第 15 章　USB 原理及应用

在 STC32G 系列 32 位单片机集成了 USB 2.0 模块，进一步扩展了 32 位 8051 单片机的应用范围。本节主要介绍了 USB 原理以及基于 USB 的 HID 和 CDC 应用实现。

本章主要内容包括 USB 协议概述、USB 2.0 程序设计实现、人机交互设备原理、人机交互设备程序设计、通信设备类原理、通信设备类程序设计，以及 USB 寄存器。

15.1　USB 协议概述

本节对 USB 2.0 协议进行了简要概述，以帮助读者从整体上对 USB 2.0 协议有一个初步了解。本节内容主要包括 USB 通信基础、USB 通信的时域构成、USB 通信模型、USB 标准请求、USB 通信过程，以及 USB 描述符。

15.1.1　USB 通信基础

USB 通信基础主要涉及到 USB 系统组成、USB 架构、USB 物理接口、USB 编码方式、USB 总线状态、USB 速度和 USB 电源。

1. USB 系统组成

USB 系统分为 USB 主机和 USB 设备。

USB 主机：提供 USB 接口和接口管理功能的硬件、软件、固件的复合体。PC 机或 OTG 设备，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。例如，主机从设备接收数据为 IN 类型，主机发送数据给设备为 OUT 类型。USB 主机的硬件主要包括 USB 主控制器和 USB 根集线器。

USB 设备：1. 集线器 HUB：扩展主机接口，设备可以通过其接入主机 2. 功能设备，如 U 盘，USB 摄像头，HID 键盘鼠标等。

物理连接：即 USB 电缆，USB 使用差分信号传输数据，USB 全速/高速模式电缆必须外层屏蔽铜质传输线，且差分数据线双绞。一条 USB 的传输线分别由地线、电源线、D+、D-四条线构成，其中 D+、D-是差分输入线，使用的电压为 3.3V，而电源线与地线可向设备提供 5V 电压，最大电流 500mA。

2. USB 架构

(1) USB 主控制器

USB 主控制器是 USB 主机上的控件，USB 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。具体为以下几种主控制器。

OHCI(open host controller interface)是支持 USB1.1 的标准，但它不仅仅是针对 USB，还支

持其他的一些接口，如 Apple 的火线（firewire，IEEE 1394）接口。与 UHCI 相比，OHCI 的硬件复杂，硬件做的事情更多，所以实现对应软件驱动的任务，相对较简单。主要用于非 X86 的 USB，如扩展卡、嵌入式开发板的 USB 主控。

UHCI（universal host controller interface），是 Intel 主导的对 USB1.0、1.1 的接口标准，与 OHCI 不兼容。UHCI 的软件驱动的任务重，需要做得比较复杂，但可以使用较便宜、较简单的硬件 USB 控制器。Intel 和 VIA 使用 UHCI，而其余的硬件提供商使用 OHCI。

EHCI（enhanced host controller interface），是 Intel 主导的 USB2.0 的接口标准。EHCI 仅提供 USB2.0 的高速功能，而依靠 UHCI 或 OHCI 来提供对全速（full-speed）或低速（low-speed）设备的支持。

xHCI（eXtensible host controller interface），USB3.0 的接口标准，它在速度、节能、虚拟化等方面都比前面 3 种有了较大的提高。xHCI 支持所有种类速度的 USB 设备（USB 3.0 SuperSpeed，USB 2.0 Low-，Full-，and High-speed，USB 1.1 Low-and Full-speed）。xHCI 的目的是为了替换前面 3 种 USB 主控制器（UHCI/OHCI/EHCI）。

（2）USB 拓扑结构

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。

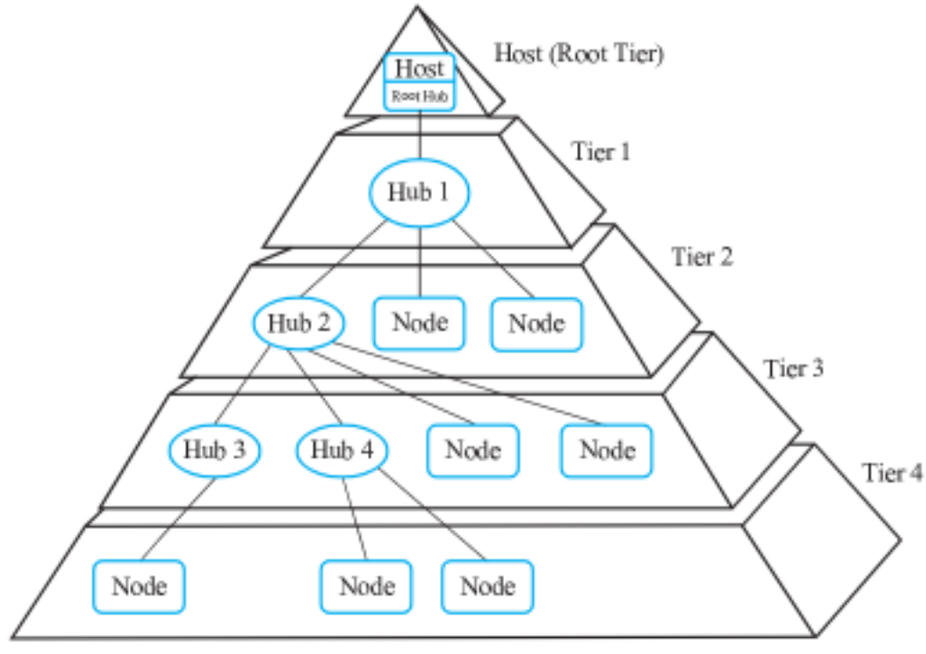

图 15.1　USB 拓扑结构

每个 USB 系统只允许有一个 HOST（主机）。允许的最大层数为 7 层（包含主机）；每层的电缆最大长度为 5 米，电缆总长度为 30 米；每层最大允许接 5 个 DEVICE（设备）；

（3）USB 协议分层

端点（endpoint）：USB 通信的基本单元，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。

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。

接口层（interface）：描述 USB 设备的具体功能，例如一个 USB 设备既有键盘的功能又有存储功能，该设备就有两个接口。

图 15.2　USB 协议分层

3. USB 物理接口

（1）USB 线缆

一个 USB 线缆包含山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 Vbus 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 15.3 所示。有一个 D+线（绿色）和一个 D-线（白色）。VBUS 线为所有相连设备提供了恒定的 4.40 V、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）

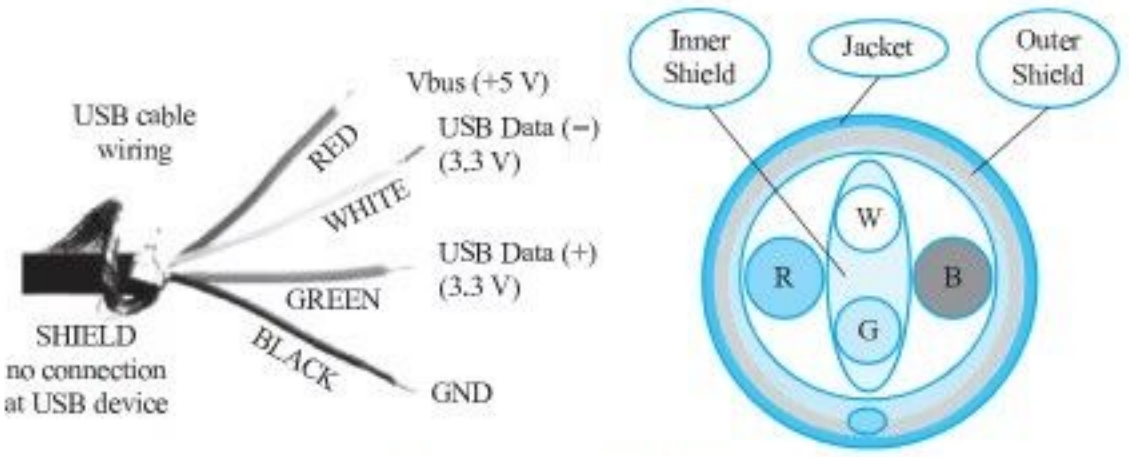

图 15.3　USB 线缆内部

（2）USB 连接器

由于本次程序设计仅涉及 TYPE-A 与 Micro-B 连接器，因此本文仅介绍这两种连接器，其余连接器的详细资料读者可查看 USB 中文网或 USB 官网进行了解。

表 15.1　USB 连接器

类型	引脚图	连接器图
TYPE-A	1 2 3 4 Type-A	
Micro-B	54321 Micro-B	

引脚定义及颜色如表 15.2、15.3 所示。

表 15.2　TYPE-A 连接器引脚定义

引脚	名称	线缆颜色	描述
1	VBUS	红色或者/橙色	+5V 供电
2	D-	白色或者/金色	差分数据-
3	D+	绿色/绿色	差分数据+
4	GND	黑色/蓝色	地

表 15.3　Micro-B 连接器引脚定义

引脚	名称	线缆颜色	描述
1	VBUS	红色	+5V 供电
2	D-	白色	差分数据-
3	D+	绿色	差分数据+
4	ID	N/A	区分另一端接口类型 A 接口(主机):接地 B 接口(设备):不连接
5	GND	黑色	地

4. USB 编码方式

USB 采用不归零反转差分(non-return to zero indicates,NRZI)编码方式,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,为保证定时信息的准确需要在每 6 个 **1** 插入一个逻辑 **0** 保证同步。图 15.4 为 NRZI 码与待发送数据波形:

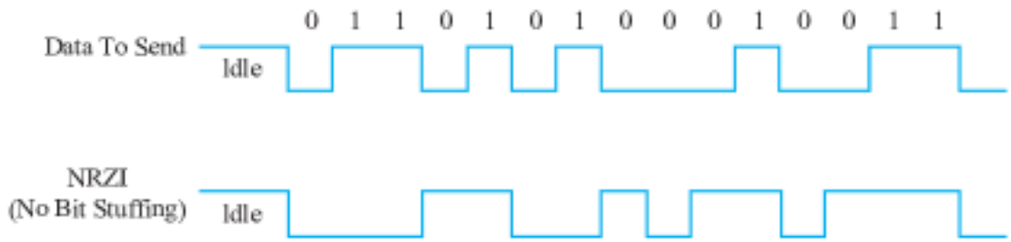

图 15.4　NRZI 码与待发送数据波形

通过在 6 个连续的逻辑 **1** 后面插入一个逻辑 **0** 可以实现位填充。USB 硬件上的接收器会自动检测额外位,并忽略它。使用差分 D+和 D-信号是为了抑制共模噪声。

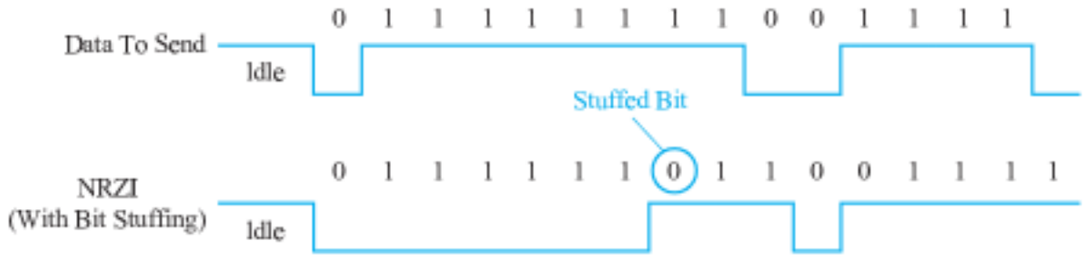

图 15.5　6 个连续的 1 的 NRZI 编码的逻辑电平

5. USB 总线状态

D+线和 D-线上的高低电平的组合表示了 USB 总线的不同状态,其状态可描述为表 15.4

表 15.4　USB 总线状态与指示

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(SEO)	D+和 D-为低电平
单端 1(SEI)	D+和 D-为高电平
J 状态: 低速 全速 高速	 差分 0 差分 1 差分 1
K 状态: 低速 全速 高速	 差分 1 差分 0 差分 0
恢复状态:	K 状态
包起始(SOF)	USB 数据总线从 idle 状态切换到 K 状态。
包末尾(EOP)	SEO 持续两个基本时间单位,以及 J 状态持续一个时间单位。

表 15.4 不同总线状态的详细内容见 USB 协议官方文档。读者若对 USB 线缆上的信号感兴趣可以使用示波器自行测量。

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)。

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: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:如手机、音频设备和压缩视频口总线速率:12 Mb/s。最大的有效数据速率:1.2 MB/s

高速设备:如视频、影像和存储设备口总线速率:480 Mb/s。最大的有效数据速率:53 MB/s

本文仅介绍 USB 的全速模式以及 USB 低速模式的连接图。USB 全速模式是 USB 设备在 D+线上有一个 1.5K 的上拉电阻。USB 低速模式是 USB 设备在 D-线上有一个 1.5K 的上拉电阻。

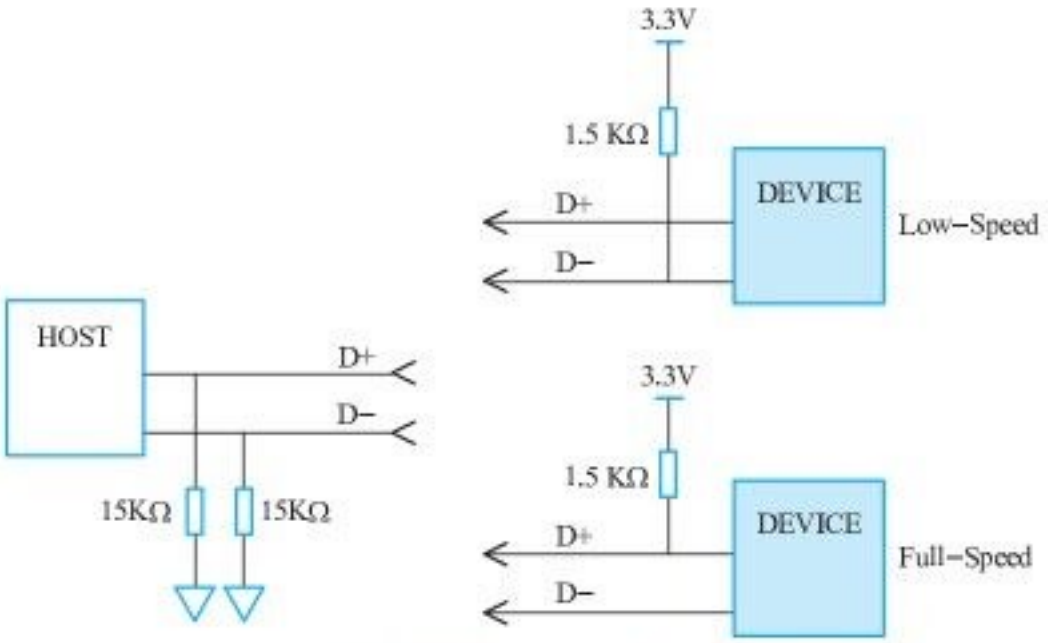

图 15.6　USB 的全速模式以及 USB 低速模式的连接图

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。但是单片机已经带有所需的内部上拉电阻,因此不需要外部上拉电阻。

7. USB 电源

USB 供电方式可以是:总线供电、自供电或者二者相结合的方式。依据程序设计此处仅给出总线供电方式的说明。

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 100 mA 的电流,高功耗设备最多 500 mA 的电流。电流超过 500 mA 的设备要自供电。

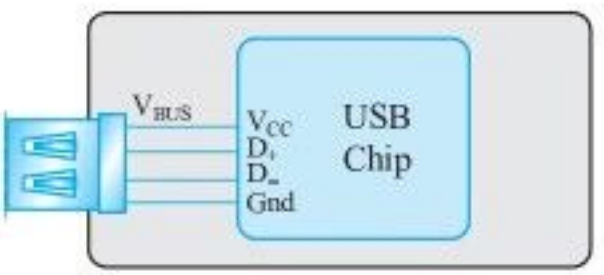

图 15.7　USB 总线供电设备

15.1.2　USB 通信的时域构成

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,也可能无握手数据包。

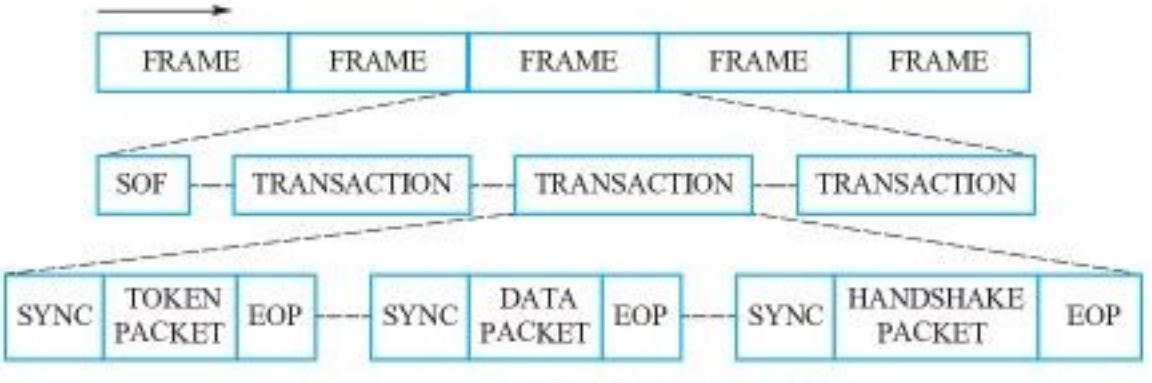

图 15.8　从时间角度观察 USB 通信

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)和一个握手数据包。

这些数据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 15.9 显示了一次数据操作的框图。

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。数据包的结构如图 15.10 所示。图 15.10 可作为数据包的模板,具体的数据包构成需要依据具体的数据包类型。

- PID:数据包 ID,共 8 位,其分为 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。
- ADDR:可选的设备地址,共 7 位,最多可支持 127 个设备。
- EP:可选的端点地址,共 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。注意,它表示端点的地址,而不是端点的编号。
- PAYLOAD DATA:可选的加载数据,0~1 023 B。
- CRC:可选,5 或 16 位。

图 15.9 数据操作框图

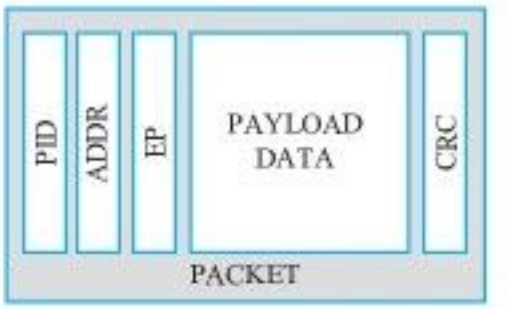

图 15.10 USB 数据包内容

1. 数据包类型

数据包类型共有四种,数据包类型由 PID 决定。

- 令牌(token)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(data)数据包:传输加载数据、由主机或设备发送。
- 握手(handshake)数据包:确认已接收到无错误的数据、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,数据包中的任意信息(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。

(1) 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类型。主机向设备发送 IN 令牌数据包,用于从设备读取数据。主机向设备发送 OUT 令牌数据包,用于将数据从主机传输给设备。主机向设备发送 SETUP 令牌数据包,用于将主机的请求传输给设备(有关该数据包的具体通信流程详见 USB 标准请求以及 USB 通信过程这节)。SOF 令牌数据包用于确定帧的起始位置。IN、OUT 和 SETUP 令牌数据包都有一个 7 位设备地址(ADDR)、4 位端点 ID(EP)和 5 位 CRC。图 15.11 显示了这四个令牌数据包的结构图。

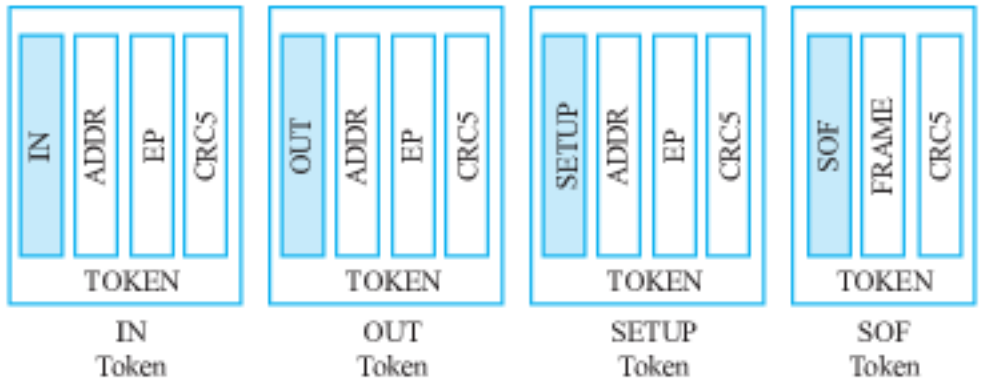

图 15.11　USB 令牌数据包类型

SOF 令牌数据包帮助设备确定帧的起始位置,从而与主机进行同步。该数据包还有助于防止设备进入挂起模式(经过 3ms 后,如果设备未收到 SOF 设备将会进入挂起模式)。SOF 数据包适用于全速和高速设备,并且每隔 1ms 发送一次,如图 15.12 所示。该数据包具有一个 8 位的 SOF PID、11 位的帧计数值(FRAME),达到最大值时进行反转,还有一个 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 125us 发送一次,而帧计数值则每经过 1ms 递增“1”。

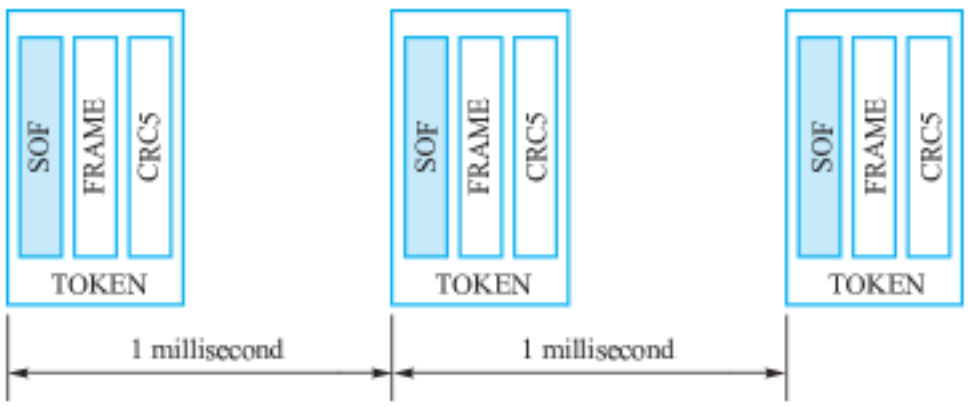

图 15.12　全速设备的 USB SOF 传输

(2) 数据数据包:加载数据的大小会因传输类型的不同而异,数据大小的范围为 0~1024 B。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。数据数据包结构如图 15.13 所示。

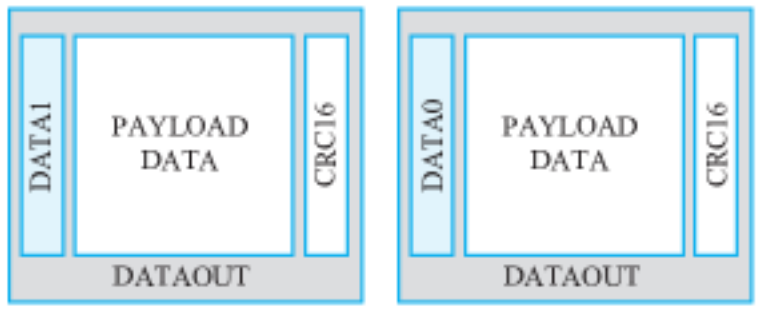

图 15.13　USB 数据数据包

在每一个数据数据包成功传输后,主机和设备将对数据包 ID 进行切换。例如,上一个数据包 ID 为 DATA0,下一个数据包 ID 则为 DATA1。数据包 ID 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,并能进行适当的处理。数据包 ID 切换的示例:ACK 在发送后,若未能正常接收,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**,在下一个数据步骤中,

主机和设备将不再同步。图 15.14 显示了一个 USB 传输中的数据切换示例。在本文所有的图中,白色框表示来自主机的数据包,黑色框则表示来自设备的数据包。

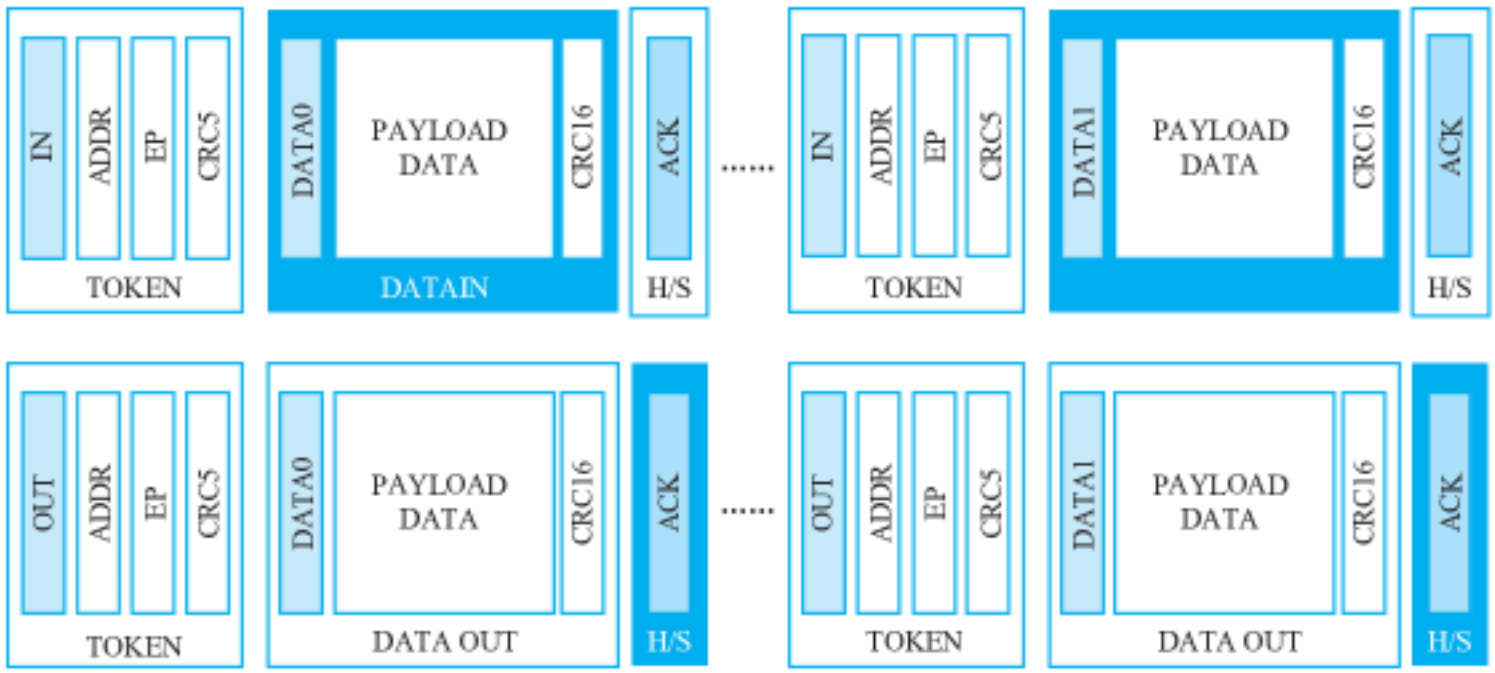

图 15.14　数据切换示例

(3) 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,如图 15.15 所示。不同 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。(低速/全速/高速)
- NAK:否定确认。(低速/全速/高速)
- STALL:设备发送错误指示。(低速/全速/高速)
- NYET:表示设备当前未能接收其他数据数据包(仅高速)

(4) 特殊数据包:USB 规范定义了四种特殊数据包,如图 15.16 所示。

- 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:发送在令牌数据包之前,用于指示一个分割数据操作。(仅高速)
- 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。(仅高速)
- 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。(仅高速)

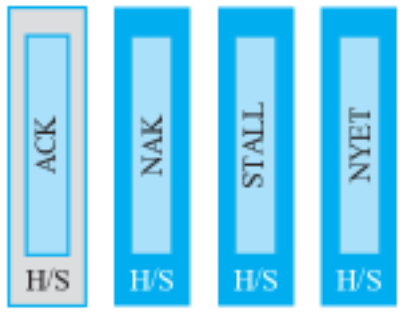

图 15.15　握手数据包的指示

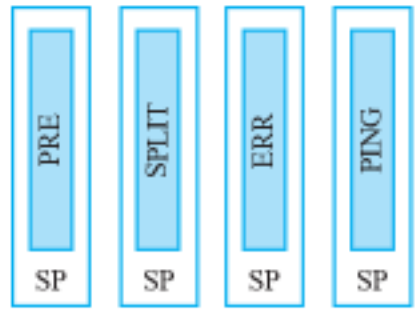

图 15.16　特殊数据包的指示

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。

（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。主机通过向设备发送一个 IN 令牌数据包，将启动此类数据传输。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 15.17 中，白框显示的是从主机发送的数据包，黑框显示的是从设备发送的数据包。

图 15.17　IN/读取/上行框图

在图 15.18 中，设备发送了 NAK 作为响应，说明主机发送请求时，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。

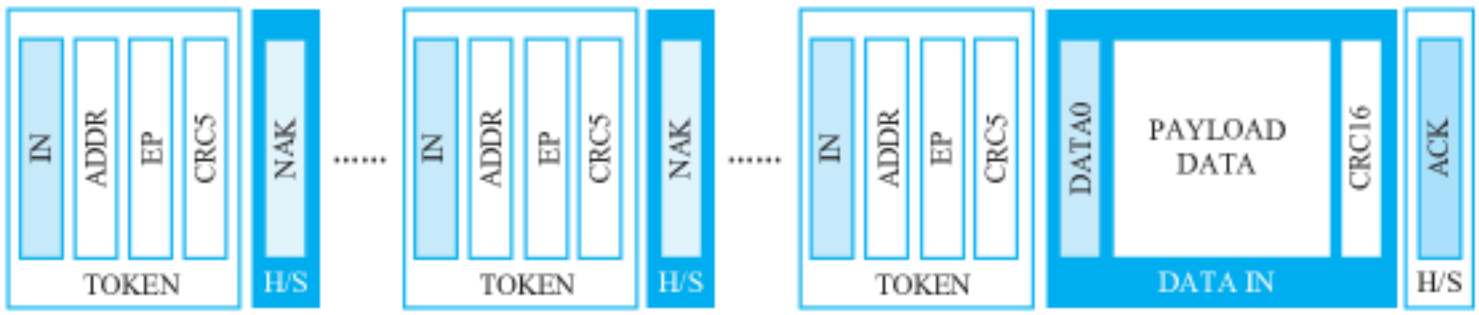

图 15.18　IN 数据传输示例

（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 15.19 中，白框显示的是从主机发送的数据包，黑框显示的是从设备发送的数据包。

图 15.19　OUT/写入/下行框图

在图 15.20 中，主机将发送 OUT 令牌数据包和 DATA0 数据包，如果设备未准备好，主机则会接收到设备所发送的 NAK 包。然后，主机会重新发送数据。注意，设备拒绝接收来自主机的数据，不会改变数据包 ID 的状态。如果主机再次尝试发送数据，设备若准备好，设备将发送一个 ACK 信号来响应主机，表示 OUT 数据传输已经成功。

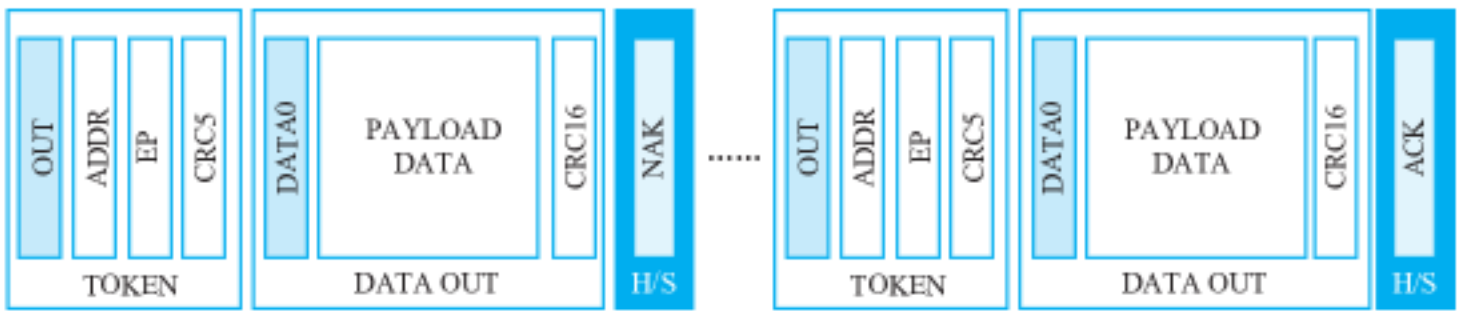

图 15.20　OUT 数据传输示例

15.1.3　USB 通信模型

本节介绍的是 USB 通信模型中的逻辑通信模型，该模型中的许多概念将会在后续的程序设计中用到，希望读者能仔细阅读。

1. USB 管道

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 15.21 所示

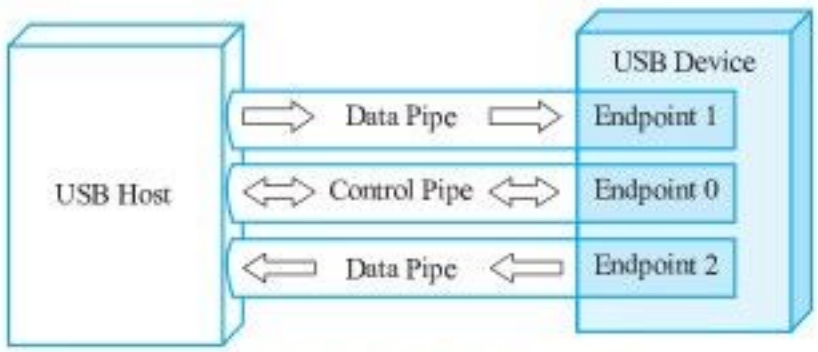

图 15.21 USB 管道与端点模型

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。设备可以有任意多个数据管道,设备可以通过中断、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。主机会通过向设备发送令牌数据包来获取设备的地址和端点编号。

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(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,例如:我们接入 U 盘,主机完成枚举过程后,我们就可以正常使用 U 盘与主机进行数据互传。

2. 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。简而言之 USB 端点中有一个叫做端点缓冲区的重要组成部分,USB 主机端通过发送 USB 数据到 USB 设备端的缓冲区实现数据的收到功能。而 USB 设备通过向缓冲区写入数据实现 USB 设备向主机发送数据的功能。

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尺寸。根据设计要求,开发者使用端点描述符用以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。它的优点是可以保证传输准确。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。实际通信中，使用的不是并中断，而是轮询。进行该传输时，主机将在规定的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意大带宽空间的传输。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，便不能预测实际的传送时间。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据流。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，通常需要使用三个缓冲区，一个正在传输数据、一个已加载数据和一个正在进行加载数据。

表 15.5　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	1.216MB/s	1.216 MB/s	1.023 MB/s

[1]同步和中断端点的共享带宽。

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。在枚举过程中,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(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一通常有一个 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。USB 硬件会进行 CRC 检验。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。

3. USB 传输方式

在 USB 管道和 USB 端点一节我们可以知道:端点的类型和 USB 数据传输的方式是相对应的。本节将详细讲述 USB 的三种传输方式。

1) 控制传输

控制传输是一种特殊的传输方式。当 USB 设备初次连接主机时,用控制传输传送控制命令等对设备进行配置。同时设备接入主机时,需要通过控制传输去获取 USB 设备的描述符以及对设备进行识别,在设备的枚举过程中都是使用控制传输进行数据交换。在后面程序设计一节中将会对控制传输的具体实现作出详细的解析。

控制传输主要应用于 USB 设备的枚举过程,其他应用场合读者可自行了解,本文不作过多介绍。

控制传输最大包长度不超过控制端点的大小。

控制传输由三个阶段构成,分别是建立阶段(或称之为设置阶段)、数据阶段和状态阶段。程序设计中控制传输部分设备对请求处理函数将主要按这三个阶段进行编写。

(1) 建立阶段

该过程具体描述为:

① 主机发送令牌包:SETUP

② 主机发送数据包:DATA0

③ 设备返回握手包:ACK 或不应答

注意:设备不能返回 NAK 或 STALL,即设备必须接收建立阶段的数据。设备只能使用 ACK 来应答(或者由于出错不应答)。

(2) 数据阶段

数据阶段是控制传输中可选的,需根据实际情况而定。数据阶段的通信是单向的,只能是主机发送数据给设备(OUT),或者设备发送数据给主机(IN)。如果通信方向发生了变化,则认为进入了状态阶段。数据阶段的第一个数据包必须为 DATA1,然后每次正确传输一个数据

包后就在 DATA0 和 DATA1 之间交替。

(3) 状态阶段

状态阶段的传输方向和数据阶段相反,即数据阶段是通信,则状态阶段必然是下行通信。且该阶段只能使用 DATA1 数据包。

控制传输之所以如此复杂,乃是为了数据传输的完整性,确保数据传输的正确无误。USB 设备在枚举过程使用的乃是控制传输,且控制传输只能在端点 0 进行。

2) 中断传输

中断传输一般用于小批量的和非连续的数据传输,通俗的来说就是用于数据量小的数据不连续的但实时性高的场合的一种传输方式,主要应用于 HID 设备中的 USB 鼠标和 USB 键盘等。

在中断端点中有介绍:USB 中断传输和我们传统意义上的中断不一样。它不是由设备主动地发起一个中断请求,主机响应,而是主机将在规定的时间间隔内检查数据。所以 USB 的中断传输的实际意义是实时轮询操作,即 USB 的中断传输是主机在一定的时间不断地主动轮询设备检查其是否有数据需要传输。

中断传输有 3 个重要参数需要在端点描述符中进行配置:即传输类型、每次传输的最大数据包大小、轮循时间间隔。

中断端点需指定一个范围在 1~255 ms 内的轮询周期。主机保留足够的带宽以确保在指定频率上直接向中断端点发出 IN 或 OUT 事务。对于中断传输,如果传输的数据长度大于端点支持的最大包长度,这时一个中断传输内会有多个事务。在传输数据时,如果最后一个事务的数据长度小于端点支持的最大包长度,则认为数据传输完成。

对于 STC32 的端点每个端点的大小为 64 字节,故端点支持的最大包长为 64 字节,若 STC32 仅模拟 HID 键盘设备或 HID 鼠标设备,中断端点每次仅要传输 1 字节数据即可。

中断传输事务

USB 中断数据流传输包括 IN 传输和 OUT 传输,分别对应于数据的读和写,其也分为 3 个阶段,分别为令牌阶段、数据段和握手段。

中断传输和批量传输的结构基本一致,只是中断传输没有 ping 和 nyet 两种包。

当主机准备接收 USB 设备的中断端点的数据时,其发送 IN 令牌包,USB 设备响应并返回 DATAx 数据包,NAK 或 STALL 握手包。

当主机向 USB 设备的中断端点发送数据时,其发送 OUT 令牌包和 DATAx 数据包,而 USB 设备将向主机返回 ACK、NAK 和 STALL 握手包。如图 15.22 为一个中断传输事务在不同阶段数据包的构成。

令牌段(Token)

- 主机发出令牌包,寻址设备。

数据段(Data)

- 设备如果接收令牌包出错,无响应;
- 设备端点不存在,设备回复 STALL 包;
- 设备端点数据未准备好,设备回复 NAK 包;
- 设备端点数据准备好,设备回复数据包。

握手段(handshake)

- 主机如果接收数据包出错,无响应;
- 主机如果接收数据包正确,设备回复 ACK 包。

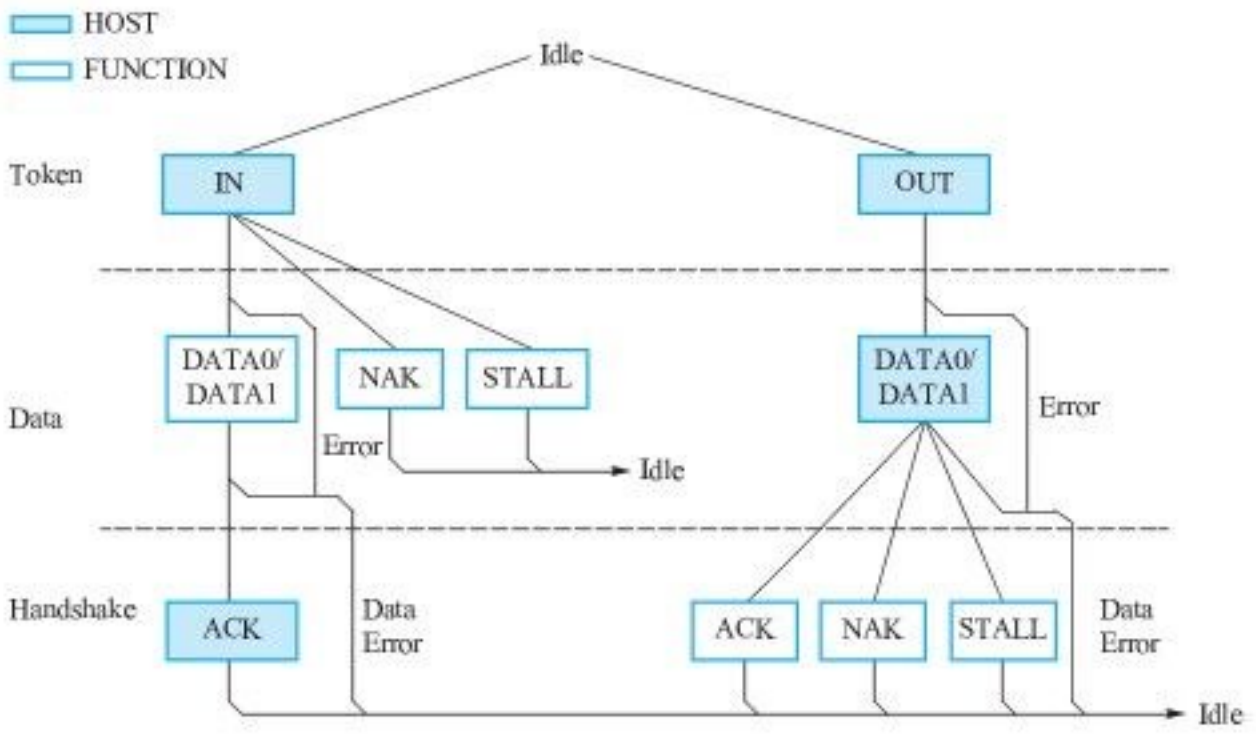

图 15.22　中断传输事务在不同阶段数据包的构成

3）批量传输

批量传输一般用于批量的和非实时的数据传输，通俗的来说就是用于数据量大但对时间要求又不高的场合的一种传输方式，类似用于 USB 打印机和 USB 扫描仪等等。

批量传输使用批量传输事务，一次批量传输事务分为三个阶段：令牌包阶段、数据包阶段、握手包阶段。

批量传输分为批量读和批量写，批量读使用批量输入事务，批量写使用批量输出事务。注意：不论输入还是输出都是以主机为参考的。

对于批量传输，如果启动批量传输，如果 USB 总线中有多余的总线带宽，批量传输会立即执行，但当带宽比较紧张时，批量传输会把带宽让给其他传输类型。所以批量传输的优先级相对其它传输优先级比较低。

批量传输数据包

只有全速和高速设备可以使用批量传输，低速模式不支持批量传输。

高速模式中，传输的数据包大小固定为 512 个字节；

全速模式中，传输的数据包大小可在 8、16、32、64 字节中选择；

当为超高速设备时数据包最大长度为 1024 字节，批量传输端点应在其端点描述符中设置最大的数据包负载大小为 1024 字节。它还指定端点可以接受或者发送到超高速总线的突发大小。对于批量端点允许的突发大小应在 1 至 16 范围。

批量功能端点必须传输数据字段小于或等于 1024 字节的数据负载。如果批量传输有比之更多的数据，在突发事务交易的所有数据的有效大小必须为 1024 字节长度，除了突发的最后一个数据有效载荷，它可能包含未使用的数据空间。

如果传输的数据量大于端点所支持的最大数据包长度，USB 主控制器会把该数据按最大数据包长度分为多个批量数据包进行传输，最后一个批量传输长度可以小于或等于最大包长度。

批量传输流程：

批量传输数据流传输包括 IN 传输和 OUT 传输，分别对应于数据的读和写，其也分为 3 个阶段，分别为为令牌阶段、数据段和握手段。

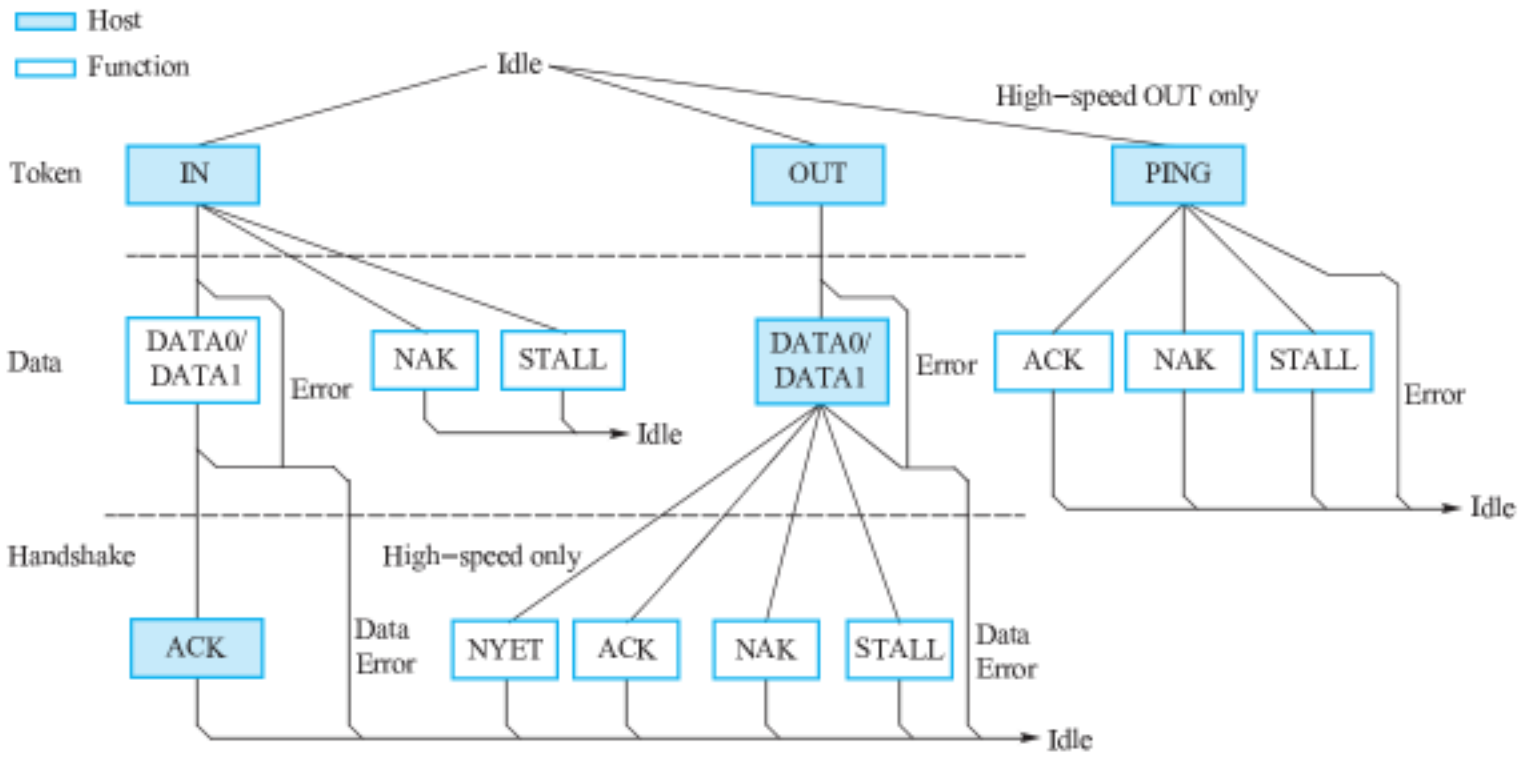

图 15.23　批量传输不同阶段数据包构成

对于批量传输的最后一个 IN 事务：

- 如果 USB 主机收到的数据长度小于端点支持的最大包长度，那么 USB 主机认为数据已经接收完成。
- 如果 USB 主机收到的数据长度等于端点支持的最大包长度，需要额外的 0 数据的包，告诉 USB 主机数据已经接收完成。

批量输出流程 OUT

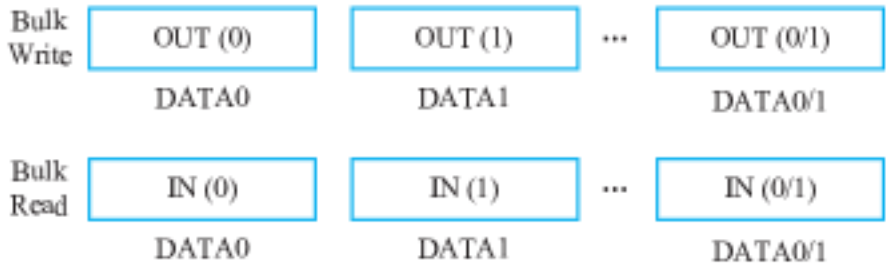

图 15.24　批量传输数据传输简要框图

令牌阶段

- 主机发送 BULK 令牌包，令牌包中包含设备地址、端点号和数据方向。

数据段

- 设备如果接收令牌包出错，无响应，让主机等待超时；
- 设备端点不存在，设备回复 STALL；
- 设备端点数据未准备好，设备回复 NAK；
- 设备端点数据准备好，设备回复数据包。

握手端设备

- 数据包正确，并有足够的空间保存数据：设备返回 ACK 握手包或 NYET 握手包（只有高速模式才有 NYET 握手包，它表示本次数据接收成功，但是没有能力接收下一次传输）。
- 数据包正确，但是没有足够的空间保存数据：设备返回 NAK 握手包。主机收到 NAK，延时一段时间后，再重新进行批量输出事务。
- 数据包正确，但端点处于挂起状态：设备返回一个 STALL 握手包。

- 数据包错误:设备不返回任何握手包,让主机等待超时。
- CRC 错误或位填充错误:设备不返回任何握手包,让主机等待超时。

15.1.4 USB 标准请求

USB 标准请求是一个大小为 8 个字节数据结构,USB 标准请求用于:一是对设备进行枚举,二是对设备的状态进行更改。USB 的标准请求的数据传输方式都是控制传输方式,所以使用的端点是设备的默认端点 0。

后续的程序设计环节我们将只从设备的角度来对主机发送的 USB 标准请求进行处理,为了了解如 USB 设备何响应 USB 主机发送的 USB 标准请求,需要了解请求的数据结构以便知道 USB 主机发送的是哪一种 USB 标准请求,需要了解每一个标准请求设备需要作何响应。

为了便于理解我们将 USB 标准请求又称作 SETUP 数据结构,在后续的程序设计中,设备对 USB 主机的标准请求的响应将会单独使用一个.c 文件进行编写。后续 SETUP 数据结构将频繁使用希望读者看到此概念知道说的是 USB 标准请求。

注意:除了 USB 标准请求外,USB 还有特定类的请求,在后续的程序设计中会介绍其中两种特定类请求,分别为 HID 特定类请求 BOT 特定类请求。

1. SETUP 数据结构

SETUP 数据结构由 5 个字段构成,其分别为:

- 1 字节的 bmRequestType;
- 1 字节的 bReqest
- 2 字节的 wValue
- 2 字节的 wInde
- 2 字节的 wLength

以上所有字段合起来共 8 字节。SETUP 数据结构可以描述为下表 15.6

表 15.6 SETUP 数据结构

偏移量	字段	大小/字节	取值	含义
0	bmRequestType	1		请求特性 D7:数据传输方向 0=从主机到设备 1=从设备到主机 D6-D5:请求的类型 0=标准类型 1=类类型 2=厂商类型 3=保留 D4-0:请求的接收者 0=设备 1=接口 2=端点 3=其他 其余:保留

续表

偏移量	字段	大小/字节	取值	含义
1	bRequest	1	数值	请求代码
2	wValue	2	数值	该域的意义由具体请求决定
4	wIndex	2	索引或偏移量	该域的意义由具体请求决定
6	wLength	2	字节数	数据过程所需要传输的字节数

2. USB 标准请求分类

请求的类型由 SETUP 数据结构的 bRequest 字段指定，bRequest 字段值指定的请求类型和大致功能可以由下表 15.7 进行描述：

表 15.7　标准请求及其功能

请求	bRequest 字段值	功能
ClearFeature	1	清除设备、接口的某种特征(或性能)
GetConfigration	8	获取指定设备当前的配置值
GetDescriptor	6	获取设备的某种标准描述符
GetInterface	10	获取设备接口当前工作的选择设置值
GetStatus	0	获取设备、接口或端点的某种状态
SetAddress	5	为设备设置唯一的地址
SetConfiguration	9	激活设备的某个配置
SetDescriptor	7	主机会更新或创立描述符
SetFeature	3	主机启用一个在设备、接口或端点上的特征
SetInterface	11	主机激活设备的某个接口的设置值
SynchFrame	12	在实时传输中，用于同步某个帧开始传输序列

3. USB 标准请求的功能

(1) ClearFeature

ClearFature 请求用于清除或禁用 USB 设备、接口或端点的某些特性，该请求无数据阶段，其每个字段具体如表 15.8。

表 15.8

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0x00 0x01 0x02	CLEAR_FEATURE	特性选择	0 接口号 端点号	0	无

- bmRequestType 值为 00,表示从主机到设备,设备接收
- bmRequestType 值为 01,表示从主机到设备,接口接收
- bmRequestType 值为 02,表示从主机到设备,端点接收

表 15.9 为特性选择字段的详细内容。

表 15.9

wValue	接收者	特性名	功能
0	端点	ENDPOINT_HALT	挂起端点
1	USB 设备	DEVICE_REMOVE_WAKEUP	远程唤醒设备
2	USB 设备	TEST_MODE	用于 USB 测试

(2) GetConfigration

GetConfiguration 用于主机读取 USB 设备当前的配置值,在 GetConfiguration 的数据阶段,USB 设备将向主机返回一个字节的配置值。

表 15.10

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0x80	GET_CONFIGURATION	0	0	1	配置值

wLength 字段值指明 USB 设备返回的配置值的大小

如果设备已经配置好,设备接收该请求会发送则发送 PACKET1 数据包,如果未被配置则发送 PACKET0 数据包。

(3) GetDescriptor

GetDescriptor 用于 USB 主机读取设备的描述符,在请求数据阶段,USB 设备将向主机返回指定的描述符。GetDescriptor 请求是 USB 通信中最常使用的请求,有关此请求的内容十分重要,可以说 STEUP 阶段的主要数据负载就在此请求的响应中。

表 15.11

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0x80	GET_DESCRIPTOR	描述符类型和索引	0 或者语言 ID	描述符长度	描述符

其中 wValue 字段的值指明需要发送的描述符类型,其具体指明的设备类型如表 15.12。

表 15.12

wValue	值	描述符类型
DESC_DEVICE	0x0100	设备描述符
DESC_CONFIGURATION	0x0200	配置描述符
DESC_STRING	0x0300 0x0301 0x0302	LANGIDDESC MANUFACTDESC MANUFACTDESC
DESC_HIDREPORT	0x2200	报告描述符

注意：

1. wValue 的第一字节(字符)表示同一中描述类型(比如字符串描述符)中具体的某个描述符(如厂商或者产品字符),第二字节表示描述类型的编号。

2. 对于全速和低速模式,获取描述符的标准请求只有三种:获取设备、配置、字符串的描述符,另外的接口和端点描述符是跟随配置描述符一并返回的,不能单独请求返回。

(4) GetInterface

GetInterface 请求用于 USB 主机读取指定接口的设置值,即获取接口描述符中 bAlternateSetting 字段中的值。在 GetInterface 请求的数据阶段,USB 设备向 USB 主机返回 1 个字节的可替换设置值。在程序设计中若主机发出该请求,我们仅需使用 PACKET0 对请求进行回复

表 15.13

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0x80	GET_INTERFACE	0	接口号	1	备用接口号

(5) GetStatus

GetStatus 请求主要用于 USB 主机读取 USB 设备、接口或端点的状态。USB 设备返回 2 字节的设备状态。其每个字段具体的值如表 15.14 所示。

表 15.14

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0x80 0x81 0x82	GET_STATUS	0	0 接口号 端点号	2	设备、接口或者端点状态

其中,wLength=2 表示 USB 设备返回的数据为 2 字节。

bmRequestType 共三个值,其代表的功能如表 15.15 所示。

表 15.15

bmRequestType	值	功能
DEVICE_RECIPIENT	0x80	获取设备状态
INTERFACE_RECIPIENT	0x81	获取接口状态
ENDPOINT_RECIPIENT	0x82	获取端点状态

1) 若为 DEVICE_RECIPIENT,设备将返回,如表 15.16 所示。

表 15.16

偏移	D15-D2	D1	D0
含义	保留为 0	远程唤醒	自供电

2) 若为 INTERFACE_RECIPIENT,设备将返回数据 0。

3) 若为 ENDPOINT_RECIPIENT,设备将返回,如表 15.17 所示。

表 15.17

偏移	D15-D1	D0
含义	保留为 0	端点是否已停止(1 停止,0 未停止)

(6) SetAddress

SetAddress 用于枚举(enumeration)阶段为设备分配一个唯一的地址,地址在 wValue 字段中且最大值为 127。该请求特别的地方在于,直到状态阶段完成,设备才完成地址设置。其他所有请求必须在状态阶段之前完成。该请求同样无数据阶段,如表 15.18 所示。

表 15.18

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0x00	SET_ADDRESS	设备地址	0	0	无

wValue 为主机给设备分配的唯一地址,设备若有 USB 控制寄存器,如 STC32G 有一个寄存器 FADDR,该寄存器将会存放主机为设备分配的唯一地址。

(7) SetConfiguration

SetConfiguration 和 SetAddress 请求很类似,区别是 wValue 字段的意义:SetAddress 中,wValue 的第一字节(低字节)表示设备的地址;SetConfiguration 则为配置的值。该值与配置描述符的配置编号一致时,表示选中该配置,通常为 1,因为大多数 USB 设备只有一种设置;若为 0,则设备进入地址设置状态,如表 15.19 所示。

表 15.19

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0x00	SET_CONFIGURATION	配置值(0x0001)	0	0	无

一般的设备只有一个配置,当有多个配置时,会让用户选择。一个设备只能工作在一个配置状态下。

(8) SetDescriptor(*)

用于修改选设备中需要修改的描述符,或者增加新的描述符,在 SetDescriptor 请求的数据阶段,主机将向 USB 设备发送指定的描述符类型,如表 15.20 所示。

表 15.20

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0x00	07	类型和索引	00 或语言 ID	0	无

(9) SetFeature

SetFeature 请求用于设置或使能 USB 设备、接口或端点的特性值,如表 15.21 所示。

表 15.21

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0x00 0x01 0x02	SET_FEATURE	特性选择	0 接口号 端点号	0	无

本次设计仅设置 USB 设备对应的端点产生 stall 信号进行回应。

(10) SetInterface

SetInterface 请求用于 USB 主机为设备指定的接口选择一个合适的替换值,如表 15.22 所示。

表 15.22

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0x01	SET_INTERFACE	可替换的接口值	接口号	0	无

SetInterface 请求只在 USB 处于配置状态时有效。

当 USB 设备的一个接口存在 1 个或多个可替换设置时,SetInterface 请求使得主机可以为其选择所需要的可替换值。

(11) SynchFrame(*)

SynchFrame 用于设置并报告端点的同步帧号,用于同步传输,只适用于同步端点。在 SynchFrame 请求的数据阶段,USB 设备将向 USB 主机返回 2 个字节的帧号数据,如表 15.23 所示。

表 15.23

bmRequestType	bRequest	wValue	wIndex	wLength	数据过程
0x82	SYNCH_FRAME	0	端点号	2	帧号

SynchFrame 请求只在 USB 设备处于配置状态时有效。

15.1.5 USB 通信过程

1. USB 枚举和配置

USB 设备首次连接时会进行该过程,USB 物理连接后最初的通信过程。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。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。主机用来了解设备信息的请求是标准请求。所有 USB 设备都要支持这类请求。

以下为枚举的全部过程。

1) 动态检测

第一步:设备连接到 USB 端口上,并被主机检测。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。

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(0V),使设备进入复位状态。D+和 D-线处于低电平状态的持续时间为 2.5μs,因此发生复位条件。集线器在 10ms 内维持复位状态。

第六步:复位期间发生一系列 J 状态和 K 状态,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 状态。高速集线器检测该 K 状态并用 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,直到它获取设备信息退出复位状态为止。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详细的信息。首先,它要知道默认管道(端点 0)的最大数据包大小。主机首先向设备发送 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)包含了有关端点 0(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 字节设备描述符后它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。

3) 配置

第十步:设备退出复位状态后,主机会发送 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,以便请求设备发送配置描述符。该请求的结果不仅是主机获取了设备的配置描述符,并且还获取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 首先只请求设备发送配置描述符(9 个字节),然后它会发送第二个 GET_DESCRIPTOR 请求,以请求设备发送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。设备将按照描述符所定义的功能进行操作。所定义的最大电源是从 Vbus 吸取的。

现在就可以在应用中使用设备。(在 HID 键盘或鼠标中对应的是中断传输,在 CDC 中对应的也是中断传输)

15.1.6 USB 描述符

本节仅对 USB 描述符的数据结构进行说明,数据结构中详细字段的值所代表的含义还请读者阅读 USB 协议文档中描述符部分的内容,本次实验的描述符文件通过打开工程目录下的 usb_desc.c 文件进行查阅。

1. 设备描述符:

设备描述符是 USB 设备的第一个描述符,每个 USB 设备都得具有设备描述符,且只能拥有一个。设备描述符中包含了 USB 规范、设备配置编号、设备支持的协议等信息。设备描述符的数据结构可以描述如表 15.24 所示。

表 15.24 设备描述符表数据结构

偏移	字段	大小(字节)	说明
0	bLength	1	设备描述符的大小,共 18 个字节,该字段固定为 0x12
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	iSeria Number	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 则显小了设备支持的 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 OxAABC 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,USB2.0 设备拥有 0X0200 值,USB1.1 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。
- bDeviceCIass、bDeviceSubCIass 和 bDeviceProtocoI 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 iSeriaINumber 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。如果无字符串,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。

2. 配置描述符:

接口数量、设备供电方法。注意:在实际获取配置描述符时,设备会将配置描述符、接口描述符、端点描述符、HID 描述符打包在一起发送。配置描述符的数据结构如表 15.25 所示。

表 15.25　配置描述符表数据结构

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	Configuration	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 是本配置的整个层次的长度。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。
- bConfigurationVaIue 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。

3. 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,此处结合程序设计进行简要介绍,在 HID 鼠标设备中,用于标识接口类别的 bInterfaceClass 字段值为 3,用于标识接口子类别的 bInterfaceSubClass 字段值为 1,用于识别 HID 鼠标设备的 bInterfaceProtocol 字段的值为 2。接口描述符的数据结构如表 15.26 所示。

表 15.26　接口描述符表数据结构

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	blnterfaceNumber	1	该接口基于零的索引
3	bAIternateSetting	1	备用设置值
4	bNumEndpoints	1	该接口所使用的端点数量(不包含 EPO)
5	blnterfaceClass	1	接口类别
6	blnterfaceSubclass	1	接口子类别
7	blnterfaceProtoco	1	接口协议
8	ilnterface	1	该接口字符串描述符索引

4. 端点描述符:

由于主机通过端点与地址的组合与设备通信,因此该描述符存放了主机所需要的设备端点的信息。端点述符的数据结构如表 15.27 所示。

表 15.27　端点描述符数据结构

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)

续表

偏移	字段	大小(字节)	说明
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 到位 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) 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	blnterval	1	中断端点的轮询间隔,单位为 ms(对于同步端点,间隔为 1ms;控制或批量端点可能忽略该字段)

5. 字符串描述符:

该描述符为主机提供了设备名称、生产厂家、序列号或不同接口等信息。字符串描述符的数据结构如表 15.28 所示。

表 15.28　字符串描述符数据结构

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	bStrinwLangID	变化	Unicode 编码字符串或 LANGID 代码

6. 使用多个 USB 描述符

每个 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 15.25 显示的是单个 USB 设备中如何分配两种接口。

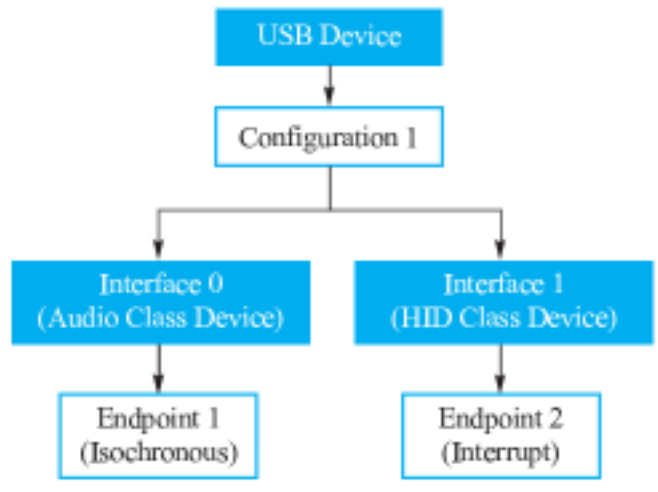

图 15.25　多个接口的设置框图

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 15.26 详细演示了该示例。

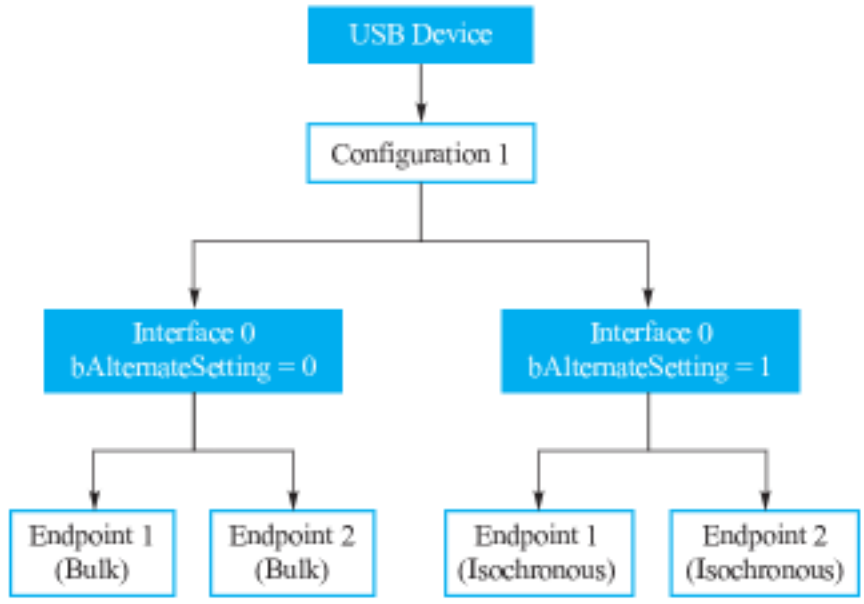

图 15.26　多接口设置框图

图 15.27 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。

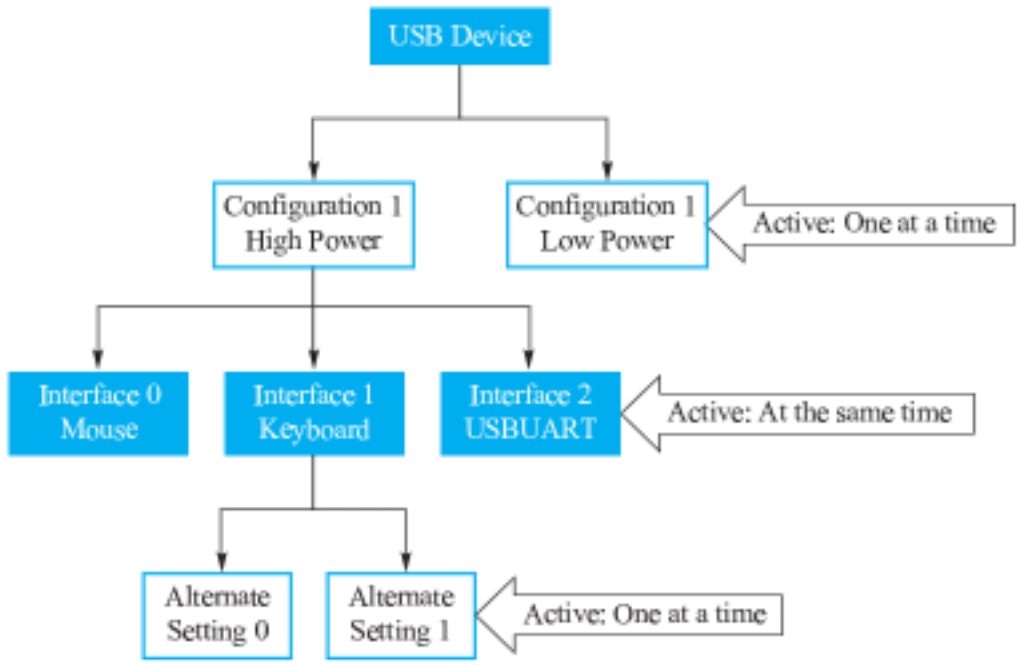

图 15.27　配置框图