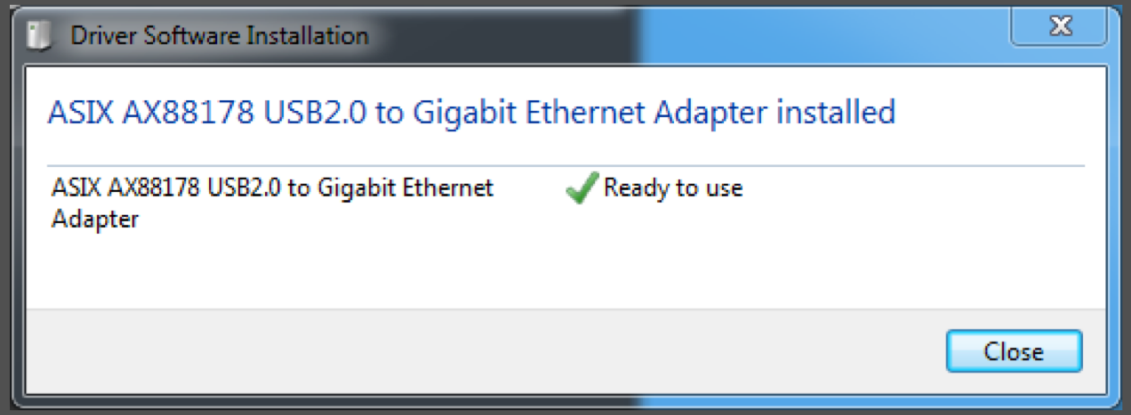
HACKING THE USB WORLD WITH FACEDANCER

ENUMERATION AND THE CONTROL ENDPOINT

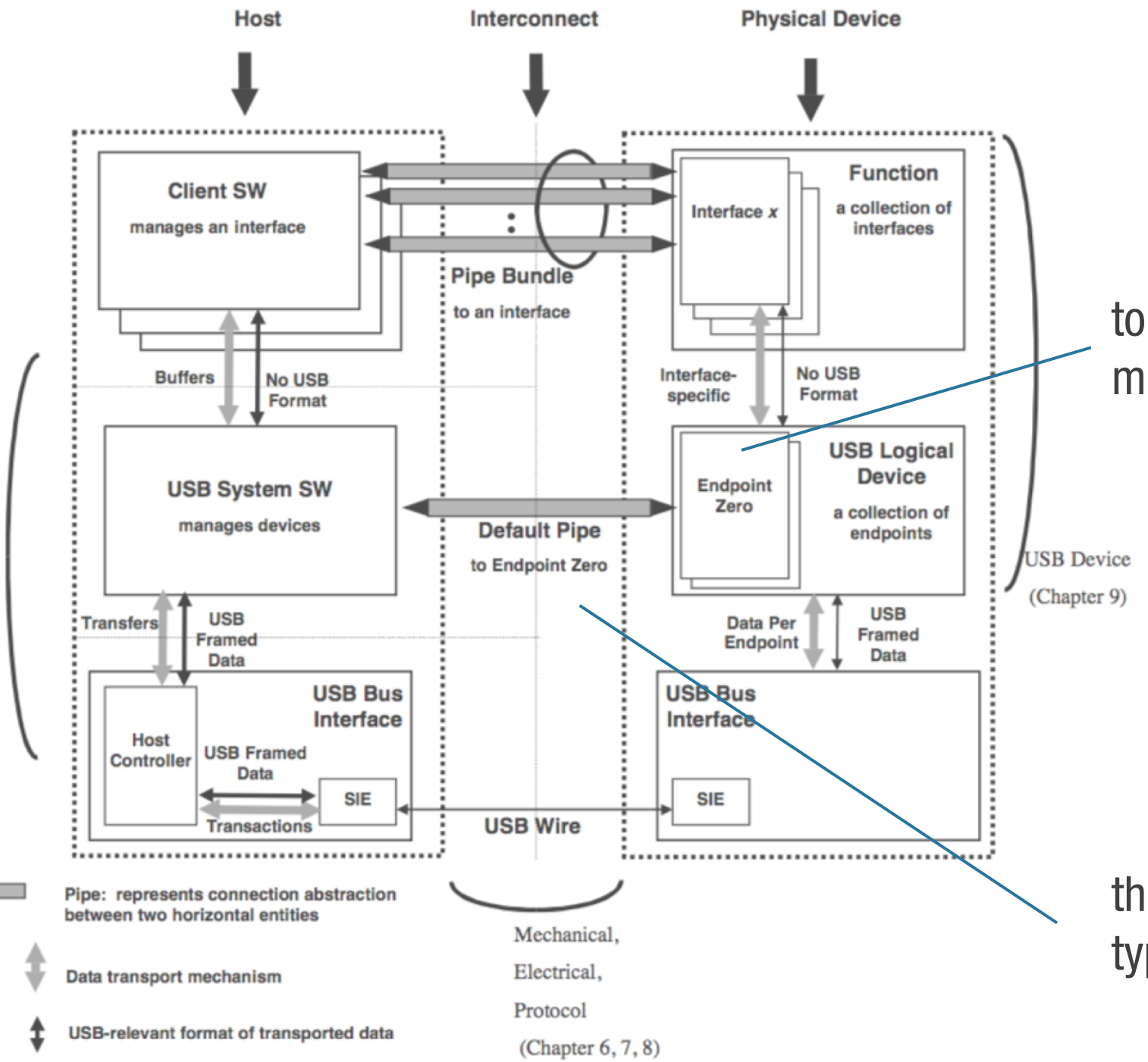
KATE TEMKIN & DOMINIC SPILL

```
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 ~$
```



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

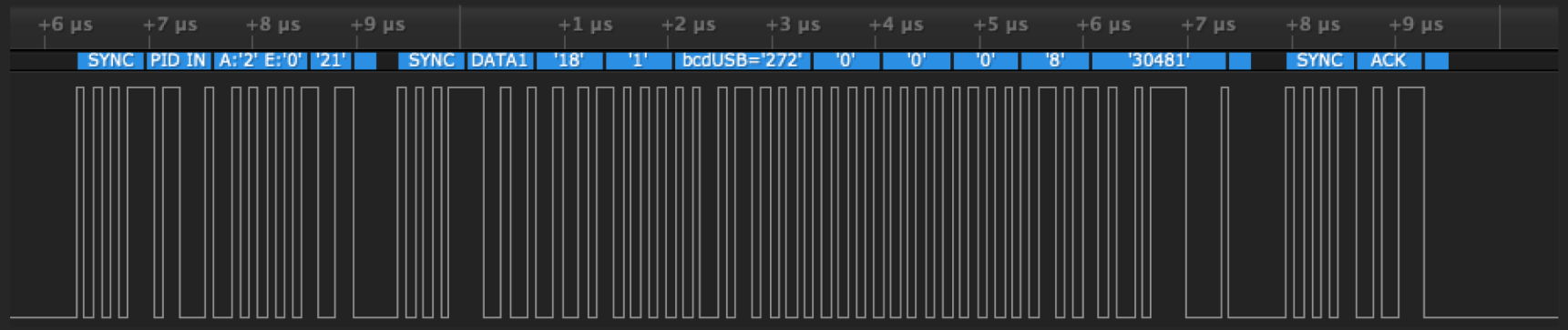
ENUMERATION



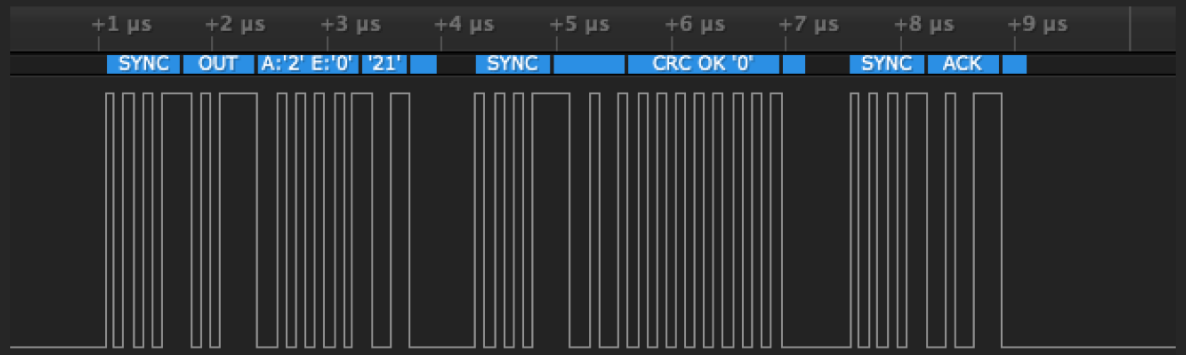
to support enumeration, every USB device must support a standard protocol on EP0

this protocol involves our first formal type of transfer: usb control requests

each control transfer is made up of a number of transactions…



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

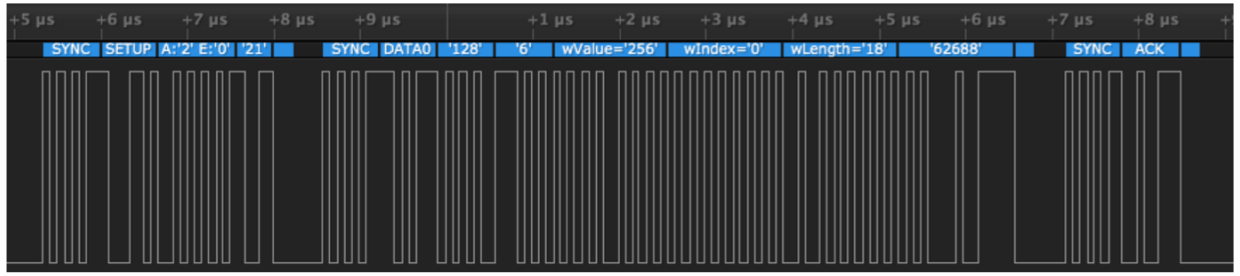


It's often helpful to think of these transactions at a higher level of abstraction…

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	

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	

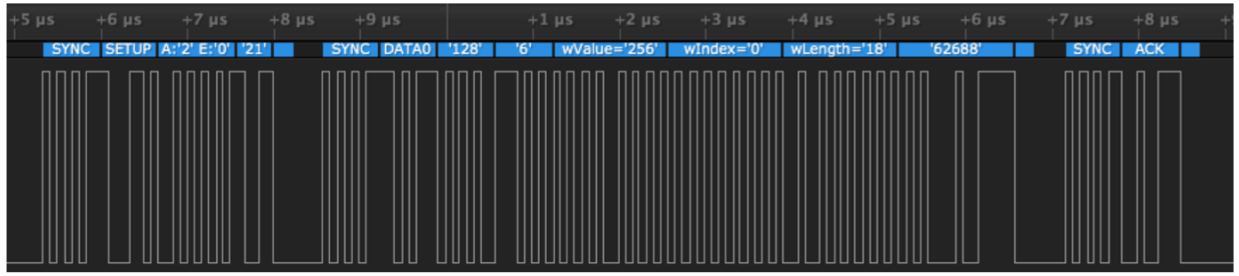
SETUP Data	Radix: auto
bmRequestType.Recipient	Device (0b00000)
bmRequestType.Type	Standard (0b00)
bmRequestType.Direction	Device-to-Host (0b1)
bRequest	Get Descriptor (0x06)
wValue	Device #0 (0x0100)
wIndex	0x0000
wLength	0x0012

SETUP STAGE

Each control transfer begins with a setup stage describing the transactions to be performed.

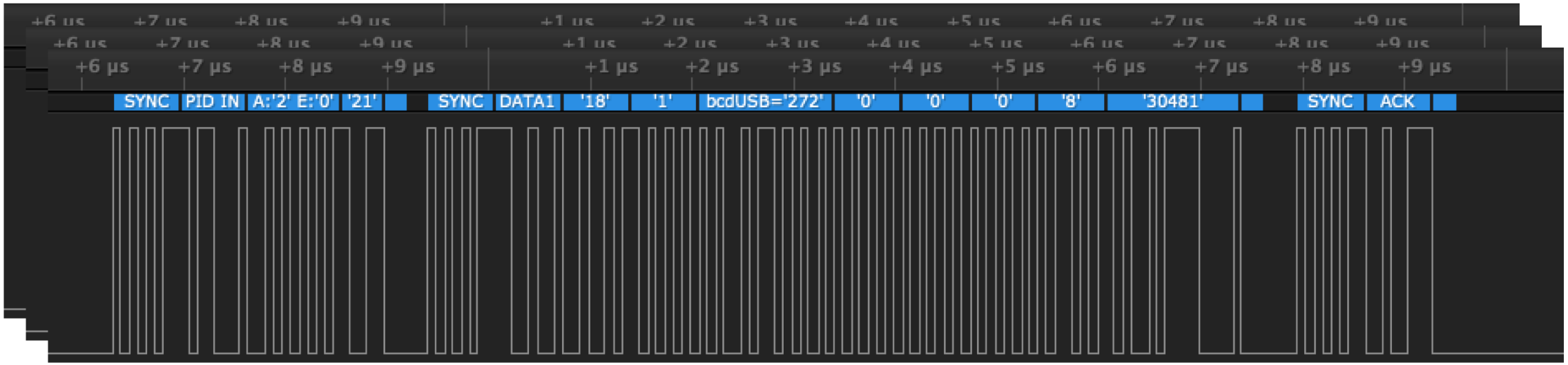
SETUP STAGE

If a device doesn't support a given request, it can STALL the setup stage by issuing a STALL handshake instead of an ACK.



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



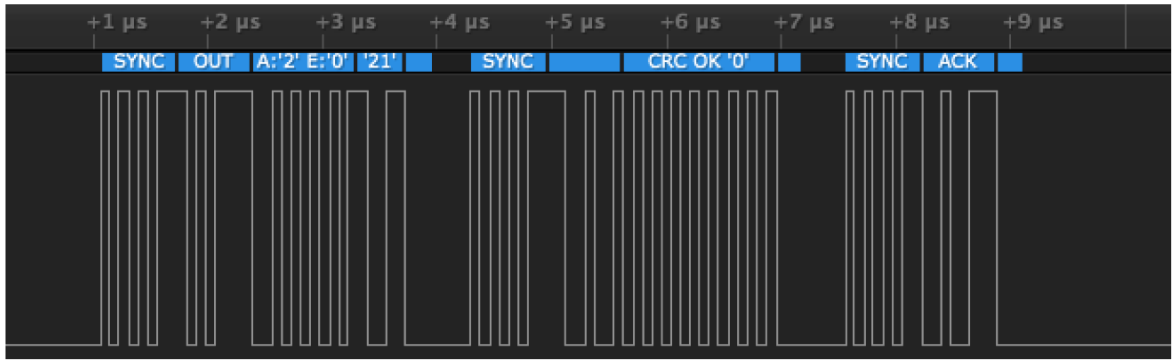
Get Device Descriptor	Index=0 Length=18
SETUP txn	80 06 00 01 00 00 12 00
IN txn [1 POLL]	12 01 10 01 00 00 00 08
IN txn [4 POLL]	03 04 01 60 00 04 01 02
IN txn [1 POLL]	03 01
OUT txn	

If the request has a non-zero length, the transfer has a data stage compromised of one or more IN or OUT data transactions.

The data stage ends when a short packet is received– that is, a packet less than the maximum packet size. (i.e. 8 bytes for LS, 64 bytes for FS)

DATA STAGE

If the device isn't ready for the data stage, it can NAK to buy time to become ready.



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	

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

The status stage is always in the direction opposite the last transaction, to ensure both sides have a chance to ACK.

STATUS STAGE

CONTROL REQUESTS

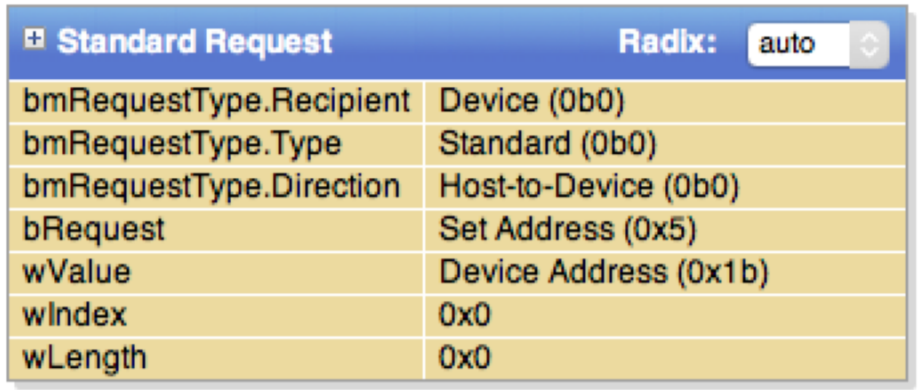
Table 9-2. Format of Setup Data

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 request's data stage, and provides arguments.

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

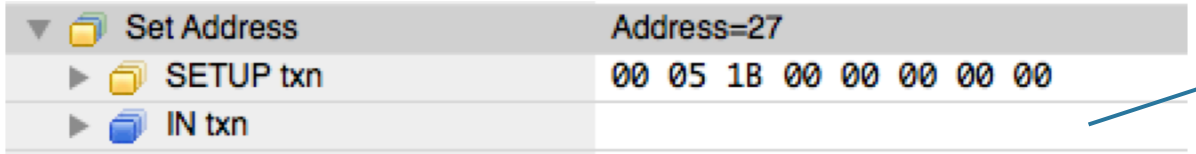
SET_ADDRESS REQUESTS



Standard Request	Radix: auto
bmRequestType.Recipient	Device (0b0)
bmRequestType.Type	Standard (0b0)
bmRequestType.Direction	Host-to-Device (0b0)
bRequest	Set Address (0x5)
wValue	Device Address (0x1b)
wIndex	0x0
wLength	0x0

Every device starts with an address of zero.

A set-address request is one of the first requests commonly issued to a device, and assigns it an address for future communications.



Set Address	Address=27
SETUP txn	00 05 1B 00 00 00 00 00
IN txn	

This request has a length of zero, and thus no data stage. It communicates only via setup arguments:

value = address to be assigned

GET_DESCRIPTOR REQUESTS

A primary goal in enumeration is to allow USB devices to self-describe. Accordingly, devices can provide summaries of themselves and their functions called descriptors.

Device Descriptor	Radix: auto
bLength	18
bDescriptorType	DEVICE (0x01)
bcdUSB	1.10 (0x0110)
bDeviceClass	Defined in Interface (0x00)
bDeviceSubClass	Defined in Interface (0x00)
bDeviceProtocol	Defined in Interface (0x00)
bMaxPacketSize0	8
idVendor	0x0403
idProduct	0x6001
bcdDevice	4.00 (0x0400)
iManufacturer	ftdi (1)
iProduct	test (2)
iSerialNumber	ftE2G0FR (3)
bNumConfigurations	1

There are a multitude of descriptors a device can provide. Some examples:

- device descriptors provide a high-level overview of the device
- string descriptors contain strings that are referenced elsewhere

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	

DEVICE DESCRIPTORS

Device Descriptor	Radix: auto
bLength	18
bDescriptorType	DEVICE (0x01)
bcdUSB	1.10 (0x0110)
bDeviceClass	Defined in Interface (0x00)
bDeviceSubClass	Defined in Interface (0x00)
bDeviceProtocol	Defined in Interface (0x00)
bMaxPacketSize0	8
idVendor	0x0403
idProduct	0x6001
bcdDevice	4.00 (0x0400)
iManufacturer	ftdi (1)
iProduct	test (2)
iSerialNumber	ftE2G0FR (3)
bNumConfigurations	1

The device descriptor provides everything a host needs to get an idea of who the device is:

- Vendor ID
- Product ID
- Device ‘class’ in formation
- References to string names.
- Max size of packets on the control endpoint.