

USB 101: 通用串行总线 2.0 简介

作者: Robert Murphy

相关产品系列: PSoC® 1/PSoC 3/PSoC 5LP/
PSoC 4200L/USB 控制器

相关代码示例: 请查阅[相关资源](#)

相关应用笔记: 请查阅[相关资源](#)

AN57294 描述了有关 USB 协议的基础信息, 着重对 USB 2.0 规范进行了讲述。本应用笔记的对象有两类: 使用 USB 实现嵌入式设计的新开发者; 需要使用并进一步了解赛普拉斯应用笔记的开发者。

目录

1	简介	2	12	USB 枚举和配置	30
2	USB 历史	2	12.1	动态检测	30
3	USB 概况	3	12.2	枚举	30
4	USB 架构	4	12.3	配置	31
5	物理接口	6	13	调试 USB 设计	31
6	USB 速度	10	13.1	在主机端进行调试	31
7	USB 电源	11	13.2	调试通信	33
8	USB 端点	13	13.3	设备端的调试	35
9	通信协议	15	14	获取 VID 和 PID	36
9.1	数据包类型	16	15	合规性测试	36
9.2	数据传输类型	18	15.1	USB-IF 合规性测试	36
10	USB 描述符	21	15.2	Microsoft 硬件认证测试	41
10.1	设备描述符	22	16	总结	42
10.2	配置描述符	23	17	相关资源	43
10.3	接口关联描述符 (IAD)	23	A	附录 A	44
10.4	接口描述符	24	A.1	示例 PSoC 3 全速 USB 设备描述符	44
10.5	端点描述符	25	B	附录 B	46
10.6	字符串描述符	26	B.1	USB 枚举的总线分析器捕获 (示例)	46
10.7	其他杂项描述符类型	26		文档修订记录	50
10.8	使用多个 USB 描述符	27		销售、解决方案以及法律信息	51
11	USB 类设备	28			

1 简介

USB 是一种接口，用于将设备连接到计算机上。通过该连接，计算机可以向设备发送数据或检索来自设备的数据。USB 为开发者提供了一个标准接口，适用于多种不同类型的应用。借助于系统的设计过程，USB 设备易于连接和使用。本应用笔记会使该过程变得更加简单。

以下是本应用笔记中会使用的概念：

- USB 历史
- USB 架构
- USB 物理接口
- USB 速度
- USB 电源
- USB 端点
- USB 通信协议
- USB 描述符
- USB 类设备
- USB 枚举与配置过程
- USB 合规性和 Windows Logo 测试

请注意，本应用笔记中未讨论 USB 3.0。欲了解有关 USB 3.0 的应用笔记，请查阅[相关资源](#)中所列出的应用笔记。另外，本文档也没有提供任何关于 USB 2.0 设备的代码示例。请参考[相关资源](#)部分中所列出的代码示例，以获得每个产品系列的代码示例链接。

2 USB 历史

USB 是一种行业标准，用于将电子外围设备（例如：键盘、鼠标、调制解调器和硬盘驱动器）连接到计算机上，它代替了尺寸大且速度慢的连接（例如：串行和并行端口）。该标准于 1994 年由 Compaq、DEC、IBM、Intel、Microsoft、NEC 和 Nortel 共同开发。目的是开发一个能适用于多种设备的接口替换掉当时存在的诸多不同连接器，并提高电子设备的数据吞吐量。

多年来，USB 规范已经经历了多次修改。初始版本是 USB 1.0 规范，于 1996 年完成。它仅支持两种速度模式：低速（LS）模式（支持 1.5 Mb/s）和全速（FS）模式（支持 12 Mb/s）。虽然低速比全速慢，但是它不易受到电磁干扰（EMI）的影响，并且可以使用成本低的组件，因此受到诸多 USB 设备开发者的喜爱。USB 1.1 规范于 1998 年面世，增加了一些说明，并改进了 USB 1.0 规范。截至 2000 年 4 月，USB 2.0 版本已经作出了巨大的修改。该版本增加了新的速度模式，即高速（HS）模式。它最高支持 480 Mb/s。该版本向后兼容 USB 1.1 和 1.0。于 2008 年 11 月发布的 USB 3.0 同样保持向后兼容性，传输速度最高达 5 Gb/s。这时，新的物理连接器也应运而生。最近，USB-IF 计划发布了 USB 3.1 规格，该版本将传输速度提高到 10 Gb/s。目前，USB 由 [USB 实施者论坛](#)（USB-IF）非盈利组织监管，该组织保管着 USB 文件和合规项目。

3 USB 概况

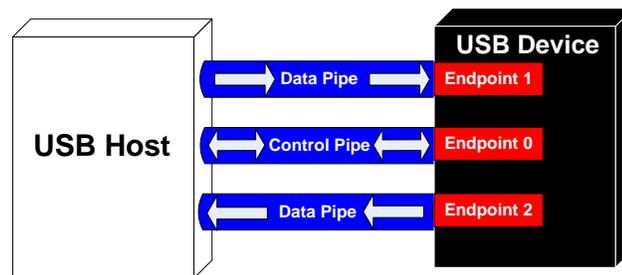
USB 系统包括一台主机（一般是一台个人计算机（PC））和多个通过分层星形拓扑连接的外围设备。该拓扑也可以包括集线器，从而能够提供更多与 USB 系统的连接点。主机本身包含两个组件，即主控制器和根集线器。主控制器是具有软件驱动器层的硬件芯片组，用于执行以下任务：

- 检测 USB 设备的插入和拔出
- 管理主机和设备间的数据流
- 提供并管理所连接设备的电源
- 监视总线上的活动

主机可以有一个或多个主控制器。通过使用外部 USB 集线器，每个控制器最多可以连接 127 个设备。根集线器是连接到主控制器的内部集线器，并且作为 USB 系统中的第一个连接层。在 PC 上，目前存在多个端口。这些端口是 PC 中根集线器的一部分。为简便起见，请通过一个被称为主机的“黑盒子”抽象视图来了解根集线器和主控制器。

USB 设备包括一个或多个设备功能，例如鼠标、键盘或音频设备。主机为每个设备提供了一个地址，用于设备与主机间的数据通信。USB 设备的通信通过管道实现。这些管道是主控制器到可寻址缓冲区（称为端点）间的连接路径。一个端点会保存收到来自主机的数据并保存将要发送给主机的数据。一个 USB 设备能够具有多个端点，并且每个端点都有相应的管道，如图 1 所示。

图 1. USB 管道模型



USB 系统中的管道共有两种，分别为控制管道和数据管道。USB 规范中定义了四种不同的数据传输类型。使用哪个管道由数据传输类型决定。

- **控制传输：**用于将指令发送到设备上、进行查询并且配置设备。该传输使用了控制管道。
- **中断传输：**用于发送少量的突发性数据，并且保证传输延迟最小。该传输使用了数据管道。
- **批量传输：**利用了全部可用的 USB 带宽来传输大量数据，但传输速度或延迟得不到保证。该传输使用了数据管道。
- **同步传输：**数据传输采用了得到保证的传输速率。随着传输延迟和总线带宽的保证，传输时间也得到保证。同步传输没有错误纠正功能，因此在重新发送有误的数据包过程中，不能停止传输。该传输使用了数据管道。

每个设备都有一个控制管道，用于控制发送给设备或从设备接收信息的状况。设备可以有任意个数据管道，通过中断、批量或同步传输类型进行数据传输。控制管道是 USB 系统中唯一一个双向管道，所有的数据管道均是单向的。

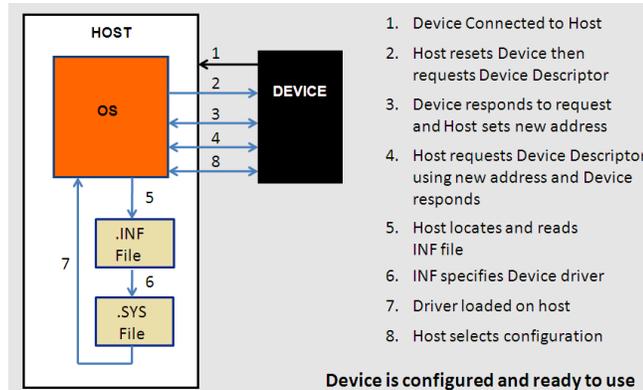
每个端点均可通过设备地址（由主机分配）和端点编号（由设备分配）进行访问。信息被发送给设备时，会通过令牌数据包来识别设备地址和端点编号（通信协议一节介绍了详细内容）。在传输数据前，主机将发送该令牌数据包。

USB 设备首次与主机相连时，将启动 USB 枚举过程。枚举是设备和主机间进行的信息交换过程，包含用于识别设备的信息。此外，枚举过程还分配设备地址、读取描述符（作为提供有关设备信息的数据结构），并分配和加载设备驱动程序。整个过程需要数秒时间。更多有关信息，请参考 USB 枚举和配置一节。完成该过程后，设备可以向主机传输数据。图 2 显示的是通用枚举过程流程图。两个文件属于主机端，用于枚举和加载驱动程序过程。

- **.INF** — 包含了安装设备时所需全部信息（驱动程序的名称和位置、Windows 注册信息和驱动程序版本信息）的文本文件。

- **.SYS** — 驱动程序需要该文件才可有效与 USB 设备进行通信。

图 2. 枚举事件序列



设备被枚举后，主机将负责总线上的全部设备之间的数据通信流向。因此，如果没有主控制器的请求，所有设备均无法传输数据。

4 USB 架构

系统中只能有一个主机，并且与设备进行的通信是从主机的角度进行的。主机是“上行”组件，设备则是“下行”组件，如图 3 表示。数据从主机转移到外设的操作是 OUT 传输。数据从外设转移到主机的操作是 IN 传输。主机（尤其是主控制器）控制着所有通信并向设备发出指令。共有三种常见的 USB 主控制器：

通用主控制器接口（UHCI）：由 Intel 生产，适用于 USB 1.0 和 USB 1.1。使用 UHCI 时需要得到 Intel 的许可。该控制器支持低速模式和全速模式。

开放主控制器接口（OHCI）：由 Compaq、Microsoft 和 National Semiconductor 生产，适用于 USB 1.0 和 1.1。该控制器支持低速模式和全速模式，并且它的效率比 UHCI 更高，因为可以执行更多硬件功能。

扩展型主控制器接口（EHCI）：在 USB-IF 要求发布单一主控制器规范后，已经生产了该控制器，它适用于 USB 2.0。EHCI 仅支持高速传输，并且将低速和全速传输委托给 OHCI 或 UHCI 控制器执行。

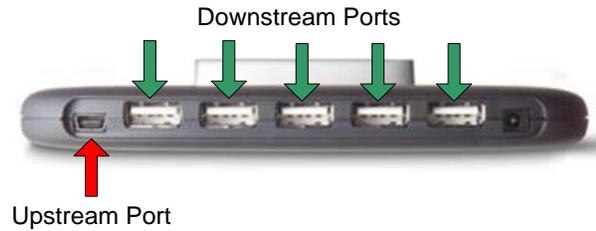
图 3. 多个外设同一个主机相连



可以将一个或多个设备连接至一个主机。每个设备均有一个地址，并且会对寻址它的主机指令做出响应。设备预计具有某种形式的功能，并不简单作为一个被动组件。设备具有一个上行端口。端口是设备上的 USB 物理连接点。

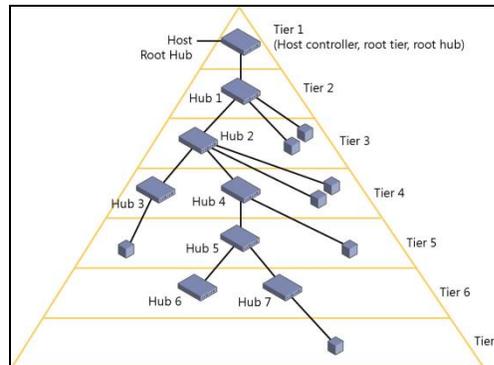
集线器是一个专用设备，允许主机同总线上的多个外设进行通信。与 USB 外设（例如鼠标）具有实际功能不同，集线器设备是透明的，并且作为直通连接使用。集线器也作为主机和设备间的通道。集线器具有多个连接点，从而可以将多个设备连接到一个主机上。一个集线器可以将与下行设备进行的通信重复使用到一个上行端口和最多七个下行端口。但集线器并没有主机功能。

图 4. 集线器连接



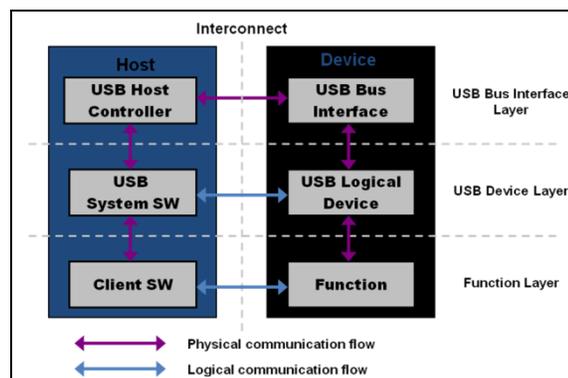
如上所述，通过使用集线器最多能够将 127 个设备连接至主控制器上。连接设备的数目限制由 USB 协议决定，它限制设备地址为 7 位。另外，由于集线器的时间限制和电缆传播的延迟，因此最多只能将五个集线器链接在一起。图 5 显示的是 USB 层次系统的框图，它表示集线器和设备的链接限制。您可以看到，随着集线器的链接限制，层次系统也限制为七层。

图 5. USB 层次连接



从另一个角度观察 USB 接口，即将其分为不同层次，如图 6 所示。总线接口层提供了物理连接、电气信号和数据包连接。该层由设备硬件处理，并通过设备的外部接口完成。设备层是 USB 系统软件的视图，用于执行 USB 操作，如发送和接收信息。该层的操作由设备的内部串行接口引擎完成。最后，功能层包括有关软件的事务。它是 USB 设备的一部分，用于处理所接收到的信息，或者收集数据并将其传输给主机。图 6 表示的是该抽象接口。

图 6. 抽象接口

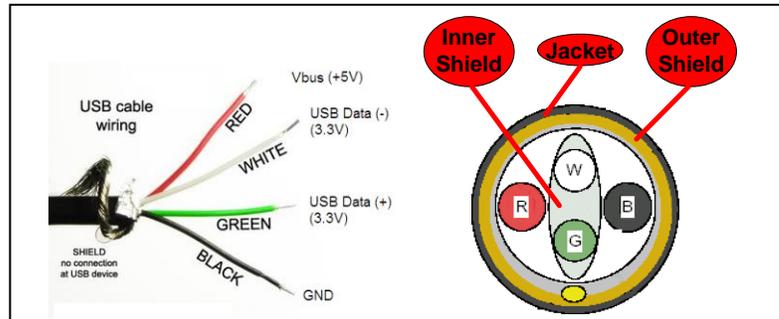


5 物理接口

从高级概述角度来看，USB 的物理接口具有两个组件：线缆和连接器。这些连接器将设备连接到主机上。

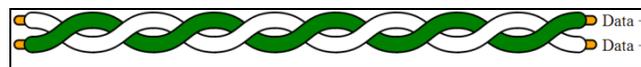
一个 USB 线缆包含由一个绝缘套保护的多个组件。该绝缘套下面是一个包含了一个带有铜面的外部扩展板。外部扩展板内包含多个连线：一个铜排流线、一个 V_{BUS} 线（红色）和一个接地线（黑色）。由铝制成的内部扩展板包含一对用双绞线制成的数据线，如图 7 所示。有一个 D+ 线（绿色）和一个 D- 线（白色）。

图 7. USB 线缆内部



在全速和高速设备内，最大线缆长度为 5 m。要想增大主机和设备间的距离，您必须使用一系列集线器和 5 m 长的线缆。市场上存在多种 USB 扩展线缆，但使用超过 5 m 的线缆违反了 USB 规范。低速设备的规范不太一样。它们的线缆长度被限制为 3 m，并且不需要使用双绞线，如图 8 所示。

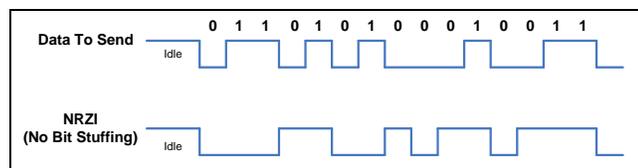
图 8. USB 双绞线数据线



V_{BUS} 线为所有相连设备提供了恒定的 4.40 ~ 5.25 V 电源。当 USB 为设备提供 5.25 V 电源时，数据线（D+ 和 D-）在 3.3 V 电压下工作。USB 接口使用不归零反转（NRZI）的差分传输，信号使用位填充方法进行编码并通过双绞线传输。

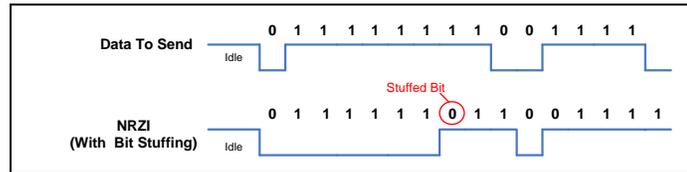
NRZI 编码是一种映射一个二进制信号的方法，以便通过某个介质（在这里是 USB 线缆）传输该信号。在该编码方案中，如果电压电平不变，则表示逻辑 1；如果电压电平变化，则表示逻辑 0，如图 9 所示。顶部是即将通过 USB 传送的数据。底部是编码的 NRZI 数据。

图 9. 将数据设置为 NRZI 编码的逻辑电平



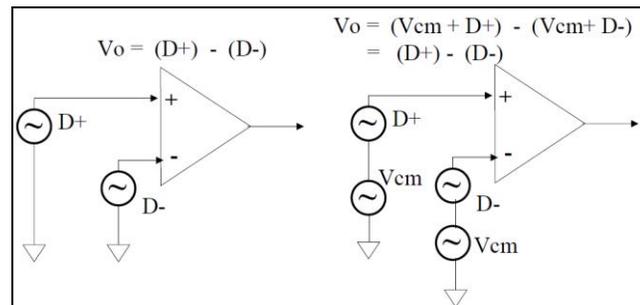
通过在 7 个连续的逻辑 1 后面插入一个逻辑 0 可以实现位填充。位填充是为了通过保持锁相环（PLL）对 USB 硬件进行同步化。如果该数据内有太多的逻辑 1，那么 NRZI 编码流中将没有足够用于实现同步化的转换。USB 硬件上的接收器会自动检测额外位，并忽略它。该额外位填充是引起 USB 上的额外开销的原因。图 10 显示的是一个带有位填充的 NRZI 数据的示例。请注意，“Data to Send”（将要发送的数据）流为 8 个逻辑 1。在该编码数据中，在第六个逻辑 1 后面插入了一个逻辑 0。这样，第七和第八个逻辑 1 将位于逻辑 0 后。

图 10. 将数据置为使用位填充方法的 NRZI 编码的逻辑电平



接收到任何数据后以及发送任何数据前，USB 设备中的硬件将处理所有编码和位填充。使用差分 D+和 D-信号是为了抑制共模噪声。如果噪声被耦合到该线缆内，它将出现在该线缆中所有传输线上。如果使用 USB 硬件中的差分的放大器（该 USB 硬件在主机和设备内使用），则可以抑制共模噪声，如图 11 所示。

图 11. USB 输入差分放大器缓冲器



USB 通信用途经过了 D+和 D-线上的各种不同信号状态。某些状态发送数据，而其他则作为特殊信号状态使用。下面内容介绍了这些状态，它们的参考列表如表 1 所示。

差分 0 和差分 1：这两个状态用于通过 USB 进行的通用数据通信。当 D+线为高电平、D-线为低电平时，该状态为差分 1。当 D+线为低电平、D-线为高电平时，该状态为差分 0。USB 数据通信的示例如图 12 所示。

J 状态和 K 状态：除了差分信号外，USB 规范还定义了其他两个差分状态：J 状态和 K 状态。它们的定义由设备速度决定。在全速和高速设备上，J 状态为差分 1 而 K 状态是差分 0。在低速设备上，该情况则相反。

单端 0 (SE0)：在 D+和 D-均为低电平时所发生的状态。该状态表示一个复位、断连或数据包的结束。

单端 1 (SE1)：在 D+和 D-均为高电平时发生的状态。不会故意生成该状态，并且不能在 USB 设计中出现。

闲置：必须在发送一个数据包的前后发生的状态。如果一个数据线为低电平，而另一个数据线为高电平，则表示闲置状态。高电平和低电平的定义由设备的速度决定。在全速设备上，闲置状态是指 D+为高电平、D-为低电平。在低速设备上，该情况则相反。

恢复：用于使设备从挂起状态唤醒。通过发送一个 K 状态实现该操作。

数据包的开始 (SOP)：当 D+和 D-线从闲置状态转换到 K 状态时，将在开始低速或全速数据包前发生。

数据包的结束 (EOP)：在低速或全速数据包结束时发生。当 SE0 状态持续两位时间（后面的内容将介绍位时间）以及 J 状态持续 1 位时间时，将发生 EOP。

复位：在 SE0 状态持续 10 ms 时发生。在 SE0 至少持续 2.5 ms 后，该设备会复位，并开始进入复位状态。

保持活动 (Keep Alive)：在低速设备中使用的信号。低速设备缺少了一个帧起始数据包（用于防止挂起状态）。每次经过 1 ms，它们都会使用一个 EOP 来防止设备进入挂起状态。

图 12. USB D+和 D-的通信

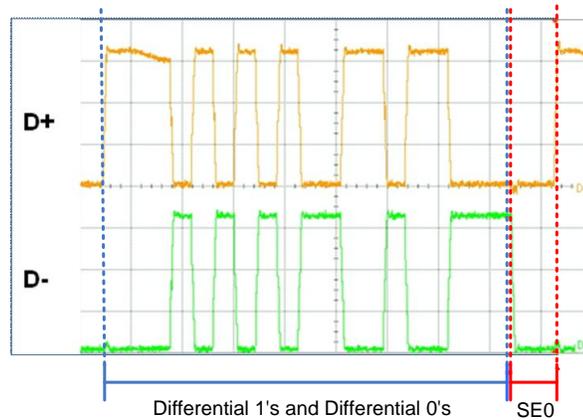


表 1. USB 通信的状态

总线状态	指示
差分 1	D+为高电平, D-为低电平
差分 0	D+为低电平, D-为高电平
单端 0 (SE0)	D+和 D-为低电平
单端 1 (SE1)	D+和 D-为高电平
J 状态:	
低速	差分 0
全速	差分 1
高速	差分 1
K 状态:	
低速	差分 1
全速	差分 0
高速	差分 0
恢复状态:	K 状态
数据包开始 (SOP)	数据线从闲置状态切换到 K 状态。
数据包结束 (EOP)	SE0 持续两位时间以及 J 状态持续 1 位时间。

图 13 和图 14 显示的是可用的不同 USB 端口和连接器。上行连接始终使用 Type A 型端口和连接器, 而设备使用 Type B 型端口和连接器。最初, USB 规范仅包含用于设备的更大的 Type A 型和 Type B 型连接器, 后来提供了 Mini 和 Micro 连接器。这些 Mini 和 Micro 连接器最初是为 USB On-the-Go (USB OTG) 开发的。USB OTG 是一个 USB 规范, 允许将通常作为从设备的设备作为主机使用。这便是图 14 将 Mini 和 Micro 端口显示为 Mini-AB 和 Micro-AB 的原因。然而, 由于 Mini-B 和 Micro-B 连接器比 Type B 型更小, 因此许多电子设备都采用了该型连接器 (尽管会降低 USB OTG 功能)。

图 13. USB 连接器尺寸比较

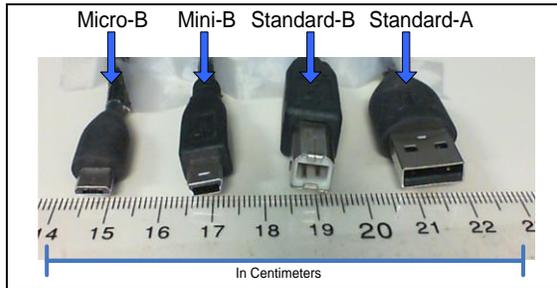


图 14. USB 端口和连接器

Type	Port Image	Connector Image
Type A		
Type B		
Mini-AB		
Mini-B		
Micro-AB		
Micro-B		

图 14 显示的 Mini 和 Micro 连接器具有五个（而不是 4 个）引脚。额外引脚是 ID 引脚，用于识别 OTG 应用中的主机和设备。由于 PSoC 不支持 USB OTG，因此本应用笔记并没提供它的相关信息。使用各种不同连接类型（Type A 和 Type B 型）的原因是为了在集线器上防止发生环回连接。某些 USB 设备包含一个电容式线缆或连接线，并且唯一的可见连接器为 Type A 型。表 2 和表 3 显示的是由连接器类型决定的 USB 连接器的引脚分布。

表 2. USB 标准的连接器引脚分布

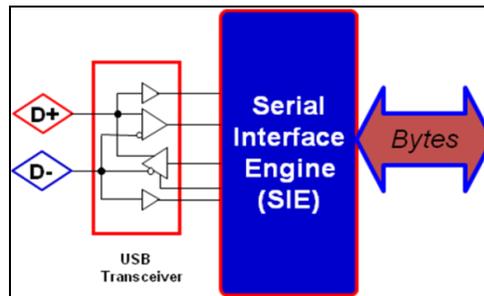
引脚	名称	颜色	功能
1	V _{BUS}	红色	+5 V
2	D-	白色	数据 (-)
3	D+	绿色	数据 (+)
4	GND	黑色	接地

表 3. USB Mini/Micro 连接器引脚分布

引脚	名称	颜色	功能
1	V _{BUS}	红色	+5 V
2	D-	白色	数据 (-)
3	D+	绿色	数据 (+)
4	ID	NA	识别 Type A 型和 Type B 型插座： A 插座： 连接到接地信号 B 插座： 未连接
5	GND	黑色	接地

需要将两个主硬件模块连接到 USB 上：一个收发器（又被称为 PHY — 物理层），一个串行接口引擎（又被称为 SIE）。该收发器提供了 USB 连接器和芯片电路（用于控制 USB 通信）间的硬件接口。SIE 是 USB 硬件的内核。它执行多种功能，如解码和编码 USB 数据、错误纠正、位填充和发信号。SIE 可以采取不同形式。与收发器不同，它们不受 USB 规范的限制。实际上，有些设备使用基于软件的 SIE 以降低成本，但也有其他设备使用基于硬件的 SIE。

图 15. 设备上的 USB 硬件接口



6 USB 速度

USB 规范已经为 USB 系统定义了以下四种速度模式：低速（Low-Speed）、全速（Full-Speed）、高速（Hi-Speed）和超高速（SuperSpeed）。目前，赛普拉斯对 PsoC 器件系列仅支持全速模式，另外对于各种专用 USB 设备则支持低速、高速和超高速等模式。因此，本应用笔记将重点介绍这三种速度模式。

图 16. USB 传输速度



新型主机一直能同低速设备进行通信。例如，高速主机能够与低速设备进行通信，但全速主机并不能同高速设备进行通信。

低速、全速和高速设备的速率分别为 1.5 Mb/s、12 Mb/s 和 480 Mb/s。但是，这些指的是总线速率，并不是数据速率。实际的数据速率受总线加载速度、传输类型、开销、操作系统等因素的影响。数据传输则受以下内容的限制：

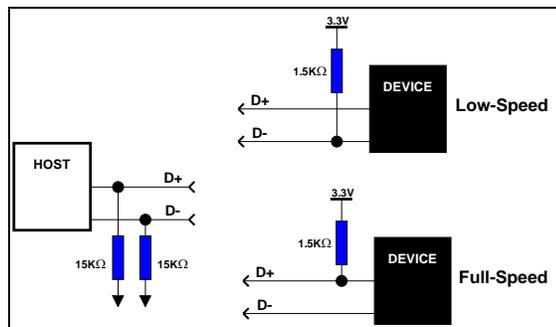
- 低速设备
 - 例如：键盘、鼠标和游戏等外设
 - 总线速率：1.5 Mb/s
 - 最大的有效数据速率：800 B/s
- 全速设备
 - 例如：手机、音频设备和压缩视频
 - 总线速率：12 Mb/s
 - 最大的有效数据速率：1.2 MB/s
- 高速设备
 - 例如：视频、影像和存储设备
 - 总线速率：480 Mb/s
 - 最大的有效数据速率：53 MB/s

建立好了 USB 设备和主机间的连接后，需要使用 D+ 或 D- 信号线上的上拉电阻来检测设备的速度。D+ 信号线上的 1.5 kΩ 大小的上拉电阻表示所连接的是一个全速设备，D- 线上 1.5 kΩ 大小的上拉电阻表示所连接的是一个低速设备，如图 17 所示。

高速设备都是作为全速设备进行初始化，因此它们在 D+ 信号线上也使用了一个 1.5 kΩ 大小的上拉电阻。设备连接好后，会在枚举的复位阶段中发出 J 状态和 K 状态序列。如果集线器支持高速设备，则不需要使用上拉电阻。

USB 进行枚举时需要使用上拉电阻。否则，USB 会认为总线上没有连接任意设备。部分设备要求在 D+/D- 信号线上使用一个外部上拉电阻。但是 PSoC 已经带有所需的内部上拉电阻，因此不需要外部上拉电阻。

图 17. USB 速度检测



USB 2.0 设备经常被误解为高速 USB 设备。所有高速设备都符合 USB 2.0 规范，这是因为 USB 2.0 规范支持高速模式。USB 2.0 规范还包含了全速和低速设备。

这些速度也影响到有关位时间的 USB 信号（如数据包结束（EOP）信号）。低速和全速 USB 设备使用了频率为 48 MHz 的时钟执行 SIE 操作，并执行使用其他时钟源的 USB 操作。该 48 MHz 时钟和总线速度决定了 USB 位时间：

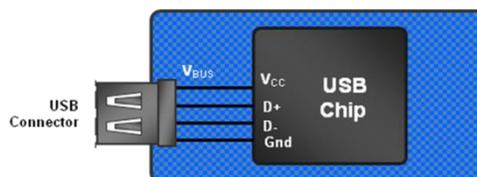
- **全速：** 时钟频率/总线速度 = 48 MHz / 12 Mb/s 时，USB 位时间为 4 个时钟周期。
- **全速：** 时钟频率/总线速度 = 48 MHz / 1.5 Mb/s 时，USB 位时间为 4 个时钟周期。

7 USB 电源

作为 USB 电源时，USB 设备可被划分为两种设备类型：总线供电和自供电。

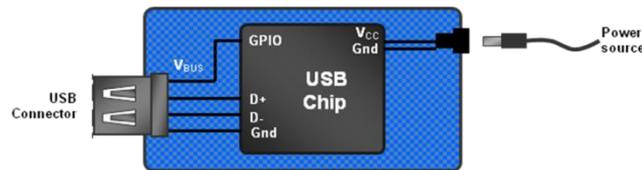
总线供电是 USB 设计的一个优势。由于设备通过总线供电，因此不需要使用笨重的内部或外部电源，它仍能够维持自身操作。总线可由主机或集线器供电。使用某个总线供电的设备时，用户将设备配置为某种状态前必须考虑其功耗。即设备枚举完成后，必须在第一次将设备连接到总线到主机将 SET_CONFIGURATION 命令传输给设备的这段时间内检查其功耗。设备被配置前不能消耗超过 100 mA 的电流（即 USB 规范中为低速、全速或高速设备定义为一个负载单位）。在配置过程中，设备要求一个预算功耗。总线供电的设备共有以下两种：高功耗和低功耗设备。低功耗设备最多消耗 100 mA 的电流，高功耗设备最多消耗 500 mA 的电流。消耗的电流超过 500 mA 的设备要自供电。

图 18. USB 总线供电设备



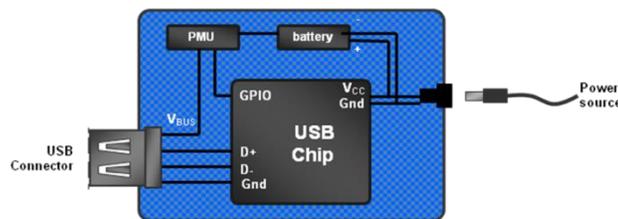
自供电设备通过使用外部电源（如直流电源适配器或电池）为自己供电。自供电设备在进行设计的过程中需要考虑到一些注意事项。USB 规范要求自供电设备一直监控自己的 V_{BUS} 线。 V_{BUS} 不存在的时间内，设备必须断开提供给 $D+/D-$ 线上的上拉电阻的电源，从而防止向主机或集线器供电。否则，会导致 USB 合规性测试发生失败。但是自供电集线器能够从总线获得最多 100 mA 的电流。

图 19. USB 自供电设备



设备还能结合两种电源模式，并成为总线供电和自供电的设备。常见的例子是设备使用电池。设备通常是自供电的；但使用 V_{BUS} 给电池充电，并且在电池电量发生变化时给设备供电。在技术方面，该设备是一个自供电设备，如 USB 描述符中显示，但该设备仍要求来自主机的预算电源。同自供电设备相似，这些混合设计中仍需要监控 V_{BUS} 大小，并且仍会断开提供给 $D+/D-$ 线上的上拉电阻的电源。在本应用中，需要实现一部分电源管理系统类型，以监控电池的电压、充电状态，并控制电池电源和外部电源间的切换。

图 20. USB 混合式供电设备



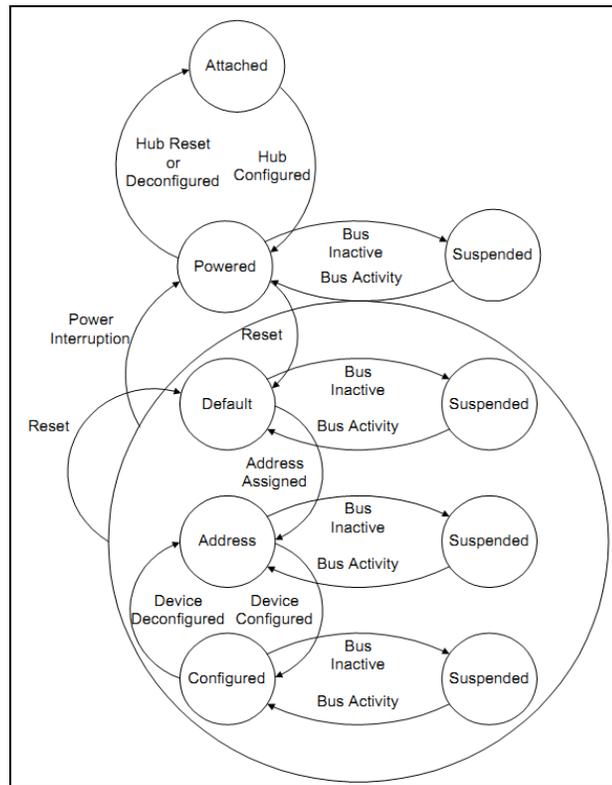
此外，无论设备的供电方式如何，所有 USB 设备都必须考虑到它们的暂停电流。设备的暂停电流是指在主机处于挂起模式（又称待机模式）时由 V_{BUS} 提供的电流。如果总线在 3 ms 时间内没有进行任意操作，设备会进入挂起模式。即使没有进行任意数据传输操作，主机仍会发出“帧开始”（SOF）令牌，以防止设备进入挂起模式。但低速设备却没有 SOF 数据包，因此这种设备是一个例外。总线上不进行低速数据的传输时，低速设备每经过 1 m 都会发送“数据包结束”（EOP）信号，将其作为“保持活动”信号。总线休闲时，设备必须进入挂起模式，并消耗不大于 2.5 mA 的电流。为满足该要求，设计师必须保证设备进入挂起状态前已经关闭了各个 LED 和其他电源库。一旦检测到总线上发生任何操作，USB 设备都会退出挂起状态。如果设备具有远程唤醒功能，它可以向主机发送恢复请求，然后等待主机确认该请求，而不是等待主机进行恢复。更多有关停止电流的信息，请参考 USB 规范中第 11.4.3 节的内容。

设计师需要了解各种同 USB 电源相对应的 USB 状态。这些状态通常出现在 USB 文档中，并适用于 USB 设备的枚举操作。

- **连接状态：** 当将某个设备插入到主机/集线器，但主机/集线器不给 V_{BUS} 线供电时，会出现这种状态。它通常在集线器检测到一个过流事件时出现。虽然仍连接着设备，但主机移除了供给它的电源。
- **供电：** 某个设备被连接到 USB 上并得到供电，但仍未接收到一个复位请求。
- **默认：** 某个设备被连接到 USB 上、得到供电，并且由主机进行了复位。这时，设备没有任何设备地址。设备会响应地址 0。
- **地址：** 某个设备被连接到 USB、得到供电、被复位，并且有一个唯一的地址。但是设备仍未得到配置。
- **配置：** 设备已经连接到 USB、得到供电、被复位、具有唯一的地址、得到配置，但尚未进入挂起状态。此时，总线供电设备能够消耗超过 100 mA 的电流。
- **挂起：** 如上面所述，设备已经建立好了连接，并且得到配置，但在 3 ms 时间内不会进行任意总线操作。

USB 规范（USB 规范中的图 9-1）具有一个框图，描述了这些电源模式的相关性和切换性。请参看图 21。

图 21. 设备状态框图



对于低速、全速和高速 USB 设备，USB 的功耗会以 2 mA 的单位进行枚举。例如，需要消耗 100 mA 电流的全速设备在进行枚举时将发送数值 50。

开发 USB 设计时，请考虑您的设备消耗总线的电流。根集线器由主机 PC 的电源供电。如果主机被连接到交流电源上，则 USB 规范要求主机为集线器上的每一个端口提供 500 mA 的电流。这样能将总线供电设备上的电流消耗限制在 500 mA。如果主机 PC 由电池供电，它可以为集线器上的每一端口提供 100 mA 或 500 mA 的电流。将设备插入到总线供电的集线器时，该设备必须是低功耗设备，并且消耗电流不能超过 100 mA。总线供电的集线器共有 500 mA 的电流可供给所有所连接的设备使用。

8 USB 端点

根据 USB 规范，设备端点是 USB 设备中一个独特的可寻址部分，它作为主机和设备间通信流的信息源或库。USB 枚举和配置一节介绍了设备向默认地址做出响应的步骤。枚举过程中，该事件在主机读取端点描述符等其他描述符信息之前发生。在该过程中，需要使用一套专用的端点用于与设备进行通信。这些专用的端点（统称为控制端点或端点 0）被定义为端点 0 IN 和端点 0 OUT。虽然端点 0 IN 和端点 0 OUT 是两个不同的端点，但对开发者来说，它们的构建和运行方式是一样的。每一个 USB 设备都需要支持端点 0。因此，该端点不需要使用独立的描述符。

除了端点 0 外，特定设备所支持的端点数量将由各自的设计要求决定。简单的设计（如鼠标）可能仅要一个 IN 端点。复杂的设计可能需要多个数据端点。USB 规范对高速和全速设备的端点数量进行了限制，即每个方向最多使用 16 个端点（16 个 IN、16 个 OUT，总共为 32 个），其中不包含控制端点 0 IN 和 0 OUT 在内。低速设备仅能使用两个端点。USB 类设备可对端点数量设定更严格的限制。例如，低速人机界面设备（HID）设计的端点可能不超过两个——通常有一个 IN 端点和一个 OUT 端点。数据端点本身具有双向特性。只有对它们进行配置后才支持单向传输（具有单向特性）。例如，端点 1 可作为 IN 或 OUT 端点使用。设备的描述符将正式使其成为一个 IN 端点。

各端点使用循环冗余校验（CRC）来检测传输中发生的错误。CRC 是一个用于检测错误的计算值。USB 规范中对实际的计算公式进行了解释，这些计算由 USB 硬件进行，这样可确保能够发出正确的响应。数据操作的接收方对数据进行 CRC 检查。如果两者匹配，那么接收方将发出一个 ACK。如果两者匹配失败，便不会发出任何握手数据包。在这种情况下，发送方将重新发送数据。

USB 规范定义了四种端点，并根据类型以及所支持的设备速度限制了数据包的大小。根据设计要求，开发者使用端点描述符指出端点类型以及数据包最大尺寸。四种端点和各自的特性如下：

控制端点 — 这些端点支持控制传输（即所有设备支持的传输）。控制传输通过总线发送和接收设备的信息。它的优点是保证传输准确。它能够立即检测到错误的发生，并重新发送数据。控制传输在低速和全速设备上使用 10% 的保留带宽（在高速设备上为 20%）并提供 USB 系统级控制。

中断端点 — 这些端点支持中断传输。这种传输非常适合需要使用高度可靠的方式来传输少量数据的设备。它通常用于 HID 设计。这种传输的名称可引起误会。实际上，它并不是一个中断，但使用了一个轮询率。进行该传输时，主机将在预计时间间隔内检查数据。通过及时检测错误并重新传输数据，该传输可确保数据操作的准确性。在低速和全速设备上，中断传输使用带宽的 90%，而在高速设备上，所用的带宽为 80%。同步端点与其共享该带宽。

中断端点的数据包最大尺寸与设备的速度相关。高速设备支持最大为 1024 字节的数据包。全速设备支持最大为 64 字节的数据包。低速设备支持最大为 8 字节的数据包。

批量端点 — 这些端点支持批量传输，即是在高度可变的时间内传输大量数据并且可用任何带宽空间的传输。它们是 USB 设备的最通用传输类型。因为用于批量传输的带宽并不是固定的，该传输的传送时间也是可变的。传送时间取决于总线上的可用带宽，由于该因素，便不能预期实际的传送时间。通过及时检测错误并重新传输数据，该传输可确保数据操作的准确性。批量传输非常适合对时间没有严格要求的大量数据传输。

批量端点的数据包最大尺寸与设备速度相关。高速设备支持最大为 512 字节的数据包。全速设备支持最大为 64 字节的数据包。低速设备不支持批量传输。

同步端点 — 这些端点支持同步传输，即具有预定带宽的连续性实时传输。由于同步传输没有错误恢复机制和握手数据包，它们需要支持容忍错误的数流。错误由 CRC 字段检测，但不会被修改。因此，同步传输可保证传输速度，但以数据的准确性作为代价。流式音乐或视频即是使用同步端点的应用示例，因为我们的耳朵和眼睛通常忽略偶尔被错过的数据。在低速和全速设备上，同步传输使用带宽的 90%（在高速设备上，所用的带宽为 80%），中断传输与其共享该带宽。

高速设备支持最大为 1024 字节的数据包。全速设备支持最大为 1023 字节的数据包。低速设备不支持同步传输。有关同步传输，请注意一些重点内容。为了保证数据传输，您通常需要使用三个缓冲区，一个正在传输数据、一个已加载数据和正在加载数据。

表 4. 端点传输类型特性

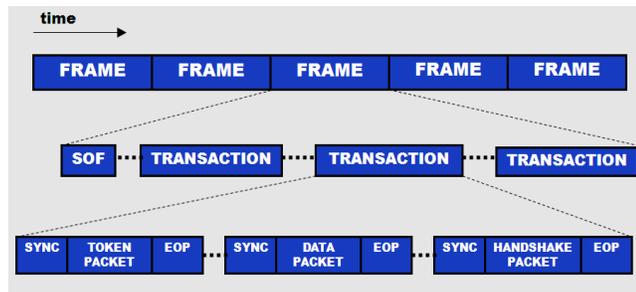
传输类型	控制	中断	批量	同步
适用场合	设备初始化和管管理	鼠标和键盘	打印机和批量存储	流式音频和视频
支持低速	有	有	无	无
修改错误	有	有	有	无
保证传输速度	无	无	无	有
使用固定带宽	有（10%）	有（90%） ^[1]	无	有（90%） ^[1]
减少延迟时间	无	有	无	有
传输的最大尺寸	64 字节	64 字节	64 字节	1023 字节（FS） 1024 字节（HS）
传输的最高速度	832 KB/s	1.216 MB/s	1.216 MB/s	1.023 MB/s

^[1]同步和中断端点的共享带宽。

9 通信协议

从时间角度来看，USB 通信由一系列帧构成。每一帧都有一个帧开始（SOF），随后是一个或多个数据操作。每一个数据操作都由一系列数据包构成。一个数据包由一个同步信号开始，结尾是一个数据包结束（EOP）信号。一个数据操作至少有一个令牌数据包。具体的数据操作可能有一个或多个数据数据包；一些数据操作可能会有一个握手数据包，也可能没有任何握手数据包。

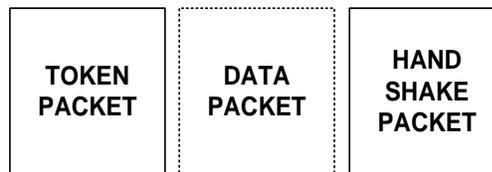
图 22. 从时间角度观察 USB 通信



数据操作是对数据包进行交换的操作，该操作使用了三种不同的数据包：一个令牌数据包、一个数据数据包（可选的）和一个握手数据包。

这些操作都在各个帧内进行，始终不会超过帧（除了高速同步传输以外）或终止其他数据操作。图 23 显示了数据操作的框图。

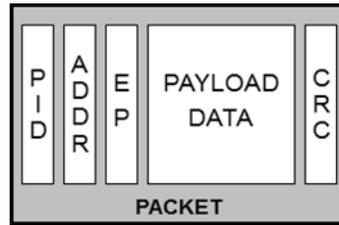
图 23. 数据操作框图



每个数据包可能带有不同的信息块。所带有的信息会因数据包类型的不同而异。下面列出了数据包可包含的各种信息。数据包的结构如图 24 所示。图 24 可作为数据包的模板；可从这里添加或提取信息。

- **数据包 ID (PID)** — (8 位：4 个类型位和 4 个错误检测位)。这些位将数据传输定义为 IN/OUT/SETUP/SOF
- **可选的设备地址** — (7 位：最多可支持 127 个设备)
- **可选的端点地址** — (4 位：最多支持 16 个端点)。USB 规范支持多达 32 个端点。虽然 4 位地址最多仅支持 16 个端点，但我们具有一个 IN PID 和一个 OUT PID，它们各自使用了端点地址 1 到 16，因此共有 32 个端点。请注意，它表示端点的地址，而不是端点的编号
- **可选的加载数据** — (0 到 1023 字节)
- **可选的 CRC** — (5 或 16 位)

图 24. USB 数据包内容



9.1 数据包类型

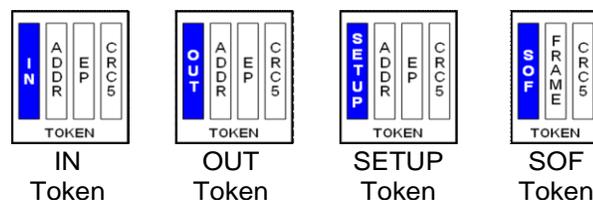
图 24 可代表四种数据包类型。

- 令牌数据包
 - 开始数据操作
 - 指定与传输有关的设备
 - 始终由主机发送
- 数据数据包
 - 传输加载数据
 - 由主机或设备发送
- 握手数据包
 - 确认已接收到无错误的数
 - 由接收方发送
- 特殊数据包
 - 支持多种不同的速度
 - 由主机传输给集线器设备

如上所述，数据包中的任何信息（除了 PID 之外）均是可选的。令牌、数据和握手数据包具有不同的信息组合。令牌数据包、数据数据包和握手数据包部分对各数据包所带有的信息进行了介绍。

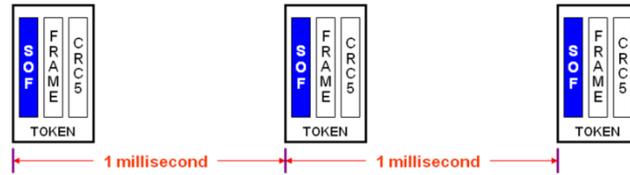
令牌数据包：令牌数据包始终由主机发送，用于定义总线上的数据传输。令牌数据包的类型取决于所执行的传输。IN 类型用于要求设备将数据传输给主机。OUT 类型用于将数据从主机传输给设备。SETUP 类型用于将命令从主机传输给设备。SOF 类型用于确定数据操作帧。IN、OUT 和 SETUP 令牌数据包都有一个 7 位设备地址、4 位端点 ID 和 5 位 CRC。图 25 显示了这四个令牌数据包的框图。

图 25. USB 令牌数据包类型



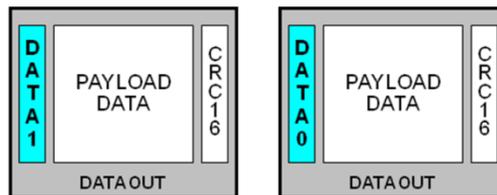
SOF 令牌数据包有助于设备确定帧的开始，并与主机进行同步化。该数据包还有助于防止设备进入挂起模式（经过 3 ms 后，如果设备未收到 SOF，会发生这种情况）。SOF 数据包适用于全速和高速设备，并且每隔 1 ms 发送一次，如图 26 所示。该数据包具有一个 8 位的 SOF PID、11 位的帧计数值（达到最大值时进行反转）和一个 5 位的 CRC。CRC 是该数据包使用的唯一一个错误检测方法。传输 SOF 数据包时，不会使用握手数据包。高速通信使用了更小的时间单位，即微帧。对于高速设备，SOF 每经过 125 u 发送一次，而帧计数值则每经过 1 m 递增“1”。

图 26. 全速设备的 USB SOF 传输



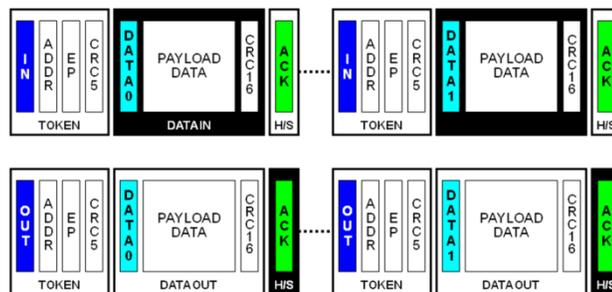
数据数据包：随后 IN、OUT 和 SETUP 令牌数据包所发送的数据包。加载数据的尺寸会因传输类型的不同而异，该尺寸范围为 0 到 1024 字节。在每一个数据数据包成功传输后，数据包 ID 在 DATA0 和 DATA1 之间切换，数据包由一个 16 位 CRC 结束。数据数据包内容如图 27 所示。

图 27. USB 数据数据包



在每一个数据数据包成功传输后，主机和设备将对数据切换进行相应的更新。数据切换的优点在于它可作为附加的错误检测方法。如果接收到的数据包 ID 同预期的不一样，则设备可判断传输中发生了错误，并可能进行适当的处理。使用数据切换的示例是 ACK 在发送后，仍未能收到时。在该示例中，发送方将数据从‘1’更新为‘0’，但接收方则没有进行相应的更新，而仍然保持为‘1’。因此，在下一个数据步骤中，主机和设备将不再同步，这样会引起错误。图 28 显示了一个 USB 传输中的数据切换示例。在该图以及本应用笔记的所有其他图中，白色框表示来自主机的传输，黑色框则表示来自设备的传输。

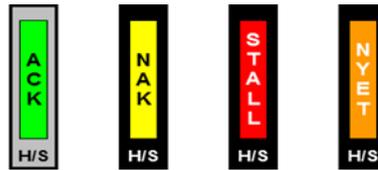
图 28. 数据切换示例



握手数据包：握手数据包指示数据操作的结束。每个握手数据包都带有一个 8 位数据包 ID，并由传输中的接收方发送。每一种 USB 速度都有不同的握手数据包响应选项。所支持的类型由 USB 速度决定：

- **ACK:** 确认数据操作成功完成。(LS/FS/HS)
- **NAK:** 否定确认。(LS/FS/HS)
- **STALL:** 设备发送错误指示。(LS/FS/HS)
- **NYET:** 表示设备当前未能接收其他数据数据包。(仅 HS)

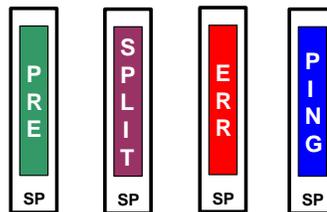
图 29. 握手数据包的指示



特殊数据包: USB 规范定义了四种特殊数据包。

- **PRE:** 主机向集线器发送的数据包，用于指示下一个数据包是低速的。
- **SPLIT:** 发送给令牌数据包之前，用于指示一个分割数据操作。（仅 HS）
- **ERR:** 由集线器返回的数据包，用于报告分割数据操作中发生了错误。（仅 HS）
- **PING:** 接收到 NYET 握手数据包后，检查批量传输 OUT 或控制写入的状态。（仅 HS）

图 30. 特殊数据包的指示



9.2 数据传输类型

USB 数据传输是指主机和设备之间的数据传输方式。一共有三种不同的数据传输类型，它们经常使用不同名称来代表相同的概念。这三种不同的数据传输类型具体如下。

9.2.1 IN/读取/上行数据传输

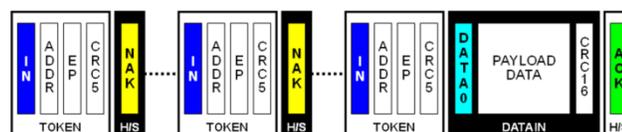
IN、读取和上行是专用术语，表示从设备到主机的数据传输方式。通过发送一个 IN 令牌数据包，主机将启动这些数据传输。目标设备将发送一个或多个数据包，主机则发送一个握手数据包来作出响应。在图 31 中，白框显示的是从主机发送的数据传输，黑框显示的是从设备发送的数据传输。

图 31. IN/读取/上行框图



在图 32 中，设备发送了 NAK 作为响应，从而指出主机发送请求时，它还没准备好发送数据。主机持续发出请求，如果设备已经准备好，它将发送一个数据包来响应主机。然后，主机将发送一个 ACK 握手数据包来确认接收到设备发送的数据。

图 32. IN 数据传输示例



9.2.2 OUT/写入/下行数据传输

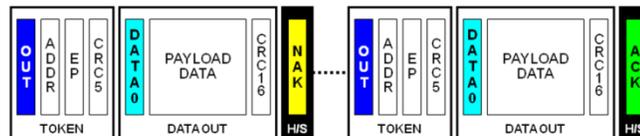
OUT、写入和下行是专用术语，指的是从主机到设备的数据传输方式。在这种数据传输类型中，主机将发送相应的令牌数据包（包括 OUT 或 SETUP），然后发送一个或多个数据包。接收设备将发送相应的握手数据包，以结束数据传输。在图 33 中，白框显示的是从主机发送的数据传输，黑框显示的是从设备发送的数据传输。

图 33. OUT/写入/下行框图



在图 34 中，主机将发送 OUT 令牌数据包和 DATA0 数据包，但会接收到设备所发送的 NAK 信号。然后，主机会重新尝试发送数据。请注意，由于握手数据包被拒绝，因此不会改变数据切换位的状态。如果主机再次尝试发送数据，设备将发送一个 ACK 信号来响应主机，从而指出 OUT 数据传输已经成功。

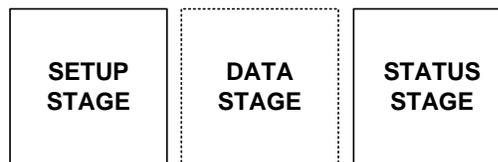
图 34. OUT 数据传输示例



9.2.3 控制数据传输

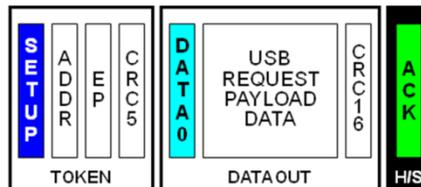
控制数据传输用于识别、配置和控制设备。这种数据传输使主机能够读取设备的信息、设置设备地址、建立配置和发送特定命令。控制数据传输始终针对设备的控制端点。控制数据传输有三个阶段：建立阶段、（可选）数据阶段和状态阶段。图 35 显示的是由主机传送的三个阶段。数据阶段外的虚线表明这是一种可选的数据传输。

图 35. 控制数据传输框图



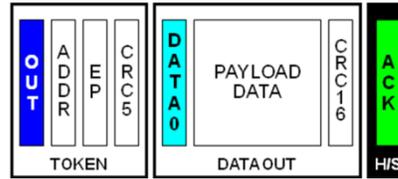
建立阶段（或建立数据包）仅用于一个控制数据传输。需要将大小为 8 字节的数据包（包含 USB 请求）从主机发送到设备。设备必须始终确认建立阶段，不能否认一个建立阶段。

图 36. 建立阶段数据传输



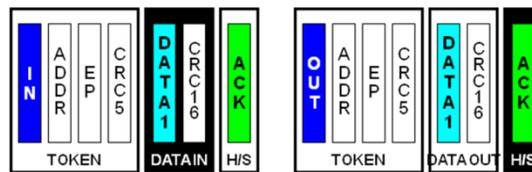
在一个控制数据传输中，可以选择使用数据阶段。该阶段可以进行多次数据传输。只有在主机和设备之间传送数据负载时，才需要使用数据阶段。通常，控制阶段的相应数据可以在建立阶段中传送。

图 37. 建立阶段数据传输



最终阶段 — 状态阶段包括单个 IN 或 OUT 数据传输，这种数据传输会报告先前阶段是否成功。数据包始终为 DATA1（与在 DATA0 和 DATA1 间切换的 IN 和 OUT 正常数据传输不同），并且包含了长度为零的数据包。接收先前数据包的设备会发送一个握手数据传输，以结束状态阶段。

图 38. 状态阶段数据传输



USB 通信一共有三种控制数据传输类型：控制写入、控制读取和控制无数据。图 39、图 40 和图 41 分别显示了这些数据传输的示例。

图 39. 控制无数据传输

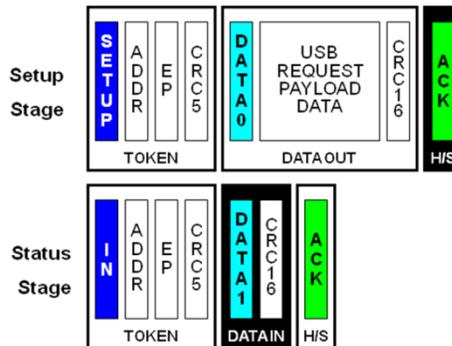


图 40. 控制写入数据传输示例

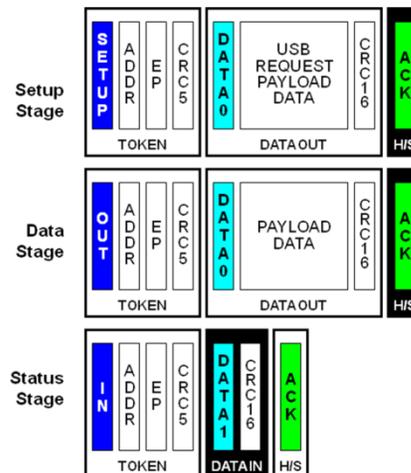
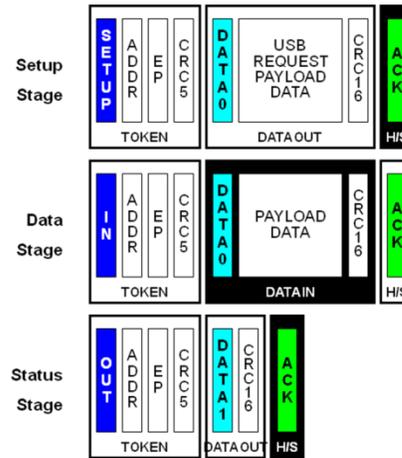


图 41. 控制读取数据传输示例



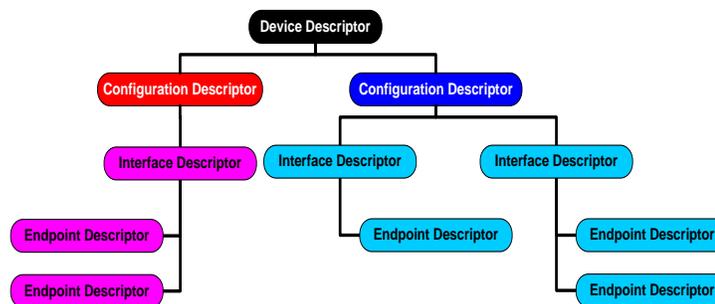
更多有关 USB 传输类型以及在 PSoC 3 和 PSoC 5 设备中执行这种传输的信息，请参考 [AN56377 — PSoC® 3/PSoC 5LP — 实现 USB 数据传输的简介](#)。

10 USB 描述符

如前面所述，当某个设备被连接到 USB 主机上，该设备会向主机提供其功能和电源要求。通常，设备会通过一个描述符表格（其固件的一部分）来提供这些信息。描述符表格是数据的结构化序列，描述了设备信息；这些值由开发人员定义。所有描述符表格都具有一个标准信息，用于介绍设备属性和电源要求。如果某个设计满足指定 USB 设备类别的要求，则该 USB 设备必须具备的其他描述符信息都将包含在设备描述符结构中。附录 A 包含一个 PSoC USB 的全功能设备描述符的示例。

如果您正在阅读或创建您自己的描述符，那么请注意，传输数据字段时，优先传输最低有效位。许多参数的长度均为 2 个字节。请确保先发送低字节，然后再发送高字节。

图 42. 带有两种配置的 USB 描述符



10.1 设备描述符

设备描述符为主机提供了许多信息，如设备需要满足的 USB 规范、设备配置编号、设备支持的协议、供应商标识（又称为 VID，每个公司只能从 USB 实施者论坛获得唯一的 VID）、产品标识（又称为 PID，与数据包 ID 不同）和一个序列号（如果设备有）。设备描述符包含了 USB 设备的重要信息。表 5 显示的是设备描述符的结构。

表 5. 设备描述符表

偏移	字段	大小（字节）	说明
0	bLength	1	该描述符的长度 = 18 个字节
1	bDescriptorType	1	描述符类型 = 设备 (01h)
2	bcdUSB	2	USB 规范版本 (BCD)
4	bDeviceClass	1	设备类别
5	bDeviceSubClass	1	设备子类别
6	bDeviceProtocol	1	设备协议
7	bMaxPacketSize0	1	端点 0 的最大数据包大小
8	idVendor	2	供应商 ID (VID, 由 USB-IF 分配)
10	idProduct	2	产品 ID (PID, 由制造商分配)
12	bcdDevice	2	设备释放编号 (BCD)
14	iManufacturer	1	制造商字符串索引
15	iProduct	1	产品字符串索引
16	iSerialNumber	1	序列号字符串索引
17	bNumConfigurations	1	受支持的配置数量

bLength 是设备描述符的总长度，以字节为单位。

bcdUSB 则显示了设备支持的 USB 版本，通常是最新版本。这是一个二进制代码形式的十进制数据，采用 0xAABC 的形式，其中 A 是主版本号，B 是次版本号，C 是子次版本号。例如，USB 2.0 设备拥有 0x0200 值，USB 1.1 设备拥有 0x0110 值。通常，主机将使用 bcdUSB 以确定需要加载的 USB 驱动器。

bDeviceClass、**bDeviceSubClass** 和 **bDeviceProtocol** 均由操作系统使用，以便在枚举过程中识别 USB 设备的驱动器。将代码填充设备描述符中的这些字段内可以防止各种不同的接口独立运行，如一个复合设备。大部分 USB 设备都在接口描述符中定义了它的类别，并将这些字段保持为 00h。

bMaxPacketSize 会报告由端点 0 支持的数据包的最大字节数量。根据设备，数据包的大小可以为 8 个字节、16 个字节、32 个字节和 64 个字节。

iManufacturer、**iProduct** 和 **iSerialNumber** 都是字符串描述符索引。字符串描述符包括有关制造商、产品和序列号等信息。如果存在字符串描述符，这些变量应该指向其索引位置。如果没有任何字符串，那么应该将零值填充到各个字段内。

bNumConfigurations 定义了设备可支持的配置总数。多个配置使设备能够根据特定条件按照特定条件进行不同的配置，如由总线供电或自供电。更多有关该问题的信息，将在后面详细介绍。

10.2 配置描述符

该描述符会提供特定设备配置的信息，如接口数量、设备由总线供电还是自供电、设备能否启动一个远程唤醒以及设备功耗。表 6 显示的是配置描述符结构。

表 6. 配置描述符表

偏移	字段	大小 (字节)	说明
0	bLength	1	该描述符的长度 = 9 个字节
1	bDescriptorType	1	描述符类型 = 配置 (02h)
2	wTotalLength	2	总长度包括接口和端点描述符在内
4	bNumInterfaces	1	本配置中接口的数量
5	bConfigurationValue	1	SET_CONFIGURATION 请求所使用的配置值，用于选择该配置
6	iConfiguration	1	描述该配置的字符串索引
7	bmAttributes	1	位 7: 预留 (设置为 1) 位 6: 自供电 位 5: 远程唤醒
8	bMaxPower	1	本配置所需的最大功耗 (单位为 2 mA)

wTotalLength 是本配置的整体长度。该值报告了本配置的字节总数以及一个配置所需的接口和端点描述符。

bNumInterfaces 则定义了在该指定配置中接口总数。最小为 1 个接口。

bConfigurationValue 将定义某一直作为参数，SET_CONFIGURATION 请求会使用该参数来选择该配置。

bmAttributes 定义了 USB 设备的参数。如果设备由总线供电，那么位 6 将被设置为 0，如果设备自供电，那么位 6 将被设置为 1。如果 USB 设备支持远程唤醒，则位 5 将被设置为 1。如果不支持远程唤醒，则位 5 将被设置为 0。

bMaxPower 定义了设备全速运行时通过总线消耗的最大功耗，以 2 mA 为单位。如果拔出自供电设备的外部电源，那么它的功耗不会超过该字段中所显示的值。

10.3 接口关联描述符 (IAD)

该描述符介绍两个或多个接口，这些接口与单个设备功能相关。接口关联描述符 (IAD) 会通知给主机各个接口已经连接好。例如，USB UART 具有两个与其相关的接口：控制接口和数据接口。IAD 通知主机这两个接口与同一个功能 (USB UART) 相关，并属于通信设备类别 (CDC)。并非所有情况下都需要使用该描述符。图 43 显示的是单个接口如何与单个设备功能相关。接口描述符定义了该功能的特性。图 43 显示的是两个单独接口与一个指定设备功能相连接的方式。这便是 IAD 需要的信息。表 7 显示的是接口关联描述符的结构。

图 43. 多个接口与多个功能相关

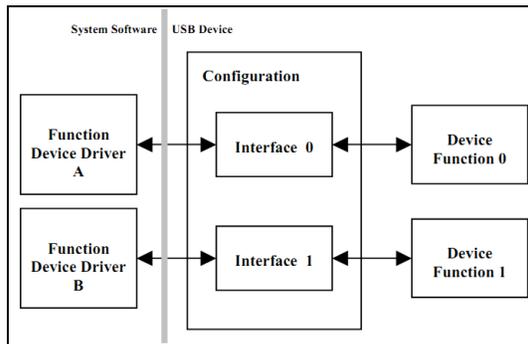


图 44. 多个接口与单个功能相关

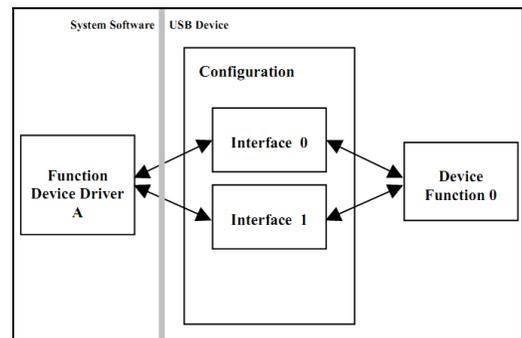


表 7. 接口关联描述符表

偏移	字段	大小 (字节)	说明
0	bLength	1	描述符大小 (以字节为单位)
1	bDescriptorType	1	描述符类型 = 接口关联 (0Bh)
2	bFirstInterface	1	标号与功能相关的第一个接口
3	bInterfaceCount	1	与功能相关的邻近接口的数量
4	bFunctionClass	1	类别代码
5	bFunctionSubClass	1	子类别代码
6	bFunctionProtocol	1	协议代码
7	iFunction	1	功能字符串描述符索引

10.4 接口描述符

一个接口描述符介绍了包含在配置中的特定接口。该接口的端点数量将显示在该描述符中。接口描述符也包含有关设备的 USB 类别的信息。一个 USB 设备可以属于多个预定义类别，表 12 中显示了多种这些类别。一个 USB 设备类别指出了设备功能，并有助于加载用于该特定功能的合适驱动器。表 8 显示的是接口描述符的结构。

表 8. 接口描述符表

偏移	字段	大小 (字节)	说明
0	bLength	1	该描述符的长度 = 9 个字节
1	bDescriptorType	1	描述符类型 = 接口 (04h)
2	bInterfaceNumber	1	该接口基于零的索引
3	bAlternateSetting	1	备用设置值
4	bNumEndpoints	1	该接口所使用的端点数量 (不包含 EP0)
5	bInterfaceClass	1	接口类别
6	bInterfaceSubclass	1	接口子类别
7	bInterfaceProtocol	1	接口协议
8	iInterface	1	该接口字符串描述符索引

10.5 端点描述符

在一个设备中所使用的全部端点都有自己的描述符。该描述符会提供主机必须获取的端点信息。这些信息包括端点的方向、传输类型和数据包的最大尺寸。表 9 显示的是端点描述符的结构。

表 9. 端点描述符

偏移	字段	大小 (字节)	说明
0	bLength	1	该描述符长度 = 7 个字节
1	bDescriptorType	1	描述符类型 = 端点 (05h)
2	bEndpointAddress	1	位 3 ... 0: 端点数量 位 6 ... 4: 预留, 复位为零 位 7: 端点的方向。控制端点可以忽略该位。 0 = OUT 端点 1 = IN 端点
3	bmAttributes	1	位 1 ... 0: 传输类型 00 = 控制 01 = 同步 10 = 批量 11 = 中断 如果该端点不是同步端点, 那么位 5 到 2 将被预留, 必须将这些位设置为零。如果该端点是同步的, 这些位将按如下内容定义: 位 3...2: 同步类型 00 = 无同步 01 = 异步 10 = 自适应 11 = 同步 位 5...4: 用途类型 00 = 数据端点 01 = 反馈端点 10 = 隐式反馈数据端点 数值 11 表示保留
4	wMaxPacketSize	2	该端点的数据包最大尺寸
6	bInterval	1	中断端点的轮询间隔, 单位为 ms (对于同步端点, 该间隔为 1 ms; 控制或批量端点可能忽略该字段)

10.6 字符串描述符

字符串描述符是另一种可选的描述符，它为用户提供了有关设备的可读信息。该描述符中所包含的信息显示了以下内容：设备名称、生产厂家、序列号或不同接口、配置的名称。如果设备没有使用字符串，必须将前面所述的所有描述符中的字符串附加字段的值设置为 00h。

各字符串是由 UNICODE UTF16LE 编码定义的，并且支持使用一个语言 ID 代码实现多种语言。在 Windows 系统中，可以在“device manager”（设备管理工具）中查找这些字符串。表 10 显示的是字符串描述符的结构。

表 10. 字符串描述符表

偏移	字段	大小（字节）	说明
0	bLength	1	该描述符的长度 = 7 个字节
1	bDescriptorType	1	描述符类型 = STRING (03h)
2..n	bString 或 wLangID	变化	Unicode 编码字符串 或 LANGID 代码

字符串描述符包括两种类型。第一种字符串描述符包含表示语言 ID 的值，即为 **wLangID**，它包含一个或多个两字节长的 ID 代码，用于表示字符串的语言。USB-IF 提供了一个包含多种 ID 代码的文档。例如，美式英语的 ID 代码是 0409h。更多有关 ID 代码的信息，请参考 [USB-IF LANGID](#) 页中的内容。出现在 **wLangID** 后面的所有字符串描述符都使用 **bString**，该字符串字段包含一个 Unicode 字符串（UTF16LE）并且每一个字符都使用两个字节来代表。

10.7 其他杂项描述符类型

报告描述符：某个 USB 类设备可能需要一组扩展式的描述符信息。开发者必须保证 USB 设备所需要的任意额外的描述符信息都包含在描述符文件里。例如，对于 HID 类设备，开发者必须将报告描述符添加到描述符文件内，用于定义其他设备属性。如果需要额外的描述符，可以在类定义规范或其他类的支持文档中查找相关的描述符格式。更多有关报告描述符的信息，请参考 [AN57473 — PSoC®3 和 PSoC 5LP 的 USB HID 初级应用笔记](#) 以及 [AN58726 — PSoC®3 和 PSoC 5LP 的 USB HID 中级应用笔记](#)。

MS OS 描述符：Microsoft 有一种描述符，即 Microsoft 操作系统特性描述符（也称为 MS OS 描述符），提供给商特定的设备。该描述符为 Microsoft Windows 提供了各种特定信息，如特定图标、注册设置、助手文件和 URL。MS OS 描述符包含一个字符串和一个或多个特性描述符。执行枚举期间，Windows 需要使用包含了 EEh 标志和内嵌标签的字符串描述符。如果设备支持 MS OS 描述符，在接收字符串描述符后，Windows 需要额外的信息。如果不支持 MS OS 描述符，则设备会发出“STALL”错误指示，作为握手响应。更多有关 MS OS 描述符的信息，请参考 [MSDN 微软操作系统描述符](#) 登录页面。

Device_Qualifier 描述符：USB 类设备中还使用了另一种描述符，称为 Device_Qualifier 描述符。它提供了支持高性能的设备信息，该信息会根据设备的不同运行速度而不同。支持两种速度配置的设备都需要具备该描述符。请求了该描述符的情况下，如果设备正在全速运行，那么它会通知主机：设备以高速运行时操作有什么区别。同样，如果设备正在以高速运行，该描述符将会向主机提供设备在全速运行下的对比信息。通过读取描述符的值，主机能够知道全速配置信息。如果请求了该描述符，而设备仅支持全速操作，那么设备会响应为“STALL”。否则，会按要求将描述符信息提供给主机。更多有关该描述符的信息，请参考 USB 规范中第 9.6.2 章的内容。

BOS 描述符：USB 2.0 版本中能够支持链路电源管理（LPM）的另一种描述符是二进制设备物体存储（BOS）描述符。对于支持 USB 2.0 的 PSoC 系列产品中，只有 PSoC 4200L 设备能够支持 LPM 特性，因此，它也支持 BOS 描述符。LPM 是从 USB 挂起模式中得到优越化的特性。它允许设备以数十纳秒的转换延迟进入和退出低功耗模式（而挂起模式进入/退出需要 3-20 纳秒的延迟）。

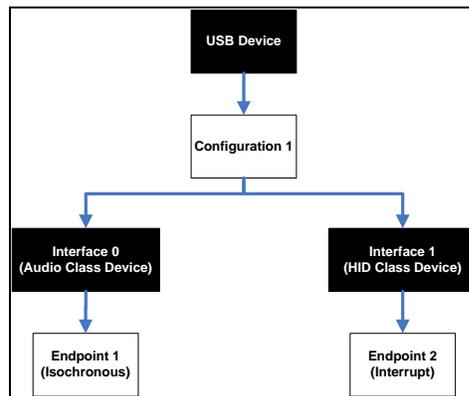
支持 LPM 特性的所有 USB 2.0 设备都需要使用 USB 2.0 的扩展描述符来报告其 LPM 能力。USB 2.0 扩展描述符是 BOS 描述符的一部分。USB 2.0 设备的标准描述符中 bcdUSB 字段的值表示设备支持通过发出 GetDescriptor(BOS) 请求读取 BOS 描述符。支持 BOS 描述符的设备的 bcdUSB 值不得小于 0201H。更多有关 BOS 描述符和 USB 2.0 版本的扩展描述符的信息，请参考 USB 3.1 规范中第 9.6.2 章的内容。

10.8 使用多个 USB 描述符

各个 USB 设备只有一个设备描述符。但是，一个设备可以有多种配置、接口、端点和字符串描述符。设备执行枚举时，终端阶段中有一个步是读取设备描述符，并选择需要使能的设备配置类型。每一次操作只能使能一种配置。例如，某个设计中存在两种配置：一种适用于自供电的设备，另一种适用于由总线供电的设备。这时，用于自供电设备的 USB 的总体性能会与使用于总线供电设备的不一样。拥有多种配置和多种配置描述符可允许设备选择性实现该功能。

同时一个设备可以有多种接口，因此，它也会有多种接口描述符。具有多种接口的 USB 设备（能够执行不同功能）被称为复合设备。USB 头戴式音频耳机便是一个复合设备示例。这种音频耳机包括一个带有两个接口的 USB 设备。其中，一个接口用于音频传输，另一个接口可用于音量调整。可以同时使能多个接口。图 45 显示的是单个 USB 设备中如何分配两种接口。

图 45. 多个接口的设置框图



最终，每个接口可以有多种配置。这些配置类型被称为备用设置。可以使用这些备用设置来更改设备的端点配置，从而保留带宽的不同能力。例如，在某种备用设置中，可以将设备的端点配置为批量传输（没有保证的总线带宽），在另一种备用设置中，可以将设备的端点配置为同步传输（有保证的总线带宽）。图 46 详细演示了该示例。

图 46. 多接口设置框图

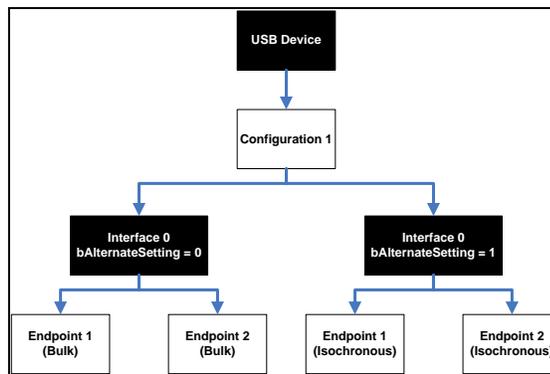
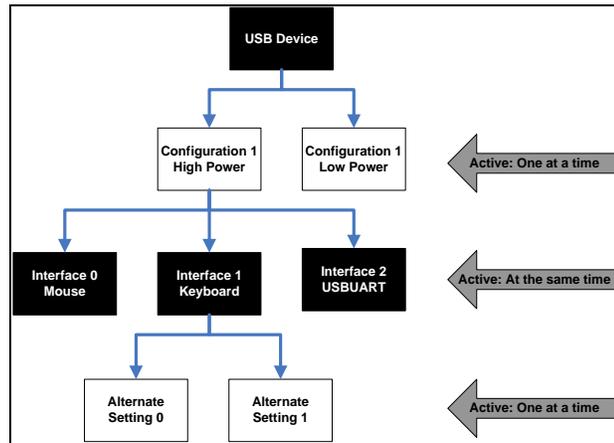


图 47 显示的是各种配置选项的总体框图，有助于根据不同的配置选项来创建准确的可配置 USB 设备。

图 47. 配置框图



11 USB 类设备

US 实施者论坛上有一个已被接受和认可的 [USB 设备类型](#) 列表。最通用的设备类型包括：

- 人机界面设备（HID）
- 大容量设备（MSD）
- 通信设备类（CDC）
- 供应商（供应商特定）

而开发上述的某个类型时都要考虑下面注意事项。第一，每个设备类型都有一个固定的最大带宽。第二，每个设备类型都受限于支持的传输类型以及必须支持的某个指令。但是，使用预定义 USB 类设备的最大优点是：它可以支持多操作系统中的跨平台支持。所有主要的操作系统都自带一个驱动程序，用于几乎所有预定义的 USB 设备类型，从而可以避免创建自定义的驱动程序。表 11 介绍的是一些更为通用的驱动程序（可以与赛普拉斯的产品结合使用）及其特性。

表 11. USB 类设备的驱动程序特性

特性	HID	CDC	WinUSB	LibUSB	CYUSB
Windows 驱动程序支持	有	需要使用“.inf”文件	需要使用“.inf”文件 ^[1]	无	无
支持 64 位	有	有	有	有	有
支持控制传输	有	无	有	有	有
支持中断传输	有	无	有	有	有
支持批量传输	无	有	有	有	有
支持同步传输	无	有	无	有	有
最高速度（全速）	~64 KB/s	~80 KB/s	~1 MB/s	~1 MB/s	~1 MB/s

[1]. Windows XP 操作系统不支持 WinUSB；必须使用 WinUSB 的协助安装程序进行配置。

不符合某个特定 USB 设备的定义的设备被称为供应商特定设备。开发者可对这些设备进行丰富的开发，并使用配置选项来创建各种应用，它们不受限于某个特定的 USB 类型，但仍符合 USB 规范。供应商特定的设备使用 WinUSB、CYUSB、LibUSB 或其他的供应商特定的驱动程序。WinUSB 的优点表现在：它是 Windows 自带的供应商特定驱动程序，并且不需经过 Windows 硬件质量实验室（WHQL）测试来得到驱动程序认证。本应用笔记将在后面的内容中对 WHQL 测试进行更详细的介绍。LibUSB 是一个开源的驱动程序项目，其支持 Windows、Mac 和 Linux 的操作系统。CyUSB 是赛普拉斯自带的供应商特定的驱动程序。CyUSB 在应用中的优点表现在：它具有宽阔的示例应用、支持文档以及由赛普拉斯提供的直接支持。

在 USB 描述符章节中，请注意：设备描述符的第四个字节和接口描述符的第六个字节用于定义 USB 设备类。USB 规范中定义了多种 USB 设备类及其附带的设备类代码。表 12 显示的是可出现在这两个字节中的一些 USB 类代码，从而为您介绍了几种可用的 USB 设备类。

表 12. USB 类代码

类别	使用说明	说明	示例
00h	设备	未指定	没有指定设备类型，可以使用接口描述符来确定需要的驱动程序
01h	接口	音频	扬声器、麦克风、声卡、MIDI
02h	两者	通信和 CDC 控制	调制解调器、以太网适配器、无线适配器
03h	接口	人机界面设备（HID）	键盘、鼠标、游戏杆
05h	接口	物理接口设备（PID）	强制反馈游戏杆
06h	接口	图像	摄像机、扫描仪
07h	接口	打印机	打印机、CNC 机械
08h	接口	大容量存储器	外部硬驱动、闪存驱动、存储卡
09h	设备	USB 集线器	USB 集线器
0Ah	接口	CDC 数据	与类 02h 结合使用
0Bh	接口	智能卡	USB 智能读卡器
0Dh	接口	内容安全	指纹识别器
0Eh	接口	视频	摄像头
0Fh	接口	个人医疗保健	心率监视器、血糖仪
DCh	两者	诊断设备	符合 USB 规范的测试设备
E0h	接口	无线控制器	蓝牙适配器
EFh	两者	其他	ActiveSync 设备
FEh	接口	专门应用	IrDA 桥接器、测试和测量类（USBTMC），USB DFU（直接固件更新）
FFh	两者	供应商特定	表示设备需要使用供应商特定的驱动程序

12 USB 枚举和配置

开发者一般将枚举视为 USB 设备中的单一程序。实际上，枚举是三个阶段的其中一个：动态检测、枚举、配置。动态检测是指识别 USB 端口状态的变化。图 17 显示的是主机/集线器端的下拉电阻。在某个设备被插入时，根据该设备的速度，相应线将被上拉。主机/集线器通过电压变换来检测总线端口状态的变化。枚举阶段紧接着设备检测阶段。枚举指的是给新插入设备分配唯一地址的过程。配置阶段是指通过交换设备请求来确定设备性能的过程。主机用来了解设备的请求是标准请求。所有 USB 设备都要支持这类请求。

下一节介绍了枚举的全部过程。

12.1 动态检测

第一步：设备被连接到 USB 端口上，并得到检测。此时，设备可从总线吸收 100 mA 的电流，并处于被供电状态。

第二步：集线器通过监控端口的电压来检测设备。集线器的 D+线和 D-线上带有下拉电阻，如图 17 所示。如上所述，根据设备的速度，D+或 D-线上会带有上拉电阻。通过监控这些线上的电压变换，集线器检测设备是否得到连接。

12.2 枚举

第三步：主机使用中断端点获得集线器状态（包括端口状态的变化），从而了解新连接的设备。主机从集线器获得设备检测情况后，它会向集线器发送一个请求，以便询问在 GET_PORT_STATUS 请求有效时所发生状态变化的详细信息。

第四步：主机收集该信息后，它通过“USB 速度”一节中所介绍的方法来检测设备的速度。最初，通过确定上拉电阻位于 D+线还是 D-线，集线器可以检测设备速度是全速还是低速。通过另一个 GET_PORT_STATUS 请求，该信息被报告给主机。

第五步：主机向集线器发送 SET_PORT_FEATURE 请求，要求它复位新连接的设备。通过将 D+和 D-线下拉至 GND（0 V），使设备进入复位状态。这些线处于低电平状态的时间长达 2.5 us，因此发生复位条件。集线器在 10 ms 内维持复位状态。

第六步：复位期间发生一系列 J-State 和 K-State，这样是为了确定设备是否支持高速传输。如果设备支持高速，它会发出一个单一的 K-State。高速集线器检测该 K-State 并用 J 和 K 顺序（组成“KJKJKJ”格式）来回应。设备检测到该格式后，它会移除 D+线上的上拉电阻。低速设备和全速设备则会忽略这一步。

第七步：通过发送 GET_PORT_STATUS 请求，主机检查设备是否仍处于复位状态。如果设备仍处于复位状态，则主机会继续发送请求，直到它得知设备退出复位状态为止。设备退出复位状态后，它便进入默认状态，如 USB 电源一节所述。现在，设备可以回应主机的请求，具体是对其默认地址 00h 进行控制传输。所有 USB 设备的起始地址均等于该默认地址。每次只能有一个 USB 设备使用该地址。因此，同时将多个 USB 设备连接到同一个端口时，它们会轮流进行枚举，而不是同时枚举。

第八步：主机开始了解有关设备的更多信息。首先，它要知道默认管道（端点 0）的最大数据包大小。主机先向设备发送 GET_DESCRIPTOR 请求。设备发给主机相应应用笔记 USB 描述符一节所介绍的描述符。在设备描述符中，第八个字节（bMaxPacketSize0）包含了有关 EP0 最大数据包尺寸的信息。Windows 主机要求 64 字节，但仅在收到 8 字节设备描述符后才转换到控制传输的状态阶段，并要求集线器复位设备。USB 规范要求，如果设备的默认地址为 00h，当它得到请求时，设备至少要返回 8 字节设备描述符。要求 64 字节是为了防止设备发生不确定行为。此外，仅在收到 8 字节后才进行复位的操作是早期 USB 设备遗留的特性。在早期 USB 设备中，当发送第二个请求来询问设备描述符时，某些设备没有正确回应。为了解决该问题，在第一个设备描述符请求后需要进行一次复位。被传输的 8 字节包含 bMaxPacketSize0 的足够信息。

第九步：主机通过 SET_ADDRESS 请求为设备分配地址。在使用新分配地址前，设备使用默认地址 00h 完成所请求的状态阶段。在该阶段后进行的所有通信均会使用新地址。如果断开与设备的连接、端口被复位或者 PC 重启，该地址可能被更改。现在，设备处于地址状态。

12.3 配置

第十步: 设备退出复位状态后, 主机将发送 `GET_DESCRIPTOR` 命令, 以便使用新分配地址读取设备的描述符。不过, 此次所有描述符均被读取。主机通过该信息了解设备及其性能。该信息包含外设接口数量、电源连接方法以及所需要的最大电源。主机先请求设备描述符, 而这一次它将收到全部描述符, 而不仅是描述符的一部分。然后, 主机将发送另一个 `GET_DESCRIPTOR` 命令, 以便询问配置描述符。该请求的结果不仅是配置描述符, 并且还包含了与配置描述符相关联的所有描述符, 如接口描述符和端点描述符。**Windows PC** 首先只询问配置描述符 (9 个字节), 然后它会发送第二个 `GET_DESCRIPTOR` 请求, 询问配置描述符以及与配置描述符相关联的所有描述符 (如接口和端点描述符)。

第十一步: 为了让主机 PC (此情况是 **Windows PC**) 成功使用设备, 主机必须加载设备驱动程序。主机会搜索一个用于管理它与设备通信的驱动程序。**Windows** 使用它的 `.inf` 文件寻找与设备产品 ID 和供应商 ID 匹配的驱动程序。也可以选择性寻找与设备发布版本号匹配的程序。如果 **Windows** 未能找到匹配的驱动程序, 它会寻找与设备的类别、子类以及协议相匹配的程序。如果设备先前已经进行了枚举, **Windows** 会使用设备所注册的信息来寻找合适的驱动程序。确定好驱动程序后, 主机可能请求设备的特定描述符或者请求设备重新发送描述符。

第十二步: 收到所有描述符后, 主机使用 `SET_CONFIGURATION` 请求进行特殊的设备配置。大部分设备只有唯一一种配置。对于支持多项配置的设备, 用户或驱动程序可选择合适的配置。

第十三步: 此时设备将处于配置状态。它将按照描述符所定义的性能进行操作。所定义的最大电源是从 V_{BUS} 吸取的。现在就可以在应用中使用设备。

13 调试 USB 设计

假设进行了下述情况: 你花了一段时间 (数小时、多日, 甚至多个星期) 来开发一个使用 **USB** 的设计。现在你需要测试该设计。固件已得到加载, 并且设备已被插入到主机上。观察原型时, 您会发现没有发生任何现象。到底是哪里出错了? 这时您需要进行调试, 从而发现出错的地方。但应该从哪里开始? 在尝试调试某个设计时, 可以使用一些工具和技巧。本节的目的是帮您实现调试过程。

13.1 在主机端进行调试

当 **USB** 设备不能正常工作时, 首先应该在主机端进行调试, 因为主机负责初始化与设备的通信。根据所观察到的结果, 可能需要调试设备。如果主机的操作系统是 **Windows**, 通过下面各个步骤, 可以在 **Device Manager** (设备管理器) 中找到连接设备的列表以及它们的状态 (如图 48 所示)。

如果主机的操作系统是 **Windows XP/Vista**, 通过下面各个步骤, 可以找到连接设备的列表:

第一步: 依次点击 **Start > System > Hardware > Device Manager**。

如果主机使用的操作系统是 **Windows 7**, 通过下面各步骤, 可以找到连接设备的列表:

第一步: 依次点击 **Start > Control Panel**。

第二步: 依次选择 **System > Device Manager**。请注意, 您可能需要将“View by”选项设置为“Large Icons”或“Small Icons”。对于“Category”, 所提供的选项可能不一样。

如果主机的操作系统是 **Windows 8/8.1**, 则通过下面各步骤, 可以找到连接设备的列表:

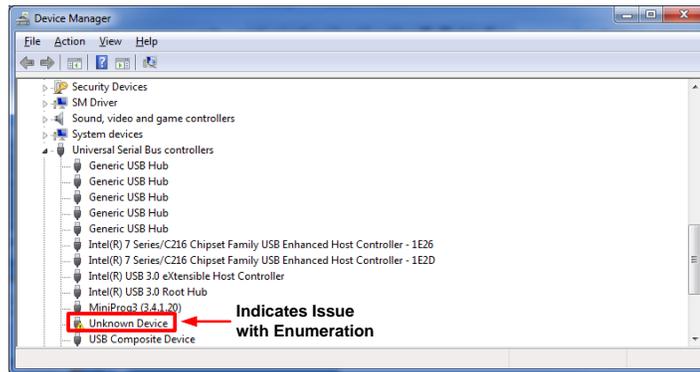
第一步: 点击 Desktop Tile (桌面标题) 以打开 Desktop (桌面)。

第二步: 点击左下角工作栏的“>>”。

第三步: 选择显示的菜单中的 **Control Panel** (控制面板)。

第四步: 选择控制面板的列表中的 **Device Manager** 项。

图 48. Windows 设备管理器



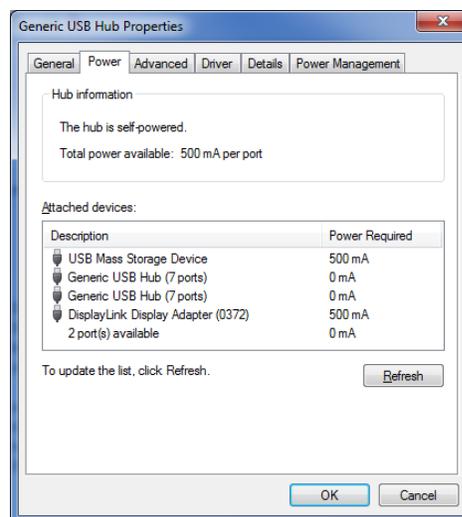
当 USB 设备的主机出现问题时，有两种最常见的问题：硬件发生故障/不正常配置硬件，设备驱动程序发生故障/设备驱动程序丢失。通过 Device Manager，您可以了解所连接设备的一些信息。

Unknown Device（未知设备）：如图 48 所示，该故障可能说明一些问题。第一，它指出了设备没有被正常安装的驱动程序。通常“Unknown Device”不会出现在“Universal Serial Bus controllers”（通用串行总线控制器）项下，而会显示在“Other devices”（其他设备）项下。在这种情况下，请双击设备，以打开设备的属性。将显示 **Error Code 1**（错误代码 1）或 **Error Code 10**（错误代码 10）。设备管理器可以报告多种错误代码。Microsoft 提供了错误代码的列表、说明以及可能的解决方案。可以在[此处](#)导航到该链接。

第二，未安装或者未正常配置的应用程序也会引起该问题。如果在 Windows 尝试安装和配置硬件过程中移除 USB 设备，将发生这种情况。右击设备，然后选择“Uninstall”，再移除设备。重新连接设备。您可能需要通过右击并选择“Update Driver Software”项来强制使驱动程序进行更新。

Device is not listed: 如果您不能在 Device Manager 中找到设备，那么便可确定该设备未满足集线器电源要求。在“Universal Serial Bus controllers”树形结构中，您可以看到一个或多个“USB Root Hub”项或“Generic USB Hub”项。双击该项，然后选择“Power”选项卡。在该选项卡下，您会看到连接设备以及每个设备所要求的电源。请查看图 49 中的一个实例。此外，该选项卡中还列出了可用的总电源。请确保所需电源不能超过可用电源。

图 49. 设备管理器 — USB 电源属性



如果上述操作不能解决问题，请尝试使用其他 USB 端口和/或线缆。一般会存在某些连接或线缆被破坏的可能性。最后解决方案是使用赛普拉斯提供的软件（如 CY SuiteUSB/EZ-USB）。该软件具备的有用工具可帮助您进行调试（不需开发主机端的测试应用）。该软件提供了下面各工具：

Bulkloop (批量回送)： 一个简单的工具，在批量传输中提供数据回送。用户定义要发送的数据格式，然后该工具通知用户要传出 (transferred OUT) 和传入 (transferred IN) 的字节数量。

USB Data Streamer (USB 数据流式)： 用于同步数据传输。通过连续不断地发送数据流并使用该工具，可以测试 USB 设备的带宽和速度性能。用户配置每次所传输的数据包的数量。该工具会记录被丢失的数据包数量、成功接收的数据包数量以及吞吐量 (单位为 KB/s)。

USB Console (USB 控制台)： 该工具使用了 CyUSB.sys 驱动程序与 USB 设备通信。它使用驱动程序提供有关设备的详细信息，并提供用于连接设备的高级控制选项。

USB Control Center： 用于同 USB 设备通信。这些设备使用了赛普拉斯 CyUSB.sys 驱动程序、Windows Mass Storage 驱动程序 (usbstor.sys) 或者 Windows HID 驱动程序 (hidusb.sys)。与 Device Manager 相同，使用该工具也可以查看所连接的 USB 设备。此外，该工具还能通过发送或请求数据提供与不同设备端点通信的机制。

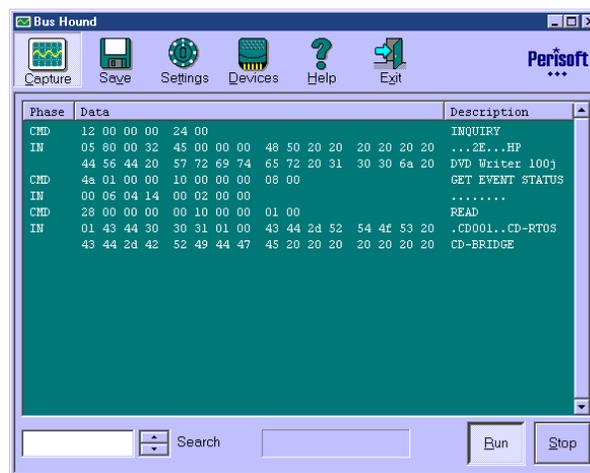
13.2 调试通信

尝试调试某个 USB 设计时，常会遇到需要查看 USB 通信的问题。可利用专业工具实现该操作。这些工具分为两类：硬件基础，是指被连接至待测系统的专用设备；软件基础，是指主机硬件所带有的软件。您通过使用 USB 分析仪可以记录设备和主机之间进行的总线通信。它们会将诸如枚举、信号错误以及常用数据传输的等 USB 通信内容解码成易读信息。这样便提供了一种简单的用于调试 USB 设计问题的方法。

软件分析仪可用于捕获设备数据传输及协议。也可以使用它们向设备发送指令。软件分析仪会在主机上更替 USB 软件堆栈。这是监控 USB 通信的必要步骤。因此，软件分析仪可呈现的信息依赖于主硬件。软件分析仪的成本比硬件计数器部分占优势。

但是，低成本也会带来一些弊端。首先，如果主机不是总线本身的一部分，便不能监控 USB 通信。因此诸如复位或者暂停之类的观察状态将不可用。另外，根据主硬件和软件堆栈，存在其他总线状态，需要主机控制器自己处理，而软件不能干预。它的最后一个弊端与定时相关。软件分析仪定时的准确度依赖于主机操作系统，这样会引起不同等级的错误。

图 50. Perisoft 公司的 Bus Hound 软件分析仪



硬件分析仪的使用已经很普遍，并且有多个厂家在生产。这些设备都是硬件中可定制部分，被插入在主机和设备之间，从而对总线通信进行监控。如果发生某种特殊情况，有些设备甚至具有启动总线通信的功能。根据生产商、所希望的 USB 规格 (USB 2.0 还是 USB 3.0) 以及通信生产功能，这些类型设备的价格可大幅度变化。通常，这些分析仪的花费要比它们软件计数器部分的更大，但是它们确实可以解决软件分析仪所面临的缺点。有多个厂商生产硬件分析仪，其中 TotalPhase 和 LeCroy 是最常见的两家。

图 51. USB 分析仪示例



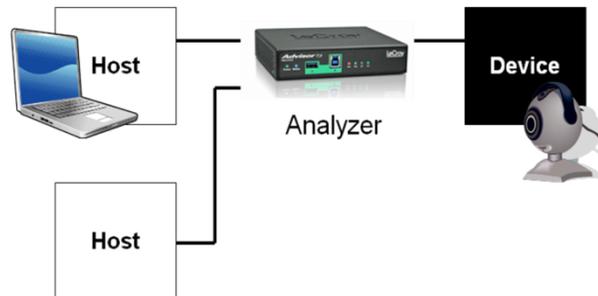
这些设备有两种非常简单的使用方法。第一个方法，USB 分析仪被插入在主机和待测设备之间，如图 52 所示。主机负责运行同测试设备有关的应用，同时也运行用于查看通信情况的分析仪软件。

图 52. 通过在单主机使用分析仪



第二个方法，可使用两台个人电脑，一台作为测试设备的主机；分析仪连接到另一台主机，负责监控总线的活动情况，如图 53 所示。您可能会疑惑，为什么一个分析仪的配置情况能够应用到另一个上。这是因为有时候需要监控测试设备的暂停以及恢复活动。由于主机处于暂停状态，因此您需要一个单独主机来继续监控通信情况。

图 53. 通过双主机使用分析仪



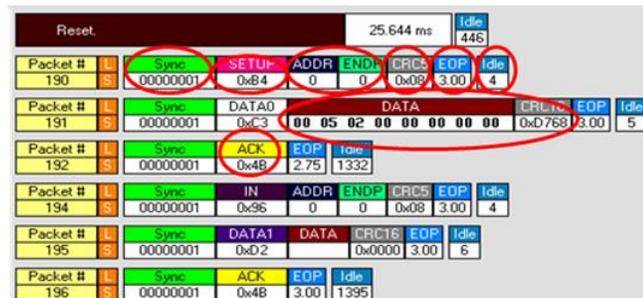
这些分析仪的价格从几百美元到几千美元不等。要注意，如果所设计的应用不要求使用 USB 3.0，那么采购一个仅支持 USB 2.0 的 USB 分析仪可以节省很多花费。另外，采购一个仅支持低速或全速模式的分析仪，也能节省很多开支。

硬件基础 USB 分析仪并不仅仅局限于某个专用硬件，在一些高端示波器上也能看到。这些示波器甚至可以在 USB 合规性测试过程中执行 USB 电气化测试。

现在已经熟悉了几种类型的 USB 分析仪，下面我们将了解分析仪所提供哪类信息。在上面的 USB 枚举和配置一节中，已经介绍了枚举所涉及各个步骤。使用一个总线分析仪，可以观察到枚举过程。图 54 显示的是一个 USB 总线分析仪跟踪。附录 B 显示的是 USB 总线分析仪上 USB 设备的枚举情况。在附录 A 中已经对其描述符进行了介绍。根据您所掌握的资料，阅读 USB 枚举和配置章节以及附录 A 中介绍的内容，这样您能够掌握足够的信息跟上并清楚总线分析仪捕获的过程。

应该清楚，附录 B 中所显示的枚举序列是多种可行枚举事件实例中的一种。这个特殊的枚举序列使用了附录 A 中所介绍描述符，并且枚举是在一个带有 WinUSB 驱动程序的 Windows XP 计算机上完成的。根据特定设备的描述符、执行枚举的操作系统以及所使用的驱动程序等方面的不同，枚举序列会有所不同。

图 54. USB 分析仪输出实例



Packet #	Speed	Sync	Command	ADDR	ENDP	CRC5	EOP	Idle
190	L/S	00000001	SETUP	0xB4	0	0x08	3.00	4
191	L/S	00000001	DATA0	0x3				
DATA: 00 05 02 00 00 00 00 00								
CRC16: 0xD768								
192	L/S	00000001	ACK	0x4B				
EOP: 2.75								
Idle: 1332								
194	L/S	00000001	IN	0x96	0	0x08	3.00	4
195	L/S	00000001	DATA1	0xD2				
DATA: 0x0000								
CRC16: 0x0000								
EOP: 3.00								
Idle: 6								
196	L/S	00000001	ACK	0x4B				
EOP: 3.00								
Idle: 1395								

为帮助理解图 54 所表达的意思，要特别注意数据包#190、数据包#191 和数据包#192 的内容。

■ 数据包#190

- **L/S:** 低速 (Low-Speed)
- **Sync:** 同步主机和设备间数据包的发送。
- **SETUP:** 表示 SETUP 令牌。
- **ADDR:** 由于它是发送给设备的第一个传输，因此当前的地址为 0。所有控制传输都从端点 0、地址 0 启动。
- **ENDP:** 给出了端点地址。ENDP 0 指出这是一个控制传输。
- **CRC5:** CRC 5 位 — 检测令牌中的错误。
- **EOP:** 数据包结束 (End of Packet) — 表示该数据包结束。

■ 数据包 #191

- **DATA0:** 表示它是一个数据令牌
- **CRC 16:** 使用于数据的 16 位 CRC 检查
- **Idle:** 当前数据包和上一个数据包之间的时间

■ 数据包 #192

- **ACK:** 表示成功完成了数据包。

总线分析仪跟踪常被使用在其他赛普拉斯 PSoC USB 应用笔记内。您必须了解总线分析仪跟踪所显示的信息，这样才能充分了解这些内容所传达的意思。这些分析仪可以显示不同抽象等级，包括了从高级的数据操作视图到低级的 D+和 D-波形视图。使用某个总线分析仪时，您可以根据需要决定分析数据的深度。

即在调试某个 USB 时，只要您能明确需要查找的内容，总线分析仪便是非常有用的工具。因为分析仪显示与所发送数据包、握手数据包和定时数据有关的信息。了解了 USB 基础内容时，用户便能够注意到分析仪捕获到的内容，并且知道缺少什么，以及找到解决问题的办法。

13.3 设备端的调试

设备端上的调试能力依赖于设备和开发工具的性能。而对于 PSoC，检查存储器位置和寄存器配置时，可使用 PSoC Designer™和 PSoC Creator™来放置断点。USB 设备的 FX 系列在 Keil uVision 和 Eclipse 环境中也支持一样的功能。这样便提供了一个强大的方法，可确保 ISR 被触发并且指令被准确发送给设备。

赛普拉斯为各种产品提供了技术参考手册，给出了不同寄存器的大体内容以及它们在系统中的作用。通过结合使用这些文档和 IDE 中的调试器，可以保证设备如预期工作。

14 获取 VID 和 PID

为了提供或销售某个 USB 产品，您必须拥有供应商 ID。这便要求您必须从 USBIF 协会获取一个 ID。请确保为您的公司获取一个供应商 ID，这样可以避免法律责任。通过以下两种方法可以获取 VID：

- 成为 USB-IF 协会的会员，要求交付年度会费。在撰写该篇文章的时候，会费为\$4000/年。会员可以获得几项额外好处。最大的利益是可获得由 USBA-IF 赞助的季节性免费合规性研讨会的资格、免费的供应商 ID（如果以前没有被分配）、免费的徽标管理（如果加入 USB-IF 徽标程序）以及参加设备工作组的资格。
- 从 USB-IF 协会购买供应商 ID，其价格为\$5000。但请注意，如果只购买 VID，您仍不能将 USB 徽标用于产品。要想将 USB 徽标使用于某个产品，您需要得到徽标认证（而不需要成为 USB-IF 的会员）。该许可证费用为 \$3,500/两年。另外，请注意该许可证费用不能为您提供 USB-IF 会员的其他利益，如参加合规性研讨会。

在您购买 VID 后，您可以将其用于您所生产的所有 USB 设备。产品 ID 不是购买来的，而是由开发者或由公司选择的。每个产品需要有它自己的 VID/PID 组合。USB-IF 协会建议每个供应商为 PID 设置一个协调的分配方案，以避免不同小组无意为不同的产品选择了同一个 PID。重复的编号会使合规性测试失败。

赛普拉斯通常被问及是否能在客户的最终产品上使用他们的 VID。答案是“否”。可以在开发环境中使用赛普拉斯 VID，但不能在产品设计中。这是因为操作系统（如 Windows）记住了由设备使用的系统文件（inf 文件、驱动程序，等等），并在每次连接设备时加载这些文件。如果我们的客户使用我们的 VID，那么其他客户同样能够将同一个 VID/PID 用于他们现有的产品。在这种情况下，您可以看到，最终产品会绑定到不同的驱动程序以导致故障。这样会使最终客户遇到故障。

15 合规性测试

当涉及到 USB 设备和主机软件的正式合规性测试时，USB 实施者论坛和 Microsoft 提供了两个选项。本节将介绍这两个选项，并简单介绍了该测试的流程以及执行该测试流程后得到的利益。

15.1 USB-IF 合规性测试

USB 的目的是提供一个通用的即插即用连接，这样即使是最不精通技术的人也可以简单地将 USB 线缆插入到主机并使其工作。为了实现该目的，USB IF 执行了合规性测试以确保所有客户对整个 USB 具有良好的体验。USB 产品可以带有一个“认证的 USB”徽标（如图 55 所示），为了向客户提供保证。

图 55. USB 合规性徽标



不能任意将这些徽标放置在某个产品内。但这些徽标必须经过由 USB-IF 管理的一系列合规性测试。可以在设备、集线器、主机系统、芯片构建模块和机械产品（如线缆和连接器）等产品上进行合规性测试。这些合规性测试可确保经过测试的设备能够确认 USB 规范的任何相关部分。通过该测试后，产品可以使用相关的 USB “认证”徽标，另外该产品将被添加到 USB 积分表内（即由 USB-IF 保持的列表），它包含满足强制合规性标准的 USB 产品列表。为了符合 USB 标准，需要完成两个操作，检查表和合规性测试。

15.1.1 USB-IF 合规性检查表

根据得到认证的设备而提供不同的检查表。在本质上，这些检查表是有关设备的规格、功能和性能的调查问卷。许多赛普拉斯设备使用于外设应用，因此该检查表将询问有关机械设计 and 布局、设备状态和信号、工作电压以及功耗等各种相关问题。该检查表中的每个问题均附带了一个章节编号，该编号参照了 USB 规范，以提供更详细的说明内容。这些检查表是获取合规性认证的第一步。

15.1.2 USB-IF 合规性测试

满足所有检查表要求后，需要经过第二个认证标准：合规性测试。通过两种方法可以在一个设备上合规性测试。作为 USB-IF 会员的公司可以参与合规性测试研讨会（也被称为 Plugfest）。也可以将该测试承包给某个独立的认证实验室。下面的链接列出了由 USB-IF 提供的[独立测试实验室](#)。如上面所述，只有您是 USB-IF 的会员，才能参与 Plugfest 研讨会。每种方法都有它自身的优点和缺点。

表 13. 测试实验室与 Plugfest 的比较

独立的测试实验室	USB-IF Plugfest
在测试过程中不需要人为干涉	参与该事件时，需要一个工程师或技术员的 1 到 4 天工作日
每个项目的价格大概为\$2K-6K	免费提供给 USB-IF 会员（不包括时间、路费等等）
咨询时需要提供费用	USB 专家会帮助进行调试，但各种资料由其他参与者共享
测试事件会随时发生	测试事件每年只组织 4 或 5 次
可用于少数原型硬件。	可用于原型主机、集线器和驱动器
高保密性	低保密性

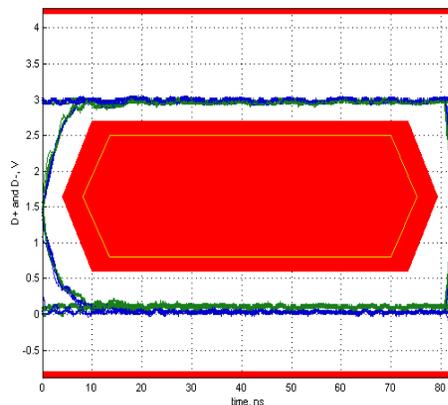
无论选择哪种测试方法，较早完成的检查表将在这个时刻正式得到提交。根据需要进行认证的产品，合规性测试流程也不一样。可以在 USB-IF 网站上找到这些要求。在全速设备中（如 PSoC），必须完成下列测试，以获取认证：电气测试、互操作性测试和功能测试。下面内容介绍了这些测试。

功能测试：也被称为“设备框架测试”，这些测试将评估一系列项目，包括设备压力测试、第 9 章中的合规性测试、特殊 USB 类别测试以及经过挂起状态后正常运行的能力。用于该测试的主要工具被称为 USBCV，[认证准备部分](#)对该工具进行了详细介绍。另外，在设备用作为 HID 的实例中，将执行其他系列测试，以确保符合 USB-IF HID 规范。

电气测试：这些测试通过评估信号质量，以确保设备在总线上没有引起反电压，以及在 USB 规范中所指定的时间内是否响应了挂起/恢复/复位指令，从而实现检查该设备是否符合 USB 电气规范。

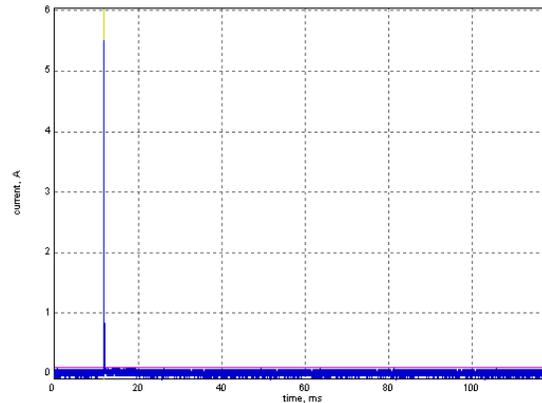
电气测试的常见结果如图 56 中的 USB 眼图所示。该框图是使用高端示波器创建而得到的。该图通过测试上升时间、下降时间、下冲、过冲以及 D+ 和 D- 线的抖动显示了设备的信号质量。

图 56. USB 测试眼图



在电气测试期间，示波器将电涌捕获到设备内，如图 57 所示。该测试的目的是为了确保集线器不消耗过量电流，否则，需要确保存在用于限制电路的合适浪涌电流。

图 57. USB 电涌测量



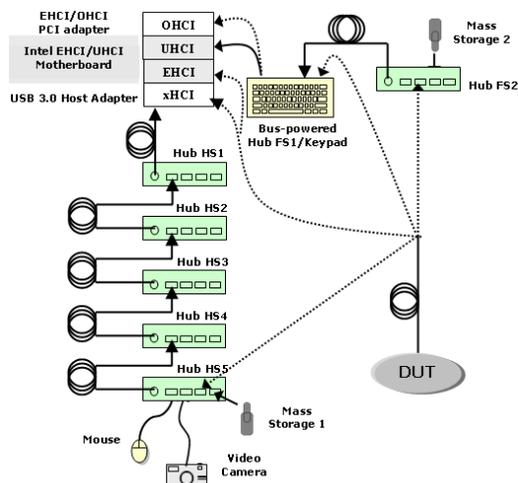
执行电气测试时需要使用一系列适配器电路板和测试硬件。尤其需要使用高端示波器。根据所使用的示波器，USB-IF 在他们的网站上提供了一系列合规性测试流程。

互操作性测试： 这些测试评估设备是否能与主机交互以及是否能与其他 USB 设备共存。该设备能与多个设备、主机和集线器组合交互。根据设备类型（低速、全速、高速或超高速），也提供了用于互操作性测试的各种不同流程。在 USB 社区中通常会听到一个与互操作性相关的术语。该术语即为“黄金树”（即 USB 外设的安排），用于包含以下特性的测试：

- 总线包含同步性、大容量、中断和流量控制等性能。
- 总线包含全速和高速流量。
- 该设备通过五个 USB 集线器连接。
- 设备和主机间的间隔为 30 m。
- 主机包含 EHCI、UHCI 和 OHCI 控制器，用于测试目的。

要想创建小点 2 中所述的高速和全速总线流量，互操作性测试也定义了黄金树所包含的测试设备。互操作性文档甚至还提供了有关制造商和用于测试的硬件模型的特殊信息。从概述的角度来看，附带硬件包含一个视频摄像机（如网络摄像机）、一个批量存储设备（如 USB 硬盘）、一个闪存介质驱动（闪存驱动或跳转驱动）、一个键盘和一个鼠标。所有硬件被连接起来，创建如下图所示的网络。

图 58. USB 黄金树连接点



互操作性测试过程包括多个步骤:

步骤 1: 热插拔并重新连接: 拔掉待测设备并重新将其重新连接到同一个集线器端口上。

步骤 2: 拓扑改变 — 拔掉待测设备并将其连接到另一个端口上。

步骤 3: 热启动 — 通过 Windows Start 菜单 (Start > Shutdown > Restart) 重新启动主机/系统。

步骤 4: 冷启动 — 先通过 Windows Start 菜单 (Start > Shutdown > Shutdown) 关闭主机/系统。然后, 按下 PC 上的电源按钮, 重新启动 PC。

注意: 执行冷启动和热启动主要是因为热启动通常不能执行完整的启动过程。它一般会跳过上电自检 (POST) 过程。此外, 热启动将不会复位所有板上设备。

互操作性的剩余测试将测试系统的各种睡眠状态, 这些状态与睡眠状态有关。具体为 S1 和 S3 状态。

- S0: 工作状态
- S1/S2: CPU 停止
- S3: RAM 挂起
- S4: 磁盘挂起 (称为休眠状态)

在系统睡眠状态网页上, Microsoft 对 PC 在这些状态中所进行的操作提供了详细的介绍。PC 进入睡眠模式时, 将处于这些状态。但是, BIOS 将决定主机 PC 是处于 S1 的还是 S3 的。通常, 可在 BIOS 的电源管理设置中找到 ACPI 睡眠类型, 如图 59 所示。因此对于 S1 和 S3 测试 (稍后将详细介绍), 需要启动测试 PC, 进入 BIOS 配置菜单, 并且适当配置 ACPI 睡眠模式, 如图 59 所示。

图 59. BIOS 中 ACPI 睡眠模式的选项



步骤 5: 使能 S1 挂起 — 在 BIOS 被配置为 S1 模式条件下, 将系统置于睡眠状态。请注意, 开始挂起状态前, 设备应处于工作状态。唤醒系统, 并保证进入挂起状态前所进行的操作能够继续执行, 并且不会发生任何错误。如果待测设备支持远程唤醒, 那么可以通过该方法来恢复系统。

步骤 6: 禁用 S1 挂起 — 在 BIOS 配置为 S1 模式条件下, 当设备处于闲置状态时, 将系统置于睡眠状态。唤醒系统, 并保证待测设备仍然正常运行。如果待测设备支持远程唤醒, 那么可以通过该方法来恢复系统。

步骤 7: 使能 S3 挂起 — 将 BIOS 配置为 S3 模式, 并使系统置于睡眠状态 (同步骤 5 一样)。请注意, 开始挂起状态前, 设备需要处于工作状态。唤醒系统, 并保证待测设备仍然正常运行。进行该测试时, 应该禁用远程唤醒功能。

测试完成且检查表提交后, 在两个周期内, 您会在 USB-IF 集成商列表中看到设备。下图显示了一个所列设备的示例。在此, 可看到经过测试的 PSoC 3。

图 60. PSoC 3 设备显示在集成商列表中

Name	Company	TID	Categories	Added
PSoC 3	Cypress	40770053	Development > Peripheral Silicon >	29 Apr 2011
CY8C3866AXI-040	Semiconductor		Low/Full Speed > Other	00:31:00

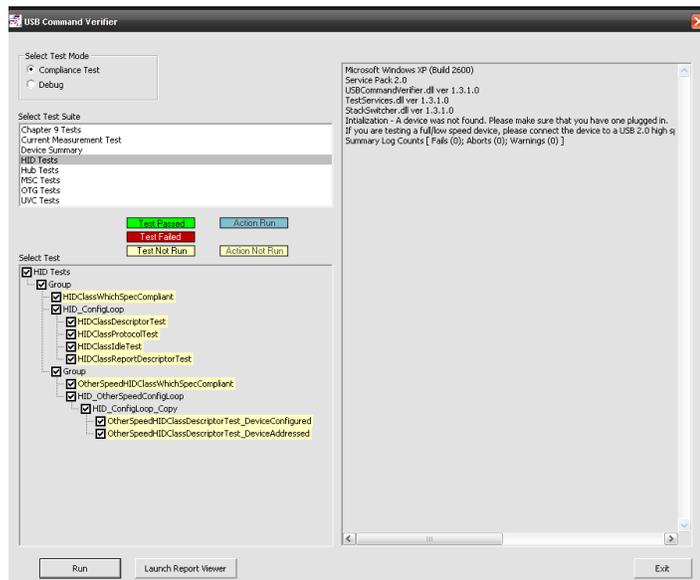
设备超过某个规范参数时，仍可使用免除测试结果的特许来通过合规性测试。获取免除测试结果特许时，在稍微超过所规范条件下，设备仍然可以符合集成商列表的要求。例如，当前的高速测试过程允许设备在第五个集线器层次失败时仍然可以通过高速电气测试。免除测试结果特许是设备制造商与 USB I/F 之间的暂时协议。在当前的 USB 设计得到改善时，免除特许将为无效。有关免除特许的决定是由合规审查委员会做出的。

15.1.3 认证准备

设备得到正式认证前，可以使用由 USB-IF 提供的多个应用程序来测试设备的合规性。这些应用位于其网站上的[工具章节](#)部分。带有 USB 外设的设备中更常用的是 USBCV 和 USBET。这两个工具都可以从 USB-IF 网站上免费下载。

USBCV 是一个命令验证的应用程序，用于测试第 9 章节中 USB 规范（设备框架）的合规性，并且测试类规范（如 HID）的合规性。图 61 显示的是 USBCV 的屏幕截图。

图 61. USBCV 应用程序



USBET 和 USBHSET 是用于测试设备的电气性能的工具。它们包括电涌、暂停模式电流以及 D+/D-信号质量的测试能力。USBHSET 用于测试高速设备的电气特性，并且也是一款用于发送某些设备命令的好工具。当使用 USBHSET 时，不需要高速设备。这一特点使本应用笔记对全速设备的测试非常有用。图 62 和图 63 显示的是这些应用的屏幕截图。另外，还有 USBHTT、OPT 以及 OET 等工具，它们适用于集线器和 USB On-The-Go (USB OTG) 设备。

图 62. USBET 应用程序

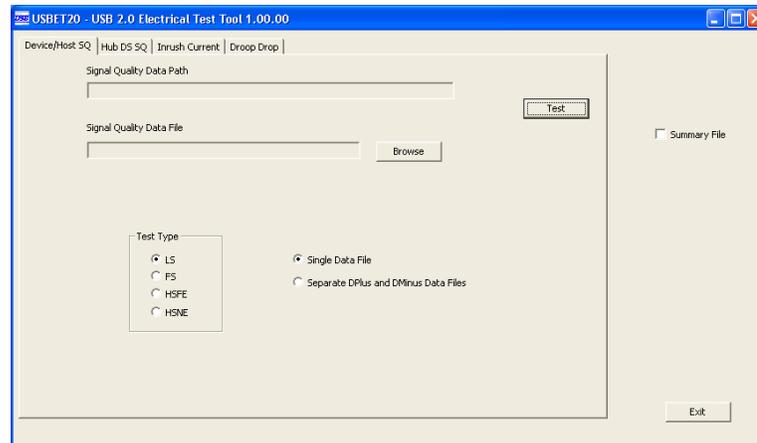


图 63. USBHSET 应用程序



15.1.4 相似的资格

您可能会面对一个问题：为什么许多不同的产品看起来都很相似。这样会发生什么问题？USB-IF 提出了一种特殊的情况，叫做“相似的资格”，在该网站上是这样描述的：

当产品非常相似时，通过测试一个产品，便能够将其他产品添加到集成商名单。许多原型设备制造商（OEM）购买了已名列在集成商名单中并且资格相似的 USB 接口板。

但是，如果在这些产品之间存在‘明显的差异’，仍需要对每个产品进行测试。‘明显的差异’的定义还存在争议，并且最终判断是有合规审查委员会决定。最终判断会报告到 USB-IF 董事会。<http://www.usb.org/developers/compliance/> 网站上会列出根据‘明显差异’做出决定的经验法则。

每个供应商必须保证不同生产模型的终端产品与已测试的样品间没有显著的差异。USB-IF 的审查指出，发货产品与所测试样品间的差异会引起重新测试。USB-I/F 徽标使用权的效果被显示在标准徽标许可协议中。

在下面场合中需要重新测试：

- 微控制器设计更新（新架构和新产品系列）。
- PCB 上的连接器封装更改。

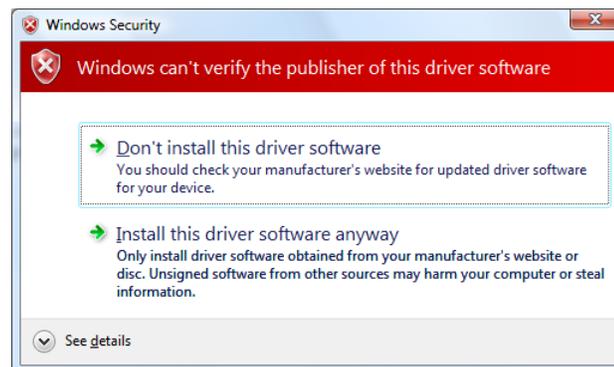
在下面场合中不需要重新测试：

- 产品封装更改（颜色、形状等）。
- 微控制器供应商更改（不更改电路板布局和固件）。只有新微控制器属于集成商名单时，才不需要重新测试。
- 微控制器固件更新（仅更改完全模块化代码，该更改不会影响 USB 功能）。
- 连接器的颜色和美观。

15.2 Microsoft 硬件认证测试

当将 USB 设备连接到 Microsoft Windows 计算机时，您常会收到一个警告信息：该装置没有数字签名的驱动程序。这些警告类似于图 64 所显示的内容。

图 64. Windows 7 驱动程序警告



这些警告是由设备开发者未对其设备进行硬件质量实验室测试而导致的，这种测试还被称为 WHQL 或 Windows 徽标测试。WHQL 测试是 Microsoft 对 USB 硬件或软件进行的测试计划，它有助于验证设备与 Windows 结合使用时会不会导致兼容性问题，从而避免引起 Windows 崩溃或无法正常操作的情况。该程序还包含以下 Windows 平台的认证：Windows 8/8.1、Windows 7、Windows Vista、Windows Server 2012/2012 R2 以及 Windows Server 2008/2008R2。

在设备或驱动程序上执行徽标测试时，该测试会经过针对该特定设备种类（如打印机或触摸屏）设计的一系列特定测试。不符合 Microsoft 所定义的设备组的设备也可以提交“未分类”的驱动程序，从而获取数字信号。该操作将删除如图 64 所示的错误，但是产品不能显示 Windows 徽标。

可以通过下面两种方式中的一种来执行测试：1) 由开发该产品的公司内部执行；2) 由第三方执行。Microsoft 不会替您执行测试。如果在室内测试，则首先要从 Microsoft 网站上下载硬件认证套件（HCK）。该套件提供了所需的全部软件、驱动程序以及工具，用于测试并将结果组合起来发送给 Microsoft。注意：这是 Microsoft 的免费下载套件。完成下载该工具套件后，请使用已安装的软件来执行所需测试，并将提交包传输给 Microsoft。注意：执行测试需要一台具有 Windows Server 副本的专用 PC，负责管理和控制测试。另外，还需要具有 Windows 操作系统的其他 PC，用于测试。您需要测试 Windows 软件的 32 位和 64 位版本。在这次写入期间，Microsoft 会针对每个 Windows 系列收 \$250。HCK 将提供一份指南，这样您能够了解测试配置和需要的各种测试。

如果通过了 WHQL 的所有测试，则通过使用 HCK 将其结果组成一个提交包，发送给 Microsoft。Microsoft 收到和接受提交包后，您将收到 Microsoft 的签字认证文件，其中包含驱动程序，但不会导致这些警告信息。WHQL 测试的通过使设备能承受如图 65 所示的以下徽标的其中的几个，具体情况取决于设备和被测试的 OS。

图 65. Microsoft Windows 认证徽标



最近几年，随着 64 位 Windows Vista 和 64 位 Windows 7 的增多，WHQL 测试需求也在增加。这些操作系统要求将所有驱动程序安装在已签名的 64 位系统中，不然进入特殊启动模式时将被禁止。更多有关 Windows 硬件认证和获取过程的信息，请参见 Microsoft 网站[此处](#)。

16 总结

对于该方面内容，您应该掌握一个基本知识，以更加了解其他 USB 材料。虽然本应用笔记包括了一些理论，但是赛普拉斯的其他一些 USB 应用笔记会指导您如何使用本应用笔记中所描述的信息。请参见[相关资源](#)章节，了解其中的部分资源。本应用笔记中所提及的任何其他信息也被包含在 USB 规范内。我们强烈建议：现在您可以探索和学习您想要了解的任何东西。

17 相关资源

应用笔记

- [AN57473](#) — PSoC 3 和 PSoC 5LP 的 USB HID 初级应用笔记
- [AN58726](#) — PSoC 3 和 PSoC 5LP 的 USB HID 中级应用笔记
- [AN82072](#) — PSoC 3 和 PSoC 5LP USB 使用标准 HID 启动器进行通用数据传输
- [AN56377](#) — PSoC 3 和 PSoC 5LP: 实现 USB 数据传输的介绍
- [AN73503](#) — PSoC 3 和 PSoC 5LP 的 USB HID 引导加载程序
- [AN75705](#) — EZ-USB FX3 入门
- [AN76348](#) — EZ-USB FX2LP 和 EZ-USB FX3 应用的区别
- [USB 高速设备应用笔记](#)

知识库文章

赛普拉斯收集了许多关于使用 USB 功能设备的问题。这些咨询和它们的解决方案被记录到赛普拉斯网站上的知识库文章 (KBA) 内, 并适用于其他客户。下面是连接到 KBA 中一些普通 USB 功能设备的超链接。

- [具有 USB 功能的 PSoC 设备的知识库文章](#)
- [专用 USB 控制器的知识库文章](#)

代码示例

虽然本应用笔记仅提供了关于 USB 2.0 的理论介绍, 但是赛普拉斯还提供了大量的代码示例, 从而可以了解 USB 的不同内容, 有助您开发自己基于 USB 的项目。可以通过以下链接获取这些代码示例。

- [PSoC 3、PSoC 4 以及 PSoC 5LP 代码示例](#) — 该链接包含应用于 PSoC 3、PSoC 4 以及 PSoC 5LP 设备的代码示例。请在该页上输入“USB”关键词来查找有关 USB 的代码示例。
- [USB 全速和低速代码示例](#) — 该链接包含应用于赛普拉斯的专用低速和全速 USB 设备的代码示例。
- [USB 高速代码示例](#) — 该链接包含应用于赛普拉斯的高速 USB 设备的代码示例

技术支持

对于在本应用笔记或其他任何相关应用笔记中没有解决方案的问题或设计过程中需要支持的问题, 请添加一个[技术支持案例](#)。

其他信息

- [官方 USB 2.0 规范](#)
- [由 Jan Axelson 完成的 USB](#)

关于作者

名称:	Robert Murphy
职务:	系统工程师高级职员
背景:	Robert Murphy 毕业于美国普渡大学, 并获得了电气工程技术学士学位。
联系方式:	rlrm@cypress.com

A 附录 A

A.1 示例 PSoC 3 全速 USB 设备描述符

```

/*****
Device Descriptors
*****/
uint8 CYCODE USBFS_1_DEVICE0_DESCR[] = {
/* Descriptor Length          */ 0x12u,
/* DescriptorType: DEVICE     */ 0x01u,
/* bcdUSB (ver 2.0)           */ 0x00u, 0x02u,
/* bDeviceClass                */ 0x00u,
/* bDeviceSubClass            */ 0x00u,
/* bDeviceProtocol            */ 0x00u,
/* bMaxPacketSize0            */ 0x08u,
/* idVendor                   */ 0xB4u, 0x04u,
/* idProduct                  */ 0x34u, 0x12u,
/* bcdDevice                  */ 0x00u, 0x00u,
/* iManufacturer              */ 0x01u,
/* iProduct                   */ 0x02u,
/* iSerialNumber              */ 0x00u,
/* bNumConfigurations         */ 0x01u
};
/*****
Config Descriptor
*****/
uint8 CYCODE USBFS_1_DEVICE0_CONFIGURATION0_DESCR[] = {
/* Config Descriptor Length    */ 0x09u,
/* DescriptorType: CONFIG     */ 0x02u,
/* wTotalLength                */ 0x19u, 0x00u,
/* bNumInterfaces              */ 0x01u,
/* bConfigurationValue        */ 0x01u,
/* iConfiguration              */ 0x00u,
/* bmAttributes                */ 0x80u,
/* bMaxPower                   */ 0x32u,
/*****
Interface Descriptor
*****/
/* Interface Descriptor Length */ 0x09u,
/* DescriptorType: INTERFACE   */ 0x04u,
/* bInterfaceNumber           */ 0x00u,
/* bAlternateSetting          */ 0x00u,
/* bNumEndpoints              */ 0x01u,
/* bInterfaceClass             */ 0xFFu,
/* bInterfaceSubClass         */ 0x00u,
/* bInterfaceProtocol         */ 0x00u,
/* iInterface                  */ 0x00u,
/*****
Endpoint Descriptor
*****/
/* Endpoint Descriptor Length  */ 0x07u,
/* DescriptorType: ENDPOINT     */ 0x05u,
/* bEndpointAddress            */ 0x81u,
/* bmAttributes                 */ 0x02u,
/* wMaxPacketSize              */ 0x40u, 0x00u,
/* bInterval                   */ 0x00u
};

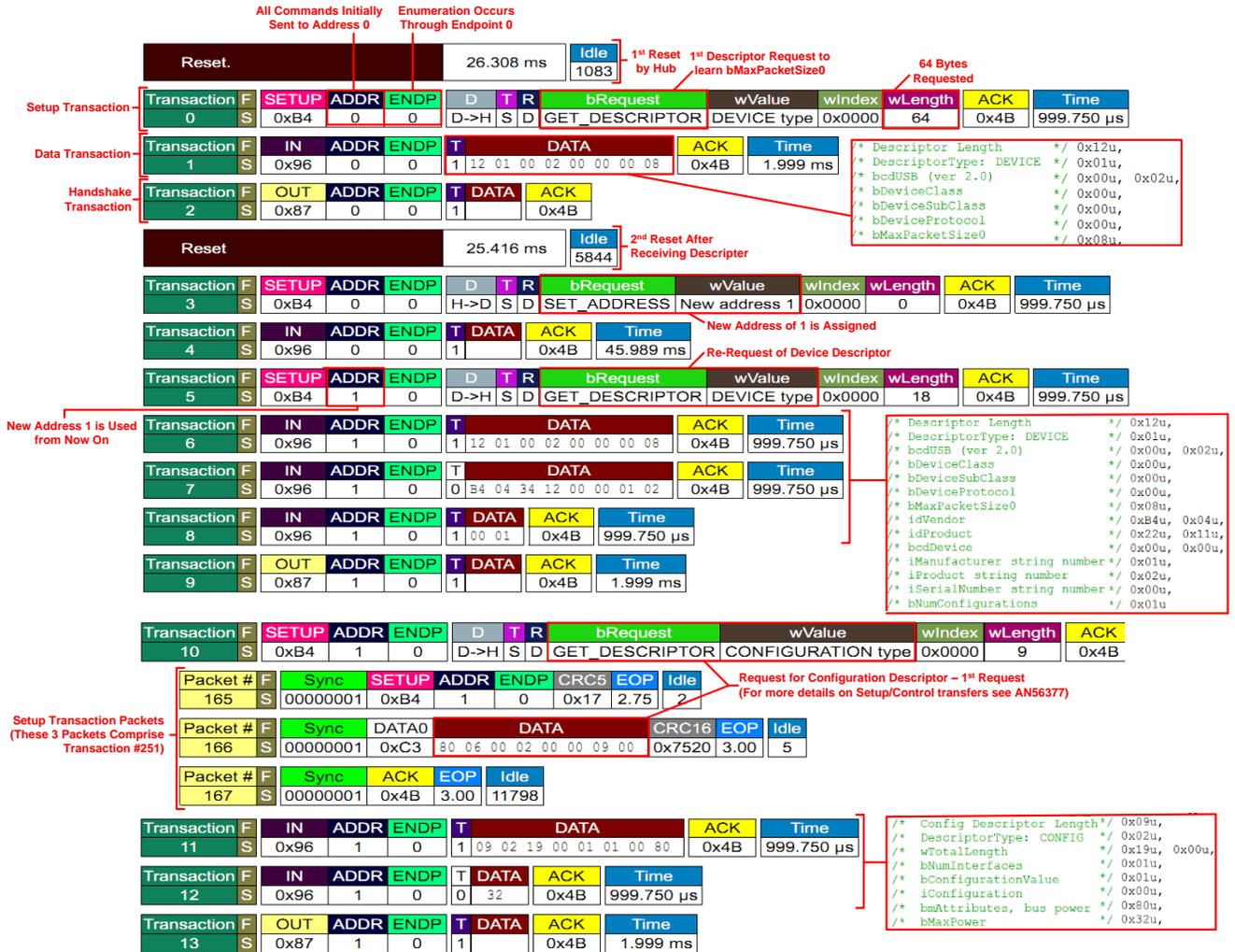
/*****
String Descriptor Table

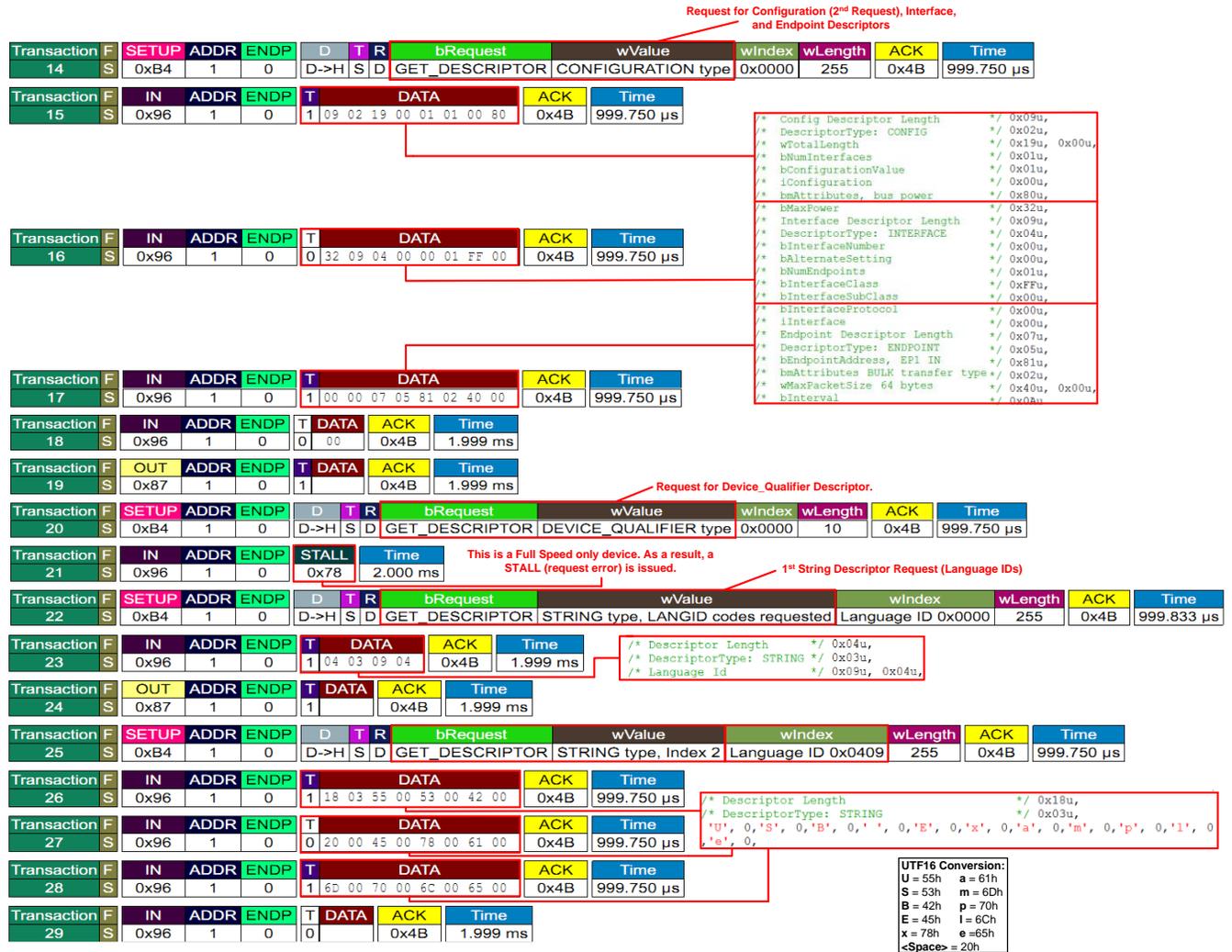
```

```
*****/
uint8 CYCODE USBFS_1_STRING_DESCRIPTOR[] = {
/*****
  Language ID Descriptor
  *****/
/* Descriptor Length          */ 0x04u,
/* DescriptorType: STRING    */ 0x03u,
/* Language Id               */ 0x09u, 0x04u,
/*****
  String Descriptor: "Cypress Semiconductor"
  *****/
/* Descriptor Length          */ 0x2Cu,
/* DescriptorType: STRING    */ 0x03u,
/* Character array: "Cypress Semiconductor" */
'C', 0, 'y', 0, 'p', 0, 'r', 0, 'e', 0, 's', 0, 's', 0, 'i', 0, 's', 0, 'e', 0
, 'm', 0, 'i', 0, 'c', 0, 'o', 0, 'n', 0, 'd', 0, 'u', 0, 'c', 0, 't', 0, 'o', 0
, 'r', 0,
/*****
  String Descriptor: "USB Example"
  *****/
/* Descriptor Length          */ 0x18u,
/* DescriptorType: STRING    */ 0x03u,
/* Character array: "USB Example" */
'U', 0, 'S', 0, 'B', 0, ' ', 0, 'E', 0, 'x', 0, 'a', 0, 'm', 0, 'p', 0, 'l', 0
, 'e', 0,
/*****
/* Marks the end of the list.          */ 0x00u};
/*****/0,
```

B 附录 B

B.1 USB 枚举的总线分析器捕获 (示例)





Re-Request of Descriptors by Driver

Transaction 30	F	OUT	ADDR	ENDP	T	DATA	ACK	Time					
S	0x87	1	0	1		0x4B	1.999 ms						
Transaction 31	F	SETUP	ADDR	ENDP	D	T	R	bRequest	wValue	wIndex	wLength	ACK	Time
S	0xB4	1	0	D->H	S	D	GET_DESCRIPTOR	STRING type, LANGID codes requested	Language ID 0x0000	255	0x4B	999.750 μs	
Transaction 32	F	IN	ADDR	ENDP	T	DATA	ACK	Time					
S	0x96	1	0	1	04 03 09 04	0x4B	1.999 ms						
Transaction 33	F	OUT	ADDR	ENDP	T	DATA	ACK	Time					
S	0x87	1	0	1		0x4B	1.999 ms						
Transaction 34	F	SETUP	ADDR	ENDP	D	T	R	bRequest	wValue	wIndex	wLength	ACK	Time
S	0xB4	1	0	D->H	S	D	GET_DESCRIPTOR	STRING type, Index 2	Language ID 0x0409	255	0x4B	999.750 μs	
Transaction 35	F	IN	ADDR	ENDP	T	DATA	ACK	Time					
S	0x96	1	0	1	18 03 55 00 53 00 42 00	0x4B	999.833 μs						
Transaction 36	F	IN	ADDR	ENDP	T	DATA	ACK	Time					
S	0x96	1	0	0	20 00 45 00 78 00 61 00	0x4B	999.750 μs						
Transaction 37	F	IN	ADDR	ENDP	T	DATA	ACK	Time					
S	0x96	1	0	1	6D 00 70 00 6C 00 65 00	0x4B	999.750 μs						
Transaction 38	F	IN	ADDR	ENDP	T	DATA	ACK	Time					
S	0x96	1	0	0		0x4B	1.999 ms						
Transaction 39	F	OUT	ADDR	ENDP	T	DATA	ACK	Time					
S	0x87	1	0	1		0x4B	29.993 ms						
Transaction 40	F	SETUP	ADDR	ENDP	D	T	R	bRequest	wValue	wIndex	wLength	ACK	Time
S	0xB4	1	0	D->H	S	D	GET_DESCRIPTOR	DEVICE type	0x0000	18	0x4B	999.750 μs	
Transaction 41	F	IN	ADDR	ENDP	T	DATA	ACK	Time					
S	0x96	1	0	1	12 01 00 02 00 00 00 08	0x4B	999.750 μs						
Transaction 42	F	IN	ADDR	ENDP	T	DATA	ACK	Time					
S	0x96	1	0	0	34 04 34 12 00 00 01 02	0x4B	999.750 μs						
Transaction 43	F	IN	ADDR	ENDP	T	DATA	ACK	Time					
S	0x96	1	0	1	00 01	0x4B	999.750 μs						
Transaction 44	F	OUT	ADDR	ENDP	T	DATA	ACK	Time					
S	0x87	1	0	1		0x4B	1.999 ms						
Transaction 45	F	SETUP	ADDR	ENDP	D	T	R	bRequest	wValue	wIndex	wLength	ACK	Time
S	0xB4	1	0	D->H	S	D	GET_DESCRIPTOR	CONFIGURATION type	0x0000	9	0x4B	999.750 μs	
Transaction 46	F	IN	ADDR	ENDP	T	DATA	ACK	Time					
S	0x96	1	0	1	09 02 19 00 01 01 00 80	0x4B	999.750 μs						
Transaction 47	F	IN	ADDR	ENDP	T	DATA	ACK	Time					
S	0x96	1	0	0	32	0x4B	999.833 μs						
Transaction 48	F	OUT	ADDR	ENDP	T	DATA	ACK	Time					
S	0x87	1	0	1		0x4B	1.999 ms						

Re-Request of Descriptors by Driver (Continued)

Transaction 49	F	SETUP	ADDR	ENDP	D	T	R	bRequest	wValue	wIndex	wLength	ACK	Time
	S	0xB4	1	0	D->H	S	D	GET_DESCRIPTOR	CONFIGURATION type	0x0000	25	0x4B	999.833 μs
Transaction 50	F	IN	ADDR	ENDP	T	DATA						ACK	Time
	S	0x96	1	0	1	09 02 19 00 01 01 00 80						0x4B	999.667 μs
Transaction 51	F	IN	ADDR	ENDP	T	DATA						ACK	Time
	S	0x96	1	0	0	32 09 04 00 00 01 FF 00						0x4B	999.750 μs
Transaction 52	F	IN	ADDR	ENDP	T	DATA						ACK	Time
	S	0x96	1	0	1	00 00 07 05 81 02 40 00						0x4B	999.750 μs
Transaction 53	F	IN	ADDR	ENDP	T	DATA						ACK	Time
	S	0x96	1	0	0	00						0x4B	999.750 μs
Transaction 54	F	OUT	ADDR	ENDP	T	DATA						ACK	Time
	S	0x87	1	0	1							0x4B	1.999 ms
Transaction 55	F	SETUP	ADDR	ENDP	D	T	R	bRequest	wValue	wIndex	wLength	ACK	Time
	S	0xB4	1	0	D->H	S	D	GET_STATUS	0x0000	Device Status requested	2	0x4B	999.750 μs
Transaction 56	F	IN	ADDR	ENDP	T	DATA						ACK	Time
	S	0x96	1	0	1	00 00						0x4B	999.750 μs
Transaction 57	F	OUT	ADDR	ENDP	T	DATA						ACK	Time
	S	0x87	1	0	1							0x4B	1.999 ms
Transaction 58	F	SETUP	ADDR	ENDP	H->D	T	R	bRequest	wValue	wIndex	wLength	ACK	Time
	S	0xB4	1	0	H->D	S	D	SET_CONFIGURATION	New configuration 1	0x0000	0	0x4B	999.750 μs
Transaction 59	F	IN	ADDR	ENDP	T	DATA						ACK	Time
	S	0x96	1	0	1							0x4B	

Request for Device Status

Bit 0 = Self Powered, Bit 1 = Remote Wakeup.
Both are set to zero since device is Bus Powered and Remote Wakeup is not supported.
Bits 15 through 2 are reserved and set at zero

Configuration is Selected.
Configuration 1

Device is Ready for Use!

文档修订记录

文档标题: AN57294 — USB 101: 通用串行总线 2.0 简介

文档编号: 002-10080

版本	ECN	变更者	提交日期	变更说明
**	5027480	LISZ	12/09/2015	本文档版本号为 Rev**, 译自英文版 001-57294 Rev*F。
*A	5809617	AESATMP9	07/11/2017	更新徽标和版权
*B	6435146	HERY	01/09/2019	更新版权。译自英文版 001-57294 Rev. *H。

销售、解决方案以及法律信息

全球销售和设计支持

赛普拉斯公司具有一个由办事处、解决方案中心、厂商代表和经销商组成的全球性网络。要想查找离您最近的办事处，请访问[赛普拉斯所在地](#)。

产品

Arm® Cortex® 微控制器	cypress.com/arm
汽车级产品	cypress.com/automotive
时钟与缓冲器	cypress.com/clocks
接口	cypress.com/interface
物联网	cypress.com/iot
存储器	cypress.com/memory
微控制器	cypress.com/mcu
PSoC	cypress.com/psoc
电源管理 IC	cypress.com/pmhc
触摸感应	cypress.com/touch
USB 控制器	cypress.com/usb
无线连接	cypress.com/wireless

PSoC®解决方案

[PSoC 1](#) | [PSoC 3](#) | [PSoC 4](#) | [PSoC 5LP](#) | [PSoC 6 MCU](#)

赛普拉斯开发者社区

[社区](#) | [项目](#) | [视频](#) | [博客](#) | [培训](#) | [组件](#)

技术支持

cypress.com/support



Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© 赛普拉斯半导体公司，2009-2019 年。本文件是赛普拉斯半导体公司及其子公司，包括 Spansion LLC（“赛普拉斯”）的财产。本文件，包括其包含或引用的任何软件或固件（“软件”），根据全球范围内的知识产权法律以及美国与其他国家签署条约由赛普拉斯所有。除非在本款中另有明确规定，赛普拉斯保留在该等法律和条约下的所有权利，且未就其专利、版权、商标或其他知识产权授予任何许可。如果软件并不附随有一份许可协议且贵方未以其他方式与赛普拉斯签署关于使用软件的书面协议，赛普拉斯特此授予贵方属人性的、非独家且不可转让的如下许可（无再许可权）（1）在赛普拉斯特软件著作权项下的下列许可权（一）对以源代码形式提供的软件，仅出于在赛普拉斯硬件产品上使用之目的且仅在贵方集团内部修改和复制软件，和（二）仅限于在有关赛普拉斯硬件产品上使用之目的将软件以二进制代码形式的向外部最终用户提供（无论直接提供或通过经销商和分销商间接提供），和（2）在被软件（由赛普拉斯公司提供，且未经修改）侵犯的赛普拉斯专利的权利主张项下，仅出于在赛普拉斯硬件产品上使用之目的制造、使用、提供和进口软件的许可。禁止对软件的任何其他使用、复制、修改、翻译或汇编。

在适用法律允许的限度内，赛普拉斯未对本文件或任何软件作出任何明示或暗示的担保，包括但不限于关于适销性和特定用途的默示保证。没有任何电子设备是绝对安全的。因此，尽管赛普拉斯在其硬件和软件产品中采取了必要的安全措施，但是赛普拉斯并不承担任何由于使用赛普拉斯产品而引起的安全问题及安全漏洞的责任，例如未经授权的访问或使用赛普拉斯产品。此外，本材料中所介绍的赛普拉斯产品有可能存在设计缺陷或设计错误，从而导致产品的性能与公布的规格不一致。（如果发现此类问题，赛普拉斯会提供勘误表）赛普拉斯保留更改本文件的权利，届时将不另行通知。在适用法律允许的限度内，赛普拉斯不对因应用或使用本文件所述任何产品或电路引起的任何后果负责。本文件，包括任何样本设计信息或程序代码信息，仅为供参考之目的提供。文件使用人应负责正确设计、计划和测试信息应用和由此生产的任何产品的功能和安全性。赛普拉斯产品不应被设计为、设定为或授权用作武器操作、武器系统、核设施、生命支持设备或系统、其他医疗设备或系统（包括急救设备和手术植入物）、污染控制或有害物质管理系统中的关键部件，或产品植入之设备或系统故障可能导致人身伤害、死亡或财产损失其他用途（“非预期用途”）。关键部件指，若该部件发生故障，经合理预期会导致设备或系统故障或会影响设备或系统安全性和有效性的部件。针对由赛普拉斯产品非预期用途产生或相关的任何主张、费用、损失和其他责任，赛普拉斯不承担全部或部分责任且贵方不应追究赛普拉斯之责任。贵方应赔偿赛普拉斯因赛普拉斯产品任何非预期用途产生或相关的所有索赔、费用、损失和其他责任，包括因人身伤害或死亡引起的主张，并使之免受损失。

赛普拉斯、赛普拉斯徽标、Spansion、Spansion 徽标，及上述项目的组合，WICED，及 PSoC、CapSense、EZ-USB、F-RAM 和 Traveo 应视为赛普拉斯在美国和其他国家的商标或注册商标。请访问 cypress.com 获取赛普拉斯商标的完整列表。其他名称和品牌可能由其各自所有者主张为该方财产。