

# PXI8720 应变/桥采集卡

## 驱动程序使用手册

V6.00.00



# 目 录

|                                   |    |
|-----------------------------------|----|
| ■ 1 规范与约定.....                    | 1  |
| 1.1 关键字缩写命名约定.....                | 2  |
| 1.2 数据类型.....                     | 3  |
| 1.3 特别约定.....                     | 4  |
| ■ 2 使用提要 .....                    | 5  |
| 2.1 驱动函数的导入方法.....                | 5  |
| 2.2 产品二次发布.....                   | 5  |
| 2.3 管理设备.....                     | 5  |
| 2.4 AI 单点采样模式 .....               | 6  |
| 2.5 AI 有限点采样模式 .....              | 7  |
| 2.6 AI 连续采样模式 .....               | 10 |
| 2.7 用户开发所必须的函数.....               | 12 |
| ■ 3 主要功能组函数介绍.....                | 13 |
| 3.1 DEV 设备对象管理函数原型说明 .....        | 13 |
| 3.2 AI 模拟量输入函数原型说明 .....          | 16 |
| ■ 4 各种结构体描述.....                  | 24 |
| 4.1 AI_PARAM (AI 工作参数结构体) .....   | 24 |
| 4.2 AI_STATUS (AI 工作状态信息结构) ..... | 31 |
| ■ 5 高级程序界面使用说明.....               | 36 |
| 5.1 总体说明.....                     | 36 |
| 5.2 应变校准说明.....                   | 37 |
| 5.3 应变计校准.....                    | 38 |

## 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       |
| 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<br>(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++/C/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<br>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<br>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<br>用 Long 代替    | LongWord/<br>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<br>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 3.3621031431120935063E-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“PXI8720\_”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“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“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“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++   | PXI8720.h   | PXI8720.lib | C:\ART\PXI8720\Include |
| Microsoft Visual Basic | PXI8720.bas | 无           | C:\ART\PXI8720\Include |
| Borland C++ Builder    | PXI8720.h   | PXI8720.lib | C:\ART\PXI8720\Include |
| Borland Delphi         | PXI8720.pas | 无           | C:\ART\PXI8720\Include |
| NI LabVIEW             | PXI8720.vi  | 无           | C:\ART\PXI8720\Include |
| NI LabWindows/CVI      | PXI8720.h   | PXI8720.lib | C:\ART\PXI8720\Include |

注：（1）、PXI8720.h 是产品的基础的头文件，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 功能。

（2）、PXI8720RSV.h 是产品的保留头文件，为了凸现 PXI8720.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 PXI8720.dll 从 Windows\System32 中复制到安装包中；
- （2）、将 PXI8720.inf、PXI8720.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 [DeviceOpen\(\)](#) 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。最后通过 [DeviceClose\(\)](#) 函数将 hDevice 释放掉。

本设备使用前，需要校准的话，请先校准各个通道。

#### 2.4 应变校准

应变校准，主要是实现在应变计或其他惠斯通电桥感应器上进行偏移清零和分流校准，此校准建议用线程实现。

- （1）BRGSTNCAL\_ReadVal 读取应变计校准的测量应变值与增益调整值,主要用此函数进行校准，无其他流程。

#### 2.5 通道校准

通道校准，基于用户设备引入的误差进行校准。

通道校准应在采样模式为有限采样下，桥配置为全桥 I，通道使能为只对当前校准通道使能。

- （1）填写好合适的参考值之后，使用 CHCAL\_GetValue 函数得到需要的未校准值（建议用线程实现）；

(2) 提交后，得到所有行的未校准值，使用 CHCAL\_Channel 进行通道校准。

## 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\(\)](#) 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) [DeviceOpen\(\)](#) 打开设备，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\(\)](#) 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) [DeviceClose\(\)](#) 关闭设备，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。

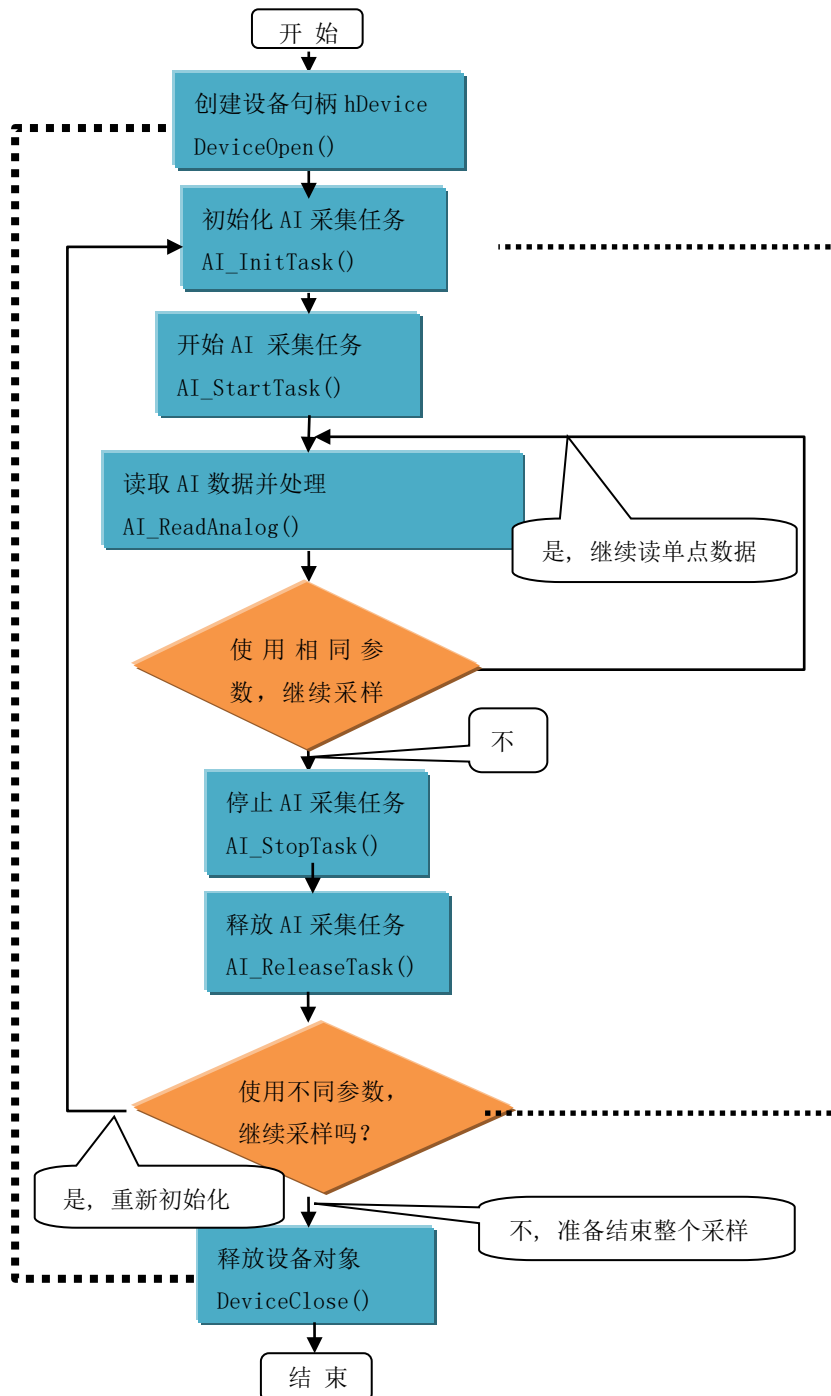


图 2-4-1 AI 实时单点采样流程图

## 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段。具体流程如图 2-5-1 所示：

- (1) [DeviceOpen\(\)](#) 创建设备句柄；
- (2) [AI\\_InitTask\(\)](#) 初始化 AI 采集任务，注意参数 nSampleMode=2；
- (3) [AI\\_StartTask\(\)](#) 开始 AI 采集任务；
- (4) [AI\\_WaitUntilTaskDone\(\)](#) 采集任务结束前等待，返回 TRUE 表示采集任务结束



- (5) [AI\\_ReadAnalog\(\)](#) 读取 AI 数据（注意指定适当的超时等待时间）；
- (6) [AI\\_StopTask\(\)](#) 停止 AI 采集任务；
- (7) [AI\\_ReleaseTask\(\)](#) 释放 AI 采集任务；
- (8) [DeviceClose\(\)](#) 关闭释放设备句柄。

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、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。

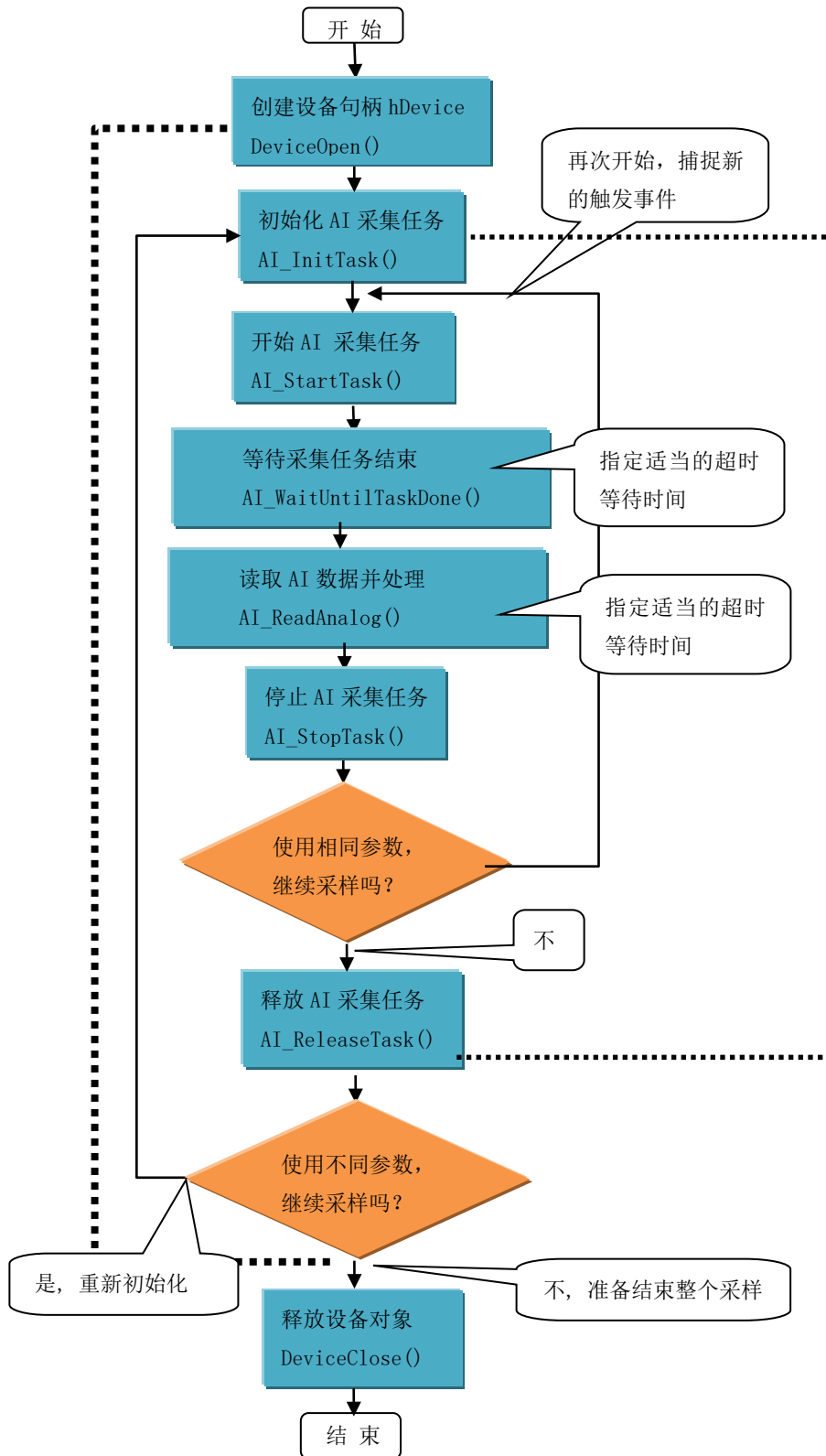


图 2-5-1 AI 有限点采样流程图例

## 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) [DeviceOpen\(\)](#) 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\(\)](#) 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) [DeviceClose\(\)](#) 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。

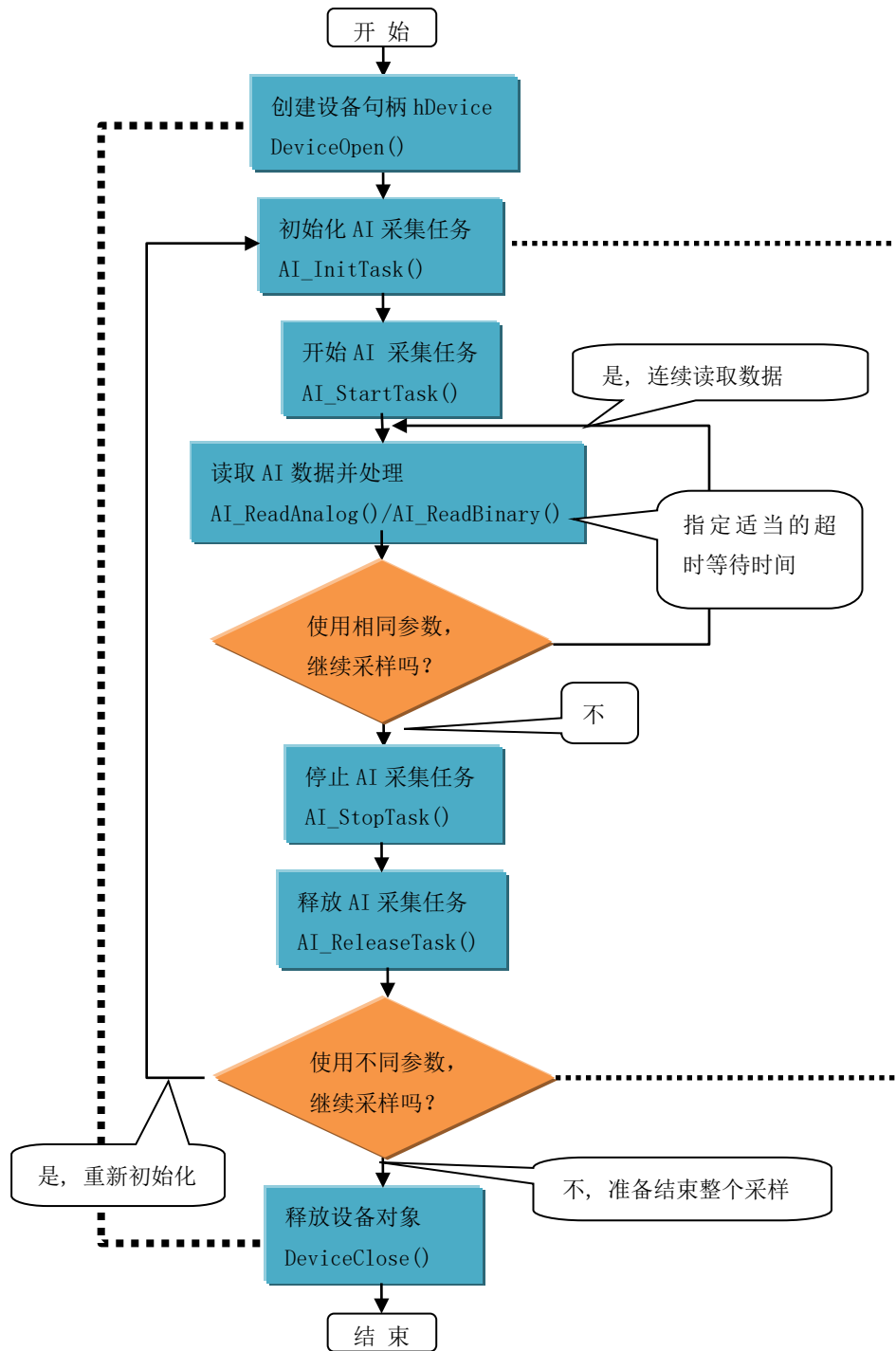


图 2-6-1 AI 连续采样流程图例



上图虚线表示对称关系。[DeviceOpen\(\)](#)和 [DeviceClose\(\)](#) 两个函数的对称关系是：最初执行一次 [DeviceOpen\(\)](#)，在结束时就须执行一次 [DeviceClose\(\)](#)，但并不是说只有 [DeviceClose\(\)](#)后 才能再次 [DeviceOpen\(\)](#)，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 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对用户都是必须的，特别是 [DeviceOpen\(\)](#)、[DeviceClose\(\)](#) 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

#### DeviceOpen()

函数原型:

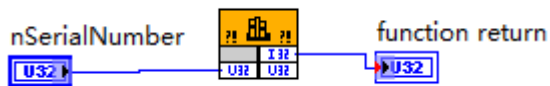
**Visual C++ / C++Builder / LabWindows/CVI:**

HANDLE DeviceOpen(U32 nSerialNumber)

**Visual Basic:**

Declare Function DEV\_Create Lib "PXI8720" (ByVal nSerialNumber As Long) As long

**LabVIEW:**



请参考 PXI8720.lvlib 库文件及相关演示 vi。

**功能:** 打开设备创建设备对象，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:**

**nSerialNumber** 入口参数，设备序号。当单采集卡使用时可以默认为 0。

**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 NULL。

**相关函数:** [DeviceOpen\(\)](#)      [DeviceList\(\)](#)      [GetDeviceSerialNumber\(\)](#)  
[DeviceClose\(\)](#)

#### DeviceList()

函数原型:

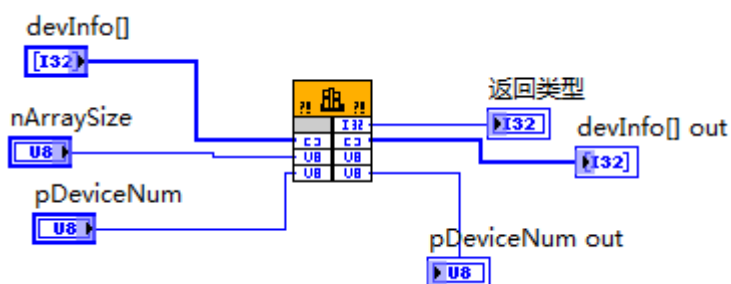
**Visual C++ / C++Builder / LabWindows/CVI:**

```
BOOL DeviceList(
    DEV_INFO devInfo[],
    U8 nArraySize,
    U8* pDeviceNum);
```

**Visual Basic:**

```
Declare Function DEV_GetCount Lib "PXI8720" (ByRef devInfo As DEV_INFO,
    ByVal nArraySize As Long,
    ByRef pDeviceNum As Long) As Long
```

**LabVIEW:**



请参考 PXI8720.lvlib 库文件及相关演示 vi。

**功能:** 枚举设备信息。

**参数:** **DevInfo[]** 出口参数 设备信息数组，由底层返回设备的信息，主要包括序列号及是否被占用。

**nArraySize** 入口参数 指定的设备信息数组的大小。

**pDeviceNum** 出口参数 返回的现已经使用的设备数量。

**返回值:** 成功返回 TRUE，失败返回 FALSE。

**相关函数:** [DeviceOpen\(\)](#)      [DeviceList\(\)](#)      [GetDeviceSerialNumber\(\)](#)  
[DeviceReset\(\)](#)      [DeviceClose\(\)](#)

**GetDeviceSerialNumber()**

函数原型:

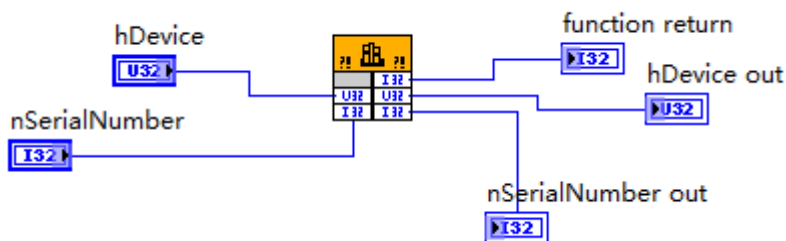
**Visual C++ / C++Builder / LabWindows/CVI:**

```
BOOL GetDeviceSerialNumber (HANDLE hDevice,
                             U32* nSerialNumber);
```

**Visual Basic:**

```
Declare Function DEV_GetCurrentIdx Lib "PXI8720" (ByVal hDevice As Long, _
                                                  ByRef nSerialNumberAs Long) As Boolean
```

**LabVIEW:**



请参考 PXI8720.lvlib 库文件及相关演示 vi。

**功能:** 得到设备序列号。

**参数:**

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

**nSerialNumber** 出口参数，取得设备的序列号，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

**相关函数：** [DeviceOpen\(\)](#)      [DeviceList\(\)](#)      [GetDeviceSerialNumber\(\)](#)  
[DeviceReset\(\)](#)      [DeviceClose\(\)](#)

### DeviceReset()

函数原型：

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

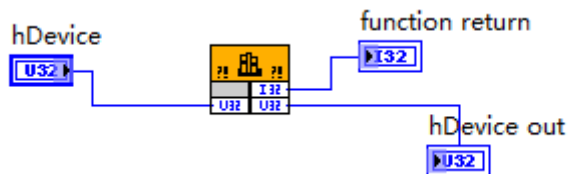
BOOL DEV\_GetSpeed(HANDLE hDevice,  
 U32\* pSpeed)

**Visual Basic:**

Declare Function DEV\_GetSpeed Lib "PXI8720" (ByVal hDevice As Long ) As Boolean

**LabVIEW:**

请参考 PXI8720.lvlib 库文件及相关演示 vi



**功能：** 复位设备。

**参数：**

**hDevice** 入口参数，设备对象句柄，由 [DeviceOpen\(\)](#)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.

**相关函数：** [DeviceOpen\(\)](#)      [DeviceList\(\)](#)      [GetDeviceSerialNumber\(\)](#)  
[DeviceReset\(\)](#)      [DeviceClose\(\)](#)

### DeviceClose()

函数原型：

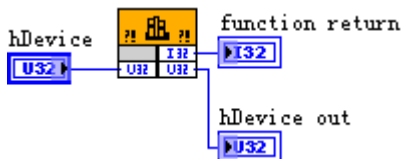
**Visual C++ / C++Builder / LabWindows/CVI:**

BOOL DeviceClose (HANDLE hDevice)

**Visual Basic:**

Declare Function DeviceClose Lib "PXI8720" (ByVal hDevice As Long ) As Boolean

**LabVIEW:**



请参考 PXI8720.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 [DeviceOpen\(\)](#)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.

相关函数：[DeviceOpen\(\)](#)      [DeviceList\(\)](#)      [GetDeviceSerialNumber\(\)](#)  
[DeviceReset\(\)](#)      [DeviceClose\(\)](#)

### 3.2 AI 模拟量输入函数原型说明

AI\_InitTask()

函数原型：

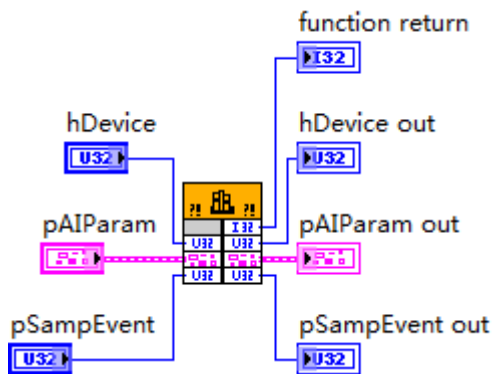
**Visual C++ / C++Builder / LabWindows/CVI:**

BOOL AI\_InitTask (HANDLE hDevice,  
 PAI\_PARAM pAIPParam,  
 HANDLE\* pSampEvent)

**Visual Basic:**

Declare Function AI\_InitTask Lib "PXI8720" (ByVal hDevice As Long, \_  
 ByRef pAIPParam As PAI\_PARAM;  
 ByRef pSampEvent As Long) As Boolean

**LabVIEW:**



请参考 PXI8720.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 [DeviceOpen\(\)](#)函数创建，该句柄指向要访问的设备。

**pAIPParam** 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 AIPParam.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\(\)](#)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\(\)](#) 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。

**相关函数：** [DeviceOpen\(\)](#)                    [AI\\_InitTask\(\)](#)                    [AI\\_StartTask\(\)](#)  
[AI\\_SendSoftTrig\(\)](#)                    [AI\\_GetStatus\(\)](#)                    [AI\\_WaitUntilTaskDone\(\)](#)  
[AI\\_ReadAnalog\(\)](#)                    [AI\\_StopTask\(\)](#)                    [AI\\_ReleaseTask\(\)](#)  
[DeviceClose\(\)](#)

### AI\_StartTask()

函数原型：

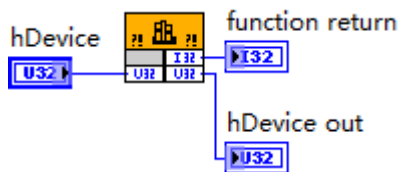
**Visual C++ / C++Builder / LabWindows/CVI:**

BOOL AI\_StartTask (HANDLE hDevice)

**Visual Basic:**

Declare Function AI\_StartTask Lib "PXI8720" (ByVal hDevice As Long) As Boolean

**LabVIEW:**



请参考 PXI8720.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 [DeviceOpen\(\)](#) 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。

**相关函数：** [DeviceOpen\(\)](#)                    [AI\\_InitTask\(\)](#)                    [AI\\_StartTask\(\)](#)  
[AI\\_SendSoftTrig\(\)](#)                    [AI\\_GetStatus\(\)](#)                    [AI\\_WaitUntilTaskDone\(\)](#)  
[AI\\_ReadAnalog\(\)](#)                    [AI\\_ReadBinary\(\)](#)                    [AI\\_StopTask\(\)](#)  
[AI\\_ReleaseTask\(\)](#)                    [DeviceClose\(\)](#)

### AI\_SendSoftTrig()

函数原型：

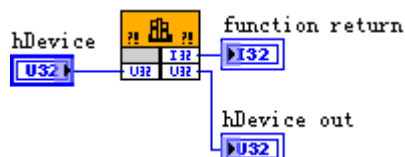
**Visual C++ / C++Builder / LabWindows/CVI:**

BOOL AI\_SendSoftTrig (HANDLE hDevice)

**Visual Basic:**

Declare Function AI\_SendSoftTrig Lib "PXI8720" (ByVal hDevice As Long) As Boolean

**LabVIEW:**



请参考 PXI8720.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。

**相关函数:** [DeviceOpen\(\)](#)                    [AI\\_InitTask\(\)](#)                    [AI\\_StartTask\(\)](#)  
[AI\\_SendSoftTrig\(\)](#)                    [AI\\_GetStatus\(\)](#)                    [AI\\_WaitUntilTaskDone\(\)](#)  
[AI\\_ReadAnalog\(\)](#)                    [AI\\_ReadBinary\(\)](#)                    [AI\\_StopTask\(\)](#)  
[AI\\_ReleaseTask\(\)](#)                    [DeviceClose\(\)](#)

## AI\_GetStatus()

函数原型:

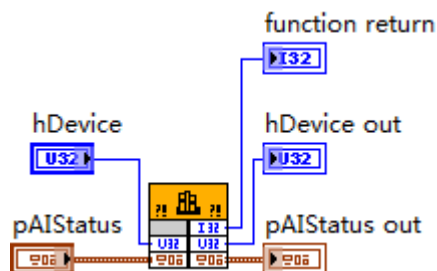
*Visual C++ / C++Builder / LabWindows/CVI:*

BOOL AI\_GetStatus (HANDLE hDevice,  
PAI\_STATUS pStatus )

*Visual Basic:*

Declare Function AI\_GetStatus Lib "PXI8720" (ByVal hDevice As Long, \_  
ByRef pAIStatus As PAI\_STATUS) As Boolean

*LabVIEW:*



请参考 PXI8720.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：**

|                                   |                                 |  |
|-----------------------------------|---------------------------------|--|
| <a href="#">DeviceOpen()</a>      | <a href="#">AI_InitTask()</a>   | <a href="#">AI_StartTask()</a>         |
| <a href="#">AI_SendSoftTrig()</a> | <a href="#">AI_GetStatus()</a>  | <a href="#">AI_WaitUntilTaskDone()</a> |
| <a href="#">AI_ReadAnalog()</a>   | <a href="#">AI_ReadBinary()</a> | <a href="#">AI_StopTask()</a>          |
| <a href="#">AI_ReleaseTask()</a>  | <a href="#">DeviceClose()</a>   |  |

### AI\_WaitUntilTaskDone()

函数原型：

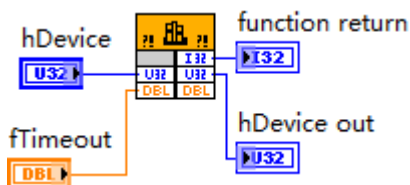
**Visual C++ / C++Builder / LabWindows/CVI:**

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

**Visual Basic:**

```
Declare Function AI_WaitUntilTaskDone Lib "PXI8720" (ByVal hDevice As Long, _
                                                    ByVal fTimeout As Double) As Boolean
```

**LabVIEW:**



请参考 PXI8720.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：**

|                                   |                                 |  |
|-----------------------------------|---------------------------------|--|
| <a href="#">DeviceOpen()</a>      | <a href="#">AI_InitTask()</a>   | <a href="#">AI_StartTask()</a>         |
| <a href="#">AI_SendSoftTrig()</a> | <a href="#">AI_GetStatus()</a>  | <a href="#">AI_WaitUntilTaskDone()</a> |
| <a href="#">AI_ReadAnalog()</a>   | <a href="#">AI_ReadBinary()</a> | <a href="#">AI_StopTask()</a>          |
| <a href="#">AI_ReleaseTask()</a>  | <a href="#">DeviceClose()</a>   |  |

### AI\_ReadAnalog()

函数原型：

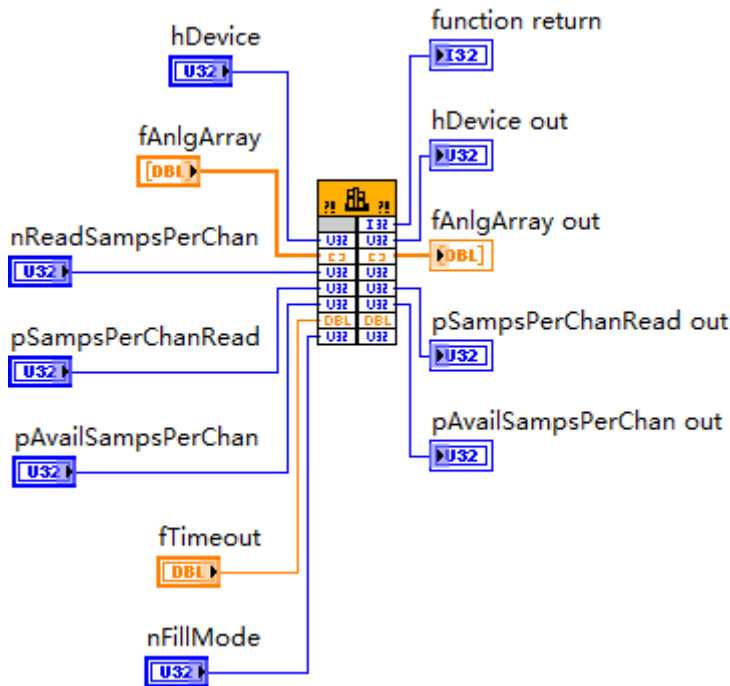
**Visual C++ / C++Builder / LabWindows/CVI:**

```
INT32 AI_ReadAnalog(HANDLE hDevice,
                   F64 fAnlgArray[],
                   U32 nReadSampsPerChan,
                   U32* pSampsPerChanRead,
                   U32* pAvailSampsPerChan,
                   F64 fTimeout,
                   U32 nFillMode);
```

**Visual Basic:**

```
Declare Function AI_ReadAnalog Lib "PXI8720" ( ByVal hDevice As Long, _
                                             ByRef fAnlgArray As Double, _
                                             ByVal nReadSampsPerChan As Long, _
                                             ByRef pSampsPerChanRead As Long, _
                                             ByRef pAvailSampsPerChan As Long, _
                                             ByVal fTimeout As Long, _
                                             ByVal nFillMode As Long) As Long
```

**LabVIEW:**



请参考 PXI8720.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 [DeviceOpen\(\)](#) 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, 本数据结果, 通过一系列算法计算得出, 数据类型为双精度浮点。各个采样通道的数据点排列顺序由 nFillMode 参数决定, 本设备已固定为 GroupByScanNumber, 它被开辟的点空间不能小于 nReadSampsPerChan\*AIPParam.nSampChanCount。

**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 AIPParam.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\*AIPParam.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。

**nFillMode** 入口参数，fAnlgArray 缓冲数据填充方式。取值范围：

| 常量名                        | 常量值 | 功能定义                               |
|----------------------------|-----|------------------------------------|
| FILLMODE_GroupByScanNumber | 1   | 按扫描序号分组填充缓冲区(交叉), <b>本设备已固定为此值</b> |
| FILLMODE_GroupByChannel    | 0   | 按通道分组填充缓冲区(不交叉)                    |

假如采样通道为 AI0、AI1、AI2、AI3，各通道采样 100 点数据时，若 nFillMode= FILLMODE\_GroupByScanNumber，则 fAnlgArray 缓冲区中的数据排列顺序为：

| 通道数据序号 | 通道顺序            |
|--------|-----------------|
| 0      | AI0 AI1 AI2 AI3 |
| 1      | AI0 AI1 AI2 AI3 |
| 2      | AI0 AI1 AI2 AI3 |
| :      | AI0 AI1 AI2 AI3 |
| 99     | AI0 AI1 AI2 AI3 |

若 nFillMode= FILLMODE\_GroupByChannel，则 fAnlgArray 缓冲区中的数据排列顺序为：

| 通道号 | 通道数据序号         |
|-----|----------------|
| AI0 | 0 1 2 3 ... 99 |
| AI1 | 0 1 2 3 ... 99 |

|     |                |
|-----|----------------|
| AI2 | 0 1 2 3 ... 99 |
| AI3 | 0 1 2 3 ... 99 |

返回值：如果函数返回结果为下表：

| 常量名                   | 常量值 | 功能定义   |
|-----------------------|-----|--------|
| PXI8720_SUCCESS       | 0   | 成功     |
| PXI8720_ERROR_FAILED  | -1  | 采集任务失败 |
| PXI8720_ERROR_TIMEOUT | -2  | 超时错误   |

相关函数：[DeviceOpen\(\)](#)                      [AI\\_InitTask\(\)](#)                      [AI\\_StartTask\(\)](#)  
[AI\\_SendSoftTrig\(\)](#)                      [AI\\_GetStatus\(\)](#)                      [AI\\_WaitUntilTaskDone\(\)](#)  
[AI\\_ReadAnalog\(\)](#)                      [AI\\_ReadBinary\(\)](#)                      [AI\\_StopTask\(\)](#)  
[AI\\_ReleaseTask\(\)](#)                      [DeviceClose\(\)](#)

### AI\_StopTask()

函数原型：

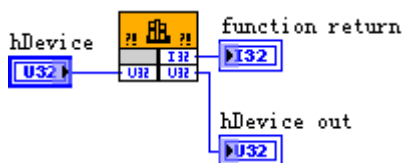
**Visual C++ / C++Builder / LabWindows/CVI:**

BOOL AI\_StopTask (HANDLE hDevice)

**Visual Basic:**

Declare Function AI\_StopTask Lib "PXI8720" (ByVal hDevice As Long)

**LabVIEW:**



请参考 PXI8720.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 [DeviceOpen\(\)](#) 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：[DeviceOpen\(\)](#)                      [AI\\_InitTask\(\)](#)                      [AI\\_StartTask\(\)](#)  
[AI\\_SendSoftTrig\(\)](#)                      [AI\\_GetStatus\(\)](#)                      [AI\\_WaitUntilTaskDone\(\)](#)  
[AI\\_ReadAnalog\(\)](#)                      [AI\\_ReadBinary\(\)](#)                      [AI\\_StopTask\(\)](#)  
[AI\\_ReleaseTask\(\)](#)                      [DeviceClose\(\)](#)

### AI\_ReleaseTask()

函数原型：

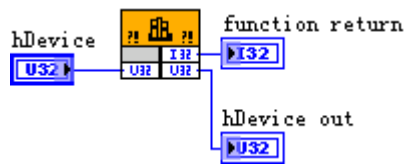
**Visual C++ / C++Builder / LabWindows/CVI:**

BOOL AI\_ReleaseTask (HANDLE hDevice)

**Visual Basic:**

Declare Function AI\_ReleaseTask Lib "PXI8720" (ByVal hDevice As Long)

**LabVIEW:**



请参考 PXI8720.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 [DeviceOpen\(\)](#) 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。

- 相关函数:**
- |                                   |                                 |  |
|-----------------------------------|---------------------------------|--|
| <a href="#">DeviceOpen()</a>      | <a href="#">AI_InitTask()</a>   | <a href="#">AI_StartTask()</a>         |
| <a href="#">AI_SendSoftTrig()</a> | <a href="#">AI_GetStatus()</a>  | <a href="#">AI_WaitUntilTaskDone()</a> |
| <a href="#">AI_ReadAnalog()</a>   | <a href="#">AI_ReadBinary()</a> | <a href="#">AI_StopTask()</a>          |
| <a href="#">AI_ReleaseTask()</a>  | <a href="#">DeviceClose()</a>   |  |



## 4 各种结构体描述

### 4.1 AI\_PARAM (AI 工作参数结构体)

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

通道参数:

```
typedef struct _PXI8720_AI_CH_PARAM // AI 通道参数结构体
{
    U32 bChannelEn;
    U32 nBridgeConfig;
    F64 fBridgeResistance;
    U32 nShuntResistance;
    U32 nExcitationVolt;
    F64 fGageFactor;
    F64 flnitBridgeVolt;
    F64 fPoissonRatio;
    F64 fWireResistance;
    U32 nMeasType;
} PXI8720_AI_CH_PARAM, *PPXI8720_AI_CH_PARAM;

typedef struct _PXI8720_AI_START_TRIG //
{
    U32 nTriggerType;
    U32 nTriggerDir;
    U32 nTriggerSens;
    U32 nDelaySamps;
} PXI8720_AI_START_TRIG, *PPXI8720_AI_START_TRIG;

typedef struct _PXI8720_AI_PAUSE_TRIG /
{
    U32 nTriggerType;
    U32 nTriggerDir;
    U32 nTriggerSens;
    U32 nReserved0;           // 保留字段(暂未定义)
} PXI8720_AI_PAUSE_TRIG, *PPXI8720_AI_PAUSE_TRIG;

typedef struct _PXI8720_AI_PARAM
{
    U32 nSampChanCount;
    U32 nReserved0;
    PXI8720_AI_CH_PARAM CHParam[4];
    U32 nSampleMode;
```

```

    U32 nSampsPerChan;
    F64 fSampleRate;

    PXI8720_AI_START_TRIG StartTrig;
    PXI8720_AI_PAUSE_TRIG PauseTrig;
    U32 nReserved1[4];
} PXI8720_AI_PARAM, *PPXI8720_AI_PARAM;

```

**Visual Basic:**

```
Private Type _PXI8720_AI_CH_PARAM
```

```

    bChannelEn As Long
    nBridgeConfig As Long
    fBridgeResistance As Double
    nShuntResistance AS Long
    nExcitationVolt    As Long

```

```

    fGageFactor AS Double
    fInitBridgeVolt As Double
    fPoissonRatio As Double
    fWireResistance As Double

```

```
    nMeasType As Long
```

```
End Type
```

```
Private Type _PXI8720_AI_START_TRIG
```

```

    nTriggerType As Long
    nTriggerDir As Long
    nTriggerSens As Long
    nDelaySamps As Long

```

```
End Type
```

```
Private Type _PXI8720_AI_PAUSE_TRIG
```

```

    nTriggerType As Long
    nTriggerDir As Long
    U32 nTriggerSens As Long
    U32 nReserved0 As Long

```

```
End Type
```

```
Private Type _PXI8720_AI_PARAM
```

```

    nSampChanCount As Long

```

nReserved0 As Long  
 CHParam(0 to 3) As PXI8720\_AI\_CH\_PARAM

nSampleMode As Long  
 nSampsPerChan As Long  
 fSampleRate As Double

StartTrig As PXI8720\_AI\_START\_TRIG  
 PauseTrig As PXI8720\_AI\_PAUSE\_TRIG

nReserved1(0 to 3) As Long

End Type

**LabVIEW:**

请参考 PXI8720.lvlib 库文件及相关演示 vi。

#### 4.1.1 AI\_CH\_PARAM(AI 通道参数结构体)

##### **bChannelEn**

AI 通道使能，=TRUE（或 1）表示允许该通道采样，=FALSE(或 0)表示禁止该通道采样。

##### **nBridgeConfig**

桥配置，取值范围如下表

| 常量名                     | 常量值   | 功能定义     | 备注 |
|-------------------------|-------|----------|----|
| PXI8720_FullBridgeI     | 10183 | 全桥 I     |    |
| PXI8720_FullBridgeII    | 10184 | 全桥 II    |    |
| PXI8720_FullBridgeIII   | 10185 | 全桥 III   |    |
| PXI8720_HalfBridgeI     | 10188 | 半桥 I     |    |
| PXI8720_HalfBridgeII    | 10189 | 半桥 II    |    |
| PXI8720_QuarterBridgeI  | 10271 | 1/4 桥 I  |    |
| PXI8720_QuarterBridgeII | 10272 | 1/4 桥 II |    |

##### **fBridgeResistance**

应变计电阻，取值范围如下表

| 常量名                           | 常量值 | 功能定义 | 备注 |
|-------------------------------|-----|------|----|
| PXI8720_BRIDGE_RESISTANCE_120 | 1   | 120Ω |    |
| PXI8720_BRIDGE_RESISTANCE_350 | 2   | 350Ω |    |
| PXI8720_BRIDGE_RESISTANCE_1K  | 3   | 1KΩ  |    |

##### **nShuntResistance**

分流校准电阻，取值范围如下表

| 常量名                              | 常量值 | 功能定义   | 备注 |
|----------------------------------|-----|--------|----|
| PXI8720_SHUNT_RESISTANCE_DISABLE | 0   | 禁止(默认) |    |
| PXI8720_SHUNT_RESISTANCE_50K     | 1   | 50Ω    |    |
| PXI8720_SHUNT_RESISTANCE_100K    | 2   | 100Ω   |    |
| PXI8720_SHUNT_RESISTANCE_33K     | 3   |        |    |

#### **nExcitationVolt**

激励电压，取值范围如下表

| 常量名                           | 常量值 | 功能定义   | 备注 |
|-------------------------------|-----|--------|----|
| PXI8720_EXCITATION_VOLT_0625  | 0   | 0.625V |    |
| PXI8720_EXCITATION_VOLT_1000  | 1   | 1V     |    |
| PXI8720_EXCITATION_VOLT_1500  | 2   | 1.5V   |    |
| PXI8720_EXCITATION_VOLT_2000  | 3   | 2V     |    |
| PXI8720_EXCITATION_VOLT_2500  | 4   | 2.5V   |    |
| PXI8720_EXCITATION_VOLT_2750  | 5   | 2.75V  |    |
| PXI8720_EXCITATION_VOLT_3300  | 6   | 3.3V   |    |
| PXI8720_EXCITATION_VOLT_5000  | 7   | 5V     |    |
| PXI8720_EXCITATION_VOLT_7500  | 8   | 7.5V   |    |
| PXI8720_EXCITATION_VOLT_10000 | 9   | 10V    |    |

#### **fGageFactor**

应变因子，应变功能时使用

#### **fInitBridgeVolt**

初始化电压，应变功能时使用，此值不可写，且在应变校准后产生合适的值。

#### **fPoissonRatio**

泊松比，应变功能时使用

#### **fWireResistance**

导线/桥电阻

#### **nMeasType**

测量类型

| 常量名                     | 常量值 | 功能定义 | 备注 |
|-------------------------|-----|------|----|
| PXI8720_MEASTYPE_BRIDGE | 0   | 桥测量  |    |
| PXI8720_MEASTYPE_STRAIN | 1   | 应变测量 |    |

#### 4.1.2 AI\_START\_TRIG (AI 开始触发结构体)

##### nTriggerType

触发类型(Trigger Type)。取值范围如下表：

| 常量名                          | 常量值 | 功能定义           | 备注  |
|------------------------------|-----|----------------|-----|
| AI_START_TRIGTYPE_NONE       | 0   | 无触发(等同于软件强制触发) | 默认值 |
| AI_START_TRIGTYPE_DIGIT_EDGE | 3   | 数字边沿触发类型       |     |

##### nTriggerDir

触发方向(Digital Trigger Direction)。

当触发类型为模拟边沿触发或数字边沿触发时(nTriggerType = TRIGTYPE\_DIGIT\_EDGE)，触发方向的取值范围如下表：

| 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)，则意味着在开始采集任务后，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 暂停触发结构体)

##### nTriggerType

触发类型(Trigger Type)。取值范围如下表：

| 常量名                    | 常量值 | 功能定义        | 备注  |
|------------------------|-----|-------------|-----|
| AI_PAUSE_TRIGTYPE_NONE | 0   | 无触发(禁用暂停触发) | 默认值 |
| AI_TRIGTYPE_DIGIT_EDGE | 3   | 数字边沿触发类型    |     |

### nTriggerSource

触发源(Trigger Source)，当触发类型不为数字电平触发时(即 nTriggerType 不等于 PAUSE\_TRIGTYPE\_DIGIT\_LVL)时触发源的取值范围如下表：

| 常量名            | 常量值 | 功能定义      | 备注  |
|----------------|-----|-----------|-----|
| AI_TRIGSRC_AI0 | 0   | 模拟触发源 AI0 | 默认值 |
| AI_TRIGSRC_AI1 | 1   | 模拟触发源 AI1 |     |
| AI_TRIGSRC_AI2 | 2   | 模拟触发源 AI2 |     |
| AI_TRIGSRC_AI3 | 3   | 模拟触发源 AI3 |     |
| AI_TRIGSRC_AI4 | 4   | 模拟触发源 AI4 |     |
| AI_TRIGSRC_AI5 | 5   | 模拟触发源 AI5 |     |
| AI_TRIGSRC_AI6 | 6   | 模拟触发源 AI6 |     |
| AI_TRIGSRC_AI7 | 7   | 模拟触发源 AI7 |     |

当触发类型为数字电平触发(即 nTriggerType=PAUSE\_TRIGTYPE\_DIGIT\_LVL)时触发源取值范围如下表：

| 常量名            | 常量值 | 功能定义      | 备注 |
|----------------|-----|-----------|----|
| AI_TRIGSRC_DTR | 0   | 数字触发源 DTR |    |

### nTriggerDir

触发方向(Digital Trigger Direction)。

当触发类型为模拟边沿触发或数字边沿触发时(nTriggerType = TRIGTYPE\_ANALOG\_EDGE/TRIGTYPE\_DIGIT\_EDGE)，触发方向的取值范围如下表：：

| 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) |     |

当触发类型为模拟窗触发(nTriggerType = TRIGTYPE\_ANALOG\_WIN)，触发方向的取值范围如下表：

| 常量名                              | 常量值 | 功能定义       | 备注  |
|----------------------------------|-----|------------|-----|
| AI_START_TRIGDIR_EnteringWin     | 0   | 入窗触发       | 默认值 |
| AI_START_TRIGDIR_LeavingWin      | 1   | 出窗触发       |     |
| AI_START_TRIGDIR_LeavingEnterWin | 2   | 变化(出入窗均有效) |     |

### 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。

#### nReserved0

保留字段(未作定义)，可强制赋 0。

### 4.1.4 AI\_PARAM(AI 工作参数结构体)

#### 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]。只读参数，它是由 AI\_InitTask()函数根据 CHParam[]通道组阵列中 bChannelEn=1 的通道个数返回的值。

#### CHParam[4]

通道组(Channel Parameter)，共 8 个单元，分别控制 8 个采样通道的选择。每个有效单元决定要采样通道使能，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)》。

#### 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：sps(sample per second)，即每秒样点。它指每个采样通道的同步采样速率，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大采样率。

## StartTrig

AI 开始触发参数(Start Trigger)，属于参数结构 AI\_START\_TRIG，每次在调用 AI\_StartTask() 之后，它决定设备实际开始采样的时机(触发条件)。请参考《[4.1.2 AI\\_START\\_TRIG \(AI 开始触发结构体\)](#)》

## PauseTrig

AI 暂停触发参数(Pause Trigger)，属于参数结构 AI\_PAUSE\_TRIG，每次在调用 AI\_StartTask() 之后，它决定设备在开始触发之后的暂停时机 (触发条件)。请参考《[4.1.2 AI\\_PAUSE\\_TRIG \(AI 暂停触发结构体\)](#)》

相关函数：[DeviceOpen\(\)](#)      [AI\\_InitTask\(\)](#)  
[DeviceClose\(\)](#)

## 4.2 AI\_STATUS (AI 工作状态信息结构)

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

```
typedef struct _PXI8720_AI_STATUS
{
    U32 bTaskDone;
    U32 bTriggered;
```



```

U32 nTaskState;
U32 nAvailSampsPerChan;
U32 nMaxAvailSampsPerChan;
U32 nBufSampsPerChan;
I64 nSampsPerChanAcquired;

```

```

U32 nExcitationVolt[4];
U32 bExcitationVoltDone;

```

```

U32 nHardOverflowCnt;
U32 nSoftOverflowCnt;
U32 nInitTaskCnt;
U32 nReleaseTaskCnt;
U32 nStartTaskCnt;
U32 nStopTaskCnt;
U32 nTransRate;

```

```

U32 nReserved;

```

```

} 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

    nExcitationVolt(0 to 4) As Long
    bExcitationVoltDone As Long

    nHardOverflowCnt As Long
    nSoftOverflowCnt As Long
    nInitTaskCnt As Long
    nReleaseTaskCnt As Long
    nStartTaskCnt As Long
    nStopTaskCnt As Long
    nTransRate As Long
    nReserved As Long

```

End Type

**LabVIEW:**

请参考 PXI8720.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。如果此值比较贴进于 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。

#### **nExcitationVolt[4]**

当前激励电压

#### **bExcitationVoltDone**

激励电压输出完成标志

#### **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\_ReleaseTask() 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。

**nReserved**

保留字段(未作定义)。

相关函数：[AI\\_GetStatus\(\)](#)

## 4.3 设备信息结构体

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

```
typedef struct _PXI8720_DEV_INFO
{
    U32 nSerialCode;
    BOOL bUsed;
}PXI8720_DEV_INFO, *PPXI8720_DEV_INFO;
```

**Visual Basic:**

```
Private Type _PXI8720_DEV_INFO
{
    nSerialCode As Long
    bUsed As Long
}
```

**LabVIEW:**

请参考 PXI8720.lvlib 库文件及相关演示 vi。

**nSerialCode**

设备的硬件序列号

**bUsed**

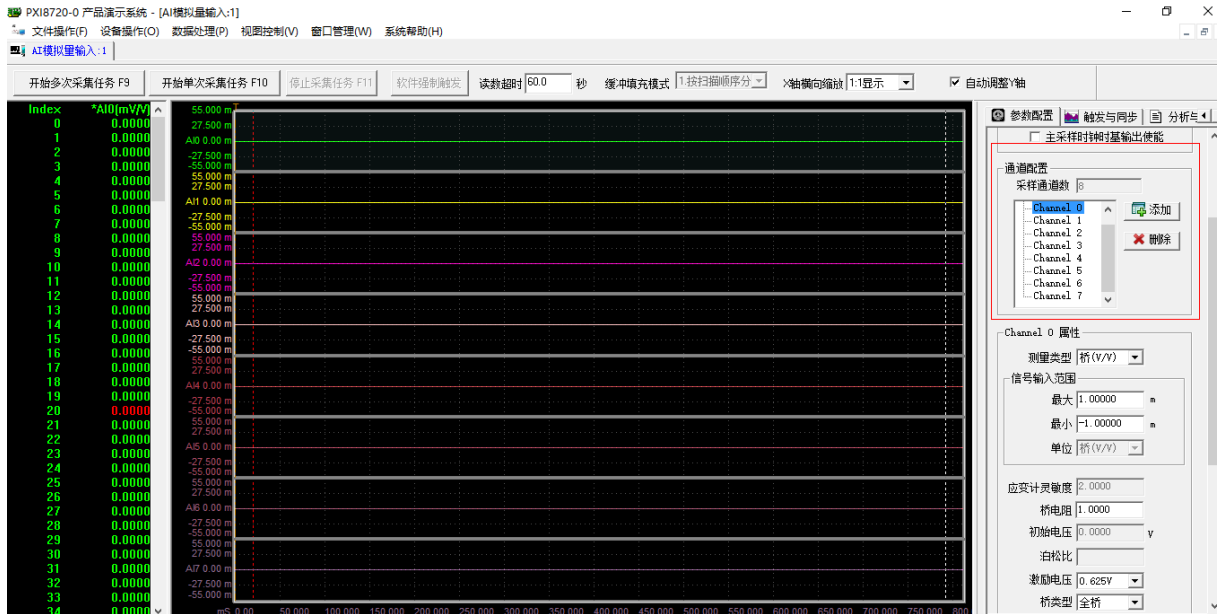
设备是否已经被调用

此结构体可用于枚举所有设备的硬件序列号，以及是否被使用。

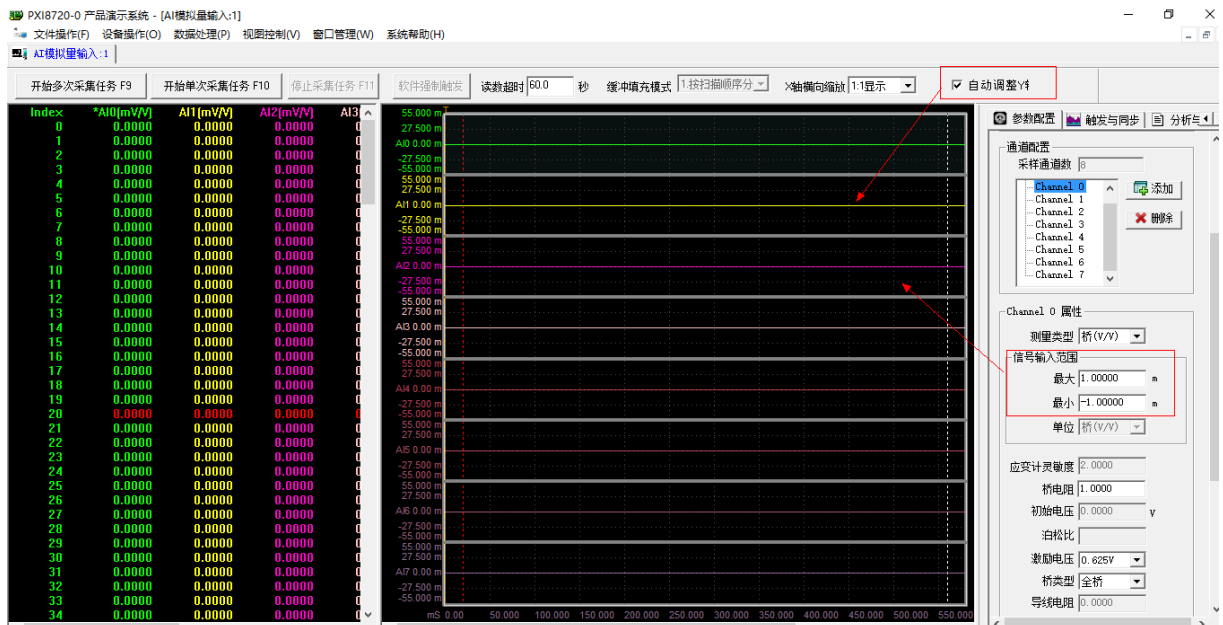
## 5 高级程序界面使用说明

### 5.1 总体说明

打开 PXI8720 界面，如下图：



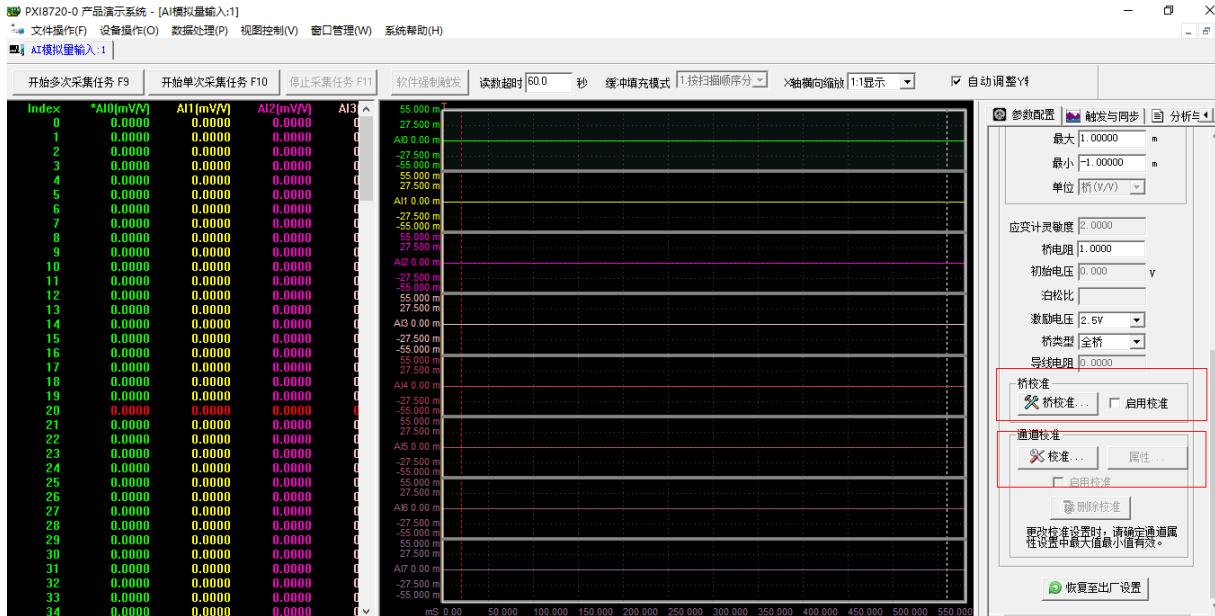
在右侧参数设置红色框里可以添加删除通道，当单击选择某通道节点时，下方会出现对应的通道属性，可在此修改对应的通道属性，其他操作和以前软件没有什么不同，只是在顶部的操作栏里添加如下图的自动调整 Y 轴：



选取自动调整 Y 轴时，曲线的 Y 轴会根据每次数据范围进行自动调整；如果不选中，则可以在通道的属性里的信号输入范围里设置 Y 轴的最大最小值。

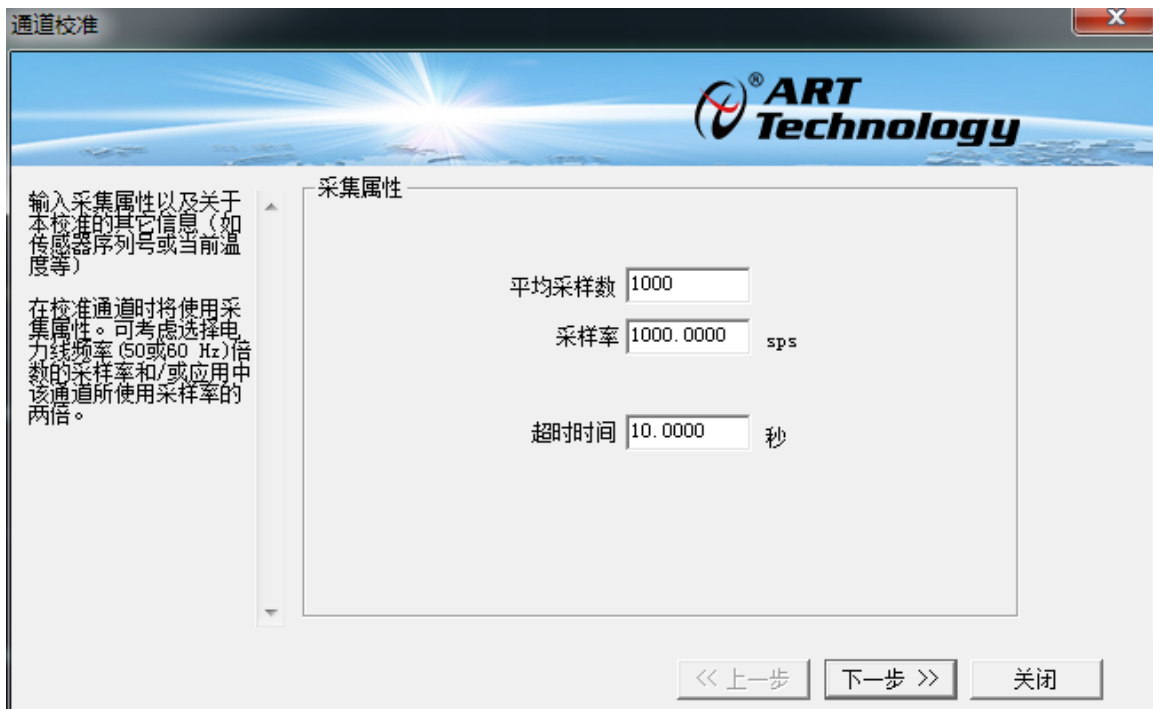
现主要说明下校准的使用。

测试前，如果需要对所需通道进行应变校准或通道校准，可在通道属性中点击应变校准或通道校准按钮，如下图：



## 5.2 应变校准说明

点击相应通道属性下的应变校准按钮，出现下图



可以在采集属性里填入认为合适的值，也可以默认。

点击下一步，出现下图：



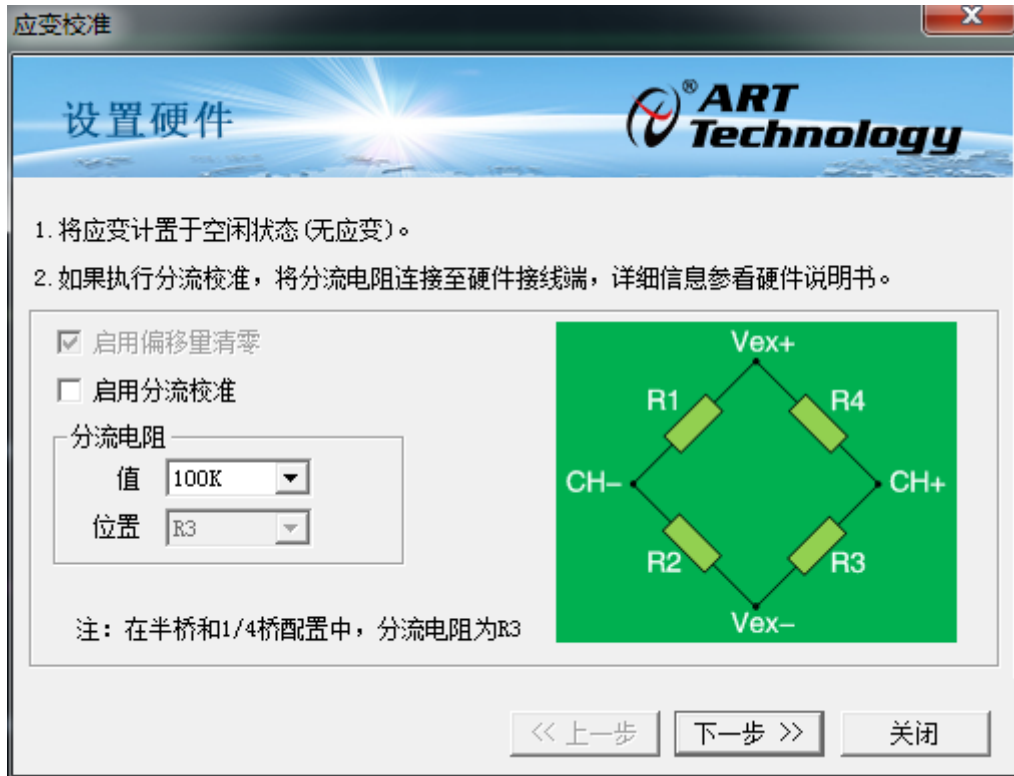
此时显示为当前通道的参考值、未校准值以及差值，软件会自动读取一行的值，这时需要对硬件进行设置，然后点击参考值的单元格填入参考值，等稳定后，点击下一步，软件会自动读取下一行的值，继续进行硬件设置，填入参考值，最多可支持7次。

完成需要的值后，点击校准按钮，即可完成本通道的校准。

另外，也可以点击右上角的显示为开始的按钮，停止软件自动读取，此时，需要手动输入参考值和未校准值，其他步骤和上述一样。

### 5.3 应变计校准

点击相应通道属性下的应变计校准按钮，出现如下：



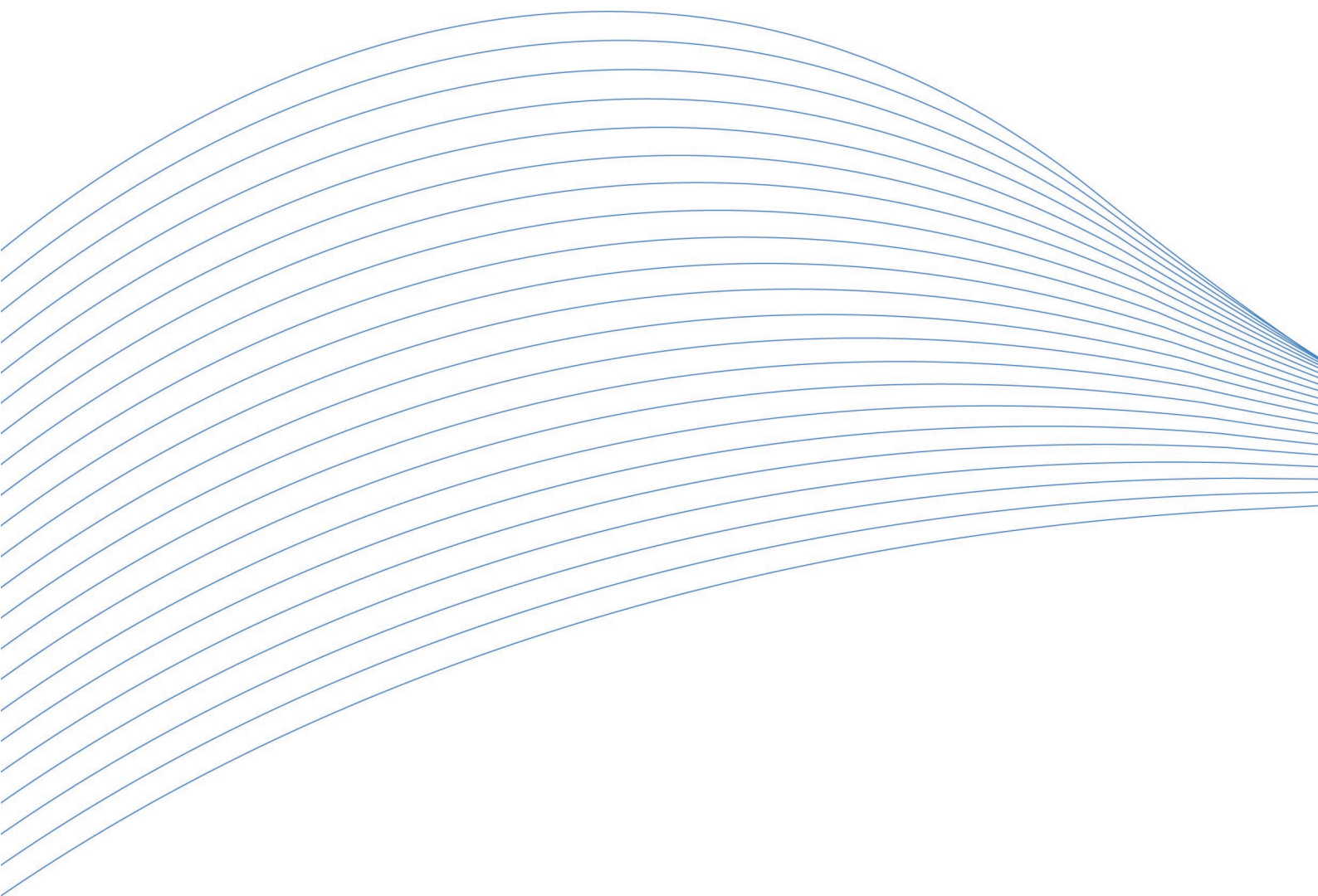
在上图中, 默认启用偏移量清零, 如需要也可以选择启用分流校准, 另外可以选择分流电阻的阻值。

然后点击下一步, 出现下图界面:



在此界面点击校准, 可以对相应通道进行校准, 点击重置数据可以恢复默认数据。





阿尔泰科技

服务热线：400-860-3335

网址：[www.art-control.com](http://www.art-control.com)