

STM32G431

基础知识与*keil & cubemx*配置



苏州大学

苗业

基础知识

1.软件开发包SDK:

- (1) 固件库
- (2) 标准库
- (3) 固件包: *HAL*库, *LL*库

2.CubeMX与keil配置:

SYS: 1 Timebase Source: SysTick

2 Debug: disabled

RCC: (HSE) Crystal/Ceramic Resonator

Project Manager-> Code Generator-> Application Structure: Basic

Project Manager-> Code Generator-> Toolchain/IDE: MDK - ARM V5

*MDK - ARM*文件夹中添加启动文件: *statup_stm32g43xx.s*
(并添加进工程文件)

魔术棒:

- 1. *Reset and run*
- 2. *Cmsis - dap link*

3.Stm32命名规则

STM32G431RBT6

32: 32 位 CPU->4G

G: 一种系列

431: 特定功能

R: 64 引脚

B: 闪存大小

T: 封装形式

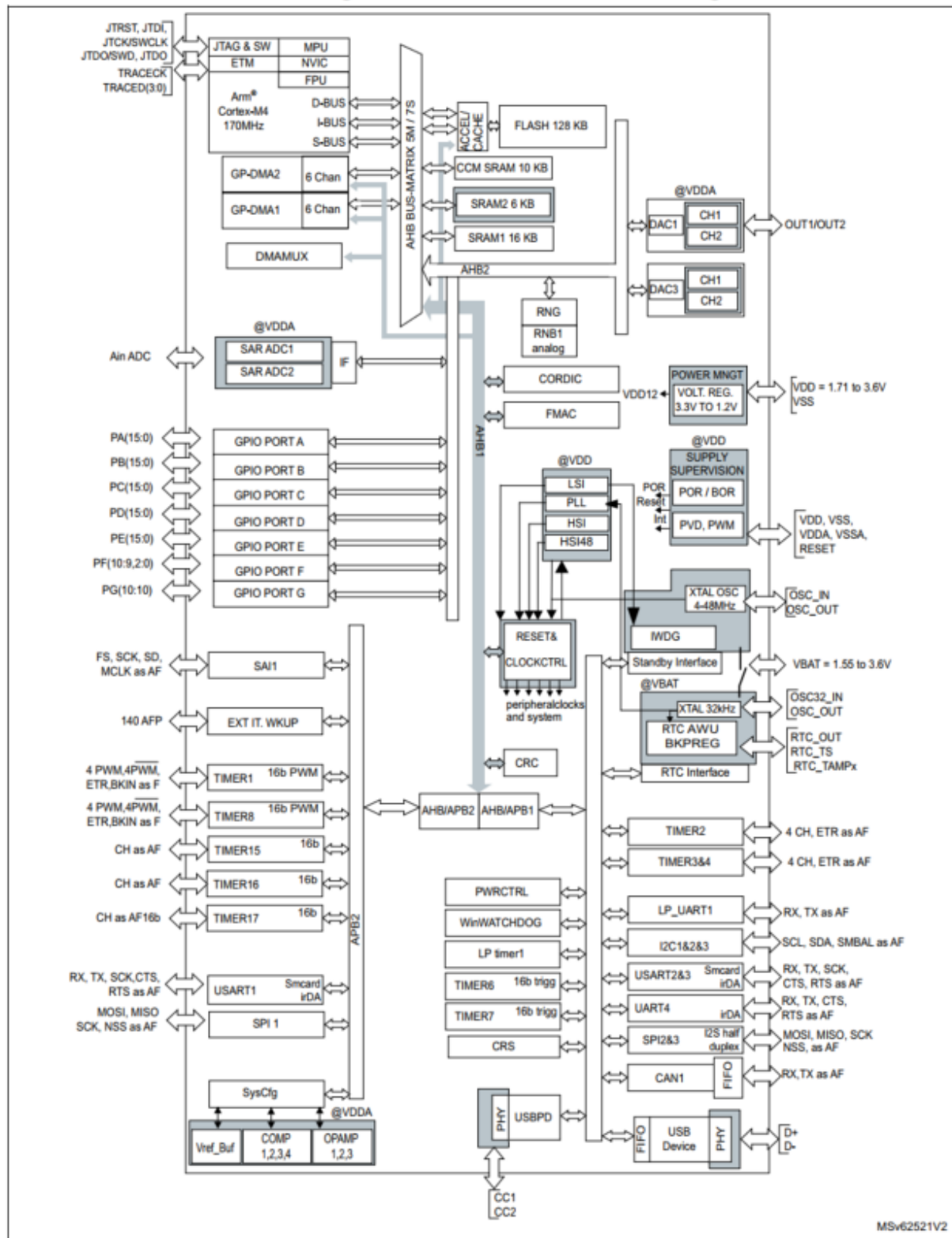
6: 适用温度范围

4.相关检索

STM32G431RBT6内部资源：芯片资料->系列微控制器参考手册

STM32G431RBT6芯片内部结构图：芯片资料->系列微控制器数据手册

Figure 1. STM32G431x6/x8/xB block diagram

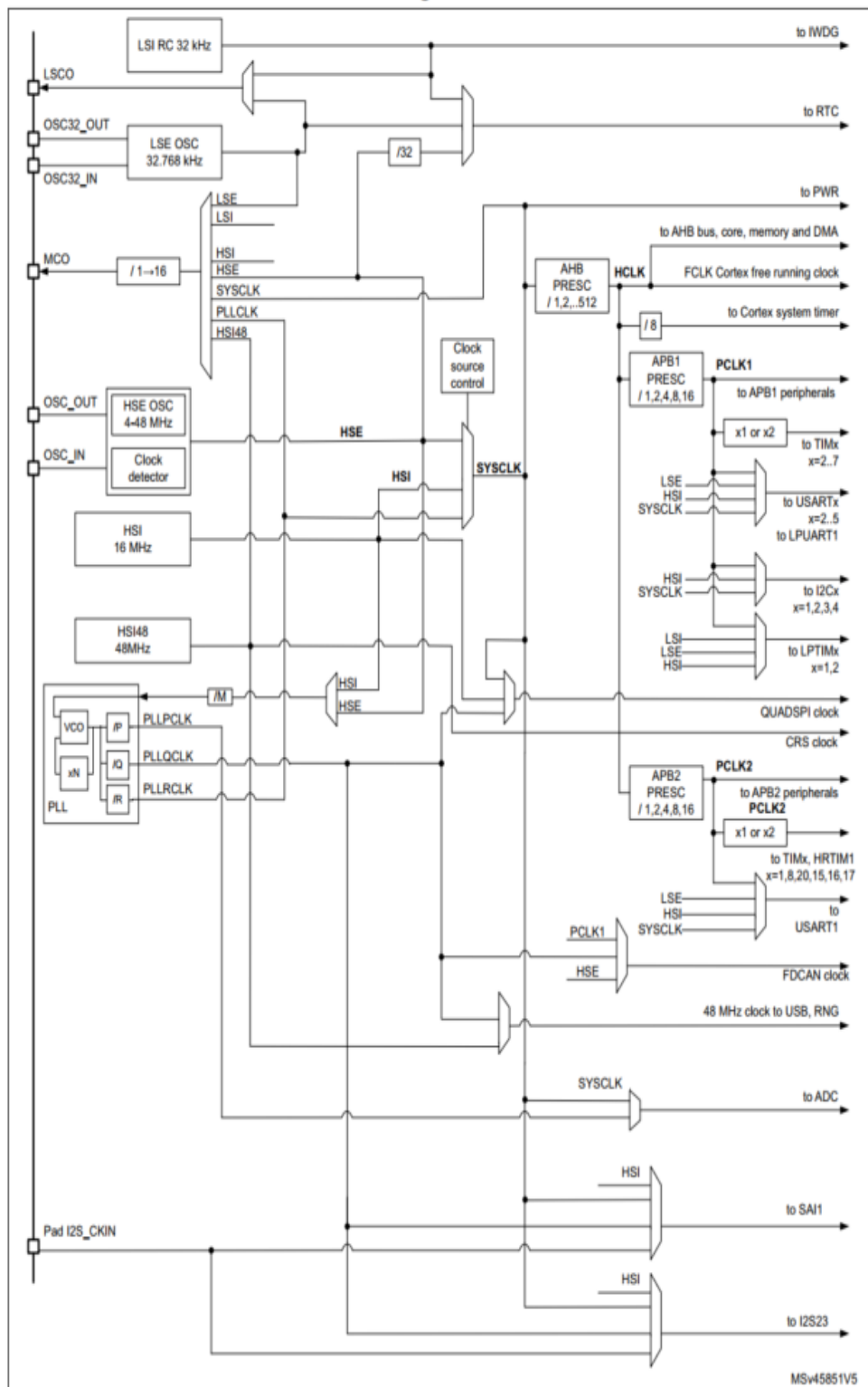


MSv62521V2

Note: AF: alternate function on I/O pins.

5.时钟系统配置

Figure 17. Clock tree

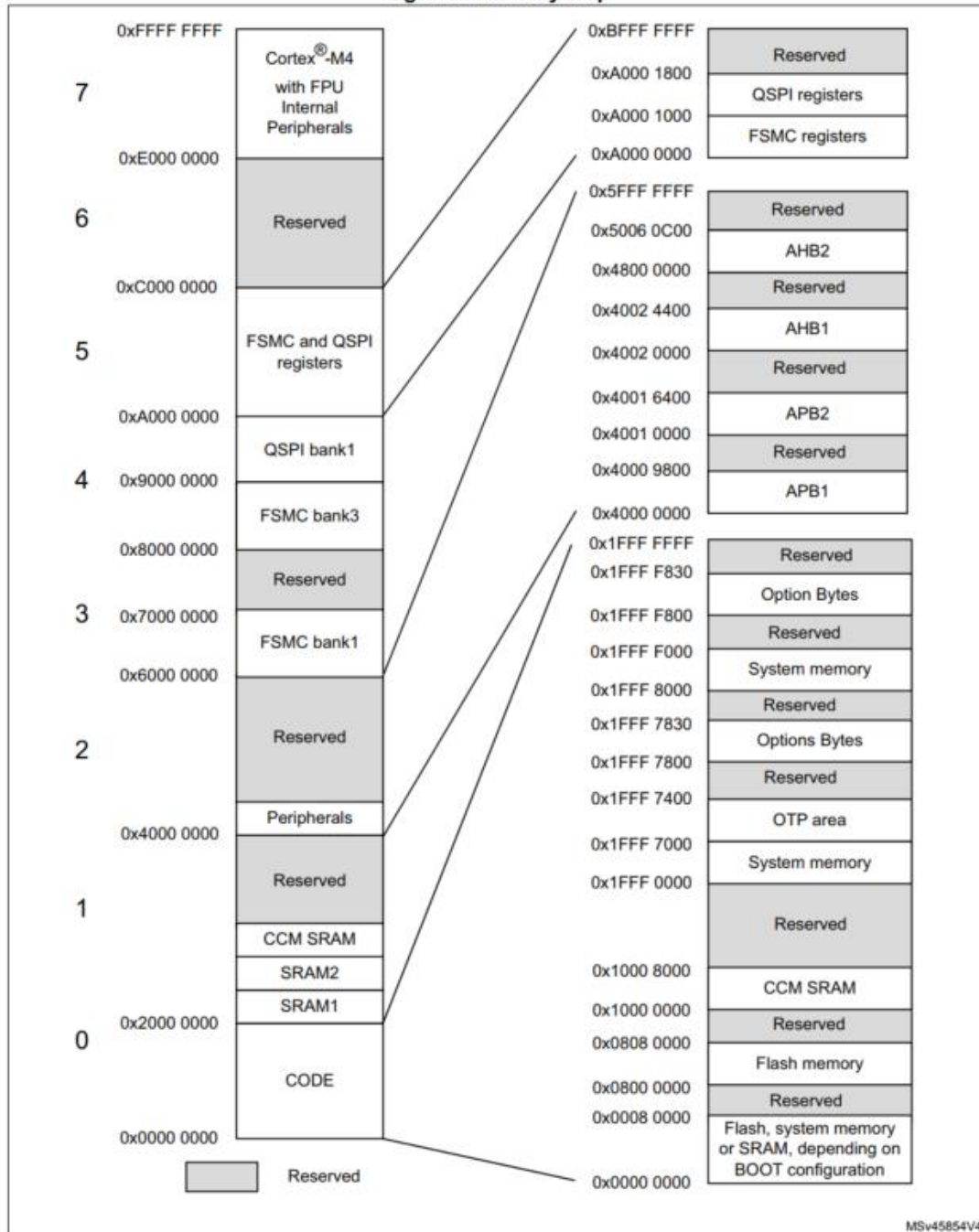


CUBEMX配置:

HSE→ Clock configure→ 24; HSE;/3;* 20;/2→ PLLCLK

6. STM32内部资源地址

Figure 2. Memory map



MSv45854V4

外设 0x40

闪存 0x08

RAM 0x20

7.代码运行顺序

(由下至上)



8. 程序工程代码结构

Drivers: 驱动文件(HAL 库)

Inc: 头文件

Src: 源文件

MDA - ARM: 工程文件

9. *GPIO*

各种寄存器的含义;
*GPIO*工作状态的分类。

10. LED

- (1) 经过锁存器连接到PC8~PC15, 低电平(RESET)点亮, 高电平(SET)熄灭;
锁存器: 引脚共用问题, 防止控制LCD时扰乱LED工作。

- (2) *GPIO Output Level: High*

- (3) *GPIO Mode: Output push and pull*
(开漏输出要么输出不确定状态, 要么输出低电平, 不可取)

- (4) PD2 引脚配置为:

GPIO Output Level: Low

- (5) LED 显示代码:

```
void LED_Dis(unsigned char ucLED)
{
    HAL_GPIO_WritePin(GPIOC, GPIO_PIN_All, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, GPIO_PIN_RESET);

    HAL_GPIO_WritePin(GPIOC, ucLED<<8, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, GPIO_PIN_SET);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_2, GPIO_PIN_RESET);
}
```

11. KEY

- (1) *PB0 PB1 PB2 PA0*

— *GPIO INPUT*

- (2) 滴答定时器

频率与主频相同, 为80MHz; LOAD值为80000;

$\frac{80000}{80M} = 1ms$, 即每 1ms 进入一次中断; uwTick每1ms加一。

HAL_DELAY();函数是对uwTick值的应用, while 循环中使该值累加 1,
直到等于输入变量值大小, 即跳出循环。

- (3) 按键电平

PB0, PB1, PB2, PA0

按钮未按下: 高电平 (SET);

按钮按下: 低电平(RESET)。

(4) 标准按键处理代码段

```
If ( (uwTick-key_setpoint)<100 ) return;
key_setpoint=uwTick;
key_val=KEY_SCAN();
key_down = key_val & (key_old ^ key_val);
key_up    = ~key_val & (key_old ^ key_val);
key_old   = key_val;
```

(5) 按键扫描代码

```
unsigned char KEY_SCAN(void)
{

    unsigned char key_Val = 0;

    if (HAL_GPIO_ReadPin(GPIOA,GPIO_PIN_0)==GPIO_PIN_RESET)
    {
        key_Val = 4;
    }

    if (HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_0)==GPIO_PIN_RESET)
    {
        key_Val = 1;
    }

    if (HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_1)==GPIO_PIN_RESET)
    {
        key_Val = 2;
    }

    if (HAL_GPIO_ReadPin(GPIOB,GPIO_PIN_2)==GPIO_PIN_RESET)
    {
        key_Val = 3;
    }
}
```



```

    }

    return key_Val;
}

```

12. *LCD*

(移植)

Lcd.c

Lcd.h

Fonts.h

Main.c 文件中只要添加 lcd.h 头文件。

一共 0~9, 十行显示

LCD 字符串显示存储变量: unsigned char lcd_disp_string[25];

例:

```
Sprintf ((char *)lcd_disp_string, "%03d", i);
```

```
LCD_DisplayStringLine (Line5,(unsigned char *)lcd_disp_string);
```

13. *USART*

(1) *NVIC* 中断处理系统

(内核器件)

1. 16 级可编程的中断优先级

2. 抢占优先级; 响应优先级 (子优先级)

抢占: 后来的若抢占优先级高, 则先执行。

响应: 同时发生, 响应优先级高的先执行。

3. 中断服务函数在 stm32g4xx_it.c 文件中找到, 跳转到

stm32g4xx_hal_gpio.c 中的 callback 函数, 在主函数中编辑定义新回调函数以代替原来弱函数。

(2) 串口

四个串口, 使用 USART1

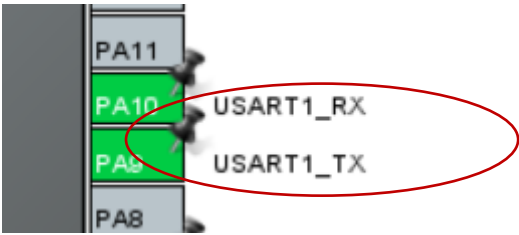
USART1: PA9 : TX

PA10 : RX

波特率: 控制数据发送频率

(3) *CUBEMX*与初始化配置

1.



2.

Mode Asynchronous

Hardware Flow Control (RS232) Disable

☐ Hardware Flow Control (RS485)

Slave Select(NSS) Management Disable

3.

✓ NVIC Settings ✓ DMA Settings ✓ GPIO Settings

✓ Parameter Settings ✓ User Constants

Configure the below parameters :

Search (Ctrl+F)

Basic Parameters

Baud Rate	9600 Bits/s
Word Length	8 Bits (including Parity)
Parity	None
Stop Bits	1

4.

✓ NVIC Settings	✓ DMA Settings	✓ GPIO Settings
✓ Parameter Settings	✓ User Constants	
NVIC Interrupt Table		
	Ena...	Preemption ...
USART1 global interrupt / USART1 wake-up interrupt t...	<input checked="" type="checkbox"/>	0

5.

Group By Peripherals

☒ GPIO
 ☒ RCC
 ☒ USART

Search Signals

Search (Ctrl+F) ☐ Show only Modified Pin

Pin ...	Signal ...	GPIO o...	GPIO ...	GPIO ...	Maxim...	Fast M...	User L...	Modifie
PA9	USAR...	n/a	Alterna...	No pull...	Low	n/a		<input checked="" type="checkbox"/>
PA10	USAR...	n/a	Alterna...	No pull...	Low	n/a		<input checked="" type="checkbox"/>

PA10 Configuration :

GPIO mode Alternate Function Open Drain

GPIO Pull-up/Pull-down No pull-up and no pull-down

Maximum output speed Low

6.

Manage Project Items



Project Items | Folders/Extensions | Books | Project Info/Layer

Project Targets: HAL_01_LED_TEST

Groups: Application/User
BSP
Drivers/STM32G4xx_HAL_Driver
Drivers/CMSIS

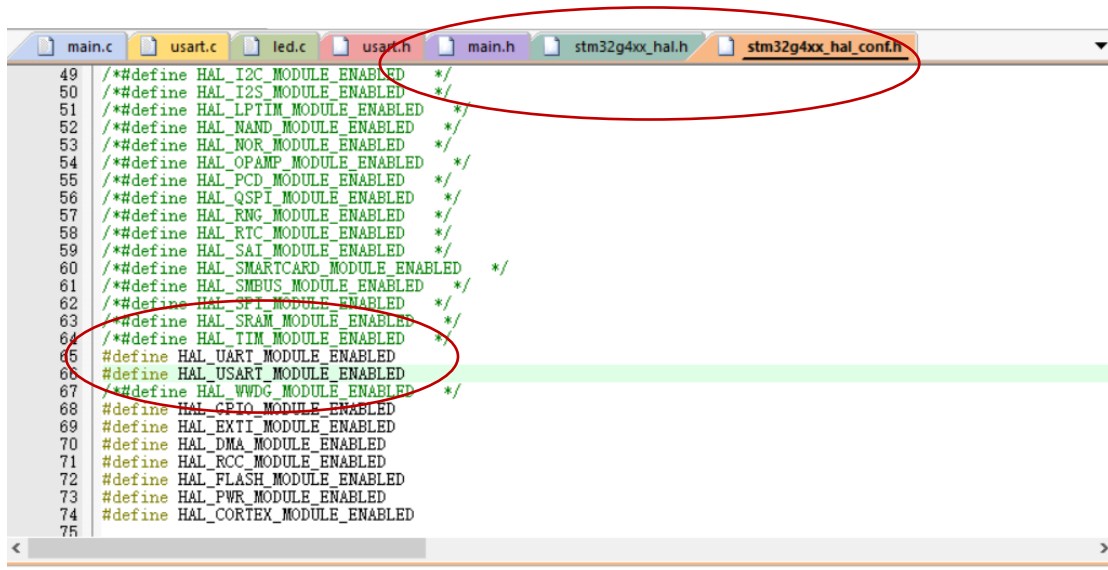
Files: stm32g4xx_hal_tim.c
stm32g4xx_hal_tim_ex.c
stm32g4xx_hal_pwr_ex.c
stm32g4xx_hal.c
stm32g4xx_hal_rcc.c
stm32g4xx_hal_rcc_ex.c
stm32g4xx_hal_flash.c
stm32g4xx_hal_flash_ex.c
stm32g4xx_hal_flash_ramfunc.c
stm32g4xx_hal_gpio.c
stm32g4xx_hal_exti.c
stm32g4xx_hal_dma.c
stm32g4xx_hal_dma_ex.c
stm32g4xx_hal_pwr.c
stm32g4xx_hal_cortex.c
stm32g4xx_hal_uart.c
stm32g4xx_hal_uart_ex.c

Set as Current Target

Add Files...

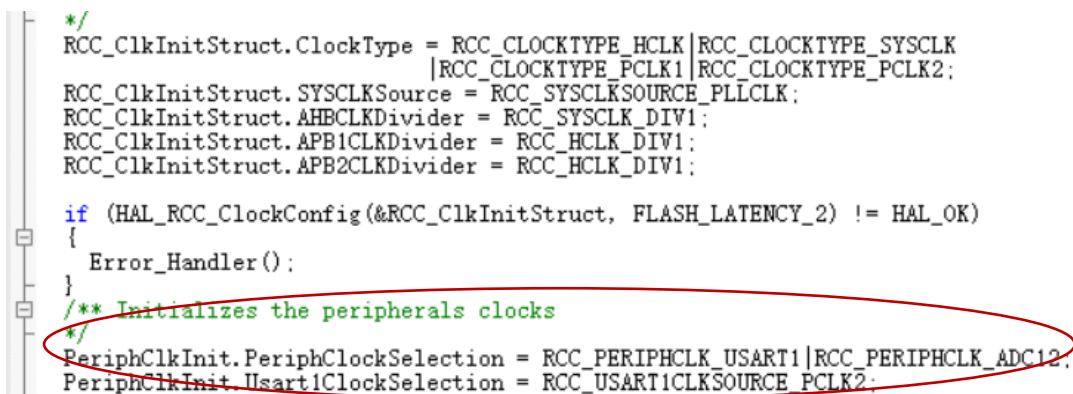
OK Cancel Help

7.



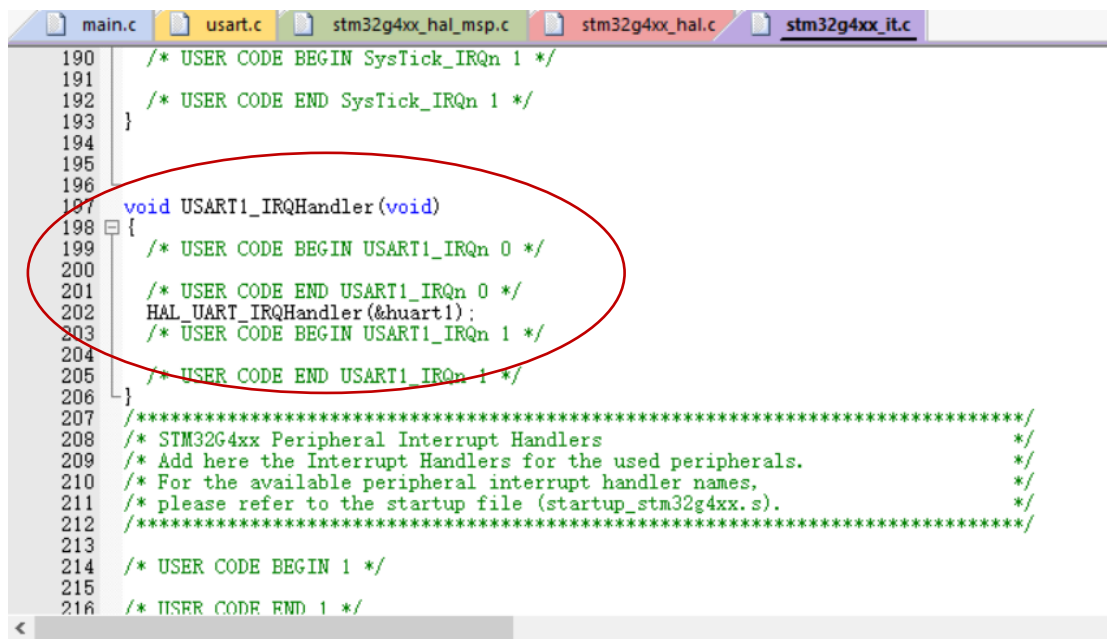
```
49  /**\n50  **\n51  **\n52  **\n53  **\n54  **\n55  **\n56  **\n57  **\n58  **\n59  **\n60  **\n61  **\n62  **\n63  **\n64  **\n65  **\n66  **\n67  **\n68  **\n69  **\n70  **\n71  **\n72  **\n73  **\n74  **\n75  **\n\n#define HAL_I2C_MODULE_ENABLED\n#define HAL_I2S_MODULE_ENABLED\n#define HAL_LPTIM_MODULE_ENABLED\n#define HAL_NAND_MODULE_ENABLED\n#define HAL_NOR_MODULE_ENABLED\n#define HAL_OPAMP_MODULE_ENABLED\n#define HAL_PCD_MODULE_ENABLED\n#define HAL_QSPI_MODULE_ENABLED\n#define HAL_RNG_MODULE_ENABLED\n#define HAL_RTC_MODULE_ENABLED\n#define HAL_SAI_MODULE_ENABLED\n#define HAL_SMARTCARD_MODULE_ENABLED\n#define HAL_SMBUS_MODULE_ENABLED\n#define HAL_SPI_MODULE_ENABLED\n#define HAL_SRAM_MODULE_ENABLED\n#define HAL_TIM_MODULE_ENABLED\n#define HAL_UART_MODULE_ENABLED\n#define HAL_USART_MODULE_ENABLED\n#define HAL_WWDG_MODULE_ENABLED\n\n#define HAL_GPIO_MODULE_ENABLED\n#define HAL_EXTI_MODULE_ENABLED\n#define HAL_DMA_MODULE_ENABLED\n#define HAL_RCC_MODULE_ENABLED\n#define HAL_FLASH_MODULE_ENABLED\n#define HAL_PWR_MODULE_ENABLED\n#define HAL_CORTEX_MODULE_ENABLED
```

8. 修改时钟配置



```
*/\nRCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLOCK\n                                |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;\nRCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;\nRCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;\nRCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;\nRCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;\n\nif (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)\n{\n    Error_Handler();\n}\n\n/* Initializes the peripherals clocks\n*/\nPeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART1|RCC_PERIPHCLK_ADC12;\nPeriphClkInit.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
```

9. It.c 文件添加中断服务函数



```
190  /* USER CODE BEGIN SysTick_IRQn 1 */\n191  /* USER CODE END SysTick_IRQn 1 */\n192  }\n193  }\n194  }\n195  }\n196  }\n197  void USART1_IRQnHandler(void)\n198  {\n199  /* USER CODE BEGIN USART1_IRQn 0 */\n200  /* USER CODE END USART1_IRQn 0 */\n201  HAL_UART_IRQHandler(&huart1);\n202  /* USER CODE BEGIN USART1_IRQn 1 */\n203  /* USER CODE END USART1_IRQn 1 */\n204  }\n205  }\n206  }\n207  /*\n208  /* STM32G4xx Peripheral Interrupt Handlers\n209  /* Add here the Interrupt Handlers for the used peripherals.\n210  /* For the available peripheral interrupt handler names,\n211  /* please refer to the startup file (startup_stm32g4xx.s).\n212  /*\n213  /*\n214  /* USER CODE BEGIN 1 */\n215  /* USER CODE END 1 */\n216  */
```

(4) 串口发送例程

```
sprintf(str, "%04d:Hello,world.\r\n", counter);
HAL_UART_Transmit(&huart1, (unsigned char *)str, strlen(str), 50 );
                                串口号          内容          长度          时间（之内）

HAL_Delay(500);

if(++counter==10000)
    counter=0;
```

(5) 串口接收配置

```
HAL_UART_Receive_IT( &huart1,  &rx,  1);
```

原弱函数：（在`stm32g4x_hal_uart.c`中查找）

```
__weak void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{

}
```

修改后（例：点灯）

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{

    LED_Dis(0XFF);
    HAL_Delay(300);
    LED_Dis(0X00);

    HAL_UART_Receive_IT(&huart1, &rx, 1);

}
```

14. IIC

双向两线通信

修改原代码

```

    unsigned short cErrTime = 0;
    SDA_Input_Mode();
    delay1(DELAY_TIME);
    SCL_Output(1);
    delay1(DELAY_TIME);
    while(SDA_Input())
    {
        cErrTime--;
        delay1(DELAY_TIME);
        if (0 == cErrTime)
        {
            SDA_Output_Mode();
            I2CStop();
            return ERROR;
        }
    }

    SCL_Output(0);
    delay1(DELAY_TIME);
    SDA_Output_Mode();
    return SUCCESS;
}

/**
 * @brief I2C发送确认信号
 * @param None

```

- (1) 24C02 (EEPROM)
 256 个字节，存储容量为 2KB。
 芯片地址为： 1010 (A2 A1 A0) {R=1, W=0}
 A0:写
 A1:读
 读:

```

void iic_read(unsigned char *pf, unsigned char addr, unsigned char
ucnum) /* 要读的内容 (之后放入数组), 读哪个 24c02 地址内容, 读的个
数
{

```

```

    I2CStart();
    I2CSendByte(0xa0);
    I2CWaitAck();

```

```

    I2CSendByte(addr);
    I2CWaitAck();

```

```

    I2CStart();
    I2CSendByte(0xa1);
    I2CWaitAck();

```

```

while(ucnum--)
{
    *pf++ = I2CReceiveByte();

    if(ucnum)
        I2CSendAck();
    else
        I2CSendNotAck();
}

I2CStop();
}

```

写：

```

void iic_write(unsigned char *pf, unsigned char addr, unsigned char
ucnum) /* 要写的内容（放入数组）， 24c02 地址， 写的个数
{

    I2CStart();
    I2CSendByte(0xa0); /*器件地址
    I2CWaitAck();

    I2CSendByte(addr); /*字节地址
    I2CWaitAck();

    while(ucnum--)

    {
        I2CSendByte(*pf++);
        I2CWaitAck();
    }

    I2CStop();
    delay1(500);
}

```

(2) MCP4017

电阻值最大为 100k

N 大小正比于电阻值大小 N 范围为 0~127

读出分压电压可知电阻值大小

写:

```
void mcp4017_write(unsigned char value)

{

    I2CStart();
    I2CSendByte(0X5E);
    I2CWaitAck();

    I2CSendByte(value);
    I2CWaitAck();
    I2CStop();

}
```

读:

```
uint8_t mcp4017_read(void)

{

    uint8_t value;
    I2CStart();
    I2CSendByte(0X5F);
    I2CWaitAck();

    value = I2CReceiveByte();
    I2CSendNotAck();
    I2CStop();

    return value;

}
```

例程:

阻值计算:

```
(float) ( mcp4017_read() * 0.7874 )
```


分压计算：

```
3.3 * (float) ( mcp4017_read() * 0.7874 ) / ( mcp4017_read() * 0.7874 + 10 );
```

15. ADC

(1) 结构

两个最高 12 位， 为 ADC1、ADC2

模拟量转换为数字量

(3.3V) (12 位: 4096)

(2) 内部配置

一般使用 ADC1 PB12

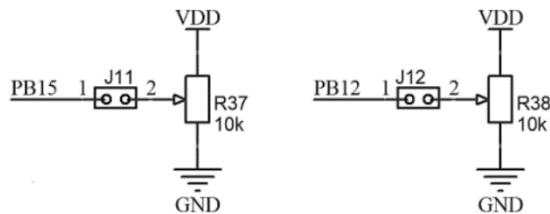
ADC2 PB15

对应两个滑动变阻器

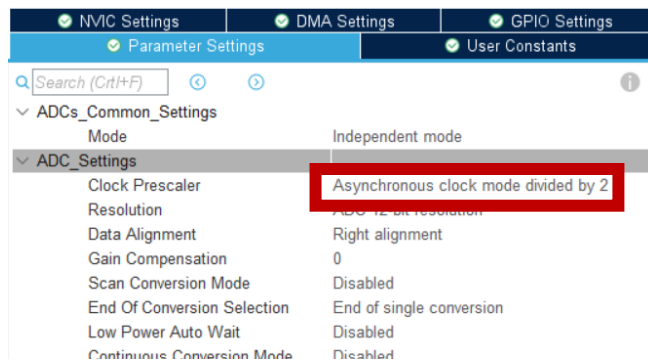
(mcp4017 对应的 ADC1 PB14 不常用)

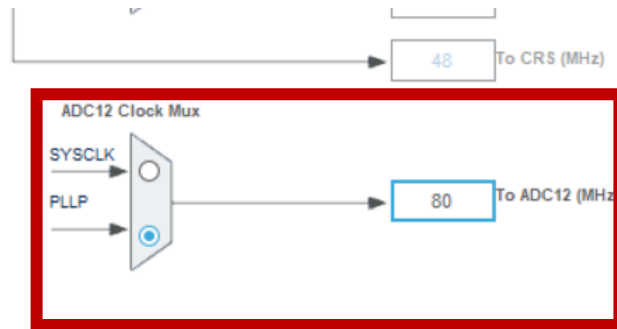
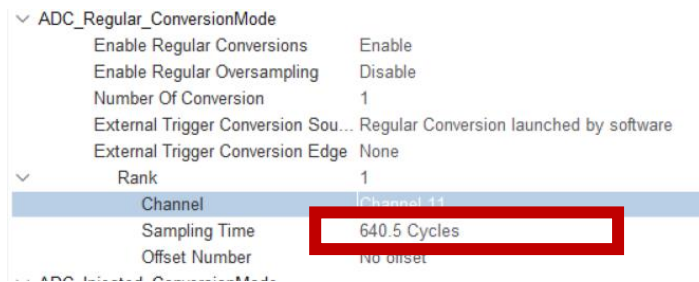
ADC

	IN1	IN2	IN3	IN4	IN5	IN6	IN7	IN8	IN9	IN10	IN11	IN12	IN13	IN14	IN15	IN16	IN17
ADC1	PA0	PA1	PA2	PA3	PB14	PC0	PC1	PC2	PC3	PF0	PB12	PB1		PB11	PB0	Tem	Vbat
ADC2	PA0	PA1	PA6	PA7	PC4	PC0	PC1	PC2	PC3	PF1	PC5	PB2	PA5	PB11	PB15		PA4

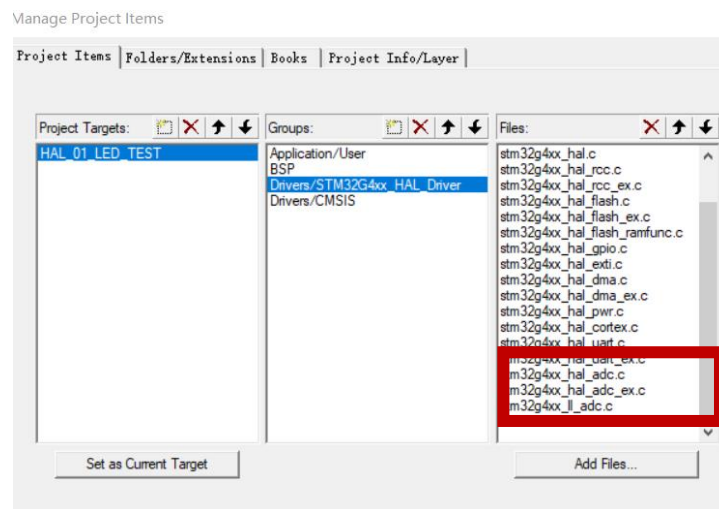


(2) CUBEMX 配置





(3) 初始化代码



```

29  /* Exported constants -----
30
31  /* ##### Module Selection #####
32  /**
33   * @brief This is the list of modules to be used in
34   */
35
36  #define HAL_MODULE_ENABLED
37
38  #define HAL_ADC_MODULE_ENABLED
39  /*#define HAL_COMP_MODULE_ENABLED */
40  /*#define HAL_CORDIC_MODULE_ENABLED */
41  /*#define HAL_CRC_MODULE_ENABLED */
42  /*#define HAL_CRYP_MODULE_ENABLED */
43  /*#define HAL_DAC_MODULE_ENABLED */
44  /*#define HAL_FDCAN_MODULE_ENABLED */
45  /*#define HAL_FMAC_MODULE_ENABLED */
46  /*#define HAL_HRTIM_MODULE_ENABLED */
47  /*#define HAL_IRDA_MODULE_ENABLED */
48  /*#define HAL_IWDG_MODULE_ENABLED */
49  /*#define HAL_I2C_MODULE_ENABLED */
50  /*#define HAL_I2S_MODULE_ENABLED */

```

```
main.c i2c.c i2c.h adc.c adc.h
1 #include "main.h"
2
3
4 extern ADC_HandleTypeDef hadc1;
5
6 /* USER CODE BEGIN Private defines */
7
8 /* USER CODE END Private defines */
9
10 void ADC1_Init(void);
```

```
main.c i2c.c i2c.h adc.c adc.h main.h
157
158 /**
159  * @brief System Clock Configuration
160  * @retval None
161  */
162 void SystemClock_Config(void)
163 {
164     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
165     RCC_OscInitTypeDef RCC_OscInitStruct = {0};
166     RCC_PeriphCLKInitTypeDef PeriphClkInit = {0};
167
168     /** Configure the main internal regulator output voltage
169     */
170     HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1)
171     /** Initializes the RCC Oscillators according to the specified
172     * in the RCC_OscInitTypeDef structure.
173     */
174     RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
175     RCC_OscInitStruct.HSEState = RCC_HSE_ON;
176     RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
177     RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
178     RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV3;
179     RCC_OscInitStruct.PLL.PLLN = 20;
180     RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
181     RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
182     RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
183     if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
184     {
185         Error_Handler();
186     }
187     /** Initializes the CPU, AHB and APB buses clocks
188     */
```

```

main.c  i2c.c  i2c.h  adc.c  adc.h  main.h  stm32g4xx_hal.h
184 {
185     Error_Handler();
186 }
187 /** Initializes the CPU, AHB and APB buses clocks
188 */
189 RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
190                             |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
191 RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
192 RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
193 RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
194 RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
195
196 if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
197 {
198     Error_Handler();
199 }
200
201 /** Initializes the peripherals clocks
202 */
203 PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART1|RCC_PERIPHCLK_ADC12;
204 PeriphClkInit.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
205 PeriphClkInit.Adc12ClockSelection = RCC_ADC12CLKSOURCE_SYSCLK;
206 if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
207 {
208     Error_Handler();
209 }
210

```

(4) 函数

uint16_t getADC(void)

{

uint16_t adc1=0;

HAL_ADC_Start(&hadc1);

adc1 = HAL_ADC_GetValue(&hadc1);

return adc1;

}

uint16_t r38_voltage;

r38_voltage = getADC1();

sprintf((char *)lcd_disp_string," R38_VOL: %6.2fV",r38_voltage*3.3/4096);

LCD_DisplayStringLine(Line8,(unsigned char *)lcd_disp_string);

16. *TIM*

(1) 基础知识

10 个定时器:

2 个基本定时器 (*TIM6 TIM7*)

3 个通用定时器 (*TIM2 TIM3 TIM4*) 全功能定时器

3 个通用定时器 (*TIM15 TIM16 TIM17*) 单通道或双通道定时器

2 个高级定时器 (*TIM1 TIM8*)

功能使用:

基本定时

PWM 脉冲输入捕获

PWM 脉冲输出

计数方式:

中心对齐 (Up_Down)

向上计数 (Up)

向下计数 (Down)

重载计数值不一定是 65535, 可以通过寄存器 (*ARR*) 修改

ARR 数值越大, 计数周期越长

分频器:

实际分频为输入数字+1

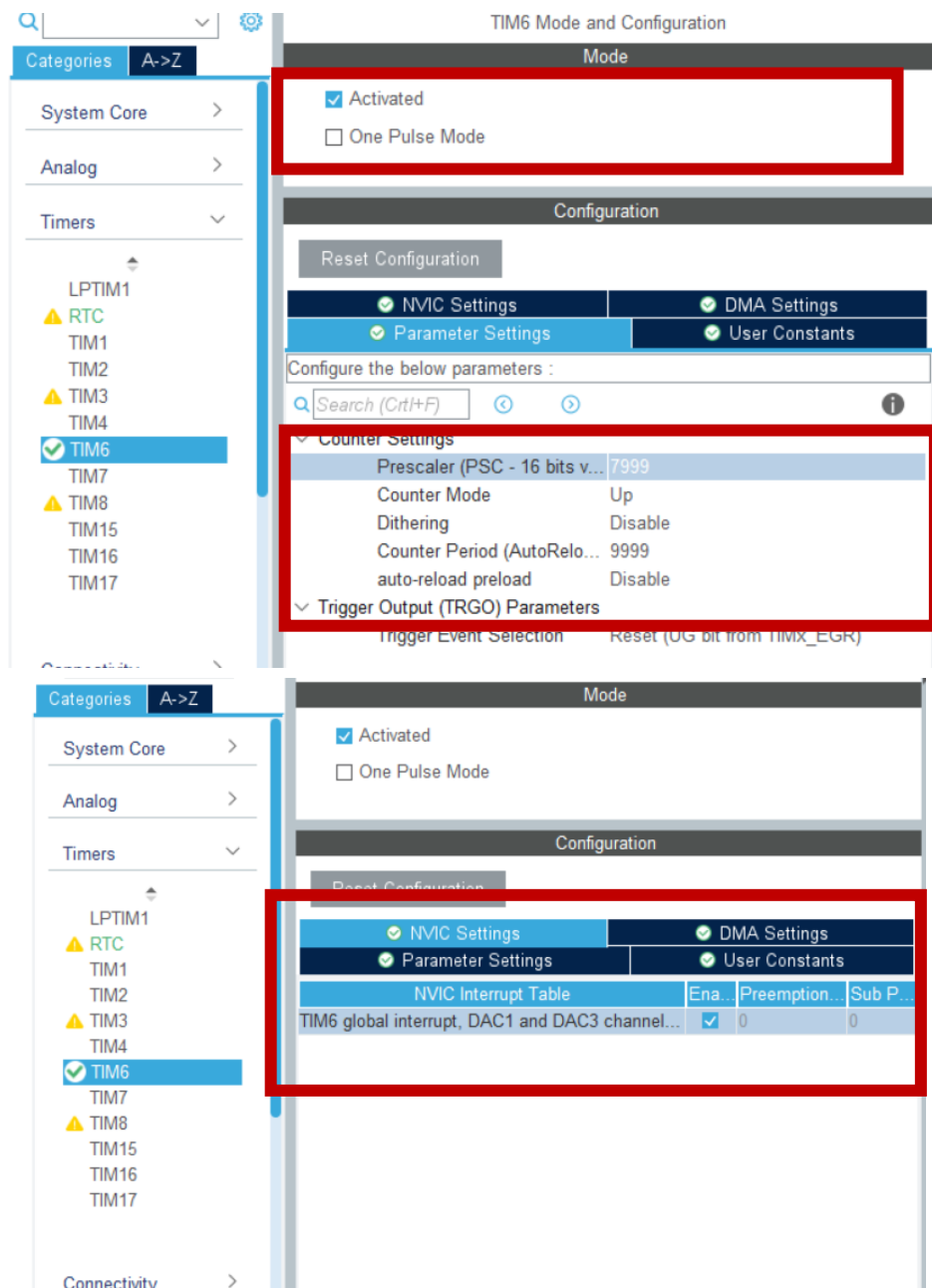
分频越高, 计数越快

(2) 基本定时器:

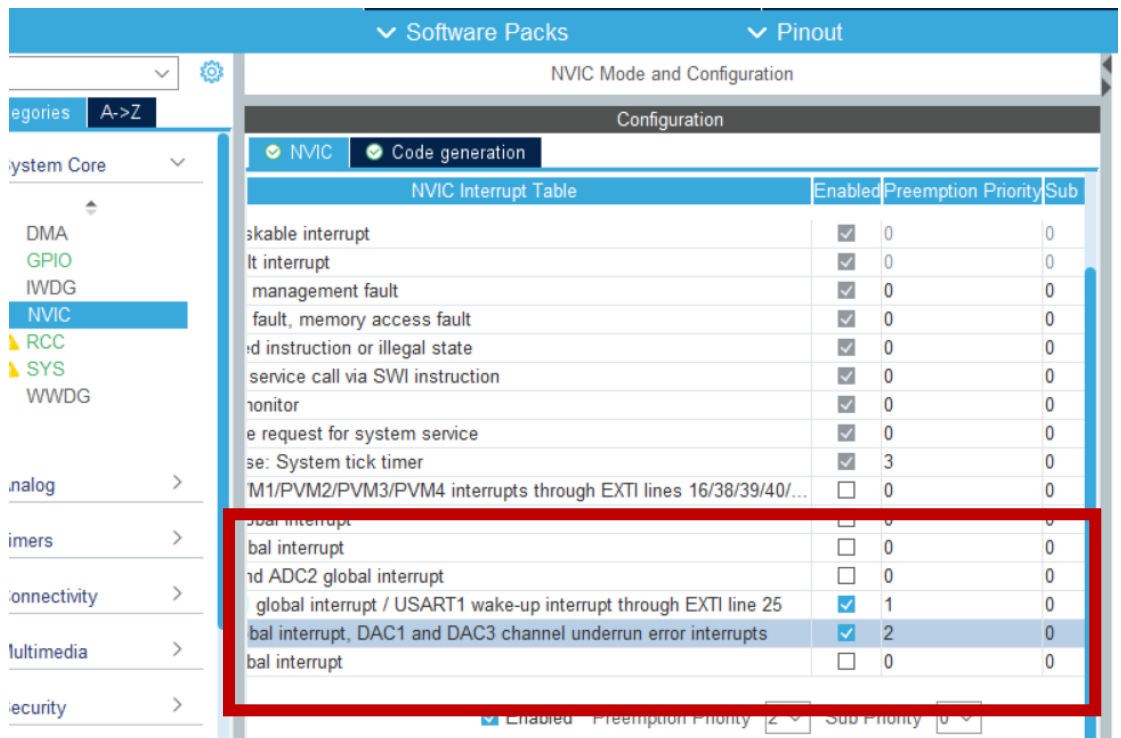
0-65536, up, 时钟来源 *APB1*

1. *CUBEMX* 配置

(1s 产生一次中断 $80M=8000 * 10000$)

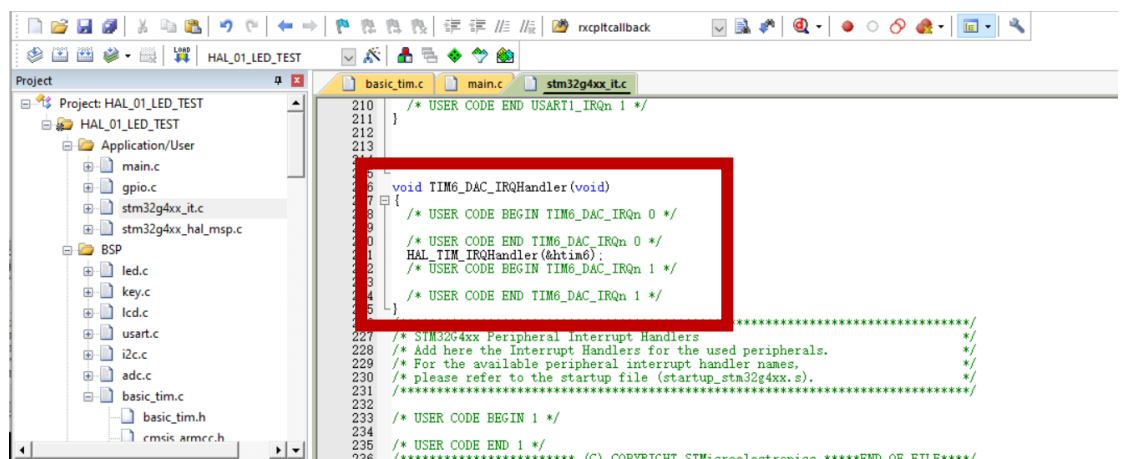
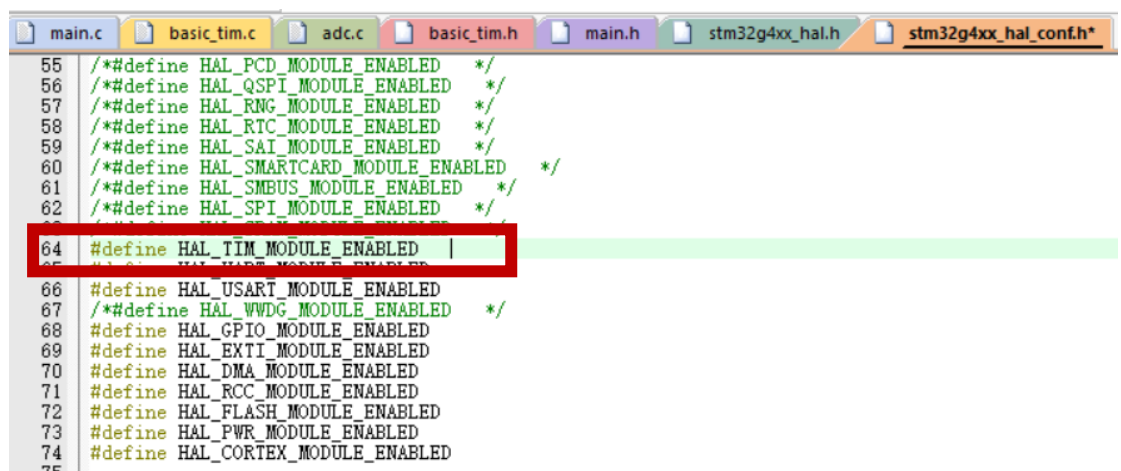


(基本定时器优先级设置比串口低一些)



2. 代码初始化

(反初始化代码段用不着)



3. 使用示例

修改此处回调函数：

```
3871 }
3872 /* TIM Update event */
3873 if (__HAL_TIM_GET_FLAG(htim, TIM_FLAG_UPDATE) != RESET)
3874 {
3875     if (__HAL_TIM_GET_IT_SOURCE(htim, TIM_IT_UPDATE) != RESET)
3876     {
3877         __HAL_TIM_CLEAR_IT(htim, TIM_IT_UPDATE);
3878         #if (USE_HAL_TIM_REGISTER_CALLBACKS == 1)
3879             htim->PeriodElapsedCallback(htim);
3880         #else
3881             HAL_TIM_PeriodElapsedCallback(htim);
3882         #endif /* USE_HAL_TIM_REGISTER_CALLBACKS */
3883     }
3884 }
3885 /* TIM Break input event */
3886 if (__HAL_TIM_GET_FLAG(htim, TIM_FLAG_BREAK) != RESET)
3887 {
3888     if (__HAL_TIM_GET_IT_SOURCE(htim, TIM_IT_BREAK) != RESET)
3889     {
3890         __HAL_TIM_CLEAR_IT(htim, TIM_IT_BREAK);
3891         #if (USE_HAL_TIM_REGISTER_CALLBACKS == 1)
3892             htim->BreakCallback(htim);
3893         #else
```

(主函数打开中断)

```
133
134
135     mcp4017_write(56);
136
137
138     //中断打开处
139
140     HAL_UART_Receive_IT(&huart1, &rx, 1);
141
142     HAL_TIM_Base_Start_IT(&htim6);
143
144
145
```

回调函数修改如下：回调函数结束前需要再次打开中断

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    i++;

    HAL_TIM_Base_Start_IT(&htim6);
}
/**
```


(3) 高级定时器（输入捕获）

0-65536 , up、down、up-down

输入捕获 与 输出比较不能同时启用

从模式控制器：接收信号，可以工作在 **Reset 模式**，enable 模式

溢出计数器写为 0 时相当于没有

输入捕获：

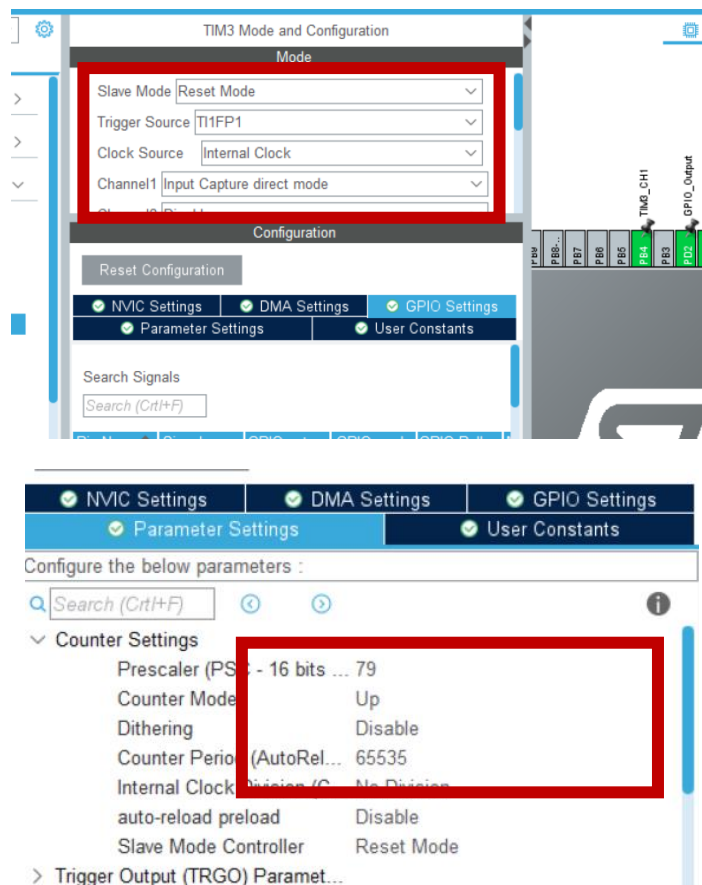
- a) 普通输入捕获模式
- b) **PWM 输入模式：**
 - 测量 pwm 周期、占空比等
 - CCR2: 脉冲宽度
 - CCR1: 周期

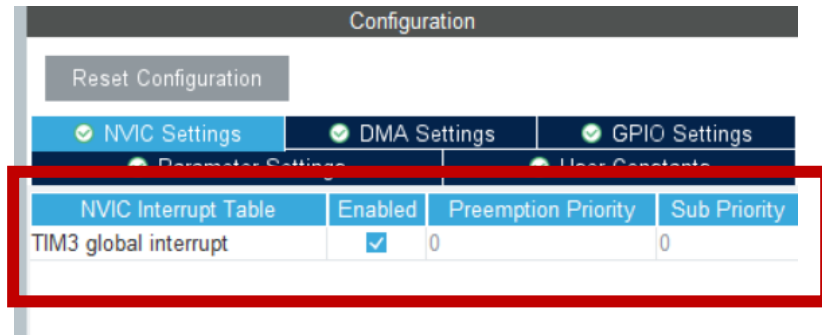
输出比较：

- a) 匹配时输出有效/无效电平模式
- b) 匹配时电平翻转模式
- c) 强制输出电平
- d) **PWM 模式：(PWM1/PWM2)**
 - 两者互补

1. CUBEMX配置

PA15 PB4





2. 代码初始化配置

```

55  /*#define HAL_PCD_MODULE_ENABLED */
56  /*#define HAL_QSPI_MODULE_ENABLED */
57  /*#define HAL_RNG_MODULE_ENABLED */
58  /*#define HAL_RTC_MODULE_ENABLED */
59  /*#define HAL_SAI_MODULE_ENABLED */
60  /*#define HAL_SMARTCARD_MODULE_ENABLED */
61  /*#define HAL_SMBUS_MODULE_ENABLED */
62  /*#define HAL_SPI_MODULE_ENABLED */
63  /*#define HAL_UART_MODULE_ENABLED */
64  #define HAL_TIM_MODULE_ENABLED
65  #define HAL_UART_MODULE_ENABLED
66  #define HAL_USART_MODULE_ENABLED
67  /*#define HAL_USB_OTF_FS_MODULE_ENABLED */
68  #define HAL_GPIO_MODULE_ENABLED
69  #define HAL_EXTI_MODULE_ENABLED
70  #define HAL_DMA_MODULE_ENABLED
71  #define HAL_RCC_MODULE_ENABLED
72  #define HAL_FLASH_MODULE_ENABLED
73  #define HAL_PWR_MODULE_ENABLED
74  #define HAL_CORTEX_MODULE_ENABLED
75

```

```

221 HAL_TIM_IRQHandler(&htim6);
222 /* USER CODE BEGIN TIM6_DAC_IRQHandler 1 */
223
224 /* USER CODE END TIM6_DAC_IRQHandler 1 */
225 }
226
227
228
229
230 void TIM3_IRQHandler(void)
231 {
232     /* USER CODE BEGIN TIM3_IRQHandler 0 */
233
234     /* USER CODE END TIM3_IRQHandler 0 */
235     HAL_TIM_IRQHandler(&htim3);
236     /* USER CODE BEGIN TIM3_IRQHandler 1 */
237
238     /* USER CODE END TIM3_IRQHandler 1 */
239 }
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

3. 使用示例（单通道测频率）

```

//中断打开处

HAL_UART_Receive_IT(&huart1, &rx, 1);

HAL_TIM_Base_Start_IT(&htim6);

HAL_TIM_Base_Start(&htim3);
HAL_TIM_IC_Start_IT(&htim3, TIM_CHANNEL_1);

```

修改此处回调函数：

```
/*  
_weak void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)  
{  
    /* Prevent unused argument(s) compilation warning */  
    UNUSED(htim);  
  
    /* NOTE : This function should not be modified, when the callback is needed,  
               the HAL_TIM_IC_CaptureCallback could be implemented in the user file  
    */  
}  
/**
```

修改如下：

```
void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)  
{  
    if ( htim-> Instance == TIM3)  
        //判断定时器  
    {  
        if(htim-> Channel == HAL_TIM_ACTIVE_CHANNEL_1)  
        {  
            //判断通道  
  
            PWM1_T_COUNT = HAL_TIM_ReadCapturedValue(&htim3, TIM_CHANNEL_1) + 1 ;  
  
            // 计数值，可通过滑动变阻器调整  
            // 频率计算： 由于是 80M 的 79 + 1 分频，计数频率为 1000000  
  
            // 则该 pwm 周期为 PWM1_T_COUNT / 1000000  
            // 频率为 1000000 / PWM1_T_COUNT  
            PWM1_T_period = (unsigned int)PWM1_T_COUNT / 1000000;  
            PWM1_T_freq = (unsigned int)(1000000 / PWM1_T_COUNT);  
  
        }  
    }  
}
```

4. 使用示例 (双通道测占空比)

TIM2 Mode and Configuration

Mode

Slave Mode Reset Mode ▼

Trigger Source TI1FP1 ▼

Clock Source Internal Clock ▼

Channel1 Input Capture direct mode ▼

Channel2 Input Capture indirect mode ▼

Channel3 Disable ▼

Channel4 Disable ▼

Configuration

Reset Configuration

✓ NVIC Settings
✓ DMA Settings
✓ GPIO Settings

✓ Parameter Settings
✓ User Constants

NVIC Interrupt Table	Enabled	Preemption Priority	Sub P
TIM2 global interrupt	<input checked="" type="checkbox"/>	0	0

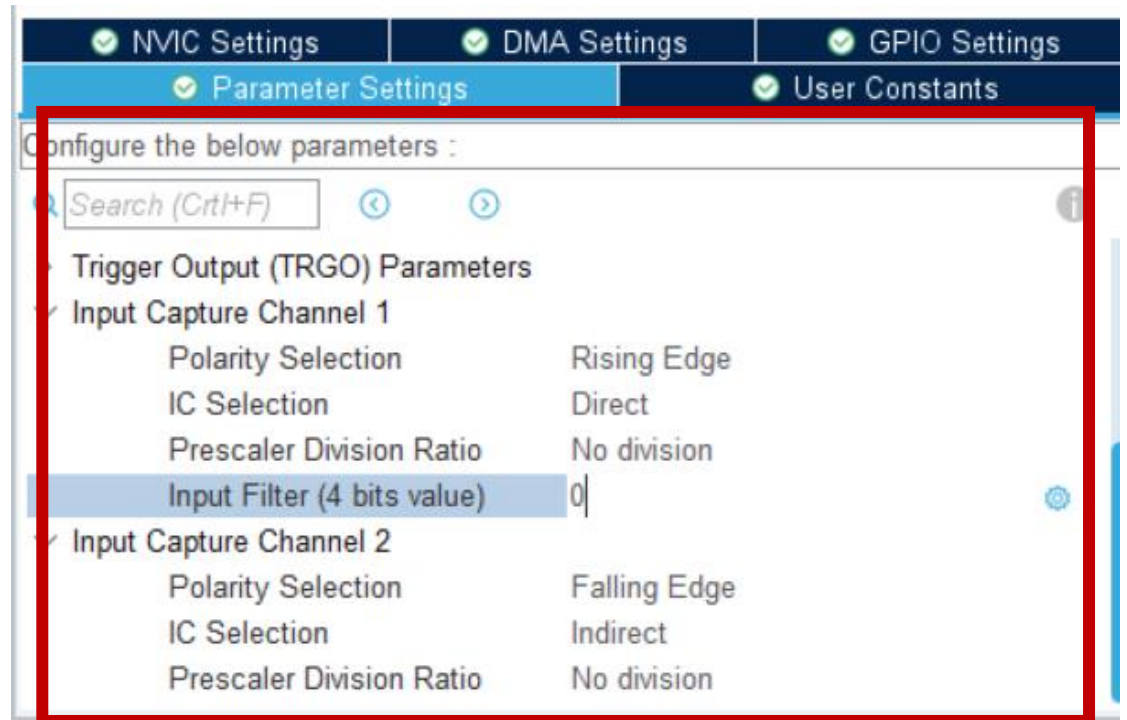
✓ NVIC Settings
✓ DMA Settings
✓ GPIO Settings

✓ Parameter Settings
✓ User Constants

Configure the below parameters :

Search (Ctrl+F)
⏪ ⏩ ⓘ

Prescaler (PSC - 16 bits val...	79
Counter Mode	Up
Dithering	Disable
Counter Period (AutoReload...	65535
Internal Clock Division (CKD)	No Division
auto-reload preload	Disable
Slave Mode Controller	Reset Mode
> Trigger Output (TRGO) Parameters	
✓ Input Capture Channel 1	
Polarity Selection	Rising Edge



代码初始化配置与单通道相同

占空比计算公式如下：

```

if ( htim -> Instance == TIM2)
    //判断定时器
{
    if(htim -> Channel == HAL_TIM_ACTIVE_CHANNEL_1)
    {
        //判断通道
        PWM2_T_COUNT = HAL_TIM_ReadCapturedValue(&htim2, TIM_CHANNEL_1) + 1 ;
        pwm2_duty = (float) PWM2_D_COUNT / PWM2_T_COUNT ;
    }
    if(htim -> Channel == HAL_TIM_ACTIVE_CHANNEL_2)
    {
        //判断通道
        PWM2_D_COUNT = HAL_TIM_ReadCapturedValue(&htim2, TIM_CHANNEL_2) + 1 ;
    }
}

```

(4) 高级定时器（输出比较）

方波 & pwm 波

方波：占空比 0.5 的 PWM 波

(方波) -> 利用电平翻转模式

1. CUBEMX配置

Pinout & Configuration | **Clock Configuration** | **Software Packs** | **Pinout**

TIM4 Mode and Configuration

Mode

Slave Mode	Disable
Trigger Source	Disable
Clock Source	Internal Clock
Channel1	Output Compare CH1
Channel2	Output Compare CH2
Channel3	Disable
Channel4	Disable

Configuration

Reset Configuration

☒ NVIC Settings | ☒ DMA Settings | ☒ GPIO Settings

☒ Parameter Settings | ☒ User Constants

Configure the below parameters :

Search (Ctrl+F)

Counter Settings

Prescaler (PSC - 16 bits value)	79
Counter Mode	Up
Dithering	Disable
Counter Period (AutoReload Regist...	65535
Internal Clock Division (CKD)	No Division
auto-reload preload	Enable

Output Compare Channel 1

Mode	Toggle on match
Pulse (16 bits value)	100
Output compare preload	Disable
CH Polarity	High

Output Compare Channel 2

Mode	Toggle on match
Pulse (16 bits value)	100
Output compare preload	Disable
CH Polarity	High

Configuration			
Reset Configuration			
✓ NVIC Settings	✓ DMA Settings	✓ GPIO Settings	
✓ Parameter Settings		✓ User Constants	
NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority
TIM4 global interrupt	<input checked="" type="checkbox"/>	0	0

2. 代码初始化配置

```
void HAL_TIM_MspPostInit(TIM_HandleTypeDef* timHandle)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    if(timHandle->Instance==TIM4)
    {
        /* USER CODE BEGIN TIM4_MspPostInit 0 */

        /* USER CODE END TIM4_MspPostInit 0 */

        __HAL_RCC_GPIOA_CLK_ENABLE();
        /**TIM4 GPIO Configuration
        PA11 -----> TIM4_CH1
        PA12 -----> TIM4_CH2
        */
        GPIO_InitStruct.Pin = GPIO_PIN_11|GPIO_PIN_12;
        GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
        GPIO_InitStruct.Pull = GPIO_NOPULL;
        GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_MEDIUM;
        GPIO_InitStruct.Alternate = GPIO_AF10_TIM4;
        HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

        /* USER CODE BEGIN TIM4_MspPostInit 1 */

        /* USER CODE END TIM4_MspPostInit 1 */
    }
}
```

```
12 |
13 |
14 | void TIM4_IRQHandler(void)
15 | {
16 |     /* USER CODE BEGIN TIM4_IRQn 0 */
17 |
18 |     /* USER CODE END TIM4_IRQn 0 */
19 |     HAL_TIM_IRQHandler(&htim4);
20 |     /* USER CODE BEGIN TIM4_IRQn 1 */
21 |
22 |     /* USER CODE END TIM4_IRQn 1 */
23 | }
24 |
25 |
26 | /* *****
27 | /* STM32G4xx Peripheral Interrupt Handler:
28 | /* Add here the Interrupt Handlers for the
```


3. 使用示例

(两路频率不等方波)

```
66 HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);
67
68
69
70 HAL_TIM_OC_Start_IT(&htim4, TIM_CHANNEL_1);
71 HAL_TIM_OC_Start_IT(&htim4, TIM_CHANNEL_1);
72
73
74
```

(修改此处回调函数)

```
1  * @param htim TIM OC handle
2  * @retval None
3  */
4  weak void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim)
5  {
6      /* Prevent unused argument(s) compilation warning */
7      UNUSED(htim);
8
9      /* NOTE : This function should not be modified, when the callback is r
10         the HAL_TIM_OC_DelayElapsedCallback could be implemented in
11         */
12 }
13
14 /**
15  * @brief Input Capture callback in non-blocking mode
16  * @param htim TIM IC handle
17  * @retval None
18  */
```

```
void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim)
{
    //方波输出中断

    if (htim->Instance == TIM4)
    //判断定时器
    {
        if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_1)
        {
            __HAL_TIM_SET_COMPARE(htim, TIM_CHANNEL_1, (__HAL_TIM_GET_COUNTER(htim))+100); //5k Hz
        }
        if(htim->Channel == HAL_TIM_ACTIVE_CHANNEL_2)
        {
            __HAL_TIM_SET_COMPARE(htim, TIM_CHANNEL_1, (__HAL_TIM_GET_COUNTER(htim))+500); //1k Hz
        }
    }
}
```


(PWM 波) -> (改变 CCR 寄存器值来改变占空比)

1. CUBEMX配置

Mode

Slave Mode Disable

Trigger Source Disable

☒ Internal Clock

Channel1 PWM Generation CH1

Channel2 PWM Generation CH2

Combined Channels Disable

☐ Activate Break Input

☐ XOR activation

(1K Hz 为例)

Parameter Settings

User Constants

Configure the below parameters :

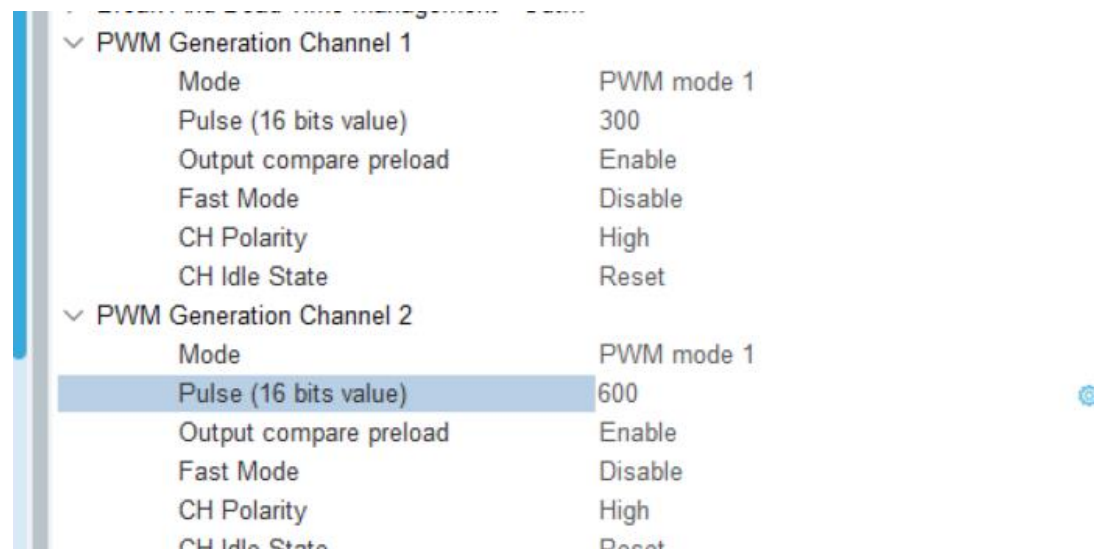
Search (Ctrl+F)

Counter Settings

Prescaler (PSC - 16 bits value)	79
Counter Mode	Up
Dithering	Disable
Counter Period (AutoReload Regis...	999
Internal Clock Division (CKD)	No Division
Repetition Counter (RCR - 8 bits v...	0
auto-reload preload	Enable

Trigger Output (TRGO) Parameters

占空比分别为 0.3 和 0.6



2. 代码初始化配置

(不用开中断)

3. 使用示例

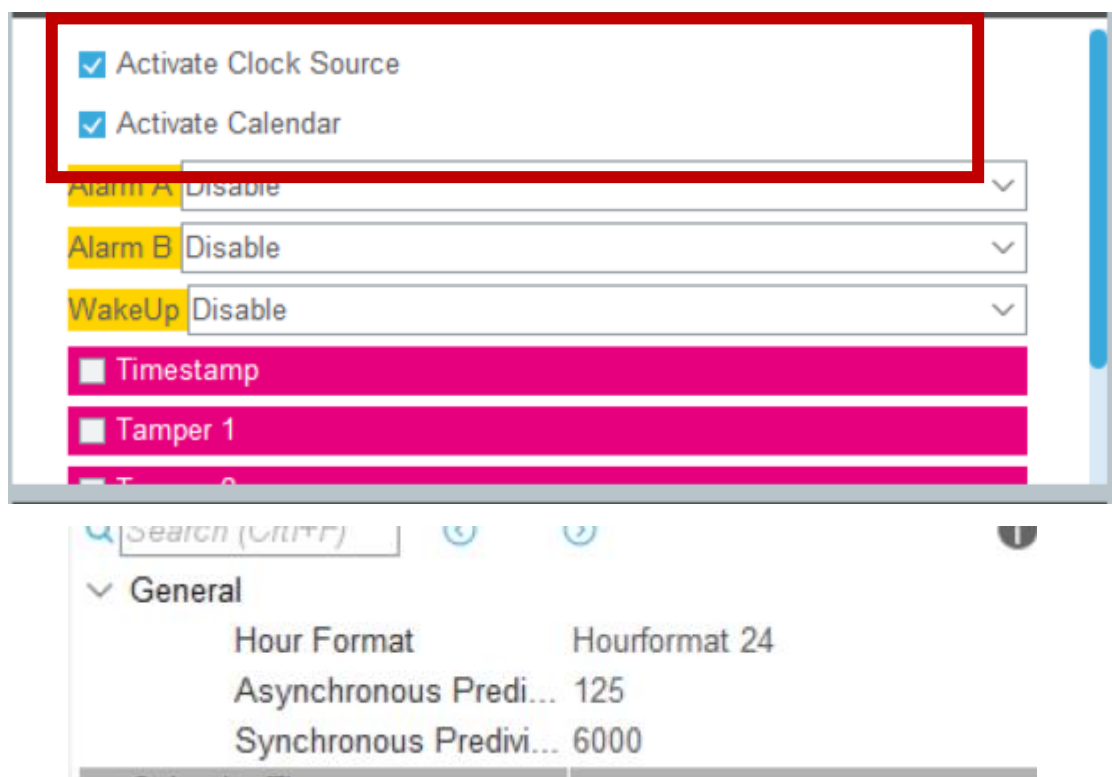
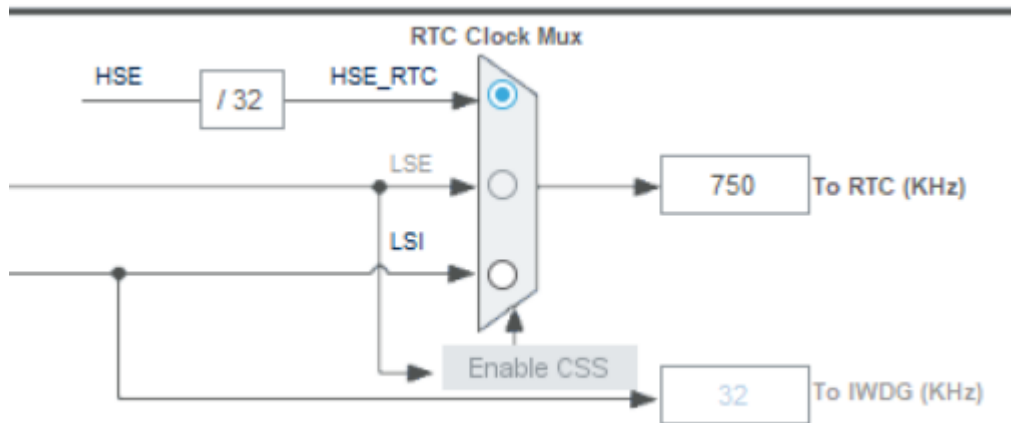
```
HAL_TIM_PWM_Start(&htim15, TIM_CHANNEL_1);  
HAL_TIM_PWM_Start(&htim15, TIM_CHANNEL_2);
```

4. 修改占空比

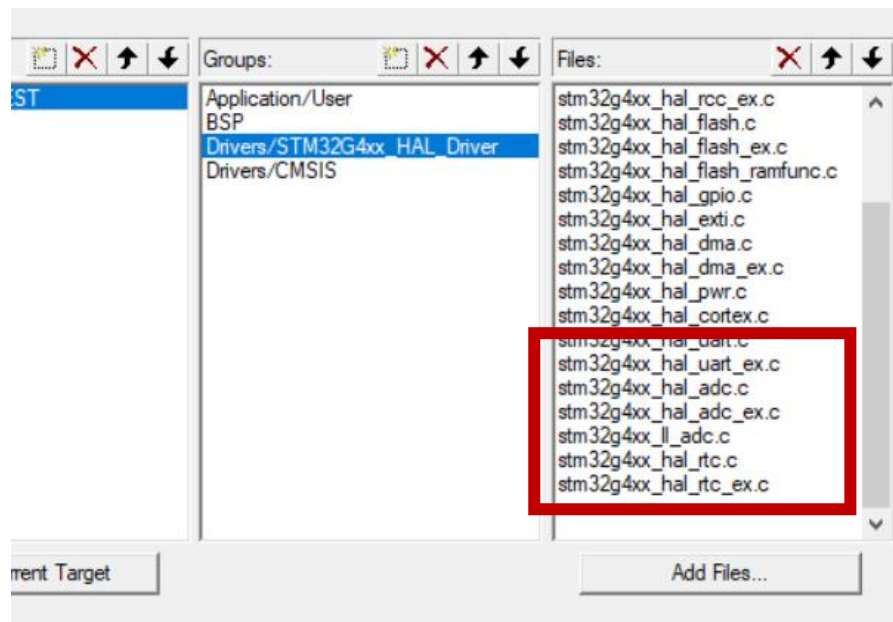
```
HAL_TIM_PWM_Start(&htim16, TIM_CHANNEL_1);  
HAL_TIM_PWM_Start(&htim17, TIM_CHANNEL_1);  
  
//修改占空比  
__HAL_TIM_SET_COMPARE(&htim16, TIM_CHANNEL_1, 800); //duty = 0.8
```

17. *RTC*

- (1) *CUBEMX*配置
BCD 码格式



(2) 代码初始化配置



```
9  /*#define HAL_I2C_MODULE_ENABLED  */
10 /*#define HAL_I2S_MODULE_ENABLED  */
11 /*#define HAL_LPTIM_MODULE_ENABLED */
12 /*#define HAL_NAND_MODULE_ENABLED  */
13 /*#define HAL_NOR_MODULE_ENABLED  */
14 /*#define HAL_OPAMP_MODULE_ENABLED */
15 /*#define HAL_PCD_MODULE_ENABLED  */
16 /*#define HAL_QSPI_MODULE_ENABLED  */
17 /*#define HAL_RNG_MODULE_ENABLED  */
18 #define HAL_RTC_MODULE_ENABLED
19 /*#define HAL_SAI_MODULE_ENABLED  */
20 /*#define HAL_SMARTCARD_MODULE_ENABLED */
21 /*#define HAL_SMBUS_MODULE_ENABLED */
22 /*#define HAL_SPI_MODULE_ENABLED  */
23 /*#define HAL_SRAM_MODULE_ENABLED */
24 /*#define HAL_TIM_MODULE_ENABLED  */
25 #define HAL_UART_MODULE_ENABLED
26 #define HAL_USART_MODULE_ENABLED
27 /*#define HAL_WWDG_MODULE_ENABLED  */
28 #define HAL_GPIO_MODULE_ENABLED
29 #define HAL_EXTI_MODULE_ENABLED
30 #define HAL_DMA_MODULE_ENABLED
31 #define HAL_RCC_MODULE_ENABLED
32 #define HAL_FLASH_MODULE_ENABLED
```

```

1  /** Initializes the peripherals clocks
2
3  4  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_RTC|RCC_PERIPHCLK_USART1
5      |RCC_PERIPHCLK_ADC12;
6  PeriphClkInit.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
7  PeriphClkInit.Adc12ClockSelection = RCC_ADC12CLKSOURCE_SYSCLK;
8  PeriphClkInit.RTCClockSelection = RCC_RTCCLKSOURCE_HSE_DIV32;
9
10 if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
11 {
12     Error_Handler();
13 }
14

```

(3) 程序实例:

变量定义:

```

RTC_TimeTypeDef  H_M_S;
RTC_DateTypeDef  Y_M_D;

```

写入时间、日期:

```

HAL_RTC_GetDate(&hrtc, &Y_M_D, RTC_FORMAT_BIN);
HAL_RTC_GetTime(&hrtc, &H_M_S, RTC_FORMAT_BIN);

```

LCD 显示:

```

sprintf((char *)led_disp_string, " Time: %02d %02d %02d", (int) H_M_S.Hours, (int) H_M_S.Minutes, (int) H_M_S.Seconds);
LCD_DisplayStringLine(Line0, (unsigned char *)led_disp_string);

sprintf((char *)led_disp_string, " Date: %02d %02d %02d", (int) Y_M_D.Year, (int) Y_M_D.Month, (int) Y_M_D.Date);
LCD_DisplayStringLine(Line2, (unsigned char *)led_disp_string);

```

18. 相关知识

1. 变量类型定义

```
1  #ifndef __int8_t_defined
2  # define __int8_t_defined
3  typedef signed char      int8_t;
4  typedef short int        int16_t;
5  typedef int              int32_t;
6  # if __WORDSIZE == 64
7  typedef long int         int64_t;
8  # else
9  __extension__
10 typedef long long int     int64_t;
11 # endif
12 #endif
13
14
15 typedef unsigned char     uint8_t;
16 typedef unsigned short int uint16_t;
17 #ifndef __uint32_t_defined
18 typedef unsigned int       uint32_t;
19 # define __uint32_t_defined
20 #endif
21 #if __WORDSIZE == 64
22 typedef unsigned long int  uint64_t;
23 #else
24 __extension__
25 typedef unsigned long long int uint64_t;
26 #endif
```

2. 常用引脚

- (1) PWM 输入捕获: PA15 PB4
- (2) PWM 输出 PWM 波: PA6 PA7

(3) ADC1: PB12 ->>>>R38

(4) ADC2: PB15 ->>>>R37

3. 注意点

(1) 24C02 读和写之间若连续进行需要相当时间的延时

HAL_DELAY(10)

(2) 这两段初始化代码需要放在主函数最开始:

HAL_Init();

SystemClock_Config();

4. 技巧

(1) 串口字符串常用组合:

```
void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if(receive_num == 0)
    {
        receive_setpoint = uwTick;
        start_flag = 1;
    }

    if(start_flag == 1)
    {
        uart_receiver[receive_num] = rx_buffer;
        receive_num++;
    }

    UART_Start_Receive_IT(&huart1, &rx_buffer, 1);
}
```

```
//part 4
if(((uwTick - receive_setpoint) >=200) && ((uwTick - receive_setpoint) <=300))
{
    if(receive_num == 0)
    {
        if((receiver_uart[0]==0x6E) && (receiver_uart[1]==0x30) && (receiver_uart[2]==0x2E) && (receiver_uart[3]==0x31) && (receiver_uart[4]==0x39))
        {
            k = receiver_uart[3] - 0x30;
            sprintf((uint8_t *)transmit_string, "ok\n");
            HAL_UART_Transmit(&huart1, (uint8_t *)transmit_string, strlen(transmit_string), 50);
            I2C_WRITE(&k,1,1);
        }
    }
    receive_num = 0;
    start_flag = 0;
}
```

增加:
&& (start_flag == 1)

(2) ADC 数字滤波操作 (取平均值)

//中值滤波处理

```
for( AD_Ctrl_Num = 0;AD_Ctrl_Num <= 9; AD_Ctrl_Num++)  
{  
    SUM_AD_R37 += ((float)getADC2());  
}  
SUM_AD_R37 /= 4096;  
SUM_AD_R37 *= 3.3;  
AVE_AD_R37 = SUM_AD_R37/10;  
SUM_AD_R37 = 0;
```

(3) EEPROM 初始化操作

多取几个位置作为标志位，同时成立时进行 I2C 读操作