PCI2007 数据采集卡

WIN2000/XP 驱动程序使用说明书

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 PCI2007_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。诸如InitDeviceProDA、WriteDeviceProDA等。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，您基本上可以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。因为上层函数的命名、参数的命名极其规范。

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绝对控制权。然后将此句柄作为参数传递给其他函数，如InitDeviceProDA可以使用hDevice句柄以程序查询方式初始化设备的DA部件，WriteDeviceProDA函数可以用hDevice句柄实现对DA数据的采样读取。最后可以通过ReleaseDevice将hDevice释放掉。

第三节、如何实现 DA 波形数据输出

当您有了hDevice设备对象句柄后，便可用InitDeviceProDA函数初始化DA部件，关于频率等参数的设置是

由这个函数的pDAPara参数结构体决定的。您只需要对这个pDAPara参数结构体的各个成员简单赋值即可实现所有硬件参数和设备状态的初始化。然后调用WriteDeviceProDA将准备好的DA数据写入板载RAM中，接着用StartDeviceProDA即可启动DA部件，开始DA输出。当您需要暂停设备时，执行StopDeviceProDA，当您需要关闭DA设备时，ReleaseDeviceDA便可帮您实现（但设备对象hDevice依然存在）。

第四节、哪些函数对您不是必须的

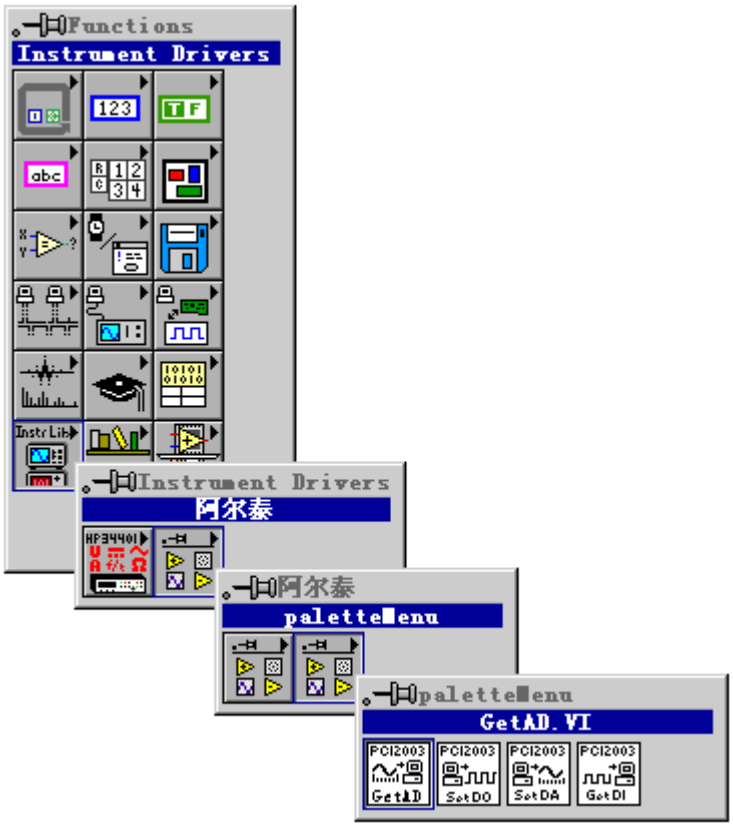
公共函数如CreateFileObject， WriteFile，ReadFile等一般来说都是辅助性函数，除非您要使用存盘功能。如果您使用上层用户函数访问设备，那么GetDeviceAddr， WriteRegisterByte， WriteRegisterWord，WriteRegisterULong， ReadRegisterByte， ReadRegisterWord， ReadRegisterULong等函数您可完全不必理会，除非您是作为底层用户管理设备。而WritePortByte， WritePortWord， WritePortULong， ReadPortByte，ReadPortWord， 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“PCI2007_”）

函数名	函数功能	备注
① 设备对象操作函数		
CreateDevice	创建 PCI 设备对象(用设备逻辑号)	上层及底层用户
GetDeviceCount	取得同一种 PCI 设备的总台数	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
ListDeviceDlgEx	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	上层及底层用户
② DA 数据采样操作函数		
ResetDeviceDA	复位整个 DA 设备状态	上层用户
SetDeviceFreqDA	可动态改变 DA 采样频率	上层用户
ClearFIFODA	清除 FIFO 中的数据	上层用户
InitDeviceProDA	初始化 PCI 设备上的 DA 部件准备传输	上层用户
StartDeviceProDA	启动 DA 设备，开始转换	上层用户
SetoutputRangeDA	设置输出范围 DA	上层用户
StopDeviceProDA	暂停 DA 设备	上层用户
GetDeviceStatusProDA	取得 FIFO 的状态	上层用户
WriteDeviceProDA	向 DA 的 FIFO 中写入批量数据	上层用户
ReleaseDeviceDA	释放 DA 设备	上层用户

使用需知：

Visual C++：

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:\PCI\PCI2007\INCLUDE\PCI2007.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 PCI2007.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 “PCI2007.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。

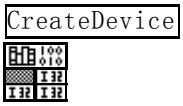
Visual Basic:

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 PCI2007.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。

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 PCI2007.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（逻辑号）**

函数原型：

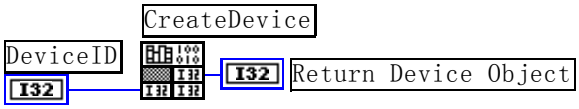
Visual C++：

HANDLE CreateDevice (int DeviceLgcID = 0)

Visual Basic：

Declare Function CreateDevice Lib "PCI2007" (Optional ByVal DeviceLgcID As Integer = 0) As Long

LabVIEW：



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

参数：

DeviceLgcID 逻辑设备 ID(Logic Device Identifier)标识号。当向同一个 Windows 系统中加入若干相同类型的 PCI 设备时，我们的驱动程序将以该设备的“基本名称”与 DeviceLgcID 标识值为后缀的标识符来确认和管理该设备。比如若用户往 Windows 系统中加入第一个 PCI2007 模板时，驱动程序逻辑号为“0”来确认和管理第一个设备，若用户接着再添加第二个 PCI2007 模板时，则系统将以逻辑号“1”来确认和管理第二个设备，若再添加，则以此类推。所以当用户要创建设备句柄管理和操作第一个 PCI 设备时，DeviceLgcID 应置 0，第二个应置 1，也以此类推。但默认值为 0。该参数之所以称为逻辑设备号，是因为每个设备的逻辑号是不能事

先由用户硬性确定的，而是由 BIOS 和操作系统加载设备时，依据主板总线编号等信息进行这个设备 ID 号分配，说得简单点，就是加载设备的顺序编号，编号的递增顺序为 0、1、2、3……。所以用户无法直接固定某一个设备的在设备列表中的物理位置，若想固定，则必须须使用物理 ID 号。

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： CreateDevice GetDeviceCount ListDeviceDlg ListDeviceDlgEx ReleaseDevice

Visual C++程序举例

```
    :
HANDLE hDevice;     // 定义设备对象句柄
int DeviceLgcID = 0;
hDevice = CreateDevice ( DeviceLgcID ); // 创建设备对象,并取得设备对象句柄
if(hDevice == INVALIDE_HANDLE_VALUE); // 判断设备对象句柄是否有效
{
    return;     // 退出该函数
}
    :
```

Visual Basic 程序举例

```
    :
    Dim hDevice As Long   ' 定义设备对象句柄
Dim DeviceLgcID As Long
DeviceLgcID = 0
hDevice = CreateDevice ( DeviceLgcID )   ' 创建设备对象,并取得设备对象句柄
If hDevice = INVALID_HANDLE_VALUE Then   ' 判断设备对象句柄是否有效
    MsgBox “创建设备对象失败”
    Exit Sub      ' 退出该过程
End If
    :
```

◆ **取得本计算机系统中 PCI2007 设备的总数量**

函数原型：

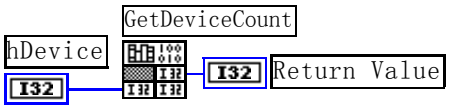
Visual C++:

int GetDeviceCount (HANDLE hDevice)

Visual Basic:

Declare Function GetDeviceCount Lib "PCI2007" (ByVal hDevice As Long) As Integer

LabVIEW：



功能：取得 PCI2007 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 PCI2007 的数量。

相关函数： CreateDevice GetDeviceCount ListDeviceDlg ListDeviceDlgEx ReleaseDevice

◆ **用对话框控件列表计算机系统中所有 PCI2007 设备各种配置信息**

函数原型：

Visual C++:

BOOL ListDeviceDlg (HANDLE hDevice)

Visual Basic:

Declare Function ListDeviceDlg Lib "PCI2007" (ByVal hDevice As Long) As Boolean

LabVIEW：

请参考相关演示程序。

功能：列表系统中 PCI2007 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 PCI2007 设备的配置情况。

相关函数： CreateDevice ReleaseDevice

◆ **用对话框控件列表计算机系统中所有 PCI2007 设备各种配置信息**

函数原型：

Visual C++:

BOOL ListDeviceDlgEx (HANDLE hDevice)

Visual Basic:

Declare Function ListDeviceDlgEx Lib "PCI2007" (ByVal hDevice As Long) As Boolean

LabVIEW：

请参考相关演示程序。

功能：列表系统中 PCI2007 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 PCI2007 设备的配置情况。

相关函数：CreateDevice　　ReleaseDevice

◆ **释放设备对象所占的系统资源及设备对象**

函数原型：

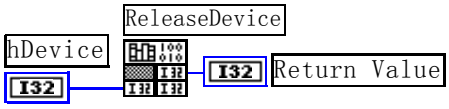
Visual C++:

BOOL ReleaseDevice(HANDLE hDevice)

Visual Basic:

Declare Function ReleaseDevice Lib "PCI2007" (ByVal hDevice As Long) As Boolean

LabVIEW：



功能：释放设备对象所占用的系统资源及设备对象自身。

参数：hDevice设备对象句柄，它应由CreateDevice创建。

返回值：若成功，则返回TRUE， 否则返回FALSE， 用户可以用GetLastErrorEx捕获错误码。

相关函数：CreateDevice

应注意的是，CreateDevice必须和ReleaseDevice函数一一对应，即当您执行了一次CreateDevice后，再一次执行这些函数前，必须执行一次ReleaseDevice函数，以释放由CreateDevice占用的系统软硬件资源，如DMA控制器、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、DA 数据采样操作函数原型说明

◆ **复位整个 DA 设备状态**

函数原型：

Visual C++:

BOOL ResetDeviceDA (HANDLE hDevice,
PPCI2007_PARA_DA pDAPara)

Visual Basic:

Declare Function ResetDeviceDA Lib "PCI2007" (ByVal hDevice as Long,_
ByRef pDAPara As PCI2007_PARA_DA) As Boolean

LabVIEW：

请参考演示源程序。

功能：复位整个 DA 设备状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pDAPara 设备对象参数结构，它决定了设备对象的各种状态及工作方式，如采样频率等。关于具体操作请参考《DA硬件参数结构》。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，用户可用 GetLastError 捕获当前错误码，并加以分析。

相关函数： CreateDevice　　ResetDeviceDA　　SetDeviceFreqDA

ClearFIFODA InitDeviceProDA StartDeviceProDA
StopDeviceProDA GetDeviceStatusProDA WriteDeviceProDA
ReleaseDeviceDA ReleaseDevice

♦ **动态改变采样频率**

函数原型：

Visual C++:

```
BOOL SetDeviceFreqDA (HANDLE hDevice,
                      ULONG Frequency,
                      int nTimerChannel)
```

Visual Basic:

```
Declare Function SetDeviceFreqDA Lib "PCI2007" (ByVal hDevice as Long, _
                                                ByVal Frequency As Long,_
                                                ByVal nTimerChannel As Integer) As Boolean
```

LabVIEW：

请参考演示源程序。

功能：在 DA 采样过程中，可动态改变采样频率。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

Frequency 时钟输出频率，单位为 Hz。

nTimerChannel 时钟通道号，取值为[0, 1]。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，用户可用 GetLastError 捕获当前错误码，并加以分析。

相关函数： CreateDevice ResetDeviceDA SetDeviceFreqDA
ClearFIFODA InitDeviceProDA StartDeviceProDA
StopDeviceProDA GetDeviceStatusProDA WriteDeviceProDA
ReleaseDeviceDA ReleaseDevice

♦ **清除 FIFO 中的数据**

函数原型：

Visual C++:

```
BOOL ClearFIFODA (HANDLE hDevice,
                  int nDAChannel)
```

Visual Basic:

```
Declare Function ClearFIFODA Lib "PCI2007" (ByVal hDevice as Long,_
                                            ByVal nDAChannel As Integer) As Boolean
```

LabVIEW：

请参考演示源程序。

功能：清除 FIFO 中的数据。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nDAChannel DA 通道号，取值为[0, 3]。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE， 用户可用 GetLastError 捕获当前错误码，并加以分析。

相关函数： CreateDevice ResetDeviceDA SetDeviceFreqDA
ClearFIFODA InitDeviceProDA StartDeviceProDA
StopDeviceProDA GetDeviceStatusProDA WriteDeviceProDA
ReleaseDeviceDA ReleaseDevice

♦ **初始化设备对象**

函数原型：

Visual C++:

```
BOOL InitDeviceProDA (HANDLE hDevice,
                      LONG ClockSource,
                      int nDAChannel)
```

Visual Basic:

```
Declare Function InitDeviceProDA Lib "PCI2007" (ByVal hDevice As Long, _
                                                ByRef ClockSource As Long,_
                                                ByVal nDAChannel As Integer) As Boolean
```

LabVIEW：

请参考相关演示程序。

功能：它负责初始化设备对象中的DA部件，为设备操作就绪有关工作做准备，如预置DA采集通道、采样频率等。但它并不启动DA设备，若要启动DA设备，须在调用此函数之后再调用StartDeviceProDA (但DA要实际输出波形，则一般要等待某种触发事件的到来)。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

ClockSource 时钟源选择，其取值为：

常量名	常量值	功能定义
PCI2007_IN_CLOCK0	0x0000	使用本设备上的内部时钟 0
PCI2007_IN_CLOCK1	0x0001	使用本设备上的内部时钟 1
PCI2007_OUT_CLOCK	0x0002	使用外部时钟

nDAChannel DA 通道号，取值为[0, 3]。

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 GetLastError 捕获当前错误码，并加以分析。

相关函数： CreateDevice, ResetDeviceDA, SetDeviceFreqDA, ClearFIFODA, InitDeviceProDA, StartDeviceProDA, StopDeviceProDA, GetDeviceStatusProDA, WriteDeviceProDA, ReleaseDeviceDA, ReleaseDevice

◆ **启动 DA 设备**

函数原型：

Visual C++:

```
BOOL StartDeviceProDA (HANDLE hDevice,
                       int nDAChannel)
```

Visual Basic:

```
Declare Function StartDeviceProDA Lib "PCI2007" (ByVal hDevice As Long,_
                                                 ByVal nDAChannel As Integer) As Boolean
```

LabVIEW：

请参考相关演示程序。

功能：启动DA设备，它必须在调用InitDeviceProDA后才能调用此函数。调用该函数后它可能立即启动，这就要取决您选择的触发方式或触发源，详细请参考后面的《触发功能详述》。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nDAChannel 设备通道号，取值范围为[0, 3]。

返回值：如果调用成功，则返回 TRUE，且 DA 准备就绪，等待触发事件的到来就开始实际的 DA 输出， 否则返回 FALSE，用户可用 GetLastError 捕获当前错误码，并加以分析。

相关函数： CreateDevice, ResetDeviceDA, SetDeviceFreqDA, ClearFIFODA, InitDeviceProDA, StartDeviceProDA, StopDeviceProDA, GetDeviceStatusProDA, WriteDeviceProDA, ReleaseDeviceDA, ReleaseDevice

◆ **暂停 DA 设备**

函数原型：

Visual C++:

```
BOOL StopDeviceProDA (HANDLE hDevice,
                      int nDAChannel)
```

Visual Basic:

```
Declare Function StopDeviceProDA Lib "PCI2007" (ByVal hDevice as Long,_
```

ByVal nDAChannel As Integer) As Boolean

LabVIEW：

请参考相关演示程序。

功能：暂停DA设备。它必须在调用StartDeviceProDA后才能调用此函数。该函数除了停止DA设备不再转换以外，不改变设备的其他任何工作参数。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nDAChannel 设备通道号，取值范围为[0, 3]。

返回值：如果调用成功，则返回 TRUE，且 DA 立刻停止转换，否则返回 FALSE，用户可用 GetLastError 捕获当前错误码，并加以分析。

相关函数： CreateDevice, ResetDeviceDA, SetDeviceFreqDA, ClearFIFODA, InitDeviceProDA, StartDeviceProDA, StopDeviceProDA, GetDeviceStatusProDA, WriteDeviceProDA, ReleaseDeviceDA, ReleaseDevice

♦取得 DA 的状态标志

函数原型：

Visual C++:

```
BOOL GetDeviceStatusProDA (HANDLE hDevice,
                           BOOL bNotEmpty[4],
                           BOOL bHalf[4],
                           BOOL bOverflow[4])
```

Visual Basic:

```
Declare Function GetDeviceStatusProDA Lib "PCI2007" (ByVal hDevice As Long,_
                                                     ByVal bNotEmpty(0 to 3) As Boolean,_
                                                     ByVal bHalf (0 to 3) As Boolean,_
                                                     ByVal bOverflow (0 to 3) As Boolean) As Boolean
```

LabVIEW：

请参考相关演示程序。

功能：在 AD 采样过程中取得 FIFO 的状态，通常用于半满方式读取。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bNotEmpty 取得非空状态，TRUE 表示已非空有效，FALSE 表示非空无效。

bHalf 取得半满状态，TRUE 表示已半满有效，FALSE 表示半满以下。

bOverflow 取得溢出状态，TRUE 表示已溢出，FALSE 表示未溢出。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，用户可用 GetLastError 捕获当前错误码，并加以分析。

相关函数： CreateDevice, ResetDeviceDA, SetDeviceFreqDA, ClearFIFODA, InitDeviceProDA, StartDeviceProDA, StopDeviceProDA, GetDeviceStatusProDA, WriteDeviceProDA, ReleaseDeviceDA, ReleaseDevice

♦向 DA 的 FIFO 中写入批量数据（通常为 FIFO 的半满长度）

函数原型：

Visual C++:

```
BOOL WriteDeviceProDA (HANDLE hDevice,
                       PLONG pDABuffer,
                       ULONG nWriteSizeWords,
                       int nDAChannel)
```

Visual Basic:

```
Declare Function WriteDeviceProDA Lib "PCI2007" (ByVal hDevice As Long,_
                                                 ByRef pDABuffer As Long,_
                                                 ByVal nWriteSizeWords As Long,_
                                                 ByVal nDAChannel As Integer) As Boolean
```

LabVIEW：

请参考相关演示程序。

功能：向 DA 的 FIFO 中写入批量数据（通常为 FIFO 的半满长度）。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
pDABuffer DA 数据用户缓冲区。
nWriteSizeWords 写入的点数（以字为单位）。
nDAChannel 设备通道号，取值范围为[0, 3]。
返回值：如果调用成功，则返回 TRUE，且 DA 立刻停止转换，否则返回 FALSE，用户可用 GetLastError 捕获当前错误码，并加以分析。

相关函数：

CreateDevice	ResetDeviceDA	SetDeviceFreqDA
ClearFIFODA	InitDeviceProDA	StartDeviceProDA
StopDeviceProDA	GetDeviceStatusProDA	WriteDeviceProDA
ReleaseDeviceDA	ReleaseDevice	

◆ **释放 DA 设备**

函数原型：

Visual C++:

```
BOOL ReleaseDeviceProDA (HANDLE hDevice,
                         int nDAChannel)
```

Visual Basic:

```
Declare Function ReleaseDeviceProDA Lib "PCI2007" (ByVal hDevice as Long, _
                                                   ByVal nDAChannel As Integer) As Boolean
```

LabView：

请参考相关演示程序。

功能：释放DA设备。它必须在调用InitDeviceProDA后的某个时刻调用此函数。该函数除了停止DA设备，还释放掉所占用的各种资源。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
nDAChannel 设备通道号，取值范围为[0, 3]。
返回值：如果调用成功，则返回 TRUE，且 DA 立刻停止转换，否则返回 FALSE，用户可用 GetLastError 捕获当前错误码，并加以分析。

相关函数：

CreateDevice	ResetDeviceDA	SetDeviceFreqDA
ClearFIFODA	InitDeviceProDA	StartDeviceProDA
StopDeviceProDA	GetDeviceStatusProDA	WriteDeviceProDA
ReleaseDeviceDA	ReleaseDevice	

第四章 硬件参数结构

DA 硬件参数结构（PCI2007_PARA_DA）

Visual C++：

```
typedef struct _PCI2007_PARA_DA
{
    LONG bEnableTimer0;      // 1:允许时钟 0 工作; 0:禁止时钟 0 工作
    LONG Frequency0;         // 时钟 0 输出频率
    LONG bEnableTimer1;      // 1:允许时钟 1 工作; 0:禁止时钟 1 工作
    LONG Frequency1;         // 时钟 1 输出频率
    LONG TriggerSource;      // 内触发和外触发方式选择
    LONG OutTriggerEdge;     // 外触发上升沿和下降沿类型选择
    LONG bClockOutput;       // 是否允许时钟输出
    LONG bImmediateClkOut;   // 是否立即启动定时器输出
```

```
    LONG ClockOutputNum;     // 选择哪一路时钟源作为输出(0:Timer0; 1:Timer1)
} PCI2007_PARA_DA, *PPCI2007_PARA_DA;
```

Visual Basic：

```
Type PCI2007_PARA_DA
    bEnableTimer0 As Long
    Frequency0 As Long
    bEnableTimer1 As Long
    Frequency1 As Long
    TriggerSource As Long
    OutTriggerEdge As Long
    bClockOutput As Long
    bImmediateClkOut As Long
    ClockOutputNum As Long
End Type
```

LabVIEW:

请参考相关演示程序。

此结构主要用于设定设备DA硬件参数值，用这个参数结构对设备进行硬件配置完全由InitDeviceProDA自动完成。用户只需要对这个结构体中的各成员简单赋值即可。

bEnableTimer0 时钟 0 工作选择。=1，允许时钟 0 工作；=0：禁止时钟 0 工作。
Frequency0 时钟 0 输出频率。
bEnableTimer1 时钟 1 工作选择。=1，允许时钟 1 工作；=0：禁止时钟 1 工作。
Frequency1 时钟 1 输出频率。
TriggerSource 内触发和外触发方式选择。

常量名	常量值	功能定义
PCI2007_IN_TRIGGER	0x00	内触发启动方式
PCI2007_OUT_TRIGGER	0x01	外触发启动方式

OutTriggerEdge 外触发上升沿和下降沿类型选择。

常量名	常量值	功能定义
PCI2007_FALLING_EDGE	0x00	外触发下降沿触发方式
PCI2007_RISING_EDGE	0x01	外触发上升沿触发方式

bClockOutput 是否允许时钟输出。

常量名	常量值	功能定义
PCI2007_DISABLE_CLOCK_OUT	0x00	禁止本卡上的自带时钟向外输出
PCI2007_ENABLE_CLOCK_OUT	0x01	允许本卡上的自带时钟向外输出

bImmediateClkOut 是否立即启动定时器输出。
ClockOutputNum 选择哪一路时钟源作为输出(=0：Timer0；=1：Timer1)。

第五章 数据格式转换与排列规则

第一节、DA 电压值转换成 LSB 原码数据的换算方法

量程(伏)	计算机语言换算公式(标准 C 语法)	Lsb 取值范围
0～5000mV	Lsb = Volt / (5000.00/65536)	[0，65535]
0～10000mV	Lsb = Volt / (10000.00/65536)	[0，65535]
±5000mV	Lsb = Volt / (10000.00/65536) + 32768	[0，65535]

±10000mV	Lsb = Volt / (20000.00/65536) + 32768	[0，65535]

第二节、关于 DA 数据 DABuffer 缓冲区中的数据排放规则

由于各个通道的段信息与波形数据均共享一个板载物理 RAM，它们的排放顺序如图 5.1，但各个通道所占 RAM 空间大小均可以通过函数 SetDevRamSizeDA()调整，系统默认值为四个通道均分整个 RAM 空间，即默认每通道 RAM 空间为 256K 点。从下图可以看出，各个通道所占 RAM 空间不一定相等，可大可小，只是四个通道的总空间不能大于板载物理 RAM 空间即可。

通道 0	通道 1	通道 2	通道 3
0 至(256K-1)	256 至(512K-1)	512 至(768K-1)	768K 至(1024K-1)

图 5.1

关于每个通道 RAM 空间的内部分配是这样的，其空间首部存放的是所有段的段信息数据，其后才是各个段的波形数据，再其后可能还有未用空间。假如有三个分段，如图 5.2：

段 0 信息	段 1 信息	段 2 信息	段 0 波形数据	段 1 波形数据	段 2 波形数据	未用空间

图 5.2

关于每个段的段信息包括的内容有：该段波形数据在 RAM 中的起始地址、终止地址、段循环次数，如表 5.2.1，注意其段起始地址和终止地址是由 PCI2007_PARA_DA 中的 SegmentInfo 决定的。

表 5.2.1

板载 RAM 内存单元(16Bit)	各单元定义	有效位
0	段 0 波形数据起始地址低 12 位	D11:D0
1	段 0 波形数据起始地址高 8 位	D7:D0
2	段 0 波形数据终止地址低 12 位	D11:D0
3	段 0 波形数据终止地址高 8 位	D7:D0
4	段 0 循环次数低 12 位	D11:D0
5	段 0 循环次数高 8 位	D7:D0
6	段 1 波形数据起始地址低 12 位	D11:D0
7	段 1 波形数据起始地址高 8 位	D7:D0
8	段 1 波形数据终止地址低 12 位	D11:D0
9	段 1 波形数据终止地址高 8 位	D7:D0
10	段 1 循环次数低 12 位	D11:D0
11	段 1 循环次数高 8 位	D7:D0
:	:	:
段信息结束后便是波形数据	段信息结束后便是波形数据	D11:D0

第六章 上层用户函数接口应用实例

第一节、简易程序演示说明

一、怎样使用WriteDeviceProDA函数进行批量DA数据输出

其详细应用实例及工程级代码请参考 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 PCI2007.h 和 Sys.cpp)。

[程序]｜**[阿尔泰测控演示系统]**｜**[Microsoft Visual C++]**｜[简易代码演示]｜[DA 批量输出演示源程序]

其简易程序默认存放路径为：系统盘\PCI\PCI2007\SAMPLES\VC\SIMPLE\DA\BULK

其他语言的演示可以用上面类似的方法找到。

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 PCI2007.h 和 DADoc.cpp)。

[程序]｜**[阿尔泰测控演示系统]**｜**[Microsoft Visual C++]**｜[高级代码演示]

其默认存放路径为：系统盘\PCI\ PCI2007\SAMPLES\VC\ADVANCED

其他语言的演示可以用上面类似的方法找到。

第七章 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“PCI2007_”）

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
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	在 VB 中可实现多线程
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
DelayTimeUs	高效高精度延时函数	不消耗 CPU 时间

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：

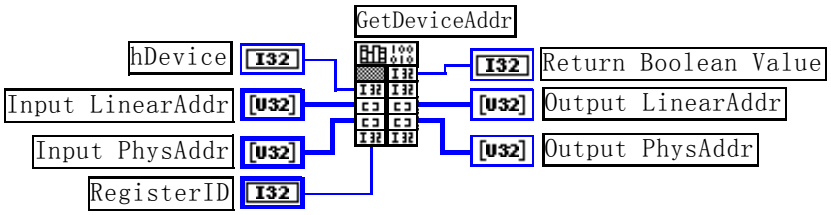
Visual C++:

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

Visual Basic:

```
Declare Function GetDeviceAddr Lib "PCI2007" (ByVal hDevice As Long, _
                                              ByRef LinearAddr As Long, _
                                              ByRef PhysAddr As Long, _
                                              ByVal RegisterID As Integer = 0) As Boolean
```

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。本设备的寄存器组 ID 定义如下：

常量名	常量值	功能定义
PCI2007_REG_RAM_PLXCHIP	0x0000	0 号寄存器对应 PLX 芯片所使用的内存模式基地址(使用 LinearAddr)
PCI2007_REG_IO_PLXCHIP	0x0001	1 号寄存器对应 PLX 芯片所使用的 IO 模式基地址(使用 PhysAddr)
PCI2007_REG_RAM_CPLD	0x0002	2 号寄存器对应板上控制单元所使用的内存模式基地址(使用 LinearAddr)
PCI2007_REG_RAM_DABUFFER	0x0003	3 号寄存器对应板上 DA 缓冲区所使用的内存模式基地址(使用 LinearAddr)

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++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
    :
```

◆ **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：

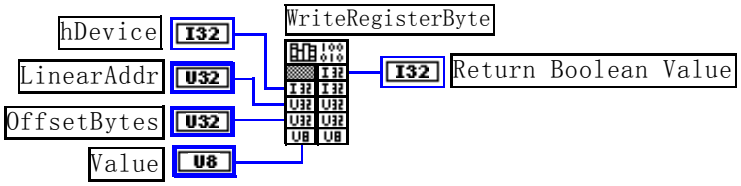
Visual C++:

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

Visual Basic:

```
Declare Function WriteRegisterByte Lib "PCI2007" (ByVal hDevice As Long, _
                                                  ByVal LinearAddr As Long, _
                                                  ByVal OffsetBytes As Long, _
                                                  ByVal Value As Byte ) As Boolean
```

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++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++:

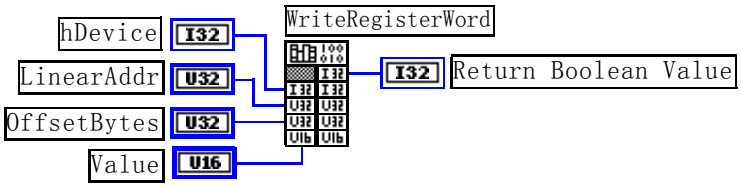
```
BOOL WriteRegisterWord( HANDLE hDevice,
                        ULONG LinearAddr,
                        ULONG OffsetBytes,
```

WORD Value)

Visual Basic:

```
Declare Function WriteRegisterWord Lib "PCI2007" (ByVal hDevice As Long, _
                                                  ByVal LinearAddr As Long, _
                                                  ByVal OffsetBytes As Long, _
                                                  ByVal Value As Integer) As Boolean
```

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++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++:

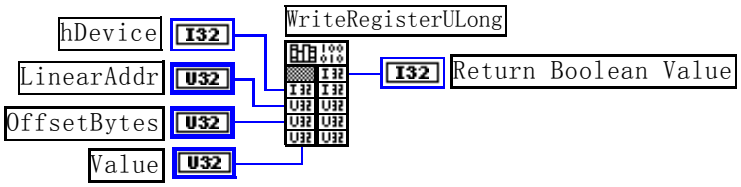
```
BOOL WriteRegisterULong( HANDLE hDevice,
                         ULONG LinearAddr,
                         ULONG OffsetBytes,
                         ULONG Value)
```

Visual Basic:

```
Declare Function WriteRegisterULong Lib "PCI2007" (ByVal hDevice As Long, _
                                                   ByVal LinearAddr As Long, _
                                                   ByVal OffsetBytes As Long, _
```

ByVal Value As Long) As Boolean

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++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：

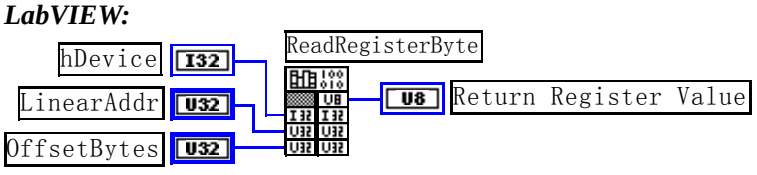
Visual C++:

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

Visual Basic:

```
Declare Function ReadRegisterByte Lib "PCI2007" (ByVal hDevice As Long, _
                                                 ByVal LinearAddr As Long, _
                                                 ByVal OffsetBytes As Long) As Byte
```

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++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：

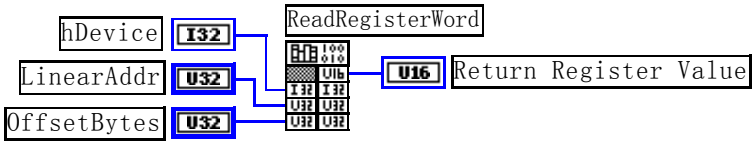
Visual C++:

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

Visual Basic:

```
Declare Function ReadRegisterWord Lib "PCI2007" (ByVal hDevice As Long, _
                                                 ByVal LinearAddr As Long, _
                                                 ByVal OffsetBytes As Long) As Integer
```

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++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：

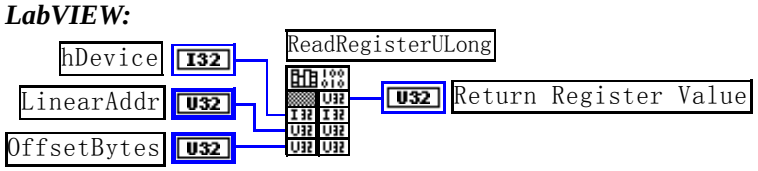
Visual C++:

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

Visual Basic:

```
Declare Function ReadRegisterULong Lib "PCI2007" (ByVal hDevice As Long, _
                                                  ByVal LinearAddr As Long, _
                                                  ByVal OffsetBytes As Long) As Long
```

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++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**

函数原型：

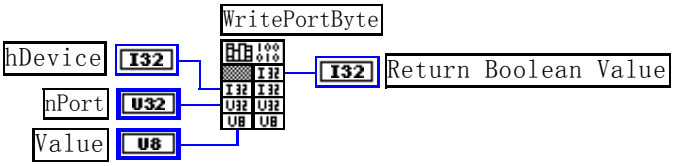
Visual C++:

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

Visual Basic:

```
Declare Function WritePortByte Lib "PCI2007" ( ByVal hDevice As Long, _
                                               ByVal nPort As Long, _
                                               ByVal Value As Byte) As Boolean
```

LabVIEW:



功能：以单字节(8Bit)方式写 I/O 端口。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

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：

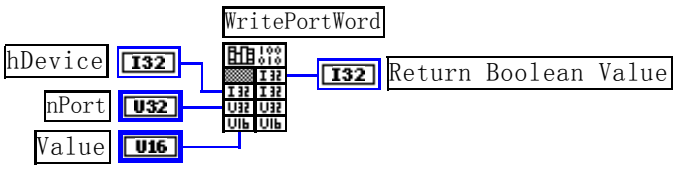
Visual C++:

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

Visual Basic:

```
Declare Function WritePortWord Lib "PCI2007" (ByVal hDevice As Long, _
                                              ByVal nPort As Long, _
                                              ByVal Value As Integer) As Boolean
```

LabVIEW:



功能：以双字(16Bit)方式写 I/O 端口。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
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**

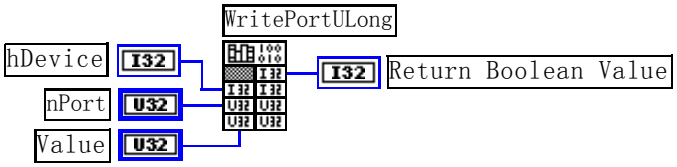
函数原型：
Visual C++:

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

Visual Basic:

```
Declare Function WritePortULong Lib "PCI2007" (ByVal hDevice As Long, _
                                               ByVal nPort As Long, _
                                               ByVal Value As Long ) As Boolean
```

LabVIEW:



功能：以四字节(32Bit)方式写 I/O 端口。
参数：
hDevice　设备对象句柄，它应由CreateDevice创建。
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**

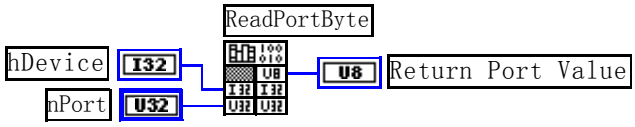
函数原型：
Visual C++:

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

Visual Basic:

```
Declare Function ReadPortByte Lib "PCI2007" (ByVal hDevice As Long, _
                                             ByVal nPort As Long ) As Byte
```

LabVIEW:

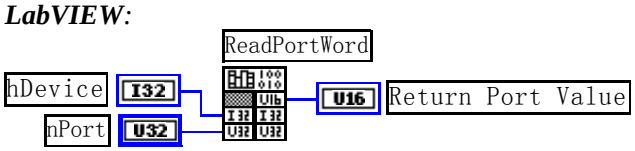


功能：以单字节(8Bit)方式读 I/O 端口。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。

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**

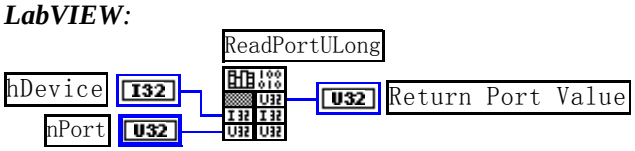
函数原型：
Visual C++:
```
WORD ReadPortWord(HANDLE hDevice,
                  UINT nPort)
```
Visual Basic:
```
Declare Function ReadPortWord Lib "PCI2007" ( ByVal hDevice As Long, _
                                              ByVal nPort As Long ) As Integer
```
LabVIEW:



功能：以双字节(16Bit)方式读 I/O 端口。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
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++:
```
ULONG ReadPortULong(HANDLE hDevice,
                    UINT nPort)
```
Visual Basic:
```
Declare Function ReadPortULong Lib "PCI2007" ( ByVal hDevice As Long, _
                                               ByVal nPort As Long ) As Long
```
LabVIEW:



功能：以四字节(32Bit)方式读 I/O 端口。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
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++:
```
BOOL CreateVBThread(HANDLE *hThread,
                    LPTHREAD_START_ROUTINE lpStartThread)
```

Visual Basic

```
Declare Function CreateVBThread Lib "PCI2007" ( ByRef hThread As Long, _
                                                ByVal lpStartThread 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。

lpStartThread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 PCI2007.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++:

```
BOOL TerminateVBThread(HANDLE hThreadHandle)
```

Visual Basic:

```
Declare Function TerminateVBThread Lib "PCI2007" (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：hThread 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：

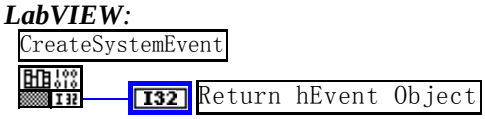
Visual C++:

```
HANDLE CreateSystemEvent(void)
```

Visual Basic:

Declare Function CreateSystemEvent Lib " PCI2007 " () As Long

LabVIEW:



功能：创建系统内核事件对象，它将被用于中断事件响应或数据采集线程同步事件。

参数：无任何参数。

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

- **释放内核系统事件**

函数原型：

Visual C++ & C++ Builder:

BOOL ReleaseSystemEvent(HANDLE hEvent);

Visual Basic:

Declare Function ReleaseSystemEvent Lib " PCI2007 " (ByVal hEvent As Long) As Boolean

Delphi:

Function ReleaseSystemEvent(hEvent : Integer) : Integer;
StdCall; External 'PCI2007' 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。

- **高效高精度延时**

函数原型：

Visual C++:

BOOL DelayTimeUs (HANDLE hDevice,
LONG nTime)

Visual Basic:

Declare Function DelayTimeUs Lib "PCI2007" (ByVal hDevice As Long, _
ByVal nTime As Long) As Boolean

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。

nTime 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。

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++:

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

Visual Basic:

Declare Function CreateFileObject Lib "PCI2007" (ByVal hDevice As Long, _
ByVal NewFileName As String, _
ByVal Mode As Integer) As Long

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。
strFileName 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:\\PCI2007\\Data.Dat”，在 Basic 中，其语法格式如：“C:\PCI2007\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
PCI2007_modeRead	0x0000	只读文件方式
PCI2007_modeWrite	0x0001	只写文件方式
PCI2007_modeReadWrite	0x0002	既读又写文件方式
PCI2007_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

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++:

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

Visual Basic:

```
Declare Function WriteFile Lib "PCI2007" ( ByVal hFileObject As Long,_
                                           ByRef pDataBuffer As Integer,_
                                           ByVal nWriteSizeBytes As Long) As Boolean
```

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++:

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

Visual Basic:

```
Declare Function ReadFile Lib "PCI2007" (ByVal hFileObject As Long, _
                                         ByRef pDataBuffer As Integer, _
                                         ByVal OffsetBytes As Long, _
                                         ByVal nReadSizeBytes As Long) As Boolean
```

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++:
```
BOOL SetFileOffset (HANDLE hFileObject,
                    ULONG nOffsetBytes)
```
Visual Basic:
```
Declare Function SetFileOffset Lib "PCI2007" ( ByVal hFileObject As Long,
                                               ByVal nOffsetBytes As Long) As Boolean
```
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。
nOffsetBytes 文件偏移位置（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

- **取得文件长度（字节）**

函数原型：
Visual C++:
```
ULONG GetFileLength (HANDLE hFileObject)
```
Visual Basic:
```
Declare Function GetFileLength Lib "PCI2007" (ByVal hFileObject As Long) As Long
```
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++:
```
BOOL ReleaseFile(HANDLE hFileObject)
```
Visual Basic:
```
Declare Function ReleaseFile Lib "PCI2007" (ByVal hFileObject As Long) As Boolean
```
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：

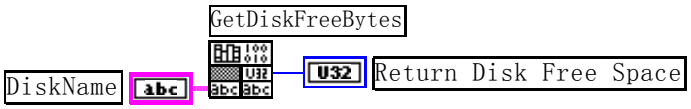
Visual C++:

ULONGLONG GetDiskFreeBytes(LPCTSTR DiskName)

Visual Basic:

Declare Function GetDiskFreeBytes Lib "PCI2007" (ByVal DiskName As String) As Currency

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。