USB 2.0
Differences and Enhancements

Dipl.-Ing. Stefan Schulze

emsys Embedded Systems GmbH, Ilmenau - Germany

e-mail: stefan.schulze@emsys.de

History

- Specification approved in April 2000
- First products hit the market in Spring 2001
- Strongly pushed by Microsoft and Intel
- High-Speed USB got several awards, e.g.
  - PC Magazine's "Technical Excellence Award" in Specifications Category
  - CNET's "Best Emerging Technology Award" at PC Expo

USB 2.0

Interoperability and Framework

Back- / Forward-Compatibility

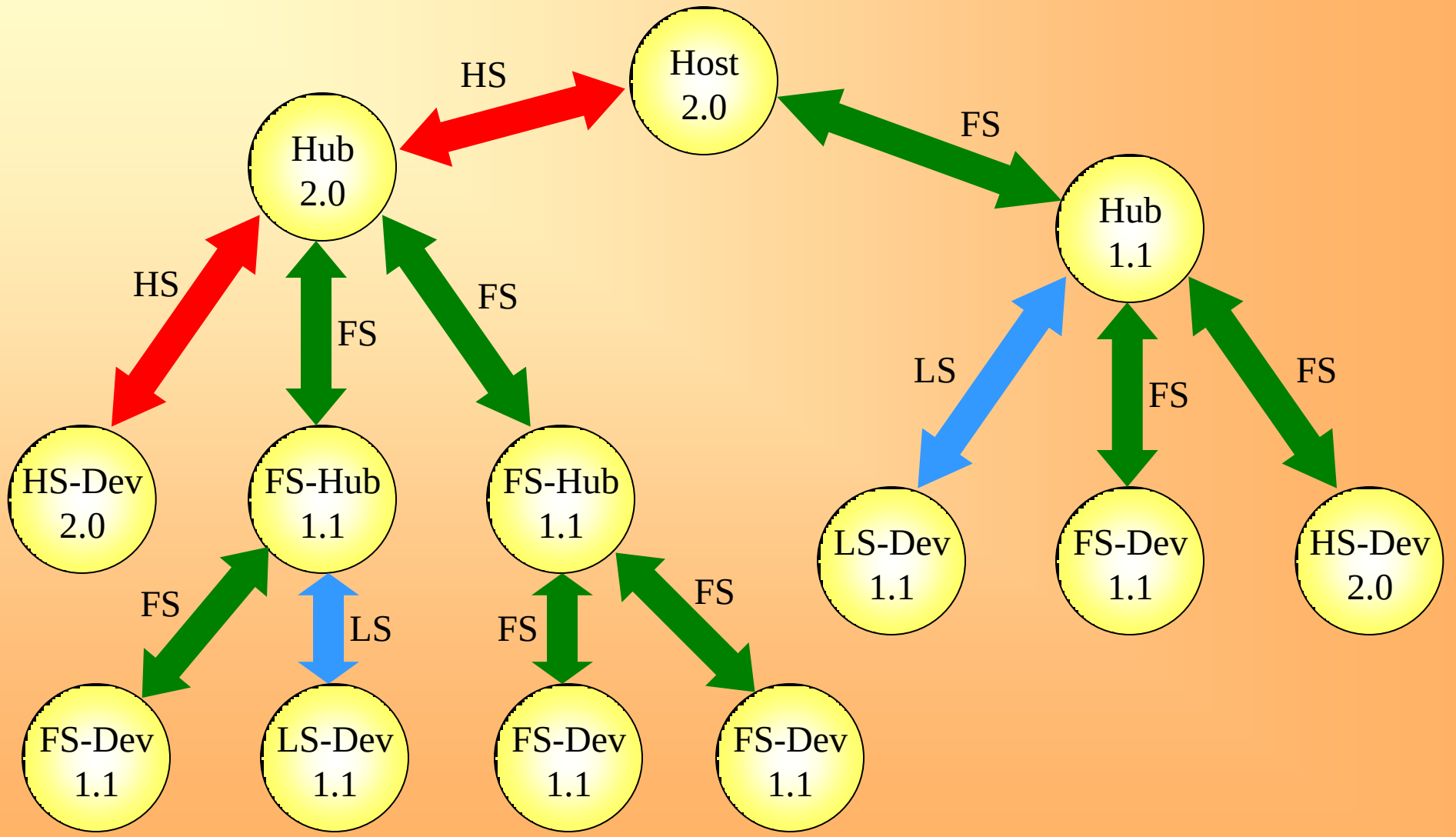
- USB 2.0 includes all the functionality of USB 1.1
- USB 1.1 and 2.0 Devices can be mixed
- Usage of the existing 1.1 compatible cables and connectors
- 3 Speed grades:

  Low-Speed: 1,5 Mbit/sec

  Full-Speed: 12 Mbit/sec

  High-Speed: 480 Mbit/sec
- Hubs work as „Speed Translators“ between High-Speed and Full-/Low-Speed Paths
- Devices using High-Speed @ 480 Mbit/sec must run on 1.1-Hubs with Full-Speed @ 12 Mbit/sec

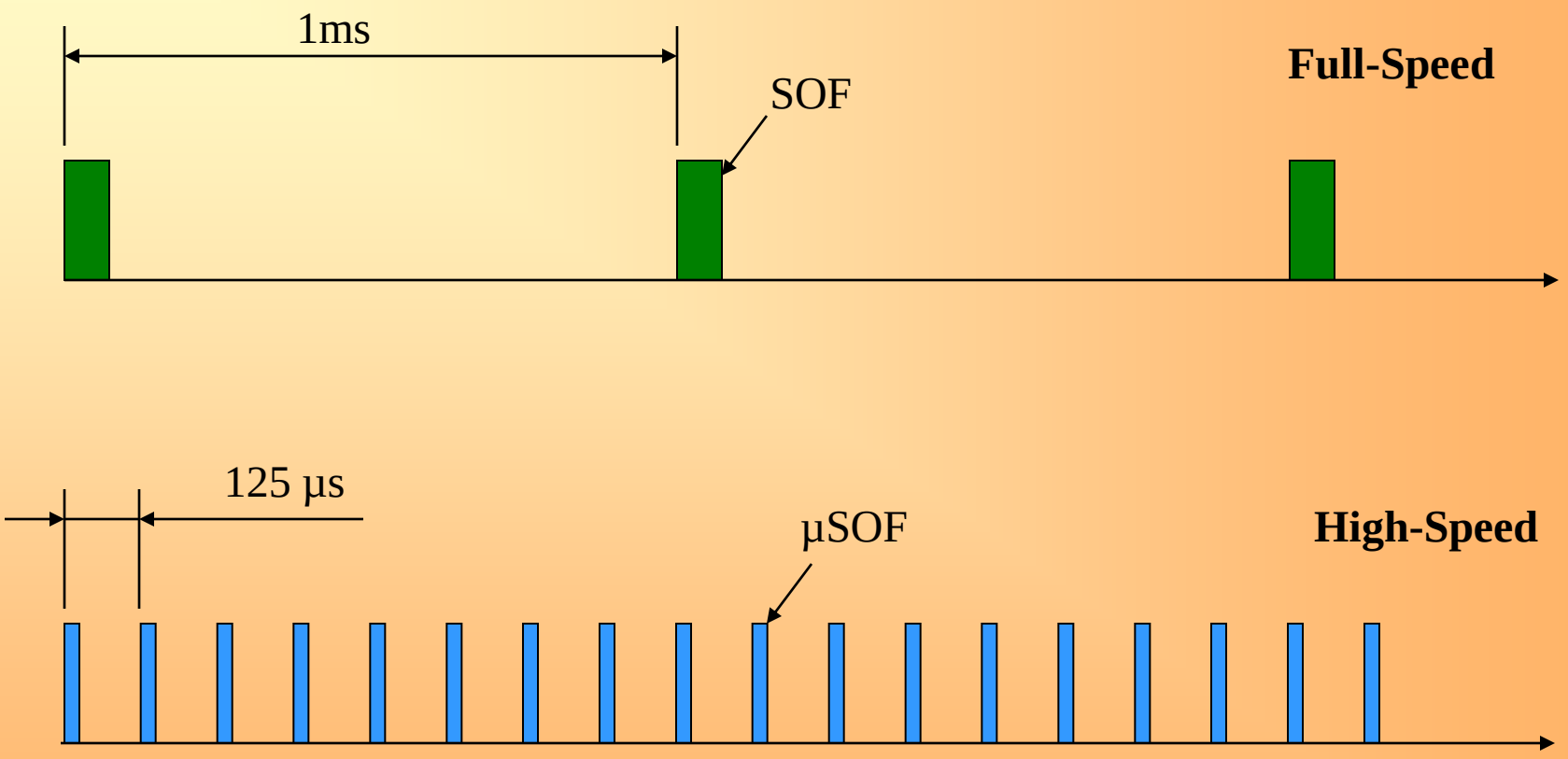
USB - 1.1/2.0 - Speed-Topology



What didn't change

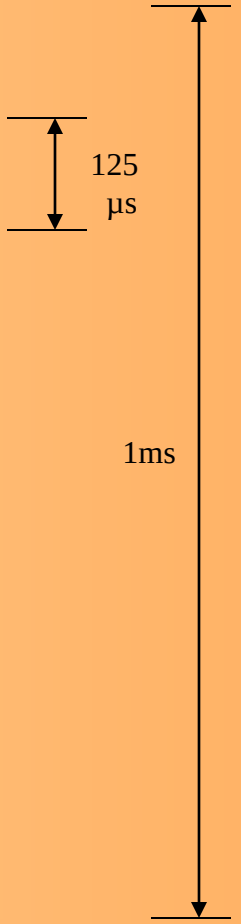
- Same host/device model
- Same tiered star topology
- Same basic protocols and tokens (with enhancements)
- Same descriptors (with enhancements)
- Same software interface on host (USBD-driver-stack)
- Same power distribution and consumption-restrictions
- Same power management features
- Same cables & connectors (must be *really* USB 1.1 compliant!)

Micro frames



Empty HS-Framework

SE0 / Idle	HS-SYNC	SOF	Frame #	CRC5	Ext.HS-EOP
59936	...0000000000000001	0xA5	0x002(0)	0x1F	0x7FFFFFFFFF
SE0 / Idle	HS-SYNC	SOF	Frame #	CRC5	Ext.HS-EOP
59936	...0000000000000001	0xA5	0x002(1)	0x1F	0x7FFFFFFFFF
SE0 / Idle	HS-SYNC	SOF	Frame #	CRC5	Ext.HS-EOP
59936	...0000000000000001	0xA5	0x002(2)	0x1F	0x7FFFFFFFFF
SE0 / Idle	HS-SYNC	SOF	Frame #	CRC5	Ext.HS-EOP
59936	...0000000000000001	0xA5	0x002(3)	0x1F	0x7FFFFFFFFF
SE0 / Idle	HS-SYNC	SOF	Frame #	CRC5	Ext.HS-EOP
59936	...0000000000000001	0xA5	0x002(4)	0x1F	0x7FFFFFFFFF
SE0 / Idle	HS-SYNC	SOF	Frame #	CRC5	Ext.HS-EOP
59936	...0000000000000001	0xA5	0x002(5)	0x1F	0x7FFFFFFFFF
SE0 / Idle	HS-SYNC	SOF	Frame #	CRC5	Ext.HS-EOP
59936	...0000000000000001	0xA5	0x002(6)	0x1F	0x7FFFFFFFFF
SE0 / Idle	HS-SYNC	SOF	Frame #	CRC5	Ext.HS-EOP
59936	...0000000000000001	0xA5	0x002(7)	0x1F	0x7FFFFFFFFF
SE0 / Idle	HS-SYNC	SOF	Frame #	CRC5	Ext.HS-EOP
59936	...0000000000000001	0xA5	0x003(0)	0x1F	0x7FFFFFFFFF



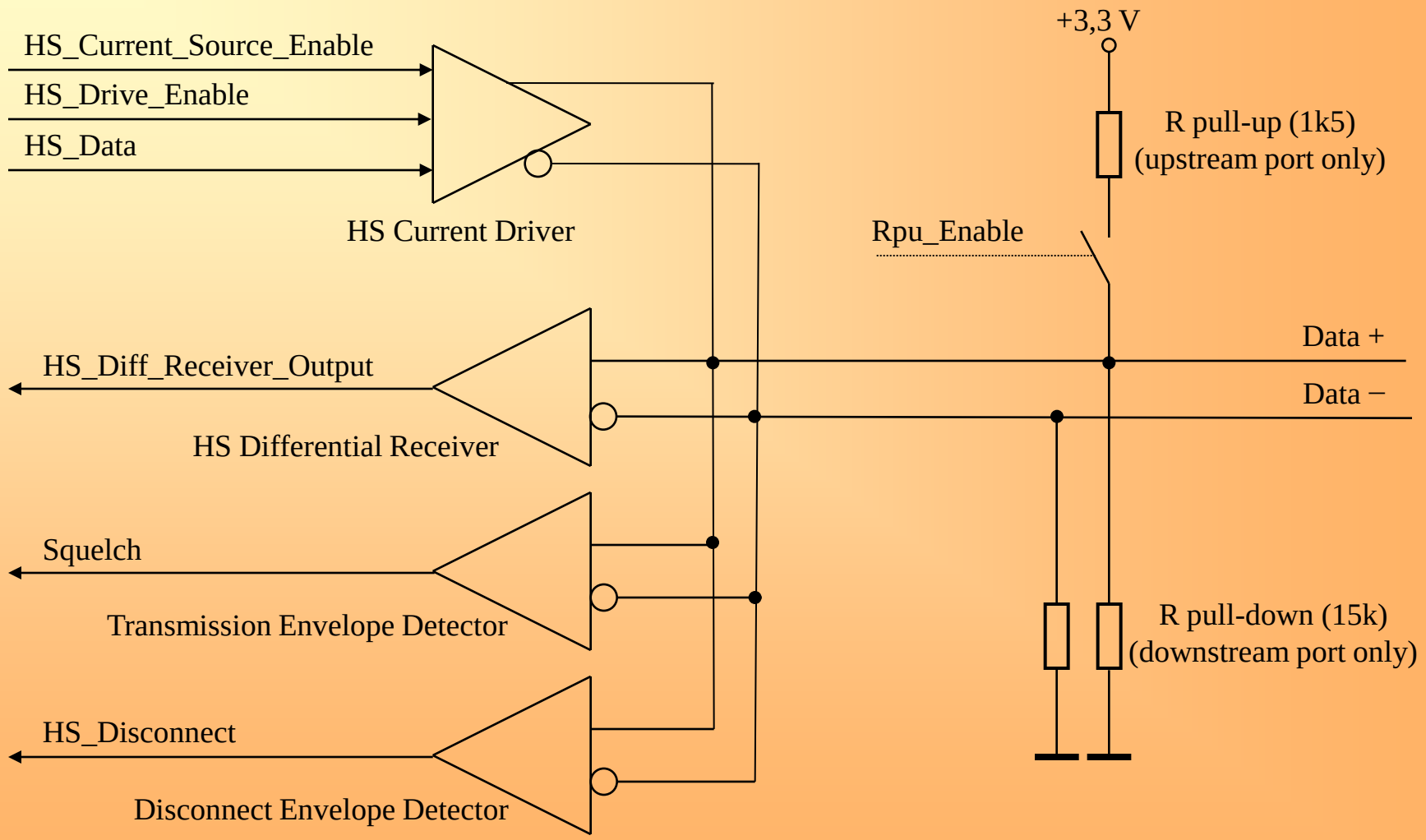
USB 2.0

Changes on Physical Layer

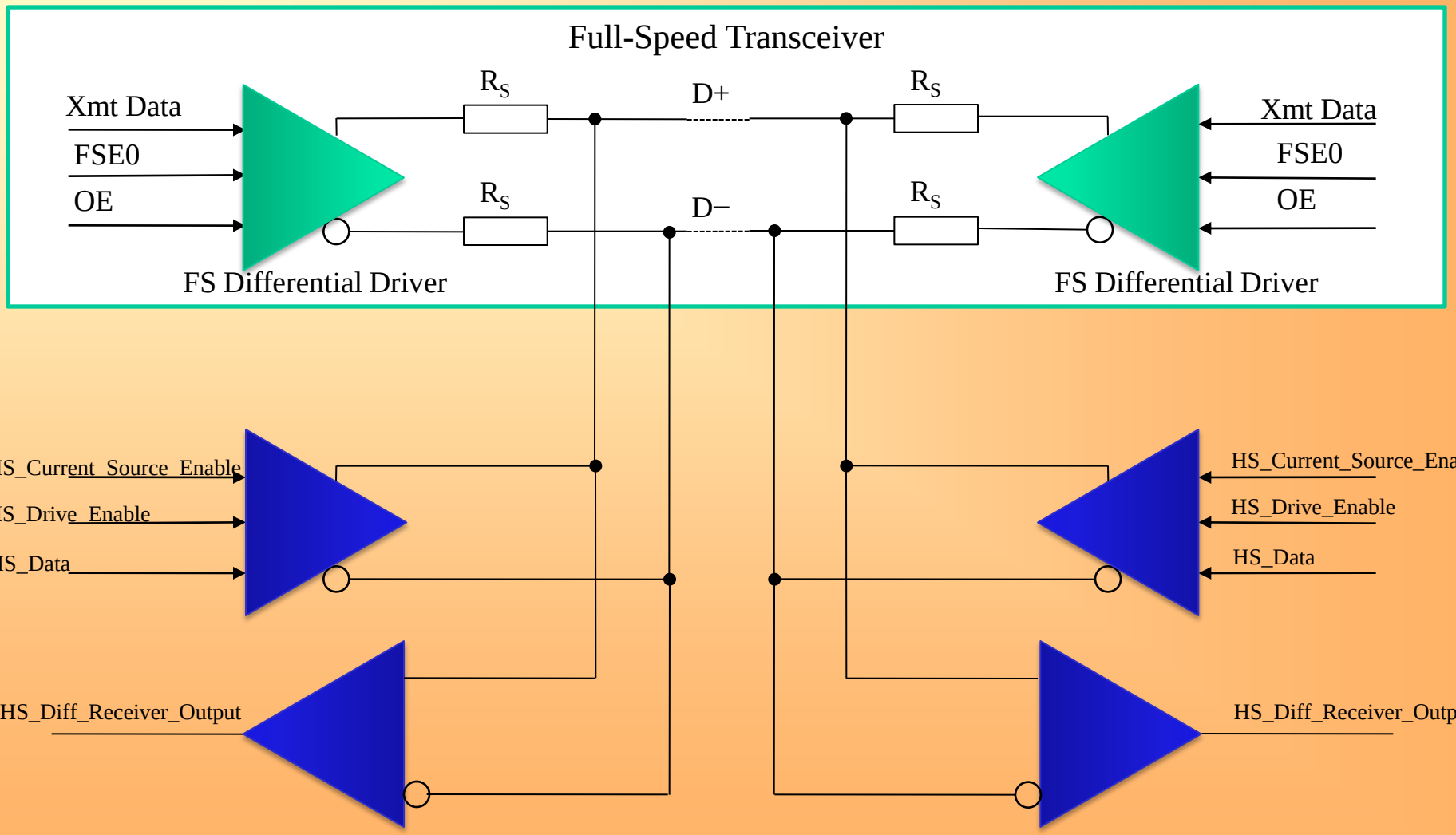
Transceiver

- Devices includes 1.1 **and** 2.0 transceivers in **parallel**
- Dual termination to ground at source and load to enable high signal integrity (45 Ω on source and target)
  - done by Full-Speed transceivers driving SE0 on both lines during High-Speed mode
- Current-drivers for High-Speed (17,78 mA)
  - leads to voltage-hub of 400 mV @ 22,5 Ω
- Keep the Voltage-drivers for Low-/Full-Speed (driven hub of 3,3 V)
- Switchable pull-up-resistor at D+ line (removed in High-Speed mode)
- When device enters HS-Mode, Pull-up resistors are disconnected

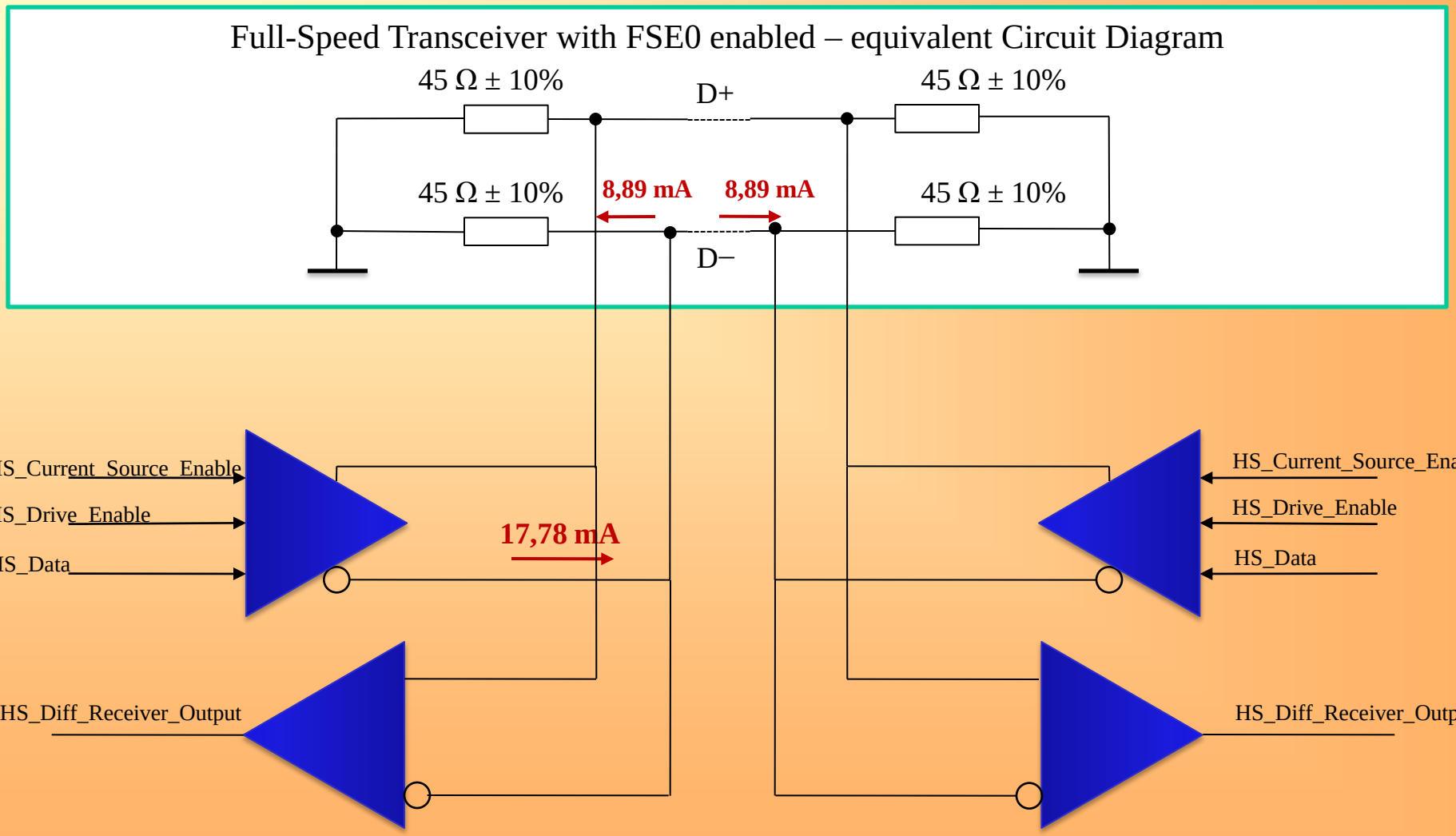
High-Speed-Transceiver



High-Speed Termination



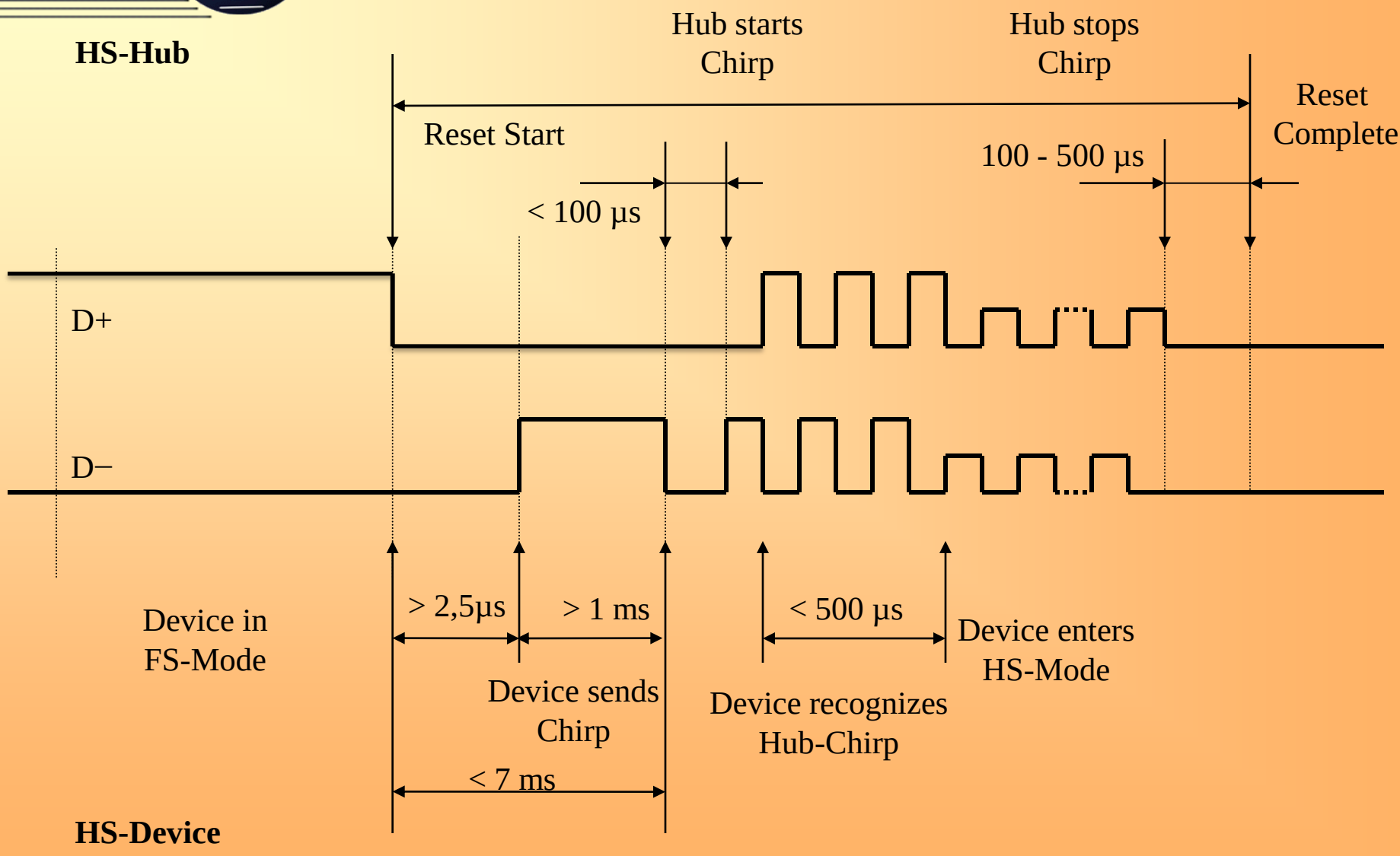
High-Speed Termination (2)



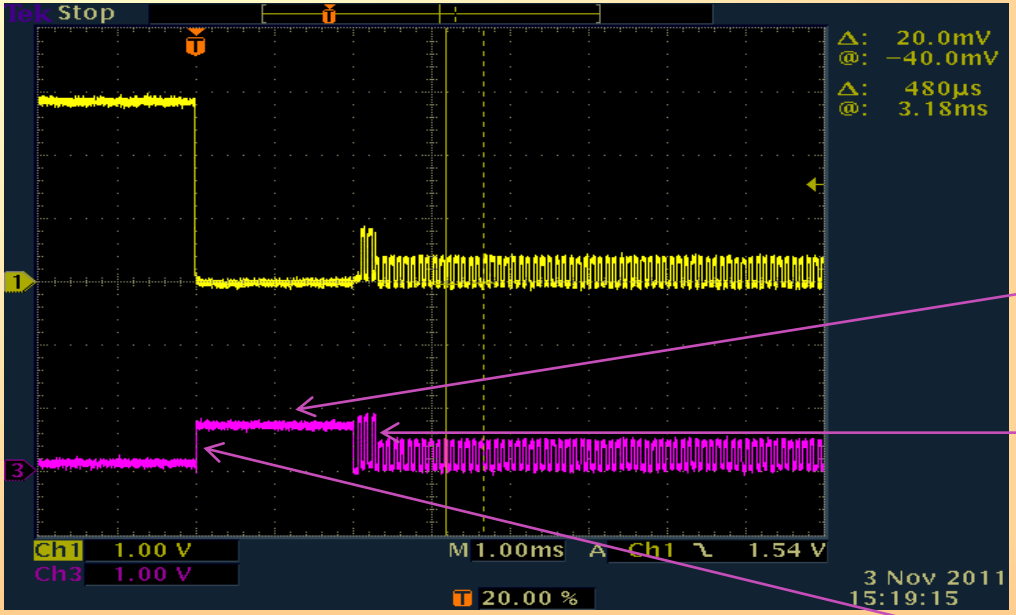
HS Differential Data Receiver

- Receives differential signaling with amplitude as small as ± 200 mV
- Reception of data is qualified by envelope detector
  - Squelch: when differential amplitude is < 100 mV
  - ! Squelch: when differential amplitude is > 150 mV
  - Prevent indication of Squelch during crossover of data-lines
  - Must react in less than 4 bit times

High-Speed-Detection during Reset



Sample Scope Trace



- Yellow: D+
- Purple: D−

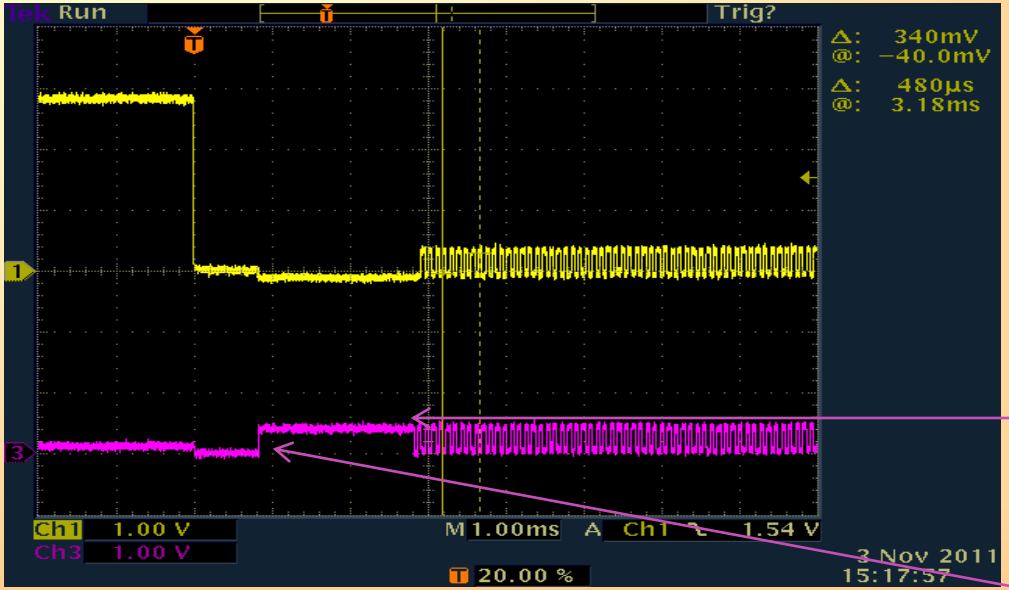
Voltage level 800 mV

Voltage level drops to 400 mV

FS-J	Device Chirp K	Host Chirp K-J-K-J-…

Chirp K driven immediately after USB Reset is detected

HS Chirp with wrong Voltage Level



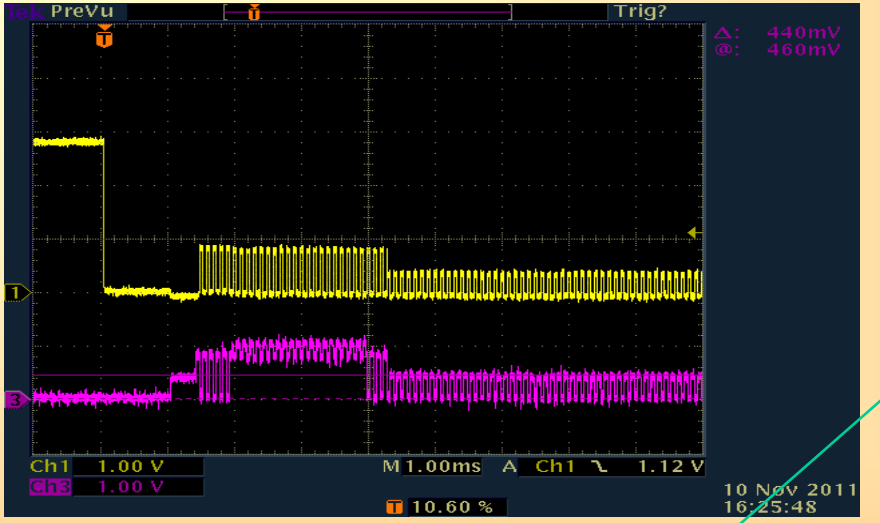
- Yellow: D+
- Purple: D−

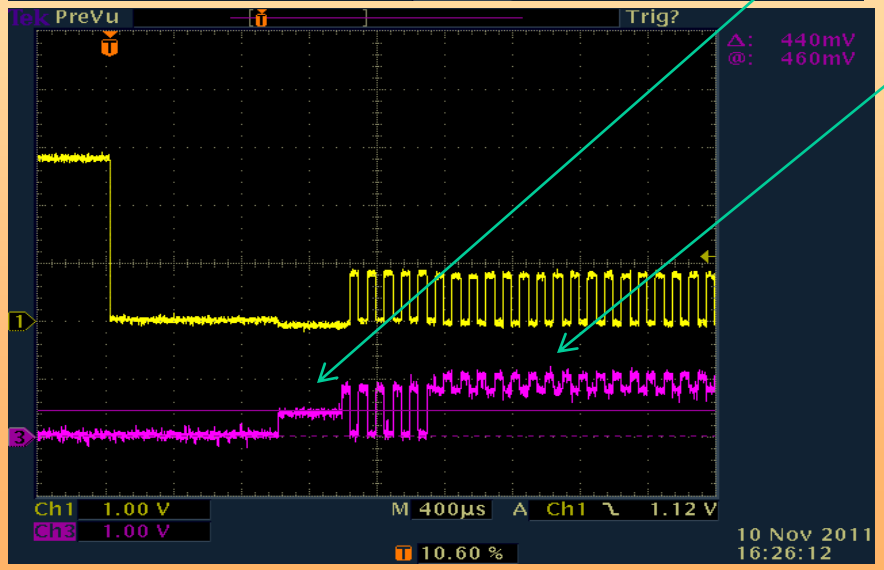
Voltage level **always** 400 mV

FS-J	Device Chirp K	Host Chirp K-J-K-J-…

Chirp K starts about 500 µs after beginning of USB Reset.

..Another wrong Chirp Sequence



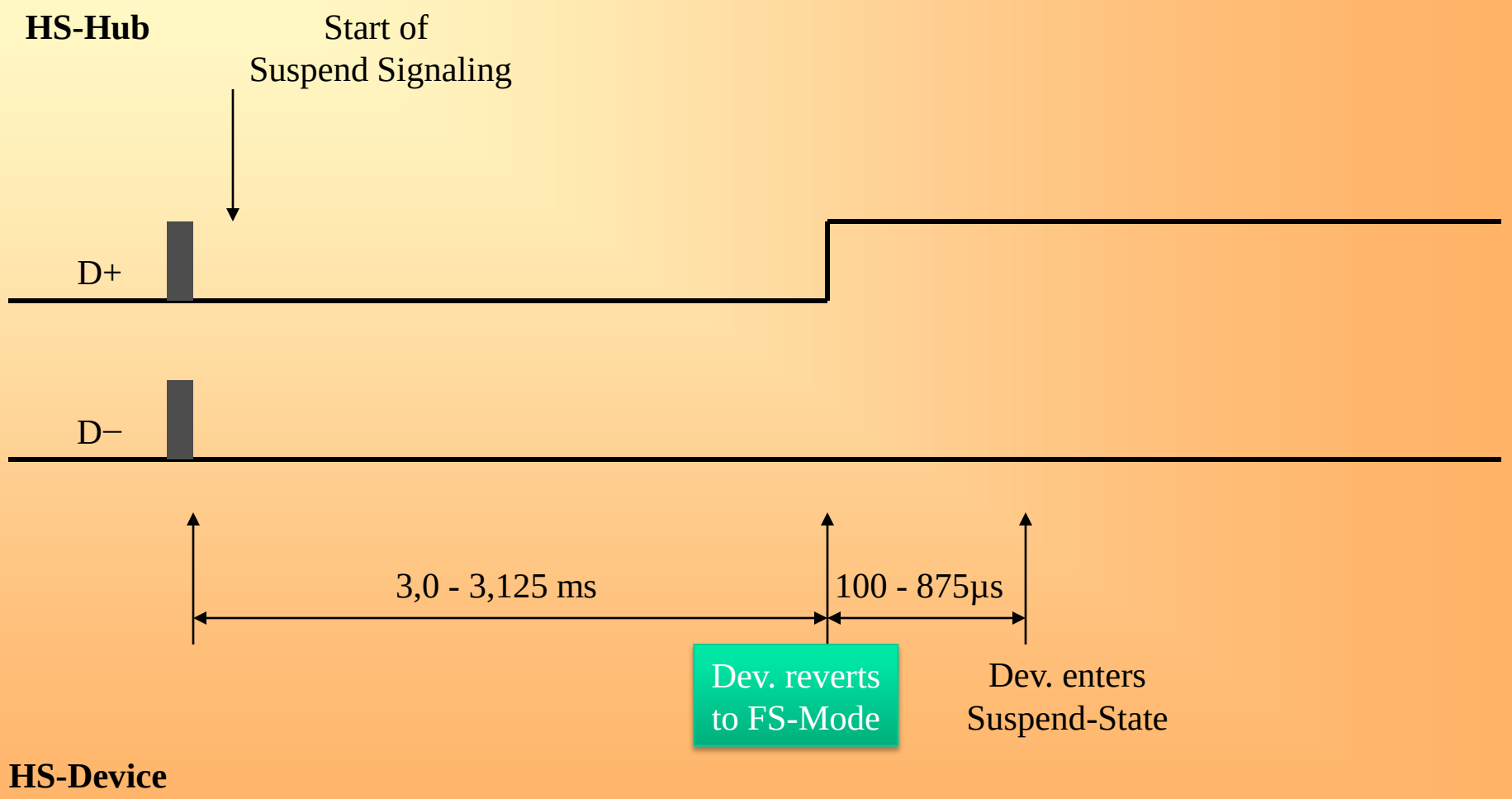


- ⇨ Chirp starts later with wrong voltage level (400 mV)
- ⇨ Chirp duration too short (< 1 ms)
- ⇨ During Host chirp K-J-K-J… response, the termination seems to change

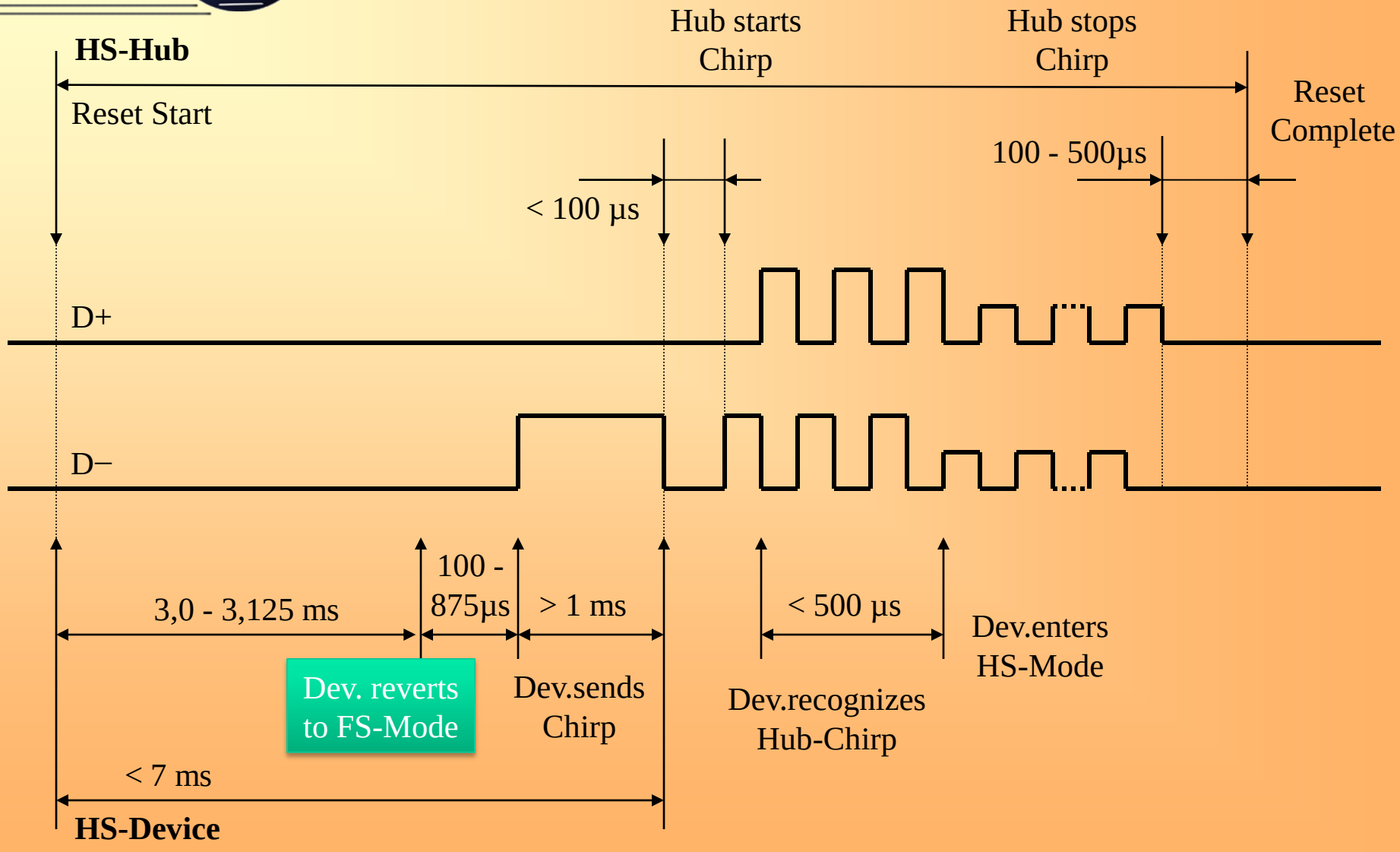
Impact of High-Speed Termination

- The High-Speed Idle State, both data lines are terminated with a low resistance of 22,5 Ω wrt. ground
- Basically the same state as SE0
- Immediate detection of SE0 state as used in Full-/Low-Speed world cannot be used any more for:
  - USB Reset detection (device cannot distinguish between USB Reset and USB Suspend state in High-Speed mode)
  - EOP detection
  - Disconnect detection

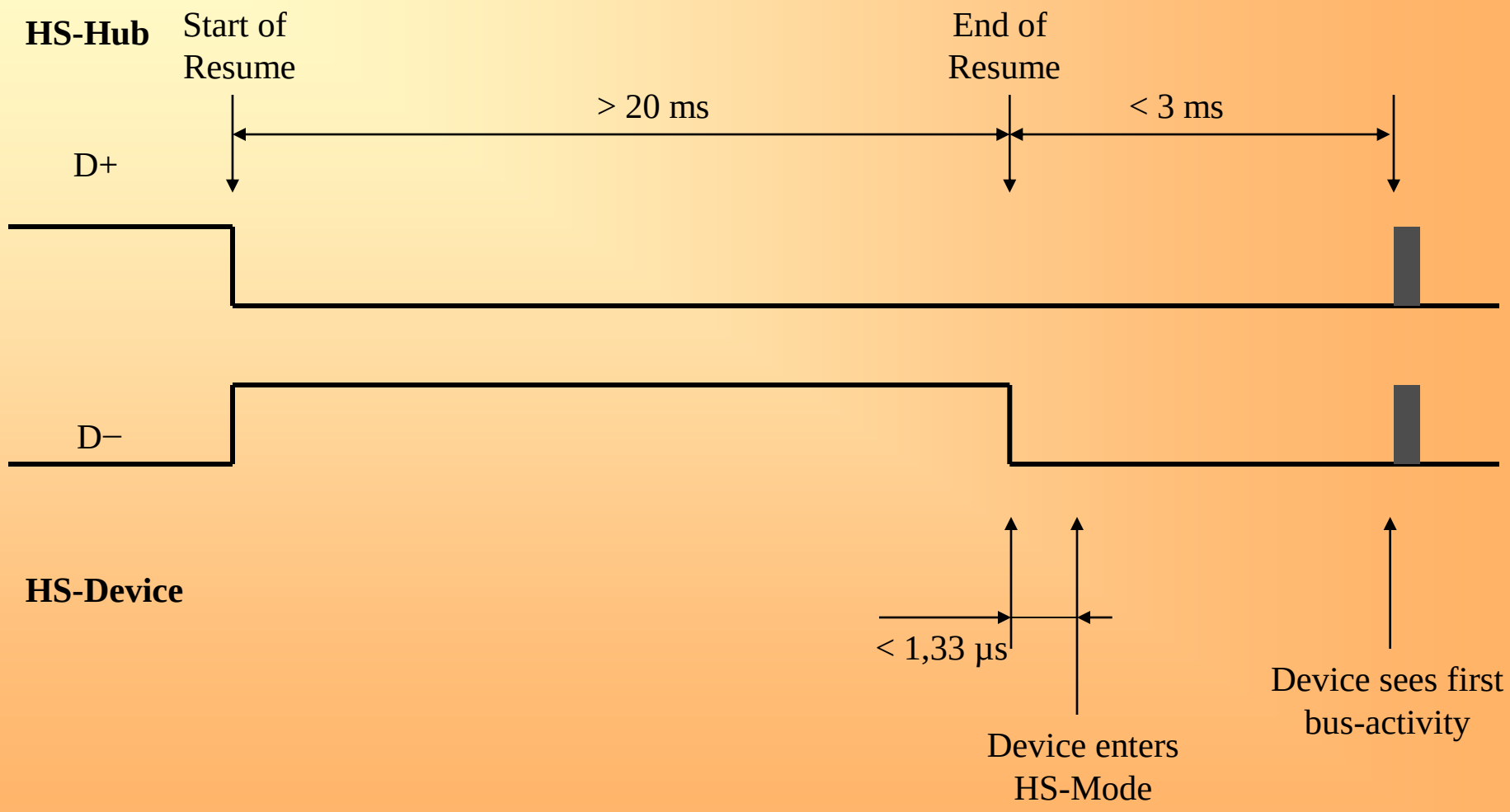
USB Suspend in High-Speed Mode



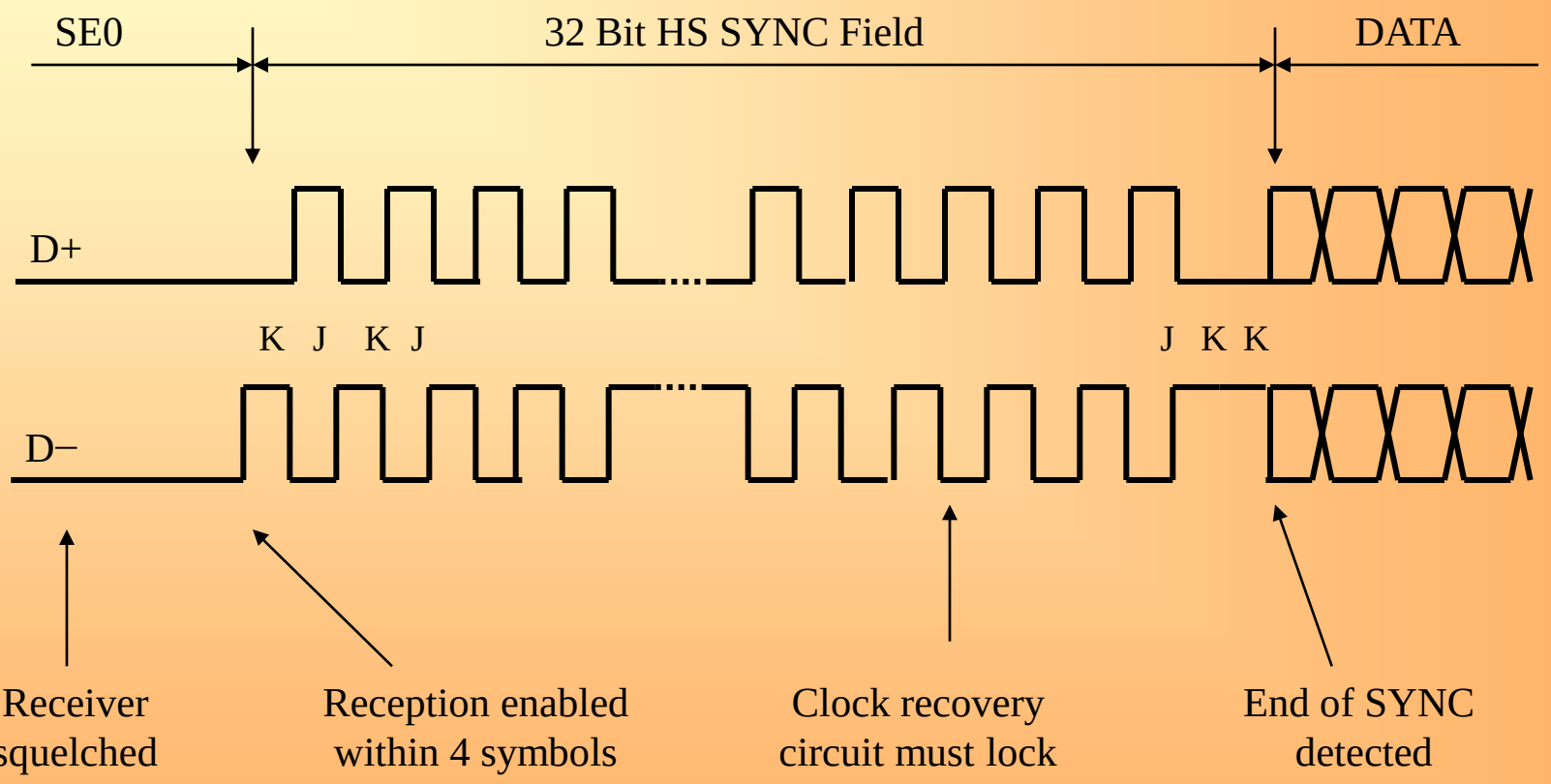
USB Reset in High-Speed Mode



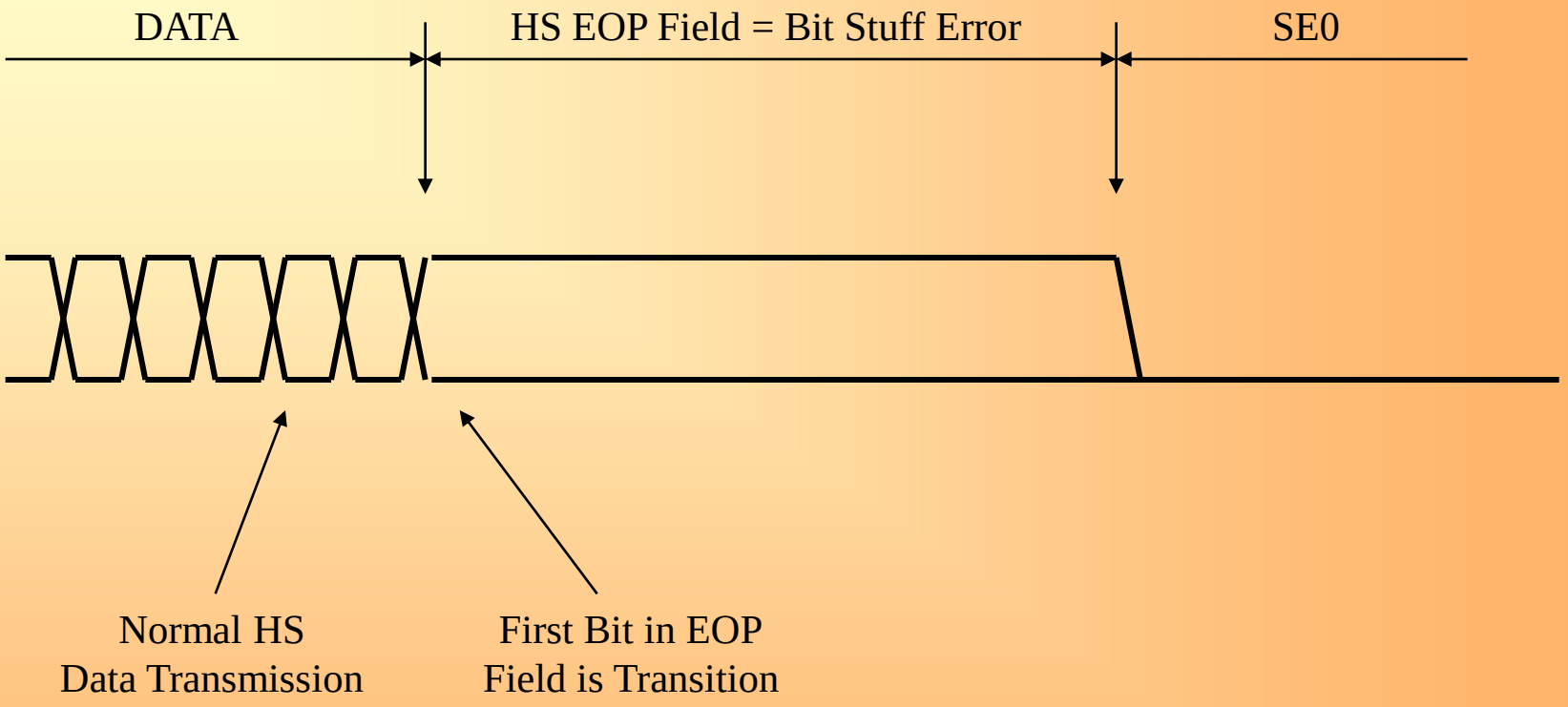
USB Resume-Signaling



High-Speed Start-of-Packet

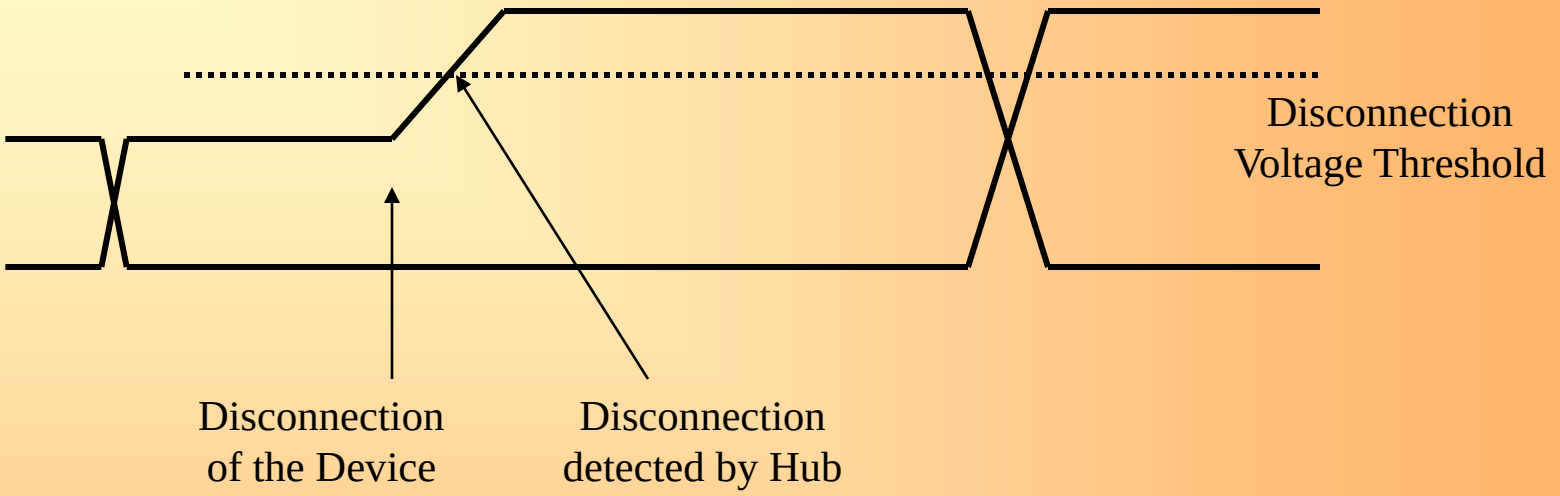


High-Speed End-of-Packet



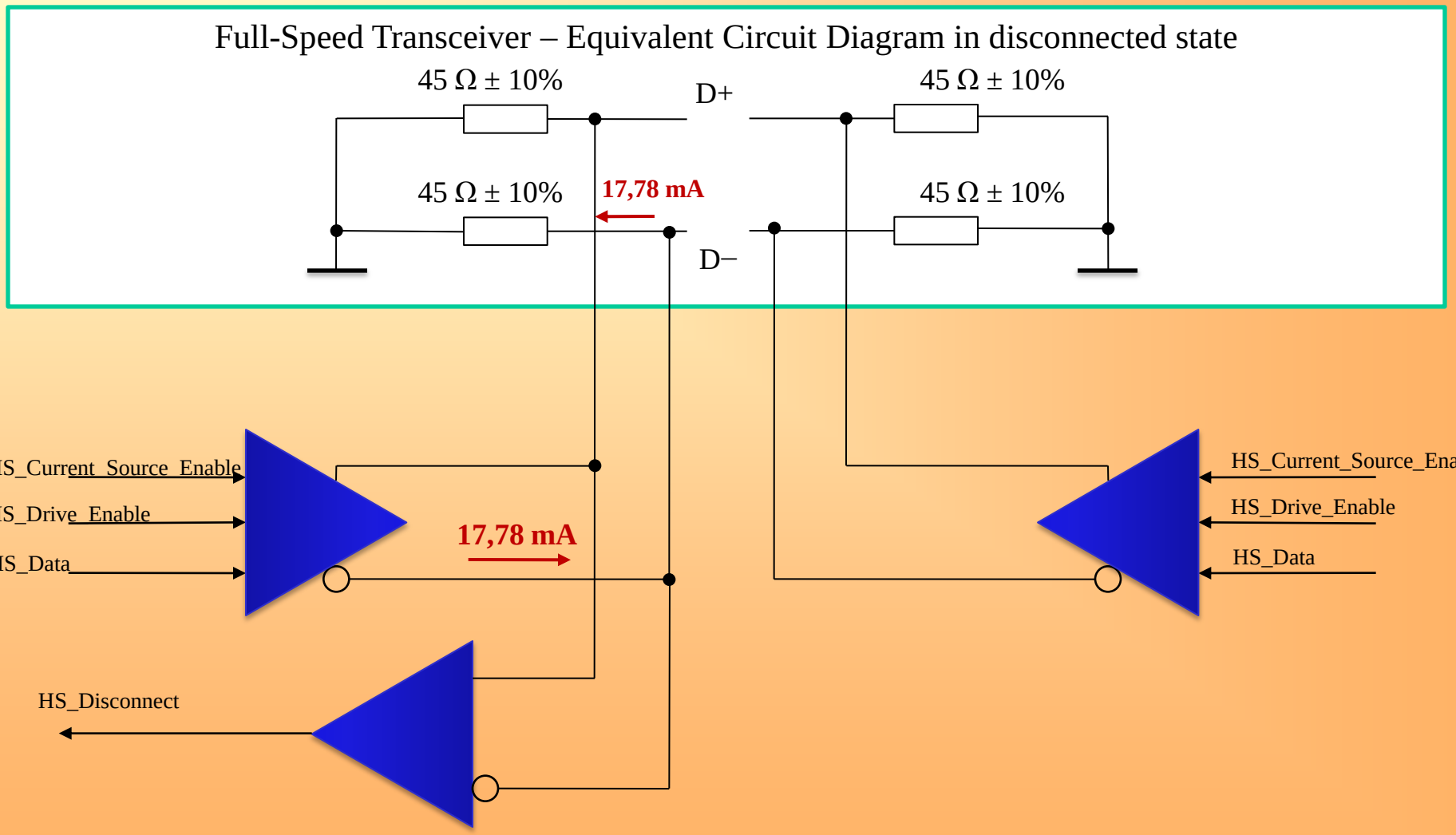
- ⇨ Standard High-Speed EOP: NRZ byte of b01111111 without bit stuffing
- ⇨ SOF High-Speed EOP: 5NRZ bytes without bit stuffing
  - ⇨ b01111111 11111111 11111111 11111111 11111111

Disconnect Detection



- On disconnect the device termination is removed
- Voltage jump from typically 400 mV to 800 mV
  - ≥ 625 mV indicates disconnect
  - ≤ 525 mV does not indicate disconnect
- Check performed during last 8 Bits of EOP from Micro-Frame-Token
- µSOF EOP is extended to 40 bits (due to max. total turn-around time)

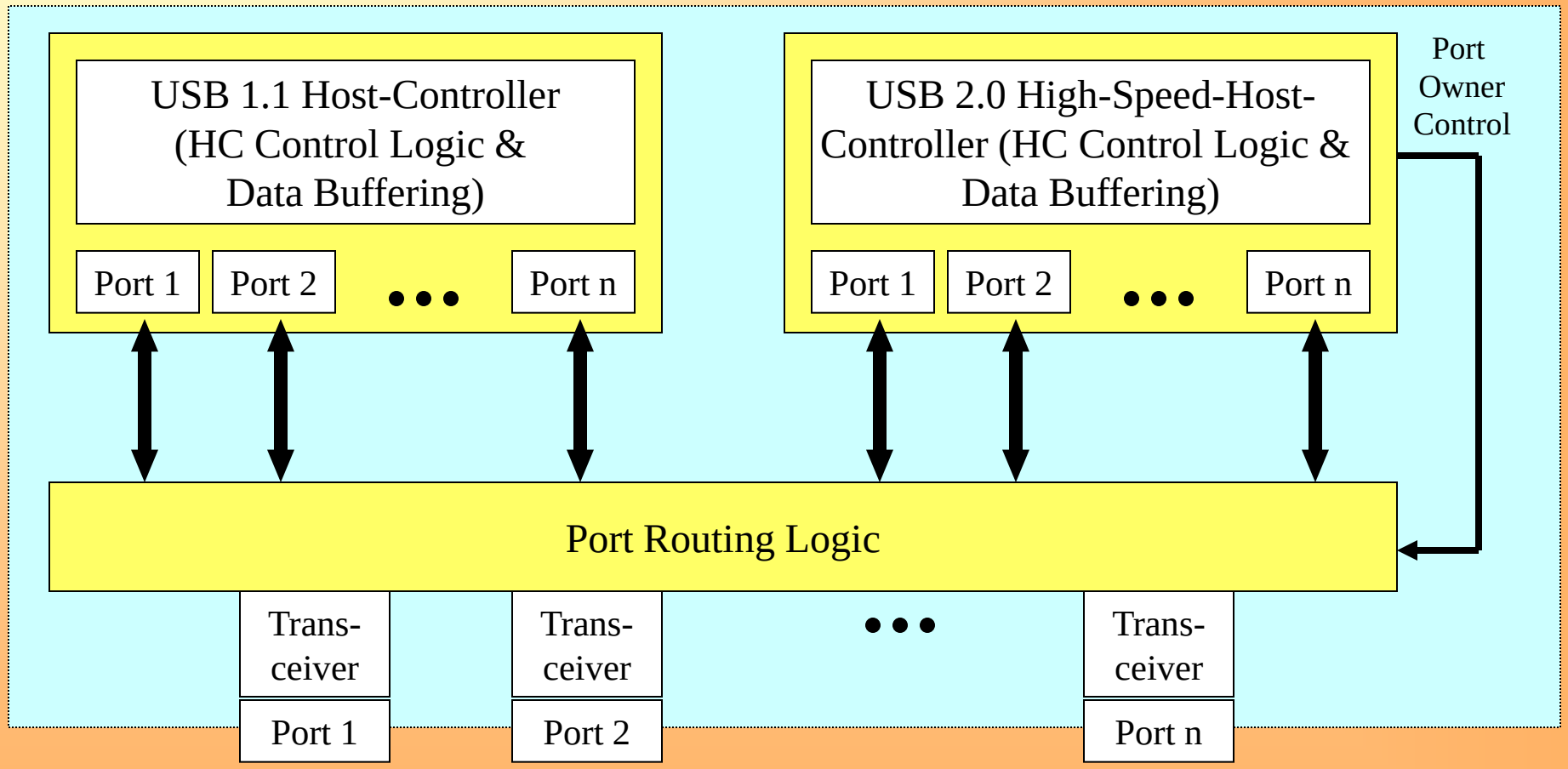
Disconnect Detection (2)



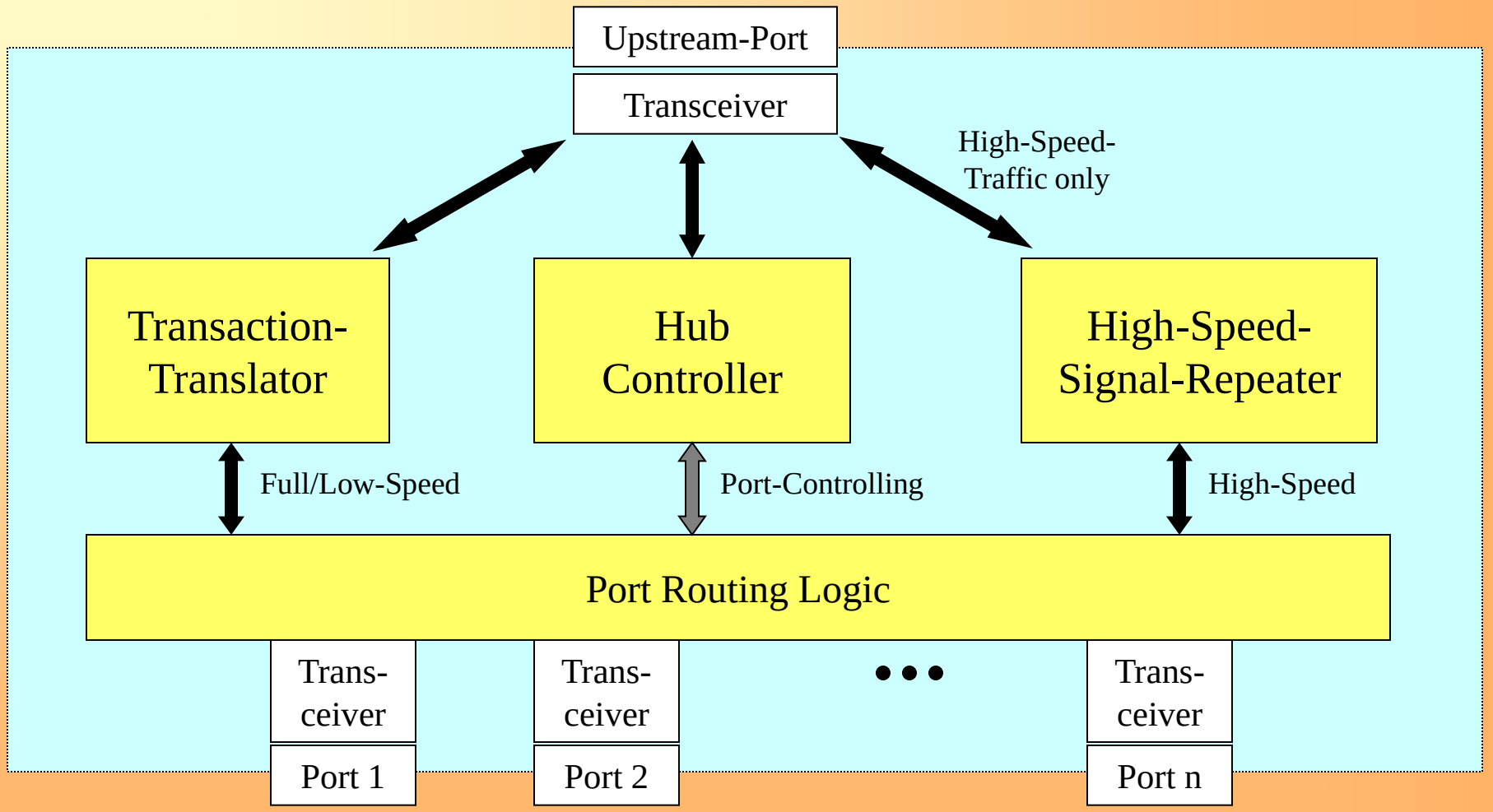
USB 2.0

Changes on Hardware-Components

Host-Hardware



Hub-Hardware



USB 2.0

Changes on Transfer Layer

Packet Sizes

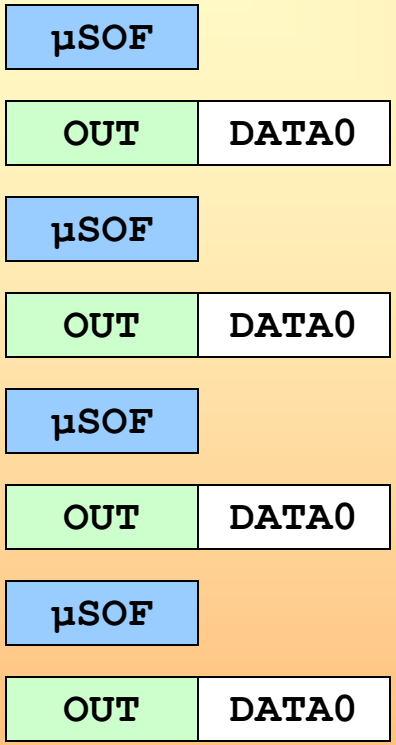
	Low-Speed @ 1,5 Mbit/sec	Full-Speed @ 12 Mbit/sec	High-Speed @ 480 Mbit/sec
Control-Transfer	8 Byte	8, 16, 32, 64 Byte	64 Byte
Bulk-Transfer	-	8, 16, 32, 64 Byte	512 Byte
Interrupt-Transfer	1...8 Byte @ 8...255 ms	1...64 Byte @ 1...255 ms	1...3072 Byte * @ $2^{\text{bInterval-1}}$
Isochronous-Transfer	-	1...1023 Byte @ 1ms	1...3072 Byte * @ $2^{\text{bInterval-1}}$

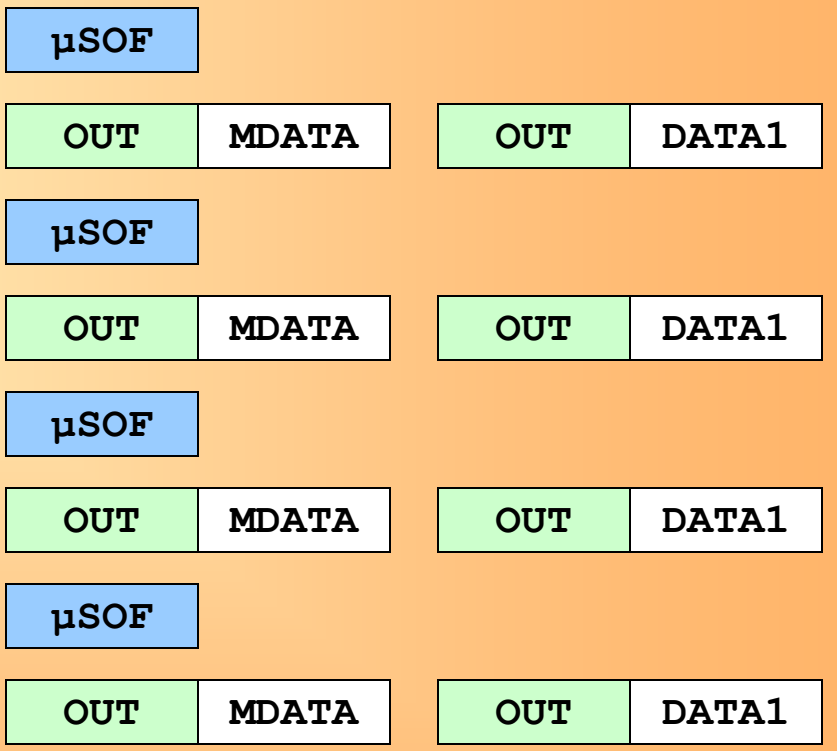
* If > 1024 byte, done with multiple transactions (up to 3 per micro-frame)

High-Bandwidth Transactions

- High-Bandwidth (MULT) transactions are limited to isochronous and interrupt endpoints
- Up to three transactions per micro frame are supported which leads to a possible transfer rate of up to 3072 bytes
- High bandwidth capability and number of transactions are defined in the endpoint descriptor:
  - MaxPacketSize[15:13]: Reserved (0)
  - MaxPacketSize[12-11]: Number of transactions per microframe (1,2 or 3)
  - MaxPacketSize[10-0]: Maximum size of data payload for one packet in bytes

High BW Isochronous OUT Transfer (I)

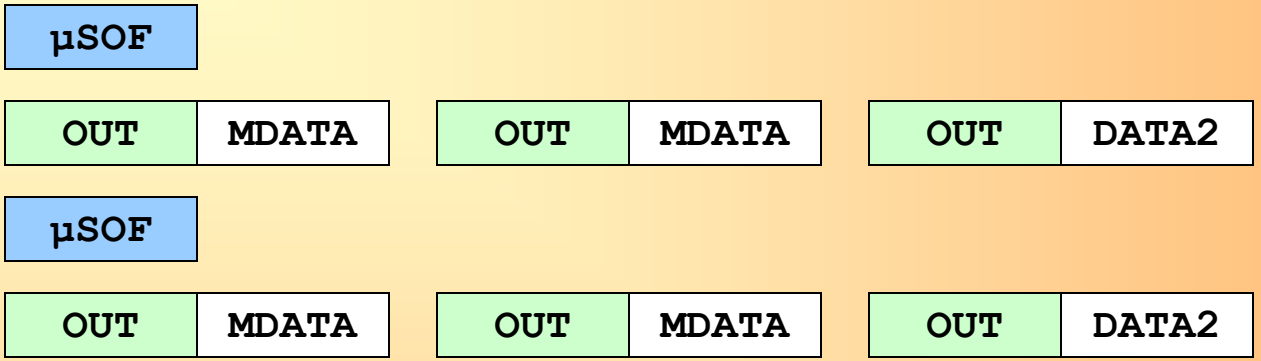




1 Transaction per micro-frame for bandwidth < 1024 Byte

2 Transaction per micro-frame with sizes 513 - 1024 Bytes each packet

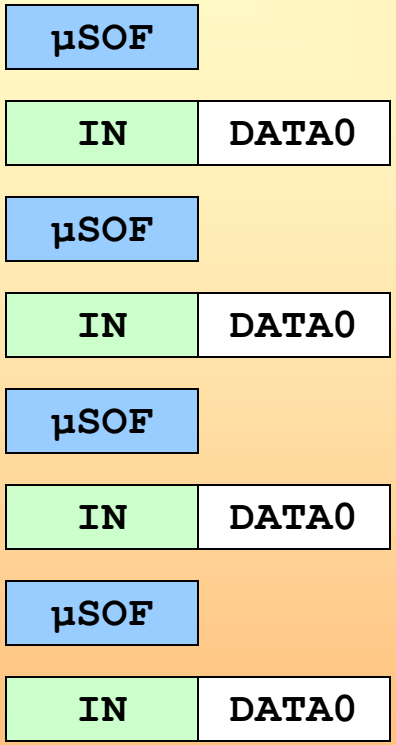
High BW Isochronous OUT Transfer (II)

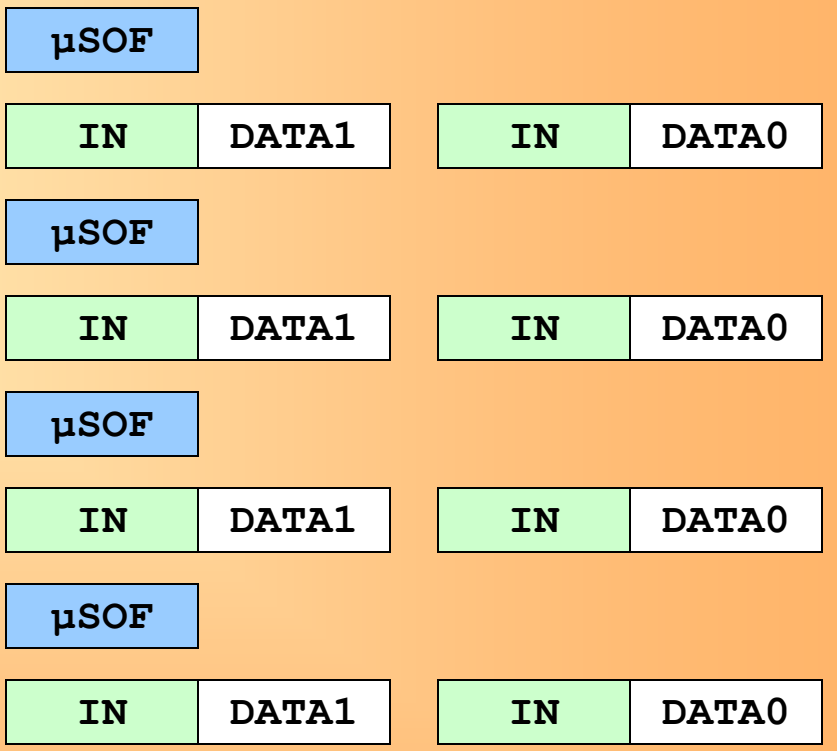


3 Transaction per micro-frame
with sizes 683 - 1024 Bytes each packet

- ⇨ Last data PID indicates how many data packets are sent in a row
- ⇨ First/second data packet uses MDATA PID
- ⇨ Sequence starts again each micro frame

High BW Isochronous IN Transfer (I)

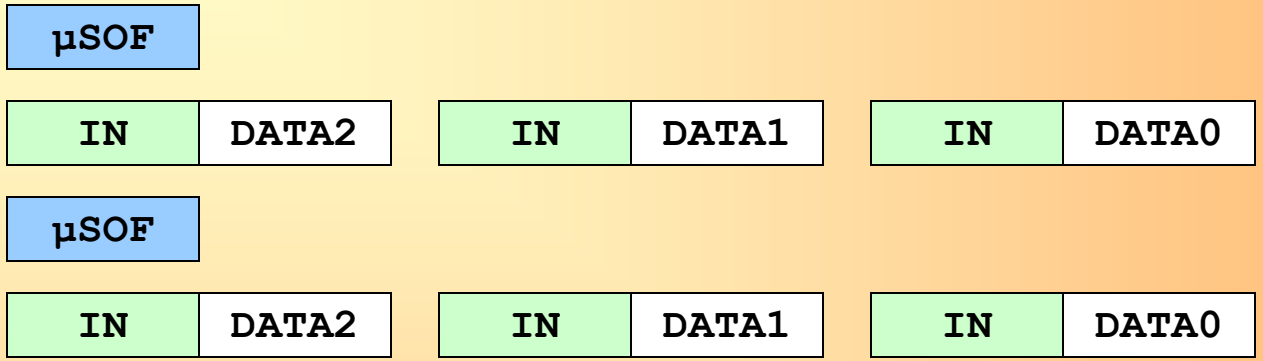




1 Transaction per micro-frame for bandwidth < 1024 Byte

2 Transaction per micro-frame with sizes 513 - 1024 Bytes each packet

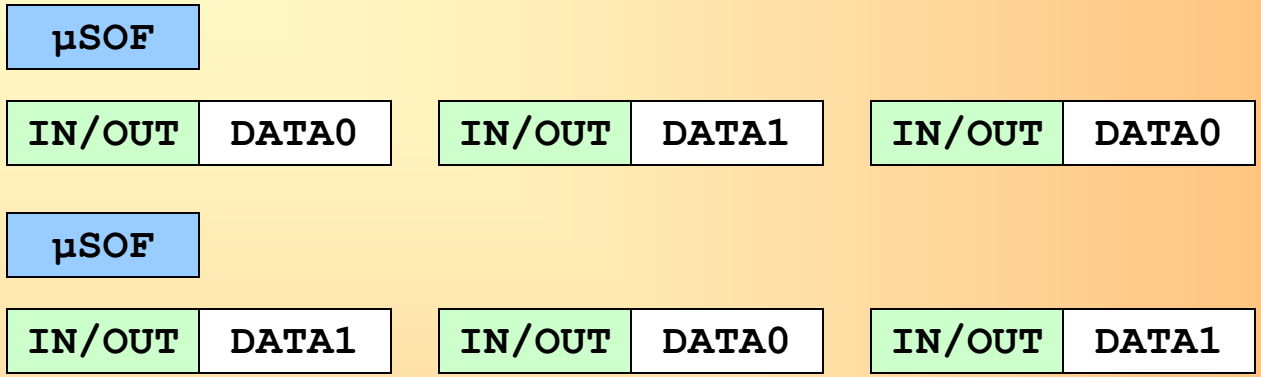
High BW Isochronous IN Transfer (II)



3 Transaction per micro-frame
with sizes 683 - 1024 Bytes each packet

- ⇨ First data PID indicates how many data packets are sent in a row
- ⇨ Next data packets use decremented data PID
- ⇨ Therefore, last data packet always has data PID DATA0
- ⇨ Sequence starts again each micro frame

High BW Interrupt IN/OUT Transfer



- Uses „normal“ DATA1/DATA0 toggle and retry on error or NAK
- Continuous data toggling on good transactions across micro-frame boundaries
- No retry during a microframe if endpoint responds with NAK
- Host may retry transaction during a microframe due to time-out

USB 2.0

Flow-Control using PING and NYET-Token

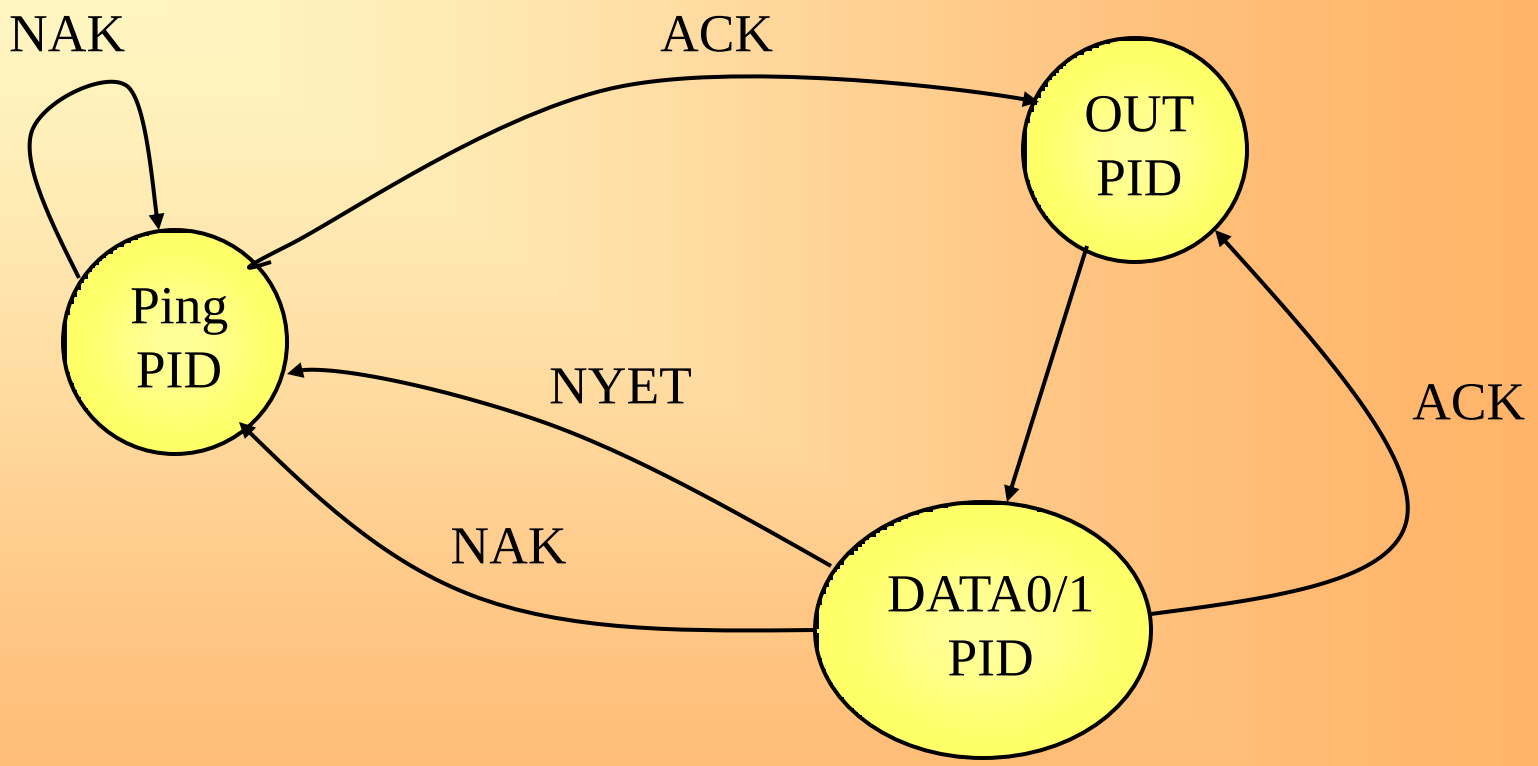
Why we need flow-control ?

- ⇨ High bus utilization due to NAK'ing devices during Bulk-OUT-Transfers
- ⇨ NAK is used and necessary for bus-level flow control
- ⇨ Applicable to Bulk-OUT and Control-Out-Transfers (not for SETUP-token)

OUT	DATA0	ACK
OUT	DATA1	NAK
OUT	DATA1	NAK
OUT	DATA1	ACK
OUT	DATA0	NAK
OUT	DATA0	NAK
OUT	DATA0	ACK

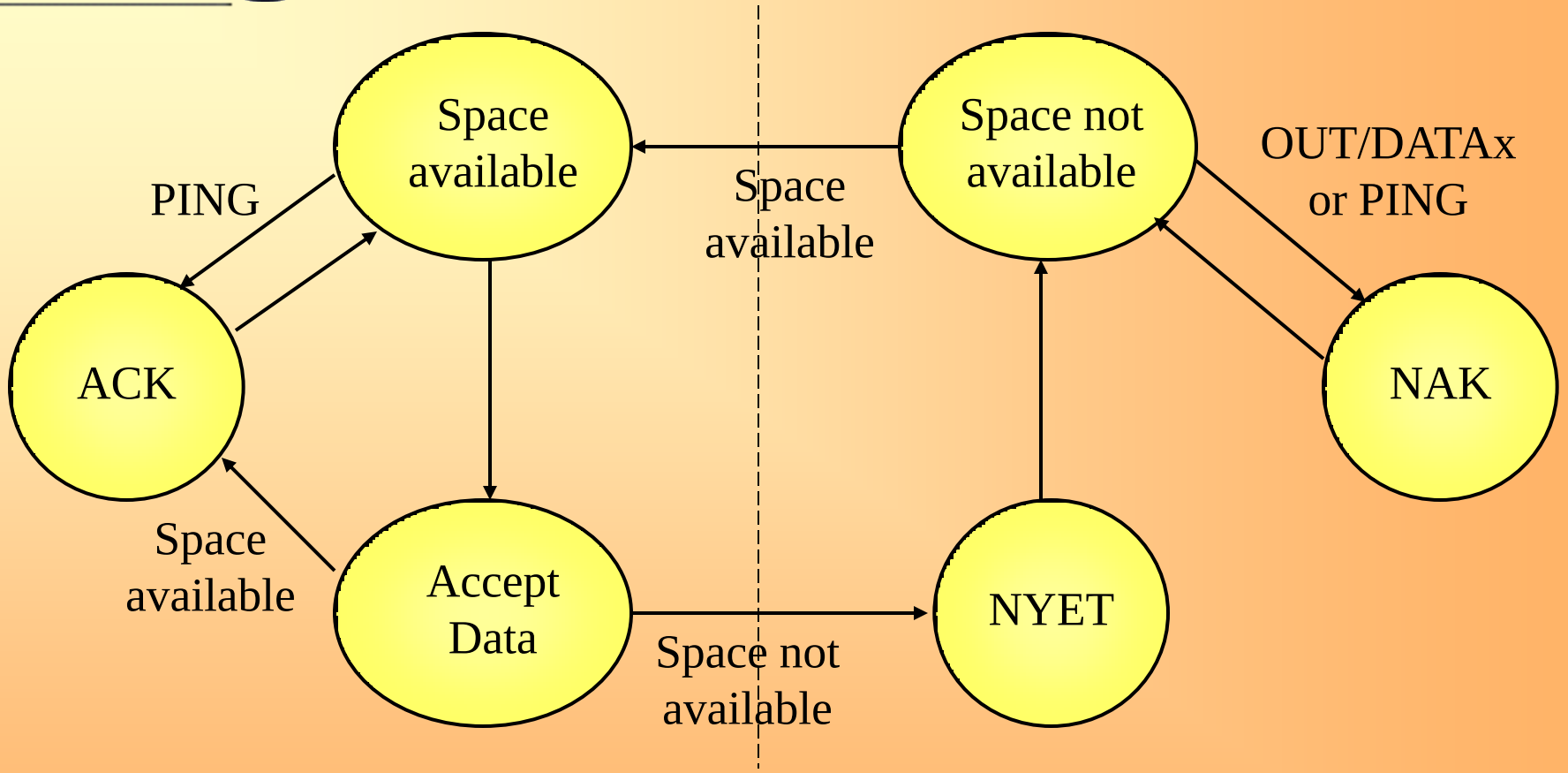
Typical Bulk-OUT-Transfer under USB 1.1

Flow-Control (Host State Machine)



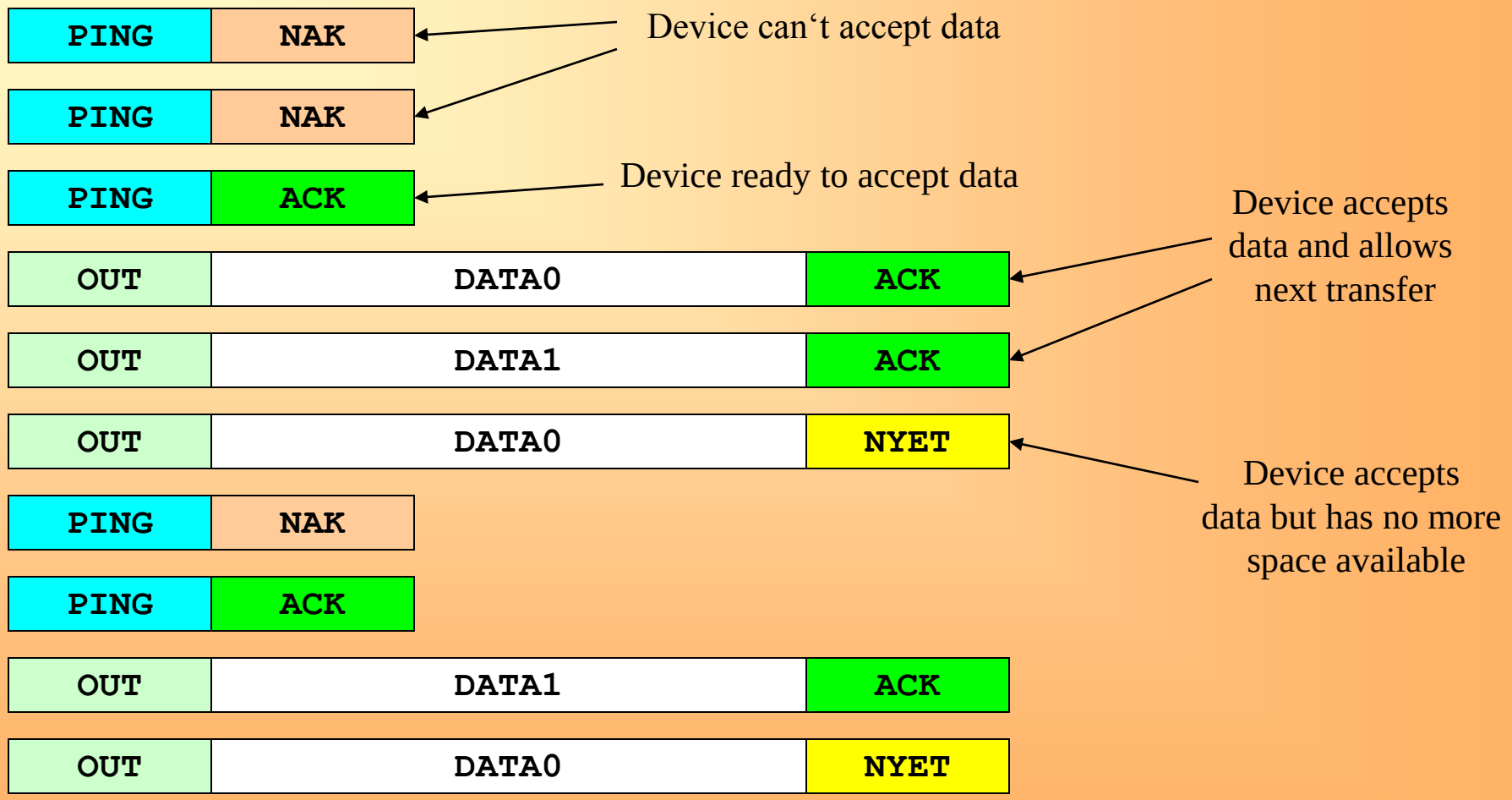
⇨ Initial state is „PING“

Flow-Control (Device State Machine)



- ⇨ ACK if have endpoint has space for at least MaxPacketSize
- ⇨ NYET if endpoint had space but have no further space
- ⇨ NAK if endpoint has no space

Example


PING
NAK
Device can‘t accept data
PING
NAK
PING
ACK
Device ready to accept data
OUT
DATA0
ACK
Device accepts data and allows next transfer
OUT
DATA1
ACK
OUT
DATA0
NYET
Device accepts data but has no more space available
PING
NAK
PING
ACK
OUT
DATA1
ACK
OUT
DATA0
NYET

USB 2.0

Split-Transactions

Focus of Split-Transactions

- Isolation of High-Speed-Traffic between 2.0-compatible Host and Hub from Full-/Low-Speed-Traffic to 1.1-Devices
- Defined only for Communication between HS-Host and HS-Hub
- Improvement of overall bus throughput and utilization
- New Split-Token defined (can only be send by host)
- Start-Split-Token used to tell the hub to initiate a full/low speed transaction
- Complete-Split-Token used to ask the hub for results of previous full/low speed transaction

Split-Token

SYNC	SPLIT	Hub-Addr	SC	Port	S	E	ET	CRC5	Field
32	8	7	1	7	1	1	2	5	Size

SPLIT-PID = 0x2D

Address of the hub
hosting the FS/LS-device

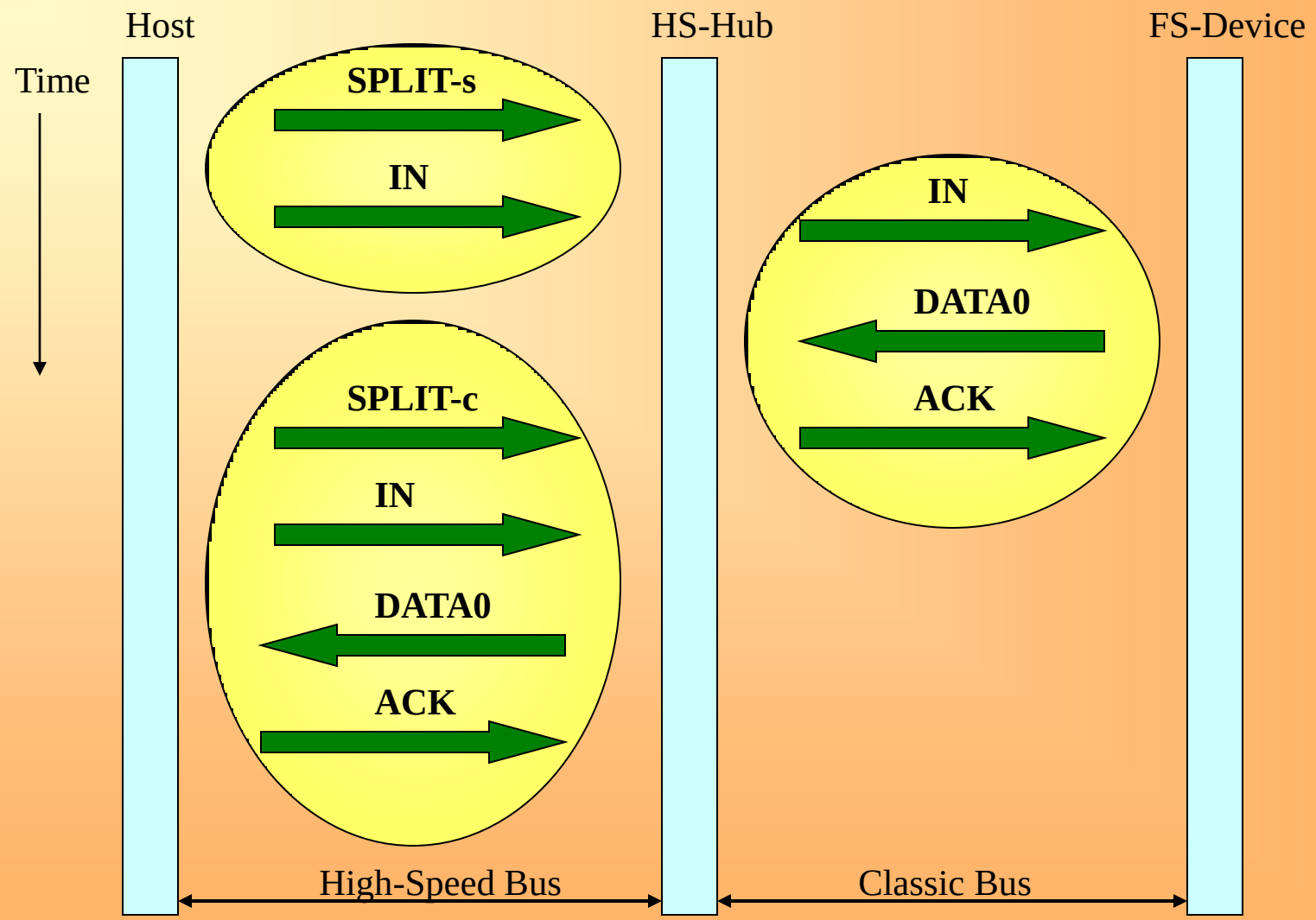
Start/Complete-Field
0 = Start-Split (SSPLIT)
1 = Complete-Split (CSPLIT)

Port-Number of the downstream
port hosting the FS/LS-device

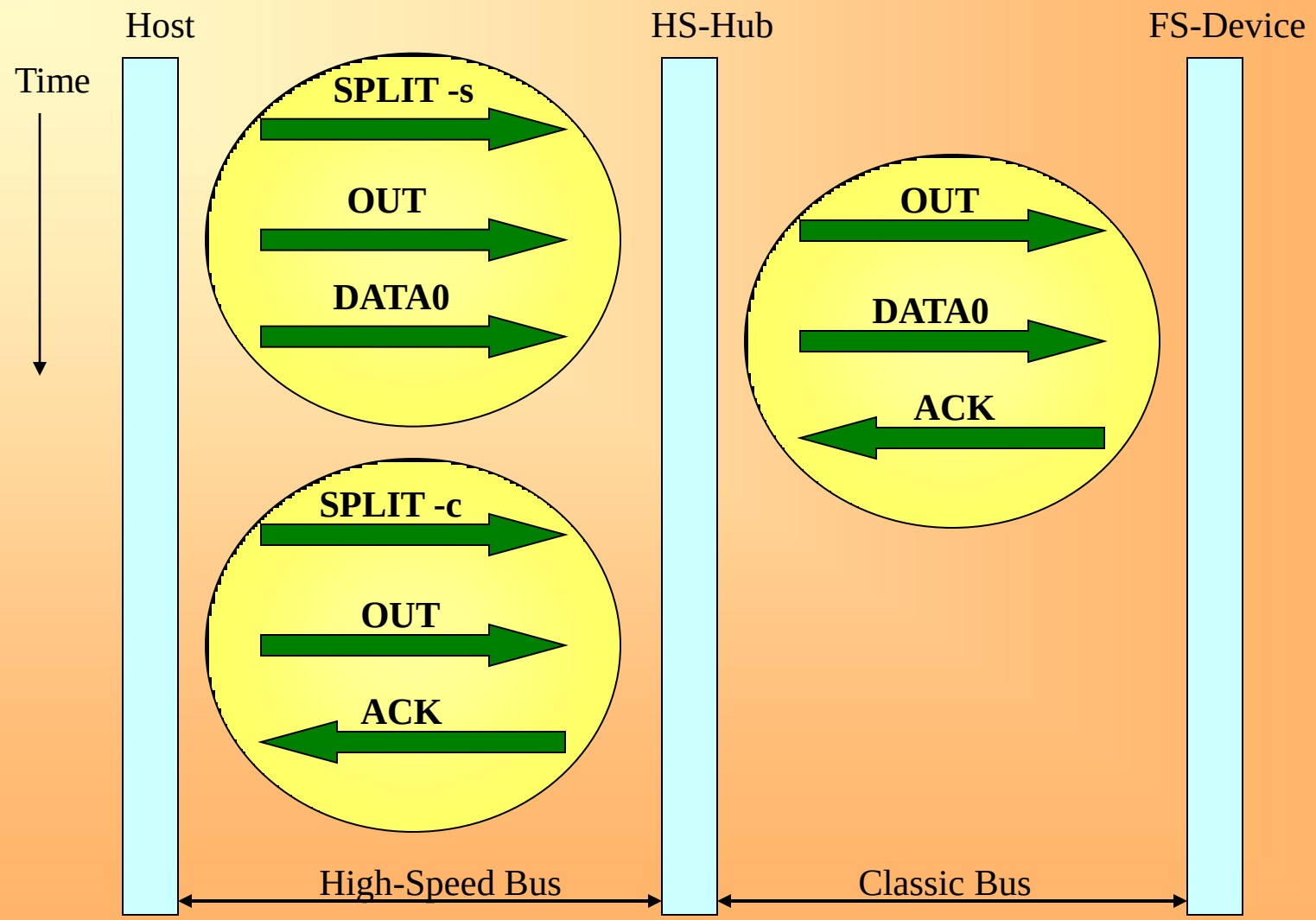
Endpoint-Type:
00 = Control
01 = Isochronous
10 = Bulk
11 = Interrupt

High-Speed to Full-Speed Data Relation
00 = HS-Data is middle of FS-Data
01 = HS-Data is end of FS-Data
10 = HS-Data is beginning of FS-Data
11 = HS-Data is all of FS-Data

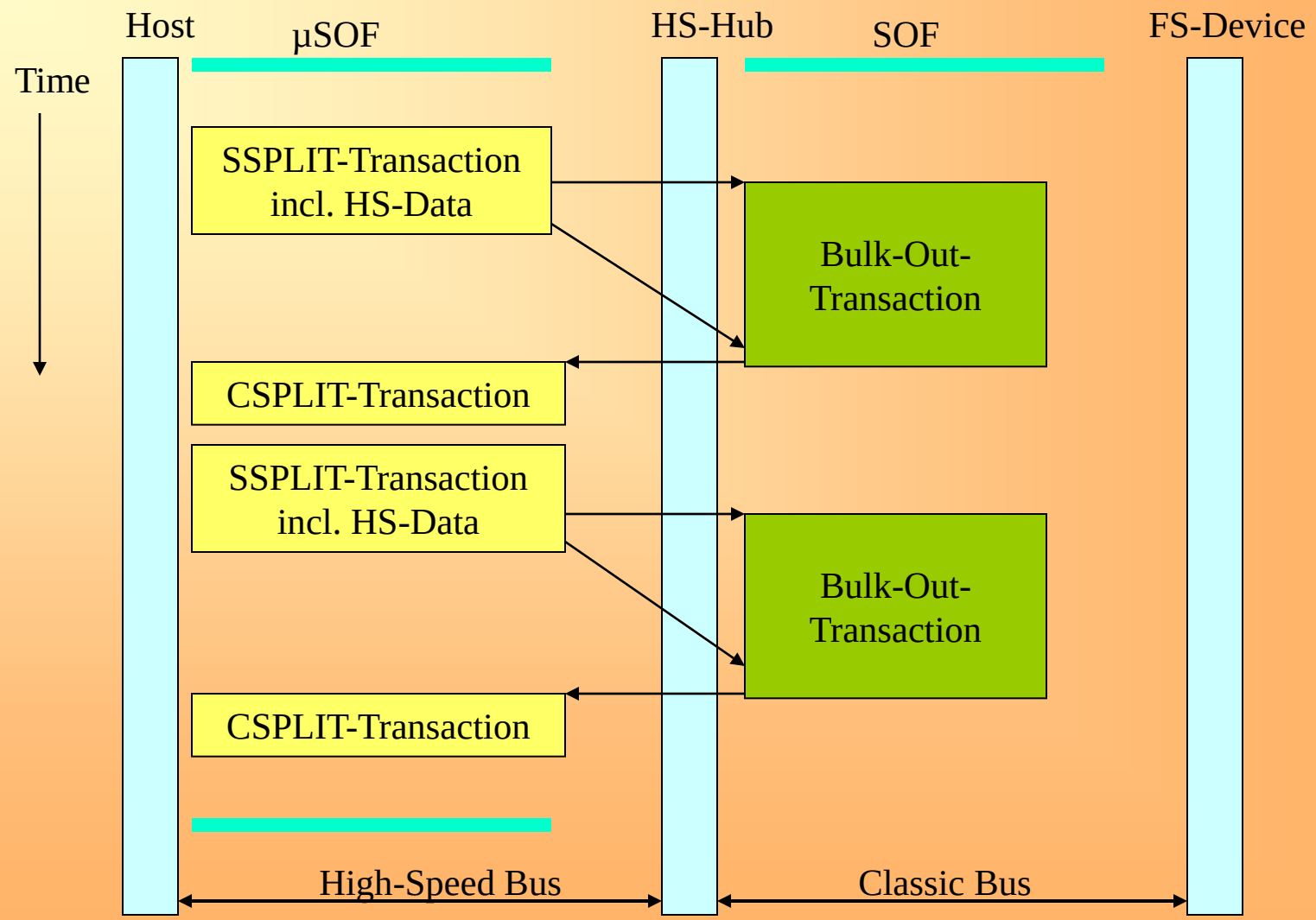
Split-IN-Transaction



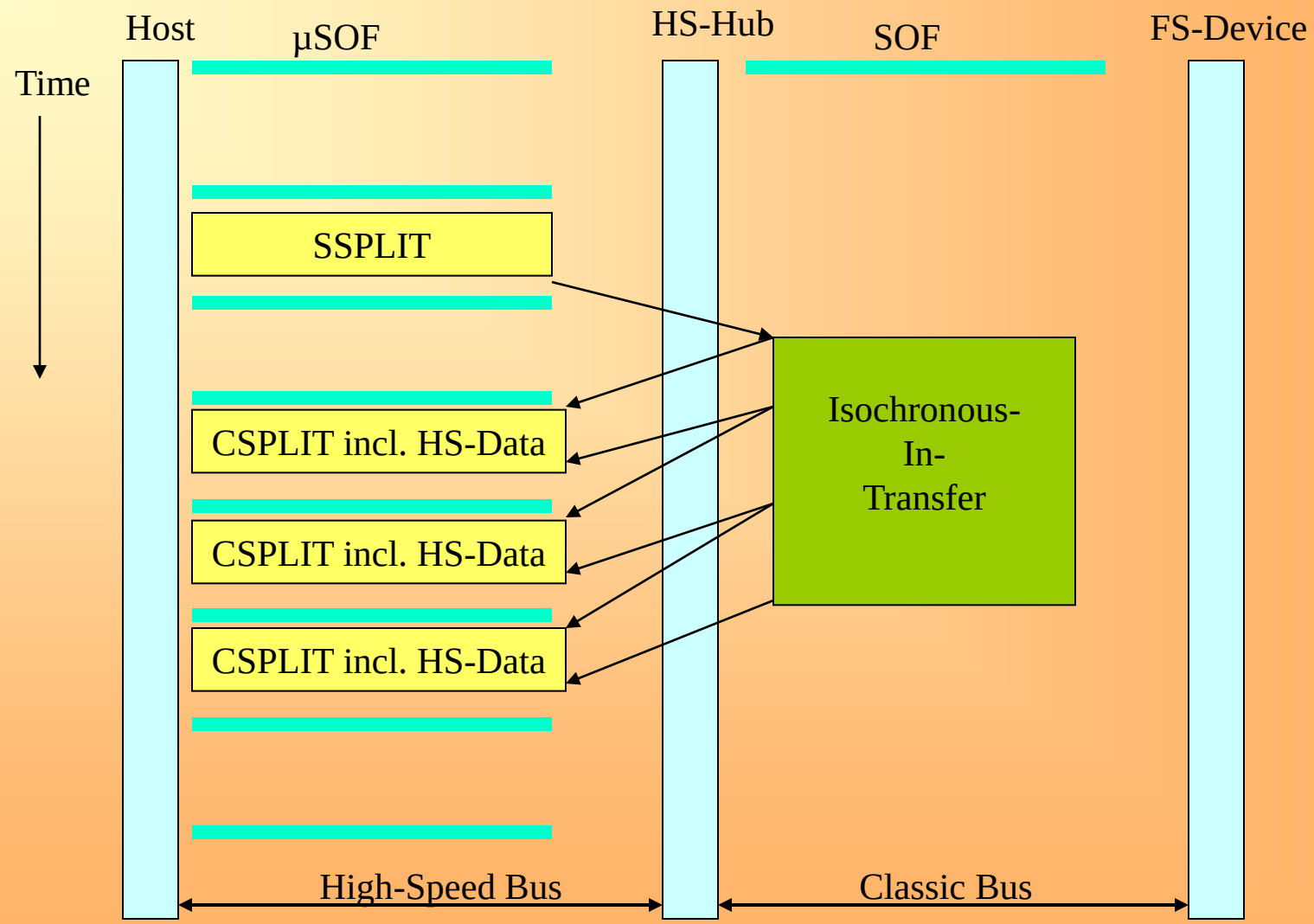
Split-OUT-Transaction



Bulk-OUT-Transfer



Isochronous-IN-Transfer



USB 2.0

Additional Descriptors for High Speed

Why needing additional Descriptors ?

- HS capable devices must work in FS, too!
  - Must not confuse „old“ Full-Speed-Host-drivers
  - Normal descriptors returned for current speed
- High-speed-capable devices can report descriptors for „other_speed“
- HS-info when in FS-port
  - Allows OS to assist end user to choose the correct HS-port
- FS-info when in HS-port
  - What happens if connected to USB 1.1 port ?

Device_Qualifier_Descriptor

Offset	Field	Length	Type	Description	Example
0	bLength	1	Number	Size of this descriptor in byte	0Bh
1	bDescriptorType	1	Constant	Device Qualifier Type = 06 h	06h
2	bcdUSB	2	BCD	USB-Spec. Version Number 2.00	00, 02h
4	bDeviceClass	1	Class	Class-Code	00h
5	bDeviceSubclass	1	Subclass	Subclass-Code	00h
6	bDeviceProtocol	1	Protocol	Protocol-Code	00h
7	wMaxPacketSize0	2	Number	Max. Packet Size of EP0 for other speed (64 Byte)	40, 00h
9	bNumConfigs	1	Number	Number of Other_Speed_Configurations	01h
10	bReserved	1	Constant	Reserved for future use, must be zero	00h

- ⇨ bcdUSB must be set 2.00 (or higher in the future)
- ⇨ wMaxPacketSize0 must be set to 64

Other Speed Configuration Descriptor

Offset	Field	Length	Type	Description	Example
0	bLength	1	Number	Size of this descriptor in byte	09h
1	bDescriptorType	1	Constant	Other Speed Configuration Descriptor-Type = 07 h	07h
2	wTotalLength	2	Number	Total length of all descriptors for this configuration	22, 00h
4	bNumInterfaces	1	Number	Number of supported interfaces	02h
5	bConfigValue	1	Number	Value to activate this configuration	01h
6	iConfiguration	1	Index	Index of string descriptor 'Configuration"	04h
7	bmAttributes	1	Bitmap	Configurations attribute * D7 = Bus Powered * D5 = Remote-Wakeup D6 = Self Powered D4..D0 = Reserved	A0h
8	bMaxPower	1	Number	Power consumption in 2mA steps (250mA/2mA = 125)	7Dh

- ⇨ This descriptor reports a speed dependent configuration for the other speed
- ⇨ Same concatenation-mechanism for interface/class/endpoint-descriptors used as for standard configuration-descriptor

Endpoint-Descriptor

Offset	Field	Length	Type	Description	Example
0	bLength	1	Number	Size of this descriptor in byte	07h
1	bDescriptorType	1	Constant	Endpoint-Descriptor-Type = 05 h	05h
2	bEndpoint Address	1	Number	Endpoint-Address: D0..D3 = Endpoint-Number D7 = Direction: 0 = OUT-Endpoint, 1 = IN-Endpoint	81h
3	bmAttributes	1	Bitmap	Supported transfer type of this endpoint Bit 0..1: 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt	03h
4	bMaxPacketSize	2	Number	Maximum packet size of this endpoint = 1024*2 byte	00,0Ch
5	bInterval	1	Number	Polling Interval of this Endpoint = 8 x 125 µs	04h

Changed Encoding of EP-Descriptor

- No change in data structure - only encoding / interpretation changes
- Larger high speed packet sizes encoded by packet size x number of transactions per micro-frame
- HS Packet Size = wMaxPacketSize[10..0] x (wMaxPacketSize[12..11] +1)
- Polling intervals for HS periodic transfers encoded in micro-frames
- Endpoint period = $2^{\text{bInterval} - 1}$ (bInterval must be between 1..16)

USB 2.0

Additional Request for High-Speed Test Mode

High-Speed-Test-Mode

- Allows standardized testing of HS transceivers
  - Upstream ports of all HS-capable devices
  - Downstream ports of Hubs (also Root-Hub in Host Controller !)

- Available Test-Modes
  - Test_J (enter HS J state - test of high output drive level on D+ line)
  - Test_K (enter HS K state - test of high output drive level on D- line)
  - Test_SE0_NAK (HS receive mode - test of low level output voltage)
  - Test_Packet (Sending a test packet - test of rise/fall times, eye patterns, jitter)
  - Test_Force_Enable (set downstream ports into HS-mode, test of disconnect)

- Enter test mode using SET_FEATURE (TEST_MODE) request
- Exit test mode via power cycle (devices) or reset (hubs)

SET_FEATURE (TEST_MODE)

- Standard-Device-Request SET_FEATURE supports for HS-Mode additional Feature Selector „TEST_MODE“

- SET_FEATURE command format:
  - bmRequestType = 00000000b Standard Request to Device
  - bRequest = 0x03 SET_FEATURE
  - wValue = 0x02 Feature-Selector = TEST_MODE
  - wIndex = 0x01 TEST_J
    0x02 TEST_K
    0x03 TEST_SE0_NAK
    0x02 TEST_PACKET
    0x05 TEST_FORCE_ENABLE
  - wLength = 0x0000

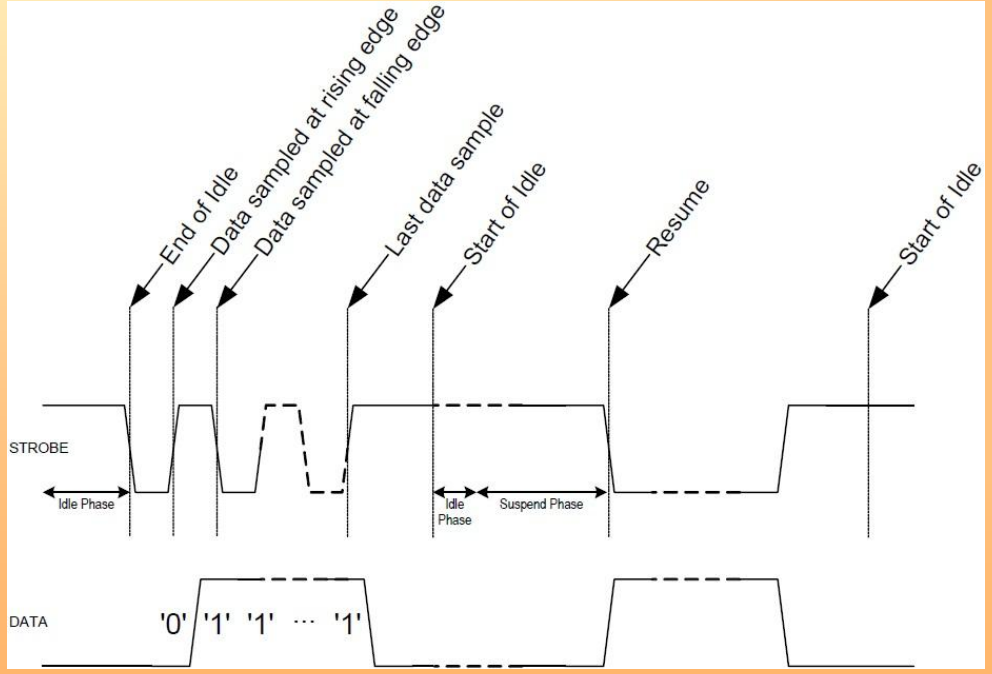
USB HSIC Basics

USB HSIC Key Features

- Bringing USB capability to multi-chip assembly
  - Improve embedded communication performance
  - Usage of existing robust protocol layer
  - Usage of existing protocols/device classes
  - Just exchanging the physical layer
- Defining requirements for low-level protocol
  - Low power with 1.2 V LVCMO signaling levels
  - 2-signal synchronous serial interface using 240 MHz DDR signaling
  - 10 cm max. cable length
  - No speed detection, the interface defaults to USB 2.0 High-Speed operation

HSIC Physical Layer

- Strobe (clock signal)
- Data (data line, sampled on rising and falling edge of Strobe)



HSIC Signal States

Signal State	Strobe	Data	Description
IDLE	1	0	1 or more Strobe-periods
SUSPEND	1	0	> 3 ms inactivity
CONNECT	0	1	2 Strobe-periods
RESUME	0	1	> 20 ms
RESET	0	0	> 2,5 us

..and the Story continues

- Intel informed at the Intel Developer Forum at the 18th of Sept. 2007 about a founded USB 3.0 Promoter Group
- Goal was creation of „SuperSpeed“ interconnect standard, which is meanwhile available. SuperSpeed devices are already at the market.
- 10 times faster than high-speed (~5 Gbit/s)
- Backward compatible
- Optimized for low power and „improved protocol efficiency“
- Optical capability
- SuperSpeed Inter-Chip Specification
- Specification available at www.usb.org