计 算 机 自 动 测 量 与 控 制 . 2000.8(1)
Computer Automated Measurement & Control

文章编号:1007 - 0257(2000) - 01 - 0059 - 03

在 Windows NT 下开发高速 DMA 数据采集程序

刘俊　柳春图
(中国科学院力学所,北京　100086)

摘要:介绍了在 WinNT 下使用 DMA 通道的方法。给出了用 Windows NT DDK 开发 DMA 数据采集程序具体细节。

关键词:DMA;DDK;数据采集
中图分类号:TP316;TP274　　**文献标识码**:A

Developing Driving Program for DMA Data Acquiring on Windows NT

LIU Jun　　LIU Chun-tu
(Institute of Mechanics,China Academy of Science,Beijing　100086)

Abstract: A method for access to the DMA channels in application based on Windows NT4.0 operation system is introduced. And the implementing process in Windows NT4.0 DDK is especially described.

Key words: DMA;DDK;data acquisition

1　前言

作为一个稳健而强大的商业操作系统,Windows NT 4.0 获得了越来越广泛的注意。它的使用界面与 Win95 相似,而遵循 Win32 API 标准的 Win95 程序可以在 WinNT 下运行。因此,对于具有 Win95 使用与编程基础的用户来说,它十分简单易学。同时,WinNT 又具有 Win95 所没有的安全性、稳健性以及强大的网络功能。由于 WinNT 在 Windows 家族中的领先地位,自然而然地成为许多原先基于 Win95/98 的数据采集程序的迁移对象。然而,基于 Win NT 的数据采集程序,由于使用了系统的硬件资源,在迁移到 WinNT 下时,首先要解决的就是在 WinNT 下的硬件驱动问题。在本文中,介绍了一种用 Windows NT DDK 进行 DMA 数据采集程序开发的方法,以供同行参考。

2　硬件

我们使用的是中泰计算机技术研究所的 PC-6340 高速模入接口卡,此卡适用于符合 EISA (AT) 总线标准的 286/386 /486 系列原装机及其兼容机,具有高速高精度的特点,其 A/D 转换启动可以选用程序或定时器触发,转换结果可以用 DMA 方式自动读取,采样频率可达到 200kHz。

在进行 AD 数据采集时,通常的操作顺序是首先通过程序或采集板上的定时器来启动 AD 采集。之后,AD 芯片切换通道和建立采样电平,这需要一定的时间。在这段时间内,程序通过查询端口状态来获取转换状态。当 AD 转换完成后,程序读取转换的结果。在这一过程中,程序在启动 AD 采集或启动定时器后,就只能被动的等待 AD 转换完成,浪费了大量的 CPU 时间。

采集板也可以通过中断来通知 CPU 读取 AD 转换的结果。但是在 Win95 和 WinNT 这样的多线程多任务系统中,频繁的中断会导致大量的任务切换,因此同样要浪费大量的 CPU 时间。同时,使用中断来进行 AD 采集,需要编写中断例程,并通过某种缓冲区机制来向主程序传送数据,以免干扰主程序的正常运行。

而选用 DMA 方式来自动采集数据时,程序在设置好定时器和 DMA 控制器后,定时器将定时进行采集,采集到的数据通过 DMA 控制器传送到内存中。当采集到规定的次数后,通过中断或某种查询机制来通知程序处理数据。这使 CPU 在数据采集和传输操作中参与最少,从而腾出 CPU 时间来进行数据的处理和显示工作。

3　开发 WinNT 驱动程序

Windows NT 操作系统设计的主要目标之一,是使 WinNT 能运行在不同的硬件平台上。这就要求操作系统中与设备打交道的部分与具体的硬件无关。因此,WinNT 提供了一个普遍的设备模型以适应绝大部分的硬件设备驱动的需要。在操作系统提供给应用

收稿日期:1999-10-27
作者简介:刘俊(1974 -),男,广西武鸣人,中科院力学所硕士研究生。

© 1994-2009 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

程序使用的 Win32 API 函数集中，并不包括与具体的硬件如串口、并口以及 DMA 等操作相关的 API 函数。相反的，应用程序可以使用 Win32 API 函数 CreateFile 来打开一个具体设备的驱动程序，用 ReadFile 和 WriteFile 来从设备中读取和输出数据，就象操作一个普通的文件一样。

从应用程序的观点来看，这使得 WinNT 可以用一致的方法来处理应用程序的数据流。当应用程序通过 CreateFile 试图打开一个数据流以便读取或输出数据时（不管这个数据流是磁盘上真实存在的文件还是一个采集卡的驱动程序，或是与其他计算机通讯的管道），NT 的内核将分析数据流的名称，通过符号连接机制决定把请求提交给哪一个设备驱动程序。当找到合适的设备驱动程序后，NT 向应用程序提供一个标志此设备驱动程序的句柄。当应用程序通过 ReadFile/ WriteFile 提交一个数据 I/ O 请求时，NT 将根据句柄调用正确的设备驱动程序来完成数据的输入和输出。

Windows NT 的驱动程序包括用户模式驱动程序，内核模式驱动程序和虚拟设备驱动程序。然而，只有内核模式驱动程序才能实现软件和硬件的直接交流。这是因为内核模式是 NT 的特权执行模式，在此模式下，程序可以访问系统硬件资源。而应用程序和用户模式驱动程序是运行在用户模式下的，被 NT 的特权保护限制不能直接访问硬件资源，只能通过内核模式驱动程序来访问硬件。在本文中，主要介绍的就是开发内核模式驱动程序的方法。

Windows NT 内核模式驱动程序是 NT 操作系统的一个部分。他运行在 Windows NT 的内核模式中，通过 NT 的一组很好定义的函数来与硬件打交道。NT 内核把对设备的读写请求从用户模式的应用程序中转换到相应的驱动程序，按照合适的顺序调用驱动程序提供的各个例程。

因此，开发 NT 设备驱动程序，就是编写一系列精心定义的例程，NT 将在系统初始化，退出以及应用程序提出 I/ O 请求时，按照合适的顺序调用这些例程。而在这些例程中，可以使用 NT 提供的服务来访问硬件设备，在设备和应用程序之间传送数据。

在我们的设备驱动程序中，主要应用了如下几个例程：DriverEntry 例程，它在操作系统引导时被调用，执行一系列的初始化操作，并建立指向其他例程的指针；数据传输例程（StartIO），在数据传输开始时由 NT 内核调用，他申请设备运行所需的 DMA 资源，并设置设备运行；资源同步回调例程（AdapterControl），当申请到所需的 DMA 资源后，NT 内核调用此资源回调例程，驱动程序可以在此例程中启动硬件的 DMA 数据传送；Unload 例程，在系统被关闭或驱动程序被卸出时调用，完成资源的释放等任务。

除了这些例程之外，数据结构在 NT 驱动程序的编写中，也是十分重要的一环。Windows NT 使用对象技术来管理所有的数据。这些对象与严格的 OOP 定义的对象略有不同，可以把这些对象理解为使用特定访问函数进行操作的数据结构。在驱动程序中，可以接触到对象中的一些域，但其他操作则必须通过访问函数进行。数据结构贯穿在驱动程序和系统的相互作用中，维护各种设备状态信息。

在我们的设备驱动程序中，使用了如下几个对象：Driver 对象，他由 NT 内核创建，在 DriverEntry 中填写指向其他例程的指针；Device 对象，他在 DeviceEntry 例程中创建，在驱动程序的各个例程之间保存和传送一些设备的特征和状态信息；I/ O 请求包（IRP），他代表了一次具体的 I/ O 操作。并贯穿在一次 I/ O 操作的整个过程，以保存和跟踪本次 I/ O 操作的状态和进展；Adapter 对象，他代表系统中的 DMA 通道，在设备驱动程序使用 DMA 之前，必须首先申请对应 DMA 通道的 Adapter 对象的拥有权，当申请到 Adapter 对象的拥有权后才能进行 DMA 操作。在了解了 NT 驱动程序的一般结构后，我们可以讨论实际设备驱动程序的开发。进行 NT 设备驱动程序的开发，需要安装 Windows NT 设备驱动程序工具包（Windows NT 4.0 DDK），此工具可以从 MSDN 光盘中获得。

因为驱动程序运行在系统的内核模式下，驱动程序的错误会导致操作系统的崩溃。因此，驱动程序的开发要求仔细的考虑，通常的策略是使用 DDK 中附带的样板代码，可以以此为骨架或抽取其中的有用部分来进行编程。当完成了驱动程序的编程之后，可以按照 DDK 的文档来进行驱动程序的编译、安装工作。

4　使用 WinNT 驱动程序

当完成了驱动程序的开发后，就可以在应用程序中，使用 Win32 API 函数来调用驱动程序的服务了。一般是用 CreateFile 来打开句柄，用 ReadFile 来从句柄中读取数据，用 CloseFile 来关闭句柄。也可以使用某种后门机制如 DeviceIOControl 来与设备驱动程序直接打交道。

5　PC6340 驱动程序代码及说明

在驱动程序的各个例程中，主要的就是对 Adapter 对象进行一系列操作，这包括：

①按照 PC6340 的硬件特性，找到一个正确的 Adapter 对象，使之与 PC6340 关联。这主要是按照板卡的 DMA 通道，以及 ISA 总线的特性确定一系列参数，来创建一个 Adapter 对象。

© 1994-2009 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

②开始数据传输之前,请求 Adapter 对象的拥有权。

③在传输的开始装载 Adapter 对象所需的资源。通过对 PC6340 的端口操作启动 DMA 数据传送。

④数据传输完成后,刷新 Adapter 对象的缓冲。

⑤释放 Adapter 对象的拥有权。完成本次数据传输。

使用 Windows NT DDK 编写 PC6340 采集卡的 DMA 驱动程序就是要编制完成如下 5 个例程。由于篇幅的限制,本文中省去了各例程的源程序代码,感兴趣者请和本文作者联系。

①DriverEntry 例程,在本例程中,除了找到驱动程序所使用的资源以及申请这些资源创建设备外,还需要根据硬件信息(在这里是 PC6340),构造 DEVICE-DESCRIPITION 结构,并调用 HalGetAdapter 找到与 DMA 相对应的 Adapter 对象。

②MyOpenClose 例程,对应于 Win32 中用于操纵本驱动程序的文件打开关闭操作。在本例程中,仅简单的返回成功以完成此 I/O 请求。

③MyReadWrite 例程,对应于 Win32 中用于读取数据的文件读操作。在本函数中,开始一个 I/O 请求包。

④MyStartIO 例程。该例程请求 Adapter 对象的拥有权,并请求足够数量的映射寄存器,确定 DMA 传输使用的偏移地址。DMA 传输的实际过程将由 Adapter Control 例程来完成。

⑤MyAdapterControl 例程。本例程中,调用 KeFlushBuffer 把数据从 CPU 的 Cache 清到物理内存。为了简单起见,在本例程中包括了 DMA 数据采集的启动,传送的整个过程,在实际应用时,可以根据实际需要,启动 DMA 传送后,挂起 I/O 请求,直到 DMA 传送完成后,使用中断方式来结束 DMA 的传送并取得数据。 □

知识窗

高速串行总线——IEEE1394

"FireWire","IEEE—1394","IEC 1883"……都是指一种高速串行总线,实际上,这是在 PC 机和消费类电器设备间进行数据传输的一种新的标准。由 Apple 计算机公司命名的"FireWire"源于低价消费类产品间连接应用的需要,它们有大量的视听数据需要记录、编辑、存储和传输。总线的性能,灵活性和易于使用性通过外部 PC 机和 C 尺寸 VXI 主机箱间的 I/O 互连得以实现。

之所以能实现 IEEE-1394 的低价格,部分是通过采用经简化电缆设计的串行数据传输。IEEE—1394 的电缆介质可允许在一个总线段上进行多达 16 次的物理连接(电缆跃距),而每一段的长度可达 4.5m,这就使 IEEE—1394 系统的电缆总距离达到 72m。数据在电缆中的一组双绞线上发送,而另一组双绞线则作为时钟使用。在数据线不使用时实现时钟跃变,因此可采用简单的异或门作时钟恢复。

IEEE—1394 的低价格和易于使用也来自其简化的电路。它的发送器和接收器都可采用管理寻址、初始化、仲裁和协议的标准芯片组。IEEE—1394 的即插即用特性也由该芯片组实现。例如,设备在总线上的结点地址可在开机时予以分配。在 IEEE—1394 总线上的数据传输可为异步或等时方式,两种方式均可用于同一总线。等时数据传输在规定时间间隔内对多个"通道"发布大量可变的数据流而不需经过应答。异步数据传输用一种"公正仲裁"协议以确保各 IEEE—1394 设备能同样地访问总线。

(上接第 55 页)

```
        5 ms - 10 ms:  →300 Baud
        11 ms - 15 ms:  →150 Baud
        16 ms - 22 ms:  →110 Baud
        23 ms - 32 ms:  →75 Baud
        33 ms - 49 ms:  →50 Baud
            ELSE:  → Timed out ; reset
    END CASE;
ELSIF Byte > = 0xF1 THEN
    →19200 Baud
ELSE
    CASE Byte IN
        0x0D:  →9600 Baud
        0xE6:  →4800 Baud
        0x78:  →2400 Baud
    0xE0, 0xF0:  →1800 Baud
        0x80:  →1200 Baud
        ELSE:  →Line noise; reset
    END CASE
END IF
```

参考文献

[1] 赵依军等. 单片微机接口技术 [M]. 北京: 人民邮电出版社, 1989.

[2] 刘利. 软硬件技术参考大全 [M]. 北京: 学苑出版社, 1993.

[3] 张世一. 数字信号处理 [M]. 北京: 北京工业学院出版社, 1987.

© 1994-2009 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net