USB3100 开发指南

目　　录

第一章 功能概述与约定

第一节、驱动函数的跨产品设计

所谓跨产品设计，就是确保在不同产品间，凡是相同的功能部分要求其命名、定义、方式、方法、概念尽可能一致，尽可能保持兼容，甚至一样。如设备对象创建函数除了“产品系列”前缀不一样外，均命名为“DEV_Create”；模拟量输入功能的初始化函数均命名为“AI_InitParam”…，这样的设计方式意义重大。当用户使用了阿尔泰公司某一种产品后就可以很轻松地重用到第二种、第三种以及更多的产品中，让用户在产品升级换型中能极大地降低开发难度，加快开发进度，从而节约更多的开发成本，保护用户的前期投资。

第二节、驱动函数的跨功能设计

所谓跨功能设计，就是确保在产品的不同功能间，凡是相同的函数动作，就要求其命名、定义、方式、方法尽可能一致，尽可能保持兼容，甚至一样。如 AI 功能中初始化函数命名为：AI_InitParam()；在 CTR 功能中初始化函数则命名为 CTR_InitParam()；在 DIO 功能中初始化函数则命名为 DIO_InitParam()等。其他基本同理。这样的设计方式意义重大。当用户使用了产品某一种功能后所熟悉的方法、方式、概念等就可以很轻松地重用到第二种、第三种以及更多的功能中，让用户在扩展用户的应用设计，升级用户的产品线时能极大地降低开发难度，加快开发进度，从而节约更多的开发成本，保护用户的前期投资。

第三节、驱动函数的跨语言设计

阿尔泰为各种编程语言提供了统一的、标准的、共享的驱动函数接口，除了语法上有差异以外，关于函数名称、参数名称、参数数量、参数取值，常量名、常量取值等都是一致的。用户只要在一种语言下用阿尔泰公司的产品做过应用系统的开发，其间所掌握的方式、方法、概念等都可以很轻松使用在另外一种语言下，能帮助用户极大的提高开发效率，保护用户的前期投入。包括不同产品之间，阿尔泰也要力求做到：功能相同，函数就相同，命名就相同，定义就相同，方式方法更要相同。一个目的就是让用户使用阿尔泰的产品越多，用户的工作就会越轻松，阿尔泰的服务也就越来越好。

现阶段阿尔泰的驱动跨越的语言有：Microsoft Visual C++、Microsoft Visual Basic、Borland C++ Builder、Borland Delphi、NI LabVIEW、NI LabWindows/CVI。对于其他语言如 Microsoft Visual FoxPro、PowerBuilder、MatLab 等也能支持。只是阿尔泰会根据需要对某些编程序言接供直接支持，而不是对所有语言同时提供支持。阿尔泰会根据需要在以后的服务中逐步提供（当然有经验的用户也可以根据现有某种语言的函数申明制作其他语言的接口函数）。

第四节、关键字缩写命名约定

缩写	全称	定义	缩写	全称	定义
DEV/Dev	Device	设备	DIR/Dir	Direction	方向
AI	Analog Input	模拟量输入	CPLG	Coupling	耦合
AO	Analog Output	模拟量输出	ATR	Analog Trigger	模拟量触发
DI	Digital Input	数字量单向输入	DTR	Digital Trigger	数字量触发
DO	Digital Output	数字量单向输出	Cur	Current	当前的
DIO	Digital Input/Output	数字量双向输入输出	ID	Identifier	标识
CTR	Counter	计数器或定时器	Idx	Index	索引
PARAM/Param	Parameter	参数	DI	Differential	差分(接地方式)
TRIG/Trig	Trigger	触发	SE	Single end	单端(接地方式)
CLK	Clock	时钟	REG	Register	寄存器
GND	Ground	地	Sens	Sensitivity	灵敏度
AGND	Analog Ground	模拟地	Pt	Point	点
DGND	Digital Ground	数字地	Pts	Points	点数
Lgc	Logical	逻辑的	Chan/CH	Channel	通道号
Phys	Physical	物理的	AUX	Auxiliary	辅助
Pio	Program I/O	软件 IO 传输模式	Buf	Buffer	缓冲
Int	Interrupt	中断传输模式	En	Enable	允许或使能
Dma	Direct Memory Access	直接内存存取	SRC/Src	Source	源

SAMP/Samp	Sample	(传输方式) 采样			

第五节、数据类型

基本数据类型

类型名称	类型描述	数据范围	各编程语言支持类型		
			C/C++/CVI/C_Builder	Visual Basic	Pascal(Delphi)
I8	有符号 8 位整型数	-128 to 127	char	无此数据类型 用 Byte 代替	ShortInt
U8	无符号 8 位整型数	0 to 255	unsigned char	Byte	Byte
I16	有符号 16 位整型数	-32768 to +32767	short	Integer	SamllInt
U16	无符号 16 位整型数	0 to 65535	unsigned short	无此数据类型 用 Integer 代替	Word
I32	有符号 32 位整型数	-2147483648 to 2147483647	int	Long	LongInt
U32	无符号 32 位整型数	0 to 4294967295	unsigned int	无此数据类型 用 Long 代替	LongWord/ Cardinal
I64	有符号 64 位整型数	-9223372036854775808 to 9223372036854775807	__int64		Int64
U64	无符号 64 位整型数	0 to 18446744073709551615	unsigned __int64		无此数据类型 用 Int64 代替
F32	32 位单精度浮点数	-3.402823E38 to 3.402823E38	float	Single	Single
F64	64 位双精度浮点数	-1.797683134862315E308 to 1.797683134862315E309	double	Double	Double
F64L	64 位多精度浮点数	1.189731495357231765e+4932 to 3.362103143112093506 3e-4932	long double		Extnded

Visual C++扩展数据类型

Visual C++基本数据类型	Visual C++扩展数据类型	Visual C++扩展指针类型
char	CHAR	PCHAR
unsigned char	UCHAR/BYTE	PUCHAR/PBYTE
short	SHORT	PSHORT
unsigned short	WORD/USHORT	PUSHORT/PWORD
int	long/LONG/ INT/BOOL	PLONG/PINT/PBOOL
unsigned long	ULONG	PULONG
float	FLOAT	PFLOAT
double	无	无

布尔变量数据类型

编程语言类型	布尔变量命名	字节数
Visual C++	bool	1
	BOOL	4
Visual Basic	Boolean	2(-1=真；0=假)
C++Builder	BOOL	4

Delphi	Boolean, ByteBool	1
	WordBool	2
	BOOL, LongBool	4

第六节、特别约定

为简化文字，同时又为了体现阿尔泰公司所注重的标准化、重用化、人性化、可扩展化，尽可能的延伸用户的后续设计，尽可能的保护用户的前期投资，阿尔泰便将文档中的产品标识前缀名“USB3100_”省略掉，只保留其产品统一的功能群组名和动作名称，如“DEV_Create”、“AI_InitParam”等关键部分，尽可能让用户看到的就是用户最最关心的功能部分，且只要功能一样，那么其命名形式、参数形式、参数取值也尽可能一样。但在实际的头文件和代码中此前缀是不能省略掉的。

凡是以“b”为前缀的变量或参数，均表示布尔型（bool），其取值总是为 TRUE 或 FALSE(即 1 或 0)；

凡是以“n”为前缀的变量或参数，均表示整型(integer)，包括 8 位、16 位、32 位、64 位有符号数和无符号数。（由于整型变量最普通，因此若没有标注前缀的变量或参数，通常可以理解为整型）；

凡是以“f”为前缀的变量或参数，均表示浮点型(float/double)；

凡是变量或参数中带有“Buffer”或“Buf”等字样的，均表示为缓冲或数组或指针(指针必须指向有一定长度的连续内存空间)；

第二章 使用提要

第一节、驱动函数的导入方法

为了用户能在演示工程中或用户的开发工程中更明确的看出驱动头文件的信息，所有语言的驱动头文件都是以产品名称为基本名，以各种语言的相关特征为扩展名。如下表：

语言	函数接口头文件	函数接口导入库	默认所在安装位置
Microsoft Visual C++	USB3100.h	USB3100.lib	C:\ART\USB3100\Include
Microsoft Visual Basic	USB3100.bas	无	C:\ART\USB3100\Include
Borland C++ Builder	USB3100.h	USB3100.lib	C:\ART\USB3100\Include
Borland Delphi	USB3100.pas	无	C:\ART\USB3100\Include
NI LabVIEW	USB3100.vi	无	C:\ART\USB3100\Include
NI LabWindows/CVI	USB3100.h	USB3100.lib	C:\ART\USB3100\Include

注① USB3100.h 是产品的最基础的头文件，强烈建议用户关注和使用该头文件中的函数接口以实现 AI、CTR、DIO 等功能。

注② USB3100RSV.h 是产品的保留头文件，“RSV”即是英文单词“Reserved”的缩写，意指“保留”的意思。这个头文件中提供的函数接口相对于 USB3100.h 来讲，完全属于辅助性、补充性的。对大多数用户来说可能是用不着的。比如对端口地址或寄存器的直接访问等操作，还有在某些编程语言下没有提供的特殊功能(如微秒级，纳秒级延时操作)，那么这个保留的头文件中的函数便提供了适当支持。因此这个头文件对用户提供的应该是一种超值的，额外的服务，而不一定是用户必须的服务。但为了凸现 USB3100.h 中关键函数的基础功能和保证用户在使用主要函数接口的简单易用性，阿尔泰不对保留头文件（RSV）中的函数作专门的文字型说明和售后的技术支持。

第二节、如何进行产品二次发布

如果用户使用阿尔泰公司的某款产品已做好了应用系统的开发，准备向市场发布，那么用户需要做的部分工作有：

一、将 USB3100.dll 从 Windows\System32 中复制到用户的安装包中；

二、将 USB3100.inf、USB3100.sys 从安装光盘相应产品文件夹下的 Driver 中复制到用户的安装盘中；

第三节、如何管理设备

由于阿尔泰的设备驱动程序采用是面向对象编程技术，即每一个设备就是一个对象，通过调用 DEV_Create() 函数可以创建无限多个设备对象的实例，并返回与设备实例关联的对象句柄 hDevice。有了这个句柄，用户就拥有了对该设备开放功能的所有控制权。然后将此句柄作为参数传递给相应的接口函数，如 AI_InitParam()可以使用 hDevice 句柄初始化 AI 工作参数，DIO_ReadPort()函数可用实现数字量的端口数据的读取等。最后可以通过 DEV_Release()函数将 hDevice 释放掉。

第四节、如何实现 AI 实时连续采样

实时连续采样，就是在一次启动后，AI 按照设定的采样速率，通道扫描模式进行长时间的，无限点的连续不间断采样。即在采样过程中可以实时读取采样到的连续数据。

AI 连续采样流程（查询同步）

(1) DEV_Create() 创建设备句柄
(2) AI_InitParam() 初始化 AI, 注意参数 nSampleMode=1, 参数 hEvent=NULL
(3) AI_Start() 启动 AI 采集
(4) AI_GetStatus() 查询到 nReadableSegments 大于 0，即刻进入到第 6 步开始读取数据段
(5) AI_ContReadChannelsV() 或者 AI_ContReadChannels() 读取 AI 数据段，若返回值大于 0，再紧接着读取
(6) AI_Stop() 停止 AI 采集
(7) AI_Release() 释放 AI 资源
(8) DEV_Release() 释放设备句柄

若每次采集参数相同的情况下，可以在 5、6 步间循环进行，即可实现高速连续不间断采样；若是在参数不变的情况下需要捕获新的触发时，可以在 4、5、6、7 之间循环进行；若每次采集参数有所不同，则可以在 2、3、4、5、6、7、8 步间重复进行。请参考看下面流程图 4.1。

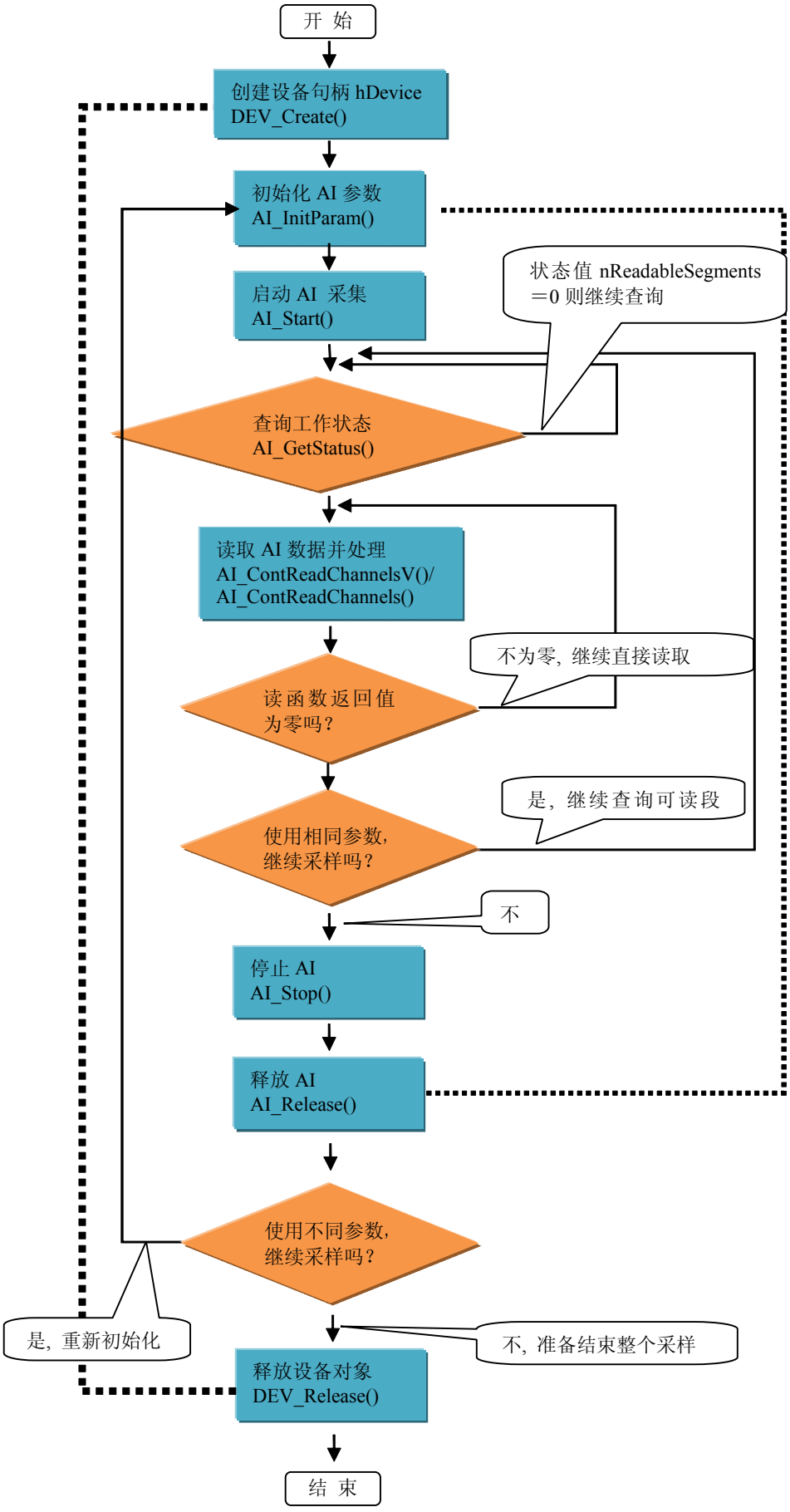


图 4.1 AI 实时连续采样(查询同步)流程图例

AI 连续采样流程（事件同步）

(1) DEV_Create()　创建设备句柄
(2) EVENT_Create()　创建事件句柄
(3) AI_InitParam()　初始化 AI, 注意参数 nSampleMode=1, 传递事件句柄 hEvent
(4) AI_Start()　启动 AI 采集
(5) WaitForSingleObject()　调用 WIN32 API 函数跟踪 hEvent 事件，等待中进入睡眠，有可读数据段时被唤醒
(6) AI_ContReadChannelsV()或者 AI_ContReadChannels()读取 AI 数据段，若返回值大于 0，再紧接着读取
(7) AI_Stop()　停止 AI 采集
(8) AI_Release()　释放 AI 资源
(9) EVENT_Release()　释放事件句柄
(10) _DEV_Release()释放设备句柄

若每次采集参数相同的情况下，可以在 5、6 步间循环进行，即可实现高速连续不间断采样；若是在参数不变的情况下需要捕获新的触发时，可以在 4、5、6、7 之间循环进行；若每次采集参数有所不同，则可以在 3、4、5、6、7、8 步间重复进行。请参考看下面流程图 4.2。

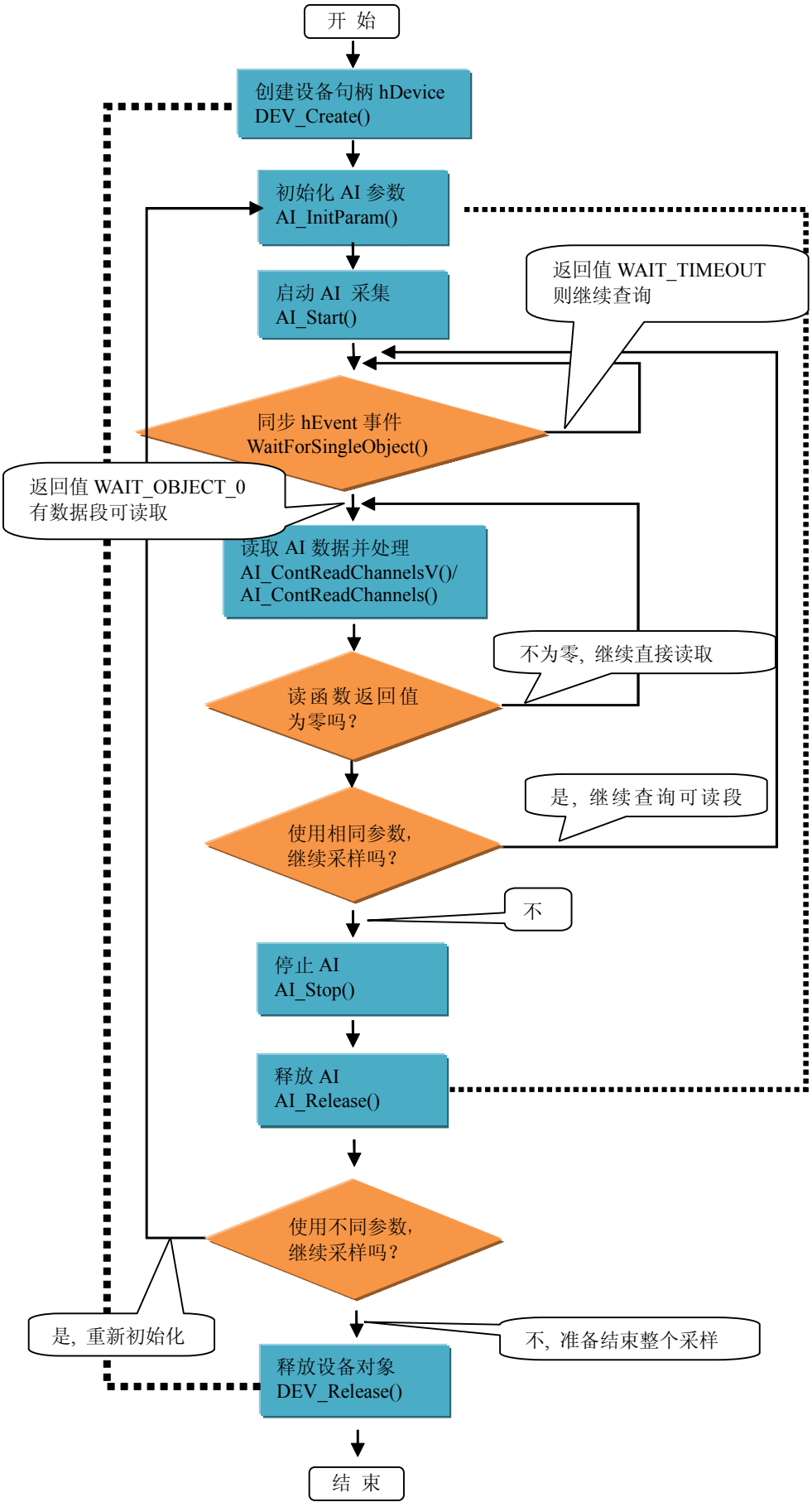


图 4.2 AI 实时连续采样(事件同步)流程图例

注意：图中虚线表示对称关系。比如较粗的虚线表示 DEV_Create()和 DEV_Release()两个函数的对称关系是：最初执行一次 DEV_Create()，在结束时就须执行一次 DEV_Release() (但并不是说只有 DEV_Release()后才能再次 DEV_Create()，因为阿尔泰的驱动程序是可以重入的)。而较细的虚线则表示 AI_InitParam()和 AI_Release()两个函数的对称关系（即只有在 AI_Release()之后才能再次 AI_InitParam(), 切记！)。

第五节、如何实现 AI 实时单点采样

实时单点采样，就是在一次启动后，用户每次发出读命令 AI_OneReadChannelsV()或 AI_OneReadChannels()时 AI 以设备最快的采样速率，和用户设定的通道扫描模式进行各采样通道单个点的采样，然后以最快的速度返回数据。具体步骤如下：

(1) DEV_Create() 创建设备句柄
(2) AI_InitParam() 初始化 AI, 参数 nSampleMode=0
(3) AI_Start() 启动 AI 采集
(4) AI_OneReadChannelsV()或者 AI_OneReadChannels() 读取 AI 各通道单点数据
(5) AI_Stop() 停止 AI 采集
(6) AI_Release() 释放 AI 资源
(7) DEV_Release() 释放设备句柄

若每次采集参数相同的情况下，可以在第 4 步处循环进行，即可实现单点实时循环采样；若每次采集参数有所不同，则可以在 2、3、4、5、6、7 步间重复进行。

具体执行流程请看下面的图 5.1。

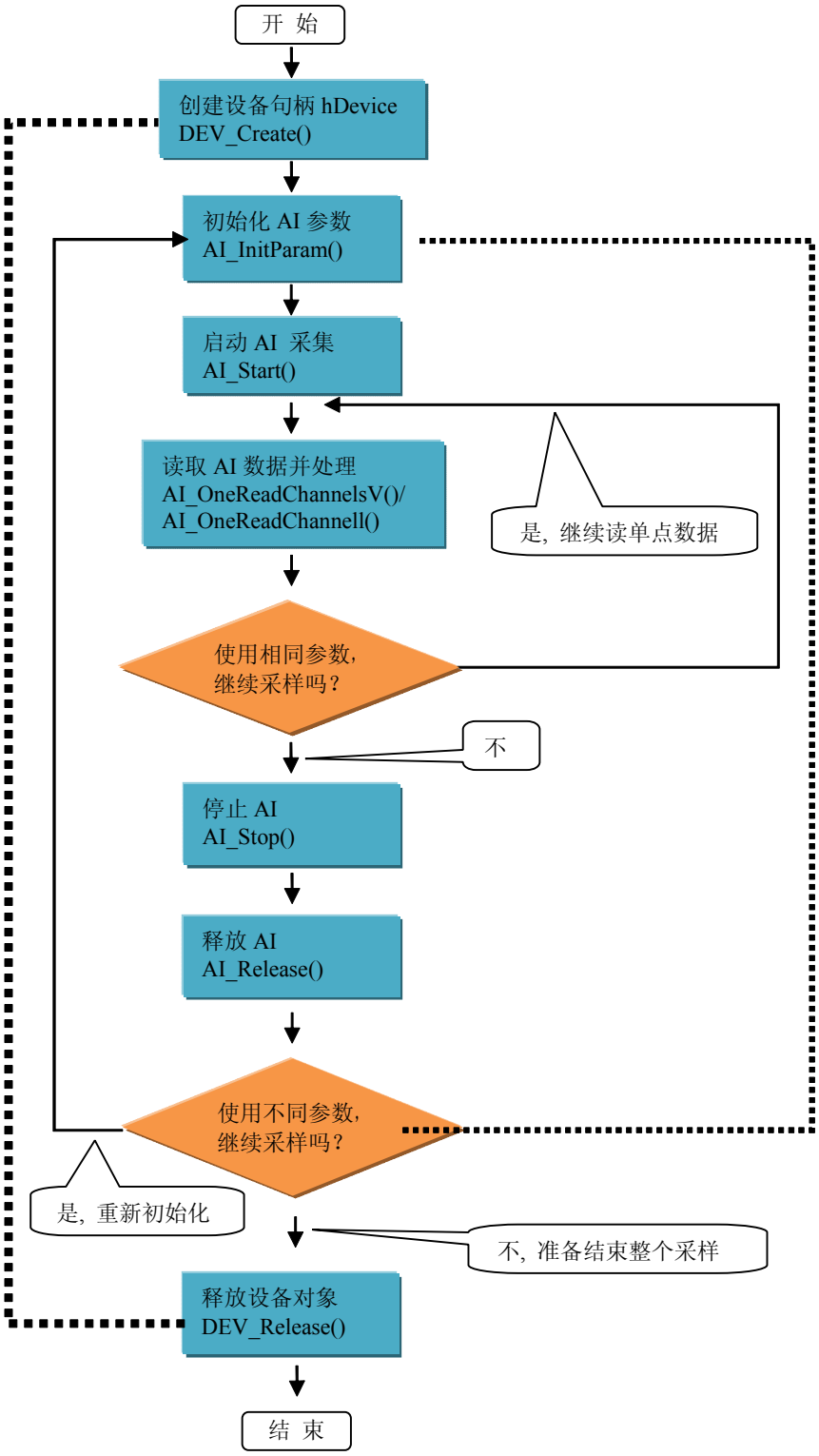


图 5.1 AI 实时单点采样流程图例

注意：图中虚线表示对称关系。比如较粗的虚线表示 DEV_Create()和 DEV_Release()两个函数的对称关系是：最初执行一次 DEV_Create()，在结束时就须执行一次 DEV_Release() (但并不是说只有后 DEV_Release()才能再次 DEV_Create()，因为阿尔泰的驱动程序是可以重入的)。而较细的虚线则表示 AI_InitParam()和 AI_Release()两个函数的对称关系（即只有在 AI_Release()之后才能再次 AI_InitParam()，切记！)。

第六节、如何实现 AI 实时单点简单采样

当用户调用 DEV_Create()函数创建了 hDevice 设备对象句柄后，便可直接调用 AI_SOneReadChannelsV()函数读取所有通道的单点电压数据。具体步骤：

(1) DEV_Create()　创建设备句柄
(2) AI_SOneReadChannels() 读取 AI 各通道单点数据
(3) DEV_Release()　释放设备句柄

若每次采集参数相同的情况下，可以在第 2 步处循环进行，

流程请看下面的图 6.1。

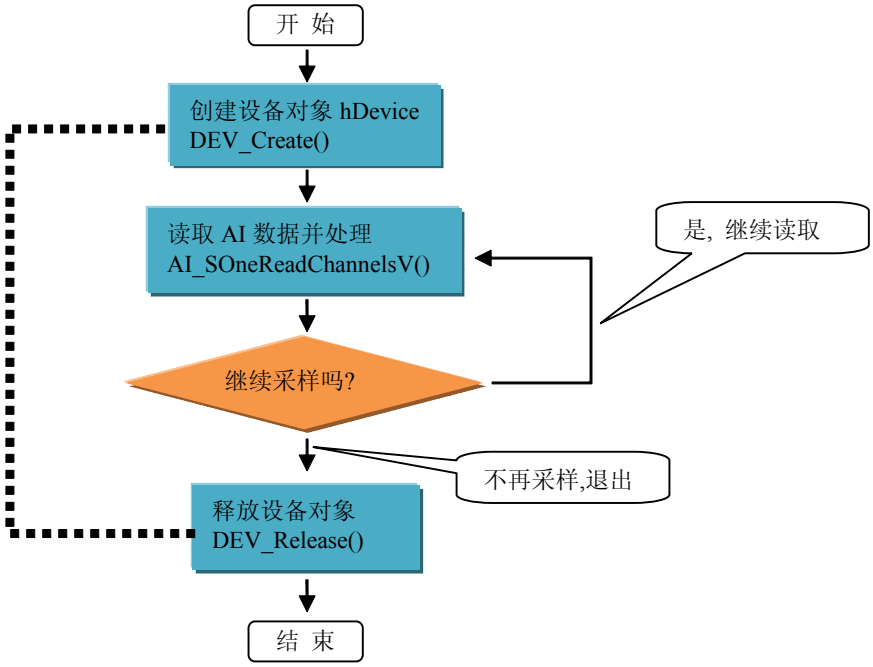


图 6.1 AI 实时单点简单采样流程图例

本方法相当于在 AI_OneReadChannels()单点读取函数的基础上再进行了一次简化封装，完整集成了初始化、启动、强制触发、状态同步查询、读取数据、停止等基本步骤。为用户需要简单应用时提供了一种最简单的方法。要是用户只想知道其外接信号的电压值，而无须关心采样范围，采样速率，通道组等参数配置，仅仅是知道外接信号的大体状态，就可以用此方法。注意：AI_SOneReadChannelsV()函数名中的 " S " 表示的就是简单(Simple)的意思。提示：在用户正在进行连续采样时，则不能调用 AI_SOneReadChannelsV()进行简单采样。

第七节、如何实现数字量的输入输出

当用户调用 DEV_Create()函数创建了 hDevice 设备对象句柄后，可调用 DIO_ReadPort()函数实现数字量的端口输入操作，调用 DIO_ReadLine()实现数字量的线输入操作，可调用 DIO_WritePort()函数实现数字量的端口输出操作，调用 DIO_WriteLine()实现数字量的线输出操作，

第八节、哪些函数对用户是必须的

首先，不管用户购买的是什么产品，其设备对象管理函数(以“DEV”为关键字段)对用户都是必须的，特别是 DEV_Create()、DEV_Release()两个函数则是重中之重。因为对于所有的设备访问都要有这两个函数的帮助。

如果用户购买的是多功能的设备，如集 AI、AO、DI、DO、CTR 为一身的产品，那么用户应该根据用户的功能需求有选择的理解和使用阿尔泰为用户提供的接口函数。由于阿尔泰的若干接口函数均以 AI、AO、DI、DO、CTR…等进行了合理的功能分组的，因此极大的缩小了用户的理解范围，很容易帮助用户对特定功能的定位与实现。

第三章 主要功能组函数介绍

由于现在两个接口头文件 USB3100.h 和 USB3100RSV.h，而这两个文件则有不同的功能使命。USB3100.h 是产

品的基础功能头文件，它是经过高度简化包装，力求展现产品功能，而不展现产品技术细节的头文件，意在让用户能通过简单的，易用的方式实现设备的主要功能。而 USB3100RSV.h 仅是一种补充式的，对极个别特殊需要的一种额外的，超值的服务，如实现对设备 I/O 式的访问，超大文件的简单快速读写访问，微秒级延时等。所以阿尔泰强制建议用户全力关注 USB3100.h 中的函数说明，而本开发指南也仅就 USB3100.h 中的接口函数加以说明。其 USB3100RSV.h 的接口函数只在其头文件中提供简单注释（而且不提供额外技术支持）

第一节、DEV 设备对象管理函数原型说明

DEV_Create()

函数原型：

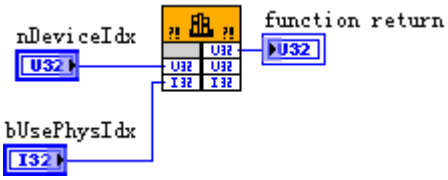
Visual C++ / C++Builder / LabWindows/CVI:

```
HANDLE DEV_Create(ULONG nDeviceIdx, BOOL bUsePhysIdx)
```

Visual Basic:

```
Declare Function DEV_Create Lib "USB3100" (ByVal nDeviceIdx As Long, _
                                           ByVal bUsePhysIdx As Long) As long
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：创建设备对象(Create device object)，并返回其设备对象句柄 hDevice。只有成功获取 hDevice，用户才能顺利调用其它相关的接口函数以实现对设备的控制。

参数：

nDeviceIdx 入口参数，设备序号(Device Index)。设备序号有两种：逻辑序号(Logical Index)和物理序号(Physical Index)。逻辑序号的定义是：当向同一台计算机系统中加入若干张相同类型的 USB 卡时，阿尔泰的驱动程序将自动以逻辑号来标识和管理具体的某张卡。比如某台计算机系统中插入该卡共四张，则操作系统依次扫描到的第一张卡时，则驱动程序将以逻辑号“0”来确认和管理这张卡，再扫描到第二张卡时，则驱动程序将以逻辑号“1”来确认和管理这张卡，以此类推。所以当用户要创建设备句柄管理和操作第一个设备时，nDeviceIdx 应置 0，第二个应置 1，以此类推。默认值为 0。该参数之所以称为设备逻辑号，是因为每个设备的逻辑号是不能事先由用户硬性决定的，而是由操作系统加载设备时，依据其接口号由小到大而确定的或者由插入 USB 的顺序决定的。而设备物理号则由可以用户事先对硬件进行配置决定的号，这个号是固定的，跟插入 USB 的顺序没有关系。究竟使用哪一种设备号，由参数 bUsePhysIdx 决定。当使用物理序号时，其取值范围为[0, 255]。

bUserPhysIdx 入口参数，是否使用物理序号，=FALSE(0)：不使用物理序号而使用逻辑序号； =TRUE(1)：使用物理序号。所有设备均支持逻辑号，但不一定支持物理号，本设备支持。

返回值：如果执行成功，则返回设备对象句柄；如果执行失败，则返回错误码 INVALID_HANDLE_VALUE(或 -1), 用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() DEV_GetCount() DEV_GetCurrentIdx() DEV_Release()

DEV_GetCount()

函数原型：

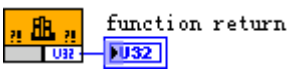
Visual C++ / C++Builder / LabWindows/CVI:

```
int DEV_GetCount(void);
```

Visual Basic:

```
Declare Function DEV_GetCount Lib "USB3100" () As Long
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：取得该设备在系统中的总数量(Get device count)。

参数：无

返回值：如果有设备存在，则返回实际的设备数量，若该设备不存在，则返回 0 值，此则有两种可能的情况存

在：其一，设备根本不存在，致使驱动程序无法安装加载。其二、设备存在，但其驱动程序未正确安装。对于具体原因，用户可以在操作系统的“设备管理器”中查看是否有该设备的信息项。在返回 0 时，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() DEV_GetCount() DEV_GetCurrentIdx() DEV_Release()

DEV_GetCurrentIdx()

函数原型：

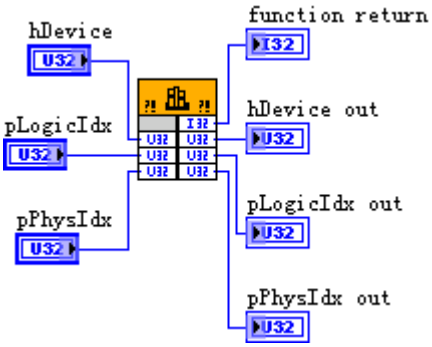
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DEV_GetCurrentIdx (HANDLE hDevice,
                        U32* pLgcIdx,
                        U32* pPhysIdx);
```

Visual Basic:

```
Declare Function DEV_GetCurrentIdx Lib "USB3100" (ByVal hDevice As Long, _
                                                  ByRef pLgcIdx As Long, _
                                                  ByRef pPhysIdx As Long) As Boolean
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：取得指定设备物理号和逻辑号(Get logical and physical index of the device)。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

pLgcIdx 出口参数，取得设备的逻辑索引号(Logical Index)，它的取值范围为[0, 255]。如果=NULL 则表示忽略此参数。

pPhysIdx 出口参数，取得设备的物理索引号(Physical Index)，它的取值范围为[0, 255]，它的具体值由 hDevice 指定的硬件决定。如果=NULL 则表示忽略此参数。

返回值：如果成功，则返回 TRUE， 否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() DEV_GetCount() DEV_GetCurrentIdx() DEV_Release()

DEV_Release()

函数原型：

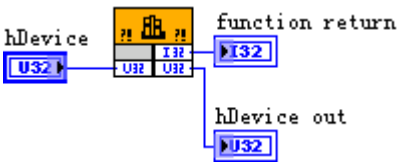
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DEV_Release(HANDLE hDevice)
```

Visual Basic:

```
Declare Function DEV_Release Lib "USB3100" (ByVal hDevice As Long ) As Boolean
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：释放设备对象（Release device object）， 包括释放所占用的系统资源。

参数：hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

返回值：若成功，则返回 TRUE， 否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Release()

应注意的是，调用 DEV_Create()创建设备对象句柄并使用完后应及时调用 DEV_Release()函数以释放由 DEV_Create()占用的某些系统资源。但并不是说只有在 DEV_Release()之后才能再次 DEV_Create()，而是可以多次创建设备对象。这个设计的目的就是为了解决驱动程序可重入的问题，为多功能、多设备、多线程、多进程间的任意组合访问和功能交叉访问提供了必要条件，但创建多少个就必须得释放多少个。

第二节、AI 模拟量输入函数原型说明

AI_InitParam()

函数原型：

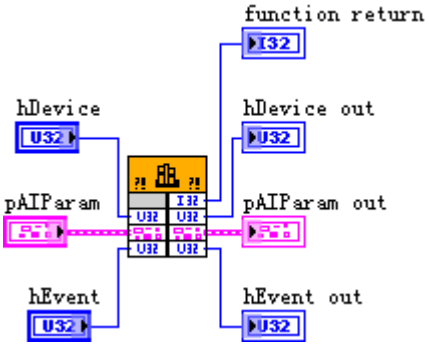
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AI_InitParam (HANDLE hDevice, PAI_PARAM pAIParam, HANDLE hEvent)

Visual Basic:

```
Declare Function AI_InitParam Lib "USB3100" (ByVal hDevice As Long, _
                                             ByRef pAIParam As AI_PARAM;
                                             ByVal hEvent As Long) As Boolean
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：初始化 AI 工作参数(Initialize work parameter for analog input)，为设备操作就绪有关状态，如预置 AI 采样率、各通道模拟量采样范围等。但它并不启动 AI 设备，若要启动 AI 设备，须在成功调用此函数之后再调用 AI_Start()函数。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

pAIParam 入口参数，AI 工作参数结构体指针，它决定了 AI 工作时的各种状态及参数，如采样率等。关于其具体定义及说明请参考《第四章 各种结构体描述》\《第一节、AI_PARAM（AI 工作参数结构）》。

hEvent 入口参数，事件句柄，该事件属性为自动发信号，初始状态为不发信号，建议由 EVENT_Create()创建。如果该参数=NULL，则视为用户不需要驱动程序触发任何事件。该事件句柄的作用是：当设备中有可读数据段（由 nPointPerChan 指定的段长）时则会触发此事件。用户可调用 WIN32 API 函数 WaitForSingleObject()来跟踪该事件。当事件未发生时，调用 WaitForSingleObject()函数的采集线程会自动阻塞(等待)状态(此时调用线程不会消耗 CPU 时间)，当事件发生时，则意味着设备中至少有一个可读数据段，则采集线程立即进入运行状态，并迅速调用 AI_ContReadChannelsV()或 AI_ContReadChannels()循环读取设备中的数据段，直至 nReadableSegments=0。如果简单循环调用 AI_GetStatus()查询 AI 的各种状态以同步数据段读操作，则会在等待中消耗许多 CPU 时间，而影响系统的整体性能。如果采用事件同步相结合的办法，则会节省大量的 CPU 时间。因为在调用 WaitForSingleObject()时，若事件未被触发，则当前线程会进入阻塞(等待)状态，而不消耗 CPU 时间，而事件一旦触发，则当前线程立即进入运行状态。总之它可以在一定程度上提升系统的整体性能，让应用程序有更多的 CPU 时间去处理采样数据或其他任务。当不再需要该事件时，建议调用 EVENT_Release()以释放事件对象。

返回值：如果初始化 AI 工作参数成功，则返回 TRUE， 否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	AI_InitParam()	AI_Start()
AI_GetStatus()	AI_SetForceTrig()	AI_ContReadChannelsV()
AI_ContReadChannels()	AI_OneReadChannelsV()	AI_OneReadChannels()

AI_SOneReadChannelsV() AI_Stop() AI_Release()
DEV_Release()

AI_Start()

函数原型：

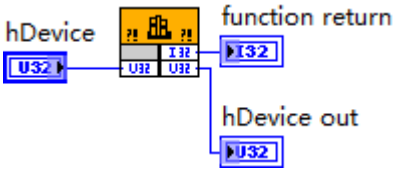
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AI_Start (HANDLE hDevice)

Visual Basic:

Declare Function AI_Start Lib "USB3100" (ByVal hDevice As Long) As Boolean

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：启动 AI 采集(Start sample for analog input)，它必须在成功调用 AI_InitParam()函数后才能调用此函数，调用该函数后 AI 立即准备就绪，但 AI 实际是否进入采样状态，要进一步依赖于触发事件的产生。

参数：**hDevice** 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

返回值：如果调用成功，则返回 TRUE，AI 立刻被启动， 否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AI_InitParam() AI_Start()
AI_GetStatus() AI_SetForceTrig() AI_ContReadChannelsV()
AI_ContReadChannels() AI_OneReadChannelsV() AI_OneReadChannels()
AI_SOneReadChannelsV() AI_Stop() AI_Release()
DEV_Release()

AI_SetForceTrig()

函数原型：

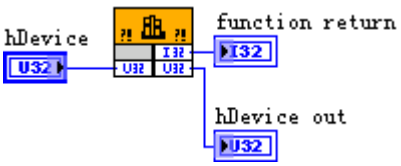
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AI_SetForceTrig (HANDLE hDevice)

Visual Basic:

Declare Function AI_SetForceTrig Lib "USB3100" (ByVal hDevice As Long) As Boolean

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：产生强制触发事件(Set force trigger event)。在启动 AI 采集后，来自外部的触发事件可能一直无法产生，用户也可能不知道外部发生了什么，但想马上让设备进入采集状态以便看到具体的信号情况，那么用户可以调用此函数以软件方式强制设备产生一个触发事件使硬件被正常触发采样一次。该方式也叫软件触发或手动触发或内触发。

参数：**hDevice** 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

返回值：如果调用成功，则返回 TRUE，即 AI 立刻被触发采样一次， 否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AI_InitParam() AI_Start()
AI_GetStatus() AI_SetForceTrig() AI_ContReadChannelsV()
AI_ContReadChannels() AI_OneReadChannelsV() AI_OneReadChannels()
AI_SOneReadChannelsV() AI_Stop() AI_Release()
DEV_Release()

AI_GetStatus()

函数原型：

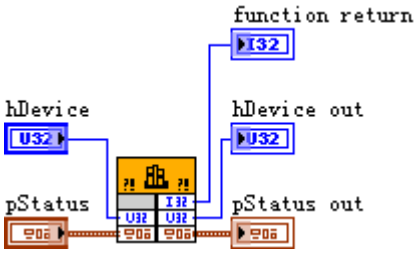
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AI_GetStatus (HANDLE hDevice, PAI_STATUS pStatus)

Visual Basic:

Declare Function AI_GetStatus Lib "USB3100" (ByVal hDevice As Long, _
ByRef pStatus As AI_STATUS) As Boolean

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：取得 AI 的各种状态(Get status for analog input)。一旦用户调用 AI_Start()函数后，应立即调用此函数查询 AI 状态去同步采样数据的读操作。nReadableSegments>0 时，表示采集任务中至少有 1 个数据段可供读取，用户应立即调用 AI_ContReadChannels() 或 AI_ContReadChannelsV() 函数循环读取若干可读段数据，直到 nReadableSegments=0。如果在启动采集后，设备迟迟不能被触发采样，用户可以根据需要调用 AI_SetForceTrig() 函数以强制触发设备采样，便可以很快得到有效的可读采样数据段。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

pStatus 出口参数，设备状态参数结构体，它返回设备当前的各种状态，如是否完成采样、是否已被触发等信息。关于具体状态信息请参考《第四章 各种结构体描述》\《第二节、AI_STATUS（AI 状态信息结构）》。

返回值：若成功获取 AI 状态，则返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError() 捕获错误码以确定具体原因。

相关函数：

DEV_Create()	AI_InitParam()	AI_Start()
AI_GetStatus()	AI_SetForceTrig()	AI_ContReadChannelsV()
AI_ContReadChannels()	AI_OneReadChannelsV()	AI_OneReadChannels()
AI_SOneReadChannelsV()	AI_Stop()	AI_Release()
DEV_Release()		

AI_ClearBuffer()

函数原型：

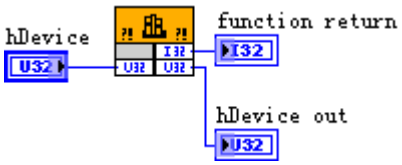
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AI_ClearBuffer (HANDLE hDevice)

Visual Basic:

Declare Function AI_ClearBuffer Lib "USB3100" (ByVal hDevice As Long) As Boolean

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：清除缓冲区，将可读段数和当前段读指针强行复位至 0，以便用户能以最短时间读取最新的采样段数据。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

返回值：若成功获取 AI 状态，则返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError() 捕获错误码以确定具体原因。

相关函数：

DEV_Create()	AI_InitParam()	AI_Start()
AI_GetStatus()	AI_SetForceTrig()	AI_ContReadChannelsV()
AI_ContReadChannels()	AI_OneReadChannelsV()	AI_OneReadChannels()

AI_SOneReadChannelsV()　AI_Stop()　AI_Release()
DEV_Release()

AI_ContReadChannelsV()

函数原型:

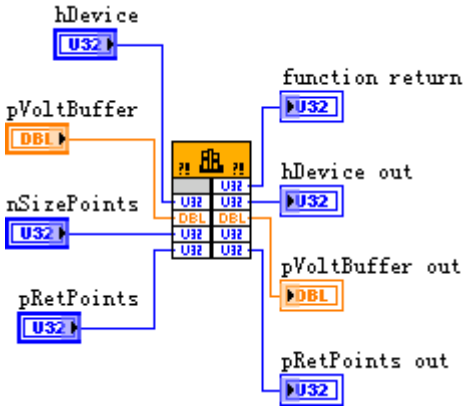
Visual C++ / C++Builder / LabWindows/CVI:

```
LONG AI_ContReadChannelsV(HANDLE hDevice,
                          F64* pVoltBuffer,
                          U32 nSizePoints,
                          U32* pRetPoints);
```

Visual Basic:

```
Declare Function AI_ContReadChannelsV Lib "USB3100" ( ByVal hDevice As Long, _
                                                      ByRef pVoltBuffer As Double, _
                                                      ByVal nSizePoints As Long, _
                                                      ByRef pRetPoints As Long) As Long
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：在连续采样模式(nSampleMode=AI_SAMPLEMODE_CONTINUOUS)中读取电压数据。它是通过 AI_ContReadChannels()读取到原码数据后转换成相应的电压值返回给用户的。它不仅全部包括 AI_ContReadChannels()的所有功能，而且还为用户实现了较为复杂的原码数据到电压值数据的转换过程，省去了对采样范围、单双极性、分辨率、换算等复杂过程的理解与控制，帮助用户降低了开发的难度。

参数:

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

pVoltBuffer 出口参数，用户缓冲区，用于接受所有采样通道电压值数据，值区间由相应采样通道的选用采样决定，单位: 伏(V)，获得的电压值数据为双精度浮点型。各个采样通道的数据点是依次交替排列的。它被开辟的点空间不能小于 nSizePoints 参数值。

nSizePoints 入口参数，请求读入的数据点数。它指定该次从设备的当前可读数据段中读取的数据点数（单位: 点）。注意此参数的值如大于数据段长 nSegmentLen（即 nSegmentLen = nPointsPerChan*nSampChanCount），则只返回 nSegmentLen 长度的数据给用户，再如果是小于 nSegmentLen，则返回给用户的实际数据长度等于 nSizePoints(在连续采样过程中，如果要保持连续不丢点，则必须大于或等于 nSegmentLen，除非用户需要每次只处理可读段的前一部分数据)。当然此参数值也不能大于 nAIBuffer[]的缓冲区长度，所以为避免出错，用户在开辟缓冲区时应考虑到 nSegmentLen 的大小，所开辟的缓冲区不能小于 nSegmentLen。

pRetPoints 出口参数，返回该次实际读取的点数。详见 nSizePoints 的说明。注意每次都要检查 pRetPoints 的返回值，如果返回值等于 0，则要慎重处理，用户可以调用 Win32 API 函数 GetLastError()以取得进一步的错误码信息 nErrorCode，其定义如下表。

常量名	常量值	功能定义
ERROR_NO_READABLE_SEGMENTS	0xE0000000+1	无可读段（此信息可以忽略）
ERROR_SAMPLE_TASK_FAIL	0xE0000000+2	采样任务失败

返回值： 返回值代表设备中还有多少个可读数据段，如果不为 0，则用户可以接着循环调用该函数继续读取数据段，直到返回值为 0 止。

相关函数： DEV_Create() AI_InitParam() AI_Start()
AI_GetStatus() AI_SetForceTrig() AI_ContReadChannelsV()
AI_ContReadChannels() AI_OneReadChannelsV() AI_OneReadChannels()
AI_SOneReadChannelsV() AI_Stop() AI_Release()
DEV_Release()

AI_ContReadChannels()

函数原型:

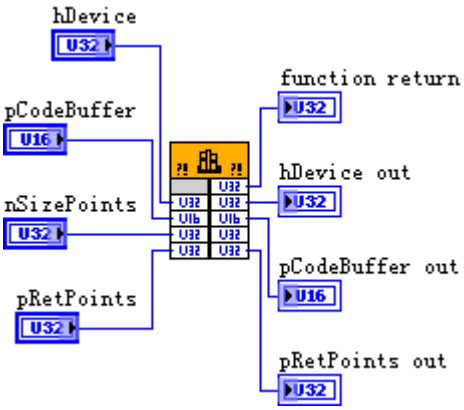
Visual C++ / C++Builder / LabWindows/CVI:

```
LONG AI_ContReadChannels (HANDLE hDevice,
                          U16* pCodeBuffer,
                          U32 nSizePoints，
                          U32* pRetPoints);
```

Visual Basic:

```
Declare Function AI_ContReadChannels Lib "USB3100" ( ByVal hDevice As Long, _
                                                     ByRef pCodeBuffer As Integer, _
                                                     ByVal nSizePoints As Long, _
                                                     ByRef pRetPoints As Long) As Long
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：在连续采样模式(nSampleMode=AI_SAMPLEMODE_CONTINUOUS)中读取原码数据。其基本功能同 AI_ContReadChannelsV()。只是此读数函数没有对采样数据进行任何处理，在需要高速连续存盘记录的过程中，为保证效率，建议用户优先使用这个函数。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

pCodeBuffer 出口参数，用户缓冲区，用于接受通道原码数据，值区间为 [0, 4095]。各个采样通道的数据点是依次交替排列的。如果用户需要将原码数据转换成电压数据请调用 AI_VoltScale()函数，它会按单个通道数据将原码转换为指定采样范围下的电压数据。它被开辟的点空间不能小于 nSizePoints 参数值。

nSizePoints 入口参数，请求读入的数据点数。它指定该次从设备的当前可读数据段中读取的数据点数（单位:点）。注意此参数的值如大于数据段长 nSegmentLen（即 nSegmentLen = nPointsPerChan*nSampChanCount），则只返回 nSegmentLen 长度的数据给用户，再如果是小于 nSegmentLen，则返回给用户的实际数据长度等于 nSizePoints(在连续采样过程中，如果要保持连续不丢点，则必须大于或等于 nSegmentLen，除非用户需要每次只处理可读段的前一部分数据)。当然此参数值也不能大于 nAIBuffer[]的缓冲区长度，所以为避免出错，用户在开辟缓冲区时应考虑到 nSegmentLen 的大小，所开辟的缓冲区不能小于 nSegmentLen。

pRetPoints 出口参数，返回该次实际读取的点数。详见 nSizePoints 的说明。注意每次都要检查 pRetPoints 的返回值，如果返回值等于 0，则要慎重处理，用户可以调用 Win32 API 函数 GetLastError()以取得进一步的错误码信息 nErrorCode，其定义如下表。

常量名	常量值	功能定义
ERROR_NO_READABLE_SEGMENTS	0xE0000000+1	无可读段（此信息可以忽略）
ERROR_SAMPLE_TASK_FAIL	0xE0000000+2	采样任务失败

返回值： 返回值代表设备中还有多少个可读数据段，如果不为 0，则用户可以接着循环调用该函数继续读取数据段，至到返回值为 0 止。

相关函数： DEV_Create() AI_InitParam() AI_Start() AI_GetStatus() AI_SetForceTrig() AI_ContReadChannelsV() AI_ContReadChannels() AI_OneReadChannelsV() AI_OneReadChannels() AI_SOneReadChannelsV() AI_Stop() AI_Release() DEV_Release()

关于实时连续采样调用流程的图例请参考《第二章 使用提要》中《 第四节、如何实现 AI 实时连续采样 》相关部分。

AI_OneReadChannelsV()

函数原型:

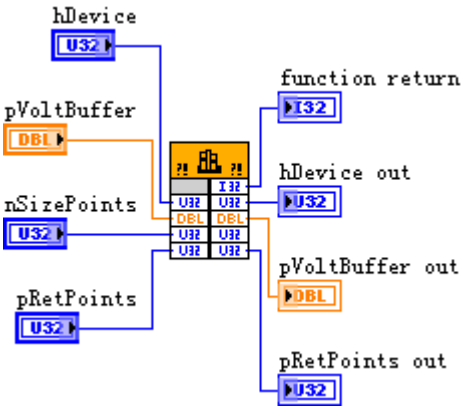
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_OneReadChannelsV(HANDLE hDevice,
                         U16* pCodeBuffer,
                         U32 nSizePoints,
                         U32* pRetPoints);
```

Visual Basic:

```
Declare Function AI_OneReadChannelsV Lib "USB3100" ( ByVal hDevice As Long, _
                                                     ByRef pCodeBuffer As Integer, _
                                                     ByVal nSizePoints As Long, _
                                                     ByRef pRetPoints As Long) As Boolean
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：在单点采样模式（nSampleMode= USB3100_AI_SAMPLEMODE_ONE_DEMAND）中实时读取各个采样通道单点电压数据。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

pVoltBuffer 出口参数，用户缓冲区，用于接受采样通道电压数据，单位：伏（V），值区间由所选采样范围决定。通道数据排列顺序按工作参数中的通道组设置顺序依次排放。

nSizePoints 入口参数，指定该次从设备中读取的数据长度（单位：点）。该参数的取值应不小于采样通道数，且最好等于采样通道总数，如果大于采样通道总数，则按实际采样通道总数返回实际的采样点数。另外此参数值也不能大于 pVoltBuffer[]的缓冲区长度，所以为避免出错，在条件允许的情况下，用户开辟的数据缓冲区 pVoltBuffer 应尽量大。

pRetPoints 出口参数，取得当前返回的数据点数，通常返回值等于实际采样通道数量。

返回值：如果调用成功，则返回 TRUE，若失败则返回 FALSE 值，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AI_InitParam() AI_Start() AI_GetStatus() AI_SetForceTrig() AI_ContReadChannelsV() AI_ContReadChannels() AI_OneReadChannelsV() AI_OneReadChannels() AI_SOneReadChannelsV() AI_Stop() AI_Release() DEV_Release()

AI_OneReadChannels()

函数原型:

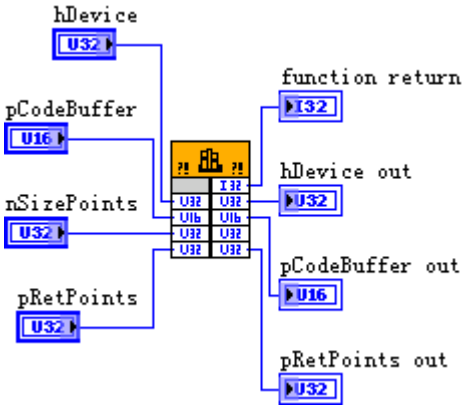
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_OneReadChannels(HANDLE hDevice,
                        U16* pCodeBuffer,
                        U32 nSizePoints,
                        U32* pRetPoints);
```

Visual Basic:

```
Declare Function AI_OneReadChannels Lib "USB3100" ( ByVal hDevice As Long, _
                                                    ByRef pCodeBuffer As Integer, _
                                                    ByVal nSizePoints As Long, _
                                                    ByRef pRetPoints As Long) As Boolean
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：在单点采样模式（nSampleMode= USB3100_AI_SAMPLEMODE_ONE_DEMAND）中实时读取所有采样通道单点原码数据。

参数:

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

pCodeBuffer 出口参数，用户缓冲区，用于返回读取的采样通道原码数据，值区间 [0, 4095]。通道数据排列顺序按工作参数中的通道组设置顺序依次排放。

nSizePoints 入口参数，指定该次从设备中读取的数据长度（单位：点）。该参数的取值应不小于采样通道数，且最好等于采样通道总数，如果大于采样通道总数，则按实际采样通道总数返回实际的采样点数。另外此参数值也不能大于 pCodeBuffer 的缓冲区长度，所以为避免出错，在条件允许的情况下，用户开辟的数据缓冲区 pVoltBuffer 应尽量大。

pRetPoints 出口参数，取得当前返回的数据点数，通常返回值等于实际采样通道数量。

返回值：如果调用成功，则返回 TRUE，若失败则返回 FALSE 值，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	AI_InitParam()	AI_Start()
AI_GetStatus()	AI_SetForceTrig()	AI_ContReadChannelsV()
AI_ContReadChannels()	AI_OneReadChannelsV()	AI_OneReadChannels()
AI_SOneReadChannelsV()	AI_Stop()	AI_Release()
DEV_Release()		

关于实时单点采样调用流程的图例请参考《第二章 使用提要》中《 第五节、如何实现 AI 实时单点采样 》相关部分。

AI_SOneReadChannelsV()

函数原型:

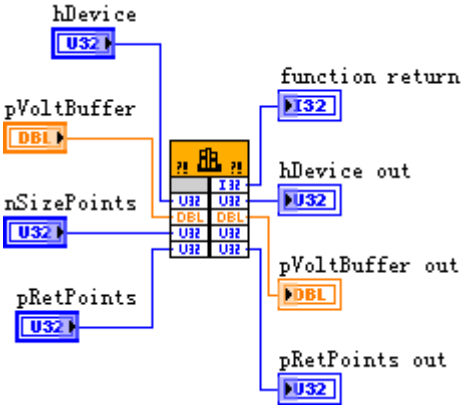
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_OneReadChannelsV(HANDLE hDevice,
                         F64* pVoltBuffer,
                         U32 nSizePoints,
                         U32* pRetPoints);
```

Visual Basic:

```
Declare Function AI_SOneReadChannelsV Lib "USB3100" ( ByVal hDevice As Long, _
                                                      ByRef pVoltBuffer As Integer, _
                                                      ByVal nSizePoints As Long, _
                                                      ByRef pRetPoints As Long) As Boolean
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：单点模式简易采样，无须用户设置任何 AI 工作参数，单独启动和查询有关状态，而且直接简单调用此函数就会实现所有通道，最高采样速率，最大采样范围，单端接地的数据采样。意在简化用户的操作。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

pVoltBuffer 出口参数，用户缓冲区，用于返回读取的采样通道电压数据，值区间[-10.0，10.0]。通道数据按 AI0－AI7 依次排放。

nSizePoints 入口参数，指定该次从设备中读取的数据长度（单位：点）。该参数的取值应不小于采样通道数 8，且最好等于采样通道总数 8，如果大于采样通道总数 8，则按实际采样通道总数返回实际的采样点数 8。另外此参数值也不能大于 pVoltBuffer 的缓冲区长度，所以为避免出错，在条件允许的情况下，用户开辟的数据缓冲区 pVoltBuffer 应尽量大。

pRetPoints 出口参数，取得当前返回的数据点数，通常返回值等于实际采样通道数量 8。

返回值：如果调用成功，则返回 TRUE，若失败则返回 FALSE 值，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	AI_InitParam()	AI_Start()
AI_GetStatus()	AI_SetForceTrig()	AI_ContReadChannelsV()
AI_ContReadChannels()	AI_OneReadChannelsV()	AI_OneReadChannels()
AI_SOneReadChannelsV()	AI_Stop()	AI_Release()
DEV_Release()		

关于单点最简单采样调用流程的图例请参考《第二章 使用提要》中《 第六节、如何实现 AI 实时单点简单采样 》相关部分。

AI_Stop()

函数原型：

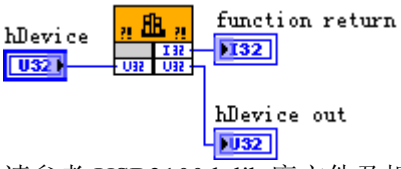
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_Stop (HANDLE hDevice)
```

Visual Basic:

```
Declare Function AI_Stop Lib "USB3100" (ByVal hDevice As Long)
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：停止 AI 采样(Stop Sample for analog input)。它必须在成功调用 AI_Start()函数后才能调用此函数。该函数除了停止 AI 采样外不改变设备的其他状态。

参数：**hDevice** 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

返回值：如果调用成功，则返回 TRUE，且 AI 立刻停止转换， 否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	AI_InitParam()	AI_Start()
AI_GetStatus()	AI_SetForceTrig()	AI_ContReadChannelsV()
AI_ContReadChannels()	AI_OneReadChannelsV()	AI_OneReadChannels()
AI_SOneReadChannelsV()	AI_Stop()	AI_Release()
DEV_Release()		

AI_Release()

函数原型：

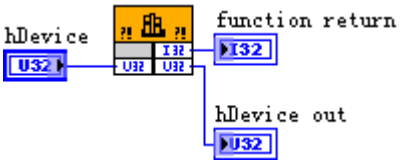
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_Release (HANDLE hDevice)
```

Visual Basic:

```
Declare Function AI_Release Lib "USB3100" (ByVal hDevice As Long)
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：释放 AI(Release AI)。它必须在重新调用 AI_InitParam() 函数之前被调用一次，即该函数必须和 AI_InitParam() 成对出现。注意此函数在内部首先执行 AI_Stop()函数停止 AI 采集后，才释放被占用的 AI 资源。

参数：**hDevice** 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

返回值：如果调用成功，则返回 TRUE，且 AI 资源被释放之后，则可以重新调用 AI_InitParam()函数初始化 AI 工作参数等。否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	AI_InitParam()	AI_Start()
AI_GetStatus()	AI_SetForceTrig()	AI_ContReadChannelsV()
AI_ContReadChannels()	AI_OneReadChannelsV()	AI_OneReadChannels()
AI_SOneReadChannelsV()	AI_Stop()	AI_Release()
DEV_Release()		

AI_VoltScale()

函数原型：

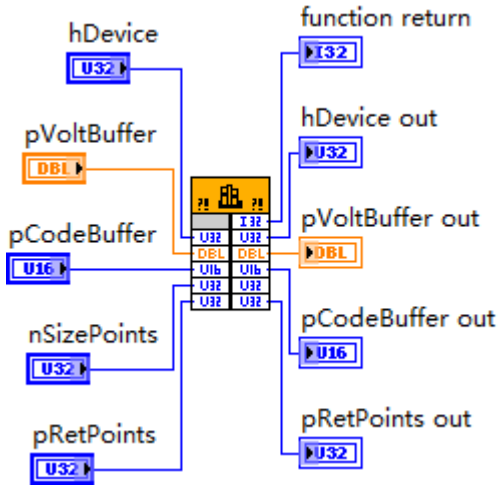
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_VoltScale(HANDLE hDevice,
                  F64* pVoltBuffer,
                  U16* pCodeBuffer,
                  U32 nSizePoints,
                  U32* pRetPoints);
```

Visual Basic:

```
Declare Function AI_VoltScale Lib "USB3100" (ByVal hDevice As Long, _
                                             ByRef pVoltBuffer As Double, _
                                             ByRef pCodeBuffer As Integer, _
                                             ByVal nSizePoints As Long, _
                                             ByRef pRetPoint As Long) As Boolean
```

LabVIEW



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：将原码数据转换为电压数据。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建。

pVoltBuffer 出口参数，用于返回量化后的电压数据，单位：伏（V），取值范围由 nSampleRange 参数选择而定。

pCodeBuffer 入口参数，用于传入待量化的原码数据，取值范围[0, 4095]

nSizePoints 入口参数，请求量化的数据点数

pRetPoints 出口参数，返回实际量化后数据点数

返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	AI_InitParam()	AI_Start()
AI_GetStatus()	AI_SetForceTrig()	AI_ContReadChannelsV()
AI_ContReadChannels()	AI_OneReadChannelsV()	AI_OneReadChannels()
AI_SOneReadChannelsV()	AI_Stop()	AI_Release()
DEV_Release()		

AI_GetMainInfo()

函数原型：

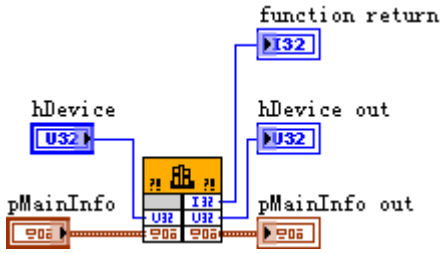
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_GetMainInfo (HANDLE hDevice,
                     PAI_MAIN_INFO pMainInfo)
```

Visual Basic:

```
Declare Function AI_GetMainInfo Lib "USB3100" ( ByVal hDevice As Long, _
                                                ByRef pMainInfo As AI_MAIN_INFO) As Boolean
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：取得 AI 功能的主要信息，如通道数、分辨率等。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建。

pMainInfo 出口参数，AI 主要信息结构指针，它负责返回 AI 的主要描述信息。关于 AI_MAIN_INFO 的详细介绍请参考 USB3100.h 或 USB3100.bas 或 USB3100.pas 函数原型定义头文件，也可参考本文《各种结构体描述》

关于该结构的有关说明。

返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AI_GetMainInfo() AI_GetRangeInfo()
AI_GetRateInfo() DEV_Release()

AI_GetRangeInfo()

函数原型：

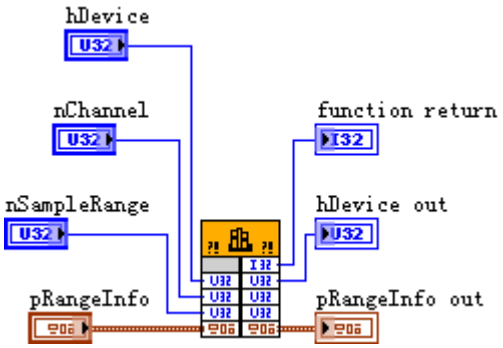
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_GetRangeInfo(HANDLE hDevice,
                     int nChannel,
                     int nSampleRange,
                     PAI_SAMP_RANGE_INFO pRangeInfo)
```

Visual Basic:

```
Declare Function AI_GetRangeInfo Lib "USB3100" (ByVal hDevice As Long, _
                                                ByVal nChannel As Long, _
                                                nSampleRange As Long, _
                                                ByRef pRangeInfo As AI_SAMP_RANGE_INFO) As Boolean
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：取得 AI 指定通道、指定采样范围档的上下限电压、幅度等信息。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建。

nChannel 入口参数，AI 通道号，由于本设备的所有通道共用一个采样范围挡，故取值恒等于 0。

nSampleRange 入口参数，AI 采样范围挡号，取值 0。

pRangeInfo 出口参数，AI 采样范围信息，它负责返回 AI 的采样范围的最大值、最小值、幅度、编码宽度等。详情请参考《 第四节、AI_SAMP_RANGE_INFO (AI 采样范围信息结构体) 》

返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AI_GetMainInfo() AI_GetRangeInfo()
AI_GetRateInfo() DEV_Release()

AI_GetRateInfo()

函数原型：

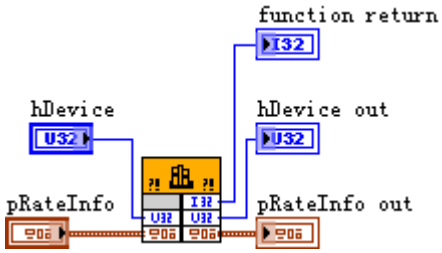
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_GetRateInfo(HANDLE hDevice, AI_SAMP_RATE_INFO pRateInfo);
```

Visual Basic:

```
Declare Function AI_GetRateInfo Lib "USB3100" (ByVal hDevice As Long, _
                                               ByRef pRateInfo As AI_SAMP_RATE_INFO) As Boolean
```

LabVIEW:

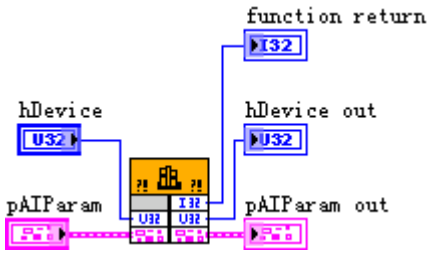


请参考 USB3100.lvlib 库文件及相关演示 vi

功能：获得采样速率信息。
参数：
hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建。
pRateInfo 出口参数，AI 采样速率信息，它负责返回 AI 的最大采样率，最小采样率等。详情请参考《 第五节、AI_SAMP_RATE_INFO (AI 采样率信息结构体) 》
返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。
相关函数： DEV_Create() AI_GetMainInfo() AI_GetRangeInfo()
AI_GetRateInfo() DEV_Release()

AI_VerifyParam()

函数原型：
Visual C++ / C++Builder / LabWindows/CVI:
BOOL AI_VerifyParam (HANDLE hDevice, PAI_PARAM pAIParam)
Visual Basic:
Declare Function AI_VerifyParam Lib "USB3100" (ByVal hDevice As Long, _
ByRef pAIParam As AI_PARAM) As Boolean
LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：校验 AI 工作参数(Verify work parameter for analog input)，这是一个辅助功能的函数，在初始化 AI 前，它是帮助用户事先校验 AI 参数的合法性，如果所有的参数合法，则返回 TRUE，如果有一个参数不合法则返回 FALSE。在校验过程中，遇到不合法的参数时，驱动程序会将相应的不合法参数及原因写入日志文件 USB3100.log，用户随时可以查看该文本文件。
参数：
hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。
pAIParam 入口和出口参数，AI 工作参数结构体指针，关于其具体定义及说明请参考《第四章 各种结构体描述》\《第一节、AI_PARAM（AI 工作参数结构）》。函数调用时，它传入用户要校验的各项参数，函数返回时，它将经过调整为合法的参数值返回给用户，以便后续初始化 AI 使用。
返回值：如果所有的参数均合法，则返回 TRUE， 如果有一个参数不合法，立即被调整为合法取值，并向日志文件 USB3100.log 载入不合法的参数名和原因，最后函数返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。
相关函数： DEV_Create() AI_InitParam() AI_Start()
AI_GetStatus() AI_SetForceTrig() AI_ContReadChannelsV()
AI_ContReadChannels() AI_OneReadChannelsV() AI_OneReadChannels()
AI_SOneReadChannelsV() AI_Stop() AI_Release()

AI_LoadParam()

函数原型：

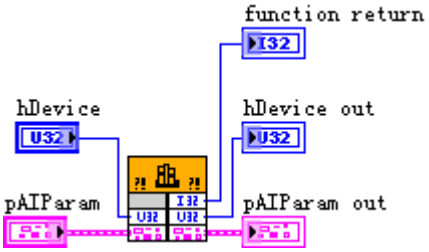
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_LoadParam (HANDLE hDevice,
                   PAI_PARAM pAIParam)
```

Visual Basic:

```
Declare Function AI_LoadParam Lib "USB3100" (ByVal hDevice As Long, _
                                             ByRef pAIParam As AI_PARAM) As Boolean
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：从系统注册表中读取设备的硬件参数(Load parameter from system registry for analog input)。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

pAIParam 出口参数，属于 PAI_PARAM 的结构指针类型，它负责返回 AI 工作参数值，关于结构指针类型 PAI_PARAM 请参考 USB3100.h 或 USB3100.bas 或 USB3100.pas 函数原型定义头文件，也可参考本文《第一节、AI_PARAM（AI 工作参数结构体）》

返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AI_InitParam() AI_LoadParam()
AI_SaveParam() AI_ResetParam() DEV_Release()

AI_SaveParam()

函数原型：

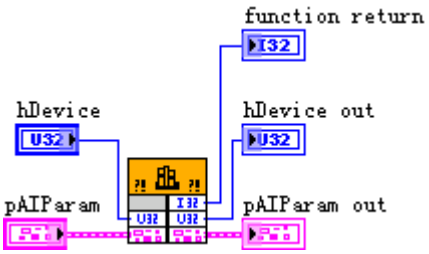
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_SaveParam (HANDLE hDevice,
                   PAI_PARAM pAIParam)
```

Visual Basic:

```
Declare Function AI_SaveParam Lib "USB3100" (ByVal hDevice As Long, _
                                             ByRef pAIParam As AI_PARAM) As Boolean
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：将用户配置好的 AI 工作参数保存在系统注册表中(Save parameter to system registry for analog input)。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

pAIParam 入口参数，AI 工作参数结构体指针，关于 AI_PARAM 的详细介绍请参考 USB3100.h 或 USB3100.bas 或 USB3100.pas 等函数原型定义头文件，也可参考本文《第一节、AI_PARAM（AI 工作参数结构体）》

返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AI_InitParam() AI_LoadParam()

AI_SaveParam()　AI_ResetParam()　DEV_Release()

AI_ResetParam()

函数原型：

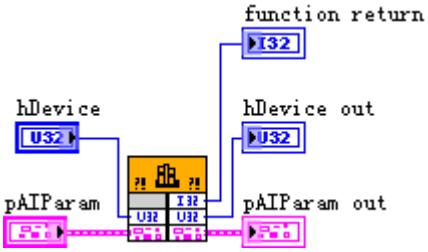
Visual C++/C++Builder/LabWindows/CVI:

```
BOOL AI_ResetParam (HANDLE hDevice,
                    PAI_PARAM pAIParam)
```

Visual Basic:

```
Declare Function AI_ResetParam Lib "USB3100" ( ByVal hDevice As Long, _
                                                ByRef pAIParam As AI_PARAM) As Boolean
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能:将系统注册表中原来的 AI 参数值复位至出厂时的默认值(Reset parameter to default value for analog input)。以防用户不小心将各参数设置不当造成一时无法看到正常的采样结果。比如用户无意间将时钟源选择成了外时钟，却未给外时钟 EXCLK 提供相应的时钟脉冲，那么 AI 永远也不会工作，而用户可能完全不知道是此参数的原因造成这样的结果，之后再想让 AI 工作起来，却不知所措。那么此函数的功能就是将所有的参数恢复成出厂默认值，而该默认值是可以让 AI 以最简单的方式工作起来。

参数:

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建。

pAIParam 出口参数，AI 工作参数结构指针，它负责在参数被复位后返回其复位后的参数值。关于 AI_PARAM 的详细介绍请参考 USB3100.h 或 USB3100.bas 或 USB3100.pas 函数原型定义头文件，也可参考本文《第一节、AI_PARAM（AI 工作参数结构体）》

返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：　DEV_Create()　AI_InitParam()　AI_LoadParam()
AI_SaveParam()　AI_ResetParam()　DEV_Release()

关于调用流程的图例请参考《第二章 使用提要》中《第六节、如何实现 AI 实时单点简单采样》相关部分。

第三节、CTR 计数器函数原型说明

CTR_InitParam()

函数原型：

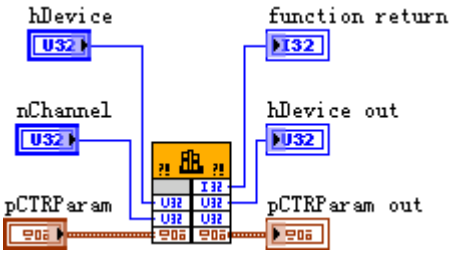
Visual C++/C++Builder/LabWindows/CVI:

```
BOOL CTR_InitParam(HANDLE hDevice, U32 nChannel, CTR_PARAM pCTRParam);
```

Visual Basic:

```
Declare Function CTR_InitParam Lib "USB3100" ( ByVal hDevice As Long, _
                                                ByVal nChannel As Long, _
                                                ByVal pCTRParam As CTR_PARAM) As Boolean
```

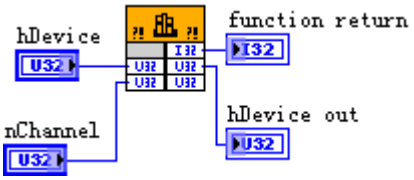
LabVIEW



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：初始化指定计数器通道的工作参数 (Initialize work parameter for Counter)。
参数：
hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。
nChannel 入口参数，计数器通道号，由于本设备只有 1 个计数器通道，故恒等于 0。
pCTRParam 入口参数，计数器工作参数结构体，详情参考《第六节、CTR_PARAM (CTR 计数器工作参数结构体)》
返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。
相关函数： DEV_Create() CTR_InitParam() CTR_Start()
CTR_ReadChannel() CTR_Stop() CTR_Release()
DEV_Release()

CTR_Start()

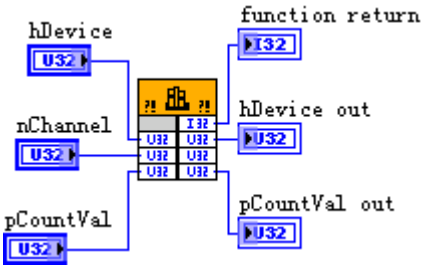
函数原型：
Visual C++ / C++Builder / LabWindows/CVI:
BOOL AI_Start (HANDLE hDevice, U32 nChannel)
Visual Basic:
Declare Function AI_Start Lib "USB3100" (ByVal hDevice As Long, ByVal nChannel As Long) As Boolean
LabVIEW:


请参考 USB3100.lvlib 库文件及相关演示 vi

功能：启动 CTR 计数(Start count for counter)，它必须在成功调用 CTR_InitParam()函数后才能调用此函数，调用该函数后 CTR 立即启动。如果在调用 CTR_Stop()之后调用此函数，则意味着让 CTR 接着之前的计数值继续计数。
参数：**hDevice** 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。
nChannel 入口参数，计数器通道号，由于本设备只有一个计数器，故恒等于 0
返回值：如果调用成功，则返回 TRUE，CTR 立刻被启动， 否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。
相关函数： DEV_Create() CTR_InitParam() CTR_Start()
CTR_ReadChannel() CTR_Stop() CTR_Release()
DEV_Release()

CTR_ReadChannel()

函数原型：
Visual C++ / C++Builder / LabWindows/CVI:
BOOL CTR_ReadChannel(HANDLE hDevice, U32 nChannel, U32* pCountVal);
Visual Basic:
Declare Function CTR_ReadChannel Lib "USB3100" (ByVal hDevice As Long, _
ByVal nChannel As Long, _
ByVal pCountVal As Long) As Boolean
LabVIEW



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：向指定模拟量输出通道写入电压数据 (Write the voltage to one channel of analog output)。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

nChannel 入口参数，计数器通道号，由于本设备只有一个计数器，故恒等于 0

pCountVal 出口参数，计数器当前计数值。

返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：DEV_Create() CTR_InitParam() CTR_Start() CTR_ReadChannel() CTR_Stop() CTR_Release() DEV_Release()

CTR_Stop()

函数原型：

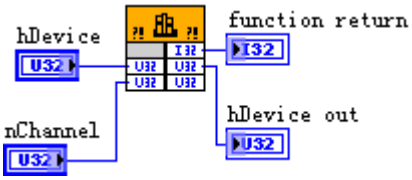
Visual C++ / C++Builder / LabWindows/CVI:

BOOL CTR_Stop(HANDLE hDevice, U32 nChannel)

Visual Basic:

Declare Function CTR_Stop Lib "USB3100" (ByVal hDevice As Long, ByVal nChannel As Long) As Boolean

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：启动 CTR 计数(Stop count for counter)，它必须在成功调用 CTR_Start()函数后才能调用此函数，调用该函数后 CTR 立即停止计数。之后还可以再调用 CTR_Start()继续计数。

参数：**hDevice** 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

nChannel 入口参数，计数器通道号，由于本设备只有一个计数器，故恒等于 0

返回值：如果调用成功，则返回 TRUE，CTR 立刻被停止， 否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：DEV_Create() CTR_InitParam() CTR_Start() CTR_ReadChannel() CTR_Stop() CTR_Release() DEV_Release()

CTR_Release()

函数原型：

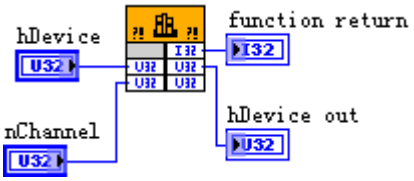
Visual C++ / C++Builder / LabWindows/CVI:

BOOL CTR_Release (HANDLE hDevice, U32 nChannel)

Visual Basic:

Declare Function CTR_Release Lib "USB3100" (ByVal hDevice As Long, ByVal nChannel As Long) As Boolean

LabVIEW:

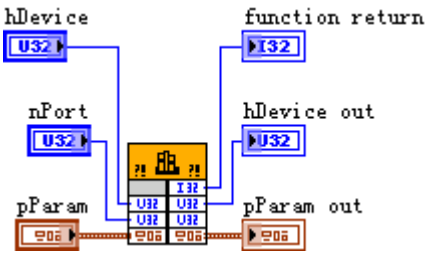


请参考 USB3100.lvlib 库文件及相关演示 vi

功能：释放 CTR (Relase Counter)，它必须在成功调用 CTR_InitParam()函数后才能调用此函数。
参数：**hDevice** 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。
nChannel 入口参数，计数器通道号，由于本设备只有一个计数器，故恒等于 0
返回值：如果调用成功，则返回 TRUE，CTR 被成功释放， 否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。
相关函数： DEV_Create() CTR_InitParam() CTR_Start()
CTR_ReadChannel() CTR_Stop() CTR_Release()
DEV_Release()

CTR 控制流程
(1) DEV_Create() 创建设备句柄
(2) CTR_InitParam() 初始化 CTR 工作参数
(3) CTR_Start() 启动 CTR 计数(或重启)
(4) CTR_ReadChannel() 实时读取 CTR 当前计数值
(5) CTR_Stop() 停止 CTR 计数（或暂停）
(6) CTR_Release() 释放 CTR 资源
(7) DEV_Release() 释放设备句柄
用户可以反复执行第(3)步，以进行数字量输入输出

第四节、DIO 数字量输入输出函数原型说明

DIO_InitParam()
函数原型：
Visual C++ / C++Builder / LabWindows/CVI:
```
BOOL DIO_InitParam(HANDLE hDevice,
                   U32 nPort,
                   DIO_PARAM pDIOParam)
```
Visual Basic:
```
Declare Function DIO_InitParam Lib "USB3100" ( ByVal hDevice As Long, _
                                                ByVal nPort As Long, _
                                                ByRef pDIOParam As DIO_PARAM) As Boolean
```
LabVIEW


请参考 USB3100.lvlib 库文件及相关演示 vi

功能：初始化 DIO 的工作参数(Initialize work parameter for digital I/O)。
参数：
hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。
nPort 入口参数，端口号，由于本设备仅支持一个端口，故恒等于 0。
pDIOParam 入口参数，DIO 工作参数结构体指针,关于该结构体的详细说明请参考《 第七节、DIO_PARAM (DIO 数字量工作参数结构体) 》

返回值：若成功，返回 TRUE; 否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() DIO_InitParam() DIO_GetParam() DIO_ReadPort() DIO_WritePort() DIO_ReadLine() DIO_WriteLine() DIO_Release() DEV_Release()

DIO_GetParam()

函数原型：

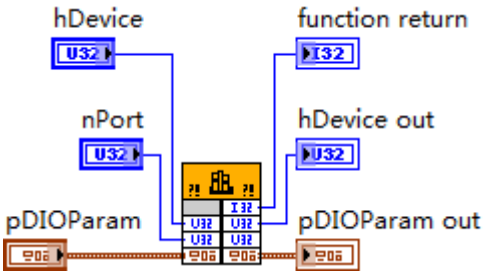
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DIO_GetParam(HANDLE hDevice,
                  U32 nPort,
                  DIO_PARAM pDIOParam)
```

Visual Basic:

```
Declare Function DIO_GetParam Lib "USB3100" ( ByVal hDevice As Long, _
                                              ByVal nPort As Long, _
                                              ByRef pDIOParam As DIO_PARAM) As Boolean
```

LabVIEW



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：获取(回读)DIO 的工作参数(Get work parameter for digital I/O)。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

nPort 入口参数，端口号，由于本设备仅支持一个端口，故恒等于 0。

pDIOParam 出口参数，DIO 工作参数结构体指针,关于该结构体的详细说明请参考《第七节、DIO_PARAM (DIO 数字量工作参数结构体)》

返回值：若成功，返回 TRUE; 否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() DIO_InitParam() DIO_GetParam() DIO_ReadPort() DIO_WritePort() DIO_ReadLine() DIO_WriteLine() DIO_Release() DEV_Release()

DIO_ReadPort()

函数原型：

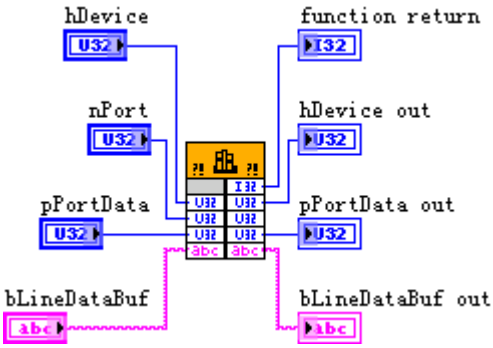
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DIO_ReadPort (HANDLE hDevice, U32 nPort, U32* pPortData, U8 bLineDataBuf[4]);
```

Visual Basic:

```
Declare Function DIO_ReadPort Lib "USB3100" (ByVal hDevice As Long, _
                                             ByVal nPort As Long, _
                                             ByRef pPortData As Long, _
                                              ByRef bLineDataBuf As Byte) As Boolean
```

LabVIEW



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：从指定的 DIO 端口读取端口数据 (Read port data from the DIO port)。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建。

nPort 入口参数，端口号，由于本设备仅支持一个端口，故恒等于 0。

pPortData 出口参数，从指定端口读入的端口数据，其中 Bit[0:3]有效，分别对应于 DIO0-DIO3 数字量输入状态。

bLineDataBuf 出口参数，线数据缓冲，返回从指定端口读入的各线数据。bLineDataBuf [0]、bLineDataBuf [1]、bLineDataBuf [2]、bLineDataBuf [3]分别代表四个 DIO 通道的线值。它的各线值与 pPortData 返回数据中的相应位的值是相同的。这个参数的提供是帮助用户简化操作的，当用户需要关注每个通道的线值时，用户不必对端口并行数据进行“或”“与”“移位”等复杂的二进制位运算，而只须直接使用 bLineDataBuf[]中相应值就可以了。

返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() DIO_InitParam() DIO_GetParam() DIO_ReadPort() DIO_WritePort() DIO_ReadLine() DIO_WriteLine() DIO_Release() DEV_Release()

DIO_WritePort()

函数原型：

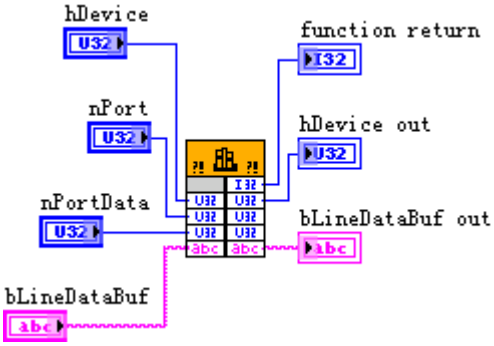
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DIO_WritePort (HANDLE hDevice, U32 nPort, U32* pPortData, U8 bLineDataBuf[4]);
```

Visual Basic:

```
Declare Function DIO_WritePort Lib "USB3100" (ByVal hDevice As Long, _
                                              ByVal nPort As Long, _
                                              ByRef pPortData As Long, _
                                               ByRef bLineDataBuf As Byte) As Boolean
```

LabVIEW



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：向指定的 DIO 端口写入端口数据 (Write port data to the DIO port)。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建。

nPort 入口参数，端口号，由于本设备仅支持一个端口，故恒等于 0。

pPortData 入口参数，向指定端口写入的并行数据，其中 Bit[0:3]有效，分别对应于 DIO0-DIO3 数字量输出状态。该参数只有在 bLineDataBuf=NULL 才有效，如果 bLineDataBuf 不等于 NULL，则 bLineDataBuf 参数优先有效，

即写入到端口中的数据值则由 bLineDataBuf 决定。

bLineBuffer 入口参数，线数据缓冲，存放要写入到指定端口的各线数据。bLineDataBuf [0]、bLineDataBuf [1]、bLineDataBuf [2]、bLineDataBuf [3]分别代表四个 DIO 通道的线值。它的各线值与 pValue 数据中的相应位的值是相同的。这个参数的提供是帮助用户简化操作的，当用户需要关注每个通道的线值时，用户不必对端口并行数据进行“或”“与”“移位”等复杂的二进制位运算，而只须直接使用 bLineDataBuf []中相应赋值就可以了。

返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() DIO_InitParam() DIO_GetParam()
DIO_ReadPort() DIO_WritePort() DIO_ReadLine()
DIO_WriteLine() DIO_Release() DEV_Release()

DIO_ReadLine()

函数原型：

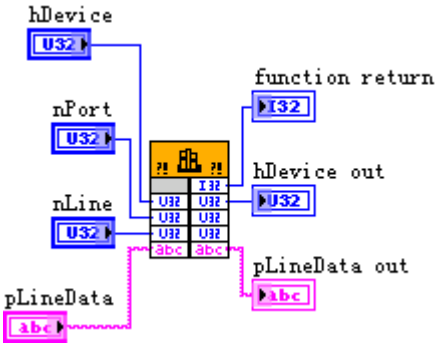
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DIO_ReadLine (HANDLE hDevice, U32 nPort, U32 nLine, U8* pLineData);
```

Visual Basic:

```
Declare Function DIO_ReadLine Lib "USB3100" (ByVal hDevice As Long, _
                                             ByVal nPort As Long, _
                                             ByVal nLine As Long,_
                                             ByRef pLineData As Byte) As Boolean
```

LabVIEW



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：从指定的 DIO 端口中的线号读取线数据 (Read line data from the line of the DIO port)。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建。

nPort 入口参数，端口号，由于本设备仅支持一个端口，故恒等于 0。

nLine 入口参数，指定端口中的线号。由于本设备有一个端口共四条线，故取值范围为 [0, 3]，分别代表 Line0(DIO0)、Line1(DIO1)、Line2(DIO2)、Line3(DIO3)。

pLineData 出口参数，从指定端口中指定线号上读入的线数据，线数据只有两种取值：0（低）或 1（高）。

返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() DIO_InitParam() DIO_GetParam()
DIO_ReadPort() DIO_WritePort() DIO_ReadLine()
DIO_WriteLine() DIO_Release() DEV_Release()

DIO_WriteLine()

函数原型：

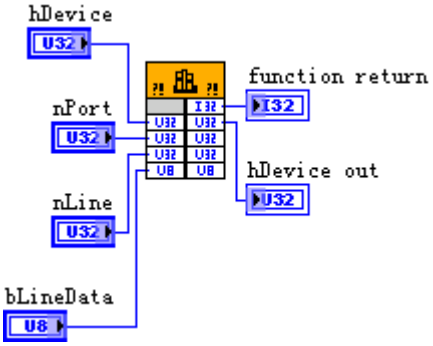
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DIO_WriteLine(HANDLE hDevice, U32 nPort, U32 nLine, U8 bLineData);
```

Visual Basic:

```
Declare Function DIO_WriteLine Lib "USB3100" (ByVal hDevice As Long, _
                                              ByVal nPort As Long, _
                                              ByVal nLine As Long,_
                                              ByVal bLineData As Byte) As Boolean
```

LabVIEW



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：向指定的 DIO 端口的指定线号上写入线数据 (Write line data to the line of the DIO port)。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建。

nPort 入口参数，端口号，由于本设备仅支持一个端口，故恒等于 0。

nLine 入口参数，指定端口中的线号。由于本设备有一个端口共四条线，故取值范围为 [0, 3]，分别代表 Line0(DIO0)、Line1(DIO1)、Line2(DIO2)、Line3(DIO3)。

bLineData 出入口参数，向指定端口中指定线号上写入的线数据，线数据只有两种取值：0（低）或 1（高）。

返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() DIO_InitParam() DIO_GetParam() DIO_ReadPort() DIO_WritePort() DIO_ReadLine() DIO_WriteLine() DIO_Release() DEV_Release()

DIO_Release()

函数原型：

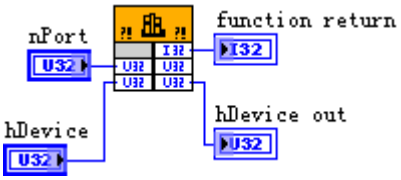
Visual C++ / C++Builder / LabWindows/CVI:

BOOL DIO_Release (HANDLE hDevice, U32 nPort)

Visual Basic:

Declare Function DIO_Release Lib "USB3100" (ByVal hDevice As Long, ByVal nPort As Long) As Boolean

LabVIEW



请参考相关演示程序。

功能：释放 DIO 资源(Release resource for digital input/output)。

参数：

hDevice 入口参数，设备对象句柄，它应由 DEV_Create()函数创建，该句柄指向用户要访问的设备。

返回值：若成功，返回 TRUE，否则返回 FALSE，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() DIO_InitParam() DIO_GetParam() DIO_ReadPort() DIO_WritePort() DIO_ReadLine() DIO_WriteLine() DIO_Release() DEV_Release()

DIO 控制流程

(1) DEV_Create() 创建设备句柄

(2) DIO_InitParam() 初始化 DIO 工作参数

(3) DIO_ReadPort()或 DIO_WritePort()、DIO_ReadLine()、DIO_WriteLine() 实时读取 DIO 端口或线数据

(4) DEV_Release() 释放设备句柄

用户可以反复执行第(3)步，以进行数字量输入输出

第五节、EVENT 事件原型说明

EVENT_Create()

函数原型：

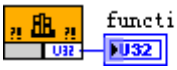
Visual C++/C++Builder/LabWindows/CVI:

```
BOOL EVENT_Create (void);
```

Visual Basic:

```
Declare Function EVENT_Create Lib "USB3100" () As Long
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：创建事件对象(Create event object)。它创建的是自动复位，初始状态为不发信号的事件对象。

参数：

无

返回值：如果成功，返回事件对象的句柄，否则返回 INVALID_HANDLE_VALUE(即-1)。

相关函数： AI_InitParam() EVENT_Create() EVENT_Release()

EVENT_Release()

函数原型：

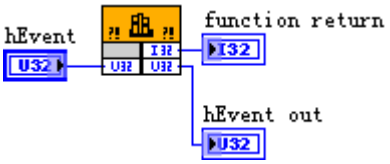
Visual C++/C++Builder/LabWindows/CVI:

```
BOOL EVENT_Release (HANDLE hEvent);
```

Visual Basic:

```
Declare Function EVENT_Release Lib "USB3100" (ByVal hEvent As Long) As Boolean
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

功能：释放事件句柄

参数：

hEvent 入口参数，事件对象句柄，它应由 EVENT_Create()函数创建。

返回值：返回 TRUE，则表示释放成功，否则返回 FALSE 表示失败，用户可以立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： AI_InitParam() EVENT_Create() EVENT_Release()

第四章 各种结构体描述

第一节、AI_PARAM（AI 工作参数结构体）

Visual C++/C++Builder/LabWindows/CVI:

```
typedef struct _AI_CH_PARAM
{
    U32 nChannel;
    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
}

typedef struct _AI_PARAM
{
    U32 nSampleMode;
```

```
    U32 nPointsPerChan;
    F64 fRate;
    U32 nSampChanCount;
    U32 nReserved0;
    USB3100_AI_CH_GROUP CHParam[8];
    U32 nGroupLoops;
    U32 nGroupInterval;

    U32 bDTriggerEn;
    U32 nDTriggerType;
    U32 nDTriggerDir;

    U32 bATriggerEn;
    U32 nATriggerType;
    U32 nATriggerDir;
    U32 nATrigChannel;
    F32 fTriggerLevel;
    U32 nTriggerSens;

    U32 nClockSource;
    U32 bClockOutput;

    U32 nReserved1;
    U32 nReserved2;
    U32 nReserved3;
} AI_PARAM, *PAI_PARAM;
```

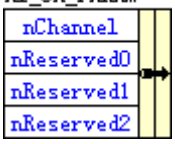
Visual Basic：

```
Private Type AI_CH_PARAM
    nChannel As Long
    nReserved0 As Long
    nReserved1 As Long
    nReserved2 As Long
End Type

Private Type AI_PARAM
    nSampleRate As Long
    nPointsPerChan As Long
    fRate As Long
    nSampChanCount As Long
    nReserved0 As Long
    CHParam(0 to 7) As AI_CH_GROUP
    nGroupLoops As Long
    nGroupInterval As Long

    bDTriggerEn As Long
    nDTriggerType As Long
    nDTriggerDir As Long

    bATriggerEn As Long
    nATriggerType As Long
    nATriggerDir As Long
    nATrigChannel As Long
    fTriggerLevel As Long
    nTriggerSens As Long

    nClockSource As Long
    bClockOutput As Long

    nReserved1 As Long
    nReserved2 As Long
```

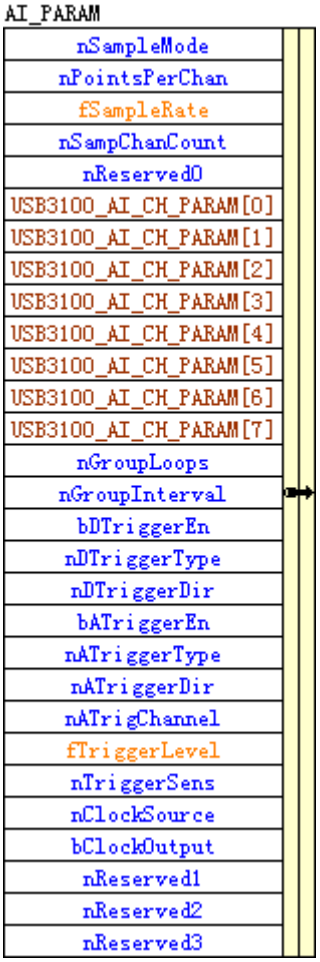
nReserved3 As Long
End Type

LabVIEW:

AI_CH_PARAM



AI_PARAM



请参考 USB3100.lvlib 库文件及相关演示 vi

一、AI_CH_PARAM(AI 通道参数结构体)

nChannel

AI 物理通道号，取值范围[0, 7]，即 AI0－AI7

nReserved0-2

保留未用

二、AI_PARAM(AI 工作参数结构体）

nSampleMode

AI 采样模式，取值[0, 1]，具体定义见下表：

常量名	常量值	功能定义	备注
AI_SAMPMODE_ONE_DEMAND	0	单点采样(按需)	
AI_SAMPMODE_CONTINUOUS	1	连续采样	

单点采样（按需要）模式：就是每次当用户调用 AI_OneReadChannelsV() 或 AI_OneReadChannels()时设备才开始采样，且以最快的速度返回，返回时设备为每个参加采样的通道仅准备一个点的数据。这种采样模式主要针对简单采样或采样实时性要求较高，数据量很少，且时间不确定的应用中，比如 PID，PLC 等快速伺服闭环控制系统。

连续采样模式：就是按照用户设定好的采样速率和通道分组模式，触发模式等参数进行长时间的，不限时的，连续等间隔的波形数据采集。这个功能主要是应用在虽然尽可能还原信号原始形态的场合。

nPointsPerChan

AI 每通道待读取点数。它决定着每次调用 AI_ContReadChannelsV()或 AI_ContReadChannels()时能返回的最大点数。取值范围为[256，8192]，则具体取值必须为 256 的整数倍长。

fSampleRate

AI 采样率(Sample Rate)，单位：每秒样点 sps(sample per second)，它指每个采样通道的采样速率，它决定了单个采样通道每秒钟采样的点数。而每个采样点的周期则由 fRate 求倒数取得，单位：秒(S)。它的最小取值等于 1sps，而最大取值则由设备的总采样率(20000sps)、采样通道数量(nSampChanCount)决定，比如采样通道数量为 3 个通道，则该参数最大取值为 20000sps/3 = 66666sps，比如采样通道数量为 4 个通道，则该参数最大取值为 20000sps/4 = 5000sps；比如采样通道数量为 8 个通道，则该参数最大取值为 20000sps/8 =2500sps。

nSampChanCount

采样通道数量(Sample Channel Count)，它决定进入采样过程的通道个数，取值范围[1，8]。它实际决定了 CHParam[]通道组阵列中有效单元的个数。比如 nSampChanCount=1，则表示仅 CHParam[0]单元决定的物理通道号有效；比如 nSampChanCount=2，则表示仅 CHParam[0]、CHParam[1]两个单元决定的物理通道号有效；比如 nSampChanCount=3，则表示仅 CHParam[0]、CHParam[1]、、CHParam[2]三个单元决定的物理通道号有效，依次类推。

CHParam[8]

通道组(Channel Parameter)，共 8 个单元，分别控制 8 个采样通道的选择。由于本设备采用更先进的任意通道组合扫描采样的方式，让用户对采样通道的选择与组合更加灵活，更能贴近用户的实际应用。该参数的有效单元个数由 nSampChanCount 参数决定。每个有效单元决定用户要采集的物理通道号，增益和参考地等工作参数。具体定义请参考《一、AI_CH_PARAM(AI 通道参数结构体) 》

nGroupLoops

通道组循环次数，取值范围为[1, 65535]。它决定着在分组采样过程中，每个通道组（由 nSampChanCount 和 CHParam[]共同决定）在每个通道组扫描时钟下循环采样的次数（即每个通道的点数）。如果 nGroupLoops=1, 且 nGroupInterval=0，则为等间隔采样。否则为分组采样。

nGroupInterval

每两个通道组间的时间间隔，单位：微秒(uS)。取值范围为[0, 107374182]。如果它等于 0，且 nGroupLoops=1 则表示处于等间隔采样。

注意：在分组采样中，组内或组循环内的通道采样间隔是相等的，且时间由 fRate*nSampChanCount 的乘积求倒数得到(单位：秒)。而这一组（包括组循环）与下一组（包括组循环）之间的时间间隔则由 nGroupInterval 决定；在等间隔采样中，所有采样数据的时间间隔均由 fRate*nSampChanCount 的乘积求倒数得到的时间决定(单位：秒)。

bDTriggerEn

数字量触发 DTR 允许(Digital Trigger Enable)。若等于 TRUE，表示外部数字量触发输入信号允许进入 AI 的触发系统，若等于 FALSE，表示不允许(即禁用)。触发信号由连接器 CN2 上的 DIO1 复用输入，因此初始化时会将 DIO1 的方向强制置为输入

nDTriggerType

数字量触发类型(Digital Trigger Type)。它的取值如下表：

常量名	常量值	功能定义	备注

AI_TRIGTYPE_EDGE	0	边沿触发	默认值
AI_TRIGTYPE_LEVEL	1	电平触发	

nDTriggerDir

数字量触发极性(Digital Trigger Direction)。它的取值如下表：

常量名	常量值	功能定义	备注
AI_TRIGDIR_FALLING	0	下降沿触发/低电平	默认值
AI_TRIGDIR_RISING	1	上升沿触发/高电平	
AI_TRIGDIR_CHANGE	2	变化(上下边沿/高低电平)	

bATriggerEn

模拟量触发 DTR 允许(Analog Trigger Enable)。若等于 TRUE，表示模拟通道触发输入信号允许进入 AI 的触发系统，若等于 FALSE，表示不允许(即禁用)。

nATriggerType

模拟量触发类型(Analog Trigger Type)。它的取值如下表：

常量名	常量值	功能定义	备注
AI_TRIGTYPE_EDGE	0	边沿触发	默认值
AI_TRIGTYPE_LEVEL	1	电平触发	

nATriggerDir

模拟量触发极性(Analog Trigger Direction)。它的取值如下表：

常量名	常量值	功能定义	备注
AI_TRIGDIR_FALLING	0	下降沿触发/低电平	默认值
AI_TRIGDIR_RISING	1	上升沿触发/高电平	
AI_TRIGDIR_CHANGE	2	变化(上下边沿/高低电平)	

nATrigChannel

模拟量触发通道，它的取值范围为[0, 7]，分别表示物理通道 AI0－AI7。但具体的触发通道必须在采样通道组阵列中选择，也就是说只有处在采样通道组 CHParam[]中的物理通道才能被选择为触发通道。

fTriggerLevel

AI 采样通道的触发电平(Trigger Level)，单位：伏(V)，分辨率 12Bit。它的取值范围要受到 nSampleRange 参数所选择的模拟量输入范围档的限定，如下表所示：

常量名	常量值	采样范围	fTriggerLevel 的取值范围(V)	编码宽度(V)
AI_SAMPRANGE_N10_P10V	0	±10V	[-10.0000, +9.9951]	0.0048828

nTriggerSens

AI 触发灵敏度（Trigger Sense)，单位：微秒(uS)。它的取值范围为[0, 1638]，它的实际功能是为避免触发信号中较高频的噪声分量也进入触发系统而设定的一种时间门限，凡是触发信号跳变后稳定保持时间不小于该门限时间则进入触发系统，否则不进入而被屏蔽掉。此参数仅对数字量和模拟量触发均有效。跳变保持时间：指的是触发信号由一个状态跳变至另一个状态时所能保持的时间。举例说明，一个数字量触发信号原来是低电平，然后跳变至高电平，保持 10 微秒后又回到低电平，即跳变保持时间指的就是高电平在这个瞬间保持的时间 10 微秒。这个时间越短越有可能是高频噪声，因此需要这个参数加以控制或过滤，以避免噪声信号产生误触发。

nClockSource

时钟源(Clock Source)。本设备的 AI 采样时钟支持本地时钟源(LOCAL)和外部时钟源(EXCLK)。本地时钟源即指板上晶振时钟源(OSCCLK)。外部时钟源则来自于由连接器 CN2 上的 DIO2 复用输入，因此当选择为外时钟后初始化 AI_InitParam() 时会将 DIO2 的方向强制置为输入。该参数的取值范围为[0，1]，具体定义如下表：

nClockSource 选项(常量名)	常量值	功能定义	备注
AI_CLKSRC_LOCAL	0	本地时钟，也叫内部时钟(通常为本地晶振时钟	默认值

		OSCCLK)	
AI_CLKSRC_EXCLK	1	外部时钟，由 CN2 中的 DIO2 复用输入	

bClockOutput

本地采样时钟是否输出。=TRUE：表示允许由 fRate 决定的采样时钟会直接输出到 CN2 的 DIO3， =FALSE：禁止输出。因此当允许时钟输出后初始化 AI_InitParam() 时会将 DIO3 的方向强制置为输出。

nReserved1-3

保留字段(未作定义)，可强制赋 0

相关函数： DEV_Create() AI_InitParam() AI_LoadParam()
AI_SaveParam() AI_ResetParam DEV_Release()

第二节、AI_STATUS（AI 工作状态信息结构）

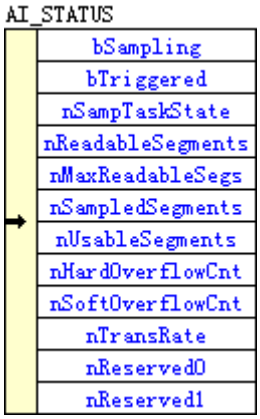
Visual C++ / C++Builder / LabWindows/CVI:

```
typedef struct _AI_STATUS
{
    U32 bSampling;
    U32 bTriggered;

    U32 nSampTaskState;
    U32 nReadableSegments;
    U32 nMaxReadableSegs;
    U32 nSampledSegments;
    U32 nUsableSegments;

    U32 nHardOverflowCnt;
    U32 nSoftOverflowCnt;
    U32 nTransSpeed;

    U32 nReserved0;
    U32 nReserved1;
 } AI_STATUS, *PAI_STATUS;
```

Visual Basic:

```
Private Type    AI_STATUS
    bSampling      As Long
    bTriggered     As Long

    nSampTaskState    As Long
    nReadableSegments    As Long
    nMaxReadableSegs As Long
    nSampledSegments As Long
    nUsableSegments    As Long

    nHardOverflowCnt    As Long
    nSoftOverflowCnt    As Long
    nTransSpeed    As Long

    nReserved0              As Long
    nReserved1              As Long
End Type
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

此结构体主要用于查询 AI 的工作状态信息，AI_GetStatus()函数使用此结构体来实时取得 AI 的工作状态信息，以便同步数据采样和处理过程。

bSampling

AI 正在采样标志。=TRUE:表示正在采样, =FALSE:表示采样结束或未开始采样。在设备上电之初或执行 AI_Stop()函数后其值为 FALSE，当执行 AI_Start()函数后为 TRUE。

bTriggered

AI 触发标志。=TRUE:表示已被触发, =FALSE:表示未被触发或正等待触发。在设备上电之初或执行 AI_Stop()后其值为 TRUE，当执行 AI_Start()后其值为 FALSE。在被正常触发后自动变为 TRUE。

nSampTaskState

采样任务状态, =1:正常, 其它值表示有异常情况。

nReadableSegments

可读段数，只有它大于 0 时才能调用 AI_ContReadChannelsV() 或 AI_ContReadChannels() 读取采样段数据。比如此状态值为 3，则表示设备中有三个数据段是可读的，是最新采样的。

nMaxReadableSegs

自启动采样后，曾经出现过的最多的可读段数。比如在某一时刻 nReadableSegments=2，则该状态值就会等于 2，只要过后 nReadableSegments 永远小于 2，则该状态值就永远等于 2，除非 nReadableSegments 后来又超过 2，比如有个一次 4，则该状态值就保持在 4，依此类推。该状态值的作用是为了验证用户程序的整体效率而提供的。如果该状态值在采样任务长期运行过程中越小越好，则表示应用程序的读取效率和处理效率都很高，设备采样溢出丢点的可能性几乎为 0。如果此值比较贴近于 nUsableSegments（通常设备固定为 16），那么溢出的可能性就比较大了。如果超越了 nUsableSegments，则意味着采样任务已经发生过溢出了，其溢出次数则可以通过 nHardOverflowCnt 和 nSoftOverflowCnt 观察到。

nSampledSegments

自启动采样后，已经采样过的段数（Sampled Segment Count）。它间接反映了用户该次采样的数据总量。

nUsableSegments

设备支持的最大可用段数（Usable Segment Count），通常固定为 16。表示设备最多可以缓冲 16 个段的数据，其段长由 AIParam.nPointsPerChan*AIParam.nSampChanCount 的乘积决定。

nHardOverflowCnt

硬件溢出计数（Hardware Overflow Count）。在启动采样后，绝大多数情况下，设备中的硬件缓存是不会溢出。但如果因为某种原因，如用户的计算机系统极度繁忙或应用程序处理不当，效率不高，则有可能会引起硬件缓存溢出，则该计数器就会自动加 1，之后用户又快速地读走了缓存中的采样数据，那么缓冲区又为不溢出状态，如果之后又溢出了，则该计数器又会自动加 1。如果用户重新启动采样，则该计数器自动清零。因此，该计数器值是作为

分析用户的应用软件设计是否高效，用户的计算机整个系统是否高效提供了有利的参考信息。

nSoftOverflowCnt

软件溢出计数（Software Overflow Count）。在启动采样后，绝大多数情况下，设备中的软件缓存是不会溢出。但如果因为某种原因，如用户的计算机系统极度繁忙或应用程序处理不当，效率不高，则有可能会引起软件缓存溢出，则该计数器就会自动加 1，之后用户又快速地读走了缓存中的采样数据，那么缓冲区又为不溢出状态，如果之后又溢出了，则该计数器又会自动加 1。如果用户重新启动采样，则该计数器自动清零。因此，该计数器值是作为分析用户的应用软件设计是否高效，用户的计算机整个系统是否高效提供了有利的参考信息。

nTransRate

设备传输速率（Transfer Rate），单位：点/秒(P/S)。它反应了在 AI 采样过程中，实时传输 AI 采样数据的平均秒速度，即每秒传输了多少点的数据。比如用户设定的单个通道采样速率(fRate)为 5000sps，采样通道数(nSampChanCount)为 4，则总采样速率为 20000sps（5000*4），那么正常情况下，该状态值应等于 20000P 左右。因此该状态值也是为判断系统性能提供的有利参考信息。

nReserved0-1

保留字段(未作定义)

相关函数：AI_GetStatus()

第三节、AI_MAIN_INFO（AI 主要信息结构体）

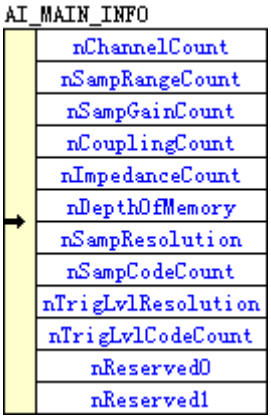
Visual C++ / C++Builder / LabWindows/CVI:

```
typedef struct _ AI_MAIN_INFO
{
    U32 nChannelCount;
    U32 nSampRangeCount;
    U32 nSampGainCount;
    U32 nCouplingCount;
    U32 nImpedanceCount;
    U32 nDepthOfMemory;
    U32 nSampResolution;
    U32 nSampCodeCount;
    U32 nTrigLvlResolution;
    U32 nTrigLvlCodeCount;

    U32 nReserved0;
    U32 nReserved1;
} AI_MAIN_INFO, *PAI_MAIN_INFO;
```

Visual Basic：

```
Private Type AI_MAIN_INFO
    nChannelCount As Long
    nSampRangeCount As Long
    nSampGainCount As Long
    nCouplingCount As Long
    nImpedanceCount As Long
    nDepthOfMemory As Long
    nSampResolution As Long
    nSampCodeCount As Long
    nTrigLvlResolution As Long
    nTrigLvlCodeCount As Long

    nReserved0 As Long
    nReserved1As Long
End Type
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

nChannelCount
物理通道数量。

nSampRangeCount
采样范围挡位数量。

nSampGainCount
输入增益挡位数量。

nCouplingCount
耦合方式挡位数量。

nImpedanceCount
阻抗挡位数量。

nDepthOfMemory
AI 各板载通道内存容量深度(点数)。

nDepthOfMemory
AI 各板载通道内存容量深度(点数)。

nSampResolution
采样分辨率(如=8 表示 8Bit; =12 表示 12Bit; =14 表示 14Bit; =16 表示 16Bit)

nSampCodeCount
AI 采样编码数量(如 256, 4096, 16384, 65536)

nTrigLvlResolution
触发电平(fTriggerLevel)分辨率(如=8 表示 8Bit; =12 表示 12Bit; =16 表示 16Bit)

nTrigLvlCodeCount
触发电平编码数量(如 256, 4096)

nReserved0-1
保留字段(未作定义)

相关函数： AI_GetMainInfo()

第四节、AI_SAMP_RANGE_INFO（AI 采样范围信息结构体）

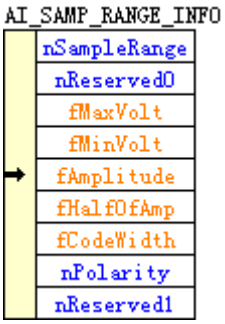
Visual C++ / C++Builder / LabWindows/CVI:

```
typedef struct _ AI_SAMP_RANGE_INFO
{
    U32 nSampleRange;
    U32 nReserved0;
    F64 fMaxVolt;
    F64 fMinVolt;
    F64 fAmplitude;
    F64 fHalfOfAmp;
    F64 fCodeWidth;
    U32 nPolarity;

    U32 nReserved1;
} AI_SAMP_RANGE_INFO, *P AI_SAMP_RANGE_INFO;
```

Visual Basic:

```
Private Type AI_SAMP_RANGE_INFO
    nSampleRange As Long
    nReserved0 As Long
    fMaxVolt As Double
    fMinVolt As Double
    fAmplitude As Double
    fHalfOfAmp As Double
    fCodeWidth As Double
    nPolarity As Long

    nReserved1As Long
End Type
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

nSampleRange

当前输入采样范围挡位号

nReserved0

保留字段

fMaxVolt

采样范围范围的上限电压值，单位：伏(V)。

fMinVolt

采样范围范围的下限电压值，单位：伏(V)。

fAmplitude

采样范围范围的幅度电压值，单位：伏(V)。它也可以由 dRangeH – dRangeL 得到。

fHalfOfAmp

幅度的二分之一电压值，单位：伏(V)。它也可以由 fAmplitude/2 得到。

fCodeWidth

编码宽度。若幅度值为 20V，且 AI 分辨率为 12Bit(即总的 LSB 个数为 4096)，那么 dVoltOfLSB 应为 20/4096，即约等于 0.00488 伏。

nPolarity

AI 采样范围的极性。

nPolarity 选项(常量名)	常量值	功能定义	备注
AI_POLAR__BIPOLAR	0	双极性，即指正负电压均可输入	默认值
AI_POLAR__UNIPOLAR	1	单极性，即指只能正电压可输入	

nReserved1

保留字段

相关函数： AI_GetRangeInfo()

第五节、AI_SAMP_RATE_INFO（AI 采样速率信息结构体）

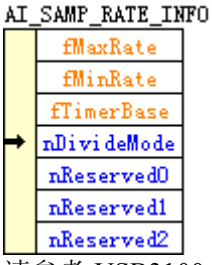
Visual C++ / C++Builder / LabWindows/CVI:

```
typedef struct _ AI_SAMP_RATE_INFO
{
    F64 fMaxRate;
    F64 fMinRate;
    F64 fTimerBase
    U32 nDivideMode;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
} AI_SAMP_RATE_INFO, *PAI_SAMP_RATE_INFO;
```

Visual Basic:

```
Private Type AI_SAMP_RATE_INFO
    fMaxRate As Double
    fMinRate As Double
    fTimerBase As Double
    nDivideMode As Long

    nReserved0 As Long
    nReserved1 As Long
    nReserved2 As Long
End Type
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

fMaxRate

AI 最大采样率，单位：点/秒(sps)。

fMinRate

AI 最小采样率，单位：点/秒(sps)。

fTimerBase

时钟基准，即板上使用的晶振大小，单位：赫兹(Hz)

nDivideMode

分频模式，0=整数分频(INTDIV), 1=DDS 分频(DDSDIV)。

nReserved0-2

保留字段

相关函数：AI_GetRateInfo()

第六节、CTR_PARAM（CTR 计数器工作参数结构体）

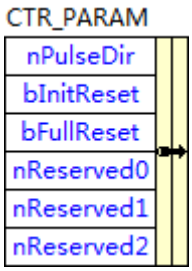
Visual C++ / C++Builder / LabWindows/CVI:

```
typedef struct _CTR_PARAM
{
    U32 nPulseDir;
    U32 bInitReset;
    U32 bFullReset;

    U32 nReserved0;
    U32 nReserved1;
} CTR_PARAM, *PCTR_PARAM;
```

Visual Basic:

```
Private Type CTR_PARAM
    nPulseDir As Long
    bInitReset As Long
    bFullReset As Long

    nReserved0 As Long
    nReserved1 As Long
End Type
```

LabVIEW:



请参考 USB3100.lvlib 库文件及相关演示 vi

nPulseDir

脉冲方向，其取值范围[0，2]，具体定义见下表：

nPolarity 选项(常量名)	常量值	功能定义	备注
AI_PULSEDIR_FALLING	0	下降沿	默认值
AI_PULSEDIR_RISING	1	上升沿	
AI_PULSEDIR_CHANGE	2	变化(上下边沿均有效)	

bInitReset

在 CTR 被初始化(调用 CTR_InitParam())时，是否需要将计数器值复位至 0，如果该参数=TRUE:表示在初始时计数器值自动复位于 0，如果＝FALSE：则表示在初始化时计数器值不变，保持初始化前的值。

bFullReset

当计数器计到满值时是否复位至 0，如果＝TRUE：表示在计数器计数到最大宽度值（32Bit）后则自动复位至 0。如果＝FALSE：则表示计满时不复位，其计数器值保持在最大宽度值上。

nReserved

保留字段

相关函数： CTR_InitParam() CTR_Start() CTR_ReadChannel() CTR_Stop() CTR_Release()

第七节、DIO_PARAM（DIO 数字量工作参数结构体）

Visual C++ / C++Builder / LabWindows/CVI:

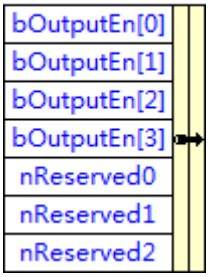
```
typedef struct _DIO_PARAM
{
    U8 bOutputEn[4];

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
} DIO_PARAM, *PDIO_PARAM;
```

Visual Basic:

```
Private Type AI_MAIN_INFO
    bOutputEn (0 to 3)As BYTE

    nReserved0 As Long
    nReserved1 As Long
    nReserved2 As Long
End Type
```

LabVIEW:

DIO_PARAM



请参考 USB3100.lvlib 库文件及相关演示 vi

bOutputEn[]

DIO 输出允许。如果 bOutputEn[n]=TRUE:表示该路 DIO 将被置为输出，否则为输入。

nReserved

保留字段

相关函数： DIO_InitParam() DIO_GetParam() DIO_ReadPort() DIO_WritePort() DIO_ReadLine() DIO_WriteLine() DIO_Release()