比较器之触摸按键实验

3.1 实验介绍

本实验使用 MSP430F5529 触摸按键模块、LED 模块。利用 MSP430F5529 中的比较器 B 来判断触摸按键的充放电，通过 CPU 时钟算出电容的振荡频率，再判断是否有按键被触摸，如果有按键触摸则通过相应的 LED 显示当前的状态。

3.2 实验目的

（1）学习电容触摸按键硬件电路原理；
（2）学习触摸按键应用实验操作及编程思想；
（3）学习触摸电容程序资源。

3.3 实验原理

人体是具有一定电容的。当我们把 PCB 板上的覆铜画成如下形式的时候，就完成了一个最基本的触摸感应按键。

近年来，电容触摸这个词不时会在我们耳边回荡。比起机械按键，电容触摸按键的优点很多：成本低、外观花哨、寿命超长、容易实现防水防尘。

电容触摸的技术听起来很高端，但是原理却不复杂。 如图 3.4.6 所示，人手指接近金属会使金属对地的电容值增大，想办法测出电容值发生的变化，就能判断按键被按下。电容发生变化的原因是，电路板的材质和结构决定了板子上有寄生电容，当人体接触 PAD 时，人体就会和 PAD 产生寄生电容，这时电路中寄生电容值就发生了变化。

MSP430F5529 的比较器可以为比较结果 0 和 1 时设定两个阈值，这样便构成一个 RC 振荡器(电容电压低于较低阈值时充电、高于较高阈值时放电)，振荡频率与电容的容值有关。当手指触摸到电容触摸按键以后，电容会由 C1 变化至 C2，振荡器的输出频率会发生变化，因此只需在固定时间内，计算出振荡器的输出频率。如图 3.1 所示由于振荡频率较高(未触摸时在几百 KHz 量级)，

可以直接用 CPU 时钟进行测量(即用变量自增来测量若干次振荡的耗时)。但为了防止测量中途出现中断影响测量，测量过程中应当关闭中断。如果在某一时刻输出频率有较大的变化的话，那就说明电容值已经被改变，即对应按键被按下了。

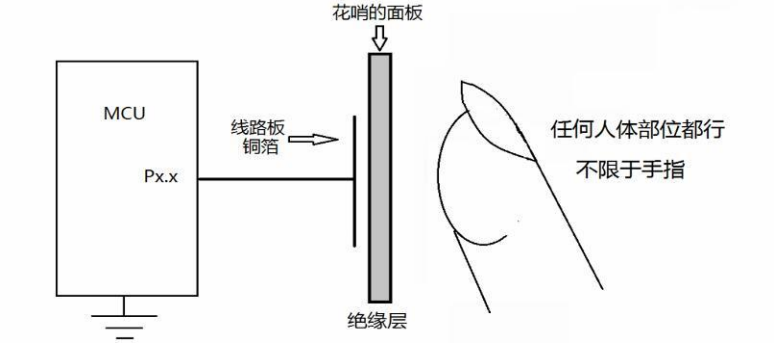

图 3.1 电容触摸原理

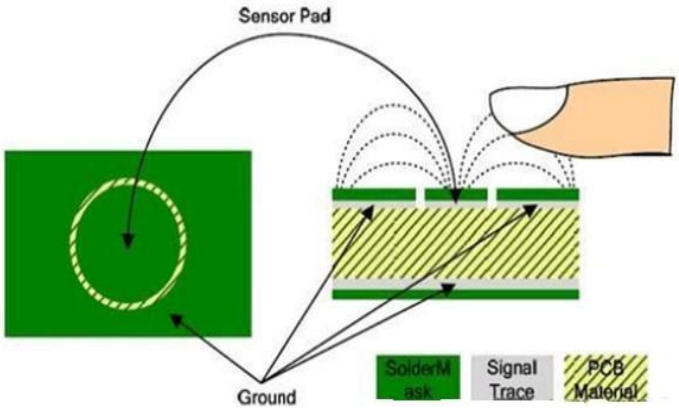

图 3.2 触摸按键工作原理图

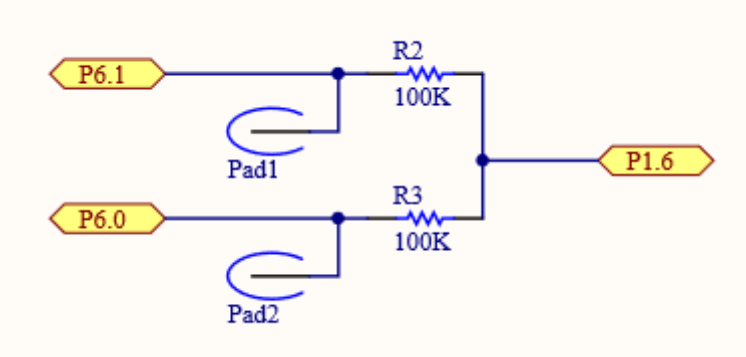

图 3.3 触摸按键模块电路原理图

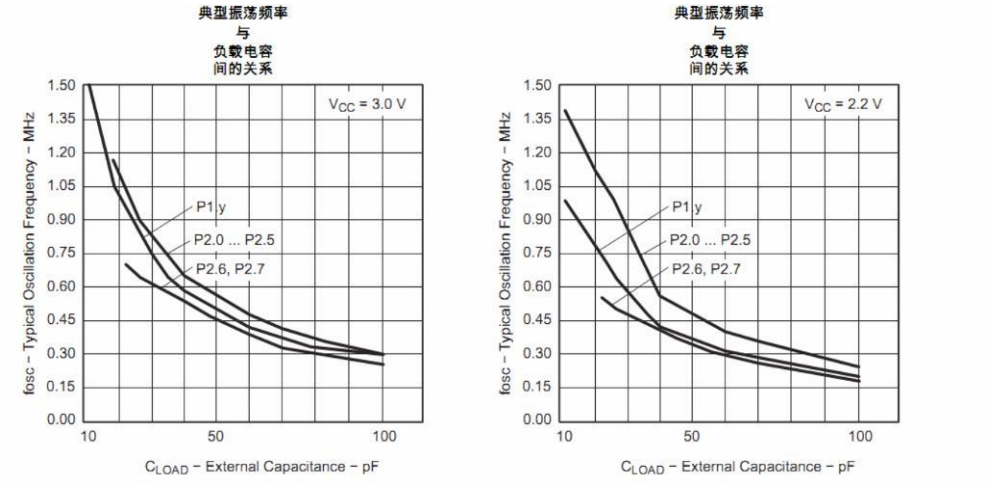

图 3.4 电容震荡特性

3.4 程序分析

3.4.1 编程思路

先初始化触摸按键和控制 LED 灯的 IO 口，然后反复的测量各个触摸按键的频率值，当它们的频率的值大于某个门限值时，操作对应的 LED 灯。在每次测量中要加入适当的延时，防止反复测量导致 LED 误操作。

3.4.2 程序流程图

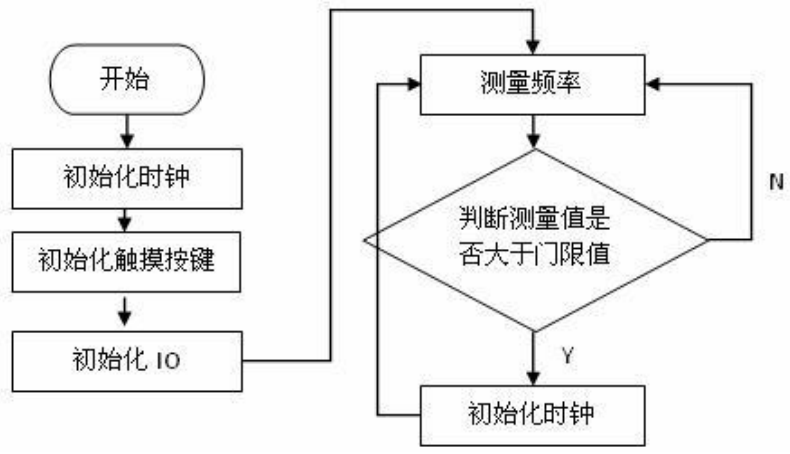

图 3.5 程序流程图

3.4.3 关键代码(伪代码)分析

主函数

```
int main( void )
{
    initClock();                                        //初始化始终
    initCapTouch();                                     //初始化触摸按键
    for(i=0;i<2;++i)
    *LED_GPIO[i]->PxDIR |= LED_PORT[i];//设置各 LED 灯所在端口为输出方向
    _EINT();
    while(1)
    {
        uint32_t temp[2] ={0,0};
        for(i=1;i>=0;i--)
        {
          temp[i]= CapTouch_ReadChannel(i);          //测量触摸按键的频率值
          if((temp[i]>captouch_max)&&(lastflag[i]==0))
          {//当测量值大于门 限值时翻转
              *LED_GPIO[i]->PxOUT ^= LED_PORT[i];   //对应的 LED 指示灯
              lastflag[i]=1;
          }
          else
              if(!(temp[i]>captouch_max))
              {
                  lastflag[i]=0;
              }
        }
        __delay_cycles(8000000);

    }


    return 0;
}
```

比触摸按键测量函数

```
uint16_t CapTouch_ReadChannel(int i)
{
  uint16_t cpu_cnt = 0;
  uint16_t osc_cnt = 0;
  CBCTL0 = CBIMEN + (i << 8);                       //外部信号加到负极
  P6OUT &= ~ALL_PORT;
  P6DIR |= ALL_PORT & ~(1 << i);
  CBCTL3 = 1 << i;
  uint16_t gie;
  _ECRIT(gie);
  while(1)
  {
    if(CBCTL1 & CBOUT)                              //控制电容的充放电
      P6OUT |= ALL_PORT;
    else
      P6OUT &= ~ALL_PORT;
```

```
        if(CBINT & CBIFG)
        {
          CBINT &= ~CBIFG;
          osc_cnt++;
        }
        if(osc_cnt >= OSC_CYCLES)
          break;
        cpu_cnt++;                                          //计算频率
      }
      _LCRIT(GIE);
      CBCTL3 = ALL_PORT;
      P6DIR &= ~ALL_PORT;
      return cpu_cnt;
    }
```

3.4.5 实验现象

触摸相应的按键 Pad1、Pad2，可以观察到对应的 LED 指示灯 L3、L4 的亮灭。