Universal Serial Bus - Basics

Dipl.Ing. Stefan Schulze

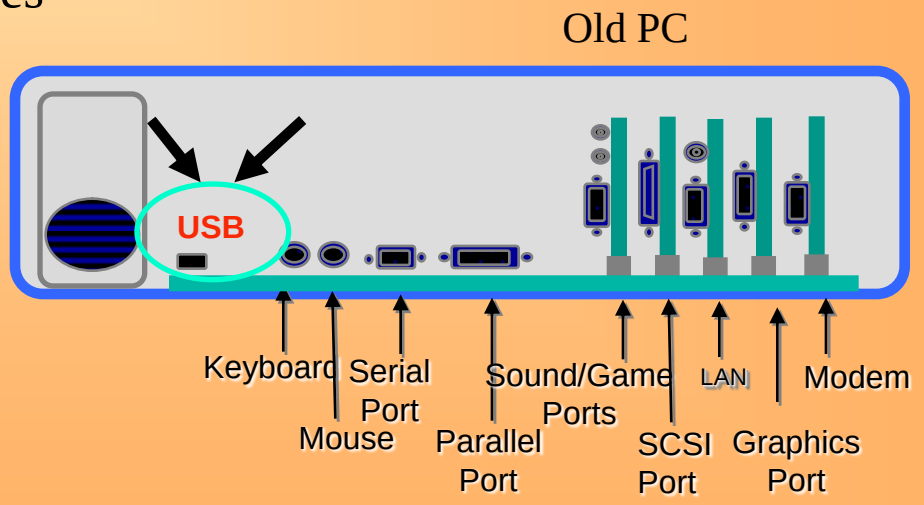
emsys Embedded Systems GmbH, Ilmenau - Germany

e-mail: stefan.schulze@emsys.de

Success Story of USB (2.0)

Why USB succeeded

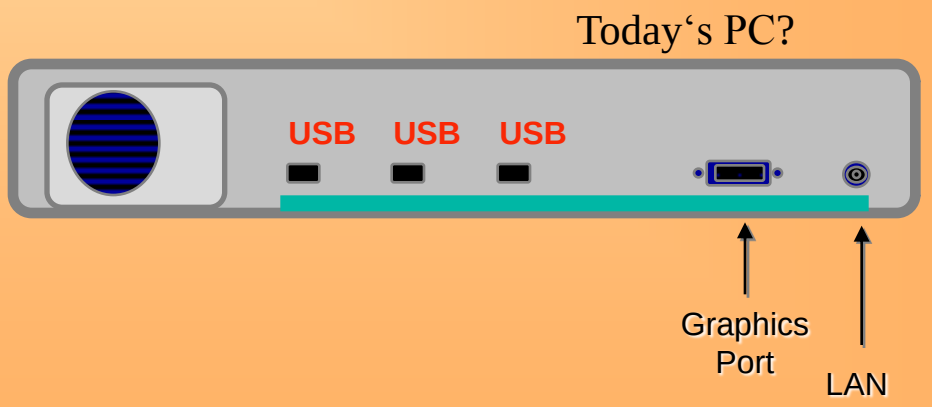
- Disadvantages of established bus systems
  - Many different connector and cable types
  - Poor possibilities to expand (limited number of slots)
  - Need of one socket for every device
  - Inefficient use of hardware resources (interrupts, I/O-areas, DMA-channels)
  - Hard to configure (jumper / DIP-switches)
  - No power supply for attached devices
  - No common API
  - No „Hot-Plug-In“



USB Advantages

⇨ Advantages of USB

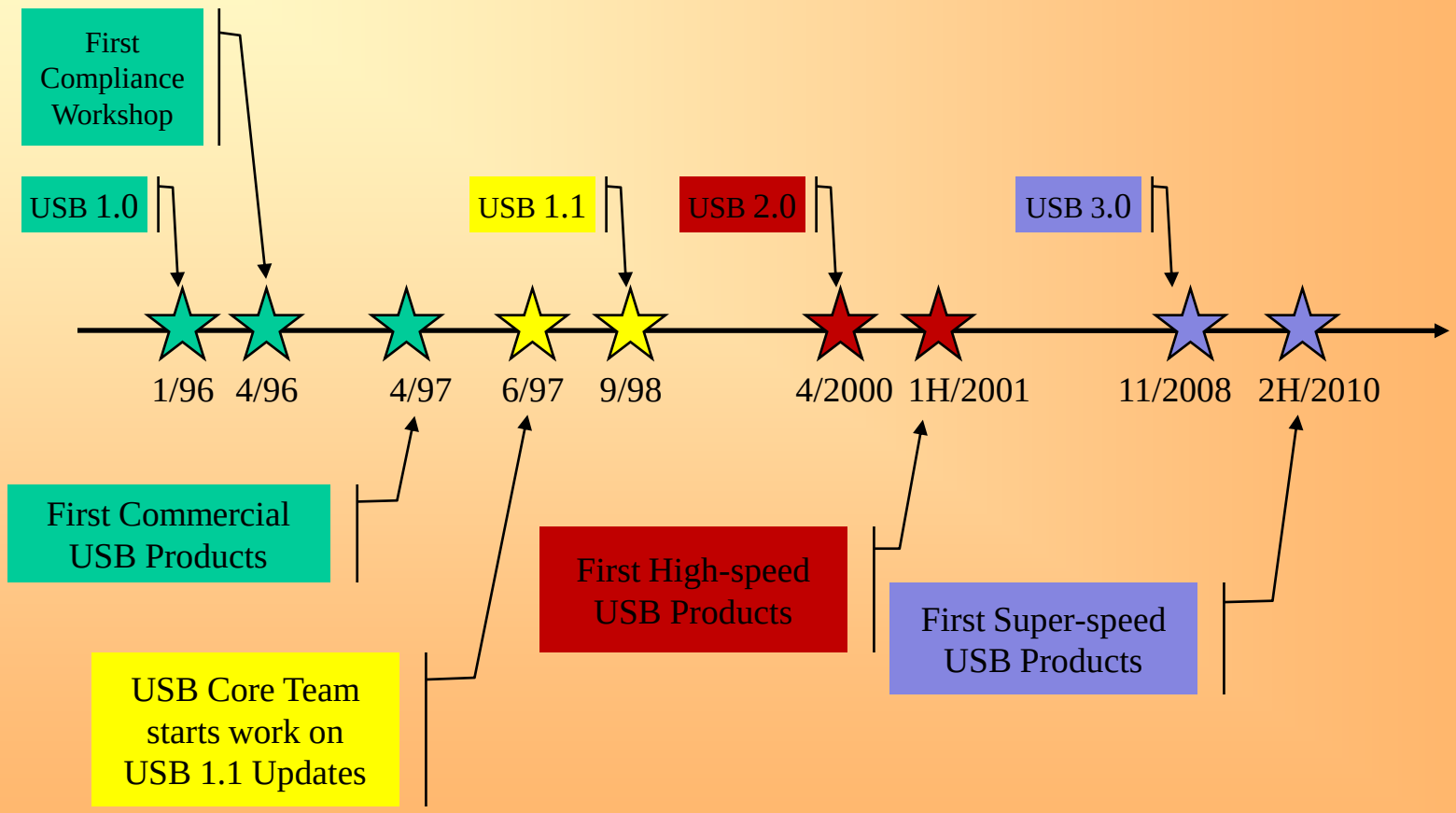
- Unified cable and connector system for all USB devices
- Usage of „Hubs“ allows extension up to 127 devices
- 5GBit/s, 480Mbit/s, **12Mbit/s, 1.5 Mbit/s** data-rate with integrated error-correction-protocol
- Power-supply over USB (5V, 500 mA) including power management
- Up to 5 m cable-segments
- Common API (Win32-Driver-Model = WDM)
- Real „plug & play“
- Real „hot plugging“



USB History

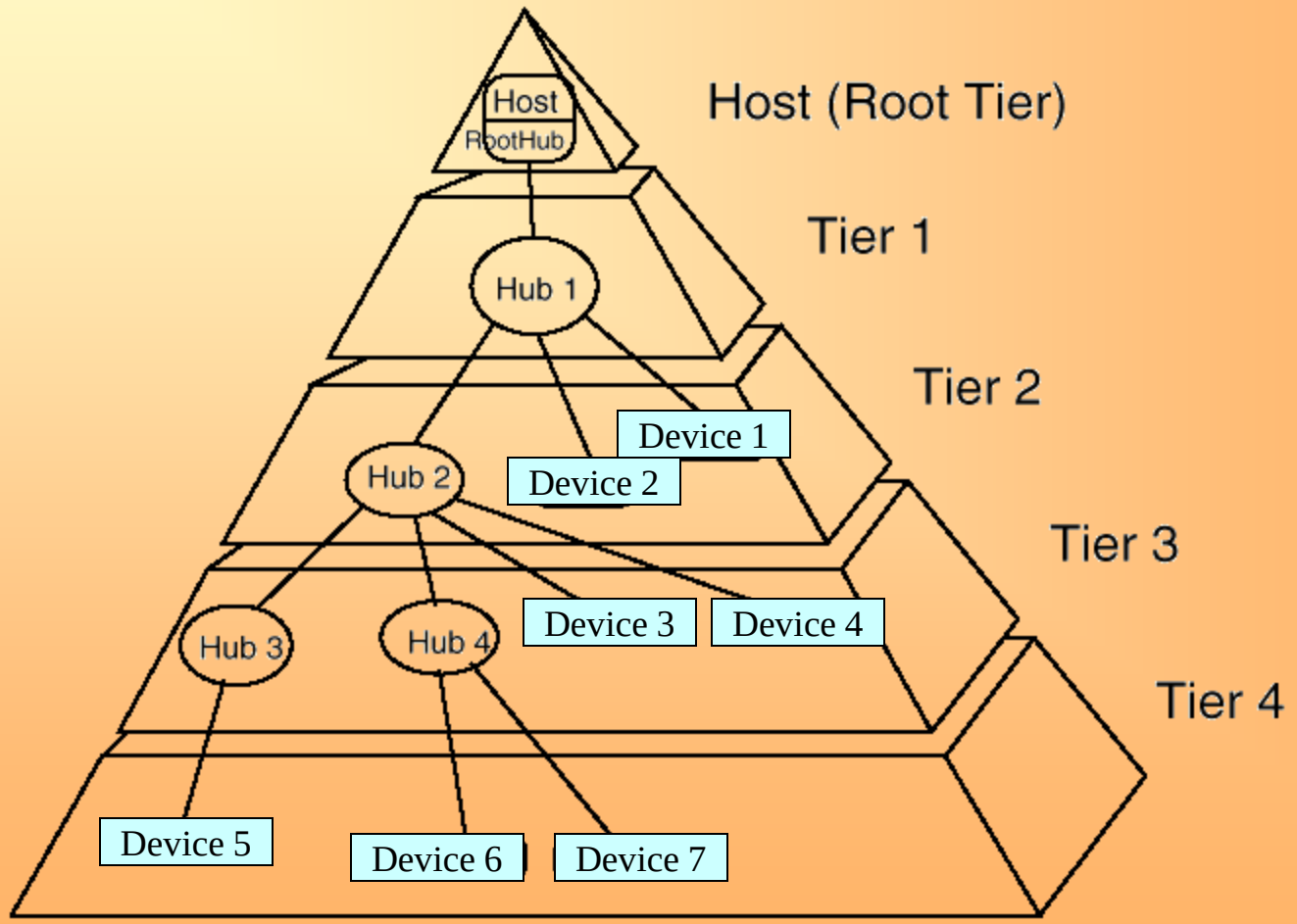
- Group for development of USB Specification founded in Q1/1995:
  - Compaq
  - Digital Equipment Corporation
  - IBM PC Company
  - **INTEL**
  - Microsoft
  - NEC
  - Northern Telecom
- First USB Specification 1.0 published in January 1996
- First chipsets, peripheral silicon available in first half of 1996
- First PCs and peripherals end of 1996
- Breakthrough in market started with availability of Windows 98 launch

USB History


First Compliance Workshop
USB 1.0
USB 1.1
USB 2.0
USB 3.0
1/96
4/96
4/97
6/97
9/98
4/2000
1H/2001
11/2008
2H/2010
First Commercial USB Products
First High-speed USB Products
First Super-speed USB Products
USB Core Team starts work on USB 1.1 Updates

USB Architecture

USB Topology



Host

- Controls USB traffic completely
- Transmits data to USB functions & requests data from USB functions
- Controls scheduling of transfers and bandwith for all devices
- Controls the enumeration process for hubs and functions
- Produces framework (Start-of-Frames if FS or HS, EOP if LS)
- Controls power management
- Integrated in all common used PCI-PC chipsets
- Merged with a root-hub (2 or more downstream ports)
- 3 versions of implementation
  - OHCI (Open Host Controller - Compaq, Microsoft)
  - UHCI (Universal Host Controller - INTEL)
  - EHCI (Enhanced Host Controller (high-speed only - INTEL)

Hub

- Provides expand mechanism of the tiered-star topology
- „Broadcaster“ in downstream direction
- „Router“ in upstream direction
- Always high- or full-speed device
- „Plug-and-Play“- management
- Self- and/or bus-powered
- Provides power management for downstream ports
- Can be merged with another USB device
  (e.g. keyboard with integrated hub)

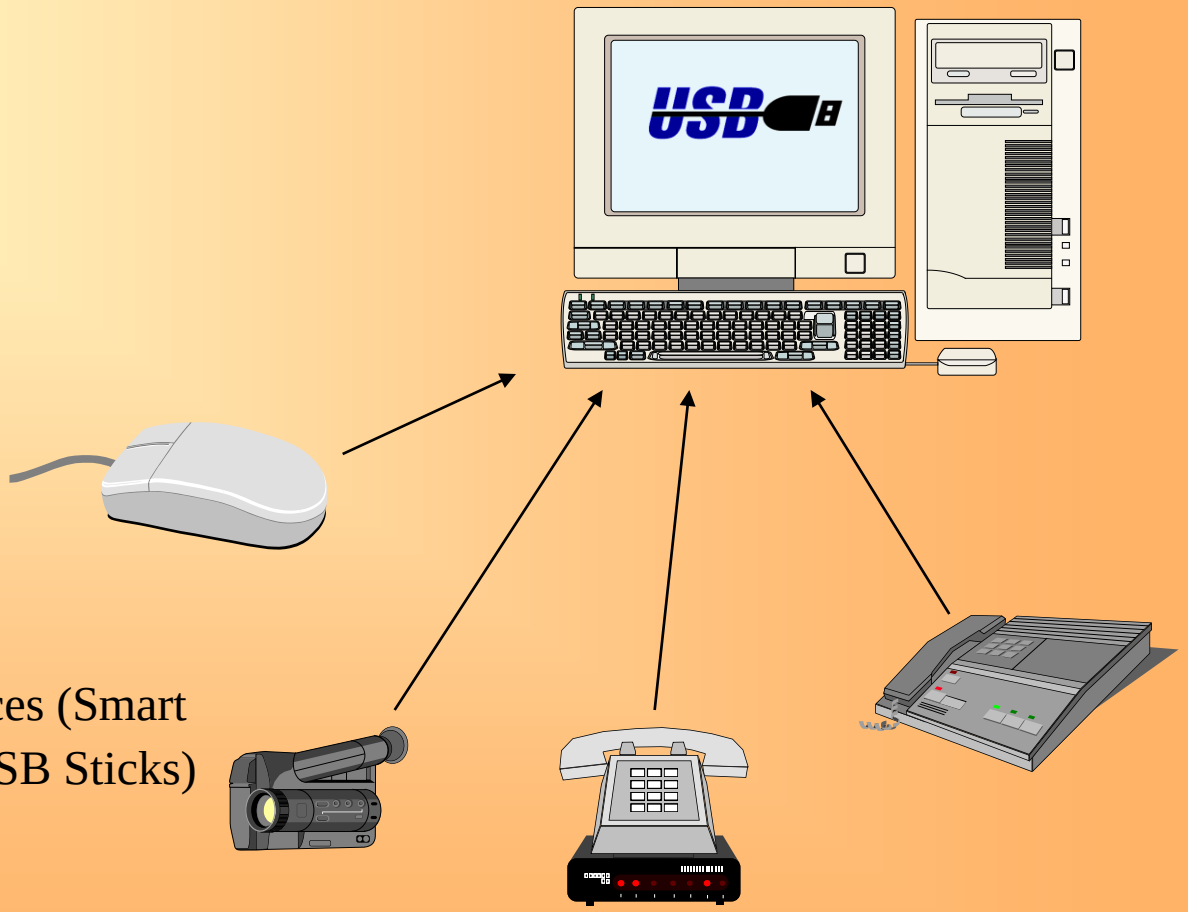
Device

- Classic „end-user device“
- Only one upstream-port
- Can be high-, full- or low-speed device
- Self powered and/or bus powered devices
- Bus powered devices can be divided into:
  - low powered: I_{max}= 100 mA
  - high powered: $_{Imax}$= 500 mA

Device

USB-Device

Keyboard / Mouse / Joystick
Camera
Microphone / Loudspeaker
Modem / Fax / ISDN /DSL
Telephone
Printer
Scanner
Measurement-Devices
external Mass-Storage-Devices (Smart cards, HD, Flash memory, USB Sticks)



Cable & Connectors

Cable

- **Full speed**:
  - Twisted pair (min. 28 AWG*) for data lines, shielded
  - Non-drilled pair (min. 28 AWG) for power lines
  - Up to 5m length
  - Delivered with A-/B-connectors
- **Low speed**:
  - 2 non-twisted pairs (min. 28 AWG) for data and power lines
  - Up to 3m length
  - Cable is always fix connected to the device !
  - Upstream end with A-connector

*Measurement of wire's cross section, as defined by the American Wire Gauge standard

Cable

- 30 ns max. cable-delay must be guaranteed !!!

AWG	resistance	max. length
28	0.232 Ohm/m	0.81 m
26	0.145 Ohm/m	1.31 m
24	0.091 Ohm/m	2.08 m
22	0.057 Ohm/m	3.33 m
20	0.036 Ohm/m	5.00 m

- Used Colors

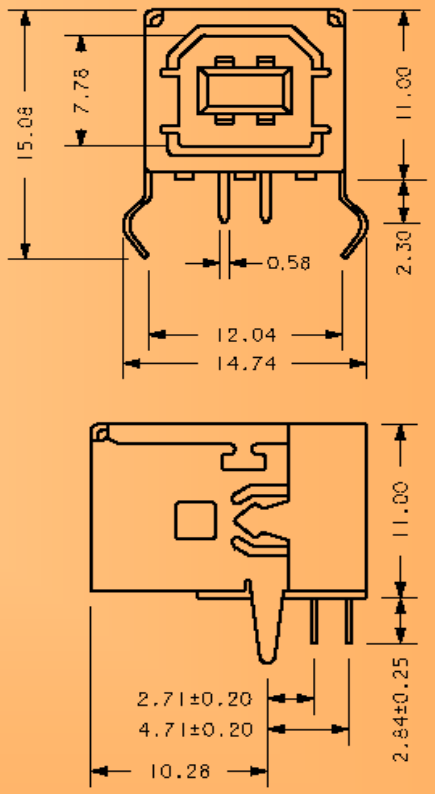
VCC	Data +	Data -	Ground
red	green	white	black

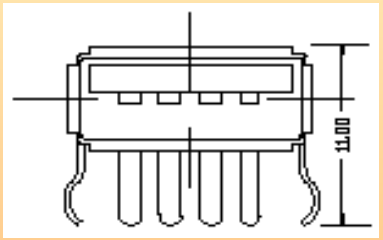
Connectors

⇨ Standard connector A/B - series

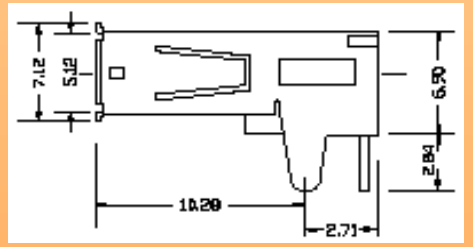
- VCC and GND with longer contacts
- Usage: A - upstream / B - downstream

B-series





A-series



Connectors

- USB is present today in a lot of small mobile devices
  - Smaller Connectors required
  - “Mini-B connector Engineering Change Notice to the USB 2.0 specification.”
  - “Micro USB Specification to the USB 2.0 Specification”, Revision 1.01 from April, 2007

Mini-B Connector

- Smaller size (6.9mm * 3.1mm)
  - Better fitted for portable devices
- Additional ID pin (unconnected)
- Compliant to USB-Spec.: 500 mA power / 480 Mbit/sec High-Speed
- Increased durability: 5000 insertion/extraction cycles

Micro-USB Connectors

- Portable devices have become more and more thin – current Mini-USB does not fit within constraints of future designs
- Micro-USB Specification was released in 2007 (Version 1.01)
- Again smaller size (6.9mm * 1.85 mm)
- Defines additional connectors:
  - Micro-B plug and receptacle (black)
  - Micro-AB receptacle (gray)
  - Micro-A plug (white)

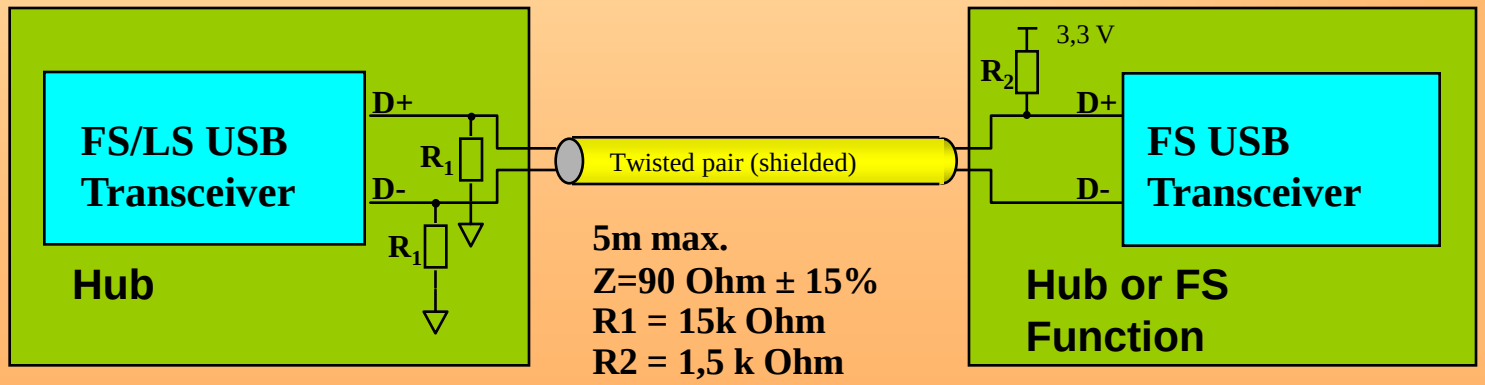
- Compliant to USB 2.0 Spec.: 500 mA power / up to 480 Mbit/sec High-Speed
- Increased durability: 10.000 cycles of insertion/extraction

Low-Level-Services

Connect/Disconnect - Full speed

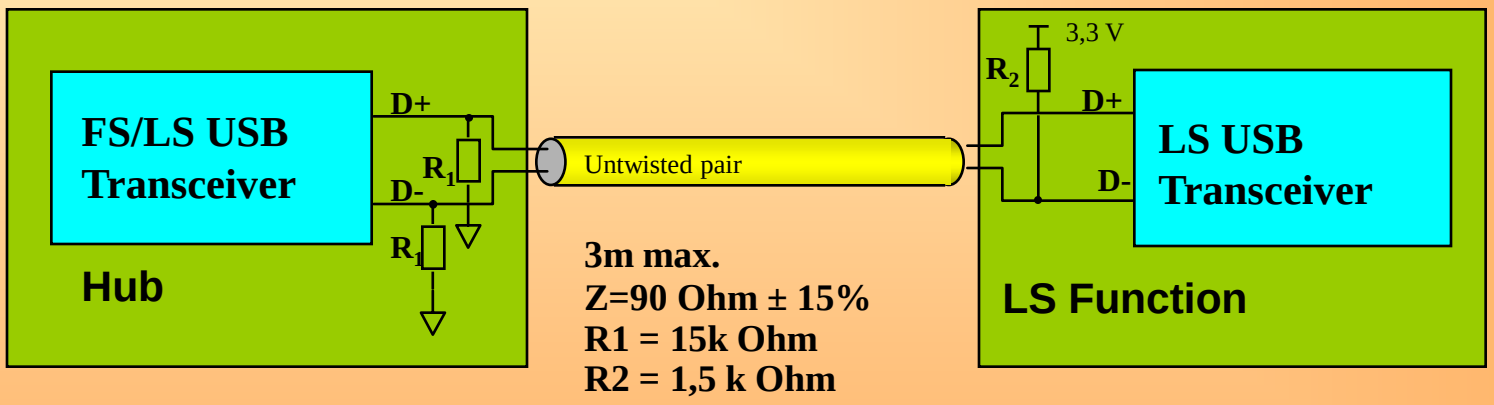
- Disconnect: hub recognizes both data lines at low (15k pull-down-resistors at D+ and D-)
- Connect: D+ line goes high (1k5 pull-up to 3,3 V in function)

- Full speed detection: pull-up resistor at D+ line



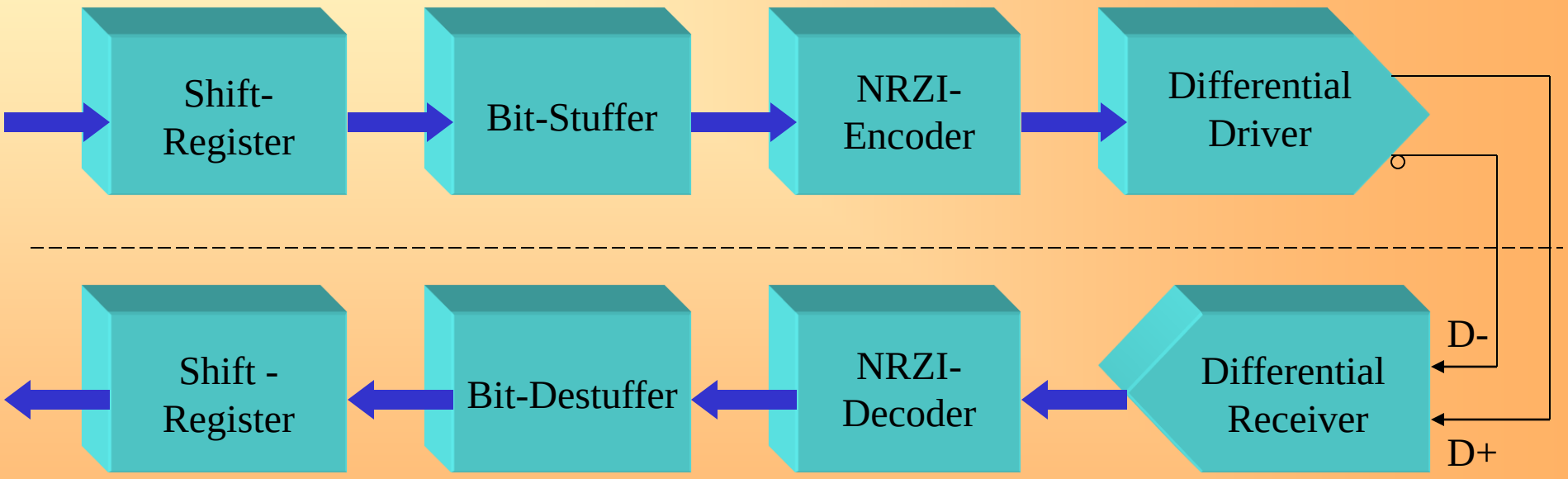
Connect-Disconnect - Low speed

⇨ Low speed detection via pull-up resistor at D- data line



Low-Level-Data-Coding

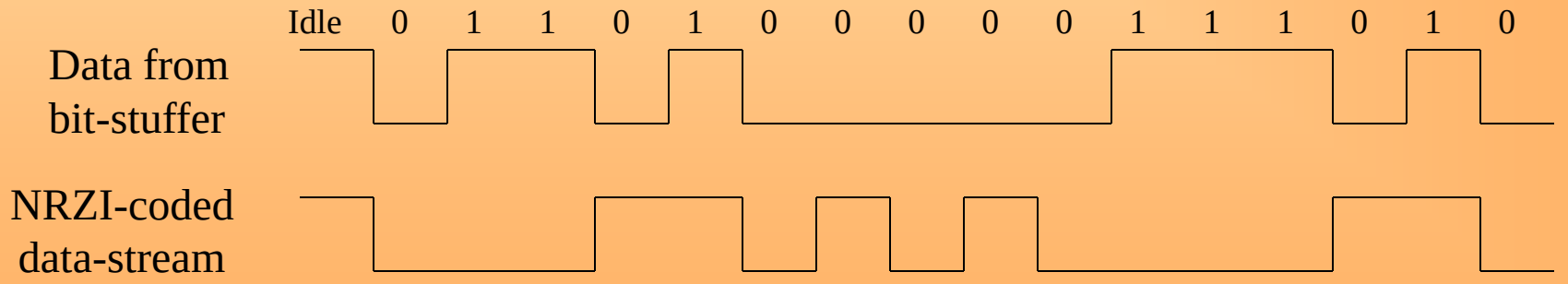
Data-flow in transmitter



Data-flow in receiver

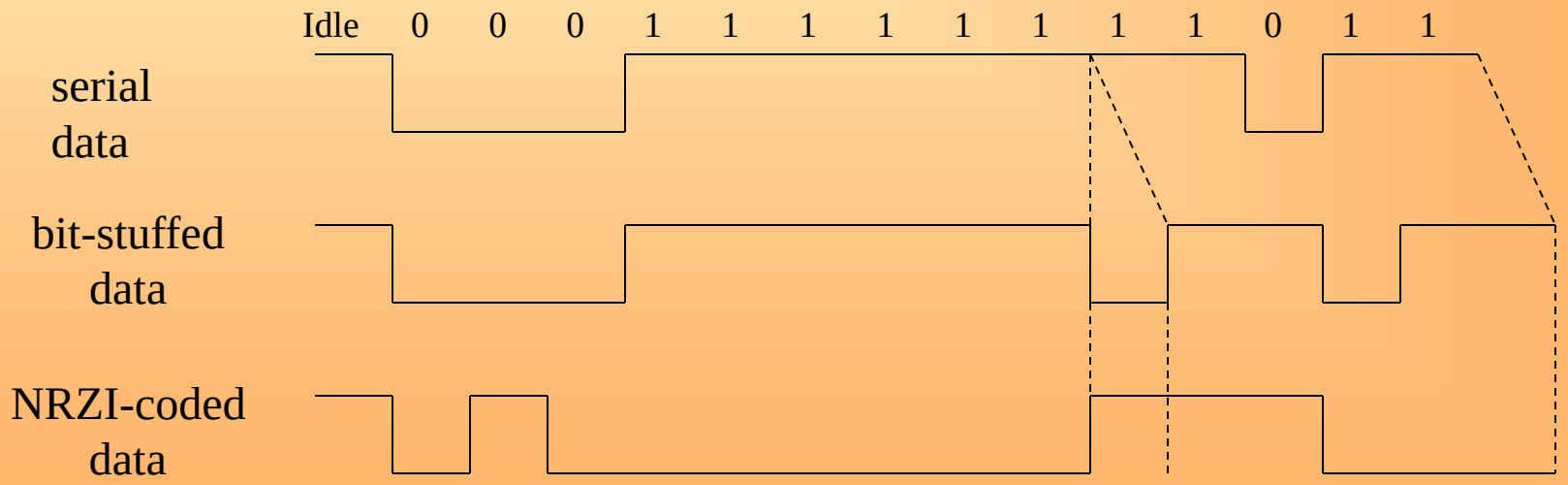
NRZI-Coding

- NRZI = Non Return to Zero, Inverted
  - Data-bit = „0“ : forces change of polarity in NRZI-stream
  - Data-bit = „1“ : no change of polarity
- Encoding of clock into data - no extra clock necessary
- Implies usage of a synchronization-header (SYNC-field)
- Problem: a series of many „1“ in the data-stream can force a lost of synchronization → solution: bit-stuffing



Bit-Stuffing

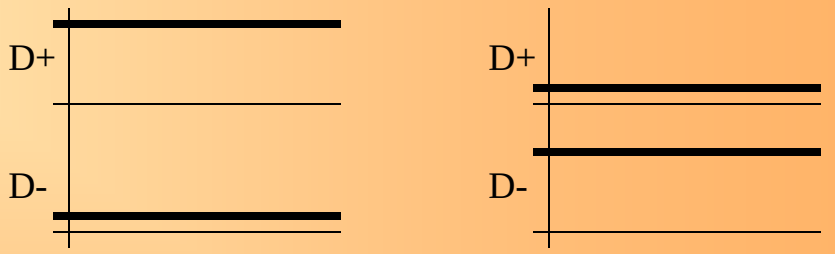
- Insertion of an additional „0“ after 6 consecutive „1“ in the data-stream → forces a change of polarity in NRZI-stream
- Receiver must detect additional „0“ after 6 consecutive „1“ and eliminate



Bus-Events (I)

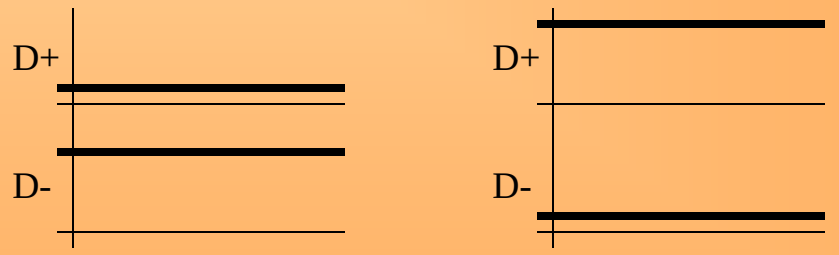
- **Differential “1”** : (D+) - (D-) > 200mV
- **Differential “0”** : (D-) - (D+) > 200mV

- **‘J’ State**:
  Full speed: differential “1”
  Low speed: differential “0”
  Idle State



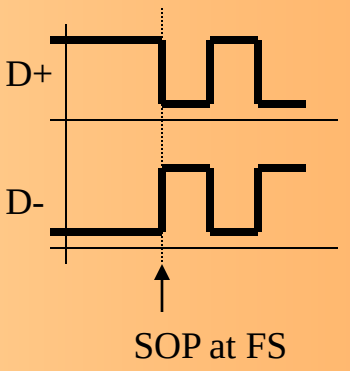
FS LS

- **‘K’ State** (inverted ‘J’ state) :
  Full speed: differential “0”
  Low speed:differential “1”
  Resume State



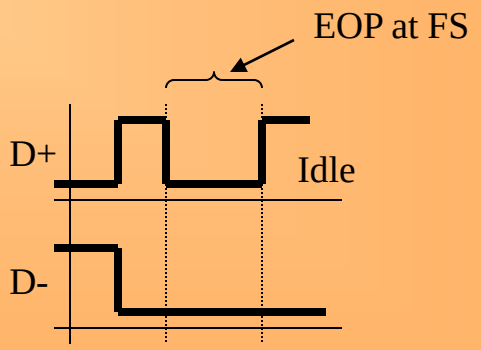
Bus-Events (II)

- **Start of Packet (SOP)** :
  Data lines switch from idle (=J-state) to ‘K’ state

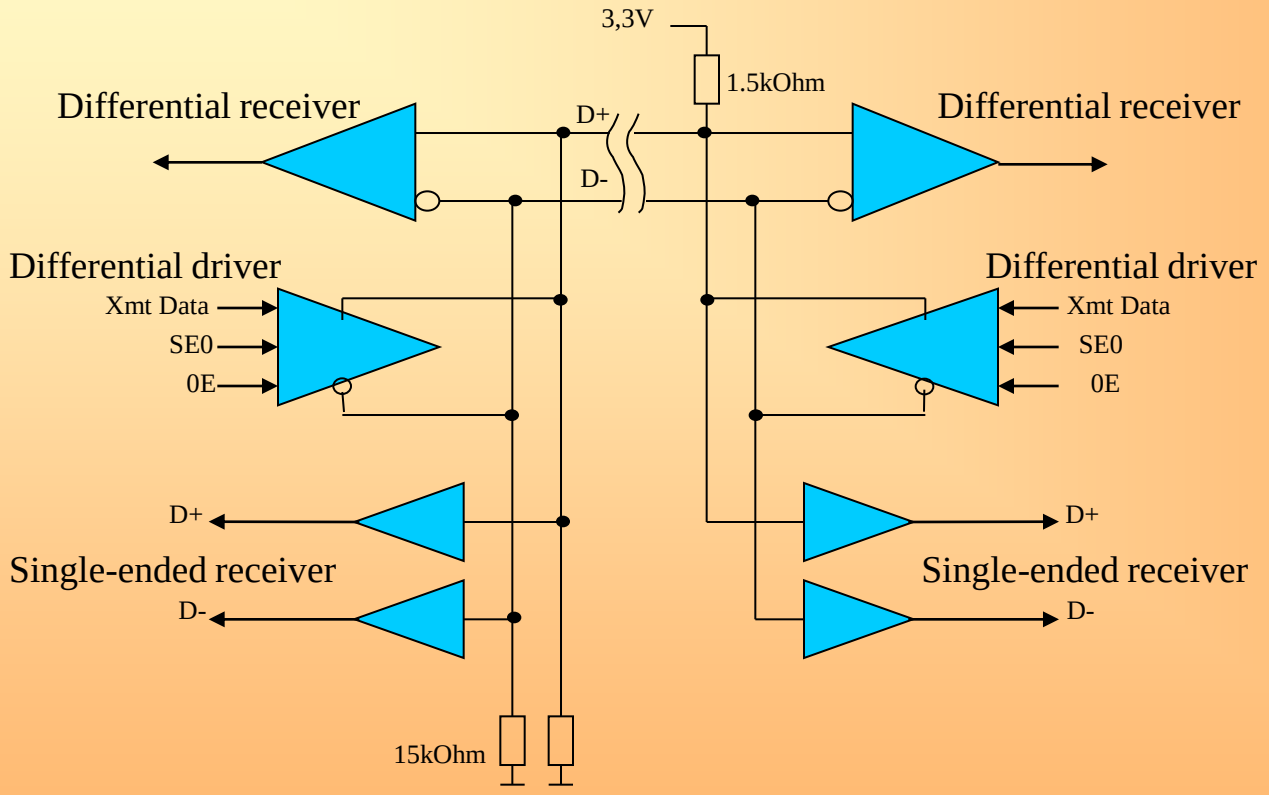


- **End of Packet (EOP)** :
  Driver:
  D+ und D- < $V_{SE\ (min)}$ for 2 bit-times followed by (1 driven) idle-state
  Receiver:
  D+ und D- < $V_{SE\ (min)}$ for > 1 bit-time followed by idle-state (J)

- **Single ended zero** (SE0) s**tate**
  End of Packet recognition (EOP)
  USB-Reset if longer than 2,5µs

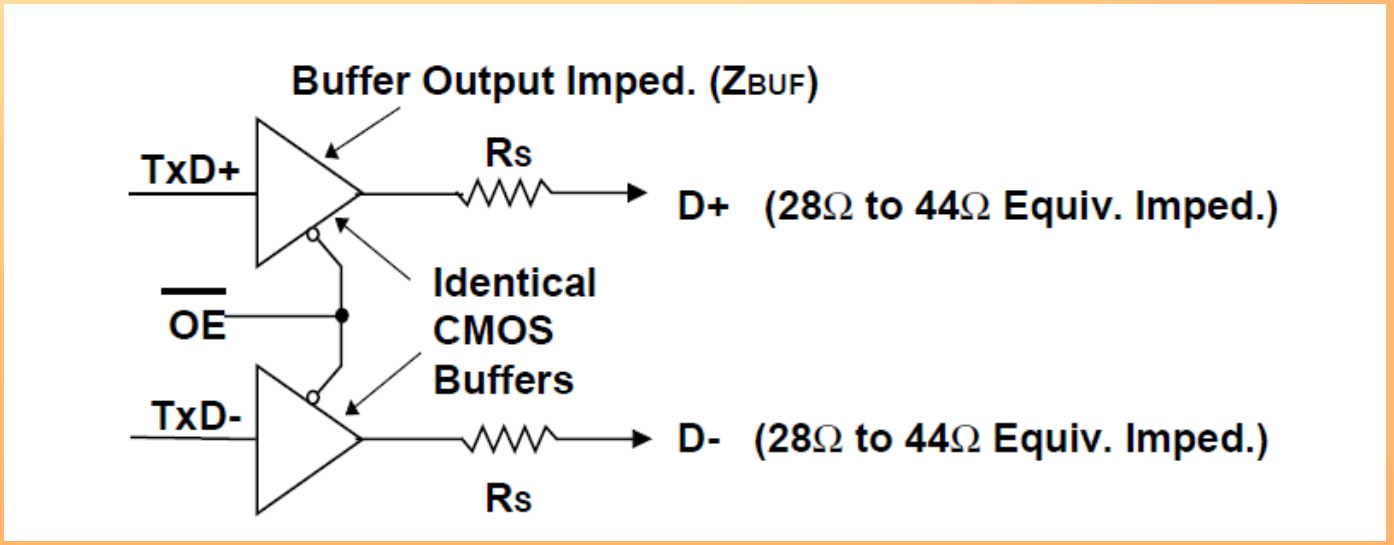


Electrical Signal Interface



Differential Driver

- Programmable slew-rate
- Full-speed: 4 .. 20 ns rise/fall-time
- Low-speed: 75 .. 300 ns rise/fall-time
- Voltage level: 0 / 3,3V
- Exception: Single-Ended-Zero (SE0) = both lines are driven with low



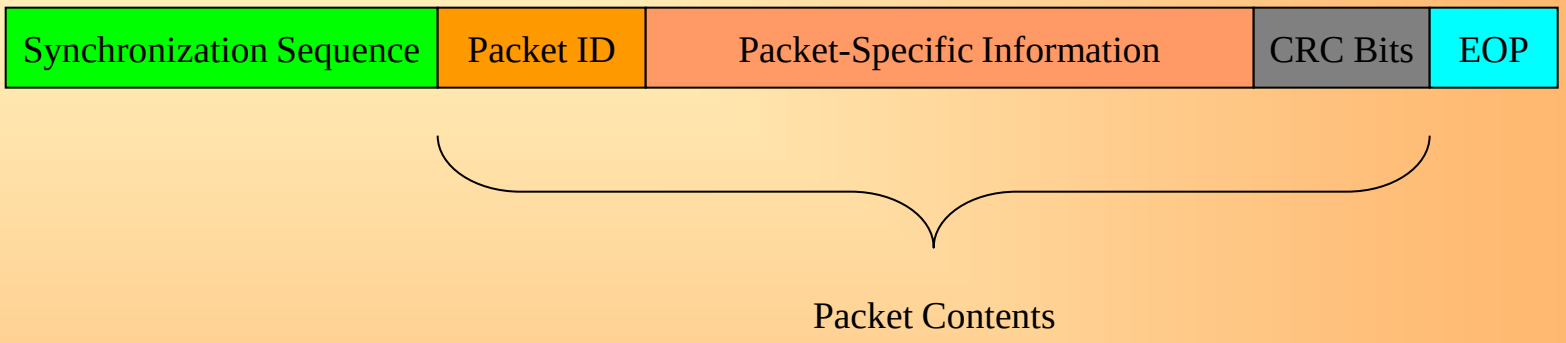
Receiver

- **Differential receiver:**
- Generation of a single signal
- Input range: -0.5 V ... 3.8 V
- Sensitivity: not less than 200 mV between D+ and D-

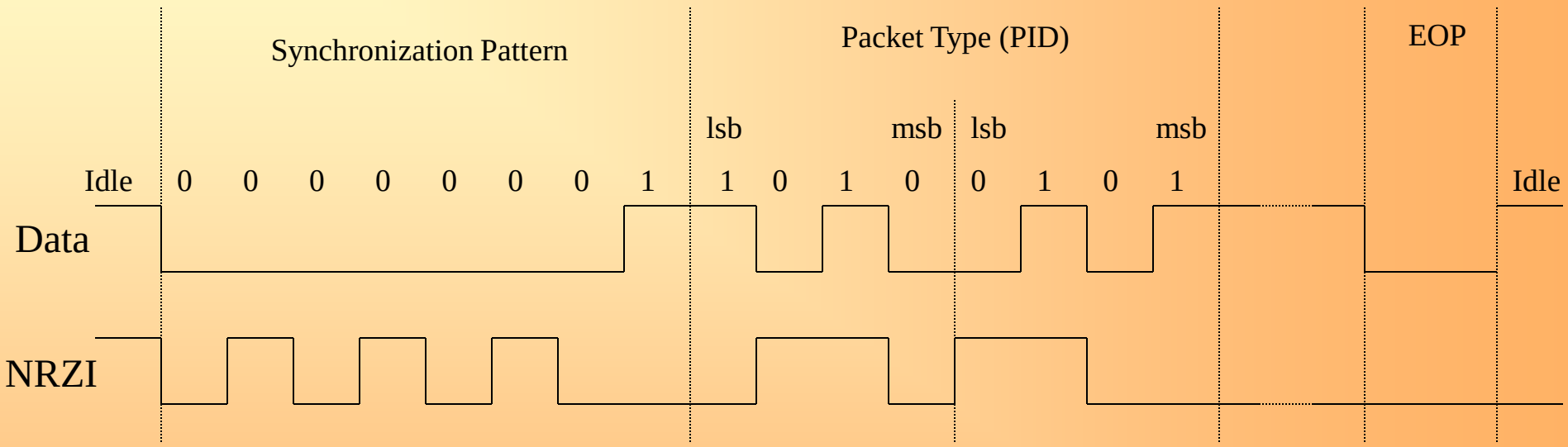
- **Single-ended receivers:**
- Detection of SE0-exception (SE0 ⇒ EOP, Reset)
- Smith-trigger-characteristic with hysteresis: 0.8V / 2.0V

Packets and USB Device Framework

Packet Format



Packet Format (2)



Token Packets

SYNC	SOF	Frame #	CRC5	EOP
00000001	0xA5	0x7FD	0x1F	2

Start-of-Frame-Token

SYNC	SETUP	ADDR	EP	CRC5	EOP
00000001	0x2D	0x01	0x00	0x17	2

Setup-Token

SYNC	IN	ADDR	EP	CRC5	EOP
00000001	0x69	0x03	0x01	0x07	2

In-Token

SYNC	OUT	ADDR	EP	CRC5	EOP
00000001	0xE1	0x02	0x02	0x01	2

Out-Token

- ⇨ Every packet starts with a SYNC-field (receiver-synchronization via PLL)
- ⇨ Packet-identification by 8-bit-PID
- ⇨ SOF: 11-bit-frame-number (wrap around-mode)
- ⇨ SETUP/IN/OUT: 7-bit-device-address and 4-bit-endpoint-number
- ⇨ Data protected by 5-bit CRC
- ⇨ Every packet ends with EOP

Data Packets

SYNC	DATA0	DATA	CRC16	EOP
00000001	0xC3	00 11 22 33 44 55 66 77	0xCBA8	2

Data-0-Token

SYNC	DATA1	DATA	CRC16	EOP
00000001	0x4B	88 99 AA BB CC DD EE FF	0x8705	2

Data-1-Token

- PID: DATA0 und DATA1
  (Data-toggle-protocol for non-isochronous-pipes)
- Length of data-field depends on maximum packet size:
  - Isochronous: up to 1023 byte
  - Bulk / Control : 8 / 16 / 32 / 64 byte
  - Interrupt : up to 8 byte LS / 64 byte FS
- Data protected by 16-bit CRC
- Every packet ends with EOP

Handshake Packets

SYNC	ACK	EOP
00000001	0xD2	2

ACK-Token

SYNC	NAK	EOP
00000001	0x5A	2

NAK-Token

SYNC	STALL	EOP
00000001	0x1E	2

STALL-Token

- PID ACK: Data accepted and transmitted without any errors
- PID NAK: Device not ready to accept or send data
- PID STALL: Critical error - endpoint is blocked
- Handshake-packets not protected with CRC
- Every packet ends with EOP

Special Packets & Static Events

SYNC	PRE
00000001	0x3C

- PID PRE: Preamble Token sent by host before sending a LS-transmission
  - Exception - packet not terminated by EOP !

SYNC	PRE	Idle	SYNC	SETUP	ADDR	EP	CRC5	EOP
00000001	0x3C	4	00000001	0x2D	0x01	0x00	0x1F	2

Full-Speed Low-Speed

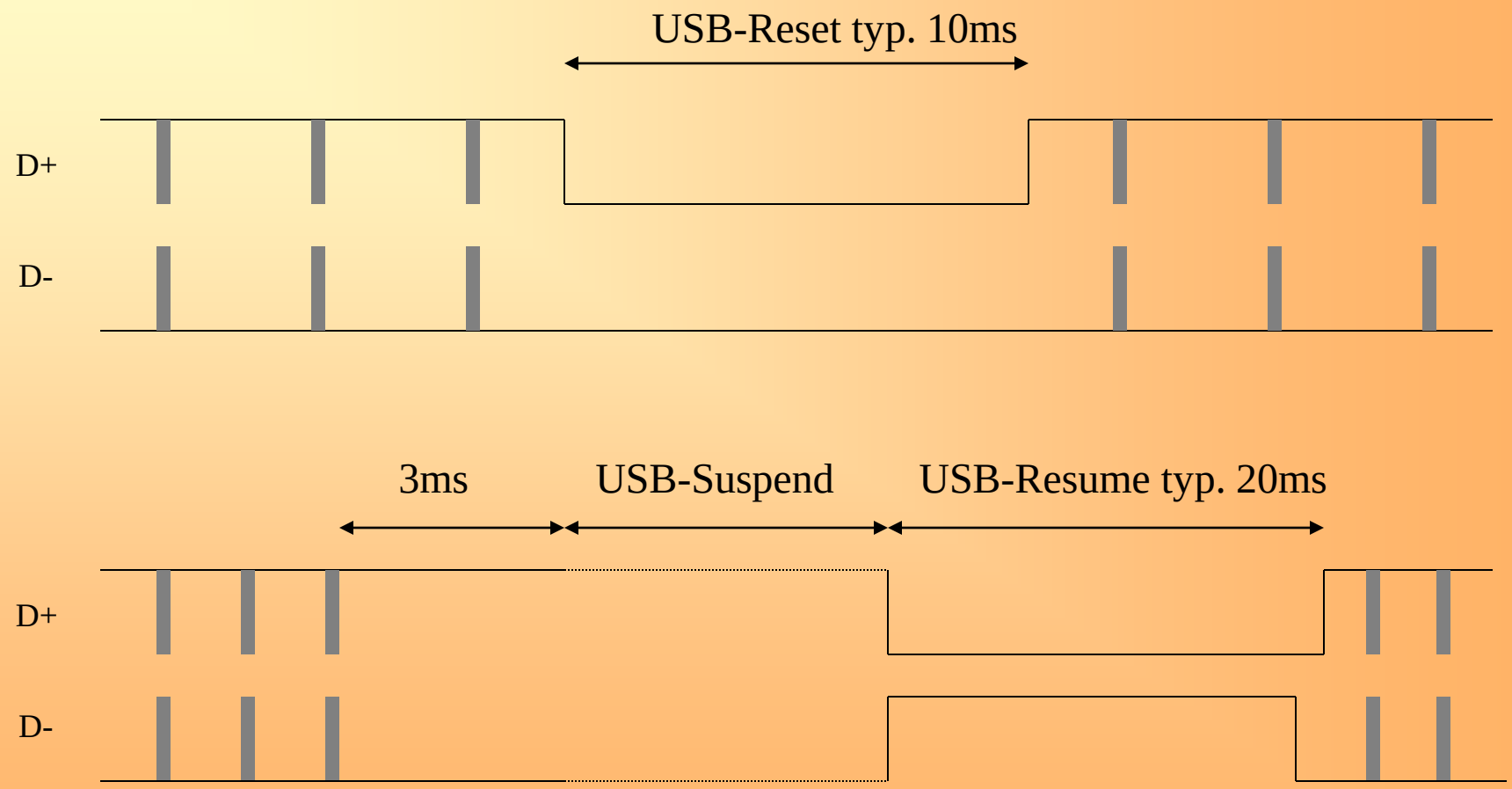
- USB-RESET: both lines are driven low longer than 2,5 µs (10 ms in LS)
- SUSPEND: Entry into suspend state if no data-traffic is available > 3 ms
- RESUME: Wake-up-signal to leave suspend state (driven inverted idle state)

RESET
10 ms

SUSPEND
x ms

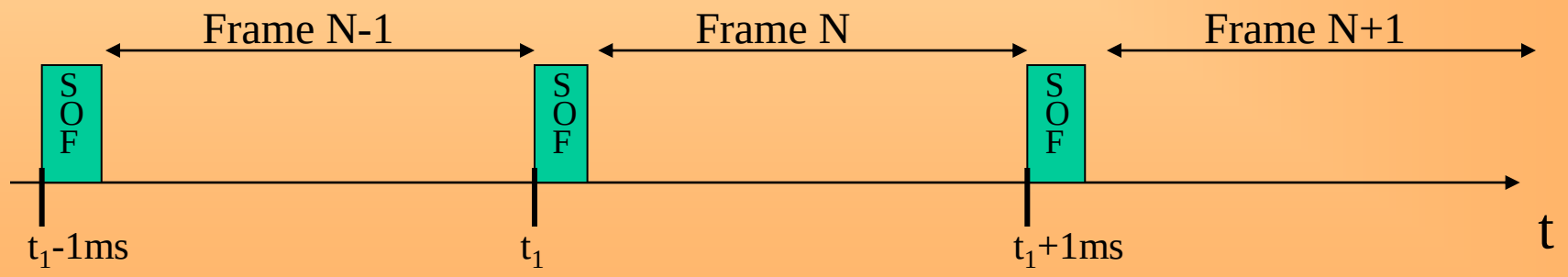
RESUME
x ms

Reset / Suspend / Resume



Frames

- Host splits bandwidth in 1 ms pieces → frames
- Every frame starts with a Start-of-Frame-Token (SOF)
- Host uses an internal 32-bit frame counter
- SOF-token includes the lower 11 bits of the frame counter



Empty FS-Framework

Idle	SYNC	SOF	Frame #	CRC5	EOP
11966	00000001	0xA5	0x7FD	0x1F	2
Idle	SYNC	SOF	Frame #	CRC5	EOP
11966	00000001	0xA5	0x7FE	0x1D	2
Idle	SYNC	SOF	Frame #	CRC5	EOP
11966	00000001	0xA5	0x7FF	0x02	2
Idle	SYNC	SOF	Frame #	CRC5	EOP
11966	00000001	0xA5	0x000	0x08	2
Idle	SYNC	SOF	Frame #	CRC5	EOP
11966	00000001	0xA5	0x001	0x17	2
Idle	SYNC	SOF	Frame #	CRC5	EOP
11966	00000001	0xA5	0x002	0x15	2
Idle	SYNC	SOF	Frame #	CRC5	EOP
11966	00000001	0xA5	0x003	0x0A	2
Idle	SYNC	SOF	Frame #	CRC5	EOP
11966	00000001	0xA5	0x004	0x14	2

12 MHz / 1000 frames =
12000 bit-times per frame

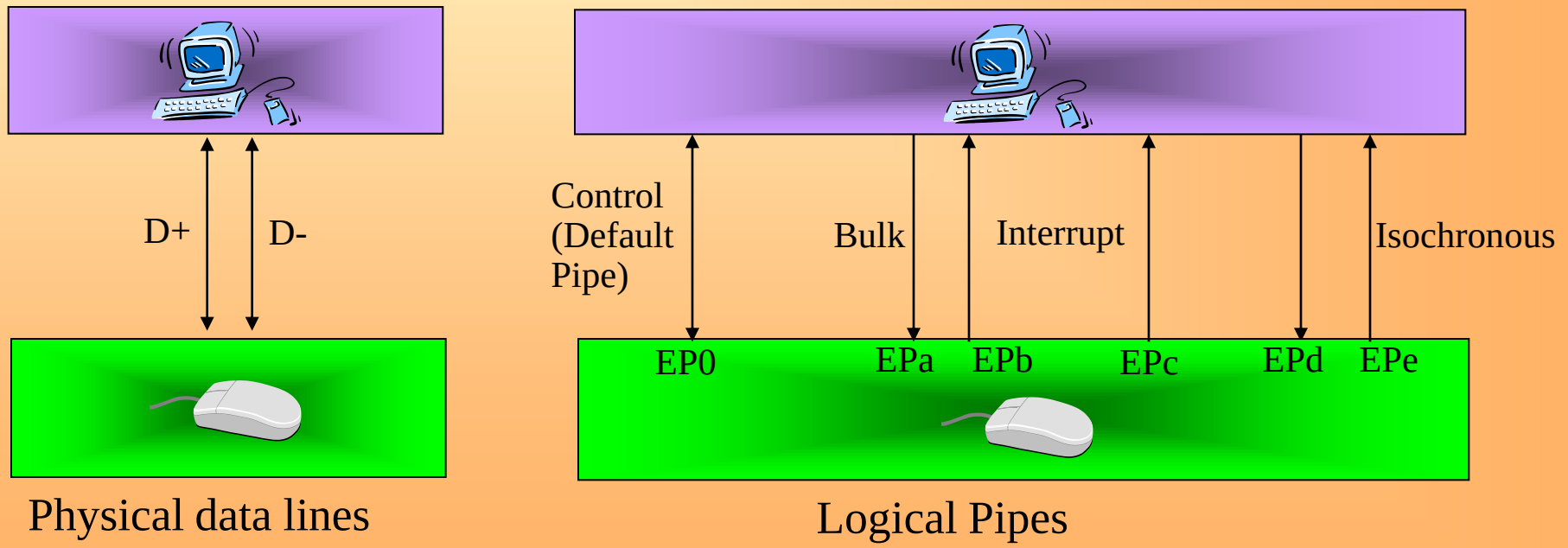
11966 bit idle or data
+ 8 bit SYNC of SOF
+ 8 bit PID SOF
+ 11 bit frame-number
+ 5 bit CRC
+ 2 bit EOP

12000 bits in one frame

Transfer Types

Endpoints and Pipes

- Endpoint (EP): logical end of a data-pipe, typical physical implementation: FIFO
- Pipe: logical data-channel sourced or terminated by an endpoint
- Pipe is always mapped to a transfer type



Control Transfer

- Endpoint 0 is always control endpoint
- One-and-only pipe which works bi-directional
- 10% of bandwidth reserved for control-transfers
- Support of error-detection and -correction

- Maximum packet sizes:
  - 8, 16, 32, 64 bytes - full speed
  - 8 bytes - low speed

- Usage: enumeration & controlling of a USB device

Interrupt Transfer

- Usable for low bandwidth demands
- Pipe will be polled by the host
- Up to 90% of bandwith reserved for periodic transfers
- Known latency with error-detection and -correction

- Maximum packet size:
  - 1 .. 8 byte LS
  - 1 .. 64 byte FS
- Usage:
  - Status-polling
  - Coordinates (mouse) / keystrokes (keyboard)
  - Status-signaling (LEDs in keyboard)

Isochronous Transfer

- Guaranteed bandwidth allows real-time transmission of data
- USB endpoint accessed exactly ones per frame
- Up to 90% of bandwith is reserved for periodic transfers
- Delayed data = useless data !
- No handshake-packets in protocol
- Data protected by CRC16 - but no error-correction

- Maximum packet size: 1...1023 byte

- Usage:
  - Audio / Video
  - ISDN / Modem

Bulk Transfer

- Access only if bandwidth available
- Error-detection and -correction
- Up to 19 transactions per frame possible

- Maximum packet size: 8, 16, 32, 64 byte

- Usage:
  - Save transmission of large data
  - Printer / scanner / mass-storage devices

Overview: Transfer Types

Transfer-type	Maximum Packet Size	Error-correction	Bus-access	Typical usage
Control	FS: 8,16,32,64 LS: 8	yes	10 % guaranteed	Device-enumeration / configuration
Interrupt	FS: 1..64 LS: 1..8	yes	Up to 90 %	Small amounts of data (mouse, keyboard, error- & status-signaling
Isochronous	FS: 1..1023	no	Up to 90 %	Real-time data: compr. video, audio, ISDN / modem
Bulk	FS: 8,16,32,64	yes	Only if bandwidth is available	Large amounts of non-time-critical data: printer / scanner / mass-storage-devices

Overview: Transfer Bandwidth

	Full-Speed	Low-Speed
Control Data	protocol overhead: 45 Bytes transactions/frame: 1/13* max size: 64 Byte max bandwidth: 0.5/6.6 Mb/s*	protocol overhead: 46 Bytes transactions/frame: 1/3* max size: 8 Byte max bandwith: 64/192 kb/s*
Interrupt Data	protocol overhead: 13 Bytes transactions/frame: 1 max size: 64 Byte max bandwidth: 512 kb/s	protocol overhead: 13 Bytes transactions/frame: 1 per 8 fr. max size: 8 Byte max bandwith: 8 kb/s
Isochronous Data	protocol overhead: 9 Bytes transactions/frame: 1 max size: 1023 Byte max bandwidth: 8.2 Mb/s	
Bulk Data	protocol overhead: 13 Bytes transactions/frame: 19 max size: 64 Byte max bandwidth: 9.7 Mb/s	

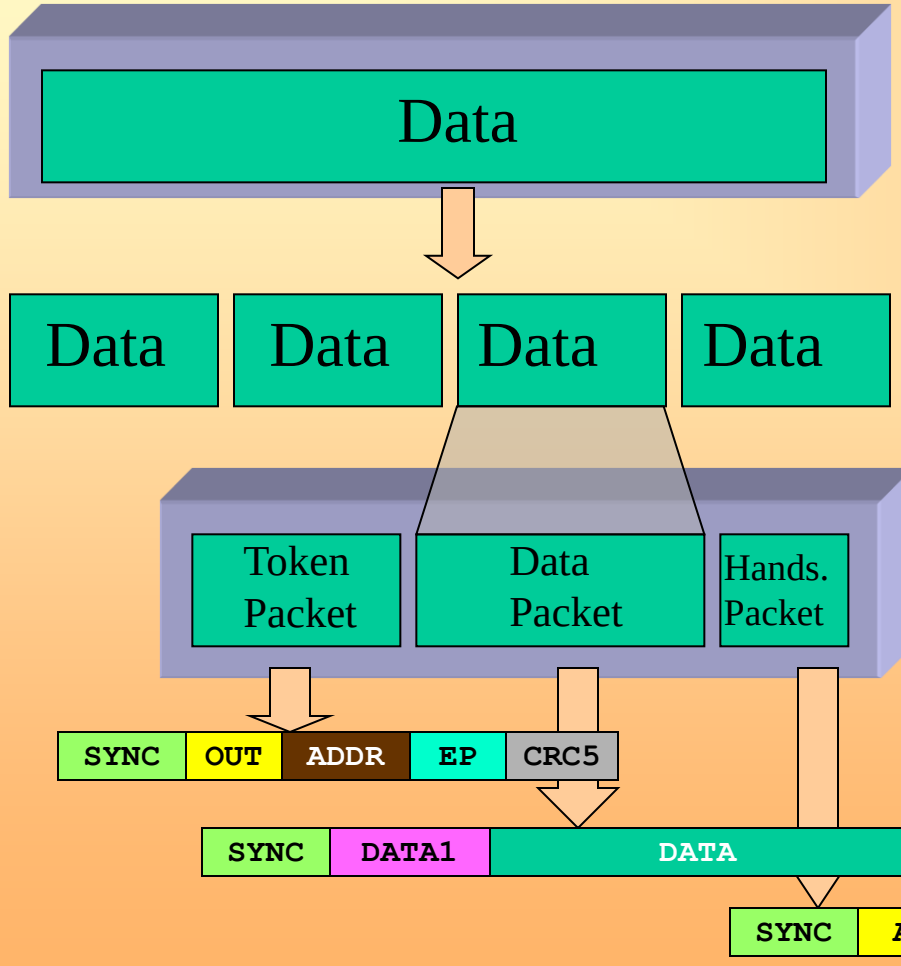
* = Universal- / Open-Host-Controller

Message Pipes vs. Stream Pipes

- Stream pipes: Data moving through a pipe has no USB-defined structure
- Message pipes: Data moving through a pipe has some USB-defined structure, e.g.
  - Control Pipe EP0
  - Mass Storage Device Class IN/OUT endpoints
  - Still Image Device Class IN/OUT endpoints
- For non-isochronous pipes, smaller payload of maximum packet size will be interpreted as „short packet“
  - Terminate the transfer, even if provided buffer is not completely filled
  - May be used as an in-band delimiter to indicate “end of unit of data”

Transactions

General Scope



Data block to transmit

Size of data-packets = Maximum Packet Size of receiving EP

Transaction level

Packet level

Phases of an USB-Transaction

Idle	SYNC	OUT	ADDR	EP	CRC5	EOP
4	00000001	0xE1	0x02	0x02	0x01	2

Token-Phase

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	00 11 22 33 44 55 66 77	0xCBA8	2

Data-Phase

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Handshake-Phase

- 1st Token-Phase: Always initiated by host (single-master-bus)
- 2nd Data-Phase: Depends on token transmitted in first phase
  - SETUP / OUT-Token : Data transfer from host to function
  - IN-Token : Data transfer from function to host
- 3rd Handshake-Phase: Transmitted by data receiver

Out-Transactions

Idle	SYNC	OUT	ADDR	EP	CRC5	EOP
4	00000001	0xE1	0x02	0x02	0x01	2

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	00 11 22 33 44 55 66 77	0xCBA8	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Out-Transaction without error

Idle	SYNC	OUT	ADDR	EP	CRC5	EOP
4	00000001	0xE1	0x02	0x02	0x01	2

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	00 11 22 33 44 55 66 77	0xCBA8	2

Idle	SYNC	NAK / 0x5A / STALL / 0x1E	EOP
5	00000001		2

Out-Transaction to busy device

Out-Transaction to device with stalled endpoint

IN-Transactions

Idle	SYNC	IN	ADDR	EP	CRC5	EOP
4	00000001	0x69	0x02	0x02	0x01	2

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	00 11 22 33 44 55 66 77	0xCBA8	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

In-Transaction without error

Idle	SYNC	IN	ADDR	EP	CRC5	EOP
4	00000001	0x69	0x02	0x02	0x01	2

Idle	SYNC	NAK / 0x5A / STALL / 0x1E	EOP
5	00000001		2

In-Transaction to device which has no data

In-Transaction to device
with stalled endpoint

Isochronous-Transfer

Idle	SYNC	SOF	Frame #	CRC5	EOP
11023	00000001	0xA5	0x001	0x17	2

Idle	SYNC	OUT	ADDR	EP	CRC5	EOP
4	00000001	0xE1	0x02	0x02	0x01	2

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	00 11 22 33 44 55 66 77	0xCBA8	2

Idle	SYNC	SOF	Frame #	CRC5	EOP
11023	00000001	0xA5	0x002	0x15	2

Idle	SYNC	OUT	ADDR	EP	CRC5	EOP
4	00000001	0xE1	0x02	0x02	0x01	2

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	88 99 AA BB CC DD EE FF	0x8705	2

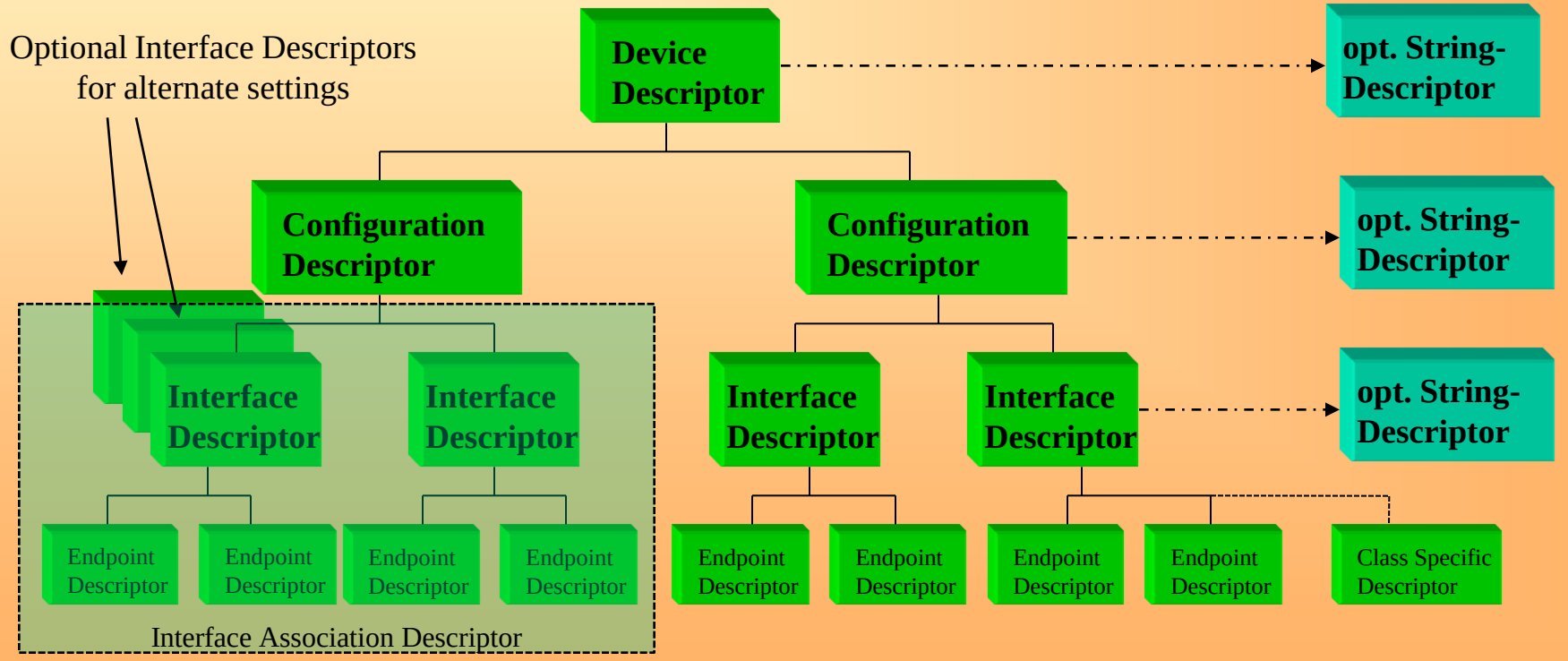
Idle	SYNC	SOF	Frame #	CRC5	EOP
11023	00000001	0xA5	0x003	0x0A	2

- ⇨ No Data-Toggle used, only DATA-0 is valid
- ⇨ No handshake phase
- ⇨ No error-recovery

Descriptors

Hierarchy of Descriptors

- Descriptors explain functionality / attributes of a USB device
- Host asks for the descriptors during enumeration phase
- Information leads to loading of dedicated device-drivers



Descriptor Types and Format

- Descriptor type
  - 01h = Device-Descriptor
  - 02h = Configuration-Descriptor
  - 03h = String-Descriptor
  - 04h = Interface-Descriptor
  - 05h = Endpoint-Descriptor
- 2 Byte each (Vendor-ID, Packet size, ...)
  - Little-Endian format (low byte first, then high byte)
- EP-Number, -Direction, -Transfer type can be selected freely (except for EP 0)

Example MSC Descriptor Settings

- USB Memory Stick
  - Bulk-Only Protocol
  - Bus-Powered, current consumption max. 250 mA
  - No support for Remote-Wakeup
- Control IN / OUT-Endpoint
  - Maximum Packet Size 64 Byte
- Bulk IN-Endpoint
  - Maximum Packet Size 64 Byte
- Bulk OUT-Endpoint
  - Maximum Packet Size 64 Byte

Device-Descriptor

Offset	Field	Length	Type	Description	Example
0	Descr.-Length	1	Number	Size of this descriptor in byte	12h
1	Descriptor-Type	1	Constant	DEVICE-Descriptor-Type = 01 h	01h
2	USB-Spec.	2	BCD	USB-Spec. Release in „Binary-Coded Decimal“ 2.00	00, 02h
4	Device-Class	1	Class	Class-Code	00h
5	Device-Subclass	1	Subclass	Subclass-Code	00h
6	Device-Protocol	1	Protocol	Protocol-Code	00h
7	Max.Packet Size	1	Number	Max. Packet Size of EP0 = 64 byte	40h
8	Vendor	2	ID	Vendor-ID (emsys = 0CC4h)	C4,0Ch
10	Product	2	ID	Product-ID (example = 0103h)	03,01h
12	Device-Release	2	BCD	Device-Release (example = 1.27)	27,01h
14	Index Manufact.	1	Index	Index of String-Descriptor 'Manufacturer'	01h
15	Index Product	1	Index	Index of String-Descriptor 'Product'	02h
16	Index Serial Nr.	1	Index	Index of String-Descriptor 'Serial-Number'	03h
17	# Configurations	1	Number	Number of the supported configurations	01h

Configuration-Descriptor

Offset	Field	Length	Type	Description	Example
0	Descr.-Length	1	Number	Size of this descriptor in byte	09h
1	Descriptor-Type	1	Constant	Configuration descriptor-Type = 02 h	02h
2	Total Length	2	Number	Total length of **all** descriptors for this configuration	20, 00h
4	Num. Interfaces	1	Number	Number of supported interfaces	01h
5	Conf.-Value	1	Number	Value to activate this configuration	01h
6	Configuration	1	Index	Index of string descriptor 'Configuration"	04h
7	Attributes	1	Bitmap	Configurations attribute * D7 = 1 * D5 = Remote-Wakeup = 0 D6 = Self Powered = 0 D4..D0 = Reserved	80h
8	Max. Power	1	Number	Power consumption in 2mA units (250mA/2mA = 125)	7Dh

Get Configuration-Descriptors answers with a collection of:

Configuration-Descriptor, Interface-Descriptor, Class-Specific-Descriptor, Endpoint-Descriptor(s), next Interface-Descriptor, ...

Total Length is the length of this collection

Interface-Descriptor

Offset	Field	Length	Type	Description	Example
0	Descr.-Length	1	Number	Size of this descriptors in byte	09h
1	Descriptor-Type	1	Constant	Interface-Descriptor-Type = 04 h	04h
2	Interface-Num.	1	Number	Number of this interface	00h
3	Alt. Setting	1	Number	Alternate setting of this interface	00h
4	Num. Endpoints	1	Number	Number of assigned endpoints for this interface	02h
5	Int. Class	1	Class	Interface-Class-Code: Mass Storage	08h
6	Int. Subclass	1	Subclass	Interface-Subclass-Code: SCSI transparent command set	06h
7	Int. Protocol	1	Protocol	Interface-Protocol-Code: Bulk-Only Transport	50h
8	Interface	1	Index	Index of String-Descriptor 'Interface'	05h

Endpoint-Descriptors of EP 1 IN/OUT

Offset	Field	Length	Type	Description	Example
0	Descr.-Length	1	Number	Size of this descriptor in byte	07h
1	Descriptor-Type	1	Constant	Endpoint-Descriptor-Type = 05 h	05h
2	Endp.-Address	1	Number	Endpoint-Address: IN- Endpoint number 2	82h
3	Attributes	1	Bitmap	Transfer type - Bulk	02h
4	Max.Packet Size	2	Number	Maximum Packet Size of this endpoint = 64 byte	40,00h
6	Interval	1	Number	Polling Interval for Bulk-EP‘s not relevant	00h

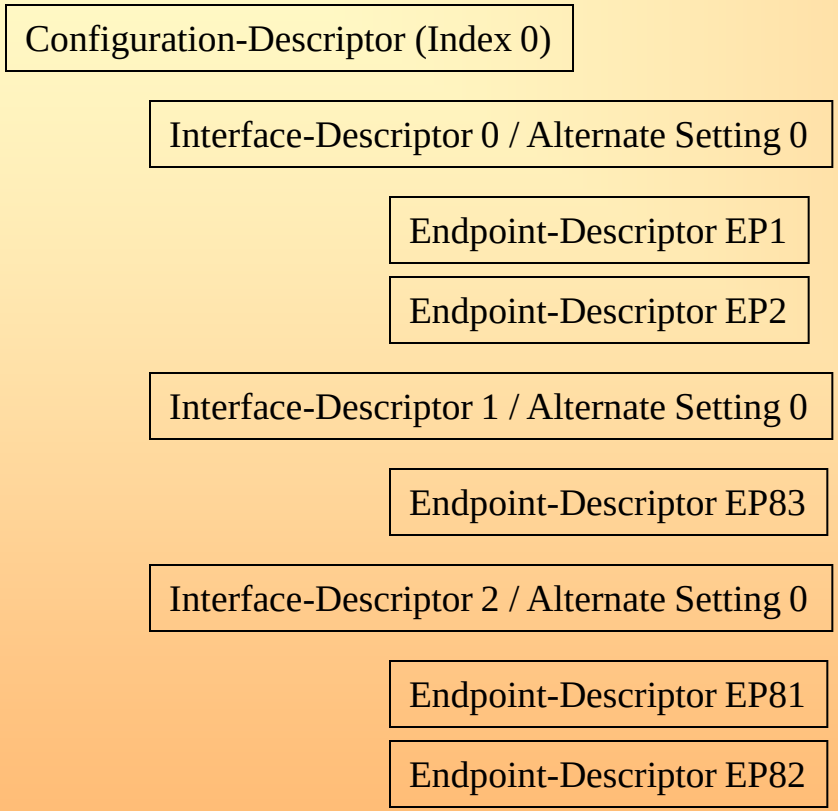
Offset	Field	Length	Type	Description	Example
0	Descr.-Length	1	Number	Size of this descriptor in byte	07h
1	Descriptor-Type	1	Constant	Endpoint-Descriptor-Type = 05 h	05h
2	Endp.-Address	1	Number	Endpoint-Address: OUT- Endpoint number 2	02h
3	Attributes	1	Bitmap	Transfer type - Bulk	02h
4	Max.Packet Size	2	Number	Maximum Packet Size of this endpoint = 64 byte	40,00h
6	Interval	1	Number	Polling Interval for Bulk-EP‘s not relevant	00h

String-Descriptor for Index 01h

Offset	Field	Length	Type	Description	Example
0	Descr Length	1	Number	Size of this descriptors in byte = 20	16h
1	Descriptor-Type	1	Constant	String-Descriptor-Type = 03 h	03h
2	1. character	2	Unicode	E	45,00h
4	2. character	2	Unicode	M	4D,00h
6	3. character	2	Unicode	S	53,00h
8	3. character	2	Unicode	Y	59,00h
10	4. character	2	Unicode	S	53,00h
12	5. character	2	Unicode	(space)	20,00h
14	6. character	2	Unicode	G	47,00h
16	7. character	2	Unicode	m	6D,00h
18	8. character	2	Unicode	b	62,00h
20	9. character	2	Unicode	H	48,00h

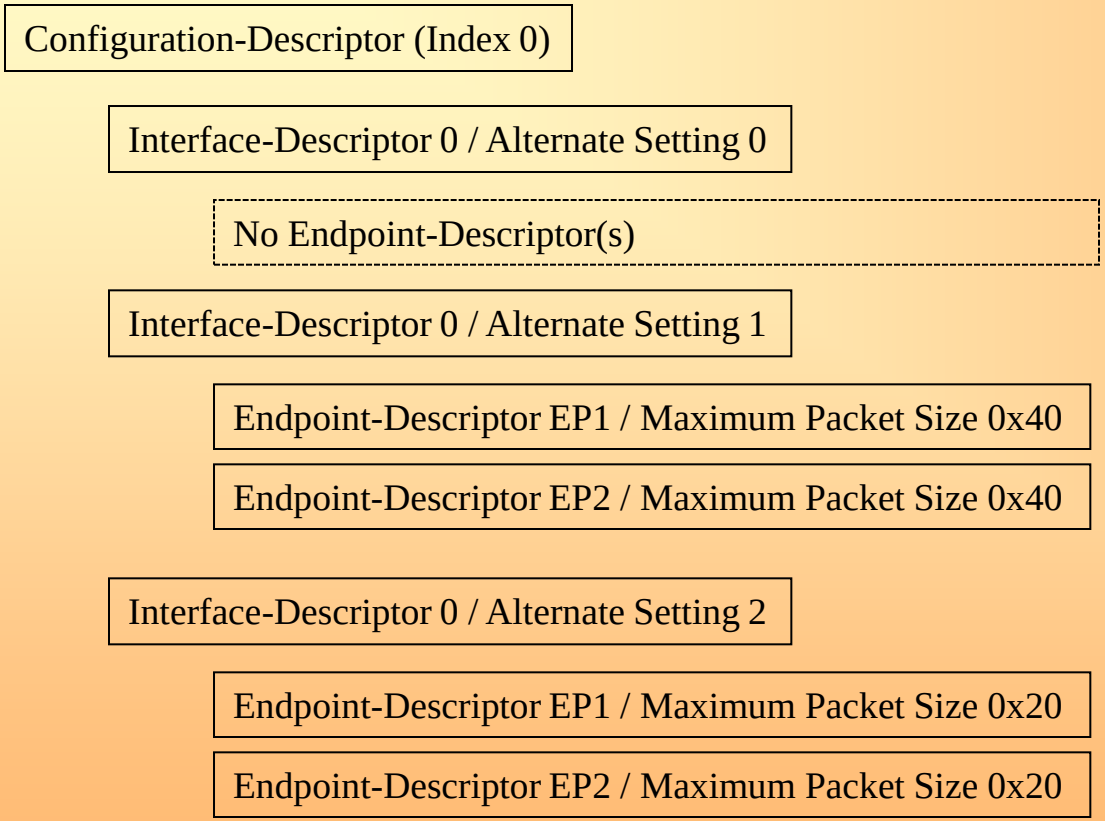
- UNICODE = 00h + ASCI-Code
- UNICODE strings are NOT NULL-terminated !

Descriptor-Order (I)



- ⇨ Devices with several configurations and interfaces
- ⇨ Only one configuration could be active!
- ⇨ Interfaces in one configuration could be active in parallel
- ⇨ Different interfaces in one configuration must use separate endpoint addresses

Descriptor-Order (II)



- ⇨ Isochronous devices with one configuration and one interface with several Alternate-Settings
- ⇨ Interface descriptors are identical except of the value for Alternate-Setting
- ⇨ Endpoint descriptors of one specific endpoint are identical except of the value for „Max. Packet Size“ (= Maximum Packet Size)

Interface Association Descriptor (IAD)

- ECN defines IADs, refer www.usb.org/developers
- Reason: ambiguity in USB Specification with respect to whether multi-function devices should be allowed to use more than one interface per logical function
- Typical use-case: CDC Devices
  - one logical function uses two interfaces
    - communication interface for control
    - data interface for payload data
- Backward compatible, because it will be ignored by the descriptor parser of older systems
- Must be provided as part of Configuration Descriptor
- Must be located before each set of Interface Descriptors

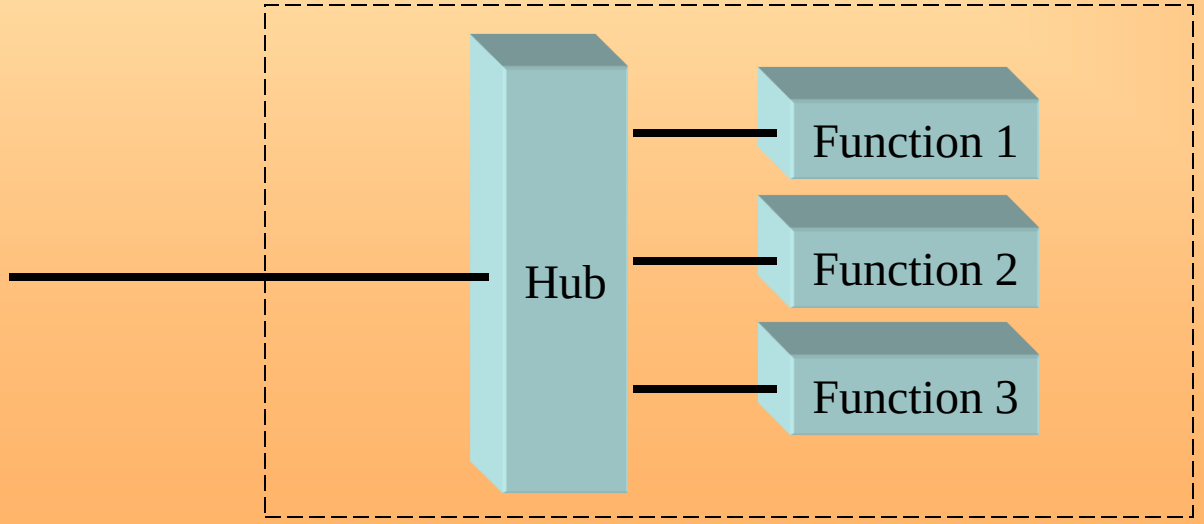
Interface Association Descriptor

Offset	Field	Length	Type	Description	Example
0	Descr.-Length	1	Number	Size of this descriptors in byte	09h
1	Descriptor-Type	1	Constant	Interface-Descriptor-Type = 11 h	11h
2	FirstIfc-Num.	1	Number	Number of first interface associated with this function	00h
3	Interface Count	1	Number	# of contiguous interfaces associated with this function	2h
4	Function Class	1	Class	Interface-Class-Code (example = vendor-specific)	02h
5	Function Subclass	1	Subclass	Interface-Subclass-Code	00h
6	Functioin Protocol	1	Protocol	Interface-Protocol-Code	00h
7	Function	1	Index	Index of String-Descriptor ‘Function’	08h

Compound vs. Composite Device

Compound-Devices

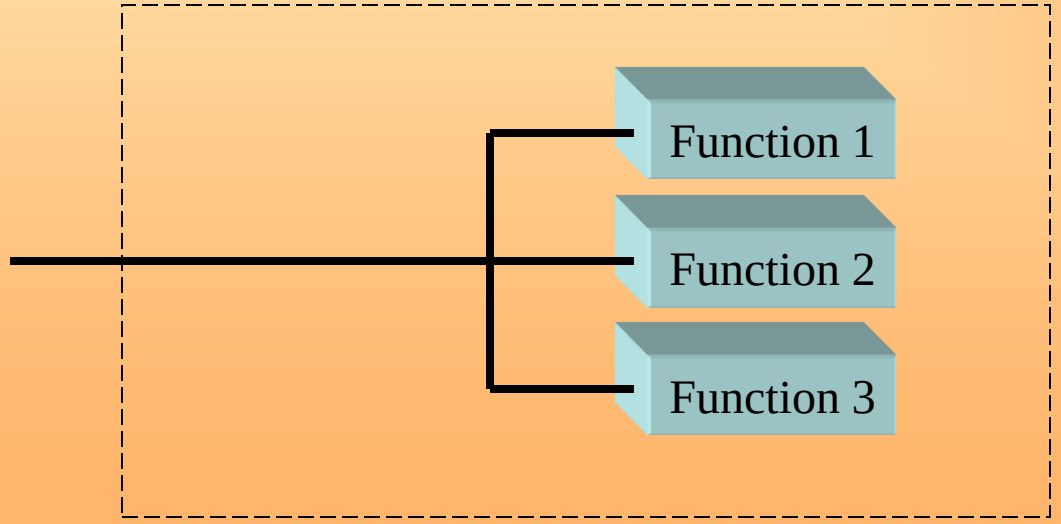
- Several devices are fix connected with a (virtual) Hub
- System is one physical device,
  but several logical devices (each with its own address for Hub / Functions)



- Example: Cherry-keyboard with integrated Hub, using INTEL 80930H-USB-Chip

Composite-Devices

- Logical devices are existing at Interface-Level
- Device has only one USB-address
- Splitting in several logical devices at PC-driver side



⇨Example:
Keyboard with integrated speaker & printer

Standard Device Requests

Definition of Device Requests

- Used by control-transfers to EP0
- Always 8 byte data-packet

- Byte 0 = Request-Type
  - D 7: direction of additional data-stage
    0 = host to device
    1 = device to host
  - D 6..5: type
    00 = standard-Request
    01 = class-specific request
    10 = vendor-specific request
    11 = reserved
  - D 4..0: recipient
    00000 = device
    00001 = interface
    00010 = endpoint
    00011 = other
    4 .. 31 = reserved

Def. of Standard Device Requests

- Byte 1 = Request-Code
  - 00h = Get Status
  - 01h = Clear Feature
  - 03h = Set Feature
  - 05h = Set Address
  - 06h = Get Descriptor
  - 07h = Set Descriptor (opt.)
  - 08h = Get Configuration
  - 09h = Set Configuration
  - 0Ah = Get (Alternate Setting for) Interface
  - 0Bh = Set (Alternate Setting for) Interface

- Byte 2,3 = Value, depending on request, see spec. / chap. 9
- Byte 4,5 = Index, depending on request, see spec. / chap. 9
- Byte 6,7 = max. length of a data-packet accepted by the host

  (only used if data requested from device)

Example (I)

Idle	SYNC	SETUP	ADDR	EP	CRC5	EOP
4	00000001	0x2D	0x00	0x00	0x08	2

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	00 05 02 00 00 00 00 00	0xD768	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Request to EP0 is always control-transfer

Value: New address

Request: **Set_Address**

Standard-Request to Device

Idle	SYNC	SETUP	ADDR	EP	CRC5	EOP
4	00000001	0x2D	0x02	0x00	0x15	2

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	00 09 01 00 00 00 00 00	0xD768	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Value: Configuration-Value

Request: **Set_Configuration**

Standard-Request to Device

Example (II)

Idle	SYNC	SETUP	ADDR	EP	CRC5	EOP
4	00000001	0x2D	0x00	0x00	0x08	2

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	80 06 00 01 00 00 12 00	0xBB29	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Standard-Request to Device, data requested

Request: **Get-Descriptor**

Descriptor-Type:
Device-Descriptor

Descriptor-Index

Max. length of accepted data: 0012h = 18

Idle	SYNC	IN	ADDR	EP	CRC5	EOP
30	00000001	0x69	0x00	0x00	0x08	2

Idle	SYNC	DATA1	DATA	CRC16	EOP
3	00000001	0x4B	12 01 00 01 09 01 00 08	0x82DD	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Idle	SYNC	IN	ADDR	EP	CRC5	EOP
30	00000001	0x69	0x00	0x00	0x08	2

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	6A 04 03 00 05 03 00 00	0x3139	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Control-Transfer-Protocol

Setup-Transactions (No Data)

⇨ 2-Stage-Control-Transfer (SET_CONFIGURATION)

Idle	SYNC	SETUP	ADDR	EP	CRC5	EOP
4	00000001	0x2D	0x02	0x00	0x08	2

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	00 09 01 00 00 00 00 00	0xD768	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Setup-Stage

Idle	SYNC	IN	ADDR	EP	CRC5	EOP
30	00000001	0x69	0x02	0x00	0x08	2

Idle	SYNC	DATA1	DATA	CRC16	EOP
3	00000001	0x4B		0x0000	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Status-Stage

- Usage of advanced handshake for control transfers
- Zero-Data-Packet acknowledges successful request

Setup-Transactions (Error)

⇨ 2-Stage-Control-Transfer (SET_CONFIGURATION) – NOT SUPPORTED

Idle	SYNC	SETUP	ADDR	EP	CRC5	EOP
4	00000001	0x2D	0x02	0x00	0x08	2

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	00 09 04 00 00 00 00 00	0xD768	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Setup-Stage

Idle	SYNC	IN	ADDR	EP	CRC5	EOP
30	00000001	0x69	0x02	0x00	0x08	2

Idle	SYNC	NAK	EOP
5	00000001	0x5A	2

Idle	SYNC	IN	ADDR	EP	CRC5	EOP
30	00000001	0x69	0x02	0x00	0x08	2

Idle	SYNC	STALL	EOP
5	00000001	0x1E	2

Status-Stage

Setup-Transactions (IN Data)

⇨ 3-Stage-Control-Transfer (GET_DEVICE_DESCRIPTOR)

Idle	SYNC	SETUP	ADDR	EP	CRC5	EOP
4	00000001	0x2D	0x00	0x00	0x08	2

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	80 06 00 01 00 00 40 00	0xBB29	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Setup-Stage

Idle	SYNC	IN	ADDR	EP	CRC5	EOP
30	00000001	0x69	0x00	0x00	0x08	2

Idle	SYNC	DATA1	DATA	CRC16	EOP
3	00000001	0x4B	12 01 00 01 09 01 00 08	0x82DD	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

n x Data-Stage

Idle	SYNC	OUT	ADDR	EP	CRC5	EOP
30	00000001	0xE1	0x00	0x00	0x08	2

Idle	SYNC	DATA1	DATA	CRC16	EOP
3	00000001	0x4B		0x0000	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Status-Stage

Setup-Transactions (OUT Data)

⇨ 3-Stage-Control-Transfer (SET_LINE_CODING)

Idle	SYNC	SETUP	ADDR	EP	CRC5	EOP
4	00000001	0x2D	0x00	0x00	0x08	2

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	21 20 00 00 00 00 07 00	0xFA4B	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Setup-Stage

Idle	SYNC	OUT	ADDR	EP	CRC5	EOP
30	00000001	0x69	0x00	0x00	0x08	2

Idle	SYNC	DATA1	DATA	CRC16	EOP
3	00000001	0x4B	60 09 00 00 01 00 08	0x2B66	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

1 x Data-Stage

Idle	SYNC	IN	ADDR	EP	CRC5	EOP
30	00000001	0xE1	0x00	0x00	0x08	2

Idle	SYNC	DATA1	DATA	CRC16	EOP
3	00000001	0x4B		0x0000	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Status-Stage

Enumeration Process

Why an Enumeration?

- Plug- & Play-functionality requires initialization & configuration of connected devices
- Descriptors explain functionality / attributes of an USB device
- Host asks for the descriptors during enumeration phase
- Information leads to dedicated loading of host-drivers
  - bandwidth management
  - power management
- Address assignment of an unique address (1..127) to every device
  - power-on-Address 0 reserved as default address
- Choosing and activating of a configuration / (alternate setting for) interface

Enumeration-Sequence

⇨ Example: Enumeration Sequence using Windows2000/XP

- USB-Reset
- Get Device Descriptor on default-address 0
- USB-Reset
- Set Device address
- Get Device Descriptor
- Get Configuration Descriptor (only first 9 bytes)
  - Configuration Descriptor
- Get String Descriptor (Language ID)
- Get String Descriptor (Serial number)

Enumeration-Sequence (2)

- ..continue
  - Get Configuration Descriptor (all)
    - Configuration Descriptor
    - Interface Descriptors
    - Endpoint Descriptors
    - Class specific Descriptors
  - Get String Descriptor (Language ID)
  - Get String Descriptor (Product String)
  - Set Configuration
  - Set Interface
  - Get class-specific Descriptor(s)
  - Enable class-specific features

Enumeration-Sequence(3)

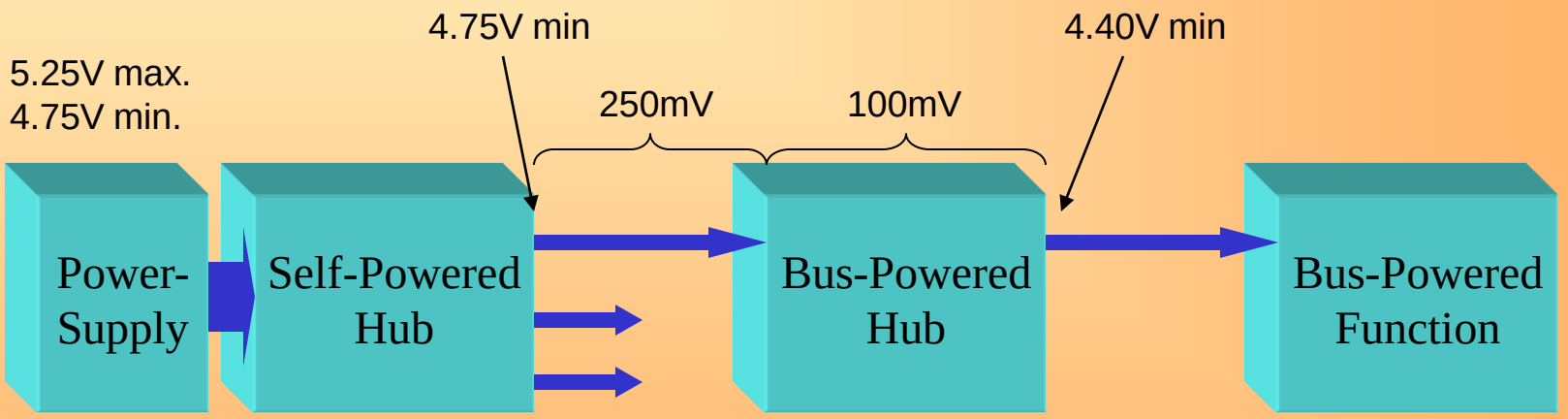
- If the hub is connected to a full-speed capable port AND
- An OS is used that supports high-speed AND
- The device has returned in the device descriptor that it is compliant to USB Specification 2.00 :

- System checks also the DeviceQualifierDescriptor
  - If STALL-returned -> OK, no message reported from OS
  - If Descriptor reported -> OS checks descriptors for „other“ speed and reports message to user that the device can work better if using another USB port

Power-Distribution and -Management

Voltage Drops at Cable & Hubs

- Hubs support power-distribution and -management
  - Power-switching for downstream ports
  - Over current protection



- Cascade of 2 bus-powered hubs not possible !
  (due to high voltage drop at cable and hubs)

Self powered vs. bus powered Hubs

	Self-powered Hub	Bus-powered Hub
Power-Source	Own power supply	upstream-port
Upstream-Power-Load	0 ... max. 100 mA	max. 500 mA
Downstream-Power-Resources	500 mA per port	max. 100 mA on max. 4 ports
Overcurrent-Protection	yes	optional

⇨ Power-Mode

- Direct Power : all downstream ports are always on
- Ganged Switching : all downstream ports controlled with one switch
- Individual Switching : every downstream port controlled separately

Self-powered / Bus-powered Functions

- Self-powered function - own power supply
  - upstream load max. 100 mA (Hybrid device)

- Bus-powered function - power supply via USB
  - Low-power function
    - max. 100mA power consumption
  - High-power function
    - max. 100mA power consumption if not configured
    - max. 500mA power consumption if configured

Suspend Power Consumption

- max. 500 µA if low-power device
- max. 2,5 mA if high-power device (500 µA / 100mA)

- Device must enter suspend state if no USB-traffic > 3 ms
  - Full-Speed : SOF every ms
  - Low-Speed : EOP every ms

- Leaving Suspend
  - USB-reset
  - Connect / disconnect of the device
  - Host drives resume signaling
  - Function drives resume signaling = Remote-Wakeup
    - e.g. press a key on your keyboard / move your mouse

ECN “USB 2.0 Connect Timing”

- Strict power management rules of the USB Specification may cause several issues:
  1. Devices with “Dead Battery” may not obey the rule to draw 100mA for only 100 ms before connecting (activating the pull-up)
  2. Devices with “Dead Battery” may violate the “V_{BUS} session valid to connect time” (100ms)
  3. Devices may not be able to drop down to suspend current within 10 ms of no bus activity during attach debounce time (T_{ATTDB}=100ms)

ECN “USB 2.0 Connect Timing”

⇨ The ECN relaxes these rules

1. Devices with “Dead Battery” can draw 100mA for the time required to power up the device and to bring it to a state where it can connect
2. The maximum time from V_{BUS} crossing session valid level to when the device is required to connect depends of the device’s power mode:

Bus power devices and devices that are already powered up	Portable devices with dead or weak battery
$T_{SVLD\_CON\_PWD}$ =1 s	$T_{SVLD\_CON\_WKB}$ = 45min

3. Devices are **not** required to drop down to suspend current during 1 second following the connect event (T_{TCON_ISUSP} = 1s)

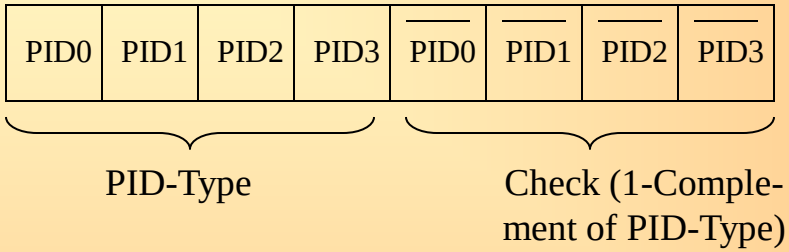
Error Handling

Error Handling

- All transfer modes (except isochronous) are protected
- Packet errors
  - PID-check
  - CRC-Errors
  - Bit-stuff-errors
- Time-out
- Data-toggle-error
- Babble
- LOA (Loss-of-Activity)

Packet-Errors

- PID-Check: PIDs protected by own structure

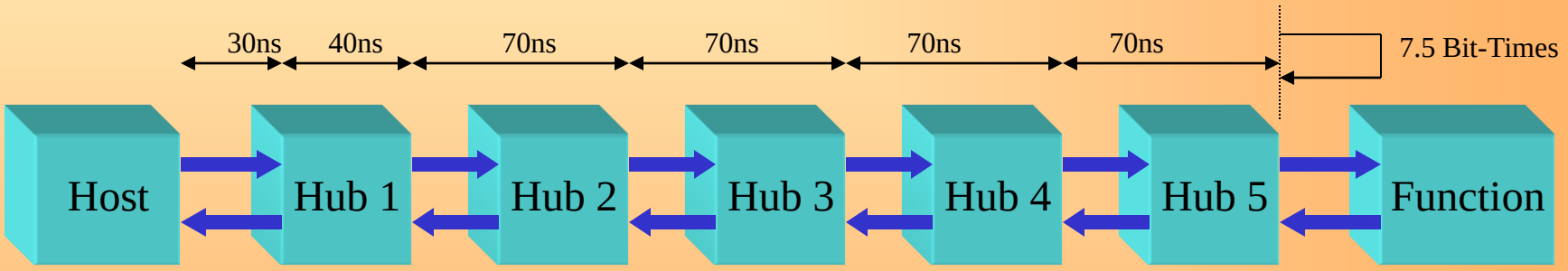


- CRC-Check:
  - SOF 11-bit frame number 5-bit-CRC
  - IN/OUT/SETUP 7-bit address und 4-bit endpoint-number 5-bit-CRC
  - DATA 0 / 1 max. 1023 byte data 16-bit-CRC

- Bit-Stuff-Errors
  - Receiver expects stuff-bit: if not found -> bit-stuff & CRC-error detected

Timeout

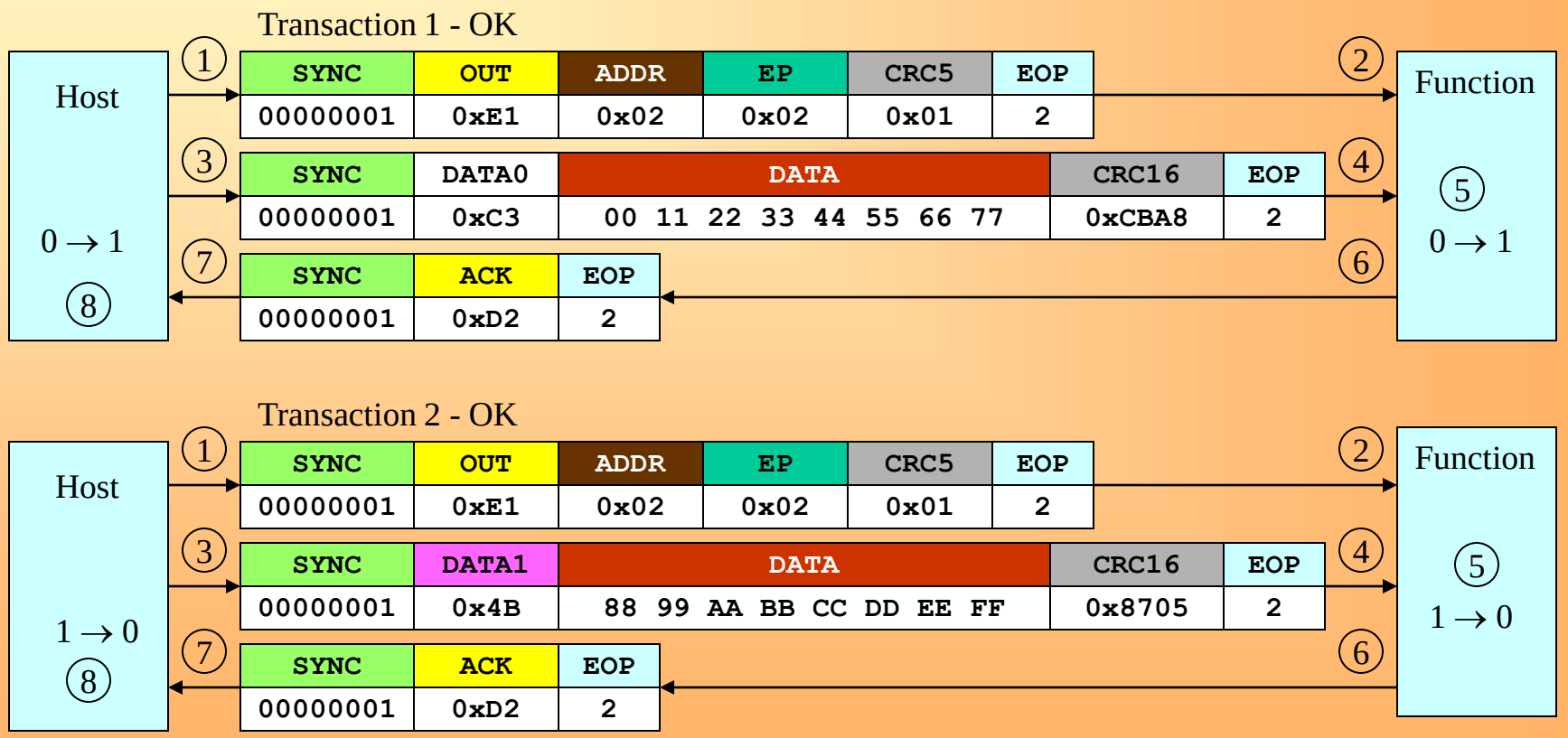
- Timeout defined by total-turnaround-time for maximum path
  - Cable delay: max. 30 ns
  - Hub delay: max. 40 ns
  - 7.5 bit-time max. response-time for function (1 bit time = 1 / 12 MHz = 83 ns)



  - Start of Time-Out-Counter: EOP
    Stop of Time-Out-Counter: SOP of following packet
  - Host-Time-Out reached by 18 FS-bit-times
  - Function-Time-Out reached by 16 ≤ FS-bit-times ≤ 18

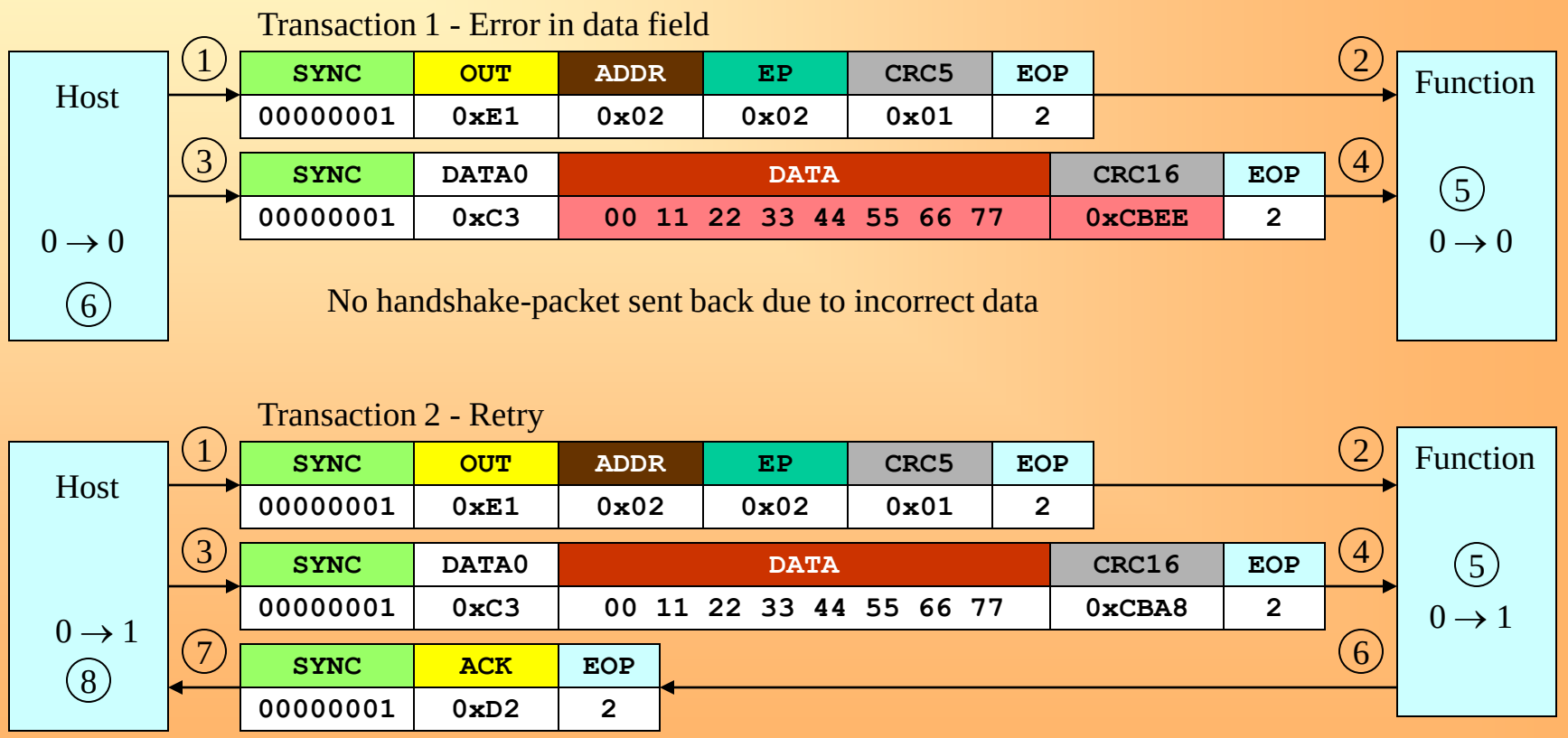
Data-Toggle-Mechanism (I)

OUT-Transaction without any errors and correct toggle-sequence



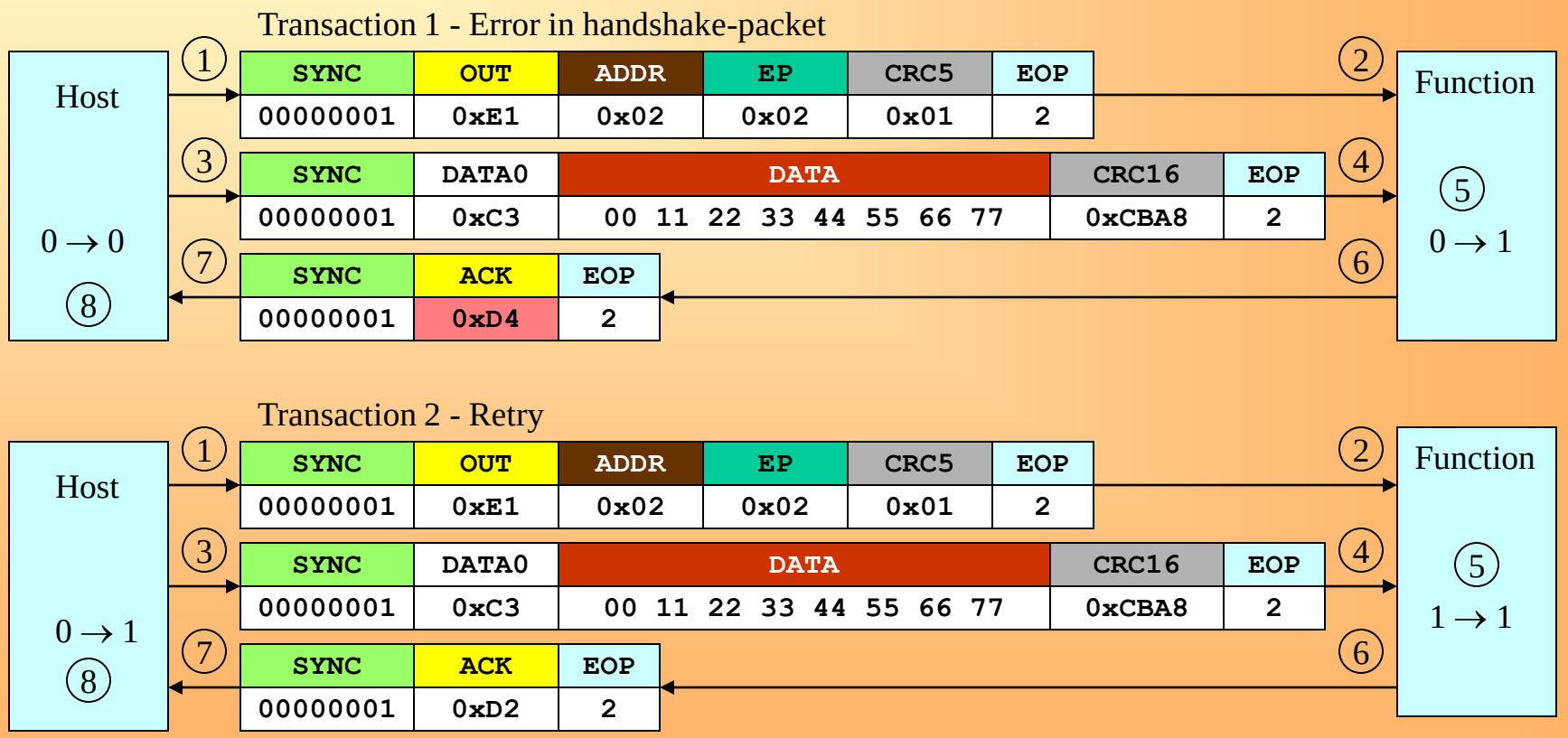
Data-Toggle-Mechanism (II)

Retry-Mechanism for OUT-Transaction with destroyed data-field



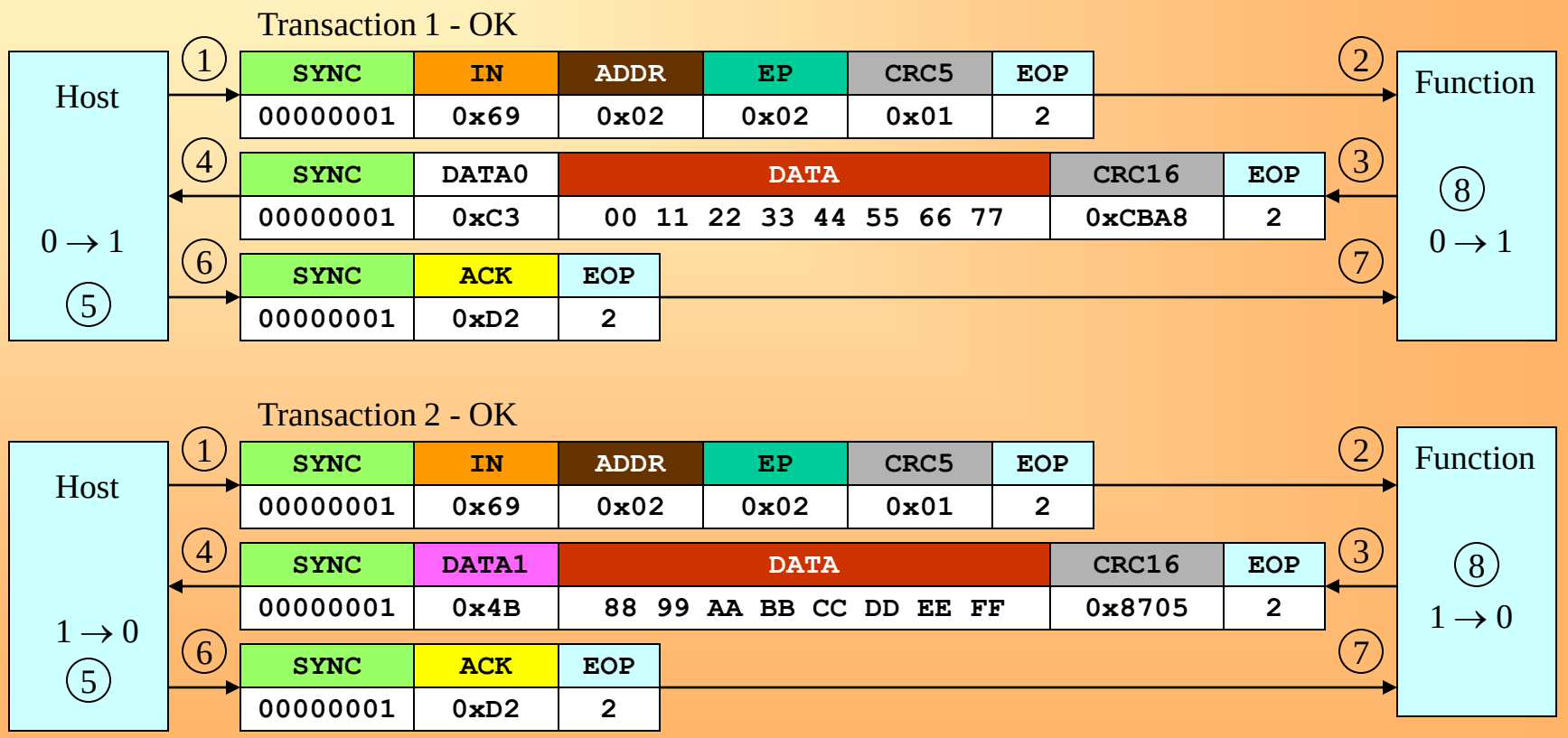
Data-Toggle-Mechanism (III)

Retry-Mechanism for OUT-Transaction with destroyed handshake



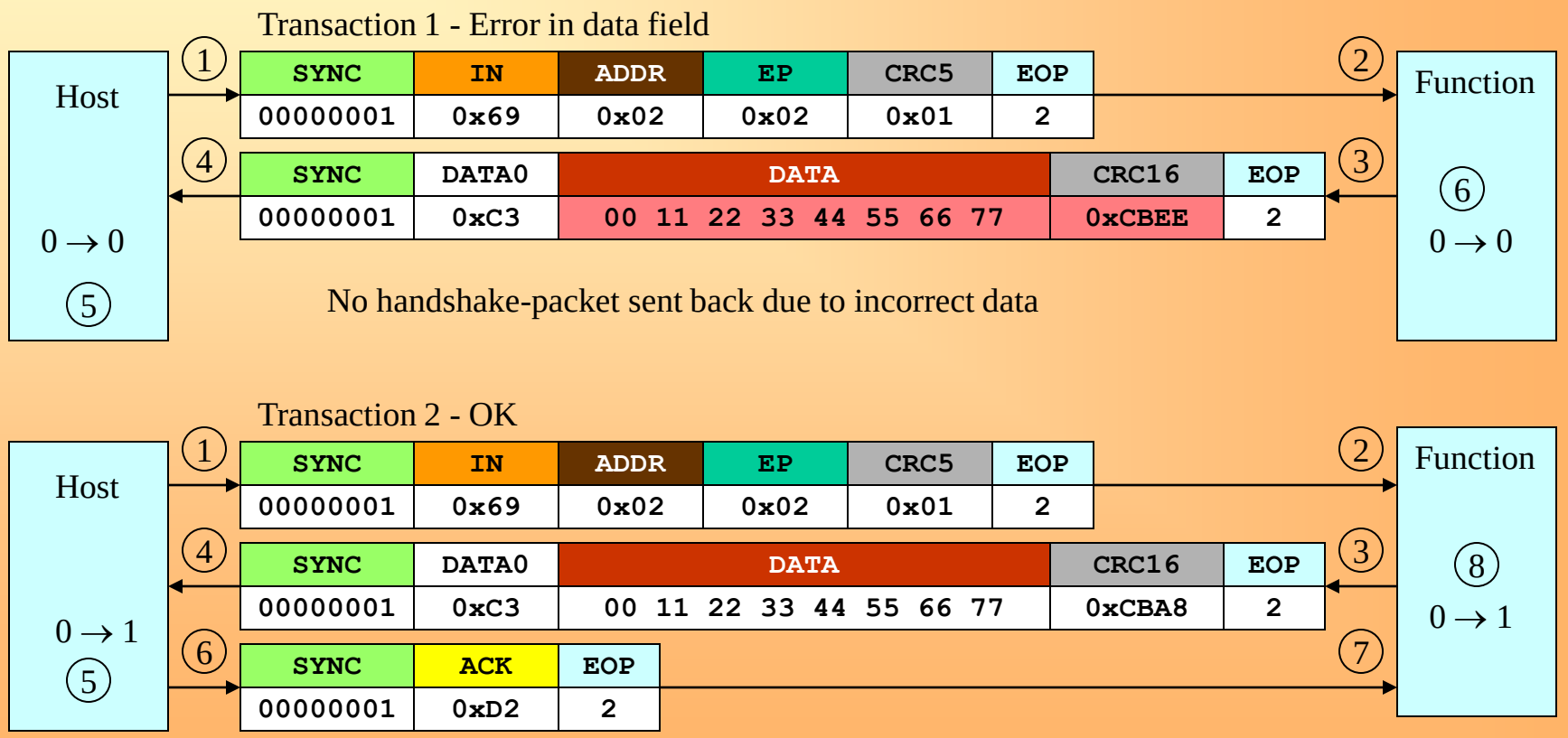
Data-Toggle-Mechanism (IV)

IN-Transaction without any errors and correct toggle-sequence



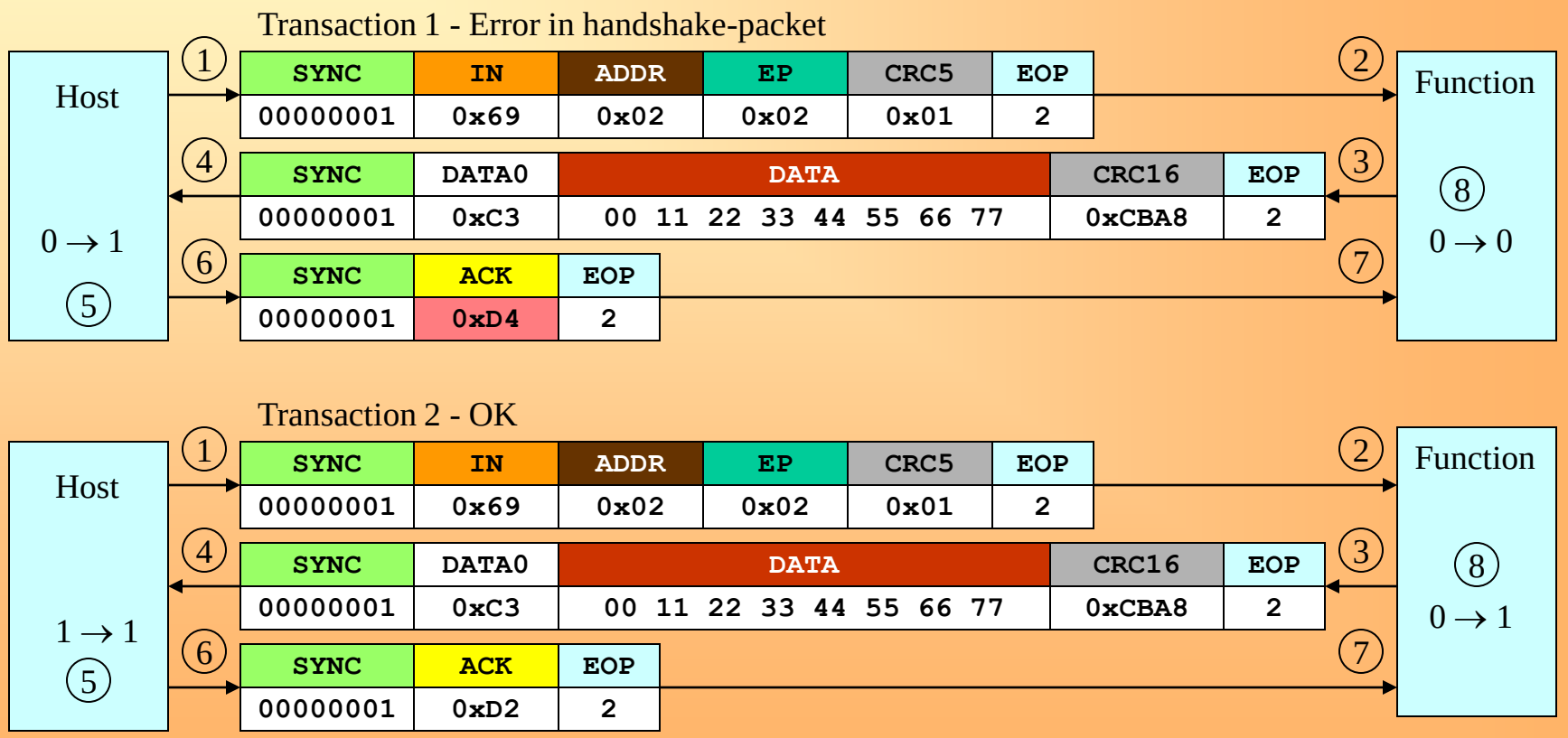
Data-Toggle-Mechanism (V)

Retry-Mechanism for IN-Transaction with destroyed data-field



Data-Toggle-Mechanism (VI)

Retry-Mechanism for IN-Transaction with destroyed handshake



Data-Toggle Special Consideration

- Data-toggle mechanism does not work if a packet is last packet of a message
- Example: SetAddress Request with corrupted handshake during status stage

Idle	SYNC	SETUP	ADDR	EP	CRC5	EOP
4	00000001	0x2D	0x02	0x00	0x08	2

Idle	SYNC	DATA0	DATA	CRC16	EOP
3	00000001	0xC3	00 05 03 00 00 00 00 00	0x57E3	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Setup-Stage

Idle	SYNC	IN	ADDR	EP	CRC5	EOP
30	00000001	0x69	0x02	0x00	0x08	2

Idle	SYNC	DATA1	DATA	CRC16	EOP
3	00000001	0x4B		0x0000	2

Idle	SYNC	ACK	EOP
5	00000001	0xD2	2

Status-Stage

Data-Toggle Special Consideration (2)

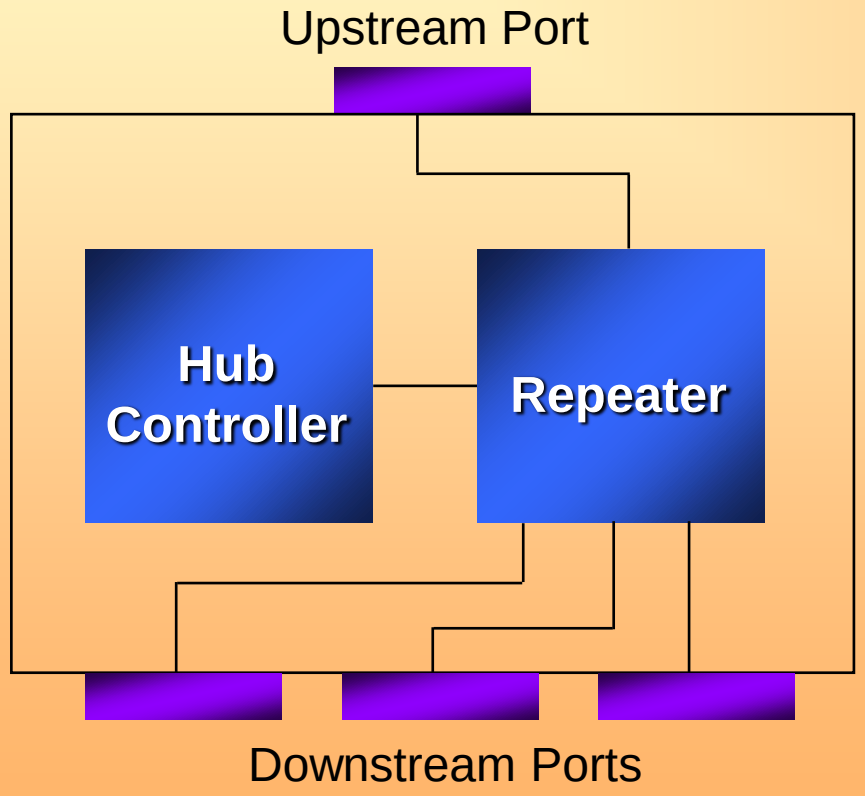
- If ACK is corrupted, Host and Device are in a different state
  - Host has detected status stage and sent ACK -> Request completed and will continue sending further requests at new address
  - Device has sent status stage, but not detected the ACK -> Request not completed (waits for retry) and still working at old address
- Same problem can occur if other transfer types are used as message pipes during a response phase

LOA / Babble

- Loss-of-Activity (LOA)
  - Start-of-Packet is OK, but lines keep in static state without reaching EOP
- Babble
  - Start-of-Packet is OK, lines keep toggling - but no EOP reached
  - Possible collision with Start-of-Frame-Tokens !
- Risk: deadlock or loss of synchronization due to destroyed packets
- Hubs recognize and prevent these errors
  - Hub-internal frame timer disables ports which are source of these errors
  - Message to host for advanced error detection and correction

Hubs

Hub-Architecture



⇨Hub Repeater

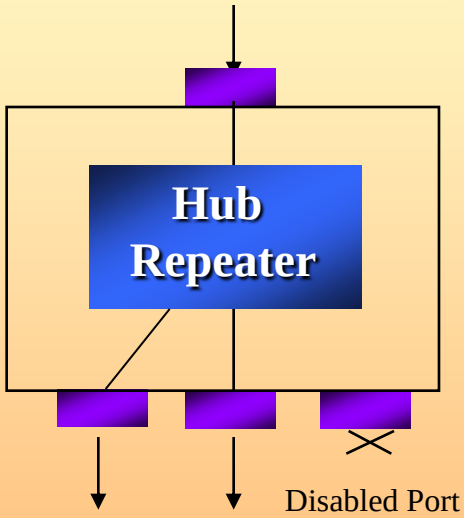
- Establishing & destroying connectivity between upstream & downstream ports
- Connect / disconnect-recognition
- Suspend / resume support
- Reset support
- Error detection & recovery (LOA / babble / power-overload)

⇨Hub Controller

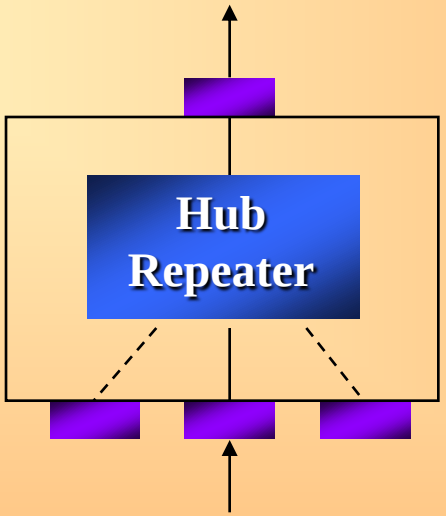
- Always two endpoints:
  - Control-Endpoint (EP0)
  - Interrupt-Endpoint (EP1)
- Always full speed device
- Managing the downstream-ports

Packet-Signaling Connectivity

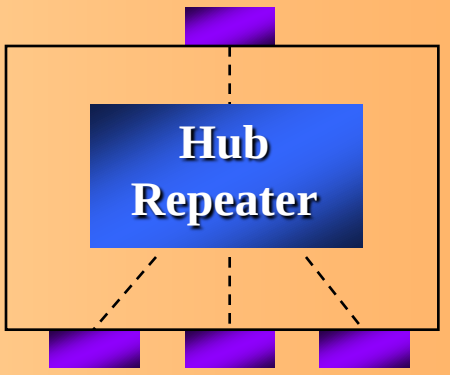
Downstream Connectivity



Upstream Connectivity



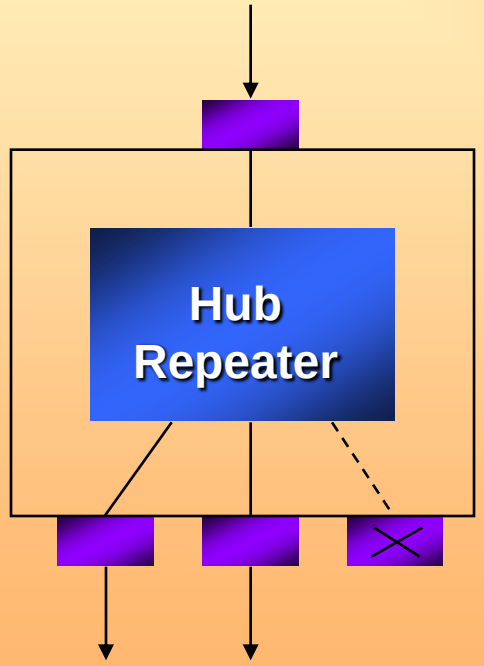
Idle (no connectivity)



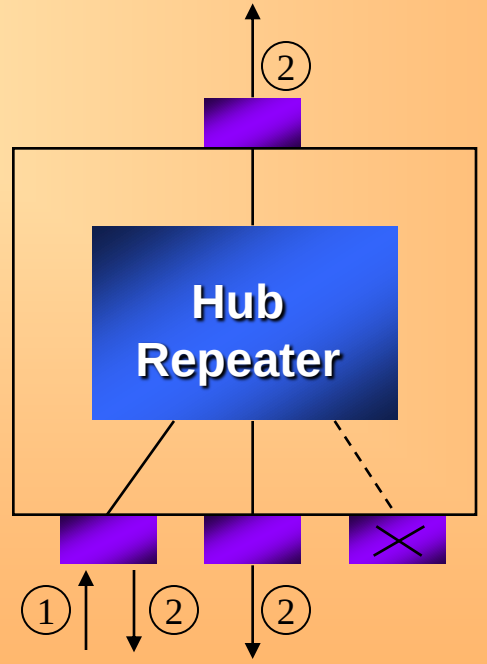
- ⇨ Connectivity between upstream & downstream-ports default disabled
- ⇨ Establishing of a (logical) connection by detection of Start-of-Packet (SOP)
- ⇨ Destroying the connection by detection of an EOP or in error case (LOA/Babble)

Resume-Connectivity

Downstream Connectivity



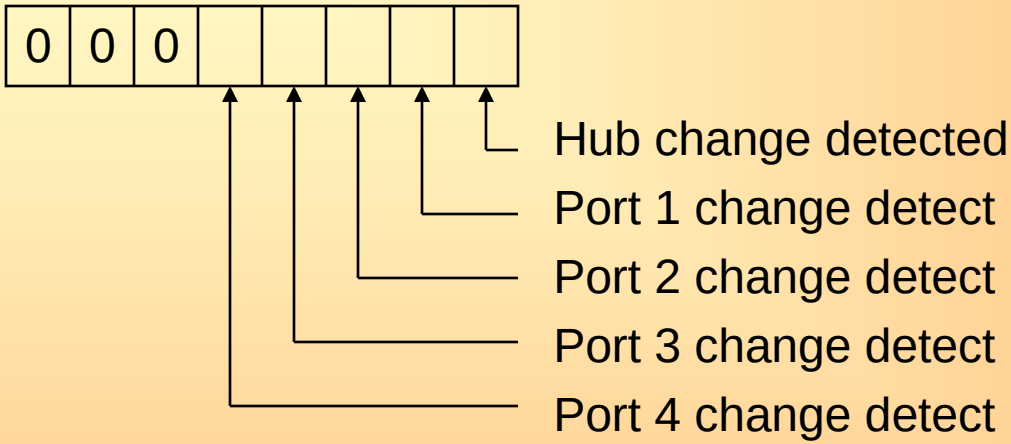
Upstream Connectivity



Hub-Class-Descriptor

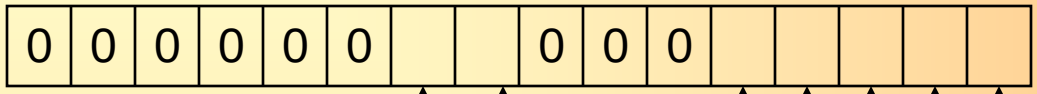
Example	Offset	Field	Length	Type	Description
09h	0	Descr.-Length	1	Number	Size of this descriptor in byte
29h	1	Descriptor-Type	1	Constant	Hub-Class-Descriptor-Type = 29 h
04h	2	# Ports	1	Number	Number of supported downstream-ports
90	3	Characteristics	2	Bitmap	Power-Mode / Overload-Protection / Compound-Device
10	5	PwrOn2Good	1	Number	Time between switching port and power good
10	6	HubContrCurrent	1	Number	Current consumption of the hub in 2mA units
02h	7	Dev. Removable	y	Bitmap	Add. Device fix connected to port: x = Bit-Number Dx: 0 = Device removable, 1 = Device fixed
00h	7+x	PortPwrCtrlMask	y	Bitmap	Port power controlled by gang mode Dx: 0 = gang mode, 1 = no gang mode

Hub-Port-Status-Change-Report



- Polling via interrupt-endpoint EP1 every x ms
- Information includes only location of change
  - 1 bit for changing the hub-environment (power-Supply, global over current)
  - 1 bit per port (corresponding to port number)
  - Extracting the particular status- & change-information via EP0-control transfers

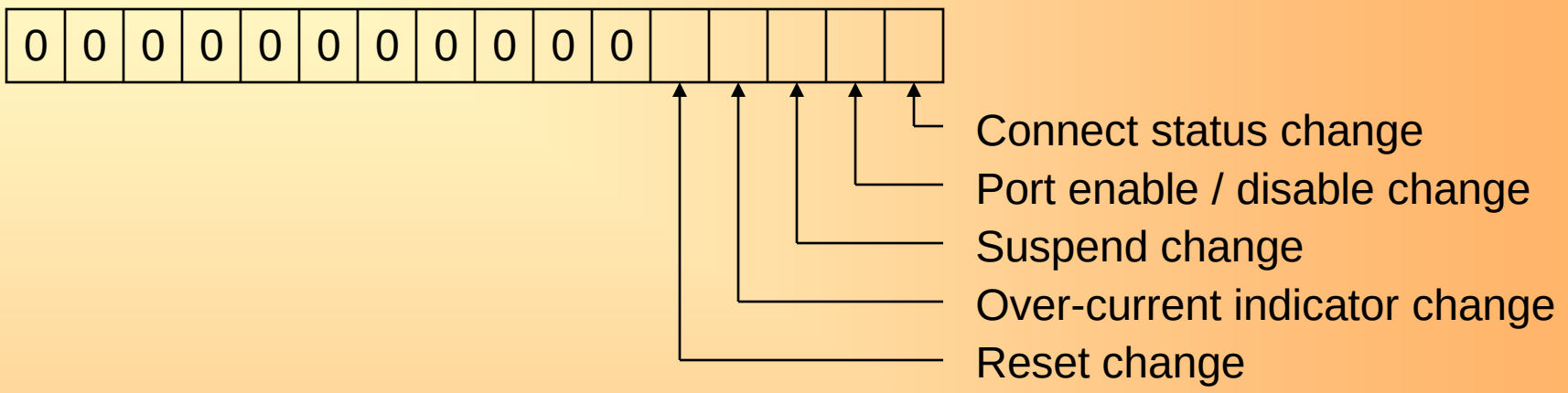
Port-Status-Bitmap



- Current connect status
- Port enabled / disabled
- Suspend
- Over-current indicator
- Reset
- Port power
- Low speed device attached

- ⇨ Messaging over EP 0 (GET_PORT_STATUS)
- ⇨ Port-enable / port-suspend / port-reset / port-power controlled with SET / RESET - Port - Feature - Requests by host
- ⇨ Current connect status / over-current-indicator / low-speed controlled by hub-hardware

Port-Change-Bitmap



- Transmission of the change-bitmap together with status-bitmap when using GET_PORT_STATUS - request
- Set bits controlled by hub hardware & firmware
- Reset bits controlled by host (via CLEAR-port-feature-requests)

Example (I)

SYNC	IN	ADDR	EP	CRC5	EOP
00000001	0x69	0x02	0x01	0x18	2

SYNC	DATA0	DATA	CRC16	EOP
00000001	0xC3	10	0x82CE	2

SYNC	ACK	EOP
00000001	0xD2	2

Hub-Port-Status-Change-Report

Polling of Interrupt-EP 1

Change at port 4 occurred

SYNC	SETUP	ADDR	EP	CRC5	EOP
00000001	0x2D	0x02	0x00	0x15	2

SYNC	DATA0	DATA	CRC16	EOP
00000001	0xC3	A3 00 00 00 04 00 04 00	0x6F96	2

SYNC	ACK	EOP
00000001	0xD2	2

GET_PORT_STATUS Port-Nr. Report-Length

SETUP-Stage: GET_PORT_STATUS

Host requests via EP0

Port-status of port 4

SYNC	IN	ADDR	EP	CRC5	EOP
00000001	0x69	0x02	0x00	0x15	2

SYNC	DATA1	DATA	CRC16	EOP
00000001	0x4B	03 01 10 00	0xC5F9	2

SYNC	ACK	EOP
00000001	0xD2	2

Port-Status-Bitmap Port-Change-Bitmap

DATA-Stage:

Hub reports port-status & change-bitmap

->Reset-change occurred

SYNC	OUT	ADDR	EP	CRC5	EOP
00000001	0xE1	0x02	0x00	0x15	2

SYNC	DATA1	DATA	CRC16	EOP
00000001	0x4B		0x0000	2

SYNC	ACK	EOP
00000001	0xD2	2

STATUS-State:

Host acknowledges with OUT token

(3-Stage-Control-Transfer)

Example (II)

SYNC	SETUP	ADDR	EP	CRC5	EOP
00000001	0x2D	0x02	0x00	0x15	2

SYNC	DATA0	DATA	CRC16	EOP
00000001	0xC3	23 01 14 00 04 00 00 00	0xF7B8	2

SYNC	ACK	EOP
00000001	0xD2	2

CLR_FEATURE C_PORT_RESET Port-Nr.

SETUP-Stage:

Host resets change-port-reset-bit

of port 4

SYNC	IN	ADDR	EP	CRC5	EOP
00000001	0x69	0x02	0x00	0x15	2

SYNC	DATA1	DATA	CRC16	EOP
00000001	0x4B		0x0000	2

SYNC	ACK	EOP
00000001	0xD2	2

STATUS-Stage:

Hub acknowledges with OUT-Token

(2-Stage-Control-Transfer)

SYNC	IN	ADDR	EP	CRC5	EOP
00000001	0x69	0x02	0x01	0x18	2

SYNC	NAK	EOP
00000001	0x5A	2

Polling of EP 1 again...

No new changes occurred

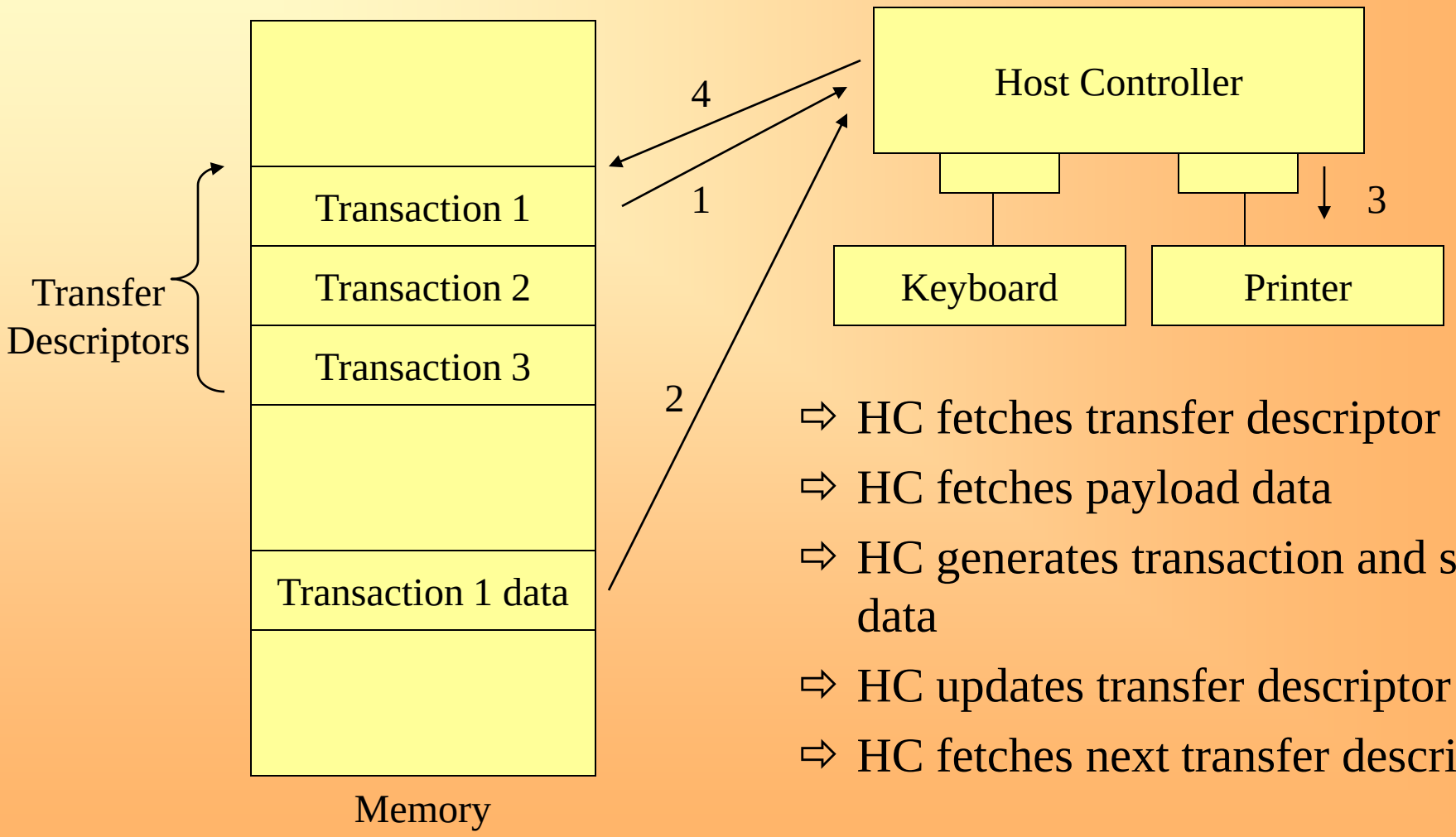
Host-Hardware

UHC / OHC

Tasks of the Host-Controllers

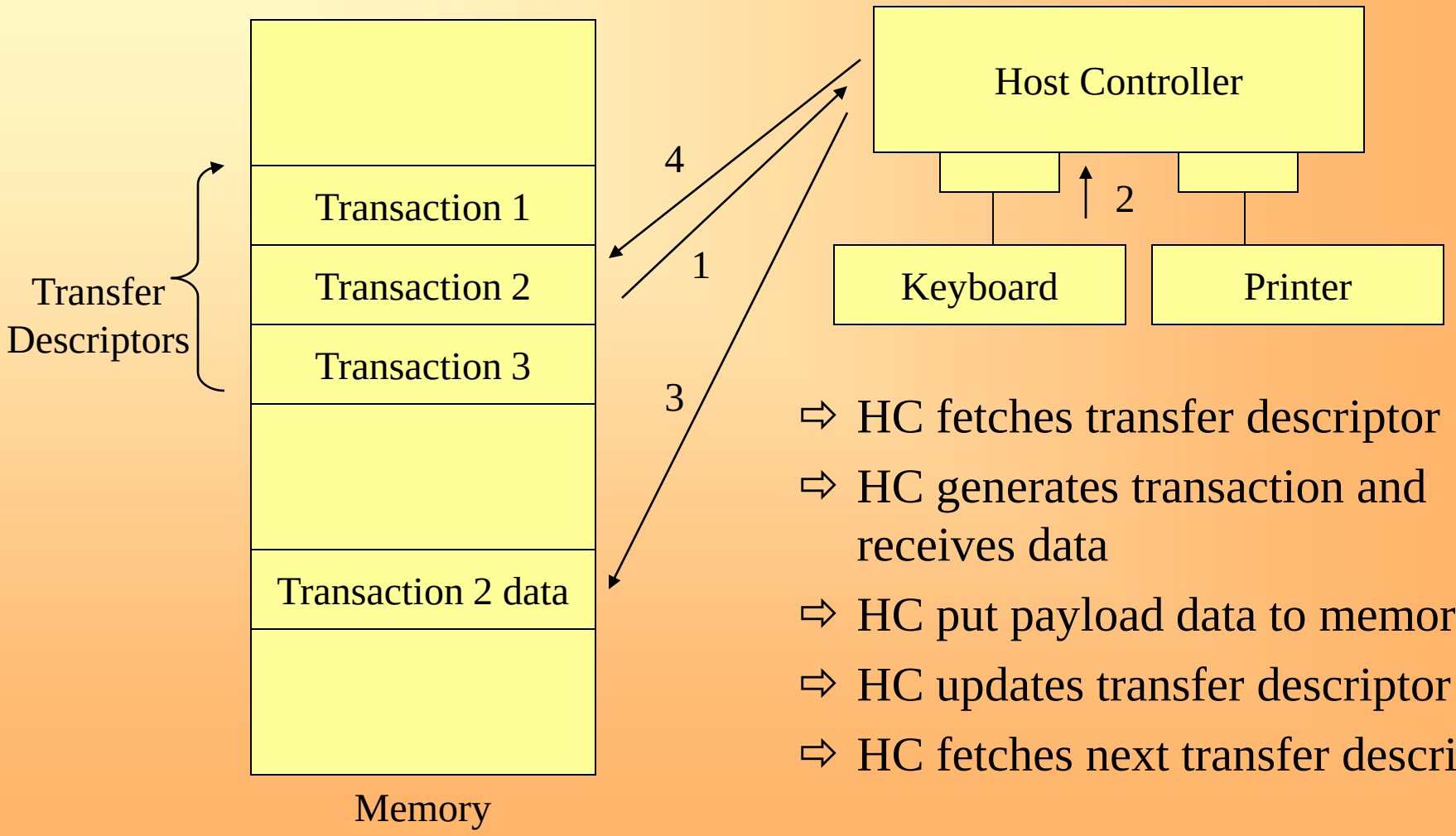
- Scheduling and performing of all USB transactions
- Generation of USB frameworks (SOF token)
- Implemented in PCI master
  - DMA-access allows read/write/modify of data-structures generated by the host controller driver
- Host controller control register mapped into PCI-I/O-address-space
- Host-controller merged with root-hub
- Host-controller driver manages root-hubs
- 2 versions of implementation: UHCI / OHCI

How Transactions are generated (1)



- ⇨ HC fetches transfer descriptor
- ⇨ HC fetches payload data
- ⇨ HC generates transaction and sends data
- ⇨ HC updates transfer descriptor
- ⇨ HC fetches next transfer descriptor

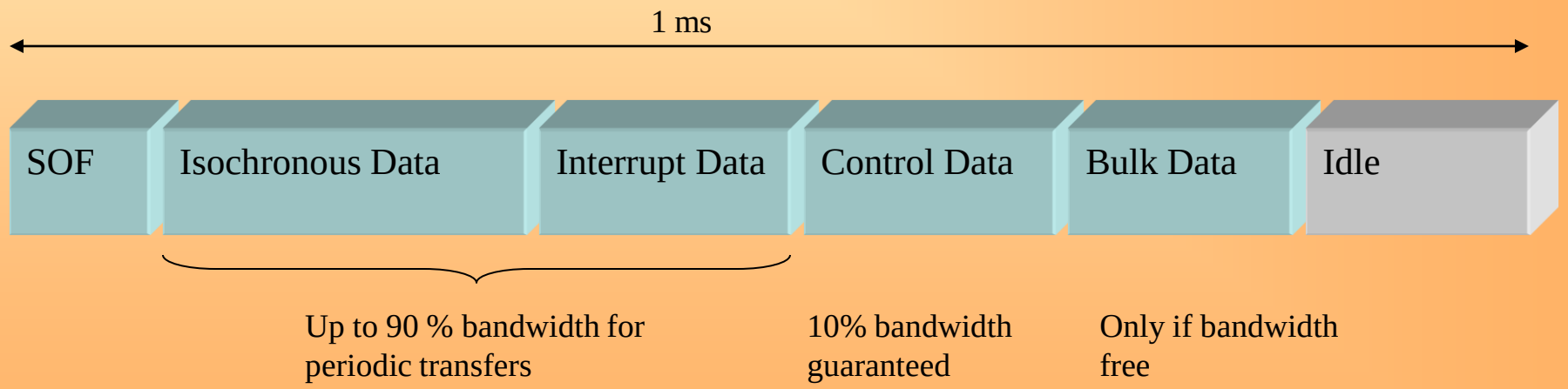
How Transactions are generated (2)



- ⇨ HC fetches transfer descriptor
- ⇨ HC generates transaction and receives data
- ⇨ HC put payload data to memory
- ⇨ HC updates transfer descriptor
- ⇨ HC fetches next transfer descriptor

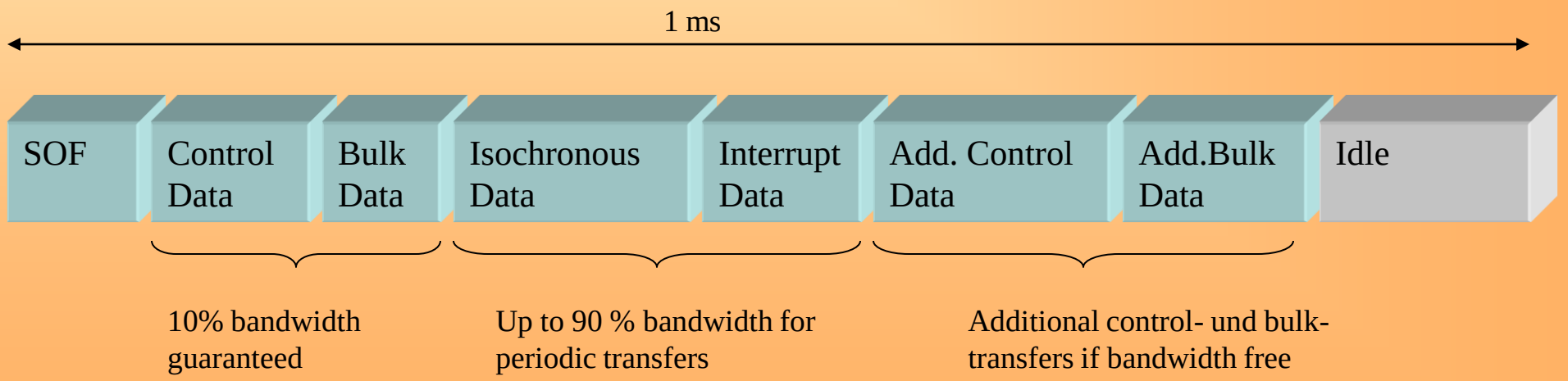
Universal Host Controller (UHC)

- Developed and specified by INTEL
- Integrated in all motherboard chipsets delivered by INTEL and VIA
- Supports root-hubs with (only) 2 downstream ports
- One control transfer per device and frame
- Scheduling:



Open Host Controller (OHC)

- Specified by Compaq / Microsoft / National Semiconductor
- Integrated in motherboard chipsets or add-on-PCI-card
- Supports root-hubs with more than 2 downstream ports
- Up to 13 control transfer per device and frame
- Control / Bulk transfers possible if less bandwidth available
- Scheduling:

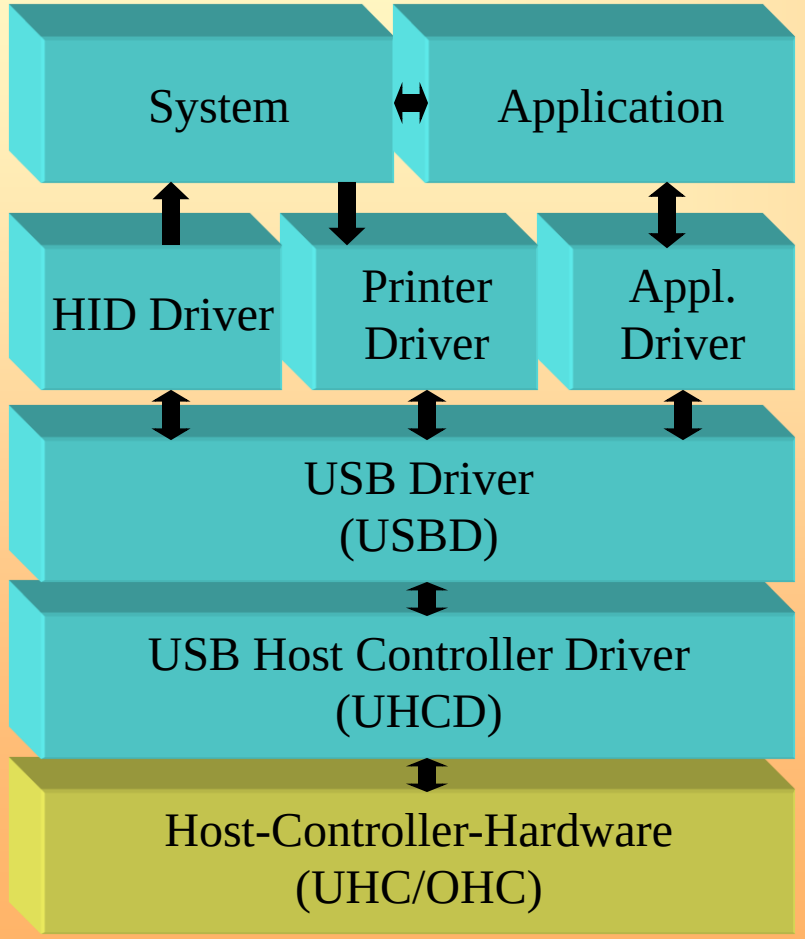


Host-Software Layer-Model

Tasks of USB-Host-Software

- Controlling the USB interface
- Configuration of all attached USB devices
- Bus- and power-management
  - power management
  - bandwidth management
- Pipelining of data from/to devices
- Error-detection and -handling

Host-Layer-Model



⇨ Class-specific driver
- looks to a USB device as a collection of endpoints / pipes
- don‘t know real EP-addresses, maximum packet size, USB-address...

⇨ Hardware-driver for UHC / OCH - implementation
- Schedule management
- Queue management
- Controller management

USB-Implementers Forum

USB-IF History

- 1995: Foundation of the USB Implementers Forum
  - Compaq www.compaq.com
  - Digital Equipment Corporation www.digital.com
  - IBM PC Company www.ibm.com
  - **INTEL** www.intel.com
  - Microsoft www.microsoft.com
  - NEC www.nec.com
  - Northern Telecom
  - nowadays more than 1000 members worldwide
- Annual fee: 4000,- $
  - Free vendor-ID (otherwise 2000,-$ for a 2 year term)

USB-IF-Tasks

- Administration of all USB activities
- Working of the USB specification
- Organizing of workshops and compliance-tests („plugfest“)
- Organizing of developer-conferences / fairs
- Controlling the usage of the USB-logos
- Controlling usage of vendor-IDs

- Support via Web: www.usb.org/developers
  - Specifications / links / presentations
  - Several test-tools running under Win2000/WinXP
  - Web board

USB-IF Benefits

- „Eligibility to participate in free USB-IF sponsored quarterly Compliance Workshops
- Free Vendor ID (if one has not been previously assigned)
- Opportunities to participate in USB-IF marketing programs and events, such as retail newsletters, store endcaps, featured products, etc
- A company listing in the USB key contacts list
- Eligibility for inclusion in the USB current products list on the usb.org web site and in periodic USB-IF retail newsletters
- A waived logo administration fee when joining the USB-IF logo program
- Discounts on Developer Conferences, products in the e-store, etc
- Eligibility to participate in Device Working Groups
- 5 free copies of the specification to new members of the USB-IF
- And many others... “

And the Story continues..

- “Universal Serial Bus Revision 2.0 Specification”
  - High Speed with up to 480 MBit/s
- “Universal Serial Bus Revision 3.0 Specification”
  - SuperSpeed with up to 5 GBit/s
- Wide and continuously growing range of USB device classes
  - “Battery Charging v1.2”
  - “Network Control Model Device Specification v1.0”
  - “USB Attached SCSI Protocol (UASP) v1.0”
  - …
- “Inter-Chip USB Supplement to USB 2.0 Specification”, Rev. 1.0, 2006
- “High-Speed Inter-Chip USB Electrical Specification”, Rev. 1.0, 2007
- “On-The-Go and Embedded Host Supplement to the USB Revision 3.0 Specification”, 2011
- “Wireless USB Specification”, Rev. 1.1, 2010