PXI2390 数据采集卡

WIN2000/XP 驱动程序使用说明书

阿尔泰科技发展有限公司
产品研发部修订

请您务必阅读《使用纲要》，他会使您事半功倍!

目　录

第一章 版权信息与命名约定

第一节、版权信息

第二节、命名约定

一、为简化文字内容，突出重点，本文中提到的函数名通常为基本功能名部分，其前缀设备名如 PXIxxxx_ 则被省略。如 PXI2390_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			

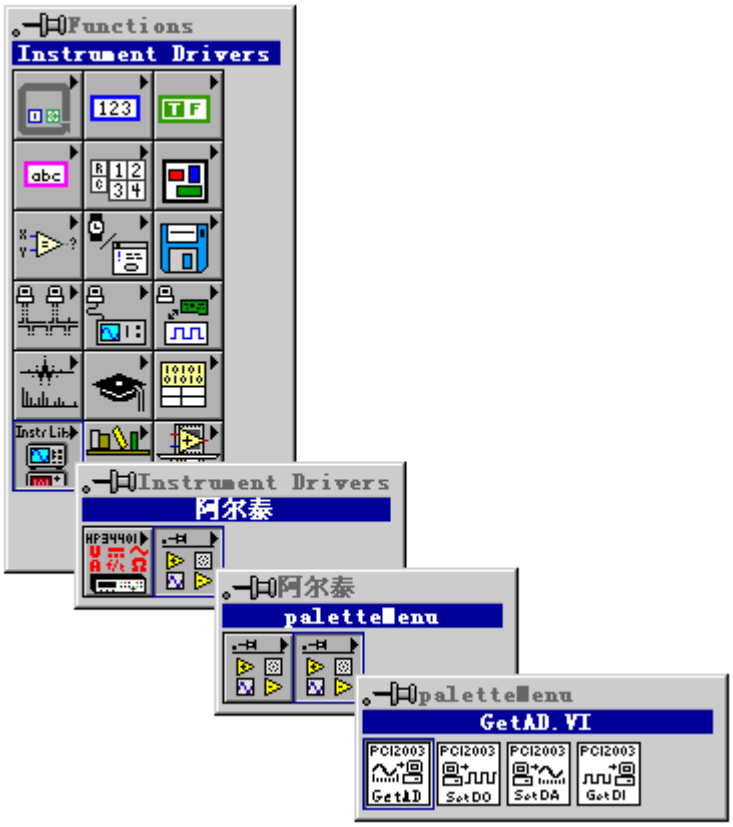
以上规则不局限于该产品。

第二章 PXI 即插即用设备操作函数接口介绍

由于我公司的设备应用于各种不同的领域，有些用户可能根本不关心硬件设备的控制细节，只关心 AD 的首末通道、采样频率等，然后就能通过一两个简易的采集函数便能轻松得到所需要的 AD 数据。这方面的用户我们称之为上层用户。那么还有一部分用户不仅对硬件控制熟悉，而且由于应用对象的特殊要求，则要直接控制设备的每一个端口，这是一种复杂的工作，但又是必须的工作，我们则把这一群用户称之为底层用户。因此总的看来，上层用户要求简单、快捷，他们最希望在软件操作上所要面对的全是他们最关心的问题，比如在正式采集数据之前，只须用户调用一个简易的初始化函数告诉设备我要使用多少个通道，采样频率是多少赫兹等，然后便可以用函数指定每次采集的点数，即可实现数据连续不间断采样。而关于设备的物理地址、端口分配及功能定义等复杂的硬件信息则与上层用户无任何关系。那么对于底层用户则不然。他们不仅要关心设备的物理地址，还要关心虚拟地址、端口寄存器的功能分配，甚至每个端口的 Bit 位都要了如指掌，看起来这是一项相当复杂、繁琐的工作。但是这些底层用户一旦使用我们提供的技术支持，则不仅可以让您不必熟悉 PXI 总线复杂的控制协议，同是还可以省掉您许多繁琐的工作，比如您不用去了解 PXI 的资源配置空间、PNP 即插即用管理，而只须用GetDeviceAddr函数便可以同时取得指定设备的物理基地址和虚拟线性基地址。这个时候您便可以用这个虚拟线性基地址，再根据硬件使用说明书中的各端口寄存器的功能说明，然后使用ReadRegisterULong和WriteRegisterULong对这些端口寄存器进行 32 位模式的读写操作，即可实现设备的所有控制。

综上所述，用户使用我公司提供的驱动程序软件包将极大的方便和满足您的各种需求。但为了您更省心，别忘了在您正式阅读下面的函数说明时，先明白自己是上层用户还是底层用户，因为在《设备驱动接口函数总列表》中的备注栏里明确注明了适用对象。

另外需要申明的是，在本章和下一章中列明的关于 LabView 的接口，均属于外挂式驱动接口，他是通过 LabView 的 Call Labrary Function 功能模板实现的。它的特点是除了自身的语法略有不同以外，每一个基于 LabView 的驱动图标与 Visual C++、Visual Basic、Delphi 等语言中每个驱动函数是一一对应的，其调用流程和功能是完全相同的。那么相对于外挂式驱动接口的另一种方式是内嵌式驱动。这种驱动是完全作为 LabView 编程环境中的紧密耦合的一部分，它可以直接从 LabView 的 Functions 模板中取得，如下图所示。此种方式更适合上层用户的需要，它的最大特点是方便、快捷、简单，而且可以取得它的在线帮助。关于 LabView 的外挂式驱动和内嵌式驱动更详细的叙述，请参考 LabView 的相关演示。



第一节、设备驱动接口函数总列表（每个函数省略了前缀“PXI2390_”）

函数名	函数功能	备注
①设备对象操作函数		
CreateDevice	创建 PXI 设备对象(用设备逻辑号)	上层及底层用户
GetDeviceCount	取得同一种 PXI 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 PXI 设备的各种配置	上层及底层用户
ReleaseDevice	关闭设备，且释放 PXI 总线设备对象	上层及底层用户
②通用函数		
SelectFunction	功能选择函数	上层用户
SetPulseOutputFre	设置脉冲输出频率	上层用户
③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
④CNT 计数定时器操作函数		
StartDeviceCNT	启动设备计数	上层用户
SetDeviceCNT	设置计数器的初值	上层用户
GetDeviceCNT	取得各路计数器的当前计数值	上层用户
StopDeviceCNT	停止设备计数	上层用户
⑤开关量简易操作函数		
GetDeviceDI	开关输入函数	上层用户
SetDeviceDO	开关输出函数	上层用户
RetDeviceDO	回读开关量状态	上层用户

使用需知：

Visual C++：

要使用如下函数关键的问题是：

首先，必须在您的源程序中包含如下语句：

#include “C:\Art\PXI2390\INCLUDE\PXI2390.H”

注：以上语句采用默认路径和默认板号，应根据您的板号和安装情况确定 PXI2390.H 文件的正确路径，当然也可以把此文件拷到您的源程序目录中。

另外，要在 VB 环境中用子线程以实现高速、连续数据采集与存盘，请务必使用 VB5.0 版本。当然如果您有 VB6.0 的最新版，也可以实现子线程操作。

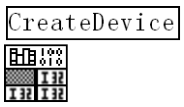
Visual Basic:

要使用如下函数一个关键的问题是首先必须将我们提供的模块文件(*.Bas)加入到您的 VB 工程中。其方法是选择 VB 编程环境中的工程(Project)菜单，执行其中的"添加模块"(Add Module)命令，在弹出的对话中选择 PXI2390.Bas 模块文件，该文件的路径为用户安装驱动程序后其子目录 Samples\VB 下面。

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

LabVIEW/CVI :

LabVIEW 是美国国家仪器公司(National Instrument)推出的一种基于图形开发、调试和运行程序的集成化环境，是目前国际上唯一的编译型的图形化编程语言。在以 PC 机为基础的测量和工控软件中，LabVIEW 的市场普及率仅次于 C++/C 语言。LabVIEW 开发环境具有一系列优点，从其流程图式的编程、不需预先编译就存在的语法检查、调试过程使用的数据探针，到其丰富的函数功能、数值分析、信号处理和设备驱动等功能，都令人称道。关于 LabView/CVI 的进一步介绍请见本文最后一部分关于 LabView 的专述。其驱动程序接口单元模块的使用方法如下：



一、在 LabView 中打开 PXI2390.VI 文件，用鼠标单击接口单元图标，比如 CreateDevice 图标 然后按 Ctrl+C 或选择 LabView 菜单 Edit 中的 Copy 命令，接着进入用户的应用程序 LabView 中，按 Ctrl+V 或选择 LabView 菜单 Edit 中的 Paste 命令，即可将接口单元加入到用户工程中，然后按以下函数原型说明或演示程序的说明连接该接口模块即可顺利使用。

二、根据 LabView 语言本身的规定，接口单元图标以黑色的较粗的中间线为中心，以左边的方格为数据输入端，右边的方格为数据的输出端，如接口单元，设备对象句柄、用户分配的数据缓冲区、要求采集的数据长度等信息从接口单元左边输入端进入单元，待单元接口被执行后，需要返回给用户的数据从接口单元右边的输出端输出，其他接口完全同理。

三、在单元接口图标中，凡标有“I32”为有符号长整型 32 位数据类型，“U16”为无符号短整型 16 位数据类型，“［U16］”为无符号 16 位短整型数组或缓冲区或指针，“［U32］”与“［U16］”同理，只是位数不一样。

第二节、设备对象管理函数原型说明

♦ 创建设备对象函数（逻辑号）

函数原型：

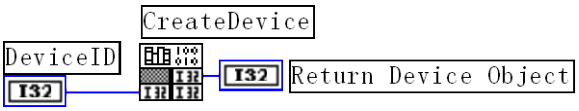
Visual C++：

HANDLE CreateDevice (int DeviceID = 0)

Visual Basic：

Declare Function CreateDevice Lib "PXI2390_32" (Optional ByVal DeviceLgcID As Long = 0) As Long

LabVIEW：



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

参数：

DeviceLgcID 逻辑设备 ID(Logic Device Identifier)标识号。当向同一个 Windows 系统中加入若干相同类型的 PXI 设备时，我们的驱动程序将以该设备的“基本名称”与 DeviceLgcID 标识值为后缀的标识符来确认和管理该设备。比如若用户往 Windows 系统中加入第一个 PXI2390 模板时，驱动程序逻辑号为“0”来确认和管理第一个设备，若用户接着再添加第二个 PXI2390 模板时，则系统将以逻辑号“1”来确认和管理第二个设备，若再添加，则以此类推。所以当用户要创建设备句柄管理和操作第一个 PXI 设备时，DeviceLgcID 应置 0，第二个应置 1，也以此类推。但默认值为 0。该参数之所以称为逻辑设备号，是因为每个设备的逻辑号是不能事先由用户硬性确定的，而是由 BIOS 和操作系统加载设备时，依据主板总线编号等信息进行这个设备 ID 号分配，说得简单点，就是加载设备的顺序编号，编号的递增顺序为 0、1、2、3……。所以用户无法直接固定某一个设备的在设备列表中的物理位置，若想固定，则必须须使用物理 ID 号，调用CreateDeviceEx函数实现。

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

相关函数： CreateDevice　GetDeviceCount　GetDeviceCurrentID　ListDeviceDlg　ReleaseDevice

Visual C++程序举例:

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

Visual Basic 程序举例:

```
    :
```

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

♦ **取得本计算机系统中 PXI2390 设备的总数量**

函数原型：

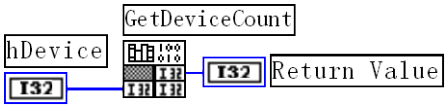
Visual C++:

int GetDeviceCount (HANDLE hDevice)

Visual Basic:

Declare Function GetDeviceCount Lib "PXI2390_32" (ByVal hDevice As Long) As Long

LabVIEW：



功能：取得 PXI2390 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 PXI2390 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 DeviceLgcID，
                         PLONG DevicePhysID)
```

Visual Basic:

```
Declare Function GetDeviceCurrentID Lib "PXI2390_32" (_
                                        ByVal hDevice As Long,_
                                        ByRef DeviceLgcID As Long,_
                                        ByRef DevicePhysID As Long) As Boolean
```

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 GetLastError 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 PXI2390 设备各种配置信息**

函数原型：

Visual C++:

BOOL ListDeviceDlg (HANDLE hDevice)

Visual Basic:

Declare Function ListDeviceDlg Lib "PXI2390_32" (ByVal hDevice As Long) As Boolean

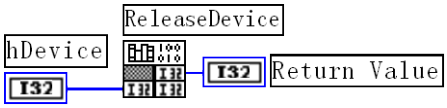
LabVIEW：

请参考相关演示程序。

功能：列表系统中 PXI2390 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 PXI2390 设备的配置情况。
相关函数：CreateDevice ReleaseDevice

♦ **释放设备对象所占的系统资源及设备对象**

函数原型：
Visual C++:
BOOL ReleaseDevice(HANDLE hDevice)
Visual Basic:
Declare Function ReleaseDevice Lib "PXI2390_32" (ByVal hDevice As Long) As Boolean
LabVIEW:



功能：释放设备对象所占用的系统资源及设备对象自身。
参数：hDevice 设备对象句柄，它应由CreateDevice创建。
返回值：若成功，则返回 TRUE， 否则返回 FALSE
相关函数：CreateDevice

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

第三节、通用函数原型说明

♦**功能选择函数**

函数原型：
Visual C++:
```
BOOL SelectFunction (HANDLE hDevice,
                     LONG lFunction,
                     ULONG nChannel)
```
Visual Basic:
```
Declare Function SelectFunction Lib "PXI2390_32" ( _
                     ByVal hDevice As Long, _
                     ByVal lFunction As Long, _
                     ByVal nChannel As Long) As Boolean
```
LabVIEW:
请参考相关演示程序。

功能：功能选择函数。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
lFunction 功能选择参数(0 代表计数器，1 代表测频，2 代表脉冲输出)
nChannel 计数器通道选择。
返回值：若成功，返回 TRUE，否则返回 FALSE，您可以调用GetLastErrorEx 函数取得错误或错误字符信息。
相关函数：CreateDevice SetPulseOutputFre SelectFunction ReleaseDevice

♦ **设置脉冲输出频率**

函数原型：
Visual C++:
```
BOOL SetPulseOutputFre (HANDLE hDevice,
                        LONG lFrequency,
```

```
                                ULONG nChannel)
```

Visual Basic:

```
Declare Function SetPulseOutputFre Lib "PXI2390_32" ( _
                                        ByVal hDevice As Long, _
                                        ByVal lFrequency As Long, _
                                        ByVal nChannel As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：设置脉冲输出频率。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

lFrequency 输出频率(1Hz—1MHz)

nChannel 计数器通道选择。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数：CreateDevice SetPulseOutputFre SelectFunction ReleaseDevice

第四节、计数测频函数原型说明

♦设置设备测频模式

函数原型：

Visual C++:

```
BOOL SetMeasureFre (HANDLE hDevice,
                    ULONG ulDelayTime,
                    LONG lChannel)
```

Visual Basic:

```
Declare Function SetMeasureFre Lib "PXI2390_32" ( _
                                    ByVal hDevice As Long, _
                                    ByVal ulDelayTime As Long, _
                                    ByVal lChannel As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：设置设备测频模式。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

ulDelayTime 确保测频准确度的定时高电平脉冲时间[10——1000ms]

nChannel 计数器通道选择。

返回值：若成功，返回 TRUE，否则返回 FALSE，您可以调用GetLastErrorEx 函数取得错误或错误字符信息。

相关函数：CreateDevice SetMeasureFre GetMeasureFreSts
GetMeasureFre ReleaseDevice

♦设置设备测频模式

函数原型：

Visual C++:

```
BOOL GetMeasureFreSts (HANDLE hDevice,
                       PULONG pCNTStatus,
                       LONG lChannel)
```

Visual Basic:

```
Declare Function GetMeasureFreSts Lib "PXI2390_32" ( _
                                        ByVal hDevice As Long, _
                                        ByRef pCNTStatus As Long, _
                                        ByVal lChannel As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：设置设备测频模式。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
pCNTStatus 测频状态：等于 0 表示测频结束，等于 1 表示正在测频计数。
nChannel 计数器通道选择。
返回值：若成功，返回 TRUE，否则返回 FALSE，您可以调用GetLastErrorEx 函数取得错误或错误字符信息。
相关函数： CreateDevice SetMeasureFre GetMeasureFreSts GetMeasureFre ReleaseDevice

♦获取被测频率值,返回被测频率值
函数原型：
Visual C++:
```
BOOL GetMeasureFre (HANDLE hDevice,
                    LONG lChannel)
```
Visual Basic:
```
Declare Function GetMeasureFre Lib "PXI2390_32" (_
                                        ByVal hDevice As Long,_
                                        ByVal lChannel As Long) As Long
```
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。
nChannel 计数器通道选择。
返回值：若成功，返回 TRUE，否则返回 FALSE，您可以调用GetLastErrorEx 函数取得错误或错误字符信息。
相关函数： CreateDevice SetMeasureFre GetMeasureFreSts GetMeasureFre ReleaseDevice

第五节、CNT 计数与定时器操作函数原型说明

♦ 启动设备计数
函数原型：
Visual C++:
```
BOOL StartDeviceCNT (HANDLE hDevice,
                     ULONG nChannel)
```
Visual Basic:
```
Declare Function StartDeviceCNT Lib "PXI2390_32" ( _
                                    ByVal hDevice As Long, _
                                    ByVal nChannel As Long) As Boolean
```
LabVIEW:
请参考相关演示程序。

功能：启动设备计数。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
nChannel 计数器通道选择。
返回值：若成功，返回 TRUE，否则返回 FALSE，您可以调用GetLastErrorEx 函数取得错误或错误字符信息。
相关函数： CreateDevice StartDeviceCNT SetDeviceCNT StopDeviceCNT ReleaseDevice

♦ 设置计数器的初值
函数原型：

Visual C++:

```
BOOL SetDeviceCNT ( HANDLE hDevice,
                    ULONG CNTVal,
                    ULONG ControlMode,
                    ULONG nChannel)
```

Visual Basic:

```
Declare Function SetDeviceCNT Lib "PXI2390_32" ( _
                    ByVal hDevice As Long, _
                    ByVal CNTVal As Long, _
                    ByVal ControlMode As Long, _
                    ByVal nChannel As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：设置计数器的初值。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

CNTVal 计数初值。

ControlMode 方式控制字。其选项值如下表：

常量名	常量值	功能定义
PXI2390_GATEMODE_POSITIVE_0	0x0000	计数结束产生中断
PXI2390_GATEMODE_RISING_1	0x0001	可编程单拍脉冲
PXI2390_GATEMODE_POSITIVE_2	0x0002	频率发生器
PXI2390_GATEMODE_POSITIVE_3	0x0003	方波发生器
PXI2390_GATEMODE_POSITIVE_4	0x0004	软件触发选通
PXI2390_GATEMODE_RISING_5	0x0005	硬件触发选通

nChannel 计数器通道选择。

返回值：若成功，返回 TRUE，否则返回 FALSE，您可以调用GetLastErrorEx 函数取得错误或错误字符信息。

相关函数： CreateDevice　StartDeviceCNT　GetDeviceCNT　StopDeviceCNT　ReleaseDevice

♦ **取得各路计数器的当前计数值**

函数原型：

Visual C++:

```
BOOL GetDeviceCNT (HANDLE hDevice,
                    PULONG pCNTVal,
                    PULONG pControlMode,
                    ULONG nChannel)
```

Visual Basic:

```
Declare Function GetDeviceCNT Lib GetDeviceCNT Lib "PXI2390_32" ( _
                    ByVal hDevice As Long, _
                    ByRef pCNTVal As Long, _
                    ByRef pControlMode As Long, _
                    ByVal nChannel As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：取得各路计数器的当前计数值。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pCNTVal 取得计数初值。

pControlMode 取得方式控制字。

nChannel 计数器通道选择。

返回值：若成功，返回 TRUE，否则返回 FALSE，您可以调用GetLastErrorEx 函数取得错误或错误字符信

息。

相关函数： CreateDevice StartDeviceCNT SetDeviceCNT StopDeviceCNT ReleaseDevice

♦停止设备计数

函数原型：

Visual C++:

```
BOOL StopDeviceCNT ( HANDLE hDevice,
                     ULONG nChannel)
```

Visual Basic:

```
Declare Function StopDeviceCNT Lib "PXI2390_32" (_
                                    ByVal hDevice As Long,_
                                    ByVal nChannel As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：停止设备计数。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nChannel 计数器通道选择。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice StartDeviceCNT SetDeviceCNT StopDeviceCNT ReleaseDevice

第六节、DIO 数字量输入输出开关量操作函数原型说明

♦ 开关量输入

函数原型：

Visual C++:

```
BOOL GetDeviceDI ( HANDLE hDevice,
                   BYTE bDISts[8])
```

Visual Basic:

```
Declare Function GetDeviceDI Lib "PXI2390_32" ( _
                                    ByVal hDevice As Long, _
                                    ByRef bDISts As Byte) As Boolean
```

LabVIEW：

请参考相关演示程序。

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

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDISts 八路开关量输入状态的参数结构，共有 8 个元素，分别对应于 DI0-DI7 路开关量输入状态位。如果 bDISts[0]等于“1”则表示 0 通道处于开状态，若为“0”则 0 通道为关状态。其他同理。

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

相关函数： CreateDevice SetDeviceDO ReleaseDevice

♦ 开关量输出

函数原型：

Visual C++:

```
BOOL SetDeviceDO (HANDLE hDevice,
                  BYTE bDOSts[8])
```

Visual Basic:

```
Declare Function SetDeviceDO Lib "PXI2390_32" ( _
                                    ByVal hDevice As Long, _
                                    ByRef bDOSts As Byte) As Boolean
```

LabVIEW：

请参考相关演示程序。

功能：负责将 PXI 设备上的输出开关量置成由 bDOSts[x]指定的相应状态。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
bDOSts 八路开关量输出状态的参数结构，共有 8 个元素，分别对应于 DO0-DO7 路开关量输出状态位。比如置 DO0 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，必须对这个参数数组中的每个元素赋初值，其值必须为“1”或“0”。
返回值：若成功，返回 TRUE，否则返回 FALSE。
相关函数： CreateDevice GetDeviceDI ReleaseDevice

♦ **回读数字量输出状态**

函数原型：
Visual C++:

```
BOOL RetDeviceDO (HANDLE hDevice,
                  BYTE bDOSts[8])
```

Visual Basic:

```
Declare Function RetDeviceDO Lib "PXI2390_32" ( _
                                    ByVal hDevice As Long, _
                                    ByRef bDOSts As Byte) As Boolean
```

LabVIEW:
请参考相关演示程序。

功能：负责将 PXI 设备上的输出开关量置成由 bDOSts[x]指定的相应状态。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
bDOSts 获得开关输出状态(注意： 必须定义为 8 个字节元素的数组)。
返回值：若成功，返回 TRUE，否则返回 FALSE。
相关函数： CreateDevice GetDeviceDI ReleaseDevice

♦ **以上函数调用一般顺序**

① CreateDevice
② SetDeviceDO(或GetDeviceDI，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（数字 I/O 的输入输出及 AD 采样可以同时进行，互不影响）。

第三章 上层用户函数接口应用实例

第一节、怎样使用GetDeviceDI函数进行更便捷的数字开关量输入操作

Visual C++:
其详细应用实例及正确代码请参考 Visual C++测试与演示系统，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程。
[程序]\[阿尔泰测控演示系统]\[PXI2390 8 路计数器和 8 路开关量卡]\[Microsoft Visual C++]\[简易代码演示]\[DIO…]

第二节、怎样使用SetDeviceDO函数进行更便捷的数字开关量输出操作

Visual C++:
其详细应用实例及正确代码请参考 Visual C++测试与演示系统，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程。
[程序]\[阿尔泰测控演示系统]\[PXI2390 8 路计数器和 8 路开关量卡]\[Microsoft Visual C++]\[简易代码演示]\[DIO…]

第四章 共用函数介绍

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

第一节、公用接口函数总列表（每个函数省略了前缀“PXI2390_”）

函数名	函数功能	备注
① PXI 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	
WriteRegisterByte	以字节(8Bit)方式写寄存器端口	底层用户
WriteRegisterWord	以字(16Bit)方式写寄存器端口	底层用户
WriteRegisterULong	以双字(32Bit)方式写寄存器端口	底层用户
ReadRegisterByte	以字节(8Bit)方式读寄存器端口	底层用户
ReadRegisterWord	以字(16Bit)方式读寄存器端口	底层用户
ReadRegisterULong	以双字(32Bit)方式读寄存器端口	底层用户
② ISA 总线 I/O 端口操作函数		
WritePortByte	以字节(8Bit)方式写 I/O 端口	用户程序操作端口
WritePortWord	以字(16Bit)方式写 I/O 端口	用户程序操作端口
WritePortULong	以无符号双字(32Bit)方式写 I/O 端口	用户程序操作端口
ReadPortByte	以字节(8Bit)方式读 I/O 端口	用户程序操作端口
ReadPortWord	以字(16Bit)方式读 I/O 端口	用户程序操作端口
ReadPortULong	以无符号双字(32Bit)方式读 I/O 端口	用户程序操作端口
③ 创建 Visual Basic 子线程，线程数量可达 32 个以上		
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、PXI 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 pbPXIBar[6])
```

Visual Basic:

```
Declare Function GetDeviceBar Lib "PXI2390_32" ( _
                                  ByVal hDevice As Long, _
                                  ByRef pbPCIBar As Long) As Boolean
```

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。

pbPXIBar 返回 PXI BAR 所有地址。

相关函数： CreateDevice GetDevVersion WriteRegisterByte WriteRegisterWord WriteRegisterULong ReadRegisterByte ReadRegisterWord ReadRegisterULong 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)
```

Visual Basic:

```
Declare Function GetDevVersion Lib "PXI2390_32" ( _
                                        ByVal hDevice As Long, _
                                        ByRef pulFmwVersion As Long, _
                                        ByRef pulDriverVersion As Long) As Boolean
```

LabVIEW:

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。

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 GetLastError 捕获当前错误码，并加以分析。

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

Visual C++程序举例:

```
    :
HANDLE hDevice;
ULONG LinearAddr, PhysAddr;
hDevice = CreateDevice(0);
if(!GetDeviceAddr(hDevice, &LinearAddr, &PhysAddr, 0))
{
    AfxMessageBox("取得设备地址失败…");
}
    :
```

Visual Basic 程序举例:

```
    :
Dim hDevice As Long
Dim LinearAddr, PhysAddr    As Long
hDevice = CreateDevice(0)
if Not GetDeviceAddr(hDevice, LinearAddr,    PhysAddr, 0) then
    MsgBox "取得设备地址失败…"
End If
    :
```

♦ **以单字节（即 8 位）方式写 PXI 内存映射寄存器的某个单元**

函数原型：

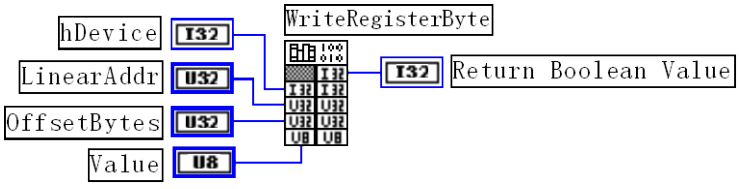
Visual C++:

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

Visual Basic:

```
Declare Function WriteRegisterByte Lib "PXI2390_32" ( _
                                        ByVal hDevice As Long, _
                                        ByVal pbLinearAddr As Long, _
                                        ByVal OffsetBytes As Long, _
                                        ByVal Value As Byte) As Boolean
```

LabVIEW:



功能：以单字节（即 8 位）方式写 PXI 内存映射寄存器。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

LinearAddr PXI 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 GetDevVersion GetDeviceBar 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 PXI 内存映射寄存器的某个单元**

函数原型：

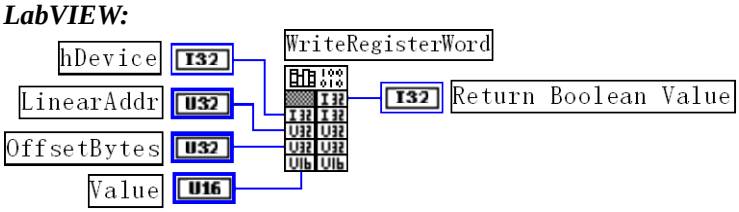
Visual C++:

```
BOOL WriteRegisterWord(HANDLE hDevice,
                       __int64 pbLinearAddr,
                       ULONG OffsetBytes,
                       WORD Value)
```

Visual Basic:

```
Declare Function WriteRegisterWord Lib "PXI2390_32" ( _
                                    ByVal hDevice As Long, _
                                    ByVal pbLinearAddr As Long, _
                                    ByVal OffsetBytes As Long, _
                                    ByVal Value As Integer) As Boolean
```

LabVIEW:



功能：以双字节（即 16 位）方式写 PXI 内存映射寄存器。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

LinearAddr　PXI 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　GetDevVersion　GetDeviceBar　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 PXI 内存映射寄存器的某个单元**

函数原型：

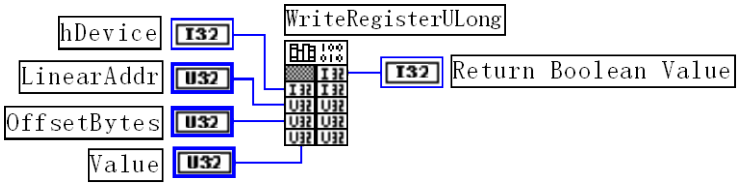
Visual C++:

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

Visual Basic:

```
Declare Function WriteRegisterULong Lib "PXI2390_32" ( _
                                    ByVal hDevice As Long, _
                                    ByVal pbLinearAddr As Long, _
                                    ByVal OffsetBytes As Long, _
                                    ByVal Value As Long) As Boolean
```

LabVIEW:



功能：以四字节（即 32 位）方式写 PXI 内存映射寄存器。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

LinearAddr PXI 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 GetDevVersion GetDeviceBar WriteRegisterByte WriteRegisterWord WriteRegisterULong ReadRegisterByte ReadRegisterWord ReadRegisterULong ReleaseDevice

Visual C++程序举例:

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

Visual Basic 程序举例:

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

♦ **以单字节（即 8 位）方式读 PXI 内存映射寄存器的某个单元**

函数原型：

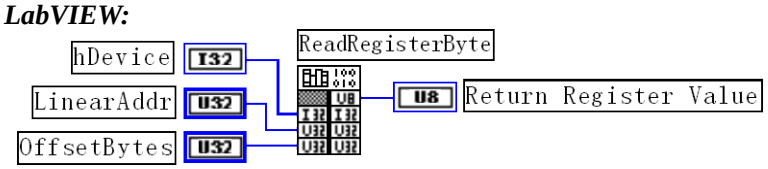
Visual C++:

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

Visual Basic:

```
Declare Function ReadRegisterByte Lib "PXI2390_32" ( _
                                    ByVal hDevice As Long, _
                                    ByVal pbLinearAddr As Long, _
                                    ByVal OffsetBytes As Long) As Byte
```

LabVIEW:



功能：以单字节（即 8 位）方式读 PXI 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 PXI 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 GetDevVersion GetDeviceBar 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 PXI 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 PXI 内存映射寄存器的某个单元**

函数原型：

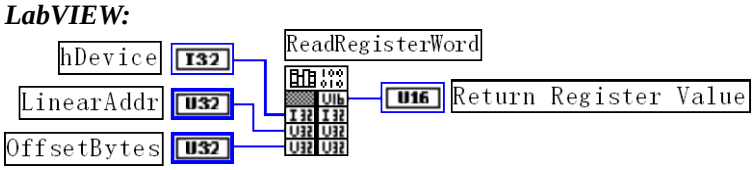
Visual C++:

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

Visual Basic:

```
Declare Function ReadRegisterWord Lib "PXI2390_32" ( _
                                    ByVal hDevice As Long, _
                                    ByVal pbLinearAddr As Long, _
                                    ByVal OffsetBytes As Long) As Integer
```

LabVIEW:



功能：以双字节（即 16 位）方式读 PXI 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 PXI 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 GetDevVersion GetDeviceBar
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 PXI 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 PXI 内存映射寄存器的某个单元**

函数原型：

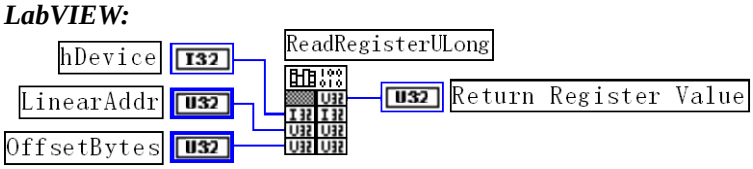
Visual C++:

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

Visual Basic:

```
Declare Function ReadRegisterULong Lib "PXI2390_32" ( _
                                    ByVal hDevice As Long, _
                                    ByVal pbLinearAddr As Long, _
                                    ByVal OffsetBytes As Long) As Long
```

LabVIEW:



功能：以四字节（即 32 位）方式读 PXI 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 PXI 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 GetDevVersion GetDeviceBar 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 PXI 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、I0 端口读写函数原型说明

- **以单字节(8Bit)方式写 I/O 端口**

函数原型：

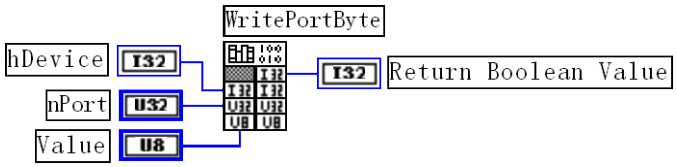
Visual C++:

```
BOOL WritePortByte (HANDLE hDevice,
                    __int64 pPort,
                    BYTE Value)
```

Visual Basic:

```
Declare Function WritePortByte Lib "PXI2390_32" ( _
                                ByVal hDevice As Long, _
                                ByVal pPort As Long, _
                                ByVal Value As Byte) As Boolean
```

LabVIEW:



功能：以单字节(8Bit)方式写 I/O 端口。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nPort 设备的 I/O 端口号。

Value 写入由 nPort 指定端口的值。

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

相关函数： CreateDevice　WritePortByte　WritePortWord　WritePortULong　ReadPortByte　ReadPortWord

- **以双字(16Bit)方式写 I/O 端口**

函数原型：

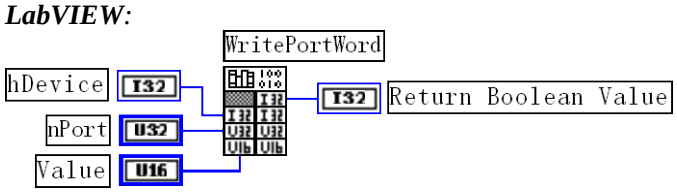
Visual C++:

```
BOOL WritePortWord (HANDLE hDevice,
                    __int64 pPort,
                    WORD Value)
```

Visual Basic:

```
Declare Function WritePortWord Lib "PXI2390_32" (_
                                ByVal hDevice As Long,_
                                ByVal pPort As Long,_
                                ByVal Value As Integer) As Boolean
```

LabVIEW:



功能：以双字(16Bit)方式写 I/O 端口。

参数：

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

♦ **以四字节(32Bit)方式写 I/O 端口**

函数原型：

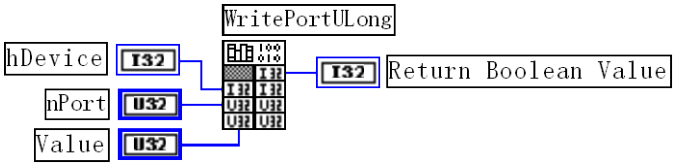
Visual C++:

```
BOOL WritePortULong(HANDLE hDevice,
                    __int64 pPort,
                    ULONG Value)
```

Visual Basic:

```
Declare Function WritePortULong Lib "PXI2390_32" (_
                                        ByVal hDevice As Long,_
                                        ByVal pPort As Long,_
                                        ByVal Value As Long) As Boolean
```

LabVIEW:



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

♦ **以单字节(8Bit)方式读 I/O 端口**

函数原型：

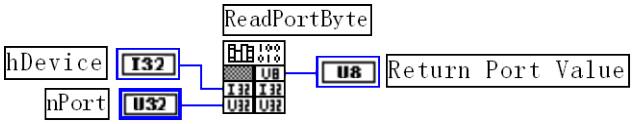
Visual C++:

```
BYTE ReadPortByte( HANDLE hDevice,
                   __int64 pPort)
```

Visual Basic:

```
Declare Function ReadPortByte Lib "PXI2390_32" (_
                                        ByVal hDevice As Long,_
                                        ByVal pPort As Long) As Byte
```

LabVIEW:



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

♦ **以双字节(16Bit)方式读 I/O 端口**

函数原型：

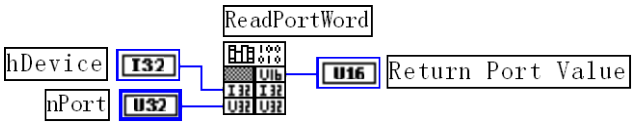
Visual C++:

```
WORD ReadPortWord(HANDLE hDevice,
                  __int64 pPort)
```

Visual Basic:

```
Declare Function ReadPortWord Lib "PXI2390_32" (_
                                    ByVal hDevice As Long,_
                                    ByVal pPort As Long) As Integer
```

LabVIEW:



功能：以双字节(16Bit)方式读 I/O 端口。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nPort　设备的 I/O 端口号。

返回值：返回由 nPort 指定的端口的值。

相关函数： CreateDevice　WritePortByte　WritePortWord　WritePortULong　ReadPortByte　ReadPortWord

♦ **以四字节(32Bit)方式读 I/O 端口**

函数原型：

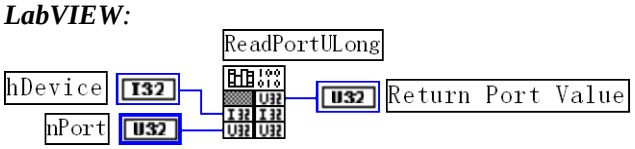
Visual C++:

```
ULONG ReadPortULong(HANDLE hDevice,
                    __int64 pPort)
```

Visual Basic:

```
Declare Function ReadPortULong Lib "PXI2390_32" (_
                                    ByVal hDevice As Long,_
                                    ByVal pPort As Long) As Long
```

LabVIEW:



功能：以四字节(32Bit)方式读 I/O 端口。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nPort　设备的 I/O 端口号。

返回值：返回由 nPort 指定端口的值。

相关函数： CreateDevice　WritePortByte　WritePortWord　WritePortULong　ReadPortByte　ReadPortWord

第四节、线程操作函数原型说明

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

♦ **创建内核系统事件**

函数原型：

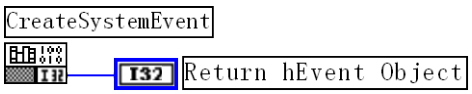
Visual C++:

```
HANDLE CreateSystemEvent(void)
```

Visual Basic:

```
Declare Function CreateSystemEvent Lib "PXI2390_32" (ByVal void) As Long
```

LabVIEW:



功能：创建系统内核事件对象，它将被用于中断事件响应或数据采集线程同步事件。
参数：无任何参数。
返回值：若成功，返回系统内核事件对象句柄，否则返回－1(或 INVALID_HANDLE_VALUE)。

♦ **释放内核系统事件**

函数原型：
Visual C++:
BOOL ReleaseSystemEvent(HANDLE hEvent)
Visual Basic:
Declare Function ReleaseSystemEvent Lib "PXI2390_32" (ByVal hEvent As Long) As Boolean
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。