ART2004 数模转换卡

WIN2000/XP 驱动程序使用说明书

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 ARTxxxx_则被省略。如 ART2004_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 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。

第二节、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，最后可以通过ReleaseDevice将hDevice释放掉。

第三节、如何实现 DA 的简便输出

当您有了 hDevice 设备对象句柄后，然后反复调用 WriteDeviceProDA 函数输出每一个 DA 数据。

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，那么GetDeviceAddr，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ART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“ART2004_”）

函数名	函数功能	备注
① 设备对象操作函数		
CreateDevice	创建 ART 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 ART 总线设备对象	上层及底层用户
DA 数据输出操作函数		
WriteDeviceDA	写 DA 数据	上层用户

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\ART2004\INCLUDE\ART2004.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 ART2004.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(ART2004.H)写进您的源程序头部。如：#include “\Art\ART2004\Include\ART2004.h”，然后再将 ART2004.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 ART2004.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 ART2004.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 ART2004.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：“ART2004”。如：

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ART2004;  // 注意： 在此加入驱动程序接口单元 ART2004
```

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 ART2004.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ReadDeviceProAD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（逻辑号）

函数原型：

Visual C++ & C++Builder：

```
HANDLE CreateDevice (int DeviceID = 0)
```

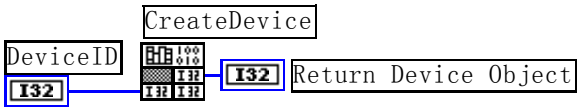
Visual Basic：

```
Declare Function CreateDevice Lib "ART2004" (Optional ByVal DeviceID As Integer = 0) As Long
```

Delphi：

```
Function CreateDevice(DeviceID : Integer = 0) : Integer;
                    StdCall; External 'ART2004' Name ' CreateDevice ';
```

LabVIEW：



功能：该函数使用逻辑号创建设备对象，并返回其设备对象句柄 hDevice。只有成功获取 hDevice，您才能实现对该设备所有功能的访问。

参数：

DeviceID 设备 ID(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

Visual C++ & C++Builder 程序举例

```
    :
HANDLE hDevice;     // 定义设备对象句柄
int DeviceLgcID = 0;
hDevice = ART2004_CreateDevice (DeviceLgcID); // 创建设备对象,并取得设备对象句柄
if(hDevice == INVALIDE_HANDLE_VALUE); // 判断设备对象句柄是否有效
  {
    return;     // 退出该函数
 }
    :
```

Visual Basic 程序举例

```
    :
Dim hDevice As Long    ' 定义设备对象句柄
Dim DeviceLgcID As Long
DeviceLgcID = 0
hDevice = ART2004_CreateDevice (DeviceLgcID)    ' 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)
```

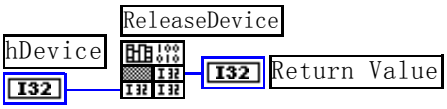
Visual Basic:

```
Declare Function ReleaseDevice Lib "ART2004" (ByVal hDevice As Long ) As Boolean
```

Delphi：

```
Function ReleaseDevice(hDevice : Integer) : Boolean;
                    StdCall; External 'ART2004' 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，用户可以用GetLastErrorEx捕获错误码。

相关函数： CreateDevice

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

第三节、DA 操作函数原型说明

♦ **输出模拟信号到指定通道**

函数原型：

Visual C++ & C++Builder:

```
BOOL WriteDeviceDA ( HANDLE hDevice,
                     WORD BaseAddr,
                     WORD nDABuffer,
                     LONG lChannel)
```

Visual Basic:

```
Declare Function WriteDeviceDA Lib "ART2004" (ByVal hDevice As Long,
                                              ByVal BaseAddr As Long,
                                              ByVal nDABuffer As Integer,_
                                              ByVal lChannel As Integer) As Boolean
```

Delphi:

```
Function WriteDeviceDA(hDevice:Integer;
                       BaseAddr : LongInt;
                       nDABuffer : SmallInt;
                       lChannel : Integer) : Boolean;
                       StdCall; External 'ART2004' Name ' WriteDeviceDA ';
```

LabVIEW:

请参考相关演示程序。

功能：设置指定通道的模拟量输出。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

BaseAddr 指定通道的基地址。

nDABuffer 指输出的 DA 原始码数据，它的取值范围为[0， 4096]。

lChannel 需要指定的模拟量通道号，其取值范围为[0，7]。

返回值：若成功，返回TRUE，否则返回FALSE，您可以调用GetLastErrorEx函数取得错误或错误字符信息。

相关函数： CreateDevice ReleaseDevice

♦ 以上函数调用一般顺序

① CreateDevice

② WriteDeviceDA

③ ReleaseDevice

用户可以反复执行第②步，以进行模拟量输出不断变化（可以和 AD 采样同时进行，互不影响）。

第四章 数据格式转换与排列规则

DA 电压值转换成 LSB 原码数据的换算方法

量程（伏）	计算机语言换算公式	Lsb 取值范围
0～5000mV	Lsb=Volt/(5000.00/4096)	[0，4095]
0～10000mV	Lsb=Volt/(10000.00/4096)	[0，4095]
±5000mV	Lsb=Volt/(10000.00/4096)+2048	[0，4095]
±10000mV	Lsb=Volt/(20000.00/4096)+2048	[0，4095]

第五章 上层用户函数接口应用实例

第一节、简易程序演示说明

怎样使用 WriteDeviceDA 函数取得 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[ART2004 数模转换卡]丨[Microsoft Visual C++]丨[简易代码演示]丨[DA 方式]

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 ART2004.h 和 ADDoc.cpp)。

[程序]丨[阿尔泰测控演示系统]丨[ART2004 数模转换卡]丨[Microsoft Visual C++]丨[高级代码演示]

其默认存放路径为：系统盘\ART\ART2004\SAMPLES\VC\ADVANCED

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“ART2004_”）

函数名	函数功能	备注
① 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
WritePortByteEx	以字节(8Bit)方式写 I/O 端口	用户程序操作端口
WritePortWordEx	以字(16Bit)方式写 I/O 端口	用户程序操作端口
WritePortULongEx	以无符号双字(32Bit)方式写 I/O 端口	用户程序操作端口
ReadPortByteEx	以字节(8Bit)方式读 I/O 端口	用户程序操作端口
ReadPortWordEx	以字(16Bit)方式读 I/O 端口	用户程序操作端口
ReadPortULongEx	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	
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、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**

Visual C++ & C++ Builder:

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

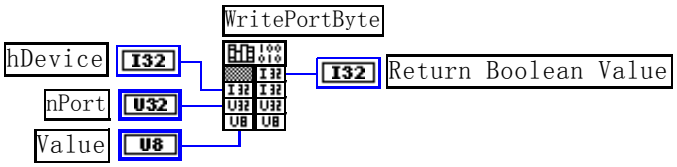
Visual Basic:

```
Declare Function WritePortByte Lib "ART2004" ( 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 'ART2004' 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。

pbPort 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： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

- **以双字(16Bit)方式写 I/O 端口**

Visual C++ & C++ Builder:

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

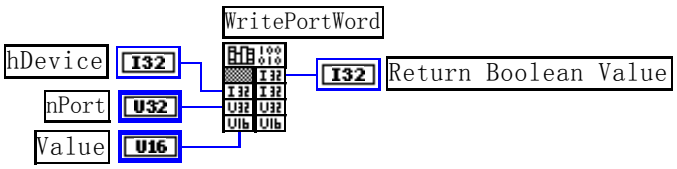
Visual Basic:

```
Declare Function WritePortWord Lib "ART2004" (ByVal hDevice As Long, _
                                              ByVal nPort As String, _
                                              ByVal Value As Integer) As Boolean
```

Delphi:

```
Function WritePortWord(hDevice : Integer;
                       nPort: LongWord;
                       Value : Word) : Boolean;
                       StdCall; External 'ART2004' 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。
pbPort 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： CreateDevice　WritePortByte　WritePortWord
WritePortULong　ReadPortByte　ReadPortWord

- **以四字节(32Bit)方式写 I/O 端口**

Visual C++ & C++ Builder:

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

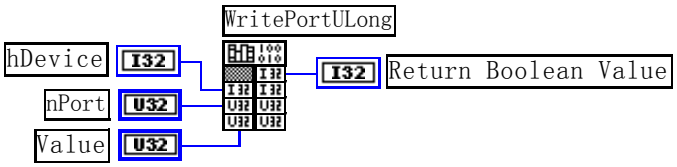
Visual Basic:

```
Declare Function WritePortULong Lib "ART2004" (ByVal hDevice As Long, _
                                               ByVal nPort As String, _
                                               ByVal Value As Long ) As Boolean
```

Delphi:

```
Function WritePortULong(hDevice : Integer;
                        nPort: LongWord;
                        Value : LongWord) : Boolean;
                        StdCall; External 'ART2004' 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。
pbPort 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： CreateDevice　WritePortByte　WritePortWord
WritePortULong　ReadPortByte　ReadPortWord

- **以单字节(8Bit)方式读 I/O 端口**

Visual C++ & C++ Builder:

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

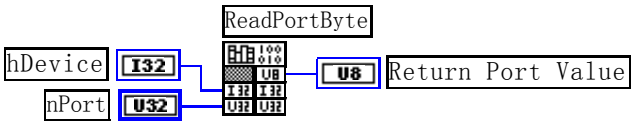
Visual Basic:

```
Declare Function ReadPortByte Lib "ART2004" (ByVal hDevice As Long, _
                                             ByVal nPort As Long) As Byte
```

Delphi:

```
Function ReadPortByte(hDevice : Integer;
                      nPort: LongWord) : Byte;
                      StdCall; External 'ART2004' 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。

pbPort 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： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

- **以双字节(16Bit)方式读 I/O 端口**

Visual C++ & C++ Builder:

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

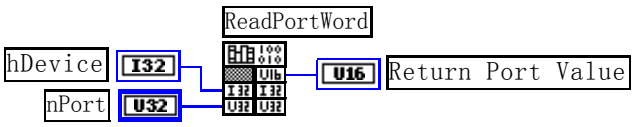
Visual Basic:

```
Declare Function ReadPortWord Lib "ART2004" ( ByVal hDevice As Long, _
                                              ByVal nPort As LongWord) As Integer
```

Delphi:

```
Function ReadPortWord(hDevice : Integer;
                      nPort: LongWord) : Word;
                      StdCall; External 'ART2004' 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。

pbPort 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： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

- **以四字节(32Bit)方式读 I/O 端口**

Visual C++ & C++ Builder:

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

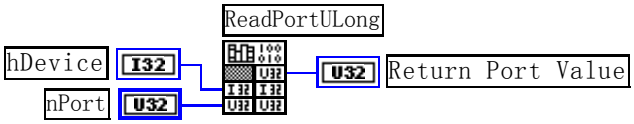
Visual Basic:

```
Declare Function ReadPortULong Lib "ART2004" ( ByVal hDevice As Long, _
                                               ByVal nPort As Long) As Long
```

Delphi:

```
Function ReadPortULong(hDevice : Integer;
                       nPort: LongWord) : LongWord;
                       StdCall; External 'ART2004' 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。
pbPort 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： CreateDevice WritePortByte WritePortWord WritePortULong ReadPortByte ReadPortWord

- **以单字节(8Bit)方式写 I/O 端口**

Visual C++ & C++ Builder:

```
BOOL WritePortByteEx (UINT nPort,
                      BYTE Value)
```

Visual Basic:

```
Declare Function WritePortByteEx Lib "ART2004" ( ByVal nPort As Long,_
                                                 ByVal Value As Byte) As Boolean
```

Delphi:

```
Function WritePortByteEx(nPort: LongWord;
                         Value : Byte) : Boolean;
                         StdCall; External 'ART2004' Name ' WritePortByteEx ';
```

LabVIEW:
请参考相关演示程序。

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 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： CreateDevice WritePortByteEx WritePortWordEx WritePortULongEx ReadPortByteEx ReadPortWordEx

- **以双字(16Bit)方式写 I/O 端口**

Visual C++ & C++ Builder:

```
BOOL WritePortWordEx(UINT nPort,
                     WORD Value)
```

Visual Basic:

```
Declare Function WritePortWordEx Lib "ART2004" (ByVal nPort As String, _
                                                ByVal Value As Integer) As Boolean
```

Delphi:

```
Function WritePortWordEx(nPort: LongWord;
                         Value : Word) : Boolean;
                         StdCall; External 'ART2004' Name ' WritePortWordEx ';
```

LabVIEW:

请参考相关演示程序。

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 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： CreateDevice WritePortByteEx WritePortWordEx WritePortULongEx ReadPortByteEx ReadPortWordEx

♦ **以四字节(32Bit)方式写 I/O 端口**

Visual C++ & C++ Builder:

```
BOOL WritePortULongEx(UINT nPort,
                      ULONG Value)
```

Visual Basic:

```
Declare Function WritePortULongEx Lib "ART2004" (ByVal nPort As String, _
                                                 ByVal Value As Long ) As Boolean
```

Delphi:

```
Function WritePortULongEx(nPort: LongWord;
                          Value : LongWord) : Boolean;
                          StdCall; External 'ART2004' Name ' WritePortULongEx ';
```

LabVIEW:

请参考相关演示程序。

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 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： CreateDevice WritePortByteEx WritePortWordEx WritePortULongEx ReadPortByteEx ReadPortWordEx

♦ **以单字节(8Bit)方式读 I/O 端口**

Visual C++ & C++ Builder:

```
BYTE ReadPortByteEx( UINT nPort)
```

Visual Basic:

```
Declare Function ReadPortByteEx Lib "ART2004" (ByVal nPort As Long) As Byte
```

Delphi:

```
Function ReadPortByteEx(nPort: LongWord) : Byte;
                        StdCall; External 'ART2004' Name ' ReadPortByteEx ';
```

LabVIEW:

请参考相关演示程序。

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 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： CreateDevice WritePortByteEx WritePortWordEx
WritePortULongEx ReadPortByteEx ReadPortWordEx

- **以双字节(16Bit)方式读 I/O 端口**

Visual C++ & C++ Builder:
WORD ReadPortWordEx(UINT nPort)
Visual Basic:
Declare Function ReadPortWordEx Lib "ART2004" (ByVal nPort As LongWord) As Integer
Delphi:
Function ReadPortWordEx(nPort: LongWord) : Word;
StdCall; External 'ART2004' Name ' ReadPortWordEx ';
LabVIEW:
请参考相关演示程序。

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 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： CreateDevice WritePortByteEx WritePortWordEx
WritePortULongEx ReadPortByteEx ReadPortWordEx

- **以四字节(32Bit)方式读 I/O 端口**

Visual C++ & C++ Builder:
ULONG ReadPortULongEx(UINT nPort)
Visual Basic:
Declare Function ReadPortULongEx Lib "ART2004" (ByVal nPort As Long) As Long
Delphi:
Function ReadPortULongEx(nPort: LongWord) : LongWord;
StdCall; External 'ART2004' Name ' ReadPortULongEx ';
LabVIEW:
请参考相关演示程序。

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 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： CreateDevice WritePortByteEx WritePortWordEx
WritePortULongEx ReadPortByteEx ReadPortWordEx

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**

Visual C++ & C++ Builder:

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

***Visual Basic*:**

```
Declare Function CreateVBThread Lib "ART2004" (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，用户可用 GetLastError 捕获当前错误码。

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 **ART2004.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

Visual C++ & C++ Builder:

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

Visual Basic:

```
Declare Function TerminateVBThread Lib "ART2004" (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，用户可用 GetLastError 捕获当前错误码。

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)

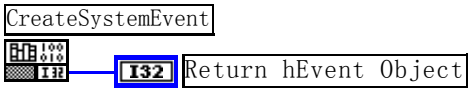
Visual Basic:

Declare Function CreateSystemEvent Lib " ART2004 " () As Long

Delphi:

Function CreateSystemEvent() : Integer;
StdCall; External 'ART2004' 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)。

♦ **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 " ART2004 " (ByVal hEvent As Long) As Boolean

Delphi:

Function ReleaseSystemEvent(hEvent : Integer) : Boolean;
StdCall; External 'ART2004' 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。

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 "ART2004" (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 'ART2004' 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:\\ART2004\\Data.Dat”，在 Basic 中，其语法格式如：“C:\ART2004\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
ART2004_modeRead	0x0000	只读文件方式
ART2004_modeWrite	0x0001	只写文件方式
ART2004_modeReadWrite	0x0002	既读又写文件方式
ART2004_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
ART2004_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 "ART2004" ( 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 'ART2004' Name ' WriteFile ';
```

LabVIEW:

详见相关演示程序。

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，用户可以用GetLastErrorEx捕获错误码。
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 "ART2004" ( 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 'ART2004' Name ' ReadFile ';
```

LabVIEW:

详见相关演示程序。

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，用户可以用GetLastErrorEx捕获错误码。
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 "ART2004" ( ByVal hFileObject As Long,
                                               ByVal nOffsetBytes As Long) As Boolean
```

Delphi:

```
Function SetFileOffset ( hFileObject : Integer;
                         nOffsetBytes : LongWord) : Boolean;
                         Stdcall; external 'ART2004' Name ' SetFileOffset ';
```

LabVIEW:

详见相关演示程序。

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。
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： 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 "ART2004" (ByVal hFileObject As Long) As Long
```
Delphi:
```
Function GetFileLength (hFileObject : Integer) : LongWord;
                        Stdcall; external 'ART2004' Name ' GetFileLength ';
```
LabVIEW:
详见相关演示程序。

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，用户可以用GetLastErrorEx捕获错误码。
相关函数： CreateFileObject WriteFile ReadFile ReleaseFile

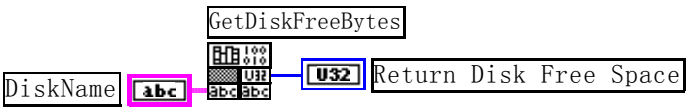
- **释放设备文件对象**

函数原型：
Visual C++ & C++ Builder:
```
BOOL ReleaseFile(HANDLE hFileObject)
```
Visual Basic:
```
Declare Function ReleaseFile Lib "ART2004" (ByVal hFileObject As Long) As Boolean
```
Delphi:
```
Function ReleaseFile(hFileObject : Integer) : Boolean;
                     Stdcall; external 'ART2004' Name ' ReleaseFile ';
```
LabVIEW:
详见相关演示程序。

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，用户可以用GetLastErrorEx捕获错误码。
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 DiskName)
```
Visual Basic:
```
Declare Function GetDiskFreeBytes Lib "ART2004" (ByVal DiskName As String ) As Currency
```
Delphi:
```
Function GetDiskFreeBytes (DiskName : String) : Currency;
                           Stdcall; external 'ART2004' Name ' GetDiskFreeBytes ';
```
LabVIEW:



功能：取得指定磁盘的可用剩余空间(以字为单位)。

参数：

DiskName 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，用户可用GetLastErrorEx捕获错误码。注意使用 64 位整型变量。