Windows 下的 USB 设备驱动程序开发

王 萍，赵 刚

(西南石油学院计算机科学学院，四川 成都 610500)

摘 要：USB 设备驱动程序的开发是 USB 设备开发的一个重要组成部分。文章讨论了在 Windows 环境下驱动程序的设计原理，详细介绍了 USB 驱动程序的分层结构及 USB 数据传输的实现，并给出了一个利用 DriverStudio 开发的驱动程序实例，以此来加深对驱动程序开发过程的理解。

关键词：USB；WDM；设备驱动程序；DriverStudio

1 引言

USB 通用串行总线是应用在微机领域的接口技术，它具备其它总线无法比拟的优点，比如支持热插拔，传输速度快且稳定，低耗能等等。因此，目前 USB 已成为微机与外围设备通信的首选接口。本文我们将重点讨论 USB 接口的通信机制，USB 驱动程序的分层结构以及在数据传输中 USB 驱动程序如何控制 USB 接口读取、写入数据以及如何实现流量控制、差错控制；并以 USB 接口芯片 EZ-USB FX2 开发的设备为例，探讨 USB 设备驱动程序的开发。

2 WDM 驱动程序模型

在 Windows 下的 USB 设备驱动程序必须符合微软定义的 Win32 驱动程序模型（ WDM） 规格。WDM 驱动程序是一种 PnP(即插即用) 驱动程序，它同时还支持电源管理和 WMI 技术，并能在 Windows98、Windows2000 和 Windows XP 间实现源代码级兼容。在 WDM 驱动程序模型中，每个硬件至少包含两个驱动程序，设备驱动程序和总线驱动程序。WDM 驱动程序采用分层结构，可和其他驱动程序相联系，接收建立在其上的驱动程序提供的服务，也可向其他驱动程序发送 IRP 请求。这种分层的好处是把 I/O 分解成一系列可控制的任务，如果每个层次有一个标准的规范，就意味着整个层次可以被替换，而上面的层次感觉不出来，下面的层次隐藏了实现的细节。

驱动程序之间使用 IRP(I/O 请求包)通信。Windows 定义了一套驱动程序使用的 IRP，每个 IRP 请求或执行一次输入 / 输出动作。我们可以把 IRP 理解为 驱动程序提出与设备上的某一端点建立一定方向的数据传输的请求。

3 USB 驱动程序的分层结构

在 USB 通信中采用了分层驱动程序模型，每一层的驱动程序负责处理一部分的 USB 通信任务。图 1 对构成一个 USB 主机的不同软件部分进行了清楚的划分。

(1) 主机控制器驱动程序(HCD) 启动主机控制器与 USB 系统软件之间的通信。它负责跟踪 IRP 的进程，确保不会超过 USB 带宽和微帧的最大值。当管道建立了 IRP 后，HCD 就将它们添加到事务列表中。事务列表描述了总线上需要处理的事务，它包含了事务的各项参数，如数据长度，设备地址，端点号，以及传送和接收数据的存储区域。当一个 IRP 完成后，HCD 会通知请求服务的驱动程序 IRP 已完成。如果 IRP 要从功能设备向设备驱动程序传输数据，数据将被存放在设备驱动程序指定的数据缓冲区中。

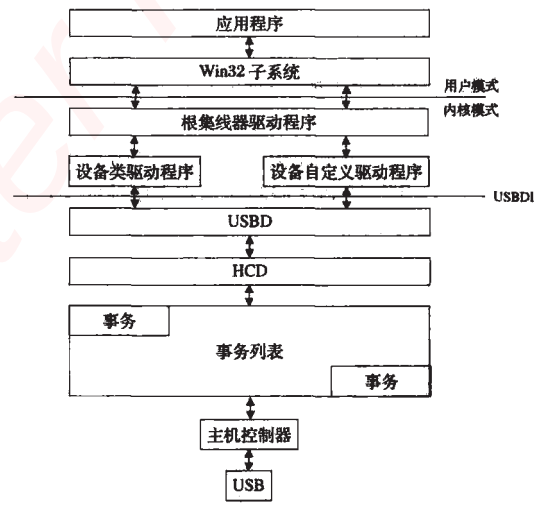

图 1 USB 驱动程序体系结构

(2) USB 总线驱动程序(USBD) 管理总线的电源、检测、USB 事务以及与根集线器驱动程序和 HCD 之间的通信。当设备连接上主机进行配置时，USBD 会判断设备的设置是否可以与总线兼容。USBD 接收来自设备驱动程序的配置请求，该请求描述了要进行的配置内容，包括端点、传输类型、数据大小等等。USBD 可以根据带宽和总线对请求类型的容纳能力接受或拒绝该配置请求。如果它接受了请求，USBD 为请求者建立其要求的传输类型管道。一旦设备配置完成且管道建立后，设备驱动程序就可以发出 IRP，在它和功能设备之间传输数据。

(3) 根集线器驱动程序负责处理根集线器端口的任何下行设备的初始化。

(4) 设备驱动程序位于根集线器驱动程序之上，当应用程序调用 API 函数读 / 写一个 USB 设备时，Windows 会将此调用传递给适当的设备驱动程序，设备驱动程序将此要求转换成 USBD 可以理解的格式，即 URB(USB Request Block)，然后将其传递给 USBD。

(5) USBD 接口(USBDI)。在 Windows 中，系统定义了与

USB 设备驱动程序密切相关的 USB 驱动程序接口(USBDI)。USB 设备驱动程序通过 USBDI 访问其下层驱动程序。USBDI 处理了连接 USB 设备的大多数繁杂的工作。Windows USBDI 由一组接口函数和内部 IOCTLs 构成。

(6) 数据传输管理。数据在 USB 总线上完成传输要靠设备驱动程序、USBD、HCD、主机控制器的协调工作。下面我们以 USB 设备枚举为例，描述数据在 USB 总线上的传输以及 USB 的各层驱动程序在总线枚举过程中执行的任务。驱动程序配置设备的过程如下：

读取设备描述符和设备配置描述符 设备驱动程序调用 USBDI 提供的函数 UsbBuildGetDescriptorRequest()构建一个获得设备描述符请求的 URB，并使用例程 IoBuildDeviceControlRequest () 建立一个含有 IOCTL_INTERNAL_USB_SUBMIT_URB 请求的 IRP，然后调用例程 IoCallDriver( ) 将 URB 提交给 USBD 处理。USBD 将传送来的 IRP 中的数据进行转换后，传递给 HCD。HCD 将 IRP 转换成事务并将其添加到事务列表中。主机控制器访问 HCD，取走事务将其转换成 USB 信息包，使其能在总线拓扑结构中传输。设备收到主机请求后，将传送设备描述符到主机。HCD 通过设置 Irp->IoStatus .Status 为 STATUS_SUCCESS 通知设备驱动程序 IRP 成功完成，数据已经存放到其指定的数据缓冲区。主机读取设备描述符，获得 USB 设备支持的配置数目，然后主机发送获得 USB 设备的一个或多个配置描述符请求，以了解更多的设备信息。配置描述符的读取过程和设备描述符类似。

配置 USB 设备并建立通信管道 任何一个 USB 设备，在使用之前必须进行配置。具体的配置过程如下：①调用 USBD_ParseConfigurationDescriptorEx 对所有可用的配置描述符进行扫描，在其中寻找满足条件的接口描述符，返回第一个匹配值，并将它们存储在 USBD_INTERFACE_LIST_ENTRY 类型的数组，即接口描述符列表，该表中存储了所有的接口信息，包括接口号码，接口替换值，接口句柄及某个接口的管道数目；②将接口描述符列表作为参数，调用函数 USBD_CreateConfigurationRequestEx 构造配置接口的 URB，并发送给 USBD。③USBD 为该接口的所有端点请求资源。如果所有的资源被成功请求，USBD 将设定设备的配置参数，同时返回该设备的接口句柄和管道句柄。至此完成设备接口的配置，建立了通信管道。

上述操作结束后，设备配置完成，设备和主机间就可以进行各种数据传输了。

4 USB 设备驱动程序开发

在本设计中要开发的是基于 EZ-USB FX2 芯片同步传输的驱动程序。需要实现的功能是将数据(由外围设备采集来的数据) 通过 USB 接口用同步传输的方式传送到 PC 机。由此对 FX2 芯片的端点配置如下：端点 0 为默认的控制端点，它是双向端点，实现控制传输。定义端点 2 为同步 IN 端点，实现同步传输的读操作。

4.1 利用 DriverStudio 开发驱动程序

在本设计中使用的开发工具是 DriverStudio。安装完 DriverStudio 后，VC 中新增 DriverStudio 菜单项，选择其中的 Driver Wizard 项，按照向导的提示步骤，将生成驱动程序的框架代码。

4.2 USB 设备驱动程序中的几个主要例程

(1) 驱动程序初始化(DriverEntry) DriverEntry 是驱动程序的入口点，当 I/O 系统将驱动程序装入内核时首先调用该例程。该例程初始化驱动程序范围的数据结构和资源，为驱动程序中的各个例程指定入口点。

(2) 添加设备(AddDevice) PnP 管理器调用 AddDevice 例程来初始化驱动程序控制的设备。调用该例程中的 IoCreateDevice 产生一个设备对象，再调用 IoAttachDeviceToDeviceStack 将其附着到设备堆栈中。

(3) DispatchPnP 例程 PnP 管理器使用 IRP 来指导驱动程序启动、停止和删除设备。OnStartDevice 负责启动设备，OnStopDevice 负责停止设备，OnRemoveDevice 负责删除设备。

(4) 派遣例程 派遣例程处理驱动程序与应用程序之间的通信，派遣例程包括 Create，Close，Read 和 DeviceControl。

Create 例程负责打开目标设备，Close 例程负责关闭目标设备对象，DeviceControl 例程负责处理系统定义的和设备类型特定的 IOCTLs。

下面将着重介绍同步传输的 USB Read 例程的设计。Read 例程负责处理 IRP_MJ_READ 请求，从设备传输数据到主机。当应用程序调用 ReadFile 函数时，系统向设备驱动程序发送一个 IRP_MJ_READ，在该 IRP 的 Readsize 中指明了调用 Read 例程所能读取数据的大小。设备驱动程序接收到该 IRP 后，调用 Read 例程处理该 IRP，分配用以接收数据的存储区。由于 DDK 中没有提供专门构建同步传输 URB 函数，所以我们必须自己构建 URB_ISOCH_TRANSFER URB。同步传输是基于包的，在每个微帧中设备读或写一个数据包。驱动程序可以用一个 URB 发送若干个连续的数据包，所以 URB_ISOCH_TRANSFER URB 的大小由传输的数据包的个数决定。驱动程序在构建 URB_ISOCH_TRANSFER URB 前，应使用宏 GET_ISO_URB_SIZE 来确定将要分配给 URB 的字节数。设备驱动程序构建完 URB 后，向下传递，然后从同步管道中读取设备传送来的数据。

5 结束语

USB 作为全新的计算机外设接口标准，具有其他总线如 PC I、ISA 不可比拟的优点，并在数据传输速度方面发生了质的飞跃。目前，它已成为计算机和外设中最常用的接口。随着社会的进步和科技的发展，USB 必将得到更广泛的应用。

参考文献：

[1] Universal Serial Bus Specification Revision 2.0.http ://www .usb.org,2000 .

[2] 微软公司.Windows 2000 驱动程序开发大全.机械工业出版社,2001 .

[3] 武安河.Windows 2000/XP WDM 设备驱动程序开发.电子工业出版社,2003 .

嵌入式资源免费下载

总线协议：

1. 基于 PCIe 驱动程序的数据传输卡 DMA 传输
2. 基于 PCIe 总线协议的设备驱动开发
3. CANopen 协议介绍
4. 基于 PXI 总线 RS422 数据通信卡 WDM 驱动程序设计
5. FPGA 实现 PCIe 总线 DMA 设计
6. PCI Express 协议实现与验证

VxWorks：

1. 基于 VxWorks 的多任务程序设计
2. 基于 VxWorks 的数据采集存储装置设计
3. Flash 文件系统分析及其在 VxWorks 中的实现
4. VxWorks 多任务编程中的异常研究
5. VxWorks 应用技巧两例
6. 一种基于 VxWorks 的飞行仿真实时管理系统
7. 在 VxWorks 系统中使用 TrueType 字库
8. 基于 FreeType 的 VxWorks 中文显示方案
9. 基于 Tilcon 的 VxWorks 简单动画开发
10. 基于 Tilcon 的某武器显控系统界面设计
11. 基于 Tilcon 的综合导航信息处理装置界面设计

Linux：

1. Linux 程序设计第三版及源代码
2. NAND FLASH 文件系统的设计与实现
3. 多通道串行通信设备的 Linux 驱动程序实现
4. Zsh 开发指南-数组
5. 常用 GDB 命令中文速览
6. 嵌入式 C 进阶之道

Windows CE:

1. Windows CE.NET 下 YAFFS 文件系统 NAND Flash 驱动程序设计
2. Windows CE 的 CAN 总线驱动程序设计
3. 基于 Windows CE.NET 的 ADC 驱动程序实现与应用的研究
4. 基于 Windows CE.NET 平台的串行通信实现
5. 基于 Windows CE.NET 下的 GPRS 模块的研究与开发
6. win2k 下 NTFS 分区用 ntldr 加载进 dos 源代码

PowerPC:

1. Freescale MPC8536 开发板原理图

ARM:

1. 基于 DiskOnChip 2000 的驱动程序设计及应用
2. 基于 ARM 体系的 PC-104 总线设计
3. 基于 ARM 的嵌入式系统中断处理机制研究
4. 设计 ARM 的中断处理
5. 基于 ARM 的数据采集系统并行总线的驱动设计
6. S3C2410 下的 TFT LCD 驱动源码

Hardware:

1. DSP 电源的典型设计
2. 高频脉冲电源设计
3. 电源的综合保护设计
4. 任意波形电源的设计