第34卷 第20期 计 算 机 工 程 2008年10月
Vol.34 No.20 Computer Engineering October 2008
·软件技术与数据库· 文章编号：1000—3428(2008)20—0086—02 文献标识码：A 中图分类号：TP311.11

Windows CE 下的串口驱动设计

司浩乐，万 波，田玉敏
(西安电子科技大学外部设备研究所，西安 710071)

摘 要：驱动程序及串口驱动的开发和设计是构建嵌入式系统的重要组成部分。该文介绍了Windows CE下的驱动程序的结构、开发过程及Windows CE的中断处理过程，提出流接口和分层相结合的驱动程序设计方案，给出Windows CE下串口驱动的具体开发过程及串口驱动的中断处理过程。

关键词：嵌入式系统；中断处理；串口驱动

Design of Serial Driver in Windows CE

SI Hao-le, WAN Bo, TIAN Yu-min
(Research Institute of Peripherals, Xidian University, Xi'an 710071)

【Abstract】The design of driver and the development of serial driver are important in constructing embedded system. This paper introduces the development of driver, architecture of driver and the interrupt processing in Windows CE. It presents the project of serial driver, which combines the layered driver with stream driver, and introduces the specific development process of serial driver and the process of interrupt processing.

【Key words】embedded system; interrupt processing; serial driver

嵌入式系统在各个方面得到了广泛的应用，嵌入式操作系统是嵌入式系统的基础，比较常用的有：VxWorks，Palm OS，嵌入式Linux，Windows CE等。微软推出的Windows CE由于具有可靠性较高、内核体积小等特点，因此成为许多嵌入式系统的首选[1]。在嵌入式系统的设计过程中，开发人员需要根据选取的硬件设备设计相应的驱动程序。本文以串口驱动的设计为例，介绍了驱动程序的开发过程。

1 串口驱动开发的相关理论

驱动程序是一个抽象物理设备或虚拟设备的功能软件，它管理设备的操作，将设备的功能导出给应用程序和操作系统，Windows CE的设备驱动程序是通过动态链接库(Dynamic Linkable Library, DLL)实现的。

Windows CE下的驱动程序按导出接口的不同可以分为内建的驱动程序(Builder-in Driver)和可安装的驱动程序(Installable Driver)；按驱动程序的结构可以分为分层驱动程序(Layered Device Driver)和单体驱动程序(Monolithic Device Driver)。

内建的驱动程序又称本地设备驱动程序(Native Device Driver)，它被静态连接到图形窗口事件子系统(GWES)中。微软为每种类型的本地驱动提供了定制的接口，本地设备驱动为所有特定类型的设备提供了一组标准的功能。常见的内建驱动有键盘、触摸屏、音频设备等。

可安装的驱动程序又称作流设备接口程序(Streams Interface Driver)，是由设备管理器动态加载的用户模式的DLL。这类驱动具有一组相同的导出函数——流接口函数，其与文件系统的应用程序接口(Application Programming Interface, API)匹配，由流接口驱动管理的设备向应用程序显示一个文件系统，应用程序通过对文件系统的操作完成对设备的操作，如图1所示。分层的驱动程序将驱动程序的代码分为模型设备驱动(Model Device Driver, MDD)和平台相关驱动(Platform Dependent Driver, PDD)，其结构如图2所示。

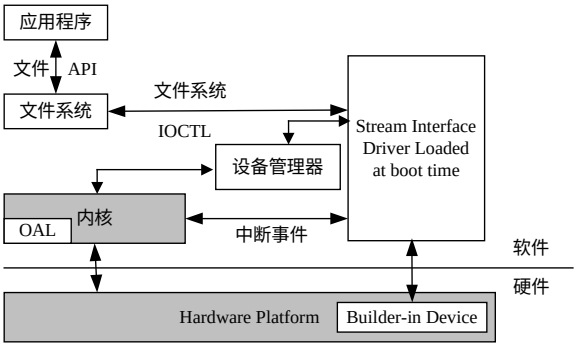

图1 流接口驱动的架构

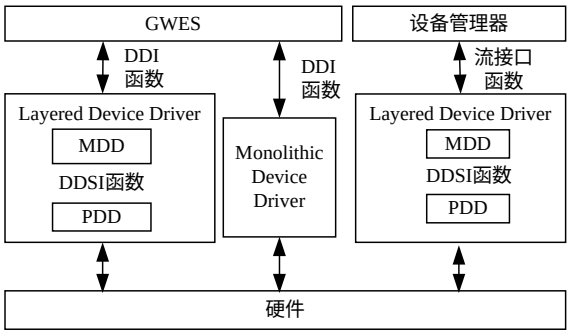

图2 驱动程序结构

MDD层包含给定类型的所有驱动程序公用的代码，主要完成以下任务：(1)链接PDD层并定义它期望调用的函数。(2)导出设备驱动接口(Device Driver Interface, DDI)函数给操

作者简介：司浩乐(1984－)，男，硕士，主研方向：嵌入式技术；万 波，讲师、博士；田玉敏，教授

收稿日期：2007-12-13 **E-mail：**sihaole@yahoo.com.cn

作系统。(3)进行中断处理。

PDD层由与特定的硬件设备相关的代码组成，向MDD层提供设备驱动服务提供者接口(Device Driver Service provider Interface, DDSI)，MDD层调用DDSI函数对设备进行操作。单体驱动程序通常由中断服务线程代码和平台特定的代码组成。所有分层驱动都可以用单体驱动的方式实现[2]。

2 串口驱动的实现

串口以中断的方式与系统进行交互，中断是驱动设计的关键，驱动程序需要实现特定设备的中断响应，对数据进行处理。Windows CE将中断过程分为ISR和IST两部分，ISR运行在内核模式下，主要实现物理中断和逻辑中断的映射，向系统返回中断标识号；IST完成具体的中断处理。中断处理的流程如图 3 所示[3]。

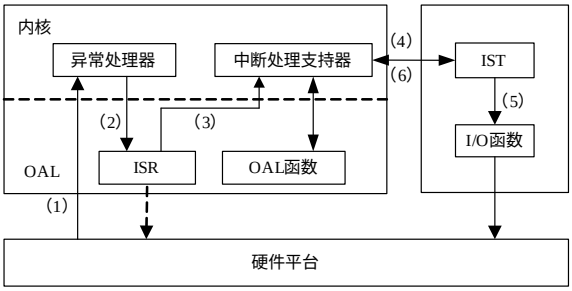

图 3 中断处理过程

(1)当一个硬件中断发生时，它将被发送到内核的异常处理器，由内核处理该异常。

(2)内核的中断处理支持器调用 OAL 函数 OEMInterrupt Disable 通知硬件屏蔽来自该设备的中断，直到处理结束。

(3)内核通过调用中断服务例程(Interrupt Service Routine, ISR)决定如何处理该中断。

(4)内核收到 ISR 的返回值，根据这个返回值知道是什么中断，从而触发与其对应的中断处理线程(Interrupt Service Thread, IST)。IST 是一个常规的 Win32 线程，驱动程序通常在 IST 中产生一个事件，之后等待该事件发生，内核使用调度算法来唤醒等待该事件的 IST。

(5)在 IST 中进行相应的数据处理。

(6)IST 完成之后调用 InterruptDone 函数通知内核，该函数调用内核的 OEMInterruptDone 函数来完成中断所有的处理，通知硬件重新开启该中断。

串口驱动程序的设计采用流接口与分层驱动相结合的方案。MDD层主要由流接口函数和中断处理函数组成。在流接口函数的实现中，关于硬件的操作部分要调用DDSI函数实现，PDD主要由与硬件相关的代码组成。串口驱动程序的执行流程如下[4]：

(1)在驱动程序被成功加载后，设备管理器调用 COM_Init 函数，该函数用于初始化所需要的变量、硬件资源等，主要包括：分配代表硬件的数据结构，调用硬件抽象接口进行硬件初始化，初始化内核事件对象及临界区，确定数据缓冲区。在 COM_Init 函数中创建同步事件对象，通过调用 PDD 层的 GetSerialObject 函数获得硬件信息，硬件的初始化通过调用 PDD 层的初始化函数 HWInit 完成。

(2)进行串口通信时，首先使用 CreateFile 函数打开串口，系统自动调用 COM_Open 函数，创建中断处理线程，为读写串口做准备。有中断发生时，执行中断处理线程，处理相应的操作。在读写串口的过程中设计了 2 个数据缓冲区，即接收缓冲区和发送缓冲区，应用程序调用 ReadFile 函数，系统自动执行 COM_Read，将接收到的数据从接收缓冲区中读出。当应用程序调用 WriteFile 函数时，系统自动调用 COM_Write，将要发送的数据写到发送数据的缓冲区中。之后调用 PDD 层提供的函数将数据发送出去，具体的硬件和软件之间的协调工作在中断处理过程中执行。

(3)当串口操作结束后，应用程序调用 CloseHandle 关闭串口，系统自动执行 COM_Close，实现与 COM_Open 相反的操作，由 COM_Close 调用 PDD 层的 HWClose 函数实现具体的硬件操作。

(4)驱动程序被卸载时，通过 COM_Deinit 函数执行与 COM_Init 相反的操作，首先确定停止中断服务程序，通过 COM_Close 关闭已经打开的串口，释放驱动中所使用的资源等操作，对硬件的操作在 PDD 层的 HWDeinit 函数中完成。

在设计的过程中需要用到重要的结构体__HW_VTBL 和__HW_INDEP_INFO，其中，__HW_INDEP_INFO 是对串口设备的抽象；__HW_VTBL 由一组函数指针组成，MDD 层的函数通过该结构体实现对 PDD 层函数的调用。对硬件的具体操作通过 PDD 层的函数实现，这些函数中只有 GetSerialObject 是 MDD 可以直接调用的，其他函数都由结构体__HW_VTBL 所定义的函数指针指定，在 PDD 和 MDD 层之间利用结构体变量实现层次之间的函数调用。

串口驱动程序及核心的中断处理过程如图 4 所示。

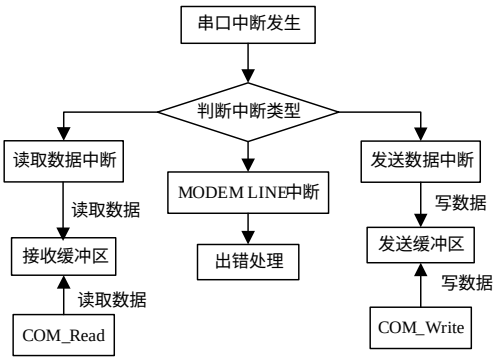

图 4 串口中断处理流程

(1)在 COM_Init 中，使用 CreateEvent 函数创建发送数据和接收数据的同步事件对象，默认为自动重启，初始为无信号。当有中断发生或者串口打开时，调用 InterruptInitialize 函数将中断 ID 和事件对象相关联，创建中断处理线程，进行具体的中断处理。

(2)当接收到的中断类型是 INTR_RX 时，将接收到的数据从串口设备接收缓冲区中读取到与此设备对应的抽象设备缓冲区中。当执行 COM_Read 函数时，直接从抽象设备的接收缓冲区中读取数据。当接收到的中断类型是 INTR_TX 时，直接将要发送的数据发送到串口设备的发送数据缓冲区中。当执行 COM_Write 函数时，将要发送的数据写到定义的发送缓冲区中，完成操作之后通知硬件发送数据。

因为在系统启动时流接口驱动是由设备管理器加载的，设备管理器从注册表读取驱动的相关信息，所以在驱动程序的设计完成后还需要在注册表中添加信息，使系统能够在启动时自动加载驱动程序。

3 结束语

本文介绍了 Windows CE 下串口驱动程序的开发，采用

(下转第 90 页)