第五讲：GPIO 之按键输入

实验目的：

1，学会使用 gpio 输入

2，理解按键消抖

3，学习吴坚鸿的单片机程序框架

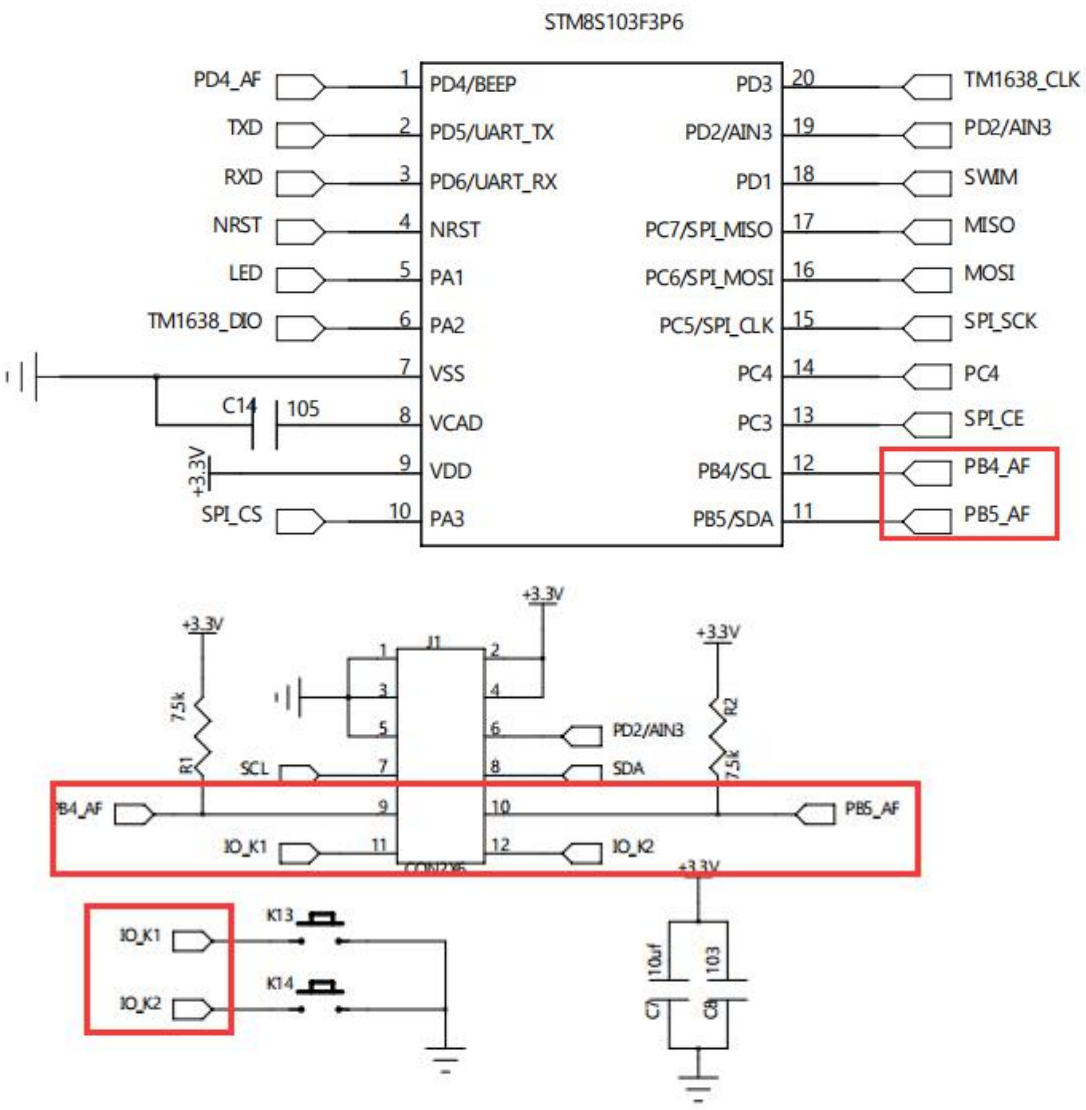
实现功能：

1，当按键按下时，led 亮；松开则 led 灭

2， 双击按键，led 亮；单击则 led 灭

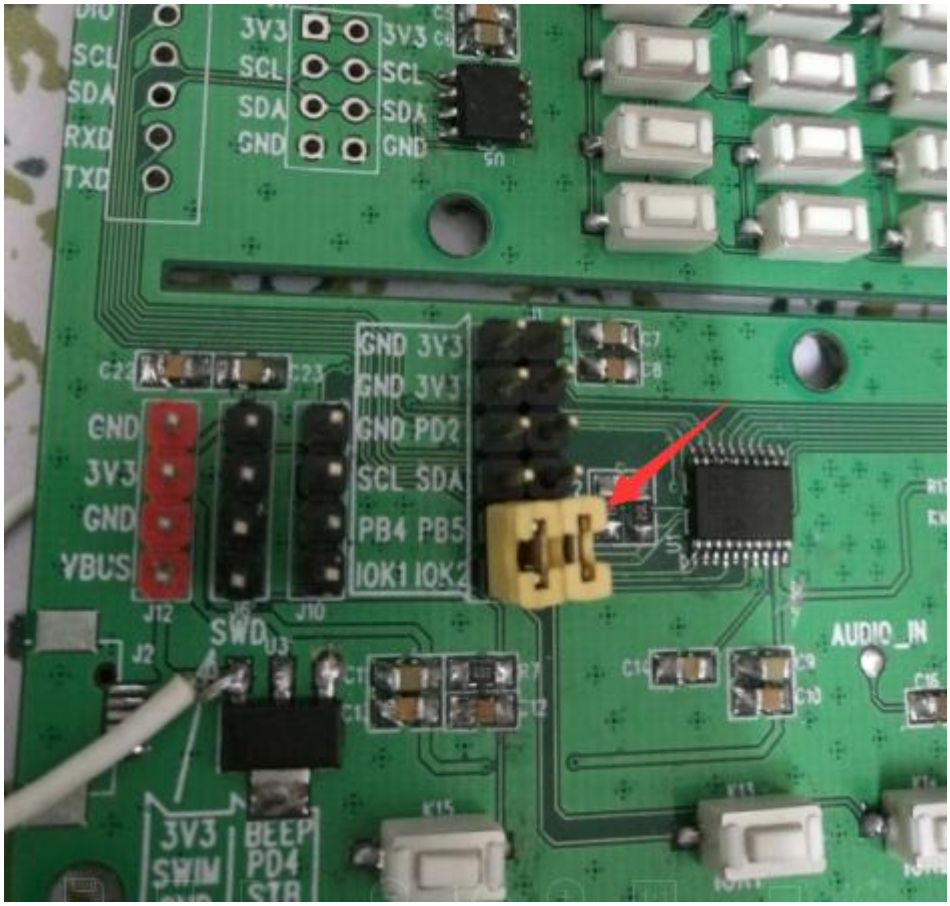
实验过程

1，查看原理图，



i2c与io key共用PB4/PB5，注意跳线

根据原理图示，PB4 和 PB5 需要短路连接到 IO_K1 和 IO_K2 才能使用 gpio 按键，硬件板上见下图：



2，先分析怎么实现第一个功能：当按键按下时，led 亮；松开则 led 灭。
当 key 按下时，PB4 为低电平；Key 松开时 PB4 为高电平。那么我们就检测 PB4 的电平状态，低电平表示按键按下，就点灯；高电平表示未按下，就灭灯。

代码如下：

```c
/**
* author : tianyx
* email  : zzztyx55@sina.com
* qq     : 609421258
* github : https://github.com/zzztyx55
*/
#include "stm8s_conf.h"


/**
* author : tianyx
* email  : zzztyx55@sina.com
* qq     : 609421258
* github : https://github.com/zzztyx55
*/
/* Private defines -----------------------------------------------------------*/
/* Private function prototypes -----------------------------------------------*/
/* Private functions ---------------------------------------------------------*/
void clk_config(void)
{
    CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);  // HSI时钟预分频, 分
    CLK_SYSCLKConfig(CLK_PRESCALER_HSIDIV1);  // 系统时钟配置, HSI, 分频
    CLK_HSICmd(ENABLE);   // 使能HSI
    while(RESET == CLK_GetFlagStatus(CLK_FLAG_HSIRDY));  // 等待HSI read
}

/**
* author : tianyx
* email  : zzztyx55@sina.com
* qq     : 609421258
* github : https://github.com/zzztyx55
*/
void main(void)
{
    // 关中断
    disableInterrupts();
    // 系统时钟配置
    clk_config();

    GPIO_DeInit(GPIOA);
    GPIO_DeInit(GPIOB);

    // PB4, key1, pull up input
    GPIO_Init(GPIOB, GPIO_PIN_4, GPIO_MODE_IN_FL_NO_IT);
    // PA1, led, output
    GPIO_Init(GPIOA, GPIO_PIN_1, GPIO_MODE_OUT_PP_HIGH_SLOW);

    // 开中断
    //enableInterrupts();

    /* Infinite loop */
    while (1)
    {
        if(RESET == GPIO_ReadInputPin(GPIOB, GPIO_PIN_4)) // 低电平,  按
            GPIO_WriteLow(GPIOA, GPIO_PIN_1);  // 点灯
        else
            GPIO_WriteHigh(GPIOA, GPIO_PIN_1); // 灭灯
    }
}
```

3, 编译调试, 当按键按下时，led 亮；松开则 led 灭。

4，再看第二个功能：双击按键，led 亮；单击则 led 灭
这个功能的重点考察 2 点：1 就是怎么处理按键消抖，2 就在于怎么实现区分检测按键双击还是单击。
思路如下：
消抖问题我们采用多次检测按键，如果多次检测按下那么就确定按键按下；
单双击处理方法：先是检测到了按键一次，然后松开后很短一段时间内，检测是否又有按键按下，如果没有，那就是单击，如果有，那就是双击。

代码如下：

```
/**
* author : tianyx
* email  : zzztyx55@sina.com
* qq     : 609421258
* github : https://github.com/zzztyx55
*/
#include "stm8s_conf.h"

/*
* 第一次按下按键后，等待双击的时间
* 如果等待超时，则认为是单击
*/
#define KEY_PRESS_TIMEOUT    50000 //15000
/*
*  按下按键时消抖时间
*/
#define KEY_DOWN_DEBOUNCE     200
/*
*  松开按键时消抖时间
*/
#define KEY_UP_DEBOUNCE       100
/*
*  连续按键中间松开的消抖时间
*/
#define KEY_BETWEEN_UP_DEBOUNCE   10


u8 led_update = 0;
u8 key_press_cnt = 0;

/**
* author : tianyx
* email  : zzztyx55@sina.com
* qq     : 609421258
* github : https://github.com/zzztyx55
*/
/* Private defines -----------------------------------------------------
/* Private function prototypes -----------------------------------------
/* Private functions ---------------------------------------------------
void clk_config(void)
{
    CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);  // HSI时钟预分频,
    CLK_SYSCLKConfig(CLK_PRESCALER_HSIDIV1);  // 系统时钟配置, HSI, 分
    CLK_HSICmd(ENABLE);   // 使能HSI
    while(RESET == CLK_GetFlagStatus(CLK_FLAG_HSIRDY));  // 等待HSI rea
}
```

```c
void key_scan_task(void)
{
    static u8 key_scan_step = 0;
    static u8 down_debounce_cnt = 0;
    static u8 up_debounce_cnt = 0;
    static u32 wait_next_key_cnt = 0;

    switch(key_scan_step)
    {
        case 0:
            if(RESET == GPIO_ReadInputPin(GPIOB, GPIO_PIN_4)) // 低电平
            {
                down_debounce_cnt++;  // 消抖
                // 连续扫描KEY_DEBOUNCE 次 PB4, 都是低电平才算按键按下
                if(down_debounce_cnt >= KEY_DOWN_DEBOUNCE)
                {
                    key_press_cnt = 1;  // 标记按键按下(1次)
                }
            }
            // 按键未按下，但是有标记，说明第一次按下按键已松开
            else if(key_press_cnt == 1)
            {
                //up_debounce_cnt++;
                //if(up_debounce_cnt >= KEY_UP_DEBOUNCE)
                {
                    up_debounce_cnt = 0;
                    down_debounce_cnt = 0; // clear
                    wait_next_key_cnt = 0; // clear
                    key_scan_step = 1;  // 跳转next step, 检测是否再次按下
                }
            }
            else  // 无按键按下，无按下标记
            {
                down_debounce_cnt = 0;  // 清零消抖计数,抗干扰
                up_debounce_cnt = 0;
            }
            break;
        case 1:
            if(RESET == GPIO_ReadInputPin(GPIOB, GPIO_PIN_4)) // 低电平
            {
                // 两次按下之间的松开时间过短，断定为上一步误将抖动当成松开
                if(up_debounce_cnt < KEY_BETWEEN_UP_DEBOUNCE)
                {
                    up_debounce_cnt = 0;
                    break; // 重新检测是否有连续按键
                }

                down_debounce_cnt++;  // 消抖
                // 连续扫描KEY_DEBOUNCE 次 PB4, 都是低电平才算按键按下
                if(down_debounce_cnt >= KEY_DOWN_DEBOUNCE)
                {
                    key_press_cnt = 2;  // 标记连续按键
                    up_debounce_cnt = 0; // clear
                    key_scan_step = 2; // 跳转next step, 等待第二次按键松开
                }
            }
            else          // 按键未按下
            {
                up_debounce_cnt++;
                wait_next_key_cnt++;
                // 等待连续按键超时，说明是按键单击
                if(wait_next_key_cnt >= KEY_PRESS_TIMEOUT)
                {
                    wait_next_key_cnt = 0; // clear
                    up_debounce_cnt = 0; // clear
                    down_debounce_cnt = 0; // clear
                    key_scan_step = 3; // 跳转next step, 报告按键事件
                }
            }

            break;
```

```
        case 2: // 等待第二次按键松开
            if(RESET == GPIO_ReadInputPin(GPIOB, GPIO_PIN_4)) // 低电平
            {
                up_debounce_cnt = 0;
            }
            else  // 松开
            {
                up_debounce_cnt++;
                if(up_debounce_cnt >= KEY_UP_DEBOUNCE)// 消抖
                {
                    key_scan_step = 3; // 跳转next step, 报告按键事件
                }
            }

            break;

        case 3:  //  报告按键事件,单双击由 key_press_cnt 记录
            led_update = 1;   // 通知led_task 有新的按键事件
            wait_next_key_cnt = 0; // clear
            up_debounce_cnt = 0; // clear
            down_debounce_cnt = 0;
            key_scan_step = 0; // 回到step0, 检测新按键
            break;

        default:
            break;
    }
}


/*
*  led_tast 收到led_update != 0的更新通知后,
*  就会检查key_press_cnt 从而知道是单击还是双击
*/
void led_task(void)
{

    if(led_update == 0)
        return;

    led_update = 0; // clear

    if(key_press_cnt == 1) // 按键单击
        GPIO_WriteHigh(GPIOA, GPIO_PIN_1);  // PA1 输出高电平,led灭
    else if(key_press_cnt == 2) // 按键双击
        GPIO_WriteLow(GPIOA, GPIO_PIN_1);   // PA1 输出低电平,led亮

    key_press_cnt = 0; // clear
}
```

```c
/**
* author : tianyx
* email  : zzztyx55@sina.com
* qq     : 609421258
* github : https://github.com/zzztyx55
*/
void main(void)
{
    // 关中断
    disableInterrupts();
    // 系统时钟配置
    clk_config();

    GPIO_DeInit(GPIOA);
    GPIO_DeInit(GPIOB);

    // PB4, key1, pull up input
    GPIO_Init(GPIOB, GPIO_PIN_4, GPIO_MODE_IN_FL_NO_IT);
    // PA1, led, output
    GPIO_Init(GPIOA, GPIO_PIN_1, GPIO_MODE_OUT_PP_HIGH_SLOW);

    // 开中断
    //enableInterrupts();

    /* Infinite loop */
    while (1)
    {
        key_scan_task();
        led_task();
    }
}


#ifdef USE_FULL_ASSERT
```

5, 编译调试，双击按键，led 亮；单击则 led 灭

注： 上面给出的消抖循环次数和等待二次按键的循环次数是根据实际调试调整的值，不具备通用性。