

1. 流水灯

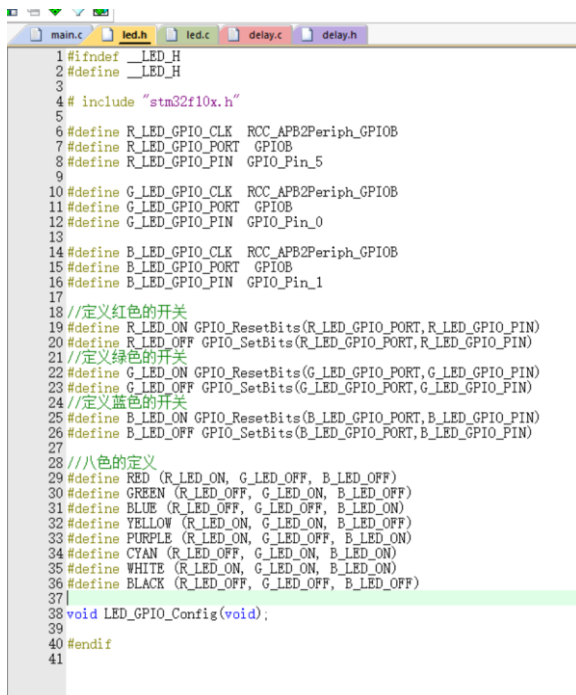
a) 首先自己创建了两个文件夹分别为 led, delay。(注意后面需要添加路径)

在 led 文件夹中: 创建 led.h 头文件和 led.c 源文件。

在 delay 文件夹中: 创建 delay.h 头文件和 delay.c 源文件。

b) 在 led 文件夹中, 编写 led.h 和 led.c 文件, 定义 LED 控制函数。

led.h 如图:

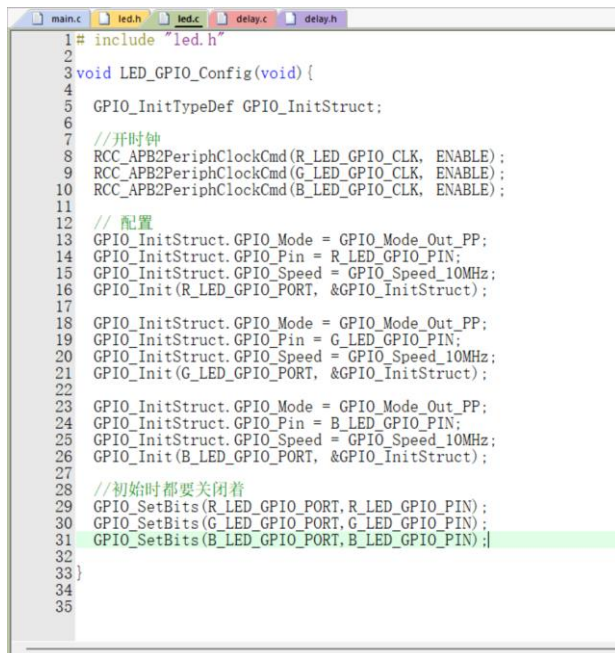


```

1 #ifndef _LED_H
2 #define _LED_H
3
4 #include "stm32f10x.h"
5
6 #define R_LED_GPIO_CLK RCC_APB2Periph_GPIOB
7 #define R_LED_GPIO_PORT GPIOB
8 #define R_LED_GPIO_PIN GPIO_Pin_5
9
10 #define G_LED_GPIO_CLK RCC_APB2Periph_GPIOB
11 #define G_LED_GPIO_PORT GPIOB
12 #define G_LED_GPIO_PIN GPIO_Pin_0
13
14 #define B_LED_GPIO_CLK RCC_APB2Periph_GPIOB
15 #define B_LED_GPIO_PORT GPIOB
16 #define B_LED_GPIO_PIN GPIO_Pin_1
17
18 //定义红色的开关
19 #define R_LED_ON GPIO_ResetBits(R_LED_GPIO_PORT, R_LED_GPIO_PIN)
20 #define R_LED_OFF GPIO_SetBits(R_LED_GPIO_PORT, R_LED_GPIO_PIN)
21 //定义绿色的开关
22 #define G_LED_ON GPIO_ResetBits(G_LED_GPIO_PORT, G_LED_GPIO_PIN)
23 #define G_LED_OFF GPIO_SetBits(G_LED_GPIO_PORT, G_LED_GPIO_PIN)
24 //定义蓝色的开关
25 #define B_LED_ON GPIO_ResetBits(B_LED_GPIO_PORT, B_LED_GPIO_PIN)
26 #define B_LED_OFF GPIO_SetBits(B_LED_GPIO_PORT, B_LED_GPIO_PIN)
27
28 //八色的定义
29 #define RED (R_LED_ON, G_LED_OFF, B_LED_OFF)
30 #define GREEN (R_LED_OFF, G_LED_ON, B_LED_OFF)
31 #define BLUE (R_LED_OFF, G_LED_OFF, B_LED_ON)
32 #define YELLOW (R_LED_ON, G_LED_ON, B_LED_OFF)
33 #define PURPLE (R_LED_ON, G_LED_OFF, B_LED_ON)
34 #define CYAN (R_LED_OFF, G_LED_ON, B_LED_ON)
35 #define WHITE (R_LED_ON, G_LED_ON, B_LED_ON)
36 #define BLACK (R_LED_OFF, G_LED_OFF, B_LED_OFF)
37
38 void LED_GPIO_Config(void);
39
40 #endif
41

```

led.c 如图:



```

1 #include "led.h"
2
3 void LED_GPIO_Config(void) {
4
5     GPIO_InitTypeDef GPIO_InitStructure;
6
7     //开时钟
8     RCC_APB2PeriphClockCmd(R_LED_GPIO_CLK, ENABLE);
9     RCC_APB2PeriphClockCmd(G_LED_GPIO_CLK, ENABLE);
10    RCC_APB2PeriphClockCmd(B_LED_GPIO_CLK, ENABLE);
11
12    //配置
13    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
14    GPIO_InitStructure.GPIO_Pin = R_LED_GPIO_PIN;
15    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
16    GPIO_Init(R_LED_GPIO_PORT, &GPIO_InitStructure);
17
18    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
19    GPIO_InitStructure.GPIO_Pin = G_LED_GPIO_PIN;
20    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
21    GPIO_Init(G_LED_GPIO_PORT, &GPIO_InitStructure);
22
23    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
24    GPIO_InitStructure.GPIO_Pin = B_LED_GPIO_PIN;
25    GPIO_InitStructure.GPIO_Speed = GPIO_Speed_10MHz;
26    GPIO_Init(B_LED_GPIO_PORT, &GPIO_InitStructure);
27
28    //初始时都要关闭着
29    GPIO_SetBits(R_LED_GPIO_PORT, R_LED_GPIO_PIN);
30    GPIO_SetBits(G_LED_GPIO_PORT, G_LED_GPIO_PIN);
31    GPIO_SetBits(B_LED_GPIO_PORT, B_LED_GPIO_PIN);
32
33 }
34
35

```

c) 在 delay 文件夹中, 编写 delay.h 和 delay.c 文件, 实现延迟功能。

delay.h 文件如下:

```
1 #ifndef __DELAY_H
2 #define __DELAY_H
3
4 #include "stm32f10x.h"
5
6 #ifdef __cplusplus
7 extern "C" {
8 #endif
9
10 void delay_init(uint32_t SystemCoreClock);
11 void delay_us(uint32_t nus);
12 void delay_ms(uint32_t nms);
13
14 #ifdef __cplusplus
15 }
16 #endif
17
18 #endif
19
```

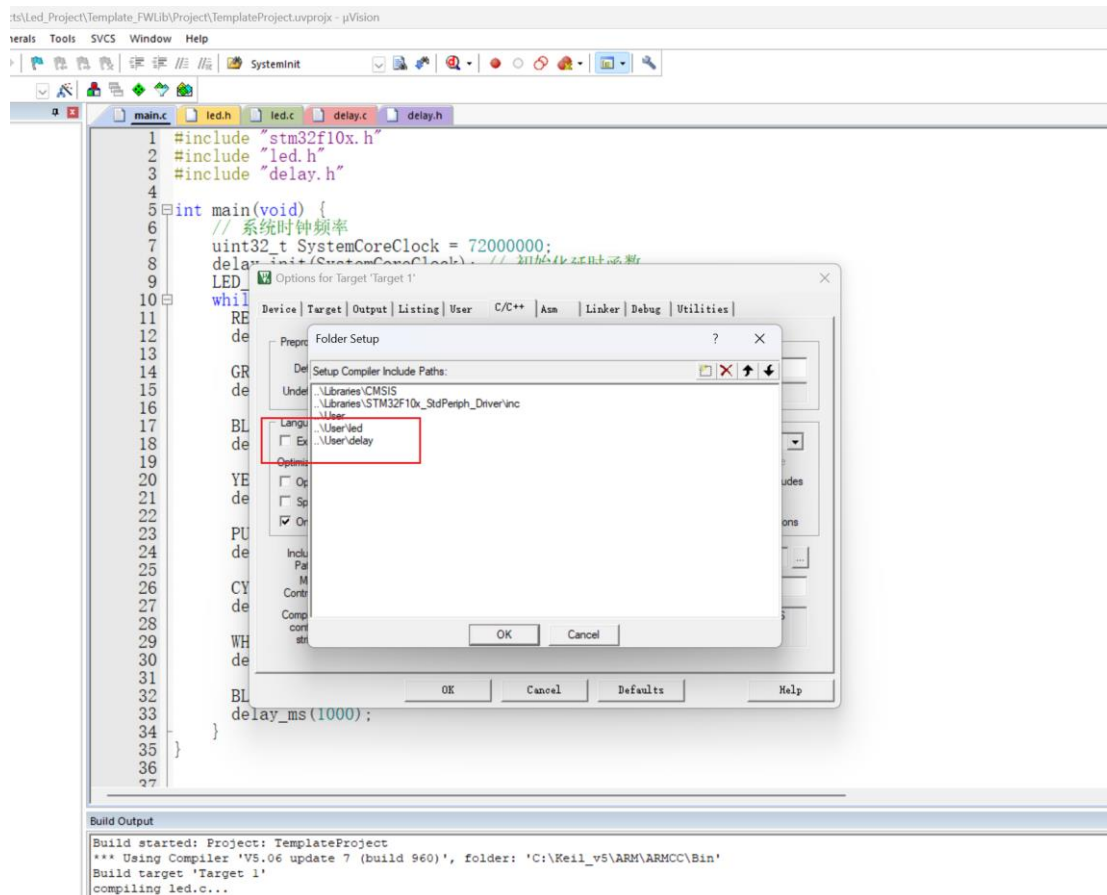
delay.c 文件如下:

```
1 #include "delay.h"
2
3 static uint32_t fac_us = 0; // us延时倍数
4 static uint32_t fac_ms = 0; // ms延时倍数
5
6 void delay_init(uint32_t SystemCoreClock) {
7     // 设置SysTick定时器的时钟源为HCLK/8
8     SysTick_CLKSourceConfig(SysTick_CLKSource_HCLK_Div8);
9     // 计算us和ms的延时倍数
10     fac_us = SystemCoreClock / 8000000; // 1us对应的计数
11     fac_ms = fac_us * 1000; // 1ms对应的计数
12 }
13
14 void delay_us(uint32_t nus) {
15     uint32_t temp;
16     SysTick->LOAD = nus * fac_us; // 设置定时器的加载值
17     SysTick->VAL = 0; // 清空计数器
18     SysTick->CTRL |= SysTick_CTRL_ENABLE_Msk; // 启动定时器
19     do {
20         temp = SysTick->CTRL;
21     } while ((temp & 0x01) && !(temp & (1 << 16))); // 等待定时器计数结束
22     SysTick->CTRL &= ~SysTick_CTRL_ENABLE_Msk; // 关闭定时器
23     SysTick->VAL = 0; // 清空计数器
24 }
25
26 void delay_ms(uint32_t nms) {
27     while (nms--) {
28         delay_us(1000);
29     }
30 }
```

d) 编写主程序 main.c

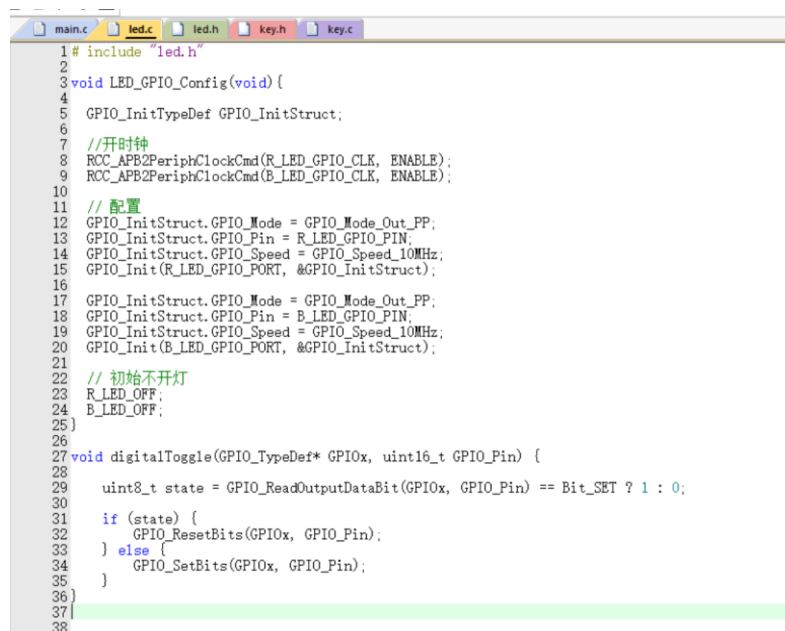
```
1 #include "stm32f10x.h"
2 #include "led.h"
3 #include "delay.h"
4
5 int main(void) {
6     // 系统时钟频率
7     uint32_t SystemCoreClock = 72000000;
8     delay_init(SystemCoreClock); // 初始化延时函数
9     LED_GPIO_Config();
10     while (1) {
11         RED;
12         delay_ms(1000);
13
14         GREEN;
15         delay_ms(1000);
16
17         BLUE;
18         delay_ms(1000);
19
20         YELLOW;
21         delay_ms(1000);
22
23         PURPLE;
24         delay_ms(1000);
25
26         CYAN;
27         delay_ms(1000);
28
29         WHITE;
30         delay_ms(1000);
31
32         BLACK;
33         delay_ms(1000);
34     }
35 }
```

e) 配置项目路径

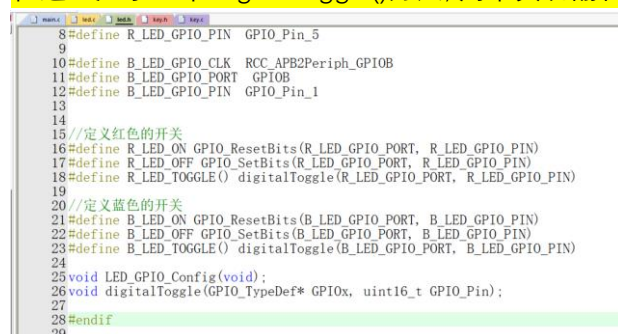


2. 按键检测

a) 对 led.c 和 led.h 的补充



在这里加了一个 digitalToggle()方法, 用来实现翻转指定 GPIO 引脚的电平状态。



b) 创建一个 key 文件夹, 添加 key.h 和 key.c 两个文件

c) 编写 key.h 文件

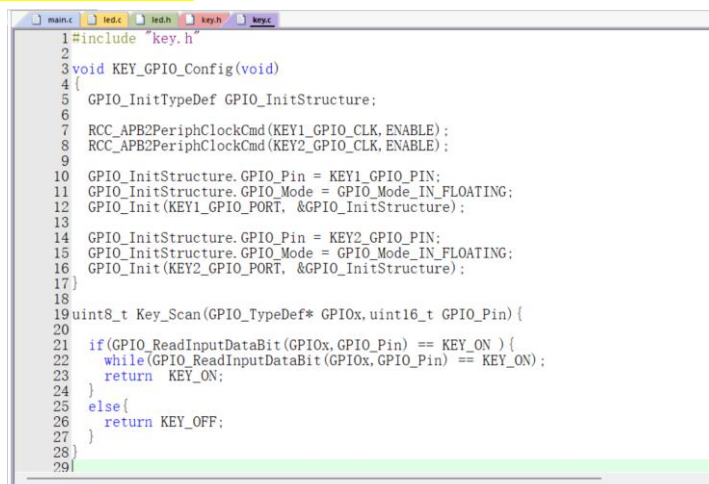
```

1 #ifndef __KEY_H
2 #define __KEY_H
3
4 #include "stm32f10x.h"
5
6 #define KEY1_GPIO_CLK    RCC_APB2Periph_GPIOA
7 #define KEY1_GPIO_PORT   GPIOA
8 #define KEY1_GPIO_PIN    GPIO_Pin_0
9
10 #define KEY2_GPIO_CLK    RCC_APB2Periph_GPIOC
11 #define KEY2_GPIO_PORT   GPIOC
12 #define KEY2_GPIO_PIN    GPIO_Pin_13
13
14 #define KEY_ON  1
15 #define KEY_OFF 0
16
17 void KEY_GPIO_Config(void);
18 uint8_t Key_Scan(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin);
19 #endif
20

```

- 包含必要的头文件：引入 stm32f10x.h
- 定义按键硬件接口：通过宏定义来指定每个按键连接的 GPIO 端口、引脚以及时钟配置，这样便于管理和修改硬件接口。
- 定义按键状态：使用宏 KEY_ON 和 KEY_OFF 来表示按键的两种状态，便于在代码中直观地识别按键是否被按下。
- 声明按键初始化函数：提供一个 KEY_GPIO_Config 函数声明，用于初始化按键对应的 GPIO 端口，设置为输入模式。
- 声明按键扫描函数：提供一个 Key_Scan 函数声明，用于检测按键状态。该函数接收 GPIO 端口和引脚作为参数，返回按键的当前状态。

d) 编写 key.c 文件



```

1 #include "key.h"
2
3 void KEY_GPIO_Config(void)
4 {
5     GPIO_InitTypeDef GPIO_InitStructure;
6
7     RCC_APB2PeriphClockCmd(KEY1_GPIO_CLK, ENABLE);
8     RCC_APB2PeriphClockCmd(KEY2_GPIO_CLK, ENABLE);
9
10    GPIO_InitStructure.GPIO_Pin = KEY1_GPIO_PIN;
11    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
12    GPIO_Init(KEY1_GPIO_PORT, &GPIO_InitStructure);
13
14    GPIO_InitStructure.GPIO_Pin = KEY2_GPIO_PIN;
15    GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IN_FLOATING;
16    GPIO_Init(KEY2_GPIO_PORT, &GPIO_InitStructure);
17}
18
19 uint8_t Key_Scan(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin) {
20
21     if (GPIO_ReadInputDataBit(GPIOx, GPIO_Pin) == KEY_ON) {
22         while (GPIO_ReadInputDataBit(GPIOx, GPIO_Pin) == KEY_ON);
23         return KEY_ON;
24     }
25     else {
26         return KEY_OFF;
27     }
28 }
29

```

- 包含按键头文件：#include "key.h" 引入了按键相关的宏定义和函数声明。
- 定义按键 GPIO 配置函数：void KEY_GPIO_Config(void) 用于初始化按键 GPIO 端口。
- 初始化 GPIO 结构体：定义了一个 GPIO_InitTypeDef 类型的变量 GPIO_InitStructure，用于配置 GPIO 端口的参数。
- 使能 GPIO 时钟：使用 RCC_APB2PeriphClockCmd 函数使能两个按键所在 GPIO 端口的时钟（KEY1 和 KEY2）。
- 配置 KEY1 GPIO 端口和 KEY2 GPIO 端口：
 - 设置 GPIO_InitStructure 的 GPIO_Pin 为 KEY1_GPIO_PIN，即按键 1 连接的引脚。
 - 设置 GPIO_InitStructure 的 GPIO_Mode 为 GPIO_Mode_IN_FLOATING，即输入浮空模式。
 - 调用 GPIO_Init 函数初始化 KEY1 和 KEY2 的 GPIO 端口。
- 定义按键扫描函数：uint8_t Key_Scan(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin) 用于检测指定 GPIO 端口和引脚的按键状态。
- 检测按键状态：
 - 使用 GPIO_ReadInputDataBit 函数读取按键的当前状态。
 - 如果按键被按下（假设低电平为按下），则进入一个等待循环，直到按键被释放（高电平）。
 - 返回 KEY_ON 表示按键被按下并释放。
- 处理未按下状态：如果按键未被按下，直接返回 KEY_OFF。

e) 编写主程序 main.c

```

1 #include "stm32f10x.h"
2 #include "led.h"
3 #include "key.h"
4
5 int main(void)
6 {
7     LED_GPIO_Config();
8     R_LED_ON;
9     KEY_GPIO_Config();
10
11     while(1)
12     {
13         if( Key_Scan(KEY1_GPIO_PORT, KEY1_GPIO_PIN) == KEY_ON) {
14             R_LED_TOGGLE();
15         }
16         if( Key_Scan(KEY2_GPIO_PORT, KEY2_GPIO_PIN) == KEY_ON) {
17             B_LED_TOGGLE();
18         }
19     }
20 }
21
22
23

```

f) 添加路径

