2　系统总体架构原理分析

USB 总线作为一种广泛使用的 PC 机外围接口，有着自己特有的模型结构。对接入该总线的实际物理设备，有着特殊的配置方式和信号响应流程，并且在数据通信模式上也与其他接口总线大相径庭。本章将结合步进电机控制系统的控制通信模型，对 USB 的系统整体架构原理，进行分析和介绍。

2. 1　USB 接口的步进电机控制装置整体架构

本步进电机控制装置的整体架构如图 2-1 所示。其组成部分及功能如下：

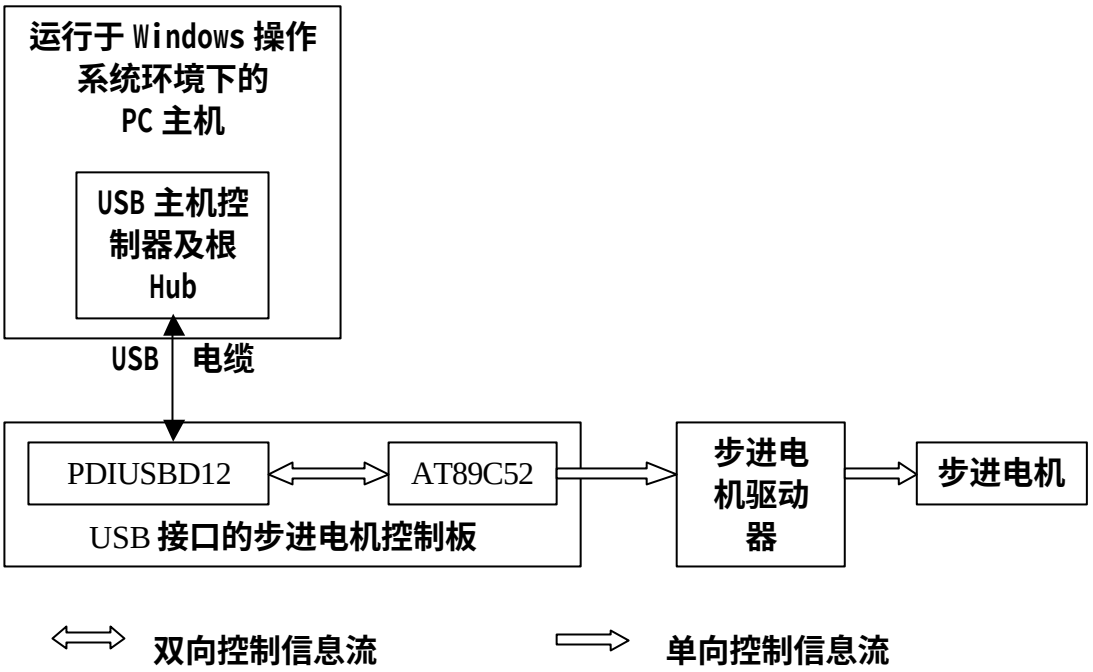

图 2-1　装置整体架构图

2. 1. 1　PC 主机

在整个步进电机控制装置中，运行于 Windows 操作系统环境下的 PC 主机是整个控制装置的命令发布和运动参数设定的中心。当运行于该主机中的应用程序,以本装置的驱动程序的 GUID 号为参数，调用 Windows API 函数 CreatFile()打开事先安装到 Windows 操作系统中的、本步进电机控制板的设备驱动程序(USB_Motor.SYS)，并得到

该函数对应返回的设备句柄。之后便可以，以该设备句柄和定义在表 5-2 中第一列的各个命令为参数，调用 Windows API 函数 DeviceIoControl()，与本步进电机控制板的设备驱动程序(USB_Motor.SYS)进行控制通信了。对步进电机控制和运动参数设置的具体命令及其含义见表 5-2。在驱动程序中与这些命令对应的处理例程见表 5-3 和表 5-4。

当驱动程序接收到应用程序下达的步进电机控制命令后，会将这些命令数据、以主机中 USB 子系统模块所规定的方式、发送到 USB 主控器，主控器进而将这些数据发送到 USB 总线，以供 USB 接口的步进电机控制板接收。这个步骤的详细过程将在本章的 2.3 节中介绍。关于主机与设备之间的通信数据、如何在 USB 总线上进行传输的细节，将在本章 2.4 节中介绍。

2. 1. 2 USB 接口的步进电机控制板

当主机中的 USB 主控器，通过 USB 电缆，将数据发送到数据总线时，步进电机控制板上的 USB 接口芯片 PDIUSBD12 会自动监测该数据是否发送给自己,并对发送给自己的数据进行接收。当一个传输接收完毕，会以中断的方式提示步进电机控制板的主控芯片(AT89C52)。步进电机控制板上的 PDIUSBD12 与 AT89C52 接口的电路设计，将在第三章中具体介绍。

AT89C52 中运行的固件代码的作用主要有两部分：第一，对 USB 传输控制协议所规定通信流程的进行响应，并对接收到的 USB 总线数据进行处理来设置相应的步进电机控制的全局变量；第二，依照步进电机控制的全局变量，并利用定时器中断对时间计时，从而对步进电机进行驱动控制。固件程序中对主机发送来的控制命令的具体处理函数见表 4-1，需要注意的是表 4-1 与表 5-3、表 5-4 以及表 5-2 中的项是一一对应的。固件程序编写的具体细节将在第四章中介绍。

2. 1. 3 步进电机驱动器

步进电机控制板通过 Driver1,Direct1 和 Driver2, Direct2 两组步进电机驱动口，来分别为 X 和 Y 两个方向上的步进电机驱动器、提供步进电机驱动的脉冲和方向控制信号。其中 Driver 为脉冲输出口；Direct 为方向控制口。

本装置选用的驱动器为：XCW 220

其特性如下：

- 适用电压范围宽(20～40VDC) 。
- 采用高频恒流斩波,脉宽调制式驱动。

- 采用细分克服低频抖动，提高运行频率和运行力矩 。
- 运行特性良好 。
- 自动半流锁定。
- 可靠性高 。
- 适配 2A 以下两相、四相混合式步进电机。
- 驱动模式：全桥 PWM 微步驱动。
- 运行方式：半步、1/8 步。
- 相电流：本模块相电流可在 0～1.5A 之间调节。

2. 1. 4 步进电机

步进电机可以分为三大类：

（1） 反应式步进电机（Variable Reluctance，简称 VR）

反应式步进电机的转子是由软磁材料制成的，转子中没有绕组。它的结构简单，成本低，步距角可以做的很小，但是动态性能较差。

（2） 永磁式步进电机（Permanent Magnet，简称 PM）

永磁式步进电机的转子是永磁材料制成的，转子本身就是一个磁源。它的输出转矩大，动态性能好。转子的极数与定子的极数相同，所以步距角一般较大。需要传给正负向脉冲信号。

（3） 混合式步进电机（Hybrid，简称 HB）

混合式步进电机综合了反应式步进电机和永磁式步进电机两者的优点，它的输出转矩大，动态性能好，步距角小，但结构复杂，成本高。

本装置采用两相混合式步进电机 XCW 220，对应选取的驱动器为 35HS34DF09。

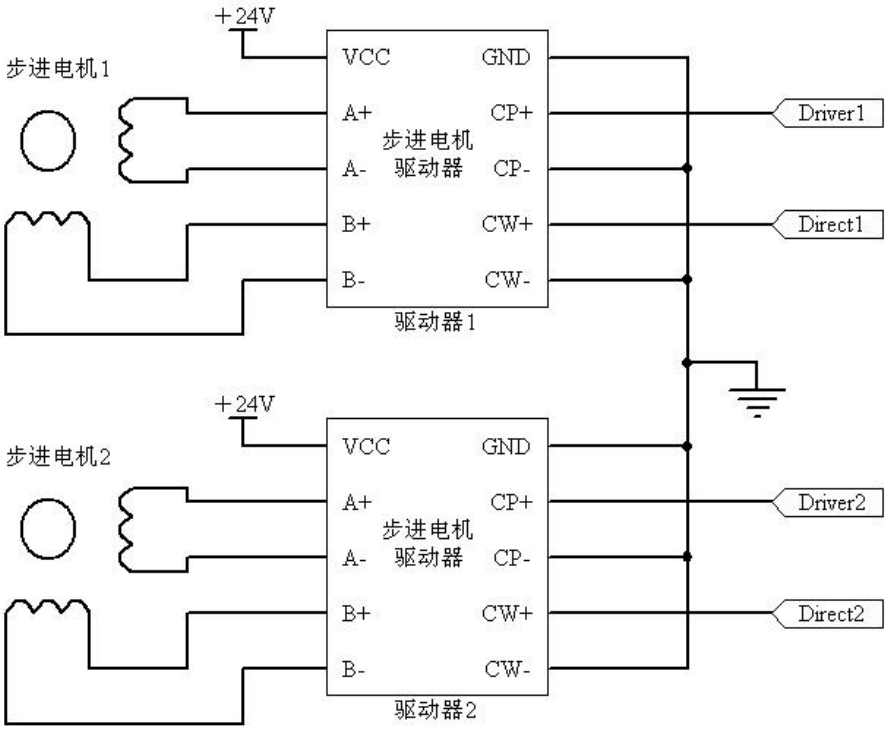

图 2-2 驱动器与步进电机连接示意图

步进电机的各相连接线，对应的接入步进电机驱动器的端口中，以供其步进驱动。由步进电机控制板输出的两组 Driver 和 Direct 信号，对应连接到步进电机驱动器上。具体连接如图 2-2 所示。

本文的目地是，研究如何利用 Windows 平台下的 USB 接口来进行步进电机的驱动控制。开发的重点和难点均集中在，如何依照 USB 体系架构的通信模型，对 USB 接口进行开发和利用，从而达到控制步进电机运动的目地。而步进电机以及相应的驱动器，在本装置中都是选用市场上已经开发好的产品，因此文中将不再进一步介绍步进电机及其驱动器这部分。

2. 2 USB 系统通信架构整体模型

2. 2. 1 USB 系统物理层次

图 2-3 展示了本系统的 USB 配置物理拓扑结构。其中的系统总线是指 PC 主机中的 PCI 总线，USB 主控制器与其它 I/O 设备一样直接连接到该系统总线上。操作系统

与主控制器通信使用 I/O 口或内存寄存器，通过普通的中断信号，操作系统可以接受主控制器的事件通知。主控制器连接一棵 USB 设备树。PC 机中的 USB 根集线器 Hub 作为其它设备的起始连接点。多个 Hub 能以菊链方式连接，可以连接到 USB 规范中定义的最大深度。本系统的步进电机控制板，作为一个 USB 设备，直接连到根 Hub 上。为了精确地表达概念，USB 规范使用术语 Function 来描述非 Hub 的设备。

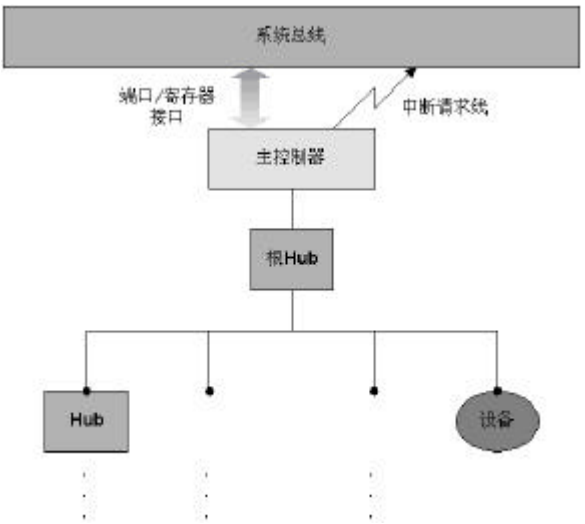

图 2-3 USB 配置物理拓扑结构

2. 2. 2 USB 系统逻辑层次

图 2-4 从通信逻辑的方面，展示了本系统中主机与设备的多层次结构的通信模型，它表明了端点和管道所扮演的角色。在最低一级，USB 电缆把主机中的 USB 主控器与步进电机控制板上的总线接口连接起来。图中的 SIE 模块是 USB 接口芯片上的协议处理模块，负责对 USB 协议的响应和总线上传输数据的收发转换。在第二级，一个控制管道把系统软件与逻辑设备连接起来。在第三级，一组数据管道把应用程序与接口的一组端点连接起来，接口的这些端点组成设备的具体功能。信息实际上是在图中两侧垂直流动，但在此，把它理解为在这些分层的管道中水平流动更清晰。该图作为系统通信原理总图，对其各个模块的具体分析将在本文中陆续展开。

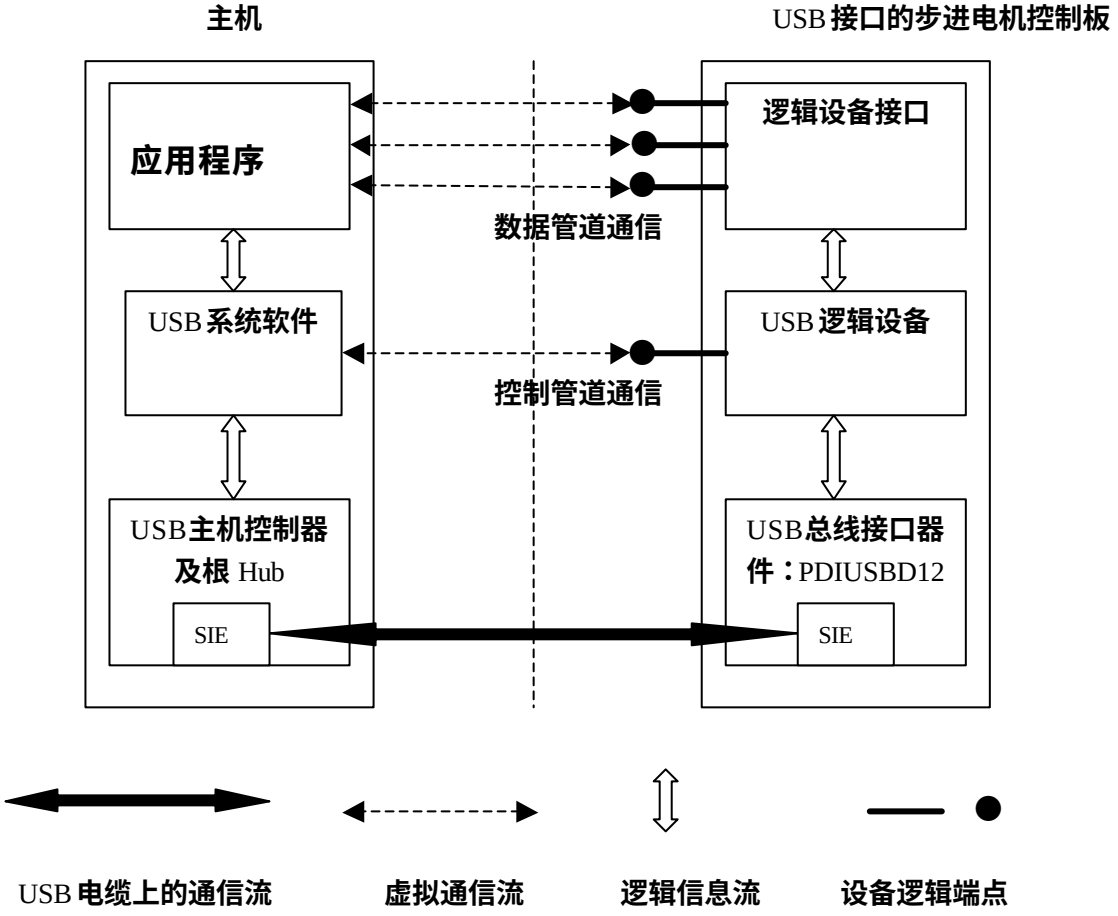

图 2-4　本系统的多层次通信模型

2. 2. 3　逻辑设备内部配置结构

每个 USB 设备有一个或多个配置(configuration)来控制其行为，如图 2-5。

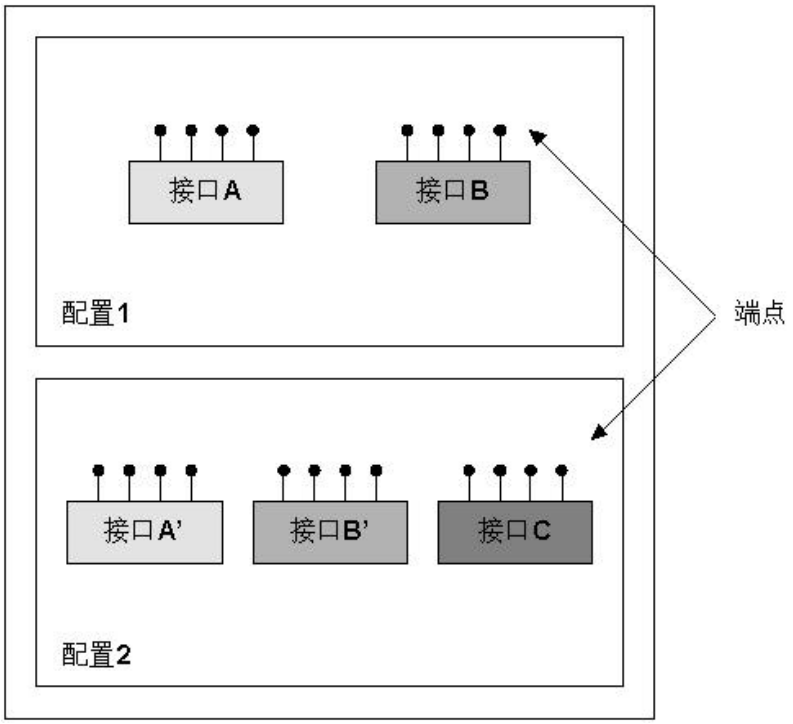

图 2-5 设备逻辑配置结构

设备的每个配置中都含有一个或更多的接口(interface),接口指出软件应该怎样访问硬件,是一组端点的集合。支持相同接口的设备本质上可以互换，因为它们以相同的方式响应相同的命令。另外，接口一般都有替换设置以适应不同的带宽需求。

2. 2. 4 设备端点

USB 的设备端点分为四类：控制端点、批量端点、中断端点、等时端点。

一个端点是一个可唯一识别的 USB 设备的一部分，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。每个端点都有两个简单的逻辑连接点，一个支持数据流进设备，一个支持数据流出设备。

一个端点的特性决定了它与应用软件进行的数据传送的类型。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(控制、批量、中断、等时)
· 端点与主机的数据传送方向（IN 、OUT）

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。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 的输入通道作为输入逻辑端点，同时将端点 0 的输出通道作为输出逻辑端点。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)。缺省控制通道支持了对控制的传送，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。

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。

PDIUSBD12 的设备端点配置将在第三章中具体介绍。

2.3 主机 USB 系统驱动调用层次

在图 2-4 所示的系统通信流模型中,本系统的 USB 设备驱动程序(USB_Motor.SYS)占据了图中“ USB 系统软件 ”方块中的顶部。一组由 Windows 操作系统提供的驱动程序占据了该方块中的底部。这些驱动程序包括主控制器驱动程序(OPENHCI.SYS 或者 UHCD.SYS)，Hub 驱动程序(USBHUB.SYS)，和一个类驱动程序(USBD.SYS)，由控制器驱动程序使用。

如果将 USBD 下面的所有驱动程序看成一个整体，本系统的 USB 设备驱动程序(USB_Motor.SYS)主要就是与这个整体进行交互，这个整体管理着硬件连接和管道通信。USB 设备的 WDM 驱动程序，也就是本系统的功能驱动程序(FDO)，占据系统软件方块中的顶部，它的工作就是把应用软件的请求翻译成 USBD 能执行的事务(transaction)。该驱动程序就是图 2-6 中的 USB_Motro.SYS，对它所执行的具体功能和编写细节将在第 5 章中详细介绍。

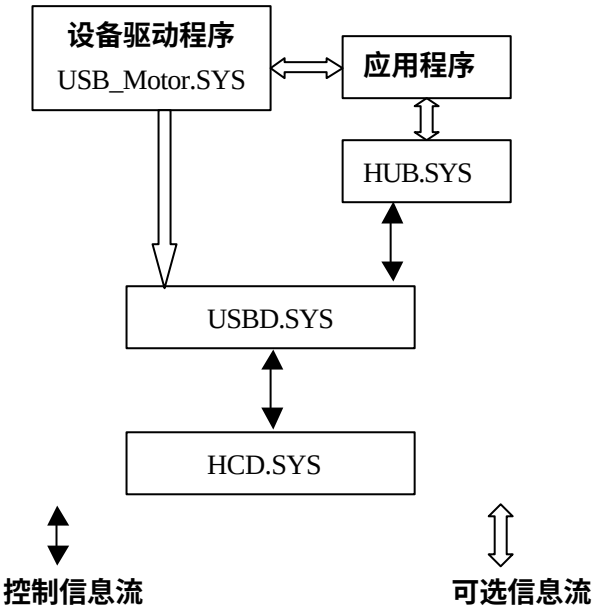

图 2-6　主机驱动层次调用图

应用程序处理实际的设备功能，如图 2-6 主机驱动层次调用图所示。

当应用程序通过 USB 管道发送或接收数据时，它首先调用 Windows API，调用进入 Windows 系统空间后，最终将使我们编写的 USB 设备驱动程序（USB_Motro.SYS）收到一个 IRP（中断请求包）。而 USB_Motro.SYS 的工作就是把客户的请求引导到有正确端点的管道上。它将把请求提交到总线驱动程序（即为 HUB.SYS、UHCD.SYS、USBD.SYS 之总集），总线驱动程序再把请求分解成多个事务(transaction)，然后这些事务被送往总线。该流程如图 2-7 所示。该图左边部分表示的是主机中的 USB 系统数据传输的层次，右边对应的是在各个层次中的数据封装模式。

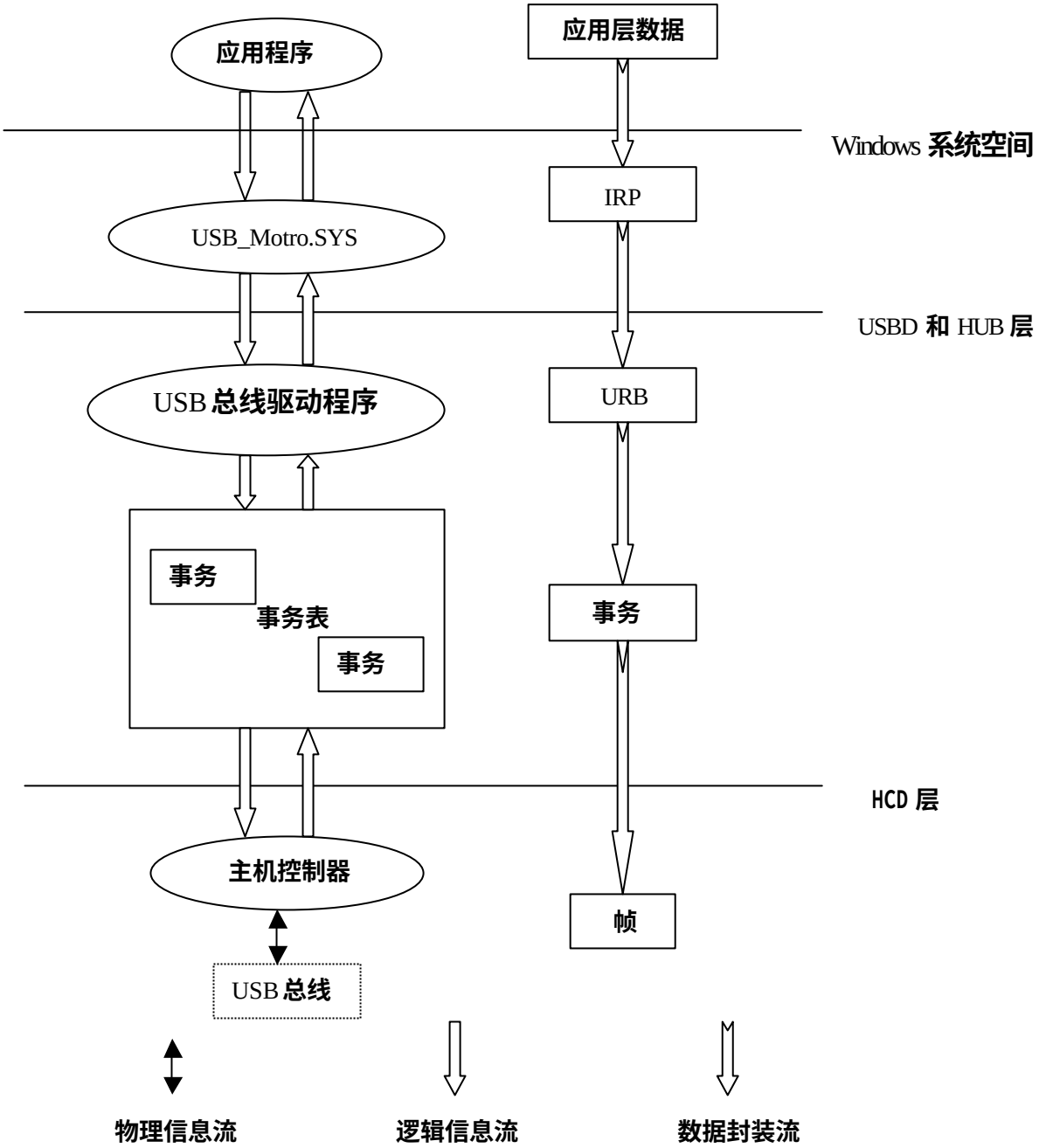

图 2-7 主机数据传输流图

2. 4 USB 总线数据传输

USB 规范对 USB 主控器和设备之间,定义了四种数据传输方式,如表 2-1 所示。它们的不同之处有：单个事务能携带的数据量、能否保证特定的周期或延迟，能否自

动校正错误。每种传输方式对应特定类型的端点。实际上，给定类型的端点(控制端点、批量端点、中断端点、等时端点)总是使用对应类型的传输。

表 2-1. 数据传输类型

传输类型	描述	纠错	包容量(字节)	延迟保证？
控制	用于发送和接收 USB 定义的结构化信息	是	少于或等于 8,16,32,64	尽最大能力保证不延迟
批量	用于发送或接收小块无结构数据	是	少于或等于 8,16,32,64	无
中断	与批量管道相似，但包括一个最大延迟	是	少于或等于 64	以保证的最小速率轮询
等时	用于发送或接收有周期保证的大块无结构数据	否	少于或等于 1023	每 1 毫秒帧中的固定部分

端点除了传输类型外还有其它几个属性。其中一个属性是单一事务中端点能够提供或消耗的最大数据量。控制和批量端点必须指定某个离散值，而中断和等时端点能指定少于或等于最大值的任何值。端点的另一个属性是传输方向，输入(数据从设备到主控制器)或输出(数据从主控制器到设备)。最后，每个端点都有一个端点号，其中包含输入输出方向，作为端点的地址使用。

当 USB 主控制器要求设备执行某些规则的功能时，USB 总线使用一个轮检(Polling)协议。当一个设备需要向主控制器发送数据时，主控制器必须注意到,并且向要发送数据的设备发出一个请读入求,使其发送数据。即 USB 设备不用传统方式中断主计算机，而是提供中断端点，主机则可以周期性地轮检中断端点。

总线上的信息流以每毫秒一帧数据的形式传输。总线驱动程序必须安排多个事务以使它们能被装入同一帧中，图 2-8 显示了这个过程。

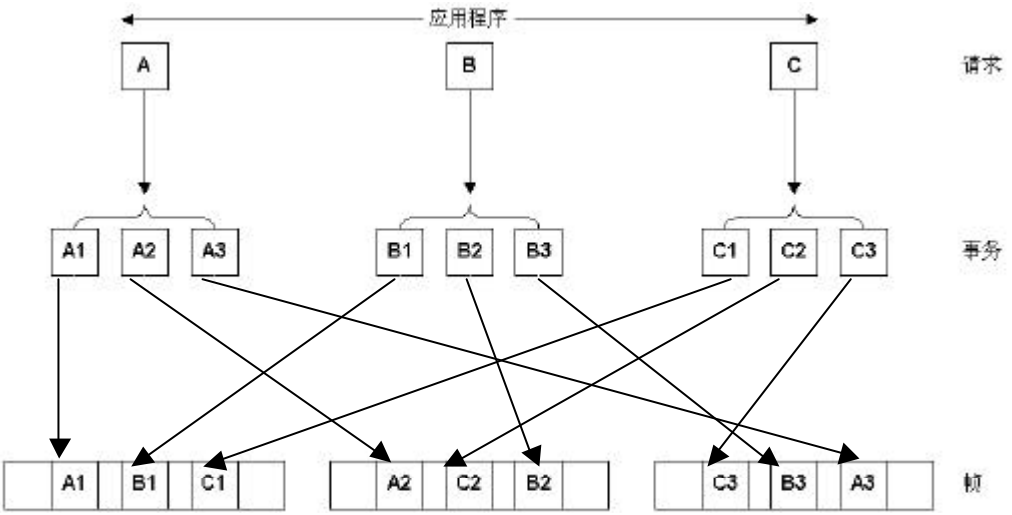

图 2-8 信息流中的事务和帧模型

在 USB 中，事务由一个或多个阶段(phase)组成。阶段有令牌(token)、数据(data)、握手(ack)三种类型。根据不同的类型，事务有一个令牌阶段、一个可选的数据阶段、和一个可选的握手阶段组成，如图 2-9 所示。在令牌阶段，主控制器向所有已配置的设备广播该令牌包。令牌包中含有设备地址，通常还有端点号，仅有被寻址的设备才会处理事务；当事务寻址设备时，任何设备都不读写总线。在数据阶段，数据被放到总线上。对于输出事务，主机把数据放到总线上，而被寻址的设备消耗这些数据。对于输入事务，情况相反，设备把数据放到总线上由主机消耗。在握手阶段，由设备或主机把握手包放到总线上，包中含有状态信息。当设备发出握手包时，ACK 包指出成功地接收了信息，NAK 包指出忙并且不试图接收信息，STALL 包指出事务被正确接收但在逻辑上无效。当主机发送握手包时，它仅能发送 ACK 包。没有发出握手包的事务就代表“在这个事务中出现了一个传输错误”。正在等待握手包的一方应认为缺少握手包可能是发生了错误并重试刚才的事务。

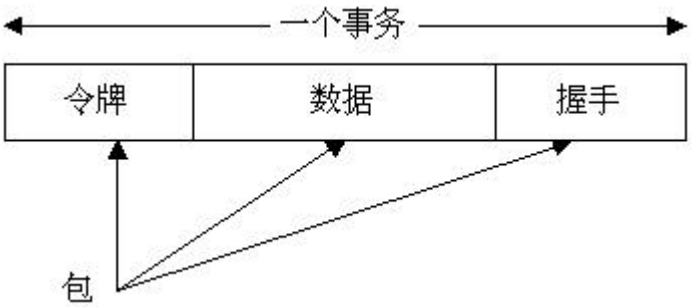

图 2-9 总线事务的阶段

2. 4. 1 控制传输

控制传输在主机和控制端点之间传送控制信息。例如，在操作系统配置 USB 设备过程中，有一步就是用控制传输从物理设备中读出各种描述符(descriptor)，配置过程的另一步是利用控制传输设置一种可能的配置并使能一个或多个接口。控制传输是一种纠错传输，在遇到传输错误时重试三遍，如果错误仍存在就放弃传输并向上层软件报告错误。如表 2-1 中所指出的，控制端点所能指定的最大数据传输长度为 8、16、32、64 字节。一个单独的事务可以包含少于最大数据传输长度的数据但不能多于。

控制事务在 USB 中具有最高优先级。设备必须处理控制事务。此外，总线驱动程序为控制事务保留 10%的带宽。对于轻量级的负载，主机能确保在 1 毫秒内完成一个控制事务。对于重量级的负载，未完成的控制传输将被强制到下一帧中传输，因此会产生一些延迟。

每个 USB 设备至少应有一个编号为 0 的控制端点以响应控制事务的输入输出。严格地讲，端点应属于配置，但端点 0 是一个例外，因为它是设备默认控制管道的终点。端点 0 甚至在设备被配置前就被激活而不管其它端点是否有效。除了端点 0，一个设备没有必要拥有另外的控制端点(尽管 USB 规范中允许有这种可能)，因为端点 0 对于大部分控制请求都可以很好地完成。如果定义了一个厂商专用的请求并且该请求不能在一帧中完成，就应该创建额外的控制端点以防止设备接收器被新事务抢先。

每个控制事务包括一个 SETUP 令牌，之后可以带一个可选的数据阶段和一个握手阶段，在握手阶段设备会发出 ACK 包或者 STALL 包，或者根本就不响应，如图 2-10。设备必须在任何时间都能接受控制传输，并且不能用 NAK 响应控制端点忙。向控制端点发送一个无效请求将导致 STALL 响应，但当设备再次接收到一个 SETUP 包时会自动清除停止状态。这个特殊的 STALL 在 USB 规范中被称为协议停止(protocol stall)。

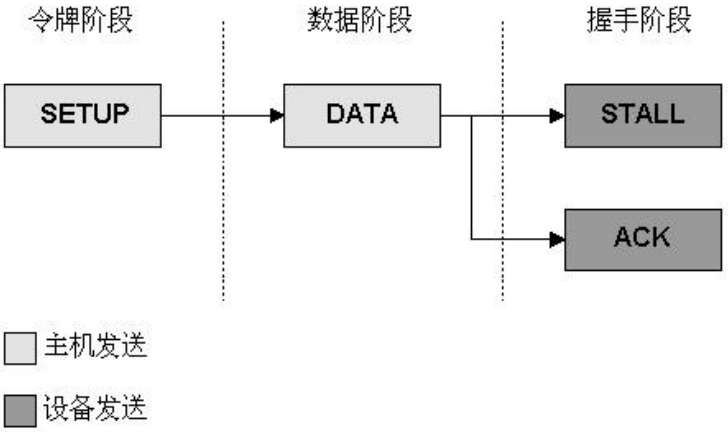

图 2-10 控制传输中的三个阶段

在下面的表 2-2 和后面的数据结构图表中，按照数据在 USB 导线上的传输顺序，列出其中的数据字节，但是每个字节中的位是以高位开始。在传输线上的字节传输是从低位开始的，但主机软件和设备固件通常以相反的方向使用字节。Intel 计算机和 USB 总线协议使用小结尾的数据表达，即低位字节占用低地址。AT89C52 微处理器，使用大结尾的数据表达。固件程序设计必须注意这一点,作出大小端字节序的调整。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 2-2。

表 2-2　SETUP 包数据的格式

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 2-3)
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.

各个域的具体说明如下：

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 2-3 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。

bRequest 域

这个域标识特别的请求。bmRequestType 域的 Type 可修改此域的含义。表 2-3 仅说明定义 Type 字位为 0 即标准设备请求时 bRequest 域值的含义。

wValue 域

此域用来传送当前请求的参数，随请求不同而变。

wIndex 域

wIndex 域用来表明是哪一个接口或端结点，图 2-11 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 2-12 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 2-11 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 2-12 所指为接口时 WIndex 格式

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。

标准设急请求

本部分描述的所有 USB 设备都定义的标准设备请求，表 2-3 将它们列出，而表 2-4、2-5 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已经被配置了的，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、或端点的远程唤醒功能，特性选择符的值在表 2-6 中说明。

表 2-3 标准设备请求

bmRequestType	bRequest	wValue	Windex	wLength	Data
00000000B 00000001B 00000010B	CLEAR_FEATURE	特性选择符	0 接口号 端点号	0	无
10000000B	GET_CONFIGURATION	0	0	1	配置值
10000000B	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	0	接口号	1	可选设置
10000000B 10000001B 10000010B	GET_STATUS	0	0 接口号 端点号	2	设备, 接口 ,或 端点状态
00000000B	SET_ADDRESS	设备地址	0	0	无
00000000B	SET_CONFIGURATION	配置值	0	0	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	0 接口号 端点号	0	无
00000001B	SET_INTERFACE	可选设置	接口号	零	无
100000010B	SYNCH_FRAME	0	端点号	2	帧号

表 2-4 标准请求码

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

续表 2-4

SET_DESCRIPTOR	7
GET_CONFIGURATION	8
SET_CONFIGURATION	9
GET_INTERFACE	10
SET_INTERFACE	11
SYNCH_FRAME	12

表 2-5　描述表种类

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

表 2-6 标准特性选择符

特性选择符	接受者	值
DEVICE_REMOTE_WAKEUP	设备	1
ENDPOINT_HALT	端点	0

2. 4. 2 批量传输

批量(bulk)传输能在主机和批量端点间一次传输最多 64 字节的数据。就象控制传输一样，批量传输是纠错传输，不同的是批量传输没有任何延迟保证。如果主机发现帧中除了预定带宽外还有剩余，它就把等待的批量传输送往总线。

图 2-13 显示了组成批量传输的各个阶段。传输开始于表明传输方向的 IN 或 OUT 令牌，这些令牌同时还指定目标设备和端点。对于输出事务，后面是一个数据阶段，把数据从主机送到设备，最后在握手阶段设备向主机提供状态反馈。如果端点正忙不能接收新数据，它就在握手阶段发出一个 NAK 包，主机以后会重试该事务。如果端点处于停止(STALL)状态，它就在握手阶段发出一个 STALL 包，主机在重试前必须清

除端点的停止状态。如果端点正确地接收并处理了数据，将在握手阶段返回一个 ACK 包。最后一种情况是端点因为某种原因没有正确地接收数据并且在握手阶段也没有发出应答包，主机将检查端点是否应答，并且自动重试三次原事务。

IN 令牌后面是一个批量输入传输。如果端点将数据准备好，设备将把数据发往主机，主机或者回应一个 ACK 包指出数据无误地接收，或者保持沉默以指出发生了某种错误。如果主机查出一个错误，ACK 的缺发将使设备数据保持有效，之后主机将重试输入操作；如果端点忙或停止，设备将会在数据阶段发出 NAK 或 STALL 握手包而不是数据。NAK 指出主机应在以后重试该输入操作，STALL 需要主机先清除端点的停止特征再重试输入操作。

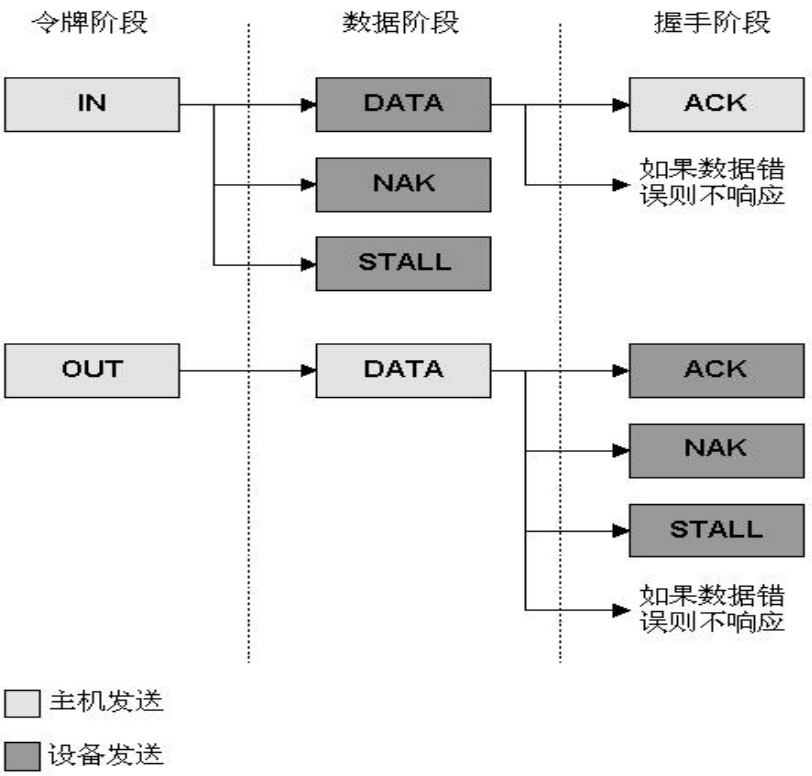

图 2-13 批量传输和中断传输中的阶段

2. 4. 3 中断传输

中断(interrupt)传输在总线操作和涉及设备方面上几乎与批量传输完全相同。它能把最大 64 字节的数据无误地在主机和中断端点间传输。中断传输和批量传输仅有的不同是它必须考虑延迟。中断端点需指定一个范围在 1-255 毫秒内的询查周期。主机保留足够的带宽以确保在指定频率上直接向中断端点发出 IN 或 OUT 事务。

USB 设备不生成异步中断，它们总是响应循检，Microsoft 的主控制器驱动程序把中断端点描述符中指定的循检周期简化为不大于 32 的 2 的幂。例如，一个指定循检周期为 31 毫秒的端点实际上是以 16 毫秒进行循检。指定周期在 32-255 毫秒之间的循检实际上是以 32 毫秒为循检周期。

2. 4. 4 等时传输

等时(isochronous)传输可以在一个总线帧内最多传输 1023 字节数据。因为等时传输有周期保证，因此特别适用于时间敏感的数据传输，如音频信号，但这种周期保证是有代价的，等时传输在数据出错时不会自动重试。USB 设计者假定等时数据流的接收者允许偶尔的数据丢失。

等时传输由 IN 或 OUT 令牌阶段后跟一个数据阶段组成。因为不进行数据纠错，所以等时传输没有握手阶段。如图 2-14。

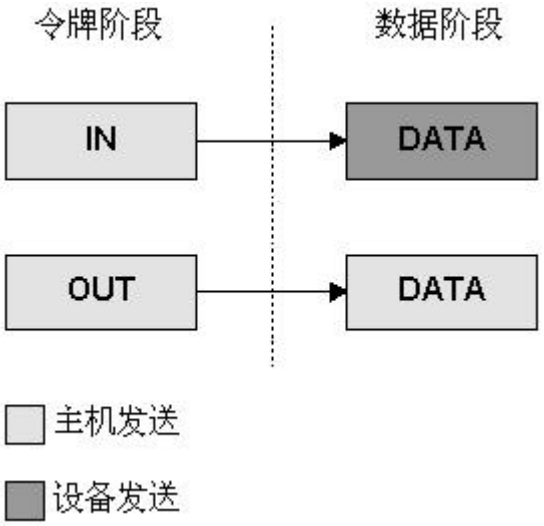

图 2-14 等时传输中的阶段

主机为等时传输和中断传输保留最多 90%的带宽。

2. 5 本章小结

本章从步进电机控制装置的通信架构模型开始，对其物理和逻辑上的通信模型进行了介绍。依照主机侧的数据流向，对主机内的系统驱动调用层次进行了详细的分析。最后依照 USB 总线物理层的传输模式，对其四种传输模式进行了介绍，其中对设备配置的重点—控制传输，进行了深入的介绍。