USB302 光栅尺转接器使用说明

东莞万濠精密仪器有限公司
版　本 ：Ver 1.00
电　话 ：0769-85330109
传　真 ：0769-85330106
日　期： 2007-12-05

1 简介

USB302 是一个基于 USB 的光栅尺数据读取与传输设备。可接三个光栅尺和一个脚踏开关。脚踏开关按下时或光栅尺 RI 点发现时，会锁定该时刻的三轴数据。

图：USB302 正面

USB302 被定义为 HID 设备，无需驱动，即插即用。接入电脑后，LINK 灯会亮，表示设备准备好。当有数据传输时 LINK 灯闪。设备采用电脑 USB 总线供电,无需外加电源.

USB-302 特点：

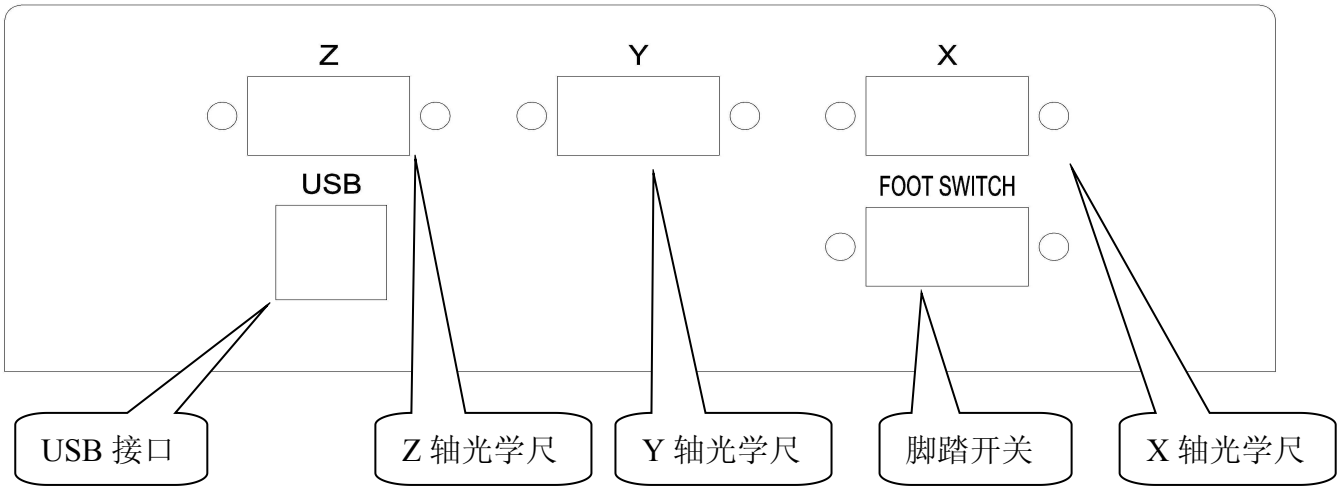
- 1：独立 3 轴 24 位可逆计数器；
- 2：在外部触发时，三轴计数器同时锁存，三轴锁存时延小于 20ns；
- 3：1.0MHZ（PC-AT）四裂相输入；
- 4：每轴 A，B 相信号支持单端输入或差动输入；
- 5：搜索光学尺 RI 点时，RI 信号，A 相信号，B 相信号极性可选择，
- 6：计数方向的正负可控制；
- 7：四阶数字滤波，可处理不正常信号；
- 8：2 组独立外部触发，可对任一轴锁存计数值；

USB-302 应用范围

- 影像测量仪
- 运动控制
- 位置监测
- 三坐标量床
- X-Y 工作平台监视器
- 机械控制
- 其它光栅尺，旋转编码器的应用场合

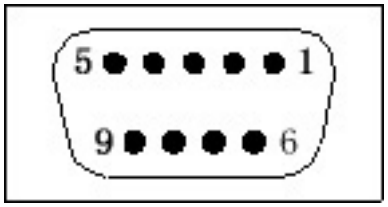
安装及接口定义

各接口说明



图：USB302 背面

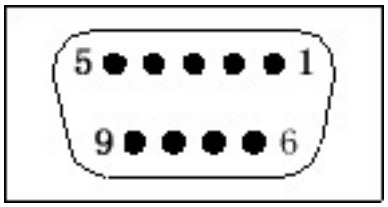
光栅尺接口 9P 双排针母座



脚号	信号名称	说明
1	VCC	+5V
2	GND	0V
3	A+	
4	B+	
5	RI+	
6	RI-	
7	A-	
8	B-	
9		接外壳

注：右表引脚定义是差分接法；如果为单端接法，则 6、7、8 脚不接

脚踏开关 9P 双排针母座



脚号	信号名称	说明
1	SW	接脚踏开关
2	SW	接脚踏开关
3	NG	空
4	NG	
5	NG	
6	NG	
7	NG	
8	NG	
9	NG	

3 函数说明

USB302 提供了 HID 设备的底层通讯接口 RationalUsbHid.dll 完成设备控制及数据交换，功能接口 RationalScale.dll 与上层程序交互。应用程序只需调用 RationalScaleScale.dll 的函数。

以下是 RationalScale.dll 函数说明

USHORT ConnectUSB302(void);

功能：找到USB302设备

返回：找到USB302，返回 0 否则返回　　ERR_CONNECT。

说明：此函数必须先于其它函数调用，找到设备后才能进行其它操作。

USHORT GetInfo(char *Manufacturer,char *Producer,char *SerialNumber,char *BufLength);

功能：读取找到的设备信息

参数：char *Manufacturer 厂商字符串，一般为： Rational

char *Producer 产品字符串，一般为： USB302

char *SerialNumber 产品序列号，一般为：XXXX-XXXX 即 “硬件版本号”-“软件版本号”

char *BufLength 设备缓冲区长度

USHORT DevHotPlug(void);

功能：软件热拔插

返回：正常返回 0 否则返回 ERR_HOTPLUG

说明：此函数将重新载入系统的USB驱动程序。

USHORT GetErrCode(char *ErrStr);

功能：获取错误代码，得到底层定义的错误号

参数：char *ErrStr 底层错误代号是一五位数，以字符串的形式返回。当发生错误的时候可调用此函数了解发生错误的具体信息

USHORT PopErrMsg(bool Enable);

功能：出错时是否弹出消息框

参数：bool Enable true 则弹出消息否则不弹出

说明：此消息框是底层弹出的，在多线程中应禁用

USHORT UsbLinearScaleEx(char Option,char Axis,PWRITEDEV WriteDevice,PREADDEV ReadDevice,void *IntStatus,void *Reserved);

功能：完成USB303的设置、读数、采点等功能

参数：Option ，　指定对USB302的命令类别

Axis，　　指定操作轴，可选 X_AXIS | Y_AXIS | Z_AXIS | XYZ_AXIS

WRITEDEV： PC端输出数据的结构体，定义如下

```
typedef struct
{
 long lValue;
 unsigned char cValue;
 }WRITEDEV,*PWRITEDEV;
```

READDEV：来自USB设备的输入数据的结构体，定义如下

```
typedef struct
{
 long lValue[3];
 unsigned char cValue[3];
}READDEV,*PREADDEV;
```

*IntStatus：中断状态值

*Reserved： 保留

以下是对各参数的详细说明及例子，例子的函数参数的声明如下

```
WRITEDEV Write;
READDEV Read;
unsigned char IntStatus;
unsigned char Reserved;
```

option：功能命令字

Option=INTENABLE 开放某中断源

Option=INTDISABLE 关闭某中断

Write.CValue= ***X_RI | Y_RI | Z_RI | XYZ_RI | X_EQU | Y_EQU | Z_EQU | XYZ_EQU | Int_Ext0 | Int_Ext1 | Int_Ext01 | Int_ALL***

以上两条命令对轴无效，在 Write.cValue 指定要打开或关闭的中断

X_RI	X轴RI中断
Y_RI	Y轴RI中断
Z_RI	Z轴RI中断
XYZ_RI	XYZ轴RI中断
X_EQU	X轴比较中断
Y_EQU	Y轴比较中断
Z_EQU	Z轴比较中断
XYZ_EQU	XYZ轴比较中断
Int_Ext0	EXT0中断
Int_Ext1	EXT1中断
Int_Ext01	EXT中断
Int_ALL	所有中断

例如：开X轴RI中断

```
Write.cValue=X_RI;
```

UsbLinearScaleEx(INTENABLE,X_AXIS,&Write,&Read,&IntStatus,&Reserved);

发生RI中断时将会锁存RI时刻的数据，并回传至PC。为了避免重复定位参考点的现象，USB303发现RI点后将会自动关闭RI中断。要想再找RI点请将RI中断再次打开。

Option=ZEROCOUNTER　　指定轴清零

在 Axis 中指定要清零的轴， Axis= X_AXIS|Y_AXIS|Z_AXIS|XYZ_AXIS

对X轴清零　UsbLinearScaleEx(ZEROCOUNTER,X_AXIS,&Write,&Read,&IntStatus,&Reserved);

XYZ三轴清零　UsbLinearScaleEx(ZEROCOUNTER,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved);

Option = PRESETCOUNT　　向指定的轴预置COUNT值

在 Axis 中指定要预置的轴，Axis=X_AXIS|Y_AXIS|Z_AXIS|XYZ_AXIS

在Write.lValue中指定要预置的数。例如，Z轴预置1000：

Write.lValue=1000;

UsbLinearScaleEx(PRESETCOUNT,Z_AXIS,&Write,&Read,&IntStatus,&Reserved);

注：当Axis= XYZ_AXIS 时，是把要预置的数同时置到三个轴中，这是三轴中的数据是一样的，如果要置三个轴的数据不同，必须分别给每一个轴置数。

Option = ENABLERIBELL　　使能光栅尺找到RI点时蜂鸣器响

在 Axis 中指定要操作的轴，Axis=X_AXIS|Y_AXIS|Z_AXIS|XYZ_AXIS(此操作与设置某一轴无关)

Write.cValue = 1 | 0

1　光栅尺找到 RI 点时会响

0　光栅尺找到 RI 点时不会响

例如：设置光栅尺找到 RI 点时会响

Write.cValue = 1;

UsbLinearScaleEx(SETDIRECT,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserve);

Option= EXT0LATCHENABLE　　EXT0中断锁存允许

Option= EXT1LATCHENABLE　　EXT1中断锁存允许

Option= EXT0LATCHDISABLE　　禁止EXT0中断

Option= EXT1LATCHDISABLE　　禁止EXT1中断

Option= RILATCHENABLE　　RI中断锁存允许

以上5个命令　在 Axis 中指定要操作的轴，Axis=X_AXIS|Y_AXIS|Z_AXIS|XYZ_AXIS

例如：　X 轴 EXT0 锁存允许

UsbLinearScaleEx(EXT0LATCHENABLE,X_AXIS,&Write,&Read,&IntStatus,&Reserved)

Y轴的RI中断锁存允许

UsbLinearScaleEx(RILATCHENABLE,Y_AXIS, &Write,&Read,&IntStatus,&Reserved);

Option= RILATCHDISABLE　　禁止RI中断锁存

无须其它参数

例如：　RI 锁存禁止

UsbLinearScaleEx(RILATCHDISABLE,X_AXIS, &Write,&Read,&IntStatus,&Reserved);

说明： 只要发送此命令同时禁止三轴RI中断锁存。

Option = SETDIRECT 设置记数方向

在 Axis 中指定要操作的轴，Axis=X_AXIS|Y_AXIS|Z_AXIS

Write.CValue = 0 | 1

0 A 相超前 B 相时 COUNT 值增加

1 A 相落后 B 相时 COUNT 值增加

例如：设置 Y 轴 A 相落后 B 相时，COUTN 值增加

```
Write.CValue = 1;
UsbLinearScaleEx(SETDIRECT,Y_AXIS,&Write,&Read,&IntStatus,&Reserve);
```

Option = SETEXT0POL 设置EXT0的触发电平

Option = SETEXT1POL 设置EXT1的触发电平

在 Axis 中指定要操作的轴，Axis=X_AXIS|Y_AXIS|Z_AXIS

Write.CValue = 0 | 1

0 下降沿有效， 1 上升沿有效

例如：设置 EXT0 中断触发下降沿有效

```
Write.CValue = 0;
UsbLinearScaleEx(SETEXT0POL,X_AXIS,&Write,&Read,&IntStatus,&Reserve);
UsbLinearScaleEx(SETEXT0POL,Y_AXIS,&Write,&Read,&IntStatus,&Reserve);
UsbLinearScaleEx(SETEXT0POL,Z_AXIS,&Write,&Read,&IntStatus,&Reserve);
```

注： 设置外部触发方式时，必须对每一轴单独设置，Axis参数不能设为XYZ_AXIS。

参考程序如下：

```
if(PolEXT0)       // PolEXT0=1表示EXT0上升沿有效
{
   Write.cValue=1;
   Result=UsbLinearScaleEx(SETEXT0POL,X_AXIS,&Write,&Read,&IntStatus,&Reserved);
   if(Result!=0)
   {
        //设置不成功处理
         return;
   }
   else
   {
         //设置成功处理
   }
   Result=UsbLinearScaleEx(SETEXT0POL,Y_AXIS,&Write,&Read,&IntStatus,&Reserved);
   if(Result!=0)
   {
        //设置不成功处理
        return;
   }
   else
   {
        //设置成功处理
   }
   Result=UsbLinearScaleEx(SETEXT0POL,Z_AXIS,&Write,&Read,&IntStatus,&Reserved);
```

```
    if(Result!=0)
    {
        //设置不成功处理
         return;
    }
    else
    {
         //设置成功处理
    }
}
Else     // PolEXT0=0表示EXT0下降沿有效
{
    Write.cValue=0;
    同以上处理
}
```

Option = SETRIMODE　　设置RI模式

在 Axis 中指定要操作的轴，Axis=X_AXIS | Y_AXIS|Z_AXIS|XYZ_AXIS

在Write.cValue中设置 RI模式， RI模式是三位（0-7）A、B、RI各占一位，可单独设定A、B、RI的电平来设置RI模式 。RI=A+2*B+4*RI

例如，设定 A相高电平，B相低电平，RI相高电平时发生RI中断。即A=1，B=0，RI=1。则RI模式

RIMode=1+2*0+4*1=5

所以 Write.cValue = 5;

UsbLinearScaleEx(SETRIMODE,X_AXIS, &Write, &Read, &IntStatus, &Reserved);

Option = GETINTSTATUS　　获取中断状态

在 Axis 中指定要操作的轴，Axis=X_AXIS | Y_AXIS|Z_AXIS，此参数可随便取一个，对结果无影响

Read.cValue[0]中是获取的中断状态值。Read.cValue[0]每位的意义如下：

B7	B6	B5	B4	B3	B2	B1	B0
Ext1_Int	Ext0_Int	Z_EQU	Z_RI	Y_EQU	Y_RI	X_EQU	X_RI

当某位为'1'时，表示某种中断发生

Ext1_Int:　　外部触发中断 1

Ext0_Int:　　外部触发中断 0

Z_EQU:　　Z 轴比较中断

Y_EQU:　　Y 轴比较中断

X_EQU:　　X 轴比较中断

Z_RI　　Z 轴 RI 中断:

Y_RI:　　Y 轴 RI 中断

X_RI:　　X 轴 RI 中断

进行一次读操作后该寄存器清零。

例如：得到中断状态值

UsbLinearScaleEx(GETINTSTATUS,X_AXIS, &Write,&Read,&IntStatus,&Reserved);

判断 Read.cValue[0] 的值再做处理

Option = GETSTATUS　读某轴光栅尺状态

Axis 指定要读取的轴，Read.lValue 返回状态值。当读取三轴时 lValue[0]|lValue[1]|lValue[2] 分别对应 X|Y|Z 三轴，当读取一个轴时状态值在 lValue[0] 中。

例如：获得X轴光学尺的状态

UsbLinearScaleEx(GETSTATUS,X_AXIS,&Write,&Read,&IntStatus,&Reserved);

Read.cValue[0]每位的意义如下：

B7	B6	B5	B4	B3	B2	B1	B0
X_Err2	X_Err1	X_UpDn	X_UFL	X_OFL	X_RI	X_B	X_A

X_Err2： 1：Error2 发生； 0：无 Error2 发生

Error2： A 相信号 B 相信号同步变化

X_Err1： 1：Error1 发生； 0：无 Error1 发生

Error1： A 相信号，B 相信号频率太高

X_UpDn： 1：X 轴 COUNT 值增加 0：X 轴 COUNT 值减少

X_UFL： 1：X 轴 COUNT 值 发生 0000 –> FFFFFF 跳变

0：没发生

X_OFL： 1：X 轴 COUNT 值 发生 FFFFFF->0000 跳变

0：没发生

X_RI： X 轴 RI 信号当前电平

X_B： X 轴 B 相信号当前电平

X_A： X 轴 A 相信号当前电平

Option = INITIAL 初始化读数

无需其它参数 例如

UsbLinearScaleEx(INITIAL,XYZ_AXIS,&Write,&Read,IntStatus,&Reserved);

初始化包括以下内容：

1：设置基地址；

2：禁止中断；

3：RI mode = RI_LOW + RI_A_LOW + RI_B_LOW;

4: X 轴，Y 轴，Z 轴计数方向 NORMAL;

5: Ext0, Ext1 触发极性为负边缘触发；

6：X 轴，Y 轴，Z 轴 COUNT 清零；

7：关闭 COUNT 比较功能；

8：Ext0,Ext1,RI 处于不锁存状态；

Option = READNOLATCH 读指定轴的以前锁存的COUNT值

该命令主要用于以前锁存的COUNT值，该命令不回送中断状态及中断时的锁存值

Axis 指定要读取的轴 Axis=X_AXIS | Y_AXIS|Z_AXIS| XYZ_AXIS;

读取的数据在 Read.lValue 中，X轴数据存在lValue[0]，Y轴数据存在lValue[1]，Z轴数据存在lValue[2]|Z轴。例如：

UsbLinearScaleEx(READNOLATCH,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved);

读数结果：Read.lValue[0] X 轴COUNT值

Read.lValue[1] Y 轴COUNT值

Read.lValue[2] Z 轴COUNT值

Option = READCOUNTER　读光栅尺COUNT值

Axis指定要读数的轴Axis=X_AXIS | Y_AXIS|Z_AXIS| XYZ_AXIS，例如以上读三轴数据。

读数结果：Read.lValue[0] X 轴COUNT值，Read.lValue[1] Y 轴COUNT值，Read.lValue[2] Z 轴COUNT值

没有中断发生时，读的是当前位置的COUNT值

当中断发生时IntStatus存的是中断时刻的中断状态寄存器的值。中断状态寄存器有以下数值：

0x01　//X 轴 RI 中断

0x04　//Y 轴 RI 中断

0x10　//Z 轴 RI 中断

0x20　//Z 轴 EQU 中断

0x40　//EXT0 中断（USB302表示脚踏开关按下）

当IntStatus是以下数据时，标志 Y 则表示此次脚踏开关按下有效， N 则无效

'Y'　//脚踏开关按下检测有效

'N'　//脚踏开关按下检测无效

例如读三轴数据：

```
UsbLinearScale(READCOUNTER,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved);
switch(IntStatus)
 {
    case 0x01:
              //X 轴RI中断Read.lValue[0] 是X 轴COUNT值，数据处理
      break;
   case 0x02: //X 轴比较中断
      break;
   case 0x04:
              //Y 轴RI中断Read.lValue[1] 是X 轴COUNT值，数据处理
      break;
   case 0x08: //Y 轴比较中断
      break;
   case 0x10:
              //Z 轴RI中断Read.lValue[2] 是X 轴COUNT值，数据处理
      break;
   case 0x20: //Z 轴比较中断
      break;
   case 0x40:
              //EXT0中断，数据处理
      break;
   case 'Y':
             //脚踏开关按下有效，数据处理
      break;
   case 'N'
             //脚踏开关按下无效
      break;
   case 0x80://EXT1
      break;
}
```

注意：读数函数必须开线程，不要开定时器，因为此函数执行与硬件有关，每次执行的时间可能不同

Option = GETRIDATA　　//从USB302获得X、Y、Z三轴RI中断时的COUNT值

无需其它参数 例如

UsbLinearScaleEx(GETRIDATA,XYZ_AXIS,&Write,&Read,IntStatus,&Reserved);

当返回IntStatus的值为0x88时：则Read.lValue[0]中保存X轴RI中断时COUNT值，Read.lValue[1] Y轴RI中断时COUNT值，Read.lValue[2] Z 轴RI中断时COUNT值；

当返回IntStatus的值为0x55时，表示USB302还没有找过RI，或者USB302已断过电，则测量软件必须找RI。

Option = SETDIRTIME　　//设置从触发起采下方向点的时间 (此命令USB302可以不用)

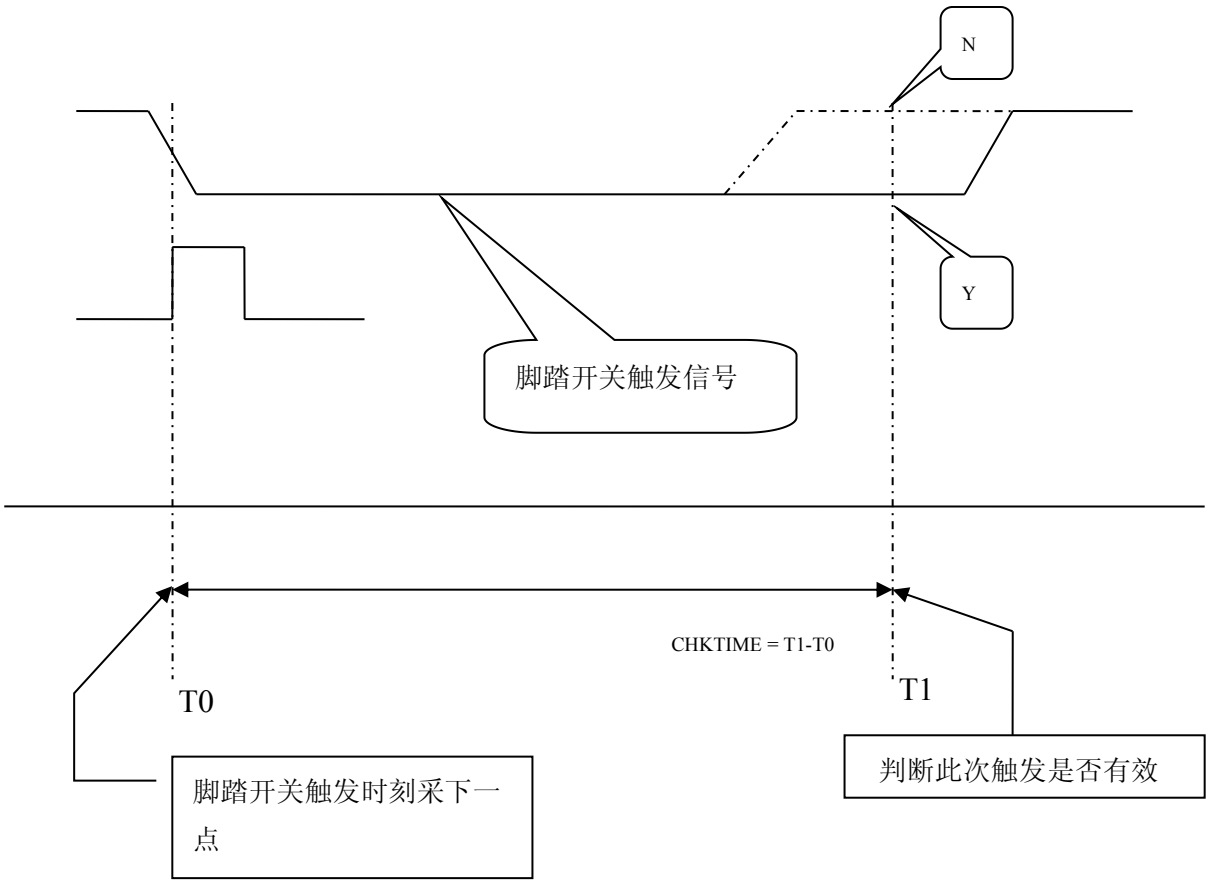
***Option = SETCHKTIME　　//设置从触发起检测触发的时间

详细解释如下图所示，

一次脚踏开关按下时间太短,可认为是一次误按,SETCHECKTIME设置脚踏开关按下时间的最小值。如果小于该值,USB302会认为是一次误按。参数Axis可以随便设一个值，无须理会。例如：

Write.cValue = 10;　　// 设置最小触发时间为100ms（此时间参数需要设置合适，不然会出现漏采点现象，此参数最好为以下值8、9、10（10mS为单位））

UsbLinearScale(SETCHECKTIME,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved);



当脚踏开关按下时USB302立刻采点，到了CHKTIME再判断此次触发是否有效。脚踏开关按下过程如上图

PC端收到的数中断状态先是 EXT0（脚踏开关按下时采点数据） 然后是 Y/N （Y 则判断此次触发有效，N 则无效）——应用软件要是用到USB302的脚踏开关采点，必须按此处理，不然会出现多采点现象。

UsbLinearScaleEx 函数执行成功时返回0，否则返回错误代号

ERR_WRITE	写错误
ERR_READ	读错误
ERR_DEVICE	硬件无法响应
ERR_OPERATION	不支持此操作
ERR_TIMEOUT	设备忙
WAITTIME	排队等待时间

4 实例

使用USB302先要初始化设置一些必要的参数，然后再读数
一般的初始化流程：

1、初始化
1、设置三轴的记数方向
2、设置EXT0，EXT1的触发极性
3、设置RI模式
4、设置EXT0、EXT1、RI锁存
5、开启需要的中断
6、设置从采下方向点的时间及触发检测的时间

初始化的实例：

```
UsbLinearScaleEx(INITIAL,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved);      // 初始化USB302

Write.cValue=0;           //设置X、Y轴记数方向为正(光栅尺A相超前B相COUNT增)
UsbLinearScaleEx(SETDIRECT,X_AXIS,&Write,&Read,&IntStatus,&Reserved);
UsbLinearScaleEx(SETDIRECT,Y_AXIS,&Write,&Read,&IntStatus,&Reserved);

Write.cValue=1;           //设置Z轴记数方向为反(光栅尺A相落后B相COUNT增)
UsbLinearScaleEx(SETDIRECT,Z_AXIS,&Write,&Read,&IntStatus,&Reserved);

Write.cValue=0;           //设置EXT0上升沿触发
UsbLinearScaleEx(SETEXT0POL,X_AXIS,&Write,&Read,&IntStatus,&Reserved);
UsbLinearScaleEx(SETEXT0POL,Y_AXIS,&Write,&Read,&IntStatus,&Reserved);
UsbLinearScaleEx(SETEXT0POL,Z_AXIS,&Write,&Read,&IntStatus,&Reserved);

Write.cValue=0;           //设置EXT1底电平触发
UsbLinearScaleEx(SETEXT1POL,X_AXIS,&Write,&Read,&IntStatus,&Reserved);
UsbLinearScaleEx(SETEXT1POL,Y_AXIS,&Write,&Read,&IntStatus,&Reserved);
UsbLinearScaleEx(SETEXT1POL,Z_AXIS,&Write,&Read,&IntStatus,&Reserved);
Write.cValue=7;           //当A，B，RI均为高时，找到光栅尺参考点
```

```
UsbLinearScaleEx(SETRIMODE,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved);

UsbLinearScaleEx(RILATCHENABLE,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved);    //允许RI发生时锁存COUNT
UsbLinearScaleEx(EXT0LATCHENABLE,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved); //允许EXT0发生时锁存COUNT
UsbLinearScaleEx(EXT1LATCHENABLE,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved); //允许EXT1发生时锁存COUNT


Write.cValue=X_RI;              //开启X轴RI中断
UsbLinearScaleEx(INTENABLE,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved);
Write.cValue=Y_RI;               //开启Y轴RI中断
UsbLinearScaleEx(INTENABLE,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved);
Write.cValue=Z_RI;              //开启Z轴RI中断
UsbLinearScaleEx(INTENABLE,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved);
Write.cValue=Int_Ext0;          //开启EXT0中断
UsbLinearScaleEx(INTENABLE,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved);
Write.cValue=Int_Ext1;          //关闭EXT1中断
UsbLinearScaleEx(INTDISABLE,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved);

Write.cValue=DirChkTime;        //设置采样点和方向点的时间间隔
UsbLinearScaleEx(SETDIRTIME,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved);
Write.cValue=ProbeChkTime;      //设置触头最小解触发时间
UsbLinearScaleEx(SETCHKTIME,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved);
```

找 RI 点，打开 USB302 RI中断及 RI 中断锁存功能即可移动光学尺找RI点。

找 X 轴 RI 的实例：

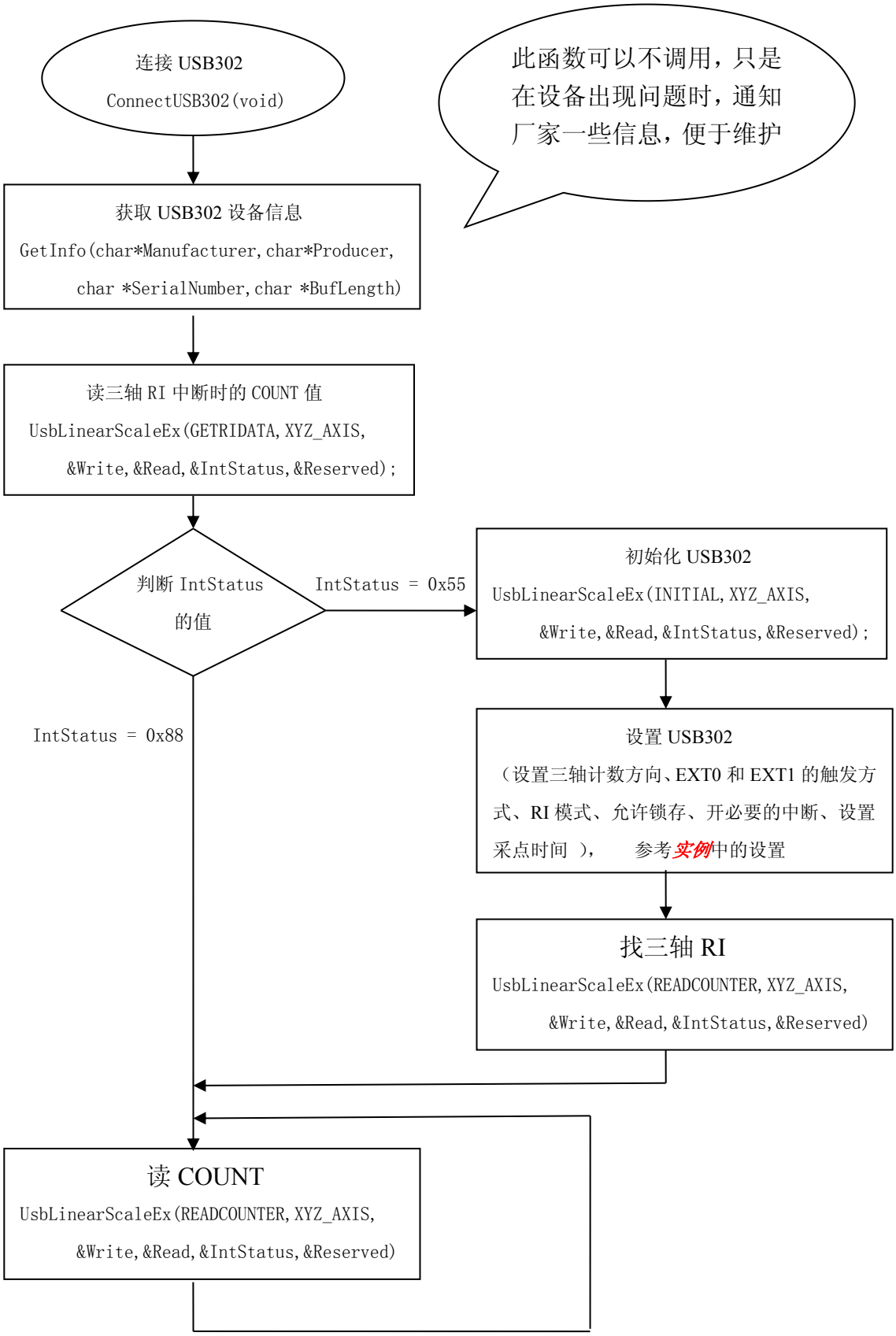
```
Write.cValue=0;             //设置X轴记数方向为正(光学尺A相超前B相COUNT增)
UsbLinearScaleEx(SETDIRECT,X_AXIS,&Write,&Read,&IntStatus,&Reserved);
Write.cValue=7;             //当A，B，RI均为高时，找到光学尺参考点
UsbLinearScaleEx(SETRIMODE,X_AXIS,&Write,&Read,&IntStatus,&Reserved);
                            //设置X轴RI锁存
UsbLinearScaleEx(RILATCHENABLE,X_AXIS,&Write,&Read,&IntStatus,&Reserved);
Write.cValue=X_RI;          //开启X轴RI中断
UsbLinearScaleEx(INTENABLE,X_AXIS,&Write,&Read,&IntStatus,&Reserved);

UsbLinearScale(READCOUNTER,XYZ_AXIS,&Write,&Read,&IntStatus,&Reserved); //读光学尺COUNT
```

再判断IntStatus的值是否为0x01,如果是Read.lValue[0]是X轴RI中断时COUNT值。

5 使用 USB302 开发一般流程

使用 USB302 开发一般流程如下：



6 注意事项

USB302找到RI点后会自动关闭该轴的RI中断，即每一轴只找一次RI，要想重新找RI，必须重新开该轴RI中断。

USB302接入电脑后LINK灯不亮，在设备管理器上看不到新增的HID设备。此种情况很可能是USB线与设备或电脑接触不良，需要更换USB线。也有可能是电脑的USB驱动程序文件丢失或被破坏，此种情况请修复系统文件或重新安装操作系统。