USB2808 数据采集卡

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

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 USB2808Inst.doc 文档。

第一章 版权信息与命名约定

第一节、版权信息

本软件产品及相关套件均属北京市阿尔泰科贸有限公司所有，其产权受国家法律绝对保护，除非本公司书面允许，其他公司、单位及个人不得非法使用和拷贝，否则将受到国家法律的严厉制裁。您若需要我公司产品及相关信息请及时与我们联系，我们将热情接待。

第二节、命名约定

一、为简化文字内容，突出重点，本文中提到的函数名通常为基本功能名部分，其前缀设备名如 USBxxxx_ 则被省略。如 USB2808_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 USB 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。

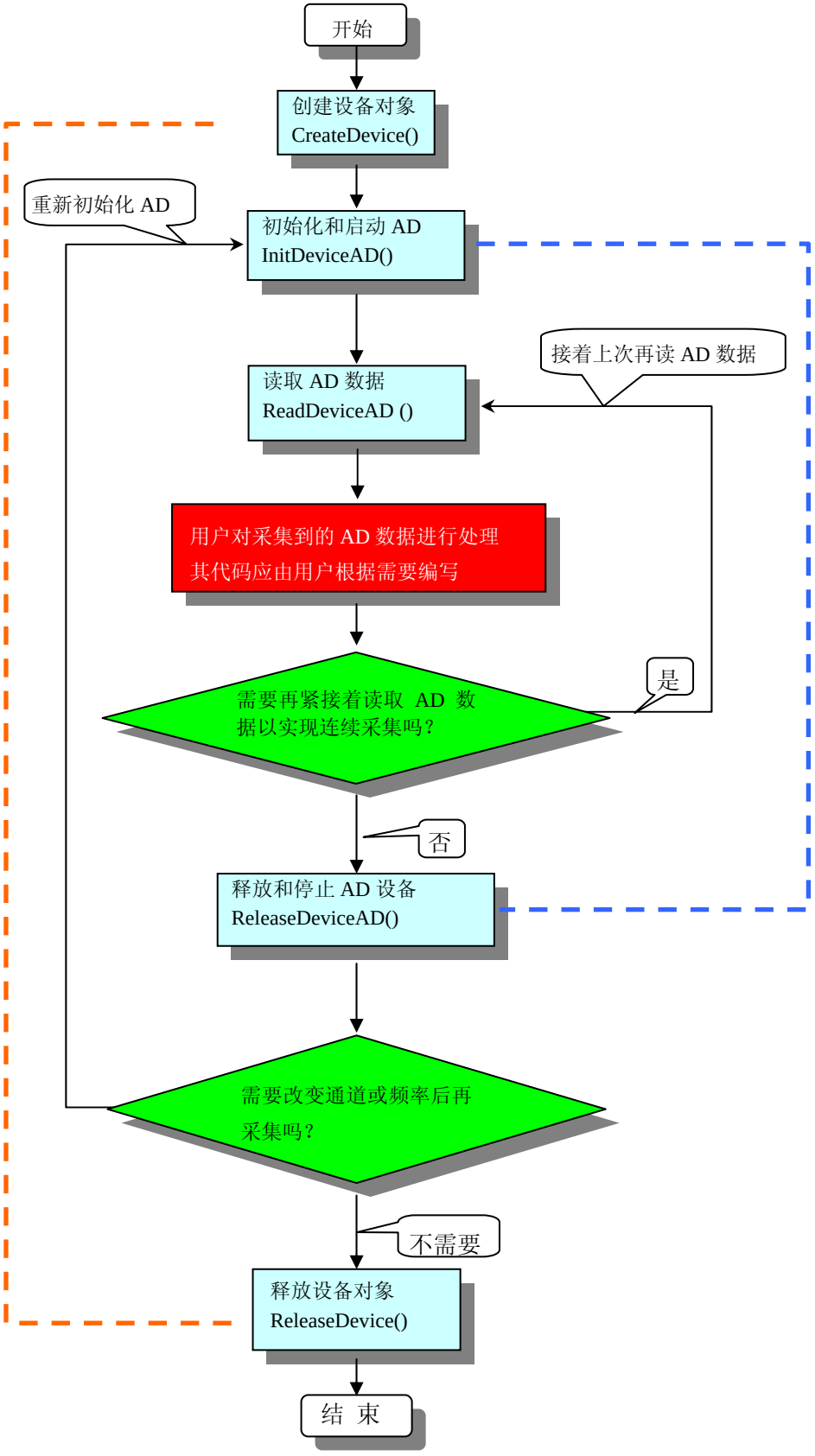


图 2.1.1 AD 采集实现过程

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 USB 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“USB2808_”）

本章函数是设备使用 USB 方式传输时所使用的。

函数名	函数功能	备注
① 设备对象操作函数		
CreateDevice	创建 USB 对象(用设备逻辑号)	
GetDeviceCount	取得设备总数	
GetDeviceCurrentID	取得设备当前 ID 号	
ListDeviceDlg	列表所有同一种 USB 各种配置	上层及底层用户
ResetDevice	复位 USB 设备	
ReleaseDevice	关闭设备，且释放 USB 总线设备对象	
② AD 采样操作函数		
InitDeviceAD	初始化 USB 设备 AD 部件，准备传数	
GetDeviceStatusAD	取得状态标志	
ReadDeviceAD	连续批量读取 USB 设备上的 AD 数据	
ReleaseDeviceAD	释放 USB 设备对象中的 AD 部件	
③ 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 AD 采样参数恢复至出厂默认值	

使用需知

Visual C++：

首先将 USB2808.h 和 USB2808.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(USB2808.lib)加入到您的工程中。

#include “USB2808.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 USB2808.h 和 USB2808.lib 两个文件均可在演示程序文件夹下面找到。

Visual Basic:

首先将 USB2808.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 USB2808.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。

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、设备对象管理函数原型说明

♦ 创建设备对象函数

函数原型：

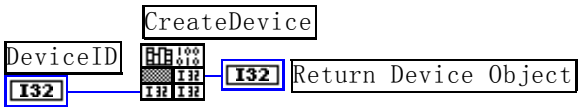
Visual C++：

HANDLE CreateDevice(int DeviceLgcID = 0)

Visual Basic：

Declare Function CreateDevice Lib "USB2808" (Optional ByVal DeviceLgcID As Integer = 0) As Long

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：

DeviceLgcID 设备逻辑 ID(Identifier)标识号。当向同一个 Windows 系统中加入若干相同类型的 USB 设备时，系统将以该设备的“基本名称”与 DeviceLgcID 标识值为名称后缀的标识符来确认和管理该设备。比如若用户往 Windows 系统中加入第一个 USB2808 AD 模板时，系统则以“USB2808”作为基本名称，再以 DeviceLgcID 的初值组合成该设备的标识符“USB2808-0”来确认和管理这第一个设备，若用户接着再添加第二个 USB2808 AD 模板时，则系统将以“USB2808-1”来确认和管理第二个设备，若再添加，则以此类推。所以当用户要创建设备句柄管理和操作第一个 USB 设备时，DeviceLgcID 应置 0，第二应置 1，也以此类推。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

♦ 取得在系统中的设备总台数

函数原型：

Visual C++:

int GetDeviceCount (HANDLE hDevice)

Visual Basic:

Declare Function GetDeviceCount Lib "USB2808" (ByVal hDevice As Long) As Integer

LabView:

请参考相关演示程序。

功能：取得在系统中物理设备的总台数。

参数：hDevice 设备对象句柄，它应由CreateDevice 创建。

返回值：若成功，则返回实际设备台数，否则返回 0，用户可以用 GetLastError 捕获错误码。

相关函数：CreateDevice ReleaseDevice

◆ **取得当前设备对象句柄指向的设备所在的设备 ID**

函数原型：

Visual C++:

```
BOOL GetDeviceCurrentID (HANDLE hDevice,
                         PLONG DeviceLgcID,
                         PLONG DevicePhysID)
```

Visual Basic:

```
Declare Function GetDeviceCurrentID Lib "USB2808" (ByVal hDevice As Long, _
                                                   ByRef DeviceLgcID As Long, _
                                                   ByRef DevicePhysID As Long ) As Boolean
```

LabView:

请参考相关演示程序。

功能：取得指定设备对象所代表的设备在设备链中的物理设备 ID 号和逻辑 ID 号。

参数：

hDevice 设备对象句柄，它应由CreateDevice 创建。

DevicePhysID 指针参数，取得指定设备的物理 ID。该号可以由用户设置板上 JP1 跳线器获得。当多卡工作时，该物理 ID 号能让用户准确的辩别每一个卡所处的编址。该编址除了用户能手动改变以外，不会随着系统加载卸载设备的顺序或者是用户插拔设备的顺序而改变其相应的物理编址，它只会改变其逻辑编址，即 DeviceLgcID 的值。比如您使用某一产品，该产品只有 32 个 AD 通道，而您需要 128 个通道采样，那么您就需要四块卡来实现此功能。在实际采样中，您需要对将 128 路信号依次分配到四个卡上，如下表：

卡号	信号通道	物理 ID 号	逻辑 ID 号
第一块	1～32	0	若第二个加载此卡，则为 1，否则为其他号
第二块	33～64	1	若最先加载此卡，则为 0，否则为其他号
第三块	65～96	2	若第四个加载此卡，则为 3，否则为其他号
第四块	97～128	3	若第三个加载此卡，则为 2，否则为其他号

从上表可知，如果使用您物理 ID 号，那么不管怎么安装您的硬件设备，那么其卡的物理编址不会发生任何变化，在软件上您用相应的物理 ID 号创建的设备对象所访问设备在物理上不会发生变化，它始终对应于您的事先分配的信号通道。而使用逻辑 ID 号则不然，同样的逻辑 ID 值可能在不同的拔插顺序下会指向不同的物理设备而使软件的通道序列与硬件上无法一一对应。

返回值：若成功，则返回由hDevice参数代表的设备在设备链中的设备ID，否则返回-1，用户可以用GetLastError捕获错误码。注意其返回的ID是一定与在CreateDevice函数中指定的DeviceLgcID参数值相等。

相关函数：CreateDevice ReleaseDevice

◆ **用对话框控件列表计算机系统中所有 USB2808 设备各种配置信息**

函数原型：

Visual C++:

```
BOOL ListDeviceDlg (void)
```

Visual Basic:

```
Declare Function ListDeviceDlg Lib "USB2808" ( )As Boolean
```

LabVIEW：

请参考相关演示程序。

功能：列表系统中 USB2808 的硬件配置信息。
参数：无。
返回值：若成功，则弹出对话框控件列表所有 USB2808 设备的配置情况。
相关函数：CreateDevice　　ReleaseDevice

◆ **复位整个 USB 设备**

函数原型：

Visual C++:

BOOL ResetDevice (HANDLE hDevice)

Visual Basic:

Declare Function ResetDevice Lib "USB2808" (ByVal hDevice As Long) As Boolean

LabView:

请参考相关演示程序。

功能：复位整个 USB 设备，相当于它与 PC 机端重新建立。其效果与重新插上 USB 设备等同。一般在出错情况下，想软复位来建决重连接问题，就可以调用该函数解决此问题。
参数：hDevice 设备对象句柄，它应由CreateDevice 创建。由它指向要复位的设备。
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

◆ **释放设备对象所占的系统资源及设备对象**

函数原型：

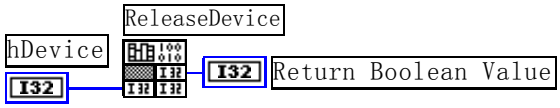
Visual C++:

BOOL ReleaseDevice(HANDLE hDevice)

Visual Basic:

Declare Function ReleaseDevice Lib "USB2808" (ByVal hDevice As Long) As Boolean

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。

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++：

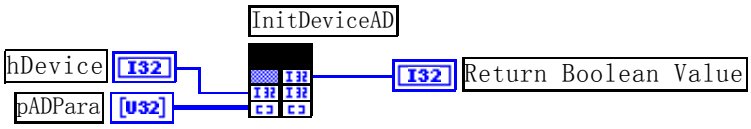
BOOL InitDeviceAD(HANDLE hDevice,

PUSB2808_PARA_AD pADPara)

Visual Basic:

Declare Function InitDeviceAD Lib "USB2808" (ByVal hDevice As Long, _
ByRef pADPara As USB2808_PARA _AD) As Boolean

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USB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USB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。

◆ **取得状态标志**

函数原型：

Visual C++:

BOOL GetDeviceStatusAD (HANDLE hDevice,
PBOOL bNotEmpty,
PBOOL bHalf,
PBOOL bOverflow)

Visual Basic:

Declare Function GetDeviceStatusAD Lib "USB2808" (ByVal hDevice As Long,_
ByRef bNotEmpty As Boolean,_
ByRef bHalf As Boolean,_
ByRef bOverflow As Boolean) As Boolean

LabVIEW:

请参考相关演示程序。

功能：一旦用户使用InitDeviceAD后，应立即用此函数查询状态（触发标志、转换标志、溢出标志）。在AD采集过程中取得FIFO的状态(可选，并不做为ReadDeviceAD的读操作同步)。

参数：

hDevice设备对象句柄，它应由CreateDevice 创建。

bNotEmpty 取得非空状态，TRUE 表示已非空有效，FALSE 表示非空无效。

bHalf 取得半满状态，TRUE 表示已半满有效，FALSE 表示半满以下。

bOverflow 取得溢出状态，TRUE 表示已溢出，FALSE 表示未溢出。

返回值：若调用成功则返回TRUE，否则返回FALSE，用户可以调用GetLastErrorEx函数取得当前错误码。

相关函数： CreateDevice InitDeviceAD ReadDeviceAD ReleaseDevice

♦ **批量读取 USB 设备上的 AD 数据**

函数原型：

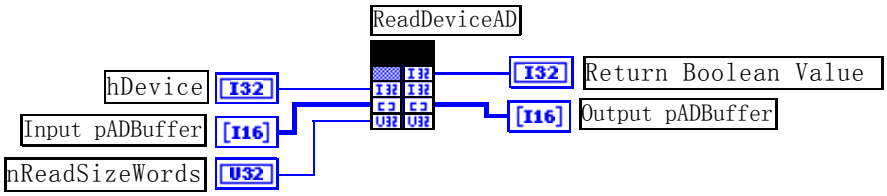
Visual C++:

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

Visual Basic:

```
Declare Function ReadDeviceAD Lib "USB2808" (ByVal hDevice As Long, _
                                             ByRef ADBuffer[]r As Integer, _
                                             ByRef nReadSizeWords As Integer, _
                                             ByVal nRetSizeWords = NULL As Long) As Boolean
```

LabView:



功能： 读取USB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。

pADBuffer 指向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，USB 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：

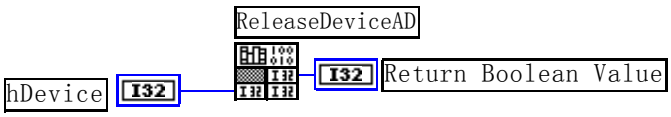
Visual C++:

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

Visual Basic:

```
Declare Function ReleaseDeviceAD Lib "USB2808" (ByVal hDevice As Long ) As Boolean
```

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 GetDeviceStatusAD
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，则执行到第步后⑤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、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：

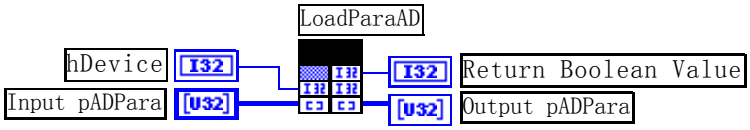
Visual C++:

```
BOOL LoadParaAD(HANDLE hDevice,
                PUSB2808_PARA_AD pADPara)
```

Visual Basic:

```
Declare Function LoadParaAD Lib "USB2808" (ByVal hDevice As Long, _
                                           ByRef pADPara As USB2808_PARA_AD) As Boolean
```

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 PUSB2808_PARA 的结构指针型，它负责返回 USB 硬件参数值，关于结构指针类型 PUSB2808_PARA 请参考相应 USB2808.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++举例：

```
        :
USB2808_PARA_AD ADPara;
HANDLE   hDevice;
HDevice = CreateDevice(0) ;     // 管理秕一个 USB 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 USB2808_PARA_AD
Dim hDevice As Long   :
 hDevice = CreateDevice(0)    ' 管理第一个 USB 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：

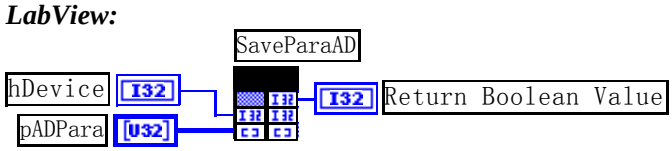
Viusal C++:

```
BOOL SaveParaAD(HANDLE hDevice,
                PUSB2808_PARA_AD pADPara)
```

Visual Basic:

```
Declare Function SaveParaAD Lib "USB2808" (ByVal hDevice As Long, _
                                           ByRef pADPara As USB2808_PARA_AD) As Boolean
```

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

♦ **将 AD 采样参数恢复至出厂默认值**

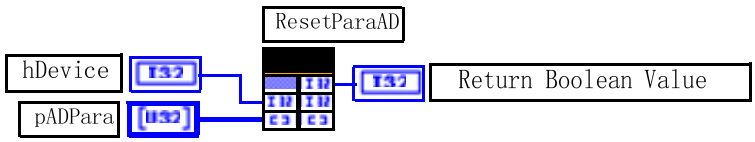
函数原型：

Viusal C++:

```
BOOL ResetParaAD (HANDLE hDevice,
                  PUSB2808_PARA_AD pADPara)
```

Visual Basic:

```
Declare Function ResetParaAD Lib "USB2808" (ByVal hDevice As Long, _
                                            ByRef pADPara As USB2808_PARA_AD) As Boolean
```



功能：将 AD 采样参数恢复至出厂默认值。

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

第四章 硬件参数结构

Visual C++:

```
typedef struct _USB2808_PARA_AD        // 板卡各参数值
{
    LONG CheckStsMode;          // 检查存储器状态(Status)的方式(非空还是半满)
    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 10Hz ～ 250KHz
    LONG Gains;                 // 采集增益
    LONG GroupInterval;         // 分组采样时的组间间隔(单位：微秒) [1, 16777215]
    LONG LoopsOfGroup;          // 组内循环次数[1-65535]
    LONG TriggerMode;           // 触发模式选择(内/外触发)
    LONG TriggerDir;            // 触发方向选择(正向、负向、正负向触发)
} USB2808_PARA_AD, *PUSB2808_PARA_AD;
```

Visual Basic：

```
Private Type USB2808_PARA_AD
    CheckStsMode As Long        ' 检查存储器状态(Status)的方式(非空还是半满)
    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 10Hz ～ 250KHz
    Gains As Long               ' 增益控制字
    GroupInterval As Long       ' 分组采样时，相邻组的时间间隔(uS)[1, 16777215]
    LoopsOfGroup As Long        ' 组内循环次数[1-65535]
    TriggerMode As Long         ' 触发模式选择(内/外触发)
    TriggerDir As Long          ' 触发方向选择(正向、负向、正负向触发方向)
End Type
```

LabView:

请参考相关演示程序。

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 USB 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 USB 设备的用户永远告别，一句话 USB 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。

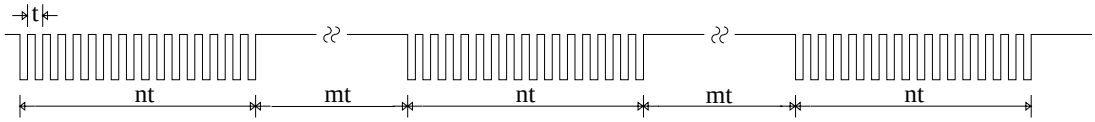
CheckStsMode 检查存储器状态(Status)的方式(非空还是半满)。

常量名	常量值	功能定义
USB2808_CHKSTSMODE_HALF	0x0000	查询 FIFO 半满标志
USB2808_CHKSTSMODE_NPT	0x0001	查询 FIFO 非空标志

ADMode AD 采样模式。它的取值如下表：

常量名	常量值	功能定义
USB2808_ADMODE_SEQUENCE	0x0000	连续采集模式
USB2808_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 10Hz ～ 250KHz。注意：

若连续采集(即ADMode = USB2808_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= USB2808_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。

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
USB2808_GAINS_1MULT	0x0000	1 倍增益(使用 AD8251 放大器)
USB2808_GAINS_2MULT	0x0001	2 倍增益(使用 AD8251 放大器)
USB2808_GAINS_4MULT	0x0002	4 倍增益(使用 AD8251 放大器)
USB2808_GAINS_8MULT	0x0003	8 倍增益(使用 AD8251 放大器)

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, 16777215]。

LoopsOfGroup 组内循环次数，取值范围为[1，65535]。

TriggerMode AD触发模式，若等于常量USB2808_TRIGMODE_SOFT则为内部软件触发，若等于常量USB2808_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
USB2808_TRIGMODE_SOFT	0x0000	软件内触发方式
USB2808_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
USB2808_TRIGDIR_NEGATIVE	0x0000	负向触发(低脉冲/下降沿触发)
USB2808_TRIGDIR_POSITIVE	0x0001	正向触发(高脉冲/上升沿触发)
USB2808_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

第五章 数据格式转换与排列规则

第一节、AD 原始数据 LSB 转换成电压值 Volt 的换算方法

在换算过程中弄清模板精度（即 Bit 位数）是很关键的，因为它决定了 LSB 数码的总宽度 CountLSB。比如 12 位的模板 CountLSB 为 4096。其他类型同理均按 2^n=LSB 总数（n 为 Bit 位数）换算即可。

量程(毫伏)	计算机语言换算公式(标准 C 语法)	Volt 取值范围 mV
±10000	Volt = (20000.00/65536) * (ADBuffer[0] &0xFFFF) – 10000.00	[-10000.00， + 9999.69]
±5000	Volt = (10000.00/65536) * (ADBuffer[0] &0xFFFF) – 5000.00	[-5000.00， + 4999.84]
±2500	Volt = (5000.00/65536)*(ADBuffer[0]&0xFFFF) –2500.00	[-2500.00，+2499.92]
0～10000	Volt = (10000.00/65536) * (ADBuffer[0] &0xFFFF)	[0.00， + 9999.84]
0～5000	Volt = (5000.00/65536)*(ADBuffer[0]&0xFFFF)	[0.00， + 4999.92]

换算举例：（设量程为±10000mV，这里只转换第一个点）

Visual C++：

```
USHORT Lsb; // 定义存放标准 LSB 原码的变量
float Volt;    // 定义存放转换后的电压值的变量
float PerLsbVolt = 20000.00/65536; // 求出每个 LSB 原码单位电压值
Lsb = (ADBuffer[0])&0xFFFF;
Volt = PerLsbVolt * Lsb – 10000.00;  // 用单位电压值与 LSB 原码数量相乘减去偏移求得实际电压值
```

Visual Basic:

```
Dim Lsb As Long   ' 定义存放标准 LSB 原码的变量
Dim Volt As Single   ' 定义存放转换后的电压值的变量
Dim PerLsbVolt As Single
PerLsbVolt = 20000.00/65536      ' 求出每个 LSB 原码单位电压值
Lsb = (ADBuffer(0) AND &HFFFF    ' 将其转换成无符号 13 位有效数据
```

Volt = PerLsbVolt * Lsb-10000.00 ' 用单位电压值与 LSB 原码数量相乘减去偏移求得实际电压值

第二节、AD 采集函数的 ADBuffer 缓冲区中的数据排放规则

当首末通道相等时，即为单通道采集，假如FirstChannel=5，LastChannel=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(CH0 – CH1)

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(CH0 ～ CH3)

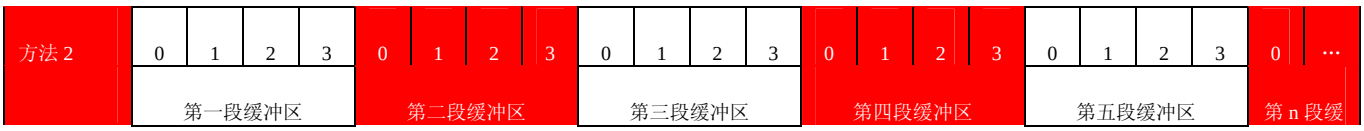
数据缓冲区索引号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	…
通道号	0	1	2	3	0	1	2	3	0	1	2	3	0	1	2	…

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问题，尤其任意通道数采集时。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。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（在USB设备上同时也应 32 的整数倍），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民，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，但对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。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			



第六章　上层用户函数接口应用实例

如果您想快速的了解驱动程序的使用方法和调用流程，以最短的时间建立自己的应用程序，那么我们强烈建议您参考相应的简易程序。此种程序属于工程级代码，可以直接打开不用作任何配置和代码修改即可编译通过，运行编译链接后的可执行程序，即可看到预期效果。

如果您想了解硬件的整体性能、精度、采样连续性等指标以及波形显示、数据存盘与分析、历史数据回放等功能，那么请参考高级演示程序。特别是许多不愿意编写任何程序代码的用户，您可以使用高级程序进行采集、显示、存盘等功能来满足您的要求。甚至可以用我们提供的专用转换程序将高级程序采集的存盘文件转换成相应格式，即可在 Excel、MatLab 第三方软件中分析数据（此类用户请最好选用通过 Visual C++制作的高级演示系统）。

第一节、简易程序演示说明

怎样使用ReadDeviceAD函数进行AD连续数据采集

其详细应用实例及工程级代码请参考 Visual C++简易演示系统及源程序，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 USB2808.h 和 Sys.cpp)。

[程序]→**[阿尔泰测控演示系统]**→ **[USB2808 AD 卡]**→ **[Microsoft Visual C++]**→ **[简易代码演示]**→ **[AD 采集演示源程序]**

其简易程序默认存放路径为：系统盘\ART\USB2808\SAMPLES\VC\SIMPLE\AD

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 USB2808.h 和 DADoc.cpp)。

[程序]→**[阿尔泰测控演示系统]**→ **[USB2808 AD 卡]**→ **[Microsoft Visual C++]**→ **[高级代码演示]**

其默认存放路径为：系统盘\ART\USB2808\SAMPLES\VC\ADVANCED

其他语言的演示可以用上面类似的方法找到。

第七章　基于 USB 总线的大容量连续数据采集详述

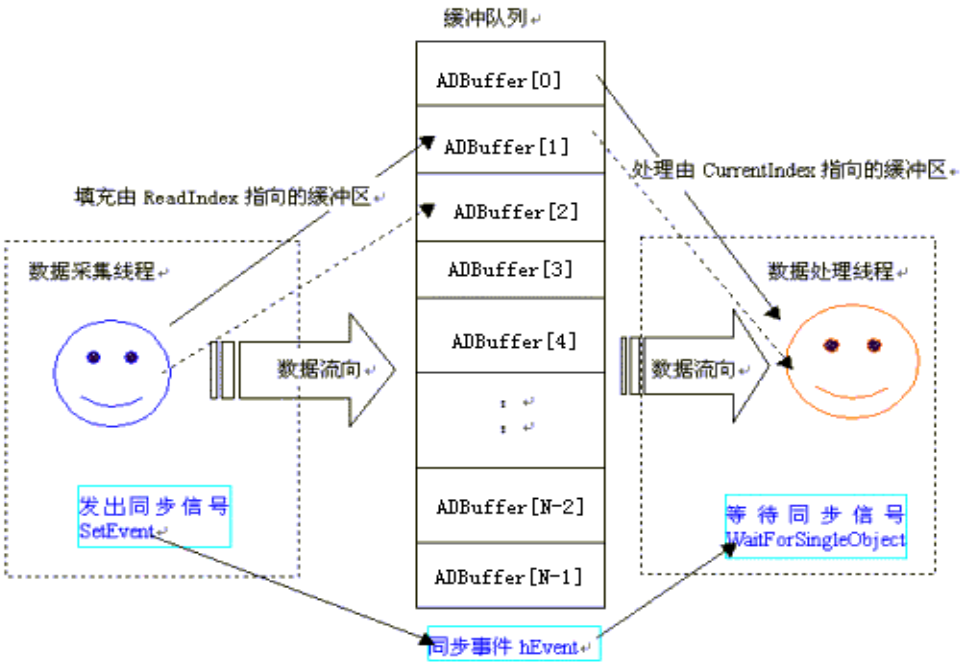
与 ISA、PCI 设备同理，使用子线程跟踪 AD 转换进度，并进行数据采集是保持数据连续不间断的最佳方案。但是与 ISA 总线设备不同的是，PCI 设备在这里不使用动态指针去同步 AD 转换进度，因为 ISA 设备环形内存池的动态指针操作是一种软件化的同步，而 PCI 设备不再有软件化的同步，而完全由硬件和驱动程序自动完成。这样一来，用户要用程序方式实现连续数据采集，其软件实现就显得极为容易。每次用 ReadDeviceAD 函数读取 AD 数据时，那么设备驱动程序会按照 AD 转换进度将 AD 数据一一放进用户数据缓冲区，当完成该次所指定的点数时，它便会返回，当您再次用这个函数读取数据时，它会接着上一次的位置传递数据到用户数据缓冲区。

但是由于我们的设备是通常工作在一个单 CPU 多任务的环境中，由于任务之间的调度切换非常平凡，特别是当用户移动窗口、或弹出对话框等，则会使当前线程猛地花掉大量的时间去处理这些图形操作，因此如果处理不当，则将无法实现高速连续不间断采集，那么如何更好的克服这些问题呢？用子线程则是必须的（在这里我们称之为数据采集线程），但这还不够，必须要求这个线程是绝对的工作者线程，即这个线程在正常采集

中不能有任何窗口等图形操作。只有这样，当用户进行任何窗口操作时，这个线程才不会被堵塞，因此可以保证其正常连续的数据采集。但是用户可能要问，不能进行任何窗口操作，那么我如何将采集的数据显示在屏幕上呢？其实很简单，再开辟一个子线程，我们称之数据处理线程，也叫用户界面线程。最初，数据处理线程不做任何工作，而是在 Win32 API 函数 WaitForSingleObject 的作用下进入睡眠状态，此时它基本不消耗 CPU 时间，即可保证其他线程代码有充分的运行机会（这里当然主要指数据采集线程），当数据采集线程取得指定长度的数据到用户空间时，则再用 Win32 API 函数 SetEvent 将指定事件消息发送给数据处理线程，则数据处理线程即刻恢复运行状态，迅速对这批数据进行处理，如计算、在窗口绘制波形、存盘等操作。

可能用户还要问，既然数据处理线程是非工作者线程，那么如果用户移动窗口等操作堵塞了该线程，而数据采集线程则在不停地采集数据，那数据处理线程难道不会因此而丢失采集线程发来的某一段数据吗？如果不另加处理，这个情况肯定有发生的可能。但是，我们采用了一级缓冲队列和二级缓冲队列的设计方案，足以避免这个问题。即假设数据采集线程每一次从设备上取出 8K数据，那么我们就创建一个缓冲队列，在用户程序中最简单的办法就是开辟一个两维数组如ADBuffer [SegmentCount][SegmentSize]， 我们将SegmentSize视为数据采集线程每次采集的数据长度，SegmentCount则为缓冲队列的成员个数。您应根据您的计算机物理内存大小和总体使用情况来设定这个数。假如我们设成 32，则这个缓冲队列实际上就是数组ADBuffer [32][8192]的形式。那么如何使用这个缓冲队列呢？方法很简单，它跟一个普通的缓冲区如一维数组差不多，唯一不同是，两个线程首先要通过改变SegmentCount字段的值，即这个下标Index的值来填充和引用由Index下标指向某一段SegmentSize长度的数据缓冲区。需要注意的是两个线程不共用一个Index下标变量。具体情况是当数据采集线程在AD部件被InitDeviceAD初始化之后，首次采集数据时，则将自己的ReadIndex下标置为 0，即用第一个缓冲区采集AD数据。当采集完后，则向数据处理线程发送消息，且两个线程的公共变量SegmentCount加 1，（注意SegmentCount变量是用于记录当前时刻缓冲队列中有多少个已被数据采集线程使用了，但是却没被数据处理线程处理掉的缓冲区数量。）然后再接着将ReadIndex偏移至 1，再用第二个缓冲区采集数据。再将SegmentCount加 1，直到ReadIndex等于 31 为止，然后再回到 0 位置，重新开始。而数据处理线程则在每次接受到消息时判断有多少由于自己被堵塞而没有被处理的缓冲区个数，然后逐一进行处理，最后再从SegmentCount变量中减去在所接受到的当前事件下所处理的缓冲区个数，具体处理哪个缓冲区由CurrentIndex指向。因此，即便应用程序突然很忙，使数据处理线程没有时间处理已到来的数据，但是由于缓冲区队列的缓冲作用，可以让数据采集线程先将数据连续缓存在这个区域中，由于这个缓冲区可以设计得比较大，因此可以缓冲很大的时间，这样即便是数据处理线程由于系统的偶而繁忙而被堵塞，也很难使数据丢失。而且通过这种方案，用户还可以在数据采集线程中对SegmentCount加以判断，观察其值是否大于了 32，如果大于，则缓冲区队列肯定因数据处理采集的过度繁忙而被溢出，如果溢出即可报警。因此具有强大的容错处理。

下图便形象的演示了缓冲队列处理的方法。可以看出，最初设备启动时，数据采集线程在往 ADBuffer[0]里面填充数据时，数据处理线程便在 WaitForSingleObject 的作用下睡眠等待有效数据。当 ADBuffer[0]被数据采集线程填充满后，立即给数据处理线程 SetEvent 发送通知 hEvent，便紧接着开始填充 ADBuffer[1]，数据处理线程接到事件后，便醒来开始处理数据 ADBuffer[0]缓冲。它们就这样始终差一个节拍。如虚线箭头所示。



下面只是简要的策略说明，其详细应用实例请参考 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**[USB2808 AD 卡]**丨**[Microsoft Visual C++ Sample ……]**

下面只是基于 C 语言的简要的策略说明，其详细应用实例及正确代码请参考 Visual C++测试与演示系统，您先点击 Windows 系统的[开始]菜单，再按下列顺序点击，即可打开基于 VC 的 Sys 工程(ADDoc.h 和 ADDoc.cpp)。

[程序]丨**[阿尔泰测控演示系统]**丨**[USB2808 AD 卡]**丨**[Microsoft Visual C++ Sample]**

然后，您着重参考 ADDoc.cpp 源文件中以下函数：

```
void CADDoc::StartDeviceAD()          // 启动线程函数
UINT ReadDataThread (PVOID hWnd)      // 读数据线程
UINT ProcessDataThread (PVOID hWnd)   // 绘制数据线程
void CADDoc::StopDeviceAD()           // 终止采集函数
```