

USB288X 数据采集卡

驱动程序使用手册

北京阿尔泰科技发展有限公司

产品研发部修订

2016

V6.00.01

■ 关于本手册

本手册为阿尔泰科技推出的 USB288X 数据采集卡驱动程序使用手册,其中包括版权信息与命名约定、使用纲要、各功能操作流程介绍、USB 设备操作函数接口介绍、上层用户函数接口应用实例、共用函数介绍、修改历史等。

文档版本: V6.00.01

USB288X: USB2881 ~ USB2886

目录

■ 关于本手册	1
■ 1 版权信息与命名约定	3
1.1 版权信息	3
1.2 命名约定	3
■ 2 使用纲要	4
2.1 使用上层用户函数，高效、简单	4
2.2 如何实现 AD 采集与开关量的简便操作	4
■ 3 各功能操作流程介绍	5
3.1 AD 模拟量输入操作流程	5
3.2 开关量输入输出操作流程	6
■ 4 USB 设备操作函数接口介绍	7
4.1 设备驱动接口函数总列表	8
4.2 设备对象管理函数原型说明	9
4.3 AD 数据读取函数原型说明	11
4.4 AD 硬件参数操作函数原型说明	13
4.5 数字 I/O 输入输出函数原型说明	14
■ 5 上层用户函数接口应用实例	16
5.1 简易程序演示说明	16
5.1.1 怎样进行开关量输入输出操作	16
5.1.2 怎样进行 AD 数据采集操作	16
5.2 高级程序演示说明	16
■ 6 共用函数介绍	17
6.1 公用接口函数总列表	17
6.2 操作函数原型说明	17
■ 7 修改历史	19

1 版权信息与命名约定

1.1 版权信息

本软件产品及相关套件均属北京阿尔泰科技发展有限公司所有，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、我公司授权的代理商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当地代理商联系或直接与我们联系，我们将热情接待。

1.2 命名约定

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 USB288X_CreateDevice 则写为 CreateDevice。

表 1-2-1: 函数名及参数中各种关键字缩写

缩写	全称	汉语意思	缩写	全称	汉语意思
Dev	Device	设备	DI	Digital Input	数字量输入
Pro	Program	程序	DO	Digital Output	数字量输出
Int	Interrupt	中断	CNT	Counter	计数器
Dma	Direct Memory Access	直接内存存取	DA	Digital convert to Analog	数模转换
AD	Analog convert to Digital	模数转换	DI	Differential	(双端或差分) 注: 在常量选项中
Npt	Not Empty	非空	SE	Single end	单端
Para	Parameter	参数	DIR	Direction	方向
SRC	Source	源	ATR	Analog Trigger	模拟量触发
TRIG	Trigger	触发	DTR	Digital Trigger	数字量触发
CLK	Clock	时钟	Cur	Current	当前的
GND	Ground	地	OPT	Operate	操作
Lgc	Logical	逻辑的	ID	Identifier	标识
Phys	Physical	物理的			



以上规则不局限于该产品。

2 使用纲要

2.1 使用上层用户函数，高效、简单

如果您只关心通道及频率等基本参数，而不必了解复杂的硬件知识和控制细节，那么我们强烈建议您使用上层用户函数，它们就是几个简单的形如 Win32 API 的函数，具有相当的灵活性、可靠性和高效性。而底层用户则是需了解硬件知识和控制细节、且又需要特殊复杂控制的用户。但不管怎样，我们强烈建议您使用上层函数（在这些函数中，您见不到任何设备地址、寄存器端口、中断号等物理信息，其复杂的控制细节完全封装在上层用户函数中。）对于上层用户函数的使用，您基本上不必参考硬件说明书，除非您需要知道板上 D 型插座等管脚分配情况。

2.2 如何实现 AD 采集与开关量的简便操作

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。

[InitDeviceAD](#) 可以使用 hDevice 句柄以程序查询方式初始化设备的 AD 部件，[StartDeviceAD](#) 可以用 hDevice 句柄实现对 AD 设备的启动，[ReadDeviceAD](#) 函数可以用 hDevice 句柄实现对 AD 数据的采样读取，[StopDeviceAD](#) 函数可以用 hDevice 句柄实现暂停设备，[ReleaseDeviceAD](#) 函数可以用 hDevice 句柄实现停止 AD 采集。

[GetDeviceDI](#) 函数可取得开关量状态，[SetDeviceDO](#) 函数可实现开关量的输出等。

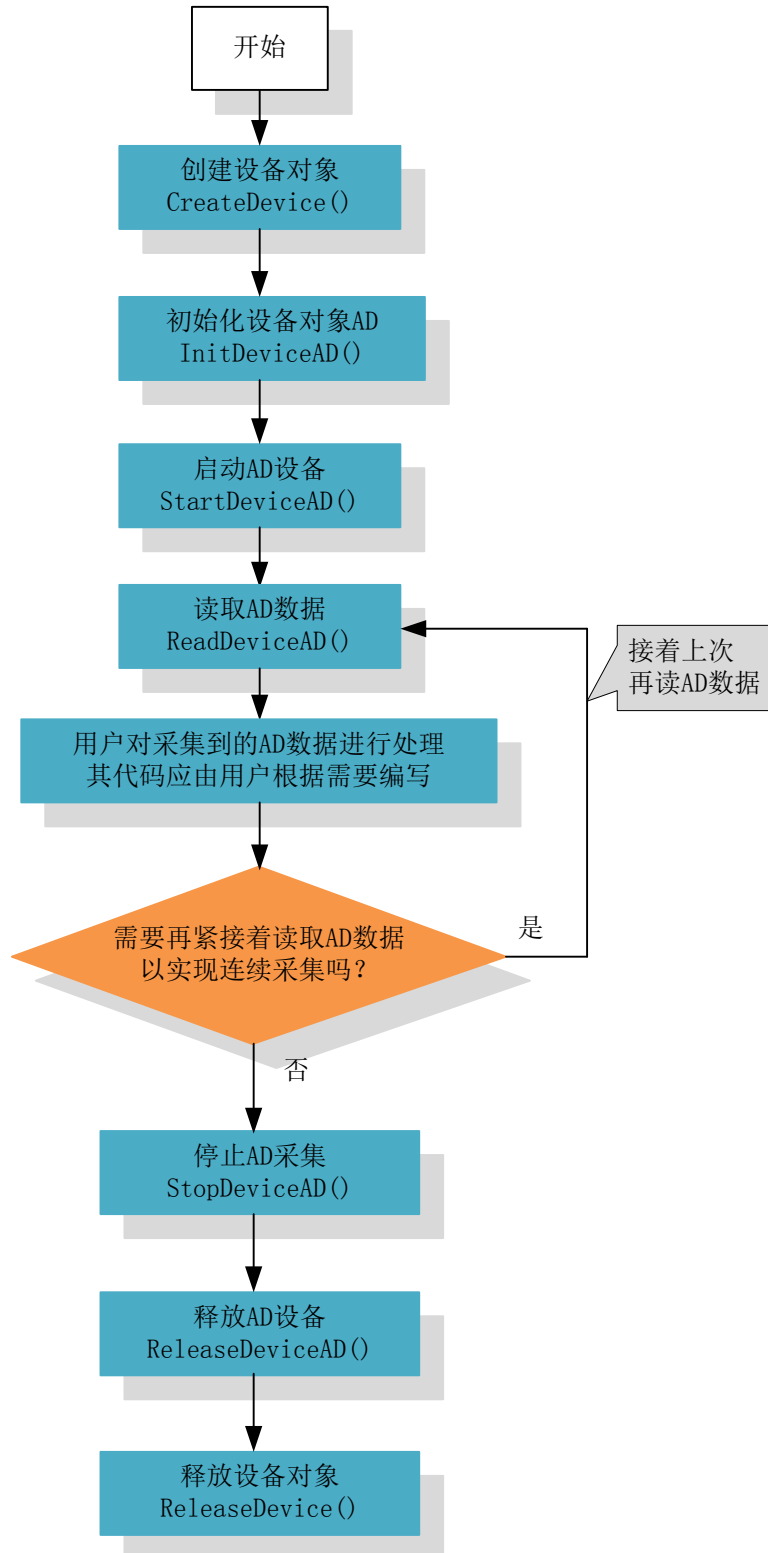
最后可以通过 [ReleaseDevice](#) 将 hDevice 释放掉。

具体流程请参考《[各功能操作流程介绍](#)》章节。

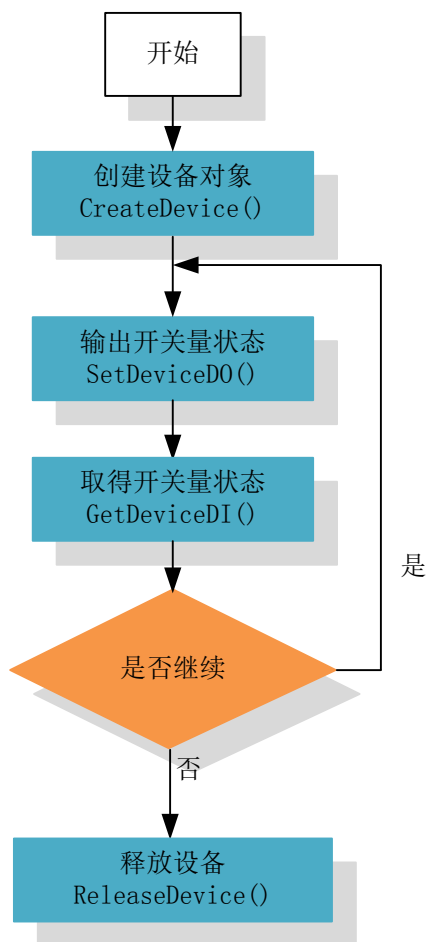
3 各功能操作流程介绍

本卡可进行 AD 模拟量输入、开关输入输出操作，各操作流程图如下。

3.1 AD 模拟量输入操作流程



3.2 开关量输入输出操作流程

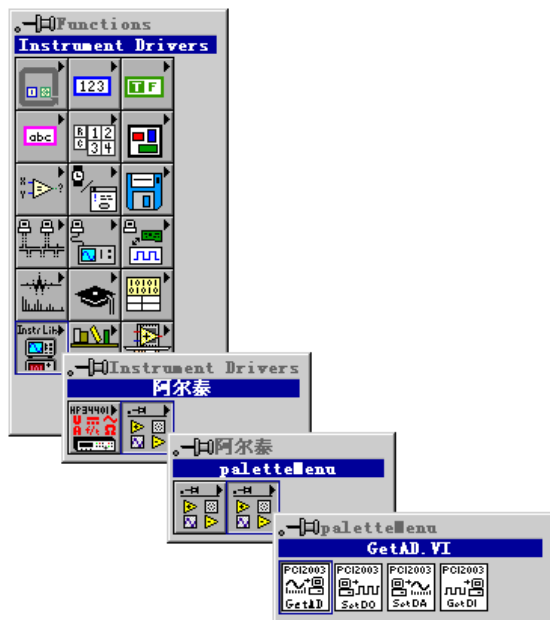


4 USB 设备操作函数接口介绍

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面对的全是他们最关心的问题，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。

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 Library 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++ 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

4.1 设备驱动接口函数总列表

表 4-1-1: 驱动接口函数总列表 (每个函数省略了前缀 “USB288X_”)

函数名	函数功能	备注
① 设备对象管理函数		
CreateDevice	用逻辑号创建设备对象	上层及底层用户
GetDeviceCurrentID	取得当前设备相应的 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	设备句柄	上层及底层用户
② AD 数据读取函数		
ADCalibration	AD 自动校准函数	上层用户
InitDeviceAD	初始化设备	上层用户
StartDeviceAD	启动设备	上层用户
StopDeviceAD	暂停设备	上层用户
ReadDeviceAD	读取设备上的 AD 数据	上层用户
ReleaseDeviceAD	停止 AD 采集, 释放 AD 对象所占资源	上层用户
③ AD 硬件参数操作函数		
SaveParaAD	将当前的 AD 采样参数保存至系统中	上层用户
LoadParaAD	将 AD 采样参数从系统中读出	上层用户
ResetParaAD	将 AD 采样参数恢复至出厂默认值	上层用户
④ 数字 I/O 输入输出函数		
GetDeviceDI	取得开关量状态	上层用户
SetDeviceDO	输出开关量状态	上层用户

使用需知:

Visual C++:



① 要使用如下函数关键的问题是必须在您的源程序中包含如下语句:

#include “C:\Art\USB288X\INCLUDE\USB288X.H” (采用默认路径和默认板号), 用户需根据自己的板号和安装情况确定 USB288X.H 文件的正确路径。

② 用户也可以把此文件拷到您的源程序目录中, 然后加入如下语句: #include “USB288X.H”

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:

(1)、在 LabView 中打开 USB288X.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。

(2)、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,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。

(3)、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。

4.2 设备对象管理函数原型说明

◆ 创建设备对象

函数原型:

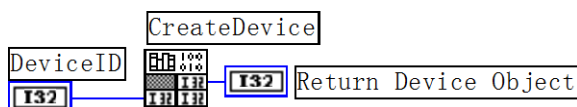
Visual C++:

```
HANDLE CreateDevice(int DeviceLgcID = 0); //创建设备对象(该函数使用系统内逻辑设备 ID)
```

Visual Basic:

```
Declare Function USB2881_CreateDevice Lib "USB2881_32" (ByVal DeviceLgcID As Long_
)As Long '创建设备对象(该函数使用系统内逻辑设备 ID)
```

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,您才能实现对设备所有功能的访问。

参数: DeviceLgcID 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 DeviceLgcID 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。

相关函数: [CreateDevice](#) [ReleaseDevice](#)

◆ 取得当前设备的逻辑 ID 和物理 ID

函数原型:

Visual C++:

```
BOOL GetDeviceCurrentID (HANDLE hDevice, // 取得当前设备的逻辑 ID 号和物理 ID 号
                          PLONG DeviceLgcID);
```

Visual Basic :

```
Declare Function USB2881_GetDeviceCurrentID Lib "USB2881_32" (_
    ByVal hDevice As Long, _
    ByRef DeviceLgcID As Long) As Boolean
'取得当前设备的逻辑 ID 号和物理 ID 号
```

LabVIEW:

请参考相关演示程序。

功能: 取得指定设备当前逻辑 ID 和物理 ID 号。

参数:

hDevice 设备对象句柄，它指向要取得逻辑和物理号的设备，它应由 [CreateDevice](#) 创建。

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

返回值: 如果初始化设备对象成功，则返回 TRUE，否则返回 FALSE。

相关函数: [CreateDevice](#) [GetDeviceCurrentID](#) [ReleaseDevice](#)

◆ 复位整个 USB 设备

函数原型:

Visual C++:

```
BOOL ResetDevice(HANDLE hDevice); // 复位整个 USB 设备
```

Visual Basic :

```
Declare Function USB2881_ResetDevice Lib "USB2881_32" (ByVal hDevice As Long) As Boolean
' 复位整个 USB 设备
```

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，则返回 TRUE，否则返回 FALSE。

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:

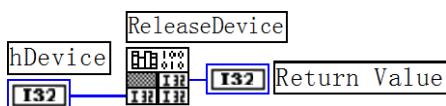
Visual C++:

```
BOOL ReleaseDevice(HANDLE hDevice); // 设备句柄
```

Visual Basic :

```
Declare Function USB2881_ReleaseDevice Lib "USB2881_32" (_
    ByVal hDevice As Long) As Boolean ' 仅释放设备对象
```

LabVIEW:



功能：释放设备对象所占用的系统资源及设备对象自身。

参数：hDevice 设备对象句柄，它应由 [CreateDevice](#) 创建。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关函数：[CreateDevice](#) [ReleaseDevice](#)



CreateDevice 函数必须和 ReleaseDevice 函数一一对应。即当您执行了一次 CreateDevice 后，再执行这些函数前，必须先执行一次 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。

4.3 AD 数据读取函数原型说明

◆ AD 自动校准函数

函数原型：

Visual C++:

```
BOOL ADCalibration(           // AD 自动校准函数
                    HANDLE hDevice); // 设备对象句柄,它由 CreateDevice 函数创建
```

Visual Basic :

```
Declare Function USB2881_ADCalibration Lib "USB2881_32" (ByVal hDevice As Long _
                                                         ) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：AD 自动校准。

参数：hDevice 设备对象句柄，它应由 [CreateDevice](#) 创建。

返回值：若成功，则返回 TRUE，否则返回 FALSE。

相关函数：[CreateDevice](#)

◆ 初始化设备

函数原型：

Visual C++:

```
BOOL InitDeviceAD(           // 初始化设备,返回 TRUE,设备即刻开始传输.
                    HANDLE hDevice,           // 设备句柄,由 CreateDevice 函数创建
                    PUSB288X_PARA_AD pADPara); // 硬件参数,仅在此函数中决定硬件状态
```

Visual Basic :

```
Declare Function USB2881_InitDeviceAD Lib "USB2881_32" (ByVal hDevice As Long, _
                                                         ByVal pADPara As USB2881_PARA_AD) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：初始化设备。

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。

相关函数：[CreateDevice](#)

◆ 启动设备

函数原型:

Visual C++:

```
BOOL StartDeviceAD(           // 在初始化之后, 启动设备
                    HANDLE hDevice); // 设备对象句柄
```

Visual Basic :

```
Declare Function USB2881_StartDeviceAD Lib "USB2881_32" (ByVal hDevice As Long_
                                                         )As Boolean
```

LabVIEW:

请参考相关演示程序。

功能: 启动设备。

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。

相关函数: [CreateDevice](#)

◆ 暂停设备

函数原型:

Visual C++:

```
BOOL StopDeviceAD(           // 在启动设备之后, 暂停设备
                   HANDLE hDevice); // 设备对象句柄
```

Visual Basic :

```
Declare Function USB2881_StopDeviceAD Lib "USB2881_32" (ByVal hDevice As Long_
                                                         ) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能: 暂停设备。

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。

相关函数: [CreateDevice](#)

◆ 读取设备上的 AD 数据

函数原型:

Visual C++:

```
BOOL ReadDeviceAD(           // 初始化设备后, 即可用此函数读取设备上的 AD 数据
                    HANDLE hDevice,           // 设备句柄,它应由 CreateDevice 函数创建
                    USHORT ADBuffer[],       // 将用于接受数据的用户缓冲区
                    LONG nReadSizeWords,     // 读取 AD 数据的长度(字)
                    PLONG nRetSizeWords,     // 实际返回数据的长度(字),
                    BOOL bEnoughRtn = TRUE,  // TRUE: 不够读取点数返回 0;
                                                // FALSE:不够读取点数返回实际点数
                    PLONG nSurplusWords = NULL); // 返回 FIFO 剩余点数
```

Visual Basic :

```
Declare Function USB2881_ReadDeviceAD Lib "USB2881_32" (ByVal hDevice As Long, _
                                                         ByVal ADBuffer As Integer, _
```

```
ByVal nReadSizeWords As Long, _
ByRef nRetSizeWords As Long, _
ByVal bEnoughRtn As Boolean, _
ByRef nSurplusWords As Long _
) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能: 读取设备上的 AD 数据。

参数: hDevice 设备对象句柄，它应由 [CreateDevice](#) 创建。

返回值: 若成功，则返回 TRUE，否则返回 FALSE。

相关函数: [CreateDevice](#)

◆ 停止 AD 采集，释放 AD 对象所占资源

函数原型:

Visual C++:

```
BOOL ReleaseDeviceAD( HANDLE hDevice); // 停止 AD 采集，释放 AD 对象所占资源
```

Visual Basic:

```
Declare Function USB2881_ReleaseDeviceAD Lib "USB2881_32" (ByVal hDevice As Long, _
) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能: 停止 AD 采集，释放 AD 对象所占资源。

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。

相关函数: [CreateDevice](#)

4.4 AD 硬件参数操作函数原型说明

◆ 将当前的 AD 采样参数保存至系统中

函数原型:

Visual C++:

```
BOOL SaveParaAD(
HANDLE hDevice, // 将当前的 AD 采样参数保存至系统中
PUSB288X_PARA_AD pADPara);
```

Visual Basic:

```
Declare Function USB2881_SaveParaAD Lib "USB2881_32" (ByVal hDevice As Long, _
ByRef pADPara As USB2881_PARA_AD _
) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能: 将当前的 AD 采样参数保存至系统中。

参数: hDevice 设备对象句柄，它应由 [CreateDevice](#) 创建。

返回值: 若成功，则返回 TRUE，否则返回 FALSE。

相关函数: [CreateDevice](#)

◆ 将 AD 采样参数从系统中读出

函数原型:

Visual C++:

```
BOOL LoadParaAD(
    HANDLE hDevice, // 将 AD 采样参数从系统中读出
    PUSB288X_PARA_AD pADPara);
```

Visual Basic :

```
Declare Function USB2881_LoadParaAD Lib "USB2881_32" (ByVal hDevice As Long, _
    ByRef pADPara As USB2881_PARA_AD) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能: 将 AD 采样参数从系统中读出。

参数: hDevice 设备对象句柄, 它应由 [CreateDevice](#) 创建。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

相关函数: [CreateDevice](#)

◆ 将 AD 采样参数恢复至出厂默认值

Visual C++:

```
BOOL ResetParaAD( // 将 AD 采样参数恢复至出厂默认值
    HANDLE hDevice,
    PUSB288X_PARA_AD pADPara);
```

Visual Basic :

```
Declare Function USB2881_ResetParaAD Lib "USB2881_32" (ByVal hDevice As Long, _
    ByRef pADPara As USB2881_PARA_AD) As Boolean
```

LabVIEW:

请参考相关演示程序。

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。

返回值: 若成功, 则返回 TRUE, 否则返回 FALSE。

相关函数: [CreateDevice](#)

4.5 数字 I/O 输入输出函数原型说明

◆ 取得数字量状态

函数原型:

Visual C++:

```
BOOL GetDeviceDI( // 取得开关量状态
    HANDLE hDevice, // 设备对象句柄,它由 CreateDevice 函数创建
    BYTE bDISTs [16]); // 开关输入状态(注意: 必须定义为 16 个字节元素的数组)
```

Visual Basic :

```
Declare Function USB2881_GetDeviceDI Lib "USB2881_32" (ByVal hDevice As Long, _
    ByRef bDISTs As Byte) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：负责将 USB 设备上的 DI0~DI15 输入开关量状态读入内存。

参数：

hDevice 设备对象句柄，它应由 [CreateDevice](#) 创建。

bDISts 十六路开关量输入状态的参数结构，共有 16 个成员变量，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。

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](#) [GetDeviceDI](#) [ReleaseDevice](#)

◆ 输出数字量状态

函数原型：

Visual C++:

```
BOOL SetDeviceDO(           // 输出开关量状态
                  HANDLE hDevice, // 设备对象句柄,它由 CreateDevice 函数创建
                  BYTE bDOSets[16]); // 开关输出状态(注意: 必须定义为 16 个字节元素的数组)
```

Visual Basic:

```
Declare Function USB2881_SetDeviceDO Lib "USB2881_32" (ByVal hDevice As Long, _
                                                    ByRef bDOSets As Byte) As Boolean
```

LabView:

请参考相关演示程序。

功能：负责将 USB 设备上的 DO0~DO15 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](#) 创建。

bDOSets 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 bDOSets[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”。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： [CreateDevice](#) [SetDeviceDO](#) [ReleaseDevice](#)



以上函数调用一般顺序详见 DIO 操作流程

5 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）。

5.1 简易程序演示说明

5.1.1 怎样进行开关量输入输出操作

其详细应用实例及正确代码请参考 Visual C++简易程序演示系统及源程序，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 USB288X.h 和 Sys.cpp)。
[程序] | [阿尔泰测控演示系统] | [USB288X 数据采集卡] | [Microsoft Visual C++] | [简易代码演示] | [DIO 简易源程序]

其默认存放路径为：系统盘\ART\USB288X\SAMPLES\VC\SIMPLE\DIO

5.1.2 怎样进行 AD 数据采集操作

其详细应用实例及正确代码请参考 Visual C++简易程序演示系统及源程序，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 USB288X.h 和 Sys.cpp)。
[程序] | [阿尔泰测控演示系统] | [USB288X 数据采集卡] | [Microsoft Visual C++] | [简易代码演示] | [AD 简易源程序]

其默认存放路径为：系统盘\ART\USB288X\SAMPLES\VC\SIMPLE\AD

5.2 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 USB288X.h 和 Sys.cpp)。

[程序] | [阿尔泰测控演示系统] | [USB288X 数据采集卡] | [Microsoft Visual C++] | [高级代码演示]

其默认存放路径为：系统盘\ART\USB288X\SAMPLES\VC\ADVANCED

其他语言的演示可以用上面类似的方法找到。

6 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。

6.1 公用接口函数总列表

表 6-1-1: 公用接口函数总列表（每个函数省略了前缀“USB288X_”）

函数名	函数功能	备注
GetDevVersion	获取设备固件及程序版本	底层用户
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	

6.2 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 (                // 获取设备固件及程序版本
                    HANDLE hDevice, // 设备对象句柄,它由 CreateDevice 函数创建
                    PULONG pulFmwVersion, // 固件版本
                    PULONG pulDriverVersion); // 驱动版本
```

Visual Basic:

```
Declare Function USB2881_GetDevVersion Lib "USB2881_32" (ByVal hDevice As Long, _
                                                         ByRef pulFmwVersion As Long, _
                                                         ByRef pulDriverVersion As Long) As Boolean
```

LabVIEW:

请参考相关演示程序。

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 固件版本。

pulDriverVersion 驱动版本。

返回值: 若成功，返回 TRUE，否则返回 FALSE。

相关函数: [CreateDevice](#) [GetDevVersion](#) [ReleaseDevice](#)

◆ 创建内核系统事件

函数原型:

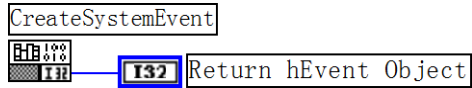
Visual C++:

```
HANDLE CreateSystemEvent(void); // 创建内核系统事件对象
```

Visual Basic:

```
Declare Function USB2881_CreateSystemEvent Lib "USB2881_32" () As Long
```

LabVIEW:



功能：创建系统内核事件对象。

参数：无任何参数。

返回值：若成功，返回系统内核事件对象句柄，否则返回-1(或 INVALID_HANDLE_VALUE)。

◆ 释放内核系统事件

函数原型：

Visual C++:

`BOOL ReleaseSystemEvent(HANDLE hEvent);` // 释放内核事件对象

Visual Basic :

`Declare Function USB2881_ReleaseSystemEvent Lib "USB2881_32" (ByVal hEvent As Long_) As Boolean`

LabVIEW:

请参见相关演示程序。

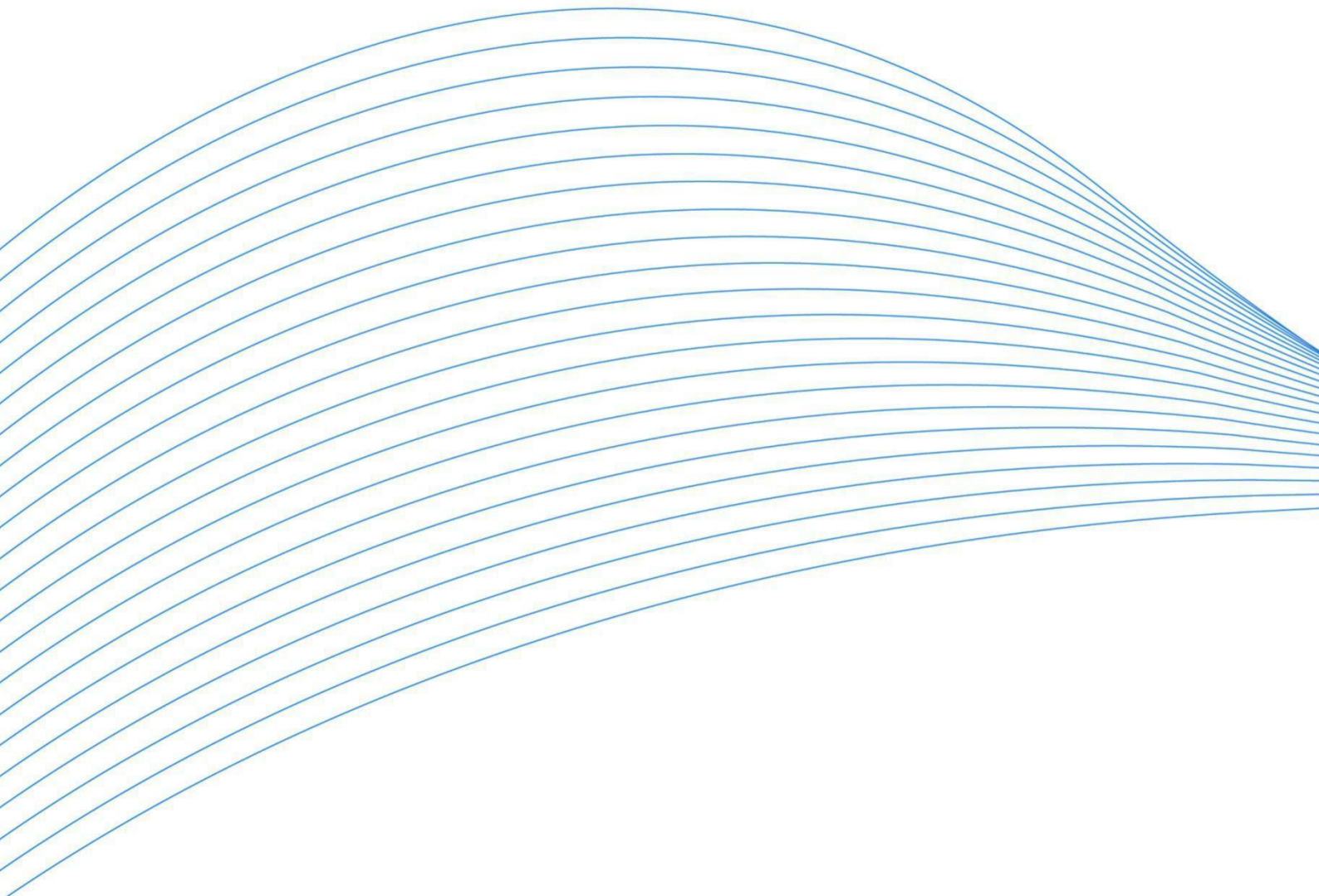
功能：释放系统内核事件对象。

参数：`hEvent` 被释放的内核事件对象。它应由 [ReleaseSystemEvent](#) 成功创建的对象。

返回值：若成功，则返回 `TRUE`。

7 修改历史

修改时间	版本号	修改内容
2015.8.26	V6.00.00	第一版
2016.3.11	V6.00.01	增加 Visual Basic 示例，完善 ReadDeviceAD 函数



北京阿尔泰科技发展有限公司

服务热线：400-860-3335

邮编：100086

传真：010-62901157