CPCI4224 通讯卡

WIN2000/XP 驱动程序使用说明书

北京阿尔泰科技发展有限公司
产品研发部修订

请您务必阅读《使用纲要》，他会使您事半功倍!

目　录

提醒用户：

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

有关设备驱动安装和产品二次发行请参考 CPCI4224Inst.doc 文档。

第一章 版权信息与命名约定

第一节、版权信息

第二节、命名约定

一、为简化文字内容，突出重点，本文中提到的函数名通常为基本功能名部分，其前缀设备名如 PCIxxxx_ 则被省略。如 CPCI4224_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。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、如何管理设备

由于我们的驱动程序采用面向对象编程，所以要使用设备的一切功能，则必须首先用CreateDevice函数创建一个设备对象句柄hDevice，有了这个句柄，您就拥有了对该设备的控制权。然后将此句柄作为参数传递给

其他函数，如SetDeviceDO函数可用实现开关量的输出等。最后可以通过ReleaseDevice将hDevice释放掉。

第三节、如何实现开关量的简便操作

当您有了hDevice设备对象句柄后，便可用SetDeviceDO函数实现开关量的输出操作，其各路开关量的输出状态由其bDOISts[16]中的相应元素决定。

第四节、哪些函数对您不是必须的

公共函数如CreateFileObject， WriteFile，ReadFile等一般来说都是辅助性函数，除非您要使用存盘功能。如果您使用上层用户函数访问设备，那么GetDeviceAddr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　设备专用函数接口介绍

第一节、设备驱动接口函数列表（每个函数省略了前缀“CPCI4224_”）

函数名	函数功能	备注
① 设备对象操作函数		
CreateDevice	创建对象(用设备逻辑号)	
GetDeviceCount	取得同一种设备的总台数	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	列表所有同一种设备的各种配置	上层及底层用户
ReleaseDevice	关闭设备，且释放总线设备对象	
② DIO 开关量函数		
EnableStsDIO	设置开关量输入或输出状态	上层用户
GetDeviceDI_PA	DIO0～DIO7 开关输入函数	上层用户
SetDeviceDO_PA	DIO0～DIO7 开关输出函数	上层用户
GetDeviceDI_PB	DIO8～DIO15 开关输入函数	上层用户
SetDeviceDO_PB	DIO8～DIO15 开关输出函数	上层用户
GetDeviceDI_PC	DIO16～DIO23 开关输入函数	上层用户
SetDeviceDO_PC	DIO16～DIO23 开关输出函数	上层用户
③ UART 通信操作函数		
SetUARTMode	通信口工作模式选择	上层用户
InitUARTPara	初始化 UART 参数	上层用户
GetUARTSendDataSts	获取发送数据过程中设备的各种状态	上层用户
GetUARTReceiveDataCount	获取接收到的数据个数	上层用户
UARTWriteData	发送数据	上层用户
UARTReadData	接收数据	上层用户
④ CAN 操作函数		
EnableCANChannle	使能 CAN 通道	上层用户
InitCAN	初始化 CAN 卡	上层用户

InitCANEx	初始化 CAN 卡	上层用户
SendFrameSingle	发送单帧操作	上层用户
SendFrameMultiple	发送多帧操作	上层用户
SendFrameMultipleEx	发送多帧操作，每帧内容可以不同	上层用户
ReceiveFrame	接收帧操作	上层用户

使用需知

Visual C++ & C++Builder：

首先将 CPCI4224.h 和 CPCI4224.lib 两个驱动库文件从相应的演示程序文件夹下复制到您的源程序文件夹中，然后在您的源程序头部添加如下语句，以便将驱动库函数接口的原型定义信息和驱动接口导入库(CPCI4224.lib)加入到您的工程中。

#include “CPCI4224.H”

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

关于 CPCI4224.h 和 CPCI4224.lib 两个文件均可在演示程序文件夹下面找到。

Visual Basic:

首先将 CPCI4224.Bas 驱动模块头文件从 VB 的演示程序文件夹下复制到您的源程序文件夹中，然后将此模块文件加入到您的 VB 工程中。其方法是选择 VB 编程环境中的工程(Project)菜单，执行其中的"添加模块"(Add Module)命令，在弹出的对话中选择 CPCI4224.Bas 模块文件即可，一旦完成以上工作后，那么使用设备的驱动程序接口就跟使用 VB 自身的各种函数，其方法一样简单，毫无二别。

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

Delphi:

首先将 CPCI4224.Pas 驱动模块头文件从 Delphi 的演示程序文件夹下复制到您的源程序文件夹中，然后将此模块文件加入到您的 Delphi 工程中。其方法是选择 Delphi 编程环境中的 View 菜单，执行其中的"Project Manager"命令，在弹出的对话中选择*.exe 项目，再单击鼠标右键，最后 Add 指令，即可将 CPCI4224.Pas 单元模块文件加入到工程中。或者在 Delphi 的编程环境中的 Project 菜单中，执行 Add To Project 命令，然后选择*.Pas 文件类型也能实现单元模块文件的添加。最后请在使用驱动程序接口的源程序文件中的头部的 Uses 关键字后面的项目中加入：“CPCI4224”。如：

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  CPCI4224;  // 注意：  在此加入驱动程序接口单元 CPCI4224
```

LabView / CVI:

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

第二节、设备对象管理函数原型说明

♦ 创建设备对象函数

函数原型：

Visual C++ & C++Builder：

HANDLE CreateDevice (int DeviceLgcID= 0)

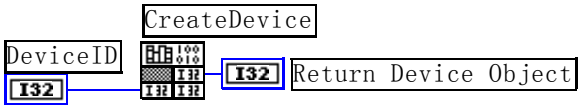
Visual Basic：

Declare Function CreateDevice Lib "CPCI4224" (Optional ByVal DeviceLgcIDAs Integer = 0) As Long

Delphi：

Function CreateDevice(DeviceLgcID: Integer = 0) : Integer;
StdCall; External 'CPCI4224' 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 设备 ID(Identifier)标识号。当向同一个 Windows 系统中加入若干相同类型的设备时，系统将以该设备的“基本名称”与 DeviceLgcID 标识值为名称后缀的标识符来确认和管理该设备。默认值为 0。

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

相关函数： ReleaseDevice

♦ 取得本计算机系统中 CPCI4224 设备的总数量

函数原型：

Visual C++ & C++Builder:

int GetDeviceCount (HANDLE hDevice)

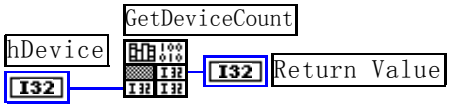
Visual Basic:

Declare Function GetDeviceCount Lib "CPCI4224" (ByVal hDevice As Long) As Integer

Delphi：

Function GetDeviceCount (hDevice : Integer) : Integer;
StdCall; External 'CPCI4224' Name ' GetDeviceCount ';

LabVIEW：



功能：取得 CPCI4224 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 CPCI4224 的数量。

相关函数： CreateDevice　GetDeviceCount　GetDeviceCurrentID
ListDeviceDlg　ReleaseDevice

♦ 取得该设备当前逻辑 ID 和物理 ID

函数原型：

Visual C++ & C++Builder:

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

Visual Basic:

```
Declare Function GetDeviceCurrentID Lib "CPCI4224" (ByVal hDevice As Long,_
                                                    ByRef DeviceLgcID As Long,_
                                                    ByRef DevicePhysIDAs Long ) As Boolean
```

Delphi：

```
Function GetDeviceCurrentID (hDevice : Integer;
                             DeviceLgcID : Pointer;
                             DevicePhysID : Pointer) : Boolean;
                             StdCall; External 'CPCI4224' Name ' GetDeviceCurrentID ';
```

LabVIEW：

请参考相关演示程序。

功能：取得指定设备逻辑和物理 ID 号。

参数：

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

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

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

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

相关函数： CreateDevice GetDeviceCount GetDeviceCurrentID ListDeviceDlg ReleaseDevice

♦ **用对话框控件列表计算机系统中所有 CPCI4224 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 "CPCI4224" (ByVal hDevice As Long ) As Boolean
```

Delphi：

```
Function ListDeviceDlg (hDevice : Integer) : Boolean;
                        StdCall; External 'CPCI4224' Name ' ListDeviceDlg ';
```

LabVIEW：

请参考相关演示程序。

功能：列表系统中 CPCI4224 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 CPCI4224 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)
```

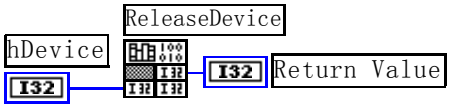
Visual Basic:

Declare Function ReleaseDevice Lib "CPCI4224" (ByVal hDevice As Long) As Boolean

Delphi：

Function ReleaseDevice(hDevice : Integer) : Boolean;
StdCall; External 'CPCI4224' Name ' ReleaseDevice ';

LabVIEW：



功能：释放设备对象所占用的系统资源及设备对象自身。

参数：hDevice设备对象句柄，它应由CreateDevice创建。

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

相关函数：CreateDevice

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

第三节、DIO 数字开关量输入输出简易操作函数原型说明

♦ **设置开关量输入或输出状态**

函数原型：

Visual C++ & C++Builder:

BOOL EnableStsDIO (HANDLE hDevice,
BOOL bSts[3])

Visual Basic:

Declare Function EnableStsDIO Lib "CPCI4224" (ByVal hDevice As Long, _
ByVal bSts (0 to 2) As Boolean) As Boolean

Delphi:

Function EnableStsDIO(hDevice : Integer;
bSts : Pointer):Boolean;
StdCall; External 'CPCI4224' Name ' EnableStsDIO ';

LabVIEW:

请参考相关演示程序。

功能：设置开关量输入或输出状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bSts bSts[0]～bSts[2]分别控制 PA、PB、PC 端口。TRUE：开关量输入，FALSE：开关量输出。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice ReleaseDevice

一、对PA口操作

♦ **PA 口 DIO0～DIO7 八路开关量输入**

函数原型：

Visual C++ & C++Builder:

BOOL GetDeviceDI_PA (HANDLE hDevice,

BYTE bDOISts [8])

Visual Basic:

Declare Function GetDeviceDI_PA Lib "CPCI4224" (ByVal hDevice As Long, _
ByVal bDOISts (0 to 7) As Byte) As Boolean

Delphi:

Function GetDeviceDI_PA (hDevice : Integer;
bDOISts : Pointer):Boolean;
StdCall; External 'CPCI4224' Name ' GetDeviceDI_PA ';

LabVIEW:

请参考相关演示程序。

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

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

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

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

相关函数： CreateDevice EnableStsDIO SetDeviceDO_PA
ReleaseDevice

♦ PA 口 DIO0～DIO7 八路开关量输出

函数原型：

Visual C++ & C++Builder:

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

Visual Basic:

Declare Function SetDeviceDO_PA Lib "CPCI4224" (ByVal hDevice As Long, _
ByVal bDOISts(0 to 7)As Byte) As Boolean

Delphi:

Function SetDeviceDO_PA (hDevice : Integer;
bDOISts : Pointer):Boolean;
StdCall; External 'CPCI4224' Name ' SetDeviceDO_PA ';

LabView：

请参考相关演示程序。

功能：负责将设备上的 DIO0～DIO7 输出开关量置成相应的状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDOISts 八路开关量输出状态的参数结构，共有 8 个成员变量，分别对应于DIO0～DIO7 路开关量输出状态位。比如置bDOISts[0]为“1”则使 0 通道处于“开”状态，若为“0”则置 0 通道为“关”状态。其他同理。请注意，在实际执行这个函数之前，必须对这个参数结构的DIO0 至DIO7 共 8 个成员变量赋初值，其值必须为“1”或“0”。具体定义请参考《DO数字开关量输出参数介绍》。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice EnableStsDIO GetDeviceDI_PA
ReleaseDevice

二、对PB口操作

♦ PB 口 DIO8～DIO15 八路开关量输入

函数原型：

Visual C++ & C++Builder:

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

Visual Basic:

```
Declare Function GetDeviceDI_PB Lib "CPCI4224" (ByVal hDevice As Long, _
                                                ByVal bDOISts(0 to 7 ) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI_PB ( hDevice : Integer;
                          bDOISts : Pointer):Boolean;
                          StdCall; External 'CPCI4224' Name ' GetDeviceDI_PB ';
```

LabVIEW:

请参考相关演示程序。

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

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

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

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

相关函数： CreateDevice　EnableStsDIO　SetDeviceDO_PB　ReleaseDevice

♦ PB 口 DIO8～DIO15 八路开关量输出

函数原型：

Visual C++ & C++Builder:

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

Visual Basic:

```
Declare Function SetDeviceDO_PB Lib "CPCI4224" ( ByVal hDevice As Long, _
                                                 ByVal bDOISts(0 to 7 )As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO_PB (hDevice : Integer;
                         bDOISts : Pointer):Boolean;
                         StdCall; External 'CPCI4224' Name ' SetDeviceDO_PB ';
```

LabView：

请参考相关演示程序。

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

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

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

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

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice EnableStsDIO GetDeviceDI_PB ReleaseDevice

三、对PC口操作

♦ PC 口 DIO16～DIO23 八路开关量输入

函数原型：

Visual C++ & C++Builder:

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

Visual Basic:

```
Declare Function GetDeviceDI_PC Lib "CPCI4224" (ByVal hDevice As Long, _
                                                ByVal bDOISts(0 to 7 ) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI_PC ( hDevice : Integer;
                          bDOISts : Pointer):Boolean;
                          StdCall; External 'CPCI4224' Name ' GetDeviceDI_PC ';
```

LabVIEW:

请参考相关演示程序。

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

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

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

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

相关函数： CreateDevice EnableStsDIO SetDeviceDO_PC ReleaseDevice

♦ PC 口 DIO16～DIO23 八路开关量输出

函数原型：

Visual C++ & C++Builder:

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

Visual Basic:

```
Declare Function SetDeviceDO_PC Lib "CPCI4224" ( ByVal hDevice As Long, _
                                                 ByVal bDOISts(0 to 7 )As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO_PC (hDevice : Integer;
                         bDOISts : Pointer):Boolean;
                         StdCall; External 'CPCI4224' Name ' SetDeviceDO_PC ';
```

LabView：

请参考相关演示程序。

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

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

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

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice EnableStsDIO GetDeviceDI_PC ReleaseDevice

❖ **以上函数调用一般顺序**

① CreateDevice

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

③ ReleaseDevice

用户可以反复执行第②步，以进行数字 I/O 的输入输出。

第四节、UART 通信操作函数原型说明

♦ **通信口工作模式选择**

函数原型：

Visual C++ & C++Builder:

```
BOOL SetUARTMode (HANDLE hDevice,
                  LONG lUARTMode,
                  LONG lUARTChannel)
```

Visual Basic:

```
Declare Function SetUARTMode Lib "CPCI4224" (ByVal hDevice As Long, _
                                             ByVal lUARTMode As Long, _
                                             ByVal lUARTChannel As Long) As Boolean
```

Delphi:

```
Function SetUARTMode (hDevice : Integer;
                      lUARTMode : LongInt;
                      lUARTChannel: LongInt):Boolean;
                      StdCall; External 'CPCI4224' Name ' SetUARTMode ';
```

LabView：

请参考相关演示程序。

功能：通信口工作模式选择。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

lUARTMode 工作模式。

常量名	常量值	功能定义
CPCI4224_UART_232	0x00	UART232
CPCI4224_UART_422	0x01	UART422
CPCI4224_UART_485	0x02	UART485

lUARTChannel 通道选择。0 代表 A 路，1 代表 B 路，2 代表 C 路，3 代表 D 路。
返回值：若成功，返回 TRUE，否则返回 FALSE。
相关函数： CreateDevice SetUARTMode InitUARTPara GetUARTSendDataSts GetUARTReceiveDataCount UARTWriteData UARTReadData ReleaseDevice

♦ **初始化 UART 参数**

函数原型：

Visual C++ & C++Builder:

```
BOOL InitUARTPara (HANDLE hDevice,
                   PCPCI4224_PARA_UART pUARTPara,
                   LONG lUARTChannel)
```

Visual Basic:

```
Declare Function InitUARTPara Lib "CPCI4224" (ByVal hDevice As Long, _
                                              ByRef pUARTPara As CPCI4224_PARA_UART,_
                                              ByVal lUARTChannel As Long) As Boolean
```

Delphi:

```
Function InitUARTPara (hDevice : Integer;
                       pUARTPara : PCPCI4224_PARA_UART;
                       lUARTChannel : Boolean):Boolean;
                       StdCall; External 'CPCI4224' Name ' InitUARTPara ';
```

LabView：

请参考相关演示程序。

功能：初始化 UART 参数。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
pUARTPara UART 参数。
lUARTChannel 通道选择。0 代表 A 路，1 代表 B 路，2 代表 C 路，3 代表 D 路。
返回值：若成功，返回 TRUE，否则返回 FALSE。
相关函数： CreateDevice SetUARTMode InitUARTPara GetUARTSendDataSts GetUARTReceiveDataCount UARTWriteData UARTReadData ReleaseDevice

♦ **获取发送数据过程中设备的各种状态**

函数原型：

Visual C++ & C++Builder:

```
BOOL GetUARTSendDataSts (HANDLE hDevice,
                         PCPCI4224_STATUS_UART pUARTSts,
                         LONG lUARTChannel)
```

Visual Basic:

```
Declare Function GetUARTSendDataSts Lib "CPCI4224" (ByVal hDevice As Long, _
                                                    ByRef pUARTSts As CPCI4224_STATUS_UART,_
                                                    ByVal lUARTChannel As Long) As Boolean
```

Delphi:

```
Function GetUARTSendDataSts (hDevice : Integer;
                             pUARTSts: PCPCI4224_STATUS_UART;
                             lUARTChannel : Boolean):Boolean;
                             StdCall; External 'CPCI4224' Name ' GetUARTSendDataSts ';
```

LabView：

请参考相关演示程序。

功能：获取发送数据过程中设备的各种状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pUARTSts 设备的各种状态。

lUARTChannel 通道选择。0 代表 A 路，1 代表 B 路，2 代表 C 路，3 代表 D 路。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice　SetUARTMode　InitUARTPara　GetUARTSendDataSts　GetUARTReceiveDataCount　UARTWriteData　UARTReadData　ReleaseDevice

- **获取接收到的数据个数**

函数原型：

Visual C++ & C++Builder:

```
BOOL GetUARTReceiveDataCount (HANDLE hDevice,
                              PULONG pulDataCount,
                              LONG lUARTChannel)
```

Visual Basic:

```
Declare Function GetUARTReceiveDataCount Lib "CPCI4224" (ByVal hDevice As Long, _
                                                         ByRef pulDataCount As Long,_
                                                         ByVal lUARTChannel As Long) As Boolean
```

Delphi:

```
Function GetUARTReceiveDataCount (hDevice : Integer;
                                  pulDataCount: Pointer;
                                  lUARTChannel : Boolean):Boolean;
                                  StdCall; External 'CPCI4224' Name ' GetUARTReceiveDataCount ';
```

LabView：

请参考相关演示程序。

功能：获取接收到的数据个数。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pulDataCount 数据个数。

lUARTChannel 通道选择。0 代表 A 路，1 代表 B 路，2 代表 C 路，3 代表 D 路。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice　SetUARTMode　InitUARTPara　GetUARTSendDataSts　GetUARTReceiveDataCount　UARTWriteData　UARTReadData　ReleaseDevice

- **发送数据**

函数原型：

Visual C++ & C++Builder:

```
BOOL UARTWriteData (HANDLE hDevice,
                    BYTE DataBuffer[],
                    LONG nWriteSizeWords,
                    LONG lUARTChannel)
```

Visual Basic:

```
Declare Function UARTWriteData Lib "CPCI4224" (ByVal hDevice As Long, _
                                               ByRef DataBuffer As Byte,_
                                               ByVal nWriteSizeWords As Long, _
                                               ByVal lUARTChannel As Long) As Boolean
```

Delphi:

```
Function UARTWriteData (hDevice : Integer;
                        DataBuffer : Pointer;
                        nWriteSizeWords : LongInt;
                        lUARTChannel : Boolean):Boolean;
                        StdCall; External 'CPCI4224' Name ' UARTWriteData ';
```

LabView：

请参考相关演示程序。

功能：发送数据。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

DataBuffer 接受原始数据的用户缓冲区。

nWriteSizeWords 写入的数据长度(字)。

lUARTChannel 通道选择。0 代表 A 路，1 代表 B 路，2 代表 C 路，3 代表 D 路。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice SetUARTMode InitUARTPara GetUARTSendDataSts GetUARTReceiveDataCount UARTWriteData UARTReadData ReleaseDevice

- **接收数据**

函数原型：

Visual C++ & C++Builder:

```
BOOL UARTReadData (HANDLE hDevice,
                   BYTE DataBuffer[],
                   LONG nReadSizeWords,
                   LONG lUARTChannel)
```

Visual Basic:

```
Declare Function UARTReadData Lib "CPCI4224" (ByVal hDevice As Long, _
                                              ByRef DataBuffer As Byte,_
                                              ByVal nReadSizeWords As Long, _
                                              ByVal lUARTChannel As Long) As Boolean
```

Delphi:

```
Function UARTReadData (hDevice : Integer;
                       DataBuffer : Pointer;
```

```
                    nReadSizeWords : LongInt;
                    lUARTChannel : Boolean):Boolean;
                    StdCall; External 'CPCI4224' Name ' UARTReadData ';
```

LabView：

请参考相关演示程序。

功能：接收数据。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

DataBuffer 接受原始数据的用户缓冲区。

nReadSizeWords 读入的数据长度(字)。

lUARTChannel 通道选择。0 代表 A 路，1 代表 B 路，2 代表 C 路，3 代表 D 路。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice　SetUARTMode　InitUARTPara　GetUARTSendDataSts　GetUARTReceiveDataCount　UARTWriteData　UARTReadData　ReleaseDevice

第四节、CAN 操作函数原型说明

- **使能 CAN 通道**

函数原型：

Visual C++ & C++Builder:

```
BOOL EnableCANChannle (HANDLE hDevice,
                       BOOL bEnableCH0,
                       BOOL bEnableCH1)
```

Visual Basic:

```
Declare Function EnableCANChannle Lib "CPCI4224" ( ByVal hDevice As Long, _
                                                   ByVal bEnableCH0 As Boolean, _
                                                   ByVal bEnableCH1As Boolean) As Boolean
```

Delphi:

```
Function EnableCANChannle (hDevice : Integer;
                           bEnableCH0 : Boolean;
                           bEnableCH1 : Boolean) : Boolean;
                           StdCall; External 'CPCI4224' Name ' EnableCANChannle ';
```

LabVIEW:

请参考相关演示程序。

功能：使能 CAN 通道。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bEnableCH0 是否使能 CAN 通道 0。=TRUE，使能通道 0；=FALSE，禁止通道 0。

bEnableCH1 是否使能 CAN 通道 1。=TRUE，使能通道 1；=FALSE，禁止通道 1。

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

相关函数： CreateDevice　EnableCANChannle　InitCAN　InitCANEx　SendFrameSingle　SendFrameMultiple　SendFrameMultipleEx　ReceiveFrame　ReleaseDevice

♦ **初始化 CAN 卡**

函数原型：

Visual C++ & C++Builder：

```
BOOL InitCAN( HANDLE hDevice,
              PCPCI4224_INIT_PARA pPara,
              LONG lCANChannel)
```

Visual Basic:

```
Declare Function InitCAN Lib "CPCI4224" (ByVal hDevice As Long, _
                                         ByRef pPara As CPCI4224_INIT_PARA,_
                                         ByVal lCANChannel As Long) As Boolean
```

Delphi:

```
Function InitCAN( hDevice : Integer;
                  pPara: PCPCI4224_INIT_PARA;
                  lCANChannel : LongInt):Boolean;
                  StdCall; External 'CPCI4224' Name 'InitCAN';
```

LabView:

请参考相关演示程序。

功能：初始化 CAN 卡。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pPara 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。

lCANChannel CAN 通道号，取值为[0, 1]。

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

相关函数： CreateDevice　EnableCANChannle　InitCAN　InitCANEx　SendFrameSingle　SendFrameMultiple　SendFrameMultipleEx　ReceiveFrame　ReleaseDevice

♦ **初始化 CAN 卡**

函数原型：

Visual C++ & C++Builder：

```
BOOL InitCANEx ( HANDLE hDevice,
                 PCPCI4224_INIT_PARA pPara,
                 BYTE byTimer0,
                 BYTE byTimer1,
                 LONG lCANChannel)
```

Visual Basic:

```
Declare Function InitCANEx Lib "CPCI4224" (ByVal hDevice As Long, _
                                           ByRef pPara As CPCI4224_INIT_PARA,_
                                           ByVal byTimer0 As Byte,_
                                           ByVal byTimer1 As Byte,_
                                           ByVal lCANChannel As Long) As Boolean
```

Delphi:

```
Function InitCANEx ( hDevice : Integer;
                     pPara: PCPCI4224_INIT_PARA;
```

```
                    byTimer0 : Byte;
                    byTimer1 : Byte;
                    lCANChannel : LongInt):Boolean;
                    StdCall; External 'CPCI4224' Name ' InitCANEx ';
```

LabView:

请参考相关演示程序。

功能：初始化 CAN 卡。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pPara 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。

byTimer0 定时器 0。

byTimer1 定时器 1。

lCANChannel CAN 通道号，取值为[0, 1]。

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

相关函数： CreateDevice　EnableCANChannle　InitCAN　InitCANEx　SendFrameSingle　SendFrameMultiple　SendFrameMultipleEx　ReceiveFrame　ReleaseDevice

注：两种设置波特率的速率方法：

一、通过下面常量列表提供的默认 CAN 通讯波特率直接赋值到 BaudRate，例

BaudRate = CPCI4224_BAUD_1MHZ;　　// 设置 1M 波特率，调用 CPCI4224_InitCAN()

二、也可以直接设置两个定时器寄存器，实现用户设置速率。有两个形式，如下：

调用 CPCI4224_InitCANEx 函数对 byTimer0 和 byTimer1 两个参数设置 CAN 通讯波特率寄存器

此时 BaudRate 参数无效

◆ **发送单帧操作**

函数原型：

Visual C++ & C++Builder:

```
LONG SendFrameSingle(HANDLE hDevice,
                     PCPCI4224_FRAME psCanFrame,
                     LONG lCANChannel)
```

Visual Basic:

```
Declare Function SendFrameSingle Lib "CPCI4224" (ByVal hDevice As Long, _
                                                 ByRef psCanFrame As CPCI4224_FRAME, _
                                                 ByVal lCANChannel As Long) As Long
```

Delphi:

```
Function SendFrameSingle ( hDevice : Integer;
                           psCanFrame: PCPCI4224_FRAME;
                           lCANChannel : LongInt): LongInt;
                           StdCall; External 'CPCI4224' Name ' SendFrameSingle ';
```

LabView:

请参考相关演示程序。

功能：发送单帧操作。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
psCanFrame CAN 帧结构体。
lCANChannel CAN 通道号。
返回值：若成功，则返回发送字节数；否则返回 0。
相关函数： CreateDevice EnableCANChannle InitCAN
InitCANEx SendFrameSingle SendFrameMultiple
SendFrameMultipleEx ReceiveFrame ReleaseDevice

◆ **发送多帧操作**

函数原型：

Visual C++ & C++Builder:

```
LONG SendFrameMultiple(HANDLE hDevice,
                       PCPCI4224_FRAME psCanFrame,
                       LONG lFrameCount,
                       LONG lCANChannel)
```

Visual Basic:

```
Declare Function SendFrameMultiple Lib "CPCI4224" (ByVal hDevice As Long, _
                                                   ByRef psCanFrame As CPCI4224_FRAME, _
                                                   ByVal lFrameCount As Long, _
                                                   ByVal lCANChannel As Long) As Long
```

Delphi:

```
Function SendFrameMultiple ( hDevice : Integer;
                             psCanFrame: PCPCI4224_FRAME;
                             lFrameCount : LongInt;
                             lCANChannel : LongInt): LongInt;
                             StdCall; External 'CPCI4224' Name ' SendFrameMultiple ';
```

LabView:

请参考相关演示程序。

功能：发送多帧操作。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
psCanFrame CAN 帧结构体。
lFrameCount 帧数量。
lCANChannel CAN 通道号。
返回值：若成功，则返回发送成功的帧数量；否则返回 0。
相关函数： CreateDevice EnableCANChannle InitCAN
InitCANEx SendFrameSingle SendFrameMultiple
SendFrameMultipleEx ReceiveFrame ReleaseDevice

◆ **发送多帧操作，每帧内容可以不同**

函数原型：

Visual C++ & C++Builder:

```
LONG SendFrameMultipleEx(HANDLE hDevice,
```

```
                                CPCI4224_FRAME psCanFrame[],
                                LONG lFrameCount,
                                LONG lCANChannel)
```

Visual Basic:

```
Declare Function SendFrameMultipleEx Lib "CPCI4224" (ByVal hDevice As Long, _
                                                     ByVal psCanFrame As CPCI4224_FRAME, _
                                                     ByVal lFrameCount As Long, _
                                                     ByVal lCANChannel As Long) As Long
```

Delphi:

```
Function SendFrameMultipleEx(hDevice : Integer;
                             psCanFrame: CPCI4224_FRAME;
                             lFrameCount : LongInt;
                             lCANChannel : LongInt): LongInt;
                             StdCall; External 'CPCI4224' Name ' SendFrameMultipleEx ';
```

LabView:

请参考相关演示程序。

功能：发送多帧操作，每帧内容可以不同。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

psCanFrame[] CAN 帧结构体数组。

lFrameCount 帧数量。

lCANChannel CAN 通道号。

返回值：若成功，则返回发送成功的帧数量；否则返回 0。

相关函数： CreateDevice　EnableCANChannle　InitCAN　InitCANEx　SendFrameSingle　SendFrameMultiple　SendFrameMultipleEx　ReceiveFrame　ReleaseDevice

- **接收帧操作**

函数原型：

Visual C++ & C++Builder:

```
LONG ReceiveFrame(HANDLE hDevice,
                  PCPCI4224_FRAME pCanFrame,
                  LONG    lCANChannel)
```

Visual Basic:

```
Declare Function ReceiveFrame Lib "CPCI4224" (ByVal hDevice As Long, _
                                              ByRef pCanFrame As CPCI4224_FRAME, _
                                              ByVal lCANChannel As Long) As Long
```

Delphi:

```
Function ReceiveFrame (hDevice : Integer;
                       pCanFrame: PCPCI4224_FRAME;
                       lCANChannel : LongInt): LongInt;
                       StdCall; External 'CPCI4224' Name ' ReceiveFrame ';
```

LabView:

请参考相关演示程序。

功能：接收帧操作。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pCanFrame CAN 帧结构体。

lCANChannel CAN 通道号。

返回值：若成功，返回接收字节数，否则返回 0。

相关函数： CreateDevice EnableCANChannle InitCAN InitCANEx SendFrameSingle SendFrameMultiple SendFrameMultipleEx ReceiveFrame ReleaseDevice

第四章 硬件参数结构

第一节、UART 通信的硬件参数结构（CPCI4224_PARA_UART）

Visual C++ & C++Builder：

```
typedef struct _CPCI4224_PARA_UART
{
    LONG lDataBit;        // 数据位长度
    LONG lStopBit;        // 停止位
    LONG lParityBit;      // 校验位
    LONG lBaudRate;       // 波特率
} CPCI4224_PARA_UART, *PCPCI4224_PARA_UART;
```

Visual Basic：

```
Private Type CPCI4224_PARA_UART
    lDataBit As Long          ' 数据位长度
    lStopBit As Long          ' 停止位
    lParityBit As Long        ' 校验位
    lBaudRate As Long         ' 波特率
End Type
```

Delphi:

```
Type   // 定义结构体数据类型
    PCPCI4224_PARA_UART = ^ CPCI4224_PARA_UART;   // 指针类型结构
    CPCI4224_PARA_UART = record          // 标记为记录型
        lDataBit : LongInt;              // 数据位长度
        lStopBit : LongInt;              // 停止位
        lParityBit : LongInt;            // 校验位
        lBaudRate : LongInt;             // 波特率
End;
```

LabVIEW:

请参考相关演示程序。

lDataBit 数据位长度。它的取值如下表：

常量名	常量值	功能定义

CPCI4224_UART_DATABITS_5	0x00	5 位
CPCI4224_UART_DATABITS_6	0x01	6 位
CPCI4224_UART_DATABITS_7	0x02	7 位
CPCI4224_UART_DATABITS_8	0x03	8 位

lStopBit 停止位。它的取值如下表：

常量名	常量值	功能定义
CPCI4224_UART_STOPBITS_1	0x00	1 位
CPCI4224_UART_STOPBITS_2	0x01	1.5 位（若数据位长度为 5 位时）或 2 位（若数据位长度为 6，7 或 8 位）

lParityBit 校验位。它的取值如下表：

常量名	常量值	功能定义
CPCI4224_UART_PARITY_NONE	0x00	无校验
CPCI4224_UART_PARITY_ODD	0x01	奇校验
CPCI4224_UART_PARITY_EVEN	0x02	偶校验

第二节、UART 通信状态参数结构（CPCI4224_STATUS_UART）

Visual C++ & C++Builder：

```
typedef struct _  CPCI4224_STATUS_UART
{
    LONG bNotEmpty; //  板载 FIFO 存储器的非空标志，=TRUE 非空，  = FALSE  空
    LONG bHalf;       //  板载 FIFO 存储器的半满标志，=TRUE 半满以上，  = FALSE  半满以下
    LONG bOverflow; //  板载 FIFO 存储器的动态溢出标志，= TRUE 已发生溢出，= FALSE 未发生溢出
} CPCI4224_STATUS_UART, * CPCI4224_STATUS_UART;
```

Visual Basic：

```
Private Type CPCI4224_STATUS_UART
    bNotEmpty As Long
    bHalf As Long
    bOverflow As Long
End Type
```

Delphi:

```
Type   // 定义结构体数据类型
    PCPCI4224_STATUS_UART = ^ CPCI4224_STATUS_UART;   // 指针类型结构
    CPCI4224_STATUS_UART = record          // 标记为记录型
        bNotEmpty : LongInt;
        bHalf : LongInt;
        bOverflow : LongInt;
End;
```

LabVIEW:

请参考相关演示程序。

bNotEmpty 板载 FIFO 存储器的非空标志，=TRUE 非空， = FALSE 空
bHalf 板载 FIFO 存储器的半满标志，=TRUE 半满以上， = FALSE 半满以下
bOverflow 板载 FIFO 存储器的动态溢出标志，= TRUE 已发生溢出， = FALSE 未发生溢出

第三节、CAN 设备初始化结构介绍（CPCI4224_INIT_PARA_）

Visual C++ & C++Builder:

```
typedef struct _CPCI4224_INIT_PARA_
{
    ULONG  WorkMode;        // 工作方式，=0 时正常模式，=1 时为只听模式。
    ULONG  FilterEnable;    // 滤波使能，=0 时禁止滤波，=1 时标准帧滤波使能，=3 扩展帧滤波使能。
    ULONG  SFType;          // 帧类型发送帧类型，=0 时为正常发送，=1 时为单次发送，
                            // =2 时为自发自收，=3 时为单次自发自收，只在此帧为发送帧时有意义。
    ULONG  CheckCode;       // 验收码
    ULONG  MaskOffCode;     // 屏蔽码
    ULONG  BaudRate;        // 总线速率
} CPCI4224_INIT_PARA, *PCPCI4224_INIT_PARA;
```

Visual Basic：

```
Private Type CPCI4224_INIT_PARA_
    WorkMode As Long
    FilterEnable As Long
    SFType As Long
    CheckCode As Long
    MaskOffCode As Long
    BaudRate As Long
End Type
```

Delphi:

```
Type   // 定义结构体数据类型
    PCPCI4224_INIT_PARA_ = ^ CPCI4224_INIT_PARA_;   // 指针类型结构
    CPCI4224_INIT_PARA_ = record            // 标记为记录型
        WorkMode: LongWord;
        FilterEnable: LongWord;
        SFType: LongWord;
        CheckCode: LongWord;
        MaskOffCode: LongWord;
        BaudRate: LongWord;
End;
```

LabView:

请参考相关演示程序。

硬件参数说明：此结构主要用于设定设备硬件参数值，用这个参数结构对设备进行硬件配置由InitCAN函数完成。

WorkMode 工作方式选择。它的其选项值如下表：

常量名	常量值	功能定义
CPCI4224_WORKMODE_NORMAL	0x0000	正常模式
CPCI4224_WORKMODE_LISTEN	0x0001	只听模式

FilterEnable 滤波使能。它的其选项值如下表：

常量名	常量值	功能定义
CPCI4224_FILTERENABLE_NONE	0x0000	禁止滤波
CPCI4224_FILTERENABLE_STANDARD	0x0001	标准帧滤波使能
CPCI4224_FILTERENABLE_EXTEND	0x0002	扩展帧滤波使能

SFType 发送帧类型选择。它的其选项值如下表：

常量名	常量值	功能定义
CPCI4224_SFTYPE_NORMAL	0x0000	正常发送
CPCI4224_SFTYPE_SINGLES	0x0001	单次发送
CPCI4224_SFTYPE_SRSELF	0x0002	自发自收
CPCI4224_SFTYPE_SIN_RSELF	0x0003	单次自发自收

CheckCode 验收码。

MaskOffCode 屏蔽码。

BaudRate 波特率选项选择。它的其选项值如下表：

常量名	常量值	功能定义
CPCI4224_BAUD_10KHZ	0x0000	10KHZ
CPCI4224_BAUD_20KHZ	0x0001	20KHZ
CPCI4224_BAUD_50KHZ	0x0002	50KHZ
CPCI4224_BAUD_100KHZ	0x0003	100KHZ
CPCI4224_BAUD_125KHZ	0x0004	125KHZ
CPCI4224_BAUD_250KHZ	0x0005	250KHZ
CPCI4224_BAUD_500KHZ	0x0006	500KHZ
CPCI4224_BAUD_800KHZ	0x0007	800KHZ
CPCI4224_BAUD_1MHZ	0x0008	1MHZ

两种设置波特率的速率方法：

一、通过下面常量列表提供的默认 CAN 通讯波特率直接赋值到 BaudRate，例

BaudRate = CPCI4224_BAUD_1MHZ;　　// 设置 1M 波特率，调用 CPCI4224_InitCAN()

二、也可以直接设置两个定时器寄存器，实现用户设置速率。有两个形式，如下：

调用 CPCI4224_InitCANEx 函数对 byTimer0 和 byTimer1 两个参数设置 CAN 通讯波特率寄存器

此时 BaudRate 参数无效

第四节、CAN 帧结构（CPCI4224_FRAME_）

Visual C++ & C++Builder:

```
typedef struct _CPCI4224_FRAME_
{
    BYTE    FrameFormat;      // 帧格式      =0 为数据帧，=1 为远程帧
    BYTE    FrameType;        // 帧类型      =0 位标准帧，=1 为扩展帧
```

```
    BYTE    DataLen;          // 数据长度 <=8
    BYTE    Data[8];          // 数据
    DWORD dwCanID;            // 设备 ID
    DWORD dwSigntime;         // 接收帧时间标识
} CPCI4224_FRAME, *PCPCI4224_FRAME;
```

Visual Basic：

```
Type CPCI4224_FRAME_
    FrameFormat As BYTE             ' 帧格式 =0 为数据帧，=1 为远程帧
    FrameType As BYTE               ' 帧类型 =0 位标准帧，=1 为扩展帧
    DataLen As BYTE                 ' 数据长度 <=8
    Data[16] As BYTE                ' 数据
    dwCanID As LongWord             ' 设备 ID
    dwSigntime As LongWord          ' 接收帧时间标识
End Type
```

Delphi:

```
Type   // 定义结构体数据类型
    PCPCI4224_FRAME_   = ^ CPCI4224_FRAME_;   // 指针类型结构
    CPCI4224_FRAME_    = record                // 标记为记录型
    FrameFormat : BYTE;                  // 帧格式 =0 为数据帧，=1 为远程帧
    FrameType : BYTE;                    // 帧类型 =0 位标准帧，=1 为扩展帧
    DataLen : BYTE;                      // 数据长度 <=8
    Data : Array[0..7] of BYTE;          // 数据
    dwCanID : LongWord;                  // 设备 ID
    dwSigntime: LongWord;                // 接收帧时间标识
End;
```

LabView:

请参考其演示程序。

各参数说明：

FrameFormat 帧格式选择，它的取值如下表：

常量名	常量值	功能定义
CPCI4224_FFORMAT_DATA	0x00	数据帧
CPCI4224_FFORMAT_REMOTE	0x01	远程帧

FrameType 帧类型选择，它的取值如下表：

常量名	常量值	功能定义
CPCI4224_FTYPE_STANDARD	0x00	标准帧
CPCI4224_FTYPE_EXTEND	0x01	扩展帧

DataLen 数据长度，取值小于等于 8。

Data[8] 数据个数由 DataLen 决定。

dwCanID 设备 ID 号。

dwSigntime 接收帧时间标识。

第五章　上层用户函数接口应用实例

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

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

第一节、简易程序演示说明

一、怎样使用函数进行开关量输入操作

其详细应用实例及正确代码请参考 Visual C++简易演示系统及源程序，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程(主要参考 CPCI4224.h 和 Sys.cpp)。

[程序]丨**[阿尔泰测控演示系统]**丿**[CPCI4224 2 路 CAN、4 路 UART、24 路 DIO 卡]**丨**[Microsoft Visual C++]**丿**[简易代码演示]**丿**[DI 开关量演示源程序]**

其默认存放路径为：系统盘\ART\CPCI4224\SAMPLES\VC\SIMPLE\DIO

二、怎样使用 UART 函数进行开关量输出操作

其详细应用实例及正确代码请参考 Visual C++简易演示系统及源程序，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程(主要参考 CPCI4224.h 和 Sys.cpp)。

[程序]丨**[阿尔泰测控演示系统]**丿**[CPCI4224 2 路 CAN、4 路 UART、24 路 DIO 卡]**丨**[Microsoft Visual C++]**丿**[简易代码演示]**丿**[UART 演示源程序]**

其默认存放路径为：系统盘\ART\CPCI4224\SAMPLES\VC\SIMPLE\UART

三、怎样使用 CAN 函数进行开关量输出操作

其详细应用实例及正确代码请参考 Visual C++简易演示系统及源程序，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程(主要参考 CPCI4224.h 和 Sys.cpp)。

[程序]丨**[阿尔泰测控演示系统]**丿**[CPCI4224 2 路 CAN、4 路 UART、24 路 DIO 卡]**丨**[Microsoft Visual C++]**丿**[简易代码演示]**丿**[CAN 演示源程序]**

其默认存放路径为：系统盘\ART\CPCI4224\SAMPLES\VC\SIMPLE\CAN

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 CPCI4224.h 和 Sys.cpp)。

[程序]丨**[阿尔泰测控演示系统]**丿**[CPCI4224 2 路 CAN、4 路 UART、24 路 DIO 卡]**丨**[Microsoft Visual C++]**丿**[高级代码演示]**

其默认存放路径为：系统盘\ART\CPCI4224\SAMPLES\VC\ADVANCED

其他语言的演示可以用上面类似的方法找到。

第六章 公共接口函数介绍

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

第一节、公用接口函数总列表（每个函数省略了前缀“CPCI4224_”）

函数名	函数功能	备注
① PCI 总线内存映射寄存器操作函数		
GetDeviceAddr	取得指定 PCI 设备寄存器操作基地址	底层用户
GetDeviceBar	取得指定的指定设备寄存器组 BAR 地址	底层用户
GetDevVersion	获取设备固件及程序版本	底层用户
WriteRegisterByte	以字节(8Bit)方式写寄存器端口	底层用户
WriteRegisterWord	以字(16Bit)方式写寄存器端口	底层用户
WriteRegisterULong	以双字(32Bit)方式写寄存器端口	底层用户
ReadRegisterByte	以字节(8Bit)方式读寄存器端口	底层用户
ReadRegisterWord	以字(16Bit)方式读寄存器端口	底层用户
ReadRegisterULong	以双字(32Bit)方式读寄存器端口	底层用户
② ISA 总线 I/O 端口操作函数		
WritePortByte	以字节(8Bit)方式写 I/O 端口	用户程序操作端口
WritePortWord	以字(16Bit)方式写 I/O 端口	用户程序操作端口
WritePortULong	以无符号双字(32Bit)方式写 I/O 端口	用户程序操作端口
ReadPortByte	以字节(8Bit)方式读 I/O 端口	用户程序操作端口
ReadPortWord	以字(16Bit)方式读 I/O 端口	用户程序操作端口
ReadPortULong	以无符号双字(32Bit)方式读 I/O 端口	用户程序操作端口
③ 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	

第二节、PCI 内存映射寄存器操作函数原型说明

◆ **取得指定内存映射寄存器的线性地址和物理地址**

函数原型：

Visual C++ & C++ Builder:

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

Visual Basic:

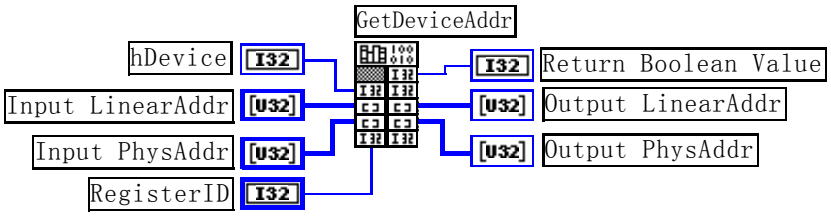
```
Declare Function GetDeviceAddr Lib "CPCI4224" (ByVal hDevice As Long, _
                                               ByRef LinearAddr As Long, _
                                               ByRef PhysAddr As Long, _
                                               ByVal RegisterID As Integer = 0) As Boolean
```

Delphi:

```
Function GetDeviceAddr(hDevice : Integer;
                       LinearAddr : Pointer;
                       PhysAddr : Pointer;
                       RegisterID : Integer = 0) : Boolean;
```

StdCall; External 'CPCI4224' Name ' GetDeviceAddr ';

LabVIEW:



功能：取得 PCI 设备指定的内存映射寄存器的线性地址。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

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

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

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

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

相关函数： CreateDevice ReleaseDevice

- **取得指定的指定设备寄存器组 BAR 地址**

函数原型：

Visual C++ & C++ Builder:

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

Visual Basic:

```
Declare Function GetDeviceBar Lib "CPCI4224" (ByVal hDevice As Long, _
                                              ByVal pulPCIBar (0 to 5) As Long) As Boolean
```

Delphi:

```
Function GetDeviceBar (hDevice : Integer;
                       pulPCIBar : Pointer) : Boolean;
                       StdCall; External 'CPCI4224' Name ' GetDeviceBar';
```

LabVIEW:

请参考相关演示程序。

功能：取得指定的指定设备寄存器组 BAR 地址。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pulPCIBar 返回 PCI BAR 所有地址。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice ReleaseDevice

♦ **获取设备固件及程序版本**

函数原型：

Visual C++ & C++ Builder:

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

Visual Basic:

```
Declare Function GetDevVersion Lib "CPCI4224" (ByVal hDevice As Long, _
                                               ByRef pulFmwVersion As Long, _
                                               ByRef pulDriverVersion As Long) As Boolean
```

Delphi:

```
Function GetDevVersion (hDevice : Integer;
                        pulFmwVersion: Pointer;
                        pulDriverVersion: Pointer) : Boolean;
                        StdCall; External 'CPCI4224' Name ' GetDevVersion ';
```

LabVIEW:

请参见相关演示程序。

功能：获取设备固件及程序版本。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pulFmwVersion 指针参数，用于取得固件版本。

pulDriverVersion 指针参数，用于取得驱动版本。

返回值：如果执行成功，则返回 TRUE，否则会返回 FALSE。

相关函数： CreateDevice ReleaseDevice

♦ **以单字节（即 8 位）方式写 PCI 内存映射寄存器的某个单元**

函数原型:

Visual C++ & C++ Builder:

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

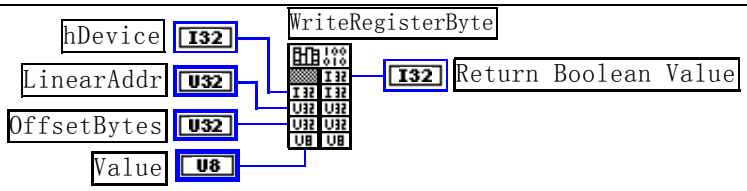
Visual Basic:

```
Declare Function WriteRegisterByte Lib "CPCI4224" (ByVal hDevice As Long, _
                                                   ByVal LinearAddr As Long, _
                                                   ByVal OffsetBytes As Long, _
                                                   ByVal Value As Byte ) As Boolean
```

Delphi:

```
Function WriteRegisterByte( hDevice : Integer;
                            LinearAddr : LongWord;
                            OffsetBytes : LongWord;
                            Value : Byte) : Boolean;
                            StdCall; External 'CPCI4224' Name ' WriteRegisterByte ';
```

LabVIEW:



功能：以单字节（即 8 位）方式写 PCI 内存映射寄存器。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

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

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

Value 输出 8 位整数。

返回值：若成功，返回 TRUE，否则返回 FALSE。

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

Visual C++ & C++ Builder 程序举例:

```
    :
HANDLE hDevice;
ULONG LinearAddr, PhysAddr, OffsetBytes;
hDevice = CreateDevice(0)
if (!GetDeviceAddr(hDevice, &LinearAddr,  &PhysAddr, 0) )
{
    AfxMessageBox “取得设备地址失败…”;
}
OffsetBytes = 100;    // 指定操作相对于线性基地址偏移 100 个字节数位置的单元
WriteRegisterByte(hDevice, LinearAddr, OffsetBytes, 0x20);  // 往指定映射寄存器单元写入 8 位的十六进制数据 20
ReleaseDevice( hDevice );  // 释放设备对象
    :
```

Visual Basic 程序举例:

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

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

函数原型:

Visual C++ & C++ Builder:

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

```
                                ULONG OffsetBytes,
                                WORD Value)
```

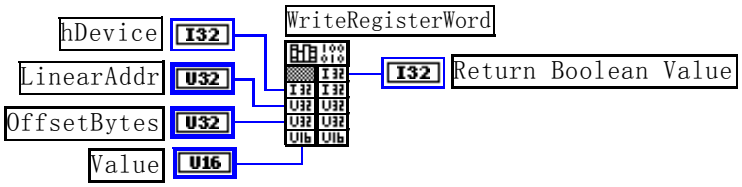
Visual Basic:

```
Declare Function WriteRegisterWord Lib "CPCI4224" (ByVal hDevice As Long, _
                                                   ByVal LinearAddr As Long, _
                                                   ByVal OffsetBytes As Long, _
                                                   ByVal Value As Integer) As Boolean
```

Delphi:

```
Function WriteRegisterWord( hDevice : Integer;
                            LinearAddr : LongWord;
                            OffsetBytes : LongWord;
                            Value : Word) : Boolean;
                            StdCall; External 'CPCI4224' Name ' WriteRegisterWord ';
```

LabVIEW:



功能：以双字节（即 16 位）方式写 PCI 内存映射寄存器。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

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

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

Value　输出 16 位整型值。

返回值：无。

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

Visual C++ & C++ Builder 程序举例:

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

Visual Basic 程序举例:

```
    :
```

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

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

函数原型:

Visual C++ & C++ Builder:

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

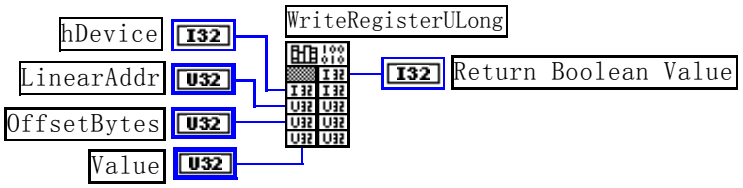
Visual Basic:

```
Declare Function WriteRegisterULong Lib "CPCI4224" (ByVal hDevice As Long, _
                                                    ByVal LinearAddr As Long, _
                                                    ByVal OffsetBytes As Long, _
                                                    ByVal Value As Long) As Boolean
```

Delphi:

```
Function WriteRegisterULong(hDevice : Integer;
                            LinearAddr : LongWord;
                            OffsetBytes : LongWord;
                            Value : LongWord) : Boolean;
                            StdCall; External 'CPCI4224' Name ' WriteRegisterULong ';
```

LabVIEW:



功能：以四字节（即 32 位）方式写 PCI 内存映射寄存器。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

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

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

Value 输出 32 位整型值。

返回值：若成功，返回 TRUE，否则返回 FALSE。

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

Visual C++ & C++ Builder 程序举例:

：

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

：

Visual Basic 程序举例:

：

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

：

- **以单字节（即 8 位）方式读 PCI 内存映射寄存器的某个单元**

函数原型:

Visual C++ & C++ Builder:

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

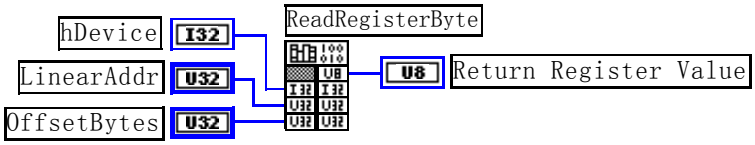
Visual Basic:

```
Declare Function ReadRegisterByte Lib "CPCI4224" (ByVal hDevice As Long, _
                                                  ByVal LinearAddr As Long, _
                                                  ByVal OffsetBytes As Long) As Byte
```

Delphi:

```
Function ReadRegisterByte(hDevice : Integer;
                          LinearAddr : LongWord;
                          OffsetBytes : LongWord) : Byte;
                          StdCall; External 'CPCI4224' Name ' ReadRegisterByte ';
```

LabVIEW:



功能：以单字节（即 8 位）方式读 PCI 内存映射寄存器的指定单元。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

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

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

返回值：返回从指定内存映射寄存器单元所读取的8位数据。

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

Visual C++ & C++ Builder 程序举例:

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

Visual Basic 程序举例:

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

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

函数原型:

Visual C++ & C++ Builder:

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

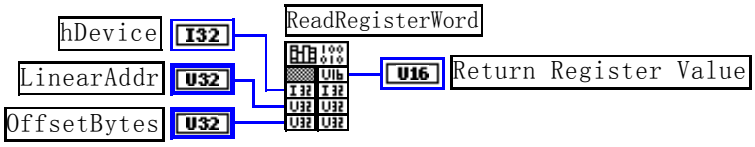
Visual Basic:

```
Declare Function ReadRegisterWord Lib "CPCI4224" (ByVal hDevice As Long, _
                                                  ByVal LinearAddr As Long, _
                                                  ByVal OffsetBytes As Long) As Integer
```

Delphi:

```
Function ReadRegisteWord(hDevice : Integer;
                         LinearAddr : LongWord;
                         OffsetBytes : LongWord) : Word;
                        StdCall; External 'CPCI4224' Name ' ReadRegisterWord ';
```

LabVIEW:



功能：以双字节（即 16 位）方式读 PCI 内存映射寄存器的指定单元。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

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

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

返回值：返回从指定内存映射寄存器单元所读取的 16 位数据。

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

Visual C++ & C++ Builder 程序举例:

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

Visual Basic 程序举例:

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

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

函数原型:

Visual C++ & C++ Builder:

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

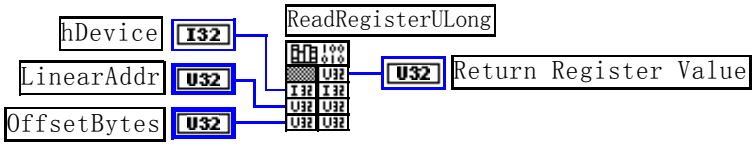
Visual Basic:

```
Declare Function ReadRegisterULong Lib "CPCI4224" (ByVal hDevice As Long, _
                                                   ByVal LinearAddr As Long, _
                                                   ByVal OffsetBytes As Long) As Long
```

Delphi:

```
Function ReadRegisterULong(hDevice : Integer;
                           LinearAddr : LongWord;
                           OffsetBytes : LongWord) : LongWord;
                          StdCall; External 'CPCI4224' Name ' ReadRegisterULong ';
```

LabVIEW:



功能：以四字节（即 32 位）方式读 PCI 内存映射寄存器的指定单元。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

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

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

返回值：返回从指定内存映射寄存器单元所读取的 32 位数据。

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

Visual C++ & C++ Builder 程序举例:

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

Visual Basic 程序举例:

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

第三节、IO 端口读写函数原型说明

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

♦ **以单字节(8Bit)方式写 I/O 端口**

Visual C++ & C++ Builder:

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

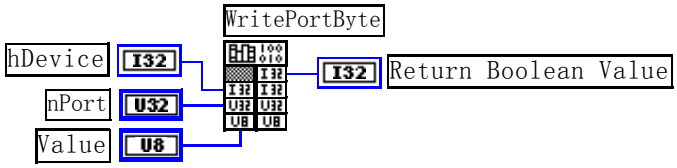
Visual Basic:

```
Declare Function WritePortByte Lib "CPCI4224" ( 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 'CPCI4224' Name ' WritePortByte ';
```

LabVIEW:



功能：以单字节(8Bit)方式写 I/O 端口。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nPort 设备的 I/O 端口号。

Value 写入由 nPort 指定端口的值。

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

相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

♦ **以双字(16Bit)方式写 I/O 端口**

Visual C++ & C++ Builder:

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

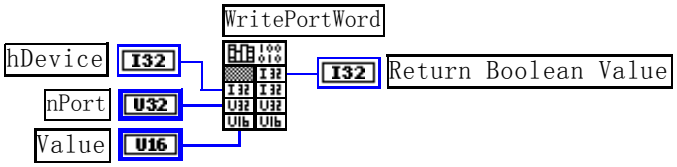
Visual Basic:

```
Declare Function WritePortWord Lib "CPCI4224" (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 'CPCI4224' Name ' WritePortWord ';
```

LabVIEW:



功能：以双字(16Bit)方式写 I/O 端口。

参数：
hDevice设备对象句柄，它应由CreateDevice创建。
nPort　设备的 I/O 端口号。
Value　写入由 nPort 指定端口的值。
返回值：若成功，返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码。
相关函数： CreateDevice　WritePortByte　WritePortWord
WritePortULong　ReadPortByte　ReadPortWord

- **以四字节(32Bit)方式写 I/O 端口**

Visual C++ & C++ Builder:

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

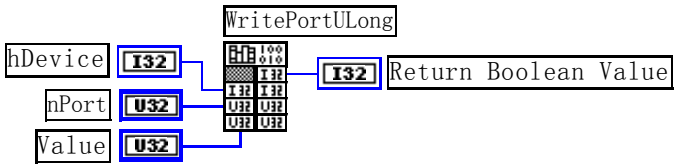
Visual Basic:

```
Declare Function WritePortULong Lib "CPCI4224" (ByVal hDevice As Long, _
                                                ByVal nPort As Long, _
                                                ByVal Value As Long ) As Boolean
```

Delphi:

```
Function WritePortULong(hDevice : Integer;
                        nPort : LongWord;
                        Value : LongWord) : Boolean;
                        StdCall; External 'CPCI4224' Name ' WritePortULong ';
```

LabVIEW:



功能：以四字节(32Bit)方式写 I/O 端口。
参数：
hDevice　设备对象句柄，它应由CreateDevice创建。
nPort　设备的 I/O 端口号。
Value　写入由 nPort 指定端口的值。
返回值：若成功，返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码。
相关函数： CreateDevice　WritePortByte　WritePortWord
WritePortULong　ReadPortByte　ReadPortWord

- **以单字节(8Bit)方式读 I/O 端口**

Visual C++ & C++ Builder:

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

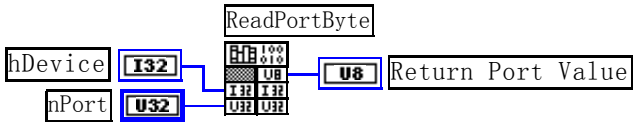
Visual Basic:

```
Declare Function ReadPortByte Lib "CPCI4224" (ByVal hDevice As Long, _
                                              ByVal nPort As Long ) As Byte
```

Delphi:

```
Function ReadPortByte(hDevice : Integer;
                      nPort : LongWord) : Byte;
                      StdCall; External 'CPCI4224' Name ' ReadPortByte ';
```

LabVIEW:



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

nPort 设备的 I/O 端口号。
返回值：返回由 nPort 指定的端口的值。
相关函数： CreateDevice WritePortByte WritePortWord
WritePortULong ReadPortByte ReadPortWord

♦ 以双字节(16Bit)方式读 I/O 端口

Visual C++ & C++ Builder:

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

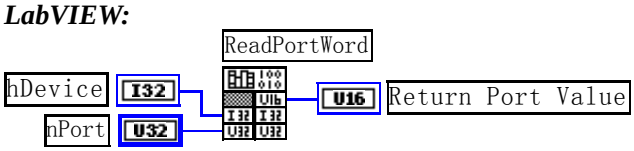
Visual Basic:

```
Declare Function ReadPortWord Lib "CPCI4224" ( ByVal hDevice As Long, _
                                               ByVal nPort As Long ) As Integer
```

Delphi:

```
Function ReadPortWord(hDevice : Integer;
                      nPort : LongWord) : Word;
                      StdCall; External 'CPCI4224' Name ' ReadPortWord ';
```

LabVIEW:



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

♦ 以四字节(32Bit)方式读 I/O 端口

Visual C++ & C++ Builder:

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

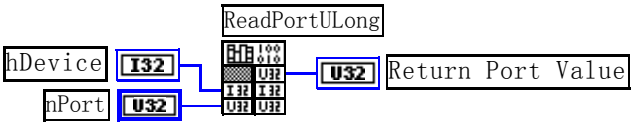
Visual Basic:

```
Declare Function ReadPortULong Lib "CPCI4224" ( ByVal hDevice As Long, _
                                                ByVal nPort As Long ) As Long
```

Delphi:

```
Function ReadPortULong(hDevice : Integer;
                       nPort : LongWord) : LongWord;
                       StdCall; External 'CPCI4224' Name ' ReadPortULong ';
```

LabVIEW:



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

第四节、线程操作函数原型说明

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

♦ 创建内核系统事件

函数原型：

Visual C++ & C++ Builder:

HANDLE CreateSystemEvent(void)

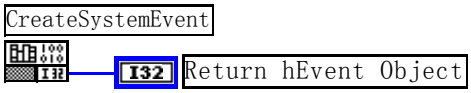
Visual Basic:

Declare Function CreateSystemEvent Lib " CPCI4224 " () As Long

Delphi:

Function CreateSystemEvent() : Integer;
StdCall; External 'CPCI4224' 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 " CPCI4224 " (ByVal hEvent As Long) As Boolean

Delphi:

Function ReleaseSystemEvent(hEvent : Integer) : Boolean;
StdCall; External 'CPCI4224' 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。