



通用串行总线(USB) 2.0

STM32 USB 开发者培训

2012

2013年5月

- **USB技术简介**
 - 技术背景
 - 架构、系统、协议和供电
 - 设备的枚举、识别
 - 各类描述符解析
 - 传输类型

- 通用串行总线(USB)协议的设计目标：
 - 易于使用的PC外设
 - 以低成本的方案支持高达480Mbps的传输速率
 - 满足声音，音频和视频类传输的实时需求
 - 灵活的协议，能混合同步和异步的消息数据传递
 - 集成商品设备技术
 - 能适应于任意外形和配置的PC
 - 提供一个标准接口，能快速应用于产品中
 - 允许扩展出新的USB设备类，以提升PC的功能
 - USB2.0协议必需向下兼容，以容纳早期版本的设备

应用范围

性能	应用	特性
<u>低速(1.5Mbps):</u> ✓交互式设备 ✓ 10-100kbps	➤键盘, 鼠标 ➤手写笔 ➤游戏手柄 ➤虚拟设备 ➤外设	•极低的成本 •易于使用 •热插拔 •同时使用多个外设
<u>全速(12Mbps):</u> ✓电话, 音频类 ✓压缩的视频类 ✓ 500kbps – 10Mbps	➤话音 ➤宽带 ➤音频 ➤麦克风	•较低的成本 •易于使用 •热插拔 •同时使用多个外设 •可保证的带宽 •可保证的延迟
<u>高速(480Mbps):</u> ✓视频, 大容量存储 ✓ 25 – 400Mbps	➤视频 ➤大容量存储 ➤图像 ➤宽带	•低成本 •易于使用 •热插拔 •同时使用多个设备 •可保证的带宽 •可保证的延迟 •高带宽

特性:

简单易用

使用统一制式的电缆和连接插座，支持热插拔

应用广泛

支持从几kbps到几百Mbps的数据带宽，支持同步和异步的传输，支持连接多达127个设备，并支持复合设备

同步带宽

提供保证的带宽和低延迟

使用灵活

支持不同大小的数据包和各种传输速率

鲁棒性佳

多种的错误校验和恢复机制

协同PC产业

协议易于实现和整合，并支持热插拔机制

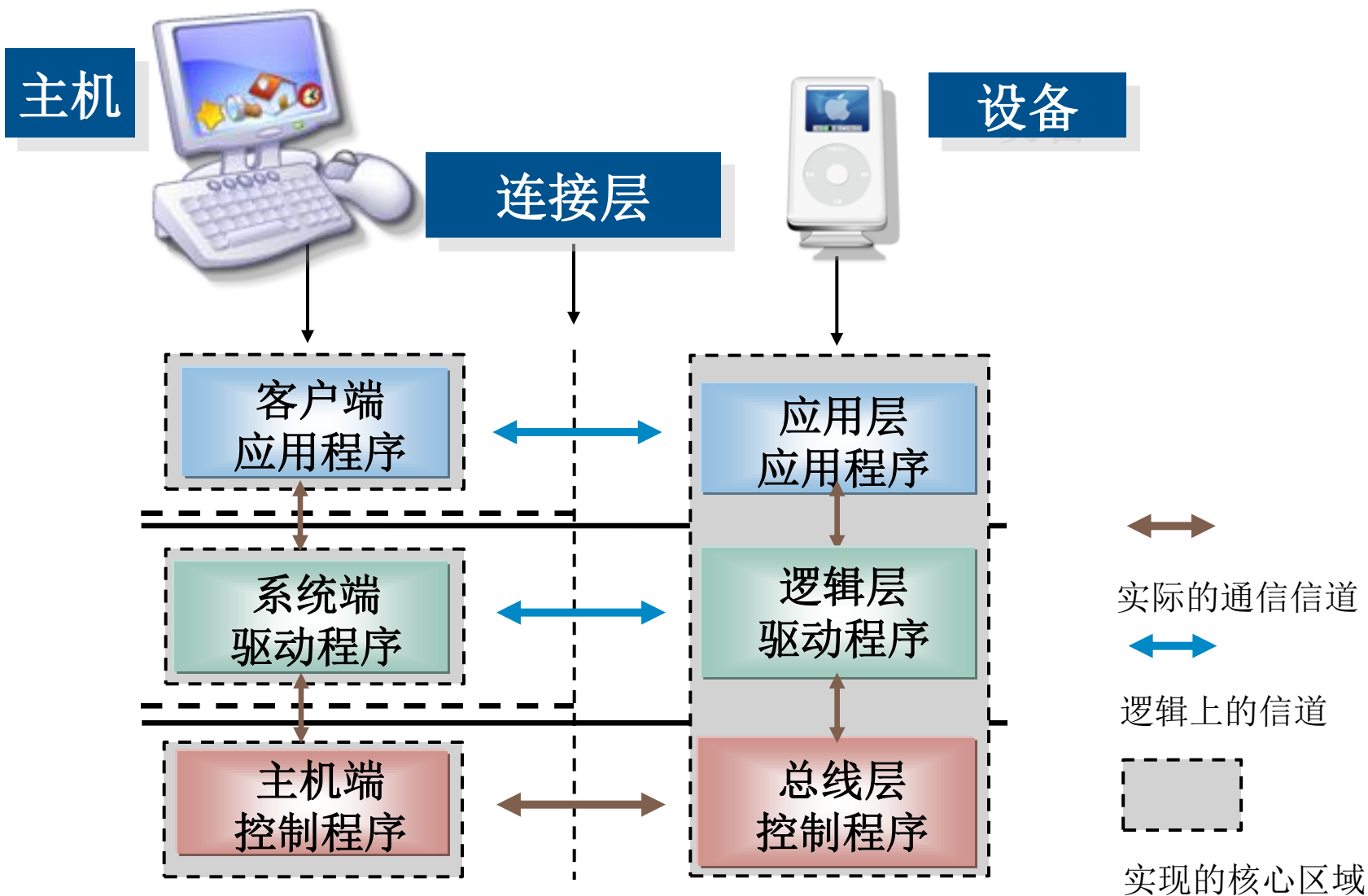
低成本实现

低成本的电缆和连接插座，商品化的实现技术

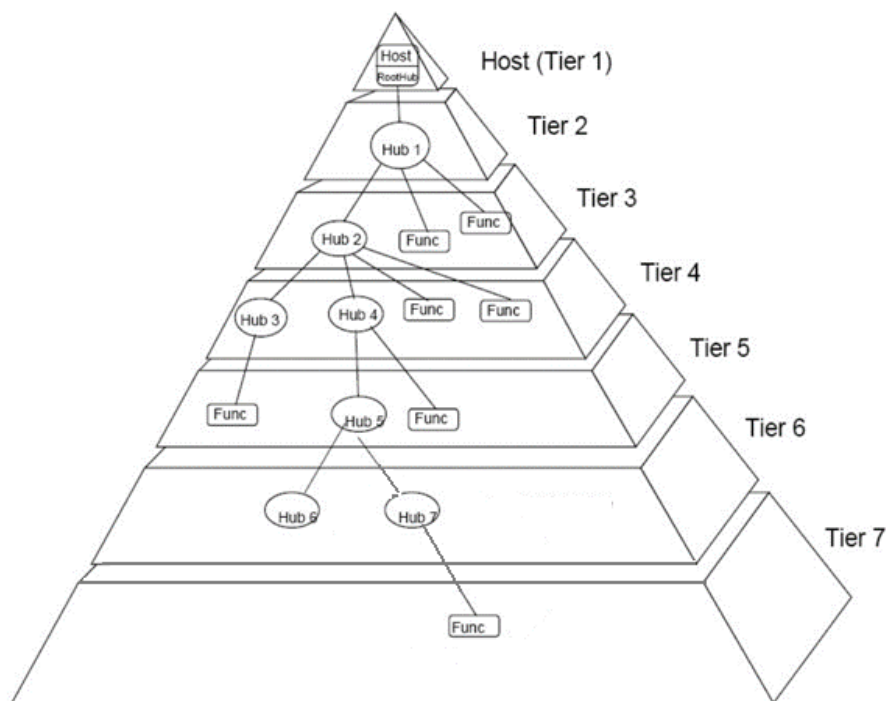
易于升级

整体结构易于升级，能适应各类新生的应用

USB 系统结构



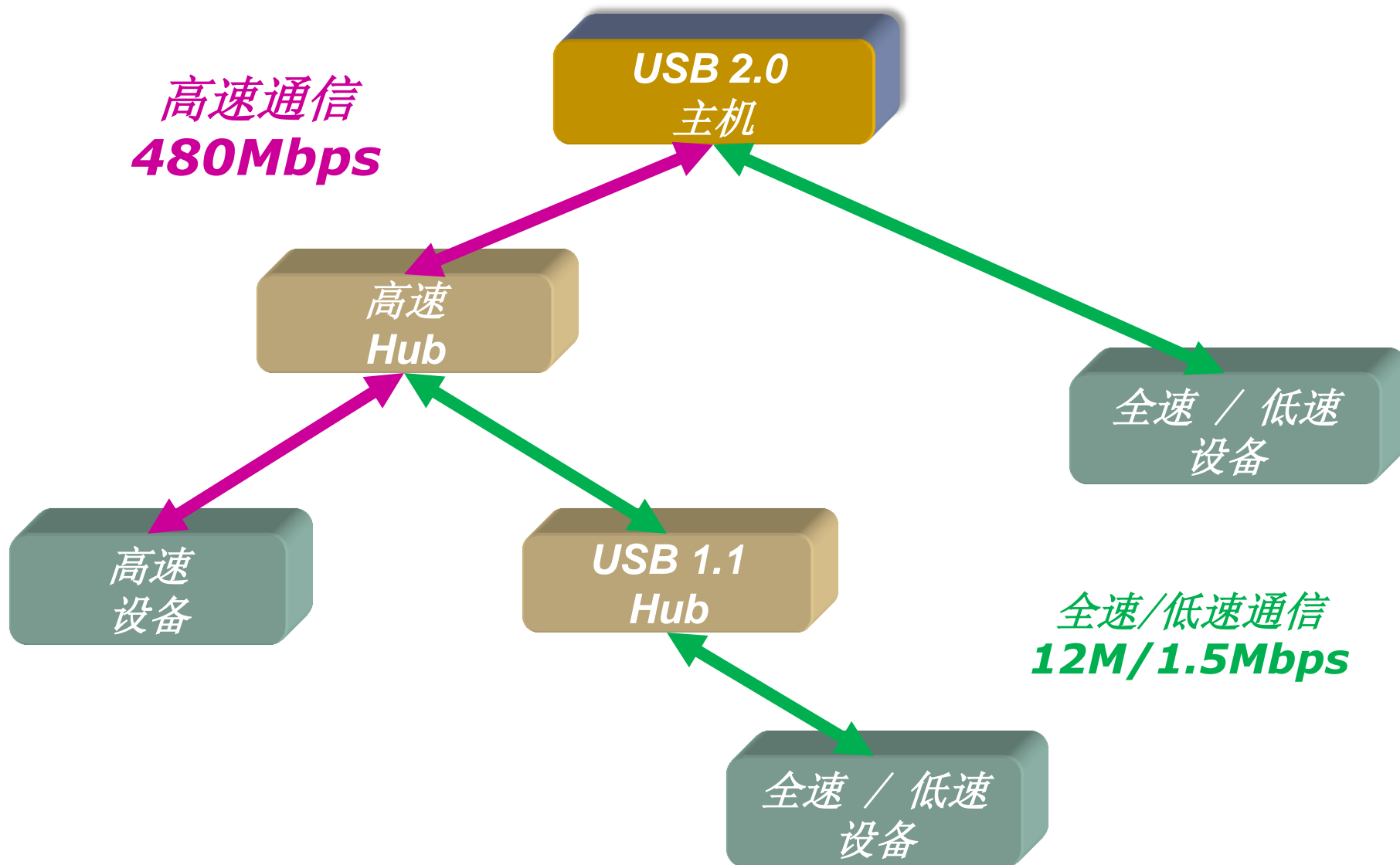
- USB总线基于分层的星状拓扑结构
- 以Hub为中心，连接周围设备
- 总线上最多可连接127个设备
- Hub串联数量最多5个
- 线缆长度最长5米



USB 拓扑结构示例

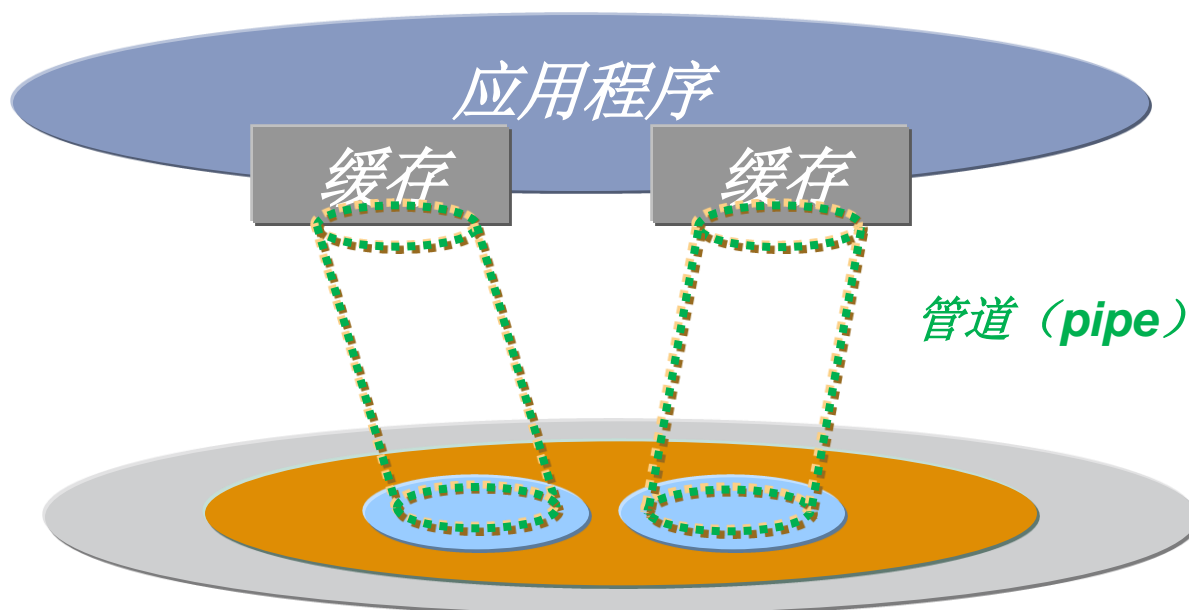


高速通信
480Mbps



主机

设备



接口（一个或多个）：用于描述特定功能，包含若干端点

端点（一个或多个）：传输的最终对象

- >> 端点号、传输类型、
- >> 传输方向、最大数据包长度...

- Two possible power configurations
 - Self-powered: power provided from external power-supply
 - Bus-powered: power provided from VBUS (5v)
- 对总线供电设备，才区分低功耗和高功耗USB设备之分
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 consumption in the device **configuration descriptor** (see slide after)

USB 配置描述符（以Joystick为例）

Configuration Descriptor

0x09,

/* 描述符的长度：9字节 */

USB_CONFIGURATION_DESCRIPTOR_TYPE,

/* 描述符的类型：0x02 配置描述符(Configuration) */

JOYSTICK_SIZ_CONFIG_DESC, 0x00,

/* 完整的描述符包括接口描述符、端点描述符和类描述符的长度 */

0x01,

/* 配置所支持的接口数目：1*/

0x01,

/* 用SetConofiguration()选择此配置，所指定的配置号： 1*/

0x00,

/* 用于描述此配置的字符描述符的索引号： 0 */

0xE0,

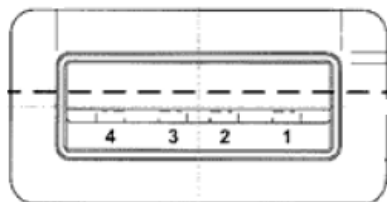
/* 供电配置： B7(1 保留), B6(自供电), B5(远程唤醒), B4-B0(0 保留) */

0x32,

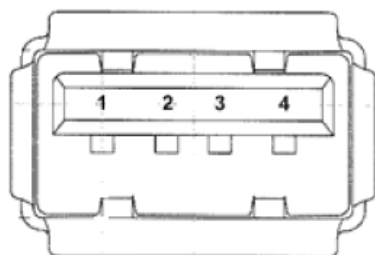
/* 最大功耗，以2mA为单位计算： 0x32表示 $50 \times 2 = 100\text{mA}$ */

- USB device should enter in USB Suspend mode when the bus is idle for more than 3 ms 该【USB挂起模式】是针对USB设备来说的
 - 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设备要退出【挂起模式】可以由Host唤醒，或者自己唤醒

USB 连接器：标准接口/Mini接口

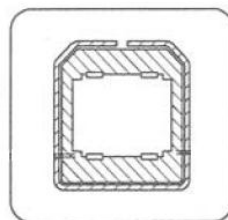


Standard A Plug

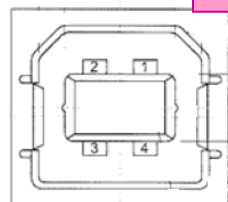


Standard A Receptacle

在PC端

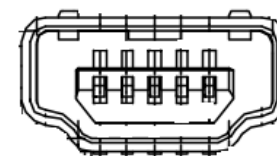


Standard B Plug

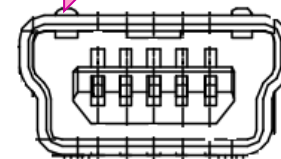


Standard B Receptacle

在设备端



Mini B Plug

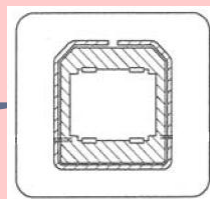
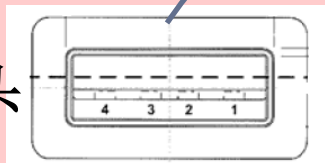


Mini B Receptacle

体积更小.....

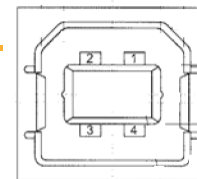
一根电缆

标准A插头

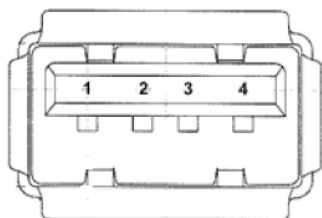


标准B插头

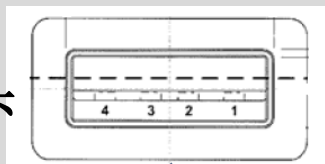
设备板子上的
标准B插座



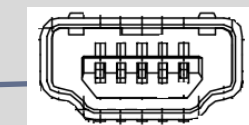
PC上的
标准A插座



标准A插头

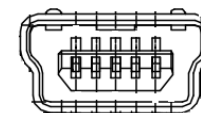


一根电缆

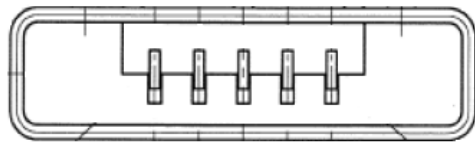


miniB插头

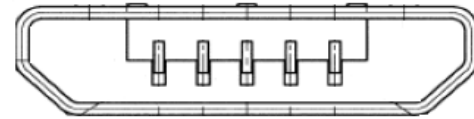
设备板子上的
miniB插座



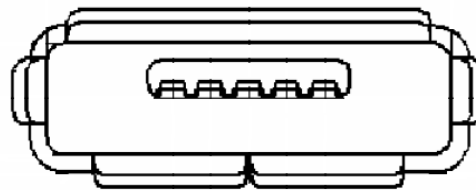
USB 连接器: Micro接口



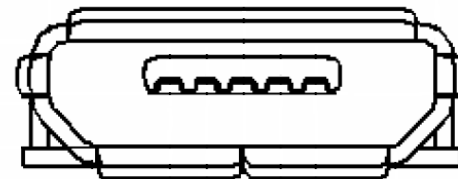
Micro A Plug



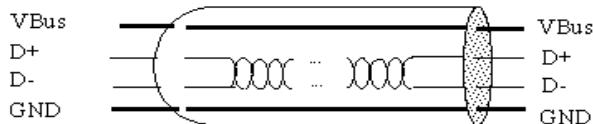
Micro B Plug



Micro AB Receptacle



Micro B Receptacle

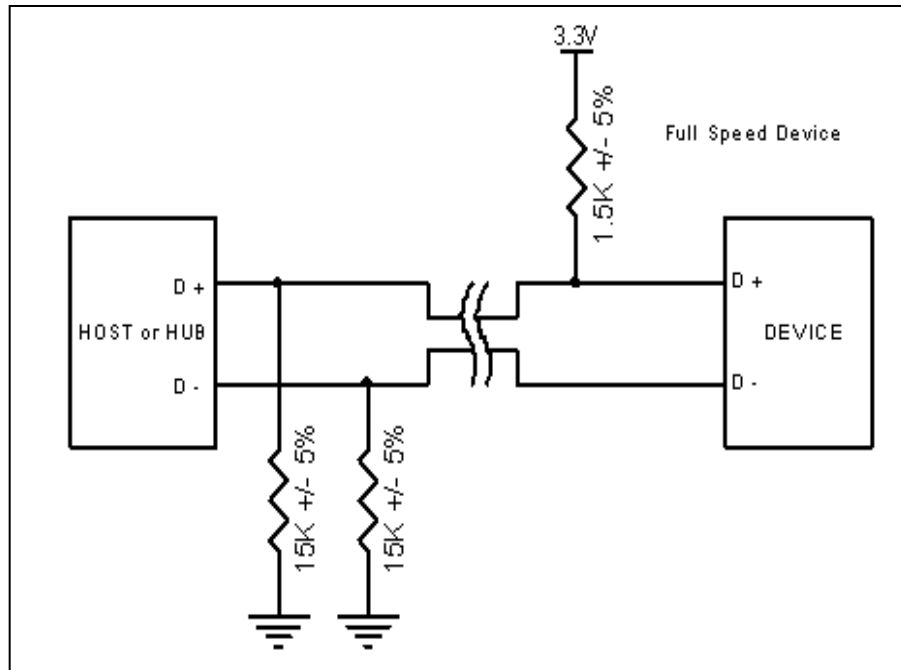


- USB使用的是差分传输模式，两个数据线D+和D-
 - 差分信号1: $D+ > V_{OH} (2.8V)$ and $D- < V_{OL} (0.3V)$
 - 差分信号0: $D- > V_{OH}$ and $D+ < V_{OL}$
 - 复位信号 (Reset) : $D+ \text{ and } D- < V_{OL}$ for $\geq 10ms$
 - IDLE状态: J状态
 - 恢复信号 (Resume) : K 状态
 - SOP: 从IDLE状态切换到K状态
 - EOP: 持续2位时间的SE0信号，后跟随1位时间的J状态
 - SYNC: 3个重复的KJ状态切换，后跟随2位时间的K状态

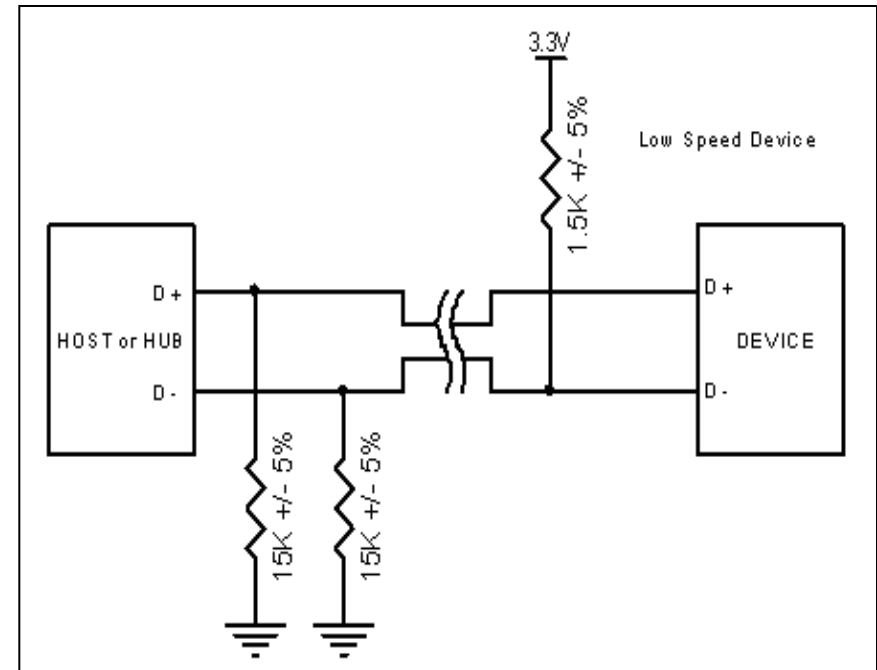
J状态	LS	差分0
	FS	差分1
K状态	LS	差分1
	FS	差分0



USB设备的插入检测和速度检测



Full/high Speed: Pull-up on D+



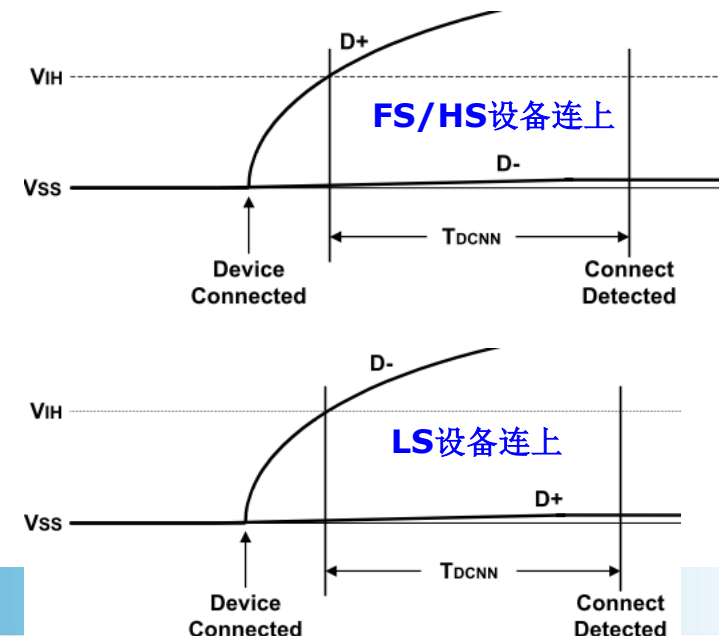
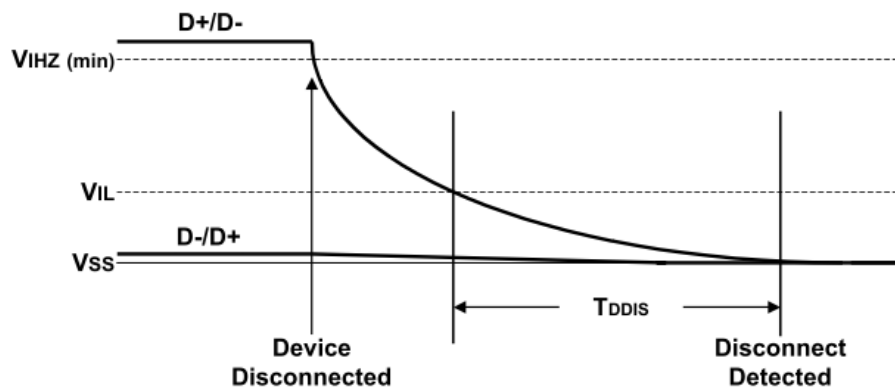
Low Speed: Pull-up on D-

- 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”

连接和断开连接



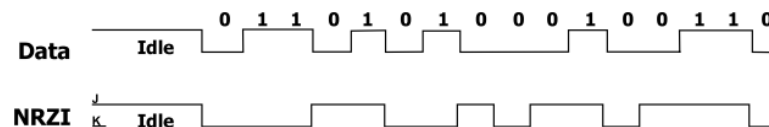
- 没有设备连上主机时
 - D+和D-数据线上的下拉电阻起作用，使得二者都在低电平；主机端看来就是个SE0状态；同样地，当数据线上的SE0状态持续一段时间了，就被主机认为是断开状态
- 设备连上主机时
 - 当主机检测到某一个数据线电平拉高并保持了一段时间，就认为有设备连上来了
 - 主机必需在驱动SE0状态以复位设备之前，立刻采样总线状态来判断设备的速度



数据编解码和位填充

■ USB采用NRZI（非归零编码）对发送的数据包进行编码

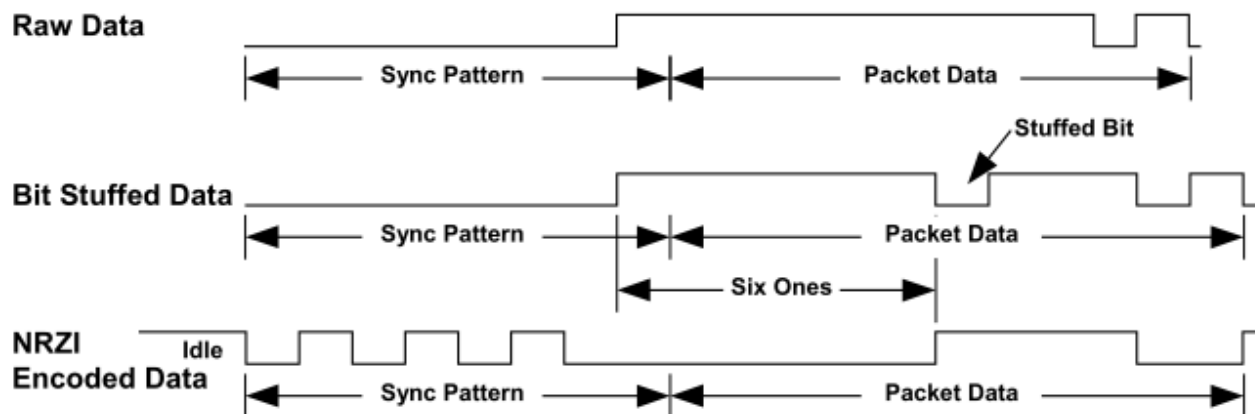
- 输入数据0， 编码成“电平翻转”
- 输入数据1， 编码成“电平不变”
- 编码出来的序列， 高电平：J状态； 低电平：K状态



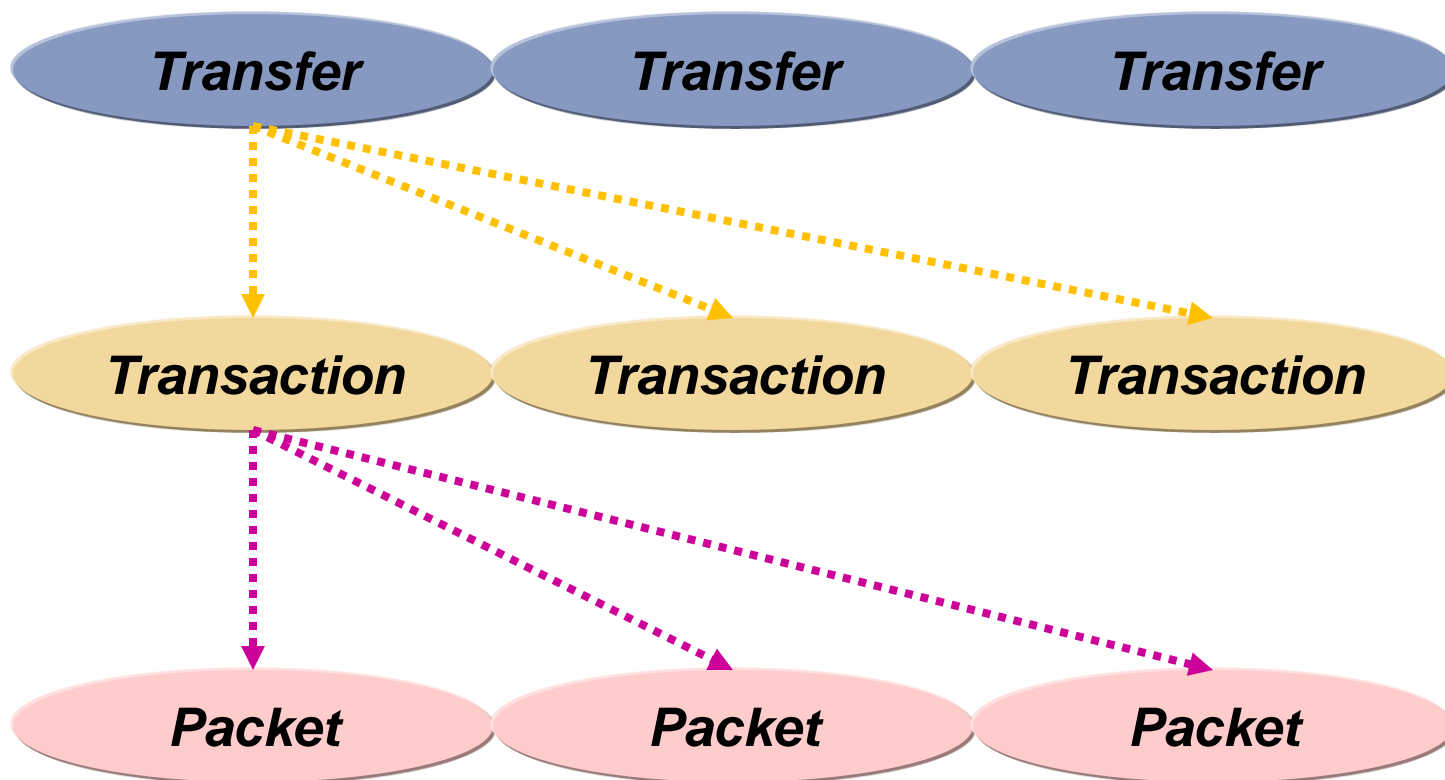
■ 位填充是为了保证发送的数据序列中有足够多的电平变化

- 填充的对象是（输入数据）， 即先填充再编码
- 数据流中每6个连续的“1”， 就要插入1个“0”， 从而保证编码数据出现电平变化
- 接收方赋值解码NRZI码流， 然后识别出填充位， 并丢弃它们

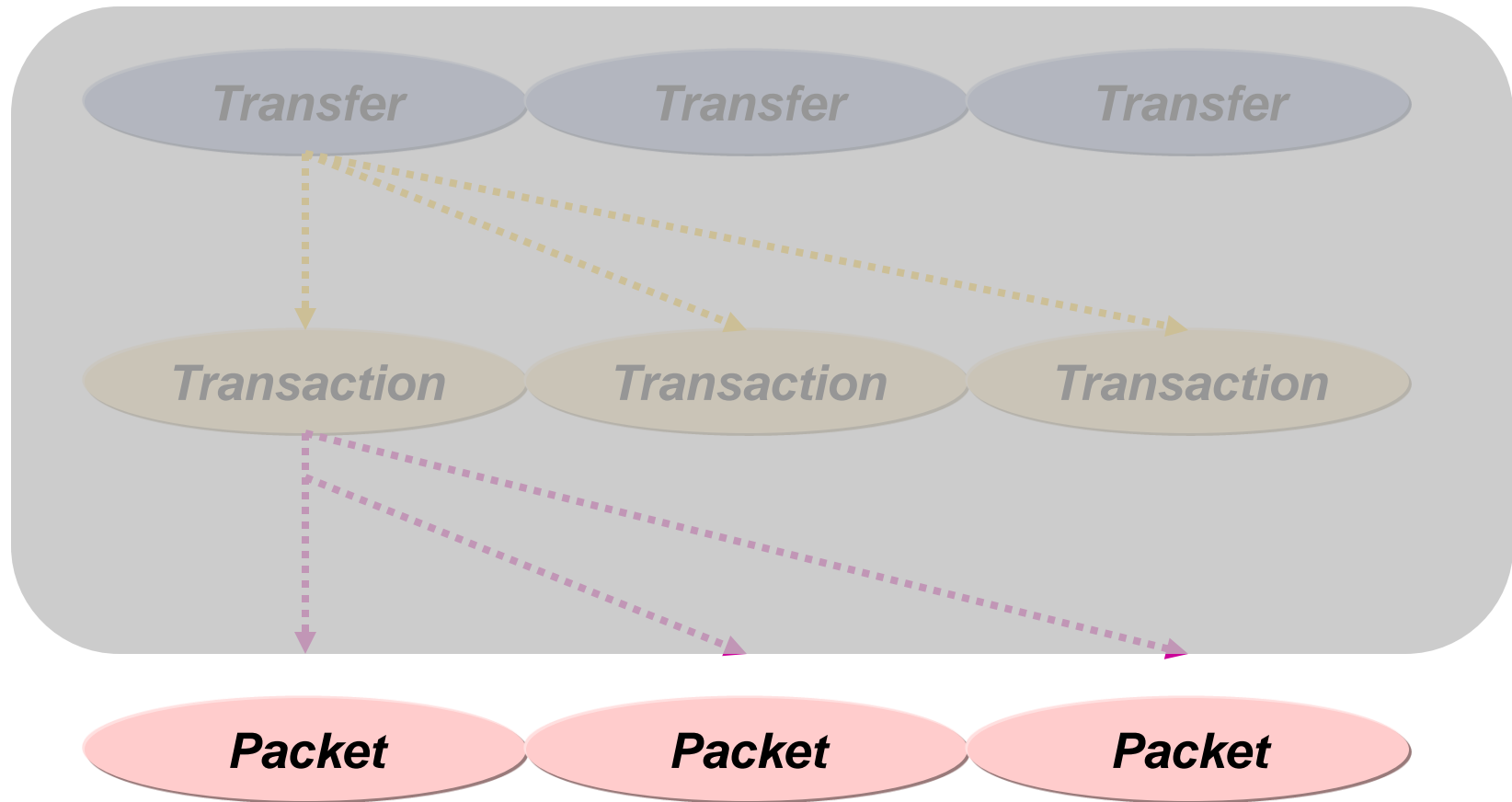
Data Encoding Sequence:



USB传输: Packet、Transaction、Transfer

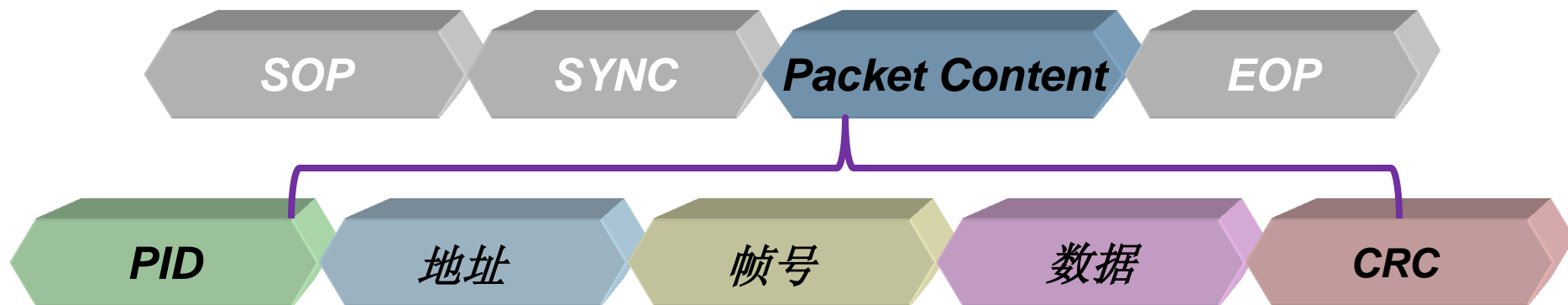


USB传输: Packet、Transaction、Transfer



Packet的组成



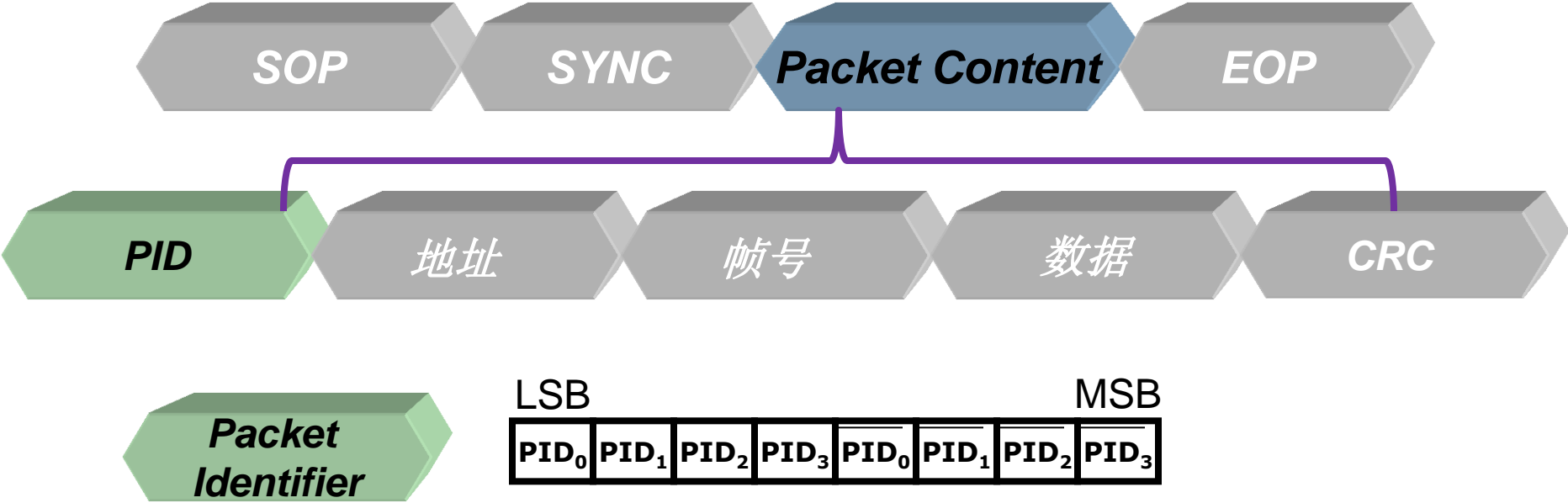


■ Packet分四大类:

- 命令 (Token) Packet
- 帧首 (Start of Frame) Packet
- 数据 (Data) Packet
- 握手 (Handshake) Packet

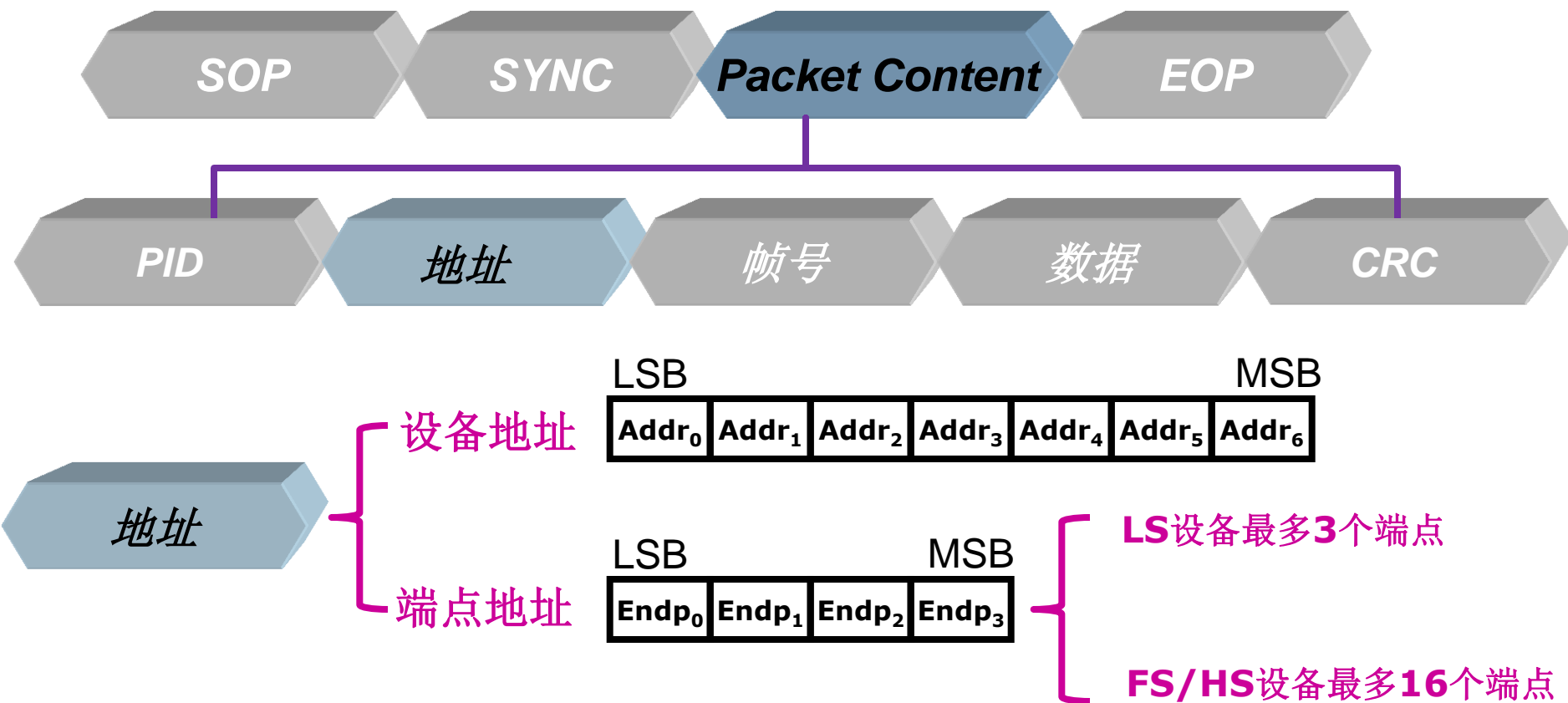
不同类型包，以上的组成部件有所不同

Packet内容之PID域

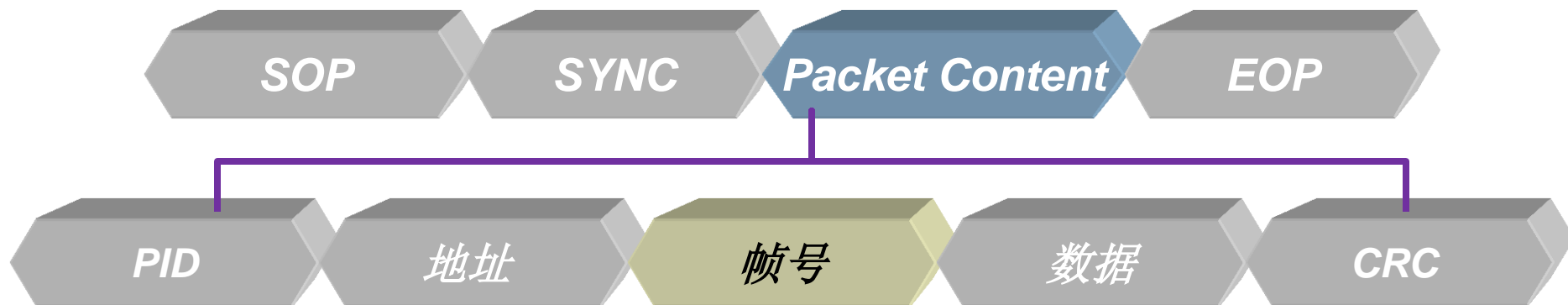


PID 类型	PID 名称	对应packet种类
Token 令牌	OUT /INT /SETUP/ SOF	令牌包、SOF包
Data 数据	DATA0 /DATA1 /DATA2/ MDATA	数据包
Handshake 握手	ACK /NAK /STALL /NYET	握手包
Special 特殊	PRE /ERR /SPLIT /PING	

Packet内容之地址域



Packet内容之帧号域

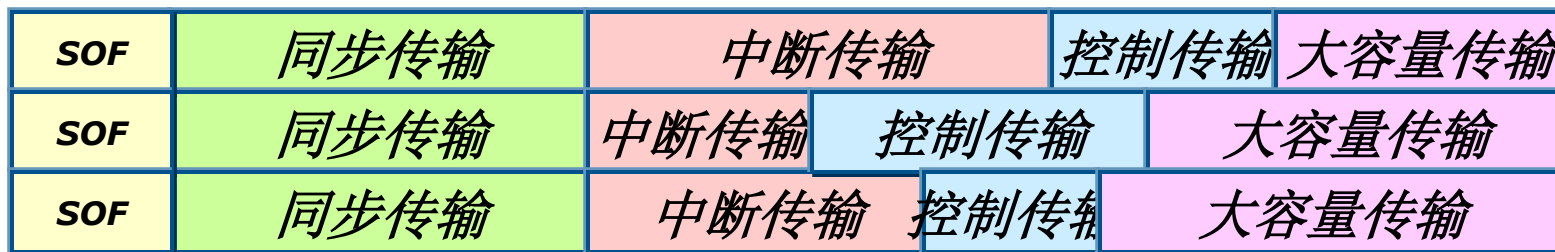


➤ 11 位

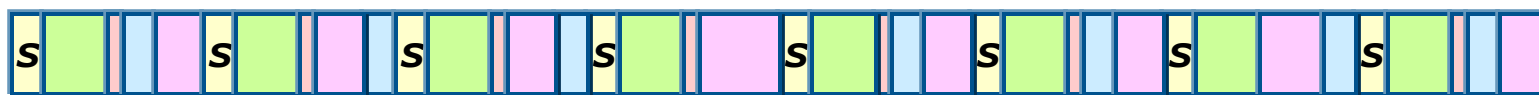
➤ 主机每发出一个帧，帧号都会自加1

➤ 当帧号达到7FFH时，将归零重新开始计数

➤ 仅在每个帧的帧首传输一次SOF包

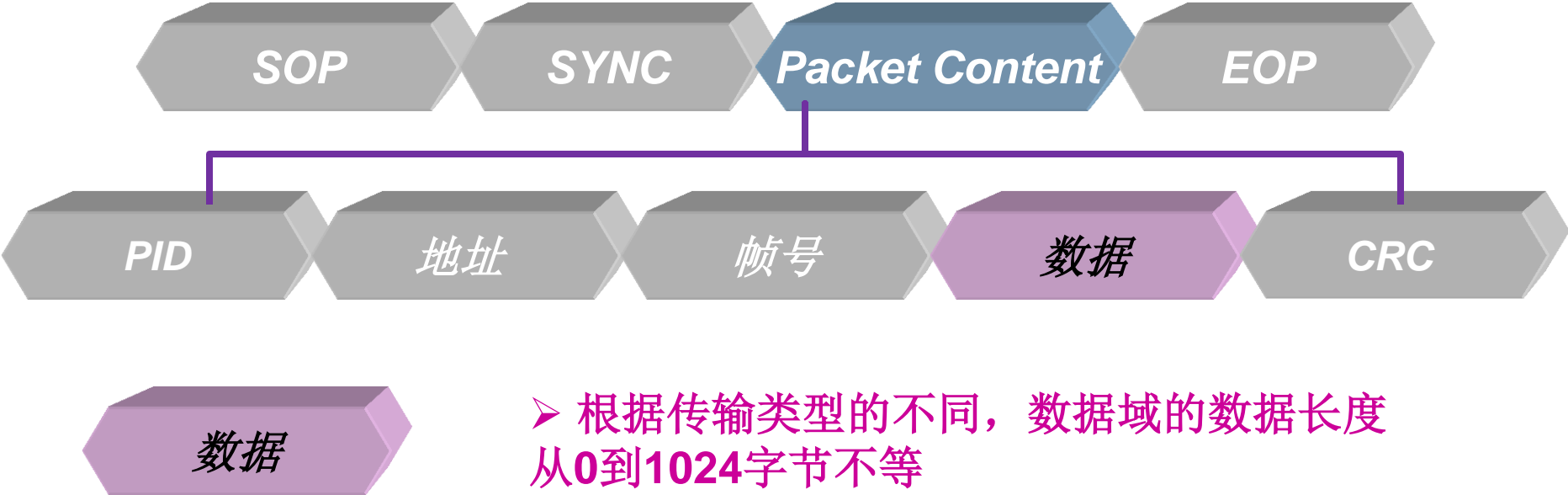


全速设备: 1ms
高速设备: 125us



全速设备: 1ms
高速设备: 125us

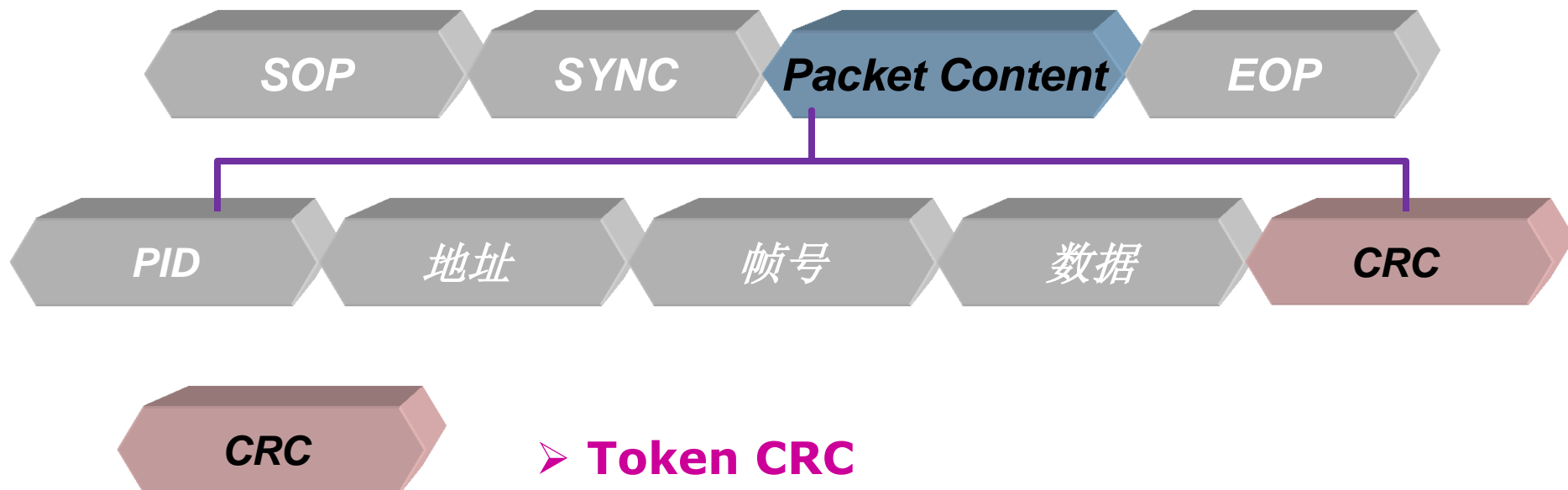
Packet内容之数据域



➤ 根据传输类型的不同，数据域的数据长度从0到1024字节不等

	Control			Bulk			Interrupt			isoch		
	HS	FS	LS	HS	FS	LS	HS	FS	LS	HS	FS	LS
数据包长度	64	64	8	512	64	N.A	1024	64	8	1024	1023	N.A

Packet内容之CRC域



➤ Token CRC

➤ 计算地址域和帧号域的CRC

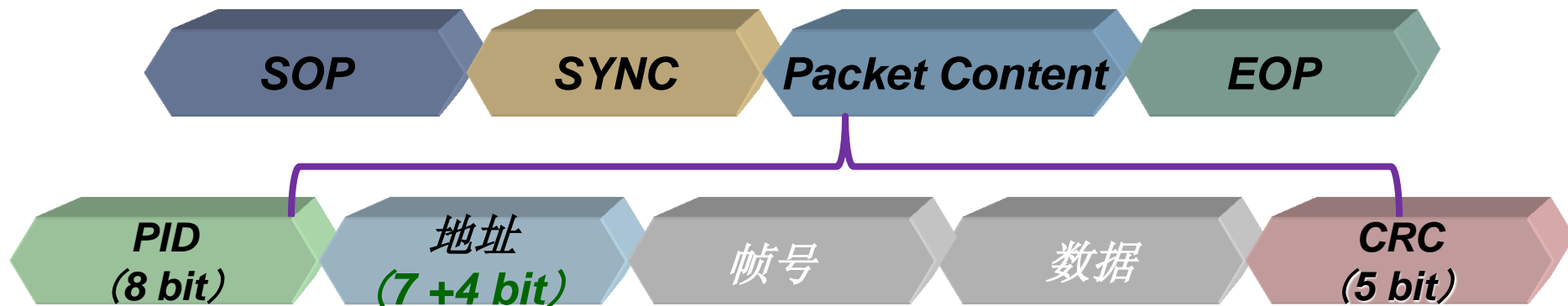
➤ $G(X) = X^5 + X^2 + 1$

➤ Data CRC

➤ 计算数据域数据的CRC

➤ $G(X) = X^{16} + X^{15} + X^2 + 1$

四种Packet类型之Token Packet



以下三种**PID**可选

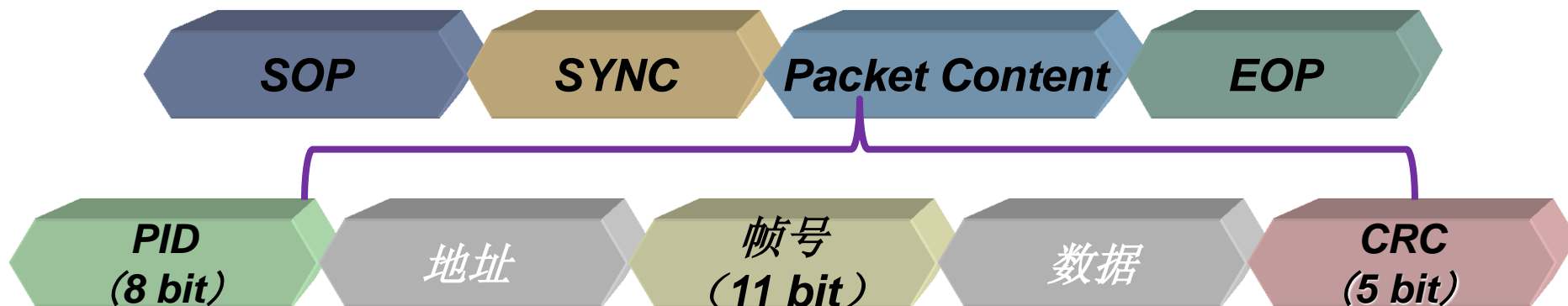
>> **IN**
>> **OUT**
>> **SETUP**

由设备地址
ADDR和端点地
址**ENDP**组成

仅对地址域计算

Sync	SETUP	ADDR	ENDP	CRC5	EOP
00000001	0xB4	3	0	0x0A	250.000 ns

四种Packet类型之SOF Packet



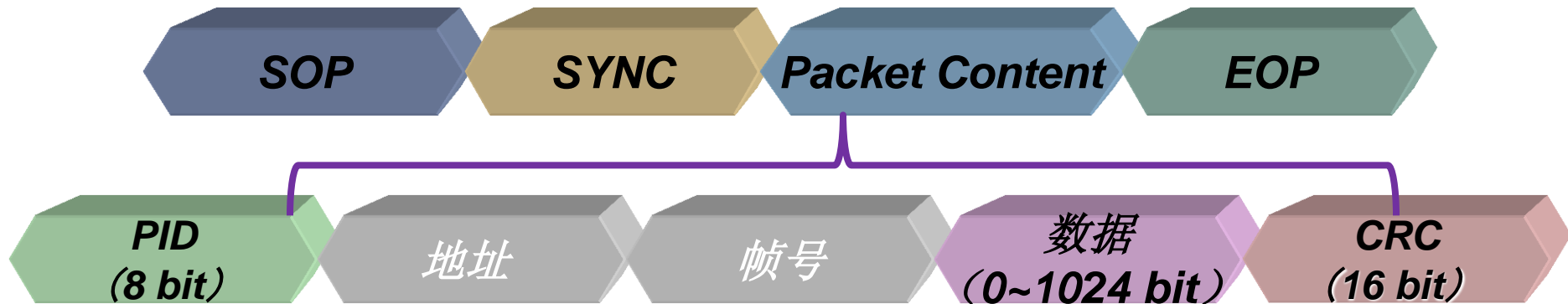
PID = SOF

LS/FS 每1ms一个帧
HS 每125us一个帧

仅对帧号域计算

Sync	SOF	Frame #	CRC5	EOP
00000001	0xA5	1611	0x11	250.000 ns

四种Packet类型之Data Packet



以下四种**PID**可选

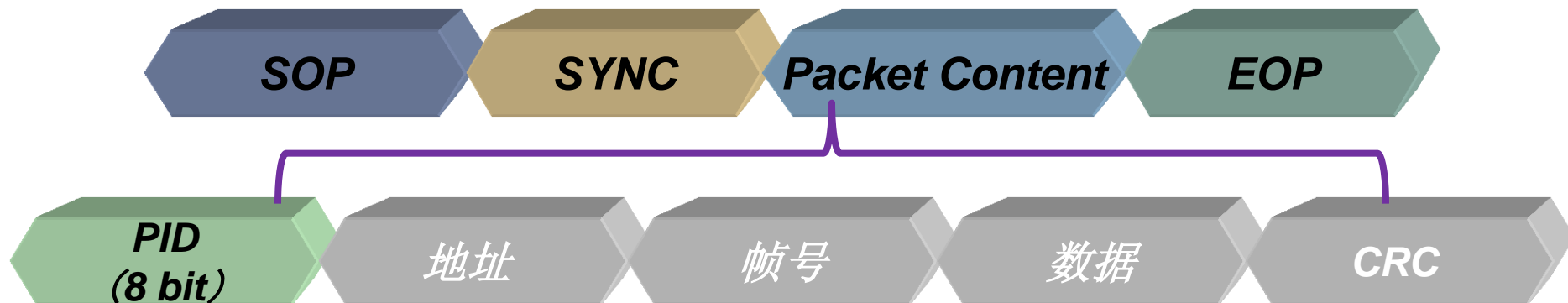
- >> **DATA0**
- >> **DATA1**
- >> **DATA2**
- >> **DATAM**

根据**packet**做在的
transfer类型不同，数
据包最大长度有所不同

仅对数据域计算

Sync	DATA0	Data								CRC16	EOP
00000001	0xC3	80	06	00	01	00	00	12	00	0x072F	250.000 ns

四种Packet类型之Handshake Packet



以下四种**PID**可选

>> **ACK**: 传输正确完成

>> **NAK**: 设备暂时没有准备好接收数据，或没有准备好发送数据

>> **STALL**: 设备不能进行传输

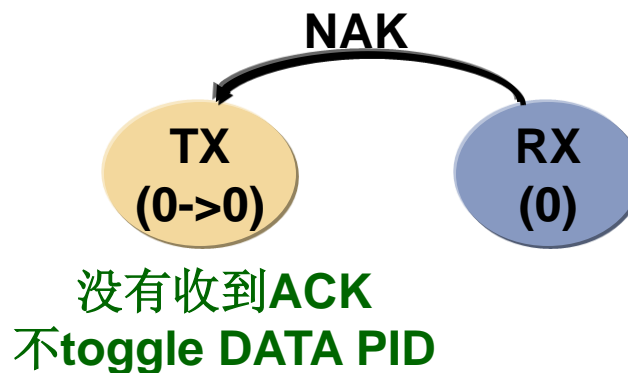
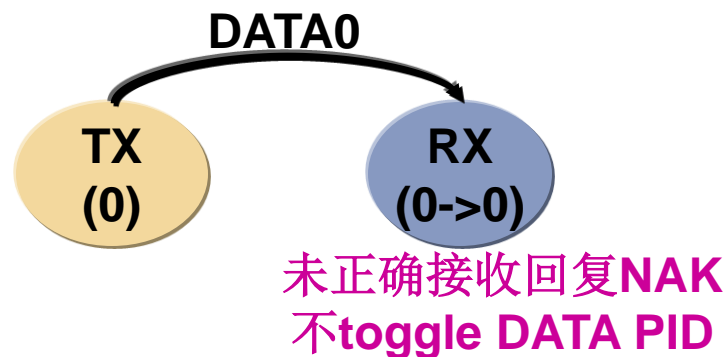
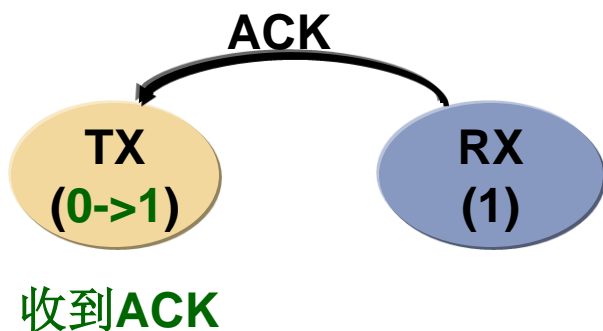
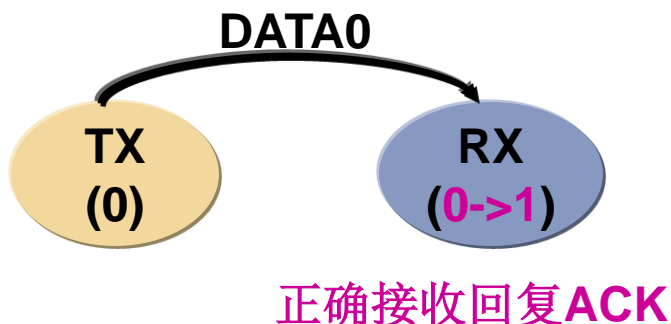
>> **NYET/ERR**: 仅用于高速传输，设备没有准备好或出错

Sync	ACK	EOP
00000001	0x4B	233.330 ns

Data PID toggle用于数据的同步



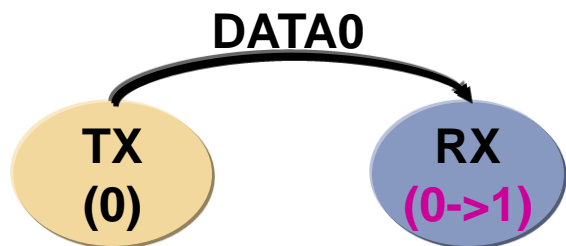
- 正确的数据传输流程
- 当数据被破坏或者没有正确接收



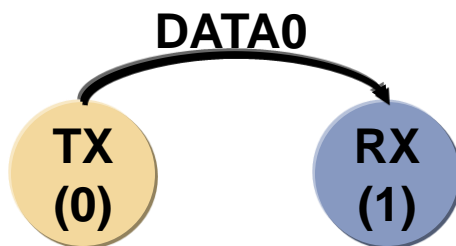
Data PID toggle用于数据的重发



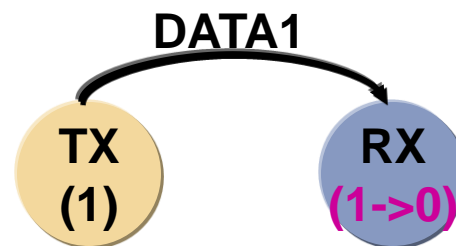
- 当ACK的传输被破坏



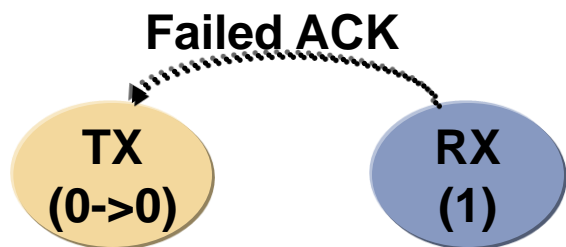
正确接收回复ACK



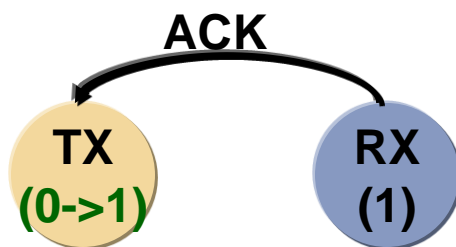
PID对不上回复ACK
但忽略掉数据



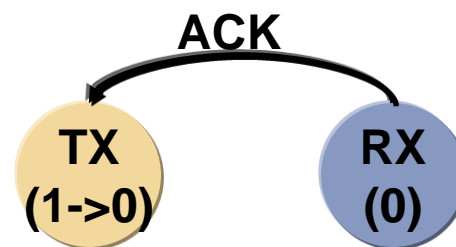
正确接收回复ACK



没有收到ACK
不toggle DATA PID

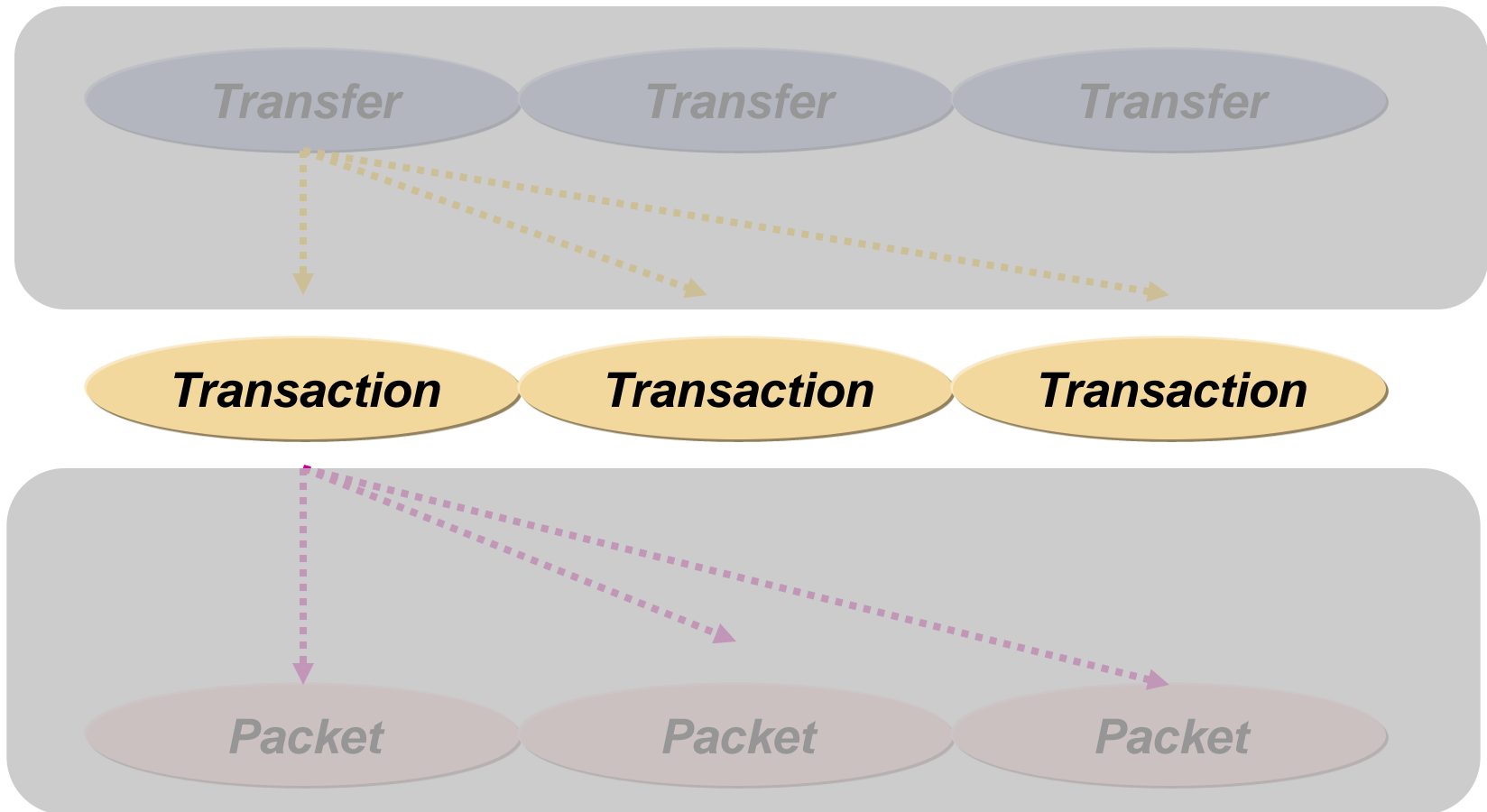


收到ACK



收到ACK

USB传输: Packet、Transaction、Transfer



- Transaction 可以分成三类
 - Setup transaction: 主机用来向设备发送控制命令
 - Data IN transaction: 主机用来从设备读取数据
 - Data OUT transaction: 主机用来向设备发送数据
- Transaction 的 packet 组成
 - Token packet: 总是由主机发出
 - Data packet: 包含此次 transaction 的数据负载
 - 可选的 Handshake packet

总是
由主机
发起

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

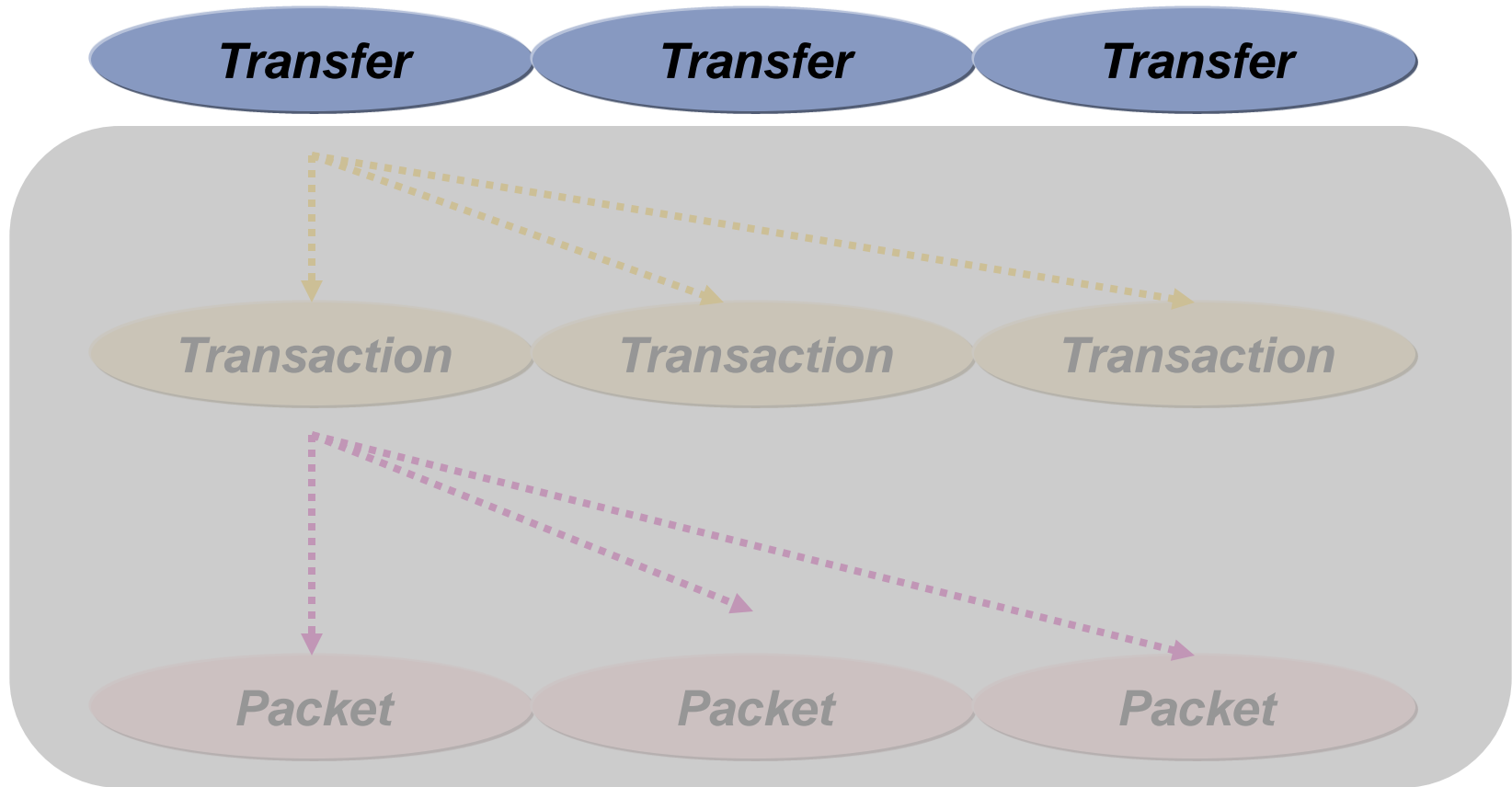
Transaction	L	IN	ADDR	ENDP	T	DATA	ACK
334	S	0x96	1	0	1	12 01 00 02 00 00 00 08	0x4B

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

Transaction	L	IN	ADDR	ENDP	T	DATA	ACK
336	S	0x96	1	0	1	00 01	0x4B

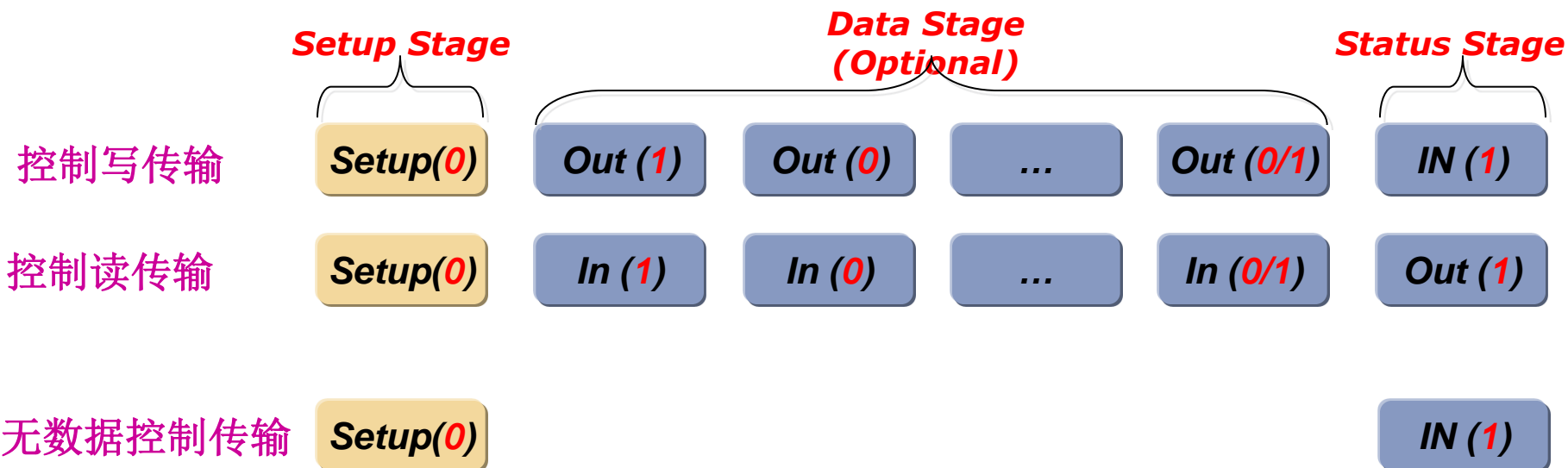
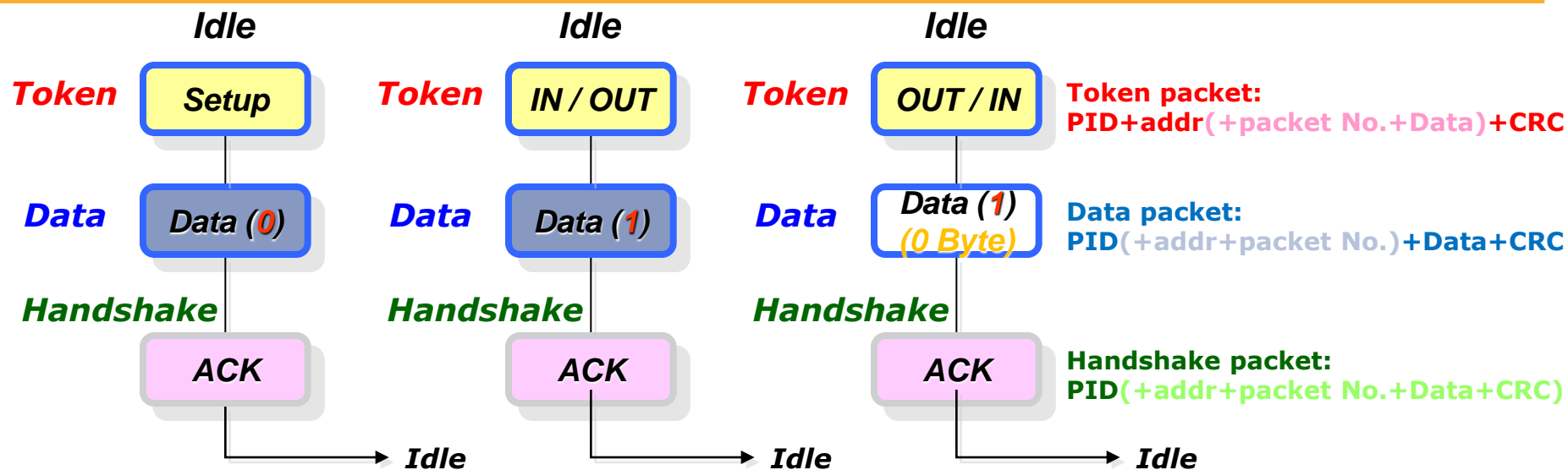
Transaction	L	OUT	ADDR	ENDP	T	DATA	ACK
337	S	0x87	1	0	1		0x4B

USB传输: Packet、Transaction、Transfer



- USB协议定义了四种传输类型：
 - 控制传输(Control Transfers): 非周期性，突发
 - 用于命令和状态的传输
 - 大容量数据传输(Bulk Transfers): 非周期性，突发
 - 大容量数据的通信，数据可以占用任意带宽，并容忍延迟
 - 同步传输(Isochronous Transfers): 周期性
 - 持续性的传输，用于传输与时效相关的信息，并且在数据中保存时间戳的信息
 - 中断传输(Interrupt Transfers): 周期性，低频率
 - 允许有限延迟的通信
- 在每个帧或微帧，主机根据transfer的不同类型要求的带宽安排transfer的传输

控制传输 (1)



控制传输 (2)



Transfer	F	Control	ADDR	ENDP	bRequest	wValue	wIndex	Descriptors	Time Stamp
0	S	GET	0	0	GET_DESCRIPTOR	DEVICE type	0x0000	DEVICE Descriptor	9 . 215 240 050

Transaction	F	SETUP	ADDR	ENDP	T	D	TP	R	bRequest	wValue	wIndex	wLength	ACK	Time Stamp
0	S	0xB4	0	0	0	D->H	S	D	0x06	0x0100	0x0000	8	0x4B	9 . 215 240 050

Packet	H	F	Sync	SETUP	ADDR	ENDP	CRC5	EOP	Idle	Time Stamp
41	↓	S	00000001	0xB4	0	0	0x08	233.330 ns	432.000 ns	9 . 215 240 050

Packet	H	F	Sync	DATA0	Data	CRC16	EOP	Idle	Time Stamp
42	↓	S	00000001	0xC3	8 bytes	0xD729	250.000 ns	218.000 ns	9 . 215 243 382

Packet	↑	F	Sync	ACK	EOP	Time	Time Stamp
43	D	S	00000001	0x4B	250.000 ns	5.973 ms	9 . 215 251 850

Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time Stamp
1	S	0x96	0	0	1	8 bytes	0x4B	9 . 221 225 066

Packet	H	F	Sync	IN	ADDR	ENDP	CRC5	EOP	Idle	Time Stamp
50	↓	S	00000001	0x96	0	0	0x08	233.330 ns	250.000 ns	9 . 221 225 066

Packet	↑	F	Sync	DATA1	Data	CRC16	EOP	Idle	Time Stamp
51	D	S	00000001	0xD2	8 bytes	0x88BE	250.000 ns	350.000 ns	9 . 221 228 216

Packet	H	F	Sync	ACK	EOP	Time	Time Stamp
52	↓	S	00000001	0x4B	250.000 ns	808.316 μs	9 . 221 236 816

Transaction	F	OUT	ADDR	ENDP	T	Data	ACK	Time Stamp
2	S	0x87	0	0	1	0 bytes	0x4B	9 . 222 045 132

Packet	H	F	Sync	OUT	ADDR	ENDP	CRC5	EOP	Idle	Time Stamp
53	↓	S	00000001	0x87	0	0	0x08	233.330 ns	434.000 ns	9 . 222 045 132

Packet	H	F	Sync	DATA1	Data	CRC16	EOP	Idle	Time Stamp
54	↓	S	00000001	0xD2	0 bytes	0x0000	250.000 ns	217.330 ns	9 . 222 048 466

Packet	↑	F	Sync	ACK	EOP	Time	Time Stamp
55	D	S	00000001	0x4B	250.000 ns	5.187 ms	9 . 222 051 600

■ 特性:

- 每个USB设备都必须有控制端点，支持控制传输来进行命令和状态的传输。USB主机驱动将通过控制传输与USB设备的控制端点通信，完成USB设备的枚举和配置

■ 方向:

- 控制传输是双向的传输，必须有IN和OUT两个方向上的特定端点号的控制端点来完成两个方向上的控制传输

数据的拆分和数据传输完毕的判定

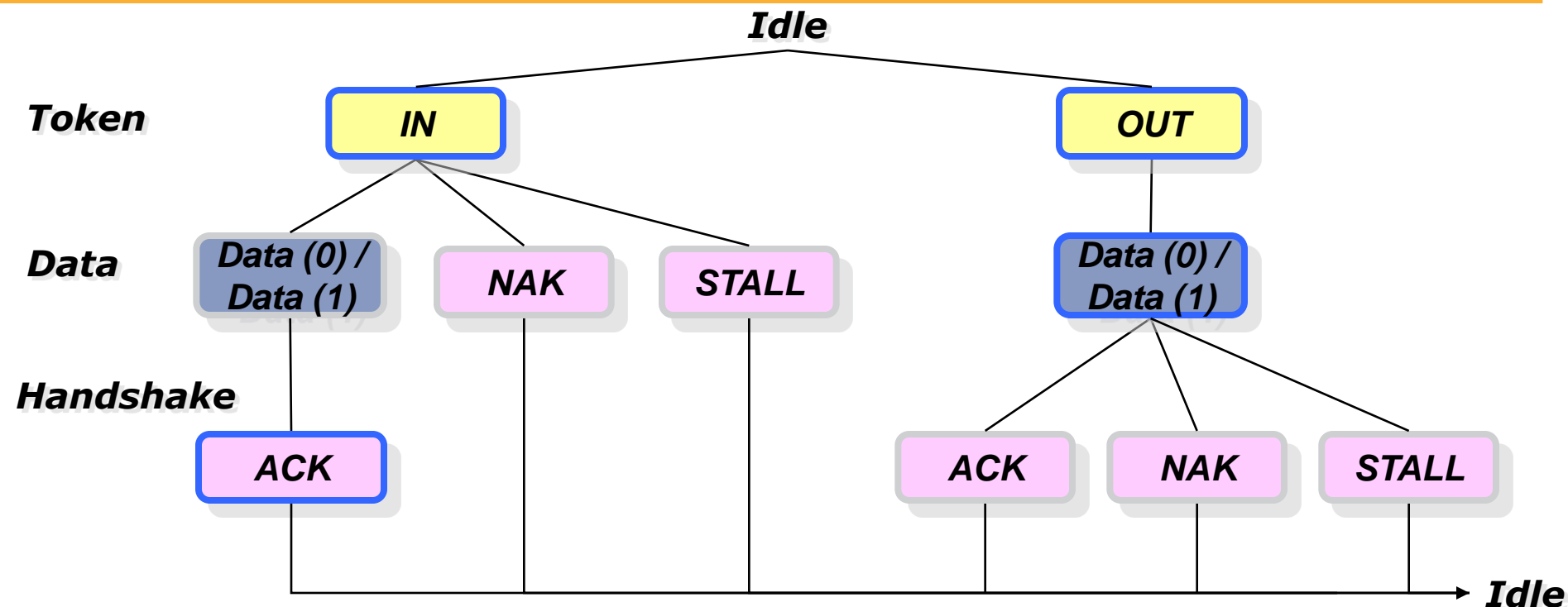
- 以高速设备的最大数据包长度64字节为例
- 要传输250字节，拆分成4个packet

64字节 + **64字节** + **64字节** + **58字节**

- 要传输正好256字节，通过最后一个0字节包告诉设备传输完成

64字节 + **64字节** + **64字节** + **64字节** + **0字节**

中断传输 (1)



中断读传输 **IN (0)** **IN (1)** **IN (0)** **IN (1)** **IN (...)**

中断写传输 **OUT (0)** **OUT (1)** **OUT (0)** **OUT (1)** **OUT (...)**

中断传输 (2)



Transfer	F	Interrupt	ADDR	ENDP	Bytes Transferred	Time Stamp
21	S	IN	5	1	2	00002.1270 0498

Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time Stamp
150	S	0x96	5	1	0	2 bytes	0x4B	00002.1270 0498

Packet	Dir	F	Sync	IN	ADDR	ENDP	CRC5	EOP	Idle	Time Stamp
572	-->	S	00000001	0x96	5	1	0x06	233.330 ns	266.650 ns	00002.1270 0498

Packet	Dir	F	Sync	DATA0	Data	CRC16	EOP	Idle	Time Stamp
573	<--	S	00000001	0xC3	2 bytes	0x3E5A	233.330 ns	600.000 ns	00002.1270 0688

Packet	Dir	F	Sync	ACK	EOP	Time	Time Stamp
574	-->	S	00000001	0x4B	233.330 ns	671.929 ms	00002.1270 0978

Transfer	F	Interrupt	ADDR	ENDP	Bytes Transferred	Time Stamp
29	S	IN	5	1	2	00002.6645 4235

Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time Stamp
339	S	0x96	5	1	1	2 bytes	0x4B	00002.6645 4235

Packet	Dir	F	Sync	IN	ADDR	ENDP	CRC5	EOP	Idle	Time Stamp
1704	-->	S	00000001	0x96	5	1	0x06	233.330 ns	283.320 ns	00002.6645 4235

Packet	Dir	F	Sync	DATA1	Data	CRC16	EOP	Idle	Time Stamp
1705	<--	S	00000001	0xD2	2 bytes	0xBD59	250.000 ns	566.670 ns	00002.6645 4426

Packet	Dir	F	Sync	ACK	EOP	Time	Time Stamp
1706	-->	S	00000001	0x4B	233.330 ns	1.408 sec	00002.6645 4715

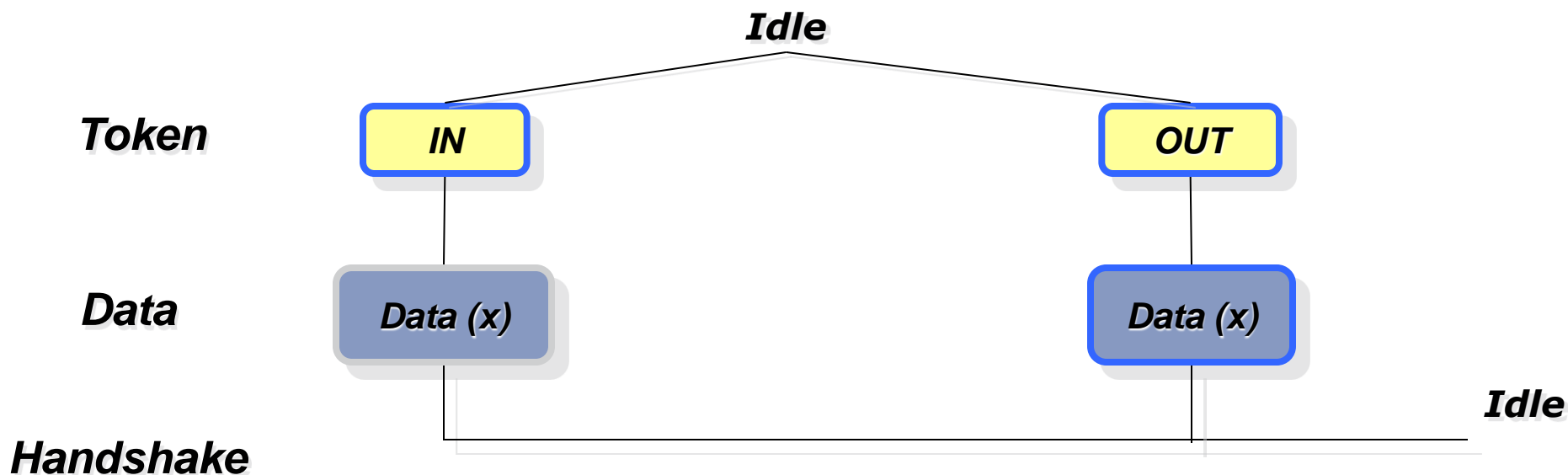
■ 特性:

- 中断传输用于那些频率不高，但对周期有一定要求的数据传输。具有保证的带宽，并能在下个周期对先前错误的传输进行重传。
- 对于全速端点，中断传输的间隔时间在1ms到255ms之间, 对于低速端点，间隔时间限制在10ms到255ms之间, 对于高速端点，间隔为 $2^{bInterval-1} \times 125\mu s$, bInterval的值在1到16之间。

■ 方向:

- 中断传输总是单向的，可以用单向的中断端点来实现某个方向上的中断传输。

同步传输 (1)



同步读传输



同步写传输



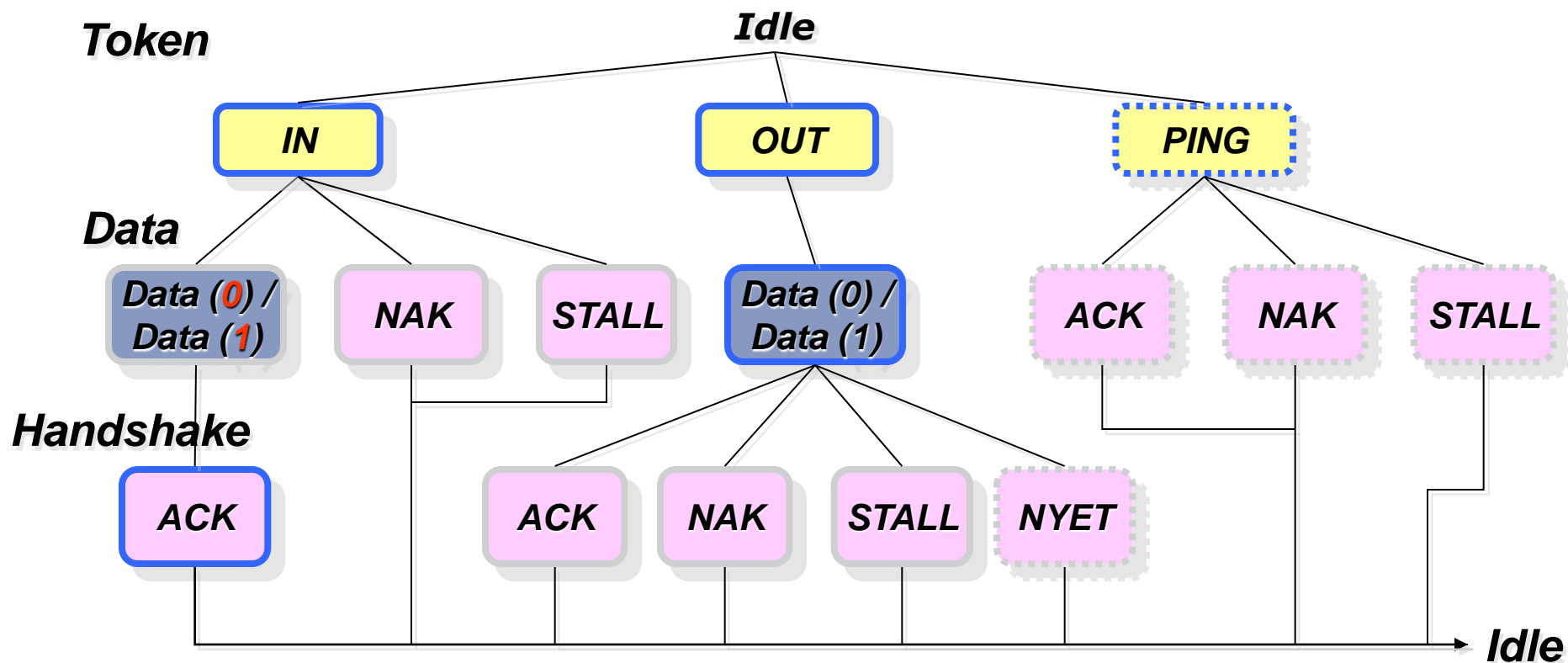
■ 特性:

- 同步传输用于传输那些需要保证带宽，并且不能忍受延迟的信息。整个带宽都将用于保证同步传输的数据完整，并且不支持出错重传

■ 方向:

- 同步传输总是单向的，可以使用单向的同步端点来实现某个方向上的同步传输。

大容量数据传输 (1)



大容量读传输

IN (0)

IN (1)

IN (0)

IN (1)

IN (...)

大容量写传输

OUT (0)

OUT (1)

OUT (0)

OUT (1)

OUT (...)

大容量数据传输 (2)



Transfer	F	Bulk	ADDR	ENDP	Bytes Transferred	Time Stamp
57	S	IN	4	1	36	00002.2702 4672

Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time Stamp
362	S	0x96	4	1	0	36 bytes	0x4B	00002.2702 4672

Packet	Dir	F	Sync	IN	ADDR	ENDP	CRC5	EOP	Idle	Time Stamp
1085	-->	S	00000001	0x96	4	1	0x19	250.000 ns	266.670 ns	00002.2702 4672

Packet	Dir	F	Sync	DATA0	Data	CRC16	EOP	Idle	Time Stamp
1086	<--	S	00000001	0xC3	36 bytes	0x2D13	250.000 ns	566.660 ns	00002.2702 4863

Packet	Dir	F	Sync	ACK	EOP	Time	Time Stamp
1087	-->	S	00000001	0x4B	250.000 ns	917.417 μ s	00002.2702 6512

Transfer	F	Bulk	ADDR	ENDP	Bytes Transferred	Time Stamp
58	S	IN	4	1	13	00002.2710 1557

Transaction	F	IN	ADDR	ENDP	T	Data	ACK	Time Stamp
363	S	0x96	4	1	1	13 bytes	0x4B	00002.2710 1557

Packet	Dir	F	Sync	IN	ADDR	ENDP	CRC5	EOP	Idle	Time Stamp
1089	-->	S	00000001	0x96	4	1	0x19	250.000 ns	249.990 ns	00002.2710 1557

Packet	Dir	F	Sync	DATA1	Data	CRC16	EOP	Idle	Time Stamp
1090	<--	S	00000001	0xD2	13 bytes	0xCB07	233.330 ns	600.000 ns	00002.2710 1747

Packet	Dir	F	Sync	ACK	EOP	Time	Time Stamp
1091	-->	S	00000001	0x4B	250.000 ns	1.019 ms	00002.2710 2477

■ 特性:

- 大容量数据传输适用于那些需要大数据量传输，但是对实时性，对延迟性和带宽没有严格要求的应用。大容量传输可以占用任意可用的数据带宽。

■ 方向:

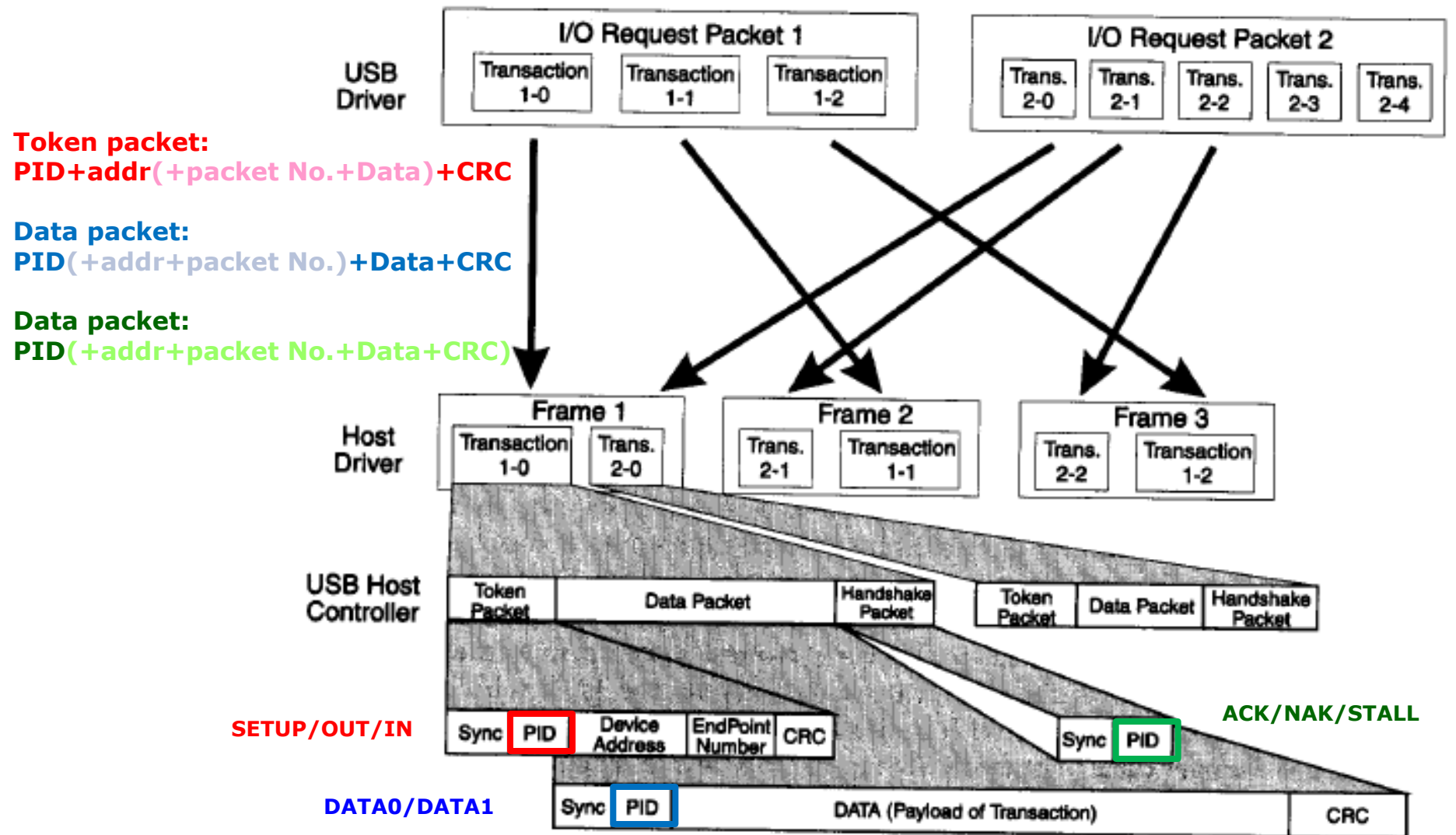
- 大容量传输是单向的，可以用单向的大容量传输端点来实现某个方向的大容量传输。

各种transfer特性比较



	Control			Bulk			Interrupt			isoch		
	HS	FS	LS	HS	FS	LS	HS	FS	LS	HS	FS	LS
带宽	保证			没有保证			有限的保留带宽			保证、传输率固定		
	20%	10%	10%				23.4 M B/s	62.5 K B/s	800 B/s		999 K B/s	2.9M B/s
最大数据包长度	64	64	8	512	64	N.A	1024	64	8	1024	1023	N.A
传输错误管理	握手包、PID翻转			握手包、PID翻转			握手包、PID翻转			无错误纠察		
组成	Setup stage + Optional Data stage + Status stage			一个或多个Data transaction (IN or OUT)			一个Data transaction (IN)			一个或多个Data transaction (IN or OUT)		

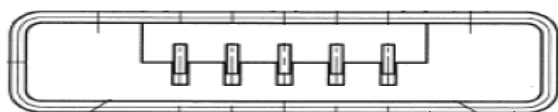
Transfer-Transaction-Packet



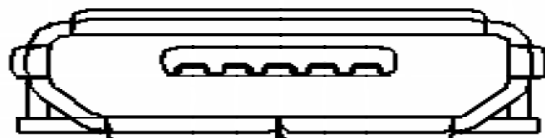
USB OTG: A设备和B设备



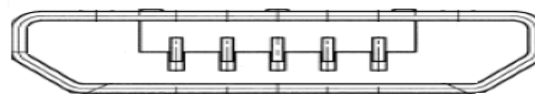
- **OTG 设备:** USB主机 + USB设备的双重角色设备
 - 设备上的**Micro-AB插槽** (**ID线引脚接到电源**)，通过插入的插座类型确定连接时刻该设备的角色
 - **Micro-A插头**，ID线接地→A设备：A设备始终给总线提供电力；它是连接时刻的主机，可通过HNP切换到设备角色，但仍需给总线提供电力
 - **Micro-B插头**，ID线悬空→B设备



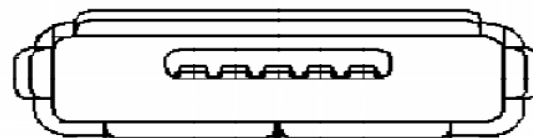
Micro-A 插头



Micro-B 插座

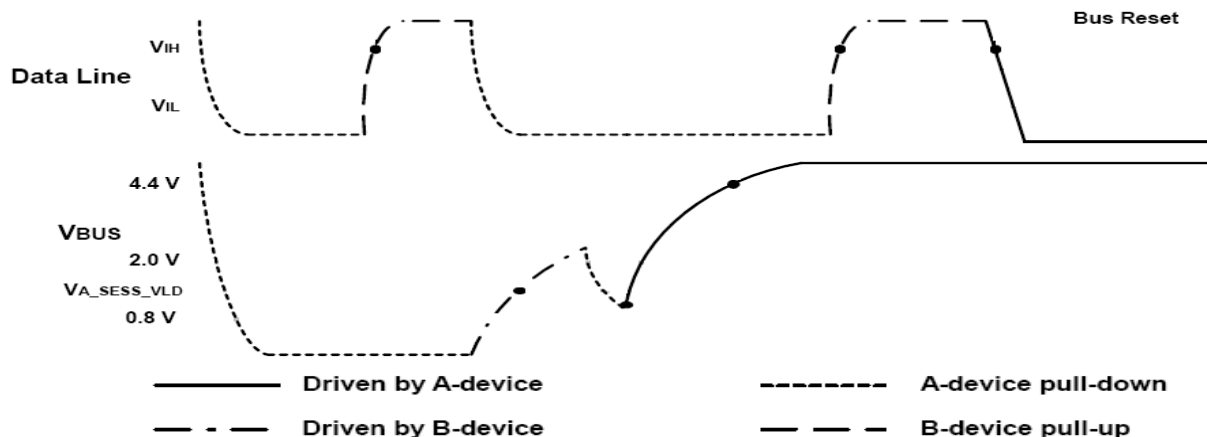


Micro-B 插头

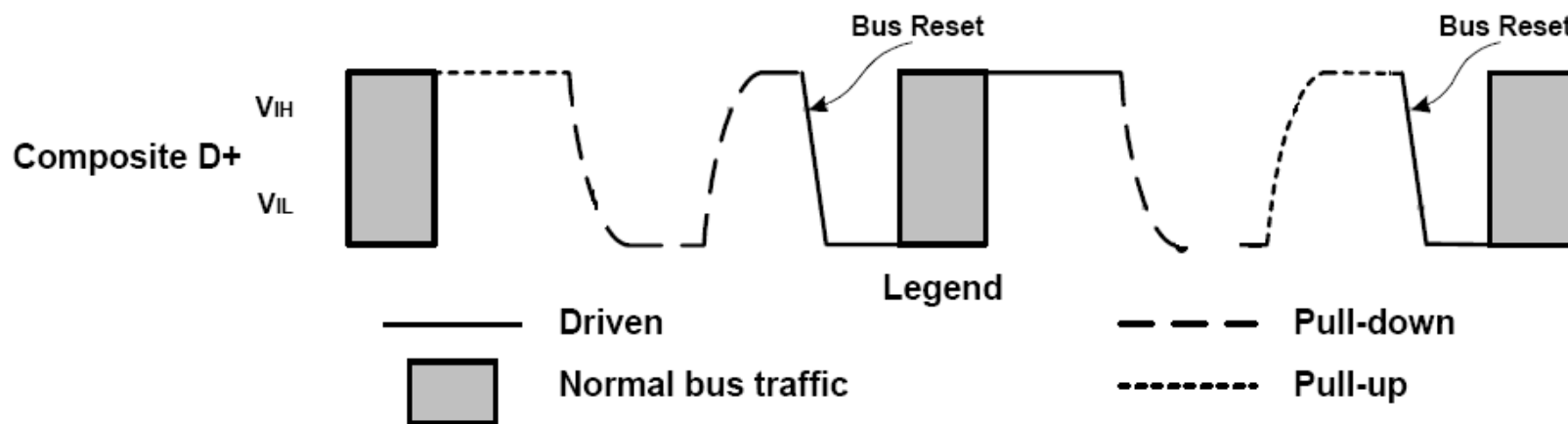


Micro-AB 插座

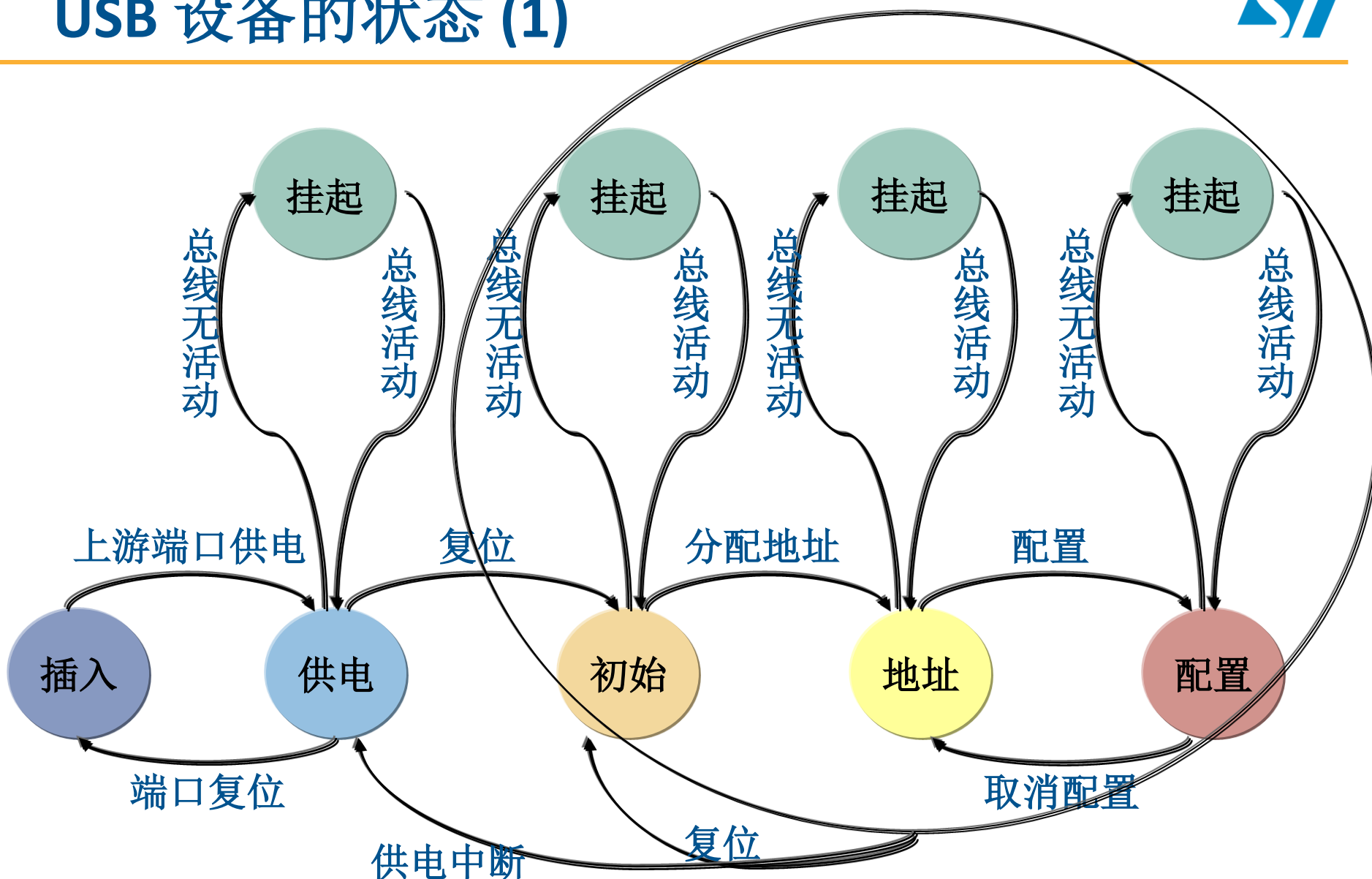
- B设备通过数据线上的脉冲请求A设备打开Vbus并开始会话
 - OTG设备未节省功耗，在总线上没有活动时挂起总线并关闭Vbus
 - 当B设备连接到A设备后，可通过[数据线脉冲](#)或[Vbus脉冲](#)发起SRP通过SRP，请求A设备打开Vbus以开始会话
 - A检测到SRP脉冲后，复位总线、发送SetFeature来判断B设备是否要求执行HNP
 - 是：B设备是OTG两用设备，A设备通过HNP切换角色，由B设备接管总线
 - 否：B设备是外设式OTG设备，A设备使能Vbus并开始通信；当A设备完成传输后，再次关闭Vbus



- OTG设备通过HNP协议切换USB主机和设备角色
 - A设备在完成对B设备的使用后，可通过查询B设备的OTG描述符判断其是否支持HNP；若支持，A设备发送SetFeature命令通知B设备在A设备挂起总线时控制总线，然后A设备挂起总线
 - B设备通过关闭D+上拉来通知A设备断开连接，于是A设备打开D+上拉。于是角色得到切换
 - B设备使用完A设备后，在总线空闲时打开D+上拉要求返回到设备模式，接着A设备关闭D+上拉，然后复位总线



USB 设备的状态 (1)

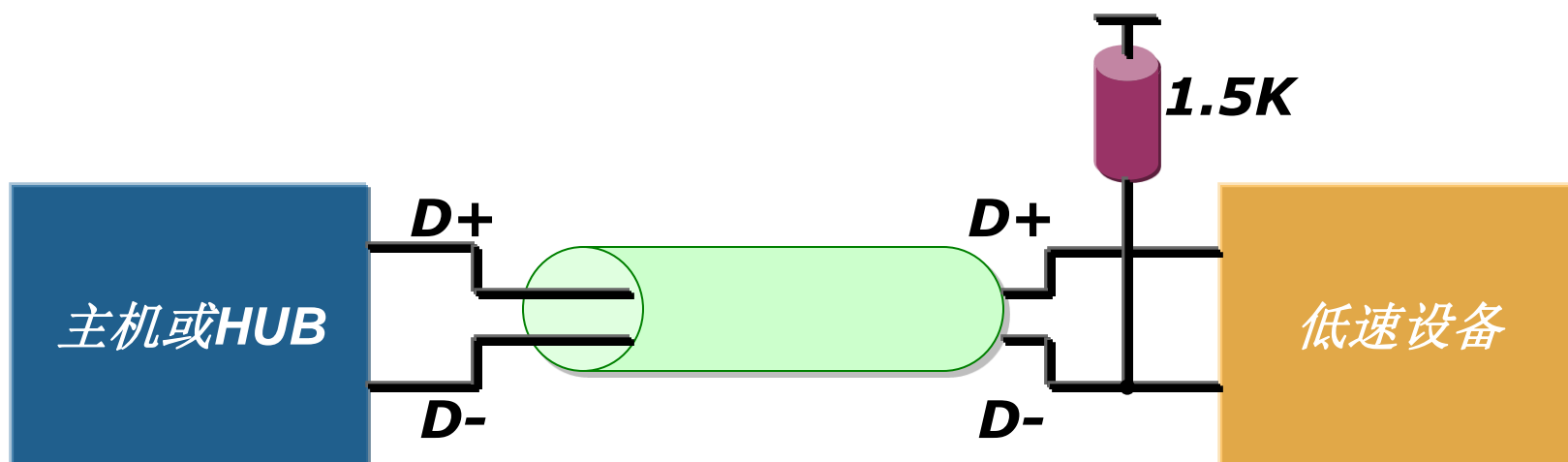
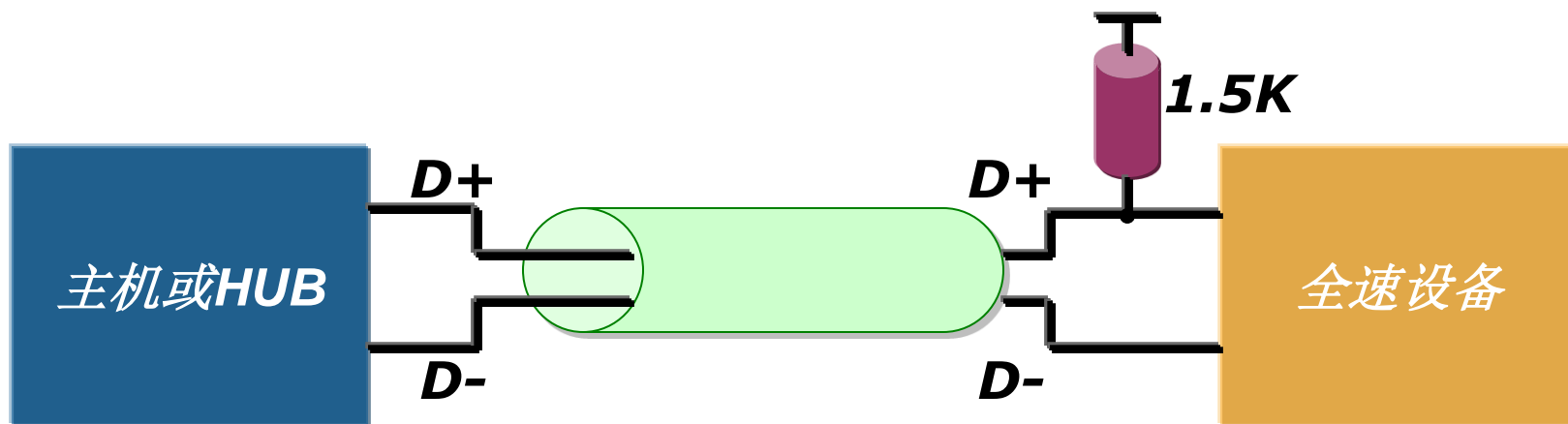


USB 设备的状态 (2)



插入	供电	初始	地址	配置	挂起	状态
No						设备未插入
Yes	No					设备已插入，但未供电
Yes	Yes	No				设备已插入并供电，但未复位
Yes	Yes	Yes	No			设备已插入，供电并复位，但未分配地址
Yes	Yes	Yes	Yes	No		设备已被分配地址，但没有被配置
Yes	Yes	Yes	Yes	Yes	No	设备已被配置，此时可以使用设备的功能
Yes	Yes	Yes	Yes	Yes	Yes	总线无活动超过3ms

USB 全速设备的插入检测



USB 设备枚举



当一个**USB**设备插入主机后，会有以下活动：

供电

复位

获取Device Descriptor前8个字节信息

复位 (可选)

分配地址

获取Device Descriptor

获取Configuration Descriptor

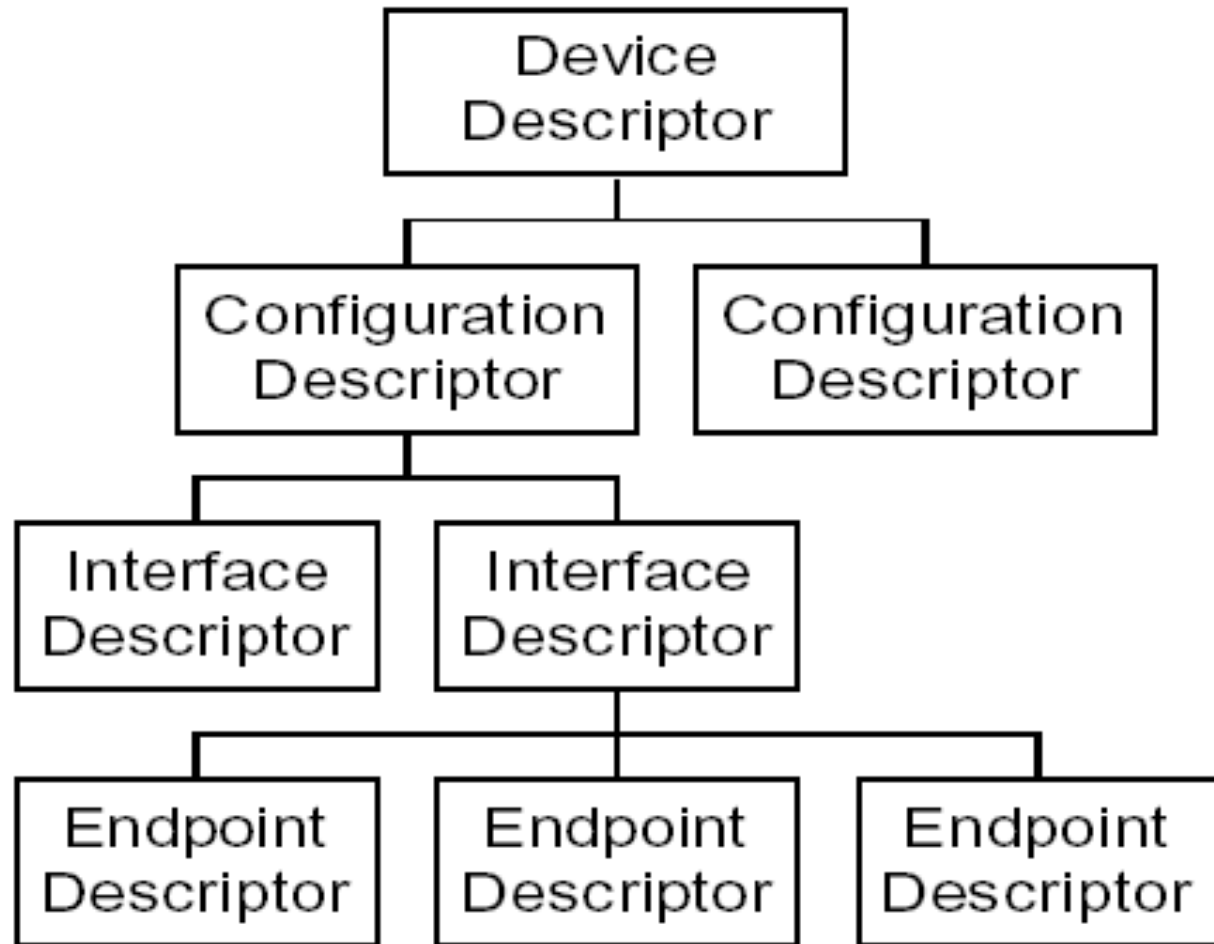
获取String Descriptor (可选)

配置

即插即用的实现：描述符（Descriptor）



比如，一个USB接口的CD-ROM，作为一个设备，它具有两种功能，读取光盘和播放CD，所以有**2个Configuration描述符**。一个功能的实现要涉及许多接口，比如当CD播放机使用时，需要音频接口，同时还需要控制CD机的接口。而控制命令接口又由许多端点组成，比如有的端点负责向CD机发送命令，有的接口负责接收来自CD机的反馈



当设备连上以后.....

- 主机(下称H):你是什么设备?
- 设备(下称D):12 01 0100.....Device Descriptor
- H:你有几种功能?
- D:09 02 09.....Configuration Descriptor
- H:每个功能有几个接口?
- D:09 04 00.....Interface Descriptor
- H:每个接口使用哪几个端点?
- D:06 05 82.....Endpoint Descriptor
- H:好了,我知道你是谁了,开始传输数据吧!
- D:OK, Read Go 😊
 - 描述子的长度
 - 描述子的类型编号

03为String描述符, 一般省略不提供

Descriptor Types	Value
DEVICE	1
CONFIGURATION	2
STRING	3
INTERFACE	4
ENDPOINT	5
DEVICE_QUALIFIER	6
OTHER_SPEED_CONFIGURATION	7
INTERFACE_POWER	8
OTG	9
DEBUG	10
<u>INTERFACE_ASSOCIATION</u>	<u>11</u>

设备描述符 (Device Descriptor)



Off set	Field	Size	Value	描述	备注
0	bLength	1	数字	该描述子包含的字节数目	一般都是 18
1	bDescriptorType	1	常数	【设备描述符的类型常数】	1
2	bcdUSB	2	BCD	该设备符合遵循的USB协议版本号	
4	bDeviceClass	1	Class	由USB-IF分配的	一般都是0
5	bDeviceSubClass	1	SubClass		
6	bDeviceProtocol	1	Protocol		
7	bMaxPacketSize0	1	数字	EP0的最大包长	HS设备必须是64
8	idVendor	2	ID	USB-IF分配的	0x0483
10	iProduct	2	ID	厂家分配的	0x5710
12	bcdDevice	2	BCD	该设备自身的版本号	
14	iManufacturer	1	Index	各自在众多字符串描述符中的序号	
15	iProduct	1	Index		
16	iSerialNumber	1	index		
17	bNumConfigurations	1	数字	该设备在当前速度下支持多少种 Configuration	

以Joystick为例



Joystick_DeviceDescriptor[] =

```
{
0x12,          /* 整个Descriptor的长度: 18字节 */
0x01,          /* Descriptor的类别: Device Descriptor(0x01) */
0x00, 0x02,    /* 设备所遵循的USB协议的版本号: 2.00 */
0x00,          /* 设备所实现的类: 由每个接口描述符描述所实现的类*/
0x00,          /* 设备所实现的子类: 由每个接口描述符描述 */
0x00,          /* 设备所遵循的协议类别: 由每个接口描述符描述*/
0x40,          /* 端点0的最大数据包长度: 64字节*/
0x83, 0x04,    /* IDVendor: 0x0483 (for ST) */
0x10, 0x57,    /* IDProduct: 0x5710 */
0x00, 0x02,    /* bcdDevice: 2.00*/
1,             /* 用于描述生产厂商的字符描述符的索引号 */
2,             /* 用于描述产品的字符描述符的索引号*/
3,             /* 用于描述产品系列号的字符描述符的索引号*/
0x01          /* 设备所支持的配置数目: 1*/
}
```

配置描述符（Configuration Descriptor）



Off set	Field	Size	Value	描述	备注
0	bLength	1	数字	该描述子包含的字节数目	一般都是9
1	bDescriptorType	1	常数	【配置描述符的类型常数】	2
2	wTotalLength	2	数字	包含了所有描述子的长度（配置、接口、端点、类相关、厂商相关）	X: 当主机要求配置描述符时，所有相关的接口、端点描述符都要返回
4	bNumInterfaces	1	数字	该配置包含几个接口	
5	bConfigurationValue	1	数字	选择该配置时，传递给SetConfiguration()的参数	
6	iConfiguration	1	index	在众多字符串描述符中的序号	
7	bmAttributes	1	位图	D7: 保留为1 D6: 是否由总线供电 D5: 是否支持远程唤醒 D4~D0: 保留为0	鼠标通常是0xE0
8	bMaxPower	1	mA	该设备在该configuration下全速工作时需要从总线获取的电流（以2mA为单位）	

Joystick_ConfigurationDescriptor[] =

```
0x09,          /* 描述符的长度: 9字节 */
0x02,          /* 描述符的类型: 0x02 配置描述符 */
JOYSTICK_SIZ_CONFIG_DESC, 0x00, /* 完整的描述符包括接口描述符、
                                端点描述符和类描述符的长度 */
0x01,          /* 配置所支持的接口数目: 1 */
0x01,          /* 用SetConfiguration选择此配置时所指定的配置号 */
0x00,          /* 用于描述此配置的字符描述符的索引号: 0 */
0xE0,          /* 供电配置: B7(1 保留), B6(自供电), B5(远程唤醒), B4-
0x32,          /* 最大功耗, 以2mA为单位计算: 0x32表示 100mA */
```

接口描述符（Interface Descriptor）



Off set	Field	Size	Value	描述	备注
0	bLength	1	数字	该描述子包含的字节数目	一般都是9
1	bDescriptorType	1	常数	【接口描述符的类型常数】	4
2	bInterfaceNumber	1	数字	该接口的序号：在该配置所支持的众多接口中的序号	从0开始
3	bAlternateSetting	1	数字	用来选择该接口的某个setting	
4	bNumEndpoints	1	数字	该接口支持多少个非0的EP	如果此项为0则该接口使用默认的控制通道
5	bInterfaceClass	1	Class	由USB-IF分配	*
6	bInterfaceSubClass	1	SubClass		**
7	bInterfaceProtocol	1	Protocol		
8	iInterface	1	index	在众多字符串描述符中的序号	

* 常用设备类Class分配：AUDIO=1；COMMUNICATION=2；HID=3；MSC=8

** 取决于Class，有各自的定义：

Class = 3 (HID)的情况下，SubClass = 1 (BOOT) = 0 (no-BOOT) Protocol = 1 (KeyBoard) = 0 (Mouse)

1表示是个启动设备，一般对PC才有意义，表示bios启动时能识别并使用该设备；否则只能在OS启动后才能用

以Joystick为例



Joystick_InterfaceDescriptor[] =

```
0x09,      /* 描述符的长度：9字节 */
4,         /* 描述符的类型：0x04接口描述符(Interface) */
0x00,      /* 选择此接口的索引号，从0开始计算：0 */
0x00,      /* 用于选择此设置的索引号：0 */
0x01,      /* 实现此接口需要使用的端点数目：1 */
0x03,      /* 此接口所遵循的类：HID Class */
0x01,      /* 此接口所遵循的子类：1=BOOT, 0=no boot */
0x02,      /* 此接口所支持的协议：0自定义、1键盘、2鼠标 */
0,         /* 用于描述此接口的字符描述符的索引号 */
```

端点描述符 (Endpoint Descriptor)



Off set	Field	Size	Value	描述	备注
0	bLength	1	数字	该描述子包含的字节数目	一般都是7
1	bDescriptorType	1	常数	【端点描述符的类型常数】	5
2	bEndpointAddress	1	端点	Bit7: 方向 (0-OUT; 1-IN) Bit6~4: 保留为0 Bit3~0: 端点号	
3	bmAttributes	1	位图	Bit1~0: 传输类型 (CTRL、ISO、BULK、INTR) Bit3~2: 同步类型 () Bit5~4: 使用类型 (数据EP、反馈EP、非显式的反馈数据EP、...)	
4	wMaxPacketSize	2	数字	该端点的最大包长	
6	bInterval	1	数字	主机查询EP的间隔，以帧或微帧为单位 FS/HS的ISO EP: 取值1~16，间隔是 $2^{(bInterval-1)}$ FS/LS的INTR EP: 取值1~255 HS的INTR EP: 取值在1~16，间隔是 $2^{(bInterval-1)}$ HS的BULK/CTRL OUT EP: 设定了设备的最大NAK率（即每个微帧最多回几个NAK）；取值0~255	

以Joystick为例



Joystick_InterfaceDescriptor[] =

```
0x07,      /* 描述符长度：7字节 */
5,         /* 描述符类型：端点描述符*/
0x81,      /* 端点的特性： B3-B0(端点号), B6-B4(0), B7(1=IN, 0=OUT)
           0x81: Endpoint1/ IN */
0x03,      /* 端点的类型： B1-B0(00=控制 01=同步 10=大容量 11=中断)
           0x03: 中断端点 */
0x04, 0x00 /* 此端点的最大有效数据长度：4 字节 */
0x20,      /* 主机查询此端点数据的间隔时间：(1ms或125us单位):
           0x20: 32 ms */
```

字符串描述符 (String Descriptor)



Get String Descriptor (value= type3, index = 0)

Off set	Field	Size	Value	描述	备注
0	bLength	1	N+2	该描述子包含的字节数目	
1	bDescriptorType	1	常数	【字符串描述符的类型常数】	3
2	wLANGID[0]	2	数字	语言编号编码0	
...	
N	wLANGID[x]	2	数字	语言编号编码x	

Get String Descriptor (value= type3, index = 刚才得到的LANGID)

Off set	Field	Size	Value	描述	备注
0	bLength	1	数字	该描述子包含的字节数目	
1	bDescriptorType	1	常数	【字符串描述符的类型常数】	
2	bString	N	数字	字符串的UNICODE编码	


```
Joystick_StringVendor[ ] =  
{  
    JOYSTICK_SIZ_STRING_VENDOR,          /* 描述符的长度 */  
    USB_STRING_DESCRIPTOR_TYPE,          /* 描述符的类型：字符描述符 */  
    'S', 0, 'T', 0, 'M', 0, 'i', 0, 'c', 0, 'r', 0, 'o', 0, 'e', 0,  
    'l', 0, 'e', 0, 'c', 0, 't', 0, 'r', 0, 'o', 0, 'n', 0, 'i', 0,  
    'c', 0, 's', 0                        /* 描述符所描述的内容： "STMicroelectronics" */  
}
```

以Joystick为例



```
Joystick_ConfigDescriptor[ ] =  
{  
    配置描述符: Configuration Descriptor  
  
    +  
  
    接口描述符: Interface Descriptor  
  
    +  
  
    类描述符: Class Descriptor  
  
    +  
  
    端点描述符: Endpoint Descriptor  
}
```

- USB家族中有三位:主机、设备与Hub, 目前常见的是主机与设备, 或主机经Hub与设备的连接方式。
- On The Go可实现设备直接对连, 比如数码相机可直接连到打印机上打印图片
- USB通讯过程中, 以众多设备共享间隔为1s的帧结构来实现数据通讯