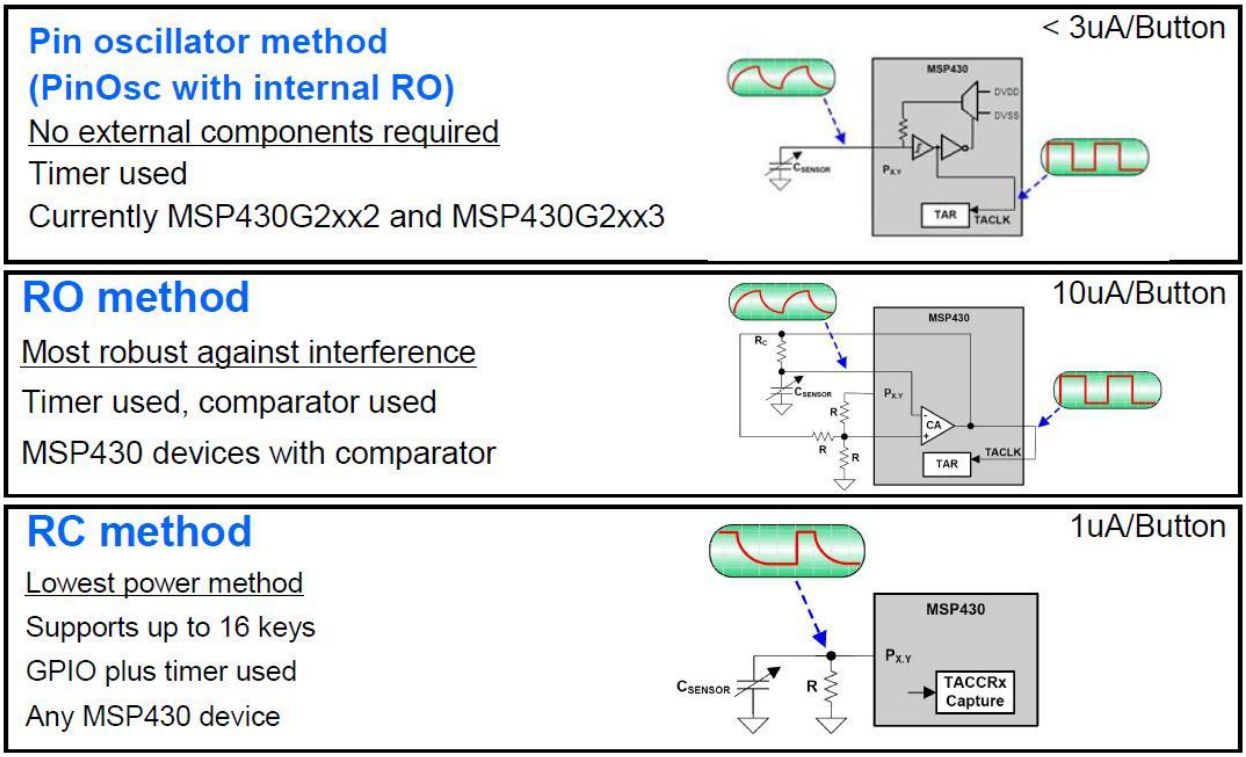
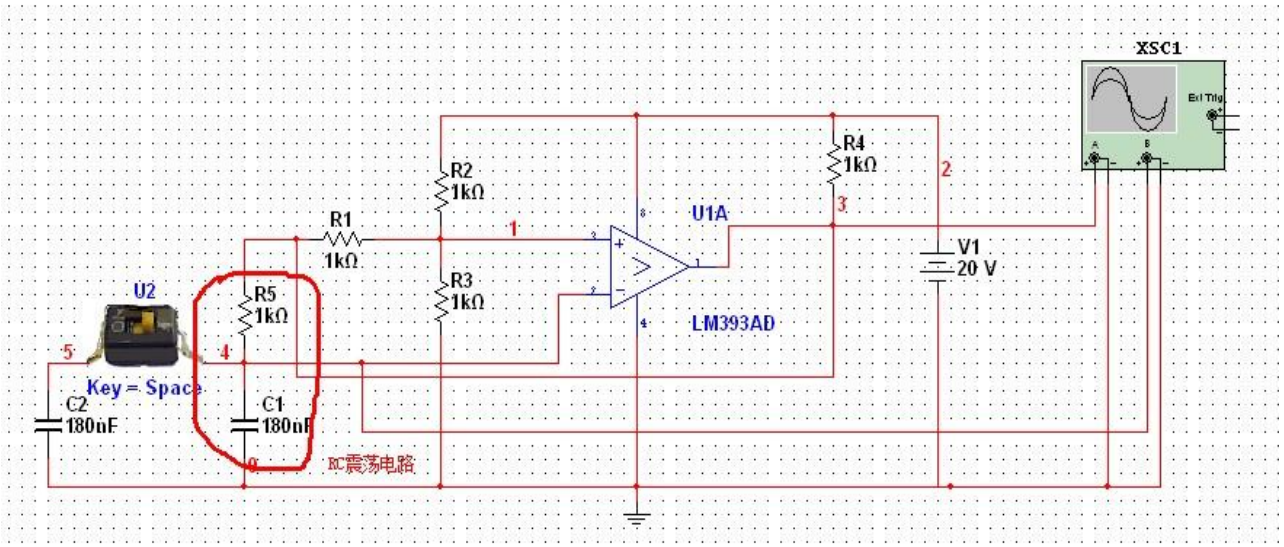
Capitiactive Touch Pad

硬件检测电路原理：

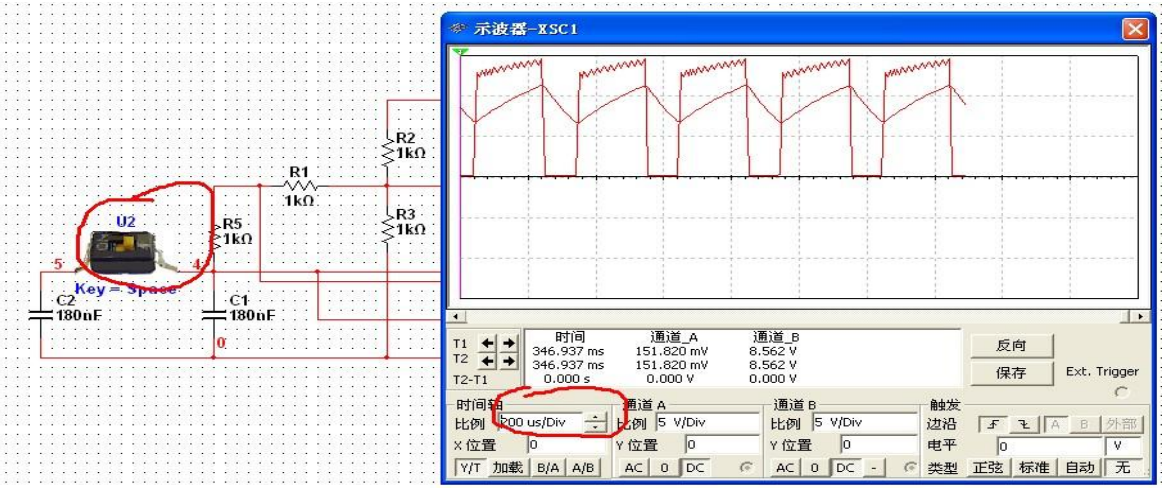
3 种常见电容检测电路介绍：1.RC 检测，基本趋于淘汰，灵敏度低，2.RO 外部震荡，外围需要一些电阻电容，加大 PCB 体积，3.PinOsc with internal RO, 外部只需挂一个电容。

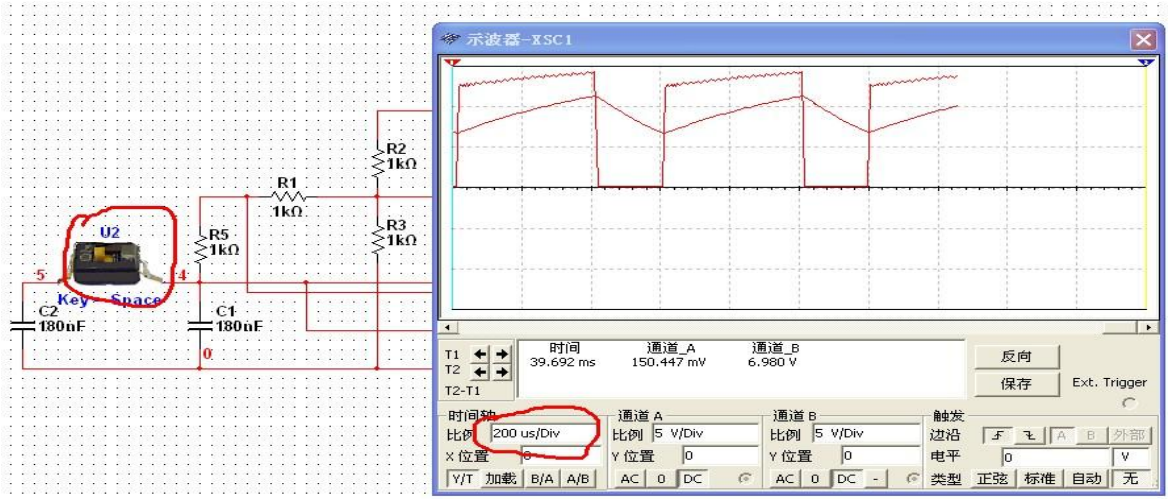


RO 电容检测：



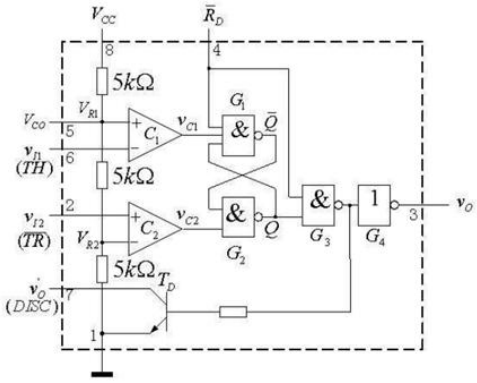
其原理就是测震荡电路频率，关键参数在电容----C1 的充放电，R5 和 C1 构成一阶 RC 震荡电路。比较器的输入电压时随着输出电压变化而变化的，而比较器负输入端电压是由 C1 充放电决定。通过计算可以发现，电容电压在 1/3VCC-2/3VCC 之间反复变化。其震荡频率和比较器输出频率保持一致，这样，通过检查比较器输出波形的频率就可以知道我们输入电容的容值。当输入电容发生变化的同时,输出频率也会改变：

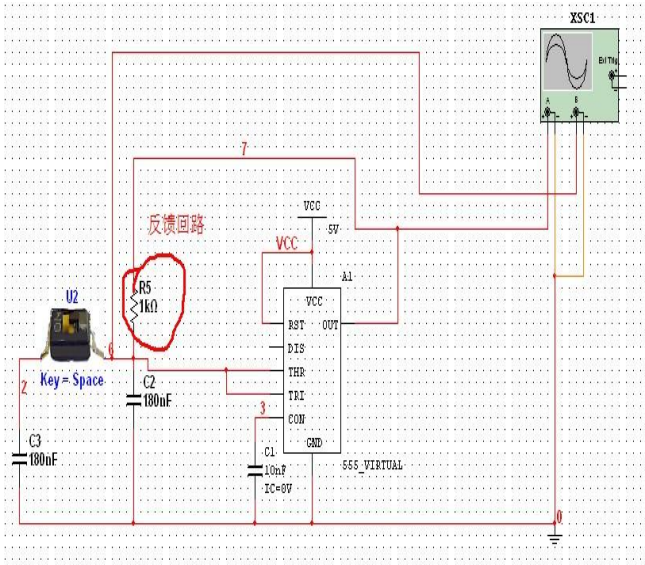




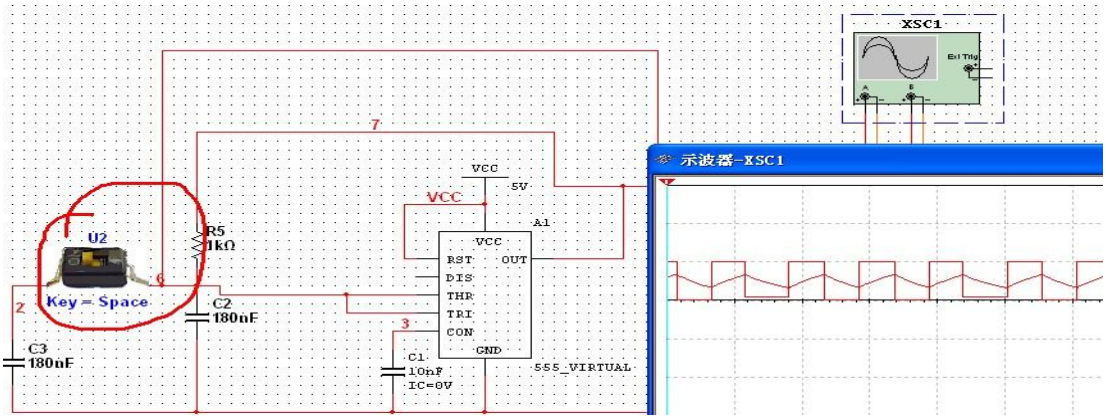
利用该原理可以在 MSP430 外搭若干 RC 电路和内部比较器可以完成比较器方波频率的检测。

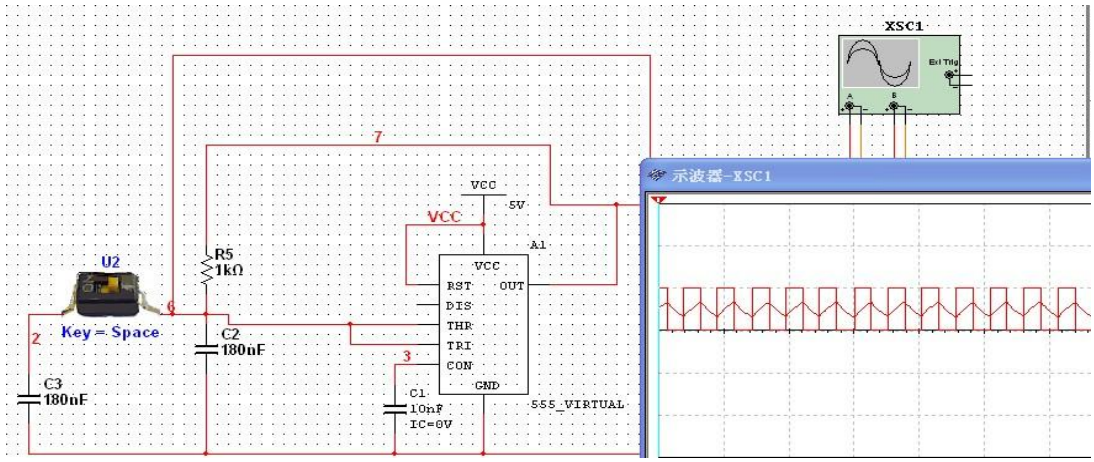
或者利用 PinOsc with internal RO 方式，只需在 MSP430 IO PIN 外挂一个可变电容来检测方波频率，其原理利用了一个施密特触发器和一个反相器构成一个震荡电路，同样是检测电容充放电时间！电容不一样，后级输出频率不一样！





NE555 内部正好由一个施密特触发器和反相器构成的一个集成芯片，短接 P2 和 P6 外挂一电容，并且把 555 输出反馈回输入管脚，构成一震荡电路，C2 上的波形如下所示：





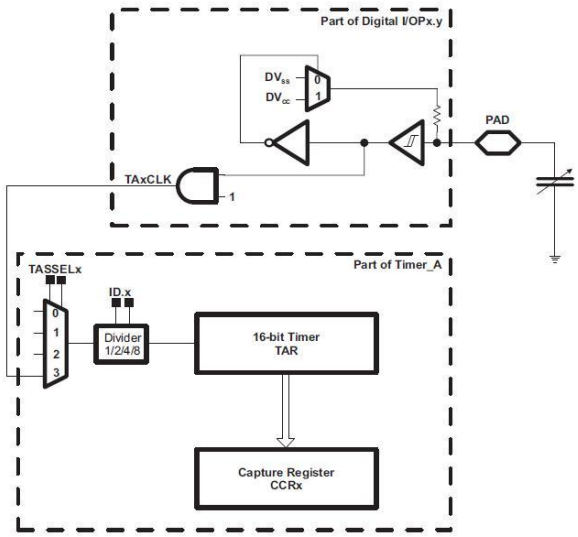
从硬件仿真上可看出，随着外部电容的变化，内部方波的频率也在变化。

MSP430 Pin Oscillator 简介：

以下是在 MSP430G2542 User Guide 截取的部分说明：

Some MSP430 devices have a pin oscillator function built-in to some pins. The pin oscillator function may be used in capacitive touch sensing applications to eliminate external passive components. Additionally,the pin oscillator may be used in sensor applications.No external components to create the oscillation Capacitive sensors can be connected directly to MSP430 pin Robust, typical built-in hysteresis of ~0.7 V. When the pin oscillator function is enabled, other pin configurations are overwritten. The output driver is turned off while the weak pullup/pulldown is enabled and controlled by the voltage level on the pin itself.The voltage on the I/O is fed into the Schmitt trigger of the pin and then routed to a timer. The connection to the

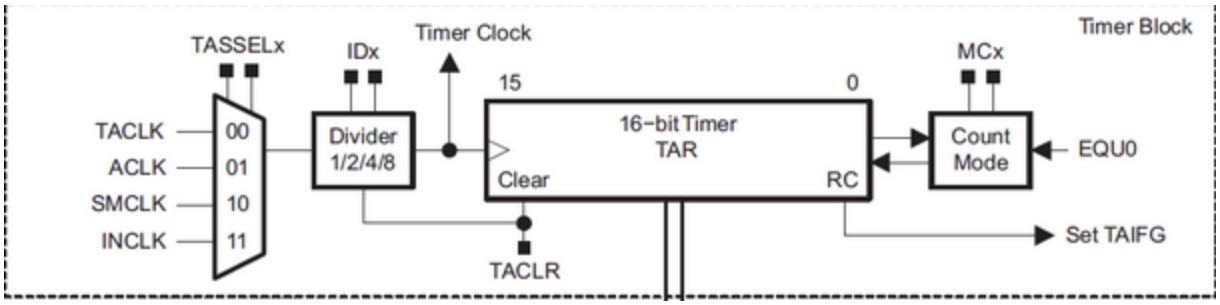
timer is device specific and, thus, defined in the device-specific data sheet. The Schmitt-trigger output is inverted and then decides if the pullup or the pulldown is enabled. Due to the inversion, the pin starts to oscillate as soon as the pin oscillator pin configuration is selected. Some of the pin-oscillator outputs are combined by a logical OR before routing to a timer clock input or timer capture channel. Therefore, only one pin oscillator should be enabled at a time. The oscillation frequency of each pin is defined by the load on the pin and by the I/O type. I/Os with analog functions typically show a lower oscillation frequency than pure digital I/Os. See the device-specific data sheet for details. Pins without external load show typical oscillation frequencies of 1 MHz to 3 MHz。



这是从 datasheet 上 GPIO 部分截取描述具有电容检测功能 IO 引脚的介绍，通过配置 Px.xSEL 管脚（es：把 P1.6 设置为电容检测引脚 P1SEL &= ~BIT6; P1SEL2 |= BIT6; 相应引脚的第二功能设置见 datasheet slas722e P38），使能电容检测硬件，到此，触摸屏的硬件设计工作已经全部完成。

软件设计：

MSP430 软件的目的性很明确，就是测量方波的主频，怎么检测？继续看上图，假设我们硬件已经设计完成，现在我们也配置好了 MSP430 Pin Oscillator 管脚，方波的输出在哪里？下面这张图出自 User guide Timer A 原理说明。

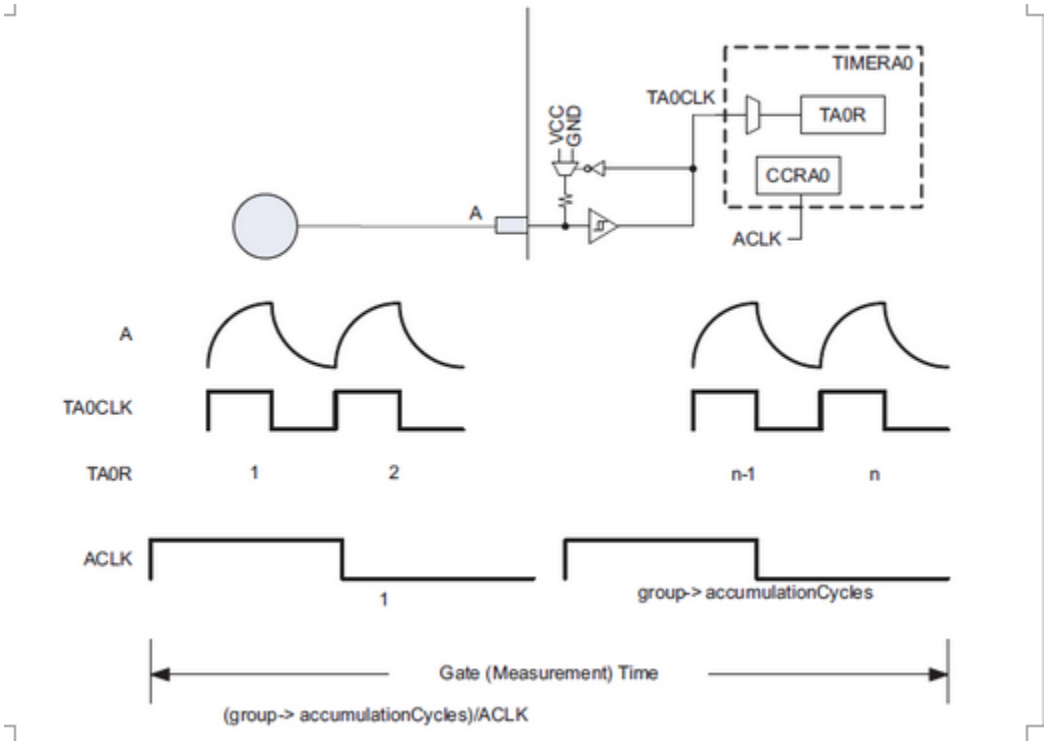


首先来下 TIMERA 的时钟源：TAxCLK 是定时器时钟，定时器时钟的选择我们可以通过配置寄存器 TASSELx，一般我们不会用到 INCLK 这路时钟，而在电容检测上，使能内部 OSC 输入，就会在 MSP430 内部史密斯触发器后端产生 INLCK 信号，其频率即为外部电容和内部上拉电阻构成的 RC 震荡电路的频率。

现在我们需要检测这个时钟的频率，如何做到？

Timer A 我们作为一个捕获模块使用，TAR 作为脉冲计数器，INCLK 作为 Timer A 时钟，CCR1 记录电平跳变时 TAR 的值。我们利用另外一个定时器（一般使用看门狗），定一段时间，定时到后，人为制造一个由低到高的电平跳变让,CCR1 来记录 TAR 跑了多少，TAR 数字就是这段时间内 INCLK 脉冲的个数，利用这个办法可以变相记录频率问题！！！！

下面以一个图总结下记录电容频率的过程：



Caution：

在理解上面的基础之上，有些东西需要我们思考：

1.看门狗定时多长合适，是否存在 TAR 在从 0-65536 计满数之后又从 0 开始计数才触发捕获。

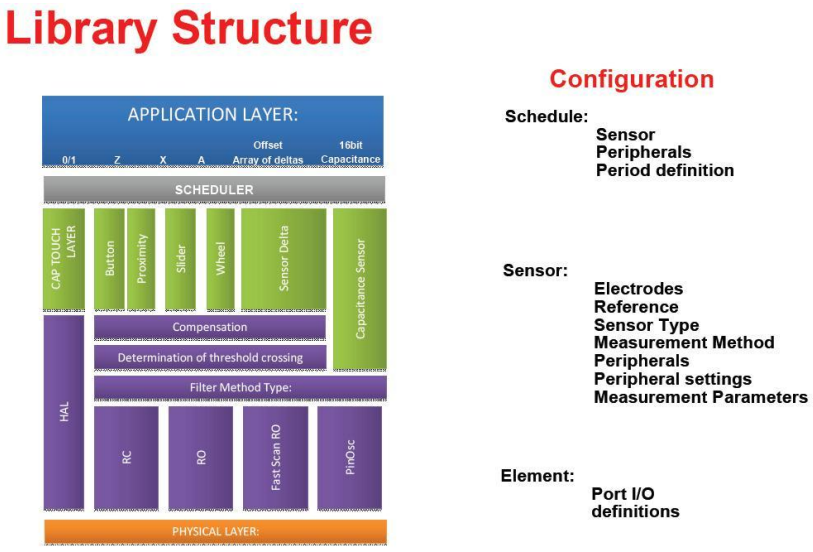
2.没有触摸时，CCR1 捕获的值是否稳定，触摸发生时，CCR1 会处于什么数值范围。

3.合适的阈值

4.按键按下时 CAP 是频率增加还是减少？

相信大家根据上面的东西能设计出自己的触摸按键是没有问题了，但是如何设计一个稳定，在各种复杂环境下能使用的的触摸按键，或者滑轮按键，TI 提供一套完整的解决方案，TI 提供一套触摸按键 lib，里面的软件完全开源，而且已经封装成 API 函数，我们只需调用相应函数即可实现单按键，多按键，滑调按键识别过程。

Lib 的使用说明见 SLAA490–April 2011，大家可在 TI 官网上找到（http://focus.ti.com/docs/toolsw/folders/print/capsenselibrary.html.）



这是 Lib 的层次介绍图，详细可结合 TI Launchpad touch sense 实验！

下面附上自己编写的触摸按键代码：（在 Lauchpad 上需要稍作修改）

```
#include  "msp430g2102.h"

unsigned int counts[2];

unsigned int Count_Baseline[2];
unsigned int threshold[2] = {1000,300};
unsigned int TouchFlag[2];

void LED_Test(void)
{
  if(TouchFlag[0]==1)
    P1OUT |= BIT4;
  else
    P1OUT &= ~BIT4;
  if(TouchFlag[1]==1)
    P1OUT |= BIT5;
  else
    P1OUT &= ~BIT5;
}

void Osc_pin(void)
{
  P1OUT = 0x00;                         // Clear Port 1 bits
  P1DIR |= (BIT0 +BIT4 +BIT5);                      // Set P1.0 as output pin

  P2SEL &= ~(BIT6 + BIT7);              // Configure XIN (P2.6) and XOUT (P2.7) to
GPIO
  P2OUT = 0x00;                         // Drive all Port 2 pins low
  P2DIR = 0xFF;                         // Configure all Port 2 pins outputs

//  P1SEL &= ~BIT6;                   // P1.2，P1.6 配置为电容检查引脚
//  P1SEL2 |= BIT6;

}

void Touch_Init(void)
{
```

```
    unsigned int i;
    for(i=0;i<2;i++)
    {
     if(i==0)                          //触摸管脚选择
     {
      P1SEL &= ~BIT6;                  //P1.6 配置为电容检查引脚
      P1SEL2 |= BIT6;

      P1SEL &= ~BIT2;
      P1SEL2 &= ~BIT2;
     }

     else if(i==1)                     //P1.2 配置为电容检查引脚
     {
      P1SEL &= ~BIT2;
      P1SEL2 |= BIT2;

      P1SEL &=  ~BIT6;
      P1SEL2 &= ~BIT6;
     }

     TA0CTL = TASSEL_3+MC_2;           // TACLK, cont mode
     TA0CCTL1 = CM_3+CCIS_2+CAP;       // Pos&Neg,GND,Cap
     IE1 |= WDTIE;                     // enable WDT interrupt
     WDTCTL = WDTPW+WDTTMSEL+WDTSSEL+WDTIS1+WDTIS0;
     //看门狗作为定时器使用,时钟设为 ACLK，64 分频  （门限）
     TA0CTL |= TACLR;                  // Clear Timer_A TAR
     __bis_SR_register(LPM3_bits+GIE);    // Wait for WDT interrupt
     TA0CCTL1 ^= CCIS0;                // 人为创造一个由低到高的电平跳变
     counts[i] = TA0CCR1;              // 捕获计数脉冲数即 RCO 电容振荡次数
     WDTCTL = WDTPW + WDTHOLD;         // Stop watchdog timer
    }
}

void Touch_Update(unsigned int average)    //基值更新
{
  unsigned int temp,i;
  unsigned int Count_Baseline_temp[2];
  for(i=0;i<2;i++)
  {
```

```
    for(temp=0;temp<average;temp++)
    {
      Touch_Init();
      Count_Baseline_temp[i] = counts[i];
      Count_Baseline[i] = Count_Baseline[i]/2 + Count_Baseline_temp[i]/2;
    }
  }
}

void Touch_Anys(void)               //有手触摸，计数器值减小
{
  int tempCnt[2],detal,temp,i;
  for(i=0;i<2;i++)
  {
    Touch_Init();
    tempCnt[i] = counts[i];
    if(tempCnt[i])                  //电容是否起振
    {
      detal = tempCnt[i] - Count_Baseline[i];  //detal 记录该次和闸值的差
      if( detal >= 0 )                      //detal 大于 0
      {
        if(detal <= threshold[i])               //detal 小于门限 更新闸值基准
        {
          Count_Baseline[i] = Count_Baseline[i] + detal/2;
          TouchFlag[i] = 0;
        }
        else if (detal > threshold[i])          //detal 大于门限  置位标识符
          TouchFlag[i] = 1;
      }
      else if( detal < 0 )                   //detal 小于 0
      {
        temp = -detal;
        detal = temp;
        if(detal <= threshold[i])
        {
          Count_Baseline[i] = Count_Baseline[i] - detal/2;
          TouchFlag[i] = 0;
        }
        else if (detal > threshold[i])
          TouchFlag[i] = 1;
```

```
    }
   }
  }
}

void main(void)
{
 // Initialize System Clocks
 WDTCTL = WDTPW + WDTHOLD;           // Stop watchdog timer
 BCSCTL1 = CALBC1_1MHZ;              // Set DCO to 1, 8, 12 or 16MHz
 DCOCTL = CALDCO_1MHZ;

 BCSCTL2 |= DIVS_2;                  // divide SMCLK by 4 for 250khz   UCLK =
250K
 BCSCTL3 |= LFXT1S_2;                // LFXT1 = VLO      ACLK = LFO = 12K

 Osc_pin();
 Touch_Init();
 Count_Baseline[0] = counts[0];
 Count_Baseline[1] = counts[1];
 Touch_Update(5);
 while(1)
 {
  Touch_Anys();
  LED_Test();
 }
}

#pragma vector=WDT_VECTOR
__interrupt void watchdog_timer(void)
{
  __bic_SR_register_on_exit(LPM3_bits);        // 看门狗定时器长度即为窗口长
度
}
```