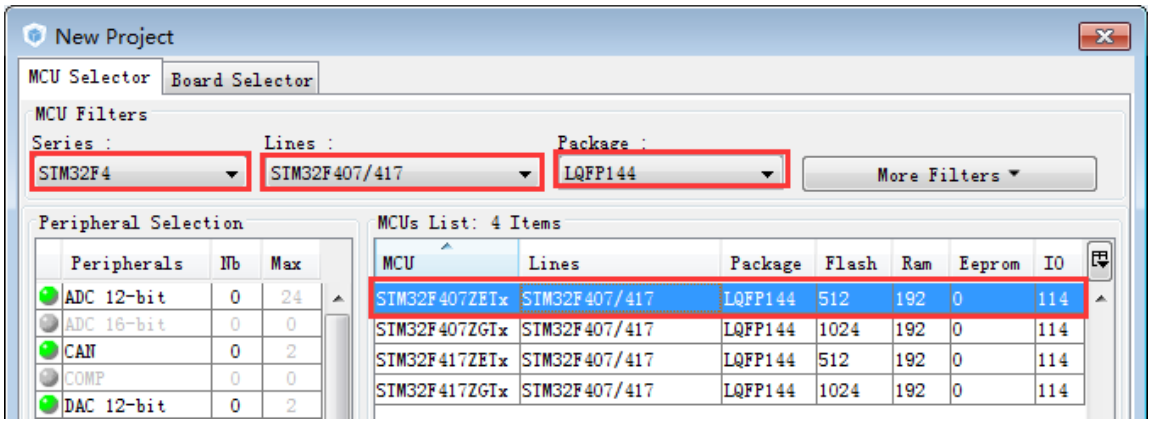
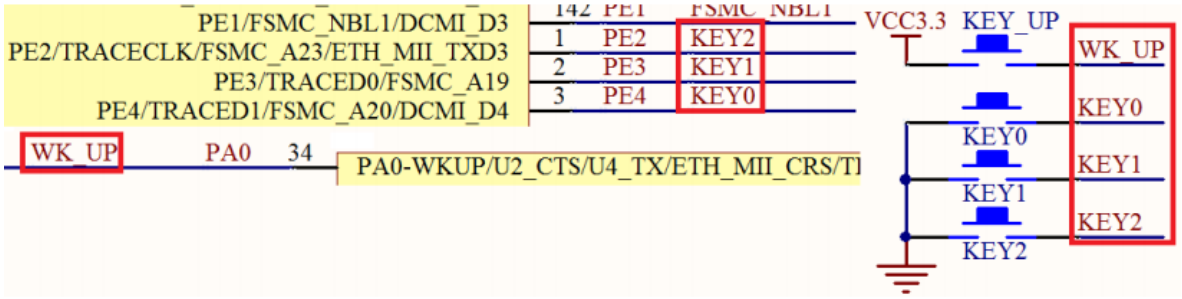
STM32Cube 学习之三：按键输入

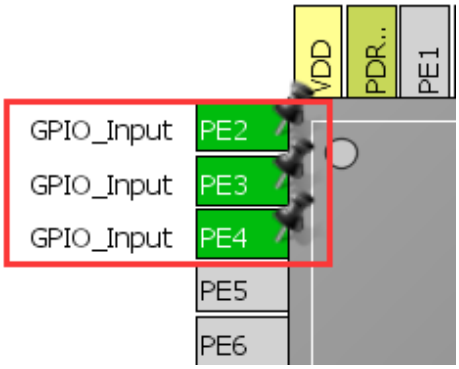
假设已经安装好 STM32CubeMX 和 STM32CubeF4 支持包。

Step1.打开 STM32CubeMX，点击“New Project”，选择芯片型号，STM32F407ZETx。

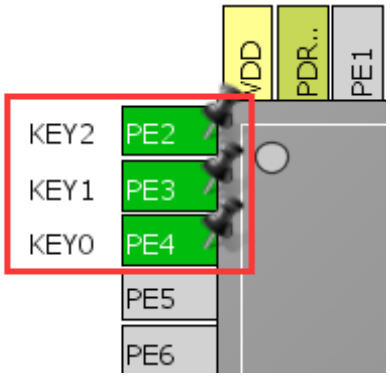


Step2.在 Pinout 界面下配置按键输入引脚。

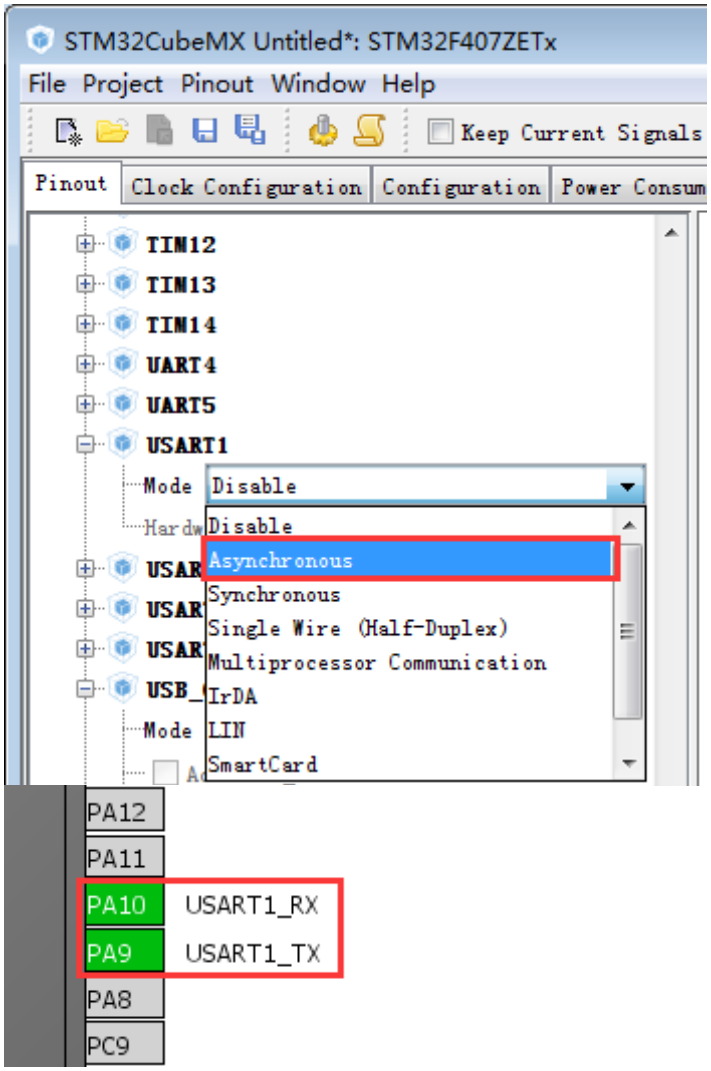




右键添加用户标签



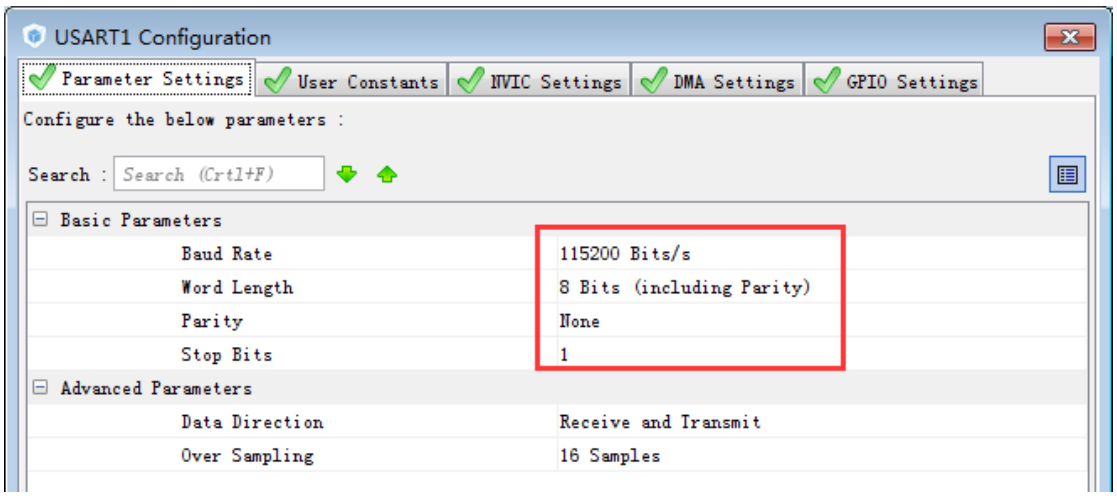
Step3.配置 USART，用于按键信息输出。



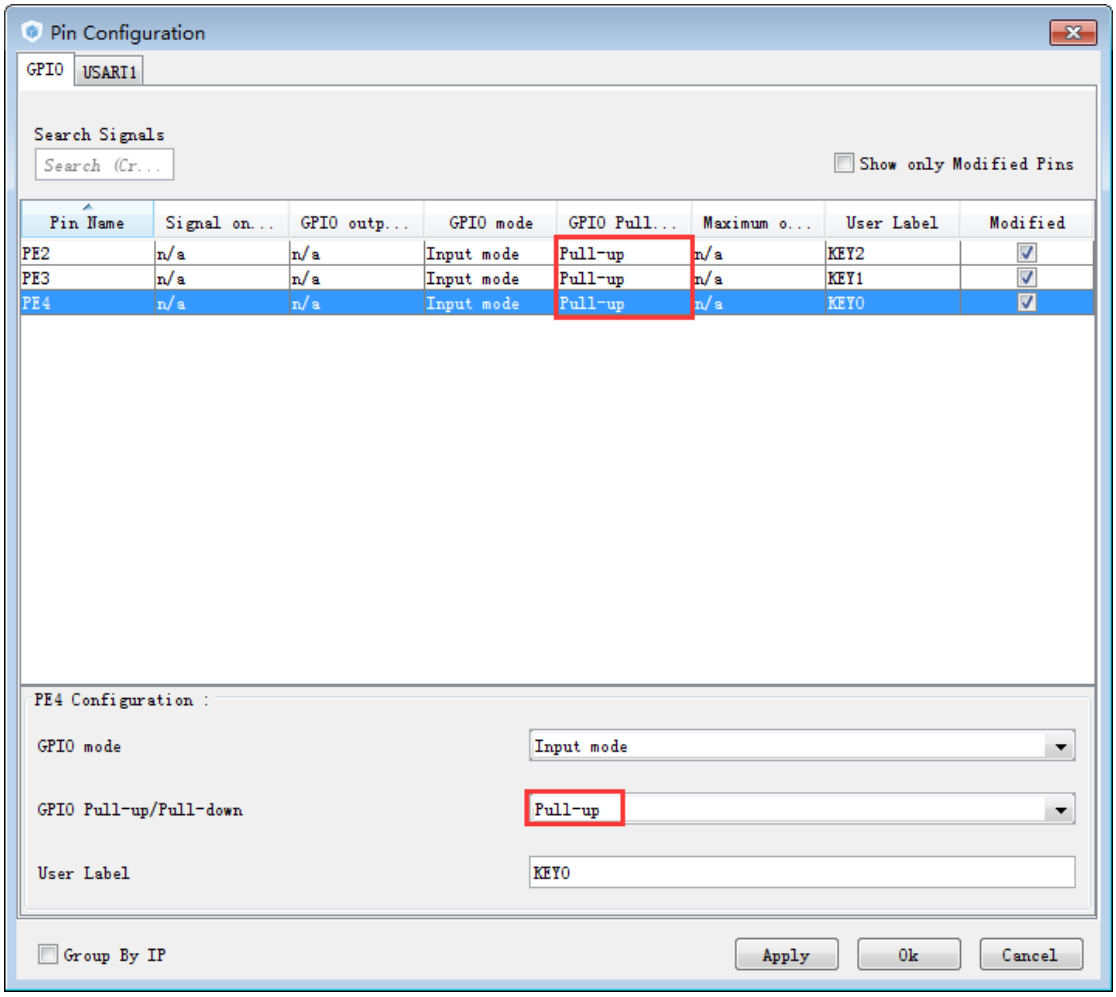
时钟树的配置不作任何修改，使用内部 16M 时钟源，内核时钟 16M。

Step4.配置串口参数和 GPIO 的参数。

在 configuration 界面中点击 USART1 按钮，可以进入参数配置界面，使用默认参数即可。

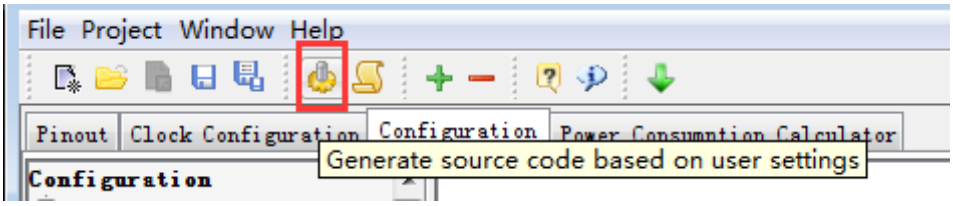


根据硬件情况，将按键输入的三个脚都设置上拉。

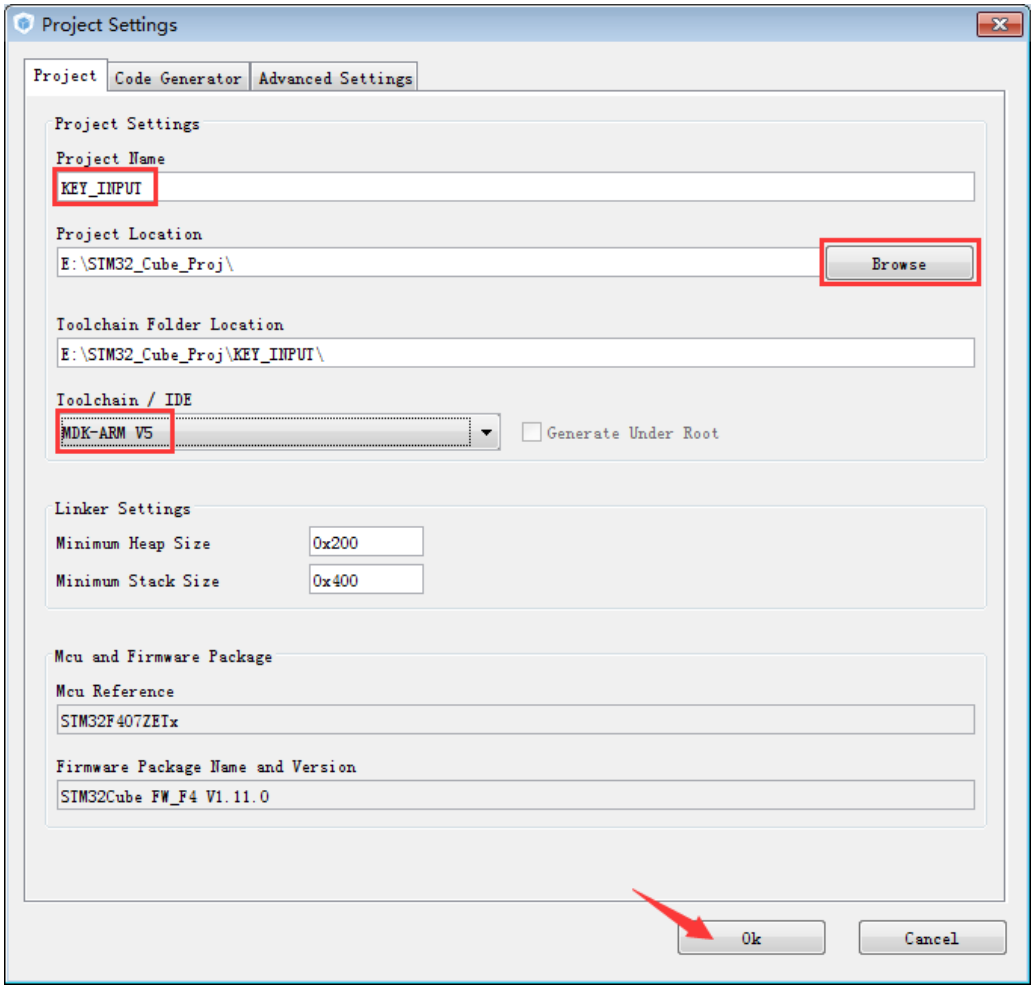


Step5.生成源代码。

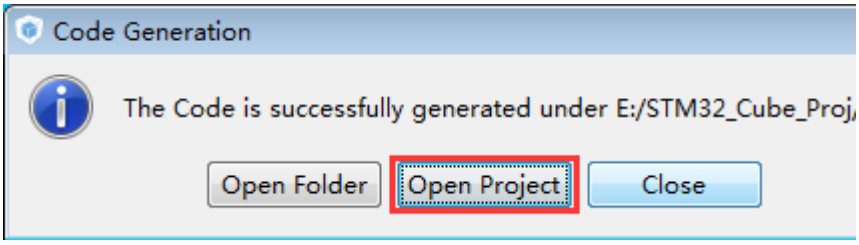
点击生成源代码按钮。



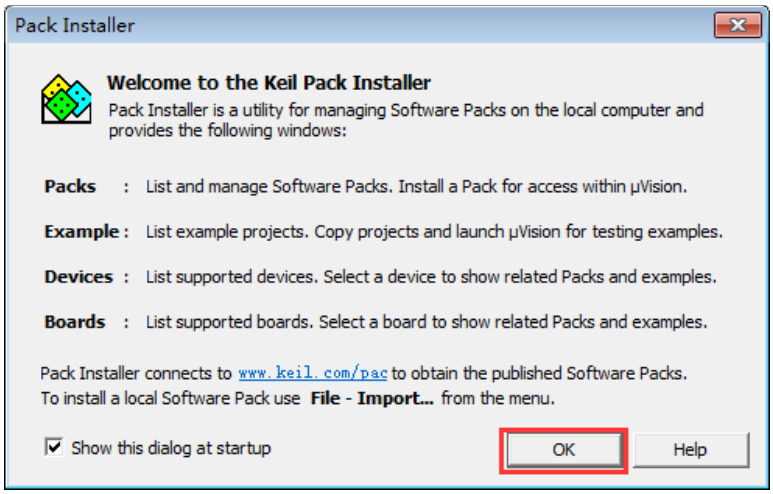
在设置界面中输入工程名，保存路径，工程 IDE 类型，点 OK 即可。



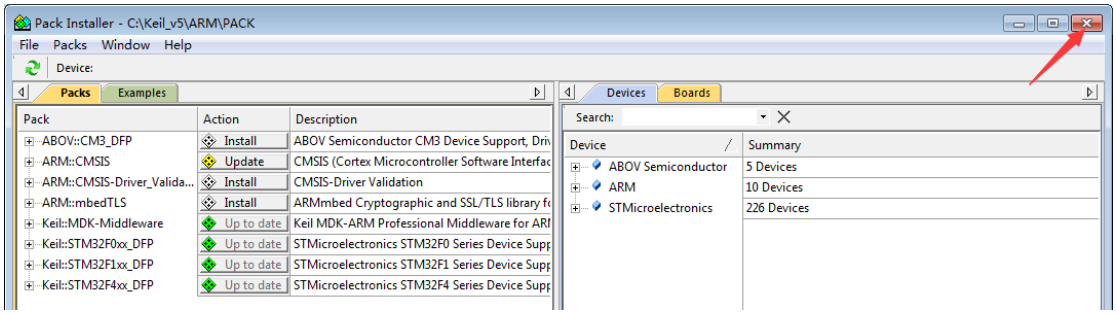
生成代码完成后可直接打开工程。



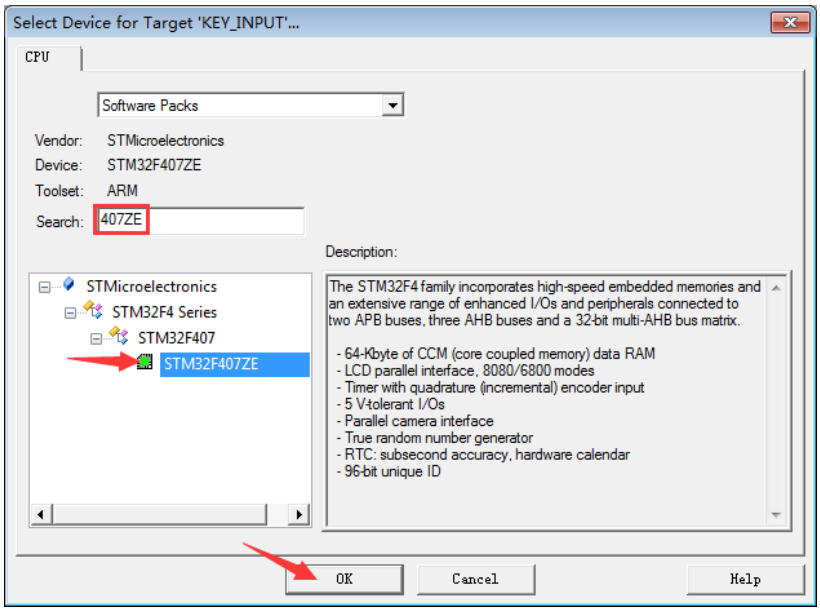
弹出如下对话框时，如果已经安装了 F4 的支持包，则点击 OK 关闭。如果没有安装，则点击界面中的 www.keil.com/...链接，找到芯片的支持包，然后安装。



关闭后面的界面。

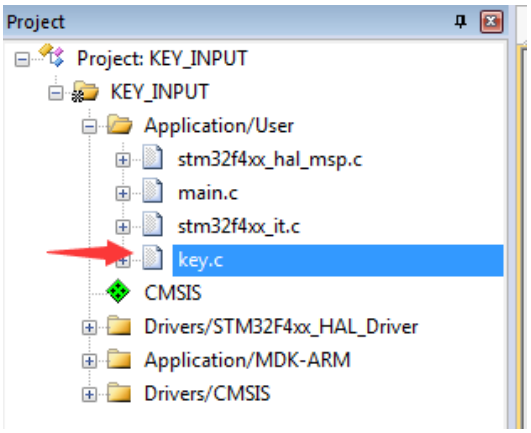


点击“是”，然后选择芯片型号。可以在搜索框中输入关键字，加快选择速度。



Step6.添加功能代码。

将自己的按键扫描驱动 key.c 和 key.h 文件添加到工程中，文件内容见附录。



在 main.c 文件中包含按键头文件

```
33  /* Includes ------------------------------------------------------
34  #include "stm32f4xx_hal.h"
35
36  /* USER CODE BEGIN Includes */
37  #include "key.h"
38  /* USER CODE END Includes */
39
```

在 main 函数中定义一个变量，用于保存按键值。

```
62  int main(void)
63  {
64
65    /* USER CODE BEGIN 1 */
66     uint8_t key_val;
67    /* USER CODE END 1 */
68
```

在 while(1)中添加按键处理的代码。

```
85    /* Infinite loop */
86    /* USER CODE BEGIN WHILE */
87    while (1)
88    {
89    /* USER CODE END WHILE */
90
91    /* USER CODE BEGIN 3 */
92      if(key_scan_flag) {
93          key_val = key_scan();
94          switch(key_val) {
95              case KEY_0_PRESS:
96                  HAL_UART_Transmit(&huart1, "KEY_0\r\n", 7, 10);
97                  break;
98              case KEY_1_PRESS:
99                  HAL_UART_Transmit(&huart1, "KEY_1\r\n", 7, 10);
100                 break;
101             case KEY_2_PRESS:
102                 HAL_UART_Transmit(&huart1, "KEY_2\r\n", 7, 10);
103                 break;
104             default:
105                 break;
106         }
107     }
108   }
109   /* USER CODE END 3 */
```

代码功能就是，在扫描到有按键有效值之后，向串口发现相应的信息。

在 main.c 的 USER CODE BEGIN 4 和 USER CODE END 4 注释之间，

重写 HAL_SYSTICK_Callback()函数。

```c
/* USER CODE BEGIN 4 */
/*============================================
函数名称: HAL_SYSTICK_Callback
功    能: 系统滴答时钟回调函数,每1ms执行一次.
输    入: 无
返    回: 无
============================================*/
void HAL_SYSTICK_Callback(void)
{
    static uint8_t tick_cnt = 0;

    tick_cnt++;
    if (10 == tick_cnt) {
        tick_cnt = 0;
        key_scan_flag = 1;
    }
}
/* USER CODE END 4 */
```

函数解析

HAL_SYSTICK_Callback()函数是 STM32CubeMX 生成的代码框架中的一个回调函数，该函数由系统 Tick 定时器中断调用，周期是 1ms。

在本例中，用该函数来将按键扫描标志变量 key_scan_flag 置 1。从代码可知，每 10ms 将 key_scan_flag 置 1 一次。在主函数的 while(1)循环中，根据该变量决定是否调用一次按键扫描函数 key_scan()。key_scan_flag 是在 key.c 中定义的一个全局变量。在 key_scan()函数在，会将 key_scan_flag 清零。

本例中的 key_scan()函数扫描方式，巧妙地利用定时器实现了按键的消抖，且占用时间很少。

附录：

key.h 文件内容

```c
#ifndef __KEY_H
#define __KEY_H

#include "stm32f4xx_hal.h"

//键值列表
#define  KEY_NO_ONE      0   // 无按键
#define  KEY_0_PRESS     1   // K0键
#define  KEY_1_PRESS     2   // K1键
#define  KEY_2_PRESS     3   // K2键

extern volatile uint8_t key_scan_flag;

extern uint8_t key_scan(void);

#endif
```

key.c 文件内容

```c
#include "key.h"

// 每10ms扫描一次按键，以下控制量单位为10ms
#define   KEY_DELAY  5                // 延时

volatile uint8_t key_scan_flag = 0;

static uint8_t cn = 0;
static uint16_t  new_value = 0;
static uint16_t  old_value = 0;

// 按键的引脚输入
uint16_t get_key_input(void)
{
    //PE4~PE2 读取IO输入【根据实际电路修改】
    return (GPIOE->IDR & (0x001C));
}

//按键扫描
uint8_t key_scan(void)
{
    uint16_t val_read;

    key_scan_flag = 0;
    val_read = get_key_input();

    if (0x001C != val_read) {   // 如果有按键按下
        if(0 == cn){    // 第一次判断
            old_value = val_read;
        }
        if (cn <= KEY_DELAY) {
            cn++;   //按键按下计时变量
        }
        if(KEY_DELAY == cn) {
            new_value = val_read;
            if(new_value == old_value) {
                switch(new_value){
                    case 0x000C: return KEY_0_PRESS;     //按K0
                    case 0x0014: return KEY_1_PRESS;     //按K1
                    case 0x0018: return KEY_2_PRESS;     //按K2
                    default :return KEY_NO_ONE;          //无任何按键按下
                }
            }
        }
    } else {
        cn=0;       //按键按下计时变量
    }
    return KEY_NO_ONE;
}
```

S.D.Lu 于 深圳
2016 年 8 月