ART2543 WIN2000/XP 驱动程序使用说明书

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

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。

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 ART2543_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			

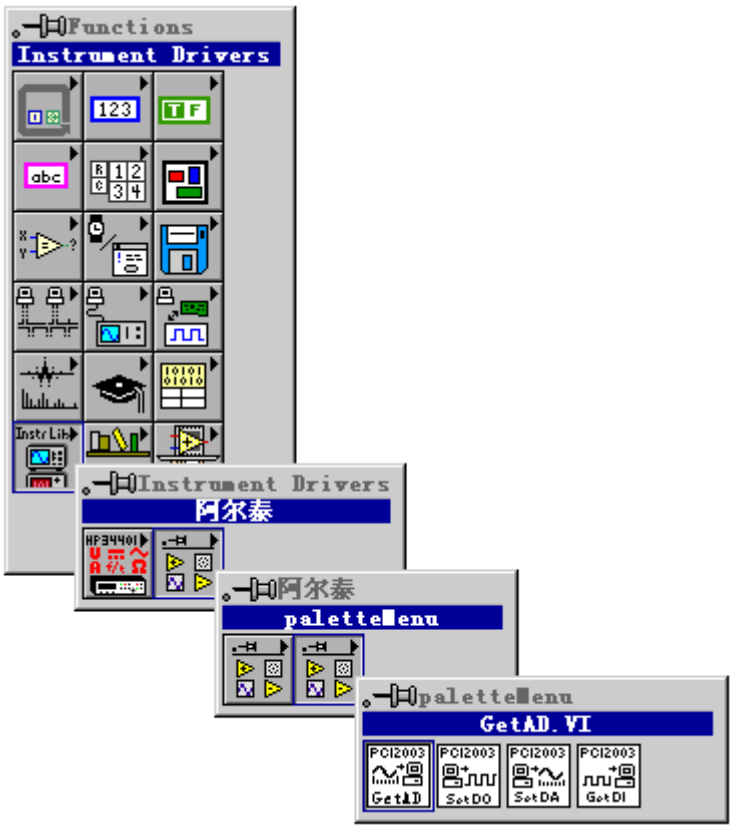
以上规则不局限于该产品。

第二章 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。这个时候您便可以用这个虚拟线性基地址，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。

另外需要申明的是，在本章和下一章中列明的关于 LabView 的接口，均属于外挂式驱动接口，他是通过 LabView 的 Call Labrary Function 功能模板实现的。它的特点是除了自身的语法略有不同以外，每一个基于 LabView 的驱动图标与 Visual C++、Visual Basic、Delphi 等语言中每个驱动函数是一一对应的，其调用流程和功能是完全相同的。那么相对于外挂式驱动接口的另一种方式是内嵌式驱动。这种驱动是完全作为 LabView 编程环境中的紧密耦合的一部分，它可以直接从 LabView 的 Functions 模板中取得，如下图所示。此种方式更适合上层用户的需要，它的最大特点是方便、快捷、简单，而且可以取得它的在线帮助。关于 LabView 的外挂式驱动和内嵌式驱动更详细的叙述，请参考 LabView 的相关演示。



LabView 内嵌式驱动接口的获取方法

第一节、设备驱动接口函数总列表（每个函数省略了前缀“ART2543_”）

函数名	函数功能	备注
设备对象操作函数		
CreateDevice	创建 PCI 设备对象(用设备逻辑号)	上层及底层用户
ReleaseDevice	关闭设备，且释放 PCI 总线设备对象	上层及底层用户
CNT 计数定时器操作函数		
SetDeviceCNT	设置计数器的初值	
GetDeviceCNT	取得各路计数器的当前计数值	
SetMeasureFre	设置设备测频模式	
GetMeasureFreSts	获取测频状态	
GetMeasureFre	获取被测频率值,返回被测频率值	

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\ART2543\INCLUDE\ART2543.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 ART2543.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(ART2543.H)写进您的源程序头部。如：#include “\Art\ART2543\Include\ART2543.h”，然后再将 ART2543.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 ART2543.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 ART2543.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 ART2543.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：“ART2543”。如：

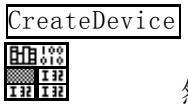
```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ART2543;  // 注意： 在此加入驱动程序接口单元 ART2543
```

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 ART2543.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，如接口单元，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

函数原型：

Visual C++ & C++Builder：

HANDLE CreateDevice (WORD wBaseAddr)

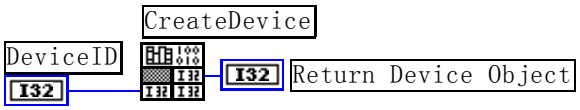
Visual Basic：

Declare Function CreateDevice Lib "ART2543" (ByVal wBaseAddr As Integer) As long

Delphi：

Function CreateDevice(wBaseAddr: Word) : Integer;
StdCall; External 'ART2543' Name ' CreateDevice ';

LabVIEW：



功能：创建设备对象。

参数：wBaseAddr 基地址。

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

Visual C++ & C++Builder 程序举例:

```
        :
HANDLE hDevice;     // 定义设备对象句柄
word   wBaseAddr;
hDevice = CreateDevice (wBaseAddr); //  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 wBaseAddr As Integer
hDevice = CreateDevice (wBaseAddr)    ' 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
        :
```

♦ **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)
```

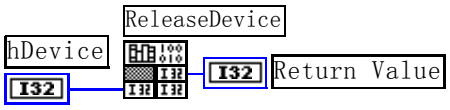
Visual Basic:

```
Declare Function ReleaseDevice Lib "ART2543" (ByVal hDevice As Long ) As Boolean
```

Delphi：

```
Function ReleaseDevice(hDevice : Integer) : Boolean;
                        StdCall; External 'ART2543' 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。

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、CNT 计数与定时器操作函数原型说明

♦ **设置计数器的初值**

函数原型：

Visual C++ & C++Builder:

```
BOOL SetDeviceCNT ( HANDLE hDevice,
                    ART2543_PARA_CNT CNTPara,
                    WORD nChannel)
```

Visual Basic:

```
Declare Function SetDeviceCNT Lib "ART2543"(ByVal hDevice As Long, _
                                            ByVal CNTPara As ART2543_PARA_CNT, _
                                            ByVal nChannel As Integer) As Boolean
```

Delphi:

```
Function SetDeviceCNT (hDevice : Integer;
                       CNTPara: ART2543_PARA_CNT;
                       nChannel: Word) : Boolean;
                       StdCall; External 'ART2543' Name ' SetDeviceCNT';
```

LabVIEW:

请参考相关演示程序。

功能：设置计数器的初值。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

CNTPara 计数器参数。

nChannel 计数器通道选择，取值范围为 0～7。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数: CreateDevice GetDeviceCNT ReleaseDevice

♦ **取得各路计数器的当前计数值**

函数原型：

Visual C++ & C++Builder:

```
BOOL GetDeviceCNT (HANDLE hDevice,
                   PART2543_PARA_CNT pCNTPara,
                   WORD nChannel)
```

Visual Basic:

```
Declare Function GetDeviceCNT Lib "ART2543" (ByVal hDevice As Long, _
                                             ByRef pCNTPara As PART2543_PARA_CNT, _
                                             ByVal nChannel As Integer) As Boolean
```

Delphi:

```
Function GetDeviceCNT ( hDevice : Integer;
                        pCNTPara: PART2543_PARA_CNT;
                        nChannel: Word) : Boolean;
                        StdCall; External 'ART2543' Name ' GetDeviceCNT ';
```

LabVIEW:

请参考相关演示程序。

功能：取得各路计数器的当前计数值。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pCNTPara 取得计数器参数。

nChannel 计数器通道选择，取值范围为 0～7。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数: CreateDevice SetDeviceCNT ReleaseDevice

◆ **设置设备测频模式**

函数原型：

Visual C++ & C++Builder:

```
BOOL SetMeasureFre ( HANDLE hDevice,
                     ULONG ulDelayTime,
                     WORD nChannel)
```

Visual Basic:

```
Declare Function SetMeasureFre Lib "ART2543"(ByVal hDevice As Long, _
                                             ByVal ulDelayTime As Long,_
                                             ByVal nChannel As Integer) As Boolean
```

Delphi:

```
Function SetMeasureFre (hDevice : Integer;
                        ulDelayTime : LongWord;
                        nChannel: Word) : Boolean;
                        StdCall; External 'ART2543' Name ' SetMeasureFre ';
```

LabVIEW:

请参考相关演示程序。

功能：设置设备测频模式。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

ulDelayTime 确保测频准确度的定时高电平脉冲时间，取值为[100——1000ms]。

nChannel 计数器通道选择，取值范围为 0～7。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数: CreateDevice GetMeasureFreSts GetMeasureFre
ReleaseDevice

◆ **获取测频状态**

函数原型：

Visual C++ & C++Builder:

```
BOOL GetMeasureFreSts(HANDLE hDevice,
                      PULONG pCNTStatus,
                      WORD nChannel)
```

Visual Basic:

```
Declare Function GetMeasureFreSts Lib "ART2543"(ByVal hDevice As Long, _
                                                ByRef pCNTStatus As Long,_
                                                ByVal nChannel As Integer) As Boolean
```

Delphi:

```
Function GetMeasureFreSts(hDevice : Integer;
                          pCNTStatus : Pointer;
                          nChannel: Word) : Boolean;
                          StdCall; External 'ART2543' Name ' GetMeasureFreSts ';
```

LabVIEW:

请参考相关演示程序。

功能：获取测频状态。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pCNTStatus 测频状态：等于 0 表示测频结束，等于 1 表示正在测频计数。

nChannel 计数器通道选择，取值范围为 0～7。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数: CreateDevice SetMeasureFre GetMeasureFre ReleaseDevice

- **获取被测频率值，返回被测频率值**

函数原型：

Visual C++ & C++Builder:

```
DOUBLE GetMeasureFre(HANDLE hDevice,
                     WORD nChannel)
```

Visual Basic:

```
Declare Function GetMeasureFre Lib "ART2543"(ByVal hDevice As Long, _
                                             ByVal nChannel As Integer) As Double
```

Delphi:

```
Function GetMeasureFre (hDevice : Integer;
                        nChannel: Word) : Double;
                        StdCall; External 'ART2543' Name ' GetMeasureFre';
```

LabVIEW:

请参考相关演示程序。

功能：获取被测频率值，返回被测频率值。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

nChannel 计数器通道选择，取值范围为 0～7。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数: CreateDevice SetMeasureFre GetMeasureFreSts ReleaseDevice

第三章 计数器参数结构

Visual C++ & C++Builder：

```
typedef struct _ART2543_PARA_CNT
{
    WORD ControlMode;          // 计数器工作模式
    ULONG CNTVal;              // 计数初值
    ULONG CNTMode;             // 计数器方式：加计数或减计数[0：减计数，1：加计数]
} ART2543_PARA_CNT, *PART2543_PARA_CNT;
```

Visual Basic：

```
Private Type ART2543_PARA_CNT
    ControlMode As Integer     ' 计数器工作模式
    CNTVal As Long             ' 计数初值
    CNTMode As Long            ' 计数器方式：加计数或减计数[0：减计数，1：加计数]
```

End Type

Delphi:

```
Type   // 定义结构体数据类型
  PART2543_PARA_CNT = ^ ART2543_PARA_CNT;   // 指针类型结构
  ART2543_PARA_CNT = record              // 标记为记录型
        ControlMode : Word;              // 计数器工作模式
        CNTVal : Longword;               // 计数初值
        CNTMode : Longword;              // 计数器方式：加计数或减计数[0：减计数，1：加计数]
End;
```

LabVIEW:

请参考相关演示程序。

ControlMode 计数器工作模式选择，其值如下表：

常量名	常量值	功能定义
ART2543_GATEMODE_POSITIVE_0	0x0000	计数结束产生中断
ART2543_GATEMODE_RISING_1	0x0001	可编程单拍脉冲
ART2543_GATEMODE_POSITIVE_2	0x0002	频率发生器
ART2543_GATEMODE_POSITIVE_3	0x0003	方波发生器
ART2543_GATEMODE_POSITIVE_4	0x0004	软件触发选通
ART2543_GATEMODE_RISING_5	0x0005	硬件触发选通

CNTVal 计数初值。

CNTMode 计数器方式：加计数或减计数[0：减计数，1：加计数]。

第四章 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“ART2543_”）

函数名	函数功能	备注
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

第二节、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)

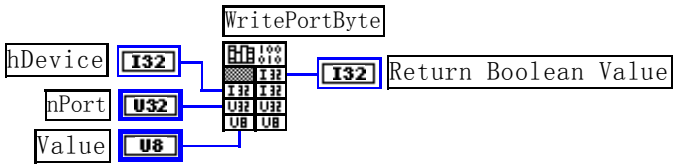
Visual Basic:

```
Declare Function WritePortByte Lib "ART2543" ( 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 'ART2543' 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。

相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord ReleaseDevice

- **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)
```

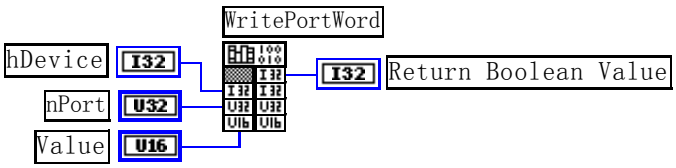
Visual Basic:

```
Declare Function WritePortWord Lib "ART2543" (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 'ART2543' 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。

相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord ReleaseDevice

♦ **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)
```

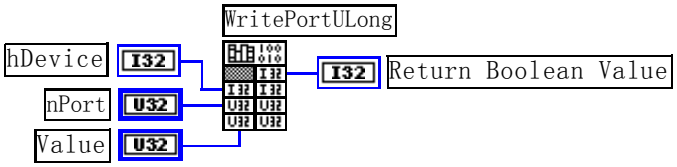
Visual Basic:

```
Declare Function WritePortULong Lib "ART2543" (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 'ART2543' 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。

相关函数： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord ReleaseDevice

♦ **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)
```

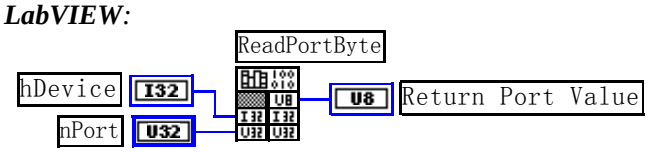
Visual Basic:

```
Declare Function ReadPortByte Lib "ART2543" (ByVal hDevice As Long, _
                                             ByVal nPort As Long ) As Byte
```

Delphi:

```
Function ReadPortByte(hDevice : Integer;
                      nPort : LongWord) : Byte;
                      StdCall; External 'ART2543' 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

ReleaseDevice

- **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)
```

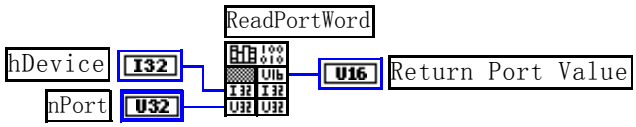
Visual Basic:

```
Declare Function ReadPortWord Lib "ART2543" ( ByVal hDevice As Long, _
                                              ByVal nPort As Long ) As Integer
```

Delphi:

```
Function ReadPortWord(hDevice : Integer;
                      nPort : LongWord) : Word;
                      StdCall; External 'ART2543' 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　ReleaseDevice

- **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)
```

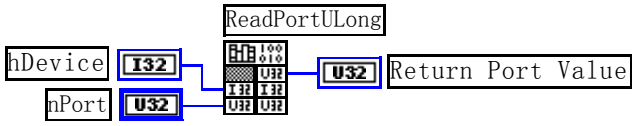
Visual Basic:

```
Declare Function ReadPortULong Lib "ART2543" ( ByVal hDevice As Long, _
                                               ByVal nPort As Long ) As Long
```

Delphi:

```
Function ReadPortULong(hDevice : Integer;
                       nPort : LongWord) : LongWord;
                       StdCall; External 'ART2543' 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　ReleaseDevice