additional Device Class Specific Descriptors
- Device Class Specific Descriptors provide the USB Device Driver with the information required to handle the device
- USB Devices support the standard requests defined in the USB Specification and class specific requests defined in the USB Device Class Specification
- USB Device Class Definitions describes specific attributes and characteristics that devices within a class may support

MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

The Communication Layers

- Device Class definitions relates to the functional layer
- Device Class Specific Descriptors provide the USB Device Driver with the information required to handle the device



Device Class Working Group

- the Device Class Working Group of the USB Implementers Forum has already defined Device Classes and the Device Class Specific Descriptors for:
  - **Mass Storage Device Class**
  - **Human Interface Device Class, HID**
  - **Printer Device Class**
  - **Power Device Class**
  - **Monitor Device Class**
  - **Audio Device Class**
  - **Communication Device Class**

USB Cable - Peer to Peer Connection

USB cable are used to connect USB hubs and USB devices to the USB Root Hub.
The cable is used to transmit data and to supply bus powered devices with power.

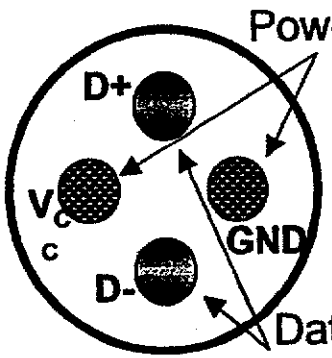
There are two different types of USB cable:

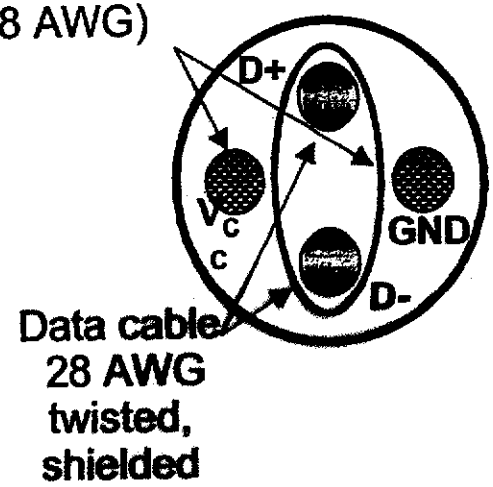
- ➪ Full Speed cable: maximum length 5 m
  one shielded twisted pair for data exchange
  one pair for power supply

- ➪ Low Speed cable: maximum length 3 m
  4 wires: two unshielded, untwisted for data exchange
  two for power supply

USB Cable for Low Speed and High Speed

Line	Color
V_{CC}	red
D-	white
D+	green
Ground	black





USB Cable Minimum Requirements

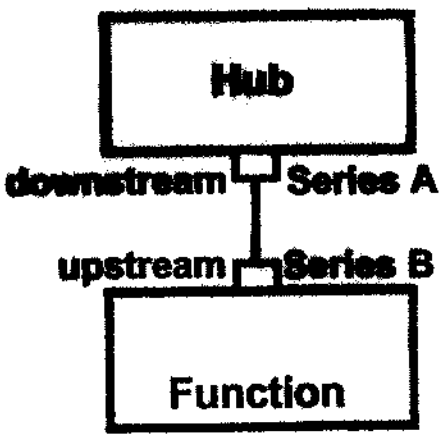
Cable Type AWG	Cable Impedance [W/m]	maximum Length [m]
28	0.232	0.81
26	0.145	1.31
24	0.091	2.08
22	0.057	3.33
20	0.036	5.00

USB Connectors

USB specifies two types of connectors: Series A connector
Series B connector

In some cases there is the cable directly connected to the USB device (e.g. mouse, keyboard).

Contact No.	Signal	Color
1	VCC	red
2	-Data	white
3	+Data	green
4	Ground	black



Four variants of connectors are available:

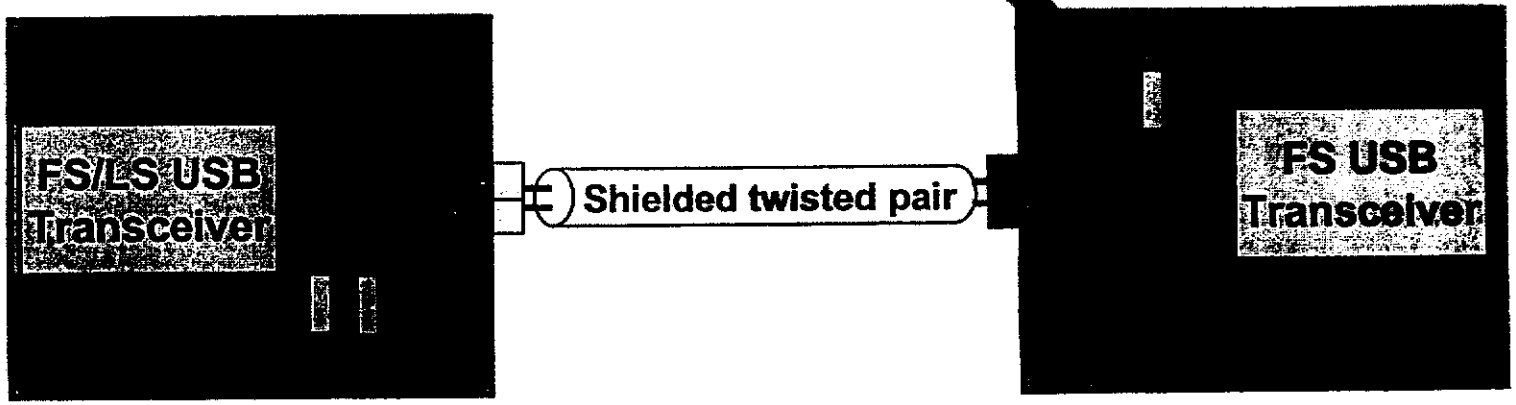
Vertical mount
Right angle mount
Stacked right angle mount
Panel mount

USB Transmission Rate

USB transmission rate can be Full Speed FS (12 Mbit/s) or Low Speed LS (1.5 Mbit/s)

⇨ Full Speed FS

- ✓ detection of full speed with **1.5 kΩ pull up** connected to +3.3V at the line **D+** on the function side
- ✓ maximum length is **5 m**
- ✓ transmission rate is 12 Mbit/s

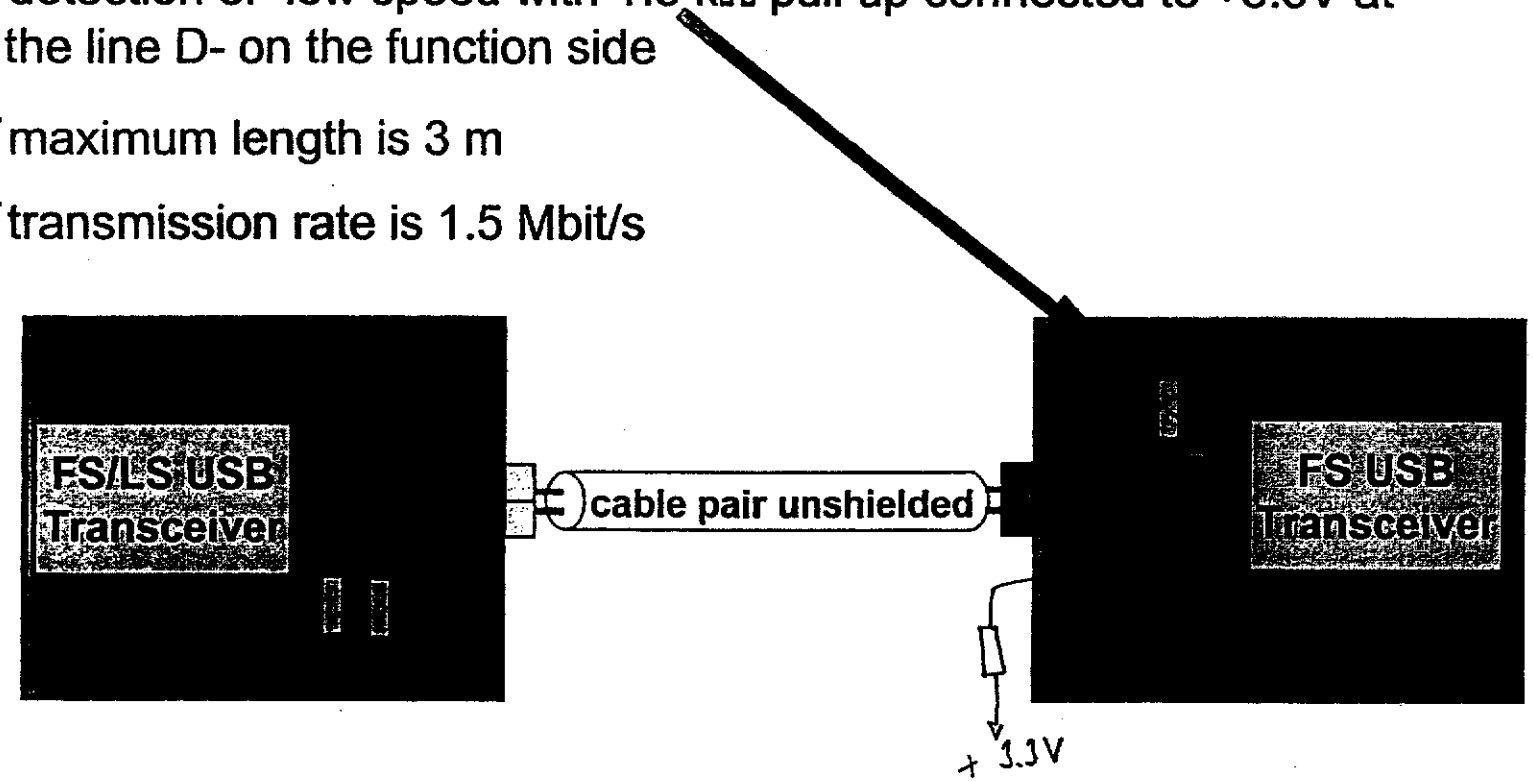


MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

USB Transmission Rate

⇨ Low Speed LS

- ✓detection of low speed with 1.5 kΩ pull up connected to +3.3V at the line D- on the function side
- ✓maximum length is 3 m
- ✓transmission rate is 1.5 Mbit/s

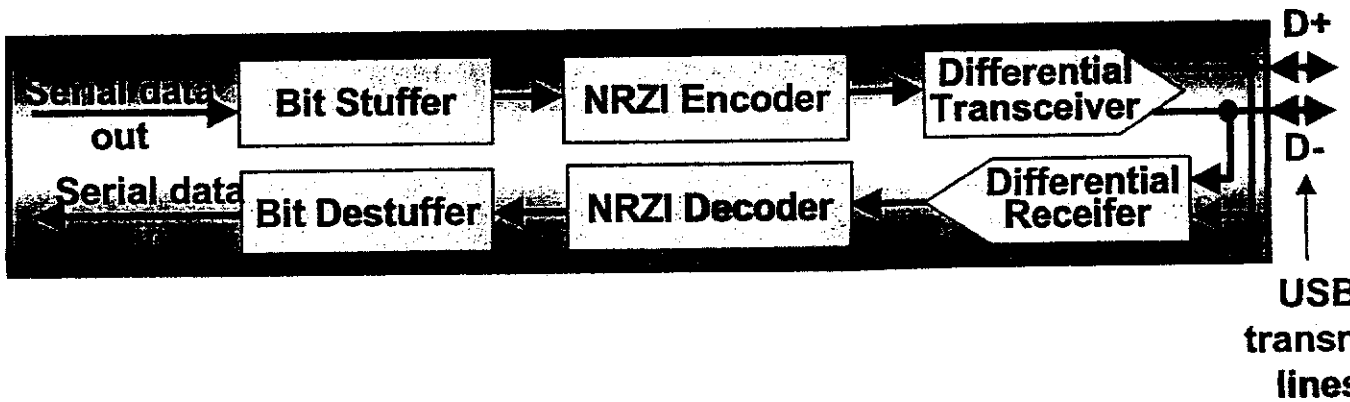


MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

Data Transmission on a USB

Data transmission on a USB Bus is processed with:

- ➪ Bit Stuffing
- ➪ NRZI Coding
- ➪ Differential Signals



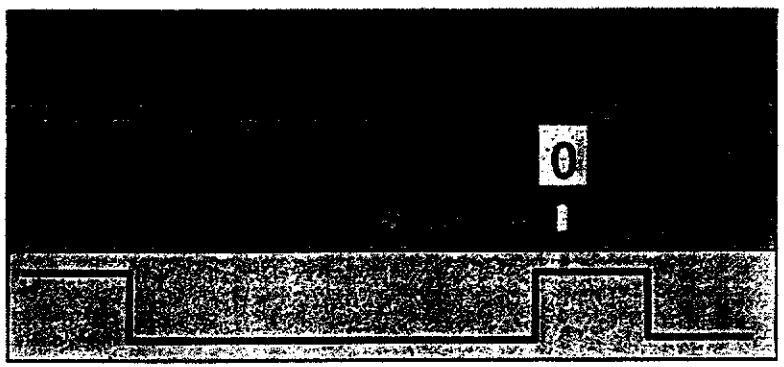
Bit Stuffing

- After six bits on "1" there is one zero bit injected to the data stream
- Receiver expects after 6 received bits on "1" a bit on "0" and eliminates it

Bit sequence (serial data)

Data with bit-stuffing

NRZI coded data
(data on the bus line)



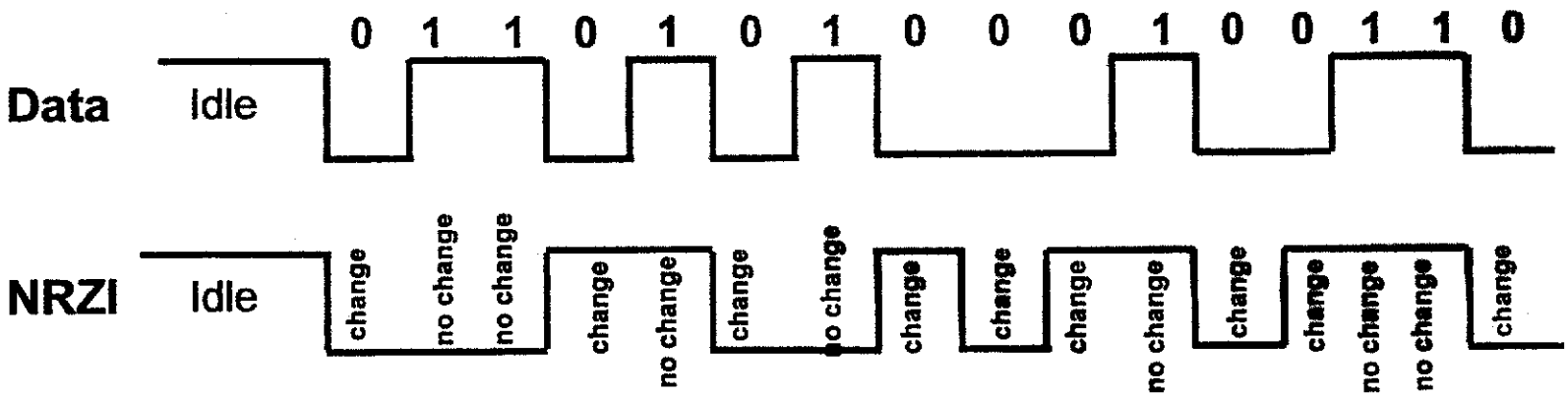
MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

NRZI Coding

Non Return to Zero - Inverted NRZI

Coding is used to ensure integrity of data delivery, without requiring a separate clock signal be delivered with the data

- **Data bit = 0** changes bit level in the USB data stream
- **Data bit = 1** no change in data stream



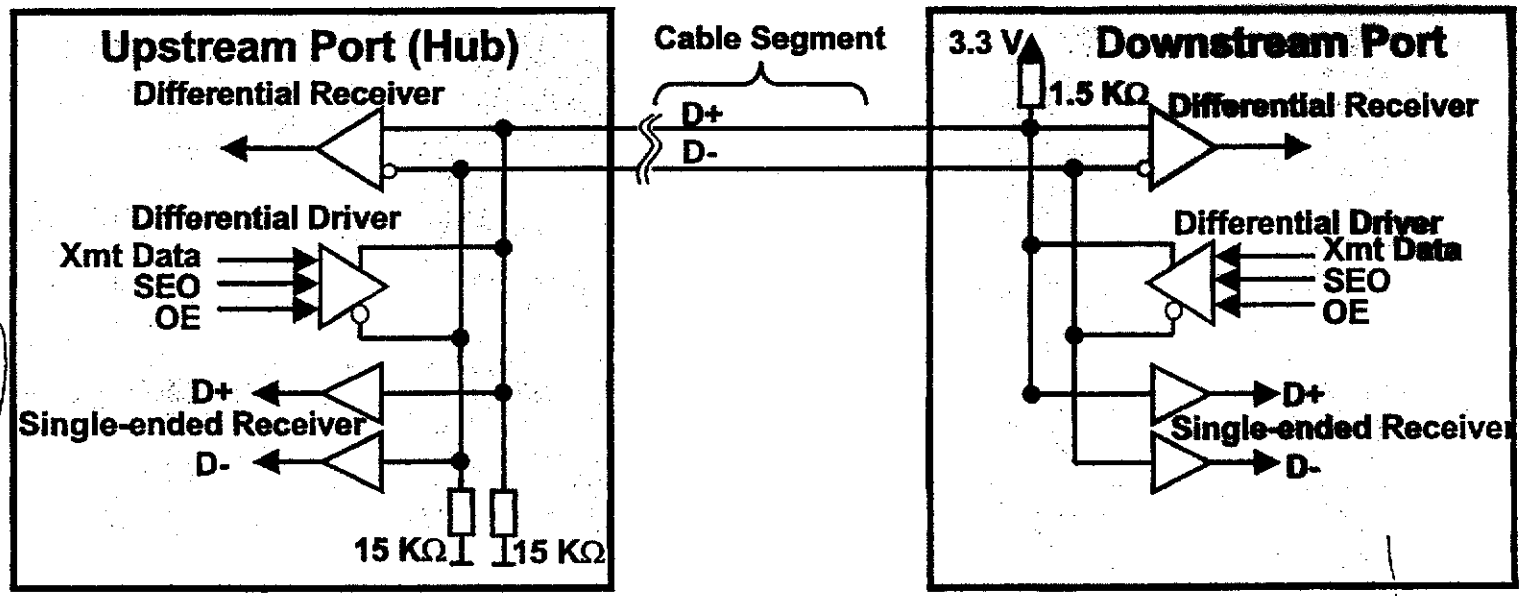
MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

Signaling Interface

Differential Signals: Differential signaling is used to ensure data integrity and eliminate noise problems

Differential Receiver: Differential receiver must feature an input sensitivity of 200mV when both signals are in the range of 0.8V to 2.5V

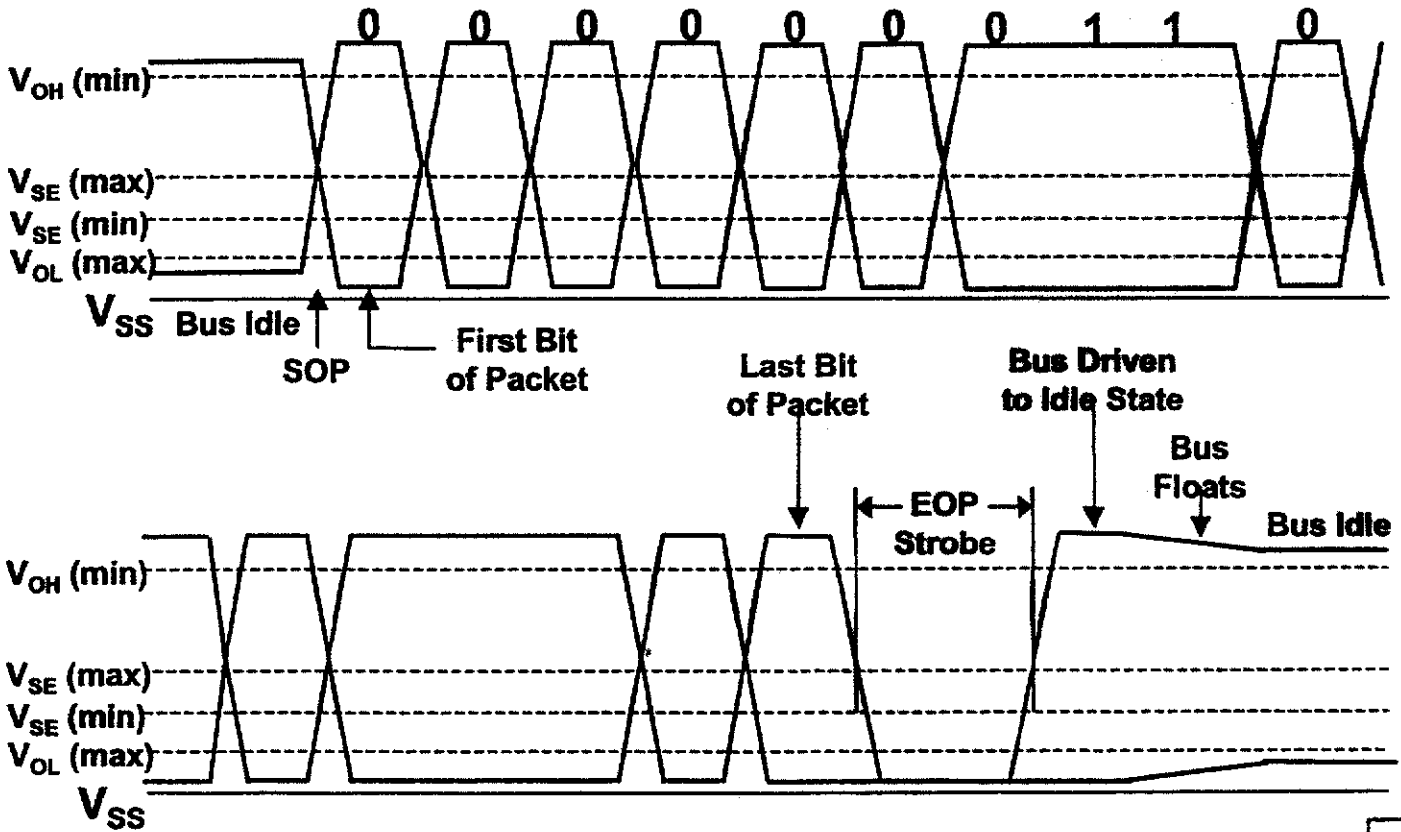
Single-Ended Receiver: Single-Ended receiver are used to detect various state conditions



MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

USB Signal Levels

Differential signaling is measured from the point where data line signals cross over. The cross over point must be between 1.3V and 2.0V.

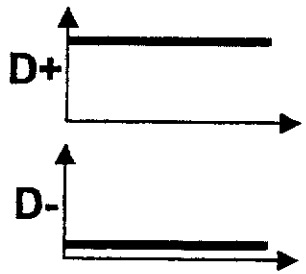


J and K Data State

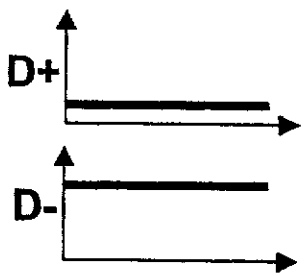
J and K data state are two logical levels used to carry differential data over the bus.
J and K data state for full speed are inverted from those of low speed.

Full Speed

J - State
differential 1
Idle State

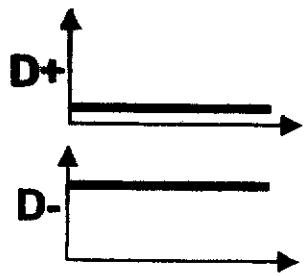


K - State
differential 0
Resume State

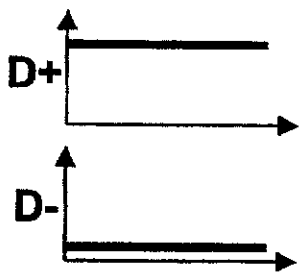


Low Speed

J - State
differential 0
Idle State



K - State
differential 1
Resume State



USB Signal States

The USB Bus uses 3.3 V DC signaling levels.

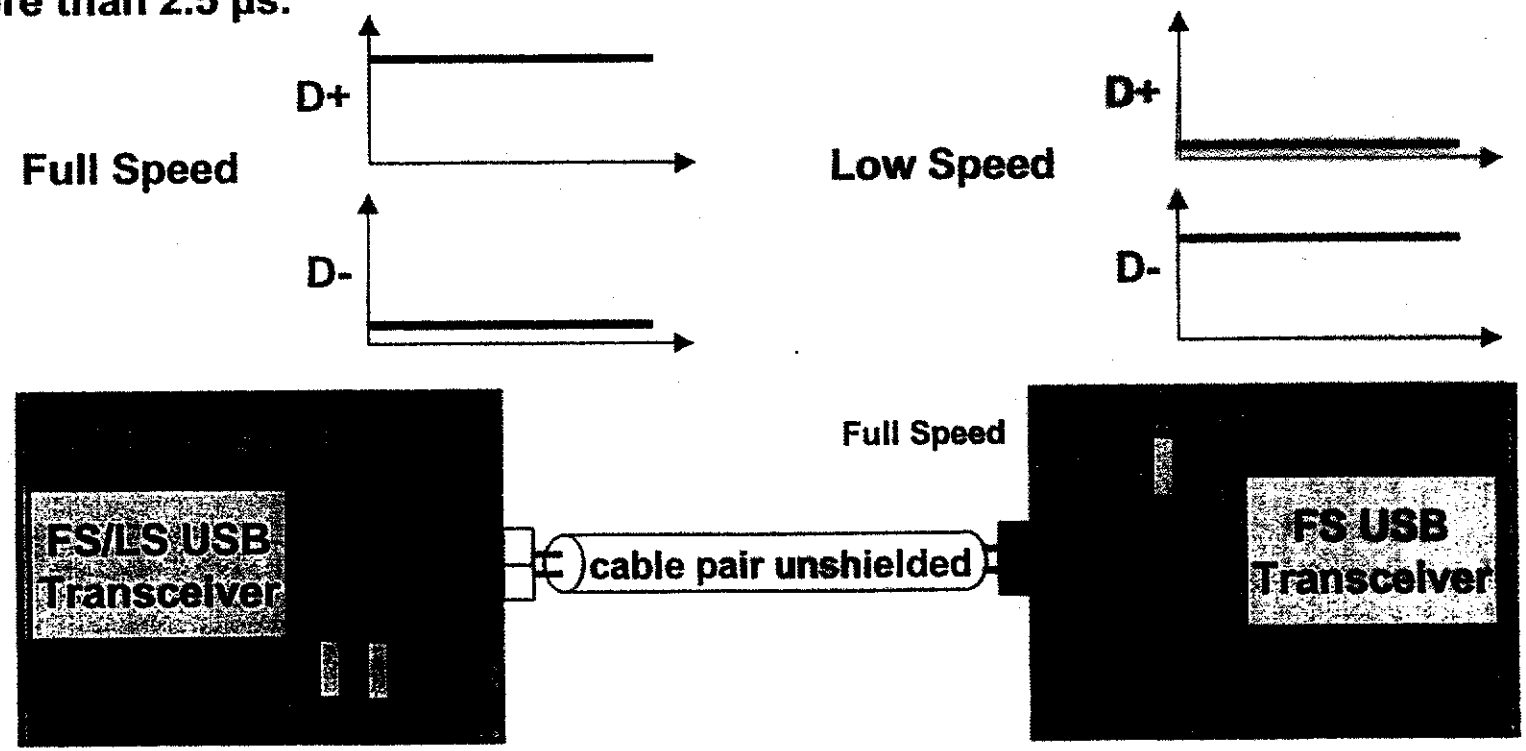
- Disconnect **D+ and D- lines less than $V_{SE(min)}$ for equal or greater than 2.5 ms**
- Connect **D+ and D- lines greater than $V_{SE(min)}$ for equal or greater than 2.5 ms**
- Reset **D+ and D- lines less than $V_{SE(min)}$ for equal or greater than 2.5 ms**
  (must be recognized within 5.5 ms)
- Differential "1" **|D+| - |D-| greater than 200 mV and D+ or D- greater $V_{SE(min)}$**
- Differential "0" **|D-| - |D+| greater than 200 mV and D+ or D- greater $V_{SE(min)}$**
- Data J-state **Differential "1" for Full Speed (pull up at line D+)**
  Differential "0" for Low Speed (pull up at line D-)
- Data K-state **Differential "0" for Full Speed (pull up at line D+)**
  Differential "1" for Low Speed (pull up at line D-)

USB Signal States

⇨Idle	**Differential "1" for Full Speed (pull up at line D+) and D+ line greater than $V_{SE(max)}$ and D- line less than $V_{SE(min)}$ Differential "0" for Low Speed (pull up at line D-) and D- line greater than $V_{SE(max)}$ and D+ line less than $V_{SE(min)}$**
⇨Resume	**Differential "0" for Full Speed (pull up at line D+) and D+ line greater than $V_{SE(max)}$ and D- line less than $V_{SE(min)}$ Differential "1" for Low Speed (pull up at line D-) and D- line greater than $V_{SE(max)}$ and D+ line less than $V_{SE(min)}$**
⇨Start of packet	**Data lines switch from Idle to K-state**
⇨End of Packet	**D+ and D- lines less than $V_{SE(min)}$ for equal or greater than 1 bit time followed by a J-state**
⇨Suspend mode	**Idle state for more than 3 ms, devices go into suspend mode**

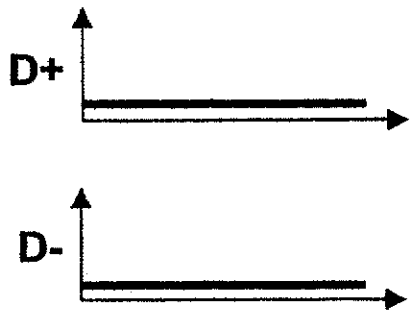
Detecting Device Attachment, Idle State

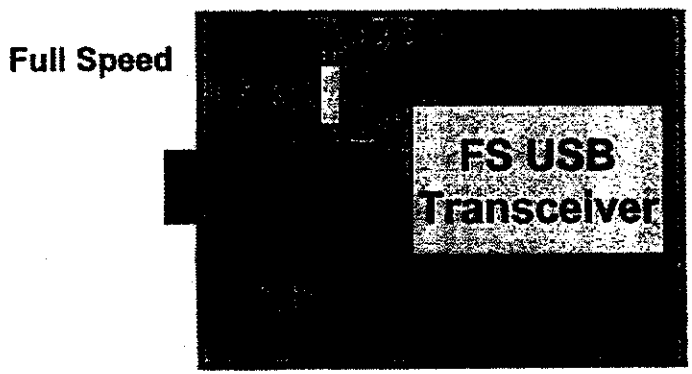
When a device is attached to the host or a hub and data lines are not being driven the resistors create a condition such that the data line with the pull-up is above 2.8 V and the other line is near ground. A connect condition is indicated if idle persists for more than 2.5 µs.



Detecting Device Detachment, Single Ended Zero, SE0

When no function is attached to a downstream port of the host or hub, the pull-down resistor will cause D+ and D- be pulled below the single ended low threshold, Vse(min). A disconnect condition is indicated if a SE0 persists for more than 2.5.µs.

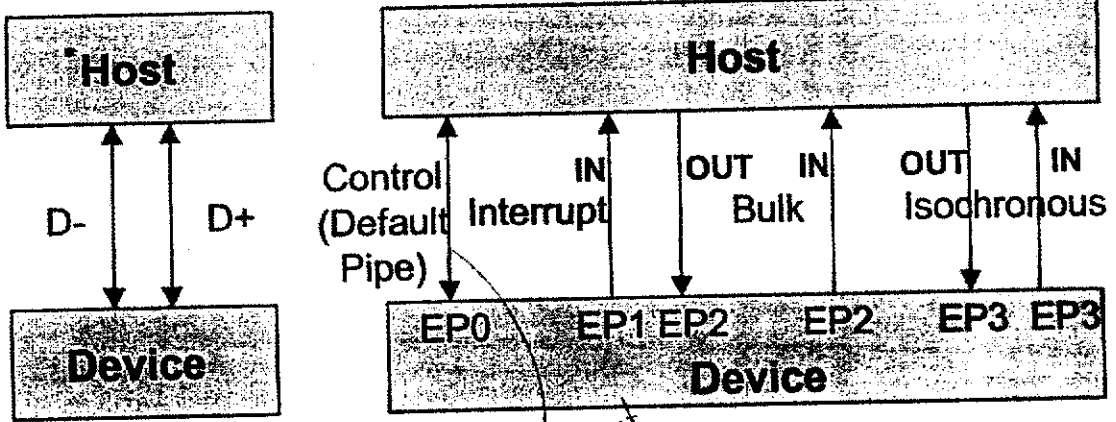




MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

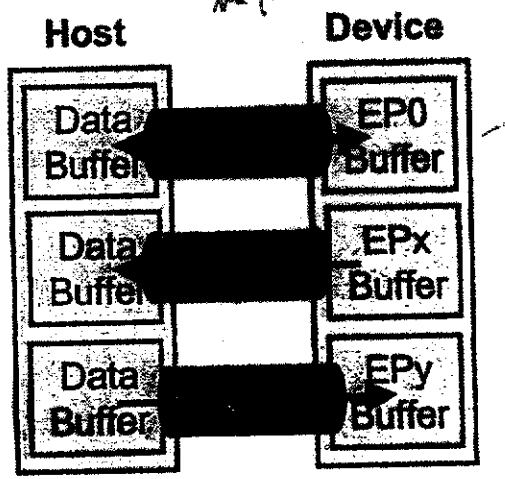
Host, Devices, Endpoints and Pipes

- an USB device is an accumulation of endpoints. There are 16 endpoints possible.
- EP0 is always bidirectional, EP1 to EP15 can be IN or OUT.
- from this follows that maximally 31 pipes are possible.
- data are transfered across the USB through pipes, that are buffers on the host and FIFOs at the endpoints on the USB device.



physical connection

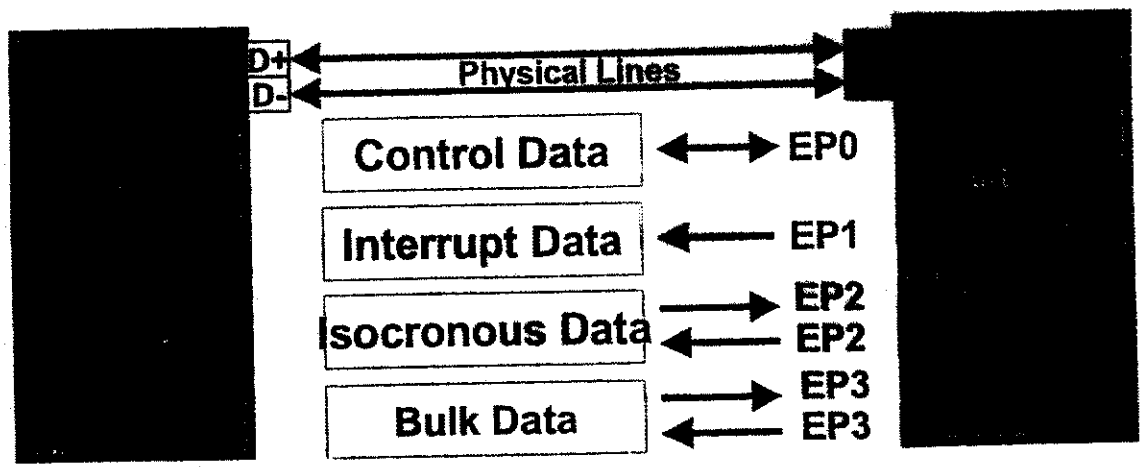
logical connection



MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

USB Transfer Types

- There are four types of data transfer possible:
  - **Control Data** — for signalling and configuration
  - **Interrupt Data** — for less than 64 byte with FS / 8 byte with LS
  - **Isochronous Data** — for audio and compressed video transmission
  - **Bulk Data** — for mass data transmission to printer or scanner
- Each of the different transfer types via **the physical lines (D+ and D-)** is assigned to a logical data transmission **channel, the Pipe**.



Control Data Transfer

- for configuration of USB devices (Hubs and Functions)
- for standard requests, device class request, vendor specific requests
  GET_CONFIGURATION, GET_DESCRIPTOR, SET_ADDRESS, ...
- always transmitted via Control Endpoint 0, EP0
- only EP0, the default pipe, offers bidirectional data transfer
- 10 % of bandwidth for each 1 ms frame are reserved for control
- FIFO length: with full speed 8 / 16 / 32 / 64 bytes
  with low speed 8 bytes
- available with full and low speed USB bus
- error detection and recovery

Interrupt Data Transfer

- for data transmission to interrupt driven devices e.g. keyboard, mice
- the host has to poll devices periodically, USB supports no hardware interrupts
- poll interval is programmable in steps of 1 ms
- the pipes for interrupt transfer are unidirectional **(direction to host)**
- max. 90 % of bandwidth for each 1 ms frame **are reserved for** interrupt and isochronous transfers
- FIFO length maximally 64 Byte
- available with full and low speed USB bus
- error detection and recovery

Isochronous Data Transfer

- ⇨for real-time applications that require a constant data transfer rate
  e.g. speaker, microphone
- ⇨ the need to provide data on a timely basis is more importend than verifying accurate delivery
- ⇨ no error handling and correction (erroneous telegrams are not retransmitted)
- ⇨ FIFO length for isochronous data max. 1023 byte
- ⇨ max. 90 % of bandwidth for each 1 ms frame are reserved for interrupt and isochronous transfers
- ⇨ only with full speed USB bus available

Bulk Data Transfer

- used for transmission of large - not time critical - data blocks
  e.g. printer, scanner etc.
- access to the USB bus only if there is available bandwidth
- max. FIFO length is 64 byte
- error handling and correction for corrupted data
- IN and OUT pipes may be configured
- only for full speed USB bus available

MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

Transfer, Transactions, Packets

- a transfers is performed using one or more transactions
  a transaction consists of a series of packets,
  token-, data- and handshake-packets

Start of Frame **Host**

Sync			CRC5
00000001	0xA5	0x111	0x04

Token **Host**

Sync		ADDR	ENDP	CRC5
00000001	0x81	0x34	0x6	0x04

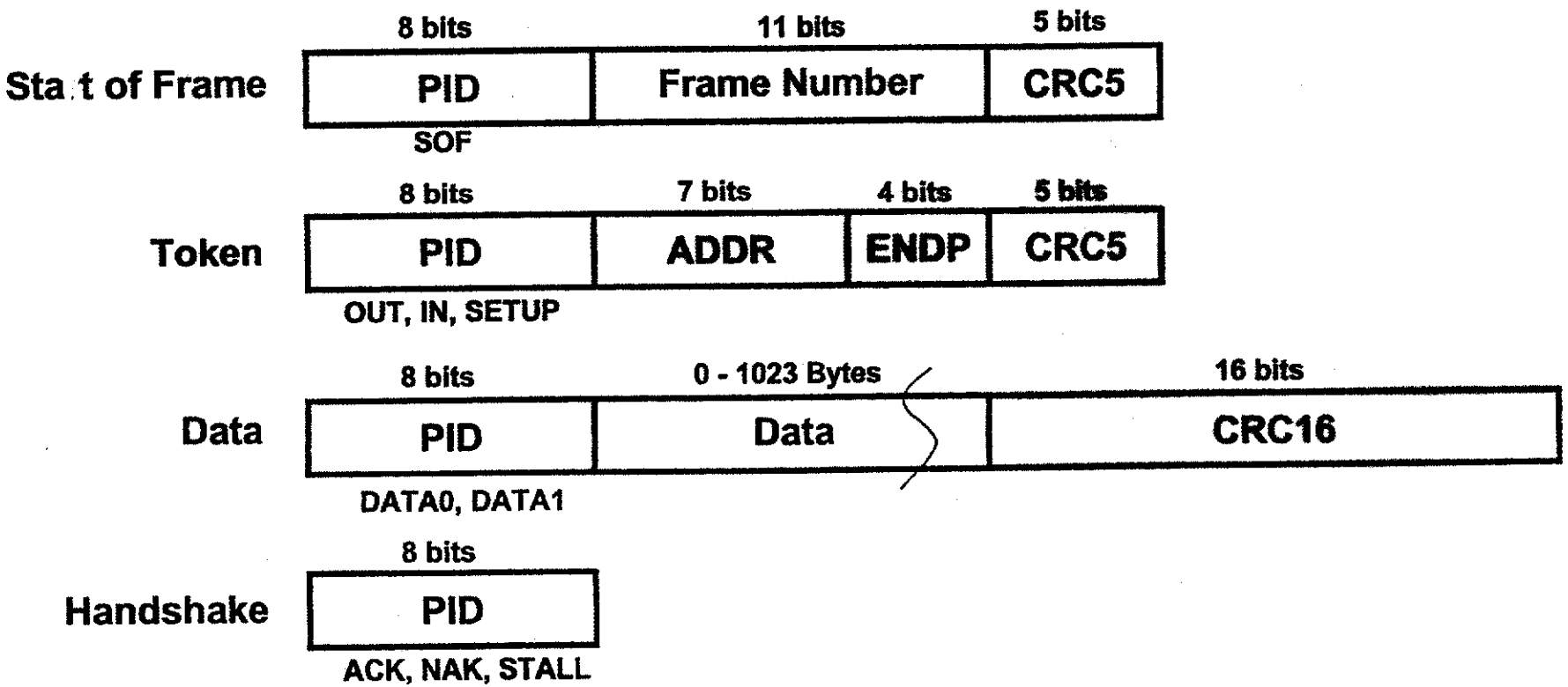
Data **Host**

Sync	DATA0		CRC16
00000001	0xC3	80 06 00 01 00 00 40 00	0xAD34

Handshake **Function**

Sync	ACK
00000001	0xD2

Packet Definition



Packet Definition cont.

➪ Bit Ordering

Bits are send out LSB first followed by next LSB, through to MSB last

➪ SYNC Field, KJKJKJKK (00000001)

All packets begin with a SYNC field for synchronization of incoming data with local clock

➪ PID Field

The Packet Identifier Field, 4 Bit + 4 Bit Checkfield indicates the type of packet.
PIDs are divided into four coding groups

Token:	IN, OUT, SOF, SETUP
Data:	DATA0, DATA1
Handshake:	ACK, NAK, STALL
Special:	PRE

Packet Definition cont.

- ADDR Field for IN, OUT and SETUP packets, 7 Bit

  the ADDR field, 7 Bit, specifies the function, that is either source or destination

- ENDP Field for IN, OUT and SETUP packages

  the Endpoint field, 4 Bit, specifies the function specific endpoint

- Frame Number Field, 11 Bit

  the frame number field is an 11-bit field that is incremented by the host for each frame

- Data Field

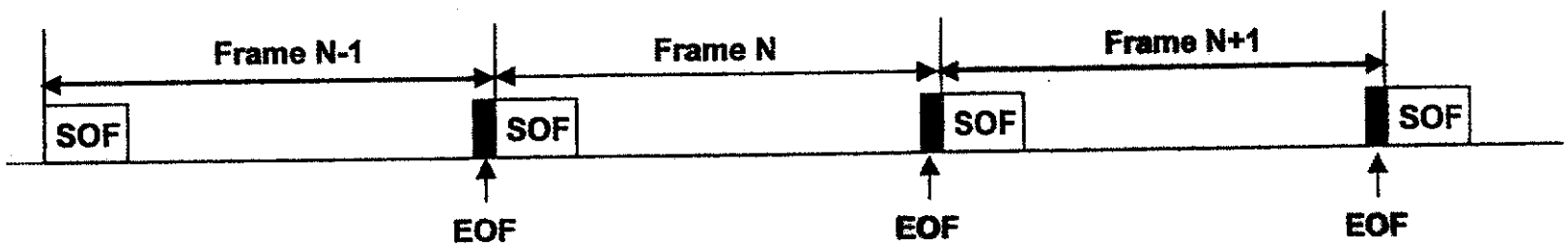
  the data field may range from 0 to 1023 bytes

- Cyclic Redundancy Check

  the CRC is used to protect all non-PID fields in token (5 Bit) and data (16 Bit) packets

Frame Generation

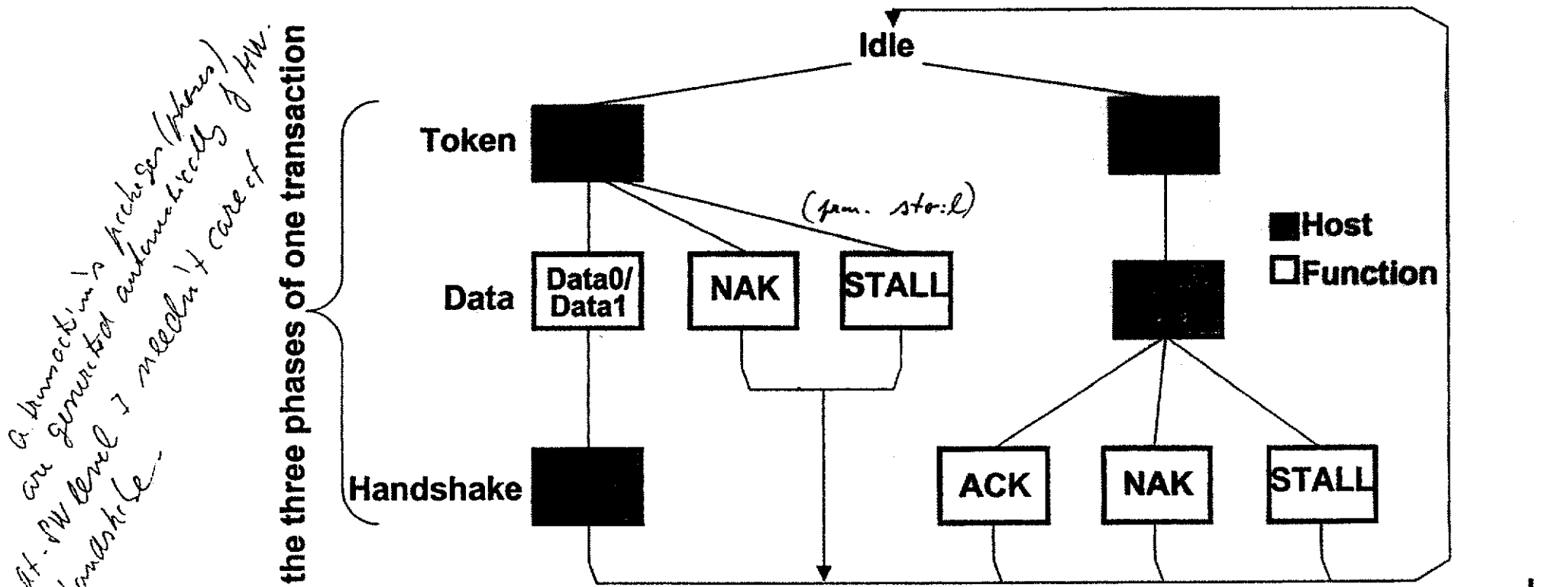
- the host controller partitions USB time into 1ms frames
- frames are generatet by issuing **S**tart **O**f **F**rame tokens at 1 ms intervalls
- after SOF the host is free to transmit other transactions for **remaining time**



Bulk Transfer

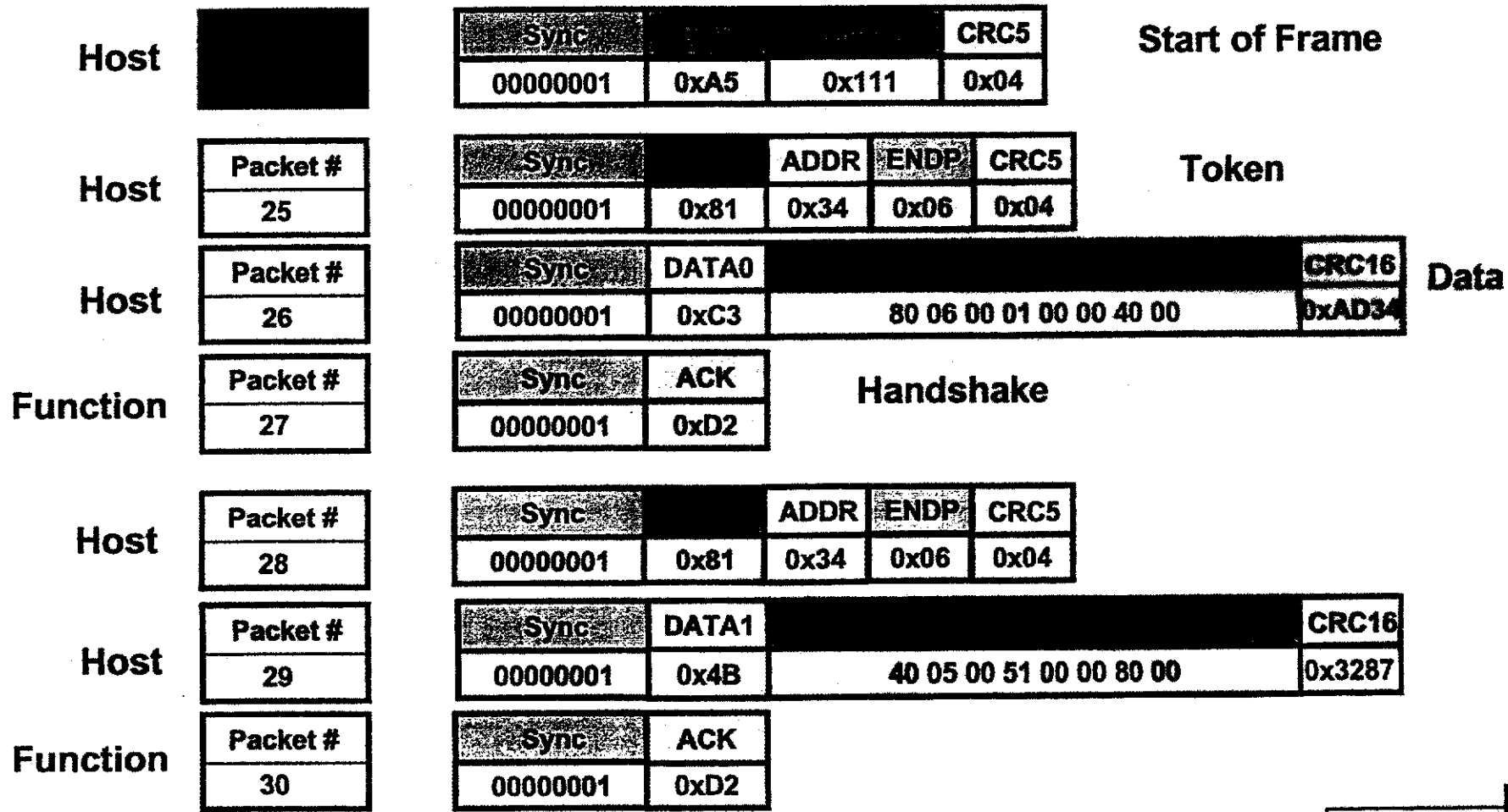
A bulk transfer typically consists of one ore more transatcions.

A transaction consists mostly of three phases: Token, Data and Handshake.



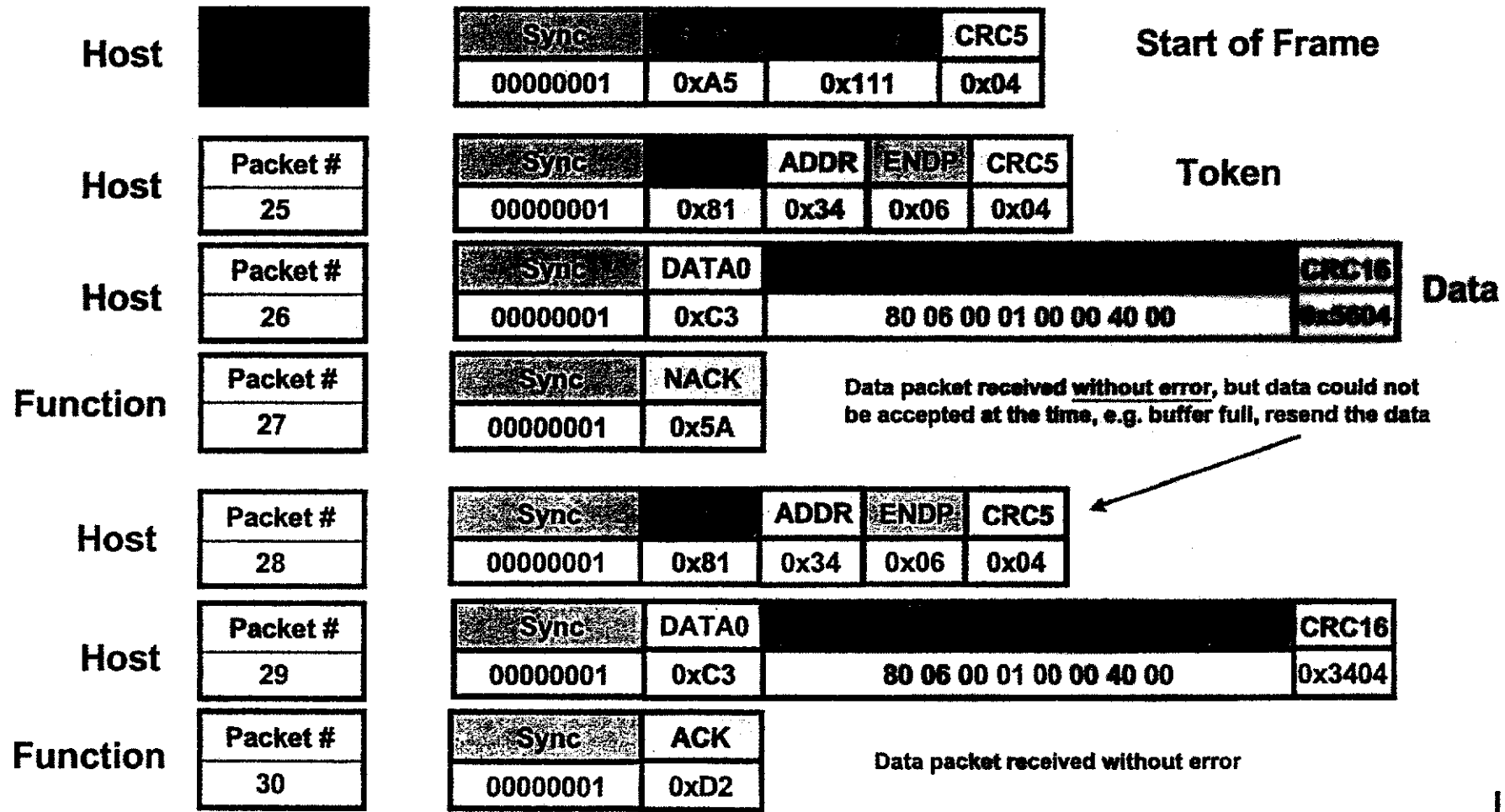
MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

Bulk Transfer, OUT



MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

Bulk Transfer, OUT



MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

Bulk Transfer, OUT

Host		Sync			CRC5
		00000001	0xA5	0x111	0x04
Host	Packet #	Sync		ADDR	ENDP	CRC5
	55	00000001	0x81	0x34	0x06	0x04
Host	Packet #	Sync	DATA0		CRC16
	56	00000001	0xC3	80 06 00 01 00 00 40 00	0x6674

Function received data packet with CRC error, no ackknowledge, after time out host sends again and an error counter is counted

Host	Packet #	Sync		ADDR	ENDP	CRC5
	57	00000001	0x81	0x34	0x06	0x04
Host	Packet #	Sync	DATA0		CRC16
	58	00000001	0xC3	80 06 00 01 00 00 40 00	0x0604
Function	Packet #	Sync	ACK		
	59	00000001	0xD2		

Data packet received without error

USB Error Detection

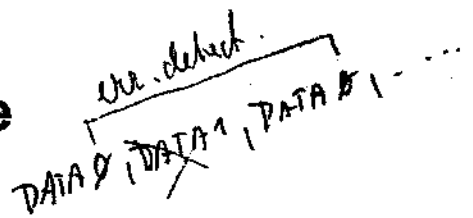
In Order to achieve a very high end to end data transfer integrity, a hardware implemented error management unit is processing error **detection** and **response.** The informed transmitter retransmits the message again (except of isochronous data).

USB error detection mechanisms:

- ⇨ **Packet error checks**

  PID check bits violation, CRC, Bit Stuff **errors**

- ⇨ **False EOP**
- ⇨ **Bus time out - no response**
- ⇨ **Data toggle error checks**
- ⇨ **Babble - transactions occurring beyond end of frame**
- ⇨ **LOA - Loss of Activity on the bus**



USB Packet Errors Types

Field	Error Type	Receiver Response
PID	PID Check, Bit Stuff	Ignore packet
Address	Bit Stuff, Address CRC	Ignore token
Frame No.	Bit Stuff, Fame No. CRC	Ignore Frame No. Field
Data	Bit Stuff, Data CRC	Discard data

➪ USB PID Error Check

- ✓each packet which is broadcasted via the USB bus starts with a Packed ID (PID)
- ✓the PID consists of four bits PID and four bits check field (PID inverted)
- ✓all USB target devices must perform the PID check and ignor erroneous packets

USB CRC Error Check

Each packet contains CRC bits (5 or 16 bits). They are used to validate the data bits which follow the PID field.

Packet type	Contents	maximum size of data	number of CRC Bits
Start of Frame	frame number	11 bits	5
IN	device and endpoint address	11 bits	5
OUT	device and endpoint address	11 bits	5
SETUP	device and endpoint address	11 bits	5
DATA0	data	1023 byte	16
DATA1	data	1023 byte	16
ACK	packet ID only	--	--
NAK	packet ID only	--	--
STALL	packet ID only	--	--
PREAMBLE	packet ID only	--	--

Only at slow speed
HUB need it for route's
change. etc.

USB CRC Error Check cont.

The generator polynomial for 5-bit CRC is: $G(x) = x^5 + x^2 + 1$

for 16-bit CRC is: $G(x) = x^{16} + x^{15} + x^2 + 1$

Note: The transmitted bit stream - including the CRC-code - on the lines includes a stuff bit always after six consecutive bits on "1".

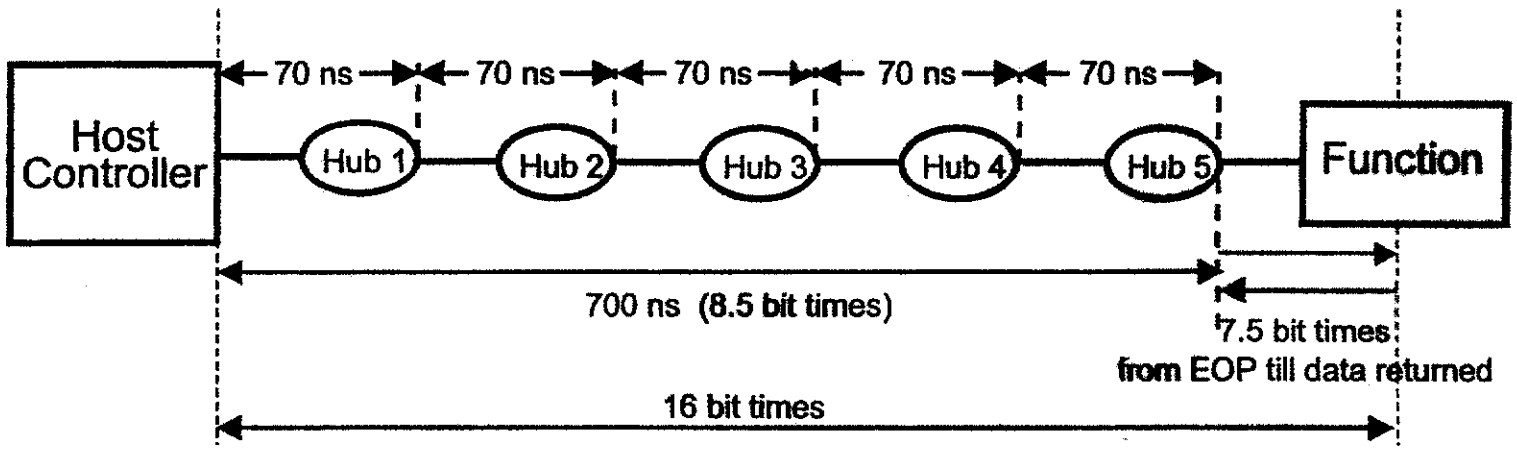
Bit Stuff Errors

The USB receiver expects a stuff bit (guaranteed transition for resynchronization) always after six consecutive bits on "1". If the stuff bit is not present, this indicates to the error detection logic a corrupted transmission:

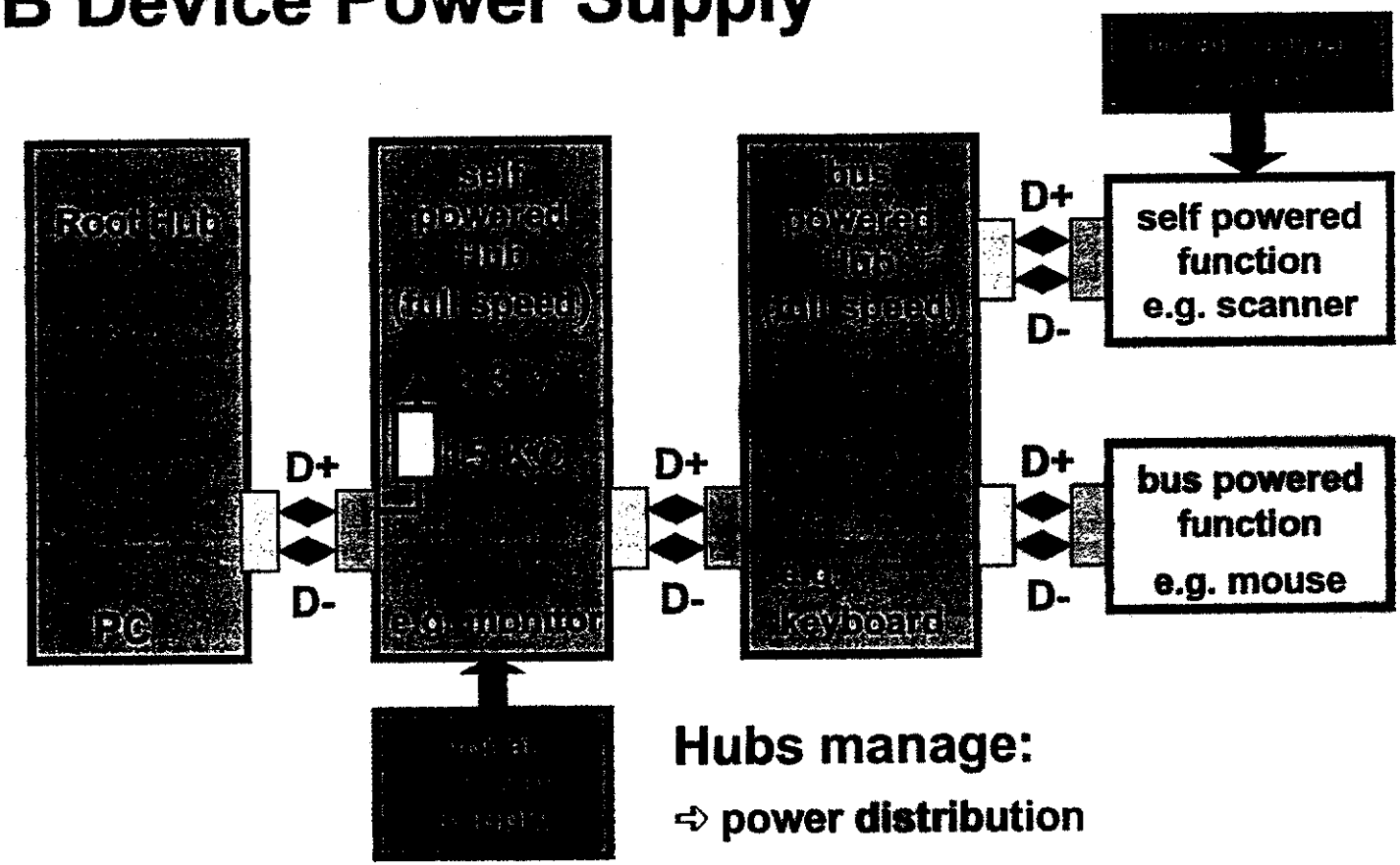
the packet has been corrupted or
the transmitter is not properly generating stuff bits or
the receiver is not decoding the NRZI coded data correctly.

Bus Time Out - Bus Turn around Timing

The USB host / USB functions are controlling time from when the transmitter completed a transmission of a packet - transition of EOP - util the receiver responds. The USB device cannot time out earlier than 16 bit times after the end of previous EOP strobe. The host must wait at least 18 bit times - before issuing the next token - to indicate a time out. This insures that all downstream devices have timed out.



USB Device Power Supply



Hubs manage:

- ⇨ power distribution
- ⇨ over current detection
- ⇨ Hubs can be self powered or bus powered

USB Self Powered Hub

- Power consumption from the upstream port max 100 mA
- Power distribution to a downstream port 500mA per port

USB Bus Powered Hub

- Power consumption from the upstream port max 500 mA
- Power distribution to a downstream port 100mA max. 4 ports

Power Modi

➪ **Direct Powered**

ports are always powered

➪ **Ganged Switching**

all ports can be switched on or off at the same time

➪ **Individual switching**

each port can be switched on or of individually

USB Self Powered Function

➪ Power consumption from the upstream port max 100 mA

USB Bus Powered Function

➪ Low Power Function

power consumption max. 100mA

➪ High Power Function

before configuration max. 100 mA

after configuration max. 500 mA

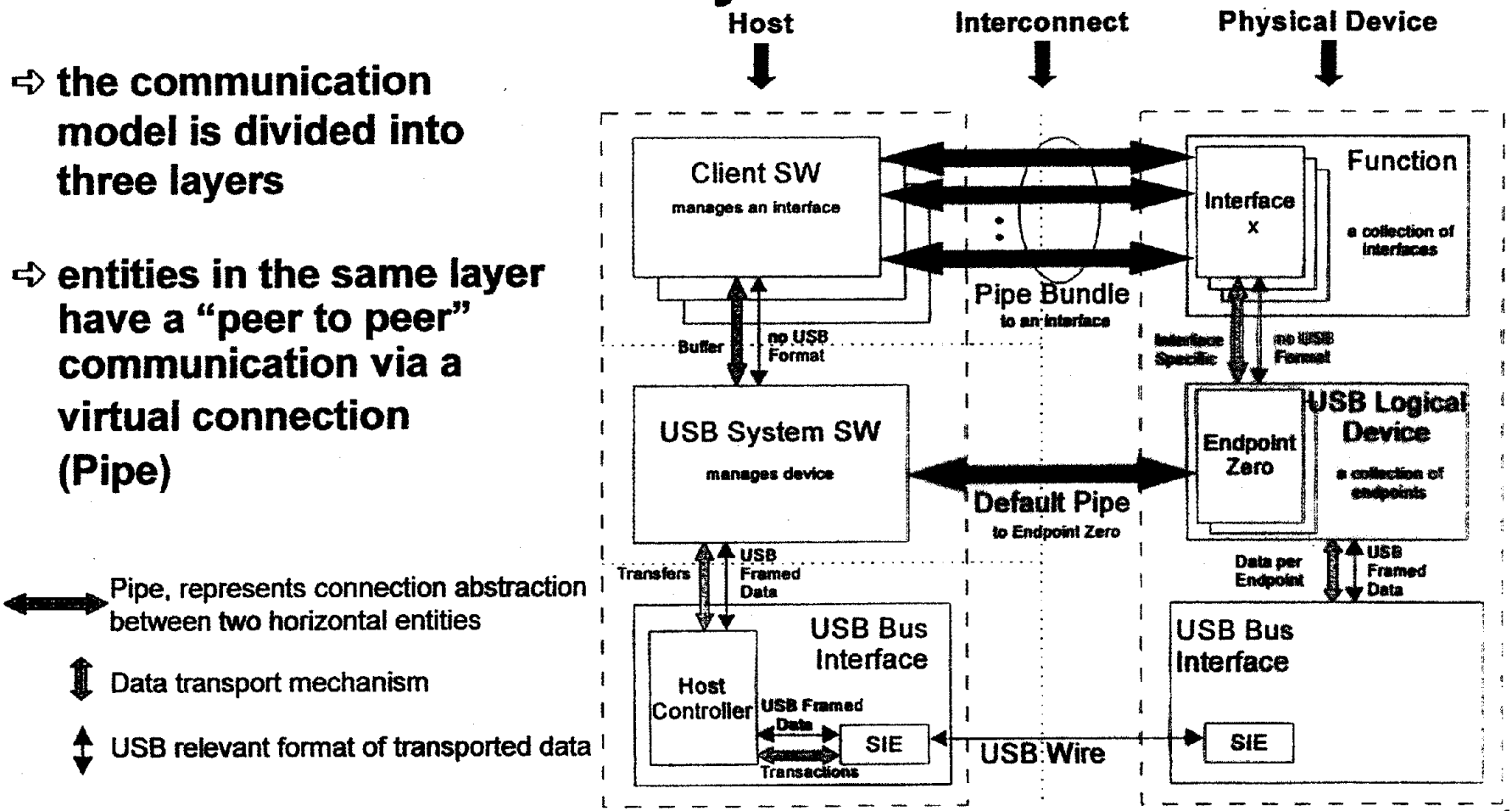
➪ Suspend State

power consumption max. 500 µA

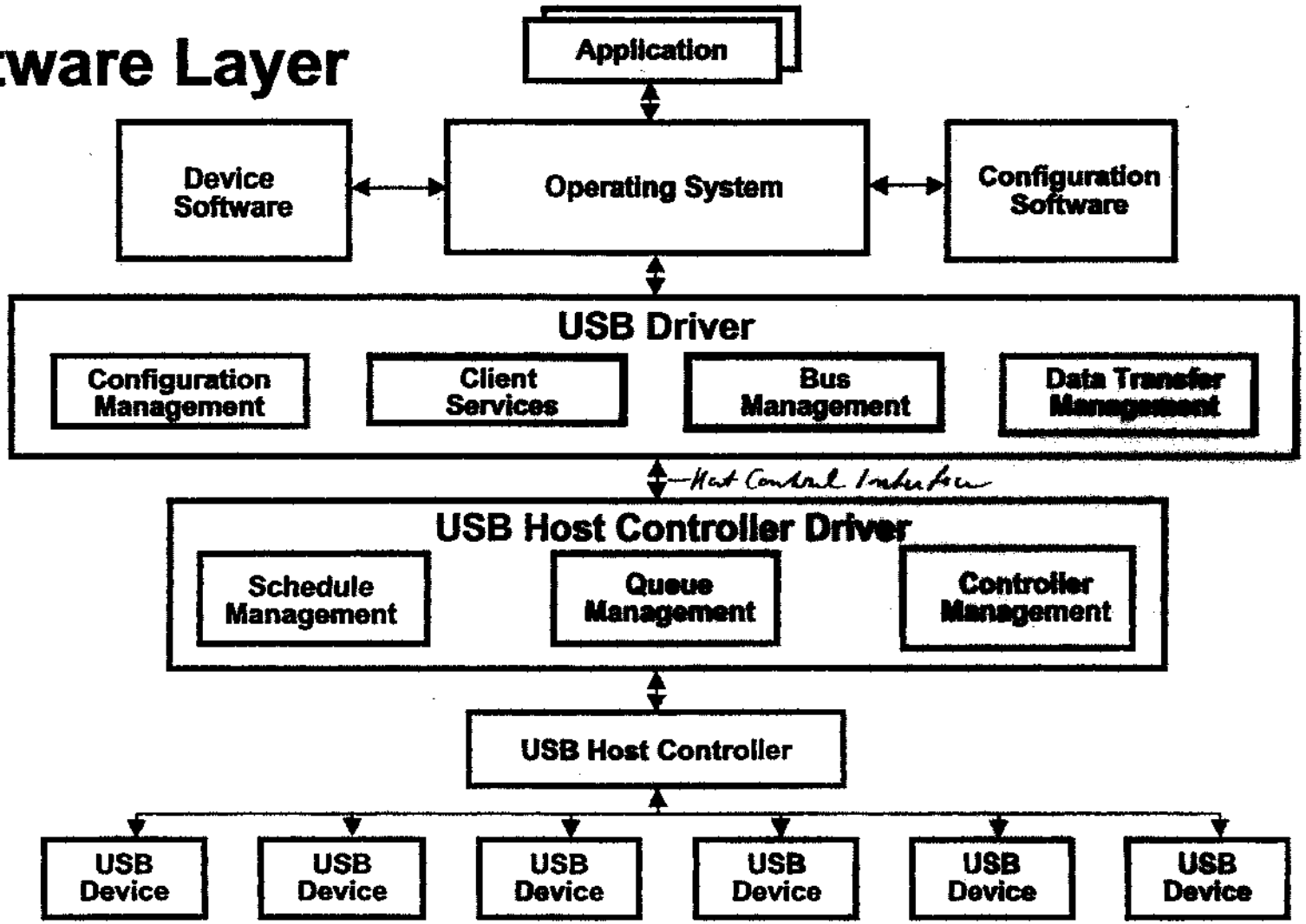
MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

The Communication Layers

- the communication model is divided into three layers
- entities in the same layer have a "peer to peer" communication via a virtual connection (Pipe)



Software Layer



Device Layer

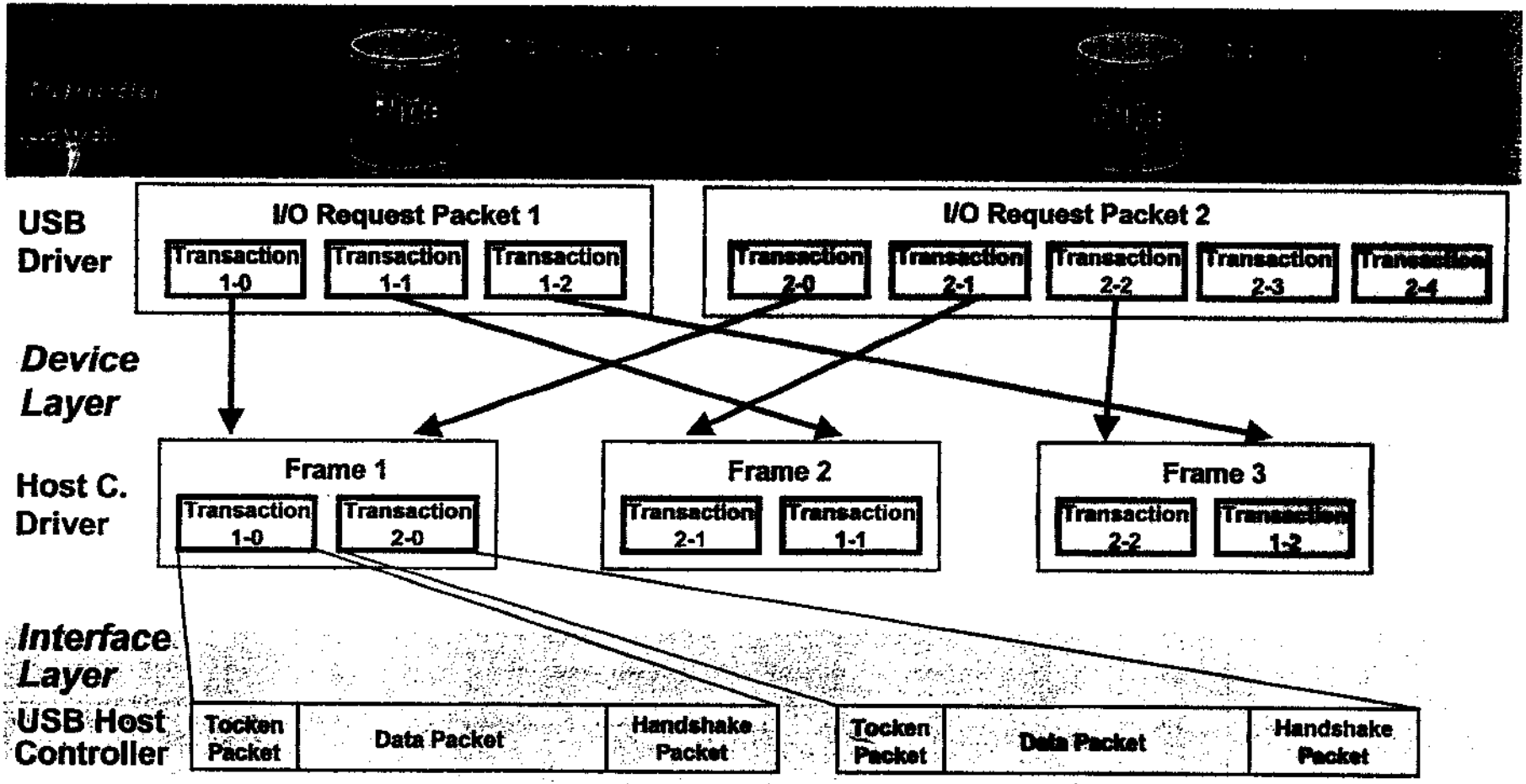
- represented by the USB driver (USBD) and Host Controller Driver (HCD)
- knows how to communicate with the USB device and organizes the into equests individual transactions
- schedules transactions to be broadcast over USB
- the software must support:
  - ✓ USB interface control, Configuration service,
  - ✓ bus and device management, Power Control, Device data access
  - ✓ event notification, Collection of status and activitiy statistics
  - ✓ error detection and handling
- the Interface between the USBD and the HCD is known as the Host Controller Driver Interface (OHCI or UHCI)

Interface Layer

- ⇨ represented by the Host Controller HW, root hub, USB interface and cable
- ⇨ broadcasts the packets over the USB

MICROCONSULT
MICROELECTRONICS CONSULTING & TRAINING GMBH

The Communication Layers



USB System Configuration

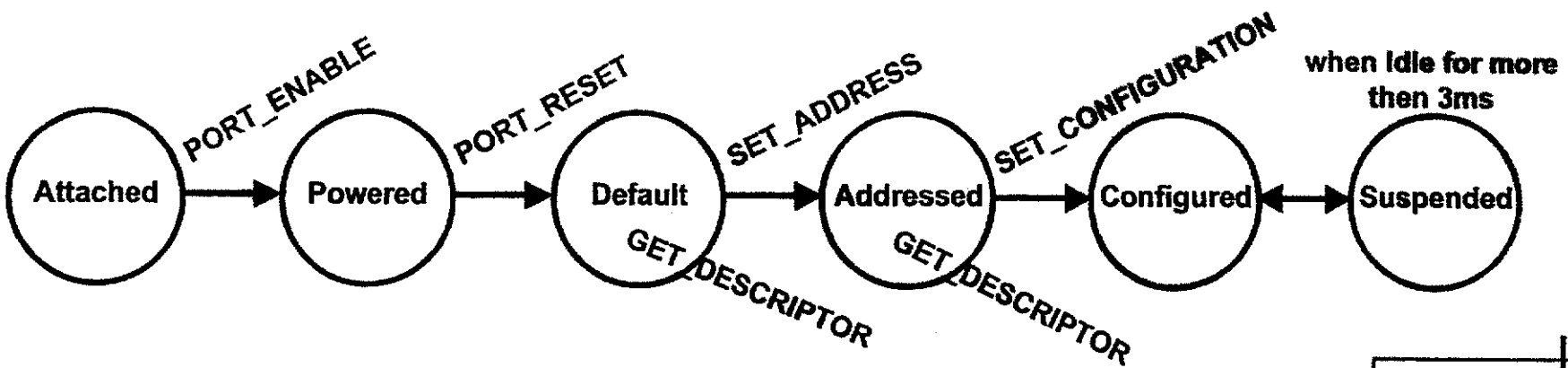
- when the System is switched on or a new device (hub or function) is attached to the system the device must be recognized and then configured

- each USB device is described by a number of descriptors that determine whitch resources (power, memory) and driver must be located to handle the device

- configuration is done by a sequence of Device Requests like enable the port, reset the device, read the device descriptors, assign an individual address, write a configuration value to the device

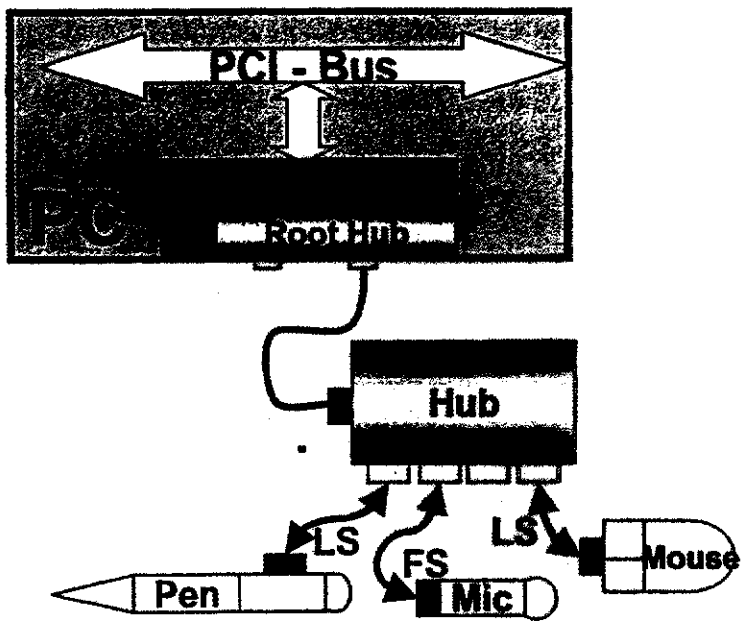
Device Enumeration

- identifying and configuring USB devices is called Device Enumeration
- Host Software is responsible for detecting and configuring attached devices
- assuming host software has initalized the root hub, its status change endpoint (an additional status endpoint for hubs) is polled to detect conneted devices
- once a USB device is attached it goes through the following stages



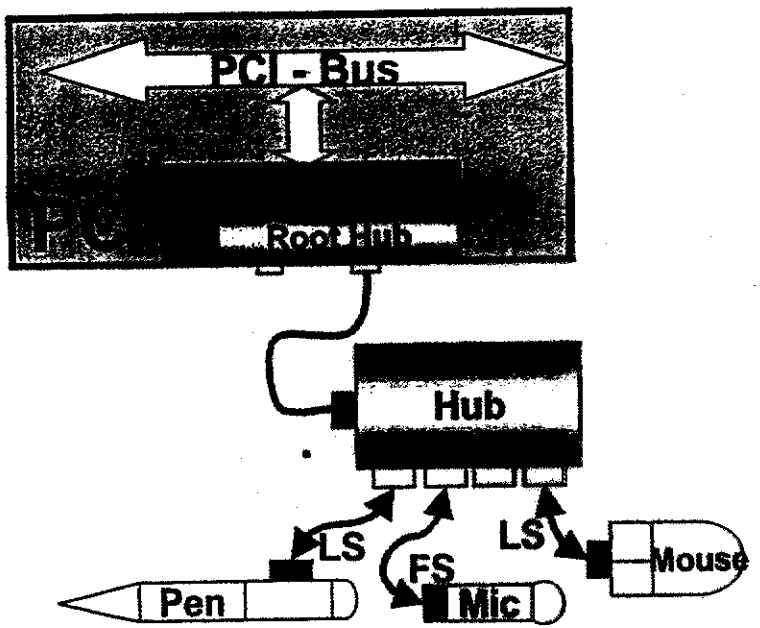
Device Enumeration Steps

- Host configures root hub
- Host requests power to be applied to ports
- Hub detects device attachment by monitoring D+ and D- signals
- Hub sets status change
- Host polls hub and identifies that a device is attached
- Host issues port enable to hub
- Host issues reset to port/device
- Port now enabled and 100 mA are available
- USB device now responds to default address zero



Device Enumeration Steps cont.

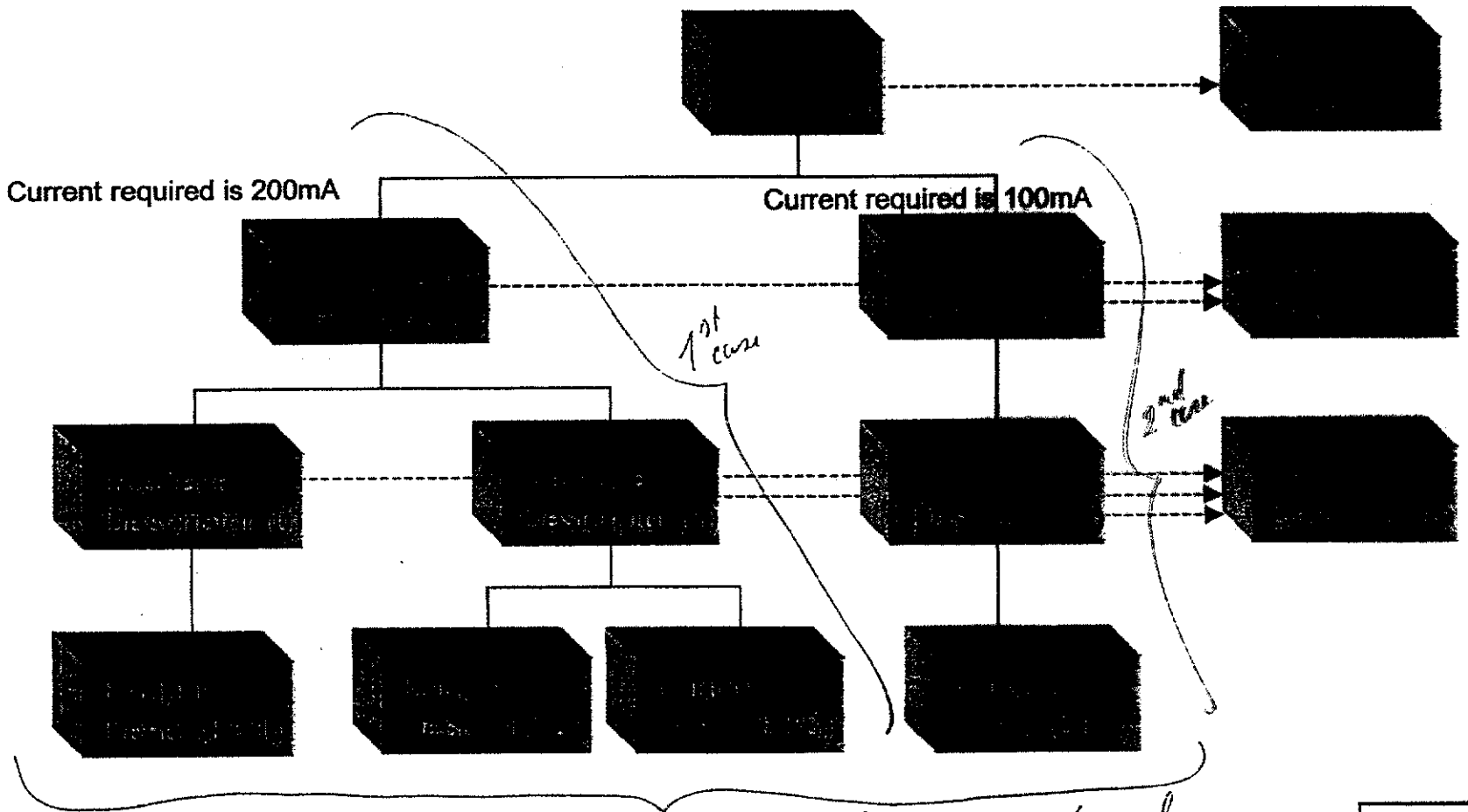
- Host reads device descriptor to determine max payload supported by default pipe
- Host assigns unique address to USB device
- Host reads device-, configuration-, interface- and endpoint descriptors
- Host verifies that USB resources needed by the device are available
- Host issues a configuration value to the device, specifying how it is to be used
- the device is now ready to be accessed



USB Device Describtion

- **one Device Descriptor**
  describes number of configurations supported by the device
- **one ore more Configuration Descriptors**
  number of interfaces and certain attributes for the configuration
- **one or more Interface Descriptors**
  number of endpoints related to the interface and certain attributes
- **one Endpoint Descriptor for each Endpoint**
  attributes associated with the endpoint
- **string Descriptors (optional)**
  human-readable information
- **class specific Descriptors**
  additional descriptors defined by a device class specification

Standard Descriptors



device with two mode of working.

1st case three functions mode
2nd case only one function mode.

USB Device Requests

- a variety of USB Request are used for configuring and controlling USB devices Standard Device Requests, Class-Specific Requests
- all requests are issued by using control transfer

Standard Device Requests are:

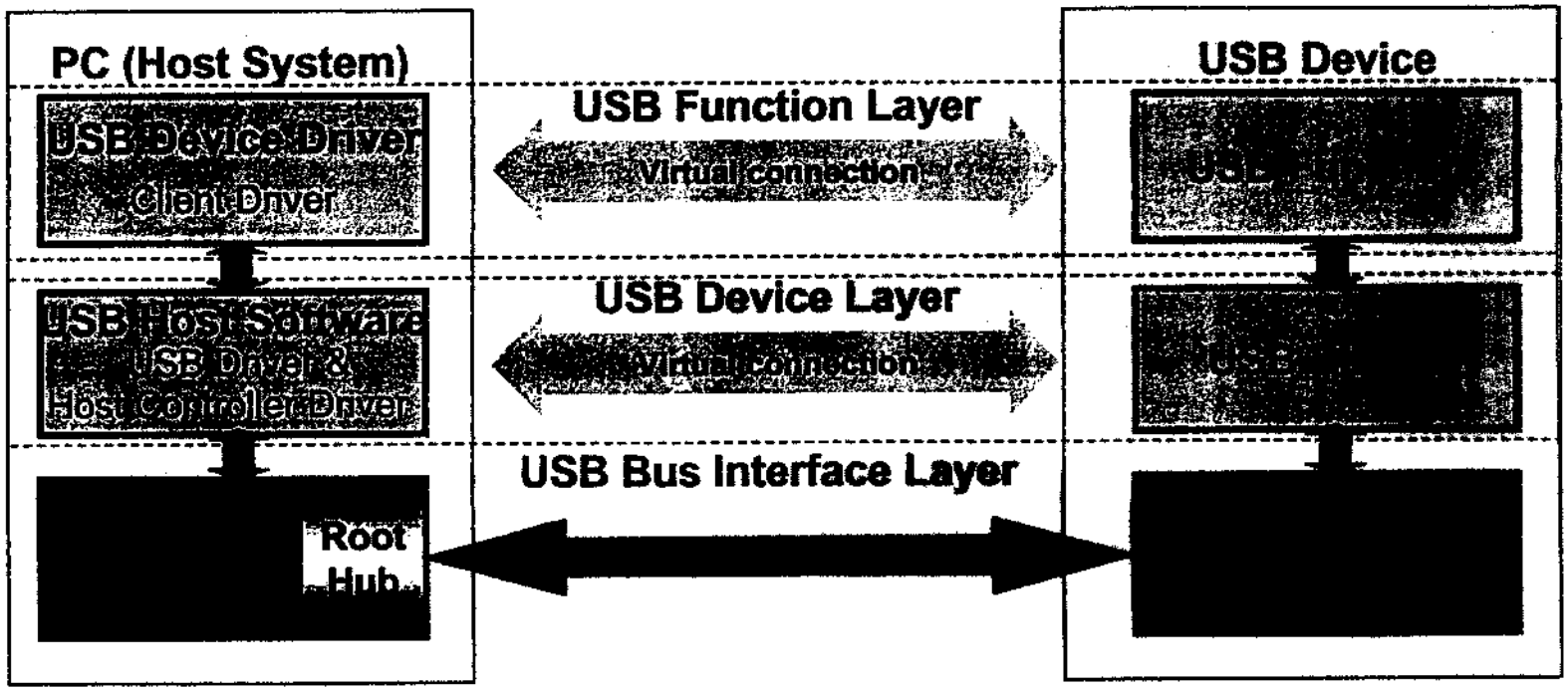
CLEAR_FEATURE
GET_CONFIGURATION
GET_DESCRIPTOR
GET_INTERFACE
GET_STATUS
SYNCH_FRAME
SET_FEATURE
SET_CONFIGURATION
SET_DESCRIPTOR
SET_INTERFACE
SET_ADDRESS

USB Device Classes

- Devices with similar attributes and services are combined to a Device Class
- Device Classes permit a device driver design that manipulates a set of devices
- USB Devices are described by a set of Standard Descriptors and additional Device Class Specific Descriptors
- Device Class Specific Descriptors provide the USB Device Driver with the information required to handle the device
- USB Devices support the standard requests defined in the USB Specification and class specific requests defined in the USB Device Class Specification
- USB Device Class Definitions describes specific attributes and characteristics that devices within a class may support

The Communication Layers

- Device Class definitions relates to the functional layer
- Device Class Specific Descriptors provide the USB Device Driver with the information required to handle the device



Device Class Working Group

- the Device Class Working Group of the USB Implementers Forum has already defined Device Classes and the Device Class Specific Descriptors for:
  - **Mass Storage Device Class**
  - **Human Interface Device Class, HID**
  - **Printer Device Class**
  - **Power Device Class**
  - **Monitor Device Class**
  - **Audio Device Class**
  - **Communication Device Class**