第 1 章　　绪论

1.1　起因

Intel 公司开发的通用串行总线架构(USB)的目的主要基于以下三方面考虑：

(一)计算机与电话之间的连接：显然用计算机来进行计算机通信将是下一代计算机基本的应用。机器和人们的数据交互流动需要一个广泛而又便宜的连通网络。然而，由于目前产业间的相互独立发展,尚未建立统一标准,而 USB 则可以广泛的连接计算机和电话。

(二)易用性：众所周知,PC 机的改装是极不灵活的。对用户友好的图形化接口和一些软硬件机制的结合，加上新一代总线结构使得计算机的冲突大量减少，且易于改装。但以终端用户的眼光来看，PC 机的输入/输出，如串行/并行端口、键盘、鼠标、操纵杆接口等，均还没有达到即插即用的特性，USB 正是在这种情况下问世的。

(三)端口扩充：外围设备的添加总是被相当有限的端口数目限制着。缺少一个双向、价廉、与外设连接的中低速的总线，限制了外围设备(诸如电话/电传/调制解调器的适配器、扫描仪、键盘、PDA)的开发。现有的连接只可对极少设备进行优化，对于 PC 机的新的功能部件的添加需定义一个新的接口来满足上述需要，USB 就应运而生。它是快速、双向、同步、动态连接且价格低廉的串行接口，可以满足 PC 机发展的现在和未来的需要。

1.2　USB 规范的目标

本书规范了 USB 的工业标准。该规范介绍了 USB 的总线特点、协议内容、事务种类、总线管理、接口编程的设计，以及建立系统、制造外围设备所需的标准。

设计 USB 的目标就是使不同厂家所生产的设备可以在一个开放的体系下广泛的使用。该规范改进了便携商务或家用电脑的现有体系结构，进而为系统生产商和外设开发商提供了足够的空间来创造多功能的产品和开发广阔的市场，并不必使用陈旧的接口，害怕失去兼容性。

1.3　适用对象

• 该规范主要面向外设开发商和系统生产商。并且提供了许多有价值的信息给操作系统/BIOS/设备驱动平台、IHVS/ISVS 适配器，以及各种计算机生产厂家使用。

• 该 USB 版本的规范可以用来设计开发新产品，改进一些经典的模型，并开发相应的软件。所有的产品都应遵循这个规范——USB 1.1。

1.4　本书结构

第一章至第四章为读者提供了一个纲要，第五章至第十章则提供了 USB 的所有的具体技术细节。

• 外设厂家应着眼于第四章至第十章

• USB 的主机控制器应用主要参考第四章至第七章和第九、十章。

• USB 设备驱动厂家主要参考第四、七、九章

《Universal Serial Bus Device Class Specification》一书可以作为本书的补充和参考。各种设备的规范是形形色色的，如有疑问，请与 USB Implements Forum 索要更多细节。

读者也可以为向操作系统厂商索取关于 USB 的一些具体特性。

第 2 章　　背景知识

本章将对 USB 背景知识作简单描述，其中主要包括设计目标、总线特性，以及现行技术特点。

2.1 USB 的设计目标

USB 的工业标准是对 PC 机现有的体系结构的扩充。USB 的设计主要遵循以下几个准则：

- 易于扩充多个外围设备；
- 价格低廉，且支持 12M 比特率的数据传输；
- 对声音音频和压缩视频等实时数据的充分支持；
- 协议灵活，综合了同步和异步数据传输；
- 兼容了不同设备的技术；
- 综合了不同 PC 机的结构和体系特点；
- 提供一个标准接口，广泛接纳各种设备；
- 赋予 PC 机新的功能，使之可以接纳许多新设备。

2.2 使用的分类

表 2-1 按照数据传输率(USB 可以达到)进行了分类。可以看到，12M 比特率可以包括中速和低速的情况。总的来说，中速的传输是同步的，低速的数据来自交互的设备，USB 设计的初衷是针对桌面电脑而不是应用于可移动的环境下的。软件体系通过对各种主机控制器提供支持以保证将来对 USB 的扩充。

性能	*应用*	*特性*
低速 • 交互设备 • 10-20kb/s	键盘、鼠标、游戏棒	低价格、热插拔、易用性
中速 • 电话、音频、压缩视频 • 500kb/s-10Mb/s	ISBN、PBX、POTS	低价格、易用性、动态插拔、限定带宽和延迟
高速 • 音频、磁盘 • 25-500Mb/s	音频、磁盘	高带宽、限定延迟、易用性

表 2-1

2.3 特色

USB 的规范能针对不同的性能价格比要求提供不同的选择，以满足不同的系统和部件及相应不同的功能，其主要特色可归结为以下几点：

终端用户的易用性：

- 为接缆和连接头提供了单一模型；
- 电气特性与用户无关；
- 自我检测外设，自动地进行设备驱动、设置；
- 动态连接，动态重置的外设。

广泛的应用性：

- 适应不同设备，传输速率从几千比特率到几十兆比特率；
- 在同一线上支持同步、异步两种传输模式；
- 支持对多个设备的同时操作；
- 可同时操作 127 个物理设备；
- 在主机和设备之间可以传输多个数据和信息流；
- 支持多功能的设备；
- 利用低层协议，提高了总线利用率。

同步传输带宽：

- 确定的带宽和低延迟适合电话系统和音频的应用；

• 同步工作可以利用整个总线带宽。

灵活性：

• 直接一系列大小的数据包，允许对设备缓冲器大小的选择；

• 通过指定数据缓冲区大小和执行时间，支持各种数据传输率；

• 通过协议对数据流进行缓冲处理。

健壮性：

• 出错处理/差错恢复机制在协议中使用；

• 对用户感觉而言，热插拔是完全实时的；

• 可以对有缺陷设备进行认定。

与 PC 产业的一致性：

• 协议的易实现性和完整性；

• 与 PC 机的即插即用的体系结构的一致；

• 对现存操作系统接口的良好衔接。

价廉物美

• 以低廉的价格提供 1.5 兆比特率的子通道设施；

• 将外设和主机硬件进行了最优化的集成；

• 促进了低价格的外设的发展；

• 廉价的电缆和连接头；

• 运用了商业技术。

升级路径：

• 体系结构的可升级性支持了在一个系统中可以有多个 USB 主机控制器。

第 3 章　　体系结构概述

本章主要内容是关于 USB 的概述和一些关键的概念。USB 是一种电缆总线，支持在主机和各式各样的即插即用的外设之间进行数据传输。由主机预定的标准的协议使各种设备分享 USB 带宽，当其它设备和主机在运行时，总线允许添加、设置、使用以及拆除外设。

后续章节将着重描述 USB 的细节。

3.1　USB 系统的描述

一个 USB 系统主要被定义为三个部分：

• USB 的互连；

• USB 的设备；

• USB 的主机。

USB 的互连是指 USB 设备与主机之间进行连接和通信的操作，主要包括以下几方面：

• 总线的拓扑结构：USB 设备与主机之间的各种连接方式；

• 内部层次关系：根据性能叠置，USB 的任务被分配到系统的每一个层次；

• 数据流模式：描述了数据在系统中通过 USB 从产生方到使用方的流动方式；

• USB 的调度：USB 提供了一个共享的连接。对可以使用的连接进行了调度以支持同步数据传输，并且避免的优先级判别的开销。

USB 的设备及主机的细节将讲述于后。

3.1.1　总线布局技术

USB 连接了 USB 设备和 USB 主机，USB 的物理连接是有层次性的星型结构。每个网络集线器是在星型的中心，每条线段是点点连接。从主机到集线器或其功能部件，或从集线器到集线器或其功能部件，从图 3-1 中可看出 USB 的拓扑结构。

3.1.1.1 USB 的主机

在任何 USB 系统中，只有一个主机。USB 和主机系统的接口称作主机控制器，主机控

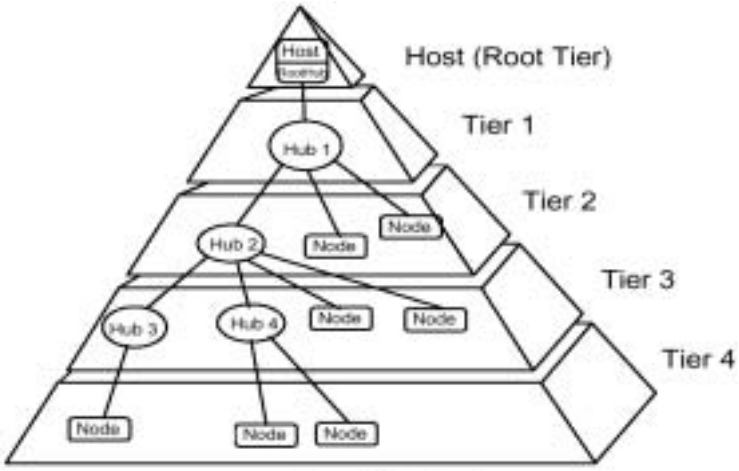


图 3－1 总线的拓扑结构

制器可由硬件、固件和软件综合实现。根集线器是由主机系统整合的，用以提供更多的连接点。关于主机更多的内容可参见 4.9 节和第 9 章。

4.1.1.2 USB 的设备

USB 的设备如下所示：

- 网络集线器，向 USB 提供了更多的连接点；
- 功能器件：为系统提供具体功能，如 ISDN 的连接，数字的游戏杆或扬声器。

USB 设备提供的 USB 标准接口的主要依据：

- 对 USB 协议的运用；
- 对标准 USB 操作的反馈，如设置和复位；
- 标准性能的描述性信息；

关于 USB 设备的更多信息请参见 3.8 节和第 8 章。

3.2 物理接口

USB 的物理接口的电气特性在第六章,机械特性在第五章详细介绍。

3．2．1 电气特性

USB 传送信号和电源是通过一种四线的电缆，图 3-2 中的两根线是用于发送信号。存在两种数据传输率：

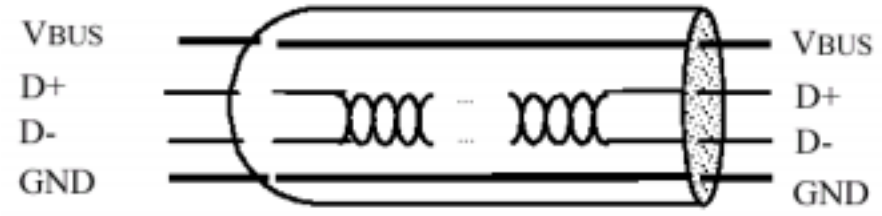


图 3-2 USB 的电缆

- USB 的高速信号的比特率定为 12Mbps；
- 低速信号传送的模式定为 1.5Mbps；

低速模式需要更少的 EMI 保护。两种模式可在用同一 USB 总线传输的情况下自动地动态切换。因为过多的低速模式的使用将降低总线的利用率，所以该模式只支持有限个低带宽的设备(如鼠标)。时钟被调制后与差分数据一同被传送出去，时钟信号被转换成 NRZI 码，并填充了比特以保证转换的连续性，每一数据包中附有同步信号以使得收方可还原出原时钟信号。

电缆中包括 VBUS 、GND 二条线，向设备提供电源 。VBUS 使用+5V 电源。USB 对电缆长度的要求很宽，最长可为几米。通过选择合适的导线长度以匹配指定的 IR drop 和其它一些特性，如设备能源预算和电缆适应度。为了保证足够的输入电压和终端阻抗。重要的终端设备应位于电缆的尾部。在每个端口都可检测终端是否连接或分离，并区分出高速，或低速设备。

3.2.2 机械特性

电缆和连接器的机械特性将在第 5 章详述。所有设备都有一个上行的连接。上行连接器和下行连接器是不可简单的互换，这样就避免了集线器间的非法的循环往复的连接，电缆中有四根导线：一对互相缠绕的标准规格线，一对符合标准的电源线，连接器有四个方向，具有屏蔽层，以避免外界干扰，并有易拆装的特性。

3.3 电源

主要包括两方面：

- 电源分配：即 USB 的设备如何通过 USB 分配得到由主计算机提供的能源；
- 电源管理：即通过电源管理系统，USB 的系统软件和设备如何与主机协调工作。

3.3.1 电源分配

每个 USB 单元通过电缆只能提供有限的能源。主机对那种直接相连的 USB 设备提供电源供其使用。并且每个 USB 设备都可能有自己的电源。那些完全依靠电缆提供能源的设备称作“总线供能”设备。相反，那些可选择能源来源的设备称作“自供电”设备。而且，集线器也可由与之相连的 USB 设备提供电源。受一定布局限制的带有“总线供能”集线器的体系结构将在第十章讨论。在图 4-4(位于 3.8)中，键盘，输入笔和鼠标均为“总线供能”设备。

3.3.2 电源管理

USB 主机与 USB 系统有相互独立的电源管理系统。USB 的系统软件可以与主机的能源管理系统结合共同处理各种电源子件如挂起、唤醒，并且有特色的是，USB 设备应用特有的电源管理特性，可让系统软件和控制其电源管理。

USB 的电源分配和电源管理特性使之可以被设计在电源传感系统中，如采用电池的笔记本电脑。

3.4 总线协议

USB 总线属一种轮讯方式的总线，主机控制端口初始化所有的数据传输。

每一总线执行动作最多传送三个数据包。按照传输前制定好的原则，在每次传送开始时，主机控制器发送一个描述传输运作的种类、方向，USB 设备地址和终端号的 USB 数据包，这个数据包通常称为标志包(token packet)。USB 设备从解码后的数据包的适当位置取出属于自己的数据。数据传输方向不是从主机到设备就是从设备到主机。在传输开始时，由标志包来标志数据的传输方向，然后发送端开始发送包含信息的数据包或表明没有数据传送。接收端也要相应发送一个握手的数据包表明是否传送成功。发送端和接收端之间的 USB 数据传输，在主机和设备的端口之间，可视为一个通道。存在两种类型的通道：流和消息。流的数据不像消息的数据，它没有 USB 所定义的结构，而且通道与数据带宽、传送服务类型，端口特性（如方向和缓冲区大小）有关。多数通道在 USB 设备设置完成后即存在。USB 中有一个特殊的通道——缺省控制通道，它属于消息通道，当设备一启动即存在，从而为设备的设置、查询状况和输入控制信息提供一个入口。

事务预处理允许对一些数据流的通道进行控制，从而在硬件级上防止了对缓冲区的高估或低估，通过发送不确认握手信号从而阻塞了数据的传输速度。当不确认信号发过后，

若总线有空闲，数据传输将再做一次。这种流控制机制允许灵活的任务安排，可使不同性质的流通道同时正常工作，这样多种流通常可在不同间隔进行工作，传送不同大小的数据包。

3.5 健壮性

USB 健壮性的特征在于：

- 使用差分的驱动接收和防护，以保证信号完整性；
- 在数据和控制信息上加了循环冗余码(CRC)；
- 对装卸的检测和系统级资源的设置；
- 对丢失或损坏的数据包暂停传输、利用协议自我恢复；
- 对流数据进行控制，以保证同步信号和硬件缓冲管理的安全；
- 数据和控制通道的建立，使功能部件的相互不利的影响独立开，消除了负作用。

3.5.1 错误检测

USB 传输介质产生的错误率是与自然界的异常现象的概率相吻合，是瞬时一现的，因此就要在每个数据包中加入检测位来发现这些瞬时的错误，并且提供了一系列硬件和软件设施来保证数据的正确性。

协议中对每个包中的控制和数据位都提供了循环冗余码校验，若出现了循环冗余码的错误则被认为是该数据包已被损坏，循环冗余码可对一位或两位的错误进行 100%的修复。

3.5.2 错误处理

协议在硬件或软件级上提供对错误的处理。硬件的错误处理包括汇报并重新进行上一次失败的传输、传输中若遇到错误，USB 主机控制器将重新进行传输，最多可再进行三次。若错误依然存在，则对客户端软件报告错误，客户端软件可用一种特定的方法进行处理。

3.6 系统设置

USB 设备可以随时的安装和折卸，因此，系统软件在物理的总线布局上必须支持这种动态变化。

3.6.1 USB 设备的安装

所有的 USB 设备都是通过端口接在 USB 上，网络集线器知道这些指定的 USB 设备，集线器有一个状态指示器指明在其某个端口上，USB 设备是否被安装或拆除了，主机将所有的集线器排成队列以取回其状态指示。在 USB 设备安装后，主机通过设备控制通道激活该端口并以预设的地址值给 USB 设备。

主机对每个设备指定唯一的 USB 地址。并检测这种新装的 USB 设备是集线器还是功能部件。主机为 USB 设备建立了控制通道，使用指定的 USB 的地址和零号端口。

如果安装的 USB 设备是集线器，并且 USB 设备连在其端口上，那上述过程对每个 USB 设备的安装都要做一遍。

如果安装的设备是功能部件，那么主机中关于该设备的软件将因设备的连接而被引发。

3.6.2 USB 设备的拆卸

当 USB 设备从集线器的端口拆除后，集线器关闭该端口，并且向主机报告该设备已不存在。USB 的系统软件将准确进行处理，如果去除的 USB 设备上集线器，USB 的系统软件将对集线器反连在其上的所有设备进行处理。

3.6.3 总线标号

总线标号就是对连接在总线上的设备指定唯一的地址的一种动作，因为 USB 允许 USB 设备在任何时刻从 USB 上安装或拆卸，所以总线标号是 USB 的系统软件始终要作的动作，而且总线标号还包括对拆除设备的检测和处理。

3.7 数据流种类

数据和控制信号在主机和 USB 设备间的交换存在两种通道：单向和双向。USB 的数据传送是在主机软件和一个 USB 设备的指定端口之间。这种主机软件和 USB 设备的端口间的联系称作通道。总的来说，各通道之间的数据流动是相互独立的。一个指定的 USB 设备可

有许多通道。例如，一个 USB 设备存在一个端口，可建立一个向其它 USB 设备的端口，发送数据的通道，它可建立一个从其它 USB 设备的端口接收数据的通道。

USB 的结构包含四种基本的数据传输类型：

• 控制数据传送：在设备连接时用来对设备进行设置，还可对指定设备进行控制，如通道控制；

• 批量数据传送：大批量产生并使用的数据，在传输约束下，具有很广的动态范围；

• 中断数据的传送：用来描述或匹配人的感觉或对特征反应的回馈。

• 同步数据的传送：由预先确定的传送延迟来填满预定的 USB 带宽。

对于任何对定的设备进行设置时一种通道只能支持上述一种方式的数据传输，数据流模式的更多细节在第四章中详述。

3.7.1 控制数据传送

当 USB 设备初次安装时，USB 系统软件使用控制数据对设备进行设置，设备驱动程序通过特定的方式使用控制数据来传送，数据传送是无损性的。

3.7.2 批量数据传送

批量数据是由大量的数据组成，如使用打印机和扫描仪时，批量数据是连续的。在硬件级上可使用错误检测可以保证可靠的数据传输，并在硬件级上引入了数据的多次传送。此外根据其它一些总线动作，被大量数据占用的带宽可以相应的进行改变。

3.7.3 中断数据传输

中断数据是少量的，且其数据延迟时间也是有限范围的。这种数据可由设备在任何时刻发送，并且以不慢于设备指定的速度在 USB 上传送。

中断数据一般由事件通告，特征及座标号组成，只有一个或几个字节。匹配定点设备的座标即为一例，虽然精确指定的传输率不必要，但 USB 必须对交互数据提供一个反应时间的最低界限。

3.7.4 同步传输

同步数据的建立、传送和使用时是连续且实时的，同步数据是以稳定的速率发送和接收实时的信息，同步数据要使接收者与发送者保持相同的时间安排，除了传输速率，同步数据对传送延迟非常敏感。所以同步通道的带宽的确定，必须满足对相关功能部件的取样特性。不可避免的信号延迟与每个端口的可用缓冲区数有关。

一个典型的同步数据的例子是语音，如果数据流的传送率不能保持，数据流是否丢失将取决于缓冲区的大小和损坏的程度。即使数据在 USB 硬件上以合适的速率传送，软件造成的传送延迟将对那些如电话会议等实时系统的应用造成损害。

实时的传送同步数据肯定会发生潜在瞬时的数据流丢失现象，换句话说，即使许多硬件机制，如重传的引入也不能避免错误的产生。实际应用中，USB 的数据出错率小到几乎可以忽略不计。从 USB 的带宽中，给 USB 同步数据流分配了专有的一部分以满足所想得到的传速率，USB 还为同步数据的传送设计了最少延迟时间。

3.7.5 指定 USB 带宽

USB 的带宽分配给各个通道，当一个通道建立后，USB 就分配给它一定的带宽，USB 设备需要提供一些数据缓冲区。若 USB 提供了更多带宽，则需更多的缓冲区。USB 的体系要保证缓冲引导的硬件的延迟限定在几毫秒内。

USB 的带宽容量可以容纳多种不同的数据流，因此保证 USB 上可以连接大量设备，如可以容纳从 1B+D 直到 T1 速率范围的电信设备。同时 USB 支持在同一时刻不同设备具有不同比特率，并具有一个动态变动的范围。

USB 规范对总线的每类转输规定的具体的原则。

3.8 USB 设备

USB 设备分为诸如集线器、分配器或文本设备等种类。集线器类指的是一种提供 USB 连接点的设备(详见第十章)，USB 设备需要提供自检和属性设置的信息，USB 设备必须在任何时刻执行与所定义的 USB 设备的状态相一致的动态。

3.8.1 设备特性

当设备被连接、编号后，该设备就拥有一个唯一的 USB 地址。设备就是通过该 USB 地址被操作的，每一个 USB 设备通过一个或多个通道与主机通讯。所有 USB 设备必须在零号

端口上有一指定的通道，每个 USB 设备的 USB 控制通道将与之相连。通过此控制通道，所有的 USB 设备都列入一个共同的准入机制，以获得控制操作的信息。

在零号端口上，控制通道中的信息应完整的描述 USB 设备、此类信息主要有以下几类：

• 标准信息：这类信息是对所有 USB 设备的共同性的定义，包括一些如厂商识别、设备种类、电源管理等的项目。设备设置、接口及终端的描述在此给出。关于这些具体的描述信息在第九章给出；

• 类别信息：此类信息给出了不同 USB 的设备类的定义，主要反映其不同点。

• USB 厂商信息：USB 设备的厂商可自由的提供各种有关信息，其格式不受该规范制约。此外，每个 USB 设备均提供 USB 的控制和状态信息。

3.8.2 设备描述

主要分为两种设备类：集线器和功能部件。只有集线器可以提供更多的 USB 的连接点，功能部件为主机提供了具体的功能。

3.8.2.1 集线器

在即插即用的 USB 的结构体系中，集线器是一种重要设备。图 3-3 所示是

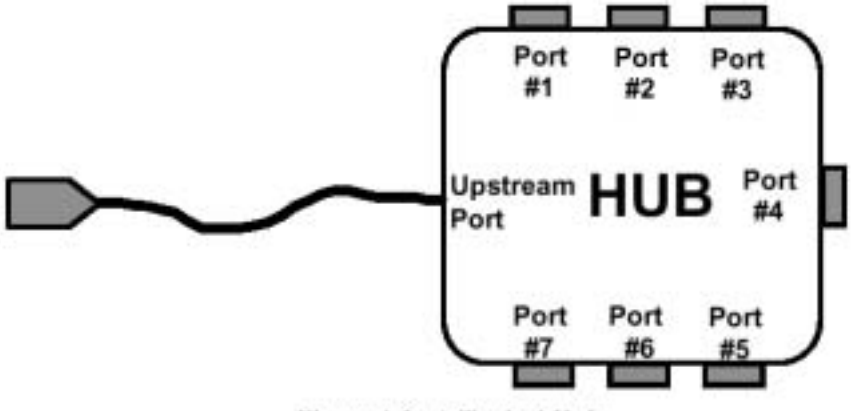


图 3 –3 典型集线器

一种典型的集线器。从用户的观点出发，集线器极大简化了 USB 的互连复杂性，而且以很低的价格和高易用性提供了设备的健壮性。

集线器串接在集中器上，可让不同性质的设备连接在 USB 上，连接点称作端口。每个集线器将一个连接点转化成许多的连接点。并且该体系结构支持多个集线器的连接。

每个集线器的上游端口向主机方向进行连接。每个集线器的下游端口允许连接另外的集线器或功能部件，集线器可检测每个下游端口的设备的安装或拆卸，并可对下游端口的设备分配能源，每个下游端口都具有独立的能力，不论高速或低速设备均可连接。集线器可将低速和高速端口的信号分开。

一个集线器包括两部分：集线控制器（Controller）和集线放大器(Repeater)。集线放大器是一种在上游端口和下游端口之间的协议控制开关。而且硬件上支持复位、挂起、唤醒的信号。集线控制器提供了接口寄存器用于与主机之间的通信、集线器允许主机对其特定状态和控制命令进行设置，并监视和控制其端口。

3.8.2.2 功能部件

功能部件是一种通过总线进行发送接收数据和控制信息的 USB 设备，通过一根电缆连接在集线器的某个端口上，功能设备一般是一种相互无关的外设。然而一个物理单元中可以有多个功能部件和一个内置集线器，并利用一根 USB 电缆，这通常被称为复合设备，即一个集线器连向主机，并有一个或多个不可拆卸的 USB 设备连在其上。

每个功能设备都包含设置信息，来描述该设备的性能和所需资源。主机要在功能部件

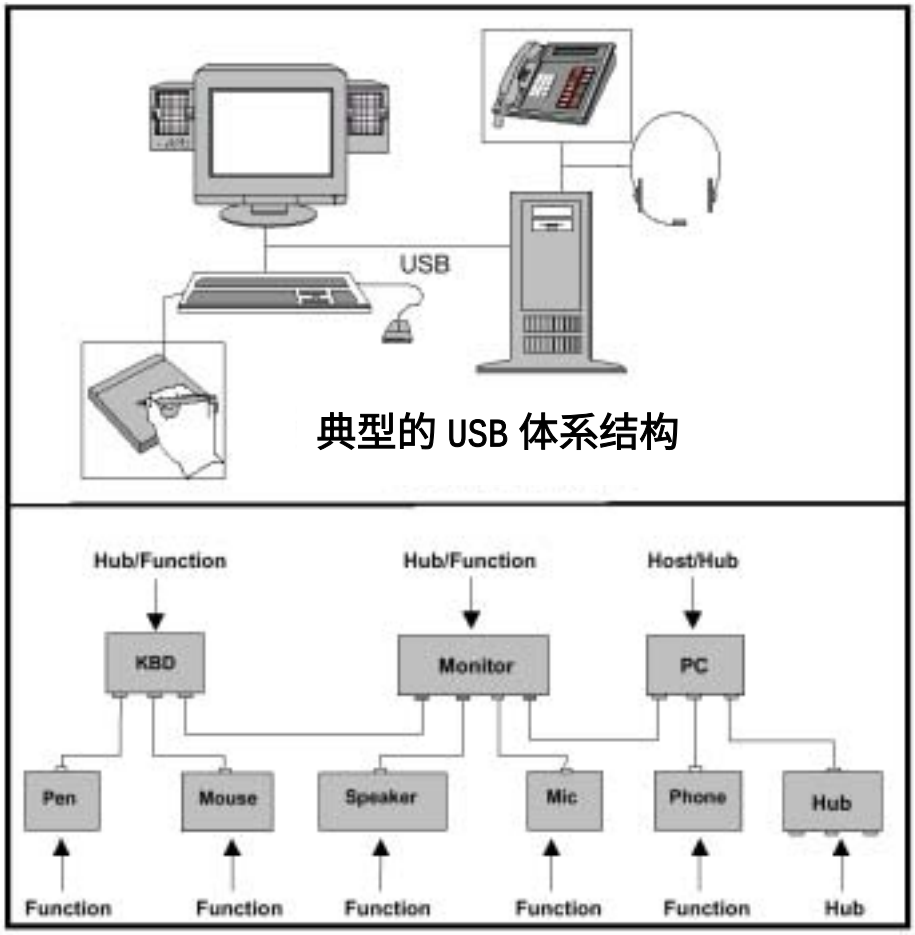


图 4 –4　台式机环境下的 集线器

使用前对其进行设置。设置信息包括 USB 带宽分配，选择设备的设置信息等。

下机列举了一些功能部件：

- 定位设备：如鼠标或光笔；
- 输入设备：如键盘；
- 电信适配器：如 ISDN。

3.9　USB 主机：硬件和软件

USB 的主机通过主机控制器与 USB 设备进行交互。主机功能如下：

- 检测 USB 设备的安装和拆卸
- 管理在主机和 USB 设备之间的控制流；
- 管理在主机和 USB 设备之间的数据流；
- 收集状态和动作信息；
- 提供能量给连接的 USB 设备。

主机上 USB 的系统软件管理 USB 设备和主机上该设备软件之间的相互交互，USB 系统软件与设备软件间有三种相互作用方式：

- 设备编号和设置；
- 同步数据传输；
- 异步数据传输；
- 电源管理

• 设备和总线管理信息。

只要可能，USB 系统软件就会使用目前的主机软件接口来管理上述几种方式。

3.10 体系结构的扩充

USB 的体系结构包含主机控制驱动器和 USB 驱动器之间的接口的扩展，使多个主机控制器和主机控制驱动器的使用成为可能。

02 术语和缩略词

ACK:确认信号
Active Device:正在使用的设备
Asynchronous Data:异步数据
Asynchronous RA:异步自适应速率
Asynchronous SRC: 异步抽样转换率
Audio Device:音频设备
AWG#(American Wire Gauge):美国电线标准
Babble:帧传输中的总线动作
Bandwidth:带宽
Big Endian:
Bit:比特
Bit Stuffing:数据填充,以使 PLL 可以提取时钟信号
b/s:每秒多少比特
B/s:每秒多少字节
Buffer:缓冲区
Bulk Transfer:批量传送
Bus Enumeration:总线标号
Byte: 字节
Capabilities:能力
Characteristics:特征
Client:客户
Configuring Software:配置软件
Control Endpoint:控制端口
Control Pipe:控制通道
Control Transfer:控制传送
CTI:计算机电信组织
Cyclic Redundancy Check(CRC):循环冗余校验
Default Address:缺省地址
Default Pipe:缺省通道
Device:设备、器件
Device Address：设备地址
Device Endpoint：设备端口
Device Resource：设备资源
Device Software：设备软件
Downstream：下行
Driver：驱动
DWORD：双字
Dynamic Insertion and Removal：动态插入与拆除
Electrically Erasable Programmable Read Only Memory EEPROM：电擦写可编程只读存储器
End User：终端用户
Endpoint：端口

Endpoint Address：端口地址
Endpoint Direction：端口指向
Endpoint Number：端口号
EOF：帧结束
EOP：包结束
External Port：外设端口
False EOP：错误的包结束标志
Frame：帧
Frame Pattern：帧结构
Full-duplex：全双工
Function：功能、功能部件
Handshake Packet：握手包
Host：主机
Host Controller：主机控制器
Host Controller Driver(HCD)：主机控制驱动
Host Resourses：主机资源
Hub：集线器
Hub Tier：Hub 层
Interrupt Request(IRQ)：中断请求
Interrupt Transfer：中断传送
I/O Request Packet（IRP)： 输出/输入请求包
Isochronous Data：同步数据
Isochronous Device：同步设备
Isochronous Sink Endpoint：同步接收端
Isochronous Sourse Endpoint：同步源端
Isochronous Transfer：同步传送
Jiffer：抖动
kb/s：传送速率每秒几千比特
kB/s：传送速率每秒几千字节
Little Endian：
LOA：有始无终的总线传输
LSb：最低比特
LSB：最低字节
Mb/s：传送速率每秒几兆比特
MB/s：传送速率每秒几兆比特
Message Pipe：消息通道
MSb：最高比特
MSB：最高字节
NAK：不确认
Non Return to Zero Invert(NRZI)：非归零翻转码
Object：对象
Packet：数据包
Packet Buffer：数据包缓冲区
Packet ID(PID)：数据包标示位

Phase：时项、相位
Phase Locked Loop(PLL)：锁相环
Physical Device：物理部件
Pipe：通道
Polling：查询
Port：口、端口
Power On Reset(POR)：电源复位
Programmable Data Rate：可编程数据速率
Protocol：协议
Rate Adaption（RA)：自适应速率
Request：请求、申请
Retire：取消、终止
Root Hub：根集线器、主机 Hub
Root Port：根集线器的下游端口
Sample：取样、抽样
Sample Rate(Fs)：抽样速率
Sample Rate Conversion(SRC)：抽样转换率
Service：服务
Sevice Interval：服务间隙
Service Jitter：服务质量的抖动参数
Sevice Rate：指定端口每单位时间的服务数目
SOP：包开始
Stage：控制传输的某个阶段
Start-of-Frame(SOF)：帧开始
Stream Pipe：流通道
Synchronization Type：同步类型
Synchronous RA：同步的 RA
Synchronous SRC：同步的 SRC
Sysem Programming Interface(SPI)：系统可编程接口
Terminaton Time Division Multiplexing(TDM)：时分复用
Timeout：超时
Token Packet：标志包
Transaction：处理事务
Transfer：传送
Transfer Type：传送类型
Turn-around Time：USB 传输中包与包之间的间隔时间，以防止传输冲突
Universal Serial Bus Driver(USBD)：USB 驱动器
Univeral Serial Bus Resources：USB 提供的资源
Upstream：上行
Virtual Device：虚拟设备
Word：字（16 位）

第四章 USB 数据流模型

本章介绍了数据如何在 USB 中传送，将涉及到系统中关于信号的发送和协议定义的一层。对于 USB 系统中这一层中各个定义的详细情况可参见第六章和第七章。本章中介绍的数据传送格式，将在第八章到第十一章中逐步扩充。所有的实现者必须阅读此章，以便了解 USB 中一些非常核心的概念。

4.1 实现者的视图

USB 提供了在一台主机和若干台附属的 USB 设备之间的通信功能，从终端用户的角度看到的 USB 系统，可简单地用图 4.1 表示：

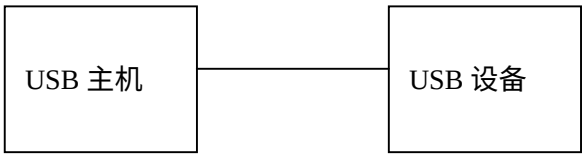


图 4—1 USB 主机/设备的简单模型

但在实际的实现上，具体的系统要比这复杂，不同层次的实现者对 USB 的有不同要求，这使得我们必须从不同的层次观察 USB 系统。USB 系统提出了一些重要的概念和情况来支持现代个人计算机所提出的可靠性要求，所以 USB 的分层理解是必须的。它能使不同层次的实现者只关心 USB 相关层次的特性功能细节，而不必掌握从硬件结构到软件系统的所有细节。USB 的这种层次结构如图 4-2 所示，

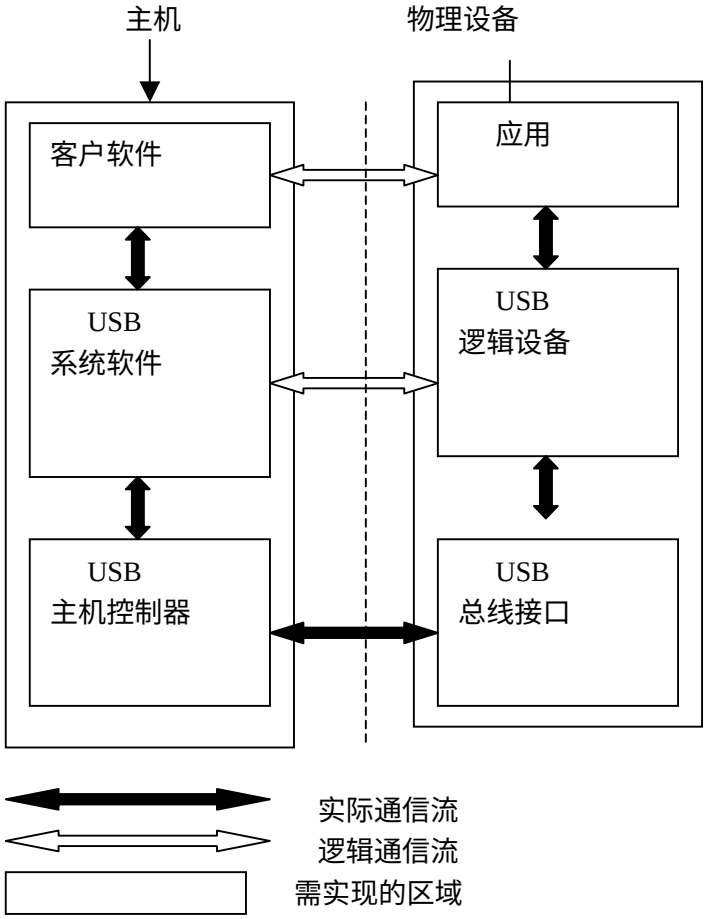


图 4—2 USB 需实现的区域

各层次的具体细节将在以后逐步介绍。特别地，有四个层次的实现是较为集中的。

• USB 物理设备(USB Physical Device)：USB 上的一种硬件，可运行一些用户程序。

• 客户软件(client software)：为一个特定的 USB 设备而在主机上运行的软件。这种软件由 USB 设备的提供者提供，或由操作系统提供。

• USB 系统软件(USB system software)：此软件用于在特定的操作系统中支持 USB，它由
操作系统提供。与具体的 USB 设备无关，也独立于客户软件。

•USB 主机控制器(USB Host Controller):总线在主机方面的接口，是软件和硬件的总和。用于支持 USB 设备通过 USB 连到主机上。

这四个 USB 系统的组成部分在功能上存在相互重叠的部分。为了支持主机与客户之间的坚
固可靠的通信，还需要在后面对这些部分进行细节性描述。

如图 4-2 所示，一台主机与一个 USB 设备间的连接是由许多层上的连接组成。USB 总线接口层提供了在主机和设备之间的物理连接、发送连接、数据包连接。USB 设备层对 USB 系统软件是可见的，系统软件基于它所见的设备层来完成对设备的一般的 USB 操作。应用层可以通过与之相配合的客户软件向主机提供一些额外的功能。USB 设备层和应用层的通信是逻辑上的，对应于这些逻辑通信的实际物理通信由 USB 总线接口层来完成。

关于 USB 的物理通信在第 5、6 章中描述，而相关的逻辑通信在第 8、9 章中介绍。本章描述一些核心概念，USB 系统的实现者必须先掌握它们，然后在往后几章中阅读更加详细的部分。

为了描述和管理 USB 通信，以下概念是很重要的：

• 总线拓朴(Bus Topology)：USB 的基本物理组成、基本逻辑组成，以及各组成部分之间的相互关系。这将在 4.2 节中描述。

• 通信流模型(communication Flow Models):描述主机与设备如何通过 USB 通信，以及通信所用的四种通信类型。这将在 4.3 到 4.8 的各节中介绍。

• 总线访问管理(BUS Access)：主机面对大量的 USB 设备的各种通信要求，如何控制、协
调总线的访问。

• 关于同步传送的考虑：4.10 节中将介绍。对要求同步传送的设备提供一些特性。非同步传送设备的实现者不必阅读此节。

4.2 总线拓朴

总线拓朴结构包括四个重要的组成部分。

• 主机和设备：USB 系统的基础组成部分。

• 物理拓朴结构：描述 USB 系统中的各组成部分是如何连接起来的。

• 逻辑拓朴结构：描述 USB 系统中各种组成部分的地位和作用，以及描述从主机和设备的角度观察到的 USB 系统。

• 客户软件层与应用层的关系：描述从客户软件层看到的应用层的情况，以及从应用层看到的客户软件层的情况。

4.2.1 USB 主机

主机的逻辑结构如图 4-3，包括

• USB 主机控制器（USB Host Controller）

• USB 系统软件集合：USB 驱动程序，主机控制器的驱动程序，主机软件

• 客户软件

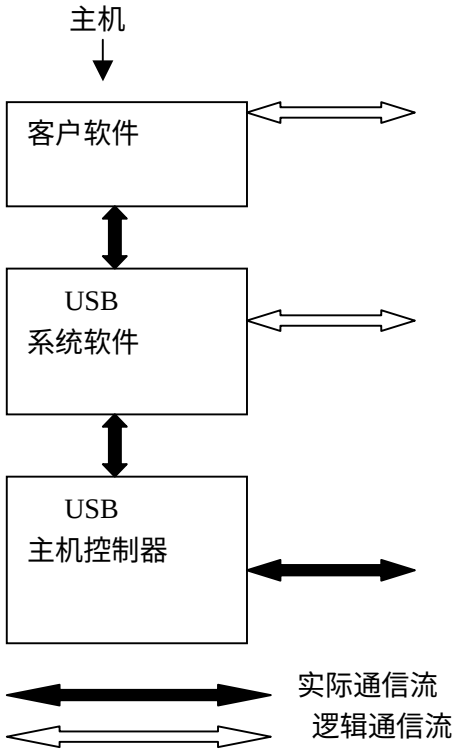


图 4—3 主机的组成

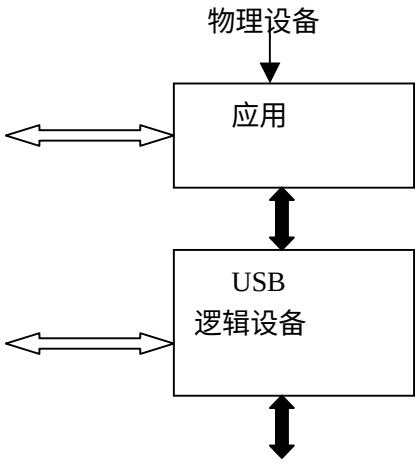
USB 主机在 USB 系统中是一个起协调作用的实体，它不仅占有特殊的物理位置，而且对于 USB 以及连到 USB 上的设备来说，还负有特殊责任。主机控制所有的对 USB 的访问。一个 USB 设备想要访问总线必须由主机给予它使用权。主机还负责监督 USB 的拓朴结构。

关于主机和它的任务的更详细、更彻底的描述，请见第 9 章。

4.2.2 USB 设备

一个 USB 设备的逻辑结构如图 4.4 所示，包括

- USB 总线接口
- USB 逻辑设备
- 应用层



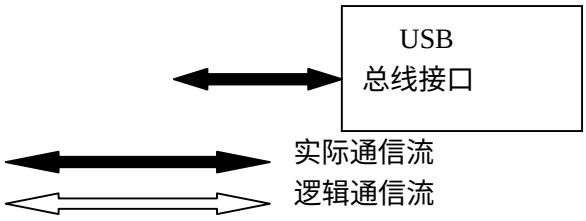


图 4-4 物理设备组成

USB 设备用于向主机提供一些额外的功能。USB 设备提供的功能是多种多样的，但面向主机的接口却是一致的。所以，对于所有这些设备，主机可以用同样的方式来管理它们与 USB 有关的部分。

为了帮助主机辨认及确定 USB 设备，这些设备本身需要提供用于确认的信息。在某一些方面的信息，所有设备都是一样的；而另一些方面的信息，由这些设备具体的功能决定。信息的具体格式是不定的，由设备所处的设备级决定。

对 USB 设备更完备的描述，见第 8 章。

4．2．3 总线的物理拓朴结构

USB 系统中的设备与主机的连接方式采用的是星形连接，如图 4-5。

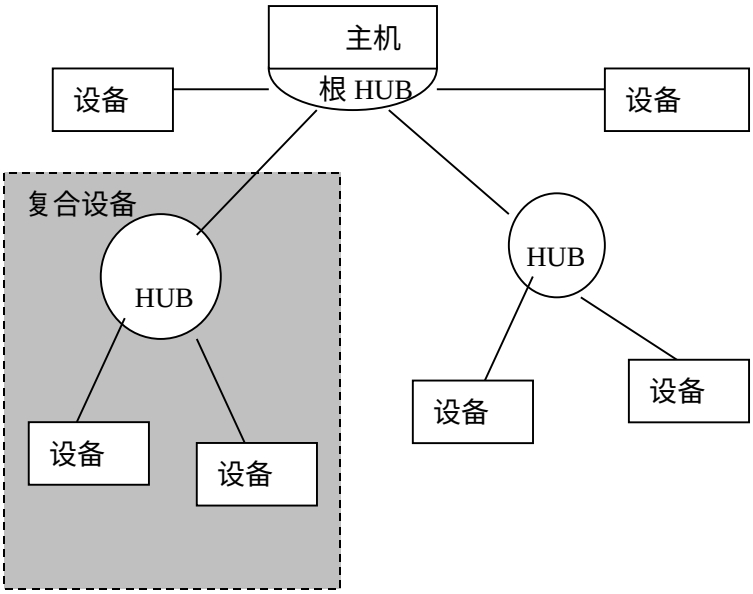


图 4—5 USB 物理总线的拓扑

图中的 Hub 是一类特殊的 USB 设备，它是一组 USB 的连接点，主机中有一个被嵌入的 Hub 叫根 Hub(root Hub)。主机通过根 Hub 提供若干个连接点。为了防止环状连接，采用星形连接来体现层次性，如图 4-5。这种连接的形状很像一棵树。

用于提供具体功能的设备叫应用设备。许多不同功能的设备放在一起被看作一个整体，叫包。例如，键盘和轨迹球可以被视作一个整体，在它的内部，提供具体功能的设备被永久地

接到 Hub 上，而这个 Hub 被接到 USB 上。所有这些设备及这个 Hub 被看作一个复合设备，而这个 Hub 又被看作这个复合设备的内部 Hub。在主机看来，这个复合设备和一个带着若干设备的单独 Hub 是一样的。图中也标出了一个复合设备。

4.2.4 总线逻辑拓朴结构

在物理结构上，设备通过 Hub 连到主机上。但在逻辑上，主机是直接与各个逻辑设备通信的，就好像它们是直接被连到主机上一样。这个逻辑关系如图 4-6 所示。与之对应的物

理结构就是图 4-5 中的结构。Hub 也是逻辑设备，但在图 4-6 中，为了简化起见，未被画出，虽然 USB 系统中的工作都是从逻辑角度来看待的，但主机必须对物理结构有个了解。例如，在处理 Hub 被移去的情况时，当一个 Hub 被移出，通过它与主机相连的设备也应一起被移去，这是由其物理结构决定的。关于 Hub 的更详细的讨论在第 10 章。

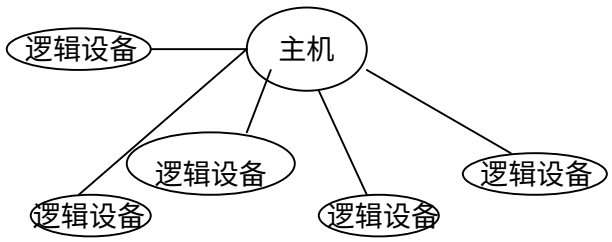


图 4—6 USB 逻辑总线的拓扑

4.2.5 客户软件层与应用层的关系

USB 系统的物理上、逻辑上的拓朴结构反映了总线的共享性。操纵 USB 应用设备的客户软件只关心设备上与它相关的接口，客户软件必须通过 USB 软件编程接口来操纵应用设备。这与另一些总线如 PCL，ELSA，PCMUA 等不同，这些总线是直接访问内存或 I/O 的。在运行中，客户软件必须独立于 USB 上的其它设备。这样，设备和客户软件的设计者就可以只关心该设备与主机硬件的相互作用和主机软件的相互作用的细节问题。图 4-7 说明了在图 4-6 的逻辑结构下，一个设备设计者看到的客户软件与相应应用的关系的视图。

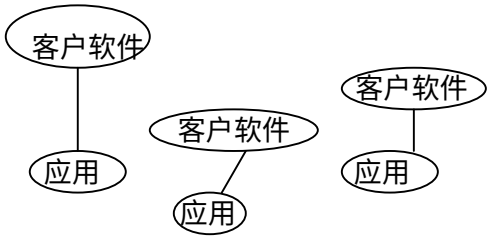


图 4-7 客户软件和应用间的关系

4.3 USB 通信流

USB 是为主机软件和它的 USB 应用设备间的通信服务的，对客户与应用间不同的交互，USB 设备对数据流有不同的要求。USB 为此提供了更好的 overall 总线使用，它允许各种不同的数据流相互独立地进入一个 USB 设备。每种通信流都采取了某种总线访问方法来完成主机上的软件与设备之间的通信。每个通信都在设备上的某个端点结束。不同设备的不同端点用于区分不同的通信流。

图 4-8 是图 4-2 的扩充，它更详尽地描述了 USB 系统，支持了逻辑设备层和应用层间的通信。实际的通信流要经过好几个接口边界，从第 5 章到第 7 章，刻画了机械上、电气上以及协议上的 USB 接口的定义。第 8 章刻划了 USB 设备的编程接口。通过此接口，可从主机侧对 USB 设备进行控制，第 9 章介绍了两个主机侧的通信接口：

• 主机控制器的驱动程序(HCD)：它位于 USB 主机控制器与 USB 系统软件之间。主机控制器可以有一系列不同的实现，而系统软件独立于任何一个具体实现。一个驱动程序可以支持不同的控制器，而不必特别了解这个具体的控制器。一个 USB 控制器的实现者必须提供一个支持它自己的控制器的主机控制器驱动器（HCD）实现。

• USB 驱动程序(USBD)：USB 系统软件与客户软件之间的接口，提供给客户软件一些方便的使用 USB 设备的功能。

一个 USB 逻辑设备对 USB 系统来说就是一个端点集合。端点可以根据它们实现的接口来分类。USB 系统软件通过一个缺省的控制通道来管理设备。而客户软件用通道束管理接口。通道束的一端为端点，一端为缓冲区。客户软件要求通信数据在主机上的一个缓冲和 USB 设备上
的一个端点之间进行。主机控制器或 USB 设备(取决于数据传送方向)将数据打包后在 USB 上传。由主机控制器(HC)协调何时用总线访问在 USB 上传递数据。

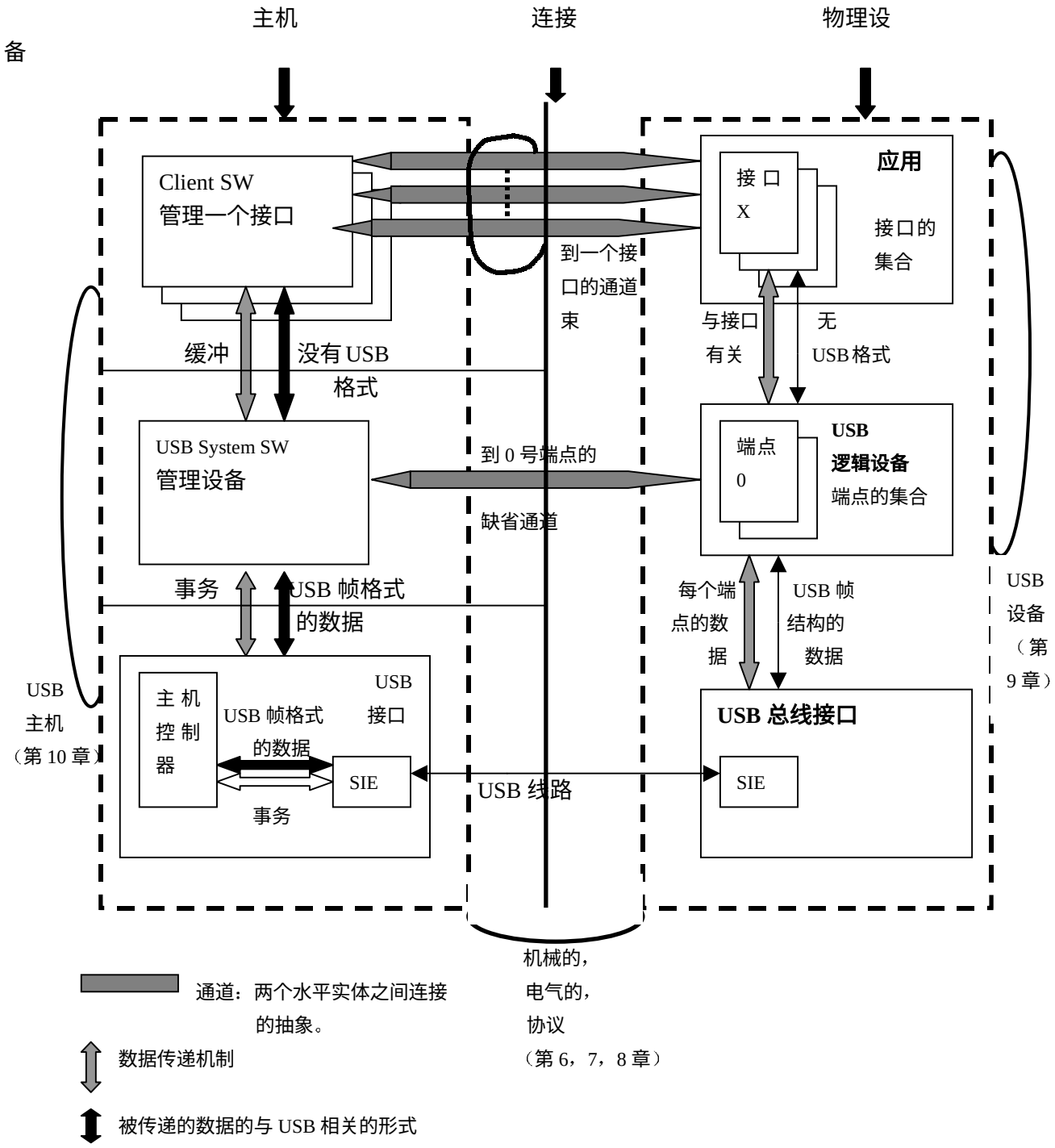
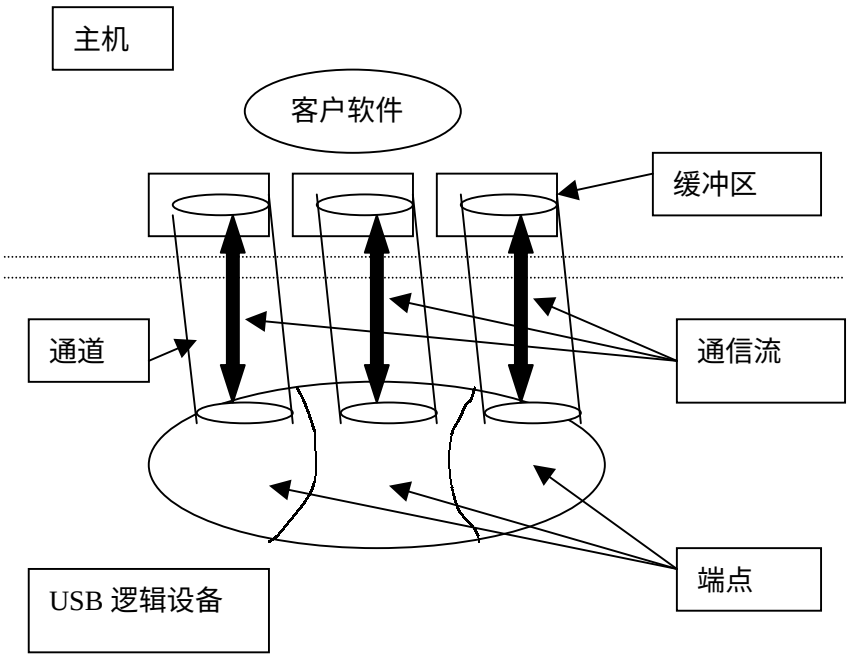


图 4-8 USB 主机/设备的细节图

图 4-9 说明了数据如何在主机侧中的内存缓冲和设备中的端点中传送。在后面，将逐步介
绍端点、通道和通信流。

主机上的软件通过一系列的通信流与逻辑设备进行通信。这一系列的通信流是由 USB 设备
的软件和硬件设计者选择的，使设备能传送由 USB 提供的字符。



界面

图 4-9 USB 通信流

4.3.1 设备端点

一个端点是一个可唯一识别的 USB 设备的 Portion，它是主机与设备间通信流的一个结束点。一系列相互独立的端点在一起构成了 USB 逻辑设备。每个逻辑设备有一个唯一的地址，这个地址是在设备连上主机时，由主机分配的，而设备中的每个端点在设备内部有唯一的端点号。这个端点号是在设备设计时被给定的。每个端点都是一个简单的连接点，或者支持数据流进设备，或者支持其流出设备，两者不可得兼。

一个端点的特性决定了它与客户软件进行的传送的类型。一个端点有以下特性：

- 端点的总线访问频率要求
- 端点的总线延迟要求
- 端点的带宽要求
- 端点的端点号
- 对错误处理的要求
- 端点能接收或发送的包的最大长度
- 端点的传送类型(详见 4.4 节)
- 端点与主机的数据传送方向

端点号不为 0 的端点在被设置前处于未知状态，是不能被主机访问的。

4.3.1 对 0 号端点的要求

所有 USB 设备都需要实现一个缺省的控制方法。这种方法将端点 0 作为输入端点，同时也将端点 0 作为输出端点。USB 系统用这个缺省方法初始化及一般地使用逻辑设备(即设置此设备)。缺省控制通道（见 4.3.2 节）支持了对控制的传送(控制传送将在 4.5 中定义)，一旦设备接上，并加电，且又收到一个总线复位命令，端点 0 就是可访问的了。

4.3.1.2 对非 0 号端点的要求

设备可以有除 0 以外的其它端点，这取决于这些设备的实现。低速设备在 0 号输入及输出端点外，只能有 2 个额外的可选端点。而高速设备可具有的额外端点数仅受限于协议的定义(协议中规定，最多 15 个额外的输入端点和最多 15 个额外的输出端点)。

除缺省控制通道的缺省端点外，其它端点只有在设备被设置后才可使用，对设备的设置是设备设置过程(见第 8 章)的一部分。

4.3.2 通道

一个 USB 通道是设备上的一个端点和主机上软件之间的联系。体现了主机上缓存和端点间
传送数据的能力。

有两不同的且互斥的通道通信格式。

- 流(Stream)：指不具有 USB 定义的格式的数据流。
- 消息(Message)：指具有某种 USB 定义的格式的数据流。

USB 不解释在通道中传送的数据的内容。消息通道要求数据组织成 USB 定义的格式，但它的内容，USB 是不管的。

特别地，有下列概念与通道相关：

- 对 USB 总线访问的申请(claim)，带宽的使用情况
- 传送类型
- 与通道相连的端点的特性，例如：端点的数据传送方向，最大数据净负荷区的长度。

数据净负荷是指在总线处理事务（transaction）中，数据包中数据区的数据(总线处理事务见第 7 章)。由两个 0 号端点组成的通道叫缺省控制通道。一旦设备加电并复位后，此通道即可使用。其它通道只在设备被设置后才存在。USB 系统软件在决定设备身份、设置要求和设置设备时使用缺省控制通道。当设备被设置后，这个设备的特定软件还可使用该通道。USB 系统软件保留缺省控制通道的拥有权，协调其它客户软件对通道的使用。

一个客户软件一般都通过 I/O 请求包(IRP)来要求数据传送。然后，或者等待，或者当传送完成后被通知。IRP 的细节是由操作系统来指定的。客户软件提出与设备上的端点建立某个方向的数据传送的请求，IRP 就可简单地理解为这个请求。一个客户软件可以要求一个通道回送所有的 IRP。当关于 IRP 的总线传送结束时，无论它是成功地完成，还是出现错误，客户软件都将获得通知说 IRP 完成了。

如果通道上没有正在传送的数据，也没有数据想使用此通道，这个通道就处于闲置状态。主机控制器对它不采取任何动作，也就是说，这个通道的端点会发现没有任何的总线动作是冲它而来的。只有当有数据在通道上时，该通道才能发现总线对它的动作。

如果一个非同步通道遇到一个迫使它给主机发 STALL 的情况(参见第 7 章)，或者在任一个 IRP 中发现 3 个总线错误。这个 IRP 将被中止。其它所有突出的 IRP 也一同被中止。通道不再接收任何 IRP，直到客户软件从这个情况中恢复过来(恢复的方式取决于软件的实现)，而且承认这个中止或出现的错误，并发一个 USBD Call 来表明它已承认。一个合适的状态信息将通知客户软件 IRP 的结果———出错或中止。同步通道的运作在 4.6 中介绍。

一个 IRP 可能会需要多个数据净荷区来传递数据。这些数据区除最后一个外，都具有数据净荷区的最大长度，最后一个数据区包含了这个 IRP 中剩下的数据。(可参见关于传送类

型的介绍，以获得更详细的了解)。对这样的一个 IRP，短包(也就是说未达到最大长度的数据区)在数据输入时无法填完 IRP 数据缓冲区。这可能会有二种不同解释，它依赖于客户软件的情况：

• 如果该客户软件可以接受变长的 IRP，那么，IRP 数据缓冲区未被填满，可以看作一个分限，说明一个 IRP 已成功结束，主机控制器可以准备接收下一个 IRP 了。

• 如果该客户软件只收定长的 IRP。那么，我们认为发生了一个错误，这 IRP 将被中止，通道也会被阻塞，通道上的数据都中止。

因为对这两种情况，主机控制器会有不同的反应，而且采取何种措施不由控制器决定，所以对每个 IRP 都必须说明客户软件的具体要求。

通道的端点可以用 NAK 信号来通知主机自己正忙，NAK 不能作为向主机反还 IRP 的中止条件。在一个给定的 IRP 处理过程中，可以遇到任意多个 NAK，NAK 不构成错误。

4.3.2.1 流通道

流通道中的数据是流的形式，也就是该数据的内容不具有 USB 要求的结构。数据从流通道一端流进的顺序与它们从流通道另一端流出时的顺序是一样的，流通道中的通信流总是单方向的。

对于在流通道中传送的数据，USB 认为它来自同一个客户。USB 系统软件不能够提供使用同一流通道的多个客户的同步控制。在流通道中传送的数据遵循先进先出原则。

流管流只能连到一个固定号码的端点上，或者流进，或者流出。(这个号码是由协议层决定的)。而具有这个号码的另一个方向的端点可以被分配给其它流通道。

流通道支持同步传送，中断传送和批传送，这些在稍后的章节会进一步解释。

4.3.2.2 消息通道

消息通道与端点的关系同流通道与端点的关系是不同的。首先，主机向 USB 设备发出一个请求；接着，就是数据的传送；最后，是一个状态阶段。为了能够容纳请求/数据/状态的变化，消息通道要求数据有一个格式，此格式保证了命令能够被可靠地传送和确认。

消息通道允许双方向的信息流，虽然大多数的通信流是单方向的。特别地，缺省控制通道也

是一个消息通道。

USB 系统软件不会让多个请求同时要求同一个消息通道。一个设备的每个消息通道在一个时间段内，只能为一个消息请求服务，多个客户软件可以通过缺省控制通道发出它们的请求，但这些请求到达设备的次序是按先进先出的原则的。设备可以在数据传送阶段和状态阶段控制信息流，这取决于这些设备与主机交互的能力(参见第 7 章)。正常情况下，在上一个消息未被处理完之前，是不能向消息通道发下一个消息的。但在有错误发生的情况下，主机会取消这次消息传送，并且不等设备将已收的数据处理完，就开始下一次的消息传送。在操作通道的软件看来，一个 IRP 中的错误，使这个 IRP 被取消，并且所有正排队等待的 IRP 一同也被取消。申请这个 IRP 的客户被通知 IRP 结束，且有出错提示。

消息通道后有两个相同号码的端点，一个用于输入，一个用于输出。两个号码必须相同。

消息通道支持控制传送，这将在 4.5 中进行介绍。

4.4 传送类型

USB 通过通道在主机缓冲区与设备端点间传送数据。在消息通道中传递的数据具有 USB 定义的格式，它的数据净荷区中包含的数据允许具有设备指定的格式。USB 要求任何在通道上传送的数据均被打包，数据的解释工作由客户软件和应用层软件负责。USB 提供了多种数据格式，使之尽可能满足客户软件和应用软件的要求。一个 IRP 需要一个或多个总线处理事务来完成。

每个传送类型在以下的几个传送特征上会有不同：

• USB 规定的数据格式
• 信息流的方向
• 数据净荷区的长度限制
• 总线访问的限制
• 延时的限制
• 出错处理

USB 设备的设计者可以决定设备上每个端点的能力。一旦为这个端点建立了一个通道，这个通道的绝大多数传送特征也就固定下来了，一直到这个通道被取消为止。也有部分传送特征可以改变，对这样的特征，将会在介绍每个传送类型时作出说明。

USB 定义了 4 种传送类型：

• 控制传送：可靠的、非周期性的、由主机软件发起的请求或者回应的传送，通常用于命令事务和状态事务。

• 同步传送：在主机与设备之间的周期性的、连续的通信，一般用于传送与时间相关的信息。这种类型保留了将时间概念包含于数据中的能力。但这并不意味着，传送这样数据的时间总是很重要的，即传送并不一定很紧急。

• 中断传送：小规模数据的、低速的、固定延迟的传送。

• 批传送：非周期性的，大包的可靠的传送。典型地用于传送那些可以利用任何带宽的数据，而且这些数据当没有可用带宽时，可以容忍等待。

这些传送类型将在后面的四个大节中进行讨论。IRP 的数据均放在数据包中的数据区被传送，这将在 7.4.3 中介绍。关于与具体传送类型有关的一些协议细节在第 7 章中介绍。

4.5 控制传送

控制传送允许访问一个设备的不同部分。控制传送用于支持在客户软件和它的应用之间的关于设置信息、命令信息、状态信息的传送。控制传送由以下几个事务组成：(1)建立联系，把请求信息从主机传到它的应用设备；(2)零个或多个数据传送事务，按照（1）事务中指明的方向传送数据；(3)状态信息回传。将状态信息从应用设备传到主机。当端点成功地完
成了被要求的操作时，回传的状态信息为“success”。7.2 中将介绍控制传送的细节，例如，什么样的包，什么样的总线事务和总线事务的顺序。而第 8 章将介绍 USB 定义的 USB 命令字。

USB 设备必须实现缺省控制通道，并将它实现成一个消息通道。这个通道由 USB 系统软件使用。USB 设备的确认信息、状态信息以及控制信息由该通道传送。如果需要的话，一个应用设备可以为端点实现额外的控制通道。

USB 设备框架(见第 8 章)定义了标准的，设备级的或由销售商提供的请求，这些请求可操作设备的状态。USB 设备框架又定义了一些描述器（descriptor），用于存放 USB 设备的各种信息。控制机制提供访问设备描述器和请求操作设备的机制。

控制传送只能通过消息通道进行。所以，使用控制传送的数据必须具有 USB 定义的数据格式(见 4.5.1 节)。

应用层和相应的客户软件不能为控制传送指定总线访问频率和带宽。这由 USB 系统软件从全局优化角度加以决定。USB 系统软件会限制设备要求的访问频率和带宽，这些限制在 4.5.3 和 4.5.4 中介绍。

4.5.1 控制传送类型的数据格式

Setup 包的数据格式属于一个命令集，这个集合能保证主机和设备之间正常通信。这个格式也允许一些销售商对设备命令的扩展。Setup 包后的数据传送也具有 USB 定义的格式，除非这个数据是销售商提供的信息。回传的状态信息仍然具有 USB 定义的格式。7.5.8 节和第

8 章将介绍控制传送的 Setup 定义和数据定义。

4.5.2 控制传送的方向

控制传送使用的是消息通道上的双向信息流。所以，一旦一个控制通道被确认之后，这个通道就使用了具有某个端点号的两个端点，一个输入，一个输出。

4.5.3 控制传送包的大小的限制

控制传送的端点决定了它所能接收或发送的最大数据净负荷区长度。USB 为高速设备定义的最大数据净负荷区长度为 8、16、32 或 64 字节，低速设备的数据净负荷区的长度只能是 8 字节。Setup 后的所有数据包都要遵守这个规定，这个规定是针对这些数据包中的数据净负荷区的，不包括包中的协议要求的额外信息，Setup 包实际上也是 8 字节。控制通道(包括缺省控制通道)总是使用 w Max Packet Size 的值。

端点在自己的设置信息中报告自己允许的最大净负荷区长度。USB 不要求数据净负荷区必须达到最大长度，当长度不够时，不必填充到最大长度。

主机控制器对高速设备的控制通道端点支持 8、16、32、64 字节的最大长度，对低速设备支持 8 字节的长度。它不能支持更大的或更小的其它长度。

对于缺省控制通道的最大数据区长度，USB 系统软件要从设备描述器的头 8 个字节中读出，设备将这 8 个字节放在一个包中发出，其中的七个字包含了缺省通道的 wMaxPacketSize。对其它的控制端点来说，USB 系统软件在它们被设置后，获得此长度，然后 USB 系统软件就会保证数据净负荷区不会超长。另外，主机总是认为数据净负荷区的最大长度至少为 8。

端点所传的数据净负荷区长度必须小于或等于其 wMaxPacketSize(参见第 8 章)，当一个数据区不能容纳所传数据时，就分几个区来传。除最后一个区外，其它区都应达到最大长度。最后一区包含最后剩下的数据。

当端点做了以下两件事时，控制传送的数据阶段可被认为结束：

- 已传了由 Setup 阶段指定的数据量。
- 传了一个数据包，它的长度为 0 或它的数据区长度小于最大长度。

数据阶段结束后，主机控制器进入状态阶段，而不是开始另一个数据传诵。如果它不这样做，端点会认为通道脱线而中止通道(通道脱线见 4.3.2)。如果主机在状态阶段时，主机收到一个大于最大长度的数据区，那么请求这次传送的 IRP 将被中止。

当数据全部传完，主机与端点之间的控制传送的数据阶段结束。如果其间，端点收到了超过最大长度的数据区，它将中止通道。

4.5.4 控制传送的总线访问的限制

无论低速设备还是高速设备都可以使用控制通道。

端点没法指明控制通道对总线访问频率的要求。USB 权衡所有控制通道的总线访问频率和正等待的 IRP，从全局优化，提供一个“最佳”传送方案。

USB 要求数据帧中的一部分被留给控制传送使用。

- 如果被引发的控制传送(引发方式由实现决定)只用了数据帧的不到 10%的时间，则剩余的时间留给批传送(参见 4.8 节)。
- 如果一个控制传送被引发又被中止，则它的中止可在本次的帧内，也可在以后的帧内。也就是说，引发和中止不必在同一个帧内。
- 如果留给控制传送的时间不够用，但恰好有一些同步和中断传送的帧时间未用，则主机控制器利用这些时间进行额外的控制传送。
- 如果对可用的帧时间有太多的控制传送在等待，那么就对它们进行排序然后传送。
- 如果各个控制传送申请的是不同的端点，主机控制器根据公平访问原则决定它们的访问顺序。公平访问原则的具体内容决定于主机控制器的实现。

• 如果一个控制传送事务频繁地被中止，不能认为给它的总线访问时间是不公平的。

这些要求使得控制传送一般可以在总线上进行规则地、最优化地传送。

对某个端点的控制传送的速率是可以变化的，USB 系统软件控制这些离散的变化。端点和其客户软件不能想当然的认为其有一个固定的传送速率，端点可能发现在一帧内有零个或若干个传送。一个端点和它相应的客户软件可占用的总线时间会因为其它设备进入或退出系统或者本设备上的其它端点进入或退出系统而改变。

总线频率和帧定时决定于一个帧内可传送的控制传送的最大个数。在任一个 USB 系统内，一个帧内的 8 字节高速数据区须少于 29 个，8 字节低速数据区须少于 4 个。表 4-1 是关于不同规格的高速的控制传送的情况，以及在一帧内可能的最大的传送数目。这张表有两个默认的前提，即控制传送有一个数据传送阶段而且这个数据传送阶段有一个长度为 0 状态阶段，表 4-1 还指出了出现两个数据区都达不到最大长度的情况，表中不包括用于管理的一些额外的位。

表 4-1 高速控制传送限制

	协议开销（46 字节）		**（9 SYNC bytes, 9 PID bytes, 6 Endpoint+CRC bytes, 6 CRC bytes, 8 Setup data bytes, and a 7_byte interpacker delay(EOP, etc.))**			
	数据净荷区	最大带宽（字节/秒）	Frame 带宽/传送	最大传送数	剩 余字节	有用数据字节/Frame
	1	32000	3%	32	23	32
	2	62000	3%	31	43	62
	4	120000	3%	30	30	120
	8	224000	4%	28	16	224
	16	384000	4%	24	36	384
	32	608000	5%	19	37	608
	64	832000	7%	13	83	832
Max		1500000				1500

因为一个帧内只留 10%的时间给非周期性传送，所以当一个系统的总线时间被排满的时候，这个系统内的所有控制传送只能去竞争每个帧内的三个控制传送名额。因为除了客户软件会要求控制传送外，USB 系统要用控制传送来传送设置信息，所以对某个客户和它的应用就不能指望它们的控制传送像它们想的一样进行。主机控制器可以自由地决定如何将某个具体的控制传送在总线上进行，可以在一个帧内，也可以跨几个帧。一个端点可能发现一个控制传送的各个总线处理事务在同一帧内或分在几个不连续的帧内。由于具体实现的不同，主机控制器可能不能提供理论上的每帧的最大控制传送数目。

低速控制传送与高速控制传送都是竞争同样多的可用帧时间。低速控制传送只是要用更多的时间来传送罢了。表 4-2 列出了不同规格的低速包的情况，以及一帧内允许的最大包数。这张表同样没包括进管理用的开销。无论低速与高速，由于一个控制传送都由几个包组成，所以都可能要用几个帧才能完成传送。

表 4-2 低速控制传送限制

	协议开销（46 字节）					
	数据净荷区	最大带宽字节/Frame	Frame 带宽/传送	最大传送数	剩余字节	有用数据字节/Frame
	1	3000	25%	3	46	3
	2	6000	26%	3	43	6

	4	12000	27%	3	37	12
	8	24000	29%	3	25	24
Max		187500				187

4.5.5 控制传送的数据顺序

要进行控制传送，先要由主机向设备发一个总线建立（Setup）信息。它描述了控制访问的类型，设备将执行此控制访问。这个阶段之后，是零个或多个控制数据信息的传送，这是进行访问的具体信息。最后，由状态信息的传送来结束这次控制传送，允许端点将这次控传的状态回送给客户软件。这次控传完成之后，可以进行对这个端点的下一个控传，如4.5.4 节所述，每次控传何时在总线上进行由主机控制器的具体实现决定。

在数据传送阶段和状态信息回传阶段，可能由于设备自身的原因，设备处于“忙”状态。此时端点可设法表明自己正忙(见第 7、8 章)，主机将试着在稍后时间重传一次。

如果在上一个控传结束之前，端点又收到一个总线建立信息，设备将结束现未完成的传送，转而处理新的控传。正常情况下，是不会早发总线建立信息的，不过当上一个控传因错误而被中止后，主机可发下一个控传的总线建立信息。在端点看来，这是在上一个控传结束前过早发出的。

一旦主机遇到一个引起中止的条件或检测到一个错误，端点可以通过接收下一个 Setup 包的 PID 来恢复，也就是说，不一定必须从别的通道进行恢复。对于缺省控制通道，如果端点收不到 Setup 的 PID 时，最终会要求设备复位来清除中止条件或错误条件。

在控传中，USB 提供了强大的错误检测功能和错误恢复和重传功能。传送器和接收器可以保持阶段的同步，既关于他们在控传的哪个阶段这个问题上保持同步。并且以最小的代价恢复。接收器可以识别一个数据重传包或状态信息重传包，因为包中带有数据重传的指示。一个发送器可以通过对方给它发的握手信息确知它发的数据重送包和状态信息包已被成功接收，除了 Setup 包以外，协议可以将一个重送的包与原来的包区分开来，Setup 包可以因为出错而重传，但无法说明此包是重传的，还是原来的。

4.6 同步传送

在非 USB 的环境下，同步传送意味着恒定速率、错误容忍(error-tolerant)的传送。在USB 环境下，要求同步传送能提供以下几点：

- 固定的延迟下，确保对 USB 带宽的访问。
- 只要数据能提供得上，就能保证通道上的恒定数据传送速度。
- 如果由于错误而造成传送失败，并不重传数据。

当 USB 同步传送类型被用来支持同步的源和目的时，使用这个传送类型的软件并不要求是同步的，4.10 中将详细介绍 USB 上的同步数据的处理。

4.6.1 同步传送的数据格式

对于同步传送的通道(同步通道)，USB 并不对数据格式做要求。

4.6.2 同步传送的方向

同步通道是一种流通道，所以是单方向的。在对端点的描述中指明了与它相连的通道的数据流方向。如果设备要同步的双向流的话，只好用两个同步通道，一个流进，一个流出。

4.6.3 同步传送中包的大小的限制

同步通道的端点确定了数据区的最大长度，USB 在设置端点期间，使用这一个信息，看是否可在每帧内为最大长度的数据区留下足够的时间。如果可以，设置端点成功；否则，不成功。

USB 系统软件可为一个控制传送的通道调整最大数据区长度，但无法为同步通道进行如此调整。在确定的 USB 设置下，同步通道要么被支持，要么不被支持。

USB 限制了同步通道的最大数据区长度为 1023 字节，表 4-3 列出了不同规格的同步传送，

以及一帧内可能的最大传送数。表中未包括管理开销的字节。

表 4-3 同步传送限制

	协议开销（9 字节）		**（2 SYNC bytes, 2 PID bytes, 2 Endpoint+CRC bytes,2 CRC bytes, and a 1_byte interpacket delay)**			
	数据净荷区	最大带宽字节/秒	Frame 带宽/传送	最大传送数	剩余字节	有用字节字节/Frame
	1	150000	1%	150	0	150
	2	272000	1%	136	4	272
	4	460000	1%	115	5	460
	8	704000	1%	88	4	704
	16	960000	2%	60	0	960
	32	1152000	3%	36	24	1152
	64	1280000	5%	20	40	1280
	128	1280000	9%	10	130	1280
	256	1280000	18%	5	175	1280
	512	1024000	35%	2	458	1024
	1023	1023000	69%	1	468	1023
Max		1500000				1500

并不是每一次的数据区都要达到最大长度。数据区的长度由发送者(客户软件或应用软件)决定，每次可以不同。USB 可保证主机控制器看到的包有多长，在总线上传的包就有多长。数据的实际长度由发送者决定，可以小于早先协商好的最大长度。总线错误可以使接收者看到的长度比实际长度有了变化。但这些错误可被检测到。具体地讲，或者通过数据上的 CRC 码，或者让接收者预先知道实际应该的长度，以此进行检测。

4.6.4 同步传送的总线方向限制

只有高速设备可以使用同步方式。

USB 设备要求一个帧内不能有超过 90%的时间用于周期性传送(同步传送或中断传送)。

同步通道的端点描述自己的总线访问频率。所有的同步通道一般在一帧内传一个包(也就是说，1ms 一个包)。但总线上的错误或者操作系统对客户软件调度上的延迟会造成一个帧内一个包也没有的情况。此时，设备将一个错误指示信息作为状态信息返回给客户软件。设备可以通过跟踪 SOF（帧开始）信号来测到此类错误。如果两个 SOF 信号间无数据包，则出错。

总线频率和帧定时限制了一个帧内的同步传送的上限，在任何 USB 系统内，最多有 150 个单字节的数据区。但由于实现上的原因，主机控制器可能无法支持到理论上的最大传送数。

4.6.5 同步传送的数据顺序

同步传送不支持因总线错误而进行的重传。接收器可以判断是否发生了一个错误，低级的 USB 协议不允许有握手信号给同步通道的发送者。一般情况下，是可以有握手信号来通知发送者包是否被成功地接收。对于同步传送来说，定时比正确性和重传更重要。考虑到总线的错误率较低，协议就认为传送一般均能成功。同步接收者可以判断自己是否在一个帧内错过了一些数据，而且能知道丢失了多少数据。4.10 节将有关于此的具体介绍。

因为没有用来指示引起中止的条件的握手信号，所以同步传送的端点从不途停止。虽然，错误信息可作为 IRP 的状态来报告，但同步通道不会因此停下。错误即使被查到，主机仍继续处理下一帧的数据。因为同步传送的协议不支持每次事务都进行握手，所以错误检测的功能可以相对弱一些。

4.7 中断传送

中断传送是为这样一类设备设计的，它们只传或收少量数据，而且并不经常进行传送，但它们有一个确定的服务周期，对中断传送有以下要求：

- 通道的最大服务期得到保证。
- 由于错误而引起的重发在下一服务期进行。

4.7.1 中断传送的数据格式

USB 对中断通道上的数据流格式无要求。

4.7.2 中断传送的方向

中断通道是一种流通道，所以是单向的。端点描述信息指明了通道的数据流方向。

4.7.3 中断传送对包的长度的限制

中断通道的端点决定自己能接收和发送的最大数据区长度，高速设备允许最大不超过 64 字节(或更少)的数据区，而低速设备只允许不超过 8 个(或更少)字的数据区，这个数字不包括协议要求的附加信息。USB 并不需求所有的包都到最大长度。如果不到的话，不用加字节填充。

所有的主机控制器都要示支持高速设备的 64 字节数据区和低速设备的 8 字节(或更少)的最大数据区，对超过最大值的数据区则不要求支持。

USB 系统软件设置中断通道的最大数据区长度。在设备设置期间，这一信息将被使用，只有此设置有效，这个数值是不会改变的。在设置有效期间，USB 系统软件根据此数值来看分给这个通道的总线时间是否充分。如果充分，则通道建立，否则不建立。与控制通道不同，USB 系统不为中断通道调整总线时间。所以对给定的 USB 系统，要么支持此通道，要么不支持。实际传送的数据区长度由发送器决定，可以小于最大长度。

端点所发的数据区中的数据长度不能超过端点的 w Max Packet Size 的值。而设备可以通过中断传送来传比此值多的数据。客户软件可以通过中断传送的 IRP 来接收这批数据，这个中断传送要求多个总线处理事务来完成，且要求每个事务后都有 IRP 完成的信号。可以设置一个缓冲区，它的长度为 w Max Packet Size 的整数倍，再加上一个零头。对需要的多个总线事务来说，除最后一个外，前面的事务都传递 w Max Packet Size 长度的包，后一个传剩下的零头。这些总线处理事务都在为通道建立的服务周期内进行。

如果一个中断传送要传的数据不能放在一个数据区中，就分几个区，前几个区都是最大长度，最后一个包含剩下的长度。当出现以下情况时，认为中断传送结束：

- 已传的数据量恰好与期望的数据量同。
- 传了一个有一个数据区的包，此包的长度小于 w Max Packet Size 或传了一个长度为零的包。

如果一个中断传送完成，那么主机控制器结束当前的 IRP，并开始下一个 IRP。如果数据区的长度比预料的长，当前 IRP 中止，并且只有等到出错条件被确认且清除后，才能开始后面的 IRP。

4.7.4 中断传送对总线访问的限制

高速设备和低速设备均可使用中断传送。

USB 要求不能有多于 90%的顺时间用于阶段传送(同步传送或中断传送)。

总线频率和帧的定时限制了一帧内能传的最大中断传送数。对任一 USB 系统来说，高速单字数据区少于 108 个，低速单字节数据区少于 14 个。由于实现上的原因，主机控制器不一定能够支持此理论上的上限。

表 4-4 列出了不同规格的高速中断传送的情况，以及一帧内可能的最大传送数。表 4-5 列的是对低速设备的相关情况。它们均不包括管理开销的字节。

表 4-4 高速中断传送限制

	协议开销（13 字节）		（3 SYNC bytes, 3 PID bytes, 2 Endpoint+CRC bytes, 2 CRC bytes, and a 3_byte interpacket delay)			
	数据净荷区	最大带宽（字节/秒）	Frame 带宽/传送	最大传送数	剩余字节	有用数据字节/Frame
	1	107000	1%	107	2	107
	2	200000	1%	100	0	200
	4	352000	1%	88	4	352
	8	568000	1%	71	9	568
	16	816000	2%	51	21	816
	32	1056000	3%	33	15	1056
	64	1216000	5%	19	37	1216
Max		1500000				1500

中断通道的端点可以指明它要求的总线访问周期。高速设备要求的时间周期可以 1ms 到 255ms，而低速设备从 10ms 到 255ms。在设置期间，USB 系统软件根据它们的要求来决定一个服务周期长度。USB 提供的服务周期长度可能比设备要求的要短些，但不会少于最短的 1ms。客户软件和设备只能够确定两次传送之间的时间长度不会比要求的周期时间长。但如果传送中出现错误，那么周期时间必然要越界。当客户软件有一个中断传送的 IRP 时，端点只是被选中。如果总线轮到此中断传送使用时，没有 IRP 处于待发状态，则端点没有机会在此时间传数据，一旦一个 IRP 出现了，它的数据在下一个轮到它的时间时被发出。

表 4-5 低速中断传送限制

	协议开销（13 字节）					
	数据净荷区	最大带宽	Frame 带宽/传送	最大传送数	剩余字节	有用数据字节/Frame
	1	13000	7%	13	5	13
	2	24000	8%	12	7	24
	4	44000	9%	11	0	44
	8	64000	11%	8	19	64
Max		187500				187

要在 USB 上进行中断传送，必须在每个周期对端口进行访问。主机无法知道何时一个端口准备好了一个中断传送，除非它访问这个端点，并同时请求一个中断传送，等待回答。如果端口无数据需要中断传送，就对其请求回送一个 NAK 信号。如果端口传送数据的会有中断情况发生，一定要用中断传送，以防中断产生时，客户软件误以为 IRP 结束。长度为 0 的数据净负荷区的传送是合法的，而且对某些实现是很有用的。

4.7.5 中断传送的数据顺序

中断传送可以利用 0/1 跳变位（toggle 位）的机制，当成功的进行了一个传送，该位就跳变一次。

主机总是认为设备是遵守完备的握手协议和重发协议(参见第 7 章)。但如果无论传送成功否，设备都在 Data1/Data 0 间跳变 PID，就忽略主机发来的握手信号。但这时，客户软件会丢失一些包。因为有错误发生时，主机控制器会把设备发的下一个包当作上一个包的重发。

一旦在中断通道上检测到一个引起中止的条件，或收到设备发来的 STALL 握手信号，所有正等待的 IRP 都会中止。由软件通过独立的控制通道来消除中止条件。清除后，设备和

主机都复位到 Data 0 的状态。如果总线上出现了一个影响传送的错误，则中断处理事务会停止。

4.8 批传送

为了支持在某些在不确定的时间进行的相当大量的数据通信，于是设计了批传送类型。它可以利用任何可获得的带宽。批传送有以下几点特性：

- 以可获得带宽访问总线。
- 如果总线出现错误，传送失败，可进行重发。
- 可以保证数据必被传送，但不保证传送的带宽和延迟。

只当有可获得的带宽时，批传送才会发生。如果 USB 有较多的空闲带宽，则批传送发生地相对频繁，如果空闲带宽较少，可能有很长时间没有批传送发生。

4.8.1 批传送的数据格式

USB 没有规定批通道上数据流的格式。

4.8.2 批传送的方向

批通道是一种流通道，所以总是单方向的。如果要进行双向传送，必须用两个通道。

4.8.3 批传送对包长度的限制

批传送的端点决定自己可以接收或传送的最大数据净负荷区长度。USB 规定最大的批数据净负荷区的长度为 8、16、32 或 64 字节。这个最大长度是指数据包中数据区的最大长度，不包括协议要求的一些管理信息。

批端点必须支持规定的最大长度中的一个，这个长度将在端点的设置信息中说明。USB 并不要求每个数据净负荷区都达到最大长度，即如果不够长度的话，不必填充至最大长度。

所有主机控制器必须分别支持 8、16、32 或 64 作为最大长度，而对更大或更小的长度可以不必支持。

在设备设置期间，USB 系统软件读取端点的最大数据净负荷区长度，以保证以后传送的数据净负荷区不会超长。

端点传送数据区的包不能超过端点的 w Max Packet Size 的值。如果一个批传送的 IRP 要传送的数据大于一个数据区的最大长度，那么要分几个数据净负荷区来传，除最后一个区外，前几个都达到最大长度。而最后一个包含剩下的数据。如果出现以下情况，则认为批传送结束：

- 已传的数据量恰好等于期望传送的量。
- 传了一个不到 w Max Packet Size 长度的包或传了一个长度为 0 的包。

一旦批传送结束，主机控制器中止当前的 IRP，并开始下一个 IRP。如果收到的一个数据净负荷区超长，则所有在等待此端点的批传送 IRP 都将被中止/取消。

4.8.4 批传送对总线访问的限制

只有高速设备可以使用批传送。

端点无法提出对批通道的总线访问频率的要求。USB 会协调所有批传送和正等待的 IRP 的总线访问请求，以获得在客户软件和应用层之间的“最佳”传送效果。总线上的控制传送的优先级比批传送高。

对于控制传送，有可保证的传送时间，而对批传送，没有。只有当有可用的总线带宽时，批传送才发生。如果有段时间没有被用于其他目的，这段时间将用于批传送。如果正等待的各个批传送是要往不同的端点去的，主机控制器将根据公平访问原则，安排它们的顺序。至于公平访问原则的具体内容，由主机控制器的实现决定。

系统中的所有批传送是竞争同一个可用的总线时间的，所以 USB 系统软件可以改变对某个特定端点进行的批传送所占有的总线时间。所以端点和它的客户软件不能够期望有一个特定的批传送的速度。当有设备被加进或移出 USB 系统或出现对其它设备上端点的请求时，

端点和它的客户软件可获得的总线时间将起一定变化。但客户软件不能主观地认为批传送与控制传送的顺序，有时，批传送会在控制传送之前进行。

总线频率和帧定时限定了在一帧内可进行的最大批处理事务的数量，即一帧内 8 字节数据净负荷区须少于 72 个，表 4-6 列出了不同规格的批处理事务的情况，以及一帧内可能的最大的事务数。数据中不包括管理开销的字节。

表 4-6 批传送限制

	协议开销（13 字节）		**（3 SYNC bytes, 3 PID bytes, 2 Endpoint+CRC bytes, 2 CRC bytes and a 3_byte interpacker delay)**			
	数据净荷区	最大带宽	Frame 带宽/传送	最大传送数	剩余字节	有用数据字节/Frame
	1	107000	1%	107	2	107
	2	200000	1%	100	0	200
	4	352000	1%	88	4	352
	8	568000	1%	71	9	568
	16	816000	2%	51	21	816
	32	1056000	3%	33	15	1056
	64	1216000	5%	19	37	1216
Max		1500000				1500

对于某个批传送，主机控制器可以自由地决定它们的各个事务在某帧或某几个帧中被传送。端点可能在一帧内看到某个批传送的各个事务，或发现它们跨几个不同的帧。由于实现上的原因，主机控制器可能无法支持到理论上的每帧最大事务数。

4.8.5 批传送的数据顺序

批传送利用 toggle 位机制来保证接收器和发送器同步，即使在有错发生的情况下，也是如此。当端点被适当的控制传送设置过后，批传送被初始定位在 DATA0，主机也将从 DATA0 开始第一个批传送。如果传送错误而导致出现一个能引起中止的条件或设备发了一个 STALL 握手信号，所有等待的 IRP 将被取消。软件通过一个独立的控制通道来清除中止条件，恢复之后，主机和设备的数据 toggle 都被定位在 DATA0。

如果出现了一个影响事务的总线错误，批传送将被中止。

4.9 传送的总线访问

要完成主机与 USB 设备间的任何数据传送，必须要使用一定的 USB 带宽。要想支持从同步设备到异步设备的各种传送，必须要能满足它们对传送的不同要求。分配总线带宽给设备的工作叫做传送管理。主机上有几个部分是用于协调 USB 上的信息流的，它们是：客户软件、USB 驱动器（USBD）和主机控制器驱动器（HCD）。实现这些部件必须要了解关于总线访问的一些核心概念：

- 传送管理：用于支持 USB 上信息流的各实体和各对象
- 事务跟踪：一种 USB 机制，跟踪在 USB 系统中的事务
- 总线时间：总线传一个信息包的时间
- 设备/软件缓冲区大小：支持一个事务所需要的空间。
- 总线带宽归还：被分配给其它传送的总线带宽未被使用时，可以重新给控制传送和批传送使用。

前几节主要着重于客户软件如何与它的应用层进行联系和什么是两实体之间的逻辑流。这节介绍主机上的不同部分如何相互协调工作来支持 USB 上的数据传送，对设备实现者来说，这个信息也是有用的，他们可以由此知道当客户请求传送时主机该做什么，以及传送请求是如何被发给设备的。

4.9.1 传送管理

传送管理涉及以下几个为不同目标工作的部分，它们共同工作使数据能在总线上传送：

• 客户软件：通过对 USBD 界面发出调用（calls）和响应调用（calls backs）的 IRP 请求，消费从应用端点来的数据或生产到消费端点去的数据。

• USBD(USB 驱动器)：通过对合适的主机控制器驱动器(HCD)的调用（calls）和响应调用（callbacks），将从设备端点来的或到设备端点去的 IRP 中的数据进行一定转换。一个客户 IRP 可能会需要几个传送来完成。

• 主机控制器驱动器(HCD)：将 IRP 转换成事务或将事务转换成 IRP(按照主机控制器的要求)，并对它们进行组织，以使主机控制器进行操作。HCD 和它的硬件间的互相作用与它的实现有关，不在 USB 说明的范围内。

• 主机控制器：进行事务，为每个事务在总线上通过包传送数据。

图 4-10 说明了当在客户软件和 USB 间有信息流动时，这些部分是如何工作的。它们之间的界面是我们了解的目标。

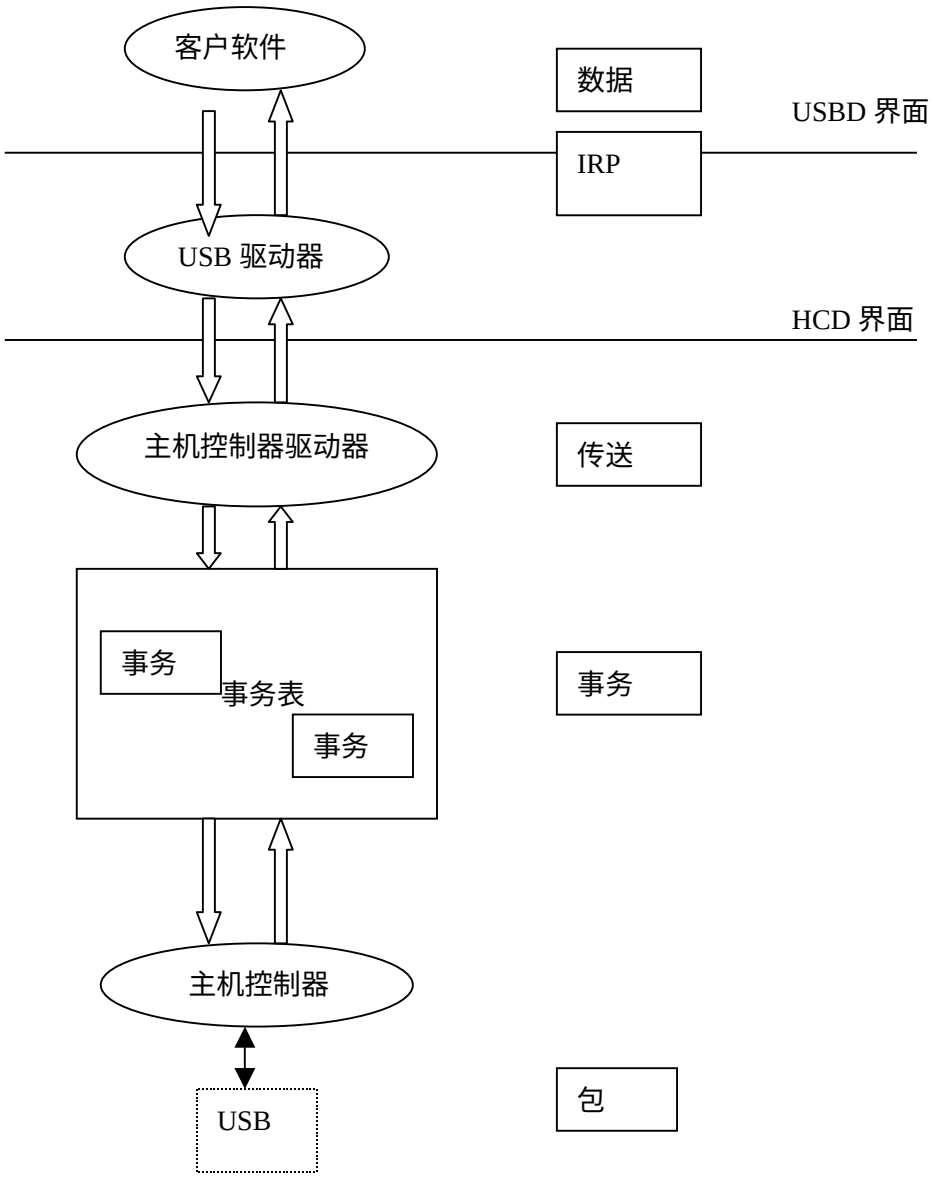


图 4-10 从客户软件到总线的 USB 信息转换

4.9.1.1 客户软件

客户软件决定某个应用应该用何种传送类型。它通过操作系统指定的界面来请求 IRP。客户软件知道它用来操作应用的通道(接口)的存在；客户知道总线访问和带宽的限制，并且遵守它们。客户软件将自己的请求发给 USB 驱动器接口。

有些客户通过操作系统提供的另一些设备级接口来操作 USB 功能，而不直接进行 USBD 的调用（calls）。但总要有一些低级的客户进行直接的 USBD 的调用（calls），以便将 IRP 传给 USBD。所有这些提交的 IRP 必须遵守通道建立时定下的带宽限制。如果一个设备从一个非 USB 环境进入 USB，客户软件将通过主存和 I/O 访问直接操作这个设备的硬件。USB 系统中的最低级的客户软件和 USBD 相互作用，来操纵 USBD 的 USB 功能。

当客户软件要求一个与它的应用层间的数据传送，而且又被满足后，客户软件将收到一个关于 IRP 完成状态的通知。如果传送中有应用层到主机的数据，客户软件收到 IRP 完成的通知后，可在数据缓冲区中取到所要的数据。

USBD 的接口在第 9 章中介绍。

4.9.1.2 USB 驱动器(USBD)

USBD 将在两个主要的时间介入总线访问：

- 在设备接上主机的设置期间
- 在正常传送中

当设备被接上，并被设置时，USBD 将确认设备的设置是否与总线兼容，USBD 从设置软件处获得设置请求，它描述了要进行的设置：端点、传送类型、传送周期、数据长度等。USBD 根据可用带宽的大小以及总线是否与请求的类型兼容来决定接受还是不接受这个设置。如果接收，USBD 就为请求者建立一个相应类型的通道，自然也带有这个传送类型所应有的各种限制。在设备设置期间，不一定必须为周期性的端点作带宽分配。做过的带宽分配也可以放弃，而并不影响设备的设置。

对 USBD 的设置是典型地由操作系统决定的，而且反过来影响操作系统的设置特征。这样做可免除一些多余的接口。

一旦设备被设置，客户软件就可以通过 IRP 来请求与应用端点进行数据传送。

4.9.1.3 主机控制器驱动器(HCD)

HCD 负责跟踪 IRP，并确保 USB 带宽和帧最大时间不被突破。当有 IRP 要求通道时，HCD 将它们加入事务表中，当 IRP 结束，HCD 将把它的完成状态通报给发它的客户软件，如果 IRP 中涉及应用设备向主机的传送，来的数据将被放在客户指定的数据缓存中。

IRP 的定义依靠于一定的操作系统。

4.9.1.4 事务表

事务表描述了当前需要被做的事务。它与 HC（主机控制器）的实现有关。只有 HCD 和它的 HC 可以访问这些特殊的描述。这些描述包括了事务的各种参数。例如，数据长度(字节)。设备地址、端点号，以及发出数据和存放收到数据的主存区域。

事务表和 HCD 与 HC 间的接口是依赖于实现的，不能作为 USB 说明的一个部分而作出明确的定义。

4.9.1.5 主机控制器(HC)

HC 可以访问事务表，并根据此表引起总线的动作。特别地，HC 提供的报告机制使事务的状态(完成、等待、中止等)是可以知道的。HC 将事务变成相应的总线动作(这种转换依赖于

具体的实现)，这些动作又引起了在以根 Hub 为根的总线拓朴结构上的 USB 包的传递。

HC 可以保证协议规定的对总线访问的限制都被遵守了，例如：包间时限、超时等，HCD 的接口向 HC 提供了参与决定是否允许一个新的通道访问总线的途径。这是因为 HC 的实现中有对两个事务间最小间隔的限制，这种限制支持了总线事务的组合。

事务表与 HC 的接口是被隐藏在 HCD 和 HC 的实现中的。

4.9.2 事务的跟踪

USB 应用设备看到的总线上的数据是放在包中的。(参见第 7 章)。HC 利用某种与实现有关的表示方法来跟踪包，跟踪的信息包括这个包是什么包，到哪个端点去/从哪个端点来，在什么时间传送，有什么顺序。绝大多数客户软件不愿直接与被打了包的数据流打交道，因为这造成了一定的复杂性和对连接的依赖性，从而又限制了实现方法。USB 系统软件(USBD 和 HCD)提供了将客户对数据运动的要求加到包上的方法。对组成一个客户软件和应用层间的数据传送的各个事务，HC 利用 IRP 来跟踪它们。图 5-11 简要说明了事务是如何被组织成四种传送类型的 IRP 的。关于传送的详细协议规定，可见第 7 章。关于客户软件对 IRP 的使用的更详细信息可见第 9 章和特定操作系统的系统说明信息。

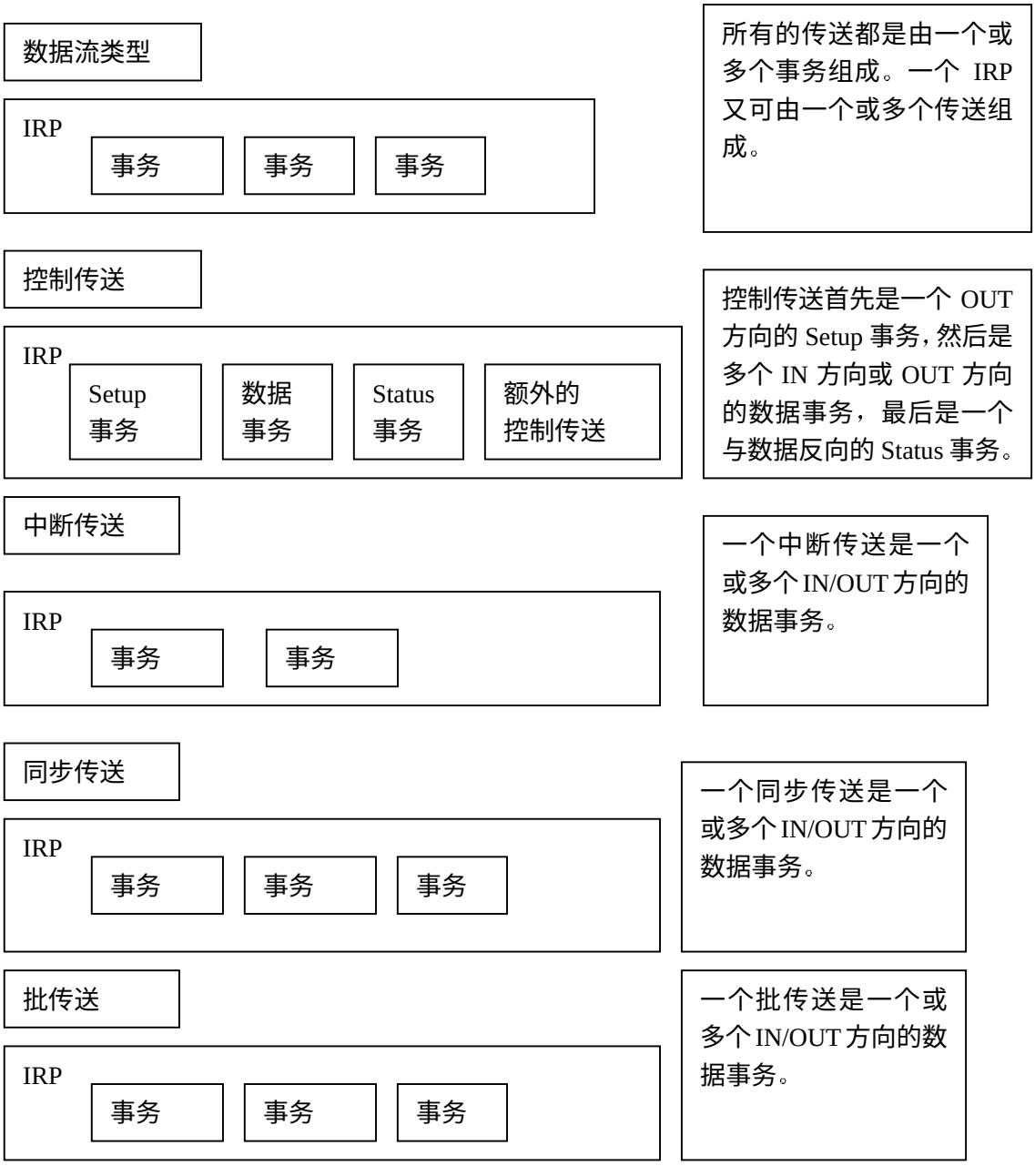


图 4-11　通信流的传送方式

虽然 IRP 要跟踪传送数据的总线事务，HC 仍可自由地选择如何传送这些事务，但必须遵守 USB 定义的限制，即一帧内只可有一个同步传送的事务。一般情况下，端点看到的各个事务的顺序与它们出现在 IRP 中的顺序是一样的，除非发生了错误。例如，图 5-12 表示了两个 IRP，每个 IRP 有 3 个事务，用 2 个通道。对任何传送类型，HC 可在第一帧内先传第一个 IRP 的第一事务，再传第二个 IRP 的第一个事务；同时在第二帧内先传第二 IRP 的第二事务，再传第一 IRP 的第二事务；如图中所示。

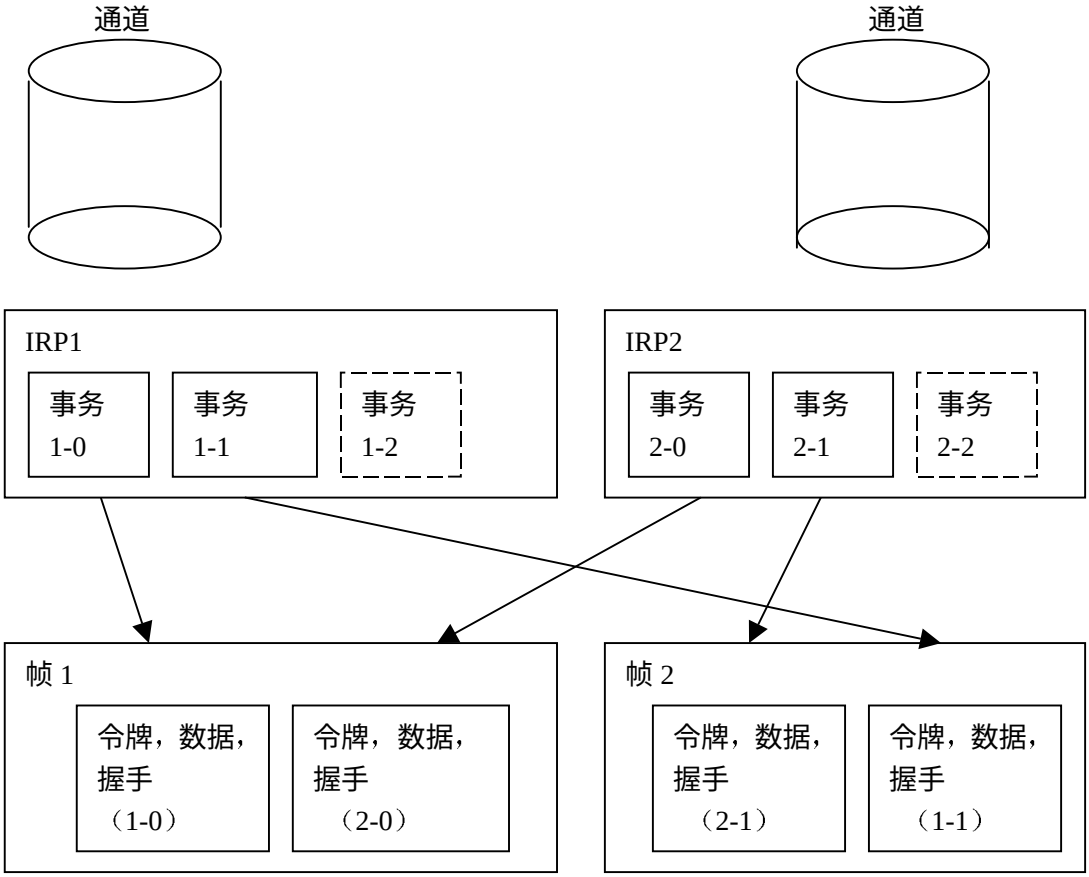


图 4-12　IRP 组织成事务/帧

如果有同步传送，HC 的自由度也就到此为止了。但对控制传送和批传送，HC 可以在这两帧中的任一帧内多传或少传几个第一 IRP 或第二 IRP 的事务。

4.9.3 计算总线事务的时间

当 USB 系统软件允许在总线上新建一个通道，它必须计算对一个给定的事务需要多少总

线时间。这个时间是按照端点报告的最大包长度来计算的，同时也必须包括协议规定的开销、信号强加的位填充的开销、协议规定的包间时间的开销、事务间的时间的开销等等。计算的结果用来检查帧内可用的时间是否够用的，以下是计算用的各方程。

KEY:

Data_bc	The byte count of data payload
Host_Delay	The time required for the host to prepare for or recover from the transmission; Host Controller implementation-specific
Floor()	The integer portion of argument
Hub_LS_Setup	The time provided by the Host Controller for hubs to enable low-speed ports; measured as the delay from the end of the PRE PID to the start of the low-speed SYNC; minimum of four full-speed bit times
BitStuffTime	Function that calculates theoretical additional time required due to bit stuffing in signaling; worst case is (1.1667*8*Data_bc)

由以上方程计算出的总线时间的单位是纳秒，而且计算结果包含了由于主机和设备之间的延迟而造成的传播延时，这些方程是典型的用于计算总线时间的方程，但实现时可选用一些较粗糙的近似计算。

某个特定的事务实际所用的总线时间总是小于计算出的总线时间，因为位填充的开销是与数据有关的，最糟的情况是位填充的时间是原时间的 1.1667(7/6)倍(方程中用 8*1.1667 乘以 Data-bc)。这就是说，当所有规则安排的事务完成后，总有些总线时间是未用的(服从于数据模式的指定)，这些未用的时间可以被重新用于别处，参见 4.9.5 节。

方程中的 Host-Delay 项是与 HC 及系统有关的，它允许 HC 由于申请访问造成的延迟或其它与实现有关的延迟的原因多要求一点额外时间。通过 HCD 接口提供的传送管理功能，这项被包括在方程的实现中。方程的实现可采用 USBD 和 HCD 软件的共同工作来完成，这些方程的结果决定了一个传送或通道是否被 USB 设置所支持。

4.9.4 应用层及软件对缓冲区大小的计算

客户软件和应用设备必须给正在等待传送的数据事务提供缓冲区。对非同步传送，这个缓冲区的大小必须恰好能装下下一个数据包，如果不止一个事务在等待同一个端点，必须为一个事务提供数据缓冲区。由于在应用层与客户软件间的事务的出现，应用可能需要一定大小的缓冲区，关于这个区的精确的最小绝对长度的计算是不在 USB 说明的范围之内的。

HC 能够支持的等待事务的数量应是不受限的，但实际上，要受限于可用内存缓冲的大小和描述器的空间大小等。对同步通道，4.10.4 节描述了影响主机侧和设备侧的缓冲要求的一些细节。一般说来，缓冲区应容纳约等于 1ms 中能传的数据量的两倍大小的数据。

4.9.5 总线带宽归还

USB 带宽和总线访问的允许都是从最糟的情况考虑的。由于不同传送类型的限制以及被算作常数的位填充的时间实际是数据相关，所以在每帧内，与计算出的时间相比，总有些时间剩余。为了提高带宽利用率，这些时间可以用于控制传送或批传送。HC 具体如何去做是与实现相关的。HC 可以考虑等待的 IRP 的传送类型和剩余时间的情况来决定如何使用这些被归还的时间。

4.10 关于同步传送的一些特别考虑

系统的同步传送的能力是由 USB 支持的。在 USB 上进行可靠的同步数据传送必须注意一些细节。同步传送可靠性由几个 USB 部分分别负责：

• 设备/应用
• 总线
• 主机控制器(HC)
• 一个或多个软件代理商

因为时间对于同步传送是很重要的，所以 USB 的设计者必须了解 USB 中的这些部分是如何处理时间问题的。

所有的同步设备必须以各自的描述器的形式来报告自己的能力。同时这些能力也要以某种形式向客户说明，让他们决定该设备是否能解决他们的问题。设备各有不同的专门功能，所以它们的价格也有差异。

在任何通信系统内，发送器和接收器必须同步，也能保证数据传送正确。如果是异步系统，发送器可检测接收器是否收到数据，并且可在出错时，进行简单的重发。

而如果是同步系统，接收器和发送器必须保持时间上和数据上的同步，才能使数据传送正确。USB 不支持同步数据的重发，所以可以使分给同步传送的带宽达到最小。也不会因为重发而丢失同步。然而，重要的是，接收器与发送器不仅在正常情况下要保持同步；在错误发生时，也要能保持同步。

在许多系统，使用一个全局时钟，所以部件与它同步。一个例子就是 PSTN(公共交换电话网)，考虑到接到 USB 上的设备有不同的固有频率，不可能有一个时钟在满足大众化 PC 产品的价格要求的同时，还能满足系统上所有设备和软件的同步要求。USB 定义了一个时间模式，在有较合理价格的前提下，使各色设备在总线上共存。

本节提供了几种选择供端点使用，使其能够尽量缩小一个设备的非 USB 实现与 USB 实现间的差异，后面有一例子说明设备的非 USB 应用与 USB 应用间的相似和相异。

本节最后还介绍以下一些重要概念：

• USB 时间模型：USB 系统中使用的时钟，它影响着同步数据传送。
• USB 帧时钟与应用时钟同步的方法：USB 帧时钟如何与应用时钟发生联系。
• SOF 跟踪：关于 SOF 令牌和 USB 的帧，以及同步端点的责任和机会。
• 数据预缓冲：在数据生成、传送和消费前，对数据积累的要求。
• 出错处理：对于同步传送的出错处理的细节。
• 为速率匹配而做的缓冲：可用方程计算同步端点所需的缓冲区大小。

4.10.1 典型的非 USB 同步应用

这个例子是具一般性的例子。更复杂或更简单的情况都是有可能的。相关的 USB 特性可能用的不太适当。

例子中有一个 8KHz 的单声道麦克风，通过一个送进入数据流的混合器驱动器（Mixer Driver）连上一个 44KHz 的立体声扩音器。混合器期望发送和接收的数据有一个确定的取样速率和编码。在输入、输出口的速率匹配器驱动器（Rate Matcher Driver）将固有取样频率和编码变为混合器期望的取样速率和编码，图 4-13 说明了这个例子。

PC 中的主控时钟(由被实时时钟驱动的软件提供)用来提醒混合器向输入源要输入数据和向输出提供输出数据。这个例子中，假设此时钟每 20ms 提醒一次。麦克风和扬声器的采样时钟各不同步，且与主控混合器时钟也不同步。麦克风的固有速率为 1 秒 8000 次取样，一次取样结果为一个字节，它以此速度生产数据。扬声器的固有频率为 1 秒 44100 次采样，一次采样结构为 4 字节，它以此速度消费数据，系统中的这三个时钟会产生浮动和抖动，速率匹配器的时钟可以与这三种时钟(混合器时钟，源时钟，目的地时钟)均不相同。

速率匹配器一直监视设备的数据速率，不时插入一个额外采样或将两个采样合并为一个，来调整设备的时钟，向主控混合器时钟看齐。调整工作可以每两秒做一次，但一般都是不定期的，不频繁的。速率匹配器提供几个额外的缓冲来完成速率匹配工作。

注意：有些应用不能进行采样的调整。这时或者用一些方法来消除主时钟与设备时钟的不一致；或者用一些方法使两时钟同步，不让不一致发生。

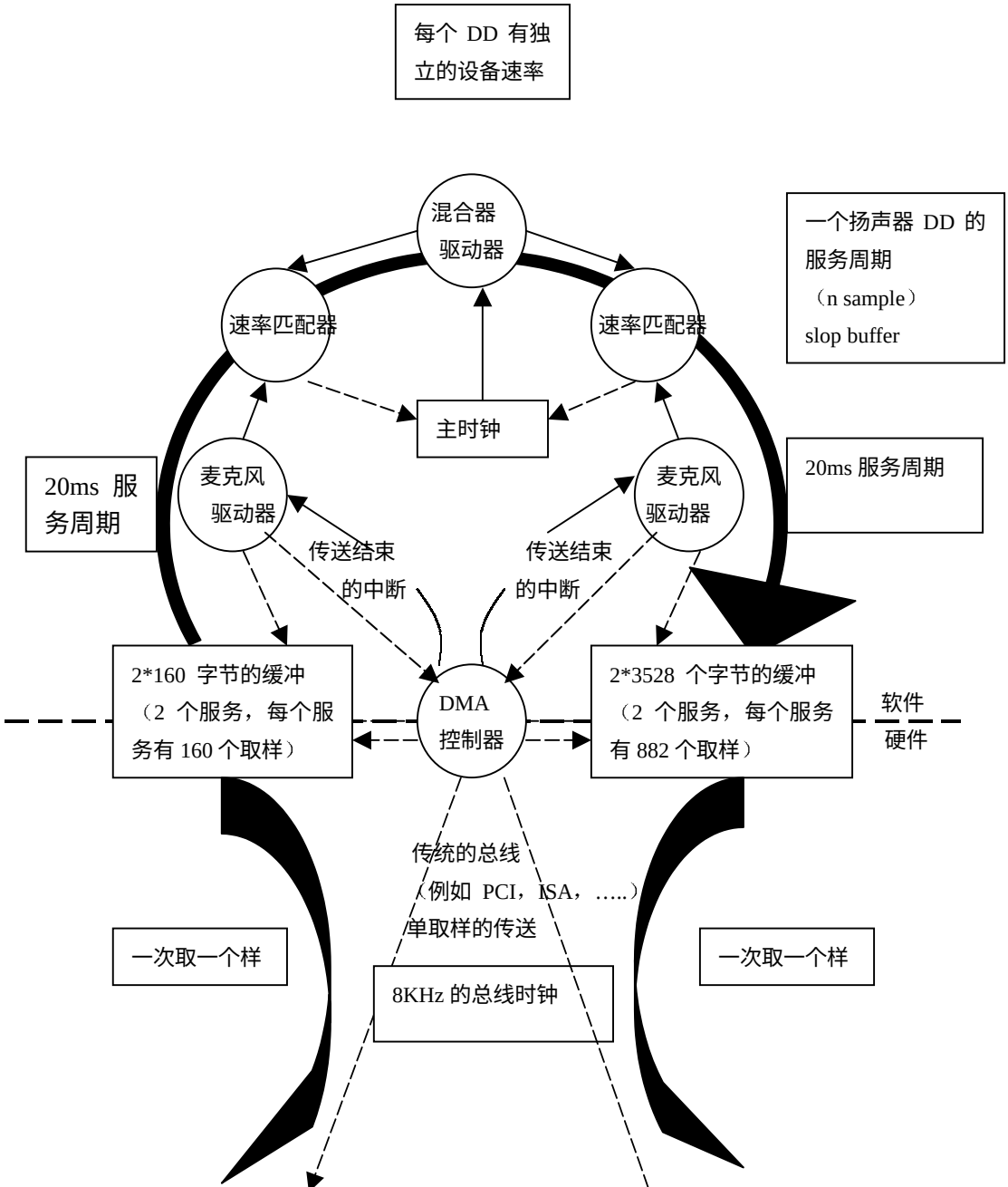
混合器总是精确地从它的输入设备处收到一个服务周期的数据(20ms)或为它的输出设备产生一个服务周期的数据。对混合器的延迟不能超过一个服务周期的长度。混合器认为这些延迟不会积累起来。

当输入设备和输出设备处理完一个服务周期的数据后，要能对 DMA 发的硬件中断做出发数据或收数据的反应。他们总是发或收一个服务周期的数据。输入设备每个服务周期(20ms)产生 160 个字节(10 个取样)。输出设备每 20ms 的服务周期消费 3528 个字节(882 个采样)。DMA 控制器在主机缓存和设备传一个采样的速率比这些设备的采样速率都要快。

输入、输出设备都提供能容纳两个服务周期数据的系统缓存。一个给 DMA 控制器处理，另一个备用，以防第一个被取空。如果当前的缓存正被逐步取空，硬件中断将提醒设备驱动器，该驱动器要求速率匹配器给它一个缓存。在当前缓存被取空以前，设备驱动器会为新给的缓存申请一个新的 IRP。

设备提供的数据缓存两个取样长度那么长，以保证当系统区对前一个采样做反应时，总可以为下一个采样同时做处理。

驱动器的服务周期可在操作系统中的中断延迟之后继续。不同的操作系统环境要求不同的服务周期来保证可靠的操作。服务周期长度的选择要能够使系统的中断负荷最小，因为系统中还有别的软件会要求处理时间。



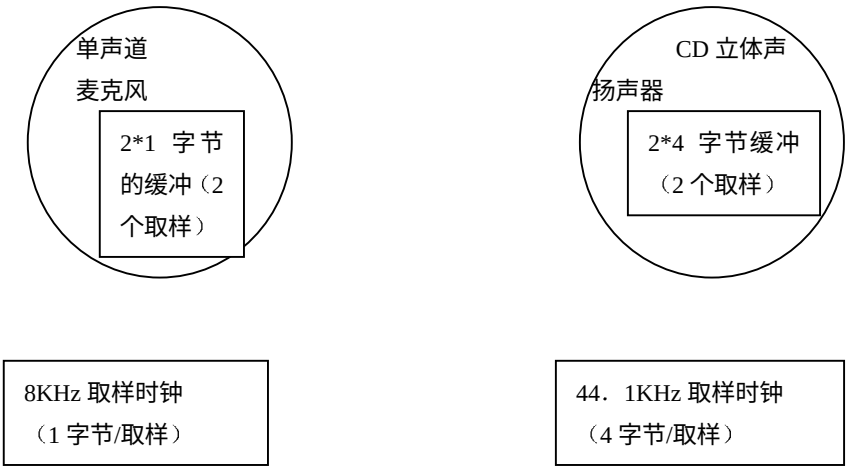


图 4-13　非 USB 同步的例

4.10.2 USB 时钟模型

USB 系统中的时间是由时钟体现。实际上，USB 系统中有多个时钟：

• 取样时钟：这个时钟决定了客户软件与应用设备间传数据取样的固有频率。在非 USB 和 USB 实现中，这个时钟不必不一样。

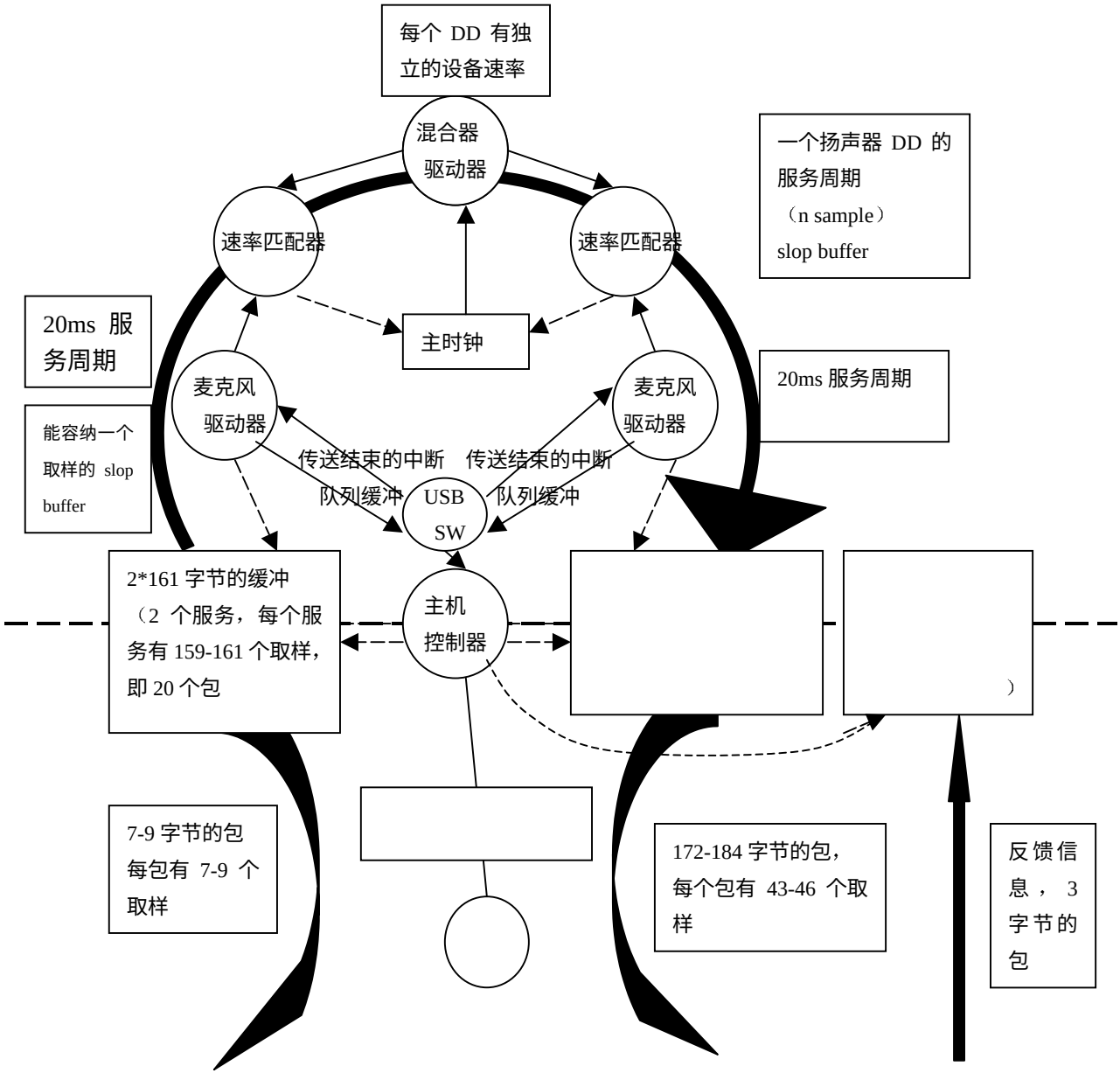
• 总线时钟：这个时钟频率为 1KHZ。总线上 SOF 包的时间体现了此时钟。这个时钟相当于非 USB 的例子中的 8KHZ 时钟。在 USB 系统中，总线时钟一般比采样时钟频率低。在非 USB 系统中，总线时钟一般比采样时钟频率高。

• 服务时钟：它取决于客户软件服务 IRP 的速度，这些 IRP 可能在两次执行之间积压下来。在非 USB 和 USB 系统中，这个时钟可以相同。

如果每个设备驱动器在高速取样的每次取样后都要被中断，现存的操作系统是无法支持变化很大的各种同步通信流的。因此，一般等有多个取样或多个取样包后，再将其交由客户软件处理，然后送给 HC，根据事先约定的总线访问要求在总线上排队。图 4-14 提供了一个 USB 系统时钟环境下的例子，与图 4-13 的非 USB 系统例子相对照。

图 4-14 表示了一个典型的信息环路，麦克风作为输入设备，扬声器作为输出设备。涉及的时钟、包、缓存也标在了图中。在以后的几节后，图 4-14 将被进一步详细地说明。

这个例子的重点是突出 USB 与前一个非 USB 例子的不同。不同之处在于缓存的区域，USB 总线时钟的同步和延迟，。设备驱动器上的客户软件在大多数情况下可不受影响。


每个 DD 有独立的设备速率
混合器驱动器
速率匹配器
速率匹配器
一个扬声器 DD 的服务周期（n sample）slop buffer
20ms 服务周期
主时钟
麦克风驱动器
麦克风驱动器
20ms 服务周期
能容纳一个取样的 slop buffer
传送结束的中断
传送结束的中断
队列缓冲
USB SW
队列缓冲
2*161 字节的缓冲（2 个服务，每个服务有 159-161 个取样，即 20 个包
主机控制器
）
7-9 字节的包每包有 7-9 个取样
172-184 字节的包，每个包有 43-46 个取样
反馈信息，3 字节的包

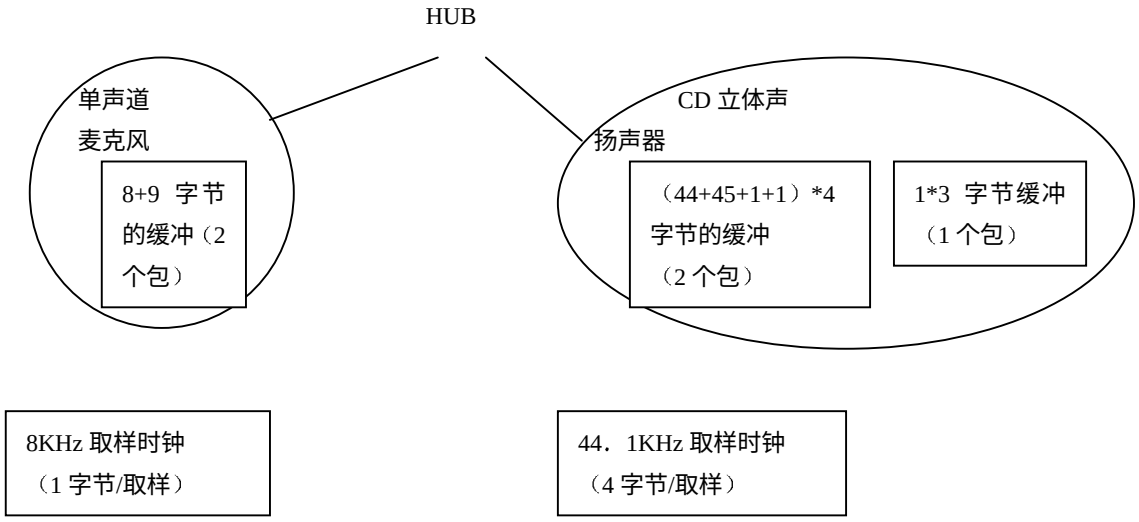


图 4-14　USB 同步的例

4.10.3 时钟同步

为了可靠地传送同步数据，前面提到的三个时钟必须同步。如果不这样，就会出现一些不希望看到的时钟间的问题：

• 时钟漂移：两个应正常同步的时钟，因为实现上的问题，在走很长一段时间后，可能会走得快慢不一。如果不及时校正，则与其中一个作为标准的时钟相比，另一个时钟就会导致数据过多或过少。

• 时钟抖动：由于温度等的原因，时钟频率会发生改变，也造成了数据实际发出的时刻与它被期望发出的时刻不同。

• 时钟间相位差异：如果两个时钟不被锁相，由于时钟的错位，在不同的时间点上会收到不一样多的数据。这会造成量化/取样的不真。

总线时钟提供了一个中心时钟，USB 硬件设备和软件可以向它同步。但在大多数 PC 的操作系统中，仅靠实时调度的支持，软件并不能精确地锁相、锁频到总线时钟上。软件知道 USB 上的数据是打包的。对同步传送，在一帧内只传一个包，而帧时钟是相对精确的，软件可以凭实际花费的帧时间调整它处理数据的量。

4.10.4 同步设备

USB 为同步设备提供了一个框架，定义了同步类型及同步端点如何提供数据速率反馈，以及同步端点如何被连结到一起。同步设备包括被取样的模拟设备(如：声音设备、电话设备)和同步数字设备。对端点同步类型的分类的依据是它们将自己的时钟同步到与它们相连设备的能力。反馈信息能指出什么是要求的时钟频率。进行连接的能力依赖于想要的连接质量、端点的同步类型，以及进行此连接的主机应用程序的能力。额外的设备级信息是需要的，要多少取决于应用程序。

注意：数据是个一般性的词，可代表取样过的模拟信号(如声音)，也可指抽象的信息。数据速率可以指模拟信号被取样的速率，也可以指数据时钟的速率。

以下信息用于决定如何连接同步端点：

• 同步类型

——异步：不同步、但目的（sink）能提供数据速率反馈

——同步：同步到 USB 的 SOF 时钟

——可调：用反馈或 feed forward 的数据速率信息实现同步

• 可获得的数据速率

• 可获得的数据格式

同步类型和数据速率信息决定了源和目的之间的速率是否匹配，如果有可接受的转换过程存在，就允许它们相连。应用程序有责任决定在可获得的处理资源的限制和其它限制(如延迟)下，是否支持一个连接。具体的 USB 设备级定义了如何描述同步类型和数据速率。

数据格式匹配和转换也是需要的，但不是仅对同步连接才要求。关于数据格式转换的细节在其它文件中可找到。

4.10.4 同步类型

有三个明显的同步类型，表 4-7 列出了它们各自的端点的同步特征(作为源或目的)。这三个类型的排列是根据它们的能力逐步增强的顺序进行的。

表 4-7 同步特性

	源	目的
异步	自由地安排 Fs 提供隐式的 feed forward（data stream）	自由地安排 Fs 提供显式的反馈（中断通道）
同步	Fs 与 SOF 同步 使用隐式的反馈（SOF）	Fs 与 SOF 同步 使用隐式的反馈（SOF）
可调	Fs 与 SOF 同步 使用显式的反馈（控制通道）	Fs 与数据流同步 使用隐式的 feed forward（data stream）

4.10.4.1.1 异步

异步不能向 SOF 或其它 USB 范围内的时钟同步。它们的源和目的的同步数据流都是在一个固定的数据速率上(单数率端点)，有一个有限的数据率的个数(32KHZ,44KHZ,48KHZ)，或一个

连续的可编程的数据率。如果可编程，是在同步端点的初始化时候进行的，异步设备需在它们的端点描述器中报告自己的可编程能力。数据速率是锁到一个 USB 的外部时钟或一个内部的自由时钟上的。这些设备将进行速率匹配的任务留给了 USB 环境中的其它部分。异步源端点通过它们每帧的生产的采样个数隐式地体现了自己的数据速率信息，异步目的端点必须用显式地反馈向一个可调的驱动器说明自己的数据率(见 4.10.4.2)。

异步源的一个例子是 CD 播放器，它依据自己的内部时钟或共振器来提供数据，另一个例子是数字声音广播(DAB)的接收器或数字卫星接收器(DSR)，它们的取样速率取决于广播侧而不是 USB 的控制，异步目的端点可以是低价格的扬声器，用自己的内部取样时钟。

另一个情况 USB 上有两个或更多的设备要求对 SOF 的产生拥有控制权以便能够作为同步设备运作。例如有两个电话设备，有各自的外部时钟。一个电话可被接到 Private Branch Exchange(PBX)上，PBX 不与 ISDN 同步。另一个电话直接接到 ISDN 上，每个设备向网上产生或接收数据都是用自己的外部驱动速率。因为只有一个设备可获得 SOF 的控制权，另一个设备的数据速率就只能与 SOF 异步。这个例子表明，每个有能力控制 SOF 的设备都可能被迫成为异步设备。

4.10.4.1.2 同步

同步端点可有自己的时钟系统，但此系统可被 SOF 同步所控制。这些端点可以：

• 将自己的取样时钟作为 SOF 时钟（1ms 1 次）的奴隶(通过一个可编程的 PCL)。

• 控制 USB 的 SOF 产生速率，显而易见，这样的话，它们的时钟自然与 SOF 同步。如果这些终点不被给予 SOF 的控制权，它们降级到异步模式。

同步端点发出或接收同步数据流的数据速率有以下几种情况。可以使用一个固定的数据速率(单频率端点)，或有限个数的几个速率(32KHZ，44.1KHZ，48KHZ)，或一个连续的可编程的数据速率。如果可编程，在同步端点的初始阶段进行设定。在一系列的 USB 帧中产生的采样数或数据单元数是可确定的和周期性的。同步设备须在端点描述器中报告自己的可编程能力。

同步源的一个例子是数字麦克风。它向 SOF 同步自己的取样时钟，并在每个帧内产生同

样多的声音采样。另一种可能是从 ISDN 的“Modem”中来的 64kb/s 的位流。如果 USB 的 SOF 产生时钟被同步到 PSTN 的时钟(也许通过同一个 ISDN 设备)，数据的产生也会被同步到 SOF，且端点将产生一个稳定的 64kb/S 的数据流，以 SOF 时钟为参照。

4.10.4.1.3 可调

可调的端点是能力最强的端点，它们可以在自己的操作范围内产生和接收任意速率的数据。对可调的源端点来说，数据产生速率是被数据的目的端点所控制的。目的给源一个数据速率的反馈(见 4.10.4.2)。这样源就知道了目的要求的速率。可调的端点可以和任何类型的目的通信。对可调的目的端点来说，数据速率的信息是被包含在数据流中的。在一个确定的平均时间段内接收的平均取样数目决定了瞬时的数据速率。如果在操作中，这个数目发生了变化，则数据速率就作出相应的调整。

数据速率的变化范围是以某个速率为中心(例如 8KHZ)，它是从几个可编程的速率或自动检测的数据速率(32KHZ，44.1KHZ，48HZ)中选择出来的，速率也可以是一个或几个范围(如 5KZ-12KHZ 或 44KHZ-49KHZ)，可调端点也必须在它们的端点描述器中报告自己的可编程能力。

可调源的一个例子是 CD 播放器。它包括一个完全可调的取样速率转换器(SRC)，所以输出取样的速率不必为 44.1HZ，而可以是在 SRC 控制范围内的任何值。可调目的端点包括高端（High-end）数字扬声器，耳机等。

4.10.4.2 反馈

异步目的向可调源提供一个反馈，来表明它想要的数据速率(Ff)是多少。要求的数据速率必须精确到 1 秒 1 个取样(1HZ)以上，这才能够使源速率达到一个高质量，并能容忍反馈中的延迟和错误。

Ff 的值包括一个小数部分和一个整数部分，小数部分是为了达到 1KHZ 频率帧的分辨率，整数部分给出了每帧内的最小取样数目。在 1KHZ 的帧频率下，小数部分要求有 10 位来表示一个取样(1000/2^10=0.98)。这是一个十位的小数部分，可以用无符号的定点二进制 0.10 格式表示。整数部分也需要 10 个位(2^10=1024)来表示每帧内 1023 个单字节取样。10 位的整数用无符号的定点二进制格式 10.0 表示。两部分合在一起的 Ff 值可用无符号的定点二进制格式 10.10。此格式需要 3 字节(24 bits)。因为最大的整数值是 1023，数 10.10 格式的数被向左靠齐成为 24 位，所以它的格式为 10.14，小数点后的头十位被承认，剩下的低 4 位被可选地用于扩展精度，或者可以被看作 0。第 7 章中还会介绍其它的多字节区中定义的位和字节的顺序。

每帧中，可调源将 Ff 加到从前一帧中剩下的小数部分上，然后产生总数中整数部分大小的取样，将小数部分的取样再算到下一帧。源可以通过许多帧的 Ff 的情况来确定一个更精确的速率。

目的端点可以通过以下方法确定 Ff。它在 2^(10-P)帧的时间内(P 为整数)，以 Fs*2^P 的频率数时钟的周期。P 的经验取值在［0，10］的范围内，因为没有点（point）用比 Fs 更慢的时钟，且没有点（point）企图在一帧内更新不止一次。计数器被读入 Ff,并且每 2^(10-P)帧时间后复位一次。只要没有时钟被漏掉，计数在很长一段时间内将是很精确的。端点只要保证计数器的位数即可，这个位数要能容纳它的最大 Ff。

数字电话端点通过将 64KHZ 的时钟分频(P=3)来得到自己的 8KHZ Fs。64KHZ 是它用于数据流串行的时钟。64KHZ 的时钟相位可以作为一个额外的位来提高精确度，相当于取了 P=4。这使 Ff 每 2^(10-4)=64 帧刷新一次。所以就需要一个 13 位的计数器来保存 Ff,3 位表示每帧 8 个取样，10 位表示小数部分。13 位的格式为 3.10，因为它在 10.14 Ff 值格式下，所以其余的位置设为 0。

P 的选择是与端点有关的，选 P 有以下的指导原则：

• P 在［1，9］的范围内

• P 的值越大越好，因为这样可以减少帧计数器的大小而提高 Ff 刷新的速度。高的刷新速度会保证对源端数据速率的更好控制，减少了用于处理 Ff 变化的缓存的大小。

• P 必须比 10 小，这样保证了至少两帧以上才会刷新一次，避免了 SOF 的抖动。

• P 不能为 0，这样可保证在一次 Ff 值丢失的情况下，与产生的采样数的偏离小于 1。

从反馈寄存器中读 Ff 由同步传送来完成。每 2^(10-P)要保告一次反馈信息。Ff 在每个刷新周期必须被报告至少一次。如果一个刷新周期内将一个同样的 Ff 值报告一次以上，不会有什么用处。端点只有在 Ff 值与上次比有变化时才报告一下。

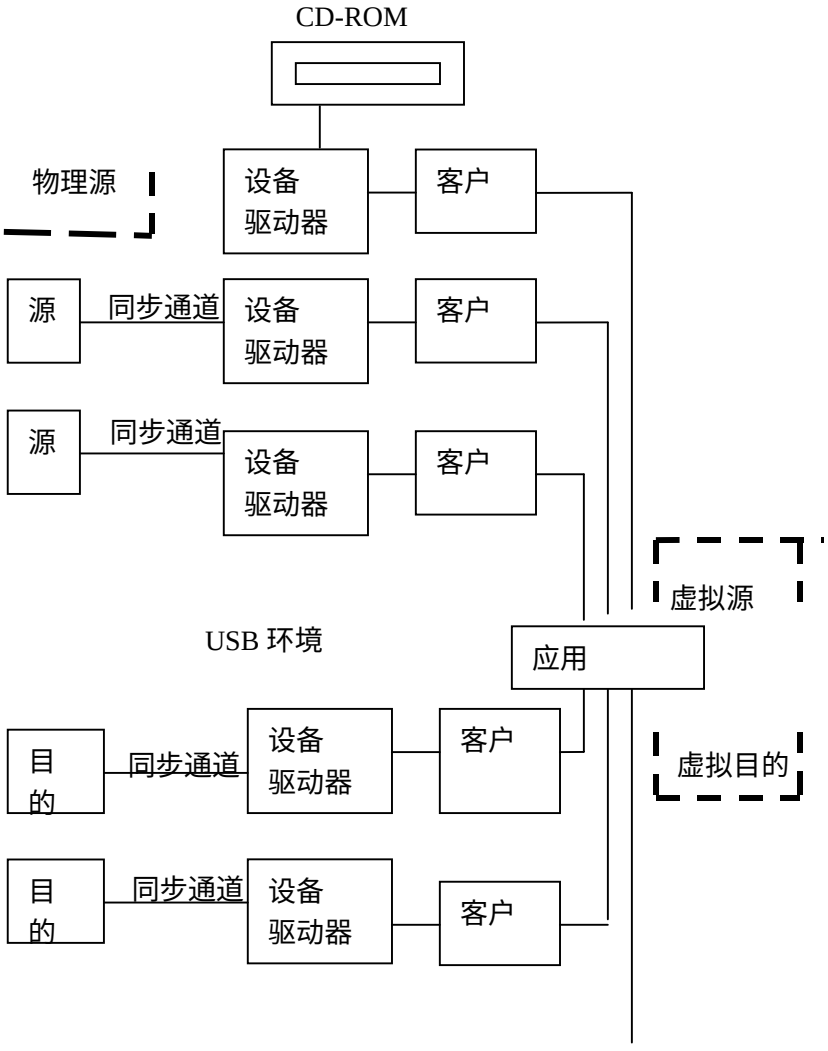
经过一个很长的阶段，源可能会多发一个或少发一个采样。这可能是因为测量 Ff 的错误或误差的积累。目的端必须有足够的缓冲能力处理这些情况。一旦目的端发现这些情况，应当将报告的 Ff 调整正确。对时钟的相对漂移，调整也是必须的，这种校正处理的实现与端点有关，不被指定。

在双向通话的连接中，可调源可以获得可调目的的数据速率，但该源是采用目的端的身分获取此速率的，所以，此时不需要反馈通道。

4.10.4 连接

连接模型被用来详细描述源到目的的连接过程。这个模型涉及到的各个不同的组成部分及它们如何互相作用来完成连接。

模型提供了多个源和多个目的的情况，图 4-15 说明了一个典型的情况(高度浓缩的及不完全的)，物理设备通过不同的物理层、软件层接到主机应用软件上(USB 说明中描述)。在客户界面层、应用层看到一个虚拟设备。从应用层的角度看，只有虚拟设备存在。物理设备与虚拟设备的关系由设备驱动器和客户软件决定。



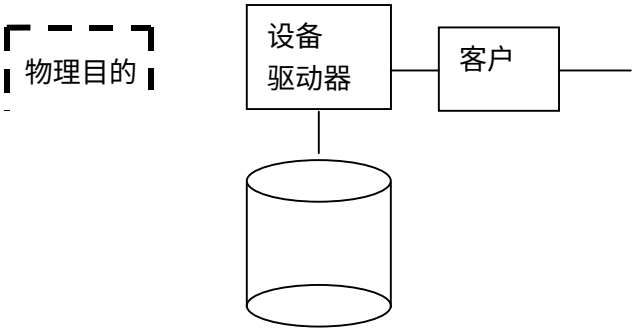


图 4-15 源与目的的典型连接

设备制造商(或操作系统销售商)必须提供设备驱动器软件和客户界面软件，以便将物理实现转变为一个 USB 需要的软件实现(虚拟设备)。如前所述，以加入软件的功能为基础，虚拟设备可以展现物理设备的不同同步行为。然而，同步的分类对物理设备和虚拟设备是一样的。所有的物理设备都必须属于 3 种类型中的一种。所以，加入设备驱动器或客户软件的功能应与物理设备的功能一样。名词“应用”必须被换成“设备驱动器/客户软件”，当物理源被联系到虚拟源，“虚拟源设备”应被换成“物理源设备”，“虚拟目的设备”应被换成“虚拟源设备”。当虚拟目的联系到物理目的，“虚拟源设备”必须被换成“虚拟目的设备”，“虚拟目的设备”应被换成“物理目的设备”。

将速率调节器(RA)功能加到设备驱动器/客户软件层上有明显的好处，它可将所有应用独立起来，使设备不必关心与速率调节有关的一些规定和问题。多速率的应用也会退变为简单一些的单速率系统。

注意：这个模型并不局限于 USB 设备。例如，一个包含 44.1KHZ 的声音的 CD-ROM 驱动器可以作为异步、也可作为同步或可调的源。异步操作时，CD-ROM 以它读盘的速率来填充自己的缓冲区、驱动器根据它的 USB 服务的间隔来取数据。同步操作时，驱动器根据 USB 服务的间隔(10ms)和名义上的取样速率(44.1KHZ)决定每个 USB 服务周期时输出 441 个取样。可调操作时，建一个取样速率转换器来匹配 CD-ROM 的输出速率和各种不同的目的端的取样速率。

使用这个参照模型，可以定义在不同的源和目的间建立连接所必须的操作。而且，这个模型还指出了这些操作必须或可以在哪层发生。第一阶段，物理设备被映射到虚拟设备，这由驱动器和/或客户软件完成，基于软件的能力，物理设备可被映射成具有完全不同的同步类型的虚拟设备。第二阶段，是对虚拟设备的应用，在软件栈的驱动器/客户层加入数据匹配功能后，与虚拟设备通信的应用就可以不必为连上它的每个设备作速率匹配工作。一旦虚拟设备的特性定了下来，实际的物理设备的特性就不必关心了。

作为一个例子，考虑在源侧将一个混合器应用连到不同的源上，每个源以各自的频率和时钟工作。在混合发生前，所有的流都必须被转换成同一个频率，并锁到一个参考时钟上。这个工作可以在物理到虚拟的映射层做，也可以在应用层为每个源单独处理。相似的工作在目的侧也要做，如果应用将混合的数据流送到不同的目的设备，它可以为每个设备做速率匹配的工作，也可以把这个工作留给驱动器/客户软件去做(如果可能的话)。

表 4-8 表明了将源端点连到目的端点时，应用必须做的工作。

表 4-8 连接的要求

	源端点		
目的端点	异步	同步	可调
异步	异步的源/目的间的 RA	异步的 SOF/目的的 RA	数据+反馈 Feedthrough

	参见注释 1。	参见注释 2。	参见注释 3。
同步	异步的源/SOF 的 RA 参见注释 4。	同步 RA 参见注释 5。	数据 Feedthrough +应用反馈 参见注释 6。
可调	数据 Feedthrough 参见注释 7。	数据 Feedthrough 参见注释 8。	数据 Feedthrough 参见注释 9。

注释：

1. Asynchronous RA in the application. Fs i is determined by the source, using the feedforward information embedded in the data stream. Fs o is determined by the sink, based on feedback information from the sink. If nominally Fs i = Fs o , the process degenerates to a feedthrough connection if slips/stuffs due to lack of synchronization are tolerable. Such slips/stuffs will cause audible degradation in audio applications.

1：应用中的异步速率调节器。Fsj 由源决定。源利用数据流中包含的 feedforward 信息来做决定 Fs0 由目的决定。它的决定依据是从目的来的反馈。如果 Fs0 由目的决定。它的决定依据是从目的来的反馈。如果 Fs0=Fsj，且由于缺少同步而引起的 slips/stuffs 可以被容忍的话，处理将退化成为一个 feedthrough 连接。这样的 slips/stuffs 在声音应用中将引起可听出来的声音质量下降。

2. Asynchronous RA in the application. Fs i is determined by the source but locked to SOF. Fs o is determined by the sink, based on feedback information from the sink. If nominally Fs i = Fs o , the process degenerates to a feedthrough connection if slips/stuffs due to lack of synchronization are tolerable. Such slips/stuffs will cause audible degradation in audio applications.

2：应用中的异步速率调节器。Fsj 由源决定，但必须锁到 SOF 上。FS0 是由目的决定的，但要基于来自目的的反馈信息。如果 Fsj=FS0，且由于缺少同步而引起的 slips/stuffs 可以被容忍的话，这个过程退化为一个 feedthrough 连接，这种 slips/stuffs 在声音应用中将引起可听的出来的声音质量下降。

3. If Fs o falls within the locking range of the adaptive source, a feedthrough connection can be established.Fs i = Fs o and both are determined by the asynchronous sink, based on feedback information from the sink. If Fs o falls outside the locking range of the adaptive source, the adaptive source is switched to synchronous mode and Note 2 applies.

3：如果 FS0 落在可调源的锁定范围内，一个 feedthrough 可以被建立。Fsj=Fs0，它们都由异步目的决定，但要基于从目的来的反馈信息。如果 Fs0 落在可调源的锁定范围外，可调源转变成同步模式，注释 2 适用。

4. Asynchronous RA in the application. Fs i is determined by the source. Fs o is determined by the sink nd locked to SOF. If nominally Fs i = Fs o , the process degenerates to a feedthrough connection if lips/stuffs due to lack of synchronization are tolerable. Such slips/stuffs will cause audible degradation n audio applications.

4：应用中的异步 RA。Fsj 由源决定，FS0 由目的决定并锁到 SOF 上。如果 Fs0=Fsj,且由于缺乏同步而引起的 slips/stuffs 可被容忍的话，这个过程退化成一个 feedthrough 连接。在声
音应用中，这些slips/stuffs将引起声音质量下降。

5. Synchronous RA in the application. Fs i is determined by the source and locked to SOF. Fs o is determined by the sink and locked to SOF. If Fs i = Fs o , the process

degenerates to a loss-free feedthrough connection.

5：应用中的同步 RA。Fsj 由源决定，且锁到 SOF 上。FS0 由目的决定，并锁到 SOF 上。如果 Fsj=FS0，这个过程退化为允许丢失的 feedthrough 连接。

6. The application will provide feedback to synchronize the source to SOF. The adaptive source appears to be a synchronous endpoint and Note 5 applies.

6：应用提供反馈来将源同步到SOF上。可调源看成一个同步端点，注释5适用。

7. If Fs i falls within the locking range of the adaptive sink, a feedthrough connection can be established. Fs i = Fs o and both are determined by and locked to the source.

If Fs i falls outside the locking range of the adaptive sink, synchronous RA is done in the host to provide an Fs o that is within the locking range of the adaptive sink.

7：如果 Fsj 落在可调目的的锁定范围，则一个 feedthrough 连接可以被建立。Fsj=Fs0，且它们都由源决定，并锁定到源上。如果 Fsj 落在可调目的的范围外，同步 RA 由主机完成，主机再提供一个 Fs0 落在可调目的的范围内。

8. If Fs i falls within the locking range of the adaptive sink, a feedthrough connection can be established.Fs o = Fs i and both are determined by the source and locked to SOF.

If Fs i falls outside the locking range of the adaptive sink, synchronous RA is done in the host to provide an Fs o that is within the locking range of the adaptive sink.

8：如果 Fsj 落在可调目的的锁定范围内，一个 feedthrough 连接可被建立。Fs0=Fsj，且都由源决定，并锁定到 SOF 上。如果 Fsj 落在可调目的的锁定范围外，同步 RA 由主机完成，主机再提供一个落在可调目的的锁定范围内的 Fs0。

9. The application will use feedback control to set Fs o of the adaptive source when the connection is set up. The adaptive source operates as an asynchronous source in the absence of ongoing feedback information and Note 7 applies.

9：当连接建立时，应用可以用反馈控制来设定可调源的Fs0，没有反馈时，可调源变成一个异步源，此时注释7适用。

在需要 RA 但 RA 不可得时，速率调节过程只能被模拟一下，即采用漏掉一两个采样或填充一两个采样的方法。此时，连接有可能被建立，但可能会有一个对质量不佳的警告；或者，连接根本不能建立。

4.10.4.3.1 音频连接

当以上所说的一般连接应用到音频连接上时，RA 过程就被取样速率转换所取代，这是一种速率调整的特殊形式。不用错误控制，而用某种形式的取样插补来匹配输进和输出的取样速率。依靠这种手段，谈话的声音质量(扭曲(distortion)、信噪比等)均有很大改善。一般地来说，更强的处理能力带来更好的质量。

4.10.4.3.2 同步数据连接

在同步数据连接中，我们使用 RA。对许多实现了差错控制的应用来说，偶尔的 slips/stuffs 是可以接受的，差错控制包括差错检测和丢弃(discard)，差错检测和重发，或者前向纠错。slips/stuffs 的发生速率主要看源的时钟和目的时钟误差的大小，它们是信通上最主要的差错来源。如果差错控制足够强大的话，连接仍可建立)。

4.10.5 数据预缓存

USB 要求设备在处理和传送数据前要将数据预缓存一下，这样主机在处理各个通道的事务时就具有更大的灵活性。

从设备到主机的传送中，端点必须在帧 x 期间积累采样，直到收到帧 X+1 的 SOF 信号。它将帧 x 的数据锁到它的包缓冲中，准备好将包含这些取样的包在帧 x+1 期间发出去。至于在帧内的何时将发送这些数据只能由 HC(主机控制器)决定，而且对每一帧的情况可以不

同。

从主机到设备的传送中，在帧 Y 期间，端点收到从主机来的包。当它收到帧 Y+1 的 SOF 信号后，它才可以开始处理在帧 Y 期间收到的数据。

这个机制允许端点将 SOF 令牌用作打入时钟(stable clock)，而且抖动和/或漂移都很小。当 HC 在总线上传包时，这个机制也允许 HC 可以精确地改变在一帧内何时将包传出。与到每帧的 SOF 距离相同的总线访问相比，这种预缓存会造成端点准备好数据和数据真正在总线上传送的时间之间的额外延迟。

图 4-16 说明了设备到主机之间的传送的时间序列(IN process)，数据 D0 在 Ti 处的帧 Fi 期间积聚，在帧 Fi+1 期间发给主机。相似的，从主机到设备的传送(OUT process)，数据 Do 由端点在帧 Fi+1 期间收到，在帧 Fi+2 期间处理。

Time:	Ti	Ti+1	Ti+2	Ti+3 ...	Tm	Tm+1	...
Frame:	Fi	Fi+1	Fi+2	Fi+3 ...	Fm	Fm+1 . ..	
Data on Bus:		D0	D1 ...	D2..	D0	D1	
OUT Process:			D0	D1 ...		D0 ...	
IN Process：	D0	D1…		D0…			

图 4-16 数据预缓冲

4.10.6 SOF 跟踪

支持同步通道的应用设备必须要能够接收和理解 SOF 令牌才能支持预缓存。由于 SOF 可能会被破坏，所以设备必须能从从被破坏的 SOF 中恢复。这个限制使得只有高速设备才可使用同步传送，因为低速设备不能发现总线上的 SOF。因为 SOF 包可能在传送中被毁坏，所以支持同步传送的设备需有合成 SOF 的能力，这个 SOF 由于总线错误而不能被他们看到。

同步传送要求适当的数据在适当的帧内被传送。USB 要求当同步传送被显示给 HC 时，它必须指出它的第一帧的帧号。HC 不会在指定的帧前开始传第一个事务。如果当前的帧内无事务在等待传送，HC 在同步通道上就什么都不传。如果同步传送要求的帧已经过去了，HC 会跳过(即不传)前面的事务直到对应当前帧的事务到来。

4.10.7 差错处理

同步传送不进行数据包的重发(即接收方不发握手信号给发送方)，所以数据传送的同步不会被打乱。但数据传送的代理商有责任知道何时发生了一个错误以及该错误如何影响通信流。特别地，对于数据包序列(A，B，C，D)，USB 有够的信息。所以一个漏掉的包(A， ，C，D)是可以被测的，并且不会在不知道的情况自动变成不正确的数据或时间序列(A，C，D)或(A ， ，B，C，D)。协议提供了 4 种机制来支持这个想法：一帧一包；SOF；CRC 以及总线事务超时。

• 同步传送要求在正常的操作中，一帧只能有一个数据事务。USB 并不干涉每帧内是什么样的数据。数据发送器（源）决定提供什么样的数据。这种数据每帧(data-perframe)结构很规则，它提供了一个框架，这个框架是丢失数据的错误检测的基础。在总线传送期间事务的任何一个阶段都可能有破坏。第 7 章将介绍每种错误对协议的影响。

•因为每帧前都有一个 SOF，接收器会在总线上看到 SOF。接收器可以凭此发现在两个 SOF 之间，它期望的帧没有出现。特别地，SOF 也有可能被破坏，所以设备必须能够重建被漏掉的 SOF(见 4.10.6)。

• 数据包可能在总线上被损坏，所以 CRC 保护可以让一个接收器发现它收到的数据包是否被破坏了。

• 当接收器成功收到一个令牌包后，如果发现有总线事务超时，它可以不再准备去接收数据包。这些细节都在协议中有定义。

一旦接收器决定不再接收某个数据包，它必须知道被丢掉的这个包的数据长度，才能在

做出相应的错误后的恢复。如果每帧的数据流都是一样的长度，这个值就是一个已知的常数。然而，有些情况，帧与帧的数据长度不同。这时，接收器和发送器要有一个机制来决定丢失的包的长度，这个机制是与它俩的具体实现有关。

总的来说，不管一个事务是否成功地被传送了，接收器和发送器均会以每帧一个事务的速度继续它们的数据/缓冲流。这样才能保证数据在时间上的同步。以上描述的一些机制，用于检测、跟踪、和报告被破坏的事务，让设备或它的客户软件可以对这种破坏以合适的方式做出反应。关于设备和应用对此如何反应是不在 USB 说明的范围内的。

4.10.8 为匹配速率而做的缓冲

如果 USB 上有许多不同的时钟在影响着同步数据流，为了使 USB 上的数据流的速率的相互匹配，缓冲是必不可少的。设备上的每个端点都必须有缓存。主机上的客户软件也必须拥有缓存，数据在被传送之前可以存放在缓存中。如果已知设备的数据速率，需传的数据包的最大长度就是可计算的，图 4-17 给出了几个方程，用于决定设备和主机上的缓存大小以及在给定的数据速率下可能出现的最大包长。

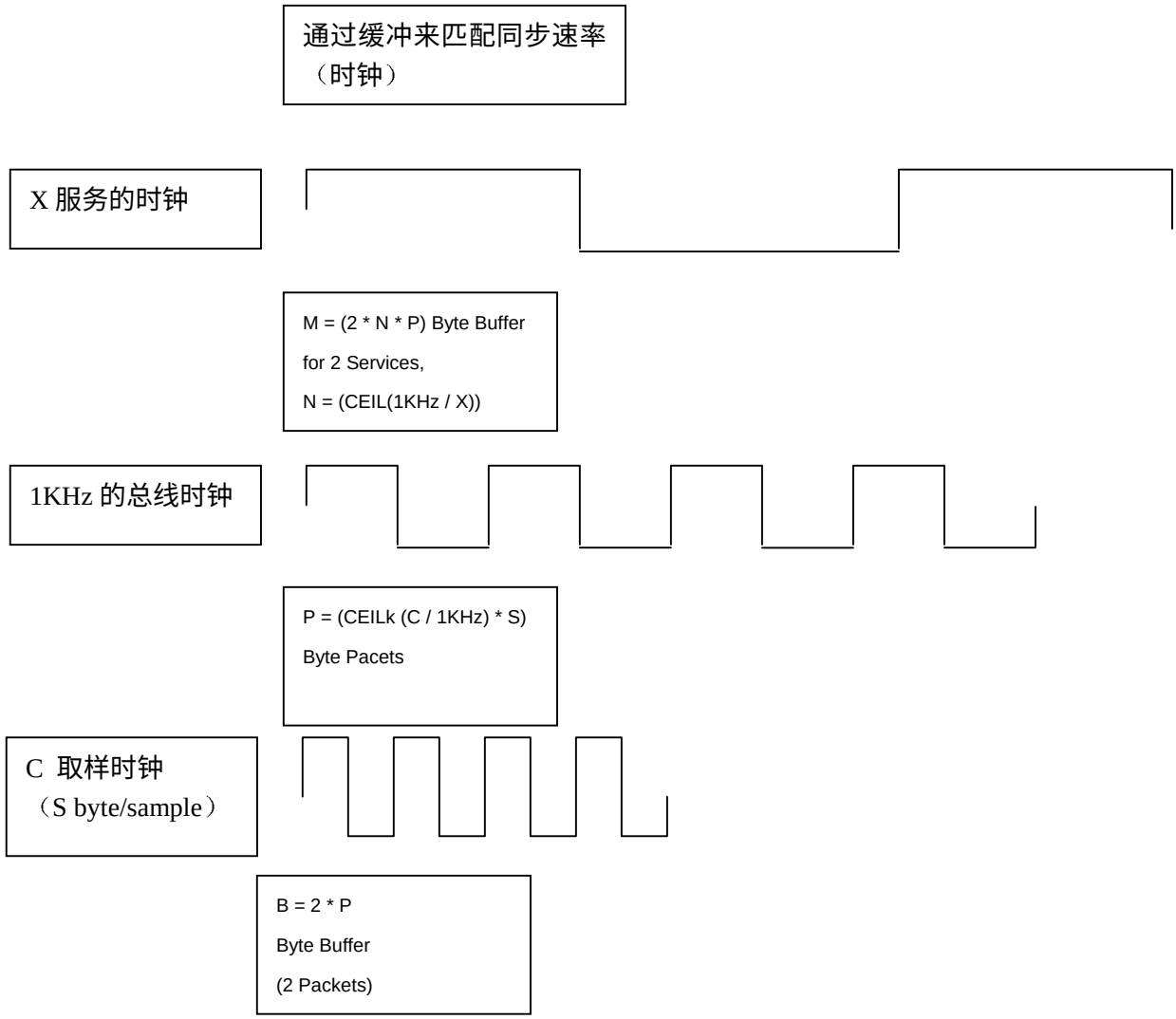


图 4-17 进行速率匹配的同步传送的包长和缓冲大小的计算

设备和客户软件利用这些方程来设计服务时钟速率(变量 x)，取样时钟速率(变量 C)，和取样的长度(变量 S)。USB 只允许一个总线时钟对应一个事务。这些方程提供的信息可成为端

点选择合适的包长，或信息/端点和它的客户软件选择合的缓存的依据。图 4-14 就是了一个典型的同步例子的实际缓存值、包长、时钟值。

USB 数据模式认为设备有自己的取样的大小和取样的速率。USB 支持包的传送，这个包应当是取样长度的整数倍，这样有利于对总线上被损坏的同步事务的错误恢复。如果一个设备没有固有的取样长度或它的取样长度超过了一个包的长度，它应将它的取样长度描述为一字节。如果一个取样被分为几个包，当任一个包被丢掉后，对错误的恢复比一般情况要困难。在有些情况中，数据同步会丢失，除非接收器知道取样的各个部分将在帧号为几的帧中被传。而且，如果由于时钟校正造成取样的个数发生变化(例如一个 non-derived 的设备时钟)，将很难知道取样的一个部分何时被传，因此 USB 不会将一个取样分几个包传。

USB 技术

——USB 的电气特性(之六)

南京大学计算机系　　周玉军

USB(Universal Serial Bus，即通用串行总线)的电气特性主要是对信号的发送及电压分布情况的描述。下面我们将分别对其进行详细介绍，首先来看看其信号的发送。

一、信号的发送

USB 通常使用一种差分的输出驱动器来控制数据信号在 USB 电缆上的发送，在了解具体的信号发送之前，我们先来谈谈有关 USB 设备的特性。

(一)USB 驱动器的特性及其使用

一个 USB 设备端的连接器是由 D+、D-及 Vbus，GND 和其它数据线构成的简短连续电路，并要求连接器上有电缆屏蔽，以免设备在使用过程中被损坏。它有两种工作 7 状态，即低态和高态。在低态时，驱动器的静态输出端的工作电压 Vol 变动范围为 0～０．3V，且接有一个 15kΩ的接地负载。处于差分的高态和低态之间的输出电压变动应尽量保持平衡，以能很好地减小信号的扭曲变形。

在任何驱动状态下，USB 设备必须能接收如图 1 所示的波形。这些波形从一个输出阻抗为 3PΩ的电压源直接进入每一个 USB 数据口。

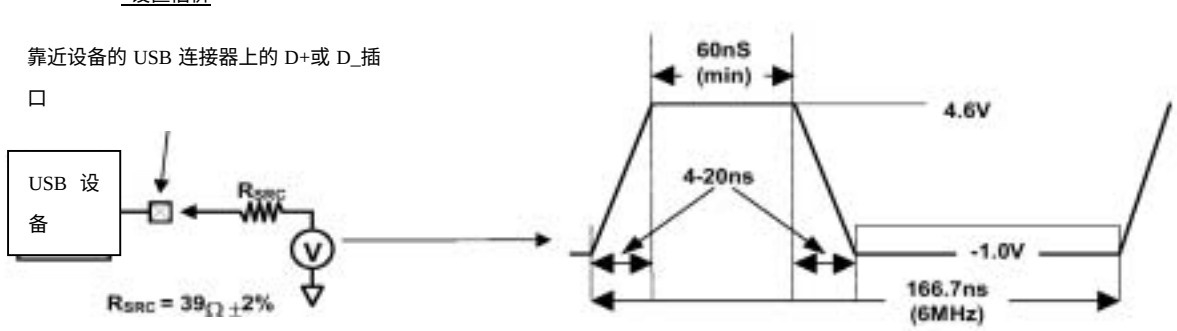


图 1　USB 信号发送的最大输出波形

下面我们将分别对 USB 高速驱动器和低速驱动器的特性作一个介绍.

高速驱动器特性

一个高速 USB 设备的连接是通过阻抗为 90Ω ±15%，最大单路时延为 26ns 的屏蔽双绞线电缆进行的，其到达的最大速率为 12Mb/s，并且每个驱动器的阻抗必须在 28Ω～44Ω之间。图 2 描述了高速驱动器的信号波形。

低速驱动器特性

一个低速 USB 设备在插口端必须要有一个带有串行 A 口连接器的可控制电缆，其速率为 1.5Mb/s。当电缆与设备相连时，在 D+/D-线上必须要有一个 200～450PF 的单终端电容器。低速电缆的传播时延必须小于 18ns，从而保证信号响在其上升沿或下降沿的第一个中点处产生，以允许电缆与一块电容器相连。图 3 列出了低速驱器的信号波形。

图 4 和图 5 分别列出了高速和低速 USB 设备在集线器的终端位置及其所连的功能设备。从图我们可以看出在电缆的下形端的电阻 Rpu 在两图中的连接位置是不同的：

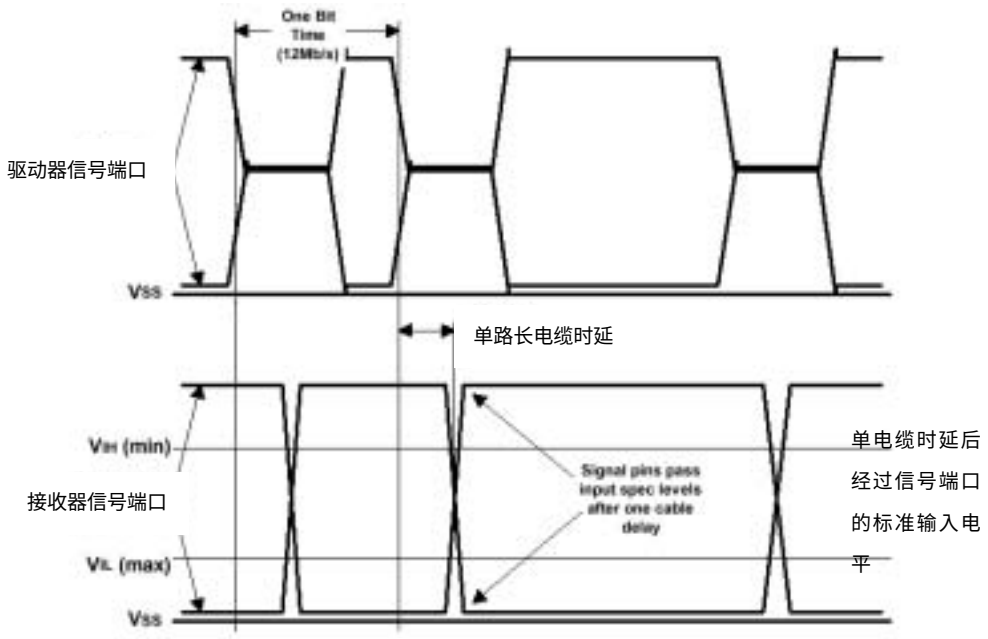


图 2　高速信号波形

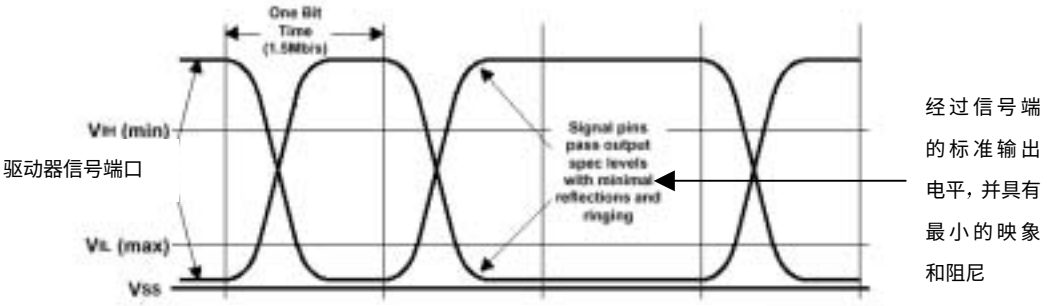


图 3 低速驱动器信号波形

- 图 4 高速设备中的 Rpu 电阻是接在 D+线上的。
- 图 5 低速设备中的 Rpu 电阻是接在 D-线上的。

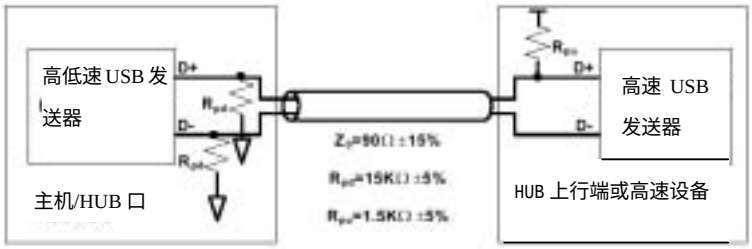


图 4　高速设备电缆和电阻连接

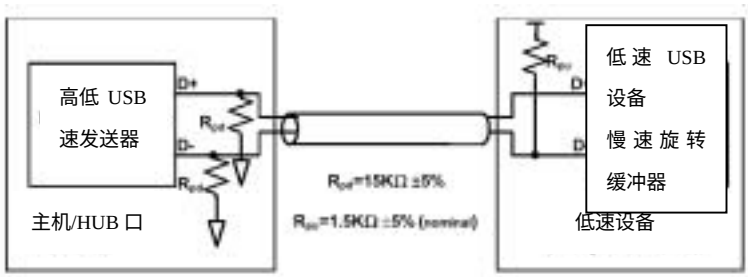


图 5 低速设备电缆和电阻连接

• 下形端口处的 Rpu 电阻是与地相连的，其电阻为 15KΩ±5%。

这个 Rpu 电阻的选取要满足一定的条件，为了在一个复位操作结束后方便地确定可被执行的总线状态，那么选取 Rpu 时要能使 D+/D-线上的电压在 2.5us 的最大复位松驰时间内可在 0～Vih 内自由变动。为了满足这一条件，带有可分电缆的设备必须使用加载电压在 3.0～3.6U 间阻抗为 1.5KΩ±5%的电阻；而具有可控电缆的设备可以使用两种方法中的任一种。注意：终端电阻不包括主机/HUB 上的 15KΩ±5%的电阻。

所有集线器和高速的功能设备上形端口(朝主机方向的)必须使用高速的驱动器，上形集线器端口既可以高速又可以低速来传送数据，但是在信号发送时总是使用高速和边缘速率。低速数据的传输不改变驱动器的特性，低速设备的上形端口必须使用低速驱动器。

所有集线器(包括主机的)外部下形端口必须能适用于两种特性的驱动器，也就是说，任何类型的设备都能被插入这些端口中。当收发器工作在高速模式时，它使用高速和边缘速率来进行信号的发送；工作在低速时，它使用低速和边缘速率来发送数据。

(二)接收器特性

一个差分输入接收器用来接收 USB 数据信号，当两个差分数据输入处在共同的 0.8～2.5V 的差分模式范围时，如图 6 所示，接收器必须具有至少 200mv 的输入灵敏度。

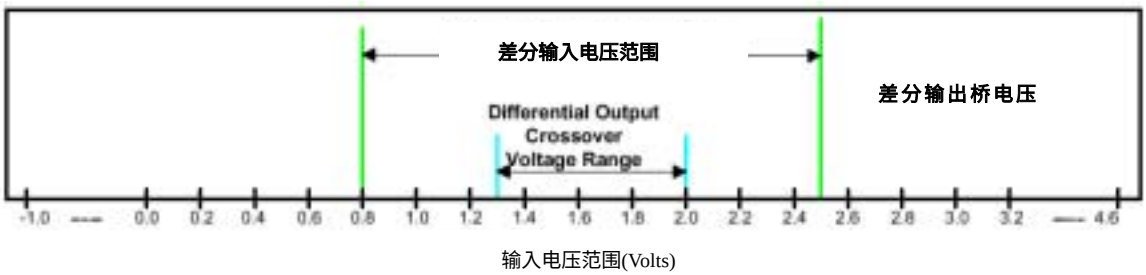


图 6　差分输入感抗范围

除了差分接收器外，还必须有为两个数据线中任一个所用的单终端接收器，此时该接收器的合并磁滞现象可以减小它们对噪声的灵敏度。

在差分信号传送期间，D+和 D-线上的电压可以小于 Vih。对于高速传送而言，这个阶段可以延续到 14ns；对于低速传送，可延续到 I/V ns 之久。接收器的逻辑设备用于保证这种情况不会被当作 SE0 态来处理。

(三)输入特性

没有终端的 D+或 D-的输入阻抗必须大于 300KΩ，一个端口的输入电容量在连结器的端口处量得。上形和下形端口可以有不同值的电容，一个集线器或主机的下形端口所允许的 D+或 D-上的最大电容量(差分的或单终端的)为 150pF；带有可分电缆的高速设备的上形端口所允许的 D+或 D-上的最大电容量为 100pF。

对于有可控电缆的高速设备，它本身在 D+和 D-可以有最大电容量为 75pF 的接地电容器，其中电缆为其余的输入电容使用。在该种设备中，单终端输入电容必须与终端所使用的一致，该终端必须能在 2.5us 内控制 D+或 D-线上的电压在 0～Vih 范围内变动。D+/D-上的电容包括设备单终端输入电容和主机/HUB 的 150pF 的输入电容。

上面我们介绍了 USB 驱动器和接收器以及输入特性，下面我们将介绍有关 USB 的信号发送情况。首先介绍信号的发送标准。

(一)信号的发送标准

表 7-1 总的概括了 USB 信号的发送标准。(表 7-1)在该表中，J 和 K 这两个数据态是两个逻辑电平，在系统中，通常被用来进行交换差分数据。差分数据信号的发送并不关心信号经过处的电平情况，它只要求桥电压在 1.3～2.0 之间。另外，在接收端，空闲态和工作态在逻辑上分别与 J 态和 K 态等价。

一般而言，数据，空闲及唤醒信号的发送标准均由端口的设备类型所决定。如果连结的是高速设备，则 USB 使用所规定的高速率来发送信号 1 并且有很快的上升沿和下降沿时间 1，甚至还可用低速率来发送数据，而对于表 7-1 中所示的低速信号发送标准仅用在低速设备与其所连接的端口之间(上升沿和下降沿时间较长)。

(二)连结与中断信号的发送

USB 设备是一个智能型的设备，当它发现主机或集线器的下形端口上没有设备连接时，存在的 Rpu 电阻将使 D+和 D-上的电压低于主机或集线器端口的单终端电压，此时该端口不是由集线器控制的，这将在下形端口产生一个 SE0 态。如果主机或集线器不在控制数据线并且下形端口的 SEO 态的持续时间超过 2.5ns，则此时 USB 设备将中断信号的发送。

如果集线器发现其中一根数据线上的电压大于它的临界值的持续时间超过 2.5us，则便开始信号的发送。

在介绍了有关信号发送的相关情况后，我们将分别对各种信号的发送进行讨论。

(一)数据信号的发送

通过差分信号来实现数据包的传送。

通过控制 D+和 D-线从空闲态到相反的逻辑电平(K 态)，就可以实现源端口的包发送(SOP)。同步字中的第一位代表了这种在电平上的转换。当它的重新发送时间低于±5ns 时，集线器必须对 SOP 中第一位的宽度变化有所限制。可以通过使用具有延迟输出使能的集线器来实现数据的匹配，这样可以使数据失真减小到最小。

SE0 态通常用来表示包的发送结束(EOP)，可以通过控制 D+和 D-两位时到达 SEO 态，然后控制 D+和 D-线一位时后到达 J 态，就可实现 EOP 信号的发送。从 SEO 态到 J 态的变化表示接收端包发送的结束。J 态持续一个位时，然后 D+和 D-上的输出驱动器均处于高阻抗状态，总线尾端的电阻此时控制总线处于空闲态。图 7 列出了包开始和结束的信号发送波形。

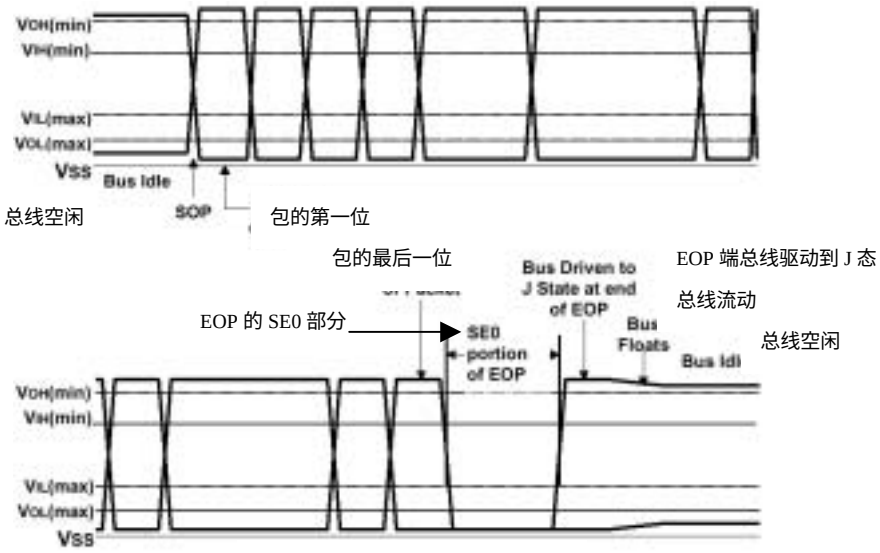


图 7　包电压电平

(二)复位信号的发送

复位就是将一个信号从挂起态唤醒。

集线器信号通过控制端口上的持久的 SE0 态来实现对下形端口的复位。复位信号清除后，设备都将处于缺省状态。

表 7-1 信号电平

总线状态	信　号　电　平		
	开始端的源连接器（一位时端）	终端的目标连接器	
		需要条件	接受条件
差分的“1”	D+>Voh(min) D-<Vol(max)	(D+)-(D-)>200mv (D+)>Vih(min)	(D+)-(D-)>200mv
差分的“0”	D->Voh(min) D+<Vol(max)	(D+)-(D+)>200mv D->Vih(min)	(D-)-(D+)>200mv
单终端“0”（SE0）	D+和 D-<Vol(max)	D+和 D-<Vil(max)	D+和 D-<Vih(min)
数据 J 态： 高速 低速	 差分的“0” 差分的“1”	 差分的“0” 差分的“1”	
数据 K 态： 高速 低速	 差分的“1” 差分的“0”	 差分的“1” 差分的“0	
空闲状态： 高速 低速	N.A.	 D->Vihz(min) D+>Vil(max) D+>Vihz(min) D-<Vil(max)	 D->Vihz(min) D+<Vih(min) D+>Vihz(min) D-<Vih(min)
唤醒状态	数据 K 状态	数据 K 状态	
包开始（SOP）	数据线从空闲态转到 K 态		
包结束（EOP）[4]	SE0 近似地为 2 位时 [1]	SE0≥1 位时 [2] 其后仅接	SE0≥1 位时 [2] 其后仅接

	其后仅接着 1 位时的 J[3]	着一位时的 J 态	着 J 态
段开连接（在下行端口处）	N.A.	SE0 持续时间大于等于 2.5 微秒	
连接（在上行端口处）	N.A.	空闲态持续时间大于等于 2 毫秒	空闲态持续时间大于等于 2.5 微秒
复位	D+和 D-小于 Vol(max)的持续时间大于等于 10 毫秒	D+和 D-小于 Vil(max)的持续时间大于等于 10 毫秒	D+和 D-小于 Vil(max)的持续时间大于等于 2.5 微秒

注释 1：以位时定义的 EOP 宽度与传送的速度有关。(标准的 EOP 宽度都在表 7-5 和表 7-6 中列出)

注释 2：以位时定义的 EOP 宽度与接收 EOP 的设备类型有关.位时是近似的.

注释 3:仅跟在 EOP 后的 J 态的宽度以位时来衡量,它与缓冲器的边缘速率有关.来自低速缓冲器的 J 态 必须要有低速的位时宽,来自高速的,则必须要有高速的位时宽.

注释 4:始终处于活动态的是低速的 EOP

根据 USB 系统软件的需求，复位信号可在任一个集线器或主机的控制端口产生，该复位信号的最小持续时间为 10ms。复位后，集线器端口将处于能动状态。USB 系统软件和主机控制器必须确保发送到根端口的复位信号持续时间足够长以便通知当前正试图进行唤醒操作的各下形设备。根端口产生的复位信号的持续时间应为 50ms，但并不要求它一直是延续的。然而，如果复位信号不是连续的，则各间断的复位信号间的时间间隔应小于 3ms。

一个设备如果见其上形端口的 SE0 态持续时间超过 2.5us，则它就把该信号作为复位信号处理。在复位信号发送结束前，它必须已产生作用。

当端口处于使能状态后，集线器将传播一个活动信号到新的复位端口。连在该端口的设备必须能识别总线的活动性，并要能防止被挂起。

在复位信号清除后的 10ms 的复位恢复时间后，集线器必须能接收所有集线器请求，设备也必须能接收一个 SetAddress()请求。如果接收这些请求失败，则设备将不能被 USB 系统软件所识别。

(三)挂起

所有的设备都必须能支持挂起状态，并可从任一电平状态进入挂起态。当设备发现它们的上形总线上的空闲态持续时间超 3.0ms 时，它们便进入挂起态。当设备的所有端口上的总线不活动时间不超过 10ms 后，设备必须被真正的挂起，此时它仅从总线上获得挂起电流。如果任一其它总线交通缺乏时，SOF 令牌将在每帧中出现一次，以防止高速设备被挂起。当任一低速设备交通缺乏时，在 SOF 令牌出现的每一帧中至少有一个低速设备处于活动态，以避免它们不被挂起。

当处在挂起状态时，设备必须继续为它的 D+(高速)或 D-(低速)上的 Rpu 电阻提供电压从而维持一个空闲态，这样上形集线器才能为设备维持正确的连结状态。

挂起又可分为全局挂起和局部挂起。

• 全局挂起

当在总线的任何地方没有通信需要时，就要用到全局挂起，此时所有总线都处在挂起状态。主机通过中止它所有的传送(包括 SOF 令牌)来发送开始全局挂起信号。当总上的每个设备识别总线的空闲态持续适当时间时，它将进入挂起状态。

• 局部挂起

可以通过向集线器端口发送 SetPortFeature(PORT-SUSPEND)来使与其相连的总线部分被挂起，此时处于那部分的设备经过上面所说的适当时延后进入挂起状态。

(四)唤醒

处在挂起状态的设备，当它的上形端口接收到任一非空闲信号时，它的操作将被唤醒。特别地，如果设备的远程唤醒功能被 USB 系统软件开启时，它将自动发信号给系统来唤醒操作。唤醒信号由主机或设备使用，以使一个挂起的总线段回到活动态。集线器在唤醒信号的生成和传播中起了十分重要的作用。设备唤醒时总有一个先后次序，我们将在后面详细介绍。

USB 系统软件必须提供 10ms 的唤醒恢复时间，在这段时间内，它将不对与被唤醒的部分总线相连的任一设备进行操作。

端口的中断与连接也可以使集线器发送一个复位信号，从而唤醒系统，但仅当集线器具有远程唤醒使能时，这些事件才能引起集线器发送唤醒信号。

上面介绍了有关几种信号的发送情况，下面将对数据信号的发送做一个详细的讨论。

(一)数据的编码与解码

在包传送时，USB 使用一种 NRII(None Return Zero Invert，即无回零反向码)编码方案。在该编码方案中，“1”表示电平不变，“0”表示电平改变。图 8 列出了一个数据流及其它的 NRII 编码，在该图的第二个波形图中，一开始的高电平表示数据线上的 J 态，后面就是 NRII 编码。

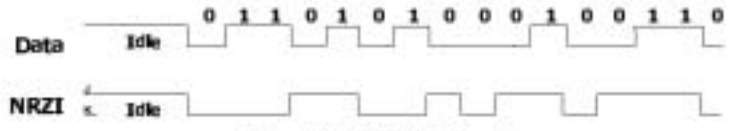


图 8　NRZI 数据编码

为了确集信号发送的准确性，当在 USB 上发送一个包时，传送设备就要进行位插入操作。所谓位插入操作是指在数据被编码前，在数据流中每六个连续的‘1’后插入一个‘0’，从而强迫 NRII 码发生变化，如图 9 所示。

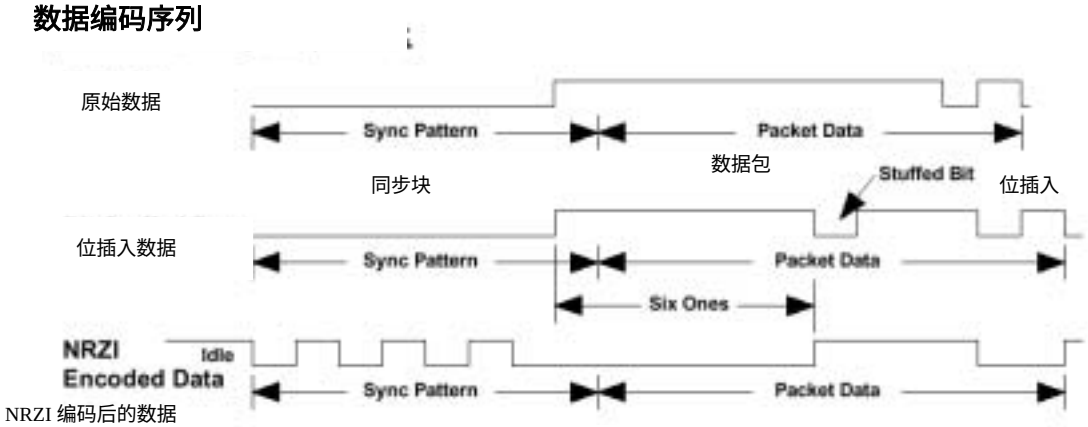


图 9　位插入

位插入操作从同步格式(如图 10 所示)开始，贯穿于整个传送过程，在同步格式端的数

据‘1’作为真正数据流的第一位。位插入操作是由传送端强制执行的，是没有例外的。如果严格遵守位插入规则，甚至在 EOP 信号结束前也要插入一位‘0’位。

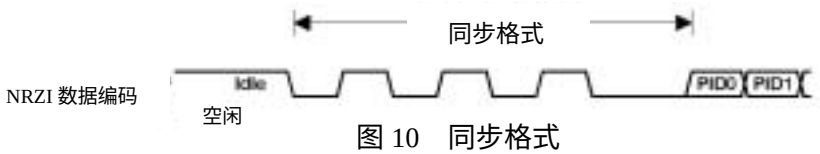


图 10　同步格式

接收端必须能对 NRZI 数据进行解码，识别插入位并去掉它们。如果接收端发现包中任一处有七个连续的“1”，则将会产生一个位插入错误，该数据包将被忽略。

关于位的插入有一个特例，那就是刚好在 EOP 前的时间间隔，EOP 前的最后一个数据位可能被集线器的转换偏移而拉长，这种情况如图 11 所示。

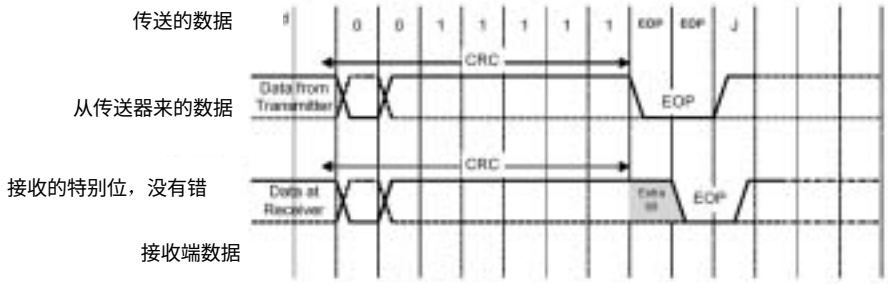


图 11 对 EOP 前的特别位的说明

(二)数据信号的发送速率

高速数据发送率通常为 12.000Mb/s，主机，集线器和高速设备的数据率误差为±0.25%(2.500ppm)。集线器控制器的数据率应该准确地知道，其误差最好控制在±0.05%(500ppm)内。

低速数据发送率为 1.50Mb/s，低速功能设备所允许的误差为±1.5%(15000ppm)。

以上所述的误差，主要由下面的几种情况所引起:

- 负载电容量的影响
- 振荡器上电压供应的稳定性影响
- 温度的影响
- 器件的老化

(三)数据源的抖动

在数据发送的边缘时间内，数据源可能发生一些变化(即抖动)。处在任何数据变化集间的时间为 N*Tperiod±抖动时间，其中 N 为发生变化的位数，Tperiod 为具有一定范围的数据率的实际时间段。数据抖动的测量与计算最大上升沿和下降沿时所用的负载相同，并且它们在数据线的交叉点处进行测量，如图 12。

- 对于高速传送，任何连续的差分数据变化的抖动时间为必须在±20ns 内，对于任何一个成对出现的差分数据变化(Jk 到下一个 Jk 的变化或 kJ 到下一个 kJ 的变化)的抖动时间必须在 1.0ns 内。
- 对于低速传送，任何连续的差分数据变化的抖动时间必须在 25ns 内，而任一成对出现搓 分数据变化的抖动时间必须在±10ns 内。

这些抖动的现象包括时间的变化，主要归咎于差分缓冲器的延迟和上升沿及下降沿时间的不匹配，内部时钟抖动，噪声及其他随机因素的影响。

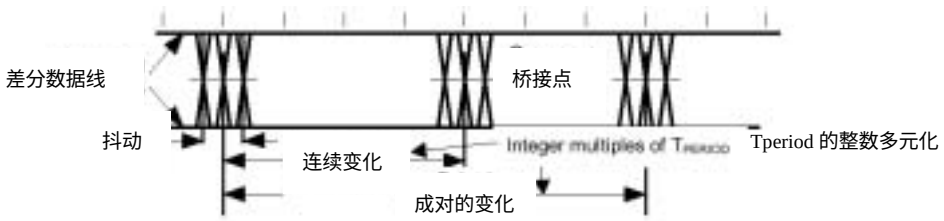


图 12 数据抖动分类

(四)接收端数据的抖动

当抖动存在时，任何设备类型的数据接收必须能正确地对差分数据进行解码。这种情况的抖动可能是由上面所说的时延不匹配所引起，也可能是由源端和目标端数据速率的不匹配所引起。在特定的应用中，只要抖动条件满足，输出驱动器的抖动可能对设备时钟的精确性产生影响。

(五)电缆的延迟

USB 中传送信号的电缆所允许的时延为 26ns，对于一个标准的 USB 可分电缆，其时延由从串行 A 口连接器端到串行 B 口连结端计算而得，并且其值小于 26ns；而对于其它电缆，其时延由从串行 A 口连结器端到该电缆所连设备端计算而得。

(六)电缆的信号衰减

对于进行高速信号发送的每根电缆而言，信号对(D+，D-)所允许的最大衰减量如下表 7-2 所示。

表 7-2 信号时延

频率（ MHZ）	衰减量（最大）dB/每根电缆
0.064	0.08
0.256	0.11
0.512	0.13
0.772	0.15
1.000	0.20
4.000	0.39
8.000	0.57
12.000	0.67
24.000	0.95
48.000	1.35
96.000	1.9

二、电压分布

所有 USB 设备的缺省电压为低电压，当设备要从低电压变化到高电压时，则是由软件来控制的。在允许设备达到高电压之前，软件必须保证有足够的电压可供使用。

USB 支持一定范围的电压来源和电压消耗供应者，包括如下的部分。

• 根端口集线器：它是直接与 USB 主机控制器相连的，并与其相同的电源来源。从外部

获得操作电压(AC 或 DC)的系统，在每个端口至少支持五个单位负载，这些端口称为高电压端口。由电池组提供电压的系统可以支持一个或五个单位负载。哪些只能支持一个单位负载的端口称为低电压端口。

• 从总线获得电压的集线器：它的所有内部功能设备和下形端口都从它的上形端口的 Vbus 上获得电压。在电压升高时，它可以接一个单位负载，经过初始设置后，它可以接五个单位负载。初始设置电压被分配给了集线器，任一固定功能设备和外部端口。它的外部端口只能接一个单位负载，当集线器处于活动或挂起态时，它必须为这个端口提供电流。该种集线器如图 13 所示。

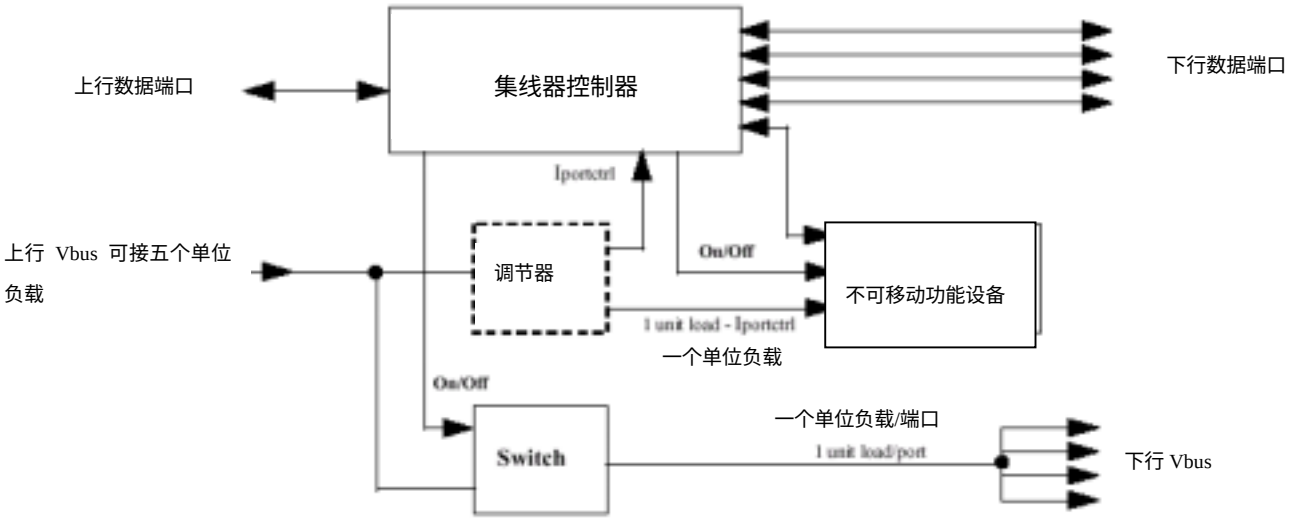


图 13　总线提供电压的集线器

• 自给电压集线器：如图 14 所示，它的任一内部功能设备和下形端口不再从 Vbus 上获得电压，但当它的其余部分电压下降时，它的 USB 接口可接一个单位负载并从 Vbus 处获得电压，以允许该接口能工作。从外部(从 USB)获得操作电压的集线器，可在每个端口接五个单位负载。由电池组提供电压的集线器，每端口可接一个或五个单位负载。

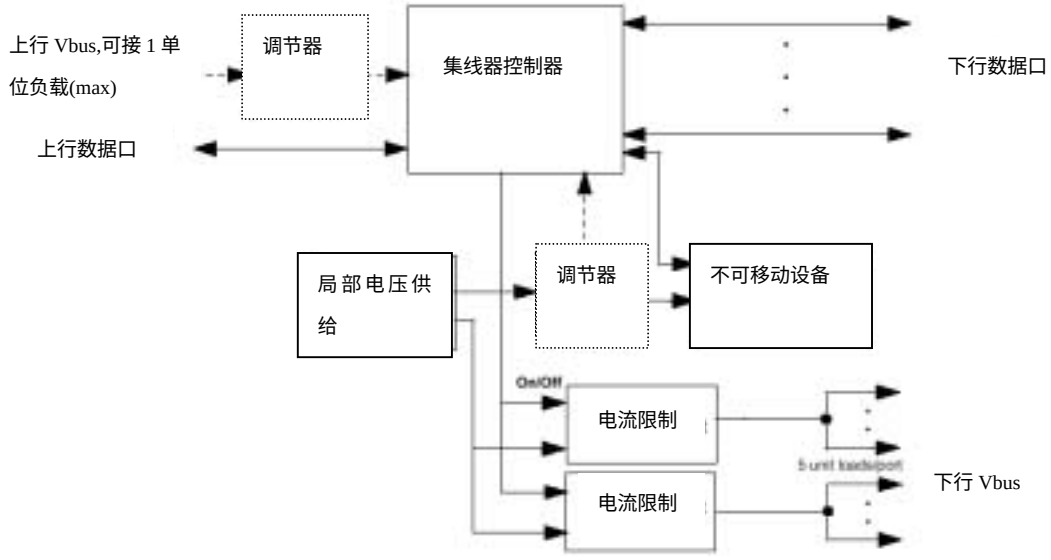


图 14　复合的自己电压集线器

• 从总线获得电压的低电压功能设备，如图 13，该种设备上的所有电压均来自 Vbus，在任

一时刻，它们最多只能接一个单位负载。

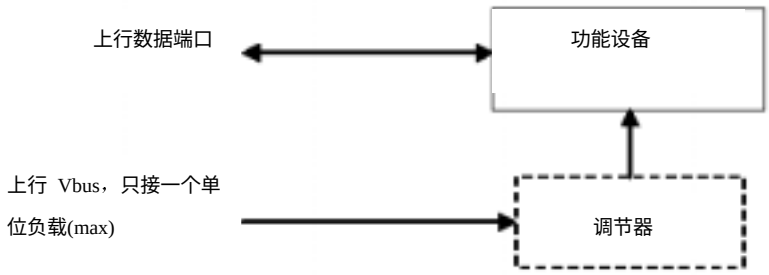


图 15　总线提供的低电压功能设备

• 从总线获得电压的高电压设备：如图 16 所示，该种设备上的所需电压均来自 Vbus。在电压升高时，它们至多只能接一个单位负载，但当初始设置后，可接五个单位负载。

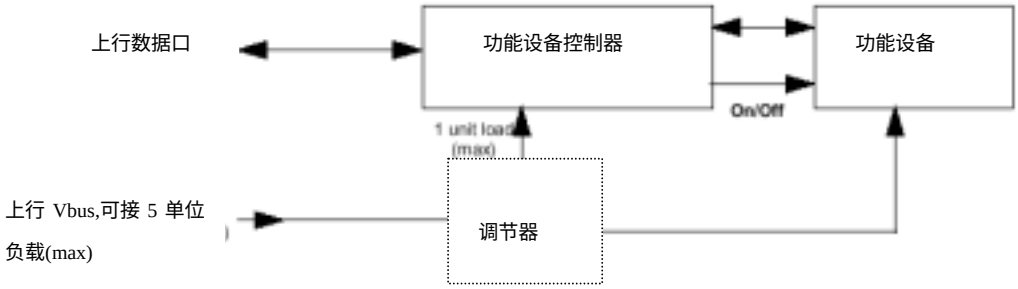


图 16　总线提供的高电压功能设备

• 自给电压功能设备，如图 17，当它的其余设备电压下降时，它可以接一个单位负载，并从 Vbus　上获取所需电压，以使 USB 接口处于活动状态。

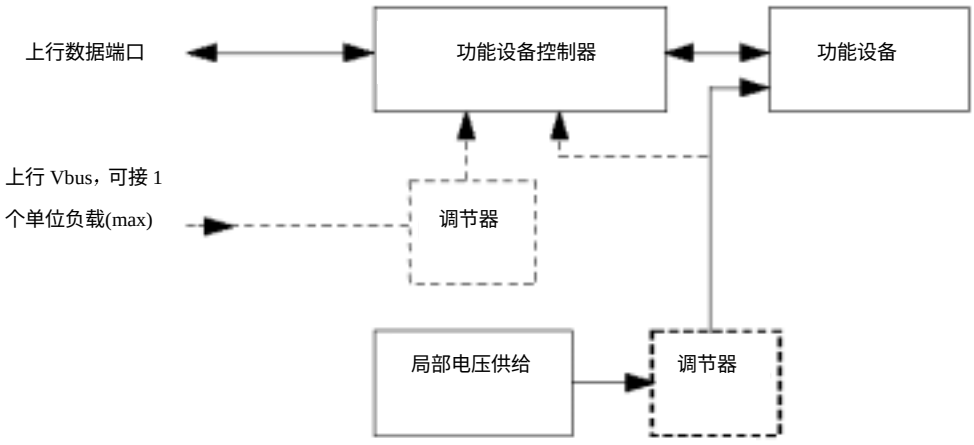


图 17　自给电压功能设备

在前面，我们已经讲过 HUB 的挂起与唤醒，下面我们来谈谈在这两种情况下的电压分布情况。

低电压设备或高电压设备工作低电压下时，它们所允许的挂起电流限制为 500uA，如果一个设备被初始设置为高电压并且具有远程唤醒功能，则在挂起期间，它的电流可达到

2.5mA.

当一个集线器处在挂起状态时，它必须仍能为每个端口提供最大电流值。对于具有远程唤功能的设备，当它的电压在升高而系统的其余部分仍处于挂起态时，上面的要求是十分必要的。

当设备被唤醒时(远程唤醒或由唤醒信号唤醒)，它们此时必须能限制 Vbus 上的流入电压，集线器 Vbus 所允许的最大电压落差为 330mV。设备必须有足够的分流电容器或要有一个可控制的电压打开顺序，以便当设备正在被唤醒的任一时间内，从集线器来的电流不能超过端口的最大电流允许值。

最后，我们来谈谈设备的动态加载与卸载。

插入或拔掉一个集线器或其它功能设备时，不应影响网络中其余设备的正常工作为前提。卸载掉一个功能设备将中止设备与主机间的通信，此时集线器向主机警告该端口已被中断。

设备从网络中卸去时，电缆的电感系统将在设备电缆的开口端产生一个很大的回流电压，它是没有破坏性的。但在电缆设备的末端必须有一些小容量的电容器，以保证产生的回流电压不会引起设备端电压极性的改变。但回流电压会产生噪音，通常利用分流电容器进行适当分流以减少噪音，分流电容器对回流电压及其产生的噪声进行缓和。

动态加载某设备可能会产生强电流，因而会使 HUB 上的 Vbus 低于它的最小工作电压，因此必须引用一些限流装置。

在动态加载期间，通过使连结器上的信号端口处于空闲，以使其免受强电流的破坏，这样为了使电压端口首先进行联系。这就保证，在信号端口连接前，分布在下形设备上的电压是可利用的。另外，在连接期间，信号线均处于高阻抗状态，为了使标准信号线上此时没有电流流动。

第八章 协议层

这章从字段（Field）和包（Packet）的定义开始，从底向上地展示USB（Univeral Serial Bus）协议。接着是对不同事务（Transaction）类型的包事务格式的描述。然后是链路层（Link layer）流程控制（Flow Control）和事务级别的故障恢复（Fault recovery）。本章的最后将讨论复执同步化（Retry synchronization），超时干扰（Babble）和总线活动丧失（Loss of bus activity）的恢复。

8.1 位定序

数据位被发送到总线的时候，首先最低有效位（LSb），跟着是下一个最低有效位，最后是最高有效位（MSb）。在以后图表中的，包以下列形式给出，即包中单个的位和字段从左到右的顺序就是它们通过总线的顺序。

8.2同步字段

所有的包都从同步（SYNC）字段开始的，同步字段是产生最大的边缘转换密度（Edge TransitionDensity）的编码序列。同步字段作为空闲状态出现在总线上，后面跟着以NRZI编码的二进制串“KJKJKJKK”。通过被定义为8位长的二进制串，输入电路以本地时钟对齐输入数据。同步字段是用于同步的机制，在以后图表当中将不被表示（参照节7.1.10）。同步字段里的最后的2位是同步字段结束的记号，并且标志了包标识符（PID，Packet Identifer）的开始。

8.3包字段格式

在后面几节将描述标记，数据和握手包的字段格式。包中位的定义是以未编码的数据格式给出。为了清楚起见，在此不考虑NRZI编码和位填充（Bit Stuffing）的影响。所有的包都分别有包开始（Start-of-Packet）和包结束（End-of-Packet）分隔符。包开始（SOP）分隔符是同步字段的一部分，而包结束（EOP）分隔符在第7章有所描述。

8.3.1包标识符字段

所有USB包的同步字段后都紧跟着包标识符（PID）。如图8-1所示，包标识符由4位的包类型字段和其后的4位的校验字段构成。包标识符指出了包的类型，并由此隐含地指出了包的格式和包上所用错误检测的类型。包标识符的4位的校验字段可以保证包标识符译码的可靠性，这样包的余项也就能被正确地解释。包标识符的校验字段通过对包类型字段的二进制的求反码产生的。如果4 个PID检验位不是它们的各自的包标识符位的补，则说明存在PID错。

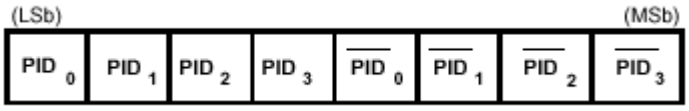


图8-1 PID 格式

主机和所有功能部件都必须对得到全部PID字段实行完整的译码。任何收到包标识符，如果含有失败的校验字段，或者经译码得到未定义的值，则该包标识符被假定是被损坏的，而且包的余项将被包接收机忽略（Ignore）。如果一个功能部件（Function）收到了包含了它所不支持的事务类型或方向的合法包标识符，则不必应答。例如，只能输入的端口（IN-only Endpoint）必须忽略输出标记（Token）。表8-1列出了包标识符类型，编码及其描述。

表8-1 PID 类型

PID 类型	PID 名	PID[3:0]	描述
标记（Token）	输出（OUT）	0001B	在主机到功能部件的事务中有地址+端口号
	输入（IN）	1001B	在功能部件到主机的事务中有地址+端口号
	帧开始（SOF）	0101B	帧开始标记和帧号
	建立（SETUP）	1101B	在主机到功能部件建立一个控制管道的事务中有地址+端口号
数据（DATA）	数据0（DATA0）	0011B	偶数据包PID
	数据1（DATA1）	1011B	奇数据包PID
握手（Handshake）	确认（ACK）	0010B	接收器收到无措数据包；
	不确认（NAK）	1010B	接收设备部不能接收数据，或发送设备不能发送数据；
	停止（STALL）	1110B	端口挂起，或一个控制管道请求不被支持。
专用（Special）	前同步（PRE）	1100B	主机发送的前同步字。打开到低速设备的下行总线通信。

*注解：PID位以最高位在前的顺序被表示。在USB上被发送的时候，最右的位（位0）将被第一个发出。

包标识符被分为4个编码组：标记，数据，握手和专用。包标识符传送的前2位（PID<0 ： 1>）指出了其属于哪个组。这说明包标识符编码的分布。

8.3.2地址字段

功能部件端口使用2个字段：功能部件地址字段和端口字段。功能部件对地址和端口字段都需要进行译码。不允许使用地址或端口别名（Aliasing），并且任何一个字段不匹配，此标记都必须被忽略。另外， 对未初始化的端口的访问将使得标记被忽略。

8.3.2.1地址字段

功能部件地址（ADDR）字段通过其地址指定功能部件，至于是数据包的发出地还是目的地，则取决于标记PID的值。如图8-2所示，ADDR<6 ： 0>指定了总共128个地址。地址字段被用于输入，建立和输出标记。由定义可知，每个ADDR值都定义了单一的功能部件。刚一复位（Reset）和加电（Power-up）的时候，功能部件的地址默认值为零，并且必须由主机在枚举过程（Enumeration Process）中编程。功能部件地址零被用作为缺省地址，不可被分配作任何别的用途。

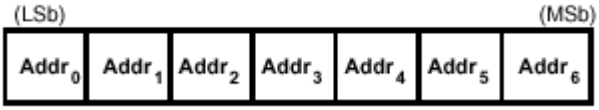


图8-2 地址字段

8.3.2.2端口字段

如图8-3所示，附加的4位的端口（ENDP）字段在功能部件需要一个以上端口时候允许更灵活的寻址。除了端口地址0之外，端口个数是由功能部件决定的。端口字段只对输入，建立和输出标记PID有定义。所有的功能部件都必须在端口0提供一个控制管道（缺省控制管道）。对于低速(Low Speed)设备，每个功能部件最多提供3个管道：在端口0的控制管道加上2个附加管道（或是2个控制管道，或是1个控制管道和1个中断端口，或是2个中断端口）。全速(Full Speed)功能部件可以支持最多可达16个的任何类型的端口。

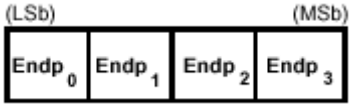


图8-3 端口字段

8.3.3帧号字段

帧号字段是一个11位的字段，主机每过一帧就将其内容加一。帧号字段达到其最大值7FFH时归零，且它仅每个帧最初时刻在SOF标记中被发送。

8.3.4数据字段

数据字段可以在0到1,023字节之间变动，但必须是整数个字节。图8-4为多字节显示格式。每个字节的范围内的数据位移出时都是最低位（LSb）在前。

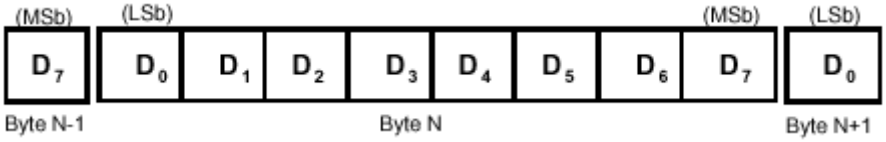


图8-4 数据字段格式

如同在第5章所描述的那样，数据包大小随着传送类型而变化。

8.3.5循环冗余校验

循环冗余校验（CRC）被用来在标记和数据包中保护所有的非PID字段。在上下文中，这些字段被认为是保护字段。PID不在含有CRC的包的CRC校验范围内。在位填充之前，在发送器中所有的CRC都由它们的各自的字段产生。同样地，在填充位被去除之后，CRC在接收器中被译码。标记和数据包的CRC可100 %判断单位错和双位错。失败的CRC指出了保护字段中至少有一个字段被损坏，并导致接收器忽略那些字段，且在大部分情况下忽略整个包。为了CRC的发生和校检，发生器和检验器里的移位寄存器置成为全1（All-ones）型。对于每个被发送或者被收到的数据位，当前余项的最高一位和数据位进行异或（XOR），然后，余项是左移1位，并且，最低一位置零。如果异或的结果是1，余项和生成多项式作异或。当检查的字段的最后的一位被发送的时候，发生器里的CRC被颠倒，再以最高位（MSb）在前发给检验器。当检验器收到CRC的最后的一位,且不发生错误的时候，余项将等于多项式的的剩余。如果剩余与包接收器中最后计算出的检验和余项（Checksum remainder）不匹配，则存在CRC误差。对于CRC，必须满足位填充的要求，且如果前6位都是1的话，这包括在CRC的最后插入零，。

8.3.5.1标记CRC

标记使用了5位的CRC字段，它覆盖了输入，建立和输出标记的ADDR和ENDP字段，或SOF标记的时间戳字段。生成多项式如下：

$$G(X) = X^5 + X^2 + 1$$

这个多项式的二进制位组合是00101B。如果所有的标记位都被准确无误地收到，接收机中的5位剩余将是01100B。

8.3.5.2数据CRC

数据CRC是作用于数据包的数据字段上的16位多项式。产生的多项式是如下

$$G(X) = X^{16} + X^{15} + X^{2} + 1$$

这个多项式的二进制位组合是1000000000000101B。如果全部的数据和CRC位被准确无误地收到，16位剩余将是1000000000001101B。

8.4包格式

这节展示标记，数据和握手包的包格式。这些图将以位被挪动到总线上的顺序显示包内的字段。

8.4.1标记包

图8-5显示了标记包的字段格式。标记由PID，ADDR和ENDP构成，其中PID指定了包是输入，输出还是建立类型。对于输出和建立事务，地址和端口字段唯一地确定了接下来将收到数据包的端口。对于输入事务的，这些字段唯一地确定了哪个端口应该传送数据包。只有主机能发出标记包。输入PID定义了从功能部件到主机的数据事务。输出和建立PID定义了从主机到功能部件的数据事务。

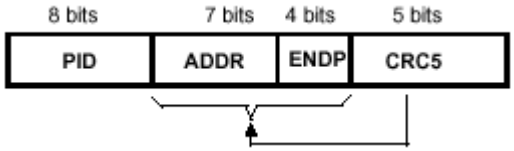


图8-5 标记包格式

如上图所示，标记包包括了覆盖地址和端口字段的5位CRC。CRC并不覆盖PID,因为它有自己的校验字段。标记和帧开始(SOF)包是由3个字节的包字段数据后面的包结束（EOP，End of Packet）界定的。如果包被译码为合法标记或SOF，但却没有在3个字节之后以EOP终止，则它被认为是无效的，并被接收器忽略。

8.4.2 帧开始（SOF,Start-of-Frame）包

主机以每1.00 ms ±0.0005 ms一次的额定速率发出帧开始（SOF）包。如图8-6中所示，SOF包是由指示包类型的PID和其后的11位的帧号字段构成。

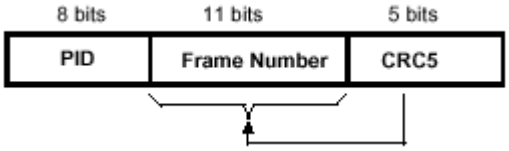


图8-6 帧开始包

SOF标记组成了仅有标记的(token-only)事务，它以相对于每帧的开始精确计算的时间间隔发送SOF记号（Marker）和伴随的帧数。包括集线器的所有全速功能部件都可收到SOF包。SOF标记不会使得接收功能部件产生返回包；因此，不能保证向任何给定的功能部件发送的SOF都能被收到。SOF包发送2个时间调配（Timing）信息。当功能部件探测到SOF的PID的时候，它被告知发生SOF。对帧时间敏感而不需要追踪帧数（例如集线器）的功能部件，仅需对SOF的PID译码 ；可忽略帧数和其CRC。如果功能部件需要追踪帧数，它必须对PID和时间戳都进

行译码。对总线时间调配信息的没有特别需要的全速设备可以忽略SOF包。

8.4.3数据包

如图8-7所示，数据包由PID，包括至少0个字节数据的数据区和CRC构成。有2种类型的数据包，根据不同的PID：DATA0和DATA1来识别。2种数据包PID是为了支持数据切换同步（Data Toggle Synchronization）（在第8.6节提到）而定义的。

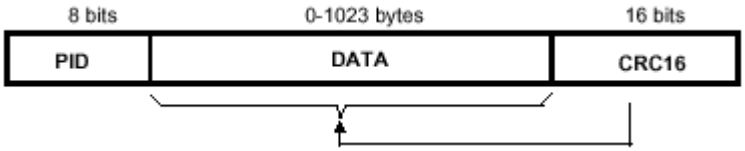


图8-7 数据包格式

数据必须以整数的字节数发出。数据CRC仅通过对包中的数据字段计算而得到，而不包括PID，它有自己的校验字段。

8.4.4握手包

如图8-8所示，握手包仅由PID构成。握手包用来报告数据事务的状态，能还在表示数据成功接收，命令的接收或拒绝，流控制（Flow Control）和停止（Halt）条件。只有支持流控制的事务类型才能返回握手信号。握手总是在事务的握手时相（Phase）中被返回，也可在数据时相代替数据被返回。握手包由1个字节的包字段后的EOP确定界限。如果包被解读为合法的的握手信号，但没有以1个字节后面的EOP终止，则它被认为是无效的，且被接收机忽略。

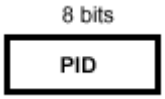


图8-8 握手包

有3种类型的握手包：

- **ACK**表示数据包没有位填充或数据字段上的CRC错，并且数据PID被正确收到。ACK在下列的情况下被发放，当时序位（Sequence Bit）匹配且接收器能接受数据的时候，或者当时序位失配，但发送方和接收器互相之间必须再同步（Resynchronize）（在第8.6节详细提到）的时候。ACK握手信号只适用于数据被传送且期待握手信号的事务中。对于输入事务，ACK由主机返回，而对于输出或建立事务则由功能部件返回；

- **NAK**表示功能部件不会从主机接受数据（对于输出事务），或者功能部件没有传输数据到主机（对于输入事务）。NAK仅由功能部件在输入事务的数据时相返回，或在输出事务的握手时相返回。主机决不能发出NAK。出于流控制的目的，NAK用于表示功能部件暂时不能传输，或者接收数据，但是最终还是能够在不需主机干涉的情况下而传输或接收数据；

- **STALL**作为输入标记的回应，或者在输出事务的数据时相之后由功能部件返回（见图8-9和图8-13）。STALL表示功能部件不能传输，或者接收数据，或者不支持一个控制管道请求。在任何条件下都不允许主机返回STALL。

STALL握手由设备用于在两个不同的场合之一。第一种情况，是当设置了与端口相联系挂起特征（Halt feature）的时候，称为“功能STALL（functional stall）”（挂起特征在这文档的第9章中详细说明）。功能STALL的特殊情况是“命令STALL（commanded stall）”。如同在第9

章中详细叙述的那样，命令STALL发生在主机显式地设置了端口的挂起特征的时候。如果功能部件的端口被挂起，则功能部件必须继续返回STALL，直到引起停止的条件通过主机干涉而被清除。

如同在节8.5.2中详细叙述的那样，第二种情况称为“协议STALL（protocol stall）”。协议STALL对于控制管道是唯一的。协议STALL和功能STALL在意义和持续时间上是不同。协议STALL在控制传送（Control Transfer）的数据或状态阶段（Stage）被返回，并且，STALL条件在下一个控制传送的开始终止（建立事务）。这节的剩下的部分将提到功能STALL的一般情况。

8.4.5握手回答（Handshake Response）

传输和接收功能部件必须根据从表8-2中到表8-4详细叙述的优先顺序返回握手。不是所有的握手都是被允许，依赖于事务类型和功能部件或主机是否发出握手。如果标记在传输到功能部件的阶段里发生了错误，则功能部件将不以任何包回应，直到下一个标记被收到并成功地译码。

8.4.5.1功能部件对输入事务回答

表8-2显示了功能部件作为对输入标记的反应而可能做的回答。如果由于停止或流控制条件，功能部件不能发送数据，它将发出STALL或NAK握手。如果功能部件能发出数据，它就发出数据。如果收到的标记被损坏，则功能部件不应答。

表8-2 功能部件对输入事务的回应

收到的标记损坏	功能部件的发送端口的挂起特征	功能部件能发送数据	采取的动作
是	不管	不管	不回应
否	置了位	不管	发送STALL握手
否	没置位	否	发送NAK握手
否	没置位	能	发送数据包

8.4.5.2主机对输入事务回答

表8-3显示了主机对输入事务回答。主机只能返回1种类型的握手：ACK。如果主机收到了损坏的数据包，它把数据丢弃且不应答。如果主机不能从功能部件接受数据，则（出于类似内部缓冲溢出的问题）这条件被认为是错误，并且主机不应答。主机能接受数据，并且如果数据包是完整无错地被接收到，则主机接受数据并发出ACK握手。

表8-3 主机对输入事务的回应

数据包损坏	主机能接受数据	主机返回的握手
是	N/A	丢弃数据，不回应
否	否	丢弃数据，不回应
否	能	接受数据，发送ACK

8.4.5.3功能部件对输出事务回答

对输出事务的握手回答由表8-4表示。假设标记译码成功，功能部件收到数据包后，可以返回三种握手类型中的任何一种。如果数据包被损坏，功能部件不返回握手。如果数据包是被完整无错地接收到，而功能部件的接收端口被停止，则功能部件返回STALL。如果事务正维持着时序位同步而探测到失配（在第8.6节有详细描述），那么功能部件返回ACK，并丢弃数据。如果功能部件能够接受数据并完整无错收到数据，它返回ACK。如果由于流控制的原因，功

能部件不能接受数据包，它返回NAK。

表8-4 功能部件对输出处理的回应（按优先顺序）

数据包损坏	接收器的挂起特征	时序位匹配	功能部件可接收数据	功能部件返回的握手
是	N/A	N/A	N/A	无
否	置了位	N/A	N/A	STALL
否	没置位	否	N/A	ACK
否	没置位	是	可	ACK
否	没置位	是	否	NAK

8.4.5.4功能部件对建立事务的回答

建立事务定义了特殊的主机-功能部件的数据事务，它允许主机初始化端口的同步位为主机的同步位。一收到建立标记，功能部件就必须接受数据。功能部件不能对建立标记用STALL或NAK应答，并且，接收功能部件必须接受建立标记后的数据包。如果非控制端口收到建立标记，它必须忽略事务且不应答。

8.5事务格式

包事务格式根据端口类型而变化。有4种端口类型：批处理（Bulk），控制（Control），中断（Interrupt）和同步（Isochronous）。

8.5.1批处理事务

批处理事务类型的特点是以错误检测和重试的方式保证主机和功能部件之间的数据的无错发送的能力。如图8-9所示，批处理事务是由标记，数据和握手包构成的三时相的事务。
在某些流控制和挂起条件下，数据时相被握手信号替换，从而产生了没有数据传输的两时相的事务。

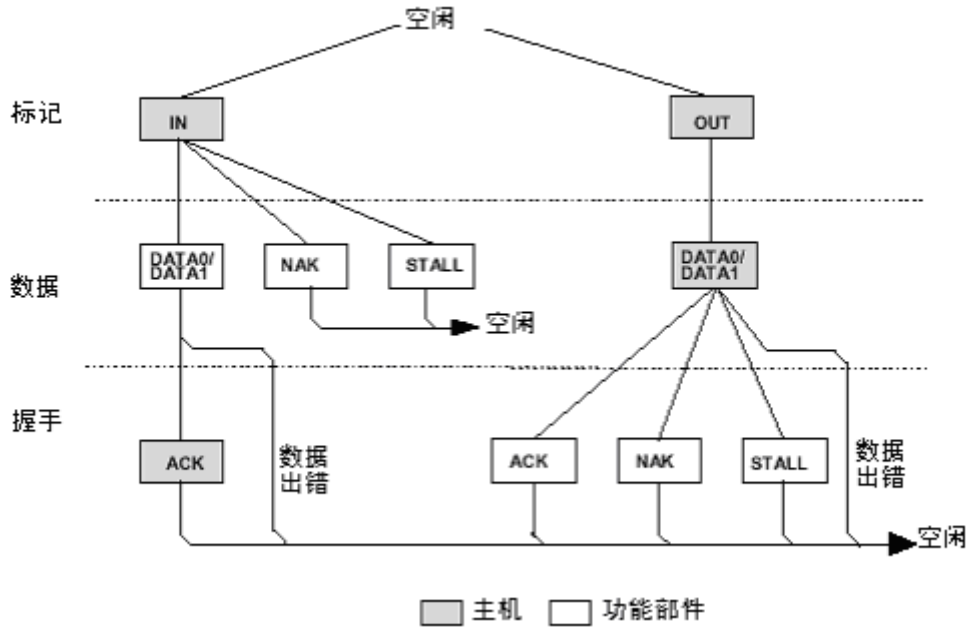


图8-9 批处理事务格式

当主机准备好了接收批处理数据的时候，它发出输入标记。功能部件端口通过返回数据包，或者如果不能返回数据，则返回NAK或STALL握手作为应答。NAK表示功能部件暂时不能返回数据，而STALL表示端口永久地被停止，需要USB系统软件干涉。如果主机收到合法的的

数据包，则它用ACK握手来应答。如果收到数据时主机检测到错误，它不返回握手包给功能部件。

当主机准备好了传送成批数据的时候，它首先发出一个后跟数据包的输出标记包。如果数据由功能部件无错地接收到，那么它将返回三个握手中的一个 ：

- ACK表示数据包无错地接收到，通知主机可以发送下一个包；
- NAK表示数据被无错地收到，但主机应该重新发送数据因为数据功能部件处于妨碍它接受数据的暂时的条件（例如缓冲满）中；
- 如果端口被停止，则返回STALL以告诉主机不要重试传输，因为功能部件上有错误条件。

如果接收到的数据有CRC或者位填充错，那么不返回任何握手。

图8-10说明了时序位和数据PID在成批读和写中的用法。数据包同步经数据时序切换位(Sequence Toggle Bit）和DATA0/DATA1 PID的使用而达到。当端口经历配置事件(Configuration event)（配置事件在节9.1.1.5和9.4.5中有解释）的时候，批处理端口的切换时序被初始化为DATA0。端口上的数据切换不是作为短包传送或IRP撤消的直接结果而被初始化的。

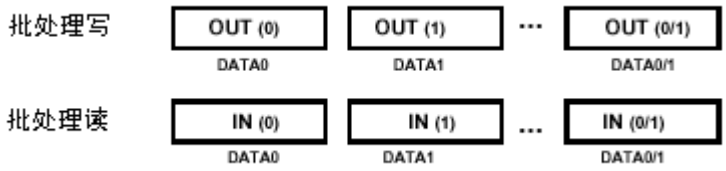


图8-10 批处理读和写

主机总是通过配置事件初始化总线传送的第一个事务为DATA0 PID。第二的事务使用DATA1 PID，并且，剩余的后继数据传送轮流切换。数据包发送器根据ACK的接收情况来切换而接收器根据数据包的接收（receipt）和验收（acceptance）的情况切换（参见第8.6节）。

8.5.2控制传送

控制传送最少有2个事务阶段：建立和状态。控制传送可以有选择性地包括建立和状态阶段之间的数据阶段。在建立阶段里，建立事务用于向功能部件的控制端口传输信息。建立事务在格式上类似于输出，但是使用的是建立而不是输出的PID。图8-11说明了建立事务的格式。建立总是在建立事务的数据时相上使用DATA0 PID。收到建立的功能部件必须接受建立数据并用ACK应答，如果数据被损坏，则丢弃数据且不返回握手。

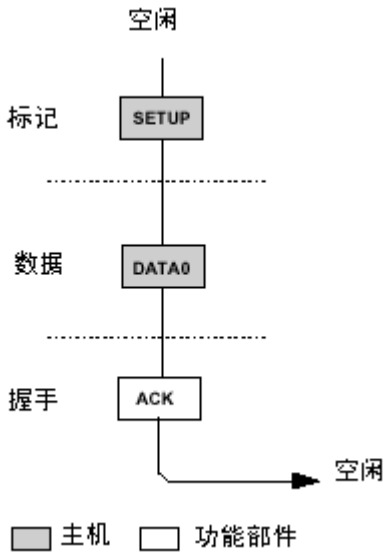


图8-11 控制建立事务

控制传送的数据阶段，如果有的话，由一个以上的输入或输出事务构成，遵守和批处理传送相同的协议规则。所有的数据阶段里的事务都必须有相同的方向（即全部输入或者全部输出）。在数据时相中要发送的数据的数量和其方向在建立阶段里被指定。如果数据的数量超过了先前确定的数据包大小，数据在支持最大的包大小的多个事务中被发送（输入或者输出）。任何剩下的数据都作为剩余在最后的事务中被发送。

控制传送的状态阶段是序列中的最后一个操作。状态阶段是以相对前面的阶段的数据流方向的变化来刻划的，并且总是使用DATA1 PID。例如，如果数据阶段由输出事务构成的，状态是单一的输入事务。如果控制传送序列没有数据阶段，那么它由建立阶段和其后的由输入事务构成的状态阶段构成。图8-12说明了事务顺序，数据时序位的值和控制读写序列的数据PID类型。时序位显示在括号中。

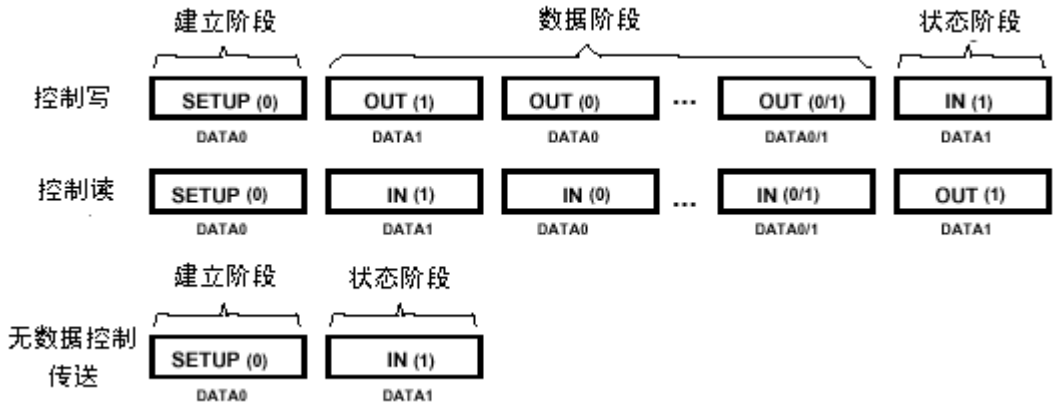


图8-12 控制读写序列

当控制端口在控制传送的数据和状态阶段中发送STALL握手的时候，必须对以后所有对此端口访问返回STALL握手，直到收到建立PID为止。

端口收到建立PID之后，不应返回STALL握手。

8.5.2.1汇报状态结果

状态阶段向主机做汇报传送中的先前的建立和数据阶段的结果。可能返回三种结果：

- 命令序列成功地完成了；
- 命令序列没能完成；
- 功能部件还在忙于完成指令。

汇报状态总是从功能部件到主机的方向。表8-5概括了每一种所需的应答类型。控制写传送在状态阶段的事务的数据时相返回状态信息。而对于控制读传送，主机在状态阶段事务的数据时相中发出零长度的数据包之后，功能部件在握手时相返回状态信息。

表8-5 状态阶段的响应

状态响应	控制写传送（在数据时相发送）	控制读传送（在握手时相发送）
功能部件完成	零长度的数据包	ACK握手
功能部件有一个错	STALL 握手	STALL握手
功能部件忙	NAK 握手	NAK握手

对于控制读，主机向控制管道发送输出标记以启动状态阶段。主机在这一时相中只发送零长度的数据包，但是，功能部件可以把任何长度包作为合法的的状态查询接受下来。应答这个数据包的管道握手表明了现在的状态。NAK说明了功能部件还在事务指令，并且主机应该继续状态阶段。ACK说明了功能部件完成指令，准备好了接受新指令。STALL说明了功能部件现在有妨碍它完成指令的错误。

对于控制写，主机发送输入标记到控制管道用以启动状态阶段。功能部件使用握手或零长度的数据包应答以说明其现状。NAK说明功能部件还在事务指令，并且主机应该继续状态阶段；返回零长度的包表明指令正常完成；而STALL表示功能部件不能完成指令。功能部件期待主机在状态阶段中对数据包以ACK应答。如果功能部件收不到ACK，它仍处于指令的状态阶段中，并且只要主机再发输入标记，它将继续返回零长度的数据包。

如果在数据阶段里指令管道被发送或者被请求返回比在建立时相（见节8.5.2.2）中的约定更多的数据，它应该返回STALL。如果控制管道在数据阶段里返回STALL，对于那个控制传送将没有状态阶段。

8.5.2.2可变长度数据阶段

控制管道可以放进可变长度的数据时相，在可变长度的数据时相中主机请求比指定的数据结构中所能包含的更多的数据。当全部的数据结构被返回到主机的时候，功能部件应该向管道返回一个短于包最大长度（MaxPacketSize）的包以表明数据阶段的结束。如果数据结构是数据是管道包最大长度（wMaxPacketSize）整数倍，功能部件将返回零长度的包以表明数据阶段的结束。

8.5.2.3最后数据事务的出错处理（Error Handling）

如果输入事务的ACK握手损坏，功能部件和主机在事务是否成功上这个问题上将暂时不一致。如果此事务的后面跟着另一个输入事务，切换重试机制（Toggle Retry Mechanism）将检测失配并从错误中恢复原状。如果此ACK是数据阶段最后的输入事务上的，则不能使用切换重试机制而必须使用另一种机制。

成功收到最后输入数据的主机将会发送ACK，然后主机将发出输出标记以启动传送的状态阶段。如果功能部件收不到结束数据阶段的ACK，功能部件将好象主机成功地收到数据那样去解释状态阶段的开始。而控制写则没有这种模棱两可的情况。如果输出事务上的ACK握手损坏，主机将不进入状态阶段而是重发最后的数据。节8.6.4有重试策略的详细分析。

8.5.2.4 控制管道返回的STALL握手

控制管道具有在控制传送中根据功能部件的问题返回STALL握手的独特能力。如果设备不能完成指令，它将在控制传送的数据和（或）状态阶段中返回STALL。与功能STALL的情况不同，协议STALL并不表示设备错误。协议STALL条件一直持续到接受到下一个建立事务，且功能部件将对于任何输入或输出事务返回STALL直到管道上的建立事务被收到为止。一般来说，协议STALL表示请求或者其参数不能被设备所理解，这样提供用来扩展USB请求的机制。

另外， 控制管道也可以支持功能STALL，但是不推荐使用。因为控制管道上的功能STALL表示它失去和主机通信的能力，所以这是一种退化的情况。如果控制管道的确要支撑功能STALL，它必须拥有能被主机设置或者清除的停止特征。第9章详细叙述事务在控制管道上停止特征的特殊情况的方法。设计良好的设备将把其全部功能和停止特征,与非控制端口联系起来。而控制管道应该预留用于为USB请求提供服务。

8.5.3中断事务

中断事务可由输入或输出构成。一收到输入标记，功能部件便可返回数据，NAK或STALL。如果端口没有新的中断信息（即没有等待事务的中断）可供返回，功能部件在数据时相里返回NAK握手。如果中断端口的停止特征被设置了，功能部件将返回STALL握手。如果中断是等待事务的，功能部件象数据包那样返回中断信息。作为对数据包接收的反应，主机如果数据无错地被接受则发出ACK握手，或者如果数据包损坏则不返回握手。图8-13说明了中断事务格式。

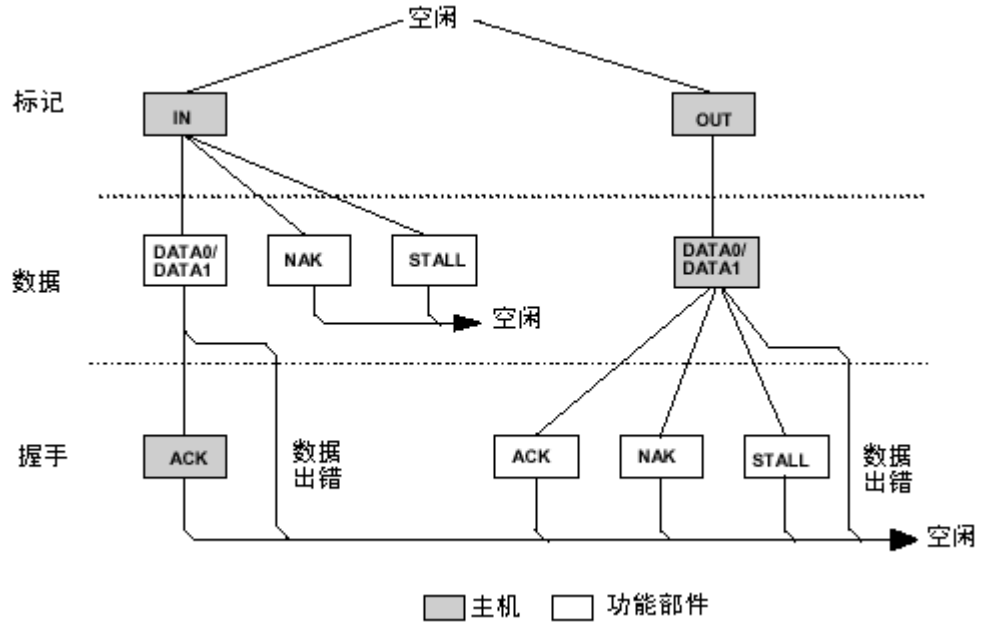


图8-13 中断事务格式

当端口为实际的中断数据使用中断传送机制的时候，必须遵循数据切换协议。这使功能部件得知主机收到了数据，并且事件条件被清除。这种“确保的”事件传送允许功能部件只发送

中断信息直到它被主机接收，而不是在USB系统软件清除中断条件之前，每次数据功能部件被选中时就必须要发送中断数据。

用于切换模式时，中断端口被端口上的任何配置事件初始化为DATA0 PID，其行为和图8-10所示的批处理一样。

另外， 中断端口可被用来为某些同步功能部件传达速率反馈信息。用于这种模式时，在每个数据包被送到主机之后，都不管握手包是否存在或类型如何，数据切换位都应该改变。只有中断输入端口支持这种能力。

8.5.4同步事务

如图8-14所示，同步（ISO）事务有标记和数据时相，而没有握手时相。主机发出输入或输出标记,后跟着端口（输入）或主机（输出）传送数据的数据时相。ISO事务不支持握手时相或重试能力。

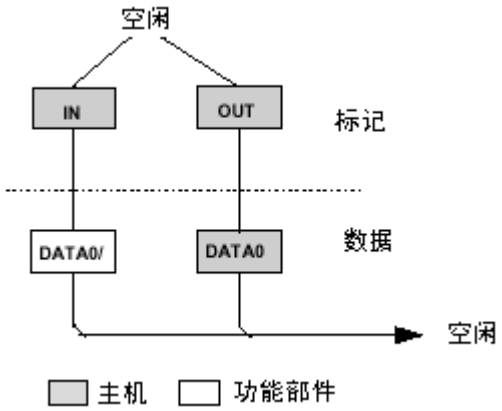


图8-14 同步事务格式

注解：设备或主机控制器都应该能接受DATA0和DATA1。设备或主机控制器应该只发送DATA0。

ISO事务不支持切换时序。

8.6数据切换同步和重试

USB提供一种机制以保证多事务中数据发送器和接收器之间的数据序列同步。这种机制提供一种保证发送器和接收器正确地解释事务的握手时相的方法。同步通过DATA0和DATA1 PID，以及分别从属于数据发送器和接收器的切换时序位的使用而完成。

仅在接收器能接受数据并且收到带有正确的数据PID的无错数据包的时候，接收器时序位才切换。而仅在数据发送器收到合法的ACK握手的时候，发送器时序位才切换。数据发送器和接收器必须在事务开始的时候同步它们的时序位。使用的同步机制随着事务类型而变化。

ISO传送不支持数据切换同步。

8.6.1 通过建立标记初始化

控制传送使用建立标记初始化主机和功能部件的时序位。如图8-15所示，主机向功能部件发送建立包，其后跟着输出事务。圆圈里的数代表发送器和接收器的时序位。功能部件必须接受数据并返回ACK。当功能部件接受事务的时候，它必须设置其时序位，以便主机和功能部件的时序位在建立事务的最后都等于1。

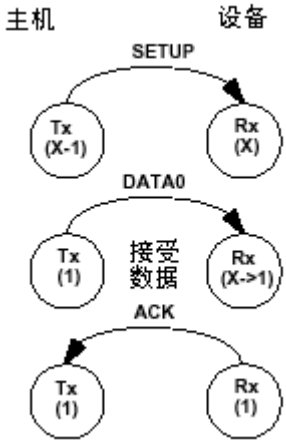


图8-15 建立初始化

8.6.2成功的数据事务

图8-16说明了有两个成功的事务的情况。对于数据发送器，这表示它根据ACK的接收情况来切换其时序位。仅当接收器收到合法的的数据包，并且包的数据PID和其时序位的当前值相匹配的时候，才切换其时序位。发送器在它收到数据包的ACK时才切换其时序位。

在每个事务中，接收器比较发送器的时序位（在数据包PID中编码为DATA0和DATA1）接收器的时序位。如果数据不能被接受，接收器必须发出NAK，并且，发送器和接收器的时序位保持不变。如果数据能被接受，并且接收器的时序位和PID相匹配，则数据被接受，并且时序位被切换。没有数据包的两时相的事务不使得发送器和接收器改变其时序位。

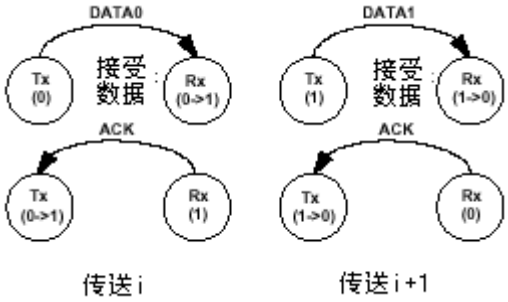


图8-16 连续的传送

8.6.3损坏，或者不被接受数据

如果数据不能被接受，或者得到的数据包被损坏，接收器将根据情况发出NAK或STALL握手，或者超时（Timeout），并且，接收器将不切换其时序位。图8-17说明了事务被返回NAK，然后被重试的情况。任何非ACK握手或是超时都将产生类似的重试动作。没有收到ACK握手的发送器，将不切换其时序位。其结果是失败的数据包事务使得发送器和接收器的时序位同步并不切换。然后事务将被重试，如果成功，将引发发送器和接收器时序位的切换。

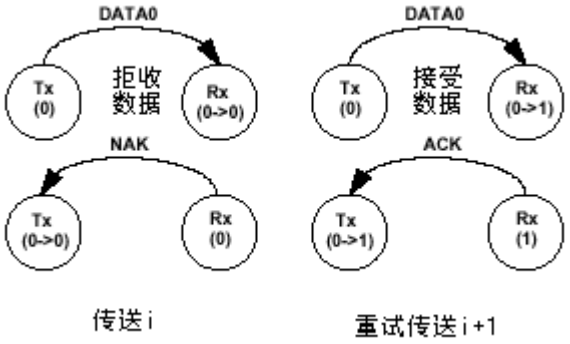


图8-17 重试的不被确认的事务

8.6.4损坏的ACK握手

发送器是根据其收到ACK握手确切地知道事务是否成功的最后并且唯一代理。如图8-18所示，丢失或者损坏的ACK握手使得发送器和接收器之间的暂时失去同步。这里发送器在发出合法的数据包，且接收机成功地收到；但是ACK握手损坏。

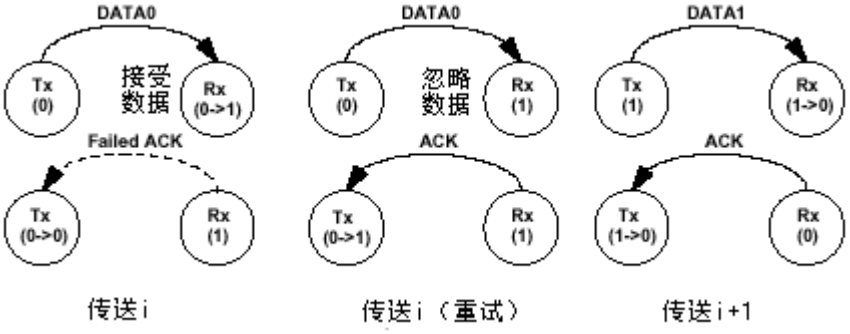


图8-18 重试的ACK损坏的事务

在事务i的最后，由它们各自的时序位间的失配可看出发送器和接收器暂时失去了同步。接收器已经收到了正确的数据，但是，发送器不知道它是否成功地发送了数据。在下一个事务中，发送器将重发使用DATA0 PID的先前的数据。接收器时序位和数据PID将不匹配，于是接收器知道它以前接受了这个数据。从而它丢弃此数据包且不切换其时序位。然后接收器发放ACK，使得发送器知道被重试的事务成功了。ACK的接收使得发送器切换其时序位。在事务i+1的开头，其时序位被切换，于是再一次同步了。

数据发送器必须保证任何被重试的数据包都和先前的事务发送的包相同（相同长度和内容）。由于类似缓冲欠载（Underrun）条件等问题，数据发送器不能传送和在先前的数据包中数据完全一样的数据中，它必须通过产生一位填充违反（Bit Stuffing Violation）；来中止事务。这将引起一个接收器的可检测的错误，从而保证接收器不会将部分的包解释为好包。发送器不应该通过发送已知的坏的CRC在接收器产生一个错误。带有“坏”CRC的坏包会被接收器解释成好包。

8.6.5低速事务

USB支持以两种速度发信号：以12.0Mb/s发信号的全速（Full-speed）和以1.5Mb/s发信号的低速（Low-speed）。在发全速下行（Downstream）信号中，集线器禁止所有挂有低速设备的端口收到下行总线通信。这是出于EMI和防止低速设备将一个全速包曲解为包是发送给它的考虑。图8-19说明了一个主机发出标记和握手并收到数据包的低速输入事务。

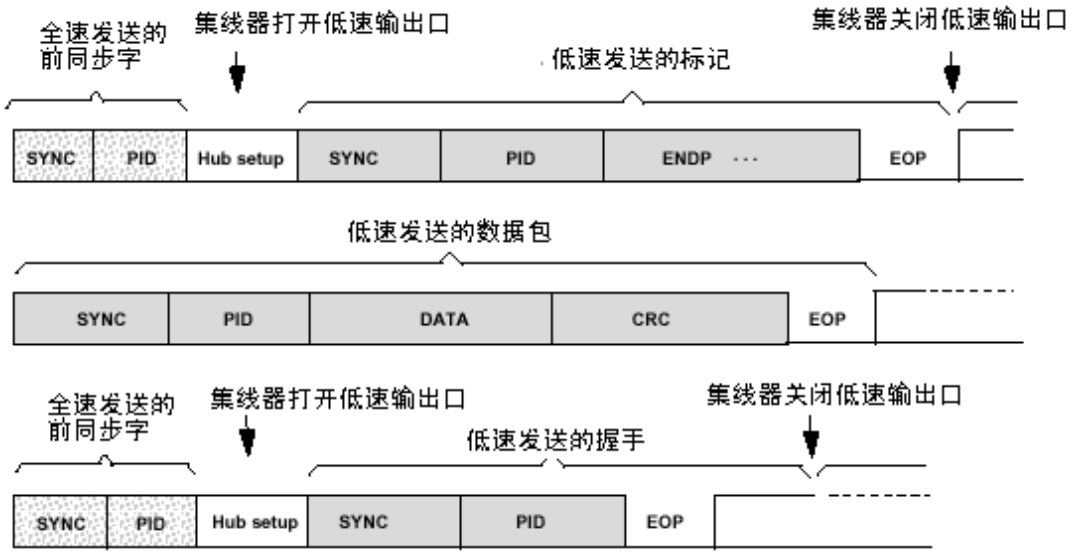


图8-19 低速事务

所有传送到低速设备的下行包都需要前同步信号（Preamble）。前同步信号由SYNC和其后的PRE（Preamble） PID构成——二者都以全速发送。集线器必须解释PRE PID ； 所有其他的USB设备都可以忽略它，把它当作未定义来处理它。在前同步信号PID的结束后，主机必须等待至少4个全速位时间（Bit time），而此间集线器必须完成启动和低速设备连接的端口上的重复器（Repeater）功能。在这个集线器建立的间隔里，集线器必须把它们的全速和低速端口置于它们各自的空闲状态。集线器必须在集线器建立间隔结束之前，准备好重复低速端口上所发的低速信号。

低速连接规则概括如下：

1.低速设备在连接过程中被识别，并且连接它们的集线器端口被识别为低速；

2.所有的下行低速包必须用打开低速集线器端口上的输出缓冲区的前同步信号（全速发送）开始；

3.低速集线器端口输出缓冲区在接受到EOP时被关上，并且直到前同步信号PID被检测到，才再一次被打开；

4.上行连接不受集线器端口是全速还是低速的影响。

低速信号是以主机用低速发送SYNC开始的，后面跟着包的剩余部分。包的结束是通过End-of-Packet（EOP）而被识别的，此时所有的集线器都断开并禁止任何连接有低速设备端口。集线器不对上行信号切换端口；低速端口对低速和全速信号在上行方向一直是允许的。

低速和全速事务维持高度的协议通用性（Commonality）。不过，低速信号确有某些限制，这包括：

- 数据有效负载（Payload）被限制在最多8个字节的范围内；

- 只中断和控制类型的传送被支持

- 低速设备不接受SOF包。

8.7错误检测和恢复

USB允许可靠的端到端（End-to-end）的通信,这种通信容许产生物理信号层上的错误。这包括可靠地检测到大量可能的错误的能力和基于事务类型的从错误中恢复的能力。例如，控制事务需要高度数据可靠性；它通过使用错误检测和重试的方法支持端到端的数据完整性。出于同步事务的带宽（Bandwidth）和等待时间（Latency）的要求的原因，它不允许进行重试，且必须对未纠正的错误有较高的容忍程度。

8.7.1包错误种类

USB使用3种错误检测机制：位填充违反，PID检验位和CRC。位填充违反在节7.1.9中定义。PID错误在节8.3.1中定义。CRC错误在节8.3.5中定义。

除SOF标记之外，任何被收到的损坏的包都使得接收器忽略它并丢弃随包而来的数据或其他的字段信息。表8-6列出了错误检测机制，它们适用的包的种类和包接收器相应的反应。

表8-6 包错误类型

字段	错误	动作
包标识符	PID校验，位填充	忽略包
地址	位填充，地址CRC	忽略标记
帧号	位填充, 帧号CRC	忽略帧号字段
数据	位填充, 数据CRC	丢弃数据

8.7.2总线周转（Turn-around）时间

设备和主机都不会发出指示以指出其收到的包有错误的。不作肯定答复则被认为是有错误的。作为这种错误汇报的方法的结果，主机和USB功能部件需要知道从发送器发完包的时候算起直到它开始收到应答为止过了多少时间。这一段时间被称为总线周转时间。当EOP的SE0-to-‘J’转换出现时，计时器开始计数，而当SOP的Idle-to-‘K’转换被检测到的时候停止计数。设备和主机都需要周转计时器（Timer）。设备总线周期时间定义为最坏情况下的往返延迟（Round trip delay）加上最大设备应答延迟（Maximum device response）（见节7.1.18）。如果发送器在最坏情况下的超时范围内没有收到应答，则认为包传输失败。USB设备超时（Timeout）（从先前的EOP的结束算起）应不少于16位的时间且不超过18位的时间。如果主机想通过超时表示一个错误条件，它必须在发出确保所有下行设备都超时的下一个标记之前最少要等待18个位的时间。

如图8-20所示，设备在标记和数据时相之间或数据和握手时相之间使用其总线周转计时器。主机在标记和数据时相之间或数据和握手时相之间使用其计时器。

如果主机收到不可靠的数据包，它必须在发出下一个标记之前等待。这一等待的间隔能确保主机不试图在错误的EOP之后立即发出标记。

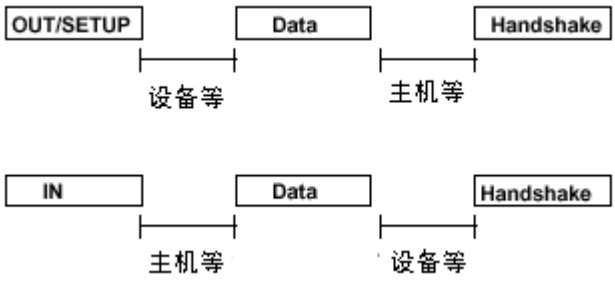


图8-20 总线周转计时器用法

8.7.3错误的EOP

错误的EOP必须被处理，以确保当前事务的包在主机或其它设备试图传输新包之前完成。如果发生这样的事件，它将构成总线冲突，并有能力损坏2个连续的事务的。错误的EOP的检查依赖于这样的事实：被插入错误EOP的包将呈现为被截短的带有CRC失效的包。（包的最后的16位是正确的CRC概率将非常低。）

主机和设备以不同的方式处理错误EOP。当设备发现一个损坏的数据包时，它不发出应答而是等待主机发送下一个标记。这保证了当主机还在传输数据包的时候，设备不试图返回握手。如果发生错误的EOP，主机数据包将最终结束，而设备将能检测下一个标记。如果设备发出带有错误的EOP的被损坏的数据包，主机将忽略此包，并且不发出握手。期待从主机传来握手信号的设备将超时。

如果主机收到损坏的数据包，它假定可能发生了错误的EOP并等待16位的时间来看是否有任何后继的上行（Upstream）通信量。如果主机在16位间隔的范围内没有检测到总线转换，并且总线处于空闲状态，则主机可以发出下一个标记。否则主机将等待设备结束发送其包的剩余部分。等待16个位的时间保证了2个条件：

- 第一个条件是确定了设备已结束发送它的包。这是由比最坏情况下6位时间的位填充间隔更长的超时间隔（没有总线转换）保证的；

- 第二的条件是保证发送器的总线周转计时器终止。

注意超时间隔是对事务速度敏感的。对于全速事务，主机必须等待16个全速位的时间；对于低速事务，它必须等待16个低速位的时间。

如果主机收到带有合法CRC的数据包，则它假定包是完整并且没有必要在发出下一个标记的过程中延迟。

8.7.4超时干扰（Babble）和活动性丧失（Loss of Activity）的恢复

USB必须能检测使其无限期地等待一个EOP或使总线在帧的结尾时不处于空闲状态的条件，并从中恢复。

- 活动性丧失（LOA）描述为SOP发出之后总线缺乏活动性（Lack of bus activity）（总线一直处于‘J’或‘K’）并且每帧结束时没有发出EOP；

- 超时干扰描述为SOP之后总线一直保持其活动性甚至超过了帧的结束。

LOA和Babble会使总线陷入僵局或者破坏下一个帧开头。无论哪一个条件都是不可接受，并且必须防止其发生。作为有责任控制连接的USB部件，集线器有责任检查和恢复Babble/LOA。通过使最近的集线器禁止与USB设备相连的端口的方法，防止所有不能在帧结束完成传输的USB设备传输超过一帧结束的数据。集线器Babble/LOA恢复机制的细节在节11.8.1中涉及到。

第八章　USB 设备架构

USB 设备可被划分三层：

• 底层是传送和接收数据包的总线接口

• 中间层处理总线接口与不同端点之间的数据路由端节点是数据的终结提供处或使用处，它可被看作数据源或数据接收端(Sink)

• 最上层的功能由串行总线设备提供，比如鼠标，或 ISDN 接口。

本章描述的是 USB 设备中间层的通用属性与操作。这些属性与操作由设备的特定功能的部分用于通过总线接口最终与主机(host)的通信。

9.1　USB 设备状态

USB 设备有若干可能的状态，其中一些对于 USB 与主机(host)来说是外置的，而另外一些对 USB 设备来说是内置的，这一节描述的就是这些外置状态。

9.1.1　外置的设备状态

本小节描述的是外部可见的 USB 设备状态(见图 8-1)。表 8-1 汇集这些外置设备状态之间的转化关系。

注意：USB 设备会响应上行端口(upstream Port)传来一个复位(reset)信号进行复位操作。当 reset 信号完成的时候，USB 设备业已复位。

连 接	加 电	缺 省	编 址	配 置	挂 起	说　明
不	——	——	——	——	——	设备尚未连接至接口.其他特性无关
是	不	——	——	——	——	设备已连接至接口,但未加电. 其他特性无关.
是	是	不	——	——	——	设备已连接至接口,并且已加电.但尚未被复位.
是	是	是	不	——	——	设备已连接至接口,已加电. 并被复位. 但尚未分配地址.设备在缺省地址处可寻址.
是	是	是	是	不	——	设备已连接至接口,已加电. 并被复位.且分配了唯一地址. 尚未被配置.
是	是	是	是	是	不	设备已连接至接口,已加电. 并被复位.且分配了唯一地址,并被配置. 设备功能可被使用.
是	是	——	——	——	是	设备在至少 3 毫秒以内探测不到总线活动,自动进如挂起. 设备功能不可用.

表 8-1.　外置(可见)的设备状态

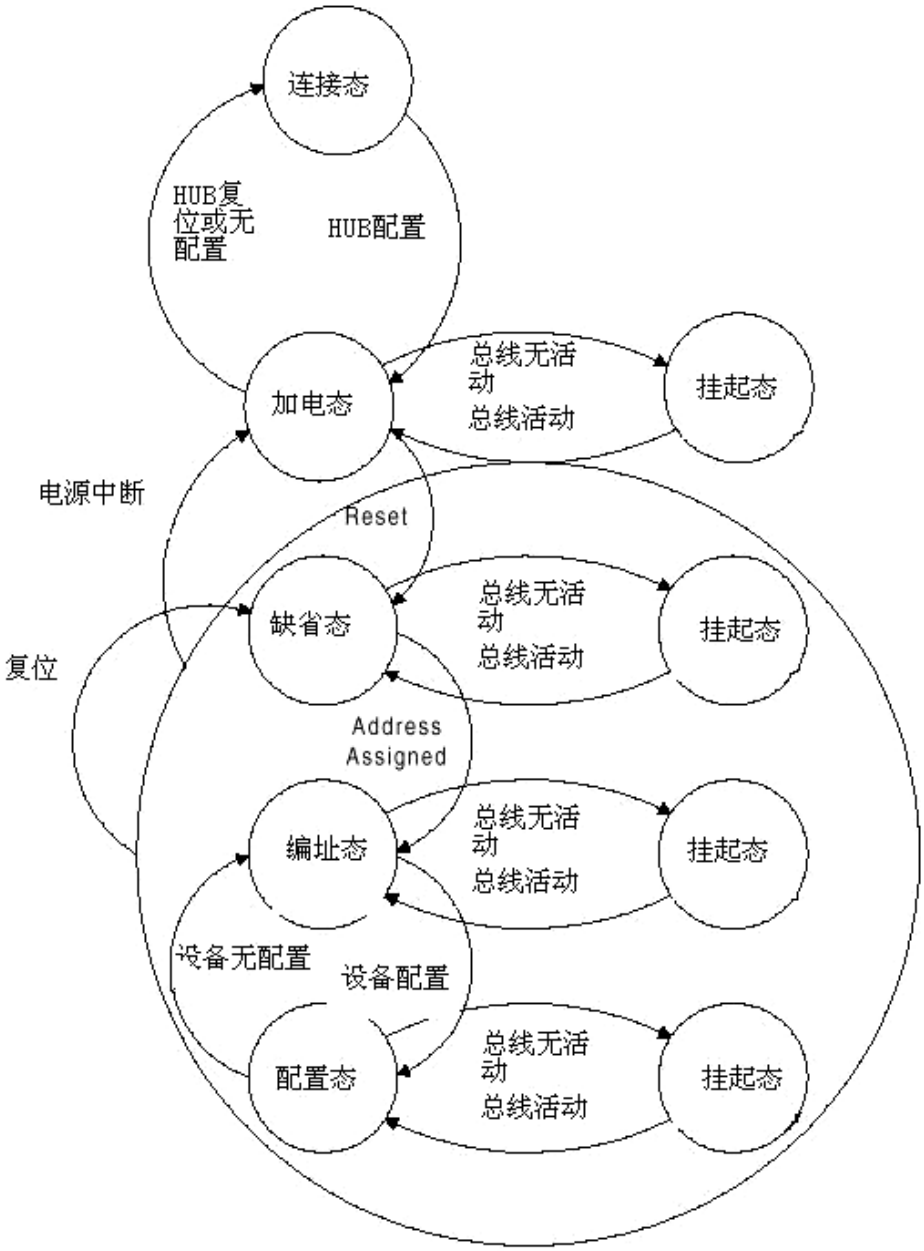

连接态
HUB复位或无配置
HUB配置
加电态
总线无活动
总线活动
挂起态
电源中断
Reset
缺省态
总线无活动
总线活动
挂起态
复位
Address Assigned
编址态
总线无活动
总线活动
挂起态
设备无配置
设备配置
配置态
总线无活动
总线活动
挂起态

图9-1 设备状态转化图

9.1.1.1 连接状态

USB 设备可被连接到 USB 接口上或从接口断开，USB 设备处在断开时的设备状态不包括在本篇说明之中。本说明中仅讨论那些处在连接状态的设备特性与操作。

9.1.1.2 加电状态(Powered)

USB 设备的电源可来自外部电源，也可从 USB 接口的集线器而来。电源来自外部电源的 USB 设备被称作自给电源式的(self-powered)。尽管自给电源式的 USB 设备可能在连接上 USB 接口以前可能已经带电，但它们直到连线上 USB 接口后才能被看作是加电状态(Powered state)。而这时候 VBUS 已经对设备产生作用了。

一个设备可能有既支持自给电源的，同时也支持总线电源式的配置。有一些支持其中的一种，而另一些设备配置可能只有在自给电源下才能被使用。设备对电源支持的能力是通过配置描述表(configuration descriptor)来反映的。当前的电源供给形式被作为设备状态的一部分被反映出来。设备可在任何时候改变它们的供电来源，比如说：从自给式向总线式改变，如果一个配置同时支持两种模式，那此状态的最大电源需求就是指设备在两种模式下从 VBUS 上获取电能的最大值。设备必须以此最大电源作为参照，而究竟处于何状态是不考虑的。如果有一配置仅支持一种电源模式，那么电源模式的改变会使得设备失去当前配置与地址，返回加电状态。如果一个设备是自给电源式，并且当前配置需要大于 100mA 电流，那么如果此设备转到了总线电源式，它必须返回地址状态(Address state)。自给电源式集线器使用 VBUS 来为集线控制器(Hub controller)提供电源，因而可以仍然保持配置状态(Configured state)，尽管自给电源停止提供电源。具体细节可参照 11.14 小节。

9.1.1.3 缺省状态

设备加电以后，在它从总线接收到复位信号之前不应对总线传输发生响应。在接收到复位信号之后，设备才在缺省地址处变得可寻址。

9.1.1.4 地址状态

所有的 USB 设备在加电复位以后都使用缺省地址。每一设备在连接或复位后由主机分配一个唯一的地址。当 USB 设备处于挂起状态时，它保持这个地址不变。

USB 设备只对缺省通道(Pipe)请求发生响应，而不管设备是否已经被分配地址或在使用缺省地址。

9.1.1.5 Configured (配置状态)

在 USB 设备正常工作以前，设备必须被正确配置。从设备的角度来看，配置包括一个将非零值写入设备配置寄存器的操作。配置一个设备或改变一个可变的设备设置会使得与这个相关接口的终端结点的所有的状态与配置值被设成缺省值。这包括将正在使用(date toggle)的结点(end point)的 (Date toggle) 被设置成 DATA0。

9.1.1.6 中止状态

为节省电源，USB 设备在探测不到总线传输时自动进入中止状态(参见第七章)。当中止时，USB 设备保持本身的内部状态，包括它的地址及配置。

所有的设备在一段特定的时间内探测不到总线活动时必须进入中止态，这一特定的时间在第 7 章中进行了说明。不管设备是被分配了非缺省的地址或者是被配置了，已经连接的设备必须在任何加电的时刻随时准备中止。总线活动的中止可能是因为主机本身进入了中止状态。另外，USB 设备必须在所连接的集线器端口失效时进入中止态。这就是所指的选择性中止(Selective suspend)。

USB 设备在总线活动来到时结束中止态。USB 设备也可以远程唤醒的电流信号来请求主机退出中止态或选择性中止态。具体设备具有的远程唤醒的能力是可选的，也就是说，如果一个设备有远程唤醒的能力，此设备必须能让主机控制此能力的有效与否。当设备复位时，远程唤醒能力必须被禁止。

9.1.2 Bus Enumeration 总线标号

当 USB 设备接上或从 USB 设备移开的时候，主机启动一个被称作总线标识(bus enumeration)的进程，来标识并管理设备状态的改变，当 USB 设备接上一个加电端口时，系统当采取以下操作：

1. USB 设备所连的集线器通过其通向主机的状态改变通道向主机，汇报本 USB 设备已连接上。(参照 11.13.3 节)。此时，USB 设备处于加电状态，它所连接的端口是无效的。

2. 主机通过寻问集线器决定此次状态改变的确切含义。

3. 主机一旦得知新设备已连上以后，它至少等待 100ms 以使得插入操作的完成以及设备电源稳定工作。然后主机发出端口使能及复位命令给那个端口。具体这些事件发生的顺序及时间判定请参看 7.1.7.1 节及图 7-19。

4.集线器将发向端口的复位信号持续 10ms(见 11.5.15 节)。当复位信号撤消后，端口已经有效了。这时 USB 设备处于缺省状态，并且可从 VBUS 汲取小于 100mA 的电能，所有设备寄存器及状态已经被复位，设备可对缺省地址产生响应。

5.主机给设备分配一个唯一的地址，设备转向编址状态。(Address state)。

6.在 USB 设备接受设备地址之前，它的缺省控制通道(Default Control Pipe)在缺省地址处自然是可寻址的，主机通过读取设备描述表，判决设备缺省通道的实际净数据负载。

7.主机从设备读取配置信息要从配置 0 读到配置 n-1，其中 n 为配置个数，此操作须花费几个毫秒。

8.基于从设备取来的配置信息及设备如何被使用的信息，主机给设备一个配置值，此刻，设备就处于配置状态(Configured state)并此配置有关的所有端节点，都按照配置各就各位，USB 设备现在可以从 VBUS 得到描述中所要求的电量了。从设备的角度来讲，它已经准备就绪了。

当 UBS 设备被取走时，集线器同样会通知主机，断开一个设备连接会使得设备所连接的端口无效，一收到断开通知后，主机就会更新的拓扑信息。

9.2 通用 USB 设备操作(Generic USB Device Operations)

所有的 USB 设备支持通用的操作集，这一节主要描述的这些操作。

9.2.1 动态插接与拔开

USB 设备必须在任意时刻允许被插接与拔开。提供连接点或端口的集线器应当负责汇报端口的状态改变情况。

当主机探测到连接操作后，会使得所连的集线器端口生效，设备也会因此而复位，一个被复位了的 USB 设备有如下特性：

- 对缺省 USB 地址发生响应
- 没有被配置
- 初始状态不是挂起

当设备从一个集线器端口移去时，集线器会使得原来连接的端口失效，并且通知主机设备已移去。

9.2.2 地址分配

当 USB 设备连接以后，由主机负责给此设备分配一个唯一的地址，这个操作

是在设备复位及端口使能操作以后。

9.2.3 配置

USB 设备在正常被使用以前，必须被配置，由主机负责配置设备。主机一般会从 USB 设备获取配置信息后再准定此设备有哪些功能。

作为配置操作的一部分，主机会设置设备的配置值，并且，如果必要的话会选择合适的接口的备选设置。

只须一个简单配置，一个设备可能支持多重接口。一个接口是一组端结点集合，它们代表了设备向主机提供的单一的功能或特性，用来与这组相关端结点通信的协议以及接口内各端结点的目的可以作为一个设备类的一部分或者由厂商制定具体定义。

另外，一个配置中的结口可能有备选设置。这些备选设置会重定义相关端结点的数目或特性。如果是这样的话，设备必须支持 GetInterface(接口请求)与 SetInterface(接口设置)请求，来汇报及选择指定的接口的设备选设置。

在每个设备配置下，每个接口描述表可能包括用来标识接口的及备选设置的域，接口被从 0～N-1 编号。n 为配置所支持的能同时使用的接口数目，类似的设置的编号也从 0 开始。当设备初始化配置后，缺省设置是备选设置 0。

为了支持通用的设备驱动程序管理一组相关的 BUS 设备，设备与接口描述表中包含了类(Class)，子类(Sub class)，及协议(Protocol)域。这些域用来标识一个设备的功能及用于通信的协议。

一个类值被分配给一组按照特性划分成 USB 类说明一部分的设备。一个类的设备可进一步划分成子类，并且在一个类或子类中，一个协议代值可定义主机软件是怎样与设备通信的。

注意：类、子类、与协议值必须一致，但在本说明范围之外。

9.2.4 数据传送

数据可能以四种方式在 USB 设备端结点与主机之间传送。四种传送方式参见第五章。在不同设置下，一个终端结点可能被用于不同的传输方式，但一旦设置选定，传送方式就选定了。

9.2.5 电源管理

USB 设备的电源管理包括以下说明部分的几条。

9.2.5.1 电源

USB 总线电源是一个有限的资源，在设备标识(device enumeration)阶段，主机估测电源的需求。如果电源的需求量超过 USB 总线所能提供的电量，主机软件则不能选择那个配置。

USB 设备应将电源需求量限制在一个单元以下，直到被配置。中止(挂起)的设备，不管是否已经配置过了，应将总线耗电降到第 7 章定义的标准以下。视接到设备的端口电源负载能力而定，USB 设备在配置了以后可从 VBUS 汲取达 5 个单元的电量。

9.2.5.2 远程唤醒

远程唤醒能力参许一个被挂起的 USB 设备发达信号给处于挂起状态的主机。这个信号会使得主机醒来，处理触发事件。USB 设备通过配置描述来向主机汇报其远程唤醒的能力。USB 设备的远程唤醒能力应能被禁止的。远程唤醒能力通过 7.1.7.5 节中电信号的方式来达到的。

9.2.6 请求处理

除 SetAddress( )请求以外(见 9.4.6 节)，在安装完成返回 ACK 信号以后，设备就开始处理请求。在某一状态成功结束以前，设备应当“完成”对请求的处理。许多请求费时较多，像这样的请求，该设备类应定义一个方法而不是等待交换状态信息阶段(StatusStage)的结束来表示该操作已经完成。像这样的操作有：集线器端口的复位至少需 10ms 来完成。当端口复位产生时，SetPortFeature (PORT-RESET)(见 11 章)请求就结束了。当端口状态改变并表明此端口已经生效时，一个信号就会产生表明复位信号已经结束。这种技术可以防止当主机知道某一个请求费时较长的情况一直探测此请求是否已完成。

9.2.6.1 请求处理的定时处理

所有的设备应当及时处理请求，USB 给定一个 5 秒的命令处理的时间上界。这个限制并不是对所有情况都适用的。这些限制在接下来的部分给予描述，应当说明的是，下面的限制包括实现的很大的范围。如果所有设备都采用最大的请求处理允许时间的话，用户是无法忍受的。应此，具体实现应当尽可能快地完成请求的处理。

9.2.6.2 复位/继续 恢复时间

当一个端口被复位或从中止态继续的时候，USB 系统软件应当等待一个 10ms 的恢复时间才能确保端口对数据传输产生响应。

一旦恢复时间段结束(从 reset 信号结束，或 resume 信号结尾的 EOP 结束开始计时)设备必须在任意时刻都能对数据传输作出响应。

9.2.6.3 设置地址的处理

在 reset/resume 恢复时间段以后，如果设备收到 SetAddress( )请求，设备必须能在 50ms 内完成请求的处理，并完成状态的转换，在 SetAddress ( )的请求下，当设备发出 0 长度的状态数据包或设备收到状态数据包的响应信号 ACK 就表明状态转换结束了。

在状态转换结束后，设备有 2ms 的 SetAddress ( )恢复时间。在这段时间结束以后，设备必须能在新地址处接受 Setup 数据包，并且，必须确保此时设备不对旧地址的信号产生响应。(当然，除非新旧地址是一样的)。

9.2.6.4 标准设备请求

对于不须传送数据标准的设备请求，一个设备必须在收到请求的 50ms 以内结束对请求的处理及状态的转换。

对于需要数据传输的标准设备请求，设备必须在收到请求的 500ms 以内返回第一个数据包。接下来的数据包必须在前一个数据包发送起的 500ms 以内开始发送。设备必须在最后一个数据包返回以后的 50ms 以内结束状态的转换。

对于需要数据传输的标准设备请求，5 秒的限制就起作用了。这意味着设备必须能在主机以设备最大能接受的速率发送数据包的情况下接收所有的数据包并且完成状态切换，数据包之间时延是主机让设备完成请求处理而加入的。

9.2.6.5 与类有关的请求

除非在类文档中特别说明，所有的类有关的请求必须按照标准请求的时间限制。

类说明文档可能要求设备反应比这部分讲的要快。标准设备请求与与类有关的设备请求可被要求反应更快。

9.2.7 请求错误

如果一设备收到一个请求，它或是在设备中无定义，或是不适用于当前设置，或是数值不对，这时就会产生一个请求错误。设备在下一个数据传输阶段或状态

交换阶段(Status stage)返回一个表明错误的 STALL PID 信号，一般在下一个数据传输返回更好，这样可减少不必要的总线活动。

9.3 USB 设备请求

所有的 USB 设备在设备的缺省控制通道(Default Control Pipe)处对主机的请求发出响应。这些请求是通过使用控制传输来达到的，请求及请求的参数通过 Setup 包发向设备，由主机负责设置 Setup 包内的每个域的值。每个 Setup 包有 8 个字节。见表 8-2。

偏移量	域	大小	值	描述
0	bmRequestType	1	位图	请求特征: D7: 传输方向 0=主机至设备 1=设备至主机 D6..5: 种类 0=标准 1=类 2=厂商 3=保留 D4..0: 接受者 0=设备 1=接口 2=端点 3=其他 4..31=保留
1	bRequest	1	值	具体请求(参见表 8-3)
2	wValue	2	值	字长域,根据不同的请求含义改变.
4	wIndex	2	索引或偏移	字长域,根据不同的请求含义改变.典型用于传送索引或偏移.
6	wLength	2		如有数据传送阶段,此为数据字节数.

表 8-2 . Setup 数据包的格式

9.3.1 bmRequestType 域

这个域表明此请求的特性。特别地，这个域表明了第二阶段控制传输方向。

如果 wLength 域被设作 0 的话，表明没有数据传送阶段，那 Direction 位就会被忽略。

USB 说明定义了一系列所有设备必须支持的标准请求。这些请求被例举在表 8-3 中。另外，一个设备类可定义更多的请求。设备厂商也可定义设备支持的请求。

请求可被导引到设备，设备接口，或某一个设备端结点(endpoint)上。这个请求域也指定了接收者。当指定的是接口或端结点(endpoint)时，wIndex 域指出那个接口或端节点。

9.3.2 bRequest 域

这个域标识特别的请求。bmRequestType 域的 Type 啦可修改此域的含义。本说明仅定义 Type 字位为 0 即标准设备请求时 bRequest 域值的含义。

9.3.3 wValue 域

此域用来传送当前请求的参数，随请求不同而变。

9.3.4 wIndex 域

wIndex 域用来表明是哪一个接口或端结点，图 8-2 表明 wIndex 的格式(当标识端结点时)。Direction 位在设为 0 时表示出结点，设为 1 时表示是入结点，Endpoint Number 是结点号。图 8-3 表明 wIndex 用于标识接口时的格式。

D7	D6	D5	D4	D3	D2	D1	D0
方向	保留(为 0)			端点号			
D16	D15	D13	D12	D11	D10	D9	D8
保留(为 0)							

图 8-2. 所指为端点时 wIndex 格式

D7	D6	D5	D4	D3	D2	D1	D0

接口号							
D16	**D15**	**D13**	**D12**	**D11**	**D10**	**D9**	**D8**
保留(为 0)							

图 8-3. 所指为接口时 WIndex 格式

9.3.5 wLength 域

这个域表明第二阶段的数据传输长度。传输方向由 bmRequstType 域的 Direction 位指出。wLength 域为 0 则表明无数据传输。在输入请求下，设备返回的数据长度不应多于 wLength，但可以少于。在输出请求下，wLength 指出主机发出的确切数据量。如果主机发送多于 wLength 的数据，设备做出的响应是无定义的。

9.4 标准设惫请求

这部分描述的所有 USB 设备都定义的标准设备请求，表 8-3 将它们列出，而表 8-4、8-5 分别结出了对应的标准请求码及描述表类型。

不管设备是否被分配了非缺省地址或设备当前是被配置了的，它们都应当对标准请求产生响应。

特性选择符被用来设置特性或使特性生效。比如说某个设备、接口、或结点的远程唤醒功能，特性选择符的值在表 8-6 中得到了说明。

对于非法请求的处理从前所述。但是非法请求并不会使得控制通道设置 Halt 特性。如果因为某种原因，设备因为一个错误状态不能通过缺省控制通道来与主机通信，设备必须被 reset 来清除错误状态并重启缺省通道。

bmRequestType	bRequest	wValue	Windex	wLength	Data
00000000B 00000001B 00000010B	CLEAR_FEATURE	特性选择符	零 接口号 端点号	零	无
10000000B	GET_CONFIGURATION	零	零	一	配置值
10000000**B**	GET_DESCRIPTOR	描述表种类和索引	零或语言标志	描述表长	描述表
10000001B	GET_INTERFACE	零	接口号	一	可选设置
10000000B 10000001B 10000010B	GET_STATUS	零	零 接口号 端点号	二	设备, 接口 ,或 端点状态
00000000B	SET_ADDRESS	设备地址	零	零	无
00000000B	SET_CONFIGURATION	配置值	零	零	无

00000000B	SET_DESCRIPTOR	描述表种类和索引	零或语言标志	描述表长	描述表
00000000B 00000001B 00000010B	SET_FEATURE	特性选择符	零 接口号 端点号	零	无
00000001B	SET_INTERFACE	可选设置	接口号	零	无
100000010B	SYNCH_FRAME	零	端点号	二	帧号

表 8-3 标准设备请求

Brequest	Value
GET_STATUS	0
CLEAR_FEATURE	1
为将来保留	2
SET_FEATURE	3
为将来保留	4
SET_ADDRESS	5
GET_DESCRIPTOR	6
SET_DESCRIPTOR	7
GET_CONFIGURATION	8
SET_CONFIGURATION	9
GET_INTERFACE	10
SET_INTERFACE	11
SYNCH_FRAME	12

表 8-4 标准请求码

描述表种类	值
DEVICE	1
CONFIGURATION	2
STRING	3
INTERFACE	4
ENDPOINT	5

表 8-5 描述表种类

特性选择符	接受者	值
DEVICE_REMOTE_WAKEUP	设备	1

ENDPOINT_HALT	端点	0

表 8-6 标准特性选择符

9.4.1 清除特性(ClearFeature())

这个请求是被用来清除一个指定的特性。

wValue 中的特性选择符的值必须根据接收者来设定适当的值。接收者是设备要用设备特性选择符，是接口就必须用接口特性选择符，是端结点就要用端节点特性描述符。

参照表 8-6 来看选择符与接收者的对应关系。

一个 ClearFeature( )请求所指的特性如果不能被清除，不存在或指的是不存在的接口或结点号，会产生 RequestError 错误。

如果 wLength 不为 0，设备响应无定义。

缺省状态：当设备处于缺省状态时，对此请求的反应无定义。

地址状态：在设备处于地址状态时这个请求是合法的，但如果该请求指的是接口或是非零号端结点，会引起请求错误。

配置状态：在此状态下，该请求合法。

9.4.2 取得配置(GetConfiguration())

此请求返回当前设备配置值。

如果返回 0 值表明设备未配置。

如果 wValue, wIndex, wLength 的值与上面表中不同，设备响应无定义。

缺省状态：该请求响应无定义

地址状态：返回 0 值

配置状态：非 0 的 bConfigurationValue 值被返回

9.4.3 取得描述符

这个请求返回存 在的描述符。

wValue 域的高一字节标识描述表类型，低一字节表示描述表的索引(参见表 8-5)。wIndex 域标识字串描述表的语言(Language ID)如果是其它语言的话就设为 0，wLength 表示要返回多少字节。如果描述表长度大于 wLength 域值，那么只有描述表的初始部分被返回。如果描述表比 wLength 域值，没发送一个短包来标

志传输的结束。一个短包被定义成一个长度短于最大负载长度或一个空(NULL)包。

这个标准请求包括三种描述符：设备、配置、及字串一个配置描述表的设备请求会一次返回配置描述表，所有的接口描述表和所有接口的端节点的描述表。第一个接口描述表紧跟着配置描述表，第一个接口的端节点的描述表随后。如果有其它的接口与端节点，它们的描述表欲跟在第一个接口与端节点描述表之后。与类有关的描述表，和/或厂商定义的描述表跟在标准描述表之后。

所有的设备必须提供一个设备描述表并且至少一个配置描述表，如果一个设备不支持一个请求的描述表，则返回请求错误。

缺省状态：此请求合法。

地址状态：此请求合法。

配置状态：此请求合法。

9.4.4 取得接口设置(GetInterface())

这个请求返回所指接口的选中的可选设置。

有些 USB 设备的接口配置有互斥的设置。这个请求使得主机决定当前设置。

如果 wValue 或 wLength 不依以上的表中设值，设备响应无定义。

如果所指的接口不存在，返回请求错误。

缺省状态：此状态下，设备对该请求响应无定义

地址状态：设备返回请求错误

配置状态：此请求合法

9.4.5 取得状态(GetStatus ( ))

这个请求返回所指接收者的状态。

bmRequestType 域的 Recipients 位段表示出接收者。

如果 wValue 或 wLength 不是上表中值，或 wIndex 在取设备状态请求时非 0 则设备响应无定义。

缺省状态：设备响应无定义

地址状态：如果所指的是接口或是一个非 0 号端节点，设备响应返回请求出错。

配置状态：如果所指接口或端节点不存在，返回请求错误。

一个 GetStatus()请求返回信息的格式如图 8-4。

D7	D6	D5	D4	D3	D2	D1	D0
保留（为零）						远程唤醒	自给电源
D15	D14	D13	D12	D11	D10	D9	D8
保留（为零）							

图 8-4 GetStatus()发向设备时返回信息

其中 Self Powered 域表明设备当前是否是自给电源。如果 Do 为 0 设备是总线电源式。如果 Do 被设成 1，设备是自给电源式的。此域不应当被 SetFeature( ) 或 ClearFeature ( )请求改变。

Remote Wakeup 域表明此设备当前是否支持远程唤醒，支持远程唤醒能力的设备缺省值是无效的(disabled)，如果 D1 被 Reset 成 0，远程唤醒能力就被 disabled。返之，设成 1 就是具有该功能，此域可被 SetFeature( )和 Clear Feature ( )使用 DEVICE-REMOTE-WAKEUP 特性选择符修改，设备复位时此域被设成 0。

一个 GetStatus ( )的接口请求返回如图 8-5 的信息。

D7	D6	D5	D4	D3	D2	D1	D0
保留（为零）							
D15	D14	D13	D12	D11	D10	D9	D8
保留（为零）							

图 8-5 GetStatus()发向接口时返回信息

一个 Get Statue( )的端节点请求返回如图 8-6 所示信息。

D7	D6	D5	D4	D3	D2	D1	D0
保留（为零）							停机
D15	D14	D13	D12	D11	D10	D9	D8
保留（为零）							

图 8-6 GetStatus()发向端点时返回信息

Halt 特性应当在所有的中断及端结点中实现。如果端节点当前被 halted 了，那么这个 Halt 特性就设成 1，否则为 0。Halt 特性可选择性地由 Set

Feature(ENDPOINT-HALT)请求来设置，一旦被 SetFeature(　)请求设置，设备的响应就会像这个域由硬件条件设置的一样，如果导止停机(halt)的条件去除了，用 Clear Feature (ENDPOINT-HALT)请求清除 halt 特性会导致端节点再也不会返回 STALL 信号。对于使用　(Date toggle)的端节点，不管一个端节点的 Halt 特性是否已被设置，一个 Clear Feature (ENDPOINT-HALT)总会导致　(date toggle)被重新初始化成 DATAO Halt 特性在收到 SetConfiguration (　)或 Set Interface(　)请求后总会被复位成 0。

Halt 特性不要求也不建议在缺省控制通道实现。然而，设备可设置缺省控制通道的 Halt 特性来反映一个功能出错的状态。如果这个特性被设了的话，设备将对除 Getstatus(　),SetFeature(　),Clear Feature (　)之外的请求返回 STALL 信号，设备可不对类有关的及厂商定制的请求返回 STALL 信号。

9.4.6　设置地址(SetAddress (　))

本请求为设备的将来存取设置地址

wValue 指出所要设置成的地址值

像在另处所述，请求实际可分成三个阶段。在第一阶段，Setup 包被送至设备，在第二个可有无的阶段，数据在设备与主机之间传送，在第三阶段，状态信息在主机与设备之间传送。数据与状态传送的方向要看是主机发数据给设备还是设备发数据给主机。状态的传送方向总是与数据传送方向是相反的，如果没有数据传输阶段则状态由设备传向主机的。

Setup 包传送以后的两个阶段的地址保持与 Setup 包传送阶段的一致。USB 设备只有在 Status 阶段过后才能改变设备地址。注意，在这方面此请求不同于其它请求。其它请求总是在状态传送阶段之前完成指定操作的。

如果所指的设备地址大于 127 或 wIndex 或 wLength 非零，设备响应无定义。

设备对 SetAddress(0)的响应无定义

缺省状态：如果地址值非 0，那设备将进入地址状态，否则地址仍留在缺省态(此非出错状态)

地址状态：如果新地址值为 0，进入缺省态，否则仍留在地址状态但使用新地址

配置状态：在此状态下设备对此请求的响应无定义。

9.4.7 设置配置值(SetConfiguration( ))

此请求设置设备配置值

wValue 域的低字节指出配置，这个配置值必须为 0 或与配置描表中的一个配置相配。如果配置值为 0，设备置地址状态。wValue 的高字节保留。

如果 wIndex, wLength 或 wValue 的高字节非 0，则设备对之的响应无定义。

缺省状态：设备响应无定义

地址状态：如果所指的配置为 0，设备停留在地址状态。如果所指的配置与描述表中的一个值相匹配，那个配置就被选中，设备转到配置有。否则，返回请求错误

配置状态：如果配置值为 0，设备进入电址状态。如果配置值非 0 并与描述表中的一个配置相匹配则设备仍留在配置态，但采用新的配置值，否则返回请求错误。

9.4.8 设置描述表(SetDescriptor ( ))

此请求用于更新或添加新的描述表。

wValue 域的高字节指出了描述的类型，低字节指出了描述表索引(参风表 8-5)。wIndex 域指出了字串描述表的语言标识，对于其它描述表来说它为 0。wLength 指出从主机传向设备的字节数。

如果设备不支持该请求则设备返回一个请求错误

缺省状态：此状态下设备对该请求反应无定义

地址状态：如果设备支持请求，则为合法

配置状态：如果设备支持请求，则为合法

9.4.9 设置特性(SetFeature ( ))

这个请求被用来设置或使一个特性生效。

wValue 域中的特性选择符必须跟接收者相配。

哪个选择符对应什么接收者的定义请参照表 8-6

SetFeature ( )请求如果指出一个不存在的特性会使得设备在交换状态阶段返回 STALL 信号。

如果 wLength 为非 0，设备响应无定义。

如果 SetFeature ( )指的是一个不存在的接口或端节点，设备返回一个请

求错误。

缺省状态：设备响应无定义。

地址状态：合法、除非请求错误。

配置状态：合法。

9.4.10 设置接口(SetInterface ( ))

此请求让主机为指定的接口选择一个设置。

如有 USB 设备接口配置中有互斥设置。此请求让主机选择所要的设置。如果设备的接口只支持缺省设置，在状态交换阶段设备返回 STALL

如果所指接口或设置不存在，设备返回请求错误。

wLength 为 0，设备响应无定义

缺省状态：设备响应无定义

地址状态：设备返回请求错误

配置状态：合法

9.4.11 同步帧(SynchFrame ( ))

该请求用来设置或汇报一个结点的同步帧。

如果一个端节点支持同步传输，端节点可能会根据某一特点的模式来以变长方式传送每一帧。主机与端节点必须在什么时候出现重复模式的第一帧出现上达成一致。模式开始帧的序号由设备返回给主机。这个帧序号由模式首帧前的 SOF 信号传向端节点。设备还可以用此请求来使得帧模式重新开始。在这种情况下，设备应当保存每个 SOF 中的帧序号并在数据传送阶段返回这些值，并在数据传送阶段的每个 (IN)之后立即开始重新开始这个模式。

这个值仅用于隐式模式的同步数据传输。如果 wValue 非 0 或 wLength 非 2，设备响应无定义。

如果所指的端节点不支持此请求，设备返回一个请求错误。

缺省状态：设备响应无定义

地址地址：设备返回请求错误

配置状态：此请求合法

9.5 描述表

USB 设备通过描述表来反映他们的属性。描述表是有定义好的格式的数据结

构，每一个描述表以一个字节打头表明本描述表的长度，紧跟其后是一个字节的描述表类信息。

使用描述表使得单个配置的特性存储变得简明，因为每个配置可能会重复使用其它有相同特性的配置描述表的部分或全部，用这种方法，描述表用一个关系数据库来表绘一个个的单独数据记录。

在适当的地方，描述表包括了指向字串描述表的引用。字串描述表提供了人能读懂的信息。字串描述表可有可无，但描述表中的字串指引域是不可少的。如果一个设备不支持字串描述表，该域就为 0。

如果描述表值中的长度域值少于本说明的定义，此描述表非法，不能被主机接受。如果返回的描述表中的长义值大于本说明定义，则过长部分当被忽略，但下一个描述表的位置由返回长度而不是实际长度来决定。

设备可以两种方式返回类相关的或厂商定义的描述表。

1.如果这两种描述表的格式与标准格式相同(以长度字节打头，紧跟着类型字节)则它们可由 GetDescriptor(Configuration)请求与标准描述一同返回。在这种情况下，类相关或厂商定义的描述表一般跟在被修改的或被扩展的描述之后。

2.如果这两种描述表使用的非标准格式。指定类相关的或厂商定义的描述表及索引的 GetDescriptor( )请求可从设备返回这两种描述表。类或厂商说明会指出正确取出这两种描述表的途径。

9.6 标准描述表的定义

本说明中有关标准描述表的定义只能被本说明的校订本修改或扩展。

注意：一个对 USB 1.0 标准结点描述表的扩展已由《声音设备类说明修订本 1.0》公布。仅此是除 USB 说明外被许可的，将来的 USB 说明的修订本会用此来对标准节点描述表进行扩展来避免与《声》的冲突。

9.6.1 设备

设备描述表给出了 USB 设备的一般信息。这包括对设备及所有设备配置起全程作用的信息。一个 USB 设备只能有一个设备描述表。

所有的 USB 设备都有缺省控制通道。缺省控制通道的最大包长在设备描述表中得到了说明。一个配置的端节点与接口定义在配置描述表中，一个配置和它的接口不包括节点描述表。除最大包长外，缺少通道的特性由本说明定义，并且对

所有的 USB 设备都一样。

bNumberConfiguration s 域表明此设备支持的配置数。表 8-7 为标准设备描述表。

偏移量	域	大小	值	描述
0	bLength	1	数字	此描述表的字节数
1	bDecriptorType	1	常量	描述表种类为设备
2	bcdUSB	2	BCD 码	此设备与描述表兼容的 USB 设备说明版本号（BCD 码）
4	bDeviceClass	1	类	设备类码 如果此域的值为 0 则一个设置下每个接口指出它自己的类，并个接口各自独立工作。 如果此域的值处于 1~FEH 之间，则设备在不同的接口上支持不同的类。并这些接口可能不能独立工作。此值指出了，这些接口集体的类定义。 如果此域设为 FFH，则此设备的类由厂商定义。
5	bDeviceSubClass	1	子类	子类码 这些码值的具体含义根据 bDeviceClass 域来看。 如 bDeviceClass 域为零，此域也须为零 如 bDeviceClass 域为 FFH，此域的所有值保留。
6	bDevicePortocol	1	协议	协议码 这些码的值视 bDeviceClass 和 bDeviceSubClass 的值而定。 如果设备支持设备基础上的类相关的协议，此码标志了设备类说明上的值。 如果此域的值为零，则此设备不在设备基础上支持设备类相关的协议。然而，它可能在接口基础上支持设备类相关的协议。 如果此域的值为 FFH，此设备使用厂商定义的协议。
7	bMaxPacketSize0	1	数字	端点 0 的最大包大小（仅 8,16,32,64 为合法值）
8	idVendor	2	ID	厂商标志（由 USB 标准付值）
10	idProduct	2	ID	产品标志（由厂商付值）
12	bcdDevice	2	BCD 码	设备发行号（BCD 码）
14	iManufacturer	1	索引	描述厂商信息的字串的索引。
15	iProduct	1	索引	描述产品信息的字串的索引。
16	iSerialNumber	1	索引	描述设备序列号信息的字串的索引。
17	bNumConfigurations	1	数字	可能的设置数

表 8-7 标准设备描述表

9.6.2 配置

配置描述表给出了一设备配置的信息，描述表包括一个 b Configuration Value 域，在 SetConfiguration( )请时被用作参数来设置所需配置。

此描述表给出了此配置下的接口数，每个接口可能独立操作。比如，一个 ISDN 设备可能配置有两个接口，每个都提供 64KB/S 的有独立数据源与数据接收者的双向通道在另一个配置下 ISDN 可能表现为单个接口，将两个通道合成一个 128KB/S 的双向通道。

当主机发出请求要得配置描述表时，所有相关接口与端节点的描述表都被返回。

一个 USB 设备有一个或多个配置。每个配置只有一个或多个接口。而每个接口又有 0 个或多个端节点。在一个配置下，一个端节不会在接口之间共享，除非端节点被同一个接口的不同设置使用。在不同配置端节点，可无此限制。

一个配置好后，设备可支持对配置的有限调整，如果一个接口有备选设置，在配置好后可选择不同设置。表 8-8 是标准配置描述表。

偏移量	域	大小	值	描述
0	bLength	1	数字	此描述表的字节数。
1	bDescriptorType	1	常量	配置描述表类型
2	wTotalLength	2	数字	此配置信息的总长（包括配置，接口，端点和设备类及厂商定义的描述表）
4	bNumInterfaces	1	数字	此配置所支持的接口个数
5	bCongfigurationValue	1	数字	在 SetConfiguration（）请求中用作参数来选定此配置。
6	iConfiguration	1	索引	描述此配置的字串描述表索引
7	bmAttributes	1	位图	配置特性： D7： 保留（设为一） D6： 自给电源 D5： 远程唤醒 D4..0：保留（设为一） 一个既用总线电源又有自给电源的设备会在 MaxPower 域指出需要从总线取的电量。并设置 D6 为一。运行时期的实际电源模式可由 GetStatus(DEVICE) 请求得到。
8	MaxPower	1	mA	在此配置下的总线电源耗费量。以 2mA 为一个单位。

表 8-8 标准配置描述表

9.6.3 接口

此描述表在一个配置内给出一个接口的信息。如果一个配置支持不止一个接口，端节点的描述表会跟在接口描述表后被返回，接口描述表总是作为配置描述表的一部分被返回。接口描述不可直接用Set Description ( )和Get Descriptor ( )存取。

一个接口可能包含备选设置，以使得端节点或他们的特性在设备配置好以后能改变。一个接口的缺省设置总是可选设置。SetInterface ( )与 GetInterface ( )用来选择与返回选择了的接口设置。

可选的接口设置使得部分的设备配置能在其它接口进行操作的情况下改变。如果一个配置对于它的一个或多个接口有备选设置，每一设置包括一个独立接口描述表和相关结点。

如果一个设备配置支持单个接口，并此接口有两个可选设置，配置描述表返回以后会紧跟着返回bInterfaceNumber与bAlternateSetting域皆为0的第一个设置的接口描述表及相关的结点描述表，而随之后是另一个设置接口描述表与结点描述表。第二个接口描述表的 bInterfaceNumber 域也应为 0，但 bAlternate Setting域应为1。

如果一个接口仅使用节点 0，则接口描述表以后就不再返回节点描述表，并且此接口表示的是一个请求接口，它使用连在节点0上的缺省通道。在这种情况下 bNumberEndpoints 域应被设置成0。

一个接口描述表的节点个数不把结点0计在内。表8-9是标准节口描述表。

偏移量	域	大小	值	说明
0	bLength	1	数字	此表的字节数
1	bDescriptorType	1	常量	接口描述表类
2	bInterfaceNumber	1	数字	接口号,当前配置支持的接口数组索引（从零开始）
3	bAlternateSetting	1	数字	可选设置的索引值。
4	bNumEndpoints	1	数字	此接口用的端点数量,如果是零则说明此接口只用缺省控制管道。

5	bInterfaceClass	1	类	类值 零值为将来的标准保留。 如果此域的值设为 FFH，则此接口类由厂商说明。 所有其它的值由 USB 说明保留。
6	bInterfaceSubClass	1	子类	子类码 这些值的定义视 bInterfaceClass 域而定。 如果 bInterfaceClass 域的值为零则此域的值必须为零。 bInterfaceClass 域不为 FFH 则所有值由 USB 所保留。
7	bInterfaceProtocol	1	协议	协议码： bInterfaceClass 和 bInterfaceSubClass 域的值而定. 如果一个接口支持设备类相关的请求此域的值指出了设备类说明中所定义的协议.
8	iInterface	1	索引	描述此接口的字串描述表的索引值。

表 8-9　标准接口描述表

9.6.4　节点

每个接口使用的结点都有自己的描述表，此描述表被主机用来决定每个节点的带宽需求。每个结点的描述表总是作为配置描述的一部分返回的，结点 0 无描述表。8-10 为标准节点描述表。

偏移量	域	大小	值	说明
0	bLength	1	数字	此描述表的字节数
1	bDescriptorType	1	常量	端点描述表类
2	bEndpointAddress	1	端点	此描述表所描述的端点的地址。此地址的编码如下： Bit 3..0 : 端点号. Bit 6..4 : 保留,为零 Bit 7: 方向,如果控制端点则略。 0：出端点 1：入端点
3	bmAttributes	1	位图	此域的值描述的是在 bConfigurationValue 域所指的配置下端点的特性。 Bit 1..0 :传送类型 00=控制传送 01=同步传送 10=批传送 11=中断传送 所有其它的位都保留。
4	wMaxPacketSize	2	数字	当前配置下此端点能够接收或发送的最大数据包的大小。 对与同步传送此值用于为每帧的数据净负荷预留时间。而通道可能在实际运行时不需要预留的带宽。实际带宽可由设备通过一种非 USB 定义的机制汇报给主机. 对于中断传送,批传送,控制传送.端点可能发送较小的数据包。并且在结束传送后既有可能间隙时间来重启，也有可能不需要这段时间。具体请参照第五章。
6	bInterval	1	数字	轮寻数据传送端点的时间间隙。 此域的值对于批传送的端点及控制传送的端点忽略。对于同步传送的端点此域必需为 1。对于中断传送的端点此域值的范围为 1 到 255。

表 8-10　　标准端点描述表

9.6.5　字串

字串描述表是可有可无的。如前所述，如果一个设备无字串描述表，所有其它描述表中有关字串描述表的索引都必须为 0。

字串描述表使用的是 UNICODE 编码，则《Unicode 标准世界范围的字符编码》1.0 版，第一和第二卷定义，字串描述表支持多语言编码。当请求字串描述表时，

请求者用一个 6 位的语言标识指出语言，此语言 ID 由微软 Windows 定义(《开发 Windows 95 及 Windows NT 的国际化软件》，Nadine Kano，微软出版，华盛顿)。所有语言的 0 号字串索引返回一个字串描述表，该字串描述表为双字节的 LANGID 数组，表示设备支持的语言，表 8-11 表示了该 LANGID 数组。USB 设备可删除所有的字串描述表。USB 设备删了字串描述表后就不能返回 LANGID 码了。LANGID 数组不是以 NULL 结尾的。它的大小为 bLength-2。

偏移量	域	大小	值	描述
0	bLength	1	N+2	此描述表的字节数
1	bDescriptorType	1	常量	字串描述表类型
2	wLANGID[0]	2	数字	语言标识（LANGID）码 0
…	…	…	…	…
N	wLANGID[x]	2	数字	语言标识（LANGID）码 X

表 8-11 体现设备所支持的语言的码字

UNICODE 字串也不是 NULL 结尾的(见表 8-12)，字串长为 bLength-2。

偏移量	域	大小	值	描述
0	bLength	1	数字	此描述表的字节数
1	bDescriptorType	1	常量	字串描述表类型
2	bString	N	数字	UNICODE 编码的字串

表 8-12 UNICODE 字串描述表

9.7 设备类定义

所有的设备必须支持本章所讲的请求与描述表，多数设备还有设备特有的扩展的请求与描述表。另外，设备还会支持一组设备共有的服务。为了定义一个设备类，下面信息必须被提供出来，来定义此类设备的表现与行为。

9.7.1 描述表

如果此类须有对标准描述表特有的定义，则此定义必须在类说明中指出。另外，如果此类定义了标准的扩展描述表集合，它们必须在类中说明。扩展的描述表定义方法与标准描述表定义方法一致，比如说，所有描述表以表长打头。

9.7.2 接口与结点的使用

当一个设备类标准化以后，此类设备使用的接口以及结点们如何被使用必须在类定义中说明，设备类在满足基本的类定义以后还可扩展一些类特性。

9.7.3 请求

此类的所有请求必须被定义。

第十章　USB 主机：硬件与软件

USB 的互连支持数据在 USB 主机与 USB 设备之间的流动。这一章主要讲述为了简化主机上的 客户软件(Software client)与设备的功能部件(function)之间的通信而必须的主机接口(host interface)。在本章中所涉及的具体实现部份并不是必要的，这些实现部份是作为例子来阐述在响应 USB 设备请求时的主机系统的行为。只要 USB 设备并不感觉到主机行为的改变，USB 主机完全可以提供一个不同的软件系统实现方法。

10.1　USB 主机概况

10.1.1　概论

图 10-1 展示了 USB 通信模型之间基本的信息流与互连关系：

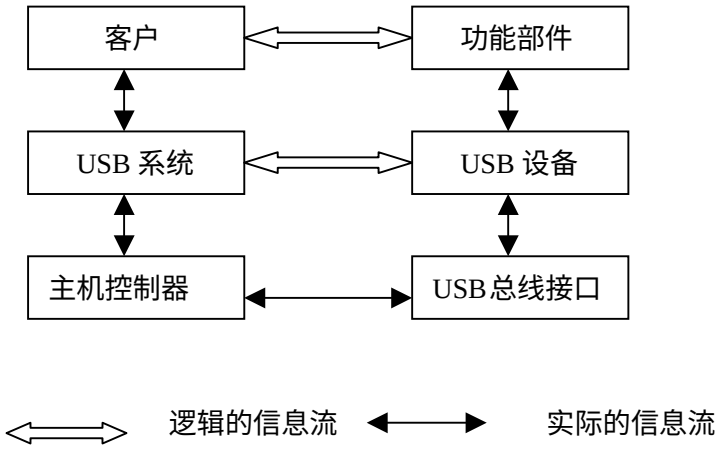


图　10-1 通信模型层次关系图

由图 10-1 可见，主机与设备都被划分成不同的层次。主机上垂直的箭头是实际的信息流。设备上对应的接口是基于不同实现的。在主机与设备之间的所有通信最终都是通过 USB 的电缆进行，然而，在上层的水平层之间存在逻辑的主机—设备信息流。主机上的客户软件和设备功能部件之间的通信是基于实际的应用需求及设备所能提供的能力。

客户软件与功能部件之间的透明通信的要求，决定主机和设备下层部件的功能以及它们的界面(interface)

这一章从主机的角度来描述上述的通信模型，图 10-2 描述了从主机角度看到的它与设备的连接。

主机在整个 USB 系统中是唯一的，它包括如下几个层次。

- USB 总线接口
- USB 系统(USB System)
- USB 客户(Client)

其中，USB 总线接口处理电气及协议层的互连(详见第 7 章及第 8 章)。从互连的角度看，USB 设备和 USB 主机都提供类似的 USB 总线接口，如串行接口引擎(Serial Interface Engine SIE)。由于主机在 USB 系统中的特殊性，USB 主机上的总线接口还必须具备主机控制器的功能(Host Controller)，主机控制器具有一个内集成的集线器(根集线器)提供与 USB 电缆的连接。

USB 系统(USB System)使用主机控制器来管理主机与 USB 设备的数据传输。USB 系统与主机控制器之间的界面基于主机控制器的硬件特性。USB 系统层相对于主机控制器而言，处理的是以客户观点见到的数据传输及客户与设备的交互。这包括附加的 USB 信息，比如协

议头(Protocol Wrappers)。USB 系统还必须管理 USB 的系统资源，以使得客户的访问成为可能。

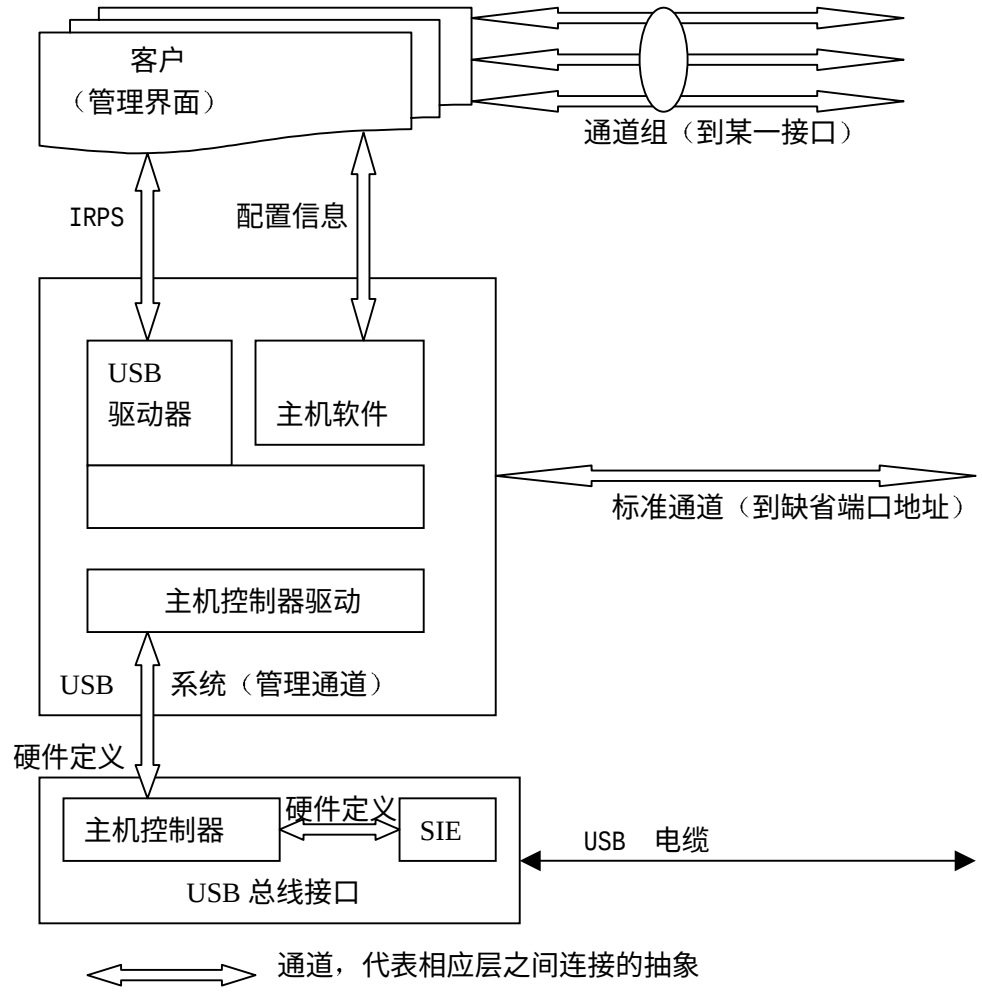


图 10-2　主机通信图

USB 系统有三个主要组成部份:

- 主机控制器驱动（Host Controller Driver）
- USB 驱动 (USB Driver)
- 主机软件 (host software)

主机控制器驱动的存在，方便地将各种不同的主机控制器实现映射到 USB 系统，客户可以不必知道设备到底接在哪个主机控制器上就能同设备进行通信。USB 驱动提供了基本的面向客户的主机界面。在 HCD 与 USB 之间的接口称为主机控制器驱动接口(Host Controller Driver Interface HCDI)。这层接口不能被客户直接访问，所以也不是由 USB 具体来完成的。一个典型的 HCDI 是由支撑各种不同主机控制器的操作系统来定义的。

USBD 提供 I/O 请求包(I/O Request Packets)形式的数据传输，以某一特定通道来传输数据。另外，USBD 为它的客户提供一个容易被支配及配置的抽象的设备。作为这种抽象的一部份，USBD 拥有标准通道(参见第 5 章及第 9 章)对设备进行一些标准的控制。这标准通道实现了 USBD 与抽象设备之间的逻辑通信。(见图 10-2)

在有些操作系统中，提供了额外的非 USB 系统软件以支持设备的配置及设备驱动程序的加载。在这样的操作系统中，设备驱动程序应使用提供的主机软件接口而不是直接访问

USBDI。

客户层描述的是直接与 USB 设备进行交互所需要的软件包。当所有的设备都已连上系统时，这些客户就可以直接通设备进行通信。一个客户不能直接访问设备的硬件。

该言之，主机可提供如下的功能

- 检测 USB 设备的连接与断开。
- 管理主机与设备之间的标准控制流。
- 管理主机与设备之间的数据流。
- 收集状态及一些活动的统计数字。
- 控制主机控制器与 USB 设备的电气接口，包括提供有限的能源。

在下面的章节中，我们将较细的阐述 USBDI 所能提供的功能。对于特定的主机平台与操作系统下的实现接口请参照相关的操作系统手册。

所有的集线器都通过状态改变通道报告它的状态的改变，其中包括设备的连上与断开等。 USBD 的一类特殊客户即：集线器驱动器拥有这些状态改变通道，接收这些状态的改变。对于像设备连结这种状态改变，集线器驱动器将加载设备的驱动程序。在有些系统中，这种集线器驱动程序是操作系统提供的主机软件的一部份，它用来管理设备。

10. 1.2 控制机构

控制信号可通过带内信号（in-band-singling）及带外信号(out-of-bard-signaling)两种方式在主机与设备之间传输。带内信号将控制信息及数据信息混在一起用同一通道传输，以至于主机根本就没有觉察到。而外带信号是通过单独的通道进行传输。

任何一个已连接的设备都有一个标准的信息通道，即标准通道。这个主机与设备之间的逻辑的连接用于传输 USB 的标准控制信息，比如对设备的配置信息等。这些标准通道为 USB 的设备提供了标准的接口，它也可以用来进行基于特定设备而不同的通信，这些通信由拥有所有这些通道的 USBD 作媒介。

一些特定的设备可能允许使用额外的信息通道来传输特定设备的控制信息。这些额外的信息通道与标准通道使用同样的协议，但是传递的信息是基于特定的设备的，也不是由 USB 具体标准化的。

USBD 支持和它的客户共享使用标准通道，它还提供给客户与设备相连的其它控制通道的访问。

10.1.3 数据流

主机控制器在主机与 USB 设备之间传递数据。这些数据被看作连续的字节流。USB 支持 4 种形式的数据传输

- 控制传输。
- 同步传输。
- 中断传输。
- 块传输。

有关于传输方式的额外信息请参见第 5 章

每个设备具有一到多个界面以用于客户与设备之间的数据传输。每个接口由一到多个在客户及设备端点之间独立传输的通道组成。USBD 根据主机软件的请求来初始化这些通道和接口。当这些配置请求提出后，主机控制器将基于主机软件所提供的参数来提供服务。

每个通道基于数据传输模式和请求的有如下几个特性：

- 数据传输的频率。
- 数据是以恒定速率提供还是随机出现的。
- 在数据传输前可延迟的时间。
- 在传输过程中数据的丢失是否是具有灾难性。

USB 设备的端口描述了与之相连接的通道的特性。USB 设备端口的特性的具体描述可参照第 9 章。

10.1.4 收集状态及活动统计数据

作为普通的为所有主机与设备之间的控制流与数据流服务的 USB 系统与主机控制器，一直处于随时接收状态变化及活动信息的状态，以使软件能及时接收并处理这些状态的变化。这里并不具体讲述需要被跟踪的状态信息及这些状态信息的特殊的格式。

10.1.5 电气接口因素

主机为连在集线器上的 USB 设备提供能量。一个集线器口所能提供的能量具体值在第七章有详细说明。

10.2 主机控制器功能

在所有的实现中，主机控制器都必须提供基本相同的功能。主机控制器对主机及设备来讲都必须满足一定的要求。下面是主机控制器所提供的功能的概况。每种功能在下面的小节中还有具体的说明。

1 状态处理(State Handling) 作为主机的一部份，主机控制器报告及管理它的状态。

2 串行化与反串行化　对于从主机输出的数据，主机控制器将协议及数据信息从它原始形状转换为字位流。而对于主机接收的数据主机控制器进行反向操作。

3 帧产生(Frame Generation)　主机控制器以每 1ms 为单位产生 SOF 标志包。

4 数据处理　主机控制器处理从主机输入输出数据的请求。

5 协议引擎　主机控制器支持 USB 具体规定的协议

6 传输差错控制　所有的主机控制器在发现和处理已定义的错误时展现相似的行为。

7 远程唤醒　所有的主机控制器都应具有将总线置于挂起状态及在远程唤醒事件下重新启动的能力。

8 集线器　集线器提供了标准的将多个 USB 设备连到主机控制器的功能。

9 主机系统接口　主机控制器在主机系统控制器之间建立一个高速的数据通道。

下面的各节将对上面提到的各功能进行详细的讨论。

10.2.1 状态处理

主机控制器具有一系列 USB 系统管理的状态。另外，主机控制器为下面两个与 USB 有关的部份提供接口。

- 状态改变传播
- 根集线器

根集线器提供与其它 USB 设备一样的标准状态给集线器驱动器。有关 USB 状态与其它之间的相互关系的详细讨论请参照第 7 章。

主机控制器的总的状态与根集线器及总体的 USB 密不可分。任何一个对设备来说可见的状态的改变都应反映设备状态的相应改变。从而保证主机控制器与设备之间的状态是一致的。

USB 设备通过使用恢复信号请求唤醒，使设备回利已配置的状态。主机控制器本身也可以通过同样的方法产生一个恢复事件。主机控制器通过使用该实现系统的某种机制来通知主机的其它部份已产生了一个恢复事件。

10.2.2 串行化与反串行化

通过物理上的传输是以字位流的形式出现的。不管是作为主机的一部份，还是作为设备的一部份，串行接口引擎(STE)处理 USB 传输过程中的串行化与文串行化工作。在主机上，串行接口引擎是主机控制器的一部份。

10.2.3 帧产生

主机控制器有义务将 USB 时间划分为以 1ms 为单位的帧。主机控制器以每 1ms 间隙产

生SOF(Start-of-Frame)标识以示新的一帧的开始(如图10-3)。SOF标识是一帧的开始部份，在 SOF 标识之后主机控制器在该帧的余下时间内传输其它的东西。当主机控制器处于正常工作状态时，SOF 标识必须以 1ms 为间隙连续的发送而不管其它的总线活动。当总线控制器处于不给总线提供能量的状态时，它不能产生 SOF 标识。当总线控制器不产生 SOF 标识时，它处于一种节能方式。

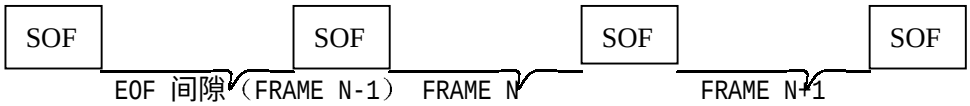


图 10-3　帧产生

SOF 标识具有取得总线的最高的优先权。集线器中的 babble circuit 在 EOF 期间停止任何传输任务，为 SOF 标识的传输提供一条空闲的总线。

主机控制器必须允许 USB 每帧的时间长度相差±1 bit 的时间(参见 10.5.3.3.4 节)。主机控制器维持着一个当前帧的序号。

帧序号具有以下各方面的作用

• 用于将两帧唯一的区别开

• 在每帧的结尾处加 1

• 对于它的后继帧有效

主机在每一个 SOF 标识中传输当前帧号的低 11 位。当接到主机控制器的请求时，主机返回请求发生时刻的帧序号。虽然主机控制器自身并不要求维持一个超过 11 位的帧序号，但是主机返回的帧序号至少是 32 位的，。

主机控制器在 EOF 期间要停止一切传输操作。当 EOF 间隙产生时，所有原定在刚才那帧上传输的事务暂停。如果主机控制器在执行传输的时候出现了 EOF，主机控制器中止该项传输请求。

10.2.4　数据处理

主机控制器接收来自 USB 系统的数据并将其传送给 USB 设备或从 USB 设备接收数据送给 USB 系统。USB 系统和主机控制器之间进行数据传输时的具体格式是基于具体的实现系统的，同时也要符合第 5 章所讲述的传输协议要求。

10.2.5　协议引擎

主机控制器管理着 USB 协议层的接口。在输出的数据中插入适当的协议信息，并且它将解释并去除输入数据中的协议信息。

10.2.6　传输差错控制

主机控制器必须能够发现如下的几种从主机的角度定义的错误：

• 超时错。该类错误发生在目标端口没有相应的反应或传输系统被严重损坏以至于目标端口根本就没有收到信息。

•数据丢失或无效传输。 主机控制器发送或接收到较应该传输的数据包为短的数据包。例如，一项传输超出了 EOF，或缺少可使用的资源，或数据包 CRC 校验出错.

• 协议错

——无效的握手 PID，例如形式错误或不恰当的握手包。

——错误的包标志。

——位插入错。

对批传输，命令传输，中断传输，主机必须维持一个错误统计记录。这些错误是由上述各种情况产生的，而不是由于端口不响应一个请求而产生的。错误统计记录反映了传输过程中出现的错误的次数。如果错误统计记录值达到三，则主机终止传输。如果传输是由于过多的传输错而被中止的话，则最后一次出现的错误将被简要的说明。每个同步传输事

务仅进行一次，而不管结果，所以对于这种传输来说是没有被维持的错误记录。

10.2.7 远程唤醒

如果 USB 系统希望将总线置于挂起状态，它将请求主机控制器终止任何形式的传输，包括 SOF。这使得所有的 USB 设备进入一种挂起状态。在这种状态下，USB 系统可以使能主机控制器响应总线唤醒事件。这使得主机控制器能响应总线的唤醒信号，重启主机系统。

10.2.8 根集线器

根集线器提供主机控制器与一个或多个 USB 设备的连接。除了主机控制器及根集线器之间的硬件软件接口是由具体的硬件实现来定义的外，根集线器提供与其它的集线器一样的功能(参见 11 章)。

10.2.8.1 端口复位(Port Resets)

7.1.7.3 节讲述了集线器为了保证每一个下行恢复请求都供给一个长时间的下行复位，所必须具备的条件，根集线器应能提供一个至少为 50ms 的复位时间。如果复位时间是由硬件控制的，并且硬件能提供的复位时间小于 50ms，则 USB 系统可以产生连续的几个复位信号以产生足够长时间的复位。

10.2.9 主机系统接口

主机控制器提供一条高速的读出与写入系统内存的总线接口。内存与 USB 电缆的物理数据的交换是在主机控制器的控制下自动进行的。当数据缓冲区需要允满或清空时，主机控制器通知 USB 系统。

10.3 软件功能概论

HCD 与 USBD 提供了基于不同抽象层次的软件界面。它们以一定的方式协同工作以实现 USB 系统的功能(见图 10-2)。对 USB 系统的功能主要表现在对 USBDI 所能提供的功能上。USBD 与 HCD 之间任务的划分没有具体的定义，但是 HCDI 必须要具备的一项功能就是它必须支持多种主机控制器的不同实现。

HCD 提供了抽象的主机控制器，且对主机控制器所见到的 USB 系统的数据传输进行了抽象。USBD 提供一个抽象的设备，且对 USBD 客户和 USB 设备功能部件之间的数据传输进行抽象。总之，USB 系统简化了 USBD 客户与 USB 设备之间的数据传输过程，并且作为对 USB 设备的面向 USB 的接口进行控制的人口。作为简化数据传输功能的一部份，USB 系统提供缓冲区管理功能，并且允许根据客户及设备功能部件的需求进行实时数据传输。

USBDI 的提供的功能将在下面将具体的阐述。这些功能的具体实现将在相关的操作系统中关于 HCDI 及 USBDI 的部份找到。通过 USBDI 实现数据传输所经历的过程将在下面得到简述。

10.3.1 设备配置

不同的操作系统环境使用不同的软件及不同的事件序列来配置设备：在这里，我们讲述 USB 系统层时，并不以某个特定的实现系统为模板。但是 USB 系统的具体实现系统必须实现某一些基本的功能。在有些操作系统中由主机软件(host software)实现这些功能，而在其它的一些操作系统中，由 USB 系统层实现该项功能。

USBD 有一类特殊的客户，称作集线器驱动器。集线器驱动器充当设备从特定的集线器连接与断开的清算中心(clearing house)。当集线器控制器接到有设备连接与断开的信息时，它会激活其它的主机软件或其它的 USBD 客户配置该设备或对这些信号作出其它的反应。上述的模型是下面讨论的基础，见图 10-4

当一个设备连上某集线器的时候，集线器驱动器从集线器那儿收到一条表明集线器状态改变的消息。集线器驱动器使用集线器提供的信息，向 USBD 请求设备的标志码(identifier)。USBD 为该设备建立标准的通道，并且返回该设备的身份码。现在该设备可以被配置和使用。对于任何一个设备，在使用之前必须要完成如下三项配置工作：

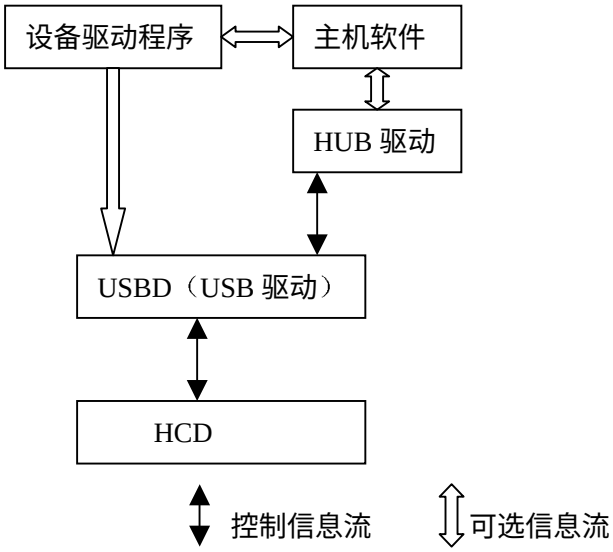


图 10-4　设备配置信息流程图

1.设备配置：这包括设置设备的所有参数，为设备分配所有可见的 USB 主机资源。USB 还允许以后更改设备的部分配置参数。一旦设备配置完以后，以设备的观点来看，它已是可用的。

2.USB 配置　为了真正的创建客户可用的 USBD 通道，额外的对设备不可见的 USB 信息要由客户具体化。这些信息被称为通道策略，它描述了客户将如何使用该通道，其中包括一个 I/O 请求包容许的最大数据量，客户的最大服务间隙，客户的标识码等等。

3.功能配置　当设备配置与 USB 配置完成后，从 USB 的观点来看，该通道是完全可用的。然而在客户真正使用通道之前，或许额外的供应商定义的(vendor)或有关设备类(class-specific)配置需要进行。这些配置是客户与设备之间的私事，它并不由 USBD 来进行标准化。

下面的段落描述了客户配置与 USB 配置需要完成的工作。

配置软件完成具体的配置工作。具体的配置软件将基于特定的操作系统实现而有所不同，但一般它包括以下几个部份。

- 集线器驱动器
- 其它主机软件
- 设备驱动程序

配置软件首先读取设备的描述符，然后对于每一种可能的配置请求该配置的描述信息。配置软件使用提供的信息加载特定的客户，比如设备驱动程序，由它与设备进行初始的交互。配置软件可能要从设备驱动程序那儿输入一些信息，然后为设备选择一种配置方案。设备配置过程设置好设备的所有端口，并且返回一组用于 USBD 客户传输数据的接口。一个接口是为某个客户所拥有的一组通道。

初始配置为每个接口使用缺省的配置，为每个端口分配缺省的带宽。 USBD 实现系统可能允许客户在一些可选配置中指定一种作为初始配置。同时，USB 系统会检测端口所需要的资源是否能得到满足，如果能满足的话，就为它分配所需要的资源（参见 10.3.2 节关于资源管理)。

至此，设备已经配置完成了，但是创建的通道还不可用。当客户通过设置通道的使用策略为每一个通道初始化以后，USB 配置也就完成了。这些需要被具体指明的使用策略包括客户的最大服务间隙和客户的标识信息。根据 USB 配置的结果，USB 系统决定除了客户提供

的数据缓冲区外还需的工作缓冲区的大小。工作缓冲区的大小是根据客户的功能选择及每次事务的需求来决定的。

当 I/O 请求包结束的时候，或者是顺利结束，或者是由于产生了错误，客户都将收到一条通知消息。在 USB 通知下，客户唤醒以检查即将到来的 I/O 请求包的状态。

客户可以进行一些配置的修改，比如说选择另一种可能的接口配置方案，改变已为某一通道分配的带宽。在进行这些配置更改时，相应的接口或通道必须处以空闲状态。

10.3.2 资源管理

当一个 USBD 为一个给定的通道进行设置的时候，USB 系统将根据相应端口的描述符检查它是否能满足该通道的资源要求。其中必须被满足的一项是通道的带宽。在检查带宽是否能满足时中要经过两步。首先计算每次事务的最大运行时间，其次根据帧的时间分配表检查上述的带宽要求能否满足。

USB 系统的软件完成为同步传输模式及中断传输模式分配所需的带宽和判断一个特定的控制传输事务和块传输事务能否装进一个给定的帧的工作。如果主机控制器的实际事务的执行时间超过了一帧 ，主机控制器有义务维护帧的完整性(参照 10.2.3)。下面的讨论叙述了 USB 系统应具备的功能。

为了决定带宽是否能满足，或者传输事务能否被装入特定的帧，必须计算事务最大执行时间。在计算过程中使用到下述信息

- 每包的最大数据字节数
- 传输模式
- 拓扑结构深度。如果要求不是很精确的话，假定设备处于最大的拓扑深度。

上述的计算必须考虑到位传输时间，由于拓扑深度而造成的信号传输延迟，还有一些与具体实现系统有关的延迟。比如主机控制器的准备及复位时间。具体的计算公式请参照第 5 章。

10.3.3 数据传输

客户与功能部件之间通信的基础是如下的接口：与特定 USB 设备相联系的一串相关通道。

主机上的一个客户拥有一个给定的设备接口。客户初始化该接口，为接口中的每个通道设定使用策略，其中包括设定一次 I/O 请求的最大数据量，和通道的最大服务间隙(servile interval)。服务间隙是以毫秒计的，它反应了在同步传输模式下每两次传输的间隙，同时它反映了在中断传输模式下轮询间隙。当一个特定的请求处理完成以后，客户被唤醒。客户通过管理每一个 I/O 请求包的大小以维持它的任务循环(duty cycle)和延迟限制(latency)。其它的使用策略信息包括客户的通知信息等。

客户为传输事务提供数据缓冲区。USB 系统根据客户的使用策略信息决定额外的工作缓冲区大小。

客户视它的数据为一个连续的串行数据流。客户以类似于其它总线的数据流管理方式管理这种数据流。在内部，USB 系统可能根据自己的策略及主机控制器的限制将单个请求分割成多个在 USB 上传输的请求。但是当 USB 系统决定进行请求的分割的时候，必须要考虑到以下两点：

- 将数据流分割成更小的块操作对客户是不可见的。
- USB 样本(Samples)在总线传输时是不被分割的。

当客户想传输数据的时候，它将 I/O 请求包发送给 USBD。同时，客户将根据传输的方向提供一个空或满的数据缓冲区。当请求完成(或是顺利完成，或是出现了错误)I/O 请求包及状态都将返回给客户。如果有必要的话，IRPS 中将包括每次事务的状态。

10.3.4 普通数据定义(Common Data Definition)

为了使客户最直接的从它的设备处取得结果，有必要使客户与设备之间传输过程中的数据拷贝及处理降至最少。为了便于上述的实现，I/O 请求包中的控制信息被标准化，以使得不同的层能直接访问到客户提供的控制信息。数据的具体格式与操作系统上 USBDI 的具体实现有类。有些数据成份可能对客户来请是不可见的，但是在客户提出请求的时候生成。

下面的数据成份定义了一次请求的相关信息。

- 与该请求有关的通道标识。通道的标识信息同样也包括了传输模式的信息。
- 特定客户的通知标志码。
- 数据缓冲区的位置及长度
- 请求的结束状态，包括总的状态及每次事务的结束状态
- 工作缓冲区的位置及长度。这与其实现系统有关。

客户与 USBD 之间交互请求的机制是由操作系统定义的。除了上面讲到的一个 IRPS 必须包含的请求信息外，还必须对请求的处理作出一些要求。这些基本的处理要求在第 5 章有描述，读者可以参考。另外，USD 提供一套机制以指定一组同步 I/O 传输请求的首次传输事务出现在同一帧内。USBD 还提供了一套机制以指定一组不可被中断的厂商定义的或有关设备类的请求到标准通道。没有其它的请求可以插入到该组非中断请求的执行流中。如果一组中间的某些请求出现了错误，整的该组请求都被中止。

10.4 主机控制器驱动器

HCD 是主机控制器硬件的抽象，同时也是对主机控制器所见的数据传输的抽象。HCDI 应符合下列条件：

- 提供一个抽象的主机控制器硬件。
- 提供主机控制器在 USB 上传输数据过程的抽象。
- 提供主机控制器为给定设备分配(或不分配)必需的资源过程的抽象。
- 根据一般集线器的定义提供根集线器。这包括支持集线器驱动程序与根集线器的直接交互。具体的说，虽然一个根集线器是用硬件及软件共同实现的，它最初响应缺省的设备地址(从客户的观点)，返回描述符信息，支持设备地址集，并且支持其它的集成器类型的请求。然而考虑到可能将主机控制器与根集线器集成在一起，所以，在对根集线器进行访问的时候可能并不需要经过总线。

HCD 提供软件界面 HCDI（HCD INTERFACE）以实现必要的抽象。HCD 的功能是进行抽象、隐藏主机控制器的硬件细节。在主机控制器之下是物理的 USB 及所有与之连接的 USB 设备。HCD 是 USB 软件中的最下一层。HCD 只有一个客户：USB 驱动器(USBD)。USBD 将客户的请求映射到相关的 HCD。一个给定的主机控制器驱动器可能管理很多的主机控制器。

客户不能直接访问 HCDI，所以 HCDI 的具体实现细节将不在下面作具体的讨论。

10.5 USBD

USBD 提供了供操作系统组件特别是设备驱动程序访问设备的一组接口。这些操作系统组件只能通过 USBD 来访问 USB。USBD 的具体实现基于不同的操作系统。下面的讨论将以 USBD 的实现系统所必须提供的基本功能为中心展开。对于在具体某一特定的操作系统环境下的 USBD 细节请参考相关的操作系统手册。一个 USBD 可以访问一个或多个 HCD，而一个 HCD 可能与一个或多个主机控制器相连。某些操作系统可能允许对 USBD 的初始化进行一些设置。从客户的观点来看，与客户进行通信的 USBD 管理着所有连接着的 USB 设备。

10.5.1 USB 概况

USBD 的客户直接命令设备或从通道直接输入和输出数据流。USBD 为客户提供两组工具。命令工具和通道工具。

命令工具允许客户配置和控制 USBD 操作同时配置及控制 USB 设备。命令工具提供了对设备标准通道的所有访问。

通道工具允许 USBD 客户管理特定设备的数据和控制数据的传输。通道工具不允客户直接访问设备的标准通道。

图 10-5 给出了 USBD 的总体框架。

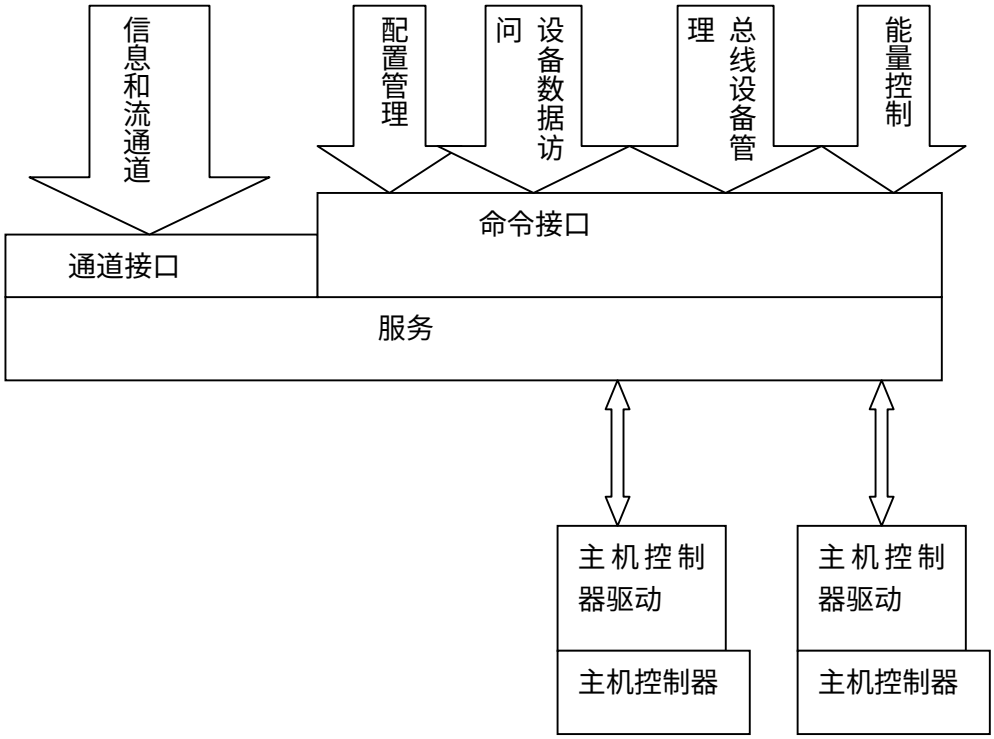


图 10-5 USB 驱动结构

10.5.1.1 USBD 初始化

具体的 USBD 初始化工作是与操作系统有关的。当 USB 系统初始化时，USB 的管理信息被创建，其中包括缺省地址(default address device)设备及它的标准通道。

当一个设备连上 USB 时，它响应特殊的缺省地址(参见第 9 章)直到他的唯一地址由主机给出。为了让 USB 系统能与新的设备进行通信，设备在刚连上总线时，设备的却省地址必须是可用的。在设备的初始化期间，设备的缺省地址被修改成一个唯一的地址。

10.5.1.2 USBD 通道使用

通道是设备与主机的逻辑连结。一个通道由且仅由一个客户所拥有。虽然通道的基本属性不因通道的拥有者而改变，但在两类不同客户所拥有的通道之间还存在一定的差异：

- 标准通道（default pipe）他们由 USBD 拥有和管理。
- 其他的通道(all other pipes) 它们由 USBD 的客户拥有和管理。

虽然标准通道经常用于完成一些客户通过命令接口所传递的请求，但是它们不能由客户直接访问的，。

10.5.1.2.1 标准通道

USBD 有义务分配及管理恰当的缓冲区以支持对客户不可见的标准通道上的数据传输。比如设置设备的地址等。对于一些客户可见的传输，比如像传送厂商定义或有关设备类的命令，或者读取设备的描述符，客户提供必须的数据缓冲区。

10.5.1.2.2 客户通道

所有不是被 USBD 拥有及管理的通道都可由客户拥有及管理。从 USBD 的观点看，一个单独的客户拥有该通道。实际上可以是一组用户管理通道，只要该组用户在使用通道时表

现出单个客户的行为。

客户有义务根据它所能取得的服务间隙内的数据提供一个足够大小的数据缓冲区。额外的工作缓冲区由 USB 系统给出。

10.5.1.3 USBD 服务功能(Service Capabilities)

USBD 提供如下种类的服务：

- 通过命令工具配置设备
- 通过命令工具及通道工具提供传输服务
- 事件通知
- 状态报告及错误恢复

10.5.2 USBD 命令工具功能

USBD 命令工具允许客户访问设备。通常，这些命令允许客户以读写形式访问某个设备数据及控制部份。客户要做的仅是提供设备的标识码和相关数据缓冲区或空缓冲区指针。

USBD 命令传输时并不需要 USB 设备是已被配置好。USBD 提供的设备配置设施大部分使用是用命令传输。

下面的是有关命令工具提供的功能。

10.5.2.1 接口状态控制

USBD 客户必须能够设置具体的接口。接口状态的改变使得所有与该接口相连的通道都进入新的状态。另外接口的所有通道都能够被重新设置或废弃。

10.5.2.2 通道状态控制

USBD 通道状态由两部份组成

- 主机状态。
- 反映的端口状态。

通道状态值总是包含上述两部份。USBD 的客户管理由 USBD 报告通道状态，且客户能够与端口交互以改变它状态。

一个 USBD 可以处于如下的状态之一

- 活跃态，通道已经过初始化　设置好使用策略且能够传输数据。客户可以询问一个特定的通道是否正在处理 I/O 请求。一个不在处理 I/O 请求的端口也可认为处于活跃态，只要它能接收新的 I/O 请求包。
- 挂起状态　通道出现错误。该状态也可能反映了设备上的相关端口被挂起。

只要设备已被配置且通道及设备相关端口没有被挂起，相应通道及端口就可被认为是处于活跃态。客户可以通过如下的几种方式来操纵通道状态。

- 废弃一个通道：中止正在等待该通道传输的所有 I/O 请求包，且在 IRPS 上标上相应废弃的记录后，返回相应客户。废弃通道不改变主机状态及相应的端口状态。
- 重设通道(resetting a pipe)　通道的 I/O 请求包被中止。主机的状态处于活跃态。如果将相应的端口状态改变的话，由 USBD 客户显示命令。
- 清除被挂起通道　通道的状态从挂赶转移到活跃态
- 挂起通道　通道的状态被设置为挂起态。

10.5.2.3 获取描述码

USBD 提供取得标准设备描述符，设备配置描述符，字串描述符和设备类或者厂商定义的描述符的功能。

10.5.2.4 取得当前配置参数

USBD 提供取得任何设备当前配置参数的描述符的功能。如果设备没有被配置，则不返回描述符。当客户请求为设备设置特定的参数时也返回当前配置的描述符。返回的配置信息中包括如下一些内容：

• 所有存放在设备上的配置描述符，包括接口的所有其他可替换配置。

• 返回接口的当前的配置的描述符。

• 接口当前配置中的某一端口(一个接口可能具有多个端口)的通道句柄(pipe handle)

• 接口当前配置中的某一端口的最大允许包长。

另外，对于任一通道，USBDI 必须提供返回该通道的当前正使用的最大包长的值的机制。

10.2.5.5 增加设备

USBDI 必须提供某种机制以便于当增加新的设备的时候，集成器驱动器能通知 USBD 并能取得该新 USB 设备的 USBD 标识。USBD 的任务包括分配设备地址并且为设备准备使用的标准通道。

10.5.2.6 设备断开

USBDI 必须提供某种机制以便于集线器控制器通知 USBD 特定的设备已断开。

10.5.2.7 管理状态

USBDI 必须提供取得和清除设备或接口或通道上与设备有关的状态的功能。

10.5.2.8 向设备发送与设备类有关的命令

USBD 的客户，特别是特定类的和自适应的驱动器使用 USBD 提供的该种机制向设备送出一个或多个设备类命令。

10.5.2.9 向设备发送特殊的厂商定义的命令

客户使用 USBDI 提供的该种机制向设备送出一到多个厂商定义的命令。

10.5.2.10 更改接口配置

USBDI 必须提供更改特定接口配置的机制。修改了配置后，接口的新通道句柄替换了旧的通道句柄。在上述的请求执行的过程中，接口必须是空闲的。

10.5.2.11 创建设备配置

配置软件向 USBD 提出进行设备配置的请求时提供一个包含配置信息的数据缓冲区。USBD 根据提供的配置信息为设备端口请求资源。如果所有的资源请求都得到满足，USBD 设置设备配置参数，并且返回当前设备所有活跃态接口句柄和与该接口中的某一端口相连的通道句柄。接口的设置可使用缺省参数。

注：进行配置的接口可能要求明确指明其配置参数。

10.5.2.12 设置描述符

对于支持该项行为的设备，USBDI 允许升级设备上的描述符或者增加新的描述符。

10.5.3 USBD 通道设施

USBD 的通道设施使客户与设备之间高速的低附加信息的数据传输成为可能。数据传输的高性能是通过将 USBD 的一部份通道管理任务转交给客户来实现的。所以通道设施比 USBD 命令设施所提供的数据传输服务更直接。通道设施不允许访问设备的标准通道。

只有在 USB 及设备的配置都顺利完成后，客户才有可能进行 USBD 的通道传输。当设备被配置的时候 USBD 根据配置参数为设备的所有通道请求资源。当特定的接口或通道空闲的时候，客户可以更改配置。

客户为输出的通道提供一个满的数据缓冲区，并且在请求完成以后取得传输状态信息。客户可以根据返回的状态信息判断传输是否顺利完成。

客户为输入通道提供空的数据缓冲区，并且在请求完成以后得到一个具有数据的缓冲区及传输的状态信息。客户可根据该状态信息判断传输数据的数量及质量。

10.5.3.1 支持的通道类型

根据所支持的数据传输类型，共有四种通道类型。

10.5.3.1.1 同步数据传送

所有排队等待某同步传输通道进行传输的数据缓冲区都被被看作是一个样本流。像所

有其它的通道一样，客户为同步通道提供一个通道使用策略，包括客户的服务间隙。当在输入过程中丢失字节或在输出过程中发现有传输问题时，客户都将被告知。

开始数据流的传输前，客户先提供第一个数据缓冲区，，为了维持同步传输的连续性，客户在当前的缓冲区还没有被允满或取空时就提供第二个数据缓冲区。

USBD 应能够将客户的数据流看作是一个样本流，也就是说，在数据传输过程中实际的打包解包过程对客户隐敝了。另外，一次传输事务经常局限于客户的某数据缓冲区。

对于一个输出通道，客户提供一个满的数据缓冲区。USBD 使用客户提供的同步方法，将在服务间隙中要传输的数据分配到在各帧中。

对于一个输入通道，客户必须提供一个足够大的空的缓冲区，以容纳下在服务间隙时间中所传输的最大数据量。当丢失数据或得到无效字节的时候，USBD 必须在该字节应该放置的缓冲区位置上留下一个空缺，并且将错误消息发给客户。不使用同步方法的一个后果就是保留的空间被认为就是最大的包长。缓冲区结束的通知信息在 IRP 结束的时候产生、注意，当输入缓冲区返回给客户的时候，它不一定是满的。

USBD 可能提供看待同步数据流的其它方法(除了以样本流的方式看待同步数据流)，USBD 同时也必须能将用户的数据流看作是包的流(Packet Stream)。

10.5.1.2 中断传送

中断数据输出是从 USBD 的客户输出到 USB 的设备。中断数据输入是由 USB 的设备输入到 USBD 的客户。USB 系统保证中断传输能满足设备端口描述符所规定的最长延迟时间的限制。

客户提供一个足够大的数据缓冲区保存中断传输的数据(通常一次 USB 传输)。当所有的数据传输完毕，或是错误太多，超出了客许的最多错误次数，IRP 被返回给用户。

10.5.3.1.3 块传送

块传输可以始于客户，也可以始于设备，它没有规定的延迟时间限制和周期。当所有的数据传输完或是由于错误太多 IRP 被返回给客户。

10.5.3.1.4 控制传送

所有的消息通道支持双向传输数据。在所有的情况下，客户先与设备进行 SETUP 阶段的通信。数据阶段的通信是可有可无的，或是输入，或是输出。最后状态返回主机。关于控制传送协议的具体情况参照第八章。

客户准备缓冲区，且根据需要提供一个空的或是满的数据缓冲区，然后开始 SETUP 阶段的第一步(一阶段由几步（phrase）完成组成)。当控制传输的所有步骤都已完成以后，客户接到一个传送结束的消息，或是出现错误的而接到一条错误指示消息。

10.5.3.2 USBD 通道设施功能

下面是所提供的通道设施功能。

10.5.3.2.1 废弃 IRPS

USBDI 允许等待在某一个特定通道传输的 IRP 被废弃。

10.5.3.2.2 管理通道策略

USBD 应能允许客户清除或者设置特定通道或整个接口的使用策略。在成功设置通道使用策略之前的所有 IRP 都将被 USBD 拒绝。

10.5.3.3.3 IPRS 排队等待

USBDI 应能允许客户向特定的通道提出 IRPs 请求。在当前的 IRPS 请求还没有完成的时候，客户可以提出新的 IRPS 请求，多个 IRPS 请求排队等待。当 IRPs 返回给客户的时候，请求状态同时也被返回。USBD 提供了一种机制以指定一组同步传输的 IRPs 使它们的首次转送事务出现在同一帧内。

10.5.3.2.4 申请成为 USB 系统的主客户

主客户可以对每一帧时间作出微小的调整，从而使 USB 设备比如 ISDN 口与主机同步。请求成为主客户的客户必须说明自己正在控制的设备接口句柄。

USBDI 允许客户请求成为该 USB 系统的主客户，并且当客户不需要的时候，使之放弃该项功能。USBD 只能将主客户的地位给予一个客户。当当前的主客户没有放弃控制权的时候，所有请求成为主客户的请求都被忽略。主客户可能显式的放弃控制。或者是由于主客户的设备被复位或断开，主客户自动地放弃其主控地位。

10.5.4 通过 USBD 设施管理 USB

使用提供的 USBD 功能设施，USB 系统通常支持下面的基本功能。

10.5.4.1 配置服务(Configuration Server)

配置服务基于每个设备。配置软件告诉 USBD 何时配置设备。一个集线器驱动器在设备管理中扮演特殊的角色，它至少支持如下的功能。

- 设备联接，断开通知。它是由集线器驱动器的中断通道所驱动。
- 设备复位　由集线器驱动器重新设置集线器口的设备的上传流来实现的。
- 通知 USBD 给设备设置唯一的地址。
- 能量控制。

USBD 还提供如下的配置功能，可由集线器驱动程序所使用或由其它的配置软件使用。

- 设备标识和访问配置信息(通过访问设备上的描述符)
- 通过命令设施来配置设备

当集线器驱动器通知 USBD 有新的设备连接，USBD 为新的设备创建标准通道。

10.5.4.1.1 配置管理服务

配置管理服务主要通过提供一组在标准通道上产生数据传送事务的接口命令来实现。值得注意的一个例外是集线器使用另外的中断通道直接将它的状态传递给集线器驱动器。

任何一个集线器在它的某一口的状态发生改变的时候初始化一次中断传输。通常，集线器口的状态改变是由于 USB 设备的连接或断开(详情请参考 11 章)

10.5.4.1.2 设备初始配置

设备配置过程在设备连上集线器的某一口时开始，集线器通过它的状态改变通道报告自己某一口的状态的改变。

配置管理服务允许配置软件从 USB 设备的一串可选配置中选择其中的一个作为设备的配置。USBD 在真正设置配置信息之前检查是否能提供足够的能量（power）并且检查数据的传输速率是否超出了 USB 的当前能力范围。

10.5.4.1.3 修改设备的配置

配置管理服务允许配置软件更改设备的配置。当 USB 能提供足够的能量，且数据传输速率也在 USB 的能力范围之内，USB 便完成更改配置的操作。如果新的配置被拒绝，则设备保留原先的配置。

配置管理服务允许配置软件将设备的状态返回到没有被配置的状态。

10.5.4.1.4 设备断开

当集线器通过它的状态改变通道报告设备断开时，错误恢复或者设备断开处理过程便开始了。

10.5.4.2 总线及设备管理

总线及设备管理服务允许客户成为 USB 的主客户，并且作为 USB 的主客户，可以调节总线上传输的每一帧的时间。一个主客户可能为当前传输的帧增加一 bit 或削减一 bit 的时间。

10.5.4.3 能量控制

对 USB 来说，有紧密接合的两个层次的能量控制：总线层及设备层的能量控制。这种

具体化提供更有效管理 USB 总线能量的设施。设备类可能定义与特定类有关的能量控制功能。

所有的设备都支持挂起状态(参见第 9 章)通过控制设备所连到的集线器口可以使设备处于挂起(suspend)状态。在挂起状态下，普通的设备操作都中止了，然而，如果设备支持远程唤醒且已经使能响应唤醒信号的话，它可以在外部事件的激励下产生恢复(resume)信号。

能量管理系统可能将设备置于挂起的状态或者干脆切断设备的能源以达到节能的目的。当设备在挂起和恢复的转换过程中，USB 并不提供保存和恢复设备状态的机制和功能。设备的类可能定义特定类的状态保存及恢复功能。

USB 系统协调设备的交互而不管设备是处于加电状态还是处于挂起状态。

10.5.4.4　事件通知

USBD 客户可以从不同的源接到不同的通知消息，其中有可能的事件源有如下几种：

• 客户发起的某些动作的结束

• 中断传输方式直接将设备的事件消息传递
给客户。例如集线器使用中断通道传递与集线器
状态改变有关的事件

• 标准的设备接口命令、设备类命令、厂商定义的命令以及普通的消息通道上的控制传输都可以被用来轮询设备的状态。

10.5.4.5　状态报告及错误恢复功能

USBD 的命令设施及通道设施都提供状态报告及错误恢复的功能。

另外，USBD 的客户可以通过命令设施取得 USB 设备的状态。

USBD 允许通道被重设置或废弃以提供客户通道错误恢复的功能。

10.5.4.6　管理远程唤醒的设备

USB 系统应使挂起的 USB 系统在恢复过程中消耗的能量最小。为了达到该目的，USB 系统显式的使能某些具备远程唤醒功能的设备，控制恢复信号在树型拓扑中的传播，有选择的唤醒挂起某些设备。

在某些错误恢复场合，USB 系统可能重新遍历子树，子树可能完全或部份被挂起。在错误恢复的过程中，USB 系统必须避免在设备恢复的过程中，向设备所连接的集线器口驱动 RESET 信号。通过管理设备的远程唤醒特征及集线器的口特征可以避免以上情况的发生，规则如下：

• 仅仅在有选择的挂起设备连接点与根集线器口之间的某一口之前，向叶子设备发生一个 SetDeviceFeature(DEVICE-REMOTE-WAKEUP)请求。

• 如果某集线器口被挂起，且挂在该口的设备具有远程唤醒的能力，那么在向该口发生复位命令前先使能该集线器口

10.5.5　将操作系统起动前 USB 的控制交给操作系统

只有一个软件驱动程序拥有主机控制器。如果主机系统在操作系统被加载之前实现 USB 服务，主机控制器必须提供一种机制，使得操作系统起动前的软件对主机控制器的访问无效，且将控制转交给操作系统。一旦操作系统取得控制权，它必须设置好总线。如果操作系统提供一种机制，将控制转再转交给系统起动前的环境，总线将处于一种不确定状态。操作系统起动前的软件可以认为这是一种加电状态(Powerup)

10.6　操作系统环境指南

正如前面所说的，USB 系统及主机软件之间的接口是基于特定的主机平台和操作系统的。特定的支持 USB 的主机平台及操作系统的组合提供了 USB 的详细说明。这说明描述了将 USB 完全溶合到主机所需的特定接口。任何一个支持 USB 的操作系统都提供全面的 USB

说明。

集线器规范

集线器规范包括两大基本部分：集线器转发中继器(Hub Repeater)和集线器控制器(Hub Controller)。同时也描述了集线器的错误恢复，重启和挂起/唤醒操作。最后简述集线器请求行为和集线器描述子。

集线器提供了 USB 设备和主机之间的电子接口。集线器支持的主要的 USB 功能有：

- 连接行为
- 电源管理
- 设备连接和未连接检测
- 总线错误检测和恢复
- 高速和低速设备的支持。

集线器由集线器转发中继器和集线器控制器组成。集线器转发中继器负责连接方面的工作。它也支持像总线错误检测和恢复，连接和未连接检测这样的异常处理。集线器控制器提供主机到集线器的通讯机制。集线器特定的状态和控制命令允许主机配置集线器和监视与控制它的每个下行端口。

集线器根据它们是在传输包、唤醒信号或者是在空闲状态而表现出不同的连接行为。一、包信号连接。包信号连接分为上行连接和下行连接两种。上行连接是面向主机的。当某个使能的下行端口检测到 SOP 时，就建立了仅到上行端口而不是任何其它下行端口的上行连接。下行连接是面向设备的。当集线器在上行端口上检测到 SOP 时就建立到所有使能下行端口的连接。未处在使能状态的端口不能向下行方向传送。集线器还有没有任何连接的空闲状态。在空闲态时集线器所有端口都处在接收模式，等待下一个包的开始。二、唤醒连接。挂起的集线器将上行端口接收到的唤醒信号送到所有使能的下行端口。当挂起的集线器检测到来使局部挂起或使能下行端口的唤醒信号时，将反射唤醒信号到上行端口和包括自己在内的所有使能下行端口。唤醒信号不被反射到未使能的或挂起的下行端口。后面将有更详细的讨论。

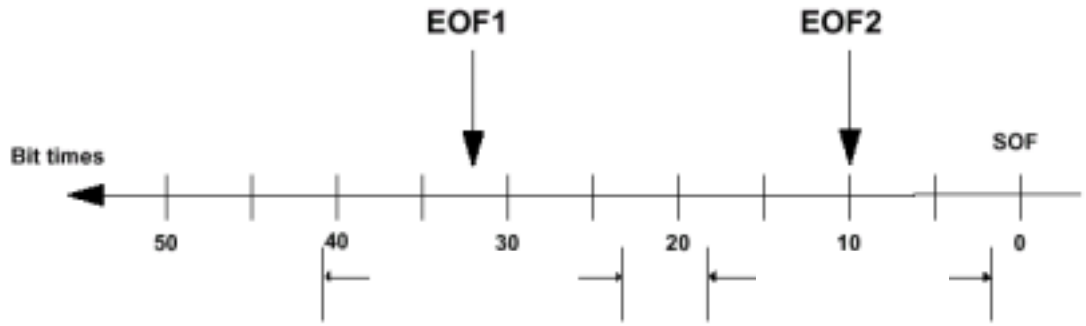
集线器是主机和设备之间建立连接的极为重要的 USB 部件。检测和防止任何连错误，尤其是能导致死锁的连接就显得非常重要。集线器必须在转发模式下处理连接错误。集线器必须也能检测和恢复发往集线器控制器的包。

每个集线器都有一个帧计时器。它的时间来自集线器的本地时钟，并通过主机发来的 SOF 与主机同步。它提供被用来检测闲置(babbling)设备和防止集线器被上行集线器设为无效的时间参考。集线器必须跟踪主机帧周期并能在即使丢失两个连续 SOF 令牌的情况下仍然保持同步。在重启或唤醒后，帧计时器未同步。只要接收到两个连续的 SOF 包，它就必须同步。

EOF1 和 EOF2 是由帧计时器产生的时刻。这些时刻用来确保设备和集线器不干扰来自主机的 SOF 包的正确传输。这些时刻仅当帧计时器与 SOF 同步时才有意义。图 10-1 给出了严格的 EOP 时刻。表 10-1 总结了主机和集线器 EOF 时刻。在 EOF2 时刻，任何有上行连接的端口将被置为无效。集线器通过在上行集线器 EOF2 之前发出 EOP 来防止被设为未使能。(如在 EOF1 时刻)。

表 10-1 集线器和主机 EOF 时刻

描述	从 SOF 开始通常的位数	说明
EOF1	32	帧结束点#1
EOF2	10	帧结束点#2



USB 主机控制器应该负责不要设备回应，如果该回应会导致设备在 EOF2 时刻发出包。这时主机应该发出异常终止序列来保证设备不作回应。而且，因为集线器将在到达 EOF1 时刻时结束上行方向的包，如果来自设备的响应(数据或握手信号)未定或者集线器到 EOF1 时刻的过程中，主机不应开始一个事务。

内部接口(Internal port)连接着集线器转发中继器和集线器控制器。除了传送串行数据到集线器控制器或从集线器控制器接收串行数据外，内部端口还是一定唤醒信号的来源。图 10-2 给出了内部端口状态自动机。表 10-2 定义了内部端口的信号和事件。

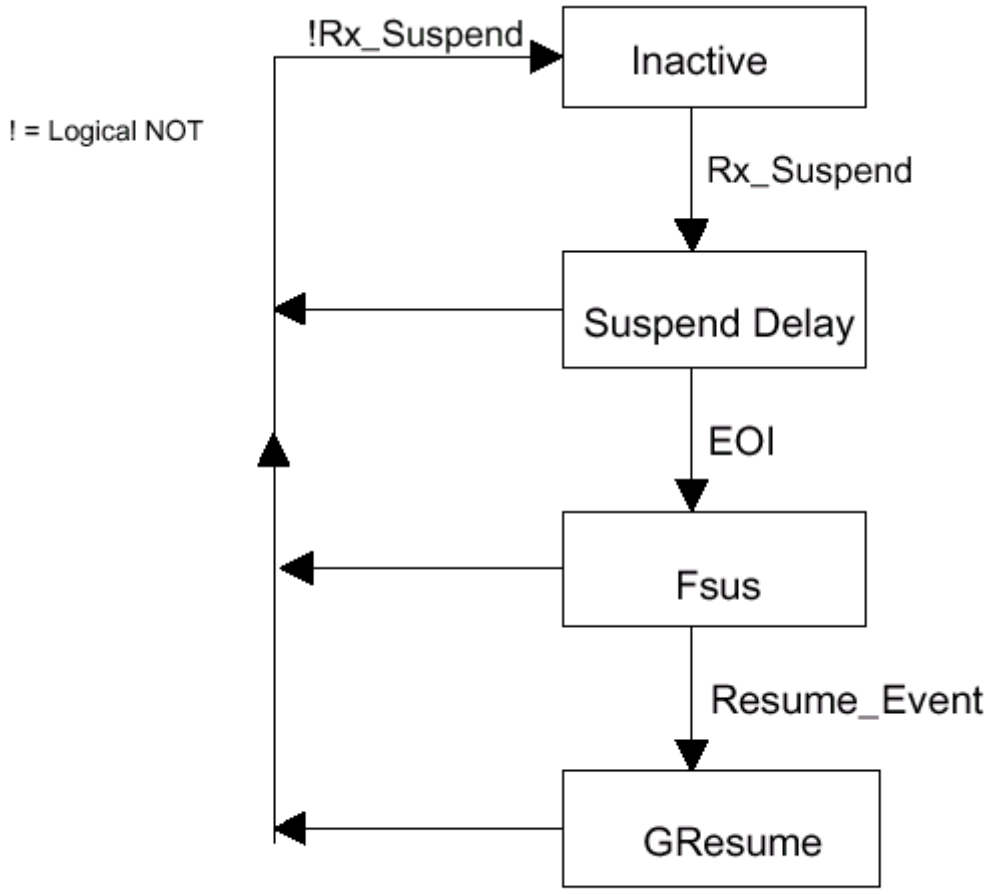


表 11-2 内部端口信号/事件定义

信号/事件名称	事件信号源	描述
EOI	内部	时间间隔结束
Rx_Suspend	接收器	接收器在挂起状态
Resume_Event	集线器控制器	集线器控制器中出现唤醒信号

只要接收器不处在挂起状态，内部端口就处在不活动状态。当接收器进入挂起状态时，内部端口就进入挂起延迟状态。该状态有 2ms 的时间限制。挂起时延状态过期后就进入充分挂起状态，在充分挂起状态时，如果集线器控制器中产生唤醒条件，就进入产生唤醒(Generate Resume)状态。在该状态时，内部端口产生追加 SOP_FD 到集线器转发中继器。

图 10-3 给出集线器下行端口的简化状态自动机。事件和信号在表 10-3 中定义。

表 11-3 下行集线器端口信号/事件定义

信号/事件名称	事件信号源	描述
Power_sorce_off	依赖于实现	端口电源关闭是由于电流过载或退出源电流（例如：外部电源撤走）
Over_current	集线器控制器	集线器或端口上出现电流过载条件
EOI	内部	时间间隔或序列结束
SE0	内部	在端口上接收到 SE0
Disconnect_Detect	内部	在端口上检测到长 SE0（参见 11.5.2）
LS	集线器控制器	低速设备接到该端口上
SOF	集线器控制器	接收到 SOF 令牌
J	内部	在端口上接收到‘J’
K	内部	在端口上接收到‘K’
Rx_Resume	接收器	上行接收器在唤醒状态
Rx_Suspend	接收器	上行接收器在挂起状态
Rptr_Exit_WFEOPFU	集线器转发中继器	集线器转发中继器离开 WFEOFFU 状态
Rptr_Enter_WFEOPFU	集线器转发中继器	集线器转发中继器进入 WFEOFFU 状态
Port_Error	内部	检测到错误条件（参见 11.8.1 节）
Configuration=0	集线器控制器	集线器控制器配置值为 0

只要集线器配置的值为零下行端口就进入未配置(Not Configured)状态，并保持该状态不变。集线器将在端口上驱动 SE0。不会有其它的活动信号发生。所有集线器都支持关掉电源(Power-off 状态)。对端口的有非零配置值的 SetConfiguration()请求将使端口从任何状态进入未供电状态。在除未配置状态外任何状态，并接收到 ClearPortFeature(PORT_POWER)时或检测到电流过载条仲时也将进入该状态。在该状态时差分单端发送器和接收器未使能。在未供电状态时接收到 SetPortFeature (PORT_POWER)请求或端口的未连接计时器超时，或 Restart_S、Restart_E 状态过期后就进入未连接状态(discorneeted)。在该状态时仅有连接检测是可能的。表明这时还没有设备接入该端口。该状态有时间限制。在该状态时，只要求接收到 SE0 信号时计时器就复位，在检测到其它信号时才开始计时。除集线器挂起时钟停止外，该计时器的时限是 2.5us 到 2ms。如果集线器挂起并有远程唤醒功能，在一个未连接端口上从 SE0 状态变化将使集线器启动时钟并对该事件计时。集线器需能在状态变化后的 12ms 内起动时钟并对该状态计时，如果挂起的集线器没有远程唤醒功能，集线器将忽略该事件直到集线器被唤醒。在未连接状态过期或对端口有相应请求或检测到错误条件时进入未使能状态。在该状态时，对接收到的 SE0 信号都要计时。除未供电状态和未连接状态外，对端口发出重启请求时就进入重启状态。此时，集线器在端口上驱动 SE0 信号。该状态的时限通常 10ms 至 20ms。在重启结束，或 SendEOP 状态结束，或在发送状态转发中继器离开 WFEOPFU 状态，或从挂起状态当挂起的上行接收器检测到‘K’时，进入使能(Enable)状态。在该状态时，

从‘J’到‘K’的状态变化就能建立上行连接。在使能状态时，如果上行接收器处在唤醒状态，或者在 Restart-S、Restart-E 状态，一检测到‘K’就进入发送状态。对于高速设备，是在端口处在使能状态且转发中继器进入 WFEOPFU 状态时进入发送状态。在该状态时，端口将传送在上行端口上接收的数据。对低速设备来说，是在上行端口上接收到一个高速 PRE PID 时从使能状态转入的。在该状态时，端口将对上行端口接收到的数据作适当变换后重新发出。端口在使能状态时接收到相应的请求时变为挂起(Suspend)状态。在该状态时，端口的差分发送器被设为未使能。端口在接收到相应的请求，或在接收器未挂起时检测到‘K’就进入唤醒状态(Resuming)。该状态名义上有 20ms 的时限。在该状态时，集线器在端口上驱动‘K’。重启状态结束后就进入 SendEOP 状态。在使能状态时接收到 SOF 且有低速设备接在该端口上也可进入该状态。该状态时，集线器将发送低速的 EOP。在 EOP 结束时该状态也结束。端口处在使能状态，且接收器处在挂起状态时，在检测到 SE0 或‘K’时就从挂起状态进入 Restart_S 状态，或从使能状态进入 Restart_E 状态。在这些状态时端口继续监视总线状态，当看见‘K’时就立刻进入发送状态，否则就进入未连接状态。

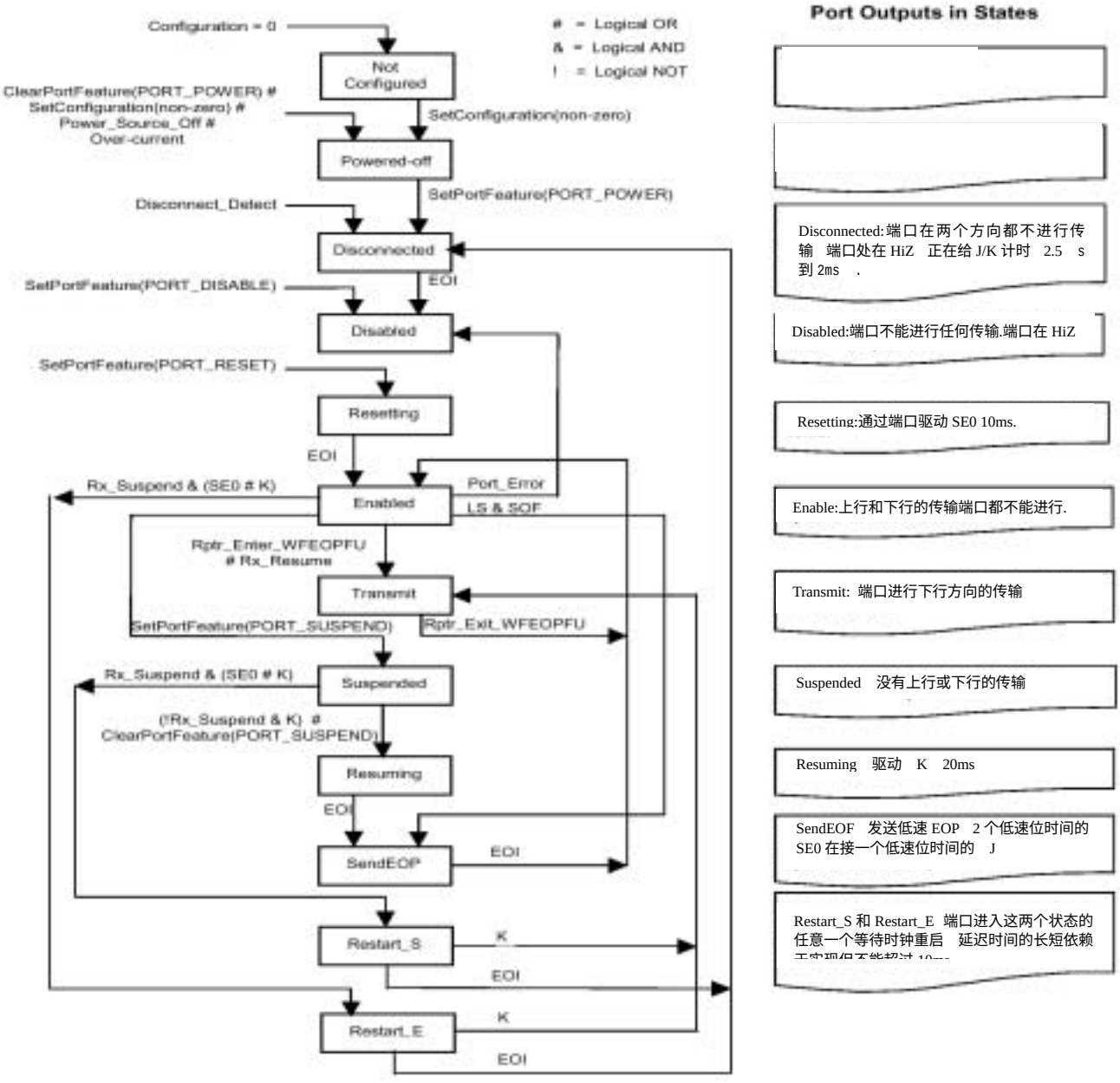


图 11-6. 下行集线器端口状态变迁图

每个端口需要一个未连接计时器。该计时器用来连续监视端口的单端接收器，来检测未连接事件。

表 11-4 上行集线器端口接收器信号/事件定义

信号/事件名称	事件信号源	描述
Tx_active	发送器	发送器在活动状态
J	内部	在上行端口上接收到一个‘J’（空闲）
EOI	内部	时间间隔结束
K	内部	在上行端口上接收到一个‘K’
Tx_resme	发送器	发送器在 Sresume 状态
SE0	内部	在上行端口上接收到一个 SE0
POR	以赖于实现	Power_On_Reset

图 10-4 给出了集线器上行接收器的状态自动机。表 10-4 定义了事件和信号。在发送器活跃时或接收器检测到 SJ 条件时从除挂起状态外的任何状态进入 ReceivingJ 状态。该状态有 3ms 的时间限制。每当进入该状态时计时器复位，且只有在发送器处在不活动状态时该计时器才计时。该状态过期后就进入挂起状态。进入该状态后控制器开始了一个 2ms 的计时器。如果计时器过期且接收器仍在该状态则控制器挂起。控制器挂起后，它可产生唤醒信号。当在总线上检测到 SK 条件且转发中继器处在 WFSOP 或 WFSOPEU 状态时，接收器从除唤醒状态外的任何状态进入 ReceivingK 状态。该状态有 2.5us 到 100us 的期限。当 ReceivingK 状态过期后就进入唤醒状态。当发送器处在 Sresume 状态，或在上行端口上检测到‘K’状态时，从挂起状态也可进入该状态。接收器检测到 SE0 条件，且转发中继器处在 WFSOP 或 SOPFU 状态时，从除总线重启状态外的任何状态进入 ReceivingSE0 状态。该状态有时间限制，最短 2.5ms，最长取决于集线器。当 RecevingSE0 状态过期后就进入总线重启状态(Bus-Reset)。只要端口上连续接收到 SE0 就保持该状态不变，该状态也在集线器本地电路产生 POR 时进入。在 POR 活跃时不能离开该状态。

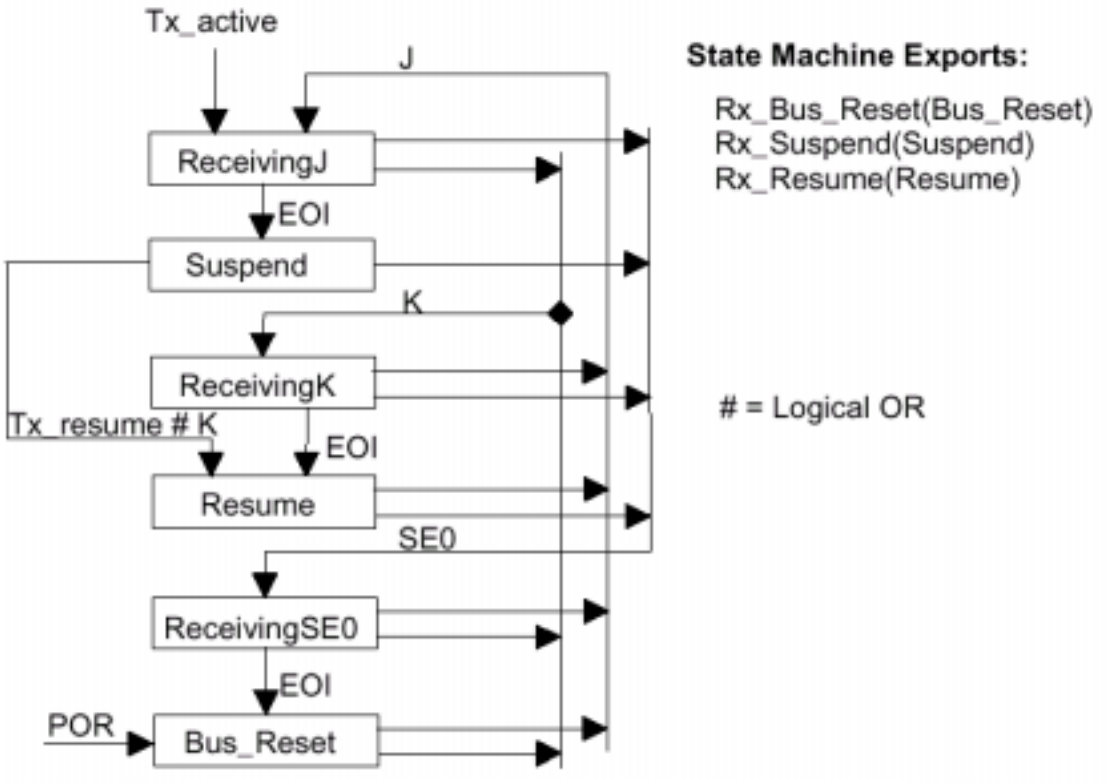
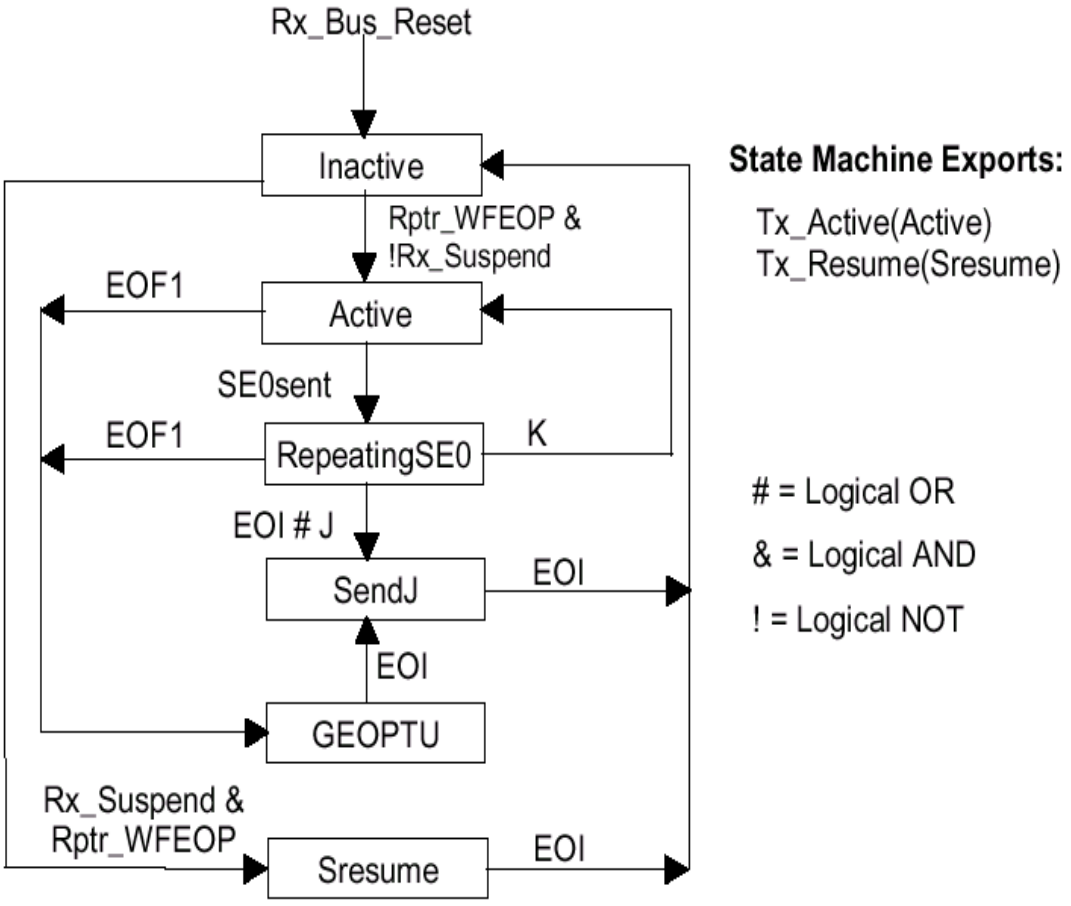


表 11-5 上行集线器端口发送器信号/事件定义

信号/事件名称	事件信号源	描述
Rx_Bus_Reset	接收器	接收器在总线重启状态
EOF1	帧计时器	集线器帧计时器到达 EOF1 时刻或在 EOF1 和帧结束之间

图 10-5 是发送器的状态变化图。表 10-5 定义了图 10-5 中出现的事件和信号。发送器用来在转发中继器有上行连接时监视上行端口。其目的是防止错误指示被传到上行方向。在接收器处在总线重启状态，或 SendJ 状态结束时，发送器进入不活动(Inactlve)状态。在 Sresume 状态结束时也进入该状态。这时差分和单端接收器都未使能，并置为高阻状态。转发中继器进入 WFEOP 状态时，发送器从不活动状态进入活动状态(Active)。如果在转发 SE0 状态时，在 SE0 后的不是 J 状态则进入活动状态。在该状态时，来自下行端口的数据在上行端口上被转发。发送器在上行端口上发送一个位时间的 SE0 时就从活动状态进入转发 SE0 状态(RepeatingSEO)。这时发送器仍然转发下行端口上的信号。该状态的时限是 23 个高速位时间。在转发 SE0 状态时，如果到达位时间 23，或转发信号从 SE0 变为‘J’就进入发送 J 状态(Send J)。在 GEOPTU 状态结束时也进入该状态。该状态持续一个高速位时间，在此期间集线器在端口上驱动 J。帧计时器到 EOF1 时刻时发送器从转发 SE0 或活动状态进入 GEOPTU 状态。GEOPTU 的含义是向上行端口发出 EOP。在该状态时，端口发送 2 个高速位时间的 SE0。如果集线器转发中继器进入 WFEOP 状态且接收器在挂起状态，发送器就从不活动状态进入发送唤醒状态(Send resume，简记为 Sresume)。这意味着下行设备(或端口对控制器)产生了唤醒信号，导致建立上行连接。端口在该状态至少停留 1ms，但不超过 15ms。

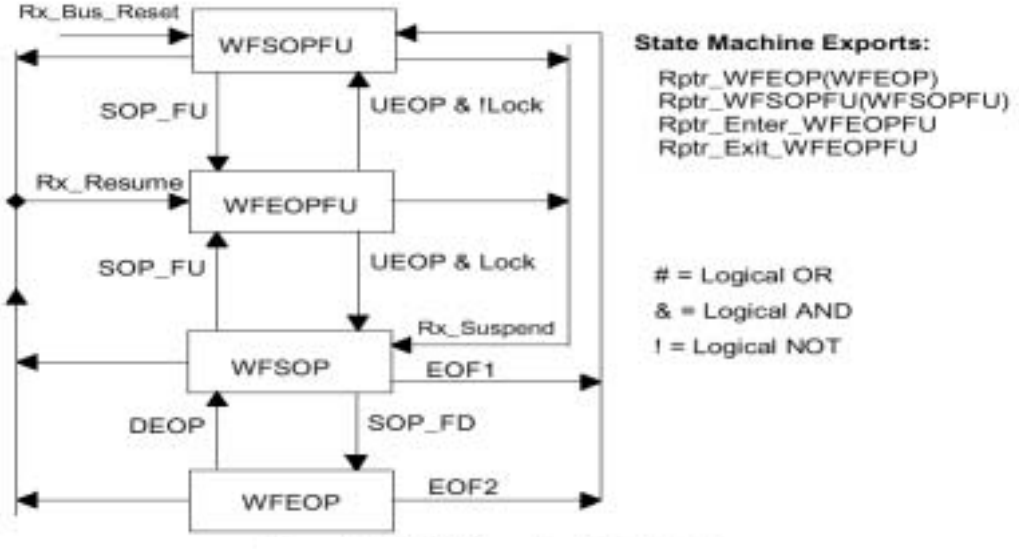


集线器转发中继器提供的功能有，在包的边界建立连接和确保有序的进入和离开挂起状

态，包括远程唤醒的处理。图 10-6 是转发中继器的状态自动机。表 10-6 是相应的信号和事件。一些状态变化是在检测到 EOP 时发生的，但并不是一检测到 EOP 就发生，要到集线器转发 SE0 到‘J’的转变且驱动‘J’至少一个位时间之后发生。一些是由 SOP 触发的。在上行接收器在总线重启状态，或在 WFSOP 状态时帧计时器到了或过了 EOF1 时刻，或在 WFEOP 状态时到达 EOF2 时刻，或在 WFEOPFU 状态且帧计时器未同步时接收到 EOP，就进入等待来自上行端口的 SOP 状态(WFSOPFU)。在该状态时，集线器等待上行接口的 SOP 并忽略下行端口的状态变化。这时并未建立连接。该状态用来在帧结束(过 EOF1 时刻)时确保集线器能接收到来自主机的 SOF。如果集线器处在 WFSOP 或 WFSOPFU 状态并检测到 SOP 时就进入等待来自上行端口的 EOP 状态(WFEOPFU)。当接收器进入唤醒状态时集线器也从其它状态进入该状态。在该状态时下行连接建立了，处在使能状态的下行端口在集线器进入该状态时被置为发送状态。集线器在 WFEOP 状态并检测到 EOP，或在 WFEOPFU 状态且帧计时器已同步时接收到 EOP，或在 WFSOPFU、WFEOPFU 状态时上行接收器进入挂起状态，就进入等待 SOP 状态(WFSOP)。该状态时等待来自上行端口，或使能下行端口的 SOP。这时连接并未建立。在 WFSOP 时检测到来自使能下行端口的 SOP 时就进入等待 EOP 状态(WFEOP)。该状态时建立了上行连接，上行发送器在集线器进入该状态时就进入活动状态。在该状态时到达 EOF2 时刻，建立连接的下行端口被设为未使能。

表 11-6. 集线器转发中继器信号/事件定义

信号/事件名称	事件信号源	描述
Rx_Bus_Reset	接收器	接收器在总线重启状态
UEOP	内部	在上行端口上接收到 EOP
DEOP	内部	在发送器进入 SendJ 状态是产生
EOF1	帧计时器	帧计时器到达 EOF1 时刻或在 EOF1 和帧结束之间
EOF2	帧计时器	帧计时器到达 EOF2 时刻或在 EOF2 和帧结束之间
Lock	帧计时器	帧计时器被琐定
Rx_Suspend	接收器	接收器在挂起状态
Rx_Resume	接收器	接收器在唤醒状态
SOP_FD	内部	从下行端口或集线器控制器接收到 SOP。在端口从空闲态转到 K 状态是产生
SOP_FU	内部	从上行端口接收到 SOP。在上行端口从空闲态转到 K 状态是产生



集线器需要评估端口上的连接状态来作出正确的状态转换。

当集线器转发中继器处在 WFEOP 状态而帧计时器到达 EOF2 时刻；或在 EOF2 时刻集线器处在 WFSOPFU 状态，但端口上并没有 J 状态时出现端口错。接入设备的速度是由设备上拉电阻的位置决定的。当接入一个设备时，集线器期望通过感知总线空闲状态来检测速度。速度检测可在端口离开未使能状态进入重启状态时进行，也可在重启结束时即在重启状态结束和使能状态开始之间进行。当集线器转发中继器处在 WFEOP 状态时在其它使能端口上检测到 SOP，这时就产生冲突条件。集线器有两种相应的处理，第一种，也是首选的一种，是“窜改”信息以使主机能检测问题。集线器通过在上行端口发送‘K’来窜改信息。这个‘K’持续到所有下行端口的传输结束，集线器用最后一个 EOP 来结束窜改的包。另一种是阻塞第二个包，在第一个结束时使集线器适当的返回到 WFSOPFU 或 WFSOP。这种方式不向主机报告问题。集线器上行连接总是高速的，而下行连接则要支持高速和低速设备。高速设备和低速设备表现出不同的行为。在总线上没有低速的传输时低速设备就会被挂起，为防止低速设备被挂起。必须在每帧中接收到 SOF 时产生一个打入信号。

集线器作为一个 USB 设备或为了传输挂起和唤醒信号都要求它支持挂起和唤醒。集线器支持全局挂起和局部挂起或唤醒。全局挂起或唤醒是指整个总线被挂起或唤醒而不影响集线器下行端口的状态；局部挂起和唤醒是指集线器下行端口被挂起或唤醒而不影响集线器状态。全局挂起或唤醒是通过主机的根端口来实现的。局部挂起和唤醒是通过对集线器发出请求来实现的。远程唤醒是指由设备发出的唤醒。

集线器的重启信号仅定义在下行方向即在上行端口上。集线器在检测到 2.5ms 或更长时间的连续 SE0 信号时开始重启，并必须在该信号结束时完成重启序列。挂起的集线器必须将重启信号解释为唤醒事件，它必须在重启信号结束之前醒过来并完成重启序列。重启结束后集线器处在下面的状态：

- 控制器缺少地位为 0；
- 集线器状态变化位都为 0；
- 转发中继器在 WFSOPFU 状态；
- 发送器在不活动状态；
- 下行端口在未配置状态并在所有下行端口上驱动 SE0 信号。

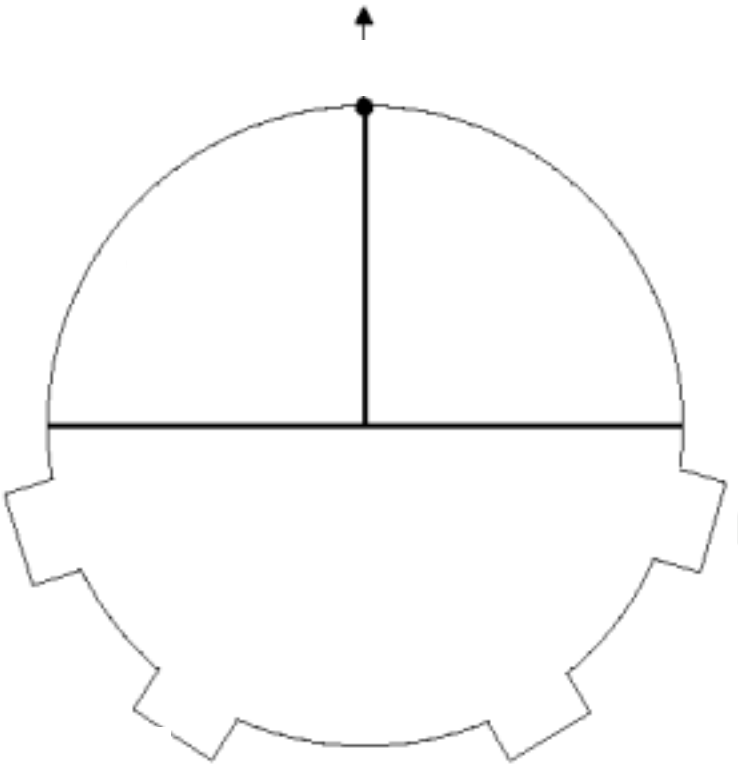
由总线供电的集线器要求有电源开关。自供电的集线器可以有电源开关但并不是必需的。有电源开关的集线器可以控制每组端口的电源，或每个端口的电源或每组有任意个端口的电源。集线器通过设置 wHubCharateristic 中的逻辑电源开关模式域来指示是否支持电源开关。集线器电流过载保护也类似，可以分组：电流过载保护的分组和电流开关的分组是独立的。

集线器的所有端口必须都能检测和生成所有的总线信号状态。这就要求集线器能在它的每个端口上驱动和监视 D+和 D-的输出。每个集线器端口必须有单端的接收器和发送器。集线器和接在它上的设备使用上拉电阻和下拉电阻的组合在 D+和 D-未被积极驱动时来控制 D+和 D-。每个集线器下行端口在每根数据线上需要一个下拉电阻(Rpd)，上行端口则需要上拉电阻(Rpu)。集线器下行端口必须支持低速和高速信号斜率（Edge rate）的传送和接收。下行端口上的信号斜率必须是可选择的，要看接入端口的设备是高速的还是低速的。上行端口总是使用高速的信号。

集线器逻辑结构如图 10-7 所示。

集线器类(Hub Class)定义了除缺省控制管道外的附加的所有设备都需要的端点(endport)：状态变化端点。主机系统通过该端点接收集线器状态变化通知。它是一个中断的端点。如果没有状态变化位被设置时，集线器在被轮询时返回 NAK。如果状态变化位被设置时就返回数据。USB 系统软件能用该数据来决定读哪些寄存器能判断状态变化的确切原因。图 10-8 给出了状态，状态变化和控制信息是怎样与设备发生连系的。集线器描述子和集线器/端口状态和

控制是通过缺省控制管道读取。集线器描述子随时可读取。当集线器在端口上检测到变化或它自身状态变化时，状态变化端点就以指定的形式向主机发出数据。集线器状态变化位能由于硬件或软件事件而设置。设置后一直保持直到被 USB 系统软件清除。USB 系统软件用与状态变化位相连的中断管道来检测集线器或端口的状态变化。USB 系统软件通过消除集线器报告的相应位来确认端口的变化。USB 设备必须被设置为符合一定的安全标准。通常这意味着自供电和集线器在下行端口上实现电流限制。当出现电流过载条件时，它导致一个或多个端口的状态变化。这种变化被报告给 USB 系统软件以采取正确的行动。集线器是通过标准的 USB 设备配置命令配置的。USB 系统软件检查集线器描述子信息来决定集线器的特征。



集线器描述子是源自 USB 设备框架。集线器描述子定义了集线器上的集线器设备和端口。主机通过集线器缺省管道来读取集线器描述子。USB 规范定义了下面的描述子：

- Device；
- Configuration；
- Interface；
- Endpoint；
- String(可选的)

集线器对请求处理过程的时间有比标准设备更严格的要求。下面列出最坏情况下的请求时间。

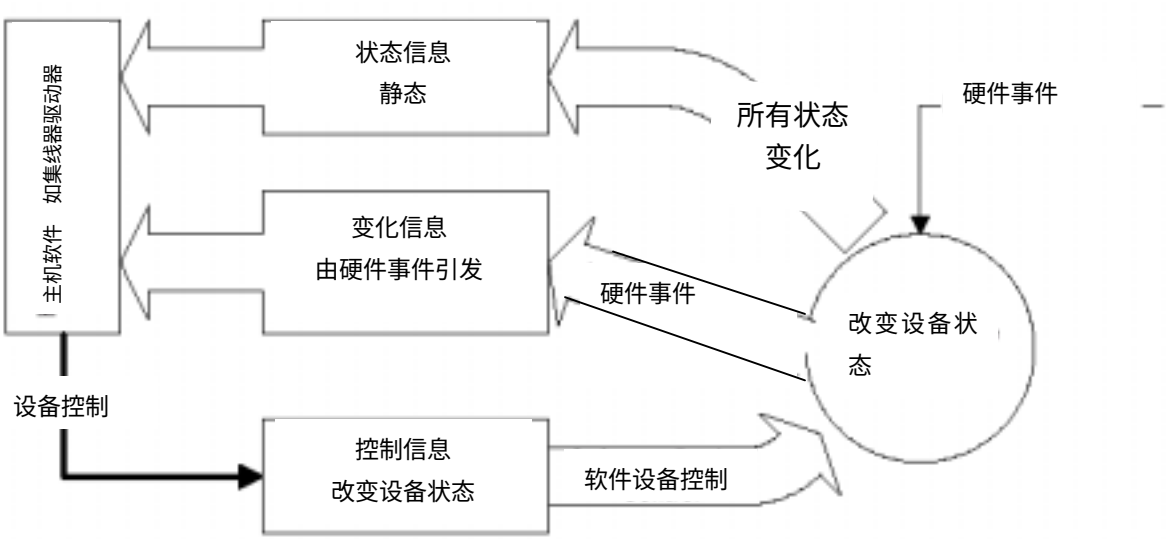
1 没有数据阶段的完成时间：50ms

2 有数据阶段的标准请求的完成时间

从装配包到第一个数据阶段的时间：50ms

每两个相继数据阶段间的时间：50ms

最后数据阶段和状态阶段之间的时间：50ms


主机软件 如集线器驱动器
状态信息
静态
所有状态
变化
硬件事件
变化信息
由硬件事件引发
硬件事件
改变设备状
态
设备控制
控制信息
改变设备状态
软件设备控制