《对 USB 协议层的深层分析》

文档说明

文档名称	对 USB 协议层的深层剖析
文档作者	frank_wang 北航 E-Mail： frank_wang@263.net frank_wang@buaa.edu.cn Web-Site： embedusb.51.net 欢迎赐教！
完成日期	Apr 17th,2004
内容说明	针对 USB 设备或主机开发过程中的调试过程，如果能对协议的过程和细节内容了解地非常清楚，则对于调试过程非常有帮助，就意味着可以预测主机（或设备）下一步应该做什么，此文档目的在于描述控制传输和批量传输的协议细节。作为对《USB 项目技术报告》(frank,2002 年 12 月)一文的补充
版权状态	保留所有权利，请勿用于盈利！欢迎与《USB 项目技术报告》一文一起转载，用于学习。
版 本	V1
当前状态	完成

A.1　基本分组格式

USB 总线上传输的分组（Packet）格式有四种：令牌（Token）、帧开始（SOF）、数据（DATA）和握手（HandShake）四种。SOF 分组对于本文中所分析的协议内容没有多少影响，下面只考虑令牌、数据和握手分组三种情况。

A.1.1　令牌分组

令牌分组(Token Packet)由 PID、ADDR 和 ENDP 构成，其中 PID 指定了分组是 IN、OUT 还是 SETUP 类型。对于 PID 为 OUT 和 SETUP 类型的传输，地址和端点域唯一地确定了接下来将收到数据（DATA）分组的端口。对于 PID 为 IN 类型的传输，这些域唯一地确定了哪个端口应该传送数据分组。只有主机能发出令牌分组。结合 USB 协议中的上述论述，可以简要地总结出以下内容：

1、发往地址 0 和端点 0 的 SETUP 类型的分组，是主机发出的配置包，它属于控制传输，其接下来的传输内容，我们有理由根据控制传输的特点来期许，这一点后面介绍。如图图 A-1，是一个典型的配置分组(SETUP Packet)。

说明：对于英文术语的翻译，可能不同人的习惯不一样，我根据大家都熟悉的叫法，并使其前后意思符合逻辑。且第一次出现时在括号中增加英文原文。错漏之处，望不吝赐教！

Sync	SETUP	ADDR	ENDP	CRC5	Idle
00000001	0xB4	0x0	0x0	0x2	0x3

A、完整的数组序列

a

SETUP	ADDR	ENDP
0xB4	0x0	0x0

B、简化的数据序列

图 A-1　配置分组

注意：图 A-1 中，A 为完整的 USB 总线上的数据序列，但为了简洁，本文后面都用 B 的形式，即隐藏了 sync、CRC 和 Idle 域的内容。这些数据来自于 USB 协议分析协，虽然我们在调试过程中可能没有协议分析协，但本文是用它来帮助我们理解协议，本文介绍的内容并非仅用来分析 USB 协议分析仪得到的数据。

2、发往地址 0 和端点 0 的 IN 和 OUT 类型的分组，是主机发出的读或写数据的控制类型的传输。其接下来的传输内容，我们有理由根据控制传输的特点来期许，这一点后面介绍。如图 A-2 和图 A-3 所示，分别是一个 IN 分组和一个 OUT 分组。

IN	ADDR	ENDP
0x96	0x0	0x0

图 A-2　配置端点读入

OUT	ADDR	ENDP
0x87	0x0	0x0

图 A-3　配置端点输出

注意：当主机完成 SetAddress 命令后，主机将为设备指定确定地址。上述分组中的地

址域将变成非 0，但配置阶段的各分组的端点域仍然为 0。

3、发往非 0 端点的批量端点的 IN 和 OUT 类型的分组，是主机与设备间的批量传输类型分组。其接下来的传输内容，我们有理由根据控制传输的特点来期许，这一点后面介绍。如图 A-4 和 A-5 所示，分别是是发往端点 1 的 IN 分组和端点 2 的 OUT 分组。

IN	ADDR	ENDP
0x96	0x1	0x1

图 A-4　批量端点读入

OUT	ADDR	ENDP
0x87	0x1	0x2

图 A-5　批量端点输出

A.1.2　数据分组

数据分组（DATA Packet）由 PID、包括至少 0 个字节数据的数据域和 CRC 构成（下面的 CRC 域均隐藏）。其中 PID 指示两种类型的数据包：DATA0 和 DATA1。两种数据包 PID 是为了支持数据切换同步（Data Toggle Synchronization）而定义的。结合 USB 协议中的上述论述，可以简要地总结出以下内容：

1、非零长度的数据分组：

数据分组的长度非零，则分组由 PID 和数据组成，如图 A-6 和 A-7 所示。

DATA0	DATA
0xC3	80 06 00 01 00 00 40 00

图 A-6　PID 为 DATA0 的非零长度的数据分组

DATA1	DATA
0xD2	12 01 00 01 00 00 00 08

图 A-7　PID 为 DATA1 的非零长度的数据分组

2、零长度的数据分组：

允许数据长度为 0 的数据分组。在 USB 里，数据长度为 0 和没有数据概念有些不一样。如图 A-8 所示。

DATA1	DATA
0xD2	

图 A-8　数据长度为 0 的数据分组

因为 USB 协议中规定，零长度的数据分组一般用于控制传送的状态阶段（大多数时候如此，在批量传输阶段，有时也见需要回复零长度的数据分组）。状态阶段的令牌分组是以相对前面的阶段的数据流方向的变化来刻划的（比如前面为 OUT，则状态阶段的令牌类型就是 IN），并且其数据分组总是使用 DATA1 PID。

A.1.3　握手分组

握手分组是除令牌分组和数据分组之外的另一种分组。握手分组仅由 PID 构成。握手分组用来报告数据传输的状态（即握手分组总是跟在 DATA0 或 DATA1 分组的后面）。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。握手分组总是在某种传输（比如控制传输、批量传输）的握手阶段（Phase）中被返回，也可在数据传输阶段代替数据被返回。

有 3 种类型的握手分组：

1、ACK 表示数据分组没有位填充或数据域上的 CRC 错，并且数据 PID 被正确收到。如果在数据分组后收到了 ACK 握手分组，那算是一个好消息，相当于在王小丫的节目中听到了“恭喜你，你答对啦!”，如图 A-9 和 A-10 所示，为分别在 DATA0 和 DATA1 数据分组后的握手分组。

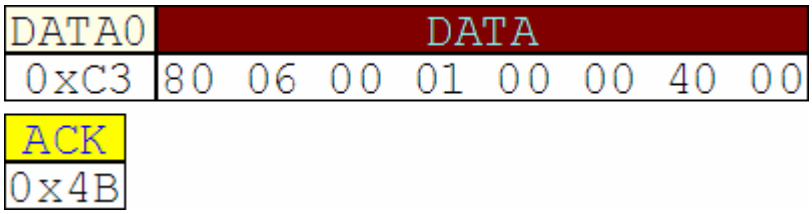

图 A-9　DATA0 后的 ACK 握手分组

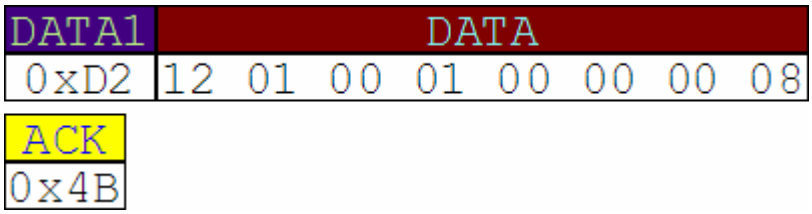

图 A-10　DATA1 后的握手分组

2、NAK 表示功能部件不会从主机接受数据（对于输出传输），或者功能部件没有传输数据到主机（对于输入传输）。NAK 仅由功能部件在输入传输的数据阶段返回，或在输出传输的握手阶段返回。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。如图 A-11 为功能部件不从主机接受数据时的握手分组；而图 A-12 为功能部件没有传输数据到主机时的握手分组。

NAK 是主机与不同速度的设备之间匹配速度的关键所在，NAK 表明 USB 设备芯片与主控芯片之间的数据传输是正常的，只不过设备现在无法处理所传输的数据。此时主机应该重试，直到设备发出 ACK 为至，重试的次数越多，则体现到宏观上，这个 USB 设备就会越慢，如果是 U 盘，那么速度慢的 U 盘肯定 NAK 了许多次主机。如果将其比喻为追女孩子，相当于女孩（设备）“拒”了许多次男孩（主机），才答应其要求。这时之所以加引号，是说并非出现了错误，而是正常现象，“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”——最终会成功的，只是需要有耐心和恒心，这一点不同于后面要讲的 STALL。

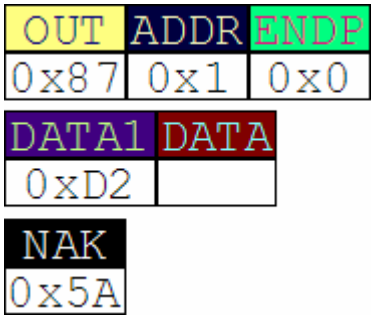

图 A-11　功能部件不会从主机接受数据

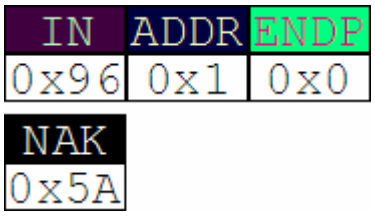

图 A-12　功能部件没有传输数据到主机

3、STALL 作为输入令牌（IN）的回应，或者在输出事务的数据阶段之后由功能部件返回，如图 A-13 所示。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。设计 USB 协议的这帮人，一定深谙男女之事，看到没有，男孩子在任何时候都不允许 STALL 女孩子，而且，一旦被 STALL 掉，那是不同于 NAK 的，NAK 的话，只是女孩子在考验男孩子的耐心和诚意，主机应该重试直至成功。而当得到 STALL 时，相当于“别费功夫了，没戏”。所以，当出现 STALL 时，主机不要简单重试，而是要从软件上对传输过程进行干预。至于被 STALL 的原因，大致有两种情况：“功能 STALL”和“协议 STALL”，在这里不详述。

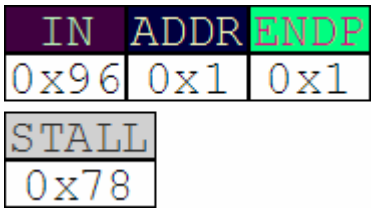

图 A-13　功能部件拒绝 IN 令牌

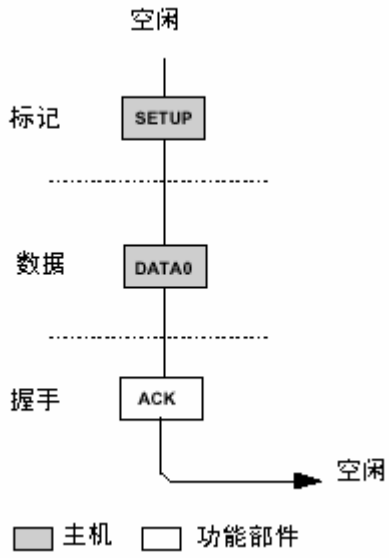

图 A-14 控制传输的建立阶段

图 A-15 是完整的建立阶段的协议数据，这个阶段的含义，可以从数据分组的数据域推断出，本例中“80 06 00 01 00 00 00 12 00”序列的含义是获取设备描述符（Get Device Descriptor），其他情况下，可以参考 USB 协议第九章中的规定，或参考《USB 项目技术报告》一本。

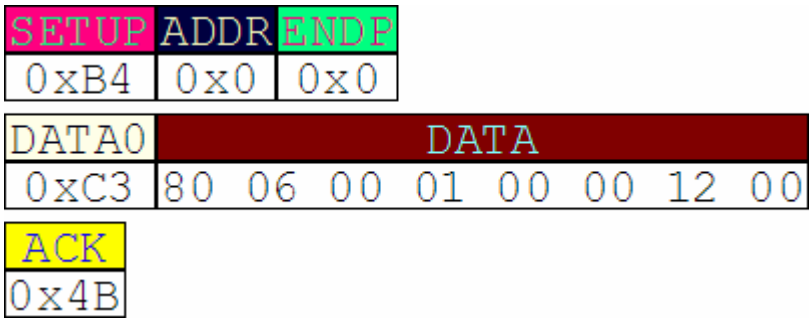

图 A-15 一个典型的控制传输的建立阶段

这里先区分一下可能存在的误解。建立和状态都是控制传输的两个阶段。建立阶段又由 SETUP 令牌分组、DATA0 数据分组、ACK（或 NAK\STALL）握手分组组成。而状态阶段是控制传输的另外一个阶段，他的作用和意义不同于 ACK 分组，ACK 握手分组是报告数据传输的状态的，而状态阶段则是反应建立阶段（由 SETUP\DATA\ACK 组成）的状态的。

如图 A-16 所示，在建立阶段和状态阶段中间，可能存在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。在数据阶段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此端点的数据分组大小（Max Packet Size），则会在多个数据传输阶段（输入或者输出）来传输，每次都按最大分组数据传输，任何剩下的数据都作为剩余在最后的传输阶段中被发送。

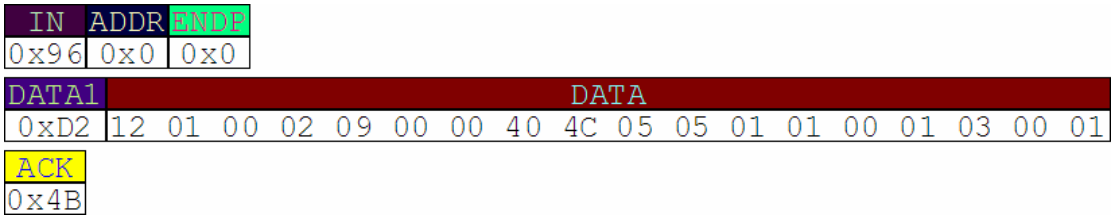

图 A-16 控制传输的数据阶段

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（即如果数据阶段为 IN 分组，则状态阶段为一个 OUT 分组；反之，如果数据阶段为 OUT 分组，则数据阶段为 IN 分组），且总是使用 DATA1 PID，如图 A-17 所示。

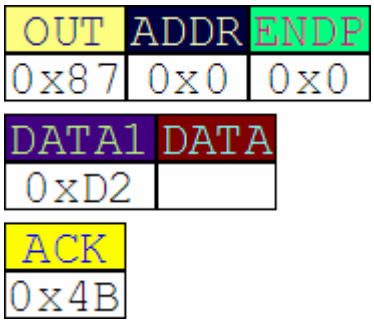

图 A-17 控制传输的状态阶段

例如，如果数据阶段是由输出事务构成的，则状态是单一的输入事务。如果控制传输序列没有数据阶段，那么它由建立阶段和其后的由输入分组构成的状态阶段构成。图 A-18 说明了传输阶段顺序、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。

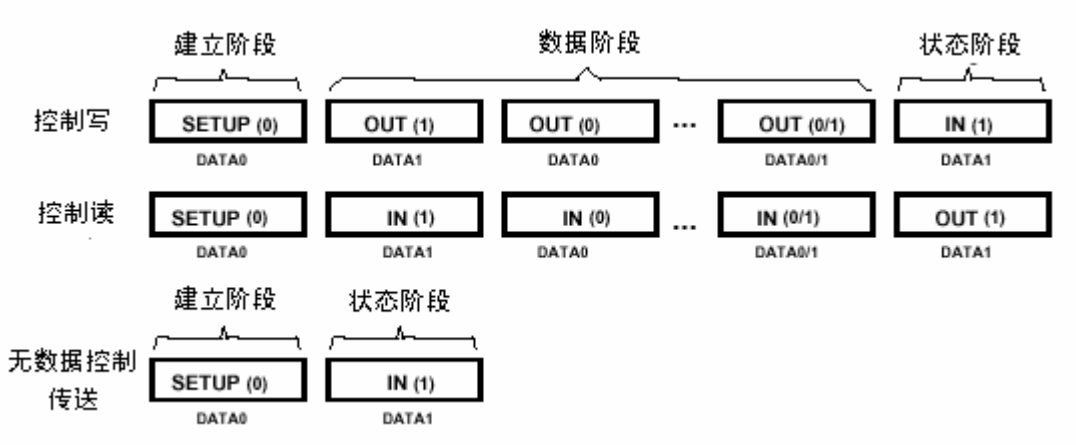

图 A-18 控制读写序列

在上面的读写序列中，每个阶段都由（令牌分组、数据分组和握手分组组成），图中只显示了令牌分组。当然，状态阶段没有数据分组。

当控制端点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点访问返回 STALL 握手，直到收到建立 PID 为止。端点收到建立 PID 之后，不应返回 STALL 握手。

下面这个图可能可以将我们前面提到的分组、传输类型、事务几个概念之间的联系交待得更清楚。结全上面的叙述，可以设想图中的 Transaction1-0 为控制传输的建立阶段（或称为一个建立事务），Transaction1-1 为数据阶段（或称为数据事务），Transaction1-2 为状态阶段（或称为状态事务），而每一个阶段都是由至少两个分组（令牌分组和握手分组）组成，如果这个阶段中包括数据，则可能有一个或一个以上的数据分组，就象图 A-19 中的 Token Packet、Data Packet 和 Handshake Packet 那样。再回想一下前面提及的不要混淆状态阶段和握手分组，即 Transaction1-2 与 Handshake Packet 的层次和意义是不一样的。

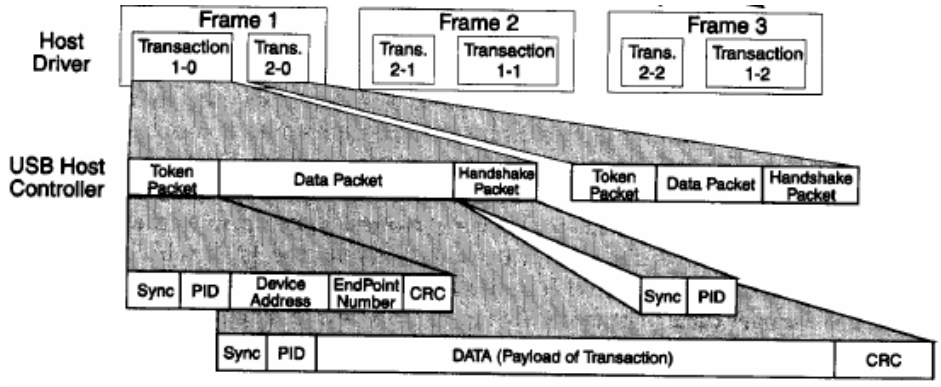

图 A-19 传输、传输阶段与分组之间的关系

A.2.2 批量传输

如果已经对以上内容有了很好的理解，那么理解批量传输就非常容易了。

批量传输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 A-20 所示，批量传输是由令牌、数据和握手分组构成的具有三个阶段的付输。

在某些流控制和挂起条件下，数据阶段会被握手信号替换，从而产生了没有数据传输的两个阶段的批量付输。

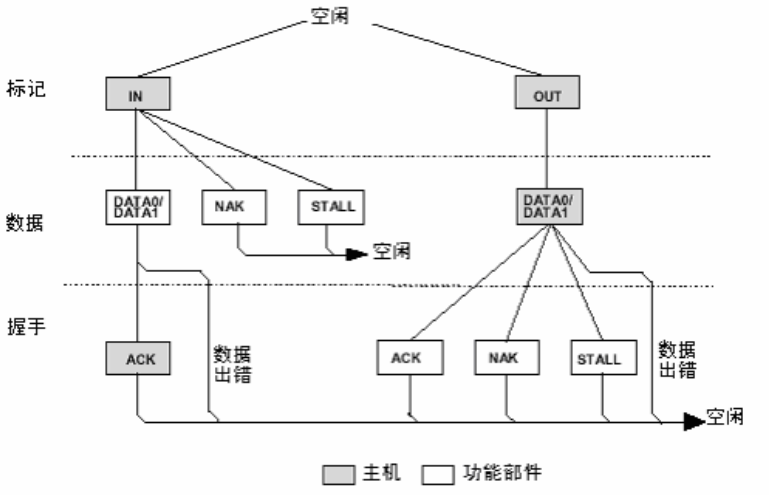

图 A-20 批量处理事务格式

如图 A-21 所示，当主机准备好了接收批量处理数据的时候，它发出 IN 令牌（当然，发向相应正确的地址号和端点号）。功能部件端点通过返回数据分组，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 表示端点永久地被停止，需要 USB 系统软件干涉。如果主机收到合法的的数据分组，则它用 ACK 握手来应答。如果收到数据时主机检测到错误，它不返回握手分组给功能部件。

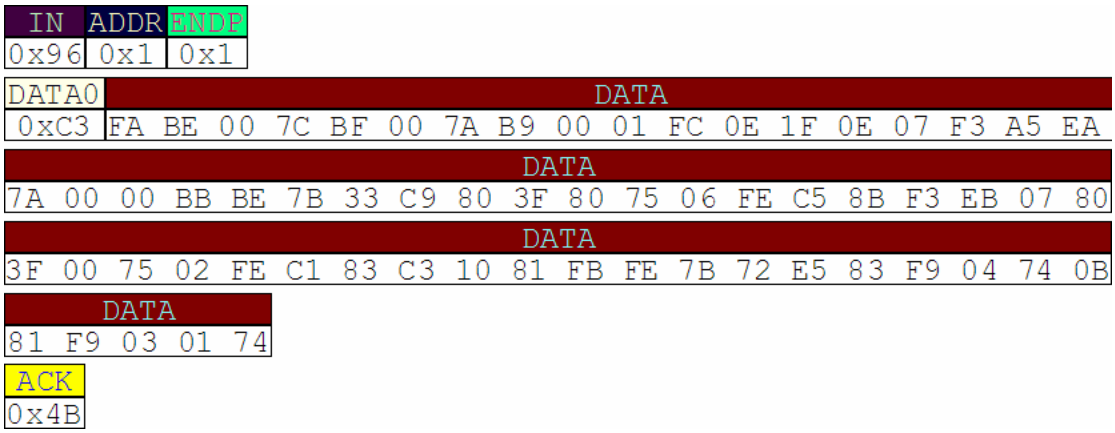

图 A-21 一个典型的批量 IN 传输

如图 A-22 所示，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分组的 OUT 令牌分组。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 表示数据分组无错地接收到，通知主机可以发送下一个分组；

◆ 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；

◆ 如果端点被停止，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。

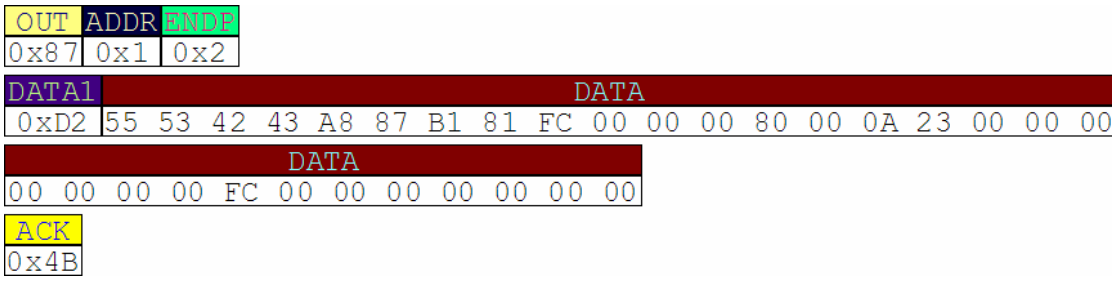

图 A-22 一个典型的批量 OUT 传输

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 A-20 说明了时序位和数据 PID 在成批读和写中的用法。数据分组同步经数据时序切换位(Sequence Toggle Bit）和 DATA0/DATA1 PID 的使用而达到。当端点经历配置事件(Configuration event)的时候，批量处理端点的切换时序被初始化为 DATA0。

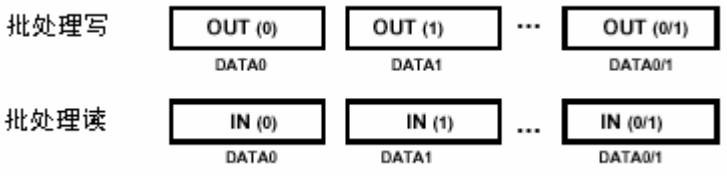

图 A-23 批量处理读和写

主机总是通过配置事件初始化总线传送的第一个数据传输为 DATA0 PID。第二个数据传输使用 DATA1 PID，并且，剩余的后继数据传送轮流切换。数据分组发送器根据 ACK 的接收情况来切换而接收器根据数据分组的接收（receipt）和验收（acceptance）的情况切换。

这里需要说明的是，数据位的切换，是在每个端点内部进行的。由于 USB 协议规定控制端点的输出和输入为同一个端点号，即端点 0；而其他端点的输入和输出端点号不能相同，对应可能是端点 1 和端点 2。所以，对于端点 0，如果先有一个 IN 分组，其后的数据分组是 DATA0，则下一个数据分组不是管是 IN，还是 OUT，一定是 DATA1 分组，因为控制端点

的 IN 和 OUT 是在同一个端点，而数据位的切换是按端点进行的。相比较批量端点的情况，因为输入输出各有一个端点，所以，当先有一个 IN 分组，其数据分组是 DATA0，而下一个是 OUT 分组时，其数据分组仍将是 DATA0，如果紧接着又在 IN 分组后又数据分组，才应该是 DATA1。当然，控制传输过程中的状态阶段的数据分组，还应该遵守特别的规定，要求总是在 DATA1。

A.3 补充说明

本文只对底层协议进行了说明，对于在配置阶段到底主机会发什么样的配置命令（这是 USB 设备开发者关心的问题）、或主机应该发出什么样的命令（这是 USB 主机开发者关心的问题）；以及对于批量传输阶段主机会发出什么样的命令或主机应该按什么顺序发出什么样的命令，应该参考 USB 协议第九章和相应的 SCSI 命令协议文档，这些内容在《USB 项目技术报告》一文中有较多的阐述，此文作为对其的补充和深入。