零死角玩转STM32

与野火同行 乐意惬无边

1 原创教程，完全开源。

2 通俗易懂，详尽解读。

3 由浅入深，结合实操。

4 配套板子，全面玩转。

5 强强联合，不断更新。

野火团队 Wild Fire Team

0、友情提示

《零死角玩转 STM32》系列教程由初级篇、中级篇、高级篇、系统篇、四个部分组成，根据野火 STM32 开发板旧版教程升级而来，且经过重新深入编写，重新排版，更适合初学者，步步为营，从入门到精通，从裸奔到系统，让您零死角玩转 STM32。M3 的世界，与野火同行，乐意惬无边。

另外，野火团队历时一年精心打造的《STM32 库开发实战指南》将于今年 10 月份由机械工业出版社出版，该书的排版更适于纸质书本阅读以及更有利于查阅资料。内容上会给你带来更多的惊喜。是一本学习 STM32 必备的工具书。敬请期待！

7、PWM（软件仿真）

7.1 实验描述及工程文件清单

实验描述	通用定时器 TIM3 产生 4 路不同占空比的 PWM 波。 TIM3 Channel1 duty cycle = (TIM3_CCR1/ TIM3_ARR)* 100 = 50% TIM3 Channel2 duty cycle = (TIM3_CCR2/ TIM3_ARR)* 100 = 37.5% TIM3 Channel3 duty cycle = (TIM3_CCR3/ TIM3_ARR)* 100 = 25% TIM3 Channel4 duty cycle = (TIM3_CCR4/ TIM3_ARR)* 100 = 12.5%
硬件连接	PA.06: (TIM3_CH1) PA.07: (TIM3_CH2) PB.00: (TIM3_CH3) PB.01: (TIM3_CH4)
用到的库文件	startup/start_stm32f10x_hd.cCMSIS/core_cm3.c CMSIS/system_stm32f10x.c FWlib/stm32f10x_gpio.c FWlib/stm32f10x_rcc.c FWlib/stm32f10x_flash.c FWlib/stm32f10x_tim.c
用户编写的文件	USER/main.c USER/stm32f10x_it.c USER/pwm_output.c

7.2 STM32 通用定时器简介

STM32 总共有 8 个定时器，TIM1 和 TIM8 是 16 位的高级定时器，TIM2、TIM3、TIM4、TIM5 是通用定时器。本实验中只是讲解通用定时器 TIM3，利用 TIM3 产生 4 路不同占空比的方波。

7.3 代码分析

首先我们需在工程中添加我们需要用到的库文件，有关库文件的配置参考前面的教程，这里不再详述。

接下来我们从 main 函数讲起：

```
1.  /*
2.   * 函数名：main
3.   * 描述  ：主函数
4.   * 输入  ：无
5.   * 输出  ：无
6.   */
7.  int main(void)
8.  {
9.    /* 配置系统时钟为 72M */
10.     SystemInit();
11.     /* TIM3 PWM 波输出初始化，并使能 TIM3 PWM 输出 */
12.     TIM3_PWM_Init();
13.
14.   while (1)
15.   {}
16. }
```

进入 mian 函数我们首先调用库函数 `SystemInit();`将我们的系统时钟配置为 72MHZ。有关库函数 `SystemInit();`的讲解请参考前面的教程。

函数用于初始化 TIM3 的 PWM 信号 I/O，配置 PWM 信号的模式，如周期、极性、占空比等。`TIM3_PWM_Init();`由我们用户在 pwm_output.c 中实现：

```
/*
 * 函数名：TIM3_Mode_Config
 * 描述  ：TIM3 输出 PWM 信号初始化，只要调用这个函数
 *         TIM3 的四个通道就会有 PWM 信号输出
 * 输入  ：无
 * 输出  ：无
 * 调用  ：外部调用
 */
void TIM3_PWM_Init(void)
{
    TIM3_GPIO_Config();
    TIM3_Mode_Config();
}
```

其中用来 TIM3_GPIO_Config();配置 GPIO，代码很简单，TIM3_Mode_Config();用来配置 PWM 信号的模式，详细代码如下，主要做了如下工作：

1->设定 TIM 信号周期

2->设定 TIM 预分频值

3->设定 TIM 分频系数

4->设定 TIM 计数模式

5->根据 TIM_TimeBaseInitStruct 这个结构体里面的值初始化 TIM

6->设定 TIM 的 OC 模式

7->TIM 输出使能

8->设定电平跳变值

9->设定 PWM 信号的极性

10->使能 TIM 信号通道

11->使能 TIM 重载寄存器 CCRX

12->使能 TIM 重载寄存器 ARR

13->使能 TIM 计数器

```
/*
 * 函数名：TIM3_Mode_Config
 * 描述  ：配置 TIM3 输出的 PWM 信号的模式，如周期、极性、占空比
 * 输入  ：无
```

```
  * 输出  : 无
  * 调用  : 内部调用
  */
static void TIM3_Mode_Config(void)
{
    TIM TimeBaseInitTypeDef  TIM TimeBaseStructure;
    TIM_OCInitTypeDef  TIM_OCInitStructure;

    /* PWM 信号电平跳变值 */
    u16 CCR1_Val = 500;
    u16 CCR2_Val = 375;
    u16 CCR3 Val = 250;
    u16 CCR4 Val = 125;

/* -----------------------------------------------------------------------
    TIM3 Configuration: generate 4 PWM signals with 4 different duty cycles:
    TIM3CLK = 36 MHz, Prescaler = 0x0, TIM3 counter clock = 36 MHz
    TIM3 ARR Register = 999 => TIM3 Frequency = TIM3 counter clock/(ARR + 1)
    TIM3 Frequency = 36 KHz.
    TIM3 Channel1 duty cycle = (TIM3 CCR1/ TIM3 ARR)* 100 = 50%
    TIM3 Channel2 duty cycle = (TIM3_CCR2/ TIM3_ARR)* 100 = 37.5%
    TIM3 Channel3 duty cycle = (TIM3_CCR3/ TIM3_ARR)* 100 = 25%
    TIM3 Channel4 duty cycle = (TIM3 CCR4/ TIM3 ARR)* 100 = 12.5%
  ----------------------------------------------------------------------- */

  /* Time base configuration */
 //当定时器从 0 计数到 999，即为 1000 次，为一个定时周期
  TIM_TimeBaseStructure.TIM_Period = 999;

   //设置预分频：不预分频，即为 36MHz
  TIM_TimeBaseStructure.TIM_Prescaler = 0;

  TIM_TimeBaseStructure.TIM_ClockDivision = 0;  //设置时钟分频系数：不分频
  TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up; //向上计数模式

  TIM TimeBaseInit(TIM3, &TIM TimeBaseStructure);

  /* PWM1 Mode configuration: Channel1 */
  TIM_OCInitStructure.TIM_OCMode = TIM_OCMode_PWM1;   //配置为 PWM 模式 1
  TIM_OCInitStructure.TIM_OutputState = TIM_OutputState_Enable;

  //设置跳变值，当计数器计数到这个值时，电平发生跳变
  TIM OCInitStructure.TIM Pulse = CCR1 Val;

  //当定时器计数值小于 CCR1_Val 时为高电平
  TIM OCInitStructure.TIM OCPolarity = TIM OCPolarity High;

  TIM_OC1Init(TIM3, &TIM_OCInitStructure);   //使能通道 1
  TIM_OC1PreloadConfig(TIM3, TIM_OCPreload_Enable);

  /* PWM1 Mode configuration: Channel2 */
  TIM OCInitStructure.TIM OutputState = TIM OutputState Enable;

  //设置通道 2 的电平跳变值，输出另外一个占空比的 PWM
  TIM OCInitStructure.TIM Pulse = CCR2 Val;

  TIM_OC2Init(TIM3, &TIM_OCInitStructure);    //使能通道 2
  TIM_OC2PreloadConfig(TIM3, TIM_OCPreload_Enable);

  /* PWM1 Mode configuration: Channel3 */
  TIM OCInitStructure.TIM OutputState = TIM OutputState Enable;

 //设置通道 3 的电平跳变值，输出另外一个占空比的 PWM
  TIM_OCInitStructure.TIM_Pulse = CCR3_Val;

  TIM_OC3Init(TIM3, &TIM_OCInitStructure);   //使能通道 3
  TIM_OC3PreloadConfig(TIM3, TIM_OCPreload_Enable);

  /* PWM1 Mode configuration: Channel4 */
  TIM OCInitStructure.TIM OutputState = TIM OutputState Enable;

//设置通道 4 的电平跳变值，输出另外一个占空比的 PWM
  TIM_OCInitStructure.TIM_Pulse = CCR4_Val;
  TIM_OC4Init(TIM3, &TIM_OCInitStructure);  //使能通道 4
```

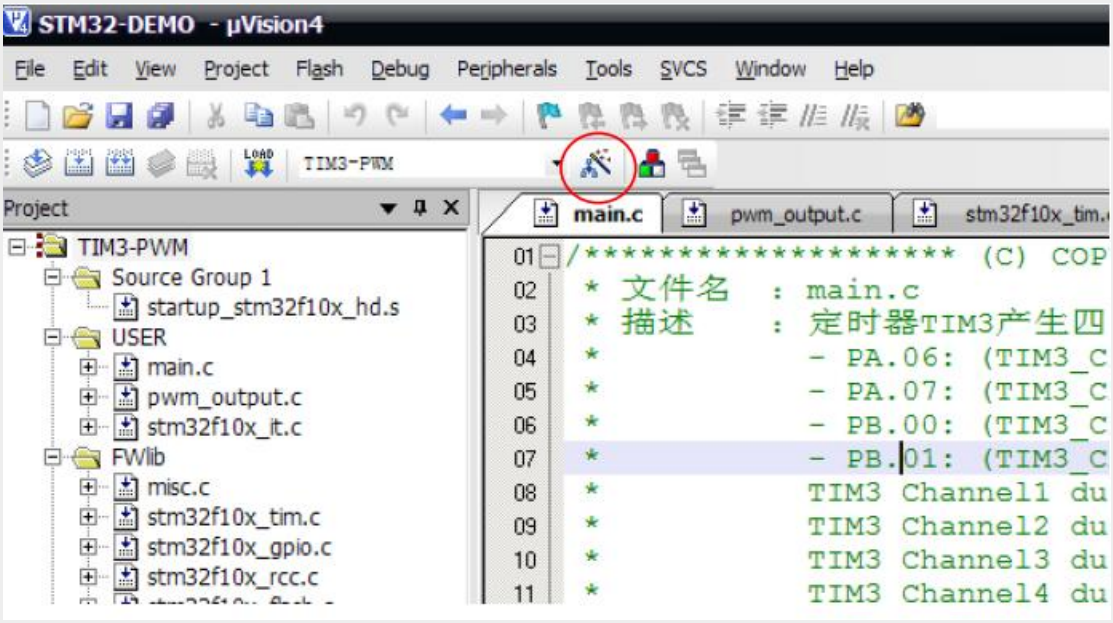
```
79.   TIM_OC4PreloadConfig(TIM3, TIM_OCPreload_Enable);
80.
81.   TIM_ARRPreloadConfig(TIM3, ENABLE);
82.
83.   /* TIM3 enable counter */
84.   TIM_Cmd(TIM3, ENABLE);   //使能定时器 3
85. }
```

现在，TIM3 的通道 1(PA.06)、2(PA.07)、3(PB.00)、4(PB.01)就会输出不同占空比的 PWM 信号了。PWM 信号可以通过示波器看到。考虑到并不是每个用户手头上都有示波器，我们在这里采用软件仿真的方式来验证我们的程序。

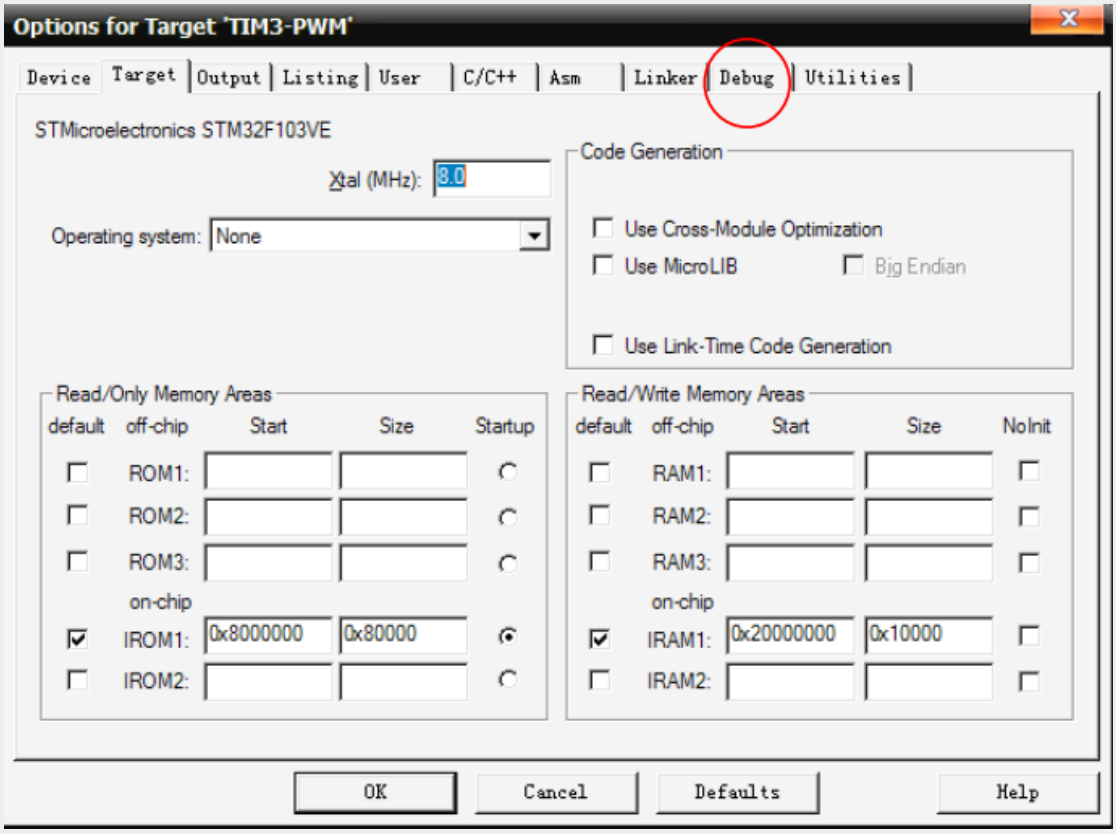
7.4 软件仿真

以前我们都是通过 JLINK 直接将我们的代码烧到开发板的 flash 中去调试，现在要换成软件仿真，得首先设置下我们的开发环境，按照如下步骤所示：

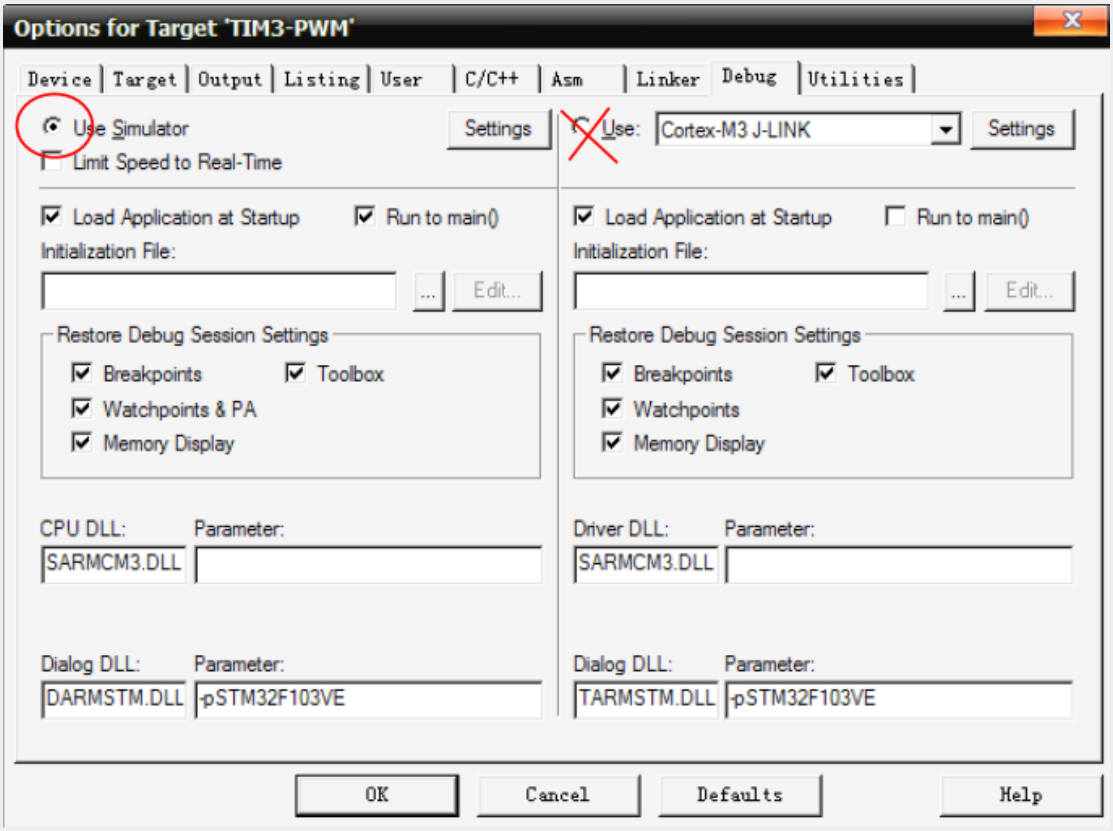
1、点击 Target Options...选项。



2、选中 Debug 选项卡。

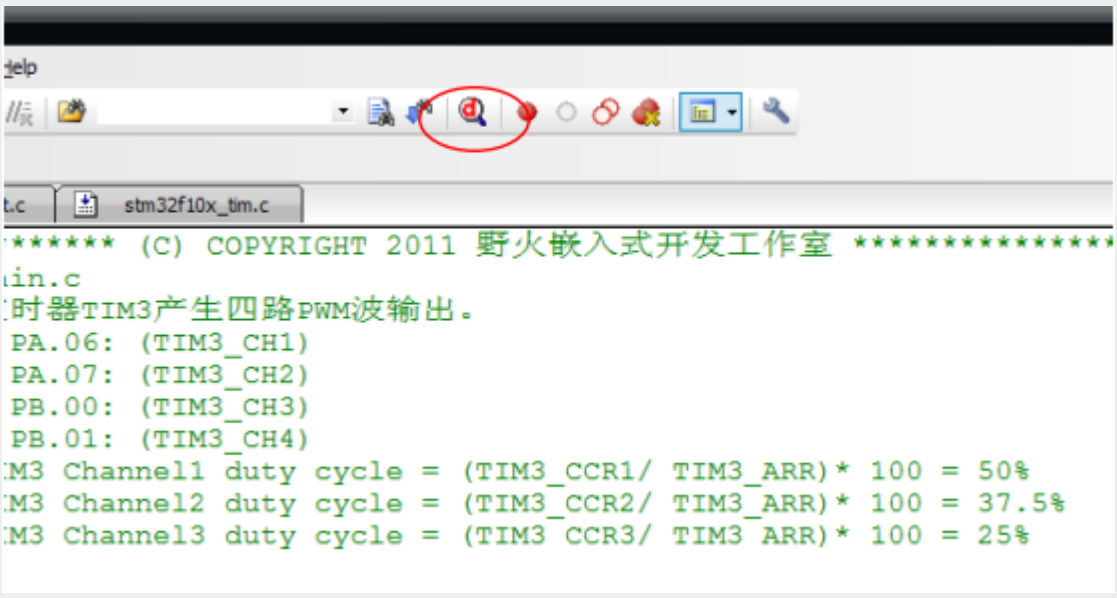


3、选中 Use Simulator 选项，然后点击 OK 即可。

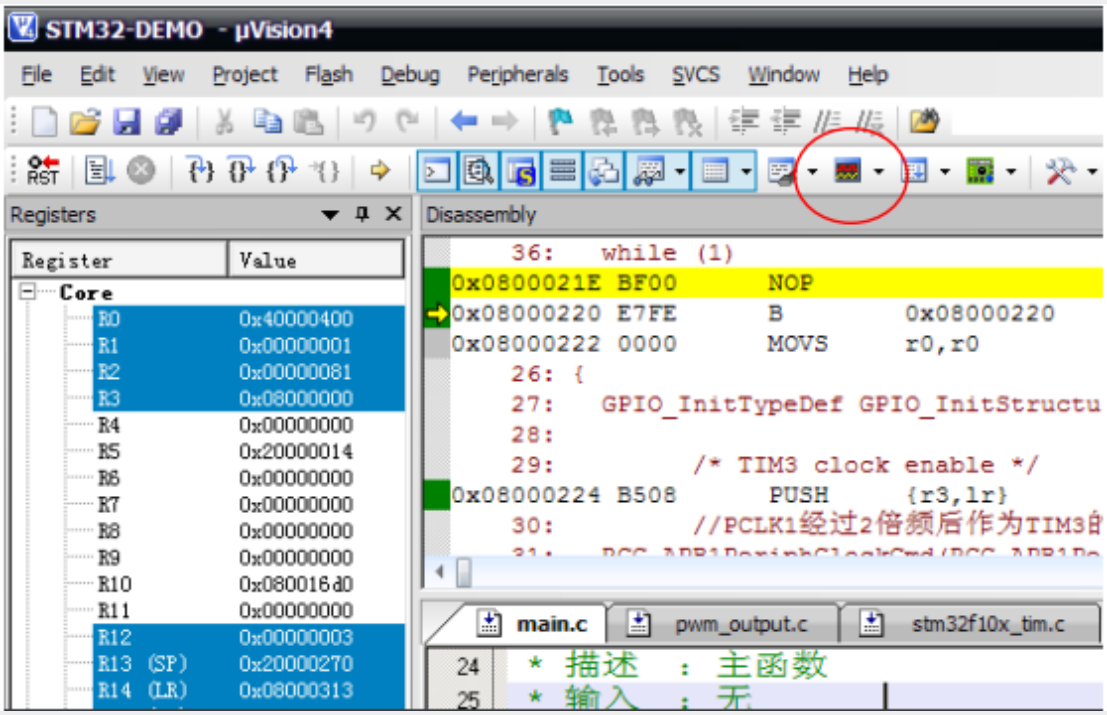


下面我们开始进行软件仿真，按照如下步骤进行：

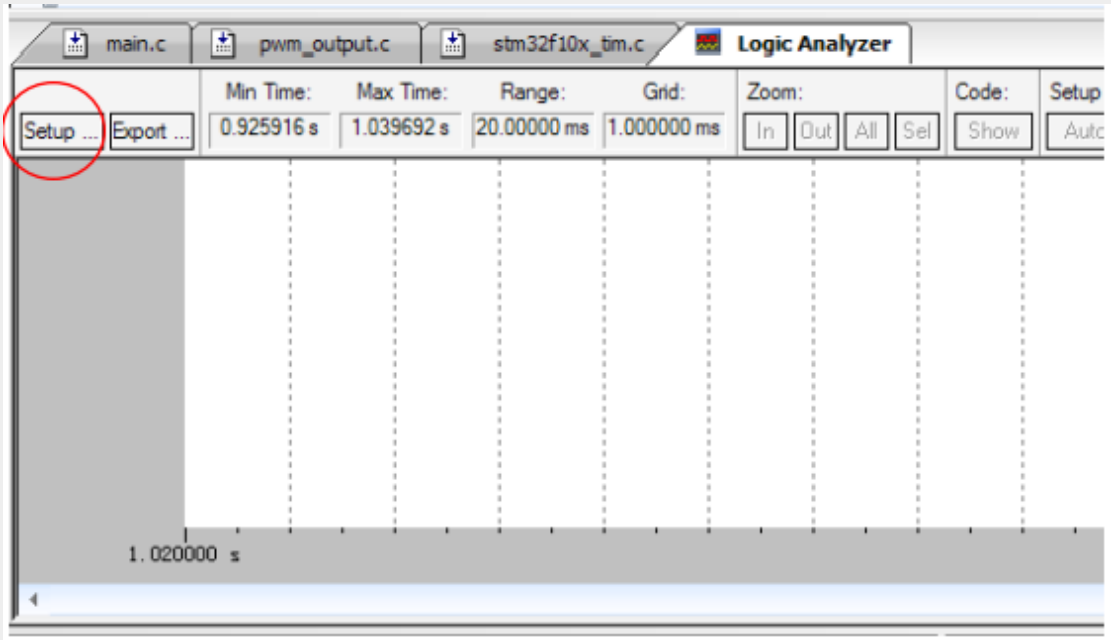
1. 1-> 点击 Start/Stop Debug Session 选项。



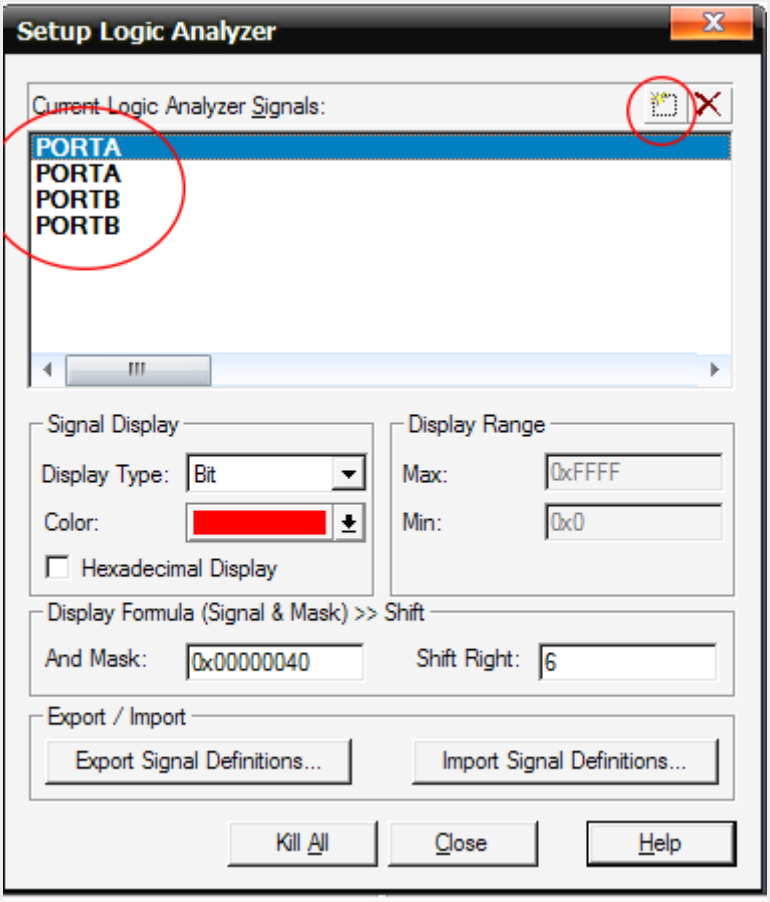
2. 2->点击 Analysis Windows 选项。



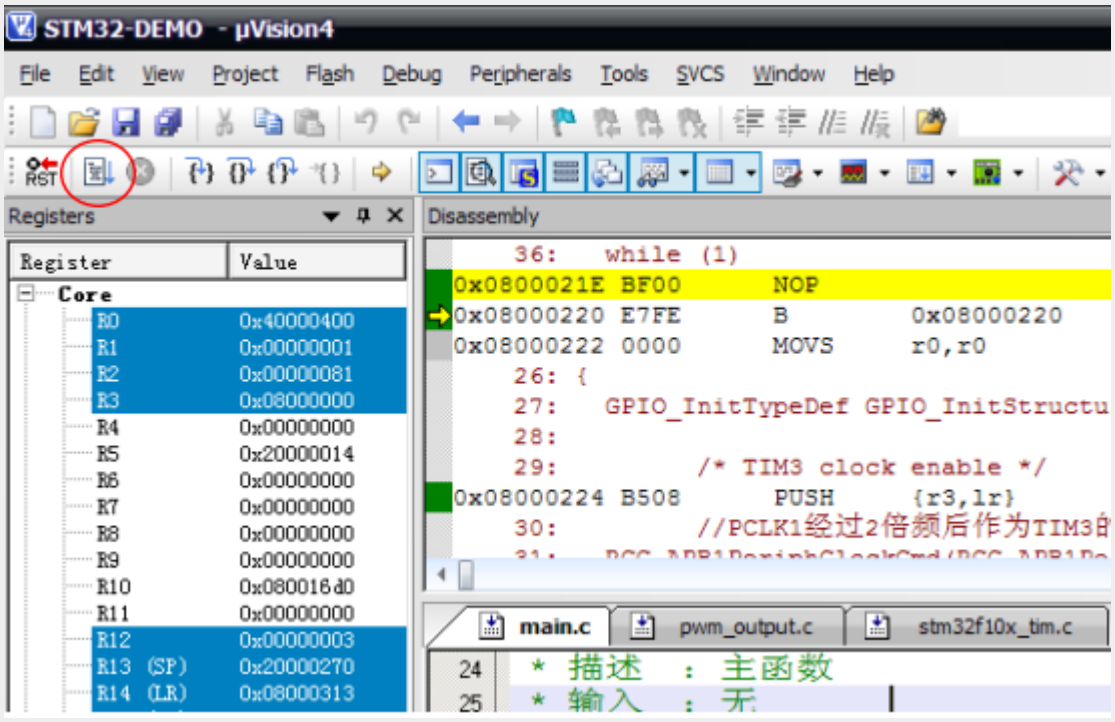
3. 3->点击 Setup... 选项卡。



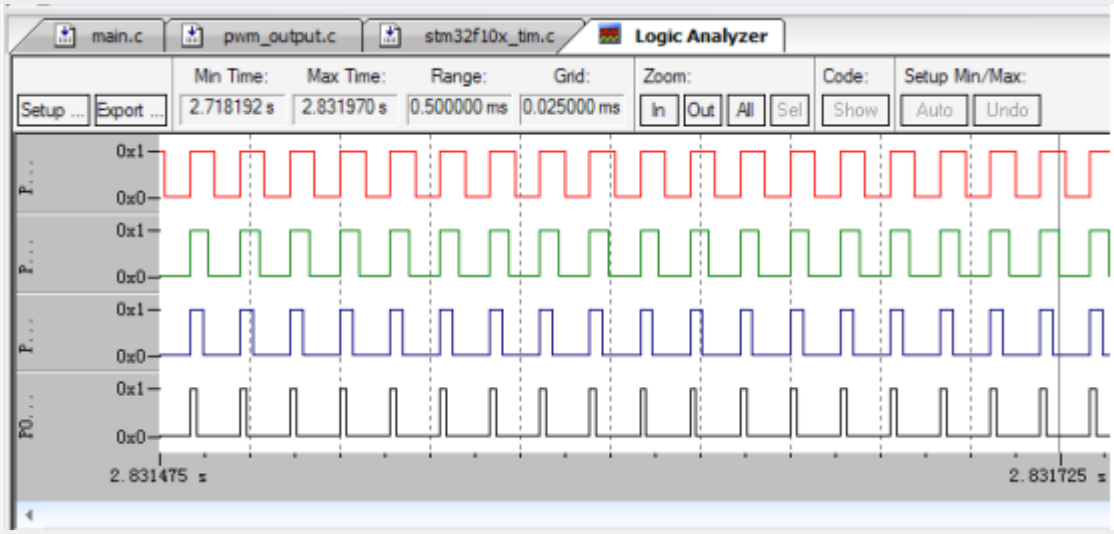
4. 4->点击 NEW(Insert)，在下面的文本框中输入 TMI3 的 PWM 通道，这里分别是：PORTA.6、PORTA.7、PORTB.0、PORTB.1。然后点击 Close。



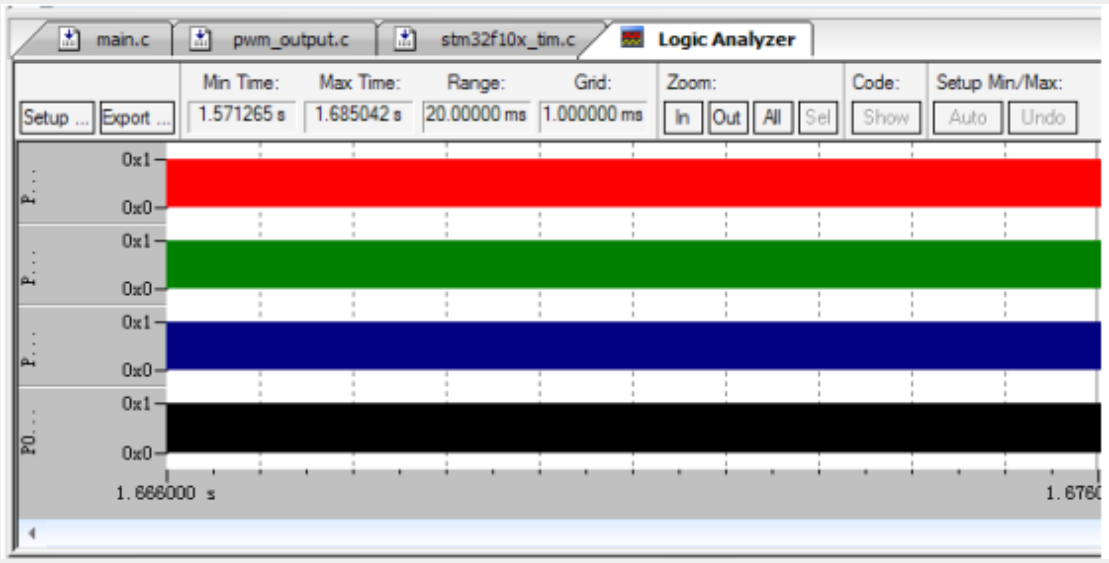
5. 5->点击运行。



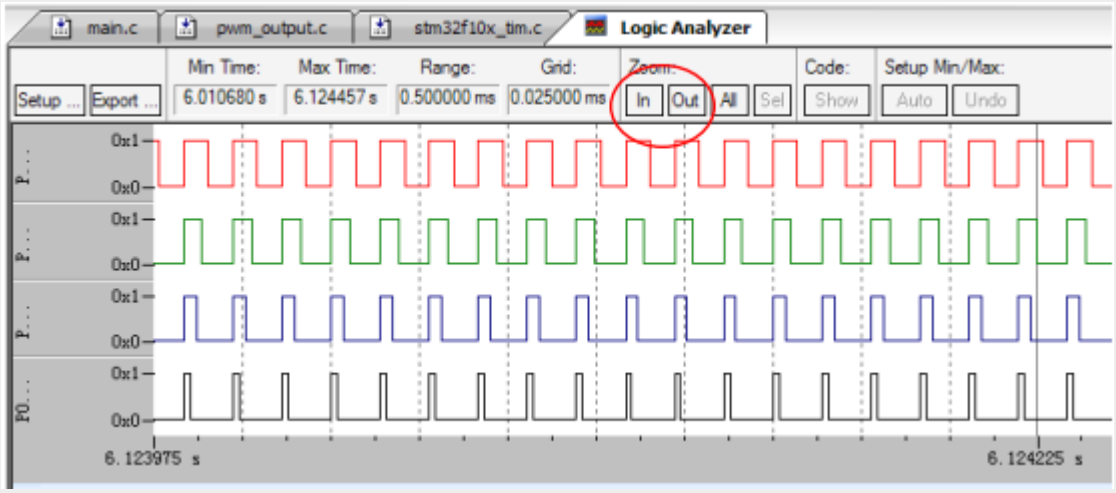
6. 6->这时候正常的话则是出现下面的 PWM 信号。



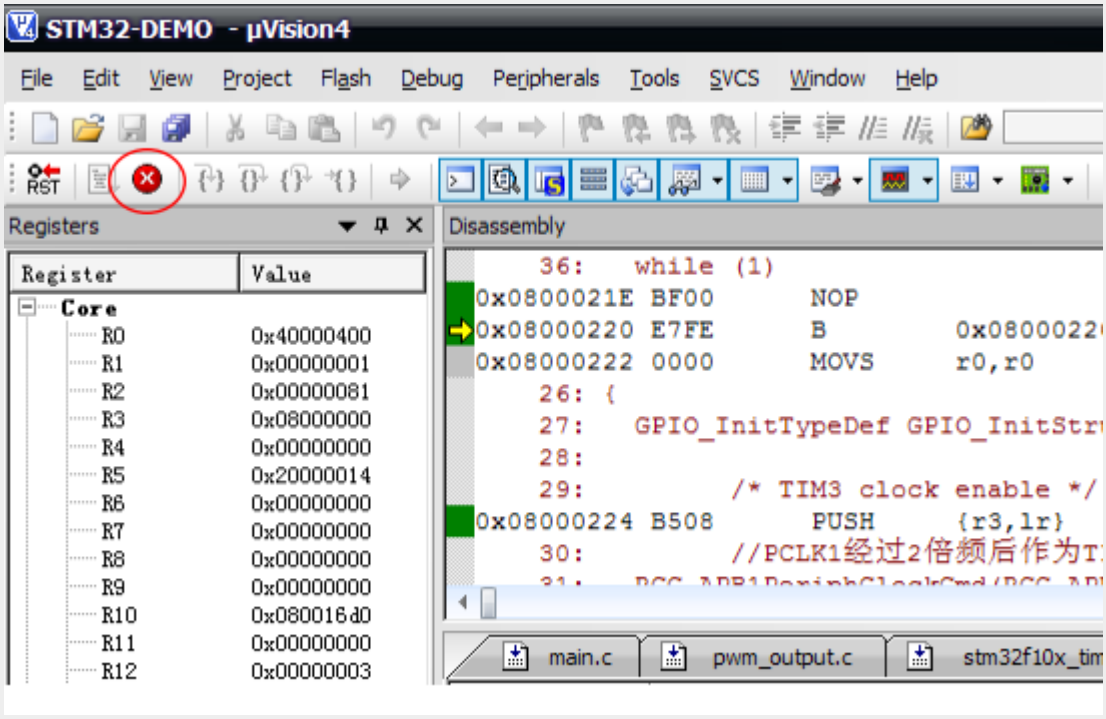
不正常的话则出现下面的情况，一团糟，根本看不到 PWM 信号。这时我们不免会有些抓狂呀，哈哈，但请大家放心，看看接下来我们是怎么解决的吧。



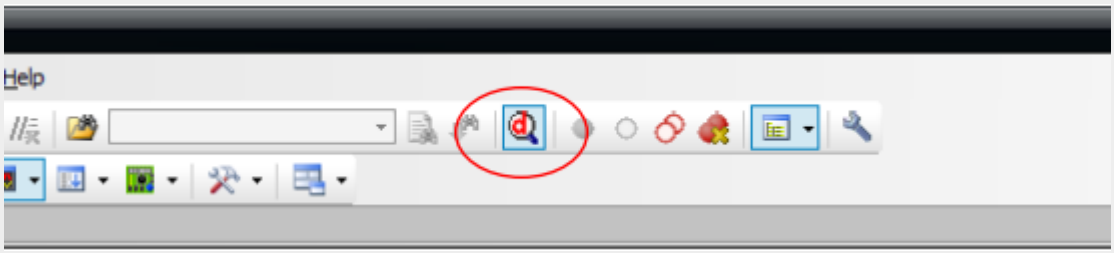
7. 7->其实出现上面的情况是因为我们显示 PWM 信号时没有放大的缘故，我们可以点击 In 这个按钮来将 PWM 信号显示的大点，如下所示，一切搞定。



8. 8->我们可以点击停止按钮，让 PWM 信号静止显示。



9. 9->仿真完毕之后，点击 Start/Stop Debug Session 选项就可以回到正常的代码编辑模式。



10. 10->要是您有示波器的话，看到的效果则更真实，更给力。