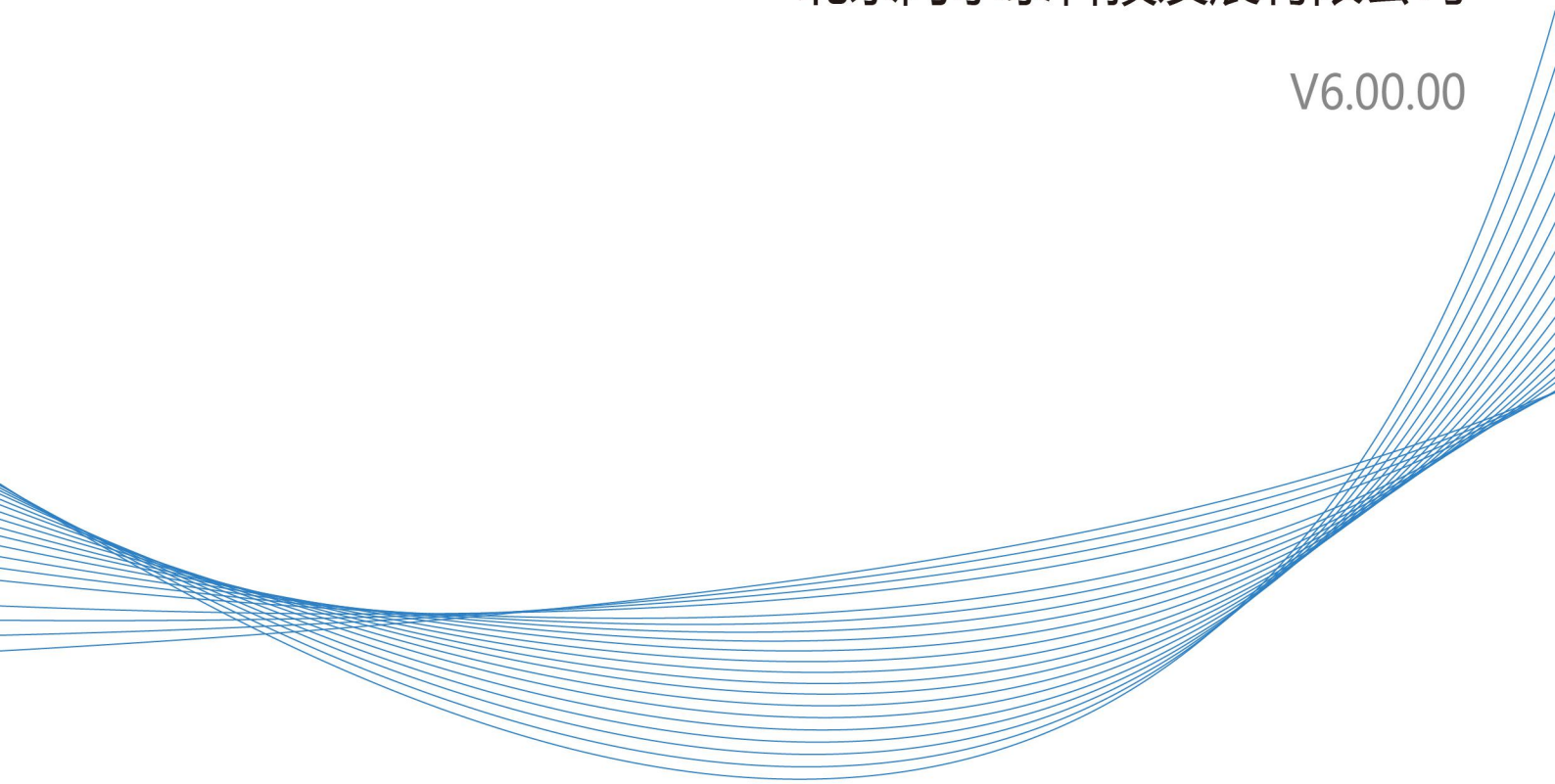
PXI9632 模拟量输出卡

驱动程序使用手册

北京阿尔泰科技发展有限公司

V6.00.00



阿尔泰科技
www.art-control.com

关于本手册

本手册为阿尔泰科技推出的 PXI9632 模拟量输出卡的驱动程序使用手册，其中包括版权信息与命名约定、使用纲要、设备操作函数接口介绍、上层用户函数接口应用实例、共用函数介绍、修改历史等。

文档版本：V6.00.00

目　录

1 版权信息与命名约定

1.1 版权信息

1.2 命名约定

为简化文字内容，突出重点，本文中提到的函数名通常为基本功能名部分，其前缀设备名如 PCIxxxx 则被省略，如 PXI9632_CreateDevice 则写为 CreateDevice。

表 1-2-1：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。

2 使用纲要

2.1 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 WriteDeviceDA。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。

2.2 如何管理设备

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

2.3 哪些函数对您不是必须的

公共函数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。

3 PXI 设备操作函数接口介绍

3.1 设备驱动接口函数总列表

表 3-1-1：驱动接口函数总列表（每个函数省略了前缀“PXI9632_”）

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	以对话框窗体方式列表系统当中的所有的该 PCI 设备	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		
WriteDeviceDA	写 DA 数据	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\PXI9632\INCLUDE\PXI9632.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 PXI9632.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。#include “PXI9632.H”

3.2 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 DeviceLgcID = 0)

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 PXI9632 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 PXI9632 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++ 程序举例

```
    :
HANDLE hDevice;     // 定义设备对象句柄
int DeviceLgcID = 0;
hDevice = PXI9632_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
 }
    :
```

- **取得本计算机系统中 PXI9632 设备的总数量**

函数原型：

Visual C++ :

int GetDeviceCount (HANDLE hDevice)

功能：取得 PXI9632 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 PXI9632 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)
```

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]。

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

- **以对话框窗体方式列表系统当中的所有的该 PCI 设备**

函数原型：

Visual C++ :

BOOL ListDeviceDlg (HANDLE hDevice)

功能：列表系统中 PXI9632 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 PXI9632 设备的配置情况。

相关函数：CreateDevice　　ReleaseDevice

♦ **释放设备对象**

函数原型：

Visual C++:

BOOL ReleaseDevice(HANDLE hDevice)

功能：释放设备对象。

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。

3.3　DA 数据输出函数原型说明

♦ **写 DA 数据**

函数原型:

Visual C++:

```
BOOL_WriteDeviceDA(HANDLE hDevice,
                   LONG OutputRange,
                   SHORT nDAData,
                   ULONG nDAChannel);
```

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。

OutputRange　输出量程，具体定义请参考上面的常量定义部分

nDAData　输出的 DA 原始数据[0, 4095]。

nDAChannel　DA 通道号，取值范围为[0, 31]。

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。

OutputRange　模拟量输出范围。

常量名	常量值	功能定义

const long _OUTPUT_0_P5000mV	0x00	0～5000mV
const long _OUTPUT_0_P10000mV	0x01	0～10000mV
const long _OUTPUT_N5000_P5000mV	0x02	±5000mV
const long _OUTPUT_N10000_P10000mV	0x03	±10000mV

相关函数： CreateDevice ReleaseDevice

4 上层用户函数接口应用实例

4.1 简易程序演示说明

怎么进行 AD 数采操作

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]丨**[阿尔泰测控演示系统]**丨**[PCI9632 DA Card]**丨**[Microsoft VS2005]**丨**[简易代码演示]**丨

4.2 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 PCI9632.h 和 ADDoc.cpp)。

[程序]丨**[阿尔泰测控演示系统]**丨**[PCI9632 DA Card]**丨**[Microsoft VS2005]**丨**[高级代码演示]**

其默认存放路径为：系统盘\ART\PCI9632\SAMPLES\VC\ADVANCED

其他语言的演示可以用上面类似的方法找到。

5 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。

5.1 公用接口函数总列表

表 5-1-1：公用接口函数总列表（每个函数省略了前缀“PXI9632_”）

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
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
②I/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	

5.2 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++ :

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

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
PXI9632_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)
PXI9632_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++ 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…");
}
    :
```

♦ **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,
                    ULONG pulPCIBar[6])
```

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

◆ **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,
                        ULONG LinearAddr,
                        ULONG OffsetBytes,
                        BYTE Value)
```

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++ 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
    :
```

◆ **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,
                       PUCHAR LinearAddr,
                       ULONG OffsetBytes,
                       WORD Value)
```

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++ 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
:
```

◆ **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,
                         PUCHAR LinearAddr,
                         ULONG OffsetBytes,
                         ULONG Value)
```

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++ 程序举例:

```
:
```

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

♦ **以单字节（即 8 位）方式读 PXI 内存映射寄存器的某个单元**

函数原型：

Visual C++ :

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

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++ 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
    :
```

♦ **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,
                       PUCHAR LinearAddr,
                       ULONG OffsetBytes)
```

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++ 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
    :
```

- **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,
                         PUCHAR LinearAddr,
                         ULONG OffsetBytes)
```

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++ 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
    :
```

5.3 I/O 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,
                    PUCHAR nPort,
                    BYTE Value)
```

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++ :

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

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++ :

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

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++ :

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

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,
                  PUCHAR nPort)
```

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,
PUCHAR nPort)

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

5.4　线程操作函数原型说明

- **创建内核系统事件**

函数原型：

Visual C++:

HANDLE CreateSystemEvent(void)

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)

功能：释放系统内核事件对象。

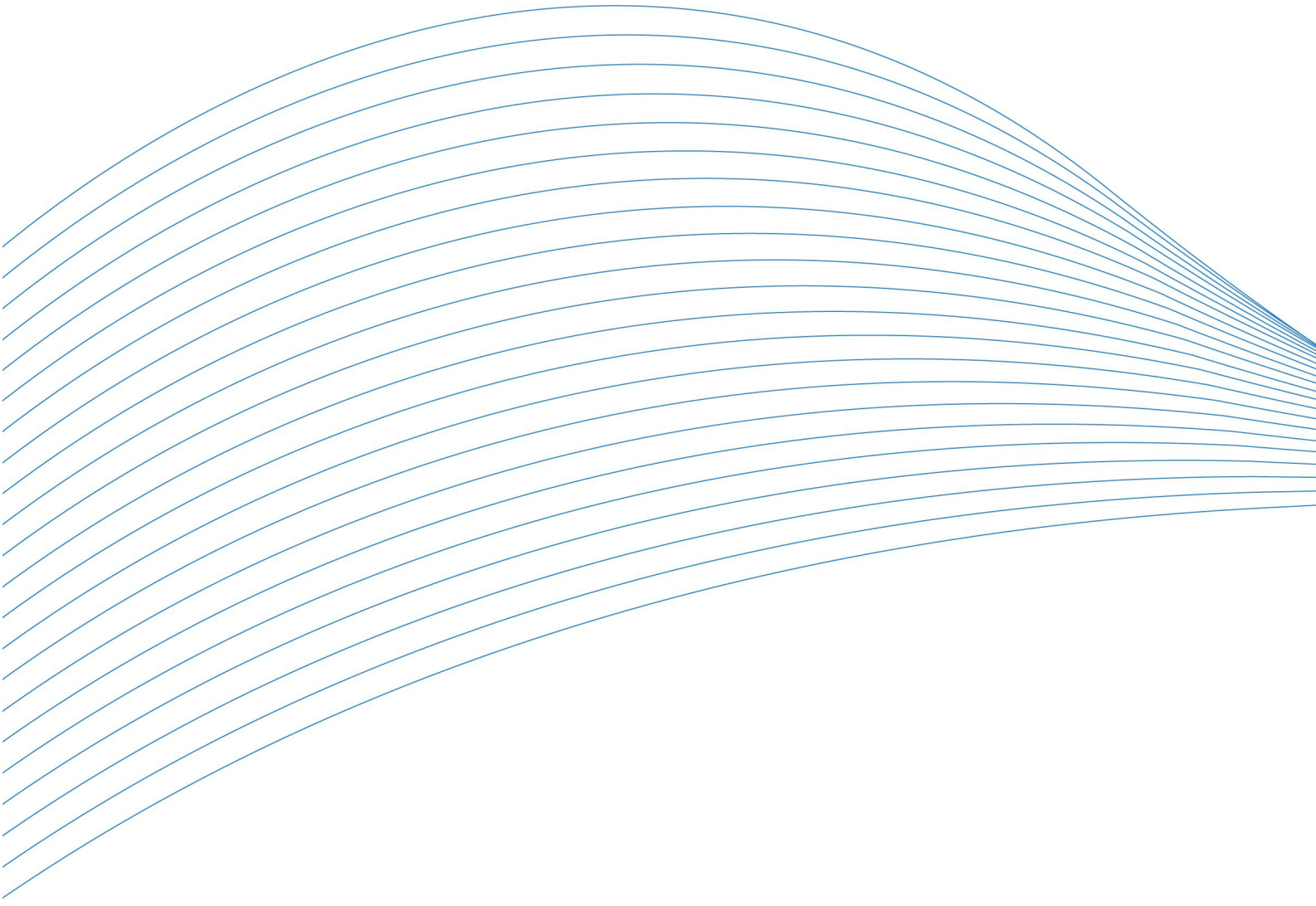
参数：

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。

6 修改历史

修改时间	版本号	修改内容
2017.1.13	V6.00.00	第一版



北京阿尔泰科技发展有限公司
服务热线：400-860-3335
邮编：100086
传真：010-62901157