USB3020 软件说明书

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

目　录 1
第一章 版权信息与命名约定 2
第一节、版权信息 2
第二节、命名约定 2
第二章 使用纲要 3
第一节、使用上层用户函数，高效、简单 3
第二节、如何管理设备 3
第三节、哪些函数对您不是必须的 3
第三章 USB 设备操作函数接口介绍 4
第一节、设备驱动接口函数总列表（每个函数省略了前缀“USB3020_”） 4
第二节、设备对象管理函数原型说明 6
第三节、DA 模拟量输出操作函数原型说明 9
第四节、DA 校准函数说明 13
第五节、DA 硬件参数系统保存与读取函数原型说明 15
第四章 硬件参数结构 17
第一节、DA 硬件参数介绍（USB3020_PARA_DA） 17
第二节、DA 状态参数结构（USB3020_STATUS_DA） 18
第五章 数据格式转换与排列规则 19
第一节、DA 电压值转换成 LSB 原码数据的换算方法 19
第二节、关于 DA 数据 DABuffer 缓冲区中的数据排放规则 19
第六章 上层用户函数接口应用实例 20
第一节、简易程序演示说明 20
第二节、高级程序演示说明 20
第七章 共用函数介绍 20
第一节、公用接口函数总列表（每个函数省略了前缀“USB3020_”） 20
第二节、USB 内存映射寄存器操作函数原型说明 20

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 USB3020_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、InitDeviceProAD 、 InitDeviceDmaAD 、 ReadDeviceProAD-Npt 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InitDeviceProAD可以使用hDevice句柄以程序查询方式初始化设备的AD部件，ReadDeviceProAD Npt 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

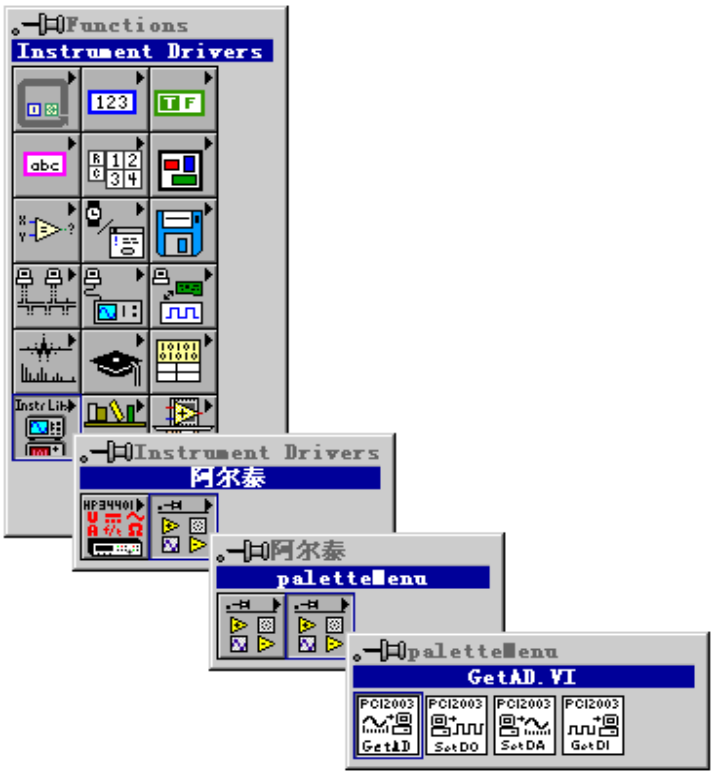
公共函数如CreateFileObject，WriteFile，ReadFile等一般来说都是辅助性函数，除非您要使用存盘功能。如果您使用上层用户函数访问设备，那么GetDeviceAddr，WriteRegisterByte，WriteRegisterWord，WriteRegisterULong，ReadRegisterByte，ReadRegisterWord，ReadRegisterULong等函数您可完全不必理会，除非您是作为底层用户管理设备。而WritePortByte，WritePortWord，WritePortULong，ReadPortByte， ReadPortWord，ReadPortULong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ReadDeviceAD_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，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“USB3020_”）

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
GetDeviceCount	取得同一种 USB 设备的总台数	上层及底层用户
GetDeviceCurrentID	取得当前设备的逻辑 ID 号和物理 ID 号	上层及底层用户
ListDeviceDlg	列表所有同一种 USB 设备的各种配置	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
ResetDevice	复位整个 USB 设备	上层及底层用户
②DA 模拟量输出操作函数		
InitDeviceDA	初始化设备，当返回 TRUE 后，设备即准备就绪	上层用户
WriteDeviceBulkDA	在 DA 输出前，用此函数将 DA 数据写入板载 RAM 中(程序方式)	上层用户
EnableDeviceDA	在初始化之后，使能设备	上层用户
SetDeviceTrigDA	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
GetDevStatusDA	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
DisableDeviceDA	在使设备之后，禁止设备	上层用户
ReleaseDeviceDA	禁止 DA 设备，且释放资源	上层用户
StartCalibration	启动 DA 校准	
GetDACalibration	设备 DA 校准	上层用户
SetDACalibration	设备 DA 校准	上层用户
StopCalibration	停止 DA 校准	上层用户
③DA 硬件参数系统保存、读取函数		
LoadParaDA	从 Windows 系统中读入硬件参数	上层用户
SaveParaDA	往 Windows 系统写入设备硬件参数	上层用户
ResetParaDA	将硬件参数结构体值复位为出厂默认值	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\USB3020\INCLUDE\USB3020.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 USB3020.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。

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 USB3020.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 USB3020.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：

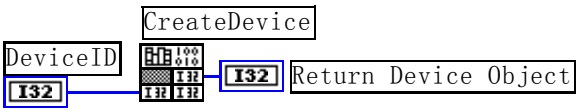
Visual C++：

HANDLE CreateDevice (int DeviceLgcID = 0)

Visual Basic：

Declare Function USB3020_CreateDevice Lib "USB3020_32" (ByVal DeviceLgcID As Integer) 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 GetDeviceCount GetDeviceCurrentID ResetDevice ListDeviceDlg ReleaseDevice

Visual C++程序举例

```
    :
HANDLE hDevice;     // 定义设备对象句柄
int DeviceLgcID = 0;
hDevice = USB3020_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 = USB3020_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

：

♦ **取得本计算机系统中 USB3020 设备的总数量**

函数原型：

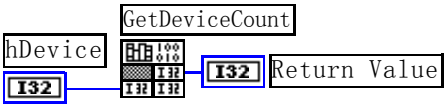
Visual C++:

int GetDeviceCount (HANDLE hDevice)

Visual Basic:

Declare Function USB3020_GetDeviceCount Lib "USB3020_32" (ByVal hDevice As Long) As Integer

LabVIEW：



功能：取得 USB3020 设备的数量。

参数：

hDevice设备对象句柄，它应由CreateDevice创建。

返回值：返回系统中 USB3020 的数量。

相关函数： CreateDevice GetDeviceCount GetDeviceCurrentID ResetDevice ListDeviceDlg ReleaseDevice

♦ **用对话框控件列表计算机系统中所有 USB3020 设备各种配置信息**

函数原型：

Visual C++:

BOOL ListDeviceDlg (void)

Visual Basic:

Declare Function USB3020_ListDeviceDlg Lib "USB3020_32" () As Boolean

LabVIEW：

请参考相关演示程序。

功能：列表系统中 USB3020 的硬件配置信息。

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 USB3020 设备的配置情况。

相关函数： CreateDevice ReleaseDevice

♦ **取得该设备当前逻辑 ID 和物理 ID 号**

函数原型：

Visual C++:

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

Visual Basic:

```
Declare Function USB3020_GetDeviceCurrentID Lib "USB3020_32" (_
                                        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：取得指定设备逻辑 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]。

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 GetDeviceCount GetDeviceCurrentID ResetDevice ListDeviceDlg ReleaseDevice

♦ **释放设备对象所占的系统资源及设备对象**

函数原型：

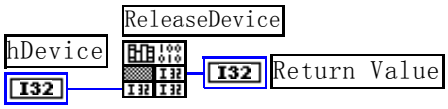
Visual C++:

BOOL ReleaseDevice (HANDLE hDevice)

Visual Basic:

Declare Function USB3020_ReleaseDevice Lib "USB3020_32" (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 GetDeviceCount GetDeviceCurrentID ResetDevice ListDeviceDlg 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。

♦ **复位整个 USB 设备**

函数原型：

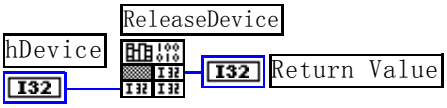
Visual C++:

BOOL ResetDevice (HANDLE hDevice)

Visual Basic:

Declare Function USB3020_ResetDevice Lib "USB3020_32" (ByVal hDevice As Long) As Boolean

LabVIEW：



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，用户可以用GetLastErrorEx捕获错误码。

相关函数： CreateDevice GetDeviceCount GetDeviceCurrentID ResetDevice ListDeviceDlg ReleaseDevice

第三节、DA 模拟量输出操作函数原型说明

♦ **初始化设备对象**

函数原型

Visual C++:

```
BOOL InitDeviceDA (HANDLE hDevice,
                   LONG SegmentCount,
                   USB3020_SEGMENT_INFO SegmentInfo[],
                   PUSB3020_PARA_DA pDAPara,
                   int nDAChannel)
```

Visual Basic:

```
Declare Function USB3020_InitDeviceDA Lib "USB3020_32" (_
                   ByVal hDevice As Long, _
                   ByVal SegmentCount As Long,_
                   ByRef SegmentInfo As USB3020_SEGMENT_INFO, _
                   ByRef pDAPara As USB3020_PARA_DA, _
                   ByVal nDAChannel As Integer) 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DA部件，为设备操作就绪有关工作做准备，如预置DA采集通道、采样频率等。但它并不启动DA设备，若要启动DA设备，须在调用此函数之后再调用EnableDeviceDA(但DA要实际输出波形，则一般要等待某种触发事件的到来)。

参数：

hDevice 设备对象句柄，它应由CreateDevice或CreateDeviceEx创建。

SegmentCount 分段总数，它的理论取值范围为[1, 65536]，但实际工作时，它要受到板载 RAM 大小、各段数据长度大小及其他通道对 RAM 的使用情况等因素的影响。在该通道分配的板载 RAM 空间一定的情况下，其各段数据长度越短，则可分配的段数越多。

SegmentInfo[] 段信息集合，属于 USB3020_SEGMENT_INFO 的结构数据类型，它的有效元素个数由 SegmentCount 参数决定。因此用户分配的段信息集合不应小于 SegmentCount 的指定的大小。注意此段信息集合将被此函数写入板载 RAM 中，它与后面的 DA 数据区共享该通道指定的板载 RAM 区域。

pDAPara 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，如采样频率等。关于具体操作请参考《DA硬件参数结构》。

nDAChannel DA 通道号，取值范围为[0, 3]。

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 SetDACalibration StopCalibration StartCalibration InitDeviceDA GetDACalibration WriteDeviceBulkDA EnableDeviceDA SetDeviceTrigDA GetDevStatusDA DisableDeviceDA ReleaseDeviceDA ReleaseDevice

◆ **批量方式将用户缓冲区中的 DA 数据传输至板载 RAM 中**

函数原型:

Visual C++:

```
BOOL WriteDeviceBulkDA (HANDLE hDevice,
                        USHORT DABuffer[],
                        LONG nWriteSizeWords,
                        PLONG nRetSizeWords,
                        int nDAChannel)
```

Visual Basic:

```
Declare Function USB3020_WriteDeviceBulkDA Lib "USB3020_32" (_
                                    ByVal hDevice As Long,_
                                    ByRef DABuffer As Integer,_
                                    ByVal nWriteSizeWords As Long,_
                                    ByRef nRetSizeWords As Long,_
                                    ByVal nDAChannel As Integer) As Boolean
```

LabView：

请参考相关演示程序。

功能：往指定通道的板载 RAM 中写入批量 DA 数据。在初始化设备之后，启动 DA 之前，便可以用此函数将 DA 数据写入板载 RAM 以供输出。但是在启动之后（即在输出过程中，不能对 RAM 进行写操作，包括读操作）。

参数：

hDevice 设备对象句柄,它应由CreateDevice或CreateDeviceEx创建。

DABuffer[] 接受DA数据的用户缓冲区地址，它可以是一个 16Bit整型数组，也可以是由其他方式分配的 16Bit整型缓冲区。关于如何将这些DA数据转换成相应的电压值，请参考《数据格式转换与排列规则》。

nWriteOffsetWords 相对于该通道物理 RAM 零位置的偏移点(字)。此参数不能超过 RAM 的最大长度(字/点)。

nWriteSizeWords 指定一次往物理缓冲区由 nWriteOffsetWords 参数指定偏移位置开始写入的数据长度。注意此参数的值与 nWriteOffsetWords 参数值之和不能大于指定通道的物理缓冲区即板上 RAM 的最大长度。同时此参数值不能大于 DABuffer[]指定的缓冲区的长度。

nRetSizeWords 返回当前写操作实际实现的数据长度。它表明该函数调用后，在 DABuffer[]中有多少数据是有效的。

nDAChannel DA 通道号，取值范围为[0, 1]。

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 SetDACalibration StopCalibration StartCalibration InitDeviceDA GetDACalibration WriteDeviceBulkDA EnableDeviceDA SetDeviceTrigDA GetDevStatusDA DisableDeviceDA ReleaseDeviceDA ReleaseDevice

◆ **启动 DA 设备**

函数原型

Visual C++:

```
BOOL EnableDeviceDA (HANDLE hDevice,
                     int nDAChannel)
```

Visual Basic:

```
Declare Function USB3020_EnableDeviceDA Lib "USB3020_32" (_
                                        ByVal hDevice As Long,_
                                        ByVal nDAChannel As Integer) As Boolean
```

LabVIEW：

请参考相关演示程序。

功能：启动DA设备，它必须在调用InitDeviceDA后才能调用此函数。调用该函数后它可能立即启动，这就要取决您选择的触发方式或触发源，详细请参考后面的《触发功能详述》。

参数：

hDevice 设备对象句柄，它应由CreateDevice或CreateDeviceEx创建。

nDAChannel 通道号，取值范围为[0, 3]。

返回值：如果调用成功，则返回TRUE，且DA准备就绪，等待触发事件的到来就开始实际的DA输出， 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　SetDACalibration　StopCalibration
StartCalibration　InitDeviceDA　GetDACalibration
WriteDeviceBulkDA　EnableDeviceDA　SetDeviceTrigDA
GetDevStatusDA　DisableDeviceDA　ReleaseDeviceDA
ReleaseDevice

♦软件产生触发事件

函数原型：

Visual C++：

```
BOOL SetDeviceTrigDA (HANDLE hDevice ,
                      BOOL bSetSyncTrig,
                      int nDAChannel)
```

Visual Basic:

```
Declare Function USB3020_SetDeviceTrigDA Lib "USB3020_32" (_
                                        ByVal hDevice As Long,_
                                        ByVal bSetSyncTrig As Boolean,_
                                        ByVal nDAChannel As Integer) As Boolean
```

请参考相关演示程序。

功能：在指定通道被启动后，可由此函数以软件产生触发事件。当然只有当用户选择触发源为软件触发时方有效。

参数：

hDevice 设备对象句柄，它应由CreateDevice或CreateDeviceEx创建。

bSetSyncTrig 是否置同步触发。

nDAChannel 通道号，取值范围为[0, 3]。

返回值：如果调用成功，则返回TRUE，则产生一个软件触发事件，在某些触发模式下会直接使DA输出波形， 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　SetDACalibration　StopCalibration
StartCalibration　InitDeviceDA　GetDACalibration
WriteDeviceBulkDA　EnableDeviceDA　SetDeviceTrigDA
GetDevStatusDA　DisableDeviceDA　ReleaseDeviceDA
ReleaseDevice

♦ 取得 DA 的状态标志

函数原型：

Visual C++:

```
BOOL GetDevStatusDA (HANDLE hDevice,
                     PUSB3020_STATUS_DA pDAStatus,
```

int nDAChannel)

Visual Basic:

```
Declare Function USB3020_GetDevStatusDA Lib "USB3020_32" (_
                                        ByVal hDevice As Long, _
                                        ByRef pDAStatus As USB3020_STATUS_DA, _
                                        ByVal nDAChannel As Integer) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：一旦用户使用EnableDeviceDA后，可以用此函数却查询DA状态，如是否被启动（bConverting），触发标志是否有效(bTrigFlag)，当前段循环次数(nCurSegLoopCount)等信息。

参数：

hDevice 设备对象句柄，它应由CreateDevice或CreateDeviceEx创建。

pDAStatus 设备状态参数结构，它返回设备当前的各种状态，如板载RAM是否发生切换、重写、触发点是否产生等信息。关于具体操作请参考《DA硬件参数结构》。

nDAChannel 通道号，取值范围为[0, 1]。

返回值：若 DA 成功取回标状态，则返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice SetDACalibration StopCalibration StartCalibration InitDeviceDA GetDACalibration WriteDeviceBulkDA EnableDeviceDA SetDeviceTrigDA GetDevStatusDA DisableDeviceDA ReleaseDeviceDA ReleaseDevice

◆ **暂停 DA 设备**

函数原型

Visual C++：

```
BOOL DisableDeviceDA (HANDLE hDevice,
                      int nDAChannel )
```

Visual Basic:

```
Declare Function USB3020_DisableDeviceDA Lib "USB3020_32" (_
                                         ByVal hDevice As Long, _
                                         ByVal nDAChannel As Integer) As Boolean
```

LabVIEW：

请参考相关演示程序。

功能：暂停DA设备。它必须在调用EnableDeviceDA后才能调用此函数。该函数除了停止DA设备不再转换以外，不改变设备的其他任何工作参数。

参数：

hDevice 设备对象句柄，它应由CreateDevice或CreateDeviceEx创建。

nDAChannel 通道号，取值范围为[0, 3]。

返回值：如果调用成功，则返回TRUE，且DA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 SetDACalibration StopCalibration StartCalibration InitDeviceDA GetDACalibration WriteDeviceBulkDA EnableDeviceDA SetDeviceTrigDA GetDevStatusDA DisableDeviceDA ReleaseDeviceDA ReleaseDevice

◆ **释放 DA 设备**

函数原型

Visual C++：

```
BOOL ReleaseDeviceDA (HANDLE hDevice,
```

int nDAChannel)

Visual Basic:

Declare Function USB3020_ReleaseDeviceDA Lib "USB3020_32" (_
ByVal hDevice As Long,_
ByVal nDAChannel As Integer) As Boolean

LabView：

请参考相关演示程序。

功能：释放DA设备。它必须在调用InitDeviceDA后的某个时刻调用此函数。该函数除了停止DA设备，还释放掉所占用的各种资源。

参数：

hDevice 设备对象句柄，它应由CreateDevice或CreateDeviceEx创建。

nDAChannel 设备通道号，取值范围为[0, 1]。

返回值：如果调用成功，则返回TRUE，且DA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 SetDACalibration StopCalibration StartCalibration InitDeviceDA GetDACalibration WriteDeviceBulkDA EnableDeviceDA SetDeviceTrigDA GetDevStatusDA DisableDeviceDA ReleaseDeviceDA ReleaseDevice

♦ **采样和传输函数一般调用顺序**

①CreateDevice (创建设备对象)
②InitDeviceDA (初始化设备)
③ WriteDeviceBulkDA(批量写入DA数据到板载RAM)
④EnableDeviceDA (启动DA设备)
⑤SetDevTrigLevelDA (若为软件触发源，则置软件触发事件)
⑥ GetDevStatusDA（循环查询DA状态）
⑦ DisableDeviceDA
⑧ ReleaseDevice
关于调用过程的图形说明请参考《绪论》。

第四节、DA 校准函数说明

♦ **启动 DA 校准**

函数原型：

Visual C++:

BOOL StartCalibration (HANDLE hDevice)

Visual Basic:

Declare Function USB3020_StartCalibration Lib "USB3020_32" (ByVal hDevice As Long) As Boolean

LabView：

请参考相关演示程序。

功能：启动 DA 校准。

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。

相关函数： CreateDevice SetDACalibration StopCalibration StartCalibration InitDeviceDA ReleaseDevice

♦设备 DA 校准

函数原型：

Visual C++:

```
BOOL GetDACalibration (HANDLE hDevice,
                       LONG OutputRange,
                       LONG CalMode,
                       PLONG pCalData,
                       int nDAChannel)
```

Visual Basic:

```
Declare Function USB3020_GetDACalibration Lib "USB3020_32" ( _
                                                  ByVal hDevice As Long, _
                                                  ByVal OutputRange As Long, _
                                                  ByVal CalMode As Long, _
                                                  ByRef pCalData As Long, _
                                                  ByVal nDAChannel As Long) As Boolean
```

LabView：

请参考相关演示程序。

功能：设备 DA 校准。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

OutputRange 输出量程，分别控制两个通道。

CalMode 0 为零点校准，1 为满度校准。

pCalData 校准值。

nDAChannel DA 通道号[0, 3]。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice SetDACalibration StopCalibration StartCalibration InitDeviceDA ReleaseDevice

♦设备 DA 校准

函数原型：

Visual C++:

```
BOOL SetDACalibration (HANDLE hDevice,
                       LONG OutputRange,
                       LONG CalMode,
                       LONG CalData,
                       int nDAChannel)
```

Visual Basic:

```
Declare Function USB3020_SetDACalibration Lib "USB3020_32" ( _
                                                  ByVal hDevice As Long, _
                                                  ByVal OutputRange As Long, _
                                                  ByVal CalMode As Long, _
                                                  ByVal CalData As Long, _
                                                  ByVal nDAChannel As Long) As Boolean
```

LabView：

请参考相关演示程序。

功能：设备 DA 校准。

参数：

hDevice 设备对象句柄，它应由CreateDevice创建。

OutputRange 输出量程,分别控制两个通道。
CalMode 0 为零点校准，1 为满度校准。
pCalData 校准值。
nDAChannel DA 通道号[0, 3]。
返回值：如果调用成功，则返回 TRUE，否则返回 FALSE。
相关函数： CreateDevice SetDACalibration StopCalibration StartCalibration InitDeviceDA ReleaseDevice

◆ **停止 DA 校准**

函数原型：
Visual C++:
BOOL StopCalibration (HANDLE hDevice)
Visual Basic:
Declare Function USB3020_StopCalibration Lib "USB3020_32" (ByVal hDevice As Long) As Boolean
LabView：
请参考相关演示程序。

功能：停止 DA 校准。
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。
相关函数： CreateDevice SetDACalibration StopCalibration StartCalibration InitDeviceDA ReleaseDevice

第五节、DA 硬件参数系统保存与读取函数原型说明

◆ **写设备硬件参数函数到 Windows 系统中**

函数原型：
Viusal C++:
```
BOOL SaveParaDA (HANDLE hDevice,
                 PUSB3020_PARA_DA pDAPara,
                 int nDAChannel)
```
Visual Basic:
```
Declare Function USB3020_SaveParaDA Lib "USB3020_32" (_
                                        ByVal hDevice As Long,_
                                        ByRef pDAPara As USB3020_PARA_DA, _
                                        ByVal nDAChannel As Integer) 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或CreateDeviceEx创建。
pDAPara设备硬件参数，关于USB3020_PARA_DA的详细介绍请参考USB3020.h或USB3020.Bas或USB3020.Pas函数原型定义文件，也可参考《硬件参数结构》关于该结构的有关说明。
nDAChannel DA 通道号，取值范围为[0, 1]。
返回值：若成功，返回 TRUE，否则返回 FALSE。
相关函数： CreateDevice LoadParaDA SaveParaDA ReleaseDevice

◆ **从 Windows 系统中读入硬件参数函数**

函数原型：

Visual C++:

```
BOOL LoadParaDA (HANDLE hDevice,
                 PUSB3020_PARA_DA pDAPara,
                 int nDAChannel)
```

Visual Basic:

```
Declare Function USB3020_LoadParaDA Lib "USB3020_32" (_
                                    ByVal hDevice As Long,_
                                    ByRef pDAPara As USB3020_PARA_DA, _
                                    ByVal nDAChannel As Integer) 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或CreateDeviceEx创建。

pDAPara 属于USB3020_PARA_DA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USB3020_PARA_DA请参考USB3020.h或USB3020.Bas或USB3020.Pas函数原型定义文件，也可参考《硬件参数结构》关于该结构的有关说明。

nDAChannel DA 通道号，取值范围为[0, 1]。

返回值：若成功，返回 TRUE，否则返回 FALSE。

相关函数： CreateDevice LoadParaDA SaveParaDA ReleaseDevice

◆ **将硬件参数结构体值复位为出厂默认值**

函数原型：

Viusal C++:

```
BOOL ResetParaDA (HANDLE hDevice,
                  PUSB3020_PARA_DA pDAPara,
                  int nDAChannel)
```

Visual Basic:

```
Declare Function USB3020_ResetParaDA Lib "USB3020_32" (_
                                     ByVal hDevice As Long, _
                                     ByRef pDAPara As USB3020_PARA_DA, _
                                     ByVal nDAChannel As Integer) As Boolean
```

LabVIEW:

请参考相关演示程序。

功能：负责将硬件参数的值复位至出厂默认值，不仅会将 pDAPara 指向的结构体成员值更新为默认值，同时会将系统中保存的参数更新为默认值。这些默认值在产品驱动第一次被安装时会出现。而且这些默认值的设定是充分的考虑到用户的实际情况，确保用户不用外部任何条件，只要开始采集数据，即可获得相应的结果。

参数：

hDevice 设备对象句柄，它应由CreateDevice或CreateDeviceEx创建。

pDAPara设备硬件参数，关于USB3020_PARA_DA的详细介绍请参考USB3020.h或USB3020.Bas或USB3020.Pas函数原型定义文件，也可参考《硬件参数结构》关于该结构的有关说明。调用此函数后，该参数指向的结构体成员将被复位至默认值。

nDAChannel DA 通道号，取值范围为[0, 1]。

返回值：若成功，返回 TRUE，它表明已成功将系统中的 DA 参数复位至默认值，同时更新了 pDAPara 指向的结构体。否则返回 FALSE。

相关函数： CreateDevice LoadParaDA SaveParaDA ResetParaDA ReleaseDevice

第四章 硬件参数结构

第一节、DA 硬件参数介绍（USB3020_PARA_DA）

Visual C++:

```
typedef struct _USB3020_SEGMENT_INFO
    {
        LONG SegLoopCount;          // 每个段在大循环中的小循环次数,取值为[1, 16777215]
        LONG SegmentSize;       // 每个段在 RAM 中的长度(单位：字/点)
    } USB3020_SEGMENT_INFO, *PUSB3020_SEGMENT_INFO;

// DA 的工作参数
    typedef struct _USB3020_PARA_DA
      {
        LONG OutputRange;       // 输出量程
        LONG Frequency;             // 点频率[0.010Hz, 1MHz], 为正数时单位 Hz，为负数时单位为 0.001Hz
        LONG LoopCount;             // 整过 RAM 的大循环次数,=0:无限循环, =n:表示 n 次循环(1<n<32768)
        LONG TriggerMode;       // 触发模式选择
        LONG TriggerSource;         // 触发源选择
        LONG TriggerDir;        // 触发方向选择
        LONG bSingleOut;        // 是否单点输出
        LONG ClockSource;       // 时钟源选择
    } USB3020_PARA_DA, *PUSB3020_PARA_DA;
```

Visual Basic：

```
Type USB3020_SEGMENT_INFO

    SegLoopCount As Long      ' 每个段在大循环中的小循环次数,取值为[1, 16777215]
    SegmentSize As Long       ' 每个段在 RAM 中的长度(单位：字/点)
End Type

' DA 的工作参数
Type USB3020_PARA_DA

    OutputRange As Long         ' 输出量程
    Frequency As Long           ' 点频率[0.010Hz, 1MHz], 为正数时单位 Hz，为负数时单位为 0.001Hz
    LoopCount As Long           ' 整过 RAM 的大循环次数,=0:无限循环, =n:表示 n 次循环(1<n<32768)
    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
    bSingleOut As Long          ' 是否单点输出
    ClockSource As Long         ' 时钟源选择
End Type
```

LabVIEW:

请参考相关演示程序。

此结构主要用于设定设备DA硬件参数值，用这个参数结构对设备进行硬件配置完全由InitDeviceDA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。

OutputRange 模拟量输出量程选择。

常量名	常量值	功能定义
USB3020_OUTPUT_N5000_P5000mV	0x01	±5000mV

USB3020_OUTPUT_N10000_P10000mV	0x02	±10000mV

TriggerMode DA 触发模式选择。

常量名	常量值	功能定义
USB3020_TRIGMODE_SINGLE	0x00	单次触发
USB3020_TRIGMODE_CONTINOUS	0x01	连续触发
USB3020_TRIGMODE_STEPED	0x02	单步触发
USB3020_TRIGMODE_BURST	0x03	紧急触发

TriggerSource DA 触发源选择。

常量名	常量值	功能定义
USB3020_TRIGSRC_SOFT_P5000mV	0x00	软件触发
USB3020_TRIGSRC_DTR_DA	0x01	DTR 硬件数字触发

bSingleOut DA 是否单点输出，=TRUE 表示单点输出，=FALSE 表示连续输出。
Frequency DA 的采样频率，其取值范围为[0.01, 1000000Hz]。
TriggerDir DA 触发方向。它的选项值如下表：

常量名	常量值	功能定义
USB3020_TRIGDIR_NEGATIVE	0x00	负向触发（低脉冲/下降沿触发）
USB3020_TRIGDIR_POSITIVE	0x01	正向触发（高脉冲/上升沿触发）
USB3020_TRIGDIR_POSIT_NEGAT	0x02	正负向触发(高/低脉冲或上升/下降沿触发)

注明：USB3020_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 DA 时钟源选择。它的选项值如下表：

常量名	常量值	功能定义
USB3020_CLOCKSRC_IN	0x00	内部时钟
USB3020_CLOCKSRC_OUT	0x01	外部时钟

各段的头信息在 RAM 中的地址空间：

常量名	常量值	功能定义
USB3020_SEGMENT_HEADER_SIZE	0x0C	段头信息的长度

第二节、DA 状态参数结构（USB3020_STATUS_DA）

Visual C++：

```
typedef struct _USB3020_STATUS_DA
  {
    LONG bTrigFlag;     // 触发标志是否有效，=TRUE 表示触点标有效， = FALSE 表示无效（即触发点未到）
    LONG bConverting;   // DA 是否正在转换， =TRUE:表示正在转换， = FALS 表示转换完成
    LONG nCurSegNum;    // 可读取的 RAM 段号，取值为[0, SegmentCount-1], (注 SegmentCount 为
                        InitDeviceDA 函数的参数)
    LONG nCurSegAddr;   // 可读取的 RAM 段地址
    LONG nCurLoopCount; // 当前总循环次数
    LONG nCurSegLoopCount; // 当前段循环次数
  } USB3020_STATUS_DA, *PUSB3020_STATUS_DA;
```

Visual Basic：

```
Type USB3020_STATUS_DA
  bTrigFlag As Long              ' 触发标志是否有效，=TRUE 表示触点标有效， = FALSE 表示无效（即触发
```

点未到）

```
        bConverting As Long          ' DA 是否正在转换， =TRUE:表示正在转换， = FALS 表示转换完成
        nCurSegNum  As  Long          ' 可读取的 RAM 段号，取值为[0, SegmentCount-1], (注 SegmentCount 为
InitDeviceDA 函数的参数)
        nCurSegAddr As Long           ' 可读取的 RAM 段地址
        nCurLoopCount As Long         ' 当前总循环次数
        nCurSegLoopCount As Long      ' 当前段循环次数
End Type
```

LabVIEW:

请参考相关演示程序。

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

bEnable DA 使能启动标志，=TRUE 表示 DA 已被使能，=FALSE 表示 DA 被禁止。

bTrigFlag 触发标志是否有效，=TRUE 表示触点标志有效，=FALSE 表示无效（即触发点未到）。

bConverting DA 是否正在转换，=TRUE 表示正在转换，=FALSE 表示转换完成。

nCurSegNum 可读取的 RAM 段号。

nCurSegAddr 可读取的 RAM 段地址。

nCurLoopCount 当前总循环次数。

nCurSegLoopCount 当前段循环次数。

相关函数： CreateDevice GetDevStatusProDA ReleaseDevice

第五章 数据格式转换与排列规则

第一节、DA 电压值转换成 LSB 原码数据的换算方法

量程(伏)	计算机语言换算公式(标准 C 语法)	Lsb 取值范围
±5000mV	Lsb = Volt / (10000.00/65536) + 32768	[0，65535]
±10000mV	Lsb = Volt / (20000.00/65536) + 32768	[0，65535]

第二节、关于 DA 数据 DABuffer 缓冲区中的数据排放规则

由于各个通道的段信息与波形数据均共享一个板载物理 RAM，它们的排放顺序如图 5.1，但各个通道所占 RAM 空间大小均可以通过函数 SetDevRamSizeDA()调整，系统默认值为四个通道均分整个 RAM 空间，即默认每通道 RAM 空间为 256K 点。从下图可以看出，各个通道所占 RAM 空间不一定相等，可大可小，只是四个通道的总空间不能大于板载物理 RAM 空间即可。

通道 0	通道 1	通道 2	通道 3
0 至(256K-1)	256 至(512K-1)	512 至(768K-1)	768K 至(1024K-1)

图 5.1

关于每个通道 RAM 空间的内部分配是这样的，其空间首部存放的是所有段的段信息数据，其后才是各个段的波形数据，再其后可能还有未用空间。假如有三个分段，如图 5.2：

段 0 信息	段 1 信息	段 2 信息	段 0 波形数据	段 1 波形数据	段 2 波形数据	未用空间

图 5.2

关于每个段的段信息包括的内容有：该段波形数据在 RAM 中的起始地址、终止地址、段循环次数，如表 5.2.1，注意其段起始地址和终止地址是由 USB3020_PARA_DA 中的 SegmentInfo 决定的。

表 5.2.1

板载 RAM 内存单元(16Bit)	各单元定义	有效位
0	段 0 波形数据起始地址低 12 位	D11:D0
1	段 0 波形数据起始地址高 8 位	D7:D0

2	段 0 波形数据终止地址低 12 位	D11:D0
3	段 0 波形数据终止地址高 8 位	D7:D0
4	段 0 循环次数低 12 位	D11:D0
5	段 0 循环次数高 8 位	D7:D0
6	段 1 波形数据起始地址低 12 位	D11:D0
7	段 1 波形数据起始地址高 8 位	D7:D0
8	段 1 波形数据终止地址低 12 位	D11:D0
9	段 1 波形数据终止地址高 8 位	D7:D0
10	段 1 循环次数低 12 位	D11:D0
11	段 1 循环次数高 8 位	D7:D0
:	:	:
段信息结束后便是波形数据	段信息结束后便是波形数据	D11:D0

第六章 上层用户函数接口应用实例

第一节、简易程序演示说明

一、怎样使用WriteDevicePro DA函数进行批量DA数据输出

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 USB3020.h 和 Sys.cpp)。

[程序]｜**[阿尔泰测控演示系统]**｜**[Microsoft Visual C++]**｜[简易代码演示]｜[DA 批量输出演示源程序]

其简易程序默认存放路径为：系统盘\USB\USB3020\SAMPLES\VC\SIMPLE\DA\BULK

其他语言的演示可以用上面类似的方法找到。

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 USB3020.h 和 DADoc.cpp)。

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

其默认存放路径为：系统盘\USB\ USB3020\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 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“USB3020_”）

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

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 USB3020_GetDevVersion Lib "USB3020_32" ( _
                                                    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