USB Simply Buffered (USB)

Device Enumeration

Environment

- Debian Etch 4.0.
- x86 system.
- Sandisk Cruzer Mini 256 MB USB thumb drive.

USB is a master/slave protocol. The host keeps sending packets for the device, and also sends packets to the device to fill in data, and return it to the host.

The steps involved in USB device enumeration are as follows:

- SetAddress
- GetDescriptor (Device)
- GetDescriptor (Configuration)
- GetDescriptor (Configuration)
- GetDescriptor (String Language Ids)
- GetDescriptor (String 2)
- GetDescriptor (String 1)
- GetDescriptor (Sring 3)
- SetConfiguration

The Sandisk USB thumb drive was connected to a GNU/Linux host PC, and the device enumeration steps were observed. The steps involved are explained below in detail:

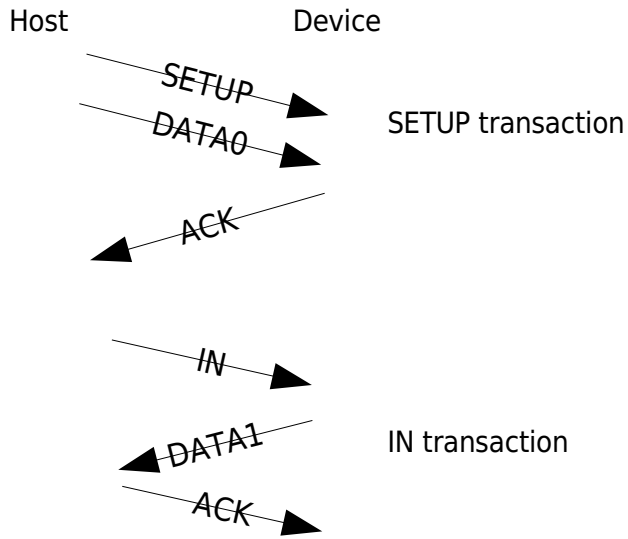
1. SetAddress

At the top-level of USB are the transfers. Transfers consists of transactions, and transactions consist of packets.

SetAddress is an example of a USB transfer. It consists of:

- SETUP transaction (->)
- IN transaction (<-)

-> means the data flow is from host to device.
<- means the data flow is from device to host.



Every transaction has packets going to and from the device.

========== *BEGIN* ==========

What is a USB packet?

The USB packet consists of the following:

	SYNC	PID	!PID	Function Address	Endpoint	Frame Number	Data	CRC
(bits)	8/32	4	4	7	4	11	0-1024	5/16

- SYNC is 8-bits for low-speed/full-speed.
  32-bits for high-speed.

- PID, and !PID in the above are each 4-bits long, as shown below:

	PID 0	PID 1	PID 2	PID 3	!PID 0	!PID 1	!PID 2	!PID 3
(bits)								

- !PID is the ones' complement of PID.
- Function address and Endpoint field constitute a 11-bit address (ADDR) field.
- Frame number (FN) field is 11-bit.
- Data field varies from 0-1024 bytes.
- Token CRCs are 5-bit, while Data CRCs are 16-bit.

What are the types of USB packets?

The different types of packets and their PID values are listed below:

PID Type	PID Name	PID <3:0>	Hex	!Hex	PID <0:3>	Packets
Token	OUT	0001	1	E	1000	PID ADDR CRC
	IN	1001	9	6	1001	PID ADDR CRC
	SOF	0101	5	A	1010	PID FN CRC
	SETUP	1101	D	2	1011	PID ADDR CRC
Data	DATA0	0011	3	C	1100	PID DATA CRC
	DATA1	1011	B	4	1101	PID DATA CRC
	DATA2	0111	7	8	1110	PID DATA CRC
	MDATA	1111	F	0	1111	PID DATA CRC
Handshake	ACK	0010	2	D	0100	PID
	NAK	1010	A	5	0101	PID
	STALL	1110	E	1	0111	PID
	NYET	0110	6	9	0110	PID
Special	PRE	1100	C	3	0011	
	ERR	1100	C	3	0011	
	SPLIT	1000	8	7	0001	
	PING	0100	4	B	0010	
	Reserved	0000	0	F	0000	

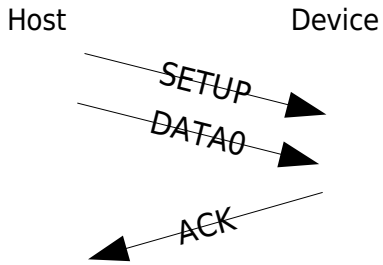
Reference: Chapter 8: Protocol Layer, pages 195-199.
========== END ==========

Let us see each transaction in detail.

1.1 SETUP transaction

The SETUP transaction has the following three packets:

- SETUP packet (->)
- DATA0 packet (->)
- ACK packet (<-)



1.1.1 SETUP packet

A SETUP packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1011	0100	0000 000	0000	01000
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

00010 0000 000 0000 0010 1101

Regrouping in groups of 4-bits:

0001 0000 0000 0000 0010 1101

Hex values:

1 0 0 0 2 D

Regrouping as a byte:

M L

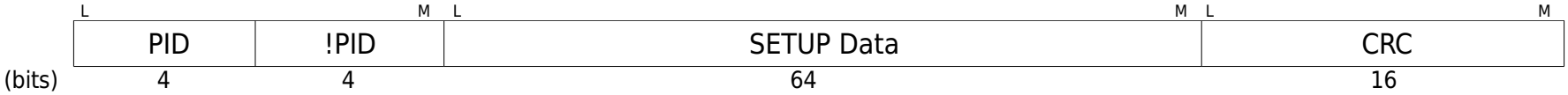
10 00 2D

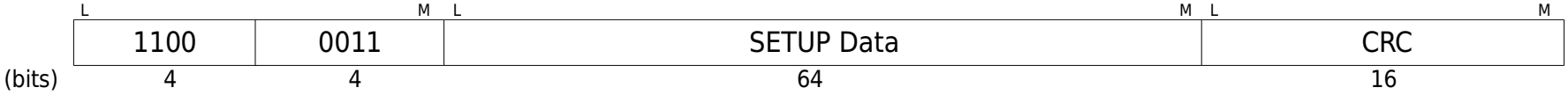
Summary (SETUP packet):

Offset	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
	0	1	0	1
00	00101101	00000000	2D	00
02	00010000		10	

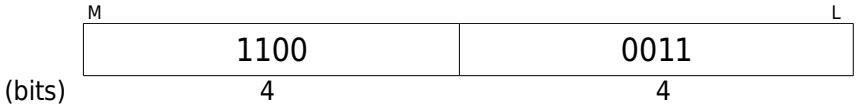
1.1.2 DATA0 packet

A DATA0 packet consists of:





The PID arranged in MSB to LSB order:



Hex values (PID):

Since this is for a SETUP packet, the data consists of 8 bytes.

========== *BEGIN* ==========

Format of SETUP packet data

A SETUP packet consists of:

	L M	L M	L M	L M	L M
	bmRequestType	bRequest	wValue	wIndex	wLength
(bytes)	1	1	2	2	2

- bmRequestType has a bitmap value with the following fields:

	D7	D6 D5	D4 D3 D2 D1 D0
(bits)	1	2	5

D7: Data transfer direction
0 = Host-to-device
1 = Device-to-host

D6..D5: Type
0 = Standard
1 = Class
2 = Vendor
3 = Reserved

D4..D0: Recipient
0 = Device
1 = Interface
2 = Endpoint
3 = Other
4..31 = Reserved

- bRequest follows Table 9-4, Standad Request Codes (page 251).

bRequest	Value
GET_STATUS	0
CLEAR_FEATURE	1
Reserved	2
SET_FEATURE	3
Reserved	4
SET_ADDRESS	5
GET_DESCRIPTOR	6
SET_DESCRIPTOR	7
GET_CONFIGURATION	8
SET_CONFIGURATION	9
GET_INTERFACE	10
SET_INTERFACE	11
SYNCH_FRAME	12

- wValue field follows Table 9-3, Standard Device Requests (page 250).

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000 00000001 00000010	CLEAR_FEATURE	Feature Selector	Zero Interface Endpoint	0	None
10000000	GET_CONFIGURATION	0	0	1	Configuration Value
10000000	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	0 or Language ID	Descriptor Length	Descriptor
10000001	GET_INTERFACE	0	Interface	1	Alternate Interface
10000000	GET_STATUS	0	Zero Interface Endpoint	2	Device, Interface, or Endpoint status
00000000	SET_ADDRESS	Device Address	0	0	None
00000000	SET_CONFIGURATION	Configuration Value	0	0	None
00000000	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	0 or Language ID	Descriptor Length	Descriptor
00000000 00000001 00000010	SET_FEATURE	Feature Selector	Zero Interface Endpoint	0	None
00000001	SET_INTERFACE	Alternate	Interface	0	None
10000001	SYNCH_FRAME	0	Endpoint	2	Frame Number

- The Descriptor Index is in the lower byte, and used only for Configuration and String descriptors.

- The descriptor types are in Table 9-5 (page 251):

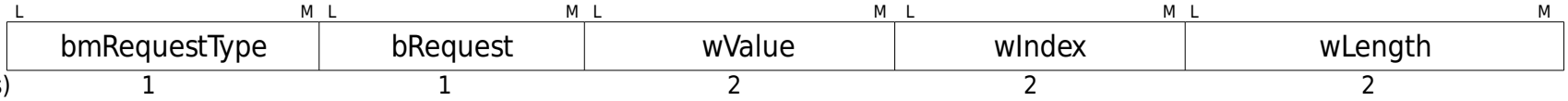
Descriptor Type	Value
DEVICE	1
CONFIGURATION	2
STRING	3
INTERFACE	4
ENDPOINT	5
DEVICE_QUALIFIER	6
OTHER_SPEED_CONFIGURATION	7
INTERFACE_POWER	8

- Standard feature selectors are in Table 9-6 (page 280):

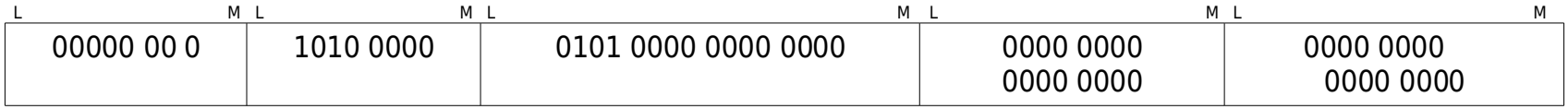
Feature Selector	Recipient	Value
DEVICE_REMOTE_WAKEUP	Device	1
ENDPOINT_HALT	Endpoint	0
TEST_MODE	Device	2

Reference: Chapter 9: USB Device Framework, pages 250-251, 280
========== *END* ==========

So, the SETUP packet consists of:



The 8 bytes for SETUP details are as follows:



bmRequestType:

D7: Data transfer direction
0 = Host-to-device

D6...D5: Type
0 = Standard

D4...D0: Recipient
0 = Device

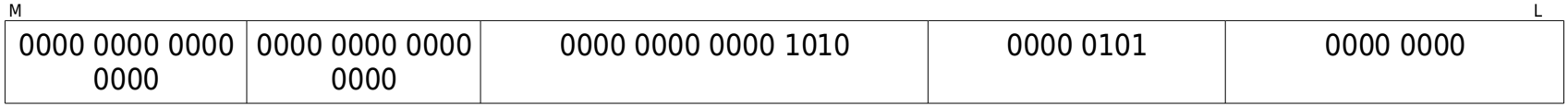
bRequest:

SET_ADDRESS request code is 5.

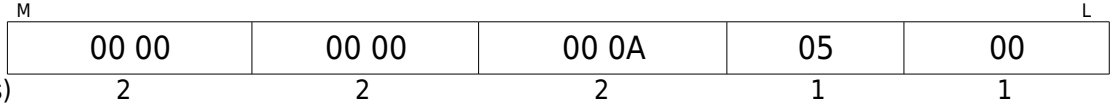
wValue:

Address to be assigned to device is 10 for SET_ADDRESS.

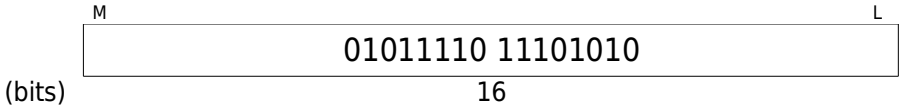
Putting it in MSB to LSB order:



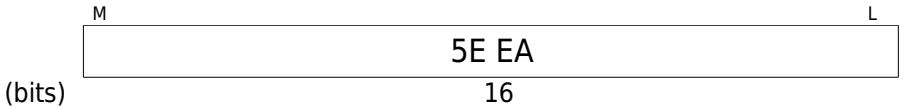
In Hex:



The CRC observed in the sample capture is:



CRC (in Hex):



Putting it all (PID + SETUP DATA + CRC) together,

Summary (DATA0 packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11000011	00000000	C3	00
02	00000101	00001010	05	0A
04	00000000	00000000	00	00
06	00000000	00000000	00	00
08	00000000	11101010	00	EA
10	01011110		5E	

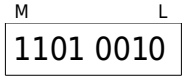
1.1.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

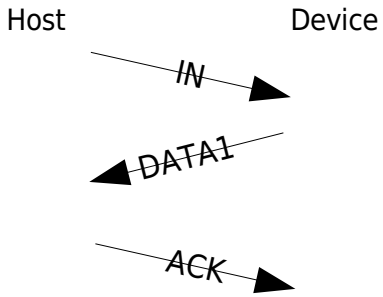
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

1.2 IN transaction

The IN transaction has the following three packets:

- IN packet (->)
- DATA1 packet (<-)
- ACK packet (->)



1.2.1 IN packet

A IN packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1001	0110	0000 000	0000	01000
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

00010 0000 000 0000 0110 1001

Regrouping in groups of 4-bits:

0001 0000 0000 0000 0110 1001

Hex values (as a byte):

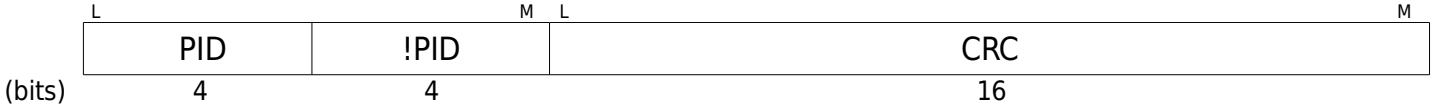
10 00 69

Summary (IN packet):

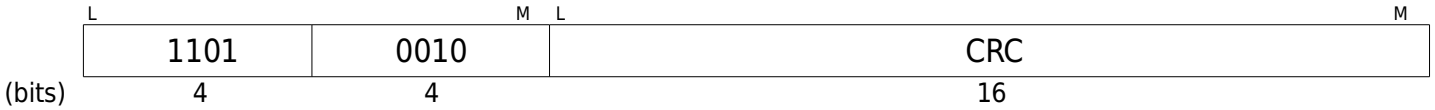
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01101001	00000000	69	00
02	00010000		10	

1.2.2 DATA1 packet

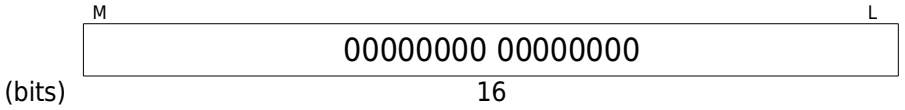
A DATA1 packet for SetAddress has no data. It consists of:



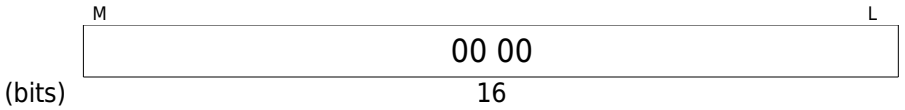
Values:



The CRC observed in the sample capture is:

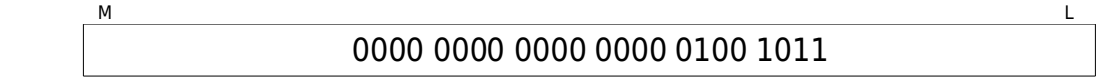


CRC (in Hex):

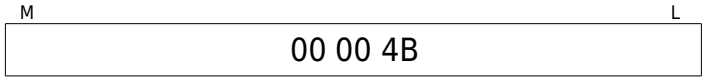


The packet arranged in MSB to LSB order:

In groups of 4 bits:



In Hex,



Summary (DATA1 packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00000000	4B	00
02	00000000		00	

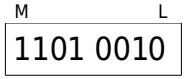
1.2.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

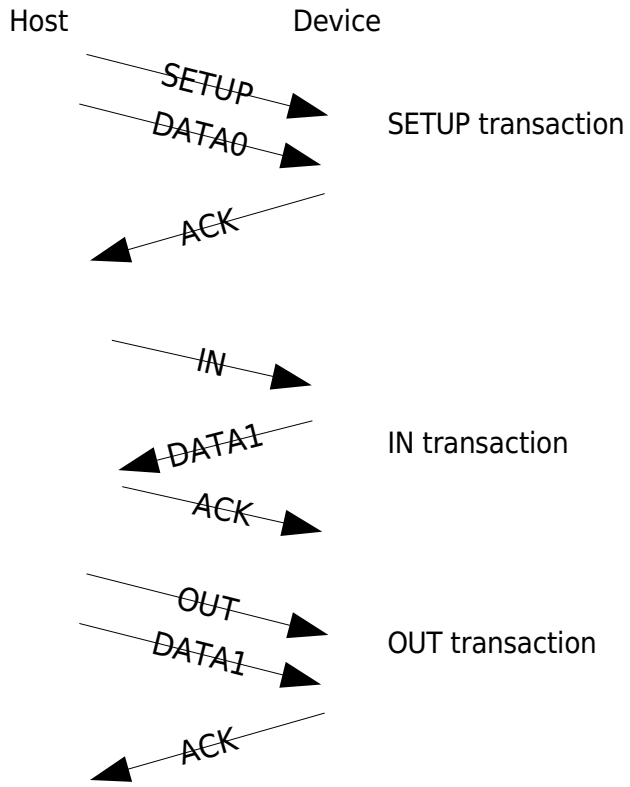
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

2. GetDescriptor (Device)

GetDescriptor (Device) transfer consists of the following transactions:

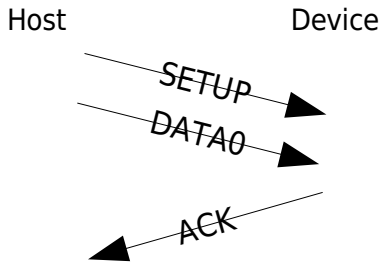
- SETUP transaction (->)
- IN transaction (<-)
- OUT transaction (->)



2.1 SETUP transaction

The SETUP transaction has the following three packets:

- SETUP packet (->)
- DATA0 packet (->)
- ACK packet (<-)



2.1.1 SETUP packet

A SETUP packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1011	0100	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

11011 0000 000 1010 0010 1101

Regrouping in groups of 4-bits:

1101 1000 0000 1010 0010 1101

Hex values:

D 8 0 A 2 D

Regrouping as a byte:

M L

D8 0A 2D

Summary (SETUP packet):

Offset	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
	0	1	0	1
00	00101101	00001010	2D	0A
02	11011000		D8	

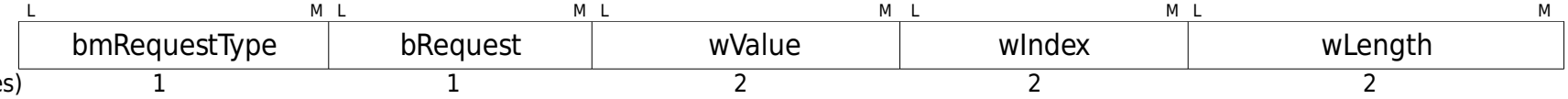
2.1.2 DATA0 packet

A DATA0 packet consists of:

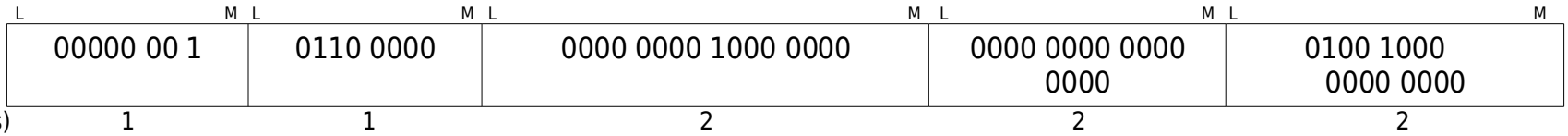
The PID arranged in MSB to LSB order:

Hex values (PID):

Since this is for a SETUP packet, the data consists of 8 bytes. The format is:



The 8 bytes for SETUP details are as follows (in decimal):



bmRequestType:

D7: Data transfer direction
1 = Device-to-host

D6...D5: Type
0 = Standard

D4...D0: Recipient
0 = Device

bRequest:

GET_DESCRIPTOR request code is 6.

wValue:

Descriptor type is DEVICE, the value is 1. Descriptor index is 0.

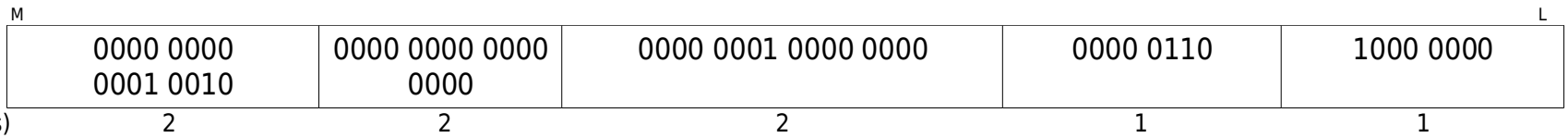
wIndex:

Value is zero.

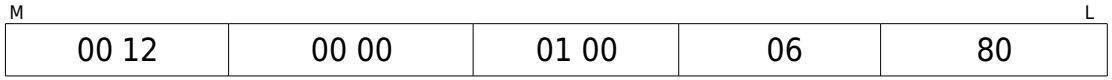
wLength:

Size of descriptor data, which is 18 bytes.

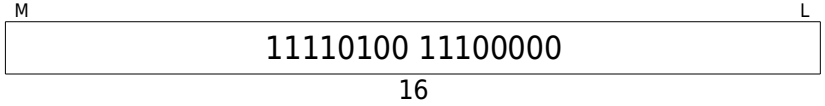
Putting it in MSB to LSB order:



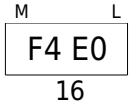
In Hex:



The CRC observed in the sample capture is:



CRC (in Hex):



Putting it all (PID + SETUP DATA + CRC) together,

Summary (DATA0 packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11000011	10000000	C3	80
02	00000110	00000000	06	00
04	00000001	00000000	01	00
06	00000000	00010010	00	12
08	00000000	11100000	00	E0
10	11110100		F4	

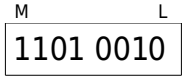
2.1.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

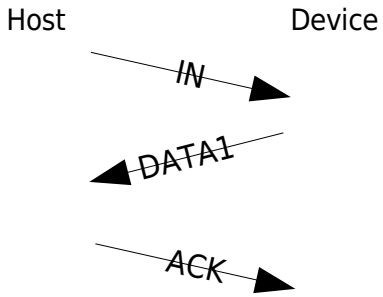
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

2.2 IN transaction

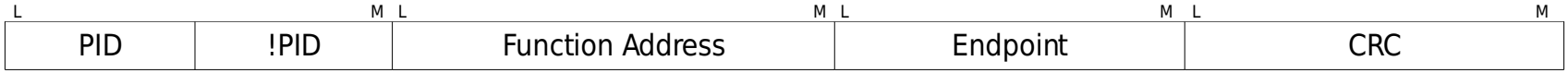
The IN transaction has the following three packets:

- IN packet (->)
- DATA1 packet (<-)
- ACK packet (->)

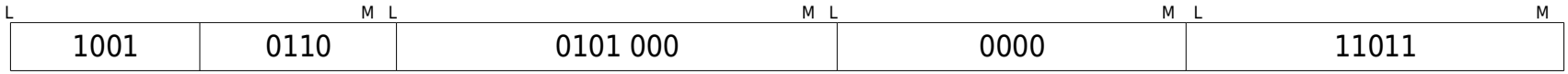


2.2.1 IN packet

A IN packet consists of:

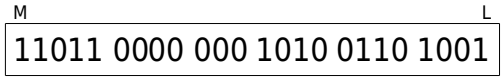


Values:

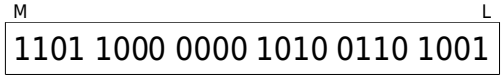


USB is little-endian. LSB goes out of the wire, first.

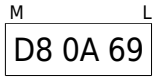
Displaying from MSB to LSB:



Regrouping in groups of 4-bits:



Hex values (as a byte):



Summary (IN packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01101001	00001010	69	0A
02	11011000		D8	

2.2.2 DATA1 packet

A DATA1 packet for GetDescriptor (Device) provides the 18 bytes of data. It consists of:

Values:

========== *BEGIN* ==========
Standard Device Descriptor

Table 9-8: Standard Device Descriptor

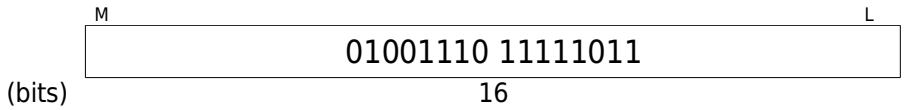
Offset	Field	Size	Value	Description
0	bLength	1	Number	Length of this descriptor (bytes)
1	bDescriptorType	1	Constant	DEVICE descriptor type
2	bcdUSB	2	BCD	USB Specification Release Number in BCD format (210H)
4	bDeviceClass	1	Class	Class code (by USB-IF)
5	bDeviceSubClass	1	SubClass	Subclass code (by USB-IF)
6	bDeviceProtocol	1	Protocol	Protocol code (by USB-IF)
7	bMaxPacketSize0	1	Number	Maximum packet size for endpoint zero (8, 16, 32 or 64)
8	idVendor	2	ID	Vendor ID (by USB-IF)
10	idProduct	2	ID	Product ID (by manufacturer)
12	bcdDevice	2	BCD	Device release number in BCD
14	iManufacturer	1	Index	Index of string descriptor describing manufacturer
15	iProduct	1	Index	Index of string descriptor describing product
16	iSerialNumber	1	Index	Index of string descriptor describing the device's serial number
17	bNumConfigurations	1	Number	Number of possible configurations

Reference: Chapter 9: USB Device Framework, pages 262-263
========== *END* ==========

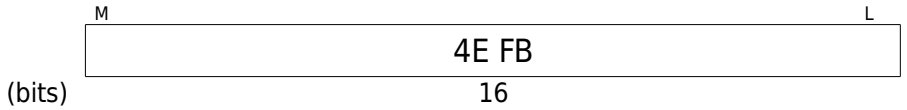
So, the data for our capture was:

Offset	Field	Size	Value	Hex
0	bLength	1	18 (bytes)	0x12
1	bDescriptorType	1	DEVICE descriptor (1)	0x01
2	bcdUSB	2	2.0	0x0200
4	bDeviceClass	1	Class defined at interface level	0x00
5	bDeviceSubClass	1	SubClass defined at interface level	0x00
6	bDeviceProtocol	1	None	0x00
7	bMaxPacketSize0	1	64	0x40
8	idVendor	2	SanDisk Corporation	0x0781
10	idProduct	2	SDCZ2 Cruzer Mini Flash Drive (thin)	0x5150
12	bcdDevice	2	0.1	0x0010
14	iManufacturer	1	1 (“SanDisk Corporation”)	0x01
15	iProduct	1	2 (“Cruzer Mini”)	0x02
16	iSerialNumber	1	3 (“SNDK40E1641758600106”)	0x03
17	bNumConfigurations	1	1	0x01

The CRC observed in the sample capture is:



CRC (in Hex):



Putting it together (PID + 18 bytes of data + CRC):

	Binary (M…L)		Hexadecimal (M…L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00010010	4B	12
02	00000001	00000000	01	00
04	00000010	00000000	02	00
06	00000000	00000000	00	00
08	01000000	10000001	40	81
10	00000111	01010000	07	50
12	01010001	00010000	51	10
14	00000000	00000001	00	01
16	00000010	00000011	02	03
18	00000001	11111011	01	FB
20	01001110		4E	

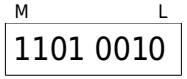
2.2.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

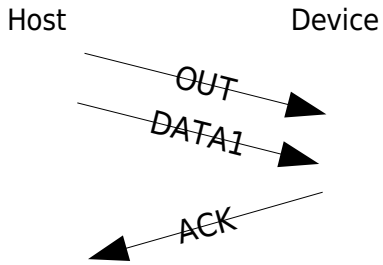
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

2.3 OUT transaction

The OUT transaction has the following three packets:

- OUT packet (->)
- DATA1 packet (->)
- ACK packet (<-)



2.3.1 OUT packet

An OUT packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1000	0111	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

11011 0000 000 1010 1110 0001

Regrouping in groups of 4-bits:

1101 1000 0000 1010 1110 0001

Hex values (as a byte):

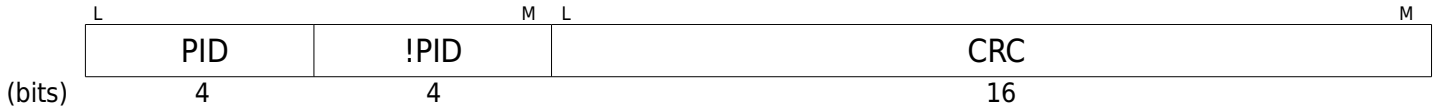
D8 0A E1

Summary (OUT packet):

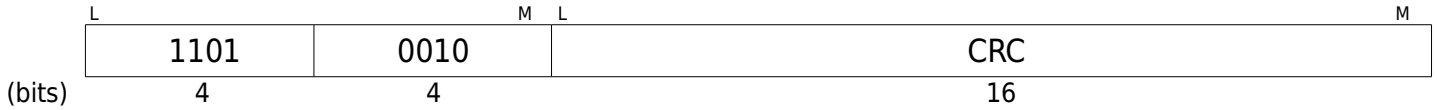
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11100001	00001010	E1	0A
02	11011000		D8	

2.3.2 DATA1 packet

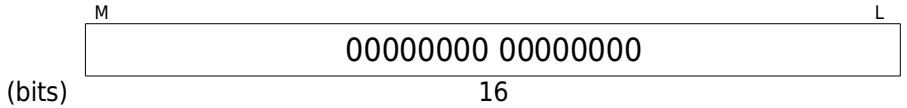
The DATA1 packet has no data. It consists of:



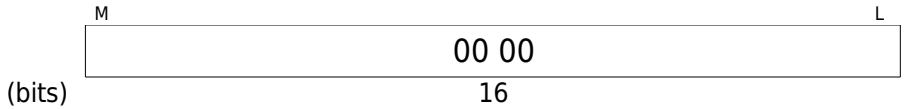
Values:



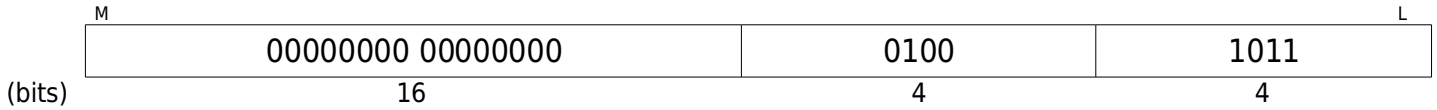
The CRC observed in the sample capture is:



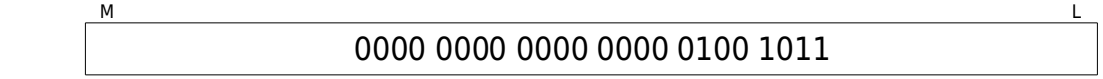
CRC (in Hex):



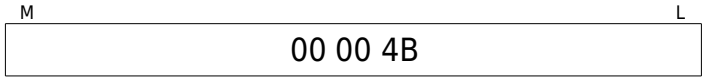
The packet arranged in MSB to LSB order:



In groups of 4 bits:



In Hex,



Summary (DATA1 packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00000000	4B	00
02	00000000		00	

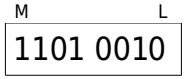
2.3.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

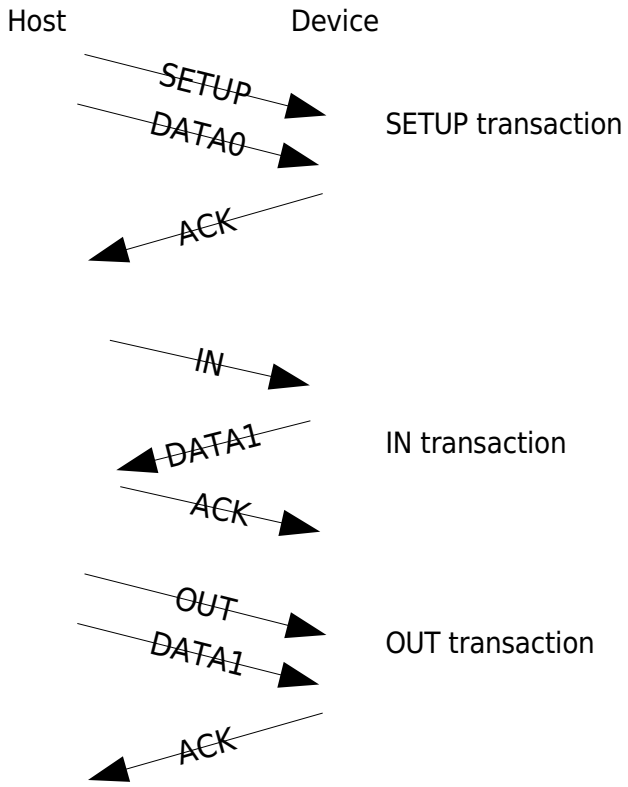
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

3. GetDescriptor (Configuration)

GetDescriptor (Configuration) transfer consists of the following transactions:

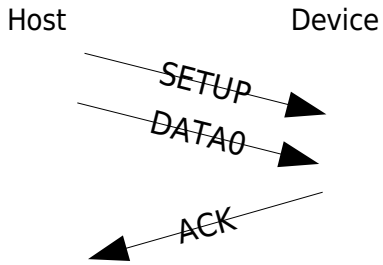
- SETUP transaction (->)
- IN transaction (<-)
- OUT transaction (->)



3.1 SETUP transaction

The SETUP transaction has the following three packets:

- SETUP packet (->)
- DATA0 packet (->)
- ACK packet (<-)



3.1.1 SETUP packet

A SETUP packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1011	0100	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

```
11011 0000 000 1010 0010 1101
```

Regrouping in groups of 4-bits:

```
1101 1000 0000 1010 0010 1101
```

Hex values:

```
D 8 0 A 2 D
```

Regrouping as a byte:

M L

D8 0A 2D

Summary (SETUP packet):

Offset	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
	0	1	0	1
00	00101101	00001010	2D	0A
02	11011000		D8	

3.1.2 DATA0 packet

A DATA0 packet consists of:

	PID	!PID	SETUP Data	CRC
(bits)	4	4	64	16

	1100	0011	SETUP Data	CRC
(bits)	4	4	64	16

The PID arranged in MSB to LSB order:

	1100	0011
(bits)	4	4

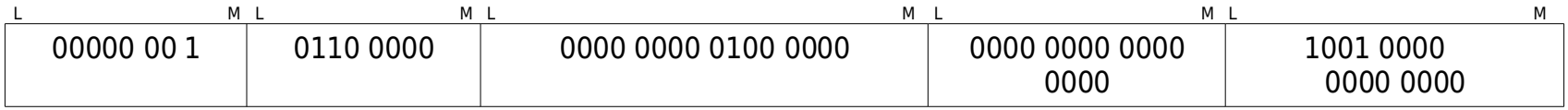
Hex values (PID):

C 3

Since this is for a SETUP packet, the data consists of 8 bytes. The format is:

	bmRequestType	bRequest	wValue	wIndex	wLength
(bytes)	1	1	2	2	2

The 8 bytes for SETUP details are as follows (in decimal):



bmRequestType:

D7: Data transfer direction
1 = Device-to-host

D6...D5: Type
0 = Standard

D4...D0: Recipient
0 = Device

bRequest:

GET_DESCRIPTOR request code is 6.

wValue:

Descriptor type is CONFIGURATION, the value is 2. Descriptor index is 0.

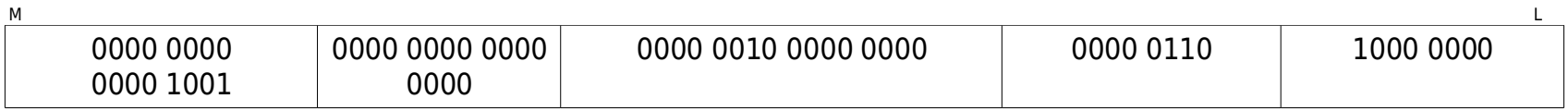
wIndex:

Value is zero.

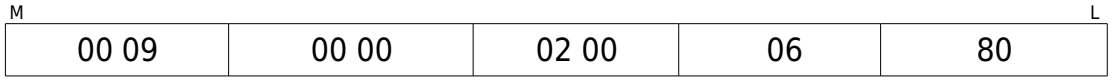
wLength:

Size of descriptor data, which is 9 bytes.

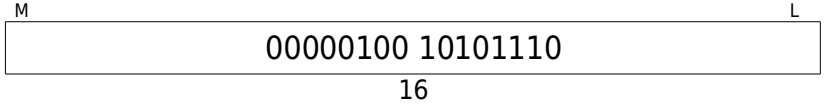
Putting it in MSB to LSB order:



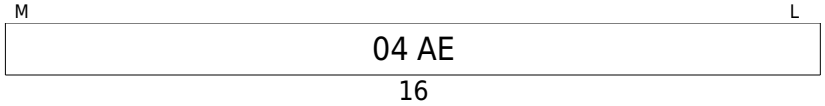
In Hex:



The CRC observed in the sample capture is:



CRC (in Hex):



Putting it all (PID + SETUP DATA + CRC) together,

Summary (DATA0 packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11000011	10000000	C3	80
02	00000110	00000000	06	00
04	00000010	00000000	02	00
06	00000000	00001001	00	09
08	00000000	10101110	00	AE
10	00000100		04	

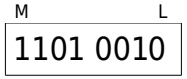
3.1.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

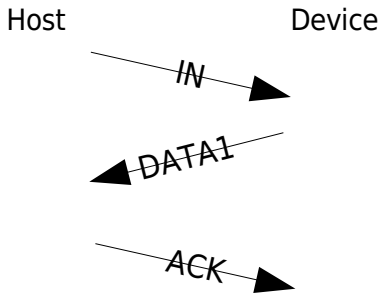
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

3.2 IN transaction

The IN transaction has the following three packets:

- IN packet (->)
- DATA1 packet (<-)
- ACK packet (->)



3.2.1 IN packet

A IN packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
	L M	L M	L M	L M	L M
(bits)	4	4	7	4	5

Values:

	1001	0110	0101 000	0000	11011
	L M	L M	L M	L M	L M
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

M L

11011 0000 000 1010 0110 1001

Regrouping in groups of 4-bits:

M L

1101 1000 0000 1010 0110 1001

Hex values (as a byte):

M L

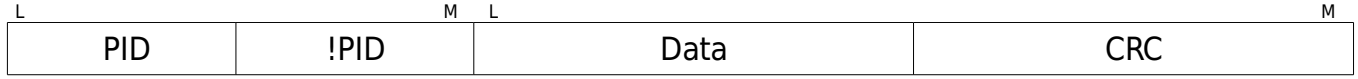
D8 0A 69

Summary (IN packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01101001	00001010	69	0A
02	11011000		D8	

3.2.2 DATA1 packet

A DATA1 packet for GetDescriptor (Configuration) provides the 9 bytes of data. It consists of:



(bits)	4	4	72	16

Values:

(bits)	4	4	72	16

========== *BEGIN* ==========
Standard Configuration Descriptor

Table 9-10: Standard Configuration Descriptor

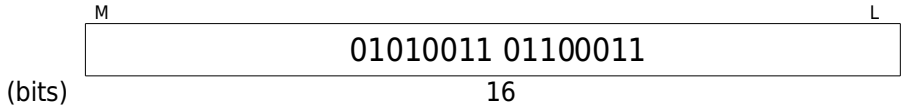
Offset	Field	Size	Value	Description
0	bLength	1	Number	Length of this descriptor (bytes)
1	bDescriptorType	1	Constant	CONFIGURATION descriptor type
2	wTotalLength	2	Number	Total length of all descriptors (configuration, interface, endpoint, class- or vendor- specific) for this configuration.
4	bNumInterfaces	1	Number	Number of interfaces
5	bConfigurationValue	1	Number	Value to be used as argument to SetConfiguration() to select this configuration
6	iConfiguration	1	Index	Index of string descriptor describing this configuration
7	bmAttributes	1	Bitmap	D7: Reserved (set to one) D6: Self-powered D5: Remote wake-up D4...0: Reserved (set to zero) Enable field by setting bit to one.
8	bMaxPower	1	mA	Maximum power consumption expressed in 2 mA units.

Reference: Chapter 9: USB Device Framework, pages 265-266
========== *END* ==========

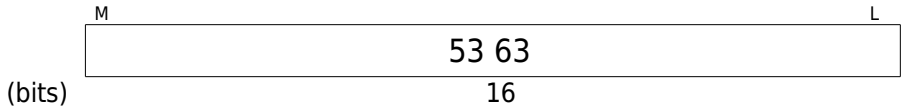
So, the data for our capture was:

Offset	Field	Size	Value	Hex
0	bLength	1	9 (bytes)	0x09
1	bDescriptorType	1	CONFIGURATION descriptor (2)	0x02
2	wTotalLength	2	32 (bytes)	0x0020
4	bNumInterfaces	1	1	0x01
5	bConfigurationValue	1	1	0x01
6	iConfiguration	1	0	0x00
7	bmAttributes	1	Not supported	0x80
8	bMaxPower	2	200 mA	0x64

The CRC observed in the sample capture is:



CRC (in Hex):



Putting it together (PID + 9 bytes of data + CRC):

	Binary (M…L)		Hexadecimal (M…L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00001001	4B	09
02	00000010	00100000	02	20
04	00000000	00000001	00	01
06	00000001	00000000	01	00
08	10000000	01100100	80	64
10	01100011	01010011	63	53

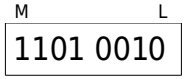
3.2.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

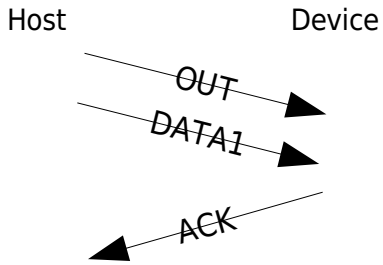
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

3.3 OUT transaction

The OUT transaction has the following three packets:

- OUT packet (->)
- DATA1 packet (->)
- ACK packet (<-)



3.3.1 OUT packet

An OUT packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1000	0111	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

```
11011 0000 000 1010 1110 0001
```

Regrouping in groups of 4-bits:

```
1101 1000 0000 1010 1110 0001
```

Hex values (as a byte):

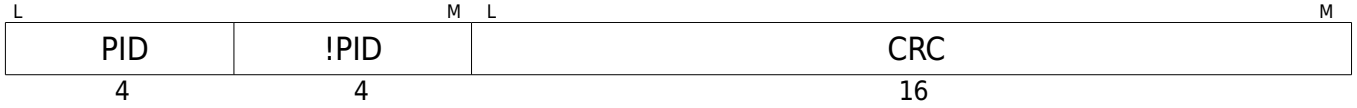
```
D8 0A E1
```

Summary (OUT packet):

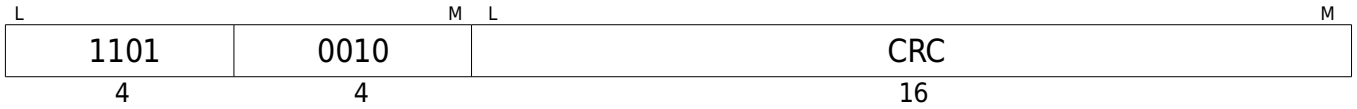
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11100001	00001010	E1	0A
02	11011000		D8	

3.3.2 DATA1 packet

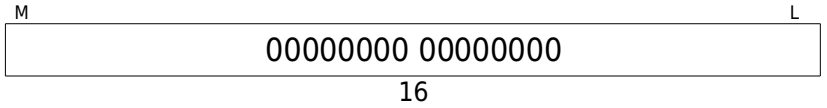
The DATA1 packet has no data. It consists of:



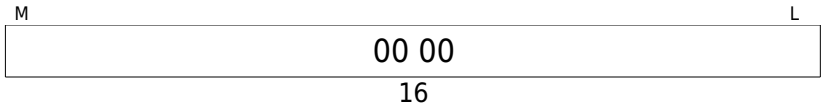
Values:



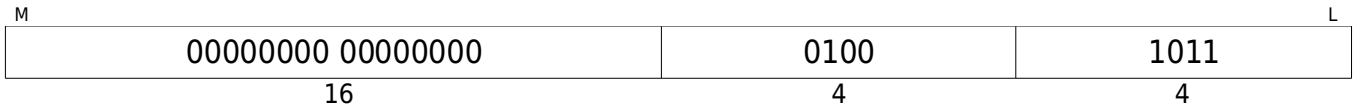
The CRC observed in the sample capture is:



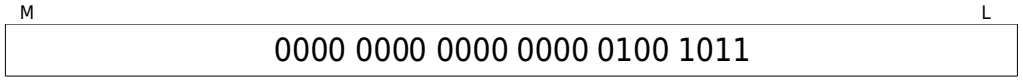
CRC (in Hex):



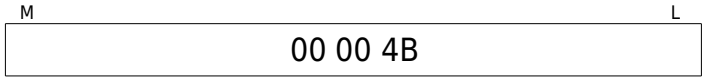
The packet arranged in MSB to LSB order:



In groups of 4 bits:



In Hex,



Summary (DATA1 packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00000000	4B	00
02	00000000		00	

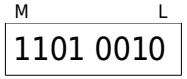
3.3.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

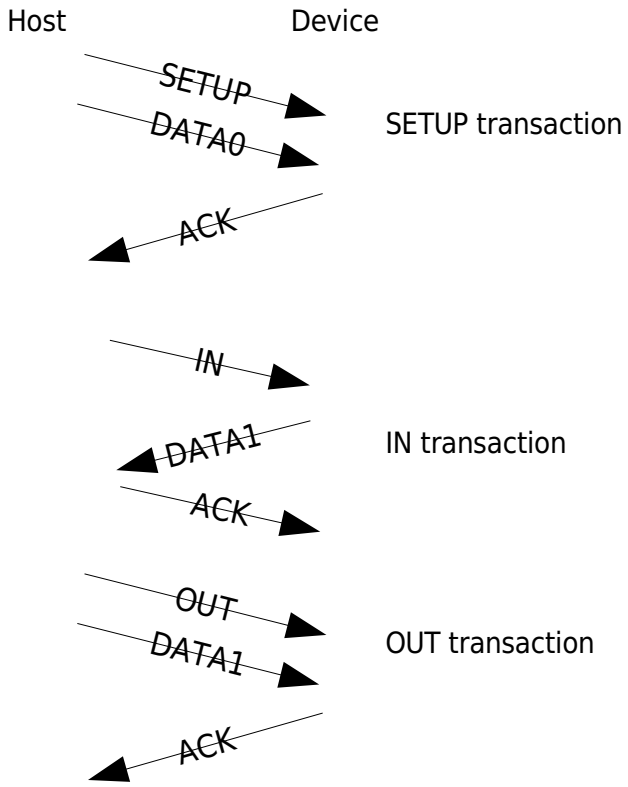
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

4. GetDescriptor (Configuration)

GetDescriptor (Configuration) transfer consists of the following transactions:

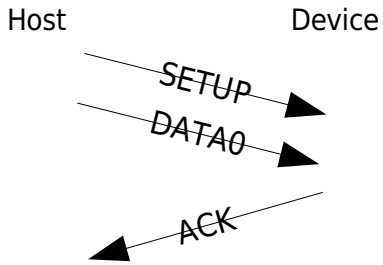
- SETUP transaction (->)
- IN transaction (<-)
- OUT transaction (->)



4.1 SETUP transaction

The SETUP transaction has the following three packets:

- SETUP packet (->)
- DATA0 packet (->)
- ACK packet (<-)



4.1.1 SETUP packet

A SETUP packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1011	0100	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

```
11011 0000 000 1010 0010 1101
```

Regrouping in groups of 4-bits:

```
1101 1000 0000 1010 0010 1101
```

Hex values:

```
D 8 0 A 2 D
```

Regrouping as a byte:

M L

D8 0A 2D

Summary (SETUP packet):

Offset	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
	0	1	0	1
00	00101101	00001010	2D	0A
02	11011000		D8	

4.1.2 DATA0 packet

A DATA0 packet consists of:

	PID	!PID	SETUP Data	CRC
(bits)	4	4	64	16

	1100	0011	SETUP Data	CRC
(bits)	4	4	64	16

The PID arranged in MSB to LSB order:

	1100	0011
(bits)	4	4

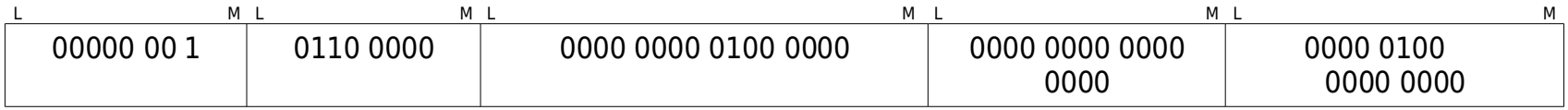
Hex values (PID):

C 3

Since this is for a SETUP packet, the data consists of 8 bytes. The format is:

	bmRequestType	bRequest	wValue	wIndex	wLength
(bytes)	1	1	2	2	2

The 8 bytes for SETUP details are as follows (in decimal):



bmRequestType:

D7: Data transfer direction
1 = Device-to-host

D6...D5: Type
0 = Standard

D4...D0: Recipient
0 = Device

bRequest:

GET_DESCRIPTOR request code is 6.

wValue:

Descriptor type is CONFIGURATION, the value is 2. Descriptor index is 0.

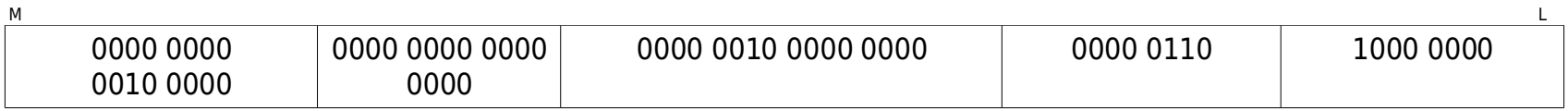
wIndex:

Value is zero.

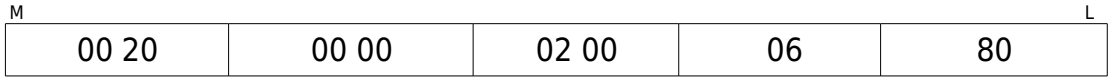
wLength:

Size of descriptor data, which is 32 bytes.

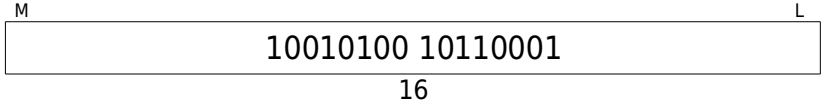
Putting it in MSB to LSB order:



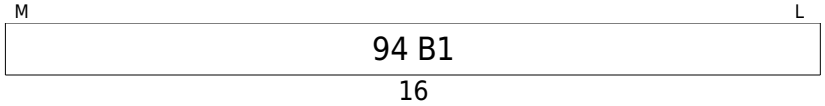
In Hex:



The CRC observed in the sample capture is:



CRC (in Hex):



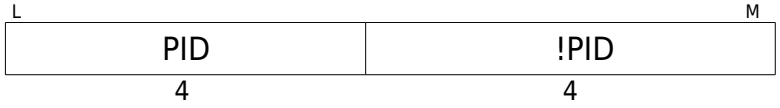
Putting it all (PID + SETUP DATA + CRC) together,

Summary (DATA0 packet):

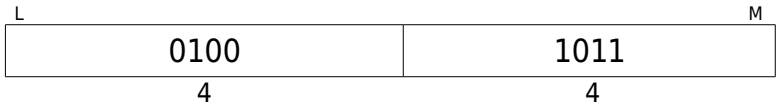
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11000011	10000000	C3	80
02	00000110	00000000	06	00
04	00000010	00000000	02	00
06	00000000	00100000	00	20
08	00000000	10110001	00	B1
10	10010100		94	

4.1.3 ACK packet

An ACK packet consists of:

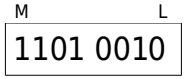


Values:



USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

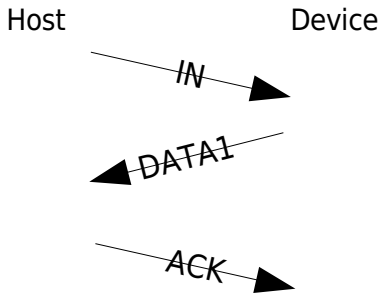
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

4.2 IN transaction

The IN transaction has the following three packets:

- IN packet (->)
- DATA1 packet (<-)
- ACK packet (->)



4.2.1 IN packet

A IN packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1001	0110	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

11011 0000 000 1010 0110 1001

Regrouping in groups of 4-bits:

1101 1000 0000 1010 0110 1001

Hex values (as a byte):

D8 0A 69

Summary (IN packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01101001	00001010	69	0A
02	11011000		D8	

4.2.2 DATA1 packet

A DATA1 packet for GetDescriptor (Configuration) provides the 32 bytes of data. It consists of:

	L … M	L … M	
	PID	!PID	Data	CRC
(bits)	4	4	256	16

Values:

	L … M	L … M	
	1101	0010	Data	CRC
(bits)	4	4	256	16

========== BEGIN ==========
Standard Interface Descriptor

Table 9-12: Standard Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Length of this descriptor (bytes)
1	bDescriptorType	1	Constant	INTERFACE descriptor type
2	bInterfaceNumber	1	Number	Number of this interface
3	bAlternateSetting	1	Number	Used to select this alternate setting for the interface identified in the prior field
4	bNumEndpoints	1	Number	Number of endpoints for this interface (excluding endpoint zero). If zero, it uses only endpoint zero.
5	bInterfaceClass	1	Number	Class code (by USB-IF)
6	bInterfaceSubClass	1	Number	Subclass code (by USB-IF)
7	bInterfaceProtocol	1	Protocol	Protocol code (by USB-IF)
8	iInterface	1	Index	Index of string descriptor describing this interface

Table 9-13: Standard Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Length of this descriptor (bytes)
1	bDescriptorType	1	Constant	ENDPOINT descriptor type
2	bEndpointAddress	1	Endpoint	Bit:7 Direction 0 = OUT endpoint 1 = IN endpoint Bit: 6...4 Reserved (set to zero) Bit: 3...0 Endpoint number
3	bmAttributes	1	Bitmap	Bit: 7...6 Reserved (set to zero) Bit: 5...4 Usage type 00 = Data endpoint 01 = Feedback endpoint 10 = Implicit feedback data endpoint 11 = Reserved Bit: 3...2 Synchronization type 00 = No synchronization 01 = Asynchronous 10 = Adaptive 11 = Synchronous Bit: 1...0: Transfer type 00 = Control 01 = Isochronous 10 = Bulk 11 = Interrupt
4	wMaxPacketSize	2	Number	Bit: 15...13 Reserved (set to zero) Bit: 12...11 Additional transaction per microframe 00 = None 01 = 1 additional 10 = 2 additional 11 = Reserved Bit: 10...0 Maxium packet size
6	bInterval	1	Number	Interval for polling endpoint for data transfers. Expressed in frames or microframes.

Reference: Chapter 9: USB Device Framework, pages 268-271
========== *END* ==========

So, the data for our capture was one configuration descriptor, one interface descriptor and two endpoint descriptors:

Configuration descriptor (9 bytes):

Offset	Field	Size	Value	Hex
0	bLength	1	9 (bytes)	0x09
1	bDescriptorType	1	CONFIGURATION descriptor (2)	0x02
2	wTotalLength	2	32 (bytes)	0x0020
4	bNumInterfaces	1	1	0x01
5	bConfigurationValue	1	1	0x01
6	iConfiguration	1	0	0x00
7	bmAttributes	1	Not supported	0x80
8	bMaxPower	2	200 mA	0x64

Interface descriptor (9 bytes):

Offset	Field	Size	Value	Hex
0	bLength	1	9 (bytes)	0x09
1	bDescriptorType	1	INTERFACE descriptor (4)	0x04
2	bInterfaceNumber	1	0	0x00
3	bAlternateSetting	1	0	0x00
4	bNumEndpoints	1	2	0x02
5	bInterfaceClass	1	Mass Storage(08)	0x08
6	bInterfaceSubClass	1	SCSI transparent command set(06)	0x06
7	bInterfaceProtocol	1	Bulk-Only Transport(50)	0x50
8	iInterface	1	0	0x00

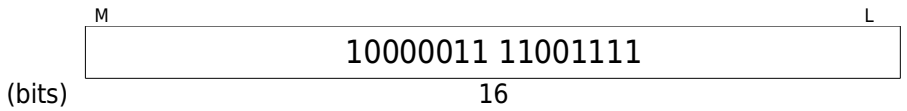
Endpoint descriptor (7 bytes):

Offset	Field	Size	Value	Hex
0	bLength	1	7 (bytes)	0x07
1	bDescriptorType	1	ENDPOINT descriptor (5)	0x05
2	bEndpointAddress	1	IN endpoint (D7), first endpoint (D0)	0x81
3	bmAttributes	1	Data endpoint (D5...D4), No synchronization (D3...D2), Bulk transfer type (D1...D0)	0x02
4	wMaxPacketSize	2	512 bytes	0x0200
6	bInterval	1	0	0x00

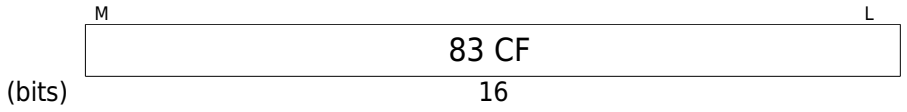
Endpoint descriptor (7 bytes):

Offset	Field	Size	Value	Hex
0	bLength	1	7 (bytes)	0x07
1	bDescriptorType	1	ENDPOINT descriptor (5)	0x05
2	bEndpointAddress	1	OUT endpoint (D7), second endpoint (D0)	0x01
3	bmAttributes	1	Data endpoint (D5...D4), No synchronization (D3...D2), Bulk transfer type (D1...D0)	0x02
4	wMaxPacketSize	2	512 bytes	0x0200
6	bInterval	1	1	0x01

The CRC observed in the sample capture is:



CRC (in Hex):



Putting it together (PID + 32 bytes of data + CRC):

	Binary (M...L)				Hex			
	Offset				Offset			
Offset	0	1			0			1
00	01001011	00001001	00000010	00100000	4B	09	02	20
04	00000000	00000001	00000001	00000000	00	01	01	00
08	10000000	01100100	00001001	00000100	80	64	09	04
12	00000000	00000000	00000010	00001000	00	00	02	08
16	00000110	01010000	00000000	00000111	06	50	00	07
20	00000101	10010001	00000010	00000000	05	81	02	00
24	00000010	00000000	00000111	00000101	02	00	07	05
28	00000001	00000010	00000000	00000010	01	02	00	02
32	00000001	11001111	10010011		01	CF	83	

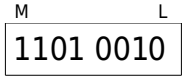
4.2.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

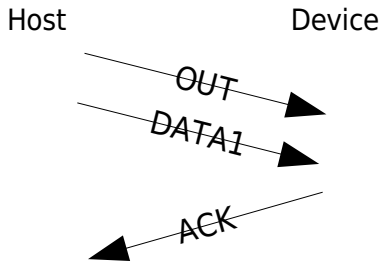
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

4.3 OUT transaction

The OUT transaction has the following three packets:

- OUT packet (->)
- DATA1 packet (->)
- ACK packet (<-)



4.3.1 OUT packet

An OUT packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1000	0111	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

```
11011 0000 000 1010 1110 0001
```

Regrouping in groups of 4-bits:

```
1101 1000 0000 1010 1110 0001
```

Hex values (as a byte):

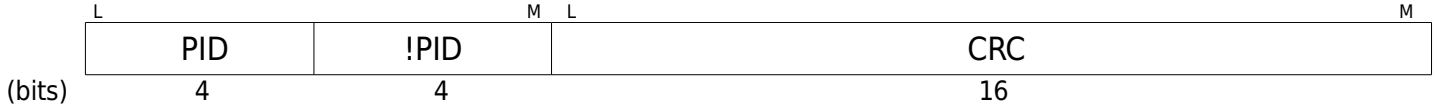
```
D8 0A E1
```

Summary (OUT packet):

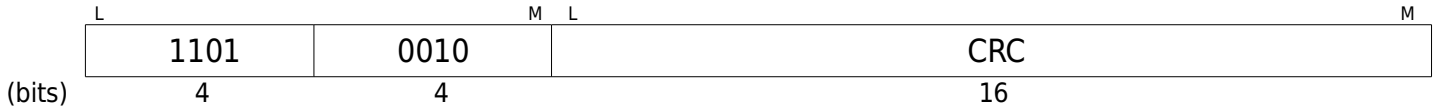
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11100001	00001010	E1	0A
02	11011000		D8	

4.3.2 DATA1 packet

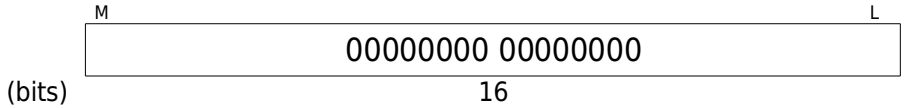
The DATA1 packet has no data. It consists of:



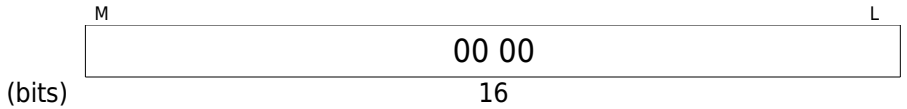
Values:



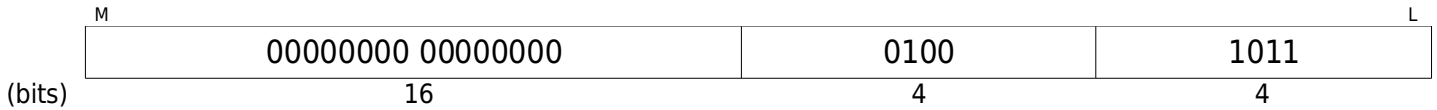
The CRC observed in the sample capture is:



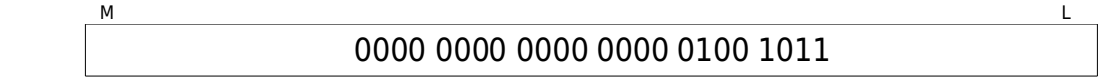
CRC (in Hex):



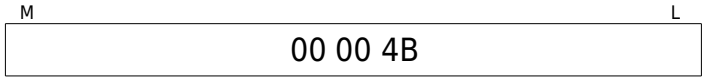
The packet arranged in MSB to LSB order:



In groups of 4 bits:



In Hex,



Summary (DATA1 packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00000000	4B	00
02	00000000		00	

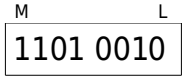
4.3.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

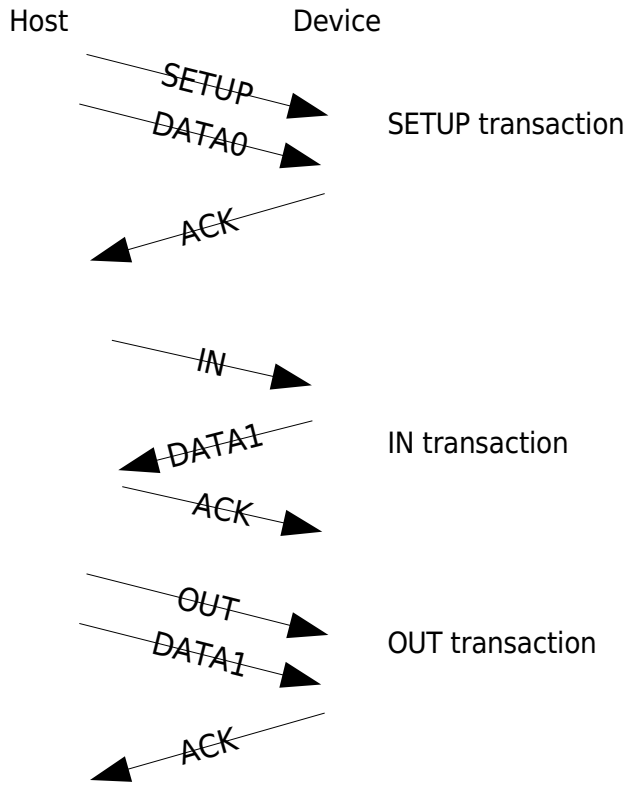
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

5. GetDescriptor (String Language IDs)

GetDescriptor (String Language IDs) transfer consists of the following transactions:

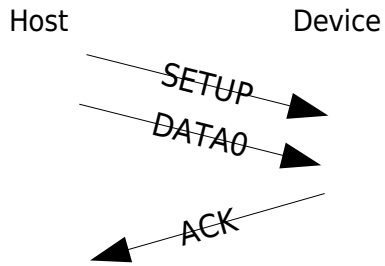
- SETUP transaction (->)
- IN transaction (<-)
- OUT transaction (->)



5.1 SETUP transaction

The SETUP transaction has the following three packets:

- SETUP packet (->)
- DATA0 packet (->)
- ACK packet (<-)



5.1.1 SETUP packet

A SETUP packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1011	0100	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

```
11011 0000 000 1010 0010 1101
```

Regrouping in groups of 4-bits:

```
1101 1000 0000 1010 0010 1101
```

Hex values:

```
D 8 0 A 2 D
```

Regrouping as a byte:

M L

D8 0A 2D

Summary (SETUP packet):

Offset	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
	0	1	0	1
00	00101101	00001010	2D	0A
02	11011000		D8	

5.1.2 DATA0 packet

A DATA0 packet consists of:

	PID	!PID	SETUP Data	CRC
(bits)	4	4	64	16

	1100	0011	SETUP Data	CRC
(bits)	4	4	64	16

The PID arranged in MSB to LSB order:

	1100	0011
(bits)	4	4

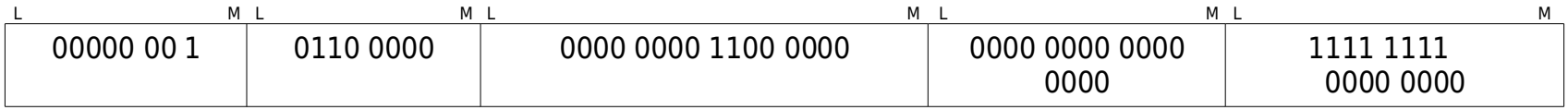
Hex values (PID):

C 3

Since this is for a SETUP packet, the data consists of 8 bytes. The format is:

	bmRequestType	bRequest	wValue	wIndex	wLength
(bytes)	1	1	2	2	2

The 8 bytes for SETUP details are as follows (in decimal):



bmRequestType:

D7: Data transfer direction
1 = Device-to-host

D6...D5: Type
0 = Standard

D4...D0: Recipient
0 = Device

bRequest:

GET_DESCRIPTOR request code is 6.

wValue:

Descriptor type is STRING, the value is 3. Descriptor index is 0.

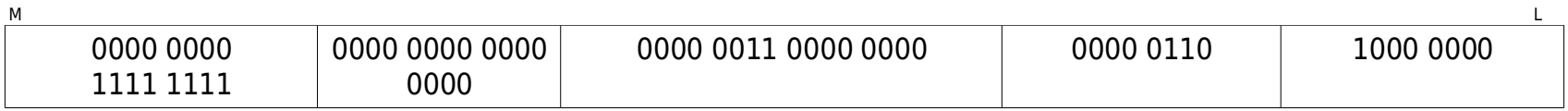
wIndex:

Value is zero.

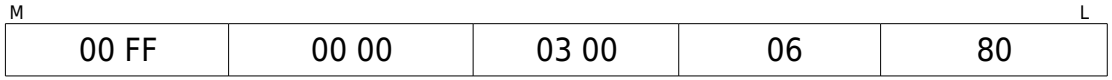
wLength:

Size of descriptor data, which is 255 bytes.

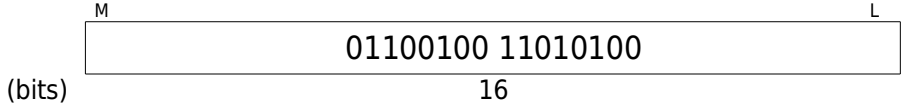
Putting it in MSB to LSB order:



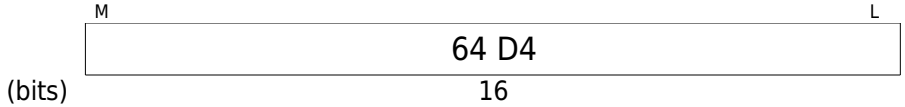
In Hex:



The CRC observed in the sample capture is:



CRC (in Hex):



Putting it all (PID + SETUP DATA + CRC) together,

Summary:

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11000011	10000000	C3	80
02	00000110	00000000	06	00
04	00000011	00000000	03	00
06	00000000	11111111	00	FF
08	00000000	11010100	00	D4
10	01100100		64	

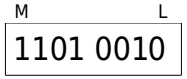
5.1.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

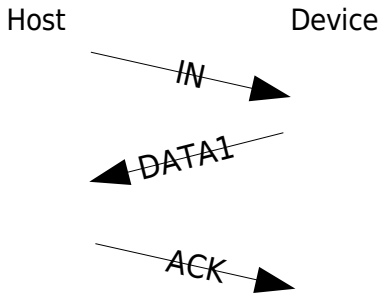
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

5.2 IN transaction

The IN transaction has the following three packets:

- IN packet (->)
- DATA1 packet (<-)
- ACK packet (->)



5.2.1 IN packet

A IN packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1001	0110	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

11011 0000 000 1010 0110 1001

Regrouping in groups of 4-bits:

1101 1000 0000 1010 0110 1001

Hex values (as a byte):

D8 0A 69

Summary (IN packet):

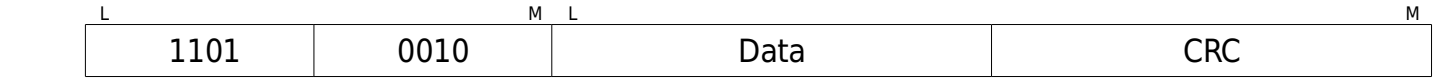
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01101001	00001010	69	0A
02	11011000		D8	

5.2.2 DATA1 packet

A DATA1 packet for GetDescriptor (String Language IDs) provides the 4 bytes of data. It consists of:

(bits)	4	4	32	16

Values:



(bits)	4	4	32	16

========== *BEGIN* ==========
String Descriptor Zero

Table 9-15: String Descriptor Zero, Specifying Languages Supported by the Device

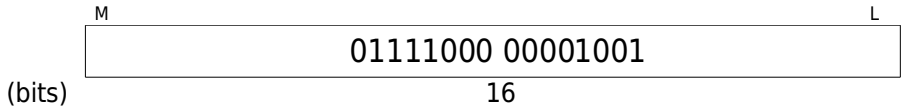
Offset	Field	Size	Value	Description
0	bLength	1	N+2	Length of this descriptor (bytes)
1	bDescriptorType	1	Constant	STRING descriptor type
2	wLANGID[0]	2	Number	LANGID code zero
...	...	...	...	...
N	wLANID[x]	2	Number	LANGID code x

Reference: Chapter 9: USB Device Framework, pages 273-274
========== *END* ==========

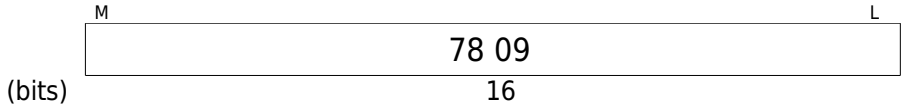
So, the data for our capture was:

Offset	Field	Size	Value	Hex
0	bLength	1	4 bytes	0x04
1	bDescriptorType	1	STRING descriptor	0x03
2	wLANGID[0]	2	English (US)	0x0409

The CRC observed in the sample capture is:



CRC (in Hex):



Putting it together (PID + 4 bytes of data + CRC):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00000100	4B	04
02	00000011	00001001	03	09
04	00000100	00001001	04	09
06	01111000		78	

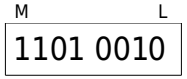
5.2.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

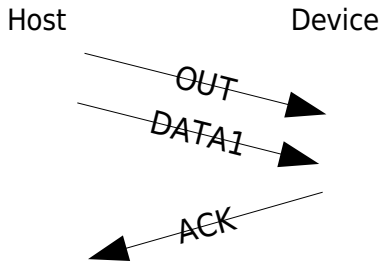
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

5.3 OUT transaction

The OUT transaction has the following three packets:

- OUT packet (->)
- DATA1 packet (->)
- ACK packet (<-)



5.3.1 OUT packet

An OUT packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1000	0111	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

```
11011 0000 000 1010 1110 0001
```

Regrouping in groups of 4-bits:

```
1101 1000 0000 1010 1110 0001
```

Hex values (as a byte):

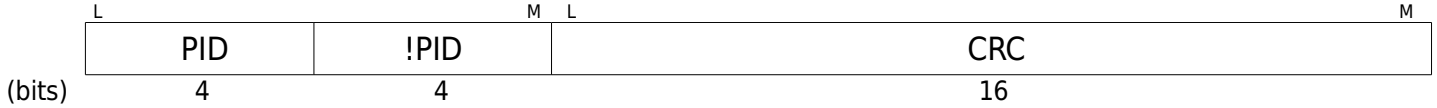
```
D8 0A E1
```

Summary (OUT packet):

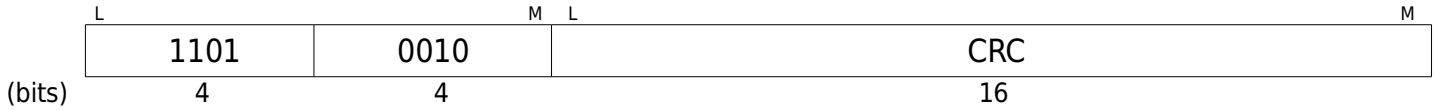
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11100001	00001010	E1	0A
02	11011000		D8	

5.3.2 DATA1 packet

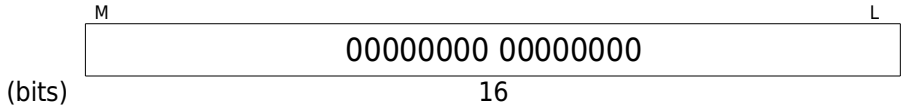
The DATA1 packet has no data. It consists of:



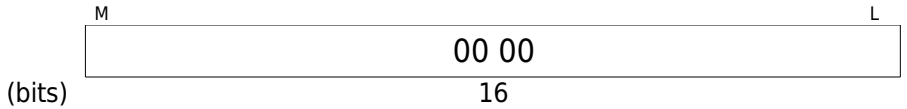
Values:



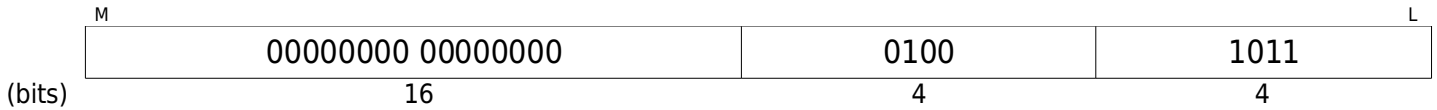
The CRC observed in the sample capture is:



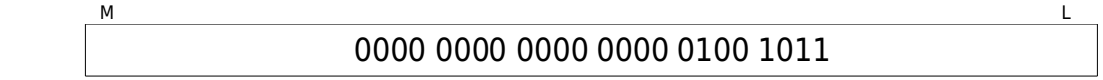
CRC (in Hex):



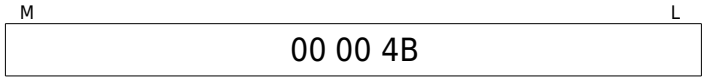
The packet arranged in MSB to LSB order:



In groups of 4 bits:



In Hex,



Summary (DATA1 packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00000000	4B	00
02	00000000		00	

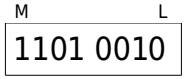
5.3.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

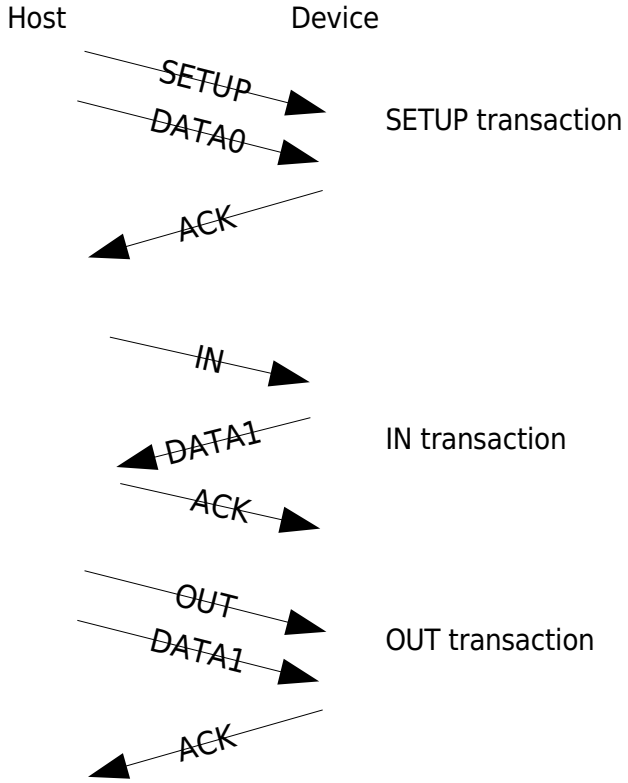
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

6. GetDescriptor (String 2)

GetDescriptor (String 2) transfer consists of the following transactions:

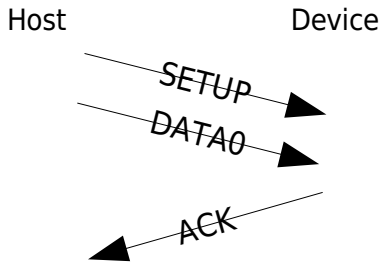
* SETUP transaction (->)
* IN transaction (<-)
* OUT transaction (->)



6.1 SETUP transaction

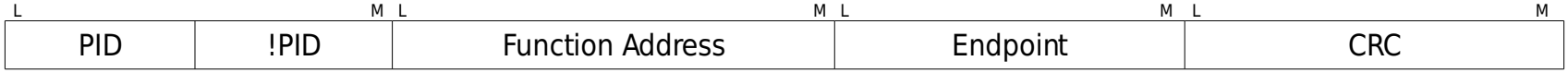
The SETUP transaction has the following three packets:

- SETUP packet (->)
- DATA0 packet (->)
- ACK packet (<-)

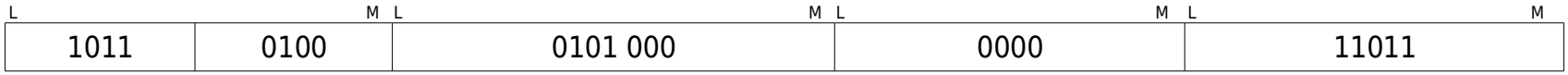


6.1.1 SETUP packet

A SETUP packet consists of:

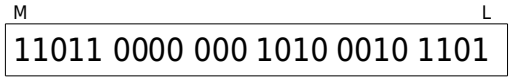


Values:

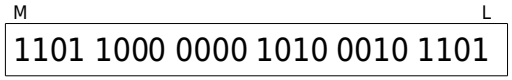


USB is little-endian. LSB goes out of the wire, first.

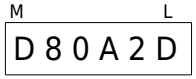
Displaying from MSB to LSB:



Regrouping in groups of 4-bits:



Hex values:



Regrouping as a byte:

M L

D8 0A 2D

Summary (SETUP packet):

Offset	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
	0	1	0	1
00	00101101	00001010	2D	0A
02	11011000		D8	

6.1.2 DATA0 packet

A DATA0 packet consists of:

	PID	!PID	SETUP Data	CRC
(bits)	4	4	64	16

	1100	0011	SETUP Data	CRC
(bits)	4	4	64	16

The PID arranged in MSB to LSB order:

	1100	0011
(bits)	4	4

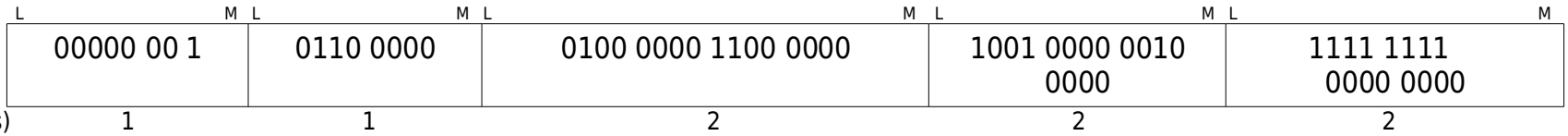
Hex values (PID):

C 3

Since this is for a SETUP packet, the data consists of 8 bytes. The format is:

	bmRequestType	bRequest	wValue	wIndex	wLength
(bytes)	1	1	2	2	2

The 8 bytes for SETUP details are as follows (in decimal):



bmRequestType:

D7: Data transfer direction
1 = Device-to-host

D6...D5: Type
0 = Standard

D4...D0: Recipient
0 = Device

bRequest:

GET_DESCRIPTOR request code is 6.

wValue:

Descriptor type is STRING, the value is 3. Descriptor index is 2.

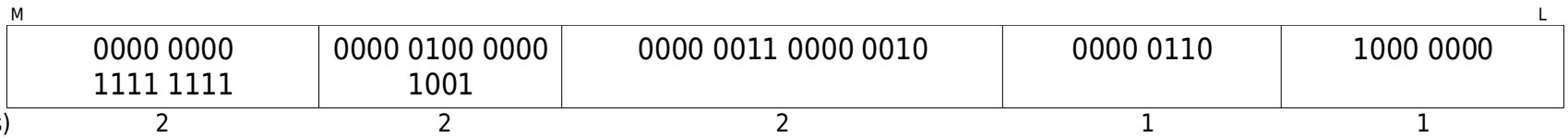
wIndex:

Language ID is English (US). The value is 0x0409.

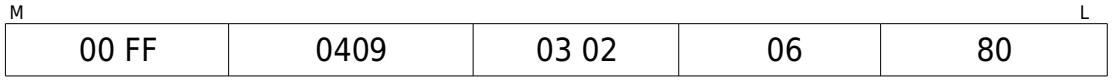
wLength:

Size of descriptor data, which is 255 bytes.

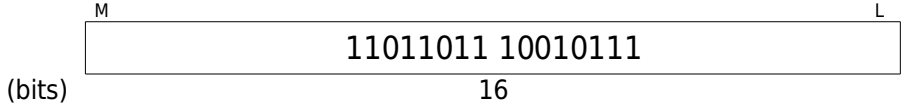
Putting it in MSB to LSB order:



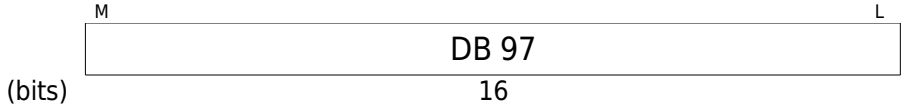
In Hex:



The CRC observed in the sample capture is:



CRC (in Hex):



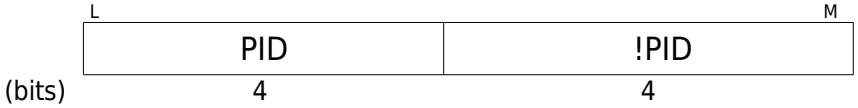
Putting it all (PID + SETUP DATA + CRC) together,

Summary (DATA0 packet):

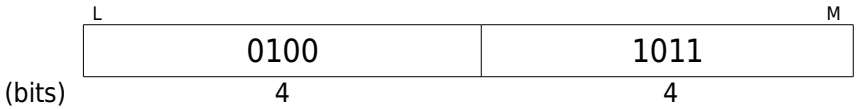
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11000011	10000000	C3	80
02	00000110	00000010	06	02
04	00000011	00001001	03	09
06	00000100	11111111	04	FF
08	00000000	10010111	00	97
10	11011011		DB	

6.1.3 ACK packet

An ACK packet consists of:

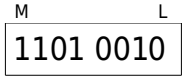


Values:



USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

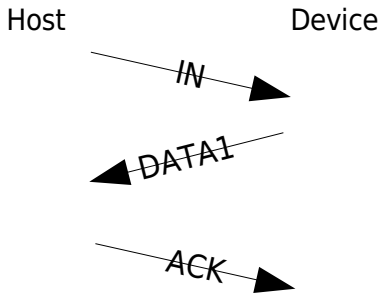
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

6.2 IN transaction

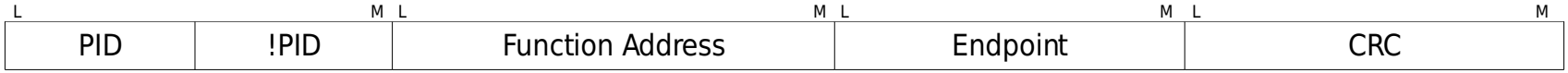
The IN transaction has the following three packets:

- IN packet (->)
- DATA1 packet (<-)
- ACK packet (->)

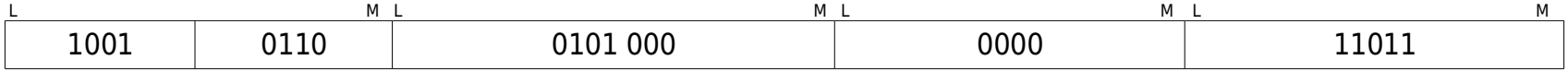


6.2.1 IN packet

A IN packet consists of:

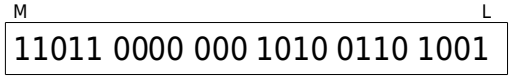


Values:

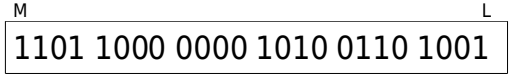


USB is little-endian. LSB goes out of the wire, first.

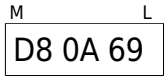
Displaying from MSB to LSB:



Regrouping in groups of 4-bits:



Hex values (as a byte):

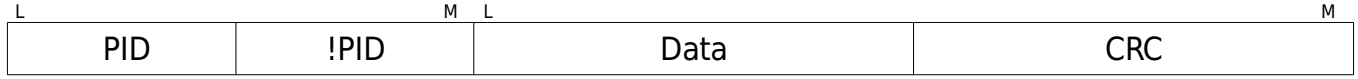


Summary (IN packet):

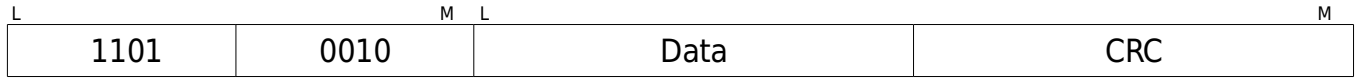
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01101001	00001010	69	0A
02	11011000		D8	

6.2.2 DATA1 packet

A DATA1 packet for GetDescriptor (String 2) provides 24 bytes of data for this device. It consists of:



Values:



========== *BEGIN* ==========
UNICODE String Descriptor

Table 9-16: UNICODE String Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	N+2	Length of this descriptor (bytes)
1	bDescriptorType	1	Constant	STRING descriptor type
2	bString	N	Number	UNICODE encoded string

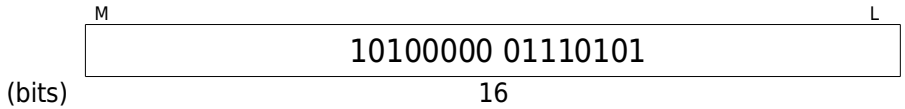
- The string is not NULL-terminated.
- The string length is computed by subtracting two from the value of the first byte in the descriptor.

Reference: Chapter 9: USB Device Framework, page 274
========== *END* ==========

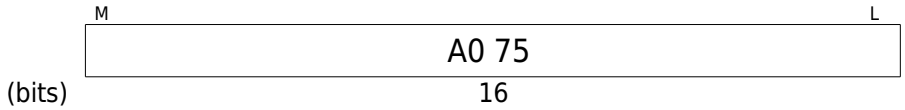
So, the data for our capture was:

Offset	Field	Size	Value	Hex
0	bLength	1	24 bytes	0x18
1	bDescriptorType	1	STRING descriptor	0x03
2	bString	2	C	0x0043
4	bString	2	r	0x0072
6	bString	2	u	0x0075
8	bString	2	z	0x007A
10	bString	2	e	0x0065
12	bString	2	r	0x0072
14	bString	2	<space>	0x0020
16	bString	2	M	0x004D
18	bString	2	i	0x0069
20	bString	2	n	0x006E
22	bString	2	i	0x0069

The CRC observed in the sample capture is:



CRC (in Hex):



Putting it together (PID + 24 bytes of data + CRC):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00011000	4B	18
02	00000011	01000110	03	43
04	00000000	01110010	00	72
06	00000000	01110101	00	75
08	00000000	01111010	00	7A
10	00000000	01100101	00	65
12	00000000	01110010	00	72
14	00000000	00100000	00	20
16	00000000	01001101	00	4D
18	00000000	01101001	00	69
20	00000000	01101110	00	6E
22	00000000	01101001	00	69
24	00000000	01110101	00	75
26	10100000		A0	

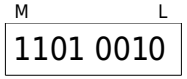
6.2.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

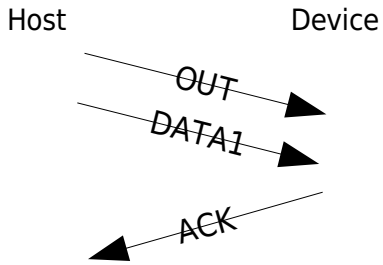
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

6.3 OUT transaction

The OUT transaction has the following three packets:

- OUT packet (->)
- DATA1 packet (->)
- ACK packet (<-)



6.3.1 OUT packet

An OUT packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1000	0111	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

```
11011 0000 000 1010 1110 0001
```

Regrouping in groups of 4-bits:

```
1101 1000 0000 1010 1110 0001
```

Hex values (as a byte):

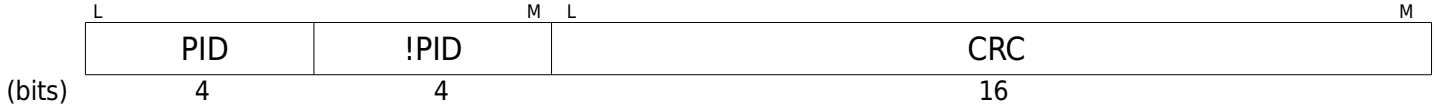
```
D8 0A E1
```

Summary (OUT packet):

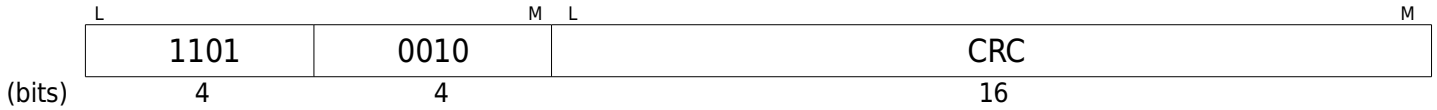
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11100001	00001010	E1	0A
02	11011000		D8	

6.3.2 DATA1 packet

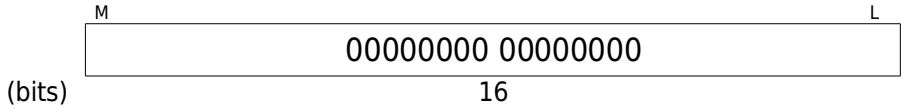
The DATA1 packet has no data. It consists of:



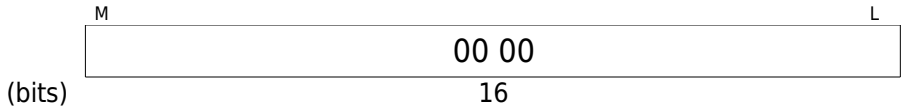
Values:



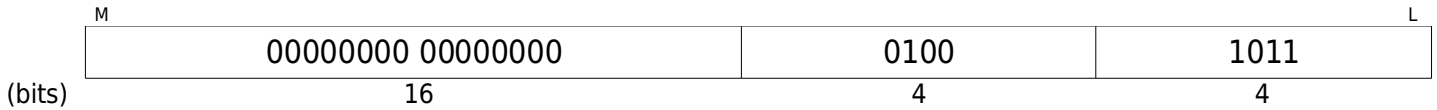
The CRC observed in the sample capture is:



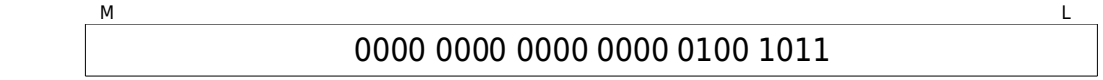
CRC (in Hex):



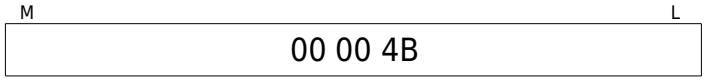
The packet arranged in MSB to LSB order:



In groups of 4 bits:



In Hex,



Summary (DATA1 packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00000000	4B	00
02	00000000		00	

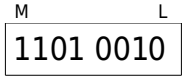
6.3.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

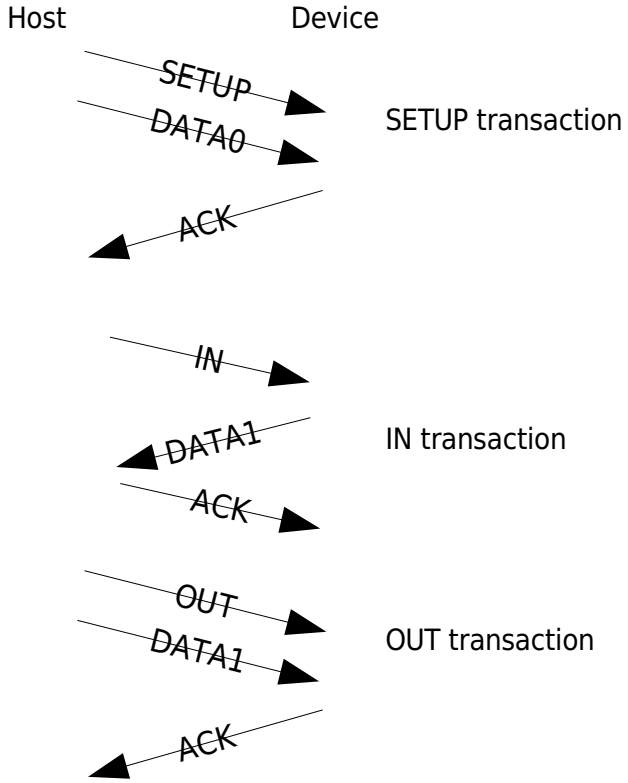
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

7. GetDescriptor (String 1)

GetDescriptor (String 1) transfer consists of the following transactions:

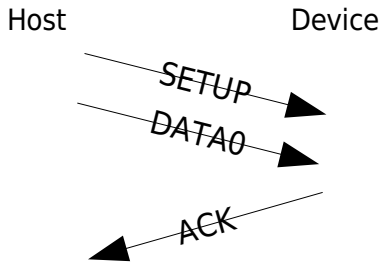
* SETUP transaction (->)
* IN transaction (<-)
* OUT transaction (->)



7.1 SETUP transaction

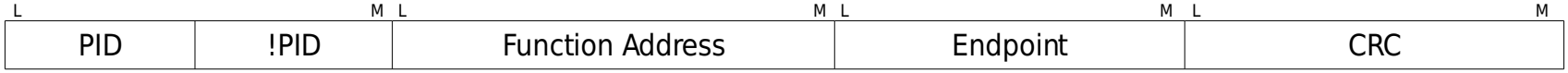
The SETUP transaction has the following three packets:

- SETUP packet (->)
- DATA0 packet (->)
- ACK packet (<-)



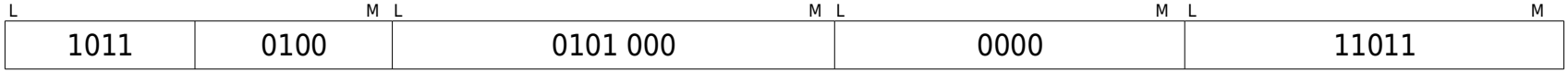
7.1.1 SETUP packet

A SETUP packet consists of:



(bits)	4	4	7	4	5

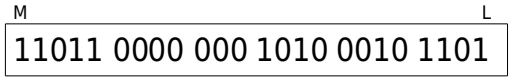
Values:



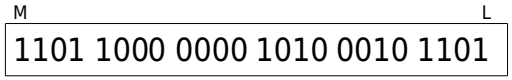
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

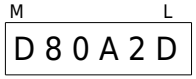
Displaying from MSB to LSB:



Regrouping in groups of 4-bits:



Hex values:



Regrouping as a byte:

M L

D8 0A 2D

Summary (SETUP packet):

Offset	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
	0	1	0	1
00	00101101	00001010	2D	0A
02	11011000		D8	

7.1.2 DATA0 packet

A DATA0 packet consists of:

	PID	!PID	SETUP Data	CRC
(bits)	4	4	64	16

	1100	0011	SETUP Data	CRC
(bits)	4	4	64	16

The PID arranged in MSB to LSB order:

	1100	0011
(bits)	4	4

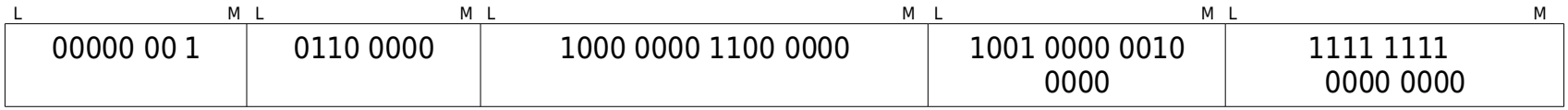
Hex values (PID):

C 3

Since this is for a SETUP packet, the data consists of 8 bytes. The format is:

	bmRequestType	bRequest	wValue	wIndex	wLength
(bytes)	1	1	2	2	2

The 8 bytes for SETUP details are as follows (in decimal):



bmRequestType:

D7: Data transfer direction
1 = Device-to-host

D6...D5: Type
0 = Standard

D4...D0: Recipient
0 = Device

bRequest:

GET_DESCRIPTOR request code is 6.

wValue:

Descriptor type is STRING, the value is 3. Descriptor index is 1.

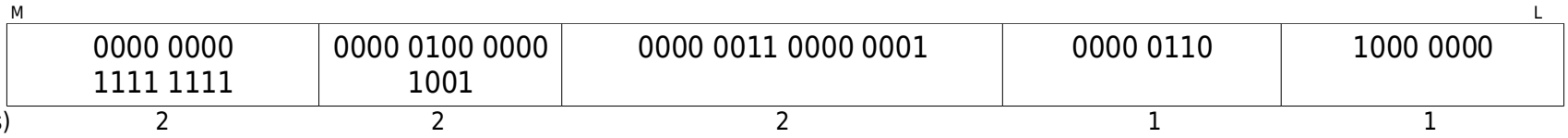
wIndex:

Language ID is English (US). The value is 0x0409.

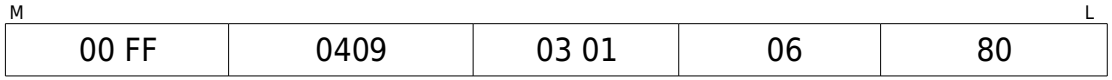
wLength:

Size of descriptor data, which is 255 bytes.

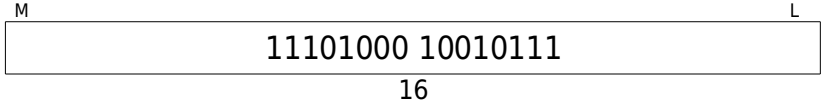
Putting it in MSB to LSB order:



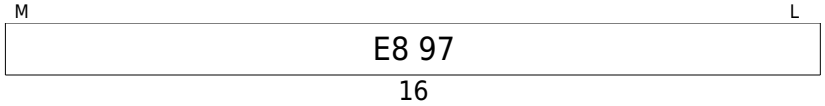
In Hex:



The CRC observed in the sample capture is:



CRC (in Hex):



Putting it all (PID + SETUP DATA + CRC) together,

Summary (DATA0 packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11000011	10000000	C3	80
02	00000110	00000010	06	01
04	00000011	00001001	03	09
06	00000100	11111111	04	FF
08	00000000	10010111	00	97
10	11101000		E8	

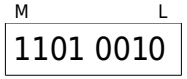
7.1.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

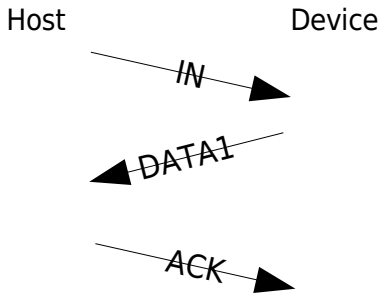
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

7.2 IN transaction

The IN transaction has the following three packets:

- IN packet (->)
- DATA1 packet (<-)
- ACK packet (->)



7.2.1 IN packet

A IN packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1001	0110	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

11011 0000 000 1010 0110 1001

Regrouping in groups of 4-bits:

1101 1000 0000 1010 0110 1001

Hex values (as a byte):

D8 0A 69

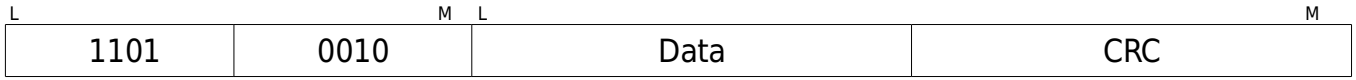
Summary (IN packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01101001	00001010	69	0A
02	11011000		D8	

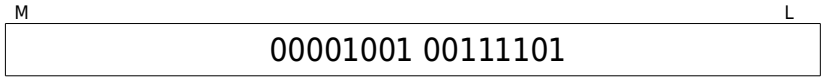
7.2.2 DATA1 packet

A DATA1 packet for GetDescriptor (String 1) provides 40 bytes of data for this device. It consists of:

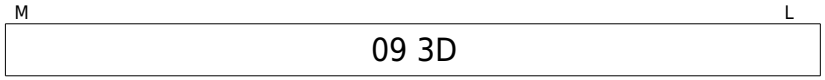
Values:



The CRC observed in the sample capture is:



CRC (in Hex):



So, the data for our capture was:

Offset	Field	Size	Value	Hex
0	bLength	1	40 bytes	0x28
1	bDescriptorType	1	STRING descriptor	0x03
2	bString	2	S	0x0053
4	bString	2	a	0x0061
6	bString	2	n	0x006E
8	bString	2	d	0x0044
10	bString	2	i	0x0069
12	bString	2	s	0x0073
14	bString	2	k	0x006B
16	bString	2		0x0020
18	bString	2	C	0x0043
20	bString	2	o	0x006F
22	bString	2	r	0x0072
24	bString	2	p	0x0070
26	bString	2	o	0x006F
28	bString	2	r	0x0072
30	bString	2	a	0x0061
32	bString	2	t	0x0074
34	bString	2	i	0x0069
36	bString	2	o	0x006F
38	bString	2	n	0x006E

Putting it together (PID + 40 bytes of data + CRC):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00101000	4B	28
02	00000011	01010011	03	53
04	00000000	01100001	00	61
06	00000000	01101110	00	6E
08	00000000	01100110	00	44
10	00000000	01101001	00	69
12	00000000	01110011	00	73
14	00000000	01101011	00	6B
16	00000000	00100000	00	20
18	00000000	01000011	00	43
20	00000000	01101111	00	6F
22	00000000	01110010	00	72
24	00000000	01110000	00	70
26	00000000	01101111	00	6F
28	00000000	01110010	00	72
30	00000000	01100001	00	61
32	00000000	01110100	00	74
34	00000000	01101001	00	69
36	00000000	01101111	00	6F
38	00000000	01101110	00	6E
40	00000000	00111101	00	3D
42	00001001		09	

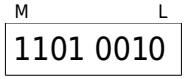
7.2.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

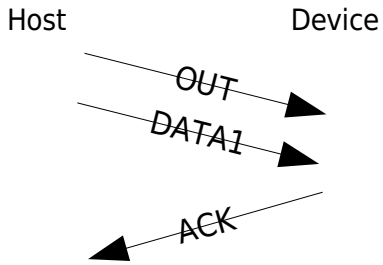
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

7.3 OUT transaction

The OUT transaction has the following three packets:

- OUT packet (->)
- DATA1 packet (->)
- ACK packet (<-)



7.3.1 OUT packet

An OUT packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1000	0111	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

```
11011 0000 000 1010 1110 0001
```

Regrouping in groups of 4-bits:

```
1101 1000 0000 1010 1110 0001
```

Hex values (as a byte):

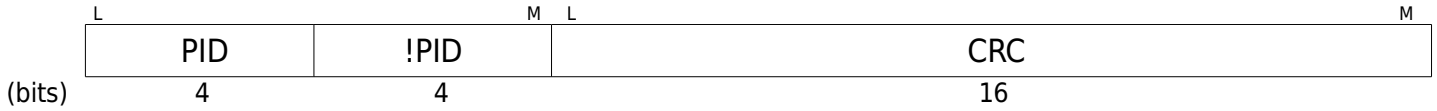
```
D8 0A E1
```

Summary (OUT packet):

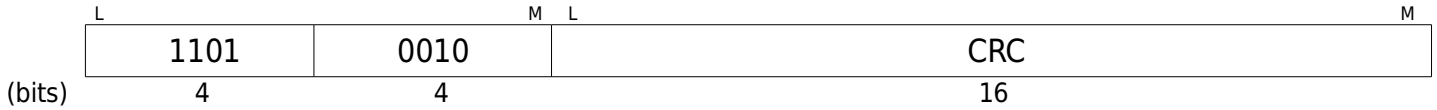
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11100001	00001010	E1	0A
02	11011000		D8	

7.3.2 DATA1 packet

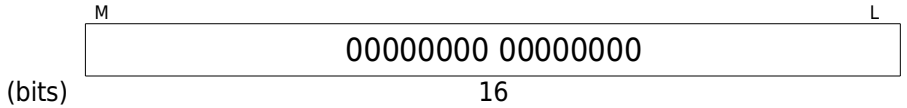
The DATA1 packet has no data. It consists of:



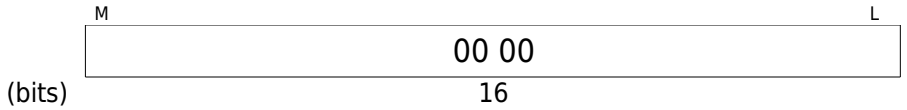
Values:



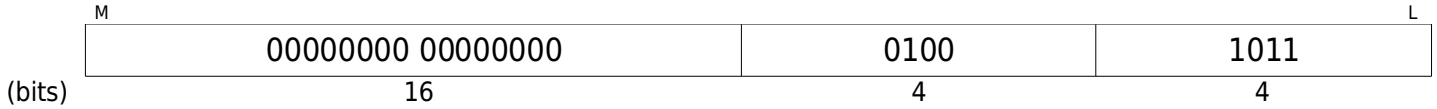
The CRC observed in the sample capture is:



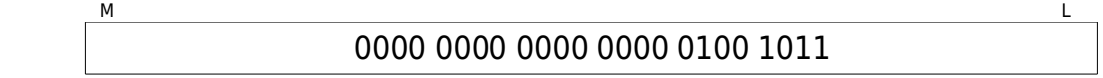
CRC (in Hex):



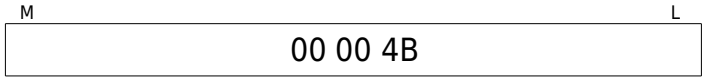
The packet arranged in MSB to LSB order:



In groups of 4 bits:



In Hex,



Summary (DATA1 packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00000000	4B	00
02	00000000		00	

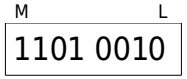
7.3.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

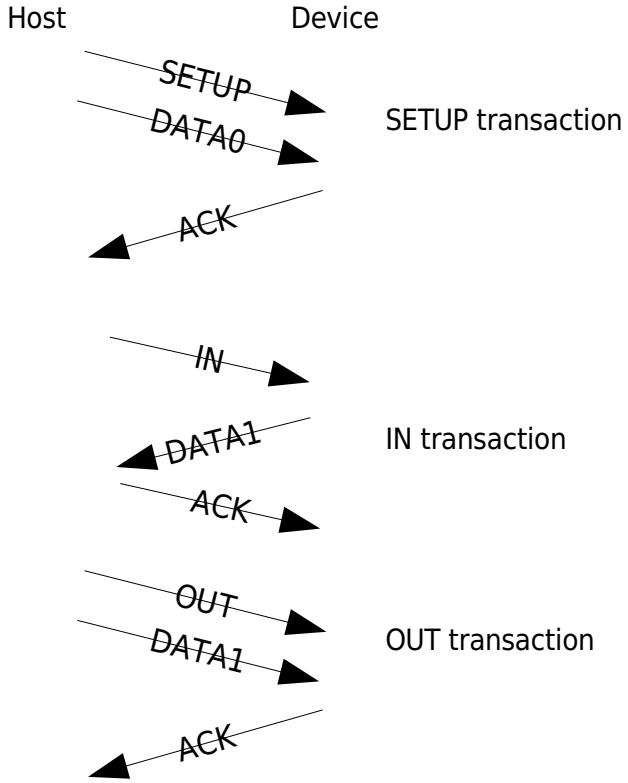
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

8. GetDescriptor (String 3)

GetDescriptor (String 3) transfer consists of the following transactions:

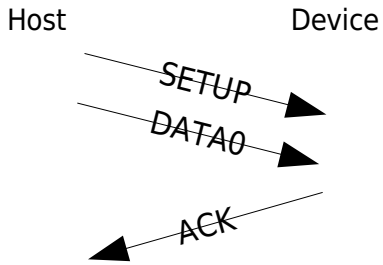
* SETUP transaction (->)
* IN transaction (<-)
* OUT transaction (->)



8.1 SETUP transaction

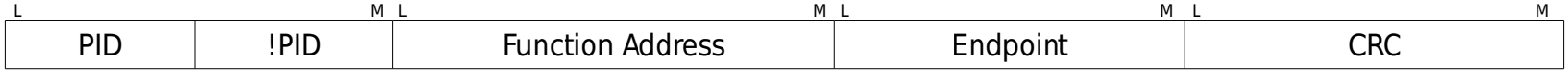
The SETUP transaction has the following three packets:

- SETUP packet (->)
- DATA0 packet (->)
- ACK packet (<-)

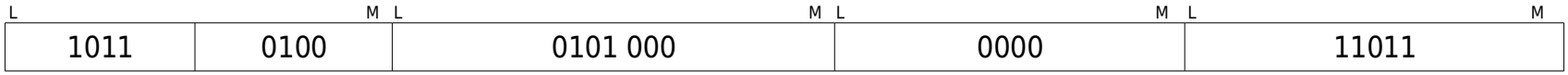


8.1.1 SETUP packet

A SETUP packet consists of:

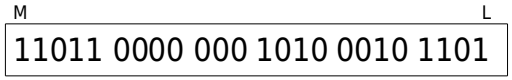


Values:

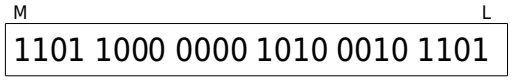


USB is little-endian. LSB goes out of the wire, first.

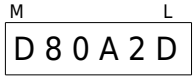
Displaying from MSB to LSB:



Regrouping in groups of 4-bits:



Hex values:



Regrouping as a byte:

M L

D8 0A 2D

Summary (SETUP packet):

Offset	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
	0	1	0	1
00	00101101	00001010	2D	0A
02	11011000		D8	

8.1.2 DATA0 packet

A DATA0 packet consists of:

	PID	!PID	SETUP Data	CRC
(bits)	4	4	64	16

	1100	0011	SETUP Data	CRC
(bits)	4	4	64	16

The PID arranged in MSB to LSB order:

	1100	0011
(bits)	4	4

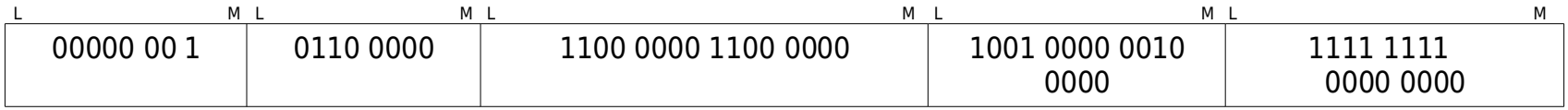
Hex values (PID):

C 3

Since this is for a SETUP packet, the data consists of 8 bytes. The format is:

	bmRequestType	bRequest	wValue	wIndex	wLength
(bytes)	1	1	2	2	2

The 8 bytes for SETUP details are as follows (in decimal):



bmRequestType:

D7: Data transfer direction
1 = Device-to-host

D6...D5: Type
0 = Standard

D4...D0: Recipient
0 = Device

bRequest:

GET_DESCRIPTOR request code is 6.

wValue:

Descriptor type is STRING, the value is 3. Descriptor index is 3.

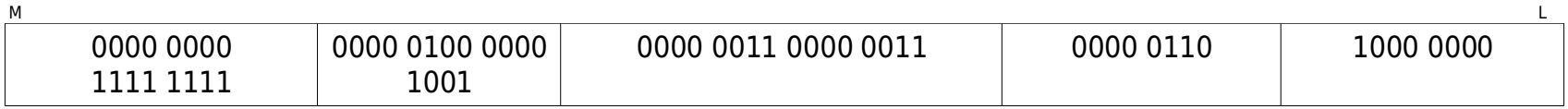
wIndex:

Language ID is English (US). The value is 0x0409.

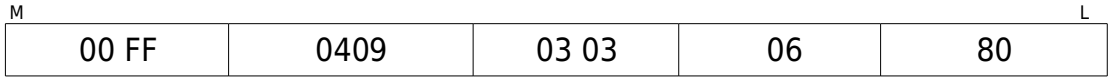
wLength:

Size of descriptor data, which is 255 bytes.

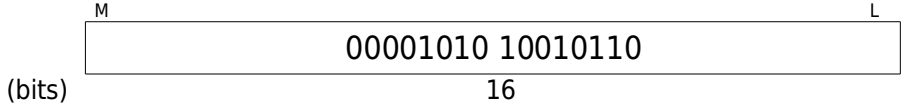
Putting it in MSB to LSB order:



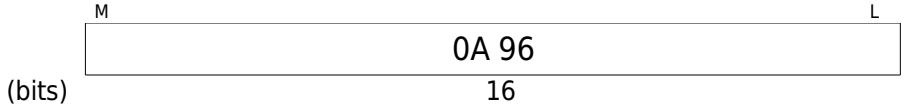
In Hex:



The CRC observed in the sample capture is:



CRC (in Hex):



Putting it all (PID + SETUP DATA + CRC) together,

Summary (DATA0 packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11000011	10000000	C3	80
02	00000110	00000011	06	03
04	00000011	00001001	03	09
06	00000100	11111111	04	FF
08	00000000	10010110	00	96
10	00001010		0A	

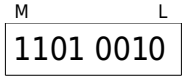
8.1.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

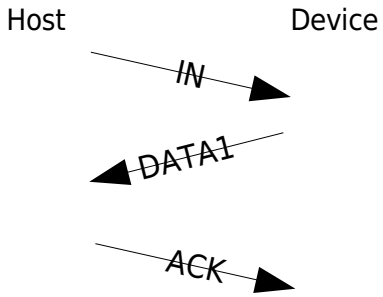
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

8.2 IN transaction

The IN transaction has the following three packets:

- IN packet (->)
- DATA1 packet (<-)
- ACK packet (->)



8.2.1 IN packet

A IN packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1001	0110	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

11011 0000 000 1010 0110 1001

Regrouping in groups of 4-bits:

1101 1000 0000 1010 0110 1001

Hex values (as a byte):

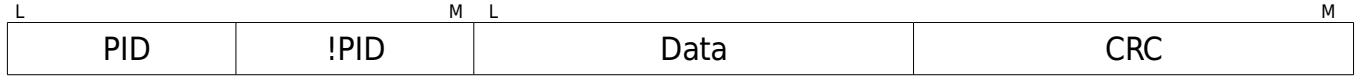
D8 0A 69

Summary (IN packet):

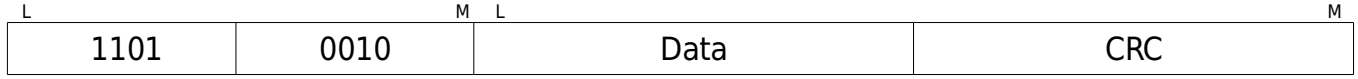
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01101001	00001010	69	0A
02	11011000		D8	

8.2.2 DATA1 packet

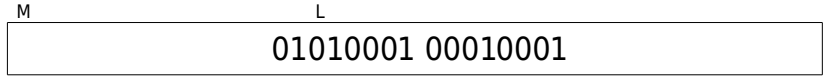
A DATA1 packet for GetDescriptor (String 3) provides 42 bytes of data for this device. It consists of:



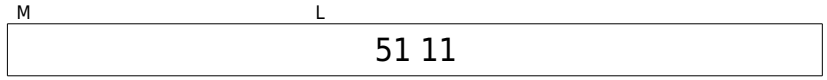
Values:



The CRC observed in the sample capture is:



CRC (in Hex):



So, the data for our capture was:

Offset	Field	Size	Value	Hex
0	bLength	1	42 bytes	0x2A
1	bDescriptorType	1	STRING descriptor	0x03
2	bString	2	S	0x0053
4	bString	2	N	0x004E
6	bString	2	D	0x0044
8	bString	2	K	0x004B
10	bString	2	4	0x0034
12	bString	2	0	0x0030
14	bString	2	E	0x0045
16	bString	2	1	0x0031
18	bString	2	6	0x0036
20	bString	2	4	0x0034
22	bString	2	1	0x0031
24	bString	2	7	0x0037
26	bString	2	5	0x0035
28	bString	2	8	0x0038
30	bString	2	6	0x0036
32	bString	2	0	0x0030
34	bString	2	0	0x0030
36	bString	2	1	0x0031
38	bString	2	0	0x0030
40	bString	2	6	0x0036

Putting it together (PID + 42 bytes of data + CRC):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00101000	4B	2A
02	00000011	01010011	03	53
04	00000000	01001110	00	4E
06	00000000	01000100	00	44
08	00000000	01001011	00	4B
10	00000000	00110100	00	34
12	00000000	00110000	00	30
14	00000000	01000101	00	45
16	00000000	00110001	00	31
18	00000000	00110110	00	36
20	00000000	00110100	00	34
22	00000000	00110001	00	31
24	00000000	00110111	00	37
26	00000000	00110101	00	35
28	00000000	00111000	00	38
30	00000000	00110110	00	36
32	00000000	00110000	00	30
34	00000000	00110000	00	30
36	00000000	00110001	00	31
38	00000000	00110000	00	30
40	00000000	00110110	00	36
42	00000000	00010001	00	11
44	01010001		51	

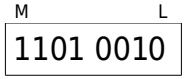
8.2.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

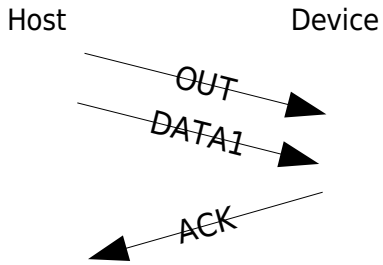
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

8.3 OUT transaction

The OUT transaction has the following three packets:

- OUT packet (->)
- DATA1 packet (->)
- ACK packet (<-)



8.3.1 OUT packet

An OUT packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1000	0111	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

11011 0000 000 1010 1110 0001

Regrouping in groups of 4-bits:

1101 1000 0000 1010 1110 0001

Hex values (as a byte):

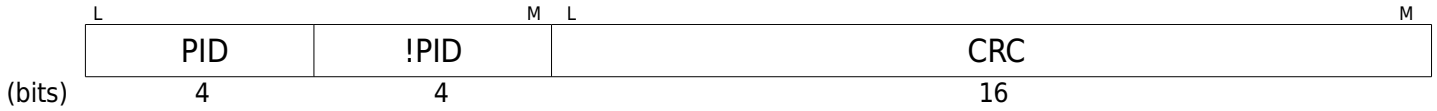
D8 0A E1

Summary (OUT packet):

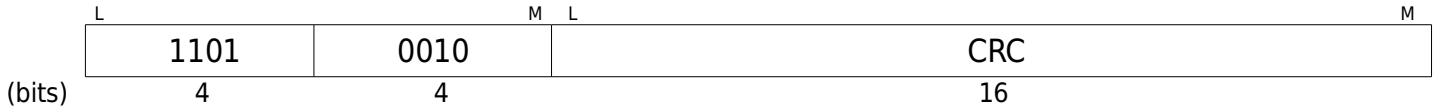
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11100001	00001010	E1	0A
02	11011000		D8	

8.3.2 DATA1 packet

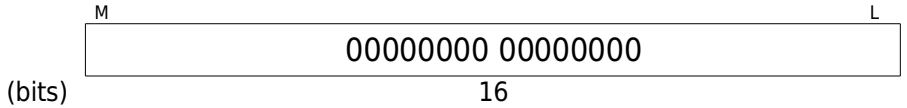
The DATA1 packet has no data. It consists of:



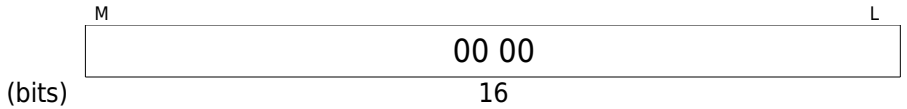
Values:



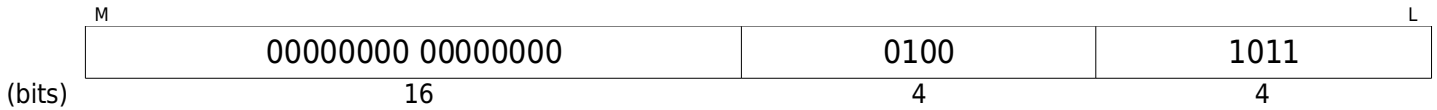
The CRC observed in the sample capture is:



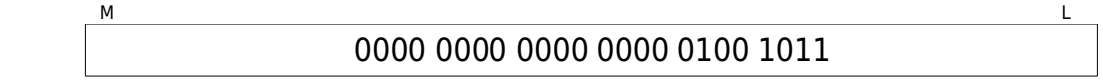
CRC (in Hex):



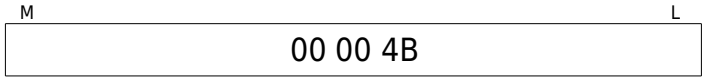
The packet arranged in MSB to LSB order:



In groups of 4 bits:



In Hex,



Summary (DATA1 packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00000000	4B	00
02	00000000		00	

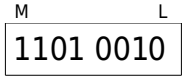
8.3.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

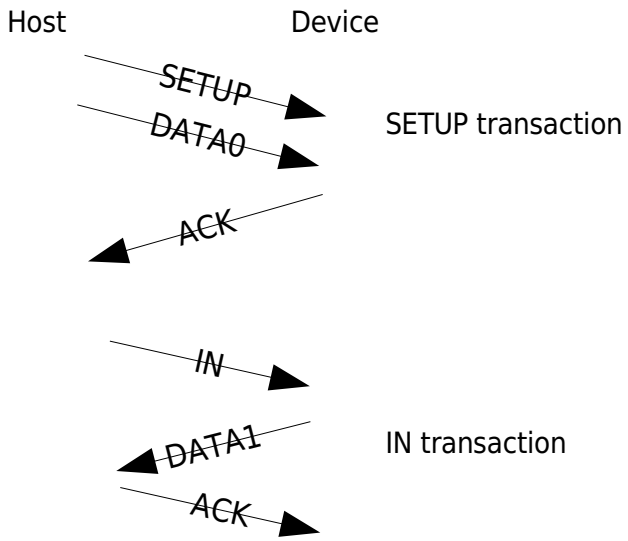
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

9. SetConfiguration

SetConfiguration transfer consists of the following transactions:

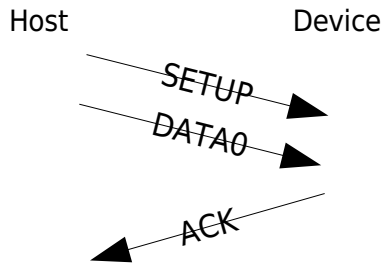
* SETUP transaction (->)
* IN transaction (<-)



9.1 SETUP transaction

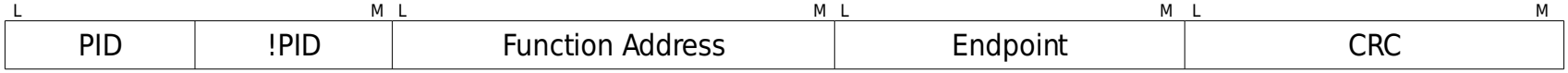
The SETUP transaction has the following three packets:

- SETUP packet (->)
- DATA0 packet (->)
- ACK packet (<-)

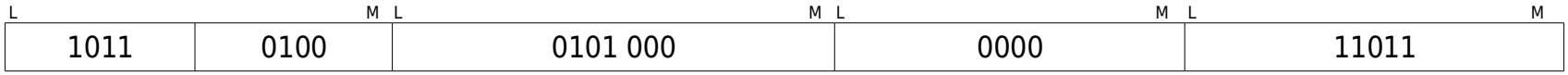


9.1.1 SETUP packet

A SETUP packet consists of:

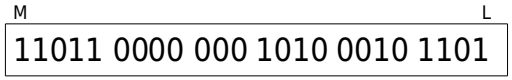


Values:

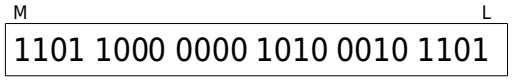


USB is little-endian. LSB goes out of the wire, first.

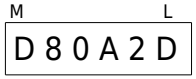
Displaying from MSB to LSB:



Regrouping in groups of 4-bits:



Hex values:



Regrouping as a byte:

M L

D8 0A 2D

Summary (SETUP packet):

Offset	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
	0	1	0	1
00	00101101	00001010	2D	0A
02	11011000		D8	

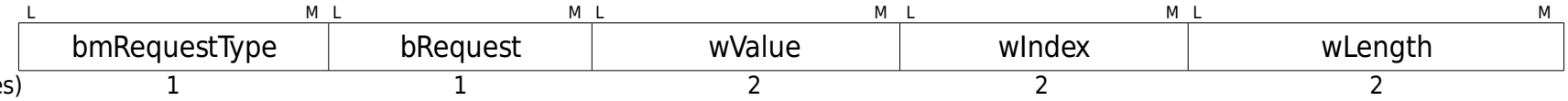
9.1.2 DATA0 packet

A DATA0 packet consists of:

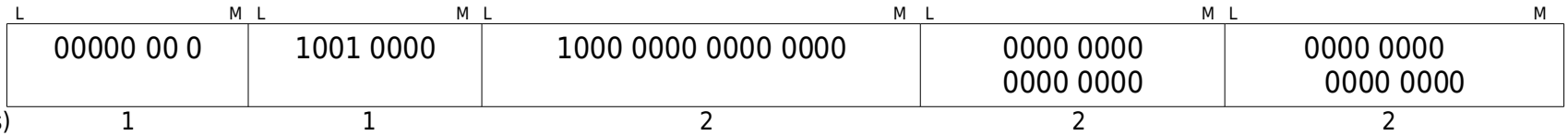
The PID arranged in MSB to LSB order:

Hex values (PID):

So, the SETUP packet consists of:



The 8 bytes for SETUP details are as follows:



bmRequestType:

D7: Data transfer direction
0 = Host-to-device

D6...D5: Type
0 = Standard

D4...D0: Recipient
0 = Device

bRequest:

SET_CONFIGURATION request code is 9.

wValue:

Configuration 1.

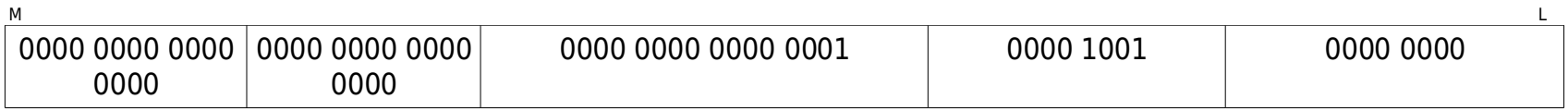
wIndex:

Zero.

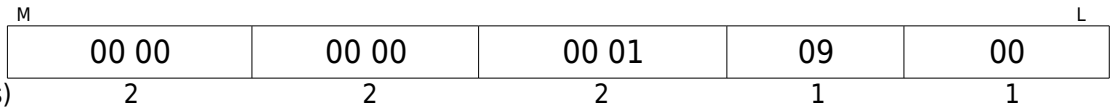
wLength:

Zero.

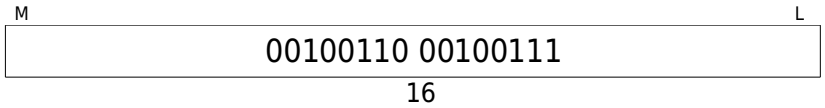
Putting it in MSB to LSB order:



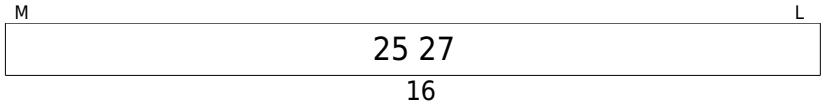
In Hex:



The CRC observed in the sample capture is:



CRC (in Hex):



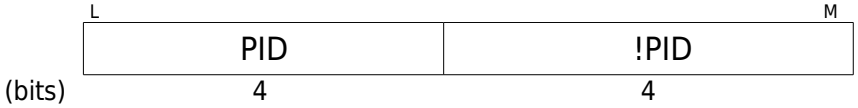
Putting it all (PID + SETUP DATA + CRC) together,

Summary (DATA0 packet):

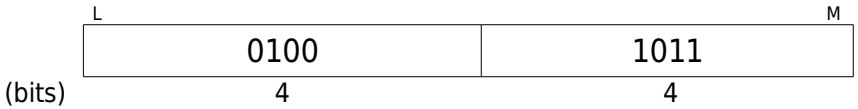
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	11000011	00000000	C3	00
02	00001001	00000001	09	01
04	00000000	00000000	00	00
06	00000000	00000000	00	00
08	00000000	00100111	00	27
10	00100110		25	

9.1.3 ACK packet

An ACK packet consists of:

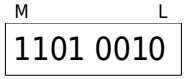


Values:



USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

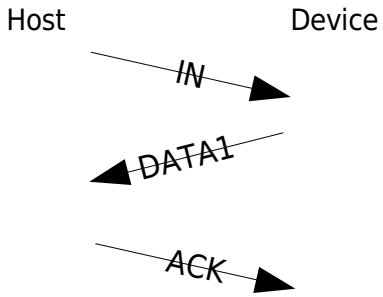
Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

9.2 IN transaction

The IN transaction has the following three packets:

- IN packet (->)
- DATA1 packet (<-)
- ACK packet (->)



9.2.1 IN packet

A IN packet consists of:

	PID	!PID	Function Address	Endpoint	CRC
(bits)	4	4	7	4	5

Values:

	1001	0110	0101 000	0000	11011
(bits)	4	4	7	4	5

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:

11011 0000 000 1010 0110 1001

Regrouping in groups of 4-bits:

1101 1000 0000 1010 0110 1001

Hex values (as a byte):

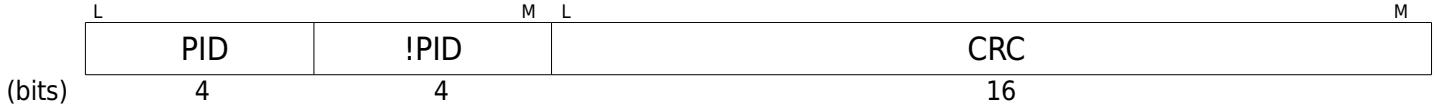
D8 0A 69

Summary (IN packet):

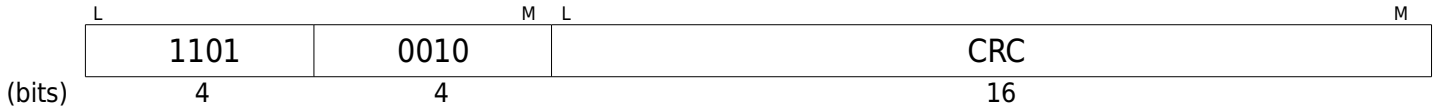
	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01101001	00001010	69	0A
02	11011000		D8	

9.2.2 DATA1 packet

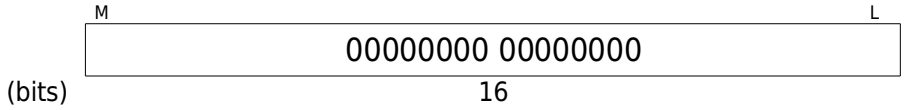
A DATA1 packet for SetAddress has no data. It consists of:



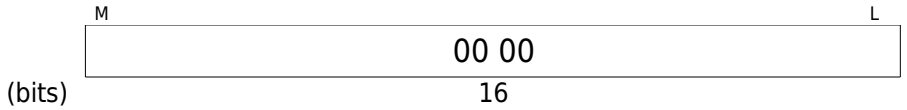
Values:



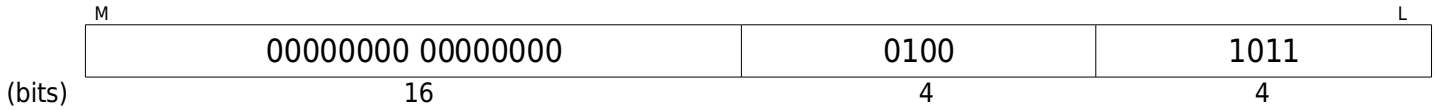
The CRC observed in the sample capture is:



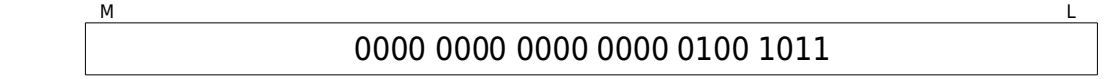
CRC (in Hex):



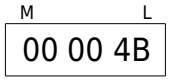
The packet arranged in MSB to LSB order:



In groups of 4 bits:



In Hex,



Summary (DATA1 packet):

	Binary (M...L)		Hexadecimal (M...L)	
	Offset		Offset	
Offset	0	1	0	1
00	01001011	00000000	4B	00
02	00000000		00	

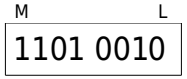
9.2.3 ACK packet

An ACK packet consists of:

Values:

USB is little-endian. LSB goes out of the wire, first.

Displaying from MSB to LSB:



Hex values:

Summary (ACK packet):

	Binary (M...L)	Hexadecimal (M...L)
	Offset	Offset
Offset	0	0
00	11010010	D2

References

[1] Universal Serial Bus Specification. Revision 2.0. April 27, 2000. http://www.usb.org

GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it,
with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document
straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.

J. Preserve the network location, if any, given in the Document for
   public access to a Transparent copy of the Document, and likewise
   the network locations given in the Document for previous versions
   it was based on. These may be placed in the "History" section.
   You may omit a network location for a work that was published at
   least four years before the Document itself, or if the original
   publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications",
   Preserve the Title of the section, and preserve in the section all
   the substance and tone of each of the contributor acknowledgements
   and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document,
   unaltered in their text and in their titles. Section numbers
   or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section
   may not be included in the Modified Version.

N. Do not retitle any existing section to be Entitled "Endorsements"
   or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections
Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright
resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special
permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.
Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

> Copyright (c) YEAR YOUR NAME.
> Permission is granted to copy, distribute and/or modify this document
> under the terms of the GNU Free Documentation License, Version 1.2
> or any later version published by the Free Software Foundation;
> with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
> A copy of the license is included in the section entitled "GNU
> Free Documentation License".

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

> with the Invariant Sections being LIST THEIR TITLES, with the
> Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.