PCI2394 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 PCI2394_CreateDevice 则写为 CreateDevice。

二、函数名及参数中各种关键字缩写规则

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

以上规则不局限于该产品。

第二章 使用纲要

第一节、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InitDeviceCNT等。而底层用户函数如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绝对控制权。然后将此句柄作为参数传递给相应的驱动函数，如GetDeviceCNT可以使用hDevice句柄以读取计数器的值，GetDeviceDI函数可以用hDevice句柄实现对DI数据读取，SetDevDOMode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、如何用读取计数器的值

读取计时器比较简单， 我们先调用CreateDevice来创建一设备对象 创建成功后可以通过设置计数器参数InitDeviceCNT来设置计数器参数来设置计时器参数。然后就可以按要求调用GetDeviceCNT得到计数器的值。

最后调用ReleaseDevice释放掉设备（必需的）。

注意：图中较粗的虚线表示对称关系。如红色虚线表示CreateDevice和ReleaseDevice两个函数的关系是：最初执行一次CreateDevice，在结束是就须执行一次ReleaseDevice。

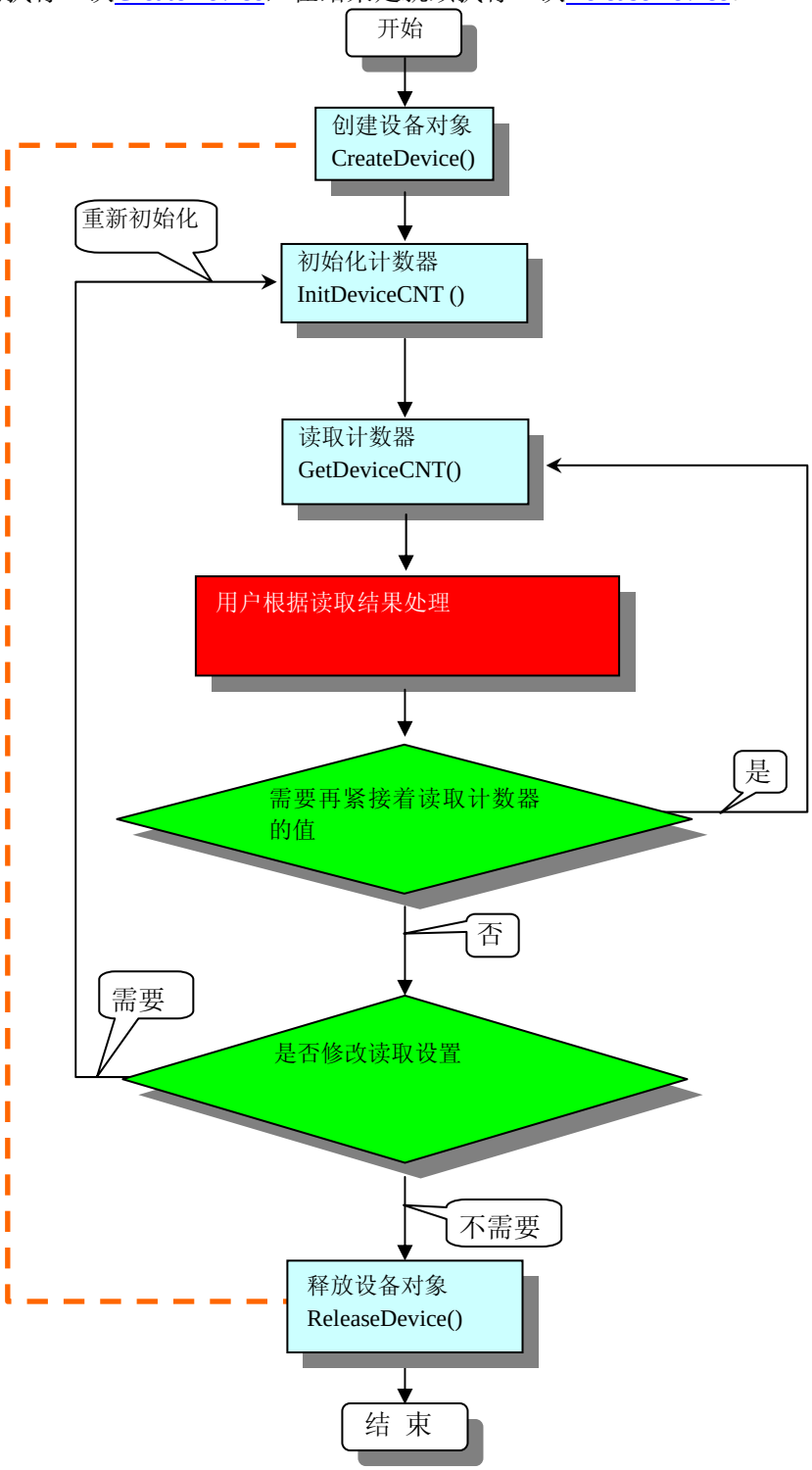


图 2.1 正常方式读取计数器过程

第四节、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SetDevDOMode函数实现设置开关量的输出操作，其各路开关量的输出状态由其byDOSts[8]中的相应元素决定。由GetDeviceDI函数实现开关量的输入操作，其各路开关量的输入状态由其byDISts[8]中的相应元素决定。

第五节、如何使用中断方式

当您有了hDevice设备对象句柄后，便可用InitDeviceInt函数初始化中断。同时应调用CreateSystemEvent函数创建一个内核事件对象句柄hEventInt赋给InitDeviceInt的相应参数，它将作为接受中断事件的变量。然后接着调用Win32 API函数WaitForSingleObject等待hEvent中断事件的发生，在中断未到时，自动使所在线程进入睡眠状态（不消耗CPU时间），反之，则立即唤醒所在线程，执行它下面的代码。当您需要修改中断配置时，执行ModifyDeviceInt，当您需要关闭中断时，ReleaseDeviceInt便可帮您实现。

第六节、哪些函数对您不是必须的

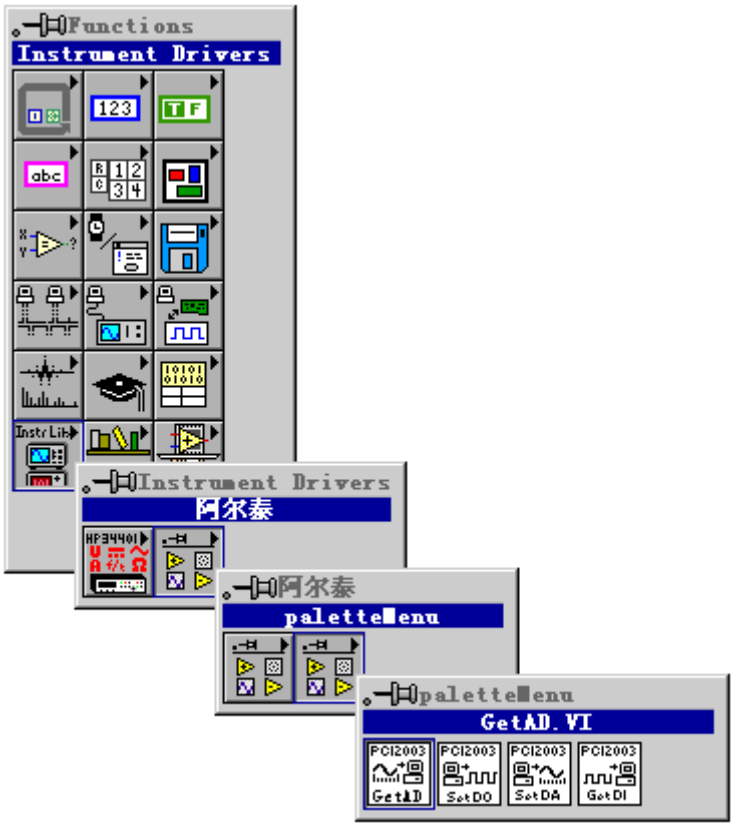
公共函数如CreateFileObject， WriteFile，ReadFile等一般来说都是辅助性函数，除非您要使用存盘功能。如果您使用上层用户函数访问设备，那么GetDeviceAddr等函数您可完全不必理会，除非您是作为底层用户管理设备。而WritePortByte， WritePortWord， WritePortULong， ReadPortByte，ReadPortWord， ReadPortULong则对PCI用户来讲，可以说完全是辅助性，它们只是对我公司驱动程序的一种功能补充，对用户额外提供的，它们可以帮助您在NT、Win2000 等操作系统中实现对您原有传统设备如ISA卡、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、只关心如何方便的读取指定通道的计数器的值。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，我们则把这一群需要直接跟设备端口打交道的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，他们最希望他们在软件操作上所要面对的全是他们最关心的问题，比如在正式读取计数器之前，只须用户调用一个简易的初始化函数（如InitDeviceCNT）告诉设备我要使用计数器模式，锁村模式，然后便可以用GetDeviceCNT函数来读取指定通道的计时器值。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“PCI2394_”）

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
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	上层及底层用户
② 计数器与定时器操作函数		
SetDeviceClock	设置设备的时钟频率	上层用户
GetDeviceClock	取得设备的时钟频率	上层用户
InitDeviceCNT	初始化指定通道计数器	上层用户
SetCNTMode	设置计数器模式	上层用户
GetCNTMode	取得计数器模式	上层用户
SetLatchMode	设置指定通道的锁存模式	上层用户
GetLatchMode	得到指定通道的锁存模式	上层用户
EnableOverflowLock	设定指定是否上溢锁定	
IsOverflowLock	得到指定通道是否上溢锁定	
EnableUnderflowLock	设置指定通道下溢锁定	
IsUnderflowLock	得到指定通道是否下溢锁定	
EnableDigitFilter	设定指定通道是否数字滤波	
IsDigitFilter	得到指定通道是否数字滤波	
GetDeviceCNT	得到指定通道的计数器的值	
SetResetMode	设置指定通道的复位模式	
GetResetMode	得到指定通道的复位模式	
ResetDeviceCNT	复位指定通道的计数器	
SetDevCompValue	设定指定通道的比较值	

GetDevCompValue	得到指定通道的比较值	
SetTimer	设置定时器	
GetTimer	得到定时器	
③ 中断实现计数器控制函数		
InitDeviceInt	初始化中断设置	
ModifyDeviceInt	修改中的设置	
GetDeviceIntSrc	取得中断源	
GetDeviceIntCount	取得中断次数	
ResetDeviceIntSrc	断开中断源	
ReleaseDeviceInt	释放中断	
④ 保存硬件和中断参数函数		
SaveDevicePara	保存计数器所有通道参数	
LoadDevicePara	取得计数器所有通道参数	
SaveDeviceINTPara	保存中断设置	
LoadDeviceINTPara	载入中断设置	
⑤ 数字量输出状态模式控制函数		
GetDeviceDI	读取数字量输入	
SetDevDOMode	设置 DO 模式	
GetDevDOMode	得到 DO 模式	

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\PCI2394\INCLUDE\PCI2394.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 PCI2394.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。

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(PCI2394.H)写进您的源程序头部。如：#include “\Art\PCI2394\Include\PCI2394.h”，然后再将 PCI2394.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 PCI2394.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 PCI2394.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:

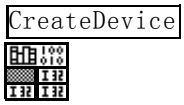
要使用如下函数一个关键的问题是首先必须将我们提供的单元模块文件（*.Pas）加入到您的 Delphi 工程中。其方法是选择 Delphi 编程环境中的 View 菜单，执行其中的"Project Manager"命令，在弹出的对话中选择*.exe 项目，再单击鼠标右键，最后 Add 指令，即可将 PCI2394.Pas 单元模块文件加入到工程中。或者在 Delphi 的编程环境中的 Project 菜单中，执行 Add To Project 命令，然后选择*.Pas 文件类型也能实现单元模块文件的添加。该文件的路径为用户安装驱动程序后其子目录 Samples\Delphi 下面。最后请在使用驱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：“PCI2394”。如：

uses

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
PCI2394;  // 注意：  在此加入驱动程序接口单元 PCI2394
```

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 PCI2394.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，如ReadDeviceProAD_Npt接口单元，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++ & C++Builder：

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

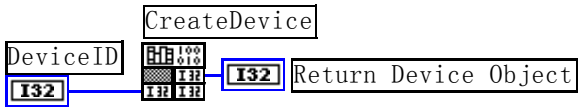
Visual Basic：

```
Declare Function CreateDevice Lib "PCI2394" (Optional ByVal DeviceLgcID As Integer = 0) As Long
```

Delphi：

```
Function CreateDevice(DeviceLgcID : Integer = 0) : Integer;
                    StdCall; External 'PCI2394' 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：

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 PCI2394 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 PCI2394 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，调用函数实现。

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 GetDeviceCurrentID ListDeviceDlg ReleaseDevice

Visual C++ & C++Builder 程序举例

```
    :
HANDLE hDevice;     // 定义设备对象句柄
int DeviceLgcID = 0;
hDevice = PCI2394_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 = PCI2394_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 PCI2394 设备的总数量**

函数原型：

Visual C++ & C++Builder:

LONG GetDeviceCount (HANDLE hDevice)

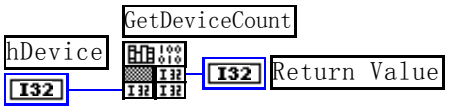
Visual Basic:

Declare Function GetDeviceCount Lib "PCI2394" (ByVal hDevice As Long) As Long

Delphi：

Function GetDeviceCount (hDevice : Integer) : LongInt;
StdCall; External 'PCI2394' Name ' GetDeviceCount ';

LabVIEW:



功能：取得 PCI2394 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 PCI2394 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:

LONG GetDeviceCurrentID (HANDLE hDevice)

Visual Basic:

Declare Function GetDeviceCurrentID Lib "PCI2394" (ByVal hDevice As Long) As Long

Delphi：

Function GetDeviceCurrentID (hDevice : Integer) : LongInt;
StdCall; External 'PCI2394' 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。

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 PCI2394 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 "PCI2394" (ByVal hDevice As Long ) As Boolean
```

Delphi：

```
Function ListDeviceDlg (hDevice : Integer) : Boolean;
                        StdCall; External 'PCI2394' Name ' ListDeviceDlg ';
```

LabVIEW:

请参考相关演示程序。

功能：列表系统中 PCI2394 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 PCI2394 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)
```

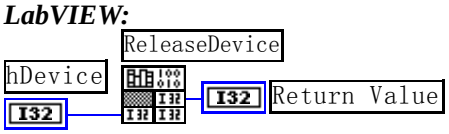
Visual Basic:

```
Declare Function ReleaseDevice Lib "PCI2394" (ByVal hDevice As Long ) As Boolean
```

Delphi：

```
Function ReleaseDevice(hDevice : Integer) : Boolean;
                        StdCall; External 'PCI2394' 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、设置和读取计数器的函数

♦ **设置板卡时钟**

函数原型：

Visual C++ & C++Builder:

```
BOOL SetDeviceClock(HANDLE hDevice,
                    LONG lSampClockSrc)
```

Visual Basic:

```
Declare Function SetDeviceClock Lib "PCI2394" (ByVal hDevice As Long,_
                                               ByVal lSampClockSrc As Long) As Boolean
```

Delphi：

```
Function SetDeviceClock (hDevice : Integer;
                         lSampClockSrc : LongInt) : Boolean;
                         StdCall; External 'PCI2394' Name ' SetDeviceClock ';
```

LabVIEW:

请参考相关演示程序。

功能：设置板卡时钟。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

lSampClockSrc 时钟参数选项。

常量名	常量值	功能定义
PCI2394_CLOCKSRC_8MHz	0x00	8Mhz 时钟
PCI2394_CLOCKSRC_4MHz	0x01	4Mhz 时钟
PCI2394_CLOCKSRC_2MHz	0x02	2Mhz 时钟
PCI2394_CLOCKSRC_1MHz	0x03	1Mhz 时钟

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 GetDeviceClock ReleaseDevice

- **取得板卡时钟**

函数原型：

Visual C++ & C++Builder:

```
LONG GetDeviceClock (HANDLE hDevice)
```

Visual Basic:

```
Declare Function GetDeviceClock Lib "PCI2394" (ByVal hDevice As Long) As Long
```

Delphi：

```
Function GetDeviceClock (hDevice : Integer) : LongInt;
                         StdCall; External 'PCI2394' Name ' GetDeviceClock ';
```

LabVIEW:

请参考相关演示程序。

功能：取得板卡时钟。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

返回值：返回时钟参数选项。

相关函数：CreateDevice SetDeviceClock ReleaseDevice

- **初始化指定通道计数器**

函数原型：

Visual C++ & C++Builder:

```
BOOL InitDeviceCNT(HANDLE hDevice,
                   PPCI2394_PARA_CNT pCNTPara,
                   int nCNTChannel = 0 )
```

Visual Basic:

```
Declare Function InitDeviceCNT Lib "PCI2394" (ByVal hDevice as Long, _
                                              ByRef pCNTPara As PCI2394_PARA_CNT,_
                                              ByVal nCNTChannel As long) As Boolean
```

Delphi:

```
Function InitDeviceCNT (hDevice : Integer;
                        pCNTPara: PPCI2394_PARA_CNT;
                        nCNTChannel: LongWord) : Boolean;
                        StdCall; External 'PCI2394' Name ' InitDeviceCNT ';
```

LabVIEW:

请参考相关演示程序。

功能：初始化计数器的设置。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pCNTPara 指定计数器配置参数。

nCNTChannel 计数器通道号（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 GetDeviceCNT ReleaseDevice

- **设置指定通道计数模式**

函数原型：

Visual C++ & C++Builder:

```
BOOL SetCNTMode(HANDLE hDevice,
                LONG lCNTMode,
                int  nCNTChannel = 0)
```

Visual Basic:

```
Declare Function SetCNTMode Lib "PCI2394" (ByVal hDevice as Long, _
                                           ByVal lCNTMode As Long,
                                           ByVal nCNTChannel As Integer = 0) As Boolean
```

Delphi:

```
Function SetCNTMode (hDevice : Integer;
                     lCNTMode : LongInt;
                     nCNTChannel: Integer = 0) : Boolean;
                     StdCall; External 'PCI2394' Name ' SetCNTMode ';
```

LabVIEW:

请参考相关演示程序。

功能：设置指定通道的计数模式。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

lCNTMode 计数模式。

常量名	常量值	功能定义
PCI2394_CNTMODE_DISABLE	0x00	计数器输入控制模式无效，但可以访问所有的寄存器
PCI2394_CNTMODE_QUADRATURE_X1	0x01	计数器差分输入，Channel A 上升沿有效开始计数
PCI2394_CNTMODE_QUADRATURE_X2	0x02	计数器差分输入，Channel A 只要有跳变就开始计数
PCI2394_CNTMODE_QUADRATURE_X4	0x03	计数器差分输入，Channel A 或 Channel B 只要有跳变就开始计数
PCI2394_CNTMODE_2_PULSE	0x04	双脉冲模式,一个做顺时针计数，另一个做逆时针计数，当 Channel B 在上升沿时有效
PCI2394_CNTMODE_1_PULSE	0x05	A 线为脉冲，B 线为方向

nCNTChannel 计数器通道号（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 InitDeviceCNT GetDeviceCNT
GetCNTMode ReleaseDevice

- **取得指定通道计数模式**

函数原型：

Visual C++ & C++Builder:

```
LONG GetCNTMode (HANDLE hDevice,
                 int  nCNTChannel = 0)
```

Visual Basic:

```
Declare Function GetCNTModeLib "PCI2394" (ByVal hDevice as Long, _
                                          ByVal nCNTChannel As Integer = 0) As Long
```

Delphi:

```
Function GetCNTMode (hDevice : Integer;
                     nCNTChannel: Integer = 0) : LongInt;
                     StdCall; External 'PCI2394' Name ' GetCNTMode';
```

LabVIEW:

请参考相关演示程序。

功能：取得指定通道的计数模式。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

nCNTChannel 计数器通道号（0-3）。

返回值：返回计数模式。

相关函数： CreateDevice InitDeviceCNT GetDeviceCNT GetCNTMode ReleaseDevice

♦ **设置指定通道锁存模式**

函数原型：

Visual C++ & C++Builder:

```
BOOL SetLatchMode(HANDLE hDevice,
                  LONG lLatchMode,
                  int nCNTChannel = 0)
```

Visual Basic:

```
Declare Function SetLatchMode Lib "PCI2394" (ByVal hDevice as Long, _
                                             ByVal lLatchMode As Long,
                                             ByVal nCNTChannel As Integer = 0) As Boolean
```

Delphi:

```
Function SetLatchMode (hDevice : Integer;
                       lLatchMode : LongInt;
                       nCNTChannel: Integer = 0) : Boolean;
                       StdCall; External 'PCI2394' Name ' SetLatchMode ';
```

LabVIEW:

请参考相关演示程序。

功能：设置指定通道的锁存模式。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

lLatchMode 锁存模式。

常量名	常量值	功能定义
PCI2394_LATCHMODE_SOFT	0x00	软件锁存计数器数据 SoftWare
PCI2394_LATCHMODE_IDX	0x01	索引锁存计数器数据 Index
PCI2394_LATCHMODE_TM	0x04	定时器锁存计数器数据 Time
PCI2394_LATCHMODE_DI0	0x10	开关量输入 0 通道锁存计数器数据
PCI2394_LATCHMODE_DI1	0x20	开关量输入 1 通道锁存计数器数据
PCI2394_LATCHMODE_DI2	0x40	开关量输入 2 通道锁存计数器数据
PCI2394_LATCHMODE_DI3	0x80	开关量输入 3 通道锁存计数器数据

nCNTChannel 计数器通道号（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 InitDeviceCNT GetDeviceCNT SetCNTMode GetLatchMode ReleaseDevice

♦ **取得指定通道锁存模式**

函数原型：

Visual C++ & C++Builder:

```
LONG GetLatchMode (HANDLE hDevice,
                   int nCNTChannel = 0)
```

Visual Basic:

```
Declare Function GetLatchMode Lib "PCI2394" (ByVal hDevice as Long, _
                                             ByVal nCNTChannel As Integer = 0) As Long
```

Delphi:

```
Function GetLatchMode (hDevice : Integer;
                       nCNTChannel: Integer = 0) : LongInt;
                       StdCall; External 'PCI2394' Name ' GetLatchMode ';
```

LabVIEW:

请参考相关演示程序。

功能：取得指定通道的锁存模式。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

nCNTChannel 计数器通道号（0-3）。

返回值：返回指定通道的锁存模式。

相关函数： CreateDevice InitDeviceCNT GetDeviceCNT SetCNTMode SetLathcMode ReleaseDevice

- **设置指定通道是否上溢锁定**

函数原型：

Visual C++ & C++Builder:

```
BOOL EnableOverflowLock(HANDLE hDevice,
                        BOOL bOverflowLock,
                        int nCNTChannel = 0);
```

Visual Basic:

```
Declare Function EnableOverflowLock Lib "PCI2394" (ByVal hDevice as Long, _
                                                   ByVal bOverflowLock As Boolean,
                                                   ByVal nCNTChannel As Integer = 0) As Boolean
```

Delphi:

```
Function EnableOverflowLock (hDevice : Integer;
                             bOverflowLock: Boolean;
                             nCNTChannel: Integer = 0) : Boolean;
                             StdCall; External 'PCI2394' Name ' EnableOverflowLock ';
```

LabVIEW:

请参考相关演示程序。

功能：设置指定通道的上溢是否锁定。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

bOverflowLock 是否锁定。

nCNTChannel 计数器通道号（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 ReleaseDevice

- **取得指定通道是否上溢锁定**

函数原型：

Visual C++ & C++Builder:

```
BOOL IsOverflowLock (HANDLE hDevice,
                     int nCNTChannel = 0);
```

Visual Basic:

```
Declare Function IsOverflowLock Lib "PCI2394" (ByVal hDevice as Long, _
                                               ByVal nCNTChannel As Integer = 0) As Boolean
```

Delphi:

```
Function IsOverflowLock (hDevice : Integer;
                         nCNTChannel: Integer = 0) : Boolean;
                         StdCall; External 'PCI2394' Name ' IsOverflowLock ';
```

LabVIEW:

请参考相关演示程序。

功能：得到指定通道是否上溢锁定(停止计数)。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

nCNTChannel 计数器通道号（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 EnableOverflowLock ReleaseDevice

♦ **设置指定通道是否下溢锁定**

函数原型：

Visual C++ & C++Builder：

```
BOOL EnableUnderflowLock (HANDLE hDevice,
                          BOOL bUnOverflowLock,
                          int nCNTChannel = 0);
```

Visual Basic:

```
Declare Function EnableUnderflowLock Lib "PCI2394" (ByVal hDevice as Long, _
                                                    ByVal bUnOverflowLock As Boolean,
                                                    ByVal nCNTChannel As Integer = 0) As Boolean
```

Delphi:

```
Function EnableUnderflowLock (hDevice : Integer;
                          bUnOverflowLock: Boolean;
                          nCNTChannel: Integer = 0) : Boolean;
                          StdCall; External 'PCI2394' Name ' EnableUnderflowLock ';
```

LabVIEW:

请参考相关演示程序。

功能：设置指定通道的下溢是否锁定。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

bUnOverflowLock 是否锁定。

nCNTChannel 计数器通道号（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 ReleaseDevice

♦ **取得指定通道是否下溢锁定**

函数原型：

Visual C++ & C++Builder：

```
BOOL IsUnderflowLock (HANDLE hDevice,
                      int nCNTChannel = 0);
```

Visual Basic:

```
Declare Function IsUnderflowLock Lib "PCI2394" (ByVal hDevice as Long, _
                                                ByVal nCNTChannel As Integer = 0) As Boolean
```

Delphi:

```
Function IsUnderflowLock (hDevice : Integer;
                          nCNTChannel: Integer = 0) : Boolean;
                          StdCall; External 'PCI2394' Name ' IsUnderflowLock';
```

LabVIEW:

请参考相关演示程序。

功能：取得指定通道的下溢是否锁定。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
nCNTChannel 计数器通道号（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 EnableUnderflowLock ReleaseDevice

♦ **设置指定通道是否数字滤波**

函数原型：
Visual C++ & C++Builder:

```
BOOL EnableDigitFilter (HANDLE hDevice,
                        BOOL bDigitFilter,
                        int nCNTChannel = 0);
```

Visual Basic:

```
Declare Function EnableDigitFilter Lib "PCI2394" (ByVal hDevice as Long, _
                                                  ByVal bDigitFilter As Boolean,
                                                  ByVal nCNTChannel As Integer = 0) As Boolean
```

Delphi:

```
Function EnableDigitFilter (hDevice : Integer;
                            bDigitFilter : Boolean;
                            nCNTChannel: Integer = 0) : Boolean;
                            StdCall; External 'PCI2394' Name ' EnableDigitFilter ';
```

LabVIEW:
请参考相关演示程序。

功能：设置指定通道的下溢是否锁定。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
bDigitFilter 是否数字滤波。
nCNTChannel 计数器通道号（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 ReleaseDevice

♦ **取得指定通道是否数字滤波**

函数原型：
Visual C++ & C++Builder：

```
BOOL IsDigitFilter (HANDLE hDevice,
                    int nCNTChannel = 0);
```

Visual Basic:

```
Declare Function IsDigitFilter Lib "PCI2394" (ByVal hDevice as Long, _
                                              ByVal nCNTChannel As Integer = 0) As Boolean
```

Delphi:

```
Function IsDigitFilter (hDevice : Integer;
                        nCNTChannel: Integer = 0) : Boolean;
                        StdCall; External 'PCI2394' Name ' IsDigitFilter;
```

LabVIEW:
请参考相关演示程序。

功能：得到指定通道是否数字滤波。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
nCNTChannel 计数器通道号（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 EnableDigitFilter ReleaseDevice

♦ **读取指定通道计数器的值**

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDeviceCNT (HANDLE hDevice,
                   PULONG pCNTValue,
                   int nCNTChannel = 0 )
```

Visual Basic:

```
Declare Function GetDeviceCNT Lib "PCI2394" (ByVal hDevice as Long, _
                                             ByRef pCNTValue As Long,_
                                             ByVal nCNTChannel As Integer = 0) As Boolean
```

Delphi:

```
Function GetDeviceCNT (hDevice : Integer;
                       pCNTValue : Pointer;
                       nCNTChannel: Integer = 0) : Boolean;
                       StdCall; External 'PCI2394' Name ' GetDeviceCNT ';
```

LabVIEW:

请参考相关演示程序。

功能：它必须在调用InitDeviceCNT后才能调用此函数。该函数读取指定通道的计数器的值。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pCNTValue 计数器的当前值。

nCNTChannel 计数器通道号（0-3）。

返回值：如果调用成功，则返回 TRUE，且 pCNTValue 保存计数器的值，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 ReleaseDevice

♦ **设置指定通道复位模式**

函数原型：

Visual C++ & C++Builder:

```
BOOL SetResetMode (HANDLE hDevice,
                   LONG lResetMode,
                   int nCNTChannel = 0);
```

Visual Basic:

```
Declare Function SetResetMode Lib "PCI2394" (ByVal hDevice as Long, _
                                             ByVal lResetMode As Long,_
                                             ByVal nCNTChannel As Integer = 0) As Boolean
```

Delphi:

```
Function SetResetMode (hDevice : Integer;
                       lResetMode : LongInt;
                       nCNTChannel: Integer = 0) : Boolean;
                       StdCall; External 'PCI2394' Name ' SetResetMode ';
```

LabVIEW:

请参考相关演示程序。

功能：设置指定通道的复位模式。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

lResetMode 计数器复位模式。

常量名	常量值	功能定义
PCI2394_RESETMODE_MIDDLE	0x00	计数器复位到 0x80000000
PCI2394_RESETMODE_ZERO	0x01	计数器复位到 0x00000000

nCNTChannel 计数器通道号（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 GetResetMode ReleaseDevice

♦ **读取指定通道复位模式**

函数原型：

Visual C++ & C++Builder:

```
LONG GetResetMode (HANDLE  hDevice,
                   int nCNTChannel = 0 )
```

Visual Basic:

```
Declare Function GetResetMode Lib "PCI2394" (ByVal hDevice as Long, _
                                             ByVal nCNTChannel As Integer = 0) As Long
```

Delphi:

```
Function GetResetMode (hDevice : Integer;
                       nCNTChannel: Integer = 0) : LongInt;
                       StdCall; External 'PCI2394' Name ' GetResetMode ';
```

LabVIEW:

请参考相关演示程序。

功能：读取指定通道复位模式。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

nCNTChannel 计数器通道号（0-3）。

返回值：返回指定通道的复位模式。

相关函数： CreateDevice SetResetMode ReleaseDevice

♦ **复位指定通道计数器值**

函数原型：

Visual C++ & C++Builder:

```
BOOL ResetDeviceCNT(HANDLE hDevice,
                    int nCNTChannel = 0)
```

Visual Basic:

```
Declare Function ResetDeviceCNT Lib "PCI2394" (ByVal hDevice as Long, _
                                               ByVal nCNTChannel As Integer = 0) As Boolean
```

Delphi:

```
Function ResetDeviceCNT (hDevice : Integer;
                         nCNTChannel: Integer = 0) : Boolean;
                         StdCall; External 'PCI2394' Name ' ResetDeviceCNT ';
```

LabVIEW:

请参考相关演示程序。

功能：复位指定通道计数器值，复位到由SetResetMode函数指定的复位值。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

nCNTChannel 计数器通道号（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 SetResetMode ReleaseDevice

♦ **设置指定通道比较器值**

函数原型：

Visual C++ & C++Builder:

```
BOOL SetDevCompValue(HANDLE hDevice,
                     ULONG CNTCompValue,
                     int nCNTChannel = 0)
```

Visual Basic:

```
Declare Function SetDevCompValue Lib "PCI2394" (ByVal hDevice as Long, _
                                                ByVal CNTCompValue As Long,_
```

```
                                        ByVal nCNTChannel As Integer = 0) As Boolean
```

Delphi:

```
Function SetDevCompValue (hDevice : Integer;
                          CNTCompValue: LongWord;
                          nCNTChannel: Integer = 0) : Boolean;
                          StdCall; External 'PCI2394' Name ' SetDevCompValue ';
```

LabVIEW:

请参考相关演示程序。

功能：设置指定通道比较器值。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

CNTCompValue 计数器比较值。

nCNTChannel 计数器通道号（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 GetDevCompValue ReleaseDevice

♦ **取得指定通道比较器值**

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDevCompValue (HANDLE hDevice,
                      PULONG pCNTCompValue,
                      int nCNTChannel = 0)
```

Visual Basic:

```
Declare Function GetDevCompValue Lib "PCI2394" (ByVal hDevice as Long, _
                                                ByRef pCNTCompValue As Long,_
                                                ByVal nCNTChannel As Integer = 0) As Boolean
```

Delphi:

```
Function GetDevCompValue (hDevice : Integer;
                          pCNTCompValue: Pointer;
                          nCNTChannel: Integer = 0) : Boolean;
                          StdCall; External 'PCI2394' Name ' GetDevCompValue';
```

LabVIEW:

请参考相关演示程序。

功能：取得指定通道比较器值。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pCNTCompValue 计数器比较值。

nCNTChannel 计数器通道号（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 SetDevCompValue ReleaseDevice

♦ **设置定时器设置**

函数原型：

Visual C++ & C++Builder:

```
BOOL SetTimer(HANDLE hDevice,
              LONG lTimerDivider,
              LONG lTimerDiviBase)
```

Visual Basic:

```
Declare Function SetTimer Lib "PCI2394" (ByVal hDevice as Long, _
                                         ByVal lTimerDivider As Long,_
                                         ByVal lTimerDiviBase As Long ) As Boolean
```

Delphi:

```
Function SetTimer (hDevice : Integer;
```

```
                    lTimerDivider : LongInt;
                    lTimerDiviBase : LongInt ) : Boolean;
                    StdCall; External 'PCI2394' Name ' SetTimer ';
```

LabVIEW:

请参考相关演示程序。

功能：设置定时器定时间隔。

定时时间间隔 = 1/（分频频率值/ lTimerDivider）

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

lTimerDivider 时钟分频数(1-255)(DV)。

lTimerDiviBase 分频时钟基准源选择(TB)，其值为时钟源频率选择表。

常量名	常量值	功能定义
PCI2394_TIMER_BASE_50KHz	0x00	50KHz 时钟基源
PCI2394_TIMER_BASE_5KHz	0x01	5KHz 时钟基源
PCI2394_TIMER_BASE_500Hz	0x02	500Hz 时钟基源
PCI2394_TIMER_BASE_50Hz	0x03	50Hz 时钟基源
PCI2394_TIMER_BASE_5Hz	0x04	5Hz 时钟基源

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 GetTimer ReleaseDevice

- **取得定时器设置**

函数原型：

Visual C++ & C++Builder:

```
ULONG GetTimer (HANDLE hDevice,
                    PLONG pTimerDivider,
                    PLONG pTimerBaseSel)
```

Visual Basic:

```
Declare Function GetTimer Lib "PCI2394" (ByVal hDevice as Long, _
                                         ByRef pTimerDivider As Long,_
                                         ByRef pTimerBaseSelAs Long ) As Long
```

Delphi:

```
Function GetTimer (hDevice : Integer;
                    lTimerDivider : Pointer;
                    pTimerBaseSel : Pointer) : LongWord;
                    StdCall; External 'PCI2394' Name ' GetTimer ';
```

LabVIEW:

请参考相关演示程序。

功能：读取定时器，返回单位为 ms，取值范围为20 ms ~ 51 s。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pTimerDivider 时钟分频数(1-255)(DV)。

pTimerBaseSel 分频时钟基准源选择量的指针，其值为时钟源频率选择表。

返回值：如果设置成功，返回定时器的时间间隔(ns)，并且如果 pTimerDivider != NULL 则返回时钟分频数，pTimerBaseSel != NULL 则返回时钟基准索引，如果失败返回 0。

相关函数： CreateDevice SetTimer ReleaseDevice

第四节、中断控制函数

- **设置计数器中断**

函数原型：

Visual C++ & C++Builder:

```
BOOL InitDeviceInt(HANDLE hDevice,
```

```
                    HANDLE hEventInt,
                    PPCI2394_PARA_INT pIntPara)
```

Visual Basic:

```
Declare Function InitDeviceInt Lib "PCI2394" (ByVal hDevice as Long, _
                                              ByVal hEventInt As Long,_
                                              ByRef pIntPara As PCI2394_PARA_INT) As Boolean
```

Delphi:

```
Function InitDeviceInt (hDevice : Integer;
                        hEventInt: Integer;
                        pIntPara : P PCI2394_PARA_INT) : Boolean;
                        StdCall; External 'PCI2394' Name ' InitDeviceInt ';
```

LabVIEW:

请参考相关演示程序。

功能：设置中断。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

hEventInt 中断事件句柄，用来捕获中断事件发生。

pIntPara 中断参数变量得指针，关于结构体的各个成员的意义请参见PPCI2394_PARA_INT。

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 ReleaseDeviceInt ReleaseDevice

♦ 修改中断配置参数

函数原型：

Visual C++ & C++Builder:

```
BOOL ModifyDeviceInt( HANDLE hDevice,
                      PPCI2394_PARA_INT pIntPara)
```

Visual Basic:

```
Declare Function ModifyDeviceInt Lib "PCI2394" (ByVal hDevice as Long, _
                                                ByRef pIntPara As PCI2394_PARA_INT) As Boolean
```

Delphi:

```
Function ModifyDeviceInt (hDevice : Integer;
                          pIntPara : P PCI2394_PARA_INT) : Boolean;
                          StdCall; External 'PCI2394' Name ' ModifyDeviceInt ';
```

LabVIEW:

请参考相关演示程序。

功能：修改中断配置参数，必修在初始化中断成功后才可以调用这个函数。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pIntPara 中断参数变量得指针，关于结构体的各个成员的意义请参见PPCI2394_PARA_INT。

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 InitDeviceInt ReleaseDeviceInt
ReleaseDevice

♦ 得到中断源

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDeviceIntSrc( HANDLE hDevice,
                      PPCI2394_PARA_INT pIntPara)
```

Visual Basic:

```
Declare Function GetDeviceIntSrc Lib "PCI2394" (ByVal hDevice as Long, _
                                                ByRef pIntPara As PCI2394_PARA_INT) As Boolean
```

Delphi:

```
Function GetDeviceIntSrc (hDevice : Integer;
```

pIntPara : P PCI2394_PARA_INT) : Boolean;
StdCall; External 'PCI2394' Name ' GetDeviceIntSrc ';

LabVIEW:

请参考相关演示程序。

功能：得到中断源。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pIntPara 中断参数变量得指针，关于结构体的各个成员的意义请参见PPCI2394_PARA_INT。

返回值：如果调用成功，则返回 TRUE，中断状态存放在 pIntPara 指向的变量，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 ReleaseDevice

- 取得中断次数

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDeviceIntCount(HANDLE hDevice,
                       PULONG pIntSrc)
```

Visual Basic:

```
Declare Function GetDeviceIntCount Lib "PCI2394" (ByVal hDevice as Long, _
                                                  ByRef pIntSrc As Long) As Boolean
```

Delphi:

```
Function GetDeviceIntCount (hDevice : Integer;
                            pIntSrc : Pointer) : Boolean;
                            StdCall; External 'PCI2394' Name ' GetDeviceIntCount ';
```

LabVIEW:

请参考相关演示程序。

功能：取得中断的次数。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pIntSrc 中断个数的存放变量的地址。

返回值：如果执行成功，返回TRUE，中断个数保存在pIntSrc指向的变量，如果执行失败，返回FALSE。

相关函数： CreateDevice InitDeviceInt ReleaseDeviceInt
ReleaseDevice

- 断开中断

函数原型：

Visual C++ & C++Builder:

```
BOOL ResetDeviceIntSrc (HANDLE hDevice)
```

Visual Basic:

```
Declare Function ResetDeviceIntSrc Lib "PCI2394" (ByVal hDevice as Long) As Boolean
```

Delphi:

```
Function ResetDeviceIntSrc (hDevice : Integer) : Boolean;
                            StdCall; External 'PCI2394' Name ' ResetDeviceIntSrc ';
```

LabVIEW:

请参考相关演示程序。

功能：断开中断。

参数：hDevice设备对象句柄，它应由CreateDevice创建。

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 InitDeviceInt 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 "PCI2394" (ByVal hDevice as Long) As Boolean
```

Delphi:

```
Function ReleaseDeviceInt (hDevice : Integer) : Boolean;
                    StdCall; External 'PCI2394' 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：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 InitDeviceInt ReleaseDevice

第五节、保存硬件和中断参数函数

♦ 保存硬件配置参数

函数原型：

Visual C++ & C++Builder:

```
BOOL SaveDevicePara(HANDLE hDevice,
                    PCI2394_PARA_CNT CNTPara[MAX_CHANNEL_COUNT],
                    PCI2394_PARA_MODE_DO DOPara[MAX_CHANNEL_COUNT],
                    int nChannelCount )
```

Visual Basic:

```
Declare Function SaveDevicePara Lib "PCI2394" (ByVal hDevice as Long, _
                    ByRef CNTPara(0 to MAX_CHANNEL_COUNT) As PCI2394_PARA_CNT,_
                    ByRef DOPara(0 to MAX_CHANNEL_COUNT) As PCI2394_PARA_MODE_DO,_
                    ByVal nChannelCount As Integer) As Boolean
```

Delphi:

```
Function SaveDevicePara (hDevice : Integer;
                    CNTPara : PCI2394_PARA_CNT;
                    DOPara : PCI2394_PARA_MODE_DO;
                    nChannelCount: Integer) : Boolean;
                    StdCall; External 'PCI2394' Name ' SaveDevicePara ';
```

LabVIEW:

请参考相关演示程序。

功能：保存PCI2394计数器所有通道参数到注册表中。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

CNTPara 计数器配置参数结构体数组。

DOPara DO配置参数结构体数组。

nChannelCount 保存参数通道的个数。

返回值：如果调用成功，则返回 TRUE，并保存各个通道的计时器配置参数和 DO 配置参数保存到注册表中，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 LoadDevicePara ReleaseDevice

♦ 载入硬件配置参数

函数原型：

Visual C++ & C++Builder:

```
BOOL LoadDevicePara(HANDLE hDevice,
                    PCI2394_PARA_CNT CNTPara[MAX_CHANNEL_COUNT],
                    PCI2394_PARA_MODE_DO DOPara[MAX_CHANNEL_COUNT],
                    int nChannelCount)
```

Visual Basic:

```
Declare Function LoadDevicePara Lib "PCI2394" (ByVal hDevice as Long, _
                    ByRef CNTPara(0 to MAX_CHANNEL_COUNT) As PCI2394_PARA_CNT,_
                    ByRef DOPara(0 to MAX_CHANNEL_COUNT) As PCI2394_PARA_MODE_DO,_
                    ByVal nChannelCount As Integer) As Boolean
```

Delphi:

```
Function LoadDevicePara (hDevice : Integer;
                    CNTPara : PCI2394_PARA_CNT;
                    DOPara : PCI2394_PARA_MODE_DO;
                    nChannelCount: Integer) : Boolean;
                    StdCall; External 'PCI2394' Name ' LoadDevicePara';
```

LabVIEW:

请参考相关演示程序。

功能：取得PCI2394计数器所有通道参数。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

CNTPara 计数器配置参数结构体数组。

DOPara DO配置参数结构体数组。

nChannelCount 保存参数通道的个数。

返回值：如果调用成功，则返回 TRUE，把各个通道的配置参数和 DO 配置参数保存到相应的结构体变量中，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 SaveDevicePara ReleaseDevice

♦ 保存中断配置参数

函数原型：

Visual C++ & C++Builder:

```
BOOL SaveDeviceINTPara (HANDLE hDevice,
                    PPCI2394_PARA_INT pINTPara)
```

Visual Basic:

```
Declare Function SaveDeviceINTPara Lib "PCI2394" (ByVal hDevice as Long, _
                    ByRef pINTPara As PCI2394_PARA_INT) As Boolean
```

Delphi:

```
Function SaveDeviceINTPara (hDevice : Integer;
                    pINTPara: PPCI2394_PARA_INT) : Boolean;
                    StdCall; External 'PCI2394' Name ' SaveDeviceINTPara ';
```

LabVIEW:

请参考相关演示程序。

功能：保存中断设置。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pINTPara 保存计数器参数结构体。

返回值：如果执行成功 返回TRUE，并把中断配置保存到注册表中， 如果执行失败返回FALSE。

相关函数： CreateDevice LoadDeviceINTPara ReleaseDevice

♦ 载入中断配置参数

函数原型：

Visual C++ & C++Builder:

```
BOOL LoadDeviceINTPara(HANDLE hDevice,
                    PPCI2394_PARA_INT pINTPara)
```

Visual Basic:

```
Declare Function LoadDeviceINTPara Lib "PCI2394" (ByVal hDevice as Long, _
                    ByRef pINTPara As PCI2394_PARA_INT) As Boolean
```

Delphi:

```
Function LoadDeviceINTPara (hDevice : Integer;
                    pINTPara: PPCI2394_PARA_INT) : Boolean;
```

StdCall; External 'PCI2394' Name ' LoadDeviceINTPara ';

LabVIEW:

请参考相关演示程序。

功能：载入中断配置参数。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pINTPara 保存计数器参数结构体。

返回值：如果执行成功 返回TRUE，并把中断配置保存在执针指向的变量中，如果执行失败返回FALSE。

相关函数： CreateDevice SaveDeviceINTPara ReleaseDevice

第六节、DIO 数字量输入输出开关量操作函数原型说明

♦ 开关量输入

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDeviceDI ( HANDLE hDevice,
                   BYTE byDISts[MAX_CHANNEL_COUNT])
```

Visual Basic:

```
Declare Function GetDeviceDI Lib "PCI2394" ( ByVal hDevice As Long, _
                                             ByVal byDISts (0 to MAX_CHANNEL_COUNT) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI ( hDevice : Integer;
                       byDISts : Pointer) : Boolean;
                       StdCall; External 'PCI2394' Name ' GetDeviceDI ';
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的输入开关量状态读入到 bDISts[x]数组参数中。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

byDISts 四路开关量输入状态的参数结构，共有 4 个元素，分别对应于 DI0～DI3 路开关量输入状态位。如果 byDISts [0]等于“1”则表示 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 byDISts[x]中的值有效；否则返回 FALSE，其 byDISts[x]中的值无效。

相关函数： CreateDevice ReleaseDevice

♦ 设置 DO 模式

函数原型：

Visual C++ & C++Builder:

```
BOOL SetDevDOMode( HANDLE hDevice,
                   PCI2394_PARA_MODE_DO DOMode[MAX_CHANNEL_COUNT])
```

Visual Basic:

```
Declare Function SetDevDOMode Lib "PCI2394" ( ByVal hDevice As Long, _
                ByRef DOMode (0 to MAX_CHANNEL_COUNT) As PCI2394_PARA_MODE_DO) As Boolean
```

Delphi:

```
Function SetDevDOMode (hDevice : Integer;
                       DOMode : Pointer) : Boolean;
                       StdCall; External 'PCI2394' Name ' SetDevDOMode ';
```

LabVIEW:

请参考相关演示程序。

功能：设置各个通道的 DO 模式。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

DOMode 通道的DO模式结构体数组。DO参数结构体定义。

返回值：若成功，返回TRUE，否则返回FALSE。

相关函数： CreateDevice GetDevDOMode ReleaseDevice

◆ 取得 DO 模式

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDevDOMode (HANDLE hDevice,
                   PCI2394_PARA_MODE_DO DOMode[MAX_CHANNEL_COUNT])
```

Visual Basic:

```
Declare Function GetDevDOMode Lib "PCI2394" ( ByVal hDevice As Long, _
        ByRef DOMode (0 to MAX_CHANNEL_COUNT) As PCI2394_PARA_MODE_DO) As Boolean
```

Delphi:

```
Function GetDevDOMode (hDevice : Integer;
                       DOMode : Pointer) : Boolean;
                       StdCall; External 'PCI2394' Name ' GetDevDOMode ';
```

LabVIEW:

请参考相关演示程序。

功能：取得各个通道的 DO 模式。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

DOMode 通道的DO模式结构体数组。DO参数结构体定义。

返回值：若成功，返回TRUE，并保存各个通道的DO模式到变量DOMode中，否则返回FALSE。

相关函数： CreateDevice SetDevDOMode ReleaseDevice

第四章 硬件参数结构

第一节、计数器硬件参数结构（PCI2394_PARA_CNT）

Visual C++ & C++Builder：

```
typedef struct _PCI2394_PARA_CNT
{
    LONG lCNTMode;          // 计数器输入控制模式(CM)
    LONG lResetMode;        // 计数器复位值(RM)
    LONG bOverflowLock;     // 是否计数器上溢锁定控制(OL) TRUE=锁定
    LONG bUnderflowLock;    // 是否计数器下溢锁定控制(UL) TRUE=锁定
    LONG bDigitFilter;      // 差分输入是否进行数字滤波(DF) TRUE=过滤
    LONG lLatchMode;        // 锁存模式设置(7 种模式)
    LONG lIndexReset;       // 索引复位
} PCI2394_PARA_CNT, *PPCI2394_PARA_CNT;
```

Visual Basic：

```
Private Type PCI2394_PARA_AD
    lCNTMode As Long          ' 计数器输入控制模式
    lResetMode As Long        ' 计数器复位值
    bOverflowLock As Long     ' 是否计数器上溢锁定控制(OL) TRUE=锁定
    bUnderflowLock As Long    ' 是否计数器下溢锁定控制(UL) TRUE=锁定
    bDigitFilter As Long      ' 差分输入是否进行数字滤波(DF) TRUE=过滤
    lLatchMode As Long        ' 锁存模式设置(7 种模式)
    lIndexReset As Long       ' 索引复位
End Type
```

Delphi:

```
Type   // 定义结构体数据类型
```

```
  PPCI2394_PARA_AD = ^ PCI2394_PARA_AD;   // 指针类型结构
  PCI2394_PARA_AD = record          // 标记为记录型
    lCNTMode : LontInt;             // 计数器输入控制模式
    lResetMode : LontInt;           // 计数器复位值
    bOverflowLock : LontInt;        // 是否计数器上溢锁定控制(OL) TRUE=锁定
    bUnderflowLock : LontInt;       // 是否计数器下溢锁定控制(UL) TRUE=锁定
    bDigitFilter : LontInt;         // 差分输入是否进行数字滤波(DF) TRUE=过滤
    lLatchMode: LontInt;            // 锁存模式设置(7 种模式)
    lIndexReset : LontInt;          // 索引复位
End;
```

LabVIEW:
请参考相关演示程序。

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

lCNTMode 计数器输入控制模式。

常量名	常量值	功能定义
PCI2394_CNTMODE_DISABLE	0x00	计数器输入控制模式无效，但可以访问所有的寄存器
PCI2394_CNTMODE_QUADRATURE_X1	0x01	计数器差分输入，Channel A 上升沿有效开始计数
PCI2394_CNTMODE_QUADRATURE_X2	0x02	计数器差分输入，Channel A 只要有跳变就开始计数
PCI2394_CNTMODE_QUADRATURE_X4	0x03	计数器差分输入，Channel A 或 Channel B 只要有跳变就开始计数
PCI2394_CNTMODE_2_PULSE	0x04	双脉冲模式,一个做顺时针计数，另一个做逆时针计数，当 Channel B 在上升沿时有效
PCI2394_CNTMODE_1_PULSE	0x05	A 线为脉冲，B 线为方向

lResetMode 计数器复位模式。

常量名	常量值	功能定义
PCI2394_RESETMODE_MIDDLE	0x00	计数器复位到 0x80000000
PCI2394_RESETMODE_ZERO	0x01	计数器复位到 0x00000000

bOverflowLock 是否计数器上溢锁定控制(OL) TRUE=锁定。
bUnderflowLock 是否计数器下溢锁定控制(UL) TRUE=锁定。
bDigitFilter 差分输入是否进行数字滤波(DF) TRUE=过滤。
lLatchMode 锁存模式设置(7 种模式)。

常量名	常量值	功能定义
PCI2394_LATCHMODE_SOFT	0x00	软件锁存计数器数据 SoftWare
PCI2394_LATCHMODE_IDX	0x01	索引锁存计数器数据 Index
PCI2394_LATCHMODE_TM	0x04	定时器锁存计数器数据 Time
PCI2394_LATCHMODE_DI0	0x10	开关量输入 0 通道锁存计数器数据
PCI2394_LATCHMODE_DI1	0x20	开关量输入 1 通道锁存计数器数据
PCI2394_LATCHMODE_DI2	0x40	开关量输入 2 通道锁存计数器数据
PCI2394_LATCHMODE_DI3	0x80	开关量输入 3 通道锁存计数器数据

lIndexReset 索引复位。

相关函数： CreateDevice LoadDevicePara SaveDevicePara
ReleaseDevice

第二节、中断参数结构（PCI2394_PARA_INT）

Visual C++ & C++Builder：

```
typedef struct _PCI2394_PARA_INT
{
    LONG bEnableInt;              // 总中断使能
    LONG bTimerInt;               // 定时器中断
    LONG bCNTOverflowInt[MAX_CHANNEL_COUNT];        // 上溢中断
    LONG bCNTUnderflowInt[MAX_CHANNEL_COUNT];       // 通道下溢中断
    LONG bCNTOverCompInt[MAX_CHANNEL_COUNT];
    LONG bCNTUnderCompInt[MAX_CHANNEL_COUNT];
    LONG bCNTIndexInt[MAX_CHANNEL_COUNT];
    LONG bDIInt[MAX_CHANNEL_COUNT];
} PCI2394_PARA_INT, *PPCI2394_PARA_INT;
```

Visual Basic：

```
Private Type    PCI2394_STATUS_AD
    bEnableInt As Long
    bTimerInt As Long
    bCNTOverflowInt[MAX_CHANNEL_COUNT] As Long
    bCNTUnderflowInt [MAX_CHANNEL_COUNT] As Long
    bCNTOverCompInt [MAX_CHANNEL_COUNT] As Long
    bCNTUnderCompInt [MAX_CHANNEL_COUNT] As Long
    bCNTIndexInt [MAX_CHANNEL_COUNT] As Long
    bDIInt [MAX_CHANNEL_COUNT] As Long
End Type
```

Delphi:

```
Type   // 定义结构体数据类型
    PPCI2394_STATUS_AD = ^ PCI2394_STATUS_AD;   // 指针类型结构
    PCI2394_STATUS_AD = record          // 标记为记录型
    bEnableInt : LongInt;
    bTimerInt : LongInt;
    bCNTOverflowInt[MAX_CHANNEL_COUNT] : Array [0…3] of LongInt;
    bCNTUnderflowInt [MAX_CHANNEL_COUNT] : Array [0…3] of LongInt;
    bCNTOverCompInt [MAX_CHANNEL_COUNT] : Array [0…3] of LongInt;
    bCNTUnderCompInt [MAX_CHANNEL_COUNT] : Array [0…3] of LongInt;
    bCNTIndexInt [MAX_CHANNEL_COUNT] : Array [0…3] of LongInt;
    bDIInt [MAX_CHANNEL_COUNT] : Array [0…3] of LongInt;
End;
```

LabVIEW:

请参考相关演示程序。

此结构体主要用于查询中断的各种状态。

bEnableInt 全部中断使能。

常量名	常量值	功能定义
PCI2394_IE_DISABLE	0x00	全部中断不允许
PCI2394_IE_ENABLE	0x01	全部中断允许

bTimerInt 定时器脉冲中断使能。

常量名	常量值	功能定义
PCI2394_TM_DISABLE	0x00	定时器脉冲中断不允许
PCI2394_TM_ENABLE	0x01	定时器脉冲中断允许

bCNTOverflowInt[] 上溢中断控制使能。

常量名	常量值	功能定义
PCI2394_OV_DISABLE	0x00	中断上溢不允许
PCI2394_OV_ENABLE	0x01	中断上溢允许

bCNTUnderflowInt[] 下溢中断控制使能。

常量名	常量值	功能定义
PCI2394_UN_DISABLE	0x00	中断下溢不允许
PCI2394_UN_ENABLE	0x01	中断下溢允许

bCNTOverCompInt[] 计数器大于比较值中断使能。

常量名	常量值	功能定义
PCI2394_OC_DISABLE	0x00	计数器大于比较值中断不允许
PCI2394_OC_ENABLE	0x01	计数器大于比较值中断允许

bCNTUnderCompInt[] 计数器小于比较值中断使能。

常量名	常量值	功能定义
PCI2394_UC_DISABLE	0x00	计数器小于比较值中断不允许
PCI2394_UC_ENABLE	0x01	计数器小于比较值中断允许

bCNTIndexInt[] 索引状态中断使能。

常量名	常量值	功能定义
PCI2394_IX_DISABLE	0x00	索引状态中断不允许
PCI2394_IX_ENABLE	0x01	索引状态中断允许

bDIInt[] 数字输入中断使能。

常量名	常量值	功能定义
PCI2394_DI_DISABLE	0x00	数字输入中断不允许
PCI2394_DI_ENABLE	0x01	数字输入中断允许

相关函数： CreateDevice　InitDeviceInt　ModifyDeviceInt　GetDeviceIntSrc　GetDeviceIntCount　ReleaseDeviceInt　ReleaseDevice

第三节、开关量参数结构设置（PCI2394_PARA_MODE_DO）

Visual C++ & C++Builder:

```
typedef struct   _PCI2394_PARA_MODE_DO
{
    BYTE bDOManualCtrl; // (DM)是否直接输出开关量 =TRUE: byDOSts 有效; =FALSE: 除 byDOSts 以外有效
    BYTE byDOSts;       // 当 bDOManualCtrl=1 有效
    BYTE bOverCompare;  // (OC)设置数字量输出通道是否置大于比较寄存器的值(0: N/A，1: 置大于)
    BYTE bUnderCompare; //(UC)设置数字量输出通道是否置小于比较寄存器的值(0: N/A，1: 置小于)
    BYTE Level;         // (LE)设置数字量输出 0 Pulse with counter clock, 1 Level with clear interrupt
} PCI2394_PARA_MODE_DO, *PPCI2394_PARA_MODE_DO;
```

Visual Basic：

```
Type PCI2394_PARA_MODE_DO
    bDOManualCtrl    As Byte  ' (DM)是否直接输出开关量 =TRUE: byDOSts 有效; =FALSE: 除 byDOSts 以外有效
    byDOSts As Byte           ' 当 bDOManualCtrl=1 有效
    bOverCompare As Byte      '(OC)设置数字量输出通道是否置大于比较寄存器的值(0: N/A，1: 置大于)
    bUnderCompare As Byte     '(UC)设置数字量输出通道是否置小于比较寄存器的值(0: N/A，1: 置小于)
```

```
    Level As Byte              '(LE)设置数字量输出  0 Pulse with counter clock, 1 Level with clear interrupt
End Type
```

Delphi:

```
Type   // 定义结构体数据类型
  P PCI2394_PARA_MODE_DO   = ^ PCI2394_PARA_MODE_DO;   // 指针类型结构
  PCI2394_PARA_MODE_DO = record          // 标记为记录型
    bDOManualCtrl    : Byte;   // (DM)是否直接输出开关量  =TRUE: byDOSts 有效; =FALSE: 除byDOSts 以外有效
    byDOSts          : Byte;   //当 bDOManualCtrl=1 有效
    bOverCompare     : Byte;   //(OC)设置数字量输出通道是否置大于比较寄存器的值(0: N/A，1: 置大于)
    bUnderCompare    : Byte;   //(UC)设置数字量输出通道是否置小于比较寄存器的值(0: N/A，1: 置小于)
    Level            : Byte;   //(UC)设置数字量输出通道是否置小于比较寄存器的值(0: N/A，1: 置小于)
End;
```

该参数结构的使用极大的方便了不熟悉硬件端口控制和二进制位操作的用户。在这里您不需要了解技术细节，只需要执行SetDevDOMode即可完成数字量输出操作。然后象Visual Basic中的属性操作那样，简单的进行属性成员分析即可确定各路状态。

bDOManualCtrl 数字量输出模式控制，设置(DM)是否直接输出开关量。=TRUE: byDOSts 有效; =FALSE: 除byDOSts以外有效。

常量名	常量值	功能定义
PCI2394_DM_NORMAL	0x00	普通输出
PCI2394_DM_INDICATED	0x01	指定输出

byDOSts 当 bDOManualCtrl=1 有效。

bOverCompare (OC)设置数字量输出通道是否置大于比较寄存器的值(0: N/A，1: 置大于)。

bUnderCompare (UC)设置数字量输出通道是否置小于比较寄存器的值(0: N/A，1: 置小于)。

Level (LE)设置数字量输出模式控制(脉冲、中断)。

常量名	常量值	功能定义
PCI2394_LE_PULSE	0x00	脉冲输出
PCI2394_LE_LEVEL	0x01	电平输出

相关函数： CreateDevice　SetDevDOMode　GetDevDOMode　ReleaseDevice

第五章 上层用户函数接口应用实例

第一节、怎样使用GetDeviceCNT函数直接取得计数器的值

Visual C++ & C++Builder:

其详细应用实例及正确代码请参考 Visual C++测试与演示系统，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。

[程序]丨[阿尔泰测控演示系统]丨[PCI2394 编码器、计数器和开关量卡]丨[Microsoft Visual C++]丨[简易代码演示]丨[CNT 简易程序]

第二节、怎样使用GetDeviceDI函数进行更便捷的数字开关量输入操作

Visual C++ & C++Builder:

其详细应用实例及正确代码请参考 Visual C++测试与演示系统，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。

[程序]丨**[阿尔泰测控演示系统]**丨**[PCI2394 编码器、计数器和开关量卡]**丨**[Microsoft Visual C++]**丨**[简易代码演示]**丨**[DIO 简易程序]**

第三节、怎样使用SetDevDOMode函数进行更便捷的数字开关量输出操作

Visual C++ & C++Builder:

其详细应用实例及正确代码请参考 Visual C++测试与演示系统，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。

[程序]丨**[阿尔泰测控演示系统]**丨**[PCI2394 编码器、计数器和开关量卡]**丨**[Microsoft Visual C++]**丨**[简易代码演示]**丨**[DIO 简易程序]**

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“PCI2394_”）

函数名	函数功能	备注
① I/O 端口操作函数		
GetDeviceAddr	取得指定 PCI 设备寄存器操作基地址	底层用户
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
WritePortULongEx	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
ReadPortULongEx	以无符号双字(32Bit)方式读 I/O 端口	用户程序操作端口
② 线程操作函数		
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	
③ 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

第二节、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。

◆ **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)
```

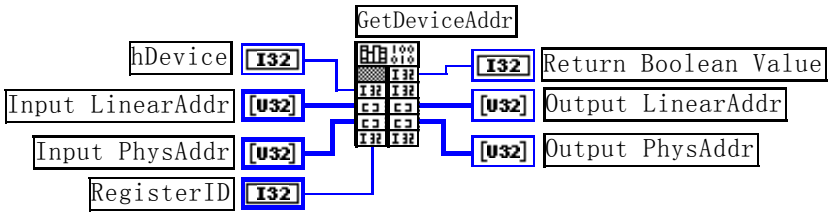
Visual Basic:

```
Declare Function GetDeviceAddr Lib "PCI2394" (ByVal hDevice As Long, _
                                              ByRef LinearAddr As Long, _
                                              ByRef PhysAddr As Long, _
                                              Optional ByVal RegisterID As Integer) As Boolean
```

Delphi:

```
Function GetDeviceAddr(hDevice : Integer;
                       LinearAddr : Pointer;
                       PhysAddr : Pointer;
                       RegisterID : Integer) : Boolean;
                       StdCall; External 'PCI2394' 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　WritePortByte　WritePortWord　WritePortULongEx　ReadPortByte　ReadPortWord　ReadPortULongEx　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
    :
```

♦ **以单字节(8Bit)方式写 I/O 端口**

Visual C++ & C++ Builder:

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

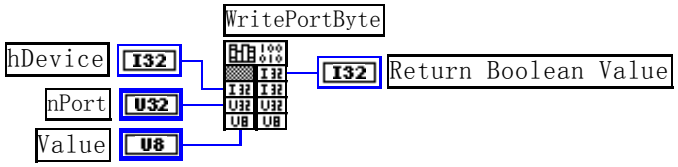
Visual Basic:

```
Declare Function WritePortByte Lib "PCI2394" ( 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 'PCI2394' 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 WritePortULongEx ReadPortByte ReadPortWord ReadPortULongEx ReleaseDevice

♦ **以双字(16Bit)方式写 I/O 端口**

Visual C++ & C++ Builder:

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

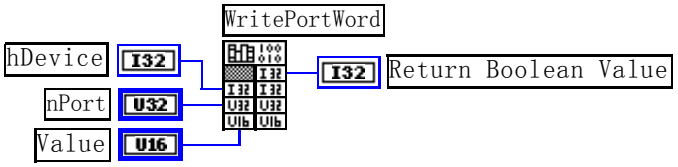
Visual Basic:

```
Declare Function WritePortWord Lib "PCI2394" (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 'PCI2394' 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 WritePortULongEx ReadPortByte ReadPortWord

ReadPortULongEx　　ReleaseDevice

- **以四字节(32Bit)方式写 I/O 端口**

Visual C++ & C++ Builder:

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

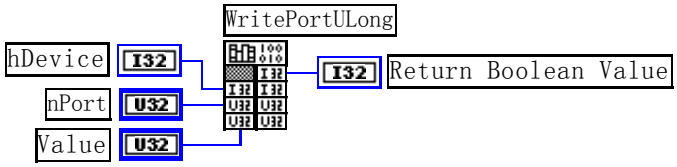
Visual Basic:

```
Declare Function WritePortULongEx Lib "PCI2394" (ByVal hDevice As Long, _
                                                 ByVal nPort As Long, _
                                                 ByVal Value As Long ) As Boolean
```

Delphi:

```
Function WritePortULongEx (hDevice : Integer;
                           nPort : LongWord;
                          Value : LongWord) : Boolean;
                           StdCall; External 'PCI2394' Name ' WritePortULongEx ';
```

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
WritePortULongEx　　ReadPortByte　　ReadPortWord
ReadPortULongEx　　ReleaseDevice

- **以单字节(8Bit)方式读 I/O 端口**

Visual C++ & C++ Builder:

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

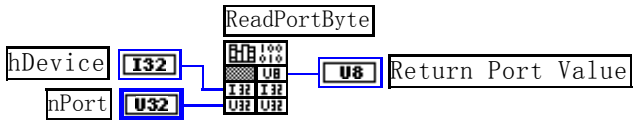
Visual Basic:

```
Declare Function ReadPortByte Lib "PCI2394" (ByVal hDevice As Long, _
                                             ByVal nPort As Long ) As Byte
```

Delphi:

```
Function ReadPortByte(hDevice : Integer;
                      nPort : LongWord) : Byte;
                      StdCall; External 'PCI2394' 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
WritePortULongEx　　ReadPortByte　　ReadPortWord
ReadPortULongEx　　ReleaseDevice

◆ **以双字节(16Bit)方式读 I/O 端口**

Visual C++ & C++ Builder:

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

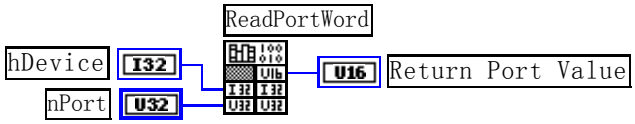
Visual Basic:

```
Declare Function ReadPortWord Lib "PCI2394" ( ByVal hDevice As Long, _
                                              ByVal nPort As Long ) As Integer
```

Delphi:

```
Function ReadPortWord(hDevice : Integer;
                      nPort : LongWord) : Word;
                      StdCall; External 'PCI2394' 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　WritePortULongEx　ReadPortByte　ReadPortWord　ReadPortULongEx　ReleaseDevice

◆ **以四字节(32Bit)方式读 I/O 端口**

Visual C++ & C++ Builder:

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

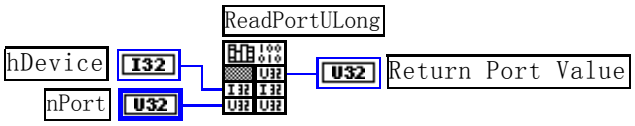
Visual Basic:

```
Declare Function ReadPortULongEx Lib "PCI2394" ( ByVal hDevice As Long, _
                                                 ByVal nPort As Long ) As Long
```

Delphi:

```
Function ReadPortULongEx (hDevice : Integer;
                          nPort : LongWord) : LongWord;
                          StdCall; External 'PCI2394' Name ' ReadPortULongEx ';
```

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　WritePortULongEx　ReadPortByte　ReadPortWord　ReadPortULongEx　ReleaseDevice

第三节、线程操作函数原型说明

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

◆ **创建内核系统事件**

函数原型：

Visual C++ & C++ Builder:

HANDLE CreateSystemEvent(void)

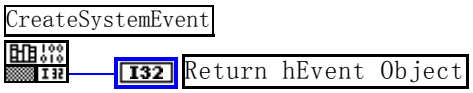
Visual Basic:

Declare Function CreateSystemEvent Lib " PCI2394 " () As Long

Delphi:

Function CreateSystemEvent() : Integer;
StdCall; External 'PCI2394' 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 " PCI2394 " (ByVal hEvent As Long) As Boolean

Delphi:

Function ReleaseSystemEvent(hEvent : Integer) : Boolean;
StdCall; External 'PCI2394' 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。

第四节、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 "PCI2394" (ByVal hDevice As Long, _
ByVal NewFileNameString, _
ByVal Mode As Integer) As Long

Delphi:

Function CreateFileObject (hDevice : Integer;
NewFileName: string;
Mode : Integer) : Integer;
Stdcall; external 'PCI2394' Name ' CreateFileObject ';

LabVIEW:

请参见相关演示程序。

功能：初始化设备文件对象， 以期待 WriteFile 请求准备文件对象进行文件操作。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

NewFileName 与新文件对象关联的磁盘文件名，可以包括盘符和路径等信息。在 C 语言中，其语法格式如：“C:\\PCI2394\\Data.Dat”，在 Basic 中，其语法格式如：“C:\PCI2394\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
PCI2394_modeRead	0x0000	只读文件方式
PCI2394_modeWrite	0x0001	只写文件方式
PCI2394_modeReadWrite	0x0002	既读又写文件方式
PCI2394_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
PCI2394_typeText	0x4000	以文本方式操作文件

返回值：若成功，则返回文件对象句柄。

相关函数： CreateDevice CreateFileObject WriteFile ReadFile ReleaseFile ReleaseDevice

♦ **通过设备对象，往指定磁盘上写入用户空间的采样数据**

函数原型：

Visual C++ & C++ Builder:

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

Visual Basic:

```
Declare Function WriteFile Lib "PCI2394" ( ByVal hFileObject As Long,_
                                           ByRef pDataBuffer As Byte,_
                                           ByVal nWriteSizeBytes As Long) As Boolean
```

Delphi:

```
Function WriteFile(hFileObject: Integer;
                   pDataBuffer : Pointer;
                   nWriteSizeBytes : LongWord) : Boolean;
                   Stdcall; external 'PCI2394' Name ' WriteFile ';
```

LabVIEW:

详见相关演示程序。

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

参数：

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

pDataBuffer 用户数据空间地址，可以是用户分配的数组空间。

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

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

相关函数： CreateFileObject WriteFile ReadFile ReleaseFile

♦ **通过设备对象,从指定磁盘文件中读采样数据**

函数原型：

Visual C++ & C++ Builder:

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

Visual Basic:

```
Declare Function ReadFile Lib "PCI2394" (  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 : LongWord;
```

nReadSizeBytes : LongWord) : Boolean;
Stdcall; external 'PCI2394' 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,
                    ULONG nOffsetBytes)
```
Visual Basic:
```
Declare Function SetFileOffset Lib "PCI2394" ( ByVal hFileObject As Long,
                                                ByVal nOffsetBytes As Long) As Boolean
```
Delphi:
```
Function SetFileOffset ( hFileObject : Integer;
                         nOffsetBytes : LongWord) : Boolean;
                         Stdcall; external 'PCI2394' 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 "PCI2394" (ByVal hFileObject As Long) As Long
```
Delphi:
```
Function GetFileLength (hFileObject : Integer) : LongWord;
                        Stdcall; external 'PCI2394' 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

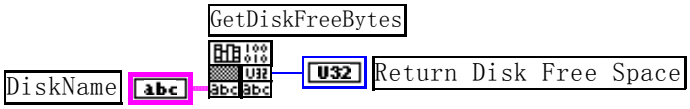
- **释放设备文件对象**

函数原型：
Visual C++ & C++ Builder:

BOOL ReleaseFile(HANDLE hFileObject)
Visual Basic:
Declare Function ReleaseFile Lib "PCI2394" (ByVal hFileObject As Long) As Boolean
Delphi:
Function ReleaseFile(hFileObject : Integer) : Boolean;
Stdcall; external 'PCI2394' Name ' ReleaseFile ';
LabVIEW:
详见相关演示程序。

功能：释放设备文件对象。
参数：hFileObject 设备对象句柄，它应由CreateFileObject创建。
返回值：若成功，则返回TRUE，否则返回FALSE，用户可以用GetLastErrorEx捕获错误码。
相关函数： CreateFileObject WriteFile ReadFile
ReleaseFile

♦ **取得指定磁盘的可用空间**
Visual C++ & C++ Builder:
ULONGLONG GetDiskFreeBytes(LPCTSTR DiskName)
Visual Basic:
Declare Function GetDiskFreeBytes Lib "PCI2394" (ByVal DiskName As String) As Currency
Delphi:
Function GetDiskFreeBytes (DiskName: String) : Currency;
Stdcall; external 'PCI2394' Name ' GetDiskFreeBytes ';
LabVIEW:



功能：取得指定磁盘的可用剩余空间(以字为单位)。
参数：DiskName 需要访问的盘符，若为 C 盘为"C:\\"， D 盘为"D:\\"，以此类推。
返回值：若成功，返回大于或等于 0 的长整型值，否则返回零值，用户可用GetLastErrorEx捕获错误码。注意使用 64 位整型变量。