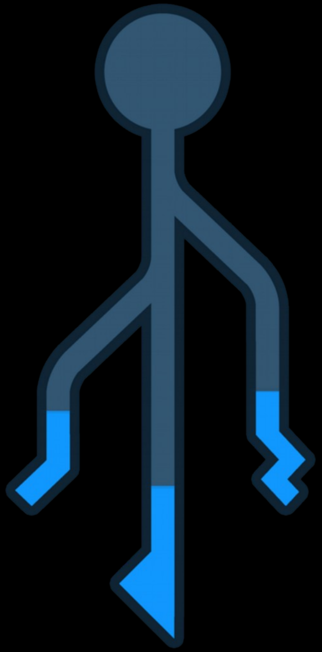


BREAKING USB

WITHOUT BREAKING THE BANK

TODAY'S TUNNEL VISION:
USB CONTROL REQUESTS

AND NOW TRUE HORRORS FROM THE DARK TIME

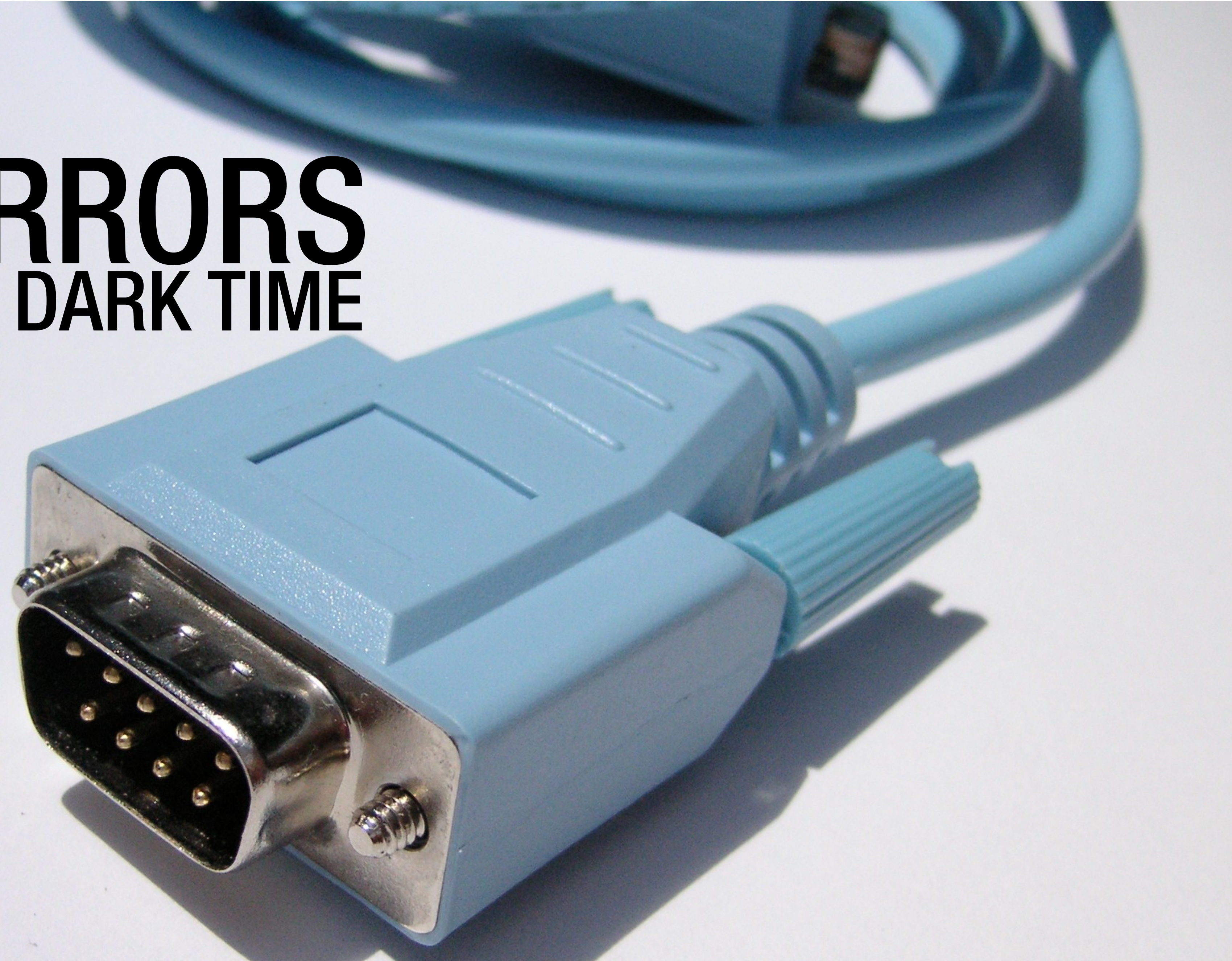
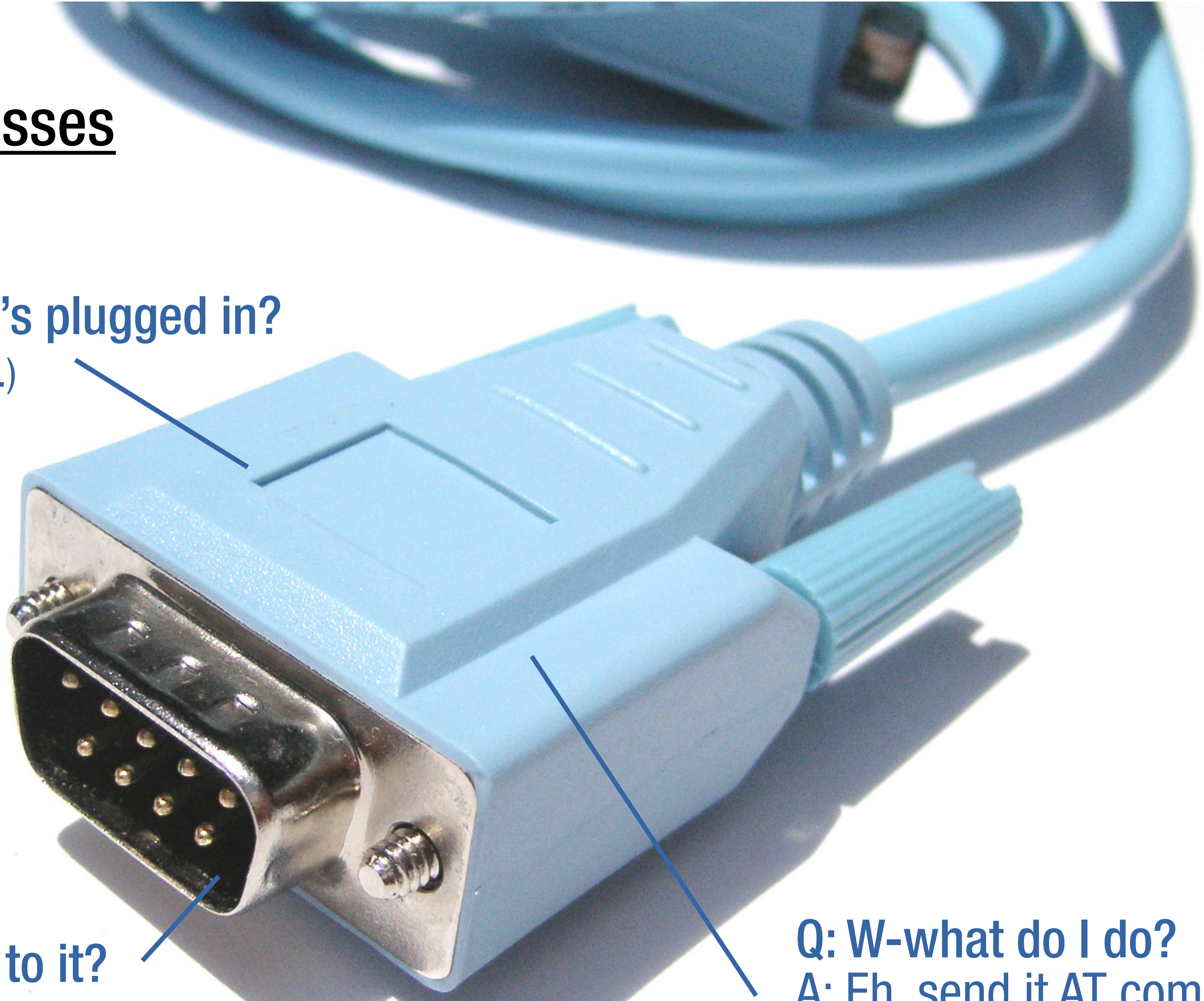


Image attribution: wikimedia user Mobius

Legacy Peripheral Busses

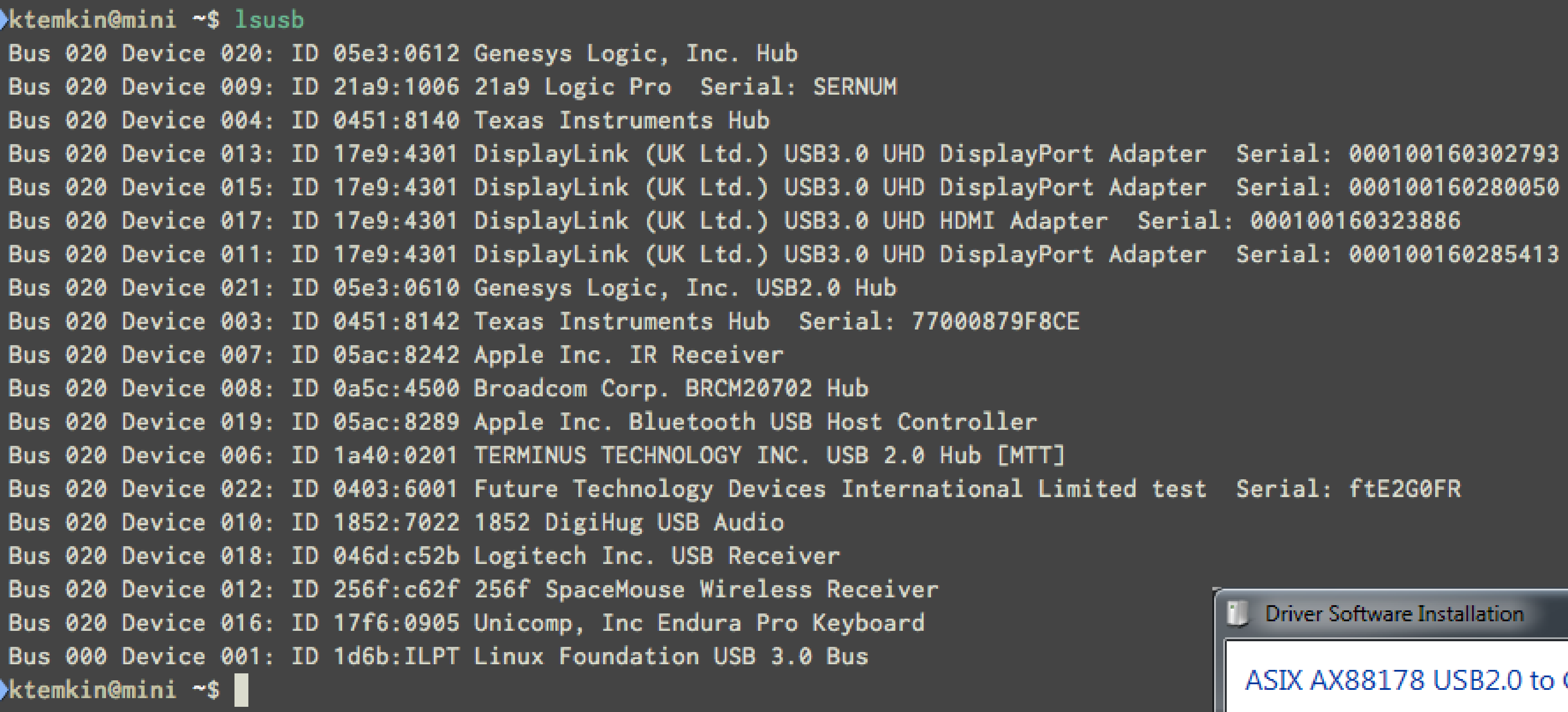
e.g. RS-232, IEEE 1284



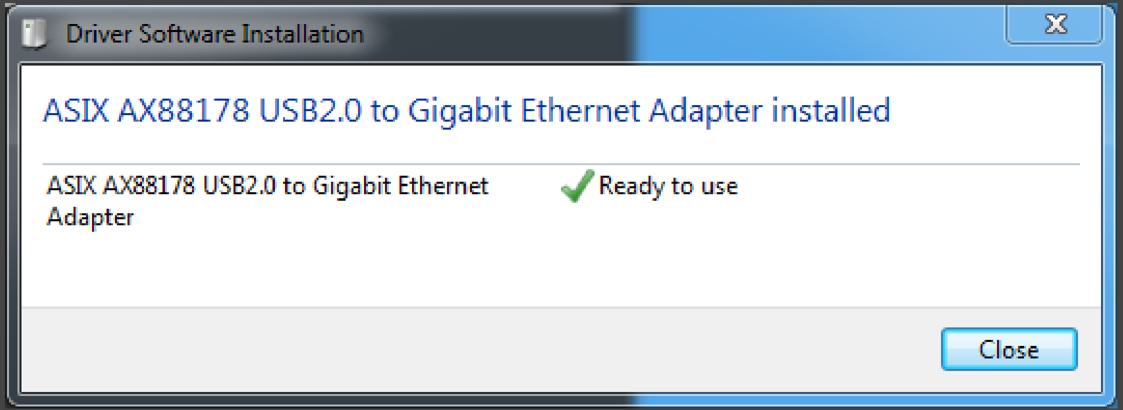
Q: How do you know what's plugged in?
A: You don't. (Use your eyeballs.)

Q: How I know how to talk to it?
A: You don't. (Did it come with a disk?.)

Q: W-what do I do?
A: Eh, send it AT commands.
It's probably a modem.



```
ktemkin@mini ~$ lsusb
Bus 020 Device 020: ID 05e3:0612 Genesys Logic, Inc. Hub
Bus 020 Device 009: ID 21a9:1006 21a9 Logic Pro  Serial: SERNUM
Bus 020 Device 004: ID 0451:8140 Texas Instruments Hub
Bus 020 Device 013: ID 17e9:4301 DisplayLink (UK Ltd.) USB3.0 UHD DisplayPort Adapter  Serial: 000100160302793
Bus 020 Device 015: ID 17e9:4301 DisplayLink (UK Ltd.) USB3.0 UHD DisplayPort Adapter  Serial: 000100160280050
Bus 020 Device 017: ID 17e9:4301 DisplayLink (UK Ltd.) USB3.0 UHD HDMI Adapter  Serial: 000100160323886
Bus 020 Device 011: ID 17e9:4301 DisplayLink (UK Ltd.) USB3.0 UHD DisplayPort Adapter  Serial: 000100160285413
Bus 020 Device 021: ID 05e3:0610 Genesys Logic, Inc. USB2.0 Hub
Bus 020 Device 003: ID 0451:8142 Texas Instruments Hub  Serial: 77000879F8CE
Bus 020 Device 007: ID 05ac:8242 Apple Inc. IR Receiver
Bus 020 Device 008: ID 0a5c:4500 Broadcom Corp. BRCM20702 Hub
Bus 020 Device 019: ID 05ac:8289 Apple Inc. Bluetooth USB Host Controller
Bus 020 Device 006: ID 1a40:0201 TERMINUS TECHNOLOGY INC. USB 2.0 Hub [MTT]
Bus 020 Device 022: ID 0403:6001 Future Technology Devices International Limited test  Serial: ftE2G0FR
Bus 020 Device 010: ID 1852:7022 1852 DigiHug USB Audio
Bus 020 Device 018: ID 046d:c52b Logitech Inc. USB Receiver
Bus 020 Device 012: ID 256f:c62f 256f SpaceMouse Wireless Receiver
Bus 020 Device 016: ID 17f6:0905 Unicomp, Inc Endura Pro Keyboard
Bus 000 Device 001: ID 1d6b:ILPT Linux Foundation USB 3.0 Bus
ktemkin@mini ~$ 
```

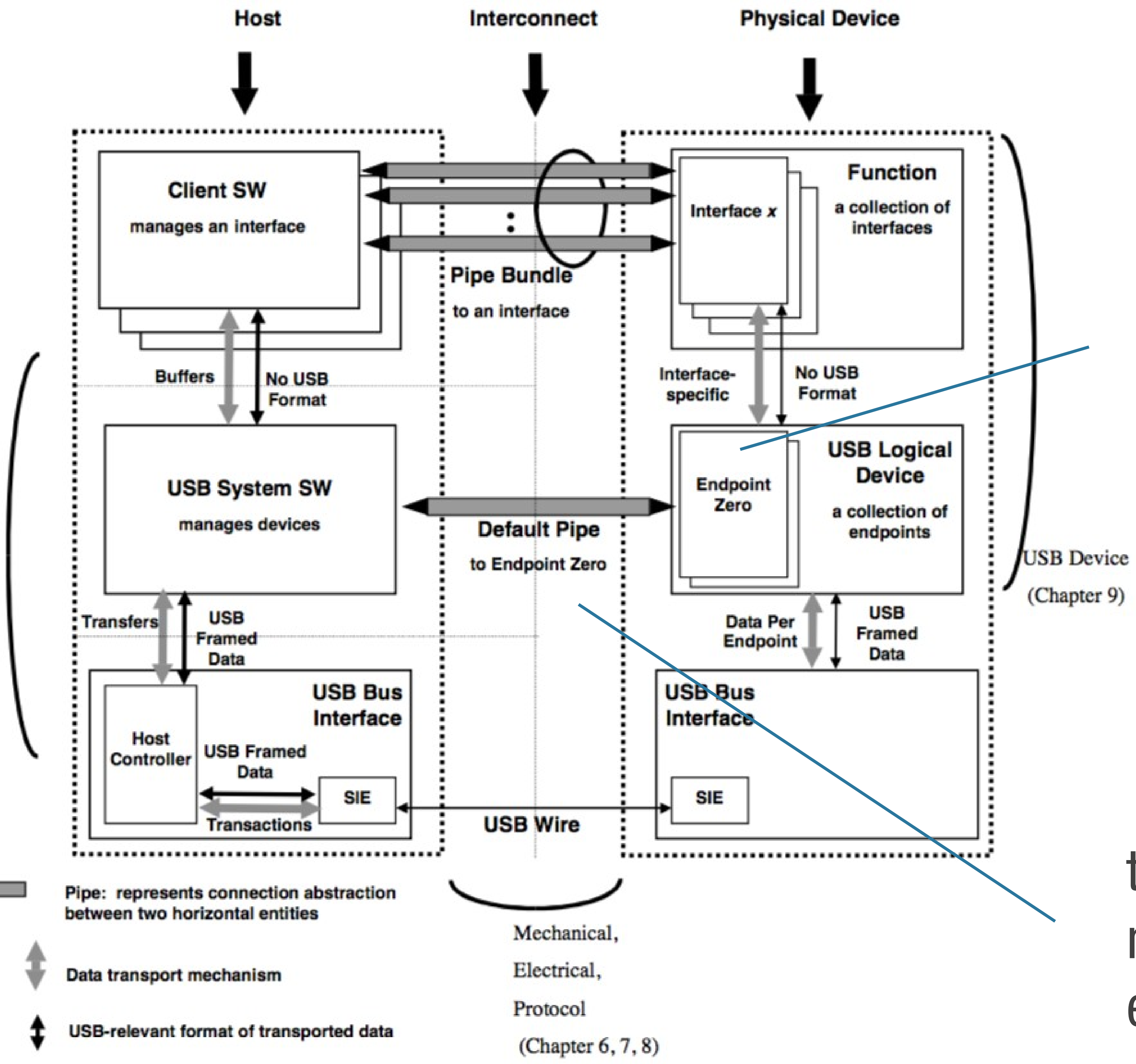


ASIX AX88178 USB2.0 to Gigabit Ethernet Adapter installed

ASIX AX88178 USB2.0 to Gigabit Ethernet Adapter ✓ Ready to use

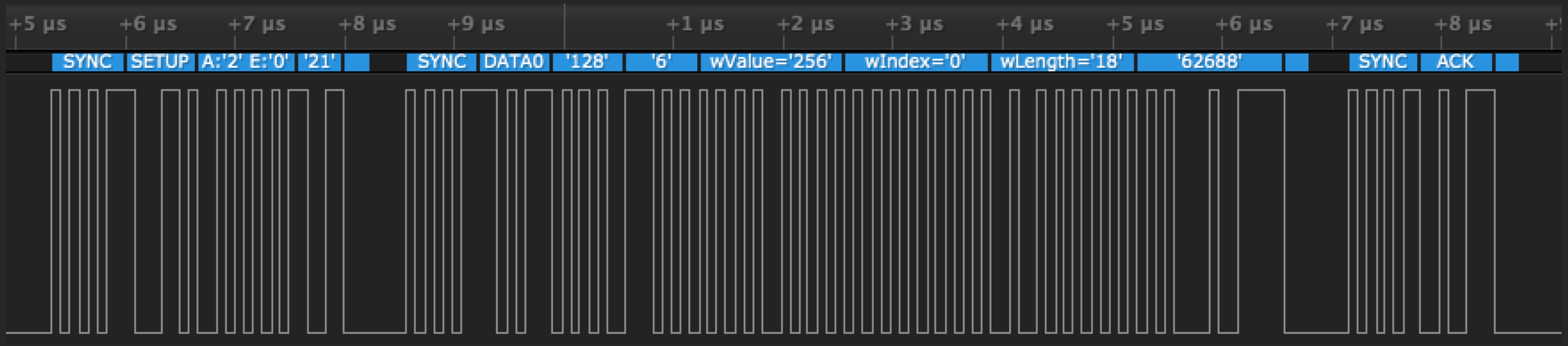
enumeration allows devices to be identified and paired with the correct drivers automatically

USB: ENUMERATION

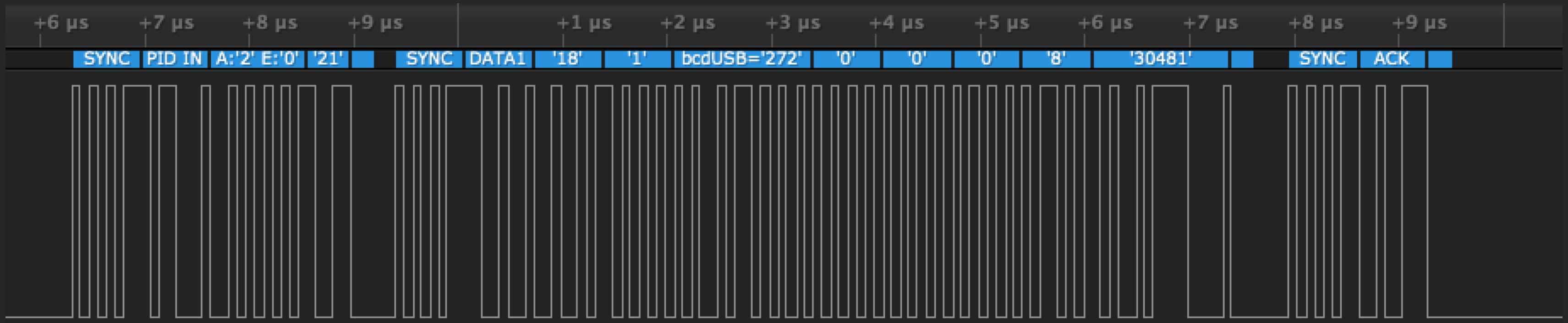


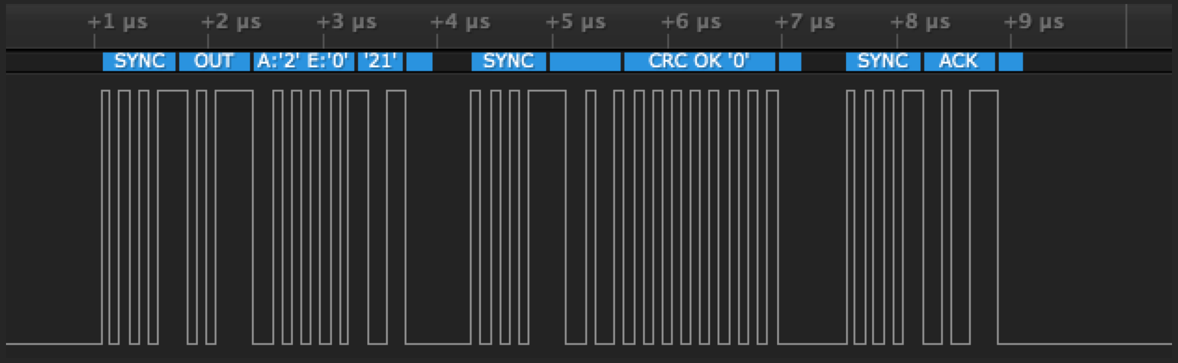
to support enumeration, every USB device must support a standard protocol on one of its channels

this protocol is carried over the most common form of USB data exchange: usb control transfers



each control transfer is made up of three stages…





…which form a simple command-and-response protocol.

SETUP STAGE

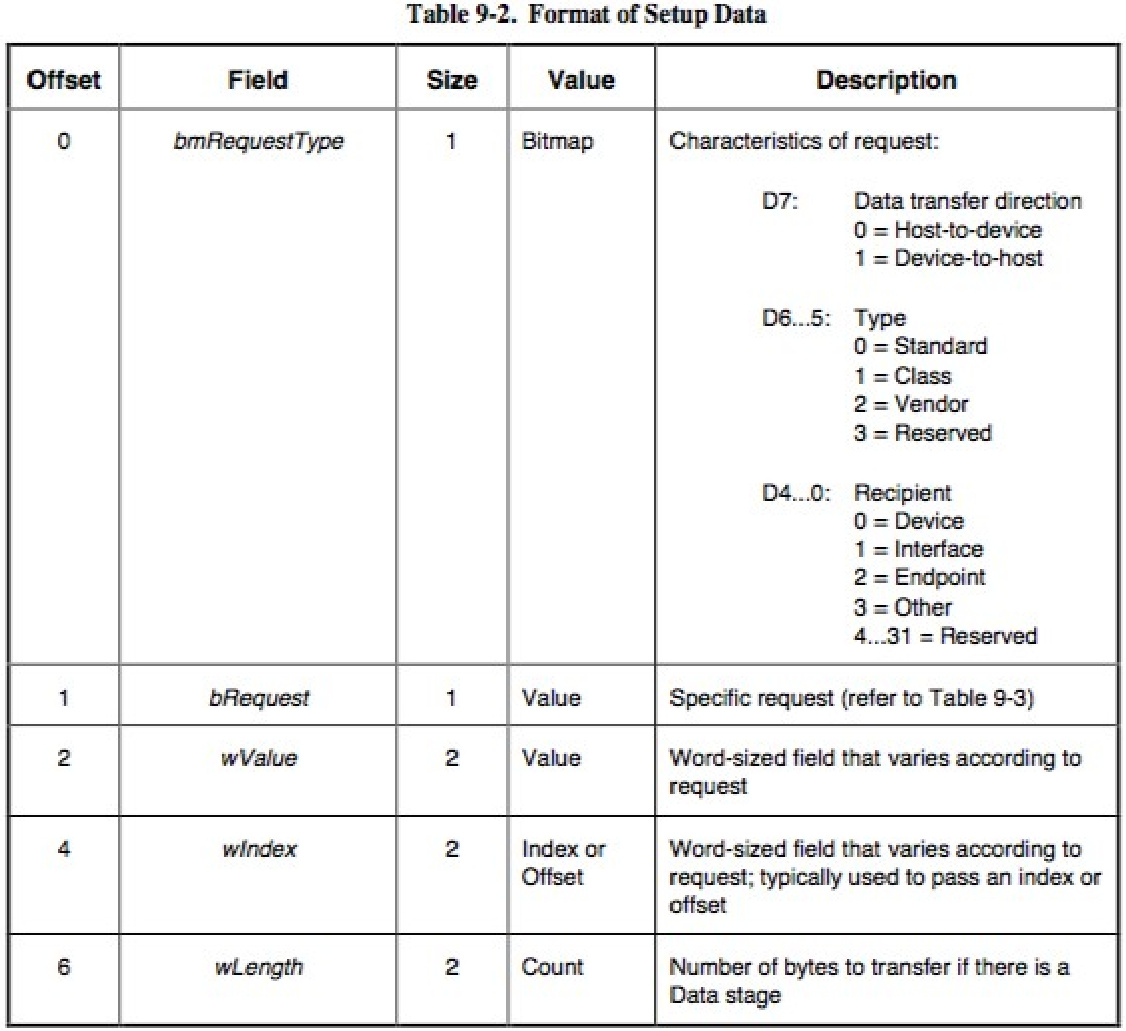
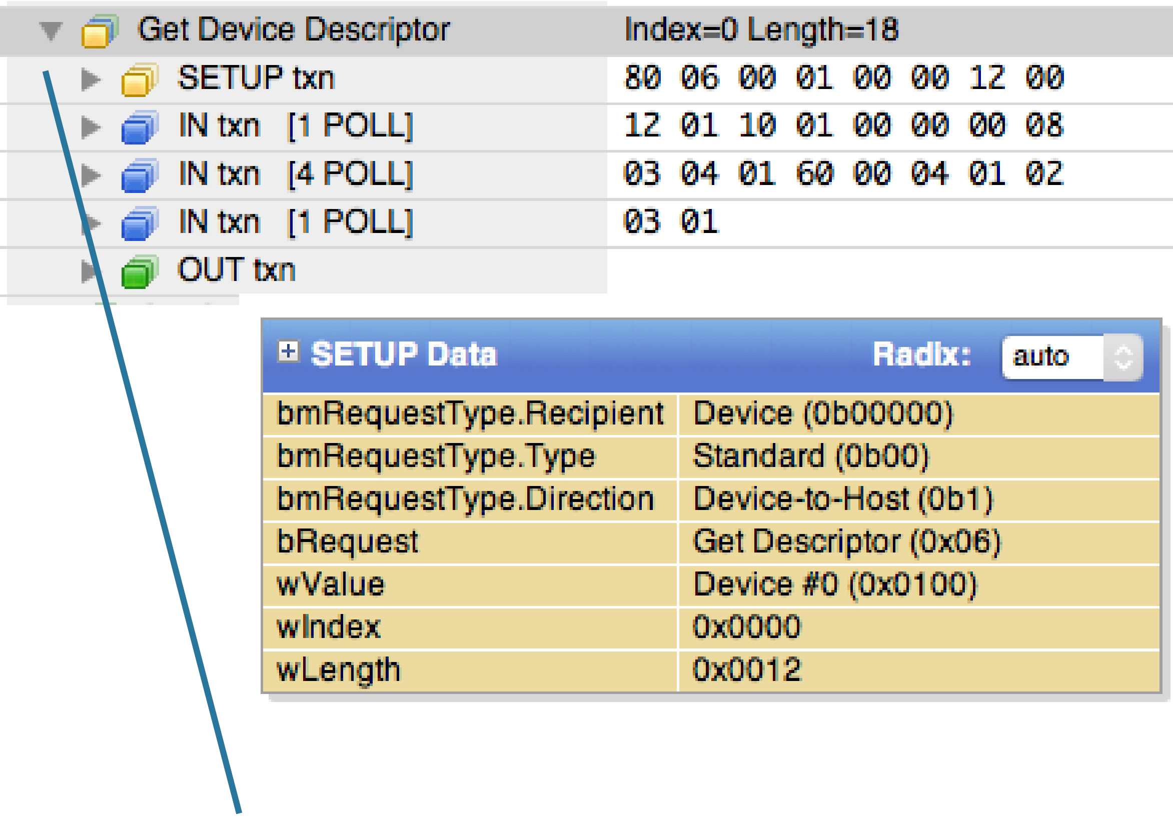


Table 9-2. Format of Setup Data

Offset	Field	Size	Value	Description
0	*bmRequestType*	1	Bitmap	Characteristics of request: D7: Data transfer direction 0 = Host-to-device 1 = Device-to-host D6...5: Type 0 = Standard 1 = Class 2 = Vendor 3 = Reserved D4...0: Recipient 0 = Device 1 = Interface 2 = Endpoint 3 = Other 4...31 = Reserved
1	*bRequest*	1	Value	Specific request (refer to Table 9-3)
2	*wValue*	2	Value	Word-sized field that varies according to request
4	*wIndex*	2	Index or Offset	Word-sized field that varies according to request; typically used to pass an index or offset
6	*wLength*	2	Count	Number of bytes to transfer if there is a Data stage



```
▼ Get Device Descriptor      Index=0 Length=18
  ▶ SETUP txn                80 06 00 01 00 00 12 00
  ▶ IN txn   [1 POLL]        12 01 10 01 00 00 00 08
  ▶ IN txn   [4 POLL]        03 04 01 60 00 04 01 02
  ▶ IN txn   [1 POLL]        03 01
  ▶ OUT txn
```

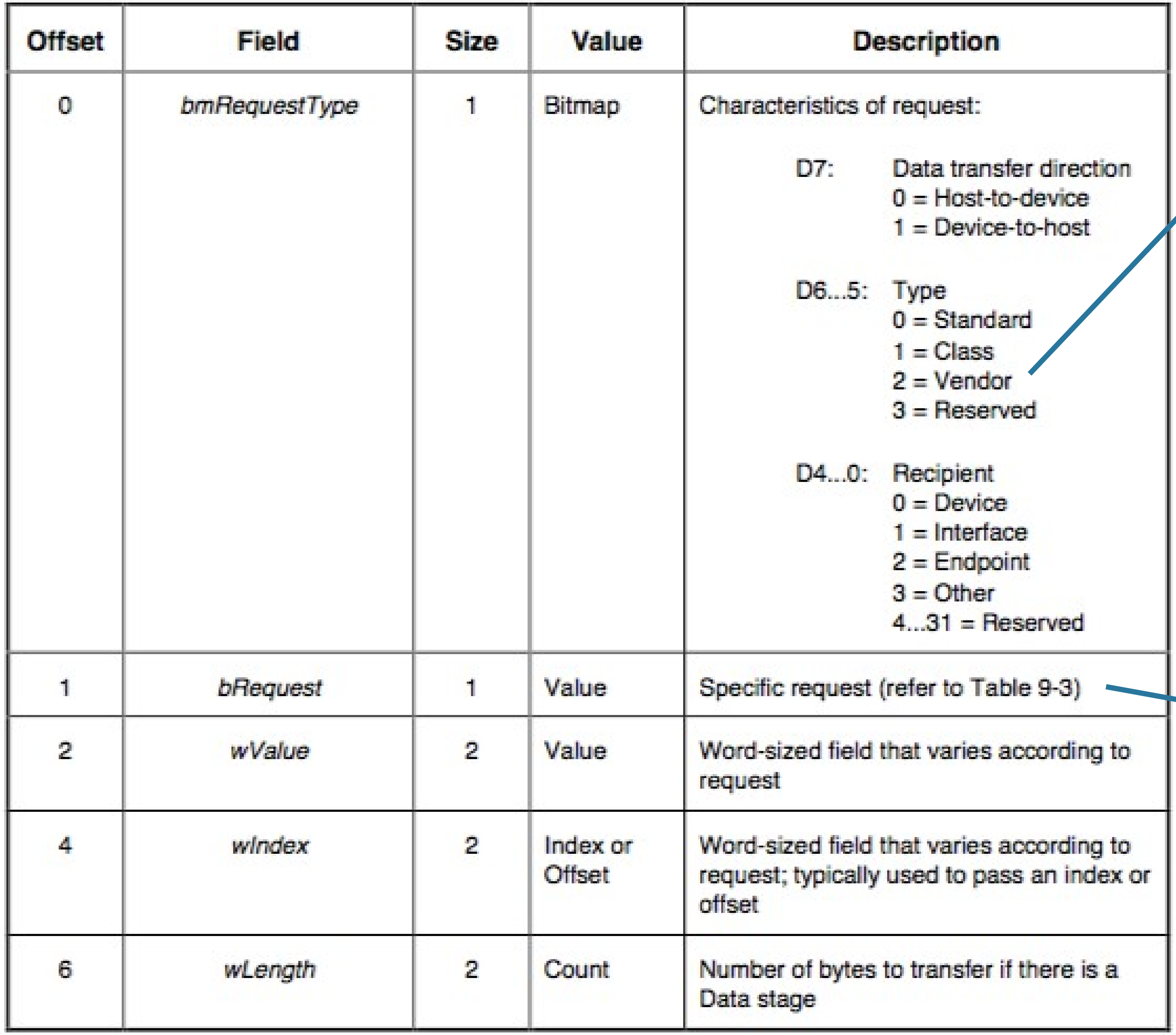
⊞ SETUP Data — Radix: auto

bmRequestType.Recipient	Device (0b00000)
bmRequestType.Type	Standard (0b00)
bmRequestType.Direction	Device-to-Host (0b1)
bRequest	Get Descriptor (0x06)
wValue	Device #0 (0x0100)
wIndex	0x0000
wLength	0x0012

Each control transfer begins with a setup stage, describing the "command" (or "*request*") the host wants the device to perform.

SETUP PACKETS

Table 9-2. Format of Setup Data



Offset	Field	Size	Value	Description
0	*bmRequestType*	1	Bitmap	Characteristics of request: D7: Data transfer direction 0 = Host-to-device 1 = Device-to-host D6...5: Type 0 = Standard 1 = Class 2 = Vendor 3 = Reserved D4...0: Recipient 0 = Device 1 = Interface 2 = Endpoint 3 = Other 4...31 = Reserved
1	*bRequest*	1	Value	Specific request (refer to Table 9-3)
2	*wValue*	2	Value	Word-sized field that varies according to request
4	*wIndex*	2	Index or Offset	Word-sized field that varies according to request; typically used to pass an index or offset
6	*wLength*	2	Count	Number of bytes to transfer if there is a Data stage

The setup stage of a control transfer describes the type of request, the size and direction of the any data to be transferred, and provides space for simple arguments.

The USB specification requires all devices to support a number of standard control requests, which are used for enumeration and configuration.

SETUP PACKETS

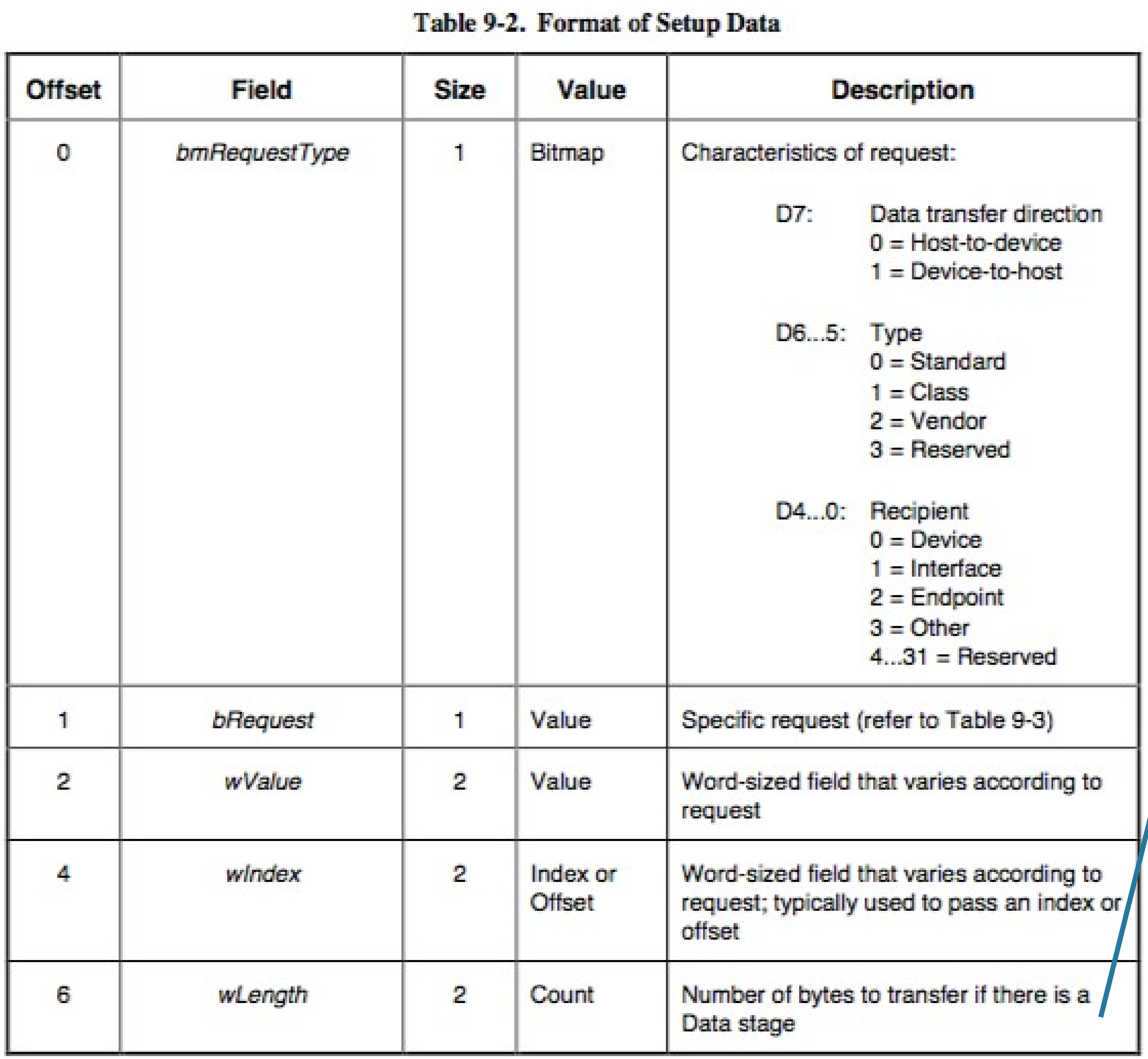


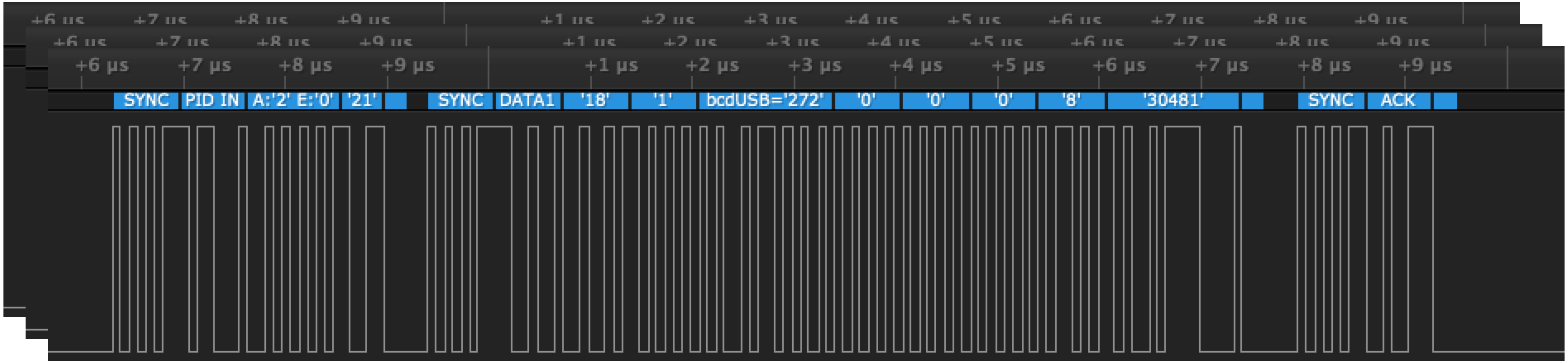
Table 9-2. Format of Setup Data

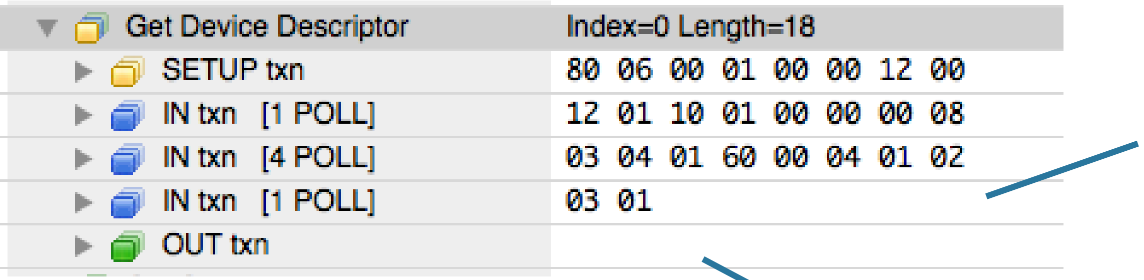
Offset	Field	Size	Value	Description
0	*bmRequestType*	1	Bitmap	Characteristics of request: D7: Data transfer direction 0 = Host-to-device 1 = Device-to-host D6...5: Type 0 = Standard 1 = Class 2 = Vendor 3 = Reserved D4...0: Recipient 0 = Device 1 = Interface 2 = Endpoint 3 = Other 4...31 = Reserved
1	*bRequest*	1	Value	Specific request (refer to Table 9-3)
2	*wValue*	2	Value	Word-sized field that varies according to request
4	*wIndex*	2	Index or Offset	Word-sized field that varies according to request; typically used to pass an index or offset
6	*wLength*	2	Count	Number of bytes to transfer if there is a Data stage

Finally, each setup packet contains a length field, which indicates the maximum amount of data to be transferred over the request.

Data is only transferred in one direction per request: IN to the host, or OUT to the device.

DATA STAGE



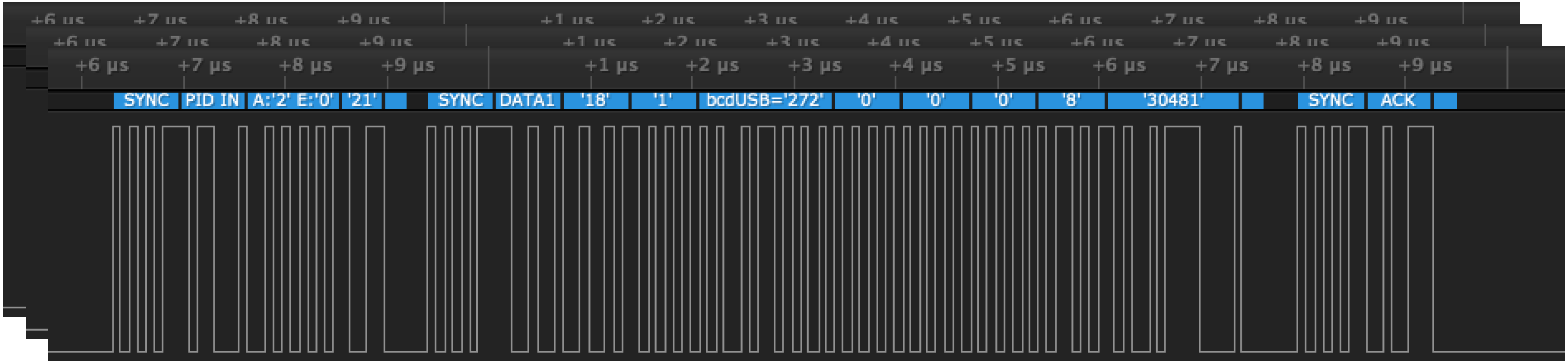


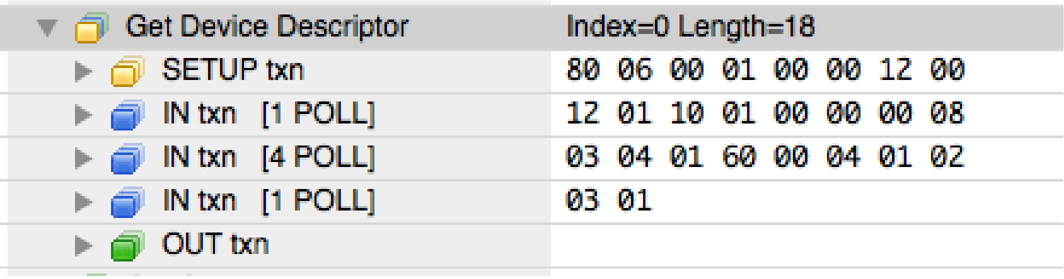
```
▼ Get Device Descriptor     Index=0 Length=18
  ▶ SETUP txn               80 06 00 01 00 00 12 00
  ▶ IN txn   [1 POLL]       12 01 10 01 00 00 00 08
  ▶ IN txn   [4 POLL]       03 04 01 60 00 04 01 02
  ▶ IN txn   [1 POLL]       03 01
  ▶ OUT txn
```

If the request has a non-zero length, the control transfer has a data stage during which data will be transferred in a single direction: either from host to device (OUT) or from device to host (IN).

An OUT transfer always carries the maximum length of data advertised. An IN transfer can contain any amount of data up to the maximum, but should never carry more.

STALLING



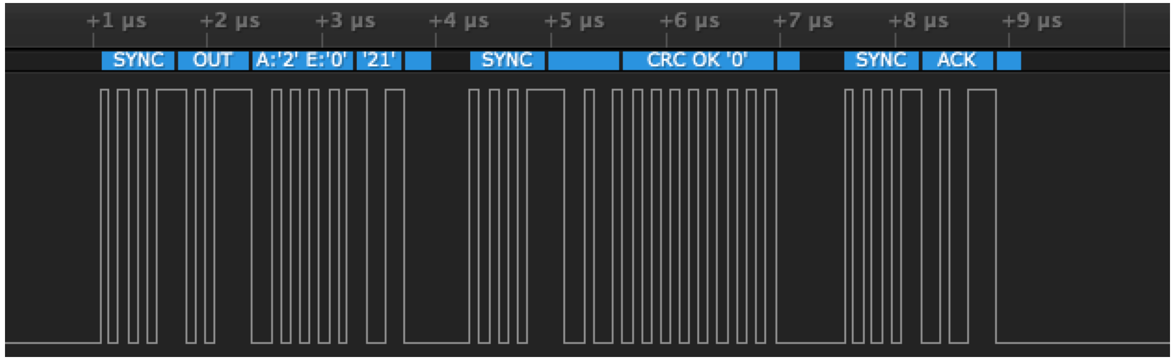


```
▼ Get Device Descriptor    Index=0 Length=18
  ▶ SETUP txn              80 06 00 01 00 00 12 00
  ▶ IN txn   [1 POLL]      12 01 10 01 00 00 00 08
  ▶ IN txn   [4 POLL]      03 04 01 60 00 04 01 02
  ▶ IN txn   [1 POLL]      03 01
  ▶ OUT txn
```

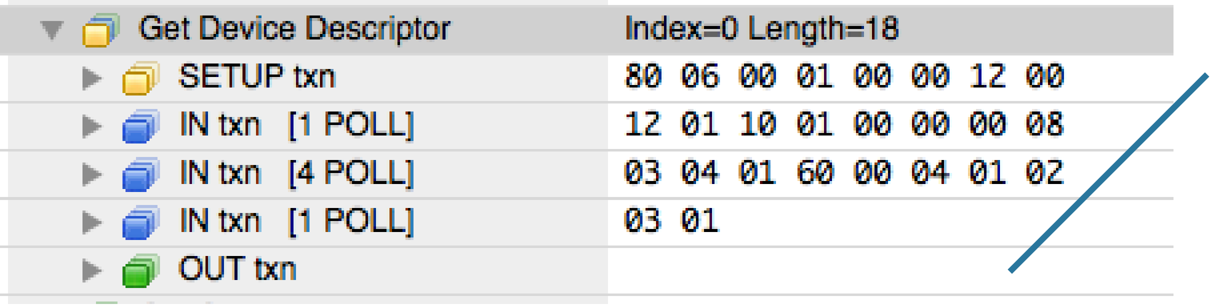
If the host ever asks for data in a way the device doesn't support, the device can respond with a STALL instead of the requested data.

Stalling a request communicates that the given request isn't expected to succeed in the future – and hints that the host shouldn't try it again.

STATUS STAGE



Each control transfer ends with a status stage, which confirms both sides agree that a transaction completed correctly.



▼ Get Device Descriptor	Index=0 Length=18
▶ SETUP txn	80 06 00 01 00 00 12 00
▶ IN txn [1 POLL]	12 01 10 01 00 00 00 08
▶ IN txn [4 POLL]	03 04 01 60 00 04 01 02
▶ IN txn [1 POLL]	03 01
▶ OUT txn	

The status stage provides an opportunity either side to report a transmission error, or for the device to report that it doesn't know how to handle the data it's been sent.

In the latter case, a device can indicate that an 'OUT request' – a request transferring data from host to device – isn't supported by responding with a STALL.

WHEW.

THAT’S A LOT OF INFO.
LET’S SEE WHAT SOME CODE LOOKS LIKE.

a playground script is available on the course website