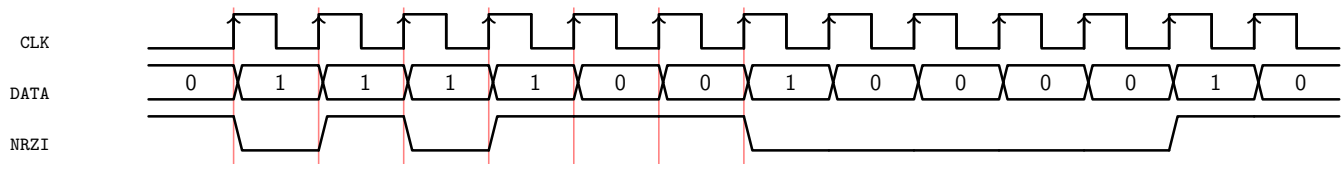
USB - A Study

Narendiran S

03-07-2021

1 Source: Dogan Ibrahim - Advanced PIC Microcontroller Projects in C From USB to RTOS with the PIC 18F Series (2008, Newnes)

- By USB Implementation Forum Inc. (Compaq, Microsfot, Intel, NECT, HP, Lucent, Philips, etc.)
- High-sepped Serial interface
- 127 devices (7-bit address filed – Address 0 is reserverd)
- Thirty meters max
- Maximum tiers (hubs) can be 6
- USB 1.1 – 11 Mbps & USB 2.0 – 480 Mbps
  - Low Speed - 1.5 Mbps
  - Full Speed - 12 Mbps
  - High Speed - 480 Mbps
- White –– Data- & Green –– Data+ carry differential data signals and signle-ended data states.
- USB signals are bi-phase and uses NRZI (Non-Return to Zero Inverted) data encoding.
- NRZI
  - Uses clock.
  - If the data bit is 1, the NRZI transitions at clock boudaries.
  - If the data bit is 0, no transitions.
  - NRZI may have long series of 0s or 1s, resulting in clock recovery difficulties.



- NRZI may have long series of 0s or 1s, resulting in clock recovery difficulties.

- Hence, a concept of `bit stuffing` is used where after every six consecutive ones, a 0 bit is stuffed.
- Packet of data transmitted by host is sent to every device connected to the bus.
- All the devices receive the signal, but only the addressed one accepts the data.
- Only one device can transmit data to the host.
- Device class - enables same device driver to be used for several devices having similar functionalities.
- Device Class

Device Class	**Description**	**Example Devices**
0x00	Reserved	-
0x01	USB Audio Device	Sound Card
0x02	USB Communications Device	Modem, Fax
0x03	USB Human Interface Device	Keyboard, Mouse
0x07	USB Printer Device	Printer
0x08	USB Mass Storage Device	Memory Card, Flash Drive
0x09	USB Hub Device	Hubs
0x0B	USB Smart Card Reader Device	Card Reader
0x0E	USB Video Device	Webcam, Scanner
0xE0	USB Wireless Device	Bluetooth

- Commom USB Terms
  - Endpoint - Either a source or a sink of data. A single USB device can have sixteen IN endpoints and sixteen OUT endpoints.
  - Trasactions - transfer of data on the bus.
  - Pipe - Logic data connection between the host and an endpoint.

1.1 Speed and Device plugged Identification on Bus

- At the device end (client side or endpoint) of the bus, a 1.5K pull-up resistor is connected from D+ or D- to 3.3V.
- For full-speed bus, D+ line is pulled to 3.3V through pull-up resistor.
- For Low-speed bus, D- line is pulled to 3.3V through pull-up resistor.
- When no device is plugged, the host sees a low on the data lines.
- When a device is plugged in, either of the D+ or D- is pulled high, and the host knows that a devie is plugged into the bus.
- The speed is determined by with line is pulled high.

1.2 USB States

USB States	Line Values (D+ and D-)	Description
IDLE	Bus in idle state, one line is high and other line is low (which line is high and which line is low depends on the speed mode(See Section 1.1))	This is the state of lines before and after a packet transmission. Also, the device is plugged in.
DETACHED	Both lines are low.	No devices is plugged/connected.
ATTACHED	Either of the line is logic high.	Devices is plugged/connected.
J State	Same as idle	Same as idle
K State	Opposite of J State	Opposite of J State
SE0	Single Ended Zero State, both lines are pulled low	-
SE1	Single Ended One State, both lines are pulled high	Illegal condition, should never happen
RESET	pulling both the lines low (SE0 State) for at least 10ms.	When host wants to communicate with device on bus, it first sends a reset condition.
EOP	-	End of Packet state - SE0 state for 2 bit times, followed by J State for 1 bit time.
KEEP ALIVE	-	Keep Alive State is achieved after EOP State. Keep Alive is sent at least once every millisecond to keep the device from suspending.
SUSPEND	-	Used to Save Power, Suspend is done by not sendingy anything to a device for 3ms. Suspended device draws less than 0.5mA and must recognie reset and resume signal.
RESUME	Reversing the polarity of the singal on the data lines for at leaset 20ms followed by a low-speed EOP signal	Suspended device is woken using resume by sending resersed polrity signal for 20 ms and then low-speed EOP

1.3 USB Bus Communication

- USB is host-centric connectivity system, so host dictates the use of USB Bus.
- Each device on the bus is assigned a unique USB address.
- No slave device can assert signal on the bus, until the host asks for it.

1.3.1 Initial Operation

i) A new device is plugged in is identified by either of D+ or D- line going high (Generally both lines are low).

ii) When new device is blugged into bus, the USB host uses the address 0 to ask basic information from the device.

iii) Then, host assigns unique USB address.

iv) Then, host ask for and gets infomations such as name of manufacturer, device capacilities, product ID, etc.

v) Then two-way transactions on the bus can begin.

1.4 Packets

- Data is transmitted on a USB bus using Packets.

- A packet starts with a 8-bit sync pattern to allow receiver clock to synchronize with the data.
- Followed by the **Packet Identifier (PID)** byte information.
  - PID byte is a 8-bit long.
  - PID is 4bit + 4-bit coplemented PID.
  - Based on the which PID, four packes are available : *Token, Data, Handshake, Special*

PID type - Packet	Bits	PID name	Description
Data Packet	1100_0011	DATA0	Data Packet PID even
	0100_1011	DATA1	Data Packet PID odd
	1000_0111	DATA2	Data Packet PID high speed
	0000_1111	MDATA	Data Packet PID high speed
Handshake Packet	1101_0010	ACK	Receiver accepts packet (Receiver acknowledges that it has received)
	0101_1010	NAK	Receiver did not accepts packet (Receiver cannot accept the packet)
	1001_0110	NYET	No response from receiver
	0001_1110	STALL	Stalled (Indicates endpoint is halted)
Token Packet	1110_0001	OUT	Host to device transaction
	0110_1001	IN	Device to host transaction
	1010_0101	SOF	Start of frame
	0010_1101	SETUP	Setup Command
Special Packet	1111_0000	Reserved	Reserved
	0111_1000	SPLIT	High-speed split transaction
	1011_0100	PING	High-speed flow control
	0011_1100	ERR	Split transaction error
	0011_1100	PRE	Host preample

- Followed by
  - 0-1024 bytes of data, 2-byte CRC checksum in case of *Data Packet*.
  - 7-bit address, a 4-bit ENDP (Endpoint Number) and a 5-bit CRC checksum in case of *Token Packet*.
  - Nothing in case of *Handshake Packet*.
- Finally, ends with the end of packet (EOP) signal.

Table 1: Data Packet

SYNC	PID	DATA	CRC	EOP
8-bits	4+4 bits	0-1024 bytes	16-bits	-

Table 2: Token Packet

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

Table 3: Handshake Packet

SYNC	PID	EOP
8-bits	4+4 bits	-

1.5 Data Flow Types

Data is transferred in a USB using

1. Bulk transfer
2. Interrupt transfer
3. Isochronous transfer
4. Control transfer

1.5.1 Bulk Transfer

- Transfer lare amounts of data with no errors
- No guarantee of bandwidth
- If an endpoint is defined as OUT and uses bulk tranfer.
  - Then host transfer data from host to endpoint.
- If an endpoint is defined as IN and uses bulk tranfer.
  - Then host transfer data from endpoint to host.
- Used when slow data rate is not an issue.
- No bulk transfer at low speed
- 8-64 packets at full speed
- 512 packets at high speed

1.5.2 Interrupt Transfer

- Transfer small amount of data with high bandwidth.
- 1-8 bytes at low speed
- 1-64 byutes packets at full speed
- upto 1024 bytes at high speed

1.5.3 Isochronous Transfer

- Guaranteed bandwidth with no guaranteed error-free
- Audio data (where speed is important but loss of data is not importnat)
- No Isochronous transfer at low speed
- 1023 bytes at full speed
- 1024 bytes at high speed

1.5.4 Control Transfer

- Bidirectional Data transferred
- using both IN and OUT endpoints.
- for initial configuation of device by host.
- 8-bytes at low Speed
- 8-64 bytes full Speed
- 64-bytes at high speed.

1.6 Enumeration

When a device is plugged into a USB bus, it is known to the host through **Enumeration**. The initial communication between the host and the device is carried out using the *control transfer* type of data flow.

- When a device is plugged in, the host becomes aware of it because one of thedata lines (D+ or D-) becomes logic high.
- The host sends a USB reset signal to the device to place the device in a known state.
- The reset device responds to address 0.
- The host sends a request on address 0 to the device to find out its maximum packet size using a *Get Descriptor* command.
- The device responds by sending a small portion of the device descriptor.
- The host sends a USB reset again.
- The host assigns a unique address to the device and sends a *Set Address* request to the device.
- After the request is completed, the device assumes the new address. At this point the host is free to reset any other newly plugged-indevices on the bus.
- The host sends a *Get Device Descriptor* request to retrieve the complete device descriptor, gathering information such as manufacturer, type of device, and maximum control packet size.
- The host sends a *Get Configuration Descriptors* request to receive the device's configuration data, such as power requirements and the types and number of interfaces supported.
- The host may request any additional descriptors from the device.

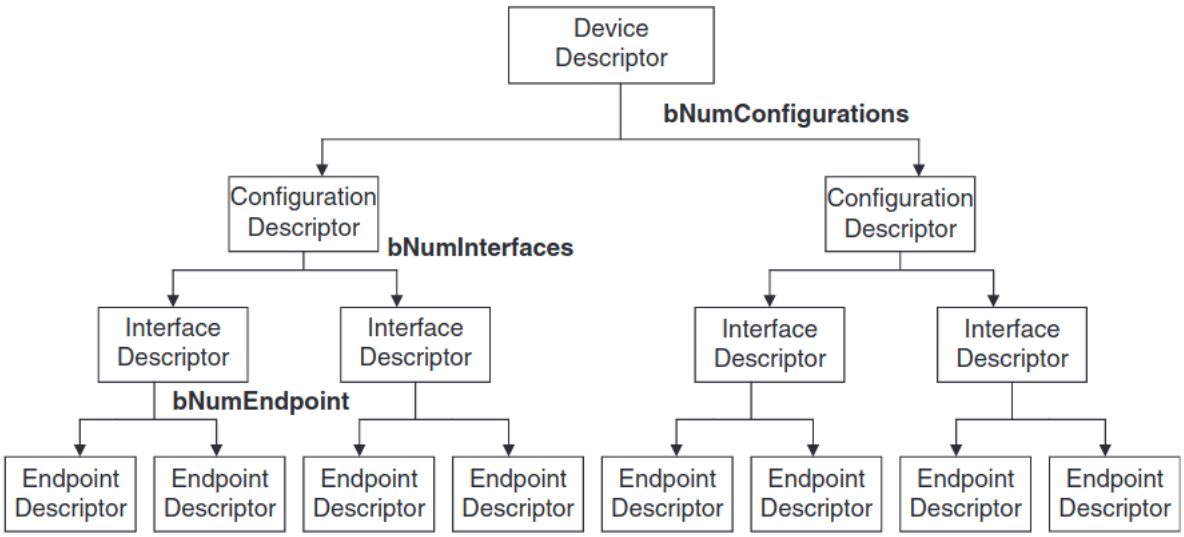
Initially, the device is addressed, but it is in an unconfigured state. After the host gathers enough information about the device, it loads a suitable device driver which configures the device by sending it a *Set Configuration* request. At this point the device has been configured, and it is ready to respond to device-specific requests.

1.7 Descriptors

All USB devices have a hierarchy of descriptors that describe various features of the device: the manufacturer ID, the version of the device, the version of USB it supports, what the device is, its power requirements, the number and type of endpoints, and so forth. The commonly used descriptors are

- Device descriptors
- Configuration descriptors
- Interface descriptors
- HID descriptors
- Endpoint descriptors

The hierarchy can be seen below



The HID descriptor comes in same level as Interface descriptor when interface belogs to HID class.

The first byte of all descriptor is the *bLength* - length of descriptor. The second byte of all descriptor is the *bDescriptorType* - type of descriptor.

1.7.1 Device Descriptor

- Top-level information read from device.
- First item, the host retrieves
- USB device has only one Device Descriptor
- Represents entier device
- information such as manufacturer, serial number, product number, the class of the device, and the number of configurations

Offset (in Bytes)	Field	Size (in Bytes)	Meaning	Description
0	bLength	1	Descriptor Size in bytes	Provides the length of this device descriptor in bytes.
1	bDescriptorType	1	Device Descritor type	Value is 0x01 since, the type is device descriptor.
2	bcdUSB	2	Highest version of USB supported	Provides the highest usb version supported by the device in BCD format, 0xJJMN (JJ - major version, M - minor version, N - subminor version)
4	bDeviceClass	1	Class Code	assigned by the USB organization and used by system to find a class driver for device.
5	bDeviceSubClass	1	Subclass code	
6	bDeviceProtocol	1	Protocol code	
7	bMaxPacketSize0	1	Maximum Packet size	For Endpoint 0, maximum input and output packet size
8	idVendor	2	Vendor ID	Vendor's ID assigned by USB organization
10	idProduct	2	Product ID	Product's ID assigned by manufacturer
12	bcdDevice	2	Device release number	Deivce release number - same format as bcdUSB
14	iManufacturer	1	Manufacturer string descripto	details about the manufacturer andthe product. These fields have no requirement and can be set to zero.
15	iProduct	1	Index of product string descriptor	
16	iSerialNumber	1	Index of serial number descriptor	
17	bNumConfigurations	1	Number of possible configurations	Number of configuation the device supports

1.7.2 Configuration Descriptors

- information about power requirements of the device, how many different interfaces it support
- may have more than one configuration for a device
- When the configuration descriptor is read by the host, it returns the entire configuration information, which includes all interface and endpoint descriptors.

Offset (in Bytes)	Field	Size (in Bytes)	Meaning	Description
0	bLength	1	Descriptor Size in bytes	Provides the length of this configuation descriptor in bytes.
1	bDescriptorType	1	Device Descritor type	Value is 0x02 since, the type is configuation descriptor.
2	wTotalLength	2	Total bytes returned	total combined size of this set of descriptors (configuration descriptor + interface descriptor + HID descriptor + endpointdescriptor)
4	bNumInterfaces	1	Number of interfaces	number of interfaces present for this configuration
5	bConfigurationValue	1	Value used to select configuration	used by the host (in command SetConfiguration) to select the configuration.
6	iConfiguration	1	Index describing configuration string	index to a string descriptor describing the configuration inreadable format.
7	bmAttributes	1	Power supply attributes	D7==1 – Bus-powered, D6==1 – Self Powered, D5==1 – Remote wakeup
8	bMaxPower	2	Max power consumption in 2mA	maximum power the device will draw from the bus in 2mAunits.

1.7.3 Interface Descriptors

- Specify the class of interface and the Number of endpoint it uses
- May be more then one interface.

Offset (in Bytes)	Field	Size (in Bytes)	Meaning	Description
0	bLength	1	Descriptor Size in bytes	Provides the length of this interface descriptor in bytes.
1	bDescriptorType	1	Device Descritor type	Value is 0x04 since, the type of this interface descriptor.
2	bInterfaceNumber	1	Number of interface	index of this interface descriptor
3	bAlternateSetting	1	Value to select alternate setting	pecify alternate interfaces that can be selected by the host using command Set Interface.
4	bNumEndpoints	1	Number of endpoints	number of endpoints used by this interface.
5	bInterfaceClass	1	Class code	assigned by USB organization
6	bInterfaceSubClass	1	subclass code	
7	bInterfaceProtocol	1	protocol code	
8	iInterface	1	Index of string descriptor to interface	index to a string descriptor of the interface.

1.7.4 HID Interface

An HID descriptor always follows an interface descriptor when the interface belongs tothe HID class.

Offset (in Bytes)	Field	Size (in Bytes)	Meaning	Description
0	bLength	1	Descriptor Size in bytes	Provides the length of this HID descriptor in bytes.
1	bDescriptorType	1	Device Descritor type	Value is 0x21 since, the type of this HID descriptor.
2	bcdHID	2	HID class	HID class specification
4	bCountryCode	1	Special country dependent code	specifies any special local changes
5	bNumDescriptors	1	Number of additional descriptors	specifes if there are any additional descriptors associated with thisclass.
6	bDescriptorType	1	Type of additional descriptor	type of the additional descriptor specified inbNumDescriptors.
7	wDescriptorLength	2	Length of additional descriptor	ength of the additional descriptor in bytes.

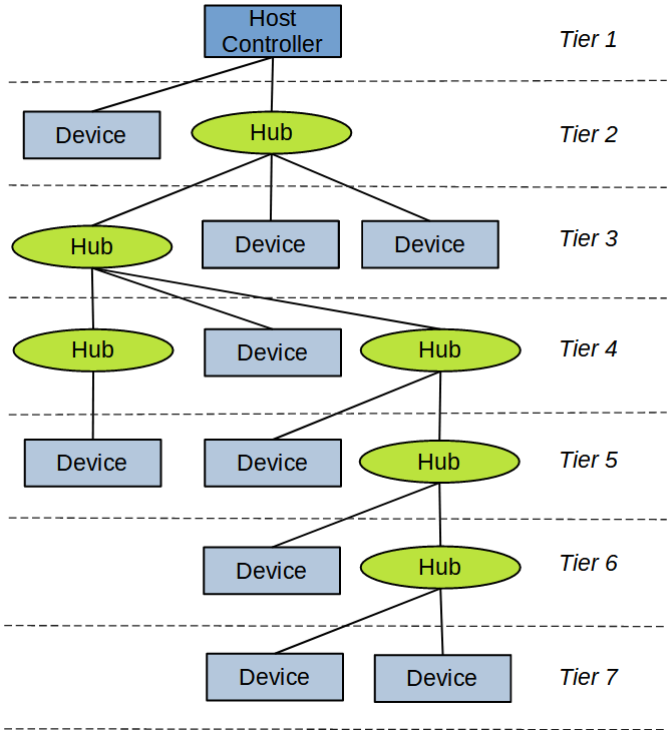
1.7.5 Endpoint Descriptors

Offset (in Bytes)	Field	Size (in Bytes)	Meaning	Description
0	bLength	1	Descriptor Size in bytes	Provides the length of this Endpoint descriptor in bytes.
1	bDescriptorType	1	Device Descritor type	Value is 0x21 since, the type of this Endpoint descriptor.
2	bEndpointAddress	1	Endpoint address	address of the endpoint.
3	bmAttributes	1	Type of endpoint	specifies what type of endpoint it is.
4	wMaxPacketSize	2	Max packet size	maximum packet size
6	bInterval	1	Polling interval	how often the endpoint should be polled (in ms)

2 Source : USB MADE SIMPLE

2.1 Introduction

- Support hot-pluggable, Specific Interupts or DMA
- Has OTG (On-the-Go), extension of USB spec to allow devices to become limited role host.
- Tiered Star topology - single host controller and upto 127 slave devices.
- Maximum tier is 6.
- Length of cable is 5 meters.



- Three types of USB Host controller.
- OHCI (Open Host Controller Interface)
  - Compaq, Microsoft and National Semiconductors cooperated to produce this standard host controller specification for USB 1.0 and USB 1.1.
  - Hardware Oriented
  - Low speed and full Speed
- UHCI (Universal Host Controller Interface)
  - Intel's software-orientd of controller for USB 1.0 and 1.1
  - Requieres license from Intel
  - Low speed and full Speed
- EHCI (Extended Host Controller Interface)
  - USB 2.0 with high speed for a single host controller specs.
  - EHCI handless high speed and hands off low and full speed to either OHCI or UHCI.
- Four Wiere - VBUS [Red] (5V) , GND [Black] (reference), D+ [Green], D- [White]
- Carry differential data signal or single-ended signal states.
- The D+ or D- resistor should, strictly speaking, be pulled up to a 3.3V supply derived from Vbus, or controlled by Vbus in such a way that the resistor never sources current to the data line when Vbus is switched off.

2.2 SPeed Identification

- At device end, a 1.5 kohm resistor pulls one of the lines to 3.3V supply derived from VBUS.
- Low speed : D- has the pull up resistor
- Full speed : D+ has the pull up resistor
- High speed : D+ has the pull up resistor
- host determine the speed by observering the lines.

2.3 Transceivers

- Transceivers differe mainly by resistors.
- Upstream is the host end, where a 15k pull down resistor pulls the line low.
- Maximum high-leve is 3.3V

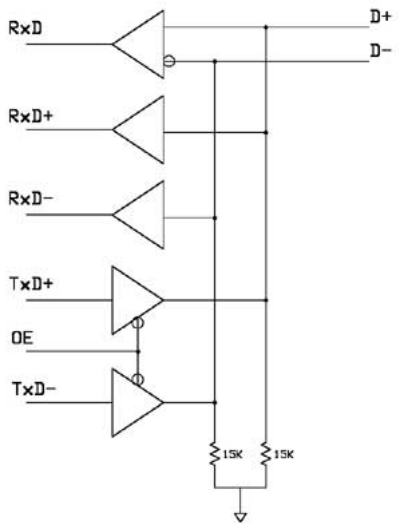


Figure 1: Upstream Host Transceiver

- Downstream is at device end, where eithere there is a pull up resistor of 1.5kOhm at D+ or D- to make then high.
- Can be able to detect states (such as SE0 - both lines are low)

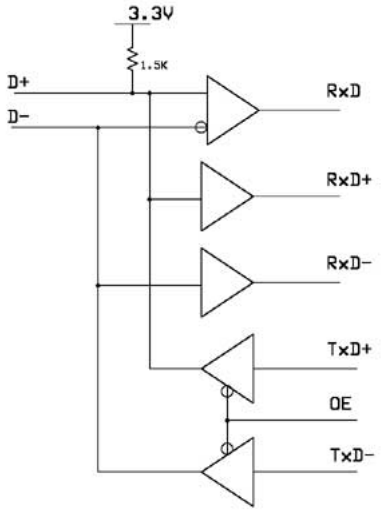


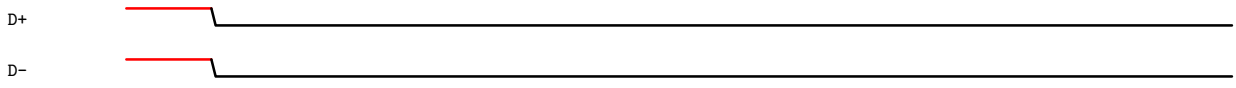
Figure 2: Downstream device Transceiver (Full Speed)

2.4 Line States

Based the 2 data lines, different line states are obtained.

2.4.1 Detached

When no device is plugged in, the host will see line low due to 15kohm pulling the lines low.



2.4.2 Attached

Either of D+ or D- goes to high level to indicated the device is plugged in.

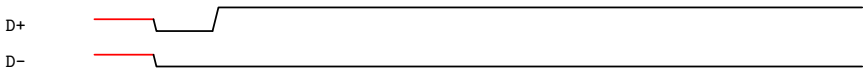


Figure 3: Full SPeed

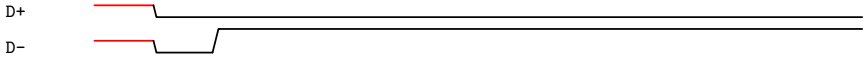


Figure 4: Low SPeed

2.4.3 Idle

When the pulled up line is high and other line is low. State of lines before and after packets are sent.

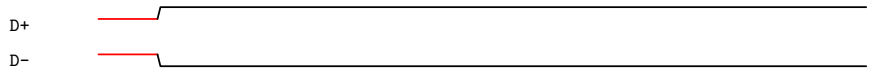


Figure 5: Full SPeed

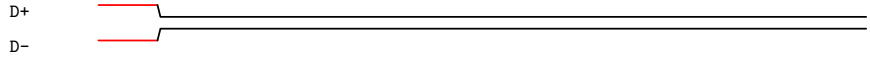


Figure 6: Low SPeed

2.4.4 J and K State

J state is same as Idle state, but the lines are driven by eigher host or device. Opposite to J state. J and K are used for full and low speed links as they are Opposite polarity.

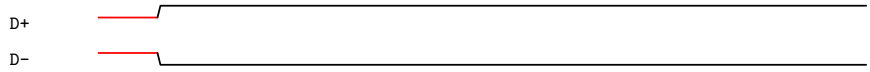


Figure 7: J state

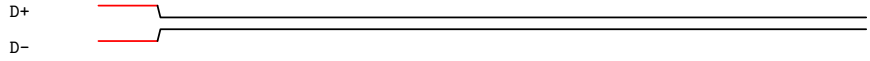
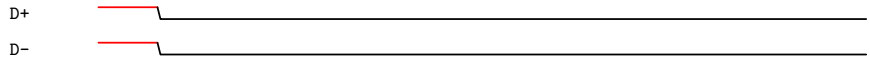


Figure 8: K state

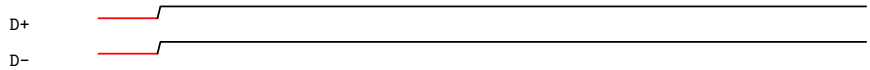
2.4.5 SE0 state (Single Ended Zero)

Both lines are pulled low. Driven low.



2.4.6 SE1 state (Single Ended One)

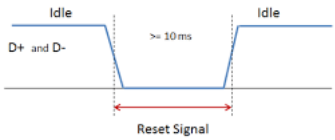
Both lines are high. Illegal, not happen.



2.4.7 Reset

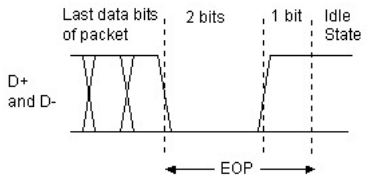
When host wants to communicate it will start by applying Reset condition, which will set device to default config.

- Pulling down lines to low for 10 ms (SE0 state)
- Device may recognie reset after 2.5 μs



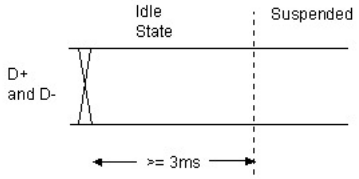
2.4.8 EOP - ENd of Pack

EOP is SE0 for 2 bit time, and J state for 1 bit time.



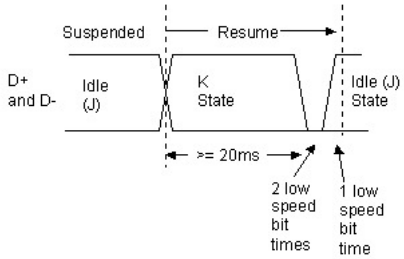
2.4.9 Suspend

Not sending any signal for 3 ms. (IDLE state) Normally a SOF packet (at full speed) or a Keep Alive signal (at low speed) is sent by the host every 1 ms, and this is what keeps the device awake.



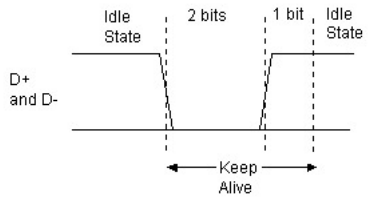
2.4.10 Resume

Wake up device which is in suspend. Reverse the polarity of the lines for atlease 20ms. Followed by low speed EOP signal.



2.4.11 Keep Alive signal

Low speed EOP. Atleaset every 1ms on low speed link to keep the device from suspending.



2.5 Packets and Transmission

- Packets - smallest element of data transmission.
- Before and after Packet, the bus is idle.
- **Serial Interface Engine (SIE)** takes care of SYNCs, Bit Stuffing, and EOP conditoions.
- USB uses LSB first. (also in a 2-byte data, LSB byte is transmitted first)
- Packets starts with a SYNC pattern to synchronize with the receiver bit clock.
- Followed by data packets and ends with EOP signal.
- NRZI encoding
- to ensure sufficent frequent transitions - Bit Stuffing (zero after 6 sucessive 1's) is done.

2.6 Definitions

2.6.1 EndPoints

- Each USB device has a number of endpoints.
- Each endpoint is a source or a sink of data.
- A device can have upto 16 OUT and 16 IN endpoints
- OUT means from host to device.
- IN means device to host.
- Endpoint 0 is combination of endpoint0 OUT and endpoint0 IN and is used for controlling device.

2.6.2 Pipes

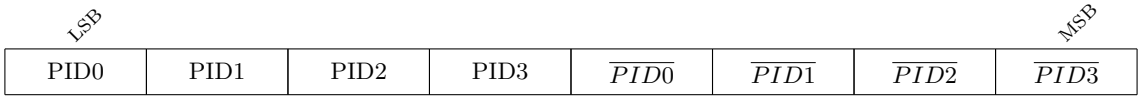
- A logical connection between host and a particular endpoint.

2.6.3 Transactions

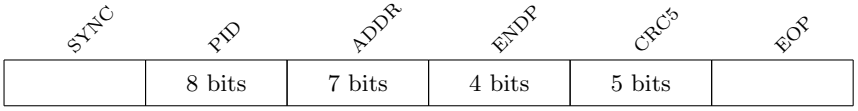
- Simple transfer of data called Transactions built up using packets.

2.7 Packet FOrmat

- The first byte of every packet is the Packet Identifier (PID).
- To make it easy for the SIE, there is no CRC check for PID.
- USB uses two CRCs - 5-bit CRC (CRC5) and 16-bit CRC (CRC16)
- First two bits of PID determines the groups in which they fall into.
- Even Here LSB's are transmitted first, hence, PID0 is transmitted first, followed by PID1, PID2, PID3, $\overline{PID0}$, $\overline{PID1}$, $\overline{PID2}$, $\overline{PID3}$

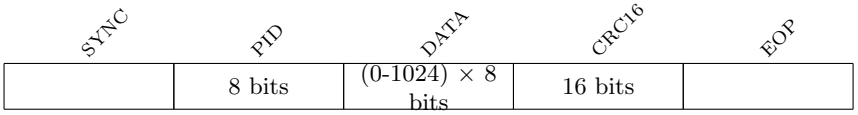


2.7.1 Token Packet



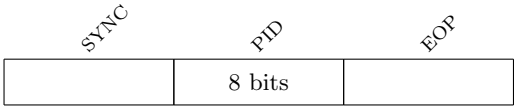
- User for SETUP, OUT, IN packets.
- Transmitted first in a transaction - identify the endpoint and purpose of transacting.
- Address is 7-bit can so can address upto 127 adddressed using USB (Address 0 is reserved).
- Endpoint can be upto 16-possible values in each directions(IN or OUT).
- The direction (IN or OUT) is said by PID.
  - OUT and SETUP PID will refer OUT endpoint.
  - IN PID will refere IN endpoint.

2.7.2 Data Packets



- Used for DATA0, DATA1, DATA2, MDATA packet.
- For data stage inside a transaction.
- DIfferent DATA for error-checkng system.
- DATA0 and DATA1 is used in Low and FUll speed links.
  - DATA0 and DATA1 packets comes alternatively when sending data.
  - This helps endpoint to know which one to expect.
- DATA2 and MDATA are used for high speed links.

2.7.3 Handshake Packet



- USed for ACK, NAK, STALL, NYET.
- status stage of a transcation.
- ACK - reciever acknowledges error free packet.
- NACK - receiver cannot accept data or tranmitting device cannot send data.
- STALL - endpoint is halted
- NYET - No response yet from receiver (high speed only)

2.7.4 SOF Packet

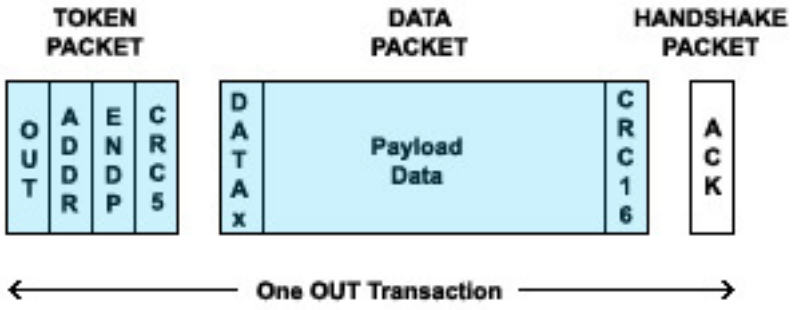
- Only on full speed links.

2.8 Transactions

- A transaction (successfull) is a sequence of three packets which perform a simple transfer of data.
- For IN and OUT transaction used for isochronous transfers, only 2 tpackets are used, the handshake packet at the end is omitted.
- This is because error-checking is not required.

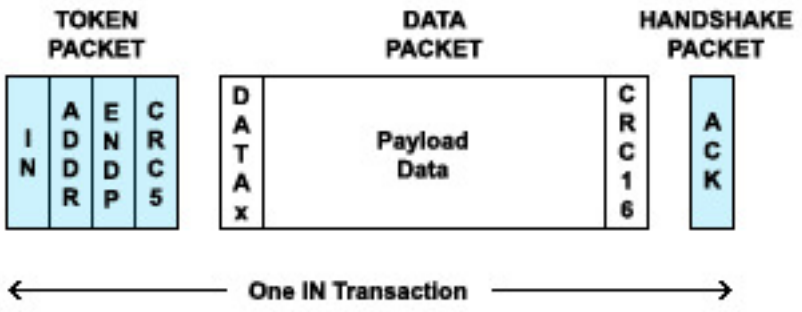
2.8.1 OUT Transaction

- A successfull OUT transaction comprises two or three sequential packets.
- FOr isochronous transfer, only two packets and handshake packet from the device is obmitted.
- On Low or full speed link, DATAx is either DATA0 or DATA1 alternating between them.



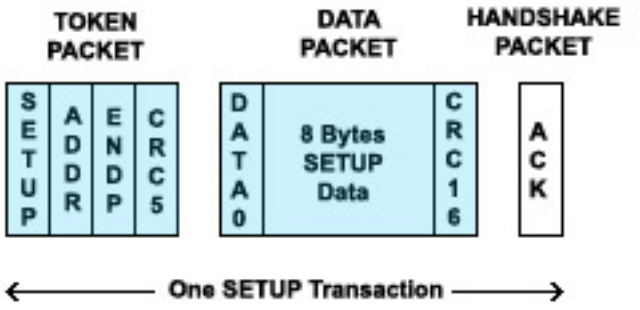
2.8.2 IN Transaction

- A successfull IN transaction comprises two or three sequential packets.
- FOr isochronous transfer, only two packets and handshake packet from the host is obmitted.
- On Low or full speed link, DATAx is either DATA0 or DATA1 alternating between them.



2.8.3 SETUP Transaction

- A successfull IN transaction comprises three sequential packets.
- similar to OUT transaction, but data payload is exactly 8-bytes long.
- SETUP PID in token packet tells the de4vice thiat this is the first transaction in COntrol transfer.
- SETUP transaction always use DATA0



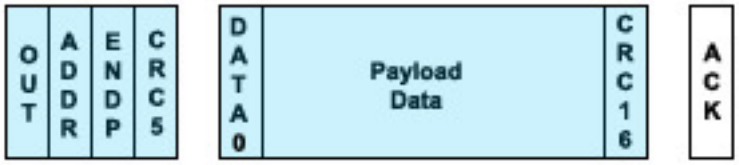
2.9 Data Flow Types

- Four Different ways to transfer data on a USB bus.
- Each has its own purpose and characteristics.
- Each one is built up using one or more transaction type.

2.9.1 Bulk Transfers

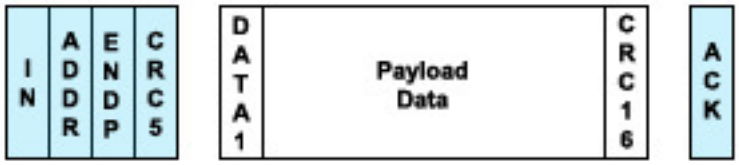
- Transfer large amounts of data with error-free but no guanrantee of bandwidth.
- Host will *schedule* bulk transfers after other transfer types have been allocated.
- If an OUT endpoint is defined as using Bulk transfers, then the host will transfer data to it using OUT transactions.
- If an IN endpoint is defined as using Bulk transfers, then the host will transfer data from it using IN transactions.
- The max packet size is 8, 16, 32 or 64 at full Speed and 512 for high speed.
- Bulk transfers are not allowed at low speed.
- See topic : Bulk Transfers on more details if needed.

Shown below is a Bulk transfer of OUT transactions in OUT endpoint.



2.9.2 Interrupt Transfer

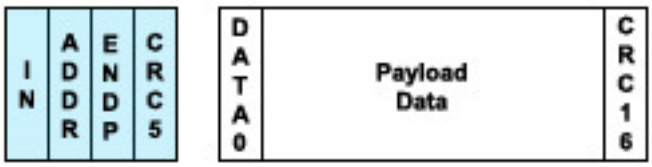
- Regularly scheduled IN or OUT transactions. (mosly IN directions)
- Host fetches one packet from device at an interval specifed by the endpoint descriptor.
- Use an interrupt transfer when you need to be regularly kept up to date of any changes of status in a device.



2.9.3 Isochronous Transfers

- Guaranteed bandwith, but no error-free.
- Applications of audio data transfer.
- either an In or OUT transcation depending on the type of endpoint.
- No need handshake packet at the last.
- may containe upto 1023 bytes at full speed.
- 1024 at high speed.
- No transfer at low speed.

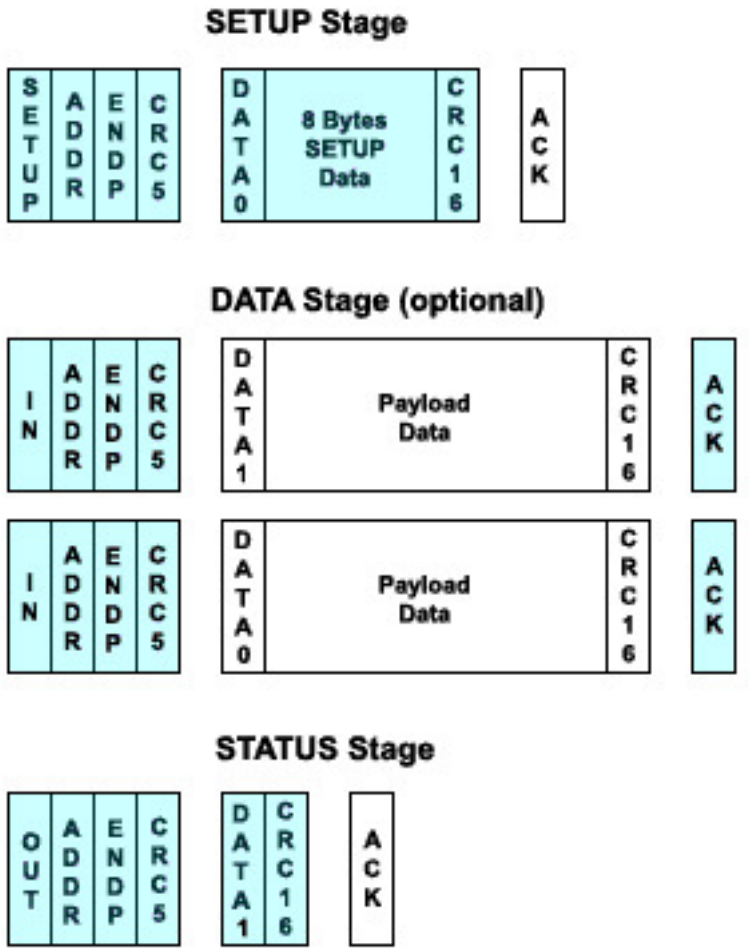
Shown below, is a isochronous transfer of IN transaction.



2.9.4 Control Transfer

- Bi-direction transfer which uses both IN and OUT endpoint.
- Each control transfer must have 2 or more transaction (the 2 transactions beeting SETUP stage (SETUUP/OUT transaction) and STAUTS stage)
- Has three stages.
  - SETUP stage carries 8 bytes called the setup packet.
  - defines the request and specfies how much data should be transffered in DATA stage.
  - DATA stage is optional,
  - But if data stage is preset , it starts with DATA1 and alternates between DATA0 and DATA1. untill all data is transffered.
  - The STAUS stage is a transaction containing a zero-lenght DATA1 packet.
  - If DATA stage was IN, then STATUS stage is OUT and vice versa.
- Control transfers are used for initial configuration of the device by the host.
- Uses the Endpoint 0 OUT and Endpoint 0 IN.

Shown below is a control transfer with data stage as IN.



2.10 Controlling Device

- When USB devices is plugged in, the host becomes aware that a device is plugged in (because of the pull resistor on one of the data lines)
- The host now signals a USB reset to the device to makes the device enter a known state at the end of reset.
- In reset state, the device responds to default address of 0.
- Now, the host will sedn a request to **Endpoint 0** with device address 0 to find out its maximum packet size.
- It can discover this using `Get_Desriptor` command.
- The device responds on address 0.
- The host now will reset the device again.
- Then it sends a `Set_Address` request with adress 0.
- After request is complete, the devices assumes the new address.
- Now, the host gets details from the device using the
  - `Get_Device` Descriptor
  - `Get_Configuration` Descriptor
  - `Get_String` Descriptor
- Once, the host gets the enough details, the host loads the suitable device driver.
- The device driver will then select a configuration for the device by sending a `Set_Configuration` request to device.
- Now, the device is in configured state.
- Now, it can respond to device specific request.
- THese are done using Control transfers on Endpoint 0 IN and OUT.