ACTS6010 可编程电阻卡

驱动程序使用手册

V6.00.00

ART Technology

前言

免责说明

订购产品前，请向厂家或经销商详细了解产品性能是否符合您的需求。

正确的运输、储存、组装、装配、安装、调试、操作和维护是产品安全、正常运行的前提。本公司对于任何因安装、使用不当而导致的直接、间接、有意或无意的损坏及隐患概不负责。

安全使用小常识

1.在使用产品前，请务必仔细阅读产品使用手册；

2.对未准备安装使用的产品，应做好防静电保护工作(最好放置在防静电保护袋中，不要将其取出)；

3.在拿出产品前，应将手先置于接地金属物体上，以释放身体及手中的静电，并佩戴静电手套和手环，要养成只触及其边缘部分的习惯；

4.为避免人体被电击或产品被损坏，在每次对产品进行拔插或重新配置时，须断电；

5.在需对产品进行搬动前，务必先拔掉电源；

6.对整机产品，需增加/减少板卡时，务必断电；

7.当您需连接或拔除任何设备前，须确定所有的电源线事先已被拔掉；

8.为避免频繁开关机对产品造成不必要的损伤，关机后，应至少等待 30 秒后再开机。

目　录

1 版权信息与命名约定

1.1 版权信息

本软件产品及相关套件均属北京阿尔泰科技发展有限公司所有，其产权受国家法律绝对保护，除非本公司书面允许，其他公司、单位、我公司授权的代理商及个人不得非法使用和拷贝，否则将受到国家法律的严厉制裁。您若需要我公司产品及相关信息请及时与当地代理商联系或直接与我们联系，我们将热情接待。

1.2 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 ACTS6010_则被省略。如 ACTS6010_DEV_Create 则写为 DEV_Create。

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。

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 InitDeviceAD、StartDeviceAD 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，您基本上不必参考硬件说明书，除非您需要知道板上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 DEV_Create 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 InitDeviceAD 可以使用 hDevice 句柄以程序查询方式初始化设备的 AD 部件，ReadDeviceAD-函数可以用 hDevice 句柄实现对 AD 数据的采样读取等。最后可以通过 DEV_Release 将 hDevice 释放掉。

2.3 哪些函数对您不是必须的

如果您使用上层用户函数访问设备，那么 GetDeviceBar，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 PCIe 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 ACTS 设备操作函数接口介绍

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

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

3.1 设备驱动接口函数总列表（每个函数省略了前缀“ACTS6010_”）

函数名	函数功能	备注
① 设备对象操作函数		
DEV_Create	创建设备对象	上层及底层用户
DEV_GetCount	取得同一种设备的总台数	上层及底层用户
DEV_GetCurrentIdx	获取设备的逻辑 ID 号和物理 ID 号	上层及底层用户
DEV_Reset	复位设备	上层及底层用户
DEV_GetMainInfo	获得板卡主要信息	上层及底层用户
DEV_ListDlg	列表所有同一种设备的各种配置	上层及底层用户
DEV_Release	关闭设备，且释放总线设备对象	上层及底层用户
② 电阻操作函数		
RES_DisconnectAll	断开所有继电器连接	上层用户
RES_SetSettlingtime	设置建立时间	上层用户
RES_GetSettlingtime	获取建立时间	上层用户
RES_GetStatus	回读板卡状态信息	上层用户
RES_SetResistance	设置电阻值	上层用户
RES_GetResistance	获取电阻值	上层用户
RES_SetPSRelayState	设置通路短路继电器状态	上层用户
RES_GetPSRelayState	获取通路短路继电器状态	上层用户

③设备信息		
DEV_SetPhysIdx	设置物理序号	上层用户
DEV_GetVersion	获取设备版本信息	上层用户
DEV_GetSerialNum	获得序列号	上层用户
DEV_GetUserPID	获得用户产品 ID 号	上层用户
DEV_SetUserPID	设置用户产品 ID 号	上层用户
DEV_GetBusInfo	获得设备总线信息	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\ACTS6010\INCLUDE\ACTS6010.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 ACTS6010.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。

3.2 设备对象管理函数原型说明

♦ **创建设备对象函数**

函数原型：

Visual C++：

HANDLE DEV_Create (ULONG nIndex, BOOL nIndexType)

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：

nIndex　设备序号。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 逻辑 ID 标识值为名称后缀的标识符来确认和管理该设备。

nIndexType 设备序号类型, =0:使用逻辑号; =1:使用物理序号。

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。

相关函数： DEV_GetCount　DEV_GetCurrentIdx　DEV_ListDlg　DEV_Release

Visual C++程序举例

```
    :
HANDLE hDevice;     // 定义设备对象句柄
ULONG DeviceLgcID = 0;
    hDevice = ACTS6010_DEV_Create (DeviceLgcID, 0); // 用逻辑 ID 创建设备对象
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 ACTS6010 设备的总数量**

函数原型：

Visual C++:

int DEV_GetCount (HANDLE hDevice)

功能：取得 ACTS6010 设备的数量。

参数：

hDevice　设备对象句柄，它应由 DEV_Create 创建。

返回值：返回系统中 ACTS6010 的数量。

相关函数： DEV_GetCount　DEV_GetCurrentIdx　DEV_ListDlg　DEV_Release

♦**取得该设备当前相应的 ID 号**

函数原型：

Visual C++:

BOOL DEV_GetCurrentIdx (HANDLE hDevice,
ULONG* pDeviceLgcID,
ULONG* pDevicePhysID）

功能：取得指定设备逻辑 ID 和物理 ID 号。

参数：

pDeviceLgcID　返回设备的逻辑 ID。

pDevicePhysID　返回设备的物理 ID 号。

返回值：如果成功，则返回 TRUE，否则返回 FALSE。

相关函数： DEV_GetCount　DEV_GetCurrentIdx　DEV_ListDlg　DEV_Release

♦ **获得板卡信息**

函数原型：

Visual C++:

BOOL DEV_GetMainInfo (HANDLE hDevice，
PACTS6010_MAIN_INFO pMainInfo)

功能：获得板卡主要信息。

参数：

hDevice　设备对象句柄，它应由 DEV_Create 创建。

pMainInfo　设备信息结构体。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关函数： DEV_Create　DEV_Release

♦ **复位设备**

函数原型：

Visual C++:

BOOL DEV_Reset (HANDLE hDevice)

功能：将设备恢复到复位状态。

参数：

hDevice　设备对象句柄，它应由 DEV_Create 创建。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关函数：DEV_Create　　DEV_Release

- **用对话框控件显示系统中设备数,并列表该设备 BAR 地址**

函数原型：

Visual C++:

BOOL DEV_ListDlg (HANDLE hDevice)

功能：显示系统中设备数,并列表该设备 BAR 地址。

参数：

hDevice　设备对象句柄，它应由 DEV_Create 创建。

返回值：若成功，则弹出对话框控件列表所有 ACTS6010 设备的配置情况。

相关函数：DEV_Create　　DEV_Release

- **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 DEV_Release (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。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关函数：DEV_Create

应注意的是，DEV_Create 必须和 DEV_Release 函数一一对应，即当您执行了一次 DEV_Create 后，再一次执行此函数前，必须执行一次 DEV_Release 函数，以释放由 DEV_Create 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 DEV_Create 函数时，那些软硬件资源才可被再次使用。

3.3 电阻操作函数

BOOL RES_DisconnectAll (
HANDLE hDevice)

功能：　断开所有继电器的连接。

参数：

hDevice　设备对象句柄，它应由 DEV_Create 创建。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关函数：DEV_Create

BOOL RES_SetSettlingtime (
HANDLE hDevice，
ULONG nSettlingtime)

功能： 设置建立时间。

参数：

hDevice 设备对象句柄，它应由 DEV_Create 创建。

nSettlingtime 设备间建立的时间，单位为 mS，取值范围为[4,16000]

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关函数：DEV_Create

BOOL RES_GetSettlingtime (
HANDLE hDevice，
ULONG* pSettlingtime)

功能： 获取建立时间。

参数：

hDevice 设备对象句柄，它应由 DEV_Create 创建。

pSettlingtime 返回设备间建立的时间，单位为 mS，取值范围为[4,16000]

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关函数：DEV_Create RES_SetSettlingtime

BOOL RES_GetStatus (
HANDLE hDevice，
ULONG* pIsDebounced,
ULONG* pEepError,
ULONG* pErrorInfo)

功能： 读取板卡的状态信息。

参数：

hDevice 设备对象句柄，它应由 DEV_Create 创建。

pIsDebounced 判断板卡是否处于繁忙状态，如果在工作处于繁忙状态，则返回 1；如果处于空闲状态则返回 0。

pEepError 判断 Eep 是否错误，返回 1 表示 EEPROM 硬件错误可能的原因是序列号未烧写或者是芯片有问题造成的。

pErrorInfo 错误信息返回，若出错信息为 0x200 指的是电阻配置错误。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关函数：DEV_Create

BOOL RES_SetResistance (
HANDLE hDevice，
ULONG nChannel,
ULONG nOutputMode,
double* pResistance)

功能： 设置电阻值。

参数：

hDevice 设备对象句柄，它应由 DEV_Create 创建。

nChannel 通道号，取值范围为[0，nChannelCount-1]。

nOutputMode 输出模式，取值范围[0，3]。

pResistance 电阻值，取值范围[nMinResistance,nMaxResistance]。
返回值：若成功，则返回 TRUE，否则返回 FALSE。
相关函数：DEV_Create

BOOL RES_GetResistance (
HANDLE hDevice，
ULONG nChannel,
ULONG pOutputMode,
double* pResistance)
功能： 获取电阻值。
参数：
hDevice 设备对象句柄，它应由 DEV_Create 创建。
nChannel 通道号，取值范围为[0，nChannelCount-1]。
nOutputMode 输出模式，取值范围[0，3] ,所有通道共用输出模式。
pResistance 电阻值，取值范围[nMinResistance,nMaxResistance]。
返回值：若成功，则返回 TRUE，否则返回 FALSE。
相关函数：DEV_Create RES_SetResistance

BOOL RES_SetPSRelayState (
HANDLE hDevice，
ULONG nChannel,
ULONG nPathRelay,
ULONG nShortCircuit)
功能： 设置通路短路继电器状态。
参数：
hDevice 设备对象句柄，它应由 DEV_Create 创建。
nChannel 通道号，取值范围为[0，nChannelCount-1]。
nPathRelay 通路继电器状态，1：闭合 0：断开。
nShortCircuit 短路继电器状态，1：闭合 0：断开。
返回值：若成功，则返回 TRUE，否则返回 FALSE。
相关函数：DEV_Create

BOOL RES_GetPSRelayState (
HANDLE hDevice，
ULONG nChannel,
ULONG* pPathRelay,
ULONG* pShortCircuit)
功能： 获取通路短路继电器状态。
参数：
hDevice 设备对象句柄，它应由 DEV_Create 创建。
nChannel 通道号，取值范围为[0，nChannelCount-1]。
pPathRelay 回读通路继电器状态，1：闭合 0：断开。
pShortCircuit 回读短路继电器状态，1：闭合 0：断开。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关函数：DEV_Create　　RES_SetPSRelayState

3.4 设备信息函数

BOOL DEV_SetPhysIdx(
　　HANDLE hDevice,
　　ULONG nPhysIdx)

功能：初设置物理序号。

参数：

nPhysIdx　物理序号[0~255]。

返回值：若成功，返回 TRUE，否则返回 FALSE。

BOOL DEV_GetVersion(
　　HANDLE hDevice,
　　ULONG* pDllVer,
　　ULONG* pDriverVer,
　　ULONG* pFirmwareVer)

功能：获得设备版本信息。

参数：

pDllVer　返回的动态库(.dll)版本号。

pDriverVer　返回的驱动(.sys)版本号。

pFirmwareVer　返回的固件版本号。

返回值：若成功，返回 TRUE，否则返回 FALSE。

BOOL DEV_GetSerialNum(
　　HANDLE hDevice,
　　ULONG* pSerialNum)

功能：获得序列号。

参数：

pSerialNum　返回的序列号。

返回值：若成功，返回 TRUE，否则返回 FALSE。

BOOL DEV_GetUserPID(
　　HANDLE hDevice,
　　ULONG* pUserPID)

功能：获得用户产品 ID 号(User Product Identification)。

参数：

pUserPID　返回的用户产品 ID。

返回值：若成功，返回 TRUE，否则返回 FALSE。

BOOL DEV_SetUserPID(
HANDLE hDevice
ULONG nUserPID)

功能：设置用户产品 ID 号(User Product Identification)。

参数：

nUserPID　用户产品 ID。

返回值：若成功，返回 TRUE，否则返回 FALSE。

BOOL DEV_GetBusInfo(
HANDLE hDevice,
ULONG* pBusNumber,
ULONG* pFunctionNumber,
ULONG* pDeviceNumber)

功能：获得设备总线信息。

参数：

pBusNumber　返回的动态库(.dll)版本号。

pFunctionNumber　返回的驱动(.sys)版本号。

pDeviceNumber　返回的固件版本号。

返回值：若成功，返回 TRUE，否则返回 FALSE。

4 硬件参数结构

板卡信息结构体(ACTS6010_MAIN_INFO)

Visual C++:

```
typedef struct _ACTS6010_MAIN_INFO
{
U32 nDeviceType;            //总线类型\设备类型 x20127011
20117011...2012(PXIE)2011(PCIE)2008(PXI)2007(PCI)
 U32 nRevisionID;           // Revision ID
 U32 nHardwareVer;              // 硬件版本
 U32 nFirmwareVer;              // 固件版本
 U32 nCheckBusH;                // 检查总线通讯焊接问题,返回 x55AA55AA
 U32 nCheckBusL;                // 检查总线通讯焊接问题,返回 xAA55AA55
 U32 nChannelCount;             // 通道数(1~16)
 U32 nTimeBase;                 // 基频(如=33000000Hz)
 U32 nMaxResistance;            // 最大阻值,单位欧姆
 U32 nMinResistance;            // 最小阻值,单位欧姆
 U32 nStepResistance;       // 步进阻值,单位微欧
 U32 nReserved0;                // 保留字段(暂未定义)
 U32 nResolution;           // 电阻分辨率(如=8 表示 Bit; =12 表示 Bit; =17 表示 Bit)
 U32 nCodeCount;                // 电阻编码数量(如, 4096)
 U32 nMaxLSB;                   // 最大码值(如, 4095)
 U32 nReserved1;                // 保留字段(暂未定义)
 U32 nReserved2;                // 保留字段(暂未定义)
 U32 nReserved3;                // 保留字段(暂未定义)
 U32 nReserved4;                // 保留字段(暂未定义)
 U32 nReserved5;                // 保留字段(暂未定义)
} ACTS6010_MAIN_INFO, *PACTS6010_MAIN_INFO;
```

RES_SetResistance 函数 nOutputMode 参数所使用的常量定义

常量名	常量值	功能定义
RES_OUT_MODE_NOWAIT	0	无等待，禁止建立时间
RES_OUT_MODE_DEFAULT	1	先断后连
RES_OUT_MODE_MBB	2	先通后断
RES_OUT_MODE_WAIT	3	立即执行后，等待建立时间

5 上层用户函数接口应用实例

5.1 简易程序演示说明

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 VS2005 的 Sys 工程。

[程序]丨**[阿尔泰测控演示系统]**丨 **[ACTS6010]**丨 **[Microsoft Visual C++]**丨**[简易代码演示]**丨**[Resistance]**

5.2 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 ACTS6010.h)。

[程序]丨**[阿尔泰测控演示系统]**丨 **[ACTS6010]**丨 **[Microsoft Visual C++]**丨 [高级代码演示]**

其默认存放路径为：系统盘\ART\ACTS6010\SAMPLES\VC\ADVANCED

其他语言的演示可以用上面类似的方法找到。

6 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。

6.1 公用接口函数总列表（每个函数省略了前缀“ACTS6010_”）

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
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	

6.2 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 DEV_Create 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;：DEV_Create　GetDeviceBar　WriteRegisterByte　WriteRegisterWord　WriteRegisterULong　ReadRegisterByte

ReadRegisterWord ReadRegisterULong DEV_Release
GetDevVersion

♦ **以单字节（即 8 位）方式写 PCIe 内存映射寄存器的某个单元**

函数原型:

Visual C++:

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

功能：以单字节（即 8 位）方式写 PCIe 内存映射寄存器。

参数：

hDevice 设备对象句柄，它应由 DEV_Create 创建。

pbLinearAddr PCIe 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;：DEV_Create GetDeviceBar WriteRegisterByte
WriteRegisterWord WriteRegisterULong ReadRegisterByte
ReadRegisterWord ReadRegisterULong DEV_Release
GetDevVersion

♦ **以双字节（即 16 位）方式写 PCIe 内存映射寄存器的某个单元**

函数原型:

Visual C++:

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

功能：以双字节（即 16 位）方式写 PCIe 内存映射寄存器。

参数：

hDevice 设备对象句柄，它应由 DEV_Create 创建。

pbLinearAddr PCIe 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：DEV_Create GetDeviceBar WriteRegisterByte
WriteRegisterWord WriteRegisterULong ReadRegisterByte

ReadRegisterWord ReadRegisterULong DEV_Release
GetDevVersion

♦ **以四字节（即 32 位）方式写 PCIe 内存映射寄存器的某个单元**

函数原型:

Visual C++:

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

功能：以四字节（即 32 位）方式写 PCIe 内存映射寄存器。

参数：

hDevice 设备对象句柄，它应由 DEV_Create 创建。

pbLinearAddr PCIe 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： DEV_Create GetDeviceBar WriteRegisterByte
WriteRegisterWord WriteRegisterULong ReadRegisterByte
ReadRegisterWord ReadRegisterULong DEV_Release
GetDevVersion

♦ **以单字节（即 8 位）方式读 PCIe 内存映射寄存器的某个单元**

函数原型:

Visual C++:

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

功能：以单字节（即 8 位）方式读 PCIe 内存映射寄存器的指定单元。

参数：

hDevice 设备对象句柄，它应由 DEV_Create 创建。

pbLinearAddr PCIe 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： DEV_Create GetDeviceBar WriteRegisterByte
WriteRegisterWord WriteRegisterULong ReadRegisterByte
ReadRegisterWord ReadRegisterULong DEV_Release
GetDevVersion

◆ **以双字节（即 16 位）方式读 PCIe 内存映射寄存器的某个单元**

函数原型:

Visual C++:

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

功能：以双字节（即 16 位）方式读 PCIe 内存映射寄存器的指定单元。

参数：

hDevice　设备对象句柄，它应由 DEV_Create 创建。

pbLinearAddr　PCIe 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： DEV_Create　GetDeviceBar　WriteRegisterByte　WriteRegisterWord　WriteRegisterULong　ReadRegisterByte　ReadRegisterWord　ReadRegisterULong　DEV_Release　GetDevVersion

◆ **以四字节（即 32 位）方式读 PCIe 内存映射寄存器的某个单元**

函数原型:

Visual C++:

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

功能：以四字节（即 32 位）方式读 PCIe 内存映射寄存器的指定单元。

参数：

hDevice　设备对象句柄，它应由 DEV_Create 创建。

pbLinearAddr　PCIe 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;： DEV_Create　GetDeviceBar　WriteRegisterByte　WriteRegisterWord　WriteRegisterULong　ReadRegisterByte　ReadRegisterWord　ReadRegisterULong　DEV_Release　GetDevVersion

6.3 I/O 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**

Visual C++:

```
BOOL WritePortByte (HANDLE hDevice,
                    __int64 pPort,
                    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 DEV_Create 创建。

pPort 指定寄存器的物理基地址。

OffsetBytes 相对于物理基地址的偏移位置(字节)。

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： DEV_Create WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

♦ **以双字(16Bit)方式写 I/O 端口**

Visual C++:

```
BOOL WritePortWord (HANDLE hDevice,
                    __int64 pPort,
                    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 DEV_Create 创建。

pPort 指定寄存器的物理基地址。

OffsetBytes 相对于物理基地址的偏移位置(字节)。

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： DEV_Create WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

♦ **以四字节(32Bit)方式写 I/O 端口**

Visual C++:

```
BOOL WritePortULong (HANDLE hDevice,
                     __int64 pPort,
                     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 DEV_Create 创建。

pPort 指定寄存器的物理基地址。

OffsetBytes 相对于物理基地址的偏移位置(字节)。
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： DEV_Create　WritePortByte　WritePortWord
WritePortULong　ReadPortByte　ReadPortWord

- **以单字节(8Bit)方式读 I/O 端口**

Visual C++:

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

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 DEV_Create 创建。
pPort 指定寄存器的物理基地址。
OffsetBytes 相对于物理基地址的偏移位置(字节)。
返回值：返回由 nPort 指定的端口的值。
相关函数： DEV_Create　WritePortByte　WritePortWord
WritePortULong　ReadPortByte　ReadPortWord

- **以双字节(16Bit)方式读 I/O 端口**

Visual C++:

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

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 DEV_Create 创建。
pPort 指定寄存器的物理基地址。
OffsetBytes 相对于物理基地址的偏移位置(字节)。
返回值：返回由 nPort 指定的端口的值。
相关函数： DEV_Create　WritePortByte　WritePortWord
WritePortULong　ReadPortByte　ReadPortWord

- **以四字节(32Bit)方式读 I/O 端口**

Visual C++:

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

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 DEV_Create 创建。
pPort 指定寄存器的物理基地址。

OffsetBytes 相对于物理基地址的偏移位置(字节)。

返回值：返回由 nPort 指定端口的值。

相关函数： DEV_Create　WritePortByte　WritePortWord　WritePortULong　ReadPortByte　ReadPortWord

6.4 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：

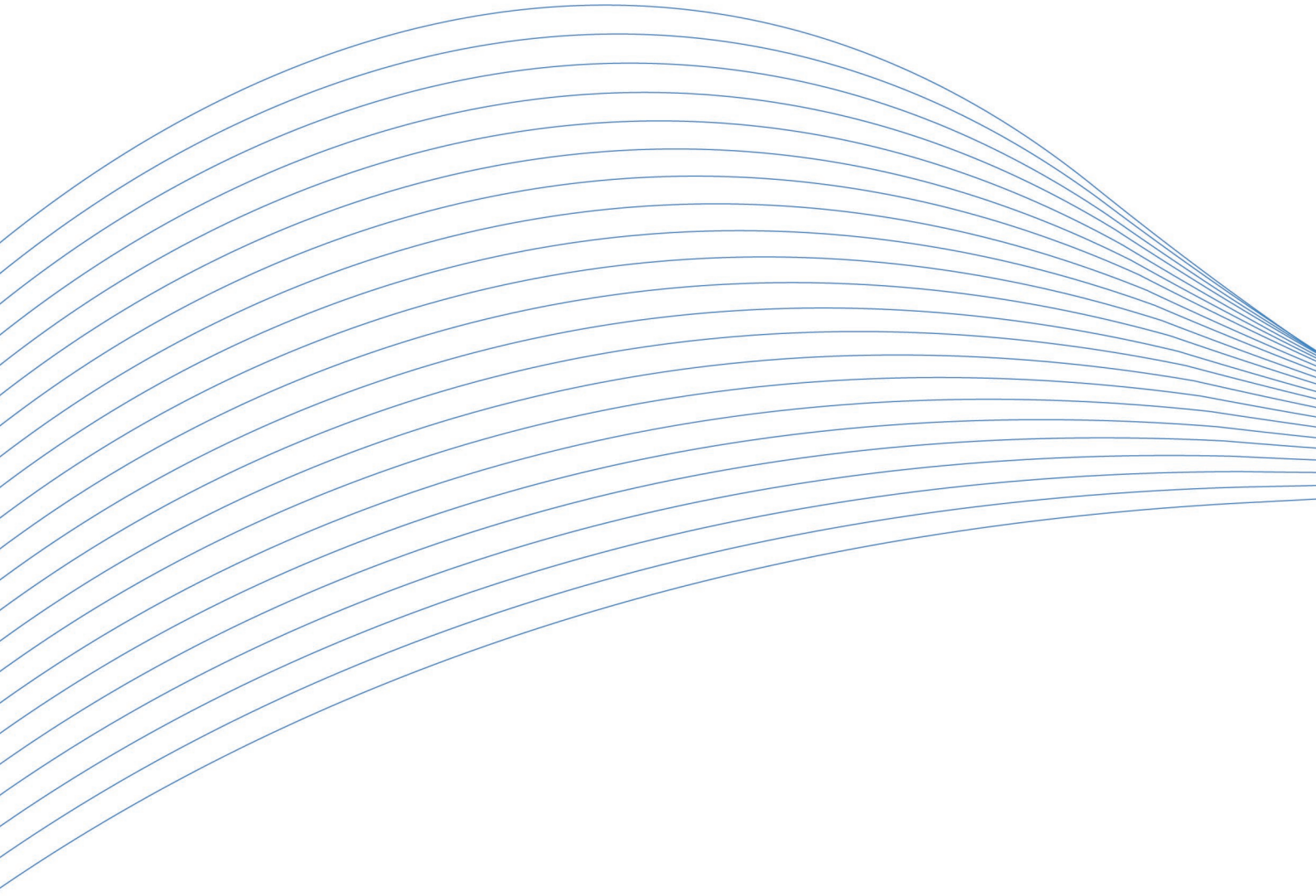
Visual C++:

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。



阿尔泰科技

服务热线：400-860-3335

网址：www.art-control.com