CPCI2327 继电器采集卡

WIN2000/XP 驱动程序使用说明书

阿尔泰科技发展有限公司
产品研发部修订

请您务必阅读《使用纲要》，他会使您事半功倍!

目　录

第一章 版权信息与命名约定

第一节、版权信息

第二节、命名约定

一、为简化文字内容，突出重点，本文中提到的函数名通常为基本功能名部分，其前缀设备名如 CPCIxxxx_ 则被省略。如 CPCI2327_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、如何实现继电器的简便操作

当您有了 hDevice 设备对象句柄后，便可用 SetDeviceDO 函数实现取得继电器状态，其各路继电器状由其 bDOSts[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“CPCI2327_”）

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)	上层及底层用户
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	上层及底层用户
② AD 数据读取函数		
StartDeviceProAD	启动 AD 设备，开始转换	上层用户
ReadDeviceProAD_Npt	连续读取当前 CPCI 设备上的 AD 数据	上层用户
GetDevStatusProAD	取得当前 CPCI 设备 FIFO 半满状态	上层用户
ReadDeviceProAD_Half	连续批量读取 CPCI 设备上的 AD 数据	上层用户
StopDeviceProAD	暂停 AD 设备	上层用户
③ 继电器状态函数		
SetDeviceDO	取得继电器状态	上层用户

使用需知

Visual C++：

首先将 CPCI2327.h 和 CPCI2327.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(CPCI2327.lib)加入到您的工程中。

#include “CPCI2327.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 CPCI2327.h 和 CPCI2327.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

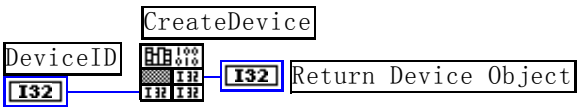
♦ 创建设备对象函数

函数原型：

Visual C++：

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： CreateDevice GetDeviceCount GetDeviceCurrentID ListDeviceDlg ReleaseDevice

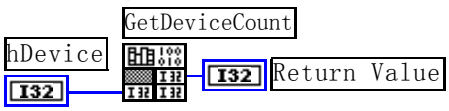
♦ 取得本计算机系统中 CPCI2327 设备的总数量

函数原型：

Visual C++:

int GetDeviceCount (HANDLE hDevice)

LabVIEW：



功能：取得 CPCI2327 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 CPCI2327 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++:

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

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 CPCI2327 设备各种配置信息**

函数原型：

Visual C++:

BOOL ListDeviceDlg (HANDLE hDevice)

LabVIEW：

请参考相关演示程序。

功能：列表系统中 CPCI2327 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 CPCI2327 设备的配置情况。

相关函数： CreateDevice　GetDeviceCount　GetDeviceCurrentID　ListDeviceDlg　ReleaseDevice

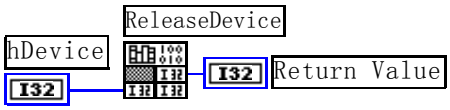
- **释放设备对象所占的系统资源及设备对象**

函数原型：

Visual C++:

BOOL ReleaseDevice(HANDLE hDevice)

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　GetDeviceCount　GetDeviceCurrentID　ListDeviceDlg　ReleaseDevice

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、AD 数据读取函数原型说明

- **启动 AD 设备(Start device AD for program mode)**

函数原型：

Visual C++:

BOOL StartDeviceProAD (HANDLE hDevice)

LabVIEW

请参考相关演示程序。

功能：启动AD设备，它必须在调用InitDeviceProAD后才能调用此函数。该函数除了启动AD设备开始转换以外，不改变设备的其他任何状态。

参数：hDevice 设备对象句柄，它应由CreateDevice或CreateDeviceEx创建。

返回值：如果调用成功，则返回TRUE，且AD立刻开始转换，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 GetDevStatusProAD
StartDeviceProAD ReadDeviceProAD_Npt
ReadDeviceProAD_Half StopDeviceProAD ReleaseDevice

♦ 读取 CPCI 设备上的 AD 数据

① 使用 FIFO 的非空标志读取 AD 数据

函数原型：

Visual C++:

```
BOOL ReadDeviceProAD_Npt( HANDLE hDevice,
                          ULONG ADBuffer[],
                          LONG nReadSizeWords,
                          PLONG nRetSizeWords)
```

LabVIEW：

请参考相关演示程序。

功能：一旦用户使用StartDeviceProAD后，应立即用此函数读取设备上的AD数据。此函数使用FIFO的非空标志进行读取AD数据。

参数：

hDevice 设备对象句柄，它应由CreateDevice或CreateDeviceEx创建。

ADBuffer 接受AD数据的用户缓冲区，它可以是一个用户定义的数组。关于如何将这些AD数据转换成相应的电压值，请参考《数据格式转换与排列规则》。

nReadSizeWords 指定一次ReadDeviceProAD_Npt操作应读取多少字数据到用户缓冲区。注意此参数的值不能大于用户缓冲区ADBuffer的最大空间。此参数值只与ADBuffer[]指定的缓冲区大小有效，而与FIFO存储器大小无效。

nRetSizeWords 返回实际读取的点数(或字数)。

返回值：其返回值表示所成功读取的数据点数(字)，也表示当前读操作在ADBuffer缓冲区中的有效数据量。通常情况下其返回值应与ReadSizeWords参数指定量的数据长度(字)相等，除非用户在这个读操作以外的其他线程中执行了ReleaseDeviceProAD函数中断了读操作，否则设备可能有问题。对于返回值不等于nReadSizeWords参数值的，用户可用GetLastErrorEx捕获当前错误码，并加以分析。

当前错误码	功能定义
0xE1000000	其他不可预知的错误
0xE2000000	表示用户提前终止读操作

注释：此函数也可用于单点读取和几个点的读取，只需要将nReadSizeWords设置成 1 或相应值即可。其使用方法请参考《高速大容量、连续不间断数据采集及存盘技术详解》章节。

相关函数： CreateDevice GetDevStatusProAD
StartDeviceProAD ReadDeviceProAD_Npt
ReadDeviceProAD_Half StopDeviceProAD ReleaseDevice

② 使用 FIFO 的半满标志读取 AD 数据

◆ **取得 FIFO 的状态标志**

函数原型：

Visual C++:

```
BOOL GetDevStatusProAD ( HANDLE hDevice,
                         PCPCI2327_STATUS_AD pADStatus)
```

LabVIEW:

请参考相关演示程序。

功能：一旦用户使用StartDeviceProAD后，应立即用此函数查询FIFO存储器的状态（半满标志、非空标志、溢出标志）。我们通常用半满标志去同步半满读操作。当半满标志有效时，再紧接着用ReadDeviceProAD_Half读取FIFO中的半满有效AD数据。

参数：

hDevice 设备对象句柄，它应由CreateDevice或CreateDeviceEx创建。

pADStatus 获得AD的各种当前状态。它属于结构体，具体定义请参考《AD状态参数结构（CPCI2327_STATUS_AD）》章节。

返回值：若调用成功则返回TRUE，否则返回FALSE，用户可以调用GetLastErrorEx函数取得当前错误码。若用户选择半满查询方式读取AD数据，则当GetDevStatusProAD函数取得的bHalf等于TRUE，应立即调用ReadDeviceProAD_Half读取FIFO中的半满数据。否则用户应继续循环轮询FIFO半满状态，直到有效为止。注意在循环轮询期间，可以用Sleep函数抛出一定时间给其他应用程序(包括本应用程序的主程序和其他子线程)，以提高系统的整体数据处理效率。

其使用方法请参考本文档的《高速大容量、连续不间断数据采集及存盘技术详解》章节。

相关函数： CreateDevice　GetDevStatusProAD　StartDeviceProAD　ReadDeviceProAD_Npt　ReadDeviceProAD_Half　StopDeviceProAD　ReleaseDevice

◆ **当 FIFO 半满信号有效时，批量读取 AD 数据**

函数原型：

Visual C++:

```
BOOL ReadDeviceProAD_Half(HANDLE hDevice,
                          ULONG ADBuffer[],
                          LONG nReadSizeWords,
                          PLONG nRetSizeWords)
```

LabVIEW:

请参考相关演示程序。

功能：一旦用户使用GetDevStatusProAD后取得的FIFO状态bHalf等于TRUE(即半满状态有效)时，应立即用此函数读取设备上FIFO中的半满AD数据。

参数：

hDevice 设备对象句柄，它应由CreateDevice或CreateDeviceEx创建。

ADBuffer 接受AD数据的用户缓冲区，通常可以是一个用户定义的数组。关于如何将这些AD数据转换成相应的电压值，请参考《数据格式转换与排列规则》。

nReadSizeWords 指定一次ReadDeviceProAD_Half操作应读取多少字数据到用户缓冲区。注意此参数的值不能大于用户缓冲区ADBuffer的最大空间，而且应等于FIFO总容量的二分之一(如果用户有特殊需要可以小于

FIFO的二分之一长)。比如设备上配置了 1K FIFO，即 1024 字，那么这个参数应指定为 512 或小于 512。

返回值：如果成功的读取由nReadSizeWords参数指定量的AD数据到用户缓冲区，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。

其使用方法请参考本部分第十章 《高速大容量、连续不间断数据采集及存盘技术详解》。

相关函数： CreateDevice GetDevStatusProAD StartDeviceProAD ReadDeviceProAD_Npt ReadDeviceProAD_Half StopDeviceProAD ReleaseDevice

◆ **暂停 AD 设备**

函数原型：

Visual C++:

BOOL StopDeviceProAD (HANDLE hDevice)

LabVIEW

请参考相关演示程序。

功能：暂停AD设备。它必须在调用StartDeviceProAD后才能调用此函数。该函数除了停止AD设备不再转换以外，不改变设备的其他任何状态。此后您可再调用StartDeviceProAD函数重新启动AD，此时AD会按照暂停以前的状态（如FIFO存储器位置、通道位置）开始转换。

参数：hDevice 设备对象句柄，它应由CreateDevice或CreateDeviceEx创建。

返回值：如果调用成功，则返回TRUE，且AD立刻停止转换，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 GetDevStatusProAD StartDeviceProAD ReadDeviceProAD_Npt ReadDeviceProAD_Half StopDeviceProAD ReleaseDevice

第四节 继电器状态函数

◆ **取得继电器状态**

函数原型：

Visual C++:

BOOL SetDeviceDO (HANDLE hDevice，BYTE bDOSts[32])

LabVIEW

请参考相关演示程序。

功能：取得继电器状态。

参数：hDevice 设备对象句柄，它应由CreateDevice或CreateDeviceEx创建。

返回值：如果调用成功，则返回 TRUE。

相关函数： CreateDevice ReleaseDevice SetDeviceDO

第四章 硬件参数结构

第一节、AD 采样的实际硬件参数介绍（CPCI2327_ PARA_MODE）

Visual C++:

```
typedef struct _CPCI2327_STATUS_AD
{
    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 存储器的溢出标志，= TRUE 已发生溢出， = FALSE 未发
```

生溢出

} CPCI2327_STATUS_AD, *PCPCI2327_STATUS_AD;

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

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 CPCI2327.h 和 Sys.cpp)。

[程序]→**[阿尔泰测控演示系统]**→ **[CPCI2327 32 路继电器卡]** → **[Microsoft Visual C++]**→ **[**简易代码演示**]**→ **[** Squre 演示源程序**]**

其默认存放路径为：系统盘\ART\CPCI2327\SAMPLES\VC\SIMPLE\RELAY

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 CPCI2327.h 和 Sys.cpp)。

[程序]→**[阿尔泰测控演示系统]**→ **[CPCI2327 32 路继电器卡]** → **[Microsoft Visual C++]**→ **[**高级代码演示**]**

其默认存放路径为：系统盘\ART\CPCI2327\SAMPLES\VC\SIMPLE\ RELAY

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“CPCI2327_”）

函数名	函数功能	备注
① CPCI 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
ReleaseSystemEvent	释放系统内核事件对象	

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

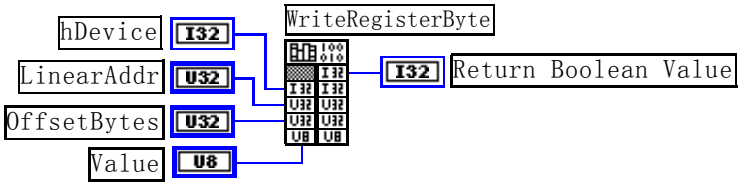
- **以单字节（即 8 位）方式写 CPCI 内存映射寄存器的某个单元**

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)
    :
```

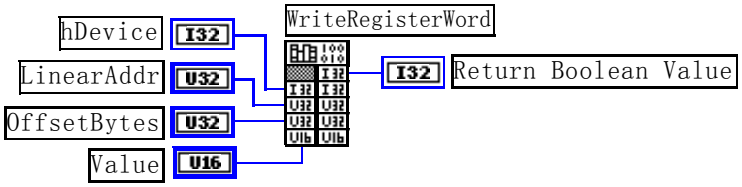
◆ **以双字节（即 16 位）方式写 CPCI 内存映射寄存器的某个单元**

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)

：

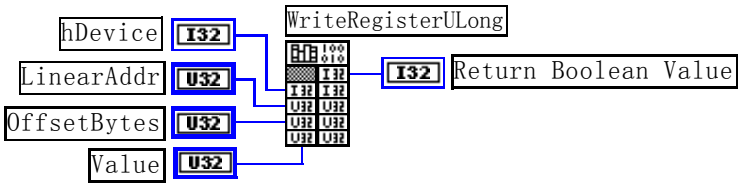
◆ **以四字节（即 32 位）方式写 CPCI 内存映射寄存器的某个单元**

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**

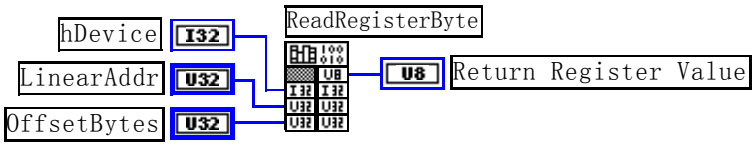
函数原型:

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
:
```

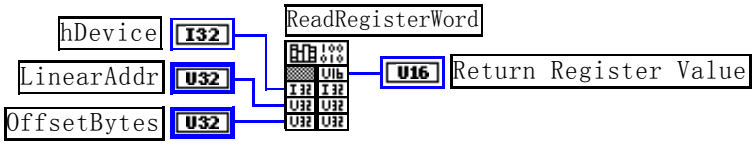
- **以双字节（即 16 位）方式读 CPCI 内存映射寄存器的某个单元**

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
    :
```

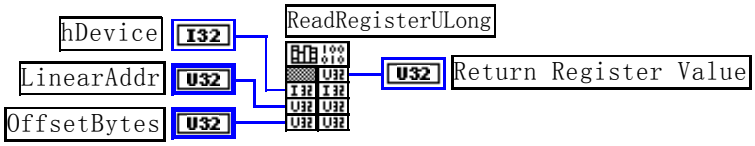
◆ **以四字节（即 32 位）方式读 CPCI 内存映射寄存器的某个单元**

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
    :
```

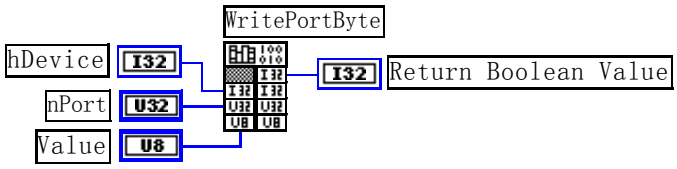
第三节、IO 端口读写函数原型说明

- **以单字节(8Bit)方式写 I/O 端口**

Visual C++:

```
BOOL WritePortByte (HANDLE hDevice,
                    __int64 pbPort,
                    ULONG offserBytes,
                    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。

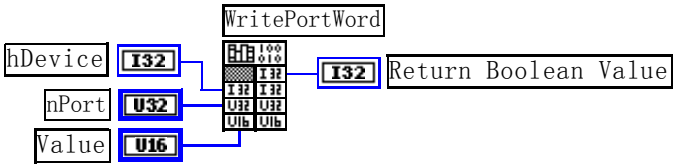
相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

- **以双字(16Bit)方式写 I/O 端口**

Visual C++:

```
BOOL WritePortWord (HANDLE hDevice,
                    __int64 pbPort，
                    ULONG offserBytes,
                    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

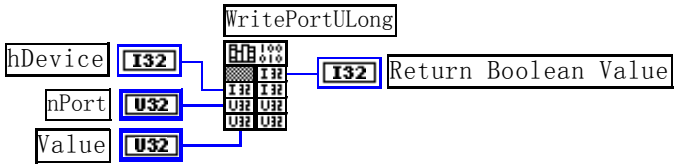
- **以四字节(32Bit)方式写 I/O 端口**

Visual C++:

```
BOOL WritePortULong(HANDLE hDevice,
```

__int64 pbPort,
ULONG offserBytes,
ULONG Value)

LabVIEW:



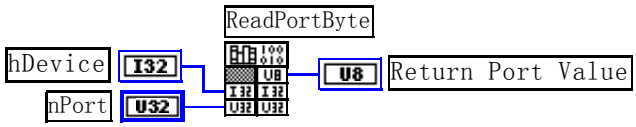
功能：以四字节(32Bit)方式写 I/O 端口。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
pbPort 设备的 I/O 端口号。
Value 写入由 pbPort 指定端口的值。
返回值：若成功，返回TRUE，否则返回FALSE，用户可用GetLastErrorEx捕获当前错误码。
相关函数： CreateDevice WritePortByte WritePortWord
WritePortULong ReadPortByte ReadPortWord

- **以单字节(8Bit)方式读 I/O 端口**

Visual C++:

BYTE ReadPortByte(HANDLE hDevice,
__int64 pbPort,
ULONG offserBytes)

LabVIEW:



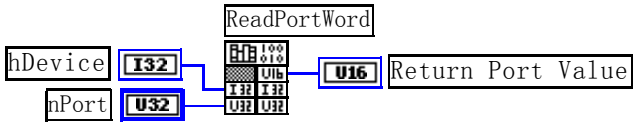
功能：以单字节(8Bit)方式读 I/O 端口。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
pbPort 设备的 I/O 端口号。
返回值：返回由 pbPort 指定的端口的值。
相关函数： CreateDevice WritePortByte WritePortWord
WritePortULong ReadPortByte ReadPortWord

- **以双字节(16Bit)方式读 I/O 端口**

Visual C++:

WORD ReadPortWord(HANDLE hDevice,
__int64 pbPort
ULONG offserBytes)

LabVIEW:



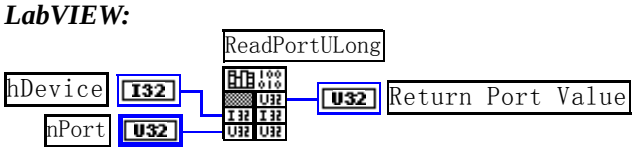
功能：以双字节(16Bit)方式读 I/O 端口。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
pbPort 设备的 I/O 端口号。
返回值：返回由 pbPort 指定的端口的值。
相关函数： CreateDevice WritePortByte WritePortWord
WritePortULong ReadPortByte ReadPortWord

- **以四字节(32Bit)方式读 I/O 端口**

Visual C++:

ULONG ReadPortULong(HANDLE hDevice,
UINT pbPort
ULONG offserBytes)

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。

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）

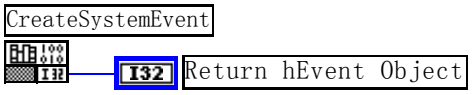
◆ **创建内核系统事件**

函数原型：

Visual C++:

HANDLE CreateSystemEvent(void)

LabVIEW:



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