USB2891 同步采集卡

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

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 USBxxxx_则被省略。如 USB2891_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。诸如InitDeviceAD、、ReadDeviceAD等。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，如InitDeviceAD可以使用 hDevice 句柄以程序查询方式初始化设备的 AD 部件，ReadDeviceAD 函数可以用 hDevice 句柄实现对 AD 数据的采样读取等。最后可以通过ReleaseDevice将 hDevice 释放掉。

第三节、哪些函数对您不是必须的

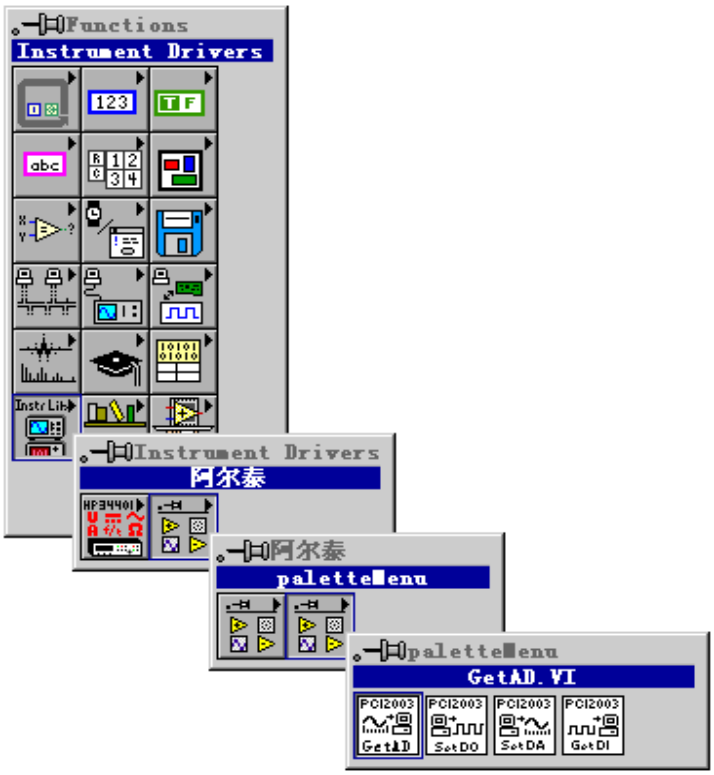
公共函数如CreateFileObject，WriteFile，ReadFile等一般来说都是辅助性函数，除非您要使用存盘功能。如果您使用上层用户函数访问设备，那么GetDeviceAddr，WriteRegisterByte，WriteRegisterWord，WriteRegisterULong，ReadRegisterByte，ReadRegisterWord，ReadRegisterULong等函数您可完全不必理会，除非您是作为底层用户管理设备。而WritePortByte，WritePortWord，WritePortULong，ReadPortByte， ReadPortWord，ReadPortULong则对 USB 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 USB 设备操作函数接口介绍

由于我公司的设备应用于各种不同的领域，有些用户可能根本不关心硬件设备的控制细节，只关心 AD 的首末通道、采样频率等，然后就能通过一两个简易的采集函数便能轻松得到所需要的 AD 数据。这方面的用户我们称之为上层用户。那么还有一部分用户不仅对硬件控制熟悉，而且由于应用对象的特殊要求，则要直接控制设备的每一个端口，这是一种复杂的工作，但又是必须的工作，我们则把这一群用户称之为底层用户。因此总的看来，上层用户要求简单、快捷，他们最希望在软件操作上所要面对的全是他们最关心的问题，比如在正式采集数据之前，只须用户调用一个简易的初始化函数（如InitDeviceAD）告诉设备我要使用多少个通道，采样频率是多少赫兹等，然后便可以用ReadDeviceProAD函数指定每次采集的点数，即可实现数据连续不间断采样。而关于设备的物理地址、端口分配及功能定义等复杂的硬件信息则与上层用户无任何关系。那么对于底层用户则不然。他们不仅要关心设备的物理地址，还要关心虚拟地址、端口寄存器的功能分配，甚至每个端口的 Bit 位都要了如指掌，看起来这是一项相当复杂、繁琐的工作。但是这些底层用户一旦使用我们提供的技术支持，则不仅可以让您不必熟悉 USB 总线复杂的控制协议，同是还可以省掉您许多繁琐的工作，比如您不用去了解 USB 的资源配置空间、PNP 即插即用管理，而只须用GetDeviceAddr函数便可以同时取得指定设备的物理基地址和虚拟线性基地址。这个时候您便可以用这个虚拟线性基地址，再根据硬件使用说明书中的各端口寄存器的功能说明，然后使用ReadRegisterULong和WriteRegisterULong对这些端口寄存器进行 32 位模式的读写操作，即可实现设备的所有控制。

综上所述，用户使用我公司提供的驱动程序软件包将极大的方便和满足您的各种需求。但为了您更省心，别忘了在您正式阅读下面的函数说明时，先明白自己是上层用户还是底层用户，因为在《设备驱动接口函数总列表》中的备注栏里明确注明了适用对象。

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



LabView 内嵌式驱动接口的获取方法

第一节、设备驱动接口函数总列表（每个函数省略了前缀“USB2891_”）

函数名	函数功能	备注
① 设备对象操作函数		
CreateDevice	创建 USB 设备对象(用设备逻辑号)	上层及底层用户
GetDeviceCurrentID	获取设备的 ID 号	上层及底层用户
ResetDevice	复位整个 USB 设备	上层及底层用户
ReleaseDevice	关闭设备，且释放 USB 总线设备对象	上层及底层用户
② AD 数据读取函数		
ADCalibration	AD 校准	上层用户
InitDeviceAD	初始化 AD 部件准备传输	上层用户
StartDeviceAD	启动 AD 设备，开始转换	上层用户
SetDeviceTrigAD	当设备使能允许后，产生软件触发事件（只有触发源为软件触发时有效）	上层用户
GetDevStatusAD	取得 AD 的状态	上层用户
ReadDeviceAD	读 AD 数据	上层用户
GetSDRAMSize	返回板载 DDR2 大小，单位为 Mb	上层用户
StopDeviceAD	暂停 AD 设备	上层用户
ReleaseDeviceAD	释放设备上的 AD 部件	上层用户
③ AD 硬件参数系统保存、读取函数		
LoadParaAD	从 Windows 系统中读入硬件参数	上层用户
SaveParaAD	往 Windows 系统写入设备硬件参数	上层用户
ResetParaAD	将注册表中的 AD 参数恢复至出厂默认值	上层用户

使用需知：

Visual C++：

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\USB2891\INCLUDE\USB2891.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 USB2891.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。

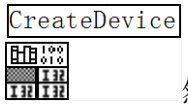
Visual Basic:

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 USB2891.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。

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 USB2891.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：

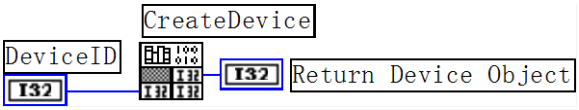
Visual C++：

HANDLE CreateDevice (int DeviceLgcID = 0)

Visual Basic：

Declare Function CreateDevice Lib "USB2891" (Optional ByVal DeviceID As Integer = 0) As Long

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： CreateDevice GetDeviceCurrentID
ResetDevice ReleaseDevice

Visual C++程序举例

```
    :
HANDLE hDevice;     // 定义设备对象句柄
int DeviceLgcID = 0;
hDevice = USB2891_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 = USB2891_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
    :
```

♦ **取得该设备当前相应的 ID 号**

函数原型：

Visual C++:

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

Visual Basic:

```
Declare Function GetDeviceCurrentID Lib "USB2891" (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：取得指定设备相应 ID 号。

参数：

hDevice 设备对象句柄，它指向要取得逻辑号的设备，它应由CreateDevice创建。

DeviceLgcID 返回设备的逻辑 ID，它的取值范围为[0, 15]。

DevicePhysID 返回设备的物理 ID 号。

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

相关函数： CreateDevice　　GetDeviceCurrentID
ResetDevice　　ReleaseDevice

♦**复位整个 USB 设备**

函数原型：

Visual C++:

```
BOOL ResetDevice (HANDLE hDevice)
```

Visual Basic:

```
Declare Function ResetDevice Lib "USB2891" (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。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

返回值：若成功，则弹出对话框控件列表所有 USB2891 设备的配置情况。

相关函数： CreateDevice　　GetDeviceCurrentID
ResetDevice　　ReleaseDevice

♦ **释放设备对象所占的系统资源及设备对象**

函数原型：

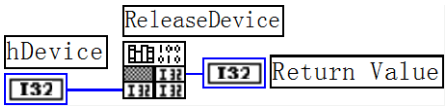
Visual C++:

BOOL ReleaseDevice (HANDLE hDevice)

Visual Basic:

Declare Function ReleaseDevice Lib "USB2891" (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GetLastErrorEx捕获错误码。

相关函数： CreateDevice GetDeviceCurrentID ResetDevice ReleaseDevice

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

第三节、AD 采样操作函数原型说明

♦ **AD 校准**

函数原型：

Visual C++：

BOOL ADCalibration (HANDLE hDevice)

Visual Basic:

Declare Function ADCalibration Lib "USB2891" (ByVal hDevice As Long) As Boolean

LabVIEW:

请参考相关演示程序。

功能：AD 功能的校准。

参数：

hDevice 设备对象句柄，它应由设备的CreateDevice创建。

InputRange 校准量程选择。

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

相关函数： CreateDevice InitDeviceAD StartDeviceAD SetDeviceTrigAD GetDevStatusAD ReadDeviceAD StopDeviceAD ReleaseDeviceAD ReleaseDevice

♦ **初始化设备对象**

函数原型：

Visual C++：

```
BOOL InitDeviceAD (HANDLE hDevice,
                   PUSB2891_PARA_AD pADPara)
```

Visual Basic:

```
Declare Function InitDeviceAD Lib "USB2891" (ByVal hDevice As Long, _
                                             ByRef pADPara As USB2891_PARA _AD) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：它负责初始化设备对象中的 AD 部件，为设备操作就绪有关工作，如预置 AD 采集通道、采样频率等。但它并不启动 AD 设备，若要启动 AD 设备，须在调用此函数之后再调用StartDeviceProAD。

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。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GetLastErrorEx捕获当前错误码，并加以分析。

相关函数: CreateDevice InitDeviceAD StartDeviceAD SetDeviceTrigAD GetDevStatusAD ReadDeviceAD StopDeviceAD ReleaseDeviceAD ReleaseDevice

♦ **启动 AD 设备(Start device AD for program mode)**

函数原型：

Visual C++:

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

Visual Basic:

```
Declare Function StartDeviceAD Lib "USB2891" (ByVal hDevice As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：启动 AD 设备，它必须在调用InitDeviceProAD后才能调用此函数。该函数除了启动 AD 设备开始转换以外，不改变设备的其他任何状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

返回值：如果调用成功，则返回 TRUE，且 AD 立刻开始转换，否则返回 FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数: CreateDevice InitDeviceAD StartDeviceAD SetDeviceTrigAD GetDevStatusAD ReadDeviceAD StopDeviceAD ReleaseDeviceAD ReleaseDevice

♦ **产生软件触发事件**

函数原型：

Visual C++:

BOOL SetDeviceTrigAD (HANDLE hDevice)

Visual Basic:

Declare Function SetDeviceTrigAD Lib "USB2891" (ByVal hDevice As Long) As Boolean

LabVIEW:

请参考相关演示程序。

功能：当设备使能允许后，产生软件触发事件（只有触发源为软件触发时有效）。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数： CreateDevice InitDeviceAD StartDeviceAD SetDeviceTrigAD GetDevStatusAD ReadDeviceAD StopDeviceAD ReleaseDeviceAD ReleaseDevice

♦ **取得 AD 状态标志**

函数原型：

Visual C++:

```
BOOL GetDevStatusAD (HANDLE hDevice,
                     PUSB2891_STATUS_AD pADStatus);
```

Visual Basic:

```
Declare Function GetDevStatusAD Lib "USB2891" (ByVal hDevice As Long, _
                                               ByRef pADStatus As USB2891_STATUS_AD) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：一旦用户使用 StartDeviceAD 后，应立即用此函数查询存储器的状态。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pADStatus 获得 AD 的各种当前状态。它属于结构体，具体定义请参考《AD 状态参数结构（USB2891_STATUS_AD）》章节。

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 StartDeviceAD SetDeviceTrigAD GetDevStatusAD ReadDeviceAD StopDeviceAD ReleaseDeviceAD ReleaseDevice

♦ **返回板载 DDR2 大小，单位为 Mb**

函数原型：

Visual C++:

ULONG GetSDRAMSize (HANDLE hDevice)

Visual Basic:

Declare Function ReadDeviceDmaAD Lib "USB2891" (ByVal hDevice As Long) As Boolean

LabVIEW：

请参考相关演示程序。

功能：返回板载 DDR2 大小，单位为 Mb。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

返回值：若调用成功则返回 TRUE，否则返回 FALSE。

相关函数: CreateDevice InitDeviceAD StartDeviceAD SetDeviceTrigAD GetDevStatusAD ReadDeviceAD StopDeviceAD ReleaseDeviceAD ReleaseDevice

- **读取 PXI 设备上的 AD 数据**

函数原型：

Visual C++:

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

Visual Basic:

```
BOOL DEVAPI ReadDeviceAD Lib "USB2891" (
                   ByVal hDevice As Long, _
                   ByRef ADBuffer As Integer, _
                   ByVal nReadSizeWords As Long, _
                   ByVal ulStartAddr As Long, _
                   ByRef nRetSizeWords As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：用此函数读取设备上的 AD 数据。

参数：

hDevice 设备对象句柄，它应由 CreateDevice 创建。

ADBuffer[] 将用于接受数据的用户缓冲区。

nReadSizeWords 读取 AD 数据的长度(字)。

ulStartAddr 数据在 RAM 中的起始地址

nRetSizeWords = NULL 实际返回数据的长度(字)。

返回值：

相关函数: CreateDevice InitDeviceAD StartDeviceAD SetDeviceTrigAD GetDevStatusAD ReadDeviceAD StopDeviceAD ReleaseDeviceAD ReleaseDevice

- **暂停 AD 设备**

函数原型：

Visual C++:

```
BOOL StopDeviceAD (HANDLE hDevice)
```

Visual Basic:

```
Declare Function StopDeviceAD Lib "USB2891" (ByVal hDevice As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：暂停 AD 设备。它必须在调用StartDeviceAD后才能调用此函数。该函数除了停止 AD 设备不再转换以外，不改变设备的其他任何状态。此后您可再调用StartDeviceAD函数重新启动 AD，此时 AD 会按照暂停以前的状态开始转换。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

返回值：如果调用成功，则返回 TRUE，且 AD 立刻停止转换，否则返回 FALSE，用户可用GetLastErrorEx捕获当前错误码，并加以分析。

相关函数: CreateDevice InitDeviceAD StartDeviceAD
SetDeviceTrigAD GetDevStatusAD ReadDeviceAD
StopDeviceAD ReleaseDeviceAD ReleaseDevice

♦ **释放设备上的 AD 部件**

函数原型：

Visual C++:

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

Visual Basic:

```
Declare Function ReleaseDeviceAD Lib "USB2891" (ByVal hDevice As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：释放设备上的 AD 部件。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

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

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

相关函数: CreateDevice InitDeviceAD StartDeviceAD
SetDeviceTrigAD GetDevStatusAD ReadDeviceAD
StopDeviceAD ReleaseDeviceAD ReleaseDevice

第四节、AD 硬件参数保存与读取函数原型说明

♦ **从 Windows 系统中读入硬件参数函数**

函数原型：

Visual C++:

```
BOOL LoadParaAD (HANDLE hDevice,
                 PUSB2891_PARA_AD pADPara)
```

Visual Basic:

```
Declare Function LoadParaAD Lib "USB2891" (ByVal hDevice As Long, _
                                           ByRef pADPara As USB2891_PARA_AD) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：负责从 Windows 系统中读取设备的硬件参数。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pADPara 属于 USB2891_PARA_AD 的结构指针类型，它负责返回 USB 硬件参数值，关于结构指针类型 USB2891_PARA_AD 请参考 USB2891.h 或 USB2891.Bas 或 USB2891.Pas 函数原型定义文件，也可参考本文《硬件参数结构》关于该结构的有关说明。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数: CreateDevice LoadParaAD SaveParaAD
ReleaseDevice

♦ **往 Windows 系统写入设备硬件参数函数**

函数原型：

Viusal C++:

```
BOOL SaveParaAD (HANDLE hDevice,
                 PUSB2891_PARA_AD pADPara)
```

Visual Basic:

```
Declare Function SaveParaAD Lib "USB2891" (ByVal hDevice As Long, _
                                           ByRef pADPara As USB2891_PARA_AD) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：负责把用户设置的硬件参数保存在 Windows 系统中，以供下次使用。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pADPara 设备硬件参数，关于 USB2891_PARA_AD 的详细介绍请参考 USB2891.h 或 USB2891.Bas 或 USB2891.Pas 函数原型定义文件，也可参考本文《硬件参数结构》关于该结构的有关说明。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice　LoadParaAD　SaveParaAD　ReleaseDevice

♦ **AD 采样参数复位至出厂默认值函数**

函数原型：

Viusal C++:

```
BOOL ResetParaAD (HANDLE hDevice,
                  PUSB2891_PARA_AD pADPara)
```

Visual Basic:

```
Declare Function ResetParaAD Lib "USB2891" (ByVal hDevice As Long, _
                                            ByRef pADPara As USB2891_PARA_AD) As Boolean
```

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 USB2891_PARA_AD 的详细介绍请参考 USB2891.h 或 USB2891.Bas 或 USB2891.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

第四章 硬件参数结构

第一节、AD 硬件参数介绍（USB2891_PARA_AD）

Visual C++:

```
 typedef struct _USB2891_PARA_AD
{
    LONG Frequency;              // 采集频率,单位为 Hz, [1, 1000000]
    LONG InputRange[16]; // 模拟量输入量程选择
    LONG Gains[16];              // 增益控制
    LONG M_Length;               // M 段长度(字), 总的取值范围 1-8M, 但是 M 加 N 长度不能大于 8M
    LONG N_Length;               // N 段长度(字), 总的取值范围 1-8M, 但是 M 加 N 长度不能大于 8M
    LONG TriggerMode;            // 触发模式选择
    LONG TriggerSource;          // 触发源选择
    LONG TriggerDir;        // 触发方向选择(正向/负向触发)
    LONG TrigLevelVolt;          // 触发电平(量程按模拟输入量程)
    LONG ClockSource;            // 时钟源选择(内/外时钟源)
    LONG ClockOutputEnable;  // 时钟输出使能(0:禁止时钟输出 1:允许)
} USB2891_PARA_AD, *PUSB2891_PARA_AD;
```

Visual Basic：

```
    Private Type USB2891_PARA_AD
        Frequency As Long            ' 采集频率,单位为 Hz, [1, 1000000]
        InputRange[16]As Long        ' 模拟量输入量程选择
        Gains[16]As Long             ' 增益控制
        M_Length As Long             'M 段长度(字), 总的取值范围 1-8M, 但是 M 加 N 长度不能大于 8M
        N_Length As Long             'N 段长度(字), 总的取值范围 1-8M, 但是 M 加 N 长度不能大于 8M
        TriggerMode As Long          ' 触发模式选择
        TriggerSource As Long        ' 触发源选择
        TriggerDir As Long           ' 触发方向选择(正向/负向触发)
        TrigLevelVolt As Long        ' 触发电平(量程按模拟输入量程)
        ClockSource As Long          ' 时钟源选择(内/外时钟源)
    End Type
```

LabVIEW:

请参考相关演示程序。

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

Frequency AD 采样频率，本设备的频率取值范围为[10,800KHz]。

Gain AD 采样程控增益。

常量名	常量值	功能定义
USB2891_GAINS_1MULT	0x00	1 倍增益
USB2891_GAINS_2MULT	0x01	2 倍增益
USB2891_GAINS_4MULT	0x02	4 倍增益
USB2891_GAINS_8MULT	0x03	8 倍增益

InputRange 模拟量输入量程选择。

常量名	常量值	功能定义
USB2891_INPUT_N10000_P10000mV	0x00	±10000mV
USB2891_INPUT_N5000_P5000mV	0x01	±5000mV
USB2891_INPUT_N2500_P2500mV	0x02	±2500mV
USB2891_INPUT_0_P10000mV	0x03	0～10000mV
USB2891_INPUT_0_P5000mV	0x04	0～5000mV

TriggerMode　AD 触发模式。

常量名	常量值	功能定义
USB2891_TRIGMODE_MIDL	0x00	中间触发
USB2891_TRIGMODE_POST	0x01	后触发
USB2891_TRIGMODE_PRE	0x02	预触发
USB2891_TRIGMODE_DELAY	0x03	硬件延时触发

TriggerSource　触发源选择。

常量名	常量值	功能定义
USB2891_TRIGMODE_SOFT	0x00	软件触发
USB2891_TRIGSRC_DTR	0x01	选择 DTR 作为触发源
USB2891_TRIGSRC_ATR	0x02	选择 ATR 作为触发源
USB2891_TRIGSRC_TRIGGER	0x03	Trigger 信号触发

TrigLevelVolt 触发电平（-10000mV～10000mV）。

TriggerDir　AD 触发方向。它的选项值如下表：

常量名	常量值	功能定义
USB2891_TRIGDIR_NEGATIVE	0x00	负向触发（下降沿触发）
USB2891_TRIGDIR_POSITIVE	0x01	正向触发（上升沿触发）
USB2891_TRIGDIR_POSIT_NEGAT	0x02	正负方向均有效

注明：USB2891_TRIGDIR_POSIT_NEGAT 在边沿类型下，则都表示不管是上边沿还是下边沿均触发。而在电平类型下，无论正电平还是负电平均触发。

ClockSource　AD 时钟源选择。它的选项值如下表：

常量名	常量值	功能定义
USB2891_CLOCKSRC_IN	0x00	内部时钟定时触发
USB2891_CLOCKSRC_OUT	0x01	外部时钟定时触发

ClockOutputEnable　AD 时钟输出使能选择。它的选项值如下表：

常量名	常量值	功能定义
USB2891_DIS_OUTPUT	0x00	禁止时钟输出
USB2891_EN_OUTPUT	0x01	允许时钟输出

相关函数： CreateDevice　LoadParaAD　SaveParaAD　ReleaseDevice

第二节、AD 状态参数结构（USB2891_STATUS_AD）

Visual C++:

```
typedef struct _USB2891_STATUS_AD
{
    LONG bADEanble;// AD 是否已经使能，=TRUE:表示已使能，=FALSE:表示未使能
    LONG bTrigger;   // AD 是否被触发，=TRUE:表示已被触发，=FALSE:表示未被触发
    LONG bComplete; // AD 是否整个转换过程是否结束，=TRUE:表示已结束，=FALSE:表示未结束
    LONG bAheadTrig;// AD 触发点是否提前，=TRUE:表示触发点提前，=FALSE:表示触发点未提前
    LONG lEndAddr;   //  数据完成的结束地址
} USB2891_STATUS_AD, *PUSB2891_STATUS_AD;
```

Visual Basic:

```
Private Type USB2891_STATUS_AD
    bADEanble As Long
    bTrigger As Long
    bComplete As Long
    bAheadTrig As Long
    lEndAddr As Long
End Type
```

LabVIEW:

请参考相关演示程序。

此结构体主要用于查询 AD 的各种状态，GetDevStatusAD函数使用此结构体来实时取得 AD 状态，以便同步各种数据采集和处理过程。

bADEanble AD 是否已经使能，=TRUE：表示已使能，=FALSE：表示未使能。

bTrigger AD 是否被触发，=TRUE：表示已被触发，=FALSE：表示未被触发。

bComplete AD 是否整个转换过程是否结束，=TRUE：表示已结束，=FALSE：表示未结束。

bAheadTrigAD 触发点是否提前，=TRUE：表示触发点提前，=FALSE：表示触发点未提前。

lEndAddr AD 数据完成的结束地址。

相关函数： CreateDevice GetDevStatusAD ReleaseDevice

第五章 数据格式转换与排列规则

第一节、AD 原码 LSB 数据转换成电压值的换算方法

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

量程(mV)	计算机语言换算公式(ANSI C 语法)	Volt 取值范围（mV）
±10000mV	Volt = (20000.00/65536) * (ADBuffer[0]&0xFFF) – 10000.00	[-10000, +9997.55]
±5000mV	Volt = (10000.00/65536)* (ADBuffer[0]&0xFFF) – 5000.00	[-5000, +4998.77]
±2500mV	Volt = (5000.00/65536) * (ADBuffer[0]&0xFFF) – 2500.00	[-2500, +2499.38]
0～10000mV	Volt = (10000.00/65536) * (ADBuffer[0]&0xFFF)	[0, +9998.77]
0～5000mV	Volt = (10000.00/65536) * (ADBuffer[0]&0xFFF)	[0, +4999.78]

下面举例说明各种语言的换算过程（以±10000mV 量程为例）

Visual C++：

```
Lsb = ADBuffer[0]&0xFFF;
Volt = (20000.00/65536) * Lsb -10000.00;
```

Visual Basic：

```
Lsb = ADBuffer [0] And &HFFF
Volt = (20000.00/65536) * Lsb – 10000.00
```

LabVIEW:

请参考相关演示程序。

第二节、AD 采集函数的 ADBuffer 缓冲区中的数据排放规则

数据缓冲区索引号	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	....
通道号	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	....

第三节、AD 测试应用程序创建并形成的数据文件格式

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

```
typedef struct _FILE_HEADER
{
    LONG HeadSizeBytes;                 // 文件头信息长度
    LONG FileType;                      // 该设备数据文件共有的成员
    LONG BusType;                       // 设备总线类型(DEFAULT_BUS_TYPE)
    LONG DeviceNum;                     // 该设备的编号(DEFAULT_DEVICE_NUM)
    LONG HeadVersion;                   // 头信息版本(D15-D8=Major,D7-D0=Minijor) = 1.0
    LONG VoltBottomRange;               // 量程下限(mV)
    LONG VoltTopRange;                  // 量程上限(mV)
    LONG StaticOverFlow;                // 同批文件识别码
    USB2891_PARA_AD ADPara;             // 保存硬件参数
    LONG HeadEndFlag;                   // 头信息结束位
} FILE_HEADER，*PFILE_HEADER;
```

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

第六章 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“USB2891_”）

函数名	函数功能	备注
① USB 总线内存映射寄存器操作函数		
GetDevVersion	获取设备固件及程序版本	底层用户
② 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
ReleaseSystemEvent	释放系统内核事件对象	

第二节、USB 内存映射寄存器操作函数原型说明

◆ **获取设备固件及程序版本**

函数原型：

Visual C++:

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

Visual Basic:

```
Declare Function GetDevVersion Lib "USB2891" (ByVal hDevice As Long, _
                                              ByRef pulFmwVersion As Long, _
                                              ByRef pulDriverVersion As Long) As Boolean
```

LabVIEW:

请参见相关演示程序。

功能：获取设备固件及程序版本。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

pulFmwVersion 指针参数，用于取得固件版本。

pulDriverVersion 指针参数，用于取得驱动版本。

返回值：如果执行成功，则返回 TRUE，否则会返回 FALSE。

相关函数: CreateDevice GetDevVersion ReleaseDevice

第三节、线程操作函数原型说明

◆ **创建内核系统事件**

函数原型：

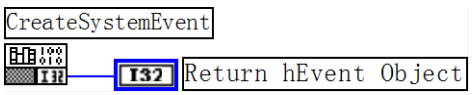
Visual C++:

```
HANDLE CreateSystemEvent (void)
```

Visual Basic:

```
Declare Function CreateSystemEvent Lib "USB2891" () As Long
```

LabVIEW:



功能：创建系统内核事件对象，它将被用于中断事件响应或数据采集线程同步事件。

参数：无任何参数。
返回值：若成功，返回系统内核事件对象句柄，否则返回－1(或 INVALID_HANDLE_VALUE)。
相关函数: CreateDevice CreateSystemEvent
ReleaseDevice ReleaseSystemEvent

♦ **释放内核系统事件**

函数原型：
Visual C++:
BOOL ReleaseSystemEvent (HANDLE hEvent)
Visual Basic:
Declare Function ReleaseSystemEvent Lib "USB2891" (ByVal hEvent As Long) As Boolean
LabVIEW:
请参见相关演示程序。
功能：释放系统内核事件对象。
参数：
hEvent 被释放的内核事件对象。它应由CreateSystemEvent成功创建的对象。
返回值：若成功，则返回 TRUE。
相关函数: CreateDevice CreateSystemEvent
ReleaseDevice ReleaseSystemEvent