NET2801 数据采集卡

WIN2000/XP 驱动程序使用说明书

北京阿尔泰科技发展有限公司
产品研发部修订

请您务必阅读《使用纲要》，他会使您事半功倍!

目　录

提醒用户：

通常情况下，WINDOWS 系统在安装时自带的 DLL 库和驱动不全，所以您不管使用那种语言编程，请您最好先安装上 Visual C++6.0 版本的软件，方可使我们的驱动程序有更完备的运行环境。

有关设备驱动安装和产品二次发行请参考 NET2801Inst.doc 文档。

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 NETxxxx_ 则被省略。如 NET2801_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、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控制权。然后将此句柄作为参数传递给其他函数，如InitDeviceAD可以使用hDevice句柄以初始化设备的AD部件并启动AD设备，ReadDeviceAD函数可以用hDevice句柄实现对AD数据的采样批量读取，SetDeviceDO函数可用实现开关量的输出等。最后可以通过ReleaseDevice将hDevice释放掉。

第二节、如何批量取得 AD 数据

当您有了hDevice设备对象句柄后，便可用InitDeviceAD函数初始化AD部件，关于采样通道、频率等的参数的设置是由这个函数的pADPara参数结构体决定的。您只需要对这个pADPara参数结构体的各个成员简单赋

值即可实现所有硬件参数和设备状态的初始化，然后这个函数启动AD设备。接着便可用ReadDeviceAD反复读取AD数据以实现连续不间断采样当您需要关闭AD设备时，ReleaseDeviceAD便可帮您实现（但设备对象hDevice依然存在）。（注：ReadDeviceAD虽然主要面对批量读取，高速连续采集而设计，但亦可用它以少量点如 32 个点读取AD数据，以满足慢速采集需要）。具体执行流程请看下面的图 2.1.1。

注意：图中较粗的虚线表示对称关系。如红色虚线表示CreateDevice和ReleaseDevice两个函数的关系是：最初执行一次CreateDevice，在结束是就须执行一次ReleaseDevice。绿色虚线InitDeviceAD与ReleaseDeviceAD成对称方式出现。

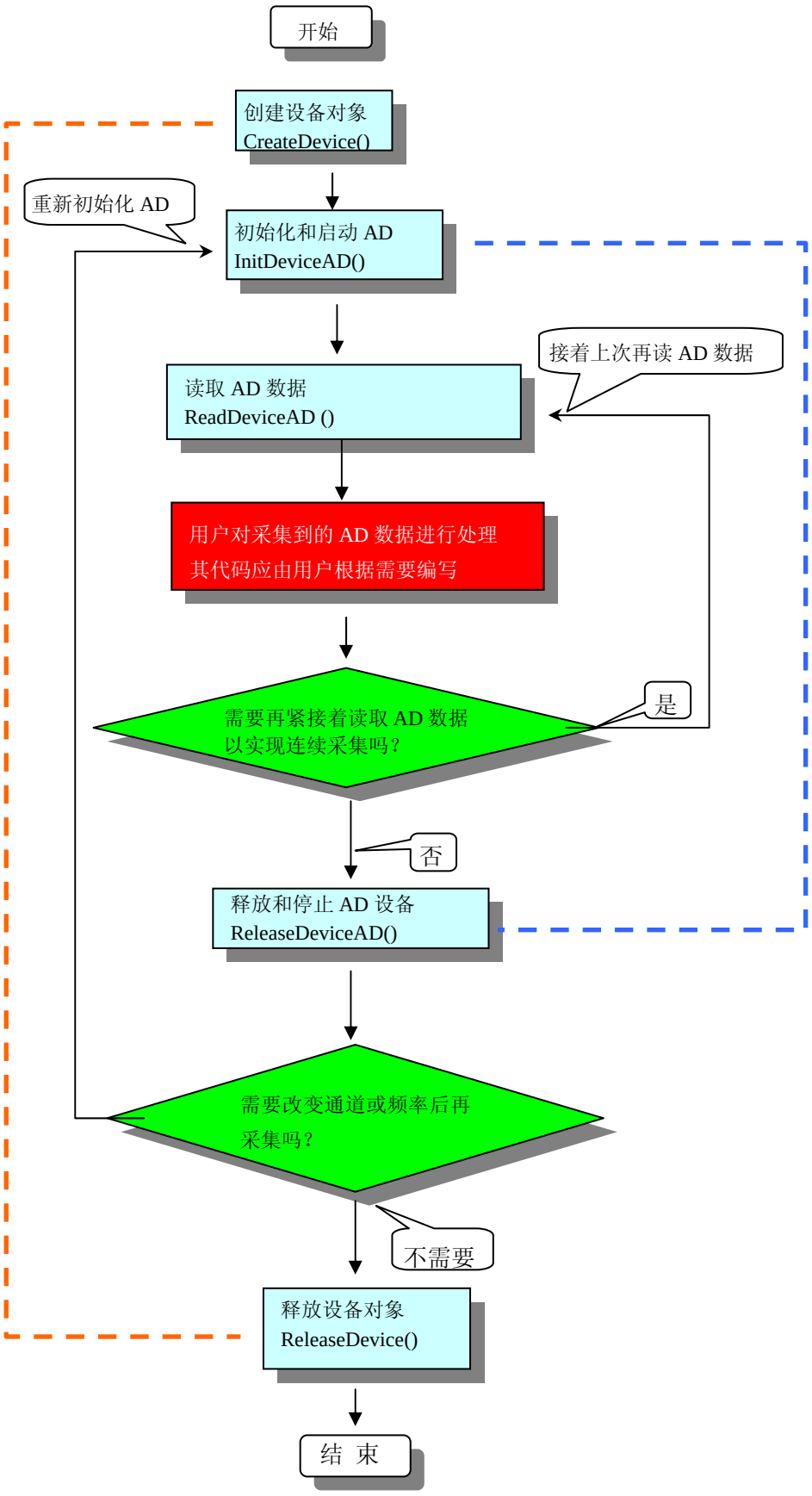


图 2.1.1 AD 采集实现过程

第三节、哪些函数对您不是必须的

当公共函数如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。

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。

第三章　设备专用函数接口介绍

第一节、设备驱动接口函数列表（每个函数省略了前缀“NET2801_”）

函数名	函数功能	备注
① 设备对象操作函数		
CreateDevice	创建设备对象	
ReleaseDevice	释放设备对象	
GetDeviceVer	获得版本	
GetNetCfg	获得设备的以太网设置参数	
SetNetCfg	设置以太网参数	
② AD 采样操作函数		
InitDeviceAD	初始化设备 AD 部件，准备传数	
StartDeviceAD	启动 AD 设备	
ReadDeviceAD	初始化设备后，即可用此函数读取设备上的 AD 数据	
ReleaseDeviceAD	释放 AD 采集，释放 AD 对象所占资源	
③ Flash 读取函数		
SetFlashPara	初始化 Flash 设备	
ReadFlashAD	在 Flash 中读取 AD	
ClearFlash	擦除 Flash	
④ 辅助函数（硬件参数设置、保存、读取函数）		
LoadParaAD	从 Windows 系统中读取硬件参数	
SaveParaAD	往 Windows 系统保存硬件参数	
ResetParaAD	将注册表中的 AD 参数恢复至出厂默认值	上层用户
SetPowerOnVal	设置上电值	
GetPowerOnVal	读取上电值	
SaveIPAddress	保存 IP 到注册表	
LoadIPAddress	载入 IP 到应用程序	
⑤ 开关量函数		
GetDeviceDI	开关输入函数	
SetDeviceDO	开关输出函数	
RetDeviceDO	回读输出开关量状态	

使用需知

Visual C++ & C++Builder：

首先将 NET2801.h 和 NET2801.lib 两个驱动库文件从相应的演示程序文件夹下复制到您的源程序文件夹中，然后在您的源程序头部添加如下语句，以便将驱动库函数接口的原型定义信息和驱动接口导入库(NET2801.lib)加入到您的工程中。

#include “NET2801.H”

在 VC 中，为了使用方便，避免重复定义和包含，您最好将以上语句放在 StdAfx.h 文件。一旦完成了以上工作，那么使用设备的驱动程序接口就跟使用 VC/C++Builder 自身的各种函数，其方法一样简单，毫无二别。

关于 NET2801.h 和 NET2801.lib 两个文件均可在演示程序文件夹下面找到。

Visual Basic:

首先将 NET2801.Bas 驱动模块头文件从 VB 的演示程序文件夹下复制到您的源程序文件夹中，然后将此模块文件加入到您的 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 NET2801.Bas 模块文件即可，一旦完成以上工作后，那么使用设备的驱动程序接口就跟使用 VB 自身的各种函数，其方法一样简单，毫无二别。

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，我们不保证能完全顺利运行。

Delphi:

首先将 NET2801.Pas 驱动模块头文件从 Delphi 的演示程序文件夹下复制到您的源程序文件夹中，然后将此模块文件加入到您的 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 NET2801.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。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：“NET2801”。如：

```
uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  NET2801;  // 注意： 在此加入驱动程序接口单元 NET2801
```

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 的驱动程序接口的详细说明请参考其演示源程序。

第二节、以太网连接使用函数原型说明

♦ 创建设备对象函数

函数原型：

Visual C++ & C++ Builder：

```
HANDLE CreateDevice (LPCSTR strIPAddr,
                     LONG lPort,
                     LONG lSendTimeout = 200,
                     LONG lRcvTimeout = 200)
```

Visual Basic：

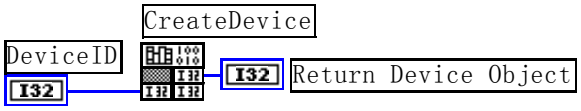
```
Declare Function CreateDevice Lib "NET2801" (ByVal strIPAddr As String,_
                                             ByVal lPort As Long,_
                                             ByVal lSendTimeout As Long = 200,_
```

ByVal lRcvTimeout As Long = 200) As Long

Delphi：

```
Function CreateDevice (strIPAddr : Pointer;
                       lPort : LongInt;
                       lSendTimeout: LongInt = 200;
                       lRcvTimeout: LongInt = 200):Integer;
                       StdCall; External 'NET2801' Name ' CreateDevice ';
```

LabView:



功能：该函数负责创建设备对象，并返回其设备对象句柄。

参数：

strIPAddr 设备 IP 地址。

lPort 端口。

lSendTimeout 发送数据的超时时间。

lRcvTimeout 接收数据的超时时间。

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

◆ **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)
```

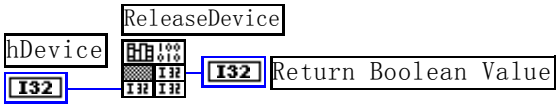
Visual Basic:

```
Declare Function ReleaseDevice Lib "NET2801" (ByVal hDevice As Long ) As Boolean
```

Delphi:

```
Function ReleaseDevice (hDevice : Integer):Boolean;
                        StdCall; External 'NET2801' 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 GetLastError 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，再一次执行这些函数前，必须执行一次ReleaseDevice函数，以释放由CreateDevice占用的系统软硬件资源，如系统内存等。只有这样，当您再次调用CreateDevice函数时，那些软硬件资源才可被再次使用。

◆ **获得版本**

函数原型：

Visual C++ & C++ Builder：

```
BOOL GetDeviceVer (HANDLE hDevice,
                   PLONG    plVerNum)
```

Visual Basic：

```
Declare Function GetDeviceVer Lib "NET2801" (ByVal hDevice As Long,_
                                             ByRef plVerNum As Long) As Boolean
```

Delphi：

```
Function GetDeviceVer (hDevice : Integer;
                       plVerNum : Pointer):Boolean;
                       StdCall; External 'NET2801' Name ' GetDeviceVer ';
```

LabView:

请参考相关演示程序。

功能：获得版本。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

plVerNum 返回版本号。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数: CreateDevice ReleaseDevice

◆ **获得设备的以太网设置参数**

函数原型：

Visual C++ & C++Builder:

```
BOOL GetNetCfg (HANDLE hDevice,
                PDEVICE_NET_INFO pNetInfo)
```

Visual Basic:

```
Declare Function GetNetCfg Lib "NET2801" (ByVal hDevice As Long, _
                                          ByRef pNetInfo As DEVICE_NET_INFO) As Boolean
```

Delphi:

```
Function GetNetCfg (hDevice : Integer;
                    pNetInfo : PDEVICE_NET_INFO):Boolean;
                    StdCall; External 'NET2801' Name ' GetNetCfg ';
```

LabView:

请参考相关演示程序。

功能：获得设备的以太网设置参数。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pNetInfo 属于 DEVICE_NET_INFO 的结构指针型，它负责返回串口设备基本信息参数，关于结构指针类型 RTU6000_DEVICE_INFO 请参考相应 NET2801.h 或该结构的帮助文档的有关说明。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数: CreateDevice SetNetCfg ReleaseDevice

◆ **设置以太网参数**

函数原型：

Visual C++ & C++Builder:

```
BOOL SetNetCfg (HANDLE hDevice,
                PDEVICE_NET_INFO NetInfo)
```

Visual Basic:

```
Declare Function SetNetCfg Lib "NET2801" (ByVal hDevice As Long, _
                                          ByVal NetInfo As DEVICE_NET_INFO) As Boolean
```

Delphi:

```
Function SetNetCfg (hDevice : Integer;
                    NetInfo: PDEVICE_NET_INFO):Boolean;
                    StdCall; External 'NET2801' Name ' SetNetCfg ';
```

LabView:

请参考相关演示程序。

功能：设置以太网参数。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

NetInfo 网络配置信息。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数: CreateDevice GetNetCfg ReleaseDevice

第三节、AD 采样操作函数原型说明

♦ 初始化设备对象

函数原型:

Visual C++ & C++Builder：

```
BOOL InitDeviceAD( HANDLE hDevice,
                   PNET2801_PARA_AD pADPara )
```

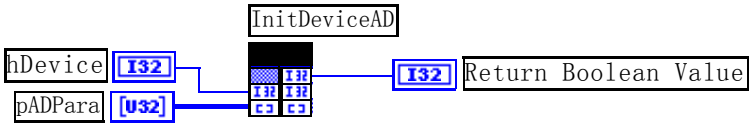
Visual Basic:

```
Declare Function InitDeviceAD Lib "NET2801" (ByVal hDevice As Long, _
                                             ByRef pADPara As NET2801_PARA _AD) As Boolean
```

Delphi:

```
Function InitDeviceAD( hDevice : Integer;
                       pADPara:PNET2801_PARA_AD):Boolean;
                       StdCall; External 'NET2801' Name 'InitDeviceAD';
```

LabView:



功能：它负责初始化设备对象中的AD部件，为设备操作就绪有关工作，如预置AD采集通道，采样频率等，然后启动AD设备开始AD采集，随后，用户便可以连续调用ReadDeviceAD读取设备上的AD数据以实现连续采集。

参数：

hDevice 设备对象句柄，它应由设备的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。请参考《AD硬件参数介绍》。

返回值：如果初始化设备对象成功，则返回 TRUE，且 AD 便被启动。否则返回 FALSE，用户可用 GetLastError 捕获当前错误码，并加以分析。

相关函数: CreateDevice ReadDeviceAD ReleaseDevice

注意：该函数将试图占用系统的某些资源，如系统内存区、DMA资源等。所以当用户在反复进行数据采集之前，只须执行一次该函数即可，否则某些资源将会发生使用上的冲突，便会失败。除非用户执行了ReleaseDeviceAD函数后，再重新开始设备对象操作时，可以再执行该函数。所以该函数切忌不要单独放在循环语句中反复执行，除非和ReleaseDeviceAD配对。

◆ **启动 AD 设备**

函数原型：

Visual C++ & C++Builder:

```
BOOL StartDeviceAD (HANDLE hDevice)
```

Visual Basic:

```
Declare Function StartDeviceAD Lib "NET2801" (ByVal hDevice As Long ) As Boolean
```

Delphi:

```
Function StartDeviceAD (hDevice : Integer):Boolean;
                        StdCall; External 'NET2801' Name ' StartDeviceAD ';
```

LabView:

请参考相关演示程序。

功能：启动 AD 设备。

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 GetLastError 捕获错误码。

相关函数：CreateDevice

◆ **初始化设备后，即可用此函数读取设备上的 AD 数据**

函数原型：

Visual C++ & C++Builder:

```
BOOL ReadDeviceAD (HANDLE hDevice,
                   USHORT ADBuffer[],
                   LONG nReadSizeWords,
                   PLONG nRetSizeWords = NULL)
```

Visual Basic:

```
Declare Function ReadDeviceAD Lib "NET2801" (ByVal hDevice As Long, _
                                             ByRef ADBuffer As Integer, _
                                             ByVal nReadSizeWords As Long, _
                                             Optional ByRef nRetSizeWords As Long = 0) As Boolean
```

Delphi:

```
Function ReadDeviceAD( hDevice : Integer;
                       ADBuffer : SmallInt;
                       nReadSizeBytes:LongWord;
                       nRetSizeWords : Pointer = 0) : Boolean;
                       StdCall; External 'NET2801' Name 'ReadDeviceAD';
```

LabView:

请参考相关演示程序。

功能：初始化设备后，即可用此函数读取设备上的AD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 用户数据缓冲区地址。接受的是从设备上采集的LSB原码数据，关于如何将LSB原码数据转换成电压值，请参考《数据格式转换与排列规则》章节。

nReadSizeWords 读取数据的长度（以字为单位），为了提高读取速率，根据特定要求，其长度必须指定为 256 字的整数倍长，如 256、512、1024 …… 8192 等字长，同时，数据长度也要为采样通道数的整数倍，以便于通道数据对齐处理，所以 nReadSizeWords 为(256*(LastChannel － FirstChannel + 1))的整数倍。否则，设备对象将失败该读操作。

nRetSizeWords 在当前操作中该函数实际读取的点数。只有当函数成功返回时该参数值才有意义，而当函数返回失败时，则该参数的值与调用此函数前的值相等，不会因为函数被调用而改变，因此最好在读取 AD 数据前，将此参数值赋初值 0。需要注意的是在函数成功返回后，若此参数值等于 0，则需要重新调用此函数读取 AD 数据，直到此参数的值不等于 0 为止。

返回值：若成功，则返回 TRUE，否则返回 FALSE，用户可以用 GetLastError 捕获错误码。

相关函数: CreateDevice InitDeviceAD ReleaseDevice

♦ 释放设备对象中的 AD 部件

函数原型：

Visual C++ & C++Builder:

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

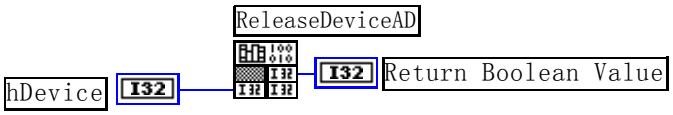
Visual Basic:

```
Declare Function ReleaseDeviceAD Lib "NET2801" (ByVal hDevice As Long ) As Boolean
```

Delphi:

```
Function ReleaseDeviceAD(hDevice : Integer):Boolean;
                    StdCall; External 'NET2801' Name 'ReleaseDeviceAD';
```

LabView:



功能：释放设备对象中的 AD 部件所占用的系统资源。

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 GetLastError 捕获错误码。

相关函数：CreateDevice InitDeviceAD ReleaseDevice

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，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。这个对应关系对于非连续采样的场合特别适用。比如用户先采集一定长度的数据后，然后对根据这些数据或其他条件，需要改变采样通道或采样频率等配置时，则可以先用ReleaseDeviceAD释放先已由InitDeviceAD占用的资源，然后再用InitDeviceAD重新分配资源和初始化设备状态，即可实现所提到的功能。

❖ 以上函数调用一般顺序

① CreateDevice
② InitDeviceAD
③ StartDeviceAD
④ ReadDeviceAD
⑤ ReleaseDeviceAD
⑥ ReleaseDevice

用户可以反复执行第④步，以实现高速连续不间断数据采集。如果在采集过程中要改变设备状态信息，如

采样通道等，则执行到第⑤步后再回到第②步用新的状态信息重新初始设备。

注意在第④步中，若其ReadDeviceAD函数成功返回，且nRetSizeWords参数值等于 0，则需要重新执行第④步，直到不等于 0 为止。

第四节、Flash 读取函数原型说明

♦ 初始化 Flash 设备

函数原型：

Visual C++ & C++Builder:

```
BOOL SetFlashPara (HANDLE hDevice,
                   const NET2801_PARA_FLASH &Para)
```

Visual Basic:

```
Declare Function SetFlashPara Lib "NET2801" (ByVal hDevice As Long, _
                                             ByRef Para As NET2801_PARA_FLASH) As Boolean
```

Delphi:

```
Function SetFlashPara ( hDevice : Integer;
                        Para : NET2801_PARA_FLASH) : Boolean;
                        StdCall; External 'NET2801' Name ' SetFlashPara ';
```

LabView:

请参考相关演示程序。

功能：初始化 Flash 设备，当返回 TRUE 后，设备即刻开始传输。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

Para Flash 参数结构体。

返回值：若成功，则返回 TRUE，否则返回 FALSE，用户可以用 GetLastError 捕获错误码。

相关函数： CreateDevice ReleaseDevice

♦ 在 Flash 中读取 AD

函数原型：

Visual C++ & C++Builder:

```
BOOL ReadFlashAD (HANDLE hDevice,
                  USHORT ADBuffer[],
                  LONG nReadSizeWords,
                  PLONG nRetSizeWords = NULL)
```

Visual Basic:

```
Declare Function ReadFlashAD Lib "NET2801" (ByVal hDevice As Long, _
                                            ByRef ADBuffer As Integer, _
                                            ByVal nReadSizeWords As Long, _
                                            Optional ByRef nRetSizeWords As Long = 0) As Boolean
```

Delphi:

```
Function ReadFlashAD ( hDevice : Integer;
                       ADBuffer : SmallInt;
                       nReadSizeBytes:LongWord;
                       nRetSizeWords : Pointer = 0) : Boolean;
                       StdCall; External 'NET2801' Name ' ReadFlashAD ';
```

LabView:

请参考相关演示程序。

功能：在 Flash 中读取 AD。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

ADBuffer 用户数据缓冲区地址。接受的是从设备上采集的LSB原码数据，关于如何将LSB原码数据转换成电压值，请参考《数据格式转换与排列规则》章节。

nReadSizeWords 读取数据的长度（以字为单位），为了提高读取速率，根据特定要求，其长度必须指定为 256 字的整数倍长，如 256、512、1024 …… 8192 等字长，同时，数据长度也要为采样通道数的整数倍，以便于通道数据对齐处理，所以 nReadSizeWords 为(256*(LastChannel － FirstChannel + 1))的整数倍。否则，设备对象将失败该读操作。

nRetSizeWords 在当前操作中该函数实际读取的点数。只有当函数成功返回时该参数值才有意义，而当函数返回失败时，则该参数的值与调用此函数前的值相等，不会因为函数被调用而改变，因此最好在读取 AD 数据前，将此参数值赋初值 0。需要注意的是在函数成功返回后，若此参数值等于 0，则需要重新调用此函数读取 AD 数据，直到此参数的值不等于 0 为止。

返回值：若成功，则返回 TRUE，否则返回 FALSE，用户可以用 GetLastError 捕获错误码。

相关函数： CreateDevice SetFlashPara ReleaseDevice

- **擦除 Flash**

函数原型：

Visual C++ & C++Builder:

```
BOOL ClearFlash (HANDLE hDevice,
                 LONG lStartBlock,
                 LONG lEndBlock)
```

Visual Basic:

```
Declare Function ClearFlash Lib "NET2801" (ByVal hDevice As Long, _
                                           ByRef lStartBlock As Integer, _
                                           ByVal lEndBlock As Long, _
                                           Optional ByRef nRetSizeWords As Long = 0) As Boolean
```

Delphi:

```
Function ClearFlash ( hDevice : Integer;
                      lStartBlock: SmallInt;
                      lEndBlock : LongWord) : Boolean;
                      StdCall; External 'NET2801' Name ' ClearFlash ';
```

LabView:

请参考相关演示程序。

功能：擦除 Flash。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

lStartBlock 起始块。

lEndBlock 结束块。

返回值：若成功，则返回 TRUE，否则返回 FALSE，用户可以用 GetLastError 捕获错误码。

相关函数： CreateDevice ReleaseDevice

第五节、AD 硬件参数系统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,
                PNET2801_PARA_AD pADPara)
```

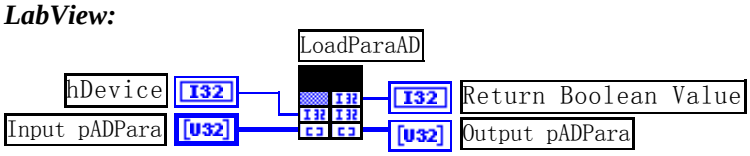
Visual Basic:

```
Declare Function LoadParaAD Lib "NET2801" (ByVal hDevice As Long, _
                                           ByRef pADPara As NET2801_PARA_AD) As Boolean
```

Delphi:

```
Function LoadParaAD( hDevice : Integer;
                     pADPara:PNET2801_PARA_AD):Boolean;
                     StdCall; External 'NET2801' Name 'LoadParaAD';
```

LabView:



功能：负责从 Windows 系统中读取设备硬件参数。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pADPara 属于 PNET2801_PARA 的结构指针型，它负责返回硬件参数值，关于结构指针类型 PNET2801_PARA 请参考相应 NET2801.h 或该结构的帮助文档的有关说明。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice SaveParaAD ReleaseDevice

Visual C++ & C++Builder 举例：

```
        :
NET2801_PARA_AD ADPara;
HANDLE   hDevice;
HDevice = CreateDevice(0) ;      // 管理秕一个设备
if(!LoadParaAD(hDevice, &ADPara))
{
      AfxMessageBox("读入硬件参数失败，请确认您的驱动程序是否正确安装");
      Return;    // 若错误，则退出该过程
}
        :
```

Visual Basic 举例：

```
        :
Dim ADPara As NET2801_PARA_AD
Dim hDevice As Long   :
 hDevice = CreateDevice(0)    ‘ 管理第一个设备
If Not LoadParaAD(hDevice, ADPara) Then
      MsgBox “读入硬件参数失败，请确认您的驱动程序是否正确安装”
      Exit Sub      ' 若错误，则退出该过程
End If
        :
```

♦ 往 Windows 系统写入设备硬件参数函数

函数原型：

Viusal C++ & C++Builder:

```
BOOL SaveParaAD(HANDLE hDevice,
                PNET2801_PARA_AD pADPara)
```

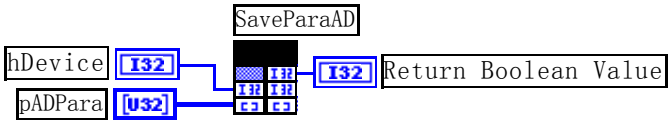
Visual Basic:

```
Declare Function SaveParaAD Lib "NET2801" (ByVal hDevice As Long, _
                                           ByRef pADPara As NET2801_PARA_AD) As Boolean
```

Delphi:

```
Function SaveParaAD (hDevice : Integer;
                     pADPara:PNET2801_PARA_AD):Boolean;
                     StdCall; External 'NET2801' Name 'SaveParaAD';
```

LabView:



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 AD设备硬件参数，请参考《硬件参数结构》章节。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice LoadParaAD ReleaseDevice

♦ **AD 采样参数复位至出厂默认值函数**

函数原型：

Viusal C++ & C++ Builder:

```
BOOL ResetParaAD (HANDLE hDevice,
                  PNET2801_PARA_AD pADPara)
```

Visual Basic:

```
Declare Function ResetParaAD Lib "NET2801" (ByVal hDevice As Long, _
                                            ByRef pADPara As NET2801_PARA_AD) As Boolean
```

Delphi:

```
Function ResetParaAD ( hDevice : Integer;
                       pADPara : PNET2801_PARA_AD) : Boolean;
                       StdCall; External 'NET2801' 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NET2801_PARA_AD的详细介绍请参考NET2801.h或NET2801.Bas或NET2801.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

注意：在您编写工程应用软件时，若要更方便的保存和读取您特有的软件参数，请不防使用我们为您提供的辅助函数：SaveParaInt、LoadParaInt、SaveParaString、LoadParaString，详细说明请参考共用函数介绍章节中的《其他函数原型说明》。

♦ **设置上电值**

函数原型：

Viusal C++ & C++ Builder:

```
BOOL SetPowerOnVal (HANDLE hDevice,
                    PNET2801_POWERON_PARA pADPara)
```

Visual Basic:

```
Declare Function SetPowerOnVal Lib "NET2801" (ByVal hDevice As Long, _
                                              ByRef pADPara As NET2801_POWERON_PARA) As Boolean
```

Delphi:

```
Function SetPowerOnVal ( hDevice : Integer;
                         pADPara : PNET2801_POWERON_PARA) : Boolean;
                         StdCall; External 'NET2801' Name ' SetPowerOnVal ';
```

LabVIEW:

请参考相关演示程序。

功能：设置上电值。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pADPara保存设备上电值参数。关于NET2801_POWERON_PARA的详细介绍请参考NET2801.h或NET2801.Bas或NET2801.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 GetPowerOnVal ReleaseDevice

♦ **读取上电值**

函数原型：

Viusal C++ & C++ Builder:

```
BOOL GetPowerOnVal (HANDLE hDevice,
                    PNET2801_POWERON_PARA pADPara)
```

Visual Basic:

```
Declare Function GetPowerOnVal Lib "NET2801" (ByVal hDevice As Long, _
                                              ByRef pADPara As NET2801_POWERON_PARA) As Boolean
```

Delphi:

```
Function GetPowerOnVal ( hDevice : Integer;
                         pADPara : PNET2801_POWERON_PARA) : Boolean;
                         StdCall; External 'NET2801' Name ' GetPowerOnVal ';
```

LabVIEW:

请参考相关演示程序。

功能：读取上电值。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

pADPara保存设备上电值参数。关于NET2801_POWERON_PARA的详细介绍请参考NET2801.h或NET2801.Bas或NET2801.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 SetPowerOnVal ReleaseDevice

- **保存 IP 到注册表**

函数原型：

Viusal C++ & C++ Builder:

```
BOOL SaveIPAddress (char szIP[],
                    LONG lPort)
```

Visual Basic:

```
Declare Function SaveIPAddress Lib "NET2801" (ByRef szIP As String, _
                                              ByVal lPort As Long) As Boolean
```

Delphi:

```
Function SaveIPAddress (szIP : char;
                        lPort : LongInt) : Boolean;
                        StdCall; External 'NET2801' Name ' SaveIPAddress ';
```

LabVIEW:

请参考相关演示程序。

功能：保存 IP 到注册表。

参数：

szIP IP 地址。

lPort 端口号。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice LoadIPAddress ReleaseDevice

- **载入 IP 到应用程序**

函数原型：

Viusal C++ & C++ Builder:

```
BOOL LoadIPAddress (char szIP[],
                    PLONG lPort)
```

Visual Basic:

```
Declare Function LoadIPAddress Lib "NET2801" (ByRef szIP As String, _
                                              ByRef lPort As Long) As Boolean
```

Delphi:

```
Function LoadIPAddress (szIP : char;
                        lPort : Pointer) : Boolean;
                        StdCall; External 'NET2801' Name ' LoadIPAddress ';
```

LabVIEW:

请参考相关演示程序。

功能：载入 IP 到应用程序。

参数：

szIP IP 地址。

lPort 端口号。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice SaveIPAddress ReleaseDevice

第六节、DIO 数字开关量输入输出简易操作函数原型说明

♦ **十六路开关量输入**

函数原型：

Visual C++ & C++Builder:

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

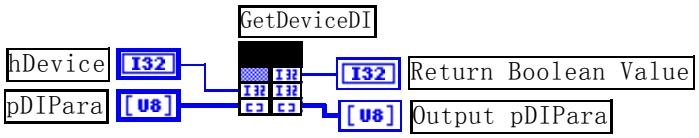
Visual Basic:

```
Declare Function GetDeviceDI Lib "NET2801" (ByVal hDevice As Long, _
                                            ByVal bDISts(0 to 15 ) As Long) As Boolean
```

Delphi:

```
Function GetDeviceDI ( hDevice : Integer;
                       bDISts:Pointer):Boolean;
                       StdCall; External 'NET2801' Name ' GetDeviceDI ';
```

LabView：



功能：负责将设备上的输入开关量状态读入内存。

参数：

hDevice 设备对象句柄，它应由CreateDevice决定。

bDISts十六路开关量输入状态的参数结构，共有 16 个成员变量，分别对应于DI0-DI15 路开关量输入状态位。如果bDISts[0]为“1”则使 0 通道处于开状态，若为“0”则 0 通道为关状态。其他同理。具体定义请参考《DI数字开关量输入参数介绍》章节。

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

♦ **十六路开关量输出**

函数原型：

Visual C++ & C++Builder:

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

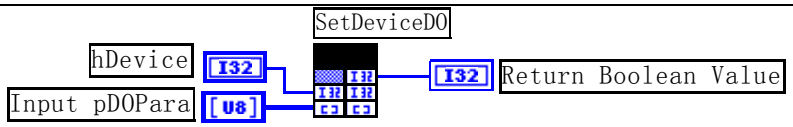
Visual Basic:

```
Declare Function SetDeviceDO Lib "NET2801" ( ByVal hDevice As Long, _
                                             ByVal bDOSts(0 to 15 )As Long) As Boolean
```

Delphi:

```
Function SetDeviceDO (hDevice : Integer;
                      bDOSts:Pointer):Boolean;
                      StdCall; External 'NET2801' Name ' SetDeviceDO ';
```

LabView：



功能：负责将设备上的输出开关量置成相应的状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice决定。

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

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 "NET2801" (ByVal hDevice As Long, _
                                           ByVal DOSts(0 to 15) As Byte) As Boolean
```

Delphi:

```
Function RetDeviceDO (hDevice : Integer;
                      DOSts : Pointer) : Boolean;
                      StdCall; External 'NET2801' Name ' RetDeviceDO';
```

LabVIEW:

请参考相关演示程序。

功能：负责将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[x] 获得开关输出状态(注意: 必须定义为 16 个字节元素的数组)。

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

第一节、AD 硬件参数介绍（NET2801_PARA_AD）

Visual C++ & C++Builder:

```
typedef struct _NET2801_PARA_AD        // 板卡各参数值
{
    LONG CheckStsMode;          // 检查状态模
    LONG ADMode;                // AD 模式选择(连续采集/分组采集)
    LONG FirstChannel;          // 首通道,取值范围为[0, 31]
    LONG LastChannel;           // 末通道,取值范围为[0, 31]
    LONG Frequency;             // 采集频率,单位为 Hz ,取值范围为[31, 250000]
    LONG GroupInterval;         // 分组采样时的组间间隔(单位：微秒) ,取值范围为[1, 32767]
    LONG LoopsOfGroup;          // 分组采样时，每组循环次数，取值范围为[1, 255]
    LONG Gains;                 // 增益控制字
    LONG TriggerMode;           // 触发模式选择(软件触发、后触发)
    LONG TriggerDir;            // 触发方向选择(正向/负向触发)
} NET2801_PARA_AD, *PNET2801_PARA_AD;
```

Visual Basic：

```
Private Type NET2801_PARA_AD
    CheckStsMode As Long        ' 检查状态模式
    ADMode As Long              ' AD 模式选择(连续采集/分组采集)
    FirstChannel As Long        ' 首通道,取值范围为[0, 31]
    LastChannel As Long         ' 末通道,取值范围为[0, 31]
    Frequency As Long           ' AD 采样频率，单位 Hz ,取值范围为[31, 250000]
    GroupInterval As Long       ' 分组间隔，单位微秒,取值范围为[1, 32767]
    LoopsOfGroup As Long        ' 分组采样时，每组循环次数，取值范围为[1, 255]
    Gains As Long               ' 增益控制字
    TriggerMode As Long         ' 触发模式选择(软件触发、后触发)
    TriggerDir As Long          ' 触发方向选择(正向/负向触发方向)
End Type
```

Delphi:

```
Type   // 定义结构体数据类型
    PNET2801_PARA_AD = ^NET2801_PARA_AD;   // 指针类型结构
    NET2801_PARA_AD = record          // 标记为记录型
        CheckStsMode : LongInt;       // 检查状态模式
        ADMode : LongInt;             // AD 模式选择(连续采集/分组采集)
        FirstChannel : LongInt;       // 首通道,取值范围为[0, 31]
        LastChannel : LongInt;        // 末通道,取值范围为[0, 31]
        Frequency : LongInt;          // 采集频率,单位为 Hz ,取值范围为[31, 250000]
        GroupInterval : LongInt;      // 分组采样时的组间间隔(单位：微秒) ,取值范围为[1, 32767]
        LoopsOfGroup : LongInt;       // 分组采样时，每组循环次数，取值范围为[1, 255]
        Gains : LongInt;              // 增益控制字
```

```
    TriggerMode : LongInt;          // 触发模式选择(软件触发、后触发)
    TriggerDir : LongInt;           // 触发方向选择(正向/负向触发)
End;
```

LabView:

首先请您关注一下这个结构与前面 ISA 总线部分中的硬件参数结构 PARA 比较，该结构实在太简短了。其原因就是在于设备是系统全自动管理的设备，什么端口地址，中断号，DMA 等将与设备的用户永远告别，一句话设备简单得就象使用电源插头一样。

硬件参数说明：此结构主要用于设定设备硬件参数值，用这个参数结构对设备进行硬件配置完全由InitDeviceAD函数完成。

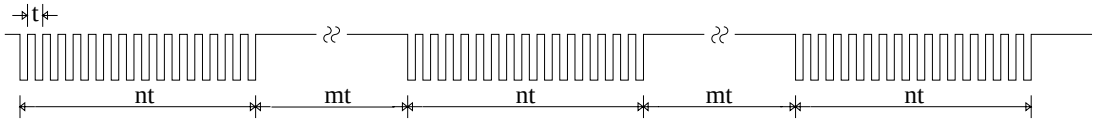
CheckStsMode 查询 FIFO 状态的模式，它用于同步 AD 的采样时序。由于有不同的采样频率，因此写入 FIFO 数据的速度也不一样，所以需要依据 FIFO 的半满或者是非空标志同步对 FIFO 数据的读取。如果选择半满查询模式，则在启动 AD 采样后不能马上读取 FIFO 中的数据，而是要等到 FIFO 的半满状态建立时才能读取数据，此种模式用于大批量高速采样情况下(实时性要求不大)。如果选择非空查询模式，则在启动 AD 采样后，只要 FIFO 中不为空，那怕只有一个数据时也能读取，此种模式用于较低速，但快速取得小批量数据的情况下(实时性要求较高)。

常量名	常量值	功能定义
NET2801_CHKSTSMODE_HALF	0x0000	查询 FIFO 半满标志(建议高频率采集时使用)
NET2801_CHKSTSMODE_NPT	0x0001	查询 FIFO 非空标志(建议高实时采集时使用)

ADMode AD 采样模式。它的取值如下表：

常量名	常量值	功能定义
NET2801_ADMODE_SEQUENCE	0x0000	连续采集模式
NET2801_ADMODE_GROUP	0x0001	分组采集模式

连续采集方式的情况是：在由Frequency指定的采样频率下所采集的全部数据在时间轴上是等简隔的，比如将Frequency指定为 100KHz， 即每隔 10 微秒采样一个点，总是这样重复下去。而分组采集方式的情况是：所有采集的数据点在时间轴上被划分成若干个等长的组，而组内通常有大于 2 个点的数据，组内各点频率由Frequency决定，组间间隔由GroupInterval决定。比如用户要求在对 0-15 通道共 16 个通道用 100KHz频率每采集一个轮回后，要求间隔 1 毫秒后，再对这 16 个通道采集下一个轮回，那么分组采集便是最佳方式，它可以将组间延时误差控制在 0.5 微秒以下。关于分组与连续采集更详细的说明请参考硬件说明书。如下图：



其中：t为所需触发A/D转换的周期Cycle，它由Frequency参数的倒数决定。Cycle = 1 / Frequency

n为每组的通道数ChannelCount决定，即ChannelCount = LastChannel-FirstChannel+1

nt为每组操作所需时间即 Cycle * ChannelCount * LoopsofGroup (此处假定LoopsofGroup=1)

mt为每组操作之间所间隔的时间，它由GroupInterval参数决定，单位为1uS。

FirstChannel 首通道值，取值范围应根据设备的总通道数设定，本设备的 AD 采样首通道取值范围为 0～31，要求首通道等于或小于末通道。

LastChannel 末通道值，取值范围应根据设备的总通道数设定，本设备的 AD 采样首通道取值范围为 0～31，要求末通道大于或等于首通道。

注：当首通道和末通道相等时，即为单通道采集。

Frequency AD 采样频率，本设备的频率取值范围为[31Hz, 250kHz]。注意：

在内时钟(即ClockSource = NET2801_CLOCKSRC_IN)方式下：

若连续采集(即ADMode = NET2801_ADMODE_SEQUENCE)时，此参数控制各处通道间的采样频率。若分组采集(即ADMode= NET2801_ADMODE_GROUP)时，则此参数控制各组组内的采样频率，而组间时间则由GroupInterval决定。

在外时钟(即ClockSource = NET2801_CLOCKSRC_OUT)方式下：

若连续采集(即ADMode = NET2801_ADMODE_SEQUENCE)时，此参数自动失效，因为外时钟的频率代替了此参数设定的频率。若为分组采集(即ADMode= NET2801_ADMODE_GROUP)时，则该参数控制各组组内的采样频率，而外时钟则是每组的触发频率。此时，GroupInterval参数无效。

GroupInterval 分组间隔，指定两组间的时间间隔，单位微秒，取值范围为[1, 32767]。

LoopsOfGroup 组循环次数(在分组采集时有效)，即指每一组采集从首通道依次采集到末通道后再回到首通道的次数，取值范围为[1, 255]。比如=1 则表示从首通道采集到末通道后则自动进入分组间隔准备采集下一组。若=2，则表示从首通道采集到末通道后再回到首通道采集到末通道一遍后进入组间间隔，依此类推。

Gains 程控增益放大倍数，被采样的外界信号经通道开关选通后进入一个程控增益放大器如 AD8251 芯片，它可以将原始模拟信号放大指定倍数后再进入 AD 转换器被转换。

常量名	常量值	功能定义
NET2801_GAINS_1MULT	0x0000	1 倍增益(使用 AD8251 放大器)/ 1 倍增益(使用 AD8253 放大器)/ 1 倍增益(使用 AD8250 放大器)
NET2801_GAINS_2MULT	0x0001	2 倍增益(使用 AD8251 放大器)/ 10 倍增益(使用 AD8253 放大器)/ 2 倍增益(使用 AD8250 放大器)
NET2801_GAINS_4MULT	0x0002	4 倍增益(使用 AD8251 放大器)/ 100 倍增益(使用 AD8253 放大器)/ 5 倍增益(使用 AD8250 放大器)
NET2801_GAINS_8MULT	0x0003	8 倍增益(使用 AD8251 放大器)/ 1000 倍增益(使用 AD8253 放大器)/ 10 倍增益(使用 AD8250 放大器)

TriggerMode AD触发模式，若等于常量NET2801_TRIGMODE_SOFT则为内部软件触发，若等于常量NET2801_TRIGMODE_POST则为外部硬件后触发。两种方式的主要区别是：外触发是当设备被InitDeviceAD函数初始化就绪后，并没有立即启动AD采集，仅当外接管脚ATR上有一个符合要求的信号时，AD转换器便被启动，且按用户预先设定的采样频率由板上的硬件定时器时定触发AD等间隔转换每一个AD数据，其触发条件由触发类型和触发方向及触发电平决定。

常量名	常量值	功能定义
NET2801_TRIGMODE_SOFT	0x0000	软件内触发方式
NET2801_TRIGMODE_POST	0x0001	硬件后触发方式

TriggerDir AD 外触发方式使用信号方向。它的其选项值如下表：

常量名	常量值	功能定义
NET2801_TRIGDIR_NEGATIVE	0x0000	负向触发(低脉冲/下降沿触发)
NET2801_TRIGDIR_POSITIVE	0x0001	正向触发(高脉冲/上升沿触发)
NET2801_TRIGDIR_POSIT_NEGAT	0x0002	正负向触发(高/低脉冲或上升/下降沿触发)

相关函数: InitDeviceAD LoadParaAD SaveParaAD

第二节、Flash 配置结构参数介绍（NET2801_PARA_FLASH）

Visual C++ & C++Builder:

```
typedef struct _NET2801_PARA_FLASH
{
    LONG FirstChannel;     // 首通道,取值范围为[0, 31]
    LONG LastChannel;      // 末通道,取值范围为[0, 31]
    LONG lStartPage;       // Flash 起始页
    LONG lStartBlock;      // Flash 起始块
} NET2801_PARA_FLASH, *PNET2801_PARA_FLASH;
```

Visual Basic：

```
Private Type NET2801_PARA_FLASH
    FirstChannel As Long
    LastChannel As Long
    lStartPage As Long
    lStartBlock As Long
End Type
```

Delphi:

```
Type   // 定义结构体数据类型
    PNET2801_PARA_FLASH = ^ NET2801_PARA_FLASH;     // 指针类型结构
    DEVICE_NET_INFO = record                        // 标记为记录型
        FirstChannel : LongInt;
        LastChannel : LongInt;
        lStartPage : LongInt;
        lStartBlock: LongInt;
    End;
```

硬件参数说明:

FirstChannel 首通道值，取值范围应根据设备的总通道数设定，本设备的 AD 采样首通道取值范围为 0～31，要求首通道等于或小于末通道。

LastChannel 末通道值，取值范围应根据设备的总通道数设定，本设备的 AD 采样首通道取值范围为 0～31，要求末通道大于或等于首通道。

注：当首通道和末通道相等时，即为单通道采集。

lStartPage Flash 起始页。

lStartBlock Flash 起始块。

第三节、保存设备上电值参数结构体介绍（NET2801_POWERON_PARA）

Visual C++ & C++Builder:

```
typedef struct _NET2801_POWERON_PARA
{
    LONG lDO[16];              // DO
    LONG CheckStsMode;         // 检查状态模式
    LONG ADMode;               // AD 模式选择(连续采集/分组采集)
    LONG CollectionMode;       // 采集方式(0：定长 Flash 采集/1：连续 FIFO 采集)
```

```
    LONG FirstChannel;          // 首通道,取值范围为[0, 31]
    LONG LastChannel;           // 末通道,取值范围为[0, 31]
    LONG Frequency;             // 采集频率,单位为 Hz,取值范围为[31, 250000]
    LONG GroupInterval;         // 分组采样时的组间间隔(单位：微秒),取值范围为[1, 32767]
    LONG LoopsOfGroup;          // 分组采样时，每组循环次数，取值范围为[1, 255]
    LONG Gains;                 // 增益控制字
    LONG TriggerMode;           // 触发模式选择(软件触发、后触发)
    LONG TriggerDir;            // 触发方向选择(正向/负向触发)
    LONG lStartPage;            // 起始页
    LONG lStartBlock;           // 起始块
    LONG lEndPage;              // 结束页
    LONG lEndBlock;             // 结束块
} NET2801_POWERON_PARA, *PNET2801_POWERON_PARA;
```

Visual Basic：

```
Private Type NET2801_PARA_AD
    lDO (0 to 15) As Long           ' DO
    CheckStsMode As Long            ' 查状态模式
    ADMode As Long                  ' AD 模式选择(连续采集/分组采集)
    CollectionMode As Long          ' 采集方式(0：定长 Flash 采集/1：连续 FIFO 采集)
    FirstChannel As Long            ' 首通道,取值范围为[0, 31]
    LastChannel As Long             ' 末通道,取值范围为[0, 31]
    Frequency As Long               ' AD 采样频率，单位 Hz ,取值范围为[31, 250000]
    GroupInterval As Long           ' 分组间隔，单位微秒,取值范围为[1, 32767]
    LoopsOfGroup As Long            ' 分组采样时，每组循环次数，取值范围为[1, 255]
    Gains As Long                   ' 增益控制字
    TriggerMode As Long             ' 触发模式选择(软件触发、后触发)
    TriggerDir As Long              ' 触发方向选择(正向/负向触发方向)
    lStartPage As Long              ' 起始页
    lStartBlock As Long             ' 起始块
    lEndPage As Long                ' 结束页
    lEndBlock As Long               ' 结束块
End Type
```

Delphi:

```
Type   // 定义结构体数据类型
    PNET2801_PARA_AD = ^NET2801_PARA_AD;   // 指针类型结构
    NET2801_PARA_AD = record            // 标记为记录型
        Ldo : Array[0..15] of LongInt;  // DO
        CheckStsMode : LongInt;         // 检查状态模式
        ADMode : LongInt;               // AD 模式选择(连续采集/分组采集)
        CollectionMode : LongInt;       // 采集方式(0：定长 Flash 采集/1：连续 FIFO 采集)
        FirstChannel : LongInt;         // 首通道,取值范围为[0, 31]
        LastChannel : LongInt;          // 末通道,取值范围为[0, 31]
        Frequency : LongInt;            // 采集频率,单位为 Hz ,取值范围为[31, 250000]
        GroupInterval : LongInt;        // 分组采样时的组间间隔(单位：微秒) ,取值范围为[1, 32767]
```

```
    LoopsOfGroup : LongInt;        // 分组采样时，每组循环次数，取值范围为[1, 255]
    Gains : LongInt;               // 增益控制字
    TriggerMode : LongInt;         // 触发模式选择(软件触发、后触发)
    TriggerDir : LongInt;          // 触发方向选择(正向/负向触发)
    lStartPage: LongInt;           // 起始页
    lStartBlock: LongInt;          // 起始块
    lEndPage: LongInt;             // 结束页
    lEndBlock: LongInt;            // 结束块
End;
```

LabView:

硬件参数说明:

lDO DO 通道。

CheckStsMode 查询 FIFO 状态的模式。

常量名	常量值	功能定义
NET2801_CHKSTSMODE_HALF	0x0000	查询 FIFO 半满标志(建议高频率采集时使用)
NET2801_CHKSTSMODE_NPT	0x0001	查询 FIFO 非空标志(建议高实时采集时使用)

ADMode AD 采样模式。它的取值如下表：

常量名	常量值	功能定义
NET2801_ADMODE_SEQUENCE	0x0000	连续采集模式
NET2801_ADMODE_GROUP	0x0001	分组采集模式

CollectionMode 采集方式(0：定长 Flash 采集/1：连续 FIFO 采集)。

FirstChannel 首通道值，取值范围应根据设备的总通道数设定，本设备的 AD 采样首通道取值范围为 0～31，要求首通道等于或小于末通道。

LastChannel 末通道值，取值范围应根据设备的总通道数设定，本设备的 AD 采样首通道取值范围为 0～31，要求末通道大于或等于首通道。

注：当首通道和末通道相等时，即为单通道采集。

Frequency AD 采样频率，本设备的频率取值范围为[31Hz, 250kHz]。注意：

GroupInterval 分组间隔，指定两组间的时间间隔，单位微秒，取值范围为[1, 32767]。

LoopsOfGroup 组循环次数(在分组采集时有效)，即指每一组采集从首通道依次采集到末通道后再回到首通道的次数，取值范围为[1, 255]。比如=1 则表示从首通道采集到末通道后则自动进入分组间隔准备采集下一组。若=2，则表示从首通道采集到末通道后再回到首通道采集到末通道一遍后进入组间间隔，依此类推。

Gains 程控增益放大倍数，被采样的外界信号经通道开关选通后进入一个程控增益放大器如 AD8251 芯片，它可以将原始模拟信号放大指定倍数后再进入 AD 转换器被转换。

常量名	常量值	功能定义
NET2801_GAINS_1MULT	0x0000	1 倍增益(使用 AD8251 放大器)/ 1 倍增益(使用 AD8253 放大器)/ 1 倍增益(使用 AD8250 放大器)
NET2801_GAINS_2MULT	0x0001	2 倍增益(使用 AD8251 放大器)/ 10 倍增益(使用 AD8253 放大器)/ 2 倍增益(使用 AD8250 放大器)
NET2801_GAINS_4MULT	0x0002	4 倍增益(使用 AD8251 放大器)/

		100 倍增益(使用 AD8253 放大器)/ 5 倍增益(使用 AD8250 放大器)
NET2801_GAINS_8MULT	0x0003	8 倍增益(使用 AD8251 放大器)/ 1000 倍增益(使用 AD8253 放大器)/ 10 倍增益(使用 AD8250 放大器)

TriggerMode AD 触发模式。

常量名	常量值	功能定义
NET2801_TRIGMODE_SOFT	0x0000	软件内触发方式
NET2801_TRIGMODE_POST	0x0001	硬件后触发方式

TriggerDir AD 外触发方式使用信号方向。它的其选项值如下表：

常量名	常量值	功能定义
NET2801_TRIGDIR_NEGATIVE	0x0000	负向触发(低脉冲/下降沿触发)
NET2801_TRIGDIR_POSITIVE	0x0001	正向触发(高脉冲/上升沿触发)
NET2801_TRIGDIR_POSIT_NEGAT	0x0002	正负向触发(高/低脉冲或上升/下降沿触发)

lStartPage 起始页。
lStartBlock 起始块。
lEndPage 结束页。
lEndBlock 结束块。
相关函数: InitDeviceAD LoadParaAD SaveParaAD

第四节、设备网络配置结构参数介绍（DEVICE_NET_INFO）

Visual C++ & C++Builder:

```
typedef struct _DEVICE_NET_INFO
{
    char strIP [16];                    // IP 地址, "192.168.2.70"
    char strSubnetMask [16];            // 子网掩码, "255.255.255.255"
    char strGateway [16];               // 网关, "192.168.2.1"
    char strMAC [20];                   // 网卡物理地址,  "00-01-02-03-04-05",用户一般不可修改
    WORD wTCPPort;                      // TCP 端口
}DEVICE_NET_INFO, *PDEVICE_NET_INFO;
```

Visual Basic：

```
Private Type DEVICE_NET_INFO
    strIP (0 to 15) As Byte             ' IP 地址, "192.168.2.70"
    strSubnetMask (0 to 15) As Byte     ' 子网掩码, "255.255.255.255"
    strGateway (0 to 15) As Byte        ' 网关, "192.168.2.1"
    strMAC (0 to 19) As Byte            ' 网卡物理地址, "00-01-02-03-04-05",用户一般不可修改
    wTCPPort As Integer                 ' TCP 端口
End Type
```

Delphi:

```
Type   // 定义结构体数据类型
```

```
PDEVICE_NET_INFO = ^ DEVICE_NET_INFO;     // 指针类型结构
DEVICE_NET_INFO = record                  // 标记为记录型
    strIP : Array[0..15] of char;         // IP 地址, "192.168.2.70"
    strSubnetMask : Array[0.. 15] of char;  // 子网掩码, "255.255.255.255"
    strGateway : Array[0.. 15] of char;   // 网关, "192.168.2.1"
    strMAC : Array[0..19] of char;        // 网卡物理地址, "00-01-02-03-04-05",用户一般不可修改
    wTCPPort : Word;                      // TCP 端口号
End;
```

硬件参数说明:

strIP IP 地址， "192.168.2.70"。

strSubnetMask 子网掩码， "255.255.255.255"。

strGateway 网关， "192.168.2.1"。

strMAC 网卡物理地址， "00-01-02-03-04-05"，用户一般不可修改。

wTCPPort TCP 端口。

第五章　共用函数接口介绍

第一节、设备驱动接口函数列表（每个函数省略了前缀“NET2801_”）

函数名	函数功能	备注
① 创建 Visual Basic 子线程，线程数量可达 32 个以上		
CreateSystemEvent	创建系统内核事件对象	用于线程同步或中断
ReleaseSystemEvent	释放系统内核事件对象	
② 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
③ 各种参数保存和读取函数		
SaveParaInt	保存整型参数到注册表	
LoadParaInt	从注册表中读取整型参数值	
SaveParaString	保存字符参数到注册表	
LoadParaString	从注册表中读取字符参数值	

第二节、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）

♦ **创建内核系统事件**

Visual C++ & C++ Builder:

HANDLE CreateSystemEvent(void)

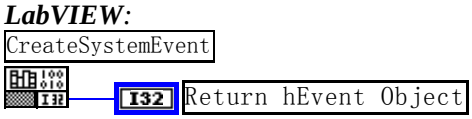
Visual Basic:

Declare Function CreateSystemEvent Lib " NET2801 " ( ) As Long

Delphi:

Function CreateSystemEvent( ) : Integer;
StdCall; External 'NET2801' 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**

Visual C++ & C++ Builder:

BOOL ReleaseSystemEvent(HANDLE hEvent)

Visual Basic:

Declare Function ReleaseSystemEvent Lib " NET2801 " (ByVal hEvent As Long) As Boolean

Delphi:

Function ReleaseSystemEvent(hEvent : Integer) : Boolean;
StdCall; External 'NET2801' 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。

第三节、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 strFileName,
int Mode)

Visual Basic:

Declare Function CreateFileObject Lib "NET2801" (ByVal hDevice As Long, _
ByVal strFileName As String, _
ByVal Mode As Integer) As Long

Delphi:

Function CreateFileObject (hDevice : Integer;
strFileName : string;
Mode : Integer) : Integer;
Stdcall; external 'NET2801' 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。

strFileName 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:\\NET2801\\Data.Dat”，在 Basic 中，其语法格式如：“C:\NET2801\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
NET2801_modeRead	0x0000	只读文件方式
NET2801_modeWrite	0x0001	只写文件方式
NET2801_modeReadWrite	0x0002	既读又写文件方式
NET2801_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

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,
               LONG nWriteSizeBytes)
```

Visual Basic:

```
Declare Function WriteFile Lib "NET2801" (ByVal hFileObject As Long,_
                                          ByRef pDataBuffer As Byte,_
                                          ByVal nWriteSizeBytes As Long) As Boolean
```

Delphi:

```
Function WriteFile(hFileObject: Integer;
                   pDataBuffer : Pointer;
                   nWriteSizeBytes : Integer) : Boolean;
                    Stdcall; external 'NET2801' 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 GetLastError 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,
               LONG OffsetBytes,
               LONG nReadSizeBytes)
```

Visual Basic:

```
Declare Function ReadFile Lib "NET2801" (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 : Integer;
                   nReadSizeBytes : Integer) : Boolean;
                   Stdcall; external 'NET2801' 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 GetLastError 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,
                    LONG nOffsetBytes)
```

Visual Basic:

```
Declare Function SetFileOffset Lib "NET2801" (ByVal hFileObject As Long,
                                              ByVal nOffsetBytes As Long) As Boolean
```

Delphi:

```
Function SetFileOffset ( hFileObject : Integer;
                         nOffsetBytes : LongInt) : Boolean;
                         Stdcall; external 'NET2801' 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 GetLastError 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 "NET2801" (ByVal hFileObject As Long) As Long
```

Delphi:

```
Function GetFileLength (hFileObject : Integer) : LongWord;
                        Stdcall; external 'NET2801' 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 GetLastError 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 "NET2801" (ByVal hFileObject As Long) As Boolean
```

Delphi:

```
Function ReleaseFile(hFileObject : Integer) : Boolean;
                     Stdcall; external 'NET2801' 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 GetLastError 捕获错误码。

相关函数： CreateFileObject WriteFile ReadFile ReleaseFile

- **取得指定磁盘的可用空间**

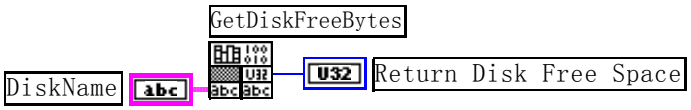
Visual C++ & C++ Builder:

LONGLONG GetDiskFreeBytes(LPCTSTR strDiskName)

Visual Basic:

Declare Function GetDiskFreeBytes Lib "NET2801" (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，用户可用 GetLastError 捕获错误码。注意使用 64 位整型变量。

第四节、各种参数保存和读取函数原型说明

♦ **将整型变量的参数值保存在系统注册表中**

函数原型：

Visual C++ & C++ Builder:

```
BOOL SaveParaInt( HANDLE hDevice,
                  LPCTSTR strParaName,
                  int nValue)
```

Visual Basic:

```
Declare Function SaveParaInt Lib "NET2801" (ByVal hDevice As Long,_
                                            ByVal strParaName As String,_
                                            ByVal nValue As Integer) As Boolean
```

Delphi:

```
Function SaveParaInt( hDevice : Integer;
                      strParaName : String;
                      nValue : Integer) : Boolean;
                      Stdcall; external 'NET2801' Name ' SaveParaInt ';
```

LabVIEW:

详见相关演示程序。

功能：将整型变量的参数值保存在系统注册表中。具体保存位置视设备逻辑号而定。如逻辑号为“0”的其他参数保存位置为：HKEY_CURRENT_USER\Software\Art\NET2801\Device-0\Others。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

strParaName 整型参数字符名。它指名该参数在注册表中的字符键项。

nValue 整型参数值。它保存在由 strParaName 命名的键项里。

返回值：若成功，则返回 TRUE，否则返回 FALSE，用户可以用 GetLastError 捕获错误码。

相关函数： SaveParaInt LoadParaInt SaveParaString LoadParaString

♦ **将整型变量的参数值从系统注册表中读出**

函数原型：

Visual C++ & C++ Builder:

```
UINT LoadParaInt( HANDLE hDevice,
                  LPCTSTR strParaName,
                  int nDefaultVal)
```

Visual Basic:

```
Declare Function LoadParaInt Lib "NET2801" (ByVal hDevice As Long,_
                                            ByVal strParaName As String,_
                                            ByVal nDefaultVal As Integer) As Long
```

Delphi:

```
Function LoadParaInt ( hDevice : Integer;
                       strParaName : String;
                       nDefaultVal: Integer) : Longword;
                       Stdcall; external 'NET2801' Name ' LoadParaInt ';
```

LabVIEW:

详见相关演示程序。

功能：将整型变量的参数值从系统注册表中读出。读出参数值的具体位置视设备逻辑号而定。如逻辑号为“0”的其他参数保存位置为：HKEY_CURRENT_USER\Software\Art\NET2801\Device-0\Others。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

strParaName 整型参数字符名。它指名该参数在注册表中的字符键项。

nDefaultVal 若 strParaName 指定的键项不存在，则由该参数指定的默认值返回。

返回值：若指定的整型参数项存在，则返回其整型值。否则返回由 nDefaultVal 指定的默认值。

相关函数： SaveParaInt LoadParaInt SaveParaString LoadParaString

♦将字符变量的参数值保存在系统注册表中

函数原型：

Visual C++ & C++ Builder:

```
BOOL SaveParaString ( HANDLE hDevice,
                      LPCTSTR strParaName,
                      LPCTSTR strParaVal)
```

Visual Basic:

```
Declare Function SaveParaString Lib "NET2801" (ByVal hDevice As Long,_
                                               ByVal strParaName As String,_
                                               ByVal strParaVal As String) As Boolean
```

Delphi:

```
Function SaveParaString (hDevice : Integer;
                         strParaName : String;
                         strParaVal: String) : Boolean;
                         Stdcall; external 'NET2801' Name ' SaveParaString';
```

LabVIEW:

详见相关演示程序。

功能：将整型变量的参数值保存在系统注册表中。具体保存位置视设备逻辑号而定。如逻辑号为“0”的其他参数保存位置为：HKEY_CURRENT_USER\Software\Art\NET2801\Device-0\Others。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。
strParaName 整型参数字符名。它指名该参数在注册表中的字符键项。
strParaVal 字符参数值。它保存在由 strParaName 命名的键项里。
返回值：若成功，则返回 TRUE，否则返回 FALSE，用户可以用 GetLastError 捕获错误码。
相关函数： SaveParaInt LoadParaInt SaveParaString LoadParaString

♦将字符变量的参数值从系统注册表中读出

函数原型：

Visual C++ & C++ Builder:

```
BOOL LoadParaString ( HANDLE hDevice,
                      LPCTSTR strParaName,
                      LPCTSTR strParaVal,
                      LPCTSTR strDefaultVal)
```

Visual Basic:

```
Declare Function LoadParaString Lib "NET2801" (ByVal hDevice As Long,_
                                               ByVal strParaName As String,_
                                               ByVal strParaVal As String,_
                                               ByVal strDefaultVal As String) As Boolean
```

Delphi:

```
Function LoadParaString ( hDevice : Integer;
                          strParaName : String;
                          strParaVal : String;
                          strDefaultVal : String) : Boolean;
                          Stdcall; external 'NET2801' Name ' LoadParaString ';
```

LabVIEW:

详见相关演示程序。

功能：将字符变量的参数值从系统注册表中读出。读出参数值的具体位置视设备逻辑号而定。如逻辑号为“0”的其他参数保存位置为：HKEY_CURRENT_USER\Software\Art\NET2801\Device-0\Others。
参数：
hDevice设备对象句柄，它应由CreateDevice创建。
strParaName 字符参数字符名。它指名该参数在注册表中的字符键项。
strParaVal 取得 strParaName 指定的键项的字符值。
strDefaultVal 若 strParaName 指定的键项不存在，则由该参数指定的默认值返回。
返回值：若成功，则返回 TRUE，否则返回 FALSE，用户可以用 GetLastError 捕获错误码。
相关函数： SaveParaInt LoadParaInt SaveParaString LoadParaString