PXI1117 数据采集卡

WIN2000/XP 驱动程序使用说明书

北京阿尔泰科技发展有限公司
产品研发部修订

请您务必阅读《使用纲要》，他会使您事半功倍!

目　录

第一章 版权信息与命名约定

第一节、版权信息

第二节、命名约定

一、为简化文字内容，突出重点，本文中提到的函数名通常为基本功能名部分，其前缀设备名如 PXIxxxx_ 则被省略。如 PXI1117_CreateDevice 则写为 CreateDevice。

二、函数名及参数中各种关键字缩写规则

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

以上规则不局限于该产品。

第二章 使用纲要

第一节、使用上层用户函数，高效、简单

如果您只关心通道及频率等基本参数，而不必了解复杂的硬件知识和控制细节，那么我们强烈建议您使用上层用户函数，它们就是几个简单的形如Win32 API的函数，具有相当的灵活性、可靠性和高效性。诸如InitDeviceDA、WriteDeviceBulkDA等。而底层用户函数如WriteRegisterULong、ReadRegisterULong、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，您基本上可以不InitDeviceDA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。因为上层函数的命名、参数的命名极其规范。

第二节、如何管理 PXI 设备

由于我们的驱动程序采用面向对象编程，所以要使用设备的一切功能，则必须首先用CreateDevice函数创建一个设备对象句柄hDevice，有了这个句柄，您就拥有了对该设备的绝对控制权。然后将此句柄作为参数传递给其他函数，如InitDeviceDA可以使用hDevice句柄以程序查询方式初始化设备的DA部件，WriteDeviceBulkDA函数可以用hDevice句柄实现对DA数据的采样读取。最后可以通过ReleaseDevice将hDevice释放掉。

第三节、如何实现 DA 波形数据输出

当您有了hDevice设备对象句柄后，便可用InitDeviceDA函数初始化DA部件，关于频率等参数InitDeviceDA

的设置是由这个函数的pDAPara参数结构体决定的。您只需要对这个pDAPara参数结构体的各个成员简单赋值即可实现所有硬件参数和设备状态的初始化。然后调用WriteDeviceBulkDA将准备好的DA数据写入板载RAM中，接着用EnableDeviceDA即可启动DA部件，开始DA输出，用户可以根据其状态作出相应的处理。当您需要暂停设备时，执行DisableDeviceDA，当您需要关闭DA设备时，ReleaseDeviceDA便可帮您实现（但设备对象hDevice依然存在）。

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，WriteRegisterByte，WriteRegisterWord，WriteRegisterULong，ReadRegisterByte，ReadRegisterWord，ReadRegisterULong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PXI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 PXI 设备操作函数接口介绍

第一节、设备驱动接口函数总列表（每个函数省略了前缀“PXI1117_”）

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 DA 数据采样操作函数		
SetDevTrigLevelDA	设置 DA 的触发电平	上层用户
SetDevFrequencyDA	可动态改变 DA 采样频率	上层用户
InitDeviceDA	初始化 PXI 设备上的 DA 部件准备传输	上层用户
WriteDeviceBulkDA	批量方式将用户缓冲区中的 DA 数据传输至板载 RAM 中	上层用户
ReadDeviceBulkDA	以批量方式将板载 RAM 中的 DA 数据回读至主机的用户缓冲区	上层用户
EnableDeviceDA	启动 DA 设备，开始转换	上层用户
DisableDeviceDA	暂停 DA 设备	上层用户
EnableDeviceTwoDA	同时启动 2 路 DA 设备	上层用户
DisableDeviceTwoDA	同时暂停 2 路 DA 设备	上层用户
ReleaseDeviceDA	释放 DA 设备	上层用户
③ DA 硬件参数系统保存、读取函数		
LoadParaDA	从 Windows 系统中读入硬件参数	上层用户
SaveParaDA	往 Windows 系统写入设备硬件参数	上层用户
ResetParaDA	将硬件参数结构体值复位为出厂默认值	上层用户

使用需知：

Visual C++ & C++Builder：

要使用如下函数关键的问题是：

首先，必须在您的源程序中包含如下语句：

#include “C:\Art\PXI1117\INCLUDE\PXI1117.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 PXI1117.H 文件的正确路径，当然也可以把此文件拷到您的源程序目录中。

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

C++ Builder:

要使用如下函数一个关键的问题是首先必须将我们提供的头文件(PXI1117.H)写进您的源程序头部。如：#include “\Art\PXI1117\Include\PXI1117.h”，然后再将 PXI1117.Lib 库文件分别加入到您的 C++ Builder 工程中。其具体办法是选择 C++ Builder 集成开发环境中的工程(Project)菜单中的“添加”（Add to Project）命令，在弹出的对话框中分别选择文件类型：Library file (*.lib) ，即可选择 PXI1117.Lib 文件。该文件的路径为用户安装驱动程序后其子目录 Samples\C_Builder 下。

Visual Basic:

要使用如下函数一个关键的问题是首先必须将我们提供的模块文件(*.Bas)加入到您的 VB 工程中。其方法是选择 VB 编程环境中的工程(Project)菜单，执行其中的"添加模块"(Add Module)命令，在弹出的对话中选择 PXI1117.Bas 模块文件，该文件的路径为用户安装驱动程序后其子目录 Samples\VB 下面。

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

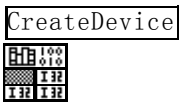
Delphi:

要使用如下函数一个关键的问题是首先必须将我们提供的单元模块文件（*.Pas）加入到您的 Delphi 工程中。其方法是选择 Delphi 编程环境中的 View 菜单，执行其中的"Project Manager"命令，在弹出的对话中选择*.exe 项目，再单击鼠标右键，最后 Add 指令，即可将 PXI1117.Pas 单元模块文件加入到工程中。或者在 Delphi 的编程环境中的 Project 菜单中，执行 Add To Project 命令，然后选择*.Pas 文件类型也能实现单元模块文件的添加。该文件的路径为用户安装驱动程序后其子目录 Samples\Delphi 下面。最后请在使用驱动程序接口的源程序文件中的头部的 Uses 关键字后面的项目中加入：“PXI1117”。如：

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  PXI1117;  // 注意： 在此加入驱动程序接口单元 PXI1117
```

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 PXI1117.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，设备对象句柄、用户分配的数据缓冲区、要求采集的数据长度等信息从接口单元左边输入端进入单元，待单元接口被执行后，需要返回给用户的数据从接口单元右边的输出端输出，其他接口完全同理。

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

第二节、设备对象管理函数原型说明

◆ **创建设备对象函数（逻辑号）**

函数原型：

Visual C++ & C++Builder：

HANDLE CreateDevice (int DeviceLgcID = 0)

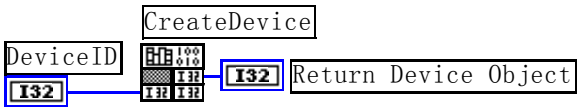
Visual Basic：

Declare Function CreateDevice Lib "PXI1117" (Optional ByVal DeviceLgcID As Integer = 0) As Long

Delphi：

Function CreateDevice(DeviceLgcID : Integer = 0) : Integer;
StdCall; External 'PXI1117' Name ' CreateDevice ';

LabVIEW：



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

参数：

DeviceLgcID 逻辑设备ID(Logic Device Identifier)标识号。当向同一个Windows系统中加入若干相同类型的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PXI1117 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PXI1117 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，调用函数实现。

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

相关函数： CreateDevice　GetDeviceCount　GetDeviceCurrentID　ListDeviceDlg　ReleaseDevice

Visual C++ & C++Builder 程序举例

```
    :
HANDLE hDevice;     // 定义设备对象句柄
int DeviceLgcID = 0;
hDevice = PXI1117_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 = PXI1117_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 PXI1117 设备的总数量**

函数原型：

Visual C++ & C++Builder:

int GetDeviceCount (HANDLE hDevice)

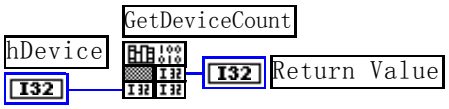
Visual Basic:

Declare Function GetDeviceCount Lib "PXI1117" (ByVal hDevice As Long) As Integer

Delphi：

```
Function GetDeviceCount (hDevice : Integer) : Integer;
                    StdCall; External 'PXI1117' Name ' GetDeviceCount ';
```

LabVIEW：



功能：取得 PXI1117 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 PXI1117 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 "PXI1117" (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 'PXI1117' 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 PXI1117 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 "PXI1117" (ByVal hDevice As Long ) As Boolean
```

Delphi：

```
Function ListDeviceDlg (hDevice : Integer) : Boolean;
                        StdCall; External 'PXI1117' Name ' ListDeviceDlg ';
```

LabVIEW：

请参考相关演示程序。

功能：列表系统中 PXI1117 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 PXI1117 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)
```

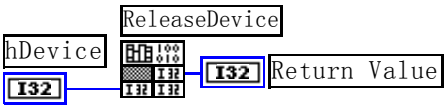
Visual Basic:

```
Declare Function ReleaseDevice Lib "PXI1117" (ByVal hDevice As Long ) As Boolean
```

Delphi：

```
Function ReleaseDevice(hDevice : Integer) : Boolean;
                        StdCall; External 'PXI1117' Name ' ReleaseDevice ';
```

LabVIEW：



功能：释放设备对象所占用的系统资源及设备对象自身。

参数：hDevice 设备对象句柄，它应由CreateDevice创建。

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

相关函数：CreateDevice

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

第三节、DA 数据采样操作函数原型说明

- **设置触发电平**

函数原型：

Visual C++ & C++Builder:

```
BOOL SetDevTrigLevelDA ( HANDLE hDevice,
                         float fTrigLevelVolt)
```

Visual Basic:

```
Declare Function SetDevTrigLevelDA Lib "PXI1117" (ByVal hDevice As Long, _
                                                  ByVal fTrigLevelVolt As Single ) As Boolean
```

Delphi:

```
Function SetDevTrigLevelDA (hDevice : Integer;
                            fTrigLevelVolt: Single) : Boolean;
                            StdCall; External 'PXI1117' Name ' SetDevTrigLevelDA';
```

LabView：

请参考相关演示程序。

功能：设置触发电平，该触发电平对所有 DA 通道同时有效。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nTrigLevelVolt 触发电平值，单位为 mV，其取值范围为[0, +10000.0]，注意本设备的触发电平在其取值范围中可分为 256 个分辨率。也就是说触发电平的设置精度为 39.0625 毫伏(10000.0/256)。

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	SetDevTrigLevelDA	SetDevFrequencyDA
InitDeviceDA	WriteDeviceBulkDA	ReadDeviceBulkDA
EnableDeviceDA	DisableDeviceDA	EnableDeviceTwoDA
DisableDeviceTwoDA	ReleaseDeviceDA	ReleaseDevice

- **动态改变采样频率**

函数原型：

Visual C++ & C++Builder:

```
BOOL SetDevFrequencyDA (HANDLE hDevice,
                        LONG nFrequency，
                        int nDAChannel)
```

Visual Basic:

```
Declare Function SetDevFrequencyDA Lib "PXI1117" (ByVal hDevice as Long, _
                                                  ByVal nFrequency As Long,_
                                                  ByVal nDAChannel As Integer) As Boolean
```

Delphi:

```
Function SetDevFrequencyDA (hDevice : Integer;
                            nFrequency: LongInt;
                            nDAChannel : Integer):Boolean;
                            StdCall; External 'PXI1117' Name ' SetDevFrequencyDA ';
```

LabVIEW：

请参考演示源程序。

功能：在 DA 采样过程中，可动态改变采样频率。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nFrequency 采样频率，取值范围为[153Hz, 1MHz]，其单位为 Hz。

nDAChannel DA 通道号，取值范围为[0, 1]。

返回值：如果调用成功，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	SetDevTrigLevelDA	SetDevFrequencyDA
InitDeviceDA	WriteDeviceBulkDA	ReadDeviceBulkDA
EnableDeviceDA	DisableDeviceDA	EnableDeviceTwoDA
DisableDeviceTwoDA	ReleaseDeviceDA	ReleaseDevice

◆ **初始化设备对象**

函数原型

Visual C++ & C++Builder：

```
BOOL InitDeviceDA(HANDLE hDevice,
                  PPXI1117_PARA_DA pDAPara,
                  int nDAChannel)
```

Visual Basic:

```
Declare Function InitDeviceDA Lib "PXI1117" ( ByVal hDevice As Long, _
                                              ByRef pDAPara As PXI1117_PARA_DA,_
                                              ByVal nDAChannel As Integer) As Boolean
```

Delphi:

```
Function InitDeviceDA(hDevice : Integer;
                      pDAPara:PPXI1117_PARA_DA;
                      nDAChannel: Integer):Boolean;
                      StdCall; External 'PXI1117' Name 'InitDeviceDA';
```

LabVIEW：

请参考相关演示程序。

功能：它负责初始化设备对象中的DA部件，为设备操作就绪有关工作做准备，如预置DA采集通道、采样频率等。但它并不启动DA设备，若要启动DA设备，须在调用此函数之后再调用EnableDeviceDA(但DA要实际输出波形，则一般要等待某种触发事件的到来)。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pDAPara 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，如采样频率等。关于具体操作请参考《DA硬件参数结构》。

nDAChannel DA 通道号，取值范围为[0, 1]。

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 SetDevTrigLevelDA SetDevFrequencyDA
InitDeviceDA WriteDeviceBulkDA ReadDeviceBulkDA
EnableDeviceDA DisableDeviceDA EnableDeviceTwoDA
DisableDeviceTwoDA ReleaseDeviceDA ReleaseDevice

♦ **批量方式将用户缓冲区中的 DA 数据传输至板载 RAM 中**

函数原型:

Visual C++ & C++Builder:

```
BOOL WriteDeviceBulkDA ( HANDLE hDevice,
                         SHORT DABuffer[],
                         LONG nWriteOffsetWords,
                         LONG nWriteSizeWords,
                         PLONG nRetSizeWords,
                         int nDAChannel)
```

Visual Basic:

```
Declare Function WriteDeviceBulkDA Lib "PXI1117" (
                                    ByVal hDevice as Long, _
                                    ByRef DABuffer() As Integer, _
                                    ByVal nWriteOffsetWords As Long, _
                                    ByVal nWriteSizeWords As Long, _
                                    ByRef nRetSizeWords As Long, _
                                    ByVal nDAChannel As Integer) As Boolean
```

Delphi:

```
Function WriteDeviceBulkDA (hDevice : Integer;
                            DABuffer[] : SmallInt;
                            nWriteOffsetWords: LongInt;
                            nWriteSizeWords:LongInt;
                            nRetSizeWords : Pointer;
                            nDAChannel : Integer):Boolean;
                            StdCall; External 'PXI1117' Name ' WriteDeviceBulkDA ';
```

LabView：

请参考相关演示程序。

功能：往指定通道的板载 RAM 中写入批量 DA 数据。在初始化设备之后，启动 DA 之前，便可以用此函数将 DA 数据写入板载 RAM 以供输出。但是在启动之后（即在输出过程中，不能对 RAM 进行写操作，包括读操作）。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

DABuffer[] 接受DA数据的用户缓冲区地址，它可以是一个 16Bit整型数组，也可以是由其他方式分配的16Bit整型缓冲区。关于如何将这些DA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》。

nWriteOffsetWords 相对于该通道物理 RAM 零位置的偏移点(字)。此参数不能超过 RAM 的最大长度(字/点)。

nWriteSizeWords 指定一次往物理缓冲区由 nWriteOffsetWords 参数指定偏移位置开始写入的数据长度。注意此参数的值与 nWriteOffsetWords 参数值之和不能大于指定通道的物理缓冲区即板上 RAM 的最大长度。同时此参数值不能大于 DABuffer[]指定的缓冲区的长度。

nRetSizeWords 返回当前写操作实际实现的数据长度。它表明该函数调用后，在 DABuffer[]中有多少数据是有效的。

nDAChannel DA 通道号，取值范围为[0, 1]。

返回值：如果调用成功，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 SetDevTrigLevelDA SetDevFrequencyDA
InitDeviceDA WriteDeviceBulkDA ReadDeviceBulkDA
EnableDeviceDA DisableDeviceDA EnableDeviceTwoDA
DisableDeviceTwoDA ReleaseDeviceDA ReleaseDevice

♦ **以批量方式将板载 RAM 中的 DA 数据回读至主机的用户缓冲区**

函数原型:

Visual C++ & C++Builder:

```
BOOL ReadDeviceBulkDA ( HANDLE hDevice,
                        SHORT DABuffer[],
                        LONG nReadOffsetWords,
                        LONG nReadSizeWords,
                        PLONG nRetSizeWords,
                        int nDAChannel)
```

Visual Basic:

```
Declare Function ReadDeviceBulkDA Lib "PXI1117" (
                        ByVal hDevice as Long, _
                        ByRef DABuffer() As Integer, _
                        ByVal nReadOffsetWords As Long, _
                        ByVal nReadSizeWords As Long, _
                        ByRef nRetSizeWords As Long, _
                        ByVal nDAChannel As Integer) As Boolean
```

Delphi:

```
Function ReadDeviceBulkDA (hDevice : Integer;
                        DABuffer[] : SmallInt;
                        nReadOffsetWords : LongInt;
                        nReadSizeWords:LongInt;
                        nRetSizeWords : Pointer;
                        nDAChannel : Integer): Boolean;
                        StdCall; External 'PXI1117' Name ' ReadDeviceBulkDA';
```

LabView：

请参考相关演示程序。

功能：从指定通道的 RAM 中的指定段以及指定段内偏移位置开始将 DA 数据从板载 RAM 中读回至主机的用户缓冲区。但是在启动之后（即在输出过程中），不能对 RAM 进行读操作，包括写操作。该函数的作用是为了验证写入的数据是否正确而提供的。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

DABuffer[] 接受DA数据的用户缓冲区地址，它可以是一个 16Bit整型数组，也可以是由其他方式分配的16Bit整型缓冲区。关于如何将这些DA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》。

nReadOffsetWords 相对于该通道物理 RAM 零位置的偏移点(字)。此参数不能超过 RAM 的最大长度(字/点)。

nReadSizeWords 指定一次从物理缓冲区由 nReadOffsetWords 参数指定偏移位置开始读入的数据长度。注意此参数的值与 nWriteOffsetWords 参数值之和不能大于指定通道的物理缓冲区即板上 RAM 的最大长度。同时此参数值不能大于 DABuffer[]指定的缓冲区的长度。

nRetSizeWords 返回当前写操作实际实现的数据长度。它表明该函数调用后，在 DABuffer[]中有多少数据是有效的。

nDAChannel DA 通道号，取值范围为[0, 1]。

返回值：如果调用成功，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	SetDevTrigLevelDA	SetDevFrequencyDA
InitDeviceDA	WriteDeviceBulkDA	ReadDeviceBulkDA
EnableDeviceDA	DisableDeviceDA	EnableDeviceTwoDA
DisableDeviceTwoDA	ReleaseDeviceDA	ReleaseDevice

♦ **启动 DA 设备**

函数原型

Visual C++ & C++Builder:

```
BOOL EnableDeviceDA (HANDLE hDevice,
                     int nDAChannel)
```

Visual Basic:

```
Declare Function EnableDeviceDA Lib "PXI1117" (ByVal hDevice As Long,_
                                               ByVal nDAChannel As Integer) As Boolean
```

Delphi:

```
Function EnableDeviceDA (hDevice : Integer;
                         nDAChannel : Integer ): Boolean;
```

StdCall; External 'PXI1117' Name ' EnableDeviceDA ';

LabVIEW：

请参考相关演示程序。

功能：启动DA设备，它必须在调用InitDeviceDA后才能调用此函数。调用该函数后它可能立即启动，这就要取决您选择的触发方式或触发源，详细请参考后面的《触发功能详述》。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nDAChannel 通道号，取值范围为[0, 1]。

返回值：如果调用成功，则返回TRUE，且DA准备就绪，等待触发事件的到来就开始实际的DA输出， 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	SetDevTrigLevelDA	SetDevFrequencyDA
InitDeviceDA	WriteDeviceBulkDA	ReadDeviceBulkDA
EnableDeviceDA	DisableDeviceDA	EnableDeviceTwoDA
DisableDeviceTwoDA	ReleaseDeviceDA	ReleaseDevice

♦ 暂停 DA 设备

函数原型

Visual C++ & C++Builder：

```
BOOL DisableDeviceDA (HANDLE hDevice,
                      int nDAChannel )
```

Visual Basic:

```
Declare Function DisableDeviceDA Lib "PXI1117" (ByVal hDevice as Long,_
                                                ByVal nDAChannel As Integer) As Boolean
```

Delphi:

```
Function DisableDeviceDA (hDevice : Integer;
                          nDAChannel As Integer) : Boolean;
                          StdCall; External 'PXI1117' Name ' DisableDeviceDA ';
```

LabVIEW：

请参考相关演示程序。

功能：暂停DA设备。它必须在调用EnableDeviceDA后才能调用此函数。该函数除了停止DA设备不再转换以外，不改变设备的其他任何工作参数。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nDAChannel 通道号，取值范围为[0, 1]。

返回值：如果调用成功，则返回TRUE，且DA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	SetDevTrigLevelDA	SetDevFrequencyDA
InitDeviceDA	WriteDeviceBulkDA	ReadDeviceBulkDA
EnableDeviceDA	DisableDeviceDA	EnableDeviceTwoDA
DisableDeviceTwoDA	ReleaseDeviceDA	ReleaseDevice

♦ 同时启动 2 路 DA 设备

函数原型

Visual C++ & C++Builder:

```
BOOL EnableDeviceTwoDA (HANDLE hDevice)
```

Visual Basic:

```
Declare Function EnableDeviceTwoDA Lib "PXI1117" (ByVal hDevice As Long) As Boolean
```

Delphi:

```
Function EnableDeviceTwoDA (hDevice : Integer): Boolean;
                            StdCall; External 'PXI1117' Name ' EnableDeviceTwoDA ';
```

LabVIEW：

请参考相关演示程序。

功能：同时启动 2 路DA设备，它必须在调用InitDeviceDA后才能调用此函数。调用该函数后它可能立即启

动，这就要取决您选择的触发方式或触发源，详细请参考后面的《触发功能详述》。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

返回值：如果调用成功，则返回TRUE，且 2 路DA准备就绪，等待触发事件的到来就开始实际的DA输出，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 SetDevTrigLevelDA SetDevFrequencyDA
InitDeviceDA WriteDeviceBulkDA ReadDeviceBulkDA
EnableDeviceDA DisableDeviceDA EnableDeviceTwoDA
DisableDeviceTwoDA ReleaseDeviceDA ReleaseDevice

♦ 同时暂停 2 路 DA 设备

函数原型

Visual C++ & C++Builder：

```
BOOL DisableDeviceTwoDA (HANDLE hDevice)
```

Visual Basic:

```
Declare Function DisableDeviceTwoDA Lib "PXI1117" (ByVal hDevice as Long) As Boolean
```

Delphi:

```
Function DisableDeviceTwoDA (hDevice : Integer) : Boolean;
                                StdCall; External 'PXI1117' Name ' DisableDeviceTwoDA ';
```

LabVIEW：

请参考相关演示程序。

功能：同时暂停 2 路DA设备。它必须在调用EnableDeviceTwoDA后才能调用此函数。该函数除了停止DA设备不再转换以外，不改变设备的其他任何工作参数。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

返回值：如果调用成功，则返回TRUE，且DA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 SetDevTrigLevelDA SetDevFrequencyDA
InitDeviceDA WriteDeviceBulkDA ReadDeviceBulkDA
EnableDeviceDA DisableDeviceDA EnableDeviceTwoDA
DisableDeviceTwoDA ReleaseDeviceDA ReleaseDevice

♦ 释放 DA 设备

函数原型

Visual C++ & C++Builder：

```
BOOL ReleaseDeviceDA ( HANDLE hDevice,
                       int nDAChannel)
```

Visual Basic:

```
Declare Function ReleaseDeviceDA Lib "PXI1117" (ByVal hDevice as Long, _
                                                ByVal nDAChannel As Integer) As Boolean
```

Delphi:

```
Function ReleaseDeviceDA (hDevice : Integer ;
                          nDAChannel As Integer) : Boolean;
                          StdCall; External 'PXI1117' Name ' ReleaseDeviceDA';
```

LabView：

请参考相关演示程序。

功能：释放DA设备。它必须在调用InitDeviceDA后的某个时刻调用此函数。该函数除了停止DA设备，还释放掉所占用的各种资源。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

nDAChannel 设备通道号，取值范围为[0, 1]。

返回值：如果调用成功，则返回TRUE，且DA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 SetDevTrigLevelDA SetDevFrequencyDA

InitDeviceDA　WriteDeviceBulkDA　ReadDeviceBulkDA
EnableDeviceDA　DisableDeviceDA　EnableDeviceTwoDA
DisableDeviceTwoDA　ReleaseDeviceDA　ReleaseDevice

♦ **采样和和传输函数一般调用顺序**

① CreateDevice (创建设备对象)
② InitDeviceDA (初始化设备)
③ WriteDeviceBulkDA(批量写入DA数据到板载RAM)
④ EnableDeviceDA (启动DA设备)/ EnableDeviceTwoDA(同时启动 2 路DA设备)
⑤ DisableDeviceDA(暂停DA设备)/ DisableDeviceTwoDA(同时暂停 2 路DA设备)
⑥ ReleaseDevice
关于调用过程的图形说明请参考《绪论》。

第四节、DA 硬件参数系统保存与读取函数原型说明

♦ **写设备硬件参数函数到 Windows 系统中**

函数原型：

Viusal C++ & C++ Builder:

```
BOOL SaveParaDA (HANDLE hDevice,
                 PPXI1117_PARA_DA pDAPara,
                 int nDAChannel)
```

Visual Basic:

```
Declare Function SaveParaDA Lib "PXI1117" (ByVal hDevice As Long, _
                                           ByRef pDAPara As PXI1117_PARA_DA,_
                                           ByVal nDAChannel As Integer) As Boolean
```

Delphi:

```
Function SaveParaDA (hDevice : Integer;
                     pDAPara:PPXI1117_PARA_DA;
                     nDAChannel : Integer):Boolean;
                     StdCall; External 'PXI1117' Name ' SaveParaDA ';
```

LabVIEW:

请参考相关演示程序。

功能：负责把用户设置的硬件参数保存在 Windows 系统中，以供下次使用。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
pDAPara 设备硬件参数，关于 PPXI1117_PARA_DA 的详细介绍请参考 PXI1117.h 或 PXI1117.Bas 或 PXI1117.Pas函数原型定义文件，也可参考《硬件参数结构》关于该结构的有关说明。
nDAChannel　DA 通道号，取值范围为[0, 1]。
返回值：若成功，返回 TRUE，否则返回 FALSE。
相关函数： CreateDevice　LoadParaDA　SaveParaDA
ReleaseDevice

♦ **从 Windows 系统中读入硬件参数函数**

函数原型：

Visual C++ & C++ Builder:

```
BOOL LoadParaDA(HANDLE hDevice,
                PPXI1117_PARA_DA pDAPara,
                int nDAChannel)
```

Visual Basic:

```
Declare Function LoadParaDA Lib "PXI1117" (ByVal hDevice As Long, _
                                           ByRef pDAPara As PXI1117_PARA_DA, _
                                           ByVal nDAChannel As Integer) As Boolean
```

Delphi:

```
Function LoadParaDA (hDevice : Integer;
                     pDAPara:PPXI1117_PARA_DA;
                     nDAChannel : Integer):Boolean;
```

```
                                StdCall; External 'PXI1117' Name ' LoadParaDA ';
```

LabVIEW:

请参考相关演示程序。

功能：负责从 Windows 系统中读取设备的硬件参数。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pDAPara属于PPXI1117_PARA_DA的结构指针类型，它负责返回PXI硬件参数值，关于结构指针类型PPXI1117_PARA_DA请参考PXI1117.h或PXI1117.Bas或PXI1117.Pas函数原型定义文件，也可参考《硬件参数结构》关于该结构的有关说明。

nDAChannel DA 通道号，取值范围为[0, 1]。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice LoadParaDA SaveParaDA ReleaseDevice

◆ **将硬件参数结构体值复位为出厂默认值**

函数原型：

Viusal C++ & C++ Builder:

```
BOOL ResetParaDA (HANDLE hDevice,
                  PPXI1117_PARA_DA pDAPara,
                  int nDAChannel)
```

Visual Basic:

```
Declare Function ResetParaDA Lib "PXI1117" (ByVal hDevice As Long, _
                                            ByRef pDAPara As PXI1117_PARA_DA,_
                                            ByVal nDAChannel As Integer) As Boolean
```

Delphi:

```
Function ResetParaDA ( hDevice : Integer;
                       pDAPara:PPXI1117_PARA_DA;
                       nDAChannel As Integer):Boolean;
                       StdCall; External 'PXI1117' Name ' ResetParaDA ';
```

LabVIEW:

请参考相关演示程序。

功能：负责将硬件参数的值复位至出厂默认值，不仅会将 pDAPara 指向的结构体成员值更新为默认值，同时会将系统中保存的参数更新为默认值。这些默认值在产品驱动第一次被安装时会出现。而且这些默认值的设定是充分的考虑到用户的实际情况，确保用户不用外部任何条件，只要开始采集数据，即可获得相应的结果。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pDAPara 设备硬件参数，关于PPXI1117_PARA_DA的详细介绍请参考PXI1117.h或PXI1117.Bas或PXI1117.Pas函数原型定义文件，也可参考《硬件参数结构》关于该结构的有关说明。调用此函数后，该参数指向的结构体成员将被复位至默认值。

nDAChannel DA 通道号，取值范围为[0, 1]。

返回值：若成功，返回 TRUE，它表明已成功将系统中的 DA 参数复位至默认值，同时更新了 pDAPara 指向的结构体。否则返回 FALSE。

相关函数： CreateDevice LoadParaDA SaveParaDA ResetParaDA ReleaseDevice

第四章 硬件参数结构

DA 硬件参数结构（PXI1117_PARA_DA）

Visual C++ & C++Builder：

```
typedef struct _PXI1117_PARA_DA
{
    LONG Frequency;          // 点频率[153Hz, 1MHz], 单位 Hz
```

```
    LONG OutputRange;          // DA 输出量程
    LONG LoopSizeWords;        // 循环长度(单位：字/点)
    LONG LoopCount;            // 整过 RAM 的大循环次数，=0:无限循环， =n:表示 n 次循环(1<n< 65535)
    LONG TriggerMode;          // 触发模式选择
    LONG TriggerType;          // 触发类型
    LONG TriggerDir;           // 触发方向选择
    LONG SynClockSource;       // 同步时钟源选择(注意：两路 DA，每次只能选一路输出)
    LONG SynClockDir;          // 同步时钟源输入输出方向选择
    LONG ClockSource;          // 时钟源选择(内/外时钟源)
    LONG bClockOutput;         // 允许时钟输出
} PXI1117_PARA_DA, *PPXI1117_PARA_DA;
```

Visual Basic：

```
Type PXI1117_PARA_DA
    Frequency As Long          '点频率[153Hz, 1MHz]，单位 Hz
    OutputRange As Long        ' 输出量程
    LoopSizeWords As Long      ' 循环长度(单位：字/点)
    LoopCount As Long          ' 整过 RAM 的大循环次数，=0:无限循环， =n:表示 n 次循环(1<n< 65535)
    TriggerMode As Long        ' 触发模式选择
    TriggerType As Long        ' 触发类型
    TriggerDir As Long         ' 触发源选择
    SynClockSource As Long     ' 同步时钟源选择(注意：两路 DA，每次只能选一路输出)
    SynClockDir As Long        ' 同步时钟源输入输出方向选择
    ClockSource As Long        ' 时钟源选择
    bClockOutput As Long       ' 允许时钟输出
End Type
```

Delphi:

```
Type   // 定义结构体数据类型
PPXI1117_PARA_DA = ^ PXI1117_PARA_DA;   // 指针类型结构
PXI1117_PARA_DA = record       // 标记为记录型
    Frequency : LongInt;       // 点频率[153Hz, 1MHz]，单位 Hz
    OutputRange : LongInt;     // 输出量程
    LoopSizeWords: LongInt;    // 循环长度(单位：字/点)
    LoopCount: LongInt;        // 整过 RAM 的大循环次数，=0:无限循环， =n:表示 n 次循环(1<n< 65535)
    TriggerMode: LongInt;      // 触发模式选择
    TriggerType : LongInt;     // 触发类型
    TriggerDir : LongInt;      // 触发方向选择
    SynClockSource: LongInt;   // 同步时钟源选择(注意：两路 DA，每次只能选一路输出)
    SynClockDir: LongInt;      // 同步时钟源输入输出方向选择
    ClockSource : LongInt;     // 时钟源选择
    bClockOutput: LongInt;     // 允许时钟输出
End;
```

LabVIEW:

请参考相关演示程序。

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

Frequency DA 输出时的点频率，即刷新频率，取值范围为[153Hz, 1MHz]，单位 Hz。

OutputRange 输出信号的量程范围，取值如下表：

常量名	常量值	功能定义
PXI1117_OUTPUT_N10000_P10000mV	0x0000	±10000mV

PXI1117_OUTPUT_N5000_P5000mV	0x0001	±5000mV
PXI1117_OUTPUT_0_P10000mV	0x0002	0～10000mV

关于电压值到设备原始码之间的换算公式请参考《DA的电压值如何转换成输出到DA转换器的LSB原码数据》。

LoopSizeWords 循环长度(单位：字/点)。

LoopCount 总循环次数，=0：无限循环, =n：表示 n 次循环(1<n<65535)。该参数只有触发模式为连续触发时有效。它表示指定通道指定有效段输出的次数。比如有效段数 SegmentCount 为 3 时，该参数为 4 时，则实际输出的段序列为：

循环第 1 次			循环第 2 次			循环第 3 次			循环第 4 次		
段 0	段 1	段 2	段 0	段 1	段 2	段 0	段 1	段 2	段 0	段 1	段 2

TriggerMode DA 触发模式选择。选项值如下表：

常量名	常量值	功能定义
PXI1117_TRIGMODE_SOFT	0x0000	软件触发(属于内触发)
PXI1117_TRIGMODE_POST	0x0001	硬件后触发(属于外触发)

TriggerType DA 触发类型。选项值如下表：

常量名	常量值	功能定义
PXI1117_TRIGTYPE_EDGE	0x0000	边沿触发
PXI1117_TRIGTYPE_PULSE	0x0001	脉冲触发(电平)

TriggerDir DA 外触发方式使用信号方向，只对硬件模拟 ATR 触发源和硬件 DTR 数字触发源有效。选项值如下表：

常量名	常量值	功能定义
PXI1117_TRIGDIR_NEGATIVE	0x0000	负向触发(低脉冲/下降沿触发)
PXI1117_TRIGDIR_POSITIVE	0x0001	正向触发(高脉冲/上升沿触发)
PXI1117_TRIGDIR_POSIT_NEGAT	0x0002	正负向触发(高/低脉冲或上升/下降沿触发)

当 TriggerDir = PXI1117_TRIGDIR_NEGATIVE 时，表示外部触发信号须由大于 TrigLevelVolt 变成小于 TrigLevelVolt 时产生触发事件（即下降沿触发）。

当 TriggerDir = PXI1117_TRIGDIR_POSITIVE 时，表示外部触发信号由小于 TrigLevelVolt 变成大于 TrigLevelVolt 时产生触发事件 (即上升沿触发)。

当 TriggerDir = PXI1117_TRIGDIR_POSIT_NEGAT 时，凡外部触发信号发生以上两种情况中的任意一种则产生触发事件。

SynClockSource 同步时钟源选择，选项值如下表。注意：两路 DA，每次只能选一路输出,。

常量名	常量值	功能定义
PXI1117_CLOCKSRC_io3V_TRIG4	0x0000	同步时钟 io3V_TRIG4
PXI1117_CLOCKSRC_io3V_TRIG5	0x0001	同步时钟 io3V_TRIG5
PXI1117_CLOCKSRC_io3V_TRIG6	0x0002	同步时钟 io3V_TRIG6
PXI1117_CLOCKSRC_io3V_TRIG7	0x0003	同步时钟 io3V_TRIG7

SynClockDir 同步时钟源输入输出方向选择，选项值如下表：

常量名	常量值	功能定义
PXI1117_SYNCLOCK_OUT	0x0000	同步时钟输出
PXI1117_SYNCLOCK_IN	0x0001	同步时钟输入

ClockSource DA 外时钟选择，选项值如下表：

常量名	常量值	功能定义
PXI1117_CLOCKSRC_IN	0x0000	内部时钟
PXI1117_CLOCKSRC_OUT	0x0001	外部时钟

bClockOutput 允许时钟输出到 CLKOUT，=TRUE：允许时钟输出，=FALSE：禁止时钟输出。两路 DA，每次只能选一个输出。

常量名	常量值	功能定义
PXI1117_CLOCKOUT_DISABLE	0x0000	禁止本卡上的自带时钟向外输出
PXI1117_CLOCKOUT_ENABLE	0x0001	允许本卡上的自带时钟向外输出

第五章　数据格式转换与排列规则

第一节、DA 的电压值如何转换成输出到 DA 转换器的 LSB 原码数据

量程(伏)	计算机语言换算公式	Lsb 取值范围
0～10000mV	Lsb = Volt / (10000.00 / 4096)	[0，4095]
±5000mV	Lsb = Volt / (10000.00 / 4096) + 2048	[0，4095]
±10000mV	Lsb = Volt / (20000.00 / 4096) + 2048	[0，4095]

请注意这里求得的LSB数据就是用于WriteDeviceBulkDA中的DABuffer[]参数的。

第二节、关于 DA 数据 DABuffer 缓冲区中的数据排放规则

由于各个通道的段信息与波形数据均共享一个板载物理 RAM，它们的排放顺序如图 5.1，系统默认值为四个通道均分整个 RAM 空间，即默认每通道 RAM 空间为 256K 点。从下图可以看出，各个通道所占 RAM 空间不一定相等，可大可小，只是四个通道的总空间不能大于板载物理 RAM 空间即可。

通道 0	通道 1	通道 2	通道 3
0 至(256K-1)	256 至(512K-1)	512 至(768K-1)	768K 至(1024K-1)

图 5.1

关于每个通道 RAM 空间的内部分配是这样的，其空间首部存放的是所有段的段信息数据，其后才是各个段的波形数据，再其后可能还有未用空间。假如有三个分段，如图 5.2：

段 0 信息	段 1 信息	段 2 信息	段 0 波形数据	段 1 波形数据	段 2 波形数据	未用空间

图 5.2

关于每个段的段信息包括的内容有：该段波形数据在 RAM 中的起始地址、终止地址、段循环次数，如表 5.2.1，注意其段起始地址和终止地址是由 PXI1117_PARA_DA 中的 SegmentInfo 决定的。

表 5.2.1

板载 RAM 内存单元(16Bit)	各单元定义	有效位
0	段 0 波形数据起始地址低 12 位	D11:D0
1	段 0 波形数据起始地址高 8 位	D7:D0
2	段 0 波形数据终止地址低 12 位	D11:D0
3	段 0 波形数据终止地址高 8 位	D7:D0
4	段 0 循环次数低 12 位	D11:D0
5	段 0 循环次数高 8 位	D7:D0
6	段 1 波形数据起始地址低 12 位	D11:D0
7	段 1 波形数据起始地址高 8 位	D7:D0
8	段 1 波形数据终止地址低 12 位	D11:D0
9	段 1 波形数据终止地址高 8 位	D7:D0
10	段 1 循环次数低 12 位	D11:D0
11	段 1 循环次数高 8 位	D7:D0
:	:	:
段信息结束后便是波形数据	段信息结束后便是波形数据	D11:D0

第六章 上层用户函数接口应用实例

第一节、简易程序演示说明

怎样使用WriteDeviceBulkDA函数进行批量DA数据输出

其详细应用实例及工程级代码请参考 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 PXI1117.h 和 Sys.cpp)。

[程序]\|**[阿尔泰测控演示系统]**\|**[PXI1117 2 路 DA 卡]**\|**[Microsoft Visual C++]**\|**[简易代码演示]**\|**[DA 演示源程序]**

其简易程序默认存放路径为：系统盘\ART\PXI1117\SAMPLES\VC\SIMPLE\DA\BULK

其他语言的演示可以用上面类似的方法找到。

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 PXI1117.h 和 DADoc.cpp)。

[程序]\|**[阿尔泰测控演示系统]**\|**[PXI1117 2 路 DA 卡]**\|**[Microsoft Visual C++]**\|**[高级代码演示]**

其默认存放路径为：系统盘\ART\ PXI1117\SAMPLES\VC\ADVANCED

其他语言的演示可以用上面类似的方法找到。

第七章 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“PXI1117_”）

函数名	函数功能	备注
① PXI 总线内存映射寄存器操作函数		
GetDeviceAddr	取得指定 PXI 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 创建 Visual Basic 子线程，线程数量可达 32 个以上		
SetEvent	设置中断事件	
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	
④ 文件对象操作函数		
CreateFileObject	初始设备文件对象	
WriteFile	请求文件对象写用户数据到磁盘文件	
ReadFile	请求文件对象读数据到用户空间	
SetFileOffset	设置文件指针偏移	
GetFileLength	取得文件长度	
ReleaseFile	释放已有的文件对象	
GetDiskFreeBytes	取得指定磁盘的可用空间(字节)	适用于所有设备

⑤ 其他函数		
GetLastErrorEx	取得驱动函数错误信息	
RemoveLastErrorEx	移除指定函数的最后一次错误信息	

第二节、PXI 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)
```

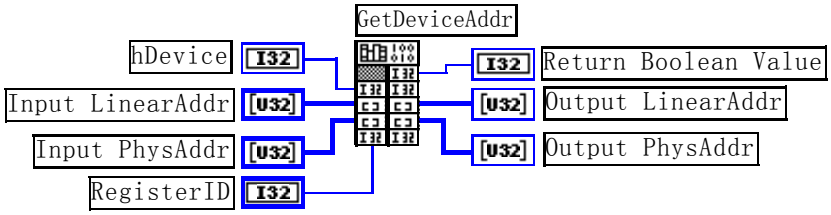
Visual Basic:

```
Declare Function GetDeviceAddr Lib "PXI1117" (ByVal hDevice As Long, _
                                              ByRef LinearAddr As Long, _
                                              ByRef PhysAddr As Long, _
                                              Optional ByVal RegisterID As Integer = 0) As Boolean
```

Delphi:

```
Function GetDeviceAddr(hDevice : Integer;
                       LinearAddr : Pointer;
                       PhysAddr : Pointer;
                       RegisterID : Integer = 0) : Boolean;
                       StdCall; External 'PXI1117' Name ' GetDeviceAddr ';
```

LabVIEW:



功能：取得 PXI 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。本设备的寄存器组 ID 定义如下：

常量名	常量值	功能定义
PXI1117_REG_MEM_CPLD	0x0000	0 号寄存器对应板上控制单元所使用的内存模式基地址(使用 LinearAddr)
PXI1117_REG_IO_CPLD	0x0001	1 号寄存器对应板上控制单元所使用的 IO 模式基地址(使用 PhysAddr)

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　GetDeviceAddr　WriteRegisterByte　WriteRegisterWord　WriteRegisterULong　ReadRegisterByte　ReadRegisterWord　ReadRegisterULong　ReleaseDevice

Visual C++ & C++ Builder 程序举例:

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

Visual Basic 程序举例:

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

◆ **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 "PXI1117" (ByVal hDevice As Long, _
                                             ByVal pulPCIBar (0 to 5) As Long) As Boolean
```

Delphi:

```
Function GetDeviceBar (hDevice : Integer;
                       pulPCIBar : Pointer) : Boolean;
                       StdCall; External 'PXI1117' 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 PXI BAR 所有地址。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice　GetDeviceAddr　WriteRegisterByte　WriteRegisterWord　WriteRegisterULong　ReadRegisterByte　ReadRegisterWord　ReadRegisterULong　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 "PXI1117" (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 'PXI1117' Name ' GetDevVersion ';
```

LabVIEW:

请参考相关演示程序。

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 固件版本。

pulDriverVersion 驱动版本。

返回值：若成功，返回 TRUE，否则返回 FALSE。

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

- **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++ & C++ Builder:

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

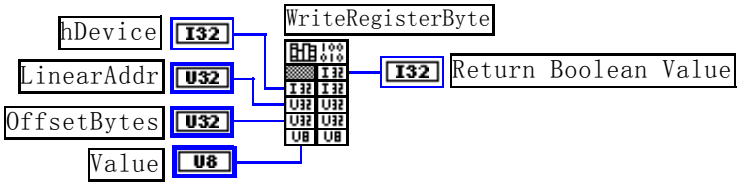
Visual Basic:

```
Declare Function WriteRegisterByte Lib "PXI1117" (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 'PXI1117' Name ' WriteRegisterByte ';
```

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　GetDeviceAddr　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 PXI 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)
```

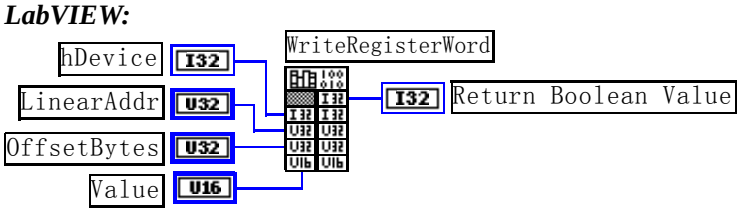
Visual Basic:

```
Declare Function WriteRegisterWord Lib "PXI1117" (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 'PXI1117' Name ' WriteRegisterWord ';
```

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　GetDeviceAddr　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 PXI 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)
```

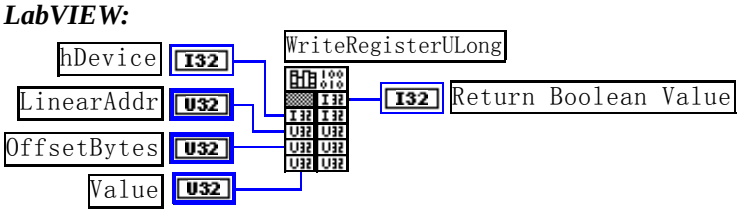
Visual Basic:

```
Declare Function WriteRegisterULong Lib "PXI1117" (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 'PXI1117' Name ' WriteRegisterULong ';
```

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	GetDeviceAddr	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 PXI 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)
```

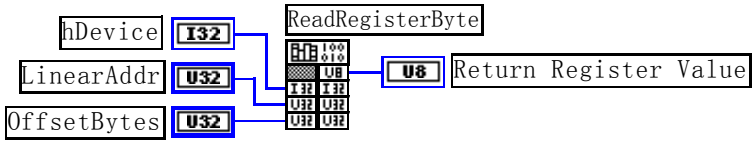
Visual Basic:

```
Declare Function ReadRegisterByte Lib "PXI1117" (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 'PXI1117' Name ' ReadRegisterByte ';
```

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 GetDeviceAddr 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 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++ & C++ Builder:

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

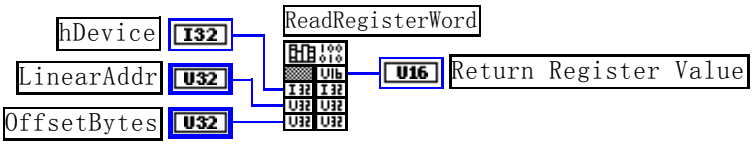
Visual Basic:

```
Declare Function ReadRegisterWord Lib "PXI1117" ( 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 'PXI1117' Name ' ReadRegisterWord ';
```

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　GetDeviceAddr　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 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++ & C++ Builder:

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

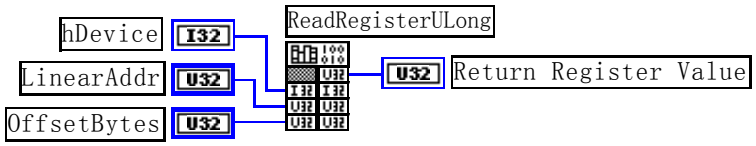
Visual Basic:

```
Declare Function ReadRegisterULong Lib "PXI1117" (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 'PXI1117' Name ' ReadRegisterULong ';
```

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 GetDeviceAddr 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 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

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**

函数原型：

Visual C++ & C++ Builder:

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

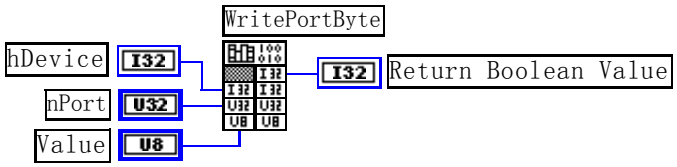
Visual Basic:

```
Declare Function WritePortByte Lib "PXI1117" ( 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 'PXI1117' 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**

函数原型：

Visual C++ & C++ Builder:

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

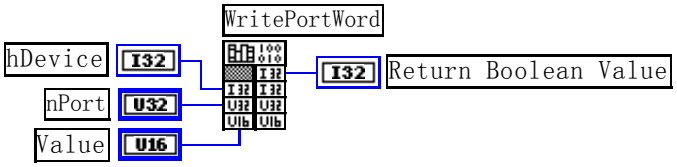
Visual Basic:

```
Declare Function WritePortWord Lib "PXI1117" (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 'PXI1117' 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**

函数原型：
Visual C++ & C++ Builder:

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

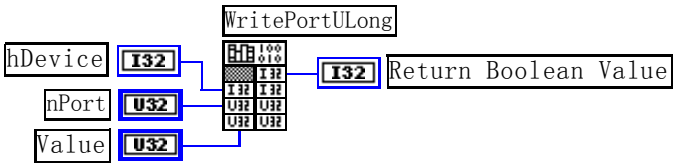
Visual Basic:

```
Declare Function WritePortULong Lib "PXI1117" (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 'PXI1117' 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**

函数原型：
Visual C++ & C++ Builder:

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

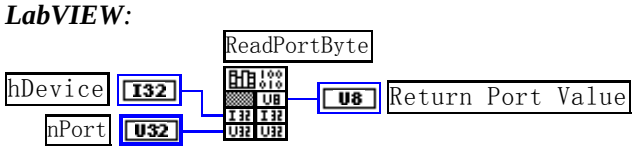
Visual Basic:

```
Declare Function ReadPortByte Lib "PXI1117" (ByVal hDevice As Long, _
                                             ByVal nPort As Long ) As Byte
```

Delphi:

```
Function ReadPortByte(hDevice : Integer;
                      nPort : LongWord) : Byte;
                      StdCall; External 'PXI1117' 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**

函数原型：
Visual C++ & C++ Builder:

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

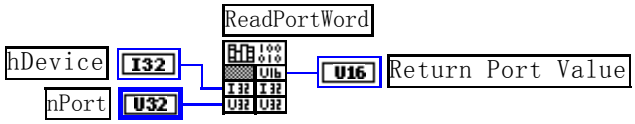
Visual Basic:

```
Declare Function ReadPortWord Lib "PXI1117" (ByVal hDevice As Long, _
                                             ByVal nPort As Long ) As Integer
```

Delphi:

```
Function ReadPortWord(hDevice : Integer;
                      nPort : LongWord) : Word;
                      StdCall; External 'PXI1117' 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**

函数原型：
Visual C++ & C++ Builder:

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

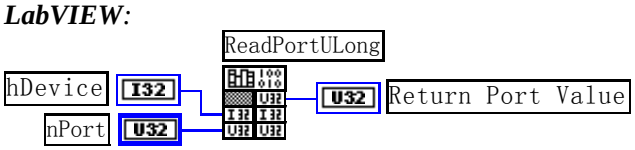
Visual Basic:

```
Declare Function ReadPortULong Lib "PXI1117" (ByVal hDevice As Long, _
                                              ByVal nPort As Long ) As Long
```

Delphi:

```
Function ReadPortULong(hDevice : Integer;
                       nPort : LongWord) : LongWord;
                       StdCall; External 'PXI1117' 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）

◆ **设置中断事件**

函数原型：

Visual C++ & C++ Builder:

```
BOOL SetEvent (HANDLE hDevice,
               HANDLE hEvent)
```

Visual Basic:

```
Declare Function SetEvent Lib " PXI1117 " (ByVal hDevice As Long,_
                                           ByVal hEvent As Long) As Boolean
```

Delphi:

```
Function SetEvent(hDevice : Integer;
                  hEvent : Integer) : Boolean;
                  StdCall; External 'PXI1117' Name ' SetEvent ';
```

LabVIEW:

请参见相关演示程序。

功能：设置中断事件。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

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。

◆ **创建内核系统事件**

函数原型：

Visual C++ & C++ Builder:

```
BOOL CreateSystemEvent (void)
```

Visual Basic:

```
Declare Function CreateSystemEvent Lib " PXI1117 " () As Boolean
```

Delphi:

```
Function CreateSystemEvent () : Boolean;
                              StdCall; External 'PXI1117' Name ' CreateSystemEvent ';
```

LabVIEW:

请参见相关演示程序。

功能：创建系统内核事件对象。

参数：无。

返回值：若成功，则返回 TRUE。

◆ **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 " PXI1117 " (ByVal hEvent As Long) As Boolean
```

Delphi:

```
Function ReleaseSystemEvent(hEvent : Integer) : Boolean;
                              StdCall; External 'PXI1117' Name ' ReleaseSystemEvent ';
```

LabVIEW:

请参见相关演示程序。

功能：释放系统内核事件对象。

参数：hEvent 被释放的内核事件对象。它应由CreateSystemEvent成功创建的对象。
返回值：若成功，则返回 TRUE。

第五节、文件对象操作函数原型说明

◆ **创建文件对象**

函数原型：

Visual C++ & C++ Builder:

```
HANDLE CreateFileObject (HANDLE hDevice,
                         LPCTSTR NewFileName,
                         int Mode)
```

Visual Basic:

```
Declare Function CreateFileObject Lib "PXI1117" (ByVal hDevice As Long, _
                                                 ByVal NewFileName As String, _
                                                 ByVal Mode As Integer) As Long
```

Delphi:

```
Function CreateFileObject (hDevice : Integer;
                           NewFileName: String;
                           Mode : Integer) : Integer;
                           Stdcall; external 'PXI1117' Name ' CreateFileObject ';
```

LabVIEW:

请参见相关演示程序。

功能：初始化设备文件对象，以期待 WriteFile 请求准备文件对象进行文件操作。
参数：
hDevice 设备对象句柄，它应由CreateDevice创建。
NewFileName 与新文件对象关联的磁盘文件名，可以包括盘符和路径等信息。在 C 语言中，其语法格式如：“C:\\PXI1117\\Data.Dat”，在 Basic 中，其语法格式如：“C:\PXI1117\Data.Dat”。
Mode 文件操作方式，所用的文件操作方式控制字定义如下(可通过或指令实现多种方式并操作)：

常量名	常量值	功能定义
PXI1117_modeRead	0x0000	只读文件方式
PXI1117_modeWrite	0x0001	只写文件方式
PXI1117_modeReadWrite	0x0002	既读又写文件方式
PXI1117_modeCreate	0x1000	如果文件不存在可以创建该文件，如果存在，则重建此文件，且清 0
PXI1117_typeText	0x4000	以文本方式操作文件

返回值：若成功，则返回文件对象句柄。
相关函数： CreateDevice　CreateFileObject　WriteFile
ReadFile　ReleaseFile　ReleaseDevice

◆ **通过设备对象，往指定磁盘上写入用户空间的采样数据**

函数原型：

Visual C++ & C++ Builder:

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

Visual Basic:

```
Declare Function WriteFile Lib "PXI1117" (ByVal hFileObject As Long,_
                                          ByRef pDataBuffer As Byte,_
                                          ByVal nWriteSizeBytes As Long) As Boolean
```

Delphi:

```
Function WriteFile(hFileObject: Integer;
                   pDataBuffer : Pointer;
                   nWriteSizeBytes : LongWord) : Boolean;
                   Stdcall; external 'PXI1117' Name ' WriteFile ';
```

LabVIEW:

详见相关演示程序。

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

参数：

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

pDataBuffer 用户数据空间地址，可以是用户分配的数组空间。

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

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

相关函数： CreateFileObject WriteFile ReadFile ReleaseFile

- **通过设备对象,从指定磁盘文件中读采样数据**

函数原型：

Visual C++ & C++ Builder:

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

Visual Basic:

```
Declare Function ReadFile Lib "PXI1117" ( ByVal hFileObject As Long, _
                                          ByRef pDataBuffer As Integer, _
                                          ByVal OffsetBytes As Long, _
                                          ByVal nReadSizeBytes As Long) As Boolean
```

Delphi:

```
Function ReadFile(hFileObject : Integer;
                  pDataBuffer : Pointer;
                  OffsetBytes : LongWord;
                  nReadSizeBytes : LongWord) : Boolean;
                  Stdcall; external 'PXI1117' Name ' ReadFile ';
```

LabVIEW:

详见相关演示程序。

功能：将磁盘数据从指定文件中读入用户内存空间中，其访问方式可由用户在创建文件对象时指定。

参数：

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

pDataBuffer 用于接受文件数据的用户缓冲区指针，可以是用户分配的数组空间。

OffsetBytes 指定从文件开始端所偏移的读位置。

nReadSizeBytes 告诉设备对象从磁盘上一次读入数据的长度(以字为单位)。

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

相关函数： CreateFileObject WriteFile ReadFile ReleaseFile

- **设置文件偏移位置**

函数原型：

Visual C++ & C++ Builder:

```
BOOL SetFileOffset (HANDLE hFileObject,
                    ULONG nOffsetBytes)
```

Visual Basic:

```
Declare Function SetFileOffset Lib "PXI1117" ( ByVal hFileObject As Long,
                                               ByVal nOffsetBytes As Long) As Boolean
```

Delphi:

```
Function SetFileOffset ( hFileObject : Integer;
                         nOffsetBytes : LongWord) : Boolean;
                         Stdcall; external 'PXI1117' Name ' SetFileOffset ';
```

LabVIEW:

详见相关演示程序。

功能：设置文件偏移位置，用它可以定位读写起点。
参数：hFileObject 文件对象句柄，它应由CreateFileObject创建。
返回值：若成功，则返回TRUE，否则返回FALSE，用户可以用GetLastErrorEx捕获错误码。
相关函数： CreateFileObject WriteFile ReadFile ReleaseFile

- **取得文件长度（字节）**

函数原型：
Visual C++ & C++ Builder:
```
ULONG GetFileLength (HANDLE hFileObject)
```
Visual Basic:
```
Declare Function GetFileLength Lib "PXI1117" (ByVal hFileObject As Long) As Long
```
Delphi:
```
Function GetFileLength (hFileObject : Integer) : LongWord;
                        Stdcall; external 'PXI1117' Name ' GetFileLength ';
```
LabVIEW:
详见相关演示程序。

功能：取得文件长度。
参数：hFileObject 设备对象句柄，它应由CreateFileObject创建。
返回值：若成功，则返回>1，否则返回 0，用户可以用GetLastErrorEx捕获错误码。
相关函数： CreateFileObject WriteFile ReadFile ReleaseFile

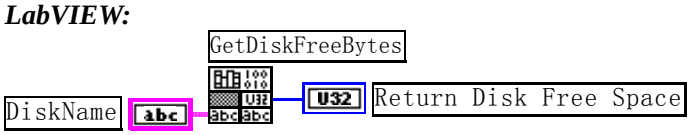
- **释放设备文件对象**

函数原型：
Visual C++ & C++ Builder:
```
BOOL ReleaseFile(HANDLE hFileObject)
```
Visual Basic:
```
Declare Function ReleaseFile Lib "PXI1117" (ByVal hFileObject As Long) As Boolean
```
Delphi:
```
Function ReleaseFile(hFileObject : Integer) : Boolean;
                     Stdcall; external 'PXI1117' Name ' ReleaseFile ';
```
LabVIEW:
详见相关演示程序。

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

- **取得指定磁盘的可用空间**

函数原型：
Visual C++ & C++ Builder:
```
ULONGLONG GetDiskFreeBytes(LPCTSTR DiskName )
```
Visual Basic:
```
Declare Function GetDiskFreeBytes Lib "PXI1117" (ByVal DiskName As String ) As Currency
```
Delphi:
```
Function GetDiskFreeBytes (DiskName: String) : Currency;
                           Stdcall; external 'PXI1117' Name ' GetDiskFreeBytes ';
```
LabVIEW:



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

第六节、其他函数原型说明

◆ **怎样获取驱动函数错误信息**

函数原型：

Visual C++ & C++ Builder:

```
DWORD GetLastErrorEx (LPCTSTR strFuncName,
                      LPTSTR strErrorMsg)
```

Visual Basic:

```
Declare Function GetLastErrorEx Lib "PXI1117" (ByVal strFuncName As String,_
                                               ByVal strErrorMsg As String) As Long
```

Delphi:

```
Function GetLastErrorEx (strFuncName: String;
                         strErrorMsg: String) : LongWord;
                         Stdcall; external 'PXI1117' Name ' GetLastErrorEx ';
```

LabVIEW:

请参见相关演示程序。

功能：将当某个驱动函数出错时，可以调用此函数获得具体的错误和错误信息字串。
参数：
strFuncName 出错函数的名称。注意此函数必须是完整名称，如 DI 初始化函数 PXI1117_InitDeviceDI 出现错误，此时调用该函数时，此参数必须为“PXI1117_InitDeviceDI”，否则得不到相应信息。
strErrorMsg 取得指定函数的错误信息串。该串为字符数组，其分配空间最好不要小于 256 字节。
返回值：返回错误码。
相关函数：无。

◆ **移除驱动函数错误信息**

函数原型：

Visual C++ & C++ Builder:

```
BOOL RemoveLastErrorEx (LPCTSTR strFuncName)
```

Visual Basic:

```
Declare Function RemoveLastErrorEx Lib "PXI1117" (ByVal strFuncName As String) As Boolean
```

Delphi:

```
Function RemoveLastErrorEx (strFuncName: String) : Boolean;
                            Stdcall; external 'PXI1117' Name ' RemoveLastErrorEx';
```

LabVIEW:

详见相关演示程序。

功能：从错误信息库中移除指定函数的最后一次错误信息。
参数：
strFuncName 出错函数的名称。注意此函数必须是完整名称，如 DI 初始化函数 PXI1117_InitDeviceDI 出现错误，此时调用该函数时，此参数必须为“PXI1117_InitDeviceDI”，否则得不到相应信息。
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：无。