USB 驱动的制作过程与体会

作者: hoguowi

联系 QQ: 175047360

电子 DIY 于 2007 年 6 月

在写 USB 的驱动文件的时候,首先要安装 WIN2KDDK.EXE,然后再安装 DriverStudio2.6

通过"开始"->"程序"->"NuMega DriverStudio"->"Tools"->"DDK Build Settings (SetDDKGo)"进行环境设置并由此进入 VC++ 在 VC 的"TOOL"->"options"->"Directories"添加头文件目录 C:\PROGRAM FILES\NUMEGA\DRIVERSTUDIO\DRIVERWORKS\INCLUDE 不添加的话向导生成的文件#include <devintf.h> // DriverWorks 将找不到头文件而出错 安装完 DDK 和 DriverStudio 后,我们接下来是要编译库 在你利用 DriverWorks 开始工作之前,你必须编译需要的库文件。你可以在 Microsoft Visual Studio 环境中,或者用命令行方式编译库文件。 下面介绍怎样在 VC 环境中编译库。 1 从"Start"->...->Tools->DDK Build Settings 2 单击"Launch Program"启动 VC++; 3.选择菜单 File|Open Workspace。打开位于 DriverStudio\DriverWorks\Source\vdwlibs.dsw 的工作空间文件。 4选择菜单 Build|Batch Build(编译|批构件),在弹出的对话框中只选 NdisVdm-Win32 NDIS VDM Checked. NdisVdm-Win32 NDIS VDM Free. VdwLibs-Win32 WDM Checked. VdwLibs-Win32 WDM Free. 这四个库,然后单击 Build 编译。应该就没有问题了,试试看吧! 5.点击 Build 编译你选择的库。

三、编译 vdwlibs.dsw 完毕没有错误后。我们通过单击"Launch Program"再次启动 VC++ 利用 DriverStudio 向导生成 USB 驱动

dataBuffer	🔽 🙀] 🖄 🖄 🏣 💁 端 🚟
	·····································
T/WDM Device I	Driver Vizard (DriverVorks) - Step 1
DriverWorke	Welcome to
DING WOINS	You are just a few mouse clicks away from creating a device driver for Windows NT or WDM.
	Press F1 to get detailed help on any visible dialog Note: For Windows 9x, short project names (8 characters and less) work best.
	Choose a Project Name and Location:
line	Project
	ISP1581
ð T	Location



图1 保存工程路径

接着下一步 Next,如图

	® + × E	84 ! B 4
I∕WDE Device I)river Vizard (Drive:	rWorks) - Step 2 (
DriverWorks	Choose one of the followi project type and target o	ng as your driver perating system:
	⊂ Wi <u>n</u> dows NT 4.0	Select this option if your driver is for NT 4.0 and you want to use the DriverWorks C++ framework
	¥DM (Windows Driver (● Model) Windows 98/Windows 2000	Select this option if your driver is for WDM and you want to use the DriverWorks C++ framework
Computero		

在此我们默认是 WDM,直接 Next 下一步



在此我们选 USB (WDM only), USB Vendor ID 填写 0471 (为飞利浦厂商号), USB Product ID 填写 1983 (为我的出生年份),

嘿嘿,这里是厂品号,你可以自己随便填写。但是注意 USB Vendor ID 和 USB Product ID 要与固件程序一致。否则驱动不了硬件。 填写完后,Next 继续

dataBuffer		- ² 84	881	🔄 🖞 🐄	DDR.
		-×-		: Et d	<u>m</u>
I/WDM Device D	river V	izard (Driv	verVorks) - Ste	p 4 🌔
DriverWorks	Add an en One confi the endpo	try for each e guration will ints; you can	ndpoint or be created add config	n your USB 1 containi gurations	device. ng all of by editing
	Name	Address	IN/OUT	Туре	Max Xfer
	<u>A</u> dd	1 • In (this • Out (this Endpoint Typ BULK Max Transfer 64 Endpoint Nam Endpoint Nam	pipe is us pipe is u e Size e	sed to rea used to wr	d from the ite to the
< Back Nex	t >	, <u>-</u> ОК	1	<u>[</u>	encel

这里我们要填我们要用到的端点。让驱动配置端点的类型与缓冲数据大小。写完后点击 Suggest,让系统自动为我们命名 Endpoint Name,要说明的是端点 0 不需要我们配置,那是控制端点必须存在的! 下面是我配置好的端点



接下来 Next 下一步

a dataBuffer		
	\$\\\ \\ \\ \\	
IT/WDE Device I	river Vizard (Driver	rWorks) - Step 5
DriverWorks	The next few steps will as	sk you some questions about
	The WDM device driver crea	ated by the DriverWizard
	has a single "driver" cla	ss and a single "device"
	class. Drivers for NT 4.0	O and earlier may have
	class. Drivers for NT 4.(0 and earlier may have
	class. Drivers for NT 4.(O and earlier may have
	class. Drivers for NT 4.(0 and earlier may have
	class. Drivers for NT 4.(0 and earlier may have This is the name of
	Class. Drivers for NT 4.(Driver Class ISP1581	O and earlier may have This is the name of the C++ class used to implement your
	class. Drivers for NT 4.(Driver Class ISP1581 File Name	O and earlier may have This is the name of the C++ class used to implement your This is the file
	class. Drivers for NT 4.(Driver Class ISP1581 <u>F</u> ile Name ISP1581	O and earlier may have This is the name of the C++ class used to implement your This is the file name used for the
	class. Drivers for NT 4.(Driver Class ISP1581 <u>F</u> ile Name ISP1581	O and earlier may have This is the name of the C++ class used to implement your This is the file name used for the driver class

这里设置 Driver Class 类名和 File Name 文件名,默认 PASS 下去,当然你也可以改好听的名字。:) 继续 Next 下一步



这里我全部打勾了,其实只需要用到 Read,Write,Decive Control 就可以了。我们主要是通过 Decive Control 来操作 USB 的

好了,继续Next

a dataBuffer	🔽 🐪 🛛 🎬 🖄 💥
	····································
IT/WDM Device I	Driver Wizard (DriverWorks) - Step 7
DriverWorks	Request read DriverWirard can generate code to read from your USB device using BULK transfer mode. For ISOCH reads, refer to the USBISOCH sample driver for more information.
	✓ Generate <u>bulk</u> transfer read code Bulk Read Endpoint Endpoint1IN
A Tomposor	



图 8 与图 9 是让驱动向导自动为我们生成端点读写程序。这里我选了端点 1。端点 2 可以 COPY 端点 1 的代码,当然要相应 改动一点程序。 快要完了。再继续 Next





这里我们不用理会,继续下一步。

itaBuffer	▲ ▲ ▲ ▲ 型 ▲ ● 単 ▲ ▲ ●
T/WDM Device I)river Vizard (DriverVorks) - Step 11 [
DriverWorks	Define one or more device classes. ISP1581Device <u>R</u> ename
	Interface Buffers Power <u>D</u> evice name <mark>ISP1581Device</mark>
	Open device C Symbolic Link: ISP1581Device
	Interface (WDM only) Class GUID: { 0x54dc82f1, 0xfc17, 0x49d8,
on (* 1997) 🔷	Exclusive If "Exclusive" is checked, only one handle can be open to the device at

图 12 这里的 Class GUID 是我们连接 USB 的接口,不需要改动。我们选择 Buffers 一栏。



图 13

这里我们进行一些小批量数据的传输,选 Buffered.如果是高速之类的数据则可以考虑 Direct 好了接着下面是比较复杂点的功能设置了。设计用户接口函数要与硬件相匹配。比如我要做 USB 通信,则要有数据的发送与

接收。但驱动向导只帮我们完成了枚举设备和一些电源管理的功能。他还不能响应何时发数据何时接收数据。这要我们来完成 这部分的功能!

下面我们添加几个功能,为读和写数据。其中下面的 HOGUOWI_IOCTL_CONTROL 是,让 PC 发相应的厂商请求给 USB 硬件响应 后面将会说到下面的功能

HOGUOWI_IOCTL_CONTROL

HOGUOWI_IOCTL_READ_DATA

HOGUOWI_IOCTL_WRITE_DATA

taBuffer	🖌 🚺 🎽	8 📾) 🏭 🖁 🎽	3 (<u>111</u>)	
	@[- >]		6 ! D	1	
T/WDE Device I)river Vizard (D	river	Vorks) -	Step 12	
DriverWorke	Define control codes	s used	to communi	cate with	
	applications or othe	er driv	ordinal	h the Devic	e 10
	HOGUOWI IOCTL C	N N	0x800	Buffered	Any
	HOGUOWI IOCTL R	N	0x801	Buffered	Any
	<	1111)	>
	<u>Add</u>	<u>E</u> dit		Remove	· _
s 🐶 .	Include File Name:	ISP158	lioctl.h		

最后一步 Next



图 15

此时已经用向导完成了 USB 驱动文件。但是我们要实现 USB 通信的读和写数据还要填加相应的代码! 下面我们将会针对我们填加的 IOCTL

HOGUOWI_IOCTL_CONTROL HOGUOWI_IOCTL_READ_DATA HOGUOWI_IOCTL_WRITE_DATA 着重讲解

用向导生成 USB 驱动完后在 VC 的 WorkSpace 里应该有两个工程 应当把要生成 SYS 驱动的工程设置为当前的 Active 的工程,然后再编译(BUILD)应该就可以了。否则会生成 EXE 文件 注意要选 CLASS 类为当前的 Active 的工程

要完成 PC 对设备的 USB 端点 1 的写则要在驱动程序中加入下面的程序才行

NTSTATUS Isp1581Device::HOGUOWI_IOCTL_WRITE_DATA_Handler(KIrp I) {

ULONG ulReturned = 0; NTSTATUS status = STATUS_SUCCESS;

```
//t << "Entering Isp1581Device::HOGUOWI_IOCTL_WRITE_DATA_Handler, " << I << EOL;
// TODO: Verify that the input parameters are correct
             If not, return STATUS_INVALID_PARAMETER
//
    // Always ok to write 0 elements.
    /*
    if (I.WriteSize() == 0)
    {
        I.Information() = 0;
         return I.PnpComplete(this, STATUS_SUCCESS);
    }
    ULONG dwTotalSize = I.WriteSize(CURRENT);
    ULONG dwMaxSize = m_Endpoint1OUT.MaximumTransferSize();
    if (dwTotalSize > dwMaxSize)
    {
         ASSERT(dwMaxSize);
         dwTotalSize = dwMaxSize;
    }
    PUCHAR pBuffer = (PUCHAR)I.loctlBuffer();
*/
    //KMemory Mem(pBuffer, dwTotalSize);
    //Mem.SetPageArray();
// TODO: Handle the HOGUOWI_IOCTL_WRITE_DATA request, or
//
             defer the processing of the IRP (i.e. by queuing) and set
//
             status to STATUS_PENDING.
         PURB pUrb = m_Endpoint1OUT.BuildBulkTransfer(
                      (unsigned char*)I.IoctlBuffer(),
                                                             // Where is data coming from?
                      I.IoctlInputBufferSize(), // How much data to read?
                      FALSE,
                                      // direction (FALSE = OUT)
                      NULL
                                     // Link to next URB
                      );
         if (pUrb == NULL)
         {
             //delete pCompInfo;
             I.Information() = 0;
             return I.PnpComplete(this, STATUS_INSUFFICIENT_RESOURCES);
         }
```

```
status = m_Endpoint1OUT.SubmitUrb(pUrb, NULL, NULL, 1500L);
ulReturned = pUrb->UrbBulkOrInterruptTransfer.TransferBufferLength;
//delete pUrb;
```

// TODO: Assuming that the request was handled here. Set I.Information

```
// to indicate how much data to copy back to the user.
I.Information() =ulReturned;
I.Status() = status;
//I.Information() = 0;
```

return status;

}

要完成 PC 对设备的读比较复杂一点,因为 SLAVE 从机 USB 不能主动发 USB 数据给 PC,要响应请求才能发数据给 PC 读。 这里我们设置硬件低层厂商请求读 READ_DATA 为 11.

则我们添加 PC 驱动发送请求代码为:其中 #define CMD_READ_DATA 11,因为我们低层响应厂商请求读 READ_DATA 为 11,所以上层驱动 也要相应的发厂商请求代码 11。随着低层硬件的改动而改动。

NTSTATUS Isp1581Device::ISP1581_IOCTL_CONTROL_Handler(KIrp I)
{
 NTSTATUS status = STATUS_SUCCESS;
 ULONG ulReturned = 0;
}

//t << "Entering ISP1581Device::ISP1581_IOCTL_CONTROL_Handler, " << I << EOL; // TODO: Verify that the input parameters are correct // If not, return STATUS_INVALID_PARAMETER

```
// TODO: Handle the the ISP1581_IOCTL_CONTROL request, or
```

```
// defer the processing of the IRP (i.e. by queuing) and set
```

```
// status to STATUS_PENDING.
```

PURB pUrb = m_Lower.BuildVendorRequest(

```
(unsigned char*)I.IoctlBuffer(),
                                                          // transfer buffer
I.IoctlInputBufferSize(),
                            // transfer buffer size
                                      // request reserved bits
0,
(unsigned char)CMD_READ_DATA,
                                                // request
0,
                                      // Value
FALSE,
                                      // bIn
TRUE,
                                      // bShortOk
NULL,
                                      // Link
                                      // Index
0
```

```
);
```

```
// transmit
```

```
status = m_Lower.SubmitUrb(pUrb, NULL, NULL, 1500L);
if(NT_SUCCESS(status))
```

```
{
```

```
ulReturned = pUrb->UrbControlVendorClassRequest.TransferBufferLength;
```

```
}
```

```
//delete pUrb;
```

// TODO: Assuming that the request was handled here. Set I.Information

```
to indicate how much data to copy back to the user.
I.Information() = ulReturned;
I.Status() = status;
//I.Information() = 0;
```

return status;

```
}
```

//

```
完成这段还不行。我们还不能读取到 USB 的数据,此时我们只能看到 PC 发出的请求代码

21.0 CTL 40 0b 00 00 00 00 00 00 VENDOR 153.1.0

21.0 DO 40 @ 153.2.0

下面添加读取子程序。

NTSTATUS Isp1581Device::HOGUOWI_IOCTL_READ_DATA_Handler(KIrp I)

{

ULONG ulReturned = 0;

NTSTATUS status = STATUS_SUCCESS;
```

```
//t << "Entering Isp1581Device::HOGUOWI_IOCTL_READ_DATA_Handler, " << I << EOL;
// TODO: Verify that the input parameters are correct
// If not, return STATUS_INVALID_PARAMETER
```

```
if (I.ReadSize() == 0)
{
    I.Information() = 0;
    return I.PnpComplete(this, STATUS_SUCCESS);
}
```

```
PUCHAR pBuffer = (PUCHAR) I.BufferedReadDest();
```

ULONG dwTotalSize = I.ReadSize(CURRENT);

```
ULONG dwMaxSize = m_Endpoint1IN.MaximumTransferSize();

if (dwTotalSize > dwMaxSize)

{

ASSERT(dwMaxSize);

dwTotalSize = dwMaxSize;

}

// TODO: Handle the the HOGUOWI_IOCTL_READ_DATA request, or

// defer the processing of the IRP (i.e. by queuing) and set

// status to STATUS_PENDING.

PURB pUrb = m_Endpoint1IN.BuildBulkTransfer(
```

pBuffer,	// Where is data coming from?
dwTotalSize,	// How much data to read?
TRUE,	// direction (TRUE = IN)
NULL,	// Link to next URB

TRUE // Allow a short transfer); status = m_Endpoint1IN.SubmitUrb(pUrb, NULL, NULL, 1500L); ulReturned = pUrb->UrbBulkOrInterruptTransfer.TransferBufferLength; //delete pUrb; // TODO: Assuming that the request was handled here. Set I.Information to indicate how much data to copy back to the user. // //I.Information() = 0;I.Information() =ulReturned; I.Status() = status; return status; } 当完成这些后。PC 应用程序就可以通过上层 USB 驱动程序对 USB 硬件的读和写了! 这里给个注意,凡是用 DriverStudio2.6 开发 USB 上层驱动的话,要修改 DriverStudio2.6 中的 BUG 具体如下: 删除 pPipeInfo->PipeFlags |= USBD_PF_CHANGE_MAX_PACKET 這一行,此行在 Kusb.cpp 中的 KUsbLowerDevice::Configure(..) 内. 如果没有按照上面做的话,在进行 USB 驱动开发中, PC 对 USB 端点的读取会造成电脑的蓝屏 驱动文件 INF 注意事项 驱动文件的 inf 文件中的 Strings Section,不能与其他 inf 文件中的 Strings Section 的相同,否则驱动文件冲突不能正常驱动硬件 ;----- Strings Section ------[Strings] ProviderName="hoguowi" MfgName="Name of hoguowi Manufacturer here" DeviceDesc="中国人" DeviceClassName="USB 调试" SvcDesc="描述" 同时把 INF 下面的 Class 改为 Class=USB 和 ClassGUID 删除掉!这样驱动才能认为是 USB 设备 ; If device fits one of the standard classes, use the name and GUID here, ; otherwise create your own device class and GUID as this example shows. Class=NewDeviceClass ClassGUID={ff646f80-8def-11d2-9449-00105a075f6b} 与此同时最关键的一步是 INF 的[Strings]字符串描述中的 ProviderName, MfgName, DeviceDesc, DeviceClassName, SvcDesc 要与底层 USB 的硬件描述相一致,实验表明只要 DeviceDesc 设备描述符与硬件底层描述相符合即可,否则枚举不成功.(最终实验表明[Strings]只要不与已经存 在的驱动字符串冲突,该 INF 的字符串可以为任意值)最后当你看见 SET CONFIG 的时候.就是激动人心的时刻.恭喜你,你的底层与上层驱动完满成功. 在 Isp1581Device.cpp 添加

NTSTATUS Isp1581Device::OnQueryCapabilities(KIrp I)

{

}

//t << "Entering Isp1581Device::OnQueryCapabilities\n";</pre>

I.CopyParametersDown();

I.SetCompletionRoutine(LinkTo(OnQueryCapabilitiesComplete), this, TRUE, TRUE, TRUE);

return m_Lower.PnpCall(this, I);

if(!m_bSurpriseRemove)

I.DeviceCapabilities()->SurpriseRemovalOK = TRUE; I.DeviceCapabilities()->Removable = TRUE; I.DeviceCapabilities()->EjectSupported = TRUE; //I.DeviceCapabilities()->WarmEjectSupported = TRUE;
}
return STATUS_SUCCESS;
}

和 class Isp1581Device : public KPnpDevice 的
public:
ULONG m_bSurpriseRemove;
MEMBER_COMPLETEIRP(Isp1581Device, OnQueryCapabilitiesComplete)
virtual NTSTATUS OnQueryCapabilities(KIrp I);
即可去掉 USB 小图标显示

写应用文件的时候把驱动文件 OpenByIntf.cpp 添进工程

------下面是网上对 USB 驱动添加厂商请求的讲解------------------下面是网上对 USB 驱动添加厂商请求的讲解-------------

在制作上位 USB 驱动添加厂商标准请求的时候,注意添加正确的功能代码,否则写上位应用程序读取数据的时候会造成电脑蓝屏。

具体请看《开发 WDM 型 USB 设备驱动程序》

LED 控制处理例程 MyUSB_IOCTL_LED_Handler()

该例程是实现本驱动程序功能的关键例程,它是用来控制设备上的 LED 灯通断,主要利用 USB Vendor Request 来向设备传送。其中, request=1 的时候表示让 LED 亮, request=0 的时候让 LED 灭。它是通过 DeviceControl 由上层应用程序传下来。上位驱动实现代码如下:

NTSTATUS MyUSBDevice::MyUSB_IOCTL_LED_Handler(KIrp I)

```
{
```

}

```
NTSTATUS status = STATUS_INVALID_PARAMETER;
//检查输入参数是否正确,如果不正确,返回 STATUS_INVALID_PARAMETER
if(I.IoctlOutputBufferSize() || !I.IoctlBuffer() ||(I.IoctlInputBufferSize() != sizeof(UCHAR)))
  return status;
//处理 MyUSB_IOCTL_LED_ON 请求
PURB pUrb = m_Lower.BuildVendorRequest(NULL, // 传输缓冲区
  0,// 传输缓冲区大小
  0,// 请求保留位
  (UCHAR)(*(PUCHAR)I.IoctlBuffer()), // 请求 1=LED_ON, 0=LED_OFF
  0);// 值
//向下传送 URB
status = m_Lower.SubmitUrb(pUrb, NULL, NULL, 5000L);
//若请求在此处理,设置 I.Information 指示多少数据拷贝回用户
I.Information () =0;
I.Status () = status;
return status;
```

```
访问硬件例程 DeviceControl()
```

上层应用软件程序就是通过此例程来将 IRP 传到下层。

```
NTSTATUS MyUSBDevice::DeviceControl(KIrp I)
{
    NTSTATUS status;
    switch (I.IoctlCode())
    {
        case MyUSB_IOCTL_LED:
        status = MyUSB_IOCTL_LED_Handler(I);
```

break; default: // 未被声明的 I/O 控制请求 status = STATUS_INVALID_PARAMETER; break; } //-------应用程序编译注意问题-------当加入 OpenByInty.cpp(该文件在用 DS2。6 生成的文件中的 EXE 目录可以找到,复制到应用软件的根目录即可)编译时,出现 -------Configuration: ISP1581 - Win32 Debug-------Linking... OpenByIntf.obj : error LNK2001: unresolved external symbol __imp__SetupDiGetClassDevsA@16 OpenByIntf.obj : error LNK2001: unresolved external symbol __imp__SetupDiDestroyDeviceInfoList@4 OpenByIntf.obj : error LNK2001: unresolved external symbol __imp__SetupDiGetDeviceInfoList@4

OpenByIntf.obj : error LNK2001: unresolved external symbol __imp__SetupDiEnumDeviceInterfaces@20

Debug/ISP1581.exe : fatal error LNK1120: 4 unresolved externals Error executing link.exe.

ISP1581.exe - 5 error(s), 0 warning(s)

连接错误,在 project/settings/link 的 Object/libray modules 加 setupapi.lib 重新编译即可。但是还要注意在 OpenByInty.cpp 最前面#include "stdafx.h"头文件,否则也编译不通过。

接下来,我们把 USB 驱动文件中的 ISP1581DeviceInterface.h 和 ISP1581ioctl.h 两个头文件复制到应用程序的根目录 ISP1581DeviceInterface.h 包含了厂商请求的接口控制句柄 IOCTL, ISP1581ioctl.h 包含了 USB 的 CLASS_GUID 唯一序列号,打开设备就是需要到这个序列号才能正确 OPEN 相应的 USB。

接下来,我们要生成 USBdevice.cpp USB 设备的读写控制文件 USBdevice.h 的头文件为: // USBdevice.cpp: implementation of the Ctestdevice class. //

#include "stdafx.h"
//#include "testpro.h"
#include "USBdevice.h"
#include "ISP1581DeviceInterface.h"
#include "ISP1581ioctl.h"
#include <winioctl.h>

Ctestdevice::Ctestdevice()
{
}
Ctestdevice::~Ctestdevice()
{

}

DWORD Ctestdevice::IniDevice()//初始化 USB 设备

{

}

{

```
DWORD dError = 0;
hDevice = OpenByInterface( &ClassGuid, 0, &dError);
```

```
if (hDevice == INVALID_HANDLE_VALUE)
```

```
CString error;
error.Format("设备不可用,错误号(%d),请重新连接!",dError);
AfxMessageBox(error);
return dError;
}
else
{
AfxMessageBox("设备可使用!");
}
```

return ERROR_SUCCESS;

```
void Ctestdevice::CloseUSB()//关闭设备
{
     CloseHandle(hDevice);
}
DWORD Ctestdevice::WriteBulkData(int nIndex, LPBYTE lpBuf, int wLength)
{
    BOOL bRet = FALSE;
    DWORD dwReturned = 0;
    DWORD dwErr;
    HANDLE hDev = OpenByInterface( & ClassGuid, nIndex, & dwErr);
    if(hDev != INVALID_HANDLE_VALUE)
    {
        bRet=DeviceIoControl(hDev, HOGUOWI_IOCTL_WRITE_DATA, lpBuf, wLength, NULL, 0, &dwReturned, NULL);
        //bRet = WriteFile(hDev, lpBuf, wLength, &dwReturned, NULL);
    }
    CloseHandle(hDev);
    return ERROR_SUCCESS;
}
DWORD Ctestdevice::ReadBulkData(int nIndex, LPBYTE lpBuf, int wLength)
{
    BOOL bRet = FALSE;
    DWORD dwReturned = 0;
    DWORD dwErr;
    HANDLE hDev = OpenByInterface( & ClassGuid, nIndex, & dwErr);
    if(hDev != INVALID_HANDLE_VALUE)
    {
        BYTE bLen;
        bLen=LOBYTE(wLength);
        if (DeviceIoControl(hDev,HOGUOWI\_IOCTL\_CONTROL, \& bLen, size of (BYTE), NULL, 0, \& dwReturned, NULL))
                       bRet=DeviceIoControl(hDev, HOGUOWI_IOCTL_READ_DATA,NULL,0,lpBuf, wLength,&dwReturned, NULL);
        /*
    if (!DeviceIoControl(hDevice,
                          HOGUOWI_IOCTL_CONTROL,
                          pBuffer,
                          Length,
                          pBuffer,
                          Length,
                          dwBytesTransferred,
                          NULL)
       )
    {
        DWORD dError=GetLastError();
        CString error;
        error.Format("块读取错误%d",dError);
        AfxMessageBox(error);
```

/* CString num;

}

}

num.Format("块传读取数据个数是%d!",*dwBytesTransferred); AfxMessageBox(num);*/ } CloseHandle(hDev); //return bRet;

return ERROR_SUCCESS;

return dError;

好了,这样我们基本把必要的代码全部整理完毕,下面我们要测试一下是否能打开 USB 设备。 在 ISP1581Dlg.cpp 中添加头文件#include "USBdevice.h" 构造类 Ctestdevice m_mydevice; 在对话框初始化函数 BOOL CUSBDlg::OnInitDialog()中的

// TODO: Add extra initialization here 添加下面代码

m_mydevice.IniDevice();//初始化 USB 设备

编译程序运行之!!!

🛛 🖪 🗟 😽 🉀 hoguowi	
bers] 🕶 💊 OnInitDialog	- 🗷 -
if (tstr0houtMenu	IsEmptu())
{	
pSysMenu->App	endMenu(MF_SEPARATOR);
3 TSP1581	MANNA (WE_SIKING, IDW_HEDDIED
}	
// Sat the 🔥 设备i	可使用 !
// when t	ain window is not a dia
SetIcon(m 确定	// Set big icon
SetIcon(m	// Set small icon
// TODO: Add extra in:	itialization here
hoguowi_USBdevice	.IniDevice();//初始化USB设备
return TRUE; // return	rn TRUE unless you set the fo
如果插了 USB 设备则如上图	所示,止确识别开旦打开了 USB 设备
• 🗠 + 🖪 🗖 😽 🉀 hogud	owi 💽 🙀 🔤 😵
members] 🔹 💊 OnInitDialog	→ 🗷 →
if (!strAboutM	enu.IsEmptu())
<``	
pSysMenu->	AppendMenu(MF_SEPARATOR);
ISP1581	X, 104_H00011
设备不可用,	错误号(259), 请重新连接! framework_du
	is not a d
s Li	備定 big icon
N	SMALL 1CON
// TODO: Add extra	initialization here
hoguowi_USBdev	ice.IniDevice();//初始化USB设备
recurn inde, 77 h	eculi inoc untess you set the

上图表示打开 USB 设备失败