苏州大学

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

$STM32G431RBT6$

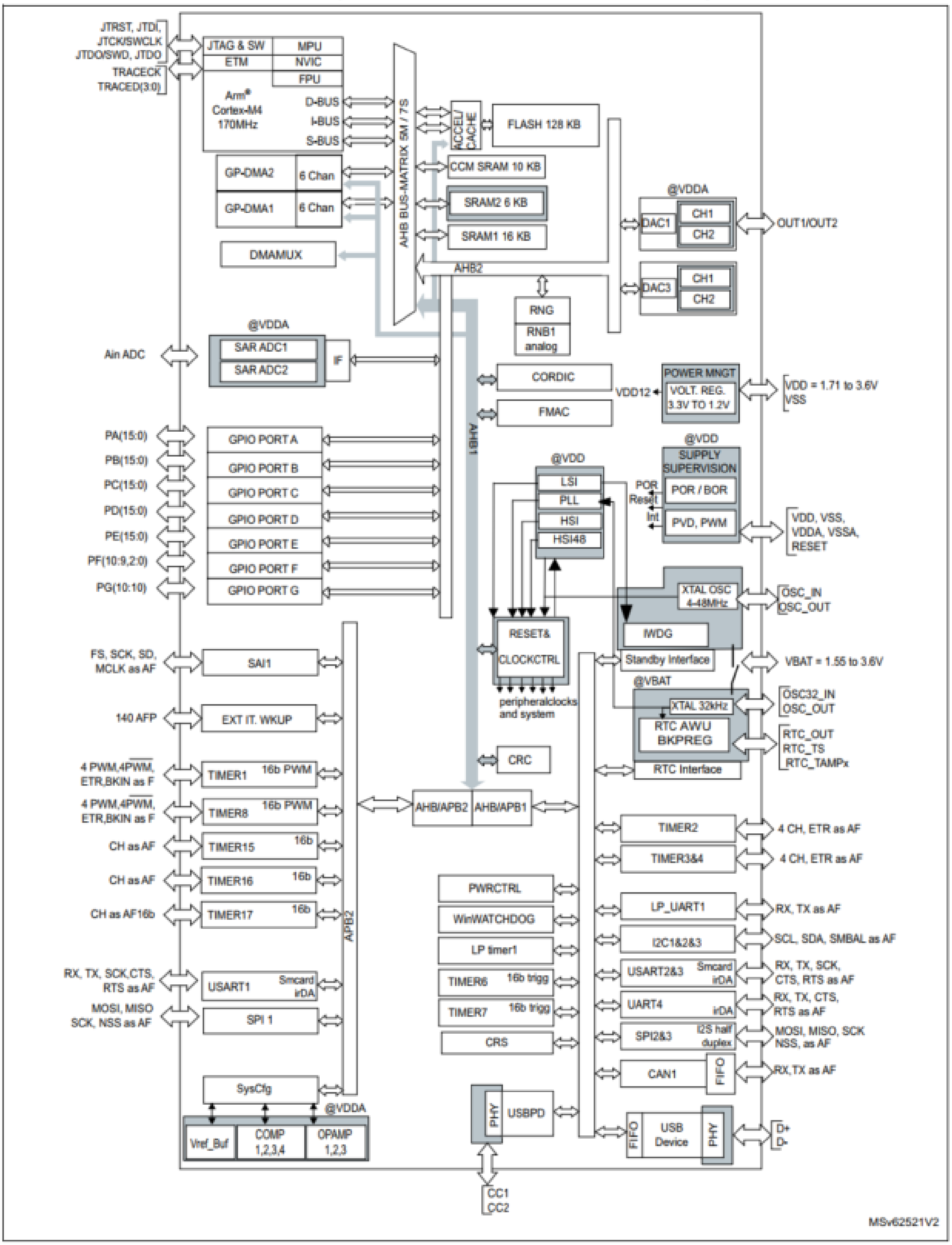
32：32 位 CPU—>4G
G：一种系列
431：特定功能
R：64 引脚
B：闪存大小
T：封装形式
6：适用温度范围

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

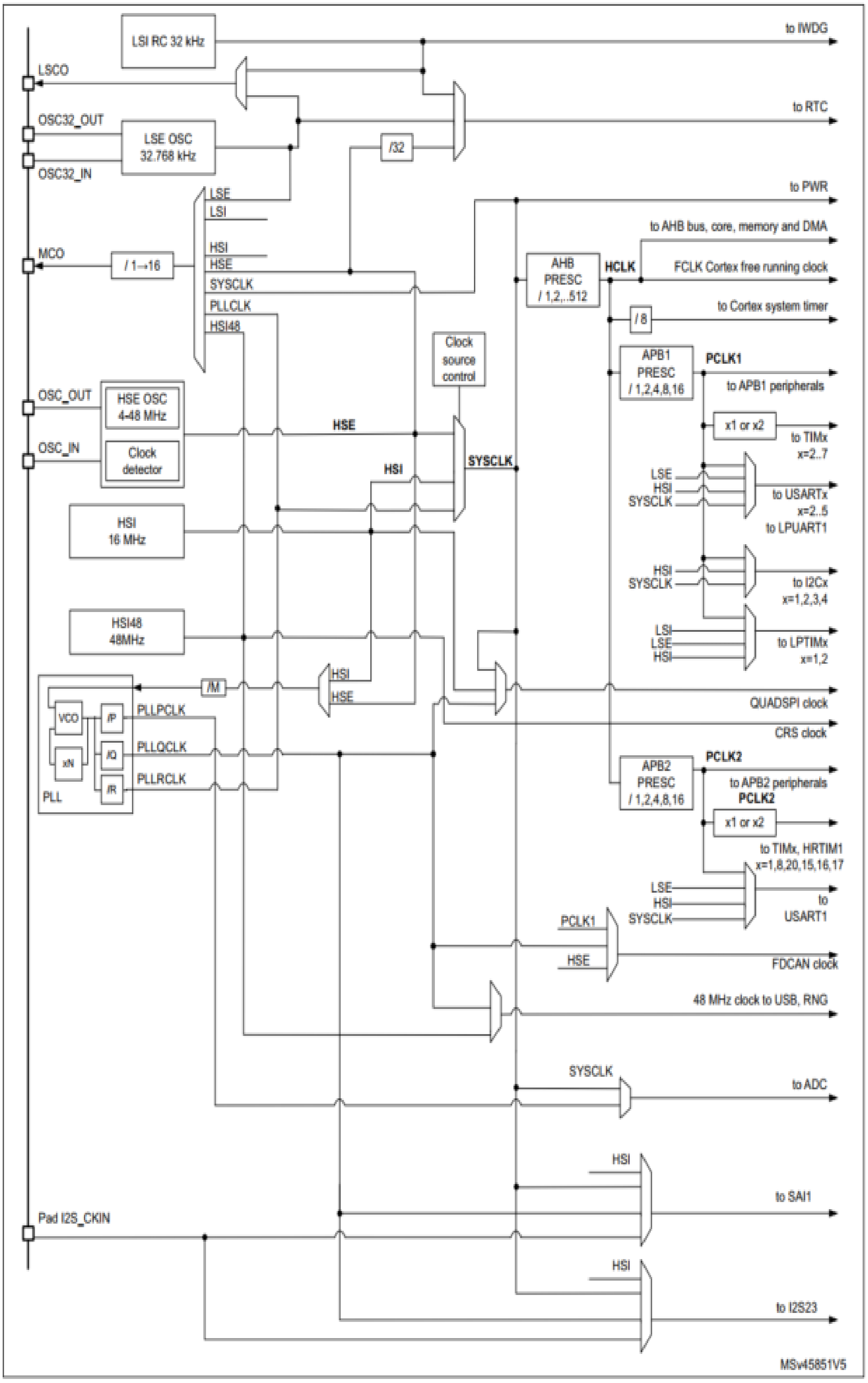
Figure 1. STM32G431x6/x8/xB block diagram



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

5.时钟系统配置

Figure 17. Clock tree

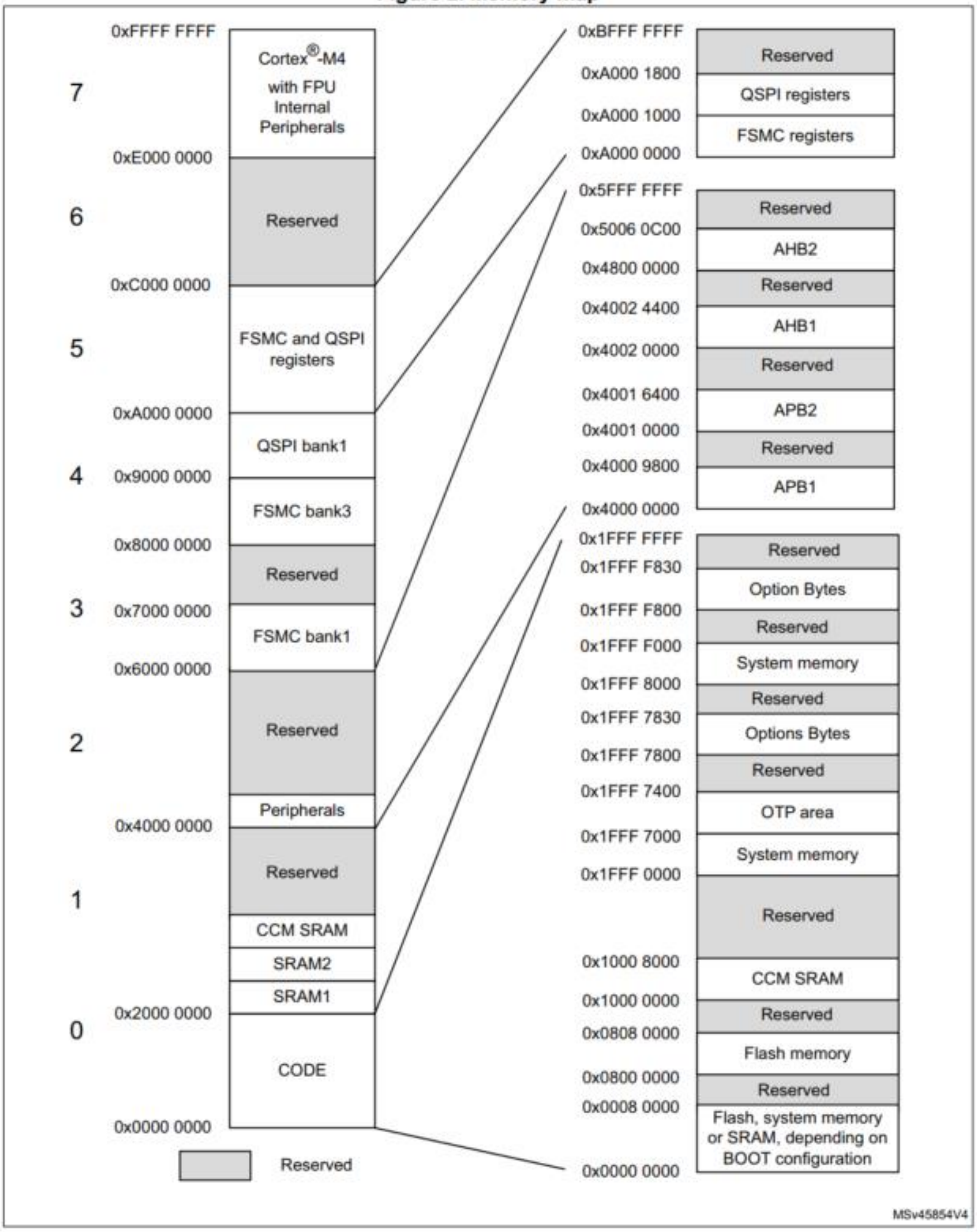


*CUBEMX*配置：

$HSE \rightarrow Clock\ configure \rightarrow 24; HSE; /3; *20; /2 \rightarrow PLLCLK$

6.$STM32$内部资源地址

Figure 2. Memory map



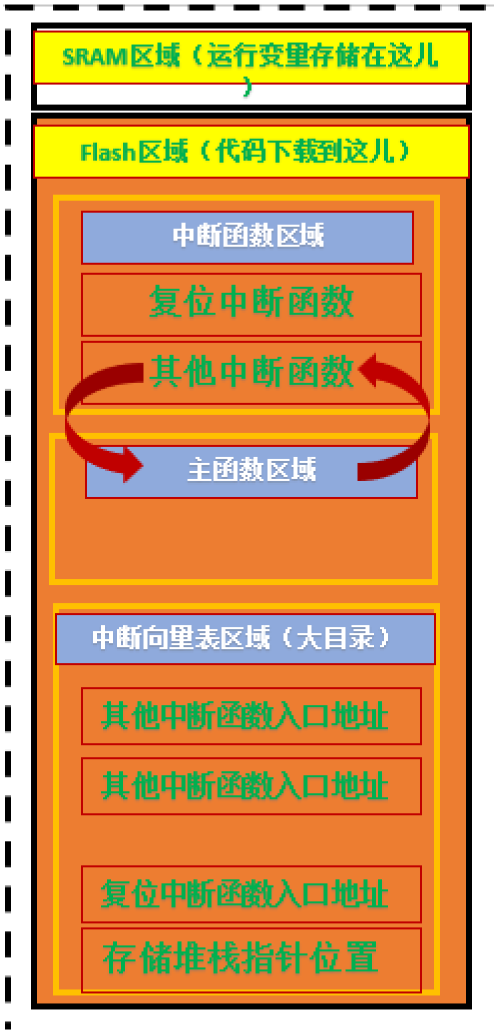
外设 $0x40$

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. $\boldsymbol{GPIO}$

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$PC8\sim PC15$，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_Disp(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

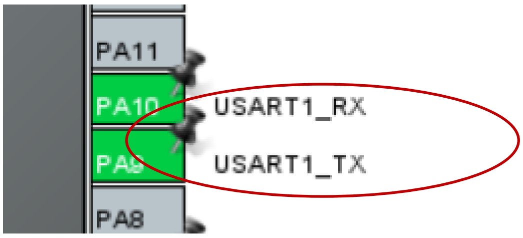
USART1 :　　PA9 : TX

　　　　　　PA10 : RX

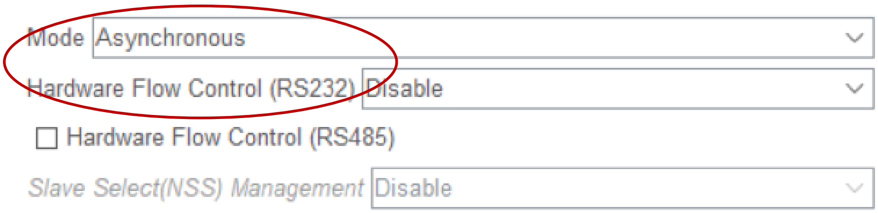
波特率：控制数据发送频率

(3) *CUBEMX*与初始化配置

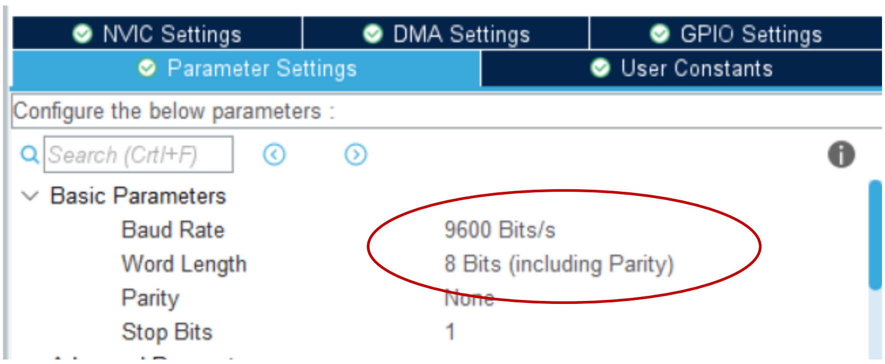
1.



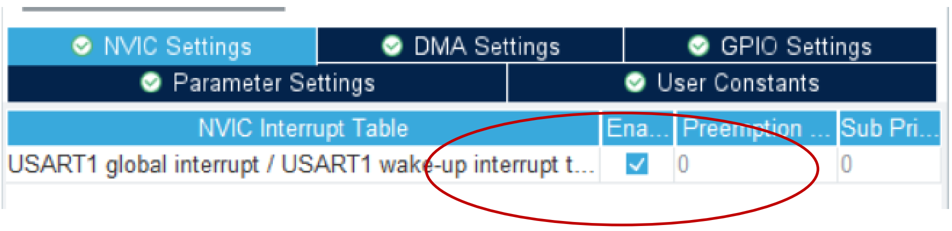
2.



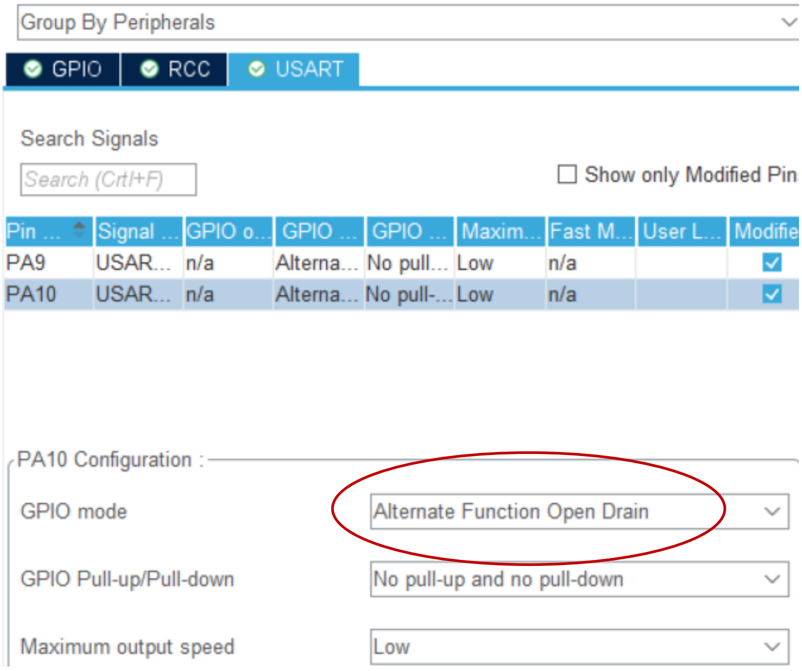
3.



4.

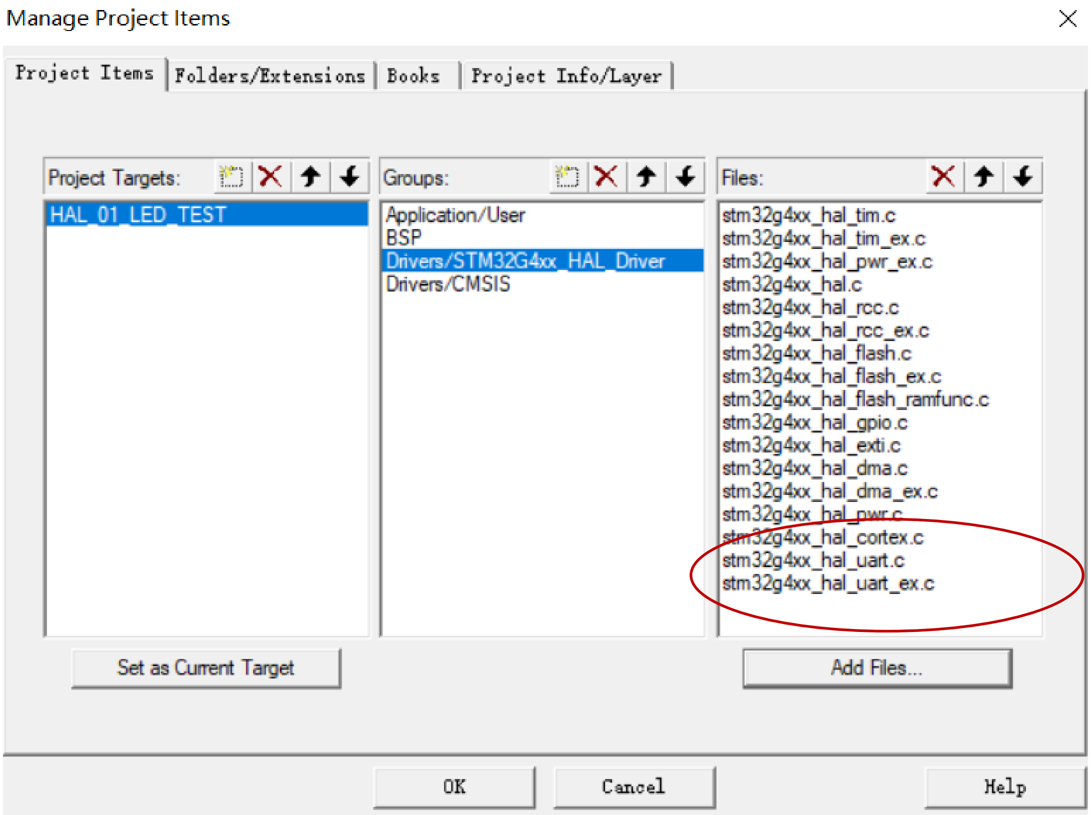


5.



Group By Peripherals
GPIO
RCC
USART
Search Signals
Search (Crtl+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
PA10 USAR... n/a Alterna... No pull-... Low n/a
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.

```
/*#define HAL_I2C_MODULE_ENABLED   */
/*#define HAL_I2S_MODULE_ENABLED   */
/*#define HAL_LPTIM_MODULE_ENABLED   */
/*#define HAL_NAND_MODULE_ENABLED   */
/*#define HAL_NOR_MODULE_ENABLED   */
/*#define HAL_OPAMP_MODULE_ENABLED   */
/*#define HAL_PCD_MODULE_ENABLED   */
/*#define HAL_QSPI_MODULE_ENABLED   */
/*#define HAL_RNG_MODULE_ENABLED   */
/*#define HAL_RTC_MODULE_ENABLED   */
/*#define HAL_SAI_MODULE_ENABLED   */
/*#define HAL_SMARTCARD_MODULE_ENABLED   */
/*#define HAL_SMBUS_MODULE_ENABLED   */
/*#define HAL_SPI_MODULE_ENABLED   */
/*#define HAL_SRAM_MODULE_ENABLED   */
/*#define HAL_TIM_MODULE_ENABLED   */
#define HAL_UART_MODULE_ENABLED
#define HAL_USART_MODULE_ENABLED
/*#define HAL_WWDG_MODULE_ENABLED   */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED
```

8. 修改时钟配置

```
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
{
  Error_Handler();
}
/** Initializes the peripherals clocks
*/
PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART1|RCC_PERIPHCLK_ADC12;
PeriphClkInit.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
```

9. It.c 文件添加中断服务函数

```
  /* USER CODE BEGIN SysTick_IRQn 1 */

  /* USER CODE END SysTick_IRQn 1 */
}



void USART1_IRQHandler(void)
{
  /* USER CODE BEGIN USART1_IRQn 0 */

  /* USER CODE END USART1_IRQn 0 */
  HAL_UART_IRQHandler(&huart1);
  /* USER CODE BEGIN USART1_IRQn 1 */

  /* USER CODE END USART1_IRQn 1 */
}
/******************************************************************************/
/* STM32G4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32g4xx.s).                    */
/******************************************************************************/

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
```

(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_Disp(0XFF);
        HAL_Delay(300);
        LED_Disp(0X00);

    HAL_UART_Receive_IT(&huart1, &rx, 1);

}
```

14. *IIC*

双向两线通信

修改原代码

```
    unsigned short cErrTime = 5;
    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

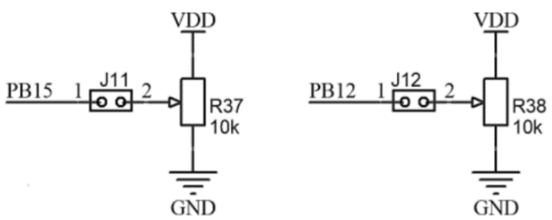
ADC2 PB15

对应两个滑动变阻器

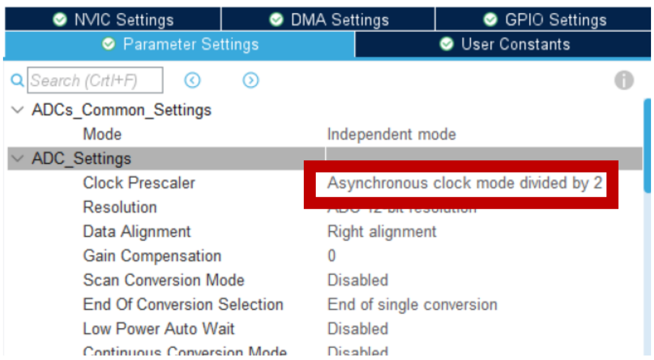
（mcp4017 对应的 ADC1 PB14 不常用）

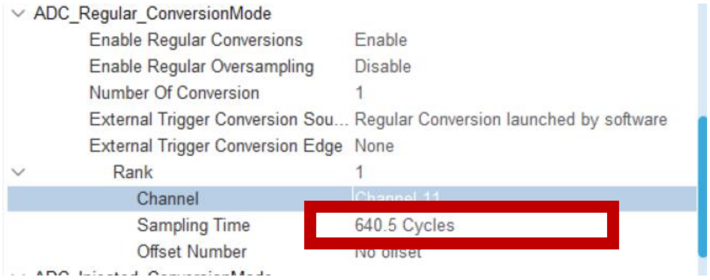
ADC

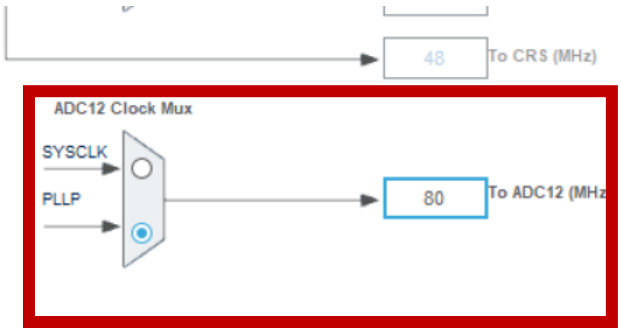
	IN1	IN2	IN3	IN4	IN5	IN6	IN7	IN8	IN9	IN10	IN11	IN12	IN13	IN14	IN15	IN16	IN17
ADC1	PA0	PA1	PA2	PA3	PB14	PC0	PC1	PC2	PC3	PF0	PB12	PB1		PB11	PB0	Tem	Vbat
ADC2	PA0	PA1	PA6	PA7	PC4	PC0	PC1	PC2	PC3	PF1	PC5	PB2	PA5	PB11	PB15		PA4



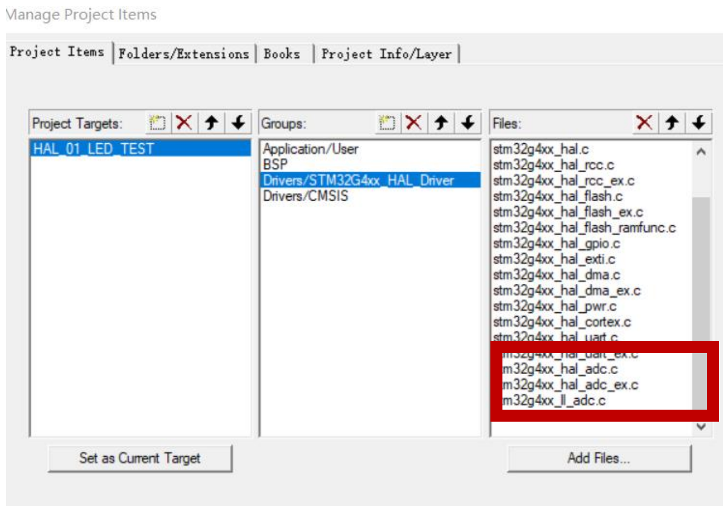
(2) CUBEMX 配置

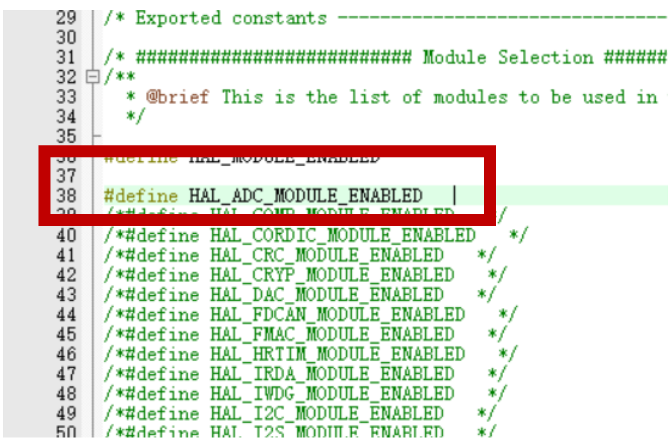



ADC_Regular_ConversionMode
Enable Regular Conversions Enable
Enable Regular Oversampling Disable
Number Of Conversion 1
External Trigger Conversion Sou... Regular Conversion launched by software
External Trigger Conversion Edge None
Rank 1
Channel
Sampling Time 640.5 Cycles
Offset Number


48
To CRS (MHz)
ADC12 Clock Mux
SYSCLK
PLLP
80
To ADC12 (MHz

(3) 初始化代码


Manage Project Items
Project Items
Folders/Extensions
Books
Project Info/Layer
Project Targets:
Groups:
Files:
HAL_01_LED_TEST
Application/User
BSP
Drivers/STM32G4xx_HAL_Driver
Drivers/CMSIS
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
m32g4xx_hal_adc.c
m32g4xx_hal_adc_ex.c
m32g4xx_ll_adc.c
Set as Current Target
Add Files...


/* Exported constants --------------------------------
/* ########################## Module Selection ######
/**
* @brief This is the list of modules to be used in
*/
#define HAL_ADC_MODULE_ENABLED
/*#define HAL_CORDIC_MODULE_ENABLED */
/*#define HAL_CRC_MODULE_ENABLED */
/*#define HAL_CRYP_MODULE_ENABLED */
/*#define HAL_DAC_MODULE_ENABLED */
/*#define HAL_FDCAN_MODULE_ENABLED */
/*#define HAL_FMAC_MODULE_ENABLED */
/*#define HAL_HRTIM_MODULE_ENABLED */
/*#define HAL_IRDA_MODULE_ENABLED */
/*#define HAL_IWDG_MODULE_ENABLED */
/*#define HAL_I2C_MODULE_ENABLED */

```c
#include "main.h"

extern ADC_HandleTypeDef hadc1;

/* USER CODE BEGIN Private defines */

/* USER CODE END Private defines */

void ADC1_Init(void);
```

```c
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInit = {0}

  /** Configure the main internal regulator output voltage
  */
  HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1)
  /** Initializes the RCC Oscillators according to the specifie
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
  RCC_OscInitStruct.HSEState = RCC_HSE_ON;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
  RCC_OscInitStruct.PLL.PLLM = RCC_PLLM_DIV3;
  RCC_OscInitStruct.PLL.PLLN = 20;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
  RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
```

```
{
  Error_Handler();
}
/** Initializes the CPU, AHB and APB buses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                            |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
{
  Error_Handler();
}

 /** Initializes the peripherals clocks
 */
PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_USART1|RCC_PERIPHCLK_ADC12;
PeriphClkInit.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
PeriphClkInit.Adc12ClockSelection = RCC_ADC12CLKSOURCE_SYSCLK;
if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
{
  Error_Handler();
}
}
```

（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）

重装载计数值不一定是 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

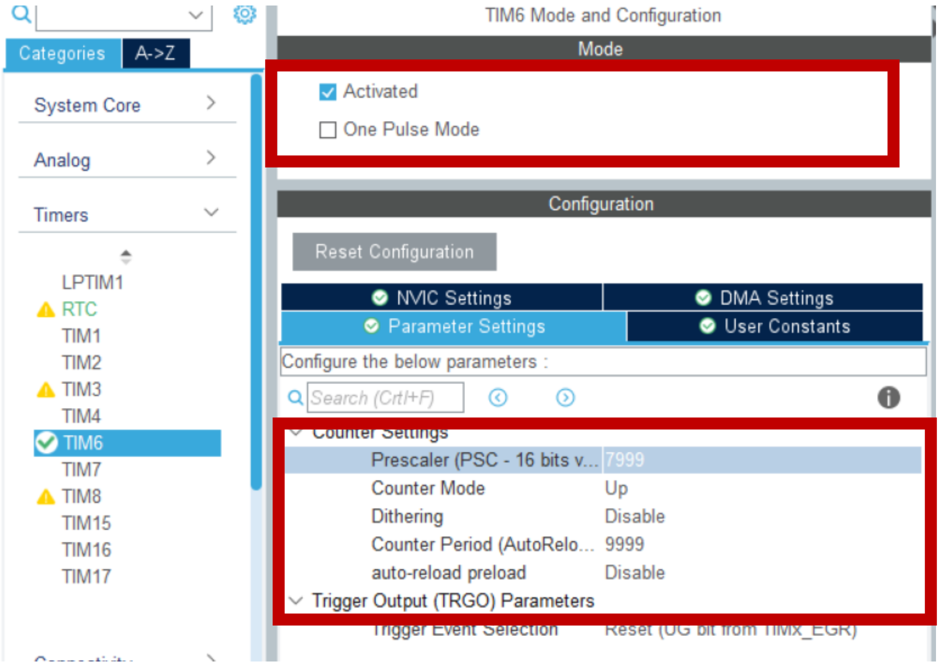
分频越高，计数越快

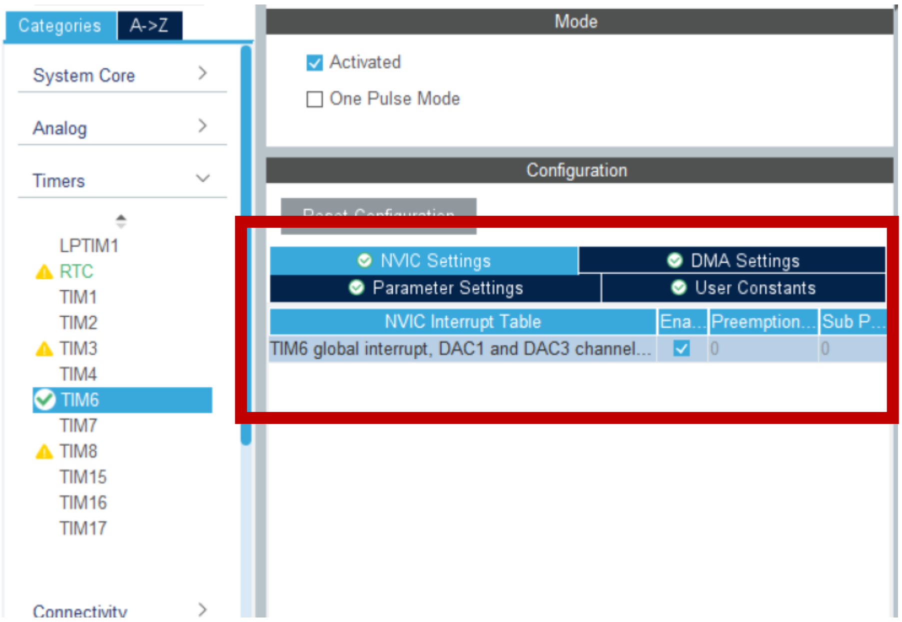
（2） 基本定时器：

0-65536， up， 时钟来源$APB1$

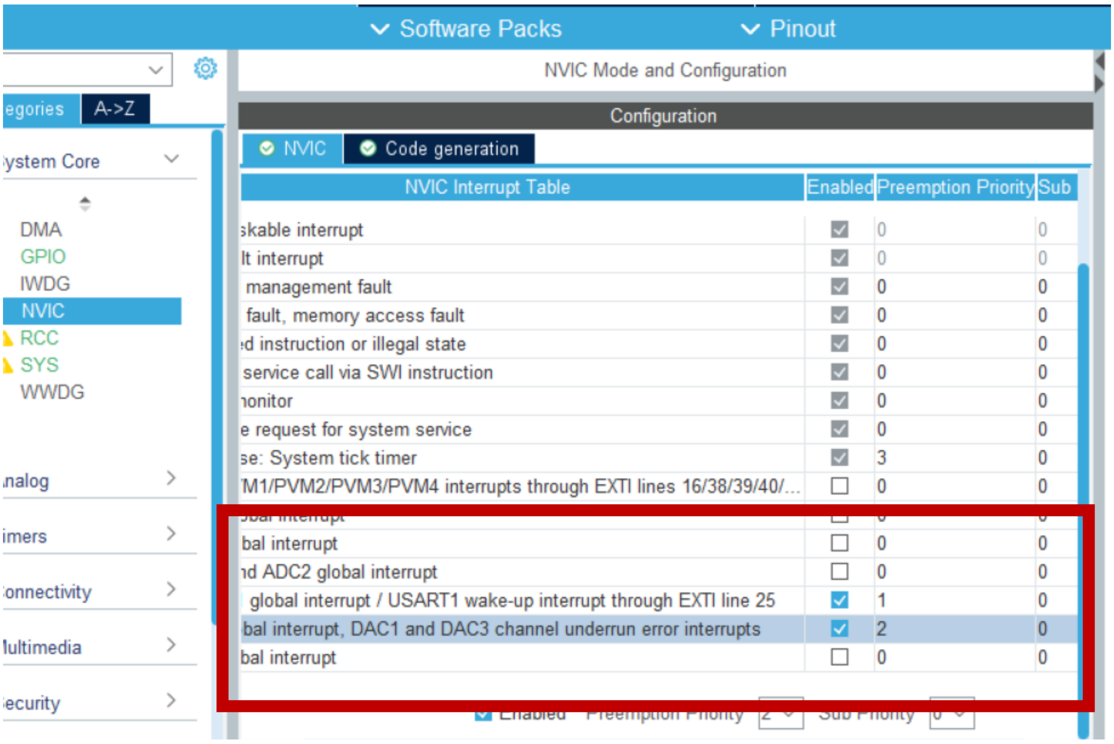
1. $CUBEMX$配置

（1s 产生一次中断 80M=8000 * 10000）



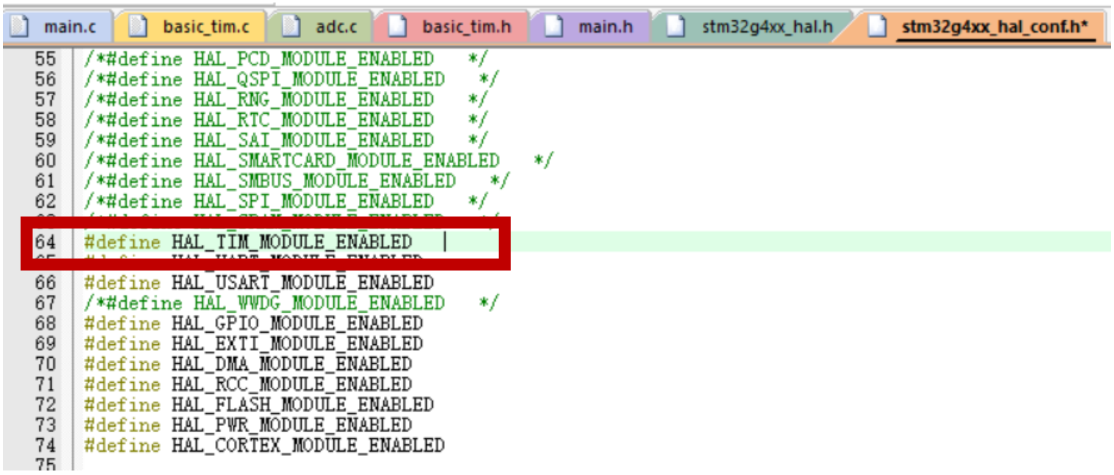


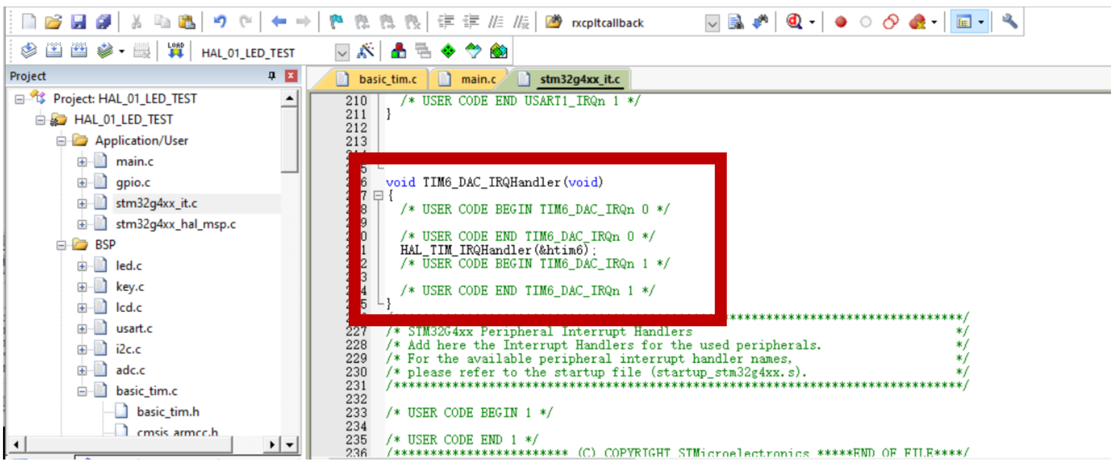
（基本定时器优先级设置比串口低一些）



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：

```c
  }
  /* TIM Update event */
  if (__HAL_TIM_GET_FLAG(htim, TIM_FLAG_UPDATE) != RESET)
  {
    if (__HAL_TIM_GET_IT_SOURCE(htim, TIM_IT_UPDATE) != RESET)
    {
      __HAL_TIM_CLEAR_IT(htim, TIM_IT_UPDATE);
#if (USE_HAL_TIM_REGISTER_CALLBACKS == 1)
      htim->PeriodElapsedCallback(htim);
#else
      HAL_TIM_PeriodElapsedCallback(htim);
#endif /* USE_HAL_TIM_REGISTER_CALLBACKS */
    }
  }
  /* TIM Break input event */
  if (__HAL_TIM_GET_FLAG(htim, TIM_FLAG_BREAK) != RESET)
  {
    if (__HAL_TIM_GET_IT_SOURCE(htim, TIM_IT_BREAK) != RESET)
    {
      __HAL_TIM_CLEAR_IT(htim, TIM_IT_BREAK);
#if (USE_HAL_TIM_REGISTER_CALLBACKS == 1)
      htim->BreakCallback(htim);
#else
```

（主函数打开中断）

```c
  mcp4017_write(56);

  //中断打开处

  HAL_UART_Receive_IT(&huart1, &rx, 1);

  HAL_TIM_Base_Start_IT(&htim6);
```

回调函数修改如下：回调函数结束前需要再次打开中断

```c
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

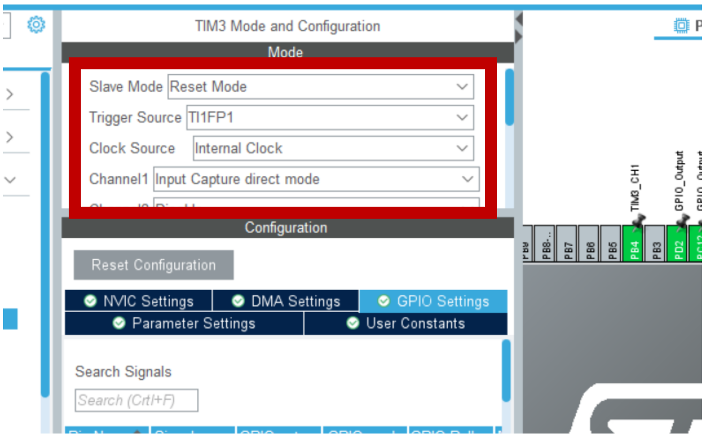
c) 强制输出电平

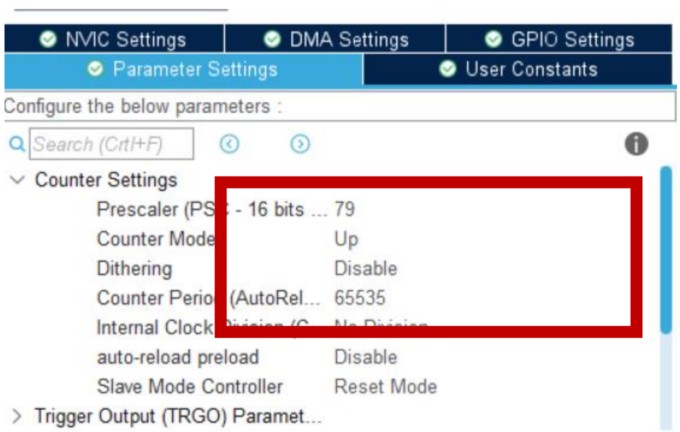
d) PWM 模式：（PWM1/PWM2）

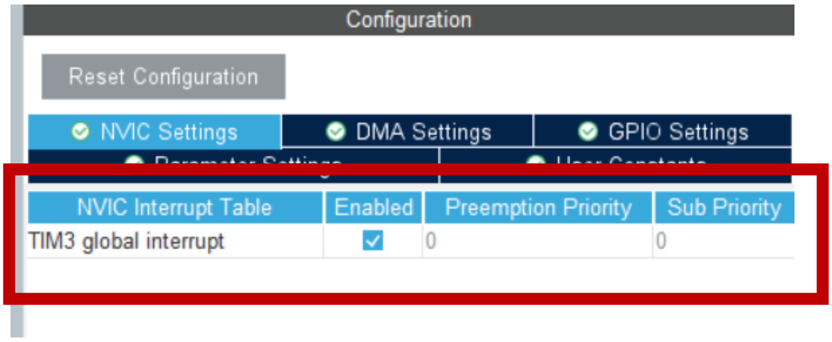
两者互补

1. *CUBEMX*配置

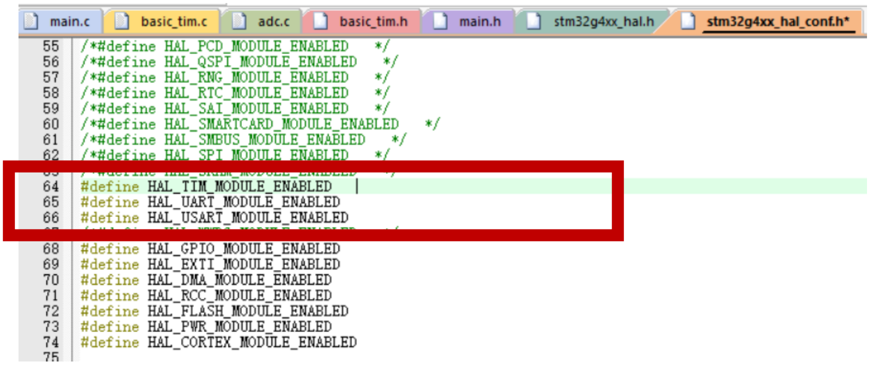
PA15 PB4

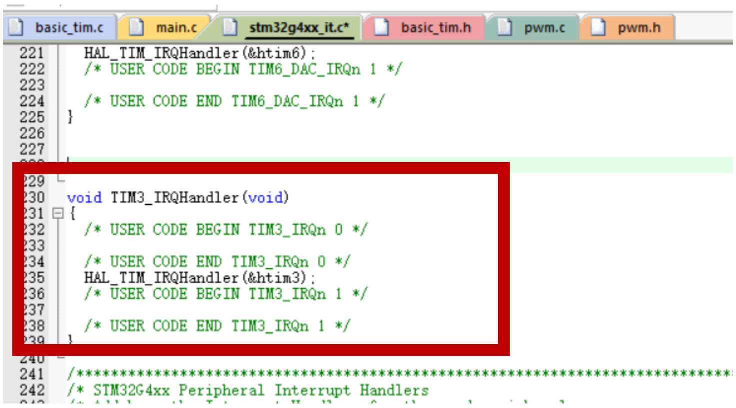






2. 代码初始化配置





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);
```

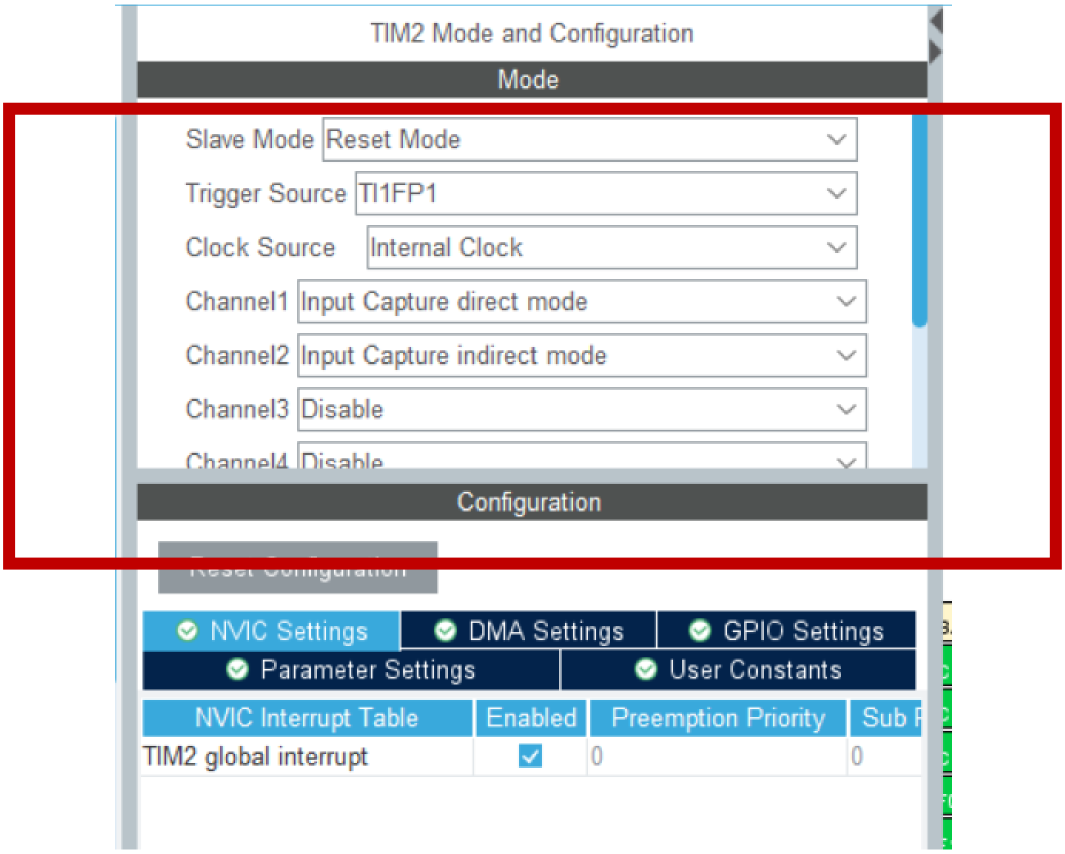
修改此处回调函数:

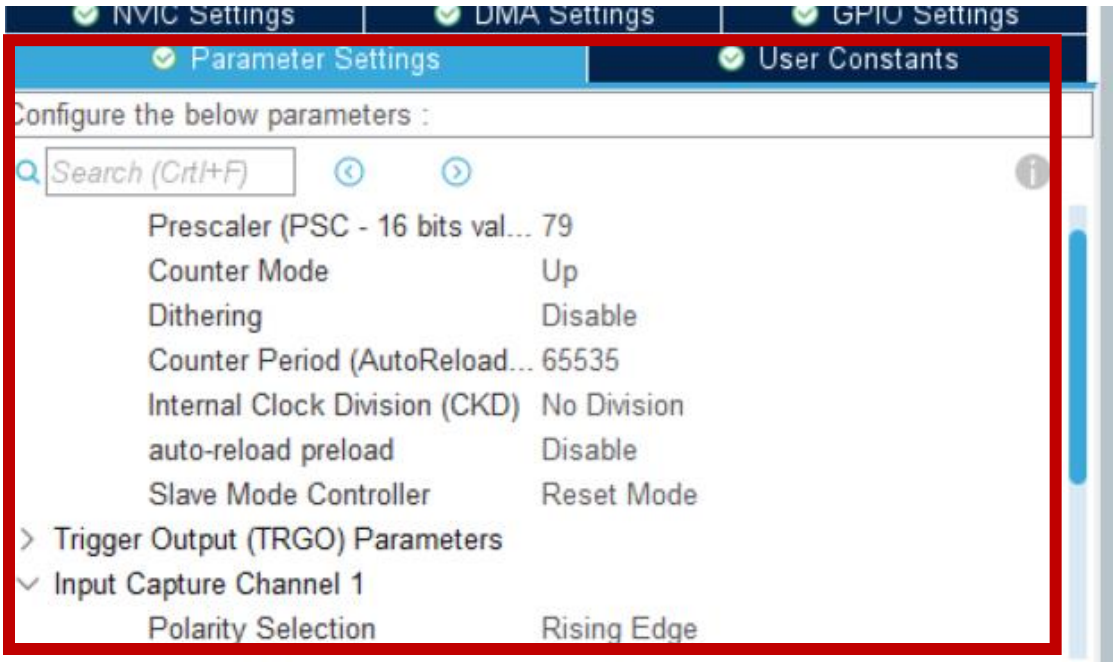
```c
*/
__weak void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
  /* Prevent unused argument(s) compilation warning */
  UNUSED(htim);

  /* NOTE : This function should not be modified, when the callback is needed,
            the HAL_TIM_IC_CaptureCallback could be implemented in the user file
   */
}

/**
```

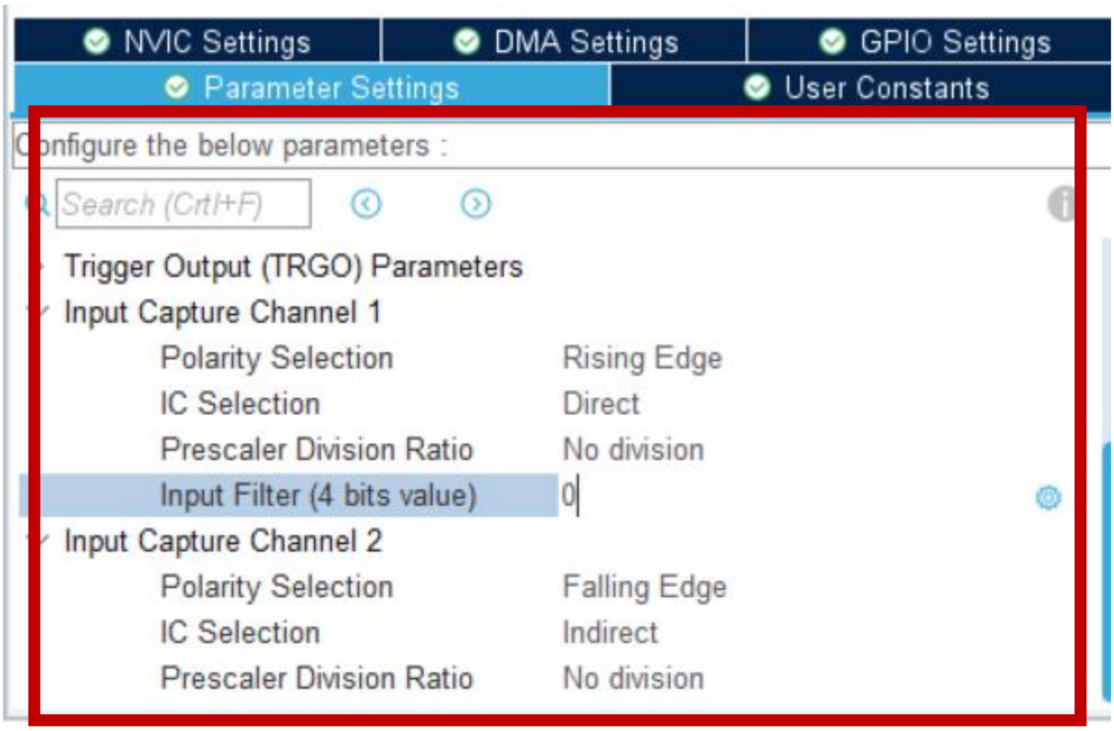
修改如下:

```c
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
Configuration
NVIC Settings
DMA Settings
GPIO Settings
Parameter Settings
User Constants
NVIC Interrupt Table
Enabled
Preemption Priority
TIM2 global interrupt
0
0


Parameter Settings
User Constants
Search (Crtl+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 ;
  }


}
```

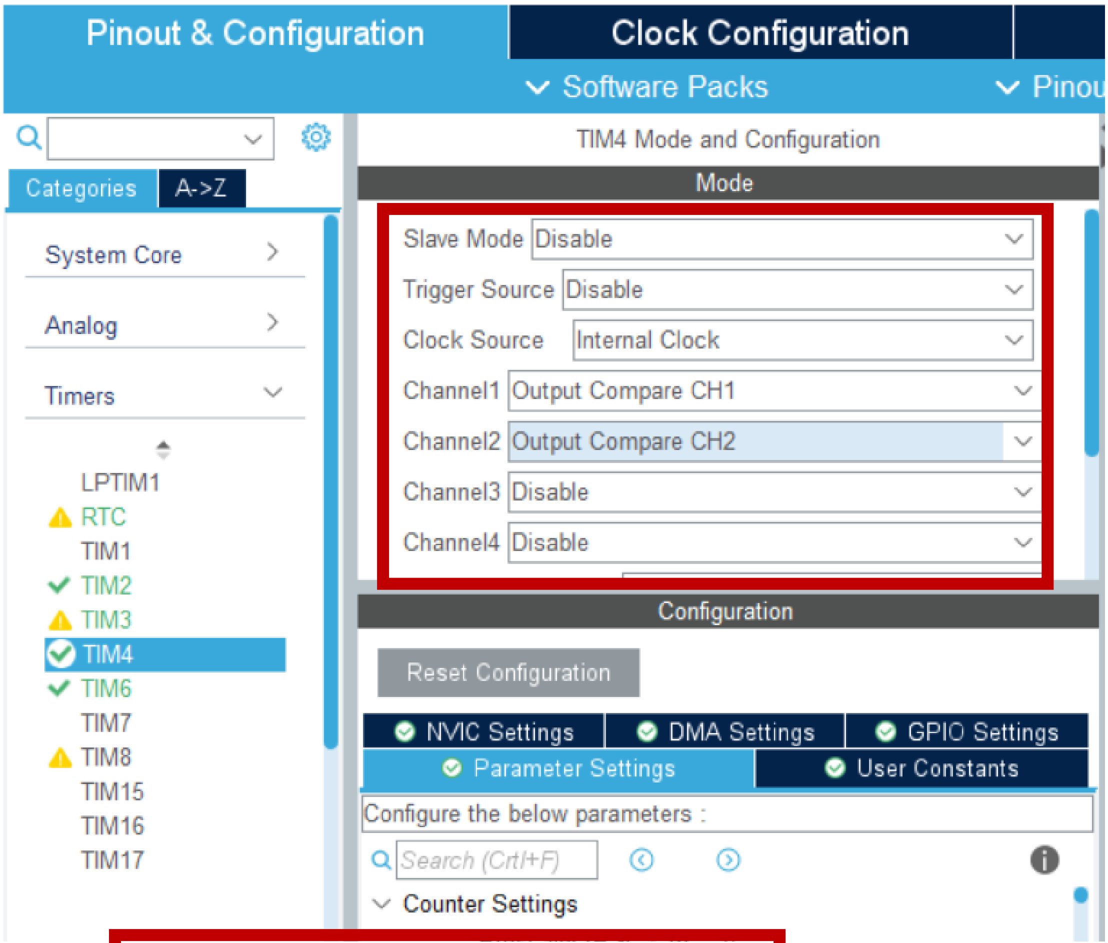
（4） 高级定时器（输出比较）

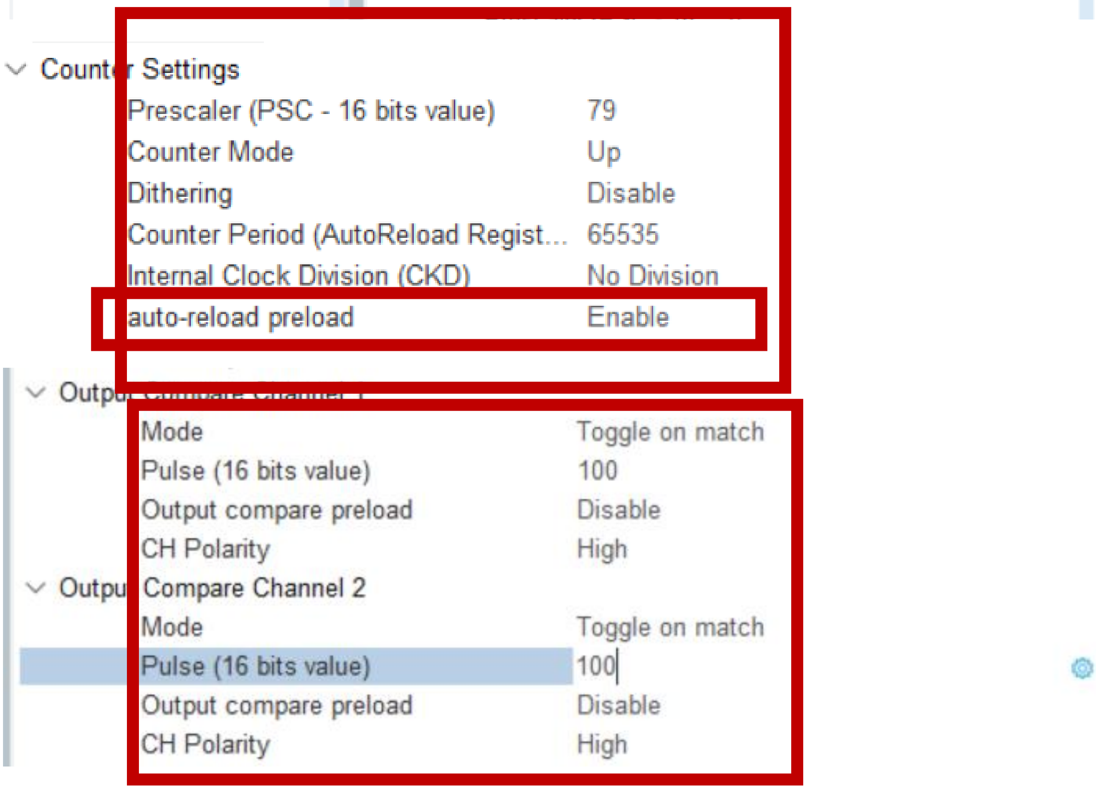
方波 & pwm 波

方波：占空比 0.5 的 PWM 波

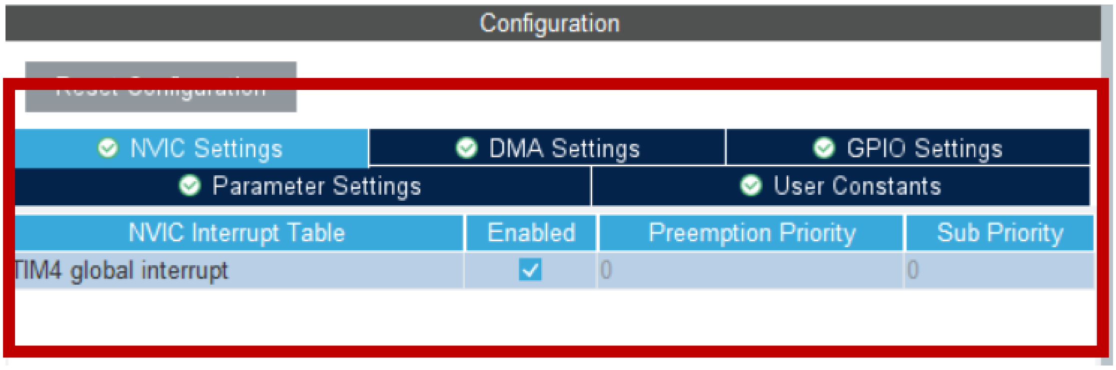
（方波）->利用电平翻转模式

1. *CUBEMX*配置


Pinout & Configuration
Clock Configuration
Software Packs
Pinou
Categories
A->Z
System Core
Analog
Timers
LPTIM1
RTC
TIM1
TIM2
TIM3
TIM4
TIM6
TIM7
TIM8
TIM15
TIM16
TIM17
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 (Crtl+F)
Counter Settings


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
Mode Toggle on match
Pulse (16 bits value) 100
Output compare preload Disable
CH Polarity High
Compare Channel 2
Mode Toggle on match
Pulse (16 bits value) 100
Output compare preload Disable
CH Polarity High

、



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_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF10_TIM4;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

  /* USER CODE BEGIN TIM4_MspPostInit 1 */

  /* USER CODE END TIM4_MspPostInit 1 */
  }

}
```

```

void TIM4_IRQHandler(void)
{
  /* USER CODE BEGIN TIM4_IRQn 0 */

  /* USER CODE END TIM4_IRQn 0 */
  HAL_TIM_IRQHandler(&htim4);
  /* USER CODE BEGIN TIM4_IRQn 1 */

  /* USER CODE END TIM4_IRQn 1 */
}

/******************************************
/* STM32G4xx Peripheral Interrupt Handler
/* Add here the Interrupt Handlers for th
```

3. 使用示例

（两路频率不等方波）

```
HAL_TIM_IC_Start_IT(&htim2, TIM_CHANNEL_1);



HAL_TIM_OC_Start_IT(&htim4 , TIM_CHANNEL_1);
HAL_TIM_OC_Start_IT(&htim4 , TIM_CHANNEL_1);


```

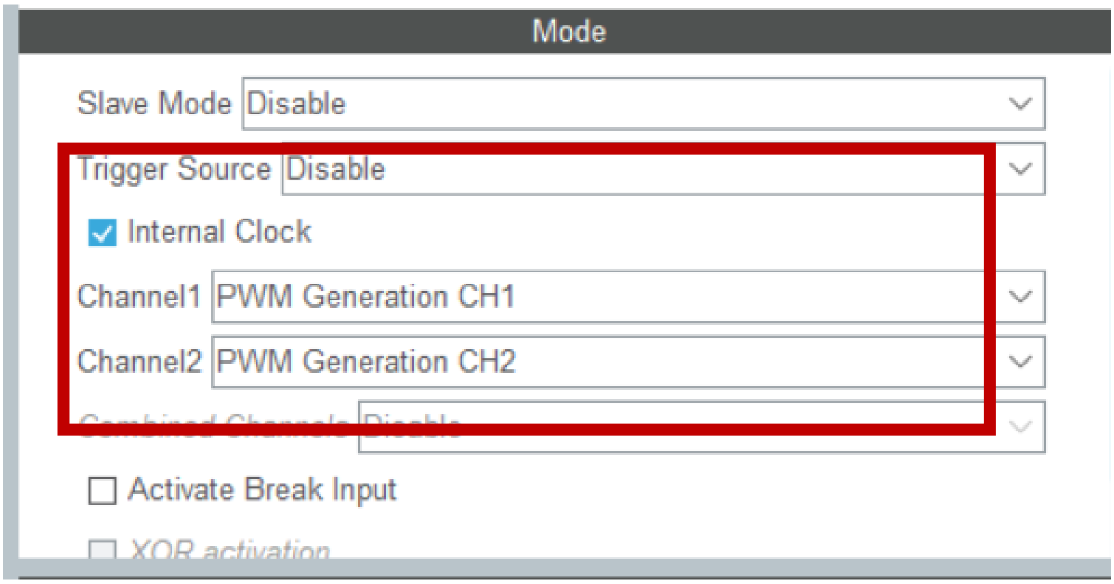
（修改此处回调函数）

```
  * @param  htim TIM OC handle
  * @retval None
  */
__weak void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim)
{
  /* Prevent unused argument(s) compilation warning */
  UNUSED(htim);

  /* NOTE : This function should not be modified, when the callback is r
            the HAL_TIM_OC_DelayElapsedCallback could be implemented in
   */
}

/**
  * @brief  Input Capture callback in non-blocking mode
  * @param  htim TIM IC handle
  * @retval None
  */
```

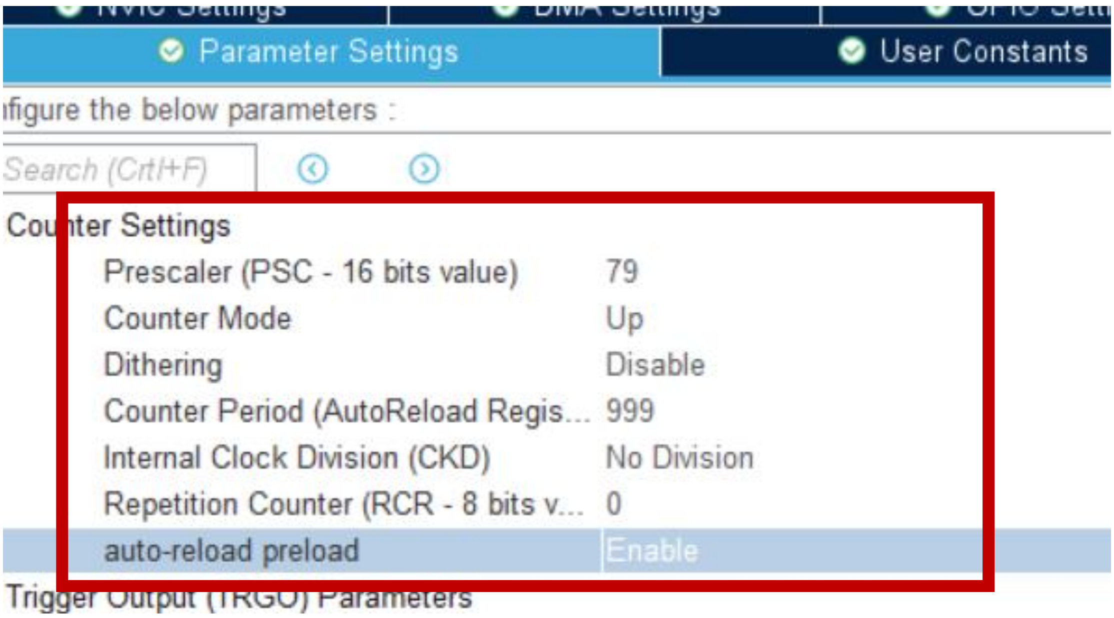
```
void HAL_TIM_OC_DelayElapsedCallback(TIM_HandleTypeDef *htim)
{
 //方波输出中断

 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



（1K Hz 为例）



占空比分别为 0.3 和 0.6

PWM Generation Channel 1	
Mode	PWM mode 1
Pulse (16 bits value)	300
Output compare preload	Enable
Fast Mode	Disable
CH Polarity	High
CH Idle State	Reset
PWM Generation Channel 2	
Mode	PWM mode 1
Pulse (16 bits value)	600
Output compare preload	Enable
Fast Mode	Disable
CH Polarity	High
CH Idle State	Reset

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);
```

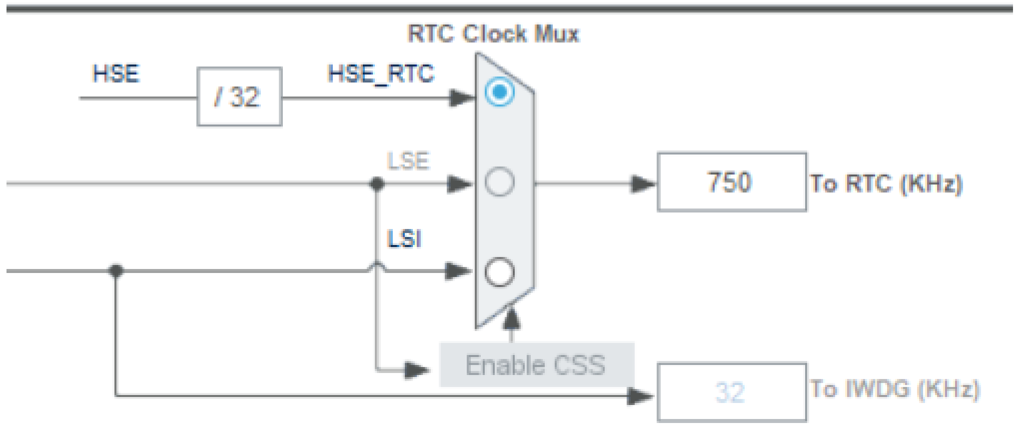
4. 修改占空比

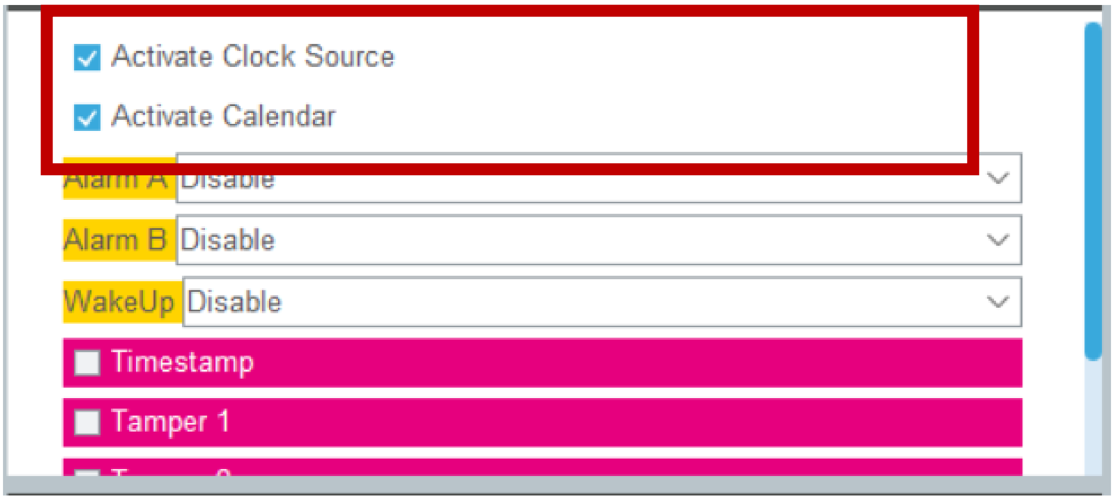
```
HAL_TIM_PWM_Start(&htim16, TIM_CHANNEL_1);
HAL_TIM_PWM_Start(&htim17, TIM_CHANNEL_1);

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

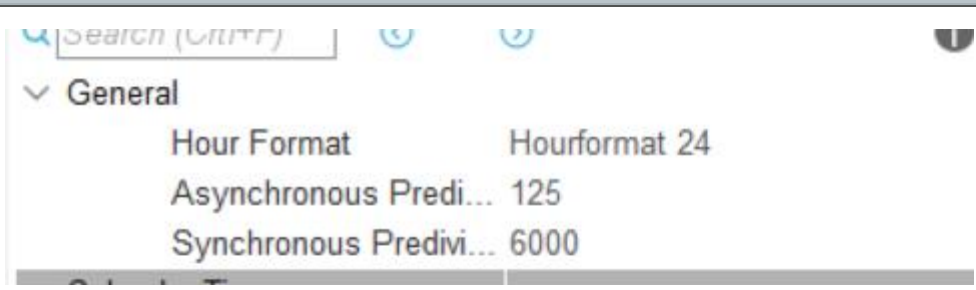
17. *RTC*

（1） *CUBEMX*配置

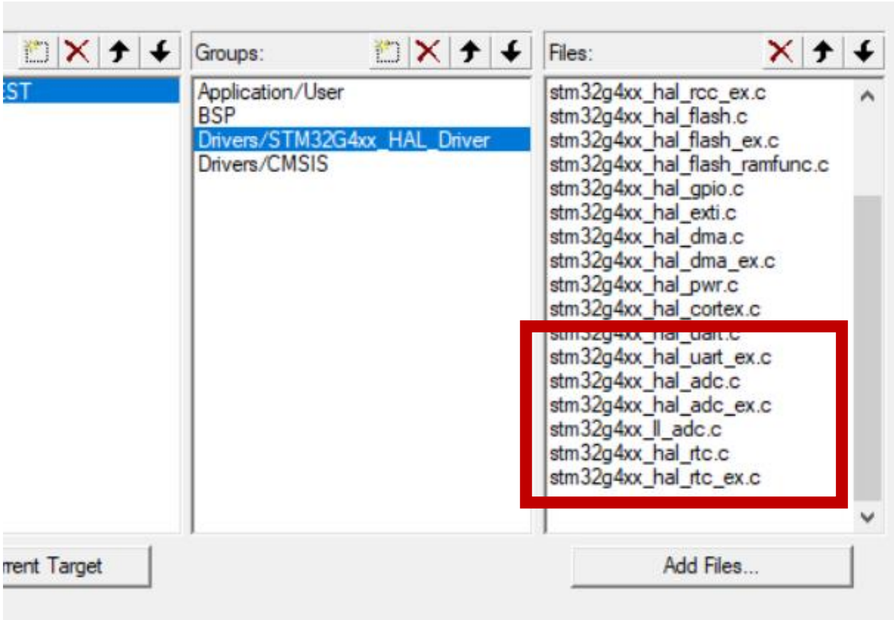
BCD 码格式

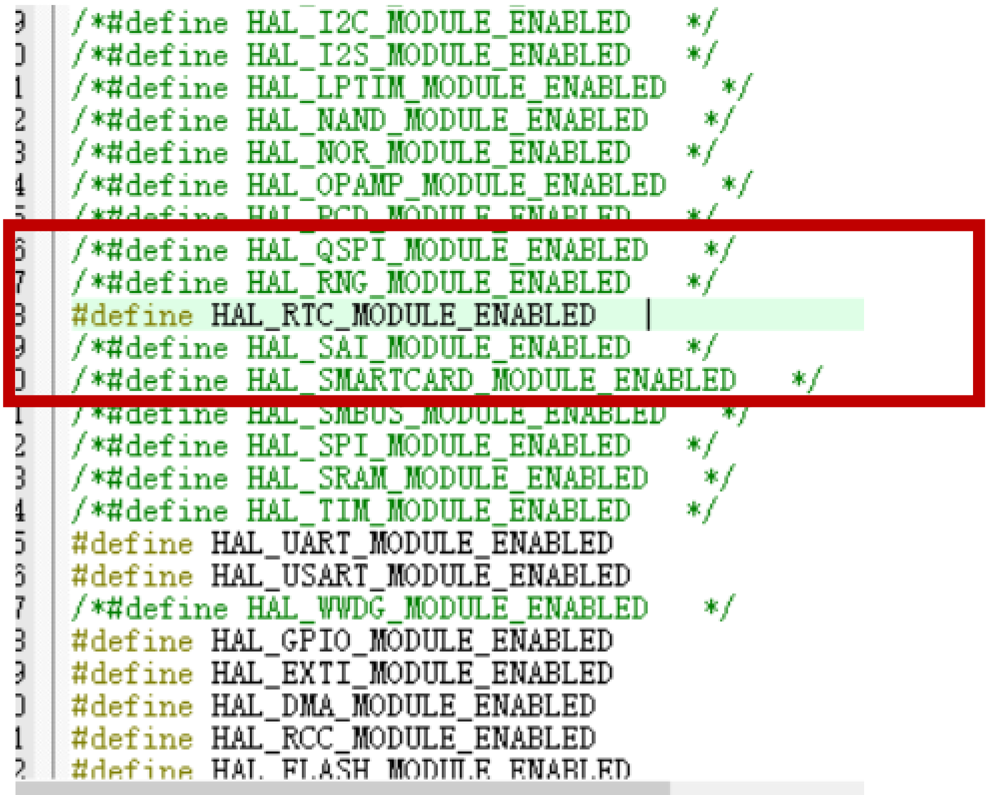

RTC Clock Mux
HSE
/ 32
HSE_RTC
LSE
LSI
750
To RTC (KHz)
Enable CSS
32
To IWDG (KHz)


Activate Clock Source
Activate Calendar
Alarm B Disable
WakeUp Disable
Timestamp
Tamper 1


General
Hour Format Hourformat 24
Asynchronous Predi... 125
Synchronous Predivi... 6000

（2） 代码初始化配置


Groups:
Files:
Application/User
BSP
Drivers/STM32G4xx_HAL_Driver
Drivers/CMSIS
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_ex.c
stm32g4xx_hal_adc.c
stm32g4xx_hal_adc_ex.c
stm32g4xx_ll_adc.c
stm32g4xx_hal_rtc.c
stm32g4xx_hal_rtc_ex.c
Add Files...


/*#define HAL_I2C_MODULE_ENABLED */
/*#define HAL_I2S_MODULE_ENABLED */
/*#define HAL_LPTIM_MODULE_ENABLED */
/*#define HAL_NAND_MODULE_ENABLED */
/*#define HAL_NOR_MODULE_ENABLED */
/*#define HAL_OPAMP_MODULE_ENABLED */
/*#define HAL_QSPI_MODULE_ENABLED */
/*#define HAL_RNG_MODULE_ENABLED */
#define HAL_RTC_MODULE_ENABLED
/*#define HAL_SAI_MODULE_ENABLED */
/*#define HAL_SMARTCARD_MODULE_ENABLED */
/*#define HAL_SPI_MODULE_ENABLED */
/*#define HAL_SRAM_MODULE_ENABLED */
/*#define HAL_TIM_MODULE_ENABLED */
#define HAL_UART_MODULE_ENABLED
#define HAL_USART_MODULE_ENABLED
/*#define HAL_WWDG_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED

```
  /** Initializes the peripherals clocks
  */
  PeriphClkInit.PeriphClockSelection = RCC_PERIPHCLK_RTC|RCC_PERIPHCLK_USART1
                              |RCC_PERIPHCLK_ADC12;
  PeriphClkInit.Usart1ClockSelection = RCC_USART1CLKSOURCE_PCLK2;
  PeriphClkInit.Adc12ClockSelection = RCC_ADC12CLKSOURCE_SYSCLK;
  PeriphClkInit.RTCClockSelection = RCC_RTCCLKSOURCE_HSE_DIV32;
  if (HAL_RCCEx_PeriphCLKConfig(&PeriphClkInit) != HAL_OK)
  {
    Error_Handler();
  }
}
```

（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 *)lcd_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 *)lcd_disp_string);

sprintf((char *)lcd_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 *)lcd_disp_string);
```

18. 相关知识

变量类型定义

```
#ifndef __int8_t_defined
# define __int8_t_defined
typedef signed char             int8_t;
typedef short int               int16_t;
typedef int                     int32_t;
# if __WORDSIZE == 64
typedef long int                int64_t;
# else
__extension__
typedef long long int           int64_t;
# endif
#endif


typedef unsigned char           uint8_t;
typedef unsigned short int      uint16_t;
#ifndef __uint32_t_defined
typedef unsigned int            uint32_t;
# define __uint32_t_defined
#endif
#if __WORDSIZE == 64
typedef unsigned long int       uint64_t;
#else
__extension__
typedef unsigned long long int  uint64_t;
#endif
```