基于stm32cubeIDE和protues的协同开发流程

起始时间：2019年10月19日星期六

**[2019年10月19日星期六]**

项目1：使用stm32cubeIDE开发基于固件库的blink

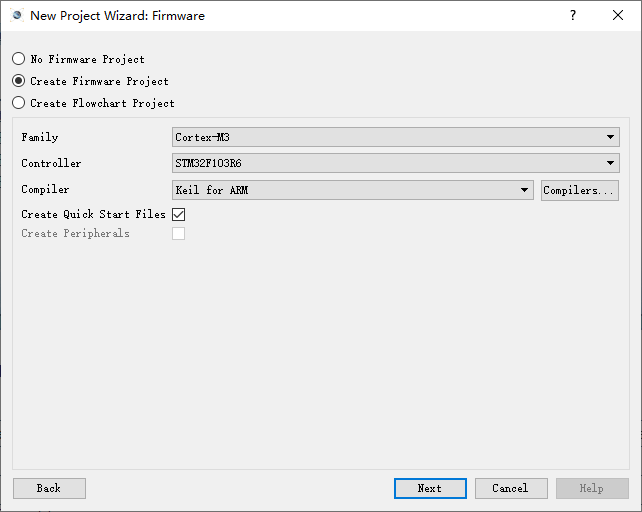
目的：以1s周期blink接于PB8引脚上的LED，为兼容MCBSTM32E，设置使用74HC244 LEDs驱动

平台：stm32cubeIDE 1.1； protues 8.7

protues 8.7仅支持stm32 M3的STM32f103X；protues 8.9将可支持M3和M4

流程:

* 1. 新建基于M3的项目

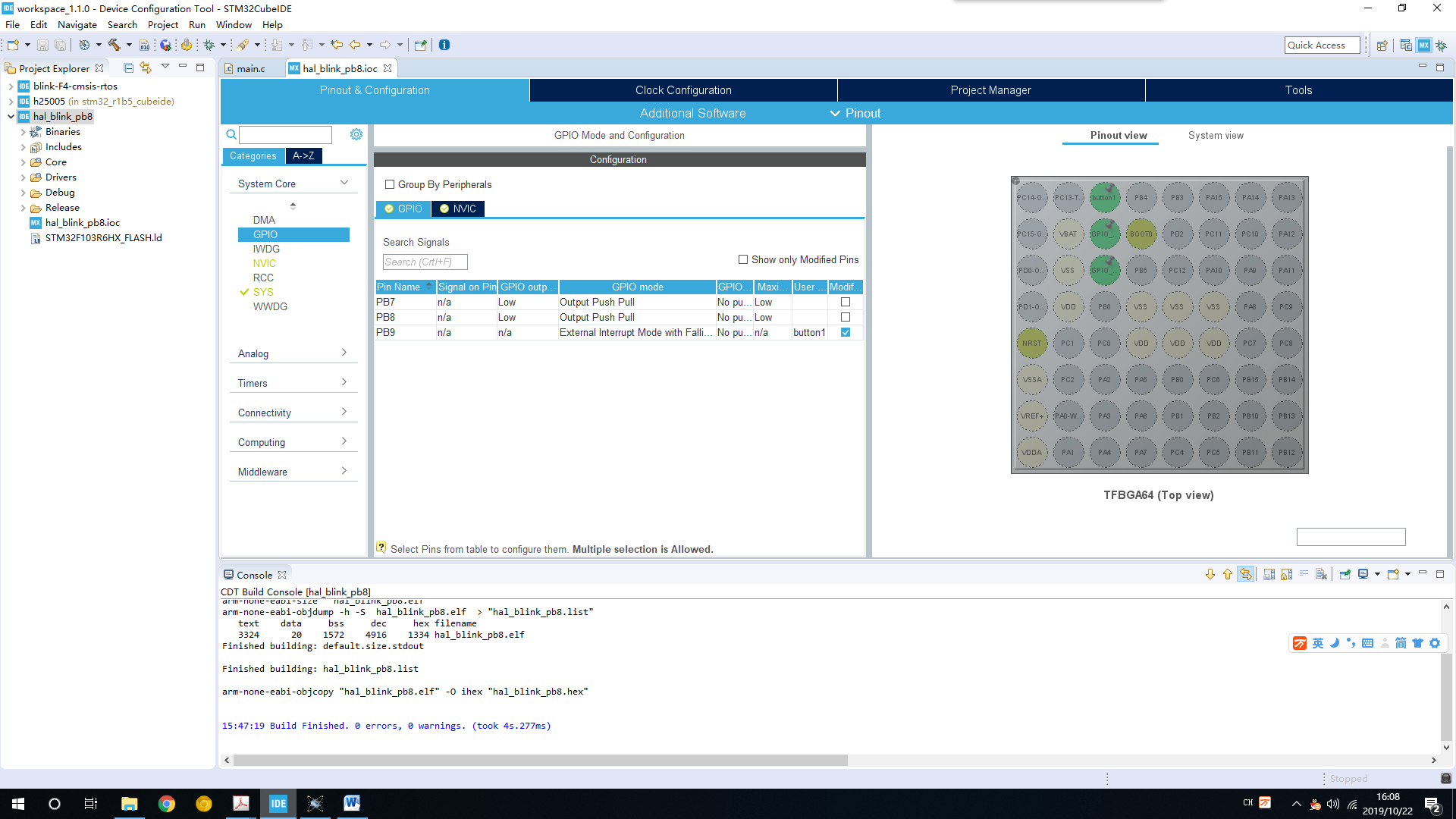


包括原理图和固件。

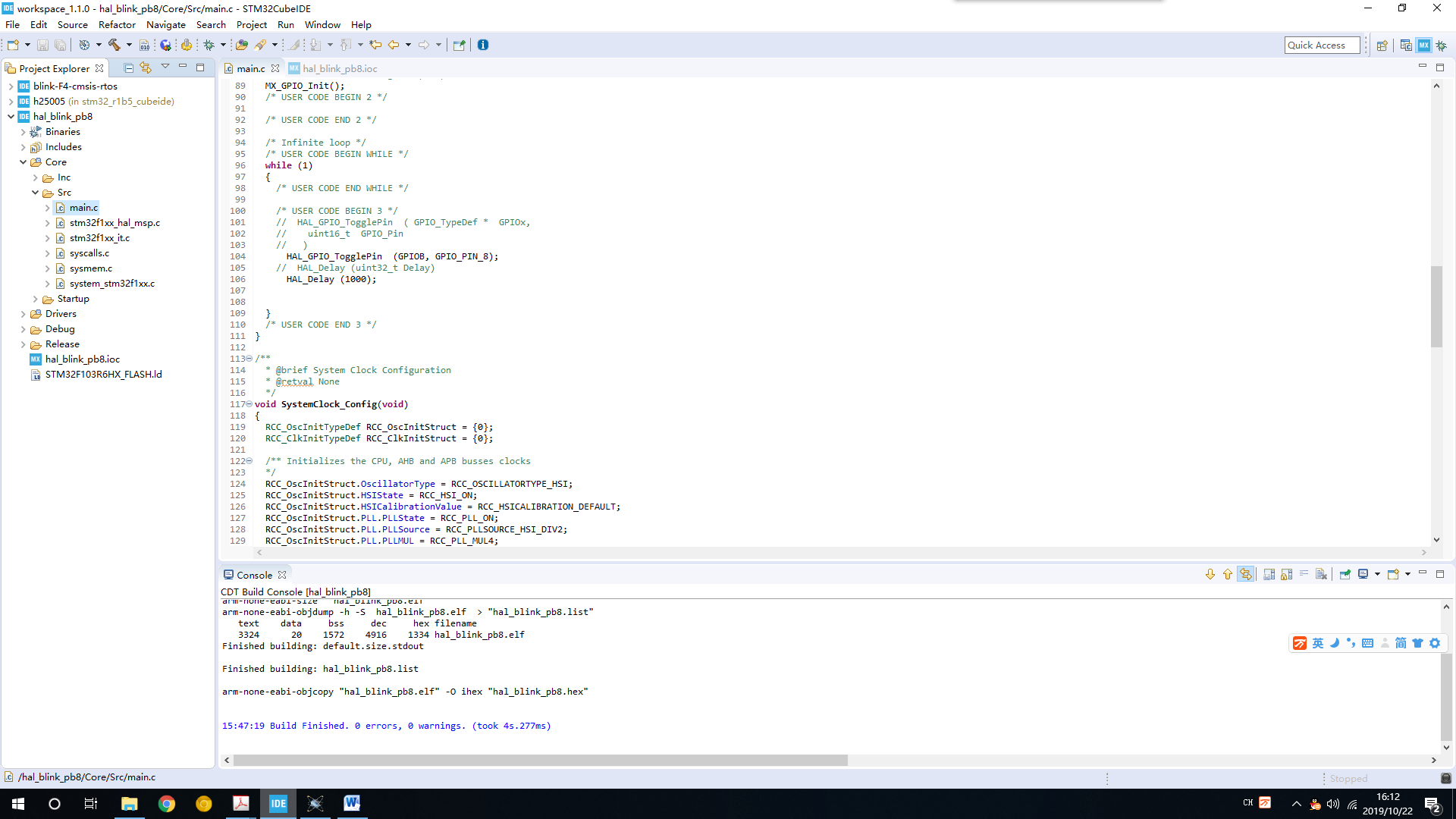
在原理图中设置如下图电路。



* 1. 在stm32CubeIDE中新建stm32Cube项目

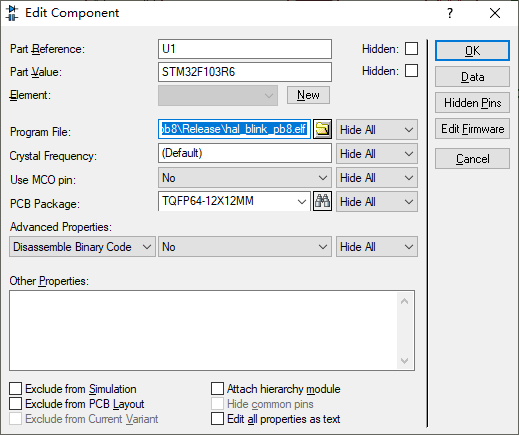


双击.ioc文件，配置PB7,8为GPIO\_OUTPUT；保存后同意‘生成代码’；打开./Core/src/main.c



增加上述2行代码。编译生成release版本的.elf文件

* 1. 在protues中引入这个release版本的.elf文件



双击protues原理图中的U1，装入release版本的.elf文件，即可进行protues仿真。

**[2019年10月22日星期二]**

项目2：实现GPIO中断输入

目的：使用1个GPIO引脚连接一个button，toggle另一个GPIO引脚的led

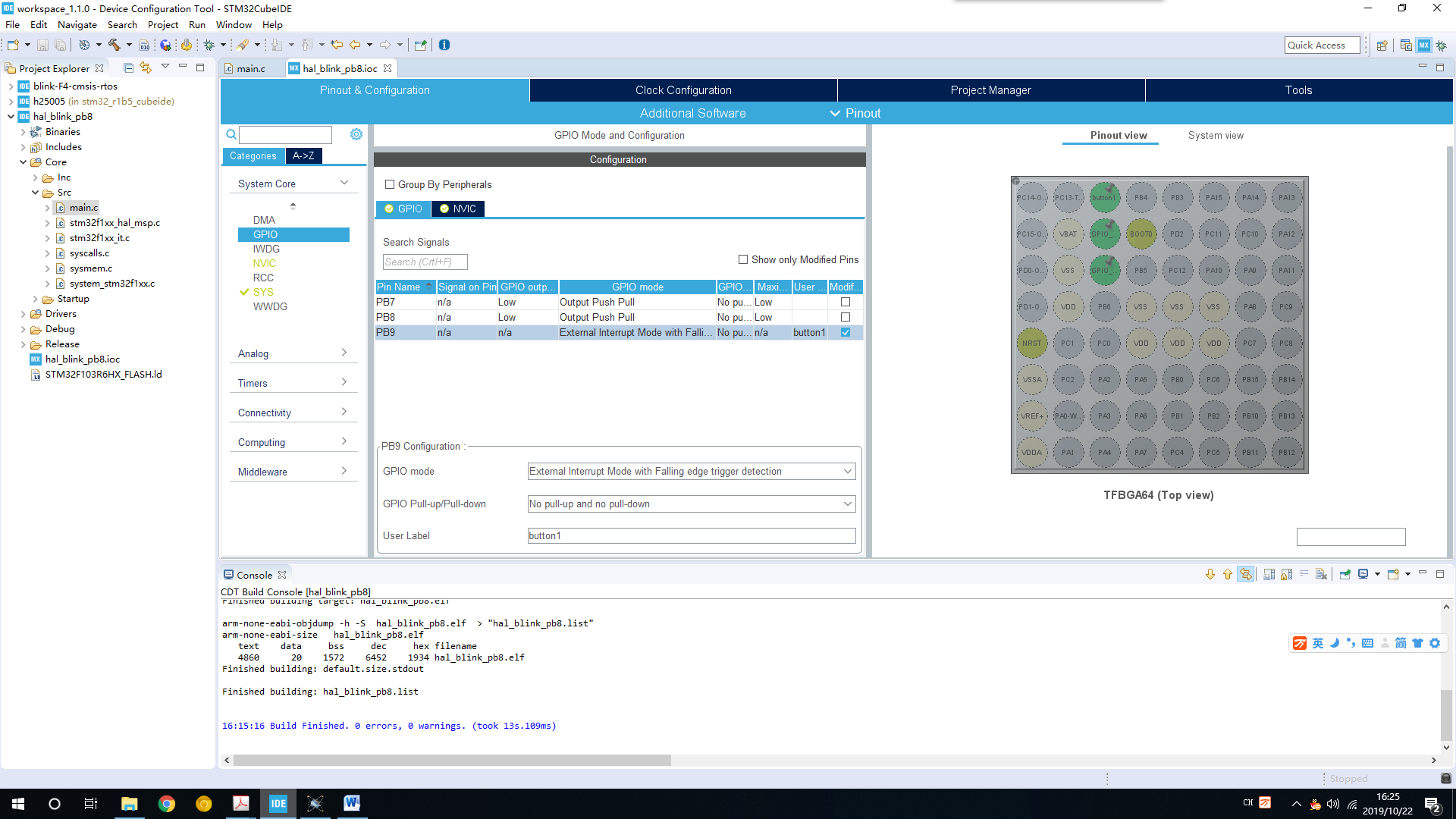
平台：stm32cubeIDE 1.1； protues 8.7

实现流程：

2.1 protues原理图

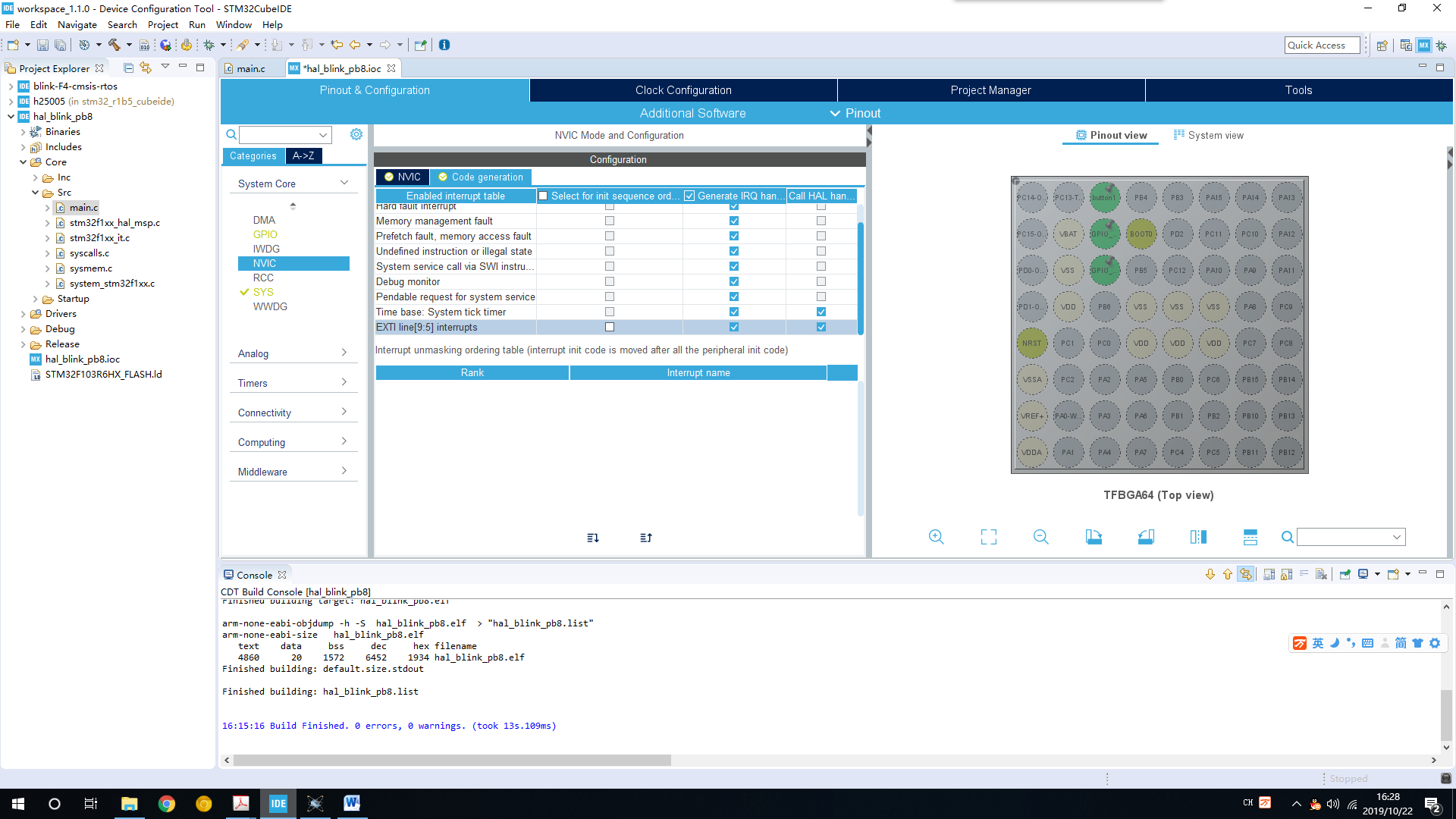


2.2 stm32CubeIDE项目



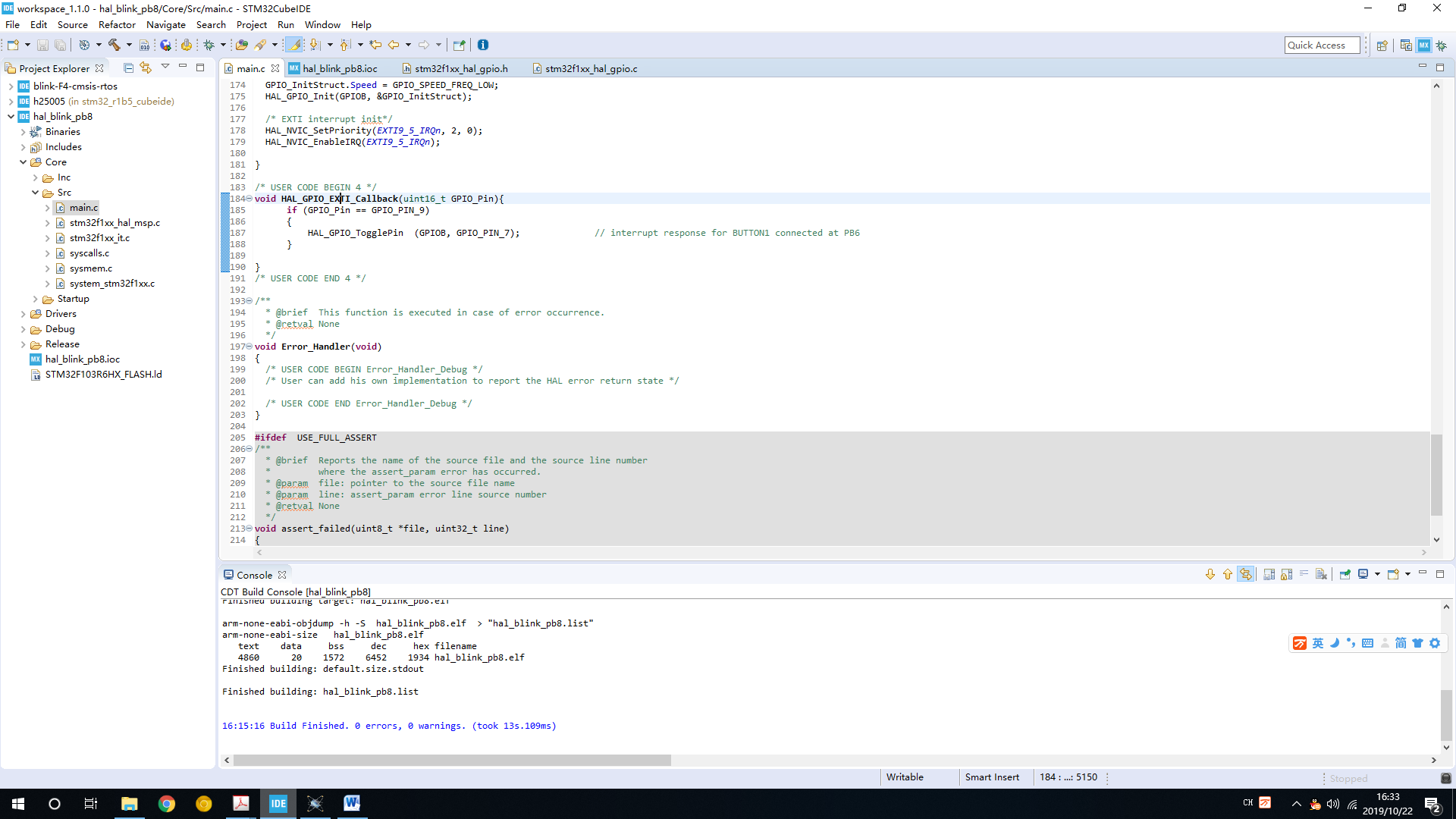
2.2.1 使用.ioc(即使用cubeMX)增加PB9为外部中断输入，使用Falling edge触发。

2.2.2 在NVIC中



配置EXTI line[9:5]，选Call HAL handle，保存并重新生成代码。

2.2.3 打开main.c

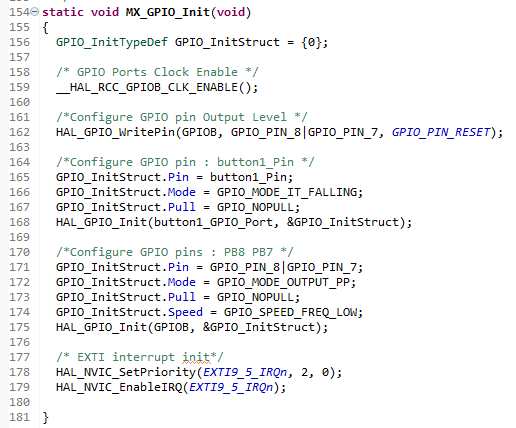


增加如上图代码；保存全部，新生成release版本的.elf；然后返回protues，在U1中引入此.elf文件即可进行仿真。

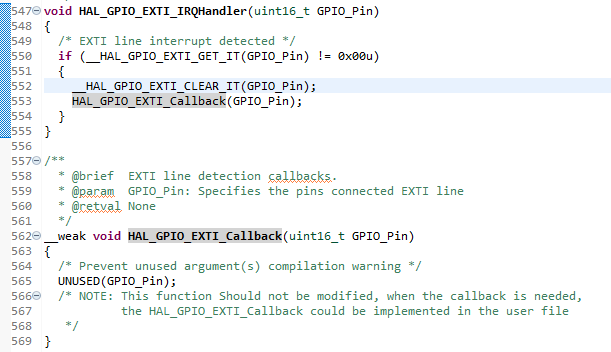
2.2.4 stm32cubeIDE过程分析

上述2.2.2小节cubeMX配置产生新代码如下：

main.c



Stm32f1xx\_hal\_gpio.c



但注意在Stm32f1xx\_hal\_gpio.c中的HAL\_GPIO\_EXI\_Callback(uint16\_t GPIO\_Pin)函数为\_\_weak类型，即它需要重新实现。

因此，使用2.2.3节方法重新实现即可。

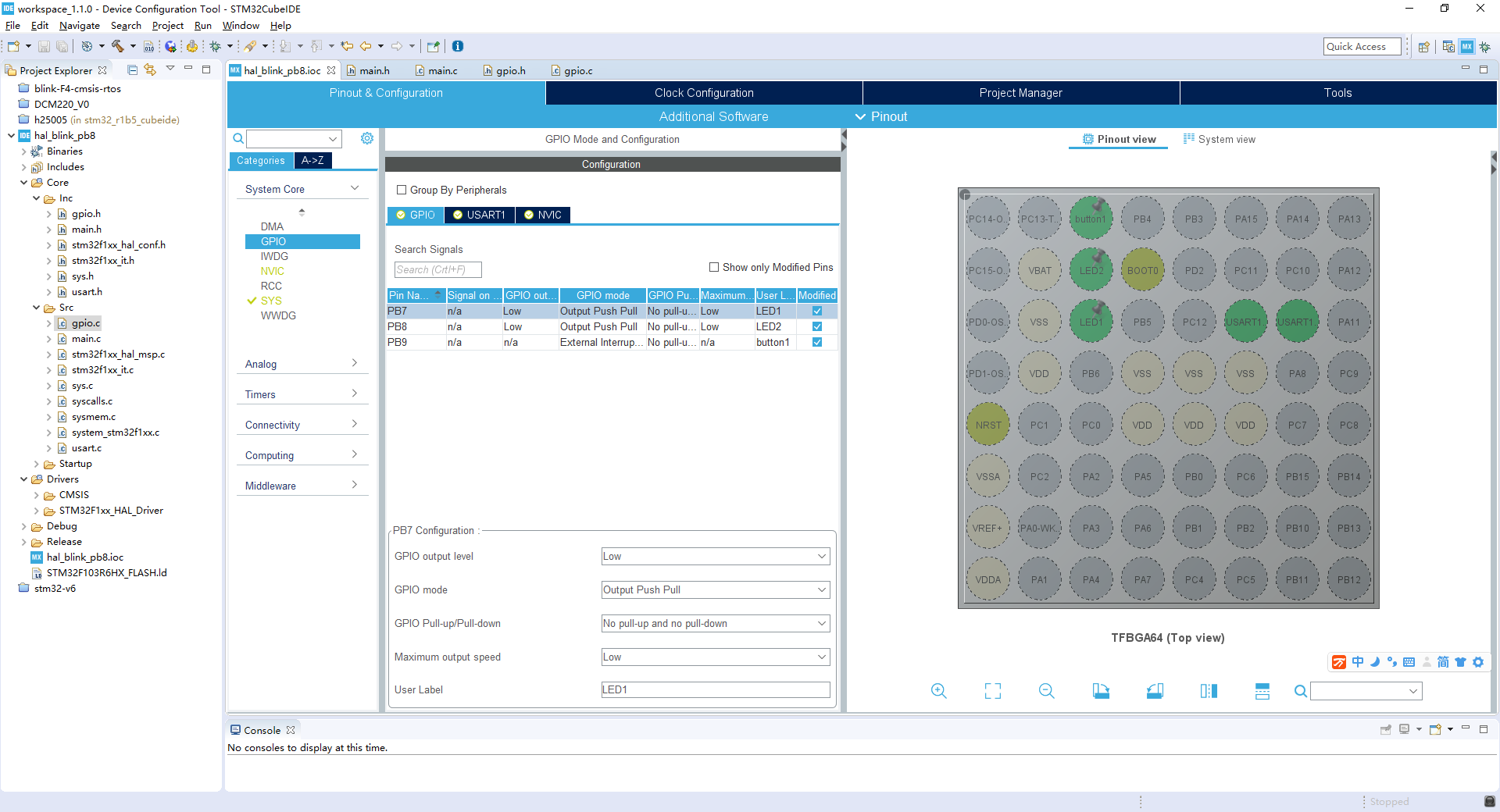
**[2019年11月07日星期三]**

项目1A：实现GPIO BSP设计（CUBEMAX的深入学习）

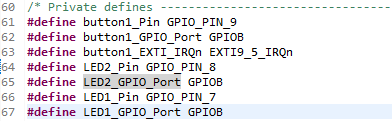
目的：对指定的LED1,2,…形成上层代码包裹

参考如下链接：<http://www.waveshare.net/study/article-630-1.html>

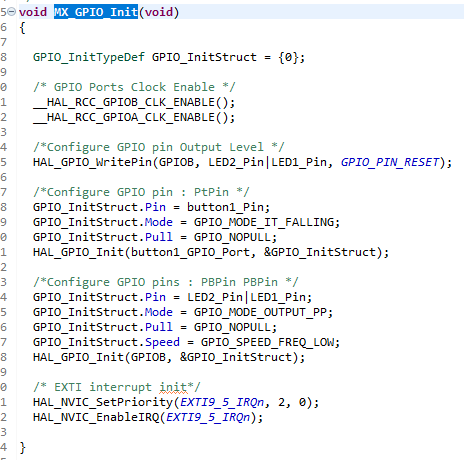
平台：stm32cubeIDE 1.1； protues 8.7



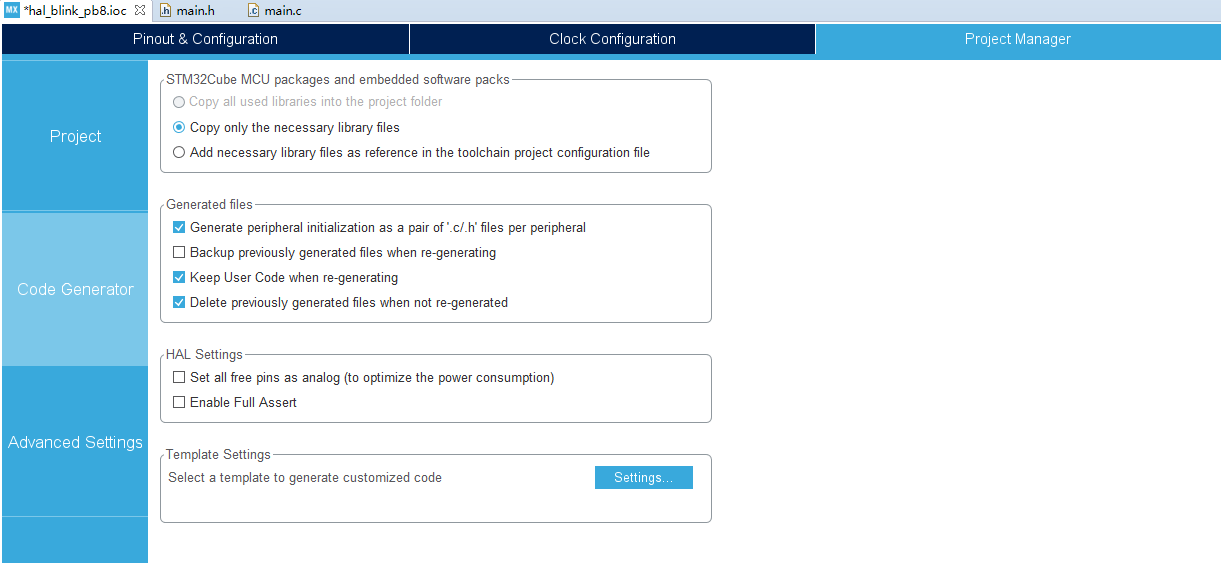
显然，用户更容易记住LED1这样的定义，CUBEMAX中通过在main.h文件里添加\_GPIO\_Port和\_Pin GPIO后缀的预定义方法帮助用户使用这些GPIO。

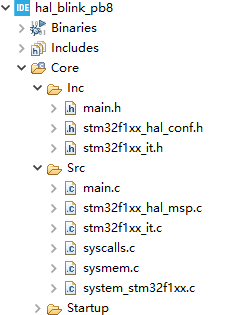
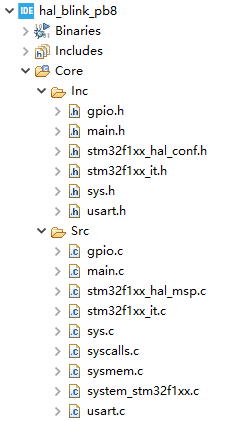


当main.c或其它地方代码对这些GPIO进行操作时，可以使用上述宏定义。



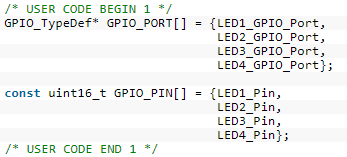
另外，CUBEMAX为了保持代码的整洁，上述通过HAL对GPIO的操作附加函数是增加在main.c文件里的；然而，为便于代码的归类处理，可以将一类IP（功能模块）分别由.h和.c来管理，如需此方法，如下图进行设置：



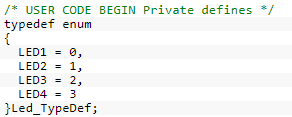
 

缺省未开启“Generate peripheral…” 开启后产生代码情况

进一步地，可以使用两个数组来保存这些GPIO\_Port和\_Pin GPIO信息，放在gpio.c的USER CODE BEGIN 1内。



并使用一个enum来索引他们，放在gpio.h的USER CODE BEGIN Private defines内。



这样，在使用多个GPIO时可以通过一个Led\_TypeDef的变量led来进行索引，如：

HAL\_GPIO\_TogglePin(GPIO\_PORT[Led], GPIO\_PIN[Led]);

上述代码封装，即为BSP建立过程。

**[2019年11月7日星期四]**

项目3：定时器及中断

**[2019年11月8日星期五]**

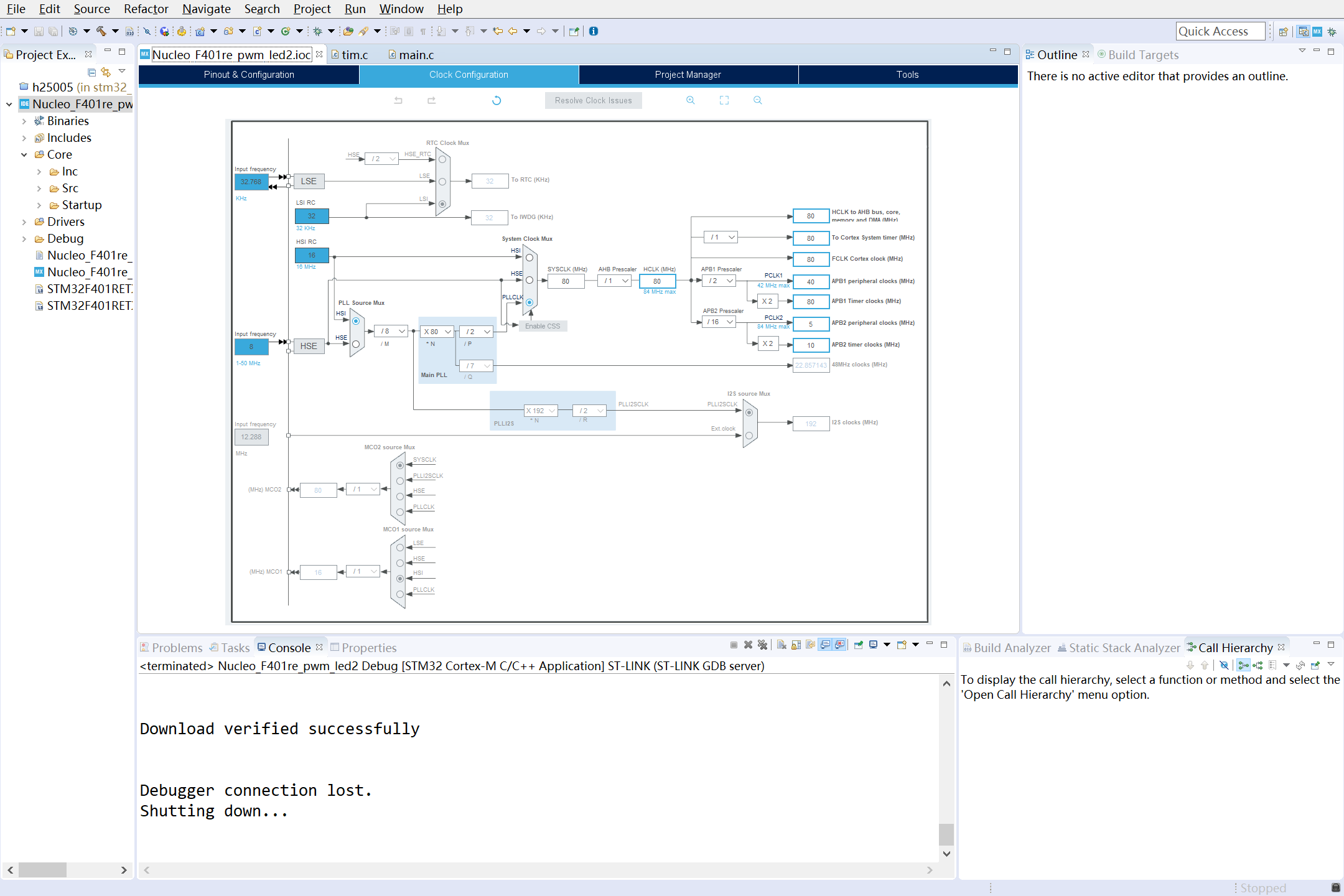
项目4：定时器的PWM

项目目标：在Nucleo64 stm32F401re平台上实现LED2的呼吸灯效果。

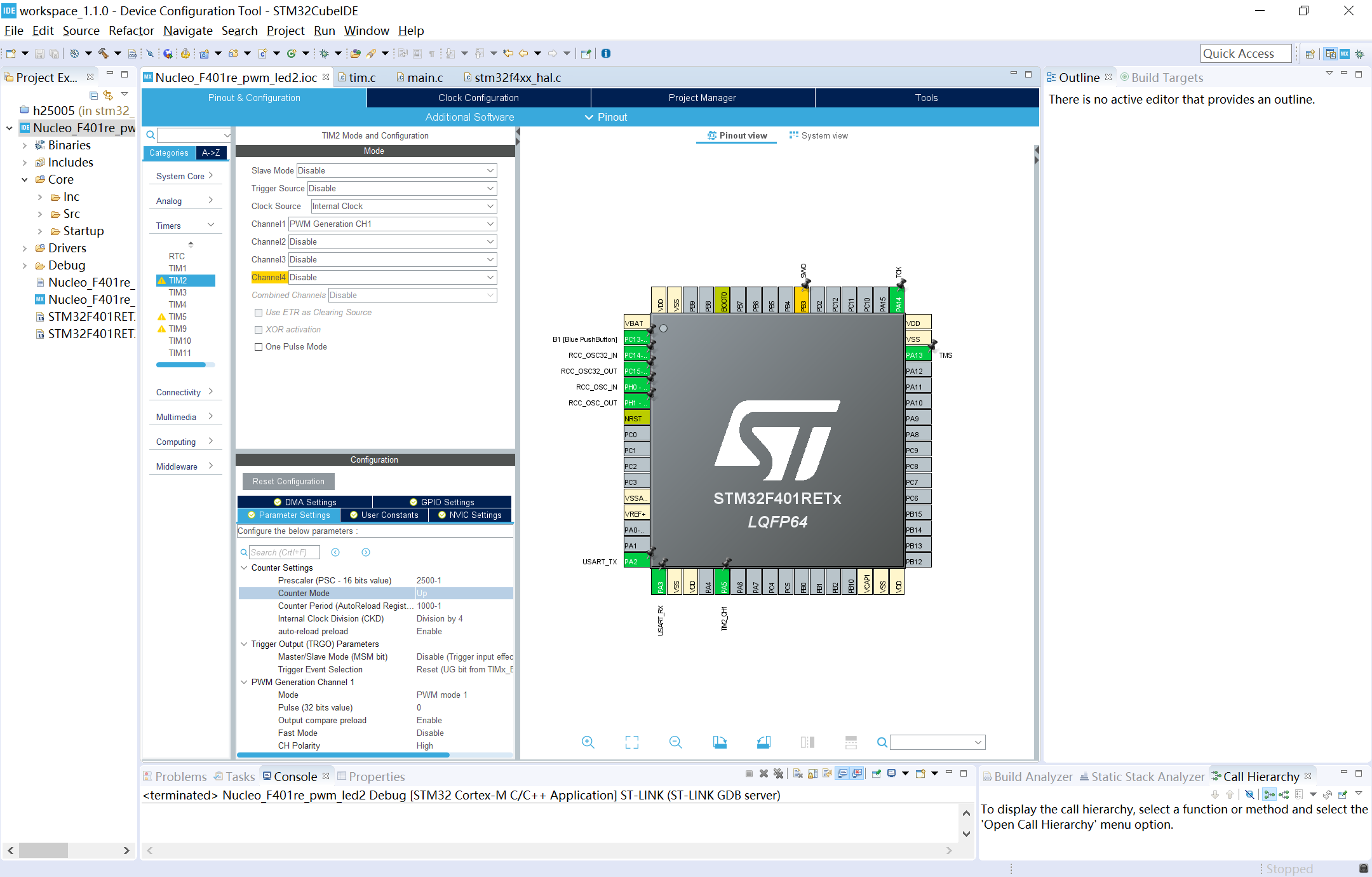
参考链接：<http://www.waveshare.net/study/article-643-1.html>

实现过程：

打开STM32CubeMX新建工程，配置时钟树如下

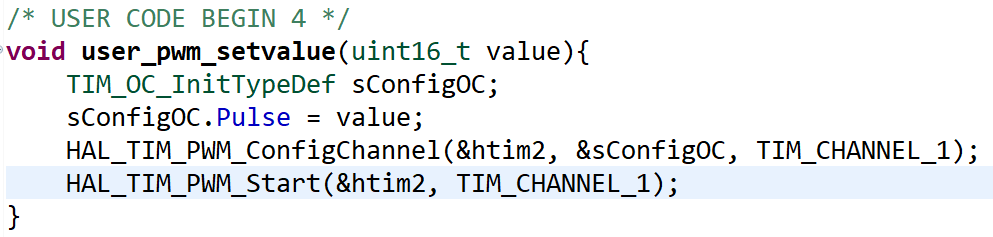


配置TIM2

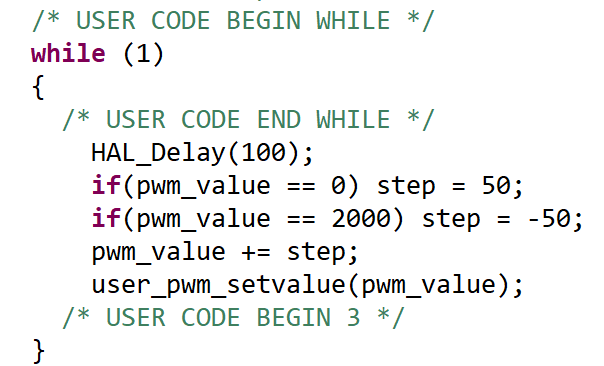


注意，不同的TIMER对应不同PA引脚PWM输出能力，如TIM2对应PA5，而TIM3对应PA6，cubeMx可以保证这些配置的约束关系。

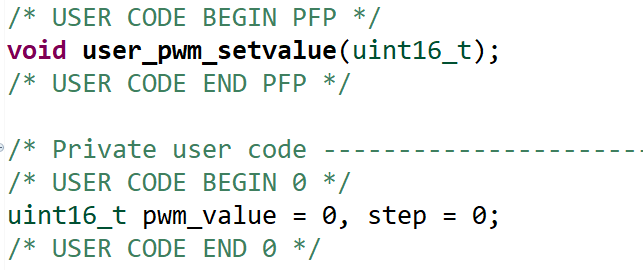
在main.c里增加脉冲宽度调整函数



然后在main.c的while中不断调用



上述pwm\_value, step需要在main.c或main.h中定义，建议如下：



项目5： UART

<https://www.cnblogs.com/lizhiqiang0204/p/11659957.html>

<https://blog.csdn.net/zdw6868/article/details/82558888>

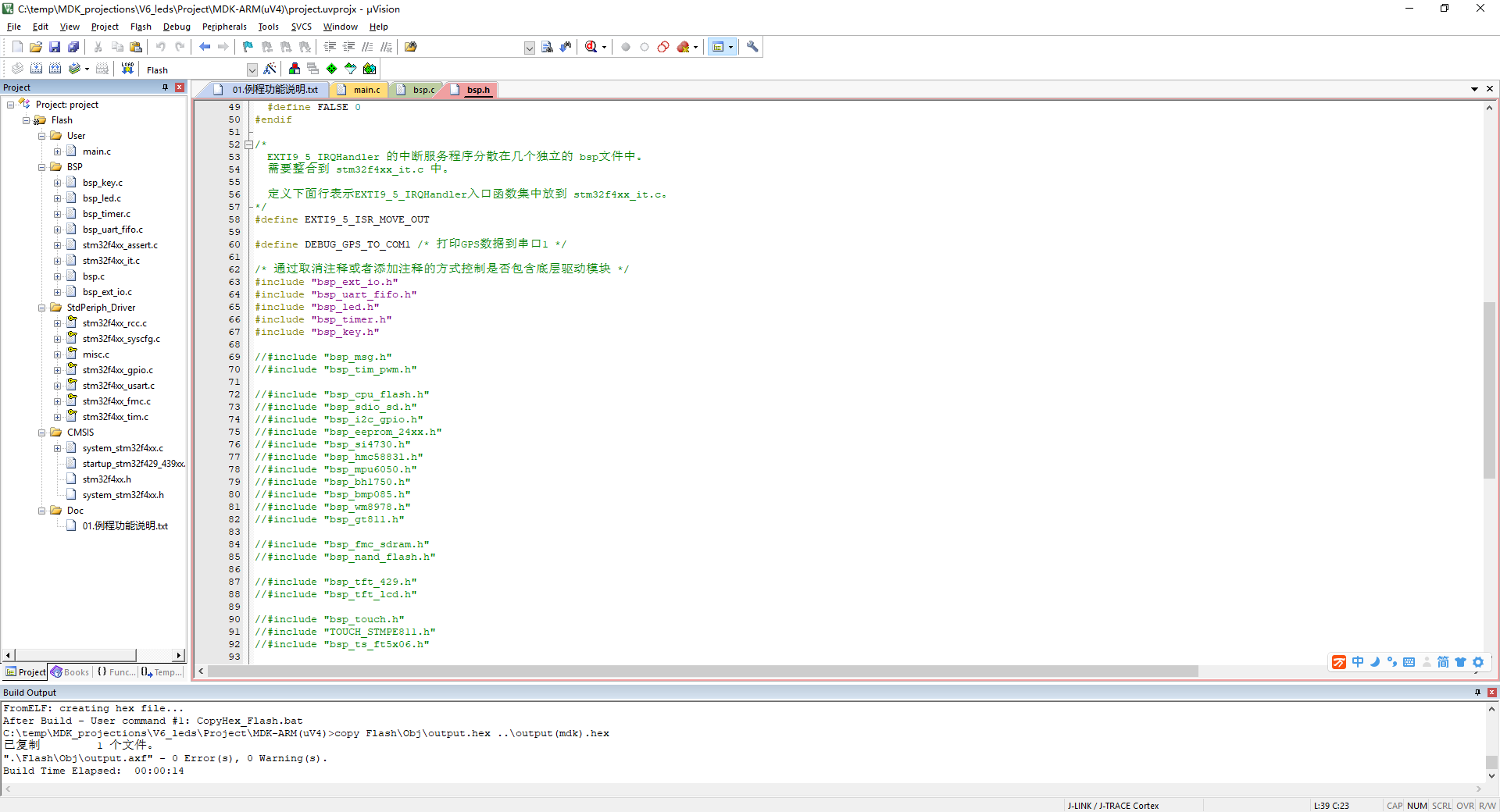
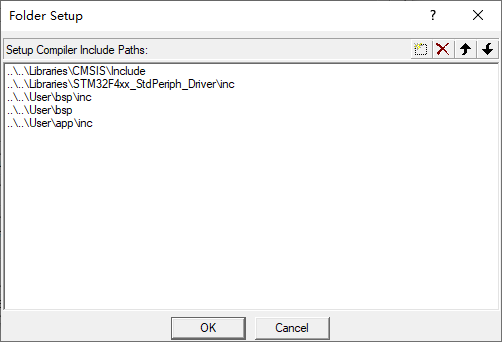
<https://blog.csdn.net/as480133937/article/details/99073783>

**[2019年11月12日星期二]**

项目6：在stm32cubeIDE中使用armfly V6的BSP

分析过程：

一、下图为定时器中断led项目

问题：armfly V6的BSP 使用是的标准外设库stm32f4\_dsp\_stdperiph\_lib，而不是cubeIDE使用的硬件抽象层库HAL，因此

**[2019年11月28日星期四]**

**关于CMSIS-RTOS，CMSIS-OS，freeRTOS**

CMSIS-RTOS 是ARM对RTX的封装；

CMSIS-OS 是ARM对freeRTOS的封装；

使用CMSIS-OS最直接的方法就是通过stm32cubeide中进行配置和管理。

另一种方法是使用CUBEMX+MDK5

**[2019年12月11日星期三]**

**关于在STM32CUBEIDE中同时使用emWin和freeRTOS**

:STemWin\_CM7\_wc32\_ARGB.a