TDC-GP2 在时差法（TOF）脉冲式激光测距中的应用

摘要：
在脉冲式激光测距仪的设计当中，时差测量（time of flight measurement）成为了一个影响整个测量精度最关键的因素。德国 acam 公司设计的时间数字转换芯片 TDC-GP2 为激光测距的时间测量提供了完美的解决方法。本文着重介绍了应用 TDC-GP2 在设计激光测距电路当中的优势，以及在应用中给出一些建议和提出了需要注意的一些问题。

1. 概述

在当今这个科技发达的社会，激光测距的应用越来越普遍。在很多领域，电力，水利，通讯，环境，建筑，地质，警务，消防，爆破，航海，铁路，反恐/军事，农业，林业，房地产，休闲/户外运动等都可以用到激光测距仪。激光测距仪一般采用两种方式来测量距离：**脉冲法和相位法**

脉冲式激光测距仪是通过测量激光从发射到返回之间的时间来计算距离的。因此时间测量对于脉冲式激光测距仪来说是非常重要的一个环节。由于激光的速度特别快，所以发射和接收到的激光脉冲之间的时间间隔非常小。。例如要测量 1 公里的距离，分辨率要求 1cm，则时间间隔测量的分辨率则要求高达 67ps。德国 acam 公司的时间数字转换器 TDC-GP2 单次测量分辨率为典型 65ps，功耗超低，集成度高，测量灵活性高，是脉冲式激光测距仪时差（TOF）测量非常理想的选择。

2. TDC-GP2 激光测距原理

TDC-GP2 的激光测距基本原理如图 1 所示：

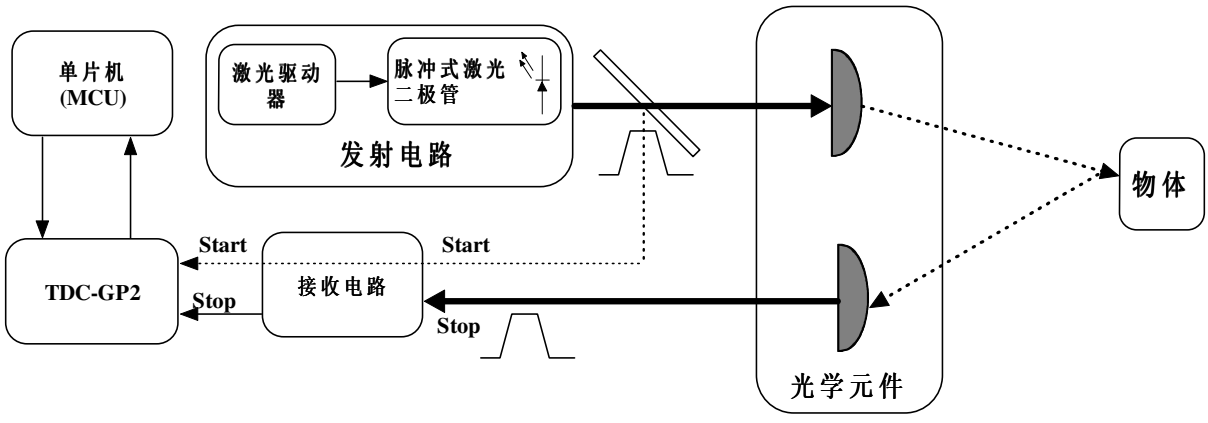


图 1：TDC-GP2 激光测距原理

激光发射装置发射出光脉冲同时将发射脉冲输入到 TDC-GP2 的 start 端口，触发时差测量。 一旦从物体传回的反射脉冲达到了光电探测器（接收电路）则给 TDC 产生一个 Stop 信号，这个时候时差测量完成。那么从 Start 到 Stop 脉冲之间的时差被 TDC-GP2 精确记录下来，用于计算所测物体与发射端的距离。在这个原理中，单片机对于 TDC-GP2 进行寄存器配置以及时间测量控制，时间测量结果传回给单片机通过算法进行距

离的精确计算，同时如果有显示装置的话，将距离显示出来。在这个原理当中距离的测量除了与 TDC-GP2 的时差测量精度有关外还与很多其他因素有关系：

- 激光峰值功率
- 激光束发散程度
- 光学元件部分
- 光传输的媒体 (空气，雨天，雾天等)
- 物体的光反射能力
- 光接收部分的灵敏程度等等

被测物体特性以及传输媒介的铁性一般是由应用的条件给出的，那么可以根据应用的条件来选择激光的发射器（波长，驱动条件，光束的特性等）和接收器（类型，灵敏度，带宽等）。测量的范围在激光峰值功率更高以及信噪比更高的情况下也会相应增加。那么时差测量的精度除了与 TDC-GP2 芯片本身测量精度有关系外还与激光的脉冲特性有关，比如脉冲的形状（宽度，上升下降沿的时间），以及探测器带宽和信号处理电路。对于 tdc-gp2 而言，脉冲信号的速度越快，带宽越宽，则测量精度相应得会越高。

那么上面所述的一些需要注意的问题在这里我们并不做讨论，我们假设其他方面都已经解决，那么这里我们着重介绍一下如何应用单片机和 TDC-GP2 来控制时间测量。

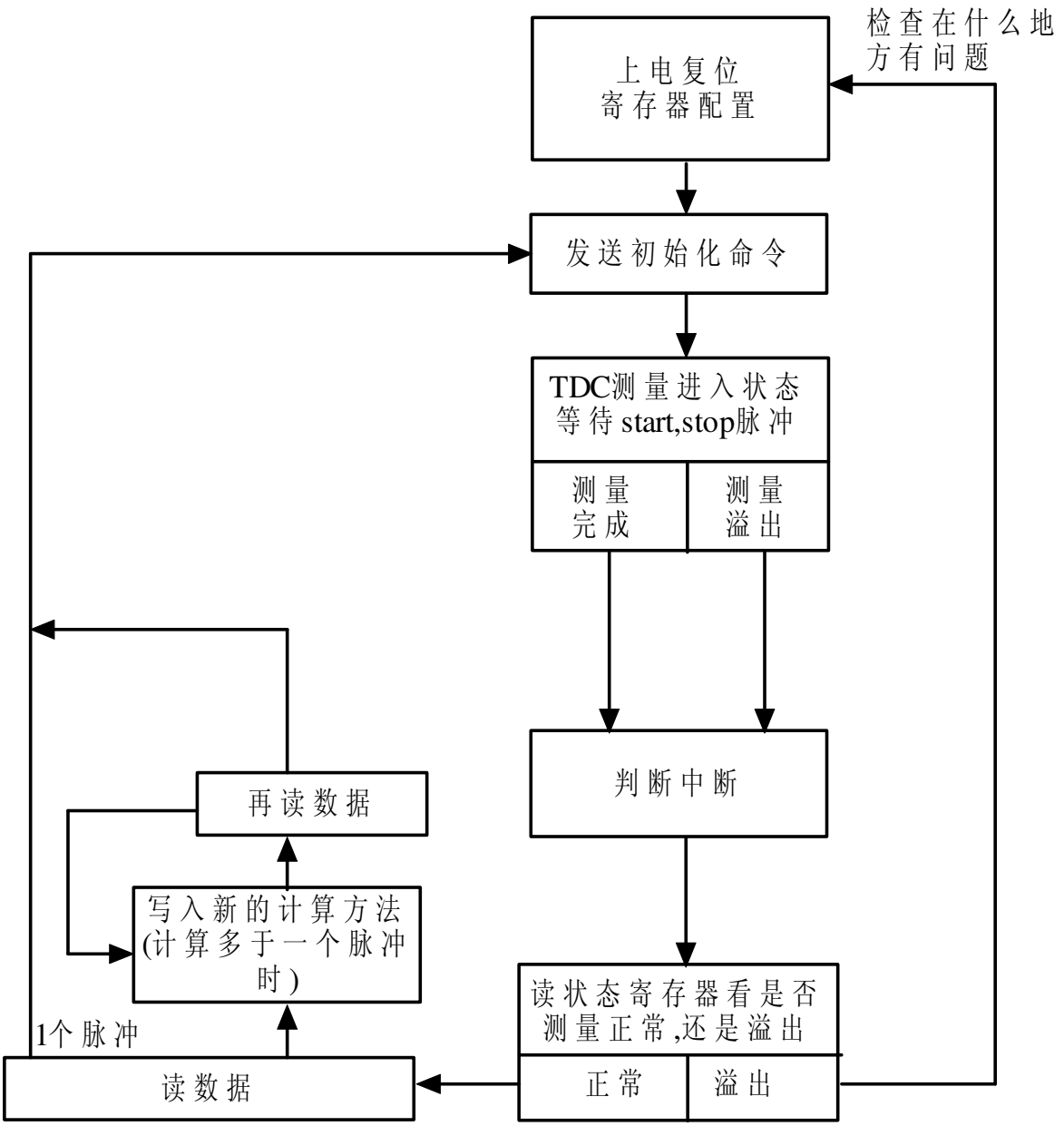
对于 tdc-gp2 而言，这颗芯片本身有两个测量范围，测量范围 1 和测量范围 2。
测量范围 1 的时间测量从 0ps-1.8us,相对于距离来讲大约为 0-270m。测量范围 2 的测量范围从 2 倍的高速时钟周期到 4ms.也就是说最高的距离测量可以到 25 公里以外.那么我们下面就以不同的测量范围来进行介绍.

测量范围 1:
0ps-1.8us

在这个测量范围下,TDC-GP2 芯片的测量工作全部是由 TDC 高速测量单元完成的。在这个测量范围中，gp2 的 start 通道，stop1，stop2 通道都可用。每个 stop 通道有 4 个脉冲的测量能力。在这个测量范围下，测量结果可以选择校准结果（32 位）或者非校准结果 16 位。推荐使用 32 位的校准结果，也就是每次测量都对 TDC 测量单元进行一次校准。

需要引起注意的问题：
- 对于 TDC-GP2 来讲触发它的脉冲宽度必须要大于 2.5ns。
- 在 start 通道的触发边沿与第一个 stop 通道的脉冲边沿之间的时间间隔要大于 3.5ns。
- 推荐自动校准结果，并且选择每次测量完成后进行自动校准。 这个功能通过设置寄存器 0 的自动校准位为 0 来开启。
- 如果计算 stop1 和 stop2 通道的脉冲时差的话，脉冲的时差范围可以降低到 0。Start 到最后一个 stop 脉冲的距离不能够超过 1.8us,这是由于硬件本身所限制的。

在这个测量模式下测量流程以及典型的寄存器设置如下：



单片机与 tdc-gp2 的通信是通过 spi 串口完成的，那么对于测量范围 1 的一个典型的测量过程为：

```
void gp2config()
{
SPIwrite8 (0x50); //上电复位

//配置寄存器：
SPIwrite32 (0x80000420) ; // 选择测量范围 1，自动校准，晶振上电后一直起振。
SPIwrite32 (0x81014100) ;// stop1 接受 1 个脉冲，定义计算方法，用 stop1 通道的第一个脉冲减去 start 脉冲
SPIwrite32 (0x82E00000) ; // 开启所有中断源
SPIwrite32 (0x83000000) ;
SPIwrite32 (0x84200000) ;
SPIwrite32 (0x85080000) ;
}
```

```
//测量循环：
void measurement()
{
SPIwrite8 (0x70) ;//初始化测量，通知 gp2 进入测量准备状态

Check INTN=0?
SPIwrite32 (0xb4) ; //发送命令读状态寄存器
SPIread8 (STAT) ;
STAT&0x0600>0？ //=>说明有测量溢出，有问题
SPIwrite32(0xb0) ; //发送命令读 reg0 结果
SPIread32(reg0) ;
}
```

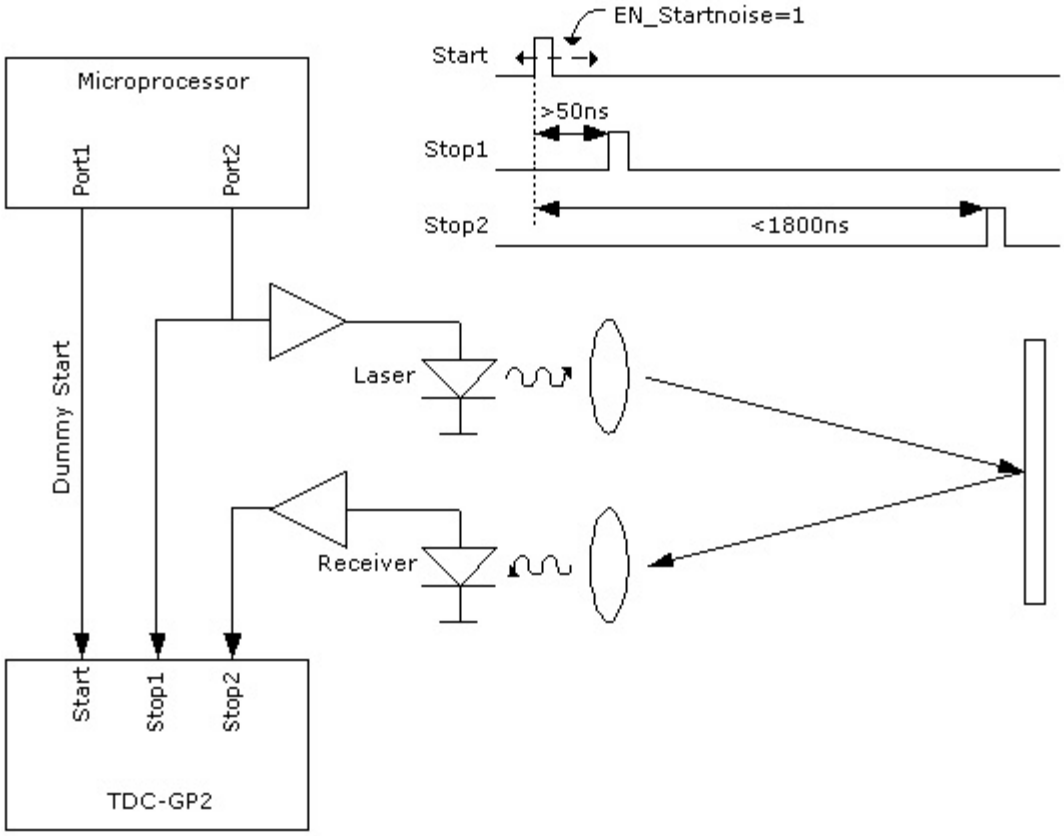
那么单片机在从 gp2 读取完数据之后，可以对数据进行处理，来计算脉冲来回的距离。
在上面的测量过程中如果 gp2 在被初始化之后，并没有接受到任何 start 信号，测量将不会发生。也不会产生中断。只有 start 信号被接受后，测量才被触发，那么无论是测量正常还是在规定时间内没有接受到 stop 脉冲，在 gp2 的 INTN 管脚都会有中断信号产生，通过判断状态寄存器的内容来判断测量是否正常。

注：在接受 start,stop 脉冲之前，必须要将 gp2 的管脚 en_start，en_stop 置高平，否则 start,stop 通道则不会被选通，测量也不会被触发！！

应用平均提高精度的方法：

上面所说的情况为，你的激光 start 脉冲给 tdc-gp2 的 start 通道，激光的返回脉冲给 tdc-gp2 的 stop 通道的情况。

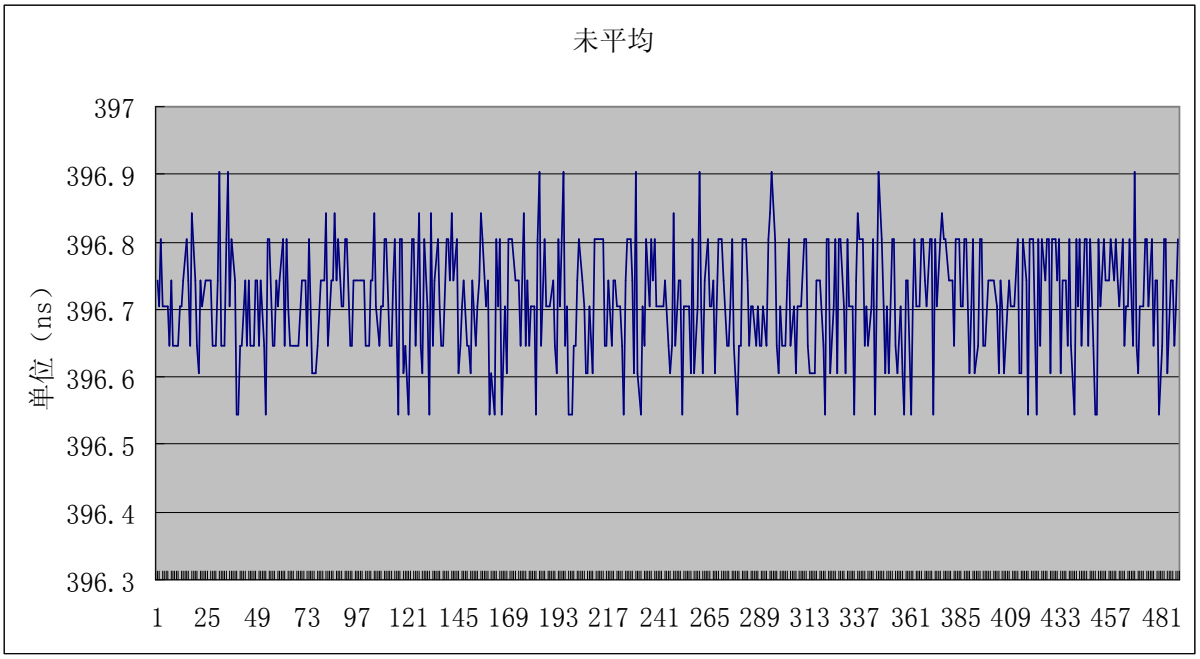
在这种情况下，gp2 的单次测量精度为 65ps。当测量的输出频率并不是非常重要的情况下，比如每秒钟输出 1 到 2 次结果，那么这个时候为了提高测量精度，我们可以通过多次测量平均的方法来消除系统误差。为了使 gp2 能够通过平均的方法来大大的减少误差，那么下面推荐的测量设计是非常有效的，可以将系统误差的**峰峰值**降低到 10ps 一下。如下图所示：



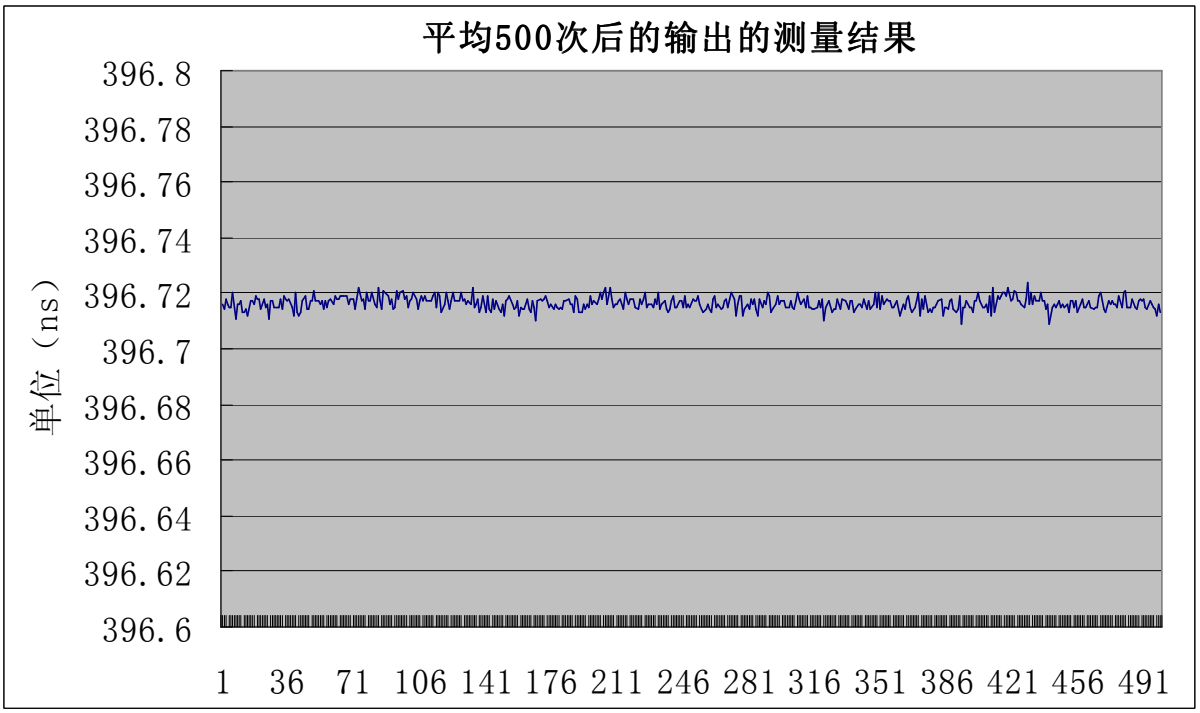
如上图所示，在这个情况下我们使用的是测量范围 1，激光的发射和接收脉冲信号是给到 stop1 和 stop2 的，而在 tdc-gp2 的 start 通道，start 信号是由单片机给出的一个不参与测量的 start 信号。测量过程如下：
首先由单片机发出一个不参与测量的但是要触发测量用的 dummy start.需要这个信号是因为 start 通道的这个信号是告诉 gp2 现在开始进入测量状态了。那么在至少 50ns 后，单片机触发激光器产生发射信号同时将这个信号输入到 stop1 通道。那么接收到的 laser 脉冲信号则输入到 stop2 通道。也就是说用 stop1 和 stop2 来测量激光发射和接收的时间差，而 start 信号是由单片机给出来触发 gp2 的。

那么之所以这样的原因是在 tdc-gp2 的内部，有一个噪声单元，通过寄存器设置可以触发这个噪声单元。噪声单元将会在 gp2 的 start 通道脉冲上加任意分布噪声，那么这样做的目的是为了在平均的时候，可以大大消除量化误差和系统误差。那么这个一位的设置为寄存器 5 中的 EN_STARTNOISE 设置。

没有平均的情况下：



应用平均的情况下：



这样做的好处为：

1.stop1 和 stop2 的时间间隔测量可以最低到 0。

2.通过这个测量之后如果平均 gp2 的测量结果，可以大大消除系统误差，跟据平均的次数不同，最多可以使 gp2 的精度提高至小于 6ps.

3.对于温度变换是相当稳定的

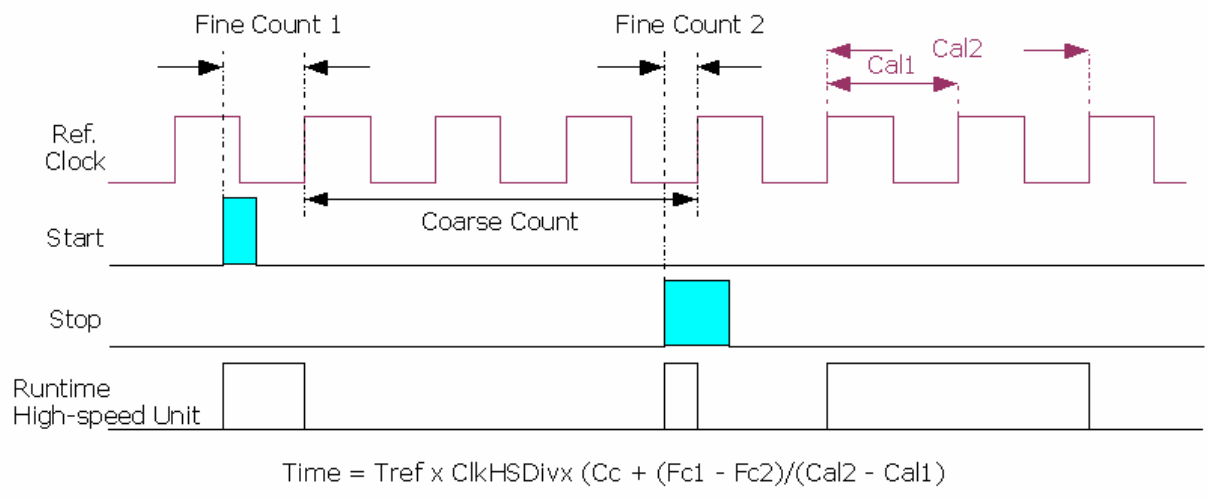
那么需要注意的是由单片机给的 start 信号与激光的 start 信号（也就是 stop1 信号）的时间要在 50ns 以上，这个时间是为了给 start 信号加噪声。在这个情况下的测量过程中需要将上面的寄存器 1 的配置稍作修改：

SPIwrite (0x81194900) ; ;//stop1 和 stop2 通道分别接受一个脉冲， 定义计算方法，用 stop2 的第一个脉冲减去 stop1 的第一个脉冲。
在应用我们的 gp2 评估测量系统测试情况下，测量 1us 时间间隔在平均 1000 次后，噪声曲线如下：

在平均 1000 次的情况下，输出的峰峰值噪声降低到 10ps 以内，相当于分辨 1mm 的距离。那么通过这种平均的方式提高测量精度，对于测量频率不高的激光测距应用是非常有帮助的。

测量范围 2：
2xTref-4ms

在测量范围 2 当中，测量是通过 TDC 测量单元和一个预计数器共同完成的。如图所示：



在测量时差相对较大的时候，tdc 的内部核心测量单元只测量如图所示从 start 信号开始到下一个参考时钟周期的上升沿，然后测量 stop 信号上升沿到下一个参考时钟周期的上升沿。那么中间的时间，则由数时钟周期 coarse count 来得出。因此在测量范围 2 结果的计算公式为上面图中的公式，其中 Tref 为时钟周期，clkhsdiv 为分频因数，Cc 为 coarse count 所数的周期个数，Fc1 为 start 信号开始到下一个参考时钟周期的上升沿时间，Fc2 为 stop 信号上升沿到下一个参考时钟周期的上升沿时间，cal1 和 cal2 为 tdc 核心测量参考时钟周期，做校准用。

在这个模式下 TDC-GP2 应用一个内部的计数器将测量范围扩展到了 4ms。那么选择这个测量范围后，测量只能够在 start 通道和 stop1 通道中进行，stop1 通道最多接受的脉冲数为 3 个。那么测量的时差范围从 2 倍的内部时钟周期最大到 4ms 的时差。 在这个精度下的典型精度保持不变，还是 65ps。

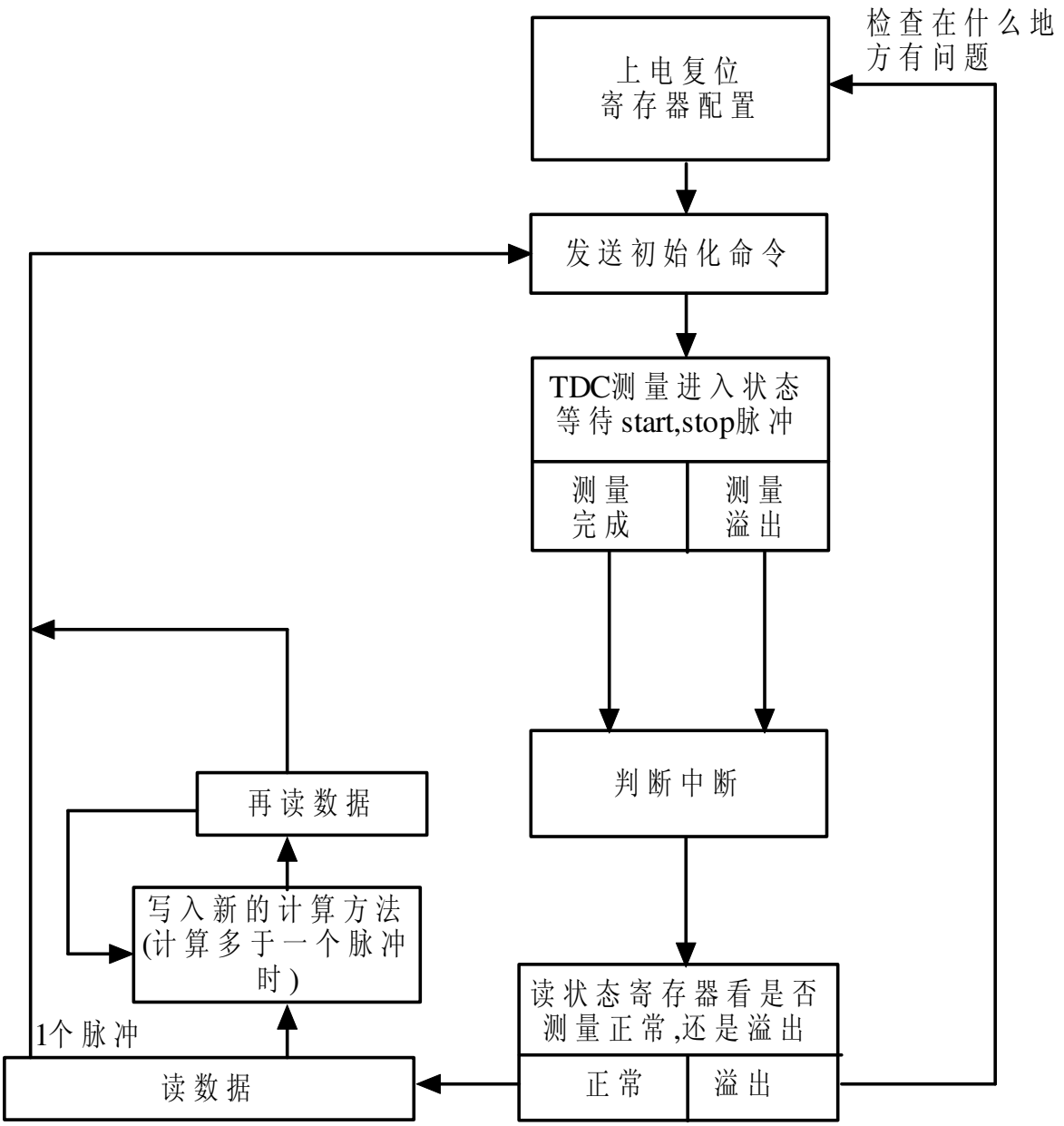
需要引起注意的问题：

- 对于 GP2 来讲触发它的脉冲宽度必须要大于 3.5ns。

- 在 start 脉冲触发之后 stop 脉冲和 start 脉冲间隔至少要大于 2 倍的内部参考时钟周期。
- 在没有应用精度可调节模式的情况下 GP2 的测量必须要进行校准。在低采样频率的情况下最简单的方法就是应用芯片自带的自动校准功能。或者也可以应用手动校准 TDC。

TDC-GP2 的测量范围 2 的优点：
- 整个的测量范围最大可以达到 4ms,精度还是 65ps,系统的精度将主要取决于光学系统中其他器件的精度。
- 对于长距离测量为了能够得到相当强度的返回信号高能的激光脉冲信号是必要的。
- 可以检测到最多 3 个脉冲反射，这个时候脉冲对的精度为 2 个时钟周期（至少 2 个时钟周期的时间间隔）。
- 在这个测量模式下可以通过平均的方式消除噪声。采样 100 次数据,那么平均之后的标准偏差将会被减少 √100=10 倍，，到 6.5ps ≡ 0.98mm。做为必要的条件 start 信号和 stop 信号必须与时钟信号完全孤立开，没有任何关联。否则采样的量化误差将会落在同一位置上。

在这个测量模式下测量流程以及典型的寄存器设置如下：




```
void gp2config()
{
SPIwrite8 (0x50); //上电复位

//配置寄存器：
SPIwrite32 (0x80008468) ; // 选择测量范围 2，自动校准，晶振上电后一直起振。
SPIwrite32 (0x81214200) ;// stop1 接受 1 个脉冲，定义计算方法，用 stop1 通道的第一个脉冲减去 start 脉冲
SPIwrite32 (0x82E00000) ; // 开启所有中断源
SPIwrite32 (0x83000000) ;
SPIwrite32 (0x84200000) ;
SPIwrite32 (0x85080000) ;
}

//测量循环：
void measurement()
```

```
{
SPIwrite8 (0x70) ;//初始化测量，通知 gp2 进入测量准备状态

Check INTN=0?
SPIwrite8 (0xb4) ; //发送命令读状态寄存器
SPIread8 (STAT) ;
STAT&0x0600>0？ //=>说明有测量溢出，有问题
SPIwrite8(0xb0) ; //发送命令读 reg0 结果
SPIread32(reg0) ;
}
```

在上面的 gp2 测量范围 2 寄存器设置中，需要注意的是在寄存器 1 中设置接受脉冲个数的时候，如果你预期接受 1 个脉冲，则需要设置为 2 个。因为在测量范围 2 中如原理图所示实际上是进行了两次测量，因此要多设置一个脉冲。
同样在测量范围 2 中，对于系统测量数据进行平均同样可以大大的提高测量精度，因此用户可以根据自己的实际情况和所需的测量频率来进行相应得平均测量。

例程代码：

下面是用 C 语言编写的 MSP430F135 系列单片机对于 gp2 控制的一些控制函数的例程代码+注释，其中描述了一些对 gp2 功能上的控制，通信为单片机用普通 io 口模拟 spi 通信操作，仅供参考：

```
//============端口设置程序================//
void port_init(void) //端口初始化
{
 P5DIR |=BIT1;//EN_STOP
 P5DIR |=BIT2;//START
 P5DIR |=BIT3;//ENSTART
 P5DIR &=~(BIT4);//INTN
 P6DIR |=BIT5;//RSTN
 P5OUT &=~(BIT1);
 P5OUT &=~(BIT2);
 P5OUT &=~(BIT3);
 P5OUT &=~(BIT0);//stop
 P6DIR |=BIT0;//SCK-输出方向
 P6DIR |=BIT1;//SSN-输出方向
 P6DIR &=~(BIT2);//SO-输入方向
 P6DIR |=BIT3;//SI-输出方向
 P6OUT &=~(BIT0);//SCK-空闲时低平，spi 时钟极性为 0。
 P6OUT |=BIT1;//SSN-片选在没有通信情况下置高
}

void spi_enable(void) //开启 spi 通信,将 ssn 置低
{
 P6OUT &=~(BIT1);//ssn 置低
 _NOP();
 return;
}
```

```
void spi_disable(void) //关闭 spi 通信,将 ssn 置高
{
 P6OUT |=BIT1;//ssn 置高
 _NOP();
 return;
}

void send_zero(void) //发送 0
{
 P6OUT |=BIT0;//SCK 高电平
 _NOP();
 P6OUT &=~(BIT3);//SI-输出一个低平
 _NOP();
 P6OUT &=~(BIT0);//SCK 低平
 _NOP();
 return;
}

void send_one(void) //发送 1
{
 P6OUT |=BIT0;//SCK 高电平
 _NOP();
 P6OUT |=BIT3;//SI-输出一个高平
 _NOP();
 P6OUT &=~(BIT0);//SCK 低平
 _NOP();
 return;
}
//=============================================//

//======================SPI 写数据====================//
void spi_write8(unsigned char wbuf8)  // spi 写入 8 位数据
{
 unsigned char cnt,tmp=0x80;
 spi_enable();
 for(cnt=8;cnt>0;cnt--)
    {
     if((wbuf8&tmp)>0)
       send_one();//发送 1
     else
       send_zero();//发送 0
     tmp /=2; //tmp 右移一位
    } //没有 spi 关闭命令，测试程序中代码关闭！
return;
}

void spi_write16(unsigned int wbuf16) // spi 写 16 位数据
{
```

```
 unsigned char cnt;
 unsigned int tmp=0x8000;
 spi_enable();
 for(cnt=16;cnt>0;cnt--)
    {
     if((wbuf16&tmp)>0)
       send_one();//发送 1
     else
       send_zero();//发送 0
     tmp /=2; //tmp 右移一位
}
_NOP();
 spi_disable();
 return;
}

void spi_write32(unsigned long wbuf32) // spi 写 32 位数据
{
 unsigned char cnt;
 unsigned long tmp=0x80000000;
 spi_enable();
 for(cnt=32;cnt>0;cnt--)
    {
     if((wbuf32&tmp)>0)
       send_one();发送 1
     else
       send_zero();//发送 0
     tmp /=2; //tmp 右移一位
}
_NOP();
 spi_disable();
 return;
}
//=============================================//
//====================SPI 读数据====================//
unsigned long spi_read32()
{
unsigned char cnt;
 unsigned long tmp=0x80000000;
 unsigned long int rbuf32=0x00000000;
 spi_enable();
 for(cnt=32;cnt>0;cnt--)
    {
     P6OUT |=BIT0;//SCK
     _NOP();
     if((P6IN &0X04)>0X00)//P6.2 SO
        rbuf32 |=tmp;
     tmp /=2;
     P6OUT &=~(BIT0);// SCK
     _NOP();
```

```
}
_NOP();
spi_disable();
 return(rbuf32);
}
//=============================================//

//===========GP2 上电复位程序=================//
void GP2_RESET(void)
{
 P6DIR |=BIT5; //设置 p6.5 输出方向
 P6OUT |=BIT5; //输出高平
 _NOP();
 P6OUT &=~(BIT5); //输出低平
 _NOP();
 P6OUT |=BIT5; //输出高平
 _NOP();
}//给 gp2RSTN 管脚一个 Reset 的方波
//=============================================//
//===========GP2 寄存器配置程序================//
void GP2_init(void)
{
 unsigned long REG0,REG1,REG2, REG3, REG4,REG5;
 unsigned char PU=0X50;
 unsigned char init=0x70;
 REG0=0X80008420;//校准陶瓷晶振时间为 8 个 32k 周期,244.14us.
          //设置高速晶振上电后一直起振.设置测量范围 1,自动校准,上升沿敏感
 REG1=0X81014100;//预期 stop1 脉冲数 1 个.计算第一个 stop1-start
 REG2=0X82E00000;//开启所有中断源
 REG3=0x83080000;//
 REG4=0x84200000;//
 REG5=0X85080000;//
 spi_write8(PU);//上电复位
_NOP();
spi_disable();

 spi_write32(REG0);
_NOP();
spi_write32(REG1);
_NOP();
spi_write32(REG2);
_NOP();
spi_write32(REG3);
_NOP();
spi_write32(REG4);
_NOP();
 spi_write32(REG5);
 _NOP();
return;
}
```

```
//=============================================//

//===========GP2 时钟校准程序==================//
void GP2_cal(void)
{
 unsigned char cal_start=0x03;
 unsigned char read_reg0=0xb0;
 unsigned long int CAL;
 float CAL_f;
P5OUT |=(BIT3);//EN_START
 spi_write8(cal_start);//启动校准
 _NOP();
spi_disable();

while((P5IN&0x10)==0x10)//判断中断置位否
     _NOP();
 spi_write8(read_reg0);  //读校准的时间数据
_NOP();
CAL=spi_read32();//通过计算校准系数为 244.146/(float(CAL)/65536*0.250))
_NOP();
return;
}
//=============================================//

//=========时间测量状态寄存器判断程序==========//
void GP2_TMSTAT()
{
 unsigned char stat=0xb4;
 unsigned long a;
 unsigned int  b;
 while((P5IN&0x10)==0x10)//判断中断置位否
     _NOP();
 spi_write8(stat);
 _NOP();
 a=spi_read32();
 b=(a>>16);
 if((b&0x0600)==0)
   BZ1 &=0XFE;
 else
   BZ1 |=0X01;//置溢出预计数器或溢出 TDC 标志
}
//=============================================//
//=============用单片机口产生 START 信号==========//
void GP2_START(void)
{
 P5OUT &=~(BIT2);//START
 _NOP();
 P5OUT |=(BIT2);
 _NOP();
 P5OUT &=~(BIT2);
```

```
 _NOP();
 return;
//=========================================//

//============通讯测试===============//
void testcomunication(void)
{
unsigned long int REG1;
unsigned char cnt;
 unsigned char tmp=0x80;
 unsigned char test_reg=0xb5; //读结果寄存器 5，反映寄存器 1 的高 8 位
 unsigned char test_reg0=0x00;
REG1=0x81112233; //写寄存器 1，随便输入，然后从结果寄存器 5 读高 8 位
spi_wriet32(REG1) ;
_NOP();
 spi_write8(test_reg);
 _NOP();
 for(cnt=8;cnt>0;cnt--)
    {
     P6OUT |=BIT0;//SCL
     _NOP();
     if((P6IN &0X04)>0X00)//P6.2 SO
        test_reg0 |=tmp;
     tmp /=2;
     P6OUT &=~(BIT0);//SCL
     _NOP();
    }
 spi_disable();
}
//=====================================//
//==========测试 fire 脉冲产生测试=============//
void fire(void)
{
unsigned char TDC_init=0x70;
unsigned char start_cycle=0x01;
P5OUT |=(BIT3);//EN_START 开启
spi_write8(TDC_init); //初始化 TDC
_NOP();
spi_disable();

spi_write8(start_cycle); //发送 fire 脉冲
_NOP();
spi_disable();

GP2_START(); //给 start 信号，否则 fire 信号只能发送一次，系统将会挂起

}
//=====================================//
```

```
//==========温度测量=============//
void TEMP(void)
{
unsigned long int A,B,C,D;
SPIwrite32 (0x800007e0) ; //选择 4 个温度测量口，Tcycle = 300us @ 4MHz ref. clock, 2
                          //个预热测量, SelClkT = 1,

 spi_write8(0x02); //启动温度测量
_NOP();
spi_disable();
while((P5IN&0x10)==0x10)//判断中断置位否
    _NOP();
GP2_TMSTAT(); //GP2 状态读取
 _NOP();
spi_write8(read_reg0);
_NOP();
A =spi_read32();//读结果寄存器 0 的温度测量时间结果
_NOP();
spi_write8(read_reg1);
_NOP();
B =spi_read32();//读结果寄存器 1 的温度测量时间结果
_NOP();
spi_write8(read_reg2);
_NOP();
C =spi_read32();//读结果寄存器 2 的温度测量时间结果
_NOP();
spi_write8(read_reg3);
_NOP();
D =spi_read32();//读结果寄存器 3 的温度测量时间结果
_NOP();

//后面可以对参考电阻和传感器电阻阻值比较，然后查表获得温度


}
//====================================//
//======时间测量测试=========//
void timemeasurement(void)
{
unsigned long int M;

 spi_write8(0x70); //初始化 TDC
_NOP();
spi_disable();

 P5OUT |=(BIT3);//EN_START
 P5OUT |=(BIT1);//EN_STOP
 GP2_START();//给 start 信号
 _NOP();
```

```
 _NOP();
 _NOP();
 _NOP();
 _NOP();
 _NOP();
 _NOP();
 _NOP();
 _NOP();
 _NOP();
 _NOP();
 _NOP();
 _NOP();//延迟
 P5OUT |=BIT0;//给 STOP 信号
 _NOP();
while((P5IN&0x10)==0x10)//判断中断置位否
    _NOP();
GP2_TMSTAT(); //GP2 状态读取
 _NOP();
spi_write8(read_reg0);  //读时间测量数据
 _NOP();
M=spi_read32();
_NOP();

}
```

在应用 tdc-gp2 时还需要注意的一些细节问题：

关于 32K 晶振和 4M 高速晶振需要注意的问题

这两个晶振都可以通过外部来给出。但是 4M 高速晶振我们建议用我们的推荐电路，因为这个晶振时钟是参与测量的，所以需要非常好的进行控制。关于晶振电路的电阻和电容请也同样按照我们手册推荐的电路给出，因为：
首先和晶体串联的那个电阻，据我们的测试没有也应该是可以的。但是推荐使用这个电阻。因为晶体输出的是方波，这将引起谐波干扰，尤其是阻抗严重不匹配的情况下，加上电阻后，该电阻将与输入电容构成 RC 积分平滑电路，将方波转换为近似正弦波，虽然信号的完整性受到一定影响，但由于该信号还要经过后级放大、整形后才作为时钟信号，因此性能并不受影响。还有一点就是减小回波干扰及导致的信号过冲。 另一个和晶体并联的电阻，是必须要有的，这个电阻是为了帮助起振用的。要想构成一个振荡器，要求放大电路有一个合适的增益，因此通常会加入反馈电阻降低电路的增益为一个合适的值，这就是加入电阻的作用。如果不加的话起振就会有问题。

在 gp2 上电之后高速晶振和 32k 晶振应该自动起振，如果没有，说明电路或者芯片由问题-这个是判断芯片损坏的一种方法。示波器的探头要调到 x10 档。

高速晶振我们推荐使用陶瓷晶振，因为起振时间快，而且 tdc-gp2 可以对陶瓷晶振进行校准。用石英晶振没有太大的作用，而且起振时间慢，对于功耗控制上比较难。

关于电源控制问题

为了防止灌电流对于芯片的损坏，tdc-gp2 的 Vio 和 Vcc 电源需要满足一下要求：

首先电源为 TTL 和 CMOS 电平兼容，需要满足公式 Vio>Vcc-0.5V，否则如果 Vcc 过大，将会形成灌电流，io 内部的保护 2 级管将会被击穿，可能使芯片过热甚至烧毁芯片，所以在控制电源上，一定要注意以上所提到的问题，避免不必要的损失。

还有就是电源的稳定性。如果在电源上有周期性的干扰的话，对于测量是非常不利的。首先时差测量的精度和电源稳定性有关，还有温度测量也同电源有关。请尽量避免由于电源线干扰，稳定电源值，从而获得高质量的测量结果。

将 gp2 Vio 电源的滤波电容加到 200uf，而且离 gp2 越近越好。

关于 spi 通信：

我们手册上所讲的 spi 通信需要将时钟相位置 1，时钟极性置 0。但是对于不同的单片机似乎有所不同。在 msp430 中的 spi 通信时钟相位和时钟极性的定义与摩托罗拉正好相反。所以在进行通信调试的时候，请改变一下时钟相位和时钟极性进行测试，看看到底适用哪种情况。

关于温度测量问题：

首先温度测量，要求电源电压要稳定，不要有周期性的噪声。在测量的时候注意选择合适的电容对传感器和参考电阻放电。传感器至少在 500 欧姆以上。如果放电时间过短，将会被视做短路。正常情况下 gp2 测量 4 个数值，那么状态寄存器将会指到第 4 个寄存器。

时钟标定：

首先在时钟标定前一定要选通 start 通道。因为实际上时钟的校准是内部的一个用 4M 周期对于 32k 晶振周期的测量。

关于通信测试:

在测量时间脉冲之前，首先我们建议先测试一下单片机和 tdc-gp2 的通信是否正常。因为在做测量之前，我们要保证首先单片机和 gp2 已经建立通信。那么如果后面出现问题，则可以不再考虑 spi 通信问题。具体测试可以首先写入寄存器 1，然后从结果寄存器 5（8 位寄存器）里面读取写寄存器 1 的高 8 位。如果 spi 通信正常，那么结果寄存器 5 应该可以时时反应结果寄存器 1 的高 8 位。

长时间测量范围的误差：

在测量范围 2 中，如果测量的时差很大，那么我们建议应用高温度稳定的高速晶振。因为测量范围 2 的测量原理实际是 tdc 测量和数高速时钟周期的融合。那么大时差的情况下，对于晶振的稳定性要求需要很高，因为数时钟周期数会很多。如果假设时钟周期稳定性为 10ppm，那么一个周期的话误差为 250ns 的百万分之 10，2.5ps,如果测量范围很大，比如说 100us，那么误差累加会有 1ns 之多。因此极力建议在测量大时间间隔时使用高稳定性的晶振。因为频率误差会以频率的个数累加起来。

关于电源稳定对于测量及测量范围影响问题：

因为 TDC-GP2 的高速测量单元的测量偏差 LSB 是与 Vcc 的电源电压有密切关系的，电源电压范围从 1.8-3.6V,那么在这个范围内 LSB 的变化是很大的，35-111ps,那么电压越高，LSB 越好，那么在低压情况下，测量范围 1 的测量范围也会随之增加。 而且电源对于测量出了会影响 LSB 的值之外，还会引入一个漂移。在电压不同情况下引入的漂移量是不同的，因此需要保证在测量过程中电源电压保持一定。

关于寄存器配置在程序中的位置：

如果你每一次测量只计算一个时差的话，那么你可以在开始的时候配置一次寄存器即可，在后面的测量循环中不需要再进行寄存器配置。如果你在测量的时候改变了 ALU 的计算方法，比如说你测量 3 个脉冲，在测量结束后要读 3 个数据，因此改变了计算符，那么需要在读完结果后，再次配置寄存器 1，将第一次的计算符写入寄存器 1。否则，下一次测量将会默认这次测量最后一次计算符为下一次计算的公式。