USB 设备驱动程序的设计与开发

杨 龙[1]， 刘 岩[1]， 董绪荣[2]
(1. 装备指挥技术学院 研究生部，北京 101416； 2. 装备指挥技术学院 测量控制系，北京 101416)

摘 要：USB 接口具有方便快速等优点，已经发展成为一种比较普遍的计算机与外设之间的接口。首先给出了 USB 系统的整体结构组成；然后从 Windows 驱动程序的结构入手介绍 Windows 驱动程序的工作原理；介绍了如何利用 Windows DDK 编写 USB 设备的驱动程序；给出了具体的应用实例。

关 键 词：USB；Windows DDK；驱动程序
中图分类号：TP 311.1
文献标识码：A **文 章 编 号**：CN11-3987(2003)01-0090-05

USB(universal serial bus)是 1995 年 Microsoft、Compaq、IBM 等公司联合制定的一种新的 PC 串行通信协议。作为一种外设接口，它较其它接口存在以下 4 个优点：①使用方便。使用 USB 接口可以连接多个不同的设备，支持热插拔，不涉及中断请求(interrupt request，IRQ)冲突等问题，能真正做到“即插即用”；②速率高。USB 接口的最高传输率目前可达 12 Mb/s，比串口快了整整 100 倍，比并口也快了十多倍；③连接灵活。USB 接口支持多个不同设备的串行连接，连接的方式也十分灵活，既可以使用串行连接，也可以使用中枢转接头(hub)；④独立供电。USB 接口提供了内置电源，能向低压设备提供 5V 的电源，新设备不需要专门的交流电源。

正是基于以上这些优点，开发 USB 接口的设备已成为一种发展趋势。

1 应用系统举例

在笔者开发的一套数据采集系统采用了 USB 接口。采集系统的整体结构如图 1 所示。

系统工作过程：微处理器按照一定的采样率收集所有传感器的数据，然后送到 USB 接口芯片，通过控制接口芯片实现与主机的数据通信。

要实现 USB 传输，需要 2 方面的软件支持：微处理器的控制软件和主机的控制软件。这里重点介绍主机软件。由于目前所有的开发工具中还没有对 USB 进行控制的函数和控件，所以必须自行开发 USB 设备驱动程序。本文主要介绍如何利用 Windows 驱动程序开发包(driver development kit，DDK)在 Windows2000 下进行 USB 设备驱动程序的设计与开发。

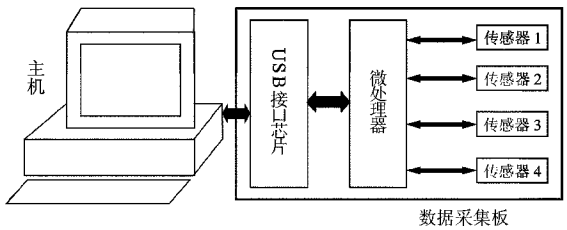

图 1 系统结构框图

2 USB 系统简介

2.1 USB 系统的硬件组成

组成 USB 系统的硬件包括主控制器和 USB 设备，其中 USB 设备又包括集线器(hub)和一般的功能性设备，如活动磁盘，游戏杆等。设备之间的物理连接采用分层星型互联结构，如图 2 所示。任何 USB 设备都必须提供一个标准的 USB 接口以满足 USB 协议。

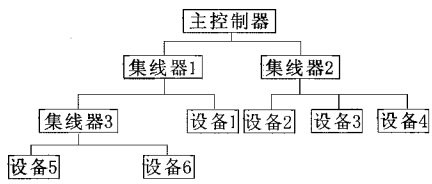

图 2 USB 硬件设备互联

收稿日期：2002-06-12
作者简介：杨 龙(1977－)，男(汉族)，陕西西安人，硕士研究生； 董绪荣(1963－)，男，教授，博士生导师.

2.2 USB 数据传输

USB 设备对于 USB 系统来说是一个端点的集合,端点被分成组,一组端点实现一个接口,设备端点和主机软件之间利用管道进行联系。设备驱动程序就是通过这些接口和管道来与设备进行通信。三者之间的关系如图 3 所示。

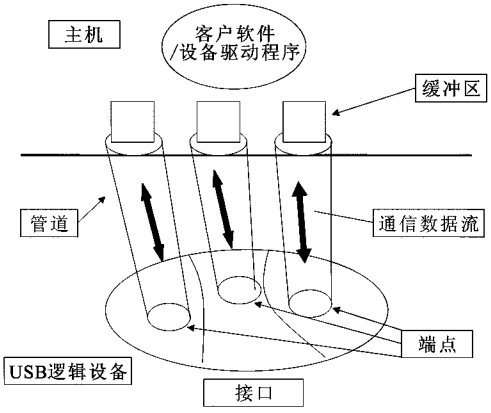

图 3　USB 数据传输模型

USB 数据传输就是指发生在主机软件和 USB 设备上特定端点(endpoint)之间的数据交换,一个设备可以具有若干个管道。一般情况下,一个管道中的数据传输与其他管道中的数据传输是相互独立的。这种发生在管道中的数据流动共有 4 种基本类型:控制传输、批量传输、中断传输和同步传输。

USB 设备驱动程序都必须使用这些管道和接口来管理设备,而不是直接通过内存或 I/O 端口存取来管理,这一点与其它总线如 PCI、EISA、PCMCIA 等不同。

3 Windows 驱动程序工作原理

3.1 Windows 内核模式驱动程序结构

驱动程序与应用程序之间最大的差别在于驱动程序的控制结构。应用程序从头至尾都在 Main 或 Winmain 函数的控制下执行,由这 2 个函数确定子例程的执行顺序。而驱动程序不存在 WinMain 或 Main 这样的入口,而只是一个由 I/O 管理组件根据需要调用的子例程的集合,I/O 管理组件根据不同的情况调用驱动程序中不同的例程[1]。构成内核模式驱动程序的主要例程包括:初始化和清除例程、I/O 系统服务调度例程、数据传输例程和资源回调例程等。以下是一个简单驱动程序的框架结构:

```
NTSTATUS DriverEntry(IN PDRIVER_OBJECT DriverObject, IN PUNICODE_STRING RegistryPath)    //初始化例程
{
    // 声明其他例程的入口点
    DriverObject->MajorFunction[IRP_MJ_CREATE] = XX_Create;
    DriverObject->DriverUnload = XX_Unload;
            ……
    // 为设备创建设备对象
    CreateDevice(pDriverObject ulDeviceNumber);
}
VOID XX_Unload (IN PDRIVER_OBJECT pDriverObject)        //清除例程
{
    // 释放所有占用的缓冲区
    if (pDevExt->deviceBuffer != NULL)
      ExFreePool(pDevExt->deviceBuffer);
    // 删除建立的符号链接
    IoDeleteSymbolicLink(&pLinkName);
    // 删除设备对象
    IoDeleteDevice( pDevExt->pDevice );
}
NTSTATUS XX_Create (IN PDEVICE_OBJECT pDevObj,IN PIRP pIrp)     // I/O 系统服务调度例程
{
    // 处理来自应用程序中 CreateFile 的调用请求
            ……
    return STATUS_SUCCESS;
}
VOID StartIo(IN PDEVICE_OBJECT pDevObj, IN PIRP pIrp)       //数据传输例程
{
    // 分配所需的任何资源,并设置设备运行
            ……
}
```

说明:这里给出的只是一个简单的框架,并没有列出驱动程序中所需的所有例程。详细情况请参阅参考文献[1]。

3.2 Windows 系统的 I/O 过程

为了实现操作系统的兼容性、可移植性和其他一些性能,Windows 操作系统不允许用户应用程序直接对各种硬件设备进行操作,而由操作系

统的I/O管理组件管理几乎所有的输入/输出操作请求,并最终由驱动程序完成所有的操作。具体过程如下[2]:

1)对I/O的每种用户请求,I/O管理程序从非分页系统内存中分配一个输入输出请求包(I/O request packet, IRP),基于用户请求的文件句柄和I/O函数,I/O管理程序将IRP传递给合适的驱动程序调度例程;

2)调度例程检查请求的参数,如果合法,把IRP传递给驱动程序的Start I/O例程;

3)Start I/O例程使用IRP的内容开始一个设备操作;

4)当操作完成,驱动程序的DpcForIsr例程将最后状态代码保存在IRP中,并将它返回给I/O管理程序;

5)I/O管理程序使用IRP中的信息完成请求,并将最后状态发送给用户。

3.3 Windows驱动程序模型的特点和构成

随着操作系统的发展,驱动程序结构也经历了一个不断完善的过程。Windows驱动程序模型(windows driver model, WDM)是一种在当前比较新的模型结构,他是建立在物理设备对象(physical device object, PDO)和功能设备对象(functional device object, FDO)的结构化分层基础上。WDM模型为了适应即插即用系统,重新定义了驱动程序分层,它至少存在总线驱动程序和功能驱动程序,根据需要还可以选择过滤器驱动程序[3]。

通常情况下,连接到总线的每个物理部件都有一个PDO,承担由硬件实现的低级设备控制的责任。更高级软件则都存在一个FDO。驱动程序和设备对象之间的对应关系如图4所示。

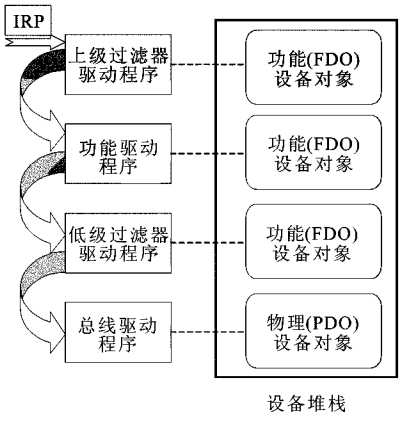

图4 驱动程序WDM模型

当存在I/O操作时,IRP请求包首先被送到上级过滤器驱动程序,然后被一级一级往下传递,最后由总线驱动程序完成对设备的操作。

4 USB设备驱动程序结构设计

4.1 USB系统软件的结构

USB系统驱动程序采用WDM结构,具体结构如图5所示。其中设备驱动程序位于整个结构的最顶层,它不直接操作硬件,而是通过一个接口传递请求。

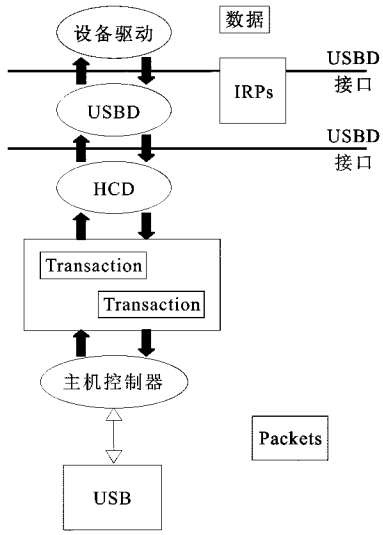

图5 USB驱动程序的WDM结构

这里USBD接口是WDM分层结构的一部分,是操作系统USB驱动程序堆栈为内核模式设备驱动程序提供的接口。

4.2 USB设备驱动程序的初始化和配置过程

当设备驱动程序被系统加载时,首先进入DriverEntry()入口例程,在该例程中主要设置程序中要处理的各个MajorFuncton IRP和AddDevice的处理函数入口点。其作用与DOS系统中设置中断向量表的操作类似,与指出哪个中断该由哪个中断处理函数进行处理一样,在这个例程中不分配任何资源。

当USB堆栈检测到设备后,即插即用管理器(plug and play manager)调用DriverEntry例程中设置的AddDevice处理函数进行资源的分配和FDO的创建。以下是该例程的一段伪代码:

```
XX_PnPAddDevice(IN PDRIVER_OBJECT DriverObject, IN PDEVICE_OBJECT PhysicalDeviceObject)
{
//为设备创建一个符号链接,该符号链接用于应用程序的CreateFile调用;
```

```
//创建设备对象和设备扩展；
//初始化设备对象和设备扩展中的各个变量；
//向 PDO 查询设备的电源性能并保存到 FDO 的设备扩展中；
//因为设备可能暂时不被使用，关闭 USB 设备电源
}
```

当 USB 堆栈分配完资源后，将会发送一个 IRP_MN_START_DEVICE，此时 DriverEntry 例程中指定的 IRP_MJ_PNP 处理函数将被激活。针对 IRP_MN_START_DEVICE，处理函数将完成如下工作：获取设备描述符；获取配置描述符；从配置描述符中选择一个接口，并用该接口配置设备。这些是 USB 驱动程序与其它驱动程序的区别之处。如果配置成功，将配置信息存放在设备扩展中。这时，驱动程序已经完全就绪，等待应用程序发出与设备进行通信的请求。

4.3　数据传输过程

当应用程序与设备进行数据传输时，应用程序首先必须调用 CreateFile 打开设备，然后利用返回的句柄对设备进行读写操作，用户在调用 CreateFile 时使用的文件名一般为“＜由 GUID 产生的外部设备名＞.\yy”，这里 yy 是内部管道 ID。对于驱动程序，当应用程序调用 CreateFile 时，相应于 IRP_MJ_CREATE 的处理函数将被激活，并完成如下工作：获取当前 IRP 的堆栈指针，通过堆栈指针获得要打开的文件名，然后通过文件名找到内部管道 ID，若对当前接口此管道 ID 合法，则将该管道打开，并将返回给应用程序一个合法句柄。

管道被打开后，用户应用程序将会调用 ReadFile 或 WriteFile 从设备读取数据或向设备写数据。此时驱动程序中相应于 IRP_MJ_READ 或 IRP_MJ_WRITE 的处理函数被激活。这些处理函数将读写请求转换成相应的 USB 请求数据包（USB request packet，URP）传递给下层驱动程序，并等待 I/O 处理的最后完成。

5　驱动程序的编译和调试

驱动程序的编写可以在任何 C 语言环境中进行，但 Windows DDK 对于编译和链接有一套完整的环境，具体参照 Windows DDK 的文档介绍。最常用的方法是将所有源程序放在一个子目录下，并编写一个 SOURCE 文件。启动 DDK 的控制台编译程序，进入源程序所在的子目录，调用 Build －cz 命令，这时将会生成一个带.sys 后缀的驱动程序文件，这就是驱动程序的最终形式。调试驱动程序的方法也有很多种，如 WinDug，SoftIce 等。总的来说 SoftIce 功能更强大，使用也更方便。

6　应用实例及结论

在笔者开发的数据采集系统中，USB 控制芯片使用的是菲利普的 PDIUSBD12，该芯片共有 3 个端点，4 种工作模式，笔者选用的是模式 3。端点的定义如表 1。主机与设备端点 2 使用同步传输方式进行数据通信。驱动程序试验效果令人满意。

表 1　端点定义表

端点号	端点索引	传输类型	端点类型	方　向	最大信息包规格/byte
0	0	控制输出	默认	输出	16
	1	控制输入		输入	16
1	2	普通输出	普通	输出	16
	3	普通输入	普通	输入	16
2	4	同步输出	同步	输出	64
	5	同步输入	同步	输入	64

本文从工程应用的角度出发，分析了驱动程序的工作原理，给出了编写驱动程序的主要过程，尽量从总体上给出编写驱动程序的大体框架，并在主要细节方面做了必要的解释，为开发人员的工程开发提供一定的参考。

参考文献：

[1] Art Baker. Windows NT 设备驱动程序设计指南[M]. 北京：机械工业出版社，1997.

[2] Art Baker，Jerry Lozano. Windows 2000 设备驱动程序设计指南[M]. 北京：机械工业出版社，2001.

[3] 刘少峰，韦克平. USB 软件系统的开发[J]. 计算机应用研究，2002，(3)：31-35.

The Design and Development of a Driver for USB Device

YANG Long[1], LIU Yan[1], DONG Xu-rong[2]

(1. Department of Postgradute, the Academy of Equipment Command & Technology, Beijing 101416, China;
2. Department of Tracking and Control, the Academy of Equipment Command & Technology, Beijing 101416, China)

Abstract: For USB interface has the advantages of convenience and higher speed, it has become a popular interface between computer and peripheral device. The paper firstly introduces the whole structure of a USB system, then it describes the working principles of Windows device driver, and introduces how to write USB device driver with Windows DDK. It also gives an example of application at the end of paper, and the result is satisfactory.

Key words: USB; Widows DDK; driver

(责任编校:程万鑫)

征订启示

本刊征订全权委托天津"全国非邮发报刊联合征订服务部",读者欲订阅本刊只需直接向联合征订服务部汇款,注明本刊联订代号及刊名即可。

联合征订服务部联系方式

汇款地址:天津市大寺泉集北里别墅17号 联合征订服务部
邮　　编:300385
网　　址:http://WWW.LHZD.COM
E-MAIL: LHZD@PUBLIC.TPT.TJ.CN
联系电话:022-23973378;022-23962479
传　　真:022-23973378

本刊信息

联订代号:7985
刊　　名:装备指挥技术学院学报
定　　价:5元/期;全年:30元