C11L14 USB

Microchip USB解决方案

课程目标

- 完成本课程后，您将能够：
  - 描述USB基础以及如何在嵌入式应用中加以运用
  - 识别与您项目有关的Microchip USB单片机、开发板和USB软件框架
  - 产生（Trap）并使用WM_DEVICECHANGE事件
  - 充分定制VB.Net USB Communicator应用

课程安排

- **USB基础 —— 严肃且重要的资料**
  - USB的简史
  - 基础/速度
  - 拓扑/物理连接
  - 架构/程序员模型
  - USB事务
  - USB传输
  - 设备类
  - 枚举
  - 描述符
  - VID/PID及USB兼容性
- **PIC18/24/32 USB单片机**
- **Microchip演示/开发解决方案**

课程安排

- 双向USB通信工作原理
- VB.Net USB类及示例应用
  - 函数释义
  - 单点及多点获取
  - 连续数据获取
  - USB描述符获取
    - 检索设备制造商序列号
    - 检索设备产品字符串
    - 检索设备制造商字符串
  - WM_DEVICECHANGE事件（设备枚举）
  - 通过窗体实例进行跟踪
  - 通过序列号进行跟踪
- Microchip演示/开发解决方案
- 双向USB通信工作原理

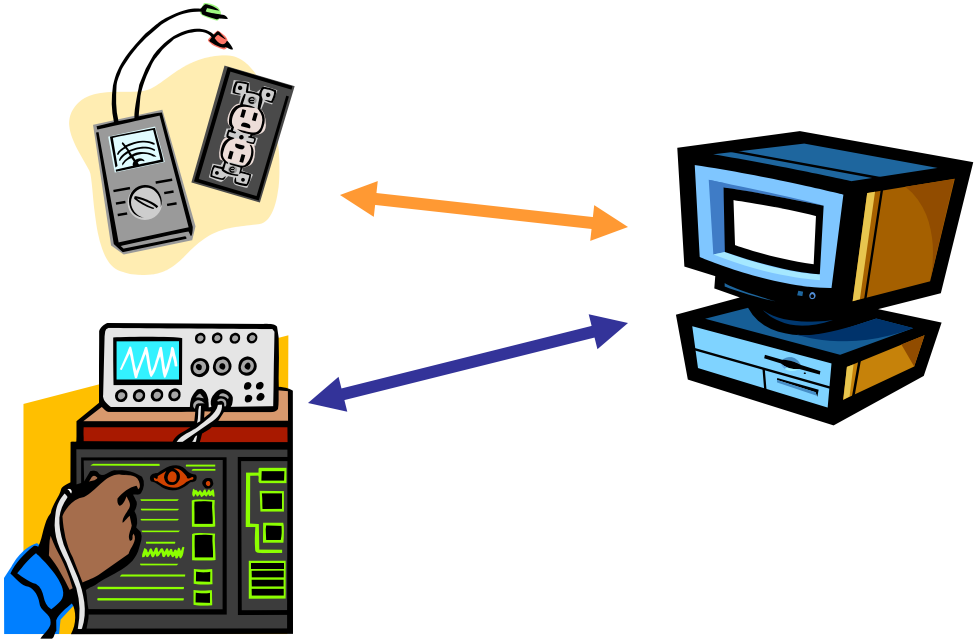
课程安排

- **USB基础 —— 严肃且重要的资料**
  - USB的简史
  - 基础/速度
  - 拓扑/物理连接
  - 拓扑/物理连接
  - 架构/程序员模型
  - USB事务
  - USB传输
  - 设备类
  - 枚举
  - 描述符
  - VID/PID及USB兼容性
- **PIC18/24/32 USB单片机**
- **Microchip演示/开发解决方案**
- **双向USB通信工作原理**

简史...

- **USB由数个公司共同开发...**
  - Compaq
  - Intel
  - Microsoft
  - …那些想使得向PC添加/移除外设更为容易的公司
- **1998 – USB 1.1**
- **2000 – USB 2.0**
- **2003 – 对USB 2.0进行了On-the-Go补充**
- **2008 – USB 3.0（详见附录）**

通用串行总线2.0



扩展计算机的功能！

数据分析，
数据记录，
固件更新，
诊断，
嵌入式应用！

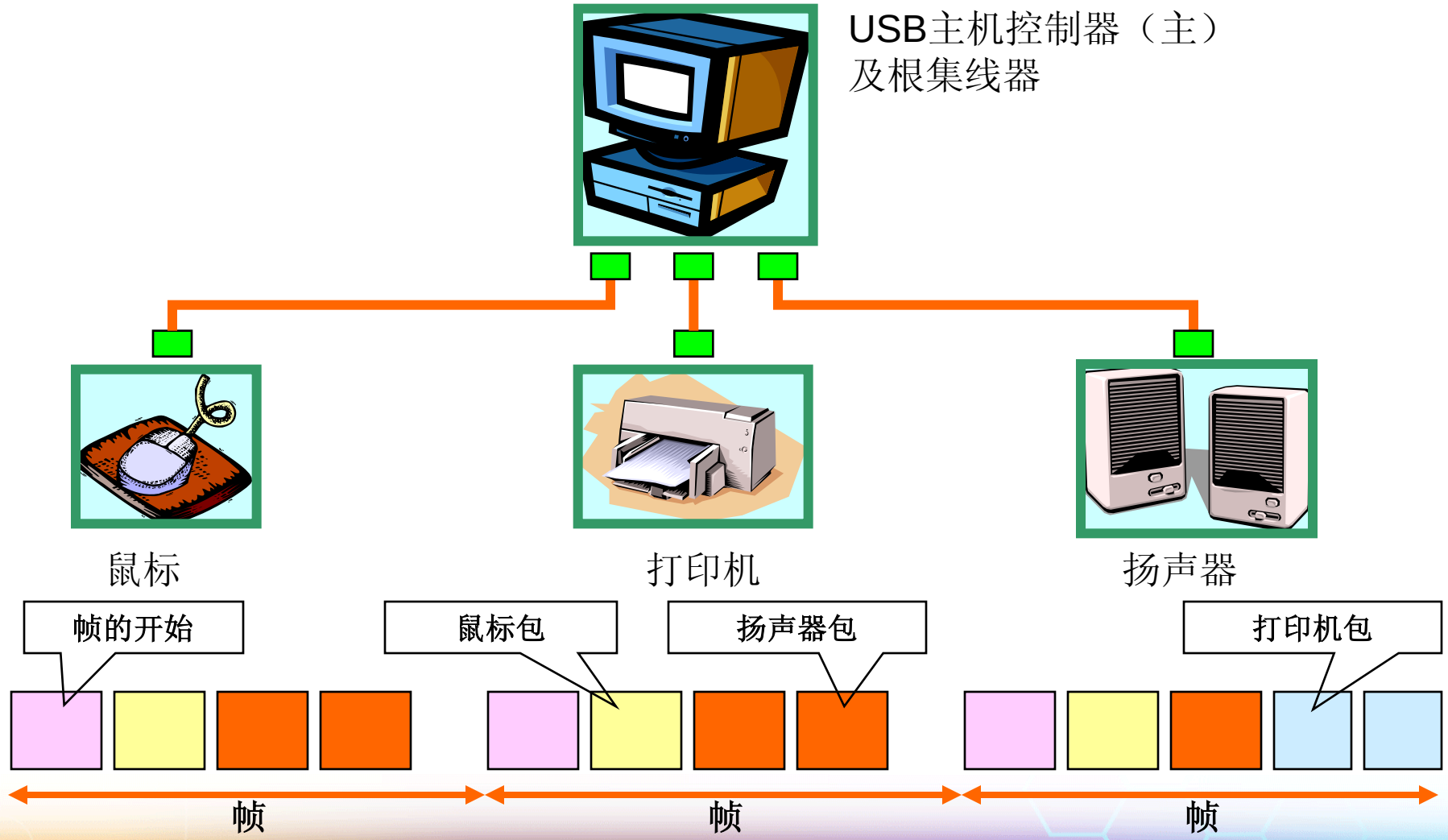
- 自动检测与配置（即插即用）
- 使用集线器可轻松扩展
- 总线供电
- 数据受CRC保护，重新发送坏包
- 三种速度：
  低速 — 1.5，全速 — 12，高速 — 480 Mbps

课程安排

- 嵌入式应用设计人员面临的连通性挑战
- **USB基础 —— 严肃且重要的资料**
  - USB的简史
  - 基础/速度
  - 拓扑/物理连接
  - 架构/程序员模型
  - USB事务
  - USB传输
  - 设备类
  - 枚举
  - 描述符
  - VID/PID及USB兼容性
- **PIC18/24/32 USB单片机**
- **Microchip演示/开发解决方案**
- **如何着手？**

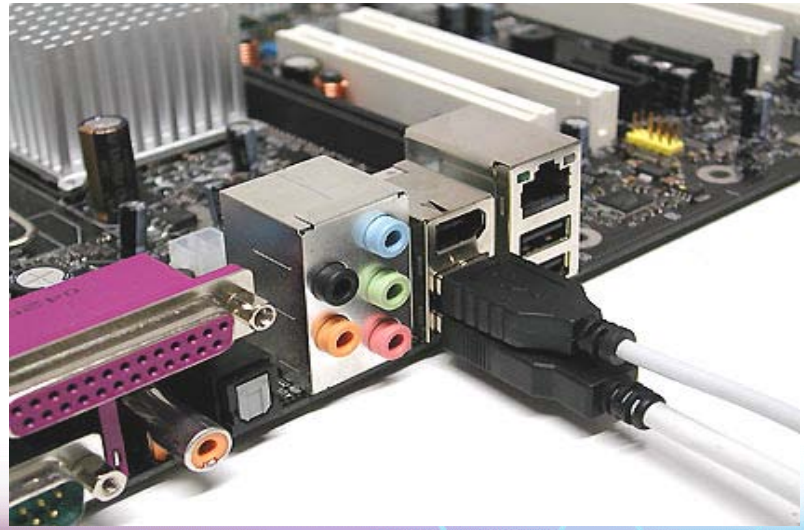
USB基础

USB是一种“单主 + 多从”轮询式总线



USB主机

- 管理并控制总线
- 发起所有的通信
- 自动检测所有设备的插入和拔出
- 枚举所有连接的设备，并把他们与驱动程序相匹配



主机的典型MCU需求

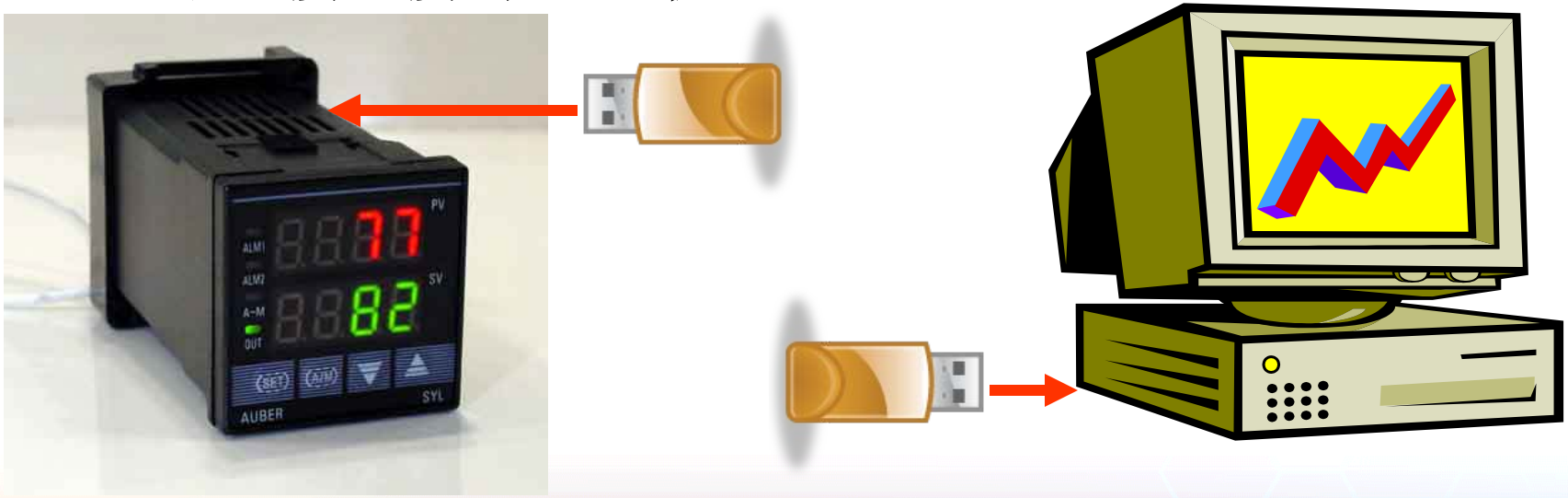
- 通常，主机是一台PC
- USB 2.0主机控制器，运行在全速（12 Mbps）或高速（480 Mbps）
- 高吞吐量的32/64位CPU，运行诸如Windows®这样的操作系统
- USB驱动程序，用来识别和枚举USB设备
- 自动更新USB驱动程序

嵌入式设计中的主机

- 可供选择的办法：在单片机中实现PC
  - 高成本，占据的空间更大
  - 根本不适合嵌入式设计
- 解决方案：实现嵌入式主机

嵌入式主机

- 嵌入式主机连接至数目固定的USB外设——USB驱动程序固化在固件中
- 优点：更小、不太复杂的嵌入式固件
- 示例：远程温度数据记录仪
  - 把数据下载至USB闪存盘
  - 在连接闪存盘时充当主机，但…
  - 不直接连接到PC主机



嵌入式主机的典型需求

- 对真实世界I/O进行服务的外设：
  - A/D转换器和比较器，等等
  - 串行接口，诸如SPI、I²C™和UART
  - PWM、定时器和I/O线
- 全速（12 Mbps）USB 2.0收发器，具有数据输入/输出缓冲功能
- 高吞吐量的16/32位MCU
- 主机固件驱动程序，用来识别和枚举USB外设
- Microchip的16/32位USB PIC®单片机，专为嵌入式主机应用而设计

USB外设

- 对主机进行响应，不能发起事务
- 需要驱动程序，以便让主机识别出
- 对主机进行响应所需的硬件/固件
- Microchip的PIC®单片机用在USB外设中

外设的典型MCU需求

- 对真实世界I/O进行服务的外设：
  - A/D转换器和比较器，等等
  - 串行接口，诸如SPI、I²C™和UART
  - PWM、定时器和I/O线
- 全速（12 Mbps）USB 2.0收发器，具有数据输入/输出缓冲功能
- 高吞吐量，以便对全速USB请求进行服务
- USB设备及外部接口固件
- 所有的USB PIC®单片机，专为USB外设应用而设计

USB设备类型

- 外设（也称为“功能”）
  - 向主机提供某种功能（能力）
    - **例如：数据获取**
- 集线器
  - 数据转发（双向），供电管理
- 混合设备（Compound Device）
  - 拥有一个集线器以及一个或多个外设
  - 主机分别处理集线器和外设功能（每个都有其各自的地址）
    - **例如：带单端口集线器的USB键盘**
- 复合设备（Composite Device）
  - 同一时间有多个接口在工作
  - 主机为每个接口装载驱动程序
    - **例如：视频头（音频和视频接口同时工作）**

USB On-The-Go（OTG）

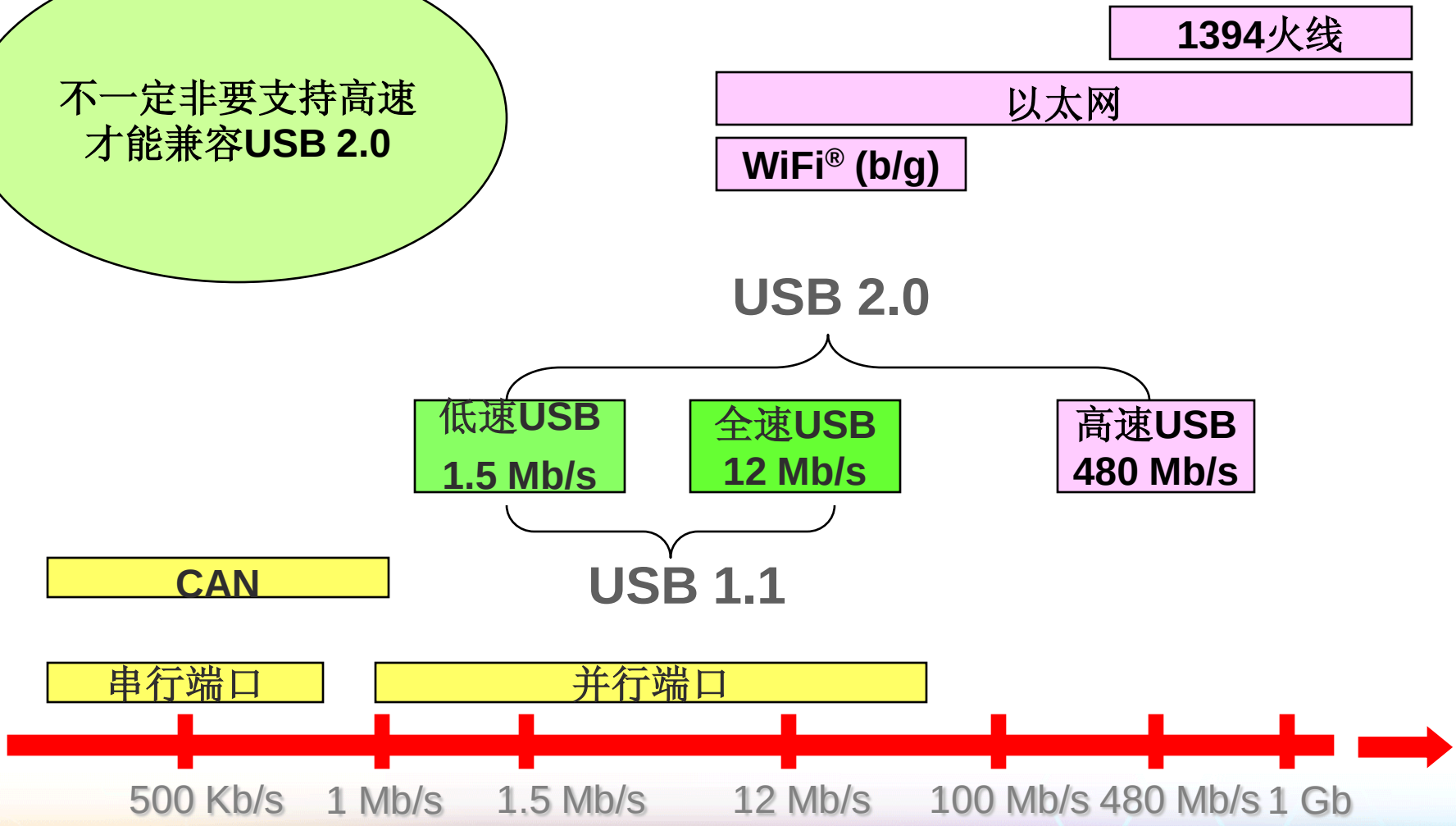
- **USB On-The-Go（OTG）允许应用作为主机或设备进行工作：**
  - **PDA（设备）连接至PC（主机）**
  - **PDA（主机）连接至拇指驱动器（设备）**
  - **PDA连接至PDA，主机与设备角色可互换（OTG模式）**

OTG设备的典型需求

- 对真实世界I/O进行服务的外设：
  - A/D转换器和比较器，等等
  - 串行接口，诸如SPI、I²C™和UART
  - PWM、定时器和I/O线
- 全速（12 Mbps）USB 2.0收发器，具有数据输入/输出缓冲功能
- 高吞吐量的16/32位MCU
- 主机软件驱动程序，用来识别和枚举USB外设
  - 当连接至主机/OTG时的设备驱动程序
- Microchip的16/32位USB PIC®单片机，专为嵌入式主机应用而设计

总线与速度比较

不一定非要支持高速
才能兼容USB 2.0



最大的神话

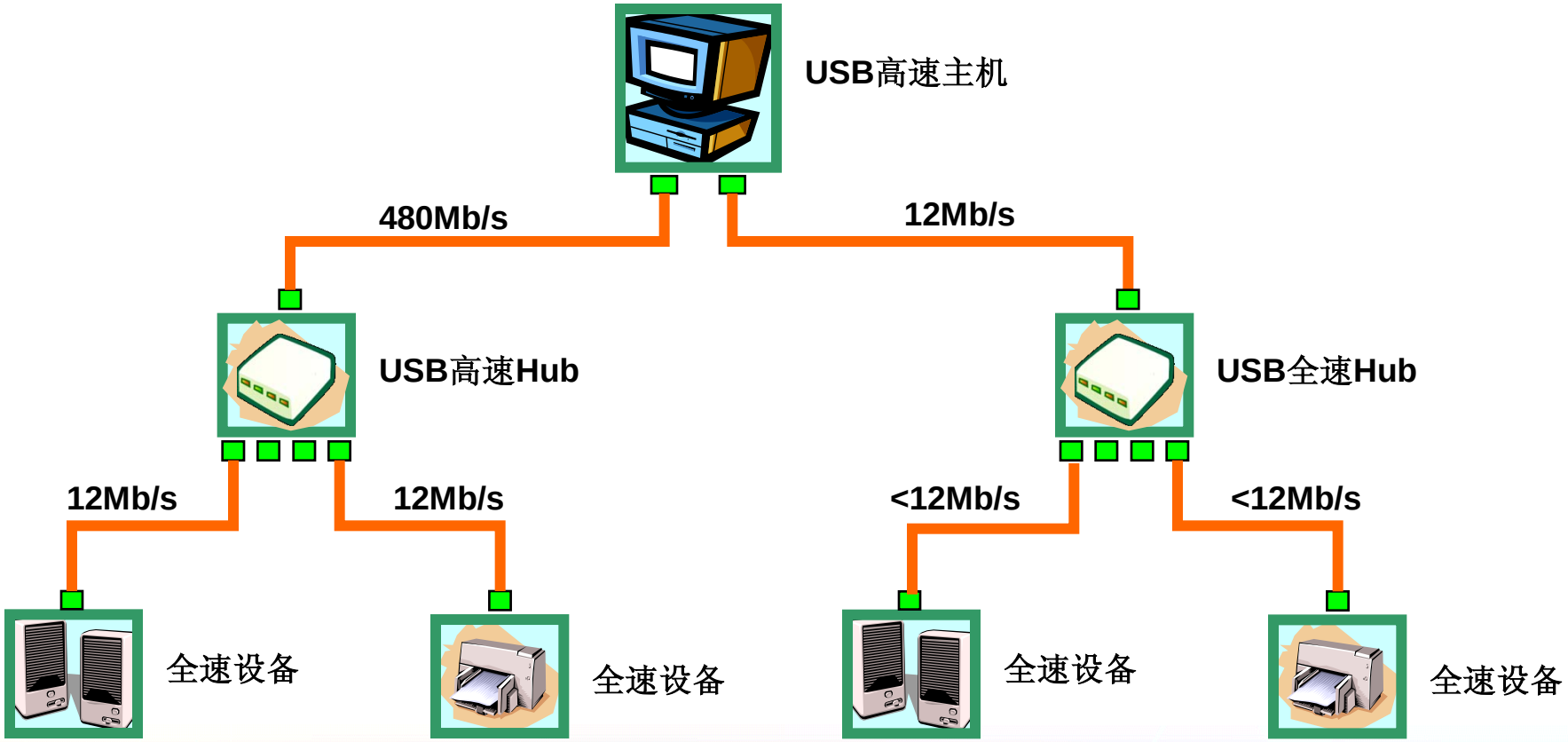
神话：全速USB外设传输数据的速度可高达1.5 MB/s（12 Mb/s）

事实：不可能，1.5 MB/s是总的总线带宽

- 必须与其他外设共享
- 协议开销
- 协议限制
- 至单个外设的真实原始数据吞吐量是 ~1.0 MB/s
- 某些情况下，仅64 KB/s

技巧

把全速外设连接至高速集线器



课程安排

- 嵌入式应用设计人员面临的连通性挑战
- **USB基础 —— 严肃且重要的资料**
  - USB简史
  - 基础/速度
  - 拓扑/物理连接
  - 架构/程序员模型
  - USB事务
  - USB传输
  - 设备类
  - 枚举
  - 描述符
  - VID/PID及USB兼容性
- **PIC18/24/32 USB单片机**
- **Microchip演示/开发解决方案**
- 如何着手

物理总线拓扑

主机（第1层）

USB主机控制器
及根集线器

第2层

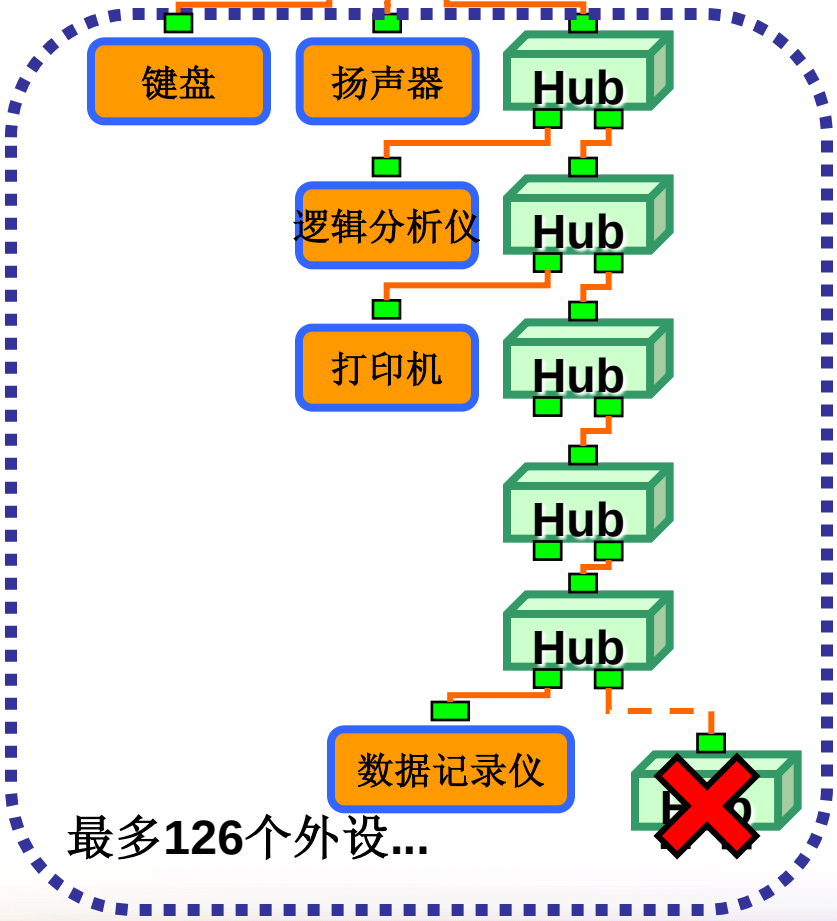
第3层

第4层

第5层

第6层

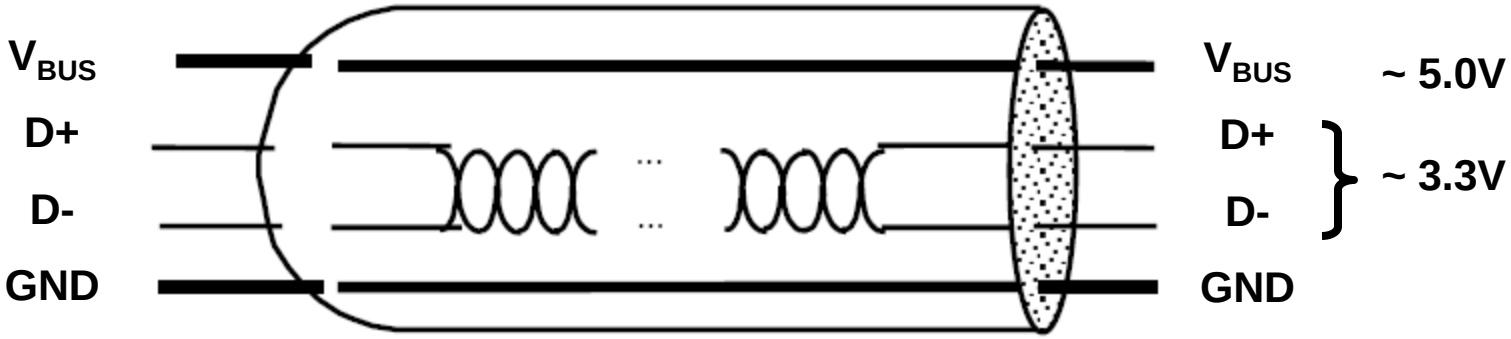
第7层



集线器：最大链接层数 = 5

PIC18 USB器件，专为USB外设而设计
PIC24/PIC32可用作嵌入式主机或外设

物理接口



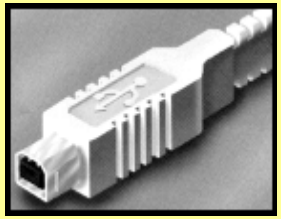
- 半双工，使用**NRZI**数据编码
- 总线向每个设备供电：
  - **4.40 - 5.25V**
  - 保证**100 mA**
  - 通过协商，最大可达**500 mA**

如需要更大的功率，
必须使用外部电源

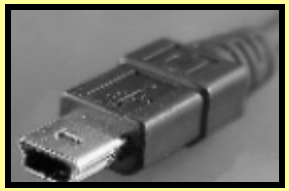
标准连接器

“A”
USB主机
全速或嵌入式

“B”
USB
设备



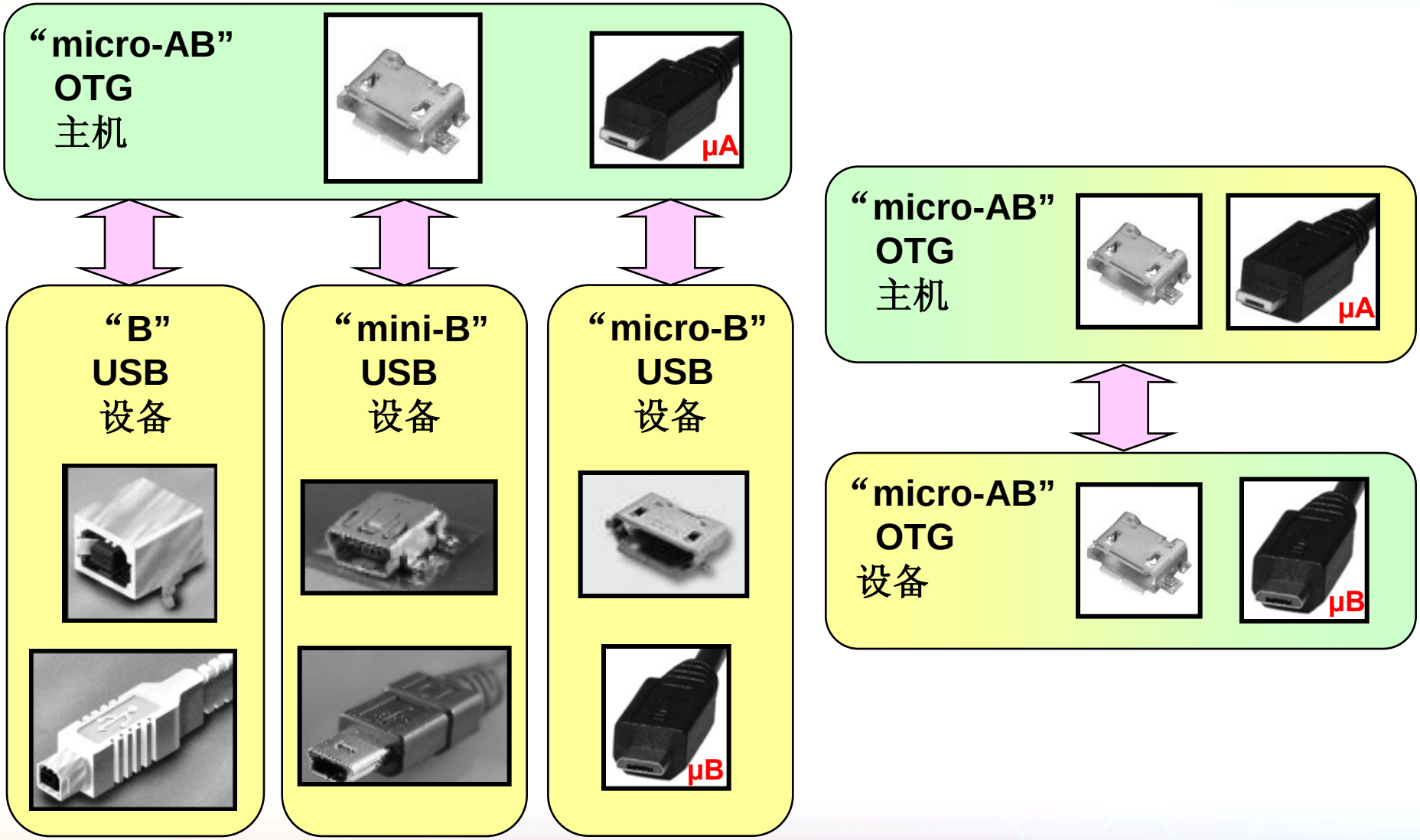
“mini-B”
USB
设备



“micro-B”
USB
设备

“micro-AB”
OTG
设备

标准连接器

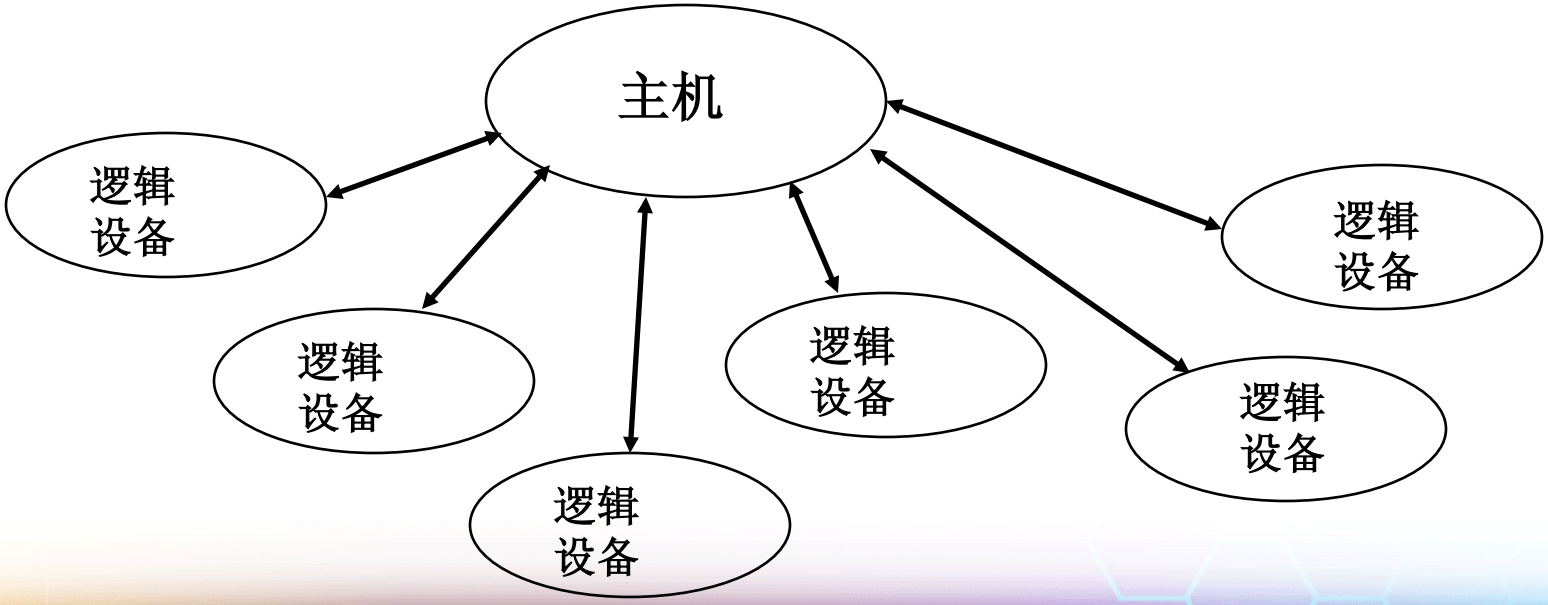


课程安排

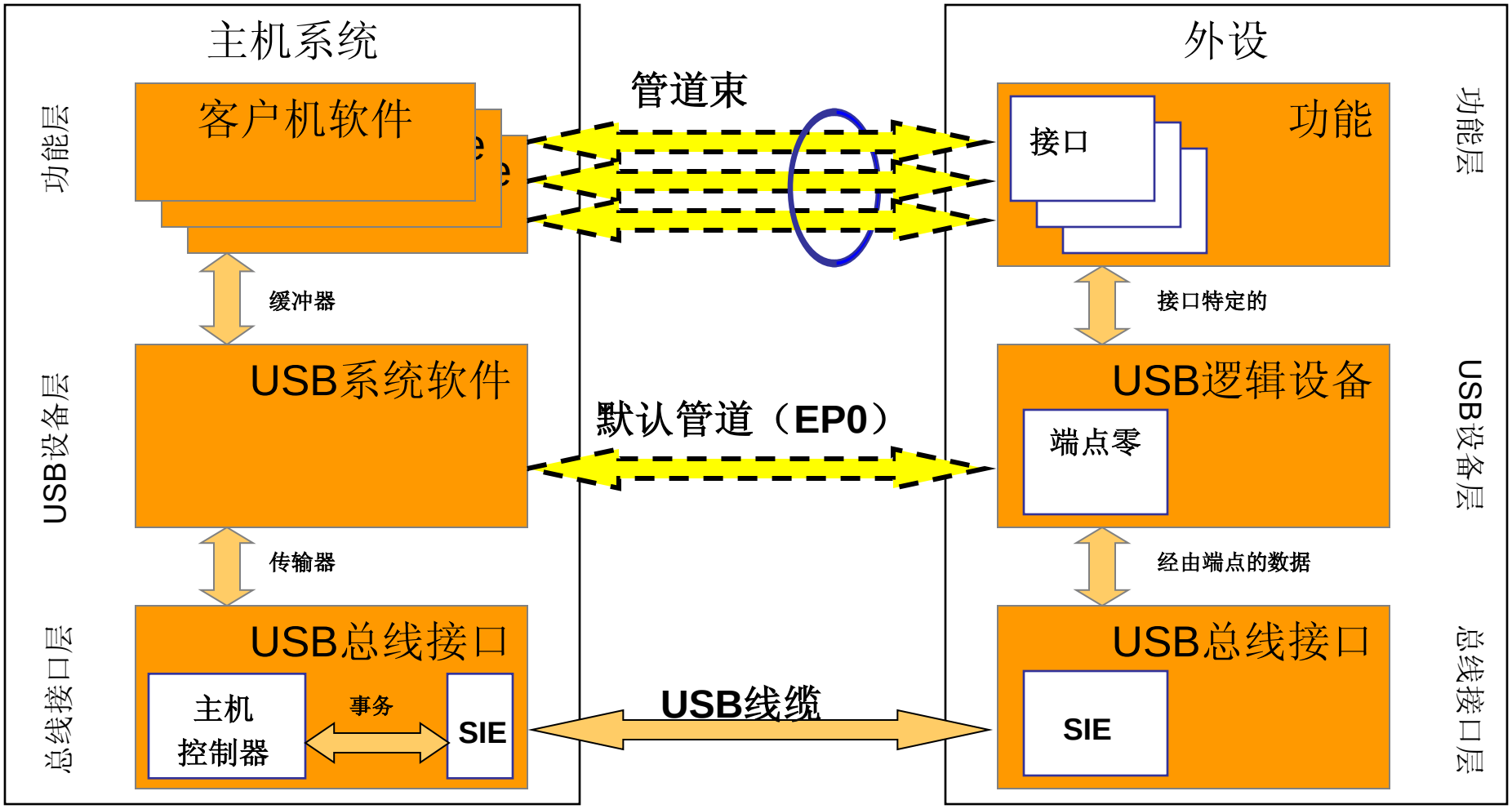
- 嵌入式设计人员面临的连通性挑战
- **USB基础 —— 严肃且重要的资料**
  - USB简史
  - 基础/速度
  - 拓扑/物理连接
  - 架构/程序员模型
  - USB事务
  - USB传输
  - 设备类
  - 枚举
  - 描述符
  - VID/PID及USB兼容性
- **PIC18/24/32 USB单片机**
- **Microchip演示/开发解决方案**
- 如何着手

逻辑总线拓扑

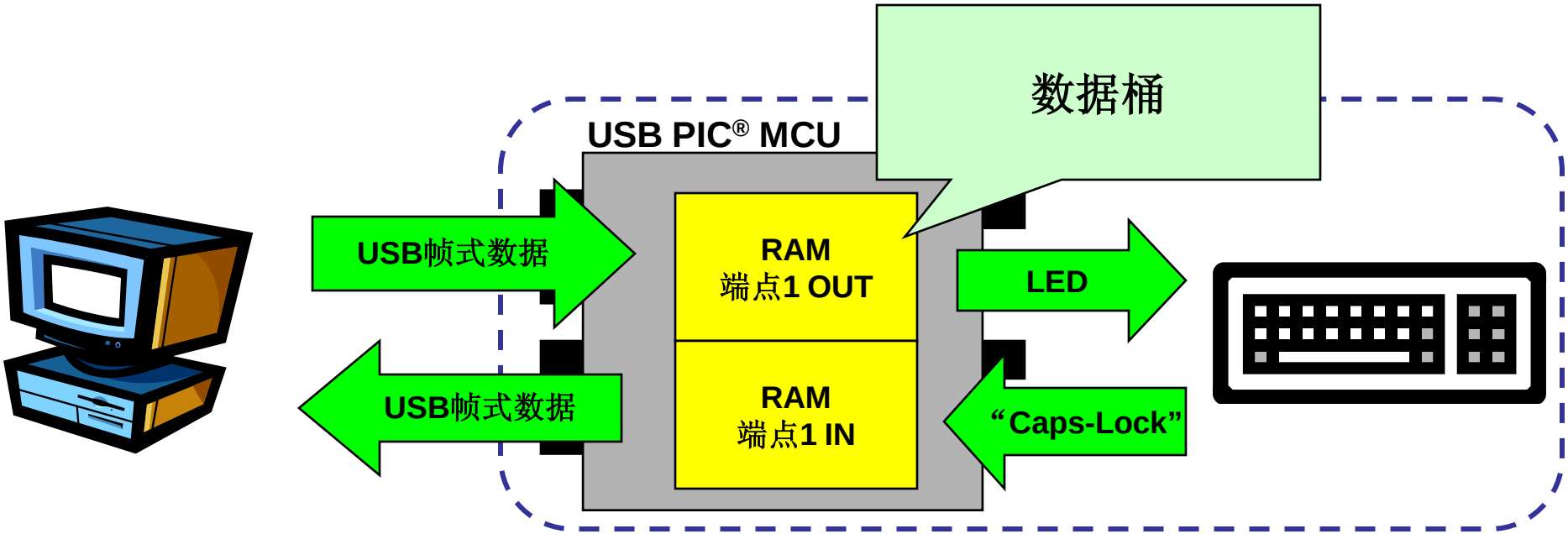
- 不是层接的星形拓扑！
- 主机软件与每个“逻辑”设备进行通信，就好像每个“逻辑”设备直接连接到根集线器一样



USB设备框架
- 硬件的软件视角 -

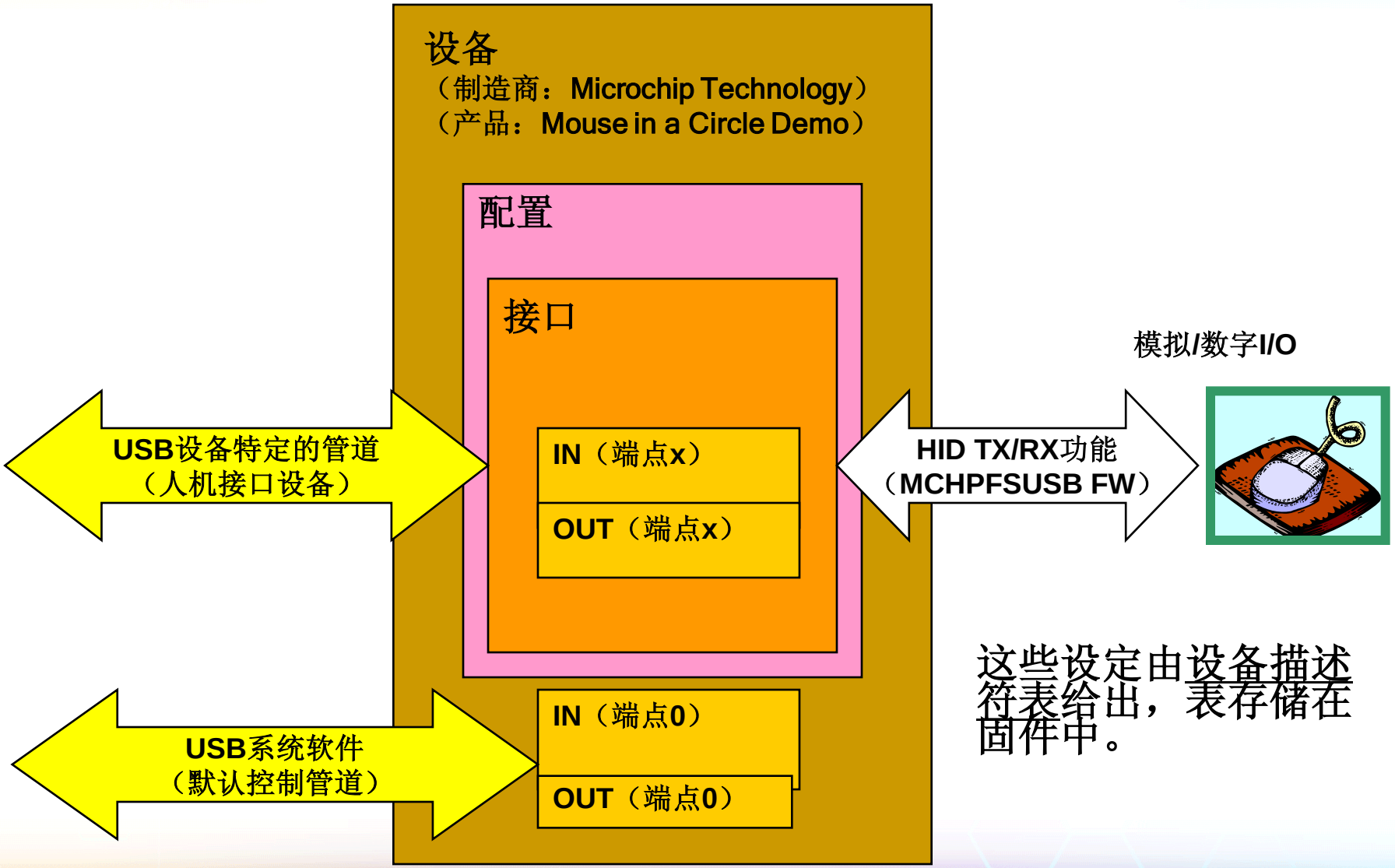


端点：外设中USB数据的源/目的



- 根据USB规范的规定，每一设备的最大端点数是：
  - 16个OUT端点 + 16个IN端点 = 32个端点
  - PIC18F87J50、PIC18F4550、PIC24F和PIC32MX支持最多32个端点
  - PIC18F14K50支持最多16个端点
- EP0 = 默认的通信管道

“逻辑”设备



这些设定由设备描述符表给出，表存储在固件中。

访问PC外设

- 老方法
- PC外设：
  - 存储器映射至x86 I/O地址空间
  - 分配具体的IRQ线
  - 分配具体的DMA通道
- 直接访问（ISA、PCI和PCMCIA）

- USB方法
- PC外设：
  - 映射至虚拟的127个设备地址空间
  - 不使用任何PC I/O、IRQ或DMA资源
- 使用设备驱动程序提供的编程接口进行间接访问

在PC上发送/接收

- 仅限高层访问
- 四种基本的函数类型
- 示例：
  - Microchip通用USB设备驱动程序

```
MPUSBOpen(…);
MPUSBRead(…);
MPUSBWrite(…);
MPUSBClose(…);
```

  - Microsoft WinUSB驱动程序

```
WinUSB_Initialize(…);
WinUsb_ReadPipe(…);
WinUsb_WritePipe(…);
WinUSB_Free(…);
```

在设备上发送/接收

- 仅限高层访问
- 示例：CDC类RS-232仿真

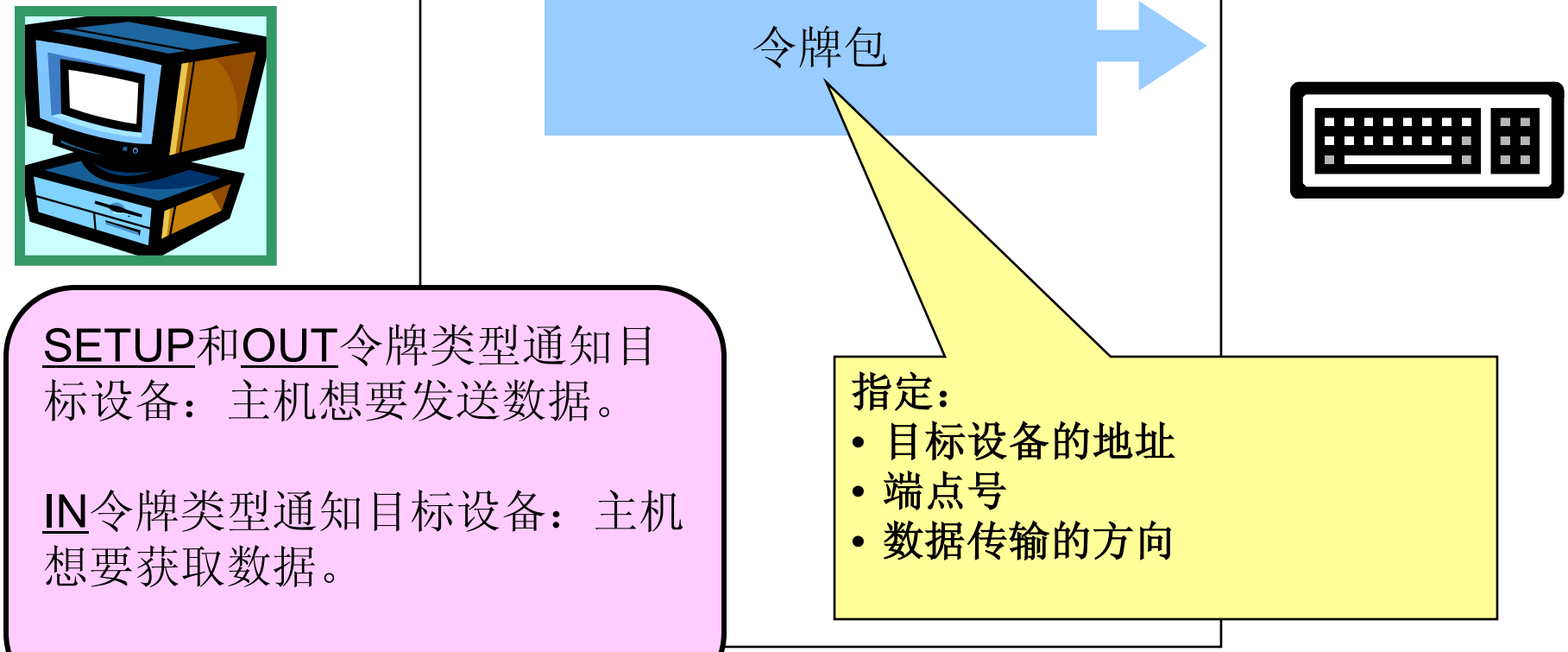
```
BOOL USBUSARTIsTxTrfReady(void);
void putUSBUSART(char *data, BYTE Length);
BYTE getsUSBUSART(char *buffer, BYTE len);
```

您再也不用直接读/写外设的特殊功能寄存器（SFR）了！

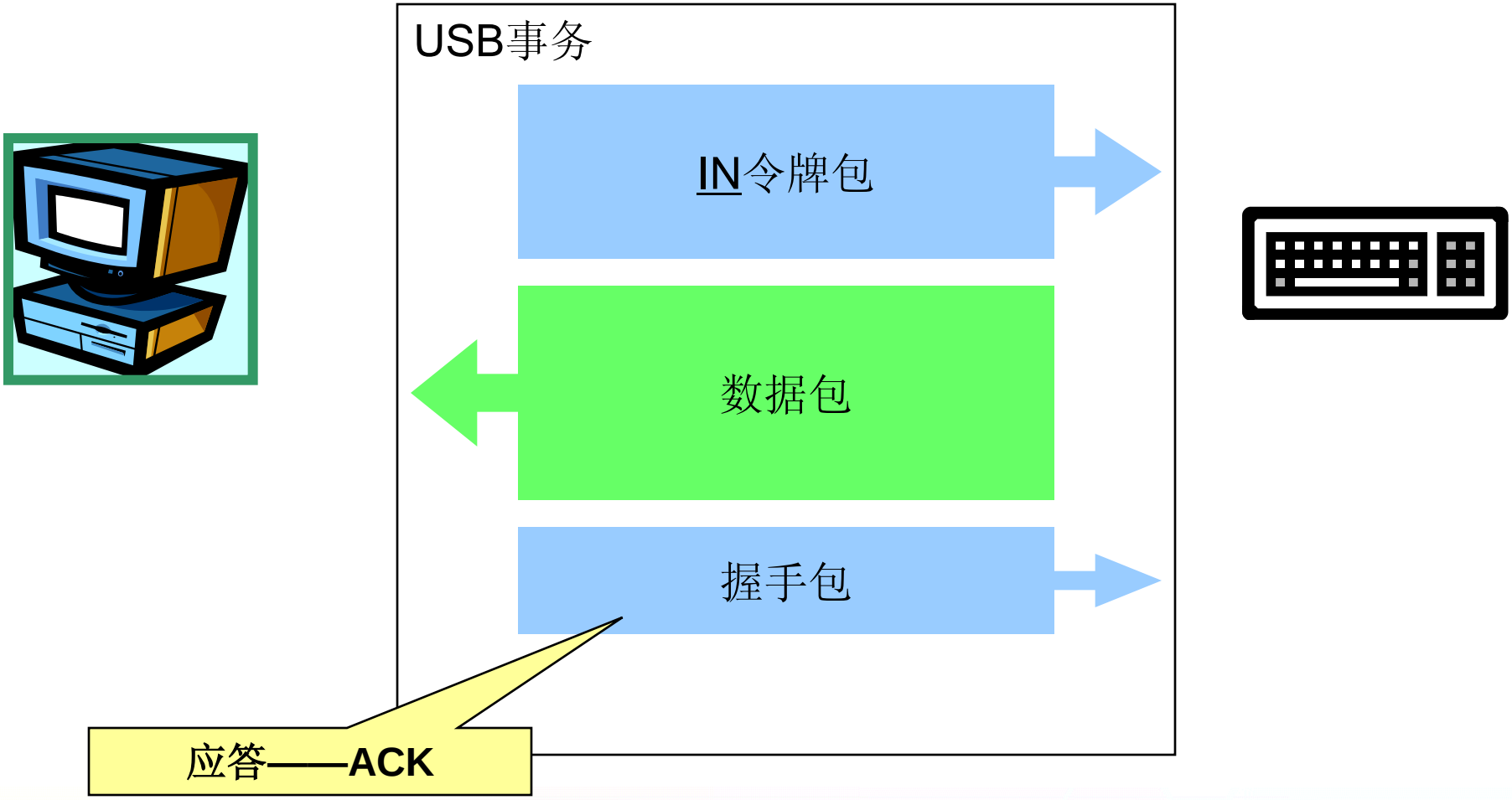
课程安排

- 嵌入式设计人员面临的连通性挑战
- **USB基础 —— 严肃且重要的资料**
  - USB简史
  - 基础/速度
  - 拓扑/物理连接
  - 架构/程序员模型
  - USB事务
  - USB传输
  - 设备类
  - 枚举
  - 描述符
  - VID/PID及USB兼容性
- **PIC18/24/32 USB单片机**
- **Microchip演示/开发解决方案**
- 如何着手

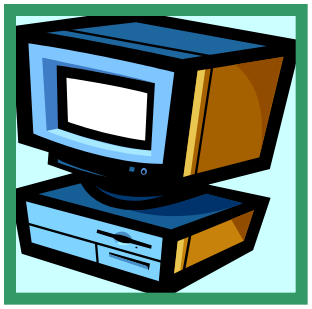
USB事务

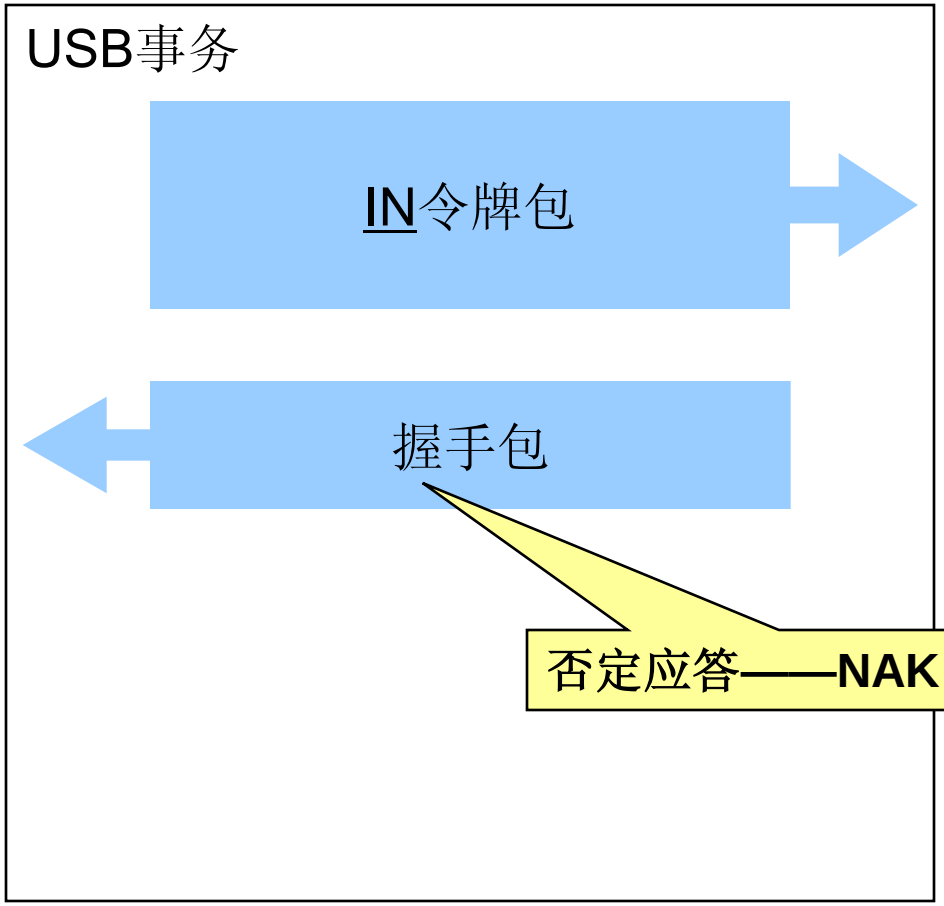


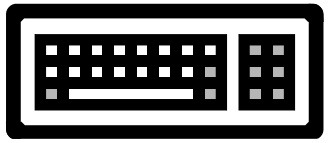
IN事务ACK



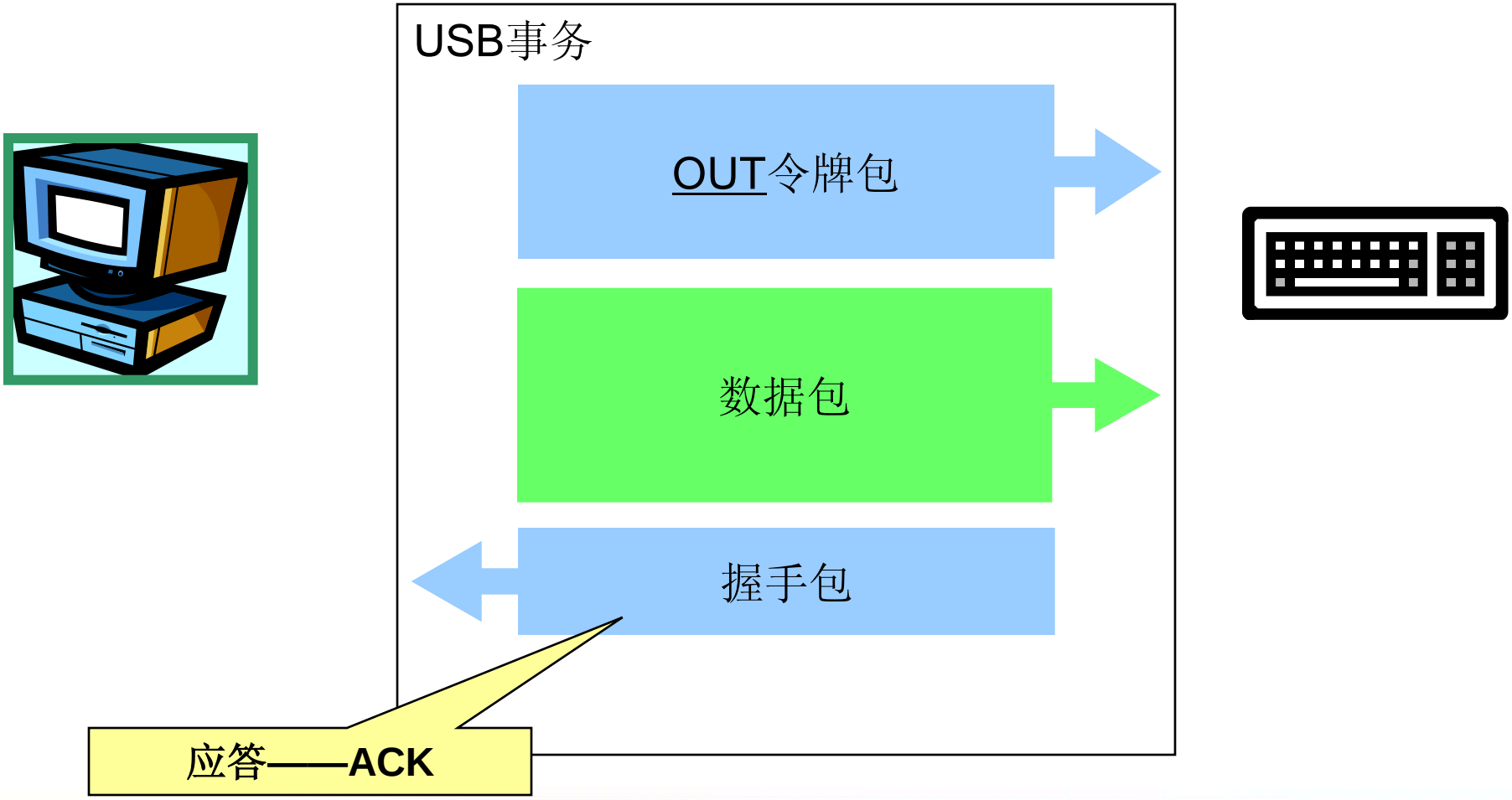
IN事务NAK



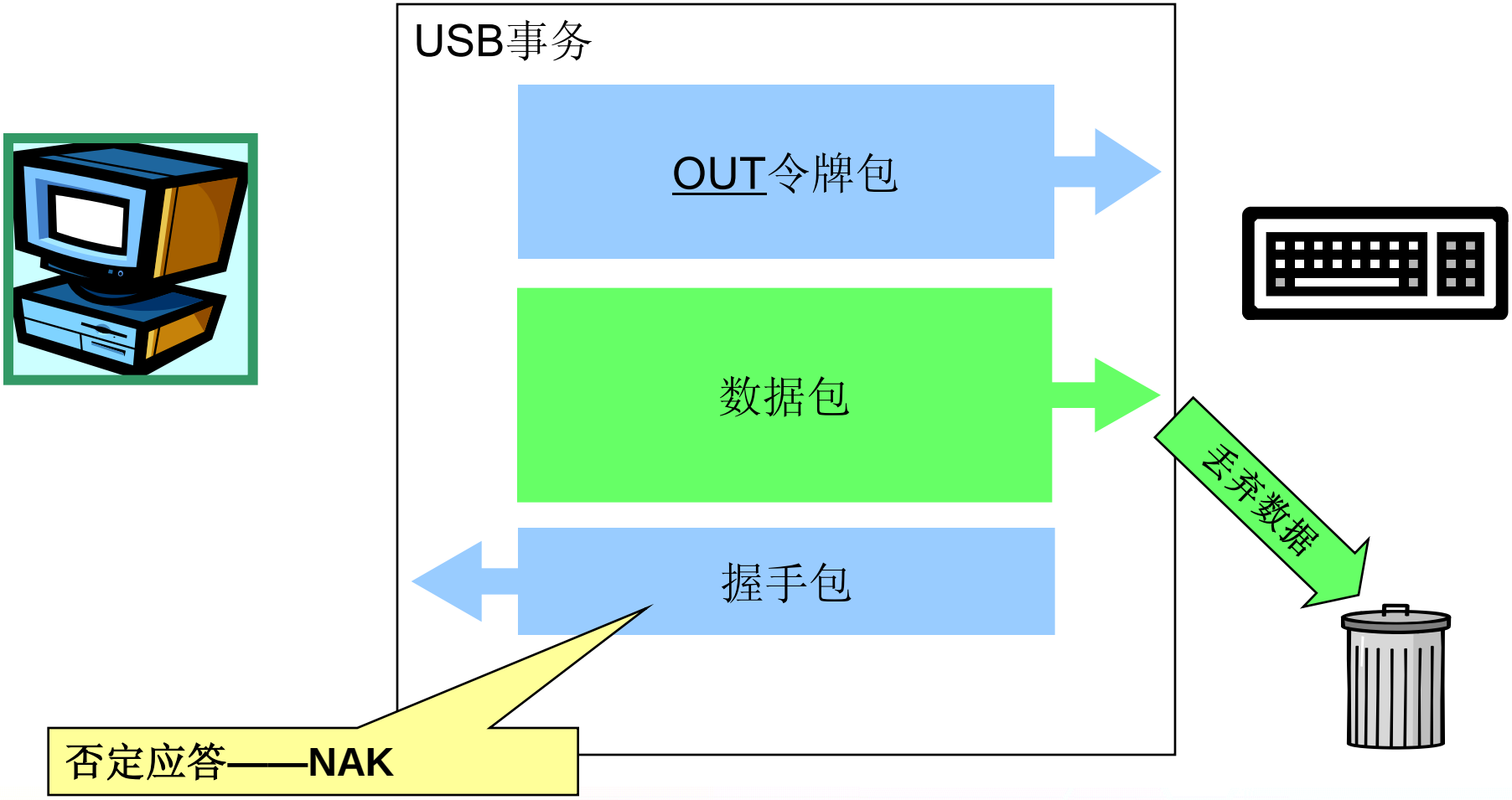




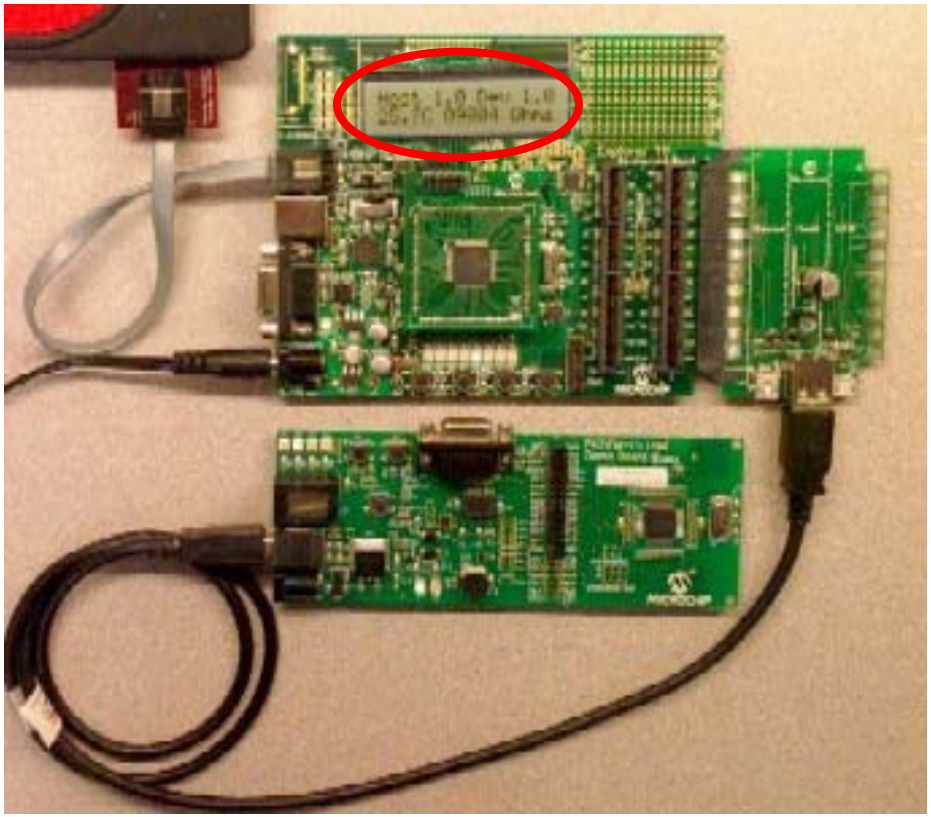
OUT事务ACK



OUT事务NAK



演示1
- 捕捉事务 -



- 为了读取电位器值，PIC24F USB嵌入式主机连续发送命令“0x37”
- PIC18F USB器件返回命令代码 + 10位电位器ADC值：<0x37><ADRESL><ADRESH>

演示1
- 捕捉事务 -

*Untitled - Total Phase Data Center

File Edit Analyzer View Help

102.8 KB

Sp	Index	m:s.ms.us	Len	Err	Dev	Ep	Record	Data
	942	4:50.636.407	1 B		01	01	OUT txn	37
	943	4:50.636.407	3 B		01	01	OUT packet	E1 81 58
	944	4:50.636.410	4 B		01	01	DATA1 packet	4B 37 01 69
	945	4:50.636.414	1 B		01	01	ACK packet	D2
	946	4:50.637.400	1.00 ms				[2 SOF]	[Frames: 1929 - 1930]
	947	4:50.638.405	3 B		01	01	IN txn	37 BF 02
	948	4:50.638.405	3 B		01	01	IN packet	69 81 58
	949	4:50.638.408	6 B		01	01	DATA1 packet	4B 37 BF 02 CE 00
	950	4:50.638.414	1 B		01	01	ACK packet	D2
	951	4:50.639.400	1.00 ms				[2 SOF]	[Frames: 1931 - 1932]

37

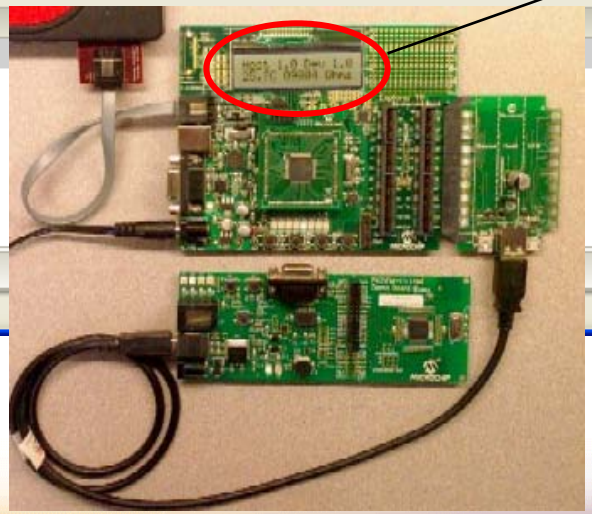
No filter: 1,373 records.

Protocol Lens: USB

Details

Offset	0	1
00000	37	
00010		
00020		
00030		
00040		
00050		

Data



Navigator

Description	Packets
Universal Serial Bus	
Unconfigured Device (0)	5
Microchip Custom USB Device (1)	260

Device Details	
Product	Microchip Custom USB Device
Serial Number	<none>
Manufacturer	<not available>
Class	Defined in Interface

VID	PID	Rev	USB
0x04d8	0x000c	0.0	2.0

Configurations	
Config 1	Self Powered, 100mA
OTG	none / corrupted
IF 0 (alt 0)	0, 0, 0
EP 1 OUT	Bulk, 64B, FS:1ms HS:125us
EP 1 IN	Bulk, 64B, FS:1ms HS:125us

BOS

BOS descriptor not detected or corrupted.

Bus Filter Info

Ready

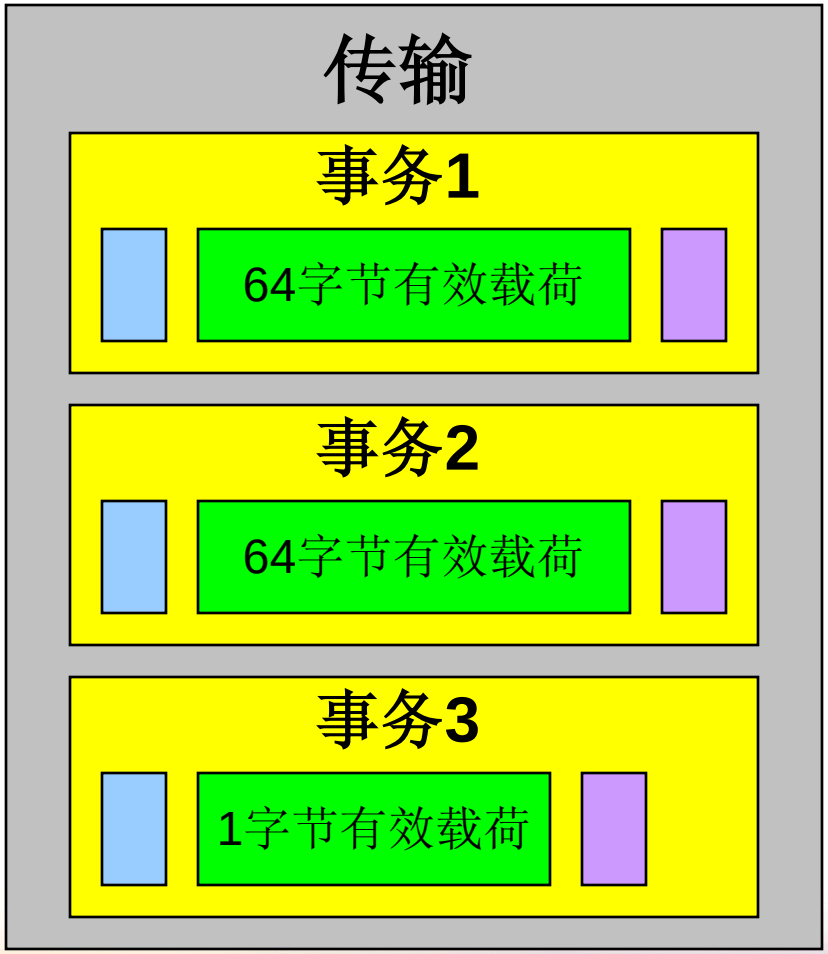
SN: 1112-111796 HW: 1.10 FW: 3.08 USB 21 ns EN

课程安排

- 嵌入式设计人员面临的连通性挑战
- **USB基础 —— 严肃且重要的资料**
  - USB简史
  - 基础/速度
  - 拓扑/物理连接
  - 架构/程序员模型
  - USB事务
  - USB传输
  - 设备类
  - 枚举
  - 描述符
  - VID/PID及USB兼容性
- **PIC18/24/32 USB单片机**
- **Microchip演示/开发解决方案**
- 如何着手

传输：一组相关的事务

```
MPUSBWrite(EP7, Pointer, Size = 129, Timeout)
```



解释：

- OUT令牌包
- 数据包
- ACK握手包

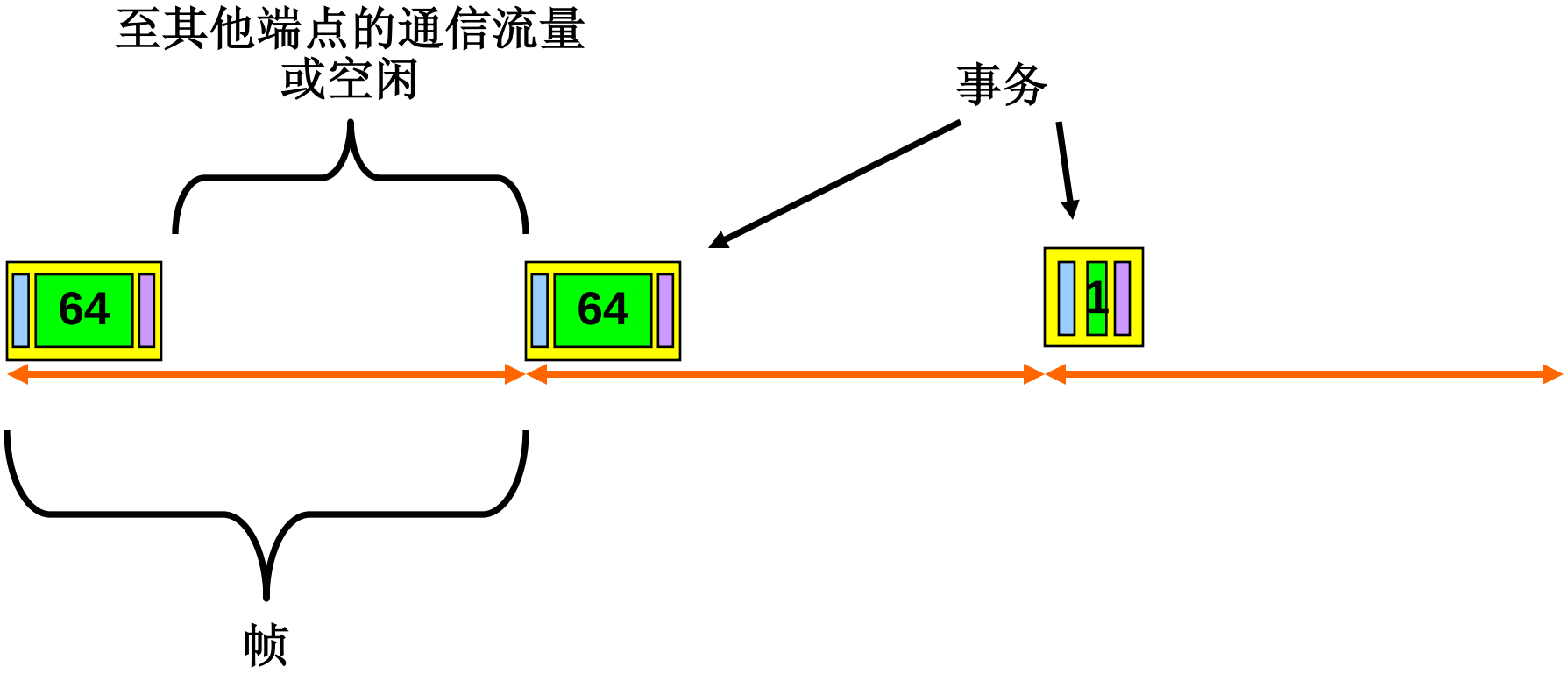
数据传输类型

传输/端点类型	轮询时间间隔	为此类型所有传输保留的BW/帧的比例（%）	最大数据字节/帧/端点数（对于每帧@最大端点长度，最大的事务数）*	数据完整性
中断	固定，周期性	90	64 (1 x 64)	是
同步	固定，周期性	90	1023 (1 x 1023)	否
批量	可变，使用闲置带宽	0	1216 (19 x 64)	是
控制	可变	10	832 (13 x 64)	是

*假定传输使用每种端点类型所允许的最大包长度

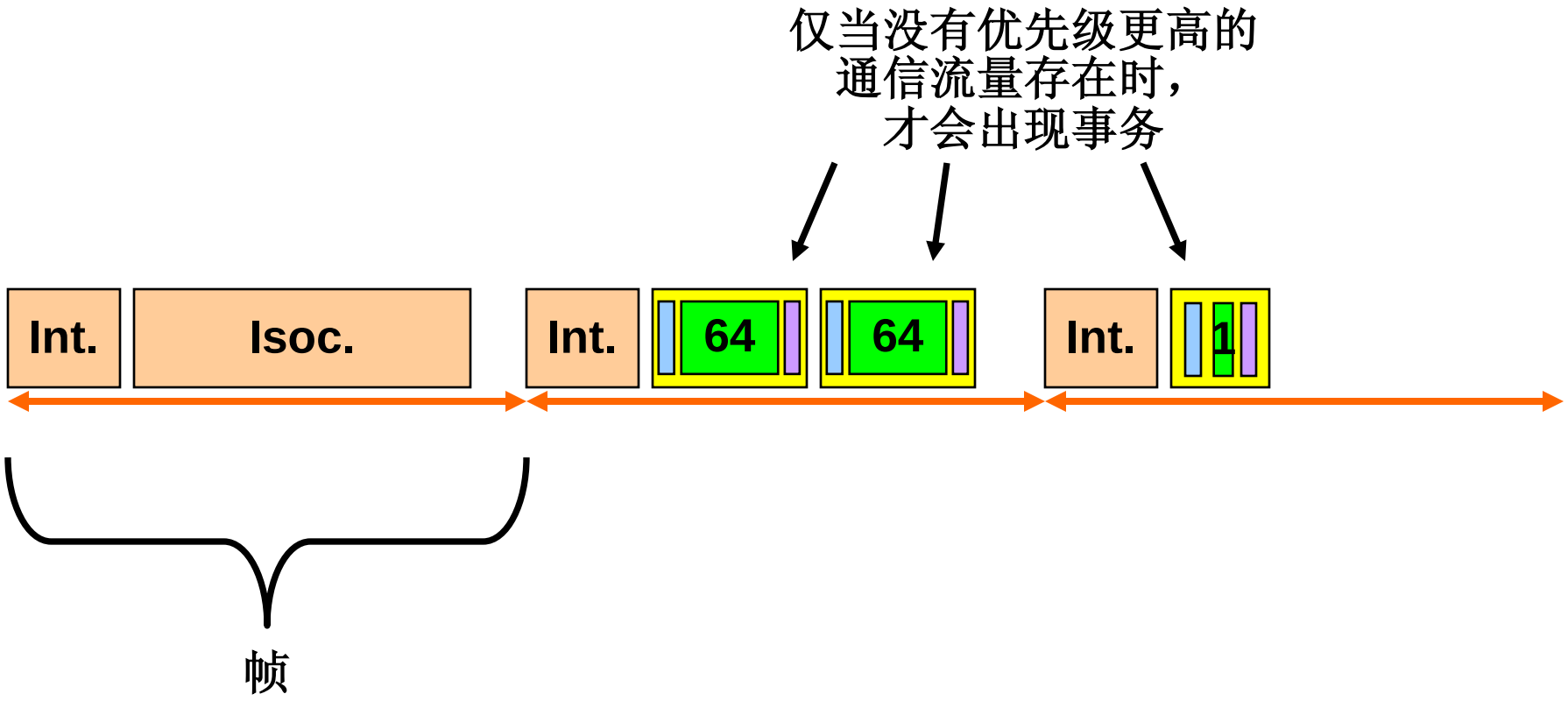
中断传输示例

```
MPUSBWrite(EP7, Pointer, Size = 129, Timeout)
```

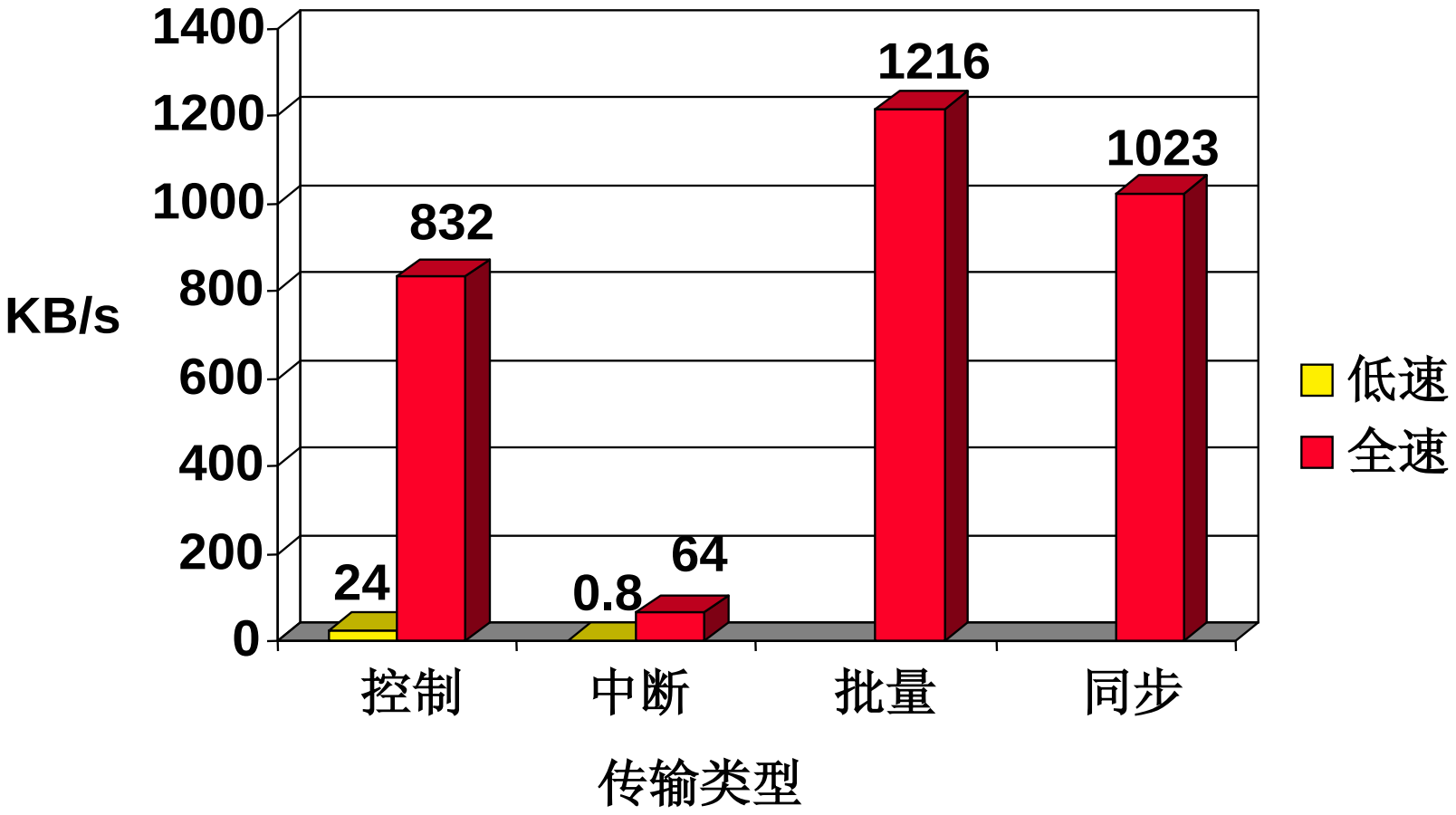


批量传输示例

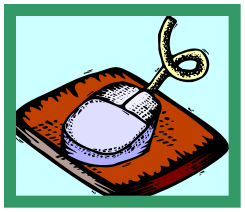
```
MPUSBWrite(EP7, Pointer, Size = 129, Timeout)
```



每端点的理论最大传输速率

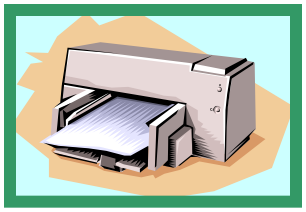


传输类型 —— 示例



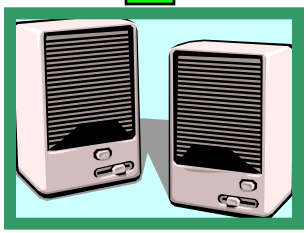
鼠标

控制
+
中断



打印机

控制
+
批量



扬声器

控制
+
同步

课程安排

- 嵌入式设计人员面临的连通性挑战
- **USB基础 —— 严肃且重要的资料**
  - USB简史
  - 基础/速度
  - 拓扑/物理连接
  - 架构/程序员模型
  - USB事务
  - USB传输
  - 设备类
  - 枚举
  - 描述符
  - VID/PID及USB兼容性
- **PIC18/24/32 USB单片机**
- **Microchip演示/开发解决方案**
- 如何着手

USB设备类

数据手套

鼠标

PICkit™ 3
入门工具包

游戏杆

键盘

人机接口设备类（HID）

拇指驱动器

外置式硬盘

海量存储
设备类（MSD）

以太网
适配器

调制解调器

通信设备类（CDC）

MPLAB®
REAL ICE™
在线仿真器

定制类
（供应商类）

更多的类....

使用标准类时的注意事项

- 预定义逻辑USB设备
  - 最大带宽是固定的
- 定义设备数据通信协议
  - 对于CDC类，PIC® MCU看上去就像是调制解调器，或者是连接虚拟COM端口的终端
- 主要优点：跨平台使用
  - 不需要定制的OS驱动程序！

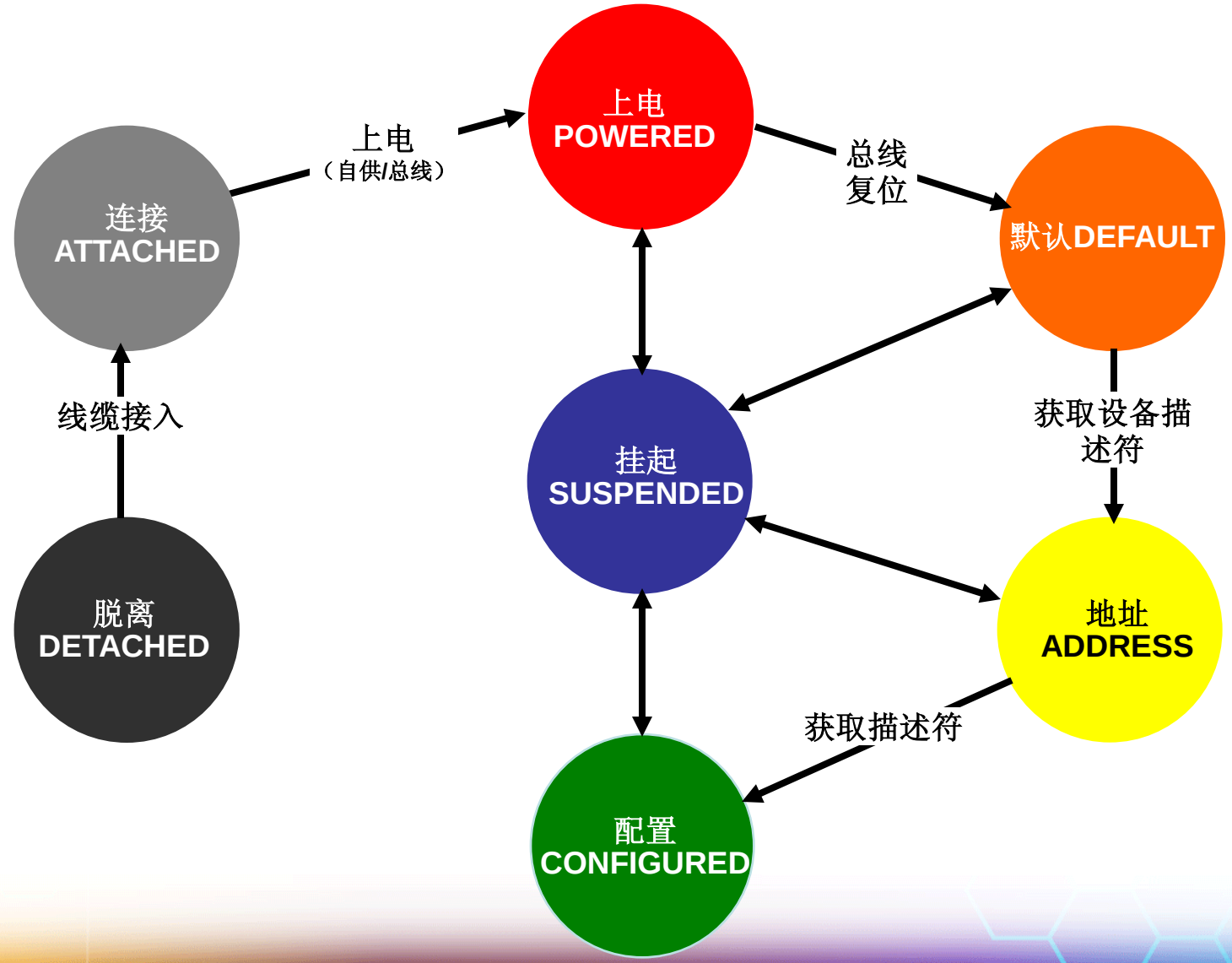
USB驱动程序选择
- Windows® PC主机 -

特性	HID	CDC	MCHPUSB	WinUSB	LibUSB
Windows内置驱动程序支持	是	需要 .inf	否	需要 .inf	否
64位PC支持	是	是	是	是	是
XP Ready	是	是	是	是	是
Vista Ready	是	是	是	是	32位
用户数据的传输类型					
控制	是	否	是	是	是
中断	是	否	是	是	是
同步	否	否	是	否	是
批量	否	是	是	是	是
最大速度	64 KB/s	~80 KB/s	~1.0 MB/s	~1.0 MB/s	~1.0 MB/s

课程安排

- 嵌入式设计人员面临的连通性挑战
- **USB基础 —— 严肃且重要的资料**
  - USB简史
  - 基础/速度
  - 拓扑/物理连接
  - 架构/程序员模型
  - USB事务
  - USB传输
  - 设备类
  - 枚举
  - 描述符
  - VID/PID及USB兼容性
- **PIC18/24/32 USB单片机**
- **Microchip演示/开发解决方案**
- **如何着手**

枚举过程


上电
POWERED
上电
（自供/总线）
连接
ATTACHED
总线
复位
默认DEFAULT
线缆接入
挂起
SUSPENDED
获取设备描
述符
脱离
DETACHED
地址
ADDRESS
获取描述符
配置
CONFIGURED

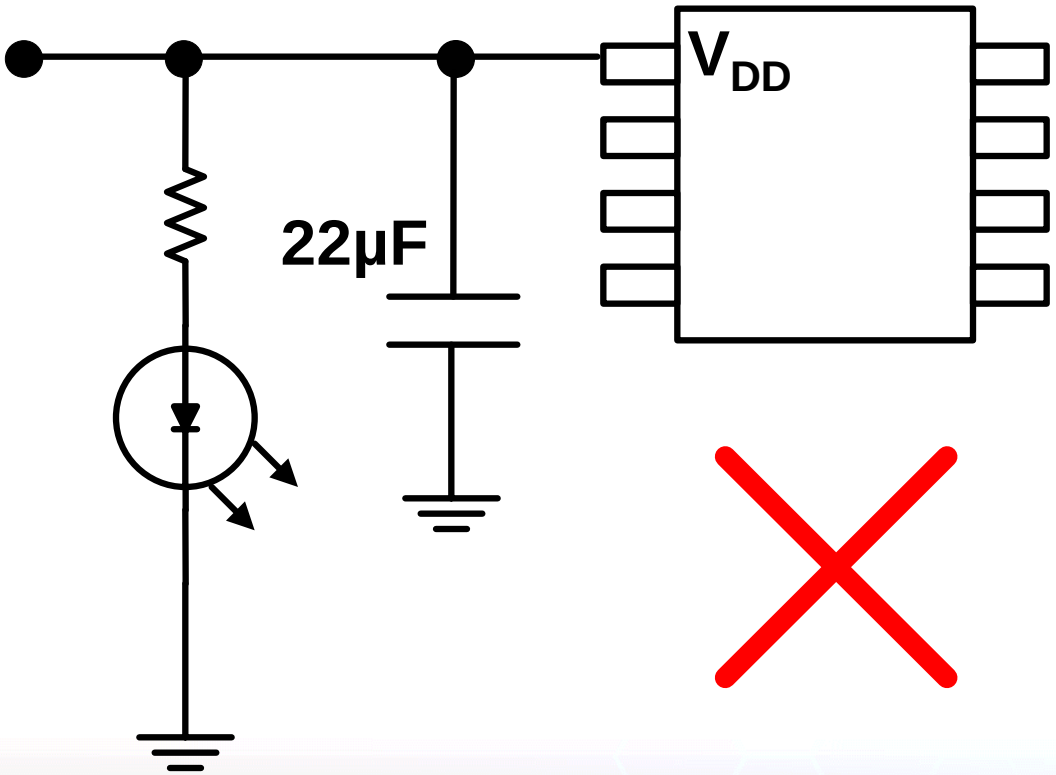
挂起（Suspend）模式

- 总线可能因主机进入“挂起”模式而停止活动，以延长电池寿命，
- 如果在3 ms内没有观察到总线活动，所有的设备都必须挂起
  - 且必须以低电流的方式工作...

挂起模式
- 总线供电设备 -

- 最大USB挂起电流：
  - 2.5 mA
- 不要：

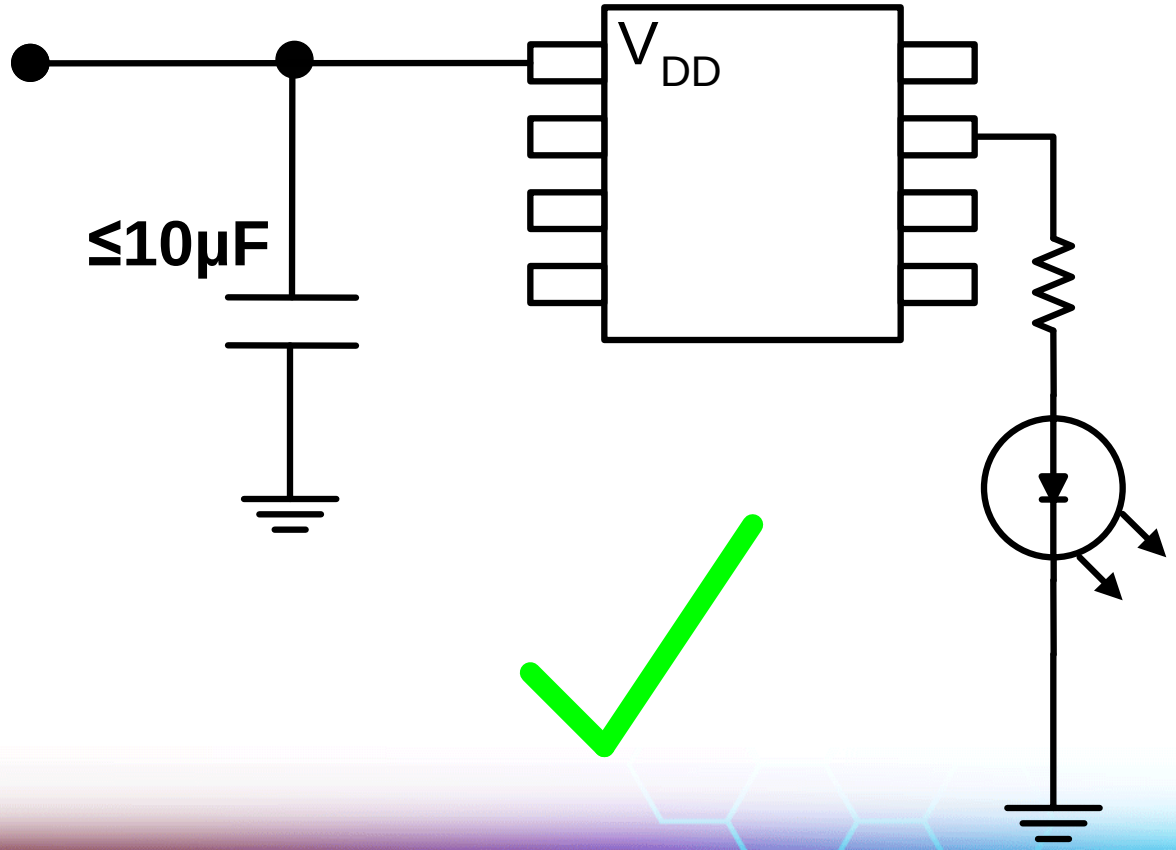
通过USB
线缆供电



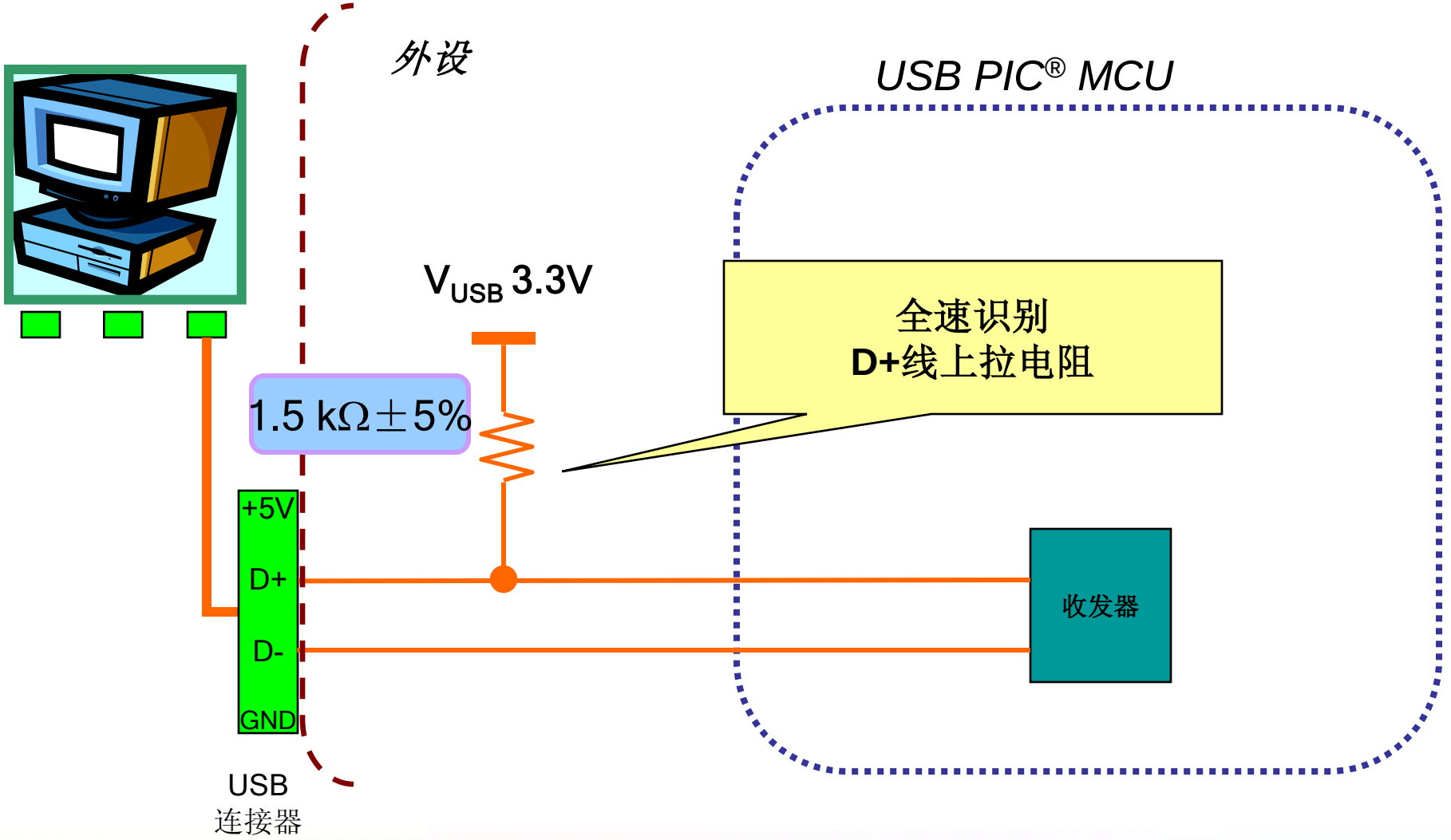
挂起模式
- 总线供电设备 -

- 最大USB挂起电流：
  - 2.5 mA
- 应：

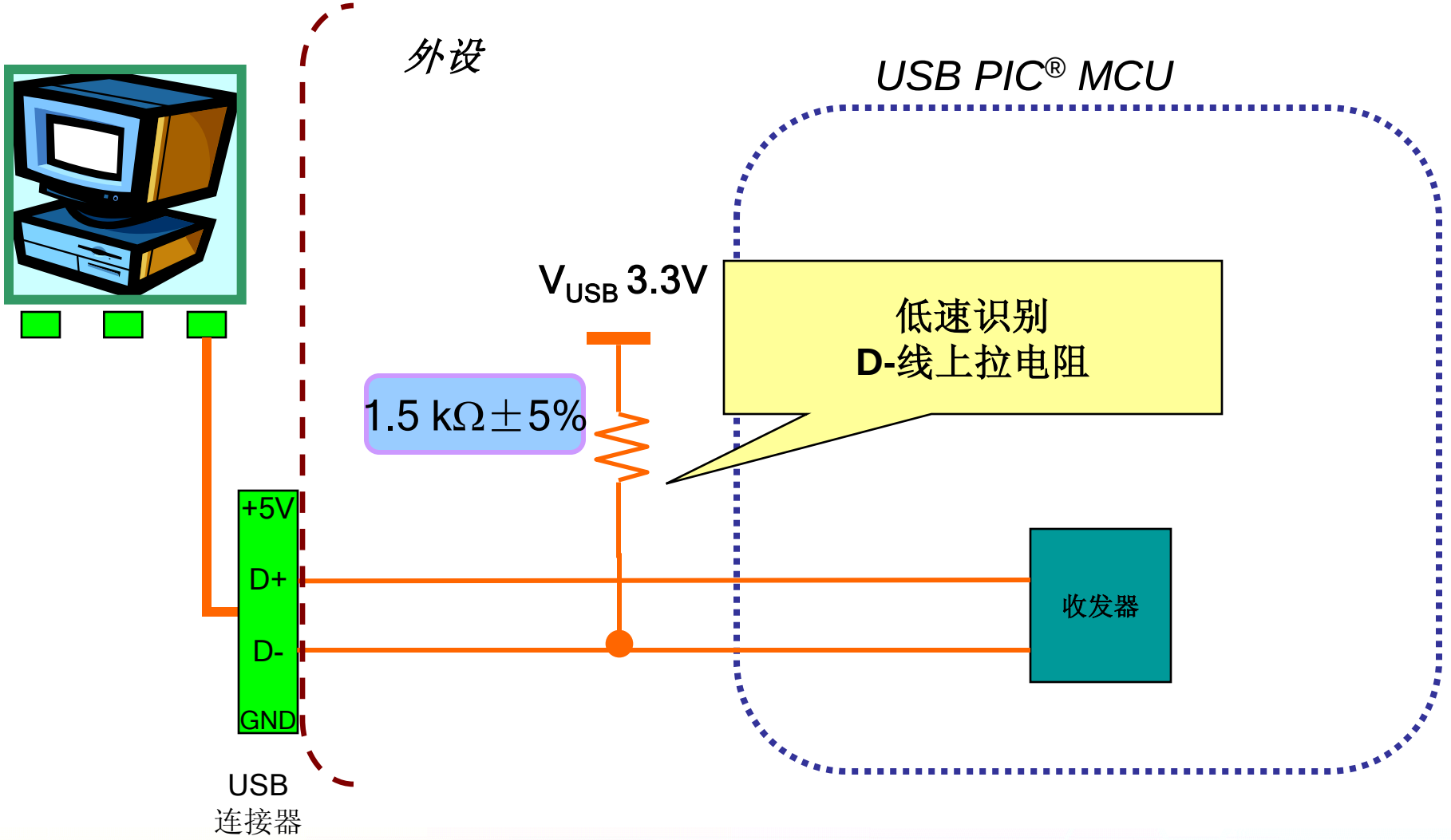
通过USB
线缆供电



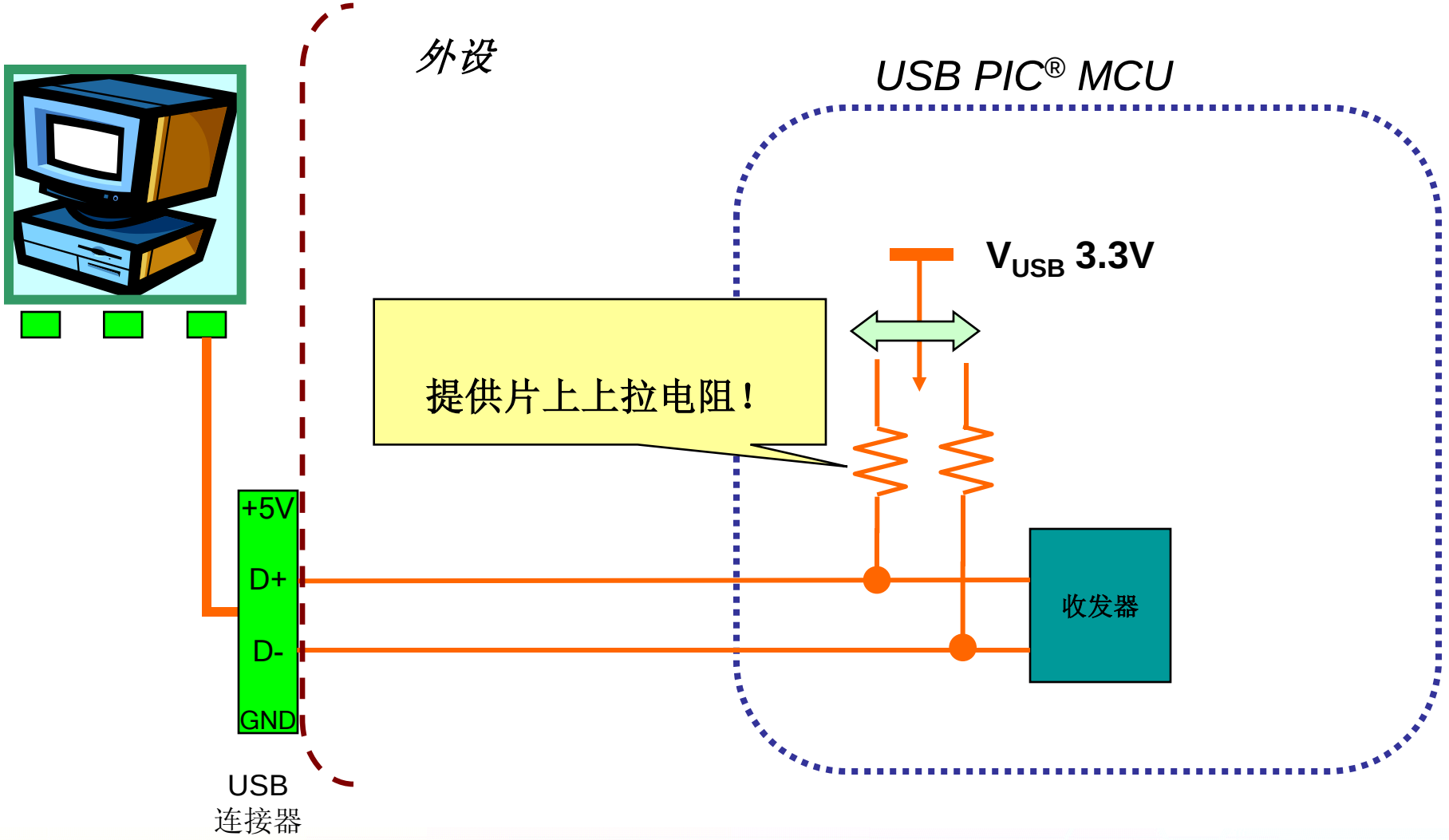
自动检测：全速



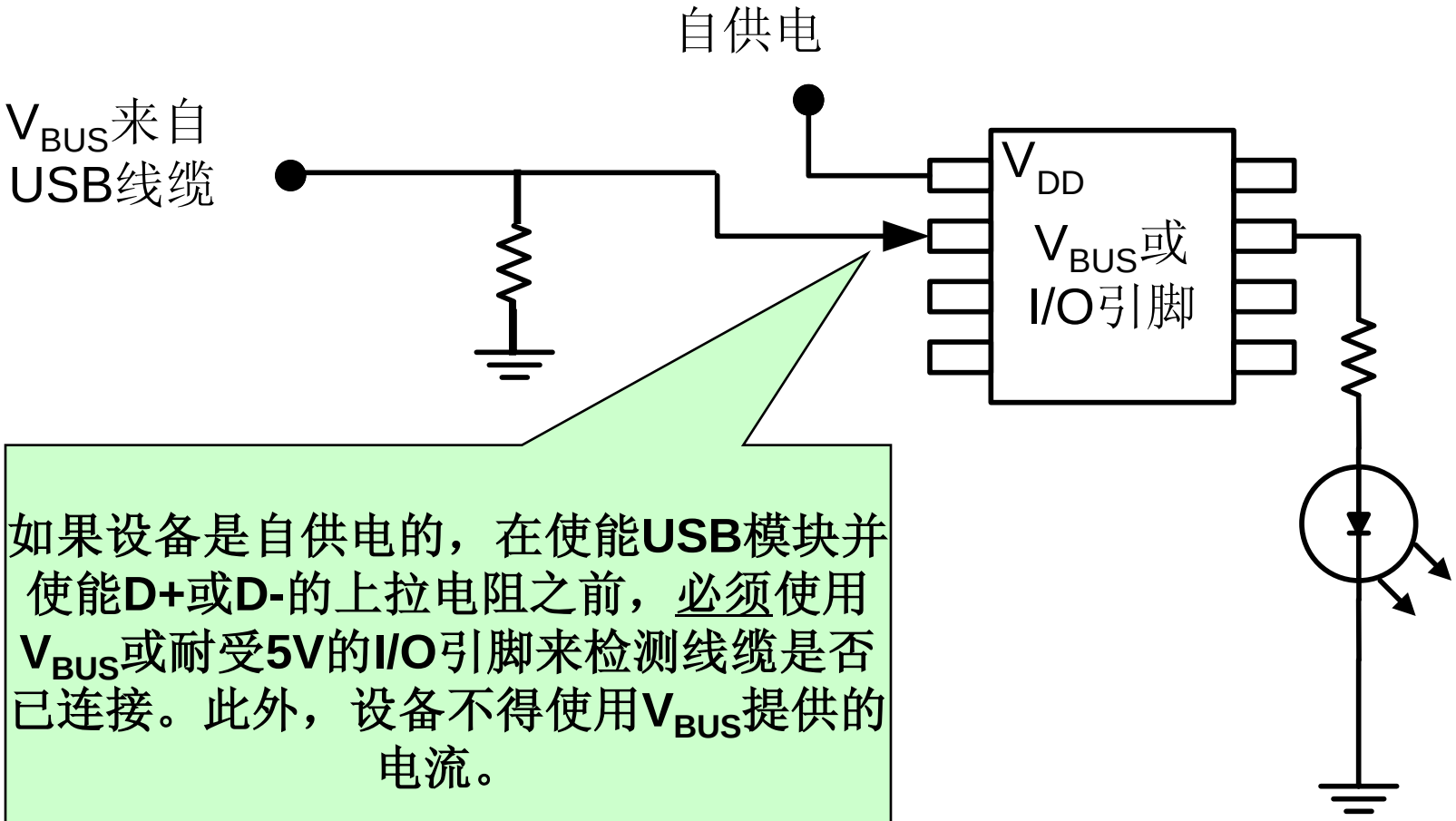
自动检测：低速



片上上拉电阻



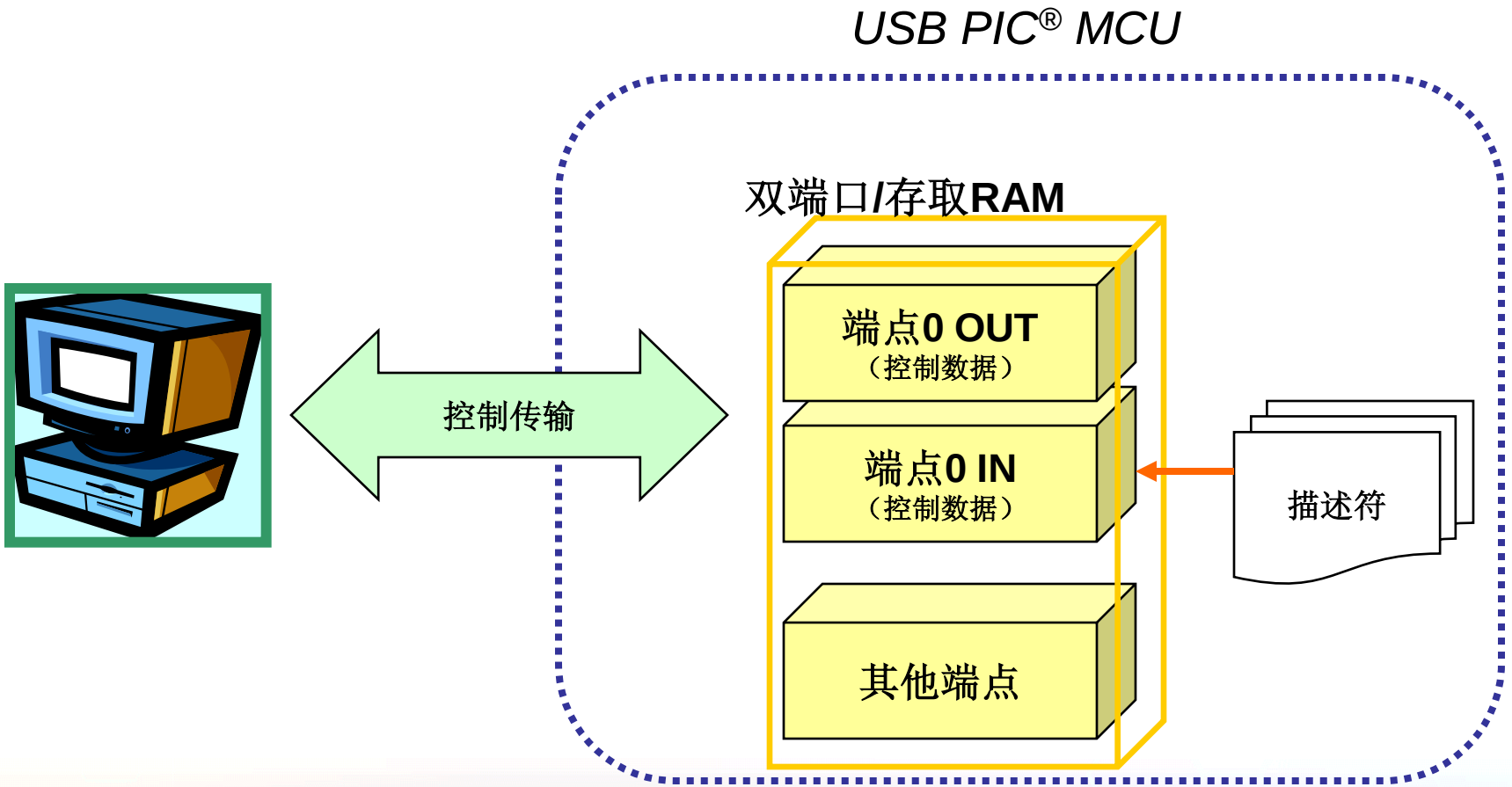
自供电设备
- 检测USB连接 -



如果设备是自供电的，在使能USB模块并使能D+或D-的上拉电阻之前，必须使用V_{BUS}或耐受5V的I/O引脚来检测线缆是否已连接。此外，设备不得使用V_{BUS}提供的电流。

地址和配置：EP0

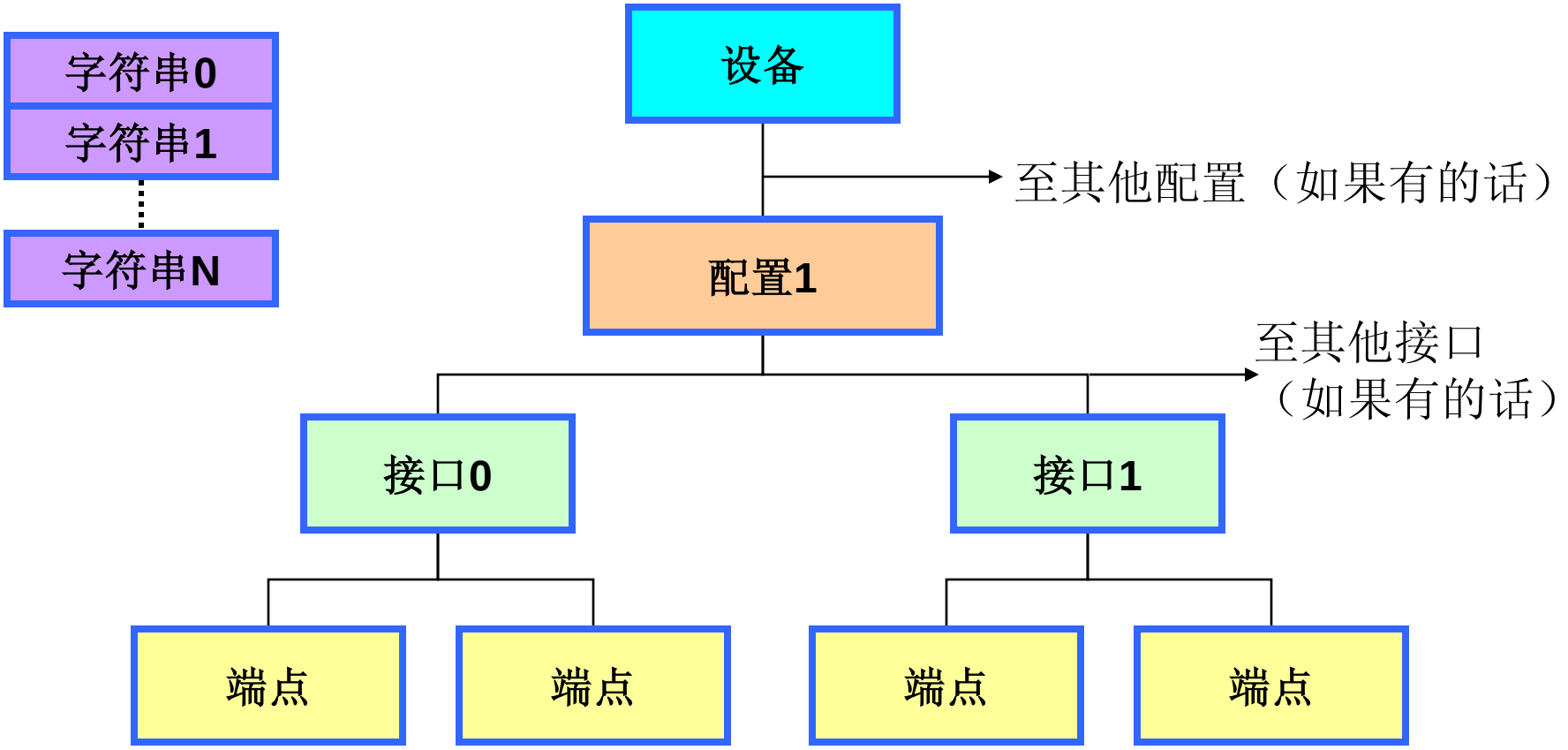
- 更多信息，请参阅USB 2.0规范的第9章。



课程安排

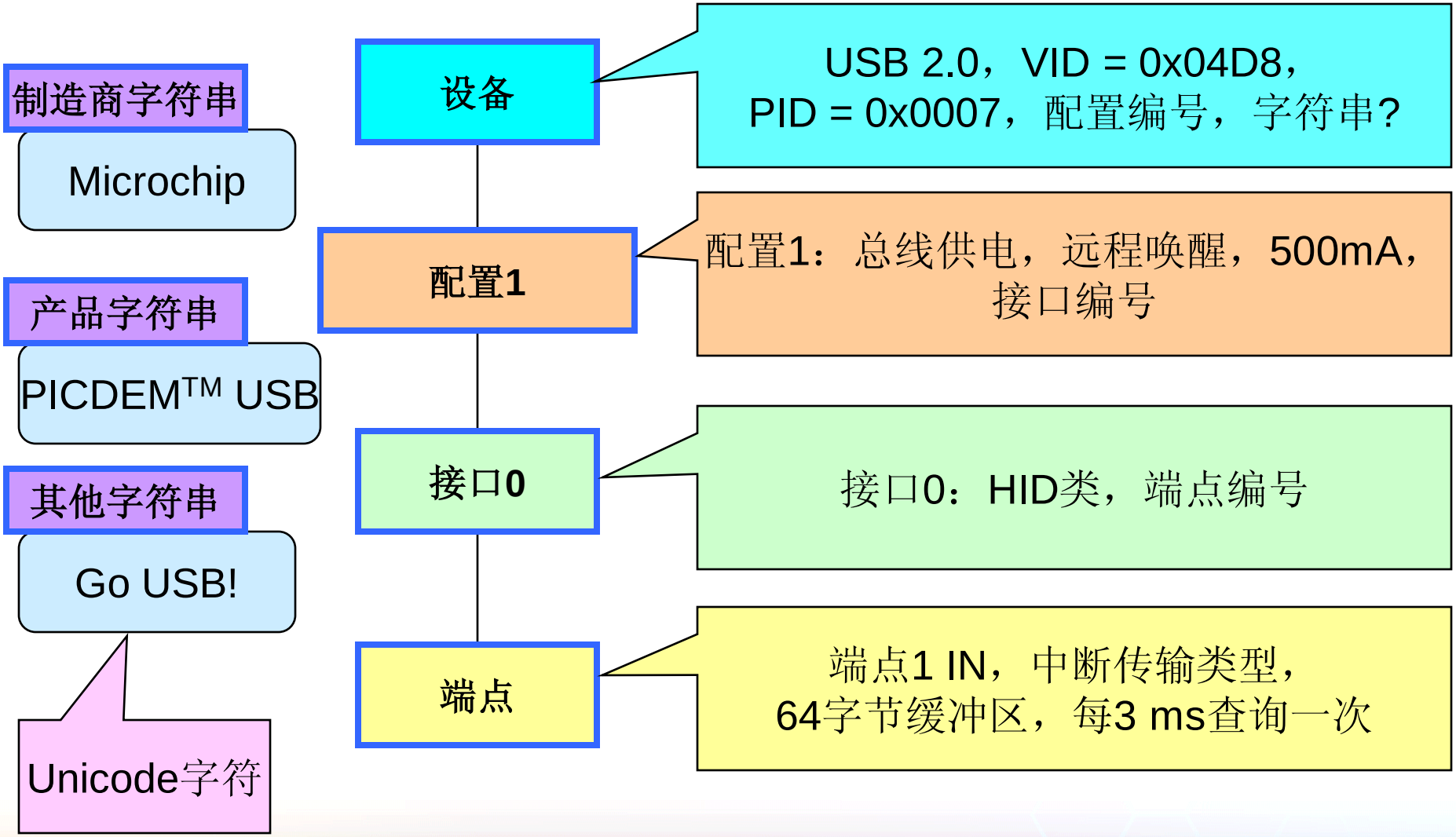
- 嵌入式设计人员面临的连通性挑战
- **USB基础 —— 严肃且重要的资料**
  - USB简史
  - 基础/速度
  - 拓扑/物理连接
  - 架构/程序员模型
  - USB事务
  - USB传输
  - 设备类
  - 枚举
  - 描述符
  - VID/PID及USB兼容性
- **PIC18/24/32 USB单片机**
- **Microchip演示/开发解决方案**
- 如何着手

描述符

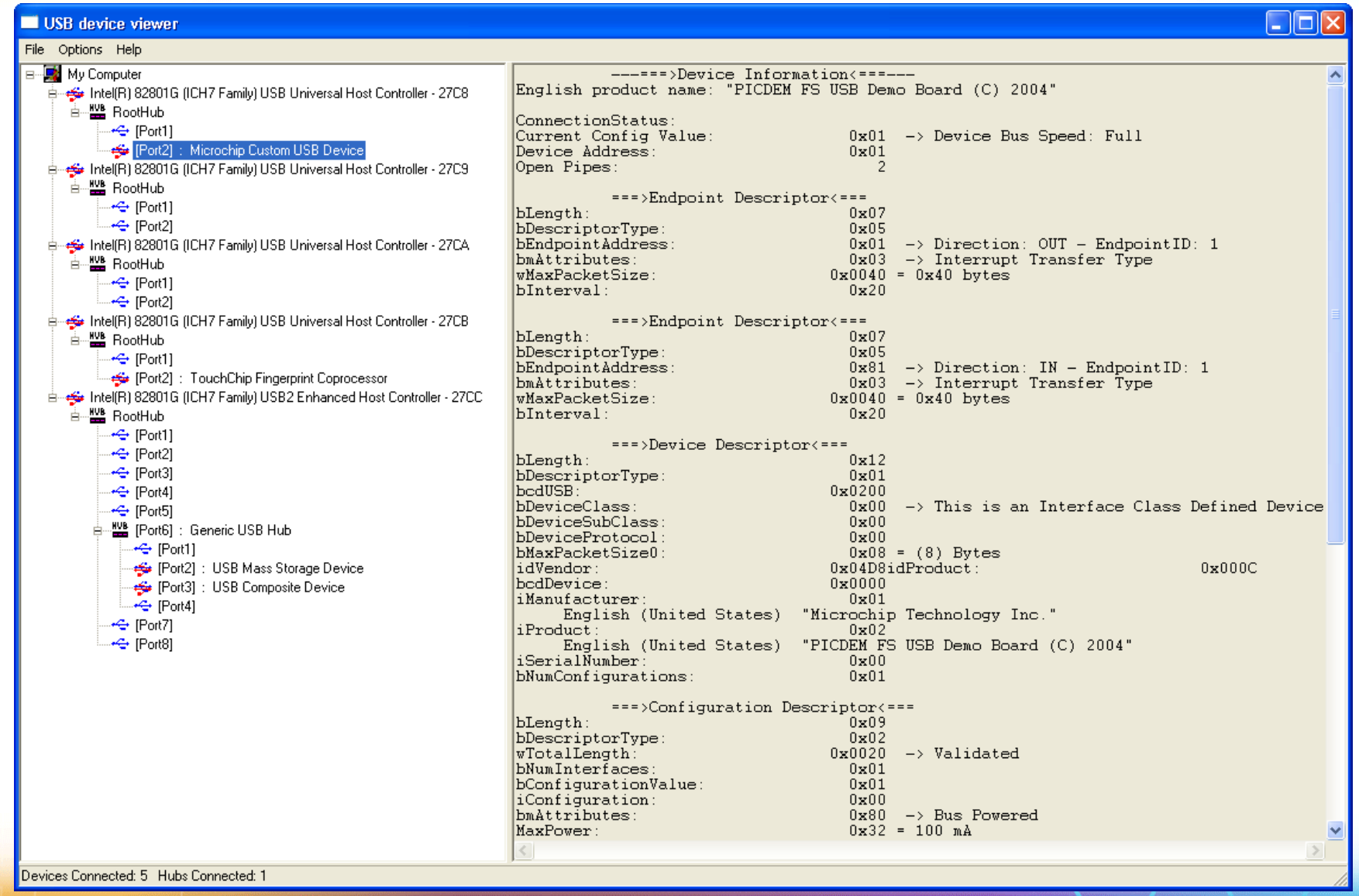


- 描述符通常存储在非易失性/闪存存储器中

描述符 —— 示例



演示
- 查看描述符信息 -



课程安排

- 嵌入式设计人员面临的连通性挑战
- **USB基础 —— 严肃且重要的资料**
  - USB简史
  - 基础/速度
  - 拓扑/物理连接
  - 架构/程序员模型
  - USB事务
  - USB传输
  - 设备类
  - 枚举
  - 描述符
  - VID/PID及USB兼容性
- **PIC18/24/32 USB单片机**
- **Microchip演示/开发解决方案**
- 如何着手

USB设备

- 供应商ID（VID）：*16位编号*
  - 销售产品必需的
  - http://www.usb.org/developers/vendor
  - **USD $2,000**
  - 如果不使用经过批准的VID的话，在技术和法律上会有麻烦
- 产品ID（PID）：*16位编号*
  - **Microchip的分许可计划（Sub-licensing Program）**
- 要求每条产品线都有惟一的VID和PID组合

USB兼容性

- 兼容性测试
  - 要使用USB徽标必须通过测试
  - 测试费用：USD ~$1,500

- 使用软硬件工具测试设备对USB协议的兼容性
  - USB Command Verifier
  - USB Electrical Analysis Tool
  - www.usb.org/developers/tools
- 通过相似性进行资格认定
- OEM协议

兼容性测试

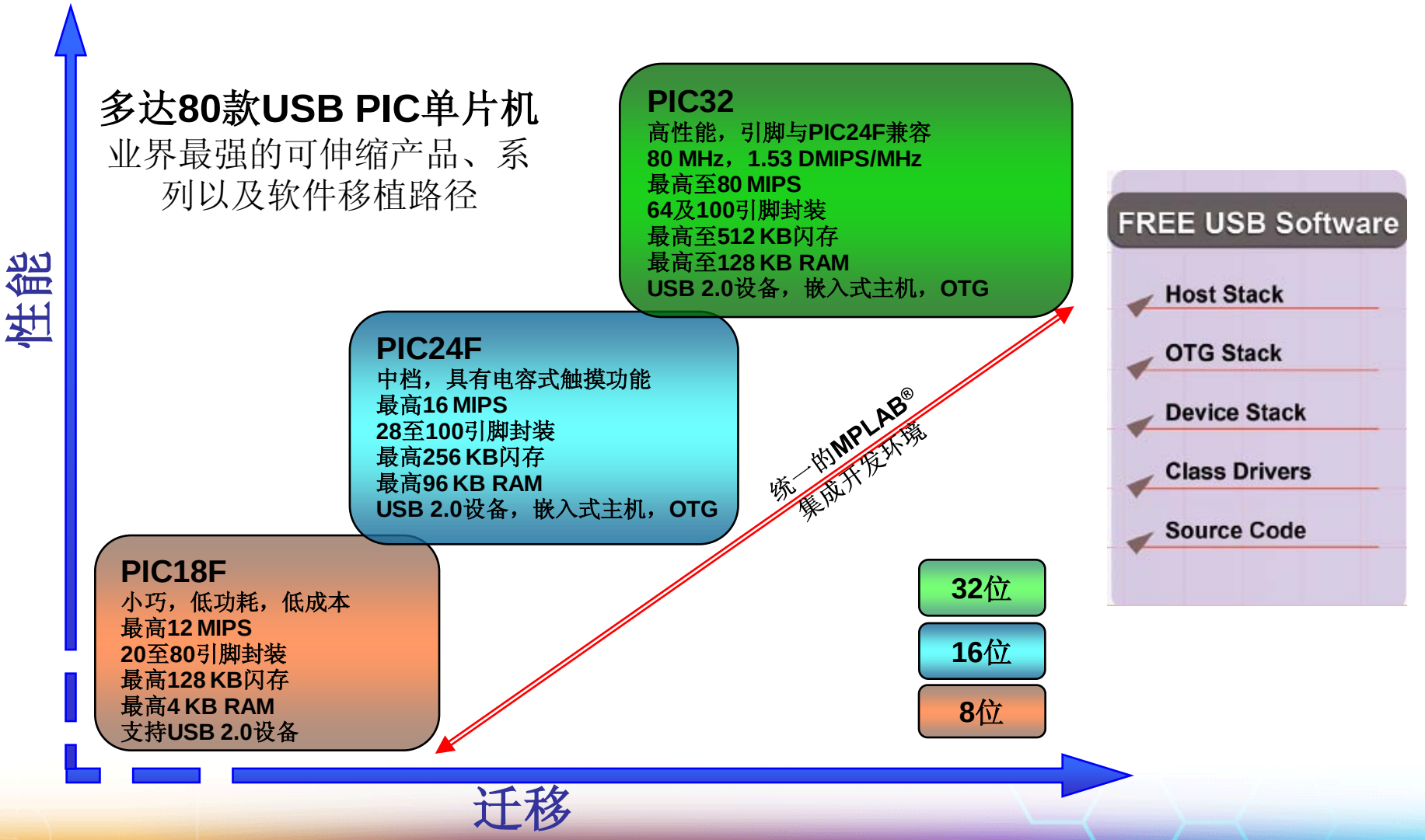
- 对于**USB**兼容性认证：
  - 独立的测试实验室
    - www.usb.org/developers/compliance/labs
- 对于**USB**兼容性测试：
  - 下载兼容性检查表
    - www.usb.org/developers/compliance/check_list
  - 测试时使用经过认证的**USB**母头和线缆
  - 知道您器件的**TID**（**Test ID**）
    - 找出USB PIC®单片机TID号，地址 www.microchip.com/usb

甚至在您开始设计之前，最好看看检查表！

课程安排

- 嵌入式设计人员面临的连通性挑战
- **USB基础 —— 严肃且重要的资料**
  - USB简史
  - 基础/速度
  - 拓扑/物理连接
  - 架构/程序员模型
  - USB事务
  - USB传输
  - 设备类
  - 枚举
  - 描述符
  - VID/PID及USB兼容性
- **PIC18/24/32 USB单片机**
- **Microchip演示/开发解决方案**
- 如何着手

可伸缩的USB PIC®单片机产品线



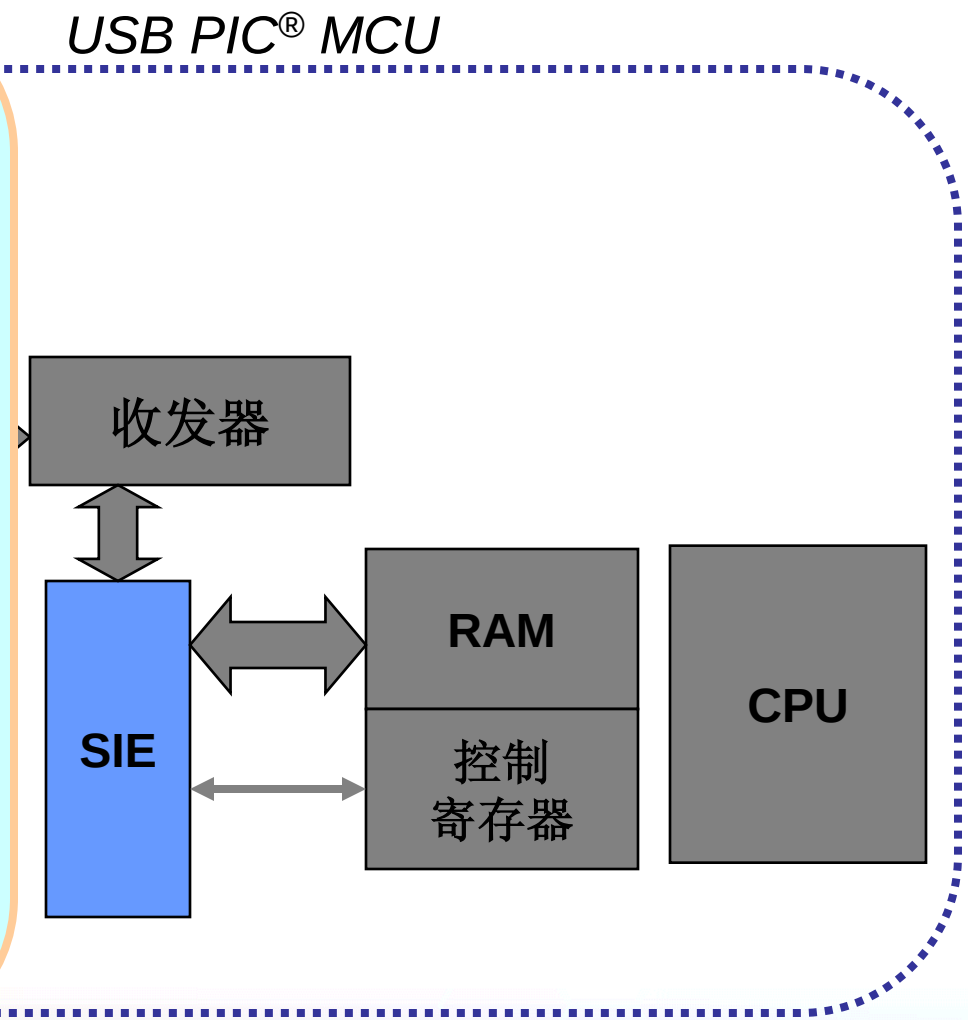
USB单片机产品线

	PIC18FXXK50	PIC18FXX5x	PIC18FXXJ5x	PIC24FJ GB系列 PIC24FJ DA系列	PIC32MX4系列 PIC32MX5系列 PIC32MX6系列 PIC32MX7系列
内核	8位	8位	8位	16位	32位
USB	USB 2.0设备	USB 2.0设备	USB 2.0设备	USB 2.0设备，嵌入式主机，双重角色，OTG	USB 2.0设备，嵌入式主机，双重角色，OTG
闪存	最高16 KB	最高32 KB	最高128 KB	最高256 KB	最高512 KB
RAM	最高768B	最高2048B	最高3904B	最高96 KB	最高128 KB
mTouch™技术支持	是，CVD	是，CVD	是， CVD或CTMU	是， CVD或CTMU	是，CVD
UART	1	1	2	最多4个	最多6个
SPI	1	1	2	最多3个	最多4个
I^2C™	1	1	2	最多3个	最多5个
外设引脚选择	无	无	有些有	有	无
ADC	10位，9通道	最高12位， 最多13个通道	最高12位， 最多13个通道	10位，最多24个通道	10位，16个通道
HW RTCC	无	无	有些有	有（除128DA110外）	有
并行主端口	无	无	有些有	有些有	有
模拟比较器	最多2个	2	2	3	2
免费的软件栈	有	有	有	有	有
免费的类驱动程序	有	有	有	有	有
可伸缩的开发环境	有	有	有	有	有
封装	20引脚	28、40和44引脚	28、44、64和80引脚	28、44、64、80、100和121引脚	64、100和121引脚

串行接口引擎（SIE）

SIE...

- 串行化及解串行USB数据
- 编/解码NRZI数据
- 处理位填充
- 检查CRC，以验证数据包
- 检测总线信号通知的事件，并通过中断告知CPU
- 处理USB事务
- 处理握手协议



课程安排

- 嵌入式应用设计人员面临的连通性挑战
- **USB基础 —— 严肃且重要的资料**
  - USB简史
  - 基础/速度
  - 拓扑/物理连接
  - 架构/程序员模型
  - USB事务
  - USB传输
  - 设备类
  - 枚举
  - 描述符
  - VID/PID及USB兼容性
- **PIC18/24/32 USB单片机**
- **Microchip演示/开发解决方案**
- 如何着手

PICDEM™全速USB演示工具包

- 含有您迅速着手所需要的一切
  - 可配合使用任何**PIC18F4550**系列单片机
- 包括供自学的课程和实验资料
- 演示套件提供用来演示、开发完整**USB**通信解决方案所需的全部硬件和软件。
- 部件编号
  - **DM163025**
- 现已供货

PIC18FXXJ50全速USB接插模块（PIM）

- 含有您迅速着手所需要的一切
  - 可配合使用任何**PIC18F87J50**或**PIC18F46J50**系列单片机
- 可插入**PICDEM™ HPC Explorer**板或**PICDEM PIC18 Explorer**板
- 可单独运行
- 部件编号
  - **MA180021 — PIC18F87J50 FS USB PIM**
  - **MA180024 — PIC18F46J50 FS USB PIM**
  - **DM183022 — PICDEM HPC EXPLORER**板
  - **DM183032 — PICDEM PIC18 EXPLORER**板
- 现已供货

PIC18入门工具包

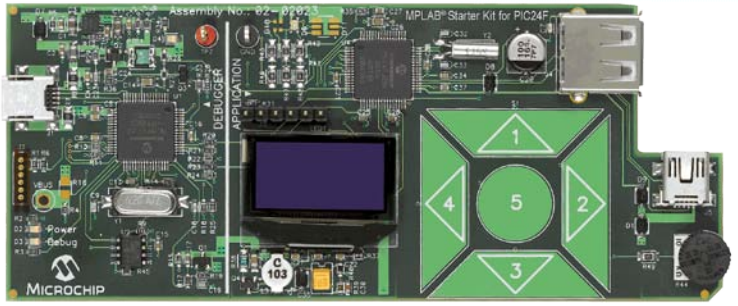
- 可用作USB鼠标、游戏杆或海量存储设备，全部使用板载电容式触摸感应衬垫
- 包括一块MicroSD™存储卡、电位器、加速度传感器以及OLED显示屏
- 板载调试器/编程器
- 完全USB供电
- 演示PIC18系列
  - USB通信
- 部件编号
  - DM180021
- 现已供货

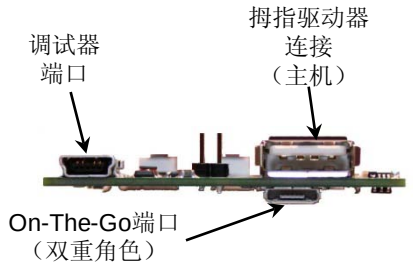
低引脚数USB开发工具包

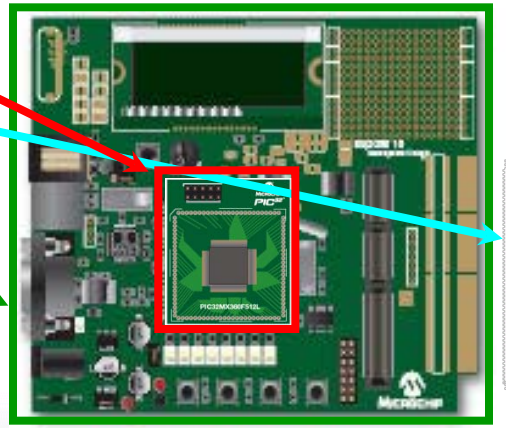
- 含有您迅速着手所需的一切
  - 可配合使用新款20引脚PIC18F USB单片机 —— PIC18F13K50和PIC18F14K50
- 包括供自学的课程和实验资料
- 快速实现常见的USB功能：
  - RS-232至串口
  - 键盘/鼠标，等等…
- 部件编号
  - DV164126（带PICkit™ 2）
  - DM164127
- 现已供货

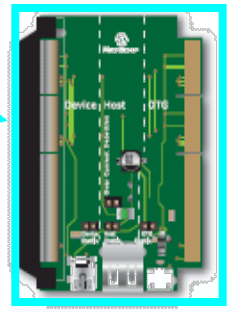
16/32位USB开发板

- PIC24F入门工具包1
  - 部件编号：DM240011
  - PIC24FJ256GB110
- PIC32 USB入门工具包II
  - 部件编号：DM320003-2
  - PIC32MX795F512L
- Explorer 16 + USB PICtail™ Plus 子板 + USB PIM
  - 部件编号：MA320002/MA240014
  - 部件编号：AC164131
  - 部件编号：DM240001
- 全部现已供货





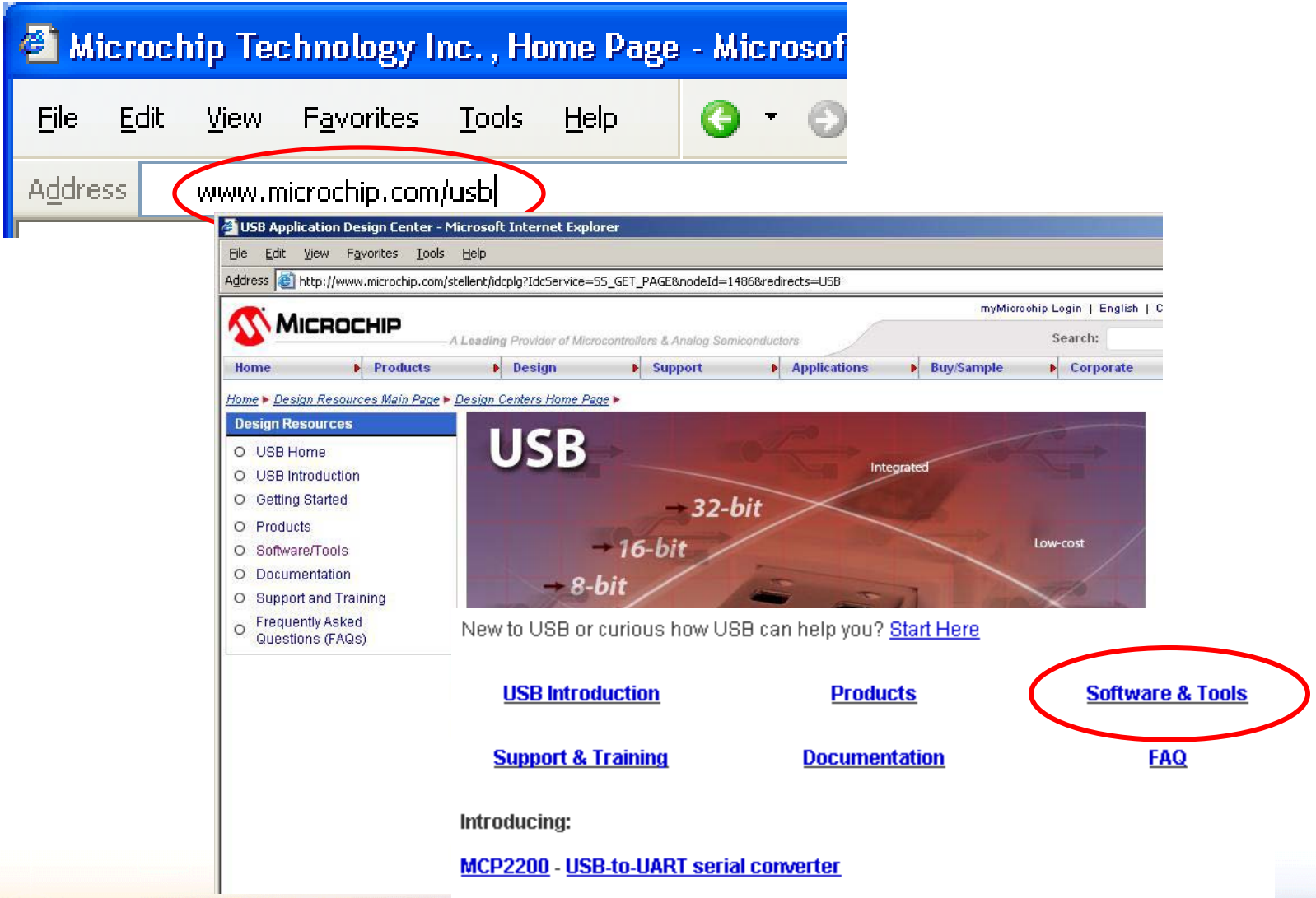




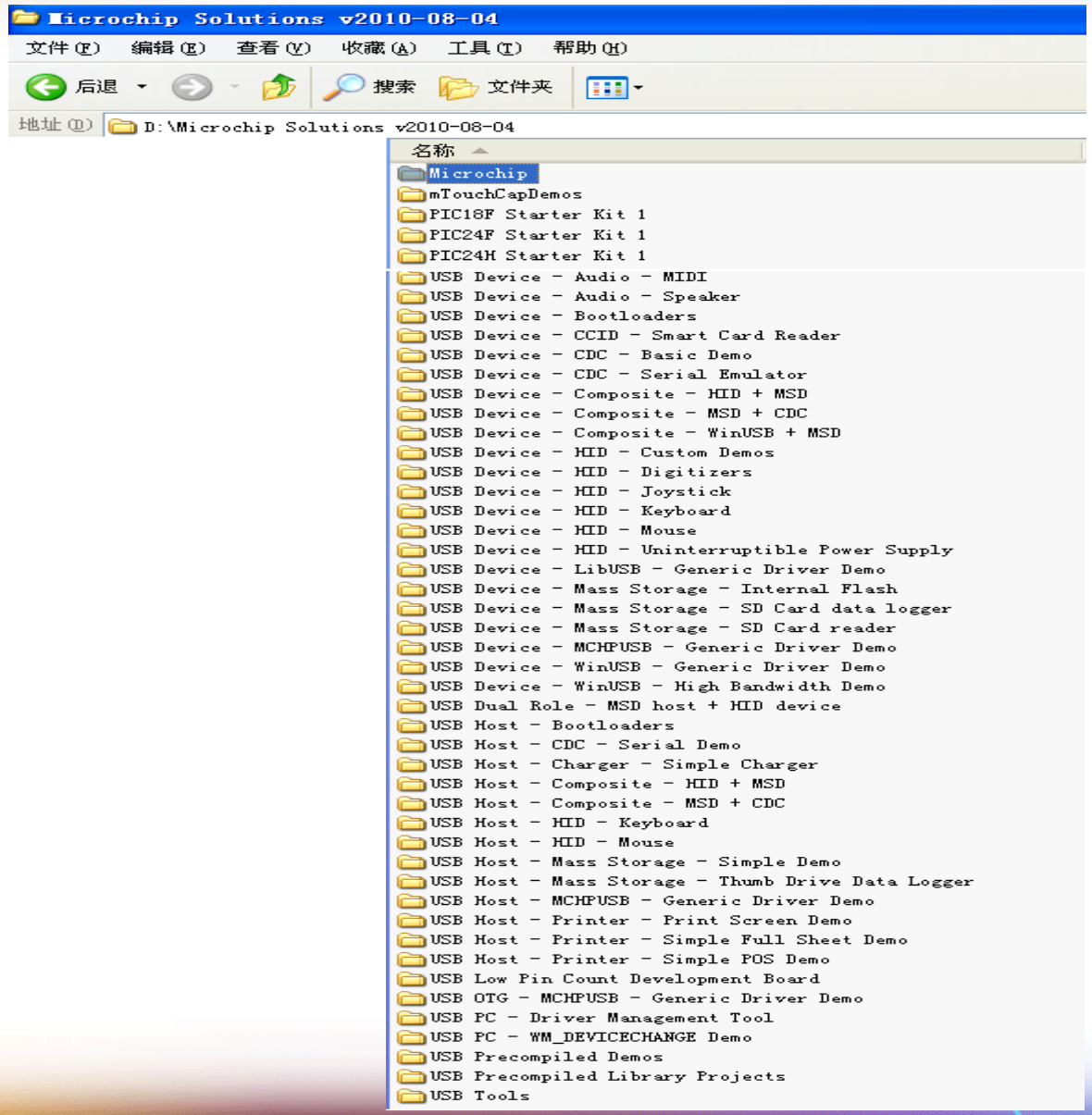
课程安排

- 嵌入式应用设计人员面临的连通性挑战
- **USB基础 —— 严肃且重要的资料**
  - USB简史
  - 基础/速度
  - 拓扑/物理连接
  - 架构/程序员模型
  - USB事务
  - USB传输
  - 设备类
  - 枚举
  - 描述符
  - VID/PID及USB兼容性
- **PIC18/24/32 USB单片机**
- **Microchip演示/开发解决方案**
- 如何着手

下载Microchip USB框架



安装目录



Microchip USB框架
- www.microchip.com/mal -

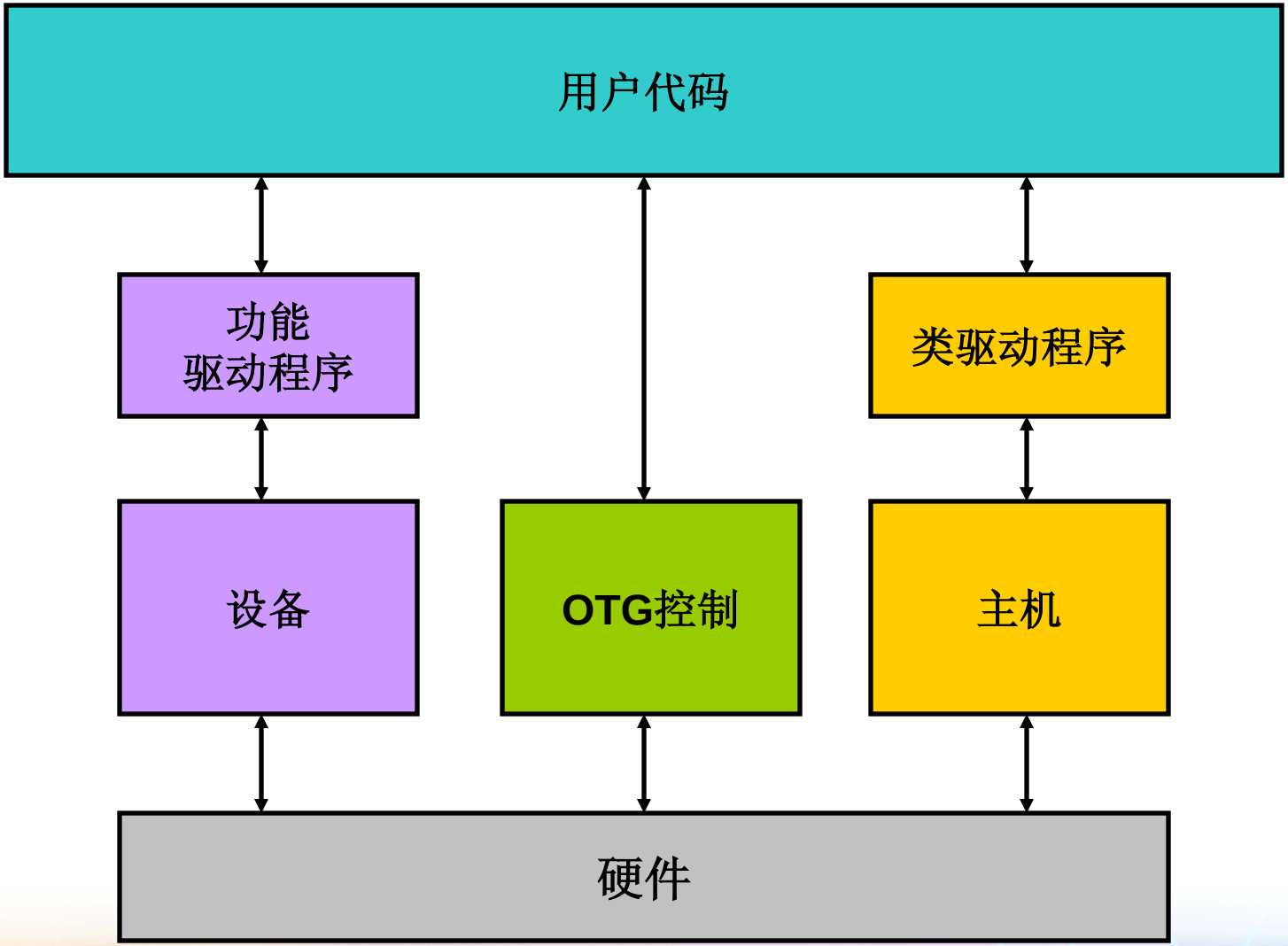
- MCHPFSUSB框架
  - PIC18F、PIC24F和PIC32 USB单片机
  - C18/C30/C32兼容
  - MPLAB® IDE项目中心
- 设备栈
  - 音频、CCID、HID、CDC、MSD和定制
  - 轮询或中断驱动
- 嵌入式主机栈
  - PIC24F和PIC32 USB单片机
  - 轮询或事件驱动机制
  - 用于CDC、充电器、定制、HID、MSD和打印机的客户机驱动程序
- On-The-Go（OTG）支持
  - PIC24F和PIC32 USB单片机

MCHPFSUSB v2.x框架

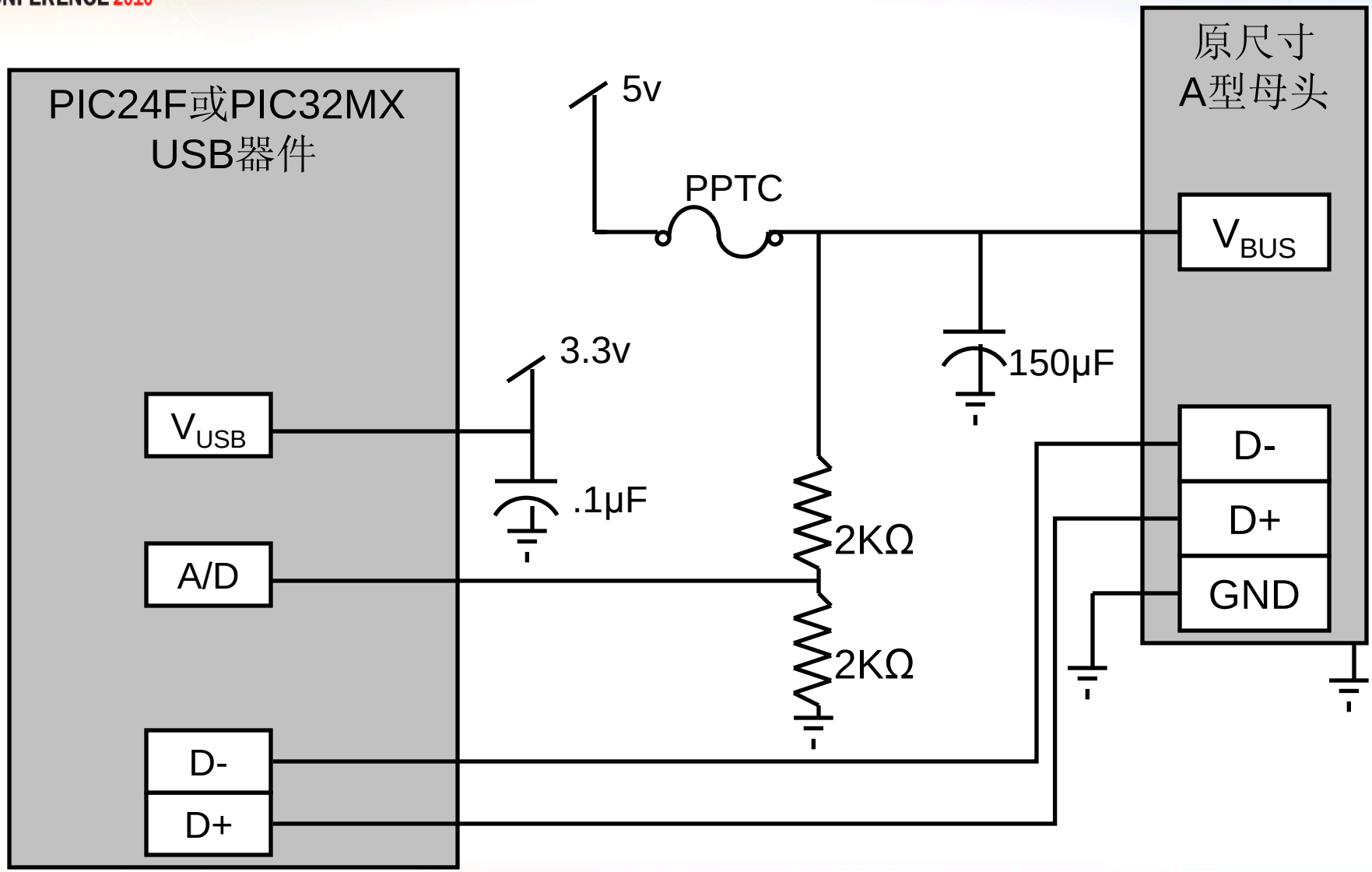
- 一般结构

```
/Your application
      /main.c
      /usb_descriptors.c
      /HardwareProfile.h
      /usb_config.h
/Microchip
      /Include
      /USB
      /TCPIP
      /Graphics …
```

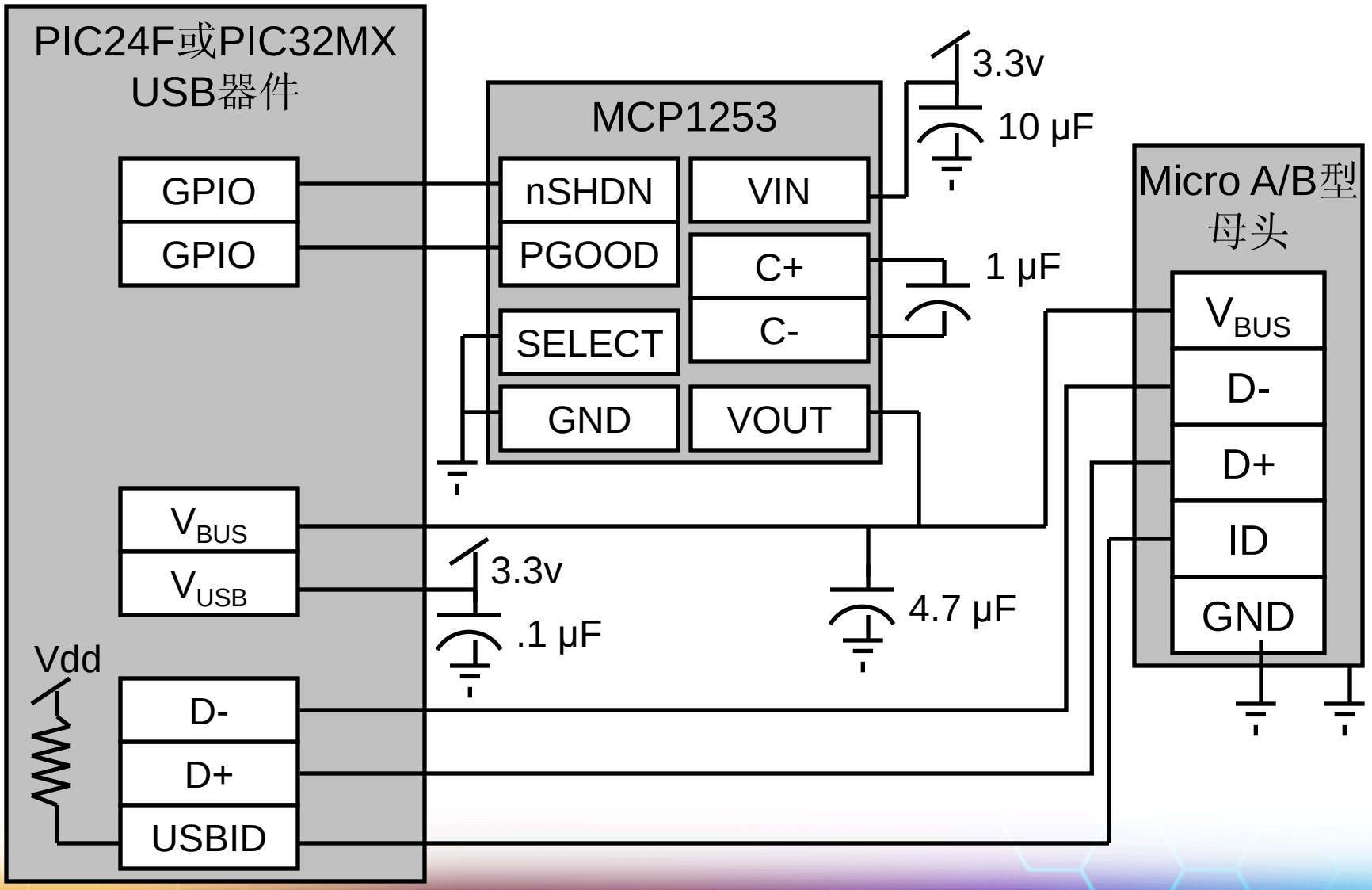
软件架构



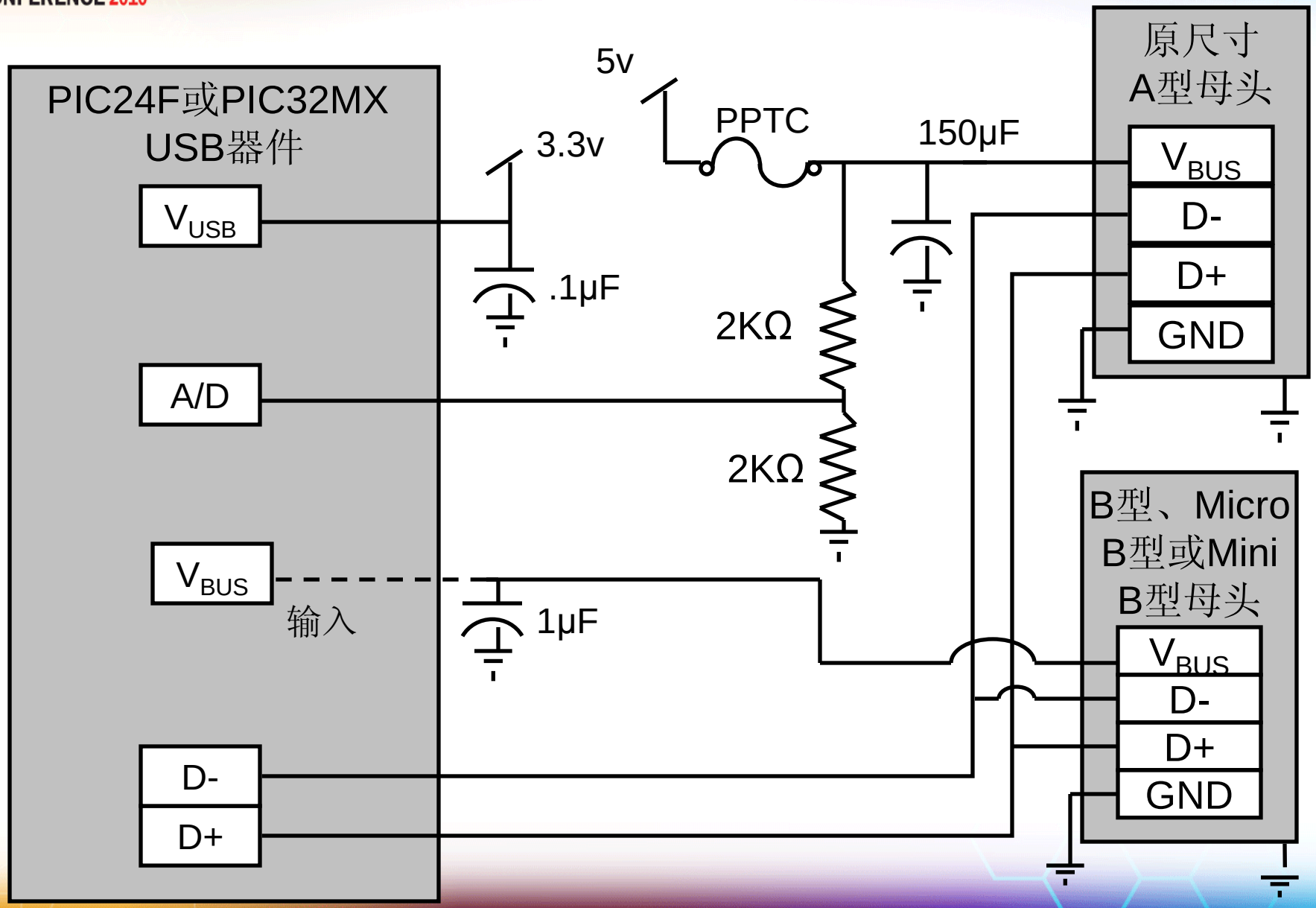
嵌入式主机电路示例



OTG电路示例



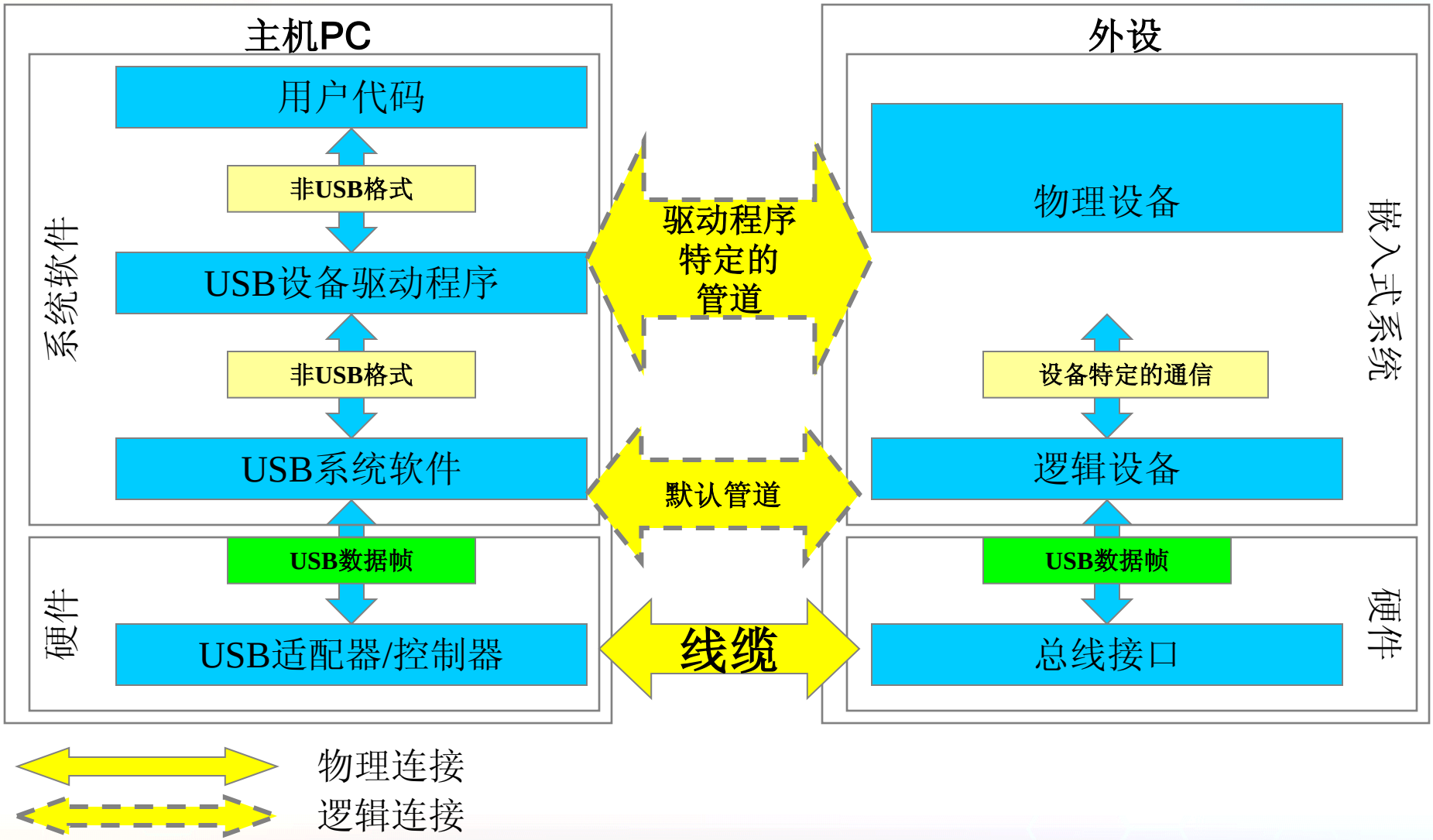
DRD电路示例



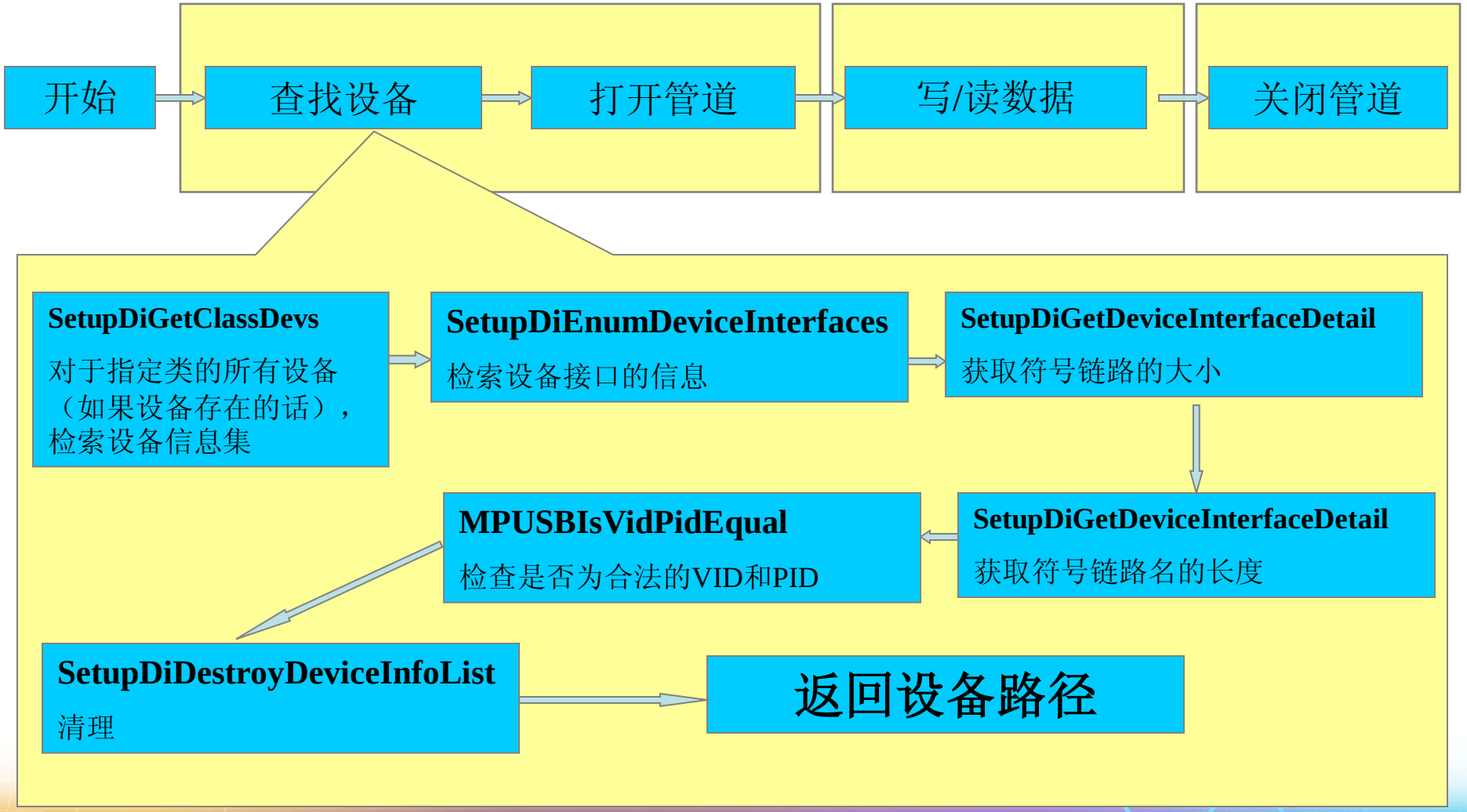
课程安排

- 双向USB通信工作原理
- **VB.Net USB类以及示例应用**
  - 函数释义
  - 单点及多点获取
  - 连续数据获取
  - USB描述符获取
    - **检索设备制造商序列号**
    - **检索设备产品字符串**
    - **检索设备制造商字符串**
  - WM_DEVICECHANGE事件（设备枚举）
  - 通过窗体实例进行跟踪
  - 通过序列号进行跟踪
- **Microchip演示/开发解决方案**
- **双向USB通信工作原理**

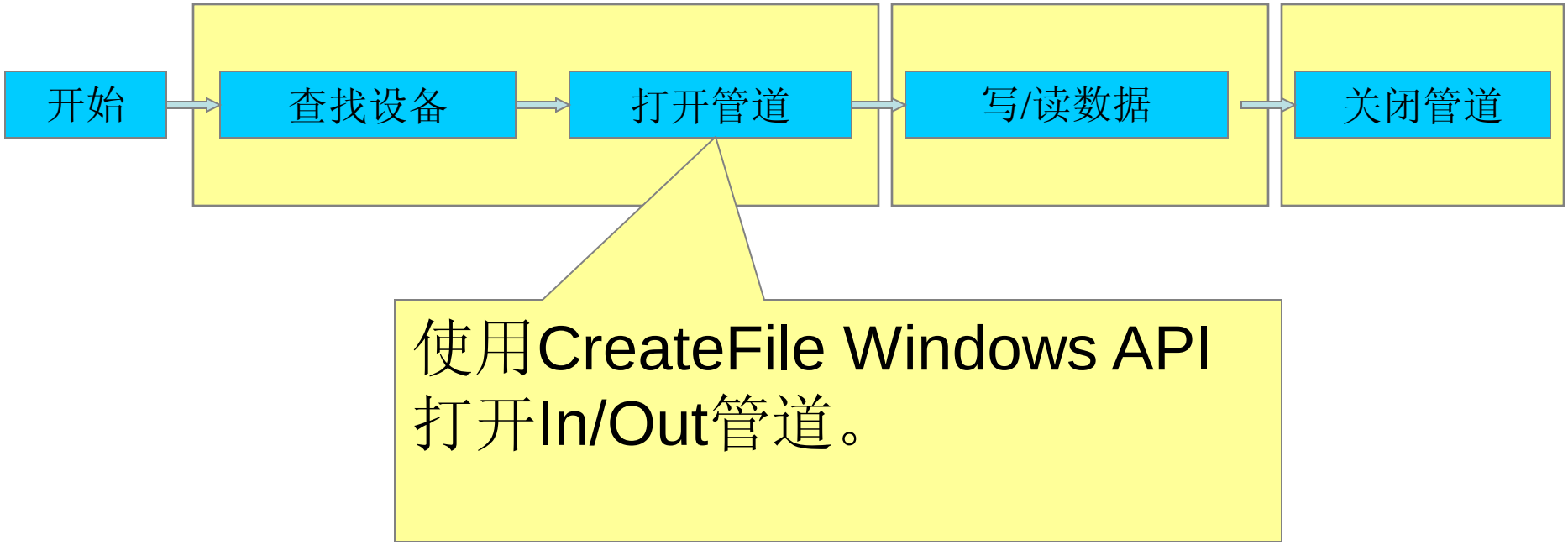
Windows侧



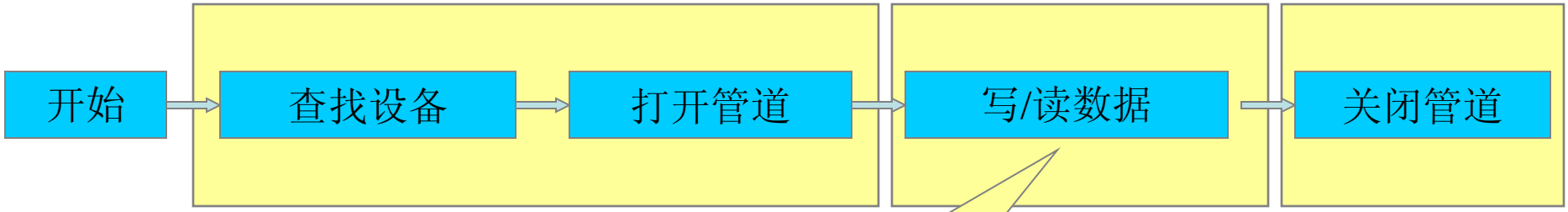
Windows USB 发送/接收数据包



Windows USB 发送/接收数据包



Windows USB 发送/接收数据包

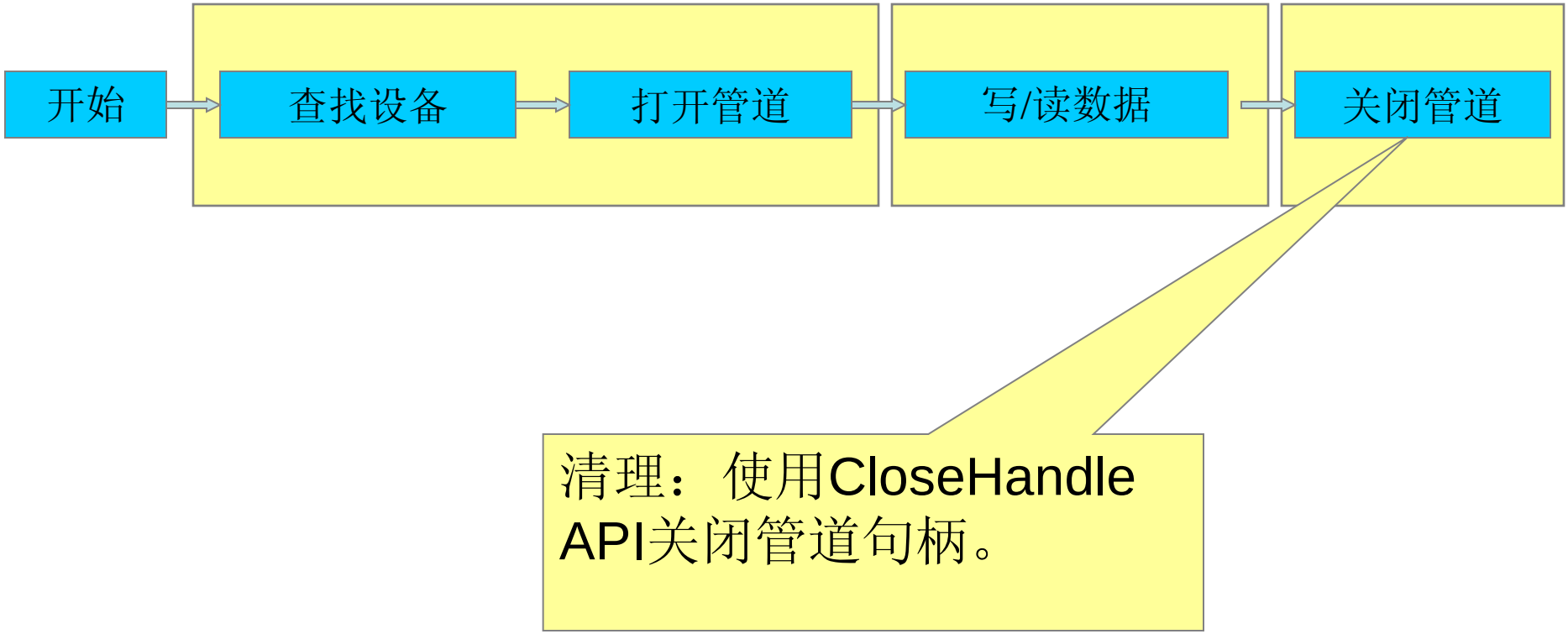


用于通信的标准文件IO Windows API是：

WriteFile和ReadFile

注：ReadFile自动生成IN令牌包。

Windows USB 发送/接收数据包



HID还是定制设备？

HID与定制设备比较

特性	定制设备	HID
是否需要驱动程序	是	否
Windows API	是	是
定制的API	是	否
定制驱动程序	是	否
传输类型		
控制	是	是
中断	是	是
批量	是	否
同步	是	否
最大速度	1.5 MB/s	64 KB/s

CDC的缺点

- **CDC概述**
  - 使用COM端口作为接口
    - **任何且每一应用都可与COM端口进行通信**
    - **COM端口号变动**
  - 不能定制

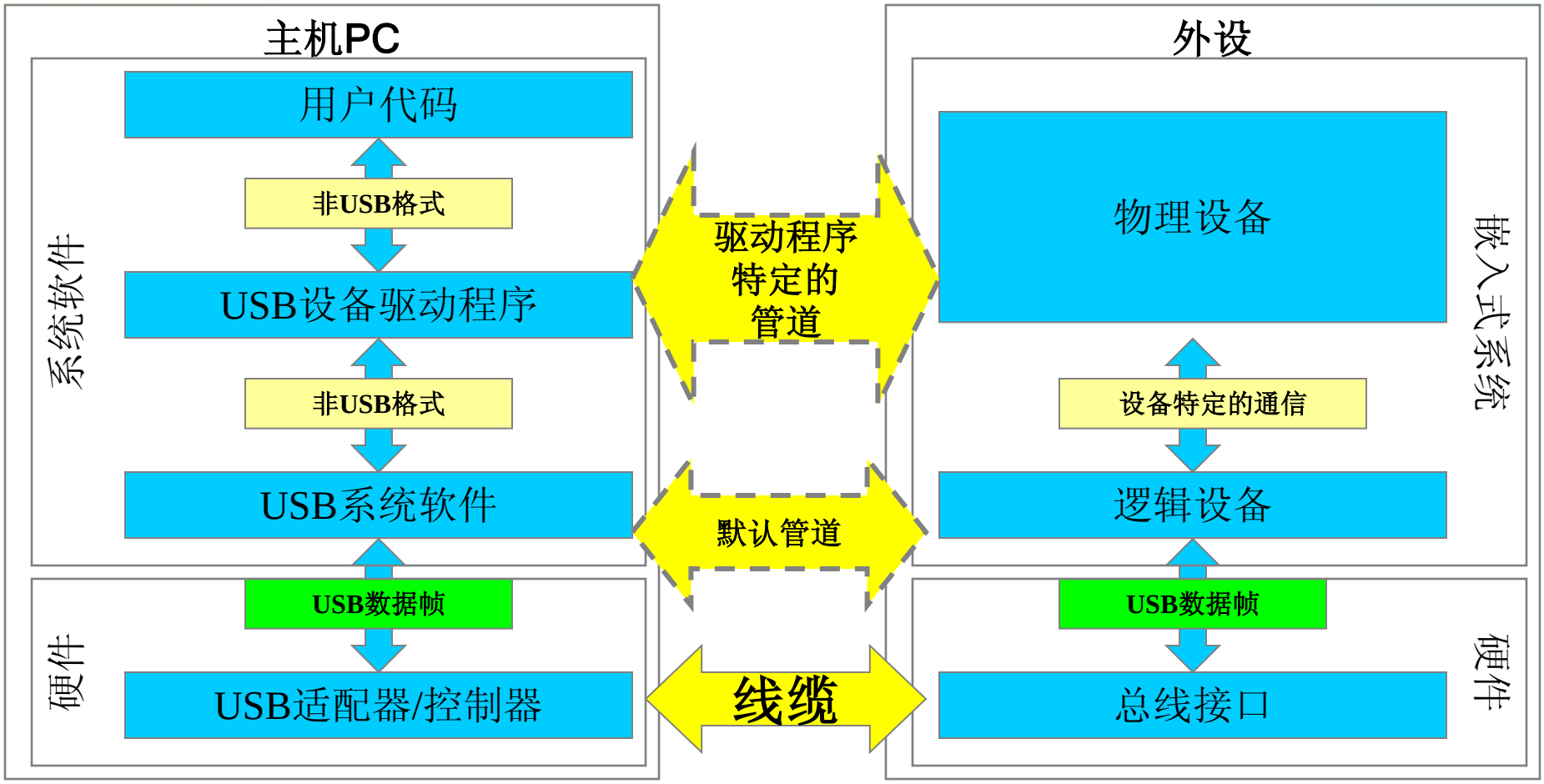
MASTERs CONFERENCE 2010

使用VB.Net，创建定制的USB系统

Windows程序设计

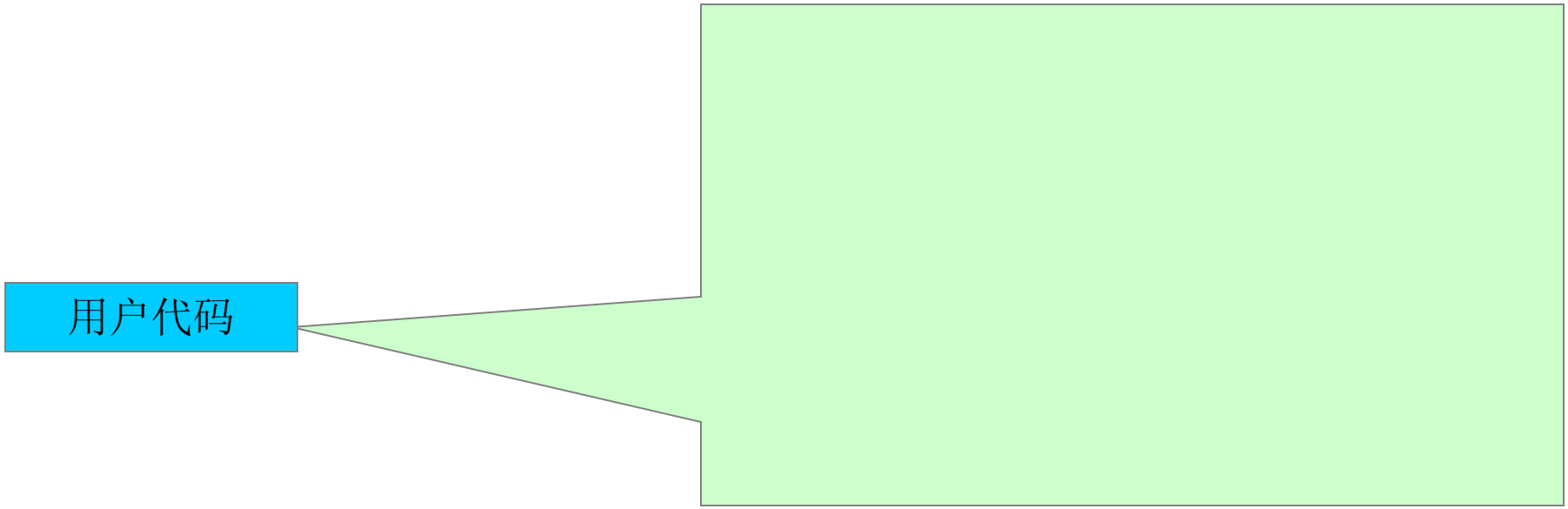
- **USB Communicator概述**
  - 用VB.Net编写
  - 基于类的应用
  - 多线程

Windows侧

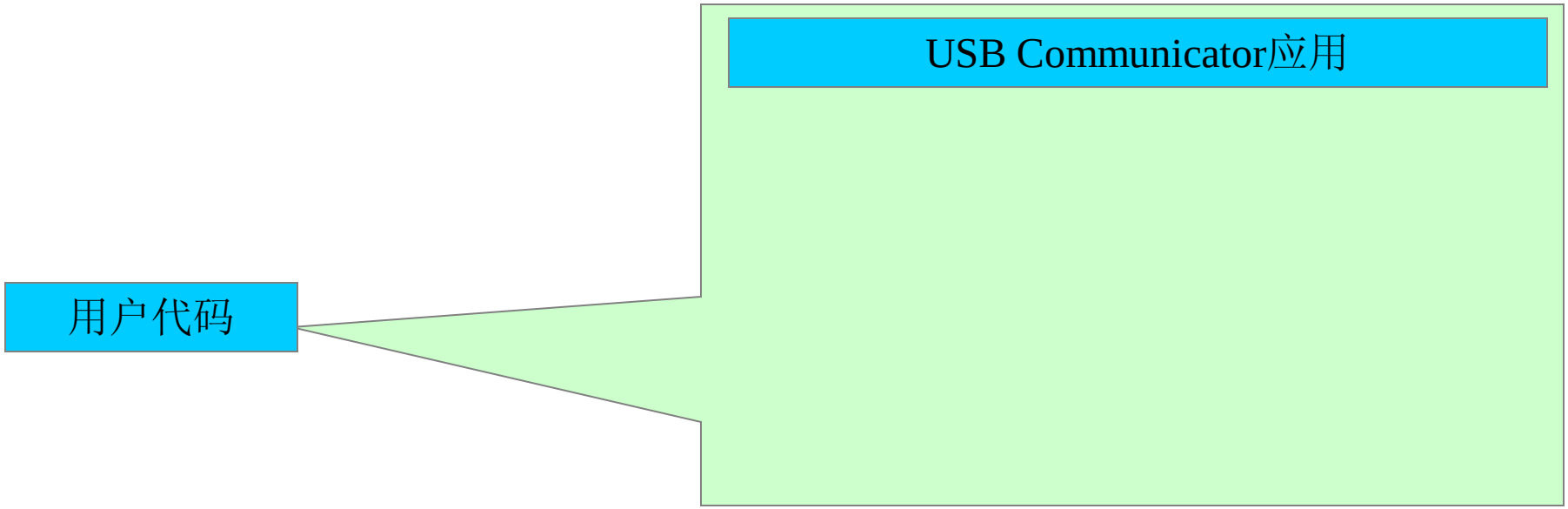


参考：http://www.microsoft.com/technet/prodtechnol/wce/support/usbce.mspx

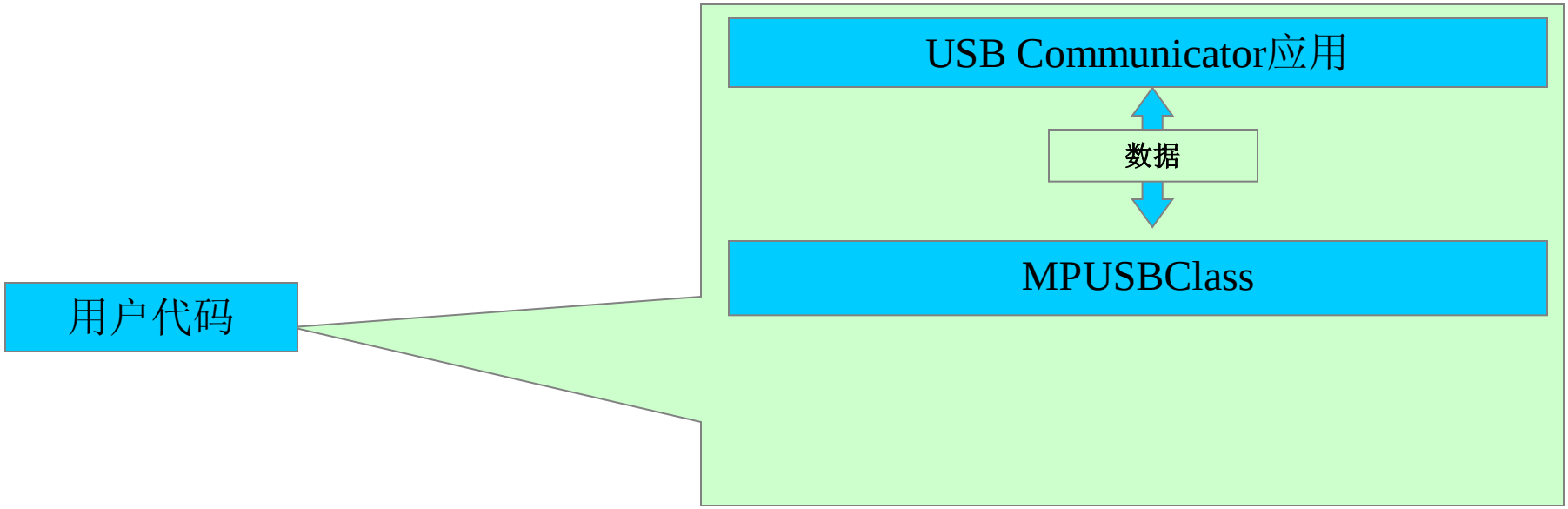
Windows侧



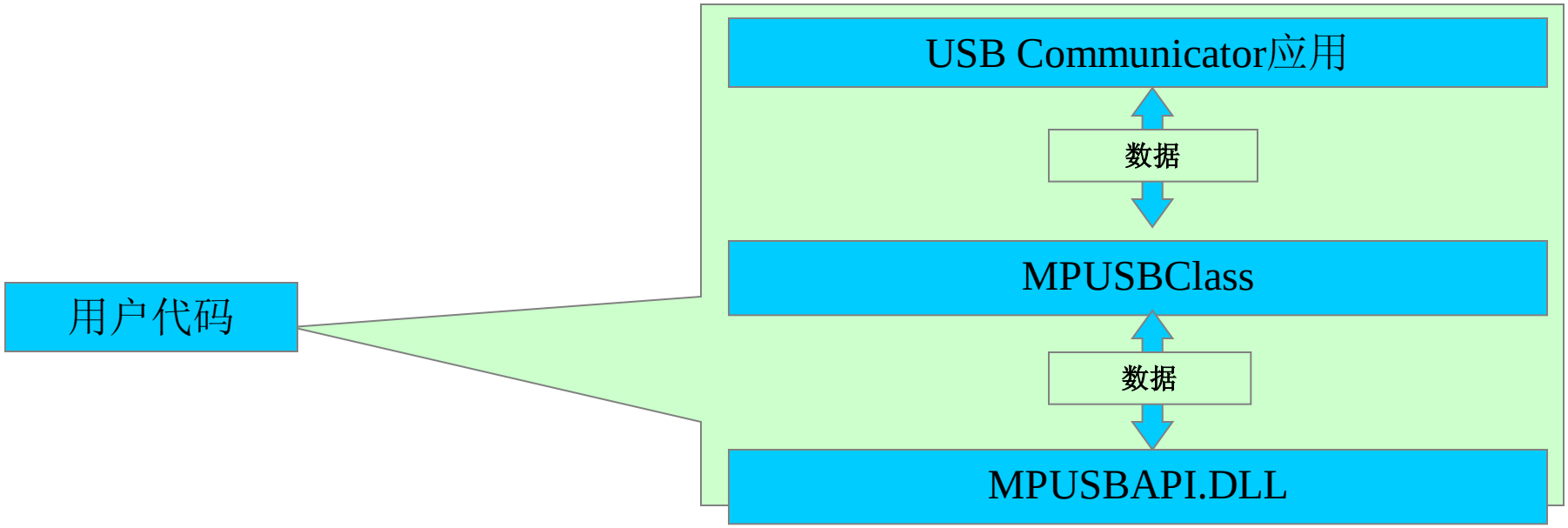
Windows侧



Windows侧

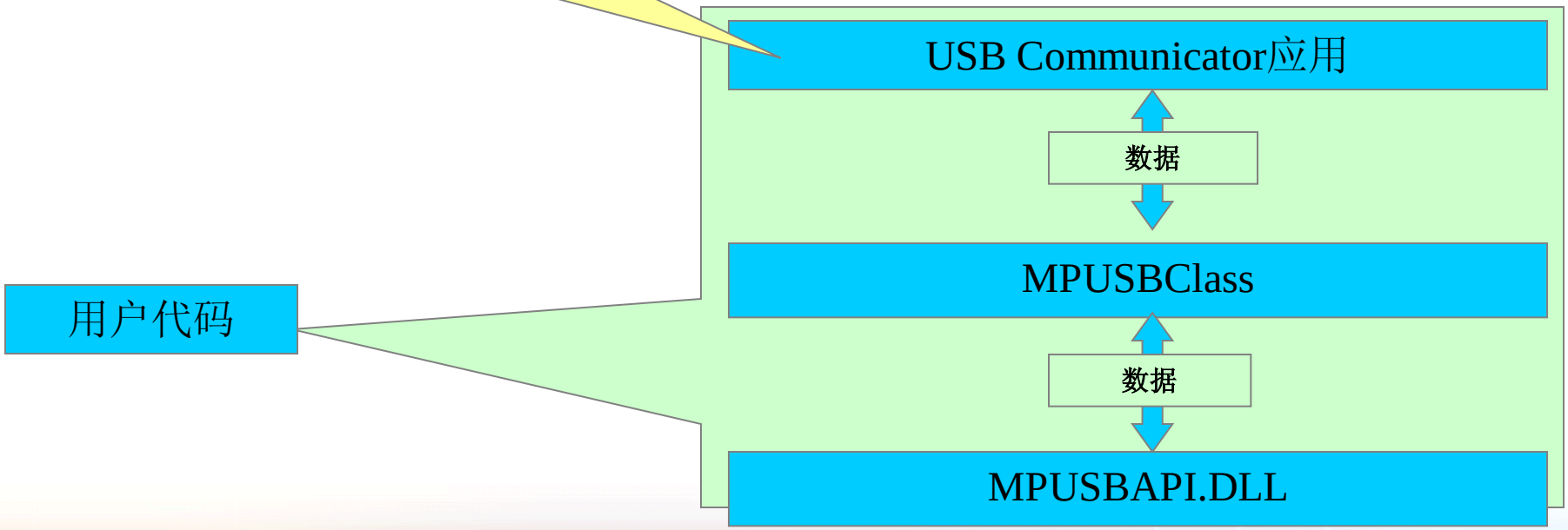


Windows侧



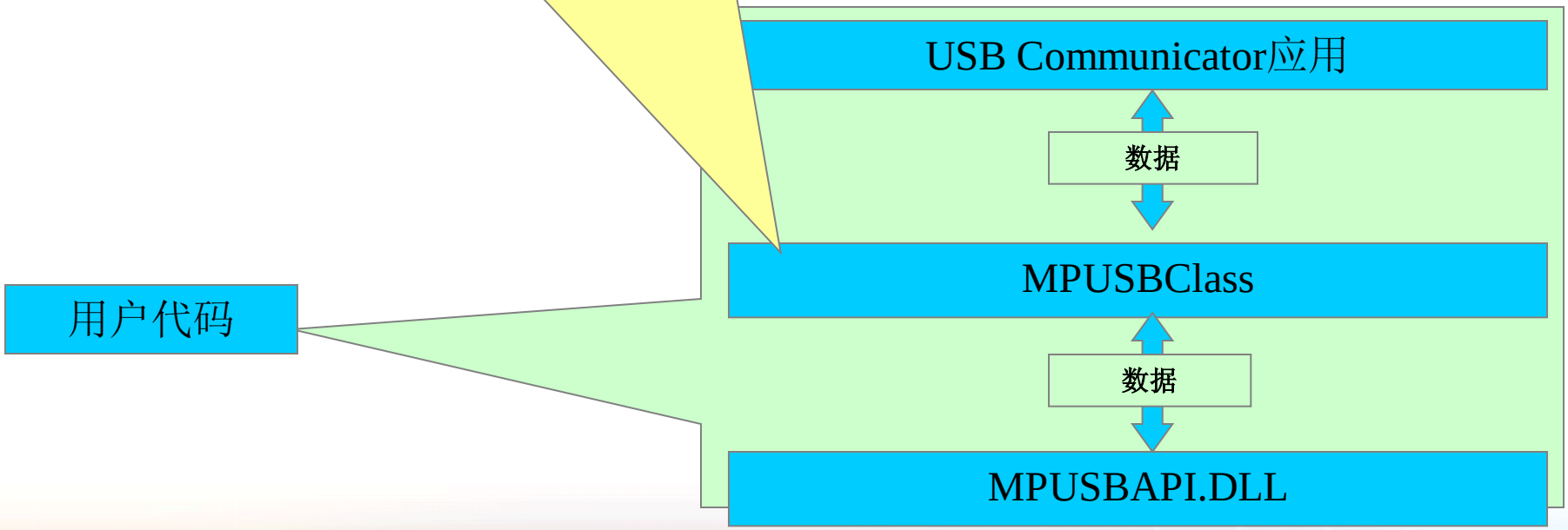
Windows侧

图形用户界面
数据处理
回调处理
响应Windows事件
WM_DEVICECHANGE
所有的决策



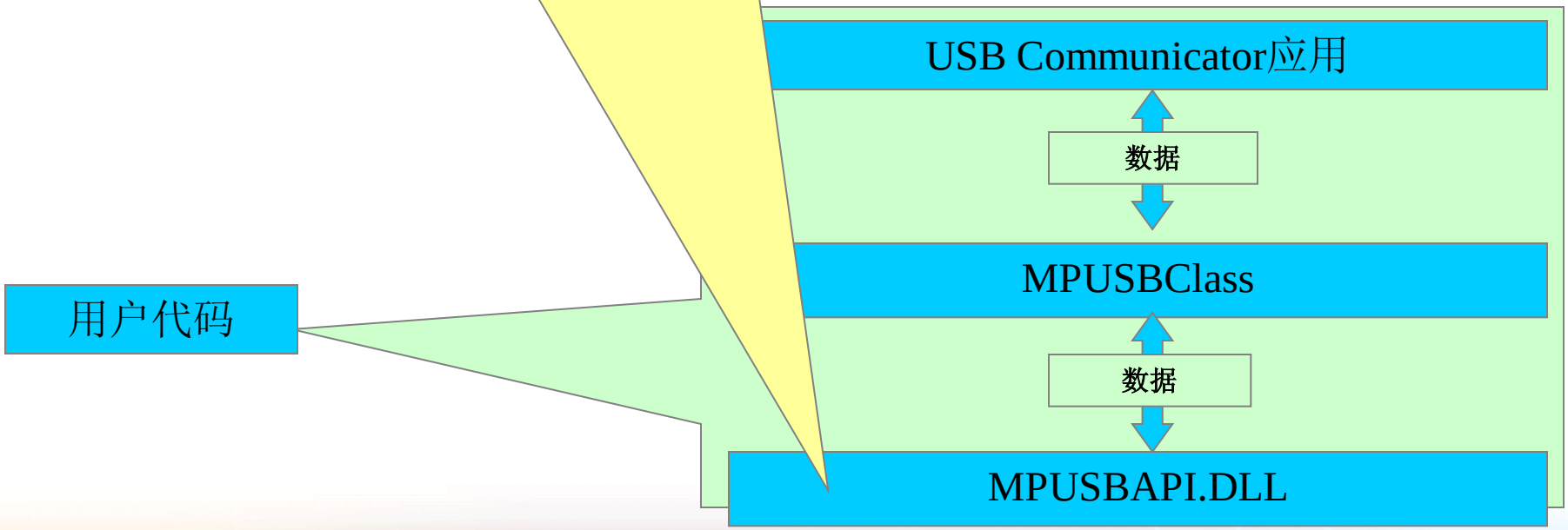
Windows侧

桥接用户应用和MPUSBAPI.DLL
易用的属性/方法/函数
对USB“友好的”接口
函数，诸如USB消息泵、SendMessage、DeviceIndex和TargetVID_PID，等等。



Windows侧

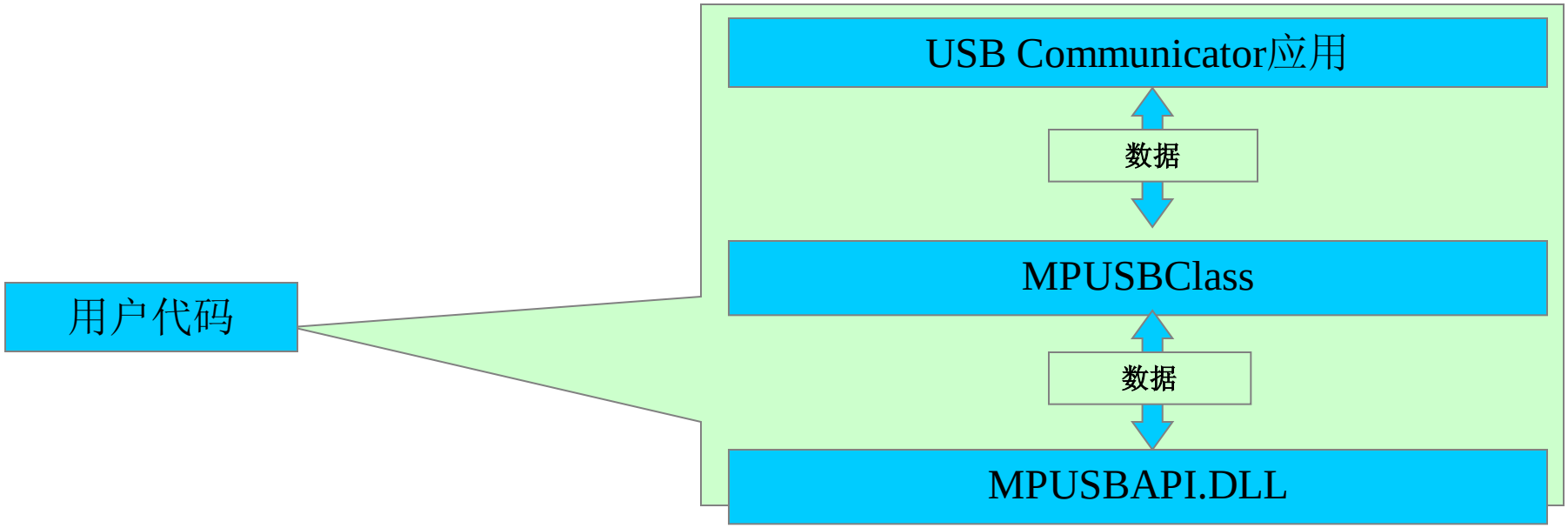
桥接用户应用和Microchip USB驱动程序
低层函数：
- USB打开/关闭管道
- 低层读/写
- GetUSBDescriptors



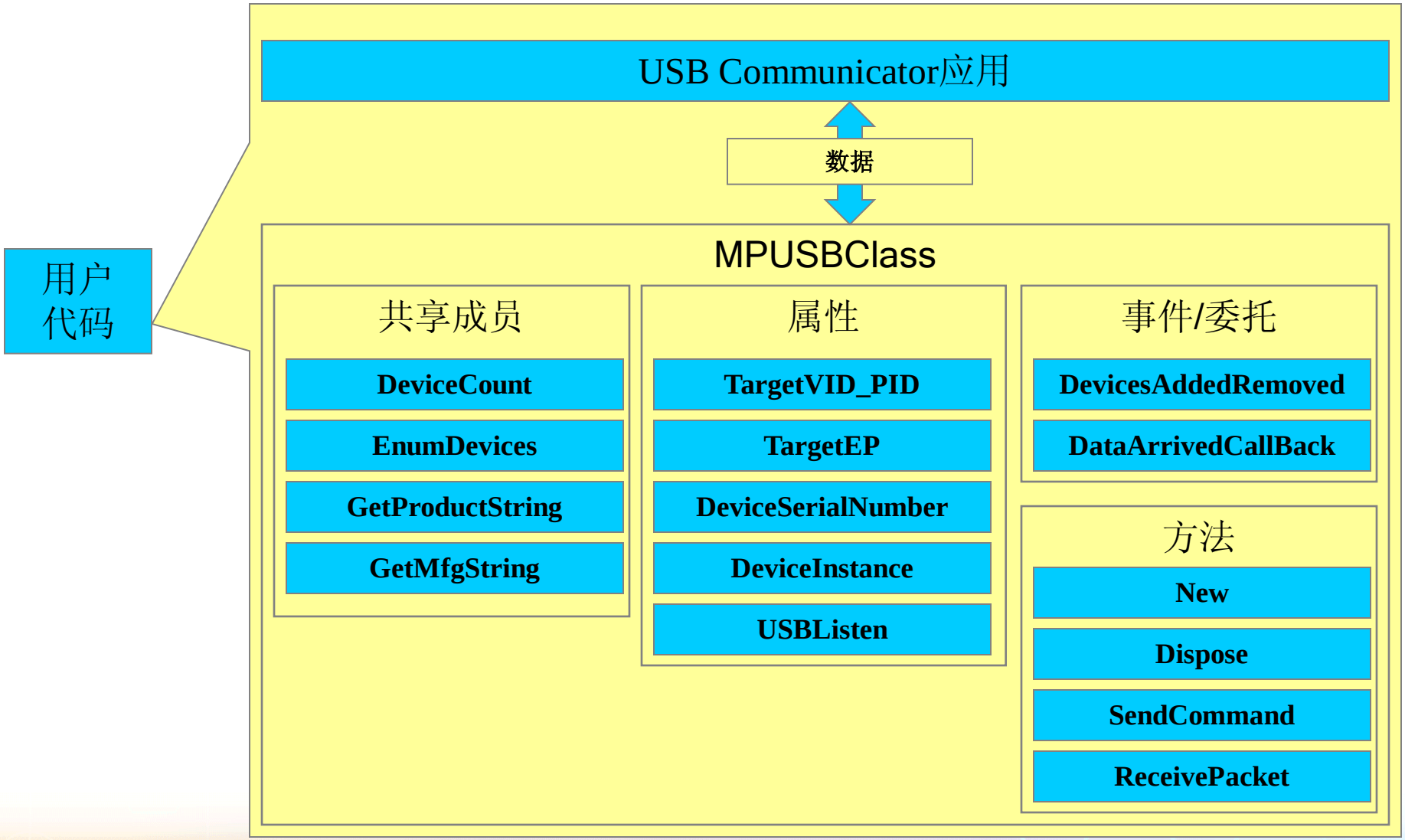
问题

- 用户应用能够直接与驱动程序进行通信吗？
  - 能
- 应用/类/DLL接口都有哪些优点？
  - 可移植性
  - 灵活性
  - 代码简洁：
    1. **Dim USBDevice As MPUSBClass**
    2. **USBDevice = New MPUSBClass(TxtData, dispDelegate)**
    3. **USBDevice.TargetVID_PID="vid_04d8&pid_000c“**
    4. **USBDevice.TargetEP=" \MCHP_EP1“**
    5. **USBDevice. DeviceSerialNumber=1033**
    6. **USBDevice.USBListen=True**
    7. **USBDevice.SendCommand(READ_DATA)**
    - 在USB总线上将自动跟踪序列号为1033的设备（即使在系统中插入了具有相同VID/PID的其他设备）。
    - 通信仅在主机和此设备之间进行。
    - 当USB数据达到时，类将自动调用子程序ProcessData。

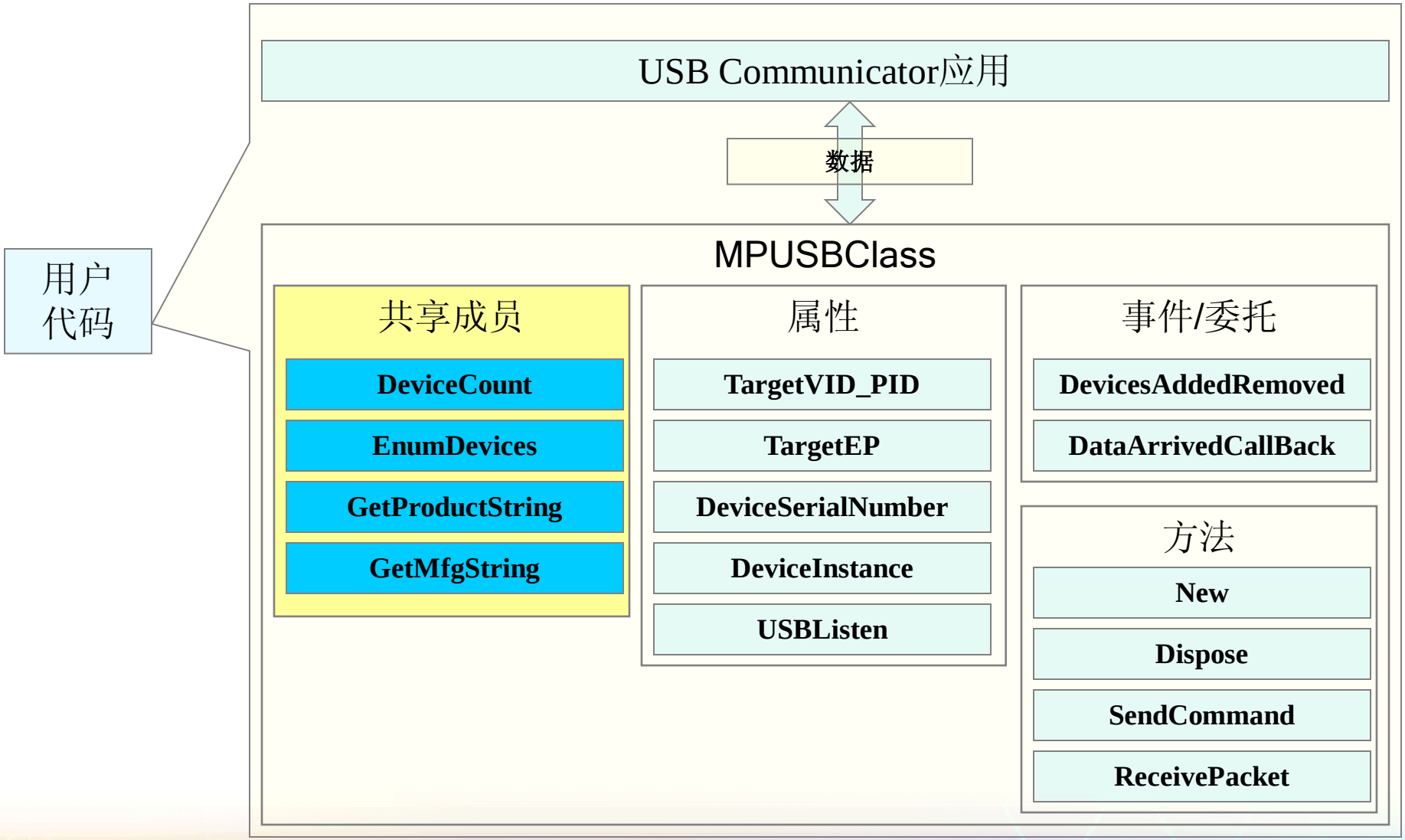
Windows侧



MPUSBClass顶层



MPUSBClass顶层



共享函数

- 为什么使用共享函数？
  - 用来与USB总线上的任何/所有设备进行通信
  - 进行枚举并获取USB描述符，等等
  - 在所有的类实例之间进行共享
    - 不属于任何一个实例

共享函数

共享成员

- **DeviceCount**
- **EnumDevices**
- **GetProductString**
- **GetMfgString**
- **GetSerialNumber**

DeviceCount Function (Integer)
数出USB总线上设备的个数。

输入： Vid_Pid – 字符串
返回： 返回USB总线上与VID/PID匹配的设备的个数

共享函数

共享成员

- **DeviceCount**
- **EnumDevices**
- **GetProductString**
- **GetMfgString**
- **GetSerialNumber**

EnumDevices Function (Boolean)
枚举USB总线上的设备。

输入：　Vid_Pid – 字符串
　　EndPoint – 字符串
输出：　字符串数组，每个字符串包括：产品字符串 + 软件版本号 + 模块实例 + SN
返回：　成功时返回True

共享函数

共享成员

- **DeviceCount**
- **EnumDevices**
- **GetProductString**
- **GetMfgString**
- **GetSerialNumber**

GetProductString Function (String)
从设备检索产品字符串。

输入：　Vid_Pid – 字符串
　　DeviceInstance – 整数
输出：　返回设备产品字符串

注：这是内部使用的函数，但可从主应用使用它。

共享函数

共享成员

- **DeviceCount**
- **EnumDevices**
- **GetProductString**
- **GetMfgString**
- **GetSerialNumber**

GetMfgString Function (String)
从设备检索制造商字符串。

输入：　Vid_Pid – 字符串
　　DeviceInstance – 整数
返回：　返回制造商字符串

注：这是内部使用的函数，但可从主应用使用它。

共享函数

共享成员

- **DeviceCount**
- **EnumDevices**
- **GetProductString**
- **GetMfgString**
- **GetSerialNumber**

GetSerialNumber Function (Integer)
从设备检索序列号。

输入：　Vid_Pid – 字符串
　　DeviceInstance – 整数
返回：　返回设备序列号

注：这是内部使用的函数，但可从主应用使用它。

共享函数

共享成员

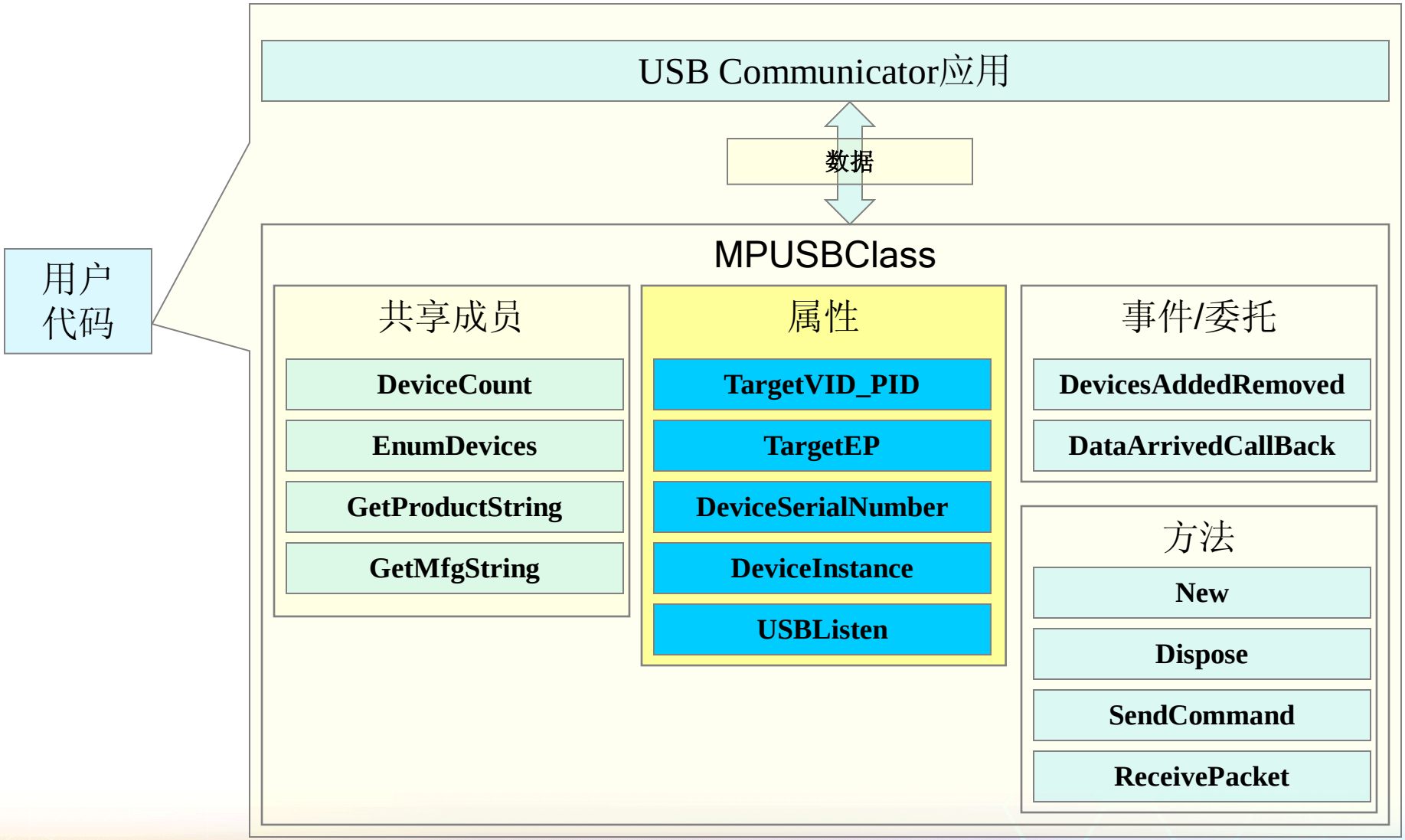
- DeviceCount
- EnumDevices
- GetProductString
- GetMfgString
- GetSerialNumber

- QueryDevice
- SendReceivePacket
- GetUSBDescriptor
- IndexFromSN

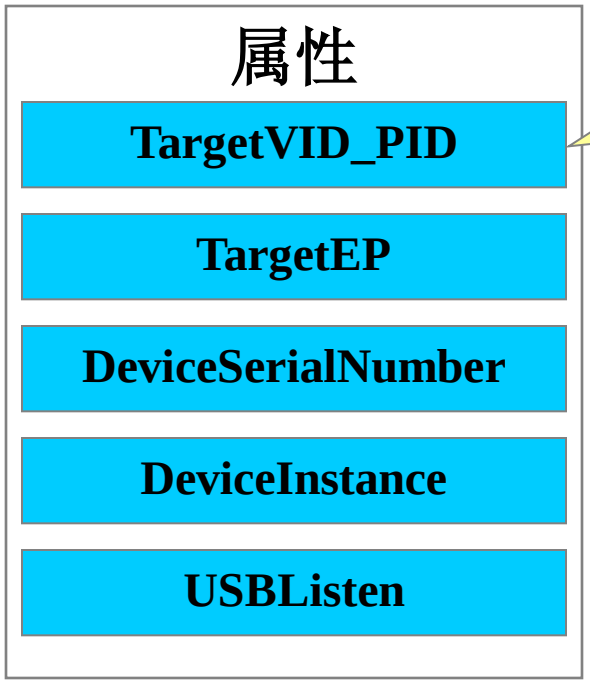
其他函数：

QueryDevice：打开In/Out管道，发送一个包并且接收一个包。
SendReceivePacket：通过打开的In/Out管道，发送/接收包。
GetUSBDescriptor：获取USB描述符。
IndexFromSN：找出与SN匹配的设备，并返回其索引。

MPUSBClass顶层



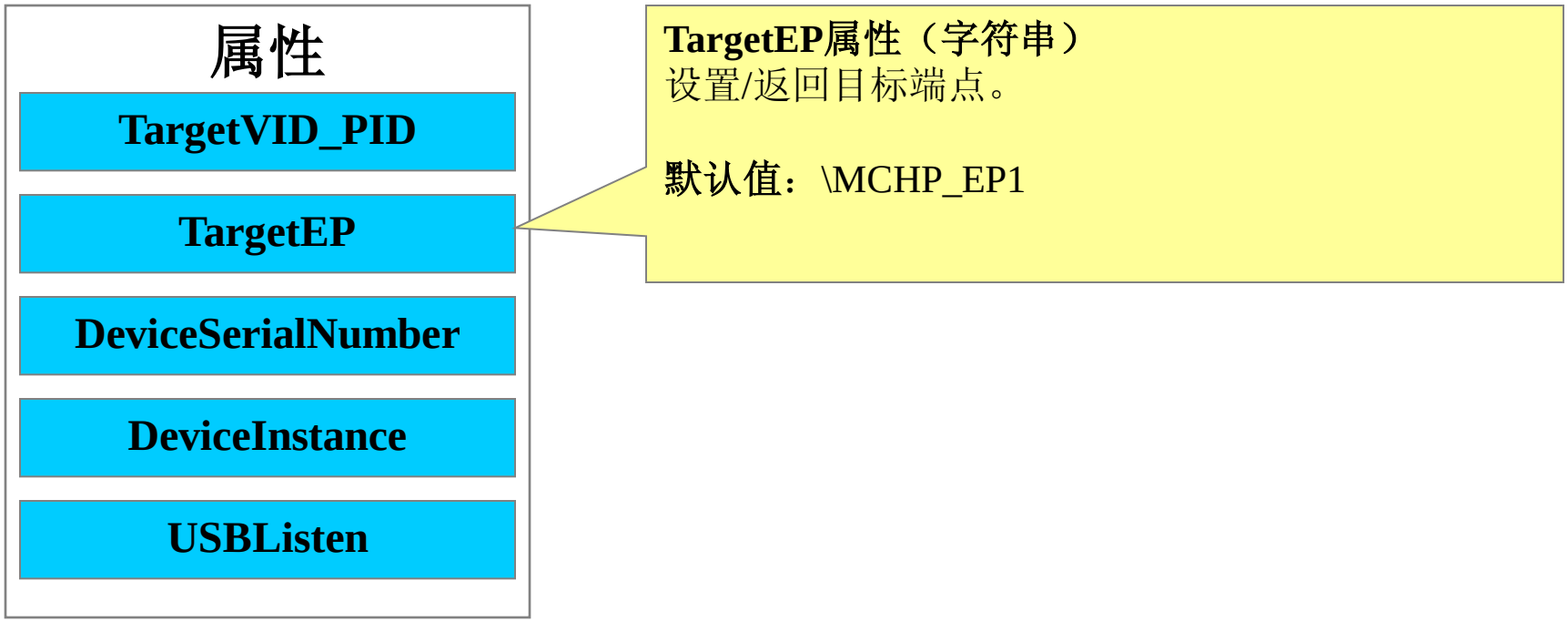
属性



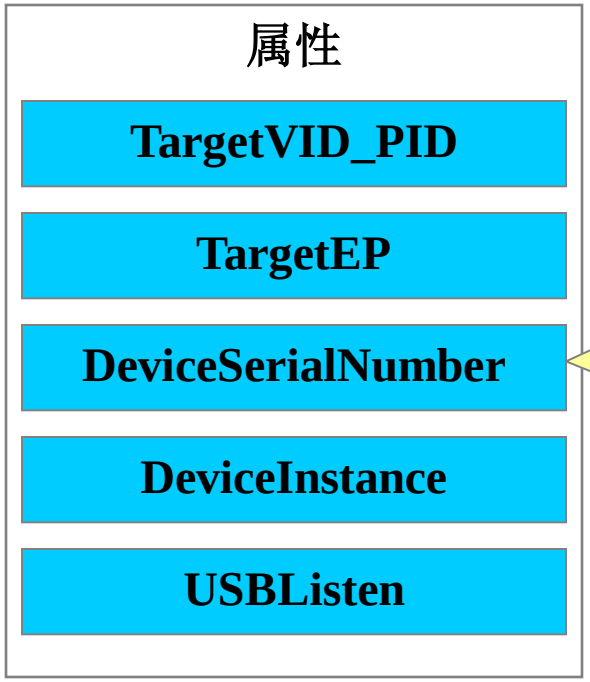
TargetVID_PID属性（字符串）
设置/返回目标设备的VID_PID。

默认值：vid_04d8&pid_000c

属性



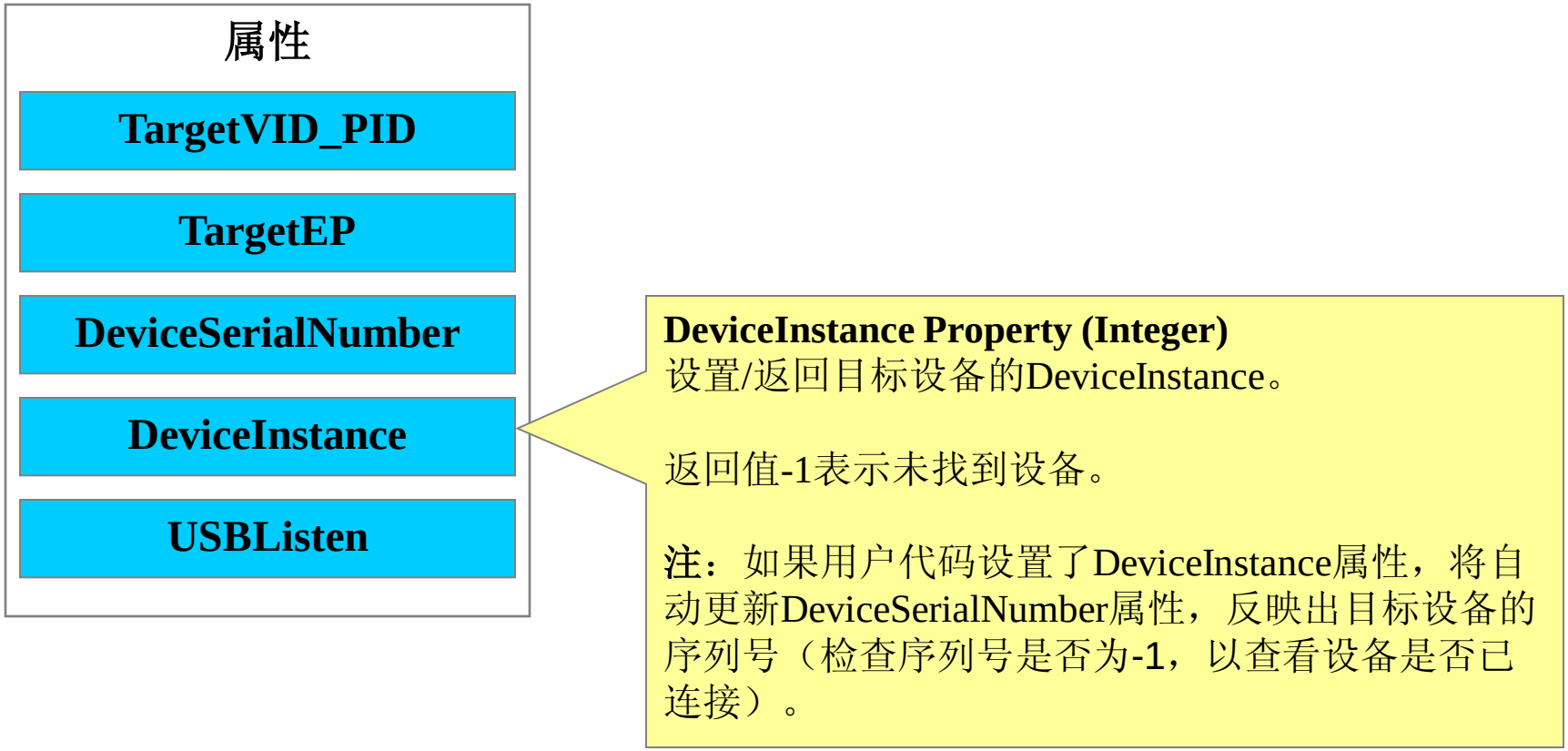
属性



DeviceSerialNumber Property (Integer)
设置/返回目标设备的序列号（这是MPUSBClass将跟踪或正在跟踪的设备）。

注：如果用户代码设置了DeviceSerialNumber属性，将自动更新DeviceInstance属性，反映出序列号匹配的目标设备的索引。如果未找到设备，DeviceInstance将置为-1。

属性



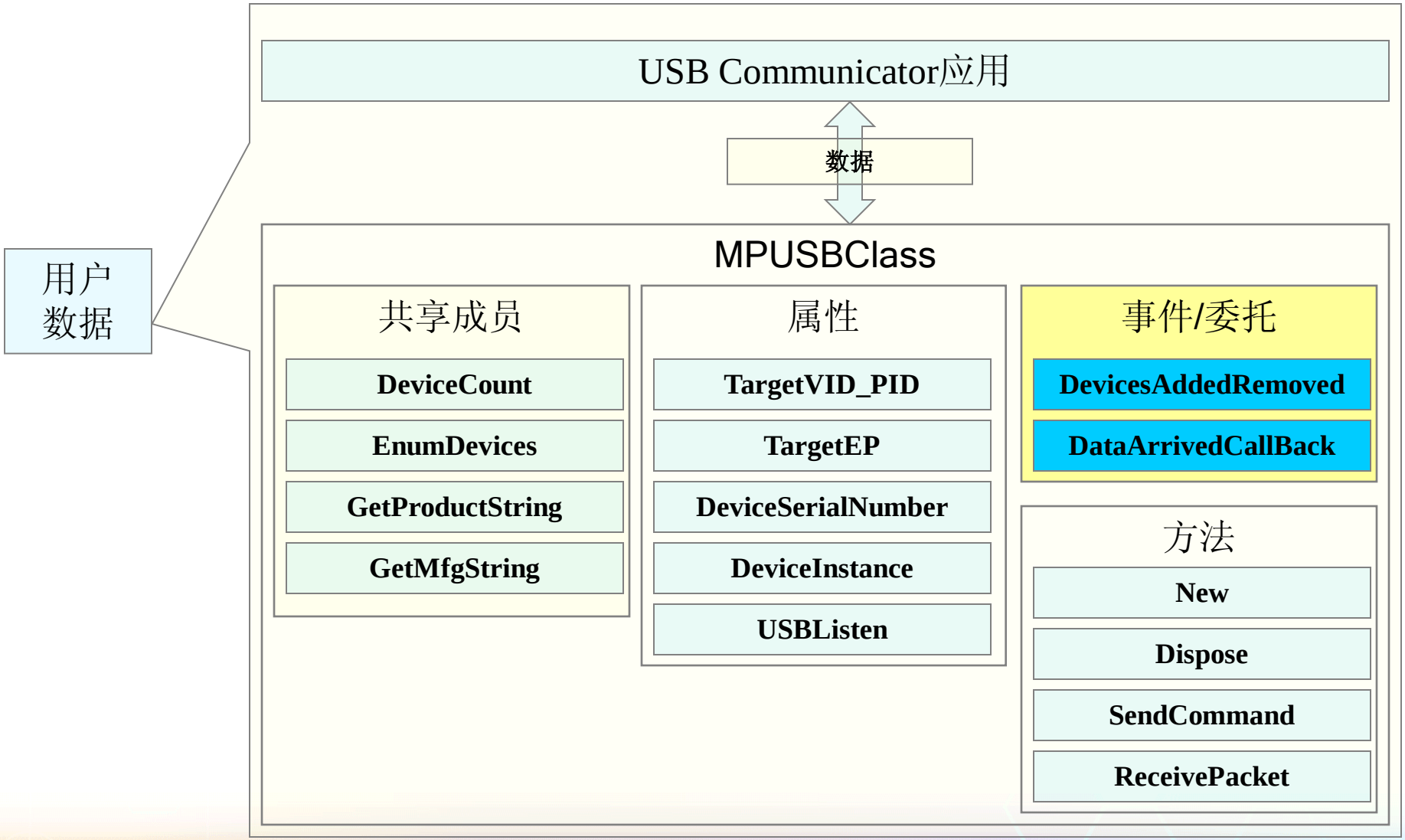
属性

USBListen Property (Boolean)
设置/返回USB消息泵的状态。

设置USBListen = True，启动USB消息泵线程，设置USBListen = False则停止线程。

注：在允许程序继续执行之前，USBListen将等待，一直到USB消息泵线程退出。

MPUSBClass顶层



事件

事件/委托

DevicesAddedRemoved

DataArrivedCallBack

DeviceAddedRemoved事件

输入：　　无
输出：　　无

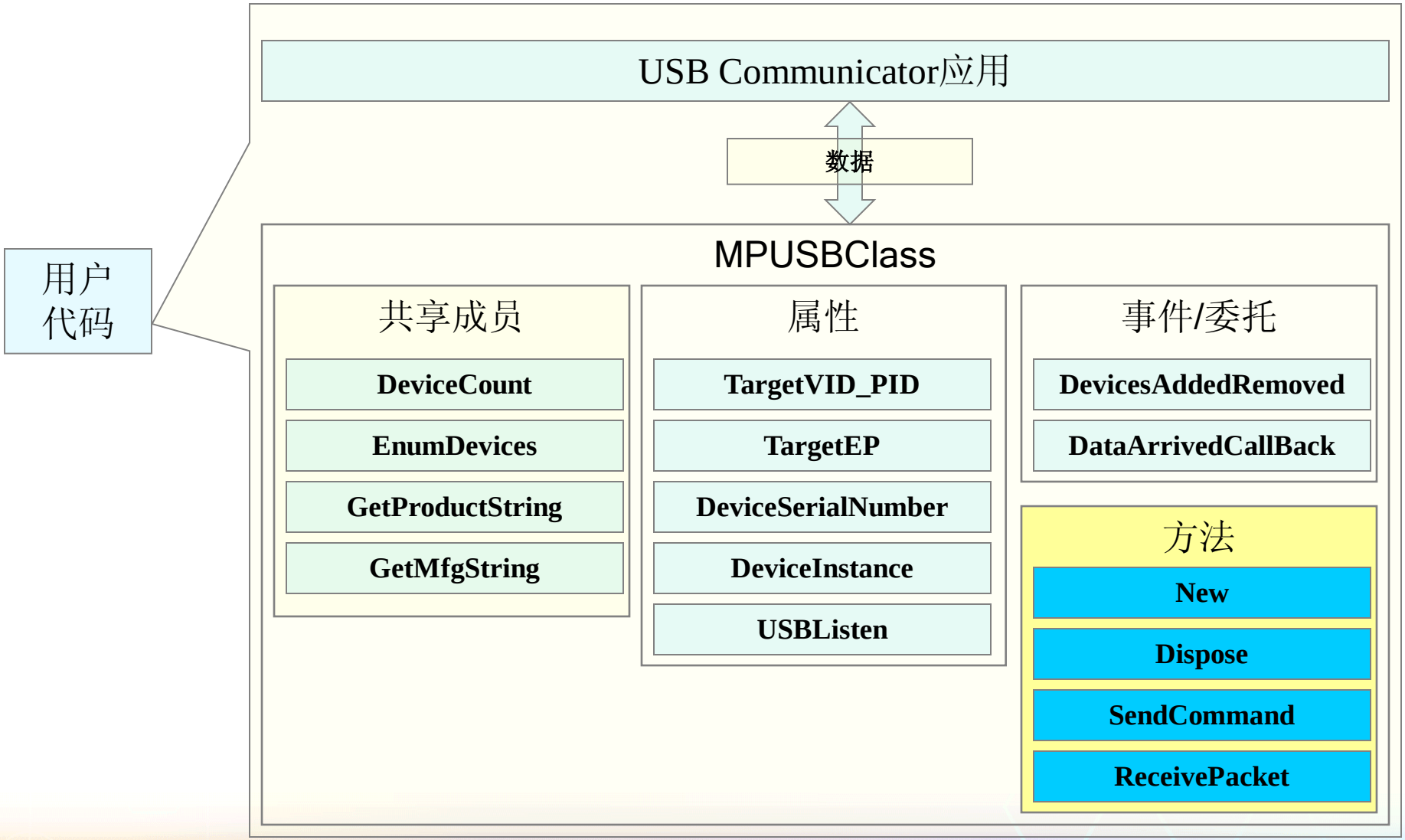
当设备添加至USB总线或从USB总线移除时产生此事件（窗体发送WM_DEVICECHANGE消息）。

DataArrivedCallBack回调子程序

输入：　　无
输出：　　ArrayData() – 含有到达数据的字节数组

在USB消息泵收到来自设备的数据时，进行调用。
注：调用DataArrivedCallBack的线程与类固有变量声明时传递给New()方法的控件为同一线程。

MPUSBClass顶层



方法

方法

- **New**
- **Dispose**
- **SendCommand**
- **ReceivePacket**

New方法

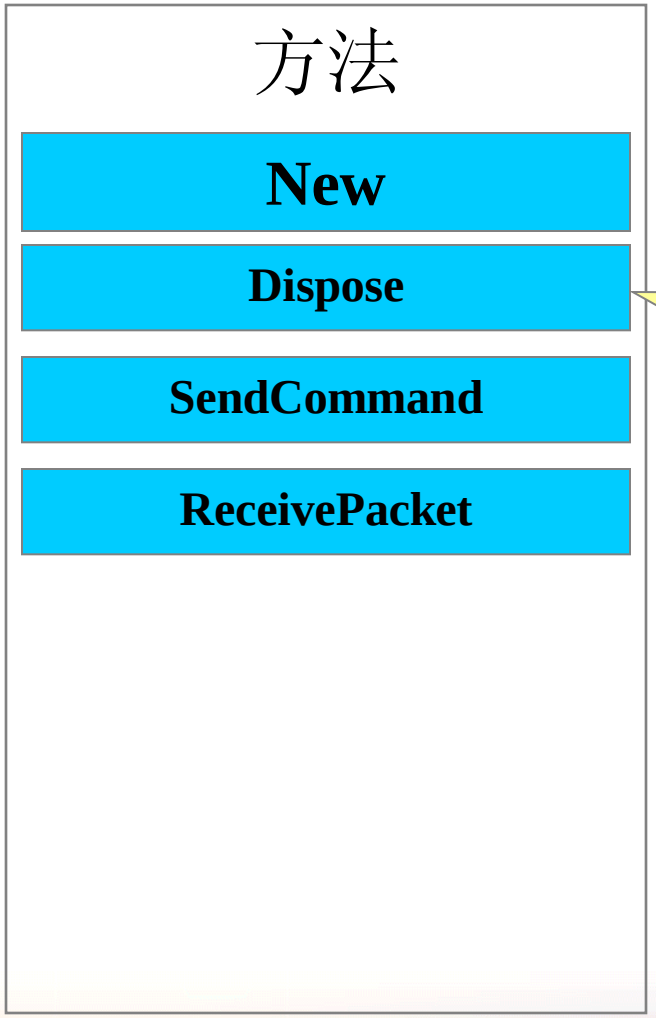
输入：

UpdateControl – 用来调用DataArrivedCallBack函数的控件。

DisplayDelegate – DataArrivedCallBack委托。当新数据到达USB总线时调用。

用来创建一个新的MPUSBClass实例，并在主应用中赋值给新声明的变量。

方法

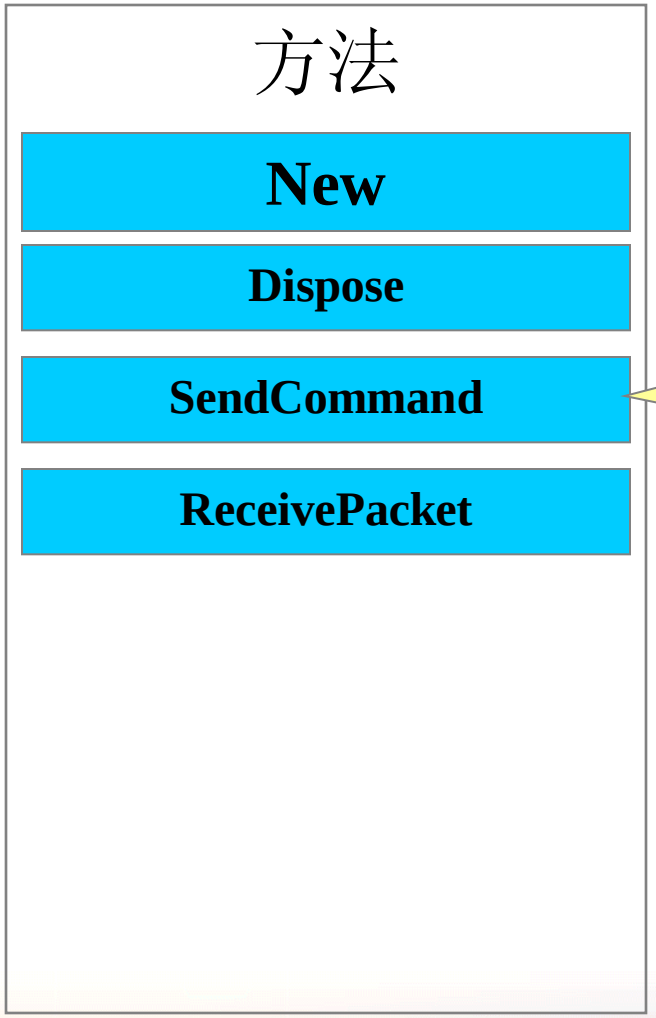


Dispose方法

输入：　　无
输出：　　无

释放与特定MPUSBClass实例关联的全部资源。

方法

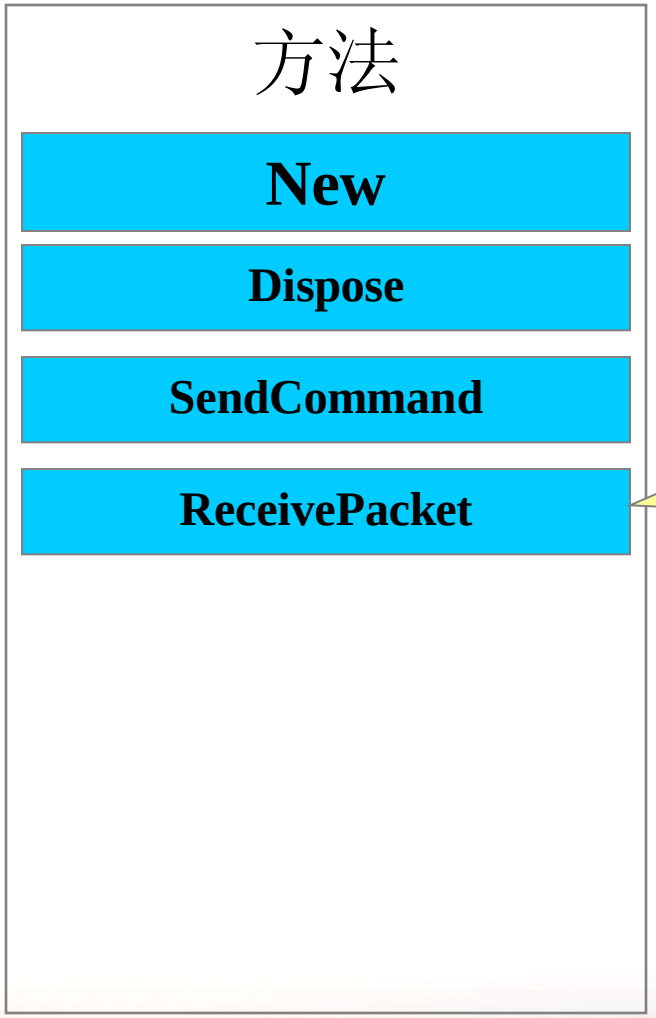


SendCommand

输入：　　cmdToSend() – 数据字节数组
数出：　　无

成功时返回1，如果失败则返回0。

方法

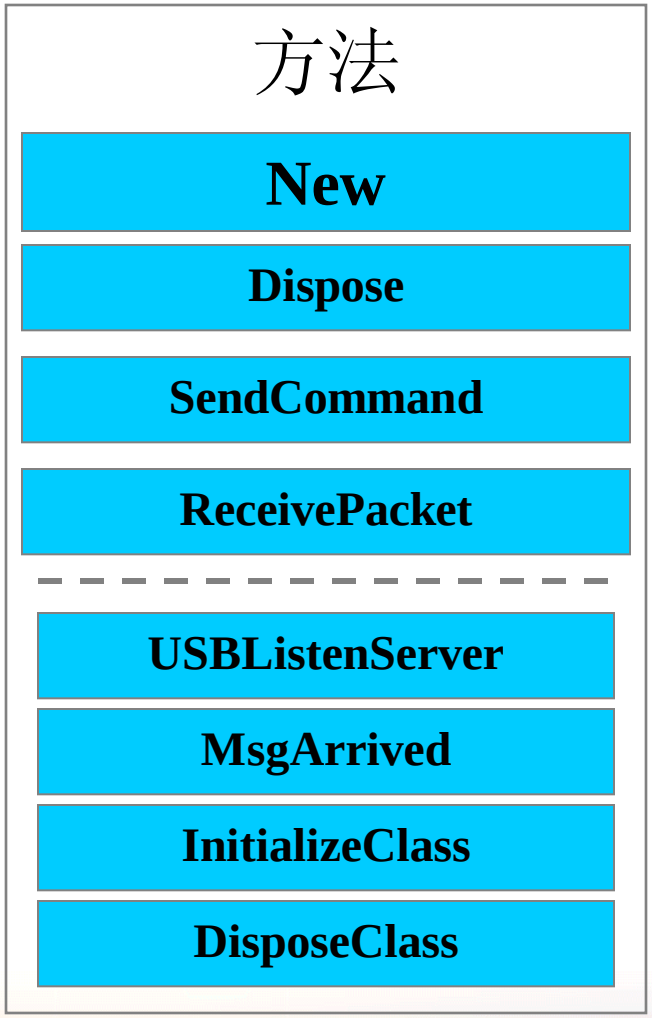


ReceivePacket

输入： 无
输出： cmdToReceive () – 数据字节数组

成功时返回1，如果失败则返回0。

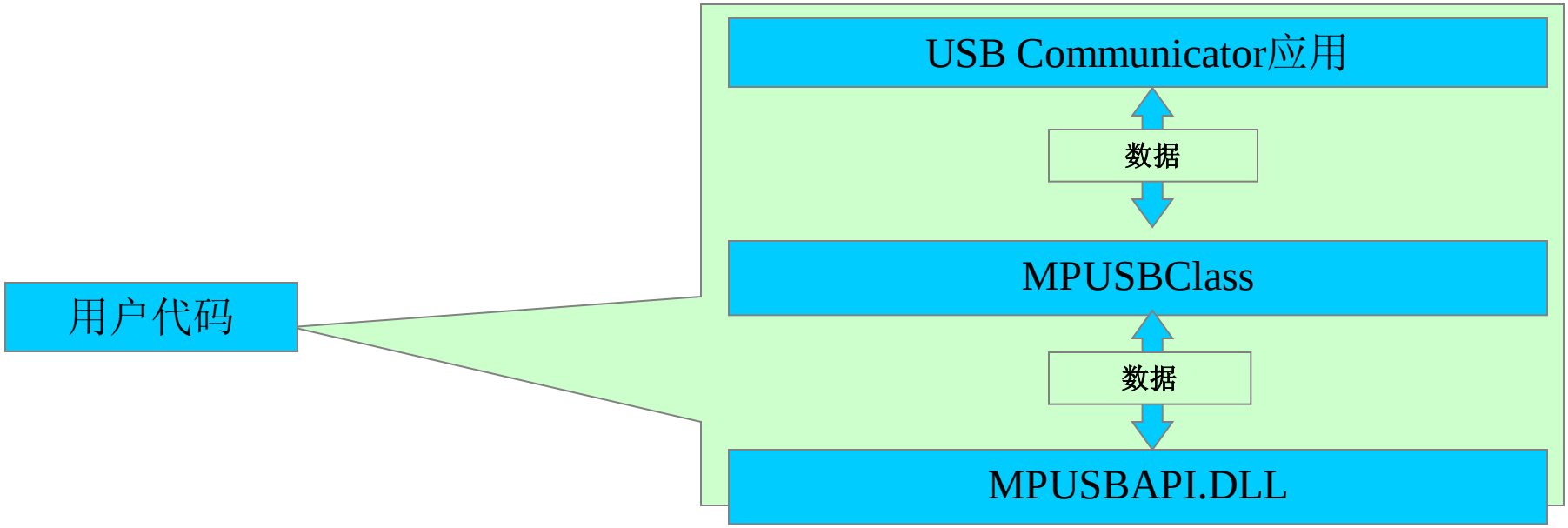
方法



Private方法
（仅可从MPUSBClass进行访问）

USBListenServer – USB消息泵
MsgArrived – 用来跟踪WM_DEVICECHANGE
InitializeClass – 为New方法调用
DisposeClass – 为Dispose方法调用

MPUSBAPI.DLL



MPUSBAPI.DLL

MPUSBAPI.DLL

MPUSBGetDLLVersion

MPUSBGetDLLVersion

输入：　无

数出：　32位版本号MMMMmmmm

返回mpusbapi.dll的版本号。

MPUSBAPI.DLL

MPUSBAPI.DLL

MPUSBGetDLLVersion

MPUSBGetDeviceCount

MPUSBGetDeviceCount

输入： VIDPID – 字符串

输出： 返回与VID和PID匹配的设备的个数。

MPUSBAPI.DLL

MPUSBAPI.DLL

MPUSBGetDLLVersion

MPUSBGetDeviceCount

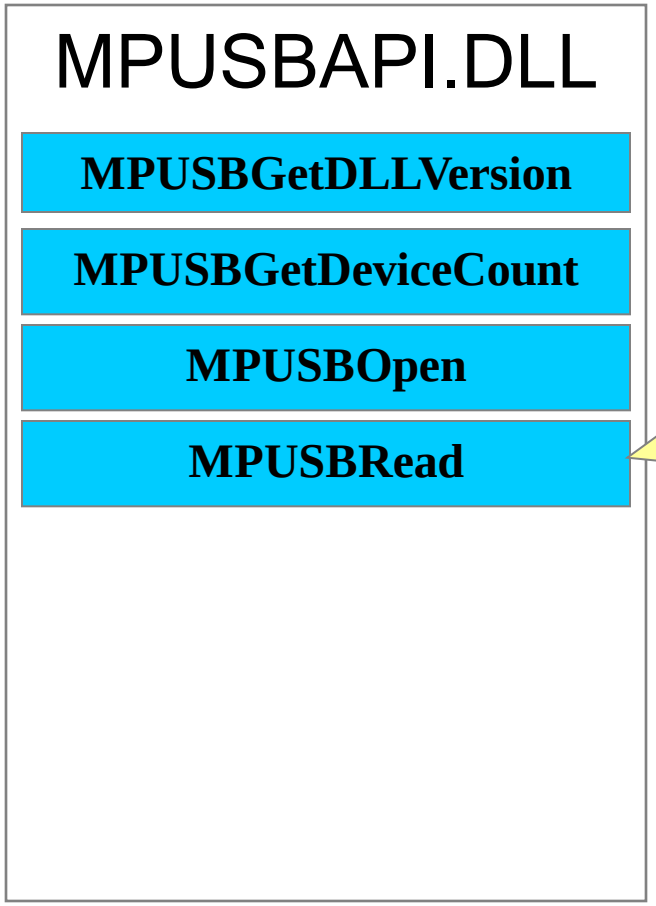
MPUSBOpen

MPUSBOpen

输入：　Instance – 设备实例
pVID_PID – 字符串
pEP – 字符串
dwDir – 方向（读/写）
dwReserved – 将来使用

返回与VID和PID匹配的端点管道的句柄。

MPUSBAPI.DLL



MPUSBRead

输入： handle – 打开管道的句柄
dwLen – 要读的字节数
dwMilliseconds – 超时时限

输出： pData – 数据缓冲区
pLength – 读取的数据长度

失败时返回0，如果成功则返回1。

MPUSBAPI.DLL

MPUSBAPI.DLL

- **MPUSBGetDLLVersion**
- **MPUSBGetDeviceCount**
- **MPUSBOpen**
- **MPUSBRead**
- **MPUSBWrite**

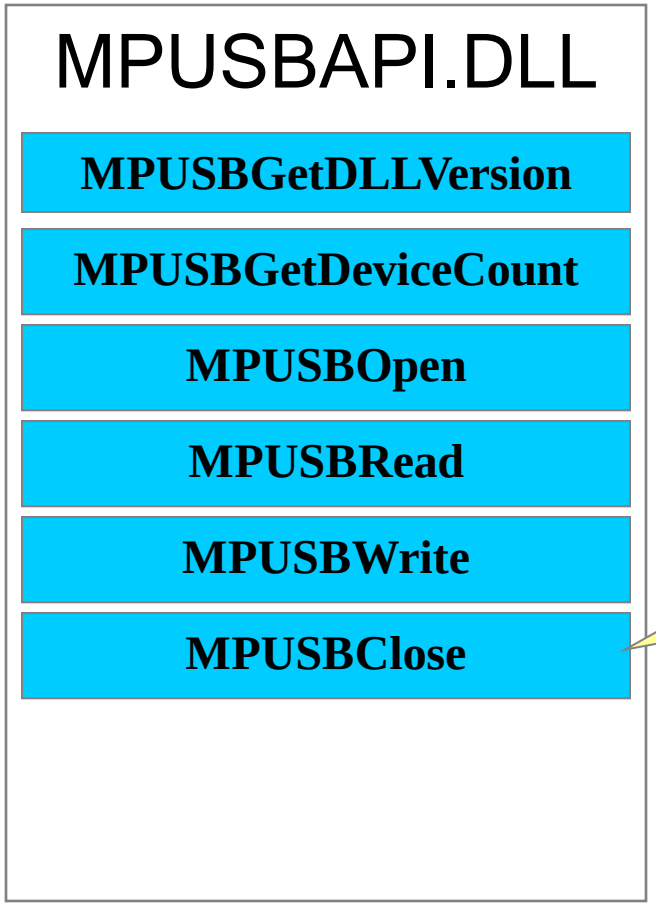
MPUSBWrite

输入： handle – 打开管道的句柄
pData – 数据缓冲区
dwLen – 要写入的字节数
dwMilliseconds – 超时时限

输出： pLength – 写入的数据长度

失败时返回0，如果成功则返回1。

MPUSBAPI.DLL



MPUSBClose

输入： handle – 打开管道的句柄

关闭指定的句柄。

MPUSBAPI.DLL

MPUSBAPI.DLL

- **MPUSBGetDLLVersion**
- **MPUSBGetDeviceCount**
- **MPUSBOpen**
- **MPUSBRead**
- **MPUSBWrite**
- **MPUSBClose**
- **MPUSBGetDeviceDescriptor**

MPUSBGetDeviceDescriptor

输入： handle – 打开管道的句柄

pDscParam – 指针，指向 **GET_DESCRIPTOR_PARAMETER**

dscLen – pDscParam中的数据长度

dwLen – 分配给输出的存储单元长度（预计返回的数据长度）

输出： pDevDsc – 指向设备描述符结构体的指针

pLength – 写入设备描述符结构体的字节数

失败时返回0，如果成功则返回1。

MPUSBAPI.DLL

MPUSBAPI.DLL

- **MPUSBGetDLLVersion**
- **MPUSBGetDeviceCount**
- **MPUSBOpen**
- **MPUSBRead**
- **MPUSBWrite**
- **MPUSBClose**
- **MPUSBGetDeviceDescriptor**
- **MPUSBReadInt**

MPUSBReadInt

输入： handle – 打开管道的句柄
dwLen – 要读取的字节数
dwMilliseconds – 超时时限

输出： pData – 数据缓冲区
pLength –读取的数据长度

失败时返回0，如果成功则返回1。

注：pEP应当是“\\MCHP_EPz_ASYNC”格式。此选项仅可用于IN中断端点。使用“_ASYNC”关键字打开的数据管道，将在端点描述符规定的时间间隔内缓冲数据，最大至100个数据集。驱动程序缓冲区已满后收到的任何数据都将被忽略。用户应用应频繁调用MPUSBReadInt()，确保不会达到100的最大限制。

开始使用USB和VB.Net

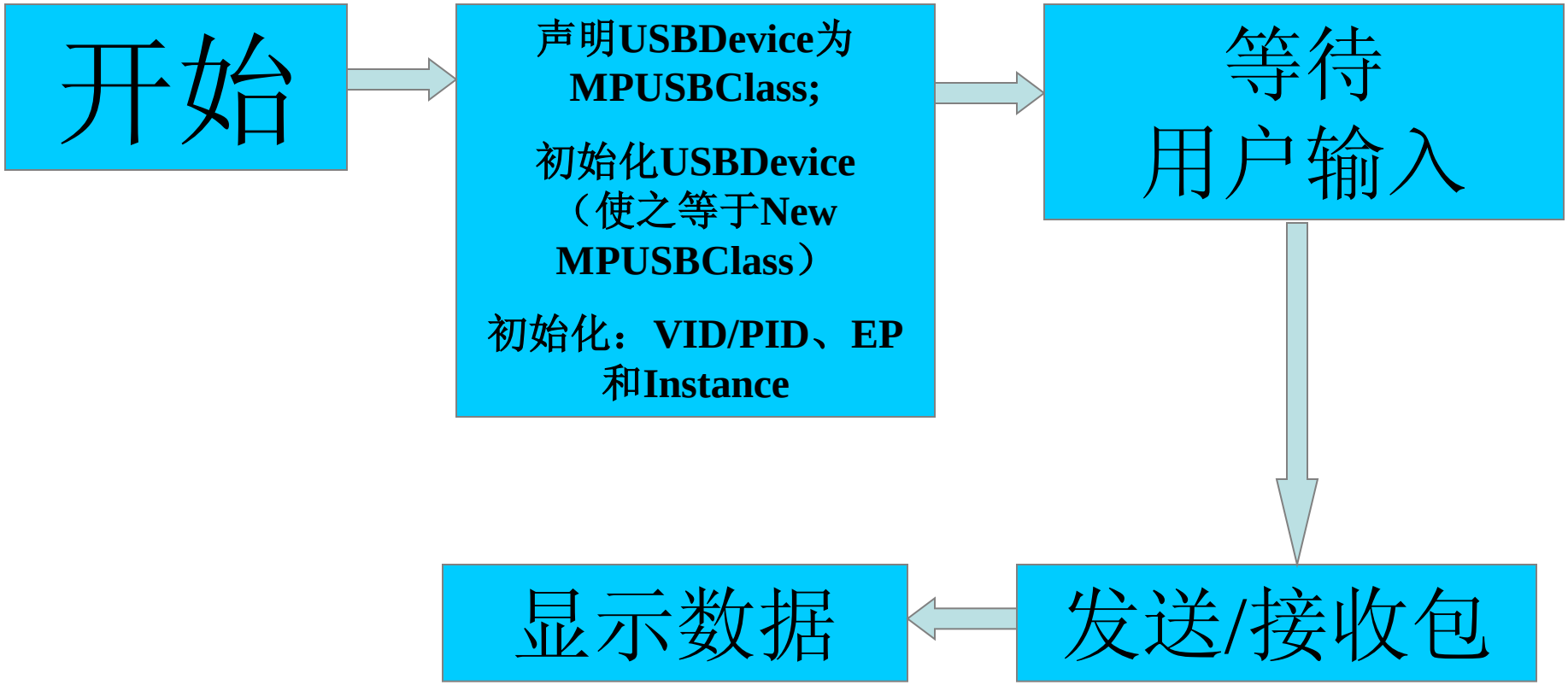
课程安排

- 双向USB通信工作原理
- VB.Net USB类以及示例应用
  - 函数释义
  - 单点及多点获取
  - 连续数据获取
  - USB描述符获取
    - 检索设备制造商序列号
    - 检索设备产品字符串
    - 检索设备制造商字符串
  - WM_DEVICECHANGE事件（设备枚举）
  - 通过窗体实例进行跟踪
  - 通过序列号进行跟踪
- Microchip演示/开发解决方案
- 双向USB通信工作原理

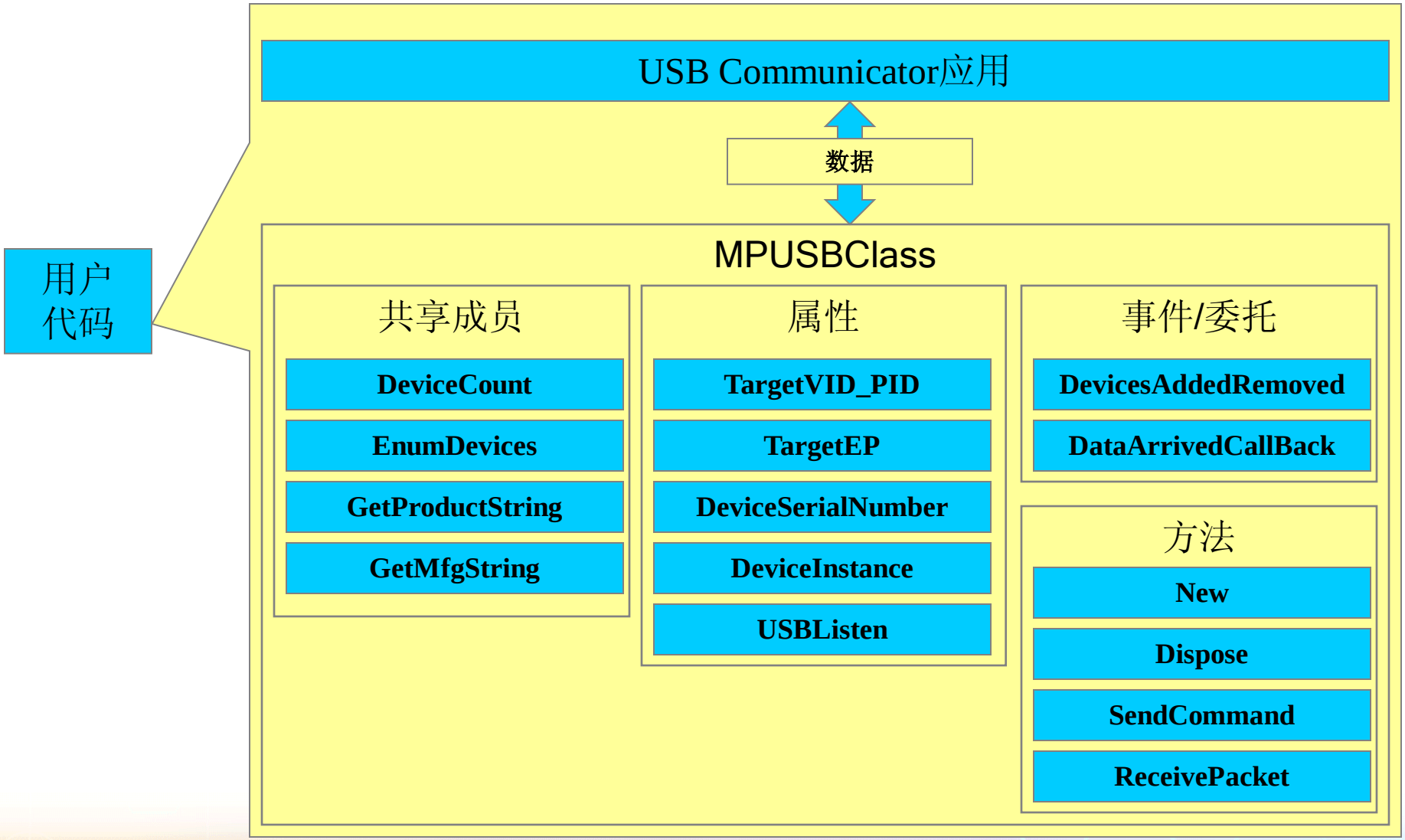
单点获取

- 程序要求
  - 有1个按钮和1个文本框的表单
    - 按钮文本置为“Read Pot”
    - 表单文本“USB Reader 1”
  - 按钮按下时：
    - 请求电位器数据
    - 检查是否有有效的包传输
      - 转换并显示电位器值（单位为欧姆）
      - 如果没有发送/接收包，显示“Error”
  - 使用“vid_04d8&pid_000c”和“\MCHP_EP1”

Windows USB
单点获取



单点获取



单点获取

- 变量声明
  - 数组声明
    - Dim DataArray(0) as Byte
    - DataArray(0) = MPUSBClass.USB_CMD_LIST.RD_TEMP
  - 类声明/用途
    - Dim MyDev as MPUSBClass
    - MyDev = New MPUSBClass(TextBox1, AddressOf DataArrived)
      - 文本框应该已存在于主表单上
      - **Sub DataArrived(data() as byte)**
    - MyDev.CustomProperty = 值
    - MyDev.SendInformation (DataArray)

单点获取

- 电位器值：
  - 10位
    - 低字节在**Data(1)**中
    - 高字节在**Data(2)**中
- 返回的数据
  - Byte (0) = 请求命令的回显
  - Byte (1…63) = 数据

显示数据，单位是欧姆：

```
Label1.Text = Utils.toOhms(DataBytes)
```

演示2

连续单点获取

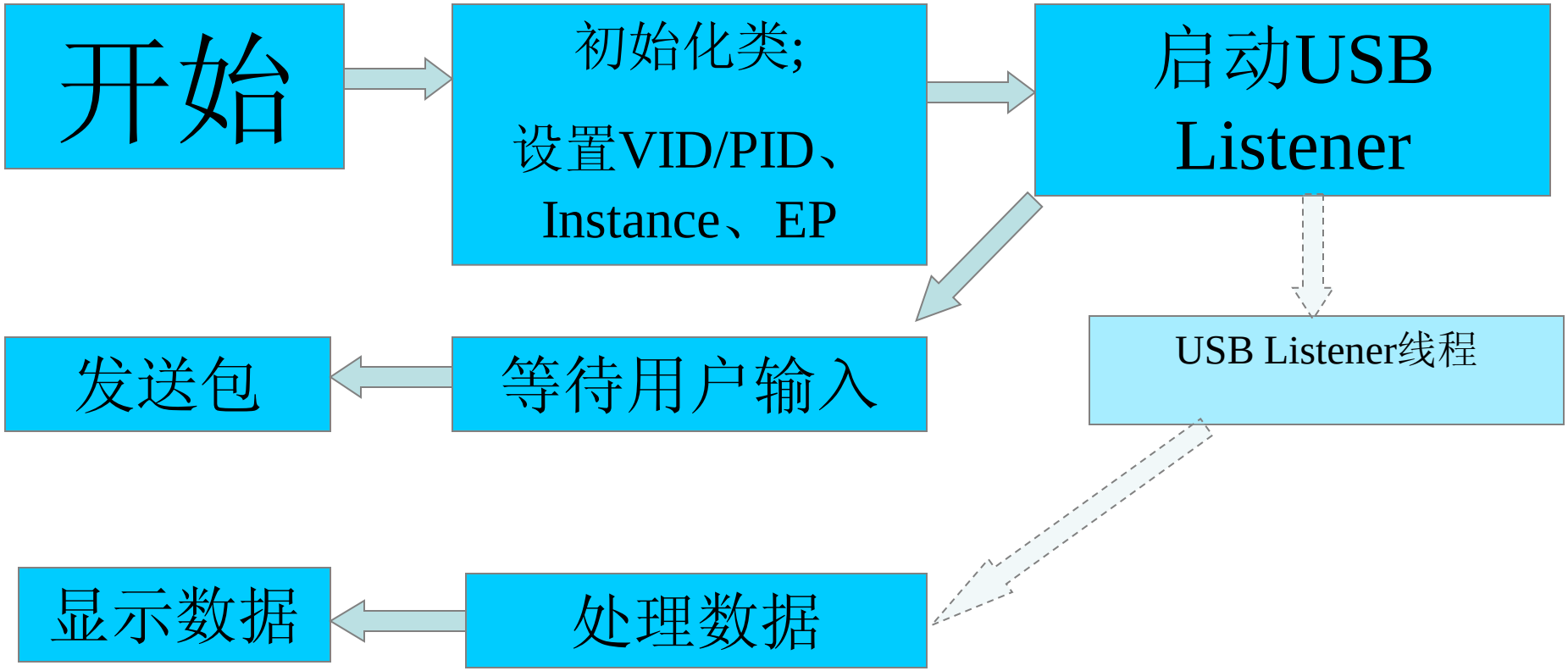
连续单点获取

- 连续发送/接收包的优点/缺点？
  - 优点
    - 容易使用
    - 容易调试
    - 容易理解
  - 缺点
    - 数据吞吐量低
      - **每个SendCommand都要打开和关闭管道**
      - **每个ReceivePacket都要打开和关闭管道**
    - 在等待设备响应时将挂起应用

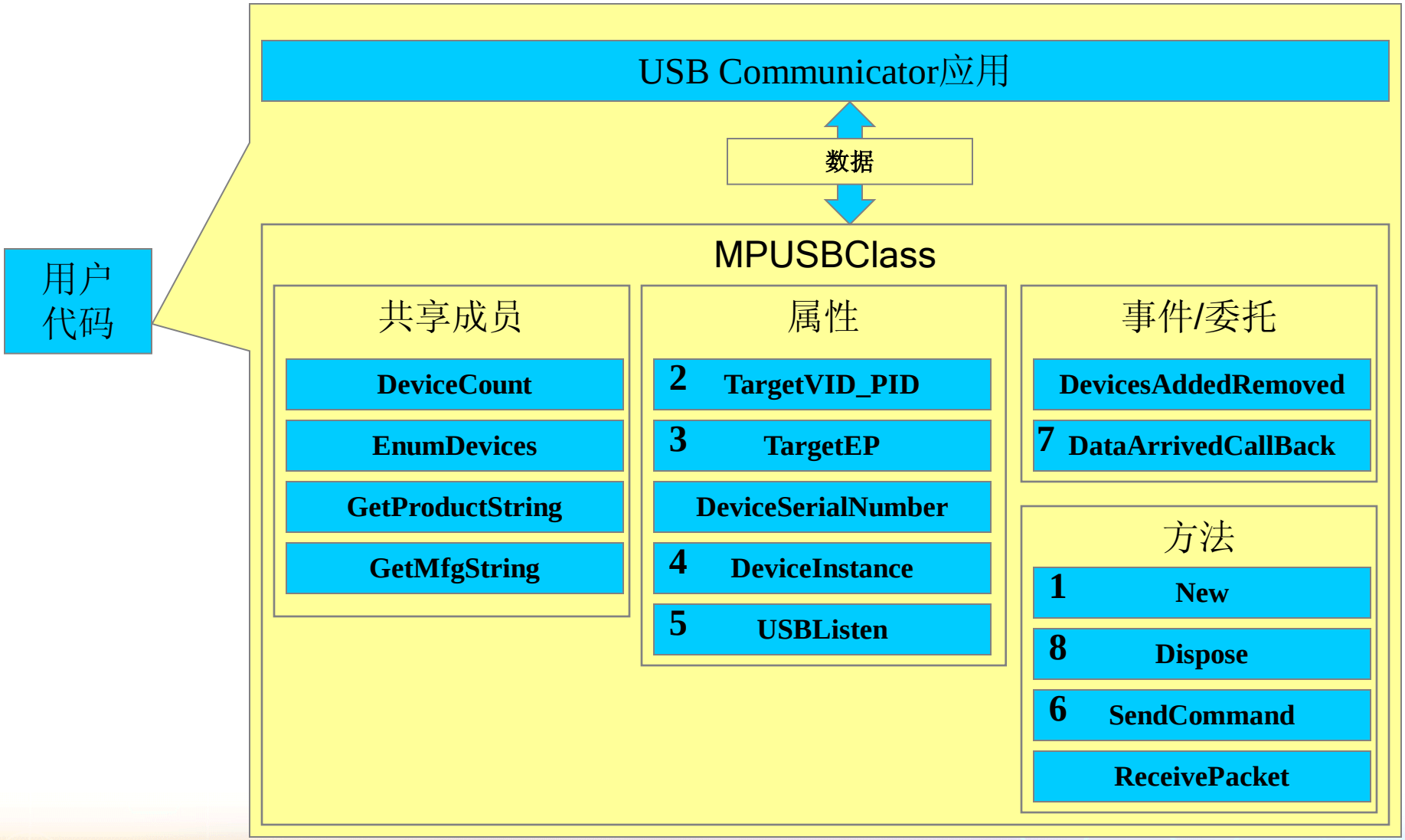
多线程数据获取

- 程序要求
  - 有1个按钮和1个文本框的表单
    - 按钮文本置为“Read Temp”
    - 表单文本“USB Reader 1”
  - 按钮按下时：
    - 请求温度数据
    - 检查是否有有效的包传输
      - 转换并显示温度（单位为°C）
      - 如果没有发送/接收包，显示“Error”
  - 使用“vid_04d8&pid_000c”和“\MCHP_EP1”

Windows USB
多线程数据获取



多线程数据获取



多线程数据获取

- 有用的技巧
  - 类声明/初始化
    - Dim MyDev as ExampleClass
    - 在代码中：
      - **MyDev=New ExampleClass(Textbox1, AdressOf ProcessData)**
        - 注：**DataSub应已被声明为sub ProcessData (DataArray() as byte)**
  - 数组声明
    - Dim DataArray(0) as MPUSBClass.USB_CMD_LIST
    - DataArray(0)= MPUSBClass.USB_CMD_LIST.RD_TEMP
  - 初始化
    - Use Form1_Load
  - 终止
    - Use Form1_FormClosing
  - 子程序
    - Sub ProcessData(DataArray() as Byte)
  - 返回的数据
    - Byte (0) = 请求命令的回显
    - Byte (1…63) = 数据

演示3

多线程数据获取

多设备通信

- 如果有下列情况，会怎么样呢？
  - 在USB总线上有多个设备
  - 需要设备选择能力（用户）
  - 需要将数据发送给指定设备
- 解决方案
  - 设备枚举
  - 设备序列号

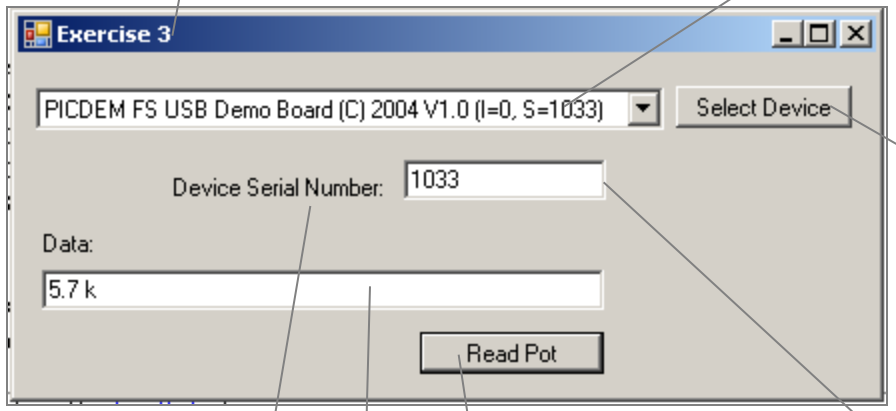
多设备通信

- **MPUSBClass的设备枚举**
  - **MPUSBClass.DeviceCount** —— 在USB总线上匹配VID/PID的设备的个数
  - **MPUSBClass.EnumDevices** —— 枚举设备，输出产品字符串 + 软件版本号（如果有的话） + 模块实例 + SN
  - 示例：
    - **PICDEM™ FS USB演示板 © 2004 V1.0 (I=0,SN=1033)**
  - 注：如果成功，**EnumDevices返回True**
  - 使用**var.length来确定输出数组的大小**

多设备通信

- 程序要求

Form.Text=Lab 3



Combo Box:

显示其Vid_Pid="vid_04d8&pid_000c"（使用Ep= "\MCHP_EP1"）的设备列表

Button:

把USBDevice.DeviceIndex置为ComboBox1.SelectedIndex

在TextBox1中显示设备序列号

（设置TextBox1.Text属性）

Label

TextBox

电阻

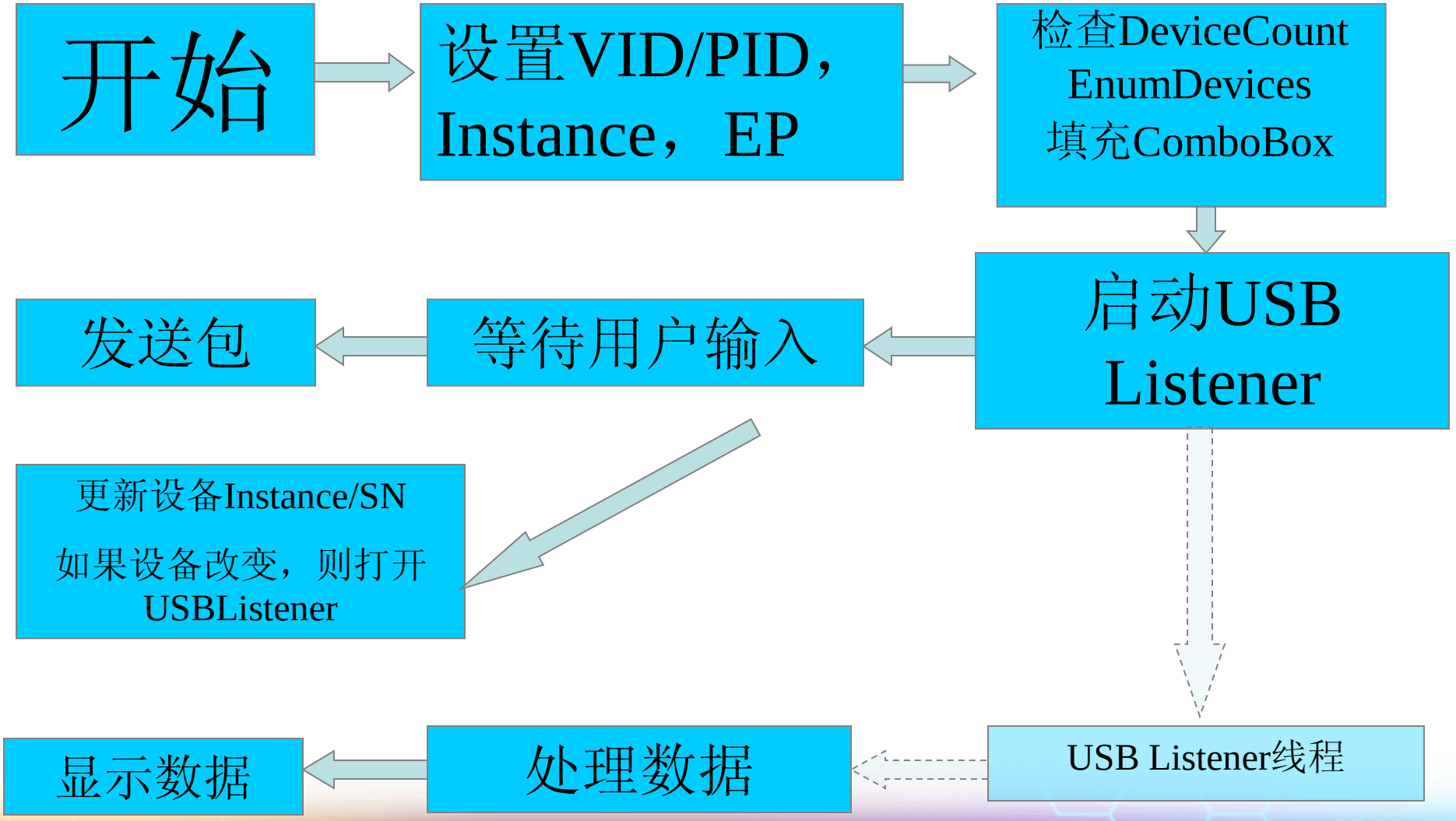
Button:

向设备请求数据

TextBox1:

显示设备序列号

多设备通信



多设备通信

- 有用的技巧
  - 类声明/初始化
    - Dim MyDev as ExampleClass
    - 在代码中：
      - **MyDev=New ExampleClass(Textbox1, AddressOf DataSub)**
  - 数组声明
    - Dim DataArray(0) as MPUSBClass.USB_CMD_LIST
    - DataArray(0) = MPUSBClass.USB_CMD_LIST.RD_TEMP
  - 初始化
    - Use Form1_Load
  - 终止
    - Use Form1_FormClosing
  - 子程序
    - Sub ProcessData(DataArray() as Byte)
  - 返回的数据
    - Byte (0) = 请求命令的回显
    - Byte (1…63) = 数据
  - **ComboBox**
    - 使用ComboBox1.Clear来清除组合框
    - 使用ComboBox1.AddRange(StringArray)，向组合框中添加字符串数组（EnumDevices的输出）

演示4

多设备通信

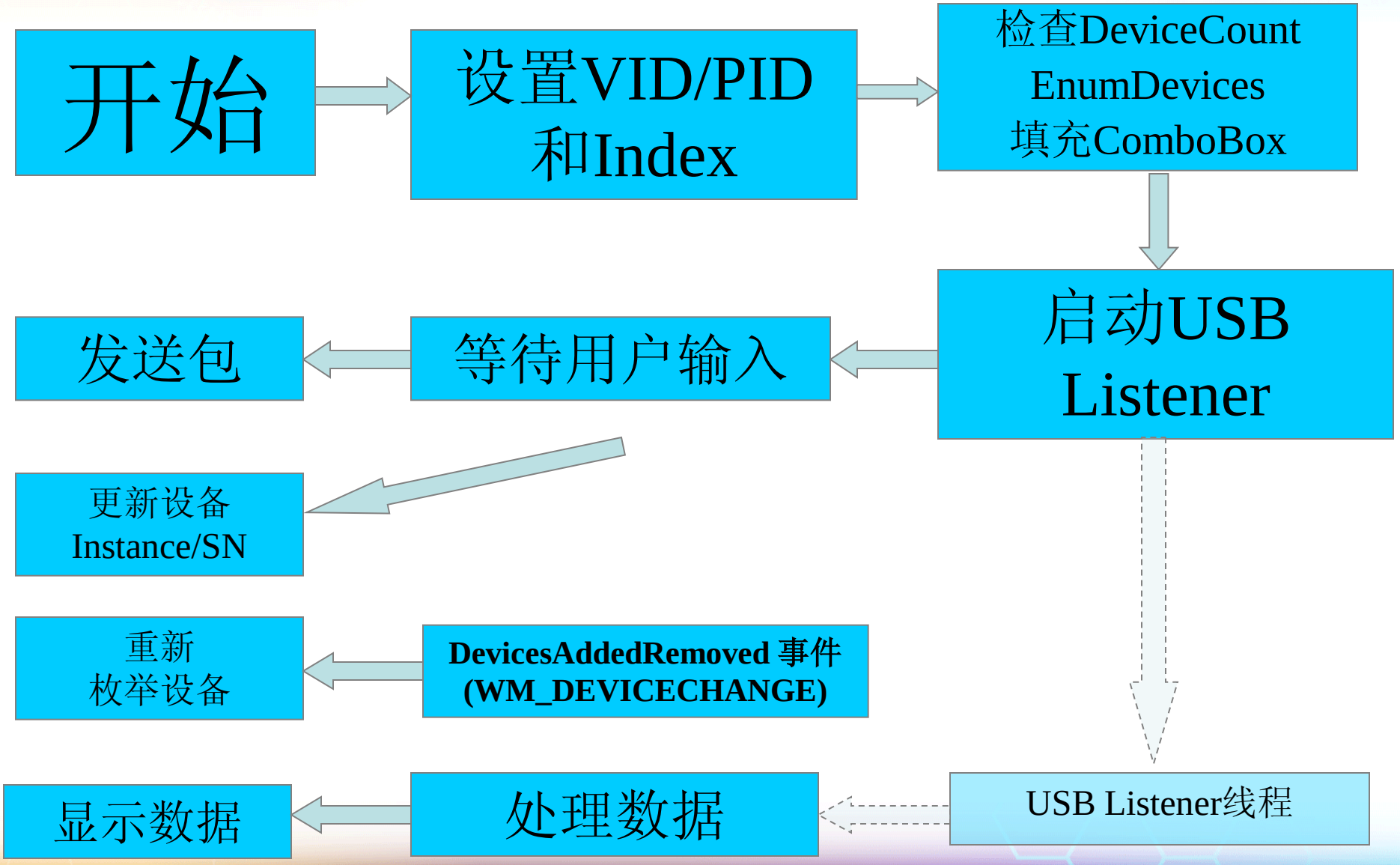
多设备通信

- **跟踪WM_DEVICECHANGE消息**

1. 创建子程序（无输入/输出）
   - **Sub DevicesChangedMessage()**
2. 添加对Form1_Load的事件处理程序初始化
   - **AddHandler USBDevice.DevicesAddedRemoved, AddressOf DevicesChangedMessage**

- 每当向系统添加或从系统移除新设备时，就将调用DevicesChangedMessage()子程序。

多设备通信



演示5
自动设备枚举
（多设备通信）

数据传输率

数据传输率概念

- USB数据传输率（USB Communicator应用）
  - 在中断（INT）模式，最大为64 KB/s
    - 修改USBListenServer，以使用MPUSBReadInt函数
    - 1 ms的间隔中未读取的话，将导致至少1个包的传输遗失（1 ms延时）
    - 或者
    - 增大MAX_READBUFFER_SIZE至6400字节（还应增大RxTimeout，使之超过100 ms）
  - 使用批量（BULK）传输
    - In和Out端点两者都必须是批量模式（嵌入式软件）
  - 避免/尽可能不更新表单控件/从数据回调函数把大量数据转换为字符串
    - 示例：在DisplayData子程序内，注释掉标记为“Start Speed Test” — “End Speed Test”的4行，将使吞吐量从17 KB/s提高至32 KB/s（中断模式）

数据传输率概念

- USB数据传输率（USB Communicator应用）
  - 请求读取多个数据包
    - 即，读6400字节而不是64字节
      - 系统资源限制了每秒的回调次数，更大的数据读取将最小化回调次数
      - 示例：在批量模式中，读取64字节变为6400字节，将使数据吞吐量从32 KB/s升至172.8 KB/s（把MAX_READBUFFER_SIZE常数改为6400，继续注释掉Speed Test行）
  - 使用重叠（Overlapped）读
    - 当前读操作完成后，把多个数据读请求设置成连续执行（最小化读操作之间的空闲时间）
  - 配合重叠读，使用乒乓缓冲区

总结

总结

- 基础/架构
  - 最多126个共享带宽的设备
  - 主机是主控设备
- 主机/设备通信
  - 事务
  - 传输
- 枚举
  - 描述符
- Microchip提供：MCU、演示板、固件、定制的驱动程序及培训

谢谢您！

请填写课程评估表

附录

USB 3.0

引言

- **USB 1.0:**
  - 1.5 Mbit/s（~183 KB/s）的低带宽速率
  - 非常类似于USB 1.1，但是传输每一位要花上8倍的时间
  - 用于节省成本（HID）

- **USB 1.1:**
  - 12 Mbit/s（~1.43 MB/s）的满带宽速率

- **USB 2.0:**
  - 480 Mbits/s（~57 MB/s）的高速率
  - 在需要时，所有的高速设备都能回到满带宽操作。相同的连接器。

- **USB 3.0:**
  - 4.8 Gbit/s（~572 MB/s）的超高速率。

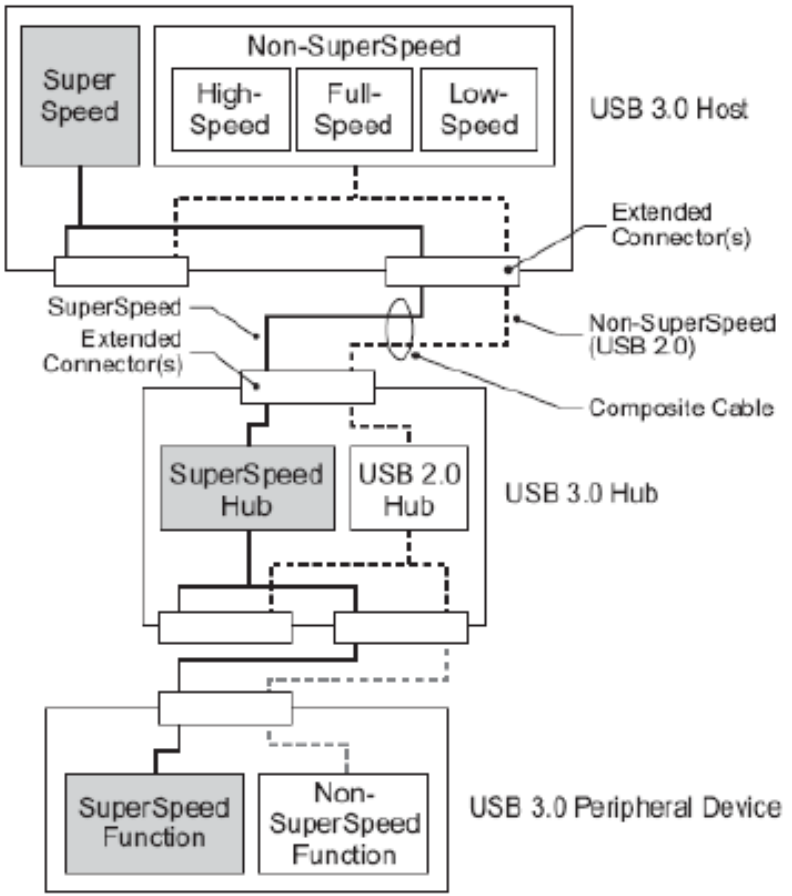
USB 3.0特性

- 与USB 2.0相比，性能提升10倍
- 向下兼容
  - 旧有设备插入新主机连接器时，仍能正常工作
  - 当新设备插入旧有系统时，仍能正常工作，不过工作在USB 2.0速度上
  - 现有的类驱动程序继续起作用
- 相同的USB设备模型
  - 管道模型 / USB框架 / 传输类型
- 高效的功率
  - 提供卓越的功率特性（尤其对闲置链路）
    - 在设备和平台上均提供
  - 不再需要轮询
- 可扩展
  - 设计协议时就考虑了支持高效的规模伸缩

	Song / Pic	256 Flash	USB Flash	SD-Movie	USB Flash	HD-Movie
	4 MB	256 MB	1 GB	6 GB	16 GB	25 GB
USB 1.0	5.3 sec	5.7 min	22 min	2.2 hr	5.9 hr	9.3 hr
USB 2.0	0.1 sec	8.5 sec	33 sec	3.3 min	8.9 min	13.9 min
USB 3.0	0.01 sec	0.8 sec	3.3 sec	20 sec	53.3 sec	70 sec

USB 3.0总线架构

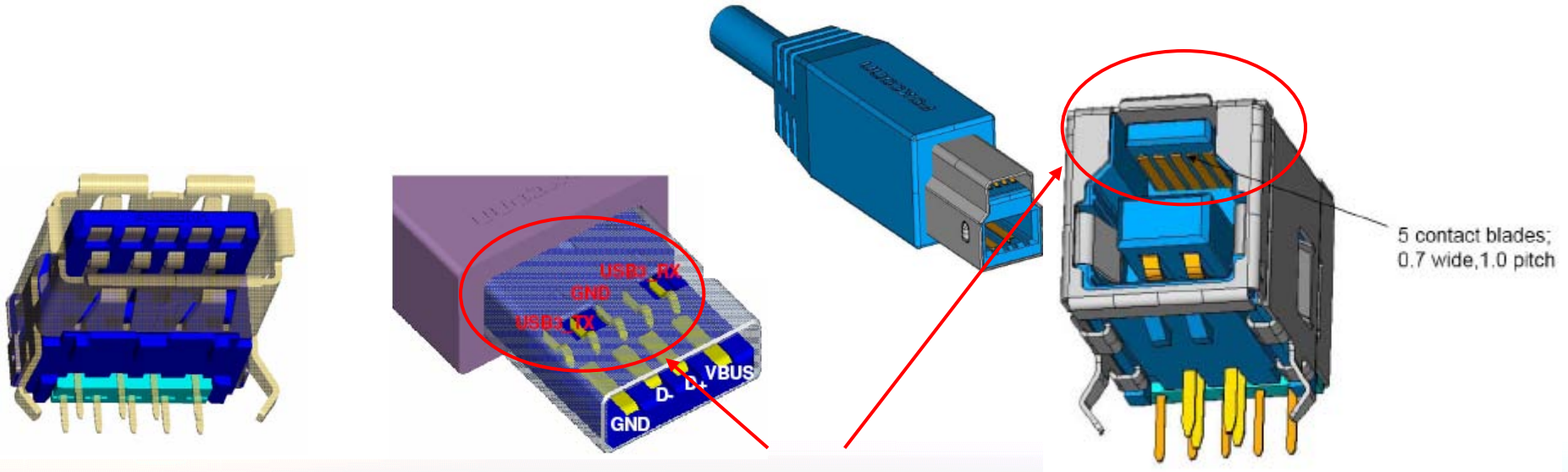
- 双总线架构，超高速总线，与USB 2.0同时工作
  - 电气/机械上向前向后兼容
  - 以最快的信号速度进行设备发现/配置
  - 集线器提供更多的连接点
- 超高速USB
  - 双单工信号
  - 包路由至设备
  - 集线器存储与转发
  - 异步通知



USB 3.0标准A/B型连接器

- 与USB 2.0标准A型连接器接口相同，但增加了用于超高速USB信号的引脚
- 完全兼容USB 2.0标准A型连接器
- 支持双排连接器

- 定义针对的是相对较大的、固定的外设，诸如HDD和打印机等
- 看上去就与USB 2.0标准B型连接器不同
  - 不过母头可接受USB 2.0标准B型的插入



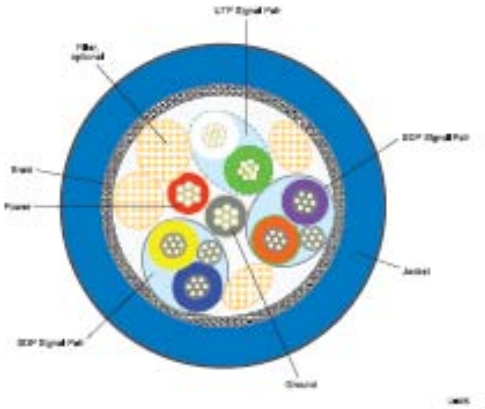
标准A型　　USB 3.0部分　　标准B型

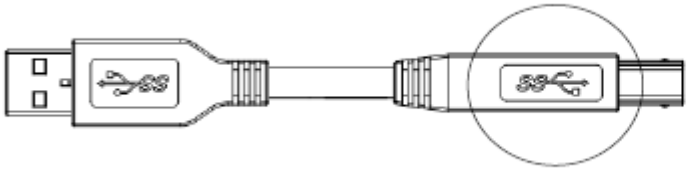
USB 3.0 Micro连接器系列

- 针对手持设备而定义
- 向后兼容**USB 2.0 Micro**连接器
- 基于**USB 2.0 Micro-B**型连接器，但具有用于超高速**USB**信号的扩展部分
- **USB 3.0 Micro-A**和**Micro-AB**型连接器与**USB Micro-B**型连接器几乎一样，除了定位销不同之外

线缆

- 用于USB 2.0的无屏蔽双绞线（UTP）线缆不能用于超高速USB
- 超高速USB需要使用屏蔽差分信号线对（SDP，双绞线或双股电缆）
  - 信号完整性以及EMI抑制（containment）





商标

Microchip的名称和徽标组合、Microchip徽标、dsPIC、KeeLoq、KeeLoq徽标、MPLAB、PIC、PICmicro、PICSTART、PIC32徽标、rfPIC及UNI/O均为Microchip Technology Incorporated在美国和其他国家或地区的注册商标。

FilterLab、Hampshire、HI-TECH C、Linear Active Thermistor、MXDEV、MXLAB、SEEVAL和The Embedded Control Solutions Company均为Microchip Technology Incorporated在美国的注册商标。

Analog-for-the-Digital Age、Application Maestro、CodeGuard、dsPICDEM、dsPICDEM.net、dsPICworks、dsSPEAK、ECAN、ECONOMONITOR、FanSense、HI-TIDE、In-Circuit Serial Programming、ICSP、Mindi、MiWi、MPASM、MPLAB Certified徽标、MPLIB、MPLINK、mTouch、Omniscient Code Generation、PICC、PICC-18、PICDEM、PICDEM.net、PICkit、PICtail、REAL ICE、rfLAB、Select Mode、Total Endurance、TSHARC、UniWinDriver、WiperLock和ZENA均为Microchip Technology Incorporated在美国和其他国家或地区的商标。

SQTP是Microchip Technology Incorporated在美国的服务标记。

在此提及的所有其他商标均为各持有公司所有。