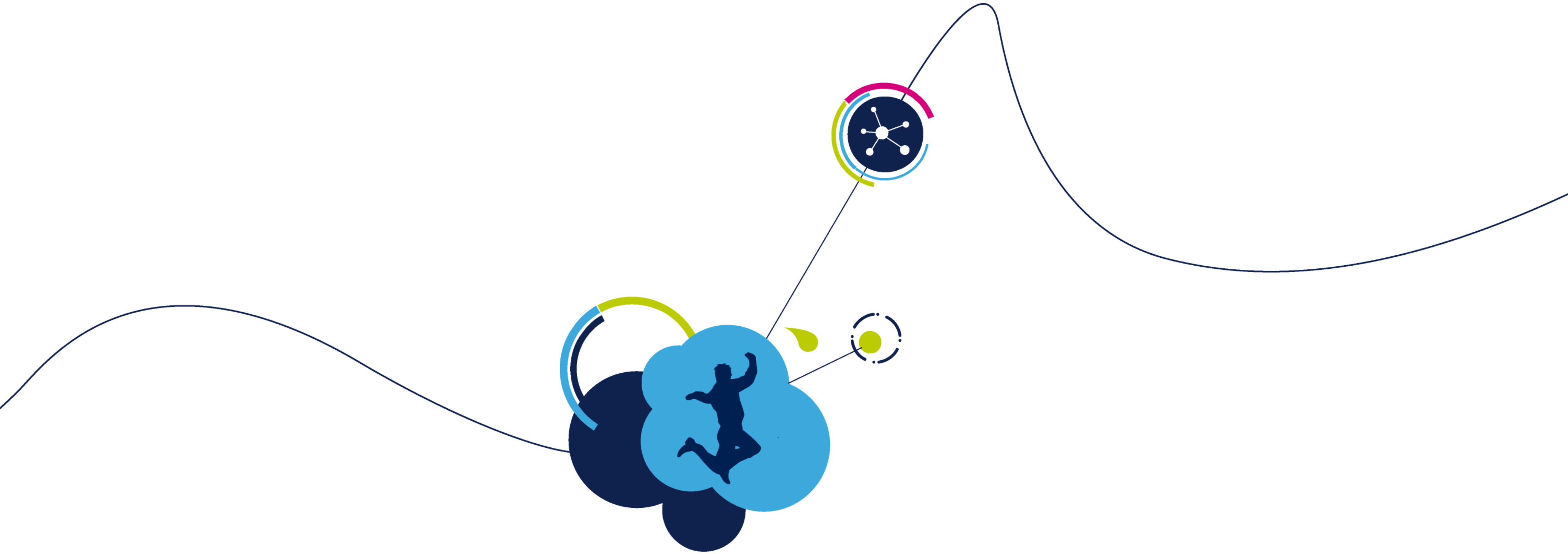
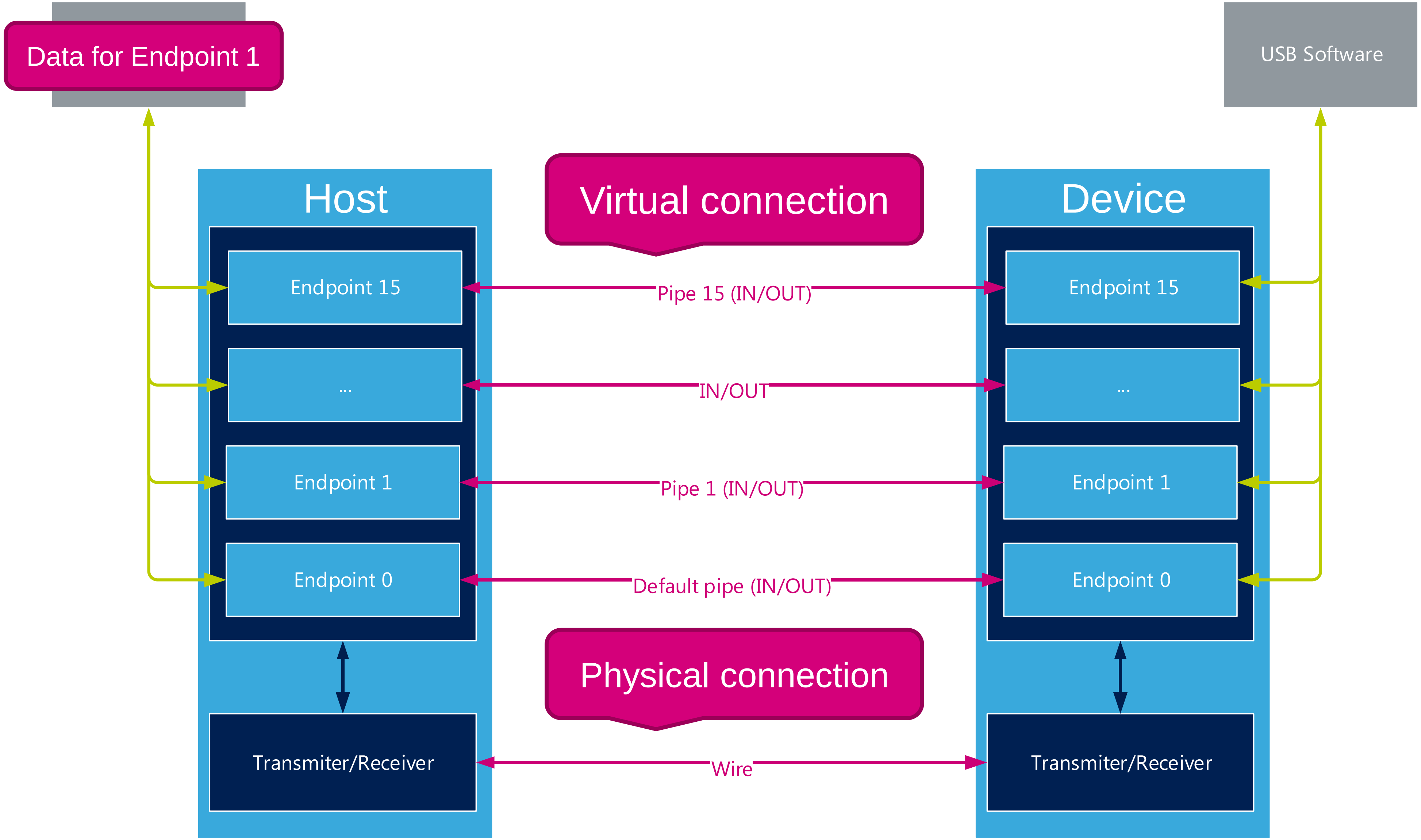


Type of USB transfers

life.augmented

Pipes & endpoints

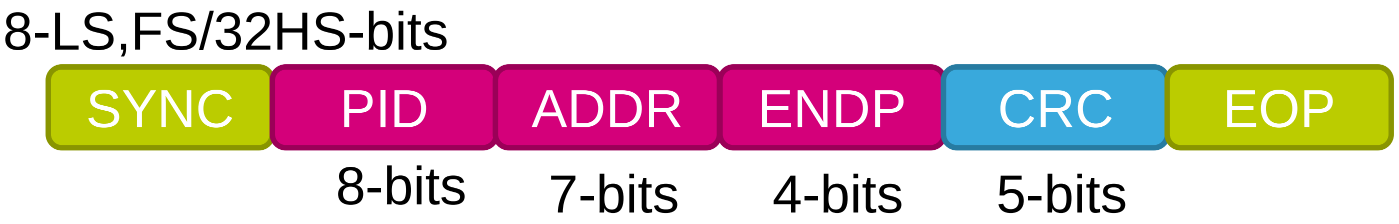


Pipes & endpoints

- Each device/host has multiple endpoints
  - Endpoint is virtual communication channel between device and host
  - Each endpoints can have different type of USB transaction
    - SETUP
    - BULK
    - Interrupt
    - Isochronous
  - Endpoint 0 is reserved for enumeration & configuration of the USB device
  - Max. 16 Endpoints (bidirectional), depends on HW
- USB pipe is connection from host to specific endpoint on specific device
  - It consists of endpoint number and device address
  - It is used from host perspective

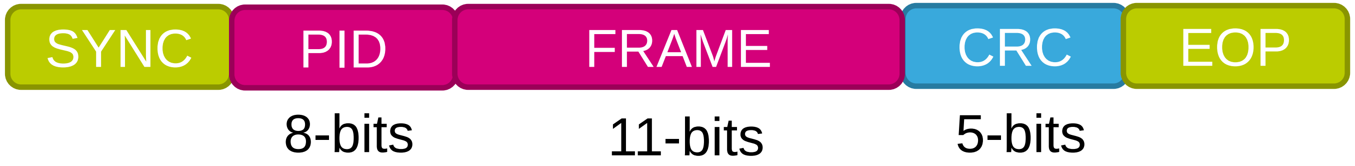
Packets & Structure 1/2

- Token packets



SYNC	PID	ADDR	ENDP	CRC	EOP
8-LS,FS/32HS-bits	8-bits	7-bits	4-bits	5-bits	

  - SYNC – synchronization field
  - PID – Packet Field Formats – first 4 bits show packet type, last 4 bits are mirrored check
  - ADDR – destination or source (special case) address
  - ENDP – destination endpoint
  - CRC – Cyclic Redundancy check for packet integrity, 5bits for token and SOF, 16bits for data
  - EOP – electrical signal for end of packet

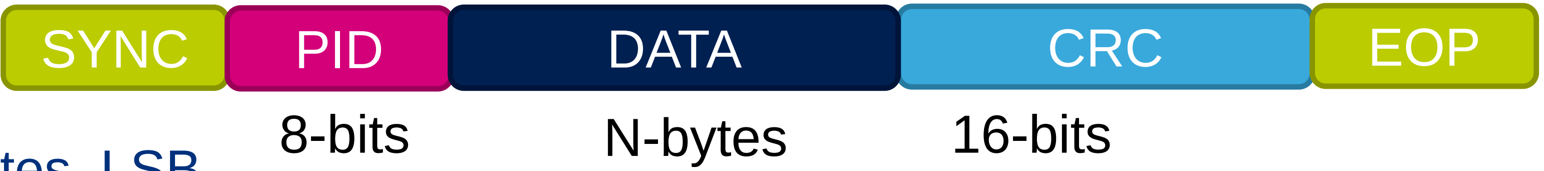


SYNC	PID	FRAME	CRC	EOP
	8-bits	11-bits	5-bits	

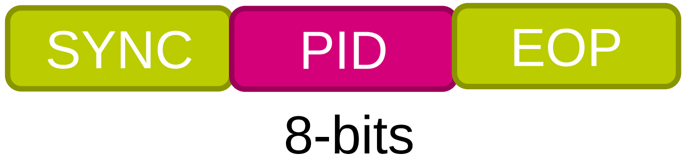
- Start-of-frame packet
  - Frame – incremented value send during each (micro)frame

Packets & Structure 1/2

- Data packet
  - Data Field – 0 to 1024 bytes, LSB

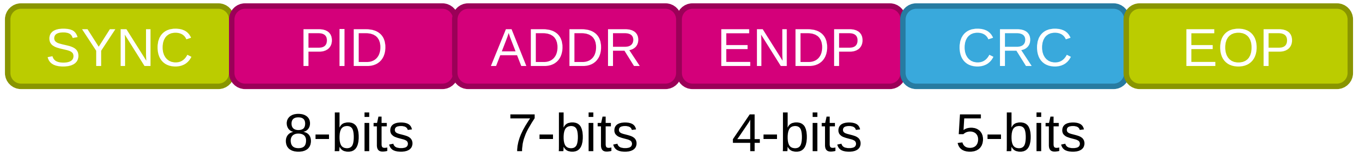


- Acknowledge packet
  - PID contains ACK, NAK, STALL or NYET



Packet & PID 1/2

- Packet



PID type	PID name	PID[3:0]
Token	OUT	0001B
	IN	1001B
	SOF	0101B
	SETUP	1101B
Data	DATA0	0011B
	DATA1	1011B
	DATA2	0111B
	MDATA	1111B

Packet marking start of transaction

Packet marking data part of transaction

Used for high-speed communication

Packet & PID 2/2

- Packet

SYNC	PID	ADDR	ENDP	CRC	EOP
	8-bits	7-bits	4-bits	5-bits	

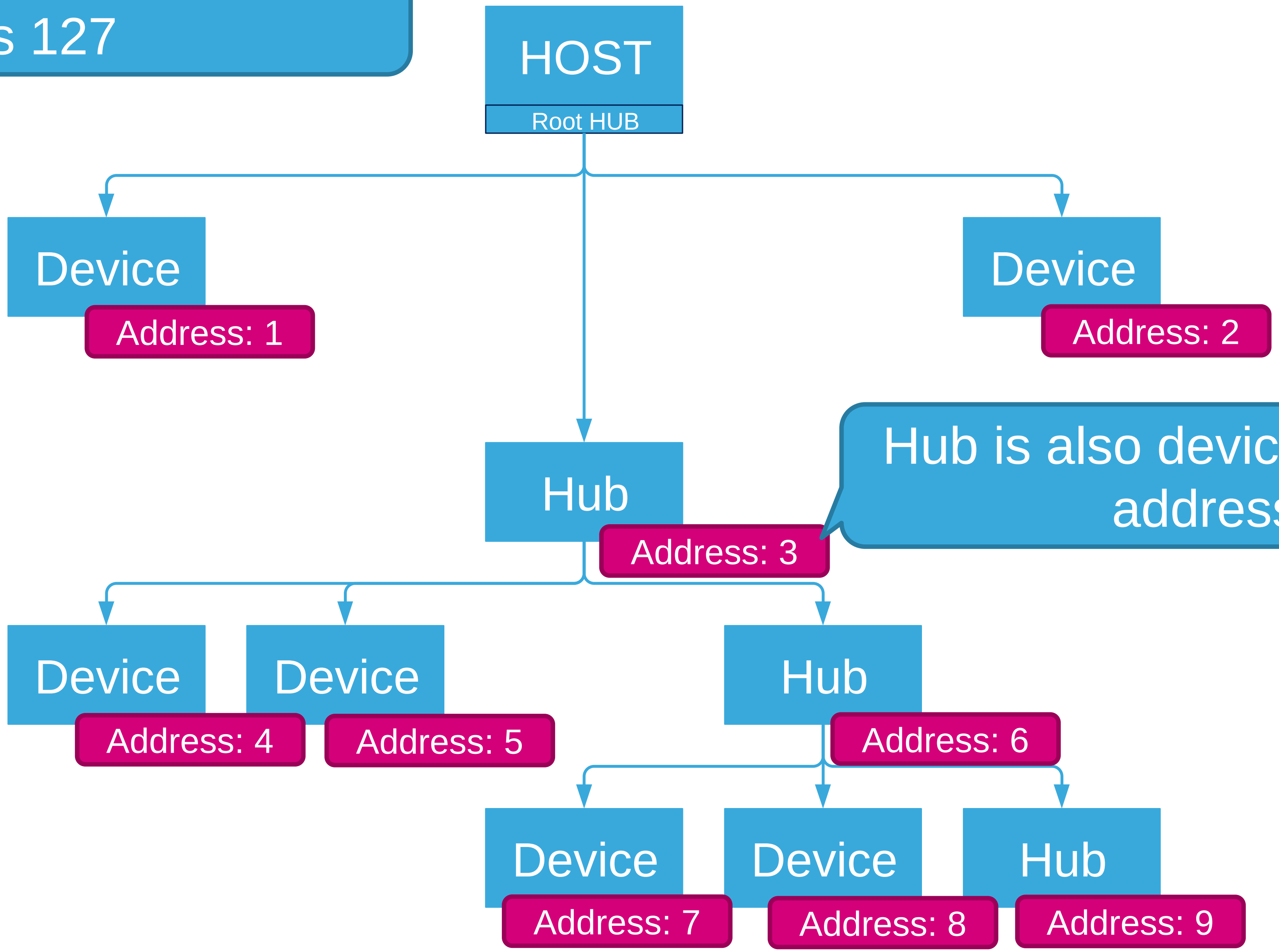
PID type	PID name	PID[3:0]
Handshake	ACK	0010B
	NAK	1010B
	STALL	1110B
	NYET	0110B
Special	PRE	1100B
	ERR	1100B
	SPLIT	1000B
	PING	0100B
	Reserved	0000B

Packet markig end of transaction

Specific packets for HS transactions

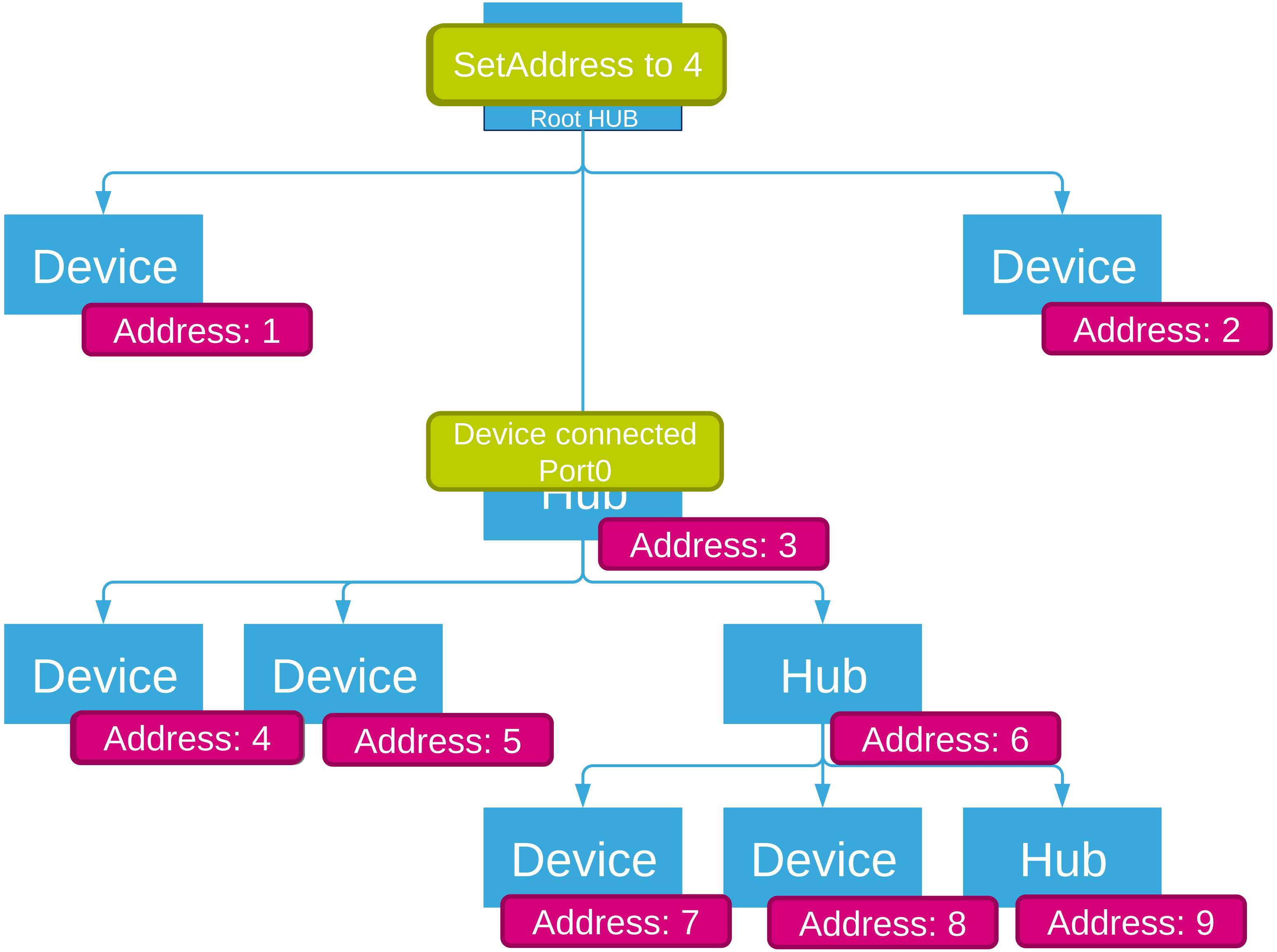
Address

Maximum number of devices is 127

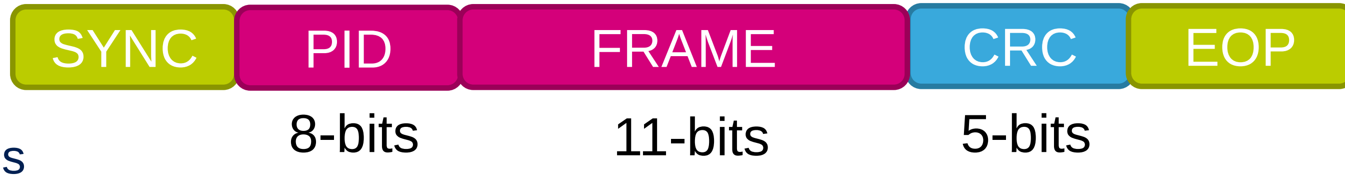


Hub is also device with own address

Address assign

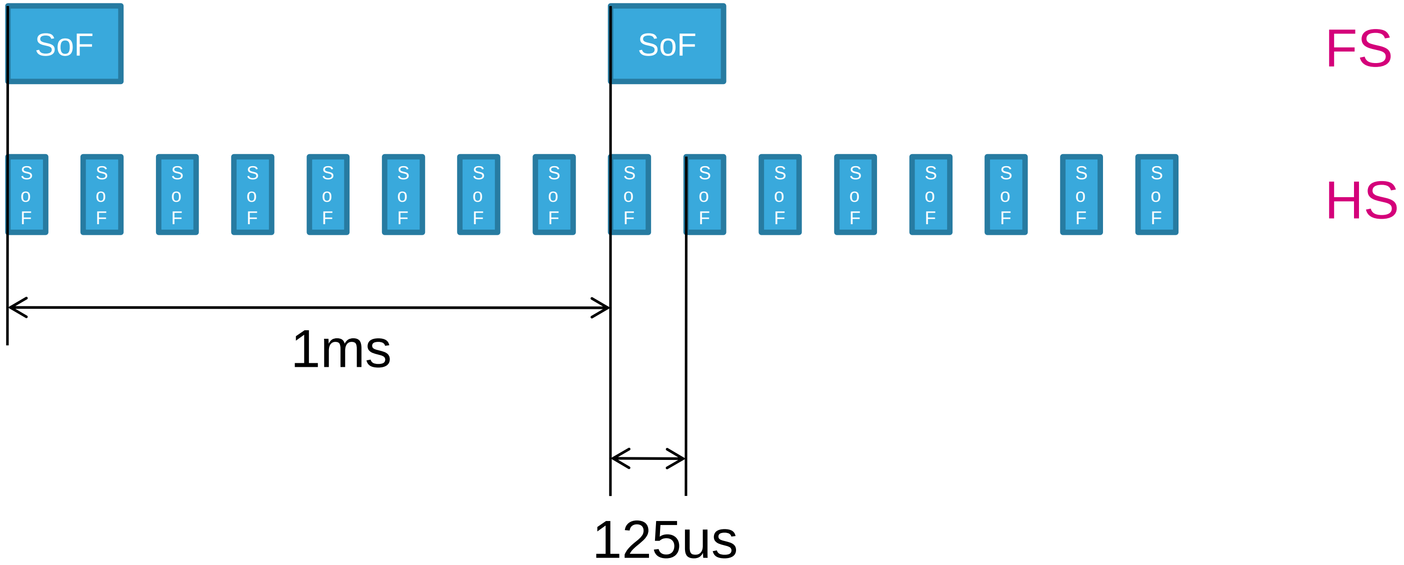


Start of frame


SYNC | PID | FRAME | CRC | EOP

8-bits | 11-bits | 5-bits

- Send in periodical intervals
- 1ms interval for LS/FS and 125us for HS devices
- Base timing reference for USB
  - Used for synchronization (e.g. audio devices)
- Our crystal less devices can use it for synchronization
- Suspend command is detected by no SOFs during 3ms (ex. PC in hybernation)



Transaction

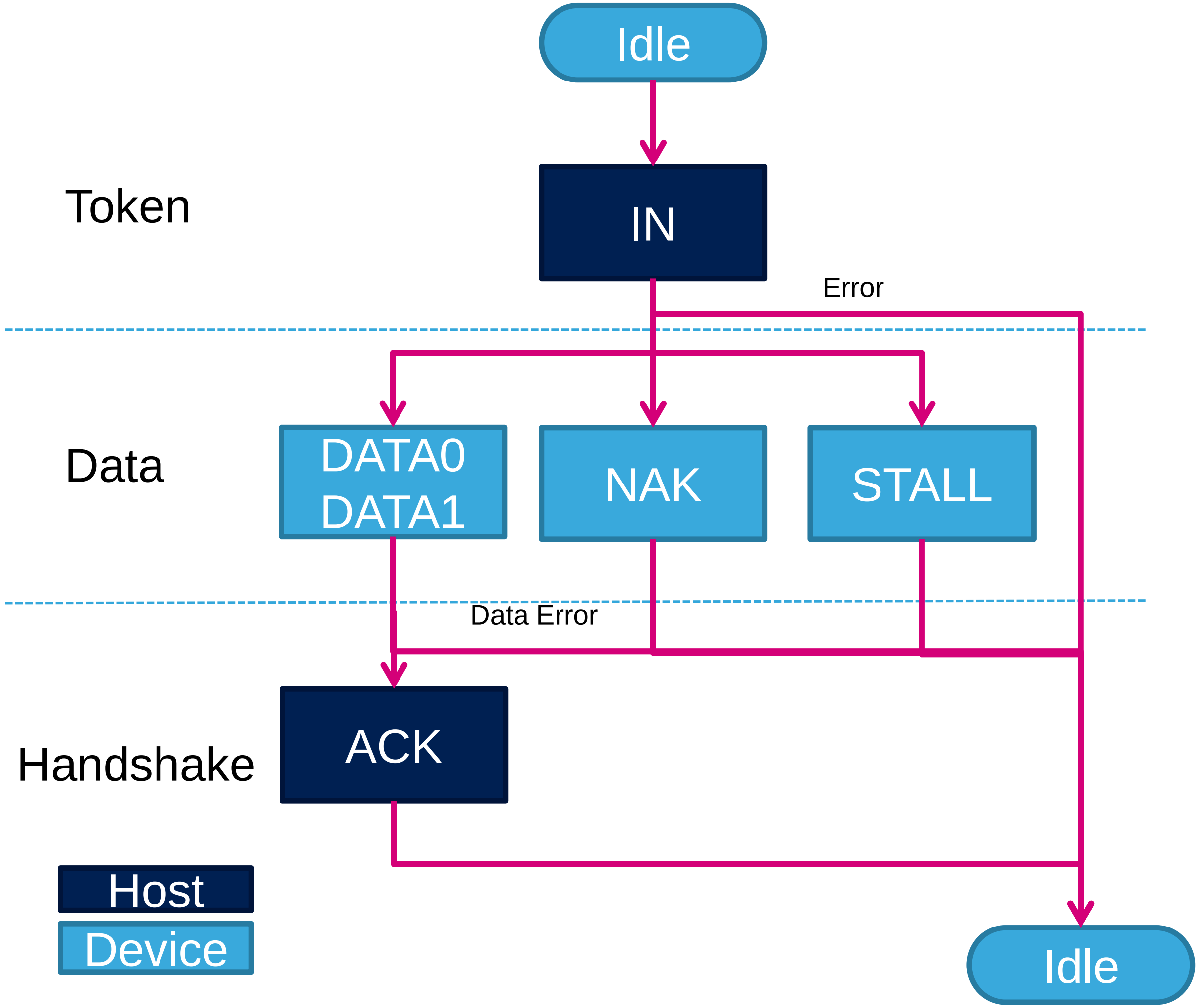
- Transaction is created from packet communication
  - First is send the TOKEN packet (SETUP, OUT, IN etc.)
  - Then DATA packet
  - Transaction ends with Handshake packet (ACK, NAK, STALL, NYET)
- Direction is from host perspective
  - IN is from device to host, OUT is from host to device
- Basic types of transactions:
  - Control Setup transaction (and control sequences)
  - Bulk transaction
  - Isochronous transaction
  - Interrupt transaction

Transaction handshake packets

Handshake	Description
ACK	Receiver accepts error-free data packet
NAK	Receiving device cannot accept data or transmitting device cannot send data
STALL	Endpoint is halted or a control pipe request is not supported
NYET	HS only: No response yet from receiver
Missing handshake	No handshake from receiver, error during transmission

Bulk transaction IN

- Transfer large non critical data
  - E.g. file transfers
- **Only if USB have bandwidth**
- Error handling
- Not in LS



Bulk IN

First correct transaction

oken to
data

Here is example where the device not receive the IN token

Second correct

Device send data to host

Because previous data were incorrect host ask for them

firms data
dgement by
ng ACK

Host wait until his response timeout expires and asks again

After complete transaction the data pid must be changed

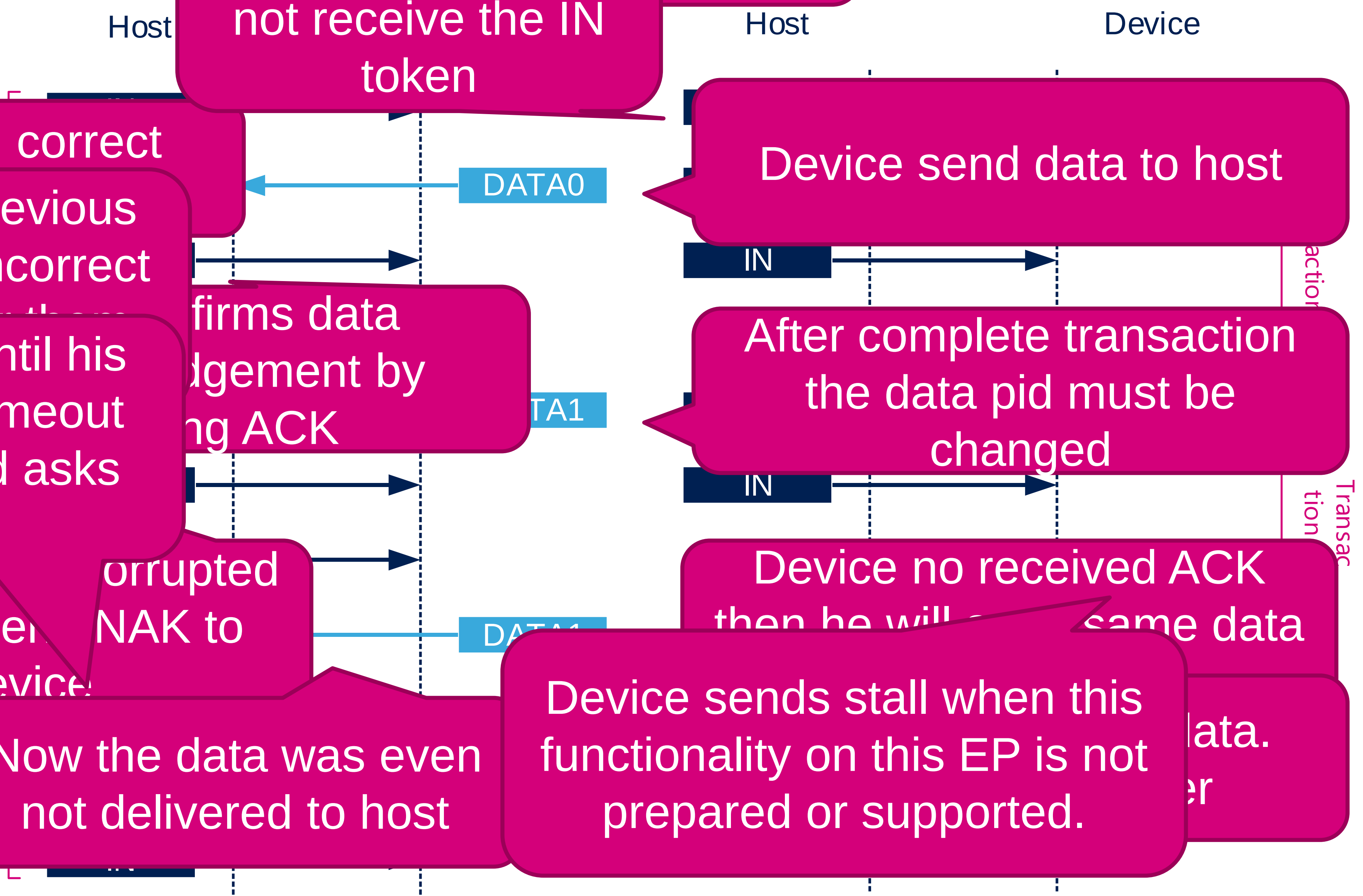
If host rece orrupted data he se NAK to device

Device no received ACK then he will same data

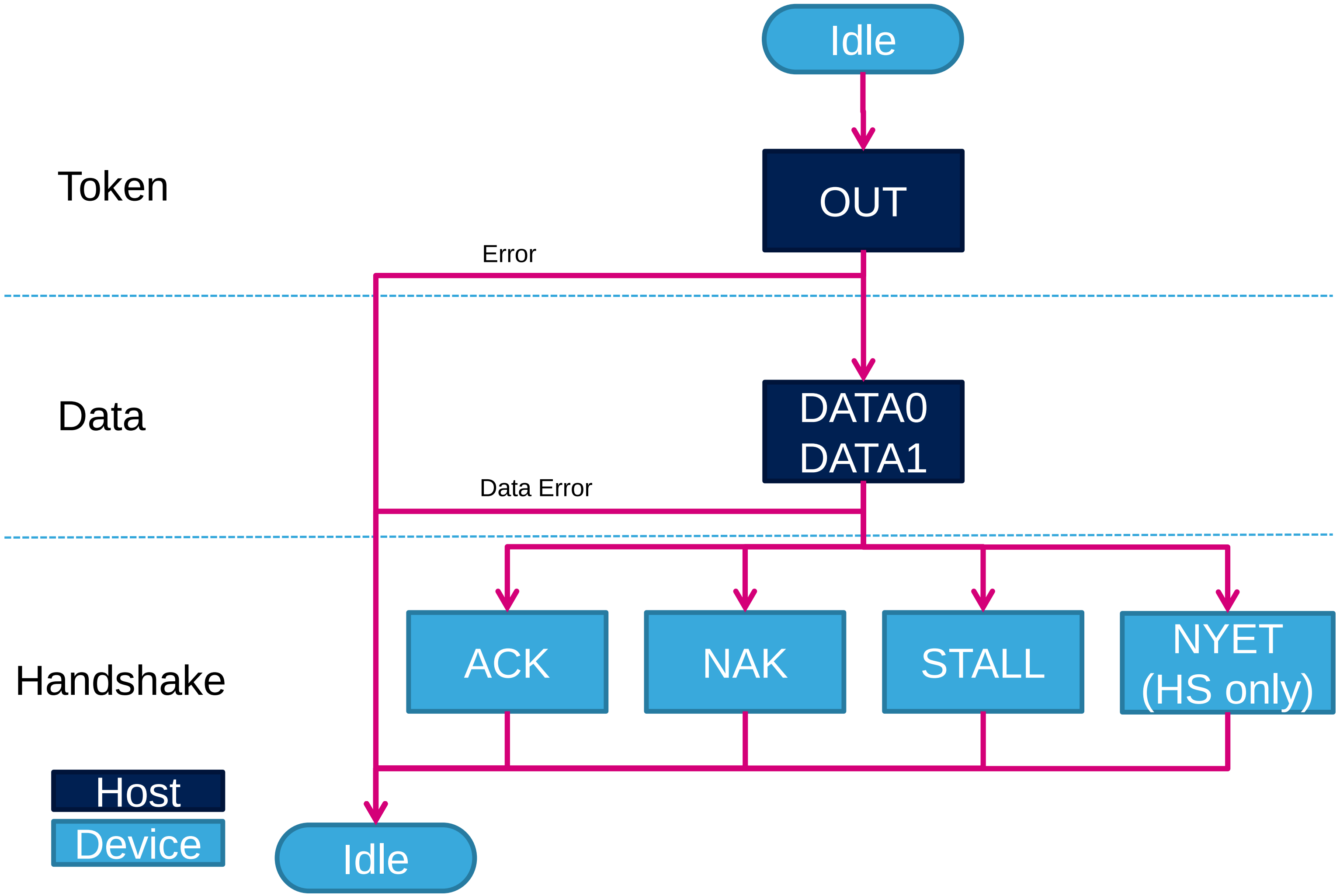
lata.
er

Now the data was even not delivered to host

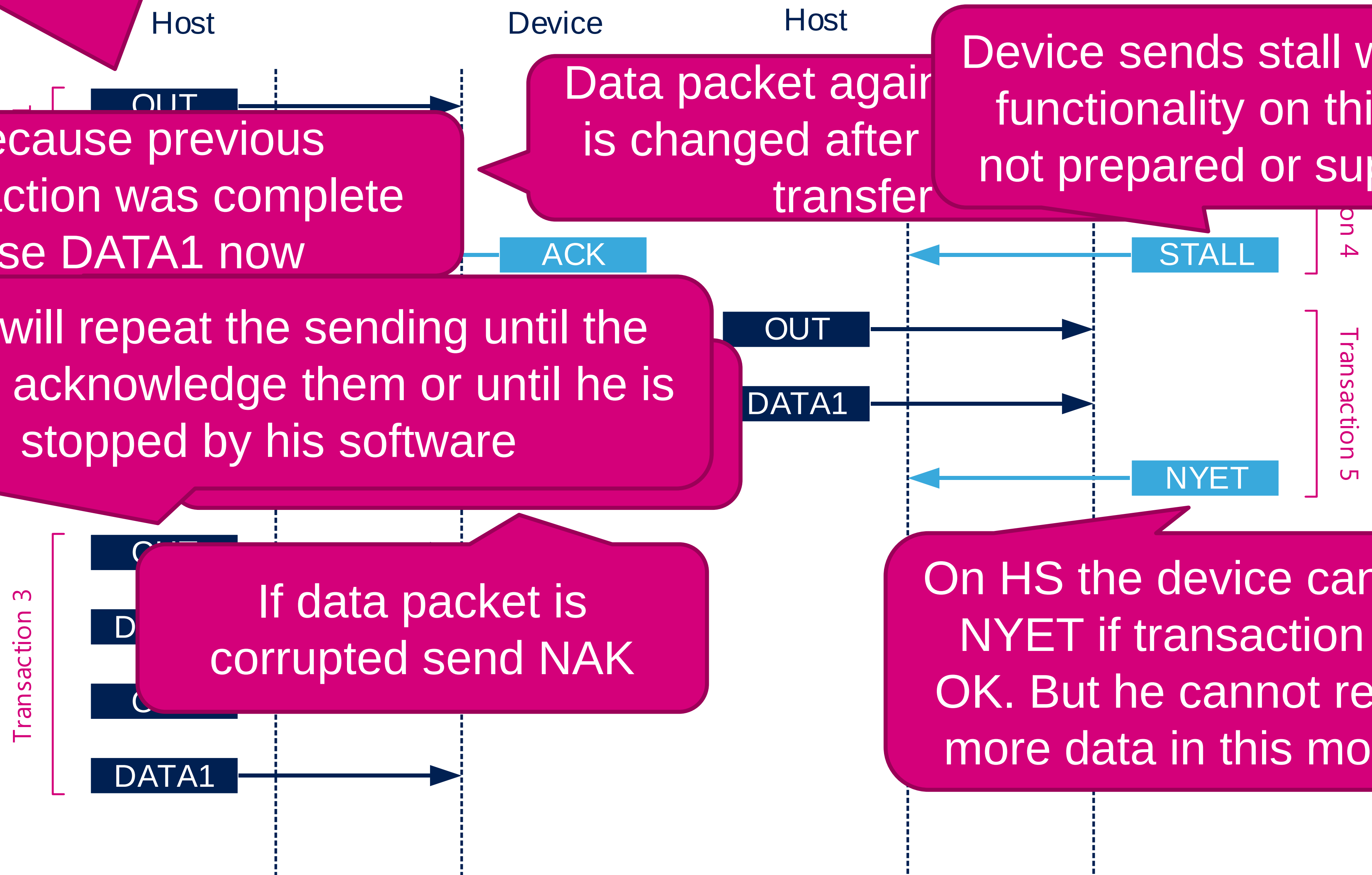
Device sends stall when this functionality on this EP is not prepared or supported.



Bulk transaction OUT



Bulk OUT



Host send OUT token to notify device that next packet contain data

Because previous transaction was complete use DATA1 now

Data packet agair is changed after transfer

Device sends stall when this functionality on this EP is not prepared or supported.

Host will repeat the sending until the device acknowledge them or until he is stopped by his software

If data packet is corrupted send NAK

On HS the device can send NYET if transaction was OK. But he cannot receive more data in this moment

Data acknowledgement and error handling

- Packets integrity is checked with CRC and PID complementary values (invalid packets are ignored)

- When device sends ACK when it has successfully accepted data

- When device is not ready to accept data
  - It sends NAK for low-speed / full-speed
  - It can send NYET for high-speed bulk transfers

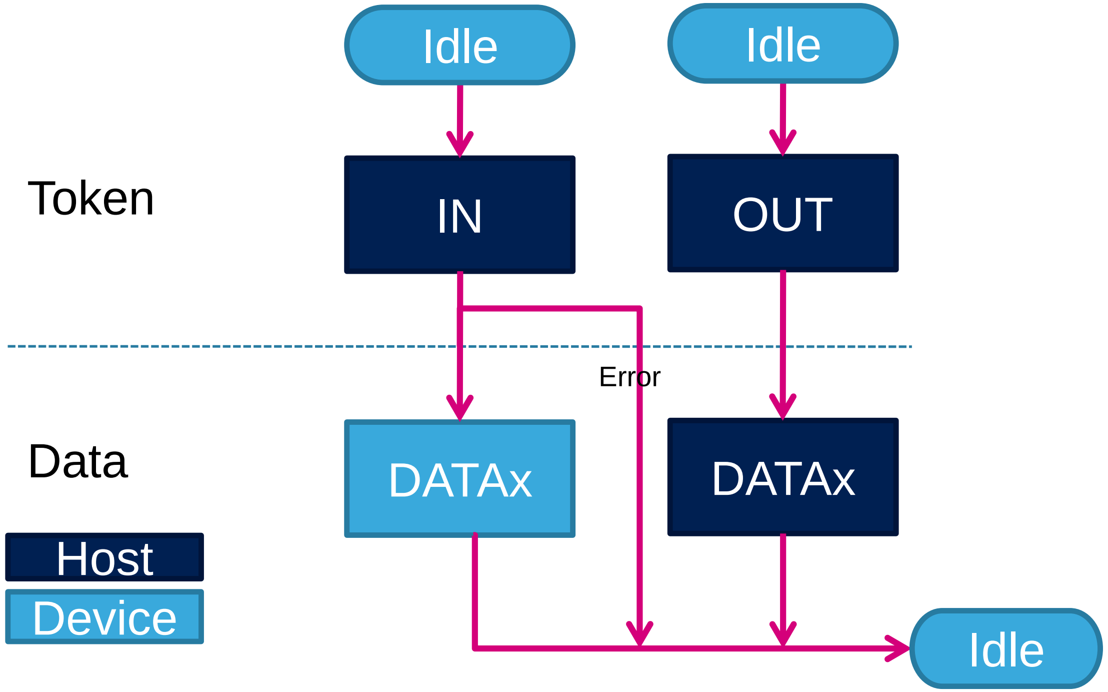
- When device sends STALL it stops the transmission
  - STALL is used for rejecting unsupported USB requests

- When device or host detects error it doesn’t send ACK

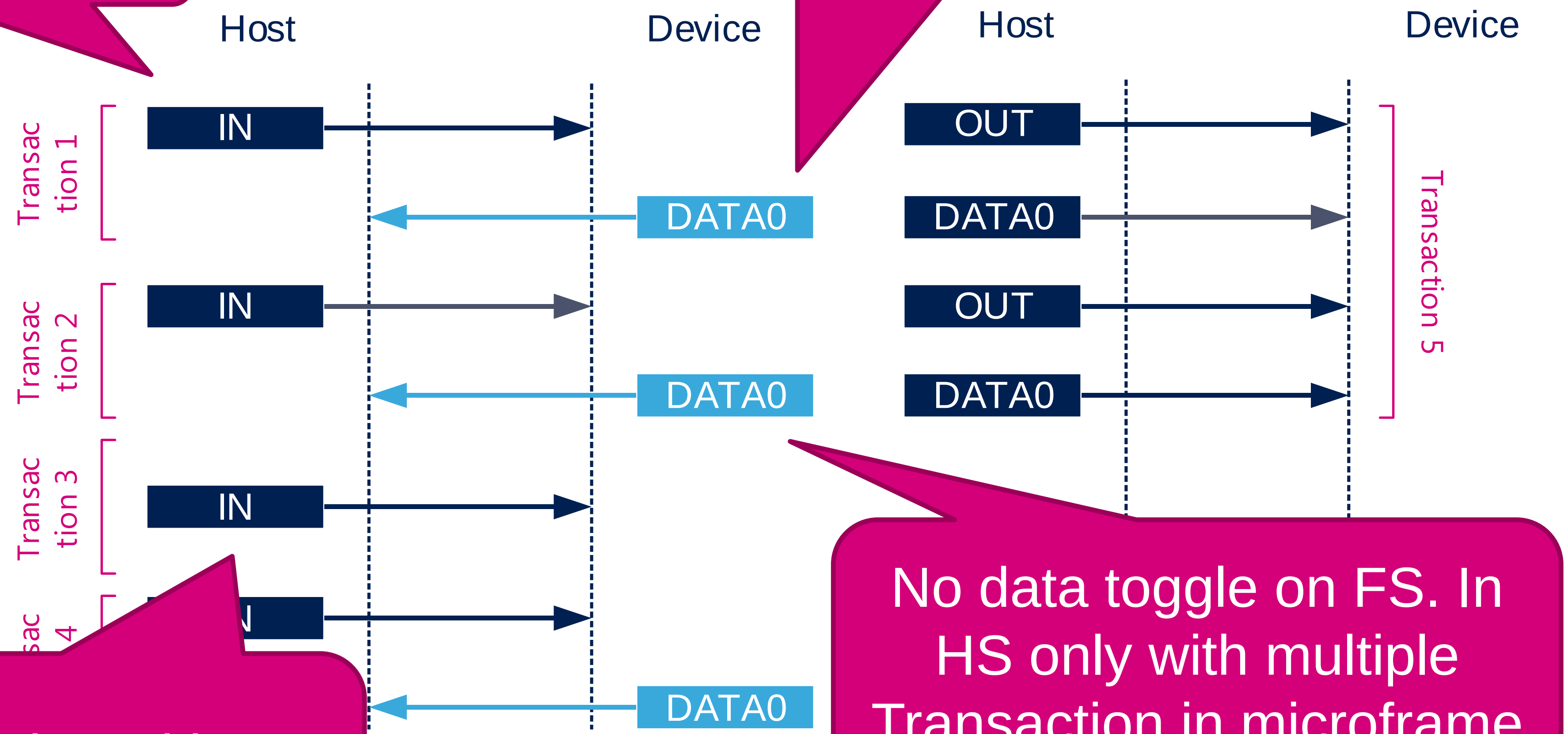
- Successive data transfers toggle the data PID (DATA0, DATA1)
  - When device sends ACK, but host doesn’t receive it (invalid PID), the host will send the same data twice
  - Data toggling can detect this case and discard data sent twice

Isochronous transaction

- For real-time applications
  - E.g. audio playback
- Provide data on timely basis
  - One packet per frame (FS) – DATA0
  - Up to 3 packets per microframe (HS)
- No error handling
  - Errors just detected
- Not in LS



Isochronous IN/OUT



Host send IN to request new data

Device respond with data

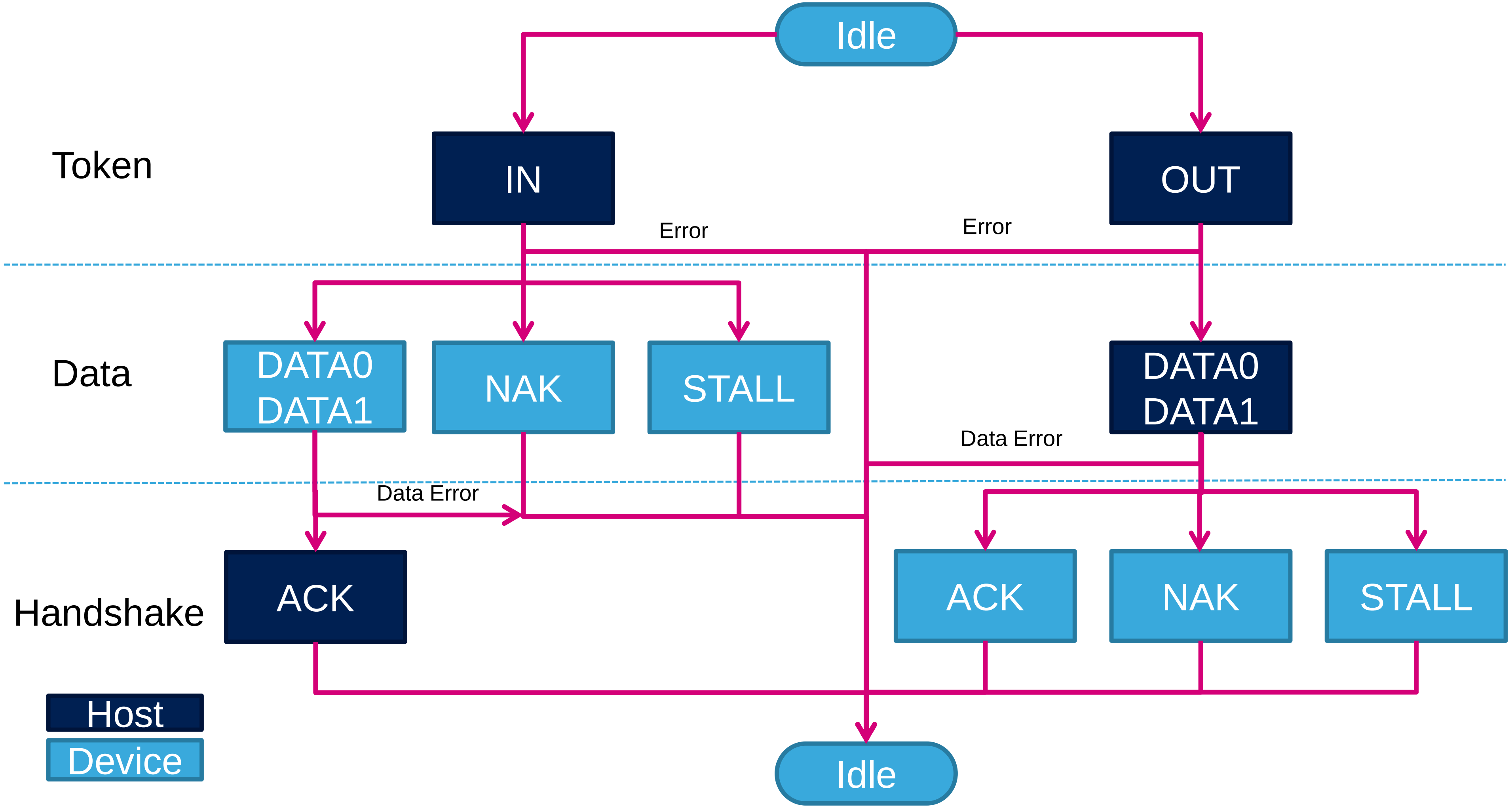
No data toggle on FS. In HS only with multiple Transaction in microframe

Previous packet without answer, data are lost

Interrupt transaction

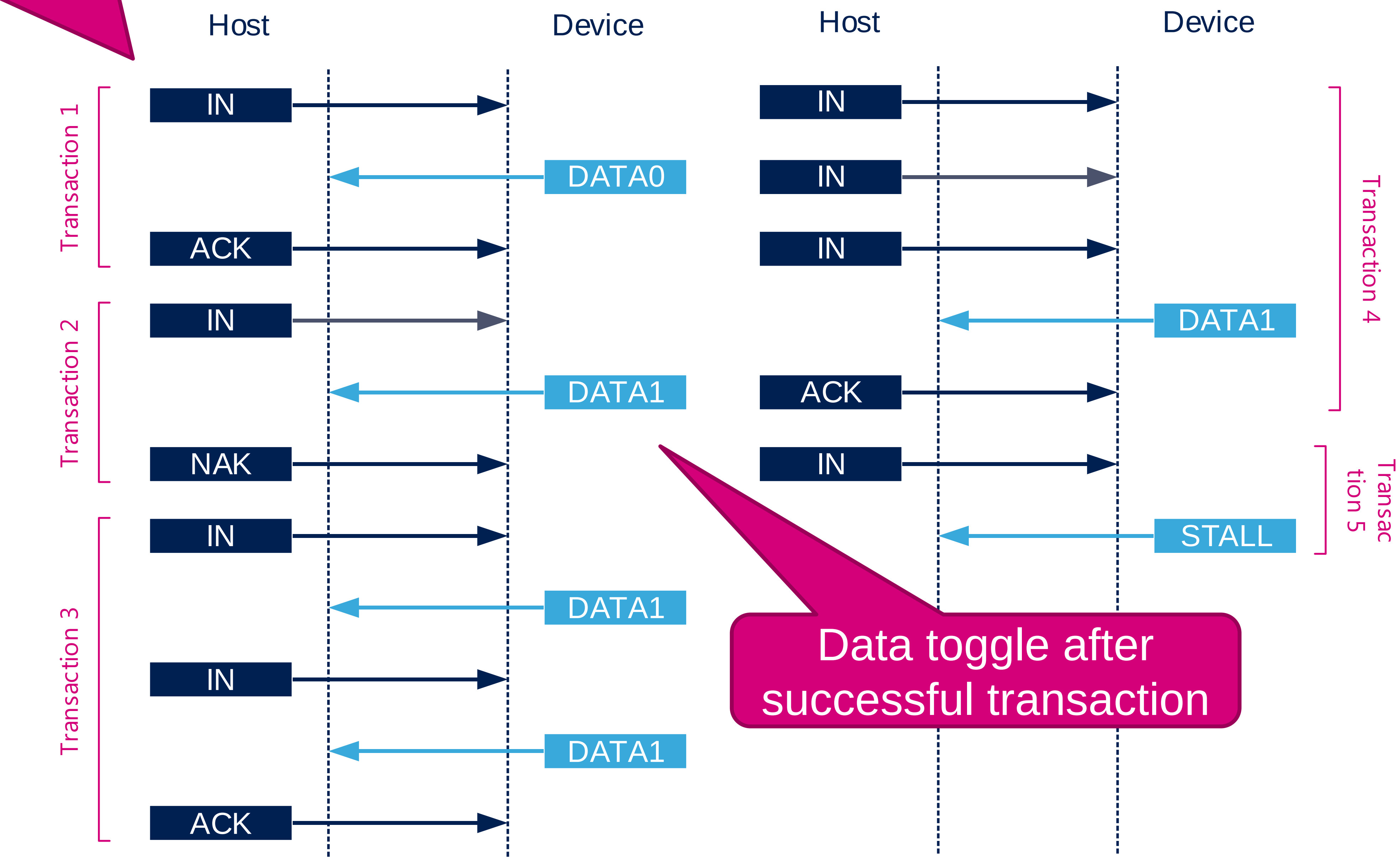
- For interrupt driven devices (mouse, keyboard, …)
- Data read periodically
- Limited packet size
- Pool interval in steps from 1ms
- Max 90% of 1ms bandwidth reserved to Isochronous and Interrupt transfers
- Available on all USB speeds (LS/FS/HS)
- Error detection and correction

Interrupt transaction



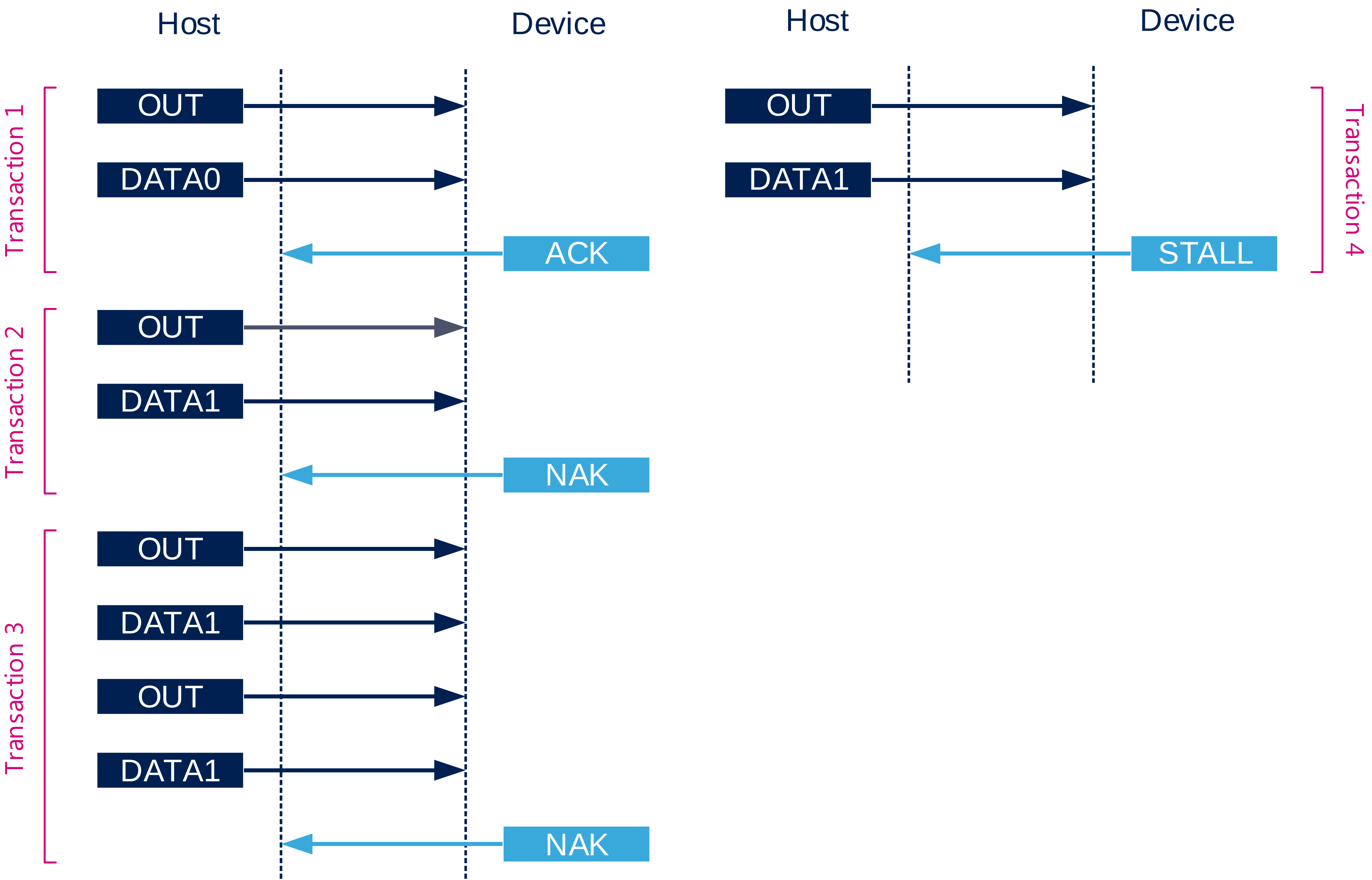
Interrupt IN

Very similar to Bulk transactions



Data toggle after successful transaction

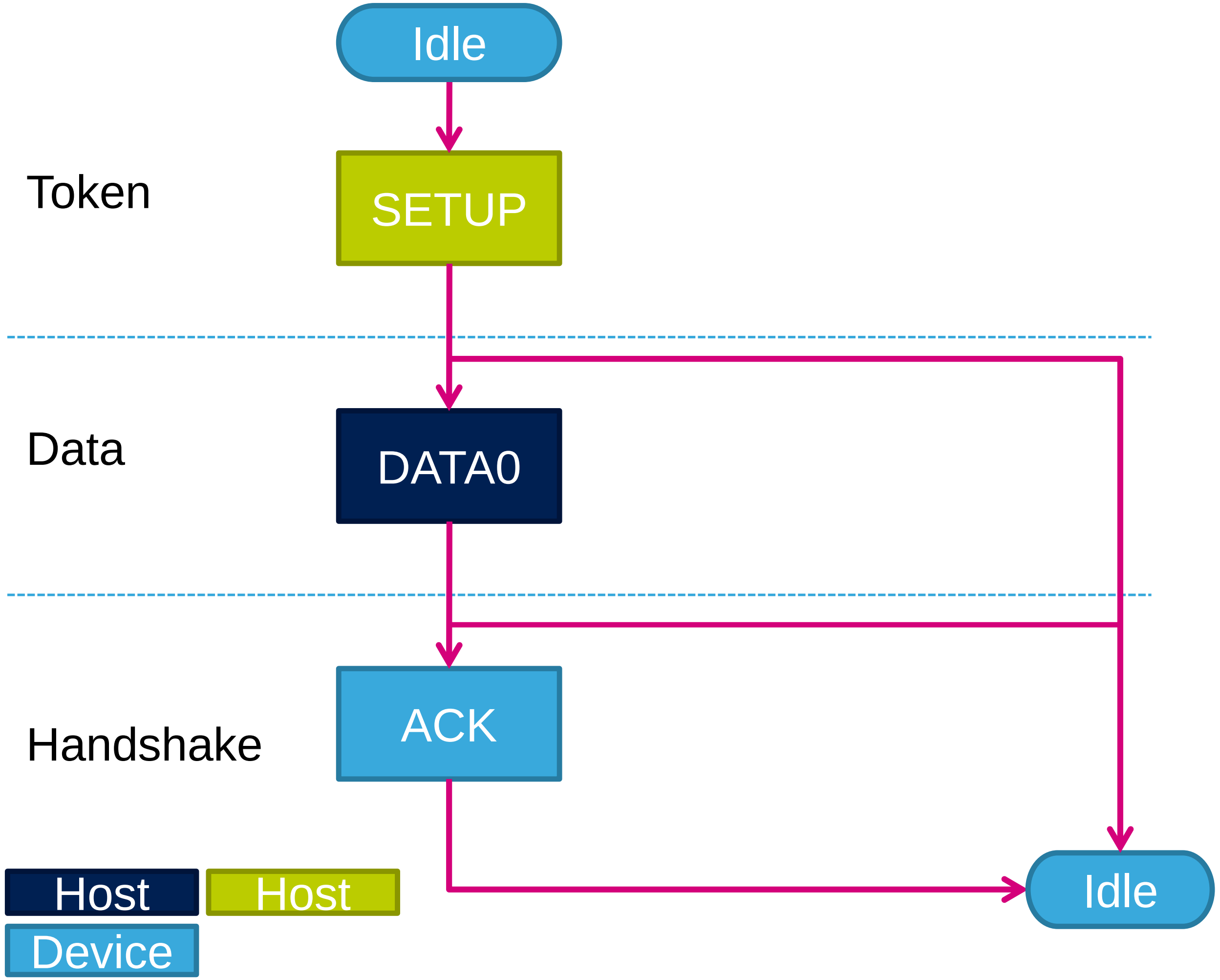
Interrupt OUT



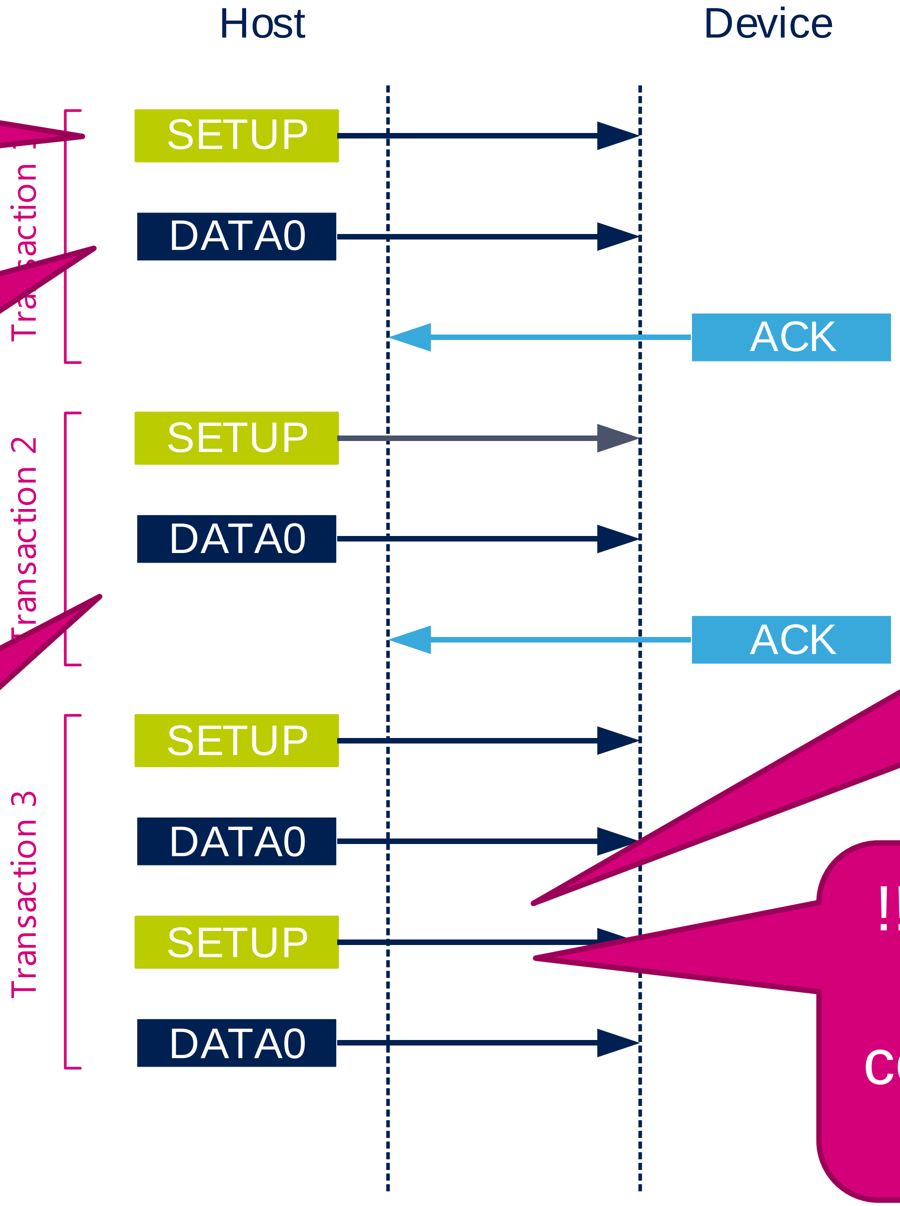
life.augmented

Control SETUP transaction

- Similar to Bulk out transaction
- Used for enumeration and configuration
- Has defined data structure
- Standard request can be refused (not supported) by sending STALL
  - In data phase



Setup



Setup token send only on EP0

Transaction is only from Host to Device with DATA0

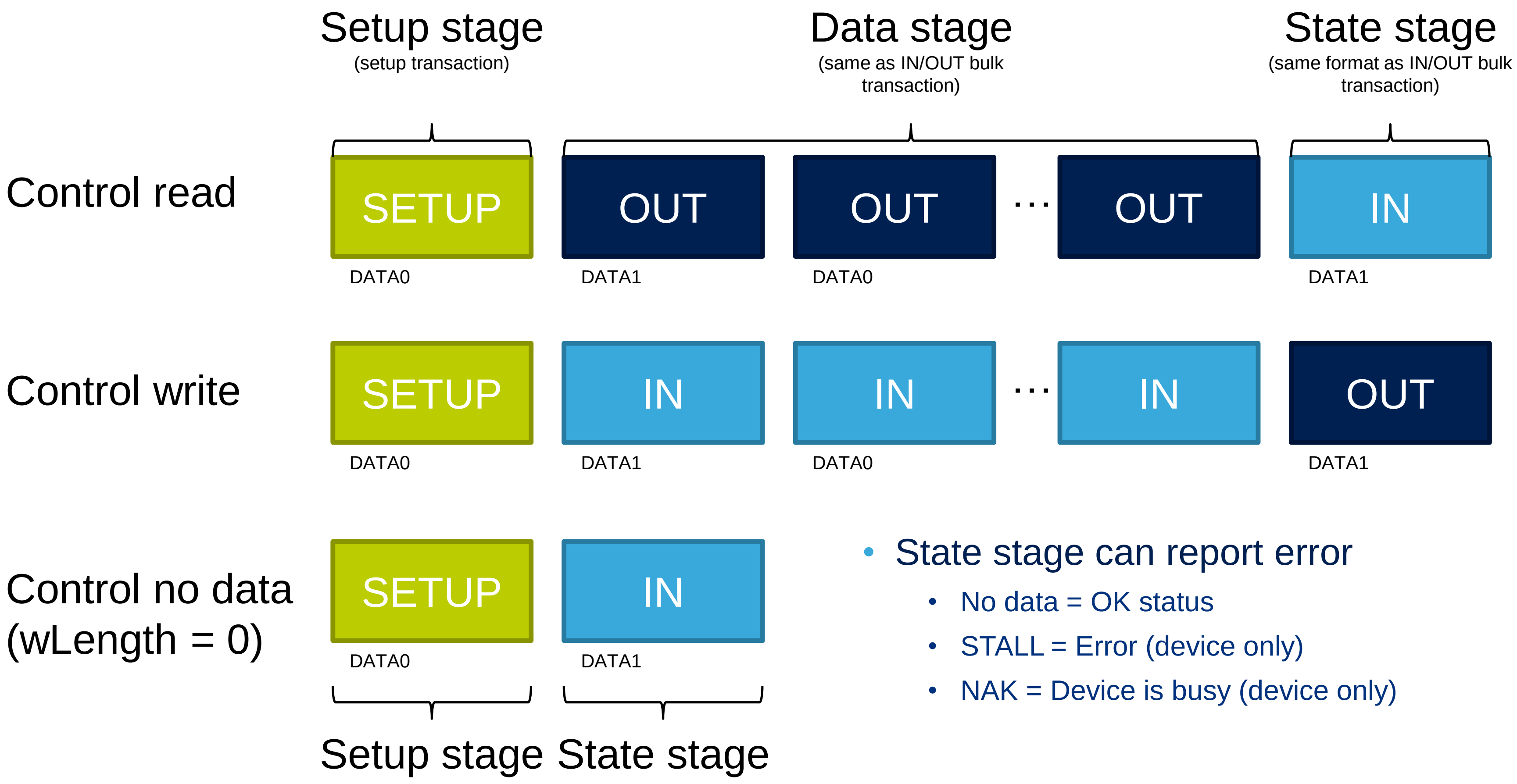
No DATA toggle like in Bulk

Device must ACK the data NAK is not allowed

If data are not acknowledged Host will repeat the Transaction

!! PC resend the SETUP 3x, if is not acknowledged, communication with DEVICE is stopped

Control read/write sequence



- State stage can report error
  - No data = OK status
  - STALL = Error (device only)
  - NAK = Device is busy (device only)

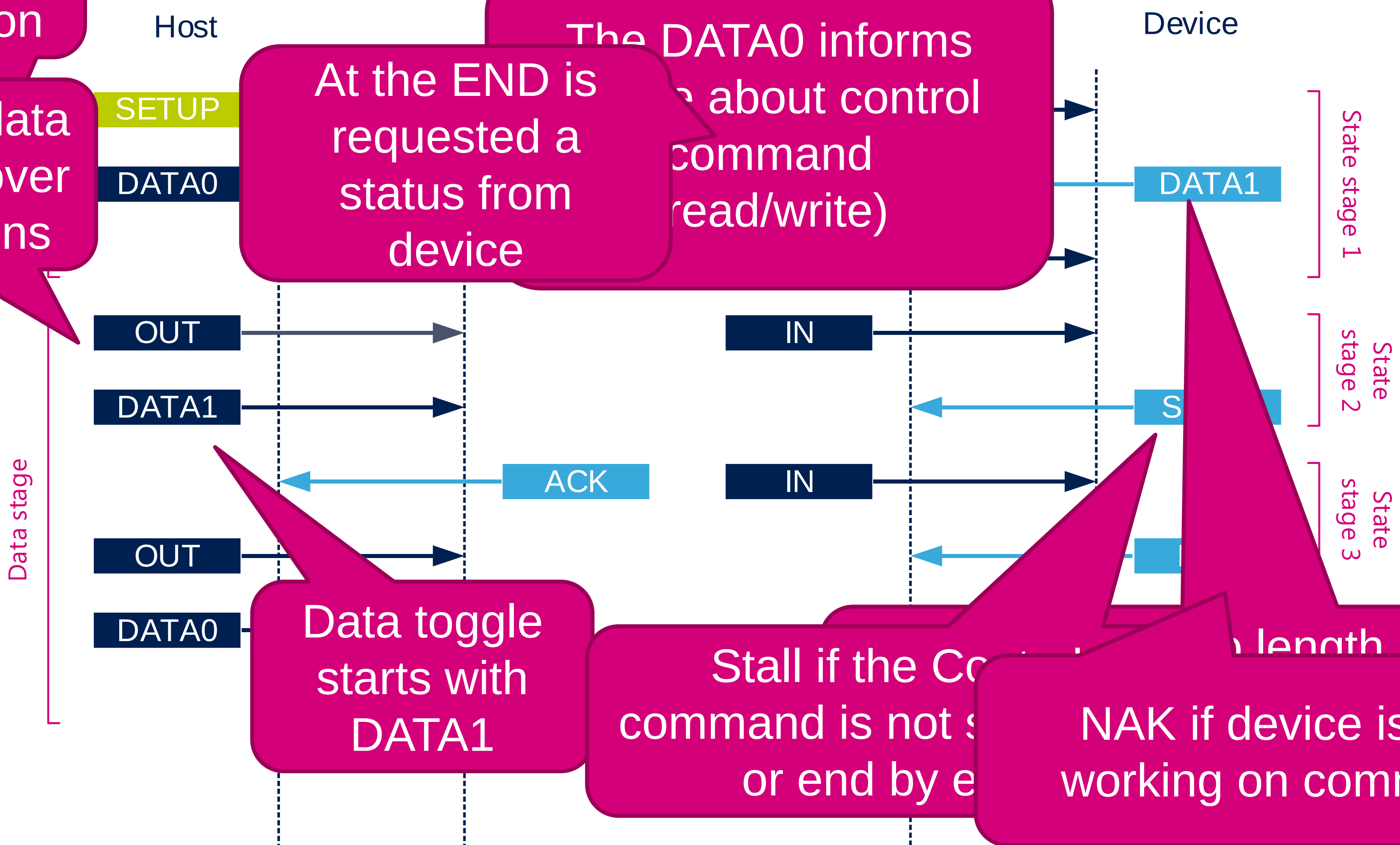
Control Sequence Write

First is send setup transation

For write data are send over OUT tokens

At the END is requested a status from device

The DATA0 informs
e about control
command
read/write)



Data toggle starts with DATA1

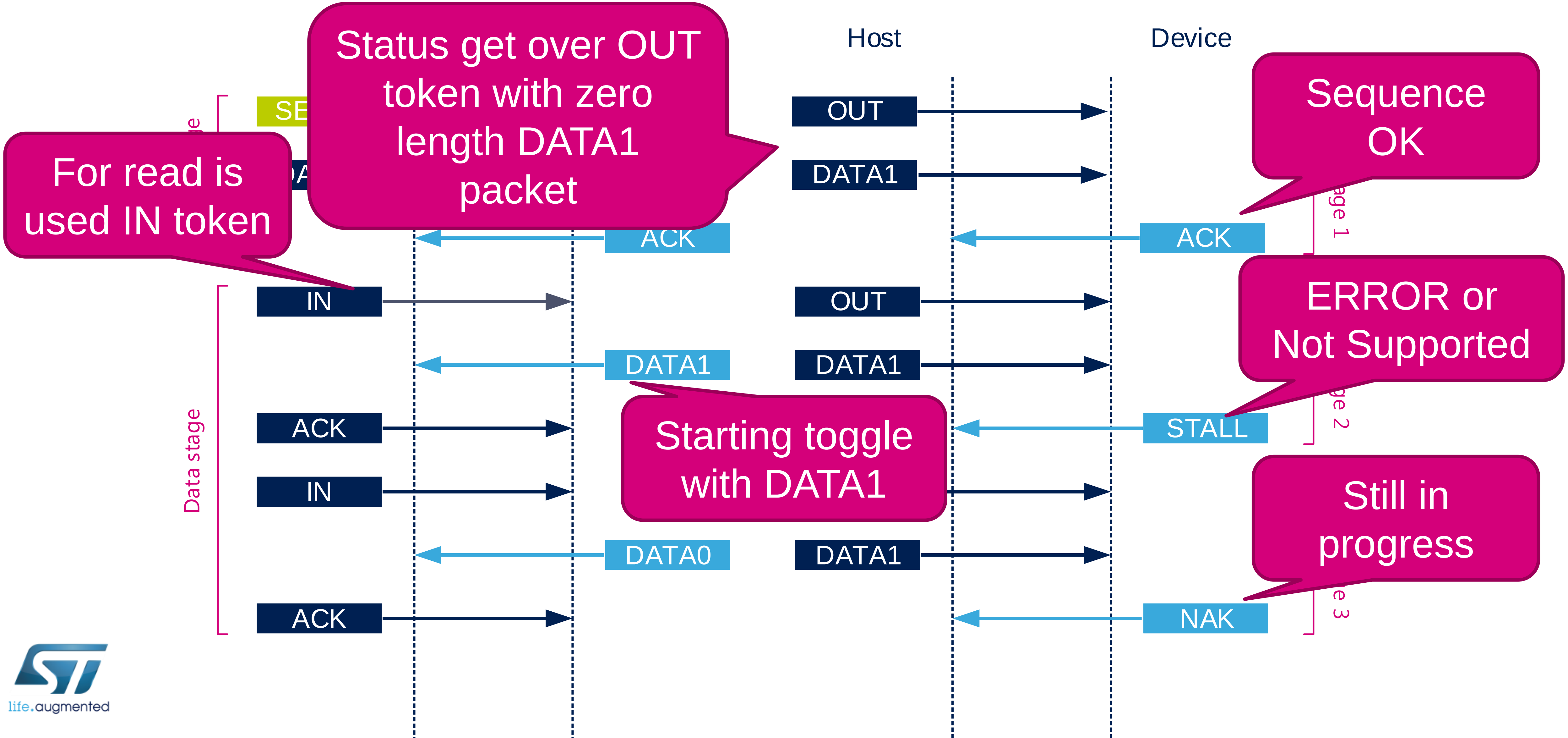
length

Stall if the Co
command is not s
or end by e

NAK if device is still working on command

life.augmented

Control Sequence Read



Status get over OUT token with zero length DATA1 packet

For read is used IN token

Sequence OK

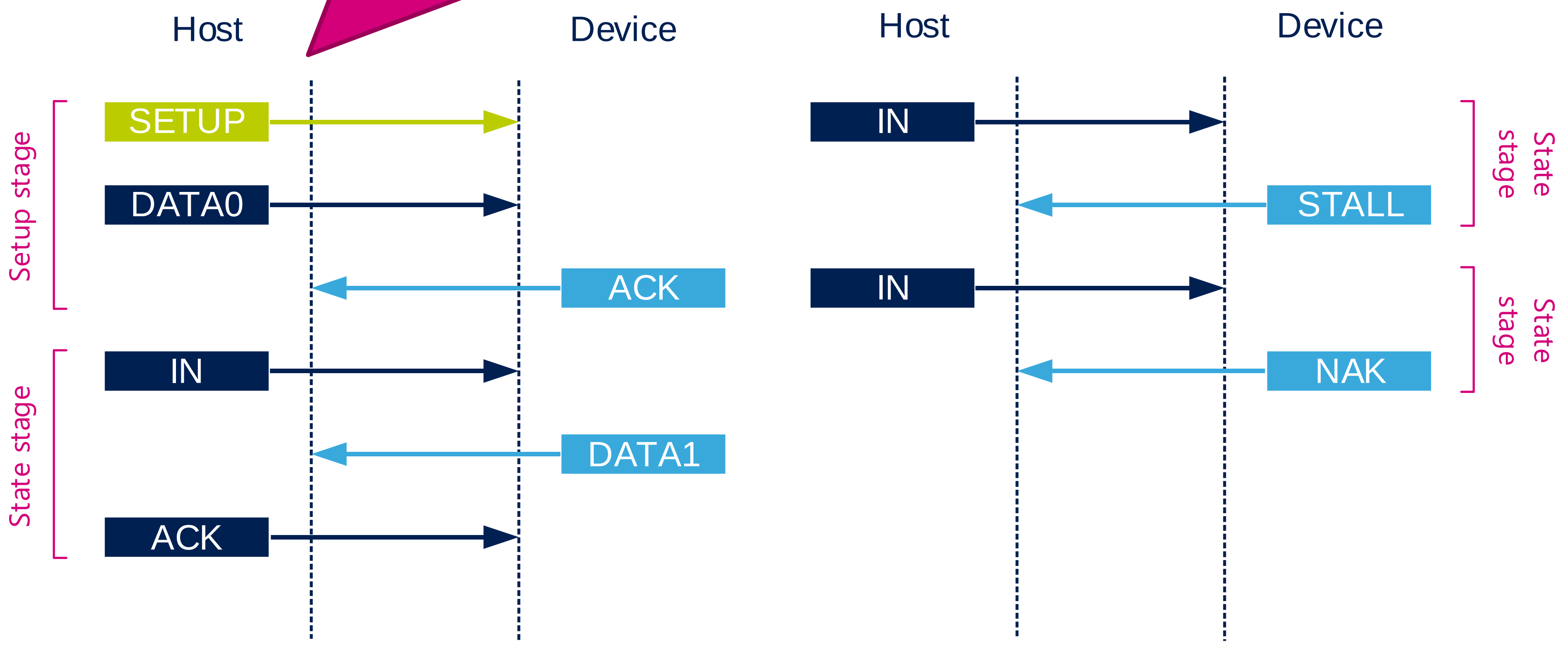
ERROR or Not Supported

Starting toggle with DATA1

Still in progress

Control Sequence with no data

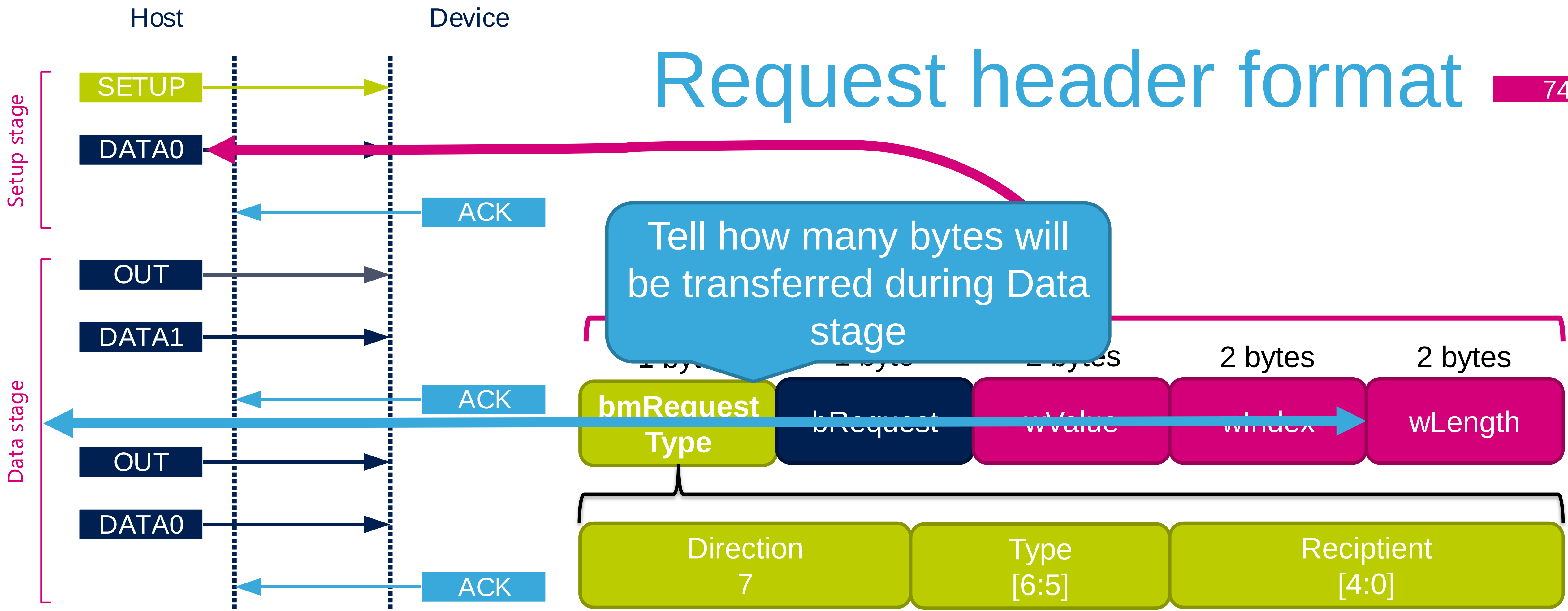
Here is missing a data stage, similar to write sequence



Naming conventions for USB fields

- Each field name contains prefix specifying the size and type
  - b – 1-byte field (integer)
  - bm – 1-byte bit field
  - bcd – 2-byte BCD encoded number
    - Usually for versions e.g. 0x0200 => version 2.0
  - w – 2-byte field
  - id – 2-byte ID
  - i – 1-byte string index
- Little endian used for multi-byte values
  - ARM (by default) and x86 architectures are both little-endian

Request header format



Tell how many bytes will be transferred during Data stage

- 0 – Host-to-device
- 1 – Device-to-host

- 0 – Standard
- 1 – Class specific
- 2 – Vendor specific
- 3 – Reserved

- 0 – Device
- 1 – Interface *
- 2 – Endpoint *
- 3 – Other
- 4-31 – Reserved

* wIndex specifies the endpoint or interface number

Requests

1 byte	1 byte	2 bytes	2 bytes	2 bytes
bmRequest Type	**bRequest**	wValue	wIndex	wLength

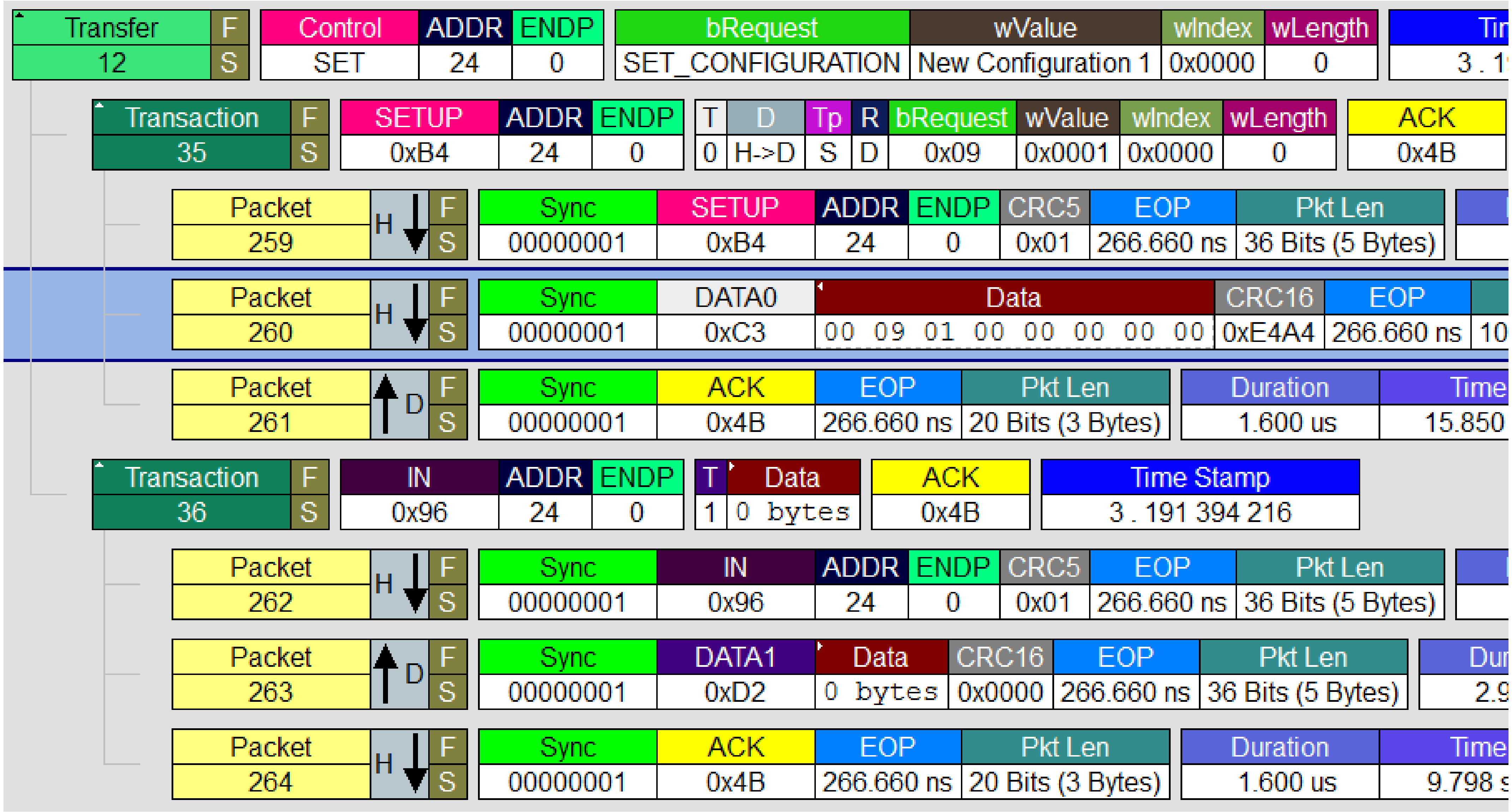
bRequest	Value
GET_STATUS	0
CLEAR_FEATURE	1
Reserved for future use	2
SET_FEATURE	3
Reserved for future use	4
SET_ADDRESS	5
GET_DESCRIPTOR	6
SET_DESCRIPTOR	7

bRequest	Value
GET_CONFIGURATION	8
SET_CONFIGURATION	9
GET_INTERFACE	10
SET_INTERFACE	11
SYNCH_FRAME	12

*More details in USB specification chapter:
9.4 Standard Device Requests

life.augmented

USB request example



Transfer	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	wLength	Tir
12	S	SET	24	0	SET_CONFIGURATION	New Configuration 1	0x0000	0	3 . 1

Transaction	F	SETUP	ADDR	ENDP	T	D	Tp	R	bRequest	wValue	wIndex	wLength	ACK
35	S	0xB4	24	0	0	H->D	S	D	0x09	0x0001	0x0000	0	0x4B

Packet	H ↓	F	Sync	SETUP	ADDR	ENDP	CRC5	EOP	Pkt Len
259		S	00000001	0xB4	24	0	0x01	266.660 ns	36 Bits (5 Bytes)

Packet	H ↓	F	Sync	DATA0	Data	CRC16	EOP	
260		S	00000001	0xC3	00 09 01 00 00 00 00 00	0xE4A4	266.660 ns	10

Packet	↑ D	F	Sync	ACK	EOP	Pkt Len	Duration	Time
261		S	00000001	0x4B	266.660 ns	20 Bits (3 Bytes)	1.600 us	15.850

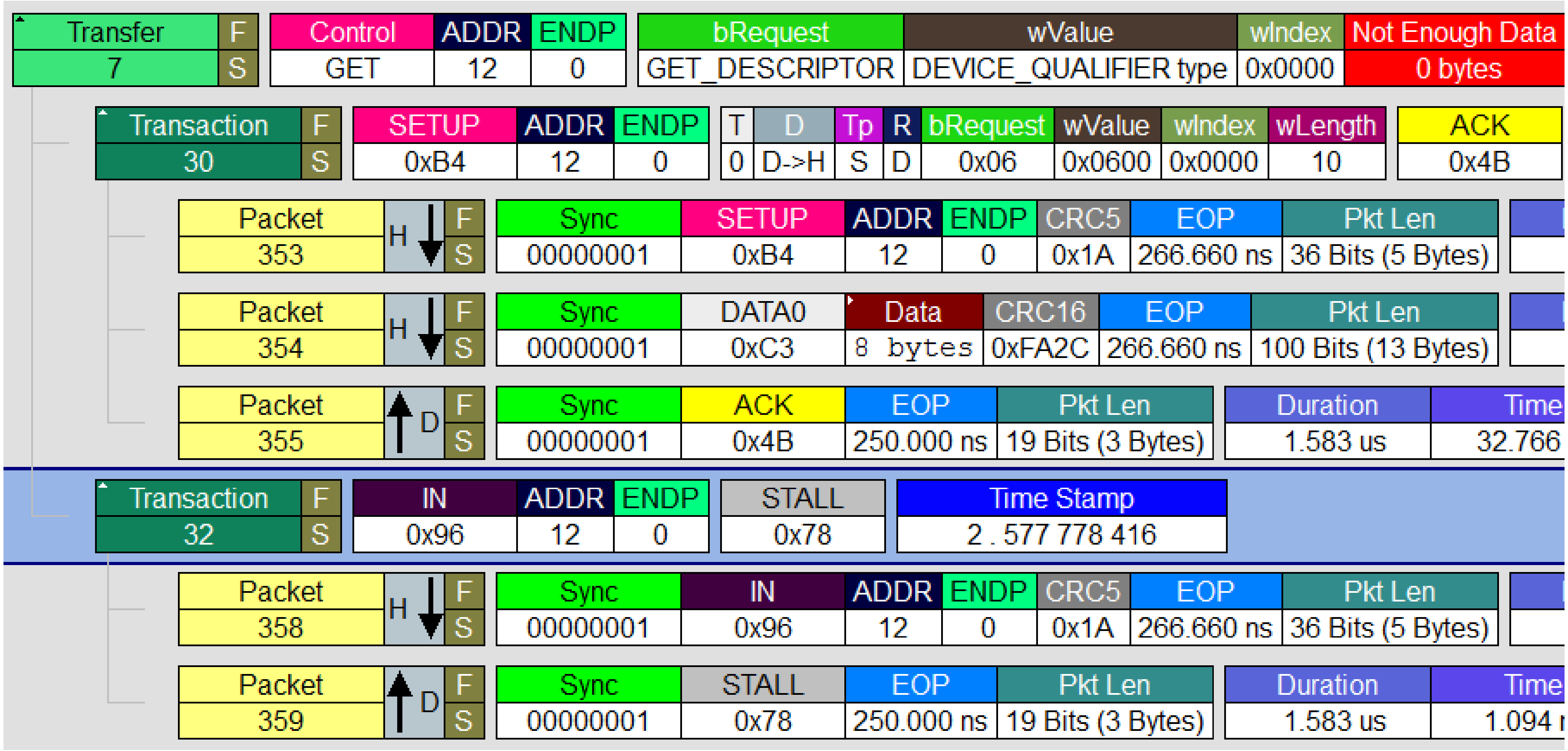
Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time Stamp
36	S	0x96	24	0	1	0 bytes	0x4B	3 . 191 394 216

Packet	H ↓	F	Sync	IN	ADDR	ENDP	CRC5	EOP	Pkt Len
262		S	00000001	0x96	24	0	0x01	266.660 ns	36 Bits (5 Bytes)

Packet	↑ D	F	Sync	DATA1	Data	CRC16	EOP	Pkt Len	Dur
263		S	00000001	0xD2	0 bytes	0x0000	266.660 ns	36 Bits (5 Bytes)	2.9

Packet	H ↓	F	Sync	ACK	EOP	Pkt Len	Duration	Time
264		S	00000001	0x4B	266.660 ns	20 Bits (3 Bytes)	1.600 us	9.798 s

USB request example (STALL)



Transfer	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Not Enough Data
7	S	GET	12	0	GET_DESCRIPTOR	DEVICE_QUALIFIER type	0x0000	0 bytes

Transaction	F	SETUP	ADDR	ENDP	T	D	Tp	R	bRequest	wValue	wIndex	wLength	ACK
30	S	0xB4	12	0	0	D->H	S	D	0x06	0x0600	0x0000	10	0x4B

Packet	H ↓	F	Sync	SETUP	ADDR	ENDP	CRC5	EOP	Pkt Len
353		S	00000001	0xB4	12	0	0x1A	266.660 ns	36 Bits (5 Bytes)

Packet	H ↓	F	Sync	DATA0	Data	CRC16	EOP	Pkt Len
354		S	00000001	0xC3	8 bytes	0xFA2C	266.660 ns	100 Bits (13 Bytes)

Packet	↑ D	F	Sync	ACK	EOP	Pkt Len	Duration	Time
355		S	00000001	0x4B	250.000 ns	19 Bits (3 Bytes)	1.583 us	32.766

Transaction	F	IN	ADDR	ENDP	STALL	Time Stamp
32	S	0x96	12	0	0x78	2 . 577 778 416

Packet	H ↓	F	Sync	IN	ADDR	ENDP	CRC5	EOP	Pkt Len
358		S	00000001	0x96	12	0	0x1A	266.660 ns	36 Bits (5 Bytes)

Packet	↑ D	F	Sync	STALL	EOP	Pkt Len	Duration	Time
359		S	00000001	0x78	250.000 ns	19 Bits (3 Bytes)	1.583 us	1.094

SPLIT and PING transactions

- Optimize high-speed bus utilization
- SPLIT
  - Used for communication with USB HUBs (split transactions), while low-speed/full-speed device is connected to HUB
  - HUB communicates on lower speed
  - Is not covered in this presentation
- PING
  - Used for bulk OUT
  - Sending long packets that are NAK blocks the bus
  - Ask device if is ready accept data

Bandwidth constraints

- Maximum packet size (depending on transfer type and speed)

	Control	Bulk	Isochronous	Isochronous (kb/s)*	Interrupt
Low-speed	8	-	-	-	8
Full-speed	64	64	1023	1023	64
High-speed	64	512	1024	1024 × 3 × 8 = 24,576	1024

- Full-speed:
  - 10% reserved for control transfers
  - 90% reserved for periodic transactions(isochronous and interrupt)
  - Rest is used for bulk transaction

- High-speed:
  - 20% reserved for control transfers
  - 80% reserved for periodic transactions(isochronous and interrupt)
  - Rest is used for bulk transaction

* 1000 bytes per second for single endpoint