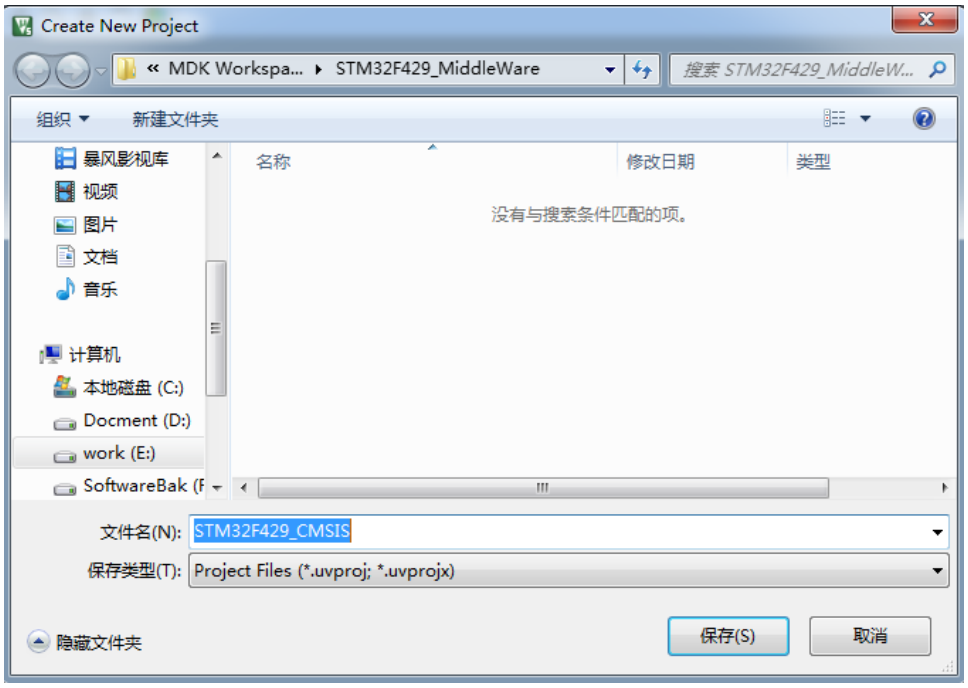
# MDK 的中间件的应用 V1.0

## 文档类别：代理产品技术文档

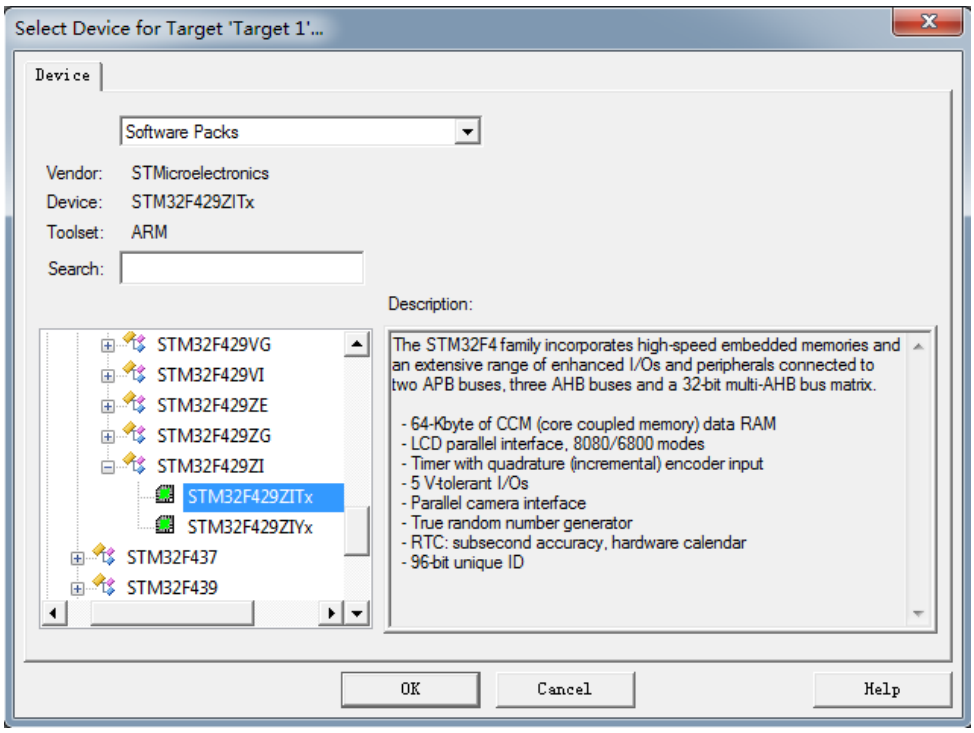
| 版本信息 | | |
|---|---|---|
| 版本号 | 编写日期 | 编写人 |
| V 1.0 | 2016 年 08 月 16 日 | 刘立明 |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |

# 一、基础工程序的创建

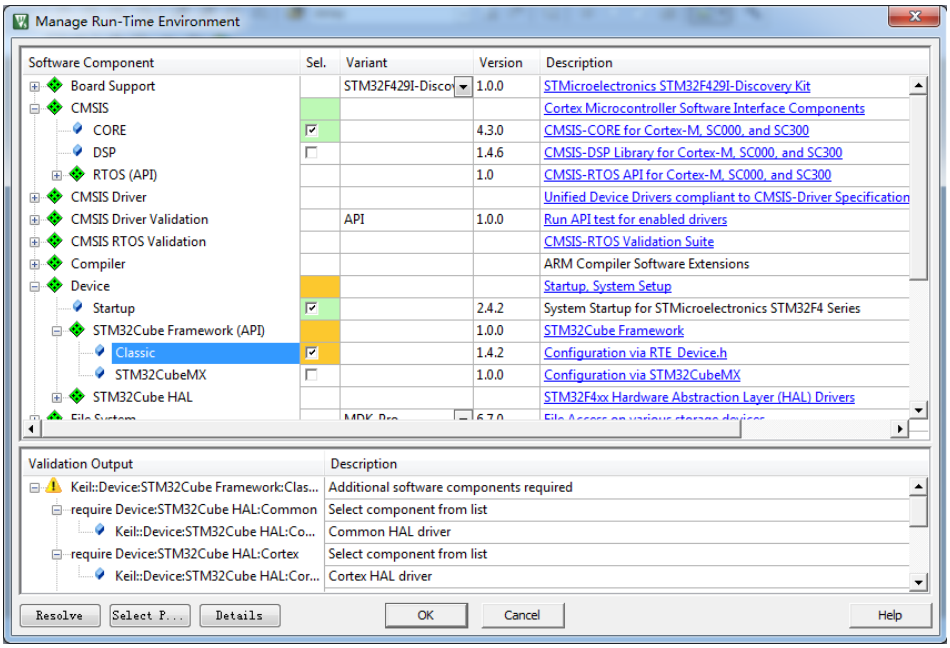
1、创建一个名为“STM32F429_CMSIS”的工程。



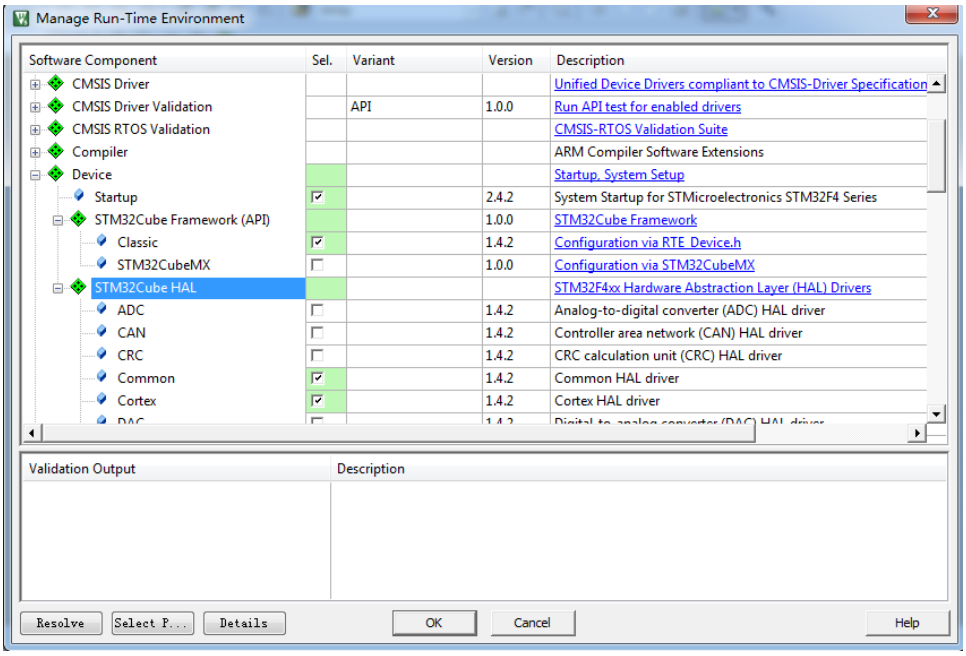

2、处理器选择“STM32F429ZITx”。



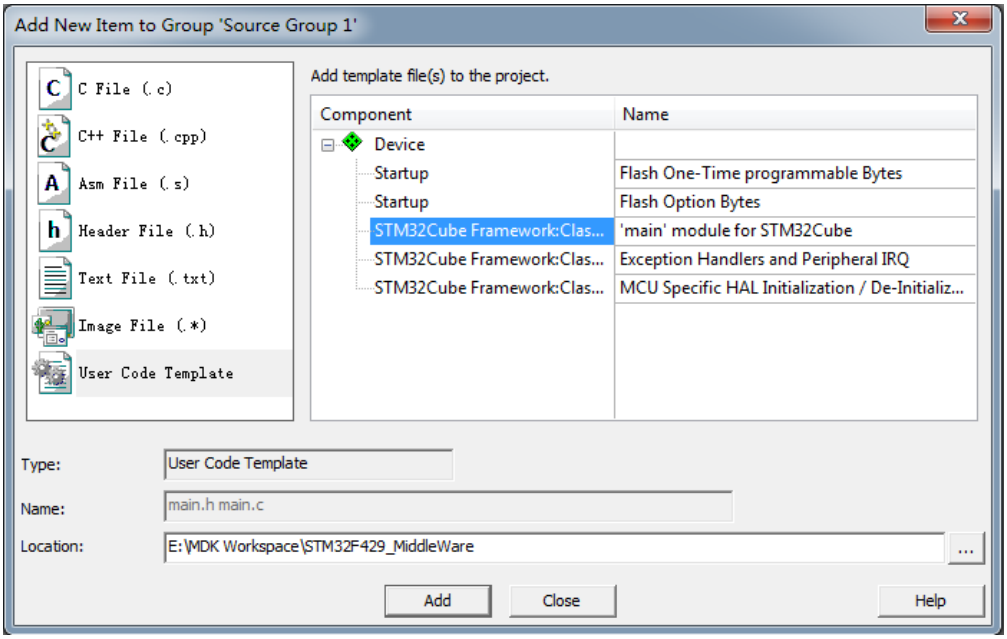
3、选择“Startup”和“Classic”组件。



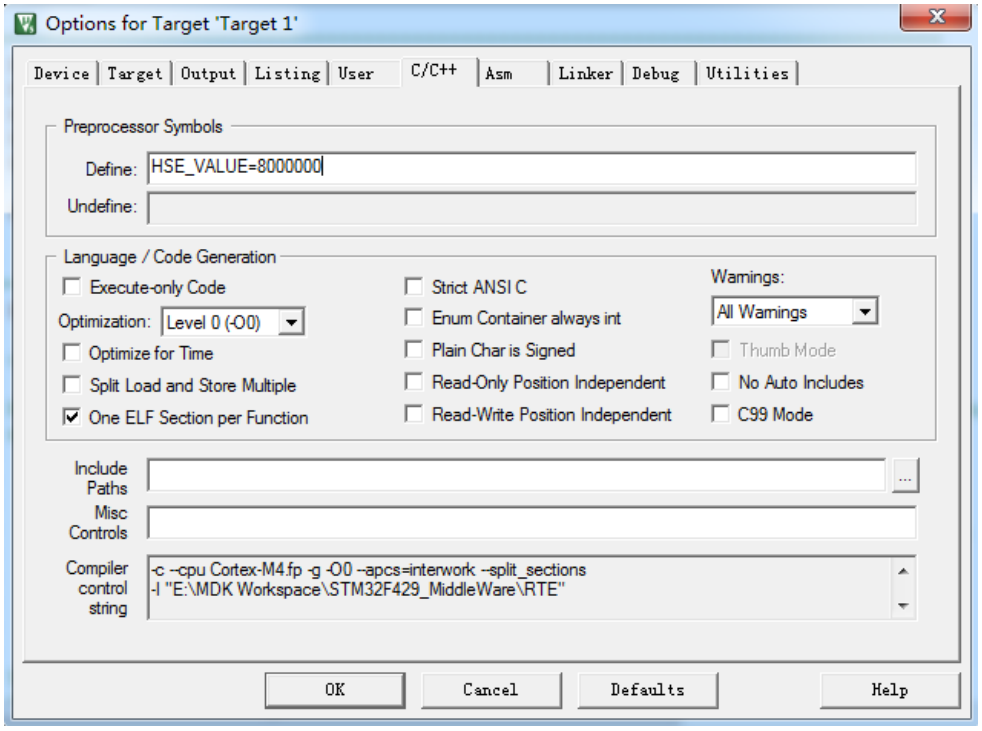

4、选择好相关联的组件。



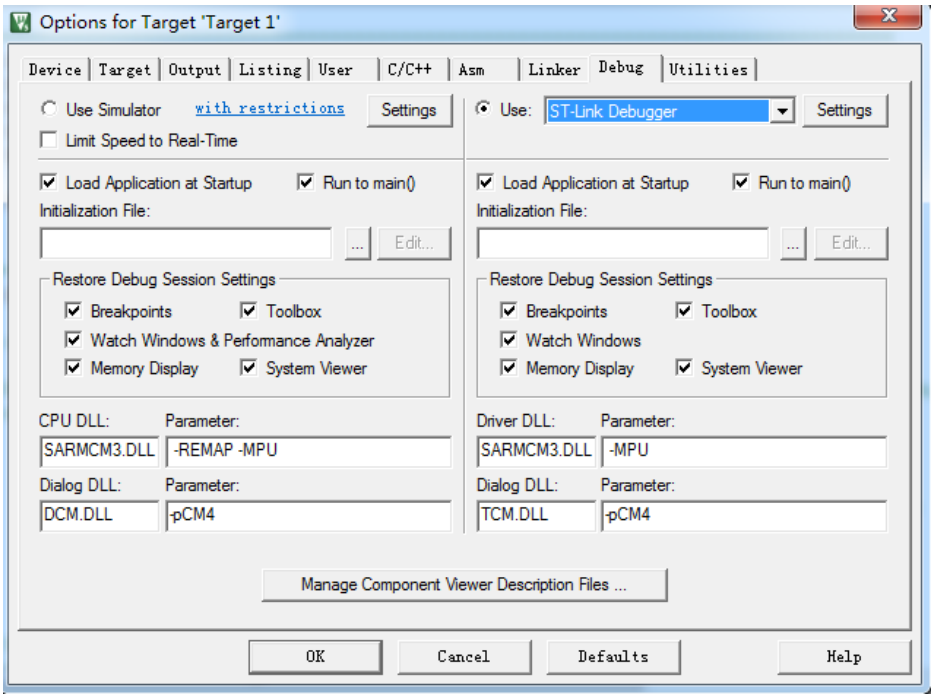
5、添加“main”主控程序。



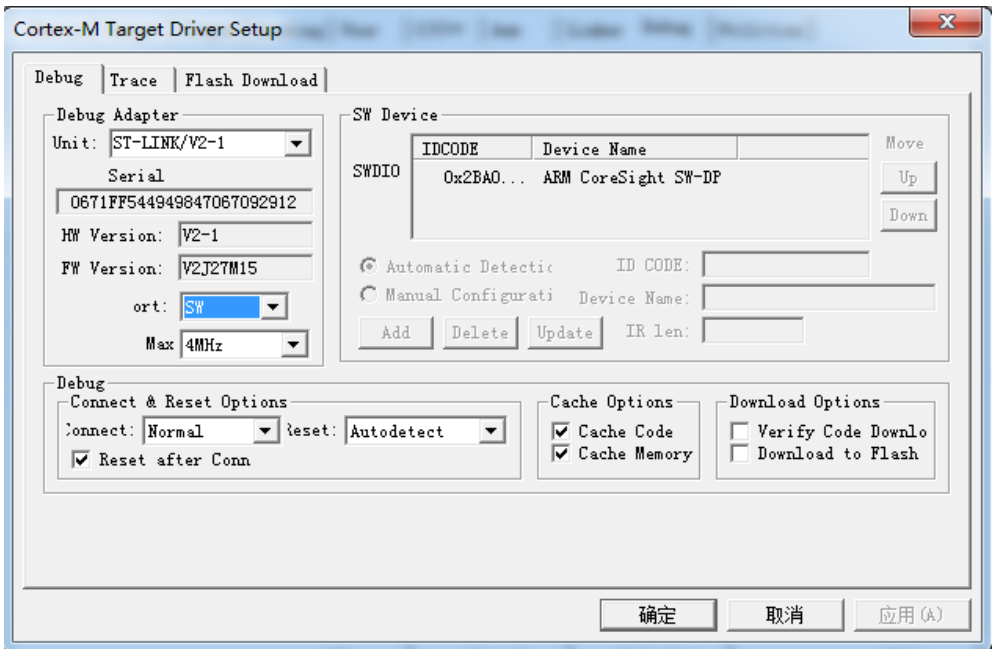

6、添加预处理符号：HSE_VALUE=8000000。



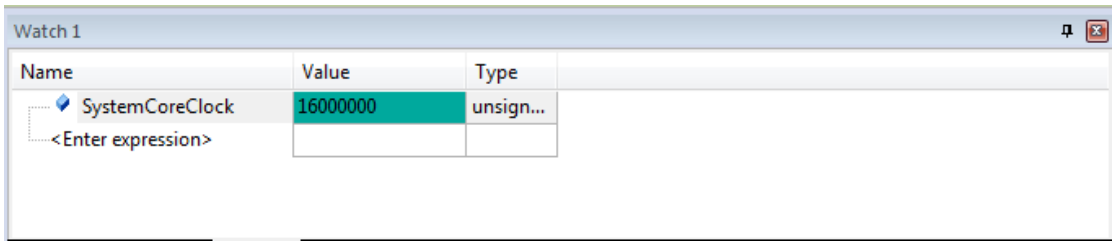
7、仿真器选择“ST-link Debugger”。



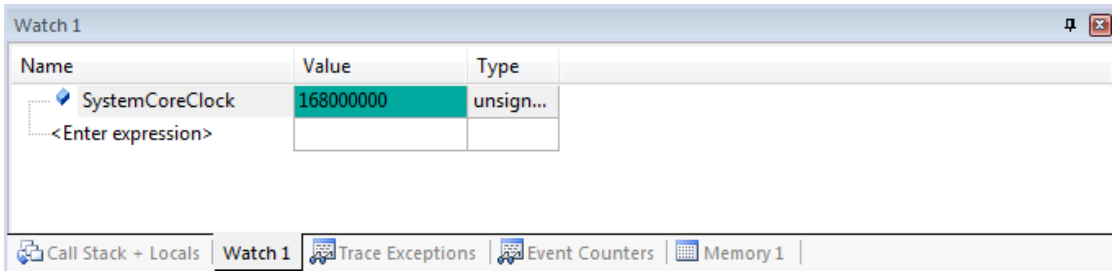

8、目标板相关驱动的设置，设置完成后，编译该工程。



9、在调试窗体中的 Watch 窗体中添加“SystemCoreClock”变量。

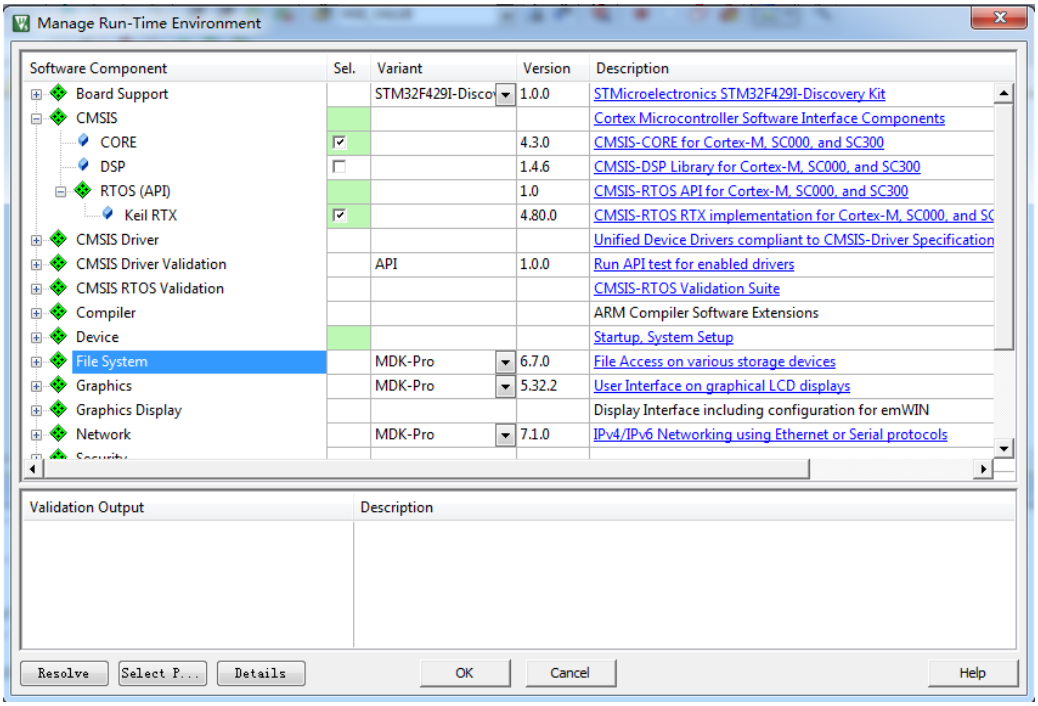



10、单步运行完“SystemClock_Config();”

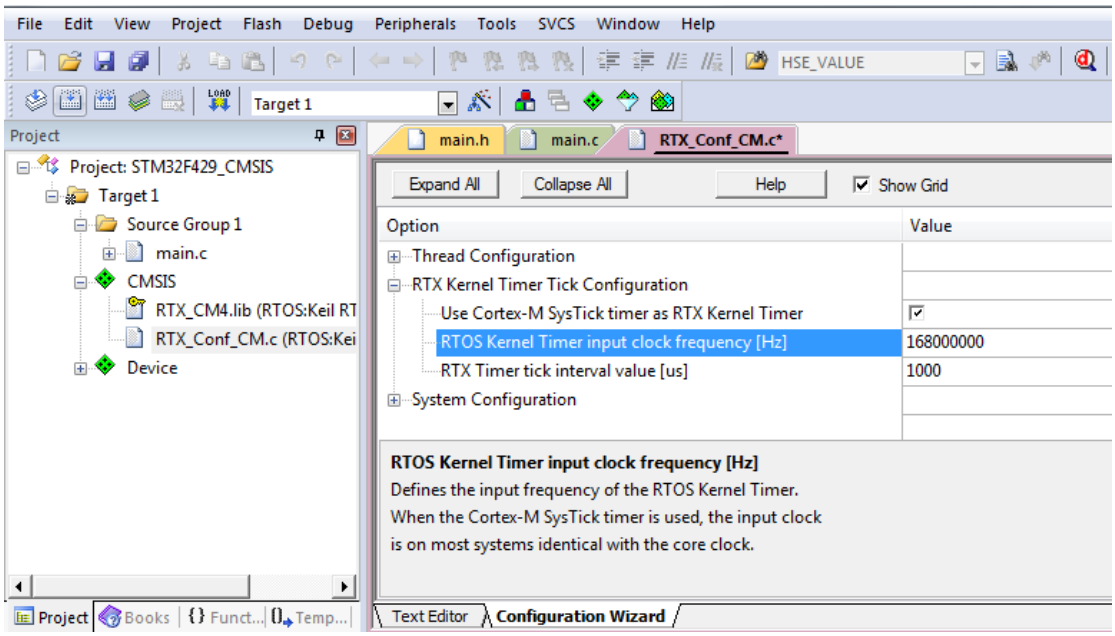


## 二、CMSIS-RTOS（RTX）的使用

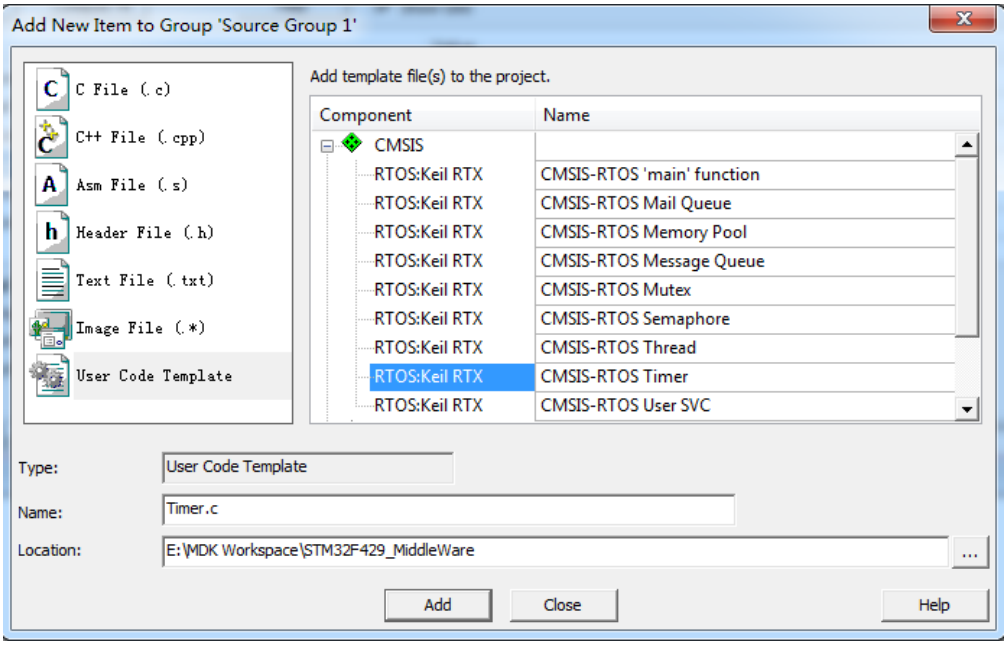
1、添加 RTX 组件。



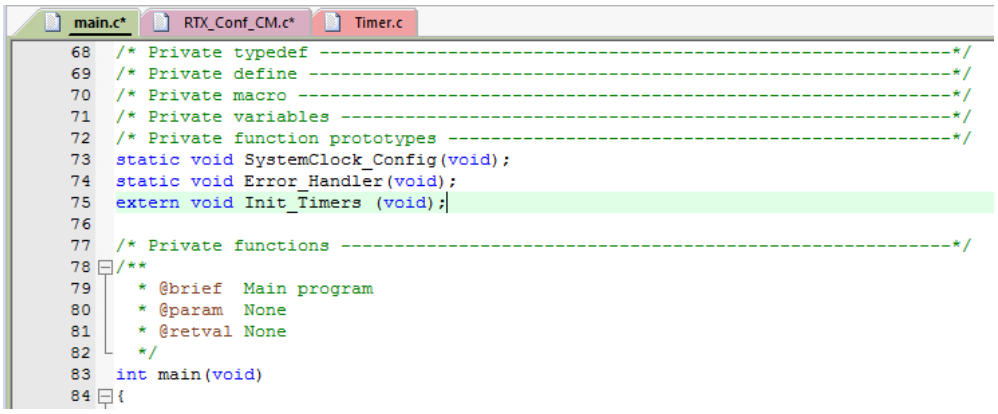

2、 修改 RTOS 的输入时钟值为：168000000.

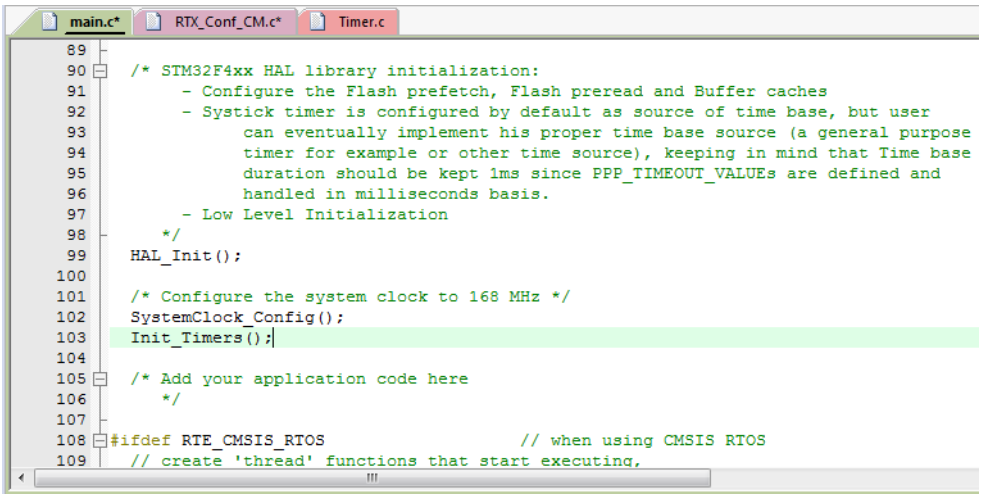


3、添加 Timer 组件。

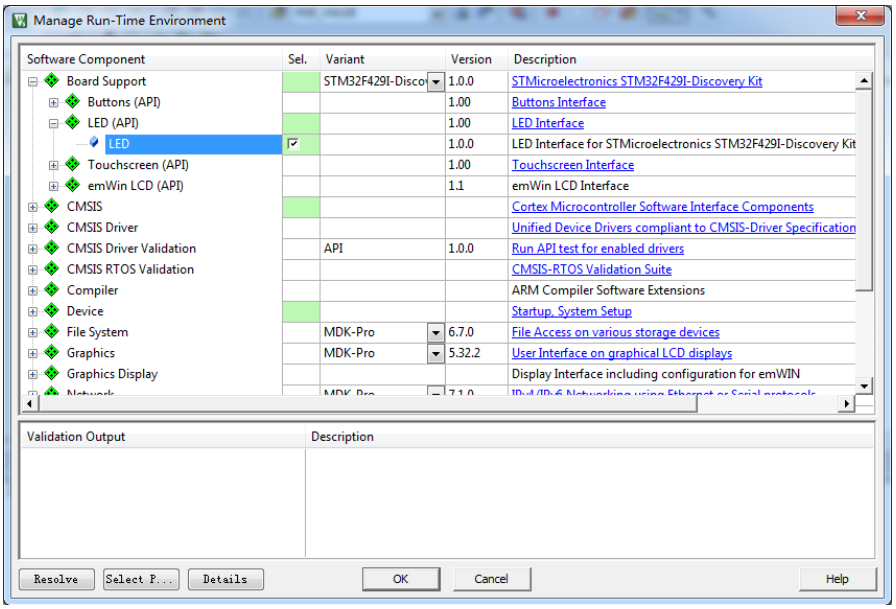



4、在“main.c”中请一下“Init_Timers”函数，并调用该函数。



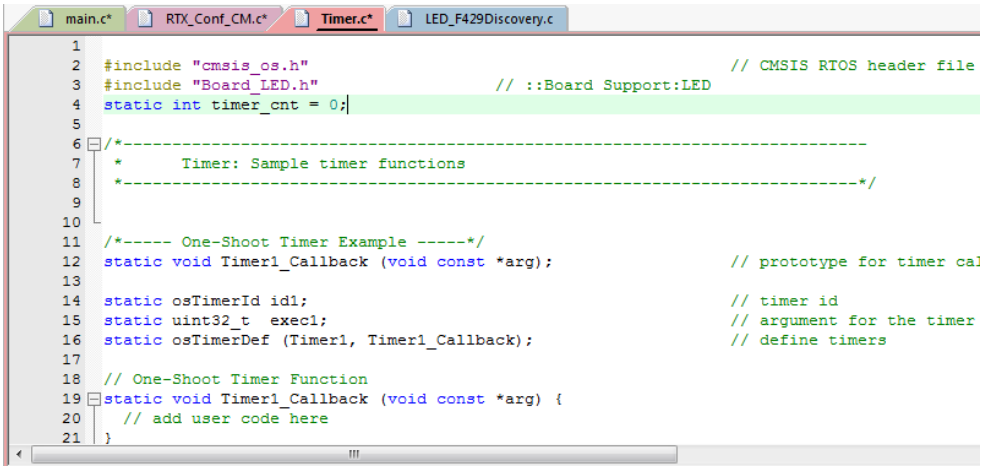



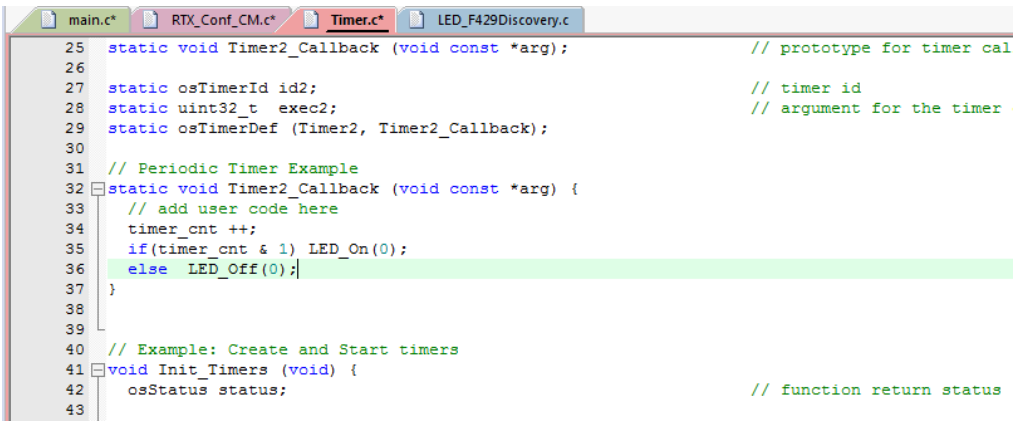
5、添加 LED 组件。




6、在“Timer.c”中添加 LED 的头文件，并添加一个时间的控制变量，再在“Timer2_Callback”函数中添加如下代码：

```
timer_cnt ++;
if(timer_cnt & 1) LED_On(0);
else  LED_Off(0);
```



7、在“main”文件中添加 LED 的头文件，并调用其初始化函数，编译工程。

```
Timer.c   main.c

41
42  /* Includes ------------------------------------------------------------------*/
43  #include "main.h"
44  #include "Board_LED.h"                    // ::Board Support:LED
45
46  #ifdef _RTE_
47  #include "RTE_Components.h"               /* Component selection */
48  #endif
49  #ifdef RTE_CMSIS_RTOS                     // when RTE component CMSIS RTOS is used
50  #include "cmsis_os.h"                     // CMSIS RTOS header file
51  #endif
```

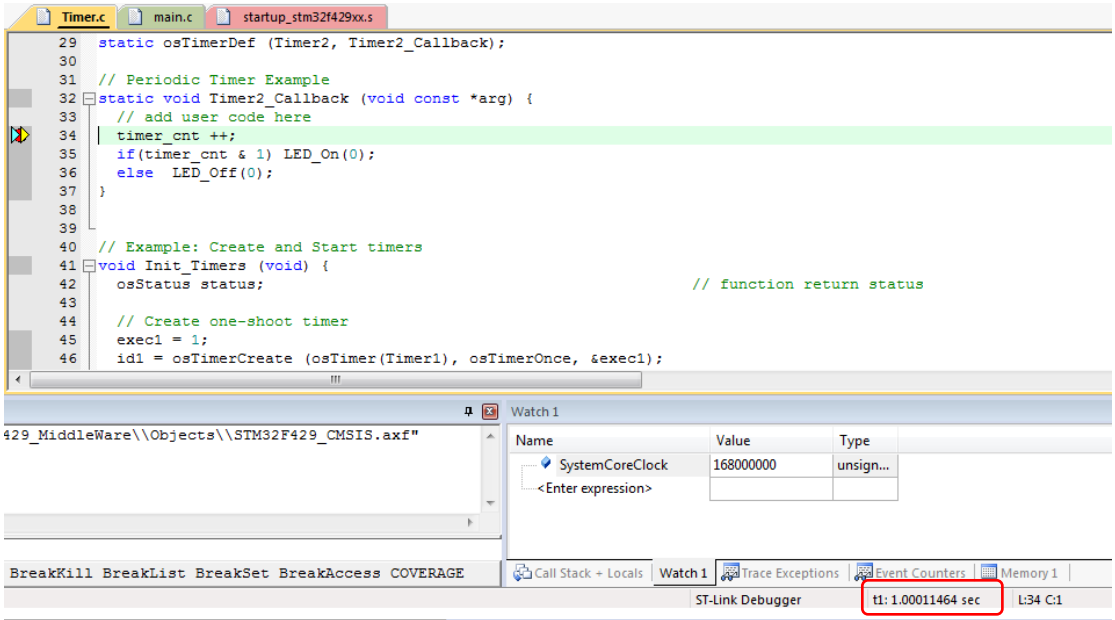
```
Timer.c   main.c

95                timer for example or other time source), keeping in mind that Time base
96                duration should be kept 1ms since PPP_TIMEOUT_VALUEs are defined and
97                handled in milliseconds basis.
98          - Low Level Initialization
99        */
100   HAL_Init();
101
102   /* Configure the system clock to 168 MHz */
103   SystemClock_Config();
104   Init_Timers();
105   LED_Initialize();
106
107   /* Add your application code here
108      */
```

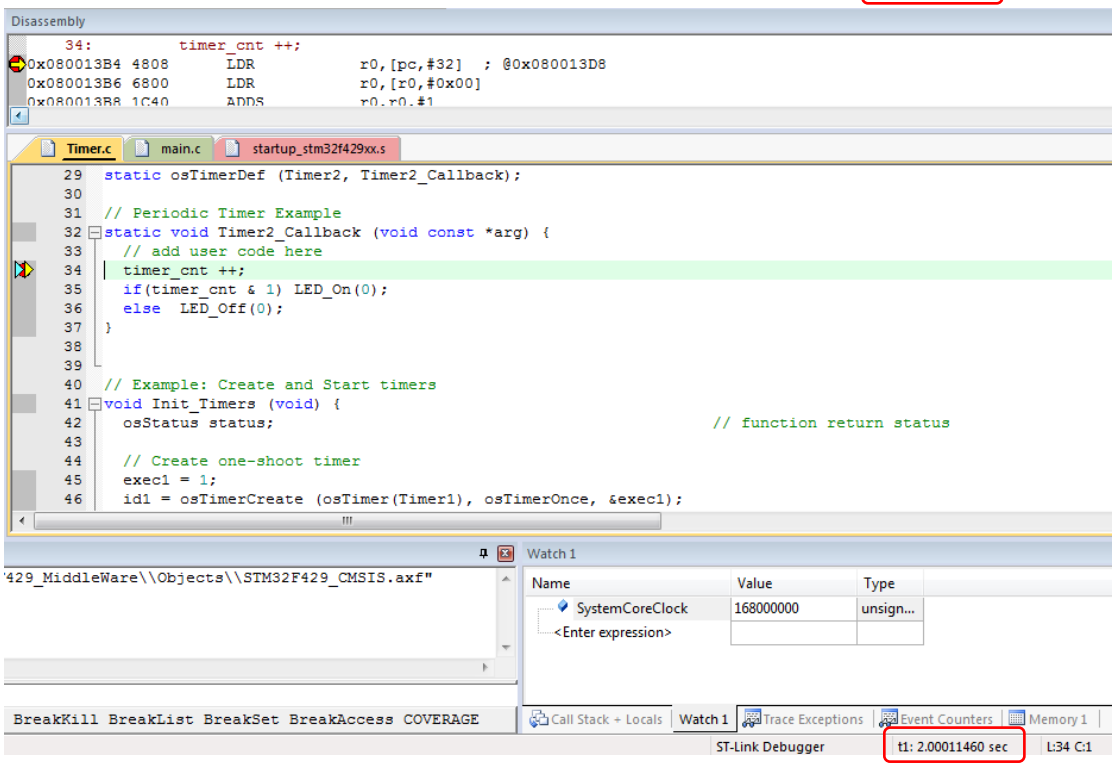
8、调试该工程，在“Timer2_Callback”中设置一个断点。

```
Disassembly
    34:         timer_cnt ++;
0x080013B4 4808      LDR          r0,[pc,#32]   ; @0x080013D8
0x080013B6 6800      LDR          r0,[r0,#0x00]
0x080013B8 1C40      ADDS         r0,r0,#1
```

```
Timer.c   main.c   startup_stm32f429xx.s

29  static osTimerDef (Timer2, Timer2_Callback);
30
31  // Periodic Timer Example
32  static void Timer2_Callback (void const *arg) {
33    // add user code here
34    timer_cnt ++;
35    if(timer_cnt & 1) LED_On(0);
36    else  LED_Off(0);
37  }
38
39
40  // Example: Create and Start timers
41  void Init_Timers (void) {
42    osStatus status;                                          // function r
43
44    // Create one-shoot timer
45    exec1 = 1;
46    id1 = osTimerCreate (osTimer(Timer1), osTimerOnce, &exec1);
```

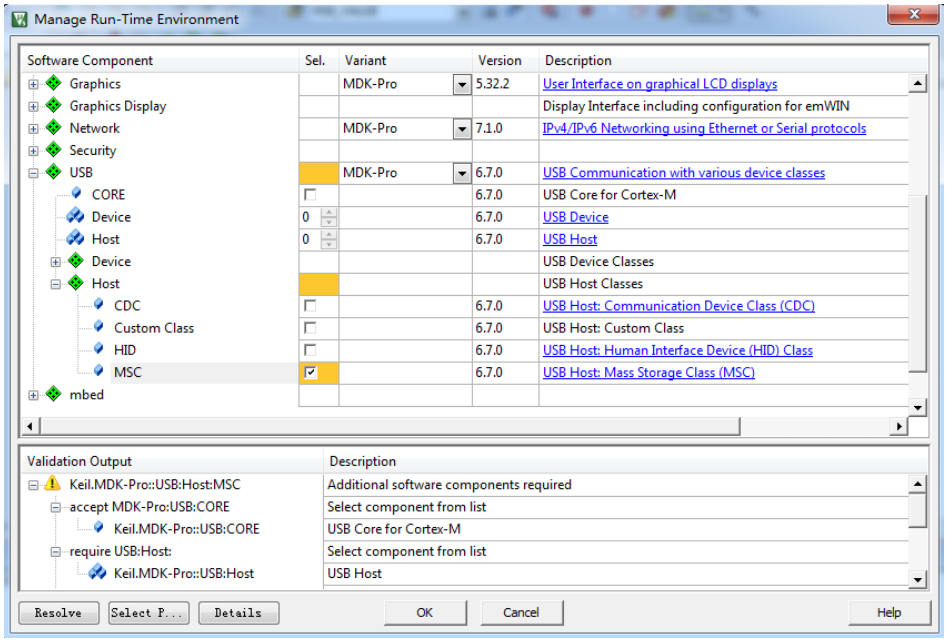
9、运行程序，可以查看到“t1”在按 1 秒进行步进。



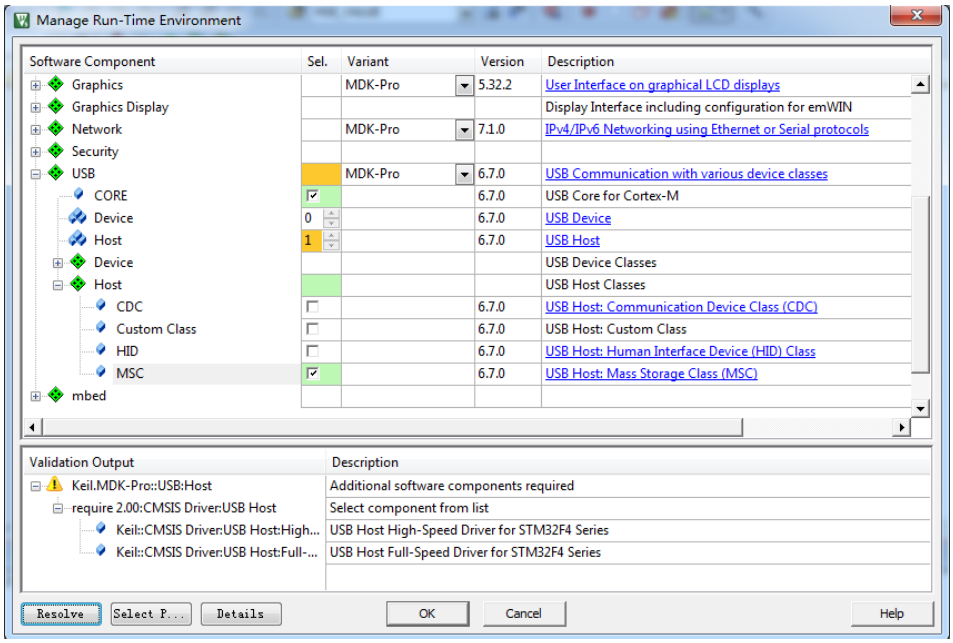



## 三、USB Host 的添加

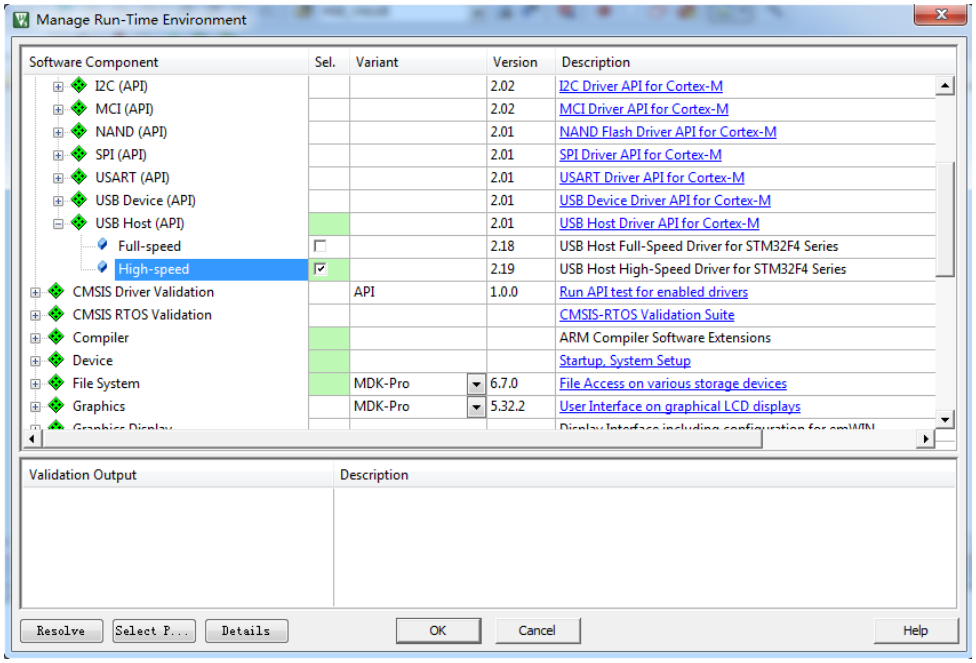
1、添加 USB Host 的 MSC 组件，



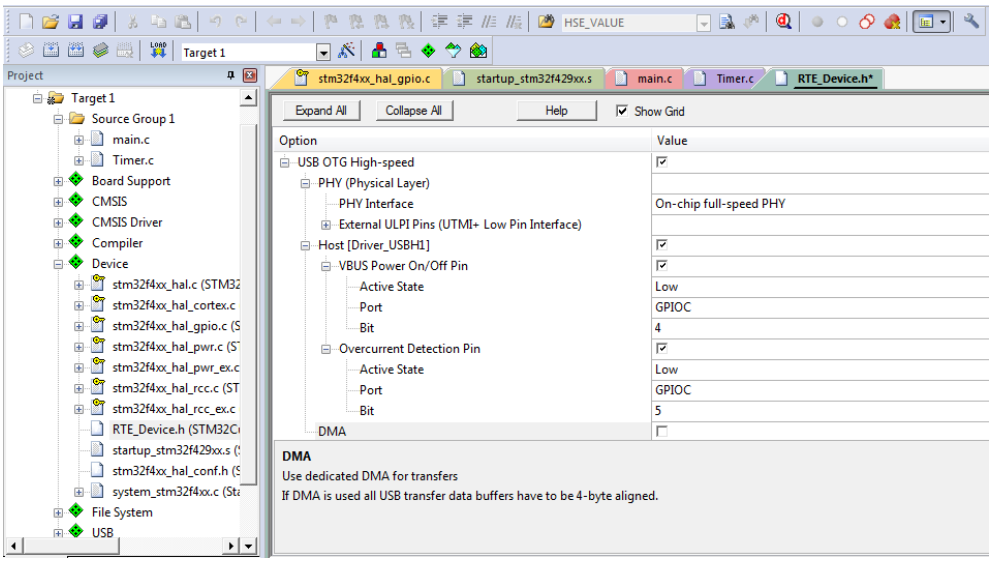

2、点击“Resolve”完成相关组件的添加。



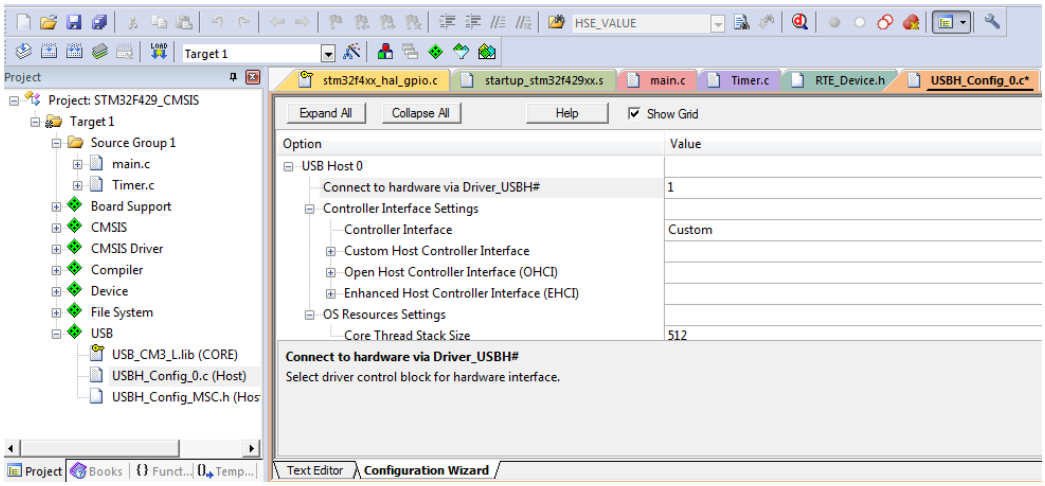
3、添加“High-speed”组件。

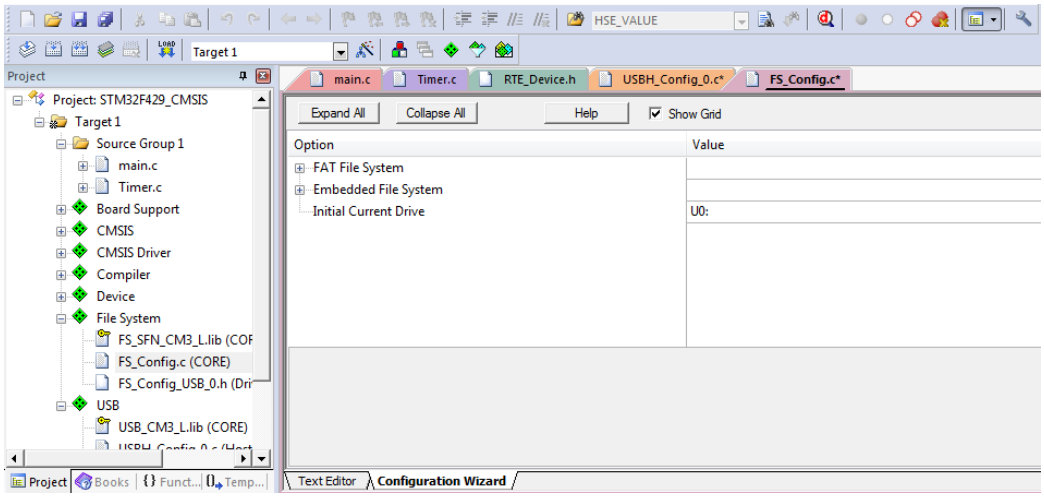



4、设置好 USB 的相关配置。

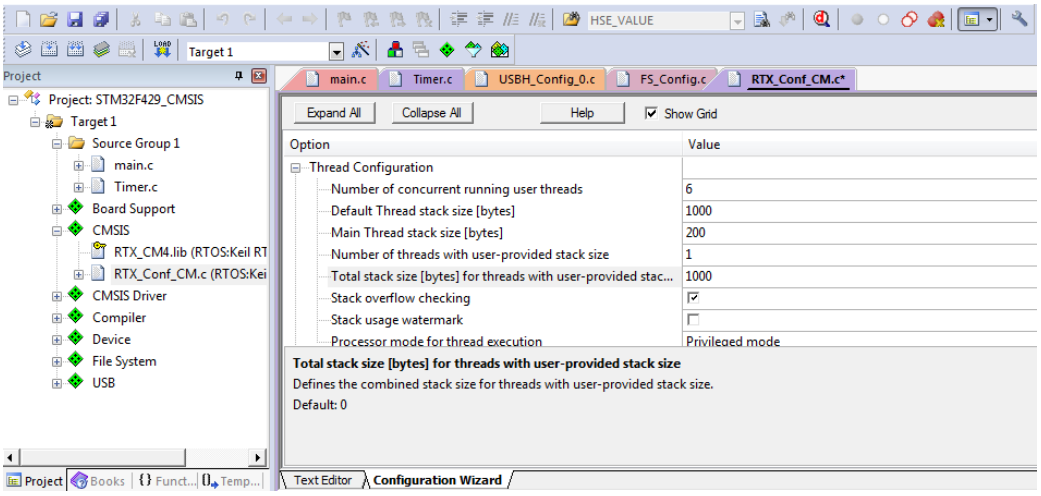


5、设置好 USB 的盘符号。

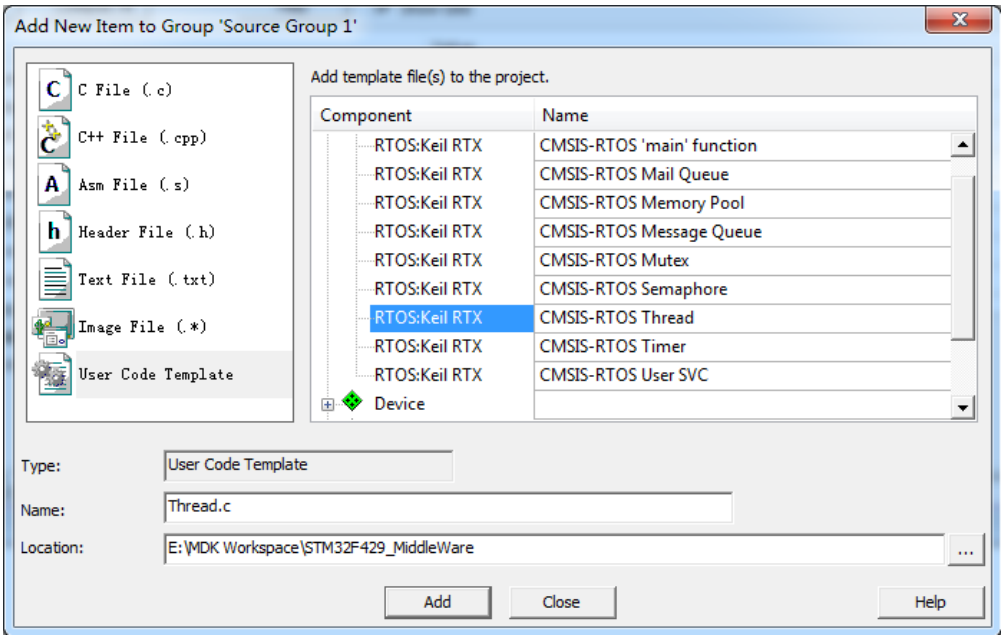


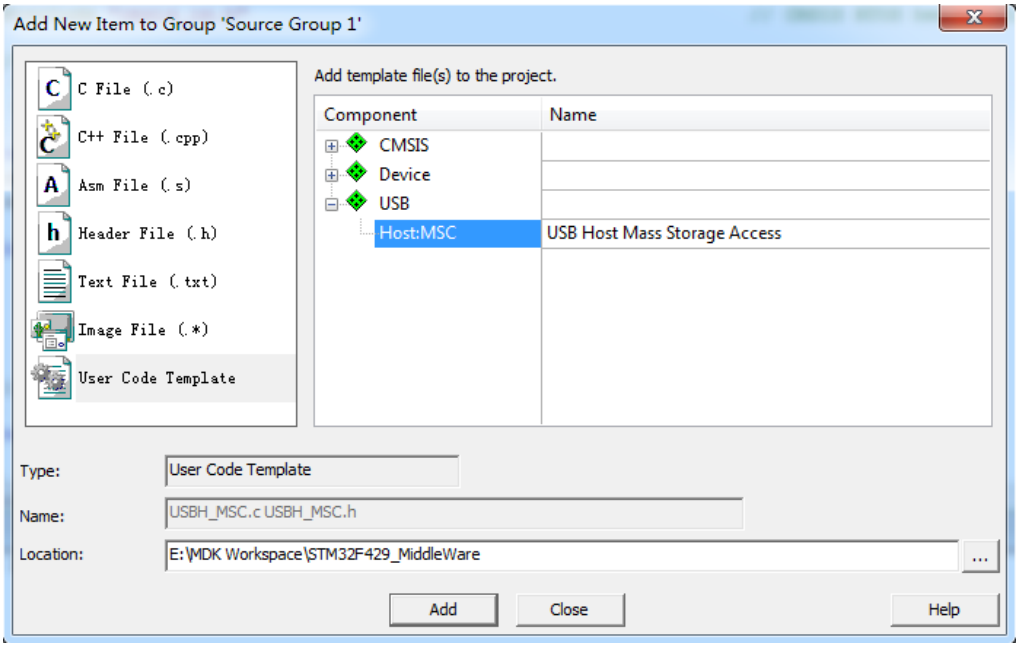

6、设置好堆栈。



7、添加一个线程。




8、添加 USB 存储的处理程序。



9、在”Thread”中添加USB Host相关的头文件，设置一个字符数组，并在“Thread”中添加如下代码：

```
static unsigned int result;
static FILE *f;

USBH_Initialize (0);

while (1) {
    result = USBH_MSC_DriveMount ("U0:");
    if (result == USBH_MSC_OK) {
        f = fopen ("Test.txt", "r");
        if (f) {
            fread (fbuf, sizeof (fbuf), 1 ,f);
            fclose (f);
        }
    }
    osDelay (1000);
}
```

main.c | Timer.c | USBH_Config_0.c | Thread.c*

```
1
2  #include "cmsis_os.h"                                        // CMSIS RTOS header file
3  #include "USBH_MSC.h"                    // Access storage via USB Host
4  char fbuf[2009] = {0};
5  /*----------------------------------------------------------------------------
6   *      Thread 1 'Thread_Name': Sample thread
7   *---------------------------------------------------------------------------*/
8
9  void Thread (void const *argument);                           // thread function
10 osThreadId tid_Thread;                                        // thread id
11 osThreadDef (Thread, osPriorityNormal, 1, 0);                 // thread object
12
13 int Init_Thread (void) {
14
15   tid_Thread = osThreadCreate (osThread(Thread), NULL);
16   if (!tid_Thread) return(-1);
17
18   return(0);
19 }
20
21 void Thread (void const *argument) {
```

main.c | Timer.c | USBH_Config_0.c | Thread.c

```
19 }
20
21 void Thread (void const *argument) {
22
23   static unsigned int result;
24   static FILE *f;
25
26   USBH_Initialize (0);
27
28   while (1) {
29     result = USBH_MSC_DriveMount ("U0:");
30     if (result == USBH_MSC_OK) {
31       f = fopen ("Test.txt", "r");
32       if (f) {
33         fread (fbuf, sizeof (fbuf), 1 ,f);
34         fclose (f);
35       }
36     }
37     osDelay (1000);
38   }
39 }
```

10、在“main”中申明“Init_Thread”函数，并在“osKernelstart”后调用该函数，编译该工程。

main.c* | Timer.c | USBH_Config_0.c | Thread.c

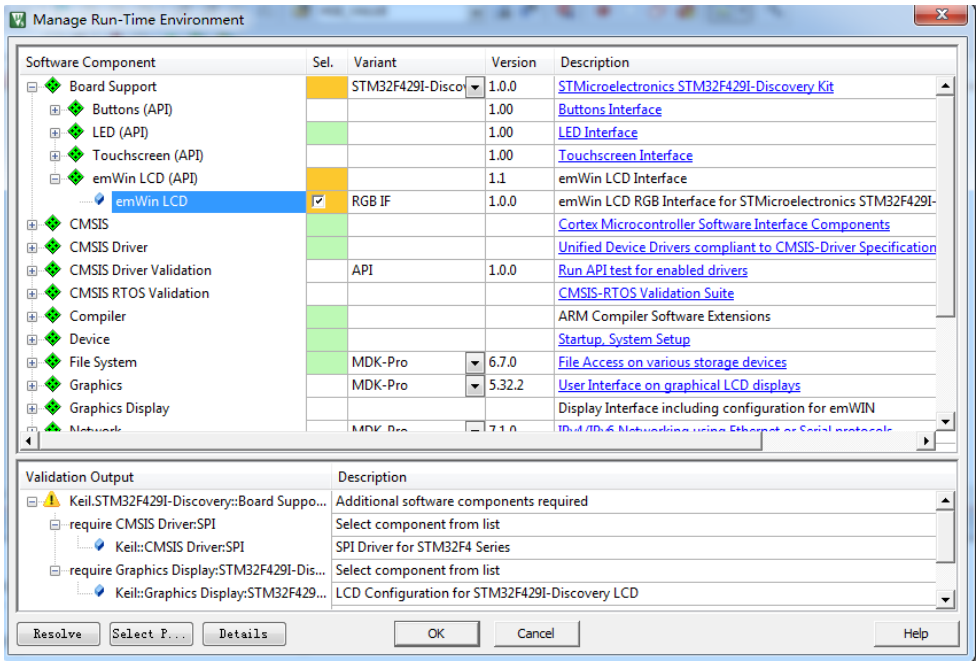
```
  * @{
  */

/* Private typedef -----------------------------------------------------------*/
/* Private define ------------------------------------------------------------*/
/* Private macro -------------------------------------------------------------*/
/* Private variables ---------------------------------------------------------*/
/* Private function prototypes -----------------------------------------------*/
static void SystemClock_Config(void);
static void Error_Handler(void);
extern void Init_Timers (void);
extern void Init_Thread(void);

/* Private functions ---------------------------------------------------------*/
/**
  * @brief  Main program
  * @param  None
  * @retval None
  */
int main(void)
{
```

main.c* | Timer.c | USBH_Config_0.c | Thread.c

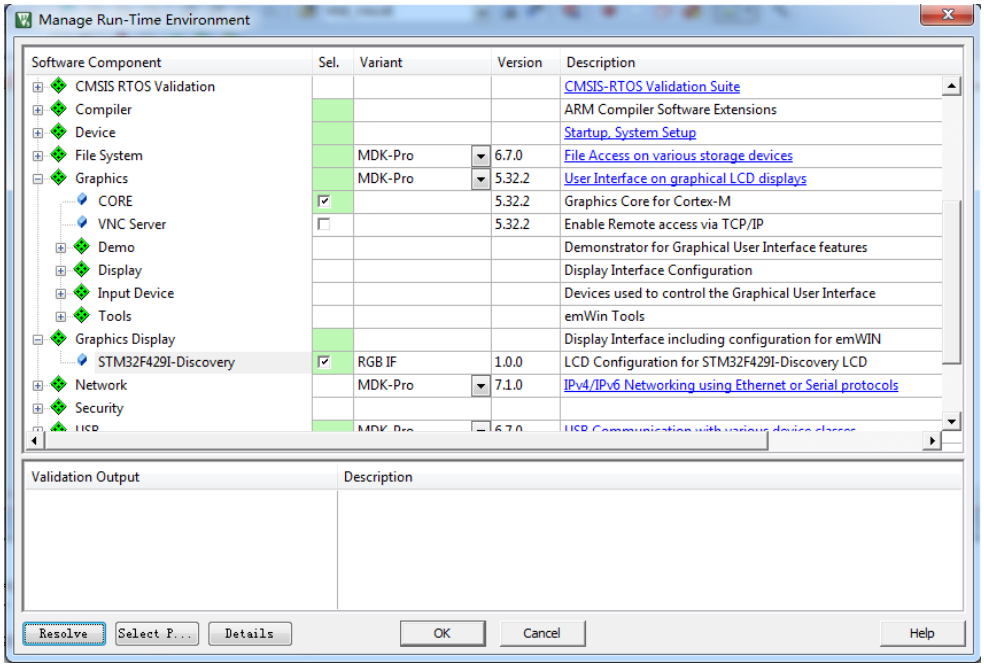
```
  Init_Timers();
  LED_Initialize();

  /* Add your application code here
     */

#ifdef RTE_CMSIS_RTOS                   // when using CMSIS RTOS
  // create 'thread' functions that start executing,
  // example: tid_name = osThreadCreate (osThread(name), NULL);

  osKernelStart();                      // start thread execution
  Init_Thread();
#endif

  /* Infinite loop */
  while (1)
  {
  }
}

/**
  * @brief  System Clock Configuration
```

## 四、图形 GUI 的添加。

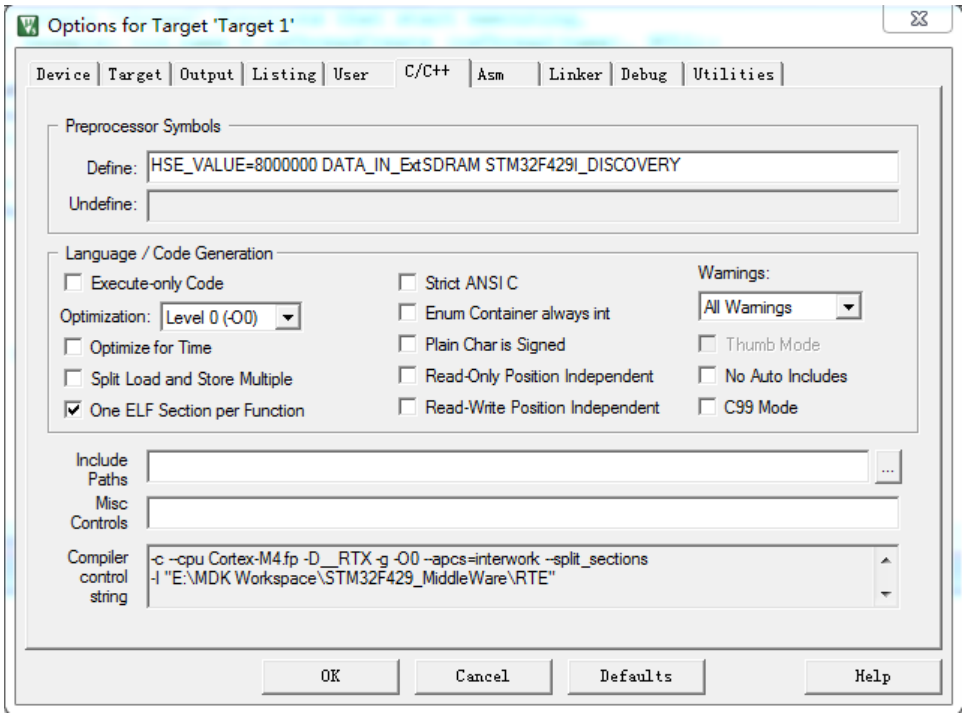
1、添加 LCD 组件。



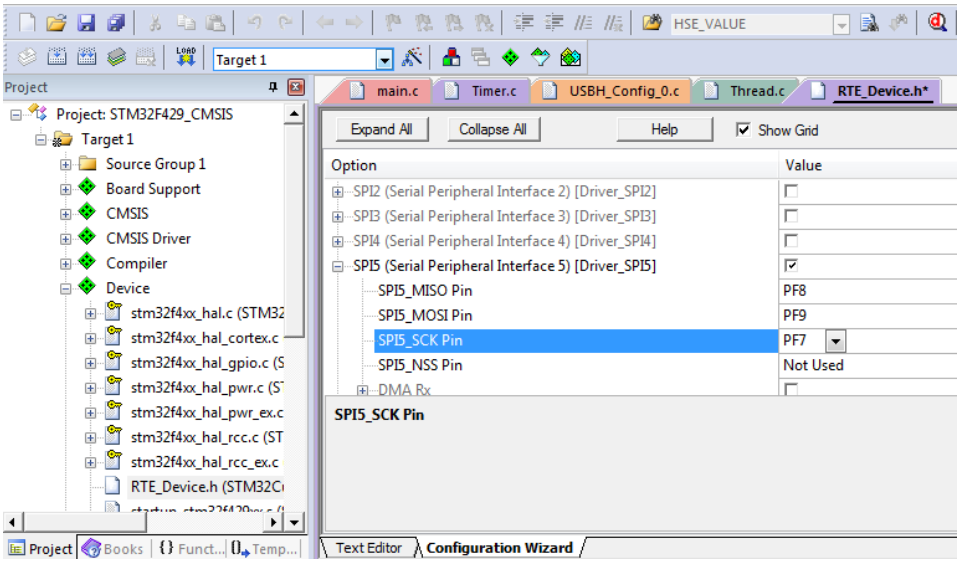

2、点击“Resolve”完成相关组件的添加。



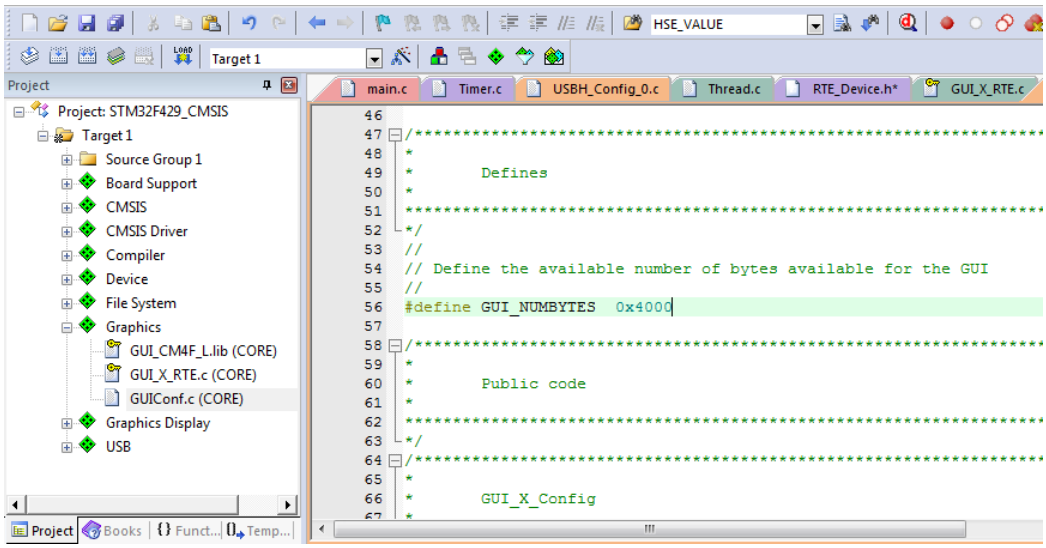
3、添加两个符号定义：DATA_IN_ExtSDRAM 和 STM32F429I_DISCOVERY。



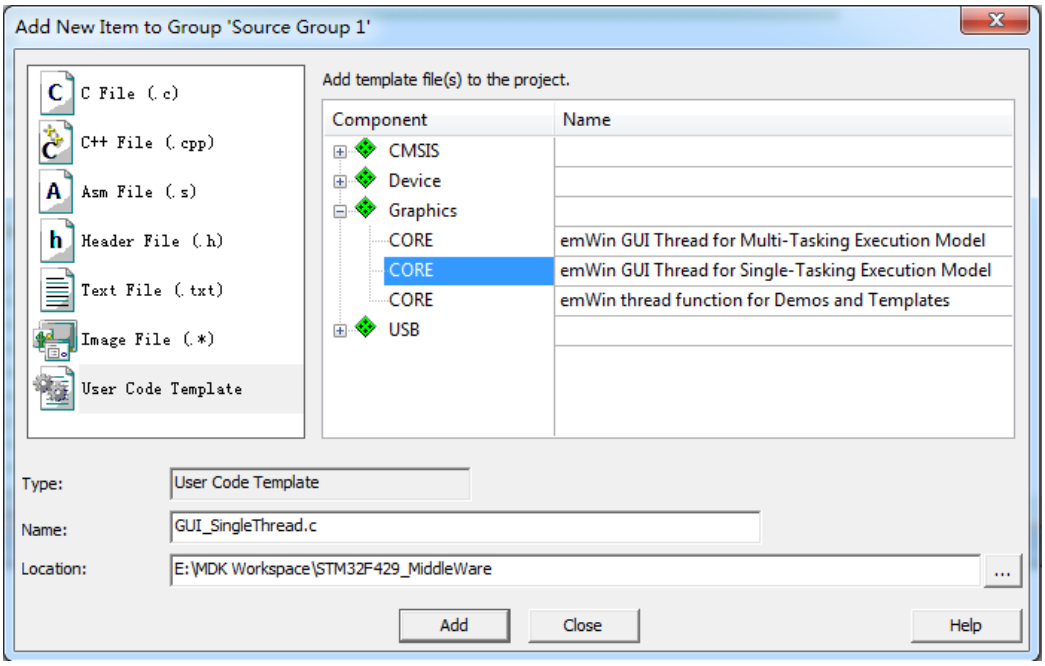

4、根据硬件原理图完成硬件接口的设置。



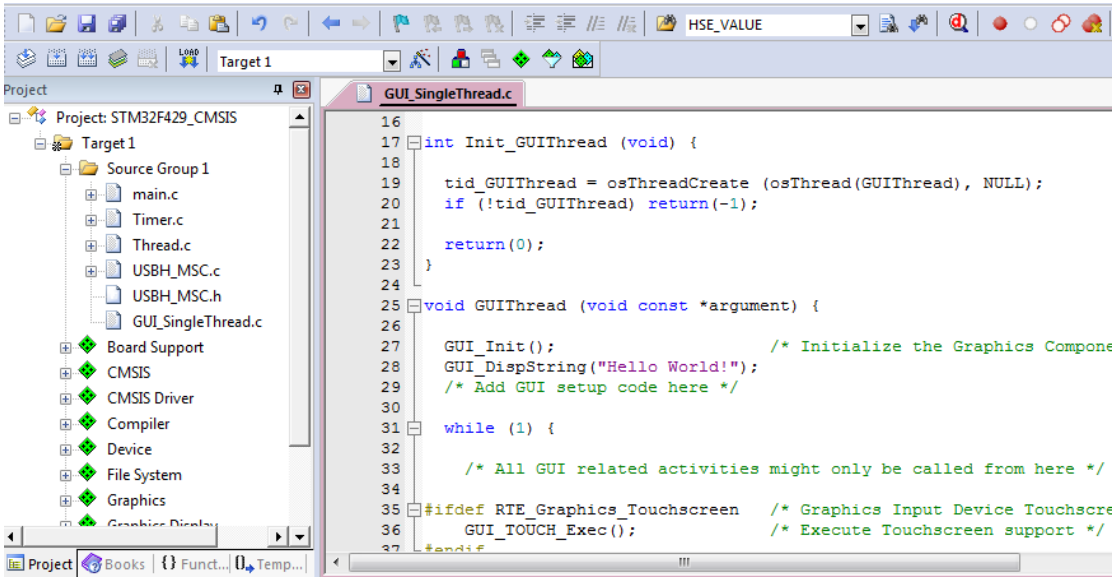
5、定义 GUI 有效的字节数。



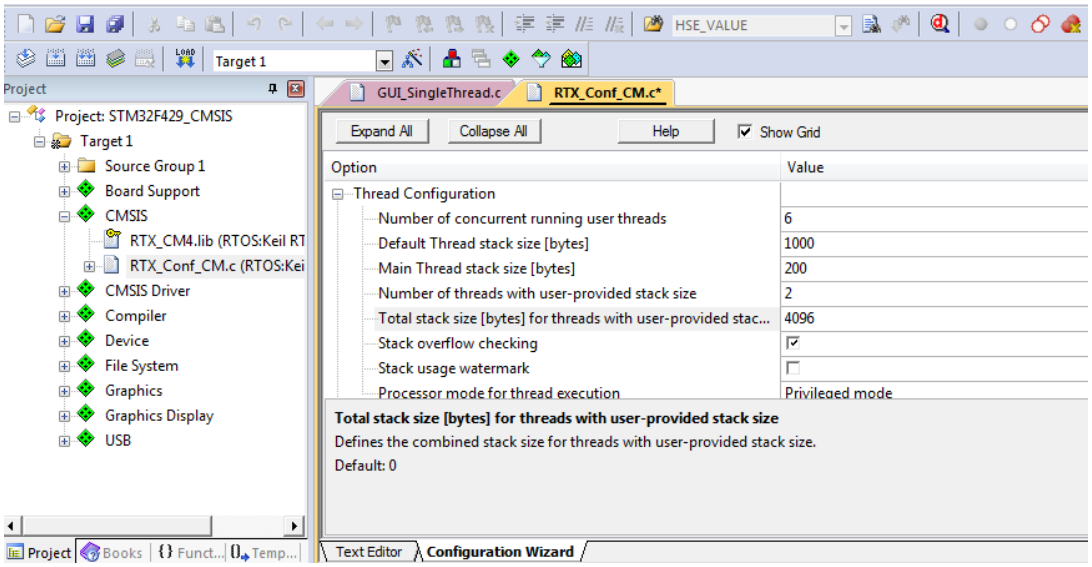

6、添加 GUI 的线程。



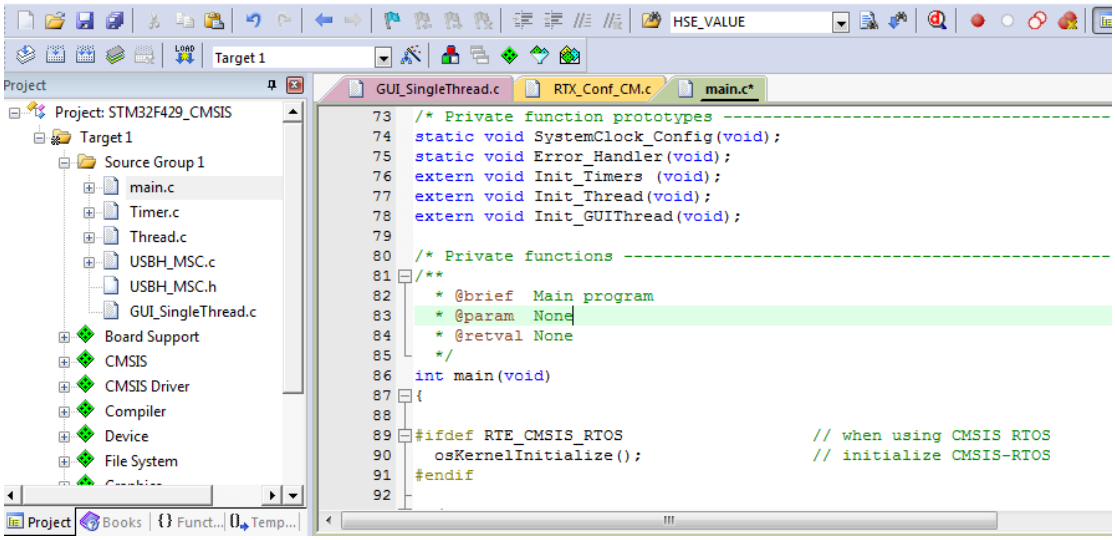
7、在 GUI 的线程中，添加：GUI_DispString("Hello World!");

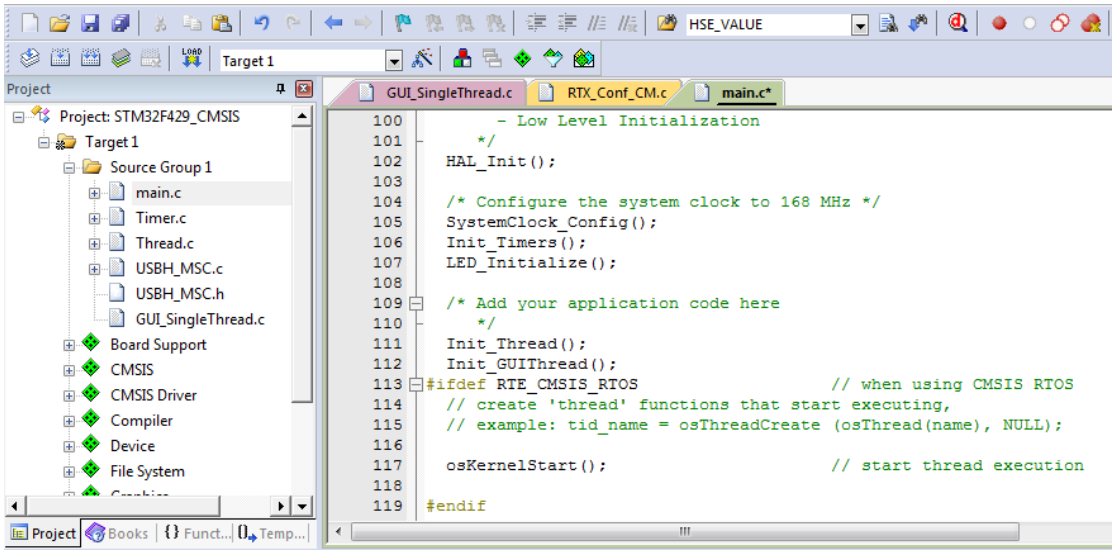



8、配置好堆栈。



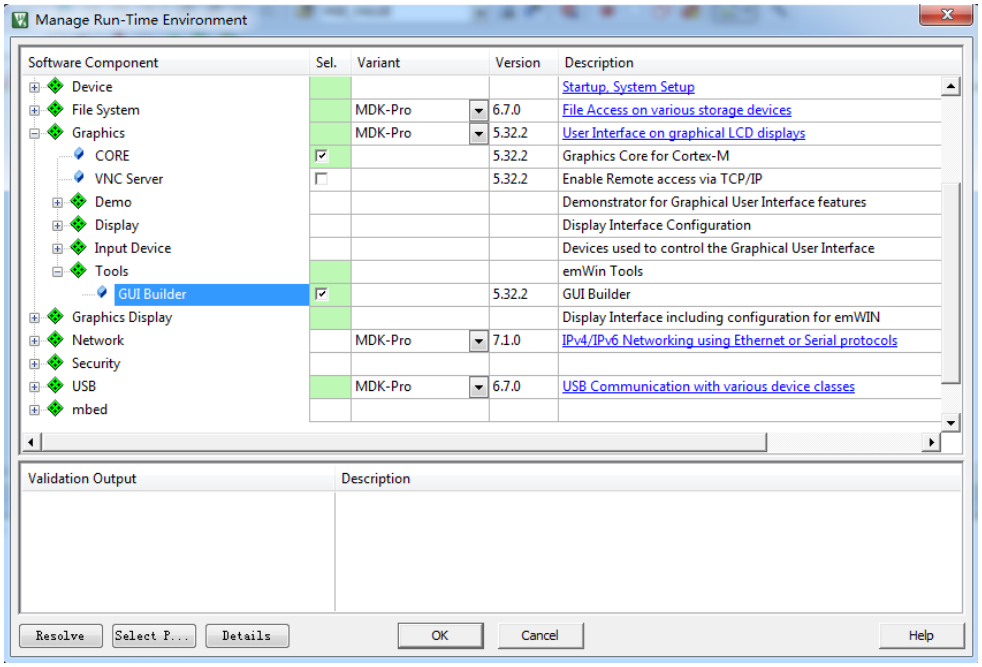
9、在“main”中申明一下 GUI 的初始化线程，并调用该线程，编译该工程。



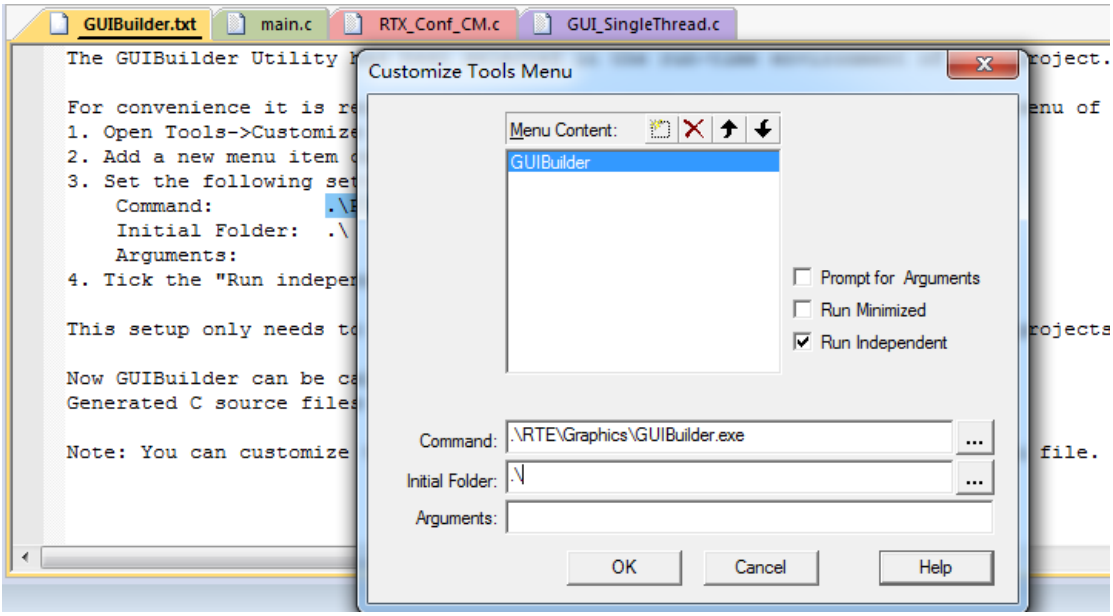



## 五、创建 GUI。

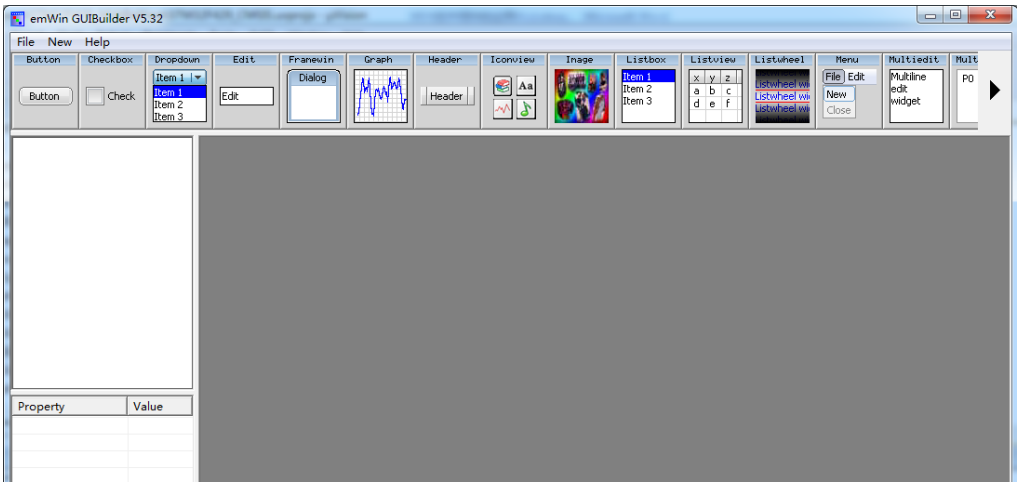
1、添加 GUI Builder 组件。



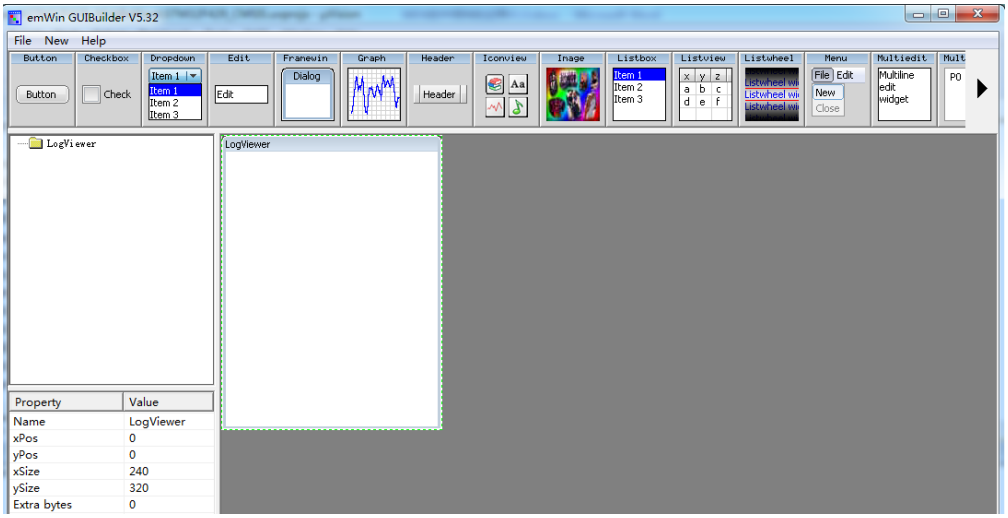

2、在“Tools”中添加 GuiBuilder，具体配置如下：



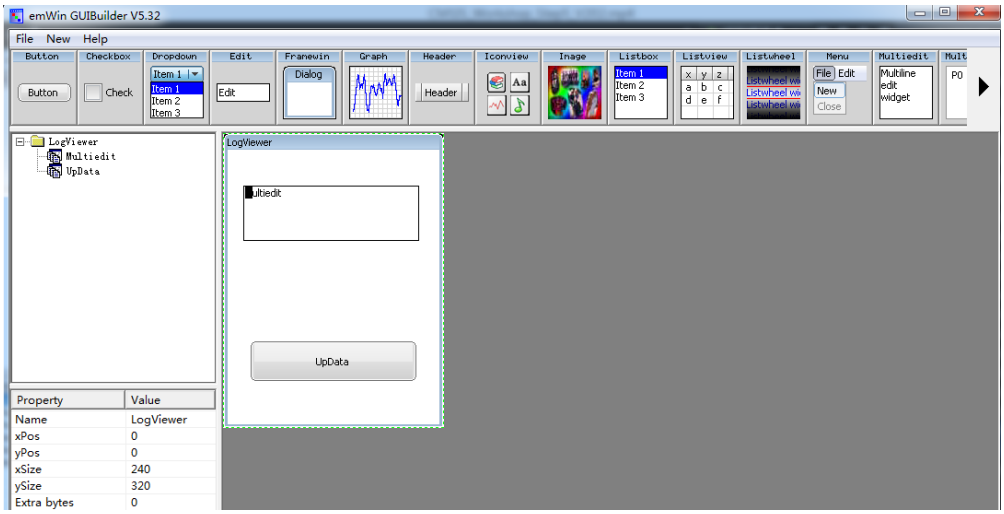
3、从“Tools”中打开 GUI Builder。



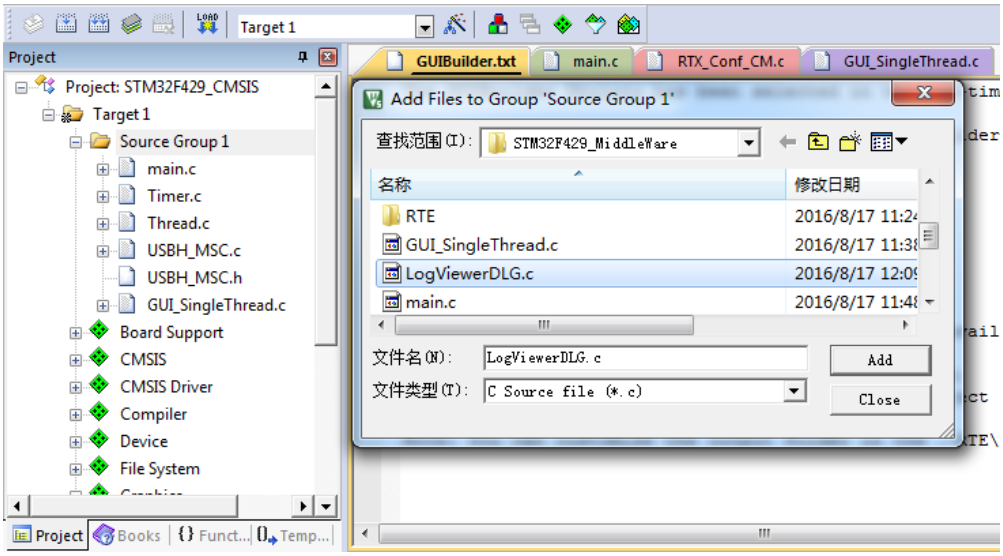

4、添加 FramWin，具体配置如下：



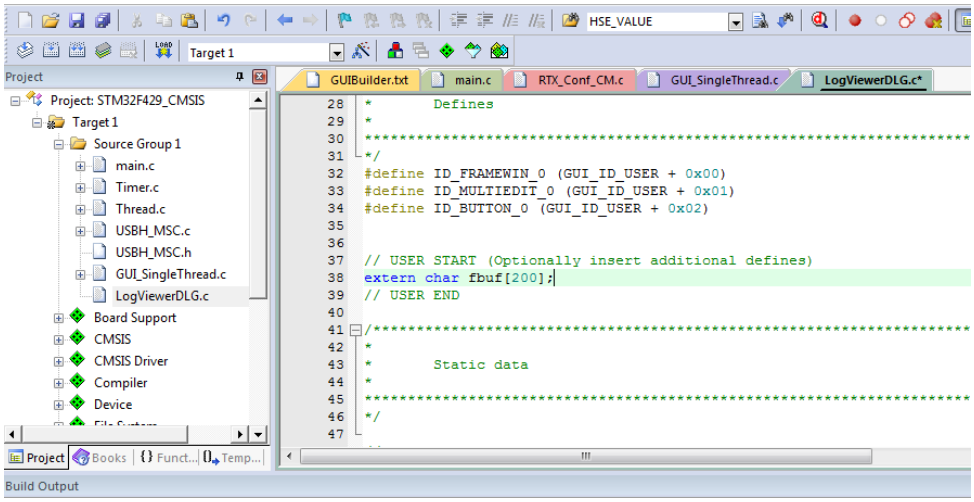

5、添加“Multiedit”和“Bottom”，保存文件，并退出 GUIBuilder。



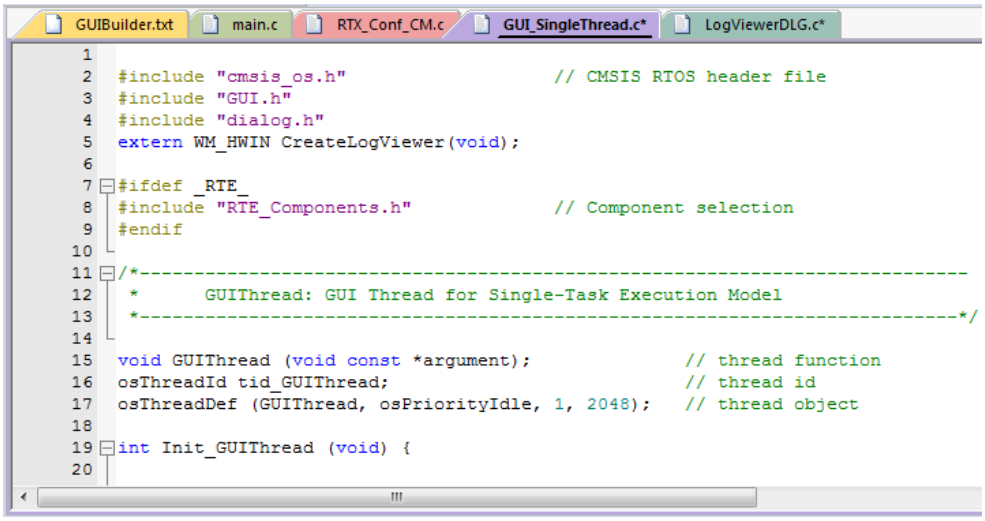
6、把刚生成的 GUI 的文件添加到工程中。




7、在“LogViewerDLG”文件中添加文件缓冲数组。




8、在 GUI 线程文件中添加：#include ″dialog.h″ 和 extern WM_HWIN CreateLogViewer(void);



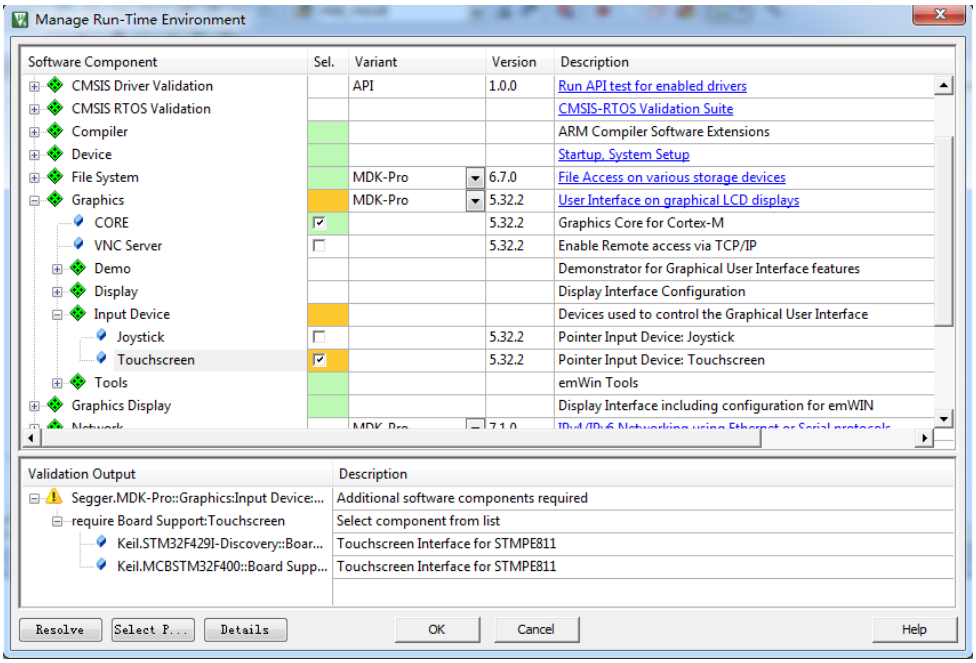
9、在 GUI 线程程序中，调用“CreateLogViewer”，保存文件并编译工程。

GUIBuilder.txt | main.c | RTX_Conf_CM.c | GUI_SingleThread.c* | LogViewerDLG.c*

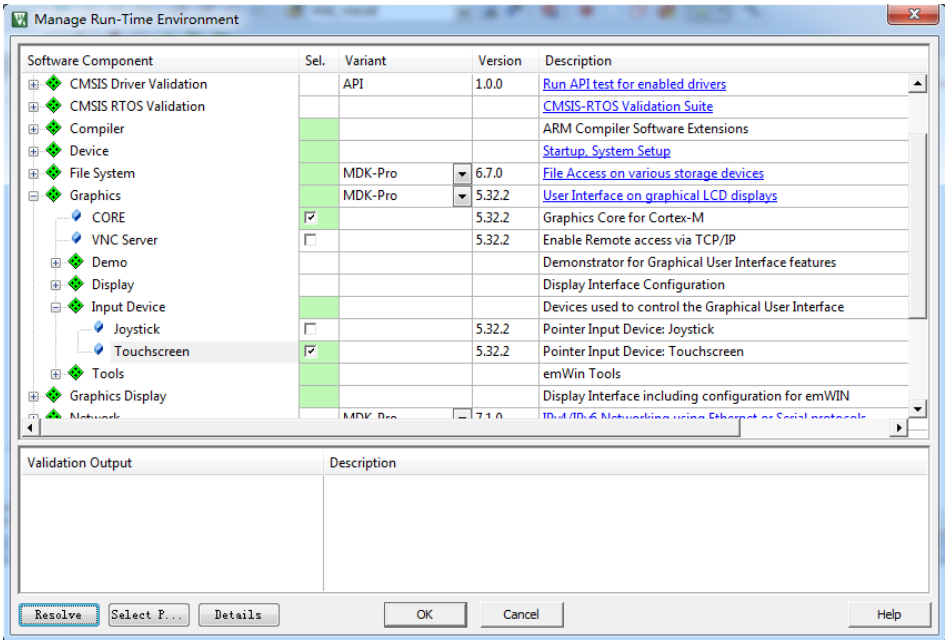
```
17  osThreadDef (GUIThread, osPriorityIdle, 1, 2048);   // thread object
18
19  int Init_GUIThread (void) {
20
21    tid_GUIThread = osThreadCreate (osThread(GUIThread), NULL);
22    if (!tid_GUIThread) return(-1);
23
24    return(0);
25  }
26
27  void GUIThread (void const *argument) {
28
29    GUI_Init();                     /* Initialize the Graphics Component */
30    //GUI_DispString("Hello World!");
31    CreateLogViewer();
32    /* Add GUI setup code here */
33
34    while (1) {
35
36      /* All GUI related activities might only be called from here */
```

## 六、触摸屏控制的添加。

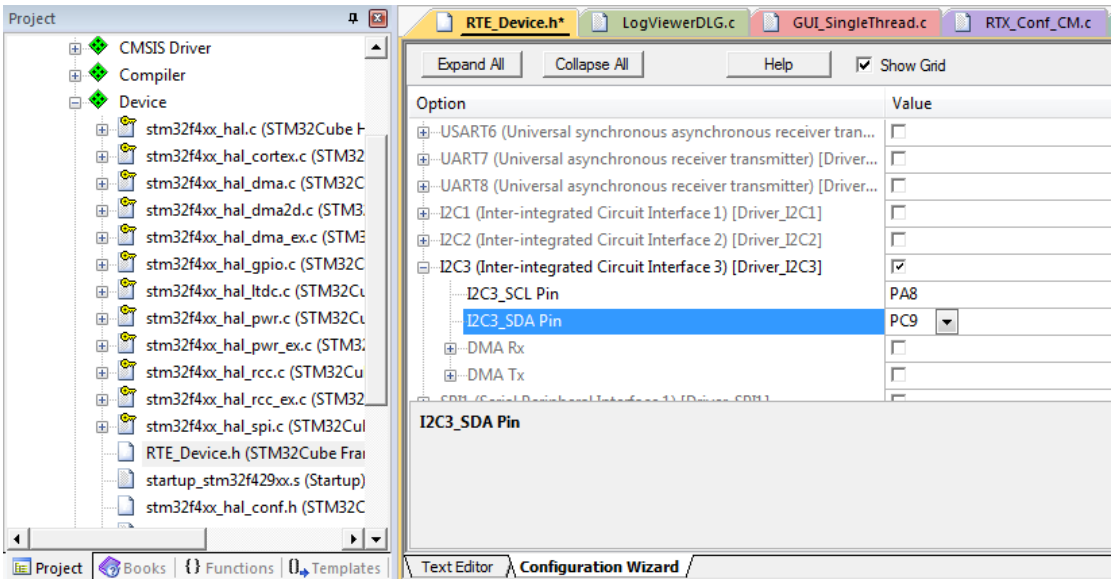
1、添加触摸屏的组件。




2、添加与触摸屏相关的组件。

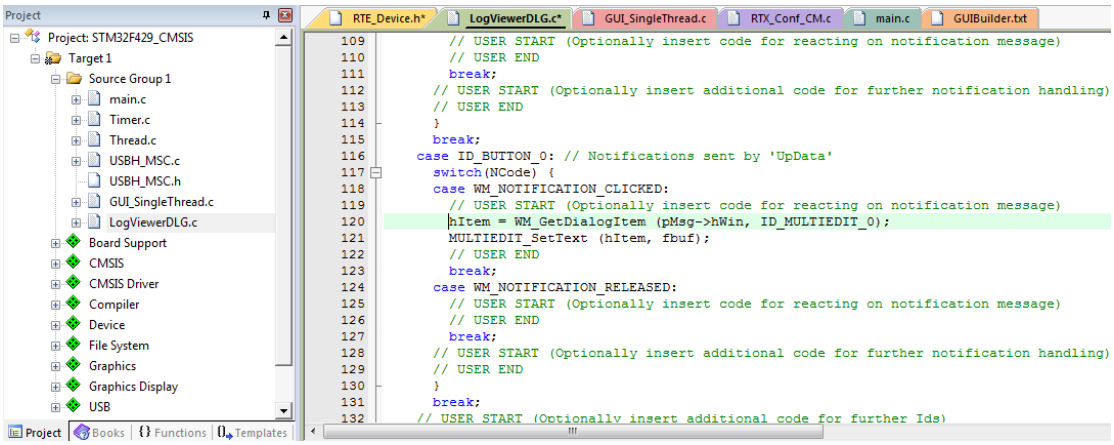


3、根据硬件原理图配置好硬件接口。




4、在“LogViewerDLG”中添加如下代码：

```
hItem = WM_GetDialogItem (pMsg->hWin, ID_MULTIEDIT_0);
MULTIEDIT_SetText (hItem, fbuf);
```




5、保存并编译工程。