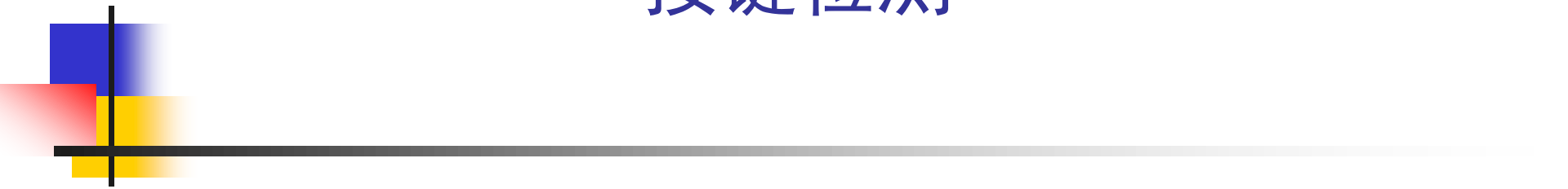
实验四：GPIO输入

--按键检测

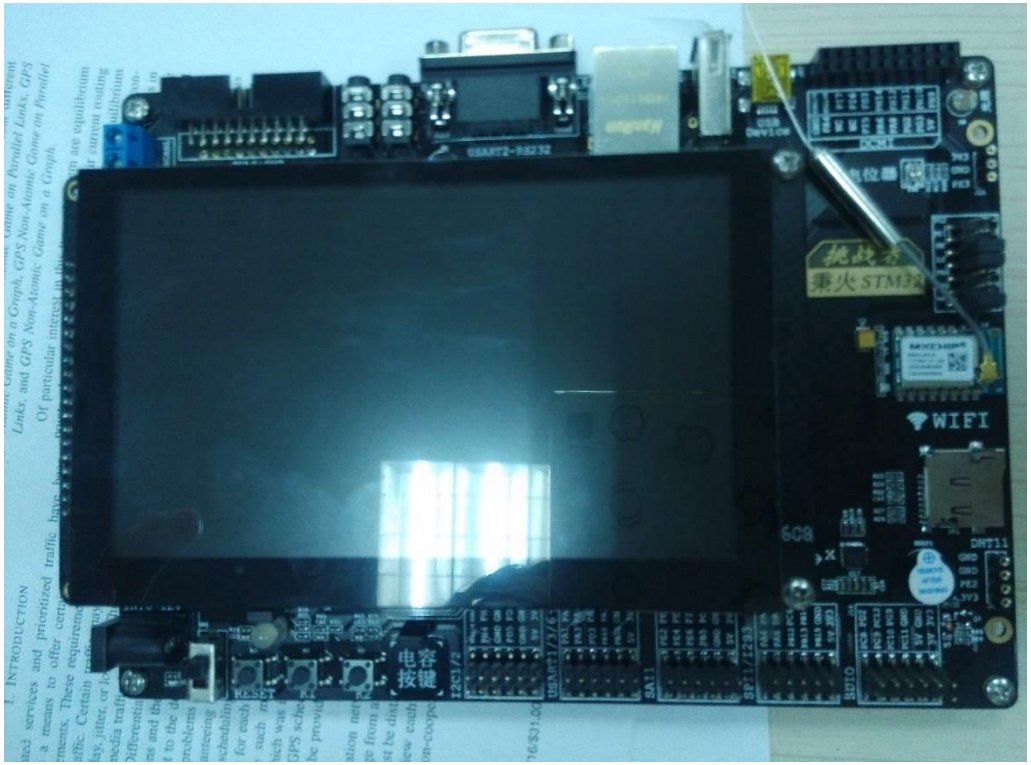


实验目的

- 掌握如何对按键进行检测，且控制不同按键点亮不同的LED灯

实验设备

- 软件Keil5(keil提供了软件仿真功能)；
- STM32开发板

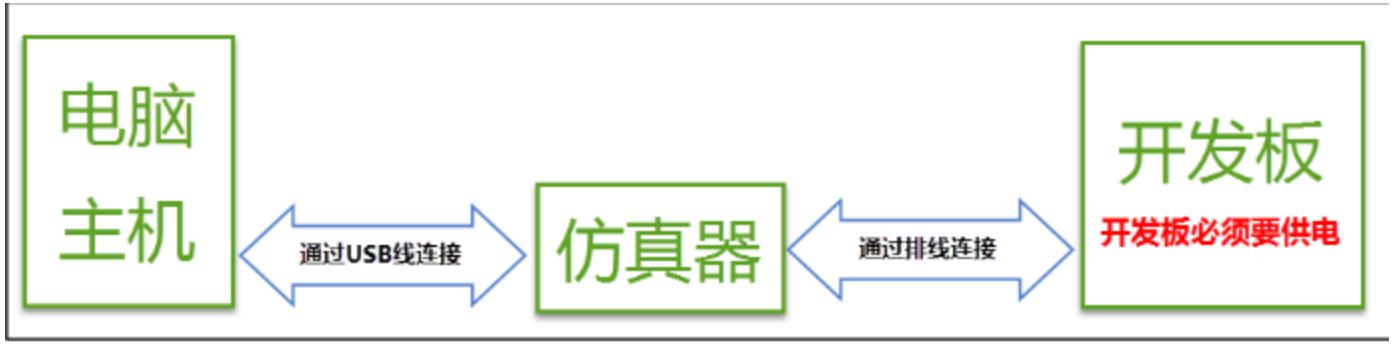


实验内容

- 学习如何操作按键，并能用轮询方式检测按键

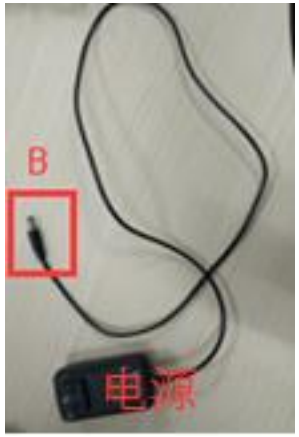
一、连接开发板

把仿真器用USB线连接电脑，如果仿真器的灯亮表示正常，可以使用。然后把仿真器的另外一端连接到开发板，给开发板上电，然后就可以通过软件KEIL给开发板下载程序。



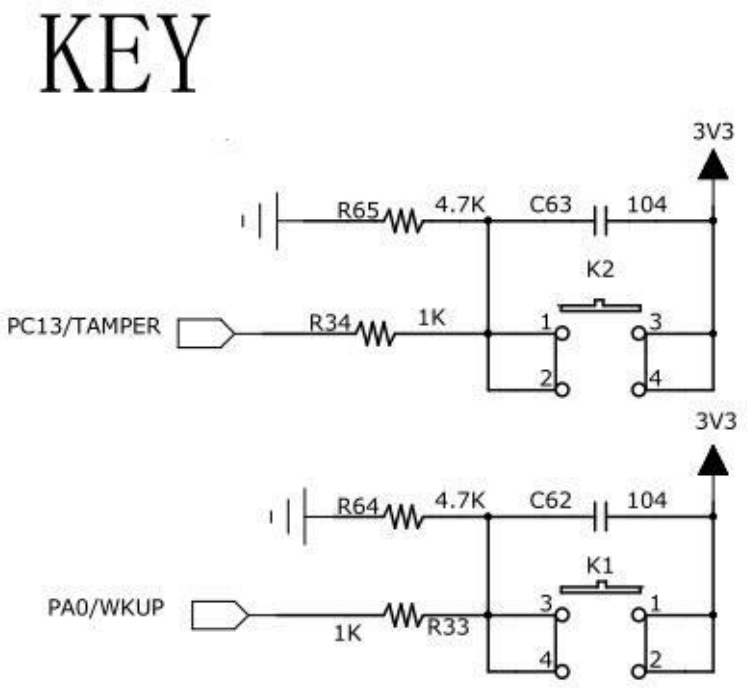
一、连接开发板

现在你们手里有三样东西，一个开发板，一个仿真器和一个电源，如图所示，将仿真器的A端插入开发板的A端，注意正反面，正面的仿真器插头有长方形凸起，仿真器的另一端用USB连接电脑，电源的B口插入开发板的B口，另一端插入电源。C处是开发板的电源开关，拨上去后LED灯亮则连接正常。



二、按键检测

1.硬件设计—按键原理图



从按键的原理图可知

- 按键在没有被按下的时候，GPIO引脚的输入状态为低电平(按键所在的电路不通，引脚接地)
- 当按键按下时，GPIO引脚的输入状态为高电平(按键所在的电路导通，引脚接到电源)。
- 只要我们检测引脚的输入电平，即可判断按键是否被按下。

二、按键检测

2.软件设计—编程要点

（1）使能GPIO端口时钟；

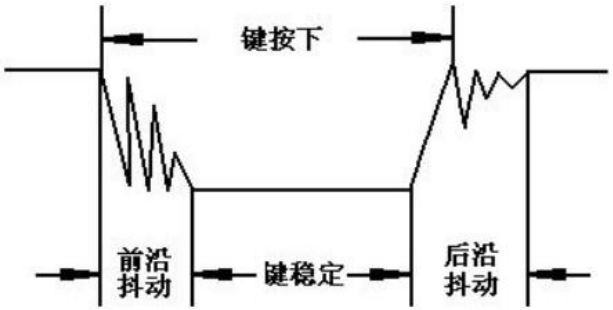
（2）初始化GPIO目标引脚为输入模式(引脚默认电平受按键电路影响，浮空/上拉/下拉均没有区别)；

（3） 编写简单测试程序，检测按键的状态，实现按键控制LED灯。

二、按键检测

2.软件设计—消抖

- 按键机械触点断开、闭合时，由于触点的弹性作用，按键开关不会马上稳定接通或一下子断开，使用按键时会产生带波纹信号，需要用软件消抖处理滤波，不方便输入检测。



二、按键检测

2.软件设计—消抖方法

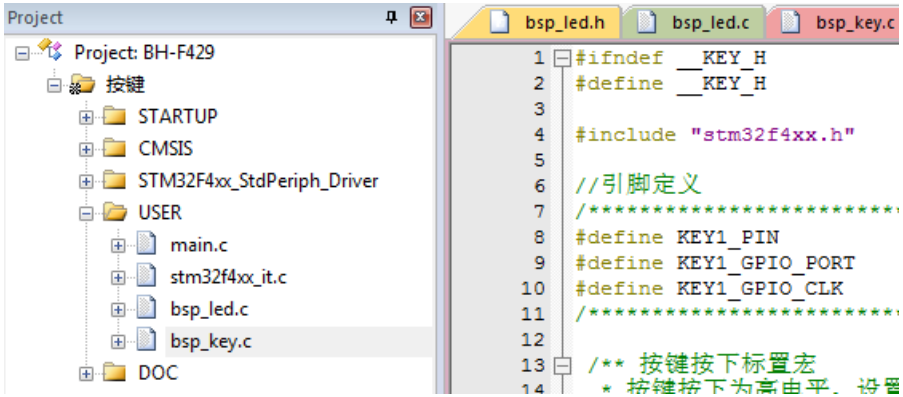
- 软件消抖：就是当检测到按键状态变化后，先等待一个延时时间，让抖动消失后再进行一次按键状态检测，如果与刚才检测到的状态相同，就可以确认按键已经稳定了。

- 硬件消抖：本实验板连接的按键带硬件消抖功能，它利用电容充放电的延时，消除了波纹，从而简化软件的处理，软件只需要直接检测引脚的电平即可。

二、按键检测

3. 示例工程分析：如何通过按键来控制LED灯亮灭

找到工程模板，在工程目录下找到后缀为“.uvprojx”的文件，用KEIL5打开即可。打开该工程，该项目涉及2个外设：按键和LED灯，需2个驱动文件：**bsp_led.c** 和**bsp_key.c**。在项目模版中有**stm32f4xx_it.c**和**main.c**，需创建创建**bsp_led.c** 和**bsp_key.c**，以及相应的**.h**文件。下面对两个.c文件进行简单的讲解。



二、按键检测

3. 示例工程分析—bsp_key.h

在分析bsp_key.c和main.c文件之前，先来看一下bsp_key.h头文件，代码见下页ppt…

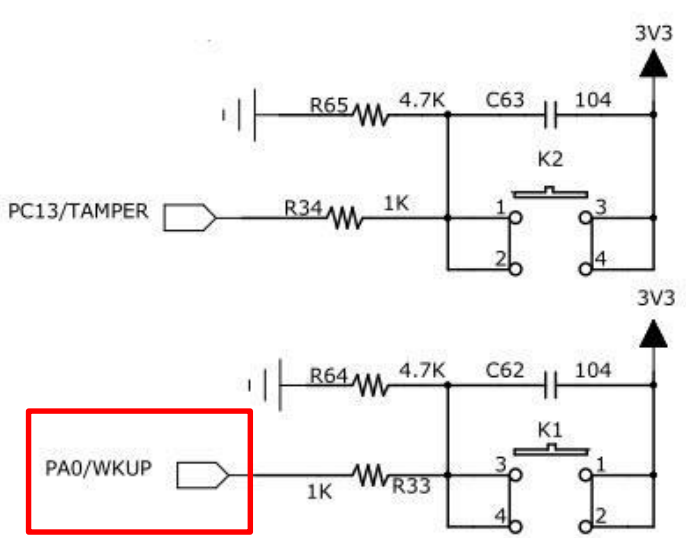
注意：对于硬件相关的各自建立自己的文件夹，如key, led文件夹，分别放“KEY，LED” 的文件夹中

二、按键检测

3.示例工程分析—bsp_key.h—按键引脚宏定义

```
6  //引脚定义
7  /*******************************************************/
8  #define KEY1_PIN                  GPIO_Pin_0
9  #define KEY1_GPIO_PORT            GPIOA
10 #define KEY1_GPIO_CLK             RCC_AHB1Periph_GPIOA
11 /*******************************************************/
```

- 以上代码根据按键1的硬件连接，把检测按键输入的GPIO端口、GPIO引脚号以及GPIO端口时钟封装起来了。



二、按键检测

3.示例工程分析—bsp_key.c—按键GPIO初始化函数

```
31 void Key_GPIO_Config(void)
32 {
33   GPIO_InitTypeDef GPIO_InitStructure;
34
35   /*开启按键GPIO口的时钟*/
36   RCC_AHB1PeriphClockCmd(KEY1_GPIO_CLK,ENABLE);
37
38   /*选择按键的引脚*/
39   GPIO_InitStructure.GPIO_Pin = KEY1_PIN;
40
41   /*设置引脚为输入模式*/
42   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN;
43
44   /*设置引脚不上拉也不下拉*/
45   GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_NOPULL;
46
47   GPIO_Init(KEY1_GPIO_PORT, &GPIO_InitStructure);
48
49 }
```

利用上面的宏，编写按键的初始化函数，函数执行流程与“使用固件库点亮LED灯”类似，这里不再阐述。

二、按键检测

3.示例工程分析—bsp_key.c—检测按键的状态

```
/**
  * @brief   检测是否有按键按下
  * @param   GPIOx:具体的端口，x可以是（A...K）
  * @param   GPIO_PIN:具体的端口位，  可以是GPIO_PIN_x（x可以是0...15）
  * @retval  按键的状态
  *   @arg KEY_ON:按键按下
  *   @arg KEY_OFF:按键没按下
  */
uint8_t Key_Scan(GPIO_TypeDef* GPIOx,uint16_t GPIO_Pin)
{
  /*检测是否有按键按下 */
  if(GPIO_ReadInputDataBit(GPIOx,GPIO_Pin) == KEY_ON )
  {
    /*等待按键释放 */
    while(GPIO_ReadInputDataBit(GPIOx,GPIO_Pin) == KEY_ON);
    return  KEY_ON;
  }
  else
    return KEY_OFF;
}
```

（1）初始化按键后，就可以通过检测对应引脚的电平来判断按键状态了。

（2）Key_Scan函数用于扫描按键状态。

【注】库函数GPIO_ReadInputDataBit来获取位状态，该函数输入GPIO端口及引脚号，函数返回该引脚的电平状态，高电平返回1，低电平返回0。Key_Scan函数中以GPIO_ReadInputDataBit的返回值与自定义的宏“KEY_ON”对比，若检测到按键按下，则使用while循环持续检测按键状态，直到按键释放，按键释放后Key_Scan函数返回一个“KEY_ON”值；若没有检测到按键按下，则函数直接返回“KEY_OFF”。

二、按键检测

4. 库函数GPIO_ReadInputDataBit（）

```
323  uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
324  {
325    uint8_t bitstatus = 0x00;
326
327    /* Check the parameters */
328    assert_param(IS_GPIO_ALL_PERIPH(GPIOx));
329    assert_param(IS_GET_GPIO_PIN(GPIO_Pin));
330
331    if ((GPIOx->IDR & GPIO_Pin) != (uint32_t)Bit_RESET)
332    {
333      bitstatus = (uint8_t)Bit_SET;
334    }
335    else
336    {
337      bitstatus = (uint8_t)Bit_RESET;
338    }
339    return bitstatus;
340  }
341
```

三、LED灯设置

- bsp_led.h和bsp_led.c的设置如上周的实验

四、main函数

3.示例工程分析—main.c—按键检测

```
24 /**
25   * @brief  主函数
26   * @param  无
27   * @retval 无
28   */
29 int main(void)
30 {
31   /* LED 端口初始化 */
32   LED_GPIO_Config();
33
34   /*初始化按键*/
35   Key_GPIO_Config();
36
37
38   /* 轮询按键状态，若按键按下则反转LED */
39   while(1)
40   {
41     if( Key_Scan(KEY1_GPIO_PORT,KEY1_PIN) == KEY_ON  )
42     {
43       /*LED1反转*/
44       LED1_TOGGLE;
45     }
46   }
47 }
```

代码中初始化LED灯及按键后，在while函数里不断调用Key_Scan函数，并判断其返回值，若返回值表示按键按下，则反转LED灯的状态。

五、课堂作业

仿照以上讲的实验例程，在原来实验例程的代码基础上修改代码，以完成第四次课堂小作业：

（1）完成示例实验

（2）示例中是通过key1按键，来控制红灯的亮灭，现在要求通过轮询实现“**key1按键控制绿灯的亮灭，key2按键控制蓝灯的亮灭**”的功能(注意：亮灭意思是按一下亮，再按一下灭)。请问一下将会显示几种颜色的灯亮。

【友情提示】：key2对应的引脚号为GPIOC的13号。