PCI2513 光电隔离
DI/DO 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 PCIxxxx_ 则被省略。如 PCI2513_CreateDevice 则写为 CreateDevice。

二、函数名及参数中各种关键字缩写

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			

第二章 使用纲要

第一节、使用上层用户函数，高效、简单

如果您只关心通道及频率等基本参数，而不必了解复杂的硬件知识和控制细节，那么我们强烈建议您使用上层用户函数，它们就是几个简单的形如 Win32 API 的函数，具有相当的灵活性、可靠性和高效性。而底层用户函数如WritePortByte、ReadPortByte……则是满足了解硬件知识和控制细节、且又需要特殊复杂控制的用户。但不管怎样，我们强烈建议您使用上层函数（在这些函数中，您见不到任何设备地址、寄存器端口、中断号等物理信息，其复杂的控制细节完全封装在上层用户函数中。）对于上层用户函数的使用，您基本上不必参考硬件说明书，除非您需要知道板上 D 型插座等管脚分配情况。

第二节、如何管理 PCI 设备

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

第三节、如何实现开关量的简便操作

当您有了 hDevice 设备对象句柄后，便可用SetDeviceDO函数实现开关量的输出操作，其各路开关量的输出状态由其 bDOISts[16]中的相应元素决定。

第四节、哪些函数对您不是必须的

公共函数如CreateFileObject等一般来说都是辅助性函数，除非您要使用存盘功能。而WritePortByte，WritePortWord， WritePortULong， ReadPortByte， ReadPortWord， ReadPortULong则对 PCI 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 PCI 即插即用设备操作函数接口介绍

由于我公司的设备应用于各种不同的领域，有些用户可能根本不关心硬件设备的控制细节，只关心首末通道、采样频率等，然后就能通过一两个简易的采集函数便能轻松得到所需要的数据。这方面的用户我们称之为上层用户。那么还有一部分用户不仅对硬件控制熟悉，而且由于应用对象的特殊要求，则要直接控制设备的每一个端口，这是一种复杂的工作，但又是必须的工作，我们则把这一群用户称之为底层用户。因此总的看来，上层用户要求简单、快捷，他们最希望在软件操作上所要面对的全是他们最关心的问题，而关于设备的物理地址、端口分配及功能定义等复杂的硬件信息则与上层用户无任何关系。那么对于底层用户则不然。他们不仅要关心设备的物理地址，还要关心虚拟地址、端口寄存器的功能分配，甚至每个端口的 Bit 位都要了如指掌，看起来这是一项相当复杂、繁琐的工作。但是这些底层用户一旦使用我们提供的技术支持，则不仅可以让您不必熟悉 PCI 总线复杂的控制协议，同是还可以省掉您许多繁琐的工作。

第一节、设备驱动接口函数总列表（每个函数省略了前缀“PCI2513_”）

函数名	函数功能	备注
① 设备对象操作函数		
CreateDevice	创建设备对象(用设备逻辑号)	
GetDeviceCount	取得同一种 PCI 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 PCI 设备的各种配置	上层及底层用户
ReleaseDevice	关闭设备，且释放 PCI 总线设备对象	
② 开关量函数		
SetDeviceDO	设置数字量输出状态	上层用户
GetDeviceDI	取得数字量输入状态	上层用户
③ PWM 函数		
SetPWMCycle	设置 0 路周期函数	上层用户
SetPWMEnable	PWM 使能函数	上层用户
④ 中断函数		
InitDeviceInt	初始化中断	上层用户
GetIntSrc	取得中断个数	上层用户
ResetIntSrc	复位中断	上层用户
ReleaseDeviceInt	释放中断资源	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\PCI2513\INCLUDE\PCI2513.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 PCI2513.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 “PCI2513.H”

第二节、设备对象操作函数原型说明

♦ **创建设备对象函数**

函数原型：

Visual C++ ：

HANDLE CreateDevice (int DeviceID = 0)

功能：该函数使用逻辑号创建设备对象

参数：DeviceID 设备 ID(Identifier)标识号。当向同一个 Windows 系统中加入若干相同类型的设备时，系统将以该设备的“基本名称”与 DeviceID 标识值为名称后缀的标识符来确认和管理该设备。默认值为 0。

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

相关函数： ReleaseDevice

♦ **取得同一种 PCI 设备的总台数**

函数原型：

Visual C++ :

int GetDeviceCount (HANDLE hDevice)

功能：取得 PCI2513 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 PCI2513 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++ :

int GetDeviceCurrentID (HANDLE hDevice)

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

♦ **用对话框控件列表同一种 PCI 设备各种配置信息**

函数原型：

Visual C++ :

BOOL ListDeviceDlg (HANDLE hDevice)

功能：列表系统中 PCI2513 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 PCI2513 设备的配置情况。

相关函数： CreateDevice ReleaseDevice

♦ **释放设备对象所占的系统资源及设备对象**

函数原型：

Visual C++ :

BOOL ReleaseDevice(HANDLE hDevice)

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、DI/DO 数字开关量输入输出简易操作函数原型说明

♦ **设置数字量输出状态**

函数原型：

Visual C++ :

```
BOOL SetDeviceDO(HANDLE hDevice,
                 BYTE bDOSts[16])
```

功能：负责将 PCI 设备上的 DO0～DO15 输出开关量置成相应的状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDOSts 十六路开关量输出状态的参数结构，共有 16 个成员变量，分别对应于 DO0～DO15 路开关量输出状态位。比如置 bDOSts[0]为“1”则使 0 通道处于“开”状态，若为“0”则置 0 通道为“关”状态。其他同理。请注意，在实际执行这个函数之前，必须对这个参数结构的 DO0 至 DO15 共 16 个成员变量赋初值，其值必须为“1”或“0”。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice GetDeviceDI ReleaseDevice

♦ **取得数字量输入状态**

函数原型：

Visual C++ :

```
BOOL GetDeviceDI(HANDLE hDevice,
                 BYTE bDISts [16])
```

功能：负责将 PCI 设备上的 DI0～DI15 输入开关量状态读入内存。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

bDISts 十六路开关量输入状态的参数结构，共有 16 个成员变量，分别对应于 DI0～DI15 路开关量输入状态位。如果 bDISts [0]为“1”则使 0 通道处于开状态，若为“0”则 0 通道为关状态。其他同理。

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

相关函数： CreateDevice SetDeviceDO ReleaseDevice

❖ **以上函数调用一般顺序**

① CreateDevice

② SetDeviceDO(或GetDeviceDI，当然这两个函数也可同时进行)

③ ReleaseDevice

用户可以反复执行第②步，以进行数字 I/O 的输入输出。

第四节、PWM 函数原型说明

♦设置 0 路周期函数

函数原型：

```
BOOL SetPWMCycle (HANDLE hDevice,
                  WORD wLoLevCyc,
                  WORD wHiLevCyc,
                   int nChannel)
```

♦PWM 使能函数

函数原型：

```
BOOL SetPWMEnable (HANDLE hDevice,
                   BOOL bEnableCH0,
                    BOOL bEnableCH1)
```

第五节、中断函数原型说明

♦ 初始化中断

函数原型：

Visual C++ :

```
BOOL InitDeviceInt (HANDLE hDevice,
                    PPCI2513_INT_PARA pIntPara)
```

功能：它负责初始化设备对象的硬件中断的方式工作。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

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，并加以分析。

相关函数： CreateDevice　InitDeviceInt　SetCOSINTCH　GetDeviceIntCount　GetDeviceIntSrc　ReleaseDeviceInt　ReleaseDevice

♦ 取得中断个数

函数原型：

Visual C++ :

```
BOOL GetIntSrc(HANDLE hDevice,
               BOOL bIntSts[16])
```

功能：在中断初始化后，用它取得中断个数。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

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，并加以分析。

相关函数： CreateDevice　InitDeviceInt　SetCOSINTCH　GetDeviceIntCount　GetDeviceIntSrc　ReleaseDeviceInt　ReleaseDevice

♦ 复位中断

函数原型：

Visual C++ :

```
BOOL ResetIntSrc (HANDLE hDevice)
```

功能：在中断初始化后，用它取得复位中断。

参数：hDevice 设备对象句柄，它应由CreateDevice创建。

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，并加以分析。

相关函数： CreateDevice　InitDeviceInt　SetCOSINTCH　GetDeviceIntCount　GetDeviceIntSrc　ReleaseDeviceInt　ReleaseDevice

◆ **释放中断资源**

函数原型：

Visual C++ :

BOOL ReleaseDeviceInt (HANDLE hDevice)

功能：释放中断资源。

参数：hDevice 设备对象句柄，它应由CreateDevice创建。

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，并加以分析。

相关函数： CreateDevice InitDeviceInt SetCOSINTCH GetDeviceIntCount GetDeviceIntSrc ReleaseDeviceInt ReleaseDevice

第五章　上层用户函数接口应用实例

如果您想快速的了解驱动程序的使用方法和调用流程，以最短的时间建立自己的应用程序，那么我们强烈建议您参考相应的简易程序。此种程序属于工程级代码，可以直接打开不用作任何配置和代码修改即可编译通过，运行编译链接后的可执行程序，即可看到预期效果。

如果您想了解硬件的整体性能、精度、采样连续性等指标以及波形显示、数据存盘与分析、历史数据回放等功能，那么请参考高级演示程序。特别是许多不愿意编写任何程序代码的用户，您可以使用高级程序进行采集、显示、存盘等功能来满足您的要求。甚至可以用我们提供的专用转换程序将高级程序采集的存盘文件转换成相应格式，即可在 Excel、MatLab 第三方软件中分析数据（此类用户请最好选用通过 Visual C++制作的高级演示系统）。

第一节、简易程序演示说明

一、怎样使用GetDeviceDI函数进行开关量输入操作

其详细应用实例及正确代码请参考 Visual C++简易演示系统及源程序，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程(主要参考 PCI2513.h 和 Sys.cpp)。

[程序]丨**[阿尔泰测控演示系统]**丿**[PCI2513 16 路开关量输入输出卡]**丨**[Microsoft Visual C++]**丿[简易代码演示]丿[DI 开关量演示源程序]

其默认存放路径为：系统盘\ART\PCI2513\SAMPLES\VC\SIMPLE\DI

二、怎样使用SetDeviceDO函数进行开关量输出操作

其详细应用实例及正确代码请参考 Visual C++简易演示系统及源程序，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程(主要参考 PCI2513.h 和 Sys.cpp)。

[程序]丨**[阿尔泰测控演示系统]**丿**[PCI2513 16 路开关量输入输出卡]**丨**[Microsoft Visual C++]**丿[简易代码演示]丿[DO 开关量演示源程序]

其默认存放路径为：系统盘\ART\PCI2513\SAMPLES\VC\SIMPLE\DO

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 PCI2513.h 和 Sys.cpp)。

[程序]丨**[阿尔泰测控演示系统]**丿**[PCI2513 16 路开关量输入输出卡]**丨**[Microsoft Visual C++]**丿[高级代码演示]

其默认存放路径为：系统盘\ART\PCI2513\SAMPLES\VC\ADVANCED

其他语言的演示可以用上面类似的方法找到。

第六章 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“PCI2513_”）

函数名	函数功能	备注
① PCI 总线内存映射寄存器操作函数		
GetDeviceBar	取得指定的指定设备寄存器组 BAR 地址	底层用户
GetDevVersion	获取设备固件及程序版本	底层用户
② ISA 总线 I/O 端口操作函数		
WritePortByte	以字节(8Bit)方式写 I/O 端口	用户程序操作端口
WritePortWord	以字(16Bit)方式写 I/O 端口	用户程序操作端口
WritePortULong	以无符号双字(32Bit)方式写 I/O 端口	用户程序操作端口
ReadPortByte	以字节(8Bit)方式读 I/O 端口	用户程序操作端口
ReadPortWord	以字(16Bit)方式读 I/O 端口	用户程序操作端口
ReadPortULong	以无符号双字(32Bit)方式读 I/O 端口	用户程序操作端口
③ 创建 Visual Basic 子线程，线程数量可达 32 个以上		
CreateSystemEvent	创建系统内核事件对象	用于线程同步或中断
ReleaseSystemEvent	释放系统内核事件对象	

第二节、PCI 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   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 PCI BAR 所有地址。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice ReleaseDevice

- **获取设备固件及程序版本**

函数原型：

Visual C++ :

```
BOOL GetDevVersion ( HANDLE hDevice,
                     PULONG pulFmwVersion,
                     PULONG pulDriverVersion)
```

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 ReleaseDevice

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++ :

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

  pbPort 设备的 I/O 端口号。

  Value 写入由 pbPort 指定端口的值。

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

  相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

- **以双字(16Bit)方式写 I/O 端口**

  函数原型：

  Visual C++ :

```
BOOL WritePortWord (HANDLE hDevice,
                    __int64 pbPort,
                    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。

  pbPort 设备的 I/O 端口号。

  Value 写入由 pbPort 指定端口的值。

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

  相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

- **以四字节(32Bit)方式写 I/O 端口**

  函数原型：

  Visual C++ :

```
BOOL WritePortULong(HANDLE hDevice,
                    __int64 pbPort,
                    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。

  pbPort 设备的 I/O 端口号。

  Value 写入由 pbPort 指定端口的值。

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

  相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

♦ **以单字节(8Bit)方式读 I/O 端口**

函数原型：

Visual C++ :

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

功能：以单字节(8Bit)方式读 I/O 端口。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pbPort 设备的 I/O 端口号。

返回值：返回由 pbPort 指定的端口的值。

相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

♦ **以双字节(16Bit)方式读 I/O 端口**

函数原型：

Visual C++ :

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

功能：以双字节(16Bit)方式读 I/O 端口。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pbPort 设备的 I/O 端口号。

返回值：返回由 pbPort 指定的端口的值。

相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

♦ **以四字节(32Bit)方式读 I/O 端口**

函数原型：

Visual C++ :

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

功能：以四字节(32Bit)方式读 I/O 端口。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pbPort 设备的 I/O 端口号。

返回值：返回由 pbPort 指定端口的值。

相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

第四节、线程操作函数原型说明

♦ **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。