

Lab2 按键控制的LED

一 实验环境

二 软件实现

一 实验环境

下载 STM32F10x_StdPeriph_Lib_V3.6.0

手动搭建 STM32 工程项目

WIN10 19045

Keil MDK v5.3.9.0.0

ARM Compiler v6.21

VSCode

二 软件实现

在作业 HM1 的基础上继续完善，实验按键 KEY0 来进行 LED 的闪烁控制。

会用到 `Bit_RESET` 和之前用到的一个函数

```
80  if (GPIO_ReadInputDataBit(GPIOx: GPIOC, GPIO_Pin: GPIO_Pin_5) == Bit_RESET)

stm32f10x_gpio.c C:\Users\22H2\Documents\GitHub\embed-lab-homework\HM2\STM32F10x FWLib\src - 定义 (7)
265  void GPIO_StructInit(GPIO_InitTypeDef* GPIO_InitStruct)
271  }
272
273  /**
274   * @brief Reads the specified input port pin.
275   * @param GPIOx: where x can be (A..G) to select the GPIO peripheral
276   * @param GPIO_Pin: specifies the port bit to read.
277   * This parameter can be GPIO_Pin_x where x can be (0..15).
278   * @retval The input port pin value.
279   */
280  uint8_t GPIO_ReadInputDataBit(GPIO_TypeDef* GPIOx, uint16_t GPIO_Pin)
281  {
282      uint8_t bitstatus = 0x00;
```

```
75 // 进入具有嵌入式特色的 while 无限循环
76 while (1)
77 {
78 // Check the state of KEY0
79 // Bit_RESET 作为一个宏定义，通常表示低电平 (0) ✓
80 if (GPIO_ReadInputDataBit(GPIOx: GPIOC, GPIO_Pin: GPIO_Pin_5) == Bit_RESET)

stm32f10x_gpio.h C:\Users\22H2\Documents\GitHub\embed-lab-homework\HM1\STM32F10x_FWLib\inc - 定义 (2)
101
102
103 /**
104 * @brief Bit_SET and Bit_RESET enumeration
105 */
106
You, 上周 | 1 author (You)
107 typedef enum
108 { Bit_RESET = 0,
109   Bit_SET
110 }BitAction;
111
112 #define IS_GPIO_BIT_ACTION(ACTION) (((ACTION) == Bit_RESET) || ((ACTION) == Bit_SET))
113
114 /**
115 * @}
81 { // If KEY0 is pressed
82   LED0_ON();
```

KEY0 的按键函数逻辑主要用到 `GPIO_Mode` 的上拉输入

```
KEY0 Function Imp
1 // 根据要求 KEY0 读取PC5
2 void KEY0_Init(void)
3 {
4   GPIO_InitTypeDef GPIO_InitStructure;
5   RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
6   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5; // KEY0对应的引脚PC5
7   GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU; // 输入上拉（上拉电阻）
8   GPIO_Init(GPIOC, &GPIO_InitStructure);
9 }
```

模拟结果如下：

Debug Peripherals Tools SVCS Window Help

Logic Analyzer

GPIOC

Property

- CRL
- CRH
- IDR
- ODR
- BSRR
- BRR
- LCKR

General Purpose I/O C (GPIOC)

Pin	CNF
PC.0	Floating Input
PC.1	Floating Input
PC.2	Floating Input
PC.3	Floating Input
PC.4	Floating Input
PC.5	Floating Input
PC.6	Floating Input
PC.7	Floating Input

Selected Port Pin Configuration

MODE: 0: Input CNF: 1: Floating Input

Configuration & Mode Settings

GPIOC_CRH: 0x44444444 GPIOC_CRL: 0x44444444

GPIOC_IDR: 0x00000020

GPIOC_ODR: 0x00000020

GPIOC_LCKR: 0x00000000

Pins: 0x00000020

Call Stack + Locals

Name	Location/Value	Type
main	0x080005F4	int f()

Simulation

t1: 0.00004249 sec L:288 C:1

0/8 22:22 C 源文件 36 KB

1/29 11:17 文本文档 1 KB

1/29 11:17 ISON 源文件 1 KB

The screenshot displays a logic analyzer interface with a digital signal trace at the top. Below the trace, a 'General Purpose I/O C (GPIOC)' configuration dialog is open. The dialog lists pins PC.0 through PC.7, all configured as 'Floating Input'. The 'Selected Port Pin Configuration' section shows 'MODE: 0: Input' and 'CNF: 1: Floating Input'. The 'Configuration & Mode Settings' section includes fields for 'GPIOC_CRH' (0x44444444) and 'GPIOC_CRL' (0x44444444). The 'GPIOC' section shows 'GPIOC_IDR' (0x00000000), 'GPIOC_ODR' (0x00000020), 'GPIOC_LCKR' (0x00000000), and 'Pins' (0x00000000). The 'Settings' section shows 'Clock Enabled'. To the right of the dialog, a table shows the 'Location/Value' as '0x080005F4' and the 'Type' as 'int f()'. The signal trace shows a square wave pulse.

勾选 PC5 模拟按键按下/释放

源代码如下：

```

1  /**
2  ****
3  * @file    Project/STM32F10x_StdPeriph_Template/main.c
4  * @author  MCD Application Team
5  * @version V3.6.0
6  * @date    20-September-2021
7  * @brief   Main program body
8  ****
9  * @attention
10 *
11 * Copyright (c) 2011 STMicroelectronics.
12 * All rights reserved.
13 *
14 * This software is licensed under terms that can be found in the LICENSE
file
15 * in the root directory of this software component.
16 * If no LICENSE file comes with this software, it is provided AS-IS.
17 *
18 ****
19 */
20
21 /* Includes -----
----*/
22 #include "stm32f10x.h"
23 #include "stm32f10x_gpio.h"
24 #include "stm32f10x_rcc.h"
25
26 /* 使用STM32F10x的 按键KEY0 来控制LED的闪烁，在HM1的基础上继续完成功能-----
-----*/
27 void LED0_Init(void)
28 {
29     // 定义一个GPIO_InitTypeDef类型的结构体 C:\Users\22H2\Documents\Boards\Keil
5\HM1\STM32F10x_FWLib\inc\stm32f10x_gpio.h
30     GPIO_InitTypeDef GPIO_InitStructure;
31     // 使能GPIOA的时钟，打开APB2总线时钟 C:\Users\22H2\Documents\Boards\Keil5\HM
1\STM32F10x_FWLib\inc\stm32f10x_rcc.h
32     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA, ENABLE);
33     // 配置GPIOB的Pin8为推挽输出
34     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;           // Specifies the GPIO p
ins to be configured.
35     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP; // 推挽输出
36

```

```

37     GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz; // Specifies the speed
38     for the selected pins.
39     GPIO_Init(GPIOA, &GPIO_InitStructure);
40 }
41
42 void LED0_ON(void)
43 {
44     GPIO_ResetBits(GPIOA, GPIO_Pin_8);
45 }
46
47 void LED0_OFF(void)
48 {
49     GPIO_SetBits(GPIOA, GPIO_Pin_8); // GPIOx: A,B,C,D,E,F,G; GPIO_Pin: Specifies the port bit to be written.
50 }
51
52 // __IO 作为一个宏（修饰），来告诉编译器后面这个变量是一个输入/输出变量
53 void Delay(__IO u32 nCount)
54 {
55     unsigned long i;
56     for (i = 0; i < nCount; i++)
57         ; // 空循环（消耗时间）
58 }
59
60 // 根据要求 KEY0 读取PC5
61 void KEY0_Init(void)
62 {
63     GPIO_InitTypeDef GPIO_InitStructure;
64     RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOC, ENABLE);
65     GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5; // KEY0对应的引脚PC5
66     GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU; // 输入上拉（上拉电阻）
67     GPIO_Init(GPIOC, &GPIO_InitStructure);
68 }
69
70 int main(void)
71 {
72     // 初始化LED0
73     // 初始化按键KEY0
74     LED0_Init();
75     KEY0_Init();
76
77     // 进入具有嵌入式特色的 while 无限循环
78     while (1)
79     {
80         // Check the state of KEY0
81         // Bit_RESET 作为一个宏定义，通常表示低电平（0）
82         if (GPIO_ReadInputDataBit(GPIOC, GPIO_Pin_5) == Bit_RESET)
83             { // If KEY0 is pressed

```

```
83         LED0_ON();
84     }
85     else
86     { // If KEY0 is released
87         LED0_OFF();
88     }
89 }
90 }
```