

# USB

Ing. Pablo Martín Gomez  
pgomez@fi.uba.ar

# USB - Historia

- Introducido y estandarizado por un grupo de compañías (Compaq, DEC, IBM, Intel, Microsoft, NEC, HP, Lucent, Philips y Nortel) en 1995
- La idea fundamental fue la de reemplazar la gran cantidad de conectores disponibles en la PC's simplificando la conexión y configuración de dispositivos logrando grandes anchos de banda



# USB - Historia

- Existen 3 versiones de USB
  - USB 1.0 Enero 1996
    - Velocidades de 1.5 Mbps hasta 12 Mbps
  - USB 1.1 Septiembre 1998
    - Primer versión popular de USB
  - USB 2.0 Abril 2000
    - La principal mejora es la inclusión de una tasa de transferencia de alta velocidad de 480 Mbps
  - USB 3.0 Noviembre 2008
    - Tasa de transferencia de 5 Gbps



# USB 3.0



- Upgrade del USB 2.0
  - retro-compatible
- También llamado “SuperSpeed” USB por la  
significante mejora respecto a especificaciones USB  
existentes
- Nuevo protocolo de comunicación para dispositivos
- Nuevos modos de transferencia
- Nuevas formas de administrar la alimentación
- Mayor longitud de cable permitida
- Similar a al tecnología PCI Express 2.0



# USB 2.0 vs. USB 3.0 – Hardware

## USB 2.0

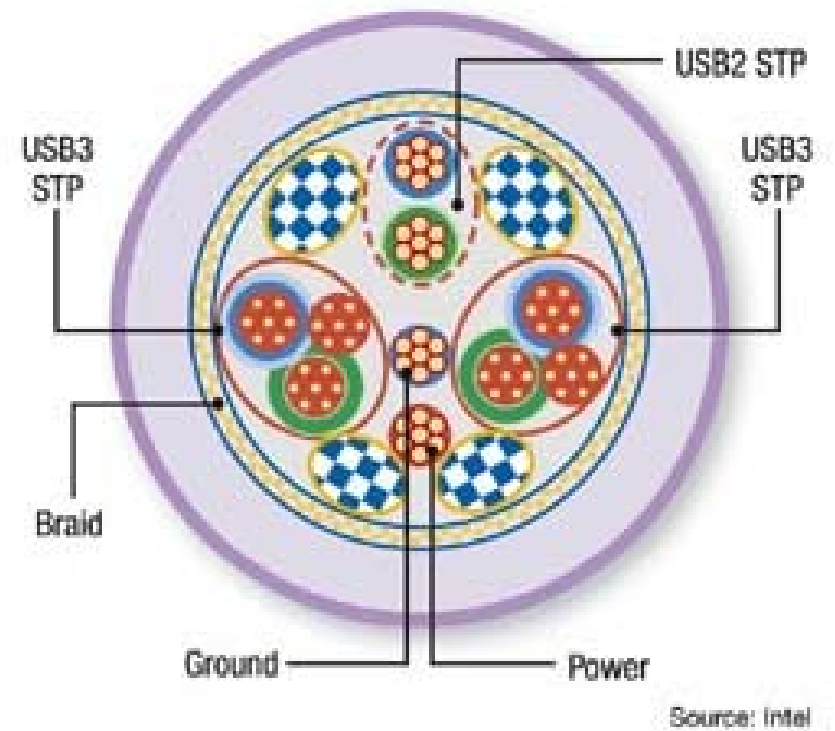
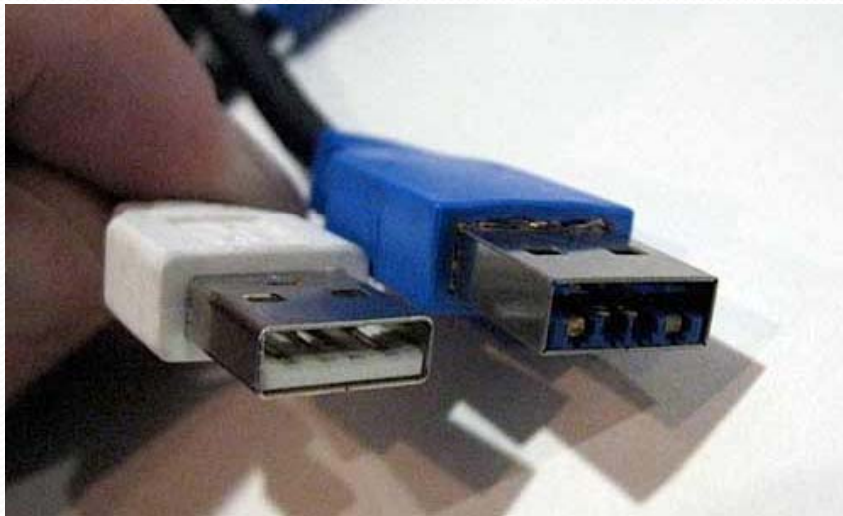
- El cable es más delgado
- Tiene 4 líneas
- Modo de transferencia de datos “half-duplex”

## USB 3.0

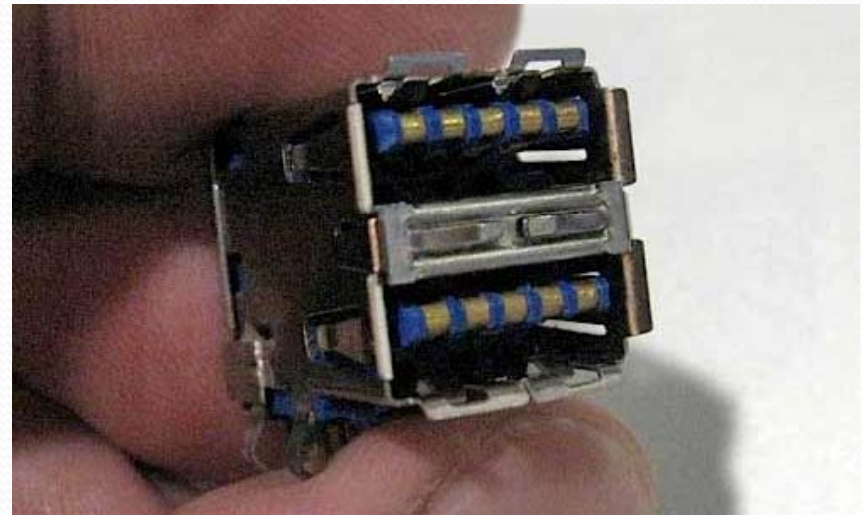
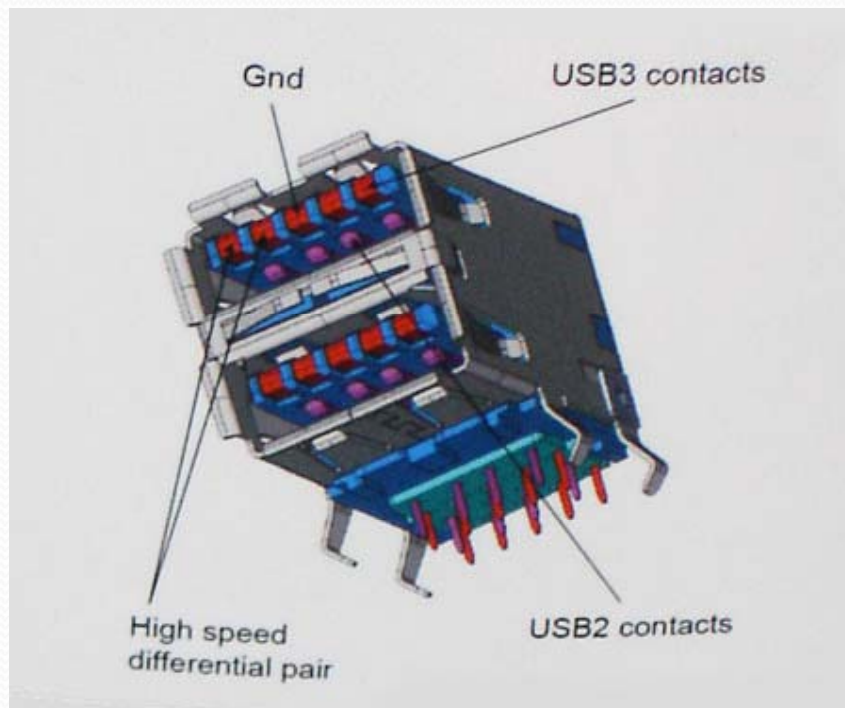
- El cable se parece al utilizado en Ethernet debido a su grosor
- Tiene 8 líneas
  - Tres pares trenzados para datos y un par para alimentación
- Modo de transferencia de datos “Full-duplex”



# USB 2.0 vs. USB 3.0



# USB 2.0 vs. USB 3.0





# USB 2.0 vs. USB 3.0

- Aunque la especificación de USB 3.0 esta diseñada para retro-compatibilidad con USB 2.0, los cables USB 3.0 no son compatibles con el conector regular B de USB 2.0







# USB 3.0 vs. otros estándares

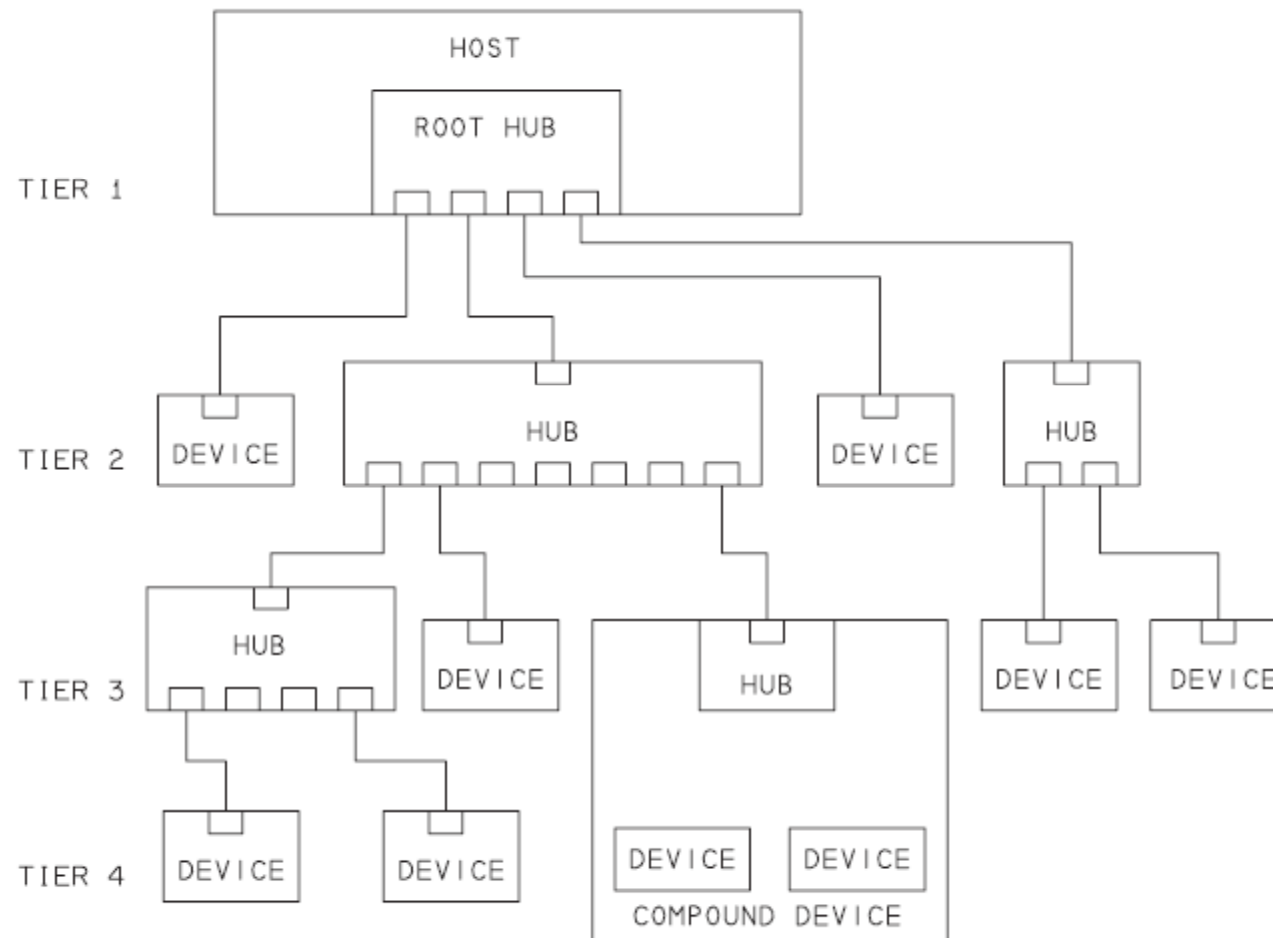
- FireWire 800 tiene como máxima tasa de transferencia: 800 Mbps
- eSATA bus tiene una máxima tasa de transferencia de 3.2 Gbps
- Ejemplo:
  - Intel mostró que la transferencia de una película de 25 GB HD demoró 70 segundos utilizando un bus USB 3.0 contra 4 horas a través de USB 2.0



# USB - Introducción

- USB significa “Universal Serial Bus”
- Controlado por “Host” (solamente uno por bus)
  - On-the-Go (Protocolo de negociación de host) – permite a dos dispositivos negociar el rol de host
- Topología estrella
  - Se pueden utilizar hubs para dividir alta y baja velocidad
- Hasta 127 dispositivos pueden ser conectados a un bus USB en cualquier momento
- Utiliza 4 líneas malladas: 2 son de alimentación (+5v & GND) y los otros 2, un par trenzado donde las señales se transmiten en modo diferencial

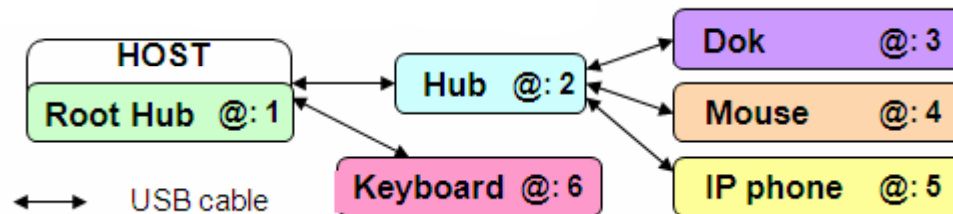
# USB - Topología física y lógica



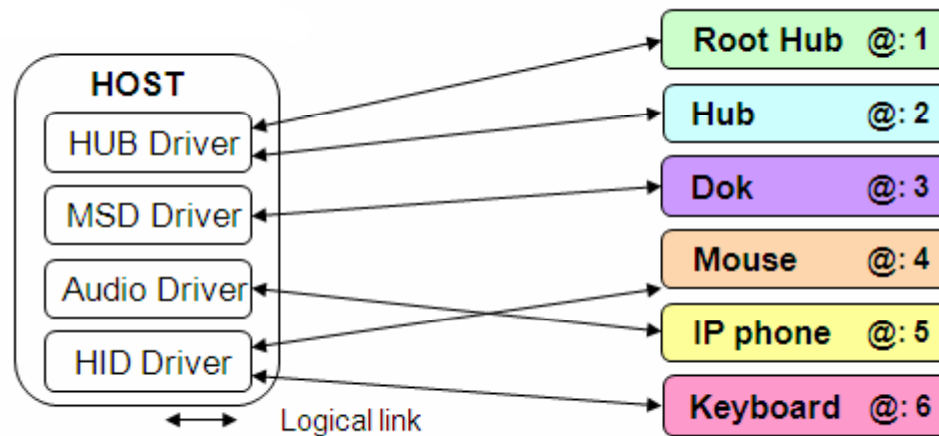


# USB - Topología física y lógica

- Topología física: estrella



- Topología lógica: punto a punto



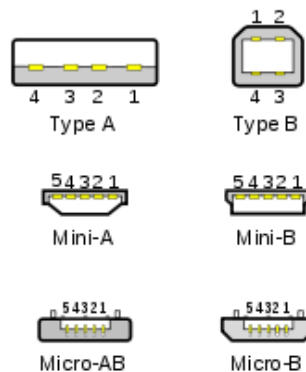


# USB - Introducción

- Soporta plug'n'plug con drivers que son cargados dinámicamente - PID/VID (Product ID/Vendor ID)
- USB soporta diferentes modos de transferencia: **Control, Interrupción (Interrupt), Masiva (Bulk) e Isócrona (Isochronous)**
- La alimentación se transporta por el Bus
  - USB distribuye la alimentación a todos los dispositivos conectados, eliminando la necesidad de una fuente externa para dispositivos de bajo consumo

# USB - Conectores

- Los conectores a cada lado del cable **no son mecánicamente intercambiables**



- El conector tipo A siempre se conecta “aguas arriba”. En general los encontramos en hosts y hubs.
- El conector tipo B siempre se conecta “aguas abajo”. Los encontramos en dispositivos.
- Los conectores micro-AB pueden ser tanto conectores micro-A como micro-B. Para USB On-the-Go.

Pin	Standard A, Standard B	Mini B, Micro B
1	VBUS	VBUS
2	D-	D-
3	D+	D+
4	GND	Open or $\geq 1M\Omega$
5	Not present	GND
Shell	Shield	Shield







# USB – Alimentación y niveles

- Alimentación

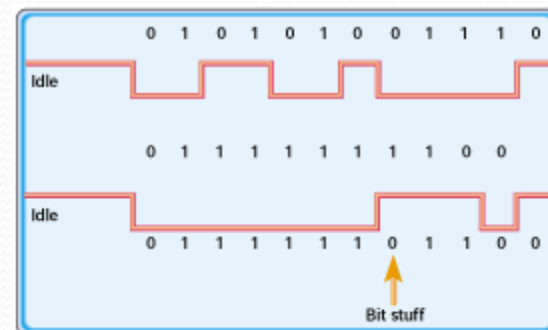
- Entrega 5 V en una de las líneas ( $5\text{ V} \pm 5\%$ ).
- La unidad de carga es 100mA (USB 2.0) y 150mA (USB 3.0).
- La máxima carga es 500 mA (USB 2.0) y 900 mA (USB 3.0).
- Los hubs alimentados por Bus solamente entregan 1 unidad de carga para los dispositivos.
- Los hubs alimentados autonomamente pueden entregar la máxima carga a todos los dispositivos.

- Niveles lógicos

- '1' - D+ 200mV mayor a D-
- '0' - D+ 200mV menor a D-

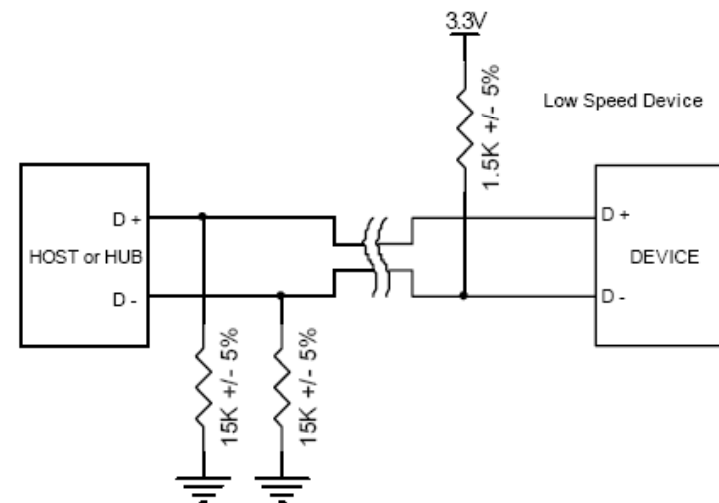
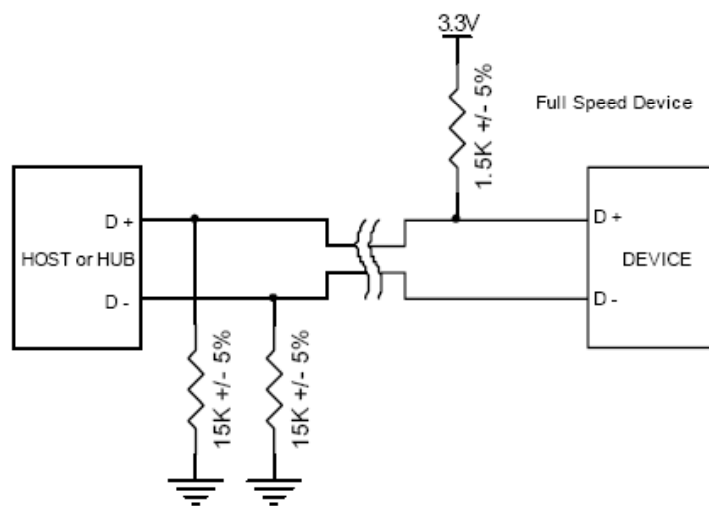
# USB - Codificación

- Utiliza codificación **NRZI** para enviar los datos con un campo de sincronización para sincronizar el clock del host y el receptor
- NRZI define un 0 lógico como una transición en el valor de tensión, y un 1 manteniendo el nivel
- Se necesita Bit stuffing porque los receptores sincronizan transiciones. Si se envían muchos 1s entonces el receptor puede perder sincronismo



# USB - Velocidad

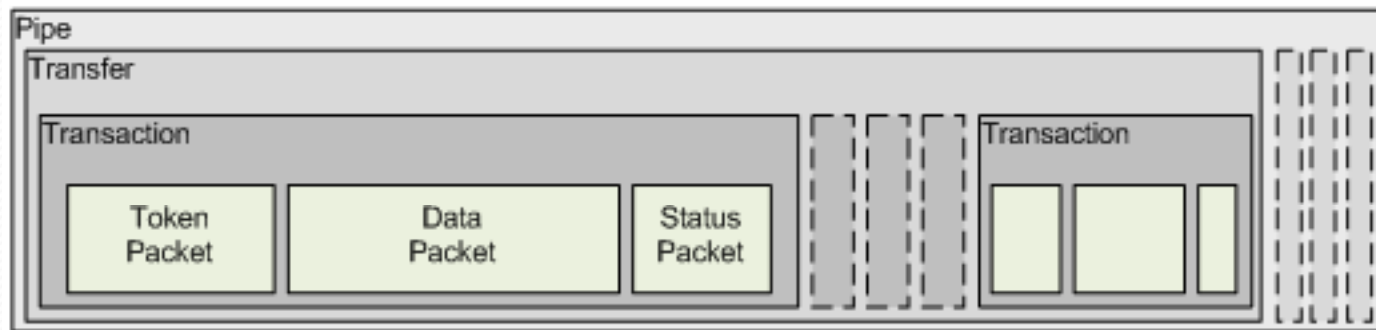
- Un dispositivo USB debe indicar su velocidad llevando D+ o D- a 3.3 volts.
- Sin resistencia de pull-up, USB asume que no hay nada conectado al Bus.
  - En el modo “high speed” el dispositivo primero se conecta en modo “full speed”, luego se remueve el resistor de pull-up para balancear la línea



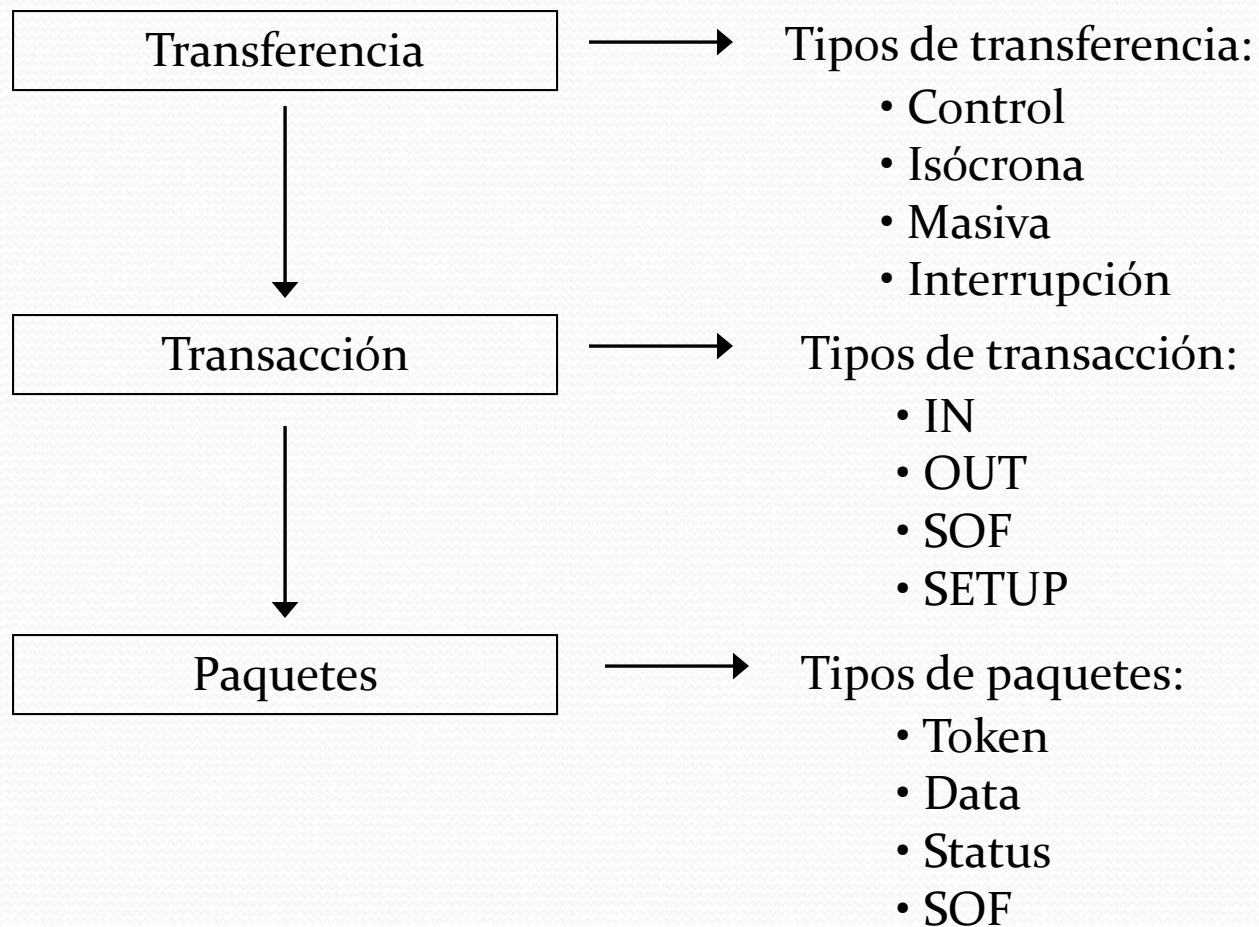


# USB – Comunicación

- A diferencia de RS-232 o interfaces serie similares donde el formato de los datos a ser enviados no está definido, USB posee **varias capas de protocolos**

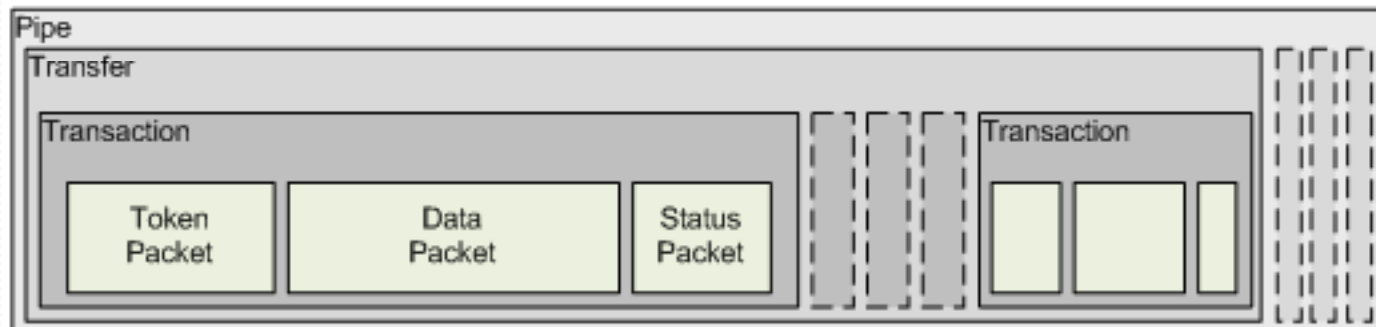


# USB – Comunicación



# USB – Paquetes

- Cada transacción USB consiste en:
  - **Paquete Token** (encabezado que define lo que se espera a continuación)
  - **Paquete de datos** (opcional - contiene el payload)
  - **Paquete de Status** (Usado como acknowledge en las transacciones y como una forma de corregir errores)







# USB - Campos del paquete

- Sync
  - Todos los paquetes deben comenzar con un campo de “sync” utilizado para sincronizar el clock receptor con el transmisor
- PID (Packet ID)
  - Utilizado para identificar el tipo de paquete que está siendo enviado (4 bits complementados)
- ADDR (Address field)
  - Especifica a que dispositivo va dirigido el paquete
  - Teniendo un tamaño de 7 bits permite soportar 127 dispositivos
  - La dirección cero no es válida ya que cualquier dispositivo al que todavía no se le ha asignado una dirección debe responder los paquetes enviados a ésta

# USB - Campos del paquete - PID

Type	PID value (msb-first)	Transmitted byte (lsb-first)	Name	Description
<i>Reserved</i>	0000	0000 1111		
Token	1000	0001 1110	<b>SPLIT</b>	High-bandwidth (USB 2.0) split transaction
	0100	0010 1101	<b>PING</b>	Check if endpoint can accept data (USB 2.0)
Special	1100	0011 1100	<b>PRE</b>	Low-bandwidth USB preamble
			<b>ERR</b>	Split transaction error (USB 2.0)
Handshake	0010	0100 1011	<b>ACK</b>	Data packet accepted
	1010	0101 1010	<b>NAK</b>	Data packet not accepted; please retransmit
	0110	0110 1001	<b>NYET</b>	Data not ready yet (USB 2.0)
	1110	0111 1000	<b>STALL</b>	Transfer impossible; do error recovery
Token	0001	1000 0111	<b>OUT</b>	Address for host-to-device transfer
	1001	1001 0110	<b>IN</b>	Address for device-to-host transfer
	0101	1010 0101	<b>SOF</b>	Start of frame marker (sent each ms)
	1101	1011 0100	<b>SETUP</b>	Address for host-to-device control transfer
Data	0011	1100 0011	<b>DATA0</b>	Even-numbered data packet
	1011	1101 0010	<b>DATA1</b>	Odd-numbered data packet
	0111	1110 0001	<b>DATA2</b>	Data packet for high-bandwidth isochronous transfer (USB 2.0)
	1111	1111 0000	<b>MDATA</b>	Data packet for high-bandwidth isochronous transfer (USB 2.0)

# USB - Campos del paquete

- ENDP (Endpoint field)
  - Formado por 4 bits permite 16 posibles “endpoints”
- CRC (Cyclic Redundancy Check)
  - Efectuado en los datos contenidos en el “payload” del paquete. Todos los paquetes “token” tienen un CRC de 5 bits mientras que los de datos tienen un CRC de 16 bits
- EOP (End of packet)
  - Señalizado a través de un “Single Ended Zero” (SEo / D+ and D- se mantienen bajos) por aproximadamente el tiempo de 2 bits seguido por una J (estado lógico, el significado depende de la velocidad) durante el tiempo de 1 bit



# USB - Tipos de paquete

- Paquetes “Token”

- **In** – Informa al dispositivo USB que el “host” desea leer información
- **Out** – Informa al dispositivo USB que el “host” desea enviar información
- **Setup** – Utilizado para comenzar transferencias de control



- Paquete de datos

- Dos tipos. Cada uno capaz de transmitir de 0 a 1023 bytes de datos

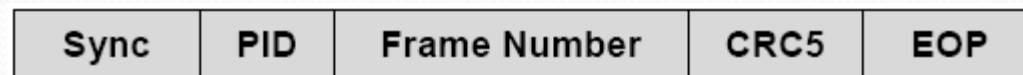


# USB – Tipos de paquete

- Paquetes de “Status” o “Handshake”
  - **ACK** (Acknowledgment) – Confirmación de que el paquete fue recibido exitosamente
  - **NAK** – Reporta que el dispositivo no puede enviar ni recibir datos temporalmente. También utilizado durante las transacciones de interrupción para informar al “host” que no hay datos para enviar
  - **STALL** – Puede significar un “control request” no soportado, una falla en el “control request” o que el endpoint falla.



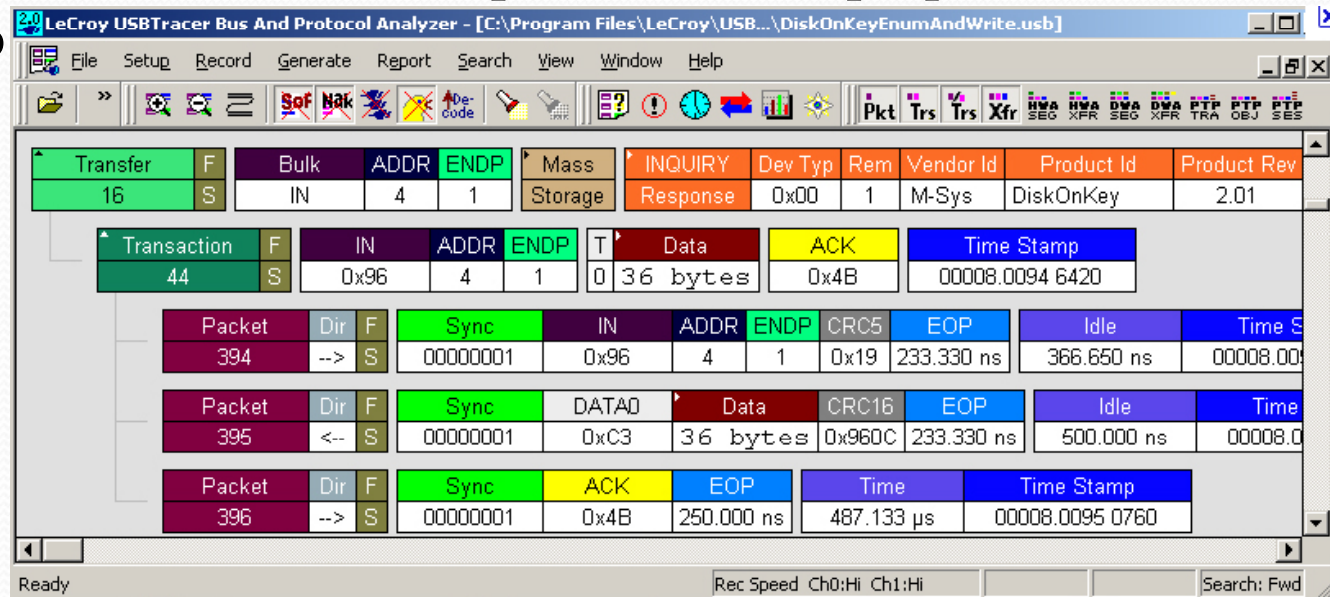
- Paquetes de comienzo de “frame” (SOF)
  - El número de frame (11 bits) es enviado por el “host” cada  $1\text{mS} \pm 500\text{nS}$





# USB – Analizador de protocolo

- Las transferencias consisten en una o más transacciones. Un pipe solamente soporta un tipo de transferencia
- En una transacción, transferencia desde “host” a dispositivo o viceversa. La dirección se define en el paquete “token”
- En general, el destinatario responde con un paquete de status indicando si fue exitosa



The screenshot displays the LeCroy USBTracer Bus And Protocol Analyzer interface. The main window shows a detailed view of a USB transaction. The top menu bar includes File, Setup, Record, Generate, Report, Search, View, Window, and Help. Below the menu is a toolbar with various icons for analysis and navigation. The main display area is divided into several sections:

Transfer	F	Bulk	ADDR	ENDP	Mass	INQUIRY	Dev Typ	Rem	Vendor Id	Product Id	Product Rev
16	S	IN	4	1	Storage	Response	0x00	1	M-Sys	DiskOnKey	2.01

Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time Stamp
44	S	0x96	4	1	0	36 bytes	0x4B	00008.0094 6420

Packet	Dir	F	Sync	IN	ADDR	ENDP	CRC5	EOP	Idle	Time S
394	-->	S	00000001	0x96	4	1	0x19	233.330 ns	366.650 ns	00008.00

Packet	Dir	F	Sync	DATA0	Data	CRC16	EOP	Idle	Time
395	<--	S	00000001	0xC3	36 bytes	0x960C	233.330 ns	500.000 ns	00008.0

Packet	Dir	F	Sync	ACK	EOP	Time	Time Stamp
396	-->	S	00000001	0x4B	250.000 ns	487.133 μs	00008.0095 0760

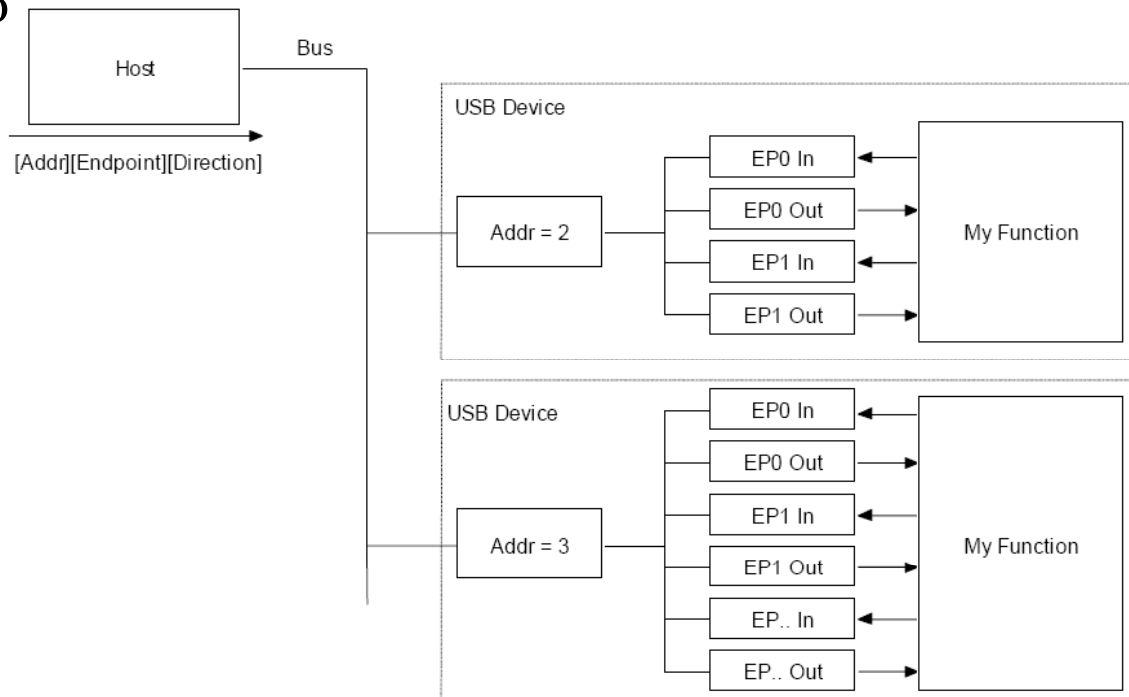
The bottom status bar shows "Ready", "Rec Speed Ch0:Hi Ch1:Hi", and "Search: Fwd".



# USB - Funciones

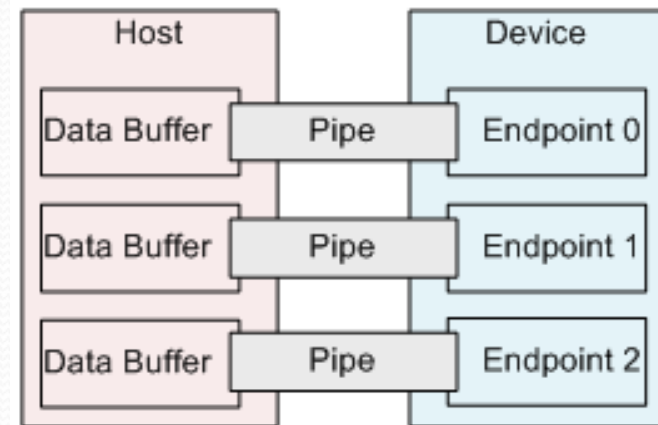
- Funciones USB

- Pueden verse como dispositivos USB que proveen capacidades o funciones tales como impresora, escáner, lector de memorias u otro periférico



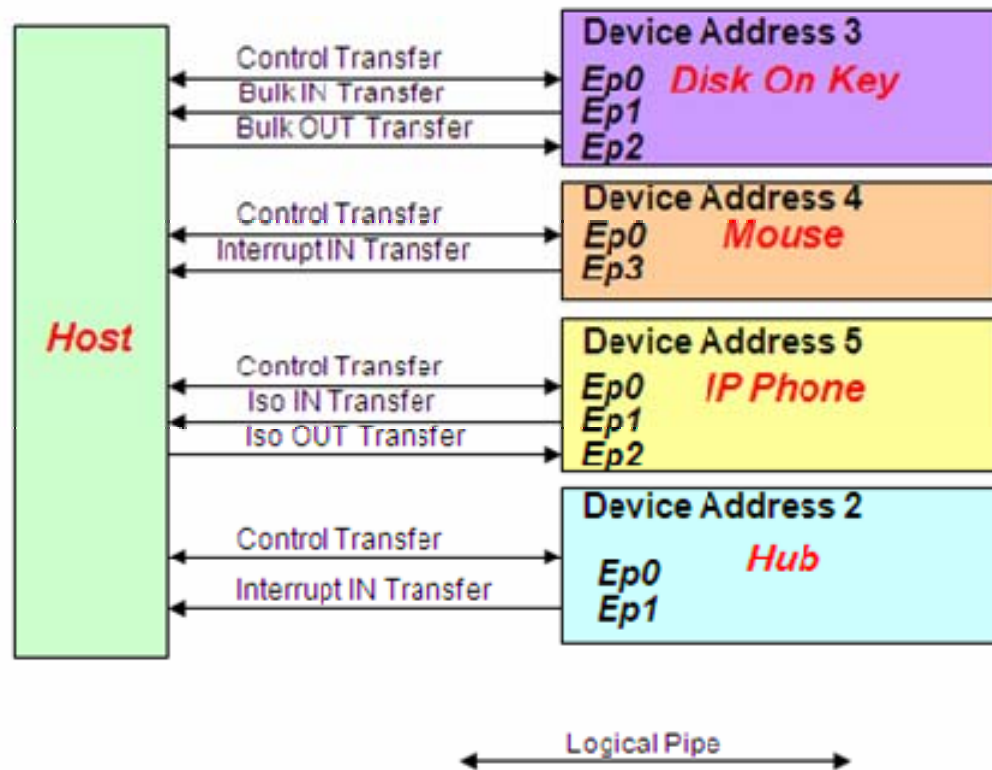
# USB - Pipes

- Son las conexiones lógicas entre “host” y “endpoint(s)”
- Tienen una serie de parámetros:
  - Ancho de banda asignado
  - Tipo de transferencia:
    - Control, Másiva (Bulk), Isócrona o Interrupción
  - Dirección del flujo de datos
  - Tamaño máximo de paquetes/buffer
  - Todos los dispositivos tienen un “default control pipe” que utiliza el endpoint cero



# USB - Endpoints

- Pueden describirse como fuentes o sumideros de datos
- Todos los dispositivos deben tener “endpoint” cero





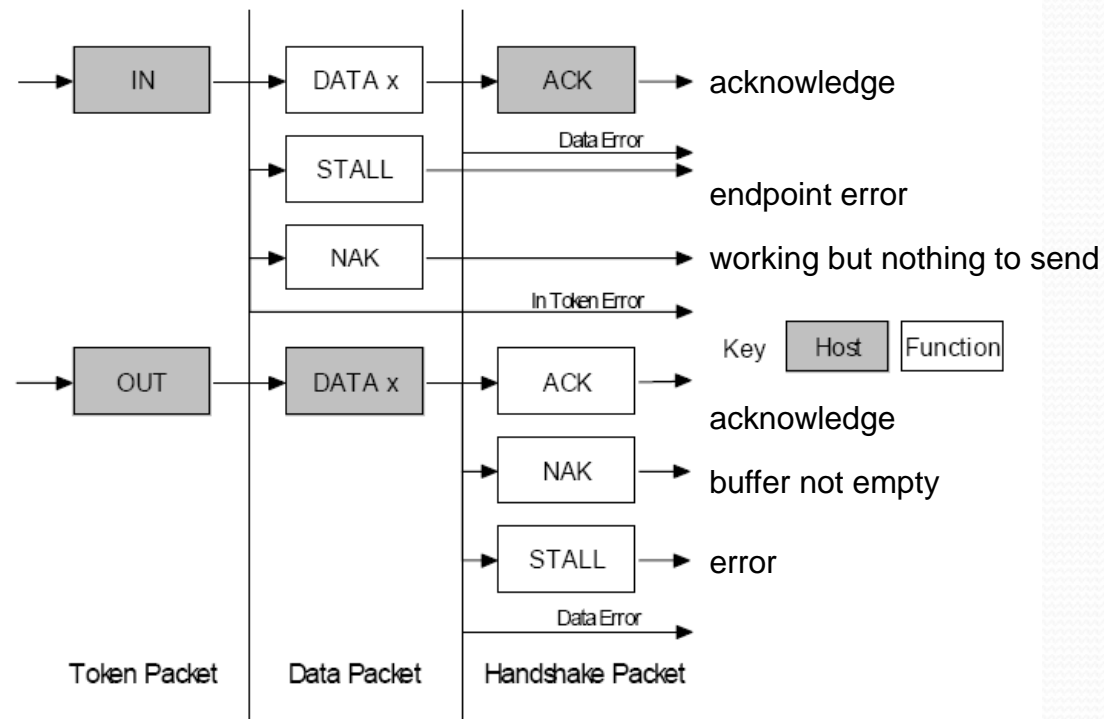


# USB – Transferencias de control

- Las transferencias de control son típicamente utilizadas para operaciones con comandos y de status
- Una transferencia de control puede tener hasta tres etapas
  - Etapa “**Setup**”: donde la petición es enviada. Contiene la dirección y el número de endpoint
  - Etapa de datos (opcional): consiste en una o multiples transferencias IN / OUT
  - Etapa de “Status”: informa el status de la totalidad de la petición. Varía en función de la dirección de la transferencia

# USB – Transferencias de control

- Formato de transferencia de control







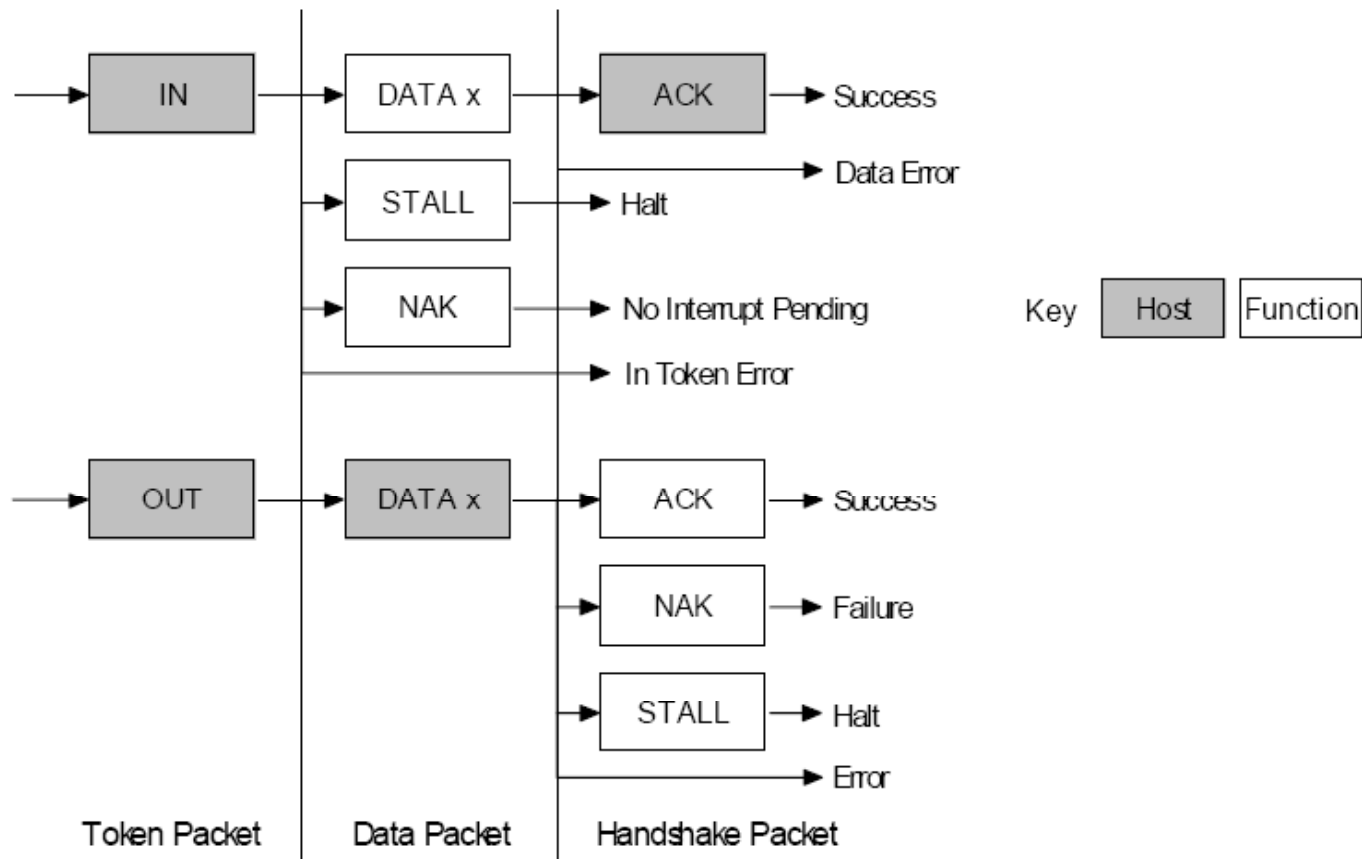
# USB – Transferencia de Interrupción

- El dispositivo que requiere atención debe esperar que el “host” le “encueste” antes que pueda informar que necesita atención
- Características
  - Latencia garantizada
  - Flujo del “pipe”: Unidireccional
  - Detecciones de errores y re-proceso en próximo período
- Interrupción IN
  - El “host” encuesta periódicamente al endpoint. La frecuencia con que encuesta está especificada en el **descriptor del endpoint**. Cada encuesta implica que el “host” envíe un IN Token
- Interrupción OUT
  - Cuando el “host” desea enviar al dispositivo datos de interrupción, solicita un OUT token seguido por un paquete de datos que contiene los datos de interrupción



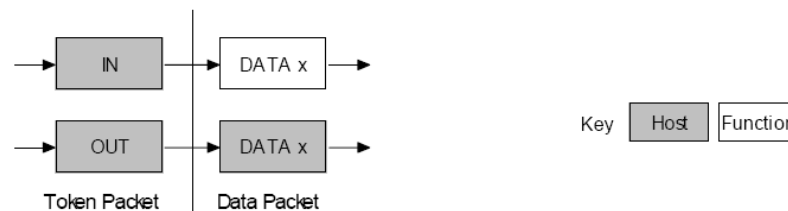
# USB – Transferencia de Interrupción

- Formato de transferencia de interrupción



# USB – Transferencia Isócrona

- Las transferencias isócronas ocurren continua y periódicamente. Típicamente contienen información sensible al tiempo, como flujo de video o audio
- Características
  - Ancho de banda USB garantizado
  - Latencia acotada
  - Flujo del “Pipe”: Unidireccional
  - Detección de errores vía CRC, pero sin re-proceso ni garantía de entrega
  - Disponible solamente en modos “full speed” y “high speed”



- Las transferencias isócronas **no tienen etapa de “handshaking”** y no pueden reportar errores o condiciones de STALL/HALT



# USB – Transferencias masivas

- Utilizado para envío masivo de datos (Ej.: datos de impresión enviados a una impresora o datos de una imagen generados por un escáner)
- Características
  - Corrección de errores (Campo CRC16 en el “data payload”)
  - Detección de errores / mecanismos de re-transmisión
- Utiliza espacio no asignado del ancho de banda del bus después que todas las otras transacciones han sido asignadas
  - Solamente utilizado en comunicaciones no sensibles al tiempo debido a que no hay garantías respecto a la latencia
- Disponible solamente en modos “full speed” y “high speed”



# USB – Resumen transferencias

Transfer Type	Control	Bulk	Interrupt	Isochronous
Typical Use	Identification and configuration	Printer, scanner, drive	Mouse, keyboard	Streaming audio, video
Support required?	yes	no	no	no
Low speed allowed?	yes	no	yes	no
Maximum packet size; maximum guaranteed packets/interval (SuperSpeed).	512; none	1024; none	1024; 3 / 125 $\mu$ s	1024; 48 / 125 $\mu$ s
Maximum packet size; maximum guaranteed packets/interval (high speed).	64; none	512; none	1024; 3 / 125 $\mu$ s	1024; 3 / 125 $\mu$ s
Maximum packet size; maximum guaranteed packets/interval (full speed).	64; none	64; none	64; 1 / ms	1023; 1 / ms
Maximum packet size; maximum guaranteed packets/interval (low speed).	8; none	not allowed	8; 1 / 10 ms	not allowed
Direction of data flow	IN and OUT	IN or OUT	IN or OUT (IN only for USB 1.0)	IN or OUT
Reserved bandwidth for all transfers of the type	10% at low/full speed, 20% at high speed & SuperSpeed	none	90% at low/full speed, 80% at high speed and SuperSpeed (isochronous and interrupt combined, maximum)	
Message or Stream data?	message	stream	stream	stream
Error correction?	yes	yes	yes	no
Guaranteed delivery rate?	no	no	no	yes
Guaranteed latency (maximum time between transfers attempts)?	no	no	yes	yes

# USB - Frames

- El tráfico en el bus USB es regulado utilizando el tiempo. La unidad de tiempo se llama “frame”
  - Velocidad “Full” y “Low”: “frames” cada 1 ms
  - Velocidad “High”: “micro-frames” cada 125  $\mu$ s
- Cada “frame” comienza con un paquete SOF
- A cada “pipe” se le asigna un espacio en cada “frame”
- 10 % asignado a transferencias de control





# USB - “Throughput” teórico

- A medida que los dispositivos son enumerados el host va contabilizando el ancho de banda solicitado por los endpoints isócronos y de interrupción
- Pueden consumir hasta un 90 % del disponible, luego el host niega el acceso

Tipo de transferencia	Low-Speed (1.5 Mbps / 187 kBps)	Full-Speed (12 Mbps / 1,5 MBps)	High-Speed (480 Mbps / 60 MBps)
Control	24 kBps	832 kBps	15.872 kBps
Interrupción	0,8 kBps	64 kBps	24.576 kBps
Masiva	-	1.216 kBps	53.248 kBps
Isócrona	-	1.023 kBps	24.576 kBps

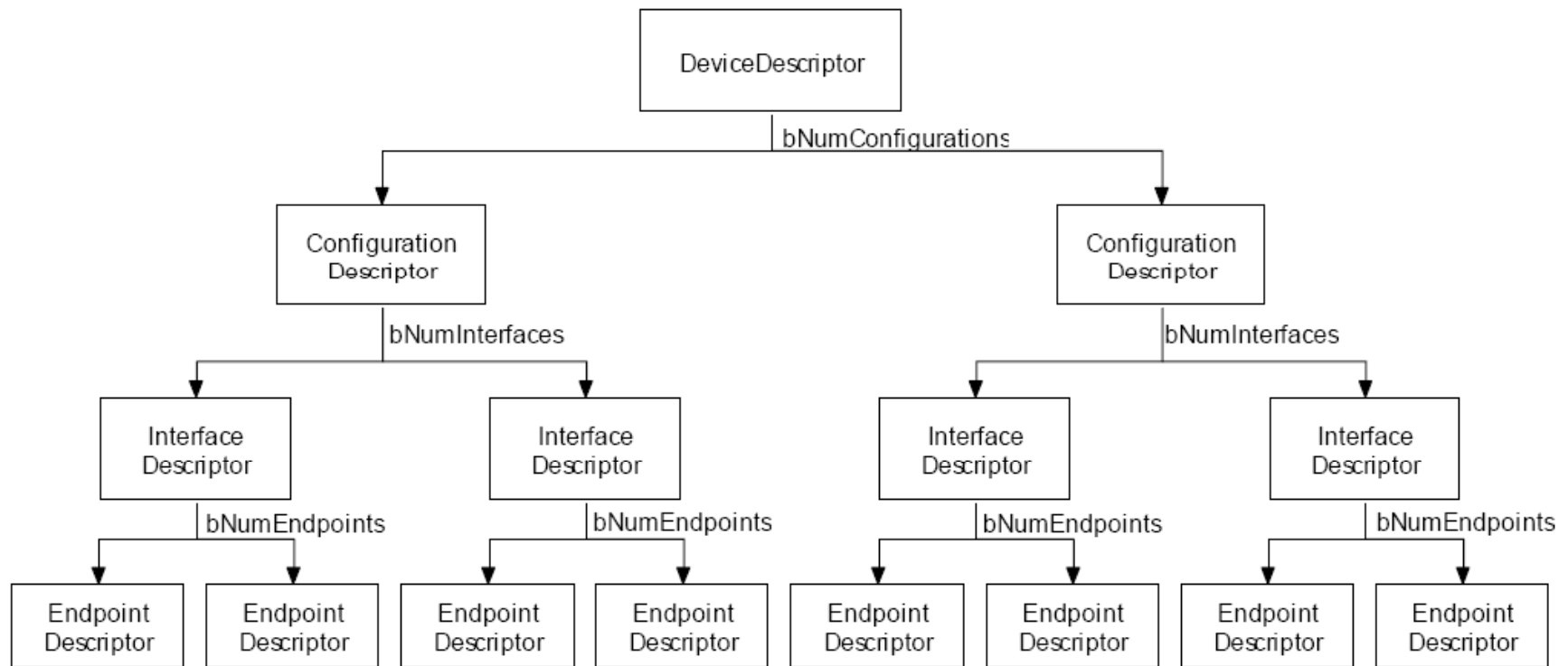




# USB – Descriptores

- Todos los dispositivos USB tienen una jerarquía de descriptores que definen al “host” información tal como:
  - que es el dispositivo
  - quien lo fabricó
  - que versión de USB soporta
  - de cuantas formas puede configurarse
  - el número de endpoints y sus tipos
- Los descriptores USB más comunes son
  - Descriptores de dispositivo (Device descriptors)
  - Descriptores de configuración (Configuration Descriptors)
  - Descriptores de interfaz (Interface Descriptors)
  - Descriptores de Endpoint (Endpoint Descriptors)
  - Descriptores de String (String Descriptors)
    - Proporciona información humanamente legible y son opcionales

# USB – Descriptors





# USB – Descriptores de dispositivo

- El descriptor de dispositivo del dispositivo USB representan a la totalidad del mismo por lo tanto, **sólo puede tener uno**
- Contienen
  - la versión de USB soportada
  - el máximo tamaño de paquete para el endpoint o
  - identificación de proveedor y producto
  - el número de posibles configuraciones que el dispositivo puede tener
- Ejemplo:
  - bDeviceClass, bDeviceSubClass y bDeviceProtocol son utilizados por el sistema operativo para encontrar un driver para el dispositivo
  - Generalmente solo bDeviceClass es especificado en este nivel
  - Se suelen especificar los demás parámetros a nivel de interfaz. Esto permite que un mismo dispositivo soporte múltiples clases



# USB – Descriptores de dispositivo

Offset (decimal)	Field	Size (bytes)	Description
0	bLength	1	Descriptor size in bytes (12h)
1	bDescriptorType	1	The constant DEVICE (01h)
2	bcdUSB	2	USB specification release number (BCD)
4	bDeviceClass	1	Class code
5	bDeviceSubclass	1	Subclass code
6	bDeviceProtocol	1	Protocol Code
7	bMaxPacketSize0	1	Maximum packet size for endpoint zero
8	idVendor	2	Vendor ID
10	idProduct	2	Product ID
12	bcdDevice	2	Device release number (BCD)
14	iManufacturer	1	Index of string descriptor for the manufacturer
15	iProduct	1	Index of string descriptor for the product
16	iSerialNumber	1	Index of string descriptor for the serial number
17	bNumConfigurations	1	Number of possible configurations



# USB – Descriptores de configuración

- Un dispositivo USB puede tener diferentes configuraciones. De todas formas, la mayoría de los dispositivos son simples y solamente tienen una
- Especifica
  - como se alimenta el dispositivo
  - cual es el máximo consumo de potencia
  - el número de interfaces que tiene
- Por lo tanto, es posible tener dos configuraciones: una para el dispositivo siendo alimentado por el bus y otra cuando lo hace externamente. Como este es el “encabezado” de los descriptores de interfaz, es también posible tener para cada una de las configuraciones, diferentes **modos de transferencia**

# USB – Descriptores de configuración

Offset (decimal)	Field	Size (bytes)	Description
0	bLength	1	Descriptor size in bytes (09h)
1	bDescriptorType	1	The constant CONFIGURATION (02h)
2	wTotalLength	2	The number of bytes in the configuration descriptor and all of its subordinate descriptors
4	bNumInterfaces	1	Number of interfaces in the configuration
5	bConfigurationValue	1	Identifier for Set Configuration and Get Configuration requests
6	iConfiguration	1	Index of string descriptor for the configuration
7	bmAttributes	1	Self/bus power and remote wakeup settings
8	bMaxPower	1	Bus power required in units of 2 mA (USB 2.0) or 8 mA (SuperSpeed).



# USB – Descriptores de interfaz

- Pueden ser vistos como “headers” de los endpoints en grupos funcionales que desarrollan una misma función en el dispositivo

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of Descriptor in Bytes
1	bDescriptorType	1	Constant	Interface Descriptor (0x04)
2	bInterfaceNumber	1	Number	Number of Interface
3	bAlternateSetting	1	Number	Value used to select alternative setting
4	bNumEndpoints	1	Number	Number of Endpoints used for this interface
5	bInterfaceClass	1	Class	Class Code (Assigned by USB Org)
6	bInterfaceSubClass	1	SubClass	Subclass Code (Assigned by USB Org)
7	bInterfaceProtocol	1	Protocol	Protocol Code
8	iInterface	1	Index	Index of String Descriptor Describing this interface

# USB – Descriptor de endpoint

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of Descriptor in Bytes (7 bytes)
1	bDescriptorType	1	Constant	Endpoint Descriptor (0x05)
2	bEndpointAddress	1	Endpoint	Endpoint Address, Encoded as follows 0..3b Endpoint Number 4..6b Reserved. Set to Zero 7b Direction (Ignored for Control Endpoints) 0 = Out Endpoint, 1 = In Endpoint
3	bmAttributes	1	Bitmap	<div style="border: 2px solid red; padding: 5px;">           Bits 0..1 Transfer Type            00 = Control            01 = Isochronous            10 = Bulk            11 = Interrupt         </div> Bits 2..7 are reserved. If Isochronous endpoint, Bits 3..2 = Synchronisation Type (Iso Mode)  00 = No Synchronisation 01 = Asynchronous 10 = Adaptive 11 = Synchronous  Bits 5..4 = Usage Type (Iso Mode)  00 = Data Endpoint 01 = Feedback Endpoint 10 = Explicit Feedback Data Endpoint 11 = Reserved
4	wMaxPacketSize	2	Number	Maximum Packet Size this endpoint is capable of sending or receiving
6	bInterval	1	Number	Interval for polling endpoint data transfers. Value in frame counts. Ignored for Bulk & Control Endpoints. Iso must equal 1 and field may range from 1 to 255 for interrupt endpoints.

- El endpoint cero siempre se asume como de control
- El “host” utilizará la información devuelta por estos descriptores para definir los requerimientos de ancho de banda del bus



# USB – Paquetes de Setup

- Todos los dispositivos USB tienen que responder a paquetes en el “default pipe” (Transferencias de control).
- Los paquetes de setup se utilizan para la detección y configuración del dispositivo. También para llevar a cabo algunas funciones comunes como asignar una dirección al dispositivo, solicitar el descriptor del dispositivo o chequear el status de algún endpoint.
- El host espera que todos los pedidos sean procesados en un período máximo de 5 segundos. También especifica períodos más estrictos para algunos pedidos específicos. Esto puede traer problemas para debuggear el código.



# USB – Paquetes de Setup

Offset	Field	Size	Value	Description
0	bmRequestType	1	Bit-Map	<b>D7 Data Phase Transfer Direction</b> 0 = Host to Device 1 = Device to Host <b>D6..5 Type</b> 0 = Standard 1 = Class 2 = Vendor 3 = Reserved <b>D4..0 Recipient</b> 0 = Device 1 = Interface 2 = Endpoint 3 = Other 4..31 = Reserved
1	bRequest	1	Value	Request
2	wValue	2	Value	Value
4	wIndex	2	Index or Offset	Index
6	wLength	2	Count	Number of bytes to transfer if there is a data phase

- bRequest define el tipo de pedido
- La especificación USB define algunos pedidos standard (Standard Request)
- También existen los llamados “Class Request” definidos por los “Class drivers”
- En función del tipo de dispositivo tendrá un set de pedidos específicos

# USB – Paquetes de Setup

bmRequestType	bRequest	wValue	wIndex	wLength	Data
00000000B 00000001B 00000010B	CLEAR_FEATURE	Feature Selector	Zero Interface Endpoint	Zero	None
10000000B	GET_CONFIGURATION	Zero	Zero	One	Configuration Value
10000000B	GET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
10000001B	GET_INTERFACE	Zero	Interface	One	Alternate Interface
10000000B 10000001B 10000010B	GET_STATUS	Zero	Zero Interface Endpoint	Two	Device, Interface, or Endpoint Status
00000000B	SET_ADDRESS	Device Address	Zero	Zero	None
00000000B	SET_CONFIGURATION	Configuration Value	Zero	Zero	None
00000000B	SET_DESCRIPTOR	Descriptor Type and Descriptor Index	Zero or Language ID	Descriptor Length	Descriptor
00000000B 00000001B 00000010B	SET_FEATURE	Feature Selector	Zero Interface Endpoint	Zero	None
00000001B	SET_INTERFACE	Alternate Setting	Interface	Zero	None
10000010B	SYNCH_FRAME	Zero	Endpoint	Two	Frame Number

- Un pedido de GET\_STATUS puede dirigirse tanto a dispositivo, interfaz o endpoint.
- En el caso de ser enviada a dispositivo responde sobre “remote wake up” y “self powered”
- En el caso de endpoint responde si esta “halted” o “stalled”























# USB – Enumeración

1. El sistema tiene un nuevo dispositivo.
2. El hub detecta un dispositivo.
3. El host es notificado del nuevo dispositivo.
4. El hub detecta si el dispositivo es low o full speed.
5. El hub resetea el dispositivo.
6. El host es notificado si el dispositivo soporta high speed.
7. El hub establece un camino de señal entre dispositivo y bus.
8. El host envía un pedido de Get Descriptor para conocer el máximo tamaño de paquete del default pipe.
9. El host asigna una dirección al dispositivo.
10. El host aprende las habilidades del dispositivo.
11. El host asigna y carga los drivers del dispositivo.
12. El driver del host selecciona una configuración.

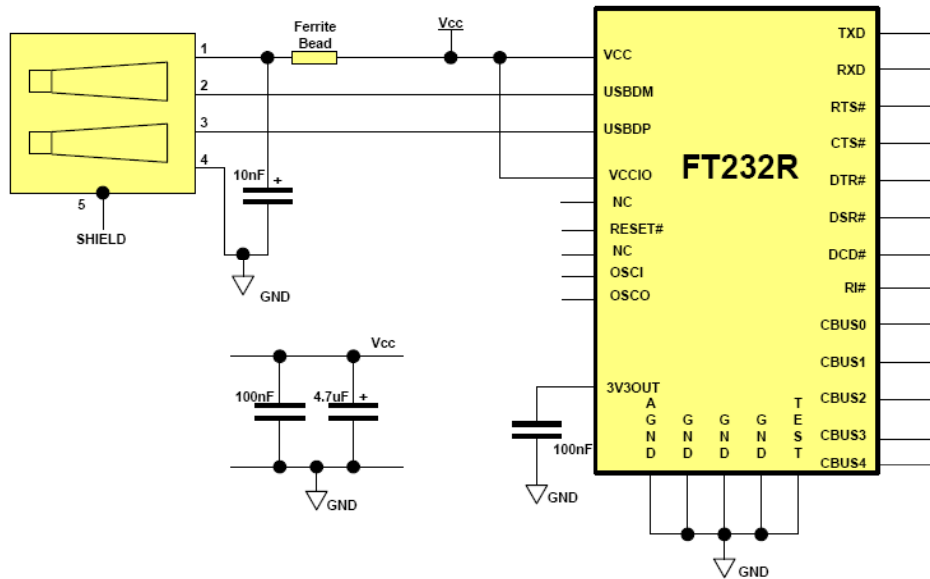


# USB – Enumeración

Item	Device	Payload
 Reset (2.3 s)		
 Suspended (114.0 ms)		
 Reset (10.0 ms)		
 High speed Detection Handshake		
 GetDescriptor (Device)	0 (5)	8 bytes (12 01 00 02 FF 00 00 08)
 Reset (10.0 ms)		
 High speed Detection Handshake		
 SetAddress (5)	0 (5)	No data
 GetDescriptor (Device)	5	18 bytes (12 01 00 02 FF 00 00 08 ...)
 GetDescriptor (Configuration)	5	9 bytes (09 02 2E 00 01 01 00 A0 32)
 GetDescriptor (Configuration)	5	46 bytes (09 02 2E 00 01 01 00 A0 ...)
 GetDescriptor (String lang IDs)	5	4 bytes (04 03 09 04)
 GetDescriptor (String iProduct)	5	24 bytes (18 03 57 00 69 00 6E 00 ...)
 GetDescriptor (String lang IDs)	5	4 bytes (04 03 09 04)
 GetDescriptor (String iProduct)	5	24 bytes (18 03 57 00 69 00 6E 00 ...)
 GetDescriptor (Device)	5	18 bytes (12 01 00 02 FF 00 00 08 ...)
 GetDescriptor (Configuration)	5	9 bytes (09 02 2E 00 01 01 00 A0 32)
 GetDescriptor (Configuration)	5	46 bytes (09 02 2E 00 01 01 00 A0 ...)
 GetStatus (Device)	5	2 bytes (00 00)
 SetConfiguration (1)	5	No data

# USB – FT232

- Precio: 4,50 U\$S (Mouser)



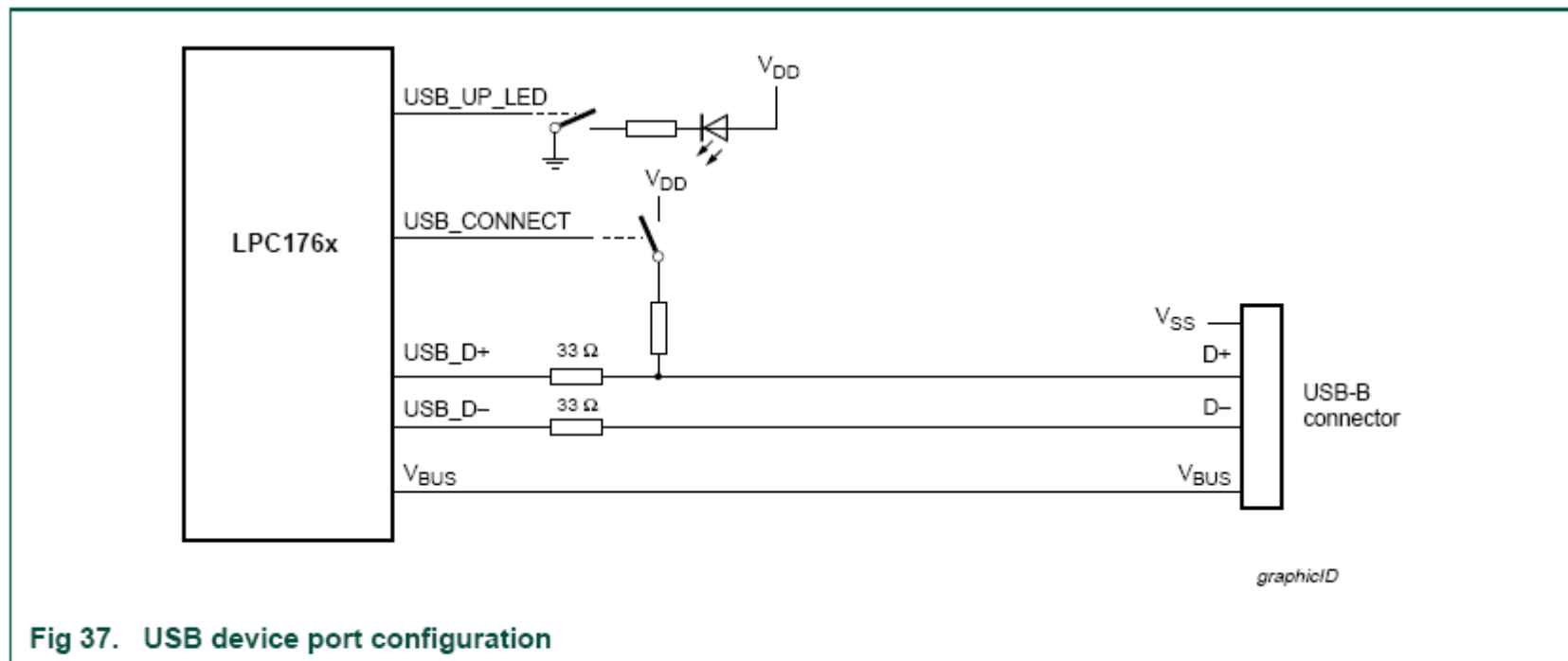
CBUS Signal Option	Available On CBUS Pin	Description
TXDEN	CBUS0, CBUS1, CBUS2, CBUS3, CBUS4	Enable transmit data for RS485
PWREN#	CBUS0, CBUS1, CBUS2, CBUS3, CBUS4	Output is low after the device has been configured by USB, then high during USB suspend mode. This output can be used to control power to external logic P-Channel logic level MOSFET switch. Enable the interface pull-down option when using the PWREN# in this way.*
TXLED#	CBUS0, CBUS1, CBUS2, CBUS3, CBUS4	Transmit data LED drive – pulses low when transmitting data via USB. See Section 7.5 for more details.
RXLED#	CBUS0, CBUS1, CBUS2, CBUS3, CBUS4	Receive data LED drive – pulses low when receiving data via USB. See Section 7.5 for more details.
TX&RXLED#	CBUS0, CBUS1, CBUS2, CBUS3, CBUS4	LED drive – pulses low when transmitting or receiving data via USB. See Section 7.5 for more details.
SLEEP#	CBUS0, CBUS1, CBUS2, CBUS3, CBUS4	Goes low during USB suspend mode. Typically used to power down an external TTL to RS232 level converter IC in USB to RS232 converter designs.
CLK48	CBUS0, CBUS1, CBUS2, CBUS3, CBUS4	48MHz Clock output.**
CLK24	CBUS0, CBUS1, CBUS2, CBUS3, CBUS4	24 MHz Clock output.**
CLK12	CBUS0, CBUS1, CBUS2, CBUS3, CBUS4	12 MHz Clock output.**
CLK6	CBUS0, CBUS1, CBUS2, CBUS3, CBUS4	6 MHz Clock output.**
CBitBangI/O	CBUS0, CBUS1, CBUS2, CBUS3	CBUS bit bang mode option. Allows up to 4 of the CBUS pins to be used as general purpose I/O. Configured individually for CBUS0, CBUS1, CBUS2 and CBUS3 in the internal EEPROM. A separate application note, AN232R-01, available from FTDI website ( <a href="http://www.ftdichip.com">www.ftdichip.com</a> ) describes in more detail how to use CBUS bit bang mode.
BitBangWRn	CBUS0, CBUS1, CBUS2, CBUS3	Synchronous and asynchronous bit bang mode WR# strobe output.
BitBangRDn	CBUS0, CBUS1, CBUS2, CBUS3	Synchronous and asynchronous bit bang mode RD# strobe output.



# Práctica USB



# USB en LPC17xx

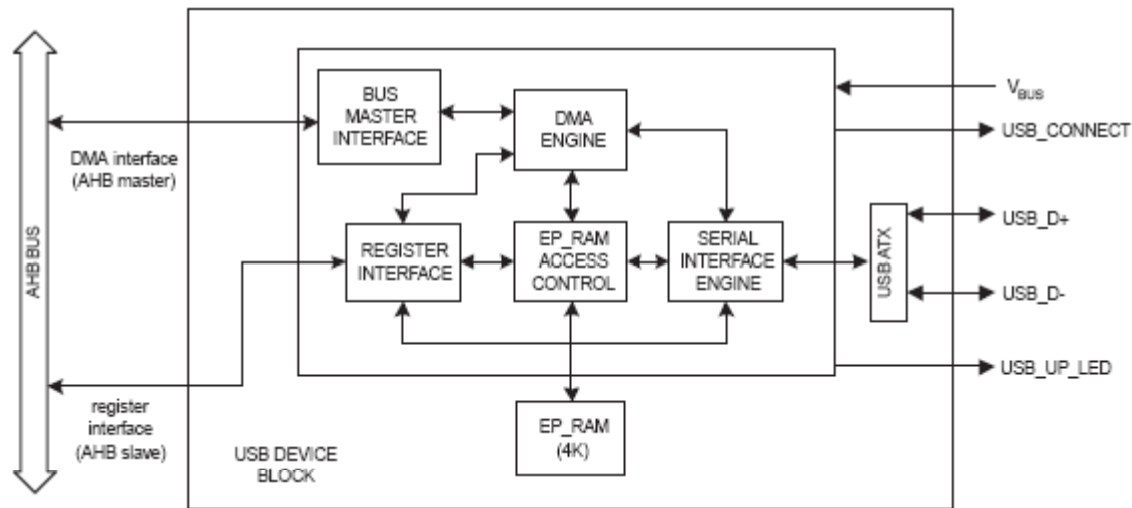


# USB en LPC17xx

Table 185. Fixed endpoint configuration

Logical endpoint	Physical endpoint	Endpoint type	Direction	Packet size (bytes)	Double buffer
0	0	Control	Out	8, 16, 32, 64	No
0	1	Control	In	8, 16, 32, 64	No
1	2	Interrupt	Out	1 to 64	No
1	3	Interrupt	In	1 to 64	No
2	4	Bulk	Out	8, 16, 32, 64	Yes
2	5	Bulk	In	8, 16, 32, 64	Yes
3	6	Isochronous	Out	1 to 1023	Yes
3	7	Isochronous	In	1 to 1023	Yes
4	8	Interrupt	Out	1 to 64	No
4	9	Interrupt	In	1 to 64	No
5	10	Bulk	Out	8, 16, 32, 64	Yes
5	11	Bulk	In	8, 16, 32, 64	Yes
6	12	Isochronous	Out	1 to 1023	Yes
6	13	Isochronous	In	1 to 1023	Yes
7	14	Interrupt	Out	1 to 64	No
7	15	Interrupt	In	1 to 64	No
8	16	Bulk	Out	8, 16, 32, 64	Yes
8	17	Bulk	In	8, 16, 32, 64	Yes
9	18	Isochronous	Out	1 to 1023	Yes
9	19	Isochronous	In	1 to 1023	Yes
10	20	Interrupt	Out	1 to 64	No
10	21	Interrupt	In	1 to 64	No
11	22	Bulk	Out	8, 16, 32, 64	Yes
11	23	Bulk	In	8, 16, 32, 64	Yes
12	24	Isochronous	Out	1 to 1023	Yes
12	25	Isochronous	In	1 to 1023	Yes
13	26	Interrupt	Out	1 to 64	No
13	27	Interrupt	In	1 to 64	No
14	28	Bulk	Out	8, 16, 32, 64	Yes
14	29	Bulk	In	8, 16, 32, 64	Yes
15	30	Bulk	Out	8, 16, 32, 64	Yes
15	31	Bulk	In	8, 16, 32, 64	Yes

# USB en LPC17xx



The SIE implementa el protocolo USB y realiza por hardware la transferencia de datos entre endpoint buffers y el bus USB.:

- levanta el patrón de sincronización
- bit stuffing/de-stuffing
- chequeo y generación de CRC
- verificación y generación de PID
- evaluación y generación de handshake



# USB – Ejemplo HID

- Descargar el workspace de:  
<http://www.lpcware.com/content/nxpfile/nxpusbplib-v098>
- Descomprimir
- Seguir los pasos en \libraries\LPCUSBLib\Readme.txt utilizando como “example”: Example\_GenericHIDDevice

1. Start up the IDE
2. Choose a workspace folder
3. Click File->Import->General->Existing Projects into Workspace->Next
4. Browse to the base of your LPCUSBLib release (the location of this file)
5. Click Finish

At this point the Project Explorer should show the following projects:

BSP	(Board Support Package)
CDL	(Common Driver Library)
Example_<app name>	(Example applications)
LPCUSBLib	(NXP's USB library)
Project_<proj name>	(Miscellaneous projects)

6. Right click on BSP->Build Configurations->Set Active and select your development board
7. Right click on CDL->Build Configurations->Set Active and select your MCU
8. Right click on LPCUSBLib->Build Configurations->Set Active and select your MCU
9. Right click on Example\_<app name>->Build Configurations->Set Active and select your MCU
10. Right click on Example\_<app name>->Properties->C/C++ Build->MCU Settings and select your MCU
11. Right click on Example\_<app name>->Build Project

# USB – Ejemplo HID

- Seguir los pasos en el readme del workspace “Example\_GenericHIDDevice”

LPCXpresso IDE

-----  
Configure projects:

Right click on the BSP project in the project explorer window click Build Configurations->Set Active->(see configuration in table below) repeat these steps for all the below projects as well.

CDL  
LPCUSBlib  
Example\_GenericHIDDevice

Configure MCU:

Right click on the Example\_GenericHIDDevice project in the project explorer window then click Properties->C/C++ Build->MCU settings->(see configuration in table below).

Configure the indexer:

Click window->Preferences->C/C++->Indexer->Use active build configuration

# USB – Ejemplo HID

- En el mismo readme, como utilizar el ejemplo:

```
=====
==
== How this example runs and what to look for ==
==
=====
```

when the example is run and the board is connected to a PC with a USB cable it will enumerate on the PC as a generic HID device. You should see the device appear in the "Device Manager" under the "Human Interface Device". If you right click on USB Input Device->Properties a window will appear called "USB Input Device Properties". If you select the "Details" tab you will see a "Properties" drop down, click on it and select "Hardware Ids". You should see the below entries for Vendor Id and Product Id.

```
USB\VID_1FC9&PID_0007&REV_0100
USB\VID_1FC9&PID_0007
```

This information should match the data in the "Device Descriptor Structure" in the "Descriptors.c" file in the Example\_GenericHIDDevice project directory.

Run the included PC application HIDClient.exe to send reports between the host and device.

Use the Device: drop down list box to select "LPCUSBlib Generic HID Demo"

Press buttons on the board and see check boxes become checked in the first row  
click the check boxes in the second row to light up LEDs on the board



# USB – Ejemplo HID

- Otra alternativa más “configurable” del lado Host se puede encontrar en: <http://ahidlib.com/index.php/ahid-and-matlab>
- Donde hay que descargar *Matlab Demo (C files included)* : <http://ahidlib.com/index.php/free-download>
- En la página algunas indicaciones. Tener en cuenta que para instalarlo en Matlab 64 bits, es necesario instalar Microsoft SDK.

## 2. Running the Demo

Unzip AHidDemoMatlab.zip to your Hard Disk and choose the adequate project (32 or 64 bit). Make the relevant folder to your current folder in Matlab.

You must compile the C files to MEX-Files before you can use AHid.dll in Matlab. For 32 bit, select the LCC compiler included in Matlab (type in: `mex -setup`). The Microsoft SDK is required to compile the C files for Matlab 64 bit.

To create the MEX-Files, type in:

```
>> mex -setup (select LCC compiler)
>> mex ahid_init.c (repeat it for all C files)
```

Finally, edit AHidDemo.m to adjust your HID Device parameters and

```
>> run AHidDemo.m
```

# USB – Ejemplo HID

- Otra alternativa más “configurable” del lado Host se puede encontrar en: <http://ahidlib.com/index.php/ahid-and-matlab>
- Donde hay que descargar *Matlab Demo (C files included)* : <http://ahidlib.com/index.php/free-download>
- En la página algunas indicaciones. Tener en cuenta que para instalarlo en Matlab 64 bits, es necesario instalar Microsoft SDK.

## 2. Running the Demo

Unzip AHidDemoMatlab.zip to your Hard Disk and choose the adequate project (32 or 64 bit). Make the relevant folder to your current folder in Matlab.

You must compile the C files to MEX-Files before you can use AHid.dll in Matlab. For 32 bit, select the LCC compiler included in Matlab (type in: `mex -setup`). The Microsoft SDK is required to compile the C files for Matlab 64 bit.

To create the MEX-Files, type in:

```
>> mex -setup (select LCC compiler)
>> mex ahid_init.c (repeat it for all C files)
```

Finally, edit AHidDemo.m to adjust your HID Device parameters and

```
>> run AHidDemo.m
```

# USB – Ejemplo HID

- Antes de correr el ejecutable hay que modificar el Vendor ID y el Product ID en el .m:

```
27 %Register Output Report (to transfer data from Host to device).
28 - out.vid = uint32(hex2dec('1fc9'));
29 - out.pid = uint32(hex2dec('204F'));
30 - out.rep_id = uint8(0);
31 - out.rep_size = uint8(1);
32 - out.rep_type = uint8(1);
33 - out.pipe = int32(0);
34 - res_reg1 = int32(0);
```

```
50 %Register Input Report (to transfer data from device to Host).
51 - in.vid = uint32(hex2dec('1fc9'));
52 - in.pid = uint32(hex2dec('204F'));
53 - in.rep_id = uint8(0);
54 - in.rep_size = uint8(1);
55 - in.rep_type = uint8(1);
56 - in.pipe = int32(0);
57 - res_reg2 = int32(0);
```



# USB – Ejemplo HID

- Por último para actuar sobre los LEDS hay que modificar “bufw”.

```
194  
195     if (Data[0]&0x01) NewLEDMask |= LEDS_LED1;  
196     if (Data[0]&0x02) NewLEDMask |= LEDS_LED2;  
197     if (Data[0]&0x04) NewLEDMask |= LEDS_LED3;  
198     if (Data[0]&0x08) NewLEDMask |= LEDS_LED4;  
199
```

# USB – Ejemplo HID

## E.10 Report Descriptor (Mouse)

Item		Value (Hex)
Usage Page (Generic Desktop),		05 01
Usage (Mouse),		09 02
Collection (Application),		A1 01
Usage (Pointer),		09 01
Collection (Physical),		A1 00
Usage Page (Buttons),		05 09
Usage Minimum (01),		19 01
Usage Maximum (03),		29 03
Logical Minimum (0),		15 00
Logical Maximum (1),		25 01
Report Count (3),		95 03
Report Size (1),		75 01
Input (Data, Variable, Absolute),	;3 button bits	81 02
Report Count (1),		95 01
Report Size (5),		75 05
Input (Constant),	;5 bit padding	81 01
Usage Page (Generic Desktop),		05 01
Usage (X),		09 30
Usage (Y),		09 31
Logical Minimum (-127),		15 81
Logical Maximum (127),		25 7F
Report Size (8),		75 08
Report Count (2),		95 02
Input (Data, Variable, Relative),	;2 position bytes (X & Y)	81 06
End Collection,		C0
End Collection		C0

- Apéndice E  
HID Class devices. P 66

# USB

Class Code (hexadecimal)	Description
00	Reserved
01	Audio
02	Communications device class: communication interface
03	Human interface device
05	Physical
06	Image
07	Printer
08	Mass storage
09	Hub
0A	Communications device class: data interface
0B	Smart Card
0D	Content Security
0E	Video
0F	Personal healthcare device (can instead be declared at the device level)
DC	Diagnostic device (can instead be declared at the device level) bInterfaceSubclass = 01h, bInterfaceProtocol = 01h. USB2 compliance device
E0	Wireless controller bInterfaceSubclass = 01h bInterfaceProtocol = 01h: Bluetooth programming interface (can also be declared at the device level) bInterfaceProtocol = 02h: UWB Radio control interface (Wireless USB) bInterfaceProtocol = 03h: remote NDIS bInterfaceSubclass = 02h. Host and device wire adapters (Wireless USB)
EF	Miscellaneous bInterfaceSubclass = 01h bInterfaceProtocol = 01h: active sync bInterfaceProtocol = 02h: Palm sync bInterfaceSubclass = 03h. Cable based association framework (Wireless USB)
FE	Application specific bInterfaceSubclass = 01h. Device firmware upgrade bInterfaceSubclass = 02h. IrDA bridge bInterfaceSubclass = 03h. Test and measurement
FF	Vendor specific (can instead be declared at the device level)





# Fuentes

<http://www.beyondlogic.org/usbnutshell/usb-in-a-nutshell.pdf>

usb in a nutshell

Jan Axelson – USB COMPLETE – 4th edition

<http://www.usb.org>

<http://www.usblyzer.com/brief-usb-overview-and-history.htm>

<http://arstechnica.com/old/content/2007/09/intel-announces-demonstrates-usb-3-0.ars>

<http://arstechnica.com/hardware/news/2008/11/usb-3-0-specification-finalized-devices-in-2010.ars>

[http://techon.nikkeibp.co.jp/english/NEWS\\_EN/20090310/166949/](http://techon.nikkeibp.co.jp/english/NEWS_EN/20090310/166949/)

[http://www.reghardware.co.uk/2008/01/09/ces\\_usb\\_3\\_revealed/](http://www.reghardware.co.uk/2008/01/09/ces_usb_3_revealed/)

<http://arstechnica.com/old/content/2007/09/intel-announces-demonstrates-usb-3-0.ars>

<http://en.wikipedia.org/wiki/Usb>

[http://news.cnet.com/8301-17938\\_105-9780794-1.html](http://news.cnet.com/8301-17938_105-9780794-1.html)

<http://thefutureofthings.com/news/5739/25gb-in-70-seconds-with-usb-3-0.html>

<http://www.intel.com/pressroom/archive/releases/20080813corp.htm>

[http:// www.at91.com/repFichier/Document-123/USB-tutorial.ppt](http://www.at91.com/repFichier/Document-123/USB-tutorial.ppt)

[http://www.computer-solutions.co.uk/info/Embedded\\_tutorials/usb\\_tutorial.htm](http://www.computer-solutions.co.uk/info/Embedded_tutorials/usb_tutorial.htm)