

CPCI2515 光隔离 DI/PWM 采集 卡

WIN2000/XP 驱动程序使用说明书



阿尔泰科技发展有限公司
产品研发部修订

请您务必阅读《[使用纲要](#)》，他会使您事半功倍!

目 录

目 录	1
第一章 版权信息与命名约定	2
第一节、版权信息	2
第二节、命名约定	2
第二章 使用纲要	2
第一节、使用上层用户函数，高效、简单	2
第二节、如何管理设备	2
第三节、如何实现开关量的简便操作	3
第四节、哪些函数对您不是必须的	3
第三章 设备专用函数接口介绍	3
第一节、设备驱动接口函数列表（每个函数省略了前缀“CPCI2515_”）	3
第二节、设备对象管理函数原型说明	4
第三节、光隔离离散量输入函数原型说明	5
第四节、计数测频函数原型说明	6
第四章 上层用户函数接口应用实例	7
第一节、简易程序演示说明	7
第二节、高级程序演示说明	8
第五章 公共接口函数介绍	8
第一节、公用接口函数总列表（每个函数省略了前缀“CPCI2515_”）	8
第二节、CPCI 内存映射寄存器操作函数原型说明	9
第三节、IO 端口读写函数原型说明	15
第四节、线程操作函数原型说明	17

第一章 版权信息与命名约定

第一节、版权信息

本软件产品及相关套件均属北京市阿尔泰科贸有限公司所有，其产权受国家法律绝对保护，除非本公司书面允许，其他公司、单位及个人不得非法使用和拷贝，否则将受到国家法律的严厉制裁。您若需要我公司产品及相关信息请及时与我们联系，我们将热情接待。

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 CPCIxxxx_ 则被省略。如 CPCI2515_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[SetDeviceDI](#)函数可用实现开关量的输入等。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 [SetDeviceDI](#) 函数实现开关量的输入操作，其各路开关量的输出状态由其 IDISts[32]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CPCI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“CPCI2515_”）

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 计数测频		
SetMeasureFre	设置设备测频模式	上层用户
GetMeasureFreSts	获取测频状态	上层用户
GetMeasureFre	获取被测频率值,返回被测频率值	上层用户
GetMeasureDutyratio	获取被测信号占空比值,返回被测信号占空比	上层用户
③光隔离离散量输入函数		
GetDeviceDI	光隔离离散量输入	上层用户

使用需知

Visual C++:

首先将 CPCI2515.h 和 CPCI2515.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 (CPCI2515.lib)加入到您的工程中。

```
#include "CPCI2515.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 CPCI2515.h 和 CPCI2515.lib 两个文件均可在演示程序文件夹下面找到。

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

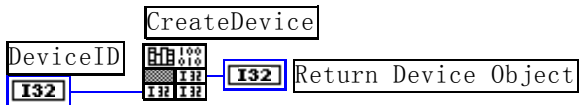
◆ 创建设备对象函数

函数原型：

Visual C++:

`HANDLE CreateDevice (int DeviceID = 0)`

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](#)

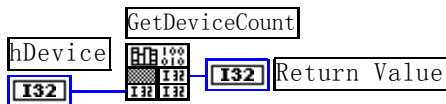
◆ 取得本计算机系统中 CPCI2515 设备的总数量

函数原型：

Visual C++:

`int GetDeviceCount (HANDLE hDevice)`

LabVIEW:



功能：取得 CPCI2515 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 CPCI2515 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++:

`int GetDeviceCurrentID (HANDLE hDevice)`

LabVIEW:

请参考相关演示程序。

函数原型:

Visual C++:

[Double GetMeasureFre \(HANDLE hDevice,
LONG IChannel\)](#)

LabView:

请参考相关演示程序。

功能: 获取被测频率值,返回被测频率值。

参数:

hDevice 设备对象句柄, 它应由[CreateDevice](#)创建。

IChannel 测频通道选择(0—7 通道)。

返回值: 若成功, 返回 TRUE, 否则返回 FALSE。

相关函数: [CreateDevice](#) [SetMeasureFre](#) [GetMeasureFreSts](#)
 [GetMeasureDutyratio](#) [GetMeasureFre](#) [ReleaseDevice](#)

◆ 获取被测信号占空比值,返回被测信号占空比

函数原型:

Visual C++:

[Double GetMeasureDutyratio \(HANDLE hDevice,
LONG IChannel\)](#)

LabView:

请参考相关演示程序。

功能: 获取被测信号占空比值,返回被测信号占空比。

参数:

hDevice 设备对象句柄, 它应由[CreateDevice](#)创建。

IChannel 测频通道选择(0—7 通道)。

返回值: 若成功, 返回 TRUE, 否则返回 FALSE。

相关函数: [CreateDevice](#) [SetMeasureFre](#) [GetMeasureFreSts](#)
 [GetMeasureDutyratio](#) [GetMeasureFre](#) [ReleaseDevice](#)

第四章 上层用户函数接口应用实例

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

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

第一节、简易程序演示说明

一、怎样使用 PWM 函数操作

其详细应用实例及正确代码请参考 Visual C++简易程序演示系统及源程序,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 CPCI2515.h 和 Sys.cpp)。

[程序] | [阿尔泰测控演示系统] | [CPCI2515 8 路 PWM、32 路 DI 卡] | [Microsoft Visual C++] | [简易代码演示] | [PWM 演示源程序]

其默认存放路径为: 系统盘\ART\CPCI2515\SAMPLES\VC\SIMPLE\PWM

二、怎样使用光隔离离散量输入函数操作

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 CPCI2515.h 和 Sys.cpp)。

[程序] | [阿尔泰测控演示系统] | [CPCI2515 8 路 PWM、32 路 DI 卡] | [Microsoft Visual C++] | [简易代码演示] | [DI 演示源程序]

其默认存放路径为: 系统盘\ART\CPCI2515\SAMPLES\VC\SIMPLE\DI

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 CPCI2515.h 和 Sys.cpp)。

[程序] | [阿尔泰测控演示系统] | [CPCI2515 8 路 PWM、32 路 DI 卡] | [Microsoft Visual C++] | [高级代码演示]

其默认存放路径为: 系统盘\ART\CPCI2515\SAMPLES\VC

其他语言的演示可以用上面类似的方法找到。

第五章 公共接口函数介绍

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

第一节、公用接口函数总列表 (每个函数省略了前缀“CPCI2515_”)

函数名	函数功能	备注
① PCI 总线内存映射寄存器操作函数		
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

第二节、CPCI 内存映射寄存器操作函数原型说明

◆ 取得指定的指定设备寄存器组 BAR 地址

函数原型:

Visual C++:

```
BOOL GetDeviceBar ( HANDLE hDevice,  
                   __int64 pbPCIBar[6])
```

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。

pbPCIBar 返回 PCI BAR 所有地址, 具体 PCI BAR 中有多少可用地址请看硬件说明书。

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++:

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

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](#)

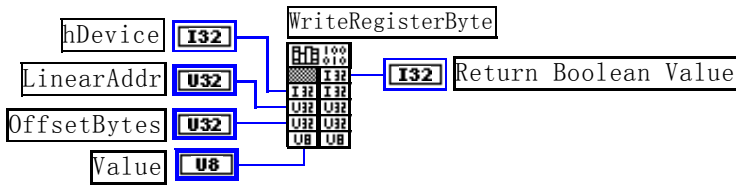
◆ 以单字节 (即 8 位) 方式写 CPCI 内存映射寄存器的某个单元

函数原型:

Visual C++:

```
BOOL WriteRegisterByte( HANDLE hDevice,  
                       __int64 pbLinearAddr,  
                       ULONG OffsetBytes,  
                       BYTE Value)
```

LabVIEW:



功能：以单字节（即 8 位）方式写 CPCI 内存映射寄存器。

参数：

hDevice 设备对象句柄，它应由 [CreateDevice](#) 创建。

pbLinearAddr 指定映射寄存器的线性基地址。

OffsetBytes 相对于基地址的偏移位置。

Value 往指定地址写入单字节数据（其地址由线性基地址和偏移位置决定）。

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++ 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 CPCI 内存映射寄存器的某个单元

函数原型：

Visual C++:

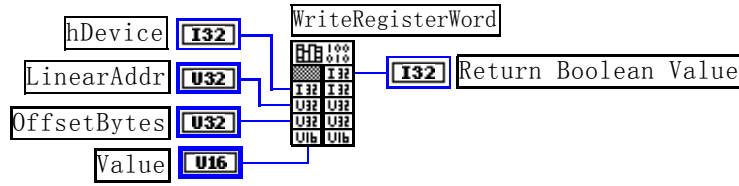
```

BOOL WriteRegisterWord(HANDLE hDevice,
    __int64 pbLinearAddr,
    ULONG OffsetBytes,

```

WORD Value)

LabVIEW:



功能：以双字节（即 16 位）方式写 CPCI 内存映射寄存器。

参数：

hDevice 设备对象句柄，它应由 [CreateDevice](#) 创建。

pbLinearAddr CPCI 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++ 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)
:

```

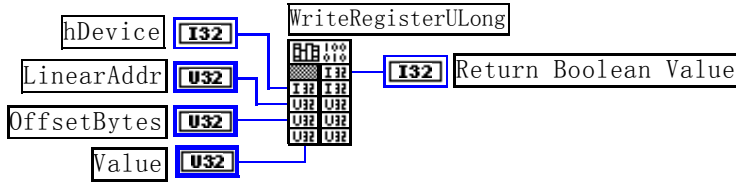
◆ 以四字节（即 32 位）方式写 CPCI 内存映射寄存器的某个单元

函数原型:

Visual C++:

BOOL WriteRegisterULONG(HANDLE hDevice,
 __int64 pbLinearAddr,
 ULONG OffsetBytes,
 ULONG Value)

LabVIEW:



功能: 以四字节（即 32 位）方式写 CPCI 内存映射寄存器。

参数:

hDevice 设备对象句柄，它应由 [CreateDevice](#) 创建。

pbLinearAddr CPCI 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++ 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
:

```

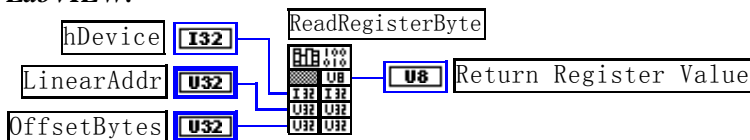
◆ 以单字节（即 8 位）方式读 CPCI 内存映射寄存器的某个单元

函数原型:

Visual C++:

BYTE ReadRegisterByte(HANDLE hDevice,
 __int64 pbLinearAddr,
 ULONG OffsetBytes)

LabVIEW:



功能：以单字节（即 8 位）方式读 CPCI 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。

pbLinearAddr CPCI 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++ 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 CPCI 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
:

```

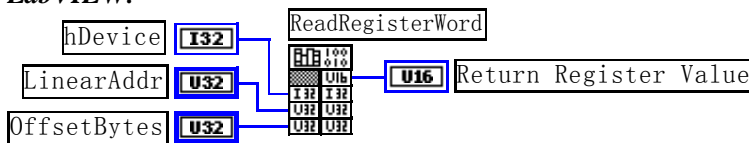
◆ 以双字节（即 16 位）方式读 CPCI 内存映射寄存器的某个单元

函数原型：

Visual C++:

WORD ReadRegisterWord(HANDLE hDevice,
 __int64 pbLinearAddr,
 ULONG OffsetBytes)

LabVIEW:



功能：以双字节（即 16 位）方式读 CPCI 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。

pbLinearAddr CPCI 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++ 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 CPCI 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
:

```

◆ 以四字节（即 32 位）方式读 CPCI 内存映射寄存器的某个单元

函数原型:

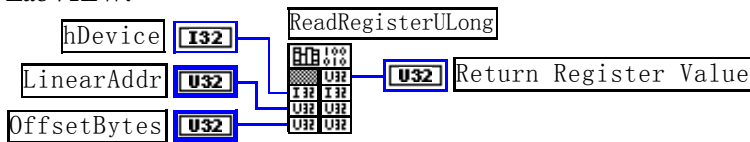
Visual C++:

```

ULONG ReadRegisterULong( HANDLE hDevice,
                        __int64 pbLinearAddr,
                        ULONG OffsetBytes)

```

LabVIEW:



功能: 以四字节（即 32 位）方式读 CPCI 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。

pbLinearAddr CPCI 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++ 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 CPCI 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
:

```

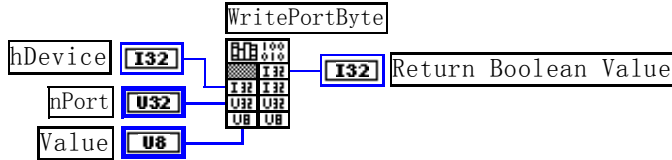
第三节、IO 端口读写函数原型说明

◆ 以单字节(8Bit)方式写 I/O 端口

Visual C++:

BOOL WritePortByte (HANDLE hDevice,
 __int64 pbPort,
 BYTE Value)

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。

pbPort 设备的 I/O 端口号。

Value 写入由 pbPort 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。

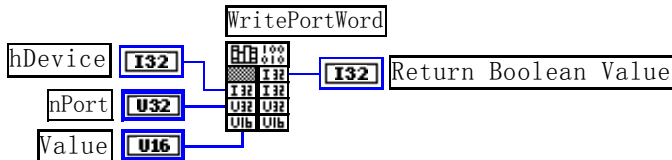
相关函数：[CreateDevice](#) [WritePortByte](#) [WritePortWord](#)
 [WritePortULong](#) [ReadPortByte](#) [ReadPortWord](#)

◆ 以双字(16Bit)方式写 I/O 端口

Visual C++:

BOOL WritePortWord (HANDLE hDevice,
 __int64 pbPort
 WORD Value)

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。

pbPort 设备的 I/O 端口号。

Value 写入由 pbPort 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。

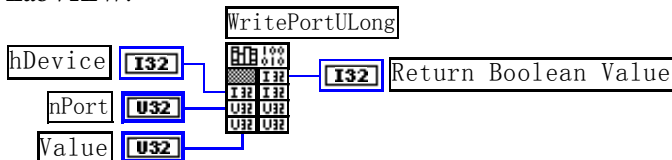
相关函数：[CreateDevice](#) [WritePortByte](#) [WritePortWord](#)
 [WritePortULong](#) [ReadPortByte](#) [ReadPortWord](#)

◆ 以四字节(32Bit)方式写 I/O 端口

Visual C++:

BOOL WritePortULong (HANDLE hDevice,
 __int64 pbPort,
 ULONG Value)

LabVIEW:



pbPort 设备的 I/O 端口号。

返回值: 返回由 pbPort 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++:

`HANDLE CreateSystemEvent(void)`

LabVIEW:

`CreateSystemEvent`

 `132` Return hEvent Object

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)。

相关函数: [CreateSystemEvent](#) [ReleaseSystemEvent](#)

◆ 释放内核系统事件

函数原型:

Visual C++:

`BOOL ReleaseSystemEvent(HANDLE hEvent)`

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。

相关函数: [CreateSystemEvent](#) [ReleaseSystemEvent](#)