PCI2322
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 PCI2322Inst.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 PCI2322_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bDOISts[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“PCI2322_”）

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 P3 口开关量函数		
EnableStsP3DIO	设置开关量输入或输出状态	上层用户
GetDeviceDI_P3A	DIO48～DIO55 开关输入函数	上层用户
SetDeviceDO_P3A	DIO48～DIO55 开关输出函数	上层用户
GetDeviceDI_P3B	DIO56～DIO63 开关输入函数	上层用户
SetDeviceDO_P3B	DIO56～DIO63 开关输出函数	上层用户
GetDeviceDI_P3C	DIO64～DIO71 开关输入函数	上层用户
SetDeviceDO_P3C	DIO64～DIO71 开关输出函数	上层用户
⑤ P4 口开关量函数		
EnableStsP4DIO	设置开关量输入或输出状态	上层用户
GetDeviceDI_P4A	DIO72～DIO79 开关输入函数	上层用户
SetDeviceDO_P4A	DIO72～DIO79 开关输出函数	上层用户
GetDeviceDI_P4B	DIO80～DIO87 开关输入函数	上层用户
SetDeviceDO_P4B	DIO80～DIO87 开关输出函数	上层用户
GetDeviceDI_P4C	DIO88～DIO95 开关输入函数	上层用户

SetDeviceDO_P4C	DIO88～DIO95 开关输出函数	上层用户
⑥ 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
⑦ 中断函数		
InitDeviceInt	初始化中断	上层用户
SetCOSINTCH	当中断源为 COS 时设置产生 COS 中断的通道	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++ & C++Builder：

首先将 PCI2322.h 和 PCI2322.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(PCI2322.lib)加入到您的工程中。

#include “PCI2322.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 PCI2322.h 和 PCI2322.lib 两个文件均可在演示程序文件夹下面找到。

Visual Basic:

首先将 PCI2322.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 PCI2322.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。

Delphi:

首先将 PCI2322.Pas 驱动模块头文件从 Delphi 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 Delphi 工程中。其方法是选择 Delphi 编程环境中的 View 菜单，执行其中的"Project Manager"命令，在弹出的对话中选择*.exe 项目，再单击鼠标右键，最后 Add 指令，即可将 PCI2322.Pas 单元模块文件加入到工程中。或者在 Delphi 的编程环境中的 Project 菜单中，执行 Add To Project 命令，然后选择*.Pas 文件类型也能实现单元模块文件的添加。最后请在使用驱动程序接口的源程序文件中的头部的 Uses 关键字后面的项目中加入：“PCI2322”。如：

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  PCI2322;  // 注意：  在此加入驱动程序接口单元 PCI2322
```

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++ & C++Builder：

```
HANDLE CreateDevice (int DeviceID = 0)
```

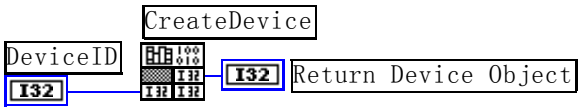
Visual Basic：

```
Declare Function CreateDevice Lib "PCI2322" (Optional ByVal DeviceID As Integer = 0) As Long
```

Delphi：

```
Function CreateDevice(DeviceID : Integer = 0) : Integer;
                    StdCall; External 'PCI2322' Name ' CreateDevice ';
```

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 PCI2322 设备的总数量

函数原型：

Visual C++ & C++Builder:

```
int GetDeviceCount (HANDLE hDevice)
```

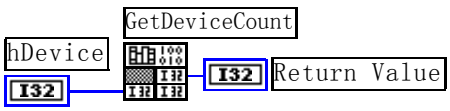
Visual Basic:

```
Declare Function GetDeviceCount Lib "PCI2322" (ByVal hDevice As Long ) As Integer
```

Delphi：

```
Function GetDeviceCount (hDevice : Integer) : Integer;
                    StdCall; External 'PCI2322' Name ' GetDeviceCount ';
```

LabVIEW：



功能：取得 PCI2322 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 PCI2322 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++ & C++Builder:

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

Visual Basic:

```
Declare Function GetDeviceCurrentID Lib "PCI2322" (ByVal hDevice As Long,_
                                                   ByRef DeviceLgcID As Long,_
                                                   ByRef DevicePhysIDAs Long ) As Boolean
```

Delphi：

```
Function GetDeviceCurrentID (hDevice : Integer;
                             DeviceLgcID : Pointer;
                             DevicePhysID : Pointer) : Boolean;
                             StdCall; External 'PCI2322' Name ' GetDeviceCurrentID ';
```

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 PCI2322 设备各种配置信息**

函数原型：

Visual C++ & C++Builder:

```
BOOL ListDeviceDlg (HANDLE hDevice)
```

Visual Basic:

```
Declare Function ListDeviceDlg Lib "PCI2322" (ByVal hDevice As Long ) As Boolean
```

Delphi：

```
Function ListDeviceDlg (hDevice : Integer) : Boolean;
                        StdCall; External 'PCI2322' Name ' ListDeviceDlg ';
```

LabVIEW：

请参考相关演示程序。

功能：列表系统中 PCI2322 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 PCI2322 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++ & C++Builder:

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

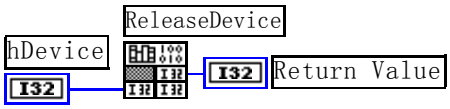
Visual Basic:

```
Declare Function ReleaseDevice Lib "PCI2322" (ByVal hDevice As Long ) As Boolean
```

Delphi：

```
Function ReleaseDevice(hDevice : Integer) : Boolean;
                    StdCall; External 'PCI2322' Name ' ReleaseDevice ';
```

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++ & C++Builder:

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

Visual Basic:

```
Declare Function EnableStsP1DIO Lib "PCI2322" (ByVal hDevice As Long, _
                                               ByVal bExtTrig As Boolean, _
                                               ByVal bP1Sts (0 to 2) As Boolean) As Boolean
```

Delphi:

```
Function EnableStsP1DIO(hDevice : Integer;
                    bExtTrig : Boolean;
                    bP1Sts : Pointer):Boolean;
                    StdCall; External 'PCI2322' Name ' EnableStsP1DIO ';
```

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++ & C++Builder:

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

Visual Basic:

```
Declare Function GetDeviceDI_P1A Lib "PCI2322" (ByVal hDevice As Long, _
                                                ByVal bDOISts (0 to 7 ) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI_P1A ( hDevice : Integer;
                           bDOISts : Pointer):Boolean;
                           StdCall; External 'PCI2322' Name ' GetDeviceDI_P1A ';
```

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。

bDOISts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bDOISts [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 bDOISts[x]中的值有效；否则返回 FALSE，其 bDOISts[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++ & C++Builder:

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

Visual Basic:

```
Declare Function SetDeviceDO_P1A Lib "PCI2322" ( ByVal hDevice As Long, _
                                                 ByVal bDOISts(0 to 7 )As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO_P1A (hDevice : Integer;
                          bDOISts : Pointer):Boolean;
                          StdCall; External 'PCI2322' Name ' SetDeviceDO_P1A ';
```

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。

bDOISts 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bDOISts[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++ & C++Builder:

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

Visual Basic:

```
Declare Function GetDeviceDI_P1B Lib "PCI2322" (ByVal hDevice As Long, _
                                                ByVal bDOISts(0 to 7 ) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI_P1B ( hDevice : Integer;
                           bDOISts : Pointer):Boolean;
                           StdCall; External 'PCI2322' Name ' GetDeviceDI_P1B ';
```

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。

bDOISts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bDOISts[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 bDOISts[x]中的值有效；否则返回 FALSE，其 bDOISts[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++ & C++Builder:

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

Visual Basic:

```
Declare Function SetDeviceDO_P1B Lib "PCI2322" ( ByVal hDevice As Long, _
                                                 ByVal bDOISts(0 to 7 )As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO_P1B (hDevice : Integer;
                          bDOISts : Pointer):Boolean;
                          StdCall; External 'PCI2322' Name ' SetDeviceDO_P1B ';
```

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。

bDOISts 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bDOISts[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++ & C++Builder:

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

Visual Basic:

```
Declare Function GetDeviceDI_P1C Lib "PCI2322" (ByVal hDevice As Long, _
                                                ByVal bDOISts(0 to 7 ) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI_P1C ( hDevice : Integer;
                           bDOISts : Pointer):Boolean;
                           StdCall; External 'PCI2322' Name ' GetDeviceDI_P1C ';
```

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。

bDOISts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bDOISts[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 bDOISts[x]中的值有效；否则返回 FALSE，其 bDOISts[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++ & C++Builder:

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

Visual Basic:

```
Declare Function SetDeviceDO_P1C Lib "PCI2322" ( ByVal hDevice As Long, _
                                                 ByVal bDOISts(0 to 7 )As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO_P1C (hDevice : Integer;
                          bDOISts : Pointer):Boolean;
                          StdCall; External 'PCI2322' Name ' SetDeviceDO_P1C ';
```

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。

bDOISts 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bDOISts[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++ & C++Builder:

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

Visual Basic:

```
Declare Function EnableStsP2DIO Lib "PCI2322" (ByVal hDevice As Long, _
                                               ByVal bExtTrig As Boolean, _
                                               ByVal bP2Sts (0 to 2) As Boolean) As Boolean
```

Delphi:

```
Function EnableStsP2DIO(hDevice : Integer;
                        bExtTrig : Boolean;
                        bP2Sts : Pointer):Boolean;
                        StdCall; External 'PCI2322' Name ' EnableStsP2DIO ';
```

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++ & C++Builder:

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

Visual Basic:

```
Declare Function GetDeviceDI_P2A Lib "PCI2322" (ByVal hDevice As Long, _
                                                ByVal bDOISts (0 to 7 ) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI_P2A ( hDevice : Integer;
                           bDOISts : Pointer):Boolean;
                           StdCall; External 'PCI2322' Name ' GetDeviceDI_P2A ';
```

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。

bDOISts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bDOISts [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 bDOISts[x]中的值有效；否则返回 FALSE，其 bDOISts[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++ & C++Builder:

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

Visual Basic:

```
Declare Function SetDeviceDO_P2A Lib "PCI2322" ( ByVal hDevice As Long, _
                                                 ByVal bDOISts(0 to 7 )As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO_P2A (hDevice : Integer;
                          bDOISts : Pointer):Boolean;
                          StdCall; External 'PCI2322' Name ' SetDeviceDO_P2A ';
```

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。

bDOISts 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bDOISts[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++ & C++Builder:

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

Visual Basic:

```
Declare Function GetDeviceDI_P2B Lib "PCI2322" (ByVal hDevice As Long, _
                                                ByVal bDOISts(0 to 7 ) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI_P2B ( hDevice : Integer;
                           bDOISts : Pointer):Boolean;
                           StdCall; External 'PCI2322' Name ' GetDeviceDI_P2B ';
```

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。

bDOISts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bDOISts[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 bDOISts[x]中的值有效；否则返回 FALSE，其 bDOISts[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++ & C++Builder:

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

Visual Basic:

```
Declare Function SetDeviceDO_P2BLib "PCI2322" (ByVal hDevice As Long, _
                                               ByVal bDOISts(0 to 7 )As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO_P2B (hDevice : Integer;
```

bDOISts : Pointer):Boolean;
StdCall; External 'PCI2322' Name ' SetDeviceDO_P2B';

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。

bDOISts 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bDOISts[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++ & C++Builder:

BOOL GetDeviceDI_P2C (HANDLE hDevice,
BYTE bDOISts[8])

Visual Basic:

Declare Function GetDeviceDI_P2C Lib "PCI2322" (ByVal hDevice As Long, _
ByVal bDOISts(0 to 7) As Byte) As Boolean

Delphi:

Function GetDeviceDI_P2C (hDevice : Integer;
bDOISts : Pointer):Boolean;
StdCall; External 'PCI2322' Name ' GetDeviceDI_P2C ';

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。

bDOISts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bDOISts[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 bDOISts[x]中的值有效；否则返回 FALSE，其 bDOISts[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++ & C++Builder:

BOOL SetDeviceDO_P2C (HANDLE hDevice,

```
                                BYTE bDOISts[8])
```

Visual Basic:

```
Declare Function SetDeviceDO_P2C Lib "PCI2322" ( ByVal hDevice As Long, _
                                                 ByVal bDOISts(0 to 7 )As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO_P2C (hDevice : Integer;
                          bDOISts : Pointer):Boolean;
                          StdCall; External 'PCI2322' Name ' SetDeviceDO_P2C ';
```

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。

bDOISts 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bDOISts[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、P3 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++ & C++Builder:

```
BOOL EnableStsP3DIO (HANDLE hDevice,
                     BOOL bExtTrig,
                     BOOL bP3Sts[3])
```

Visual Basic:

```
Declare Function EnableStsP3DIO Lib "PCI2322" (ByVal hDevice As Long, _
                                               ByVal bExtTrig As Boolean, _
                                               ByVal bP3Sts (0 to 2) As Boolean) As Boolean
```

Delphi:

```
Function EnableStsP3DIO(hDevice : Integer;
                        bExtTrig : Boolean;
                        bP3Sts : Pointer):Boolean;
                        StdCall; External 'PCI2322' Name ' EnableStsP3DIO ';
```

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。

bP3Sts bP3Sts[0]～bP3Sts[2]分别控制 P3A、P3B、P3C 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P3A口操作

♦ P3A 口 DIO48～DIO55 八路开关量输入

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDeviceDI_P3A (HANDLE hDevice,
                      BYTE bDOISts [8])
```

Visual Basic:

```
Declare Function GetDeviceDI_P3A Lib "PCI2322" (ByVal hDevice As Long, _
                                                ByVal bDOISts (0 to 7 ) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI_P3A ( hDevice : Integer;
                           bDOISts : Pointer):Boolean;
                           StdCall; External 'PCI2322' Name ' GetDeviceDI_P3A ';
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO48～DIO55 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。

bDOISts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DIO48～DIO55 路开关量输入状态位。如果bDOISts [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 bDOISts[x]中的值有效；否则返回 FALSE，其 bDOISts[x]中的值无效。

相关函数： CreateDevice EnableStsP3DIO SetDeviceDO_P3A ReleaseDevice

♦ P3A 口 DIO48～DIO55 八路开关量输出

函数原型：

Visual C++ & C++Builder:

```
BOOL SetDeviceDO_P3A (HANDLE hDevice,
                      BYTE bDOISts[8])
```

Visual Basic:

```
Declare Function SetDeviceDO_P3A Lib "PCI2322" ( ByVal hDevice As Long, _
                                                 ByVal bDOISts(0 to 7 )As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO_P3A (hDevice : Integer;
                          bDOISts : Pointer):Boolean;
                          StdCall; External 'PCI2322' Name ' SetDeviceDO_P3A ';
```

LabView：

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO48～DIO55 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。

bDOISts 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DIO48～DIO55 路开关量输出状态位。比如置bDOISts[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DIO48 至DIO55 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 EnableStsP3DIO GetDeviceDI_P3A ReleaseDevice

二、对P3B口操作

♦ P3B 口 DIO56～DIO63 八路开关量输入

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDeviceDI_P3B (HANDLE hDevice,
                      BYTE bDOISts[8])
```

Visual Basic:

```
Declare Function GetDeviceDI_P3B Lib "PCI2322" (ByVal hDevice As Long, _
                                                ByVal bDOISts(0 to 7 ) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI_P3B ( hDevice : Integer;
                           bDOISts : Pointer):Boolean;
                           StdCall; External 'PCI2322' Name ' GetDeviceDI_P3B ';
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO56～DIO63 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。

bDOISts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DIO56～DIO63 路开关量输入状态位。如果bDOISts[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 bDOISts[x]中的值有效；否则返回 FALSE，其 bDOISts[x]中的值无效。

相关函数： CreateDevice EnableStsP3DIO SetDeviceDO_P3B ReleaseDevice

♦ P3B 口 DIO8～DIO15 八路开关量输出

函数原型：

Visual C++ & C++Builder:

```
BOOL SetDeviceDO_P3B (HANDLE hDevice,
                      BYTE bDOISts[8])
```

Visual Basic:

```
Declare Function SetDeviceDO_P3B Lib "PCI2322" ( ByVal hDevice As Long, _
                                                 ByVal bDOISts(0 to 7 )As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO_P3B (hDevice : Integer;
                          bDOISts : Pointer):Boolean;
                          StdCall; External 'PCI2322' Name ' SetDeviceDO_P3B ';
```

LabView：

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO56～DIO63 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。

bDOISts 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DIO56～DIO63 路开关量输出状态位。比如置bDOISts[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DIO56 至DIO63 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 EnableStsP3DIO GetDeviceDI_P3B ReleaseDevice

三、对P3C口操作

♦ P3C 口 DIO64～DIO71 八路开关量输入

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDeviceDI_P3C (HANDLE hDevice,
                      BYTE bDOISts[8])
```

Visual Basic:

```
Declare Function GetDeviceDI_P3C Lib "PCI2322" (ByVal hDevice As Long, _
                                                ByVal bDOISts(0 to 7 ) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI_P3C ( hDevice : Integer;
                           bDOISts : Pointer):Boolean;
                           StdCall; External 'PCI2322' Name ' GetDeviceDI_P3C ';
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO64～DIO71 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。

bDOISts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DIO64～DIO71 路开关量输入状态位。如果bDOISts[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 bDOISts[x]中的值有效；否则返回 FALSE，其 bDOISts[x]中的值无效。

相关函数： CreateDevice EnableStsP3DIO SetDeviceDO_P3C ReleaseDevice

♦ P3C 口 DIO16～DIO23 八路开关量输出

函数原型：

Visual C++ & C++Builder:

```
BOOL SetDeviceDO_P3C (HANDLE hDevice,
                      BYTE bDOISts[8])
```

Visual Basic:

```
Declare Function SetDeviceDO_P3C Lib "PCI2322" ( ByVal hDevice As Long, _
                                                 ByVal bDOISts(0 to 7 )As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO_P3C (hDevice : Integer;
                          bDOISts : Pointer):Boolean;
                          StdCall; External 'PCI2322' Name ' SetDeviceDO_P3C ';
```

LabView：

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO64～DIO71 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。

bDOISts 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DIO64～DIO71 路开关量输出状态位。比如置bDOISts[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DIO64 至DIO71 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　EnableStsP3DIO　GetDeviceDI_P3C　ReleaseDevice

第六节、P4 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++ & C++Builder:

```
BOOL EnableStsP4DIO (HANDLE hDevice,
                     BOOL bExtTrig,
                     BOOL bP4Sts[3])
```

Visual Basic:

```
Declare Function EnableStsP4DIO Lib "PCI2322" (ByVal hDevice As Long, _
                                               ByVal bExtTrig As Boolean, _
                                               ByVal bP4Sts (0 to 2) As Boolean) As Boolean
```

Delphi:

```
Function EnableStsP4DIO(hDevice : Integer;
                        bExtTrig : Boolean;
                        bP4Sts : Pointer):Boolean;
                        StdCall; External 'PCI2322' Name ' EnableStsP4DIO ';
```

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。

bP4Sts bP4Sts[0]～bP4Sts[2]分别控制 P4A、P4B、P4C 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P4A口操作

♦ P4A 口 DIO72～DIO79 八路开关量输入

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDeviceDI_P4A (HANDLE hDevice,
                      BYTE bDOISts [8])
```

Visual Basic:

```
Declare Function GetDeviceDI_P4A Lib "PCI2322" (ByVal hDevice As Long, _
                                                ByVal bDOISts (0 to 7 ) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI_P4A ( hDevice : Integer;
                           bDOISts : Pointer):Boolean;
                           StdCall; External 'PCI2322' Name ' GetDeviceDI_P4A ';
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO72～DIO79 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。

bDOISts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DIO72～DIO79 路开关量输入状态位。如果bDOISts [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 bDOISts[x]中的值有效；否则返回 FALSE，其 bDOISts[x]中的值无效。

相关函数： CreateDevice EnableStsP4DIO SetDeviceDO_P4A ReleaseDevice

♦ P4A 口 DIO24～DIO31 八路开关量输出

函数原型：

Visual C++ & C++Builder:

```
BOOL SetDeviceDO_P4A (HANDLE hDevice,
                      BYTE bDOISts[8])
```

Visual Basic:

```
Declare Function SetDeviceDO_P4A Lib "PCI2322" ( ByVal hDevice As Long, _
                                                 ByVal bDOISts(0 to 7 )As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO_P4A (hDevice : Integer;
                          bDOISts : Pointer):Boolean;
                          StdCall; External 'PCI2322' Name ' SetDeviceDO_P4A ';
```

LabView：

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO72～DIO79 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。

bDOISts 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DIO72～DIO79 路开关量输出

状态位。比如置bDOISts[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DIO72 至DIO79 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 EnableStsP4DIO GetDeviceDI_P4A ReleaseDevice

二、对P4B口操作

♦ P4B 口 DIO80～DIO87 八路开关量输入

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDeviceDI_P4B (HANDLE hDevice,
                      BYTE bDOISts[8])
```

Visual Basic:

```
Declare Function GetDeviceDI_P4B Lib "PCI2322" (ByVal hDevice As Long, _
                                                ByVal bDOISts(0 to 7 ) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI_P4B ( hDevice : Integer;
                           bDOISts : Pointer):Boolean;
                           StdCall; External 'PCI2322' Name ' GetDeviceDI_P4B ';
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO80～DIO87 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。

bDOISts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DIO80～DIO87 路开关量输入状态位。如果bDOISts[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 bDOISts[x]中的值有效；否则返回 FALSE，其 bDOISts[x]中的值无效。

相关函数： CreateDevice EnableStsP4DIO SetDeviceDO_P4B ReleaseDevice

♦ P4B 口 DIO32～DIO39 八路开关量输出

函数原型：

Visual C++ & C++Builder:

```
BOOL SetDeviceDO_P4B (HANDLE hDevice,
                      BYTE bDOISts[8])
```

Visual Basic:

```
Declare Function SetDeviceDO_P4B Lib "PCI2322" (ByVal hDevice As Long, _
                                                ByVal bDOISts(0 to 7 )As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO_P4B (hDevice : Integer;
                          bDOISts : Pointer):Boolean;
                          StdCall; External 'PCI2322' Name ' SetDeviceDO_P4B';
```

LabView：

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO80～DIO87 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。

bDOISts 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DIO80～DIO87 路开关量输出状态位。比如置bDOISts[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DIO80 至DIO87 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 EnableStsP4DIO GetDeviceDI_P4B ReleaseDevice

三、对P4C口操作

♦ P4C 口 DIO88～DIO95 八路开关量输入

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDeviceDI_P4C (HANDLE hDevice,
                      BYTE bDOISts[8])
```

Visual Basic:

```
Declare Function GetDeviceDI_P4C Lib "PCI2322" (ByVal hDevice As Long, _
                                                ByVal bDOISts(0 to 7 ) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI_P4C ( hDevice : Integer;
                           bDOISts : Pointer):Boolean;
                           StdCall; External 'PCI2322' Name ' GetDeviceDI_P4C ';
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO88～DIO95 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。

bDOISts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DIO88～DIO95 路开关量输入状态位。如果bDOISts[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 bDOISts[x]中的值有效；否则返回 FALSE，其 bDOISts[x]中的值无效。

相关函数： CreateDevice EnableStsP4DIO SetDeviceDO_P4C ReleaseDevice

♦ P4C 口 DIO40～DIO47 八路开关量输出

函数原型：

Visual C++ & C++Builder:

```
BOOL SetDeviceDO_P4C (HANDLE hDevice,
                      BYTE bDOISts[8])
```

Visual Basic:

```
Declare Function SetDeviceDO_P4C Lib "PCI2322" ( ByVal hDevice As Long, _
```

ByVal bDOISts(0 to 7)As Byte) As Boolean

Delphi:

```
Function SetDeviceDO_P4C (hDevice : Integer;
                          bDOISts : Pointer):Boolean;
                          StdCall; External 'PCI2322' Name ' SetDeviceDO_P4C ';
```

LabView：

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO88～DIO95 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。

bDOISts 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DIO88～DIO95 路开关量输出状态位。比如置bDOISts[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DIO88 至DIO95 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 EnableStsP4DIO GetDeviceDI_P4C ReleaseDevice

❖ **以上函数调用一般顺序**

① CreateDevice

② SetDeviceDO_PX（或 GetDeviceDI_PX，当然这两个函数也可同时进行）

③ ReleaseDevice

用户可以反复执行第②步，以进行数字 I/O 的输入输出。

第七节、计数器控制函数原型说明

♦ **设置计数器初值**

函数原型：

Visual C++ & C++Builder:

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

Visual Basic:

```
Declare Function SetDevCNTInitValue Lib "PCI2322" (ByVal hDevice As Long, _
                                                   ByVal InitCntrVal As Long) As Boolean
```

Delphi:

```
Function SetDevCNTInitValue(hDevice : Integer;
                            InitCntrVal : LongWord):Boolean;
                            StdCall; External 'PCI2322' Name ' SetDevCNTInitValue ';
```

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++ & C++Builder:

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

Visual Basic:

```
Declare Function EnableDevCNT Lib "PCI2322" (ByVal hDevice As Long, _
                                             ByVal bEnalbe As Boolean) As Boolean
```

Delphi:

```
Function EnableDevCNT (hDevice : Integer;
                       bEnalbe : Boolean):Boolean;
                       StdCall; External 'PCI2322' Name ' EnableDevCNT ';
```

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++ & C++Builder:

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

Visual Basic:

```
Declare Function GetDevCNTVal Lib "PCI2322" (ByVal hDevice As Long, _
                                             ByRef pCntrValue As Long) As Boolean
```

Delphi:

```
Function GetDevCNTVal (hDevice : Integer;
                       pCntrValue: Pointer):Boolean;
                       StdCall; External 'PCI2322' Name ' GetDevCNTVal ';
```

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++ & C++Builder:

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

Visual Basic:

```
Declare Function ResetDevCNT Lib "PCI2322" (ByVal hDevice As Long) As Boolean
```

Delphi:

```
Function ResetDevCNT (hDevice : Integer):Boolean;
                        StdCall; External 'PCI2322' Name ' ResetDevCNT ';
```

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

第八节、中断函数原型说明

- **初始化中断**

函数原型：

Visual C++ & C++Builder:

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

Visual Basic:

```
Declare Function InitDeviceInt Lib "PCI2322" ( ByVal hDevice As Long, _
                                               ByVal hEventInt As Long, _
                                               ByVal INTSelect As Long, _
                                               ByVal INTSource As Long) As Boolean
```

Delphi:

```
Function InitDeviceInt (hDevice : Integer;
                        hEventInt : Integer;
                        INTSelect : LongInt;
                        INTSource : LongInt) : Boolean;
                        StdCall; External 'PCI2322' Name ' InitDeviceInt ';
```

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
PCI2322_SELECT_INT1	0x0000	总中断源为 INT1
PCI2322_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
PCI2322_INT1SRC_COS	0x0000	COS--P1&P2 中断源
PCI2322_INT1SRC_P1C0P1C3	0x0001	P1C0 与 P1C3 口产生中断源
PCI2322_INT1SRC_P1C0	0x0002	P1C0 下降沿产生中断
PCI2322_INT1SRC_EXCNT	0x0003	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
PCI2322_INT2SRC_COS	0x0000	COS--P3&P4 中断源
PCI2322_INT2SRC_P1C2P2C3	0x0001	P2C0 或 P2C3 口产生中断
PCI2322_INT2SRC_P2C0	0x0002	P2C0 下降沿产生中断
PCI2322_INT2SRC_INCNT	0x0003	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

♦ 当中断源为 COS 时设置产生 COS 中断的通道

函数原型：

Visual C++ & C++Builder:

```
BOOL SetCOSINTCH(HANDLE hDevice,
                 BOOL bP1Enalbe,
                 BOOL bP2Enalbe,
                 BOOL bP3Enalbe,
                 BOOL bP4Enalbe)
```

Visual Basic:

```
Declare Function SetCOSINTCH Lib "PCI2322" (ByVal hDevice As Boolean, _
                                            ByVal bP1Enalbe As Boolean, _
                                            ByVal bP2Enalbe As Boolean, _
                                            ByVal bP3Enalbe As Boolean,_
                                            ByVal bP4Enalbe As Boolean) As Boolean
```

Delphi:

```
Function SetCOSINTCH (hDevice : Integer;
                      bP1Enalbe: Boolean;
                      bP2Enalbe: Boolean;
                      bP3Enalbe: Boolean;
                      bP4Enalbe: Boolean) : Boolean;
                      StdCall; External 'PCI2322' Name ' SetCOSINTCH ';
```

LabVIEW:

请参考相关演示程序。

功能：当中断源为 COS 时设置产生 COS 中断的通道。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bP1Enalbe P1 口产生 COS 中断使能。

bP2Enalbe P2 口产生 COS 中断使能。

bP3Enalbe P3 口产生 COS 中断使能。

bP4Enalbe P4 口产生 COS 中断使能。

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++ & C++Builder:

```
LONG GetDeviceIntCount (HANDLE hDevice)
```

Visual Basic:

```
Declare Function GetDeviceIntCount Lib "PCI2322" (ByVal hDevice As Long) As Long
```

Delphi:

```
Function GetDeviceIntCount (hDevice : Integer) : LongInt;
                            StdCall; External 'PCI2322' Name ' GetDeviceIntCount ';
```

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++ & C++Builder:

```
BOOL ReleaseDeviceInt (HANDLE hDevice)
```

Visual Basic:

```
Declare Function ReleaseDeviceInt Lib "PCI2322" (ByVal hDevice As Long) As Boolean
```

Delphi:

Function ReleaseDeviceInt (hDevice : Integer) : Boolean;
StdCall; External 'PCI2322' Name ' ReleaseDeviceInt ';

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 PCI2322.h 和 Sys.cpp)。

[程序]\|**[阿尔泰测控演示系统]**\|**[PCI2322 96 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\PCI2322\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 PCI2322.h 和 Sys.cpp)。

[程序]\|**[阿尔泰测控演示系统]**\|**[PCI2322 96 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\PCI2322\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 PCI2322.h 和 Sys.cpp)。

[程序]\|**[阿尔泰测控演示系统]**\|**[PCI2322 96 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\PCI2322\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“PCI2322_”）

函数名	函数功能	备注
① PCI 总线内存映射寄存器操作函数		
GetDeviceAddr	取得指定 PCI 设备寄存器操作基地址	底层用户
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
WriteRegisterByte	以字节(8Bit)方式写寄存器端口	底层用户
WriteRegisterWord	以字(16Bit)方式写寄存器端口	底层用户
WriteRegisterULong	以双字(32Bit)方式写寄存器端口	底层用户
ReadRegisterByte	以字节(8Bit)方式读寄存器端口	底层用户
ReadRegisterWord	以字(16Bit)方式读寄存器端口	底层用户
ReadRegisterULong	以双字(32Bit)方式读寄存器端口	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		
CreateVBThread	在 VB 环境中建立子线程对象	
TerminateVBThread	终止 VB 的子线程	
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	
④ 文件对象操作函数		
CreateFileObject	初始设备文件对象	
WriteFile	请求文件对象写用户数据到磁盘文件	
ReadFile	请求文件对象读数据到用户空间	
SetFileOffset	设置文件指针偏移	
GetFileLength	取得文件长度	
ReleaseFile	释放已有的文件对象	
GetDiskFreeBytes	取得指定磁盘的可用空间(字节)	适用于所有设备
⑤ 各种参数保存和读取函数		
SaveParaInt	保存整型参数到注册表	
LoadParaInt	从注册表中读取整型参数值	
SaveParaString	保存字符参数到注册表	
LoadParaString	从注册表中读取字符参数值	
⑥ 其他函数		
GetLastErrorEx	取得驱动函数错误信息	

RemoveLastErrorEx	移除指定函数的最后一次错误信息	
DelayTimeUs	高效高精度延时函数	

第二节、PCI 内存映射寄存器操作函数原型说明

◆ **取得指定内存映射寄存器的线性地址和物理地址**

函数原型：

Visual C++ & C++ Builder:

```
BOOL GetDeviceAddr( HANDLE hDevice,
                    PULONG LinearAddr,
                    PULONG PhysAddr,
                    int RegisterID = 0)
```

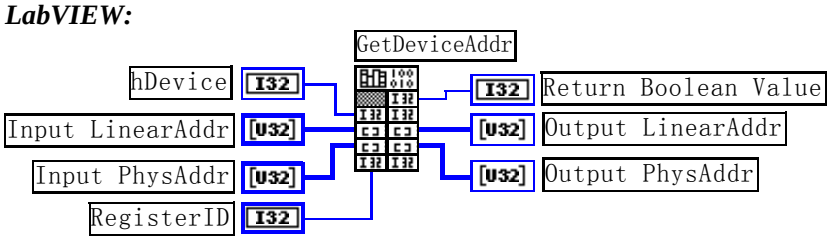
Visual Basic:

```
Declare Function GetDeviceAddr Lib "PCI2322" (ByVal hDevice As Long, _
                                              ByRef LinearAddr As Long, _
                                              ByRef PhysAddr As Long, _
                                              ByVal RegisterID As Integer = 0) As Boolean
```

Delphi:

```
Function GetDeviceAddr(hDevice : Integer;
                       LinearAddr : Pointer;
                       PhysAddr : Pointer;
                       RegisterID : Integer = 0) : Boolean;
                       StdCall; External 'PCI2322' Name ' GetDeviceAddr ';
```

LabVIEW:



功能：取得 PCI 设备指定的内存映射寄存器的线性地址。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

LinearAddr 指针参数，用于取得的映射寄存器指向的线性地址，RegisterID 指定的寄存器组属于 MEM 模式时该值不应为零，也就是说它可用于 WriteRegisterX 或 ReadRegisterX（X 代表 Byte、ULong、Word）等函数，以便于访问设备寄存器。它指明该设备位于系统空间的虚拟位置。但如果 RegisterID 指定的寄存器组属于 I/O 模式时该值通常为零，您不能通过以上函数访问设备。

PhysAddr 指针参数，用于取得的映射寄存器指向的物理地址，它指明该设备位于系统空间的物理位置。如果由 RegisterID 指定的寄存器组属于 I/O 模式，则可用于 WritePortX 或 ReadPortX（X 代表 Byte、ULong、Word）等函数，以便于访问设备寄存器。

RegisterID 指定映射寄存器的 ID 号，其取值范围为[0，5]，通常情况下，用户应使用 0 号映射寄存器，特殊情况下，我们为用户加以申明。

返回值：如果执行成功，则返回TRUE，它表明由RegisterID指定的映射寄存器的无符号 32 位线性地址和物理地址被正确返回，否则会返回FALSE，同时还要检查其LinearAddr和PhysAddr是否为 0，若为 0 则依然视为失败。用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数： CreateDevice ReleaseDevice

♦ **取得指定的指定设备寄存器组 BAR 地址**

函数原型：

Visual C++ & C++ Builder:

```
BOOL GetDeviceBar ( HANDLE hDevice,
                    ULONG    pulPCIBar[6])
```

Visual Basic:

```
Declare Function GetDeviceBar Lib "PCI2322" (ByVal hDevice As Long, _
                                             ByVal pulPCIBar (0 to 5) As Long) As Boolean
```

Delphi:

```
Function GetDeviceBar (hDevice : Integer;
                       pulPCIBar : Pointer) : Boolean;
                       StdCall; External 'PCI2322' Name ' GetDeviceBar';
```

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++ & C++ Builder:

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

Visual Basic:

```
Declare Function GetDevVersion Lib "PCI2322" (ByVal hDevice As Long, _
                                              ByRef pulFmwVersion As Long, _
                                              ByRef pulDriverVersion As Long) As Boolean
```

Delphi:

```
Function GetDevVersion (hDevice : Integer;
                        pulFmwVersion: Pointer;
                        pulDriverVersion: Pointer) : Boolean;
                        StdCall; External 'PCI2322' Name ' GetDevVersion ';
```

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

♦ **以单字节（即 8 位）方式写 PCI 内存映射寄存器的某个单元**

函数原型:

Visual C++ & C++ Builder:

```
BOOL WriteRegisterByte( HANDLE hDevice,
                        ULONG LinearAddr,
                        ULONG OffsetBytes,
                        BYTE Value)
```

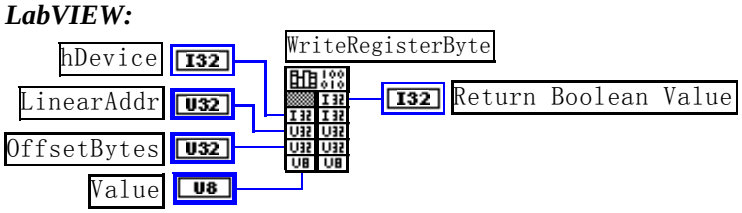
Visual Basic:

```
Declare Function WriteRegisterByte Lib "PCI2322" (ByVal hDevice As Long, _
                                                  ByVal LinearAddr As Long, _
                                                  ByVal OffsetBytes As Long, _
                                                  ByVal Value As Byte ) As Boolean
```

Delphi:

```
Function WriteRegisterByte( hDevice : Integer;
                            LinearAddr : LongWord;
                            OffsetBytes : LongWord;
                            Value : Byte) : Boolean;
                            StdCall; External 'PCI2322' Name ' WriteRegisterByte ';
```

LabVIEW:



功能：以单字节（即 8 位）方式写 PCI 内存映射寄存器。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

LinearAddr PCI设备内存映射寄存器的线性基地址，它的值应由GetDeviceAddr确定。

OffsetBytes 相对于LinearAddr线性基地址的偏移字节数，它与LinearAddr两个参数共同确定WriteRegisterByte函数所访问的映射寄存器的内存单元。

Value 输出 8 位整数。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数: CreateDevice WriteRegisterByte WriteRegisterWord
WriteRegisterULong ReadRegisterByte ReadRegisterWord
ReadRegisterULong ReleaseDevice

Visual C++ & C++ Builder 程序举例:

```
    :
HANDLE hDevice;
ULONG LinearAddr, PhysAddr, OffsetBytes;
hDevice = CreateDevice(0)
if (!GetDeviceAddr(hDevice, &LinearAddr,   &PhysAddr, 0) )
{
    AfxMessageBox “取得设备地址失败…”;
```

```
}
OffsetBytes = 100;     // 指定操作相对于线性基地址偏移 100 个字节数位置的单元
WriteRegisterByte(hDevice, LinearAddr, OffsetBytes, 0x20);   // 往指定映射寄存器单元写入 8 位的十六进制数据 20
ReleaseDevice( hDevice );   // 释放设备对象
    :
```

Visual Basic 程序举例:

```
    :
Dim hDevice As Long
Dim LinearAddr, PhysAddr, OffsetBytes As Long
hDevice = CreateDevice(0)
GetDeviceAddr( hDevice, LinearAddr, PhysAddr, 0)
OffsetBytes = 100
WriteRegisterByte( hDevice, LinearAddr, OffsetBytes, &H20)
ReleaseDevice(hDevice)
    :
```

♦ **以双字节（即 16 位）方式写 PCI 内存映射寄存器的某个单元**

函数原型:

Visual C++ & C++ Builder:

```
BOOL WriteRegisterWord(HANDLE hDevice,
                       ULONG LinearAddr,
                       ULONG OffsetBytes,
                       WORD Value)
```

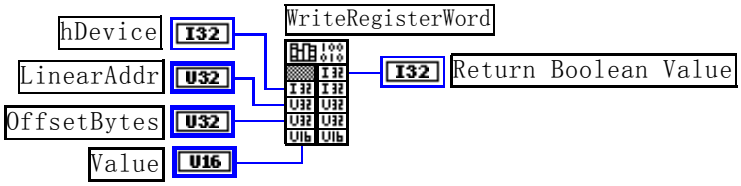
Visual Basic:

```
Declare Function WriteRegisterWord Lib "PCI2322" (ByVal hDevice As Long, _
                                                  ByVal LinearAddr As Long, _
                                                  ByVal OffsetBytes As Long, _
                                                  ByVal Value As Integer) As Boolean
```

Delphi:

```
Function WriteRegisterWord( hDevice : Integer;
                            LinearAddr : LongWord;
                            OffsetBytes : LongWord;
                            Value : Word) : Boolean;
                            StdCall; External 'PCI2322' Name ' WriteRegisterWord ';
```

LabVIEW:



功能：以双字节（即 16 位）方式写 PCI 内存映射寄存器。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

LinearAddr　PCI设备内存映射寄存器的线性基地址，它的值应由GetDeviceAddr确定。

OffsetBytes　相对于LinearAddr线性基地址的偏移字节数，它与LinearAddr两个参数共同确定WriteRegisterWord函数所访问的映射寄存器的内存单元。

Value 输出 16 位整型值。

返回值：无。

相关函数: CreateDevice WriteRegisterByte WriteRegisterWord WriteRegisterULong ReadRegisterByte ReadRegisterWord ReadRegisterULong ReleaseDevice

Visual C++ & C++ Builder 程序举例:

```
    :
HANDLE hDevice;
ULONG LinearAddr, PhysAddr, OffsetBytes;
hDevice = CreateDevice(0)
if (!GetDeviceAddr(hDevice, &LinearAddr,  &PhysAddr, 0) )
{
    AfxMessageBox "取得设备地址失败…";
}
OffsetBytes = 100;    // 指定操作相对于线性基地址偏移 100 个字节数位置的单元
WriteRegisterWord(hDevice, LinearAddr, OffsetBytes, 0x2000);  // 往指定映射寄存器单元写入 16 位的十六进制数据
ReleaseDevice( hDevice );  // 释放设备对象
    :
```

Visual Basic 程序举例:

```
    :
Dim hDevice As Long
Dim LinearAddr, PhysAddr, OffsetBytes As Long
hDevice = CreateDevice(0)
GetDeviceAddr( hDevice, LinearAddr, PhysAddr, 0)
OffsetBytes=100
WriteRegisterWord( hDevice, LinearAddr, OffsetBytes, &H2000)
ReleaseDevice(hDevice)
    :
```

♦ **以四字节（即 32 位）方式写 PCI 内存映射寄存器的某个单元**

函数原型:

Visual C++ & C++ Builder:

```
BOOL WriteRegisterULong( HANDLE hDevice,
                         ULONG LinearAddr,
                         ULONG OffsetBytes,
                         ULONG Value)
```

Visual Basic:

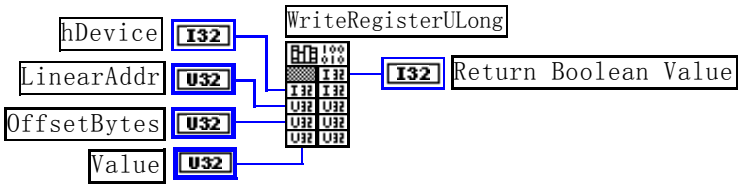
```
Declare Function WriteRegisterULong Lib "PCI2322" (ByVal hDevice As Long, _
                                                   ByVal LinearAddr As Long, _
                                                   ByVal OffsetBytes As Long, _
                                                   ByVal Value As Long) As Boolean
```

Delphi:

```
Function WriteRegisterULong(hDevice : Integer;
                            LinearAddr : LongWord;
                            OffsetBytes : LongWord;
```

Value : LongWord) : Boolean;
StdCall; External 'PCI2322' Name ' WriteRegisterULong ';

LabVIEW:



功能：以四字节（即 32 位）方式写 PCI 内存映射寄存器。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

LinearAddr PCI设备内存映射寄存器的线性基地址，它的值应由GetDeviceAddr确定。

OffsetBytes 相对于LinearAddr线性基地址的偏移字节数，它与LinearAddr两个参数共同确定WriteRegisterULong函数所访问的映射寄存器的内存单元。

Value 输出 32 位整型值。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数: CreateDevice WriteRegisterByte WriteRegisterWord WriteRegisterULong ReadRegisterByte ReadRegisterWord ReadRegisterULong ReleaseDevice

Visual C++ & C++ Builder 程序举例:

```
    :
HANDLE hDevice;
ULONG LinearAddr, PhysAddr, OffsetBytes;
hDevice = CreateDevice(0)
if (!GetDeviceAddr(hDevice, &LinearAddr,  &PhysAddr, 0) )
{
        AfxMessageBox "取得设备地址失败…";
}
OffsetBytes=100;      // 指定操作相对于线性基地址偏移 100 个字节数位置的单元
WriteRegisterULong(hDevice, LinearAddr, OffsetBytes, 0x20000000);  // 往指定映射寄存器单元写入 32 位的十六进制数据
ReleaseDevice( hDevice );  // 释放设备对象
    :
```

Visual Basic 程序举例:

```
    :
Dim hDevice As Long
Dim LinearAddr, PhysAddr, OffsetBytes As Long
hDevice = CreateDevice(0)
GetDeviceAddr( hDevice, LinearAddr, PhysAddr, 0)
OffsetBytes = 100
WriteRegisterULong( hDevice, LinearAddr, OffsetBytes, &H20000000)
ReleaseDevice(hDevice)
    :
```

- **以单字节（即 8 位）方式读 PCI 内存映射寄存器的某个单元**

函数原型:

Visual C++ & C++ Builder:

```
BYTE ReadRegisterByte( HANDLE hDevice,
                       ULONG LinearAddr,
                       ULONG OffsetBytes)
```

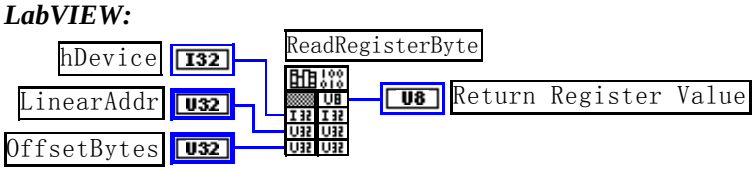
Visual Basic:

```
Declare Function ReadRegisterByte Lib "PCI2322" (ByVal hDevice As Long, _
                                                 ByVal LinearAddr As Long, _
                                                 ByVal OffsetBytes As Long) As Byte
```

Delphi:

```
Function ReadRegisterByte(hDevice : Integer;
                          LinearAddr : LongWord;
                          OffsetBytes : LongWord) : Byte;
                         StdCall; External 'PCI2322' Name ' ReadRegisterByte ';
```

LabVIEW:



功能：以单字节（即 8 位）方式读 PCI 内存映射寄存器的指定单元。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

LinearAddr　PCI设备内存映射寄存器的线性基地址，它的值应由GetDeviceAddr确定。

OffsetBytes　相对于LinearAddr线性基地址的偏移字节数，它与LinearAddr两个参数共同确定ReadRegisterByte函数所访问的映射寄存器的内存单元。

返回值：返回从指定内存映射寄存器单元所读取的 8 位数据。

相关函数: CreateDevice　WriteRegisterByte　WriteRegisterWord　WriteRegisterULong　ReadRegisterByte　ReadRegisterWord　ReadRegisterULong　ReleaseDevice

Visual C++ & C++ Builder 程序举例:

```
    :
HANDLE hDevice;
ULONG LinearAddr,   PhysAddr, OffsetBytes;
BYTE   Value;
hDevice = CreateDevice(0);   // 创建设备对象
GetDeviceAddr(hDevice, &LinearAddr, &PhysAddr, 0);    // 取得 PCI 设备 0 号映射寄存器的线性基地址
OffsetBytes = 100;    // 指定操作相对于线性基地址偏移 100 个字节数位置的单元
Value = ReadRegisterByte(hDevice, LinearAddr, OffsetBytes);   // 从指定映射寄存器单元读入 8 位数据
ReleaseDevice( hDevice );   // 释放设备对象
    :
```

Visual Basic 程序举例:

```
    :
Dim hDevice As Long
Dim LinearAddr, PhysAddr, OffsetBytes As Long
Dim Value As Byte
hDevice = CreateDevice(0)
```

```
GetDeviceAddr( hDevice, Linearddr, PhysAddr, 0)
OffsetBytes = 100
Value = ReadRegisterByte( hDevice, LinearAddr, OffsetBytes)
ReleaseDevice(hDevice)
    :
```

◆ **以双字节（即 16 位）方式读 PCI 内存映射寄存器的某个单元**

函数原型:

Visual C++ & C++ Builder:

```
WORD ReadRegisterWord( HANDLE hDevice,
                       ULONG LinearAddr,
                       ULONG OffsetBytes)
```

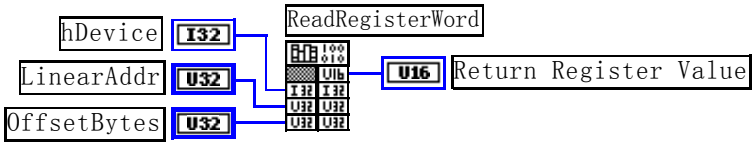
Visual Basic:

```
Declare Function ReadRegisterWord Lib "PCI2322" ( ByVal hDevice As Long, _
                                                  ByVal LinearAddr As Long, _
                                                  ByVal OffsetBytes As Long) As Integer
```

Delphi:

```
Function ReadRegisteWord(hDevice : Integer;
                         LinearAddr : LongWord;
                         OffsetBytes : LongWord) : Word;
                         StdCall; External 'PCI2322' Name ' ReadRegisterWord ';
```

LabVIEW:



功能：以双字节（即 16 位）方式读 PCI 内存映射寄存器的指定单元。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

LinearAddr PCI设备内存映射寄存器的线性基地址，它的值应由GetDeviceAddr确定。

OffsetBytes 相对于LinearAddr线性基地址的偏移字节数，它与LinearAddr两个参数共同确定ReadRegisterWord函数所访问的映射寄存器的内存单元。

返回值：返回从指定内存映射寄存器单元所读取的 16 位数据。

相关函数: CreateDevice WriteRegisterByte WriteRegisterWord WriteRegisterULong ReadRegisterByte ReadRegisterWord ReadRegisterULong ReleaseDevice

Visual C++ & C++ Builder 程序举例:

```
    :
HANDLE hDevice;
ULONG LinearAddr,   PhysAddr, OffsetBytes;
WORD Value;
hDevice = CreateDevice(0);   // 创建设备对象
GetDeviceAddr(hDevice, &LinearAddr, &PhysAddr, 0);    // 取得 PCI 设备 0 号映射寄存器的线性基地址
OffsetBytes = 100;    // 指定操作相对于线性基地址偏移 100 个字节数位置的单元
Value = ReadRegisterWord(hDevice, LinearAddr, OffsetBytes);   // 从指定映射寄存器单元读入 16 位数据
```

```
ReleaseDevice( hDevice );  // 释放设备对象
    :
```

Visual Basic 程序举例:

```
    :
Dim hDevice As Long
Dim LinearAddr, PhysAddr, OffsetBytes As Long
Dim Value As Word
hDevice = CreateDevice(0)
GetDeviceAddr( hDevice, Linearddr, PhysAddr, 0)
OffsetBytes = 100
Value = ReadRegisterWord( hDevice, LinearAddr, OffsetBytes)
ReleaseDevice(hDevice)
    :
```

◆ **以四字节（即 32 位）方式读 PCI 内存映射寄存器的某个单元**

函数原型:

Visual C++ & C++ Builder:

```
ULONG ReadRegisterULong( HANDLE hDevice,
                         ULONG LinearAddr,
                         ULONG OffsetBytes)
```

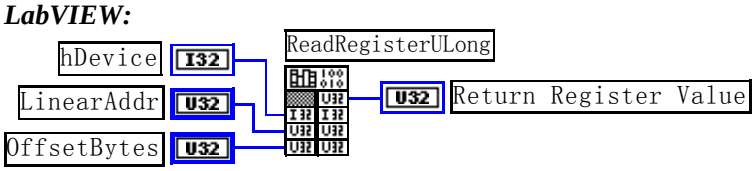
Visual Basic:

```
Declare Function ReadRegisterULong Lib "PCI2322" (ByVal hDevice As Long, _
                                                  ByVal LinearAddr As Long, _
                                                  ByVal OffsetBytes As Long) As Long
```

Delphi:

```
Function ReadRegisterULong(hDevice : Integer;
                           LinearAddr : LongWord;
                           OffsetBytes : LongWord) : LongWord;
                          StdCall; External 'PCI2322' Name ' ReadRegisterULong ';
```

LabVIEW:



功能：以四字节（即 32 位）方式读 PCI 内存映射寄存器的指定单元。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

LinearAddr　PCI设备内存映射寄存器的线性基地址，它的值应由GetDeviceAddr确定。

OffsetBytes　相对与LinearAddr线性基地址的偏移字节数，它与LinearAddr两个参数共同确定WriteRegisterULong函数所访问的映射寄存器的内存单元。

返回值：返回从指定内存映射寄存器单元所读取的 32 位数据。

相关函数:　CreateDevice　WriteRegisterByte　WriteRegisterWord　WriteRegisterULong　ReadRegisterByte　ReadRegisterWord　ReadRegisterULong　ReleaseDevice

Visual C++ & C++ Builder 程序举例:

```
    :
HANDLE hDevice;
ULONG LinearAddr,   PhysAddr, OffsetBytes;
ULONG   Value;
hDevice = CreateDevice(0);   // 创建设备对象
GetDeviceAddr(hDevice, &LinearAddr, &PhysAddr, 0);    // 取得 PCI 设备 0 号映射寄存器的线性基地址
OffsetBytes = 100;    // 指定操作相对于线性基地址偏移 100 个字节数位置的单元
Value = ReadRegisterULong(hDevice, LinearAddr, OffsetBytes);   // 从指定映射寄存器单元读入 32 位数据
ReleaseDevice( hDevice );   // 释放设备对象
    :
```

Visual Basic 程序举例:

```
    :
Dim hDevice As Long
Dim LinearAddr, PhysAddr, OffsetBytes As Long
Dim Value As Long
hDevice = CreateDevice(0)
GetDeviceAddr( hDevice, Linearddr, PhysAddr, 0)
OffsetBytes = 100
Value = ReadRegisterULong( hDevice, LinearAddr, OffsetBytes)
ReleaseDevice(hDevice)
    :
```

第三节、IO 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++ & C++ Builder:

```
BOOL WritePortByte (HANDLE hDevice,
                    UINT nPort,
                    BYTE Value)
```

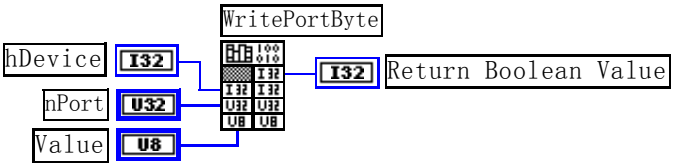
Visual Basic:

```
Declare Function WritePortByte Lib "PCI2322" ( ByVal hDevice As Long, _
                                               ByVal nPort As Long, _
                                               ByVal Value As Byte) As Boolean
```

Delphi:

```
Function WritePortByte(hDevice : Integer;
                       nPort : LongWord;
                       Value : Byte) : Boolean;
                       StdCall; External 'PCI2322' Name ' WritePortByte ';
```

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。

nPort 设备的 I/O 端口号。

Value 写入由 nPort 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++ & C++ Builder:

```
BOOL WritePortWord (HANDLE hDevice,
                    UINT nPort,
                    WORD Value)
```

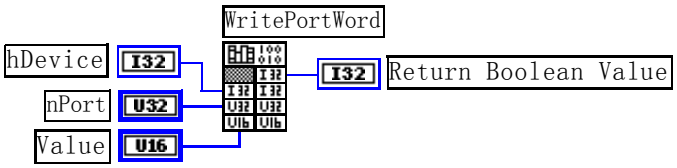
Visual Basic:

```
Declare Function WritePortWord Lib "PCI2322" (ByVal hDevice As Long, _
                                              ByVal nPort As Long, _
                                              ByVal Value As Integer) As Boolean
```

Delphi:

```
Function WritePortWord(hDevice : Integer;
                       nPort : LongWord;
                       Value : Word) : Boolean;
                       StdCall; External 'PCI2322' Name ' WritePortWord ';
```

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。

nPort 设备的 I/O 端口号。

Value 写入由 nPort 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++ & C++ Builder:

```
BOOL WritePortULong(HANDLE hDevice,
                    UINT nPort,
                    ULONG Value)
```

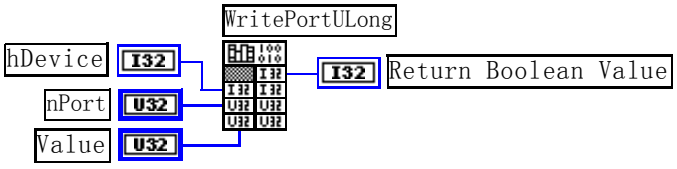
Visual Basic:

```
Declare Function WritePortULong Lib "PCI2322" (ByVal hDevice As Long, _
                                               ByVal nPort As Long, _
                                               ByVal Value As Long ) As Boolean
```

Delphi:

```
Function WritePortULong(hDevice : Integer;
                        nPort : LongWord;
                        Value : LongWord) : Boolean;
                        StdCall; External 'PCI2322' Name ' WritePortULong ';
```

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。

nPort　设备的 I/O 端口号。
Value　写入由 nPort 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++ & C++ Builder:

```
BYTE ReadPortByte( HANDLE hDevice,
                   UINT nPort)
```

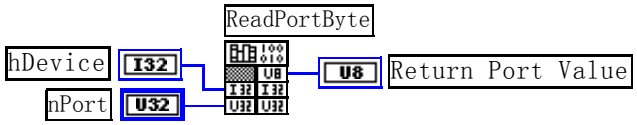
Visual Basic:

```
Declare Function ReadPortByte Lib "PCI2322" (ByVal hDevice As Long, _
                                             ByVal nPort As Long ) As Byte
```

Delphi:

```
Function ReadPortByte(hDevice : Integer;
                      nPort : LongWord) : Byte;
                      StdCall; External 'PCI2322' Name ' ReadPortByte ';
```

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。
nPort　设备的 I/O 端口号。
返回值：返回由 nPort 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++ & C++ Builder:

```
WORD ReadPortWord(HANDLE hDevice,
                  UINT nPort)
```

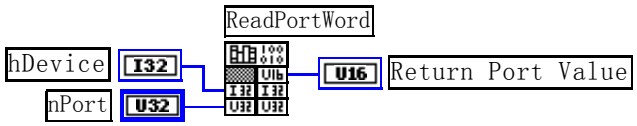
Visual Basic:

```
Declare Function ReadPortWord Lib "PCI2322" ( ByVal hDevice As Long, _
                                              ByVal nPort As Long ) As Integer
```

Delphi:

```
Function ReadPortWord(hDevice : Integer;
                      nPort : LongWord) : Word;
                      StdCall; External 'PCI2322' Name ' ReadPortWord ';
```

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。
nPort　设备的 I/O 端口号。
返回值：返回由 nPort 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++ & C++ Builder:

```
ULONG ReadPortULong(HANDLE hDevice,
                    UINT nPort)
```

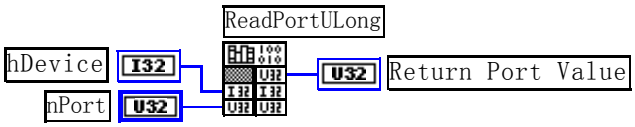
Visual Basic:

```
Declare Function ReadPortULong Lib "PCI2322" ( ByVal hDevice As Long, _
                                               ByVal nPort As Long ) As Long
```

Delphi:

```
Function ReadPortULong(hDevice : Integer;
                       nPort : LongWord) : LongWord;
                       StdCall; External 'PCI2322' Name ' ReadPortULong ';
```

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。

nPort　设备的 I/O 端口号。

返回值：返回由 nPort 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）

◆ **在 VB 环境中，创建子线程对象，以实现多线程操作**

函数原型：

Visual C++ & C++ Builder:

```
BOOL CreateVBThread(HANDLE *hThread,
                    LPTHREAD_START_ROUTINE StartThread)
```

Visual Basic：

```
Declare Function CreateVBThread Lib "PCI2322" ( ByRef hThread As Long, _
                                                ByVal StartThread As Long ) As Boolean
```

功能：该函数在 VB 环境中解决了不能实现或不能很好地实现多线程的问题。通过该函数用户可以很轻松地实现多线程操作。

参数：

hThread 若成功创建子线程，该参数将返回所创建的子线程的句柄，当用户操作这个子线程时将用到这个句柄，如启动线程、暂停线程以及删除线程等。

StartThread作为子线程运行的函数的地址，在实际使用时，请用AddressOf关键字取得该子线程函数的地址，再传递给CreateVBThread函数。

返回值：当成功创建子线程时，返回TRUE，且所创建的子线程为挂起状态，用户需要用Win32 API函数ResumeThread函数启动它。若失败，则返回FALSE，用户可用GetLastErrorEx捕获当前错误码。

相关函数： CreateVBThread　TerminateVBThread

注意: StartThread 指向的函数或过程必须放在 VB 的模块文件中，如 PCI2322.Bas 文件中。

Visual Basic 程序举例:

```
' 在模块文件中定义子线程函数(注意参考实例)
Function NewRoutine() As Long        ' 定义子线程函数
```

```
        :                    ' 线程运行代码
    NewRoutine = 1    ' 返回成功码
    End Function
    '-------------------------------------------------------------------
    ' 在窗体文件中创建子线程
        :
     Dim hNewThread As Long
    If Not CreateVBThread(hNewThread, AddressOf NewRoutine) Then ' 创建新子线程
        MsgBox "创建子线程失败"
        Exit Sub
    End If
    ResumeThread (hNewThread)   '启动新线程          :
        :
```

◆ **在 VB 中，删除子线程对象**

函数原型：

Visual C++ & C++ Builder:

BOOL TerminateVBThread(HANDLE hThreadHandle)

Visual Basic:

Declare Function TerminateVBThread Lib "PCI2322" (ByVal hThreadHandle As Long) As Boolean

功能：在VB中删除由CreateVBThread创建的子线程对象。

参数：hThreadHandle 指向需要删除的子线程对象的句柄，它应由CreateVBThread创建。

返回值：当成功删除子线程对象时，返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码。

相关函数： CreateVBThread　　TerminateVBThread

Visual Basic 程序举例:

```
        :
If Not TerminateVBThread (hNewThread)   ' 终止子线程
    MsgBox "创建子线程失败"
    Exit Sub
End If
        :
```

◆ **创建内核系统事件**

函数原型：

Visual C++ & C++ Builder:

HANDLE CreateSystemEvent(void)

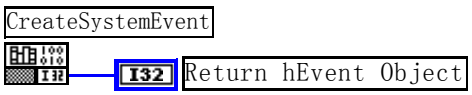
Visual Basic:

Declare Function CreateSystemEvent Lib " PCI2322 " () As Long

Delphi:

```
Function CreateSystemEvent() : Integer;
                        StdCall; External 'PCI2322' Name ' CreateSystemEvent ';
```

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++ & C++ Builder:

```
BOOL ReleaseSystemEvent(HANDLE hEvent)
```

Visual Basic:

```
Declare Function ReleaseSystemEvent Lib " PCI2322 " (ByVal hEvent As Long) As Boolean
```

Delphi:

```
Function ReleaseSystemEvent(hEvent : Integer) : Boolean;
                                StdCall; External 'PCI2322' Name ' ReleaseSystemEvent ';
```

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。

第五节、文件对象操作函数原型说明

♦ **创建文件对象**

函数原型：

Visual C++ & C++ Builder:

```
HANDLE CreateFileObject (HANDLE hDevice,
                         LPCTSTR NewFileName,
                         int Mode)
```

Visual Basic:

```
Declare Function CreateFileObject Lib "PCI2322" (ByVal hDevice As Long, _
                                                 ByVal NewFileName As String, _
                                                 ByVal Mode As Integer) As Long
```

Delphi:

```
Function CreateFileObject (hDevice : Integer;
                           NewFileName : string;
                           Mode : Integer) : Integer;
                           Stdcall; external 'PCI2322' Name ' CreateFileObject ';
```

LabVIEW:

请参见相关演示程序。

功能：初始化设备文件对象，以期待 WriteFile 请求准备文件对象进行文件操作。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
NewFileName 与新文件对象关联的磁盘文件名，可以包括盘符和路径等信息。在 C 语言中，其语法格式如：

“C:\\PCI2322\\Data.Dat”，在 Basic 中，其语法格式如：“C:\PCI2322\Data.Dat”。

Mode　文件操作方式，所用的文件操作方式控制字定义如下(可通过或指令实现多种方式并操作)：

常量名	常量值	功能定义
PCI2322_modeRead	0x0000	只读文件方式
PCI2322_modeWrite	0x0001	只写文件方式
PCI2322_modeReadWrite	0x0002	既读又写文件方式
PCI2322_modeCreate	0x1000	如果文件不存在可以创建该文件，如果存在，则重建此文件，且清 0
PCI2322_typeText	0x4000	以文本方式操作文件

返回值：若成功，则返回文件对象句柄。

相关函数： CreateDevice　CreateFileObject　WriteFile　ReadFile　ReleaseFile　ReleaseDevice

- **通过设备对象，往指定磁盘上写入用户空间的采样数据**

函数原型：

Visual C++ & C++ Builder:

```
BOOL WriteFile(HANDLE hFileObject,
               PVOID pDataBuffer,
               ULONG nWriteSizeBytes)
```

Visual Basic:

```
Declare Function WriteFile Lib "PCI2322" ( ByVal hFileObject As Long,_
                                           ByRef pDataBuffer As Byte,_
                                           ByVal nWriteSizeBytes As Long) As Boolean
```

Delphi:

```
Function WriteFile(hFileObject: Integer;
                   pDataBuffer : Pointer;
                   nWriteSizeBytes : LongWord) : Boolean;
                   Stdcall; external 'PCI2322' Name ' WriteFile ';
```

LabVIEW:

详见相关演示程序。

功能：通过向设备对象发送“写磁盘消息”，设备对象便会以最快的速度完成写操作。注意为了保证写入的数据是可用的，这个操作将与用户程序保持同步，但与设备对象中的环形内存池操作保持异步，以得到更高的数据吞吐量，其文件名及路径应由CreateFileObject函数中的strFileName指定。

参数：

hFileObject 设备对象句柄，它应由CreateFileObject创建。

pDataBuffer 用户数据空间地址，可以是用户分配的数组空间。

nWriteSizeBytes 告诉设备对象往磁盘上一次写入数据的长度(以字节为单位)。

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： CreateFileObject　WriteFile　ReadFile　ReleaseFile

- **通过设备对象,从指定磁盘文件中读采样数据**

函数原型：

Visual C++ & C++ Builder:

```
BOOL ReadFile( HANDLE hFileObject,
               PVOID pDataBuffer,
               ULONG OffsetBytes,
               ULONG nReadSizeBytes)
```

Visual Basic:

```
Declare Function ReadFile Lib "PCI2322" ( ByVal hFileObject As Long, _
                                          ByRef pDataBuffer As Integer, _
                                          ByVal OffsetBytes As Long, _
                                          ByVal nReadSizeBytes As Long) As Boolean
```

Delphi:

```
Function ReadFile(hFileObject : Integer;
                  pDataBuffer : Pointer;
                  OffsetBytes : LongWord;
                  nReadSizeBytes : LongWord) : Boolean;
                  Stdcall; external 'PCI2322' Name ' ReadFile ';
```

LabVIEW:

详见相关演示程序。

功能：将磁盘数据从指定文件中读入用户内存空间中，其访问方式可由用户在创建文件对象时指定。

参数：

hFileObject 设备对象句柄，它应由CreateFileObject创建。

pDataBuffer 用于接受文件数据的用户缓冲区指针，可以是用户分配的数组空间。

OffsetBytes 指定从文件开始端所偏移的读位置。

nReadSizeBytes 告诉设备对象从磁盘上一次读入数据的长度(以字为单位)。

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： CreateFileObject WriteFile ReadFile ReleaseFile

- **设置文件偏移位置**

函数原型：

Visual C++ & C++ Builder:

```
BOOL SetFileOffset (HANDLE hFileObject,
                    ULONG nOffsetBytes)
```

Visual Basic:

```
Declare Function SetFileOffset Lib "PCI2322" ( ByVal hFileObject As Long,
                                               ByVal nOffsetBytes As Long) As Boolean
```

Delphi:

```
Function SetFileOffset ( hFileObject : Integer;
                         nOffsetBytes : LongWord) : Boolean;
                         Stdcall; external 'PCI2322' Name ' SetFileOffset ';
```

LabVIEW:

详见相关演示程序。

功能：设置文件偏移位置，用它可以定位读写起点。

参数：hFileObject 文件对象句柄，它应由CreateFileObject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： CreateFileObject WriteFile ReadFile ReleaseFile

- **取得文件长度（字节）**

函数原型：

Visual C++ & C++ Builder:

```
ULONG GetFileLength (HANDLE hFileObject);
```

Visual Basic:

```
Declare Function GetFileLength Lib "PCI2322" (ByVal hFileObject As Long) As Long
```

Delphi:

```
Function GetFileLength (hFileObject : Integer) : LongWord;
                        Stdcall; external 'PCI2322' Name ' GetFileLength ';
```

LabVIEW:

详见相关演示程序。

功能：取得文件长度。

参数：hFileObject 设备对象句柄，它应由CreateFileObject创建。

返回值：若成功，则返回>1，否则返回 0，用户可以用GetLastErrorEx捕获错误码。

相关函数： CreateFileObject WriteFile ReadFile ReleaseFile

- **释放设备文件对象**

函数原型：

Visual C++ & C++ Builder:

```
BOOL ReleaseFile(HANDLE hFileObject)
```

Visual Basic:

```
Declare Function ReleaseFile Lib "PCI2322" (ByVal hFileObject As Long) As Boolean
```

Delphi:

```
Function ReleaseFile(hFileObject : Integer) : Boolean;
                     Stdcall; external 'PCI2322' Name ' ReleaseFile ';
```

LabVIEW:

详见相关演示程序。

功能：释放设备文件对象。

参数：hFileObject 设备对象句柄，它应由CreateFileObject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： CreateFileObject WriteFile ReadFile ReleaseFile

- **取得指定磁盘的可用空间**

Visual C++ & C++ Builder:

```
ULONGLONG GetDiskFreeBytes(LPCTSTR DiskName)
```

Visual Basic:

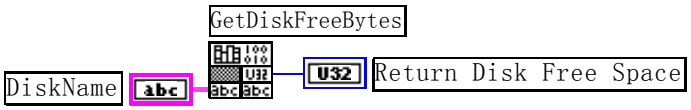
```
Declare Function GetDiskFreeBytes Lib "PCI2322" (ByVal DiskName As String ) As Currency
```

Delphi:

```
Function GetDiskFreeBytes (DiskName : String) : Currency;
```

Stdcall; external 'PCI2322' Name ' GetDiskFreeBytes ';

LabVIEW:



功能：取得指定磁盘的可用剩余空间(以字为单位)。

参数：DiskName 需要访问的盘符，若为 C 盘为"C:\\"，D 盘为"D:\\"，以此类推。

返回值：若成功，返回大于或等于 0 的长整型值，否则返回零值，用户可用GetLastErrorEx捕获错误码。注意使用 64 位整型变量。

第六节、各种参数保存和读取函数原型说明

♦ **将整型变量的参数值保存在系统注册表中**

函数原型：

Visual C++ & C++ Builder:

```
BOOL SaveParaInt(HANDLE hDevice,
                 LPCTSTR strParaName,
                 int nValue)
```

Visual Basic:

```
Declare Function SaveParaInt Lib "PCI2322" ( ByVal hDevice As Long,_
                                             ByVal strParaName As String,_
                                             ByVal nValue As Integer) As Boolean
```

Delphi:

```
Function SaveParaInt( hDevice : Integer;
                      strParaName : String;
                      nValue : Integer) : Boolean;
                     Stdcall; external 'PCI2322' Name ' SaveParaInt ';
```

LabVIEW:

详见相关演示程序。

功能：将整型变量的参数值保存在系统注册表中。具体保存位置视设备逻辑号而定。如逻辑号为“0”的其他参数保存位置为：HKEY_CURRENT_USER\Software\Art\PCI2322\Device-0\Others。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

strParaName 整型参数字符名。它指名该参数在注册表中的字符键项。

nValue 整型参数值。它保存在由 strParaName 命名的键项里。

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： SaveParaInt LoadParaInt SaveParaString LoadParaString

♦ **将整型变量的参数值从系统注册表中读出**

函数原型：

Visual C++ & C++ Builder:

```
UINT LoadParaInt(HANDLE hDevice,
                 LPCTSTR strParaName,
                 int nDefaultVal)
```

Visual Basic:

```
Declare Function LoadParaInt Lib "PCI2322" (ByVal hDevice As Long,_
                                            ByVal strParaName As String,_
                                            ByVal nDefaultVal As Integer) As Long
```

Delphi:

```
Function LoadParaInt ( hDevice : Integer;
                       strParaName : String;
                       nDefaultVal: Integer) : LongWord
                      Stdcall; external 'PCI2322' Name ' LoadParaInt ';
```

LabVIEW:

详见相关演示程序。

功能：将整型变量的参数值从系统注册表中读出。读出参数值的具体位置视设备逻辑号而定。如逻辑号为“0”的其他参数保存位置为：HKEY_CURRENT_USER\Software\Art\PCI2322\Device-0\Others。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

strParaName 整型参数字符名。它指名该参数在注册表中的字符键项。

nDefaultVal 若 strParaName 指定的键项不存在，则由该参数指定的默认值返回。

返回值：若指定的整型参数项存在，则返回其整型值。否则返回由 nDefaultVal 指定的默认值。

相关函数： SaveParaInt LoadParaInt SaveParaString LoadParaString

- **将字符变量的参数值保存在系统注册表中**

函数原型：

Visual C++ & C++ Builder:

```
BOOL SaveParaString ( HANDLE hDevice,
                      LPCTSTR strParaName,
                      LPCTSTR strParaVal)
```

Visual Basic:

```
Declare Function SaveParaString Lib "PCI2322" (ByVal hDevice As Long,_
                                               ByVal strParaName As String,_
                                               ByVal strParaVal As String) As Boolean
```

Delphi:

```
Function SaveParaString (hDevice : Integer;
                         strParaName : String;
                         strParaVal: String) : Boolean;
                        Stdcall; external 'PCI2322' Name ' SaveParaString';
```

LabVIEW:

详见相关演示程序。

功能：将整型变量的参数值保存在系统注册表中。具体保存位置视设备逻辑号而定。如逻辑号为“0”的其他参数保存位置为：HKEY_CURRENT_USER\Software\Art\PCI2322\Device-0\Others。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

strParaName 整型参数字符名。它指名该参数在注册表中的字符键项。

strParaVal 字符参数值。它保存在由 strParaName 命名的键项里。

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： SaveParaInt LoadParaInt SaveParaString LoadParaString

♦ **将字符变量的参数值从系统注册表中读出**

函数原型：

Visual C++ & C++ Builder:

```
BOOL LoadParaString ( HANDLE hDevice,
                      LPCTSTR strParaName,
                      LPCTSTR strParaVal,
                      LPCTSTR strDefaultVal)
```

Visual Basic:

```
Declare Function LoadParaString Lib "PCI2322" (ByVal hDevice As Long,_
                                               ByVal strParaName As String,_
                                               ByVal strParaVal As String,_
                                               ByVal strDefaultVal As String) As Boolean
```

Delphi:

```
Function LoadParaString (hDevice : Integer;
                         strParaName : String;
                         strParaVal : String;
                         strDefaultVal : String) : Boolean;
                        Stdcall; external 'PCI2322' Name ' LoadParaString ';
```

LabVIEW:

详见相关演示程序。

功能：将字符变量的参数值从系统注册表中读出。读出参数值的具体位置视设备逻辑号而定。如逻辑号为“0”的其他参数保存位置为：HKEY_CURRENT_USER\Software\Art\PCI2322\Device-0\Others。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

strParaName 字符参数字符名。它指名该参数在注册表中的字符键项。

strParaVal 取得 strParaName 指定的键项的字符值。

strDefaultVal 若 strParaName 指定的键项不存在，则由该参数指定的默认值返回。

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： SaveParaInt LoadParaInt SaveParaString LoadParaString

第七节、其他函数原型说明

♦ **怎样获取驱动函数错误信息**

函数原型：

Visual C++ & C++ Builder:

```
DWORD GetLastErrorEx (LPCTSTR strFuncName,
                      LPTSTR strErrorMsg)
```

Visual Basic:

```
Declare Function GetLastErrorEx Lib "PCI2322" (ByVal strFuncName As String,_
                                               ByVal strErrorMsg As String) As Long
```

Delphi:

```
Function GetLastErrorEx (strFuncName: String;
                         strErrorMsg: String) : LongWord;
                         Stdcall; external 'PCI2322' Name ' GetLastErrorEx ';
```

LabVIEW:

请参见相关演示程序。

功能：将当某个驱动函数出错时，可以调用此函数获得具体的错误和错误信息字串。

参数：

strFuncName 出错函数的名称。注意此函数必须是完整名称，如 AD 初始化函数 PCI2322_ InitDeviceAD 出现错误，此时调用该函数时，此参数必须为“PCI2322_ InitDeviceAD”，否则得不到相应信息。

strErrorMsg 取得指定函数的错误信息串。该串为字符数组，其分配空间最好不要小于 256 字节。

返回值：返回错误码。

相关函数：无。

- **移除驱动函数错误信息**

函数原型：

Visual C++ & C++ Builder:

```
BOOL RemoveLastErrorEx (LPCTSTR strFuncName)
```

Visual Basic:

```
Declare Function RemoveLastErrorEx Lib "PCI2322" (ByVal strFuncName As String) As Boolean
```

Delphi:

```
Function RemoveLastErrorEx (strFuncName: String) : Boolean;
                            Stdcall; external 'PCI2322' Name ' RemoveLastErrorEx';
```

LabVIEW:

详见相关演示程序。

功能：从错误信息库中移除指定函数的最后一次错误信息。

参数：

strFuncName 出错函数的名称。注意此函数必须是完整名称，如 AD 初始化函数 PCI2322_ InitDeviceAD 出现错误，此时调用该函数时，此参数必须为“PCI2322_ InitDeviceAD”，否则得不到相应信息。

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：无。

- **高效高精度延时**

函数原型：

Visual C++ & C++ Builder:

```
BOOL DelayTimeUs (HANDLE hDevice,
                  LONG nTimeUs)
```

Visual Basic:

```
Declare Function DelayTimeUs Lib "PCI2322" (ByVal hDevice As Long, _
                                            ByVal nTimeUs As Long) As Boolean
```

Delphi:

```
Function DelayTimeUs (hDevice: Integer;
                      nTimeUs : LongInt) : Boolean;
                      StdCall; External 'PCI2322' Name ' DelayTimeUs ';
```

LabVIEW:

请参考相关演示程序。

功能：微秒级延时函数。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
nTimeUs 时间常数。单位 1 微秒。
返回值：若成功，返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获错误码。