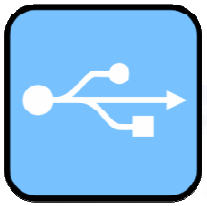
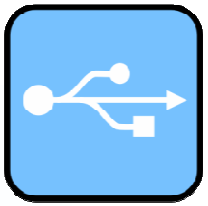
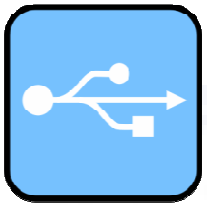




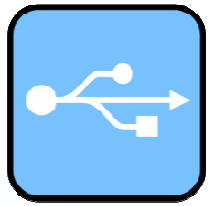
USB

-Arquitetura e Características-

Disciplina: Programação de Periféricos
Professor: Eduardo A. Bezerra
Alunos:
Carolina Metzler
Nícolas Marroni



USB – conceitos



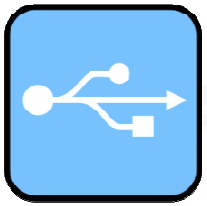
Histórico:

Em 1995, surge uma aliança promovida por várias empresas (como NEC, Intel e Microsoft) com o intuito de desenvolver uma tecnologia que permitisse o uso de um tipo de conexão comum entre computador e periféricos, essa aliança foi chamada de *USB Implementers Forum*.

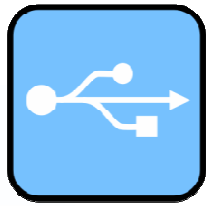
Em pouco tempo, surgia o USB, um barramento que adota um tipo de conector que deve ser comum a todos os aparelhos que o usarem. Assim, uma porta USB pode ser usada para instalar qualquer dispositivo que use esse mesmo padrão.

Conceitos do padrão USB:

- Fácil utilização na expansão de periféricos do PC
- Solução de baixo custo com velocidades acima de 12Mbit/s
- Suporte completo para dados de voz, áudio e vídeo em tempo real
- Conexão de dispositivos “on-the-fly”



USB – características



- Fácil utilização por usuários comuns

  Detalhes elétricos, com as terminações do barramento isoladas do usuário

  Auto-identificação de periféricos, configuração e mapeamento de drivers automático

- Amplo alcance de aplicações

  Suporta até 127 periféricos.

  Suporta operações concorrentes de vários dispositivos (conexões múltiplas).

- Flexibilidade

  Suporta um grande número de pacotes de tamanhos variados

- Robustez

  Erros de conexão e falha no mecanismo de recuperação são implementados no protocolo.

  A inserção e remoção dinâmica de periféricos é identificada e percebida em tempo real pelo usuário.

USB - tipos

USB Low Speed: transmite dados a uma taxa de ate 1.5Mbit/s, sendo mais usado para dispositivos com interação humana (Human Interface Devices (HID)) como teclados, mouse, etc.

USB Full Speed: usado em aplicações de telefonia, áudio e vídeo (ISDN, PBX) chegando a uma velocidade de 12Mbit/s.

USB Hi Speed: usado em aplicações de vídeo, armazenamento, transfere dados a uma taxa de ate 480Mbit/s.

USB On-The-Go: muito utilizado um dispositivos portáteis, possui conector de tamanho reduzido, capacidade de conexão com outros dispositivos usb sem a necessidade de um host.

USB Wireless: combina a velocidade de e a fácil utilização do usb padrão USB 2.0 com o a conveniência da tecnologia sem fio.

USB - conectores

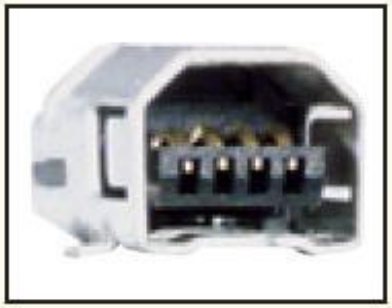
Type A Plug
(4 position)

Type A Jack
(4 position)

Type B Plug
(4 position)

Type B Jack
(4 position)

Mini Type B Plug
(4 position)

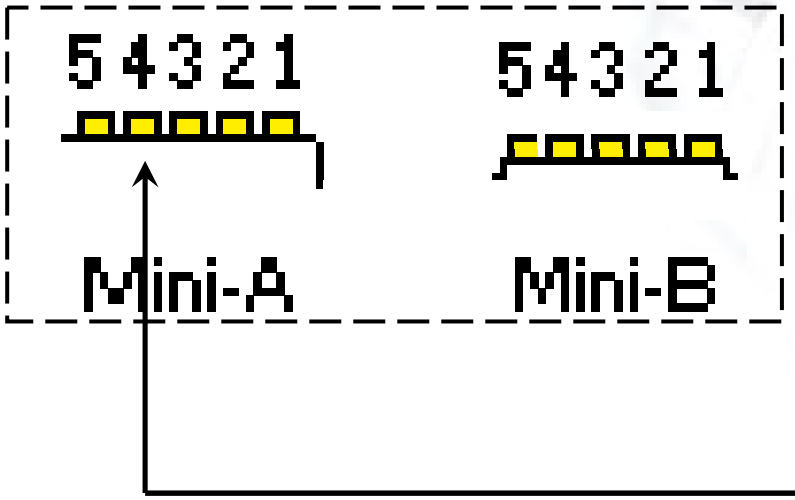


Mini Type B Jack
(4 position)

Mini Type B Plug
(5 position)

Mini Type B Jack
(5 position)

USB – pino ID



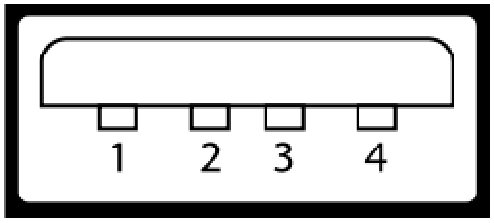
Pin	Function
1	V_{BUS} (4.4–5.25 V)
2	D–
3	D+
4	ID
5	Ground

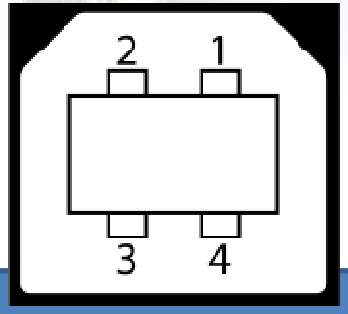
USB – pinagem

• O cabo padrão de usb é constituído de 4 fios

Pin	Signal Name	Description
1	VBUS	Red
2	D-	White
3	D+	Green
4	GND	Black

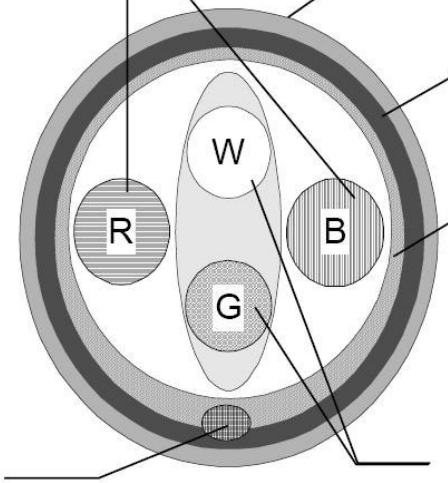
VBUS: responsável por fornecer a alimentação (+5v) para os dispositivos usb
D- : condutor
D+: condutor
GND: terra





USB – barramento físico

on-Twisted Power Pair:
Red: VBUS
Black: Power Ground



Polyvinyl Chloride (PVC) Jacket

Outer Shield > 65% Interwoven Tinned Copper Braid

Inner Shield Aluminum Metallized Polyester

Twisted Signaling Pair:
White: D-
Green:D+

28 AWG Tinned Copper Drain Wire

Cable Construction

CSMUAA, CSMUAB and CSMUAX cable series use 20 AWG power conductors. All MB (Mini B) cable series use 28 AWG power conductors.

USB – barramento físico

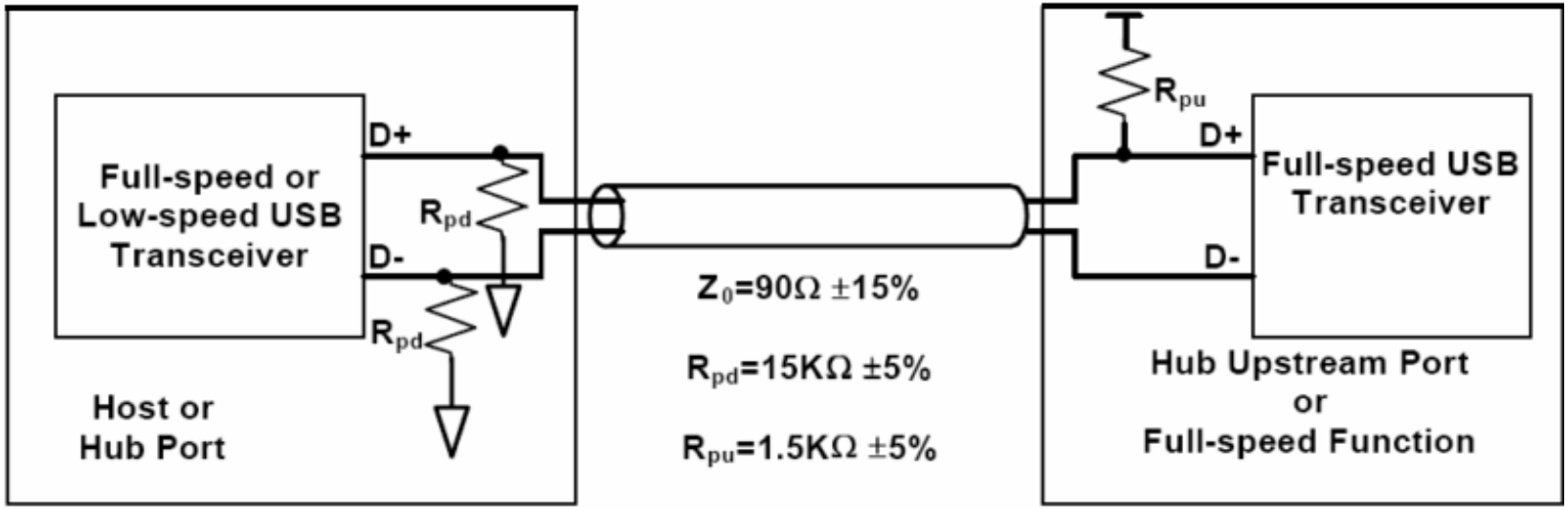


Figure 7-10. Full-speed Device Cable and Resistor Connections

- A diferença entre a USB full-speed e low-speed está exatamente na colocação do resistor pull-up, sendo que na full-speed o resistor de pull-up é ligado ao D+ e na low-speed o resistor é ligado ao D-.

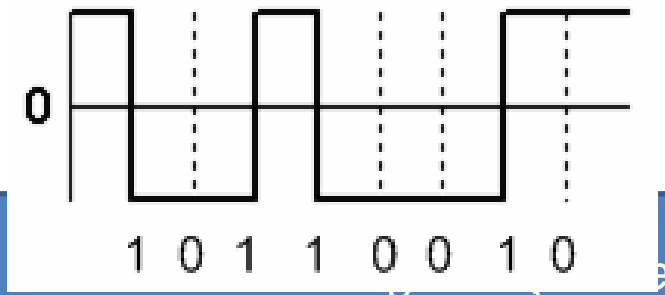
- Os dois fios de dados são trancados e conduzem os dados na forma de um sinal digital diferencial, ou seja, possuem um sinal de magnitude igual, porem de polaridade invertida.

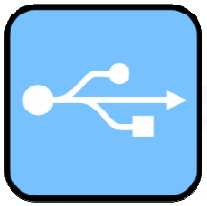
USB – codificação NRZI

- O USB usa o codificação NRZI (Non return to zero, inverted) este é um método de mapear um sinal binário para um sinal físico para transmissão.

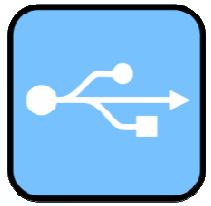
- Quando se deseja transmitir um bit 0, a codificação NRZI usada exige que se invertam os estados nas linhas D+ e D-. Já para o caso da transmissão de bits 1's, os estados são mantidos.

- Na especificação USB, julgou-se conveniente que o barramento não ficasse em estado J (D+ em alto e D- em baixo) por muito tempo durante um pacote. Para evitar isso, depois de cada seis bits 1's que devam ser transmitidos em seqüência, um bit 0 deve ser transmitido de forma redundante. Esse bit deve ser naturalmente desprezado na recepção. Diz-se que esse bit extra é um bit tolo ou "stuff".





USB – comunicação serial



Quando não existe atividade no barramento, ele entra no estado ocioso (“idle”) neste caso ele estará no estado J.

A comunicação entre dois terminais USB é realizada mediante transações, que são constituídas pela transmissão de conjuntos de pacotes.

Todas as funções (periféricos) recebem todos os pacotes, mas apenas a função endereçada utiliza o pacote enviado.

Um pacote é constituído de pelo menos três campos: sincronismo (SYNC), identificação do tipo de pacote (PID) e fim de pacote (2X SE0)

Estados	D+	D-
J	alto	baixo
K	baixo	alto
SE0	baixo	baixo

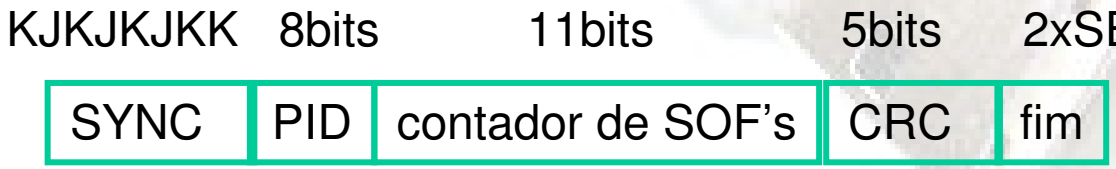
USB – tipos de pacotes

- A especificação usb 2.0 prevê 9 identificadores de pacotes, outros 6 identificadores foram reservados para uso futuro.
- Os identificadores de pacotes podem ainda ser classificados em 3 tipos: token, data e hadshake.

PID	NOME	TIPO
0101	SOF	TOKENS
1101	SETUP	
1001	IN	
0001	OUT	
0011	DATA0	DATA
1011	DATA1	
0010	ACK	HANDSHAKE
1010	NAK	
1110	STALL	

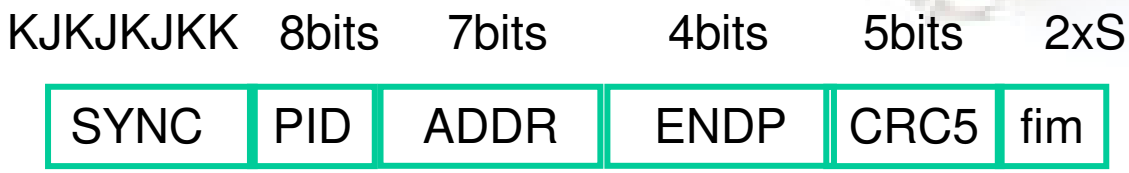
USB – SOF start of frame

- O pacote SOF e transmitido pelo hub raiz a uma taxa fixa de 1.000 pacotes/segundo (um a cada 1ms) e não tem grandes aplicações praticas.
- O intervalo entre dois SOF's consecutivos é chamado de quadro, ou frame.
- Neste pacote, após transmitir o PID 0101 (SOF), transmite também o valor de um contador crescente se SOF's de 11 bits, em seguida o hub transmite 5 bits redundantes para checagem de erros de transmissão, usando a codificação tipo CRC (cyclic redundancy check)



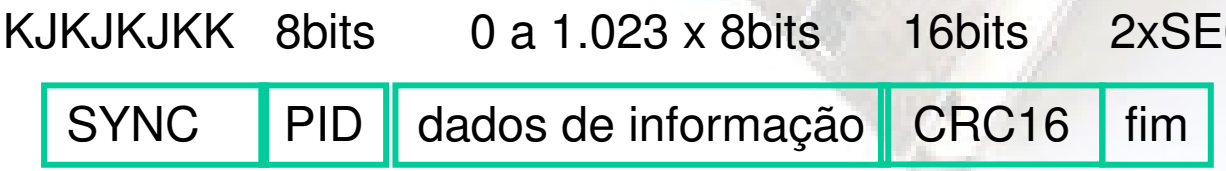
USB – Setup, In, Out

- Estes três tokens são transmitidos por hubs e tem a finalidade de selecionar quem sera o próximo endereço(dispositivo) e o próximo endpoint que serão os alvos da transação seguinte, possivelmente com pacotes DATA0 e DATA1.
- Token IN indica que o próximo pacote do tipo data sera para leitura de dados da função
- Token OUT indica que a transição sera para a escrita na função.
- Token SETUP é semelhante ao token OUT, porem não pode ser rejeitado pelo função e é sempre endereçado ao endpoint 0
- Os tokens SETUP, IN e OUT possuem ainda o campo que seleciona o endereço do dispositivo (ADDR) contem 7 bits mais 4 bits para selecionar o endpoint (ENDP), novamente 5 bits redundantes são utilizados para verificar erros de transmissão.



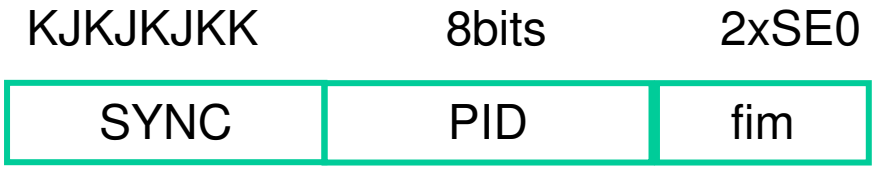
USB – DATA1 e DATA0

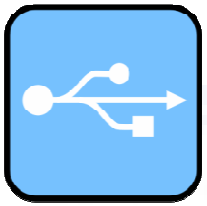
- Os pacotes DATA1 e DATA0 transportam a informação propriamente dita que sera trocada entre o hub e a função
- Caso o ultimo pacote de token transmitido tenha sido um OUT ou um SETUP todo o pacote, incluindo os dados de informação será obrigatoriamente do hub para a função
- Caso o ultimo token tenha sido do tipo IN, o fluxo é do tipo upstream (da função para o hub)



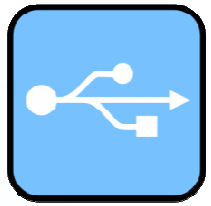
USB – ACK, NAK e STALL

- Os pacotes ACK, NAK e STALL são compostos exclusivamente pelos campos obrigatórios, ou seja, SYNC, PID e FIM.
- O fluxo de transmissão é contrário ao fluxo de dados.
- Caso o pacote do tipo data tenha seguido o fluxo upstream, o pacote handshake seguinte terá seguido o fluxo dowstream e vice-versa.
- O uso ou não dos handshake vai depender de como a função foi configurada
- Um ACK serve para informar que o ultimo pacote do tipo data ou token foi recebido corretamente
- O NAK indica que o receptor esta ocupado ou inapto a realizar a comunicação aquele momento.
- Um STALL indica que o receptor detectou erros de comunicação, como, por exemplo, enviar um token OUT para um endpoit só de leitura.





USB- transações (endpoints types)



Possui quatro tipos básicos de transferências de dados:

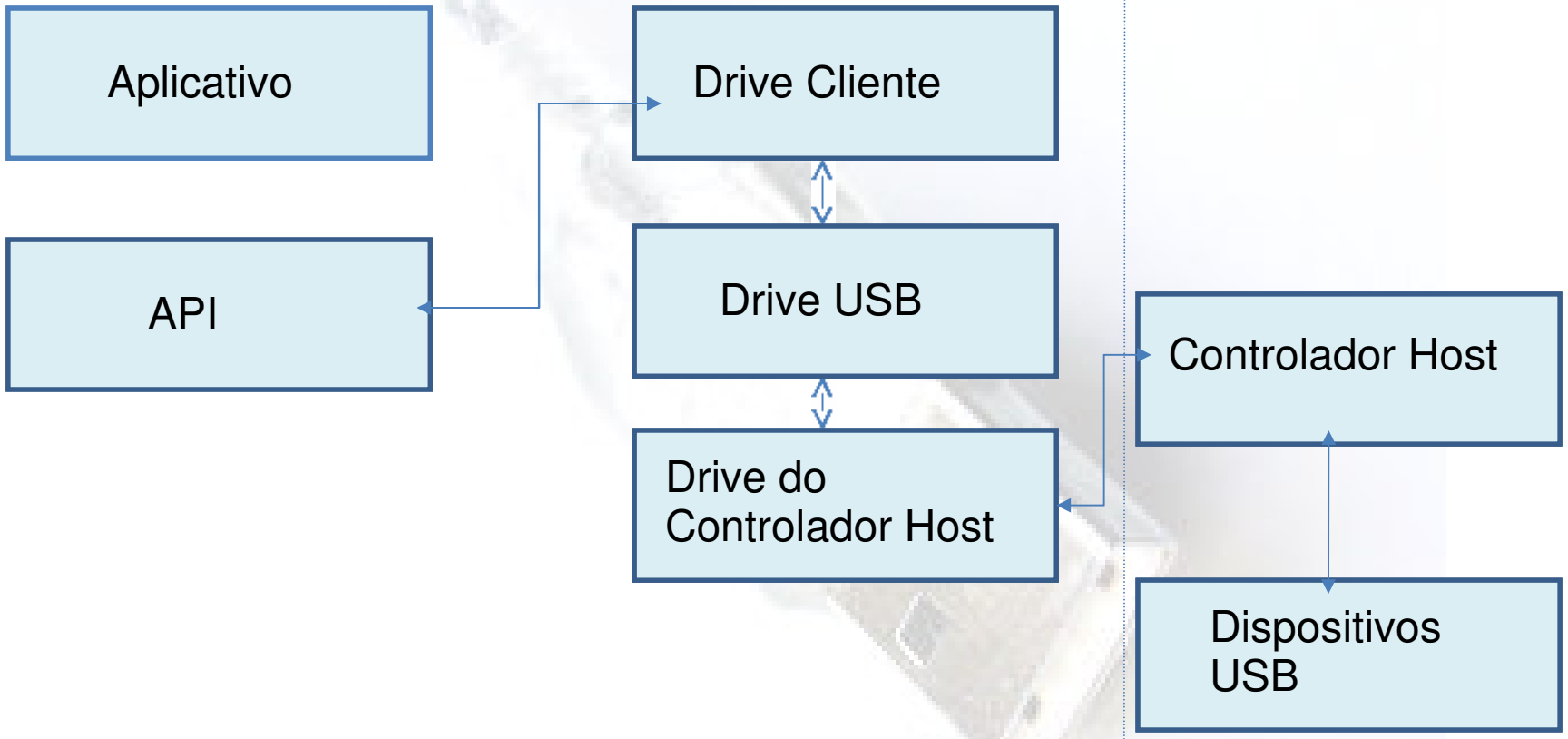
Control: Usada para configurar um dispositivo no instante de sua conexão e pode ser usada para outros propósitos específicos, incluindo controle de outros *pipes (canais entre o host e endpoints)* no dispositivo.

Bulk: Gerada e consumida em grandes quantidades e simultaneamente. Possui uma ampla e dinâmica latitude em transmissões de reserva.

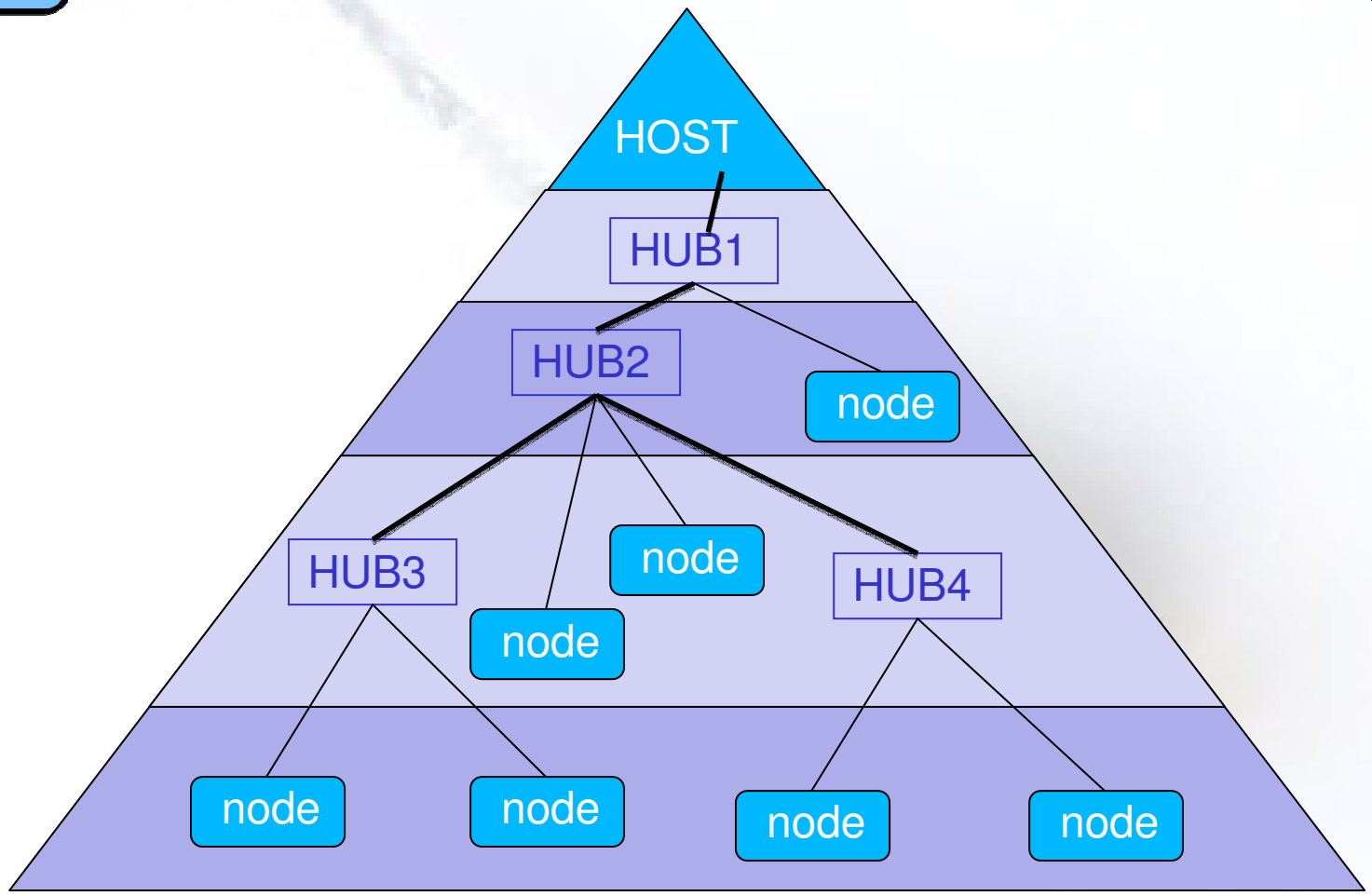
Interrupt: Usada para caracteres ou coordenadas com percepções humanas ou características de respostas regenerativas.

Isochronous: Ocupa uma quantidade pré-negociável da banda de transmissão do barramento, com a distribuição de pulsos. Chamada também de transferência de correntes em tempo real (*streaming real-time transfers*).

USB- funcionamento



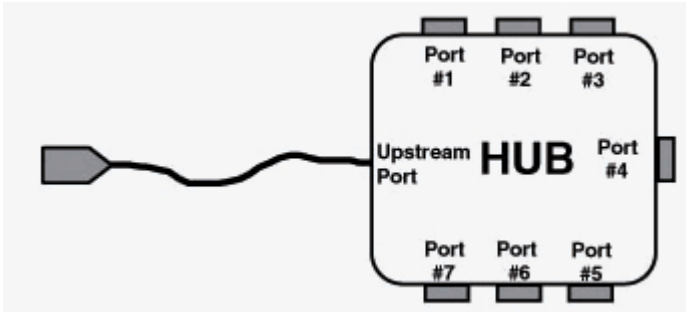
USB – topologia

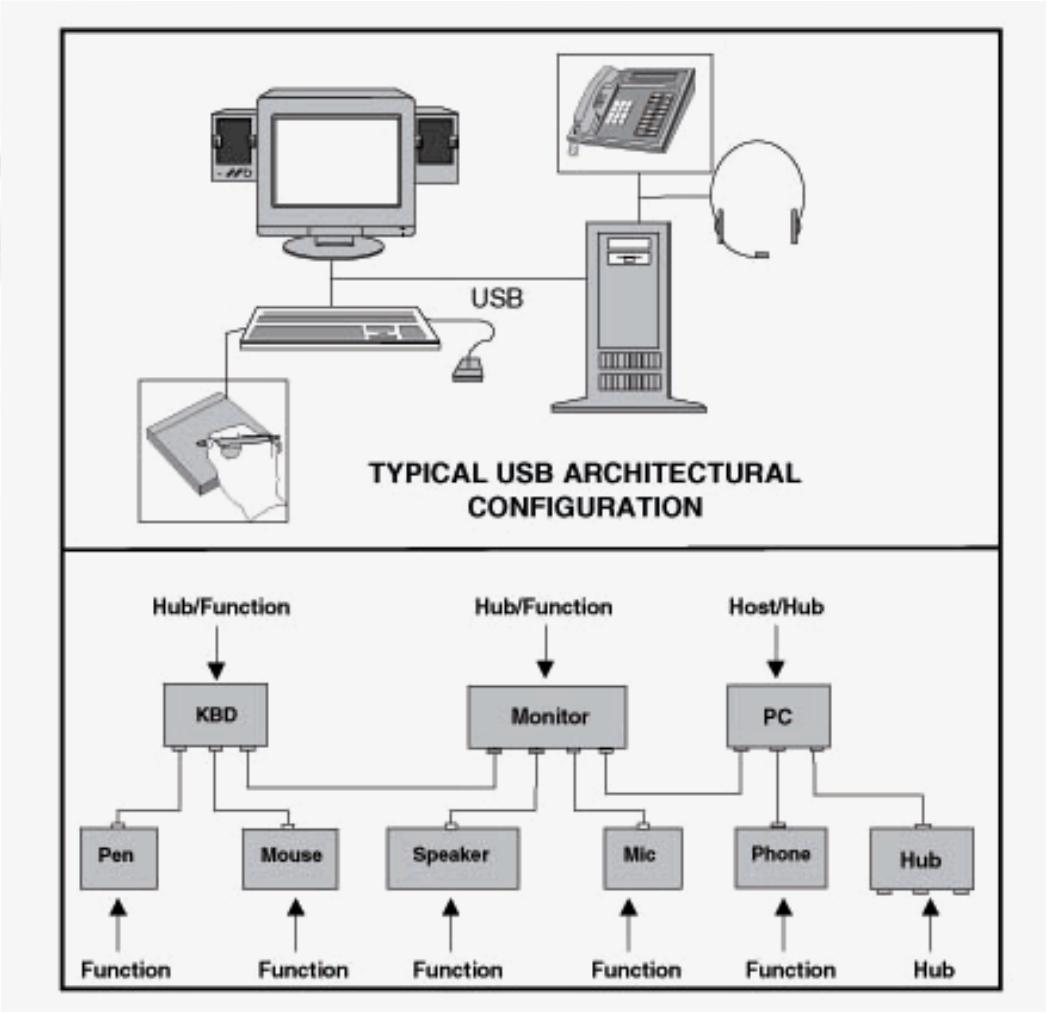


Observação: os nodes são as funções dos HUBs.

USB – hub

Duas partes constituem um *hub*: o controlador e o repetidor. Este último é um protocolo de mudança controlado entre as portas secundárias e a primária. Também possui suporte de *hardware* para reiniciar e suspender a transmissão de sinais. O controlador do *host* propicia aos registradores da interface permitir a comunicação de/para o *host*. Comandos de controle e estados específicos permitem ao *host* configurar um *hub*, além de monitorar e controlar as suas portas.





USB – topologia

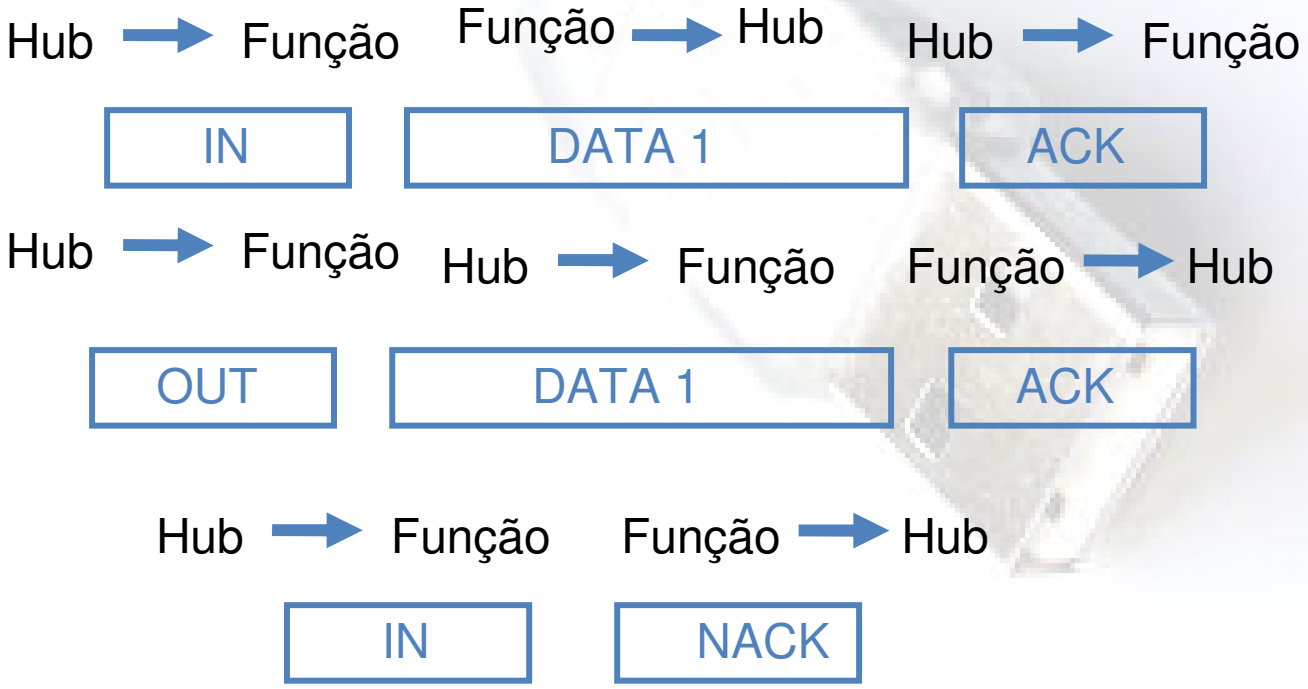
O computador pessoal é o hub base para o sistema USB e é denominado host. O software/hardware no computador pessoal que controla o hub e todo o sistema USB é denominado “controlador do barramento”. Cada sistema USB possui apenas um controlador de barramento.

O software que implemente o protocolo USB se encarrega de gerenciar toda a árvore de hubs e funções, que é construída de forma simples bastando utilizar as regras para conexão ilustradas na figura anterior. Não existe um limite teórico para o numero de hubs a serem utilizados, porem existe um numero máximo de 127 funções a serem utilizadas em um sistema (árvore) USB. Esse limite é imposto pelos 7 bits utilizados no endereçamento das funções (um endereço e reservado).

USB – transações

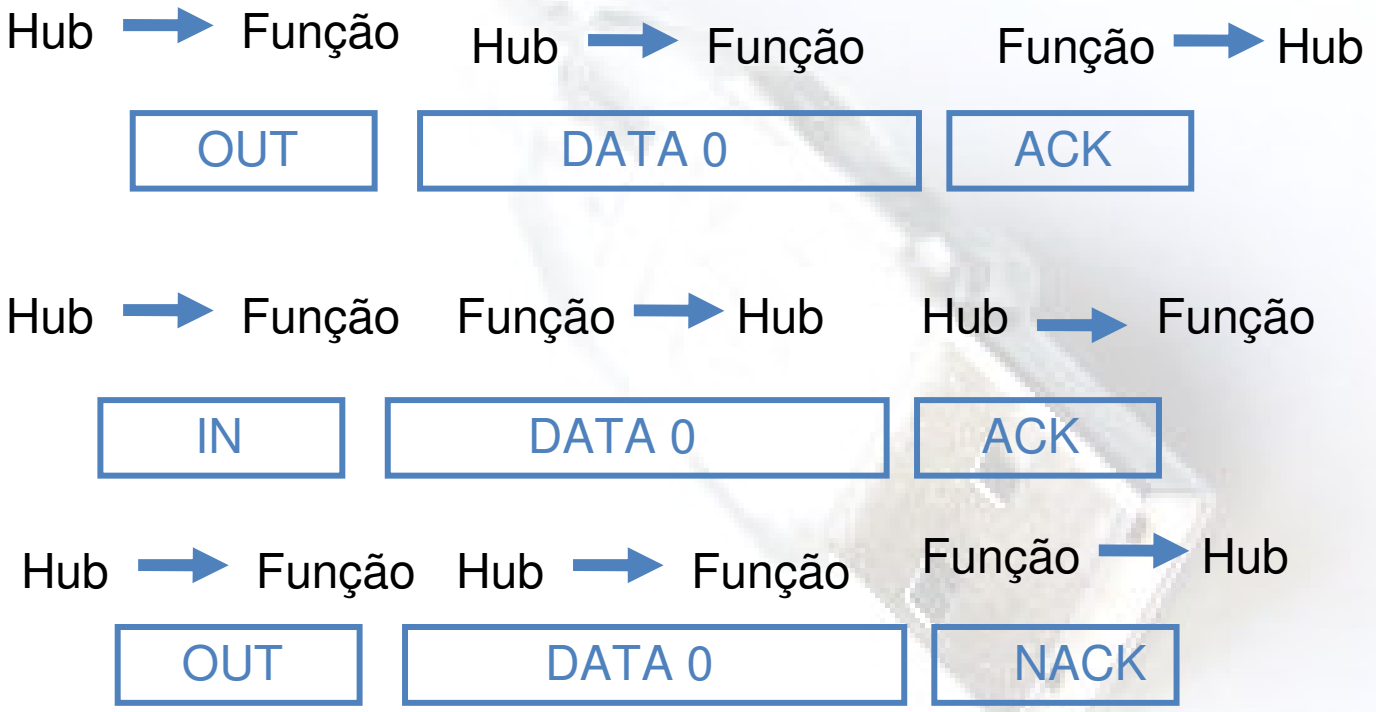
Transações são chamadas também de endpoints types.

Interrupt e bulk



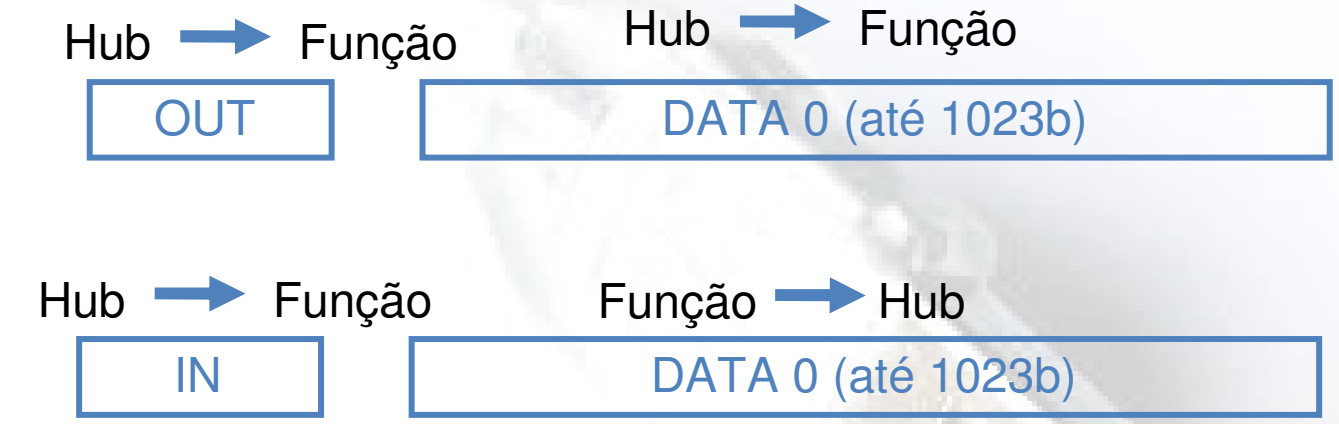
USB – transações

Bulk

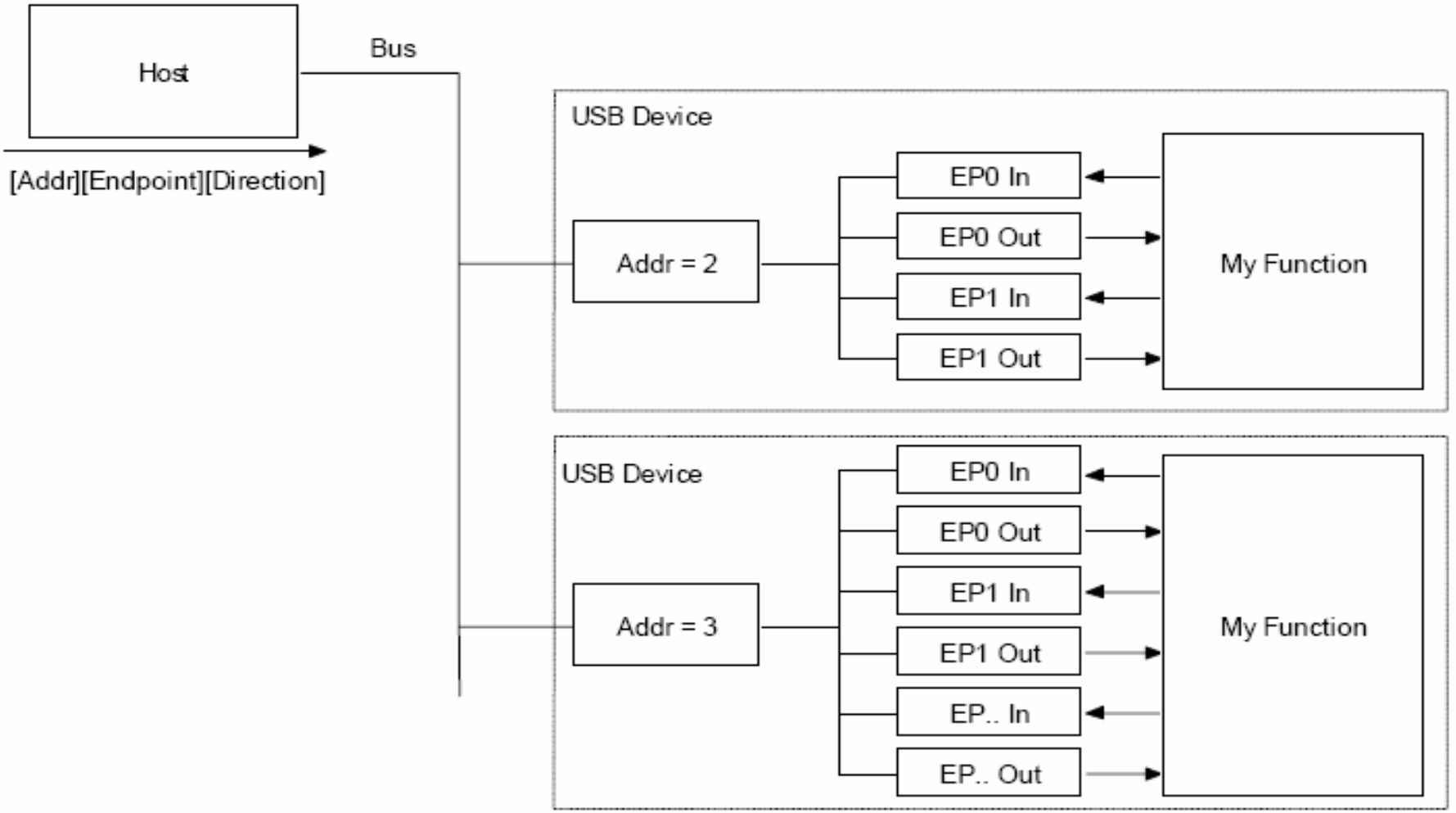


USB – transações

Isochonous



USB – funções


Host
Bus
[Addr][Endpoint][Direction]
USB Device
Addr = 2
EP0 In
EP0 Out
EP1 In
EP1 Out
My Function
USB Device
Addr = 3
EP0 In
EP0 Out
EP1 In
EP1 Out
EP.. In
EP.. Out
My Function

USB – enumeração

O Processo de Enumeração é a designação dada ao mecanismo de configuração das funções e dispositivos conectados ao *Host;*

1. Dispositivo é conectado ao *Host*;
2. O *Host* detecta eletricamente o *Host*;
3. O Controlador *Host* requisita ao dispositivo que ele reinicie;
4. O *Host* caracteriza um caminho de comunicação com o dispositivo;
5. O *Host* requisita o tamanho máximo do *pipe* (caminho de comunicação) do dispositivo;
6. O *Host* atribui um único endereço ao dispositivo;
7. O *Host* requisita os descritores e carrega o *device driver* apropriado;

USB – descritores

Todos devices USB possuem uma hierarquia de descritores que enviam informações ao HOST do defice (tipo, fabricante, versao de USB que suporta, qual o tipo de configuração, numero de endpoints, etc)

Campo	Nº Bytes
Comprimento	1
Tipo	1
Comprimento Total	2
Num. Interfaces	1
Índice da Configuração	1
Nome da Configuração	1
Atributos	1
Carga Máxima	1

Descritor de configuração

Campo	Nº Bytes
Comprimento	1
Tipo	1
Índice da interface	1
Alternativo	1
Endpoints	1
Classe	1
Subclasse	1
Protocolo	1
Nome da Interface	1

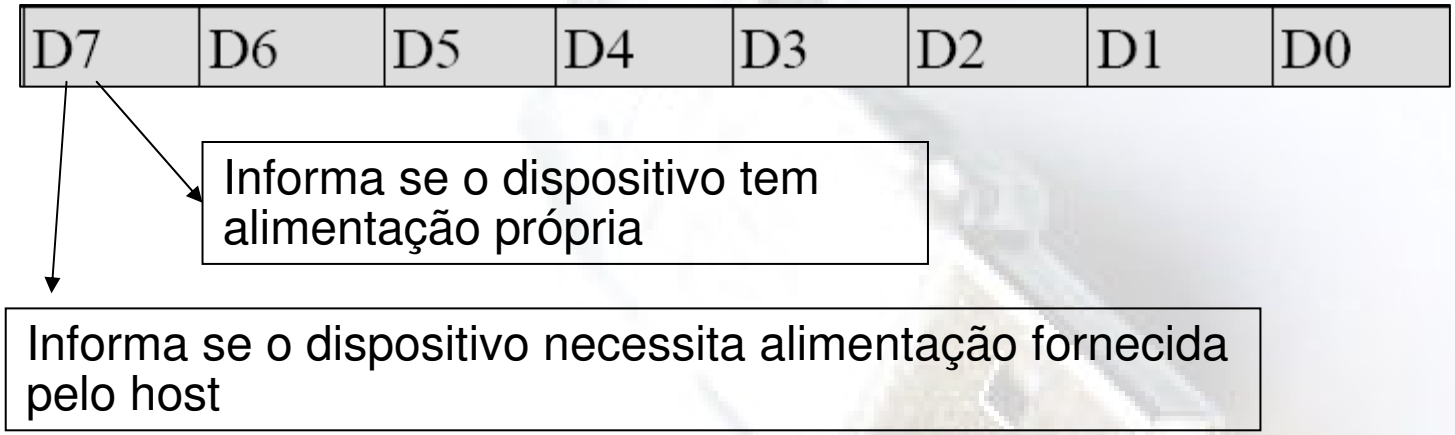
Descritor de interface

Campo	Nº Bytes
Comprimento	1
Tipo	1
Versão HID	2
Código do País	1
Descritores Restantes	1
Código 22h (relatório)	1
Tamanho do Relatório	2

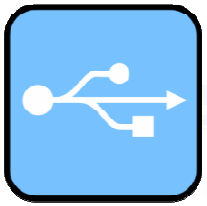
Descritor de HID

USB – descritores

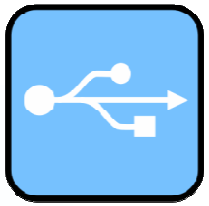
Campo Atributos (Descritor de Configuração)



Classe HID(descritor HID): driver de funções básicas IO, que faz parte do SISOP



USB – descritores



Campo	Nº Bytes
Comprimento	1
Tipo	1
Endereço do endpoint	1
Atributos	1
Tamanho Máximo do Pacote	2
Intervalo de Pooling	1

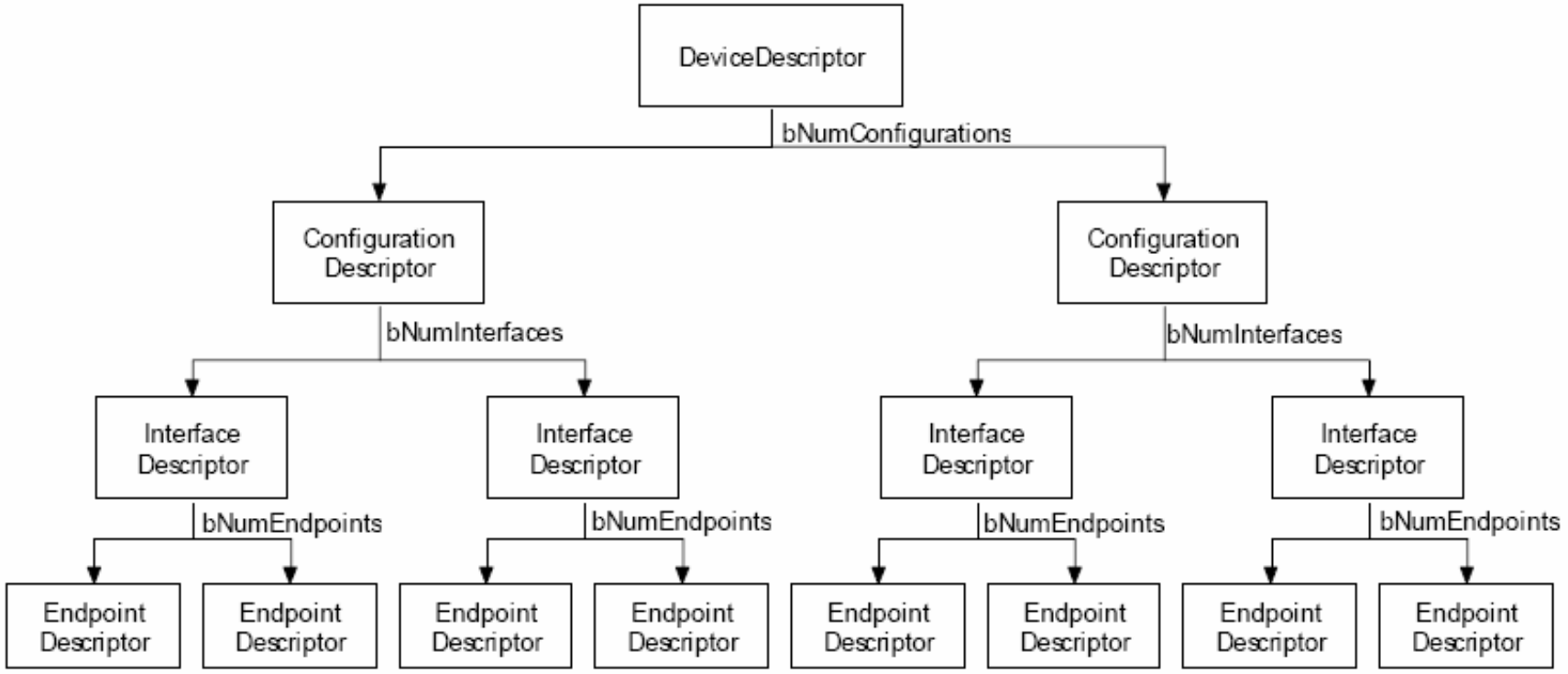
Descritor de endpoint

Este tem a função de informar o HOST sobre um canal de comunicação (pipe) específico. Podemos configurar um dispositivo para ter mais de um pipe

Campo	Nº Bytes
Comprimento	1
Tipo	1
versão USB	2
classe	1
subclasse	1
protocolo	1
Tamanho EP0	1
ID fabricante	2
ID produto	2
versão	2
Nome fabricante	1
Nome do produto	1
Numero serial	1
Número de configurações	1

Descritor de dispositivos

USB – descritores



USB – programação

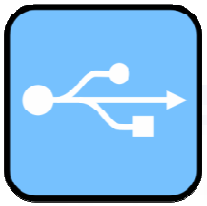
Windows

Problemas encontrados: Códigos fechados, dificuldade para compilar, pouca documentação

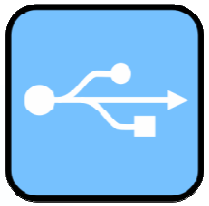
X

Linux

Mais fácil por ser open source, mais recursos online e pessoas divulgando material, não precisa de APIs proprietárias.



USB – programação



Principais funções para programação no Windows

- **HidD_GetHidGuid** (GUID *hidGUID)
- HDEVINFO **SetupDiGetClassDevs** (GUID hidGUID, PSTR *filterString, hwndParent, DWORD flags)
- BOOL **SetupDiEnumDeviceInterfaces** (infoset, infoData, interfaceClassGuid, index, deviceInterfaceData)
- BOOL **SetupDiGetDeviceInterfaceDetail** (infoSet, deviceInterfaceData, interfaceDetail, interfaceDetailSize, requiredSize, infoData)
- HANDLE **CreateFile** (name, access, sharemode, security, creation, flags, template)
- BOOL **HidD_GetAttributes** (deviceHandle, &deviceAttributes))

Bibliografia

- USB in a Nutshell _http://www.beyondlogic.org/
- Usb_bezerra.pdf _http:/www.inf.pucrs.br/~eduardob
- Arquitetura USB _http://www.pads.ufrj.br/~rapoport/usb/usb4.html
- Material USB_http://www.pads.ufrj.br/~rapoport/usb/usb4.html
- Programação_www2.hawaii.edu/~hermany
- Programação_http://www.lrr.in.tum.de/Par/arch/usb/usbdoc/
- Programação_http://msdn2.microsoft.com/en-us/library/aa363451.aspx
- Programação_http://www.lrr.in.tum.de/Par/arch/usb/usbdoc/node20.html