GPIO 输入按键检测

1. 实验目的

- 掌握轻触按键检测电路的设计以及硬件原理，掌握高低电平式按键检测的方法，了解 ADC 通过电阻分压检测多个按键的原理。
- 掌握编写轻触按键检测程序，检测单个按键状态。

2. 实验内容

- 编写程序实现单个轻触按键的检测。

3. 硬件电路设计

3.1. 开发板用户按键硬件电路

轻触按键又称轻触开关（下文简称按键），是电路中常用的一种开关元器件，也是一种常用的人机接口。广泛用于家电、数码产品、便携仪、电脑等电子设备中。

进取者 STC15 开发板上设计了 1 个用户按键 S3，需短接 J27 端子将该用户按键 S3 连接到单片机的 P5.4 引脚。程序中通过读取这些按键对应的 GPIO 的状态（高电平或低电平）可判断该按键是否按下，这种电路的形式称为高低电平接法，这种检测按键的方法称为按键高低电平检测。

图 1：开发板按键检测电路及实物图

✧ 1 个用户按键占用的单片机的引脚如下表：

表 1：用户按键引脚分配

KEY	颜色	引脚	说明
按键 S3	白色	P5.4	非独立 GPIO

✧ 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。进取者 STC15 开发板上 W5500 模块用到了 P5.4 口。

轻触按键，顾名思义我们只需要施加很小的力量即可改变开关连接的状态。轻触按键在

所需外力作用下（按下按键）触点导通，无外力作用时（释放按键）触点断开，如下图所示：

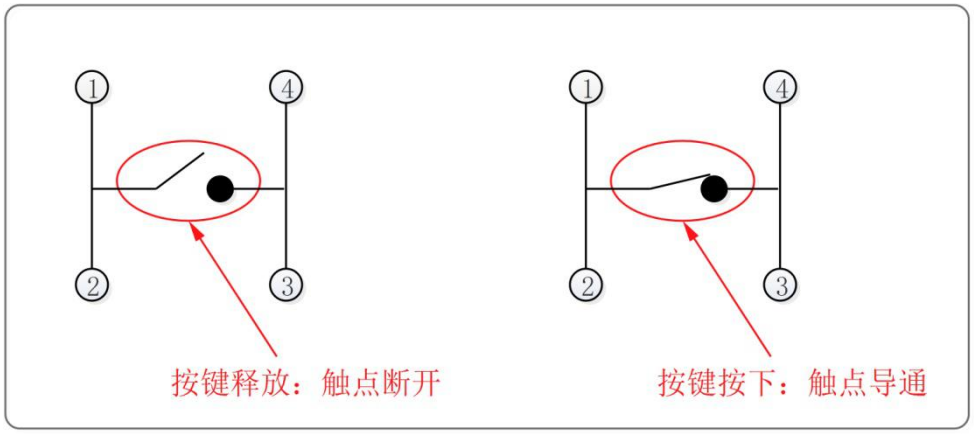

图 2：轻触按键原理

3.2. 按键检测接法

高低电平式接法是最常见的按键检测的接法，顾名思义，该接法就是需要单片机引脚具有高低电平的检测能力，也就是常见的 GPIO 引脚即可。高低电平式接法又可分为两种：独立式接法和行列式接法。

行列式接法是利用单片机的 GPIO 口组成行与列，在行与列的每一个交点处连接按键。故也称为矩阵式按键，该接线方法最大的优势是可以使用较少的 GPIO 口实现较多按键的检测，这个在传感器实验部分矩阵按键实验中会详细介绍。

独立式接法的含义就是使用单片机的一个 GPIO 引脚检测一个按键的状态，有多少按键需检测就需要多少个 GPIO 引脚，对每个按键的检测相互独立。

独立式按键接法一般会用低电平有效的方式，即按键按下是 GPIO 输入为低电平，如下图所示。

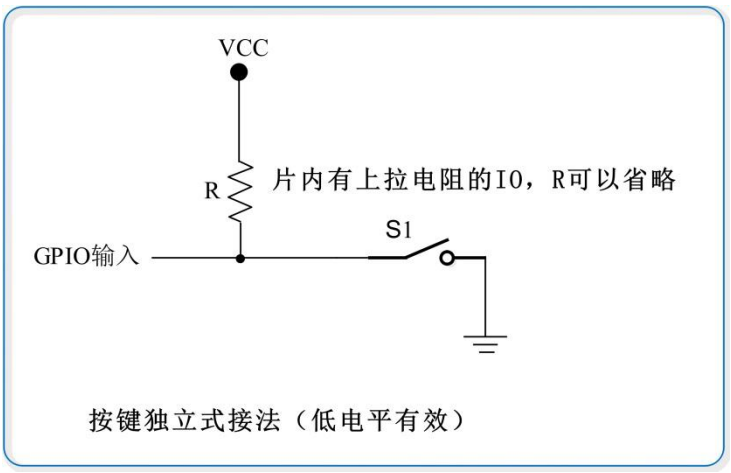

图 3：独立式接法

上图中的电阻 R 的作用是将 GPIO 输入端口不确定的信号通过该电阻钳位在高电平状态。

我们知道数字电路有三种状态：高电平、低电平和高阻状态，有些应用场合并不希望出现高阻状态，这时加上拉电阻即可让 GPIO 输入端口保持确定的状态。

按键释放时，因为上拉电阻 R 的关系，GPIO 输入检测是高电平，按键闭合时，GPIO 短接到 GND，输入检测是低电平。这样，单片机根据 GPIO 的输入状态即可确定按键是否按下。

■ **按键检测知识扩展：ADC 通过电阻分压检测多个按键**

按键检测除了高低电平检测的方法之外，还有一种方法是使用 ADC 通过电阻分压检测多个按键，这种按键检测的电路形式称为分压式接法。

分压式接法，使用的单片机引脚必须具有 ADC 功能，根据检测口测得的不同的电压值来识别是哪个按键被按下。如下图所示，是分压式接法的原理示意图。这种方法最大的好处是节省 IO 资源，它只需一个具有 ADC 功能的 IO 即可实现多个按键的检测，它适用于 IO 资源紧张的场合，如一些电磁炉的按键使用的就是这种方法。

相对于高低电平检测，这种方法在编程上要复杂一些，需要事先计算好分压的电压值，存储于“表”中，程序运行时，采样到电压值后查表即可获知是哪个按键按下。

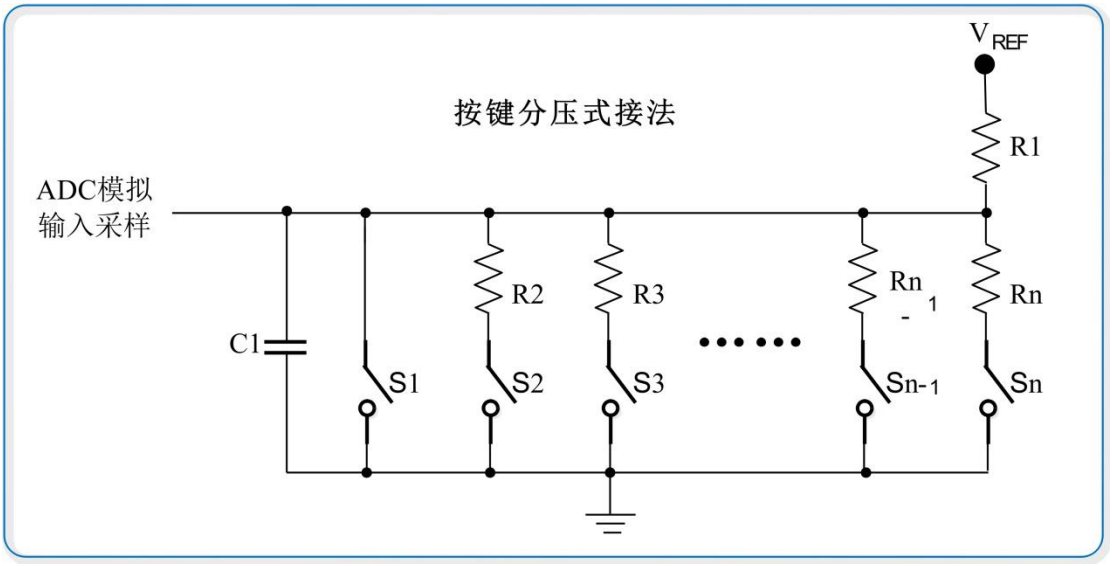

图 4：分压式接法原理

3.3. 按键检测电路考虑因素

按键检测电路设计的时候，需要我们考虑两个方面：按键释放时 GPIO 口状态的确定和按键消抖。

1) 按键释放时 GPIO 口状态的确定

按键检测电路中，当按键释放后要能保证 GPIO 口电平是确定的，即按键释放时 GPIO 口固定为高电平或低电平。开发板 R13 电阻就是满足按键 S3 释放时，在 P5.4 口上保持高电平。

2) 按键消抖

对于按键硬件上的消抖，一般常用的方式是在按键上并接一个容值约 0.1uF 左右电容，利用电容两端的电压不能突变的特性，消除抖动时产生的毛刺电压。虽然电容可以起到消除抖动的作用，但是在考虑按键灵敏度的情况下，电容是无法完全消除抖动的，消除抖动还需要软件的配合。

开发板上按键电路没有增加硬件消抖，开发板使用的是软件消抖，这对于一般的按键检测已经完全足够。

3) GPIO 口保护

开发板按键检测电路中还有一个电阻 R11，该电阻阻值 100 欧姆，串在单片机 GPIO 口和按键引脚中，起到保护 GPIO 口的目的。

分析：如果 GPIO 口不小心误配置为输出模式，并且输出高电平，则分析下电路可知，此时如果没有电阻 R11，如果按下按键 S3，则单片机 GPIO 口（控制输出高电平）直接和 GND 相连，会损坏 GPIO 口。

4. 软件设计

4.1. 寄存器解析

下图是对端口数据寄存器 P0、P1、P2、P3、P4、P5、P6、P7 的描述，端口数据寄存器各位代表对应端口的 GPIO 口，在完成对配置寄存器设置后，可直接读取端口引脚电平。

符号	地址	B7	B6	B5	B4	B3	B2	B1	B0
P0	80H	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.0
P1	90H	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
P2	A0H	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0
P3	B0H	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.0
P4	C0H	P4.7	P4.6	P4.5	P4.4	P4.3	P4.2	P4.1	P4.0
P5	C8H	-	-	P5.5	P5.4	P5.3	P5.2	P5.1	P5.0
P6	E8H	P6.7	P6.6	P6.5	P6.4	P6.3	P6.2	P6.1	P6.0
P7	F8H	P7.7	P7.6	P7.5	P7.4	P7.3	P7.2	P7.1	P7.0

图 5：端口数据寄存器

4.2.GPIO 输入按键检测实验（单个 c 文件）

✧ 注：本节的实验源码是在“实验 2-1-3：流水灯（单个 c 文件）”的基础上修改。本节对应的实验源码是：“实验 2-2-1： GPIO 输入按键检测（单个 c 文件）”。

4.2.1. 头文件引用和路径设置

■ 需要宏定义部分及引用的头文件

因为在“main.c”文件中使用了 STC15 的头文件“15W4KxxS4.h”，所以需要引用下面的头文件。在头文件“15W4KxxS4.h”中需要确定主时钟取值，所以宏定义主时钟值。

```
1.  #define MAIN_Fosc       11059200L   //定义主时钟
2.  #include    "15W4KxxS4.H"
```

在程序设计中会用到定义变量的类型，为了定义变量方便，将较为复杂的“unsigned int”和“unsigned char ”进行了宏定义。

```
1.  #define  uint16   unsigned int
2.  #define  uint8    unsigned char
```

这样，再定义变量时可直接使用“uint16”和“uint8”来取代“unsigned int”和“unsigned char ”即可。

■ 需要包含的头文件路径

本例需要包含的头文件路径如下表：

表 2：头文件包含路径

序号	路径	描述
1	..\User	15W4KxxS4.h 头文件在该路径，所以要包含。

MDK 中点击魔术棒，打开工程配置窗口，按照下图所示添加头文件包含路径。

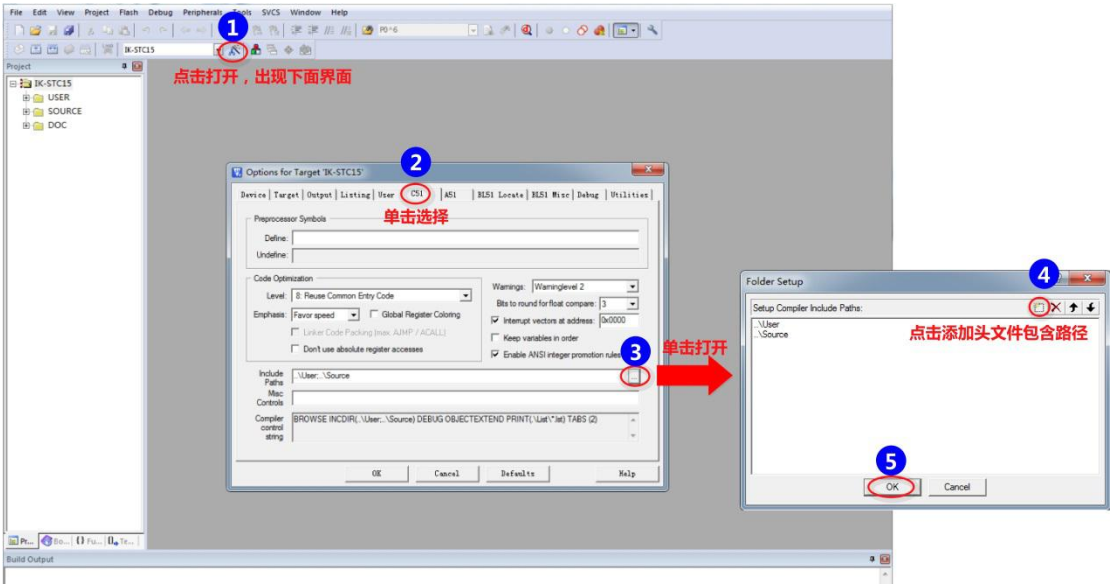

图 6：添加头文件包含路径

4.2.2. 编写代码

首先根据开发板按键及指示灯 GPIO 分配，定义寄存器位变量，代码如下。

```
1.  /***********************
2.  引脚别名定义
3.  ***********************/
4.  sbit KEY=P5^4;      //用户按键 S3 用 IO 口 P54
5.  sbit LED_B=P0^6;    //蓝色 LED 用 IO 口 P06
```

然后，在主函数中对 P0.6 和 P5.4 口进行模式配置，检测主循环中按键状态，确认按键按下可控制兰色指示灯 DS1 亮。

代码清单：主函数

```
1.  int main(void)
2.  {
3.  ////////////////////////////////////////////////
4.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
5.  //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
6.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
7.  //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
8.  ////////////////////////////////////////////////
9.      P0M1 &= 0xBF;   P0M0 &= 0xBF;     //设置 P0.6 为准双向口
10.     P5M1 &= 0xEF;   P5M0 &= 0xEF;     //设置 P5.4 为准双向口
11.     //P5M1 |= 0x10; P5M0 &= 0xEF;     //设置 P5.4 为高阻输入
```

```
12.
13.   while(1)
14.   {
15.         if(KEY == 0)          //检测用户按键 S3 对应引脚 P5.4 是否是低电平 (按键按下，
      引脚为低电平)
16.       {
17.         delay_ms(10);       //软件延时 10ms，如果延时后按键 S3 的电平依然没有变化，说明
      按键确实被有效操作，简称按键消抖
18.         if(KEY== 0)           //检测用户按键 S3 对应引脚 P5.4 是否依然是低电平
19.         {
20.             LED_B=0;                  //点亮蓝色指示灯 DS1
21.             while(KEY == 0)          //等待按键 S3 释放，即如果 P5.4 一直为低电平，会
      一直执行空命令
22.             {
23.                 ;                  //条件 KEY == 0 成立，会执行这个空命令
24.             }
25.         LED_B=1;                  //熄灭蓝色指示灯 DS1
26.        }
27.       }
28.    }
29. }
```

4.2.3. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-2-1：GPIO 输入按键检测（单个 c 文件）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\key\projec”目录下的工程“key.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“key.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，按下按键 S3 可以观察到兰色 LED 灯被点亮。松开按键 S3，兰色 LED 灯熄灭。

■ **思考题 1：**分析软件消抖是如何实现的？

4.3.流水灯实验（多个 c 文件）

✧ 注：本节的实验源码是在“实验 2-2-1： GPIO 输入按键检测（单个 c 文件）”的基础上修改。本节对应的实验源码是：“实验 2-2-2： GPIO 输入按键检测（多个 c 文件）”。

4.3.1. 工程需要用到的 c 文件

本例需要用到的 c 文件如下表所示，工程需要添加下表中的 c 文件。

表 3：实验需要用到的 c 文件

序号	文件名	后缀	功能描述
1	led	.c	包含与用户 led 控制有关的用户自定义函数。
2	key	.c	包含与用户按键检测有关的用户自定义函数。
3	delay	.c	包含用户自定义延时函数。

4.3.2. 头文件引用和路径设置

■ 需要引用的头文件

因为在“main.c”文件中使用了控制 led 的函数和延时函数（延时函数没有在 main.c 中定义），所以需要引用下面的头文件。

```
1. #include    "led.h"
2. #include    "delay.h"
3. #include    "key.h"
```

■ 需要包含的头文件路径

本例需要包含的头文件路径如下表：

表 4：头文件包含路径

序号	路径	描述
1	..\ Source	led.h、key.h 和 delay.h 头文件在该路径，所以要包含。
2	..\User	15W4KxxS4.h 头文件在该路径，所以要包含。

MDK 中点击魔术棒，打开工程配置窗口，按照下图所示添加头文件包含路径。

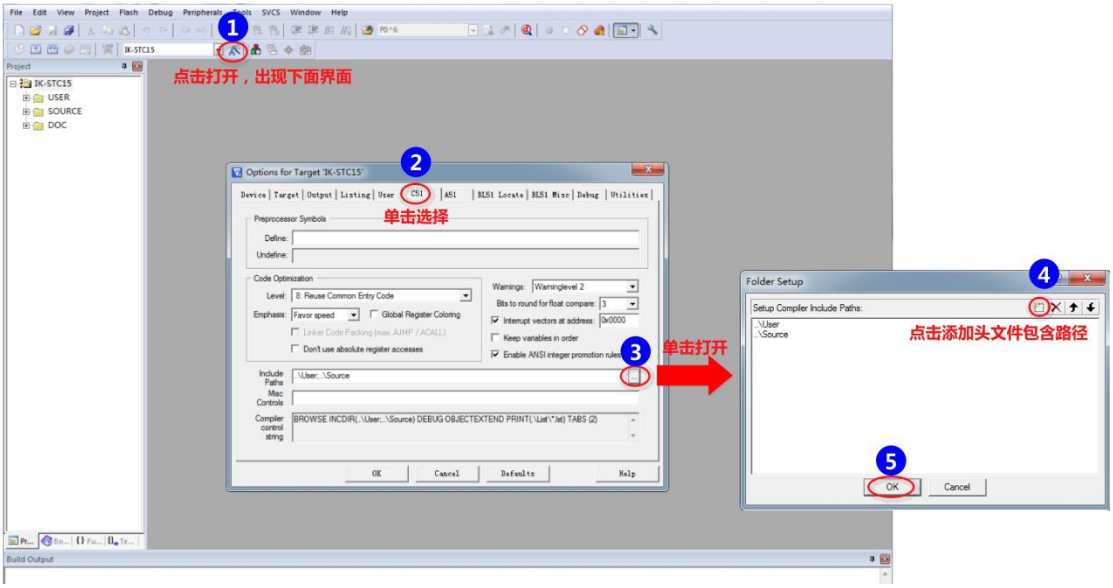

图 7：添加头文件包含路径

4.3.3. 编写代码

首先在 key.h 中，定义寄存器位变量 P5^4，引用延时函数的头文件，声明按键检测函数供外部调用。代码如下。

```
#include "delay.h"

/***********************
引脚别名定义
***********************/
sbit KEY=P5^4;      //用户按键 S3 用 IO 口 P5.4

extern void KEY_Scan(void);
```

然后，在 key.c 文件中编写一个按键检测函数 KEY_Scan，代码如下。

程序清单：延时函数

```
/*********************************************************************************
功能描述：按键检测
入口参数：无
返回值：无
 *********************************************************************************/
void KEY_Scan(void)
{
      if(KEY == 0)   //检测用户按键 S3 对应引脚 P5.4 是否是低电平（按键按下，引脚为低电平）
      {
```

```
        delay_ms(10);      //软件延时 10ms，如果延时后按键 S3 的电平依然没有变化，说明按键确实被有效操作，简称按键消抖
        if(KEY== 0)        //检测用户按键 S3 对应引脚 P5.4 是否依然是低电平
        {
            led_on(LED_1);           //点亮蓝色指示灯 DS1
            while(KEY == 0)   //等待按键 S3 释放，即如果 P5.4 一直为低电平，会一直执行空命令
                {
                    ;                       //条件 KEY == 0 成立，会执行这个空命令
                }
        led_off(LED_1);          //熄灭蓝色指示灯 DS1
        }
    }
}
```

因为按键检测函数 KEY_Scan 中使用了控制 led 的函数、KEY 的位变量和延时函数，所以需要在 key.c 文件中引用下面的头文件。

```
#include "led.h"
#include "key.h"
```

■ **思考题 1：**既然用到了延时函数，为什么没有`#include "delay.h"` 呢？

最后，在主函数中对 P0.6 和 P5.4 口进行模式配置，并在主循环中调用按键检测函数，可观察按下按键 S3 兰色指示灯 DS1 亮。

代码清单：主函数

```
int main(void)
{
//////////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
//////////////////////////////////////////////////
    P0M1 &= 0xBF;   P0M0 &= 0xBF;     //设置 P0.6 为准双向口
    P5M1 &= 0xEF;   P5M0 &= 0xEF;     //设置 P5.4 为准双向口
    //P5M1 |= 0x10; P5M0 &= 0xEF;     //设置 P5.4 为高阻输入

  while(1)
  {
```

```
15.         KEY_Scan();     //按键输入检测函数
16.     }
17. }
```

4.3.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-2-2：GPIO 输入按键检测（多个 c 文件）”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\key\projec”目录下的工程“key.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“key.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，按下按键 S3 可以观察到兰色 LED 灯被点亮。松开按键 S3，兰色 LED 灯熄灭。