ART2007 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

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 ART2007_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。诸如InitDeviceAD、ReadDeviceAD、SetDeviceDO等。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bDOSts[16]中的相应元素决定。由GetDeviceDI函数实现开关量的输入操作，其各路开关量的输入状态由其bDISts[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， 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，有些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 设备操作函数接口介绍

第一节、设备驱动接口函数总列表（每个函数省略了前缀“ART2007_”）

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
GetDeviceCurrentBaseAddr	取得当前设备基地址	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
程序方式 AD 读取函数		
InitDeviceAD	初始化 AD 部件准备传输	上层用户
ReadDeviceAD	连续读取当前 PCI 设备上的 AD 数据	上层用户
ReleaseDeviceAD	释放设备上的 AD 部件	上层用户
AD 硬件参数系统保存、读取函数		
LoadParaAD	从 Windows 系统中读入硬件参数	上层用户
SaveParaAD	往 Windows 系统写入设备硬件参数	上层用户
ResetParaAD	将注册表中的 AD 参数恢复至出厂默认值	上层用户
LoadBaseAddr	将基地址从系统中读出	
SaveBaseAddr	将基地址保存至系统中	
开关量简易操作函数		
GetDeviceDI	开关输入函数	上层用户
SetDeviceDO	开关输出函数	上层用户
RetDeviceDO	回读开关量输出状态	

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\ART2007\INCLUDE\ART2007.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 ART2007.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(ART2007.H)写进您的源程序头部。如：#include “\Art\ART2007\Include\ART2007.h”，然后再将 ART2007.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 ART2007.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 ART2007.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:

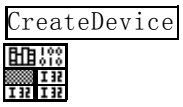
要使用如下函数一个关键的问题是首先必须将我们提供的单元模块文件（*.Pas）加入到您的 Delphi 工程

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 ART2007.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：“ART2007”。如：

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ART2007;  // 注意： 在此加入驱动程序接口单元 ART2007
```

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ART2007.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ReadDeviceAD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 BaseAddress)
```

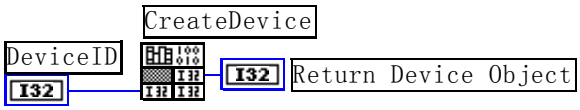
Visual Basic：

```
Declare Function CreateDevice Lib "ART2007" (ByVal BaseAddress As Integer) As Long
```

Delphi：

```
Function CreateDevice(BaseAddress :Word):Integer; StdCall; External 'ART2007' Name 'CreateDevice';
```

LabView:



功能：该函数使用基地址创建设备对象，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：BaseAddress 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： CreateDevice ReleaseDevice

Visual C++ & C++Builder 程序举例:

：

```
HANDLE hDevice;     // 定义设备对象句柄
hDevice = CreateDevice (BaseAddress ); // 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
hDevice = CreateDevice (BaseAddress )    ' 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

Else
     Exit Sub        ' 退出该过程
End If
            :
```

- **取得当前设备基地址**

函数原型：

Visual C++ & C++ Builder：

WORD GetDeviceCurrentBaseAddr (HANDLE hDevice)

Visual Basic：

Declare Function GetDeviceCurrentBaseAddr Lib "ART2007" (ByVal hDevice As Long) As Integer

Delphi：

Function GetDeviceCurrentBaseAddr(hDevice : Integer):Word;
StdCall; External 'ART2007' Name 'CreateDevice';

LabView:

请参考相关演示程序。

功能：取得当前设备基地址。
参数：hDevice设备对象句柄，它应由CreateDevice创建。
返回值：如果执行成功，则返回设备基地址。
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++ & C++Builder:

BOOL ReleaseDevice(HANDLE hDevice)

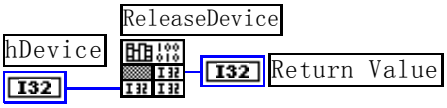
Visual Basic:

Declare Function ReleaseDevice Lib "ART2007" (ByVal hDevice As Long) As Boolean

Delphi：

Function ReleaseDevice(hDevice : Integer) : Boolean;
StdCall; External 'ART2007' 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、AD 采样操作函数原型说明

♦ **初始化 AD 设备（ Initlize device AD for program mode）**

函数原型

Visual C++ & C++Builder::

```
BOOL InitDeviceAD ( HANDLE hDevice,
                    PART2007_PARA_AD pADPara)
```

Visual Basic:

```
Declare Function InitDeviceAD Lib "ART2007" (ByVal hDevice As Long, _
                                             ByRef pADPara As ART2007_PARA_AD) As Boolean
```

Delphi:

```
Function InitDeviceAD (hDevice : Integer;
                       pADPara : PART2007_PARA_AD) : Boolean;
                       StdCall; External 'ART2007' Name ' InitDeviceAD';
```

LabVIEW：

请参考相关演示程序。

功能：它负责初始化设备对象中的 AD 部件，为设备的操作就绪做有关准备工作，如预置 AD 采集通道、采样频率等。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pADPara 设备对象参数结构， 它决定了设备对象的各种状态及工作方式，如AD采样通道、采样频率等。关于ART2007_PARA_AD具体定义请参考ART2007.h(.Bas或.Pas或.VI)驱动接口文件及本文档中的《AD硬件参数结构》。

返回值：如果初始化设备对象成功，则返回 TRUE， 否则返回 FALSE。

相关函数： CreateDevice　InitDeviceAD　ReadDeviceAD　ReleaseDeviceAD　ReleaseDevice

♦ **读取 PCI 设备上的 AD 数据**

函数原型:

Visual C++ & C++Builder:

```
BOOL ReadDeviceAD(HANDLE hDevice,
                  WORD ADBuffer[],
                  LONG nReadSizeWords,
                  PLONG nRetSizeWords)
```

Visual Basic:

```
Declare Function ReadDeviceAD Lib "ART2007" (ByVal hDevice As Long, _
                                             ByRef ADBuffer As Integer,_
                                             ByVal nReadSizeWords As Long,_
                                             Optional ByRef nRetSizeWords As Long) As Boolean
```

Delphi:

```
Function ReadDeviceAD (hDevice : Integer;
                       ADBuffer : Pointer;
                       nReadSizeWords : LongInt;
                       nRetSizeWords : Pointer) : Boolean;
                       StdCall; External 'ART2007' Name ' ReadDeviceAD ';
```

LabVIEW：

请参考相关演示程序。

功能：读取PCI设备AD部件上的批量数据。它不负责初始化AD部件，待读完整过指定长度的数据才返回。它必须在InitDeviceAD之后，ReleaseDeviceAD之前调用。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

ADBuffer 接受AD数据的用户缓冲区，它可以是一个用户定义的数组。关于如何将这些AD数据转换成相应的电压值，请参考《数据格式转换与排列规则》。

nReadSizeWords指定一次ReadDeviceAD操作应读取多少字数据到用户缓冲区。注意此参数的值不能大于用

户缓冲区ADBuffer的最大空间。此参数值只与ADBuffer[]指定的缓冲区大小有效，而与FIFO存储器大小无效。

nRetSizeWords 返回实际读取的点数(或字数)。

返回值：如果调用成功，则返回 TRUE，且 AD 立刻开始转换，否则返回 FALSE。

注释：此函数也可用于单点读取和几个点的读取，只需要将nReadSizeWords设置成 1 或相应值即可。其使用方法请参考《AD高速大容量、连续不间断数据采集及存盘技术详解》章节。

相关函数： CreateDevice　InitDeviceAD　ReadDeviceAD　ReleaseDeviceAD　ReleaseDevice

♦ **释放设备上的 AD 部件**

函数原型：

Visual C++ & C++ Builder:

```
BOOL ReleaseDeviceAD(HANDLE hDevice)
```

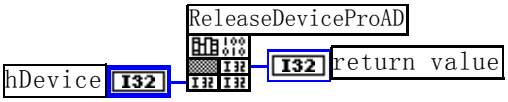
Visual Basic:

```
Declare Function ReleaseDeviceAD Lib "ART2007" (ByVal hDevice As Long ) As Boolean
```

Delphi：

```
Function ReleaseDeviceAD (hDevice : Integer) : Boolean;
                          StdCall; External 'ART2007' Name ' ReleaseDeviceAD ';
```

LabVIEW:



功能：释放设备上的 AD 部件。

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。

应注意的是，InitDeviceAD必须和ReleaseDeviceAD函数一一对应，即当您执行了一次InitDeviceAD后，再一次执行这些函数前，必须执行一次ReleaseDeviceAD函数，以释放由InitDeviceAD占用的系统软硬件资源，如映射寄存器地址、系统内存等。只有这样，当您再次调用InitDeviceAD函数时，那些软硬件资源才可被再次使用。

相关函数： CreateDevice　InitDeviceAD　ReadDeviceAD　ReleaseDeviceAD　ReleaseDevice

♦ **程序查询方式采样函数一般调用顺序**

① CreateDevice
② InitDeviceAD
③ ReadDeviceAD
④ ReleaseDeviceAD
⑤ ReleaseDevice

注明：用户可以反复执行第③步，以实现高速连续不间断大容量采集。

第四节、AD 硬件参数保存与读取函数原型说明

♦ **从 Windows 系统中读入硬件参数函数**

函数原型：

Visual C++ & C++ Builder:

```
BOOL LoadParaAD(HANDLE hDevice,
                PART2007_PARA_AD pADPara)
```

Visual Basic:

```
Declare Function LoadParaAD Lib "ART2007" (ByVal hDevice As Long, _
                                           ByRef pADPara As ART2007_PARA_AD) As Boolean
```

Delphi:

```
Function LoadParaAD (hDevice : Integer;
                     pADPara : PART2007_PARA_AD) : Boolean;
                     StdCall; External 'ART2007' Name ' LoadParaAD ';
```

LabVIEW:

请参考相关演示程序。

功能：负责从 Windows 系统中读取设备的硬件参数。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
pADPara属于PART2007_PARA_AD的结构指针类型，它负责返回PCI硬件参数值，关于结构指针类型PART2007_PARA_AD请参考ART2007.h或ART2007.Bas或ART2007.Pas函数原型定义文件，也可参考本文《硬件参数结构》关于该结构的有关说明。
返回值：若成功，返回 TRUE，否则返回 FALSE。
相关函数： CreateDevice LoadParaAD SaveParaAD ReleaseDevice

♦ **往 Windows 系统写入设备硬件参数函数**
函数原型：
Visual C++ & C++ Builder:
```
BOOL SaveParaAD (HANDLE hDevice,
                 PART2007_PARA_AD pADPara)
```
Visual Basic:
```
Declare Function SaveParaAD Lib "ART2007" (ByVal hDevice As Long, _
                                           ByRef pADPara As ART2007_PARA_AD) As Boolean
```
Delphi:
```
Function SaveParaAD (hDevice : Integer;
                     pADPara : PART2007_PARA_AD) : Boolean;
                     StdCall; External 'ART2007' Name ' SaveParaAD ';
```
LabVIEW:
请参考相关演示程序。

功能：负责把用户设置的硬件参数保存在 Windows 系统中，以供下次使用。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
pADPara设备硬件参数，关于ART2007_PARA_AD的详细介绍请参考ART2007.h或ART2007.Bas或ART2007.Pas函数原型定义文件，也可参考本文《硬件参数结构》关于该结构的有关说明。
返回值：若成功，返回 TRUE，否则返回 FALSE。
相关函数： CreateDevice LoadParaAD SaveParaAD ReleaseDevice

♦ **AD 采样参数复位至出厂默认值函数**
函数原型：
Visual C++ & C++ Builder:
```
BOOL ResetParaAD (HANDLE hDevice,
                  PART2007_PARA_AD pADPara)
```
Visual Basic:
```
Declare Function ResetParaAD Lib "ART2007" ( ByVal hDevice As Long, _
                                             ByRef pADPara As ART2007_PARA_AD) As Boolean
```
Delphi:
```
Function ResetParaAD ( hDevice : Integer;
                       pADPara : PART2007_PARA_AD) : Boolean;
                       StdCall; External 'ART2007' Name ' ResetParaAD ';
```
LabVIEW:
请参考相关演示程序。

功能：将系统中原来的 AD 参数值复位至出厂时的默认值。以防用户不小心将各参数设置错误造成一时无法确定错误原因的后果。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。

pADPara设备硬件参数，它负责在参数被复位后返回其复位后的值。关于ART2007_PARA_AD的详细介绍请参考ART2007.h或ART2007.Bas或ART2007.Pas函数原型定义文件，也可参考本文《硬件参数结构》关于该结构的有关说明。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice LoadParaAD SaveParaAD ResetParaAD ReleaseDevice

- **读板卡基地址函数**

函数原型：

Visual C++ & C++ Builder:

```
BOOL LoadBaseAddr (HANDLE hDevice,
                   PWORD pBaseAddr)
```

Visual Basic:

```
Declare Function LoadBaseAddr Lib "ART2007" ( ByVal hDevice As Long, _
                                              ByRef pBaseAddr As Integer) As Boolean
```

Delphi:

```
Function LoadBaseAddr ( hDevice : Integer;
                        pBaseAddr: Pointer) : Boolean;
                        StdCall; External 'ART2007' Name ' LoadBaseAddr ';
```

LabVIEW:

请参考相关演示程序。

功能：将基地址从系统中读出。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pBaseAddr 指向基地址。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice SaveBaseAddr ReleaseDevice

- **保存板卡基地址函数**

函数原型：

Visual C++ & C++ Builder:

```
BOOL SaveBaseAddr (HANDLE hDevice,
                   WORD wBaseAddr)
```

Visual Basic:

```
Declare Function SaveBaseAddr Lib "ART2007" ( ByVal hDevice As Long, _
                                              ByVal wBaseAddr As Integer) As Boolean
```

Delphi:

```
Function SaveBaseAddr ( hDevice : Integer;
                        wBaseAddr : Word) : Boolean;
                        StdCall; External 'ART2007' Name ' SaveBaseAddr';
```

LabVIEW:

请参考相关演示程序。

功能：将基地址保存至系统中。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

wBaseAddr 基地址。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice LoadBaseAddr ReleaseDevice

第五节、DIO 数字量输入输出操作函数原型说明

- **开关量输入**

函数原型：

Visual C++ & C++Builder:

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

Visual Basic:

```
Declare Function GetDeviceDI Lib "ART2007" ( ByVal hDevice As Long, _
                                             ByVal bDISts(0 to 15) As Byte) As Boolean
```

Delphi:

```
Function GetDeviceDI ( hDevice : Integer;
                       bDISts : Pointer) : Boolean;
                       StdCall; External 'ART2007' Name ' GetDeviceDI ';
```

LabVIEW

请参考相关演示程序。

功能：负责将 PCI 设备上的输入开关量状态读入到 bDISts[x]数组参数中。

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 个元素，分别对应于 DI0-DI15 路开关量输入状态位。如果 bDISts[0]等于“1”则表示 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

♦ **开关量输出**

函数原型：

Visual C++ & C++Builder:

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

Visual Basic:

```
Declare Function SetDeviceDO Lib "ART2007" (ByVal hDevice As Long, _
                                            ByVal bDOSts(0 to 15) As Byte) As Boolean
```

Delphi:

```
Function SetDeviceDO (hDevice : Integer;
                      bDOSts : Pointer) : Boolean;
                      StdCall; External 'ART2007' Name ' SetDeviceDO ';
```

LabVIEW

请参考相关演示程序。

功能：负责将 PCI 设备上的输出开关量置成由 bDOSts[x]指定的相应状态。

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 个元素，分别对应于 DO0-DO15 路开关量输出状态位。比如置 DO0 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，必须对这个参数数组中的每个元素赋初值，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

♦ **回读数字量输出状态**

函数原型：

Visual C++ & C++Builder:

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

Visual Basic:

```
Declare Function RetDeviceDOLib "ART2007" (ByVal hDevice As Long, _
                                           ByVal bDOSts (0 to 15) As Byte) As Boolean
```

Delphi:

```
Function RetDeviceDO (hDevice : Integer;
                      bDOSts: Pointer) : Boolean;
                      StdCall; External 'ART2007' Name ' RetDeviceDO';
```

LabVIEW:

请参考相关演示程序。

功能：负责将 PCI 设备上的输出开关量置成由 bDOSts[x]指定的相应状态。

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 个元素，分别对应于 DO0-DO15 路开关量输出状态位。比如置 DO0 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，必须对这个参数数组中的每个元素赋初值，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

◆ **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（数字 I/O 的输入输出及 AD 采样可以同时进行，互不影响）。

第四章 硬件参数结构

AD 硬件参数结构（ART2007_PARA_AD）

Visual C++ & C++Builder：

```
typedef struct _ART2007_PARA_AD
{
    LONG FirstChannel;          // 首通道[0, 31]
    LONG LastChannel;           // 末通道[0, 31],要求末通道必须大于或等于首通道
    LONG InputRange;            // 模拟量输入量程范围
    LONG GroundingMode;         // 接地方式(单端或双端选择)
    LONG Gains;                 // 程控增益
} ART2007_PARA_AD, *PART2007_PARA_AD;
```

Visual Basic：

```
Private Type ART2007_PARA_AD
    FirstChannel As Long            ' 首通道[0, 31]
    LastChannel As Long             ' 末通道[0, 31]，要求末通道必须大于或等于首通道
    InputRange As Long              ' 模拟量输入量程范围
    GroundingMode As Long           ' 接地方式(单端或双端选择)
    Gains As Long                   ' 程控增益
End Type
```

Delphi:

```
Type   // 定义结构体数据类型
  PART2007_PARA_AD = ^ ART2007_PARA_AD;   // 指针类型结构
  ART2007_PARA_AD = record                  // 标记为记录型
        FirstChannel : LontInt;           // 首通道[0, 31]
        LastChannel : LontInt;            // 末通道[0, 31]，要求末通道必须大于或等于首通道
        InputRange : LontInt;             // 模拟量输入量程范围
        GroundingMode : LontInt;          // 接地方式(单端或双端选择)
        Gains As Long;                    // 程控增益
End;
```

LabVIEW:

请参考相关演示程序。

该结构实在太简易了，其原因就是 PCI 设备是系统全自动管理的设备，再加上驱动程序的合理设计与封装，什么端口地址、中断号、DMA 等将与 PCI 设备的用户永远告别，一句话 PCI 设备是一种更易于管理和使用的设备。

此结构主要用于设定设备AD硬件参数值，用这个参数结构对设备进行硬件配置完全由InitDeviceAD函数自动完成。用户只需要对这个结构体中的各成员简单赋值即可。

FirstChannel　AD采样首通道，其取值范围为[0, 31]，它应等于或小于LastChannel参数。
LastChannel　AD采样末通道，其取值范围为[0, 31]，它应等于或大于FirstChannel参数。

InputRange 被测模拟信号输入范围，取值如下表：

常量名	常量值	功能定义
ART2007_INPUT_N10000_P10000mV	0x00	±10000mV
ART2007_INPUT_N5000_P5000mV	0x01	±5000mV
ART2007_INPUT_N2500_P2500mV	0x02	±2500mV
ART2007_INPUT_0_P10000mV	0x03	0～10000mV

关于各个量程下采集的数据ADBuffer[]如何换算成相应的电压值，请参考《AD原码LSB数据转换成电压值的换算方法》章节。

GroundingMode AD 接地方式选择。它的选项值如下表：

常量名	常量值	功能定义
ART2007_ GNDMODE_SE	0x00	单端方式(SE:Single end)
ART2007_ GNDMODE_DI	0x01	双端方式(DI:Differential)

Gains　程控增益，即将模拟量输入放大指定倍数后再给 AD 转换器转换。其取值范围如下表：

常量名	常量值	功能定义
ART2007_GAINS_1MULT	0x00	1 倍增益
ART2007_GAINS_2MULT	0x01	2 倍增益
ART2007_GAINS_4MULT	0x02	4 倍增益
ART2007_GAINS_8MULT	0x03	8 倍增益

相关函数： CreateDevice　LoadParaAD　SaveParaAD　ReleaseDevice

第五章 数据格式转换与排列规则

第一节、AD 原码 LSB 数据转换成电压值的换算方法

首先应根据设备实际位数屏蔽掉不用的高位，然后依其所选量程，按照下表公式进行换算即可。这里只以缓冲区 ADBuffer[]中的第 1 个点 ADBuffer[0]为例。

量程(mV)	计算机语言换算公式(ANSI C 语法)	Volt 取值范围（mV）
±10000mV	Volt = (20000.0 / 8192) * (ADBuffer[0] & 0x0FFF) -10000.0	[-10000, +9997.55]
±5000mV	Volt = (10000.0 / 8192) * (ADBuffer[0] & 0x0FFF) -5000.0	[-5000, +4998.77]
±2500mV	Volt = (5000.0 / 8192) * (ADBuffer[0] & 0x0FFF) -2500.0	[-2500，+2499.38]
0~10000mV	Volt = (10000.0 / 8192) * (ADBuffer[0] & 0x0FFF)	[0，+9998.77]

下面举例说明各种语言的换算过程（以±10000mV 量程为例）

Visual C++&C++Builder：

```
Lsb = (ADBuffer[0])&0x0FFF;
Volt = (20000.00/8192) * Lsb -10000.00;
```

Visual Basic：

```
Lsb = (ADBuffer [0]) And &H0FFF
Volt = (20000.00/8192) * Lsb – 10000.00
```

Delphi:

Lsb： = (ADBuffer[0]) And $0FFF;

Volt：= (200000.0/8192) * Lsb – 10000.00;

LabVIEW:

请参考相关演示程序。

第二节、AD 采集函数的 ADBuffer 缓冲区中的数据排放规则

单通道采集，当通道总数首末通道相等时，假如此时首末通道=5，其排放规则如下：

数据缓冲区索引号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	…
通道号	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	…

两通道采集(假如FirstChannel=0, LastChannel=1)：

数据缓冲区索引号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	…
通道号	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	…

四通道采集(假如FirstChannel=0, LastChannel=3)：

数据缓冲区索引号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	…
通道号	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	…

其他通道方式以此类推。

如果用户是进行连续不间断循环采集，即用户只进行一次初始化设备操作，然后不停的从设备上读取 AD 数据，那么需要用户特别注意的是应处理好各通道数据排列和对齐的问题，尤其是在任意通道数采集时。否则，用户无法将规则排放在缓冲区中的各通道数据正确分离出来。那怎样正确处理呢？我们建议的方法是，每次从设备上读取的点数应置为所选通道数量的整数倍长，这样便能保证每读取的这批数据在缓冲区中的相应位置始终固定对应于某一个通道的数据。比如用户要求对 1、2 两个 AD 通道的数据进行连续循环采集，则置每次读取长度为其 2 的整倍长 2n(n 为每个通道的点数)，这里设为 2048。试想，如此一来，每次读取的 2048 个点中的第一个点始终对应于 1 通道数据，第二个点始终对应于 2 通道，第三个点再应于 1 通道，第四个点再对应于 2 通道……以此类推。直到第 2047 个点对应于 1 通道数据，第 2048 个点对应 2 通道。这样一来，每次读取的段长正好包含了从首通道到末通道的完整轮回，如此一来，用户只须按通道排列规则，按正常的处理方法循环处理每一批数据。而对于其他情况也是如此，比如 3 个通道采集，则可以使用 3n(n 为每个通道的点数)的长度采集。为了更加详细地说明问题，请参考下表（演示的是采集 1、2、3 共三个通道的情况）。由于使用连续采样方式，所以表中的数据序列一行的数字变化说明了数据采样的连续性，即随着时间的延续，数据的点数连续递增，直至用户停止设备为止，从而形成了一个有相当长度的连续不间断的多通道数据链。而通道序列一行则说明了随着连续采样的延续，其各通道数据在其整个数据链中的排放次序，这是一种非常规则而又绝对严格的顺序。但是这个相当长度的多通道数据链则不可能一次通过设备对象函数如 ReadDeviceAD 函数读回，即便不考虑是否能一次读完的问题，仅对于用户的实时数据处理要求来说，一次性读取那么长的数据，则往往是相当矛盾的。因此我们就得分若干次分段读取。但怎样保证既方便处理，又不易出错，而且还高效呢？还是正如前面所说，采用通道数的整数倍长读取每一段数据。如表中列举的方法 1（为了说明问题，我们每读取一段数据只读取 2n 即 3*2＝6 个数据）。从方法 1 不难看出，每一段缓冲区中的数据在相同缓冲区索引位置都对应于同一个通道。而在方法 2 中由于每次读取的不是通道整数倍长，则出现问题，从表中可以看出，第一段缓冲区中的 0 索引位置上的数据对应的是第 1 通道，而第二段缓冲区中的 0 索引位置上的数据则对应于第 2 通道的数据，而第三段缓冲区中的数据则对应于第 3 通道……，这显然不利于循环有效处理数据。

在实际应用中，我们在遵循以上原则时，应尽可能地使每一段缓冲足够大，这样，可以一定程度上减少数据采集程序和数据处理程序的 CPU 开销量。

数据序列	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	…
通道序列	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	…
方法 1	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	3	4	5	0	1	2	…
缓冲区号	第一段缓冲						第二段缓冲区						第三段缓冲区						第 n 段缓冲			
方法 2	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	3	0	…
	第一段缓冲区				第二段缓冲区				第三段缓冲区				第四段缓冲区				第五段缓冲区				第 n 段缓	

第三节、AD 测试应用程序创建并形成的数据文件格式

首先该数据文件从始端 0 字节位置开始往后至第 HeadSizeBytes 字节位置宽度属于文件头信息，而从 HeadSizeBytes 开始才是真正的 AD 数据。HeadSizeBytes 的取值通常等于本头信息的字节数大小。文件头信息包含的内容如下结构体所示。对于更详细的内容请参考 Visual C++高级演示工程中的 UserDef.h 文件。

```
typedef struct _FILE_HEADER
{
    LONG HeadSizeBytes;        // 文件头信息长度
    LONG FileType;
    // 该设备数据文件共有的成员
    LONG BusType;              // 设备总线类型(DEFAULT_BUS_TYPE)
    LONG DeviceNum;            // 该设备的编号(DEFAULT_DEVICE_NUM)

    LONG VoltBottomRange;      // 量程下限(mV)
    LONG VoltTopRange;         // 量程上限(mV)

    ART2007_PARA_AD ADPara;    // 保存硬件参数

    LONG CrystalFreq;          // 晶振频率
    LONG HeadEndFlag;          // 头信息结束位
} FILE_HEADER,  *PFILE_HEADER;
```

AD 数据的格式为 16 位二进制格式，它的排放规则与在 ADBuffer 缓冲区排放的规则一样，即每 16 位二进制(字)数据对应一个 16 位 AD 数据。您只需要先开辟一个 16 位整型数组或缓冲区，然后将磁盘数据从指定位置(即双字节对齐的某个位置)读入数组或缓冲区，然后访问数组中的每个元素，即是对相应 AD 数据的访问。

第六章 上层用户函数接口应用实例

第一节、怎样使用ReadDeviceAD函数直接取得AD数据

Visual C++ & C++Builder:

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**[ART2007 32 路 AD 和 16 路开关量卡]**丨**[Microsoft Visual C++]**丨**[简易代码演示]**丨**[AD 采集简易应用程序]**

第二节、怎样使用GetDeviceDI函数进行更便捷的数字开关量输入操作

Visual C++ & C++Builder:

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**[ART2007 32 路 AD 和 16 路开关量卡]**丨**[Microsoft Visual C++]**丨**[简易代码演示]**丨**[DIO 开关量应用程序]**

第三节、怎样使用SetDeviceDO函数进行更便捷的数字开关量输出操作

Visual C++ & C++Builder:

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**[ART2007 32 路 AD 和 16 路开关量卡]**丨**[Microsoft Visual C++]**丨**[简易代码演示]**丨**[DIO 开关量应用程序]**

第七章 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“ART2007_”）

函数名	函数功能	备注
①PC104 总线 I/O 端口操作函数		
GetDevVersion	获取设备固件及程序版本	
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
②创建 Visual Basic 子线程，线程数量可达 32 个以上		
CreateVBThread	在 VB 环境中建立子线程对象	在 VB 中可实现多线程
TerminateVBThread	终止 VB 的子线程	
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	
DelayTimeUs	高效高精度延时函数	不消耗 CPU 时间
③ 文件对象操作函数		
CreateFileObject	初始设备文件对象	
WriteFile	请求文件对象写用户数据到磁盘文件	
ReadFile	请求文件对象读数据到用户空间	
SetFileOffset	设置文件指针偏移	
GetFileLength	取得文件长度	
ReleaseFile	释放已有的文件对象	
GetDiskFreeBytes	取得指定磁盘的可用空间(字节)	适用于所有设备

第二节、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。

♦获取设备固件及程序版本

函数原型:

Visual C++ & C++ Builder:

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

Visual Basic:

```
Declare Function GetDevVersion Lib "ART2007" (ByVal hDevice As Long, _
                                              ByRef pulFmwVersion As Long, _
                                              ByRef pulDriverVersion As Long) As Boolean
```

Delphi:

```
Function GetDevVersion (hDevice : Integer;
                        pulFmwVersion: Pointer;
                        pulDriverVersion: Pointer) : Boolean;
                        StdCall; External 'ART2007' Name ' GetDevVersion ';
```

LabVIEW:

请参见相关演示程序。

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 指针参数，用于取得固件版本。
pulDriverVersion 指针参数，用于取得驱动版本。
返回值：如果执行成功，则返回 TRUE，否则会返回 FALSE。
相关函数： CreateDevice ReleaseDevice

- **以单字节(8Bit)方式写 I/O 端口**

Visual C++ & C++ Builder:

```
BOOL WritePortByte (HANDLE hDevice,
                    UINT nPort,
                    BYTE Value)
```

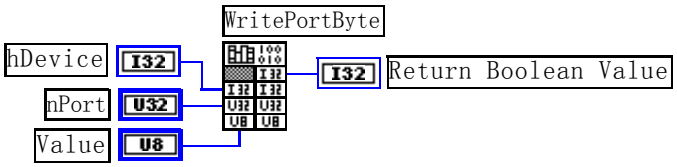
Visual Basic:

```
Declare Function WritePortByte Lib "ART2007" ( 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 'ART2007' 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
ReadPortULong ReleaseDevice

- **以双字(16Bit)方式写 I/O 端口**

Visual C++ & C++ Builder:

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

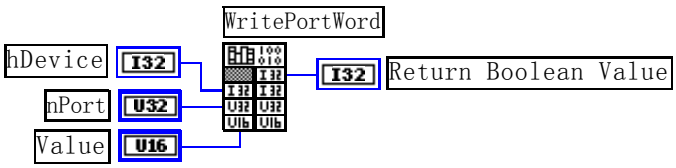
Visual Basic:

```
Declare Function WritePortWord Lib "ART2007" (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 'ART2007' 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
ReadPortULong　ReleaseDevice

- **以四字节(32Bit)方式写 I/O 端口**

Visual C++ & C++ Builder:

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

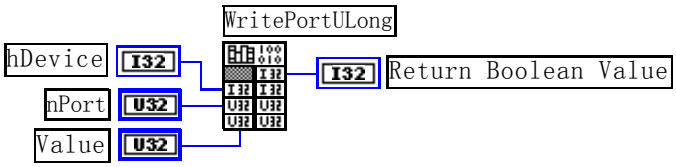
Visual Basic:

```
Declare Function WritePortULong Lib "ART2007" (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 'ART2007' 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
ReadPortULong　ReleaseDevice

- **以单字节(8Bit)方式读 I/O 端口**

Visual C++ & C++ Builder:

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

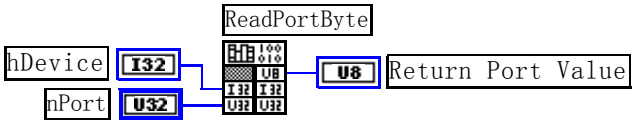
Visual Basic:

```
Declare Function ReadPortByte Lib "ART2007" (ByVal hDevice As Long, _
                                             ByVal nPort As Long ) As Byte
```

Delphi:

```
Function ReadPortByte(hDevice : Integer;
                      nPort : LongWord) : Byte;
                      StdCall; External 'ART2007' 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
ReadPortULong　ReleaseDevice

♦ **以双字节(16Bit)方式读 I/O 端口**

Visual C++ & C++ Builder:

```
WORD ReadPortWord(HANDLE hDevice,
                  UINT nPort)
```

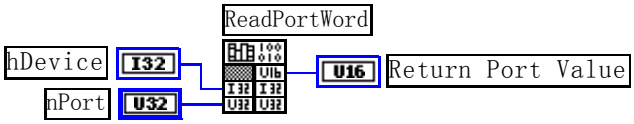
Visual Basic:

```
Declare Function ReadPortWord Lib "ART2007" ( ByVal hDevice As Long, _
                                              ByVal nPort As Long ) As Integer
```

Delphi:

```
Function ReadPortWord(hDevice : Integer;
                      nPort : LongWord) : Word;
                      StdCall; External 'ART2007' 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
ReadPortULong　ReleaseDevice

♦ **以四字节(32Bit)方式读 I/O 端口**

Visual C++ & C++ Builder:

```
ULONG ReadPortULong(HANDLE hDevice,
                    UINT nPort)
```

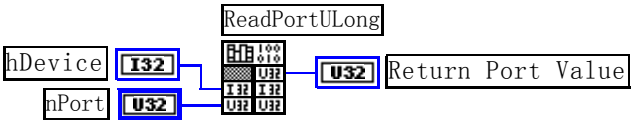
Visual Basic:

```
Declare Function ReadPortULong Lib "ART2007" ( ByVal hDevice As Long, _
                                               ByVal nPort As Long ) As Long
```

Delphi:

```
Function ReadPortULong(hDevice : Integer;
                       nPort : LongWord) : LongWord;
                       StdCall; External 'ART2007' 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　ReadPortULong　ReleaseDevice

第三节、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）

♦ **在 VB 环境中，创建子线程对象，以实现多线程操作**

函数原型：

Visual C++ & C++ Builder:

```
BOOL CreateVBThread(HANDLE *hThread,
                    LPTHREAD_START_ROUTINE RoutineAddr)
```

Visual Basic

```
Declare Function CreateVBThread Lib "ART2007" ( ByRef hThread As Long, _
                                                ByVal RoutineAddr As Long ) As Boolean
```

功能：该函数在 VB 环境中解决了不能实现或不能很好地实现多线程的问题。通过该函数用户可以很轻松地实现多线程操作。

参数：

hThread 若成功创建子线程，该参数将返回所创建的子线程的句柄，当用户操作这个子线程时将用到这个句柄，如启动线程、暂停线程以及删除线程等。

RoutineAddr 作为子线程运行的函数的地址，在实际使用时，请用AddressOf关键字取得该子线程函数的地址，再传递给CreateVBThread函数。

返回值：当成功创建子线程时，返回 TRUE，且所创建的子线程为挂起状态，用户需要用 Win32 API 函数 ResumeThread 函数启动它。若失败，则返回 FALSE。

相关函数： CreateVBThread　TerminateVBThread

注意: RoutineAddr 指向的函数或过程必须放在 VB 的模块文件中，如 ART2007.Bas 文件中。

Visual Basic 程序举例:

```
' 在模块文件中定义子线程函数(注意参考实例)
Function NewRoutine() As Long        ' 定义子线程函数
        :                            ' 线程运行代码
NewRoutine = 1    ' 返回成功码
End Function
'
' 在窗体文件中创建子线程
        :
Dim hNewThread As Long
    If Not CreateVBThread(hNewThread, AddressOf NewRoutine) Then ' 创建新子线程
        MsgBox "创建子线程失败"
        Exit Sub
    End If
    ResumeThread (hNewThread)   '启动新线程          :
     :
```

♦ **在 VB 中，删除子线程对象**

函数原型：

Visual C++ & C++ Builder:

```
BOOL TerminateVBThread(HANDLE hThread)
```

Visual Basic:

```
Declare Function TerminateVBThread Lib "ART2007" (ByVal hThread As Long) As Boolean
```

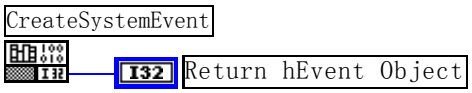
功能：在VB中删除由CreateVBThread创建的子线程对象。
参数：hThread 指向需要删除的子线程对象的句柄，它应由CreateVBThread创建。
返回值：当成功删除子线程对象时，返回 TRUE，否则返回 FALSE。
相关函数： CreateVBThread TerminateVBThread

Visual Basic 程序举例:

```
    :
If Not TerminateVBThread (hNewThread)   ' 终止子线程
    MsgBox "创建子线程失败"
    Exit Sub
End If
    :
```

◆ **创建内核系统事件**

函数原型：
Visual C++ & C++ Builder:
HANDLE CreateSystemEvent(void)
Visual Basic:
Declare Function CreateSystemEvent Lib " ART2007 " () As Long
Delphi:
Function CreateSystemEvent() : Integer;
StdCall; External 'ART2007' Name ' CreateSystemEvent ';
LabVIEW:



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)。
相关函数： ReleaseSystemEvent

◆ **释放内核系统事件**

函数原型：
Visual C++ & C++ Builder:
BOOL ReleaseSystemEvent(HANDLE hEvent)
Visual Basic:
Declare Function ReleaseSystemEvent Lib " ART2007 " (ByVal hEvent As Long) As Boolean
Delphi:
Function ReleaseSystemEvent(hEvent : Integer) : Boolean;
StdCall; External 'ART2007' Name ' ReleaseSystemEvent ';
LabVIEW:
请参见相关演示程序。

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。
相关函数： CreateSystemEvent

◆ **高效高精度延时**

函数原型：
Visual C++ & C++ Builder:
BOOL DelayTimeUs (HANDLE hDevice,
LONG nTimeUs)
Visual Basic:
Declare Function DelayTimeUs Lib "ART2007" (ByVal hDevice As Long, _
ByVal nTimeUs As Long) As Boolean
Delphi:

```
Function DelayTimeUs (hDevice: Integer;
                      nTimeUs : LongInt) : Boolean;
                      StdCall; External 'ART2007' Name ' DelayTimeUs ';
```

LabVIEW:

请参考相关演示程序。

功能：微秒级延时函数。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

nTimeUs 时间常数。单位 1 微秒。

返回值：若成功，返回 TRUE，否则返回 FALSE。

第四节、文件对象操作函数原型说明

◆ **创建文件对象**

函数原型：

Visual C++ & C++ Builder:

```
HANDLE CreateFileObject ( HANDLE hDevice,
                          LPCTSTR NewFileName,
                          int Mode)
```

Visual Basic:

```
Declare Function CreateFileObject Lib "ART2007" (ByVal hDevice As Long, _
                                                 ByVal NewFileName As String, _
                                                 ByVal Mode As Integer) As Long
```

Delphi:

```
Function CreateFileObject (hDevice : Integer;
                           NewFileName : string;
                           Mode : Integer) : Integer;
                           Stdcall; external 'ART2007' Name ' CreateFileObject ';
```

LabVIEW:

请参见相关演示程序。

功能：初始化设备文件对象， 以期待 WriteFile 请求准备文件对象进行文件操作。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

NewFileName 与新文件对象关联的磁盘文件名，可以包括盘符和路径等信息。在 C 语言中，其语法格式如：“C:\\ART2007\\Data.Dat”，在 Basic 中，其语法格式如：“C:\ART2007\Data.Dat”。

Mode 文件操作方式，所用的文件操作方式控制字定义如下(可通过或指令实现多种方式并操作)：

常量名	常量值	功能定义
ART2007_modeRead	0x0000	只读文件方式
ART2007_modeWrite	0x0001	只写文件方式
ART2007_modeReadWrite	0x0002	既读又写文件方式
ART2007_modeCreate	0x1000	如果文件不存在可以创建该文件， 如果存在，则重建此文件，且清 0
ART2007_typeText	0x4000	以文本方式操作文件

返回值：若成功，则返回文件对象句柄。

相关函数： CreateDevice　CreateFileObject　WriteFile　ReadFile　ReleaseFile　ReleaseDevice

◆ **通过设备对象，往指定磁盘上写入用户空间的采样数据**

函数原型：

Visual C++ & C++ Builder:

```
BOOL WriteFile(HANDLE hFileObject,
               PVOID pDataBuffer,
               ULONG nWriteSizeBytes)
```

Visual Basic:

```
Declare Function WriteFile Lib "ART2007" ( ByVal hFileObject As Long,_
                                           ByRef pDataBuffer As Byte,_
                                           ByVal nWriteSizeBytes As Long) As Boolean
```

Delphi:

```
Function WriteFile(hFileObject: Integer;
                   pDataBuffer : Pointer;
                   nWriteSizeBytes : LongWord) : Boolean;
                   Stdcall; external 'ART2007' Name ' WriteFile ';
```

LabVIEW:

请参考相关演示程序。

功能：通过向设备对象发送“写磁盘消息”，设备对象便会以最快的速度完成写操作。注意为了保证写入的数据是可用的，这个操作将与用户程序保持同步，但与设备对象中的环形内存池操作保持异步，以得到更高的数据吞吐量，其文件名及路径应由CreateFileObject函数中的strFileName指定。

参数：

hFileObject 设备对象句柄，它应由CreateFileObject创建。

pDataBuffer 用户数据空间地址，可以是用户分配的数组空间。

nWriteSizeBytes 告诉设备对象往磁盘上一次写入数据的长度(以字节为单位)。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关函数： CreateFileObject WriteFile ReadFile ReleaseFile

- **通过设备对象,从指定磁盘文件中读采样数据**

函数原型：

Visual C++ & C++ Builder:

```
BOOL ReadFile(HANDLE hFileObject,
              PVOID pDataBuffer,
              ULONG OffsetBytes,
              ULONG nReadSizeBytes)
```

Visual Basic:

```
Declare Function ReadFile Lib "ART2007" ( ByVal hFileObject As Long, _
                                          ByRef pDataBuffer As Integer, _
                                          ByVal OffsetBytes As Long, _
                                          ByVal nReadSizeBytes As Long) As Boolean
```

Delphi:

```
Function ReadFile( hFileObject : Integer;
                   pDataBuffer : Pointer;
                   OffsetBytes : LongWord;
                   nReadSizeBytes : LongWord) : Boolean;
                   Stdcall; external 'ART2007' Name ' ReadFile ';
```

LabVIEW:

请参考相关演示程序。

功能：将磁盘数据从指定文件中读入用户内存空间中，其访问方式可由用户在创建文件对象时指定。

参数：

hFileObject 设备对象句柄，它应由CreateFileObject创建。

pDataBuffer 用于接受文件数据的用户缓冲区指针，可以是用户分配的数组空间。

OffsetBytes 指定从文件开始端所偏移的读位置。

nReadSizeBytes 告诉设备对象从磁盘上一次读入数据的长度(以字为单位)。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关函数： CreateFileObject WriteFile ReadFile ReleaseFile

- **设置文件偏移位置**

函数原型：

Visual C++ & C++ Builder:

```
BOOL SetFileOffset (HANDLE hFileObject,
                    ULONG nOffsetBytes)
```

Visual Basic:

```
Declare Function SetFileOffset Lib "ART2007" ( ByVal hFileObject As Long,
                                               ByVal nOffsetBytes As Long) As Boolean
```

Delphi:

```
Function SetFileOffset ( hFileObject : Integer;
                         nOffsetBytes : LongWord) : Boolean;
                         Stdcall; external 'ART2007' Name ' SetFileOffset ';
```

LabVIEW:

请参考相关演示程序。

功能：设置文件偏移位置，用它可以定位读写起点。

参数：

hFileObject 文件对象句柄，它应由CreateFileObject创建。

nOffsetBytes 指定从文件开始端所偏移的读位置。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关函数： CreateFileObject WriteFile ReadFile ReleaseFile

- **取得文件长度（字节）**

函数原型：

Visual C++ & C++ Builder:

```
ULONG GetFileLength (HANDLE hFileObject);
```

Visual Basic:

```
Declare Function GetFileLength Lib "ART2007" (ByVal hFileObject As Long) As Long
```

Delphi:

```
Function GetFileLength (hFileObject : Integer) : LongWord;
                        Stdcall; external 'ART2007' Name ' GetFileLength ';
```

LabVIEW:

请参考相关演示程序。

功能：取得文件长度。

参数：hFileObject 设备对象句柄，它应由CreateFileObject创建。

返回值：若成功，则返回>1，否则返回 0。

相关函数： CreateFileObject WriteFile ReadFile ReleaseFile

- **释放设备文件对象**

函数原型：

Visual C++ & C++ Builder:

```
BOOL ReleaseFile(HANDLE hFileObject)
```

Visual Basic:

```
Declare Function ReleaseFile Lib "ART2007" (ByVal hFileObject As Long) As Boolean
```

Delphi:

```
Function ReleaseFile(hFileObject : Integer) : Boolean;
                     Stdcall; external 'ART2007' Name ' ReleaseFile ';
```

LabVIEW:

请参考相关演示程序。

功能：释放设备文件对象。

参数：hFileObject 设备对象句柄，它应由CreateFileObject创建。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关函数： CreateFileObject WriteFile ReadFile ReleaseFile

- **取得指定磁盘的可用空间**

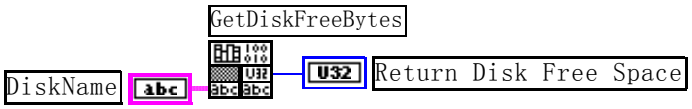
Visual C++ & C++ Builder:

ULONGLONG GetDiskFreeBytes(LPCTSTR strDiskName)

Visual Basic:

Declare Function GetDiskFreeBytes Lib "ART2007" (ByVal strDiskName As String) As Currency

LabVIEW:



功能：取得指定磁盘的可用剩余空间(以字为单位)。

参数：strDiskName 需要访问的盘符，若为 C 盘为"C:\\"， D 盘为"D:\\"，以此类推。

返回值：若成功，返回大于或等于 0 的长整型值，否则返回零值。注意使用 64 位整型变量。