我的代码中需调用其它程序进行操作，而用一般的POSTMESSAGE SENDINPUT被屏蔽了  
  
因此使用WINIO进行一些稍微底层的硬件接口操作  
  
现在只能进行键盘操作 鼠标不行  
  
测试环境是2003server虚拟机 没办法ps/2接口的机器不多  
  
以下试了几种方式的代码 鼠标均没反映

键盘部分：（可用）

//--KBCWait4IBE函数等待状态寄存器-------------------------------------------

void KBCWait4IBE()

{

DWORD dwRegVal=0;

do

{

GetPortVal(0x64,&dwRegVal,1);

}

while(dwRegVal & 0x2);

}

void KEY\_DOWN(int vk\_in)

{

Sleep(1000);

KBCWait4IBE();

SetPortVal( 0X64, 0xD2, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal(0X60, 0xE2, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X64, 0xD2, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X60, MapVirtualKey(vk\_in, 0), 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X64, 0xD2, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal(0X60, 0xE0, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X64, 0xD2, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal(0X60, (MapVirtualKey(vk\_in, 0) | 0x80), 1);

}

void \_stdcall SetByte( BYTE chPort, BYTE chScanCode )

{

\_\_asm

{

push eax;

push edx;

//等待状态寄存器标志位OBF置0

wait\_obf:

in al, 0x64;

and al, 01b;

jnz wait\_obf;

// 写入数据

mov al, chScanCode;

mov dl, chPort;

mov dh, 0;

out dx, al;

pop edx;

pop eax;

}

}

void SetKey(char SCanCode)

{

//无论向0x60,还是0x64写东西前都要等状态寄存器OBF变0

\_asm{

Loop1:

in al, 0x64

and al, 10b

jnz Loop1

//向$64端口写命令

// mov al, 0xD2//写键盘输出缓存命令

mov al, 0xD3//写鼠标输出缓存命令

out 0x64, al

//无论向0x60,还是0x64写东西前都要等状态寄存器OBF变0

Loop2:

in al, 0x64

and al, 10b

jnz Loop2

//向$60端口写参数

mov al, SCanCode

out 0x60, al

}

}

鼠标部分：（不可用）

void Mouse\_Down()

{

// Sub MyMouseKey(MouseFun As Long, MyMouseX As Long, MyMouseY As Long, MyMouseZ As Long)

// ' 左键按下(MouseFun=9)。MyMouseX、MyMouseY、MyMouseZ 为0

// ' 右键按下(MouseFun=10)。MyMouseX、MyMouseY、MyMouseZ 为0

// ' 中键按下(MouseFun=12)。MyMouseX、MyMouseY、MyMouseZ 为0

// ' 任意键放开(MouseFun=8)。MyMouseX、MyMouseY、MyMouseZ 为0

// ' ------------------------------------

// ' 鼠标上移(MouseFun=8)。MyMouseY为移动距离，最大为255，最小为1。MyMouseX、MyMouseZ 为0

// ' 鼠标下移(MouseFun=40)。MyMouseY为移动距离，最大为1，最小为255。MyMouseX、MyMouseZ 为0

// ' 鼠标左移(MouseFun=24)。MyMouseX为移动距离，最大为1，最小为255。MyMouseY、MyMouseZ 为0

// ' 鼠标右移(MouseFun=8)。MyMouseX为移动距离，最大为255，最小为1。MyMouseY、MyMouseZ 为0

// ' ------------------------------------

Sleep(1000);

KBCWait4IBE();

SetPortVal( 0X64, 0xD3, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X60, 9, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X64, 0xD3, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X60, 0, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X64, 0xD3, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X60, 0, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X64, 0xD3, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X60, 0, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X64, 0xD3, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X60, 8, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X64, 0xD3, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X60, 0, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X64, 0xD3, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X60, 0, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X64, 0xD3, 1 );

Sleep(10);

KBCWait4IBE();

SetPortVal( 0X60, 0, 1 );

// KBCWait4IBE '等待缓冲区为空

// SetPortVal 100, 211, 1 '发送鼠标写入命令

// KBCWait4IBE '等待缓冲区为空

// SetPortVal 96, MouseFun, 1 '发送鼠标动作命令

// '-------------------------------------

// KBCWait4IBE '等待缓冲区为空

// SetPortVal 100, 211, 1 '发送鼠标写入命令

// KBCWait4IBE '等待缓冲区为空

// SetPortVal 96, MyMouseX, 1 '发送鼠标动作命令

// '-------------------------------------

// KBCWait4IBE '等待缓冲区为空

// SetPortVal 100, 211, 1 '发送鼠标写入命令

// KBCWait4IBE '等待缓冲区为空

// SetPortVal 96, MyMouseY, 1 '发送鼠标动作命令

// '-------------------------------------

// KBCWait4IBE '等待缓冲区为空

// SetPortVal 100, 211, 1 '发送鼠标写入命令

// KBCWait4IBE '等待缓冲区为空

// SetPortVal 96, MyMouseZ, 1 '发送鼠标动作命令

// End Sub

}

//--------------------------------------------------------------

void SetMouse(char byCode)

{

//无论向0x60,还是0x64写东西前都要等状态寄存器OBF变0

\_asm

{

Loop1:

in al, 0x64

and al, 10b

jnz Loop1

//向$64端口写命令

mov al, 0xD3//写鼠标输出缓存命令

out 0x64,al

Loop2:

in al, 0x64

and al, 10b

jnz Loop2

//向$60端口写参数

mov al, byCode

out 0x60,al

}

}

现在给大家再总结一下 大概就是下载WINRING0开源组件（因为这个有win7 64位数字签名，WINIO和一些网上很多的虚拟鼠标虚拟键盘都不太好用）  
  
主要代码如下（键盘和鼠标）

void OnKey(char vk\_in)

{

Sleep(5);

//KBCWait4IBE();

WriteIoPortByte( 0X64, 0xD2);

Sleep(5);

//KBCWait4IBE();

WriteIoPortByte(0X60, 0xE2);

Sleep(5);

//KBCWait4IBE();

WriteIoPortByte( 0X64, 0xD2);

Sleep(5);

//KBCWait4IBE();

WriteIoPortByte( 0X60, MapVirtualKey(vk\_in, 0));

Sleep(5);

//KBCWait4IBE();

WriteIoPortByte( 0X64, 0xD2);

Sleep(5);

//KBCWait4IBE();

WriteIoPortByte(0X60, 0xE0);

Sleep(5);

//KBCWait4IBE();

WriteIoPortByte( 0X64, 0xD2);

Sleep(5);

//KBCWait4IBE();

WriteIoPortByte(0X60, (MapVirtualKey(vk\_in, 0) | 0x80));

}

void OnMouse()

{

Sleep(5);

WriteIoPortByte( 0X64, 0xD3);

Sleep(5);

WriteIoPortByte( 0X60, 0x09);

Sleep(5);

WriteIoPortByte( 0X64, 0xD3);

Sleep(5);

WriteIoPortByte( 0X60, 0x00);

Sleep(5);

WriteIoPortByte( 0X64, 0xD3);

Sleep(5);

WriteIoPortByte( 0X60, 0x00);

Sleep(5);

WriteIoPortByte( 0X64, 0xD3);

Sleep(5);

WriteIoPortByte( 0X60, 0x00);

//松开

Sleep(5);

WriteIoPortByte( 0X64, 0xD3);

Sleep(5);

WriteIoPortByte( 0X60, 0x08);

Sleep(5);

WriteIoPortByte( 0X64, 0xD3);

Sleep(5);

WriteIoPortByte( 0X60, 0x00);

Sleep(5);

WriteIoPortByte( 0X64, 0xD3);

Sleep(5);

WriteIoPortByte( 0X60, 0x00);

Sleep(5);

WriteIoPortByte( 0X64, 0xD3);

Sleep(5);

WriteIoPortByte( 0X60, 0x00);

}

////////////////////////////////////

WinIO使用方法

int Init\_WinIo()  
{  
HMODULE h1 = LoadLibrary(\_T("WinIo64.dll"));  
if (h1 != NULL)  
{  
bRet1 iz = (bRet1)::GetProcAddress(h1, "InitializeWinIo");  
if (!iz)  
{  
printf("False in InitializeWinIo GetProcAddress!\n");  
return 1;  
}  
SPV = (bRet2)::GetProcAddress(h1, "SetPortVal");  
if (!SPV)  
{  
printf("False in SetPortVal GetProcAddress!\n");  
return 1;  
}  
GPV = (bRet3)::GetProcAddress(h1, "GetPortVal");  
if (!GPV)  
{  
printf("False in GetProtVal GetProcAddress!\n");  
return 1;  
}  
SD = (bVoid)::GetProcAddress(h1, "ShutdownWinIo");  
if (!SD)  
{  
printf("False in ShutdownWinIo GetProcAddress!\n");  
return 1;  
}  
}  
else  
{  
printf("Can not find WinIo32.dll or WinIo32.dll init fail!!\n");  
return 1;  
}  
printf("suscessful!!!\n");  
//GetLastError();  
  
return 0;  
}  
  
SPV(0xcf8, BusCount, 4);  
GPV(0xcfc, p, 4);  
  
读出来的数据是乱的， 每次都不一样~~~~

BusCount 和 p是否都是字符串，如果是需要在外面申请空间。  
  
HMODULE h1 = LoadLibrary(\_T("WinIo64.dll"));这个h1应该放在外面吧，局部变量在函数运行结束后要被收回的

WinIO程序库允许在32位的Windows应用程序中直接对I/O端口和[物理内存](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=1168027&ss_c=ssc.citiao.link" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)进行存取操作。通过使用一种内核模式的[设备驱动器](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=68753490" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)和其它几种底层编程技巧，它绕过了Windows系统的保护机制。

WinNT/2000/XP下，WinIO[函数库](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=26244278" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)只允许被具有管理者权限的[应用程序](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=729595" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)调用。如果使用者不是以管理者的身份进入的，则WinIO.DLL不能够被安装，也不能激活WinIO[驱动器](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=2022122" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)。通过在管理者权限下安装驱动器软件就可以克服这种限制。然而，在这种情况下，ShutdownWinIo函数不能在应用程序结束之前被调用，因为该函数将WinIO驱动程序从[系统注册表](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=1958568" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)中删除。

该函数库提供8个函数功能调用：

[bool](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=783501" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank) \_stdcall InitializeWinIo();

本函数[初始化](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=267400" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)WioIO函数库。

必须在调用所有其它功能函数之前调用本函数。

如果[函数调用](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=8407495" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)成功，[返回值](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=581547" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)为非零值。

如果调用失败，则返回值为0。

[void](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=659573" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank) \_stdcall ShutdownWinIo();

本函数在内存中清除WinIO库

本函数必须在中止应用函数之前或者不再需要WinIO库时调用，

bool \_stdcall GetPortVal(WORD wPortAddr, PDWORD [pdw](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7886326" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)PortVal, BYTE bSize);

使用此函数从一个输入或输出端口读取一个字节/字/双字数据。

参数：

wPortAddr – 输入输出[端口地址](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7532286" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)

pdwPortVal – 指向双字[变量](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=511389" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)的[指针](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=252787" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)，接收从端口得到的数据。

bSize – 需要读的字节数，可以是1 (BYTE), 2 (WORD) or 4 (DWORD).

如果调用成功，则返回非零值。

如果函数调用失败，则函数返回值为零。

bool \_stdcall SetPortVal(WORD wPortAddr, DWORD dwPortVal, BYTE bSize);

使用本函数将一个字节/字/双字的数据写入输入或[输出接口](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=68460973" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)。

参数：

wPortAddr – 输入输出口地址

dwPortVal – 要写入口的数据

bSize – 要写的数据个数，可以是 1 (BYTE), 2 (WORD) or 4 (DWORD).

如果调用成功，则返回非零值。

如果函数调用失败，则函数返回值为零。

PBYTE \_stdcall MapPhysToLin(PBYTE pbPhysAddr, DWORD dwPhysSize, HANDLE \*pPhysicalMemoryHandle)

使用此函数将[物理内存](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=1168027" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)的一部分映射到一个32位应用程序的[线性地址](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=75918172" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)空间。

下面是一个例子：

PBYTE pbLinAddr;

HANDLE PhysicalMemoryHandle;

pbLinAddr = MapPhysToLin(0xA0000, 65536, &PhysicalMemoryHandle);

该函数将把[物理地址](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=64979662" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)范围为0xA0000 - 0xAFFFF的[地址空间](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=11026955" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)映射到与应用程序对应的线性地址空间。[返回值](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=581547" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)为一个与物理地址0xA0000相关的线性地址。如果出现错误，则返回值为NULL。

参数：

pbPhysAddr – 指向物理地址的指针

dwPhysSize – 需要映射的字节数

pPhysicalMemoryHandle – 变量指针，如果调用成功，负责接收物理内存句柄。随后本句柄在调用UnmapPhysicalMemory函数时作为其第一个[参数](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=115950" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)。

bool \_stdcall UnmapPhysicalMemory(HANDLE PhysicalMemoryHandle, PBYTE

pbLinAddr)

使用本函数解除原先使用MapPhysToLin函数映射的一段线性物理内存区域，该区域被映射到应用程序所属的线性地址空间。

Windows 9x 应用程序不需要调用此函数。

参数：

PhysicalMemoryHandle – 物理内存区域所属的句柄，此参数由对MapPhysToLin函数的调用返回。 　　pbLinAddr – MapPhysToLin函数调用返回的线性地址。

bool \_stdcall GetPhysLong(PBYTE pbPhysAddr, PDWORD pdw

PhysVal);

从指定的物理地址读取一个双字数据。

参数：

pbPhysAddr – 指向物理地址的指针。

pdwPhysVal – 指向一个双字变量的指针，接收从物理内存中传来的数据。

如果此函数调用成功，返回一个非零值。

如果函数调用失败，则返回一个零值。

bool \_stdcall SetPhysLong(PBYTE pbPhysAddr, DWORD dwPhysVal);

将一个双字型数据写入指定的物理地址。

参数：

pbPhysAddr – 指向物理地址的指针。

pdwPhysVal – 指定待写入物理内存地址出的双字型数据。

如果此函数调用成功，返回一个非零值。

如果函数调用失败，则返回一个零值。

[WinIo库在VC应用程序中的使用(WinIo库下载)](http://baike.sogou.com/Create.e;jsessionid=839A61DF30354C87D8C3FD2ED0700811?sp=2&sp=l42465538&sp=6" \o "编辑本段" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)

为了在VC中能正常使用WinIo库,必须按以下步骤进行配置：

(1)：将WinIo.dll、WinIo.sys、WINIO.VXD三个文件放在程序可执行文件所在目录下；

(2)：将WinIo.lib添加到工程中,WinIo.lib及winio.h文件必须放在工程目录下；

(3)：在StdAfx.h[头文件](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=252971" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)中加入#include "winio.h"语句；

(4)：调用InitializeWinIo函数初始化WinIo驱动库；

(5)：调用读写IO口的GetPortVal或SetPortVal函数；

(6)：调用ShutdownWinIo函数；

在非管理员权限下运行，必须首先完成以下步骤：

(1)：将WinIo.dll、WinIo.sys、WINIO.VXD三个文件放在任一WinIo应用程序可执行文件所在目录下；

(2)：以管理员或其它具有管理员权限的用户身份登陆；

(3)：调用InstallWinIoDriver函数，第一个参数设置为WinIo.sys文件所在[目录路径](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=72343852" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)，第二个参数设置为[false](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=7590224" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)；

(4)：[重新启动](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=4999339" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)系统；

(5)：以普通用户身份登录，现在可以调用WinIo[库函数](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=8431666" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)；

(6)：当不再需要WinIo库时，可以再次以管理员身份或其它具有管理员权限的用户身份登陆系统，调用RemoveWinIoDriver[卸载](http://baike.sogou.com/lemma/ShowInnerLink.htm?lemmaId=126278" \t "http://blog.csdn.net/zdhlwt2008/article/details/_blank)该库；