Introduction to USB

- Introduction
- Pin Out
- USB 2.0 Standard
- USB Topology
- USB Device attachment & speed detection
- USB Device Power
- USB Suspend mode
- USB Transaction
- USB Transfer
- USB Control Transfer
- USB Bulk Transfer
- USB Interrupt Transfer
- USB Isochronous transfers
- Host constraints for Interrupt & Isochronous Transfers
- USB High-Speed mode specific features PING/NYET Protocol
- USB High-Speed mode specific features SPLIT Protocol
- LINK

NOTE:
This short introduction to USB was made using info from WikipediA and Developing USB applications using the STM32 ARM Cortex-M3 microcontroller by:Anis BEN ABDALLAH

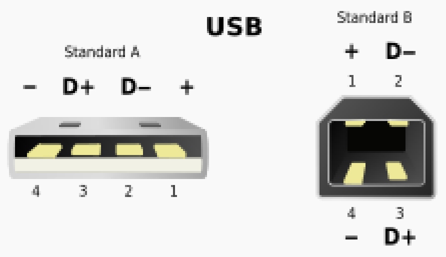
Introduction

Universal Serial Bus (USB) is an industry standard developed in the mid-1990s that defines the cables, connectors and communications protocols used in a bus for connection, communication, and power supply between computers and electronic devices.

USB was designed to standardize the connection of computer peripherals (including keyboards, pointing devices, digital cameras, printers, portable media players, disk drives and network adapters) to personal computers, both to communicate and to supply electric power. It has become commonplace on other devices, such as smartphones, PDAs and video game consoles.
USB has effectively replaced a variety of earlier interfaces, such as serial and parallel ports, as well as separate power chargers for portable devices.

Pin Out

Pin out



The standard-A USB plug (left) and standard-B plug (right)

Pin 1	V_{CC} (+5 V, red wire)
Pin 2	Data− (white wire)
Pin 3	Data+ (green wire)
Pin 4	Ground (black wire)

For more details regarding the USB connectors see here.

WikypediA source

USB 2.0 Standard

- USB 2.0 speeds
  - Low speed: **1.5** Mbits/s
  - Full speed : **12** Mbits/s
  - High speed: **480** Mbits/s

- USB keeps high compatibility at protocol level between all supported speeds

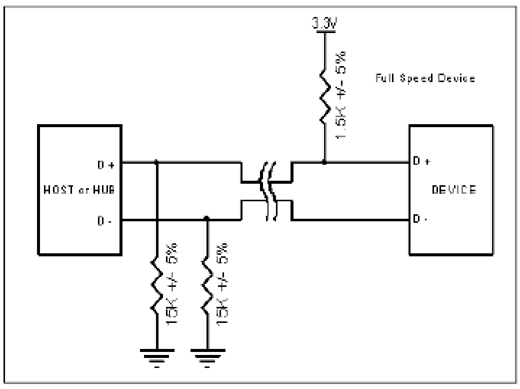
- Bus components
  - **USB host** or **Root hu**b: initiates all the transaction on the bus
  - **USB function**: is a device with one or more interfaces that expose capabilities to the host (ex: mouse, keyboard,..)
  - **USB hub**: allows to connect multiple devices to the USB host. It has an upstream port for communication with the host and multiple downstream ports for direct connection to devices

USB Topology

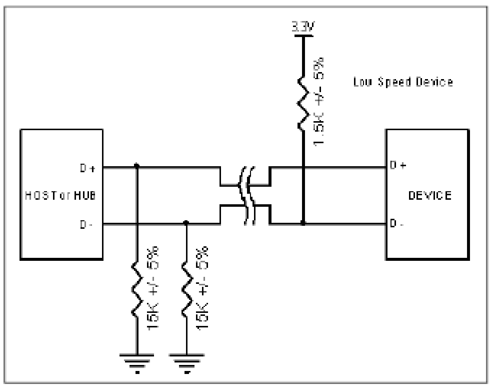
- USB bus has a Tiered **Star topology**
- At the center of each star is a hub with functions as end connections
- A maximum of **127 devices** can be connected in the bus
- A maximum of **5 hubs** can be connected in series the maximum cable length is **5 meter**

USB Device attachment & speed detection

- The 1.5K pull-up allows the host to detect the device attachment and its supported speed
- High-speed device is detected first as full-speed device then high-speed capability is detected through bus handshake mechanism called “chirp sequence”



Full/high Speed: Pull-up on D+



Low Speed: Pull-up on D-

USB Device Power

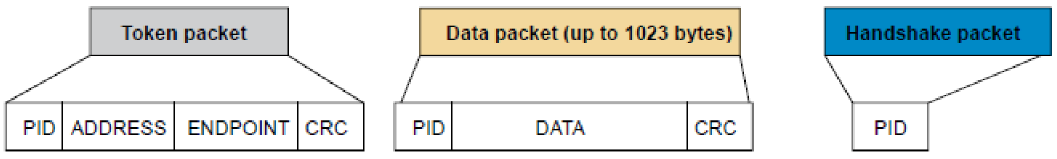
- Two possible power configurations
  - **Self-powered**: device power provided from external power-supply
  - **Bus-powered**: power provided from VBUS (5v)
- For bus-powered device, two options are possible:
  - **Low-power devices**: maximum power consumption is **100mA**
  - **High-power devices**: maximum power consumption is 100mA during bus enumeration and **500mA** after configuration
- **During device enumeration**, the device indicates to host its **power configuration** (self-powered/bus-powered) and its **power consumptio**n in the **device configuration descriptor**

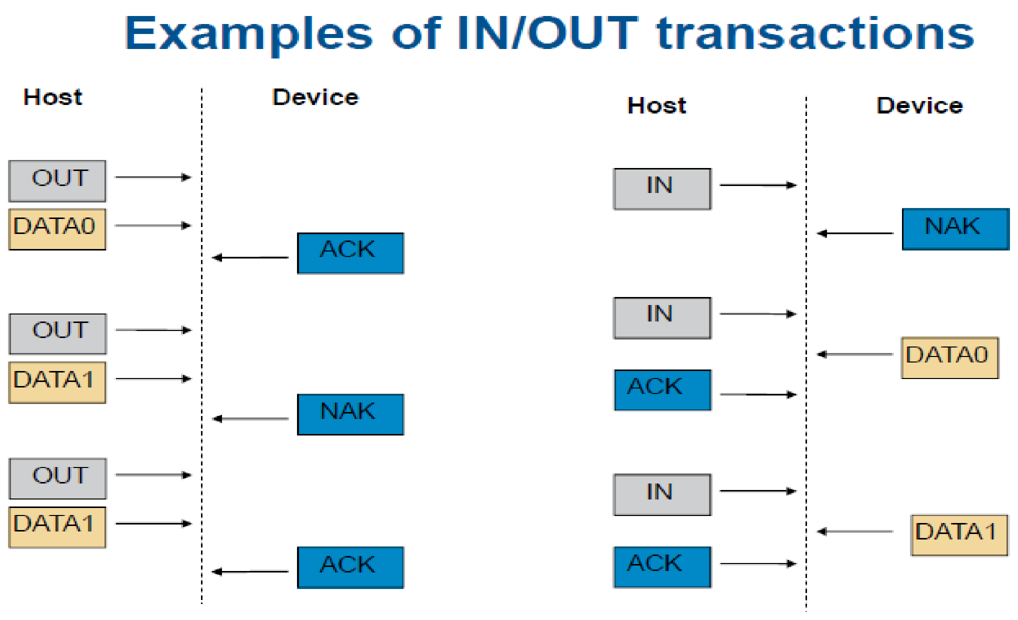
USB Suspend mode

- USB device should enter in **USB Suspend mode when the bus is in idle state for more than 3 ms**
- In suspend mode, when the device is bus powered, the current drawn from VBUS power **shouldn't exceed 2.5mA**
- USB host prevents device from entering in suspend mode by periodically issuing Start of Frame (SOF) or Keep Alive for LS
  - For High-speed, SOF is sent every micro-frame: 125us +/- 65ns
  - For Full-speed, SOF is sent every frame: 1ms +/- 500ns
  - For Low-speed, Keep Alive (End of Packet) is sent every 1ms in absence of low-speed data
- Exist from Suspend mode can be
  - Initiated from host by issuing the resume signaling
  - Initiated from device by issuing the remote wakeup signaling

USB Transaction

- **Token packet** (SETUP, IN, OUT) always issued by the host, includes:
  - PID (IN: Device to host data transaction or SETUP/OUT: host to device data transaction)
  - Target device address
  - Target endpoint number CRC
- **Data packet** (DATA0, DATA1, DATA2, MDATA) includes:
  - PID: DATA0, DATA1, DATA2 or MDATA (DATA2 and MDATA are used only in HS mode)
  - Carries the data payload of a transaction sent by the host or device
  - DATA PID toggle used to synchronize HOST and DEVICE to avoid repeated packet transfer in case of corrupted or lost handshake
  - CRC
- **Handshake packet** (ACK, NAK, STALL, NYET)
  - ACK: packet reception acknowledged (sent from host or device)
  - NAK: packet reception not acknowledged (sent from device only)
  - STALL: control request not supported or endpoint halted (sent from device only)
  - NYET: device not ready to accept further packets (only for high-speed device)



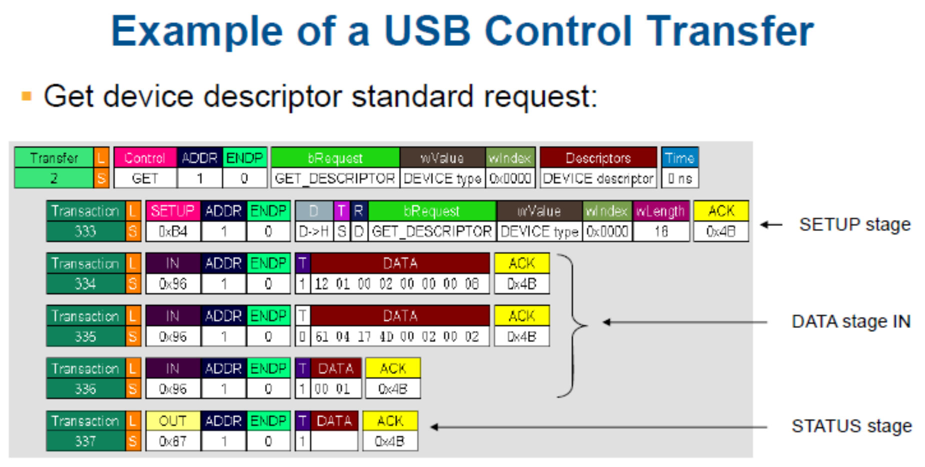


USB Transfer

- A USB transfer is composed of one or multiple bus transactions
- **Four types of USB transfers are defined:**
  - **Control**: used for control and device configuration requests (ex: device enumeration)
  - **Bulk**: used for data transfers with no guaranteed delivery rate (ex: printer, mass-storage drive,..)
  - **Interrupt**: used for devices that need to be polled periodically for data transfers (ex: mouse, keyboard, joystick)
  - **Isochronous**: used for data streaming applications, that requires a guaranteed delivery rate, but no error checking (ex: audio, video devices)
- During each frame (in LS/FS) or micro-frame (in HS), the host will schedule the need transfers with different bandwidth allocation for each transfer type

USB Control Transfer

- Used for standard control requests during device enumeration process or during class operation
- **All devices should support control transfer through endpoint 0** (bidirectional)
- It is given reserved bus bandwidth for 10% for FS/LS and 20% for HS
- Control transfer has 3 stages
  - **SETUP** stage: one SETUP transaction for issuing the control request (ex: Get Descriptor)
  - **Optional DATA** stage IN or OUT: one or multiple data transactions
    The maximum data packet size during the optional data stage is 8 bytes for LS and 64 bytes in FS/HS
  - **Status** stage: one IN or OUT transaction with a Zero Length data packet to check if control transfer request executed correctly or not.
- Transfer error management done through handshake packet and data PID toggle mechanism



Example of a USB Control Transfer

- Get device descriptor standard request:

Transfer	L	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	Time
2	S	GET	1	0	GET_DESCRIPTOR	DEVICE type	0x0000	DEVICE descriptor	0 ns

Transaction	L	SETUP	ADDR	ENDP	D	T	R	bRequest	wValue	wIndex	wLength	ACK
333	S	0xB4	1	0	D->H	S	D	GET_DESCRIPTOR	DEVICE type	0x0000	18	0x4B

Transaction	L	IN	ADDR	ENDP	T	DATA	ACK
334	S	0x96	1	0	1	12 01 00 02 00 00 00 08	0x4B
335	S	0x96	1	0	0	61 04 17 4D 00 02 00 02	0x4B
336	S	0x96	1	0	1	00 01	0x4B

Transaction	L	OUT	ADDR	ENDP	T	DATA	ACK
337	S	0x87	1	0	1		0x4B

USB Bulk Transfer

- Used to **transfer large amount of data** without guaranteed delivery rate (sending data to printer, drive,..)
- Lowest priority transfer with no reserved bus bandwidth but can occupy the full bandwidth if no other transfer on the bus
- Supported only by **full-speed** and **high-speed** devices
- Consist of one or more IN or OUT transactions during each frame/micro-frame (unidirectional)
- The max packet size is **64 bytes for FS** and **512 bytes for HS**
- Transfer error management done through handshake packet and data PID toggle mechanism

USB Interrupt Transfer

- Interrupt transfers are used to poll devices to determine if they have data that needs to be transferred (mouse, keyboard,..)
- Interrupt IN or OUT data transfers are scheduled periodically within a maximum polling period negotiated during device enumeration but host is free to initiate more IN/OUT transactions if there is bandwidth available
- Limited reserved bandwidth for Low/Full speed devices
  - For **low-speed** the packet max length is **8 bytes** with a guaranteed maximum latency of up to 1 packet each 10 frames => **800 Bytes/s**
  - For **full-speed** the packet max length is **64 bytes** with a guaranteed maximum latency of up to 1 packet each frame => **62.5 KBytes/s**
- High bandwidth with high-speed
  - For **high-speed** the packet max length is **1024 bytes** with up to 3 packets each micro-frame
- Transfer error management done through handshake packet and data PID toggle mechanism

USB Isochronous transfers

- Used mainly for streaming real-time data like audio and video
- Needs a guaranteed bandwidth with a constant transfer rate but there is no error checking
- The requested bandwidth is negotiated between host and device during enumeration
- Transfer is in one direction and can consist of one or more data OUT or IN transactions with no handshake packet
  - In **FS**, the max packet length is **1023 bytes** with a maximum of **one packet per frame**

  - In **HS**, the max packet length is **1024 bytes** with a maximums of **3 packets per frame**
- Devices that use isochronous transfer need in most of the cases to establish a synchronous connection (ex: speaker, microphone, video camera,...)
  - Minimal or no data buffering
- The synchronization between the data source (producer) and the sink (consumer) can be achieved by
  - Having the source and sink clocks synchronized to the SOF packet
  - Doing a clock adaptation either on the source

Host constraints for Interrupt & Isochronous Transfers

- The host may not be able to provide the requested bandwidth to device, in this case the host will try other possible configurations with lower bandwidth requirements (if provided by the device)
- If still no bandwidth available, the host will refuse device configuration
- Host software may have some latency for processing data and issuing transfer requests on time due to other processes taking CPU time
- In order to avoid multiple SW calls for handling data to be transmitted or received, large chunks of data transfers should be scheduled

USB High-Speed mode specific features PING/NYET Protocol

- For control and bulk transfers, when a high-speed device is not ready to receive further data OUT packets, it can send the **NYET** handshake
- When the host receives the **NYET** handshake, it should send the **PING** packet periodically to check if device is ready or not to resume receiving data packets
- When **ACK** is received for a PING request, the host will resume sending data packet

USB High-Speed mode specific features SPLIT Protocol

- The **SPLIT** protocol is used when the **HS host need to communicate with a low/full speed device** which is **connected to a high-speed hub**
- The host will do the data transaction with the HS hub in high-speed, then the hub as a host will initiate the same transaction in low/full speed with the device
- The data transaction done by the host with the HUB is preceded with the Start SPLIT (SSPLIT) token
- The host will later use CSPLIT token to retrieve the device response from the HUB

LINK

- Universal Serial Bus Revision V.2.0 specification
- WikipediA USB
- Developing USB applications using the STM32 ARM Cortex-M3 microcontroller by:Anis BEN ABDALLAH