第 2-12 讲：NE555 频率测量

1. 学习目的

1. 了解 555 定时器原理及使用注意事项。
2. 掌握 IAP15F2K61S2/IAP15W4K61S4 读取 555 频率的程序设计。

2. 硬件电路设计

2.1. 555 定时器简介

555 定时器是一种集成电路芯片，常被用于定时器、脉冲产生器和振荡电路。555 可作为电路中的延时器件、触发器或起振元件使用。

555 定时器于 1971 年由西格尼蒂克公司（后被飞利浦公司并购）推出，由于其易用性、低价格和良好的可靠性，被广泛应用于电子电路的设计中。许多厂家都生产 555 芯片，包括采用双极型晶体管的传统型号和采用 CMOS 设计的版本。

2.1.1. 芯片 555 内部结构

555 芯片有 8 个引脚，引脚定义如表 1。芯片内部结构示意图如下，由此可看到内部结构图中有 3 个 5KΩ的电阻，这也是 555 名称的由来。这 3 个电阻分压 VCC，易推算出 VR1 处电压为 2/3VCC，VR2 处电压为 1/3VCC。这 2 个电压值对后面理解 555 芯片的典型应用电路很关键。

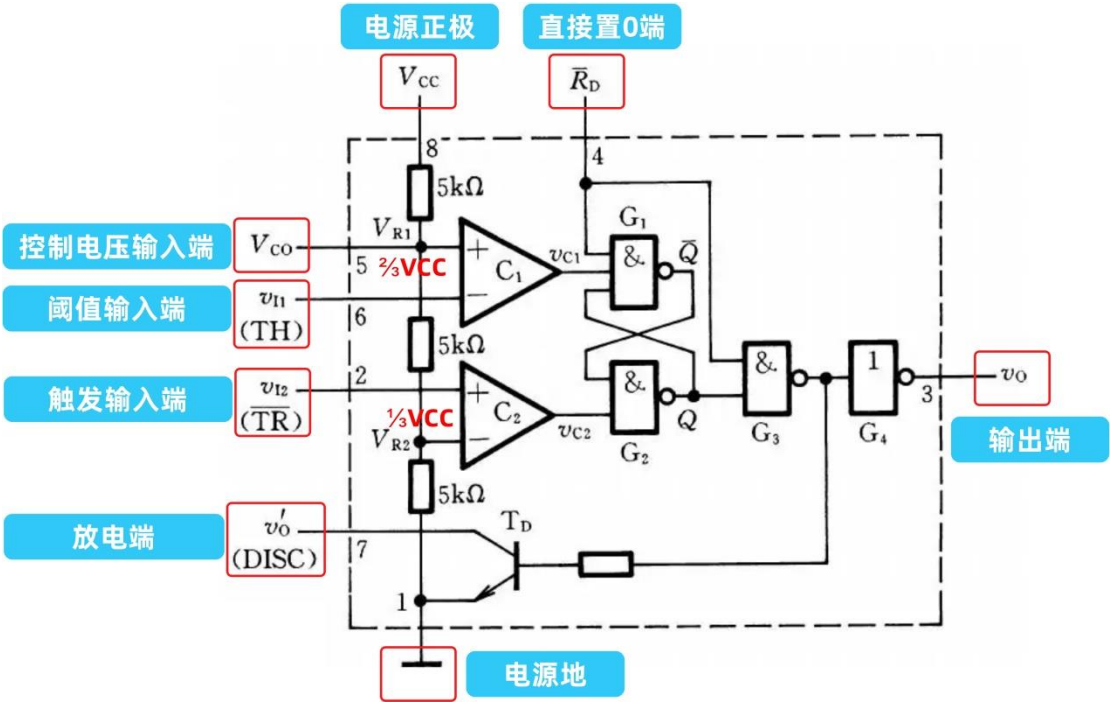

图 1：555 芯片内部结构示意图

表 1：555 芯片引脚描述

序号	针脚名称	功能描述
1	GND	芯片供电地。
2	TRIG	当此引脚电压降至 1/3VCC（或由控制端决定的阈值电压）时输出端给出高电平。
3	OUT	接收端，输出高电平或低电平。
4	RST	当此引脚接高电平时定时器工作，当此引脚接地时芯片复位，输出低电平。
5	CVOLT	控制芯片的阈值电压。（当此管脚接空时默认两阈值电压为 1/3VCC 与 2/3VCC）
6	THR	当此引脚电压升至 2/3VCC（或由控制端决定的阈值电压）时输出端给出低电平。
7	DISC	内接 OC 门，用于给电容放电。
8	VCC	芯片供电正。

2.1.2. 555 定时器典型应用

下图给出最简单的 555 定时器振荡电路，芯片第 3 引脚通过电阻 R1 可为电容 C1 充电。当电路上电时，电容 C1 里面没有电量，输出引脚 3 为高电平，那么流过电阻 R1 的电流可为电容 C1 充电。当芯片第 6 引脚检测到 2/3 的 VCC 时，控制输出引脚 3 变为低电平，电容 C1 停止充电。当芯片第 2 引脚检测到 1/3 的 VCC 时，控制输出引脚 3 变为高电平并重新通过电阻 R1 为电容 C1 充电。如此不断重复这一过程，则会在输出端输出脉冲信号。

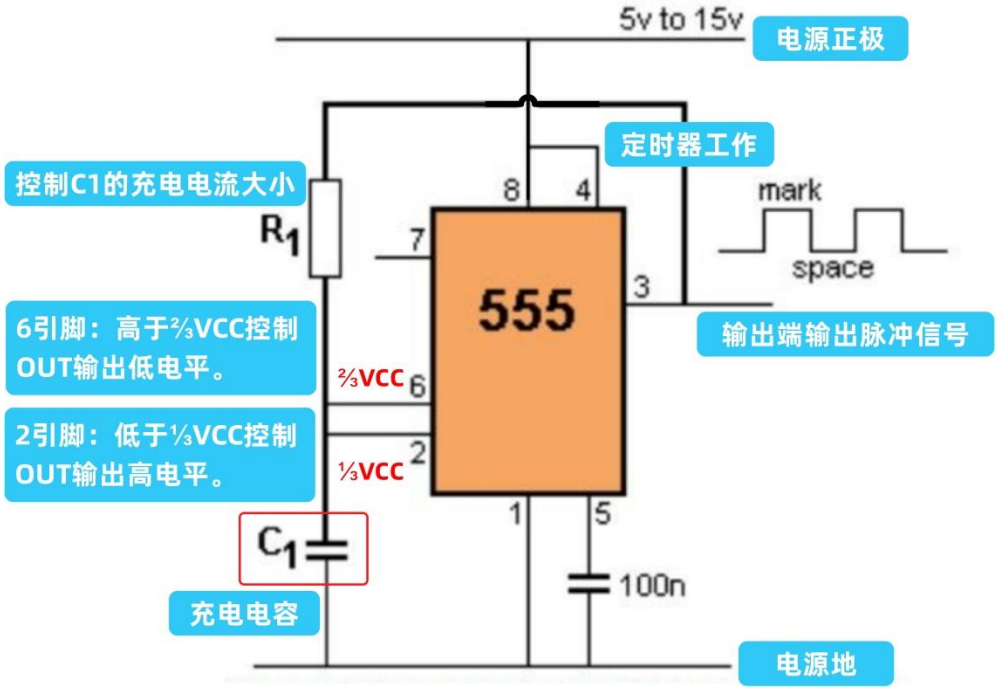

图 2： 555 定时器振荡电路

2.2. 555 定时器电路

PK107D 开发板上的 555 定时器硬件电路如下图所示。此电路就是上文介绍的 555 典型的应用电路，555 定时器振荡电路，此时的充电电容是 C45，而用于给充电电容充电的电阻由 R47 和可调电阻 RV5 串联而成，故通过调节可调电阻 RV5 可改变给充电电容 C45 充电电流的大小，这样就改变了芯片第 6 引脚充电到 2/3VCC 电压的时间，从而可以改变输出引脚产生脉冲的宽度（即改变了输出脉冲信号的频率）。

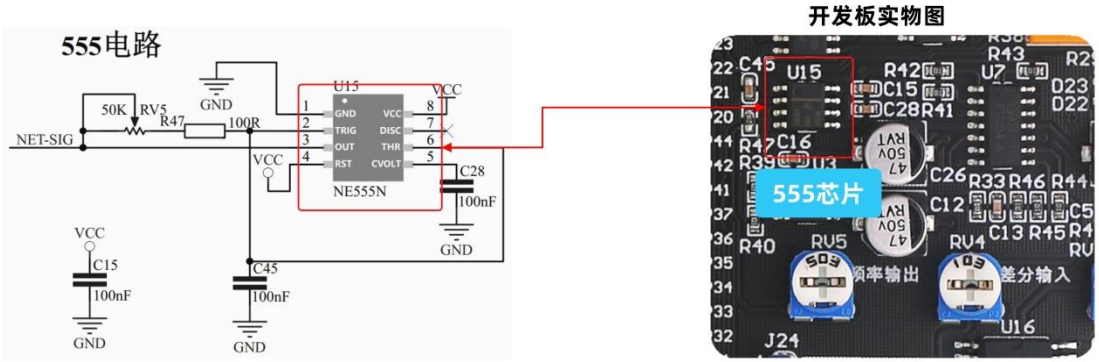

图 3：555 定时器振荡电路

✧ 注：555 定时器振荡电路的输出端引至开发板 J25 端子 SIGNAL，单片机定时器/计数器 0 作为计数器使用的外部引脚为 P3.4，所以可将 J25 端子的 SIGNAL 与 P34 短接。

2.3. 计数器应用流程

Timer 工作于计数器时的应用流程如下图所示。Timer 配置为计数器之后，设置计数寄存器的初值并根据需要开启中断，配置完成后，启动计数器即可。

这里要注意一下，计数器是对外部引脚的脉冲计数，他不会对外部脉冲分频的，所以不需要设置定时器的速度的（1T/12T）。

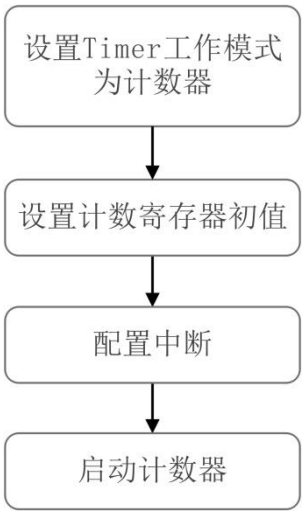

图 4：计数器应用流程

2.3.1. 设置 Timer 为计数器

Timer 设置为计数器和设置为定时器所用到的寄存器完全一样，区别是设置为定时器是将对应寄存器的位清零（=0），设置为计数器是将对应寄存器的位置位（=1）。

Timer 设置为计数器后，不再对系统时钟进行计数，而是对特定的引脚的输入脉冲进行计数。Timer 设置为计数器后，对应的外部输入引脚如下表所示，这些引脚都是固定的，无需软件配置的。如 Timer0 配置为计数器后，P3.4 即为其外部输入引脚，而且也只有 P3.4 能作为他的外部输入引脚。

表 2： Timer 的外部输入引脚

Timer	外部输入引脚
Timer0	P3.4
Timer1	P3.5
Timer2	P3.1
Timer3	P0.5
Timer4	P0.7

✧ **说明：**Timer 3 和 Timer4 是 IAP15W4K61S4 才有的外设，学习 IAP15F2K61S2 可忽略。

2.3.2. 设置计数器初值和配置中断

Timer 在计数器模式下，对应的外部输入引脚每输入一个脉冲，计数器值加 1，如果开启了中断，当计数器溢出时产生中断。因此，计数器中断和初值配置是相关的，下面两种配置是常用的两种场景。

1) 只想得到外部脉冲的计数：这时可以设置计数器初值为 0，不用开启中断，当需要获取计数值时读取 Timer 的计数寄存器即可。
2) 每个外部脉冲都产生一次中断：这时可以配置计数器初值为 0xFFFF，开启中断，外部输入引脚，输入一个脉冲后计数器溢出产生中断。

2.3.3. 启动和停止计数器

Timer 可以配置为定时器或计数器，无论在哪种模式下，Timer0 都是通过置位“定时器 0/1 控制寄存器（TCON）”中的“TR0 位”启动，通过清零“TR0 位”停止，启动后，定时器将继续从他之前被停止时的值继续计数。

1) 定时器 0/1 的启动和停止：由 TCON 寄存器中的 TR0 和 TR1 位控制。

定时器 0/1 控制寄存器（TCON）：

符号	地址	B7	B6	B5	B4	B3	B2	B1	B0
TCON	88H	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

- TR1：定时器 T1 的运行控制位，该位由软件置位和清零。当 GATE（TMOD.7）=0，TR1=1 时就允许 T1 开始计数，TR1=0 时禁止 T1 计数。当 GATE（TMOD.7）=1，TR1=1 且 INT1 输入高电平时，才允许 T1 计数。
- TR0：定时器 T0 的运行控制位，该位由软件置位和清零。当 GATE（TMOD.3）=0，

TR0=1 时就允许 T0 开始计数，TR0=0 时禁止 T0 计数。当 GATE（TMOD.3）=1，TR0=1 且 INT0 输入高电平时，才允许 T0 计数，TR0=0 时禁止 T0 计数。

2) 定时器 2 的启动和停止：由“辅助寄存器 1（AUXR）”寄存器中的 T2R 位控制。

辅助寄存器 1（AUXR）：

符号	地址	B7	B6	B5	B4	B3	B2	B1	B0
AUXR	8EH	T0x12	T1x12	UART_M0x6	T2R	T2_C/T	T2x12	EXTRAM	S1ST2

■ T2R：定时器 2 的运行控制位。

0：定时器 2 停止计数。

1：定时器 2 开始计数。

3) 定时器 3/4 的启动和停止：由“定时器 4/3 控制寄存器（T4T3M）”寄存器中的 T4R 和 T3R 位控制。

定时器 4/3 控制寄存器（T4T3M）：

符号	地址	B7	B6	B5	B4	B3	B2	B1	B0
T4T3M	D1H	T4R	T4_C/T	T4x12	T4CLKO	T3R	T3_C/T	T3x12	T3CLKO

■ T4R：定时器 4 的运行控制位。

0：定时器 4 停止计数。

1：定时器 4 开始计数。

■ T3R：定时器 3 的运行控制位

0：定时器 3 停止计数。

1：定时器 3 开始计数。

✍ **示例**：启动定时器 0 计数

```
TR0 = 1;        //启动定时器 0 计数
```

✍ **示例**：停止定时器 0 计数

```
TR0 = 0;        //停止定时器 0 计数
```

3. 软件设计

3.1. 555 频率测量实验（数码管显示）

✧ 注：本节的实验是在“实验 2-9-1：DS18B20 温度传感器 - 数码管显示”的基础上修改，本节对应的实验源码是：“实验 2-12-1：NE555 频率测量 - 数码管显示”。

3.1.1. 实验内容

1) IAP15F2K61S2/IAP15W4K61S4 单片机计数器 0 的外部输入引脚 P3.4 通过短路帽连接 555 定时器振荡电路的输出引脚 SIGNAL，这样可测试振荡电路输出信号的频率。

2) 配置 Timer0 作为计数器使用，实现对 P3.4 引脚上脉冲信号的计数。

3) 开启定时器 1，定时 1s 去读计数器 0 计数的个数，该值即是所测信号的频率，可通过数码管显示出来。

3.1.2. 代码编写

1. 新建一个名称为“timer.c”的文件及其头文件“timer.h”并保存到工程的“Source”文件夹，并将“timer.c”加入到 Keil 工程中的“SOURCE”组。
2. 引用头文件

因为在“main.c”文件中使用了“timer.c”文件中的函数，所以需要引用下面的头文件“timer.h”。

代码清单：引用头文件

```
1.  //引用 timer 的头文件
2.  #include    " timer.h"
```

3. 计数器初始化

将 Timer0 配置为计数器后，P3.4 即为其外部输入引脚。

本例中，我们是对外部输入脉冲计数，因此，计数器初值设置为 0，并且不需要开启中断，计数器初始化代码清单如下。

代码清单：计数器初始化

```
1.  /*******************************************************************************
2.  功能描述：初始化 Timer0 为计数器，开启 P3.4 的内部上拉电阻
3.  参    数：无
4.  返 回 值：无
5.  *******************************************************************************/
6.  void timer0_counter_init(void)
7.  {
8.      TMOD = 0x04;        //配置 Timer0 为计数器
9.      TL0 = 0x00;         //计数器初始值设置为 0
10.     TH0 = 0x00;         //计数器初始值设置为 0
11.     TF0 = 0;            //清除 TF0 标志
12. }
```

4. 读取计数值

当我们需要读取计数值的时候，先关掉计数器 0，然后从“TH0”和“TL0”读取计数值，然后清零“TH0”和“TL0”寄存器，以方便下次读取。最后会再次打开计数器 0，代码清单如下。

代码清单：读取计数值

```
1. /*******************************************************************************
2.  * 描  述 : 读取当前计数值
3.  * 入  参 : 无
4.  * 返回值 : 读取的计数值
5.  *******************************************************************************/
6. u16 get_timer0_count(void)
7. {
8.     u8 tempH,tempL;
9.     timer0_stop();
```

```
    tempH = TH0;              //读取计数值
    tempL = TL0;
    TL0 = 0x00;                 //计数器初始值设置为 0
    TH0 = 0x00;                 //计数器初始值设置为 0
    timer0_start();           //启动 timer0
    return (tempH<<8)+tempL; //返回读取的计数值

}
```

5. 主函数

主函数中完成相关的初始化，之后，在主循环中由定时器 1 定时 1s 读取一次计数器 0 的计数信息，并通过数码管显示测量结果，代码清单如下。

代码清单：主函数

```
/*********************************************************************************
功能描述：主函数
入口参数：无
返 回 值：int 类型
*********************************************************************************/
int main(void)
{
    u16 conut_number; //用来保存读取的计数值
    static  u8   frebuff[6];

    P2M1 &= 0x1F;   P2M0 |= 0xE0;     //设置 P2.5、P2.6、P2.7 为推挽输出
    P0M1 &= 0x00;   P0M0 |= 0xFF;     //设置 P0.0 ~ P0.7 为推挽输出

    SEG_off();                    //控制 8 位数码管/点阵不显示
    leds_off();                   //熄灭 D1~D8 指示灯
    ULN2003_off();                //控制 ULN2003 输出高电平，关闭蜂鸣器、继电器等
    delay_ms(10);                 //延时

    timer0_counter_init();        //timer0 初始化
    timer0_start();               //启动 timer0
    timer1_init();                //timer1 初始化
    timer1_start();               //启动 timer1
    timer2_init();                //timer2 初始化
    timer2_start();               //启动 timer2
    EA = 1;                       //使能总中断
    delay_ms(200);

    while(1)
    {
        if(flag)   //1s 时间内的计数器数值即是频率值
```

```
31.     {
32.         conut_number = get_timer0_count();          //读取计数值
33.         //数码管显示读取的计数值
34.         if(conut_number>=10000)     //实测频率值 5 位数
35.         {
36.             frebuff[0]=conut_number%100000/10000;
37.             frebuff[1]=conut_number%100000%10000/1000;
38.             frebuff[2]=conut_number%100000%10000%1000/100;
39.             frebuff[3]=conut_number%100000%10000%1000%100/10;
40.             frebuff[4]=conut_number%100000%10000%1000%100%10;
41.
42.             LEDseg_DispUpdata(LEDSEG_4,frebuff[0],LEDSEG_DP_OFF);//更新第 4 个数码管显示内容
43.             LEDseg_DispUpdata(LEDSEG_5,frebuff[1],LEDSEG_DP_OFF);//更新第 5 个数码管显示内容
44.             LEDseg_DispUpdata(LEDSEG_6,frebuff[2],LEDSEG_DP_OFF);//更新第 6 个数码管显示内容
45.             LEDseg_DispUpdata(LEDSEG_7,frebuff[3],LEDSEG_DP_OFF);//更新第 7 个数码管显示内容
46.             LEDseg_DispUpdata(LEDSEG_8,frebuff[4],LEDSEG_DP_OFF);//更新第 8 个数码管显示内容
47.         }
48.         else if(conut_number>=1000)     //实测频率值 4 位数
49.         {
50.             frebuff[0]=conut_number%10000/1000;
51.             frebuff[1]=conut_number%10000%1000/100;
52.             frebuff[2]=conut_number%10000%1000%100/10;
53.             frebuff[3]=conut_number%10000%1000%100%10%10;
54.
55.             LEDseg_DispUpdata(LEDSEG_4,17,LEDSEG_DP_OFF);//更新第 4 个数码管显示内容
56.             LEDseg_DispUpdata(LEDSEG_5,frebuff[0],LEDSEG_DP_OFF);//更新第 5 个数码管显示内容
57.             LEDseg_DispUpdata(LEDSEG_6,frebuff[1],LEDSEG_DP_OFF);//更新第 6 个数码管显示内容
58.             LEDseg_DispUpdata(LEDSEG_7,frebuff[2],LEDSEG_DP_OFF);//更新第 7 个数码管显示内容
59.             LEDseg_DispUpdata(LEDSEG_8,frebuff[3],LEDSEG_DP_OFF);//更新第 8 个数码管显示内容
60.         }
61.         else if(conut_number>=100)      //实测频率值 3 位数
62.         {
63.             frebuff[0]=conut_number%1000/100;
64.             frebuff[1]=conut_number%1000%100/10;
65.             frebuff[2]=conut_number%1000%100%10%10;
66.
67.             LEDseg_DispUpdata(LEDSEG_4,17,LEDSEG_DP_OFF);//更新第 4 个数码管显示内容
68.             LEDseg_DispUpdata(LEDSEG_5,17,LEDSEG_DP_OFF);//更新第 5 个数码管显示内容
69.             LEDseg_DispUpdata(LEDSEG_6,frebuff[0],LEDSEG_DP_OFF);//更新第 6 个数码管显示内容
70.             LEDseg_DispUpdata(LEDSEG_7,frebuff[1],LEDSEG_DP_OFF);//更新第 7 个数码管显示内容
71.             LEDseg_DispUpdata(LEDSEG_8,frebuff[2],LEDSEG_DP_OFF);//更新第 8 个数码管显示内容
72.         }
73.         else if(conut_number>=10)      //实测频率值 2 位数
```

```
            {
                frebuff[0]=conut_number%100/10;
                frebuff[1]=conut_number%100%10;

                LEDseg_DispUpdata(LEDSEG_4,17,LEDSEG_DP_OFF);//更新第 4 个数码管显示内容
                LEDseg_DispUpdata(LEDSEG_5,17,LEDSEG_DP_OFF);//更新第 5 个数码管显示内容
                LEDseg_DispUpdata(LEDSEG_6,17,LEDSEG_DP_OFF);//更新第 6 个数码管显示内容
                LEDseg_DispUpdata(LEDSEG_7,frebuff[0],LEDSEG_DP_OFF);//更新第 7 个数码管显示内容
                LEDseg_DispUpdata(LEDSEG_8,frebuff[1],LEDSEG_DP_OFF);//更新第 8 个数码管显示内容
            }
            else
            {
                frebuff[0]=conut_number%100%10;

                LEDseg_DispUpdata(LEDSEG_4,17,LEDSEG_DP_OFF);//更新第 4 个数码管显示内容
                LEDseg_DispUpdata(LEDSEG_5,17,LEDSEG_DP_OFF);//更新第 5 个数码管显示内容
                LEDseg_DispUpdata(LEDSEG_6,17,LEDSEG_DP_OFF);//更新第 6 个数码管显示内容
                LEDseg_DispUpdata(LEDSEG_7,17,LEDSEG_DP_OFF);//更新第 7 个数码管显示内容
                LEDseg_DispUpdata(LEDSEG_8,frebuff[0],LEDSEG_DP_OFF);//更新第 8 个数码管显示内容
            }
            flag = 0;           //标志清零
        }
    }
}
```

3.1.3. 硬件连接

本实验程序的编写都是基于 IO 模式，所以 J24 端子需要使用短路帽将该端子第 1 引脚和第 2 引脚短接，即选择为 IO 模式。同时 J6 端子需要使用短路帽将该端子第 1 引脚和第 2 引脚短接，即选择独立按键。另外，本实验 Timer0 的外部输入引脚是 P3.4，故可以将 J25 端子的 P34 与 SIGNAL 短接，如下图所示。

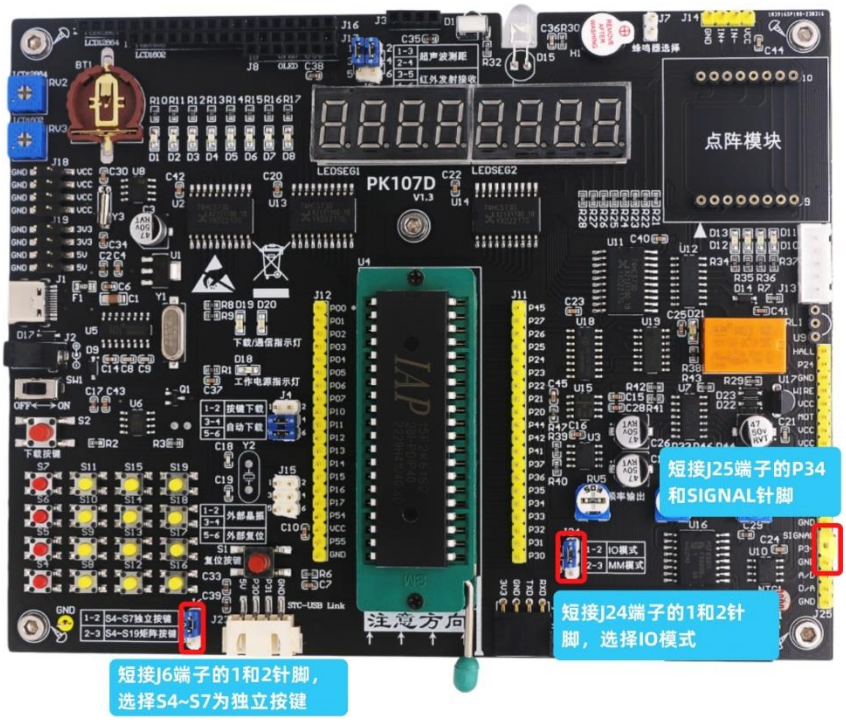

图 5：硬件连接

3.1.4. 实验步骤

1) 解压“…\第 3 部分：配套例程源码”目录下的压缩文件“实验 2-12-1：NE555 频率测量 - 数码管显示”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”（这样做的目的是为了防止中文路径或者工程存放的路径过深导致打开工程出现问题）。
2) 双击“…\ne555\project”目录下的工程文件“ne555.uvproj”。
3) 点击编译按钮编译工程，编译成功后生成的 HEX 文件“ne555.hex”位于工程的“…\ne555\Project\Object”目录下。
4) 打开 STC-ISP 软件下载程序，下载使用内部 IRC 时钟，IRC 频率选择：12MHz。
5) 程序运行后，可以在数码管上实时显示待测信号频率值。可用螺丝刀旋转 RV5 可调电阻以改变待测信号频率，观察数码管上频率测量值的变化情况。

✧ **说明：**555 定时器振荡频率测量信息也可以通过串口调试助手显示。

我们也编写好了串口调试助手显示频率信息的例程，该例程在资料的“…\第 3 部分：配套例程源码目录下，其实验名称如下，读者在编写的过程中可以参考。

- 实验 2-12-2：NE555 频率测量 - 串口显示。