USB7104B 数据采集卡编程手册

北京中泰联创科技有限公司

目 录

版权信息

第一章　编程方法

1.1 概述

USB7104B 是 16 位 2.5MHz 并行采集设备，板载 32MB 硬件缓冲区保证其可以长时间连续采集。在安装好驱动后，就可以按照下面介绍的方法编程操作设备了。

注意：本设备采集通道数超过 2 路时必须使用外供电，否则设备会工作异常。

1.2 模拟量（AD）编程方法

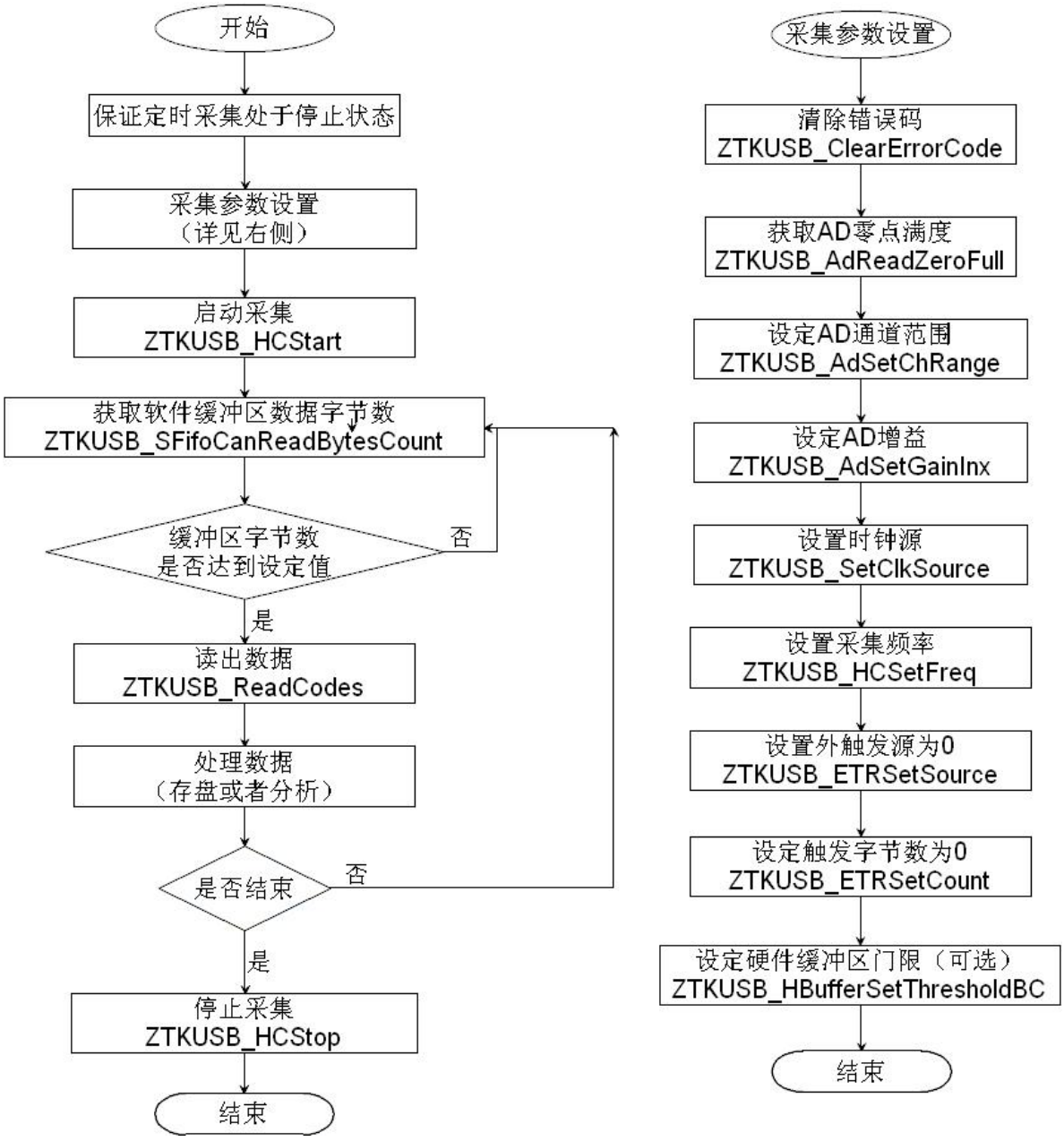
相关函数：

ZTKUSB_ClearErrorCode
ZTKUSB_AdSetGainInx
ZTKUSB_AdReadZeroFull
ZTKUSB_AdSetChRange
ZTKUSB_HCSetFreq
ZTKUSB_HCStart
ZTKUSB_HCStop
ZTKUSB_SFifoCanReadBytesCount
ZTKUSB_SFifoLostBC
ZTKUSB_SFifoLostBCClear
ZTKUSB_ReadCodes
ZTKUSB_HBufferSetThresholdBC
ZTKUSB_HBufferGetThresholdBC
ZTKUSB_SetClkSource
ZTKUSB_ETRSetSource
ZTKUSB_ETRSetThresholdValue
ZTKUSB_ETRSetCount
ZTKUSB_ETRSetEdgeOrLevel
ZTKUSB_ETRSetUpOrDown
ZTKUSB_ETRBufferCanReadBC

内部定时采集：

本设备板载 32MB 内存，内存分为两个缓冲区交替存储数据，只有当一个缓冲区满之后才能够开始读取数据，因此从开始采集到可以开始读取数据有一个迟滞，迟滞的时间取决于门限设置，具体内容请参考“2.3 门限设置规则”。虽然读取数据有迟滞，但是采集数据是在调用启动采集函数后马上进行的，只是要等到缓冲区内数据量达到门限时才能读出，采集并没有延时。

流程图如下：

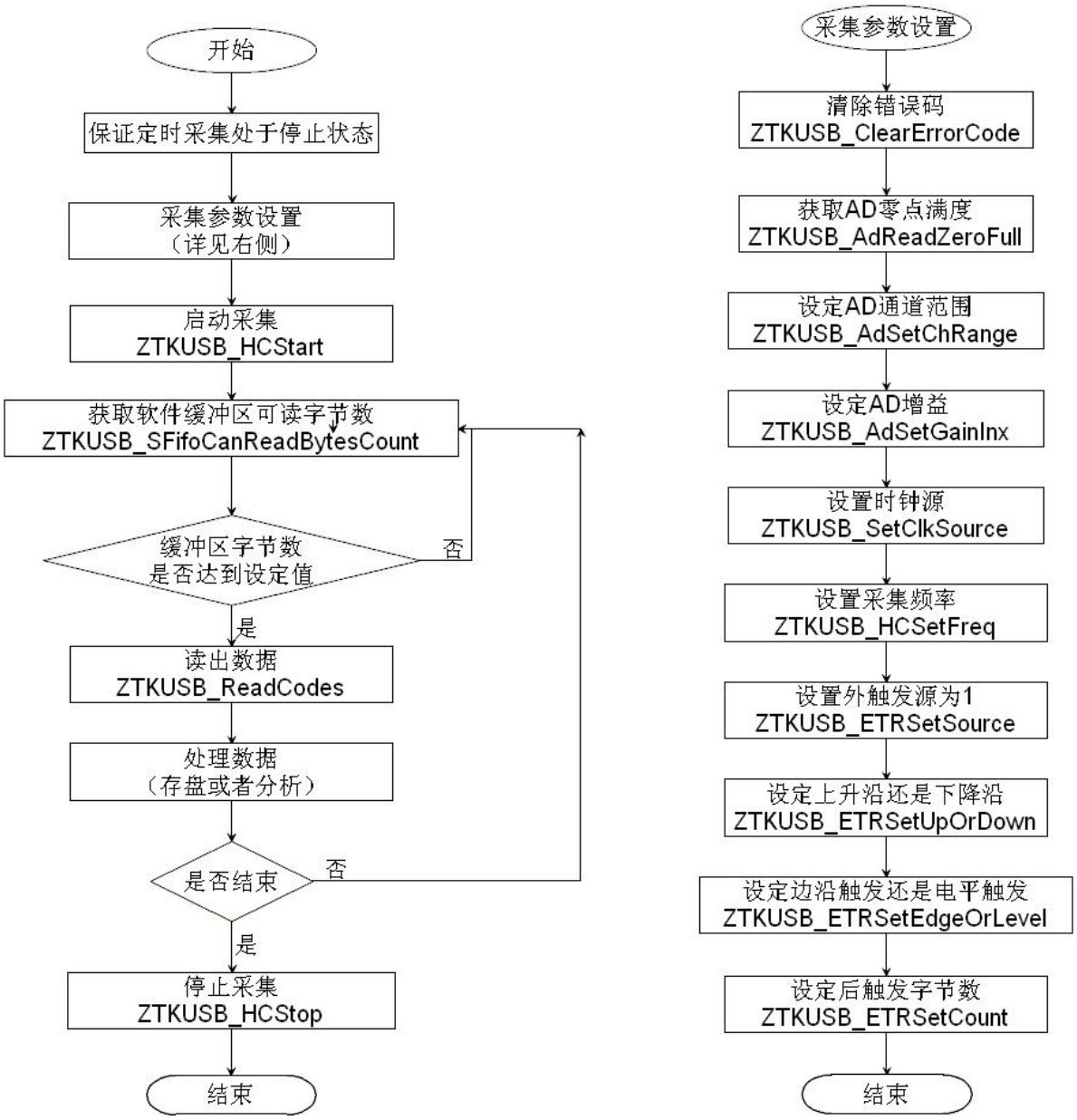


数字外触发采集：

数字外触发使用 ETR 管脚接入的信号判断是否进行采集，属于硬件控制采集，所以响应速度较快。

用户可设定外触发采集个数。注意采集个数一定要设定成通道的整数倍，否则缓冲区中的数据将会发生通道混乱。

流程图如下：



模拟外触发采集：

模拟外触发本质上仍然是内部定时采集，驱动根据指定通道的 AD 采集值来判断是否满足触发条件，根据计算机运算速度的不同，其响应速度也有所不同。

使用模拟外触发请注意以下限制：

1.必须先调用 ZTKUSB_AdSetChRange 设置需要采集的通道范围，然后再调用 ZTKUSB_ETRSetCount 设置预触发和后触发数据个数。

2.软件缓冲区大小为 32MB，也就是 33554432 字节，因此前触发和后触发字节数之和不能超过此数值。因为每个数据占 2 个字节，所以前触发和后触发设置值之和不能超过 16777216。

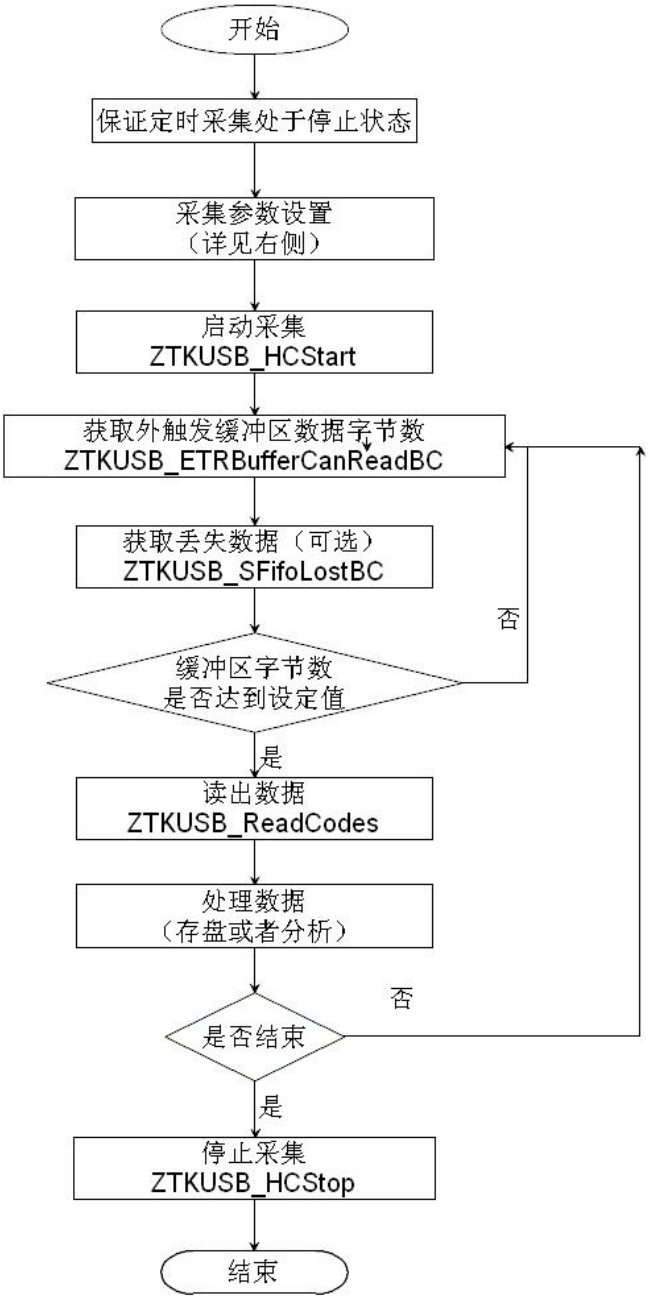
3.前触发个数和后触发个数均需要是通道数目的整数倍，例如需要采集的通道时 4 通道，则设定值必须是 4 的整数倍。

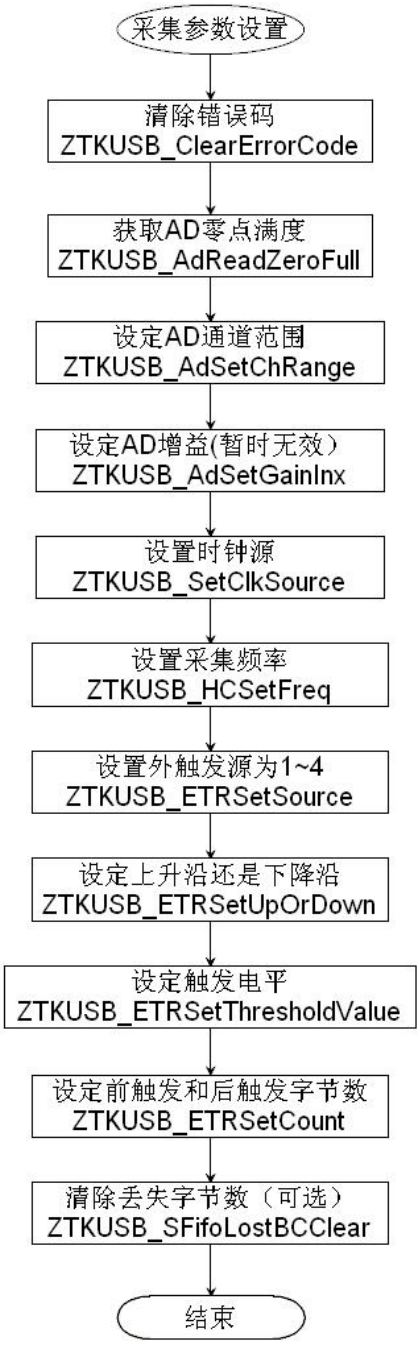
4.前触发字节数和后触发字节数不能够为 0。

5.外触发缓冲区字节数到达设定字节数后要尽快一次性将数据读出，否则驱动缓冲区将会溢出，导致数据丢失。

6.模拟目前外触发只支持上升沿和下降沿采集

流程图如下：


开始
保证定时采集处于停止状态
采集参数设置
（详见右侧）
启动采集
ZTKUSB_HCStart
获取外触发缓冲区数据字节数
ZTKUSB_ETRBufferCanReadBC
获取丢失数据（可选）
ZTKUSB_SFifoLostBC
缓冲区字节数
是否达到设定值
否
是
读出数据
ZTKUSB_ReadCodes
处理数据
（存盘或者分析）
是否结束
否
是
停止采集
ZTKUSB_HCStop
结束


采集参数设置
清除错误码
ZTKUSB_ClearErrorCode
获取AD零点满度
ZTKUSB_AdReadZeroFull
设定AD通道范围
ZTKUSB_AdSetChRange
设定AD增益(暂时无效)
ZTKUSB_AdSetGainInx
设置时钟源
ZTKUSB_SetClkSource
设置采集频率
ZTKUSB_HCSetFreq
设置外触发源为1~4
ZTKUSB_ETRSetSource
设定上升沿还是下降沿
ZTKUSB_ETRSetUpOrDown
设定触发电平
ZTKUSB_ETRSetThresholdValue
设定前触发和后触发字节数
ZTKUSB_ETRSetCount
清除丢失字节数（可选）
ZTKUSB_SFifoLostBCClear
结束

1.3 计数器编程方法

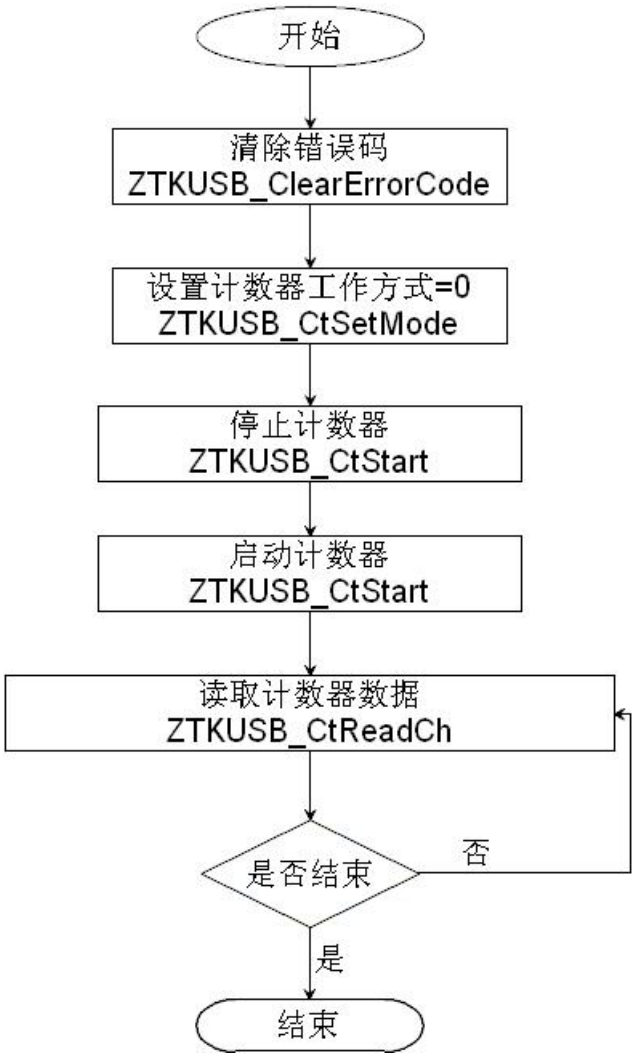
相关函数：

ZTKUSB_ClearErrorCode
ZTKUSB_CtSetMode
ZTKUSB_CtStart
ZTKUSB_CtReadCh
ZTKUSB_CtSetFreqBase

计数采集：

使用 ZTKUSB_CtStart 将计数器设成停止状态时，计数器值将被清零。因此如果用户希望计数器清零，则将计数器设成停止状态再设成启动状态即可。

下面是具体操作的流程图：



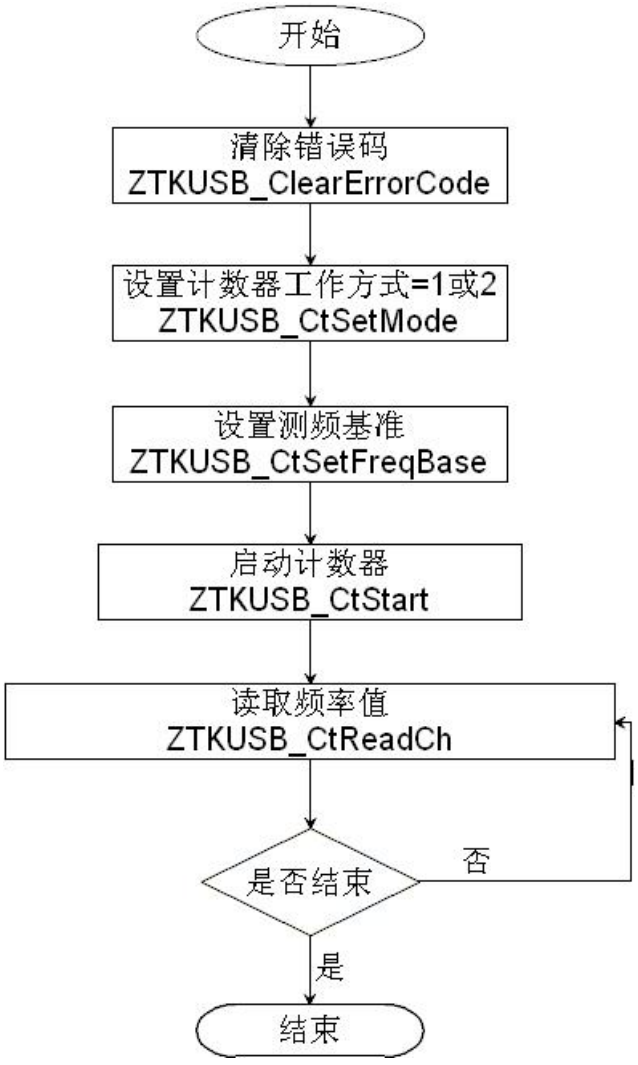
频率采集

本设备的测频基准时钟是由 1MHz 时钟分频而得，因此测频基准最小值是 1uS（0.001mS）。最大值是 2 的 32 次方微秒（4294967296uS，约为 1 小时），用到最大值的可能性几乎不存在。

可以根据被测信号频率的快慢选择测低频还是测高频，一般 1KHz 以下的频率值为低频值，1KHz 以上的频率值为高频值。

用户要根据自己被测信号的特点选择合适的工作方式，这样才能够获得较高的测频精度与测频速度。若用户不清楚被测信号的频率范围，可以先使用测低频的方式获得被测信号的大致频率值。如果发现被测频率较高，则可以进一步使用测高频工作方式得到较为精确的频率值。

具体流程图如下：



1.4 开关量编程方法

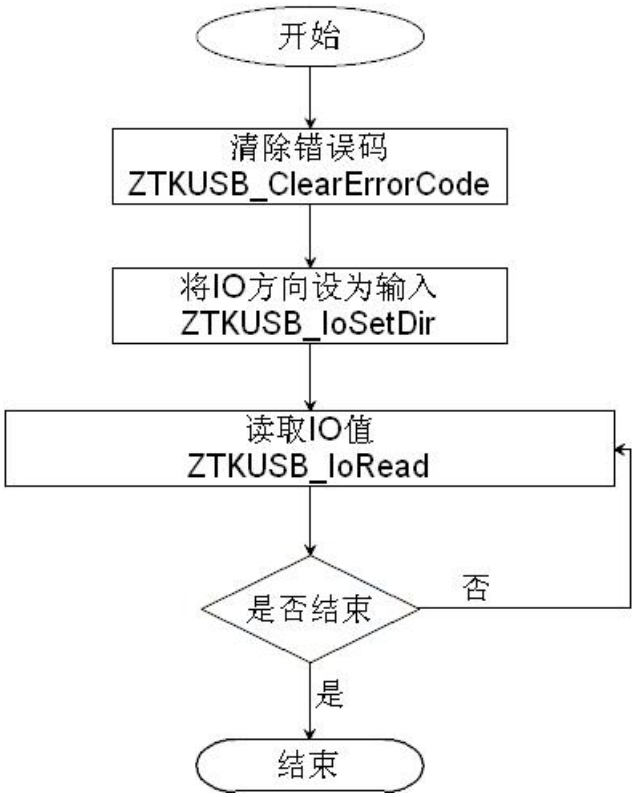
相关函数：

ZTKUSB_ClearErrorCode
ZTKUSB_SetDoFunction
ZTKUSB_IoSetDir
ZTKUSB_IoRead
ZTKUSB_IoWrite
ZTKUSB_DiRead
ZTKUSB_DoWrite

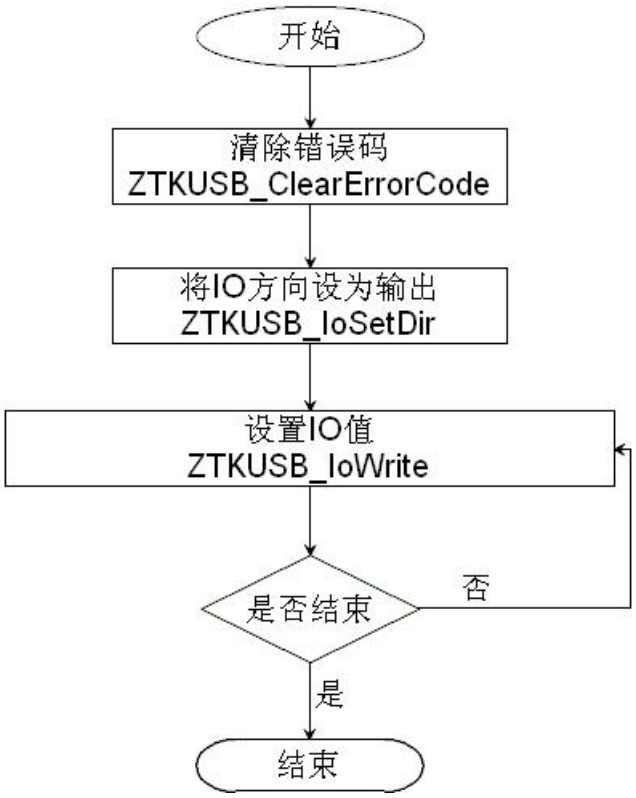
IO 操作：

本设备的 IO 是指可以设置输入和输出方向的 8 路 IO。注意 8 路要么同时设成输出，要么同时设成输入。

IO 读流程图：

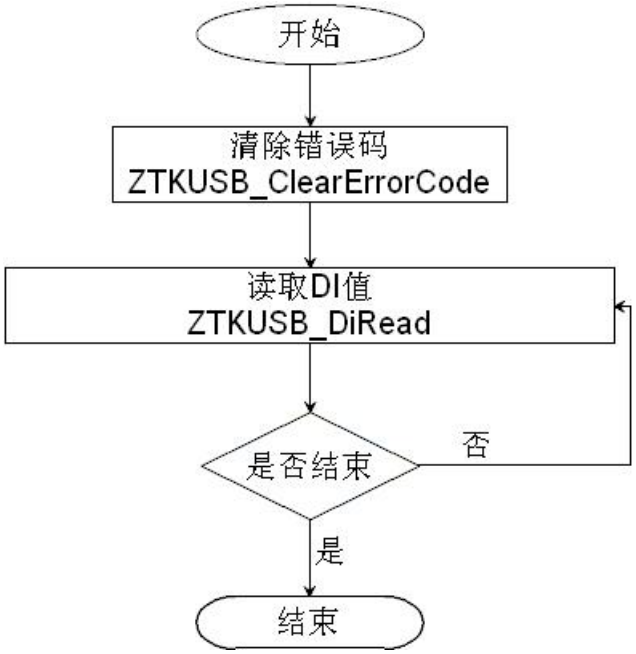


IO 写流程图：



DI 操作：

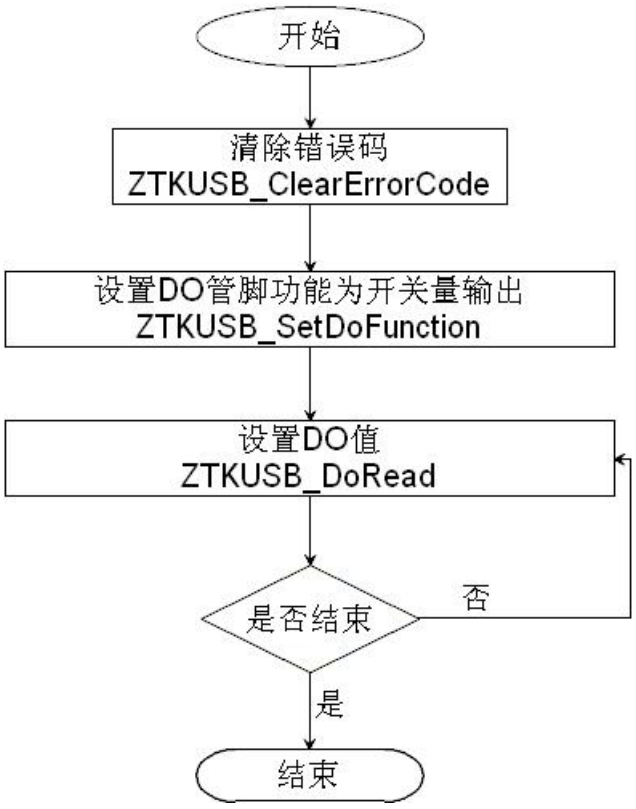
流程图如下：



DO 操作：

DO 管脚和 PWM 管脚复用，因此如果对应管脚用于 DO 功能，则无法用于 PWM 输出，

反之亦然。
流程图如下：



1.5 编码器编程方法

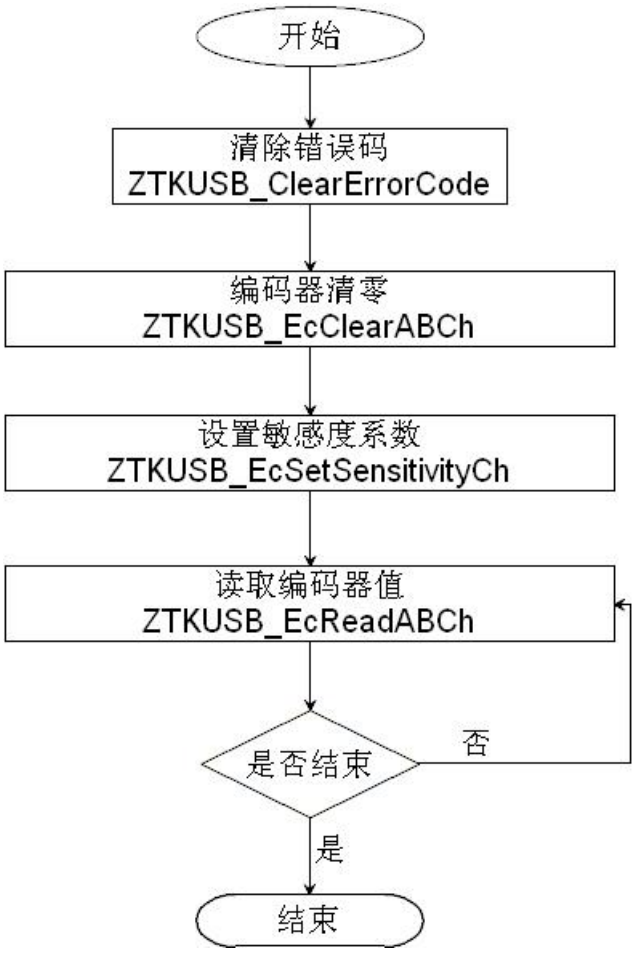
相关函数：

ZTKUSB_ClearErrorCode
ZTKUSB_EcClearABCh
ZTKUSB_EcSetSensitivityCh
ZTKUSB_EcReadABCh

编码器采集：

如果用户的编码器输出有杂波，可以使用灵敏度系数来进行滤波，灵敏度系数设定值越大，可以测量的编码器动作频率就越低。

流程图如下：



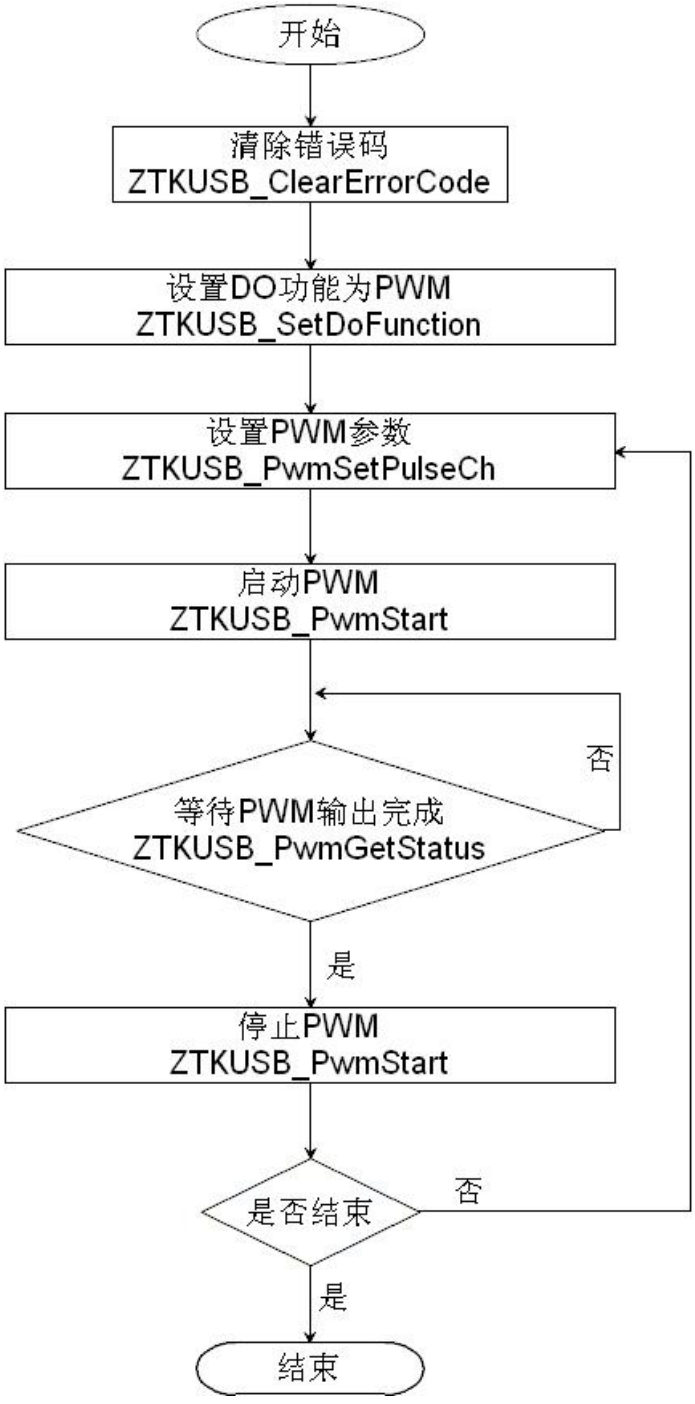
1.6 PWM 编程方法

相关函数：

ZTKUSB_ClearErrorCode
ZTKUSB_SetDoFunction
ZTKUSB_PwmSetPulseCh
ZTKUSB_PwmStart
ZTKUSB_PwmGetStatus

PWM 输出控制：

流程图如下：



第二章　基本原理

2.1 AD 原码值与物理值的转换

AD **原码值**是从设备读回的二进制数据，它是一个数字量，需要经过相应的转换才能得到实际的电压值或者电流值，我们称转换后的值为**物理值**。

本设备原码值使用 16 位 2 进制补码。在编程时，使用 16 位有符号整型来储存原码值。理想情况下，采集范围内最小输入物理值对应的原码值是-32768，最大输入物理值对应的原码值是 32767，零点值对应的是 0。但是在实际中，每个通道会有一点细微的差别。为了补偿这个差别，我们在生产过程中使用标准信号对设备进行校准，将每一路的 AD 信号对应的实际零点值和满度值记录到设备的 ROM 中。用户使用“读取零点满度信息”命令可以得到所有通道的零点值和满度值。得到这两个值后，可以计算出实际的物理值。

我们储存的零点值是在接地（0V）时对应的原码值，满度值是在接 90%满度电压时对应的原码值，下面以“原码值”代表从设备读回的数据，原码值与物理值的换算关系如下：

±10V：

实际电压值 = （原码值 - 零点值）/（满度值-零点值）* 9(V)

±5V：

实际电压值 = （原码值 - 零点值）/（满度值-零点值）* 4.5(V)

0~20mA：

0~20mA 实际使用的采集范围是±5V，板上使用 249Ω精密电阻将电流信号转换成电压信号，计算公式如下：

实际电流值 = （原码值 - 零点值）/（满度值-零点值）* 4500 / 249 (mA)

2.2 AD 数据排列方式

启动采集后，驱动自动从设备读出数据放到软件缓冲区中，用户可以使用函数 ZTKUSB_ReadCodes 读回这些数据，这些数据都是 16 位有符号整型。下面举例说明使能不同的通道时数据的排列方式。

下面举例说明使能不同的通道时数据的排列方式：

使能 AD1~AD4 通道：

数据索引	0	1	2	3	4	5	6	7	……
通道号	1	2	3	4	1	2	3	4	……

使能 AD2、AD3 通道：

数据索引	0	1	2	3	4	5	6	7	……
通道号	2	3	2	3	2	3	2	3	……

使能 AD1 通道：

数据索引	0	1	2	3	4	5	6	7	……
通道号	1	1	1	1	1	1	1	1	……

2.3 门限设置规则

对于一般用户，无需考虑门限值的设定，驱动会根据采集的设定自动调整门限值。

驱动内部和门限值相关的函数有设定频率函数（ZTKUSB_HCSetFreq）和外触发数据个数函数（ZTKUSB_ETRSetCount），如果用户想自行设定门限值，需要在调用这两个函数之后进行，否则设定值会被覆盖。

本设备具有 2 个 16MB 缓冲区交替存储数据，只有当一个缓冲区到达读取门限后才能够读取里面的数据，门限设定的最大值为 16MB，当 4 通道全部使能，2.5MPS 采集频率时，缓冲区数据量到达门限值的时间约为 0.8S。用户可以将门限值设小来实现更快的读取时间，但是当门限值变小后，可用的缓冲区也相应变小。比如门限值设为 8MB 时，则可用的缓冲区大小就是 2 两个 8MB，如果用户在两个缓冲区全部填满之前没有即使读取数据，数据将会丢失。

使用 ZTKUSB_HBufferSetThresholdBC 函数可以设定缓冲区门限，虽然这个函数只有一个参数，但是驱动内部将其分解成了两个整数，我们将其称为 bank 和 addr，门限大小和这两个整数有如下关系：

门限 = bank * addr * 65536

因此门限字节数不能是连续的整数，而是一个按照上面的规则，和设定值比较接近的整数。用户无需关心 bank 和 addr 值，直接调用 ZTKUSB_HBufferGetThresholdBC 函数就可以返回实际门限值。

注意：门限设置值过小会导致数据丢失。

2.4 采集频率设置详解

本设备采集使用的系统时钟是 80MHz，受限于此，无法设置任意频率。

本设备最高采集频率是 2.5MHz，用户可设置采集频率 2.5MHz，接下来用户可设置的最高频率是 1.25MHz。在 1.25MHz 以下有下面规律：

采集频率 = 80MHz / （分频系数 + 1）

其中“分频系数”已经封装在了 ZTKUSB_HCSetFreq 函数中，用户无法直接指定“分频系数”，但是函数会根据 freqHz 参数的设置找到最接近的分频系数，并且计算出实际的频率从 realFreq 参数中返回。用户在设定频率的时候使用上面的公式，会更容易获得需要的采集频率，分频系数是 16 位无符号整型，因此最低可设置的采集频率是 80MHz/65536≈1.22KHz。

2.5 多块设备的区分

本设备上面的拨码开关 S1 用于区分多块设备，S1 的设置值称为 SID。使用多块设备时，请用户将其设置成不同的值，驱动程序会根据 SID 排序设备。也就是 SID 最小的值对应的设备序号 DevNo 为 1。

注意 SID 和 DevNo 值并不一致。DevNo 代表了是第几个设备。例如有两个设备，一个设备的 SID=2，另一个为 3，则 SID=2 的设备对应 DevNo 为 1，SID=3 的设备对应 DevNo 为 2。SID 的设定值可以不连续，但是 DevNo 的值必然连续。

2.6 错误码详解

所有函数调用后都将返回一个错误码，如果返回值为 0 表示调用成功，如果返回值为负值，则说明发生了错误，发生错误后用户需要调用 ZTKUSB_ClearErrorCode 函数清除错误码，否则一些函数有可能无法正确执行。一些错误不会引起设备工作异常，用户简单的清除错误码即可。但是一些错误会导致设备异常，用户需要查找错误点，排除错误后才能继续后面的操作。

不同的错误码代表不同的含义，具体如下：

-1：未归类错误

-2：设备号错误，当设备号设置小于 1，或者大于设备数量时，返回此错误。

-3：通道号错误。

-4：调用系统函数错误，一般是和内核驱动沟通时发生错误。

-5：已经启动采集（调用了 HCStart，还没有调用 HCStop）。一些操作不可以在启动采集后调用。

-6：内存错误，当计算机内存剩余空间不够驱动使用的时候发生。

-7：参数设置错误。当参数设置了不支持的数值时会发生此错误

-8：找不到正确的起始通道号。

-9：超时，它不一定是错误，有可能是下位机还没有准备好数据，比如外触发采集时还没有触发信号到来就会返回这个错误码

-10：建立读线程错误，一般是内存不足引起。

-11：读取数据量超过软件缓冲区数量。

-12：读缓冲区溢出，说明应用程序读取速度小于设备采集速度。

-13：设备丢失，此时可能是 USB 连接线出现问题。

第三章　关于赠送版虚拟仪器软件

购买我公司采集产品即可获赠 ZTLV 虚拟仪器软件，该软件在 labview 运行环境下使用。LabView2011 运行引擎可从 www.ztic.cn 下载。

该软件只发布可执行文件与必要 Windows 动态链接库文件，不提供源代码（VI 文件），如您需要请我们联系。

3.1 主要功能介绍

实时波形显示：在采集过程中实时显示数据曲线波形图。

实时数值显示：在采集过程中实时显示所采集数值的结果

数据存储与回放：对采集的信号进行存储与回放及分析。

显示及打印报表：以曲线或图表的形式打印报表。

图形分析功能：对数据曲线进行缩放、平移、定位（游标）、对比

注：具体功能按钮说明及界面显示分布请参见《中泰联创科技有限公司虚拟仪器软件》使用说明书。

3.2 扩展功能

NI labview 开放的图形化编程环境和中泰工控模块化硬件的结合，帮助客户简化开发，提高效率。为满足不同客户的使用需求，我们可以为您量身定制并提供优质的软件服务。

第四章　函数原型与功能详解

本章函数原型均为 C 语言方式，VC++可以在包含了“ZT_Type.h”头文件后直接使用下面的声明。

4.1 数据类型

I8：8 位有符号整型
U8：8 位无符号整型
I16：16 位有符号整型
U16：16 位无符号整型
I32：32 位有符号整型
U32：32 位无符号整型
F64：64 位双精度浮点型

4.2 函数说明

ZTKUSB_GetCount

函数原型：

I32 _stdcall ZTKUSB_GetCount(I32* devCount);

函数功能：

返回当前连接设备数量。

参数说明：

devCount：当前连接设备数量，需要用户分配空间。

返回值：

0 表示没有错误

ZTKUSB_GetSID

函数原型：

ZTKUSB_API I32 _stdcall ZTKUSB_GetSID(I32 devNo, U32* sID);

函数功能：

返回当前连接设备数量。

参数说明：

devNo：设备号，从 1 开始

sID：指定设备的 S1 拨码值，需要用户分配空间。

返回值：

0 表示没有错误

ZTKUSB_ClearErrorCode

函数原型：

I32 _stdcall ZTKUSB_ClearErrorCode(I32 devNo);

函数功能：

清除错误码

参数说明：

devNo：设备号，从 1 开始

返回值：

恒为 0

ZTKUSB_AdReadZeroFull

函数原型：

I32 _stdcall ZTKUSB_AdReadZeroFull(I32 devNo, I32 chNo, I32 rangeInx,
F64* zeroCode, F64* fullCode);

函数功能：

返回指定通道指定范围的 AD 零点满度原码值

参数说明：

devNo：设备号，从 1 开始

chNo：通道号，1~4

rangeInx：采集范围，0 表示±10V，1 表示±5V

zeroCode：零点原码，对应 0V；需要用户分配空间

fullCode：满度原码，对应满度值的 90%；需要用户分配空间

返回值：

0 表示没有错误

ZTKUSB_AdSetChRange

函数原型：

I32 _stdcall ZTKUSB_AdSetChRange(I32 devNo, I32 beginChNo, I32 endChNo);

函数功能：

设定允许采集的通道范围，只有在 beginChNo 到 endChNo 范围内的 AD 通道数据才能采集到。

参数说明：

devNo：设备号，从 1 开始

beginChNo：起始通道号，1~endChNo；

endChNo：终止通道号，beginChNo~4；

返回值：

0 表示没有错误

ZTKUSB_AdSetGainInx

函数原型：

I32 _stdcall ZTKUSB_AdSetGainInx(I32 devNo, I32 chNo, I32 gainInx);

函数功能：

设定 AD 增益。

参数说明：

devNo：设备号，从 1 开始

chNo：通道号，1~4；

gainInx：增益索引，0 对应 1 倍增益
1 对应 2 倍增益
2 对应 5 倍增益
3 对应 10 倍增益

返回值：

0 表示没有错误

ZTKUSB_SetClkSource

函数原型：

I32 _stdcall ZTKUSB_SetClkSource(I32 devNo, I32 clkSource);

函数功能：

设定数据采集时钟源。

参数说明：

devNo：设备号，从 1 开始

clkSource：时钟源，0 使用内部时钟
1 使用外部时钟

返回值：

0 表示没有错误

ZTKUSB_HCSetFreq

函数原型：

I32 _stdcall ZTKUSB_HCSetFreq(I32 devNo, F64 freqHz, F64* realFreq);

函数功能：

设定数据采集频率。由于设备内部使用 80MHz 时钟基础时钟获得的采集时钟，因此会有数字化误差，无法实现任意频率采集，一些频率只能得到最接近的值，这个频率值由 realFreq 参数返回。具体请参考“采集频率设置详解”

参数说明：

devNo：设备号，从 1 开始

freqHz：设定频率；可以设置 2500000Hz（2.5MHz）
以及 1221Hz~1250000Hz（1.25MHz）频率范围
注意 1250000Hz~2500000Hz 之间的频率是无法设置的。

realFreq：实际采集频率，需要用户分配空间

返回值：

0 表示没有错误

ZTKUSB_ETRSetSource

函数原型：

I32 _stdcall ZTKUSB_ETRSetSource(I32 devNo, I32 etrSource);

函数功能：

设定触发源。

参数说明：

devNo：设备号，从 1 开始

etrSource：触发源；0，表示不使用外触发，启动采集后就开始采集数据
1，使用硬件外触发，启动采集后等待外触发信号才开始采集
2~5，使用 AD 通道值作为触发源，分别对应 AD1~AD4

返回值：

0 表示没有错误

ZTKUSB_ETRSetEdgeOrLevel

函数原型：

I32 _stdcall ZTKUSB_ETRSetEdgeOrLevel(I32 devNo, I32 edgeOrLevel);

函数功能：

设定是使用边沿触发还是电平触发。当使用 AD 通道作为触发源时本函数无效

参数说明：

devNo：设备号，从 1 开始

edgeOrLevel：0，边沿触发，上升沿或者下降沿到来后开始采集
1，电平触发，高电平或者低电平时采集

返回值：

0 表示没有错误

ZTKUSB_ETRSetUpOrDown

函数原型：

I32 _stdcall ZTKUSB_ETRSetUpOrDown(I32 devNo, I32 upOrDown);

函数功能：

设定触发条件，当电平触发时，使用高电平还是低电平，当使用边沿触发时，使用上升沿还是下降沿。注意当使用 AD 通道作为触发源时只有边沿触发。

参数说明：

devNo：设备号，从 1 开始

upOrDown：0，使用上升沿（高电平）
1，使用下降沿（低电平）

返回值：

0 表示没有错误

ZTKUSB_ETRSetThresholdValue

函数原型：

I32 _stdcall ZTKUSB_ETRSetThresholdValue(I32 devNo, I32 chNo,
F64 etrValue, F64* realValue);

函数功能：

设定触发电压，只有当触发源为 AD 通道值时才有效。由于数字化误差，设定值和实际值会有些许误差。

参数说明：

devNo：设备号，从 1 开始

chNo：通道号，1~4

etrValue：设定触发电压

realValue：真实触发电压，需要用户分配空间。

返回值：

0 表示没有错误

ZTKUSB_ETRSetCount

函数原型：

I32 _stdcall ZTKUSB_ETRSetCount(I32 devNo,
U32 preCount, U32 postCount,
U32* realPreCount, U32* realPostCount);

函数功能：

设置外触发信号到来前后采集的数据字节数，当触发源为硬件外触发时（1）preCount 参数无效。realPostCount 也有一定的限制，具体请参考“门限设置规则”。

参数说明：

devNo：设备号，从 1 开始

preCount：预触发个数，也就是触发条件满足前所有使能通道总采集数据个数，只有在触发源为 AD 通道时才有效。

postCount：后触发个数，也就是触发条件满足后所有使能通道总采集数据个数

realPreCount：预触发个数的实际值，请参考“模拟外触发采集”章节。

realPostCount：预触发个数的实际值，请参考“模拟外触发采集”和“门限设置规则”

返回值：

0 表示没有错误

ZTKUSB_SFifoLostBCClear

函数原型：

I32 _stdcall ZTKUSB_SFifoLostBCClear(I32 devNo);

函数功能：

软件缓冲区丢弃字节数目清零

参数说明：

devNo：设备号，从 1 开始

返回值：

0 表示没有错误

ZTKUSB_SFifoLostBC

函数原型：

I32 _stdcall ZTKUSB_SFifoLostBC(I32 devNo, U32* lostBC);

函数功能：

本函数只有在使用 AD 通道作为触发源时才有效。软件缓冲区丢弃了多少字节数据，当软件缓冲区满后为了保证缓冲区中是最新数据，所以要丢弃以前的数据。被丢弃的数据将不会再参与外触发判断，如果此时有外触发信号，将不会被检测到。

参数说明：

devNo：设备号，从 1 开始

lostBC：丢弃字节数，需要用户分配数据

返回值：

0 表示没有错误

ZTKUSB_ETRBufferCanReadBC

函数原型：

I32 _stdcall ZTKUSB_ZTKUSB_ETRBufferCanReadBC(I32 devNo,
U32* canReadBC);

函数功能：

外触发缓冲区可读字节数

参数说明：

devNo：设备号，从 1 开始

canReadBC：可读字节数，需要用户分配数据

返回值：

0 表示没有错误

ZTKUSB_HCStart

函数原型：

I32 _stdcall ZTKUSB_HCStart(I32 devNo);

函数功能：

启动数据采集

参数说明：

devNo：设备号，从 1 开始

返回值：

0 表示没有错误

ZTKUSB_HCStop

函数原型：

I32 _stdcall ZTKUSB_HCStop(I32 devNo);

函数功能：

停止数据采集

参数说明：

devNo：设备号，从 1 开始

返回值：

0 表示没有错误

ZTKUSB_ReadCodes

函数原型：

I32 _stdcall ZTKUSB_ReadCodes(I32 devNo, I32 bytesRead,
I16* pOutBuffer, I32* realBytesRead,
I32 timeOutMS = 5000);

函数功能：

读取数据原码，其排列方式请参考“AD 数据排列方式”；当触发源为 AD 通道时从外触发缓冲区中读取数据，其余则从软件缓冲区中读取数据。

参数说明：

devNo：设备号，从 1 开始

bytesRead：要读取的字节数

pOutBuffer：数据缓冲区，需要用户分配，其大小为 bytesRead/2。

realBytesRead：实际读取字节数

timeOutMS：超时设置，以毫秒为单位，默认值为 5000

返回值：

0 表示没有错误

ZTKUSB_SFifoCanReadBytesCount

函数原型：

I32 _stdcall ZTKUSB_SFifoCanReadBytesCount(I32 devNo, U32* canReadBC);

函数功能：

返回当前软件缓冲区可读字节数。

参数说明：

devNo：设备号，从 1 开始

canReadBC：可读字节数，需要用户分配空间。

返回值：

0 表示没有错误

ZTKUSB_HBufferSetThresholdBC

函数原型：

I32 _stdcall ZTKUSB_HBufferSetThresholdBC(I32 devNo, U32 thresholdBC);

函数功能：

设定硬件缓冲区读取门限

参数说明：

devNo：设备号，从 1 开始

thresholdBC：门限设定字节数，具体请看“门限设置规则”

返回值：

0 表示没有错误

ZTKUSB_HBufferGetThresholdBC

函数原型：

I32 _stdcall ZTKUSB_HBufferGetThresholdBC(I32 devNo, U32* thresholdBC);

函数功能：

读取硬件缓冲区门限

参数说明：

devNo：设备号，从 1 开始

thresholdBC：当前门限设定字节数，具体请看“门限设置规则”，需要用户分配空间

返回值：

0 表示没有错误

ZTKUSB_CtSetMode

函数原型：

I32 _stdcall ZTKUSB_CtSetMode(I32 devNo, I8* ctMode, I32 chCount);

函数功能：

设置所有计数器工作方式

参数说明：

devNo：设备号，从 1 开始

ctMode：工作方式数组，数组元素 0~2 对应 CT1~CT3，以 CT1 为例：

ctMode[0]=0，计数

ctMode[0]=1，测低频

ctMode[0]=2，测高频

chCount：通道数，此处请固定设为 3，否则函数会返回错误。

返回值：

0 表示没有错误

ZTKUSB_CtSetFreqBase

函数原型：

I32 _stdcall ZTKUSB_CtSetFreqBase(I32 devNo, F64* freqBaseMS,
I32 chCount, F64* realFreqBase);

函数功能：

设定所有计数器时钟基准，注意只有在测频方式下时钟基准才有意义。

参数说明：

devNo：设备号，从 1 开始

freqBaseMS：以毫秒为单位的测频基准，最小值是.001mS
最大值建议不要超过 10000mS

chCount：通道数，此处请固定设为，否则函数会返回错误。

realFreqBase：以毫秒为单位的实际测频基准，由于数字量化误差，实际测频基准并不总是设定的测频基准。

返回值：

0 表示没有错误

ZTKUSB_CtStart

函数原型：

I32 _stdcall ZTKUSB_CtStart(I32 devNo, I8* startOrStop, I32 chCount);

函数功能：

设定所有计数器是否启动。

参数说明：

devNo：设备号，从 1 开始

startOrStop：启动设置数组，数组元素~2 对应 CT1~CT3，以 CT1 为例：
startOrStop[0]=0，禁止计数
startOrStop[0]=1，启动计数

chCount：通道数，此处请固定设为，否则函数会返回错误。

返回值：

0 表示没有错误

ZTKUSB_CtReadCh

函数原型：

I32 _stdcall ZTKUSB_CtReadCh(I32 devNo, I32 chNo, F64* ctValue);

函数功能：

读取指定通道的计数器值，当计数方式下，返回计数值
测频方式下，返回频率值

参数说明：

devNo：设备号，从 1 开始

chNo：通道号，1~3

ctValue：计数器值。当计数方式下，返回计数值，测频方式下，返回频率值

返回值：

0 表示没有错误

ZTKUSB_SetDoFunction

函数原型：

I32 _stdcall ZTKUSB_SetDoFunction(I32 devNo, I8* doFun, I32 chCount);

函数功能：

设定所有多功能 DO 管脚的工作方式

参数说明：

devNo：设备号，从 1 开始

doFun：DO 功能数组，数组元素~2 对应 DO1~DO3，以 DO1 为例：

doFun[0]=0，用于开关量输出

doFun[0]=1，用于 PWM 输出

chCount：通道数，此处请固定设为 3，否则函数会返回错误。

返回值：

0 表示没有错误

ZTKUSB_IoSetDir

函数原型：

I32 _stdcall ZTKUSB_IoSetDir(I32 devNo, I8* groupDir, I32 groupCount);

函数功能：

设定所有可编程输入输出开关量的方向

参数说明：

devNo：设备号，从 1 开始

groupDir：开关量的方向，数组元素对应 IO1~IO8

groupDir[0]=0，表示输入

groupDir[0]=1，表示输出

chCount：组数，此处请固定设为 1，否则函数会返回错误。

返回值：

0 表示没有错误

ZTKUSB_IoWrite

函数原型：

I32 _stdcall ZTKUSB_IoWrite(I32 devNo, I8* ioBits, I32 chCount);

函数功能：

设定所有可编程输出输出开关量（IO）的端口状态，当对应组开关量设置为输入时此函数没有效果

参数说明：

devNo：设备号，从 1 开始

ioBits：开关量的值，数组元素 0~7 对应 IO1~IO8，以 IO1 为例

ioBits[0]=0，表示输出低电平

ioBits[0]=1，表示输出高电平

chCount：组数，此处请固定设为 8，否则函数会返回错误。

返回值：

0 表示没有错误

ZTKUSB_IoRead

函数原型：

I32 _stdcall ZTKUSB_IoRead(I32 devNo, I8* ioBits, I32 chCount);

函数功能：

读取所有可编程输出输出开关量（IO）的端口状态，当对应组开关量设置为输出时此函数读回之前设定的状态

参数说明：

devNo：设备号，从 1 开始

ioBits：开关量的值，数组元素 0~7 对应 IO1~IO8，以 IO1 为例

ioBits[0]=0，表示输入低电平

ioBits[0]=1，表示输入高电平

chCount：组数，此处请固定设为 8，否则函数会返回错误。

返回值：

0 表示没有错误

ZTKUSB_DoWrite

函数原型：

I32 _stdcall ZTKUSB_DoWrite(I32 devNo, I8* doBits, I32 chCount);

函数功能：

设置所有 DO 端口状态

参数说明：

devNo：设备号，从 1 开始

doBits：开关量的值，数组元素 0~7 对应 DO1~DO8，以 DO1 为例

doBits[0]=0，表示输出低电平

doBits[0]=1，表示输出高电平

chCount：组数，此处请固定设为 8，否则函数会返回错误。

返回值：

0 表示没有错误

ZTKUSB_DoRead

函数原型：

I32 _stdcall ZTKUSB_DoRead(I32 devNo, I8* doBits, I32 chCount);

函数功能：

读取所有 DO 端口的设置状态

参数说明：

devNo：设备号，从 1 开始

doBits：开关量的值，数组元素 0~7 对应 DO1~DO8，以 DO1 为例

doBits[0]=0，表示设置为低电平
doBits[0]=1，表示设置为高电平
chCount：组数，此处请固定设为 8，否则函数会返回错误。

返回值：

0 表示没有错误

ZTKUSB_DiRead

函数原型：

I32 _stdcall ZTKUSB_DiRead(I32 devNo, I8* diBits, I32 chCount);

函数功能：

读取所有 DO 端口的设置状态

参数说明：

devNo：设备号，从 1 开始
diBits：开关量的值，数组元素 0~7 对应 DI1~DI8，以 DI1 为例
diBits[0]=0，表示设置为低电平
diBits[0]=1，表示设置为高电平
chCount：组数，此处请固定设为 8，否则函数会返回错误。

返回值：

0 表示没有错误

ZTKUSB_EcReadABCh

函数原型：

I32 _stdcall ZTKUSB_EcReadABCh(I32 devNo, I32 chNo, F64* abValue);

函数功能：

读取指定通道的编码器 AB 值

参数说明：

devNo：设备号，从 1 开始
chNo：通道号，1~3
abValue：编码器 AB 值

返回值：

0 表示没有错误

ZTKUSB_EcSetSensitivityCh

函数原型：

I32 _stdcall ZTKUSB_EcSetSensitivityCh(I32 devNo, I32 chNo, F64 sensUS);

函数功能：

设定指定通道的编码器灵敏度，以微秒为单位

参数说明：

devNo：设备号，从 1 开始
chNo：通道号，1~3
sensCode：灵敏度，以微秒为单位，1uS~32768uS,设置为的是后表示禁止灵敏度

功能。

返回值：

0 表示没有错误

ZTKUSB_EcClearABCh

函数原型：

I32 _stdcall ZTKUSB_EcClearABCh(I32 devNo, I32 chNo);

函数功能：

将指定通道的编码器 AB 值清零

参数说明：

devNo：设备号，从 1 开始

chNo：通道号，1~3

返回值：

0 表示没有错误

ZTKUSB_PwmSetPulseCh

函数原型：

I32 _stdcall ZTKUSB_PwmSetPulseCh(I32 devNo, I32 chNo, F64 freq,
F64 dutyCycle, I32 pulseCount,
F64* realFreq, F64* realDutyCycle);

函数功能：

设定指定通道的 PWM 输出各项参数

参数说明：

devNo：设备号，从 1 开始

freq：输出频率。.001Hz~1MHz，在某些频率下会有误差。

dutyCycle：占空比。0~1，具体调节档位取决于输出频率，输出频率越低，可调节的档位越多。

pulseCount：脉冲个数，如果设为，则表示连续输出

realFreq：真实频率值，由于数字量化误差，设定频率和真实频率之间会有些误差

realDutyCycle：真实占空比值，由于数字量化误差，设定占空比和真实占空比之间会有些误差

返回值：

0 表示没有错误

ZTKUSB_PwmStart

函数原型：

I32 _stdcall ZTKUSB_PwmStart(I32 devNo, I8* startOrStop, I32 chCount);

函数功能：

设定所有通道的 PWM 输出开始或者停止

参数说明：

devNo：设备号，从 1 开始

startOrStop：启动设置数组，数组元素 0~2 对应 PWM1~PWM3，以 PWM1 为例：
startOrStop[0]=0，停止输出
startOrStop[0]=1，启动输出
chCount：通道数，此处请固定设为 3，否则函数会返回错误。

返回值：

0 表示没有错误

ZTKUSB_PwmGetStatus

函数原型：

I32 _stdcall ZTKUSB_PwmGetStatus(I32 devNo, I8* chStatus, I32 chCount);

函数功能：

读取所有通道的 PWM 输出是否完成

参数说明：

devNo：设备号，从 1 开始

chStatus：是否完成，数组元素 0~2 对应 PWM1~3，以 PWM1 为例：
chStatus[0]=0，输出没有完成
chStatus[0]=1，输出已经完成

chCount：通道数，此处请固定设为 3，否则函数会返回错误。

返回值：

0 表示没有错误

更新记录

时间	更改内容	更改人
2015.3.31	初次发布	ZCD
2015.5.20	添加模拟外触发使用条件	ZCD
2016.3.8	完善 ZTKUSB_SFifoLostBC 函数	ZCD
2017.7.26	修正函数说明错误	ZCD