Product Applied

摘要：采用Vireo Software公司出品的VxD开发软件包VtoolsD，开发出了Windows98环境下的数据采集卡虚拟设备驱动程序。其中VxD部分完成对物理设备的直接控制和通信，DLL部分完成设备驱动（Ring0）与应用程序（Ring3）之间的接口，应用程序只需调用DLL就可以完成对物理设备的控制。应用程序通过调用Windows提供的QueryPerformanceFrequency和QueryPerformanceCounter函数，实现对采样的精确定时，其周期最小可达0.1ms。

关键词：Windows98 虚拟设备驱动程序VxD 中断 定时器

PS-2116型号数据采集接口板卡的驱动模块开发

梅 辉 （西安西北工业大学民航学院 710072）

1 VxD的功能结构与运行原理

1.1 VxD功能结构

在Win16位的版本中，整个加载设备驱动程序的过程是：系统在启动时，Windows寻找System.ini文件，读出此文件中[386nh]节的内容，此节的内容是device = xxxx驱动程序，这样设备驱动程序在系统初始化时被加载。VxD起始于Windows 386增强模式，它是Win95为解决这一问题而加入的。

虚拟机管理器和VxD的集合是构成Win98系统的核心，在一般情况下，VxD的作用是用来支持硬件设备管理，是虚拟化的某一具体硬件设备的驱动程序，其功能是向应用程序提供与硬件接口环境，用以同步和协调各虚拟机对设备的访问，并可以向其他的VxD通过服务，还能为实模式软件提供应用程序编程接口。而VxD的作用不仅仅限于此，它还可作为设备驱动程序而不是虚拟化设备，还有些VxD与硬件并没有什么关系，它仅向其他的VxD或应用程序提供服务。这样软件在虚拟机上运行时并不认为虚设备和实设备有什么不同，在软件一级来看，它们是完全一样的。

一个标准的VxD由5部分组成，这五部分分别存放在5个不同的段中：

- VxD-CODE（必须段）

保护模式代码段，包括虚拟设备的设备控制过程（DCP）、回调函数、服务例程和API函数。

- VxD-DATA（必须段）

保护模式数据段，包括虚拟设备的设备描述块（DDB），由各服务例程地址构成的服务表以及虚拟设备使用的全部数据。

- VxD-ICODE（可选段）

保护模式初始化段，通常包括仅用于虚拟设备初始化阶段的过程和服务。

- VxD-IDATA（可选段）

保护模式初始化数据段，通常包括虚拟设备在初始化过程和服务中使用数据。

- VxD-REAL-INIT（可选段）

实模式初始化段，包括虚拟设备初始化过程和初始化用到的数据。

1.2 运行原理

每个VxD的初始化过程是不同的，而且模式不同过程也不同。总的来说，过程大致是：Windows在初始化时，每一个VxD按照设备描述块中定义的初始化顺序进行初始化工作，一般来说，应当按照优先级顺序或装入顺序初始化VxD。VxD的INITVxD的装载与初始化是由VMM来完成的，VMM通过调用VxD中定义的实模式与保护模式初始化过程来完成VxD的初始化。在初始化阶段的任何时刻，VxD都可以通过设置标志位告诉VMM终止VxD的装入，VMM完成每个VxD的初始化后，才开始对Windows其他模块进行装入执行。

VxD被正确初始化载入后，一个运行在虚拟机上要和系统软硬件打交道的程序对设备的访问过程是：

(1) 此程序发出访问请求，这个请求经过各种渠道最后到达虚拟机管理器；

(2) VMM将请求解释后传给该设备的VxD；

(3) VxD对请求进行理解，协调处理当前转台后将请求发送给物理设备；

(4) 物理设备返回数据由VxD传给VMM，VMM再传递给虚拟机上的程序。

2 数据采集虚拟设备驱动程序(VxD)的开发

2.1 开发工具简介

VxD的开发，通常只能使用DDK，用32位汇编语言开发，但这需要对Win9x的内核结构相当了解，另一方法是借助Vireo Software公司出品的VxD开发软件包VtoolsD。

VtoolsD开发包包括1个可视化编程的VxD代码生成器QucikVxD、ANSI C的运行库、VMM/VxD服务库、VxD的C++类库、使用工具以及

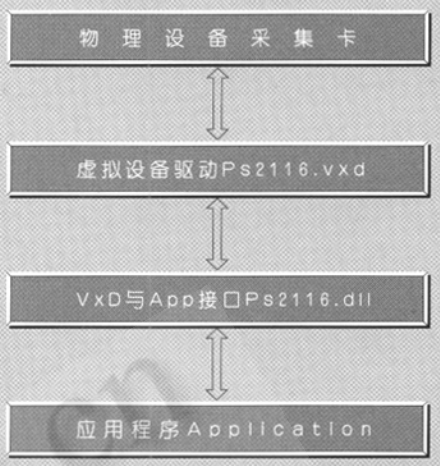

图1 设备驱动解决方案流程图

大量实例和文档。

VtoolsD的类库提供了VxD程序框架，绝大部分的VMM和VxDs的服务都可以通过引用类成员函数来实现，用户只需调用这些已彻底测试的可重用类，即可实现大多数底层功能，这就减少了错误来源，大量节省VxD的开发调试时间。

2.2 数据采集卡接口板卡结构

本文选择北京众人精密测控技术公司的PS－2116型号的数据采集接口板卡，它采样IBM－PC总线标准所设计的12位A/D卡，主要包括一片16路多路开关、一片A/D转换器、采样保持器、24路数字量并行I/O接口。

模拟信号经多路开关送入采样/保持器(S/H)，再送入A/D转换器进行A/D转换，可单端输入。数字量I/O口由并行芯片8255构成，通过设定8255的控制字，可方便的改变数字量输入/输出方式和路数。

该数据采集接口板卡的主要技术指标如下：

- A/D通道数：单端16路
- A/D转换器位数：12位
- 输入信号电压范围：0V～±5V
- 转换时间：≤25 μs
- 总误差：≤0.1%FSR
- 输出码制：偏移二进制码
- I/O通道数：24路
- 输入/输出电平：TTL电平
- 输入阻抗：≥10MΩ

由于IBM－PC机中使用A_0～$A_9$10根地址线做I/O断口寻址，所以PS－2116接口板也使用这10根线做I/O译码，其中A_0～A_2的3根线用作译码，其余7根线作为板选译码，板选地址称为I/O端口基地址，采取的是开关可选译码电路出厂时将I/O端口基地址设为0100H，在本系统中，8255以方式2工作。

2.3 数据采集设备驱动

本数据采集系统通过采集卡上的A/D芯片将模拟信号数字化，每次采样结束后直接触发中断。为了实现和控制以上过程，驱动程序需解决以下问题：

(1) I/O端口读写。采集卡的命令字和状态字的读写多通过I/O端口操作实现。虽然I/O操作可在Ring3级代码实现，但是优先级太低，延时大，不符合实时要求。

(2) 访问物理地址。保护模式下，程序运行在线性物理地址空间，内存采用平板模式(FlatMode)，不能直接访问物理地址。这就需要设备驱动程序实现物理地址到线性地址的转换

(3) 中断。中断信号接收和中断服务程序都属于Win98系统核心级操作，在Ring3级代码中无法实现，必须由VxD处理。这也是设备驱动程序要解决的核心问题。

为了解决以上问题，本文提出的解决方案由虚拟设备驱动程序Ps2116.vxd和Ring3间接口程序Ps2116.dll两部分构成，其流程如图1所示。

其中VxD部分完成对物理设备的直接控制和通信，DLL部分完成设备驱动（Ring0）与应用程序（Ring3）间的接口。这样，应用程序就感觉不到Ring0级的设备驱动存在，只需调用DLL就可以完成对物理设备的控制。

2.3.1 VxD的编制

(1) I/O操作。I/O读写实现较简单，可调用VtoolsD的VMM的服务函数-outp，outpb，-inp和-inpb。

(2) 访问物理地址。Ring3代码不能实现访问物理地址访问，这是由于它只能读写线性地址。VtoolsD提供VMM服务函数MapPhysToLinear可实现物理地址到线性地址转换。函数原形为PVOID MapPhysToLinear (CONST VOID *PhysAddr, DWORD nBytes, WORD Flags)，其中参数PhysAddr、nBytes分别为物理地址和此物理内存大小，Flags必须置为0，函数返回值为线性地址。

(3) 中断处理。VtoolsD提供类VhardwareInt来实现对某个IRQ端口的虚拟化，并处理该IRQ的中断服务。使用VhardwareInt首先应该调用构造函数VhardwareInt创建实例，在重载OnVirtualInt成员函数，实现中断服务；最后调用hook成员函数，将IRQ虚拟化并与VhardwareInt类OnHardware相挂接。中断信号可由应用程序传入，且应用程序事先将窗口句柄和自定义的待检测消息值由DLL传入设备驱

参考文献

1 杨强，李堂秋，Win9x虚拟设备驱动程序编程指南 [M]，清华大学出版社，1999。

2 David J. Kruglinski 等，Visual C++ 6.0技术内幕 [M]，希望电子出版社。

3 马明建，周长城，数据采集与处理[M]，西安交通大学出版社，1998。

4 雷霖，微机自动监测 [M]，电子科技大学出版社，1998。

5 刘乐善，叶永坚，叶济忠，微型计算机接口技术原理与应用 [M]，华中理，工大学出版社，1996。

6 Jeffrey Richter，王建华等译，Windows核心编程 [M]，机械工业出版社，2000。

动程序，并建立此消息与自定义的处理函数相映射。中断发生后，中断服务函数OnVirtualInt中仅向用户程序窗口发送指定消息，用户程序收到驱动程序发出的消息后，将自动调用处理函数。对应用程序来说，对中断的处理变成对指定消息的处理。

(4) VxD与Win32的通信机制。VxD与Win32的通信是通过一个特殊的消息：W32-DEVICEIOCONTROL来实现的。VMM代替DeviceControl函数的应用向VxD发送此消息，消息参数可确定VxD消息响应函数、输入输出缓冲区指针及缓冲区大小，并绑定在DIOCPARAMETERS结构中，通过这一接口，不仅仅可以读写设备，而且还能在应用程序和VxD之间互传指针，从而达到特殊应用目的，完成以上操作由类VDevice的成员函数OnW32DeviceIoContaol完成。

2.3.2 DLL文件的编写

在此解决方案中，DLL仅起到应用程序与VxD间的桥梁作用，使用户在开发应用程序时感觉不到底层的设备驱动程序存在，只要像使用SDK一样调用DLL。

调用VxD首先由CreatFile函数打开设备驱动，得到设备句柄，在通过DeviceIoControl函数与VxD中的OnWin32DeviceIoControl通信。其原型如下：

```
BOOL DeviceIoControl(
    HANDLE hDevice, // 设备句柄
    DWORD dwIoControlCode, //VxD 服务操作码
    LPVOID lpInBuffer,
    DWORD nInBufferSize, //传入参数指针和大小
    LPVOID lpOutBuffer,
    DWORD nOutBufferSize, // 返回指针和大小
    LPDWORD lpByteRetured, // 返回字节数
    LPOVERLAPPED lpOverlapped // 用于异步方式的设备驱动程序，通常置为NULL VxD所需的中断号、指定消息值、用户窗口句柄、I/O读写参数、需转换到线性空间的物理地址和设备开关、中断的指令及返回应用程序I/O读写结果，转换后的线性地址等均由DeviceControl双向传送。
```

DLL提供给应用程序调用的函数如下：

```
--declspec(dllimport) HANDLE --stdcall OpenChannel(DWORD Channel,DWORD Flags); // 打开通道
--declspec(dllimport) BOOL --stdcall CloseChannel(HANDLE Channel); // 关闭通道
--declspec(dllimport) LONG --stdcall ReadChannel(HANDLE Channel, PVOID Buf, DWORD Length, LPOVERLAPPED Overlap); // 启动指定的通道执行AD转换操作，将结果存入指定的缓冲区。
--declspec(dllimport) LONG --stdcall ReadMultiChannel(HANDLE Channel, PIOControlHeadParam, PVOID Buf, DWORD Length, LPOVERLAPPED Overlap); // 按顺序启动多个通道执行AD转换，结果按顺序存入指定的缓冲区。
--declspec(dllimport) LONG --stdcall GetAsyncResult(HANDLE Channel, LPOVERLAPPED Overlap, BOOL bWait); //返回指定的OVERLAPPED结构最近的一次异步IO操作的结果。
--declspec(dllimport) LONG --stdcall MultiChannelBufLen(PIOControlHead Param); // 返回Param指定的一组连续的AD转换操作的总次数。
--declspec(dllimport) BYTE --stdcall PortIn (HANDLE Channel, WORD PortOffset); // 读数据采集卡上一个指定的IO端口。
--declspec(dllimport) void --stdcall PortOut (HANDLE Channel, WORD PortOffset, BYTE Value); // 写数据采集卡上一个指定的IO端口。
```

3 结论

本文通过深入分析Windows98系统内核，借助Vireo Software公司出品的VxD开发软件包VtoolsD，利用系统提供的时间控制函数QueryPerformanceFrequency和QueryPerformanceCounter，编制了数据采集卡PS2116的驱动程序和与之相关的动态连接库文件。