Windows CE 操作系统下 EZ-USB 设备驱动程序的设计与实现①

EZ-USB Device Driver Based on Windows CE

王　尉　雷跃明　刘　巍　(重庆大学 软件学院 重庆 400044)

摘　要: 本文对基于 WindowsCE 系统下的 EZ-USB 设备驱动程序的设计与实现进行了详细阐述，介绍了 WindowsCE 操作系统中的驱动模型与 USB 通信协议，站在驱动开发者的角度为读者剖析了 USB 设备驱动的安装，USB 设备的识别以及应用程序如何与 USB 外设进行通信等过程，并运用 Micosoft Platform Builder 开发工具实现了 EZ-USB FX2 芯片的设备驱动。

关键词: WindowsCE 嵌入式操作系统 USB 设备驱动 EZ-USB FX2

1　介绍

USB(universal serial bus)的中文名是通用串行总线，它是一种新的总线标准，也是 pc 领域的一种新型接口技术。USB 总线具有接口统一、即插即用、可热插拔和高传输速率等优点。USB 接口凭借其自身的方便性和统一性被一些专业领域中的特殊设备采用作为传输通信的接口，如家庭娱乐数字多媒体设备、医疗卫生数字影像设备、嵌入式设备等。随着有越来越多的带 USB 接口的廉价外设可供使用，有越来越多的嵌入式系统工程师想把 USB 接口技术应用到嵌入式系统的设计中。

Windows CE 是微软公司开发的嵌入式窗口操作系统，但却不是削减的 Windows 版本，它是从整体上为有限资源的平台设计的多线程、完整优先权、多任务的操作系统。

本课题要构建基于 Windows CE 的 USB 设备驱动，选取的 USB 控制芯片是 Cypress 公司生产的基于 8051 CPU 的 EZ-USB FX2 系列芯片。Cypress 半导体公司的 EZ-USB FX2 芯片以其良好的性能和独特的设计在 USB 接口开发领域占有重要地位，然而 Cypress 公司没有为其产品提供 Windows CE 系统下驱动，因此本课题将为 Cypress 公司的 EZ-USB FX2 提供基于 Windows CE 的设备驱动程序。

2　Windows CE 驱动模型

从驱动程序加载的接口方式来区分，Windows CE 设备驱动分为本机设备驱动(Built-In Driver)和可加载驱动(Loadable Driver)。本机设备驱动在系统启动时，在 GWES(Graphic Windows Event SubSystem)进程空间中被加载，它们不是以独立的 DLL 形式存在的。可加载驱动也称为流驱动(Stream Driver)。这些驱动可以在系统启动时间或启动后的任何时候由设备管理器动态加载，通常它们是以用户模式的 DLL 存在。

流驱动程序可通过文件系统的 API 来从设备管理器或应用程序获得命令，如 ReadFile, Writefile, IOControl 等。应用程序可以直接和流接口驱动进行交互，并把流驱动当成文件来操作。为了实现流接口驱动，首先要实现流接口的各个入口点，它们是标准文件 I/O 接口。表 1 是对其中一些接口的介绍，其中 XXX 为设备名前缀，设备管理器在注册表中通过前缀来识别设备。

EZ-USB FX2 的设备前缀为 FX2，所以在本设备驱动 DLL 中，XXX 将由 FX2 替代。

3　USB 通信协议

完整的 USB 协议包括 USB 封包，传输类型，描述

① 收稿时间:2008-09-06

符，设备请求等。

表 1　流驱动接口

流接口函数	描　　述
XXX_Init	负责初始化设备及分配资源，由设备管理器调用或通过使用ActivateDevice函数被调用
XXX_Deinit	卸载设备及回收资源，由资源管理器调用或通过使用DeactivateDevice函数被调用
XXX_Open	打开设备，应用程序调用CreateFile后，经文件系统映射本函数被调用
XXX_Close	关闭设备，应用程序调用CloseHandle后，经文件系统映射本函数被调用
XXX_IOControl	执行与设备相关的I/O操作

USB 规范定义了四种传输类型，分别是控制传输，同步传输，中断传输和批量传输。在与 USB 设备通信时，针对不同设备的特性以及应用的需要选择所需的传输类型。控制传输用于命令或状态相关的操作。同步传输是与时间相关的，此传输需要维持一定的传输速度。中断传输是低频率的，有固定延迟的传输。批量传输用于传输大量数据。

USB 描述符是 USB 设备用来报告其自身状态的一种数据结构。每个描述符开始的第一个字节记录了此描述符的长度，第二个字节表示此描述符的类型。USB 规范定义了若干描述符，其中有四种描述符是每个设备必须具有的，其他描述符是可选的，必须的四种描述符是设备描述符，配置描述符，接口描述符和端点描述符。一个设备只有一个设备描述符，设备描述符描述 USB 设备的基本信息，如制造商 ID(Vendor ID--VID)，产品 ID(Product ID--ID)，设备可用的配置数量。VID 和 PID 可以唯一的标识一个设备。

USB 接口的通信协议中，主机是控制的主体，设备接收主机发送的命令来执行指定的操作。主机与设备间必须以某种特定的命令格式进行通信，这就是 USB 设备请求。

4　USB 设备驱动开发

4.1 USB 设备驱动架构

USB 系统软件由上层的 USB 设备驱动与下层的 USB 函数构成。上层的 USB 设备驱动利用下层的 USB 函数来建立主机到设备的连接，通过这个连接，设备驱动能够对设备进行控制、配置并与设备进行通信。下层的 USB 函数需要执行以下几个任务：

(1) 管理 USB 设备驱动与主机根集线器之间的所有通信。

(2) 在适当的时候载入和卸载 USB 设备驱动。

(3) 按照 USB 协议规定的帧和包格式传输数据。

(4) 能与任何 USB 设备的通用端点建立通信，在此基础上执行通用配置与状态相关的任务。

下层的 USB 函数其自身也由两部分组成，即位于上层的 USB 总线驱动模块(Universal Serial Bus Driver--USBD)与位于下层的主机控制器驱动(Host Controller Driver--HCD)模块。为让 USB 设备驱动能通过 USB 总线驱动访问 USB 设备，USB 总线驱动模块提供了一个接口，叫 USB 总线驱动接口(USBDI)。根据主机控制器驱动模块所提供的功能，USB 总线驱动模块实现了这些接口函数。有了这些接口函数，USB 设备驱动就可以与 USB 外设进行通信了。USB 设备驱动架构见图 1。

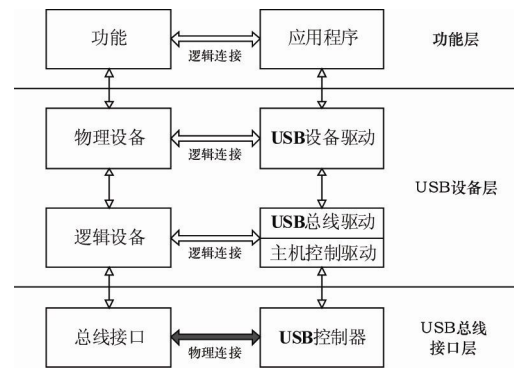

图 1　USB 设备驱动架构

下列是在数据传输时的典型操作流程：

(1) 用户应用程序通过调用 DeviceIOControl 函数向 USB 设备驱动发出请求。

(2) USB 设备驱动使用 USBDI 函数来向 USBD 模块发出请求从而使一个传输得到了初始化。

(3) USBD 模块把请求传给 HCD 模块。

(4) HCD 模块根据总线与 USB 外设特性把请求分成若干事务，并为这些事务进行发送计划。

(5) 主机控制器硬件执行并完成这些事务。

4.2 通用设备驱动与 I/O 控制码

EZ-USB 通用设备驱动是基于 EZ-USB 的计算机外围设备接口的通用设备驱动程序，简称 GPD。应用程序可以通过 I/O 控制来访问 EZ-USB GPD，首先调用文件系统 I/O 函数 CreateFile 来取得访问设备驱动程序的句柄，然后使用文件系统 I/O 函数

DeviceIOControl 函数来提交 I/O 控制码。

GPD 中定义的 IO 操作分四类，第一类是标准设备请求 IOCTL，如获取各种描述符。第二类是数据传输 IOCTL，如批量读写，同步读写等操作。第三类是辅助 IOCTL，如获取设备驱动的版本信息。第四类是 EZ-USB 设备专用 IOCTL，如固件下载。开发一个设备驱动涉及许多的 I/O 控制码，本文以固件下载为例，在后面的章节详细的向读者展示 I/O 控制码对应的功能是如何实现的。

4.3 USB 设备驱动的安装与加载

每当插入一个设备后，设备管理器会获取该设备的 VID 和 PID，并在注册表中查找是否有某个子键与插入设备的 VID 和 PID 相同。如果有，表示设备驱动已经安装了，这个子键下包含了关于该设备驱动的详细信息，如设备驱动的 DLL 名，驱动的前缀，占用的资源等。否则，就表示没有插入设备的驱动程序。此时会弹出一个对话框，提示用户输入设备驱动的 DLL 名。输入正确的驱动 DLL 名后，设备管理器会调用 DLL 中的钩子函数 USBInstallDriver，该函数负责安装设备驱动，它会往注册表中写入相关信息，如果安装成功，则在注册表中将新增一个与设备 VID 和 PID 相应的子键，该子键包含了设备驱动的详细信息。安装完后，设备管理器从新开始在注册表中查找与插入设备的 VID 和 PID 相应的子键。

如果设备管理器在注册表中找到了与设备 VID 和 PID 对应的子键，则会从中读取相关信息用于驱动加载。首先，设备管理器调用注册表中指定的 DLL 中的 USBDeviceAttach 函数，并设备的 VID、PID 等作为参数传给该函数用作判断驱动是否确实可以控制设备。若可以，则设备管理器会以驱动的方式(驱动方式加载指 DLL 被加载后将长驻内存中不能被内存管理器调页出去)来加载这个 DLL。USB 设备插入主机后进行的一系列驱动安装与加载过程见图 2。

4.4 固件下载

EZ-USB FX2 是集成了 8051 芯片与片内 RAM 的单片机，相较传统的单片机采用串口线与主机进行通信而 FX2 通过 USB 总线来通信。FX2 可以通过 USB 总线来下载固件程序至单片机上运行。

在下载固件之前需要了解 EZ-USB FX2 的内存分布情况。EZ-USB FX2 设备的内存分为两个部分，其中一部分被称为片内内存，另一部分被称为片外内存。

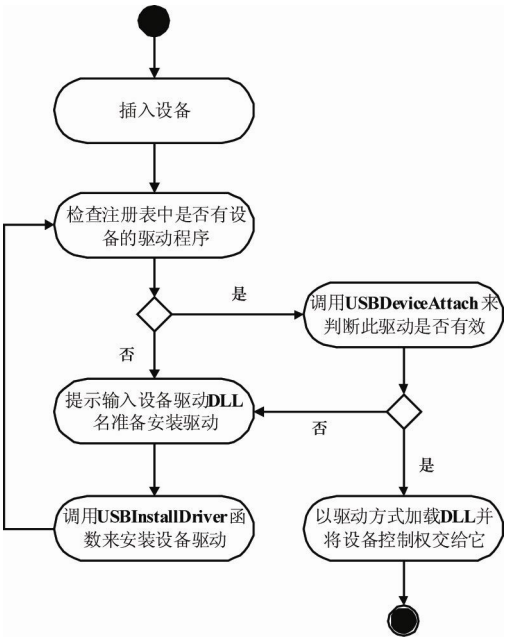

图 2　驱动安装与加载图

片内内存由 3 个 RAM 区组成，即从 0x0000 到 0x1FFF 的主 RAM，0xE000 到 0xE1FF 的临时 RAM 区，0xE200 到 0xFFFF 的寄存器缓冲区。三个片内内存区中间的地址空间 0x2000 到 0xDFFF 是片外内存区。内存分布如图 3 所示。

图 3　USB FX2 内存分布

EZ-USB FX2 设备对于这两块内存的写入命令是不一样的，对于片内内存它的写入命令是 0xA0，对于片外内存它的写入命令是 0xA3。

进行固件下载时，应用程序以 I/O 控制码 IOCTL_EZUSB_ANCHOR_DOWNLOAD 调用函数 DeviceIOControl，并传入指向固件程序的指针。设备驱动使用 USB 标准传输中的控制传输来完成固件下载。因为控制传输一次只能传输少量数据，传输的数据量远小于固件程序本身大小，所以需要把固件程序分成若干较小部分分次进行传输。

为使固件下载过程更方便，在此将定义一个名为 INTEL_HEX_RECORD 的结构体。该结构体对应于一

次控制传输，它描述了一次控制传输中要下载的数据长度、要下载到 EZ-USB FX2 设备上的内存地址、要下载的数据及是否下载完的标志。结构体定义如下：

```
typedef struct _INTEL_HEX_RECORD
{    // Data 数组中有效数据的长度
     BYTE   Length;
     // 要下载到 FX2 内存中的位置
     WORD   Address;
     // 为 1 则表示数据下载完毕
BYTE   Type;
// 实际数据
BYTE   Data[16];
} INTEL_HEX_RECORD, *PINTEL_HEX_RECORD;
```

固件程序是 intel hex 格式的文件，下载前需要把它转成 INTEL_HEX_RECORD 结构体数组。下载时用一个结构体指针指向结构数组中的第一个元素，根据指针所指对象的各字段进行下载操作，操作完毕后移动指针指向数组中下一元素，再根据所指对象的各字段进行下载，重复这一过程直到指针所指对象的 Type 字段不为 1，固件下载流程图见图 4。

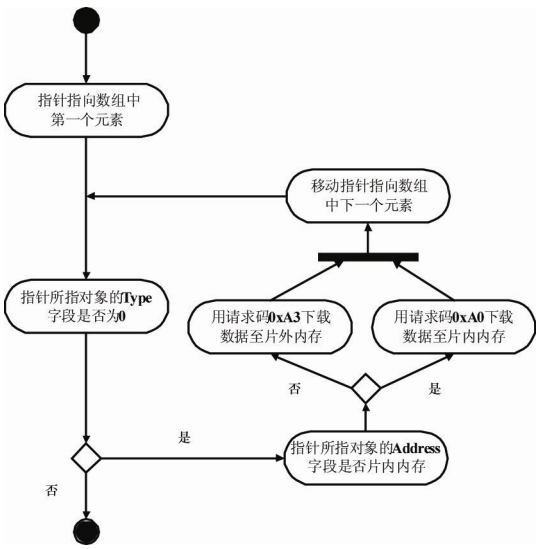

图 4　件下载流程图

5　总结

随着 USB 规范的不断完善，USB 的应用领域也得到了拓展，基于 USB 总线的各种设备如同雨后春笋般涌现出来。另一方面，嵌入式技术也获得了迅速发展，成为当今最热门的技术课题之一。因此越来越多的嵌入式系统工程师想把把 USB 接口技术应用到嵌入式系统的设计中，在此背景下本文提出了基于 Windows CE 的 USB 设备驱动的设计与实现，并选取 Cypress 公司开发的 EZ-USB FX2 芯片为其构建在 Windows CE 下的设备驱动。

参考文献

1 Axelson J. 陈逸译.USB 大全.北京:中国电力出版社, 2001:49－119.
2 钱峰.EZ-USB FX2 单片机原理、编程及应用.北京:北京航空航天大学出版社, 2006:288－307.
3 许永和. EZ-USB FX 系列单片机--USB 外围设备设计与应用.北京:北京航空航天大学出版社, 2002:1－141.
4 傅曦,陈黎,董磊邃,石卫华.Windows CE 嵌入式开发入门—基于 Xscale 架构.北京:人民邮电出版社, 2006:227－263.
5 张冬泉,谭难林,王雪梅,焦凤川.Windows CE 实用开发技术.北京:电子工业出版社, 2006:286－335.