PCI2321
高驱动数字量输入输出卡

WIN2000/XP 驱动程序使用说明书

北京阿尔泰科技发展有限公司
产品研发部修订

请您务必阅读《使用纲要》，他会使您事半功倍!

目　录

提醒用户：

通常情况下，WINDOWS 系统在安装时自带的 DLL 库和驱动不全，所以您不管使用那种语言编程，请您最好先安装上 Visual C++6.0 版本的软件，方可使我们的驱动程序有更完备的运行环境。

有关设备驱动安装和产品二次发行请参考 PCI2321Inst.doc 文档。

第一章 版权信息与命名约定

第一节、版权信息

第二节、命名约定

一、为简化文字内容，突出重点，本文中提到的函数名通常为基本功能名部分，其前缀设备名如 PCIxxxx_ 则被省略。如 PCI2321_CreateDevice 则写为 CreateDevice。

二、函数名及参数中各种关键字缩写规则

缩写	全称	汉语意思	缩写	全称	汉语意思
Dev	Device	设备	DI	Digital Input	数字量输入
Pro	Program	程序	DO	Digital Output	数字量输出
Int	Interrupt	中断	CNT	Counter	计数器
Dma	Direct Memory Access	直接内存存取	DA	Digital convert to Analog	数模转换
AD	Analog convert to Digital	模数转换	DI	Differential	(双端或差分) 注：在常量选项中
Npt	Not Empty	非空	SE	Single end	单端
Para	Parameter	参数	DIR	Direction	方向
SRC	Source	源	ATR	Analog Trigger	模拟量触发
TRIG	Trigger	触发	DTR	Digital Trigger	数字量触发
CLK	Clock	时钟	Cur	Current	当前的
GND	Ground	地	OPT	Operate	操作
Lgc	Logical	逻辑的	ID	Identifier	标识
Phys	Physical	物理的			

以上规则不局限于该产品。

第二章 使用纲要

第一节、如何管理 PCI 设备

由于我们的驱动程序采用面向对象编程，所以要使用设备的一切功能，则必须首先用CreateDevice函数创建一个设备对象句柄hDevice，有了这个句柄，您就拥有了对该设备的控制权。然后将此句柄作为参数传递给其他函数，如SetDeviceDO函数可用实现开关量的输出等。最后可以通过ReleaseDevice将hDevice释放掉。

第二节、如何实现开关量的简便操作

当您有了hDevice设备对象句柄后，便可用SetDeviceDO函数实现开关量的输出操作，其各路开关量的输出状态由其 bDIOSts[8]中的相应元素决定。

第三节、哪些函数对您不是必须的

当公共函数如CreateFileObject，WriteFile，ReadFile等一般来说都是辅助性函数，除非您要使用存盘功能。

它们只是对我公司驱动程序的一种功能补充，对用户额外提供的。

第三章　PCI 设备专用函数接口介绍

第一节、设备驱动接口函数列表（每个函数省略了前缀“PCI2321_”）

本章函数是设备使用 PCI 方式传输时所使用的。

函数名	函数功能	备注
① 设备对象操作函数		
CreateDevice	创建 PCI 对象(用设备逻辑号)	
GetDeviceCount	取得同一种 PCI 设备的总台数	上层及底层用户
GetDeviceCurrentID	取得指定设备的逻辑 ID 和物理 ID	上层及底层用户
ListDeviceDlg	列表所有同一种 PCI 设备的各种配置	上层及底层用户
ReleaseDevice	关闭设备，且释放 PCI 总线设备对象	
② P1 口开关量函数		
EnableStsP1DIO	设置开关量输入或输出状态	上层用户
GetDeviceDI_P1A	DIO0～DIO7 开关输入函数	上层用户
SetDeviceDO_P1A	DIO0～DIO7 开关输出函数	上层用户
GetDeviceDI_P1B	DIO8～DIO15 开关输入函数	上层用户
SetDeviceDO_P1B	DIO8～DIO15 开关输出函数	上层用户
GetDeviceDI_P1C	DIO16～DIO23 开关输入函数	上层用户
SetDeviceDO_P1C	DIO16～DIO23 开关输出函数	上层用户
③ P2 口开关量函数		
EnableStsP2DIO	设置开关量输入或输出状态	上层用户
GetDeviceDI_P2A	DIO24～DIO31 开关输入函数	上层用户
SetDeviceDO_P2A	DIO24～DIO31 开关输出函数	上层用户
GetDeviceDI_P2B	DIO32～DIO39 开关输入函数	上层用户
SetDeviceDO_P2B	DIO32～DIO39 开关输出函数	上层用户
GetDeviceDI_P2C	DIO40～DIO47 开关输入函数	上层用户
SetDeviceDO_P2C	DIO40～DIO47 开关输出函数	上层用户
④计数器控制函数		
SetDevCNTInitValue	设置计数器初值	上层用户
EnableDevCNT	是否使能计数器	上层用户
GetDevCNTVal	取得所有计数器的计数值	上层用户
ResetDevCNT	计数器复位清零	上层用户
⑤中断函数		
InitDeviceInt	初始化中断	上层用户
GetDeviceIntCount	在中断初始化后，用它取得中断服务程序产生的次数	上层用户
ReleaseDeviceInt	释放中断资源	上层用户

使用需知

Visual C++：

首先将 PCI2321.h 和 PCI2321.lib 两个驱动库文件从相应的演示程序文件夹下复制到您的源程序文件夹中，然后在您的源程序头部添加如下语句，以便将驱动库函数接口的原型定义信息和驱动接口导入库

(PCI2321.lib)加入到您的工程中。

#include “PCI2321.H”

在 VC 中，为了使用方便，避免重复定义和包含，您最好将以上语句放在 StdAfx.h 文件。一旦完成了以上工作，那么使用设备的驱动程序接口就跟使用 VC/C++Builder 自身的各种函数，其方法一样简单，毫无二别。

关于 PCI2321.h 和 PCI2321.lib 两个文件均可在演示程序文件夹下面找到。

Visual Basic:

首先将 PCI2321.Bas 驱动模块头文件从 VB 的演示程序文件夹下复制到您的源程序文件夹中，然后将此模块文件加入到您的 VB 工程中。其方法是选择 VB 编程环境中的工程(Project)菜单，执行其中的"添加模块"(Add Module)命令，在弹出的对话中选择 PCI2321.Bas 模块文件即可，一旦完成以上工作后，那么使用设备的驱动程序接口就跟使用 VB 自身的各种函数，其方法一样简单，毫无二别。

请注意，因考虑 Visual C++和 Visual Basic 两种语言的兼容问题，在下列函数说明和示范程序中，所举的 Visual Basic 程序均是需要编译后在独立环境中运行。所以用户若在解释环境中运行这些代码，我们不保证能完全顺利运行。

LabView / CVI:

LabVIEW 是美国国家仪器公司(National Instrument)推出的一种基于图形开发、调试和运行程序的集成化环境，是目前国际上唯一的编译型的图形化编程语言。在以 PC 机为基础的测量和工控软件中，LabVIEW 的市场普及率仅次于 C++/C 语言。LabVIEW 开发环境具有一系列优点，从其流程图式的编程、不需预先编译就存在的语法检查、调试过程使用的数据探针，到其丰富的函数功能、数值分析、信号处理和设备驱动等功能，都令人称道。关于 LabView/CVI 的驱动程序接口的详细说明请参考其演示源程序。

第二节、设备对象管理函数原型说明

♦ 创建设备对象函数

函数原型：

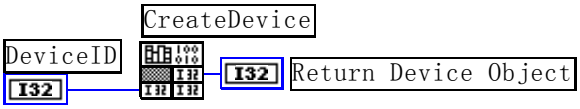
Visual C++：

HANDLE CreateDevice (int DeviceID = 0)

Visual Basic：

Declare Function CreateDevice Lib"PCI2321_32" (ByVal DeviceLgcID As Long) As Long

LabVIEW：



功能：该函数使用逻辑号创建设备对象，并返回其设备对象句柄 hDevice。只有成功获取 hDevice，您才能实现对该设备所有功能的访问。

参数：DeviceID 设备 ID(Identifier)标识号。当向同一个 Windows 系统中加入若干相同类型的设备时，系统将以该设备的“基本名称”与 DeviceID 标识值为名称后缀的标识符来确认和管理该设备。默认值为 0。

返回值：如果执行成功，则返回设备对象句柄；如果没有成功，则返回错误码 INVALID_HANDLE_VALUE。由于此函数已带容错处理，即若出错，它会自动弹出一个对话框告诉您出错的原因。您只需要对此函数的返回值作一个条件处理即可，别的任何事情您都不必做。

相关函数：ReleaseDevice

♦ 取得本计算机系统中 PCI2321 设备的总数量

函数原型：

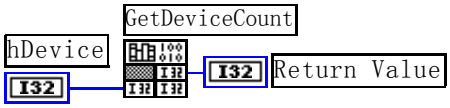
Visual C++:

int GetDeviceCount (HANDLE hDevice)

Visual Basic:

Declare Function GetDeviceCount Lib "PCI2321_32" (ByVal hDevice As Long) As Integer

LabVIEW：



功能：取得 PCI2321 设备的数量。

参数：hDevice设备对象句柄，它应由CreateDevice创建。

返回值：返回系统中 PCI2321 的数量。

相关函数： CreateDevice GetDeviceCount GetDeviceCurrentID ListDeviceDlg ReleaseDevice

♦ **取得该设备当前逻辑 ID 和物理 ID**

函数原型：

Visual C++:

BOOL GetDeviceCurrentID (HANDLE hDevice，
PLONG DeviceLgcID，
PLONG DevicePhysID)

Visual Basic:

Declare Function GetDeviceCurrentID Lib "PCI2321_32" (ByVal hDevice As Long,_
ByRef DeviceLgcID As Long,_
ByRef DevicePhysIDAs Long) As Boolean

LabVIEW：

请参考相关演示程序。

功能：取得指定设备逻辑和物理 ID 号。

参数：

hDevice 设备对象句柄，它指向要取得逻辑和物理号的设备，它应由CreateDevice创建。

DeviceLgcID 返回设备的逻辑 ID，它的取值范围为[0, 15]。

DevicePhysID 返回设备的物理 ID，它的取值范围为[0, 15]，它的具体值由卡上的拔码器 DID 决定。

返回值：如果初始化设备对象成功，则返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数： CreateDevice GetDeviceCount GetDeviceCurrentID ListDeviceDlg ReleaseDevice

♦ **用对话框控件列表计算机系统中所有 PCI2321 设备各种配置信息**

函数原型：

Visual C++:

BOOL ListDeviceDlg (HANDLE hDevice)

Visual Basic:

Declare Function ListDeviceDlg Lib "PCI2321_32" (ByVal hDevice As Long) As Boolean

LabVIEW：

请参考相关演示程序。

功能：列表系统中 PCI2321 的硬件配置信息。
参数：hDevice设备对象句柄，它应由CreateDevice创建。
返回值：若成功，则弹出对话框控件列表所有 PCI2321 设备的配置情况。
相关函数：CreateDevice　　ReleaseDevice

- **释放设备对象所占的系统资源及设备对象**

函数原型：

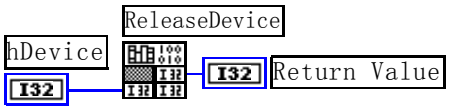
Visual C++:

```
BOOL ReleaseDevice(HANDLE hDevice)
```

Visual Basic:

```
Declare Function ReleaseDevice Lib "PCI2321_32" (ByVal hDevice As Long ) As Boolean
```

LabVIEW:



功能：释放设备对象所占用的系统资源及设备对象自身。
参数：hDevice设备对象句柄，它应由CreateDevice创建。
返回值：若成功，则返回TRUE，否则返回FALSE，用户可以用GetLastErrorEx捕获错误码。
相关函数：CreateDevice

应注意的是，CreateDevice必须和ReleaseDevice函数一一对应，即当您执行了一次CreateDevice后，再一次执行这些函数前，必须执行一次ReleaseDevice函数，以释放由CreateDevice占用的系统软硬件资源，如系统内存等。只有这样，当您再次调用CreateDevice函数时，那些软硬件资源才可被再次使用。

第三节、P1 口 DIO 数字开关量输入输出简易操作函数原型说明

- **设置开关量输入或输出状态**

函数原型：

Visual C++:

```
BOOL EnableStsP1DIO (HANDLE hDevice,
                     BOOL bExtTrig,
                     BOOL bP1Sts[3])
```

Visual Basic:

```
Declare Function EnableStsP1DIO Lib"PCI2321_32" (_
                                ByVal hDevice As Long,_
                                ByVal bExtTrig As Boolean,_
                                ByRef bP1Sts As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：设置开关量输入或输出状态。
参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bExtTrig 是否使能外部触发功能，TRUE：使能，FALSE：禁止。

bP1Sts bP1Sts[0]～bP1Sts[2]分别控制 P1A、P1B、P1C 端口。FALSE：开关量输入，TRUE：开关量输出。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice ReleaseDevice

一、对P1A口操作

♦ P1A 口 DIO0～DIO7 八路开关量输入

函数原型：

Visual C++:

```
BOOL GetDeviceDI_P1A (HANDLE hDevice,
                      BYTE bDIOSts [8])
```

Visual Basic:

```
Declare Function GetDeviceDI_P1A Lib "PCI2321_32" (_
                                    ByVal hDevice As Long,_
                                    ByRef bDIOsts As Byte) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO0～DIO7 输入开关量状态读入内存。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDIOSts八路开关量输入状态的参数结构，共有 8 个成员变量，分别对应于DIO0～DIO7 路开关量输入状态位。如果 bDIOSts [0]为“1”则使 0 通道处于开状态，若为“0”则 0 通道为关状态。其他同理。具体定义请参考《DI数字开关量输入参数介绍》章节。

返回值：若成功，返回 TRUE，其 bDIOSts[x]中的值有效；否则返回 FALSE，其 bDIOSts[x]中的值无效。

相关函数： CreateDevice EnableStsP1DIO SetDeviceDO_P1A ReleaseDevice

♦ P1A 口 DIO0～DIO7 八路开关量输出

函数原型：

Visual C++:

```
BOOL SetDeviceDO_P1A (HANDLE hDevice,
                      BYTE bDIOSts[8])
```

Visual Basic:

```
Declare Function SetDeviceDO_P1A Lib "PCI2321_32" (_
                                    ByVal hDevice As Long,_
                                    ByRef bDIOsts As Byte) As Boolean
```

LabView：

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO0～DIO7 输出开关量置成相应的状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDIOSts 八路开关量输出状态的参数结构，共有 8 个成员变量，分别对应于DIO0～DIO7 路开关量输出状态位。比如置 bDIOSts[0]为“1”则使 0 通道处于“开”状态，若为“0”则置 0 通道为“关”状态。其他同理。

请注意，在实际执行这个函数之前，必须对这个参数结构的DIO0 至DIO7 共 8 个成员变量赋初值，其值必须为“1”或“0”。具体定义请参考《DO数字开关量输出参数介绍》。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice EnableStsP1DIO GetDeviceDI_P1A ReleaseDevice

二、对P1B口操作

♦ P1B 口 DIO8～DIO15 八路开关量输入

函数原型：

Visual C++:

```
BOOL GetDeviceDI_P1B (HANDLE hDevice,
                      BYTE bDIOSts[8])
```

Visual Basic:

```
Declare Function GetDeviceDI_P1B Lib"PCI2321_32" (_
                                  ByVal hDevice As Long, _
                                  ByRef bDIOsts As Byte) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO8～DIO15 输入开关量状态读入内存。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDIOSts八路开关量输入状态的参数结构，共有 8 个成员变量，分别对应于DIO8～DIO15 路开关量输入状态位。如果 bDIOSts[0]为“1”则使 0 通道处于开状态，若为“0”则 0 通道为关状态。其他同理。具体定义请参考《DI数字开关量输入参数介绍》章节。

返回值：若成功，返回 TRUE，其 bDIOSts[x]中的值有效；否则返回 FALSE，其 bDIOSts[x]中的值无效。

相关函数： CreateDevice EnableStsP1DIO SetDeviceDO_P1B ReleaseDevice

♦ P1B 口 DIO8～DIO15 八路开关量输出

函数原型：

Visual C++:

```
BOOL SetDeviceDO_P1B (HANDLE hDevice,
                      BYTE bDIOSts[8])
```

Visual Basic:

```
Declare Function SetDeviceDO_P1B Lib "PCI2321_32" (_
                                  ByVal hDevice As Long,_
                                  ByRef bDIOsts As Byte) As Boolean
```

LabView：

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO8～DIO15 输出开关量置成相应的状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDIOSts 八路开关量输出状态的参数结构，共有 8 个成员变量，分别对应于DIO8～DIO15 路开关量输出状

态位。比如置 bDIOSts[0]为“1”则使 0 通道处于“开”状态，若为“0”则置 0 通道为“关”状态。其他同理。请注意，在实际执行这个函数之前，必须对这个参数结构的DIO8 至DIO15 共 8 个成员变量赋初值，其值必须为“1”或“0”。具体定义请参考《DO数字开关量输出参数介绍》。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice EnableStsP1DIO GetDeviceDI_P1B ReleaseDevice

三、对P1C口操作

♦ P1C 口 DIO16～DIO23 八路开关量输入

函数原型：

Visual C++:

```
BOOL GetDeviceDI_P1C (HANDLE hDevice,
                      BYTE bDIOSts[8])
```

Visual Basic:

```
Declare Function GetDeviceDI_P1C Lib "PCI2321_32" (_
                                    ByVal hDevice As Long,_
                                    ByRef bDIOsts As Byte) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO16～DIO23 输入开关量状态读入内存。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDIOSts八路开关量输入状态的参数结构，共有 8 个成员变量，分别对应于DIO16～DIO23 路开关量输入状态位。如果 bDIOSts[0]为“1”则使 0 通道处于开状态，若为“0”则 0 通道为关状态。其他同理。具体定义请参考《DI数字开关量输入参数介绍》章节。

返回值：若成功，返回 TRUE，其 bDIOSts[x]中的值有效；否则返回 FALSE，其 bDIOSts[x]中的值无效。

相关函数： CreateDevice EnableStsP1DIO SetDeviceDO_P1C ReleaseDevice

♦ P1C 口 DIO16～DIO23 八路开关量输出

函数原型：

Visual C++:

```
BOOL SetDeviceDO_P1C (HANDLE hDevice,
                      BYTE bDIOSts[8])
```

Visual Basic:

```
Declare Function SetDeviceDO_P1C Lib "PCI2321_32" (_
                                    ByVal hDevice As Long,_
                                    ByRef bDIOsts As Byte) As Boolean
```

LabView：

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO16～DIO23 输出开关量置成相应的状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDIOSts 八路开关量输出状态的参数结构，共有 8 个成员变量，分别对应于DIO16～DIO23 路开关量输出状态位。比如置 bDIOSts[0]为“1”则使 0 通道处于“开”状态，若为“0”则置 0 通道为“关”状态。其他同理。请注意，在实际执行这个函数之前，必须对这个参数结构的DIO16 至DIO23 共 8 个成员变量赋初值，其值必须为“1”或“0”。具体定义请参考《DO数字开关量输出参数介绍》。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice EnableStsP1DIO GetDeviceDI_P1C ReleaseDevice

第四节、P2 口 DIO 数字开关量输入输出简易操作函数原型说明

♦ 设置开关量输入或输出状态

函数原型：

Visual C++:

```
BOOL EnableStsP2DIO (HANDLE hDevice,
                     BOOL bExtTrig,
                     BOOL bP2Sts[3])
```

Visual Basic:

```
Declare Function EnableStsP2DIO Lib"PCI2321_32" (_
                                    ByVal hDevice As Long,_
                                    ByVal bExtTrig As Boolean,_
                                    ByRef bP2Sts As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：设置开关量输入或输出状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bExtTrig 是否使能外部触发功能，TRUE：使能，FALSE：禁止。

bP2Sts bP2Sts[0]～bP2Sts[2]分别控制 P2A、P2B、P2C 端口。FALSE：开关量输入，TRUE：开关量输出。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice ReleaseDevice

一、对P2A口操作

♦ P2A 口 DIO24～DIO31 八路开关量输入

函数原型：

Visual C++:

```
BOOL GetDeviceDI_P2A (HANDLE hDevice,
                      BYTE bDIOSts [8])
```

Visual Basic:

```
Declare Function GetDeviceDI_P2A Lib "PCI2321_32" (_
                                    ByVal hDevice As Long, _
                                    ByRef bDIOsts As Byte) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO24～DIO31 输入开关量状态读入内存。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDIOSts八路开关量输入状态的参数结构，共有 8 个成员变量，分别对应于DIO24～DIO31 路开关量输入状态位。如果 bDIOSts [0]为“1”则使 0 通道处于开状态，若为“0”则 0 通道为关状态。其他同理。具体定义请参考《DI数字开关量输入参数介绍》章节。

返回值：若成功，返回 TRUE，其 bDIOSts[x]中的值有效；否则返回 FALSE，其 bDIOSts[x]中的值无效。

相关函数： CreateDevice EnableStsP2DIO SetDeviceDO_P2A ReleaseDevice

♦ P2A 口 DIO24～DIO31 八路开关量输出

函数原型：

Visual C++:

```
BOOL SetDeviceDO_P2A (HANDLE hDevice,
                      BYTE bDIOSts[8])
```

Visual Basic:

```
Declare Function SetDeviceDO_P2A Lib "PCI2321_32" (_
                                    ByVal hDevice As Long,_
                                    ByRef bDIOsts As Byte) As Boolean
```

LabView：

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO24～DIO31 输出开关量置成相应的状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDIOSts 八路开关量输出状态的参数结构，共有 8 个成员变量，分别对应于DIO24～DIO31 路开关量输出状态位。比如置 bDIOSts[0]为“1”则使 0 通道处于“开”状态，若为“0”则置 0 通道为“关”状态。其他同理。请注意，在实际执行这个函数之前，必须对这个参数结构的DIO24 至DIO31 共 8 个成员变量赋初值，其值必须为“1”或“0”。具体定义请参考《DO数字开关量输出参数介绍》。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice EnableStsP2DIO GetDeviceDI_P2A ReleaseDevice

二、对P2B口操作

♦ P2B 口 DIO32～DIO39 八路开关量输入

函数原型：

Visual C++:

```
BOOL GetDeviceDI_P2B (HANDLE hDevice,
                      BYTE bDIOSts[8])
```

Visual Basic:

```
Declare Function GetDeviceDI_P2B Lib "PCI2321_32" (_
                                    ByVal hDevice As Long, _
                                    ByRef bDIOsts As Byte) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO32～DIO39 输入开关量状态读入内存。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDIOSts八路开关量输入状态的参数结构，共有 8 个成员变量，分别对应于DIO32～DIO39 路开关量输入状态位。如果bDIOSts[0]为“1”则使 0 通道处于开状态，若为“0”则 0 通道为关状态。其他同理。具体定义请参考《DI数字开关量输入参数介绍》章节。

返回值：若成功，返回 TRUE，其 bDIOSts[x]中的值有效；否则返回 FALSE，其 bDIOSts[x]中的值无效。

相关函数： CreateDevice EnableStsP2DIO SetDeviceDO_P2B ReleaseDevice

♦ P2B 口 DIO32～DIO39 八路开关量输出

函数原型：

Visual C++:

```
BOOL SetDeviceDO_P2B (HANDLE hDevice,
                      BYTE bDIOSts[8])
```

Visual Basic:

```
Declare Function SetDeviceDO_P2BLib"PCI2321_32" (_
                                    ByVal hDevice As Long, _
                                    ByRef bDIOsts As Byte) As Boolean
```

LabView：

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO32～DIO39 输出开关量置成相应的状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDIOSts 八路开关量输出状态的参数结构，共有 8 个成员变量，分别对应于DIO32～DIO39 路开关量输出状态位。比如置bDIOSts[0]为“1”则使 0 通道处于“开”状态，若为“0”则置 0 通道为“关”状态。其他同理。请注意，在实际执行这个函数之前，必须对这个参数结构的DIO32 至DIO39 共 8 个成员变量赋初值，其值必须为“1”或“0”。具体定义请参考《DO数字开关量输出参数介绍》。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice EnableStsP2DIO GetDeviceDI_P2B ReleaseDevice

三、对P2C口操作

♦ P2C 口 DIO40～DIO47 八路开关量输入

函数原型：

Visual C++:

```
BOOL GetDeviceDI_P2C (HANDLE hDevice,
                      BYTE bDIOSts[8])
```

Visual Basic:

```
Declare Function GetDeviceDI_P2C Lib"PCI2321_32" (_
                                    ByVal hDevice As Long,_
                                    ByRef bDIOsts As Byte) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO40～DIO47 输入开关量状态读入内存。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDIOSts八路开关量输入状态的参数结构，共有 8 个成员变量，分别对应于DIO40～DIO47 路开关量输入状态位。如果 bDIOSts[0]为“1”则使 0 通道处于开状态，若为“0”则 0 通道为关状态。其他同理。具体定义请参考《DI数字开关量输入参数介绍》章节。

返回值：若成功，返回 TRUE，其 bDIOSts[x]中的值有效；否则返回 FALSE，其 bDIOSts[x]中的值无效。

相关函数： CreateDevice EnableStsP2DIO SetDeviceDO_P2C ReleaseDevice

♦ **P2C 口 DIO40～DIO47 八路开关量输出**

函数原型：

Visual C++:

```
BOOL SetDeviceDO_P2C (HANDLE hDevice,
                      BYTE bDIOSts[8])
```

Visual Basic:

```
Declare Function SetDeviceDO_P2C Lib "PCI2321_32" (_
                                    ByVal hDevice As Long,_
                                    ByRef bDIOsts As Byte) As Boolean
```

LabView：

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO40～DIO47 输出开关量置成相应的状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDIOSts 八路开关量输出状态的参数结构，共有 8 个成员变量，分别对应于DIO40～DIO47 路开关量输出状态位。比如置 bDIOSts[0]为“1”则使 0 通道处于“开”状态，若为“0”则置 0 通道为“关”状态。其他同理。请注意，在实际执行这个函数之前，必须对这个参数结构的DIO40 至DIO47 共 8 个成员变量赋初值，其值必须为“1”或“0”。具体定义请参考《DO数字开关量输出参数介绍》。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice EnableStsP2DIO GetDeviceDI_P2C ReleaseDevice

第五节、计数器控制函数原型说明

♦ **设置计数器初值**

函数原型：

Visual C++:

```
BOOL SetDevCNTInitValue(HANDLE hDevice,
                        ULONG InitCntrVal)
```

Visual Basic:

```
Declare Function SetDevCNTInitValue Lib "PCI2321_32" (_
                                    ByVal hDevice As Long,_
```

ByVal InitCntrVal As Long) As Boolean

LabView：

请参考相关演示程序。

功能：设置计数器初值。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

InitCntrVal 32 位计数初值。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice SetDevCNTInitValue EnableDevCNT GetDevCNTVal ResetDevCNT ReleaseDevice

♦ **是否使能计数器**

函数原型：

Visual C++:

```
BOOL EnableDevCNT (HANDLE hDevice,
                   BOOL bEnalbe)
```

Visual Basic:

```
Declare Function EnableDevCNT Lib"PCI2321_32" (_
                                   ByVal hDevice As Long,_
                                   ByVal bEnalbe As Boolean) As Boolean
```

LabView：

请参考相关演示程序。

功能：是否使能计数器。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bEnalbe =TRUE：使能计数器，=FALSE：禁止计数器。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice SetDevCNTInitValue EnableDevCNT GetDevCNTVal ResetDevCNT ReleaseDevice

♦ **取得所有计数器的计数值**

函数原型：

Visual C++:

```
BOOL GetDevCNTVal (HANDLE hDevice,
                   PULONG pCntrValue)
```

Visual Basic:

```
Declare Function GetDevCNTVal Lib"PCI2321_32" (_
                                   ByVal hDevice As Long,_
                                   ByRef pCntrValue As Long) As Boolean
```

LabView：

请参考相关演示程序。

功能：取得所有计数器的计数值。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
pCntrValue 计数值。
返回值：若成功，返回 TRUE，否则返回 FALSE。
相关函数： CreateDevice SetDevCNTInitValue EnableDevCNT
GetDevCNTVal ResetDevCNT ReleaseDevice

- **计数器复位清零**

函数原型：

Visual C++:

```
BOOL ResetDevCNT (HANDLE hDevice)
```

Visual Basic:

```
Declare Function ResetDevCNT Lib"PCI2321_32" (ByVal hDevice As Long) As Boolean
```

LabView：

请参考相关演示程序。

功能：计数器复位清零。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
返回值：若成功，返回 TRUE，否则返回 FALSE。
相关函数： CreateDevice SetDevCNTInitValue EnableDevCNT
GetDevCNTVal ResetDevCNT ReleaseDevice

第六节、中断函数原型说明

- **初始化中断**

函数原型：

Visual C++:

```
BOOL InitDeviceInt (HANDLE hDevice,
                    HANDLE hEventInt,
                    LONG INTSelect,
                    LONG INTSource)
```

Visual Basic:

```
Declare Function InitDeviceInt Lib"PCI2321_32" (_
                                ByVal hDevice As Long,_
                                ByVal hEventInt As Long,_
                                ByVal INTSelect As Long, _
                                ByVal INTSource As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：它负责初始化设备对象的硬件中断的方式工作。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。

hEventInt中断事件对象句柄，它应由CreateSystemEvent函数创建。它被创建时是一个不发信号且自动复位的内核系统事件对象。当硬件中断发生，这个内核系统事件被触发。用户应在数据采集子线程中使用WaitForSingleObject这个Win32 函数来接管这个内核系统事件。当中断没有到来时，WaitForSingleObject将使所在线程进入睡眠状态，此时，它不同于程序轮询方式，它并不消耗CPU时间。当hEventInt事件被触发成发信号状态，那么WaitForSingleObject将唤醒所在线程，可以工作了，比如取FIFO中的数据、分析数据等，且复位该内核系统事件对象，使其处于不发信号状态，以便在取完FIFO数据等工作后，让所在线程再次进入睡眠状态。所以利用中断方式采集数据，其效率是最高的。

INTSelect 选择 INT1 或 INT2 为总中断源。

常量名	常量值	功能定义
PCI2321_SELECT_INT1	0x0000	总中断源为 INT1
PCI2321_SELECT_INT2	0x0001	总中断源为 INT2

INTSource INT1 或 INT2 的中断源选择。

当选择 INT1 作为总中断源时：

常量名	常量值	功能定义
PCI2321_INT1SRC_P1C0	0x0000	P1C0 下降沿产生中断
PCI2321_INT1SRC_P1C0P1C3	0x0001	P1C0 与 P1C3 口产生中断源
PCI2321_INT1SRC_EXCNT	0x0002	外部计数器中断源

当选择 INT2 作为总中断源时：

常量名	常量值	功能定义
PCI2321_INT2SRC_P1C2P2C3	0x0000	P2C0 或 P2C3 口产生中断
PCI2321_INT2SRC_P2C0	0x0001	P2C0 下降沿产生中断
PCI2321_INT2SRC_INCNT	0x0002	内部定时器中断源

返回值：若成功，返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数： CreateDevice InitDeviceInt SetCOSINTCH GetDeviceIntCount ReleaseDeviceInt ReleaseDevice

◆ **取得中断服务程序产生的次数**

函数原型：

Visual C++:

LONG GetDeviceIntCount (HANDLE hDevice)

Visual Basic:

Declare Function GetDeviceIntCount Lib "PCI2321_32" (ByVal hDevice As Long) As Long

LabVIEW:

请参考相关演示程序。

功能：在中断初始化后，用它取得中断服务程序产生的次数。

参数：hDevice 设备对象句柄，它应由CreateDevice创建。

返回值：若成功，返回取得中断服务程序产生的次数，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数： CreateDevice InitDeviceInt SetCOSINTCH GetDeviceIntCount ReleaseDeviceInt ReleaseDevice

◆ **释放中断资源**

函数原型：

Visual C++:

BOOL ReleaseDeviceInt (HANDLE hDevice)

Visual Basic:

Declare Function ReleaseDeviceInt Lib"PCI2321_32" (ByVal hDevice As Long) As Long

LabVIEW:

请参考相关演示程序。

功能：释放中断资源。

参数：hDevice 设备对象句柄，它应由CreateDevice创建。

返回值：若成功，返回TRUE，否则返回FALSE。用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数： CreateDevice　InitDeviceInt　SetCOSINTCH　GetDeviceIntCount　ReleaseDeviceInt　ReleaseDevice

第四章　上层用户函数接口应用实例

如果您想快速的了解驱动程序的使用方法和调用流程，以最短的时间建立自己的应用程序，那么我们强烈建议您参考相应的简易程序。此种程序属于工程级代码，可以直接打开不用作任何配置和代码修改即可编译通过，运行编译链接后的可执行程序，即可看到预期效果。

如果您想了解硬件的整体性能、精度、采样连续性等指标以及波形显示、数据存盘与分析、历史数据回放等功能，那么请参考高级演示程序。特别是许多不愿意编写任何程序代码的用户，您可以使用高级程序进行采集、显示、存盘等功能来满足您的要求。甚至可以用我们提供的专用转换程序将高级程序采集的存盘文件转换成相应格式，即可在 Excel、MatLab 第三方软件中分析数据（此类用户请最好选用通过 Visual C++制作的高级演示系统）。

第一节、简易程序演示说明

一、怎样使用 GetDeviceDI 函数进行开关量输入操作

其详细应用实例及正确代码请参考 Visual C++简易演示系统及源程序，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程(主要参考 PCI2321.h 和 Sys.cpp)。

[程序]\|**[阿尔泰测控演示系统]**\|**[PCI2321 48 路开关量输入输出卡]**\|**[Microsoft Visual C++]**\|**[**简易代码演示**]**\|**[**DI 开关量演示源程序**]**

其默认存放路径为：系统盘\ART\PCI2321\SAMPLES\VC\SIMPLE\DI

二、怎样使用 SetDeviceDO 函数进行开关量输出操作

其详细应用实例及正确代码请参考 Visual C++简易演示系统及源程序，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程(主要参考 PCI2321.h 和 Sys.cpp)。

[程序]\|**[阿尔泰测控演示系统]**\|**[PCI2321 48 路开关量输入输出卡]**\|**[Microsoft Visual C++]**\|**[**简易代码演示**]**\|**[**DO 开关量演示源程序**]**

其默认存放路径为：系统盘\ART\PCI2321\SAMPLES\VC\SIMPLE\DO

第二节、高级程序演示说明

高级程序演示了本设备的所有功能，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程(主要参考 PCI2321.h 和 Sys.cpp)。

[程序]\|**[阿尔泰测控演示系统]**\|**[PCI2321 48 路开关量输入输出卡]**\|**[Microsoft Visual C++]**\|**[**高级代码演示**]**

其默认存放路径为：系统盘\ART\PCI2321\SAMPLES\VC\ADVANCED

其他语言的演示可以用上面类似的方法找到。

第五章 公共接口函数介绍

这部分函数不参与本设备的实际操作，它只是为您编写数据采集与处理程序时的有力手段，使您编写应用程序更容易，使您的应用程序更高效。

第一节、公用接口函数总列表（每个函数省略了前缀“PCI2321_”）

函数名	函数功能	备注
① PCI 总线内存映射寄存器操作函数		
GetDeviceBar	取得指定的指定设备寄存器组 BAR 地址	底层用户
GetDevVersion	获取设备固件及程序版本	底层用户
② ISA 总线 I/O 端口操作函数		
WritePortByte	以字节(8Bit)方式写 I/O 端口	用户程序操作端口
WritePortWord	以字(16Bit)方式写 I/O 端口	用户程序操作端口
WritePortULong	以无符号双字(32Bit)方式写 I/O 端口	用户程序操作端口
ReadPortByte	以字节(8Bit)方式读 I/O 端口	用户程序操作端口
ReadPortWord	以字(16Bit)方式读 I/O 端口	用户程序操作端口
ReadPortULong	以无符号双字(32Bit)方式读 I/O 端口	用户程序操作端口
③ 创建 Visual Basic 子线程，线程数量可达 32 个以上		
CreateSystemEvent	创建系统内核事件对象	用于线程同步或中断
ReleaseSystemEvent	释放系统内核事件对象	用于线程同步或中断

第二节、PCI 内存映射寄存器操作函数原型说明

◆ **取得指定的指定设备寄存器组 BAR 地址**

函数原型：

Visual C++:

```
BOOL GetDeviceBar ( HANDLE hDevice,
                    __int64 pbPCIBar[6])
```

Visual Basic:

```
Declare Function GetDeviceBar Lib "PCI2321_32" (_
                                    ByVal hDevice As Long, _
                                    ByRef pulPCIBar As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：取得指定的指定设备寄存器组 BAR 地址。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pulPCIBar 返回 PCI BAR 所有地址。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice ReleaseDevice

♦ **获取设备固件及程序版本**

函数原型：

Visual C++:

```
BOOL GetDevVersion ( HANDLE hDevice,
                     PULONG pulFmwVersion,
                     PULONG pulDriverVersion)
```

Visual Basic:

```
 Declare Function GetDevVersion Lib"PCI2321_32" (_
                                   ByVal hDevice As Long,_
                                   ByRef pulFmwVersion As Long,_
                                   ByRef pulDriverVersion As Long) As Boolean
```

LabVIEW:

请参见相关演示程序。

功能：获取设备固件及程序版本。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pulFmwVersion 指针参数，用于取得固件版本。

pulDriverVersion 指针参数，用于取得驱动版本。

返回值：如果执行成功，则返回 TRUE，否则会返回 FALSE。

相关函数： CreateDevice ReleaseDevice

第三节、I0 端口读写函数原型说明

注意：若您想在 WIN2K 系统的 User 模式中直接访问 I/O 端口，那么您可以安装光盘中 ISA\CommUser 目录下的公用驱动，然后调用其中的 WritePortByteEx 或 ReadPortByteEx 等有“Ex”后缀的函数即可。

♦ **以单字节(8Bit)方式写 I/O 端口**

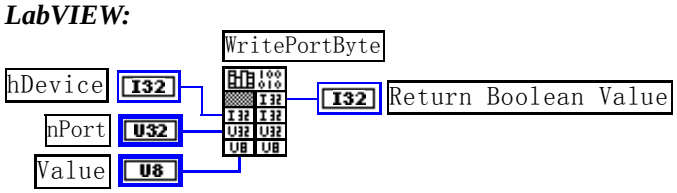
Visual C++:

```
BOOL WritePortByte (HANDLE hDevice,
                    __int64 pbPort,
                    BYTE Value)
```

Visual Basic:

```
Declare Function WritePortByte Lib "PCI2321_32" ( ByVal hDevice As Long, _
                                                  ByVal pbPort As Long, _
                                                  ByVal Value As Byte) As Boolean
```

LabVIEW:



功能：以单字节(8Bit)方式写 I/O 端口。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pbPort 设备的 I/O 端口号。

Value 写入由 pbPort 指定端口的值。

返回值：若成功，返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码。

相关函数： CreateDevice WritePortByte WritePortWord
WritePortULong ReadPortByte ReadPortWord

◆ **以双字(16Bit)方式写 I/O 端口**

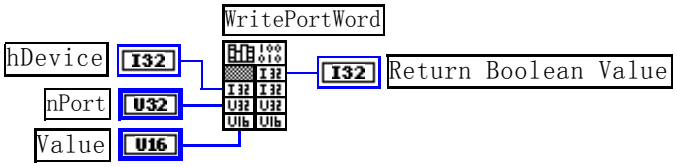
Visual C++:

```
BOOL WritePortWord (HANDLE hDevice,
                    __int64 pbPort,
                    WORD Value)
```

Visual Basic:

```
Declare Function WritePortWord Lib "PCI2321_32" (ByVal hDevice As Long, _
                                                 ByVal pbPort As Long, _
                                                 ByVal Value As Integer) As Boolean
```

LabVIEW:



功能：以双字(16Bit)方式写 I/O 端口。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pbPort 设备的 I/O 端口号。

Value　写入由 pbPort 指定端口的值。

返回值：若成功，返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码。

相关函数： CreateDevice　WritePortByte　WritePortWord　WritePortULong　ReadPortByte　ReadPortWord

◆ **以四字节(32Bit)方式写 I/O 端口**

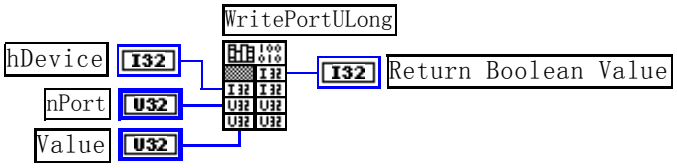
Visual C++:

```
BOOL WritePortULong(HANDLE hDevice,
                    __int64 pbPort,
                    ULONG Value)
```

Visual Basic:

```
Declare Function WritePortULong Lib "PCI2321_32" (ByVal hDevice As Long, _
                                                  ByVal pbPort As Long, _
                                                  ByVal Value As Long ) As Boolean
```

LabVIEW:



功能：以四字节(32Bit)方式写 I/O 端口。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pbPort 设备的 I/O 端口号。

Value　写入由 pbPort 指定端口的值。

返回值：若成功，返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码。

相关函数： CreateDevice　WritePortByte　WritePortWord　WritePortULong　ReadPortByte　ReadPortWord

◆ **以单字节(8Bit)方式读 I/O 端口**

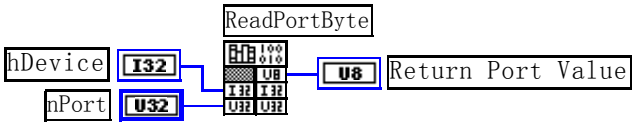
Visual C++:

```
BYTE ReadPortByte( HANDLE hDevice,
                   __int64 pbPort)
```

Visual Basic:

```
Declare Function ReadPortByte Lib "PCI2321_32" (ByVal hDevice As Long, _
                                                ByVal pbPort As Long ) As Byte
```

LabVIEW:



功能：以单字节(8Bit)方式读 I/O 端口。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
pbPort 设备的 I/O 端口号。
返回值：返回由 pbPort 指定的端口的值。
相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

- **以双字节(16Bit)方式读 I/O 端口**

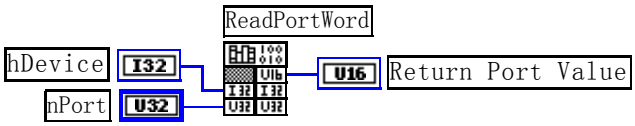
Visual C++:

```
WORD ReadPortWord(HANDLE hDevice,
                  __int64 pbPort)
```

Visual Basic:

```
Declare Function ReadPortWord Lib "PCI2321_32" ( ByVal hDevice As Long, _
                                                 ByVal pbPort As Long ) As Integer
```

LabVIEW:



功能：以双字节(16Bit)方式读 I/O 端口。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
pbPort 设备的 I/O 端口号。
返回值：返回由 pbPort 指定的端口的值。
相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

- **以四字节(32Bit)方式读 I/O 端口**

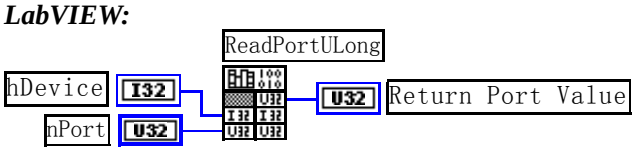
Visual C++:

```
ULONG ReadPortULong(HANDLE hDevice,
                    __int64 pbPort)
```

Visual Basic:

```
Declare Function ReadPortULong Lib "PCI2321_32" ( ByVal hDevice As Long, _
                                                  ByVal pbPort As Long ) As Long
```

LabVIEW:



功能：以四字节(32Bit)方式读 I/O 端口。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
pbPort 设备的 I/O 端口号。
返回值：返回由 pbPort 指定端口的值。
相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

第四节、线程操作函数原型说明

（如果您的 VB6.0 中线程无法正常运行，可能是 VB6.0 语言本身的问题，请选用 VB5.0）

♦ 创建内核系统事件

函数原型：

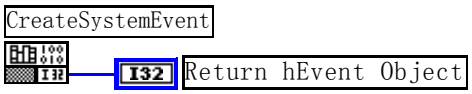
Visual C++:

HANDLE CreateSystemEvent(void)

Visual Basic:

Declare Function CreateSystemEvent Lib "PCI2321_32" () As Long

LabVIEW:



功能：创建系统内核事件对象，它将被用于中断事件响应或数据采集线程同步事件。

参数：无任何参数。

返回值：若成功，返回系统内核事件对象句柄，否则返回－1(或 INVALID_HANDLE_VALUE)。

♦ 释放内核系统事件

函数原型：

Visual C++:

BOOL ReleaseSystemEvent(HANDLE hEvent)

Visual Basic:

Declare Function ReleaseSystemEvent Lib"PCI2321_32" (ByVal hEvent As Long) As Boolean

LabVIEW:

请参见相关演示程序。

功能：释放系统内核事件对象。

参数：hEvent　被释放的内核事件对象。它应由CreateSystemEvent成功创建的对象。

返回值：若成功，则返回 TRUE。