USB2861 数据采集卡

驱动程序使用手册

V6.00.02

ART Technology

目　录

1 规范与约定

1.1 关键字缩写命名约定

缩写	全称	汉语意思	缩写	全称	汉语意思
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(传输方式)	SRC/Src	Source	源
SAMP/Samp	Sample	采样			

1.2 数据类型

1.2.1 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)	Long	LongInt
U32	无符号 32 位整型数	0 to 4294967295	unsigned int(long)	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 1844674407370955161	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

1.2.2 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

1.2.3 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

1.3 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，尽可能的延伸后续设计，保护用户的前期投资，便将文档中的产品标识前缀名“USB2861_”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_InitTask”等关键部分，尽可能让用户看到的就是最关心的功能部分，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，因此如果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)。

2 使用提要

2.1 驱动函数的导入方法

为了用户在演示工程中或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++	USB2861.h	USB2861.lib	C:\ART\USB2861\Include
Microsoft Visual Basic	USB2861.bas	无	C:\ART\USB2861\Include
Borland C++ Builder	USB2861.h	USB2861.lib	C:\ART\USB2861\Include
Borland Delphi	USB2861.pas	无	C:\ART\USB2861\Include
NI LabVIEW	USB2861.vi	无	C:\ART\USB2861\Include
NI LabWindows/CVI	USB2861.h	USB2861.lib	C:\ART\USB2861\Include

注：（1）、USB2861.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。

（2）、USB2861RSV.h 是产品的保留头文件，为了凸现 USB2861.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。

2.2 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：

（1）、将 USB2861.dll 从 Windows\System32 中复制到安装包中；

（2）、将 USB2861.inf、USB2861.sys 从安装光盘相应产品文件夹下的 Driver 中复制到安装盘中。

2.3 管理设备

阿尔泰的设备驱动程序采用的是面向对象编程技术，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。如 AI_InitTask()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。最后通过 DEV_Release()函数将 hDevice 释放掉。

2.4 AI 单点采样模式

单点采样，就是在一次开始后，用户每次发出读命令 AI_ReadAnalog()或 AI_ReadBinary()时 AI 以设备最快速度获取各采样通道单个点的数据。具体流程如图 2-4-1 所示：

(1) DEV_Create()　创建设备句柄；

(2) AI_InitTask()　初始化 AI 采集任务，参数 nSampleMode=0；

(3) AI_StartTask()　开始 AI 采集任务；

(4) AI_ReadAnalog()　或者 AI_ReadBinary()　读取 AI 各采样通道单点数据；

(5) AI_StopTask()　停止 AI 采集任务；

(6) AI_ReleaseTask() 释放 AI 采集任务；

(7) DEV_Release() 释放设备句柄。

如果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；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。

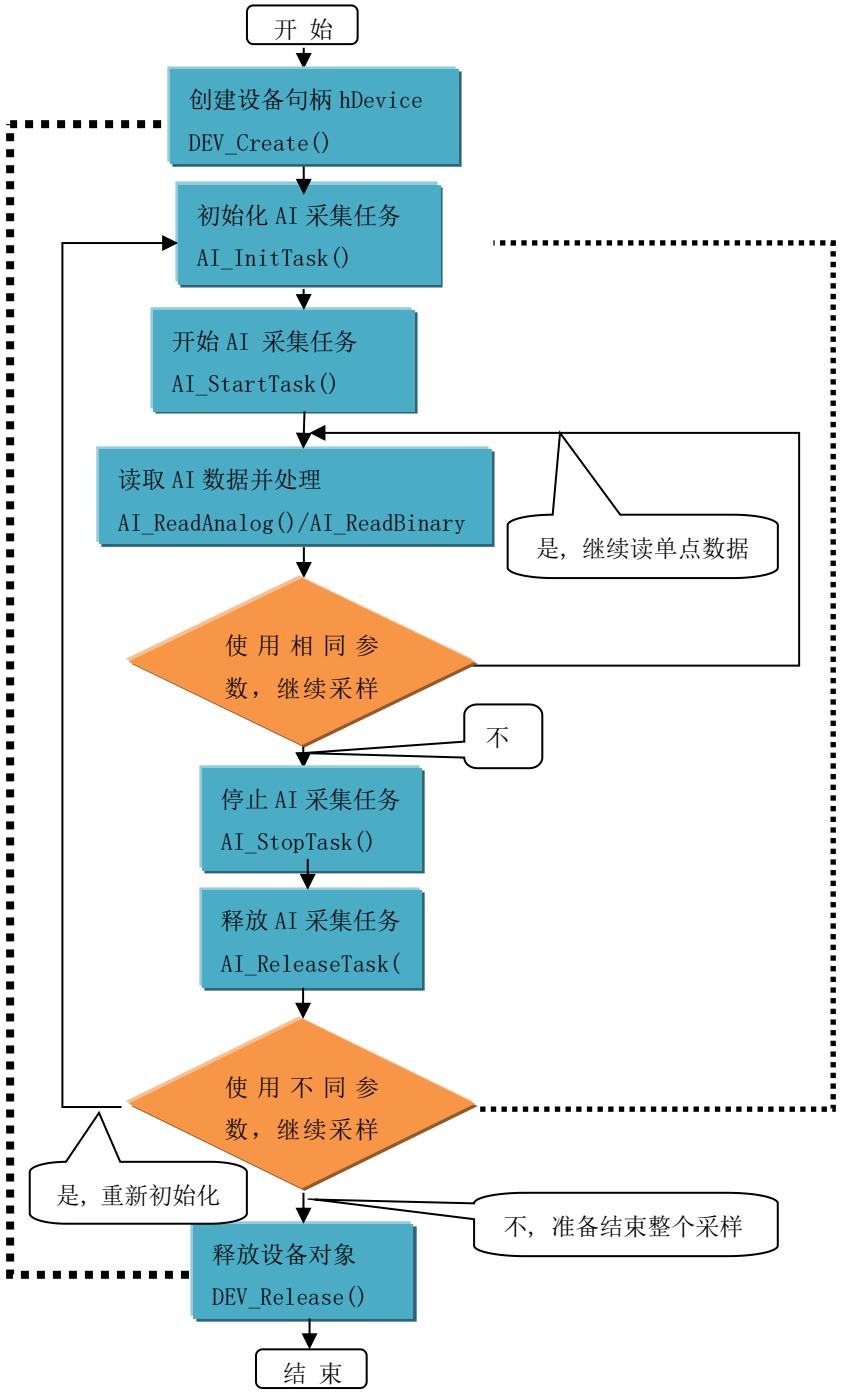


图 2-4-1 AI 实时单点采样流程图

2.5 AI 有限点采样模式

有限点采样模式，就是在一次开始后，AI 按照设定的采样速率，触发条件进行指定时间的或指定点数的连续采样，达到指定点数后设备会自动停止。且在采样结束后才可以读取到完整的数据片断。具体流程如图 2-5-1 所示：

(1) DEV_Create()　创建设备句柄；

(2) AI_InitTask()　初始化 AI 采集任务，注意参数 nSampleMode=3；

(3) AI_StartTask()　开始 AI 采集任务；

(4) AI_ReadAnalog() 或者 AI_ReadBinary() 读取 AI 数据（注意指定适当的超时等待时间）；

(5) AI_StopTask()　停止 AI 采集任务；

(6) AI_ReleaseTask()　释放 AI 采集任务；

(7) DEV_Release() 释放设备句柄。

如果在采集参数相同的情况下，多次捕捉触发事件采样多个片断的数据，可以在 3、4、5 步间循环进行，即可实现高速多次跟踪多个触发事件；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。

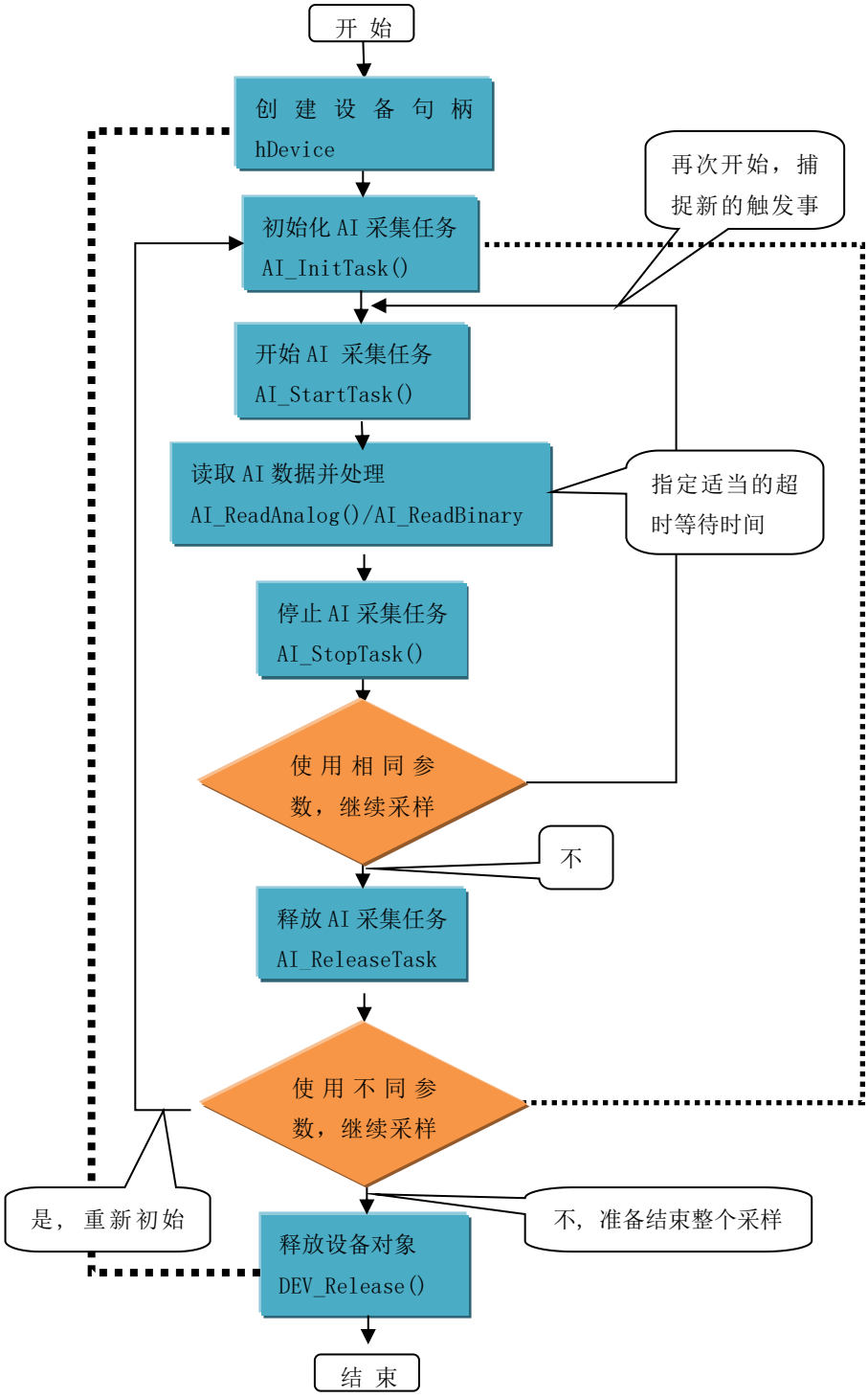


图 2-5-1　　AI 有限点采样流程图例

2.6 AI 连续采样模式

连续采样模式，就是在一次开始后，AI 按照设定的采样速率，触发条件进行长时间的，无限点的连续不间断采样，设备永远不会自动停止（除非用户手动强制停止）。且在采样过程中可以实时读取采样的连续数据。具体流程如图 2-6-1 所示：

（1）DEV_Create()　创建设备句柄；

（2）AI_InitTask()　初始化 AI 采集任务，注意参数 nSampleMode=3；

（3）AI_StartTask()　开始 AI 采集任务；

（4）AI_ReadAnalog() 或者 AI_ReadBinary() 读取 AI 数据（注意指定适当的超时等待时间）；

（5）AI_StopTask()　停止 AI 采集任务；

（6）AI_ReleaseTask() 释放 AI 采集任务；

（7）DEV_Release() 释放设备句柄。

如果每次采集参数相同的情况下，可以在 4 步间循环进行，即可实现高速连续不间断采样；

如果是在参数不变的情况下需要捕获新的触发时，可以在 3、4、5 之间循环进行；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。请参考看下面流程图 2-6-1。

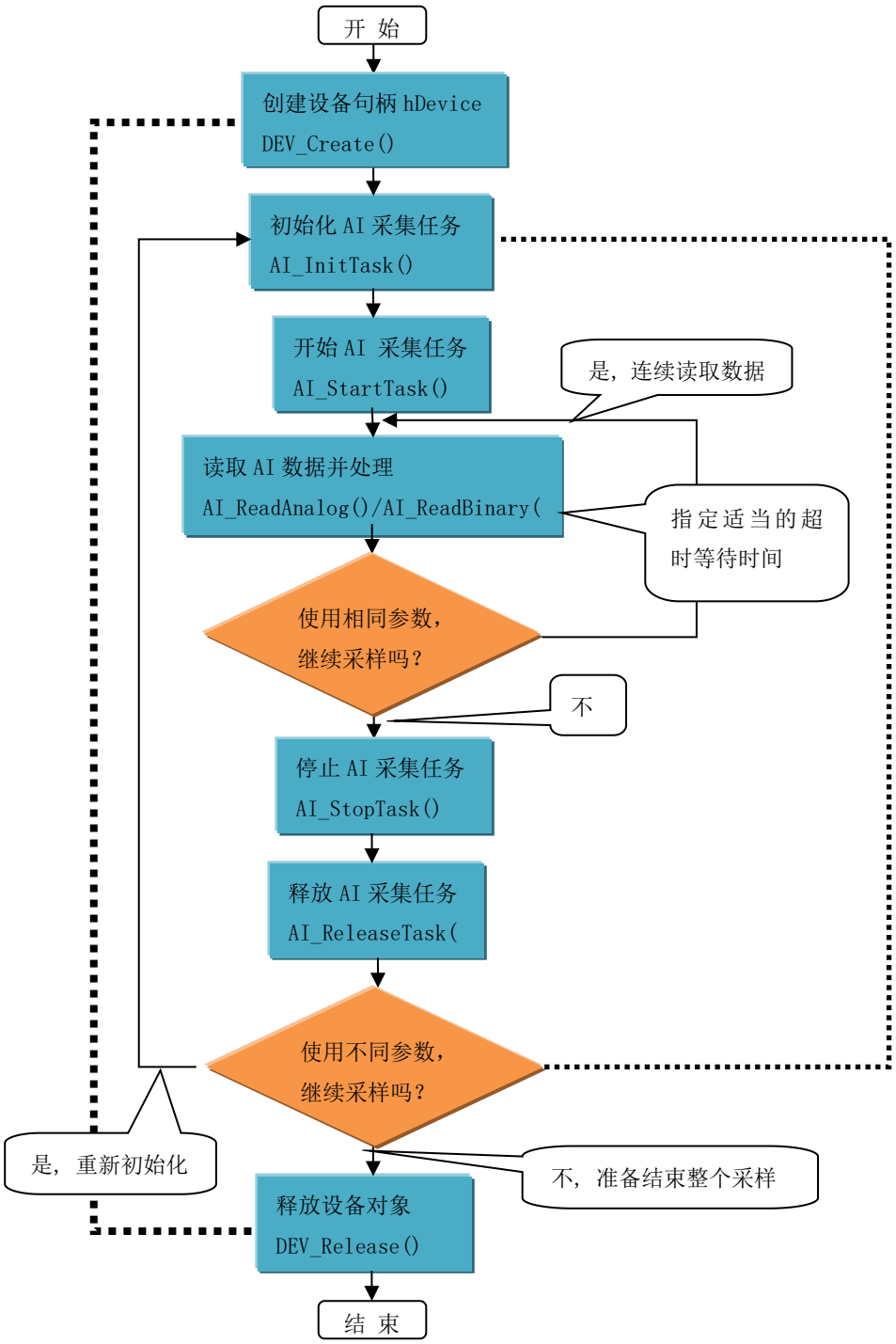


图 2-6-1　　AI 连续采样流程图例

上图虚线表示对称关系。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。AI_InitTask()和 AI_ReleaseTask()两个函数的对称关系是只有在 AI_ReleaseTask()之后才能再次 AI_InitTask()。

2.7　AO 单点采样模式

单点采样，就是在一次开始后，用户每次发出写命令 AO_WriteAnalog()或 AO_WriteBinary()时 AO 以设备最快速度写入各采样通道单个点的数据。具体步骤如下：

（1） DEV_Create() 创建设备句柄；
（2） AO_InitTask() 初始化 AO 生成任务，参数 nSampleMode=0；
（3） AO_StartTask() 开始 AO 生成任务；
（4） AO_WriteAnalog()或者 AO_WriteBinary() 写入 AO 各采样通道单点数据；
（5） AO_StopTask() 停止 AO 生成任务；
（6） AO_ReleaseTask() 释放 AO 生成任务；
（7） DEV_Release() 释放设备句柄。

如果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；
如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。
具体执行流程请看下面的图 2-7-1。

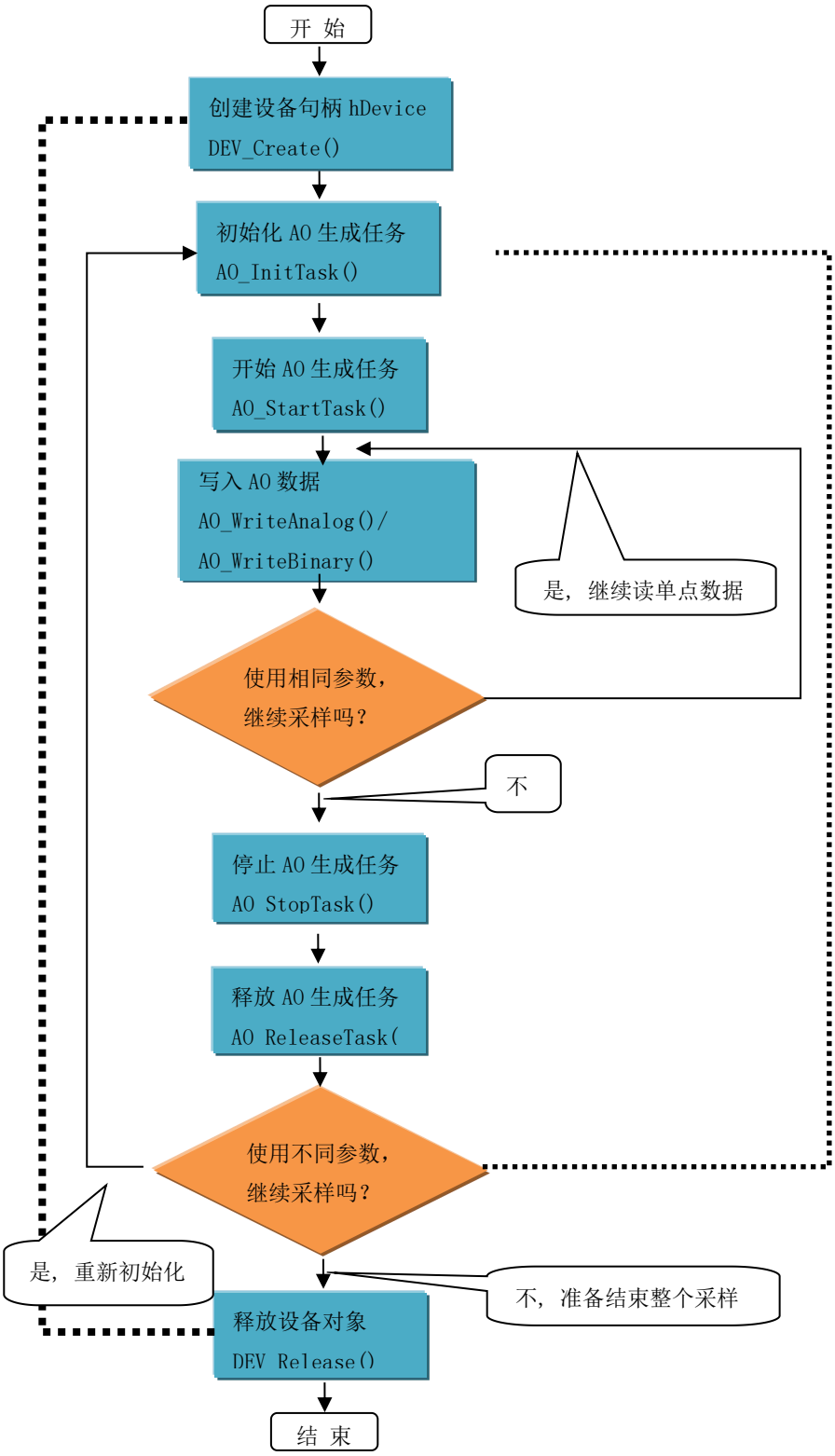


图 2-7-1　AO 实时单点采样流程图例

2.8　AO 有限点采样模式

有限点采样模式，就是在开始前，先写入指定点数的数据到任务中，开始后，AO 按照设定的采样速率，触发条件进行指定时间的或指定点数的连续采样，达到指定点数后设备会自动停止。且

在采样结束后才可以写入下一批数据到任务中以待开始采样。

AO 有限点采样流程:

（1） DEV_Create() 创建设备句柄；

（2） AO_InitTask() 初始化 AO 生成任务，参数 nSampleMode=2；

（3） AO_WriteAnalog()或者 AO_WriteBinary() 写入 AO 各采样通道波形数据；

（4） AO_StartTask() 开始 AO 生成任务；

（5） AO_GetStatus() 取得 AOStatus.bTaskDone 若等于 1 表示生成任务结束（或者调用 AO_WaitUntilTaskDone()）

（6） AO_StopTask() 停止 AO 生成任务；

（7） AO_ReleaseTask() 释放 AO 生成任务；

（8） DEV_Release() 释放设备句柄。

如果在采集参数相同的情况下，多次捕捉触发事件输出多个片断的波形数据，可以在 3、4、5、6 步间循环进行，即可实现高速多次跟踪多个触发事件；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。请参考看下面流程图 2-8-1。

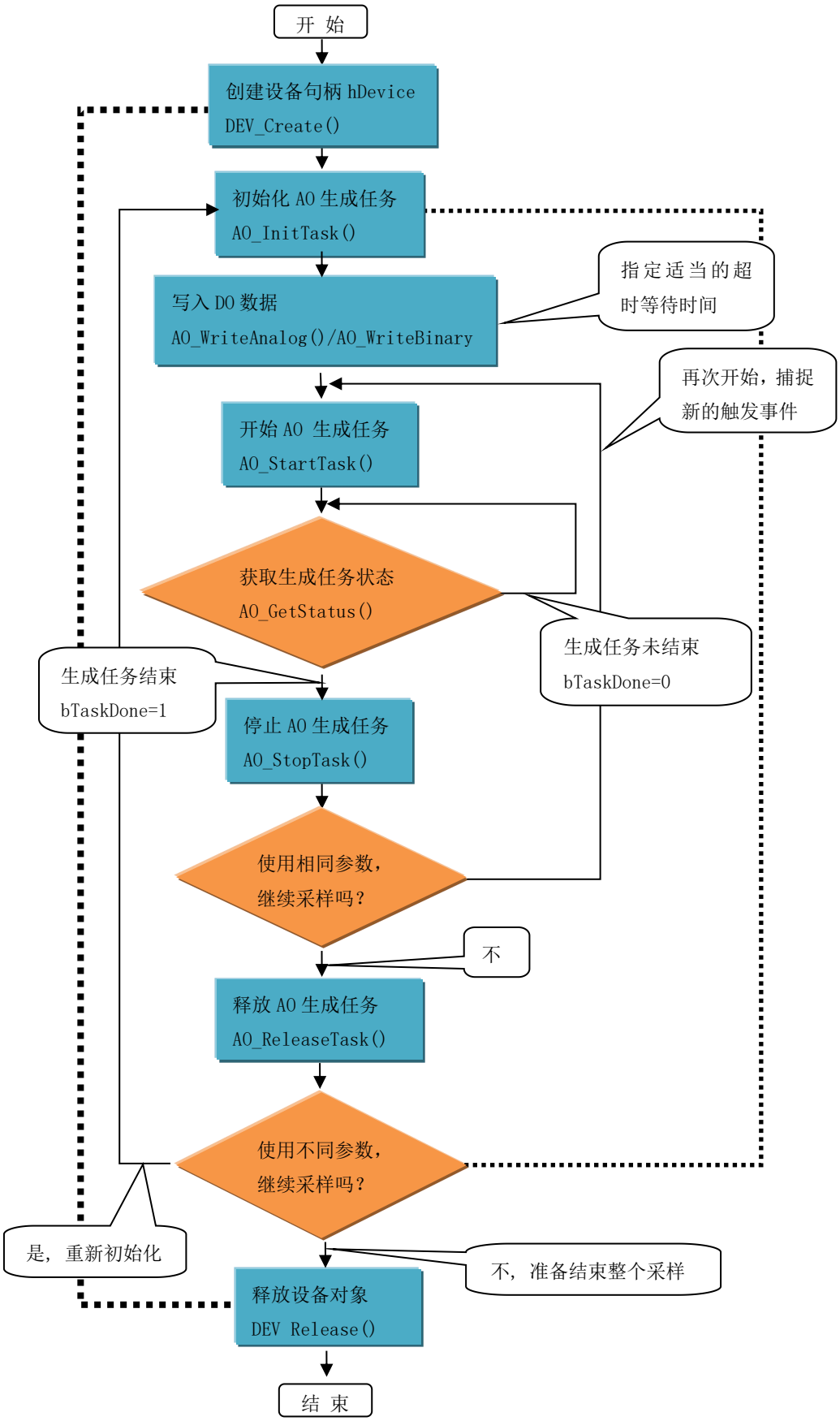


图 2-8-1 AO 有限点采样流程图例（AO_GetStatus()同步）

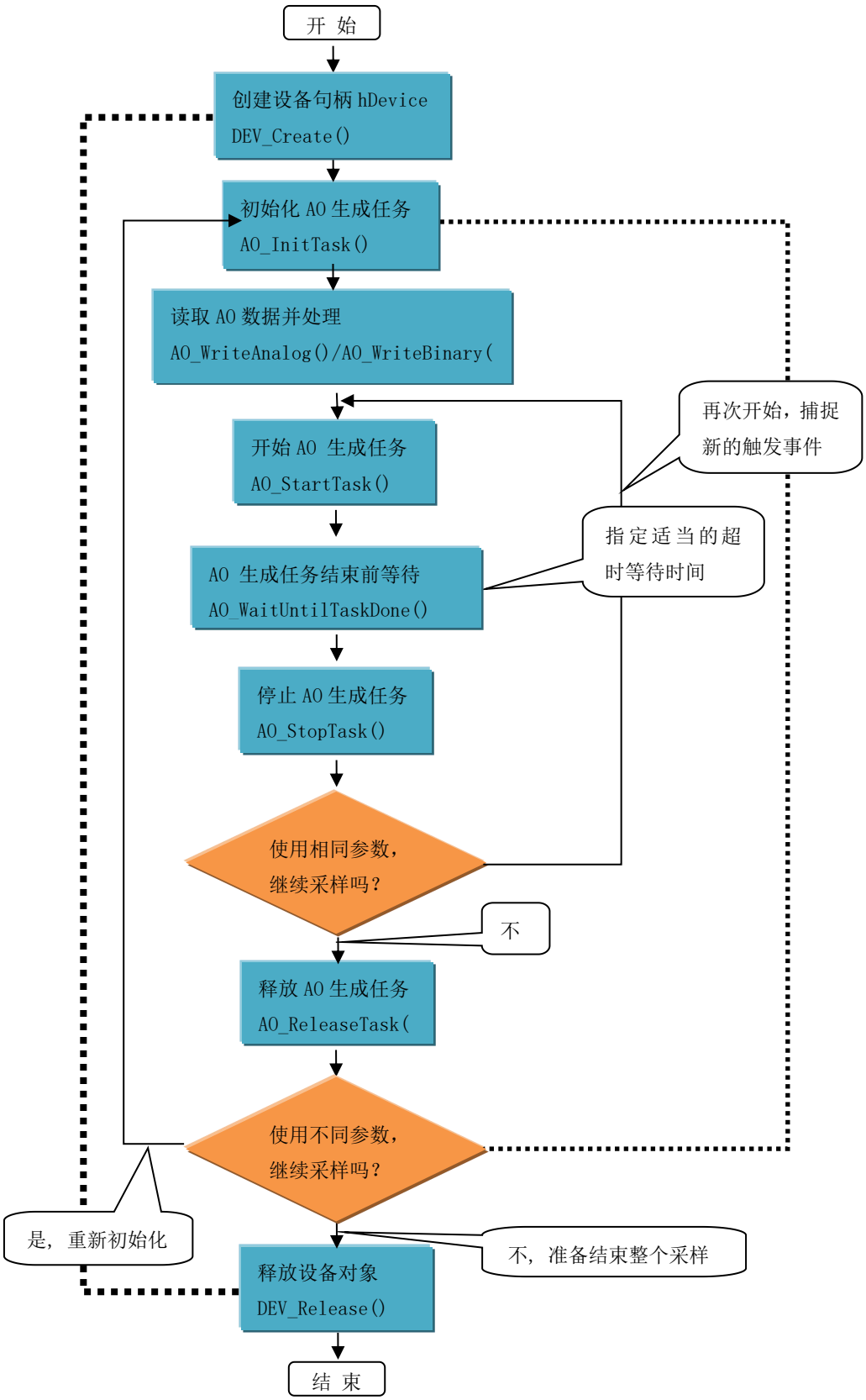


图 2-8-2　AO 有限点采样流程图例（AO_WaitUntilTaskDone ()同步）

2.9 AO 连续采样模式

连续采样模式，就是在一次开始后，AO 按照设定的采样速率，触发条件进行长时间的，无限点的连续不间断采样，设备永远不会自动停止（除非用户手动强制停止）。且在采样过程中可以实时改变输出波形数据。

AO 连续采样流程(重生成模式):

(1) DEV_Create() 创建设备句柄；

(2) AO_InitTask() 初始化 AO 生成任务，参数 nSampleMode=3；bRegenModeEn=TRUE；

(3) AO_WriteAnalog()或者 AO_WriteBinary() 写入 AO 各采样通道波形数据；

(4) AO_StartTask() 开始 AO 生成任务；

(5) AO_GetStatus() 取得 AO 生成任务的各种状态或者处理其他事务（生成任务后台输出连续数据）

(6) AO_StopTask() 停止 AO 生成任务；

(7) AO_ReleaseTask() 释放 AO 生成任务；

(8) DEV_Release() 释放设备句柄。

如果每次采集参数相同的情况下，可以在 5 步跟踪生成任务状态或处理相关事务，生成任务在后台连续不间断输出数据；

如果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 之间循环进行；

如果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。请参考看下面流程图 2-9-1。

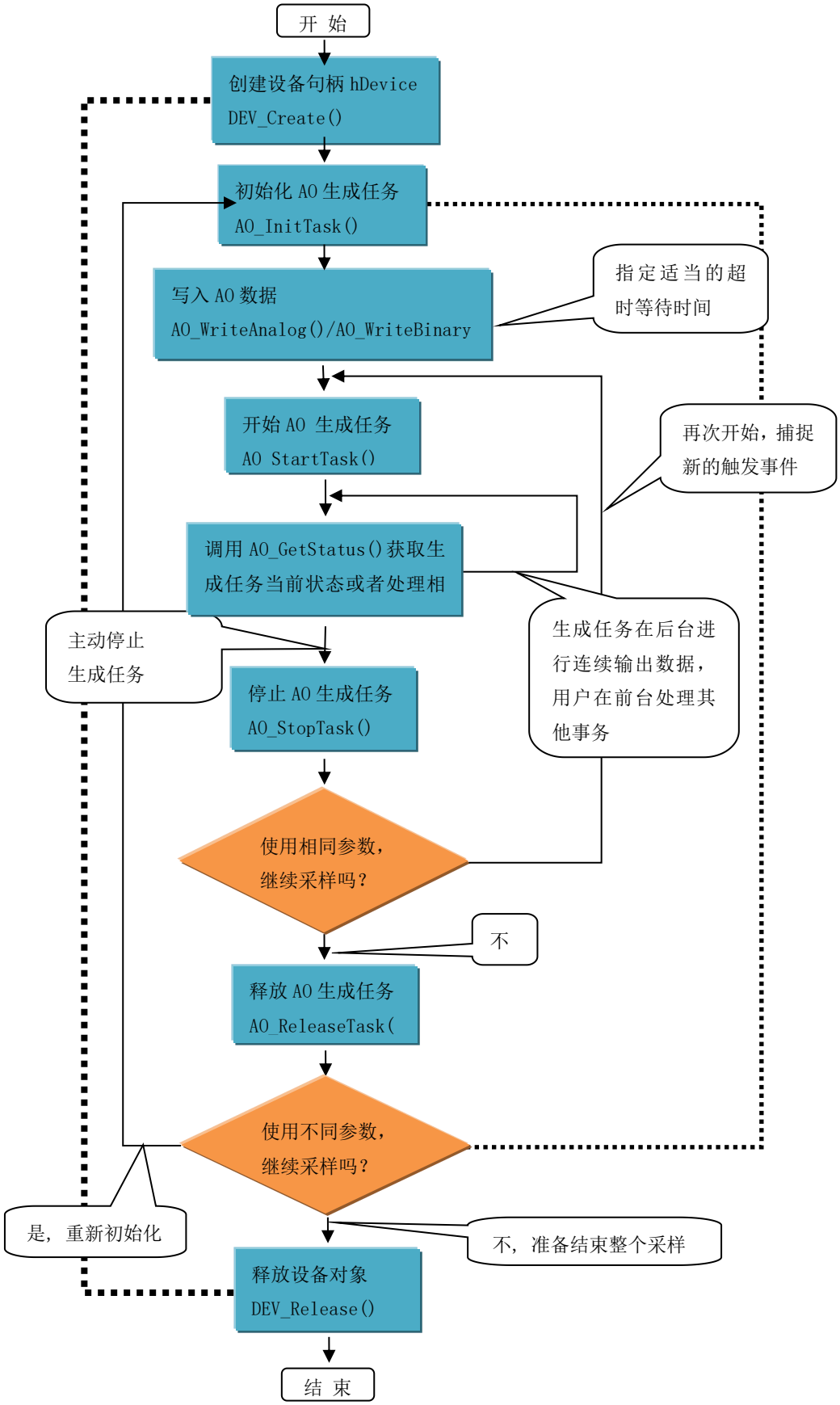


图 2-9-1　　AO 连续采样流程图例（重生成模式）

AO 连续采样流程(非重生成模式):

(1) DEV_Create() 创建设备句柄；

(2) AO_InitTask() 初始化 AO 生成任务，参数 nSampleMode=3；bRegenModeEn=FALSE;

(3) AO_WriteAnalog()或者 AO_WriteBinary() 写入 AO 各采样通道波形数据；

(4) AO_StartTask() 开始 AO 生成任务；

(5) AO_WriteAnalog()或者 AO_WriteBinary() 及时连续写入后续波形数据到生成任务中

(6) AO_StopTask() 停止 AO 生成任务；

(7) AO_ReleaseTask() 释放 AO 生成任务；

(8) DEV_Release() 释放设备句柄。

如果每次采集参数相同的情况下，可以在 5 步及时连续写入后续波形数据到生成任务中（注意要及时迅速，否则会出现缓冲下溢的风险，从而造成断波现象）；

如果是在参数不变的情况下需要捕获新的触发时，可以在 3、4、5、6 之间循环进行；

如果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。请参考看下面流程图 2-9-2。

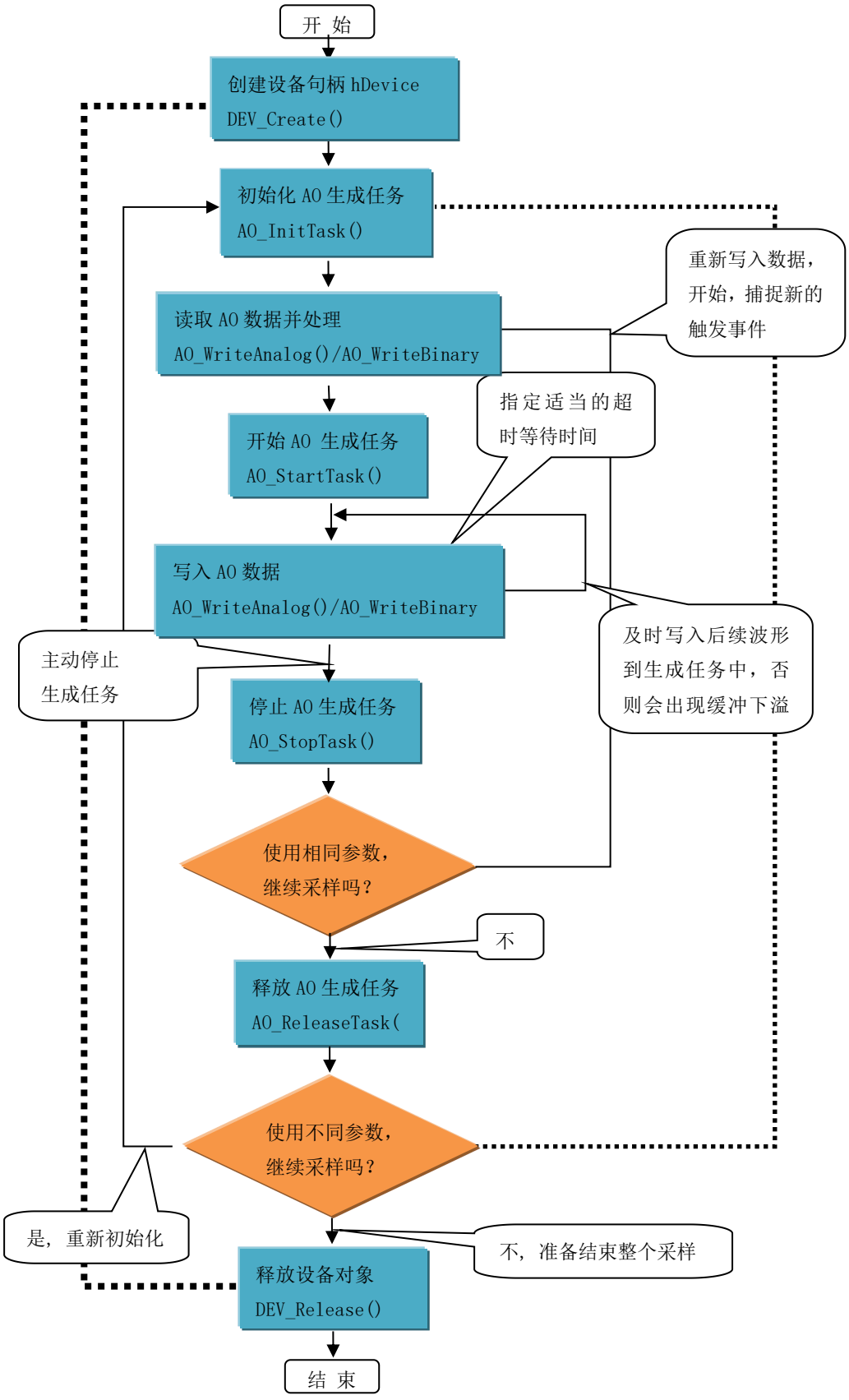


图 2-9-2　　AO 连续采样流程图例（非重生成模式）

上面图 2-7-1、图 2-8-1、图 2-9-1 中虚线表示对称关系。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 AO_InitTask()和 AO_ReleaseTask()两个函数的对称关系（即只有在 AO_ReleaseTask()之后才能再次 AO_InitTask()）。

2.10 DI 单点采样模式

单点采样，就是在一次开始后，用户每次发出读命令 DI_ReadDigitalOneU8()或 DI_ReadDigitalOneLines()时 DI 以设备最快速度获取各采样通道单个点的数据。具体流程如图 2.10 所示：

（1） DEV_Create() 创建设备句柄；

（2） DI_InitTask() 初始化 DI 采集任务，参数 nSampleMode=0；

（3） DI_StartTask() 开始 DI 采集任务；

（4） DI_ReadDigitalOneU8()或者 DI_ReadDigitalOneLines()读取 DI 单点数据；

（5） DI_StopTask() 停止 DI 采集任务；

（6） DI_ReleaseTask() 释放 DI 采集任务；

（7） DEV_Release() 释放设备句柄。

如果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；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。

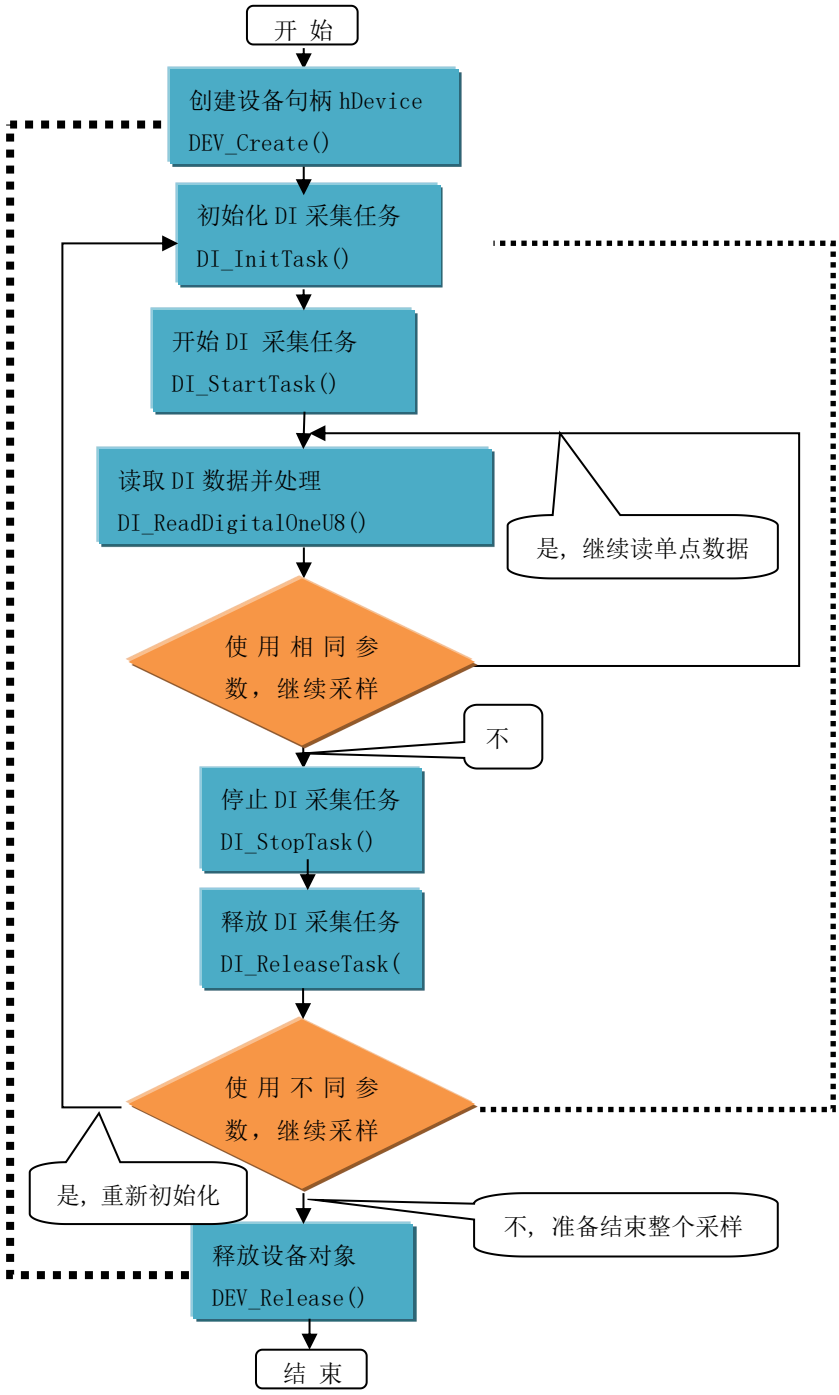


图 2.10　DI 实时单点采样流程图例

2.11　DI 有限点采样模式

有限点采样模式，就是在一次开始后，DI 按照设定的采样速率，触发条件进行指定时间的或指定点数的连续采样，达到指定点数后设备会自动停止。且在采样结束后才可以读取到完整的数据片断。具体流程如图 2.11 所示：

(1) DEV_Create()　创建设备句柄；

(2) DI_InitTask()　初始化 DI 采集任务，参数 nSampleMode=2；

(3) DI_StartTask()　开始 DI 采集任务；

(4) DI_ReadDigitalU8()读取 DI 批量数据；

(5) DI_StopTask()　停止 DI 采集任务；

(6) DI_ReleaseTask() 释放 DI 采集任务；

(7) DEV_Release()　释放设备句柄。

如果在采集参数相同的情况下，多次捕捉触发事件采样多个片断的数据，可以在 3、4、5 步间循环进行，即可实现高速多次跟踪多个触发事件；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。

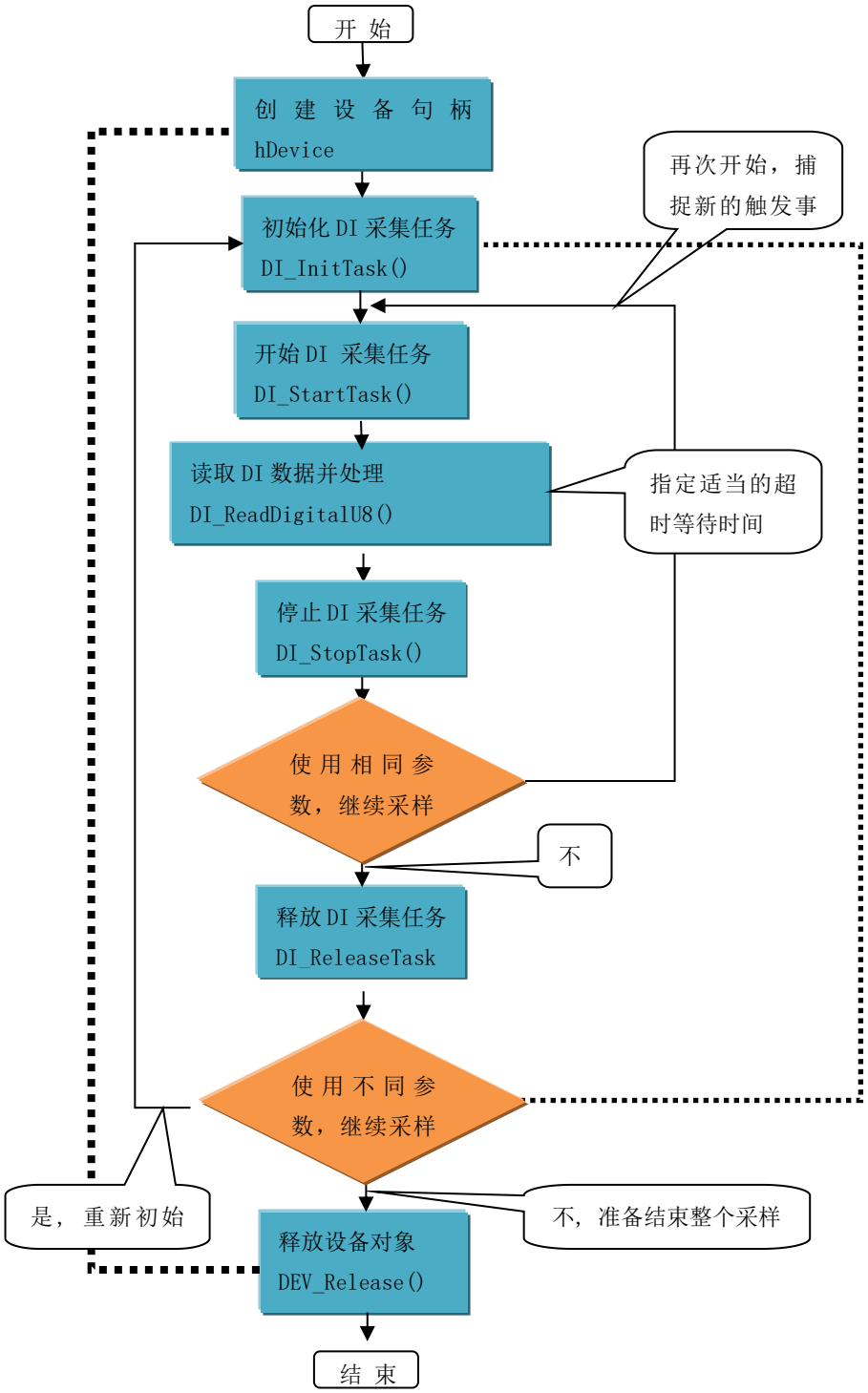


图 2.11　　DI 有限点采样流程图例

2.12 DI 连续采样模式

连续采样模式，就是在一次开始后，DI 按照设定的采样速率，触发条件进行长时间的，无限点的连续不间断采样，设备永远不会自动停止（除非用户手动强制停止）。且在采样过程中可以实时读取采样的连续数据。具体流程如图 2.12 所示：

(1) DEV_Create() 创建设备句柄；

(2) DI_InitTask() 初始化 DI 采集任务，参数 nSampleMode=3；

(3) DI_StartTask() 开始 DI 采集任务；

(4) DI_ReadDigitalU8()读取 DI 批量数据；

(5) DI_StopTask() 停止 DI 采集任务；

(6) DI_ReleaseTask() 释放 DI 采集任务；

(7) DEV_Release() 释放设备句柄。

如果每次采集参数相同的情况下，可以在 4 步间循环进行，即可实现高速连续不间断采样；

如果是在参数不变的情况下需要捕获新的触发时，可以在 3、4、5 之间循环进行；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。请参考看下面流程图 2.12。

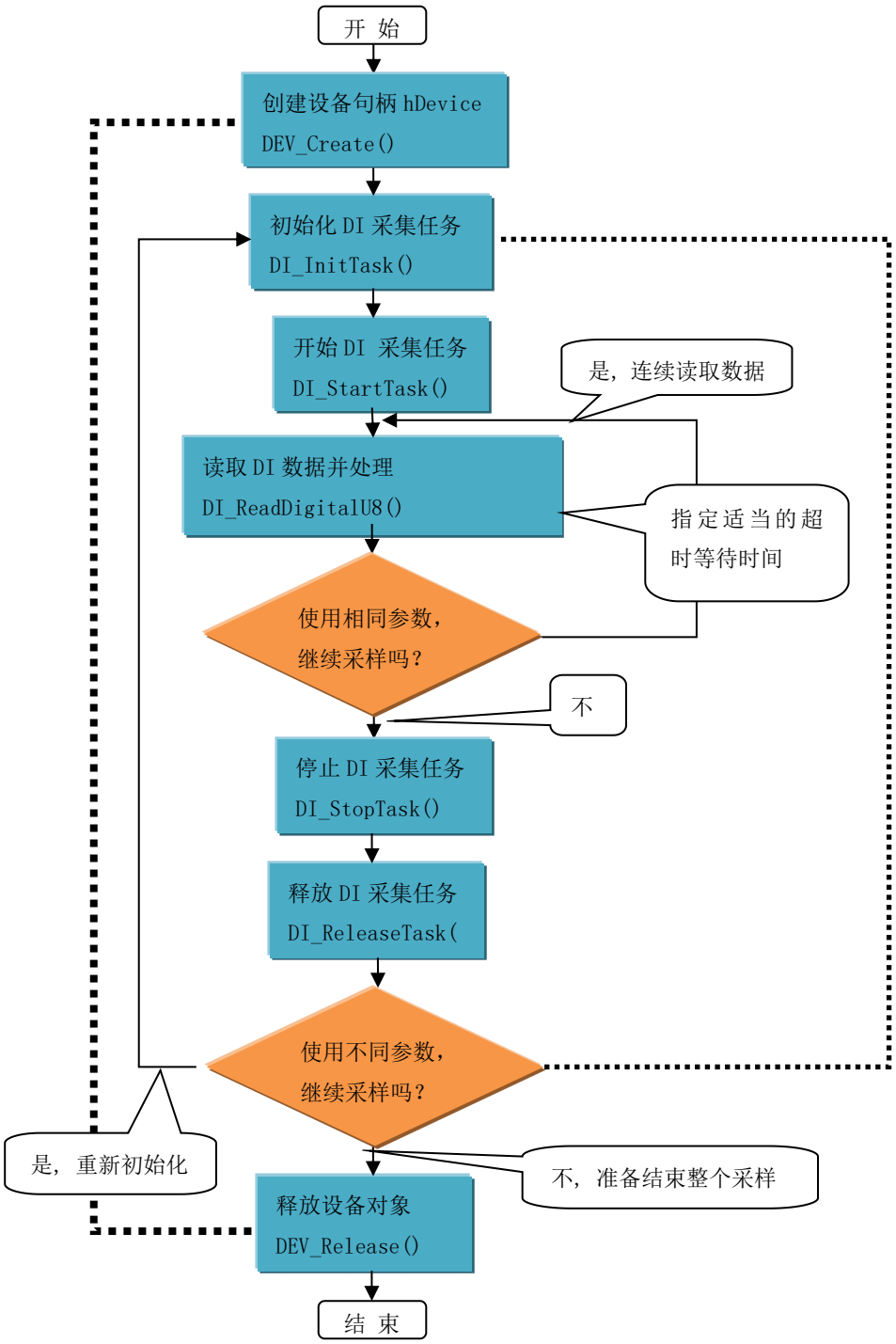


图 2.12　　DI 连续采样流程图例

上图虚线表示对称关系。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。DI_InitTask()和 DI_ReleaseTask()两个函数的对称关系是只有在 DI_ReleaseTask()之后才能再次 DI_InitTask()。

2.13　DO 单点采样模式

单点采样，就是在一次开始后，用户每次发出写命令 DO_WriteDigitalOneU32()或

DO_WriteDigitalOneLines()/DO_WriteDigitalOneLine()时 DO 以设备最快速度写入各采样通道单个点的数据。具体步骤如下：

(1) DEV_Create()　创建设备句柄；

(2) DO_InitTask()　初始化 DO 生成任务，参数 nSampleMode=0；

(3) DO_StartTask()　开始 DO 生成任务；

(4) DO_WriteDigitalOneU32()或 DO_WriteDigitalOneLines()/DO_WriteDigitalOneLine()　写入 DO 各采样通道单点数据；

(5) DO_StopTask()　停止 DO 生成任务；

(6) DO_ReleaseTask()　释放 DO 生成任务；

(7) DEV_Release()　释放设备句柄。

如果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；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。

具体执行流程请看下面的图 2.13。

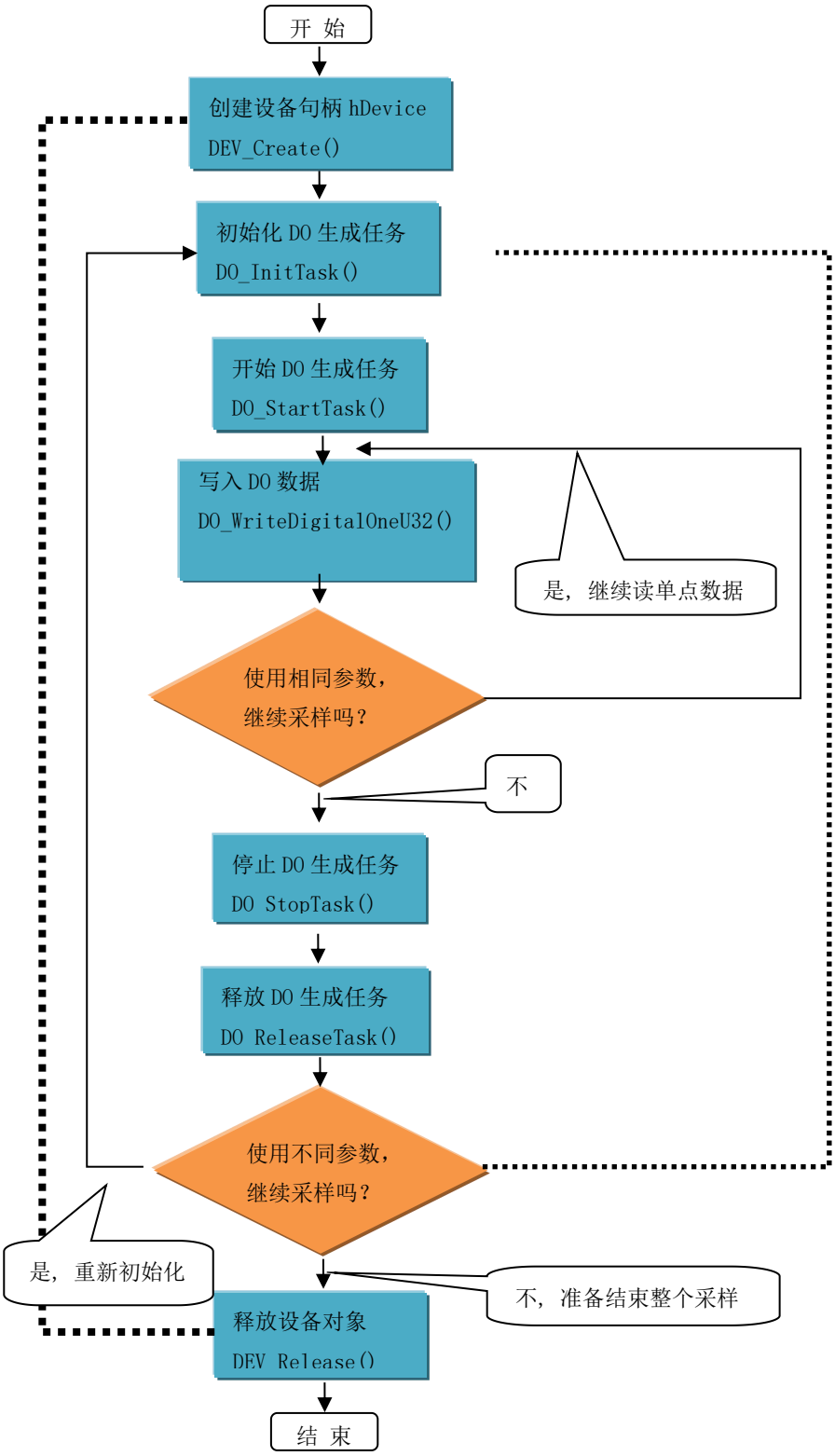


图 2.13　DO 实时单点采样流程图例

2.14 DO 有限点采样模式

有限点采样模式，就是在开始前，先写入指定点数的数据到任务中，开始后，AO 按照设定的采样速率，触发条件进行指定时间的或指定点数的连续采样，达到指定点数后设备会自动停止。且在采样结束后才可以写入下一批数据到任务中以待开始采样。

DO 有限点采样流程:

(1) DEV_Create() 创建设备句柄；

(2) DO_InitTask() 初始化 DO 生成任务，参数 nSampleMode=2；

(3) DO_WriteDigitalU8() 写入 DO 批量数据；

(4) DO_StartTask() 开始 DO 生成任务；

(5) DO_GetStatus() 取得 DOStatus.bTaskDone 若等于 1 表示生成任务结束（或者调用 DO_WaitUntilTaskDone()）

(6) DO_StopTask() 停止 DO 生成任务；

(7) DO_ReleaseTask() 释放 DO 生成任务；

（8）DEV_Release() 释放设备句柄。

如果在采集参数相同的情况下，多次捕捉触发事件输出多个片断的波形数据，可以在 3、4、5、6 步间循环进行，即可实现高速多次跟踪多个触发事件；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。请参考看下面流程图 2.14。

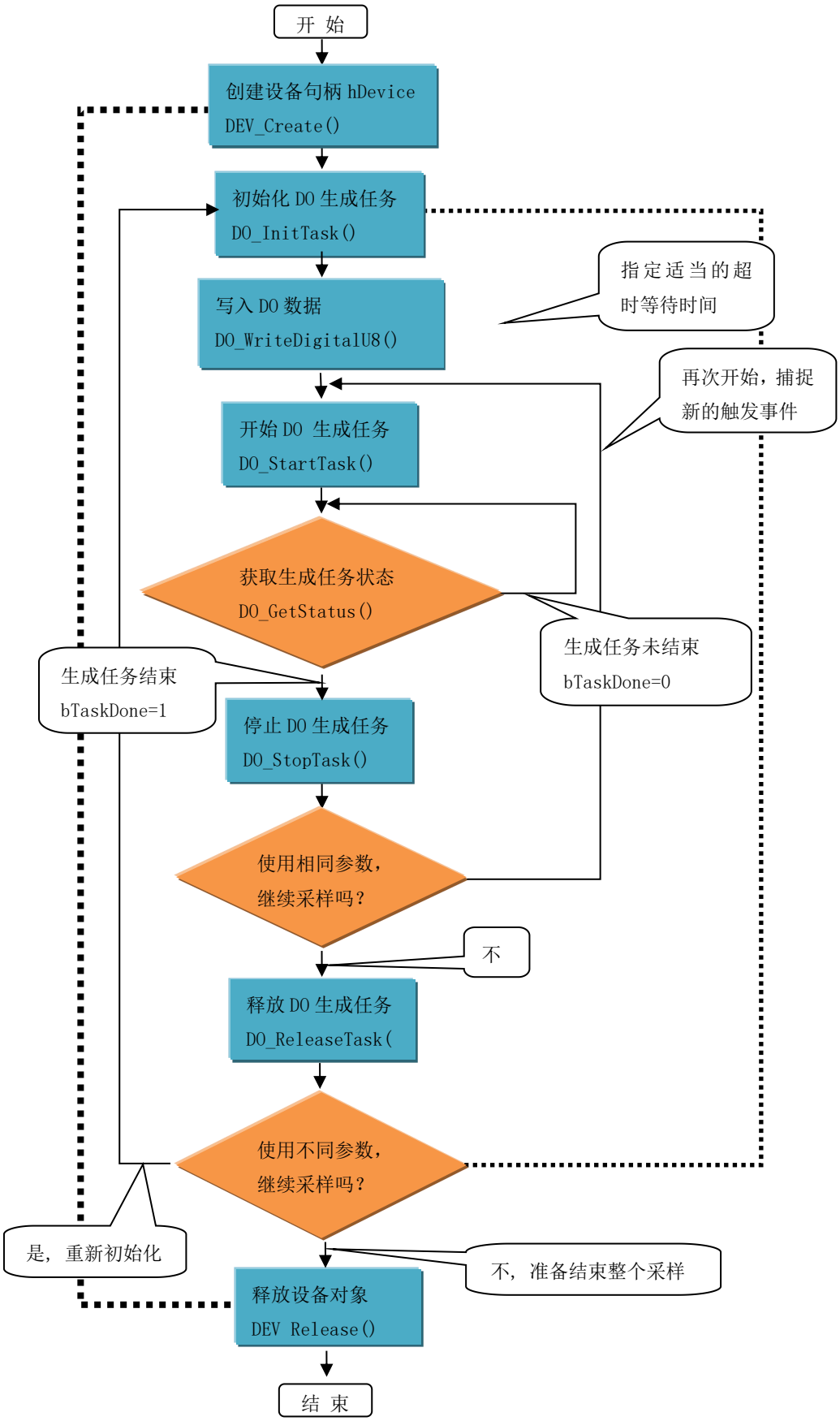


图 2.14　DO 有限点采样流程图例（DO_GetStatus()同步）

2.15 DO 连续采样模式

连续采样模式，就是在一次开始后，DO 按照设定的采样速率，触发条件进行长时间的，无限点的连续不间断采样，设备永远不会自动停止（除非用户手动强制停止）。且在采样过程中可以实时改变输出波形数据。

DO 连续采样流程(重生成模式):

(1) DEV_Create() 创建设备句柄；

(2) DO_InitTask() 初始化 DO 生成任务，参数 nSampleMode=3；bRegenModeEn=TRUE；

(3) DO_WriteDigitalU8()写入 DO 批量数据；

(4) DO_StartTask() 开始 DO 生成任务；

(5) DO_GetStatus() 取得 DO 生成任务的各种状态或者处理其他事务

(6) DO_StopTask()停止 DO 生成任务；

(7) DO_ReleaseTask() 释放 DO 生成任务；

(8) DEV_Release() 释放设备句柄。

如果每次采集参数相同的情况下，可以在 5 步跟踪生成任务状态或处理相关事务，生成任务在后台连续不间断输出数据；

如果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 之间循环进行；

如果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。请参考看下面流程图 2.15.1。

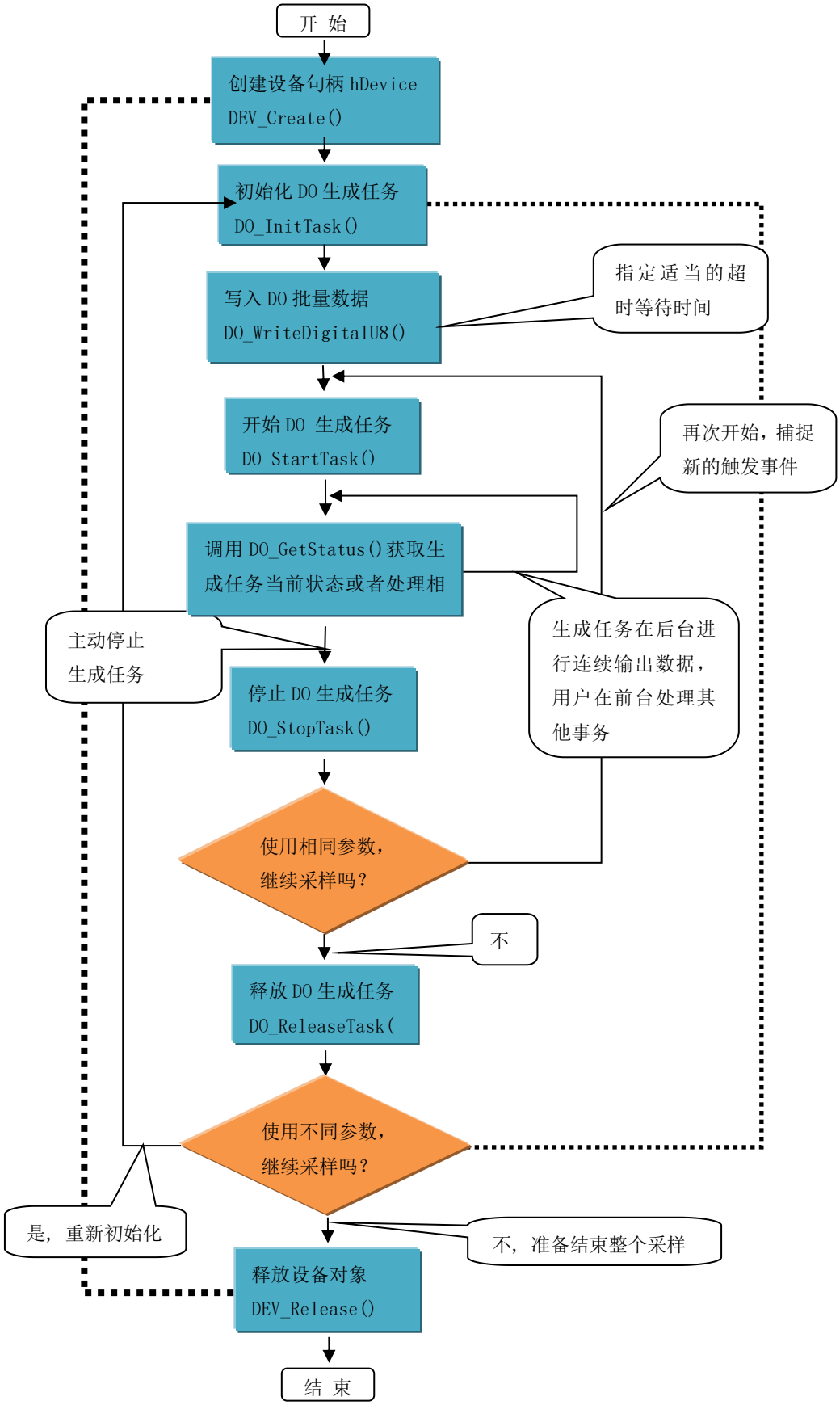


图 2.15.1　　DO 连续采样流程图例（重生成模式）

DO 连续采样流程(非重生成模式):

（1）DEV_Create()　创建设备句柄；

（2）DO_InitTask()　初始化 DO 生成任务，参数 nSampleMode=3；bRegenModeEn=FALSE;

（3）DO_WriteDigitalU8() 写入 DO 批量数据；

（4）DO_StartTask() 开始 DO 生成任务；

（3）DO_WriteDigitalU8() 及时连续写入后续批量数据到生成任务中

（4）DO_StopTask()　停止 DO 生成任务；

（5）DO_ReleaseTask() 释放 DO 生成任务；

（6）DEV_Release()　释放设备句柄。

如果每次采集参数相同的情况下，可以在 5 步及时连续写入后续波形数据到生成任务中（注意要及时迅速，否则会出现缓冲下溢的风险，从而造成断波现象）；

如果是在参数不变的情况下需要捕获新的触发时，可以在 3、4、5、6 之间循环进行；

如果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。请参考看下面流程图 2.15.2。

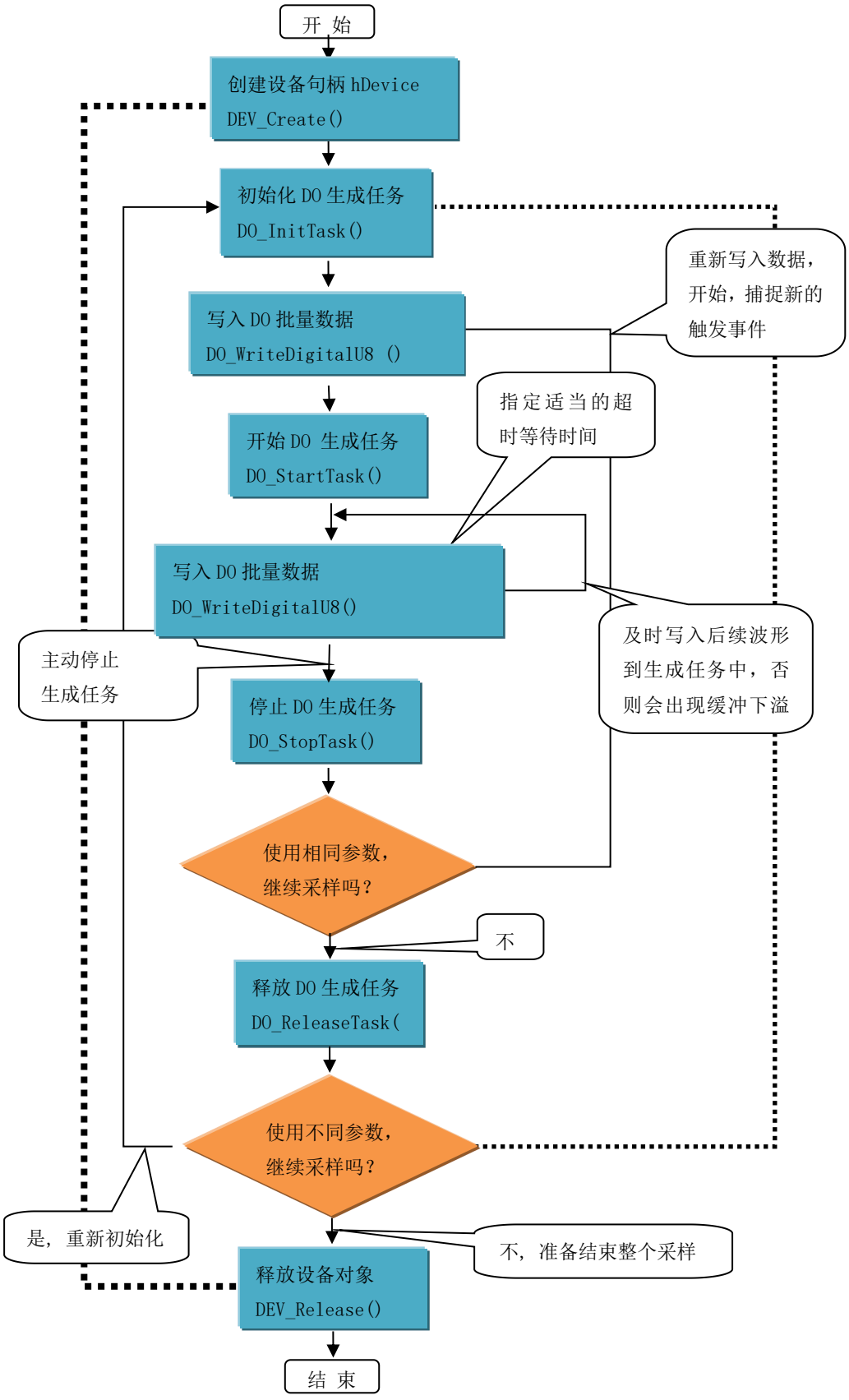


图 2.15.2　　DO 连续采样流程图例（非重生成模式）

上面图 2.13、图 2.14、图 2.15 中虚线表示对称关系。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 DO_InitTask()和 DO_ReleaseTask()两个函数的对称关系（即只有在 DO_ReleaseTask()之后才能再次 DO_InitTask()）。

2.16 CI 单点采样模式

单点采样，就是在一次开始后，用户每次发出读命令 CI_ReadCounterScalarU32()或 CI_ReadCounterScalarF64()时 CI 以设备最快速度获取单个点的数据。现以频率测量为例，具体流程如图 2.16 所示：

(1) DEV_Create()　创建设备句柄；

(2) CI_CreateFreqChan()　创建频率测量通道；

(3) CTR_StartTask()　开始 CI 采集任务；

(4) CI_ReadCounterScalarU32()或者 CI_ReadCounterScalarF64()读取 CI 单点数据；

(5) CTR_StopTask()　　停止 CI 采集任务；

(6) CTR_ReleaseTask()　释放 CI 采集任务；

(7) DEV_Release()　释放设备句柄。

如果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；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。

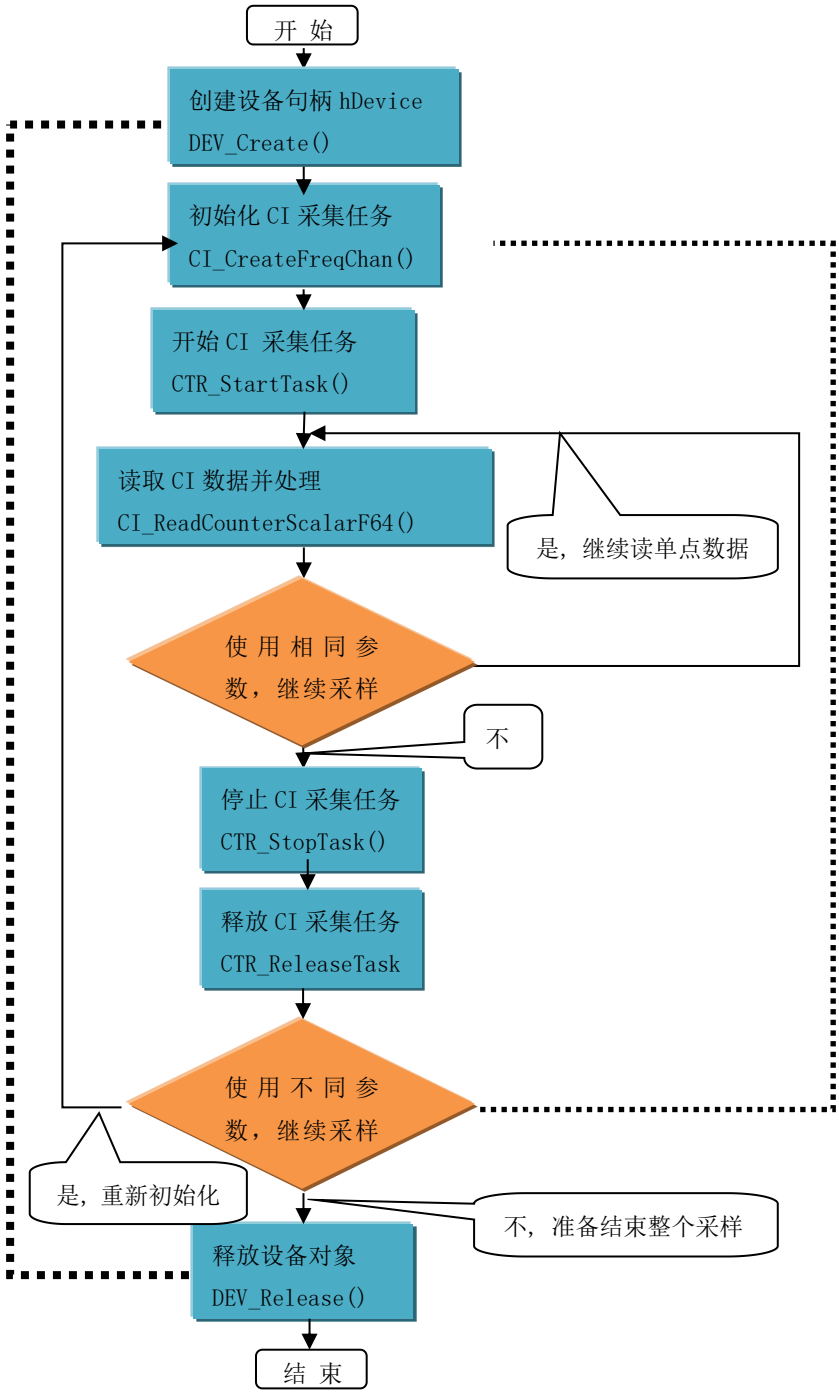


图 2.16　CI 实时单点采样流程图例

2.17　CI 有限点采样模式

有限点采样模式，就是在一次开始后，CI 按照设定的采样速率，触发条件进行指定时间的或指定点数的连续采样，达到指定点数后设备会自动停止。且在采样结束后才可以读取到完整的数据片断。具体流程如图 2.17 所示：

（1） DEV_Create()　创建设备句柄；

（2） CI_CreateCountEdgesChan()　创建边沿计数测量通道；

（3） CTR_CfgSampClkTiming()　配置采样速率，时钟源，采样模式=2

（4） CTR_StartTask()　开始 CI 采集任务；

（5） CI_ReadCounterU32() 读取 CI 批量数据；
（6） CTR_StopTask() 停止 CI 采集任务；
（7） CTR_ReleaseTask() 释放 CI 采集任务；
（8） DEV_Release() 释放设备句柄。

如果在采集参数相同的情况下，多次捕捉触发事件采样多个片断的数据，可以在 4、5、6 步间循环进行，即可实现高速多次跟踪多个触发事件；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。

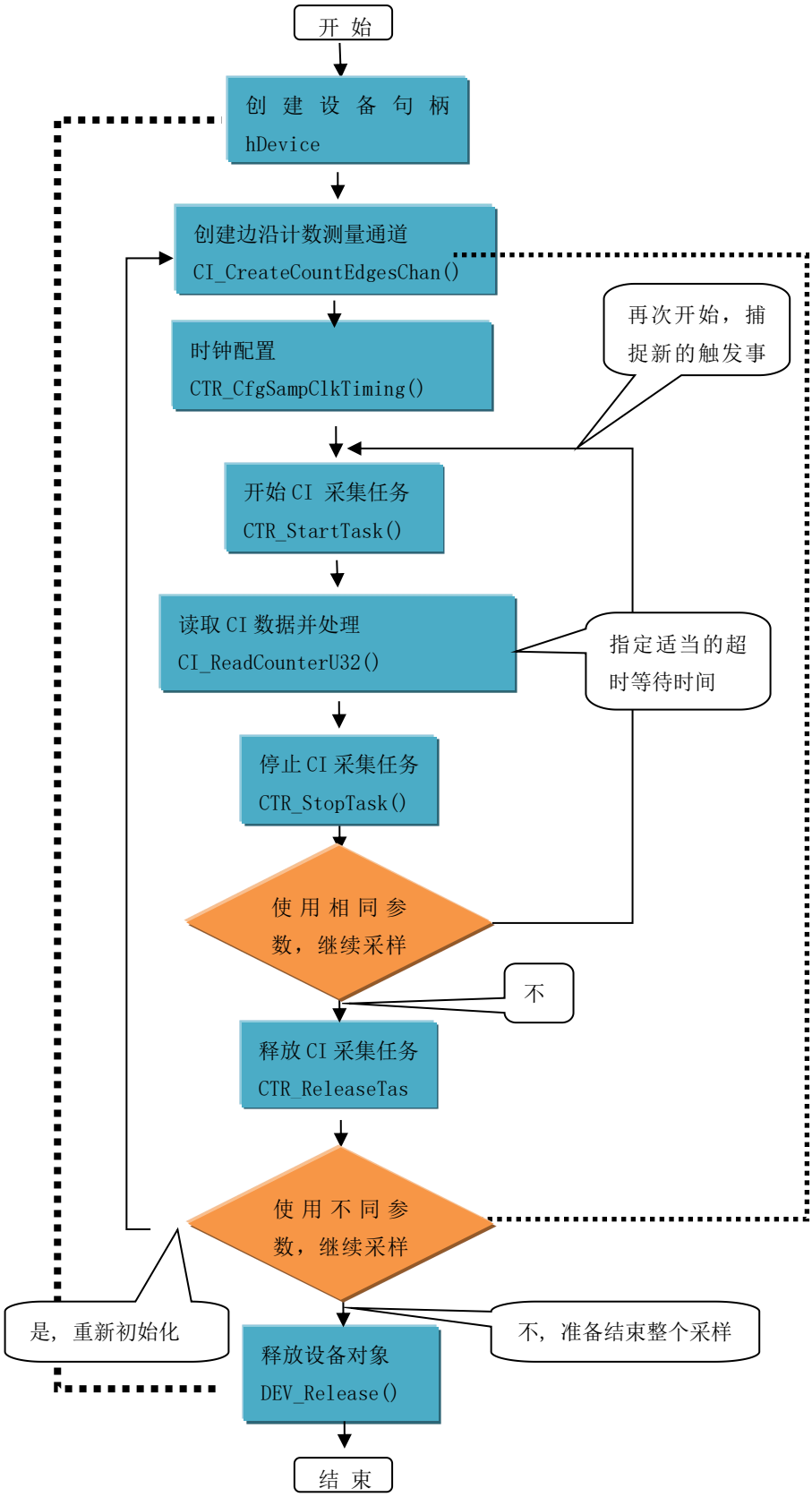


图 2.17　　CI 有限点采样流程图例

2.18 CI 连续采样模式

连续采样模式，就是在一次开始后，CI 按照设定的采样速率，触发条件进行长时间的，无限点的连续不间断采样，设备永远不会自动停止（除非用户手动强制停止）。且在采样过程中可以实时读取采样的连续数据。具体流程如图 2.18 所示：

(1) DEV_Create()　创建设备句柄；

(2) CI_CreateCountEdgesChan()　创建边沿计数测量通道；

(3) CTR_CfgSampClkTiming() 配置采样速率，时钟源，采样模式=2

(4) CTR_StartTask()　开始 CI 采集任务；

(5) CI_ReadCounterU32() 读取 CI 批量数据(循环连续读取)；

(6) CTR_StopTask()　　停止 CI 采集任务；

(7) CTR_ReleaseTask() 释放 CI 采集任务；

(8) DEV_Release()　释放设备句柄。

如果每次采集参数相同的情况下，可以在 5 步间循环进行，即可实现高速连续不间断采样；

如果是在参数不变的情况下需要捕获新的触发时，可以在 3、4、5、6 之间循环进行；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。请参考看下面流程图 2-6-1。

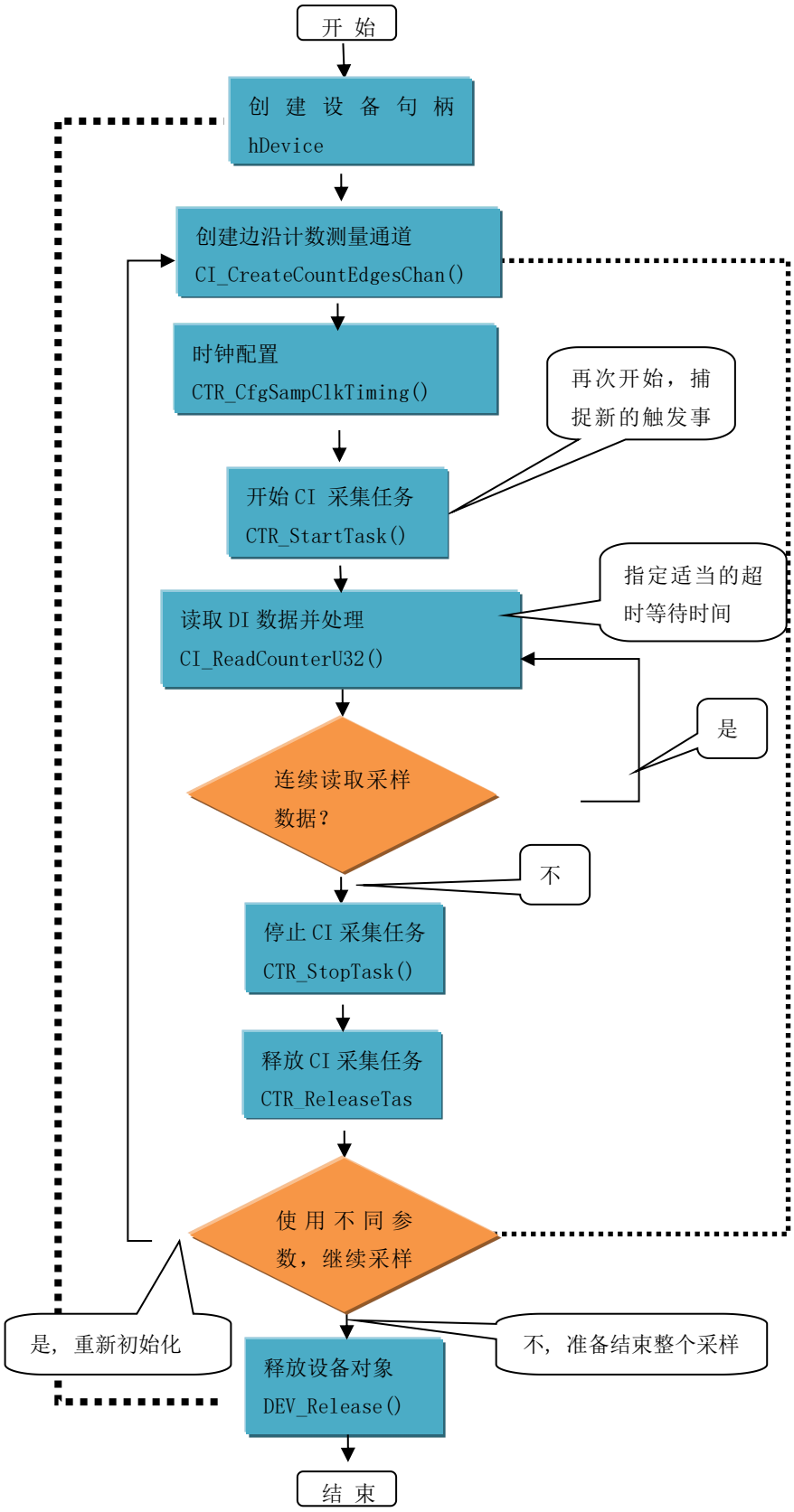


图 2.18　　CI 连续采样流程图例

上图虚线表示对称关系。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。CI_CreateFreqChan()等 CI 测量通道创建函数和 CTR_ReleaseTask()两个函数的对称关系是只有在 CTR_ReleaseTask()之后才能再次 CI_CreateFreqChan()等 CI 测量通道创建函数。

2.19 CO 单点采样模式

单点采样，就是单个固定脉冲输出。现基于时间的脉冲输出为例，具体流程如图 2.19 所示：

（1） DEV_Create() 创建设备句柄；

（2） CO_CreatePulseChanTime() 创建脉冲时间生成通道；

（3） CTR_StartTask() 开始 CO 生成任务；

（4）CTR_WaitUntilTaskDone() 等待生成任务结束；

（5）CTR_StopTask() 停止 CO 生成任务；

（6）CTR_ReleaseTask() 释放 CO 生成任务；

（7）DEV_Release() 释放设备句柄。

如果每次采集参数相同的情况下，可以在第 3、4、5 步处循环进行，即可实现单点实时循环采样；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。

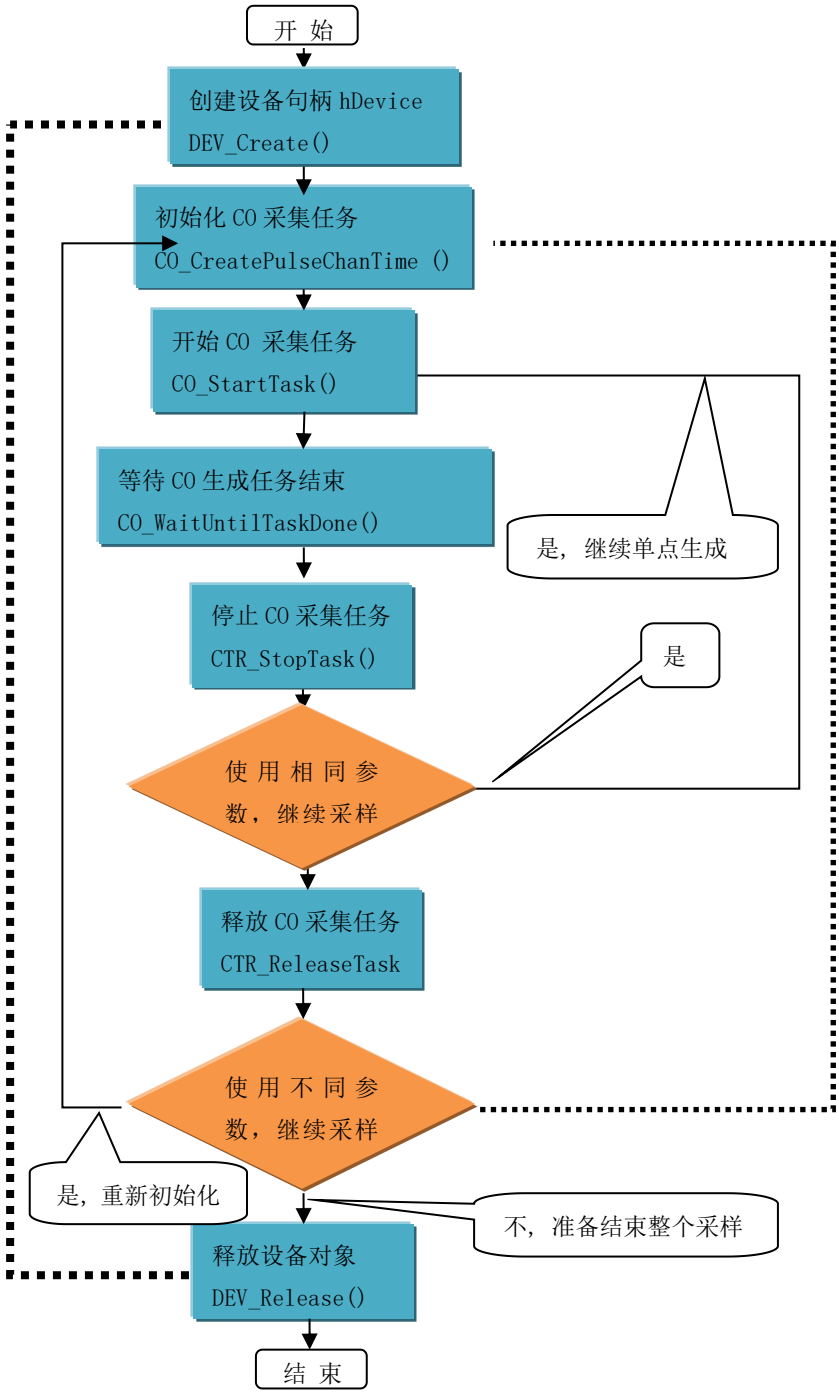


图 2.19　CO 实时单点采样流程图例

2.20　CO 有限点采样模式

有限点采样模式，就是在开始任务后，CO 按照设定的采样速率，触发条件进行指定时间的或指定点数的连续采样，达到指定点数后设备会自动停止。根据在采样过程中是否需要写入新的采样数据又分为非重生成模式和重生成模式，由配置函数 CO_SetWriteRegenMode()指定，当不调用 CO_SetWriteRegenMode()时默认为可重生成模式。

创建脉冲输出通道的方式有：

（1） CO_CreatePulseChanFreq() 创建脉冲频率生成通道，以频率和占空比指定脉冲

（2） CO_CreatePulseChanTime() 创建脉冲时间生成通道，以低电平和高电平的时间长度指定

脉冲

（3） CO_CreatePulseChanTicks() 创建脉冲滴答生成通道，以低电平和高电平的滴答计数指定脉冲。

CO 有限点采样模式可进一步分为固定脉冲输出与可变脉冲输出。在固定脉冲输出方式下，仅支持隐式时钟，不支持显式时钟；在可变脉冲输出方式下，支持隐式时钟，也支持显示时钟(但不支持内部时钟)。

2.20.1 固定脉冲(非缓冲)输出

固定脉冲也称为非缓冲输出，就是在启动任务前即创建脉冲输出通道时就确定了整个脉冲序列的频率、占空比等，即每个脉冲都使用一样的固定的脉冲宽度和占空比，在输出过程中不能改变每个脉冲的频率、宽度、占空比等。在固定脉冲输出方式下不支持非重生成模式。

现以低电平和高电平的时间长度指定脉冲为例，讲述有限、固定脉冲、重生成时的控制流程：

（1） DEV_Create() 创建设备句柄；

（2） CO_CreatePulseChanTime() 创建脉冲时间输出通道；

（3） CTR_CfgImplicitTiming() 配置隐式时钟（nSampleMode=2）；

（4） CTR_StartTask() 开始 CO 生成任务；

（5） CTR_WaitUntilTaskDone() 等待生成任务结束；

（6） CTR_StopTask() 停止 CO 生成任务；

（7） CTR_ReleaseTask() 释放 CO 生成任务；

（8） DEV_Release() 释放设备句柄。

如果在采集参数相同的情况下，多次捕捉触发事件生成多个片断的脉冲序列，可以在 4、5、6 步间循环进行，即可实现高速多次跟踪多个触发事件；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。如下图：

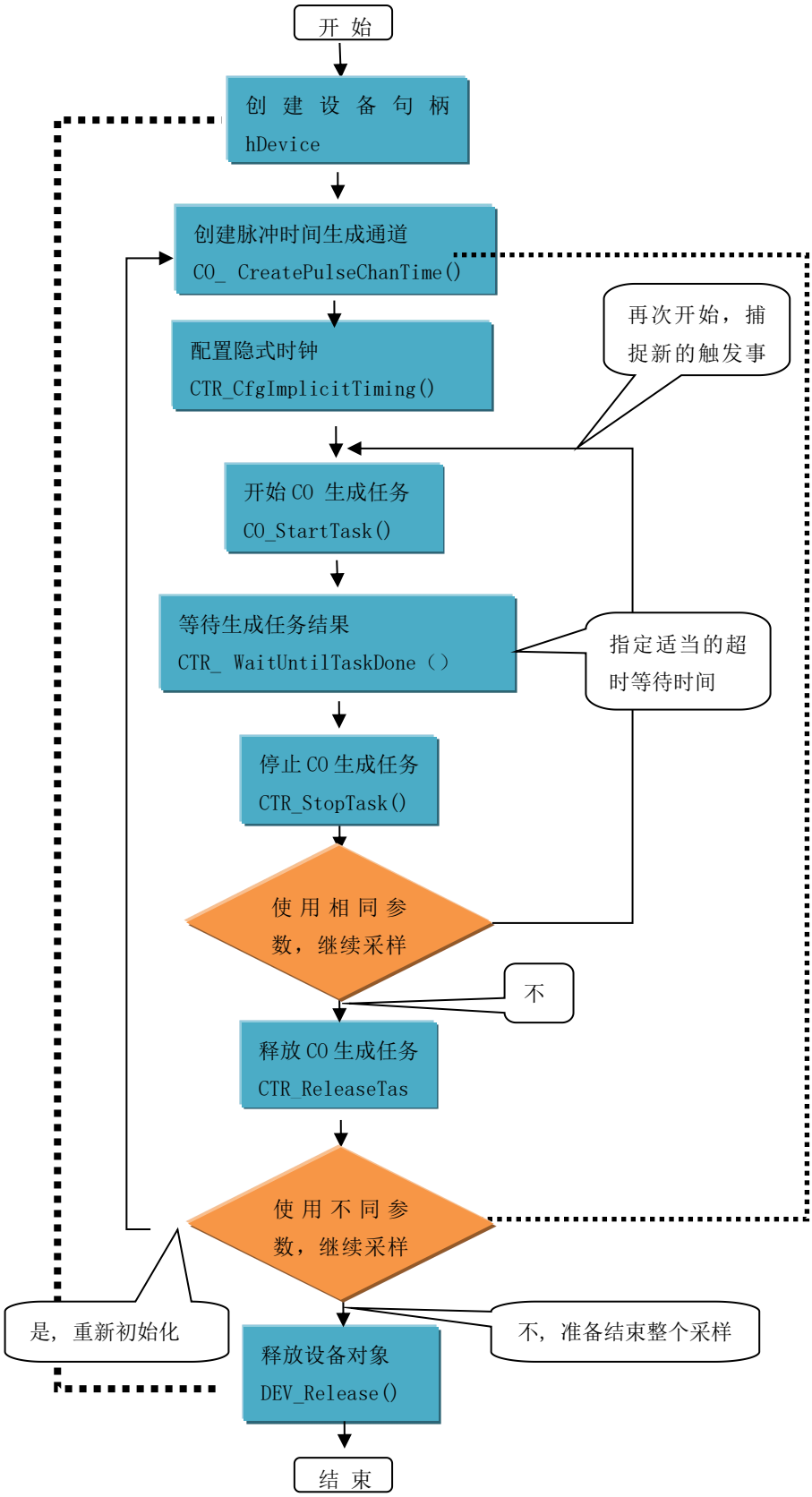


图 2.20.1　　CO 有限点采样、固定脉冲流程图例

2.20.2 可变脉冲(缓冲)输出

可变脉冲也称为缓冲输出，就是在启动任务前即创建脉冲输出通道时就不指整个脉冲序列的频率、占空比等，即每个脉冲宽度和占空比均由后面写入缓冲中的数据序列决定，在输出过程中每个脉冲的频率、宽度、占空比是可以不一样的。

现以低电平和高电平的时间长度指定脉冲为例，讲述有限、可变脉冲、重生成时的控制流程：

（1） DEV_Create() 创建设备句柄；

（2） CO_CreatePulseChanTime() 创建脉冲时间输出通道；

（3） CTR_CfgSampClkTiming() 配置显式时钟，不支持内时钟，nSampleMode=2

（4） CO_WriteTime() 写入脉冲序列数据(低电平和高电平时间长度值)

（5） CTR_StartTask() 开始 CO 生成任务；

（6） CTR_WaitUntilTaskDone() 等待生成任务结束；

（7） CTR_StopTask() 停止 CO 生成任务；

（8）CTR_ReleaseTask() 释放 CO 生成任务；

（9） DEV_Release() 释放设备句柄。

如果在采集参数相同的情况下，多次捕捉触发事件生成多个片断的脉冲序列，可以在 4、5、6 步间循环进行，即可实现高速多次跟踪多个触发事件；

如果每次采集参数有所不同，则可以在 2、3、4、5、6 步间重复进行。如下图：

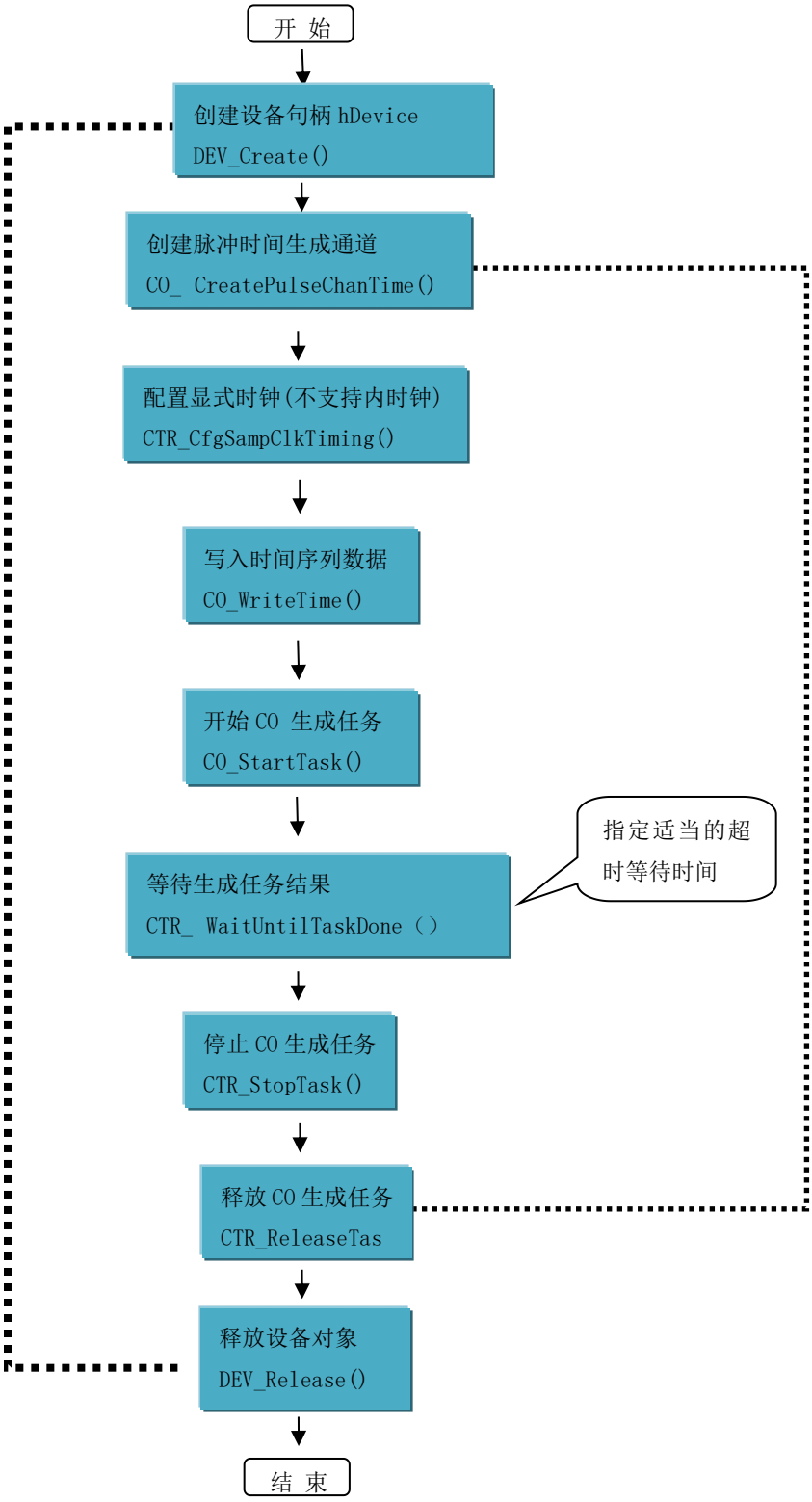


图 2.20.2　CO 有限点采样、可变脉冲流程图例

2.21 CO 连续采样模式

连续采样模式，就是在一次开始后，CO 按照设定的采样速率，触发条件进行长时间的，无限点的连续不间断采样，设备永远不会自动停止（除非用户手动强制停止）。根据在采样过程中是否需要写入新的采样数据又分为非重生成模式和重生成模式，由配置函数 CO_SetWriteRegenMode()指定，当不调用 CO_SetWriteRegenMode()时默认为可重生成模式。

创建脉冲输出通道的方式有：

（1） CO_CreatePulseChanFreq() 创建脉冲频率生成通道，以频率和占空比指定脉冲

（2） CO_CreatePulseChanTime() 创建脉冲时间生成通道，以低电平和高电平的时间长度指定脉冲

（3） CO_CreatePulseChanTicks() 创建脉冲滴答生成通道，以低电平和高电平的滴答计数指定脉冲。

CO 连续采样模式可进一步分为固定脉冲输出与可变脉冲输出两种方式。在固定脉冲输出方式下，仅 支持隐式时钟，不支持显式时钟；在可变脉冲输出方式下，支持隐式时钟，也支持显示时钟(但不支持内部时钟)。

2.21.1 固定脉冲(非缓冲)输出

固定脉冲也称为非缓冲输出，就是在启动任务前即创建脉冲输出通道时就确定了整个脉冲序列的频率、占空比等，即每个脉冲都使用一样的固定的脉冲宽度和占空比，在输出过程中不能改变每个脉冲的频率、宽度、占空比等。在固定脉冲输出方式下不支持非重生成模式。

现以低电平和高电平的时间长度指定脉冲为例，讲述连续、固定脉冲、重生成时的控制流程：

(1) DEV_Create() 创建设备句柄；

(2) CO_CreatePulseChanTime() 创建脉冲时间生成通道；

(3) CTR_CfgImplicitTiming() 配置隐式时钟，nSamleMode=3

(4) CTR_StartTask() 开始 CO 生成任务；

(5) CTR_WaitUntilTaskDone() 等待 CO 生成任务结束；

(6) CTR_StopTask() 停止 CO 生成任务；

(7) CTR_ReleaseTask() 释放 CO 生成任务；

(8) DEV_Release() 释放设备句柄。

如果每次采集参数相同的情况下，可以在 4、5、6 步间循环进行，即可实现高速连续不间断采样；

如果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。如下图：

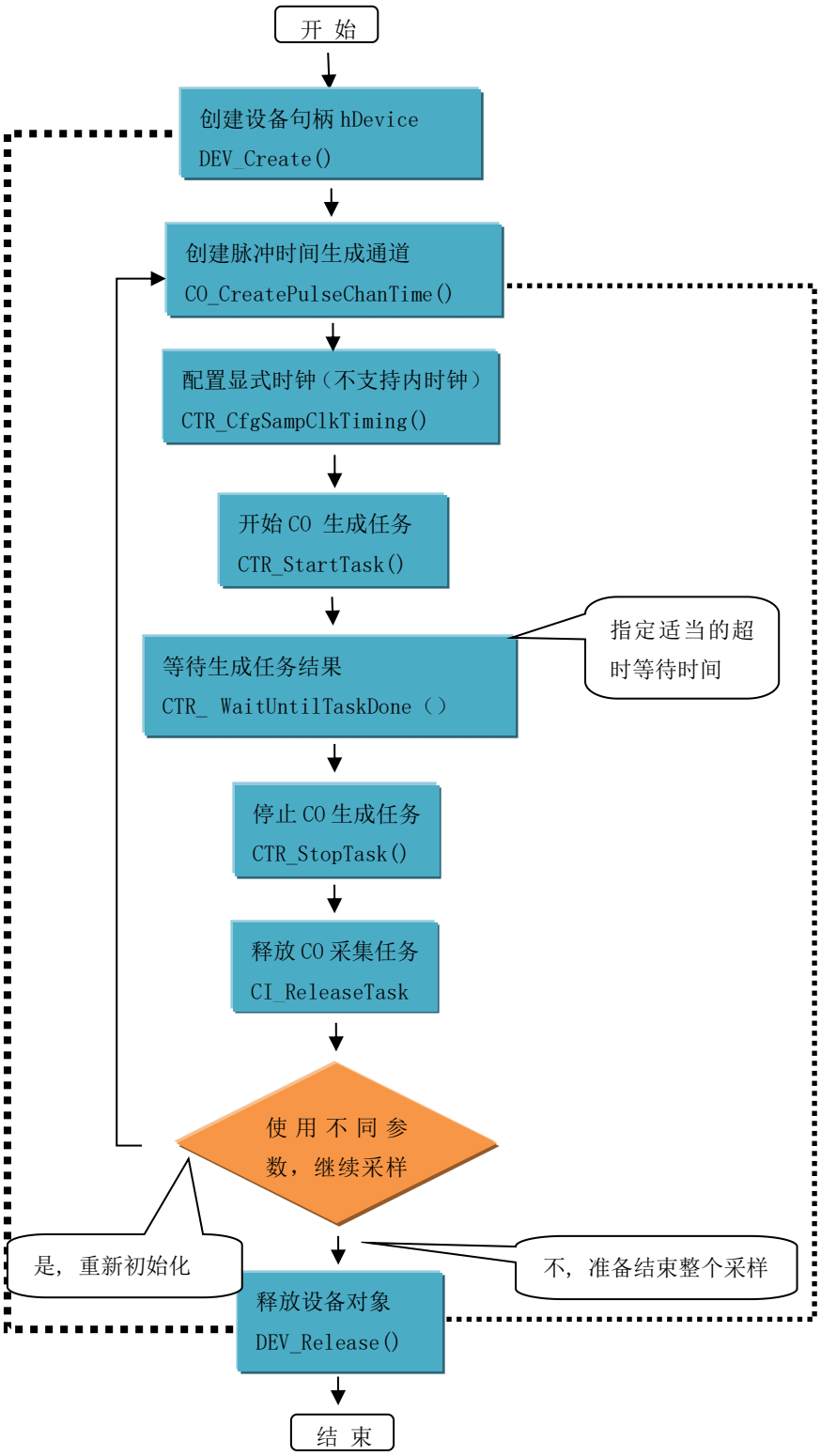


图 2.21.1　　CO 连续采样、固定脉冲流程图

2.21.2 可变脉冲(缓冲)重生成输出

可变脉冲也称为缓冲输出，就是在启动任务前即创建脉冲输出通道时就不指整个脉冲序列的频率、占空比等，即每个脉冲宽度和占空比均由后面写入缓冲中的数据序列决定，在输出过程中每个脉冲的频率、宽度、占空比是可以不一样的。在可变脉冲输出方式下既支持重生成又支持非重生成模式。

现以低电平和高电平的时间长度指定脉冲为例，讲述连续、可变脉冲、重生成时的控制流程：

(1) DEV_Create()　创建设备句柄；

(2) CO_CreatePulseChanTime()　创建脉冲时间生成通道；

(3) CTR_CfgSampClkTiming()　配置显式时钟(不支持内时钟)，nSamleMode=3

(4) CO_WriteTime()　写入脉冲序列数据(低电平和高电平时间长度值)

(4) CTR_StartTask()　开始 CO 生成任务；

(5) CTR_WaitUntilTaskDone()　等待 CO 生成任务结束；

(6) CTR_StopTask()　　停止 CO 生成任务；

(7) CTR_ReleaseTask()　释放 CO 生成任务；

(8) DEV_Release()　释放设备句柄。

如果每次采集参数相同的情况下，可以在 4、5、6 步间循环进行，即可实现高速连续不间断采样；

如果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。如下图：

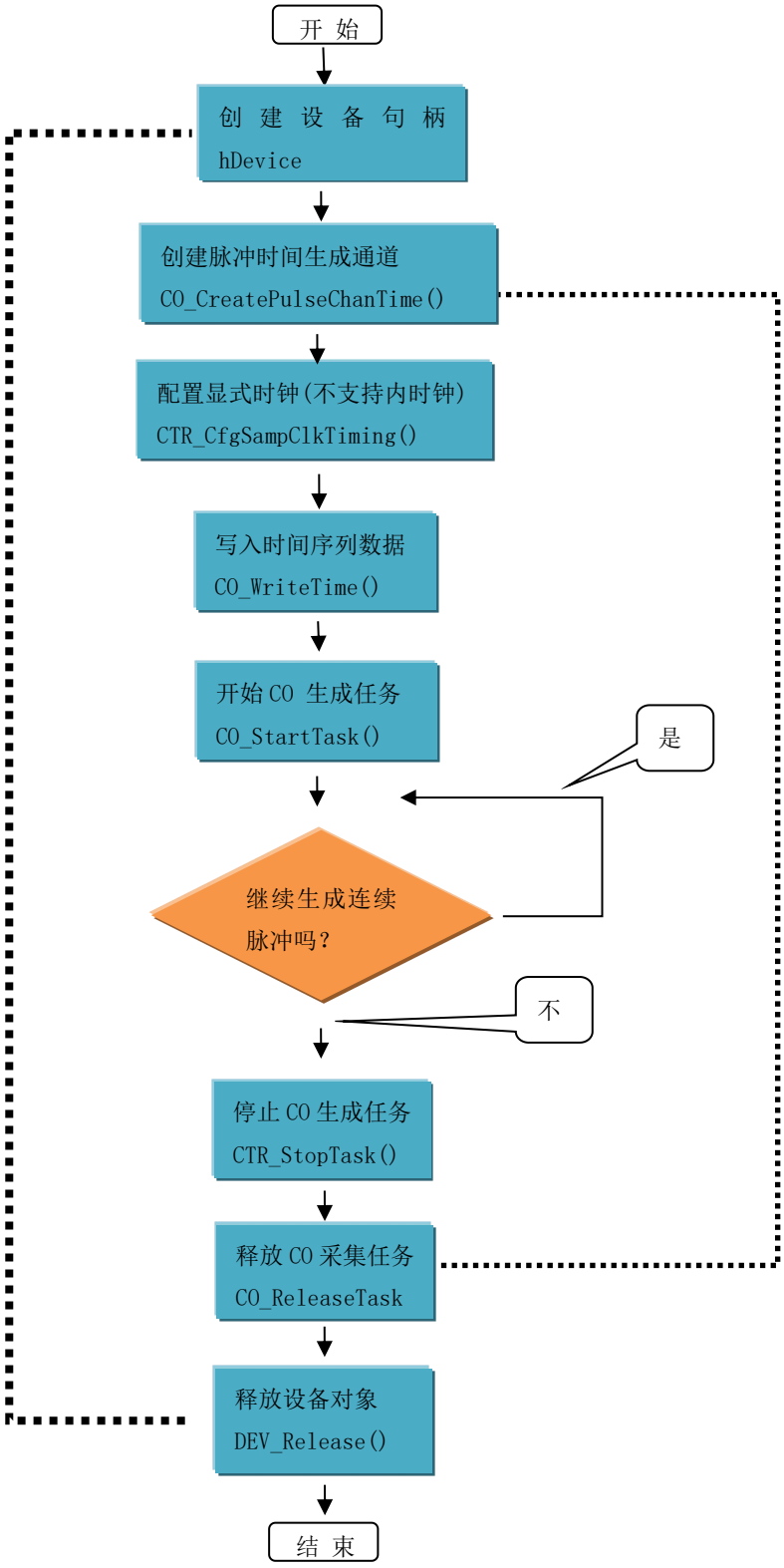


图 2.21.2　　CO 连续采样、可变脉冲流程图例

2.21.3　可变脉冲(缓冲)非重生成输出

可变脉冲也称为缓冲输出，就是在启动任务前即创建脉冲输出通道时就不指整个脉冲序列的频率、占空比等，即每个脉冲宽度和占空比均由后面写入缓冲中的数据序列决定，在输出过程中每个

脉冲的频率、宽度、占空比是可以不一样的。在可变脉冲输出方式下既支持重生成又支持非重生成模式。

现以低电平和高电平的时间长度指定脉冲为例，讲述连续、可变脉冲、非重生成时的控制流程：

(1) DEV_Create()　创建设备句柄；

(2) CO_CreatePulseChanTime()　创建脉冲时间生成通道；

(3) CTR_CfgSampClkTiming() 配置显式时钟(不支持内时钟)，nSamleMode=3

(4) CO_WriteTime() 写入脉冲序列数据(低电平和高电平时间长度值)

(4) CTR_StartTask()　开始 CO 生成任务；

(5) CO_WriteTime() 写入后续脉冲序列数据(低电平和高电平时间长度值)

(6) CTR_WaitUntilTaskDone() 等待 CO 生成任务结束；

(7) CTR_StopTask()　　停止 CO 生成任务；

(8) CTR_ReleaseTask() 释放 CO 生成任务；

(9) DEV_Release()　释放设备句柄。

如果每次采集参数相同的情况下，可以在 4、5、6 步间循环进行，即可实现高速连续不间断采样；

如果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。如下图：

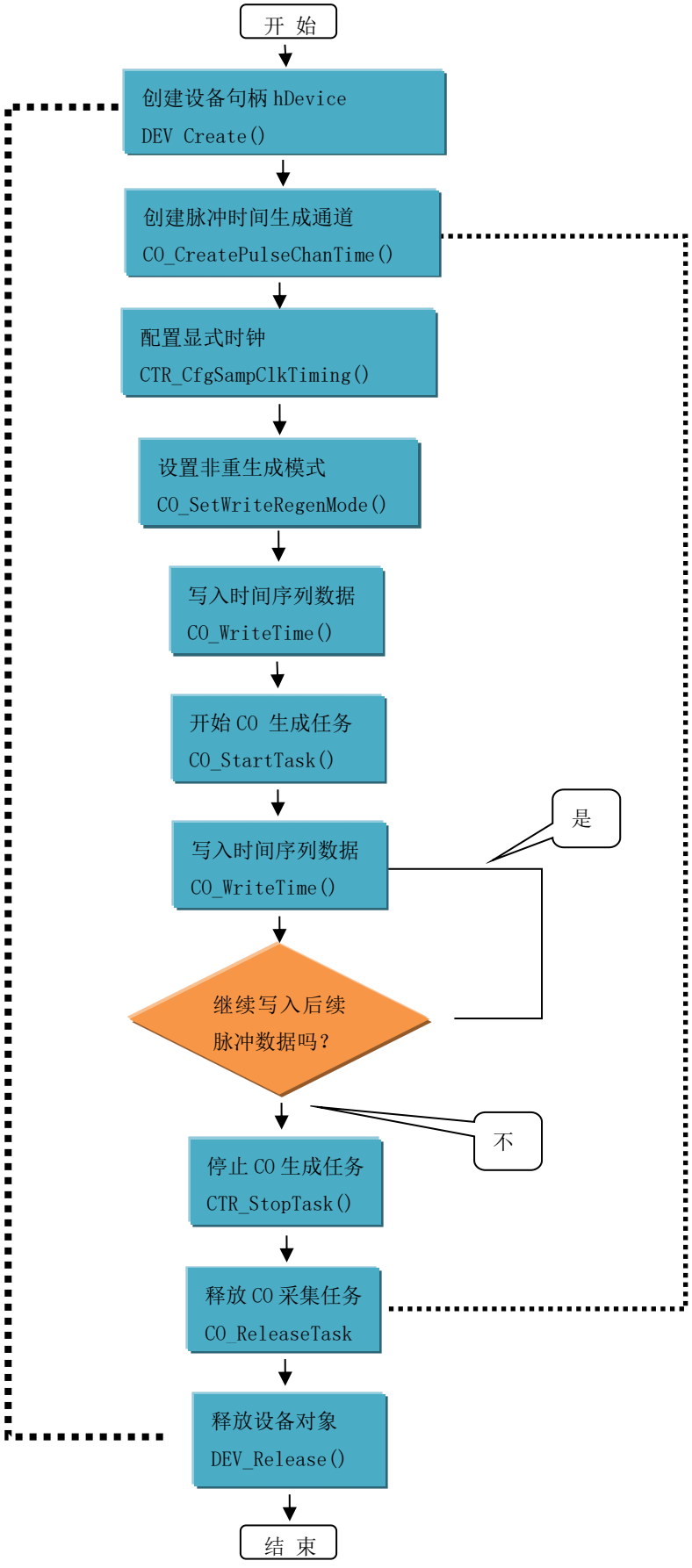


图 2.21.3　　CO 连续采样、可变脉冲流程图例

上图虚线表示对称关系。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。CO_CreatePulseChanTime()等 CO 脉冲生成通道创建函数和 CTR_ReleaseTask()两个函数的对称关系是只有在 CTR_ReleaseTask()之后才能再次 CO_CreatePulseChanTime()等 CO 脉冲生成通道创建函数。

2.22 用户开发所必须的函数

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()两个函数则是必不可少的。因为对于所有的设备访问都要有这两个函数的帮助。

3 主要功能组函数介绍

3.1 DEV 设备对象管理函数原型说明

DEV_Create()

函数原型：

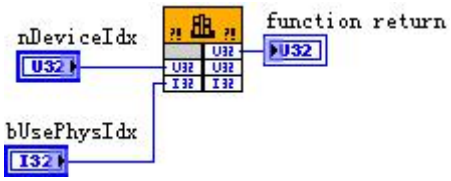
Visual C++ / C++Builder / LabWindows/CVI:

HANDLE DEV_Create(ULONG nDeviceIdx, BOOL bUsePhysIdx)

Visual Basic：

Declare Function DEV_Create Lib "USB2861" (ByVal nDeviceIdx As Long, _
ByVal bUsePhysIdx As Long) As long

LabVIEW：



请参考 USB2861.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，驱动程序自动以逻辑号来管理每张卡。比如某台计算机系统中插入该卡共四张，则根据操作系统的加载顺序依次分配的逻辑号为 0、1、2、3。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)，可立即调用 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 "USB2861" () As Long

LabVIEW：

请参考 USB2861.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，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，可立即调用 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：

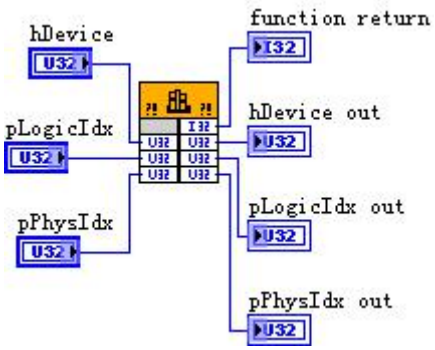
Visual C++ / C++Builder / LabWindows/CVI:

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

Visual Basic:

```
Declare Function DEV_GetCurrentIdx Lib "USB2861" (ByVal hDevice As Long, _
                                                  ByRef pLgcIdx As Long, _
                                                  ByRef pPhysIdx As Long) As Boolean
```

LabVIEW：



请参考 USB2861.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，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pLgcIdx 出口参数，取得设备的逻辑索引号(Logical Index)，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)，取值范围为[0, 255]，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，可立即调用 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_GetSpeed()

函数原型：

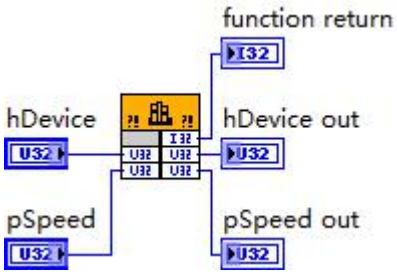
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DEV_GetSpeed(HANDLE hDevice,
                  U32* pSpeed)
```

Visual Basic:

```
Declare Function DEV_GetSpeed Lib "USB2861" (ByVal hDevice As Long, _
                                             ByRef pSeed As Long ) As Boolean
```

LabVIEW：



请参考 USB2861.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，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pSpeed 出口参数，返回 USB 总线速度版本号，若为 USB1.0 则返回 1，若为 USB2.0 则返回 2，若为 USB3.0 则返回 3。

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

相关函数：　DEV_Release()

DEV_Release()

函数原型：

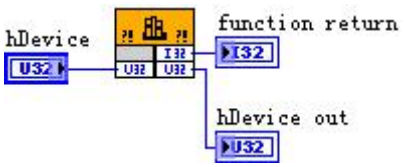
Visual C++ / C++Builder / LabWindows/CVI:

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

Visual Basic:

```
Declare Function DEV_Release Lib "USB2861" (ByVal hDevice As Long ) As Boolean
```

LabVIEW：



请参考 USB2861.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，由 DEV_Create()函数创建，该句柄指向要访问的设备。

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

相关函数： DEV_Release()

3.2 AI 模拟量输入函数原型说明

AI_InitTask()

函数原型：

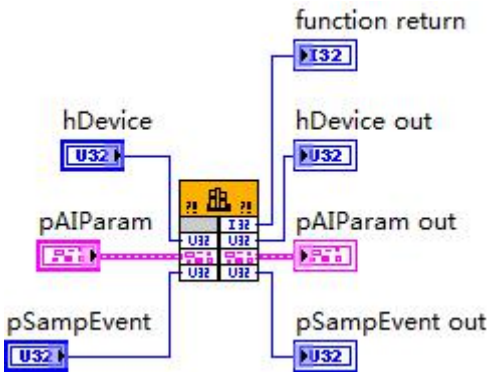
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_InitTask (HANDLE hDevice, PAI_PARAM pAIParam, HANDLE* pSampEvent)
```

Visual Basic:

```
Declare Function AI_InitTask Lib "USB2861" (ByVal hDevice As Long, _
                                            ByRef pAIParam As PAI_PARAM;
                                            ByRef pSampEvent As Long) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：初始化 AI 任务参数(Initialize task 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。但它并不开始 AI 设备，若要开始 AI 设备，须在成功调用此函数之后再调用 AI_StartTask()函数。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pAIParam 入口参数，AI 工作参数结构体指针，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《4 各种结构体描述》\《4.1 AI_PARAM（AI 工作参数结构）》。

pSampEvent 出口参数，事件句柄，该事件属性为自动发信号，初始状态为不发信号，它由任务自动创建。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：当任务中每通道有 AIParam.nSampsPerChan 个采样点的数据时则会触发此事件。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，则意味着任务中至少有 nSampsPerChan 个数据可读，则采集线程立即进入运行状态，并迅速调用 AI_ReadAnalog()或 AI_ReadBinary()循环读取任务中的数据，直至 nAvailSampsPerChan 小于 nReadSampsPerChan。

如果简单循环调用 AI_GetStatus()查询 AI 的各种状态以同步数据读操作，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。当然最简单的办法则是使用 AI_ReadAnalog()或 AI_ReadBinary()函数的超时机制，即利用 fTimeout 参数的合理设置来同步读入操作，而无须关注和跟踪 pSampEvent 事件。

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

相关函数： DEV_Create() AI_InitTask() AI_StartTask()
AI_SendSoftTrig() AI_GetStatus() AI_WaitUntilTaskDone()
AI_ReadAnalog() AI_ReadBinary() AI_StopTask()
AI_ReleaseTask() DEV_Release()

AI_StartTask()

函数原型：

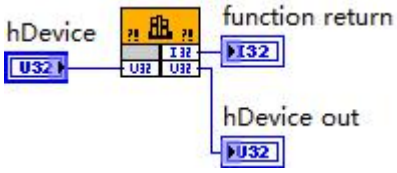
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AI_StartTask (HANDLE hDevice)

Visual Basic:

Declare Function AI_StartTask Lib "USB2861" (ByVal hDevice As Long) As Boolean

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：开始 AI 采集(Start task for analog input)。必须在成功调用 AI_InitTask()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 实际是否进入采集记录过程，须依赖于触发事件的产生。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

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

相关函数： DEV_Create() AI_InitTask() AI_StartTask()
AI_SendSoftTrig() AI_GetStatus() AI_WaitUntilTaskDone()
AI_ReadAnalog() AI_ReadBinary() AI_StopTask()
AI_ReleaseTask() DEV_Release()

AI_SendSoftTrig()

函数原型：

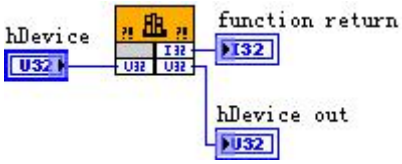
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AI_SendSoftTrig (HANDLE hDevice)

Visual Basic:

Declare Function AI_SendSoftTrig Lib "USB2861" (ByVal hDevice As Long) As Boolean

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：发送软件触发事件(Send software trigger event for analog input)。设备进入等待触发状态，若用户需软件触发或需要随时手动给触发事件时，可调用此函数。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，由 DEV_Create()函数创建，该句柄指向要访问的设备。

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，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AI_InitTask() AI_StartTask()
AI_SendSoftTrig() AI_GetStatus() AI_WaitUntilTaskDone()
AI_ReadAnalog() AI_ReadBinary() AI_StopTask()
AI_ReleaseTask() DEV_Release()

AI_GetStatus()

函数原型：

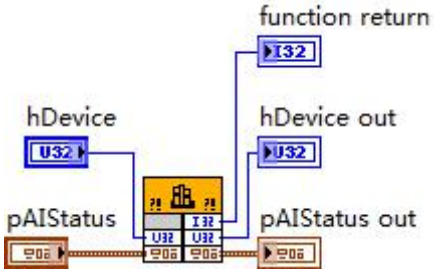
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_GetStatus (HANDLE hDevice, PAI_STATUS pStatus )
```

Visual Basic:

```
Declare Function AI_GetStatus Lib "USB2861" (ByVal hDevice As Long, _
                                              ByRef pAIStatus As AI_STATUS) As Boolean
```

LabVIEW:



请参考 USB2861.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)。一旦调用 AI_StartTask()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。nAvailSampsPerChan>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_ReadBinary()或AI_ReadAnalog()函数循环读取若干可读段数据，直到 nAvailSampsPerChan>nReadSampsPerChan 时可调用读数函数。如果在开始采集任务后，设备迟迟不能被触发采样，可以根据需要调用 AI_SendSoftTrig()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。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pAIStatus 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《4　各种结构体描述》\《4.2　AI_STATUS（AI 状态信息结构）》。

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

相关函数： DEV_Create()　AI_InitTask()　AI_StartTask()　AI_SendSoftTrig()　AI_GetStatus()　AI_WaitUntilTaskDone()　AI_ReadAnalog()　AI_ReadBinary()　AI_StopTask()　AI_ReleaseTask()　DEV_Release()

AI_WaitUntilTaskDone()

函数原型：

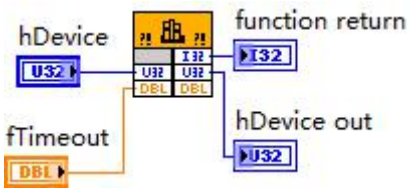
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_WaitUntilTaskDone (HANDLE hDevice, F64 fTimeout)
```

Visual Basic:

```
Declare Function AI_WaitUntilTaskDone Lib "USB2861" (ByVal hDevice As Long, _
                                                     ByVal fTimeout As Double) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：在 AI 的采集任务结束前等待(Wait until task done for analog input)。一旦调用 AI_StartTask()函数后，可以调用此函数等待采集任务结束。它通常用在有限点采样模式中。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

fTimeout 入口参数，超时时间，单位：秒（S）。指定该次等待所用时间，比如设定为 10.0，即 10 秒钟的时间，如果在 10 秒内采集任务结束，则函数立即返回 TRUE，否则 10 秒钟后函数返回值 FALSE，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即总是等到采集任务结束才返回）则赋值小于 0 即可，如-1.0；如果 fTimeout=0.0，则意味着该函数只是简单查询采集任务是否结束，如果采集任务结束了，则返回 TRUE，否则返回 FALSE。

返回值：如果采集任务结束，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AI_InitTask() AI_StartTask()
AI_SendSoftTrig() AI_GetStatus() AI_WaitUntilTaskDone()
AI_ReadAnalog() AI_ReadBinary() AI_StopTask()
AI_ReleaseTask() DEV_Release()

AI_ReadAnalog()

函数原型:

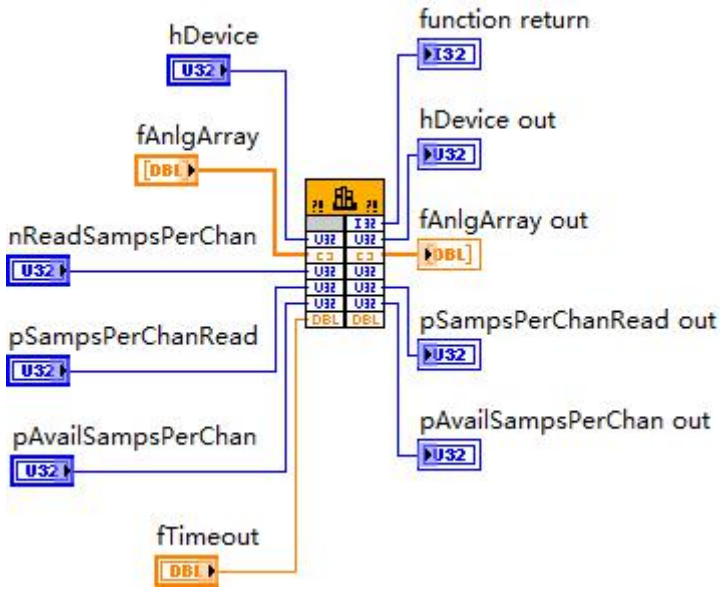
Visual C++ / C++Builder / LabWindows/CVI:

```
LONG AI_ReadAnalog(HANDLE hDevice,
                   F64 fAnlgArray[],
                   U32 nReadSampsPerChan,
                   U32* pSampsPerChanRead,
                   U32* pAvailSampsPerChan,
                   F64 fTimeout);
```

Visual Basic:

```
Declare Function AI_ReadAnalog Lib "USB2861" ( ByVal hDevice As Long, _
                                               ByRef fAnlgArray As Double, _
                                               ByVal nReadSampsPerChan As Long, _
                                               ByRef pSampsPerChanRead As Long,_
                                               ByRef pAvailSampsPerChan As Long,_
                                               ByVal fTimeout As Long) As Long
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取模拟量采样数据(主要是电压数据)（Read analog data from the task）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

fAnlgArray 出口参数，用户缓冲区，用于接收所有采样通道模拟量数据，值区间由相应采样通道的选用采样范围决定，单位：伏(V)，数据类型为双精度浮点。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 nReadSampsPerChan*AIParam.nSampChanCount。如果选择的不是 1 倍增益挡位，调用 AI_GetGainInfo() 获得相应增益挡位下的放大倍数 fAmpFactor，然后将该函数读取到的电压值再除以 fAmpFactor 后才是实际信号的测量结果。

nReadSampsPerChan 入口参数，每通道请求读入的数据点数。

在有限点和连续采样模式中，它指定该次从设备的当前可读数据位置读取的数据点数（单位：点）。注意此参数的值如大于当前的可读数点 nAvailSampsPerChan 则会继续等待直到至少有 nReadSampsPerChan 个点可读后读函数才会返回。等待期间，如果所等时间超过 fTimeout 指定时间也会返回，并置超时错误码。

在连续采样过程中，如果要保持连续不丢点，此参数应尽可能接近于甚至等于当前的可读点数(nAvailSampsPerChan),但不能大于 AIParam.nSampsPerChan。当然此参数值也不能大于 fAnlgArray 的缓冲区长度，所以为避免出错，所开辟的缓冲区不能小于 nReadSampsPerChan*AIParam.nSampChanCount。

pSampsPerChanRead 出口参数，返回每通道实际读取的点数。在单点采样模式中，如果读取成功，返回的每通道已读取点数总是为 1。注意每次都要检查 pSampsPerChanRead 的返回值，如果返回值等于 0，则要慎重处理。

pAvailSampsPerChan 出口参数，返回该次读操作完成时的每通道还可读而未读的数据点数。它跟 AI_GetStatus()函数取得的状态信息 AIStatus.nAvailSampsPerChan 是同一个状态信息。返回可读点数的意义在于通过数据读操作直接提供给用户，避免再次调用 AI_GetStatus()函数，即在读数据函数返回时判断可读点数，若大于 nReadSampsPerChan，则可紧接着再次调用读数据函数，直到 pAvailSampsPerChan 返回值小于 nReadSampsPerChan。在单点采样模式中,它的返回值总是为 0。

fTimeout 入口参数，超时时间，单位：秒（S）。指定等待写入额定点数的时间。比如设定为 10.0，如果在 10 秒内读取的点数达到 nReadSampsPerChan 时立即返回 TRUE，否则 10 秒钟后函数返回 FALSE，并通过 pAvailSampsPerChan 告之实际读入的点数，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即等待请求数据点数完

全从任务中读到才返回）则置为负数，如-1.0 即可。如果 fTimeout=0.0，则意味着该函数仅简单判断能否立即读取到请求的点数，如果不能，则不等待，立即返回 FALSE，否则返回 TRUE。

返回值：如果函数调用成功则返回 TRUE，否则返回 FALSE，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_AVAILABLE_SAMPS	0xE0000000+1	无有效点数据
ERROR_SAMPLE_TASK_FAIL	0xE0000000+2	采集任务失败
ERROR_TIMEOUT	1460L	超时错误(见微软定义 WinError.h)

相关函数： DEV_Create() AI_InitTask() AI_StartTask() AI_SendSoftTrig() AI_GetStatus() AI_WaitUntilTaskDone() AI_ReadAnalog() AI_ReadBinary() AI_StopTask() AI_ReleaseTask() DEV_Release()

AI_ReadBinary()

函数原型:

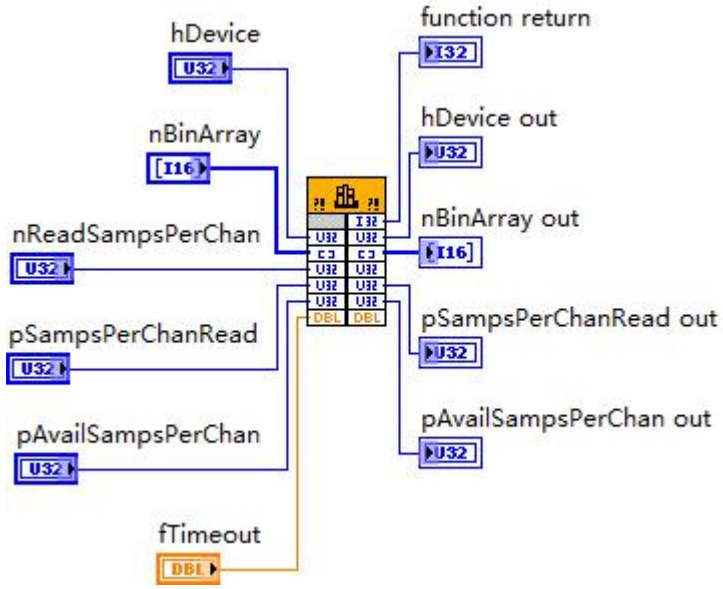
Visual C++ / C++Builder / LabWindows/CVI:

```
LONG AI_ReadBinary(HANDLE hDevice,
                   U16 nBinArray[],
                   U32 nReadSampsPerChan,
                   U32* pSampsPerChanRead,
                   U32* pAvailSampsPerChan,
                   F64 fTimeout);
```

Visual Basic:

```
Declare Function AI_ReadAnalog Lib "USB2861" ( ByVal hDevice As Long, _
                                               ByRef nBinArray As Integer, _
                                               ByVal nReadSampsPerChan As Long, _
                                               ByRef pSampsPerChanRead As Long,_
                                               ByRef pAvailSampsPerChan As Long,_
                                               ByVal fTimeout As Long) As Long
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取二进制原码采样数据（Read binary data from the task）。该函数不对采样结果进行物理量的换算，它是设备采样后的直接结果。二进制原码具有数据紧凑、数据量小、传输快、处理快、存盘也快的特点。

参数：

hDevice 同 AI_ReadAnalog()。

nBinArray 出口参数，用户缓冲区，用于接收所有采样通道二进制原码数据，值区间[-32768，32767]。各个采样通道的数据点是依次交替排列的。它点空间不能小于 nReadSampsPerChan*AIParam.nSampChanCount。如果选择的不是 1 倍增益挡位，那么还得调用 AI_GetGainInfo() 获得相应增益挡位下的放大倍数 fAmpFactor，然后将该函数读取到的原码数据再除以 fAmpFactor 后才是实际信号的测量结果。

nReadSampsPerChan 同 AI_ReadAnalog()。

pSampsPerChanRead 同 AI_ReadAnalog()。

pAvailSampsPerChan 同 AI_ReadAnalog()。

fTimeout 同 AI_ReadAnalog()

返回值：同 AI_ReadAnalog()

相关函数： DEV_Create() AI_InitTask() AI_StartTask() AI_SendSoftTrig() AI_GetStatus() AI_WaitUntilTaskDone() AI_ReadAnalog() AI_ReadBinary() AI_StopTask() AI_ReleaseTask() DEV_Release()

关于实时连续采样调用流程的图例请参考《2 使用提要》中《2.6 AI 连续采样模式 》相关部分。

该函数读取的是原码数据，如果换算成电压值后，还得考虑所选择的增益挡位是多少？如果选择的不是 1 倍增益挡位，那么还得调用 AI_GetGainInfo() 获得相应增益挡位下的放大倍数 fAmpFactor，然后将之前换算后的电压值再除以 fAmpFactor 后才是实际信号的测量结果。

AI_StopTask()

函数原型：

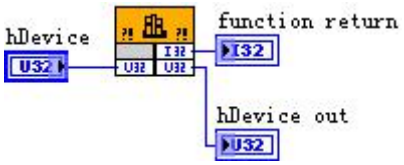
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AI_StopTask (HANDLE hDevice)

Visual Basic:

Declare Function AI_StopTask Lib "USB2861" (ByVal hDevice As Long)

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

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

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

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，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AI_InitTask() AI_StartTask()
AI_SendSoftTrig() AI_GetStatus() AI_WaitUntilTaskDone()
AI_ReadAnalog() AI_ReadBinary() AI_StopTask()
AI_ReleaseTask() DEV_Release()

AI_ReleaseTask()

函数原型：

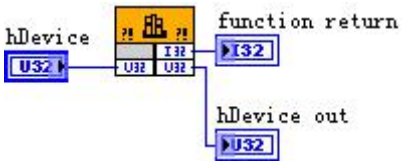
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AI_ReleaseTask (HANDLE hDevice)

Visual Basic:

Declare Function AI_ReleaseTask Lib "USB2861" (ByVal hDevice As Long)

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：释放 AI(Release task for analog input)。必须在重新调用 AI_InitTask()函数之前被调用一次，即该函数必须和 AI_InitTask()成对出现。注意此函数在内部首先执行 AI_StopTask()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，由 DEV_Create()函数创建，该句柄指向要访问的设备。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AI_InitTask() AI_StartTask()

AI_SendSoftTrig()　　AI_GetStatus()　　AI_WaitUntilTaskDone()
AI_ReadAnalog()　　AI_ReadBinary()　　AI_StopTask()
AI_ReleaseTask()　　DEV_Release()

AI_ScaleBinToVolt()

函数原型：

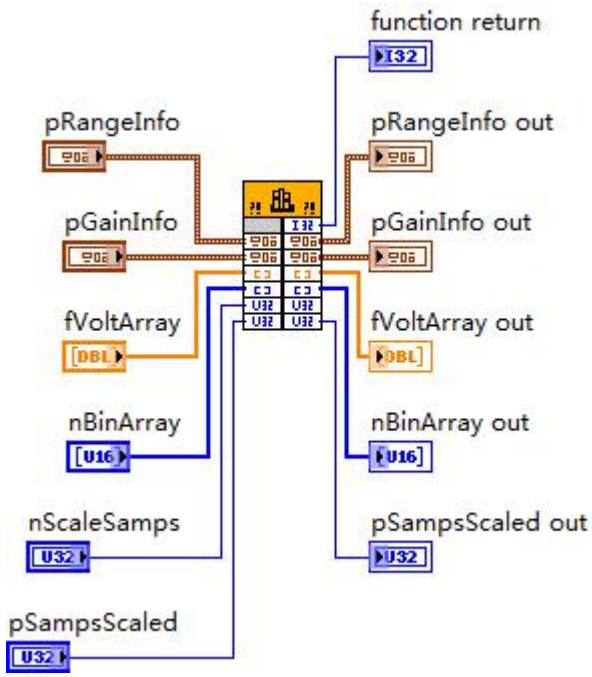
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_ScaleBinToVolt(HANDLE hDevice,
                       PAI_VOLT_RANGE_INFO pRangeInfo,
                       PVOID pGainInfo,
                       F64 fVoltArray[],
                       I16 nBinArray[],
                       U32 nScaleSamps,
                       U32* pSampsScaled);
```

Visual Basic:

```
Declare Function AI_ScaleBinToVolt Lib "USB2861" (
                                   ByRef pRangeInfo As AI_VOLT_RANGE_INFO, _
                                         ByRef pGainInfo As Long, _
                                         ByRef fVoltArray As Double, _
                                         ByRef nBinArray As Integer, _
                                         ByVal nScaleSamps As Long, _
                                         ByRef pSampsScaled As Long) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：将二进制原码数据转换为电压数据(Scale binary data to voltage data)。与 AI_ScaleVoltToBin()函数的功能相反。

参数：

pRangeInfo 入口参数，范围信息。它由 AI_GetVoltRangeInfo()函数获取。

pGainInfo 入口参数，增益信息。因本设备 AI 无增益功能，该参数无效，恒等于 NULL。

fVoltArray 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 pRangeInfo 参数指定。

nBinArray 入口参数，用于传入待量化的原码数据，取值范围[0, 65535]。

nScaleSamps 入口参数，请求量化的数据点数。

pSampsScaled 出口参数，返回实际量化后数据点数。

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

相关函数： DEV_Create() AI_ScaleBinToVolt() AI_ScaleVoltToBin() AI_GetMainInfo() AI_GetVoltRangeInfo() AI_GetGainInfo() AI_GetRateInfo() DEV_Release()

AI_ScaleVoltToBin()

函数原型：

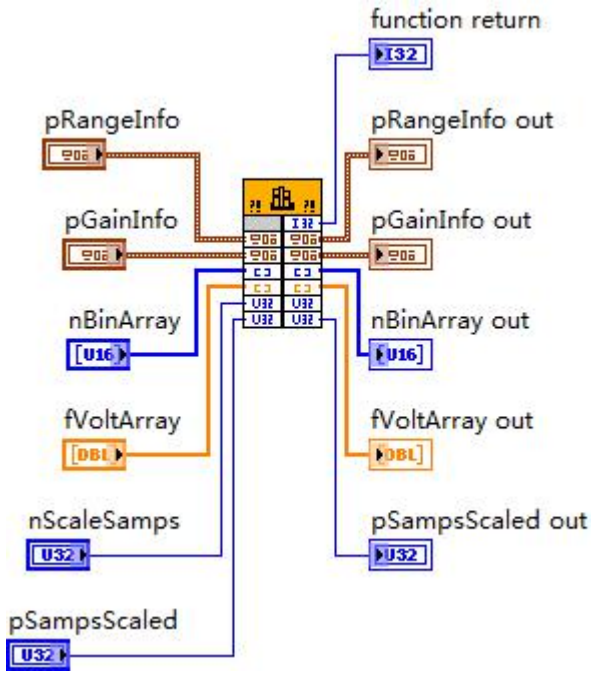
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_ScaleVoltToBin(
                    PAI_VOLT_RANGE_INFO pRangeInfo,
                    PVOID pGainInfo,
                    U16 nBinArray[],
                    F64 fVoltArray[],
                    U32 nScaleSamps,
                    U32* pSampsScaled);
```

Visual Basic:

```
Declare Function AI_ScaleVoltToBin Lib "USB2861" (
                                        ByRef pRangeInfo As AI_VOLT_RANGE_INFO,
_
                                        ByRef pGainInfo As Long, _
                                        ByRef nBinArray As Integer, _
                                        ByRef fVoltArray As Double, _
                                        ByVal nScaleSamps As Long, _
                                        ByRef pSampsScaled As Long) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：将电压数据转换为原码数据(Scale voltage data to binary data)。与 AI_ScaleBinToVolt()函数的功能相反。

参数：

pRangeInfo 入口参数，范围信息。它由 AI_GetVoltRangeInfo()函数获取。

pGainInfo 入口参数，增益信息。因本设备 AI 无增益功能，该参数无效，恒等于 NULL。

nBinArray 出口参数，用于传入待量化的原码数据，取值范围[0, 65535]。

fVoltArray 入口参数，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。

nScaleSamps 入口参数，请求量化的数据点数。

pSampsScaled 出口参数，返回实际量化后数据点数。

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

相关函数： DEV_Create() AI_ScaleBinToVolt() AI_ScaleVoltToBin() AI_GetMainInfo() AI_GetVoltRangeInfo() AI_GetGainInfo() AI_GetRateInfo() DEV_Release()

AI_GetMainInfo()

函数原型：

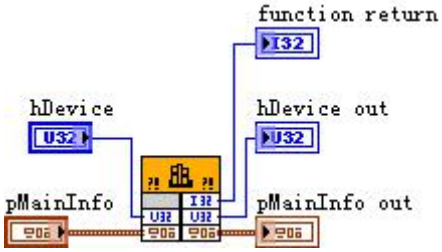
Visual C++ / C++Builder / LabWindows/CVI:

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

Visual Basic:

```
Declare Function AI_GetMainInfo Lib "USB2861" ( ByVal hDevice As Long, _
                                                ByRef pMainInfo As AI_MAIN_INFO) As Boolean
```

LabVIEW:



请参考 USB2861.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（Get main information for analog input）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pMainInfo 出口参数，AI 主要信息结构指针，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 USB2861.h 或 USB2861.bas 或 USB2861.pas 函数原型定义的头文件，也可参考本文《4　各种结构体描述》关于该结构的有关说明。

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

相关函数： DEV_Create()　AI_ScaleBinToVolt()　AI_ScaleVoltToBin()
AI_GetMainInfo()　AI_GetVoltRangeInfo()　AI_GetGainInfo()
AI_GetRateInfo()　DEV_Release()

AI_GetVoltRangeInfo()

函数原型：

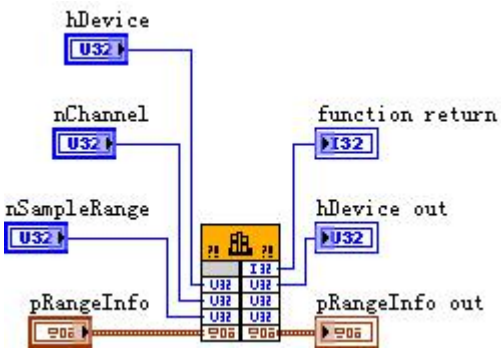
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AI_GetVoltRangeInfo(HANDLE hDevice,
                         int nChannel,
                         int nSampleRange,
                         PAI_VOLT_RANGE_INFO pRangeInfo)
```

Visual Basic:

```
Declare Function AI_GetVoltRangeInfo Lib "USB2861" (ByVal hDevice As Long, _
                                                    ByVal nChannel As Long, _
                                                    nSampleRange As Long, _
                                                   ByRef pRangeInfo As AI_VOLT_RANGE_INFO)
As Boolean
```

LabVIEW:



请参考 USB2861.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（Get range information for analog input）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

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

nSampleRange 入口参数，AI 采样范围，同 AIParam.nSampleRange。

pRangeInfo 出口参数，AI 采样范围信息，负责返回 AI 的采样范围大值、最小值、幅度、编码宽度等。详情请参考《 4.4 AI_VOLT_RANGE_INFO (AI 采样范围信息结构体) 》

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

相关函数： DEV_Create() AI_ScaleBinToVolt() AI_ScaleVoltToBin() AI_GetMainInfo() AI_GetVoltRangeInfo() AI_GetGainInfo() AI_GetRateInfo() DEV_Release()

AI_GetRateInfo()

函数原型：

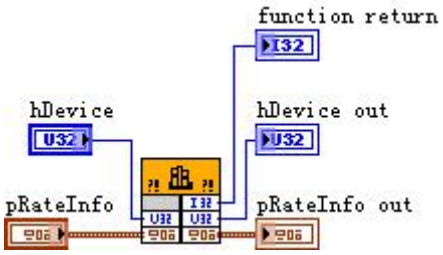
Visual C++ / C++Builder / LabWindows/CVI:

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

Visual Basic:

```
Declare Function AI_GetRateInfo Lib "USB2861" (ByVal hDevice As Long, _
                                               ByRef pRateInfo As AI_RATE_INFO) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：获得采样速率信息（Get rate information for analog input）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pRateInfo 出口参数，AI 采样速率信息，负责返回 AI 的最大采样率，最小采样率等。详情请参考《 4.5 AI_SAMP_RATE_INFO (AI 采样率信息结构体) 》。

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

相关函数： DEV_Create() AI_ScaleBinToVolt() AI_ScaleVoltToBin() AI_GetMainInfo() AI_GetVoltRangeInfo() AI_GetGainInfo() AI_GetRateInfo() DEV_Release()

AI_VerifyParam()

函数原型：

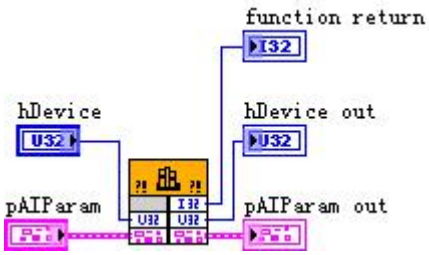
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AI_VerifyParam (HANDLE hDevice, PAI_PARAM pAIParam)

Visual Basic:

Declare Function AI_VerifyParam Lib "USB2861" (ByVal hDevice As Long, _
ByRef pAIParam As AI_PARAM) As Boolean

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：校验 AI 工作参数(Verify task parameter for analog input)，这是一个辅助功能的函数，在初始化 AI 前，先校验 AI 参数的合法性。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

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《4　各种结构体描述》\《4.1　AI_PARAM（AI 工作参数结构体）》。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 USB2861.log 记录不合法的参数名和原因，然后返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： AI_InitTask()　AI_VerifyParam()　AI_LoadParam()
AI_SaveParam()　AI_ResetParam()

AI_LoadParam()

函数原型：

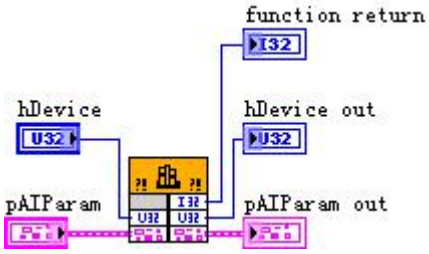
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AI_LoadParam (HANDLE hDevice,
PAI_PARAM pAIParam)

Visual Basic:

Declare Function AI_LoadParam Lib "USB2861" (ByVal hDevice As Long, _
ByRef pAIParam As AI_PARAM) As Boolean

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：从 USB2861.ini 参数配置文件中读取设备的硬件参数(Load parameter from USB2861.ini for analog input)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pAIParam 出口参数，属于 PAI_PARAM 的结构指针类型，负责返回 AI 工作参数值，关于结构指针类型 PAI_PARAM 请参考 USB2861.h 或 USB2861.bas 或 USB2861.pas 函数原型定义的头文件，也可参考本文《4.1 AI_PARAM（AI 工作参数结构体）》。

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

相关函数： AI_InitTask() AI_VerifyParam() AI_LoadParam()
AI_SaveParam() AI_ResetParam()

AI_SaveParam()

函数原型：

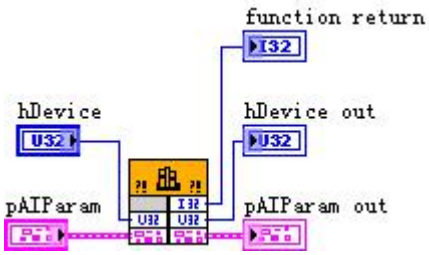
Visual C++ / C++Builder / LabWindows/CVI:

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

Visual Basic:

```
Declare Function AI_SaveParam Lib "USB2861" (ByVal hDevice As Long, _
                                             ByRef pAIParam As AI_PARAM) As Boolean
```

LabVIEW:



请参考 USB2861.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 USB2861.ini 参数配置文件中(Save parameter to USB2861.ini for analog input)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

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 USB2861.h 或 USB2861.bas 或 USB2861.pas 等函数原型定义的头文件，也可参考本文《4.1 AI_PARAM（AI 工作参数结构体）》。

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

相关函数： AI_InitTask() AI_VerifyParam() AI_LoadParam()
AI_SaveParam() AI_ResetParam()

AI_ResetParam()

函数原型：

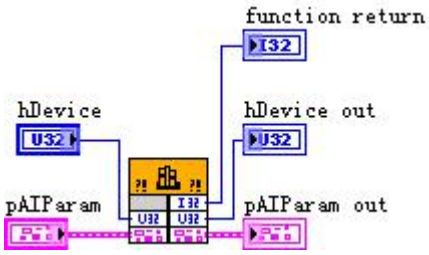
Visual C++ / C++Builder / LabWindows/CVI:

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

Visual Basic:

Declare Function AI_ResetParam Lib "USB2861" (ByVal hDevice As Long, _
ByRef pAIParam As AI_PARAM) As Boolean

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：将 USB2861.ini 参数配置文件中原来的 AI 参数值复位至出厂时的默认值(Reset parameter to default value for analog input)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pAIParam 出口参数，AI 工作参数结构指针，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 USB2861.h 或 USB2861.bas 或 USB2861.pas 函数原型定义的头文件，也可参考本文《4.1 AI_PARAM（AI 工作参数结构体）》。

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

相关函数： AI_InitTask() AI_VerifyParam() AI_LoadParam()
AI_SaveParam() AI_ResetParam()

3.3 AO 模拟量输出函数原型说明

AO_InitTask()

函数原型：

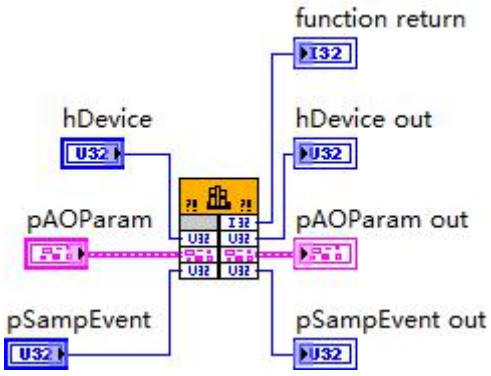
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AO_InitTask (HANDLE hDevice, PAO_PARAM pAOParam, HANDLE* pSampEvent)

Visual Basic:

Declare Function AO_InitTask Lib "USB2861" (ByVal hDevice As Long, _
ByRef pAOParam As PAO_PARAM;
ByRef pSampEvent As Long) As Boolean

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：初始化 AO 任务参数(Initialize task parameter for analog output)，为设备操作就绪有关状态，如预置 AO 采样率、各通道模拟量采样范围等。但它并不开始 AO 设备，如果要开始 AO 设备，须在成功调用此函数之后再调用 AO_StartTask()函数。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pAOParam 入口参数，AO 工作参数结构体指针，决定了 AO 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《4　各种结构体描述》\《4.6　AO_PARAM（AO 工作参数结构体）》。

pSampEvent 出口参数，事件句柄，该事件属性为自动发信号，初始状态为不发信号，它由任务自动创建。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：当任务中每完成 AOParam.nSampsPerChan 个点数的数据输出时则会触发此事件。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 nSampsPerChan 个点的数据可写，则采集线程立即进入运行状态，并迅速调用 AO_WriteAnalog()或 AO_WriteBinary()循环往任务中写入数据，直至 nAvailSampsPerChan 小于 nWriteSampsPerChan,。如果简单循环调用 AO_GetStatus()查询 AO 的各种状态以同步数据写操作，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，如果事件未被触发，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。当然最简单的办法则是使用 AO_WriteAnalog()或 AO_WriteBinary()函数的超时机制，即利用 fTimeout 参数的合理设置来同步写入操作，而无须关注和跟踪 pSampEvent 事件。

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

相关函数： DEV_Create()　AO_InitTask()　AO_StartTask()　AO_GetStatus()　AO_GetStatus()　AO_WaitUntilTaskDone()　AO_WriteAnalog()　AO_WriteBinary()　AO_StopTask()　AO_ReleaseTask()　DEV_Release()

AO_StartTask()

函数原型：

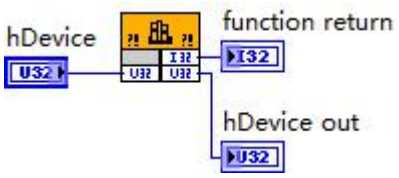
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AO_StartTask (HANDLE hDevice)

Visual Basic:

Declare Function AO_StartTask Lib "USB2861" (ByVal hDevice As Long) As Boolean

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：开始 AO 采集(Start task for analog output)，必须在成功调用 AO_InitTask()函数后才能调用此函数，调用该函数后 AO 立即准备就绪，但 AO 实际是否进入采样记录状态，须依赖于触发事件的产生。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

返回值：如果调用成功，则返回 TRUE，AO 立刻被开始， 否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AO_InitTask() AO_StartTask()
AO_GetStatus() AO_GetStatus() AO_WaitUntilTaskDone()
AO_WriteAnalog() AO_WriteBinary() AO_StopTask()
AO_ReleaseTask() DEV_Release()

AO_SendSoftTrig()

函数原型：

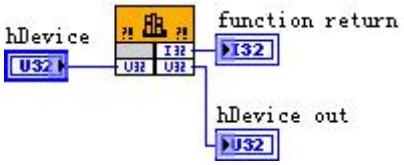
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AO_SendSoftTrig (HANDLE hDevice)

Visual Basic:

Declare Function AO_SendSoftTrig Lib "USB2861" (ByVal hDevice As Long) As Boolean

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：产生强制触发事件(Send software trigger event for analog output)。在开始 AO 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，由 DEV_Create()函数创建，该句柄指向要访问的设备。

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

相关函数： DEV_Create() AO_InitTask() AO_StartTask()
AO_GetStatus() AO_GetStatus() AO_WaitUntilTaskDone()
AO_WriteAnalog() AO_WriteBinary() AO_StopTask()
AO_ReleaseTask() DEV_Release()

AO_GetStatus()

函数原型：

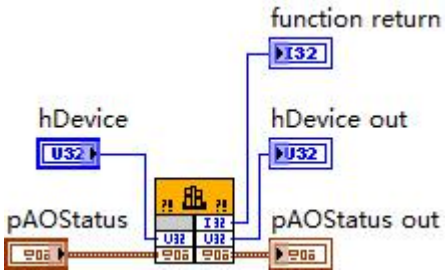
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AO_GetStatus (HANDLE hDevice, PAO_STATUS pAOStatus)

Visual Basic:

Declare Function AO_GetStatus Lib "USB2861" (ByVal hDevice As Long, _
ByRef pAOStatus As AO_STATUS) As Boolean

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：取得 AO 的各种状态(Get status for analog output)。一旦调用 AO_StartTask()函数后，可以调用此函数查询 AO 状态去同步采样数据的写操作。nAvailSampsPerChan>1 时，表示至少可以往生成任务中写入 1 个点的数据。如果在开始生成任务后，设备迟迟不能被触发采样，可以根据需要调用 AO_SendSoftTrig()函数以强制触发设备采样，很快得到有效的可写入采样数据点数。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pAOStatus 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《4 各种结构体描述》\《4.6 AO_PARAM（AO 工作参数结构体）》。

返回值：如果成功获取 AO 状态，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AO_InitTask() AO_StartTask()
AO_GetStatus() AO_GetStatus() AO_WaitUntilTaskDone()
AO_WriteAnalog() AO_WriteBinary() AO_StopTask()
AO_ReleaseTask() DEV_Release()

AO_WaitUntilTaskDone()

函数原型：

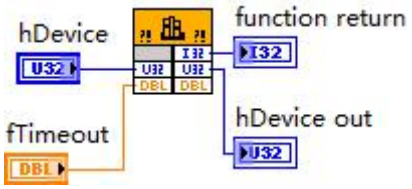
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AO_WaitUntilTaskDone (HANDLE hDevice, F64 fTimeout)

Visual Basic:

Declare Function AO_WaitUntilTaskDone Lib "USB2861" (ByVal hDevice As Long, _
ByVal fTimeout As Double) As Boolean

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：在 AO 的生成任务结束前等待(Wait until task done for analog output)。一旦调用 AO_StartTask()函数后，可以调用此函数等待生成任务结束。它通常用在有限点采样模式中。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

fTimeout 入口参数，超时时间，单位：秒（S）。指定该次等待所用时间，比如设定为 10.0，即 10 秒钟的时间，如果在 10 秒内生成任务结束，则函数立即返回 TRUE，否则 10 秒钟后函数返回值 FALSE，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即总是等到生成任务结束才返回）则赋值小于 0 即可，如-1.0；如果 fTimeout=0.0，则意味着该函数只是简单查询生成任务是否结束，如果生成任务结束了，则返回 TRUE，否则返回 FALSE。

返回值：如果生成任务结束，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AO_InitTask() AO_StartTask()
AO_GetStatus() AO_GetStatus() AO_WaitUntilTaskDone()
AO_WriteAnalog() AO_WriteBinary() AO_StopTask()
AO_ReleaseTask() DEV_Release()

AO_WriteAnalog()

函数原型:

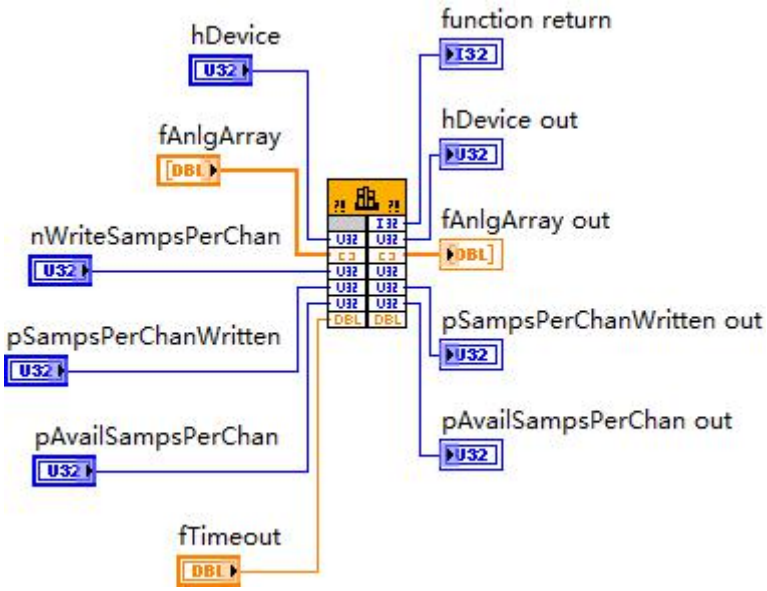
Visual C++ / C++Builder / LabWindows/CVI:

```
LONG AO_WriteAnalog(HANDLE hDevice,
                    F64 fAnlgArray[],
                    U32 nWriteSampsPerChan,
                    U32* pSampsPerChanWritten,
                    U32* pAvailSampsPerChan,
                    F64 fTimeout);
```

Visual Basic:

```
Declare Function AO_WriteAnalog Lib "USB2861" ( ByVal hDevice As Long, _
                                                ByRef fAnlgArray As Double, _
                                                ByVal nWriteSampsPerChan As Long, _
                                                ByRef pSampsPerChanWritten As Long,_
                                                ByRef pAvailSampsPerChan As Long,_
                                                ByVal fTimeout As Long) As Long
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：写入模拟量采样数据(主要是电压数据) （Write analog data to the task）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

fAnlgArray 入口参数，用户缓冲区，用于接收模拟量数据，值区间由相应采样通道的选用采样范围决定，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 nWriteSampsPerChan*nSampChanCount(实际采样通道数)。

nWriteSampsPerChan 入口参数，每通道请求写入的数据点数。

在有限点和连续采样模式中，它指定该次从设备的当前可写数据位置写入的数据点数（单位：点）。注意此参数的值如大于当前的可读数点 nAvailSampsPerChan 则会继续等待直到至少有 nWriteSampsPerChan 个点写入后写函数才会返回。等待期间，如果所等时间超过 fTimeout 指定时间也会返回，并置超时错误码。

在连续采样过程中，如果置波形非重生成模式，如果要保持连续不丢点，此参数应尽可能接近于甚至等于当前的可读点数(nAvailSampsPerChan),但不能大于 AOParam.nSampsPerChan。当然此参数值也不能大于 fAnlgArray 的缓冲区长度，所以为避免出错，所开辟的缓冲区不能小于 nWriteSampsPerChan*nSampChanCount(实际采样通道数)。

pSampsPerChanWritten 出口参数，返回每通道实际写入的点数。在单点采样模式中，如果写入成功，返回的每通道已写入点数总是为 1。注意每次都要检查 pSampsPerChanRead 的返回值，如果返回值等于 0，则要慎重处理。

pAvailSampsPerChan 出口参数，返回该次写操作完成时的每通道还可写而未写的数据点数。它跟 AO_GetStatus()函数取得的状态信息 AOStatus.nAvailSampsPerChan 是同一个状态信息。返回可读点数的意义在于通过数据写操作直接提供给用户，避免再次调用 AO_GetStatus()函数，即简化了实现方法，又提高了效率，即在读数据函数返回时判断可写点数，如果大于 nWriteSampsPerChan，则可紧接着再次调用写数据函数，直到 pAvailSampsPerChan 返回值小于 nWriteSampsPerChan。在单点采样模式中,它的返回值总是为 0。

fTimeout 入口参数，超时时间，单位：秒（S）。指定等待写入额定点数的时间。比如设定为 10.0，如果在 10 秒内写入点数达到 nWriteSampsPerChan 时立即返回 TRUE，否则 10 秒钟后函数返回值 FALSE，并通过 pAvailSampsPerChan 告之实际写入的点数，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即等待请求数据点数完

全写入任务中才返回）则置为负数，如-1.0 即可。如果 fTimeout=0.0，则意味着该函数仅简单判断能否立即写入请求的点数，如果不能，则不等待，立即返回 FALSE，否则返回 TRUE。

返回值： 如果函数调用成功则返回 TRUE，否则返回 FALSE，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_AVAILABLE_SAMPS	0xE0000000+1	无有效点数据
ERROR_SAMPLE_TASK_FAIL	0xE0000000+2	生成任务失败
ERROR_TIMEOUT	1460L	超时错误(见微软定义 WinError.h)

相关函数： DEV_Create() AO_InitTask() AO_StartTask() AO_GetStatus() AO_GetStatus() AO_WaitUntilTaskDone() AO_WriteAnalog() AO_WriteBinary() AO_StopTask() AO_ReleaseTask() DEV_Release()

AO_WriteBinary()

函数原型:

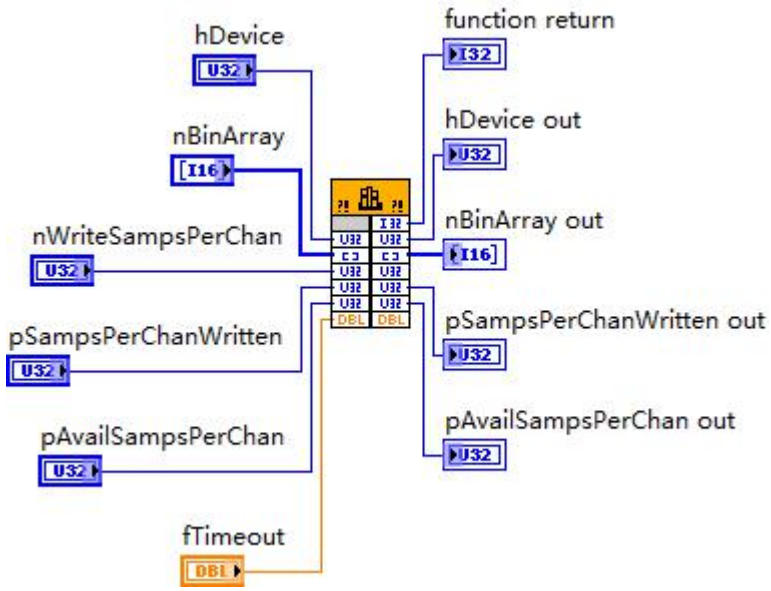
Visual C++ / C++Builder / LabWindows/CVI:

```
LONG AO_WriteBinary(HANDLE hDevice,
                    I16 nBinArray[],
                    U32 nWriteSampsPerChan,
                    U32* pSampsPerChanWritten,
                    U32* pAvailSampsPerChan,
                    F64 fTimeout);
```

Visual Basic:

```
Declare Function AO_WriteBinary Lib "USB2861" ( ByVal hDevice As Long, _
                                                ByRef nBinArray As Integer, _
                                                ByVal nWriteSampsPerChan As Long, _
                                                ByRef pSampsPerChanWritten As Long,_
                                                ByRef pAvailSampsPerChan As Long,_
                                                ByVal fTimeout As Long) As Long
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi

功能：写入二进制原码采样数据（Write binary data to the task）。它是将用户二进制原码数据写入生成任务中。二进制原码具有数据紧凑、数据量小、传输快、处理快、存盘也快的特点。

参数：

hDevice 同 AO_WriteAnalog()。

nBinArray 入口参数，用户缓冲区，用于接收二进制原码数据，值区间[-32768，32767]。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 nWriteSampsPerChan*nSampChanCount(实际采样通道数)。

nWriteSampsPerChan 同 AO_WriteAnalog()。

pSampsPerChanWritten 同 AO_WriteAnalog()。

pAvailSampsPerChan 同 AO_WriteAnalog()。

fTimeout 同 AO_WriteAnalog()。

返回值：同 AO_WriteAnalog()。

相关函数： DEV_Create() AO_InitTask() AO_StartTask() AO_GetStatus() AO_GetStatus() AO_WaitUntilTaskDone() AO_WriteAnalog() AO_WriteBinary() AO_StopTask() AO_ReleaseTask() DEV_Release()

AO_ReadbackAnalog()

函数原型:

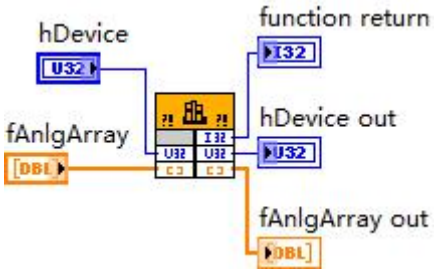
Visual C++ / C++Builder / LabWindows/CVI:

```
LONG AO_ReadbackAnalog(HANDLE hDevice,
                       F64 fAnlgArray[2]);
```

Visual Basic:

```
Declare Function AO_ReadbackAnalog Lib "USB2861" ( ByVal hDevice As Long, _
                                                   ByRef fAnlgArray As Double) As Long
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi

功能：回读 AO 的电压数据（Readback voltage data from the task）。

参数：

hDevice **hDevice** 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nAnlgArray 出口参数，用户缓冲区，用于接收回读的电压数据。nAnlgArray[0]为 AO0 的数据，nAnlgArray[1]为 AO1 的数据。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AO_InitTask() AO_StartTask()
AO_GetStatus() AO_GetStatus() AO_WaitUntilTaskDone()
AO_WriteAnalog() AO_WriteBinary() AO_StopTask()
AO_ReleaseTask() AO_ReadbackAnalog() AO_ReadbackBinary()
DEV_Release()

AO_ReadbackBinary()

函数原型:

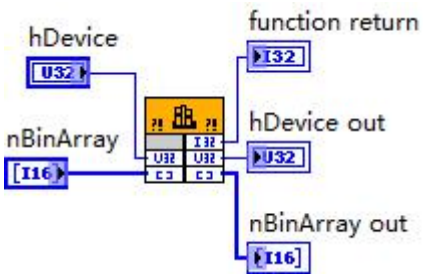
Visual C++ / C++Builder / LabWindows/CVI:

```
LONG AO_ReadbackBinary(HANDLE hDevice,
                       I16 nBinArray[2]);
```

Visual Basic:

```
Declare Function AO_ReadbackBinary Lib "USB2861" ( ByVal hDevice As Long, _
                                                   ByRef nBinArray As Integer) As Long
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi

功能：回读 AO 的电压数据（Readback binary data from the task）。

参数：

hDevice　**hDevice** 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nBinArray 出口参数，用户缓冲区，用于接收回读的二进制原码数据。nBinArray[0]为 AO0 的数据，nBinArray [1]为 AO1 的数据。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：DEV_Create()　AO_InitTask()　AO_StartTask()　AO_GetStatus()　AO_GetStatus()　AO_WaitUntilTaskDone()　AO_WriteAnalog()　AO_WriteBinary()　AO_StopTask()　AO_ReleaseTask()　AO_ReadbackAnalog()　AO_ReadbackBinary()　DEV_Release()

AO_StopTask()

函数原型：

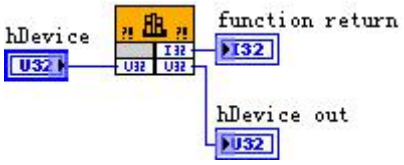
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AO_StopTask (HANDLE hDevice)

Visual Basic:

Declare Function AO_StopTask Lib "USB2861" (ByVal hDevice As Long)

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：停止 AO 生成任务(Stop task for analog output)。必须在成功调用 AO_StartTask()函数后才能调用此函数。该函数除了停止 AO 生成外不改变设备的其他状态。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

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

相关函数：DEV_Create()　AO_InitTask()　AO_StartTask()　AO_GetStatus()　AO_GetStatus()　AO_WaitUntilTaskDone()　AO_WriteAnalog()　AO_WriteBinary()　AO_StopTask()　AO_ReleaseTask()　DEV_Release()

AO_ReleaseTask()

函数原型：

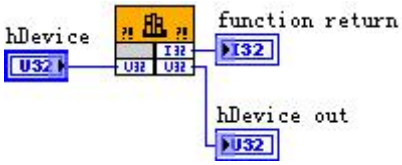
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AO_ReleaseTask(HANDLE hDevice)

Visual Basic:

Declare Function AO_ReleaseTask Lib "USB2861" (ByVal hDevice As Long)

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

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

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create() AO_InitTask() AO_StartTask()
AO_GetStatus() AO_GetStatus() AO_WaitUntilTaskDone()
AO_WriteAnalog() AO_WriteBinary() AO_StopTask()
AO_ReleaseTask() DEV_Release()

AO_ScaleBinToVolt()

函数原型：

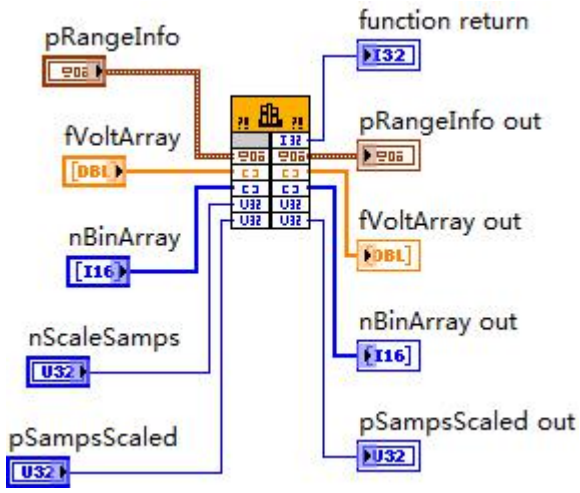
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AO_ScaleBinToVolt(
                    PAO_VOLT_RANGE_INFO pRangeInfo
                    F64 fVoltArray[],
                    I16 nBinArray[],
                    U32 nScaleSamps,
                    U32* pSampsScaled);
```

Visual Basic:

```
Declare Function AO_ScaleBinToVolt Lib "USB2861" (
                                    ByRef pRangeInfo As AO_VOLT_RANGE_INFO,
_
                                    ByRef fVoltArray As Double, _
                                    ByRef nBinArray As Integer, _
                                    ByVal nScaleSamps As Long, _
                                    ByRef pSampsScaled As Long) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：将二进制原码数据转换为电压数据(Scale binary data to voltage data)。与 AO_ScaleVoltToBin()函数的功能相反。

参数：

pRangeInfo 入口参数，范围信息，它由 AO_GetVoltRangeInfo()函数获取。

fVoltArray 出口参数，用于返回量化后的电压数据，单位：伏（V），取值范围由参数 pRangeInfo 指定。

nBinArray 入口参数，用于传入待量化的原码数据，取值范围[-32768, 32767]。

nScaleSamps 入口参数，请求量化的数据点数。

pSampsScaled 出口参数，返回实际量化后数据点数。

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

相关函数： DEV_Create() AO_ScaleBinToVolt() AO_ScaleVoltToBin() AO_GetMainInfo() AO_GetVoltRangeInfo() AO_GetRateInfo() DEV_Release()

AO_ScaleVoltToBin()

函数原型：

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

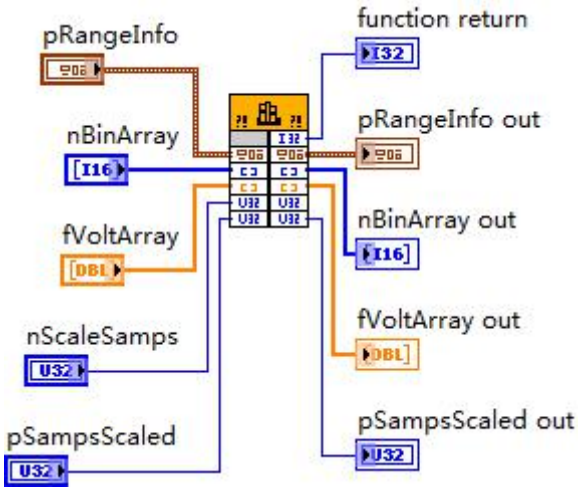
```
BOOL AO_ScaleVoltToBin(
                    ByRef pRangeInfo As AO_VOLT_RANGE_INFO, _
                    I16 nBinArray[],
                    F64 fVoltArray[],
                    U32 nScaleSamps,
                    U32* pSampsScaled);
```

Visual Basic:

```
Declare Function AO_ScaleBinToVolt Lib "USB2861" (
                                    ByRef pRangeInfo As AO_VOLT_RANGE_INFO,
_
```

ByRef nBinArray As Integer, _
ByRef fVoltArray As Double, _
ByVal nScaleSamps As Long, _
ByRef pSampsScaled As Long) As Boolean

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：将电压数据转换为原码数据(Scale voltage data to binary data)。与 AO_ScaleBinToVolt()函数的功能相反。

参数：

pRangeInfo 入口参数，范围信息，它由 AO_GetVoltRangeInfo()函数获取。

nBinArray 出口参数，用于传入待量化的原码数据，取值范围[-32768, 32767]。

fVoltArray 入口参数，用于返回量化后的电压数据，单位：伏（V），取值范围由参数 pRangeInfo 指定。

nScaleSamps 入口参数，请求量化的数据点数。

pSampsScaled 出口参数，返回实际量化后数据点数。

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

相关函数： DEV_Create() AO_ScaleBinToVolt() AO_ScaleVoltToBin() AO_GetMainInfo() AO_GetVoltRangeInfo() AO_GetRateInfo() DEV_Release()

AO_GetMainInfo()

函数原型：

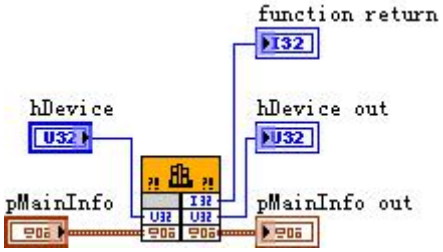
Visual C++ / C++Builder / LabWindows/CVI:

BOOL AO_GetMainInfo (HANDLE hDevice,
PAO_MAIN_INFO pMainInfo)

Visual Basic:

Declare Function AO_GetMainInfo Lib "USB2861" (ByVal hDevice As Long, _
ByRef pMainInfo As AO_MAIN_INFO) As Boolean

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：取得 AO 功能的主要信息，如通道数、分辨率等（Get main information for analog output）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pMainInfo 出口参数，AO 主要信息结构指针，负责返回 AO 的主要描述信息。关于 AO_MAIN_INFO 的详细介绍请参考 USB2861.h 或 USB2861.bas 或 USB2861.pas 函数原型定义的头文件，也可参考本文《4.6　AO_PARAM（AO 工作参数结构体）》关于该结构的有关说明。

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

相关函数： DEV_Create()　AO_ScaleBinToVolt()　AO_ScaleVoltToBin()　AO_GetMainInfo()　AO_GetVoltRangeInfo()　AO_GetRateInfo()　DEV_Release()

AO_GetVoltRangeInfo()

函数原型：

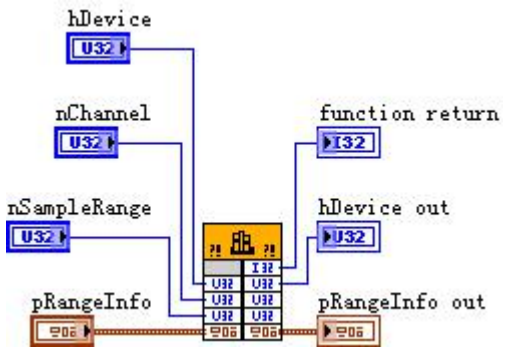
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AO_GetVoltRangeInfo(HANDLE hDevice,
                         int nChannel,
                         int nSampleRange,
                         PAO_VOLT_RANGE_INFO pRangeInfo)
```

Visual Basic:

```
Declare Function AO_GetVoltRangeInfo Lib "USB2861" (ByVal hDevice As Long, _
                                                    ByVal nChannel As Long, _
                                                    nSampleRange As Long, _
                                                    ByRef pRangeInfo As AO_VOLT_RANGE_INFO) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：取得 AO 指定通道、指定采样范围档的上下限电压、幅度等信息（Get range information for analog output）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，物理通道号，取值范围[0，3]。

nSampleRange 入口参数，AO 采样范围，同 AOParam.nSampleRange。

pRangeInfo 出口参数，AO 采样范围信息，负责返回 AO 的采样范围大值、最小值、幅度、编码宽度等。请参考《4.10　AO_VOLT_RANGE_INFO (AO 采样范围信息结构体)》

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

相关函数： DEV_Create()　AO_ScaleBinToVolt()　AO_ScaleVoltToBin()　AO_GetMainInfo()　AO_GetVoltRangeInfo()　AO_GetRateInfo()　DEV_Release()

AO_GetRateInfo()

函数原型：

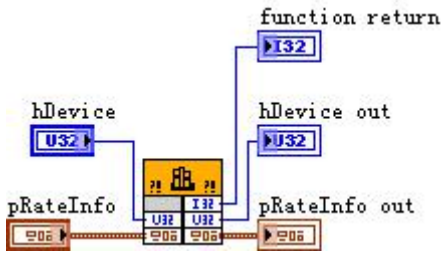
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AO_GetRateInfo(HANDLE hDevice,
                    AO_RATE_INFO pRateInfo);
```

Visual Basic:

```
Declare Function AO_GetRateInfo Lib "USB2861" (ByVal hDevice As Long, _
                                               ByRef pRateInfo As AO_RATE_INFO) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：获得采样速率信息（Get rate information for analog output）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pRateInfo 出口参数，AO 采样速率信息，负责返回 AO 的最大采样率，最小采样率等。详情请参考《4.10　AO_SAMP_RATE_INFO (AO 采样速率信息结构体)》。

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

相关函数： DEV_Create()　AO_ScaleBinToVolt()　AO_ScaleVoltToBin()　AO_GetMainInfo()　AO_GetVoltRangeInfo()　AO_GetRateInfo()　DEV_Release()

AO_VerifyParam()

函数原型：

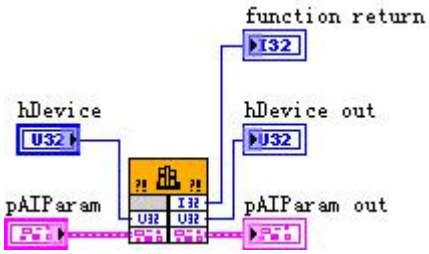
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AO_VerifyParam (HANDLE hDevice,
                     PAO_PARAM pAOParam);
```

Visual Basic:

```
Declare Function AO_VerifyParam Lib "USB2861" (ByVal hDevice As Long, _
                                               ByRef pAOParam As AO_PARAM) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：校验 AO 工作参数(Verify task parameter for analog output)，这是一个辅助功能的函数，在初始化 AO 前，事先校验 AO 参数的合法性。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pAOParam 入口和出口参数，AO 工作参数结构体指针，关于其具体定义及说明请参考《4　各种结构体描述》\《4.6　AO_PARAM（AO 工作参数结构体）》。函数调用时，它传入要校验的各项参数，函数返回时，它将经过调整为合法的参数值返回给调用者，以便后续初始化 AO 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 USB2861.log 记录不合法的参数名和原因，然后返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DEV_Create()　AO_InitTask()　AO_VerifyParam()　AO_LoadParam()　AO_SaveParam()　AO_ResetParam()　DEV_Release()

AO_LoadParam()

函数原型：

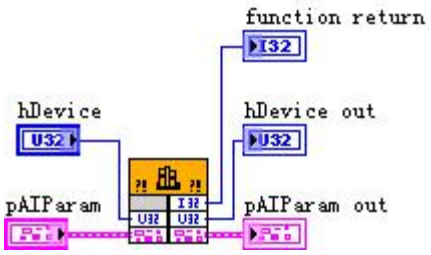
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AO_LoadParam (HANDLE hDevice,
                   PAO_PARAM pAOParam);
```

Visual Basic:

```
Declare Function AO_LoadParam Lib "USB2861" (ByVal hDevice As Long, _
                                             ByRef pAOParam As AO_PARAM) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：从 USB2861.ini 参数配置文件中读取设备的硬件参数(Load parameter from USB2861.ini for analog output)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pAOParam 出口参数，属于 PAO_PARAM 的结构指针类型，负责返回 AO 工作参数值，关于结构指针类型 PAO_PARAM 请参考 USB2861.h 或 USB2861.bas 或 USB2861.pas 函数原型定义的头文件，也可参考本文《4.6 AO_PARAM（AO 工作参数结构体）》。

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

相关函数： DEV_Create() AO_InitTask() AO_VerifyParam() AO_LoadParam() AO_SaveParam() AO_ResetParam() DEV_Release()

AO_SaveParam()

函数原型：

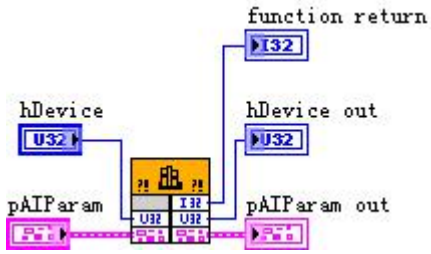
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AO_SaveParam (HANDLE hDevice,
                   PAO_PARAM pAOParam)
```

Visual Basic:

```
Declare Function AO_SaveParam Lib "USB2861" (ByVal hDevice As Long, _
                                             ByRef pAOParam As AO_PARAM) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：将配置好的 AO 工作参数保存在 USB2861.ini 参数配置文件中(Save parameter to USB2861.ini for analog output)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pAOParam 入口参数，AO 工作参数结构体指针，关于 AO_PARAM 的详细介绍请参考 USB2861.h 或 USB2861.bas 或 USB2861.pas 等函数原型定义的头文件，也可参考本文《4.6 AO_PARAM（AO 工作参数结构体）》。

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

相关函数： DEV_Create() AO_InitTask() AO_VerifyParam() AO_LoadParam() AO_SaveParam() AO_ResetParam() DEV_Release()

AO_ResetParam()

函数原型：

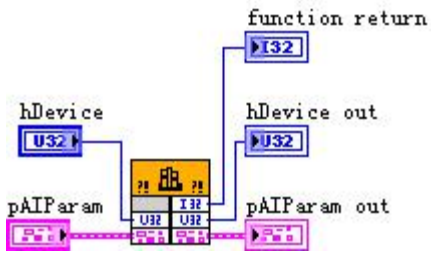
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL AO_ResetParam (HANDLE hDevice,
                    PAO_PARAM pAOParam)
```

Visual Basic:

```
Declare Function AO_ResetParam Lib "USB2861" ( ByVal hDevice As Long, _
                                               ByRef pAOParam As AO_PARAM) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：将 USB2861.ini 参数配置文件中原来的 AO 参数值复位至出厂时的默认值(Reset parameter to default value for analog output)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pAOParam 出口参数，AO 工作参数结构指针，负责在参数被复位后返回其复位后的参数值。关于 AO_PARAM 的详细介绍请参考 USB2861.h 或 USB2861.bas 或 USB2861.pas 函数原型定义的头文件，也可参考本文《4.6 AO_PARAM（AO 工作参数结构体）》。

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

相关函数： DEV_Create() AO_InitTask() AO_VerifyParam() AO_LoadParam() AO_SaveParam() AO_ResetParam() DEV_Release()

3.4 DI 数字量输入函数原型说明

DI_InitTask()

函数原型：

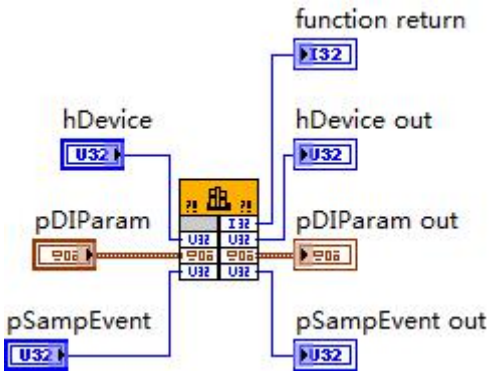
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DI_InitTask(HANDLE hDevice, PDI_PARAM pDIParam, HANDLE* pSampEvent);
```

Visual Basic:

```
Declare Function DI_InitTask Lib "USB2861" ( ByVal hDevice As Long, _
                                             ByVal pDIParam As DI_PARAM,_
                                             ByRef pSampEvent As Long) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：初始化 DI 采集任务(仅基于 Port0)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pDIParam 入口参数，端口工作参数结构体，详情参考《4.11 DI_PARAM (DI 数字量工作参数结构体)》。

pSampEvent 出口参数，返回采样事件对象句柄,当设备中出现可读数据段时会触发此事件，参数=NULL,表示不需要此事件句柄

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

相关函数：

DEV_Create() DI_InitTask() DI_StartTask()

DI_SendSoftTrig()	DI_GetStatus()	DI_WaitUntilTaskDone()
DI_ReadDigitalU8()	DI_ReadDigitalOneLines()	DI_ReadDigitalOneU8()
DI_StopTask()	DI_ReleaseTask()	DEV_Release()

DI_StartTask()

函数原型：

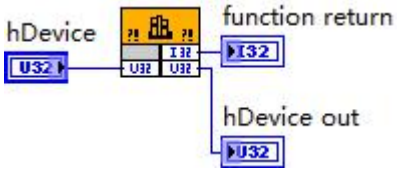
Visual C++ / C++Builder / LabWindows/CVI:

BOOL DI_StartTask(HANDLE hDevice)

Visual Basic:

Declare Function DI_StartTask Lib "USB2861" (ByVal hDevice As Long) As Boolean

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：开始 DI 采集(Start task for digital input)。必须在成功调用 DI_InitTask()函数后才能调用此函数，调用该函数后 DI 立即准备就绪，但 DI 实际是否进入采集记录过程，须依赖于触发事件的产生。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

返回值：如果调用成功，则返回 TRUE，DI 立刻被开始， 否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	DI_InitTask()	DI_StartTask()
DI_SendSoftTrig()	DI_GetStatus()	DI_WaitUntilTaskDone()
DI_ReadDigitalU8()	DI_ReadDigitalOneLines()	DI_ReadDigitalOneU8()
DI_StopTask()	DI_ReleaseTask()	DEV_Release()

DI_SendSoftTrig()

函数原型：

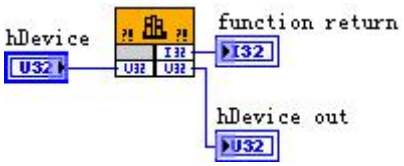
Visual C++ / C++Builder / LabWindows/CVI:

BOOL DI_SendSoftTrig (HANDLE hDevice)

Visual Basic:

Declare Function DI_SendSoftTrig Lib "USB2861" (ByVal hDevice As Long) As Boolean

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：发送软件触发事件(Send software trigger event for digital input)。设备进入等待触发状态，若用户需软件触发或需要随时手动给触发事件时，可调用此函数。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，由 DEV_Create()函数创建，该句柄指向要访问的设备。

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

相关函数：

DEV_Create()	DI_InitTask()	DI_StartTask()
DI_SendSoftTrig()	DI_GetStatus()	DI_WaitUntilTaskDone()
DI_ReadDigitalU8()	DI_ReadDigitalOneLines()	DI_ReadDigitalOneU8()
DI_StopTask()	DI_ReleaseTask()	DEV_Release()

DI_GetStatus()

函数原型：

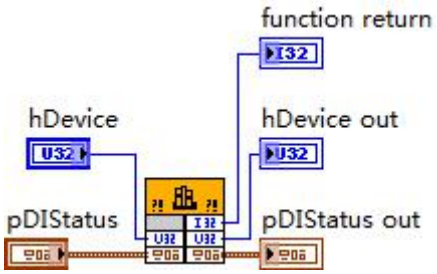
Visual C++ / C++Builder / LabWindows/CVI:

BOOL DI_GetStatus (HANDLE hDevice, PDI_STATUS pDIStatus)

Visual Basic:

```
Declare Function DI_GetStatus Lib "USB2861" (ByVal hDevice As Long, _
                                              ByRef pDIStatus As DI_STATUS) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：取得 DI 的各种状态(Get status for digital input)。一旦调用 DI_StartTask()函数后，应立即调用此函数查询 DI 状态去同步采样数据的读操作。nAvailSampsPerChan>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 DI_ReadDigitalU8()函数循环读取若干可读段数据，直到 nAvailSampsPerChan>nReadSampsPerChan 时可调用读数函数。如果在开始采集任务后，设备迟迟不能被触发采样，可以根据需要调用 DI_SendSoftTrig()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。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pDIStatus 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《4.12 DI_STATUS（DI 工作状态信息结构）》。

返回值：如果成功获取 DI 状态，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	DI_InitTask()	DI_StartTask()
DI_SendSoftTrig()	DI_GetStatus()	DI_WaitUntilTaskDone()
DI_ReadDigitalU8()	DI_ReadDigitalOneLines()	DI_ReadDigitalOneU8()

DI_StopTask()　　DI_ReleaseTask()　　DEV_Release()

DI_WaitUntilTaskDone()

函数原型：

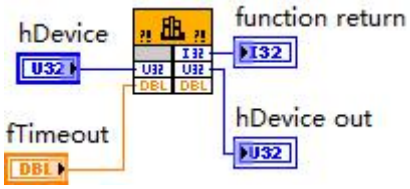
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DI_WaitUntilTaskDone (HANDLE hDevice, F64 fTimeout)
```

Visual Basic:

```
Declare Function DI_WaitUntilTaskDone Lib "USB2861" (ByVal hDevice As Long, _
                                                     ByVal fTimeout As Double) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：在 DI 的采集任务结束前等待(Wait until task done for analog input)。一旦调用 DI_StartTask()函数后，可以调用此函数等待采集任务结束。它通常用在有限点采样模式中。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

fTimeout 入口参数，超时时间，单位：秒（S）。指定该次等待所用时间，比如设定为 10.0，即 10 秒钟的时间，如果在 10 秒内采集任务结束，则函数立即返回 TRUE，否则 10 秒钟后函数返回值 FALSE，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即总是等到采集任务结束才返回）则赋值小于 0 即可，如-1.0；如果 fTimeout=0.0，则意味着该函数只是简单查询采集任务是否结束，如果采集任务结束了，则返回 TRUE，否则返回 FALSE。

返回值：如果采集任务结束，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()　　DI_InitTask()　　DI_StartTask()
DI_SendSoftTrig()　　DI_GetStatus()　　DI_WaitUntilTaskDone()
DI_ReadDigitalU8()　　DI_ReadDigitalOneLines()　　DI_ReadDigitalOneU8()
DI_StopTask()　　DI_ReleaseTask()　　DEV_Release()

DI_ReadDigitalU8()

函数原型:

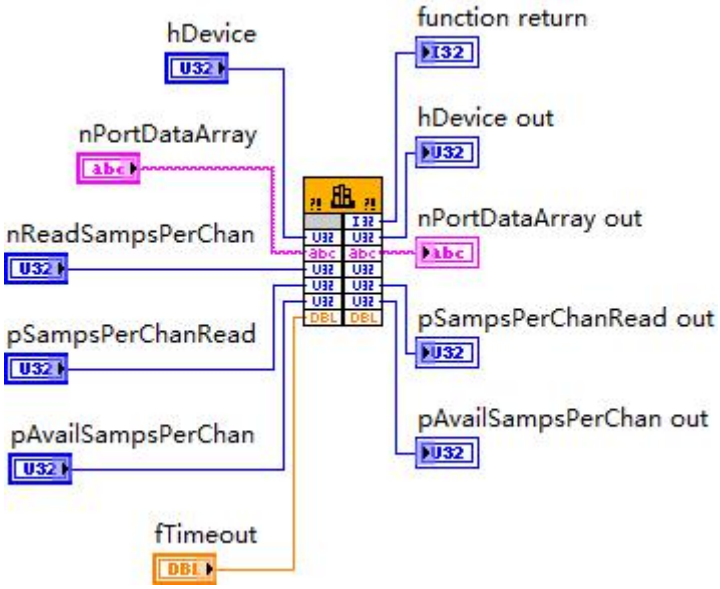
Visual C++ / C++Builder / LabWindows/CVI:

```
LONG DI_ReadDigitalU8 (HANDLE hDevice,
                       U8 nPortDataArray[],
                       U32 nReadSampsPerChan,
                       U32* pSampsPerChanRead,
                       U32* pAvailSampsPerChan,
                       F64 fTimeout);
```

Visual Basic:

```
Declare Function DI_ReadDigitalU8 Lib "USB2861" ( ByVal hDevice As Long, _
                                                  ByRef nPortDataArray As Byte, _
                                                  ByVal nReadSampsPerChan As Long, _
                                                  ByRef pSampsPerChanRead As Long,_
                                                  ByRef pAvailSampsPerChan As Long,_
                                                  ByVal fTimeout As Long) As Long
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取数字量采样数据（Read digital data from the task）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nPortDataArray 出口参数，用户缓冲区，用于接收 Port0 端口二进制原码数据。它点空间不能小于 nReadSampsPerChan。

nReadSampsPerChan 入口参数，每通道(端口)请求读入的数据点数。

在有限点和连续采样模式中，它指定该次从设备的当前可读数据位置读取的数据点数（单位：点）。注意此参数的值如大于当前的可读数点 nAvailSampsPerChan 则会继续等待直到至少有 nReadSampsPerChan 个点可读后读函数才会返回。等待期间，如果所等时间超过 fTimeout 指定时间也会返回，并置超时错误码。

在连续采样过程中，如果要保持连续不丢点，此参数应尽可能接近于甚至等于当前的可读点数(nAvailSampsPerChan),但不能大于 DIParam.nSampsPerChan。当然此参数值也不能大于 fAnlgArray 的缓冲区长度，所以为避免出错，所开辟的缓冲区不能小于 nReadSampsPerChan。

pSampsPerChanRead 出口参数，返回每通道实际读取的点数。在单点采样模式中，如果读取成功，返回的每通道已读取点数总是为 1。注意每次都要检查 pSampsPerChanRead 的返回值，如果返回值等于 0，则要慎重处理。

pAvailSampsPerChan 出口参数，返回该次读操作完成时的每通道还可读而未读的数据点数。它跟 DI_GetStatus()函数取得的状态信息 DIStatus.nAvailSampsPerChan 是同一个状态信息。返回可读点数的意义在于通过数据读操作直接提供给用户，避免再次调用 DI_GetStatus()函数，即在读数据函数返回时判断可读点数，若大于 nReadSampsPerChan，则可紧接着再次调用读数据函数，直到 pAvailSampsPerChan 返回值小于 nReadSampsPerChan。在单点采样模式中,它的返回值总是为 0。

fTimeout 入口参数，超时时间，单位：秒（S）。指定等待写入额定点数的时间。比如设定为 10.0，如果在 10 秒内读取的点数达到 nReadSampsPerChan 时立即返回 TRUE，否则 10 秒钟后函数返回 FALSE，并通过 pAvailSampsPerChan 告之实际读入的点数，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即等待请求数据点数完全从任务中读到才返回）则置为负数，如-1.0 即可。如果 fTimeout=0.0，则意味着该函数仅简单判断能否立即读取到请求的点数，如果不能，则不等待，立即返回 FALSE，否则返回 TRUE。

返回值：如果函数调用成功则返回 TRUE，否则返回 FALSE，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_AVAILABLE_SAMPS	0xE0000000+1	无有效点数据
ERROR_SAMPLE_TASK_FAIL	0xE0000000+2	采集任务失败
ERROR_TIMEOUT	1460L	超时错误(见微软定义 WinError.h)

相关函数：

DEV_Create()　DI_InitTask()　DI_StartTask()
DI_SendSoftTrig()　DI_GetStatus()　DI_WaitUntilTaskDone()
DI_ReadDigitalU8()　DI_ReadDigitalOneLines()　DI_ReadDigitalOneU8()
DI_StopTask()　DI_ReleaseTask()　DEV_Release()

DI_ReadDigitalOneLines()

函数原型:

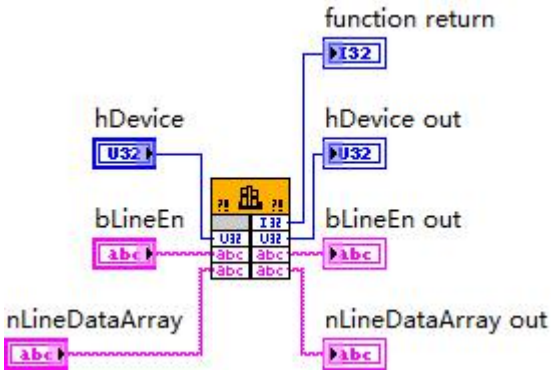
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DI_ReadDigitalU8 (HANDLE hDevice,
                       U8 bLineEn[24],
                       U8 nLineDataArray[]);
```

Visual Basic:

```
Declare Function DI_ReadDigitalU8 Lib "USB2861" ( ByVal hDevice As Long, _
                                                  ByRef bLineEn As Byte, _
                                                  ByVal nLineDataArray As Byte) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取数字量线数据（Read digital line data from the task）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。。

bLineEn[24] 出口参数，线号选择, bLineEn[0]=1 选择 P0.0, bLineEn[1]=1 选择 P0.1, bLineEn[2]=1

选择 P0.2... bLineEn[3]=1 选择 P1.0。

nLineDataArray [] 出口参数，线数据缓冲。

返回值：如果函数调用成功则返回 TRUE，否则返回 FALSE，可以调用 Win32 API 函数 GetLastError()以取得进一步的错误码信息 nErrorCode。

相关函数：

DEV_Create()	DI_InitTask()	DI_StartTask()
DI_SendSoftTrig()	DI_GetStatus()	DI_WaitUntilTaskDone()
DI_ReadDigitalU8()	DI_ReadDigitalOneLines()	DI_ReadDigitalOneU8()
DI_StopTask()	DI_ReleaseTask()	DEV_Release()

DI_ReadDigitalOneU8()

函数原型:

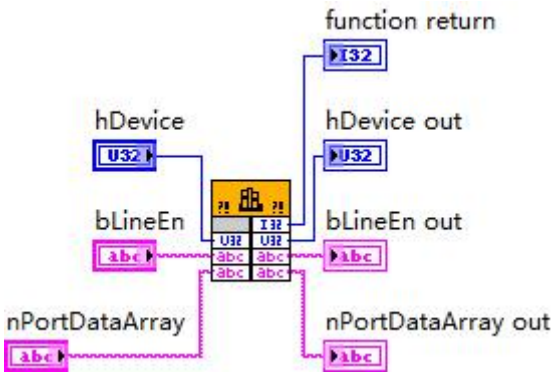
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DI_ReadDigitalOneU8(HANDLE hDevice,
                         U8 bLineEn[24],
                         U8 nPortDataArray[]);
```

Visual Basic:

```
Declare Function DI_ReadDigitalOneU8 Lib "USB2861" ( ByVal hDevice As Long, _
                                                     ByRef bLineEn As Byte, _
                                                     ByVal nPortDataArrayAs Byte) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取单点数字量数据（Read digital data from the task）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。。

bLineEn[24] 入口参数，线号使能, bLineEn[0]=1 选择 P0.0, bLineEn[1]=1 选择 P0.1, bLineEn[2]=1 选择 P0.2... bLineEn[3]=1 选择 P1.0。

nPortDataArray[] 出口参数，端口数据缓冲，返回采样的数据

返回值：如果函数调用成功则返回 TRUE，否则返回 FALSE，可以调用 Win32 API 函数 GetLastError()以取得进一步的错误码信息 nErrorCode。

相关函数：

DEV_Create()	DI_InitTask()	DI_StartTask()
DI_SendSoftTrig()	DI_GetStatus()	DI_WaitUntilTaskDone()
DI_ReadDigitalU8()	DI_ReadDigitalOneLines()	DI_ReadDigitalOneU8()

DI_StopTask()　　DI_ReleaseTask()　　DEV_Release()

DI_StopTask()

函数原型：

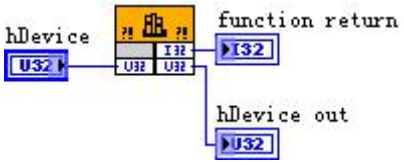
Visual C++ / C++Builder / LabWindows/CVI:

BOOL DI_StopTask (HANDLE hDevice)

Visual Basic:

Declare Function DI_StopTask Lib "USB2861" (ByVal hDevice As Long)

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：停止 DI 采样(Stop task for analog input)。必须在成功调用 DI_StartTask()函数后才能调用此函数。该函数除了停止 DI 采集外不改变设备的其他状态。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

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

相关函数：

DEV_Create()　　DI_InitTask()　　DI_StartTask()
DI_SendSoftTrig()　　DI_GetStatus()　　DI_WaitUntilTaskDone()
DI_ReadDigitalU8()　　DI_ReadDigitalOneLines()　　DI_ReadDigitalOneU8()
DI_StopTask()　　DI_ReleaseTask()　　DEV_Release()

DI_ReleaseTask()

函数原型：

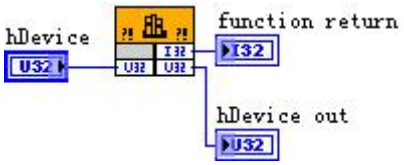
Visual C++ / C++Builder / LabWindows/CVI:

BOOL DI_ReleaseTask (HANDLE hDevice)

Visual Basic:

Declare Function DI_ReleaseTask Lib "USB2861" (ByVal hDevice As Long)

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

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

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数

GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()　　DI_InitTask()　　DI_StartTask()
DI_SendSoftTrig()　　DI_GetStatus()　　DI_WaitUntilTaskDone()
DI_ReadDigitalU8()　　DI_ReadDigitalOneLines()　　DI_ReadDigitalOneU8()
DI_StopTask()　　DI_ReleaseTask()　　DEV_Release()

DI_GetMainInfo()

函数原型：

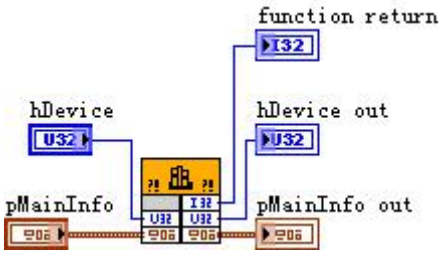
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DI_GetMainInfo (HANDLE hDevice,
                     PDI_MAIN_INFO pMainInfo)
```

Visual Basic:

```
Declare Function DI_GetMainInfo Lib "USB2861" ( ByVal hDevice As Long, _
                                                ByRef pMainInfo As DI_MAIN_INFO) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：取得 DI 功能的主要信息，如通道数、分辨率等（Get main information for analog input）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pMainInfo 出口参数，DI 主要信息结构指针，负责返回 DI 的主要描述信息。关于 DI_MAIN_INFO 的详细介绍请参考 USB2861.h 或 USB2861.bas 或 USB2861.pas 函数原型定义的头文件，也可参考本文《4.13　DI_MAIN_INFO (DI 主要信息结构体)》。

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

相关函数：

DEV_Create()　　DI_GetMainInfo()　　DI_GetRateInfo()　　DEV_Release()

DI_GetRateInfo()

函数原型：

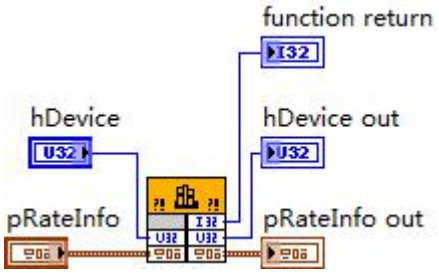
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DI_GetRateInfo(HANDLE hDevice, DI_RATE_INFO pRateInfo);
```

Visual Basic:

```
Declare Function DI_GetRateInfo Lib "USB2861" (ByVal hDevice As Long, _
                                               ByRef pRateInfo As DI_RATE_INFO) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：获得采样速率信息（Get rate information for analog input）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pRateInfo 出口参数，DI 采样速率信息，负责返回 DI 的最大采样率，最小采样率等。详情请参考《4.14 DI_SAMP_RATE_INFO (DI 采样速率信息结构体)》。

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

DEV_Create()　　DI_GetMainInfo()　　DI_GetRateInfo()　　DEV_Release()

DI_VerifyParam()

函数原型：

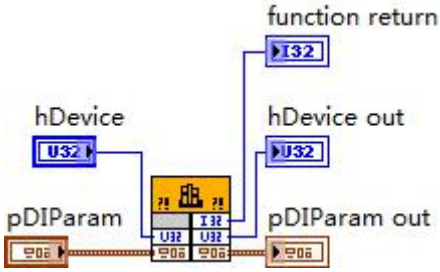
Visual C++ / C++Builder / LabWindows/CVI:

BOOL DI_VerifyParam (HANDLE hDevice, PDI_PARAM pDIParam)

Visual Basic:

Declare Function DI_VerifyParam Lib "USB2861" (ByVal hDevice As Long, _
ByRef pDIParam As DI_PARAM) As Boolean

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：校验 DI 工作参数(Verify task parameter for analog input)，这是一个辅助功能的函数，在初始化 DI 前，先校验 DI 参数的合法性。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pDIParam 入口和出口参数，DI 工作参数结构体指针，关于其具体定义及说明请参考《4.12 DI_PARAM (DI 数字量工作参数结构体)》。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 DI 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 USB2861.log 记录不合法的参数名和原因，然后返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DI_InitTask()　　DI_VerifyParam()　　DI_LoadParam()

DI_SaveParam()　　DI_ResetParam()

DI_LoadParam()

函数原型：

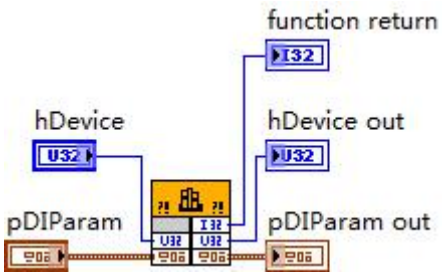
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DI_LoadParam (HANDLE hDevice,
                   PDI_PARAM pDIParam)
```

Visual Basic:

```
Declare Function DI_LoadParam Lib "USB2861" (ByVal hDevice As Long, _
                                             ByRef pDIParam As DI_PARAM) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：从 USB2861.ini 参数配置文件中读取设备的硬件参数(Load parameter from USB2861.ini for analog input)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pDIParam 出口参数，属于 PDI_PARAM 的结构指针类型，负责返回 DI 工作参数值，关于结构指针类型 PDI_PARAM 请参考 USB2861.h 或 USB2861.bas 或 USB2861.pas 函数原型定义的头文件，也可参考本文《4.12　DI_PARAM (DI 数字量工作参数结构体)》。

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

相关函数：　DI_InitTask()　　DI_VerifyParam()　　DI_LoadParam()
DI_SaveParam()　　DI_ResetParam()

DI_SaveParam()

函数原型：

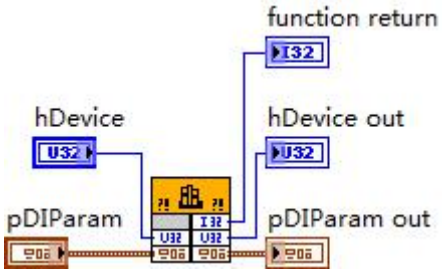
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DI_SaveParam (HANDLE hDevice,
                   PDI_PARAM pAIParam)
```

Visual Basic:

```
Declare Function DI_SaveParam Lib "USB2861" (ByVal hDevice As Long, _
                                             ByRef pAIParam As DI_PARAM) As Boolean
```

LabVIEW:



请参考 USB2861.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 USB2861.ini 参数配置文件中(Save parameter to USB2861.ini for analog input)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pDIParam 入口参数，DI 工作参数结构体指针，关于 DI_PARAM 的详细介绍请参考 USB2861.h 或 USB2861.bas 或 USB2861.pas 等函数原型定义的头文件，也可参考本文《4.12 DI_PARAM (DI 数字量工作参数结构体)》。

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

相关函数： DI_InitTask() DI_VerifyParam() DI_LoadParam()
DI_SaveParam() DI_ResetParam()

DI_ResetParam()

函数原型：

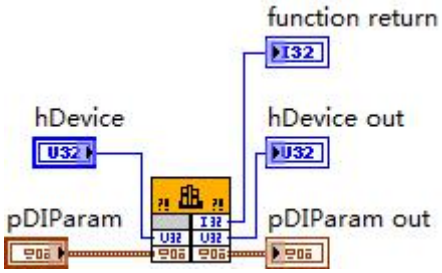
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DI_ResetParam (HANDLE hDevice,
                    PDI_PARAM pDIParam)
```

Visual Basic:

```
Declare Function DI_ResetParam Lib "USB2861" ( ByVal hDevice As Long, _
                                               ByRef pDIParam As DI_PARAM) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：将 USB2861.ini 参数配置文件中原来的 DI 参数值复位至出厂时的默认值(Reset parameter to default value for analog input)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pDIParam 出口参数，DI 工作参数结构指针，负责在参数被复位后返回其复位后的参数值。关于 DI_PARAM 的详细介绍请参考 USB2861.h 或 USB2861.bas 或 USB2861.pas 函数原型定义的头文

件，也可参考本文《4.12　DI_PARAM (DI 数字量工作参数结构体)》。

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

相关函数： DI_InitTask()　DI_VerifyParam()　DI_LoadParam()
DI_SaveParam()　DI_ResetParam()

3.5　DO 数字量输出函数原型说明

DO_InitTask()

函数原型：

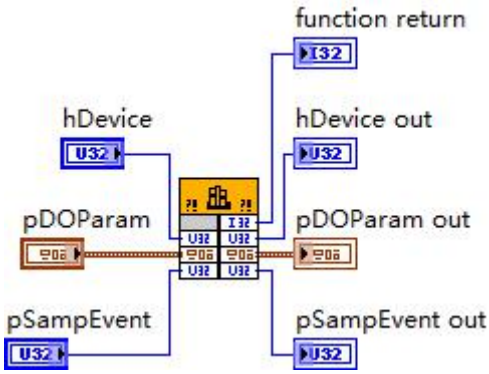
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DO_InitTask(HANDLE hDevice, PDO_PARAM pDOParam, HANDLE* pSampEvent);
```

Visual Basic:

```
Declare Function DO_InitTask Lib "USB2861" ( ByVal hDevice As Long, _
                                             ByVal pDOParam As DO_PARAM,_
                                             ByRef pSampEvent As Long) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：初始化 DO 生成任务(仅基于 Port0)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pDOParam 入口参数，端口工作参数结构体，详情参考《4.15　DO_PARAM (DO 数字量工作参数结构体)》。

pSampEvent 出口参数，返回采样事件对象句柄,当设备中出现可读数据段时会触发此事件，参数=NULL,表示不需要此事件句柄

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

相关函数：

DEV_Create()　DO_InitTask()　DO_StartTask()
DO_SendSoftTrig()　DO_GetStatus()　DO_WaitUntilTaskDone()
DO_WriteDigitalU8()　DO_WriteDigitalOneU32()　DO_WriteDigitalOneLines()
DO_WriteDigitalOneLine()　DO_ReadbackDigitalOneU8()　DO_StopTask()
DO_ReleaseTask()　DEV_Release()

DO_StartTask()

函数原型：

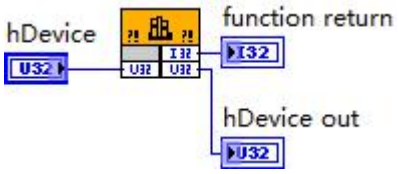
Visual C++ / C++Builder / LabWindows/CVI:

BOOL DO_StartTask (HANDLE hDevice)

Visual Basic:

Declare Function DO_StartTask Lib "USB2861" (ByVal hDevice As Long) As Boolean

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：开始 DO 采集(Start task for digital input)。必须在成功调用 DO_InitTask()函数后才能调用此函数，调用该函数后 DO 立即准备就绪，但 DO 实际是否进入采集记录过程，须依赖于触发事件的产生。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

返回值：如果调用成功，则返回 TRUE，DO 立刻被开始， 否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	DO_InitTask()	DO_StartTask()
DO_SendSoftTrig()	DO_GetStatus()	DO_WaitUntilTaskDone()
DO_WriteDigitalU8()	DO_WriteDigitalOneU32()	DO_WriteDigitalOneLines()
DO_WriteDigitalOneLine()	DO_ReadbackDigitalOneU8()	DO_StopTask()
DO_ReleaseTask()	DEV_Release()	

DO_SendSoftTrig()

函数原型：

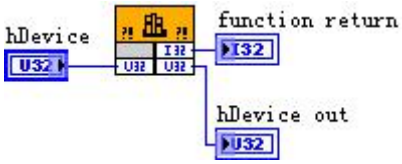
Visual C++ / C++Builder / LabWindows/CVI:

BOOL DO_SendSoftTrig (HANDLE hDevice)

Visual Basic:

Declare Function DO_SendSoftTrig Lib "USB2861" (ByVal hDevice As Long) As Boolean

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：发送软件触发事件(Send software trigger event for digital input)。设备进入等待触发状态，若用户需软件触发或需要随时手动给触发事件时，可调用此函数。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，由 DEV_Create()函数创建，该句柄指向要访问的设备。

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

相关函数：

DEV_Create()	DO_InitTask()	DO_StartTask()
DO_SendSoftTrig()	DO_GetStatus()	DO_WaitUntilTaskDone()
DO_WriteDigitalU8()	DO_WriteDigitalOneU32()	DO_WriteDigitalOneLines()
DO_WriteDigitalOneLine()	DO_ReadbackDigitalOneU8()	DO_StopTask()
DO_ReleaseTask()	DEV_Release()	

DO_GetStatus()

函数原型：

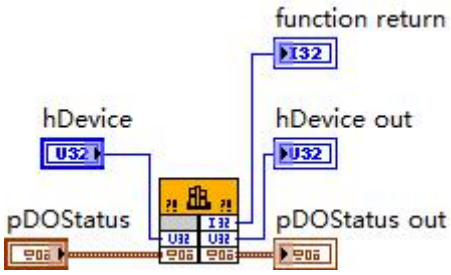
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DO_GetStatus (HANDLE hDevice, PDO_STATUS pDOStatus )
```

Visual Basic:

```
Declare Function DO_GetStatus Lib "USB2861" (ByVal hDevice As Long, _
                                             ByRef pDOStatus As DO_STATUS) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：取得 DO 的各种状态(Get status for digital input)。一旦调用 DO_StartTask()函数后，应立即调用此函数查询 DO 状态去同步采样数据的读操作。nAvailSampsPerChan>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 DO_ReadDigitalU8()函数循环读取若干可读段数据，直到 nAvailSampsPerChan>nReadSampsPerChan 时可调用读数函数。如果在开始采集任务后，设备迟迟不能被触发采样，可以根据需要调用 DO_SendSoftTrig()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。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pDOStatus 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《4.16 DO_STATUS（DO 工作状态信息结构）》。

返回值：如果成功获取 DO 状态，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	DO_InitTask()	DO_StartTask()
DO_SendSoftTrig()	DO_GetStatus()	DO_WaitUntilTaskDone()
DO_WriteDigitalU8()	DO_WriteDigitalOneU32()	DO_WriteDigitalOneLines()
DO_WriteDigitalOneLine()	DO_ReadbackDigitalOneU8()	DO_StopTask()

DO_ReleaseTask()　　DEV_Release()

DO_WaitUntilTaskDone()

函数原型：

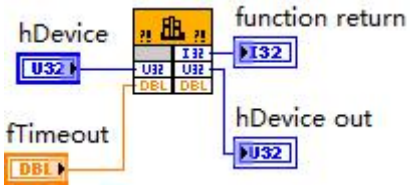
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DO_WaitUntilTaskDone (HANDLE hDevice, F64 fTimeout)
```

Visual Basic:

```
Declare Function DO_WaitUntilTaskDone Lib "USB2861" (ByVal hDevice As Long, _
                                                     ByVal fTimeout As Double) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：在 DO 的采集任务结束前等待(Wait until task done for analog input)。一旦调用 DO_StartTask() 函数后，可以调用此函数等待采集任务结束。它通常用在有限点采样模式中。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

fTimeout 入口参数，超时时间，单位：秒（S）。指定该次等待所用时间，比如设定为 10.0，即 10 秒钟的时间，如果在 10 秒内采集任务结束，则函数立即返回 TRUE，否则 10 秒钟后函数返回值 FALSE，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即总是等到采集任务结束才返回）则赋值小于 0 即可，如-1.0；如果 fTimeout=0.0，则意味着该函数只是简单查询采集任务是否结束，如果采集任务结束了，则返回 TRUE，否则返回 FALSE。

返回值：如果采集任务结束，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	DO_InitTask()	DO_StartTask()
DO_SendSoftTrig()	DO_GetStatus()	DO_WaitUntilTaskDone()
DO_WriteDigitalU8()	DO_WriteDigitalOneU32()	DO_WriteDigitalOneLines()
DO_WriteDigitalOneLine()	DO_ReadbackDigitalOneU8()	DO_StopTask()
DO_ReleaseTask()	DEV_Release()	

DO_WriteDigitalU8()

函数原型:

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

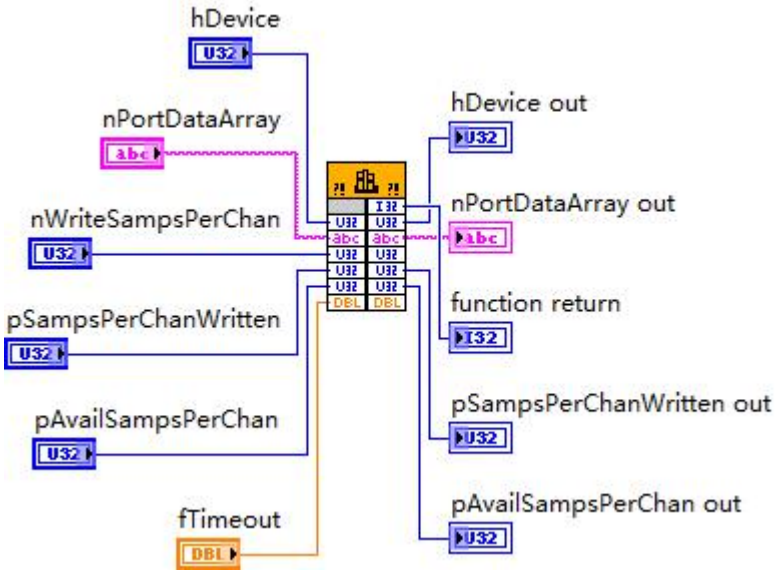
```
LONG DO_WriteDigitalU8 (HANDLE hDevice,
                        U8 nPortDataArray[],
                        U32 nWriteSampsPerChan,
                        U32* pSampsPerChanWritten,
                        U32* pAvailSampsPerChan,
```

F64 fTimeout);

Visual Basic:

```
Declare Function DO_WriteDigitalU8 Lib "USB2861" ( ByVal hDevice As Long, _
                                                   ByRef nPortDataArray As Byte, _
                                                   ByVal nWriteSampsPerChan As Long, _
                                                   ByRef pSampsPerChanWritten As Long,_
                                                   ByRef pAvailSampsPerChan As Long,_
                                                   ByVal fTimeout As Long) As Long
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取向生成任务中写入 DO 输出的二进制原码数据(Write port data to the task)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nPortDataArray 入口参数，用户缓冲区，用于存储向 Port0 端口输出的原码数据。它点空间不能小于 nWriteSampsPerChan。

nWriteSampsPerChan 入口参数，每通道请求写入的数据点数。

在有限点和连续采样模式中，它指定该次从设备的当前可写数据位置写入的数据点数（单位：点）。注意此参数的值如大于当前的可读数点 nAvailSampsPerChan 则会继续等待直到至少有 nWriteSampsPerChan 个点写入后写函数才会返回。等待期间，如果所等时间超过 fTimeout 指定时间也会返回，并置超时错误码。

在连续采样过程中，如果置波形非重生成模式，如果要保持连续不丢点，此参数应尽可能接近于甚至等于当前的可读点数(nAvailSampsPerChan),但不能大于 DOParam.nSampsPerChan。当然此参数值也不能大于 fAnlgArray 的缓冲区长度，所以为避免出错，所开辟的缓冲区不能小于 nWriteSampsPerChan*nSampChanCount(实际采样通道数)。

pSampsPerChanWritten 出口参数，返回每通道实际写入的点数。在单点采样模式中，如果写入成功，返回的每通道已写入点数总是为 1。注意每次都要检查 pSampsPerChanRead 的返回值，如果返回值等于 0，则要慎重处理。

pAvailSampsPerChan 出口参数，返回该次写操作完成时的每通道还可写而未写的数据点数。它跟 DO_GetStatus()函数取得的状态信息 DOStatus.nAvailSampsPerChan 是同一个状态信息。返回可读点数的意义在于通过数据写操作直接提供给用户，避免再次调用 DO_GetStatus()函数，即简化了

实现方法，又提高了效率，即在读数据函数返回时判断可写点数，如果大于 nWriteSampsPerChan，则可紧接着再次调用写数据函数，直到 pAvailSampsPerChan 返回值小于 nWriteSampsPerChan。在单点采样模式中,它的返回值总是为 0。

fTimeout 入口参数，超时时间，单位：秒（S）。指定等待写入额定点数的时间。比如设定为 10.0，如果在 10 秒内写入点数达到 nWriteSampsPerChan 时立即返回 TRUE，否则 10 秒钟后函数返回值 FALSE，并通过 pAvailSampsPerChan 告之实际写入的点数，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即等待请求数据点数完全写入任务中才返回）则置为负数，如-1.0 即可。如果 fTimeout=0.0，则意味着该函数仅简单判断能否立即写入请求的点数，如果不能，则不等待，立即返回 FALSE，否则返回 TRUE。

返回值： 如果函数调用成功则返回 TRUE，否则返回 FALSE，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_AVAILABLE_SAMPS	0xE0000000+1	无有效点数据
ERROR_SAMPLE_TASK_FAIL	0xE0000000+2	生成任务失败
ERROR_TIMEOUT	1460L	超时错误(见微软定义 WinError.h)

相关函数：

DEV_Create() DO_InitTask() DO_StartTask()
DO_SendSoftTrig() DO_GetStatus() DO_WaitUntilTaskDone()
DO_WriteDigitalU8() DO_WriteDigitalOneU32() DO_WriteDigitalOneLines()
DO_WriteDigitalOneLine() DO_ReadbackDigitalOneU8() DO_StopTask()
DO_ReleaseTask() DEV_Release()

DO_WriteDigitalOneU32()

函数原型:

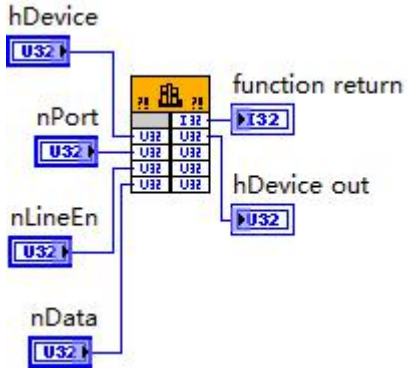
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DO_WriteDigitalOneU32 (HANDLE hDevice,
                            U32 nPort,
                            U32 nLineEn,
                            U32 nData);
```

Visual Basic:

```
Declare Function DO_WriteDigitalOneU32 Lib "USB2861" ( ByVal hDevice As Long, _
                                                       ByVal nPort As Long, _
                                                       ByVal nLineEn As Long,_
                                                       ByVal nData As Long) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：写入单点生成数据(端口数据)(Write port data to the task)

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。。

nPort 入口参数，端口号[0, 2]。

nLineEn 入口参数，线号使能, Bit[31,0]分别控制 Line[31,0]的使能, 0=禁止，1=使能。

nData 入口参数，输出的采样数据，Bit[31,0]分别确定 Line[31, 0]的数据位值。

返回值：如果函数调用成功则返回 TRUE，否则返回 FALSE，可以调用 Win32 API 函数 GetLastError()以取得进一步的错误码信息 nErrorCode。

相关函数：

DEV_Create()	DO_InitTask()	DO_StartTask()
DO_SendSoftTrig()	DO_GetStatus()	DO_WaitUntilTaskDone()
DO_WriteDigitalU8()	DO_WriteDigitalOneU32()	DO_WriteDigitalOneLines()
DO_WriteDigitalOneLine()	DO_ReadbackDigitalOneU8()	DO_StopTask()
DO_ReleaseTask()	DEV_Release()	

DO_WriteDigitalOneLines()

函数原型:

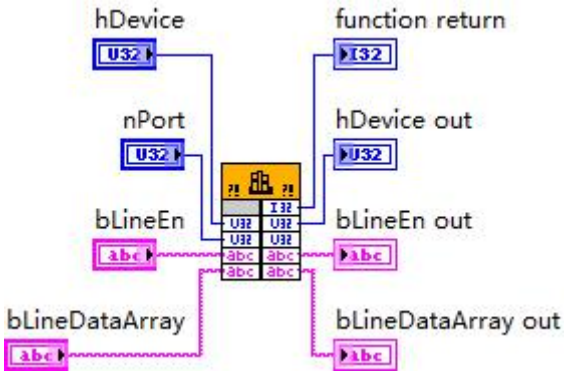
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DO_WriteDigitalOneLines(HANDLE hDevice,
                             U32 nPort,
                             U8 bLineEn[32],
                             U8 bLineDataArray[32]);
```

Visual Basic:

```
Declare Function DO_WriteDigitalOneLines Lib "USB2861" ( ByVal hDevice As Long, _
                                                         ByVal nPort As Long,_
                                                         ByRef bLineEn As Byte, _
                                                         ByRef bLineDataArray As Byte) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取单点数字量数据（Read digital data from the task）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nPort 入口参数，端口号[0, 2]。

bLineEn[32] 入口参数，线号使能, bLineEn[0]=1 选择 P0.0, bLineEn[1]=1 选择 P0.1, bLineEn[2]=1 选择 P0.2... bLineEn[3]=1 选择 P1.0。

bLineDataArray[32] 入口参数，输出的采样数据， nLineDataArray[31,0]分别确定 Line[31, 0]的数据位值

返回值：如果函数调用成功则返回 TRUE，否则返回 FALSE，可以调用 Win32 API 函数 GetLastError()以取得进一步的错误码信息 nErrorCode。

相关函数：

DEV_Create()	DO_InitTask()	DO_StartTask()
DO_SendSoftTrig()	DO_GetStatus()	DO_WaitUntilTaskDone()
DO_WriteDigitalU8()	DO_WriteDigitalOneU32()	DO_WriteDigitalOneLines()
DO_WriteDigitalOneLine()	DO_ReadbackDigitalOneU8()	DO_StopTask()
DO_ReleaseTask()	DEV_Release()	

DO_WriteDigitalOneLine()

函数原型:

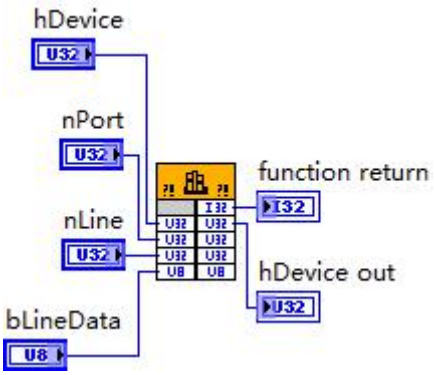
Visual C++ / C++Builder / LabWindows/CVI:

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

Visual Basic:

```
Declare Function DO_WriteDigitalOneLine Lib "USB2861" ( ByVal hDevice As Long, _
                                                        ByVal nPort As Long,_
                                                        ByVal nLine As Long, _
                                                        ByVal bLineData Byte) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取单点数字量数据（Read digital data from the task）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nPort 入口参数，端口号[0, 2]。

nLine 入口参数，线号[0, 7]。

bLineData 入口参数，输出的单线的采样数据。

返回值：如果函数调用成功则返回 TRUE，否则返回 FALSE，可以调用 Win32 API 函数

GetLastError()以取得进一步的错误码信息 nErrorCode。

相关函数：

DEV_Create()	DO_InitTask()	DO_StartTask()
DO_SendSoftTrig()	DO_GetStatus()	DO_WaitUntilTaskDone()
DO_WriteDigitalU8()	DO_WriteDigitalOneU32()	DO_WriteDigitalOneLines()
DO_WriteDigitalOneLine()	DO_ReadbackDigitalOneU8()	DO_StopTask()
DO_ReleaseTask()	DEV_Release()	

DO_ReadbackDigitalOneU8()

函数原型:

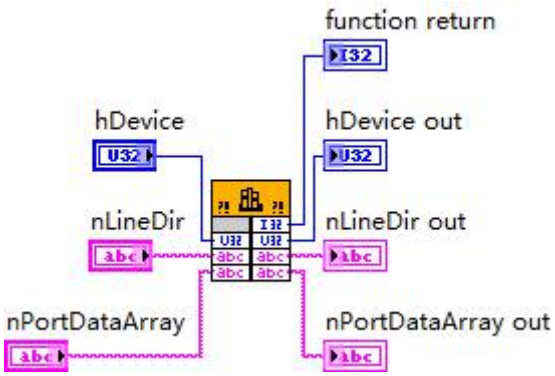
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DO_ReadbackDigitalOneU8(HANDLE hDevice,
                             U8 nLineDir[24],
                             U8 nPortDataArray[]);
```

Visual Basic:

```
Declare Function DO_ReadbackDigitalOneU8 Lib "USB2861" ( ByVal hDevice As Long, _
                                                         ByRef nLineDir As Byte, _
                                                         ByRef nPortDataArray As Byte) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取单点数字量数据（Read digital data from the task）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nLineDir[24] 出口参数，返回线方向, nLineDir[0]=1 表示 Line0 输出, =0:表示 Line0 为输入; nLineDir[1]=1 表示 Line0 输出, =0:表示 Line1 为输入。

nPortDataArray [] 入口参数，端口数据缓冲，返回回读的 DO 数据。

返回值：如果函数调用成功则返回 TRUE，否则返回 FALSE，可以调用 Win32 API 函数 GetLastError()以取得进一步的错误码信息 nErrorCode。

相关函数：

DEV_Create()	DO_InitTask()	DO_StartTask()
DO_SendSoftTrig()	DO_GetStatus()	DO_WaitUntilTaskDone()
DO_WriteDigitalU8()	DO_WriteDigitalOneU32()	DO_WriteDigitalOneLines()
DO_WriteDigitalOneLine()	DO_ReadbackDigitalOneU8()	DO_StopTask()
DO_ReleaseTask()	DEV_Release()	

DO_StopTask()

函数原型：

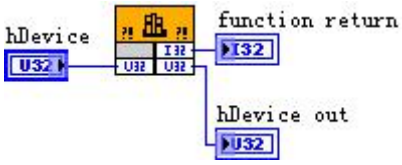
Visual C++ / C++Builder / LabWindows/CVI:

BOOL DO_StopTask (HANDLE hDevice)

Visual Basic:

Declare Function DO_StopTask Lib "USB2861" (ByVal hDevice As Long)

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：停止 DO 采样(Stop task for analog input)。必须在成功调用 DO_StartTask()函数后才能调用此函数。该函数除了停止 DO 采集外不改变设备的其他状态。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

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

相关函数：

DEV_Create()	DO_InitTask()	DO_StartTask()
DO_SendSoftTrig()	DO_GetStatus()	DO_WaitUntilTaskDone()
DO_WriteDigitalU8()	DO_WriteDigitalOneU32()	DO_WriteDigitalOneLines()
DO_WriteDigitalOneLine()	DO_ReadbackDigitalOneU8()	DO_StopTask()
DO_ReleaseTask()	DEV_Release()	

DO_ReleaseTask()

函数原型：

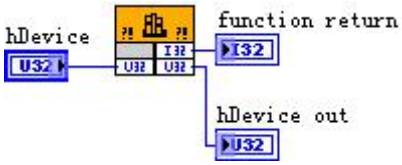
Visual C++ / C++Builder / LabWindows/CVI:

BOOL DO_ReleaseTask (HANDLE hDevice)

Visual Basic:

Declare Function DO_ReleaseTask Lib "USB2861" (ByVal hDevice As Long)

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

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

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create() DO_InitTask() DO_StartTask()
DO_SendSoftTrig() DO_GetStatus() DO_WaitUntilTaskDone()
DO_WriteDigitalU8() DO_WriteDigitalOneU32() DO_WriteDigitalOneLines()
DO_WriteDigitalOneLine() DO_ReadbackDigitalOneU8() DO_StopTask()
DO_ReleaseTask() DEV_Release()

DO_GetMainInfo()

函数原型：

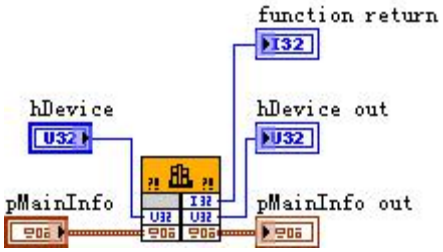
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DO_GetMainInfo (HANDLE hDevice,
                     PDO_MAIN_INFO pMainInfo)
```

Visual Basic:

```
Declare Function DO_GetMainInfo Lib "USB2861" ( ByVal hDevice As Long, _
                                                ByRef pMainInfo As DO_MAIN_INFO) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：取得 DO 功能的主要信息，如通道数、分辨率等（Get main information for analog input）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pMainInfo 出口参数，DO 主要信息结构指针，负责返回 DO 的主要描述信息。关于 DO_MAIN_INFO 的详细介绍请参考 USB2861.h 或 USB2861.bas 或 USB2861.pas 函数原型定义的头文件，也可参考本文《4.17 DO_MAIN_INFO (DO 主要信息结构体)》 关于该结构的有关说明。

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

相关函数：

DEV_Create() DO_GetMainInfo() DO_GetRateInfo() DEV_Release()

DO_GetRateInfo()

函数原型：

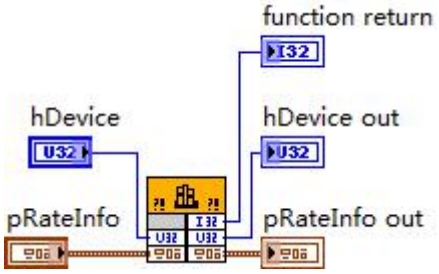
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DO_GetRateInfo(HANDLE hDevice, DO_RATE_INFO pRateInfo);
```

Visual Basic:

```
Declare Function DO_GetRateInfo Lib "USB2861" (ByVal hDevice As Long, _
                                               ByRef pRateInfo As DO_RATE_INFO) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：获得采样速率信息（Get rate information for analog input）。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pRateInfo 出口参数，DO 采样速率信息，负责返回 DO 的最大采样率，最小采样率等。详情请参考《4.18 DO_SAMP_RATE_INFO (DO 采样速率信息结构体)》。

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

DEV_Create()　　DO_GetMainInfo()　　DO_GetRateInfo()　　DEV_Release()

DO_VerifyParam()

函数原型：

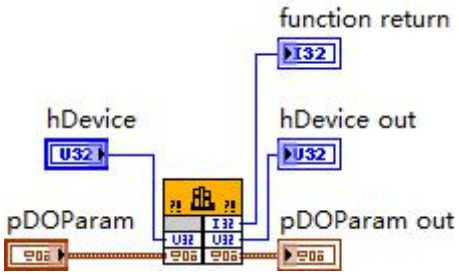
Visual C++ / C++Builder / LabWindows/CVI:

BOOL DO_VerifyParam (HANDLE hDevice, PDO_PARAM pDOParam)

Visual Basic:

Declare Function DO_VerifyParam Lib "USB2861" (ByVal hDevice As Long, _
ByRef pDOParam As DO_PARAM) As Boolean

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：校验 DO 工作参数(Verify task parameter for analog input)，这是一个辅助功能的函数，在初始化 DO 前，先校验 DO 参数的合法性。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pDOParam 入口和出口参数，DO 工作参数结构体指针，关于其具体定义及说明请参考《4.12 DO_PARAM (DO 数字量工作参数结构体)》。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 DO 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 USB2861.log 记录不合法的参数名和原因，然后返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数： DO_InitTask() DO_VerifyParam() DO_LoadParam()
DO_SaveParam() DO_ResetParam()

DO_LoadParam()

函数原型：

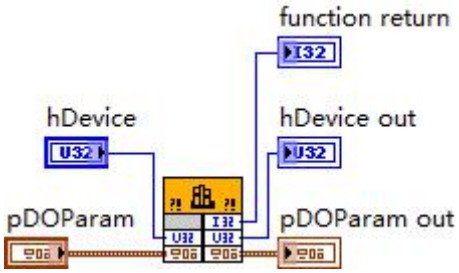
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DO_LoadParam (HANDLE hDevice,
                   PDO_PARAM pDOParam)
```

Visual Basic:

```
Declare Function DO_LoadParam Lib "USB2861" (ByVal hDevice As Long, _
                                             ByRef pDOParam As DO_PARAM) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：从 USB2861.ini 参数配置文件中读取设备的硬件参数(Load parameter from USB2861.ini for analog input)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pDOParam 出口参数，属于 PDO_PARAM 的结构指针类型，负责返回 DO 工作参数值，关于结构指针类型 PDO_PARAM 请参考 USB2861.h 或 USB2861.bas 或 USB2861.pas 函数原型定义的头文件，也可参考本文《4.12 DO_PARAM (DO 数字量工作参数结构体)》。

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

相关函数： DO_InitTask() DO_VerifyParam() DO_LoadParam()
DO_SaveParam() DO_ResetParam()

DO_SaveParam()

函数原型：

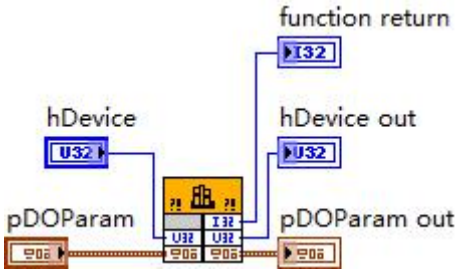
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DO_SaveParam (HANDLE hDevice,
                   PDO_PARAM pAIParam)
```

Visual Basic:

```
Declare Function DO_SaveParam Lib "USB2861" (ByVal hDevice As Long, _
                                             ByRef pAIParam As DO_PARAM) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：将用户配置好的 DO 工作参数保存在 USB2861.ini 参数配置文件中(Save parameter to USB2861.ini for analog input)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pDOParam 入口参数，DO 工作参数结构体指针，关于 DO_PARAM 的详细介绍请参考 USB2861.h 或 USB2861.bas 或 USB2861.pas 等函数原型定义的头文件，也可参考本文《4.15.3 DO_PARAM (DO 工作参数结构体)》。

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

相关函数： DO_InitTask() DO_VerifyParam() DO_LoadParam()
DO_SaveParam() DO_ResetParam()

DO_ResetParam()

函数原型：

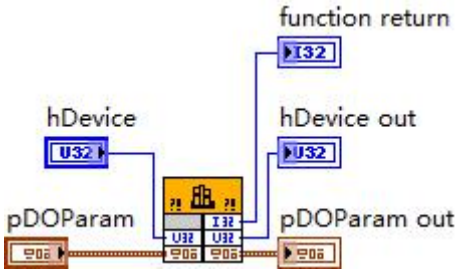
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL DO_ResetParam (HANDLE hDevice,
                    PDO_PARAM pDOParam)
```

Visual Basic:

```
Declare Function DO_ResetParam Lib "USB2861" ( ByVal hDevice As Long, _
                                               ByRef pDOParam As DO_PARAM) As Boolean
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：将 USB2861.ini 参数配置文件中原来的 DO 参数值复位至出厂时的默认值(Reset parameter to default value for analog input)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

pDOParam 出口参数，DO 工作参数结构指针，负责在参数被复位后返回其复位后的参数值。关于 DO_PARAM 的详细介绍请参考 USB2861.h 或 USB2861.bas 或 USB2861.pas 函数原型定义的头

文件，也可参考本文《4.15　DO_PARAM (DO 数字量工作参数结构体)　》。

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

相关函数： DO_InitTask()　DO_VerifyParam()　DO_LoadParam()
DO_SaveParam()　DO_ResetParam()

3.6　CTR 计数器函数原型说明

CI_CreateFreqChan()

函数原型：

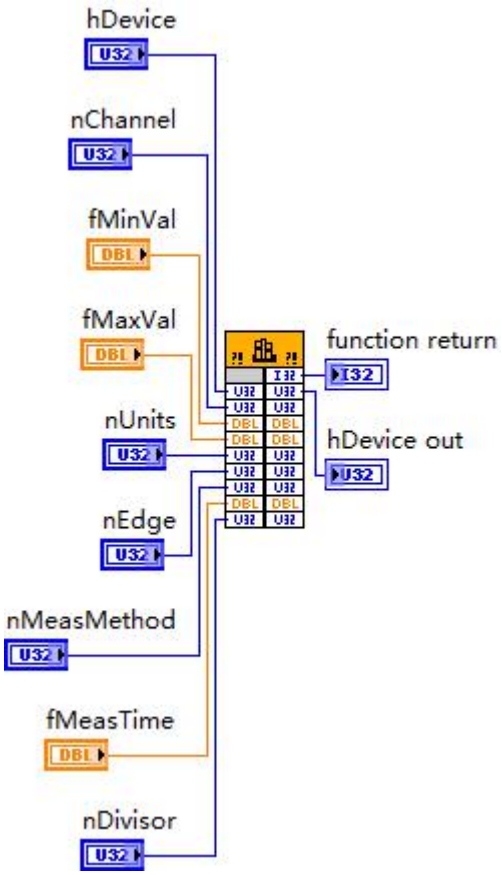
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CI_CreateFreqChan (
                        HANDLE hDevice,
                        U32 nChannel,
                        F64 fMinVal,
                        F64 fMaxVal,
                        U32 nUnits,
                        U32 nEdge,
                        U32 nMeasMethod,
                        F64 fMeasTime,
                        U32 nDivisor);
```

Visual Basic:

```
Declare Function CI_CreateFreqChan Lib "USB2861" (
                                    ByVal hDevice As Long, _
                                    ByVal nChannel As Long, _
                                    ByVal fMinVal As Double,_
                                    ByVal fMaxVal As Double,_
                                    ByVal nUnits As Long,_
                                    ByVal nEdge As Long,_
                                    ByVal nMeasMethod As Long,_
                                    ByVal fMeasTime As Double,_
                                    ByVal nDivisor As Long) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：创建频率测量通道(Create frequency measurement channel for counter input)。GATE:被测信号输入引脚。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fMinVal 入口参数，最小值是换算后测量的预期最小值。单位：默认为 Hz，另支持时钟滴答计数。

fMaxVal 入口参数，最大值是换算后测量的预期最大值。单位：默认为 Hz，另支持时钟滴答计数。

nUnits 入口参数，换算后的单位是要被使用的单位。单位：秒，时钟滴答或自定义,它不仅决定着 fMinVal 和 fMaxVal 的单位类型，同时也决定着测量结果的单位类型。

常量名	常量值	功能定义	备注
CI_FREQ_UINTS_Hz	0	赫兹	
CI_FREQ_UINTS_Ticks	1	时钟基准计数	

nEdge 入口参数，被测信号的边沿，=0:下降沿测量，=1:上升沿测量。

nMeasMethod 入口参数，测量方法,=0:低频测量, =1:高频测量, =2:宽范围测量。

fMeasTime 入口参数，测量时间,单位：秒(S),仅当测量方法为高频测量(nMeasMethod=1)时有效,取值范围[0.000001, 42.94]。

nDivisor 入口参数，分频器，仅当测量方法为宽范围测量(nMeasMethod=2)时有效,取值范围[2, 4294967295]。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CI_CreatePeriodChan()

函数原型：

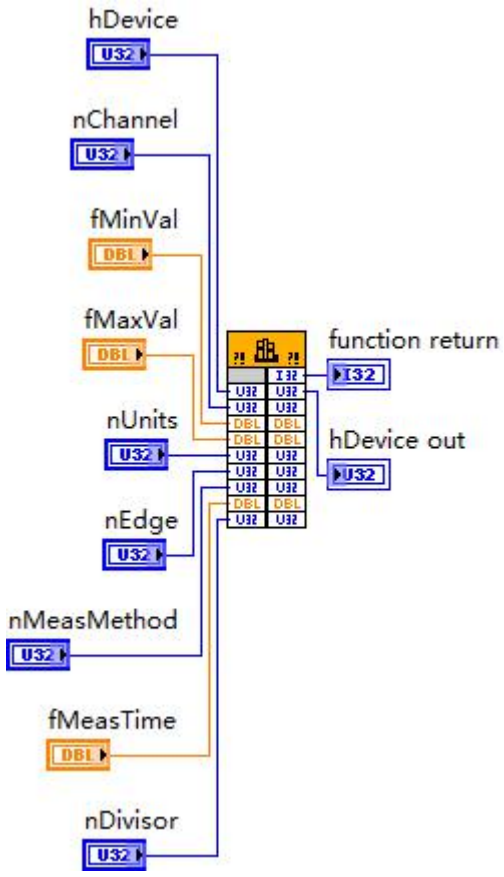
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CI_CreatePeriodChan(
                        HANDLE hDevice,
                        U32 nChannel,
                        F64 fMinVal,
                        F64 fMaxVal,
                        U32 nUnits,
                        U32 nEdge,
                        U32 nMeasMethod,
                        F64 fMeasTime,
                        U32 nDivisor);
```

Visual Basic:

```
Declare Function CI_CreatePeriodChan Lib "USB2861" (
                                        ByVal hDevice As Long, _
                                        ByVal nChannel As Long, _
                                        ByVal fMinVal As Double,_
                                        ByVal fMaxVal As Double,_
                                        ByVal nUnits As Long,_
                                        ByVal nEdge As Long,_
                                        ByVal nMeasMethod As Long,_
                                        ByVal fMeasTime As Double,_
                                        ByVal nDivisor As Long) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：创建周期测量通道(Create period measurement channel for counter input)。GATE:被测信号输入引脚

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fMinVal 入口参数，最小值是换算后测量的预期最小值。单位：默认为 Hz，另支持时钟滴答计数。

fMaxVal 入口参数，最大值是换算后测量的预期最大值。单位：默认为 Hz，另支持时钟滴答计数。

nUnits 入口参数，换算后的单位是要被使用的单位。单位：秒，时钟滴答或自定义,它不仅决定着 fMinVal 和 fMaxVal 的单位类型，同时也决定着测量结果的单位类型。

常量名	常量值	功能定义	备注
CI_PERIOD_UINTS_Seconds	0	秒	
CI_PERIOD_UINTS_Ticks	1	时钟基准计数	

nEdge 入口参数，被测信号的边沿，=0:下降沿测量，=1:上升沿测量。

nMeasMethod 入口参数，测量方法,=0:低频测量, =1:高频测量, =2:宽范围测量。

fMeasTime 入口参数，测量时间,单位：秒(S),仅当测量方法为高频测量(nMeasMethod=1)时有效,取值范围[0.000001, 42.94]。

nDivisor 入口参数，分频器，仅当测量方法为宽范围测量(nMeasMethod=2)时有效,取值范围[2,

4294967295]。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CI_CreateCountEdgesChan()

函数原型：

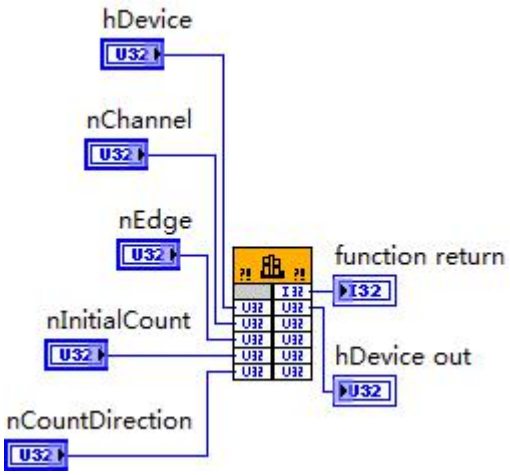
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CI_CreateCountEdgesChan(
                        HANDLE hDevice,
                        U32 nChannel,
                        U32 nEdge,
                        U32 nInitialCount,
                        U32 nCountDirection);
```

Visual Basic:

```
Declare Function CI_CreateCountEdgesChan Lib "USB2861" (
                                        ByVal hDevice As Long, _
                                        ByVal nChannel As Long, _
                                        ByVal nEdge As Long,_
                                        ByVal nInitialCountAs Long,_
                                        ByVal nCountDirection As Long) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：创建边沿计数通道。 SRC:边沿计数的源输入引脚; GATE:控制是否暂停输入引脚; AUX:控制计数方向输入引脚。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

nEdge 入口参数，被测信号的边沿，=0:下降沿测量，=1:上升沿测量。

nInitialCount 入口参数，初始计数值, 取值范围为[0, 2^32-1]。

nCountDirection 入口参数，计数方向, =0:减计数，=1:加计数, =2:外部控制

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CI_CreateTwoEdgeSepChan()

函数原型：

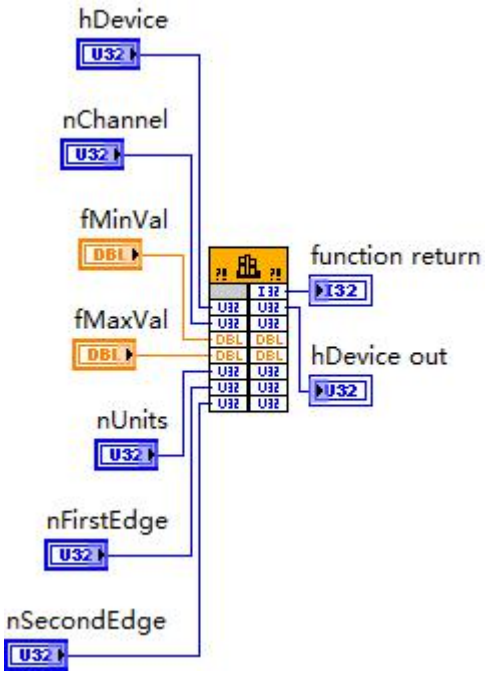
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CI_CreateTwoEdgeSepChan(
                        HANDLE hDevice,
                        U32 nChannel,
                        F64 fMinVal,
                        F64 fMaxVal,
                        U32 nUnits,
                        U32 nFirstEdge,
                        U32 nSecondEdge);
```

Visual Basic:

```
Declare Function CI_CreateTwoEdgeSepChan Lib "USB2861" (
                                        ByVal hDevice As Long, _
                                        ByVal nChannel As Long, _
                                        ByVal fMinVal As Double,_
                                        ByVal fMaxVal As Double,_
                                        ByVal nUnits As Long,_
                                        ByVal nFirstEdge As Long,_
                                        ByVal nSecondEdge As Long) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：创建两边沿间隔测量通道(Two Edge Separation)，获得两个边沿之间的时间间隔, AUX:第一个有效边沿输入引脚; GATE:第二个有效边沿输入引脚。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fMinVal 入口参数，最小值是换算后测量的预期最小值。单位：默认为 Hz, 另支持时钟滴答计数。

fMaxVal 入口参数，最大值是换算后测量的预期最大值。单位：默认为 Hz, 另支持时钟滴答计数。

nUnits 入口参数，换算后的单位是要被使用的单位。单位：秒，时钟滴答或自定义,它不仅决定着 fMinVal 和 fMaxVal 的单位类型，同时也决定着测量结果的单位类型。

nFirstEdge 入口参数，测量时的第一个信号(即启动信号)的边沿, =0:下降沿， =1:上升沿

nSecondEdge 入口参数，测量时的第二个信号(即停止信号)的边沿, =0:下降沿， =1:上升沿

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CI_CreatePulseWidthChan()

函数原型：

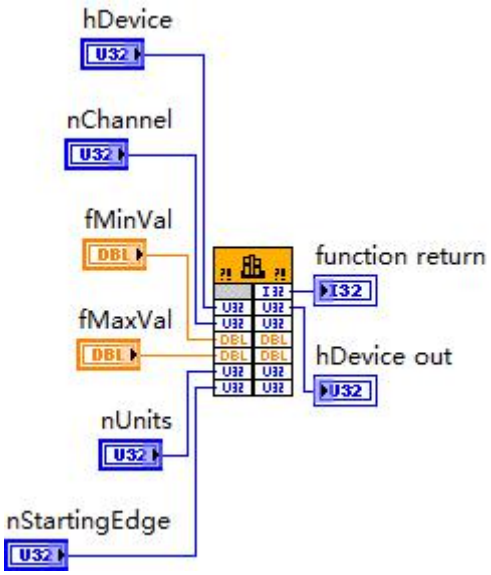
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CI_CreatePulseWidthChan(
                              HANDLE hDevice,
                              U32 nChannel,
                              F64 fMinVal,
                              F64 fMaxVal,
                              U32 nUnits,
                              U32 nStartingEdge);
```

Visual Basic:

```
Declare Function CI_CreatePulseWidthChan Lib "USB2861" (
                                                        ByVal hDevice As Long, _
                                                        ByVal nChannel As Long, _
                                                        ByVal fMinVal As Double,_
                                                        ByVal fMaxVal As Double,_
                                                        ByVal nUnits As Long,_
                                                        ByVal nStartingEdge As Long) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：创建脉冲宽度测量通道,获得脉冲的低电平或高电平的宽度(时间或时钟滴答计数), GATE:被测信号输入引脚。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fMinVal 入口参数，最小值是换算后测量的预期最小值。单位：默认为 Hz，另支持时钟滴答计数。

fMaxVal 入口参数，最大值是换算后测量的预期最大值。单位：默认为 Hz，另支持时钟滴答计数。

nUnits 入口参数，换算后的单位是要被使用的单位。单位：秒，时钟滴答或自定义,它不仅决定

着 fMinVal 和 fMaxVal 的单位类型，同时也决定着测量结果的单位类型。

nStartingEdge 入口参数，开始边沿, =0:测量脉冲的低电平时间(或滴答数), =1:测量脉冲的高电平时间(或滴答数)。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CI_CreateSemiPeriodChan()

函数原型：

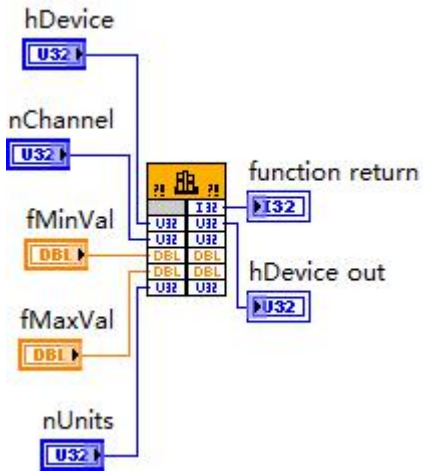
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CI_CreateSemiPeriodChan(
                        HANDLE hDevice,
                        U32 nChannel,
                        F64 fMinVal,
                        F64 fMaxVal,
                        U32 nUnits);
```

Visual Basic:

```
Declare Function CI_CreateSemiPeriodChan Lib "USB2861" (
                                        ByVal hDevice As Long, _
                                        ByVal nChannel As Long, _
                                        ByVal fMinVal As Double,_
                                        ByVal fMaxVal As Double,_
                                        ByVal nUnits As Long) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：创建半周期测量通道，半周期测量是对两个连续边沿的时间间隔进行测量，对于每个脉冲即有两个半周期存在，从上升沿开始测量。GATE:被测信号输入引脚

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fMinVal 入口参数，最小值是换算后测量的预期最小值。单位：默认为 Hz，另支持时钟滴答计数。

fMaxVal 入口参数，最大值是换算后测量的预期最大值。单位：默认为 Hz，另支持时钟滴答计数。

nUnits 入口参数，换算后的单位是要被使用的单位。单位：秒，时钟滴答或自定义,它不仅决定着 fMinVal 和 fMaxVal 的单位类型，同时也决定着测量结果的单位类型。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CI_CreatePulseChanFreq()

函数原型：

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

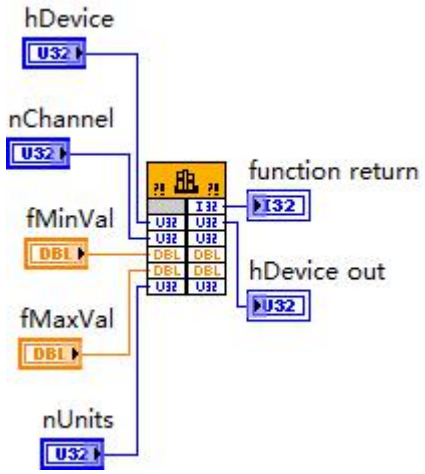
```
BOOL CI_CreatePulseChanFreq(
                        HANDLE hDevice,
                        U32 nChannel,
                        F64 fMinVal,
```

F64 fMaxVal,
U32 nUnits);

Visual Basic:

```
Declare Function CI_CreatePulseChanFreq Lib "USB2861" (
                                        ByVal hDevice As Long, _
                                        ByVal nChannel As Long, _
                                        ByVal fMinVal As Double,_
                                        ByVal fMaxVal As Double,_
                                        ByVal nUnits As Long) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：创建脉冲频率测量通道,获得脉冲的成对的频率和占空比(由半周期测量衍生,也称为半周期频率测量)。GATE:被测信号输入引脚。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fMinVal 入口参数，最小值是换算后测量的预期最小值。单位：默认为 Hz，另支持时钟滴答计数。

fMaxVal 入口参数，最大值是换算后测量的预期最大值。单位：默认为 Hz，另支持时钟滴答计数。

nUnits 入口参数，换算后的单位是要被使用的单位。单位：秒，时钟滴答或自定义,它不仅决定着 fMinVal 和 fMaxVal 的单位类型，同时也决定着测量结果的单位类型。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()

CTR_StopTask()　　CTR_ReleaseTask()　　DEV_Release()

CI_CreatePulseChanTime()

函数原型：

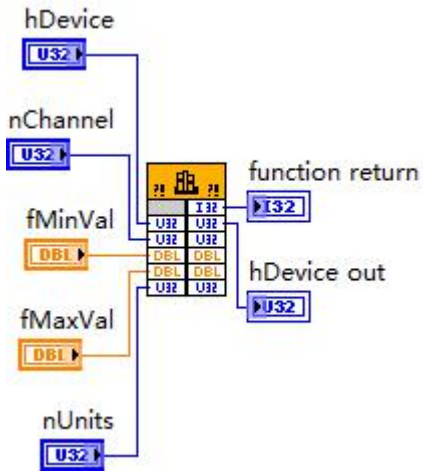
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CI_CreatePulseChanTime(
                        HANDLE hDevice,
                        U32 nChannel,
                        F64 fMinVal,
                        F64 fMaxVal,
                        U32 nUnits);
```

Visual Basic:

```
Declare Function CI_CreatePulseChanTime Lib "USB2861" (
                                        ByVal hDevice As Long, _
                                        ByVal nChannel As Long, _
                                        ByVal fMinVal As Double,_
                                        ByVal fMaxVal As Double,_
                                        ByVal nUnits As Long) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：创建脉冲时间测量通道,获得脉冲的成对的高电平时间和低电平时间(由半周期测量衍生,也称为半周期时间测量)。GATE:被测信号输入引脚

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fMinVal 入口参数，最小值是换算后测量的预期最小值。单位：默认为 Hz，另支持时钟滴答计数。

fMaxVal 入口参数，最大值是换算后测量的预期最大值。单位：默认为 Hz，另支持时钟滴答计数。

nUnits 入口参数，换算后的单位是要被使用的单位。单位：秒，时钟滴答或自定义,它不仅决定着 fMinVal 和 fMaxVal 的单位类型，同时也决定着测量结果的单位类型。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CI_CreatePulseChanTicks()

函数原型：

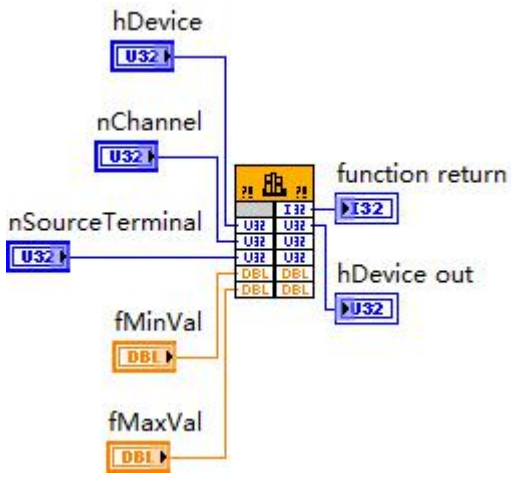
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CI_CreatePulseChanTicks(
                        HANDLE hDevice,
                        U32 nChannel,
                        F64 fMinVal,
                        F64 fMaxVal,
                        U32 nUnits);
```

Visual Basic:

```
Declare Function CI_CreatePulseChanTicks Lib "USB2861" (
                                        ByVal hDevice As Long, _
                                        ByVal nChannel As Long, _
                                        ByVal fMinVal As Double,_
                                        ByVal fMaxVal As Double,_
                                        ByVal nUnits As Long) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：创建脉冲滴答测量通道,获得脉冲的成对的高电平的时钟滴数量和低电平的时钟滴数量(由

半周期测量衍生,也称为半周期时钟滴答测量)。GATE:被测信号输入引脚

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fMinVal 入口参数，最小值是换算后测量的预期最小值。单位：默认为 Hz, 另支持时钟滴答计数。

fMaxVal 入口参数，最大值是换算后测量的预期最大值。单位：默认为 Hz, 另支持时钟滴答计数。

nUnits 入口参数，换算后的单位是要被使用的单位。单位：秒，时钟滴答或自定义,它不仅决定着 fMinVal 和 fMaxVal 的单位类型，同时也决定着测量结果的单位类型。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CI_CreateLinEncoderChan()

函数原型：

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

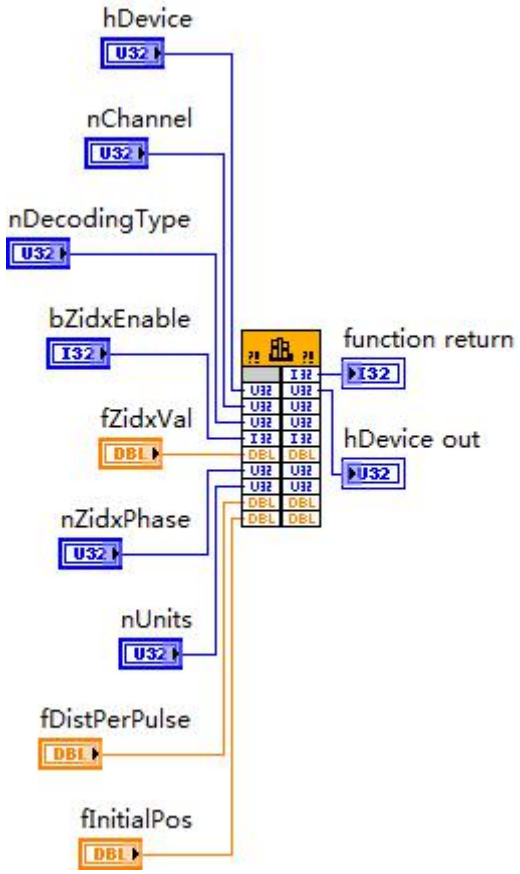
```
BOOL CI_CreateLinEncoderChan(
                        HANDLE hDevice,
                        U32 nChannel,
                        U32 nDecodingType,
                        BOOL bZidxEnable,
                        F64 fZidxVal,
                        U32 nZidxPhase,
                        U32 nUnits,
                        F64 fDistPerPulse,
                        F64 fInitialPos);
```

Visual Basic:

```
Declare Function CI_CreateLinEncoderChan Lib "USB2861" (
                                        ByVal hDevice As Long, _
                                        ByVal nChannel As Long, _
                                        ByVal nDecodingType As Long,_
                                        ByVal bZidxEnableAs Boolean,_
                                        ByVal fZidxValAs Double,_
                                        ByVal nZidxPhase As Long,_
```

ByVal nUnitsAs Long,_
ByVal fDistPerPulse As Double,_
ByVal fInitialPos As Double) As Boolean

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：创建线性编码器测量通道。A:编码器通道 A 输入引脚; B:编码器通道 B 输入引脚; Z:编码器通道 Z 输入引脚。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

nDecodingType 入口参数，解码类型, 0=X1, 1=X2, 2=X4, 3=双脉冲, 4=单脉冲

常量名	常量值	功能定义	备注
CI_DECODING_TYPE_X1	0	X1	
CI_DECODING_TYPE_X2	1	X2	
CI_DECODING_TYPE_X4	2	X4	
CI_DECODING_TYPE_TWO_PULSE	3	双脉冲解码	
CI_DECODING_TYPE_SINGLE_PULSE	4	单脉冲解码	

bZidxEnable 入口参数，启用 Z 索引指定了是否为测量启用 z 索引。=TRUE:启动, =FALSE:禁用。

fZidxVal 入口参数，Z 索引值, Z 索引值指定当信号 Z 为高且信号 A 和信号 B 处于 Z 索引相位中指定的状态时测量的值，使用单位(nUnits)中指定的单位。

nZidxPhase 入口参数，Z 索引相位，具体定义如下：

常量名	常量值	功能定义	备注
CI_ZIDX_PHASE_AHighBHigh	0	A 高 B 高	
CI_ZIDX_PHASE_AHighBLow	1	A 高 B 低	
CI_ZIDX_PHASE_ALowBHigh	2	A 低 B 高	
CI_ZIDX_PHASE_ALowBLow	3	A 低 B 低	

nUnits 入口参数，换算后的单位指定用于测量的单位

常量名	常量值	功能定义	备注
CI_LIN_ENDCODER_UINTS_Meters	0	米	
CI_LIN_ENDCODER_UINTS_Inches	1	英寸	
CI_LIN_ENDCODER_UINTS_Ticks	2	时钟滴答	

fDistPerPulse 入口参数，脉冲间隔是编码器生成的每个脉冲之间的距离。该值使用由单位(nUnits)所指定的单位。

fInitialPos 入口参数，初始位置是测量开始时编码器的位置。该值的单位由单位(nUnits)指定。

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

相关函数：

DEV_Create()　CI_CreateFreqChan()　CI_CreatePeriodChan()
CI_CreateCountEdgesChan()　CI_CreateTwoEdgeSepChan()　CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()　CI_CreatePulseChanFreq()　CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()　CI_ReadCounterU32()　CI_ReadCounterF64()
CTR_CfgSampClkTiming()　CTR_CfgImplicitTiming()　CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()　CTR_CfgDigEdgeStartTrig()　CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()　CTR_SendSoftTrig()　CTR_StartTask()
CTR_StopTask()　CTR_ReleaseTask()　DEV_Release()

CI_CreateAngEncoderChan()

函数原型：

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

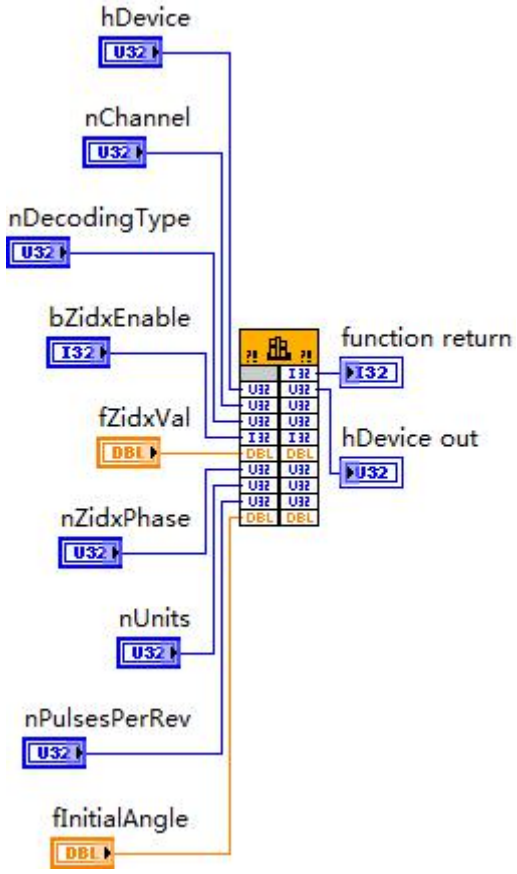
```
BOOL CI_CreateAngEncoderChan (
                HANDLE hDevice,
                U32 nChannel,
                U32 nDecodingType,
                BOOL bZidxEnable,
                F64 fZidxVal,
                U32 nZidxPhase,
                U32 nUnits,
```

```
                                U32 nPulsesPerRev,
                                F64 fInitialAngle);
```

Visual Basic:

```
Declare Function CI_CreateAngEncoderChan Lib "USB2861" (
                                                ByVal hDevice As Long, _
                                                ByVal nChannel As Long, _
                                                ByVal nDecodingType As Long,_
                                                ByVal bZidxEnableAs Boolean,_
                                                ByVal fZidxValAs Double,_
                                                ByVal nZidxPhase As Long,_
                                                ByVal nUnitsAs Long,_
                                                ByVal nPulsesPerRev As Long,_
                                                ByVal fInitialAngle As Double) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：**创建角度编码器测量通道。A:编码器通道 A 输入引脚; B:编码器通道 B 输入引脚; Z:编码器通道 Z 输入引脚。**

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

nDecodingType 入口参数，解码类型, 0=X1, 1=X2, 2=X4, 3=双脉冲, 4=单脉冲

常量名	常量值	功能定义	备注
CI_DECODING_TYPE_X1	0	X1	
CI_DECODING_TYPE_X2	1	X2	
CI_DECODING_TYPE_X4	2	X4	
CI_DECODING_TYPE_TWO_PULSE	3	双脉冲解码	
CI_DECODING_TYPE_SINGLE_PULSE	4	单脉冲解码	

bZidxEnable 入口参数，启用 Z 索引指定了是否为测量启用 z 索引。=TRUE:启动, =FALSE:禁用。

fZidxVal 入口参数，Z 索引值, Z 索引值指定当信号 Z 为高且信号 A 和信号 B 处于 Z 索引相位中指定的状态时测量的值，使用单位(nUnits)中指定的单位。

nZidxPhase 入口参数，Z 索引相位，具体定义如下：

常量名	常量值	功能定义	备注
CI_ZIDX_PHASE_AHighBHigh	0	A 高 B 高	
CI_ZIDX_PHASE_AHighBLow	1	A 高 B 低	
CI_ZIDX_PHASE_ALowBHigh	2	A 低 B 高	
CI_ZIDX_PHASE_ALowBLow	3	A 低 B 低	

nUnits 入口参数，换算后的单位指定用于测量的单位

常量名	常量值	功能定义	备注
CI_LIN_ENDCODER_UINTS_Meters	0	米	
CI_LIN_ENDCODER_UINTS_Inches	1	英寸	
CI_LIN_ENDCODER_UINTS_Ticks	2	时钟滴答	

nPulsesPerRev 入口参数，编码器每转一圈所生成脉冲的数量。该值是 A 信号或 B 信号上的脉冲数量，而不是 A 信号与 B 信号上总的脉冲数量。

fInitialAngle 入口参数，初始角是编码器在开始时的角度。该值的单位由单位(nUnits)指定。

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

相关函数：

DEV_Create()
CI_CreateFreqChan()
CI_CreatePeriodChan()
CI_CreateCountEdgesChan()
CI_CreateTwoEdgeSepChan()
CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()
CI_CreatePulseChanFreq()
CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()
CI_ReadCounterScalarU32()
CI_ReadCounterScalarF64()
CI_ReadCounterU32()
CI_ReadCounterF64()
CTR_CfgSampClkTiming()
CTR_CfgImplicitTiming()
CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()
CTR_CfgDigEdgeStartTrig()
CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()
CTR_SendSoftTrig()
CTR_StartTask()
CTR_StopTask()
CTR_ReleaseTask()
DEV_Release()

CI_ReadCounterU32()

函数原型：

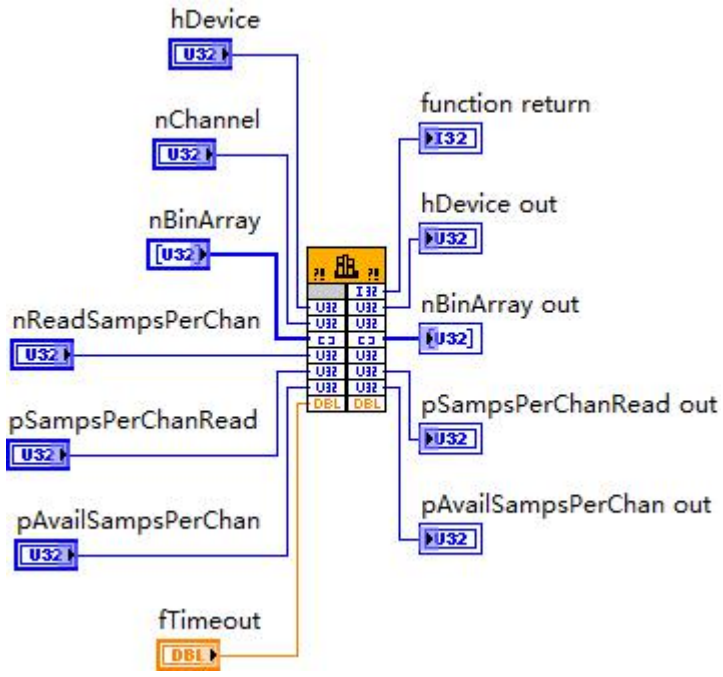
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CI_ReadCounterU32(HANDLE hDevice,
                       U32 nChannel,
                       U32 nBinArray[],
                       U32 nReadSampsPerChan,
                       U32* pSampsPerChanRead,
                       U32* pAvailSampsPerChan,
                       F64 fTimeout);
```

Visual Basic:

```
Declare Function CI_ReadCounterU32 Lib "USB2861" ( ByVal hDevice As Long, _
                                                   ByVal nChannel As Long, _
                                                   ByRef nBinArray As Long,_
                                                   ByVal nReadSampsPerChan As Long,_
                                                   ByRef pSampsPerChanRead As Long,_
                                                   ByRef pAvailSampsPerChan As Long,_
                                                   ByVal fTimeout As Double) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取缓冲测量结果,主要应用于有限点和连续等缓冲采样中(Read the measure value from buffer for counter input)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

nBinArray 出口参数，用户缓冲区，用于接收当前计数器通道的测量结果（原始数据），数据类型为 32 位无符号整型数据，值范围为[0, 2^32-1]。

nReadSampsPerChan 入口参数，每通道请求读入的数据点数。

在有限点和连续采样模式中，它指定该次从设备的当前可读数据位置读取的数据点数（单位：点）。注意此参数的值如大于缓冲区中当前的可读数点则会继续等待直到至少有 nReadSampsPerChan 个点可读后读函数才会返回。等待期间，如果所等时间超过 fTimeout 指定时间也会返回，并置超时错误码。

在连续采样过程中，如果要保持连续不丢点，此参数应尽可能地加大此参数的取值，以接近于甚至等于任务中当前的可读点数(nAvailSampsPerChan)。当然此参数值也不能大于 **nBinArray** 的缓冲区长度，所以为避免出错，所开辟的缓冲区不能小于 nReadSampsPerChan。

pSampsPerChanRead 出口参数，返回每通道实际读取的点数。在单点采样模式中，如果读取成功，返回的每通道已读取点数总是为 1。注意每次都要检查 pSampsPerChanRead 的返回值，如果返回值等于 0，则要慎重处理。

pAvailSampsPerChan 出口参数，返回该次读操作完成时的每通道还可读而未读的数据点数。返回可读点数的意义在于通过数据读操作函数直接提供给用户，即在读数据函数返回时判断可读点数，若大于 nReadSampsPerChan，则可紧接着再次调用读数据函数，直到 pAvailSampsPerChan 返回值小于 nReadSampsPerChan。在单点采样模式中,它的返回值总是为 0。

fTimeout 入口参数，超时时间，单位：秒（S）。指定等待写入额定点数的时间。比如设定为 10.0，如果在 10 秒内读取的点数达到 nReadSampsPerChan 时立即返回 TRUE，否则 10 秒钟后函数返回 FALSE，并通过 pAvailSampsPerChan 告之实际读入的点数，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即等待请求数据点数完全从任务中读到才返回）则置为负数，如-1.0 即可。如果 fTimeout=0.0，则意味着该函数仅简单判断能否立即读取到请求的点数，如果不能，则不等待，立即返回 FALSE，否则返回 TRUE。

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

相关函数： DEV_Create() CI_CreateFreqChan() CTR_StartTask()
CI_ReadCounterU32() CI_ReadCounterF64() CTR_StopTask()
CTR_ReleaseTask() DEV_Release()

CI_ReadCounterF64()

函数原型：

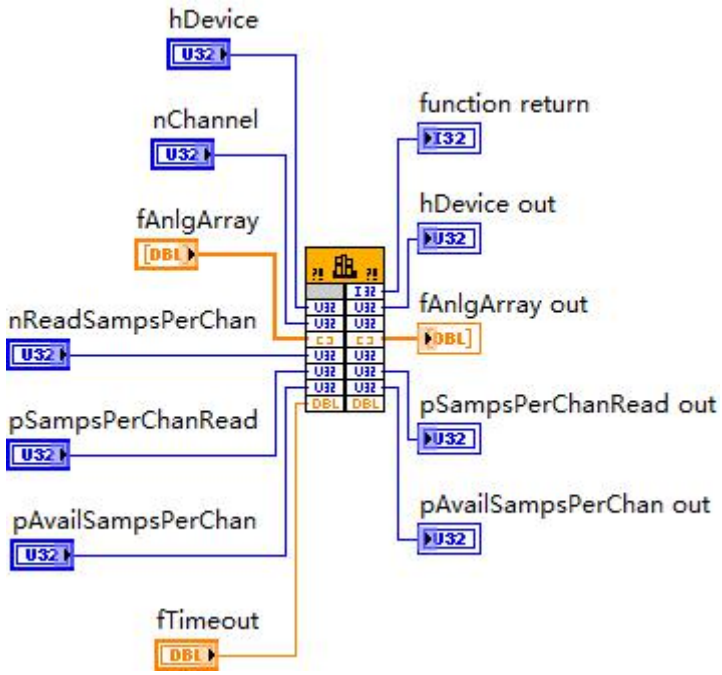
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CI_ReadCounterF64(HANDLE hDevice,
                       U32 nChannel,
                       F64 fAnlgArray[],
                       U32 nReadSampsPerChan,
                       U32* pSampsPerChanRead,
                       U32* pAvailSampsPerChan,
                       F64 fTimeout);
```

Visual Basic:

```
Declare Function CI_ReadCounterF64 Lib "USB2861" ( ByVal hDevice As Long, _
                                                   ByVal nChannel As Long, _
                                                   ByRef fAnlgArray As Double,_
                                                   ByVal nReadSampsPerChan As Long,_
                                                   ByRef pSampsPerChanRead As Long,_
                                                   ByRef pAvailSampsPerChan As Long,_
                                                   ByVal fTimeout As Double) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取缓冲测量结果,主要应用于有限点和连续等缓冲采样中(Read the measure value from buffer for counter input)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fAnlgArray 出口参数，用户缓冲区，用于接收当前计数器通道的测量结果（换算数据），数据类型为 64 位双精度浮点数据。测量数据所代表的物理意义由创建的计数器测量通道类型决定，比如事先调用 CI_CreateFreqChan()函数创建的测量通道类型为频率，那么所返回的测量数据表示的是测量信号的频率结果，其单位由创建测量通道函数的参数 nUnits 确定。其他测量通道类型同理。

nReadSampsPerChan 入口参数，每通道请求读入的数据点数。

在有限点和连续采样模式中，它指定该次从设备的当前可读数据位置读取的数据点数（单位：点）。注意此参数的值如大于缓冲区中当前的可读数点则会继续等待直到至少有 nReadSampsPerChan 个点可读后读函数才会返回。等待期间，如果所等时间超过 fTimeout 指定时间也会返回，并置超时错误码。

在连续采样过程中，如果要保持连续不丢点，此参数应尽可能地加大此参数的取值，以接近于甚至等于任务中当前的可读点数(nAvailSampsPerChan)。当然此参数值也不能大于 **nBinArray** 的缓冲区长度，所以为避免出错，所开辟的缓冲区不能小于 nReadSampsPerChan。

pSampsPerChanRead 出口参数，返回每通道实际读取的点数。在单点采样模式中，如果读取成功，返回的每通道已读取点数总是为 1。注意每次都要检查 pSampsPerChanRead 的返回值，如果返回值等于 0，则要慎重处理。

pAvailSampsPerChan 出口参数，返回该次读操作完成时的每通道还可读而未读的数据点数。返回可读点数的意义在于通过数据读操作函数直接提供给用户，即在读数据函数返回时判断可读点数，若大于 nReadSampsPerChan，则可紧接着再次调用读数据函数，直到 pAvailSampsPerChan 返回值小于 nReadSampsPerChan。在单点采样模式中,它的返回值总是为 0。

fTimeout 入口参数，超时时间，单位：秒（S）。指定等待写入额定点数的时间。比如设定为 10.0，如果在 10 秒内读取的点数达到 nReadSampsPerChan 时立即返回 TRUE，否则 10 秒钟后函数

返回 FALSE，并通过 pAvailSampsPerChan 告之实际读入的点数，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即等待请求数据点数完全从任务中读到才返回）则置为负数，如-1.0 即可。如果 fTimeout=0.0，则意味着该函数仅简单判断能否立即读取到请求的点数，如果不能，则不等待，立即返回 FALSE，否则返回 TRUE。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CI_ReadCounterScalarU32()

函数原型：

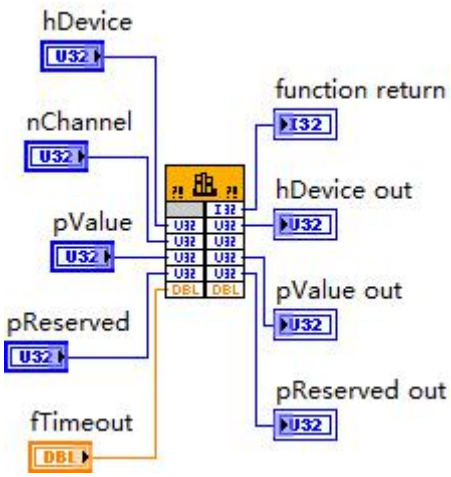
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CI_ReadCounterScalarU32(HANDLE hDevice,
                             U32 nChannel,
                             U32* pValue,
                             U32* pReserved,
                             F64 fTimeout);
```

Visual Basic:

```
Declare Function CI_ReadCounterScalarU32 Lib "USB2861" ( ByVal hDevice As Long, _
                                                         ByVal nChannel As Long, _
                                                         ByRef pValue As Long,_
                                                         ByRef pReserved As Long,_
                                                         ByVal fTimeout As Double) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取经过换算后的计数器单点测量结果(原码数据)。主要用应用于单点采样中。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

pValue 出口参数，返回经过换算后单点测量结果数据（原码数据），数据类型为 32 位无符号整型数据，值范围为[0, 2^32-1]。

pReserved 出口参数，保留(仅在半周期测量中返回低电平的时间宽度)。

fTimeout 入口参数，超时时间，单位：秒（S）。等待指定的时间。比如设定为 10.0，如果在 10 秒内读取的点数被采样到就立即返回 TRUE，否则 10 秒钟后函数返回 FALSE，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即等待请求数据点数完全从任务中读到才返回）则置为负数，如-1.0 即可。如果 fTimeout=0.0，则意味着该函数仅简单判断能否立即读取到请求的点数，如果不能，则不等待，立即返回 FALSE，否则返回 TRUE。

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

相关函数： DEV_Create() CI_CreateFreqChan() CTR_StartTask()
CI_ReadCounterU32() CI_ReadCounterF64() CTR_StopTask()
CTR_ReleaseTask() DEV_Release()

CI_ReadCounterScalarF64()

函数原型：

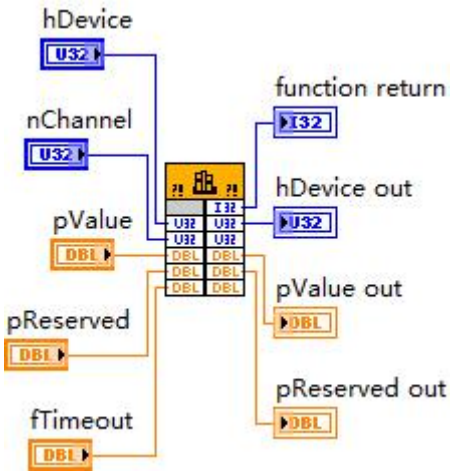
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CI_ReadCounterScalarF64(HANDLE hDevice,
                             U32 nChannel,
                             F64* pValue,
                             F64* pReserved,
                             F64 fTimeout);
```

Visual Basic:

```
Declare Function CI_ReadCounterScalarF64 Lib "USB2861" ( ByVal hDevice As Long, _
                                                         ByVal nChannel As Long, _
                                                         ByRef pValue As Double,_
                                                         ByRef pReserved As Double,_
                                                         ByVal fTimeout As Double) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取经过换算后的计数器单点测量结果(物理量)。主要用应用于单点采样中。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

pValue 出口参数，返回经过换算后单点测量结果数据（物理数据），数据类型为双精度浮点数据。

pReserved 出口参数，保留(仅在半周期测量中返回低电平的时间宽度)。

fTimeout 入口参数，超时时间，单位：秒（S）。等待指定的时间。比如设定为 10.0，如果在 10 秒内读取的点数被采样到就立即返回 TRUE，否则 10 秒钟后函数返回 FALSE，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即等待请求数据点数完全从任务中读到才返回）则置为负数，如-1.0 即可。如果 fTimeout=0.0，则意味着该函数仅简单判断能否立即读取到请求的点数，如果不能，则不等待，立即返回 FALSE，否则返回 TRUE。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CI_ReadCtrFreq()

函数原型：

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

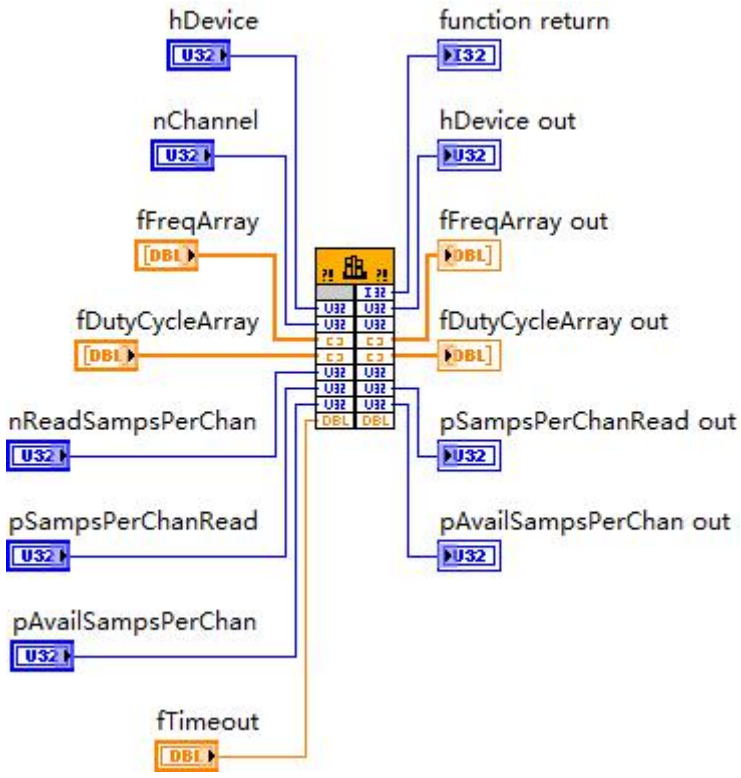
```
BOOL CI_ReadCtrFreq (HANDLE hDevice,
                     U32 nChannel,
```

```
                                F64 fFreqArray [],
                                F64 fDutyCycleArray[],
                                U32 nReadSampsPerChan,
                                U32* pSampsPerChanRead,
                                U32* pAvailSampsPerChan,
                                F64 fTimeout);
```

Visual Basic:

```
Declare Function CI_ReadCtrFreq Lib "USB2861" ( ByVal hDevice As Long, _
                                                ByVal nChannel As Long, _
                                                ByRef fFreqArray As Double,_
                                                ByRef fDutyCycleArray As Double,_
                                                ByVal nReadSampsPerChan As Long,_
                                                ByRef pSampsPerChanRead As Long,_
                                                ByRef pAvailSampsPerChan As Long,_
                                                ByVal fTimeout As Double) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取缓冲测量结果(频率和占空比),主要应用于有限点和连续等缓冲采样中,须事先调用 CI_CreatePulseChanFreq()创建脉冲测量通道。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fFreqArray 出口参数，用户缓冲区，用于接收当前计数器通道的脉冲测量的频率结果，数据类型为 64 位双精度浮点数据。事先调用 CI_CreatePulseChanFreq()函数创建的基于频率的脉冲测量通道，那么所返回的测量数据表示的是测量信号的频率结果，其单位由创建测量通道函数的参数 nUnits

确定。其他测量通道类型同理。

fDutyCycleArray 出口参数，用户缓冲区，用于接收当前计数器通道的脉冲测量的占空比结果，数据类型为 64 位双精度浮点数据。事先调用 CI_CreatePulseChanFreq()函数创建的基于频率的脉冲测量通道，那么所返回的测量数据表示的是测量信号的占空比结果。其他测量通道类型同理。

nReadSampsPerChan 入口参数，每通道请求读入的数据点数。

在有限点和连续采样模式中，它指定该次从设备的当前可读数据位置读取的数据点数（单位：点）。注意此参数的值如大于缓冲区中当前的可读数点则会继续等待直到至少有 nReadSampsPerChan 个点可读后读函数才会返回。等待期间，如果所等时间超过 fTimeout 指定时间也会返回，并置超时错误码。

在连续采样过程中，如果要保持连续不丢点，此参数应尽可能地加大此参数的取值，以接近于甚至等于任务中当前的可读点数(nAvailSampsPerChan)。当然此参数值也不能大于 **nBinArray** 的缓冲区长度，所以为避免出错，所开辟的缓冲区不能小于 nReadSampsPerChan。

pSampsPerChanRead 出口参数，返回每通道实际读取的点数。在单点采样模式中，如果读取成功，返回的每通道已读取点数总是为 1。注意每次都要检查 pSampsPerChanRead 的返回值，如果返回值等于 0，则要慎重处理。

pAvailSampsPerChan 出口参数，返回该次读操作完成时的每通道还可读而未读的数据点数。返回可读点数的意义在于通过数据读操作函数直接提供给用户，即在读数据函数返回时判断可读点数，若大于 nReadSampsPerChan，则可紧接着再次调用读数据函数，直到 pAvailSampsPerChan 返回值小于 nReadSampsPerChan。在单点采样模式中,它的返回值总是为 0。

fTimeout 入口参数，超时时间，单位：秒（S）。指定等待写入额定点数的时间。比如设定为 10.0，如果在 10 秒内读取的点数达到 nReadSampsPerChan 时立即返回 TRUE，否则 10 秒钟后函数返回 FALSE，并通过 pAvailSampsPerChan 告之实际读入的点数，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即等待请求数据点数完全从任务中读到才返回）则置为负数，如-1.0 即可。如果 fTimeout=0.0，则意味着该函数仅简单判断能否立即读取到请求的点数，如果不能，则不等待，立即返回 FALSE，否则返回 TRUE。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CI_ReadCtrTime()

函数原型：

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

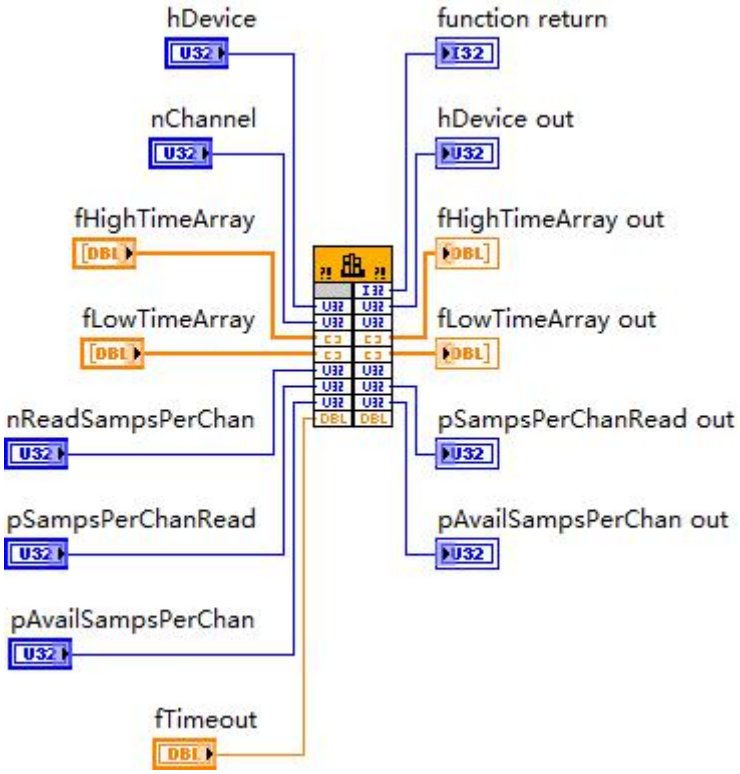
```
BOOL CI_ReadCtrTime (HANDLE hDevice,
                     U32 nChannel,
                     F64 fHighTimeArray[],
```

```
                                F64 fLowTimeArray[],
                                U32 nReadSampsPerChan,
                                U32* pSampsPerChanRead,
                                U32* pAvailSampsPerChan,
                                F64 fTimeout);
```

Visual Basic:

```
Declare Function CI_ReadCtrTime Lib "USB2861" ( ByVal hDevice As Long, _
                                                ByVal nChannel As Long, _
                                                ByRef fHighTimeArray As Double,_
                                                ByRef fLowTimeArray As Double,_
                                                ByVal nReadSampsPerChan As Long,_
                                                ByRef pSampsPerChanRead As Long,_
                                                ByRef pAvailSampsPerChan As Long,_
                                                ByVal fTimeout As Double) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取脉冲测量结果(高电平和低电平时间),主要应用于有限点和连续等缓冲采样中,须事先调用 CI_CreatePulseChanTime()创建脉冲测量通道。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fHighTimeArray 出口参数，用户缓冲区，用于接收当前计数器通道的脉冲测量的高电平时间结果，数据类型为 64 位双精度浮点数据。事先调用 CI_CreatePulseChanTime()函数创建的基于时间的脉冲测量通道，那么所返回的测量数据表示的是测量信号的高电平时间结果，其单位由创建测量通道函数的参数 nUnits 确定。其他测量通道类型同理。

fLowTimeArray 出口参数，用户缓冲区，用于接收当前计数器通道的脉冲测量的高电平时间结果，数据类型为 64 位双精度浮点数据。事先调用 CI_CreatePulseChanTime()函数创建的基于时间的脉冲测量通道，那么所返回的测量数据表示的是测量信号的低电平时间结果。其他测量通道类型同理。

nReadSampsPerChan 入口参数，每通道请求读入的数据点数。

在有限点和连续采样模式中，它指定该次从设备的当前可读数据位置读取的数据点数（单位：点）。注意此参数的值如大于缓冲区中当前的可读数点则会继续等待直到至少有 nReadSampsPerChan 个点可读后读函数才会返回。等待期间，如果所等时间超过 fTimeout 指定时间也会返回，并置超时错误码。

在连续采样过程中，如果要保持连续不丢点，此参数应尽可能地加大此参数的取值，以接近于甚至等于任务中当前的可读点数(nAvailSampsPerChan)。当然此参数值也不能大于 **nBinArray** 的缓冲区长度，所以为避免出错，所开辟的缓冲区不能小于 nReadSampsPerChan。

pSampsPerChanRead 出口参数，返回每通道实际读取的点数。在单点采样模式中，如果读取成功，返回的每通道已读取点数总是为 1。注意每次都要检查 pSampsPerChanRead 的返回值，如果返回值等于 0，则要慎重处理。

pAvailSampsPerChan 出口参数，返回该次读操作完成时的每通道还可读而未读的数据点数。返回可读点数的意义在于通过数据读操作函数直接提供给用户，即在读数据函数返回时判断可读点数，若大于 nReadSampsPerChan，则可紧接着再次调用读数据函数，直到 pAvailSampsPerChan 返回值小于 nReadSampsPerChan。在单点采样模式中,它的返回值总是为 0。

fTimeout 入口参数，超时时间，单位：秒（S）。指定等待写入额定点数的时间。比如设定为 10.0，如果在 10 秒内读取的点数达到 nReadSampsPerChan 时立即返回 TRUE，否则 10 秒钟后函数返回 FALSE，并通过 pAvailSampsPerChan 告之实际读入的点数，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即等待请求数据点数完全从任务中读到才返回）则置为负数，如-1.0 即可。如果 fTimeout=0.0，则意味着该函数仅简单判断能否立即读取到请求的点数，如果不能，则不等待，立即返回 FALSE，否则返回 TRUE。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CI_ReadCtrTicks()

函数原型：

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

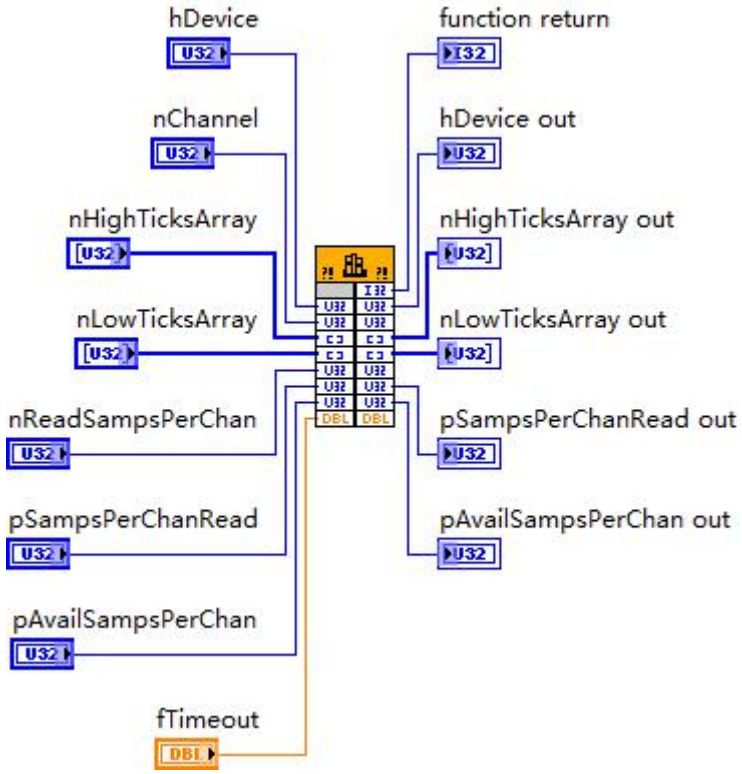
```
BOOL CI_ReadCtrTicks (HANDLE hDevice,
                      U32 nChannel,
                      F64 fHighTicksArray[],
```

```
                                F64 fLowTicksArray[],
                                U32 nReadSampsPerChan,
                                U32* pSampsPerChanRead,
                                U32* pAvailSampsPerChan,
                                F64 fTimeout);
```

Visual Basic:

```
Declare Function CI_ReadCtrTicks Lib "USB2861" ( ByVal hDevice As Long, _
                                                 ByVal nChannel As Long, _
                                                 ByRef fHighTicksArray As Double,_
                                                 ByRef fLowTicksArray As Double,_
                                                 ByVal nReadSampsPerChan As Long,_
                                                 ByRef pSampsPerChanRead As Long,_
                                                 ByRef pAvailSampsPerChan As Long,_
                                                 ByVal fTimeout As Double) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取脉冲测量结果(高电平和低电平时钟滴答),主要应用于有限点和连续等缓冲采样中,须事先调用 CI_CreatePulseChanTicks()创建脉冲测量通道。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fHighTicksArray 出口参数，用户缓冲区，用于接收当前计数器通道的脉冲测量的高电平滴答结果，数据类型为 64 位双精度浮点数据。事先调用 CI_CreatePulseChanTicks()函数创建的基于滴答的脉冲测量通道，那么所返回的测量数据表示的是测量信号的高电平滴答结果，其单位由创建测量通道函数的参数 nUnits 确定。其他测量通道类型同理。

fLowTicksArray 出口参数，用户缓冲区，用于接收当前计数器通道的脉冲测量的高电平滴答结果，数据类型为 64 位双精度浮点数据。事先调用 CI_CreatePulseChanTicks()函数创建的基于滴答的脉冲测量通道，那么所返回的测量数据表示的是测量信号的低电平滴答结果。其他测量通道类型同理。

nReadSampsPerChan 入口参数，每通道请求读入的数据点数。

在有限点和连续采样模式中，它指定该次从设备的当前可读数据位置读取的数据点数（单位：点）。注意此参数的值如大于缓冲区中当前的可读数点则会继续等待直到至少有 nReadSampsPerChan 个点可读后读函数才会返回。等待期间，如果所等时间超过 fTimeout 指定时间也会返回，并置超时错误码。

在连续采样过程中，如果要保持连续不丢点，此参数应尽可能地加大此参数的取值，以接近于甚至等于任务中当前的可读点数(nAvailSampsPerChan)。当然此参数值也不能大于 **nBinArray** 的缓冲区长度，所以为避免出错，所开辟的缓冲区不能小于 nReadSampsPerChan。

pSampsPerChanRead 出口参数，返回每通道实际读取的点数。在单点采样模式中，如果读取成功，返回的每通道已读取点数总是为 1。注意每次都要检查 pSampsPerChanRead 的返回值，如果返回值等于 0，则要慎重处理。

pAvailSampsPerChan 出口参数，返回该次读操作完成时的每通道还可读而未读的数据点数。返回可读点数的意义在于通过数据读操作函数直接提供给用户，即在读数据函数返回时判断可读点数，若大于 nReadSampsPerChan，则可紧接着再次调用读数据函数，直到 pAvailSampsPerChan 返回值小于 nReadSampsPerChan。在单点采样模式中,它的返回值总是为 0。

fTimeout 入口参数，超时时间，单位：秒（S）。指定等待写入额定点数的时间。比如设定为 10.0，如果在 10 秒内读取的点数达到 nReadSampsPerChan 时立即返回 TRUE，否则 10 秒钟后函数返回 FALSE，并通过 pAvailSampsPerChan 告之实际读入的点数，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即等待请求数据点数完全从任务中读到才返回）则置为负数，如-1.0 即可。如果 fTimeout=0.0，则意味着该函数仅简单判断能否立即读取到请求的点数，如果不能，则不等待，立即返回 FALSE，否则返回 TRUE。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CI_ReadCtrFreqScalar()

函数原型：

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

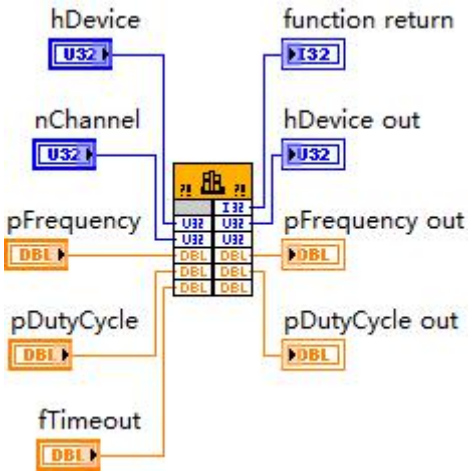
```
BOOL CI_ReadCtrFreqScalar (HANDLE hDevice,
                           U32 nChannel,
                           F64* pFrequency,
```

F64* pDutyCycle,
F64 fTimeout);

Visual Basic:

Declare Function CI_ReadCtrFreqScalar Lib "USB2861" (ByVal hDevice As Long, _
ByVal nChannel As Long, _
ByRef pFrequency As Double,_
ByRef pDutyCycle As Double,_
ByVal fTimeout As Double) As Boolean

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取经过换算后的脉冲测量结果数据(频率和占空比) ,须事先调用 CI_CreatePulseChanFreq()创建脉冲测量通道

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

pFrequency 出口参数，返回单点的频率值,单位由函数 CI_CreatePulseChanFreq()的参数 nUnits 决定

pDutyCycle 出口参数，返回单点的占空比，取值范围(0.0, 1.0),是脉冲的高电平时间除以脉冲周期的结果。

fTimeout 入口参数，超时时间，单位：秒（S）。等待指定的时间。比如设定为 10.0，如果在 10 秒内读取的点数被采样到就立即返回 TRUE，否则 10 秒钟后函数返回 FALSE，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即等待请求数据点数完全从任务中读到才返回）则置为负数，如-1.0 即可。如果 fTimeout=0.0，则意味着该函数仅简单判断能否立即读取到请求的点数，如果不能，则不等待，立即返回 FALSE，否则返回 TRUE。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()

CTR_CfgSampClkTiming() CTR_CfgImplicitTiming() CTR_WaitUntilTaskDone()
CTR_DisableStartTrig() CTR_CfgDigEdgeStartTrig() CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig() CTR_SendSoftTrig() CTR_StartTask()
CTR_StopTask() CTR_ReleaseTask() DEV_Release()

CI_ReadCtrTimeScalar()

函数原型：

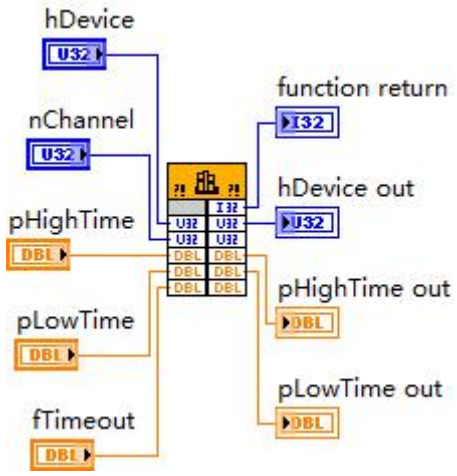
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CI_ReadCtrTimeScalar (HANDLE hDevice,
                           U32 nChannel,
                           F64* pHighTime,
                           F64* pLowTime,
                           F64 fTimeout);
```

Visual Basic:

```
Declare Function CI_ReadCtrTimeScalar Lib "USB2861" ( ByVal hDevice As Long, _
                                                      ByVal nChannel As Long, _
                                                      ByRef pHighTime As Double,_
                                                      ByRef pLowTime As Double,_
                                                      ByVal fTimeout As Double) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取经过换算后的脉冲测量结果数据(高电平和低电平时间) ,须事先调用 CI_CreatePulseChanTime()创建脉冲测量通道

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

pHighTime 出口参数，返回单点的高电平时间,单位由函数 CI_CreatePulseChanTime()的参数 nUnits 决定

pLowTime 出口参数，返回单点的低电平时间,单位由函数 CI_CreatePulseChanTime()的参数 nUnits 决定。

fTimeout 入口参数，超时时间，单位：秒（S）。等待指定的时间。比如设定为 10.0，如果在

10 秒内读取的点数被采样到就立即返回 TRUE，否则 10 秒钟后函数返回 FALSE，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即等待请求数据点数完全从任务中读到才返回）则置为负数，如-1.0 即可。如果 fTimeout=0.0，则意味着该函数仅简单判断能否立即读取到请求的点数，如果不能，则不等待，立即返回 FALSE，否则返回 TRUE。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CI_ReadCtrTicksScalar ()

函数原型：

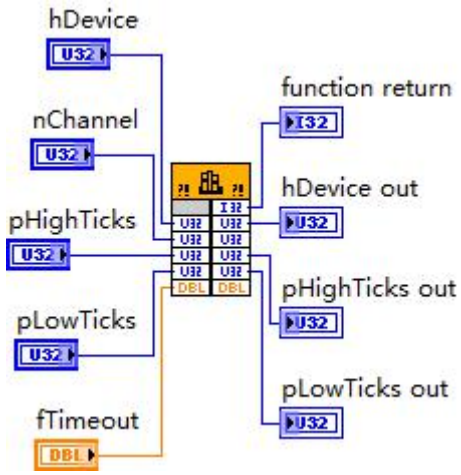
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CI_ReadCtrTicksScalar (HANDLE hDevice,
                            U32 nChannel,
                            U32* pHighTicks,
                            U32* pLowTicks,
                            F64 fTimeout);
```

Visual Basic:

```
Declare Function CI_ReadCtrTicksScalar Lib "USB2861" ( ByVal hDevice As Long, _
                                                       ByVal nChannel As Long, _
                                                       ByRef pHighTicks As Long,_
                                                       ByRef  pLowTicks As Long,_
                                                       ByVal fTimeout As Double) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：读取经过换算后的脉冲测量结果数据(高电平和低电平时钟滴答) ,须事先调用 CI_CreatePulseChanTicks()创建脉冲测量通道

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

pHighTicks 出口参数，返回单点的高电平时间,单位由函数 CI_CreatePulseChanTime()的参数 nUnits 决定

pLowTicks 出口参数，返回单点的低电平时间,单位由函数 CI_CreatePulseChanTime()的参数 nUnits 决定。

fTimeout 入口参数，超时时间，单位：秒（S）。等待指定的时间。比如设定为 10.0，如果在 10 秒内读取的点数被采样到就立即返回 TRUE，否则 10 秒钟后函数返回 FALSE，如果采样速率极慢或触发事件长时间都不能达到的情况下，建议该超时时间应足够长；如果想禁止超时返回（即等待请求数据点数完全从任务中读到才返回）则置为负数，如-1.0 即可。如果 fTimeout=0.0，则意味着该函数仅简单判断能否立即读取到请求的点数，如果不能，则不等待，立即返回 FALSE，否则返回 TRUE。

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

相关函数：

DEV_Create()	CI_CreateFreqChan()	CI_CreatePeriodChan()
CI_CreateCountEdgesChan()	CI_CreateTwoEdgeSepChan()	CI_CreatePulseWidthChan()
CI_CreateSemiPeriodChan()	CI_CreatePulseChanFreq()	CI_CreateLinEncoderChan()
CI_CreateAngEncoderChan ()	CI_ReadCounterU32()	CI_ReadCounterF64()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CO_CreatePulseChanFreq()

函数原型：

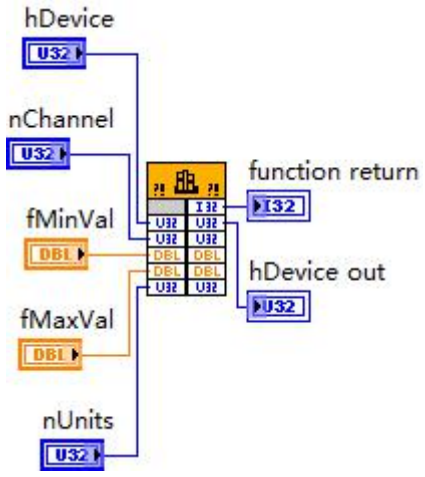
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CO_CreatePulseChanFreq (
                        HANDLE hDevice,
                        U32 nChannel,
                        U32 nUnits,
                        U32 nIdleState,
                        F64 fInitialDelay,
                        F64 fFreq,
                        F64 fDutyCycle);
```

Visual Basic:

```
Declare Function CO_CreatePulseChanFreq Lib "USB2861" (
                                        ByVal hDevice As Long, _
                                        ByVal nChannel As Long, _
                                        ByVal nUnits As Long,_
                                        ByVal nIdleState As Long,_
                                        ByVal fInitialDelay As Double,_
                                        ByVal fFreq As Double,_
                                        ByVal fDutyCycle As Double) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：创建脉冲频率生成通道。OUT:脉冲输出引脚,创建通道后若再调用 CO_WriteFreq()则为可变脉冲输出,否则为固定脉冲输出

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

nUnits 入口参数，指定参数 fFreq 的单位，默认为 Hz。

常量名	常量值	功能定义	备注
CO_FREQ_UINTS_Hz	0	赫兹	
CO_FREQ_UINTS_KHz	1	千赫兹	
CO_FREQ_UINTS_MHz	2	兆赫兹	

nIdleState 入口参数，空闲时的状态，即在 StartTask()前和 StopTask()后的状态，=0:低电平, =1:

高电平。

fInitialDelay 入口参数，初始延迟时间，即在 StartTask()开始往后到输出脉冲之间的时间值(单位：秒)。

fFreq 入口参数，生成脉冲的频率，其单位由 nUnits 参数指定。

fDutyCycle 入口参数，生成脉冲的占空比。

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

相关函数：

DEV_Create() CO_CreatePulseChanFreq() CO_CreatePulseChanTime()
CO_CreatePulseChanTicks() CO_WriteFreq() CO_WriteTime()
CO_WriteTicks() CO_SetWriteRegenMode()
CO_SetEnableInitialDelayOnRetrigger() CTR_SetStartTrigRetriggerable()
CTR_CfgSampClkTiming() CTR_CfgImplicitTiming() CTR_WaitUntilTaskDone()
CTR_DisableStartTrig() CTR_CfgDigEdgeStartTrig() CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig() CTR_SendSoftTrig() CTR_StartTask()
CTR_StopTask() CTR_ReleaseTask() DEV_Release()

CO_CreatePulseChanTime()

函数原型：

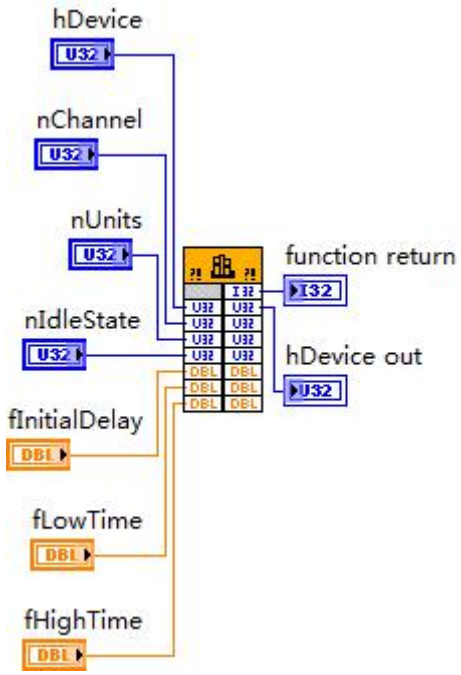
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CO_CreatePulseChanTime (
                    HANDLE hDevice,
                    U32 nChannel,
                    U32 nUnits,
                    U32 nIdleState,
                    F64 fInitialDelay,
                    F64 fLowTime,
                    F64 fHighTime);
```

Visual Basic:

```
Declare Function CO_CreatePulseChanTime Lib "USB2861" (
                                ByVal hDevice As Long, _
                                ByVal nChannel As Long, _
                                ByVal nUnits As Long,_
                                ByVal nIdleState As Long,_
                                ByVal fInitialDelay As Double,_
                                ByVal fLowTime Double,_
                                ByVal fHighTimeAs Double) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：创建脉冲时间生成通道。OUT:脉冲输出引脚,创建通道后若再调用 CO_WriteTime()则为可变脉冲输出,否则为固定脉冲输出

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

nUnits 入口参数，指定参数 fFreq 的单位，默认为 Hz。

常量名	常量值	功能定义	备注
CO_TIME_UINTS_Seconds	0	秒	
CO_TIME_UINTS_Milliseconds	1	毫秒	
CO_TIME_UINTS_Microseconds	2	微秒	

nIdleState 入口参数，空闲时的状态，即在 StartTask()前和 StopTask()后的状态，=0:低电平, =1:高电平。

fInitialDelay 入口参数，初始延迟时间，即在 StartTask()开始往后到输出脉冲之间的时间值(单位：秒)。

fLowTime 入口参数，脉冲低电平的时间,时间单位由 nUnits 指定。

fHighTime 入口参数，脉冲高电平的时间,时间单位由 nUnits 指定。

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

相关函数：

DEV_Create()　CO_CreatePulseChanFreq()　CO_CreatePulseChanTime()

CO_CreatePulseChanTicks()　CO_WriteFreq()　CO_WriteTime()

CO_WriteTicks()　CO_SetWriteRegenMode()　CTR_CfgSampClkTiming()

CTR_CfgImplicitTiming()　CTR_WaitUntilTaskDone()　CTR_DisableStartTrig()

CTR_CfgDigEdgeStartTrig()　CTR_DisablePauseTrig()　CTR_CfgDigEdgePauseTrig()

CTR_SendSoftTrig()　CTR_StartTask()　CTR_StopTask()

CTR_ReleaseTask()　　DEV_Release()

CO_CreatePulseChanTicks()

函数原型：

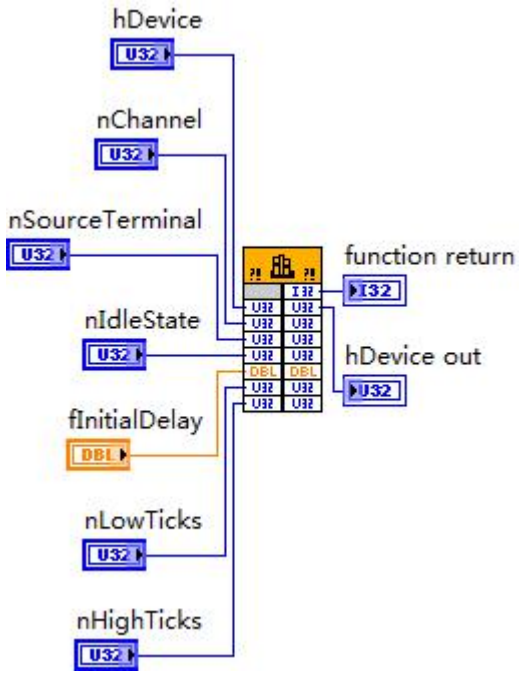
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CO_CreatePulseChanTicks(
                        HANDLE hDevice,
                        U32 nChannel,
                        U32 nUnits,
                        U32 nIdleState,
                        F64 fInitialDelay,
                        U32 nLowTicks,
                        U32 nHighTicks);
```

Visual Basic:

```
Declare Function CO_CreatePulseChanTicks Lib "USB2861" (
                                        ByVal hDevice As Long, _
                                        ByVal nChannel As Long, _
                                        ByVal nUnits As Long,_
                                        ByVal nIdleState As Long,_
                                        ByVal fInitialDelay As Double,_
                                        ByVal nLowTicks As Long,_
                                        ByVal nHighTicks As Long) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：创建脉冲滴答生成通道。OUT:脉冲输出引脚,,创建通道后若再调用 CO_WriteTicks()则为可变脉冲输出,否则为固定脉冲输出。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

nUnits 入口参数，指定参数 fFreq 的单位，默认为 Hz。

常量名	常量值	功能定义	备注
CO_TIME_UINTS_Seconds	0	秒	
CO_TIME_UINTS_Milliseconds	1	毫秒	
CO_TIME_UINTS_Microseconds	2	微秒	

nIdleState 入口参数，空闲时的状态，即在 StartTask()前和 StopTask()后的状态，=0:低电平, =1:高电平。

fInitialDelay 入口参数，初始延迟时间，即在 StartTask()开始往后到输出脉冲之间的时间值(单位：秒)。

fLowTime 入口参数，脉冲低电平的时间,时间单位由 nUnits 指定。

fHighTime 入口参数，脉冲高电平的时间,时间单位由 nUnits 指定。

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

相关函数：

DEV_Create() CO_CreatePulseChanFreq() CO_CreatePulseChanTime()
CO_CreatePulseChanTicks() CO_WriteFreq() CO_WriteTime()
CO_WriteTicks() CO_SetWriteRegenMode()
CO_SetEnableInitialDelayOnRetrigger() CTR_SetStartTrigRetriggerable()
CTR_CfgSampClkTiming() CTR_CfgImplicitTiming() CTR_WaitUntilTaskDone()
CTR_DisableStartTrig() CTR_CfgDigEdgeStartTrig() CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig() CTR_SendSoftTrig() CTR_StartTask()
CTR_StopTask() CTR_ReleaseTask() DEV_Release()

CO_WriteFreq()

函数原型：

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

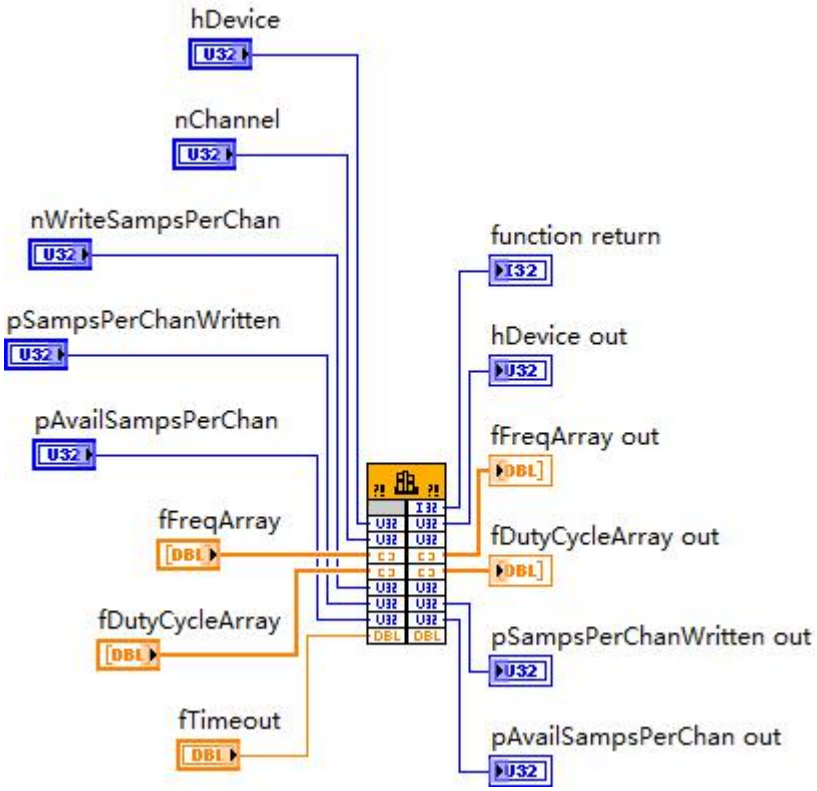
```
BOOL CO_WriteFreq(
                    HANDLE hDevice,
                    U32 nChannel,
                    F64 fFreqArray[],
                    F64 fDutyCycleArray[],
                    U32 nWriteSampsPerChan,
                    U32* pSampsPerChanWritten,
                    U32* pAvailSampsPerChan,
                    F64 fTimeout);
```

Visual Basic:

```
Declare Function CO_WriteFreq Lib "USB2861" (
                                ByVal hDevice As Long, _
                                ByVal nChannel As Long,_
                                ByRef fFreqArray As Double, _
                                ByRef fDutyCycleArray As Double,_
```

ByVal nWriteSampsPerChan As Long,_
ByRef pSampsPerChanWritten As Long,_
ByRef pAvailSampsPerChan As Long,_
ByVal fTimeout As Double) As Boolean

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：创建脉冲滴答生成通道。OUT:脉冲输出引脚,,创建通道后若再调用 CO_WriteTicks()则为可变脉冲输出,否则为固定脉冲输出。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fFreqArray[]入口参数，频率数据缓冲区,存放若干点的频率数据,单位由函数 CO_CreatePulseChanFreq()的参数 nUnits 指定

fDutyCycleArray[] 入口参数，占空比数据缓冲区,存放若干点的占空比数据，取值范围[0.01-0.99]

nWriteSampsPerChan 入口参数，每通道请求写入的点数(单位：样点)。

pSampsPerChanWritten 出口参数，返回每通道实际写入的点数(单位：点), =NULL,表示无须返回。

pAvailSampsPerChan 出口参数，返回当前可写入的采样点数, =NULL,表示无须返回。

fTimeout 入口参数，超时时间，单位：秒

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

相关函数：

DEV_Create()　CO_CreatePulseChanFreq()　CO_CreatePulseChanTime()
CO_CreatePulseChanTicks()　CO_WriteFreq()　CO_WriteTime()

CO_WriteTicks()　　CO_SetWriteRegenMode()

CO_SetEnableInitialDelayOnRetrigger()　　CTR_SetStartTrigRetriggerable()

CTR_CfgSampClkTiming()　　CTR_CfgImplicitTiming()　　CTR_WaitUntilTaskDone()

CTR_DisableStartTrig()　　CTR_CfgDigEdgeStartTrig()　　CTR_DisablePauseTrig()

CTR_CfgDigEdgePauseTrig()　　CTR_SendSoftTrig()　　CTR_StartTask()

CTR_StopTask()　　CTR_ReleaseTask()　　DEV_Release()

CO_WriteTime()

函数原型：

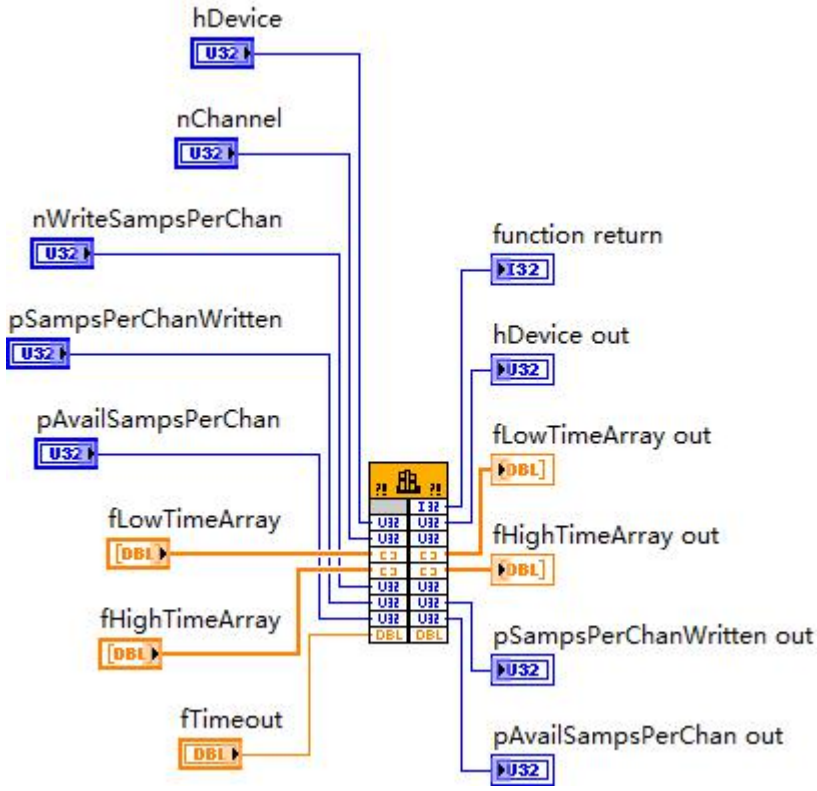
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CO_WriteTime (
                    HANDLE hDevice,
                    U32 nChannel,
                    F64 fLowTimeArray[],
                    F64 fHighTimeArray[],
                    U32 nWriteSampsPerChan,
                    U32* pSampsPerChanWritten,
                    U32* pAvailSampsPerChan,
                    F64 fTimeout);
```

Visual Basic:

```
Declare Function CO_WriteTime Lib "USB2861" (
                                    ByVal hDevice As Long, _
                                    ByVal nChannel As Long,_
                                    ByRef fLowTimeArray As Double, _
                                    ByRef fHighTimeArray As Double,_
                                    ByVal nWriteSampsPerChan As Long,_
                                    ByRef pSampsPerChanWritten As Long,_
                                    ByRef pAvailSampsPerChan As Long,_
                                    ByVal fTimeout As Double) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：向生成任务中写入可变时间数据序列(Write variable time data to the task),事先得由 CO_CreatePulseChanTime()创建时间输出通道。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fLowTimeArray[]入口参数，低电平时间数据缓冲区,存放若干点的时间数据,单位由函数 CO_CreatePulseChanTime()的参数 nUnits 指定。

fHighTimeArray[] 入口参数，高电平时间数据缓冲区,存放若干点的时间数据,单位由函数 CO_CreatePulseChanTime()的参数 nUnits 指定。

nWriteSampsPerChan 入口参数，每通道请求写入的点数(单位：样点)。

pSampsPerChanWritten 出口参数，返回每通道实际写入的点数(单位：点), =NULL,表示无须返回。

pAvailSampsPerChan 出口参数，返回当前可写入的采样点数, =NULL,表示无须返回。

fTimeout 入口参数，超时时间，单位：秒

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

相关函数：

DEV_Create()	CO_CreatePulseChanFreq()	CO_CreatePulseChanTime()
CO_CreatePulseChanTicks()	CO_WriteFreq()	CO_WriteTime()
CO_WriteTicks()	CO_SetWriteRegenMode()	
CO_SetEnableInitialDelayOnRetrigger()		CTR_SetStartTrigRetriggerable()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()

CTR_StopTask()　　CTR_ReleaseTask()　　DEV_Release()

CO_WriteTicks()

函数原型：

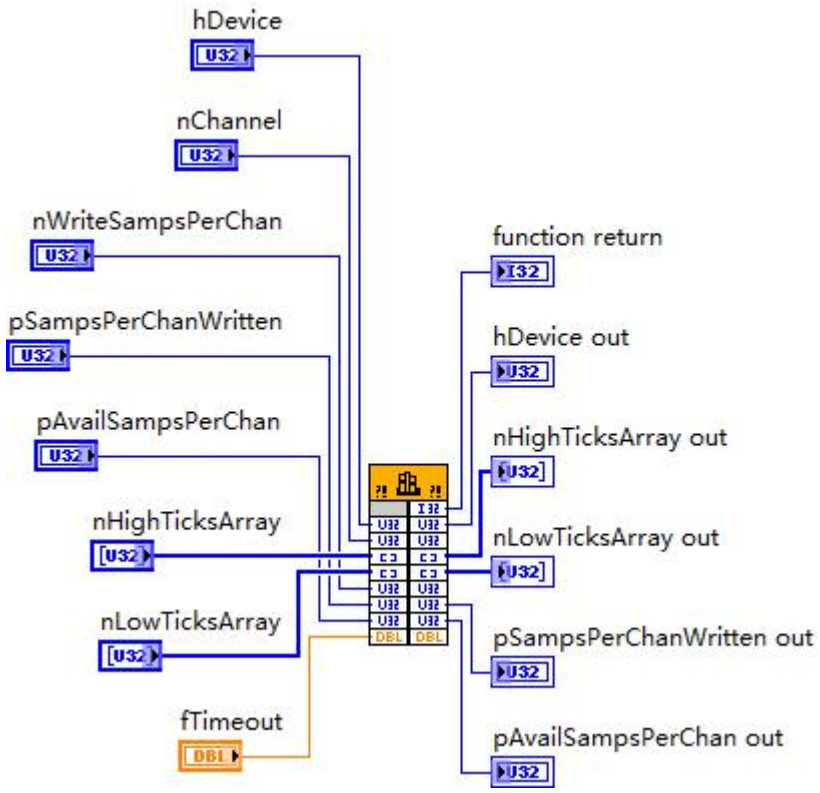
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CO_WriteTicks(
                    HANDLE hDevice,
                    U32 nChannel,
                    U32 nLowTicksArray[],
                    U32 nHighTicksArray[],
                    U32 nWriteSampsPerChan,
                    U32* pSampsPerChanWritten,
                    U32* pAvailSampsPerChan,
                    F64 fTimeout);
```

Visual Basic:

```
Declare Function CO_WriteTicks Lib "USB2861" (
                    ByVal hDevice As Long, _
                    ByVal nChannel As Long,_
                    ByRef nLowTicksArray As Long, _
                    ByRef nHighTicksArray As Long,_
                    ByVal nWriteSampsPerChan As Long,_
                    ByRef pSampsPerChanWritten As Long,_
                    ByRef pAvailSampsPerChan As Long,_
                    ByVal fTimeout As Double) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：向生成任务中写入可变时钟滴答数据序列(Write variable ticks data to the task),事先得由 CO_CreatePulseChanTicks()创建时钟滴答输出通道。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

nLowTicksArray[]入口参数，低电平的时钟滴答数据缓冲区,存放若干点的时钟滴答数据。

nHighTicksArray[] 入口参数，高电平的时钟滴答数据缓冲区,存放若干点的时钟滴答数据。

nWriteSampsPerChan 入口参数，每通道请求写入的点数(单位：样点)。

pSampsPerChanWritten 出口参数，返回每通道实际写入的点数(单位：点), =NULL,表示无须返回。

pAvailSampsPerChan 出口参数，返回当前可写入的采样点数, =NULL,表示无须返回。

fTimeout 入口参数，超时时间，单位：秒

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

相关函数：

DEV_Create()	CO_CreatePulseChanFreq()	CO_CreatePulseChanTime()
CO_CreatePulseChanTicks()	CO_WriteFreq()	CO_WriteTime()
CO_WriteTicks()	CO_SetWriteRegenMode()	
CO_SetEnableInitialDelayOnRetrigger()		CTR_SetStartTrigRetriggerable()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CO_SetWriteRegenMode()

函数原型：

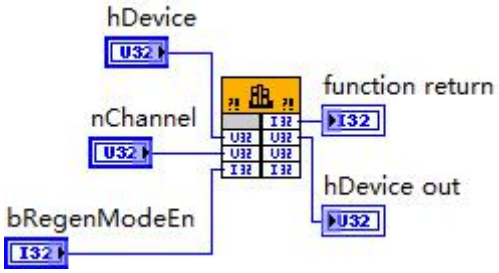
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CO_SetWriteRegenMode (
                    HANDLE hDevice,
                    U32 nChannel,
                    BOOL bRegenModeEn)
```

Visual Basic:

```
Declare Function CO_SetWriteRegenMode Lib "USB2861" (
                                ByVal hDevice As Long,_
                                ByVal nChannel As Long,_
                                ByVal bRegenModeEn As Long) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：设置写入重生成模式,仅在可变脉冲输出(缓冲输出)时有效。不调用此函数时，其默认值为连续重生成模式。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

bRegenModeEn 入口参数，是否允许可重生成, =TRUE:表示可重生成, =FALSE:表示非重生成。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	CO_CreatePulseChanFreq()	CO_CreatePulseChanTime()
CO_CreatePulseChanTicks()	CO_WriteFreq()	CO_WriteTime()
CO_WriteTicks()	CO_SetWriteRegenMode()	
CO_SetEnableInitialDelayOnRetrigger()		CTR_SetStartTrigRetriggerable()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CO_SetEnableInitialDelayOnRetrigger()

函数原型：

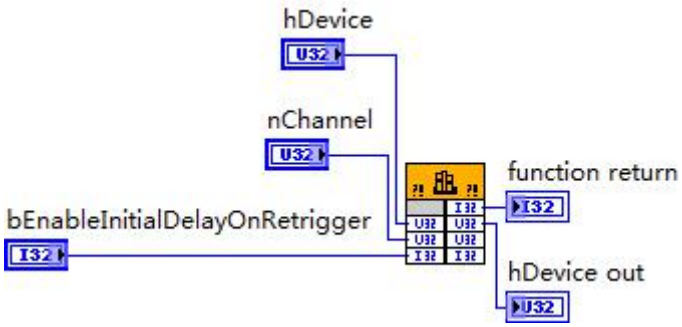
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CO_SetEnableInitialDelayOnRetrigger (
                    HANDLE hDevice,
                    U32 nChannel,
                    BOOL bEnableInitialDelayOnRetrigger)
```

Visual Basic:

```
Declare Function CO_SetEnableInitialDelayOnRetrigger Lib "USB2861" (
                                        ByVal hDevice As Long,
                                        ByVal nChannel As Long,
                                        ByVal bEnableInitialDelayOnRetrigger As Long)
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：设置在重触发中是否有延时状态。不调用此函数时使用默认值即有限点时为禁止，单点输出时使能。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

bEnableInitialDelayOnRetrigger 入口参数，是否允许在重触发后延时,=TRUE:允许延时，=FALSE:禁止延时。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	CO_CreatePulseChanFreq()	CO_CreatePulseChanTime()
CO_CreatePulseChanTicks()	CO_WriteFreq()	CO_WriteTime()
CO_WriteTicks()	CO_SetWriteRegenMode()	
CO_SetEnableInitialDelayOnRetrigger()		CTR_SetStartTrigRetriggerable()
CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()	CTR_WaitUntilTaskDone()
CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()	CTR_DisablePauseTrig()
CTR_CfgDigEdgePauseTrig()	CTR_SendSoftTrig()	CTR_StartTask()
CTR_StopTask()	CTR_ReleaseTask()	DEV_Release()

CTR_CfgSampClkTiming()

函数原型：

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

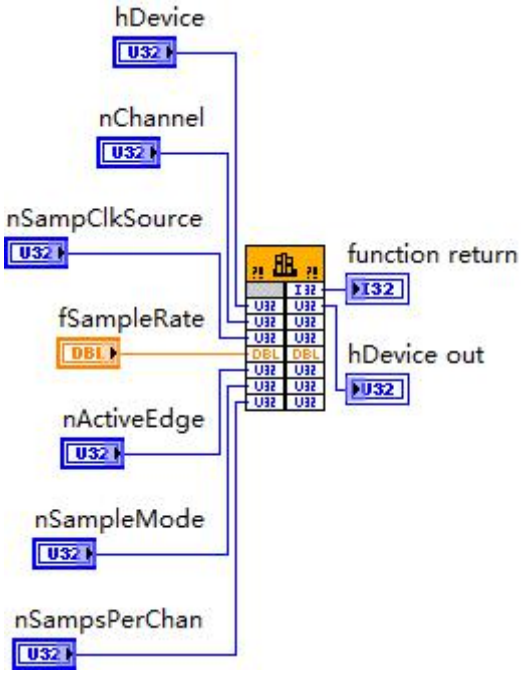
```
BOOL CTR_CfgSampClkTiming(
                        HANDLE hDevice,
                        U32 nChannel,
                        U32 nSampClkSource,
                        F64 fSampleRate,
                        U32 nActiveEdge,
                        U32 nSampleMode,
                        U32 nSampsPerChan);
```

Visual Basic:

```
Declare Function CTR_CfgSampClkTiming Lib "USB2861" (
                                    ByVal hDevice As Long, _
                                    ByVal nChannel As Long,_
```

ByVal nSampClkSourceAs Long, _
ByVal fSampleRate As Double,_
ByVal nActiveEdge As Long,_
ByVal nSampleModeAs Long,_
ByVal nSampsPerChanAs Long) As Boolean

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：配置采样时钟及采样模式(显式时钟)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

nSampClkSource 入口参数，采样时钟源,详见上面的 nSampClkSource 选项定义。

常量名	常量值	功能定义	备注
CTR_SAMPCLKSRC_LOCAL	0	本地时钟源(板载内部时钟源)	
CTR_SAMPCLKSRC_PFI0	1	外部时钟源 PFI0	
CTR_SAMPCLKSRC_PFI1	1	外部时钟源 PFI1	
CTR_SAMPCLKSRC_PFI2	1	外部时钟源 PFI2	
CTR_SAMPCLKSRC_PFI3	1	外部时钟源 PFI3	
CTR_SAMPCLKSRC_PFI4	1	外部时钟源 PFI4	
CTR_SAMPCLKSRC_PFI5	1	外部时钟源 PFI5	
CTR_SAMPCLKSRC_PFI6	1	外部时钟源 PFI6	
CTR_SAMPCLKSRC_PFI7	1	外部时钟源 PFI7	
CTR_SAMPCLKSRC_PFI8	1	外部时钟源 PFI8	
CTR_SAMPCLKSRC_PFI9	1	外部时钟源 PFI9	
CTR_SAMPCLKSRC_PFI10	1	外部时钟源 PFI10	
CTR_SAMPCLKSRC_PFI11	1	外部时钟源 PFI11	
CTR_SAMPCLKSRC_PFI12	1	外部时钟源 PFI12	
CTR_SAMPCLKSRC_PFI13	1	外部时钟源 PFI13	
CTR_SAMPCLKSRC_PFI14	1	外部时钟源 PFI14	
CTR_SAMPCLKSRC_PFI15	1	外部时钟源 PFI15	

fSampleRate 入口参数，采样速率。

nActiveEdge 入口参数，时钟边沿

常量名	常量值	功能定义	备注
CTR_SAMPCLKEDGE_FALLING	0	下降沿	
CTR_SAMPCLKEDGE_RISING	1	上升沿	

nSampleMode 入口参数，采样模式。

常量名	常量值	功能定义	备注
CTR_SAMPMODE_ONE_DEMAND	0	单点采样(按需)	
CTR_SAMPMODE_ONE_HWTIMED	1	单点采样(硬件定时,Hardware Timed，本设备暂时不支持)	
CTR_SAMPMODE_FINITE	2	有限点采样	
CTR_SAMPMODE_CONTINUOUS	3	连续采样	

nSampsPerChan 入口参数，每通道采样点数。

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

相关函数：

DEV_Create()　　CTR_CfgSampClkTiming()　　CTR_CfgImplicitTiming()

CTR_WaitUntilTaskDone()　CTR_DisableStartTrig()　CTR_CfgDigEdgeStartTrig()
CTR_SetStartTrigRetriggerable()　CTR_DisablePauseTrig()　CTR_CfgDigEdgePauseTrig()
CTR_SendSoftTrig()　CTR_StartTask()　CTR_StopTask()
CTR_ReleaseTask()　DEV_Release()

CTR_CfgImplicitTiming()

函数原型：

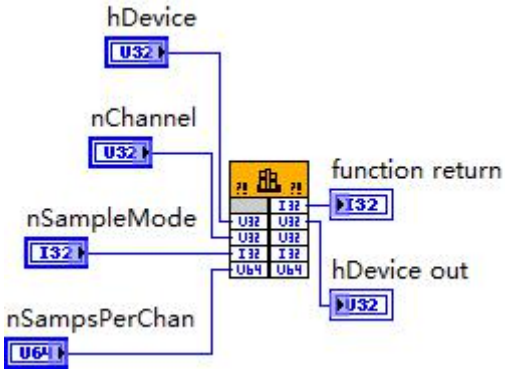
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CTR_CfgImplicitTiming(
                    HANDLE hDevice,
                    U32 nChannel,
                    U32 nSampleMode,
                    U32 nSampsPerChan);
```

Visual Basic:

```
Declare Function CTR_CfgImplicitTiming Lib "USB2861" (
                                ByVal hDevice As Long, _
                                ByVal nChannel As Long,_
                                ByVal nSampleModeAs Long,_
                                ByVal nSampsPerChanAs Long) As Boolean
```

LabVIEW



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：配置隐式时钟(无须指定时钟源和采样速率,如半周期测量、频率测量,内部使用固定的基准时钟)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

nSampleMode 入口参数，采样模式。

常量名	常量值	功能定义	备注
CTR_SAMPMODE_ONE_DEMAND	0	单点采样(按需)	
CTR_SAMPMODE_ONE_HWTIMED	1	单点采样(硬件定时,Hardware Timed，本设备暂时不支持)	
CTR_SAMPMODE_FINITE	2	有限点采样	
CTR_SAMPMODE_CONTINUOUS	3	连续采样	

nSampsPerChan 入口参数，每通道采样点数。

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

相关函数：

DEV_Create()	CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()
CTR_WaitUntilTaskDone()	CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()
CTR_SetStartTrigRetriggerable()	CTR_DisablePauseTrig()	CTR_CfgDigEdgePauseTrig()
CTR_SendSoftTrig()	CTR_StartTask()	CTR_StopTask()
CTR_ReleaseTask()	DEV_Release()	

CTR_WaitUntilTaskDone()

函数原型：

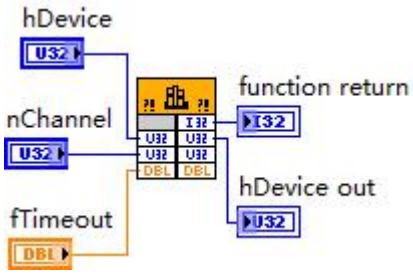
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CTR_WaitUntilTaskDone (HANDLE hDevice, U32 nChannel, F64 fTimeout)
```

Visual Basic:

```
Declare Function CTR_WaitUntilTaskDone Lib "USB2861" (
                                        ByVal hDevice As Long,_
                                        ByVal nChannel As Long,_
                                        ByVal fTimeout As Double,_
                                        ) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：采样任务结束前等待,返回 TRUE 表示采样任务结束。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

fTimeout 入口参数，用于等待的时间，单位：秒(S)。

返回值：如果采样任务结束，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()
CTR_WaitUntilTaskDone()	CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()
CTR_SetStartTrigRetriggerable()	CTR_DisablePauseTrig()	CTR_CfgDigEdgePauseTrig()
CTR_SendSoftTrig()	CTR_StartTask()	CTR_StopTask()
CTR_ReleaseTask()	DEV_Release()	

CTR_DisableStartTrig()

函数原型：

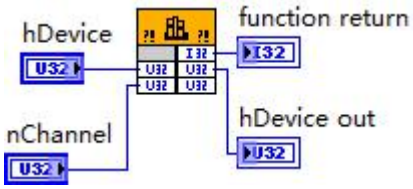
Visual C++ / C++Builder / LabWindows/CVI:

BOOL CTR_DisableStartTrig(HANDLE hDevice, U32 nChannel)

Visual Basic:

```
Declare Function CTR_DisableStartTrig Lib "USB2861" (
                                        ByVal hDevice As Long,_
                                        ByVal nChannel As Long,_
                                        ) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：禁止开始触发。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

返回值：如果返回 TRUE，则表示禁止成功，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()
CTR_WaitUntilTaskDone()	CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()
CTR_SetStartTrigRetriggerable()	CTR_DisablePauseTrig()	CTR_CfgDigEdgePauseTrig()
CTR_SendSoftTrig()	CTR_StartTask()	CTR_StopTask()
CTR_ReleaseTask()	DEV_Release()	

CTR_CfgDigEdgeStartTrig()

函数原型：

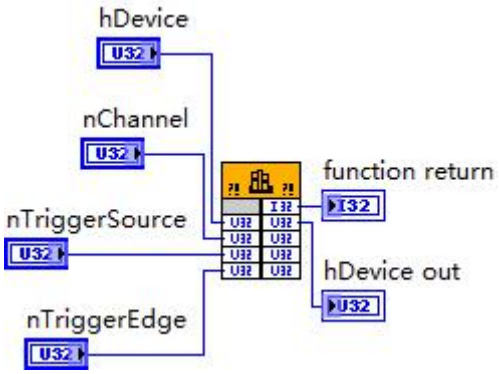
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CTR_CfgDigEdgeStartTrig(HANDLE hDevice,
                                        U32 nChannel,_
                                        U32 nTriggerSource,_
                                        U32 nTriggerEdge);
```

Visual Basic:

```
Declare Function CTR_CfgDigEdgeStartTrig Lib "USB2861" (
                                        ByVal hDevice As Long,_
                                        ByVal nChannel As Long,_
                                        ByVal nTriggerSource As Long,_
                                        ByVal nTriggerEdge As Long,_
                                        ) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：配置开始触发(数字边沿)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

nTriggerSource 入口参数，数字触发源，取值范围[0, 15]，分别代表 PFI0, PFI1...PFI15。

nTriggerEdge 入口参数，数字触发边沿。

常量名	常量值	功能定义	备注
CTR_TRIGDIR_FALLING	0	下降沿/低电平	
CTR_TRIGDIR_RISING	1	上升沿/高电平	
CTR_TRIGDIR_CHANGING	2	变化(即上下边沿/高低电平均有效)	

返回值：如果返回 TRUE，则表示禁止成功，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create() CTR_CfgSampClkTiming() CTR_CfgImplicitTiming()
CTR_WaitUntilTaskDone() CTR_DisableStartTrig() CTR_CfgDigEdgeStartTrig()
CTR_SetStartTrigRetriggerable() CTR_DisablePauseTrig() CTR_CfgDigEdgePauseTrig()
CTR_SendSoftTrig() CTR_StartTask() CTR_StopTask()
CTR_ReleaseTask() DEV_Release()

CTR_SetStartTrigRetriggerable()

函数原型：

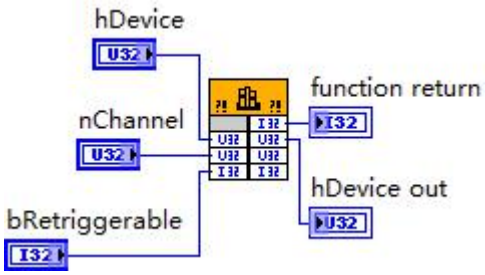
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CTR_SetStartTrigRetriggerable(
                    HANDLE hDevice,
                    U32 nChannel,
                    BOOL bRetriggerable)
```

Visual Basic:

```
Declare Function CTR_SetStartTrigRetriggerable Lib "USB2861" (
                                        ByVal hDevice As Long,
                                        ByVal nChannel As Long,
                                        ByVal bRetriggerable As Long) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：允许开始触发的重触发(单点和有限点支持重触发),不调用此函数时为默认禁止。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

bRetriggerable 入口参数，是否允许重触发,=TRUE:允许重触发，=FALSE:禁止重触发。

返回值：如果调用成功，则返回 TRUE，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create()	CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()
CTR_WaitUntilTaskDone()	CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()
CTR_SetStartTrigRetriggerable()	CTR_DisablePauseTrig()	CTR_CfgDigEdgePauseTrig()
CTR_SendSoftTrig()	CTR_StartTask()	CTR_StopTask()
CTR_ReleaseTask()	DEV_Release()	

CTR_DisablePauseTrig()

函数原型：

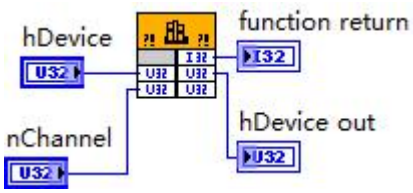
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CTR_DisablePauseTrig(HANDLE hDevice, U32 nChannel)
```

Visual Basic:

```
Declare Function CTR_DisablePauseTrig Lib "USB2861" (
                                        ByVal hDevice As Long,_
                                        ByVal nChannel As Long,_
                                        ) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：禁止暂停触发。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

返回值：如果返回 TRUE，则表示禁止成功，否则返回 FALSE，可立即调用 WIN32 API 函数

GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create() CTR_CfgSampClkTiming() CTR_CfgImplicitTiming()
CTR_WaitUntilTaskDone() CTR_DisableStartTrig() CTR_CfgDigEdgeStartTrig()
CTR_SetStartTrigRetriggerable() CTR_DisablePauseTrig() CTR_CfgDigEdgePauseTrig()
CTR_SendSoftTrig() CTR_StartTask() CTR_StopTask()
CTR_ReleaseTask() DEV_Release()

CTR_CfgDigEdgePauseTrig()

函数原型：

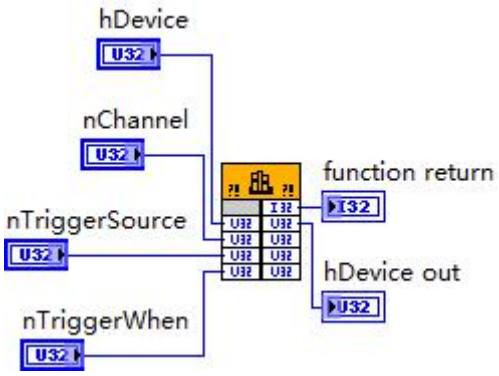
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CTR_CfgDigEdgePauseTrig(HANDLE hDevice,
                             U32 nChannel,_
                             U32 nTriggerSource,_
                             U32 nTriggerWhen);
```

Visual Basic:

```
Declare Function CTR_CfgDigEdgePauseTrig Lib "USB2861" (
                                        ByVal hDevice As Long,_
                                        ByVal nChannel As Long,_
                                        ByVal nTriggerSource As Long,_
                                        ByVal nTriggerWhenAs Long,_
                                        ) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：配置暂停触发(数字边沿)。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

nTriggerSource 入口参数，数字触发源，取值范围[0, 15]，分别代表 PFI0, PFI1...PFI15。

nTriggerWhen 入口参数，数字触发时机，0=低电平暂停触发，1=高电平暂停触发。

返回值：如果返回 TRUE，则表示禁止成功，否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create() CTR_CfgSampClkTiming() CTR_CfgImplicitTiming()
CTR_WaitUntilTaskDone() CTR_DisableStartTrig() CTR_CfgDigEdgeStartTrig()

CTR_SetStartTrigRetriggerable() CTR_DisablePauseTrig() CTR_CfgDigEdgePauseTrig()
CTR_SendSoftTrig() CTR_StartTask() CTR_StopTask()
CTR_ReleaseTask() DEV_Release()

CTR_SendSoftTrig()

函数原型：

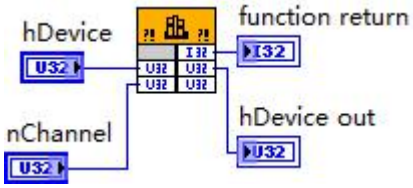
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CTR_SendSoftTrig(HANDLE hDevice, U32 nChannel)
```

Visual Basic:

```
Declare Function CTR_SendSoftTrig Lib "USB2861" (
                                        ByVal hDevice As Long,
                                        ByVal nChannel As Long) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：发送软件触发事件(Send Software Trigger),软件触发也叫强制触发。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

返回值：如果返回 TRUE，则表示成功发送了软件强制触发， 否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create() CTR_CfgSampClkTiming() CTR_CfgImplicitTiming()
CTR_WaitUntilTaskDone() CTR_DisableStartTrig() CTR_CfgDigEdgeStartTrig()
CTR_SetStartTrigRetriggerable() CTR_DisablePauseTrig() CTR_CfgDigEdgePauseTrig()
CTR_SendSoftTrig() CTR_StartTask() CTR_StopTask()
CTR_ReleaseTask() DEV_Release()

CTR_StartTask()

函数原型：

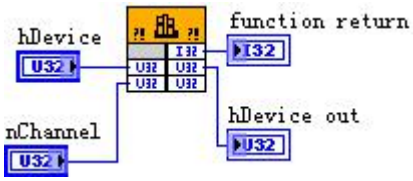
Visual C++ / C++Builder / LabWindows/CVI:

```
BOOL CTR_StartTask(HANDLE hDevice, U32 nChannel)
```

Visual Basic:

```
Declare Function CTR_StartTask Lib "USB2861" (
                                        ByVal hDevice As Long,
                                        ByVal nChannel As Long) As Boolean
```

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：开始 CTR 计数(Start task for counter)，必须在成功创建计数器通道之后才能调用此函数，调用该函数后 CTR 采样任务立即开始。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

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

相关函数：

DEV_Create()	CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()
CTR_WaitUntilTaskDone()	CTR_DisableStartTrig()	CTR_CfgDigEdgeStartTrig()
CTR_SetStartTrigRetriggerable()	CTR_DisablePauseTrig()	CTR_CfgDigEdgePauseTrig()
CTR_SendSoftTrig()	CTR_StartTask()	CTR_StopTask()
CTR_ReleaseTask()	DEV_Release()	

CTR_StopTask()

函数原型：

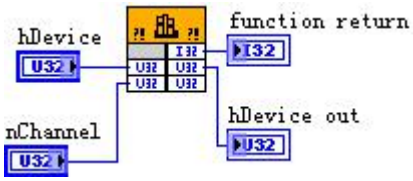
Visual C++ / C++Builder / LabWindows/CVI:

BOOL CTR_StopTask(HANDLE hDevice, U32 nChannel)

Visual Basic:

Declare Function CTR_StopTask Lib "USB2861" (ByVal hDevice As Long, ByVal nChannel As Long) As Boolean

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：开始 CTR 计数(Stop task for counter)，必须在成功调用 CTR_StartTask()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_StartTask()继续计数。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

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

相关函数：

DEV_Create()	CTR_CfgSampClkTiming()	CTR_CfgImplicitTiming()

CTR_WaitUntilTaskDone() CTR_DisableStartTrig() CTR_CfgDigEdgeStartTrig()
CTR_SetStartTrigRetriggerable() CTR_DisablePauseTrig() CTR_CfgDigEdgePauseTrig()
CTR_SendSoftTrig() CTR_StartTask() CTR_StopTask()
CTR_ReleaseTask() DEV_Release()

CTR_ReleaseTask()

函数原型：

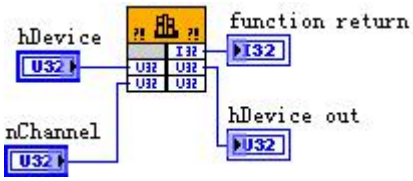
Visual C++ / C++Builder / LabWindows/CVI:

BOOL CTR_ReleaseTask (HANDLE hDevice, U32 nChannel)

Visual Basic:

Declare Function CTR_ReleaseTask Lib "USB2861" (ByVal hDevice As Long, ByVal nChannel As Long) As Boolean

LabVIEW：



请参考 USB2861.lvlib 库文件及相关演示 vi。

功能：释放 CTR (Relase task for Counter)，必须在成功调用 CTR_InitTask()函数后才能调用此函数。

参数：

hDevice 入口参数，设备对象句柄，由 DEV_Create()函数创建，该句柄指向要访问的设备。

nChannel 入口参数，计数器通道号，取值范围[0, 3]。

返回值：如果调用成功，则返回 TRUE，CTR 被成功释放， 否则返回 FALSE，可立即调用 WIN32 API 函数 GetLastError()捕获错误码以确定具体原因。

相关函数：

DEV_Create() CTR_CfgSampClkTiming() CTR_CfgImplicitTiming()
CTR_WaitUntilTaskDone() CTR_DisableStartTrig() CTR_CfgDigEdgeStartTrig()
CTR_SetStartTrigRetriggerable() CTR_DisablePauseTrig() CTR_CfgDigEdgePauseTrig()
CTR_SendSoftTrig() CTR_StartTask() CTR_StopTask()
CTR_ReleaseTask() DEV_Release()

4 各种结构体描述

4.1 AI_PARAM（AI 工作参数结构体）

4.1.1 AI_CH_PARAM(AI 通道参数结构体)

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

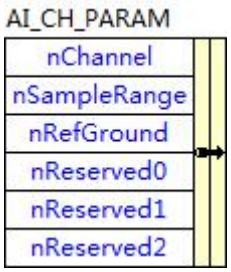
```
typedef struct _AI_CH_PARAM
{
    U32 nChannel;
    U32 nSampleRange;
    U32 nRefGround;
```

```
    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
}
```

Visual Basic：

```
Private Type AI_CH_PARAM
    nChannel As Long
    nSampleRange As Long
    nRefGround As Long

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

LabVIEW:



nChannel

AI 物理通道号，取值范围[0, 63]，即 AI0－AI63。

nSampleRange

AI 采样范围(Sample Range)，取值范围如下表：

常量名	常量值	采样范围
AI_SAMPRANGE_N10_P10V	0	±10V
AI_SAMPRANGE_N5_P5V	1	±5V
AI_SAMPRANGE_N2_P2V	2	±2V
AI_SAMPRANGE_N1_P1V	3	±1V

nRefGround

参考地(Referenced Ground)，取值范围[0, 2]，具体定义参考下表

常量名	常量值	功能定义	备注
AI_REFGND_RSE	0	单端输入，有参考地(Referenced Single Endpoint)	
AI_REFGND_NRSE	1	单端输入，无参考地(Non Referenced Single Endpoint)	
AI_REFGND_DI	2	差分输入(Differential)，也叫双端输入	

当参考地选择为单端时，则采样信号通过 AI0-AI31 分 32 路输入；当参考地为差分输入时，采样信号通过[AI0+, AI32-]；[AI1+、AI33-]； [AI2+、AI34-]； [AI7+、AI35-]…共 32 路输入。

nReserved0-3

保留字段。

4.1.2 AI_START_TRIG (AI 开始触发参数结构体)

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

```
typedef struct _AI_START_TRIG
{
    U32 nTriggerType;
    U32 nTriggerSource;
    U32 nTriggerDir;
    F32 fTriggerLevelTop;
    F32 fTriggerLevelBtm;
    U32 nTriggerSens;
    U32 nDelaySamps;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
} AI_START_TRIG, *PAI_START_TRIG;
```

Visual Basic：

```
Private Type AI_START_TRIG
    nTriggerType As Long
    nTriggerSource As Long
    nTriggerDir As Long

    fTriggerLevelTop As Long
    fTriggerLevelBtm As Long
    nTriggerSens As Double
    nDelaySamps As Long

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

nTriggerType

触发类型

常量名	常量值	功能定义
AI_START_TRIGTYPE_NONE	0	无触发(等同于软件强制触发)

AI_START_TRIGTYPE_ANALOG_EDGE	1	模拟边沿触发类型
AI_START_TRIGTYPE_ANALOG_WIN	2	模拟窗触发类型
AI_START_TRIGTYPE_DIGIT_EDGE	3	数字边沿触发类型
AI_START_TRIGTYPE_DIGIT_PATTERN		数字模式触发类型(保留,未提供)

nTriggerDir

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	默认值
AI_TRIGDIR_RISING	1	上升沿触发	
AI_TRIGDIR_CHANGING	2	变化触发(上或下边沿触发)	

nTriggerSens

AI 触发灵敏度（Trigger Sense)，单位：微秒(uS)。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。

nDelaySamps

触发延迟点数（Delay Samples）。取值范围 32 位有效[0, 4294967295]。其值等于 0 时为后触发（Post Trigger）；其值大于 0 时为正延迟触发(Delay Trigger)。就是在开始触发产生时，再延迟若干个采样点后再记录数据，其延迟的点数就是由 nDelaySamps 指定，而单个点的时间周期由 nSampleRate 指定的每通道采样速率决定的。比如 nDelaySamps=100，nSampleRate=1000Hz（即每采样点周期为 1 毫秒），则意味着在开始生成任务后，当发生开始触发时再等待 100 毫秒（1 毫秒*100）才实际进入数据采样与记录阶段。在有些应用中，用户关注的焦点信号是在触发事件之后的一段时间才发生，那么这个功能就是满足此种应用的。

4.1.3 AI_PAUSE_TRIG (AI 暂停触发参数结构体)

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

```
typedef struct _AI_START_TRIG
{
    U32 nTriggerType;
    U32 nTriggerSource;
    U32 nTriggerDir;
    F32 fTriggerLevelTop;
    F32 fTriggerLevelBtm;
    U32 nTriggerSens;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
```

```
} AI_PAUSE_TRIG, *PAI_PAUSE_TRIG;
```

Visual Basic：

```
Private Type AI_PAUSE_TRIG
    nTriggerType As Long
    nTriggerSource As Long
    nTriggerDir As Long

    fTriggerLevelTop As Long
    fTriggerLevelBtm As Long
    nTriggerSens As Double

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

nTriggerType

触发类型

常量名	常量值	功能定义
AI_PAUSE_TRIGTYPE_NONE	0	无触发(等同于软件强制触发)
AI_PAUSE _TRIGTYPE_ANALOG_EDGE	1	模拟边沿触发类型
AI_PAUSE _TRIGTYPE_ANALOG_WIN	2	模拟窗触发类型
AI_PAUSE _TRIGTYPE_DIGIT_EDGE	3	数字边沿触发类型
AI_PAUSE _TRIGTYPE_DIGIT_PATTERN		数字模式触发类型(保留,未提供)

nTriggerDir

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	默认值
AI_TRIGDIR_RISING	1	上升沿触发	
AI_TRIGDIR_CHANGING	2	变化触发(上或下边沿触发)	

nTriggerSens

AI 触发灵敏度（Trigger Sense)，单位：微秒(uS)。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。

4.1.4 AI_PARAM(AI 工作参数结构体）

```
typedef struct _AI_PARAM
```

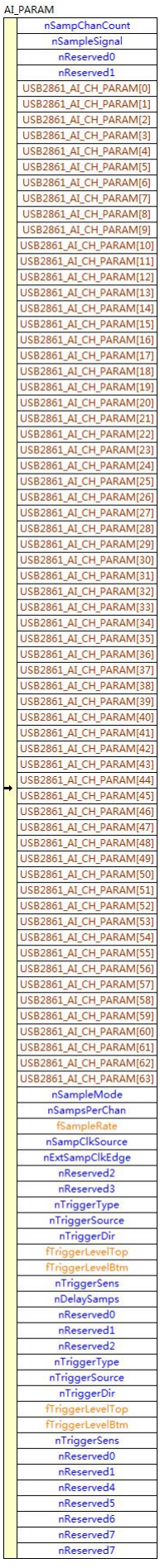
```
{
    U32 nSampChanCount;
    U32 nSampleSignal;
    U32 nReserved0;
    U32 nReserved1;
    AI_CH_PARAM CHParam[64];

    U32 nSampleMode;
    U32 nSampsPerChan;
    F64 fSampleRate;
    U32 nSampClkSource;
    U32 nExtSampClkEdge;
    U32 nReserved2;
    U32 nReserved3;

    AI_START_TRIG StartTrig;
    AI_PAUSE_TRIG PauseTrig;

    U32 nReserved4;
    U32 nReserved5;
    U32 nReserved6;
    U32 nReserved7;
} AI_PARAM, *PAI_PARAM;
```

Visual Basic：

```
Private Type AI_PARAM
    nSampChanCount As Long
    nSampleSignal As Long
    nReserved0 As Long
    nReserved1 As Long
    CHParam(0 to 63) As AI_CH_PARAM

    nSampleMode As Long
    nSampsPerChan As Long
    fSampleRate As Long
    nSampClkSource As Long
    nExtSampClkEdge As Long
    nReserved2 As Long
    nReserved3 As Long

    StartTrig As AI_START_TRIG
    PauseTrig As AI_PAUSE_TRIG

    nReserved4 As Long
```

```
    nReserved5 As Long
    nReserved6 As Long
    nReserved7 As Long
End Type
```

LabVIEW:


AI_PARAM
nSampChanCount
nSampleSignal
nReserved0
nReserved1
USB2861_AI_CH_PARAM[0]
USB2861_AI_CH_PARAM[1]
USB2861_AI_CH_PARAM[2]
USB2861_AI_CH_PARAM[3]
USB2861_AI_CH_PARAM[4]
USB2861_AI_CH_PARAM[5]
USB2861_AI_CH_PARAM[6]
USB2861_AI_CH_PARAM[7]
USB2861_AI_CH_PARAM[8]
USB2861_AI_CH_PARAM[9]
USB2861_AI_CH_PARAM[10]
USB2861_AI_CH_PARAM[11]
USB2861_AI_CH_PARAM[12]
USB2861_AI_CH_PARAM[13]
USB2861_AI_CH_PARAM[14]
USB2861_AI_CH_PARAM[15]
USB2861_AI_CH_PARAM[16]
USB2861_AI_CH_PARAM[17]
USB2861_AI_CH_PARAM[18]
USB2861_AI_CH_PARAM[19]
USB2861_AI_CH_PARAM[20]
USB2861_AI_CH_PARAM[21]
USB2861_AI_CH_PARAM[22]
USB2861_AI_CH_PARAM[23]
USB2861_AI_CH_PARAM[24]
USB2861_AI_CH_PARAM[25]
USB2861_AI_CH_PARAM[26]
USB2861_AI_CH_PARAM[27]
USB2861_AI_CH_PARAM[28]
USB2861_AI_CH_PARAM[29]
USB2861_AI_CH_PARAM[30]
USB2861_AI_CH_PARAM[31]
USB2861_AI_CH_PARAM[32]
USB2861_AI_CH_PARAM[33]
USB2861_AI_CH_PARAM[34]
USB2861_AI_CH_PARAM[35]
USB2861_AI_CH_PARAM[36]
USB2861_AI_CH_PARAM[37]
USB2861_AI_CH_PARAM[38]
USB2861_AI_CH_PARAM[39]
USB2861_AI_CH_PARAM[40]
USB2861_AI_CH_PARAM[41]
USB2861_AI_CH_PARAM[42]
USB2861_AI_CH_PARAM[43]
USB2861_AI_CH_PARAM[44]
USB2861_AI_CH_PARAM[45]
USB2861_AI_CH_PARAM[46]
USB2861_AI_CH_PARAM[47]
USB2861_AI_CH_PARAM[48]
USB2861_AI_CH_PARAM[49]
USB2861_AI_CH_PARAM[50]
USB2861_AI_CH_PARAM[51]
USB2861_AI_CH_PARAM[52]
USB2861_AI_CH_PARAM[53]
USB2861_AI_CH_PARAM[54]
USB2861_AI_CH_PARAM[55]
USB2861_AI_CH_PARAM[56]
USB2861_AI_CH_PARAM[57]
USB2861_AI_CH_PARAM[58]
USB2861_AI_CH_PARAM[59]
USB2861_AI_CH_PARAM[60]
USB2861_AI_CH_PARAM[61]
USB2861_AI_CH_PARAM[62]
USB2861_AI_CH_PARAM[63]
nSampleMode
nSampsPerChan
fSampleRate
nSampClkSource
nExtSampClkEdge
nReserved2
nReserved3
nTriggerType
nTriggerSource
nTriggerDir
fTriggerLevelTop
fTriggerLevelBtm
nTriggerSens
nDelaySamps
nReserved0
nReserved1
nReserved2
nTriggerType
nTriggerSource
nTriggerDir
fTriggerLevelTop
fTriggerLevelBtm
nTriggerSens
nReserved0
nReserved1
nReserved4
nReserved5
nReserved6
nReserved7
nReserved7

请参考 USB2861.lvlib 库文件及相关演示 vi

nSampChanCount

采样通道数量(Sample Channel Count)，进入采样过程的通道个数，取值范围[1，8]。即决定了 CHParam[]通道组阵列中有效单元的个数。若 nSampChanCount=1，则表示仅 CHParam[0]单元决定的物理通道号有效；若 nSampChanCount=2，则表示仅 CHParam[0]、CHParam[1]两个单元决定的物理通道号有效；若 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[64]

通道组(Channel Parameter)，共 64 个单元，分别控制 64 个采样通道的选择。该参数的有效单元个数由 nSampChanCount 参数决定。每个有效单元决定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《4.1.1 AI_CH_PARAM(AI 通道参数结构体)》。

nSampleSignal

AI 采样信号(Sample Signal)，取值范围为[0, 11]，如下表：

常量名	常量值	功能定义	备注
AI_SAMPSIGNAL_AI	0	AI 通道输入信号	默认值
AI_SAMPSIGNAL_0V	1	0V(AGND)	
AI_SAMPSIGNAL_4D096V	2	4.096V(DC)	
AI_SAMPSIGNAL_N4D096V	3	-4.096V(DC)	
AI_SAMPSIGNAL_AO0	4	AO0	
AI_SAMPSIGNAL_NAO0	5	-AO0	
AI_SAMPSIGNAL_AO1	6	AO1	
AI_SAMPSIGNAL_NAO1	7	-AO1	
AI_SAMPSIGNAL_AO2	8	AO2	
AI_SAMPSIGNAL_NAO2	9	-AO2	
AI_SAMPSIGNAL_AO3	10	AO3	
AI_SAMPSIGNAL_NAO3	11	-AO3	

nSampleMode

AI 采样模式(Sample Mode)，取值[0, 3]，具体定义见下表：

常量名	常量值	功能定义	备注
AI_SAMPMODE_ONE_DEMAND	0	软件按需单点采样	
AI_SAMPMODE_ONE_HWTIMED	1	硬件定时单点采样	本设备不支持
AI_SAMPMODE_FINITE	2	有限点采样	
AI_SAMPMODE_CONTINUOUS	3	连续采样	

软件按需单点采样模式：就是调用 AI_StartTask()后，AI 任务只是就绪，但并不实际采样数据，而要等到每次软件调用 AI_ReadAnalog()或 AI_ReadBinary()函数时任务才开始实际采样，

且每个通道仅采样一个点的数据，并以最快的速度返回。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，比如应用在对时间边界没有严格要求的 PID，PLC 等快速伺服闭环控制系统。不支持触发和外时钟。

硬件定时单点采样模式：在调用 AI_StartTask()后，AI 采集任务就会按照 AIParam.fSampleRate 设定的速率定时地，不断的为每个通道采样单点数据，每次调用 AI_ReadAnalog()或 AI_ReadBinary()时也为每个通道仅返回一个点的数据，且以最快的速度返回。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，且时间有严格要求的确定的应用中，比如应用在对时间边界有严格要求的 PID，PLC 等快速伺服闭环控制系统。不支持触发和外时钟。

有限点采样模式：按照设定好的采样速率和触发条件，触发模式等参数进行定长时间的、限制点数的、连续的、等间隔的波形数据采集。在开始采集任务后，达到触发条件并采样完成指定点数的数据后，采集任务就会自动停止。这个功能主要是应用在频繁捕捉触发事件、有点数限制和时间长度限制、尽可能还原外界信号的场合。

连续采样模式：按照设定好的采样速率和触发条件，触发模式等参数进行长时间的、不限点数的、连续的、等间隔的波形数据采集，在开始采集任务后，只要不软件停止采集任务，其采集任务是永远不会终止的。这个功能主要是应用在不限制点数和时间长度的，且不丢点的，尽可能还原外界信号的场合。

nSampsPerChan

AI 每通道待读取点数(Samples Per Channel)。

单点采样模式：该参数无意义；

有限点采样模式：该参数表示每通道采样点数。取值范围为[2，1024*1024*16]样点，最大取值还要受制于系统可用内存，采样通道数等；

连续采样模式：决定着触发采样事件 hSampEvent 时的点数条件。比如指定该参数值为 1024 点，则每采样到不小于 1024 点时就会触发采样事件 hSampEvent，它也决定着每次调用 AI_ReadAnalog()或 AI_ReadBinary()时能最快返回的点数边界（请注意：是不小于 nSampsPerChan，意思是不可能正好等于 nSampsPerChan 时就能得到事件通知，而是通常在超过 nSampsPerChan 时才能得到事件通知）。即该参数的取值大小决定着每两次读到数据的时间间隔，也就是实时响应度。点数越小，实时响应就越高，反之越低。但不能为了一味的追求实时响应度就将该参数的值设得很小，这个还要看采样速率的高低，如果采样速率很高，而 nSampsPerChan 的值很小，则可能造成任务负担过重而发生缓冲区溢出，以致造成丢点现象的发生。建议实时响应度不低于 20 个毫秒是比较合适的。比如每通道采样速率为 100Ksps，即 10 微秒一个点，则 nSampsPerChan 不小于 2000 个点是比较合适的。该参数的取值范围为[2, 1024*1024]，具体还要受制于系统可用内存和采样通道数。

fSampleRate

AI 采样速率(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，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 fSampleRate 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(100000sps)、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 100000sps/3 = 33333sps，比如采样通道数量为 4 个通道，则该参数最大取值为 100000sps/4 = 25000sps；比如采样通道数量为 8 个通道，则该参数最大取值为 100000sps/8 = 12500sps。

nSampClkSource

时钟源(Sample Clock Source)。该参数的取值范围为[0，1]，具体定义如下表：

常量名	常量值	功能定义	备注
AI_SAMPCLKSRC_LOCAL	0	本地时钟源(板载内部时钟源)	
AI_SAMPCLKSRC_PFI0	1	外部时钟源 PFI0	
AI_SAMPCLKSRC_PFI1	1	外部时钟源 PFI1	
AI_SAMPCLKSRC_PFI2	1	外部时钟源 PFI2	
AI_SAMPCLKSRC_PFI3	1	外部时钟源 PFI3	
AI_SAMPCLKSRC_PFI4	1	外部时钟源 PFI4	
AI_SAMPCLKSRC_PFI5	1	外部时钟源 PFI5	
AI_SAMPCLKSRC_PFI6	1	外部时钟源 PFI6	
AI_SAMPCLKSRC_PFI7	1	外部时钟源 PFI7	
AI_SAMPCLKSRC_PFI8	1	外部时钟源 PFI8	
AI_SAMPCLKSRC_PFI9	1	外部时钟源 PFI9	
AI_SAMPCLKSRC_PFI10	1	外部时钟源 PFI10	
AI_SAMPCLKSRC_PFI11	1	外部时钟源 PFI11	
AI_SAMPCLKSRC_PFI12	1	外部时钟源 PFI12	
AI_SAMPCLKSRC_PFI13	1	外部时钟源 PFI13	
AI_SAMPCLKSRC_PFI14	1	外部时钟源 PFI14	
AI_SAMPCLKSRC_PFI15	1	外部时钟源 PFI15	

nExtSampClkEdge

外部采样时钟边沿, 0=下降沿, 1=上升沿(仅当 nSampClkSource 外部采样时钟源时有效)

nReserved0-7

保留字段(未作定义)，可强制赋 0。

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

4.2 AI_STATUS（AI 工作状态信息结构）

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

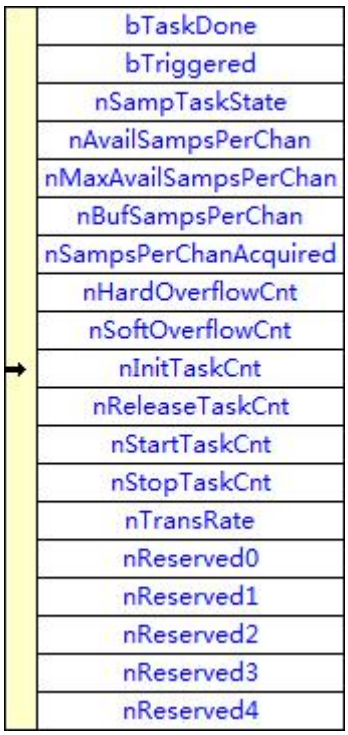
```
typedef struct _AI_STATUS
{
    U32 bTaskDone;
    U32 bTriggered;
```

```
    U32 nTaskState;
    U32 nAvailSampsPerChan;
    U32 nMaxAvailSampsPerChan;
    U32 nBufSampsPerChan;
    U64 nSampsPerChanAcquired;

    U32 nHardOverflowCnt;
    U32 nSoftOverflowCnt;
    U32 nInitTaskCnt;
    U32 nReleaseTaskCnt;
    U32 nStartTaskCnt;
    U32 nStopTaskCnt;
    U32 nTransRate;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
    U32 nReserved3;
    U32 nReserved4;
} AI_STATUS, *PAI_STATUS;
```

Visual Basic：

```
Private Type    AI_STATUS
    bTaskDone      As Long
    bTriggered     As Long
    nTaskState     As Long
    nAvailSampsPerChan     As Long
    nMaxAvailSampsPerChan As Long
    nBufSampsPerChan As Long
    nSampsPerChanAcquired As Long
    nSampsPerChanAcquiredH32 As Long
    nHardOverflowCnt     As Long
    nSoftOverflowCnt     As Long
    nInitTaskCnt As Long
    nReleaseTaskCnt As Long
    nStartTaskCnt As Long
    nStopTaskCnt As Long
    nTransRate     As Long
    nReserved0              As Long
    nReserved1              As Long
    nReserved2              As Long
    nReserved3              As Long
    nReserved4              As Long
End Type
```

LabVIEW:



请参考 USB2861.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。

bTaskDone

AI 采集任务完成标志(Task Done)。=TRUE:表示采集任务已结束； =FALSE:表示采集任务正在进行中。在设备上电之初或执行 AI_StopTask()函数后其值为 TRUE，当执行 AI_StartTask()函数后为 FALSE。在有限点采集任务中，如果达到触发条件和采样点数后，任务会自动停止，此标志会被自动置成 TRUE。在连续采集任务中，只有调用 AI_StopTask()函数手动停止采集任务，此标志才会被置成 TRUE。

bTriggered

AI 触发标志。=TRUE:表示已被触发, =FALSE:表示未被触发或等待触发。在设备上电之初或当执行 AI_StartTask()后其值为 FALSE。在被正常触发后自动变为 TRUE。执行 AI_StopTask()后其值不变。

nTaskState

任务状态(Task State)，当等于 1 时表示正常，其它值表示有异常情况。

nAvailSampsPerChan

有效点数，表示采集任务缓冲中的有效数据点数（Available Samples Per Channel）。如果它小于 nReadSampsPerChan 的时候调用 AI_ReadAnalog() 或 AI_ReadBinary()时，则读数函数会自动进入超时等待睡眠状态，直至可读点数达到指定读取点数 nReadSampsPerChan 才会返回，如果等待时间超过 fTimeout 的值，也会返回 FALSE，并置超时错误状态。如果它大于 nReadSampsPerChan 的时候调

用 AI_ReadAnalog() 或 AI_ReadBinary()时，则读数函数会迅速返回指定点数的数据并返回 TRUE。在连续采样模式中，如果 nAvailSampsPerChan 的值大于或等于 nBufSampsPerChan，则意味着采样缓冲区已经发生溢出，其溢出次数可以通过 nHardOverflowCnt 和 nSoftOverflowCnt 观察到。这个状态值的监测主要针对于连续采样模式和有限点采样模式，在单点采样模式下，它总是为 0。

nMaxAvailSampsPerChan

自开始采样后，曾经出现过的最多的有效点数（Available Samples Per Channel）。比如在某一时刻 nAvailSampsPerChan=200，则该状态值就会等于 200，只要过后 nAvailSampsPerChan 永远小于 200，则该状态值就永远等于 200，除非 nAvailSampsPerChan 后来又超过 200，比如有个一次 350，则该状态值就保持在 350，依此类推。该状态值的作用是为了验证程序的整体效率而提供的。该状态值在采集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 nBufSampsPerChan（即每通道缓冲区点数），那么溢出的可能性就比较大了。如果超越了 nBufSampsPerChan，则意味着采集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。这个状态值的监测主要针对于连续采样模式，对于有限点和单点采样无监测意义。

nBufSampsPerChan

采集任务支持的每通道缓冲区点数（Samples per channel in task buffer）。表示在任务缓冲中，每通道最多可容量的数据点数。在有限点采样模式下，其每通道缓冲区点数直接由 AI 参数 AIParam.nSampsPerChan 决定。在连续采样模式下，采集任务会根据采样参数中的 nSampsPerChan，nSampChanCount 以及 nSampleRate 来决定使用缓冲区的大小，并由 nBufSampsPerChan 状态值得到其大小。单点采样模式，该状态值始终为 0

nSampsPerChanAcquired

自开始采集任务后，每通道已经采样过的点数（Samples Acquired Per Channel）。注意此状态值是 64Bit 的。

nHardOverflowCnt

硬件溢出计数（Hardware Overflow Count）。在开始采集任务后，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。如果用户重新开始采样，则该计数器自动清零。因此，该计数信息是作为分析采集应用软件设计是否高效，计算机系统是否高效提供了有利的参考信息。对于软件按需单点采样无任何意义。

nSoftOverflowCnt

软件溢出计数（Software Overflow Count）。在开始采样后，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。如果重新开始采样，则该计数器自动清零。因此，该计数器值是作为分析应用软件设计是否高效，计算机系统是否高效提供了有利的参考信息。这个计数信息主要服务于连续采样模式。对于有限点采样和单点采样没有任何意义。

nInitTaskCnt

调用 AI_InitTask()的次数，用于检测初始化采集任务与释放采集任务是否前后匹配，如该计数值始终比 nReleaseTaskCnt 大 1，则表示每调用一次 AI_InitTask()后就会相应的调用 AI_ReleastTask()一次。

nReleaseTaskCnt

调用 AI_ReleaseTask()的次数。原理同上。

nStartTaskCnt

调用 AI_StartTask()的次数。用于检测开始采集任务与停止采集任务是否前后匹配，如该计数值始终比 nStopTaskCnt 大 1，则表示每调用一次 AI_StartTask()后就会相应的调用 AI_StopTask()一次。

nStopTaskCnt

调用 AI_StopTask()的次数。原理同上。

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（它是指所有采样通道的数据传输速率）。比如设定的单个通道采样速率(fSampleRate)为 125000sps，采样通道数(nSampChanCount)为 4，则总采样速率为 500000sps（125000*4），那么正常情况下，该状态值应等于 500000 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。

nReserved0-4

保留字段(未作定义)。

相关函数：AI_GetStatus()

4.3 AI_MAIN_INFO (AI 主要信息结构体)

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

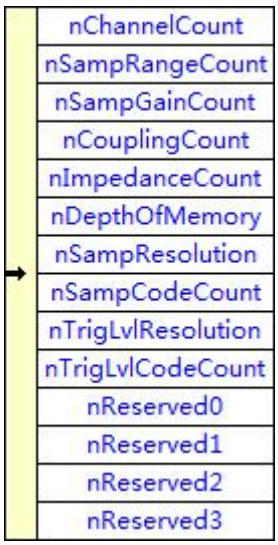
```
typedef struct _ AI_MAIN_INFO
{
    U32 nChannelCount;
    U32 nSampRangeCount;
    U32 nSampleGainCount;
    U32 nCouplingCount;
    U32 nImpedanceCount;
    U32 nDepthOfMemory;
    U32 nSampResolution;
    U32 nSampCodeCount;
    U32 nTrigLvlResolution;
    U32 nTrigLvlCodeCount;

    U32 nReserved0;
```

```
    U32 nReserved1;
    U32 nReserved2;
    U32 nReserved2;
} AI_MAIN_INFO, *PAI_MAIN_INFO;
```

Visual Basic：

```
Private Type AI_MAIN_INFO
    nChannelCount As Long
    nSampRangeCount As Long
    nSampleGainCount 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
    nReserved2 As Long
    nReserved3As Long
End Type
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

nChannelCount

物理通道数量(Channel Count)。

nSampRangeCount

采样范围挡位数量(Sample Range Count)。

nSampleGainCount

采样增益挡位数量(Sample Gain Count)。

nCouplingCount

耦合方式挡位数量(Coupling Count)。

nImpedanceCount

阻抗挡位数量(Impedance Count)。

nDepthOfMemory

各板载通道内存容量深度(Memory Depth)，单位：点数。

nSampResolution

采样分辨率(Sample Resolution)(如=8 表示 8Bit; =12 表示 12Bit; =14 表示 14Bit; =16 表示 16Bit)。

nSampCodeCount

采样编码数量(Sample Code Count)(如 256, 4096, 16384, 65536)

nTrigLvlResolution

触发电平(Trigger Level Resolution)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(Trigger Level Code Count) (如 256, 4096)

nReserved0-3

保留字段(未作定义)

相关函数：AI_GetMainInfo()

4.4 AI_VOLT_RANGE_INFO (AI 采样范围信息结构体)

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

```
typedef struct _ AI_VOLT_RANGE_INFO
{
    U32 nSampleRange;
    U32 nReserved0;
    F64 fMaxVolt;
    F64 fMinVolt;
    F64 fAmplitude;
    F64 fHalfOfAmp;
    F64 fCodeWidth;
    F64 fOffsetVolt;
    F64 fOffsetCode;
    char strDesc[16];
```

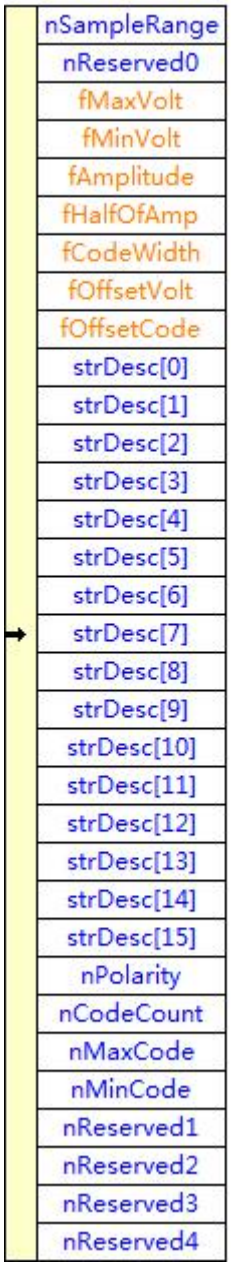
```
    U32 nPolarity;
    U32 nCodeCount;
    I32 nMaxCode;
    I32 nMinCode;

    U32 nReserved1;
    U32 nReserved2;
    U32 nReserved3;
    U32 nReserved4;
} AI_VOLT_RANGE_INFO, *P AI_VOLT_RANGE_INFO;
```

Visual Basic：

```
Private Type AI_VOLT_RANGE_INFO
    nSampleRange As Long
    nReserved0 As Long
    fMaxVolt As Double
    fMinVolt As Double
    fAmplitude As Double
    fHalfOfAmp As Double
    fCodeWidth As Double
    fOffsetVolt As Double
     fOffsetCode As Double
    strDesc(0 To 15) As Byte

    nPolarity As Long
    nCodeCount As Long
    nMaxCode As Long;
    nMinCode As Long;

    nReserved1As Long
    nReserved2As Long
    nReserved3As Long
    nReserved4As Long
End Type
```

LabVIEW:

AI_VOLT_RANGE_INFO



请参考 USB2861.lvlib 库文件及相关演示 vi

nSampleRange

当前采样范围索引号（Sample Range Index）

nReserved0

保留字段

fMaxVolt

采样范围的上限电压值(Max Voltage)，单位：伏(V)。

fMinVolt

采样范围的下限电压值(Min Voltage)，单位：伏(V)。

fAmplitude

采样范围的幅度值，单位：伏(V)。它也可以由 fMaxVolt – fMinVolt 得到。

fHalfOfAmp

幅度的二分之一(Half Of Amplitude)，单位：伏(V)。它也可以由 fAmplitude/2 得到。

fCodeWidth

编码宽度(Code Width)。如果幅度值为 20V，且分辨率为 12Bit(即总的 LSB 个数为 4096)，那么 fCodeWidth 应为 20/4096，即约等于 0.00488 伏。

fOffsetVolt

偏移电压,单位:伏(V),一般用于零偏校准(本设备无效)。

fOffsetCode

偏移码值,一般用于零偏校准,它代表的电压值等价于 fOffsetVolt(本设备无效)。

strDesc[16]

关于采样范围的字符描述信息(Description String)，如"±10V", "0-10V"等。

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	

nCodeCount

编码总数量，如比 12 位的 AI，其编码数量为 4096， 14 位的为 16384， 16 位的为 65536。

nMaxCode

采样二进制原码数据的变化最大值

nMinCode

采样二进制原码数据的变化最值值

nReserved1-4

保留字段。

相关函数： AI_GetVoltRangeInfo()

4.5 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 nRateType;

    U32 nReserved0;
    U32 nReserved1;
} 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
    nRateType As Long
    nReserved0 As Long
    nReserved1 As Long
End Type
```

LabVIEW:

请参考 USB2861.lvlib 库文件及相关演示 vi

fMaxRate

AI 最大采样率(Max Rate)，单位：点/秒(sps)。

fMinRate

AI 最小采样率(Min Rate)，单位：点/秒(sps)。

fTimerBase

时钟基准(Timer Base)，即板上使用的晶振大小，单位：赫兹(Hz)。

nDivideMode

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

nRateType

速率类型,指 fMaxRate 和 fMinRate 的类型, =0:表示为所有采样通道的总速率, =1:表示为每个采

样通道的速率

nReserved0-1

保留字段。

相关函数：AI_GetRateInfo()

4.6 AO_PARAM（AO 工作参数结构体）

4.6.1 AO_CH_PARAM(AI 通道参数结构体)

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

```
typedef struct _AO_CH_PARAM
{
    U32 bChannelEn;
    U32 nSampleRange;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
    U32 nReserved3;
}
```

bChannelEn

AO 物理通道使能(Channel Enable)，取值范围[0, 3]，即 AO0-AO3，只有使能的通道才能输出波形。如 bChannelEn[0]=TRUE, bChannelEn[1]=FALSE 时，则表示 AO0 允许输出，AO1 禁止输出。

nSampleRange

AO 采样范围(Sample Range)，取值范围为[0, 0]，定义如下表：

常量名	常量值	采样范围
AO_SAMPRANGE_N10_P10V	0	±10V
AO_SAMPRANGE_N10_P5V	1	±5V

nReserved0-3

保留字段

4.1.2 AO_START_TRIG (AI 开始触发参数结构体)

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

```
typedef struct _AO_START_TRIG
{
    U32 nTriggerType;
    U32 nTriggerSource;
    U32 nTriggerDir;
    F32 fReserved0;
```

```
    F32 fReserved1;
    U32 nTriggerSens;
    U32 nDelaySamps;

    U32 nReserved2;
    U32 nReserved3;
    U32 nReserved4;
} AO_START_TRIG, *PAO_START_TRIG;
```

Visual Basic：

```
Private Type AO_START_TRIG
    nTriggerType As Long
    nTriggerSource As Long
    nTriggerDir As Long

    fReserved0 As Single;
    fReserved1 As Single;
    nTriggerSens As Double
    nDelaySamps As Long

    nReserved2 As Long
    nReserved3 As Long
    nReserved4 As Long
End Type
```

nTriggerType

触发类型

常量名	常量值	功能定义
AO_START_TRIGTYPE_NONE	0	无触发(等同于软件强制触发)
AO_START_TRIGTYPE_ANALOG_EDGE	1	模拟边沿触发类型
AO_START_TRIGTYPE_ANALOG_WIN	2	模拟窗触发类型
AO_START_TRIGTYPE_DIGIT_EDGE	3	数字边沿触发类型
AO_START_TRIGTYPE_DIGIT_PATTERN		数字模式触发类型(保留,未提供)

nTriggerDir

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
AO_TRIGDIR_FALLING	0	下降沿触发	默认值
AO_TRIGDIR_RISING	1	上升沿触发	
AO_TRIGDIR_CHANGING	2	变化触发(上或下边沿触发)	

nTriggerSens

AI 触发灵敏度（Trigger Sense)，单位：微秒(uS)。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。

nDelaySamps

触发延迟点数（Delay Samples）。取值范围 32 位有效[0, 4294967295]。其值等于 0 时为后触发（Post Trigger）；其值大于 0 时为正延迟触发(Delay Trigger)。就是在开始触发产生时，再延迟若干个采样点后再记录数据，其延迟的点数就是由 nDelaySamps 指定，而单个点的时间周期由 nSampleRate 指定的每通道采样速率决定的。比如 nDelaySamps=100，nSampleRate=1000Hz（即每采样点周期为 1 毫秒），则意味着在开始生成任务后，当发生开始触发时再等待 100 毫秒（1 毫秒*100）才实际进入数据采样与记录阶段。在有些应用中，用户关注的焦点信号是在触发事件之后的一段时间才发生，那么这个功能就是满足此种应用的。

4.1.3 AO_PAUSE_TRIG (AI 暂停触发参数结构体)

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

```
typedef struct _AO_START_TRIG
{
    U32 nTriggerType;
    U32 nTriggerSource;
    U32 nTriggerDir;
    F32 fReserved0;
    F32 fReserved1;
    U32 nTriggerSens;

    U32 nReserved2;
    U32 nReserved3;
} AO_ START_TRIG, *PAO_ START_TRIG;
```

Visual Basic：

```
Private Type AO_PAUSE_TRIG
    nTriggerType As Long
    nTriggerSource As Long
    nTriggerDir As Long
    fReserved0 As Single;
    fReserved1 As Single;
    nTriggerSens As Double

    nReserved2 As Long
    nReserved3 As Long
End Type
```

nTriggerType

触发类型

常量名	常量值	功能定义
AO_PAUSE_TRIGTYPE_NONE	0	无触发(等同于软件强制触发)
AO_PAUSE _TRIGTYPE_ANALOG_EDGE	1	模拟边沿触发类型
AO_PAUSE _TRIGTYPE_ANALOG_WIN	2	模拟窗触发类型
AO_PAUSE _TRIGTYPE_DIGIT_EDGE	3	数字边沿触发类型
AO_PAUSE _TRIGTYPE_DIGIT_PATTERN		数字模式触发类型(保留,未提供)

nTriggerDir

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
AO_TRIGDIR_FALLING	0	下降沿触发	默认值
AO_TRIGDIR_RISING	1	上升沿触发	
AO_TRIGDIR_CHANGING	2	变化触发(上或下边沿触发)	

nTriggerSens

AI 触发灵敏度（Trigger Sense)，单位：微秒(uS)。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。

4.6.2 AO_PARAM(AI 工作参数结构体）

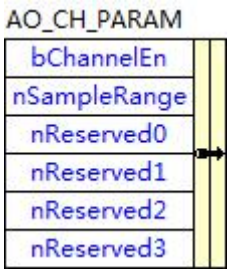
```
typedef struct _AO_PARAM
{
    U32 nSampChanCount;
    AO_CH_PARAM CHParam[4];

    U32 nSampleMode;
    U32 nSampsPerChan;
    F64 fSampleRate;
    U32 nClockSource;
    U32 bClockOutput;
    U32 bRegenModeEn
    U32 nReserved0;

    U32 bDTriggerEn;
    U32 nDTriggerDir;
    U32 bATriggerEn;
    U32 nATriggerDir;
    F32 fTriggerLevel;
```

```
    U32 nTriggerSens;
    I32 nDelaySamps;
    U32 nReserved1;

    U32 nReserved2;
    U32 nReserved3;
    U32 nReserved4;
    U32 nReserved5;
} AO_PARAM, *PAO_PARAM;
```

Visual Basic：

```
Private Type AO_PARAM
    nSampChanCount As Long
    CHParam(0 to 3) As AO_CH_PARAM

    nSampleMode As Long
    nSampsPerChan As Long
    fSampleRate As Long
    nSampClkSource As Long
    nExtSampClkEdge As Long
    bRegenModeEn As Long
    nReserved0 As Long

    AO_START_TRIG StartTrig;
    AO_PAUSE_TRIG PauseTrig;

    nReserved1 As Long
    nReserved2 As Long
    nReserved3 As Long
    nReserved4 As Long
End Type
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi

nSampChanCount

采样通道数，本参数为只读，只能由 AO_InitTask()函数返回实际的采样通道数量。

CHParam[4]

通道组(Channel Parameter)，共 4 个单元，分别控制 4 个采样通道的选择。每个单元决定要采集的物理通道号，采样范围等工作参数。

nSampleMode

AI 采样模式(Sample Mode)，取值[0, 1]，具体定义见下表：

常量名	常量值	功能定义	备注
AO_SAMPMODE_ONE_DEMAND	0	软件按需单点采样	
AO_SAMPMODE_ONE_HWTIMED	1	硬件定时单点采样	本设备不支持
AO_SAMPMODE_FINITE	2	有限点采样	
AO_SAMPMODE_CONTINUOUS	3	连续采样	

软件按需单点采样模式：就是调用 AO_StartTask()后，AO 任务只是就绪，但并不实际采样数据，而要等到每次软件调用 AO_WriteAnalog()或 AO_WriteBinary()函数时任务才开始实际采样，且每个通道仅采样一个点的数据，并以最快的速度返回，此时该点数据立即输出到设备上。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，比如应用在对时间边界没有严格要求的 PID，PLC 等快速伺服闭环控制系统。不支持触发和外时钟。

硬件定时单点采样模式：在调用 AO_StartTask()后，AO 生成任务就会按照 AOParam.fSampleRate 设定的速率定时地，不断的为每个通道采样单点数据，每次调用 AO_WriteAnalog()或 AO_WriteBinary()时也为每个通道仅写入一个点的数据到设备缓冲区中，且以最快的速度返回，此时该数据并没有实际输出，而是要等到下一个采样时钟边沿上才将该点数据实际输出到设备上。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，且时间有严格要求的确定的应用中，比如应用在对时间边界有严格要求的 PID，PLC 等快速伺服闭环控制系统。不支持触发和外时钟(但本设备不支持)。

有限点采样模式：按照设定好的采样速率和触发条件，触发模式等参数进行定长时间的、限制点数的、连续的、等间隔的波形数据采集。在开始生成任务后，达到触发条件并采样完成指定点数的数据后，生成任务就会自动停止。这个功能主要是应用在频繁捕捉触发事件、有点数限制和时间长度限制、尽可能还原外界信号的场合。

连续采样模式：按照设定好的采样速率和触发条件，触发模式等参数进行长时间的、不限点数的、连续的、等间隔的波形数据采集，在开始生成任务后，只要不软件停止生成任务，其生成任务是永远不会终止的。这个功能主要是应用在不限制点数和时间长度的，且不丢点的，尽可能还原外界信号的场合。

nSampsPerChan

AO 每通道待读取点数(Samples Per Channel)。

单点采样模式下，该参数无意义；

有限点采样模式下，该参数表示每通道采样点数。取值范围为[2，1024*1024*16]样点，最大取值还要受制于系统可用内存，采样通道数等；

连续采样模式下，决定着触发采样事件 hSampEvent 时的点数条件。比如指定该参数值为 1024 点，则每采样到不小于 1024 点时就会触发采样事件 hSampEvent，它也决定着每次调用 AO_WriteAnalog()或 AO_WriteBinary()时能最快返回的点数边界（请注意：是不小于 nSampsPerChan，意思是不可能正好等于 nSampsPerChan 时就能得到事件通知，而是通常在超过 nSampsPerChan 时才能得到事件通知）。即该参数的取值大小决定着每两次读到数据的时间间隔，也就是实时响应度。点

数越小，实时响应就越高，反之越低。但不能为了一味的追求实时响应度就将该参数的值设得很小，这个还要看采样速率的高低，如果采样速率很高，而 nSampsPerChan 的值很小，则可能造成任务负担过重而发生缓冲区溢出，以致造成丢点现象的发生。建议实时响应度不低于 20 个毫秒是比较合适的。比如每通道采样速率为 100Ksps，即 10 微秒一个点，则 nSampsPerChan 不小于 2000 个点是比较合适的。该参数的取值范围为[2, 1024*1024]，具体还要受制于系统可用内存和采样通道数。

fSampleRate

AO 采样速率(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，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 fSampleRate 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最高采样率 10000sps 决定。

nSampClkSource

时钟源(Sample Clock Source)。该参数的取值范围为[0，1]，具体定义如下表：

常量名	常量值	功能定义	备注
AO_SAMPCLKSRC_LOCAL	0	本地时钟源(板载内部时钟源)	
AO_SAMPCLKSRC_PFI0	1	外部时钟源 PFI0	
AO_SAMPCLKSRC_PFI1	1	外部时钟源 PFI1	
AO_SAMPCLKSRC_PFI2	1	外部时钟源 PFI2	
AO_SAMPCLKSRC_PFI3	1	外部时钟源 PFI3	
AO_SAMPCLKSRC_PFI4	1	外部时钟源 PFI4	
AO_SAMPCLKSRC_PFI5	1	外部时钟源 PFI5	
AO_SAMPCLKSRC_PFI6	1	外部时钟源 PFI6	
AO_SAMPCLKSRC_PFI7	1	外部时钟源 PFI7	
AO_SAMPCLKSRC_PFI8	1	外部时钟源 PFI8	
AO_SAMPCLKSRC_PFI9	1	外部时钟源 PFI9	
AO_SAMPCLKSRC_PFI10	1	外部时钟源 PFI10	
AO_SAMPCLKSRC_PFI11	1	外部时钟源 PFI11	
AO_SAMPCLKSRC_PFI12	1	外部时钟源 PFI12	
AO_SAMPCLKSRC_PFI13	1	外部时钟源 PFI13	
AO_SAMPCLKSRC_PFI14	1	外部时钟源 PFI14	
AO_SAMPCLKSRC_PFI15	1	外部时钟源 PFI15	

nExtSampClkEdge

外部采样时钟边沿, 0=下降沿, 1=上升沿(仅当 nSampClkSource 外部采样时钟源时有效)

bRegenModeEn

AO 重生成模式允许(Regeneration Mode Enable)，=TRUE 表示允许，=FALSE 表示禁止。重生成指超过一次生成相同的波形数据。通过该参数可配置生成任务允许或禁止重生成模式。默认情况下，

采样时钟定时允许重生成模式。当重生成被禁止时，新数据必须连续写入设备。

当重生成模式被允许时，只须一次传递波形数据段至生成任务中，开始任务后，数据从任务缓冲区中连续重复生成波形。开始任务后试图写入新数据至任务缓冲区将会返回一个错误。另外，任务缓冲区必须能容纳开始任务前写入的波形数据段。如果输出的波形数据在任务开始前就已经完全确定，在输出过程中不再临时改变，那么这种方式就是最佳选择（而且流程更易于控制、方法更简单）。

当重生成模式被禁止时，将数据连续写入任务缓冲区，以供任务实时连续输出至设备，即使这些数据没有变化。如在开始任务后写入数据至设备，新数据会一直反复生成直到写入更多的新数据。如果输出的波形数据在任务开始前无法完全确定，而是在开始后才能根据某些条件的变化来临时决定输出数据内容，那么这种方式就是最佳选择。

nReserved0-4

保留字段(未作定义)，可强制赋 0

相关函数： DEV_Create() AO_InitTask() AO_LoadParam() AO_SaveParam() AO_ResetParam DEV_Release()

4.7 AO_STATUS（AO 工作状态信息结构）

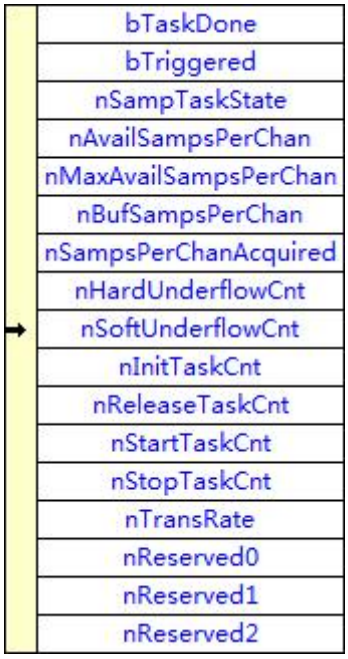
Visual C++ / C++Builder / LabWindows/CVI:

```
typedef struct _AO_STATUS
{
    U32 bTaskDone;
    U32 bTriggered;

    U32 nTaskState;
    U32 nAvailSampsPerChan;
    U32 nMaxAvailSampsPerChan;
    U32 nBufSampsPerChan;
    U64 nSampsPerChanAcquired;

    U32 nHardOverflowCnt;
    U32 nSoftOverflowCnt;
    U32 nInitTaskCnt;
    U32 nReleaseTaskCnt;
    U32 nStartTaskCnt;
    U32 nStopTaskCnt;
    U32 nTransRate;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
} AO_STATUS, *PAO_STATUS;
```

Visual Basic：

```
Private Type    AO_STATUS
        bTaskDone      As Long
        bTriggered     As Long

        nTaskState     As Long
        nAvailSampsPerChan     As Long
        nMaxReadableSegs As Long
        nBufSampsPerChan As Long
        nSampsPerChanAcquired As Long
        nSampsPerChanAcquiredH32 As Long

        nHardOverflowCnt     As Long
        nSoftOverflowCnt     As Long
        nInitTaskCnt As Long
        nReleaseTaskCnt As Long
        nStartTaskCnt As Long
        nStopTaskCnt As Long
        nTransRate     As Long

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

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

此结构体主要用于查询 AO 的工作状态信息，AO_GetStatus()函数使用此结构体来实时取得 AO

的工作状态信息，以便同步数据采样和处理过程。

bTaskDone

AO 生成任务完成标志(Task Done)。=TRUE:表示生成任务已结束, =FALSE:表示生成任务正在进行中。在设备上电之初或执行 AO_StopTask()函数后其值为 TRUE，当执行 AO_StartTask()函数后为 FALSE。在有限点生成任务中，如果达到触发条件和采样点数后，任务会自动停止，此标志会被自动置成 TRUE。在连续生成任务中，只有调用 AO_StopTask()函数手动停止生成任务，此标志才会被置成 TRUE。

bTriggered

AO 触发标志。=TRUE:表示已被触发, =FALSE:表示未被触发或等待触发。在设备上电之初或当执行 AO_StartTask()后其值为 FALSE。在被正常触发后自动变为 TRUE。执行 AO_StopTask()后其值不变。

nTaskState

任务状态(Task State), 若等于 1 表示正常, 其它值表示有异常情况。

nAvailSampsPerChan

每通道有效点数，表示生成任务缓冲中的可写入数据点数（Available Samples Per Channel）。在初始化生成任务后，nAvailSampsPerChan 会被复位至 nBufSampsPerChan 的值。每写入一个采样数据则 nAvailSampsPerChan 会自动减 1，直至 0 值。每输出一个数据到设备时该状态值会自动加 1，直至 nBufSampsPerChan。在写入数据前应判断此值，如果它小于或等于参数 nWriteSampsPerChan 的时就调用 AO_WriteAnalog()或 AO_WriteBinary()时，则写数据函数会自动进入超时等待睡眠状态，直至可写点数达到指定写入点数 nWriteSampsPerChan 才会返回，如果等待时间超过 fTimeout 的值，也会返回 FALSE，并置超时错误状态。如果 nAvailSampsPerChan 大于 nWriteSampsPerChan 的时候调用 AO_WriteAnalog()或 AO_WriteBinary()时，则写数据函数会迅速写入指定点数的数据并返回 TRUE。在连续采样和非重生成模式中，如果 nAvailSampsPerChan 的值等于 0，表示写入的数据点数刚好填满任务缓冲区，不能再写入数据，否则存在着缓冲区溢出的风险（即造成断波）。如果 nAvailSampsPerChan 的值大于或等于 nBufSampsPerChan，表示任务缓冲区已经被腾空，存在着数据下溢的风险（即造成断波），解决的办法是应迅速写入后续数据，保证缓冲区不被任务腾空。其下溢次数可以通过 nHardOverflowCnt 和 nSoftOverflowCnt 观察到。这个状态值的监测主要针对于连续采样模式和有限点采样模式，在单点采样模式下，它总是为 0。

nMaxAvailSampsPerChan

自开始采样后，曾经出现过的最大的可有效点数（Available Samples Per Channel）。比如在某一时刻 nAvailSampsPerChan=200，则该 nMaxAvailSampsPerChan 就会等于 200，只要过后 nAvailSampsPerChan 永远小于 200，则该状态值就永远等于 200，除非 nAvailSampsPerChan 后来又超过 200，比如有个一次 350，则该状态值就保持在 350，依此类推。它的值越小反映了任务缓冲区越接近满状态（因为没有更多的空闲位置可写入新数据），此时越不易发生输出缓冲下溢而断波的可能。反之越大，则反映了任务缓冲区越接近于空状态（因为需要更多的空闲位置需要及时写入新数据），此时越容易发生输出缓冲下溢而断波的可能。因此该状态值的作用是为了验证程序的整体效率而提供的。该状态值在生成任务长期运行过程中越小越好，则表示应用程序的写入效率和处理效率都很高，设备采样下溢丢点的可能性几乎为 0。如果此状态值等于或大于了 nBufSampsPerChan，则意味着生成任务已经发生过下溢了，其溢出次数则可以通过 nHardUnderflowCnt 和 nSoftUnderflowCnt

观察到。这个状态值的监测主要针对于连续非重生成模式，对于有限点和单点采样无监测意义。

nBufSampsPerChan

生成任务支持的每通道缓冲区点数（Samples per channel in task buffer）。表示在任务缓冲中，每通道最多可容量的数据点数。在有限点采样模式下，其每通道缓冲区点数直接由 AO 参数 AOParam.nSampsPerChan 决定。在连续采样模式下，生成任务会根据采样参数中的 nSampsPerChan，nSampChanCount 以及 nSampleRate 来决定使用缓冲区的大小，并由 nBufSampsPerChan 状态值得到其大小。对于单点采样模式，该状态值始终为 0。

nSampsPerChanAcquired

自开始生成任务后，每通道已经采样过的点数（Samples Acquired Per Channel）。注意此状态值是 64Bit 的。

nHardUnderflowCnt

硬件下溢计数（Hardware Underflow Count）。在开始生成任务后，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，如计算机系统极度繁忙或应用程序处理不当，效率不高，任务没有及时往设备中写入数据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。如果用户重新开始采样，则该计数器自动清零。因此，该计数信息是作为分析采集应用软件设计是否高效，计算机系统是否高效提供了有利的参考信息。对于软件按需单点采样无任何意义。

nSoftUnderflowCnt

软件下溢计数（Software Underflow Count）。在开始生成任务后，绝大多数情况下，设备中的软件缓存是不会下溢。但如果因为某种原因，如计算机系统极度繁忙或应用程序处理不当，效率不高，没有及时往任务缓冲区中写入新的数据则有可能会引起软件缓存下溢，则该计数器就会自动加 1，之后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。如果重新开始采样，则该计数器自动清零。因此，该计数器值是作为分析应用软件设计是否高效，计算机系统是否高效提供了有利的参考信息。这个计数信息主要服务于连续采样和非重生成模式。对于有限点采样和单点采样没有任何意义。

nInitTaskCnt

调用 AO_InitTask()的次数，用于检测初始化生成任务与释放生成任务是否前后匹配，如该计数值始终比 nReleaseTaskCnt 大 1，则表示每调用一次 AO_InitTask()后就会相应的调用 AO_ReleastTask()一次。

nReleaseTaskCnt

调用 AO_ReleaseTask()的次数。原理同上。

nStartTaskCnt

调用 AO_StartTask()的次数。用于检测开始生成任务与停止生成任务是否前后匹配，如该计数值始终比 nStopTaskCnt 大 1，则表示每调用一次 AO_StartTask()后就会相应的调用 AO_StopTask()一次。

nStopTaskCnt

调用 AO_StopTask()的次数。原理同上。

nTransRate

设备传输速率（Transfer Rate），单位：点/秒(P/S)。它反应了在 AO 采样过程中，实时传输 AO 采样数据的平均秒速度，即每秒传输了多少点的数据（它是指所有采样通道的数据传输速率）。比如设定的单个通道采样速率(fSampleRate)为 100000sps，采样通道数(nSampChanCount)为 2，则总采样速率为 200000sps（100000*2），那么正常情况下，该状态值应等于 200000 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。

nReserved0-4

保留字段(未作定义)。

相关函数：AO_GetStatus()

4.8 AO_MAIN_INFO (AO 主要信息结构体)

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

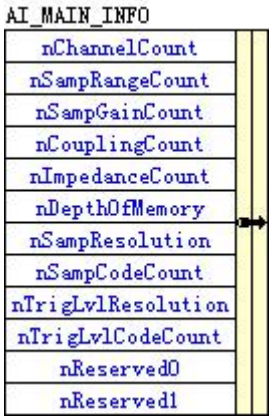
```
typedef struct _ AO_MAIN_INFO
{
    U32 nChannelCount;
    U32 nSampRangeCount;
    U32 nSampleGainCount;
    U32 nCouplingCount;
    U32 nImpedanceCount;
    U32 nDepthOfMemory;
    U32 nSampResolution;
    U32 nSampCodeCount;
    U32 nTrigLvlResolution;
    U32 nTrigLvlCodeCount;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
    U32 nReserved2;
} AO_MAIN_INFO, *PAO_MAIN_INFO;
```

Visual Basic：

```
Private Type AO_MAIN_INFO
    nChannelCount As Long
    nSampRangeCount As Long
    nSampleGainCount 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
    nReserved2 As Long
    nReserved3As Long
End Type
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

nChannelCount

物理通道数量(Channel Count)。

nSampRangeCount

采样范围挡位数量(Sample Range Count)。

nSampleGainCount

采样增益挡位数量(Sample Gain Count)。

nCouplingCount

耦合方式挡位数量(Coupling Count)。

nImpedanceCount

阻抗挡位数量(Impedance Count)。

nDepthOfMemory

各板载通道内存容量深度(Memory Depth)，单位：点数。

nSampResolution

采样分辨率(Sample Resolution)(如=8 表示 8Bit; =12 表示 12Bit; =14 表示 14Bit; =16 表示 16Bit)。

nSampCodeCount

采样编码数量(Sample Code Count)(如 256, 4096, 16384, 65536)。

nTrigLvlResolution

触发电平(Trigger Level Resolution)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(Trigger Level Code Count) (如 256, 4096)。

nReserved0-3

保留字段(未作定义)。

相关函数： AO_GetMainInfo()

4.9 AO_VOLT_RANGE_INFO (AO 采样范围信息结构体)

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

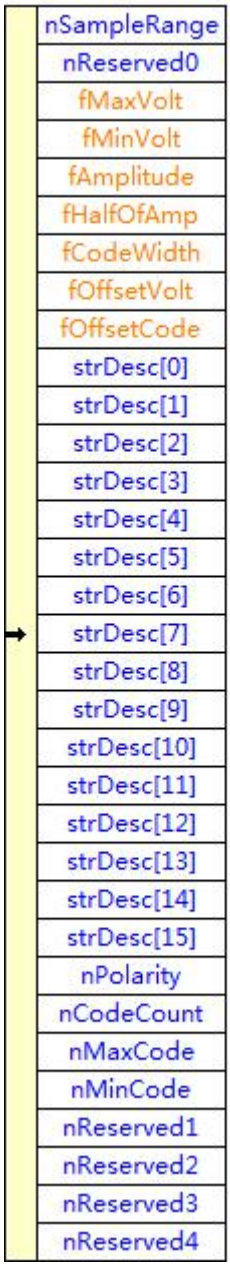
```
typedef struct _ AO_VOLT_RANGE_INFO
{
    U32 nSampleRange;
    U32 nReserved0;
    F64 fMaxVolt;
    F64 fMinVolt;
    F64 fAmplitude;
    F64 fHalfOfAmp;
    F64 fCodeWidth;
    F64 fOffsetVolt;
    F64 fOffsetCode;
    char strDesc[16];

    U32 nPolarity;
    U32 nCodeCount;
    I32 nMaxCode;
    I32 nMinCode;

    U32 nReserved1;
    U32 nReserved2;
    U32 nReserved3;
    U32 nReserved4;
} AO_VOLT_RANGE_INFO, *P AO_VOLT_RANGE_INFO;
```

Visual Basic：

```
Private Type AO_VOLT_RANGE_INFO
    nSampleRange As Long
    nReserved0 As Long
    fMaxVolt As Double
    fMinVolt As Double
    fAmplitude As Double
    fHalfOfAmp As Double
    fCodeWidth As Double
    fOffsetVolt As Double
     fOffsetCode As Double
    strDesc(0 To 15) As Byte

    nPolarity As Long
    nCodeCount As Long
    nMaxCode As Long;
    nMinCode As Long;

    nReserved1As Long
    nReserved2As Long
    nReserved3As Long
    nReserved4As Long
End Type
```

LabVIEW:

AO_VOLT_RANGE_INFO



请参考 USB2861.lvlib 库文件及相关演示 vi

nSampleRange

当前采样范围索引号（Sample Range Index）。

nReserved0

保留字段

fMaxVolt

采样范围的上限电压值(Max Voltage)，单位：伏(V)。

fMinVolt

采样范围的下限电压值(Min Voltage)，单位：伏(V)。

fAmplitude

采样范围的幅度值，单位：伏(V)。它也可以由 fMaxVolt – fMinVolt 得到。

fHalfOfAmp

幅度的二分之一(Half Of Amplitude)，单位：伏(V)。它也可以由 fAmplitude/2 得到。

fCodeWidth

编码宽度(Code Width)。如果幅度值为 20V，且分辨率为 12Bit(即总的 LSB 个数为 4096)，那么 fCodeWidth 应为 20/4096，即约等于 0.00488 伏。

fOffsetVolt

偏移电压(Offset Volt),单位:伏(V),一般用于零偏校准(暂时未用)。

fOffsetCode

偏移码值,一般用于零偏校准,它代表的电压值等价于 fOffsetVolt(本设备无效)。

strDesc[16]

关于采样范围的字符描述信息(Description String)，如"±10V", "0-10V"等。

nPolarity

AO 样范围的极性。

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

nCodeCount

编码总数量，如比 12 位的 AI，其编码数量为 4096， 14 位的为 16384， 16 位的为 65536。

nMaxCode

采样二进制原码数据的变化最大值

nMinCode

采样二进制原码数据的变化最值值

nReserved1-4

保留字段。

相关函数： AO_GetVoltRangeInfo()

4.10 AO_SAMP_RATE_INFO (AO 采样速率信息结构体)

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

typedef struct _AO_SAMP_RATE_INFO

```
{
    F64 fMaxRate;
    F64 fMinRate;
    F64 fTimerBase
    U32 nDivideMode;
    U32 nRateType;

    U32 nReserved0;
    U32 nReserved1;
} AO_SAMP_RATE_INFO, *PAO_SAMP_RATE_INFO;
```

Visual Basic：

```
Private Type AO_SAMP_RATE_INFO
    fMaxRate As Double
    fMinRate As Double
    fTimerBase As Double
    nDivideMode As Long
     nRateType As Long

    nReserved0 As Long
    nReserved1 As Long
End Type
```

LabVIEW:

请参考 USB2861.lvlib 库文件及相关演示 vi。

fMaxRate

AI 最大采样率(Max Rate)，单位：点/秒(sps)。

fMinRate

AI 最小采样率(Min Rate)，单位：点/秒(sps)。

fTimerBase

时钟基准(Timer Base)，即板上使用的晶振大小，单位：赫兹(Hz)。

nDivideMode

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

nRateType
速率类型,指 fMaxRate 和 fMinRate 的类型, =0:表示为所有采样通道的总速率, =1:表示为每个采样通道的速率。

nReserved0-1
保留字段。

相关函数： AO_GetRateInfo()

4.11 DI_PARAM (DI 数字量工作参数结构体)

4.11.1 DI_START_TRIG (DI 开始触发参数结构体)

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

```
typedef struct _DI_START_TRIG
{
    U32 nTriggerType;
    U32 nTriggerSource;
    U32 nTriggerDir;
    F32 fTriggerLevelTop;
    F32 fTriggerLevelBtm;
    U32 nTriggerSens;
    U32 nDelaySamps;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
} DI_START_TRIG, *PDI_START_TRIG;
```

Visual Basic：

```
Private Type DI_START_TRIG
    nTriggerType As Long
    nTriggerSource As Long
    nTriggerDir As Long

    fTriggerLevelTop As Long
    fTriggerLevelBtm As Long
    nTriggerSens As Double
    nDelaySamps As Long

    nReserved0 As Long
    nReserved1 As Long
    nReserved2 As Long
```

End Type

nTriggerType

触发类型

常量名	常量值	功能定义
DI_START_TRIGTYPE_NONE	0	无触发(等同于软件强制触发)
DI_START_TRIGTYPE_ANALOG_EDGE	1	模拟边沿触发类型
DI_START_TRIGTYPE_ANALOG_WIN	2	模拟窗触发类型
DI_START_TRIGTYPE_DIGIT_EDGE	3	数字边沿触发类型
DI_START_TRIGTYPE_DIGIT_PATTERN		数字模式触发类型(保留,未提供)

nTriggerDir

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
DI_TRIGDIR_FALLING	0	下降沿触发	默认值
DI_TRIGDIR_RISING	1	上升沿触发	
DI_TRIGDIR_CHANGING	2	变化触发(上或下边沿触发)	

nTriggerSens

DI 触发灵敏度（Trigger Sense)，单位：微秒(uS)。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。

nDelaySamps

触发延迟点数（Delay Samples)。取值范围 32 位有效[0, 4294967295]。其值等于 0 时为后触发（Post Trigger)；其值大于 0 时为正延迟触发(Delay Trigger)。就是在开始触发产生时，再延迟若干个采样点后再记录数据，其延迟的点数就是由 nDelaySamps 指定，而单个点的时间周期由 nSampleRate 指定的每通道采样速率决定的。比如 nDelaySamps=100，nSampleRate=1000Hz（即每采样点周期为 1 毫秒)，则意味着在开始生成任务后，当发生开始触发时再等待 100 毫秒（1 毫秒*100）才实际进入数据采样与记录阶段。在有些应用中，用户关注的焦点信号是在触发事件之后的一段时间才发生，那么这个功能就是满足此种应用的。

4.12.2 DI_PAUSE_TRIG (DI 暂停触发参数结构体)

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

```
typedef struct _DI_START_TRIG
{
    U32 nTriggerType;
    U32 nTriggerSource;
    U32 nTriggerDir;
    F32 fTriggerLevelTop;
```

```
    F32 fTriggerLevelBtm;
    U32 nTriggerSens;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
} DI_PAUSE_TRIG, *PDI_PAUSE_TRIG;
```

Visual Basic：

```
Private Type DI_PAUSE_TRIG
    nTriggerType As Long
    nTriggerSource As Long
    nTriggerDir As Long

    fTriggerLevelTop As Long
    fTriggerLevelBtm As Long
    nTriggerSens As Double

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

nTriggerType

触发类型

常量名	常量值	功能定义
DI_START_TRIGTYPE_NONE	0	无触发(等同于软件强制触发)
DI_START_TRIGTYPE_ANALOG_EDGE	1	模拟边沿触发类型
DI_START_TRIGTYPE_ANALOG_WIN	2	模拟窗触发类型
DI_START_TRIGTYPE_DIGIT_EDGE	3	数字边沿触发类型
DI_START_TRIGTYPE_DIGIT_PATTERN	4	数字模式触发类型(保留,未提供)

nTriggerDir

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
DI_TRIGDIR_FALLING	0	下降沿触发	默认值
DI_TRIGDIR_RISING	1	上升沿触发	
DI_TRIGDIR_CHANGING	2	变化触发(上或下边沿触发)	

nTriggerSens

DI 触发灵敏度（Trigger Sense)，单位：微秒(uS)。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。

4.12.3 DI_PARAM (DI 工作参数结构体)

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

```
typedef struct _DI_PARAM
{
    // 通道参数
    U32 nSampLineCount;
    U32 nReserved0;
    U8 bLineEn[8];

    // 时钟参数
    U32 nSampleMode;
    U32 nSampsPerChan;
    F64 fSampleRate;
    U32 nSampClkSource;
    U32 nExtSampClkEdge;
    U32 nReserved1;
    U32 nReserved2;

    // 触发参数
    USB2861_DI_START_TRIG StartTrig;
    USB2861_DI_PAUSE_TRIG PauseTrig;

    // 其他参数
    U32 nReserved3;          // 保留字段(暂未定义)
    U32 nReserved4;          // 保留字段(暂未定义)
    U32 nReserved5;          // 保留字段(暂未定义)
    U32 nReserved6;          // 保留字段(暂未定义)
} DI_PARAM, *PDI_PARAM;
```

Visual Basic：

```
Private Type DI_PARAM
    // 通道参数
    nSampLineCount As Long
    nReserved0 As Long
    bLineEn(0 To 7) As Byte

    // 时钟参数
    nSampleMode As Long
    nSampsPerChan As Long
    fSampleRate As Double
```

```
    nSampClkSource As Long
    nExtSampClkEdge As Long
    nReserved1 As Long
    nReserved2 As Long

    // 触发参数
    StartTrig As DI_START_TRIG
    PauseTrig As DI_PAUSE_TRIG

    // 其他参数
    nReserved3 As Long
    nReserved4 As Long
    nReserved5 As Long
    nReserved6 As Long
End Type
```

LabVIEW:

请参考 USB2861.lvlib 库文件及相关演示 vi。

nSampLineCount

采样线总数[1, 8](此参数为只读，当执行 DI_InitTask()后会返回实际的线数量,它是根据 bLineEn[]的线使能情况统计的)。

bLineEn[8]

线号使能(Line Enable),仅针对于 Port0 有效

nSampleMode

DI 采样模式(Sample Mode)，取值[0, 3]，具体定义见下表：

常量名	常量值	功能定义	备注
DI_SAMPMODE_ONE_DEMAND	0	软件按需单点采样	
DI_SAMPMODE_ONE_HWTIMED	1	硬件定时单点采样	本设备不支持
DI_SAMPMODE_FINITE	2	有限点采样	
DI_SAMPMODE_CONTINUOUS	3	连续采样	

软件按需单点采样模式：就是调用 DI_StartTask()后，DI 任务只是就绪，但并不实际采样数据，而要等到每次软件调用 DI_ReadDigitalOneU8()或 DI_ReadDigitalOneLines()函数时任务才开始实际采样，且每个通道仅采样一个点的数据，并以最快的速度返回。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，比如应用在对时间边界没有严格要求的 PID，PLC 等快速伺服闭环控制系统。不支持触发和外时钟。

硬件定时单点采样模式：在调用 DI_StartTask()后，DI 采集任务就会按照 DIParam.fSampleRate 设定的速率定时地，不断的为每个通道采样单点数据，每次调用 DI_ReadDigitalOneU8()或 DI_ReadDigitalOneLines()时也为每个通道仅返回一个点的数据，且以最快的速度返回。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

间有严格要求的确定的应用中，比如应用在对时间边界有严格要求的 PID，PLC 等快速伺服闭环控制系统。不支持触发和外时钟。

有限点采样模式：按照设定好的采样速率和触发条件，触发模式等参数进行定长时间的、限制点数的、连续的、等间隔的波形数据采集。在开始采集任务后，达到触发条件并采样完成指定点数的数据后，采集任务就会自动停止。这个功能主要是应用在频繁捕捉触发事件、有点数限制和时间长度限制、尽可能还原外界信号的场合。

连续采样模式：按照设定好的采样速率和触发条件，触发模式等参数进行长时间的、不限点数的、连续的、等间隔的波形数据采集，在开始采集任务后，只要不软件停止采集任务，其采集任务是永远不会终止的。这个功能主要是应用在不限制点数和时间长度的，且不丢点的，尽可能还原外界信号的场合。

nSampsPerChan

AI 每通道待读取点数(Samples Per Channel)。

单点采样模式：该参数无意义；

有限点采样模式：该参数表示每通道采样点数。取值范围为[2，1024*1024*16]样点，最大取值还要受制于系统可用内存，采样通道数等；

连续采样模式：决定着触发采样事件 hSampEvent 时的点数条件。比如指定该参数值为 1024 点，则每采样到不小于 1024 点时就会触发采样事件 hSampEvent，它也决定着每次调用 DI_ReadDigitalU8()时能最快返回的点数边界（请注意：是不小于 nSampsPerChan，意思是不可能正好等于 nSampsPerChan 时就能得到事件通知，而是通常在超过 nSampsPerChan 时才能得到事件通知）。即该参数的取值大小决定着每两次读到数据的时间间隔，也就是实时响应度。点数越小，实时响应就越高，反之越低。但不能为了一味的追求实时响应度就将该参数的值设得很小，这个还要看采样速率的高低，如果采样速率很高，而 nSampsPerChan 的值很小，则可能造成任务负担过重而发生缓冲区溢出，以致造成丢点现象的发生。建议实时响应度不低于 20 个毫秒是比较合适的。比如每通道采样速率为 100Ksps，即 10 微秒一个点，则 nSampsPerChan 不小于 2000 个点是比较合适的。该参数的取值范围为[2, 1024*1024]，具体还要受制于系统可用内存和采样通道数。

fSampleRate

DI 采样速率(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，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 fSampleRate 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最高采样率 10000sps 决定。

nSampClkSource

时钟源(Sample Clock Source)。该参数的取值范围为[0，1]，具体定义如下表：

常量名	常量值	功能定义	备注
DI_SAMPCLKSRC_LOCAL	0	本地时钟源(板载内部时钟源)	
DI_SAMPCLKSRC_PFI0	1	外部时钟源 PFI0	
DI_SAMPCLKSRC_PFI1	1	外部时钟源 PFI1	
DI_SAMPCLKSRC_PFI2	1	外部时钟源 PFI2	
DI_SAMPCLKSRC_PFI3	1	外部时钟源 PFI3	
DI_SAMPCLKSRC_PFI4	1	外部时钟源 PFI4	
DI_SAMPCLKSRC_PFI5	1	外部时钟源 PFI5	
DI_SAMPCLKSRC_PFI6	1	外部时钟源 PFI6	
DI_SAMPCLKSRC_PFI7	1	外部时钟源 PFI7	
DI_SAMPCLKSRC_PFI8	1	外部时钟源 PFI8	
DI_SAMPCLKSRC_PFI9	1	外部时钟源 PFI9	
DI_SAMPCLKSRC_PFI10	1	外部时钟源 PFI10	
DI_SAMPCLKSRC_PFI11	1	外部时钟源 PFI11	
DI_SAMPCLKSRC_PFI12	1	外部时钟源 PFI12	
DI_SAMPCLKSRC_PFI13	1	外部时钟源 PFI13	
DI_SAMPCLKSRC_PFI14	1	外部时钟源 PFI14	
DI_SAMPCLKSRC_PFI15	1	外部时钟源 PFI15	

nExtSampClkEdge

外部采样时钟边沿, 0=下降沿, 1=上升沿(仅当 nSampClkSource 外部采样时钟源时有效)

StartTrig

开始触发参数，详见《4.12.1 DI_START_TRIG (DI 开始触发参数结构体)》

PauseTrig

暂停触发参数，详见《4.12.2 DI_PAUSE_TRIG (DI 暂停触发参数结构体)》

nReserved0-6

保留字段(未作定义)，可强制赋 0

相关函数： DI_InitTask()

4.12 DI_STATUS（DI 工作状态信息结构）

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

```
typedef struct _DI_STATUS
{
    U32 bTaskDone;
```

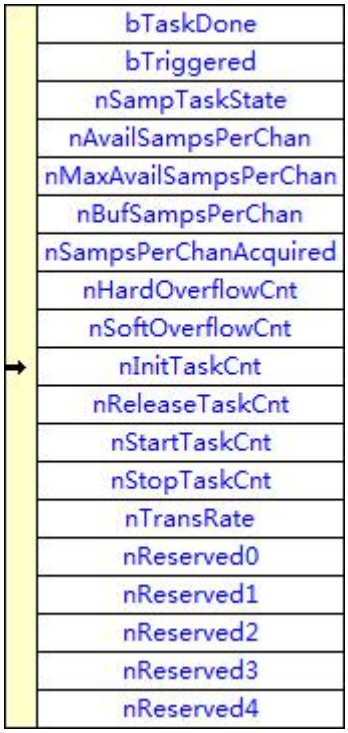
```
    U32 bTriggered;

    U32 nTaskState;
    U32 nAvailSampsPerChan;
    U32 nMaxAvailSampsPerChan;
    U32 nBufSampsPerChan;
    U64 nSampsPerChanAcquired;

    U32 nHardOverflowCnt;
    U32 nSoftOverflowCnt;
    U32 nInitTaskCnt;
    U32 nReleaseTaskCnt;
    U32 nStartTaskCnt;
    U32 nStopTaskCnt;
    U32 nTransRate;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
    U32 nReserved3;
    U32 nReserved4;
} DI_STATUS, *PDI_STATUS;
```

Visual Basic：

```
Private Type    DI_STATUS
    bTaskDone     As Long
    bTriggered    As Long
    nTaskState    As Long
    nAvailSampsPerChan    As Long
    nMaxAvailSampsPerChan As Long
    nBufSampsPerChan As Long
    nSampsPerChanAcquired As Long
    nSampsPerChanAcquiredH32 As Long
    nHardOverflowCnt    As Long
    nSoftOverflowCnt    As Long
    nInitTaskCnt As Long
    nReleaseTaskCnt As Long
    nStartTaskCnt As Long
    nStopTaskCnt As Long
    nTransRate    As Long
    nReserved0              As Long
    nReserved1              As Long
    nReserved2              As Long
    nReserved3              As Long
```

```
    nReserved4              As Long
End Type
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

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

bTaskDone

DI 采集任务完成标志(Task Done)。=TRUE:表示采集任务已结束； =FALSE:表示采集任务正在进行中。在设备上电之初或执行 DI_StopTask()函数后其值为 TRUE，当执行 DI_StartTask()函数后为 FALSE。在有限点采集任务中，如果达到触发条件和采样点数后，任务会自动停止，此标志会被自动置成 TRUE。在连续采集任务中，只有调用 DI_StopTask()函数手动停止采集任务，此标志才会被置成 TRUE。

bTriggered

DI 触发标志。=TRUE:表示已被触发, =FALSE:表示未被触发或等待触发。在设备上电之初或当执行 DI_StartTask()后其值为 FALSE。在被正常触发后自动变为 TRUE。执行 DI_StopTask()后其值不变。

nTaskState

任务状态(Task State), 当等于 1 时表示正常, 其它值表示有异常情况。

nAvailSampsPerChan

有效点数，表示采集任务缓冲中的有效数据点数（Available Samples Per Channel）。如果它小于 nReadSampsPerChan 的时候调用 DI_ReadDigitalU8()时，则读数函数会自动进入超时等待睡眠状态，

直至可读点数达到指定读取点数 nReadSampsPerChan 才会返回，如果等待时间超过 fTimeout 的值，也会返回 FALSE，并置超时错误状态。如果它大于 nReadSampsPerChan 的时候调用 DI_ReadDigitalU8() 时，则读数函数会迅速返回指定点数的数据并返回 TRUE。在连续采样模式中，如果 nAvailSampsPerChan 的值大于或等于 nBufSampsPerChan，则意味着采样缓冲区已经发生溢出，其溢出次数可以通过 nHardOverflowCnt 和 nSoftOverflowCnt 观察到。这个状态值的监测主要针对于连续采样模式和有限点采样模式，在单点采样模式下，它总是为 0。

nMaxAvailSampsPerChan

自开始采样后，曾经出现过的最多的有效点数（Available Samples Per Channel）。比如在某一时刻 nAvailSampsPerChan=200，则该状态值就会等于 200，只要过后 nAvailSampsPerChan 永远小于 200，则该状态值就永远等于 200，除非 nAvailSampsPerChan 后来又超过 200，比如有个一次 350，则该状态值就保持在 350，依此类推。该状态值的作用是为了验证程序的整体效率而提供的。该状态值在采集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 nBufSampsPerChan（即每通道缓冲区点数），那么溢出的可能性就比较大了。如果超越了 nBufSampsPerChan，则意味着采集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。这个状态值的监测主要针对于连续采样模式，对于有限点和单点采样无监测意义。

nBufSampsPerChan

采集任务支持的每通道缓冲区点数（Samples per channel in task buffer）。表示在任务缓冲中，每通道最多可容量的数据点数。在有限点采样模式下，其每通道缓冲区点数直接由 DI 参数 DIParam.nSampsPerChan 决定。在连续采样模式下，采集任务会根据采样参数中的 nSampsPerChan，nSampChanCount 以及 nSampleRate 来决定使用缓冲区的大小，并由 nBufSampsPerChan 状态值得到其大小。单点采样模式，该状态值始终为 0

nSampsPerChanAcquired

自开始采集任务后，每通道已经采样过的点数（Samples Acquired Per Channel）。注意此状态值是 64Bit 的。

nHardOverflowCnt

硬件溢出计数（Hardware Overflow Count）。在开始采集任务后，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。如果用户重新开始采样，则该计数器自动清零。因此，该计数信息是作为分析采集应用软件设计是否高效，计算机系统是否高效提供了有利的参考信息。对于软件按需单点采样无任何意义。

nSoftOverflowCnt

软件溢出计数（Software Overflow Count）。在开始采样后，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。如果重新开始采样，则该计数器自动清零。因此，该计数器值是作为分析应用软件设计是否高效，计算机系统是否高效提供了有利的参考信息。这个计数信息主要服务于连续采样模式。对于有限点采样和单点采样没有

任何意义。

nInitTaskCnt

调用 DI_InitTask()的次数，用于检测初始化采集任务与释放采集任务是否前后匹配，如该计数值始终比 nReleaseTaskCnt 大 1，则表示每调用一次 DI_InitTask()后就会相应的调用 DI_ReleastTask()一次。

nReleaseTaskCnt

调用 DI_ReleaseTask()的次数。原理同上。

nStartTaskCnt

调用 DI_StartTask()的次数。用于检测开始采集任务与停止采集任务是否前后匹配，如该计数值始终比 nStopTaskCnt 大 1，则表示每调用一次 DI_StartTask()后就会相应的调用 DI_StopTask()一次。

nStopTaskCnt

调用 DI_StopTask()的次数。原理同上。

nTransRate

设备传输速率（Transfer Rate），单位：点/秒(P/S)。它反应了在 DI 采样过程中，实时传输 DI 采样数据的平均秒速度，即每秒传输了多少点的数据（它是指所有采样通道的数据传输速率）。比如设定的单个通道采样速率(fSampleRate)为 125000sps，采样通道数(nSampChanCount)为 4，则总采样速率为 500000sps（125000*4），那么正常情况下，该状态值应等于 500000 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。

nReserved0-4

保留字段(未作定义)。

相关函数：DI_GetStatus()

4.13 DI_MAIN_INFO (DI 主要信息结构体)

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

```
typedef struct _DI_MAIN_INFO
{
    U32 nPortCount;
    U32 nLineCount[4];
    U32 nImpedanceCount;
    U32 nDepthOfMemory;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
    U32 nReserved2;
} DI_MAIN_INFO, *PDI_MAIN_INFO;
```

Visual Basic：

```
Private Type DI_MAIN_INFO
    nPortCount As Long
    nLineCount(0 To 3) As Long
    nImpedanceCount As Long
    nDepthOfMemory As Long

    nReserved0 As Long
    nReserved1As Long
    nReserved2 As Long
    nReserved3As Long
End Type
```

LabVIEW:

请参考 USB2861.lvlib 库文件及相关演示 vi。

nPortCount

端口数量(Port Count)。

nLineCount[4]

各端口线的数量(Line Count Per Port)。

nImpedanceCount

阻抗挡位数量(Impedance Count)。

nDepthOfMemory

各板载通道内存容量深度(Memory Depth)，单位：点数。

nReserved0-3

保留字段(未作定义)

相关函数：DI_GetMainInfo()

4.14 DI_SAMP_RATE_INFO (DI 采样速率信息结构体)

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

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

```
    U32 nReserved0;
    U32 nReserved1;
} DI_SAMP_RATE_INFO, *PDI_SAMP_RATE_INFO;
```

Visual Basic：

```
Private Type DI_SAMP_RATE_INFO
    fMaxRate As Double
    fMinRate As Double
    fTimerBase As Double
    nDivideMode As Long
    nRateType As Long
    nReserved0 As Long
    nReserved1 As Long
End Type
```

LabVIEW:

请参考 USB2861.lvlib 库文件及相关演示 vi

fMaxRate

DI 最大采样率(Max Rate)，单位：点/秒(sps)。

fMinRate

DI 最小采样率(Min Rate)，单位：点/秒(sps)。

fTimerBase

时钟基准(Timer Base)，即板上使用的晶振大小，单位：赫兹(Hz)。

nDivideMode

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

nRateType

速率类型,指 fMaxRate 和 fMinRate 的类型, =0:表示为所有采样通道的总速率, =1:表示为每个采样通道的速率

nReserved0-1

保留字段。

相关函数 DI_GetRateInfo()

4.15 DO_PARAM (DO 数字量工作参数结构体)

4.15.1 DO_START_TRIG (DO 开始触发参数结构体)

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

```
typedef struct _DO_START_TRIG
{
    U32 nTriggerType;
    U32 nTriggerSource;
    U32 nTriggerDir;
    F32 fTriggerLevelTop;
    F32 fTriggerLevelBtm;
    U32 nTriggerSens;
    U32 nDelaySamps;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
} DO_START_TRIG, *PDO_START_TRIG;
```

Visual Basic：

```
Private Type DO_START_TRIG
    nTriggerType As Long
    nTriggerSource As Long
    nTriggerDir As Long

    fTriggerLevelTop As Long
    fTriggerLevelBtm As Long
    nTriggerSens As Double
    nDelaySamps As Long

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

nTriggerType

触发类型

常量名	常量值	功能定义
DO_START_TRIGTYPE_NONE	0	无触发(等同于软件强制触发)
DO_START_TRIGTYPE_ANALOG_EDGE	1	模拟边沿触发类型
DO_START_TRIGTYPE_ANALOG_WIN	2	模拟窗触发类型
DO_START_TRIGTYPE_DIGIT_EDGE	3	数字边沿触发类型
DO_START_TRIGTYPE_DIGIT_PATTERN		数字模式触发类型(保留,未提供)

nTriggerDir

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
DO_TRIGDIR_FALLING	0	下降沿触发	默认值
DO_TRIGDIR_RISING	1	上升沿触发	
DO_TRIGDIR_CHANGING	2	变化触发(上或下边沿触发)	

nTriggerSens

DO 触发灵敏度（Trigger Sense)，单位：微秒(uS)。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。

nDelaySamps

触发延迟点数（Delay Samples）。取值范围 32 位有效[0, 4294967295]。其值等于 0 时为后触发（Post Trigger）；其值大于 0 时为正延迟触发(Delay Trigger)。就是在开始触发产生时，再延迟若干个采样点后再记录数据，其延迟的点数就是由 nDelaySamps 指定，而单个点的时间周期由 nSampleRate 指定的每通道采样速率决定的。比如 nDelaySamps=100，nSampleRate=1000Hz（即每采样点周期为 1 毫秒），则意味着在开始生成任务后，当发生开始触发时再等待 100 毫秒（1 毫秒*100）才实际进入数据采样与记录阶段。在有些应用中，用户关注的焦点信号是在触发事件之后的一段时间才发生，那么这个功能就是满足此种应用的。

4.15.2 DO_PAUSE_TRIG (DO 暂停触发参数结构体)

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

```
typedef struct _DO_START_TRIG
{
    U32 nTriggerType;
    U32 nTriggerSource;
    U32 nTriggerDir;
    F32 fTriggerLevelTop;
    F32 fTriggerLevelBtm;
    U32 nTriggerSens;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
} DO_PAUSE_TRIG, *PDO_PAUSE_TRIG;
```

Visual Basic：

```
Private Type DO_PAUSE_TRIG
    nTriggerType As Long
```

```
    nTriggerSource As Long
    nTriggerDir As Long

    fTriggerLevelTop As Long
    fTriggerLevelBtm As Long
    nTriggerSens As Double

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

nTriggerType

触发类型

常量名	常量值	功能定义
DO_START_TRIGTYPE_NONE	0	无触发(等同于软件强制触发)
DO_START_TRIGTYPE_ANALOG_EDGE	1	模拟边沿触发类型
DO_START_TRIGTYPE_ANALOG_WIN	2	模拟窗触发类型
DO_START_TRIGTYPE_DIGIT_EDGE	3	数字边沿触发类型
DO_START_TRIGTYPE_DIGIT_PATTERN		数字模式触发类型(保留,未提供)

nTriggerDir

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
DO_TRIGDIR_FALLING	0	下降沿触发	默认值
DO_TRIGDIR_RISING	1	上升沿触发	
DO_TRIGDIR_CHANGING	2	变化触发(上或下边沿触发)	

nTriggerSens

AO 触发灵敏度（Trigger Sense），单位：微秒(uS)。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。

4.15.3 DO_PARAM (DO 工作参数结构体)

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

```
typedef struct _DO_PARAM
{
    // 通道参数
    U32 nSampLineCount;
    U32 nReserved0;
```

```
    U8 bLineEn[8];

    // 时钟参数
    U32 nSampleMode;
    U32 nSampsPerChan;
    F64 fSampleRate;
    U32 nSampClkSource;
    U32 nExtSampClkEdge;
    U32 nReserved1;
    U32 nReserved2;

    // 触发参数
    USB2861_DO_START_TRIG StartTrig;
    USB2861_DO_PAUSE_TRIG PauseTrig;

    // 其他参数
    U32 nReserved3;          // 保留字段(暂未定义)
    U32 nReserved4;          // 保留字段(暂未定义)
    U32 nReserved5;          // 保留字段(暂未定义)
    U32 nReserved6;          // 保留字段(暂未定义)
} DO_PARAM, *PDO_PARAM;
```

Visual Basic：

```
Private Type DO_PARAM
    // 通道参数
    nSampLineCount As Long
    nReserved0 As Long
    bLineEn(0 To 7) As Byte

    // 时钟参数
    nSampleMode As Long
    nSampsPerChan As Long
    fSampleRate As Double
    nSampClkSource As Long
    nExtSampClkEdge As Long
    nReserved1 As Long
    nReserved2 As Long

    // 触发参数
    StartTrig As DO_START_TRIG
    PauseTrig As DO_PAUSE_TRIG

    // 其他参数
    nReserved3 As Long
```

```
    nReserved4 As Long
    nReserved5 As Long
    nReserved6 As Long
End Type
```

LabVIEW:

请参考 USB2861.lvlib 库文件及相关演示 vi。

nSampLineCount

采样线总数[1, 8](此参数为只读，当执行 DO_InitTask()后会返回实际的线数量,它是根据 bLineEn[]的线使能情况统计的)。

bLineEn[8]

线号使能(Line Enable),仅针对于 Port0 有效

nSampleMode

DI 采样模式(Sample Mode)，取值[0, 1]，具体定义见下表：

常量名	常量值	功能定义	备注
DO_SAMPMODE_ONE_DEMAND	0	软件按需单点采样	
DO _SAMPMODE_ONE_HWTIMED	1	硬件定时单点采样	本设备不支持
DO _SAMPMODE_FINITE	2	有限点采样	
DO _SAMPMODE_CONTINUOUS	3	连续采样	

软件按需单点采样模式：就是调用 DO_StartTask()后，AO 任务只是就绪，但并不实际采样数据，而要等到每次软件调用 AO_WriteAnalog()或 AO_WriteBinary()函数时任务才开始实际采样，且每个通道仅采样一个点的数据，并以最快的速度返回，此时该点数据立即输出到设备上。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，比如应用在对时间边界没有严格要求的 PID，PLC 等快速伺服闭环控制系统。不支持触发和外时钟。

硬件定时单点采样模式：在调用 AO_StartTask()后，AO 生成任务就会按照 AOParam.fSampleRate 设定的速率定时地，不断的为每个通道采样单点数据，每次调用 AO_WriteAnalog()或 AO_WriteBinary()时也为每个通道仅写入一个点的数据到设备缓冲区中，且以最快的速度返回，此时该数据并没有实际输出，而是要等到下一个采样时钟边沿上才将该点数据实际输出到设备上。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，且时间有严格要求的确定的应用中，比如应用在对时间边界有严格要求的 PID，PLC 等快速伺服闭环控制系统。不支持触发和外时钟(但本设备不支持)。

有限点采样模式：按照设定好的采样速率和触发条件，触发模式等参数进行定长时间的、限制点数的、连续的、等间隔的波形数据采集。在开始生成任务后，达到触发条件并采样完成指定点数的数据后，生成任务就会自动停止。这个功能主要是应用在频繁捕捉触发事件、有点数限制和时间长度限制、尽可能还原外界信号的场合。

连续采样模式：按照设定好的采样速率和触发条件，触发模式等参数进行长时间的、不限点数的、连续的、等间隔的波形数据采集，在开始生成任务后，只要不软件停止生成任务，其生成任务是永远不会终止的。这个功能主要是应用在不限制点数和时间长度的，且不丢点的，尽可能还原外界信号的场合。

nSampsPerChan

AO 每通道待读取点数(Samples Per Channel)。

单点采样模式下，该参数无意义；

有限点采样模式下，该参数表示每通道采样点数。取值范围为[2，1024*1024*16]样点，最大取值还要受制于系统可用内存，采样通道数等；

连续采样模式下，决定着触发采样事件 hSampEvent 时的点数条件。比如指定该参数值为 1024 点，则每采样到不小于 1024 点时就会触发采样事件 hSampEvent，它也决定着每次调用 AO_WriteAnalog()或 AO_WriteBinary()时能最快返回的点数边界（请注意：是不小于 nSampsPerChan，意思是不可能正好等于 nSampsPerChan 时就能得到事件通知，而是通常在超过 nSampsPerChan 时才能得到事件通知）。即该参数的取值大小决定着每两次读到数据的时间间隔，也就是实时响应度。点数越小，实时响应就越高，反之越低。但不能为了一味的追求实时响应度就将该参数的值设得很小，这个还要看采样速率的高低，如果采样速率很高，而 nSampsPerChan 的值很小，则可能造成任务负担过重而发生缓冲区溢出，以致造成丢点现象的发生。建议实时响应度不低于 20 个毫秒是比较合适的。比如每通道采样速率为 100Ksps，即 10 微秒一个点，则 nSampsPerChan 不小于 2000 个点是比较合适的。该参数的取值范围为[2, 1024*1024]，具体还要受制于系统可用内存和采样通道数。

fSampleRate

DO 采样速率(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，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 fSampleRate 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最高采样率 10000sps 决定。

nSampClkSource

时钟源(Sample Clock Source)。该参数的取值范围为[0，1]，具体定义如下表：

常量名	常量值	功能定义	备注
DO_SAMPCLKSRC_LOCAL	0	本地时钟源(板载内部时钟源)	
DO_SAMPCLKSRC_PFI0	1	外部时钟源 PFI0	
DO_SAMPCLKSRC_PFI1	1	外部时钟源 PFI1	
DO_SAMPCLKSRC_PFI2	1	外部时钟源 PFI2	
DO_SAMPCLKSRC_PFI3	1	外部时钟源 PFI3	
DO_SAMPCLKSRC_PFI4	1	外部时钟源 PFI4	
DO_SAMPCLKSRC_PFI5	1	外部时钟源 PFI5	
DO_SAMPCLKSRC_PFI6	1	外部时钟源 PFI6	
DO_SAMPCLKSRC_PFI7	1	外部时钟源 PFI7	
DO_SAMPCLKSRC_PFI8	1	外部时钟源 PFI8	
DO_SAMPCLKSRC_PFI9	1	外部时钟源 PFI9	
DO_SAMPCLKSRC_PFI10	1	外部时钟源 PFI10	
DO_SAMPCLKSRC_PFI11	1	外部时钟源 PFI11	
DO_SAMPCLKSRC_PFI12	1	外部时钟源 PFI12	
DO_SAMPCLKSRC_PFI13	1	外部时钟源 PFI13	
DO_SAMPCLKSRC_PFI14	1	外部时钟源 PFI14	
DO_SAMPCLKSRC_PFI15	1	外部时钟源 PFI15	

nExtSampClkEdge

外部采样时钟边沿, 0=下降沿, 1=上升沿(仅当 nSampClkSource 外部采样时钟源时有效)

StartTrig

开始触发参数，详见《4.15.1 DO_START_TRIG (DO 开始触发参数结构体)》

PauseTrig

暂停触发参数，详见《4.15.2 DO_PAUSE_TRIG (DO 暂停触发参数结构体)》

nReserved0-6

保留字段(未作定义)，可强制赋 0

相关函数： DO_InitTask()

4.16 DO_STATUS（DO 工作状态信息结构）

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

```
typedef struct _DO_STATUS
{
    U32 bTaskDone;
```

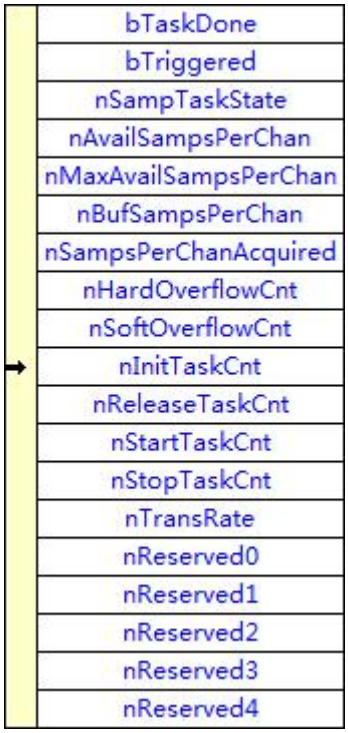
```
    U32 bTriggered;

    U32 nTaskState;
    U32 nAvailSampsPerChan;
    U32 nMaxAvailSampsPerChan;
    U32 nBufSampsPerChan;
    U64 nSampsPerChanAcquired;

    U32 nHardOverflowCnt;
    U32 nSoftOverflowCnt;
    U32 nInitTaskCnt;
    U32 nReleaseTaskCnt;
    U32 nStartTaskCnt;
    U32 nStopTaskCnt;
    U32 nTransRate;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
    U32 nReserved3;
    U32 nReserved4;
} DO_STATUS, *PDO_STATUS;
```

Visual Basic：

```
Private Type    DO_STATUS
    bTaskDone     As Long
    bTriggered    As Long
    nTaskState    As Long
    nAvailSampsPerChan    As Long
    nMaxAvailSampsPerChan As Long
    nBufSampsPerChan As Long
    nSampsPerChanAcquired As Long
    nSampsPerChanAcquiredH32 As Long
    nHardOverflowCnt    As Long
    nSoftOverflowCnt    As Long
    nInitTaskCnt As Long
    nReleaseTaskCnt As Long
    nStartTaskCnt As Long
    nStopTaskCnt As Long
    nTransRate    As Long
    nReserved0              As Long
    nReserved1              As Long
    nReserved2              As Long
    nReserved3              As Long
```

```
    nReserved4              As Long
End Type
```

LabVIEW:



请参考 USB2861.lvlib 库文件及相关演示 vi。

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

bTaskDone

DI 采集任务完成标志(Task Done)。=TRUE:表示采集任务已结束； =FALSE:表示采集任务正在进行中。在设备上电之初或执行 DO_StopTask()函数后其值为 TRUE，当执行 DO_StartTask()函数后为 FALSE。在有限点采集任务中，如果达到触发条件和采样点数后，任务会自动停止，此标志会被自动置成 TRUE。在连续采集任务中，只有调用 DO_StopTask()函数手动停止采集任务，此标志才会被置成 TRUE。

bTriggered

DI 触发标志。=TRUE:表示已被触发, =FALSE:表示未被触发或等待触发。在设备上电之初或当执行 DO_StartTask()后其值为 FALSE。在被正常触发后自动变为 TRUE。执行 DO_StopTask()后其值不变。

nTaskState

任务状态(Task State)，当等于 1 时表示正常，其它值表示有异常情况。

nAvailSampsPerChan

有效点数，表示采集任务缓冲中的有效数据点数（Available Samples Per Channel）。如果它小于 nReadSampsPerChan 的时候调用 DO_ReadDigitalU8()时，则读数函数会自动进入超时等待睡眠状态，

直至可读点数达到指定读取点数 nReadSampsPerChan 才会返回，如果等待时间超过 fTimeout 的值，也会返回 FALSE，并置超时错误状态。如果它大于 nReadSampsPerChan 的时候调用 DO_ReadDigitalU8()时，则读数函数会迅速返回指定点数的数据并返回 TRUE。在连续采样模式中，如果 nAvailSampsPerChan 的值大于或等于 nBufSampsPerChan，则意味着采样缓冲区已经发生溢出，其溢出次数可以通过 nHardOverflowCnt 和 nSoftOverflowCnt 观察到。这个状态值的监测主要针对于连续采样模式和有限点采样模式，在单点采样模式下，它总是为 0。

nMaxAvailSampsPerChan

自开始采样后，曾经出现过的最多的有效点数（Available Samples Per Channel）。比如在某一时刻 nAvailSampsPerChan=200，则该状态值就会等于 200，只要过后 nAvailSampsPerChan 永远小于 200，则该状态值就永远等于 200，除非 nAvailSampsPerChan 后来又超过 200，比如有个一次 350，则该状态值就保持在 350，依此类推。该状态值的作用是为了验证程序的整体效率而提供的。该状态值在采集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 nBufSampsPerChan（即每通道缓冲区点数），那么溢出的可能性就比较大了。如果超越了 nBufSampsPerChan，则意味着采集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。这个状态值的监测主要针对于连续采样模式，对于有限点和单点采样无监测意义。

nBufSampsPerChan

采集任务支持的每通道缓冲区点数（Samples per channel in task buffer）。表示在任务缓冲中，每通道最多可容量的数据点数。在有限点采样模式下，其每通道缓冲区点数直接由 DO 参数 DIParam.nSampsPerChan 决定。在连续采样模式下，采集任务会根据采样参数中的 nSampsPerChan，nSampChanCount 以及 nSampleRate 来决定使用缓冲区的大小，并由 nBufSampsPerChan 状态值得到其大小。单点采样模式，该状态值始终为 0

nSampsPerChanAcquired

自开始采集任务后，每通道已经采样过的点数（Samples Acquired Per Channel）。注意此状态值是 64Bit 的。

nHardOverflowCnt

硬件溢出计数（Hardware Overflow Count）。在开始采集任务后，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。如果用户重新开始采样，则该计数器自动清零。因此，该计数信息是作为分析采集应用软件设计是否高效，计算机系统是否高效提供了有利的参考信息。对于软件按需单点采样无任何意义。

nSoftOverflowCnt

软件溢出计数（Software Overflow Count）。在开始采样后，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。如果重新开始采样，则该计数器自动清零。因此，该计数器值是作为分析应用软件设计是否高效，计算机系统是否高效提供了有利的参考信息。这个计数信息主要服务于连续采样模式。对于有限点采样和单点采样没有

任何意义。

nInitTaskCnt

调用 DO_InitTask()的次数，用于检测初始化采集任务与释放采集任务是否前后匹配，如该计数值始终比 nReleaseTaskCnt 大 1，则表示每调用一次 DO_InitTask()后就会相应的调用 DO_ReleastTask()一次。

nReleaseTaskCnt

调用 DO_ReleaseTask()的次数。原理同上。

nStartTaskCnt

调用 DO_StartTask()的次数。用于检测开始采集任务与停止采集任务是否前后匹配，如该计数值始终比 nStopTaskCnt 大 1，则表示每调用一次 DO_StartTask()后就会相应的调用 DO_StopTask()一次。

nStopTaskCnt

调用 DO_StopTask()的次数。原理同上。

nTransRate

设备传输速率（Transfer Rate），单位：点/秒(P/S)。它反应了在 DO 采样过程中，实时传输 DO 采样数据的平均秒速度，即每秒传输了多少点的数据（它是指所有采样通道的数据传输速率）。比如设定的单个通道采样速率(fSampleRate)为 125000sps，采样通道数(nSampChanCount)为 4，则总采样速率为 500000sps（125000*4），那么正常情况下，该状态值应等于 500000 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。

nReserved0-4

保留字段(未作定义)。

相关函数：DO_GetStatus()

4.17 DO_MAIN_INFO (DO 主要信息结构体)

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

```
typedef struct _DO_MAIN_INFO
{
    U32 nPortCount;
    U32 nLineCount[4];
    U32 nImpedanceCount;
    U32 nDepthOfMemory;

    U32 nReserved0;
    U32 nReserved1;
    U32 nReserved2;
    U32 nReserved2;
} DO_MAIN_INFO, *PDO_MAIN_INFO;
```

Visual Basic：

```
Private Type DO_MAIN_INFO
    nPortCount As Long
    nLineCount(0 To 3) As Long
    nImpedanceCount As Long
    nDepthOfMemory As Long

    nReserved0 As Long
    nReserved1As Long
    nReserved2 As Long
    nReserved3As Long
End Type
```

LabVIEW:

请参考 USB2861.lvlib 库文件及相关演示 vi。

nPortCount

端口数量(Port Count)。

nLineCount[4]

各端口线的数量(Line Count Per Port)。

nImpedanceCount

阻抗挡位数量(Impedance Count)。

nDepthOfMemory

各板载通道内存容量深度(Memory Depth)，单位：点数。

nReserved0-3

保留字段(未作定义)

相关函数：DO_GetMainInfo()

4.18 DO_SAMP_RATE_INFO (DO 采样速率信息结构体)

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

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

```
    U32 nReserved0;
    U32 nReserved1;
} DO_SAMP_RATE_INFO, *PDO_SAMP_RATE_INFO;
```

Visual Basic：

```
Private Type DO_SAMP_RATE_INFO
    fMaxRate As Double
    fMinRate As Double
    fTimerBase As Double
    nDivideMode As Long
    nRateType As Long
    nReserved0 As Long
    nReserved1 As Long
End Type
```

LabVIEW:

请参考 USB2861.lvlib 库文件及相关演示 vi

fMaxRate

DO 最大采样率(Max Rate)，单位：点/秒(sps)。

fMinRate

DO 最小采样率(Min Rate)，单位：点/秒(sps)。

fTimerBase

时钟基准(Timer Base)，即板上使用的晶振大小，单位：赫兹(Hz)。

nDivideMode

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

nRateType

速率类型,指 fMaxRate 和 fMinRate 的类型, =0:表示为所有采样通道的总速率, =1:表示为每个采样通道的速率

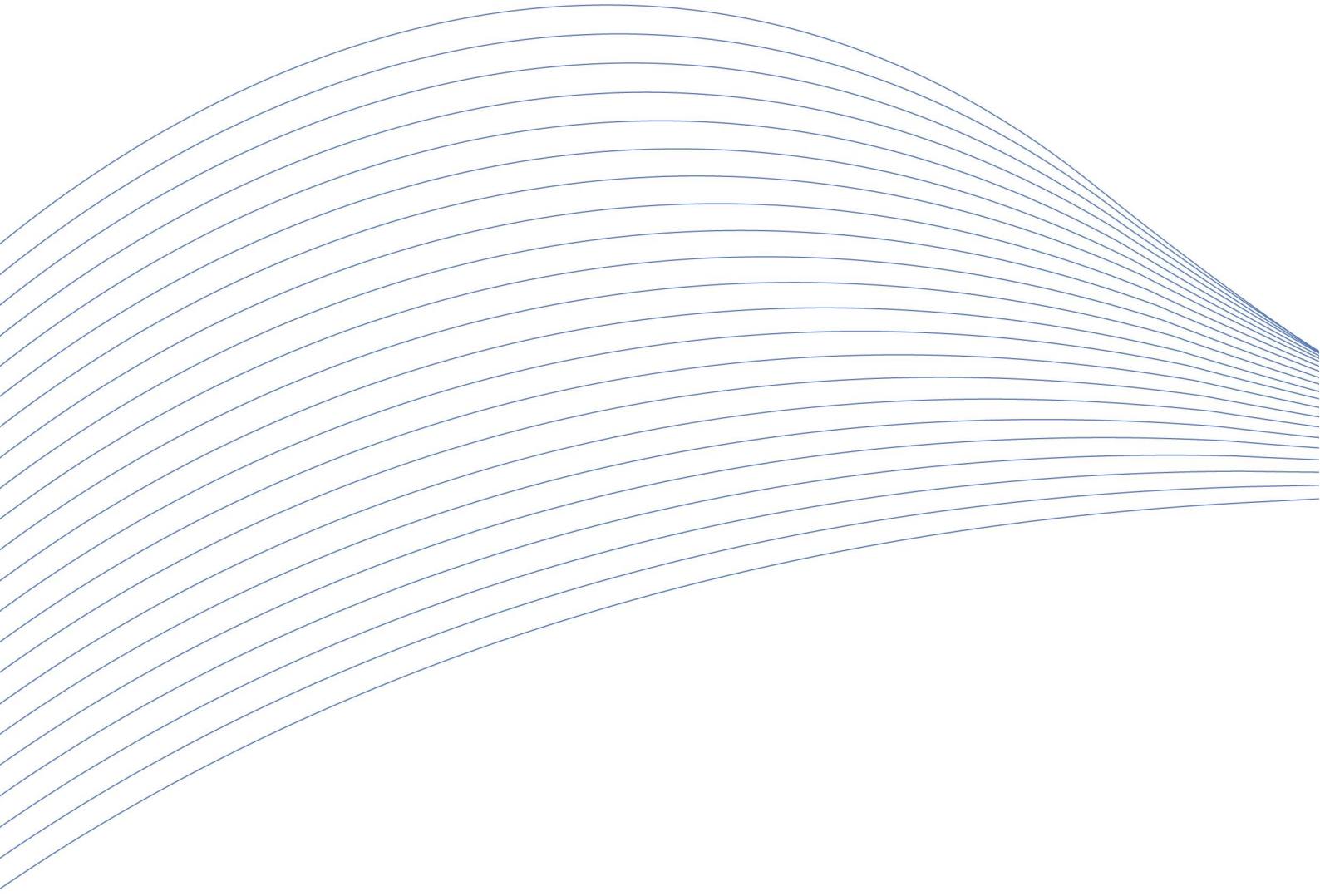
nReserved0-1

保留字段。

相关函数 DO_GetRateInfo()

5 修改历史

修改时间	版本号	修改内容
2018.10.26	V6.00.00	第一版
2019.06.26	V6.00.02	进一步完善计数器部分的内容



阿尔泰科技

服务热线：400-860-3335

网址：www.art-control.com