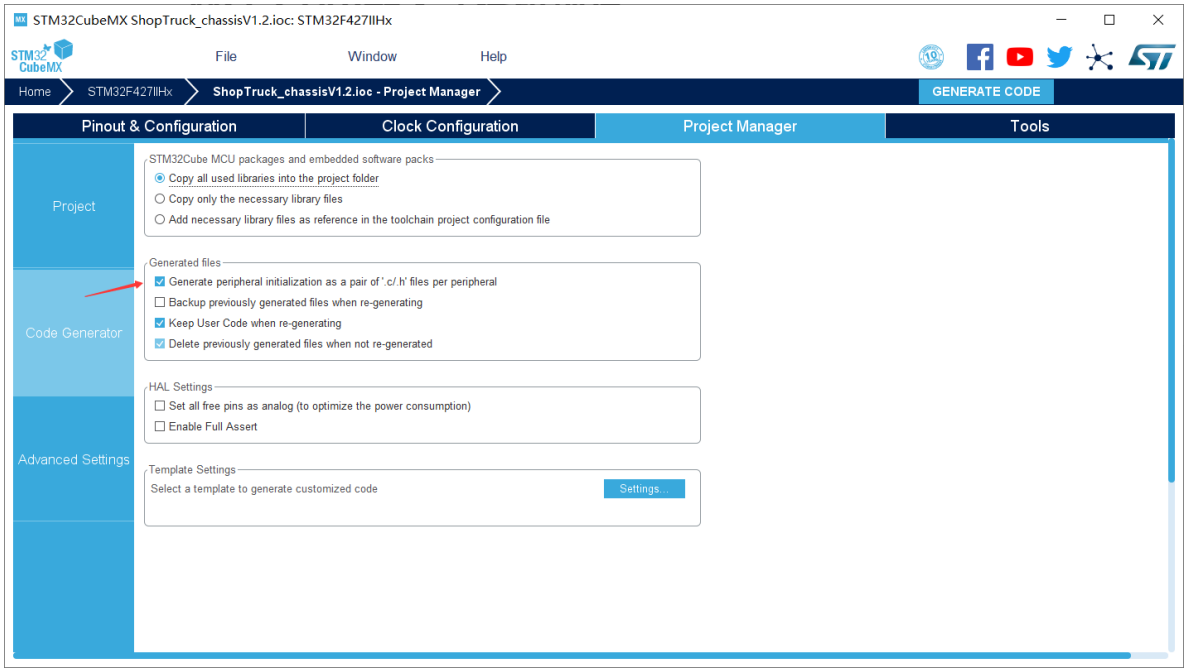
嵌入式代码规范

工程结构

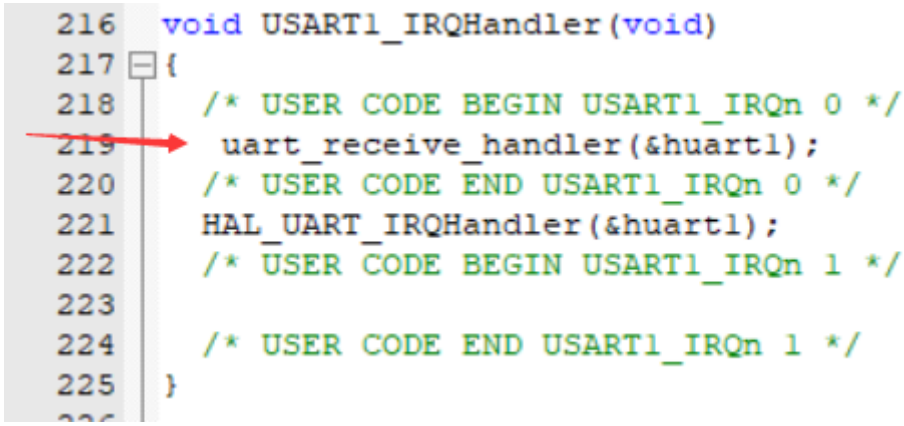
基础CubeMX生成的工程

- 首先，生成工程时，勾选“对每个外设分别生成初始化文件”



- 在工程中，“main.c”等文件默认存放在“./MDK-ARM/”目录下
- BSP文件夹和操作系统线程（任务）文件夹放在根目录下
    - 所有的工程建议BSP文件夹下放置配置相关文件，如bsp_can.h,bsp_can.c为CAN通信的配置文件
    - 所有的车以FreeRTOS实时操作系统为主要框架，所以所有**有关功能具体实现**的代码放在Task文件夹下
      1.在freertos.c下创建任务句柄，创建任务绑定任务实现函数
      2.在Task文件下创建myTask.c用于实现任务实现函数
      所有的任务都参照这样的模式，使得代码阅读便捷，开发简单
- 根目录下，添加“REFERENCE”文件夹，存放相关参考资料以及**必须的代码说明**
    - 修改说明：修改了CubeMX生成的基础文件内容必须进行说明

      例：官方遥控器在stm32F4xx_it.c文件中进行了修改



```
216  void USART1_IRQHandler(void)
217  {
218    /* USER CODE BEGIN USART1_IRQn 0 */
219     uart_receive_handler(&huart1);
220    /* USER CODE END USART1_IRQn 0 */
221    HAL_UART_IRQHandler(&huart1);
222    /* USER CODE BEGIN USART1_IRQn 1 */
223
224    /* USER CODE END USART1_IRQn 1 */
225  }
```

    - 问题说明：代码调试过程中需记录遇到的问题以及解决方案
- 最后，给出示例如下

```
--+ Project
    --+ Drivers
    --+ inc
    --+ MDK-ARM
    --+ Middlewares
    --+ Src
    --+ Task
    --+ BSP
    --+ REFERENCE
        --+ link.md
        --+ ...
    --+ Project.ioc
    --+ README.md
```

函数注释

- 声明函数时，需要对函数的功能、参数、返回值进行说明，并且注上作者和联系方式
    - 参数说明：参数名（参数类型）

```
/**
    * @brief  初始化W25Q256芯片
    * @note   调用该函数前必须已经初始化了QSPI外设，FLASH芯片上电后默认SPI传输模式
    * @author 江榕煜（V1），周森（V2）
    * @param  None
    * @retval 初始化是否成功，HAL_OK或者HAL_ERROR
    */
HAL_StatusTypeDef FLASH256devInit(void);
```

- 函数编写时，写清逻辑关系

```
if (rc_device_get_state(&rc, RC_S2_UP))             //若sw2为UP，底盘跟随云台，云台两轴角度由遥控器控制
{
    pgimbal->gimbal_init.step=GIMBAL_CALI_START_STEP;
    pit_delta = -rc.ch4 ;       //遥控器通道4为pit轴增量（pit角速度）
    yaw_delta = -rc.ch3;        //遥控器通道3为yaw轴增量（yaw角速度)
    gimbal_set_pitch_delta(pit_delta);          //改变云台pit轴目标角度
    gimbal_set_yaw_delta(yaw_delta);                //改变云台yaw轴目标角度
}
```

- 注意**代码对齐**

```
for(;;)
{  //这种大括号要求一对是上下对齐的，并且独占一行
    if()//二级代码要缩进一个Tab键，往后依次类推，同级的并行
    {

    }
    setMotoSpeed();
}
```

- 对单行代码进行注释使用双斜杠位于代码后方

```
yaw_delta = -rc.ch3;          //遥控器通道3为yaw轴增量（yaw角速度）
```

对多行代码进行功能注释使用斜杠星位于代码块上方,并且超出代码一小节以划分代码功能模块

```
/*正转，伸出救援卡*/
    pid_calc(&pid_spd[4], moto_chassis[4].speed_rpm, -3000);
    setMotoSpeed(&hcan1, pid_spd[4].pos_out, 0,0,0,IDMARK_FIVE_EIGHT);
    pidDelay(2300,4,-3000);//自定义函数，用于稳定延时过程电机的转动
/*停住等待感应*/
    pidDelay(2000,4,0);
/*反转，收回救援卡*/
    pid_calc(&pid_spd[4], moto_chassis[4].speed_rpm, 3000);
    setMotoSpeed(&hcan1, pid_spd[4].pos_out, 0,0,0,IDMARK_FIVE_EIGHT);
    pidDelay(2300,4,-3000);
/*等待3秒，便于切换模式*/
    pidDelay(3000,4,0);
```

- 注意，代码中不理解的，或者有bug的，必须注明！可以使用多重感叹号进行提醒

```
HAL_TIM_PWM_Start(&htim8,TIM_CHANNEL_2);//启动Tim8的通道2
__HAL_TIM_SET_COMPARE(&htim8, TIM_CHANNEL_2,1000);//参数需要与实际情况联
调！！！！！
 HAL_TIM_PWM_Start(&htim8,TIM_CHANNEL_1);
__HAL_TIM_SET_COMPARE(&htim8, TIM_CHANNEL_1,1500);
osDelay(2000);
```

可移植性和健壮性

- 多使用宏定义和HAL句柄
- 可变参量使用宏定义
- 有能力考虑条件编译，使得代码更加强大
- 该部分参考代码：参考FLASH驱动

文件权限和留名

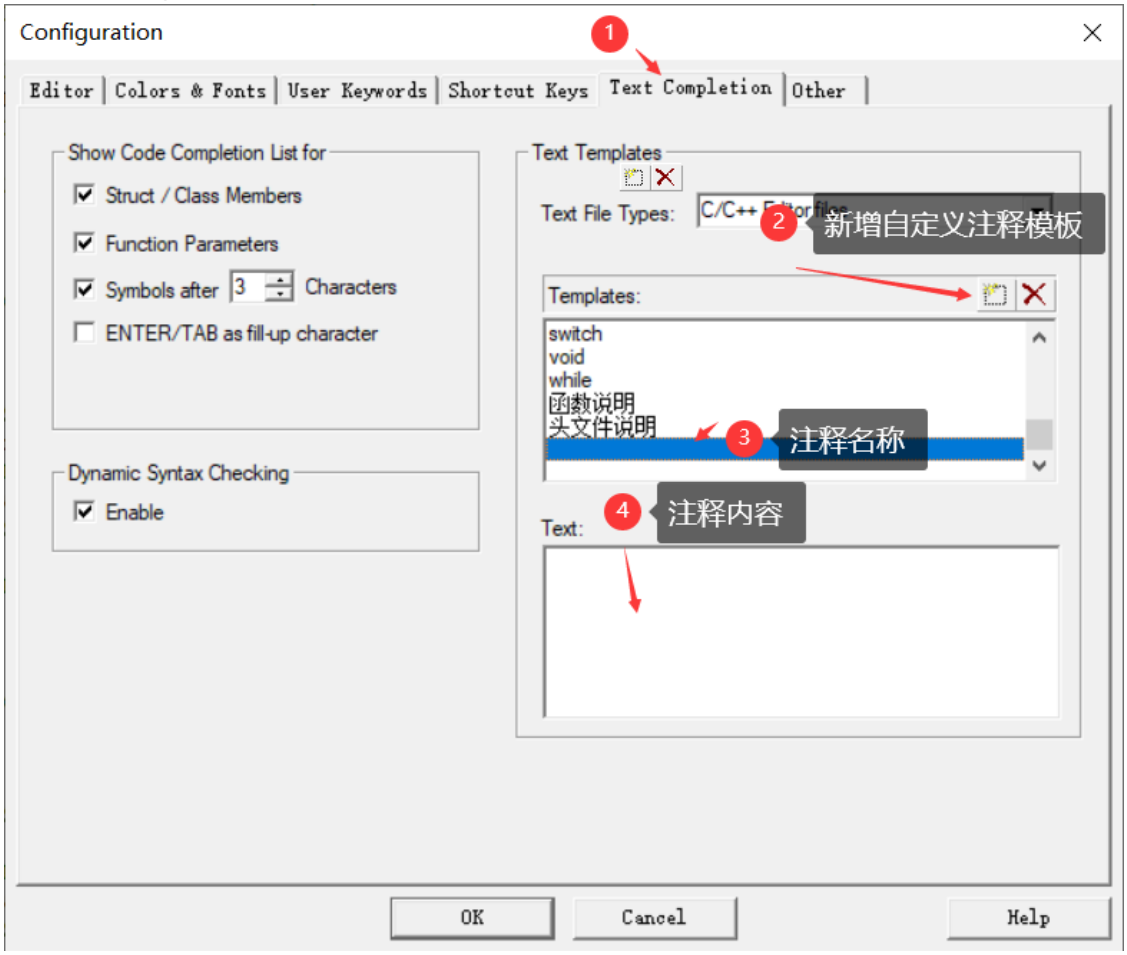
- 文件类型文件用途
- 文件使用时要注意的
- 说明文件属于团队or你个人

- 证书声明（如果你用的话）
- 创作/修改者的留名

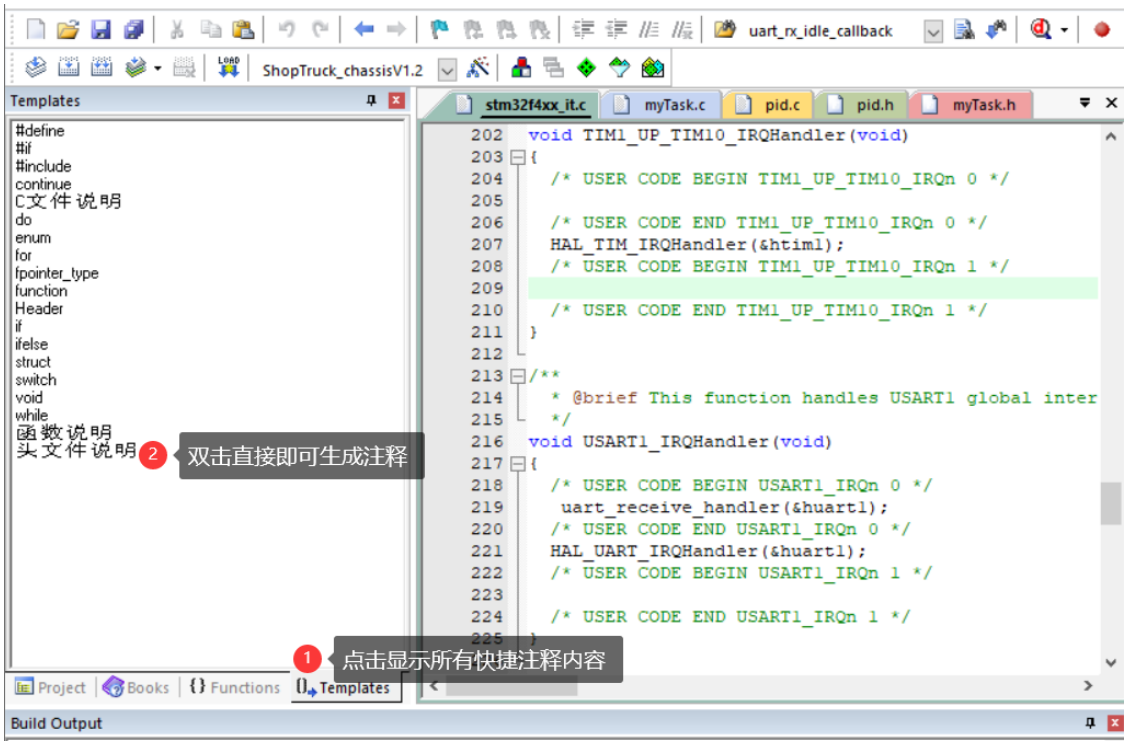
```
/*
 * FLASH_W25Q256.h - The C head file of the SPI FLASH(W25Q256) driver
 * NOTE: This file is based on HAL library of stm32 platform
 *       The default initialization device is in QSPI mode!!!
 *
 * Copyright (c) 2020-, FOSH Project
 *
 * SPDX-License-Identifier: Apache-2.0
 *
 * Change Logs:
 * Date           Author            Notes           mail
 * 2020-03-20     StudyHooligen     first version   2373180028@qq.com
 */
#ifndef _FLASH_W25Q256_H_

#define _FLASH_W25Q256_H_
```

Keil自定义注释

- Edit --> Configuration 进入如下界面设置



- 点击“OK”
- 即可快捷使用



代码命名规范

命名简洁为主，电机机构体，PID结构体等分为独立文件进行编写，便于移植使用

电机结构体

```
具体见官方示例代码，后续补充整理自主代码
```

PID结构体

```
具体见官方示例代码，后续补充整理自主代码
```

- 电机命名

```
moto_功能
例子：moto_chassis      //底盘电机
      moto_upraise     //抬升电机
```

- PID相关命名

```
pid_功能
例子：pid_spd    //调节速度的PID
```

- 任务函数相关命名

```
任务TaskFunction
  例子：rescueTaskFunction    //救援任务功能函数
任务TaskHandle
  例子：rescueTaskHandle      // 救援任务句柄
```