第 2-8 讲：4×4 矩阵按键识别

1. 学习目的

1. 了解独立按键和矩阵按键的区别，掌握矩阵按键线反转法识别原理。
2. 掌握 IAP15F2K61S2/IAP15W4K61S4 使用线反转法识别 4×4 矩阵按键的程序设计。

2. 矩阵按键原理

单片机设计中，按键作为一种常用的人机接口被广泛应用，在家用、娱乐、工控等设备上都可以见到按键的应用。我们最常用的两种按键接入方式是独立按键和矩阵按键。

2.1. 独立按键和矩阵按键

独立按键如下图所示，每个按键占用单片机的一个 GPIO，单片机通过读取 GPIO 的状态即可判断按键是否按下，这种方式编程简单，适合在按键较少的场合下使用。

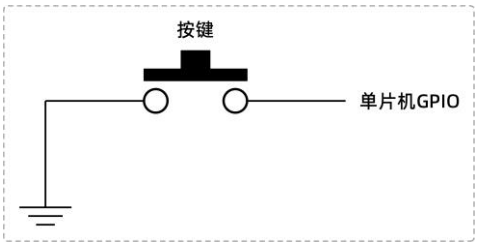

图 1：独立按键

当按键较多时，使用独立按键会占用单片机过多的 IO 资源，这会降低单片机 IO 的利用率，不利于单片机的选型（我们必须选择具有大量 IO 资源的单片机才能完成独立按键的接入），同时，也使得硬件布线变得复杂。

为了解决这个问题，当按键较多时，我们通常会使用矩阵按键方式接入。矩阵按键接入是利用单片机的 GPIO 口组成行与列，在行与列的每一个交点处连接按键，该接入方式最大优势是提高了 IO 的利用率。根据行和列的不同可以有很多种矩阵按键组合，如 3x3 矩阵按键、4×4 矩阵按键等等。其中，典型的是 4×4 矩阵按键，本章我们以 4×4 矩阵按键为例来说明矩阵按键扫描原理和编程，下图是常用的两种 4×4 矩阵按键模块。

图 2：4×4 矩阵按键

2.2. 矩阵按键识别原理

下图是 4×4 矩阵按键的原理示意图，可以看到每个按键都有对应的行号和列号，通过行号和列号的组合就可以确定是哪个按键，如行 1 列 2 就可以确定按键 S2。由此，我们自然想到，如果行和列都用数字信号表示（0 和 1），那么每个按键都可以通过一个数字组合来表示，这个数字组合就是按键对应的编码。

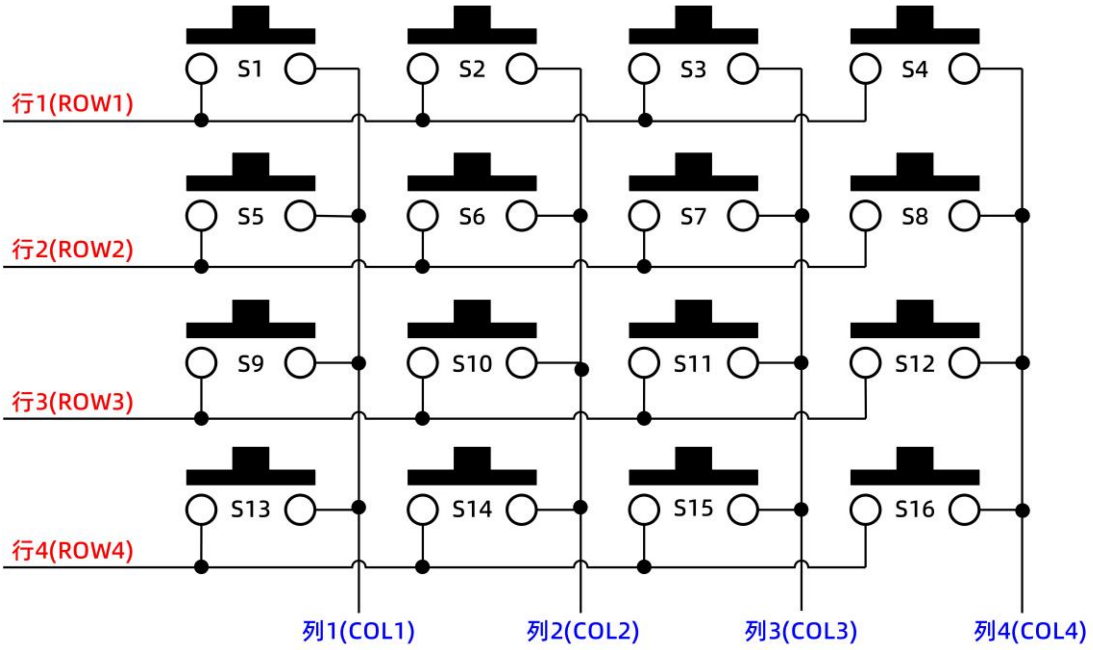

图 3：4×4 矩阵按键原理示意图

矩阵按键的识别通常需要两步操作，第一步是检测键盘上是否有键按下，第二步是识别具体是哪一个按键按下。对于矩阵按键的识别，常用的方法有行列扫描法和线反转法，其中，线反转法只需 2 步即可完成按键识别，速度快于行列扫描法，接下来，我们以线反转法为例，具体看一下矩阵按键的检测过程。

1) 行线（ROW1~ ROW3）配置为输出、列线（COL1~ COL4）配置为输入。行线输出低电平，读取列线数据，如果列线数据全为“1”，则无按键按下，如果有按键按下，则对应列的输入为 0。如 S1 键按下，可以得到列编码为 1110。
2) 线翻转，即行线配置为输入、列线配置为输出。列线输出低电平，读取行线数据，由此获取按下的按键对应的行号。
3) 由行线和列线即可得到具体是哪个按键按下。

3. 硬件电路设计

PK107D 开发板上设计了 4×4 矩阵按键电路，如下图所示。

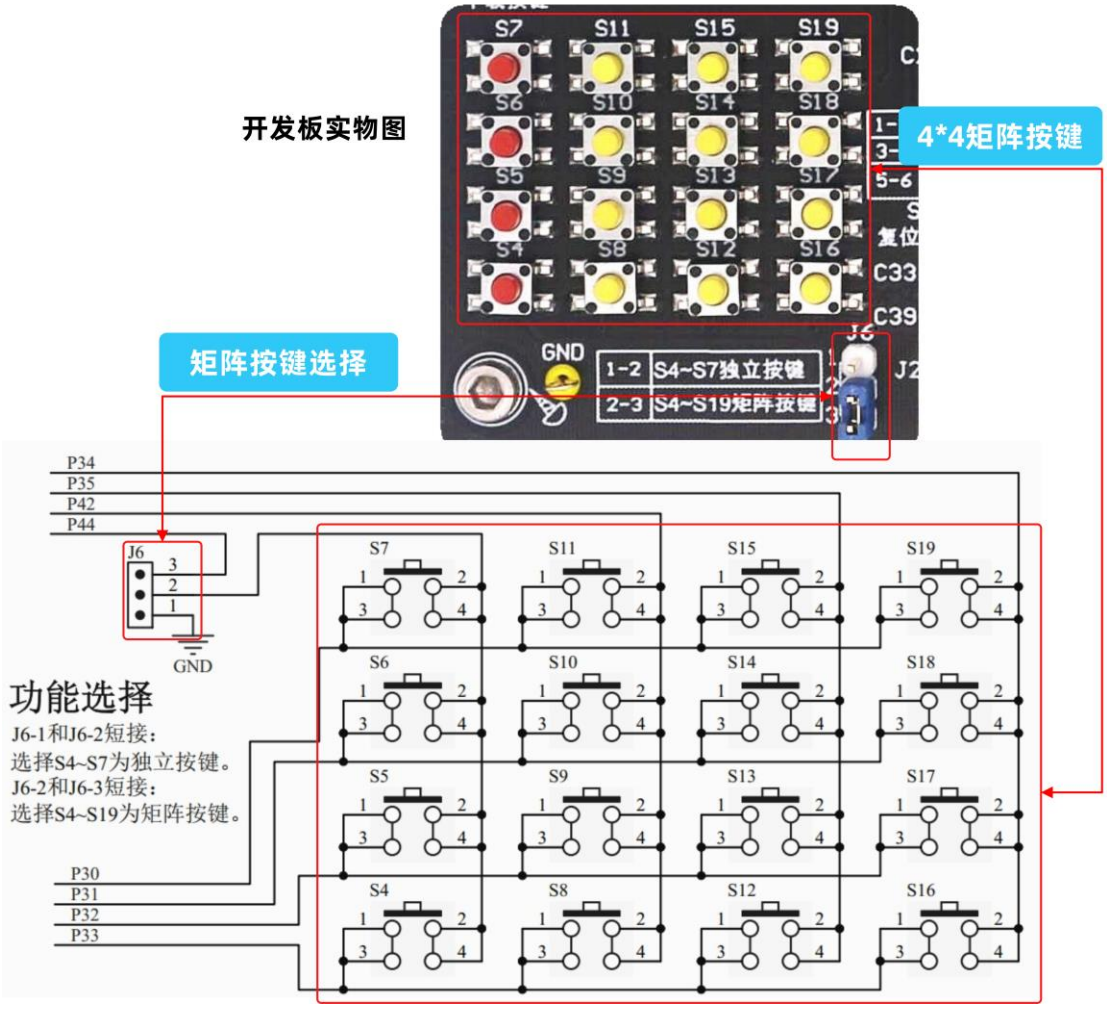

图 4：4×4 矩阵按键电路原理图

矩阵按键使用的单片机的引脚如下表：

表 1：IAP15F2K61S2/IAP15W4K61S4 单片机矩阵按键引脚分配

名称	编号	引脚	说明
行	行 1	P3.3	非独立 IO
	行 2	P3.2	非独立 IO
	行 3	P3.1	非独立 IO
	行 4	P3.0	非独立 IO
列	列 1	P4.4	非独立 IO
	列 2	P4.2	非独立 IO
	列 3	P3.5	非独立 IO
	列 4	P3.4	非独立 IO

✧ 注：独立 GPIO 表示开发板没有其他的电路使用这个 GPIO，非独立 GPIO 说明开发板有其他电路用到了该 GPIO。读者在使用非独立 GPIO 使用时需要注意电路的连接，以避免多个电路使用了同一个 GPIO。

4. 软件设计

4.1. 4×4 矩阵按键检测实验

✧ 注：本节的实验是在“实验 2-5-1：串口 1 数据收发实验”的基础上修改，本节对应的实验源码是：“实验 2-8-1：4×4 矩阵按键检测”。

4.1.1. 实验内容

用线反转方式识别矩阵按键，成功识别后，通过串口输出按键的键值（S4~S19）。

4.1.2. 代码编写

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

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

代码清单：引用头文件

```
1.  //引用矩阵按键的头文件
2.  #include   "keyboard.h"
```

3. 4×4 矩阵按键识别

线反转法识别矩阵按键代码清单如下，因为开发板要考虑综合资源分配，使用的 GPIO 不连续，所以，IO 配置相对麻烦一些，读者在自己的设计中使用矩阵按键时，如果条件允许，应尽量使用连续编号的 GPIO。

代码清单：4×4 矩阵按键识别

```
1. /************************************************************************
2. 功能描述：读取开发板上的 8 个矩阵按键(S4~S19)的状态
3. 参     数：无
4. 返 回 值：有按键按下，返回对应的按键编号，否则返回 BUTTONS_RELEASED(没有按键按下)
5.  ************************************************************************/
6. u8 keyboard_scan(void)
7. {
8.         static u8 X_temp,Y_temp,temp;
9.
10.        X_temp=0xF0;          //列值赋初值
11.        Y_temp=0x0F;          //行值赋初值
12.
13.        P3M1 &= 0xC0;   P3M0 |= 0x3F;     //设置 P3.0~P3.5 为强推挽输出
14.        P4M1 &= 0xEB;   P4M0 |= 0x14;     //设置 P4.2，P4.4 为强推挽输出
15.
16.        ROW1=1;ROW2=1;ROW3=1;ROW4=1;  //行置高
17.        COL1=0;COL2=0;COL3=0;COL4=0;  //列置低
18.
```

```
19.         //所用到行 IO 口配置为输入，进行检测
20.         delay_ms(10);
21.         P3M1 &= 0xF0;   P3M0 &= 0xF0;     //设置 P3.0~P3.3 为准双向口
22.         delay_ms(10);
23.
24.         if(ROW1 == 0)         //检测行 1 电平是否为低电平
25.         {
26.             delay_ms(10);
27.             if(ROW1 == 0)
28.             Y_temp &= 0x0E;
29.         }
30.         if(ROW2 == 0)         //检测行 2 电平是否为低电平
31.         {
32.             delay_ms(10);
33.             if(ROW2 == 0)
34.             Y_temp &= 0x0D;
35.         }
36.         if(ROW3 == 0)         //检测行 3 电平是否为低电平
37.         {
38.             delay_ms(10);
39.             if(ROW3 == 0)
40.             Y_temp &= 0x0B;
41.         }
42.         if(ROW4 == 0)         //检测行 4 电平是否为低电平
43.         {
44.             delay_ms(10);
45.             if(ROW4 == 0)
46.             Y_temp &= 0x07;
47.         }
48.
49.
50.         P3M1 &= 0xC0;   P3M0 |= 0x3F;     //设置 P3.0~P3.5 为强推挽输出
51.         P4M1 &= 0xEB;   P4M0 |= 0x14;     //设置 P4.2，P4.4 为强推挽输出
52.
53.         ROW1=0;ROW2=0;ROW3=0;ROW4=0;  //行置低
54.         COL1=1;COL2=1;COL3=1;COL4=1;  //列置高
55.
56.         //所用到列 IO 口配置为输入，进行检测
57.         delay_ms(10);
58.         P3M1 &= 0xCF;   P3M0 &= 0xCF;     //设置 P3.4~P3.5 为准双向口
59.         P4M1 &= 0xEB;   P4M0 &= 0xEB;     //设置 P4.2、P4.4 为准双向口
60.         delay_ms(10);
61.
```

```
62.         if(COL1 == 0)          //检测列 1 电平是否为低电平
63.         {
64.             delay_ms(10);
65.             if(COL1 == 0)
66.             X_temp &= 0xE0;
67.         }
68.         if(COL2 == 0)          //检测列 2 电平是否为低电平
69.         {
70.             delay_ms(10);
71.             if(COL2 == 0)
72.             X_temp &= 0xD0;
73.         }
74.         if(COL3 == 0)          //检测列 3 电平是否为低电平
75.         {
76.             delay_ms(10);
77.             if(COL3 == 0)
78.             X_temp &= 0xB0;
79.         }
80.         if(COL4 == 0)          //检测列 4 电平是否为低电平
81.         {
82.             delay_ms(10);
83.             if(COL4 == 0)
84.             X_temp &= 0x70;
85.         }
86.
87.         //将行值和列值合并，得到按键对应的编码值，该值与 16 个按键一一对应
88.         temp = X_temp|Y_temp;
89.         temp = ~temp;
90.
91.         //将按键检测的原始编码值解析对应按键值信息
92.         switch (temp)
93.         {
94.                 case 0x11:return BUTTON1_PRESSED;     //S4 按键
95.                 case 0x12:return BUTTON2_PRESSED;     //S5 按键
96.                 case 0x14:return BUTTON3_PRESSED;     //S6 按键
97.                 case 0x18:return BUTTON4_PRESSED;     //S7 按键
98.                 case 0x21:return BUTTON5_PRESSED;     //S8 按键
99.                 case 0x22:return BUTTON6_PRESSED;     //S9 按键
100.                case 0x24:return BUTTON7_PRESSED;     //S10 按键
101.                case 0x28:return BUTTON8_PRESSED;     //S11 按键
102.                case 0x41:return BUTTON9_PRESSED;     //S12 按键
103.                case 0x42:return BUTTON10_PRESSED;  //S13 按键
104.                case 0x44:return BUTTON11_PRESSED;  //S14 按键
```

```
105.                case 0x48:return BUTTON12_PRESSED;  //S15 按键
106.                case 0x81:return BUTTON13_PRESSED;  //S16 按键
107.                case 0x82:return BUTTON14_PRESSED;  //S17 按键
108.                case 0x84:return BUTTON15_PRESSED;  //S18 按键
109.                case 0x88:return BUTTON16_PRESSED;  //S19 按键
110.                default:  return BUTTONS_RELEASED;
111.            }
112.    }
```

4. 主函数

主函数中，调用 4×4 矩阵按键识别函数实时检测是否有按键按下，如果有按键按下，通过数码管显示按键的编号（S4~S19）代码清单如下。

代码清单：主函数

```
1. /*****************************************************************************
2. 功能描述：主函数
3. 入口参数：无
4. 返回值：int 类型
5.  *****************************************************************************/
6. int main(void)
7. {
8.     u8 temp;
9.
10.    P2M1 &= 0x1F;   P2M0 |= 0xE0;     //设置 P2.5、P2.6、P2.7 为推挽输出
11.    P0M1 &= 0x00;   P0M0 |= 0xFF;     //设置 P0.0 ~ P0.7 为推挽输出
12.    P3M1 &= 0xC0;   P3M0 &= 0xC0;     //设置 P3.0 ~ P3.5 为准双向口
13.    P4M1 &= 0xEB;   P4M0 &= 0xEB;     //设置 P4.2、P4.4 为准双向口
14.    ULN2003_off();     //控制步进电机、蜂鸣器、继电器等不工作
15.    leds_off();        //熄灭 D1~D8 指示灯
16.    SEG_off();         //控制 8 位数码管/点阵不显示
17.    delay_ms(10);      //延时
18.
19.    timer2_init();     //定时器 2 初始化
20.    timer2_start();    //启动定时器 2
21.    EA = 1;            //使能总中断
22.    delay_ms(200);
23.    LEDseg_off();      //关闭数码管显示
24.
25.    while(1)
26.    {
27.        temp = keyboard_scan();           //获取开发板矩阵按键检测值
28.
```

```
29.         if(temp == BUTTON1_PRESSED)         //按键 S4 按下
30.         {
31.             LEDseg_DispUpdata(LEDSEG_7,17,LEDSEG_DP_OFF);    //更新第 7 个数码管显示内容
32.             LEDseg_DispUpdata(LEDSEG_8,temp,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
33.         }
34.         else if(temp == BUTTON2_PRESSED)    //按键 S5 按下
35.         {
36.             LEDseg_DispUpdata(LEDSEG_7,17,LEDSEG_DP_OFF);    //更新第 7 个数码管显示内容
37.             LEDseg_DispUpdata(LEDSEG_8,temp,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
38.         }
39.         else if(temp == BUTTON3_PRESSED)    //按键 S6 按下
40.         {
41.             LEDseg_DispUpdata(LEDSEG_7,17,LEDSEG_DP_OFF);    //更新第 7 个数码管显示内容
42.             LEDseg_DispUpdata(LEDSEG_8,temp,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
43.         }
44.         else if(temp == BUTTON4_PRESSED)    //按键 S7 按下
45.         {
46.             LEDseg_DispUpdata(LEDSEG_7,17,LEDSEG_DP_OFF);    //更新第 7 个数码管显示内容
47.             LEDseg_DispUpdata(LEDSEG_8,temp,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
48.         }
49.         else if(temp == BUTTON5_PRESSED)    //按键 S8 按下
50.         {
51.             LEDseg_DispUpdata(LEDSEG_7,17,LEDSEG_DP_OFF);    //更新第 7 个数码管显示内容
52.             LEDseg_DispUpdata(LEDSEG_8,temp,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
53.         }
54.         else if(temp == BUTTON6_PRESSED)    //按键 S9 按下
55.         {
56.             LEDseg_DispUpdata(LEDSEG_7,17,LEDSEG_DP_OFF);    //更新第 7 个数码管显示内容
57.             LEDseg_DispUpdata(LEDSEG_8,temp,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
58.         }
59.         else if(temp == BUTTON7_PRESSED)    //按键 S10 按下
60.         {
61.             LEDseg_DispUpdata(LEDSEG_7,17,LEDSEG_DP_OFF);    //更新第 7 个数码管显示内容
62.             LEDseg_DispUpdata(LEDSEG_8,temp,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
63.         }
64.         else if(temp == BUTTON8_PRESSED)    //按键 S11 按下
65.         {
66.             LEDseg_DispUpdata(LEDSEG_7,17,LEDSEG_DP_OFF);    //更新第 7 个数码管显示内容
67.             LEDseg_DispUpdata(LEDSEG_8,temp,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
68.         }
69.         else if(temp == BUTTON9_PRESSED)    //按键 S12 按下
70.         {
71.             LEDseg_DispUpdata(LEDSEG_7,17,LEDSEG_DP_OFF);    //更新第 7 个数码管显示内容
```

```
72.             LEDseg_DispUpdata(LEDSEG_8,temp,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
73.         }
74.         else if(temp == BUTTON10_PRESSED)   //按键 S13 按下
75.         {
76.             LEDseg_DispUpdata(LEDSEG_7,temp/10,LEDSEG_DP_OFF); //更新第 7 个数码管显示内容
77.             LEDseg_DispUpdata(LEDSEG_8,temp%10,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
78.         }
79.         else if(temp == BUTTON11_PRESSED)   //按键 S14 按下
80.         {
81.             LEDseg_DispUpdata(LEDSEG_7,temp/10,LEDSEG_DP_OFF); //更新第 7 个数码管显示内容
82.             LEDseg_DispUpdata(LEDSEG_8,temp%10,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
83.         }
84.         else if(temp == BUTTON12_PRESSED)   //按键 S15 按下
85.         {
86.             LEDseg_DispUpdata(LEDSEG_7,temp/10,LEDSEG_DP_OFF); //更新第 7 个数码管显示内容
87.             LEDseg_DispUpdata(LEDSEG_8,temp%10,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
88.         }
89.         else if(temp == BUTTON13_PRESSED)   //按键 S16 按下
90.         {
91.             LEDseg_DispUpdata(LEDSEG_7,temp/10,LEDSEG_DP_OFF); //更新第 7 个数码管显示内容
92.             LEDseg_DispUpdata(LEDSEG_8,temp%10,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
93.         }
94.         else if(temp == BUTTON14_PRESSED)   //按键 S17 按下
95.         {
96.             LEDseg_DispUpdata(LEDSEG_7,temp/10,LEDSEG_DP_OFF); //更新第 7 个数码管显示内容
97.             LEDseg_DispUpdata(LEDSEG_8,temp%10,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
98.         }
99.         else if(temp == BUTTON15_PRESSED)   //按键 S18 按下
100.        {
101.                LEDseg_DispUpdata(LEDSEG_7,temp/10,LEDSEG_DP_OFF); //更新第 7 个数码管显示内容
102.                LEDseg_DispUpdata(LEDSEG_8,temp%10,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
103.            }
104.            else if(temp == BUTTON16_PRESSED)   //按键 S19 按下
105.            {
106.                LEDseg_DispUpdata(LEDSEG_7,temp/10,LEDSEG_DP_OFF); //更新第 7 个数码管显示内容
107.                LEDseg_DispUpdata(LEDSEG_8,temp%10,LEDSEG_DP_OFF); //更新第 8 个数码管显示内容
108.            }
109.        }
110.    }
```

4.1.3. 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 2 引脚和第 3 引脚短接，即选择矩阵按键。

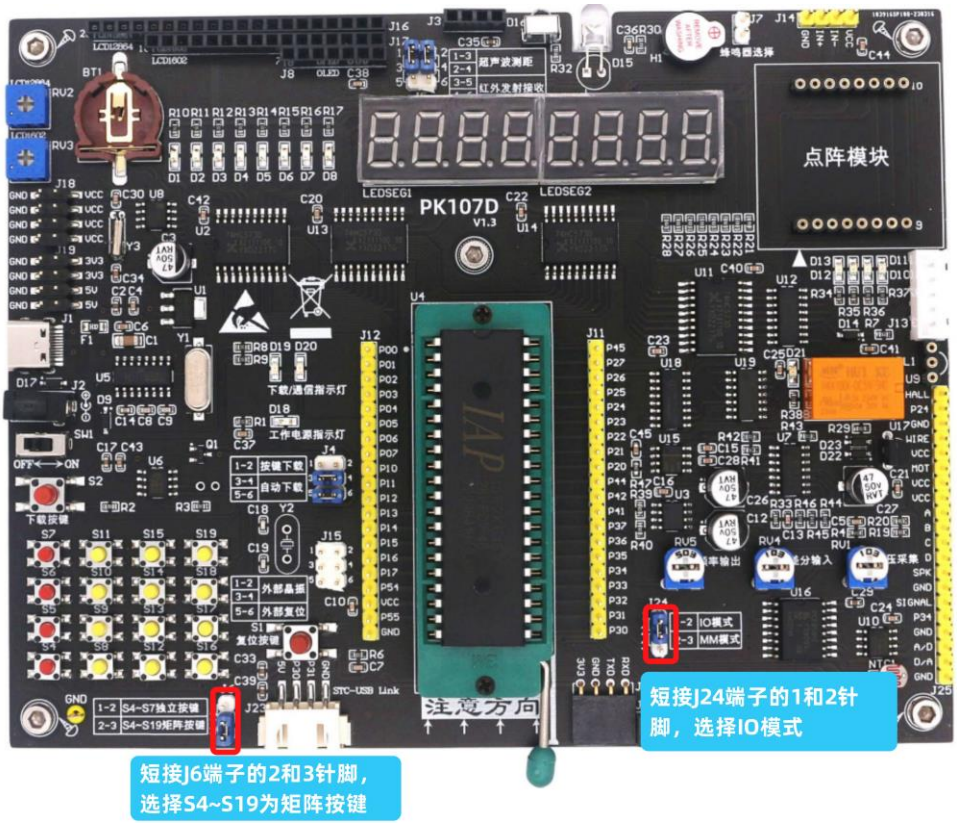

图 5：硬件连接

4.1.4. 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 2-8-1：4×4 矩阵按键检测”，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“…\keyboard \project”目录下的工程文件“keyboard.uvproj”。
3) 点击编译按钮编译工程，编译成功后生成的 HEX 文件“keyboard.hex”位于工程的“…\ keyboard\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，按动 4×4 矩阵按键中的按键，可以观察到数码管上会显示对应的键值。

我们也编写好了 2×4 矩阵按键和 2×2 矩阵按键的例子，这些例子在资料的“…\第 3 部分：配套例程源码”目录下，他们的实验名称如下，读者在编写的过程中可以参考。

- 实验 2-8-2：2x4 矩阵按键检测。
- 实验 2-8-3：2x2 矩阵按键检测。