PCI7062A数字多用表卡

驱动程序使用手册

北京阿尔泰科技发展有限公司

V6.00.00

ART Technology

目　录

1 版权信息与命名约定

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。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。

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 GreateDevice 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相应的驱动函数，如 SetDeviceDO 函数可用实现开关量的输出等。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

公共函数如 CreateFileObject， WriteFile，ReadFile 等一般来说都是辅助性函数，除非您要使用存盘功能。如果您使用上层用户函数访问设备，那么 GetDeviceAddr 等函数您可完全不必理会，除非您是作为底层用户管理设备。而 WritePortByte， WritePortWord， WritePortULong， ReadPortByte，ReadPortWord， ReadPortULong 则对 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 即插即用设备操作函数接口介绍

3.1 设备驱动接口函数总列表（每个函数省略了前缀“PCI7062A_”）

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	
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 设备功能控制函数			
GetDeviceSts	获得设备工作状态	上层用户	
SetDeviceInputRange	设置输入量程	上层用户	
SetDeviceFuntion	万用表功能档选择	上层用户	
SetDeviceDigits	设置万用表分辨率	上层用户	
SetDCInputImpedance	设置直流输入阻抗	上层用户	
SetFilterSpeed	设置滤波速度	上层用户	
ReadDeviceData	读取设备采集数据	上层用户	
ReadDeviceValue	读取设备实际量化数据	上层用户	
ReadBaseFreCNT	读取基本频率计数值	上层用户	
SetDevTrigMode	设置触发模式	上层用户	
SingleTrig	触发模式为单次触发模式有效	上层用户	
SetDevTrigDir	设置触发方向	上层用户	
EnableDevExtOutput	使能外部输出	上层用户	
SetDevExtOutputPolarity	设置外部输出极性	上层用户	
SetExtOutputPulseWidth	设置外部输出脉冲宽度	上层用户	
ReadMeasFreCNT	读取被测信号计数值	上层用户	
ReadMeasRatioCNT	读取被测信号高电平脉冲的计数值	上层用户	
③ 设备校准函数			
WriteCalibrationData	写入校准数据	上层用户	
ReadCalibrationData	读出校准数据	上层用户	
ReadACVFreGainsData	写入交流校准时频率修正增益码值	上层用户	

WriteACVFreGainsData	读出交流校准时频率修正增益码值	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\PCI7062A\INCLUDE\PCI7062A.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 PCI7062A.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。然后加入如下语句：

#include “PCI7062A.H”

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。当然你有 VB6.0 的最新版本，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)，在弹出的对话框中选择 PCI7062A.Bas 模块文件，改问价的路径为用户安装驱动程序后其子目录 Sample\VB 下面。

请注意，因考虑 Visual C++和 Visual Basic 两种语言的兼容问题，在下面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。

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)

Visual Basic：

Declare Function CreateDevice Lib “PCI7062A”(Optional ByVal DeviceLgcID As Integer = 0) As Long

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：DeviceID 设备 ID 标识号。当我想同一个 Windows 系统加入若干相同类型的设备时，设备将以该设备的“基本名称”与 DeviceID 标识值为名称后缀的标识符来确认和管理设备，默认值为 0。

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

相关函数： CreateDevice GetDeviceCount
ListDeviceDlg ReleaseDevice

Visual C++程序举例

：

```
HANDLE hDevice;     // 定义设备对象句柄
int DeviceLgcID = 0;
hDevice = PCI7062A_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 = PCI7062A_CreateDevice (DeviceLgcID)      ‘ 创建设备对象,并取得设备对象句柄
If hDevice = INVALIDE_HANDLE_VALUE Then     ‘ 判断设备对象句柄是否有效
    MsgBox “创建设备对象失败”
    ExitSub          ‘退出该程序
End if
    :
```

- **取得本计算机系统中 PCI7062A 设备的总数量**

函数原型：

Visual C++:

int GetDeviceCount (HANDLE hDevice)

Visual Basic:

Declare Function GetDeviceCount Lib “PCI7062A” (ByVal hDevice As Long) As Integer

功能：取得 PCI7062A 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 PCI7062A 的数量。

相关函数： CreateDevice　　GetDeviceCount
ListDeviceDlg　　ReleaseDevice

- **用对话框控件列表计算机系统中所有 PCI7062A 设备各种配置信息**

函数原型：

Visual C++:

BOOL ListDeviceDlg (HANDLE hDevice)

Visual Basic:

Declare Function ListDeviceDlg Lib “PCI7062A” (ByVal hDevice As Long) As Boolean

功能：列表系统中 PCI7062A 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 PCI7062A 设备的配置情况。

相关函数： CreateDevice　　GetDeviceCount
ListDeviceDlg　　ReleaseDevice

♦ **释放设备对象所占的系统资源及设备对象**

函数原型：

Visual C++:

BOOL ReleaseDevice(HANDLE hDevice)

Visual Basic:

Declare Function ReleaseDevice Lib “PCI7062A” (ByVal hDevice As Long) As Boolean

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。

3.3 设备功能控制函数原型说明

♦ **获取设备工作状态**

函数原型：

Visual C++:

```
BOOL GetDeviceSts ( HANDLE hDevice,
                    PBOOl pbStatus)
```

Visual Basic:

```
Declare Function GetDeviceSts Lib “PCI7062A” (ByVal hDevice As Long, _
                                              ByVal pbStatus As Long) As Boolean
```

功能：获取设备工作状态。

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

PBStatus 设备状态，当返回为 TRUE 时表示可以读取有效数据。否则数据无效。

返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以调用 GetLastErrorEx 函数取得当前错误码。

相关函数： SetDeviceDigits CreateDevice ReleaseDevice
SetFilterSpeed ReadDeviceData GetDeviceSts
SetDeviceInputRange SetDeviceFuntion ReadBaseFreCNT
ReadMeasFreCNT ReadMeasRatioCNT

♦ **万用表功能档选择**

函数原型：

Visual C++:

```
BOOL SetDeviceFunction( HANDLE hDevice,
                        LONG lFunction)
```

Visual Basic:

Declare Function SetDeviceFunction Lib “PCI7062A” (ByVal hDevice As Long, _
ByVal lFunction As Long) As Boolean

功能：万用表功能档选择。

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

IFunction 功能档选择。

返回值：若成功，返回 TRUE，否则返回 FALSE。用户可以调用 GetLastErrorEx 函数取得当前错误码。

相关函数： SetDeviceDigits CreateDevice ReleaseDevice SetFilterSpeed ReadDeviceData GetDeviceSts SetDeviceInputRange SetDeviceFuntion ReadBaseFreCNT ReadMeasFreCNT ReadMeasRatioCNT

♦ 读取设备采集数据

函数原型：

Visual C++:

BOOL ReadDeviceValue(HANDLE hDevice,
double* pdValue)

Visual Basic:

Declare Function ReadDeviceValue Lib “PCI7062A” (ByVal hDevice As Long, _
ByRef pdValue As Double) As Boolean

功能：读取设备采集数据。

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

pdValue 采集数据。

返回值：若成功，返回 TRUE，否则返回 FALSE。用户可以调用 GetLastErrorEx 函数取得当前错误码。

相关函数： SetDeviceDigits CreateDevice ReleaseDevice SetFilterSpeed ReadDeviceData GetDeviceSts SetDeviceInputRange SetDeviceFuntion ReadBaseFreCNT ReadMeasFreCNT ReadMeasRatioCNT

♦ 读取基准频率计数值

函数原型：

Visual C++:

BOOL ReadBaseFreCNT(HANDLE hDevice,
PLONG pluBaseFreCNTValue)

Visual Basic:

Declare Function ReadBaseFreCNT Lib “PCI7062A” (ByVal hDevice As Long, _
ByRef pluBaseFreCNTValue As Long) As Boolean

功能：读取基准频率计数值。

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

pluBaseFerCNTValue　基准频率计数值。

返回值：若成功，返回 TRUE，否则返回 FALSE。用户可以调用 GetLastErrorEx 函数取得当前错误码。

相关函数： SetDeviceDigits　CreateDevice　ReleaseDevice　SetFilterSpeed　ReadDeviceData　GetDeviceSts　SetDeviceInputRange　SetDeviceFuntion　ReadBaseFreCNT　ReadMeasFreCNT　ReadMeasRatioCNT

♦ **读取被测信号计数值**

函数原型：

Visual C++:

```
BOOL ReadMeasFreCNT( HANDLE hDevice,
                     PLONG pluMeasFreCNTValue)
```

Visual Basic:

```
Declare Function ReadMeasFreCNTLib "PCI7062A" (ByVal hDevice As Long, _
                                               ByRef pluMeasFreCNTValue As Long) As Boolean
```

功能：被测信号计数值。

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

pluMeasFreCNTValue　被测信号计数值。

返回值：若成功，返回 TRUE，否则返回 FALSE。用户可以调用 GetLastErrorEx 函数取得当前错误码。

相关函数： SetDeviceDigits　CreateDevice　ReleaseDevice　SetFilterSpeed　ReadDeviceData　GetDeviceSts　SetDeviceInputRange　SetDeviceFuntion　ReadBaseFreCNT　ReadMeasFreCNT　ReadMeasRatioCNT

♦ **读取被测信号高电平脉冲的计数值**

函数原型：

Visual C++:

```
BOOL ReadMeasRatioCNT( HANDLE hDevice,
                       PLONG pluMeasRatioCNTValue)
```

Visual Basic:

```
Declare Function ReadMeasRatioCNT Lib "PCI7062A" (ByVal hDevice As Long, _
                                                  ByRef pluMeasRatioCNTValue As Long) As Boolean
```

功能：读取被测信号高电平脉冲的计数值。

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

pluMeasRatioCNTValue　被测信号高电平脉冲的计数值。

返回值：若成功，返回 TRUE，否则返回 FALSE。用户可以调用 GetLastErrorEx 函数取得当前错误码。

相关函数： SetDeviceDigits　CreateDevice　ReleaseDevice　SetFilterSpeed　ReadDeviceData　GetDeviceSts　SetDeviceInputRange　SetDeviceFuntion　ReadBaseFreCNT

ReadMeasFreCNT　　ReadMeasRatioCNT

◆ **设置输入量程**

函数原型：

Visual C++:

```
BOOL SetDeviceInputRange( HANDLE hDevice,
                          LONG lInputRange)
```

Visual Basic:

```
Declare Function SetDeviceInputRange Lib "PCI7062A" (ByVal hDevice As Long, _
                                                     ByVal lInputRange As Long) As Boolean
```

功能：设置输入量程。

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

lInputRange　输入量程。

返回值：若成功，返回 TRUE，否则返回 FALSE。用户可以调用 GetLastErrorEx 函数取得当前错误码。

相关函数： SetDeviceDigits　CreateDevice　ReleaseDevice
SetFilterSpeed　ReadDeviceData　GetDeviceSts
SetDeviceInputRange　SetDeviceFuntion　ReadBaseFreCNT
ReadMeasFreCNT　ReadMeasRatioCNT

3.4 设备校准函数原型说明

◆ **读出校准数据**

函数原型：

Visual C++:

```
BOOL ReadCalibrationData( HANDLE hDevice,
                          LONG lFunction,
                          LONG lInputRange,
                          LONG lCalMode,
                          PLONG lCalData)
```

Visual Basic:

```
Declare Function ReadCalibrationData Lib "PCI7062A" (ByVal hDevice As Long, _
                                                     ByVal  lFunction As Long, _
                                                     ByVal  lInputRange As Long, _
                                                     ByVal  lCalMode As Long, _
                                                     ByRef  lCalData As Long) As Boolean
```

功能：读出校准数据。

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

lFunction 功能档选择。此处只有电压、电流和电阻档选择。

lInputRange　量程选择。

LCalMode 校准模式，0 为零点校准，1 为满度校准。

LCalData 校准值的范围为 0-16777216.

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice ReleaseDevice ReadCalibrationData WriteCalibrationData

♦ **写入校准数据**

函数原型：

Visual C++:

```
BOOL WriteCalibrationData( HANDLE hDevice,
                           LONG lFunction,
                           LONG lInputRange,
                           LONG lCalMode,
                           LONG lCalData)
```

Visual Basic:

```
Declare Function WriteCalibrationData Lib "PCI7062A" (ByVal hDevice As Long, _
                                                      ByVal  lFunction As Long, _
                                                      ByVal  lInputRange As Long, _
                                                      ByVal  lCalMode As Long, _
                                                      ByVal  lCalData As Long) As Boolean
```

功能：写入校准数据。

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

lFunction 功能档选择。此处只有电压、电流和电阻档选择。

lInputRange 量程选择。

LCalMode 校准模式，0 为零点校准，1 为满度校准。

LCalData 校准值的范围为 0-16777216.

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice ReleaseDevice ReadCalibrationData WriteCalibrationData

♦ **写入交流校准时频率修正增益码值**

函数原型：

Visual C++:

```
BOOL WriteACVFreGainsData( HANDLE hDevice,
                           LONG lInputRange,
                           LONG lFreGains)
```

Visual Basic:

```
Declare Function WriteACVFreGainsData Lib "PCI7062A" (ByVal hDevice As Long, _
                                                      ByVal lInputRange As Long, _
                                                      ByVal lFreGains As Long) As Boolean
```

功能：写入交流校准时频率修正增益码值

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

lInputRange 量程选择。

lFerGains 频率修正增益码值。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice ReleaseDevice ReadCalibrationData WriteCalibrationData

◆ **读出交流校准时频率修正增益码值**

函数原型：

Visual C++:

```
BOOL ReadACVFreGainsData( HANDLE hDevice,
                          LONG lInputRange,
                          PLONG lFreGains)
```

Visual Basic:

```
Declare Function ReadACVFreGainsData Lib "PCI7062A" (ByVal hDevice As Long, _
                                                     ByVal lInputRange As Long, _
                                                     ByRef  lFreGains As Long) As Boolean
```

功能：读出交流校准时频率修正增益码值

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

lInputRange 量程选择。

lFerGains 频率修正增益码值。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice ReleaseDevice ReadCalibrationData WriteCalibrationData

4 功能参数选择结构

函数 PCI7062A_SetDeviceFunction 中 lFunction 参数所使用功能档选择如下：

常量名	常量值	功能定义
PCI7062A_FUNCTIONG_DCV	0x00	直流电压档
PCI7062A_FUNCTIONG_ACV	0x01	交流电压档
PCI7062A_FUNCTIONG_DCI	0x02	直流电流档
PCI7062A_FUNCTIONG_ACI	0x03	交流电流档
PCI7062A_FUNCTIONG_2W	0x04	2 线电阻档
PCI7062A_FUNCTIONG_4W	0x05	4 线电阻档
PCI7062A_FUNCTIONG_FRE	0x06	频率档
PCI7062A_FUNCTIONG_CAP	0x07	电容档
PCI7062A_FUNCTIONG_DIODE	0x08	二极管档
PCI7062A_FUNCTIONG_ONOFF	0x09	通断档

直流电压量程选择

常量名	常量值	功能定义
PCI7062A_DCV_INPUT_N0_200MV	0x00	0-200mV 量程
PCI7062A_DCV_INPUT_N0_2000MV	0x01	0-2000mV 量程
PCI7062A_DCV_INPUT_N0_20000MV	0x02	0-20000mV 量程
PCI7062A_DCV_INPUT_N0_200000MV	0x03	0-200000mV 量程
PCI7062A_DCV_INPUT_N0_300000MV	0x04	0-300000mV 量程

交流电压量程选择

常量名	常量值	功能定义
PCI7062A_ACV_INPUT_N0_200MV	0x00	0-200mV 量程
PCI7062A_ACV_INPUT_N0_2000MV	0x01	0-2000mV 量程
PCI7062A_ACV_INPUT_N0_20000MV	0x02	0-20000mV 量程
PCI7062A_ACV_INPUT_N0_200000MV	0x03	0-200000mV 量程
PCI7062A_ACV_INPUT_N0_300000MV	0x04	0-300000mV 量程

直流电流量程选择

常量名	常量值	功能定义
PCI7062A_DCI_INPUT_N0_2MA	0x00	0-2mA 量程
PCI7062A_DCI_INPUT_N0_20MA	0x01	0-20mA 量程
PCI7062A_DCI_INPUT_N0_200MA	0x02	0-200mA 量程
PCI7062A_DCI_INPUT_N0_1A	0x03	0-1A 量程

交流电流量程选择

常量名	常量值	功能定义
PCI7062A_ACI_INPUT_N0_20MA	0x00	0-20mA 量程
PCI7062A_ACI_INPUT_N0_200MA	0x01	0-200mA 量程
PCI7062A_ACI_INPUT_N0_1A	0x02	0-1A 量程

2W 电阻量程选择

常量名	常量值	功能定义
PCI7062A_2W_INPUT_N0_P200	0x00	0-200Ω量程
PCI7062A_2W_INPUT_N0_P2K	0x01	0-2KΩ量程
PCI7062A_2W_INPUT_N0_P20K	0x02	0-20KΩ量程
PCI7062A_2W_INPUT_N0_P200K	0x03	0-200KΩ量程
PCI7062A_2W_INPUT_N0_P1M	0x04	0-1MΩ量程
PCI7062A_2W_INPUT_N0_P10M	0x05	0-10MΩ量程
PCI7062A_2W_INPUT_N0_P100M	0x06	0-100MΩ量程

4W 电阻量程选择

常量名	常量值	功能定义
PCI7062A_4W_INPUT_N0_P200	0x00	0-200Ω量程
PCI7062A_4W_INPUT_N0_P2K	0x01	0-2KΩ量程
PCI7062A_4W_INPUT_N0_P20K	0x02	0-20KΩ量程
PCI7062A_4W_INPUT_N0_P200K	0x03	0-200KΩ量程
PCI7062A_4W_INPUT_N0_P1M	0x04	0-1MΩ量程
PCI7062A_4W_INPUT_N0_P10M	0x05	0-10MΩ量程
PCI7062A_4W_INPUT_N0_P100M	0x06	0-100MΩ量程

电容量程选择

常量名	常量值	功能定义
PCI7062A_CAP_INPUT_N0_P2NF	0x00	0-2nF 量程
PCI7062A_CAP_INPUT_N0_P20NF	0x01	0-20nF 量程
PCI7062A_CAP_INPUT_N0_P200NF	0x02	0-200nF 量程

PCI7062A_CAP_INPUT_N0_P2UF	0x03	0-2uF 量程
PCI7062A_CAP_INPUT_N0_P20UF	0x04	0-20uF 量程
PCI7062A_CAP_INPUT_N0_P200UF	0x05	0-200uF 量程

二极管档量程

常量名	常量值
PCI7062A_DIODE_INPUT_N0_2V	0x00

万用表分辨率 lDigits 参数使用选项

常量名	常量值	功能定义
PCI7062A_DIGITS_5_12	0x00	5.5Digists
PCI7062A_DIGITS_4_12	0x01	4.5Digists
PCI7062A_DIGITS_3_12	0x02	3.5Digists

直流输入阻抗 lImpedance 参数使用选项

常量名	常量值	功能定义
PCI7062A_DC_IMPEDANCE_10M	0x00	直流输入阻抗为 10MΩ
PCI7062A_DC_IMPEDANCE_20M	0x01	直流输入阻抗为大于 20MΩ

滤波速度 lSpeed 参数使用选项

常量名	常量值	功能定义
PCI7062A_AC_FILTERSPEED_SLO	0x00	交流慢速滤波
PCI7062A_AC_FILTERSPEED_QUICK	0x01	交流快速滤波

触发模式 lTrigMode 参数使用选项

常量名	常量值	功能定义
PCI7062A_TRIGMODE_AUTO	0x00	自动触发
PCI7062A_TRIGMODE_EXT	0x01	外部触发
PCI7062A_TRIGMODE_SINGLE	0x02	单次触发

触发方向 lTrigDir 参数使用选项

常量名	常量值	功能定义
PCI7062A_TRIGDIR_NEGATIVE	0x00	负向触发(下降沿触发)
PCI7062A_TRIGDIR_POSITIVE	0x01	正向触发(上升沿触发)

外部输出极性 lPolarity 参数使用选项

常量名	常量值	功能定义
PCI7062A_POLARITY_HIGH	0x00	高电平脉冲输出
PCI7062A_POLARITY_LOW	0x01	低电平脉冲输出

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（每个函数省略了前缀“PCI7062A_”）

函数名	函数功能	备注
① PCIe 总线内存映射寄存器操作函数		
GetDeviceBar	取得指定的设备寄存器组 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 附加操作函数		
CreateSystemEvent	创建系统内核事件对象	用于线程同步或中断
ReleaseSystemEvent	释放系统内核事件对象	

5.2 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 pbPCIBar[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。

pbPCIBar[6]　返回 PCI BAR 所有地址,具体 PCI BAR 中有多少可用地址请看硬件说明书。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数;：CreateDevice　　GetDeviceBar

♦ **往指定寄存器空间位置写入单字节数据**

函数原型:

Visual C++:

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

功能：往指定寄存器空间位置写入单字节数据。

参数：

hDevice　设备对象句柄，它应由 CreateDevice 创建。

pbLinearAddr　指定寄存器的线性基地址，它的等于 GetDeviceAddr 中的 pbLinearAddr 参数返回值。

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 往指定地址写入单字节数据。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数;：CreateDevice　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
:
```

◆ **写双字节数据**

函数原型:

Visual C++:

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

功能：写双字节数据。

参数：

hDevice　设备对象句柄，它应由 CreateDevice 创建。

pbLinearAddr　指定寄存器的线性基地址，它的等于 GetDeviceAddr 中的 pbLinearAddr 参数返回值。

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 往指定地址写入单字节数据。

返回值：无。

相关函数： CreateDevice　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
    :
```

◆ **写四字节数据**

函数原型:

Visual C++:

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

功能：写四字节数据。

参数：

hDevice　设备对象句柄，它应由 CreateDevice 创建。

pbLinearAddr　指定寄存器的线性基地址，它的等于 GetDeviceAddr 中的 pbLinearAddr 参数返回值。

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　往指定地址写入单字节数据。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice　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
    :
```

◆ **读入单字节数据**

函数原型:

Visual C++:

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

功能：读入单字节数据。

参数：

hDevice　设备对象句柄，它应由 CreateDevice 创建。

pbLinearAddr　指定寄存器的线性基地址，它的等于 GetDeviceAddr 中的 pbLinearAddr 参数返回值。

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　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
    :

♦ **读入双字节数据**

函数原型:

Visual C++:

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

功能：读入双字节数据。

参数：

hDevice　设备对象句柄，它应由 CreateDevice 创建。

pbLinearAddr　指定寄存器的线性基地址，它的等于 GetDeviceAddr 中的 pbLinearAddr 参数返回值。

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　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
    :

♦ **读入四字节数据（其余同上）**

函数原型:

Visual C++:

ULONG ReadRegisterULong (HANDLE hDevice,
                         __int64 pbLinearAddr,

ULONG OffsetBytes)

功能：读入四字节数据。

参数：

hDevice　设备对象句柄，它应由 CreateDevice 创建。

pbLinearAddr　指定寄存器的线性基地址，它的等于 GetDeviceAddr 中的 pbLinearAddr 参数返回值。

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　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 PCI 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　IO 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++:

```
BOOL WritePortByte (HANDLE hDevice,
                    __int64 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 GetDeviceAddr WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord ReadPortULong ReleaseDevice

- **以双字(16Bit)方式写 I/O 端口**

函数原型：

Visual C++:

```
BOOL WritePortWord (HANDLE hDevice,
                    UINT 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 GetDeviceAddr WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord ReadPortULong ReleaseDevice

- **以四字节(32Bit)方式写 I/O 端口**

函数原型：

Visual C++:

```
BOOL WritePortULong(HANDLE hDevice,
                    UINT 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 GetDeviceAddr WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord ReadPortULong ReleaseDevice

- **以单字节(8Bit)方式读 I/O 端口**

函数原型：

Visual C++:

```
BYTE ReadPortByte( HANDLE hDevice,
                   UINT 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　GetDeviceAddr　WritePortByte
WritePortWord　WritePortULong　ReadPortByte
ReadPortWord　ReadPortULong　ReleaseDevice

- **以双字节(16Bit)方式读 I/O 端口**

函数原型：
Visual C++:

```
WORD ReadPortWord(HANDLE hDevice,
                  UINT 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　GetDeviceAddr　WritePortByte
WritePortWord　WritePortULong　ReadPortByte
ReadPortWord　ReadPortULong　ReleaseDevice

- **以四字节(32Bit)方式读 I/O 端口**

函数原型：
Visual C++:

```
ULONG ReadPortULong(HANDLE hDevice,
                    UINT 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　GetDeviceAddr　WritePortByte
WritePortWord　WritePortULong　ReadPortByte
ReadPortWord　ReadPortULong　ReleaseDevice

5.4　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 "PCI7062A"( ) As Long
```

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)

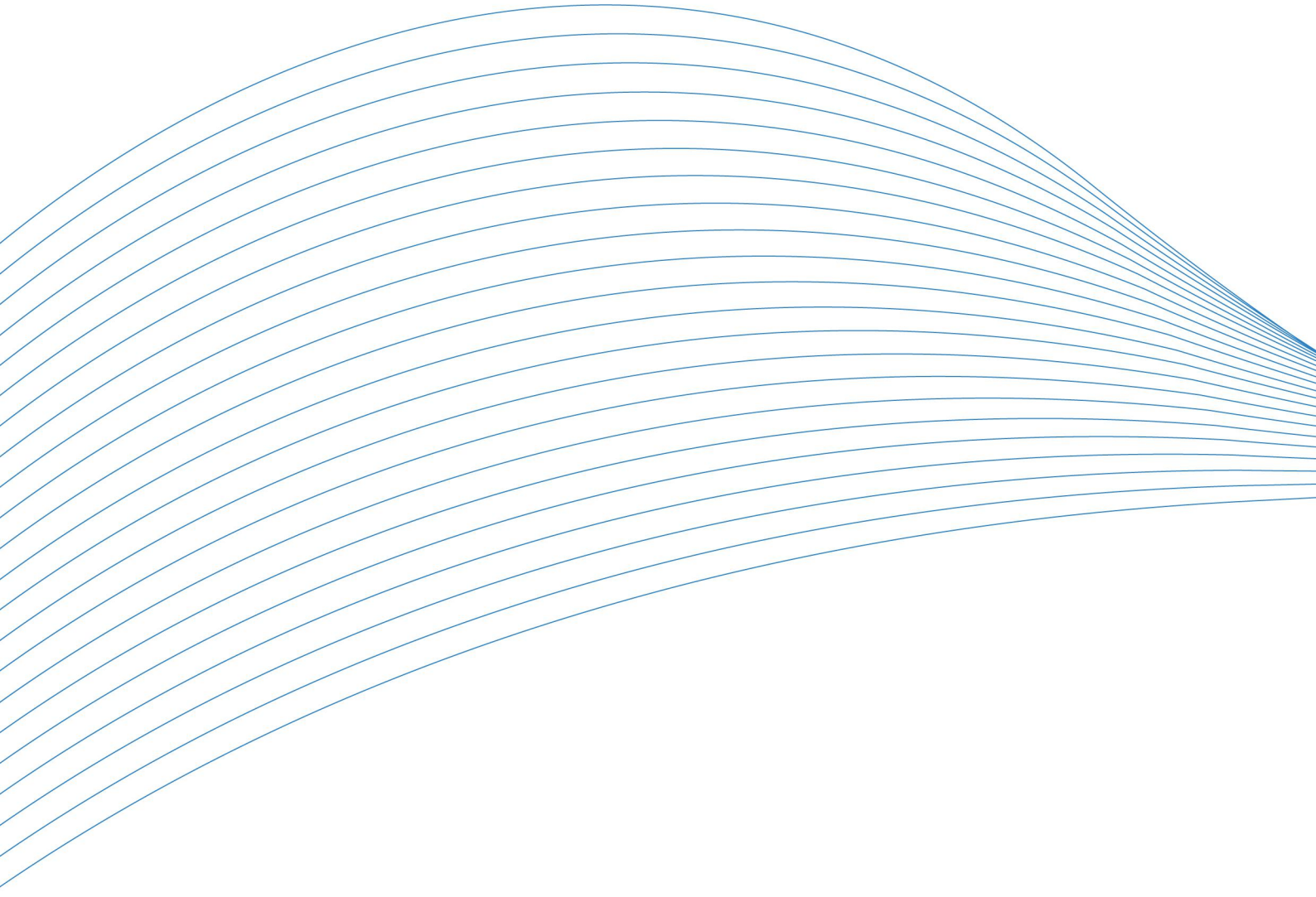
Visual Basic:

Declare Function ReleaseSystemEvent Lib “PCI7062A”(ByVal Event As Long) As Boolean

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。



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