利用 Keil 进行 STM32F103RB 芯片软件仿真模拟方法

目录

1. 利用 cubemx 建立 STM32F103 芯片工程

（1）MCU 管脚配置

在 cubemx 中可以直接建立 STMF103RB NUCLEO 开发板的默认工程，这个工程中已经配置好一个按钮和 led 小灯。其中按钮 PC13，LED 灯为 PA5 管脚。用默认配置就可以了。

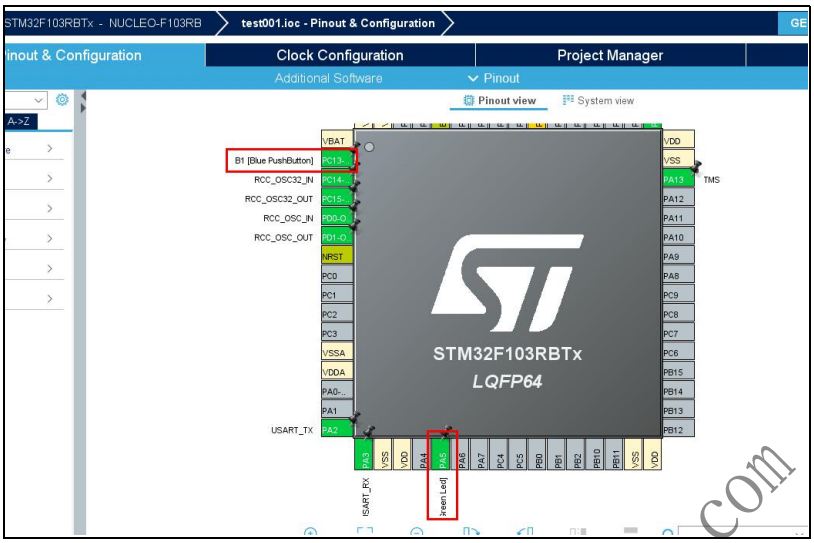

图 1 STMF103RB NUCLEO

（2）系统时钟配置

由于软件仿真 MCU 没有外部硬件，因此把时钟配置内部时钟。

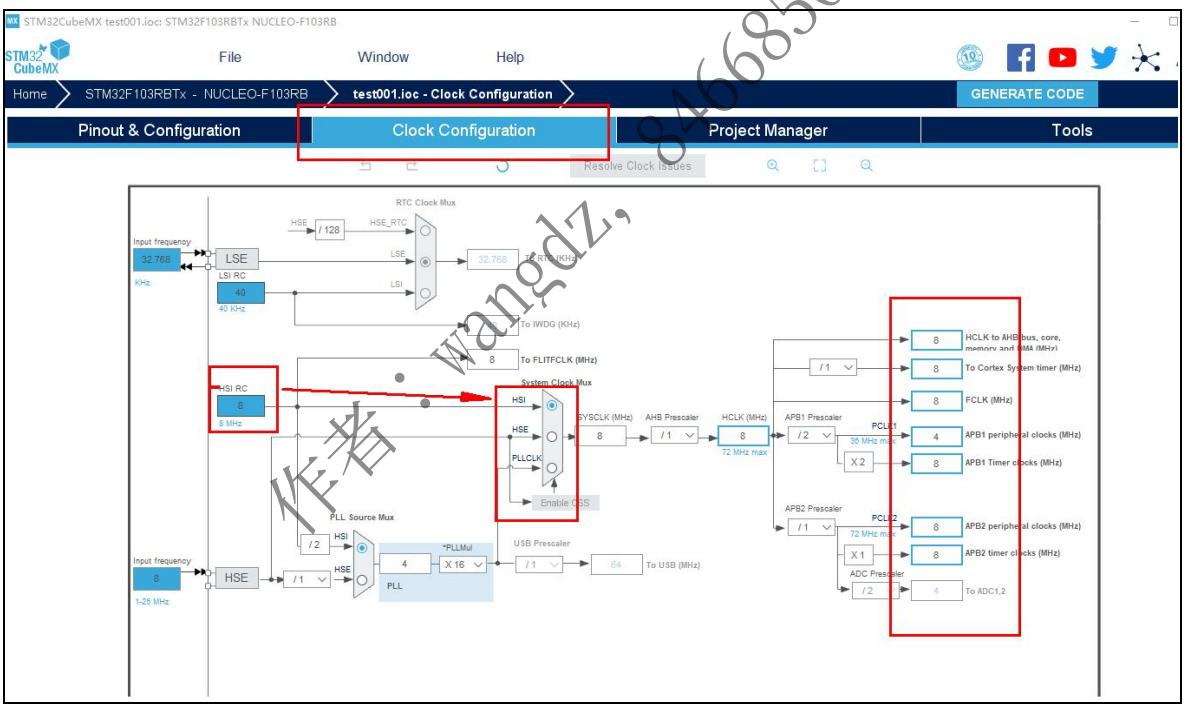

图 2 系统时钟配置

（3）生成代码

需要把当前的配置存储为工程，按下图建立工程名字和项目存储的位置，注意，这里的工程和文件夹名字尽量不要使用中文，中文有时导致莫名的问题。最后点击 4 的生成代码工具。工程编译为 keil 代码项目，最后点击“open project”打开 keil 下的工程。

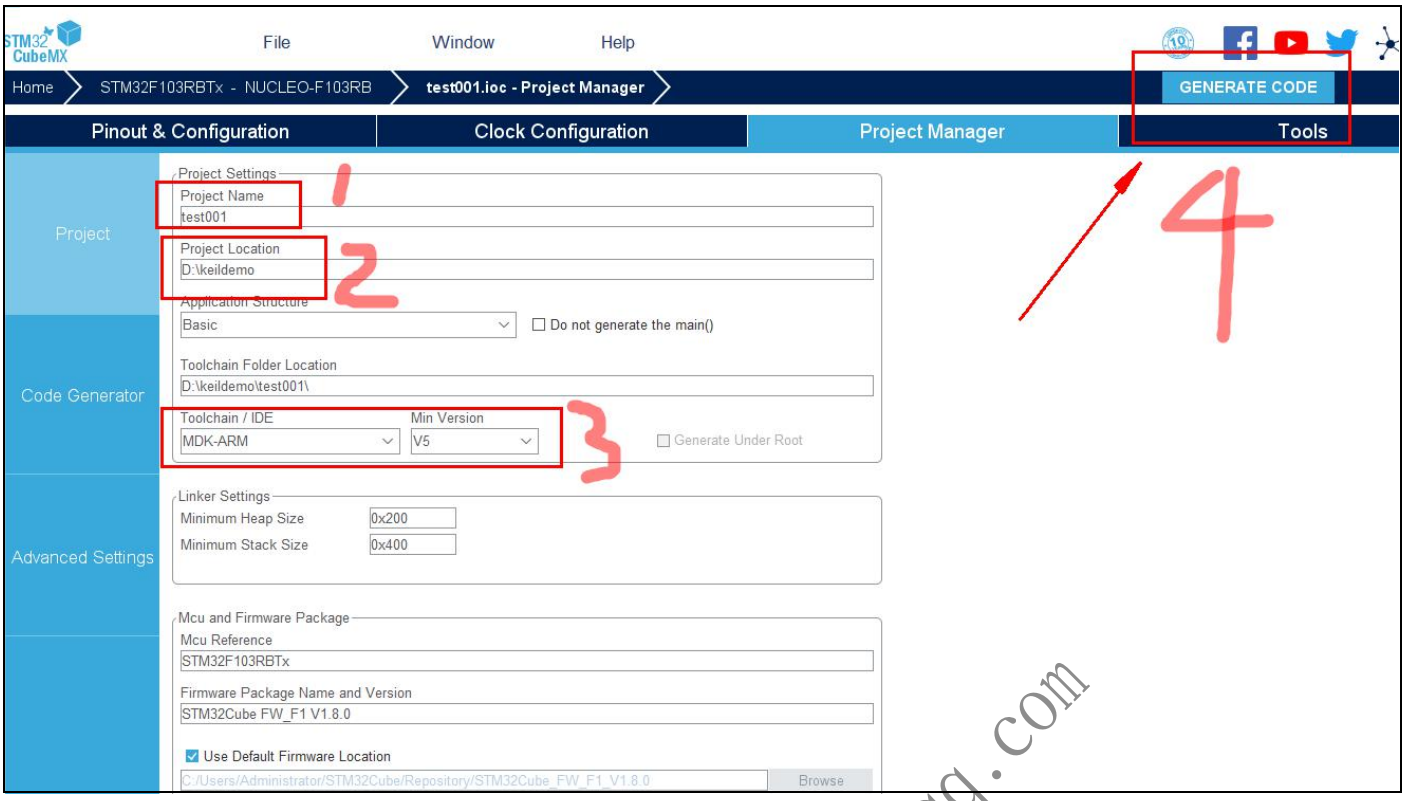

图 3 项目保存及生成

图 4 打开工程

2. 修改 KEIL 程序代码

（1）修改 main.cd 代码

如下图所示，找到项目中的 main.c 程序，然后在 while(1)循环中添加小灯翻转的代码

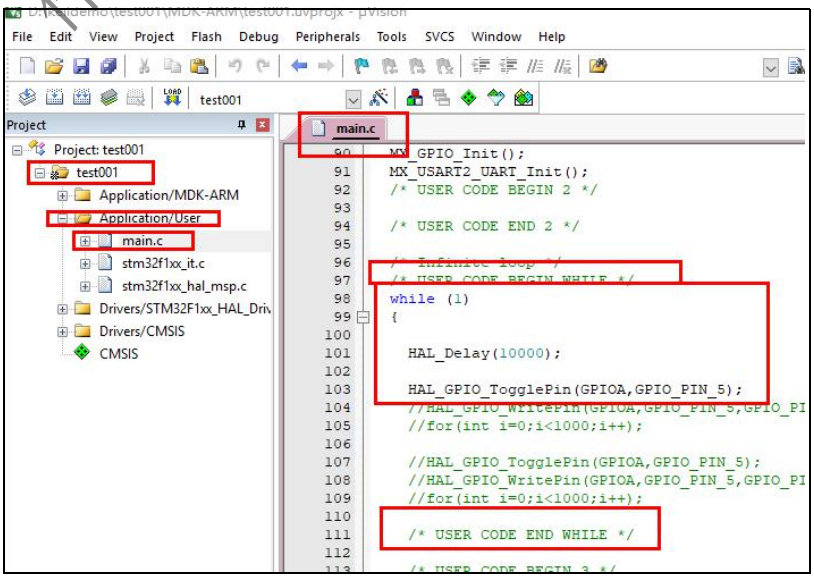

图 5 修改主程序

（2）编译程序代码

修改完代码后，保存工程。然后点击左上的编译按钮，在下面出现报错信息，如果 error 为 0 说明程序就没有问题，如果与 warning 需要分析一下，看看是否需要必须解决。

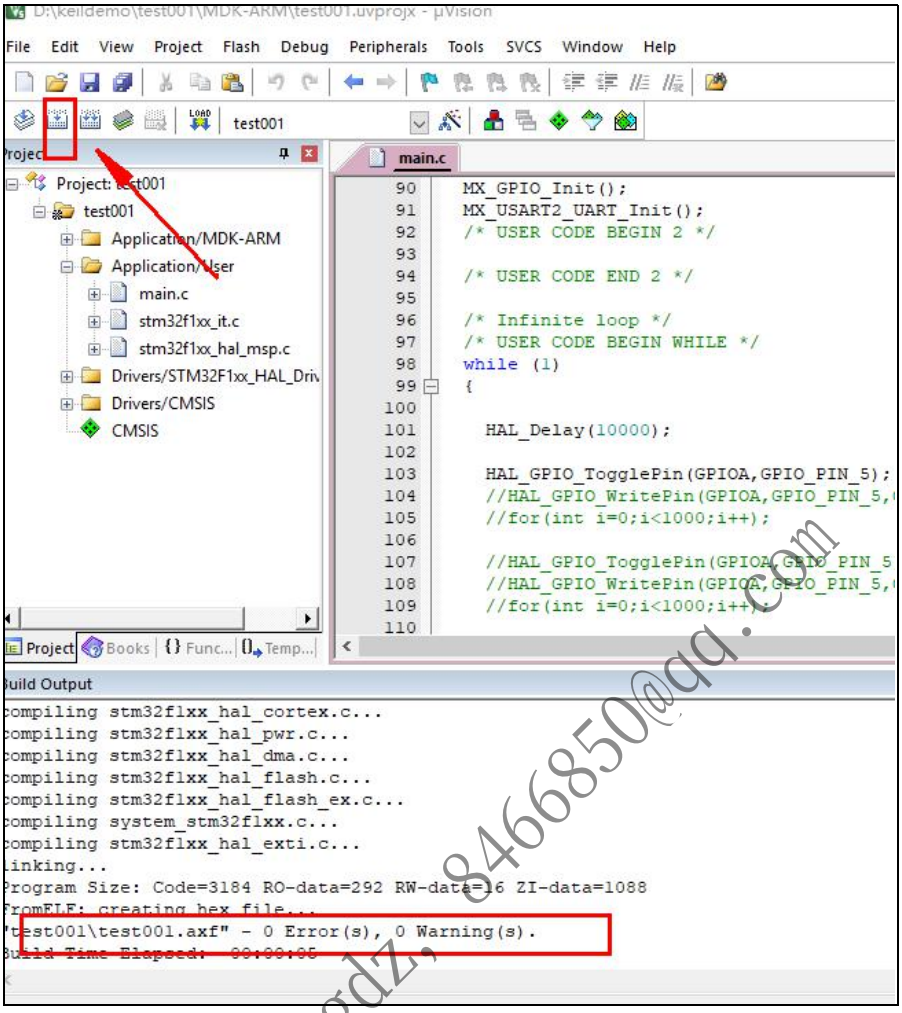

图 6 编译 keil 项目

3. 配置 STM32F013RB 虚拟仿真参数

（1）调用设置窗口

在工程的文件夹名字上点击右键，在右键菜单中选择 Option 选项

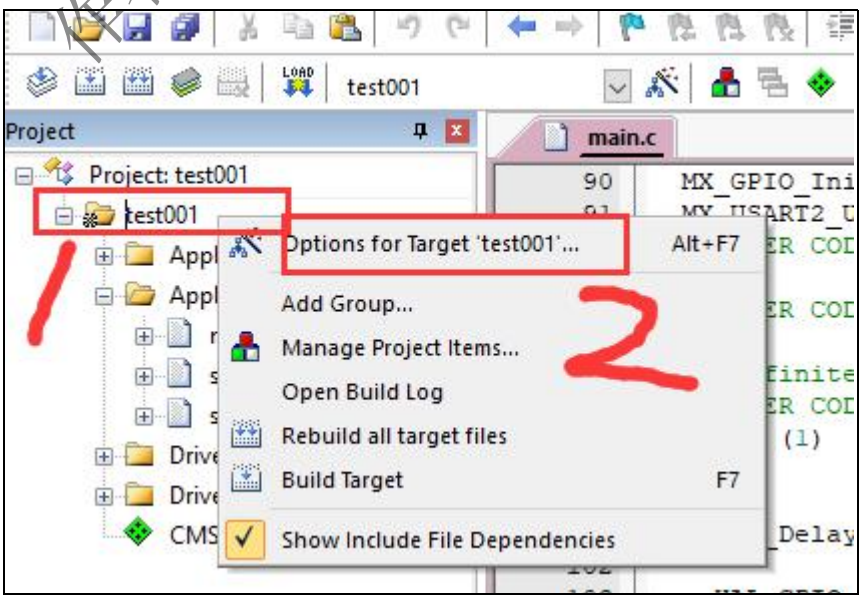

图 7 调出配置窗口

（2）配置仿真参数

在弹出的 option 菜单中，选择 debug 菜单，按下图所示，选择 2 处为使用仿真模式。在 3 和 4 处填写你要仿真的芯片，参数如下：

① dialog dll 要改成 DARMSTM.DLL，

② parameter 需要改成 -pSTM32F103RB (这个参数是根据不同的芯片改写的）

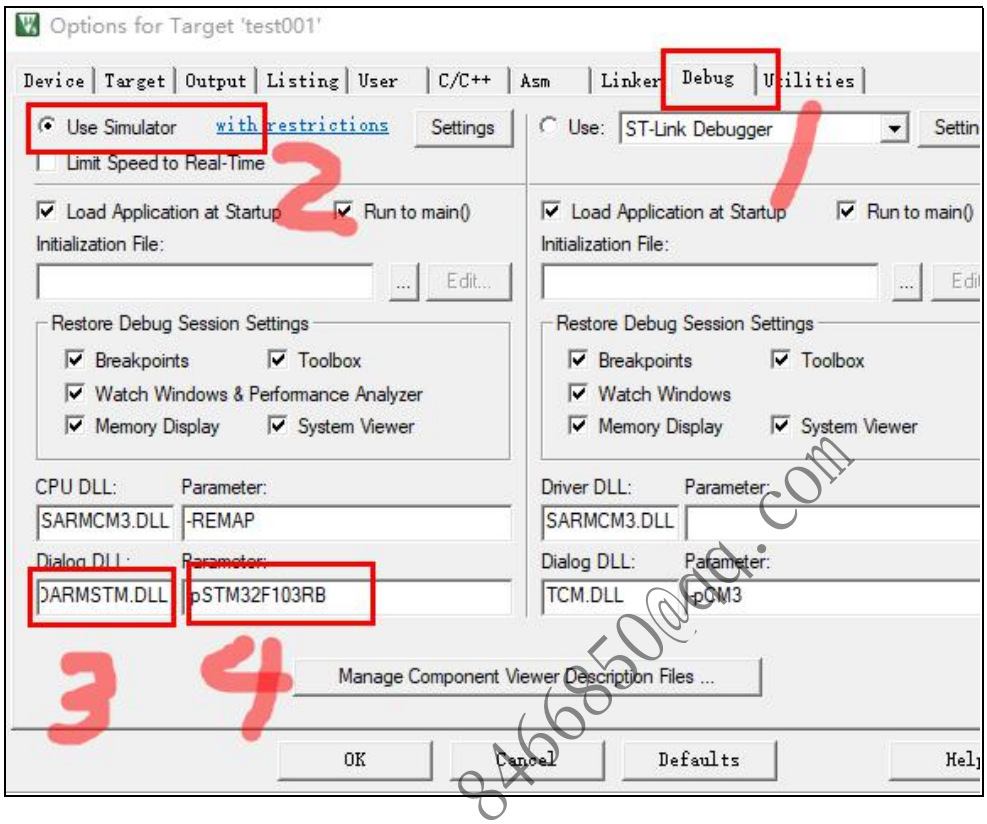

图 8 STM32F103RB 芯片仿真参数配置

4. GPIO 输出--管脚监控

（1）启动调试状态

点击 debug 菜单下的 start debug。。。命令，启动系统进入调试状态。

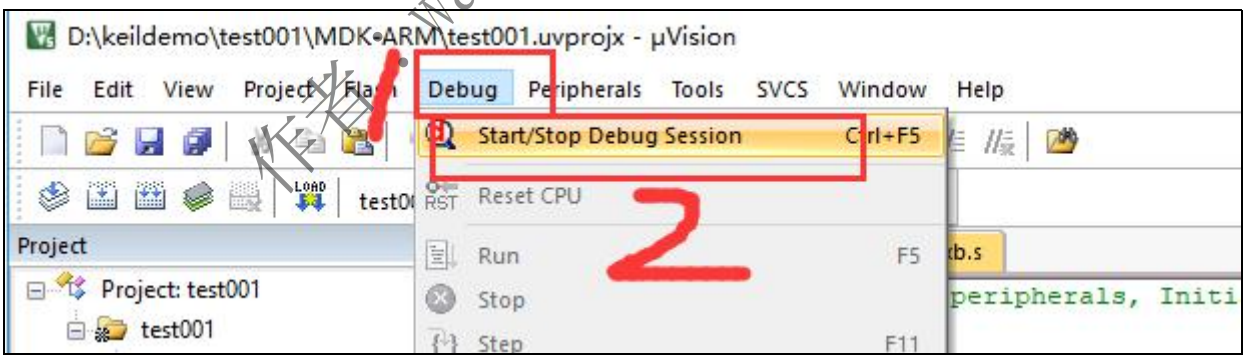

图 9 进入 debug 调试状态

（2）启动逻辑分析仪

在 debug 调试状态下，才可以看到逻辑分析仪的配置命令。按下图所示的步骤操作。

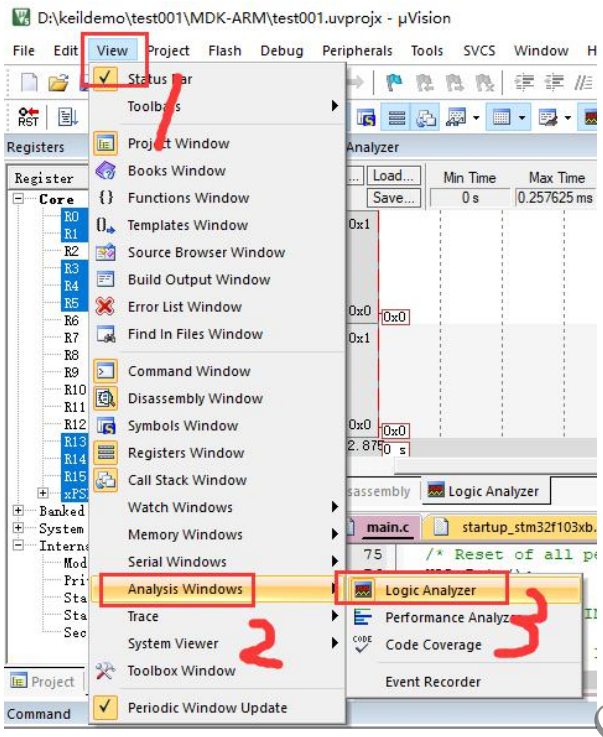

图 10 启动逻辑分析仪命令

（3）配置监控 GPIO 管脚

点击逻辑分析仪的“setup”命令，启动配置窗口，点击窗口中右上的新建按钮，在下面输入 PORTA.5，表示监控 A 端口的 5 号管脚，就是 PA5.

如下图所示，配置 display 为 bit 方式，颜色可以自定义，最后选择 hex16 进制模式。

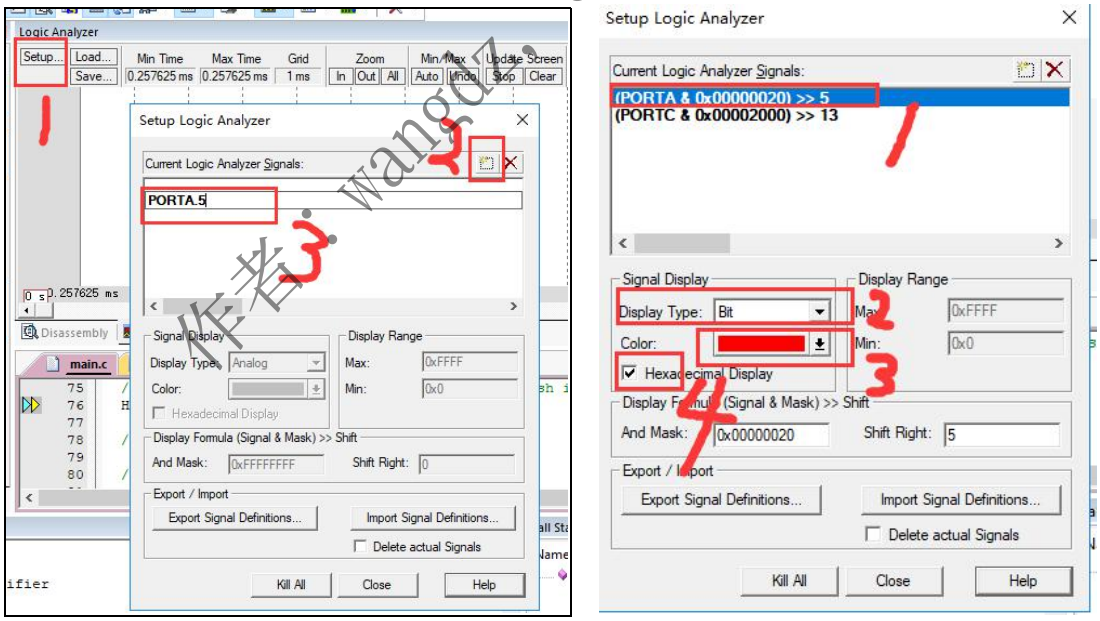

图 11 配置监控的 GPIO 端口

（4）启动 GPIO 监控

如下图所示，点击左上角的 RUN 按钮，代码就可以正常运行了，在右侧的逻辑分析一种就可以看到 PA5 端口的输出状态。绿色的线是 PC13 输入按钮的管脚。

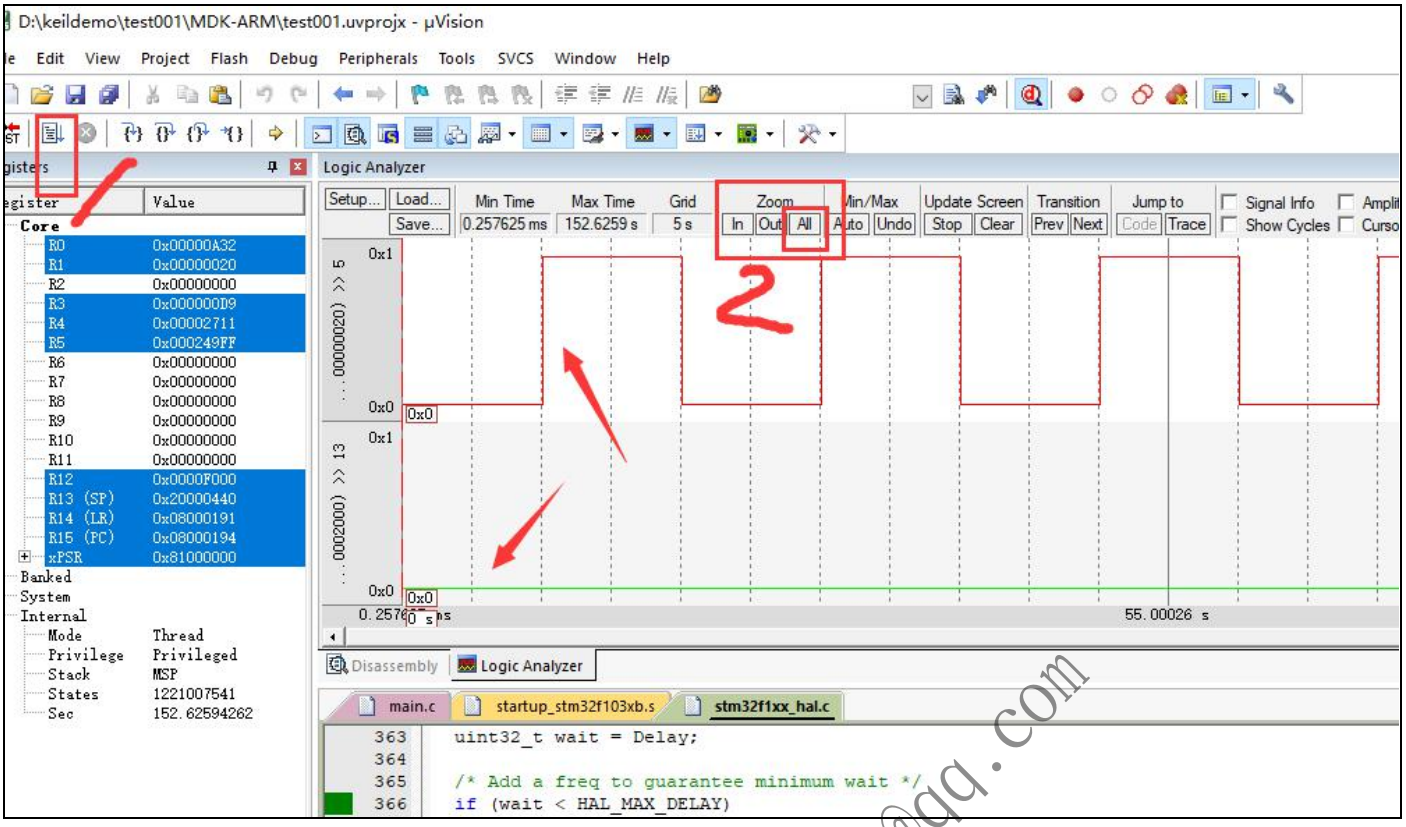

图 12 GPIO 管脚逻辑仪监测

5. GPIO 通用输入--管脚模拟与监控

（1）逻辑仪器配置

逻辑仪按 4.3 的方法配置 PC13 管脚进行监控

（2）配置模拟输入

如下图所示，调用 peripherals 菜单下的 General I/O 下的 GPIOC 窗口，点击窗口最下面的 pin 对应的右侧的管脚号，例如 PC13，就会看到上面的逻辑分析的变化，说明进行了模拟输出。

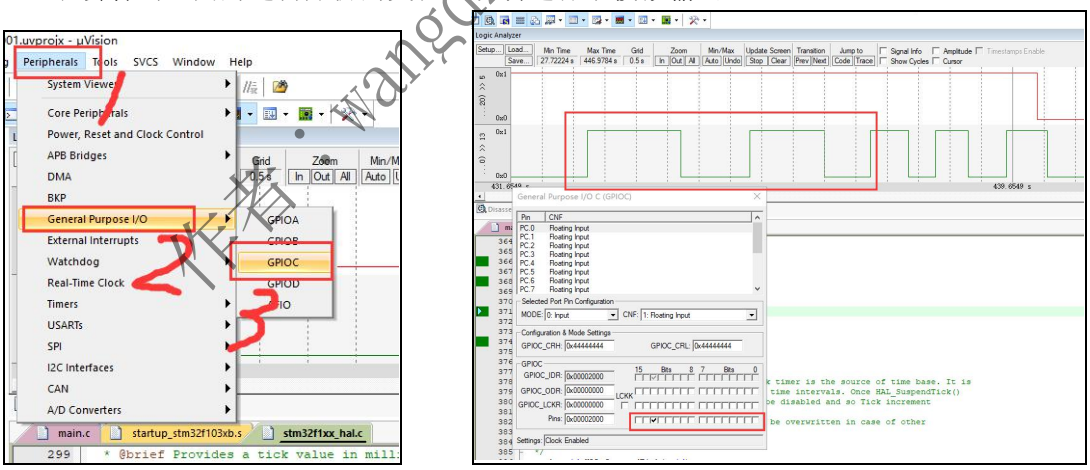

图 13 GPIO 输入及监测

（3）修改程序验证输入与输出的配合

① 在 cubmx 中修改 pc13 的功能为外部输入，如下图所示，修改两个两个地方

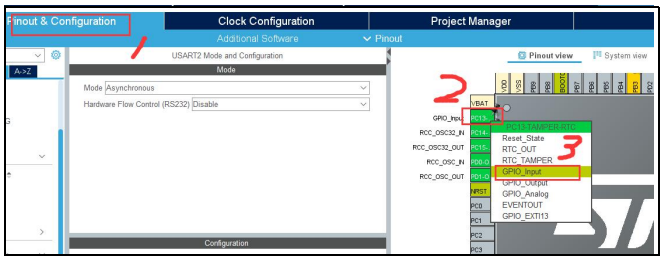

图 14 在管脚图中修改 PC13 为 input

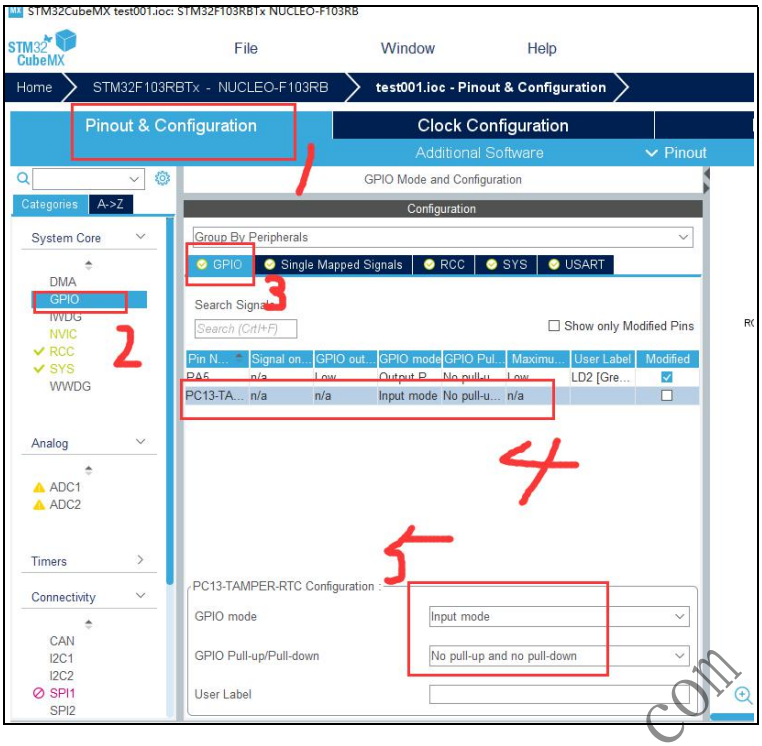

图 15 配置 PC13 管脚 input 参数

② 修改 main 程序功能，在 while 中循环检查 pc13 状态，如果输入为高电平，在 PA5 输出为低电平，如果 PC13 为底电平，则 PA5 输出高电平。在 while 循环中输入如下代码。

注意：每次在 cubemx 中重新生成 keil 工程需要重新配置仿真参数，切记。

```
int x1=HAL_GPIO_ReadPin(GPIOC,GPIO_PIN_13);
if(x1==1)
    HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5,GPIO_PIN_RESET);
else if(x1==0)
    HAL_GPIO_WritePin(GPIOA,GPIO_PIN_5,GPIO_PIN_SET);
```

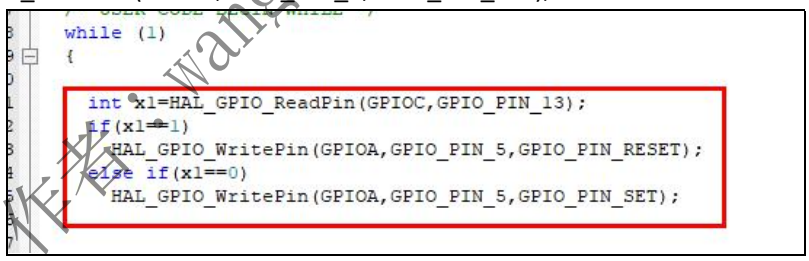

图 16 输入与输出配合代码

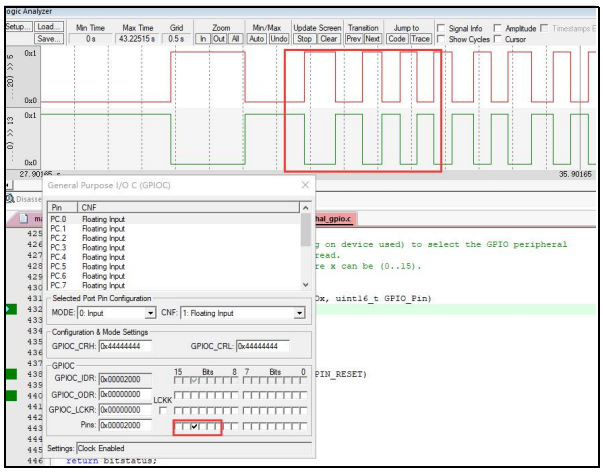

图 17 仿真结果

6. GPIO 输入外部中断---仿真模拟

（1）配置 PC13 为外部中断模式

如下图 18 所示，配置 PC13 为外部中断输入，中断采样方式为上升沿。

如图 19 所示，开启 EXTI 外部中断响应，配置优先级为 4。

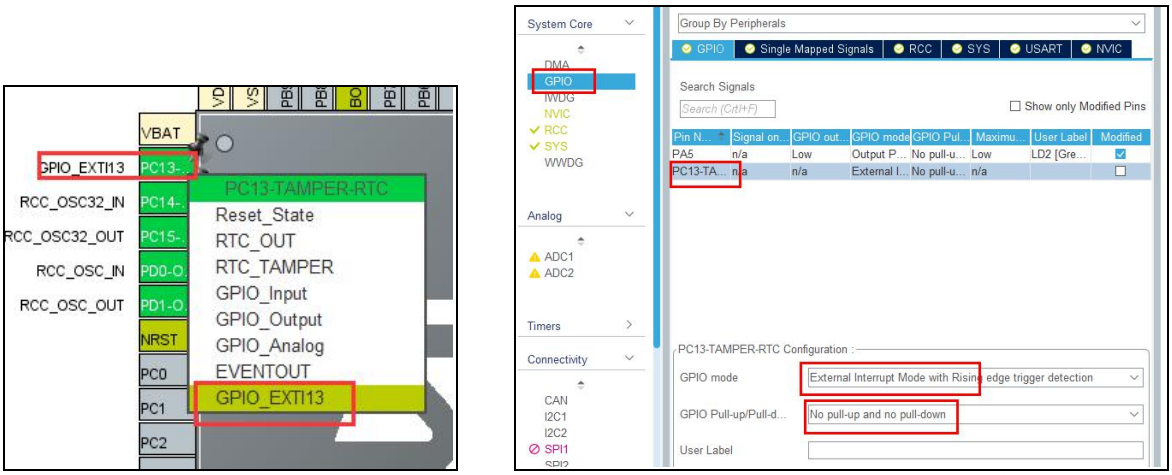

图 18 配置管脚为外部中断

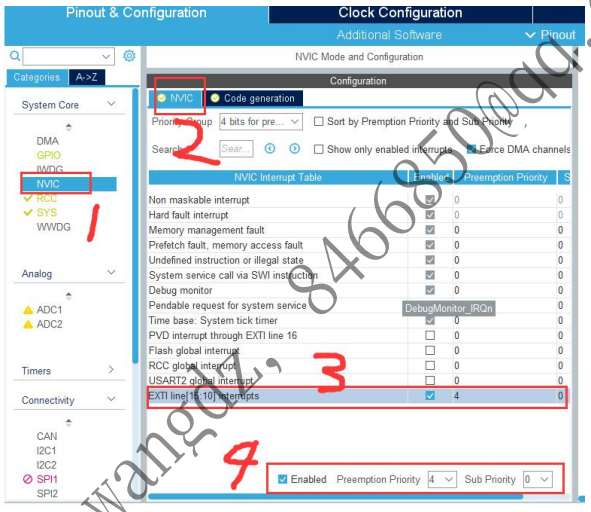

图 19 开启中断响应服务

（2）编写中断服务

① 清除 while 循环中的无用数据。

② 在 main 程序开启 PC13 对应的软中断。如下图所示

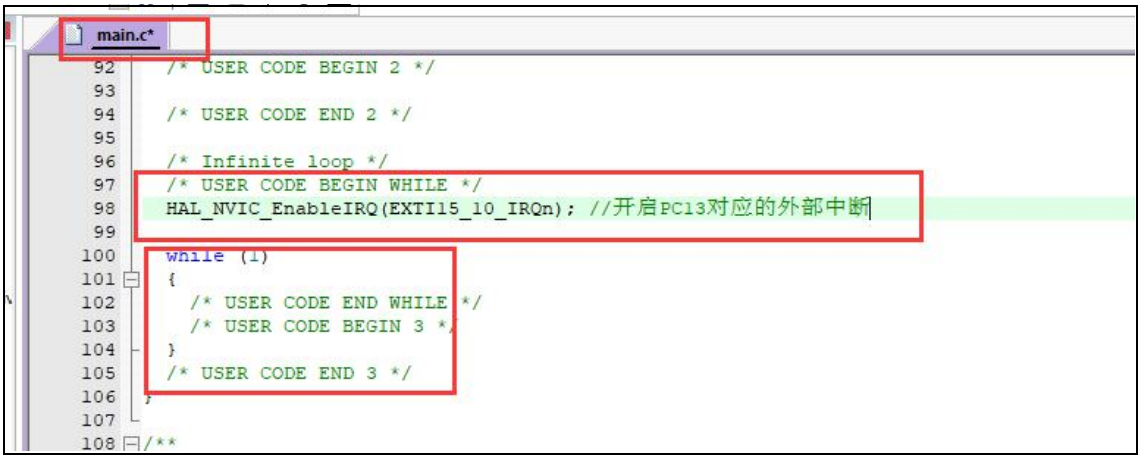

图 20 开启中断

③ 添加独立的中断响应程序，如图 20 所示。注意放的位置是 main.C 的文件中 user code 之间。

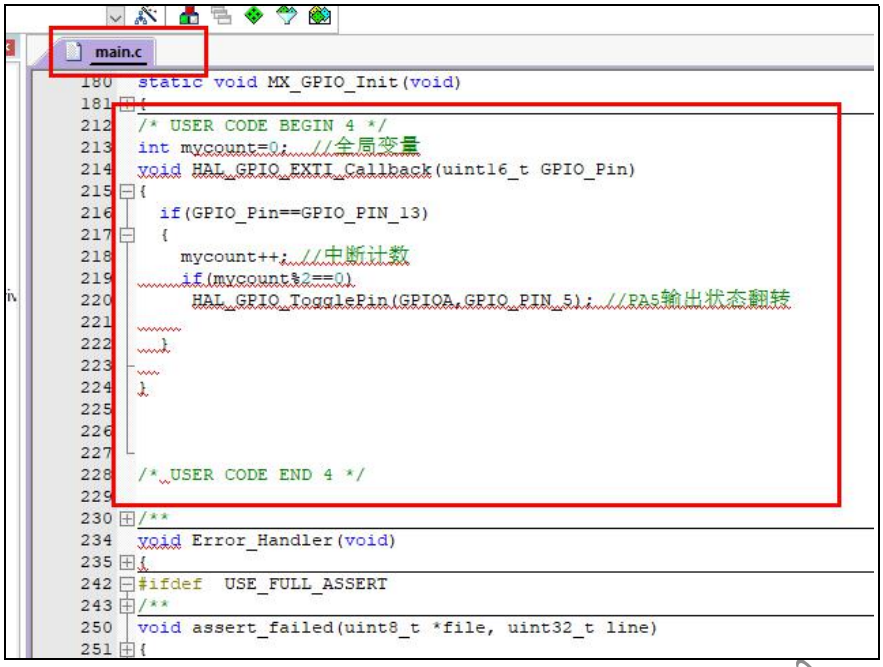

图 21 在 main.C 文件中添加中断服务程序

（3）配置仿真参数，开启仿真

每次在 cubemx 中重新生成 keil 工程需要重新配置仿真参数，切记。

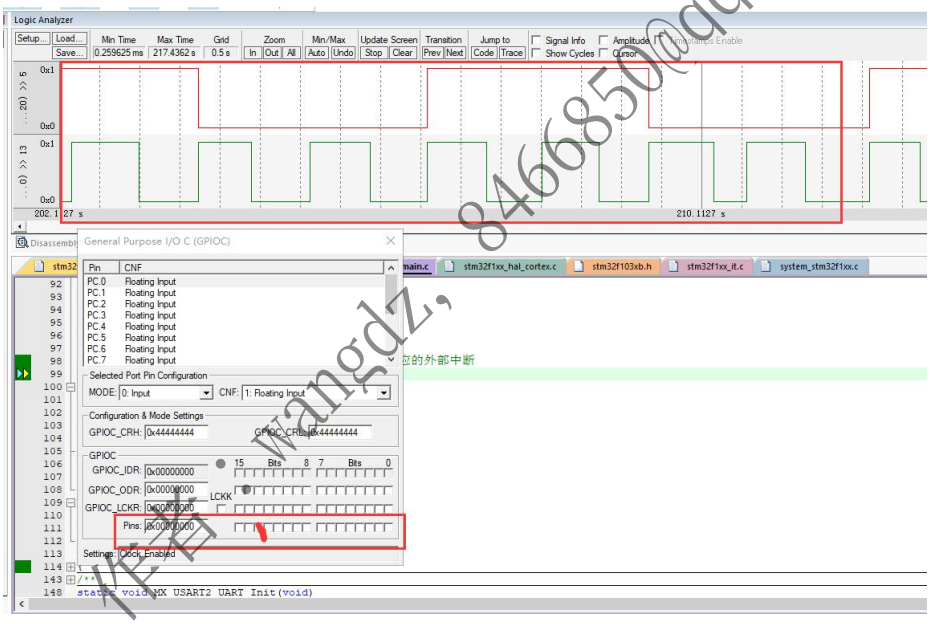

图 22 外部中断仿真

7. 定时器模拟仿真

（1）在 cubeMX 中配置定时器。

思路：

① 首先检查系统的时钟，确定定时器的主频时钟。

② 开启定时内部时钟，配置计时参数。

③ 开启定时器中断

④ 生成 keil 代码

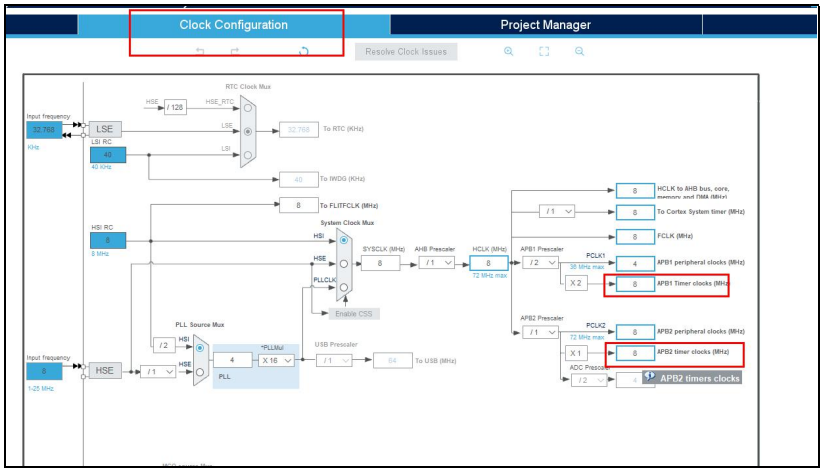

图 23 检查时钟总线

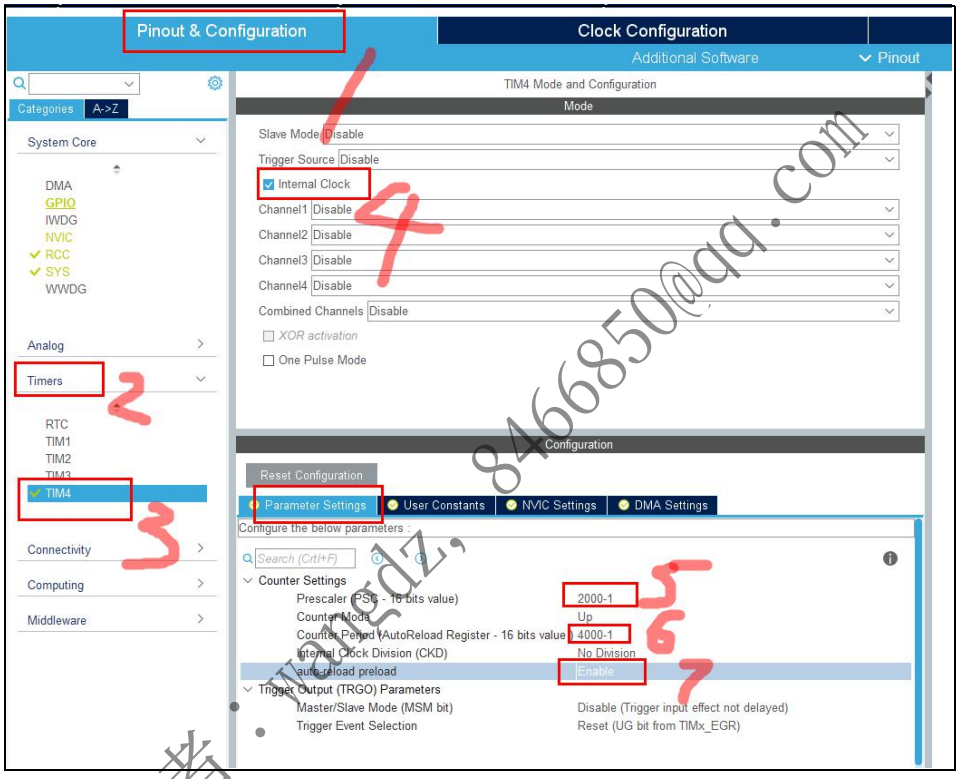

图 24 配置定时时间参数，1 秒一次

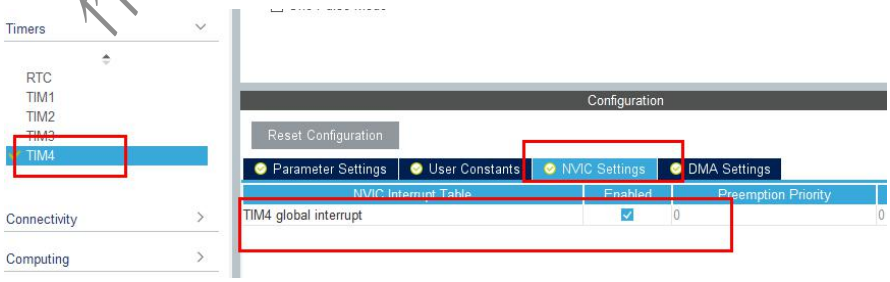

图 25 开启定时中断

（2）在 main.c 中启动定时器

在 main.C 中开启定时器中断服务。

```
main.c
101   /* USER CODE BEGIN WHILE */
102   HAL_NVIC_EnableIRQ(EXTI15_10_IRQn); //开启PC13对应的外部中断
103   HAL_UART_Receive_IT(&huart2,myUart2ReBuf,1); //开启串口接收，接收1个字符产生中断
104
105   HAL_TIM_Base_Start_IT(&htim4); //开启定时器4
106   //HAL_TIM_Base_Stop_IT(&htim4); //关闭定时器4
107   while (1)
108   {
109       HAL_UART_Transmit(&huart2,"1234\r\n",6,1000);//循环发送数据
110       HAL_Delay(3000);
111     /* USER CODE END WHILE */
112
113     /* USER CODE BEGIN 3 */
114   }
```

图 26 开启定时器 4

（3）添加定时器中断服务程序。

```
294
295  //TIM定时器中断服务程序
296  void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
297  {
298
299    if(htim==&htim4)
300    {
301      HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_5); //PA5输出状态翻转
302
303    }
304
305  }
306
307
308
309  /* USER CODE END 4 */
310
```

图 27 添加定时中断服务程序

（4）启动仿真

按前面配置就可以了，利用逻辑仪查看在定时器服务程序中控制的输出端口 PA5 状态。从图中可以看出，每 1 秒钟 PA5 的状态翻转一次。

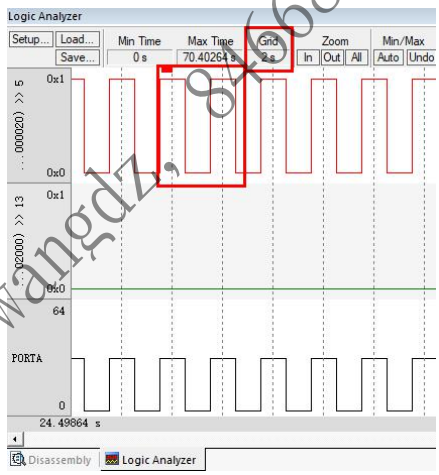

图 28 定时器仿真结果

8. 串口功能仿真

（1）安装虚拟串口工具

下载工具 https://dl.pconline.com.cn/download/825163.html

并进行安装，虚拟两个串口。这两个串口内部是联通的。就是 COM8 的输入发送给 COM9，COM9 的输入发送给 COM8。

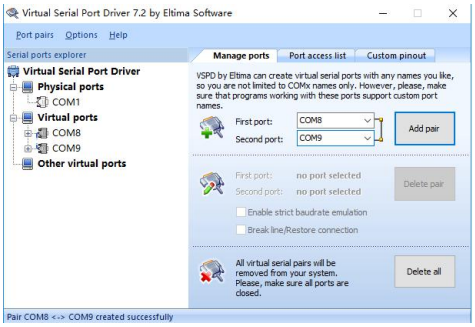

图 29 虚拟串口安装

（2）安装串口调试工具

找一个通用的串口工具，网上很多，推荐一个　串口调试助手。测试一下虚拟串口

https://dl.pconline.com.cn/download/525064.html

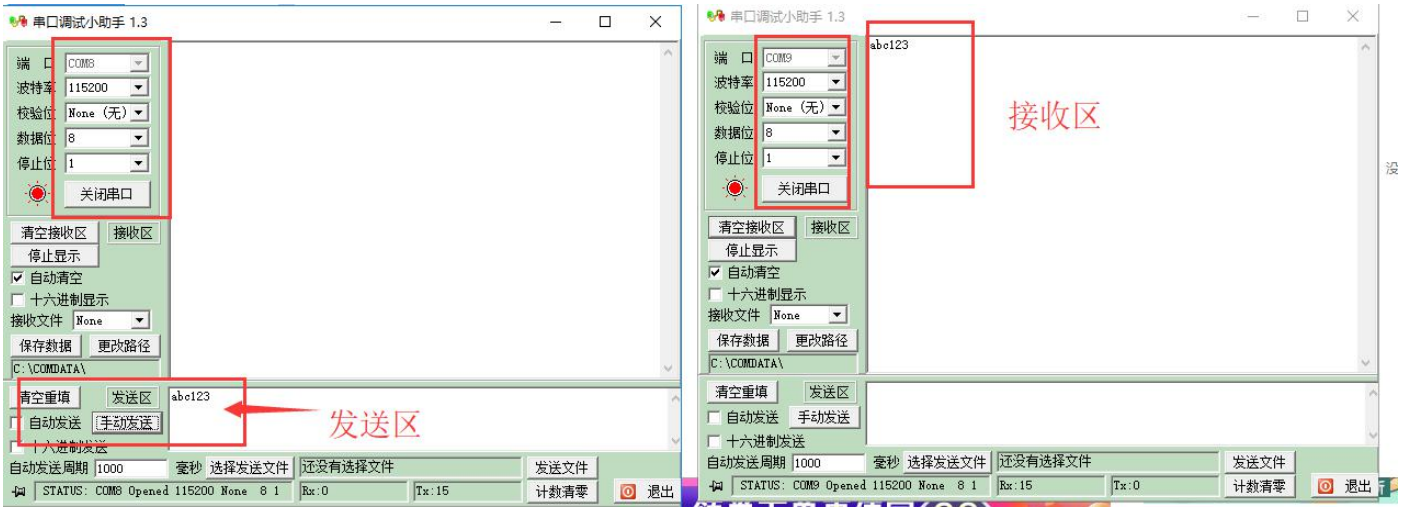

图 30 虚拟串口测试

（3）检查 STM32F103 的串口配置

在 cubmx 中已经进行了串口 2 的配置。开启串口的接收中断功能。就是串口发送采用拥塞模式，串口接收采用中断模式，这是大量工程应用中使用的模式。串口还有很多模式，都可以设置模拟。

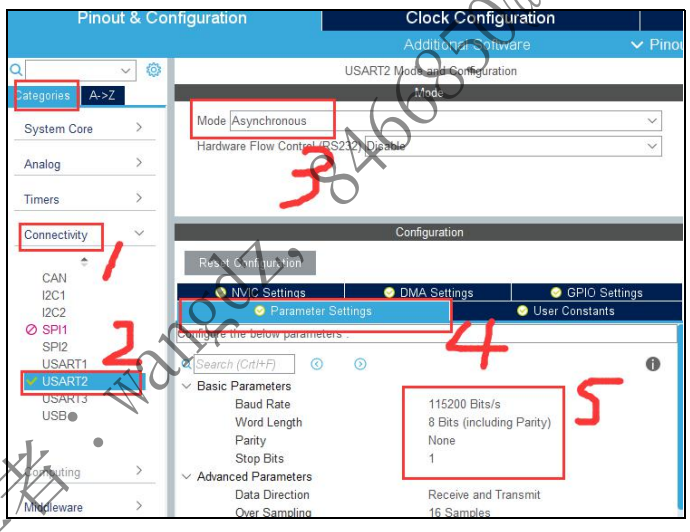

图 31 uart2 串口配置参数

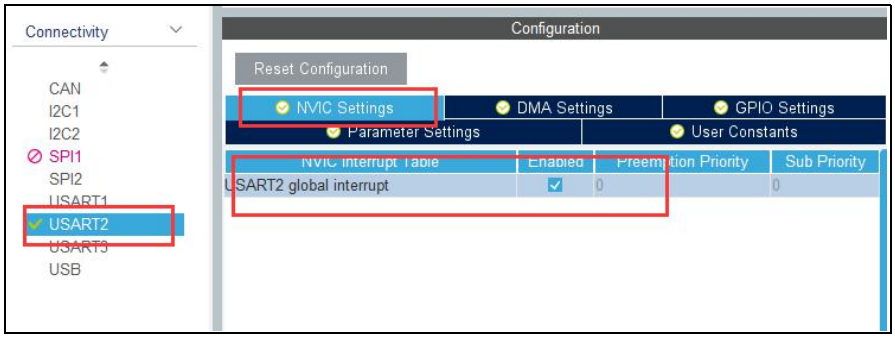

图 32 开启 UART2 串口中断

（4）修改 main 主程序

思路：定义一个全局数组 myUart2ReBuf 用来接收串口数据。每接收 1 个字符产生一次接收中断，然后进行处理，准备接收下一个字符。在 main 程序的 while 循环中，循环发送字符串数据。

程序如图 27 到 29 所示。

下面是用到的部分代码。

```
uint8_t myUart2ReBuf[100];
HAL_UART_Receive_IT(&huart2,myUart2ReBuf,1); //
HAL_UART_Transmit(&huart2,"1234\r\n",6,1000);

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
{
    if(huart==&huart2)
    {
        HAL_UART_Receive_IT(&huart2,myUart2ReBuf,1);
        uint8_t mydata=myUart2ReBuf[0];
        HAL_UART_Transmit(&huart2,&mydata,1,1000);
        HAL_GPIO_TogglePin(GPIOA,GPIO_PIN_5);
    }
}
```

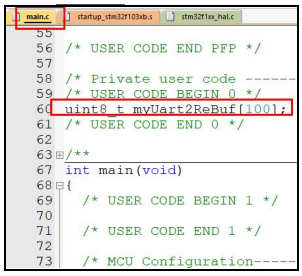

图 33 定义全局数组

图 34 main 程序代码

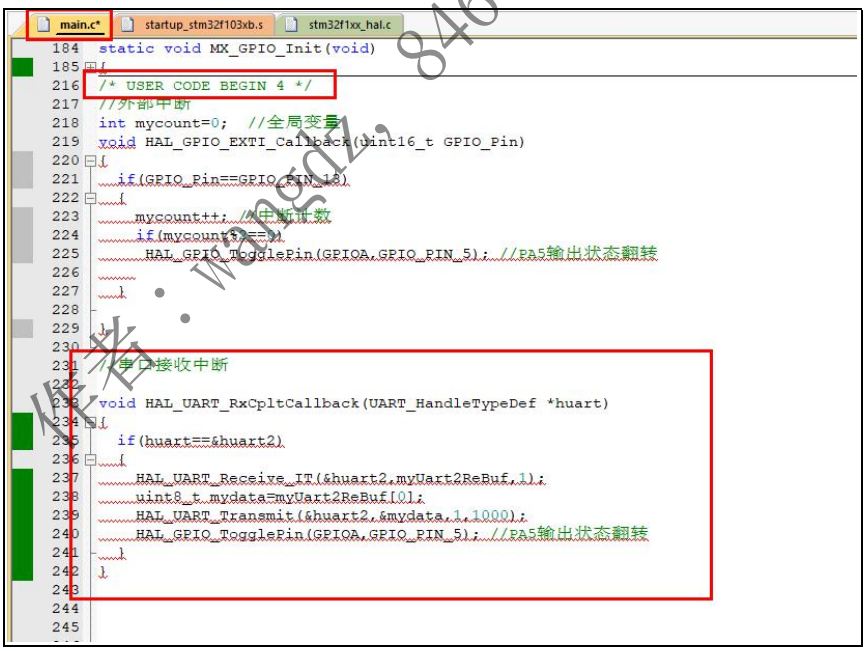

图 35 添加串口接收中断服务程序

（5）MCU 串口与虚拟绑定

在 MCU 中用的 UART2，需要绑定在刚才产生的虚拟串口 COM8 上。

首先启动 debug 调试模式，在此模式下，在左下窗口输入如下两条命令

MODE COM8　115200,0,8,1　　//启动串口 8，波特率为 115200，无奇偶校验，8 个数字为，1 个停止位

ASSIGN COM8　<S2IN>S2OUT　　//COM8 绑定在串口 2 的输入和输出，S2 代表串口 2.

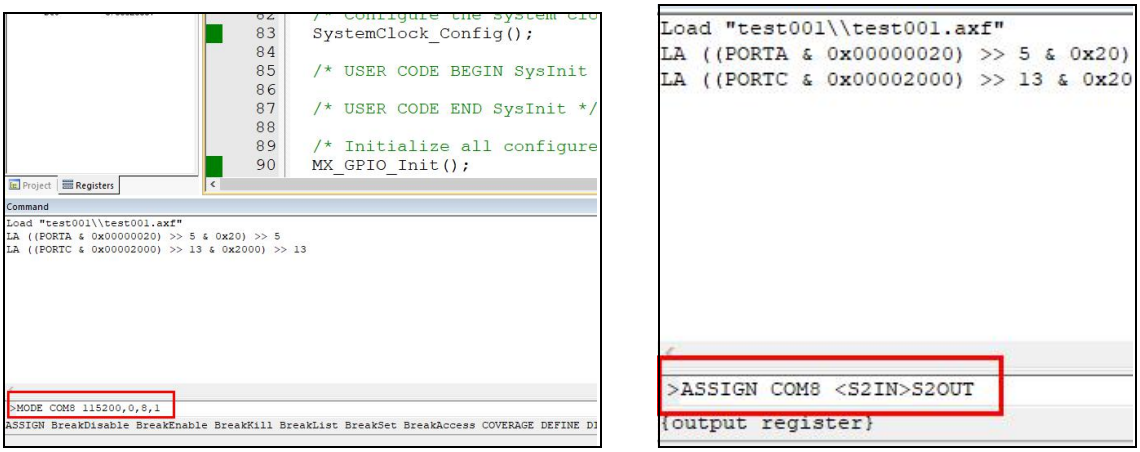

图 36 在调试下启动串口绑定模式

（6）仿真开始

利用 RUN 按钮，开始调试模式。然后利用串口助手开启 COM9，COM9 和 COM8 是互通的，com8 跟 UART2 也是绑定的，因此 COM9 就是跟 UART2 联通的。Com9 就接收到了 uart2 发送的数据，并能给 UART2 发送数据，效果如下图所示。

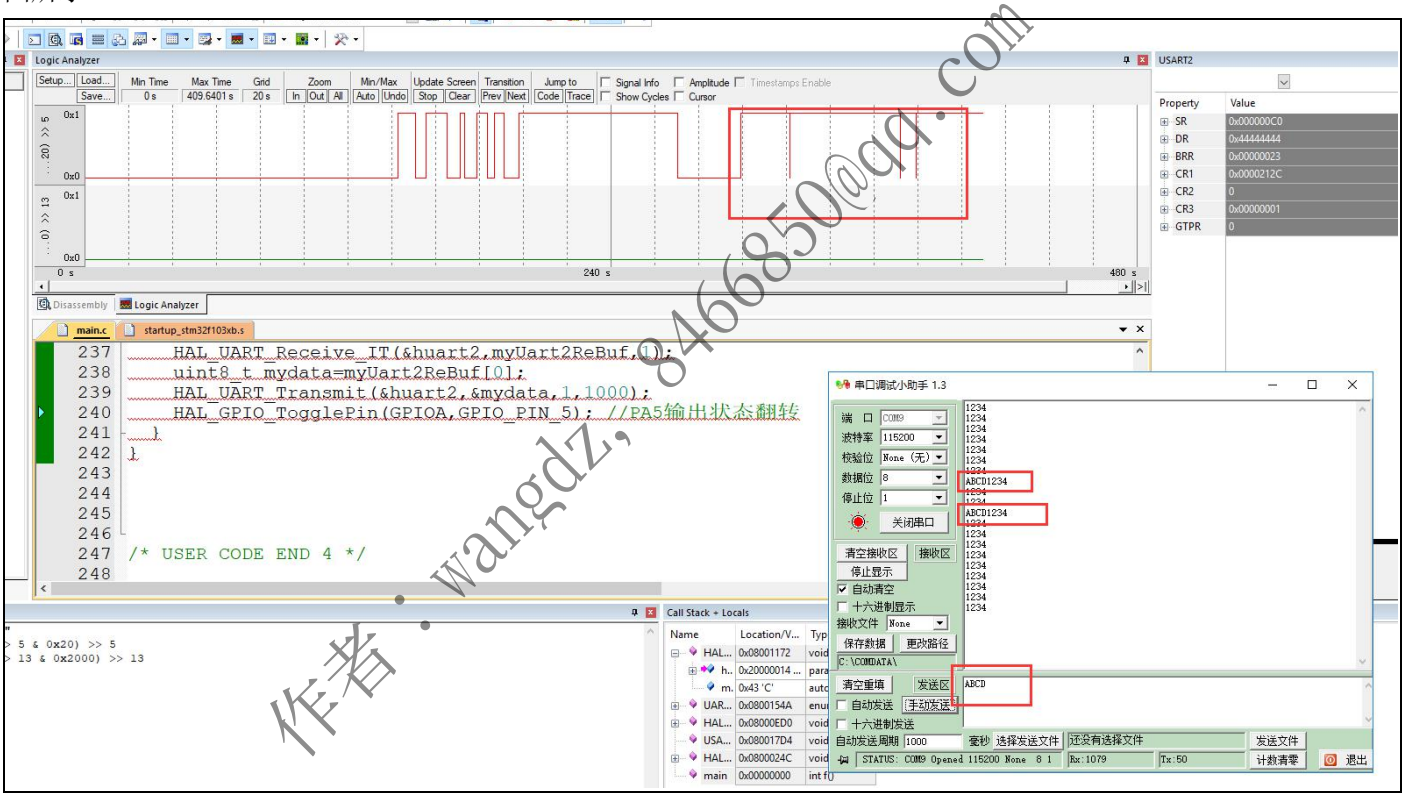

图 37 虚拟串口调试效果

9. 不能用于 AD/DA 的软件模拟仿真