PCI2323 数据采集卡

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

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 PCI2323_CreateDevice 则写为 CreateDevice。

二、函数名及参数中各种关键字缩写

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			

第二章 使用纲要

第一节、使用上层用户函数，高效、简单

如果您只关心通道及频率等基本参数，而不必了解复杂的硬件知识和控制细节，那么我们强烈建议您使用上层用户函数，它们就是几个简单的形如Win32 API的函数，具有相当的灵活性、可靠性和高效性。而底层用户函数如WriteRegisterULong、ReadRegisterULong、WritePortByte、ReadPortByte……则是满足了解硬件知识和控制细节、且又需要特殊复杂控制的用户。但不管怎样，我们强烈建议您使用上层函数（在这些函数中，您见不到任何设备地址、寄存器端口、中断号等物理信息，其复杂的控制细节完全封装在上层用户函数中。）对于上层用户函数的使用，您基本上不必参考硬件说明书，除非您需要知道板上D型插座等管脚分配情况。

第二节、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。

第三节、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[16]中的相应元素决定。

第四节、哪些函数对您不是必须的

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。如果您使用上层用户函数访问设备，那么GetDeviceAddr， WriteRegisterByte， WriteRegisterWord，

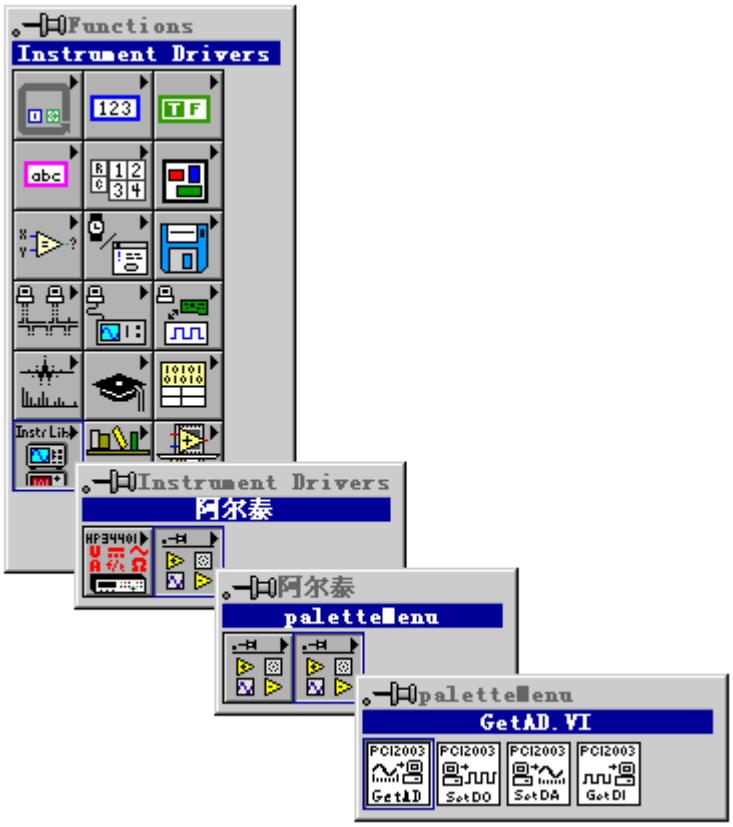
WriteRegisterULong， ReadRegisterByte， ReadRegisterWord， ReadRegisterULong等函数您可完全不必理会，除非您是作为底层用户管理设备。而WritePortByte， WritePortWord， WritePortULong， ReadPortByte，ReadPortWord， ReadPortULong则对PCI用户来讲，可以说完全是辅助性，它们只是对我公司驱动程序的一种功能补充，对用户额外提供的，它们可以帮助您在NT、Win2000 等操作系统中实现对您原有传统设备如ISA卡、串口卡、并口卡的访问，而没有这些函数，您可能在基于Windows NT架构的操作系统中无法继续使用您原有的老设备。

第三章 PCI 即插即用设备操作函数接口介绍

由于我公司的设备应用于各种不同的领域，有些用户可能根本不关心硬件设备的控制细节，只关心首末通道、采样频率等，然后就能通过一两个简易的采集函数便能轻松得到所需要的数据。这方面的用户我们称之为上层用户。那么还有一部分用户不仅对硬件控制熟悉，而且由于应用对象的特殊要求，则要直接控制设备的每一个端口，这是一种复杂的工作，但又是必须的工作，我们则把这一群用户称之为底层用户。因此总的看来，上层用户要求简单、快捷，他们最希望在软件操作上所要面对的全是他们最关心的问题，而关于设备的物理地址、端口分配及功能定义等复杂的硬件信息则与上层用户无任何关系。那么对于底层用户则不然。他们不仅要关心设备的物理地址，还要关心虚拟地址、端口寄存器的功能分配，甚至每个端口的Bit位都要了如指掌，看起来这是一项相当复杂、繁琐的工作。但是这些底层用户一旦使用我们提供的技术支持，则不仅可以让您不必熟悉PCI总线复杂的控制协议，同是还可以省掉您许多繁琐的工作，比如您不用去了解PCI的资源配置空间、PNP即插即用管理，而只须用GetDeviceAddr函数便可以同时取得指定设备的物理基地址和虚拟线性基地址。这个时候您便可以用这个虚拟线性基地址，再根据硬件使用说明书中的各端口寄存器的功能说明，然后使用ReadRegisterULong和WriteRegisterULong对这些端口寄存器进行 32 位模式的读写操作，即可实现设备的所有控制。

综上所述，用户使用我公司提供的驱动程序软件包将极大的方便和满足您的各种需求。但为了您更省心，别忘了在您正式阅读下面的函数说明时，先明白自己是上层用户还是底层用户，因为在《设备驱动接口函数总列表》中的备注栏里明确注明了适用对象。

另外需要申明的是，在本章和下一章中列明的关于 LabView 的接口，均属于外挂式驱动接口，他是通过 LabView 的 Call Labrary Function 功能模板实现的。它的特点是除了自身的语法略有不同以外，每一个基于 LabView 的驱动图标与 Visual C++、Visual Basic、Delphi 等语言中每个驱动函数是一一对应的，其调用流程和功能是完全相同的。那么相对于外挂式驱动接口的另一种方式是内嵌式驱动。这种驱动是完全作为 LabView 编程环境中的紧密耦合的一部分，它可以直接从 LabView 的 Functions 模板中取得，如下图所示。此种方式更适合上层用户的需要，它的最大特点是方便、快捷、简单，而且可以取得它的在线帮助。关于 LabView 的外挂式驱动和内嵌式驱动更详细的叙述，请参考 LabView 的相关演示。



LabView 内嵌式驱动接口的获取方法

第一节、设备驱动接口函数总列表（每个函数省略了前缀“PCI2323_”）

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 开关量函数		
GetDeviceDI	取得数字量输入状态	上层用户
SetDeviceDO	设置数字量输出状态	上层用户
③ 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
GetDeviceIntSrc	在中断初始化后，用它取得中断源	上层用户
ReleaseDeviceInt	释放中断资源	上层用户

使用需知：

Visual C++ & C++Builder：

要使用如下函数关键的问题是：

首先，必须在您的源程序中包含如下语句：

#include “C:\Art\PCI2323\INCLUDE\PCI2323.H”

注：以上语句采用默认路径和默认板号，应根据您的板号和安装情况确定 PCI2323.H 文件的正确路径，当然也可以把此文件拷到您的源程序目录中。然后加入如下语句：

#include “PCI2323.H”

另外，要在 VB 环境中用子线程以实现高速、连续数据采集与存盘，请务必使用 VB5.0 版本。当然如果您有 VB6.0 的最新版，也可以实现子线程操作。

C++ Builder:

要使用如下函数一个关键的问题是首先必须将我们提供的头文件(PCI2323.H)写进您的源程序头部。如：#include "\Art\PCI2323\Include\PCI2323.h"，然后再将 PCI2323.Lib 库文件分别加入到您的 C++ Builder 工程中。其具体办法是选择 C++ Builder 集成开发环境中的工程(Project)菜单中的“添加”（Add to Project）命令，在弹出的对话框中分别选择文件类型：Library file (*.lib) ， 即可选择 PCI2323.Lib 文件。该文件的路径为用户安装驱动程序后其子目录 Samples\C_Builder 下。

Visual Basic:

要使用如下函数一个关键的问题是首先必须将我们提供的模块文件(*.Bas)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 PCI2323.Bas 模块文件，该文件的路径为用户安装驱动程序后其子目录 Samples\VB 下面。

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，我们不能保证完全顺利运行。

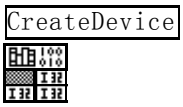
Delphi:

要使用如下函数一个关键的问题是首先必须将我们提供的单元模块文件（*.Pas）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 PCI2323.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。该文件的路径为用户安装驱动程序后其子目录 Samples\Delphi 下面。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：“PCI2323”。如：

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  PCI2323;  // 注意： 在此加入驱动程序接口单元 PCI2323
```

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 的进一步介绍请见本文最后一部分关于 LabView 的专述。其驱动程序接口单元模块的使用方法如下：



一、在 LabView 中打开 PCI2323.VI 文件，用鼠标单击接口单元图标，比如 CreateDevice 图标 然后按 Ctrl+C 或选择 LabView 菜单 Edit 中的 Copy 命令，接着进入用户的应用程序 LabView 中，按 Ctrl+V 或选择 LabView 菜单 Edit 中的 Paste 命令，即可将接口单元加入到用户工程中，然后按以下函数原型说明或演示程序的说明连接该接口模块即可顺利使用。

二、根据 LabView 语言本身的规定，接口单元图标以黑色的较粗的中间线为中心，以左边的方格为数据输入端，右边的方格为数据的输出端，设备对象句柄、用户分配的数据缓冲区、要求采集的数据长度等信息从接口单元左边输入端进入单元，待单元接口被执行后，需要返回给用户的数据从接口单元右边的输出端输出，其他接口完全同理。

三、在单元接口图标中，凡标有“I32”为有符号长整型 32 位数据类型，“U16”为无符号短整型 16 位数据类型，“［U16］”为无符号 16 位短整型数组或缓冲区或指针，“［U32］”与“[U16]”同理，只是位数不一样。

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)

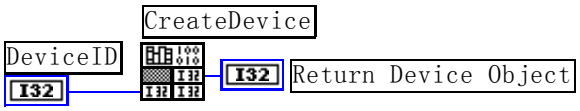
Visual Basic：

Declare Function CreateDevice Lib "PCI2323" (Optional ByVal DeviceID As Integer = 0) As Long

Delphi：

Function CreateDevice(DeviceID : Integer = 0) : Integer;
StdCall; External 'PCI2323' 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 PCI2323 设备的总数量**

函数原型：

Visual C++ & C++Builder:

int GetDeviceCount (HANDLE hDevice)

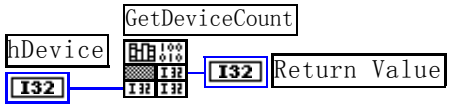
Visual Basic:

Declare Function GetDeviceCount Lib "PCI2323" (ByVal hDevice As Long) As Integer

Delphi：

Function GetDeviceCount (hDevice : Integer) : Integer;
StdCall; External 'PCI2323' Name ' GetDeviceCount ';

LabVIEW：



功能：取得 PCI2323 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 PCI2323 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 "PCI2323" (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 'PCI2323' 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

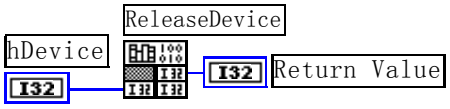
- **用对话框控件列表计算机系统中所有 PCI2323 设备各种配置信息**

函数原型：
Visual C++ & C++Builder:
```
BOOL ListDeviceDlg (HANDLE hDevice)
```
Visual Basic:
```
Declare Function ListDeviceDlg Lib "PCI2323" (ByVal hDevice As Long ) As Boolean
```
Delphi：
```
Function ListDeviceDlg (hDevice : Integer) : Boolean;
                    StdCall; External 'PCI2323' Name ' ListDeviceDlg ';
```
LabVIEW：
请参考相关演示程序。

功能：列表系统中 PCI2323 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 PCI2323 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 "PCI2323" (ByVal hDevice As Long ) As Boolean
```
Delphi：
```
Function ReleaseDevice(hDevice : Integer) : Boolean;
                    StdCall; External 'PCI2323' 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、DIO 数字开关量输入输出简易操作函数原型说明

- **十六路开关量输入**

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDeviceDI(HANDLE hDevice,
                 BYTE bDOISts [16])
```

Visual Basic:

```
Declare Function GetDeviceDI Lib "PCI2323" (ByVal hDevice As Long, _
                                            ByVal bDOISts (0 to 15 ) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI(hDevice : Integer;
                     bDOISts : Pointer):Boolean;
                     StdCall; External 'PCI2323' Name ' GetDeviceDI ';
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO0～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 十六路开关量输入状态的参数结构，共有 16 个成员变量，分别对应于 DIO0～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。

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 SetDeviceDO ReleaseDevice

♦ **十六路开关量输出**

函数原型：

Visual C++ & C++Builder:

```
BOOL SetDeviceDO(HANDLE hDevice,
                 BYTE bDOISts[16])
```

Visual Basic:

```
Declare Function SetDeviceDO Lib "PCI2323" (ByVal hDevice As Long, _
                                            ByVal bDOISts(0 to 15 )As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO(hDevice : Integer;
                     bDOISts : Pointer):Boolean;
                     StdCall; External 'PCI2323' Name ' SetDeviceDO ';
```

LabView：

请参考相关演示程序。

功能：负责将 PCI 设备上的 DIO0～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 十六路开关量输出状态的参数结构，共有 16 个成员变量，分别对应于 DIO0～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 DIO0 至 DIO15 共 16 个成员变量赋初值，其值必须为“1”或“0”。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice GetDeviceDI ReleaseDevice

❖ **以上函数调用一般顺序**

① CreateDevice

② SetDeviceDO（或GetDeviceDI，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。

第四节、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 INTSrc)
```

Visual Basic:

```
Declare Function InitDeviceInt Lib "PCI2323" ( ByVal hDevice As Long, _
                                               ByVal hEventInt As Long, _
                                               ByVal INTSrc As Long) As Boolean
```

Delphi:

```
Function InitDeviceInt (hDevice : Integer;
                        hEventInt : Integer;
                        INTSrc : LongInt) : Boolean;
                        StdCall; External 'PCI2323' 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。

INTSrc 中断源选择。

常量名	常量值	功能定义
PCI2323_INTSRC_COSINT	0x0000	使能 COS 中断(使能该中断后，还需使能数字量输入，使能任意数字量输入通道，当所使能的数字量输入通道状态改变时产生该中断)
PCI2323_INTSRC_DI0INT	0x0001	使能 DI0 中断(数字量输入通道 0 上升沿产生中断)，=TRUE 使能，=FALSE 禁止
PCI2323_INTSRC_DI1INT	0x0002	使能 DI1 中断(数字量输入通道 1 上升沿产生中断)，=TRUE 使能，=FALSE 禁止
PCI2323_INTSRC_DI01INT	0x0003	使能 DI0 与 DI1 中断(数字量输入通道 0 或 1 上升沿产生中断)，=TRUE 使能，=FALSE 禁止

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　GetDeviceIntSrc　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 bDIInt[16])
```

Visual Basic:

```
Declare Function SetCOSINTCH Lib "PCI2323" (ByVal hDevice As Boolean, _
                                            ByVal bDIInt(0 to 15) As Boolean) As Boolean
```

Delphi:

```
Function SetCOSINTCH (hDevice : Integer;
                      bDIInt: Boolean) : Boolean;
```

StdCall; External 'PCI2323' 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。

bDIInt[16] 16 路 DI 通道中断使能。

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 GetDeviceIntSrc 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 "PCI2323" (ByVal hDevice As Long) As Long
```

Delphi:

```
Function GetDeviceIntCount (hDevice : Integer) : LongInt;
                    StdCall; External 'PCI2323' 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 GetDeviceIntSrc ReleaseDeviceInt ReleaseDevice

◆ **取得中断源**

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDeviceIntSrc(HANDLE hDevice,
                     PPCI2323_STATUS_INT pINTSts)
```

Visual Basic:

```
Declare Function GetDeviceIntSrc Lib "PCI2323" (ByVal hDevice As Boolean, _
                                                ByVal pINTSts As PPCI2323_STATUS_INT) As Boolean
```

Delphi:

```
Function GetDeviceIntSrc(hDevice : Integer;
                         pINTSts : PCI2323_STATUS_INT) : Boolean;
                         StdCall; External 'PCI2323' Name ' GetDeviceIntSrc ';
```

LabVIEW:

请参考相关演示程序。

功能：在中断初始化后，用它取得中断源。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pINTSts 中断信息控制参数。

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 GetDeviceIntSrc 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 "PCI2323" (ByVal hDevice As Long) As Boolean

Delphi:

Function ReleaseDeviceInt (hDevice : Integer) : Boolean;
StdCall; External 'PCI2323' 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　GetDeviceIntSrc　ReleaseDeviceInt
ReleaseDevice

第四章 硬件参数结构

中断状态参数结构（PCI2323_STATUS_INT）

Visual C++ & C++Builder：

```
typedef struct _PCI2323_STATUS_INT
{
    LONG bCOSINT;                    // COS 触发中断(16 路数字量输入状态改变产生中断)
    LONG bINT0;                      // INT0 触发中断
    LONG bINT1;                      // INT1 触发中断
} PCI2323_STATUS_INT, *PPCI2323_STATUS_INT;
```

Visual Basic：

```
Private Type PCI2323_STATUS_INT
    bCOSINT As Long
    bINT0 As Long
    bINT1 As Long
End Type
```

Delphi:

```
type   // 定义结构体数据类型
    PPCI2323_STATUS_INT = ^ PCI2323_STATUS_INT;   // 指针类型结构
    PCI2323_STATUS_INT = record          // 标记为记录型
        bCOSINT : LongInt;
        bINT0 : LongInt;
        bINT1 : LongInt;
End;
```

LabVIEW:

请参考相关演示程序。

硬件参数说明:

bCOSINT COS 触发中断(16 路数字量输入状态改变产生中断)。

bINT0 INT0 触发中断。

bINT1 INT1 触发中断。

第五章 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 PCI2323.h 和 Sys.cpp)。

[程序]丨**[阿尔泰测控演示系统]**丨**[PCI2323 16 路开关量输入输出卡]**丨**[Microsoft Visual C++]**丨[简易代码演示]丨[DI 开关量演示源程序]

其默认存放路径为：系统盘\ART\PCI2323\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 PCI2323.h 和 Sys.cpp)。

[程序]丨**[阿尔泰测控演示系统]**丨**[PCI2323 16 路开关量输入输出卡]**丨**[Microsoft Visual C++]**丨[简易代码演示]丨[DO 开关量演示源程序]

其默认存放路径为：系统盘\ART\PCI2323\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 PCI2323.h 和 Sys.cpp)。

[程序]丨**[阿尔泰测控演示系统]**丨**[PCI2323 16 路开关量输入输出卡]**丨**[Microsoft Visual C++]**丨[高级代码演示]

其默认存放路径为：系统盘\ART\PCI2323\SAMPLES\VC\ADVANCED

其他语言的演示可以用上面类似的方法找到。

第六章 共用函数介绍

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“PCI2323_”）

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)
```

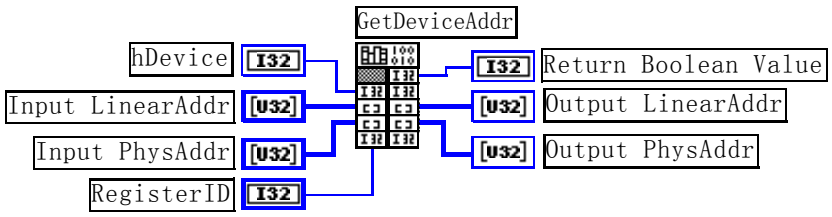
Visual Basic:

```
Declare Function GetDeviceAddr Lib "PCI2323" (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 'PCI2323' 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　GetDeviceAddr　WriteRegisterByte　WriteRegisterWord　WriteRegisterULong　ReadRegisterByte　ReadRegisterWord　ReadRegisterULong　ReleaseDevice

Visual C++ & C++ Builder 程序举例:

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

Visual Basic 程序举例:

```
    :
Dim hDevice As Long
Dim LinearAddr, PhysAddr    As Long
hDevice = CreateDevice(0)
if Not GetDeviceAddr(hDevice, LinearAddr,    PhysAddr, 0) then
    MsgBox "取得设备地址失败…"
End If
    :
```

◆ **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 "PCI2323" (ByVal hDevice As Long, _
                                             ByVal pulPCIBar (0 to 5) As Long) As Boolean
```

Delphi:

```
Function GetDeviceBar (hDevice : Integer;
                       pulPCIBar : Pointer) : Boolean;
                       StdCall; External 'PCI2323' 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 "PCI2323" (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 'PCI2323' Name ' GetDevVersion ';
```

LabVIEW:

请参考相关演示程序。

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 固件版本。
pulDriverVersion 驱动版本。
返回值：若成功，返回 TRUE，否则返回 FALSE。
相关函数: CreateDevice GetDeviceAddr WriteRegisterByte
WriteRegisterWord WriteRegisterULong ReadRegisterByte
ReadRegisterWord ReadRegisterULong 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)
```

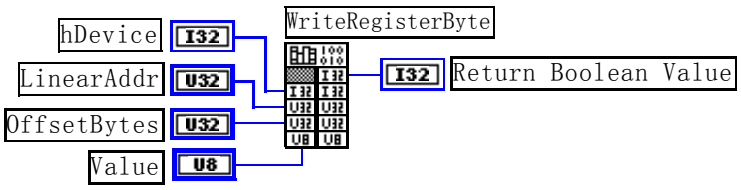
Visual Basic:

```
Declare Function WriteRegisterByte Lib "PCI2323" (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 'PCI2323' 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　GetDeviceAddr　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)
```

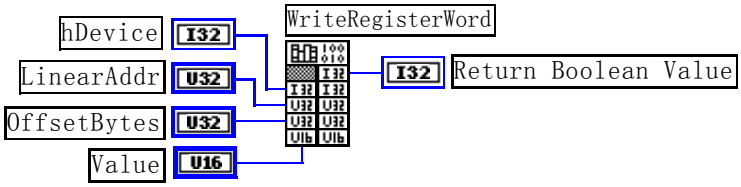
Visual Basic:

```
Declare Function WriteRegisterWord Lib "PCI2323" (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 'PCI2323' 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　GetDeviceAddr　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:

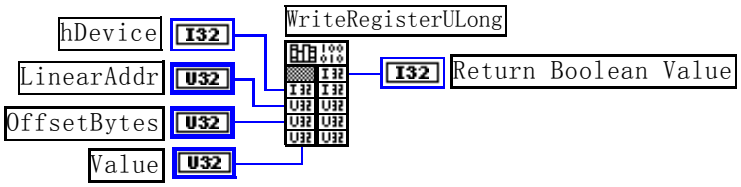
```
Declare Function WriteRegisterULong Lib "PCI2323" (ByVal hDevice As Long, _
                                                   ByVal LinearAddr As Long, _
                                                   ByVal OffsetBytes As Long, _
                                                   ByVal Value As Long) As Boolean
```

Delphi:

```
Function WriteRegisterULong(hDevice : Integer;
                            LinearAddr : LongWord;
                            OffsetBytes : LongWord;
                            Value : LongWord) : Boolean;
```

StdCall; External 'PCI2323' 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 GetDeviceAddr 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)
```

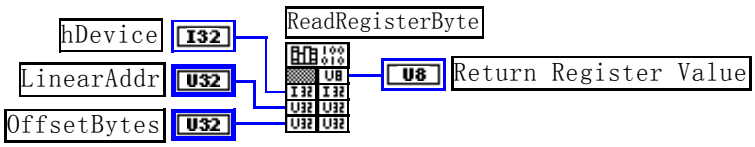
Visual Basic:

```
Declare Function ReadRegisterByte Lib "PCI2323" (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 'PCI2323' 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　GetDeviceAddr　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)
```

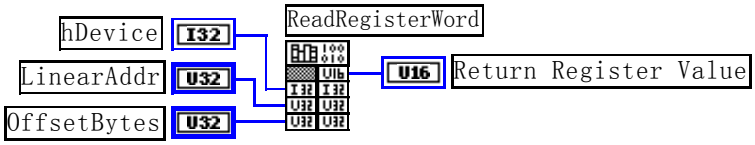
Visual Basic:

```
Declare Function ReadRegisterWord Lib "PCI2323" (  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 'PCI2323' 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 GetDeviceAddr 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)
```

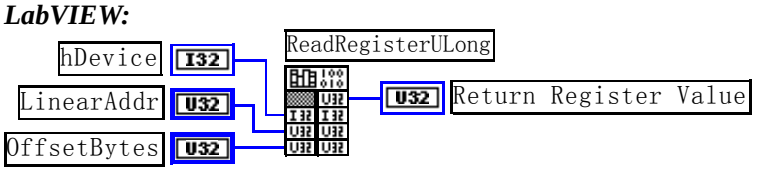
Visual Basic:

```
Declare Function ReadRegisterULong Lib "PCI2323" (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 'PCI2323' 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　GetDeviceAddr　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、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**

函数原型：

Visual C++ & C++ Builder:

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

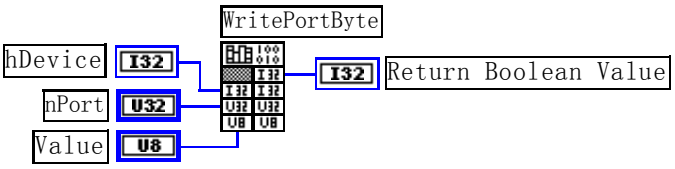
Visual Basic:

```
Declare Function WritePortByte Lib "PCI2323" ( 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 'PCI2323' 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**

函数原型：
Visual C++ & C++ Builder:

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

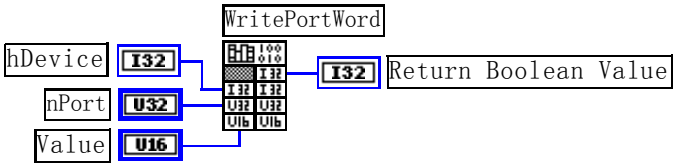
Visual Basic:

```
Declare Function WritePortWord Lib "PCI2323" (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 'PCI2323' 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**

函数原型：
Visual C++ & C++ Builder:

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

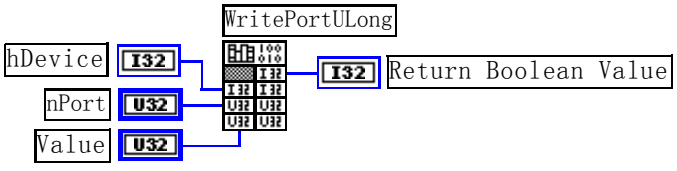
Visual Basic:

```
Declare Function WritePortULong Lib "PCI2323" (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 'PCI2323' 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**

函数原型：
Visual C++ & C++ Builder:

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

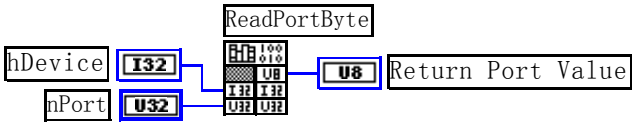
Visual Basic:

```
Declare Function ReadPortByte Lib "PCI2323" (ByVal hDevice As Long, _
                                             ByVal nPort As Long ) As Byte
```

Delphi:

```
Function ReadPortByte(hDevice : Integer;
                      nPort : LongWord) : Byte;
                      StdCall; External 'PCI2323' 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**

函数原型：
Visual C++ & C++ Builder:

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

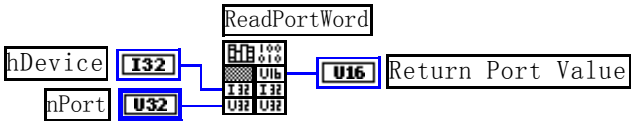
Visual Basic:

```
Declare Function ReadPortWord Lib "PCI2323" ( ByVal hDevice As Long, _
                                              ByVal nPort As Long ) As Integer
```

Delphi:

```
Function ReadPortWord(hDevice : Integer;
                      nPort : LongWord) : Word;
                      StdCall; External 'PCI2323' 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**

函数原型：

Visual C++ & C++ Builder:

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

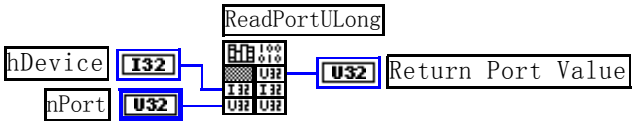
Visual Basic:

```
Declare Function ReadPortULong Lib "PCI2323" ( ByVal hDevice As Long, _
                                               ByVal nPort As Long ) As Long
```

Delphi:

```
Function ReadPortULong(hDevice : Integer;
                       nPort : LongWord) : LongWord;
                       StdCall; External 'PCI2323' 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 "PCI2323" ( 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 PCI2323.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 "PCI2323" (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)

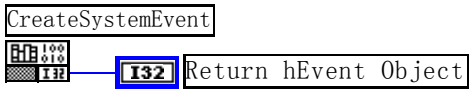
Visual Basic:

Declare Function CreateSystemEvent Lib " PCI2323 " () As Long

Delphi:

Function CreateSystemEvent() : Integer;
StdCall; External 'PCI2323' 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 " PCI2323 " (ByVal hEvent As Long) As Boolean
```

Delphi:

```
Function ReleaseSystemEvent(hEvent : Integer) : Boolean;
                                    StdCall; External 'PCI2323' 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 "PCI2323" (ByVal hDevice As Long, _
                                                 ByVal NewFileNameAs String, _
                                                 ByVal Mode As Integer) As Long
```

Delphi:

```
Function CreateFileObject (hDevice : Integer;
                           NewFileName: String;
                           Mode : Integer) : Integer;
                           Stdcall; external 'PCI2323' 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 新文件名。

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
PCI2323_modeRead	0x0000	只读文件方式
PCI2323_modeWrite	0x0001	只写文件方式
PCI2323_modeReadWrite	0x0002	既读又写文件方式
PCI2323_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
PCI2323_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,
               LONG nWriteSizeBytes)
```

Visual Basic:

```
Declare Function WriteFile Lib "PCI2323" ( ByVal hFileObject As Long,_
                                           ByRef pDataBuffer As Integer,_
```

```
                                        ByVal nWriteSizeBytes As Long) As Boolean
```

Delphi:

```
Function WriteFile(hFileObject: Integer;
                   pDataBuffer : Pointer;
                   nWriteSizeBytes : LongInt) : Boolean;
                   Stdcall; external 'PCI2323' 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,
               LONG nOffsetBytes,
               LONG nReadSizeBytes)
```

Visual Basic:

```
Declare Function ReadFile Lib "PCI2323" (  ByVal hFileObject As Long, _
                                           ByRef pDataBuffer As Integer, _
                                           ByVal nOffsetBytes As Long, _
                                           ByVal nReadSizeBytes As Long) As Boolean
```

Delphi:

```
Function ReadFile(hFileObject : Integer;
                  pDataBuffer : Pointer;
                  nOffsetBytes : LongInt;
                  nReadSizeBytes : LongInt) : Boolean;
                  Stdcall; external 'PCI2323' 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。

nOffsetBytes 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,
                    LONG nOffsetBytes)
```

Visual Basic:

```
Declare Function SetFileOffset Lib "PCI2323" ( ByVal hFileObject As Long,_
                                               ByVal nOffsetBytes As Long) As Boolean
```

Delphi:

```
Function SetFileOffset ( hFileObject : Integer;
                         nOffsetBytes : LongInt) : Boolean;
                         Stdcall; external 'PCI2323' 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 "PCI2323" (ByVal hFileObject As Long) As Long
```

Delphi:

```
Function GetFileLength (hFileObject : Integer) : LongWord;
                        Stdcall; external 'PCI2323' 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 "PCI2323" (ByVal hFileObject As Long) As Boolean
```

Delphi:

```
Function ReleaseFile(hFileObject : Integer) : Boolean;
                     Stdcall; external 'PCI2323' 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**

函数原型：

Visual C++ & C++ Builder:

```
ULONGLONG GetDiskFreeBytes(LPCTSTR strDiskName)
```

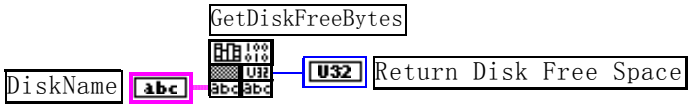
Visual Basic:

```
Declare Function GetDiskFreeBytes Lib "PCI2323" (ByVal strDiskName As String ) As Currency
```

Delphi:

```
Function GetDiskFreeBytes (strDiskName : String) : Currency;
                        Stdcall; external 'PCI2323' 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：strDiskName 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 "PCI2323" ( 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 'PCI2323' 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\PCI2323\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 "PCI2323" (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 'PCI2323' 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\PCI2323\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 "PCI2323" (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 'PCI2323' 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\PCI2323\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 "PCI2323" (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 'PCI2323' 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\PCI2323\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 "PCI2323" (ByVal strFuncName As String,_
ByVal strErrorMsg As String) As Long
Delphi:
Function GetLastErrorEx (strFuncName: String;
strErrorMsg: String) : LongWord;
Stdcall; external 'PCI2323' 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，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 "PCI2323" (ByVal strFuncName As String) As Boolean
Delphi:
Function RemoveLastErrorEx (strFuncName: String) : Boolean;
Stdcall; external 'PCI2323' 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，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 "PCI2323" (ByVal hDevice As Long, _
                                            ByVal nTimeUs As Long) As Boolean
```

Delphi:

```
Function DelayTimeUs (hDevice: Integer;
                      nTimeUs : LongInt) : Boolean;
                      StdCall; External 'PCI2323' 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。