从 0 开始移植 FreeRTOS

个人见解：我接触 STM32 时间不长，当初学的时候没有跑过这些小的系统，后来转而去学 arm9 的 linux 系统下面的驱动，进而接触到 linux 操作系统，觉得这些操作系统用起来确实很方便，比裸奔要好很多，多线程，虽然需要内存空间会大几 K，基本不影响程序运行。学 FreeRTOS，主要是公司要求，要使用免费的，所有老板都是这样，免费。

其实网上资料最多的是 UCOS 的系统，UCOS 系统是很稳定，我也自己移植成功了，不过呢也就多学点东西，UCOS-II 的移植，关键是不能用在商业上，商业上使用收费，虽然源码在你手上，但是这个确实也涉及到了侵权之类问题。UCOS-III 更坑，不给源码了，搞个球啊。

FreeRTOS 属于免费系统，开源，研究的人会越来越多，两者的移植我都试过，其实也都是申请空间，申请优先级，可以说两者使用从大的方面说没有太大区别。学会一种，另外一种也容易入手。FreeRTOS 关键就没有图形用户界面（GUI），比较弱势一点。

我在网上找了半天也没有看到像样的移植手册，对于刚刚入手 FreeRTOS 的人来说没有像样的例子，学起来就很坑。所以自己写了这个例程。

当然这里面可能也有些错误，那个各位发现了告诉我，联系我的 QQ。

移植好的文件在最后有下载链接。

目标：移植 LED 闪烁程序+AD 的 8 路采集程序

第一步:准备工作

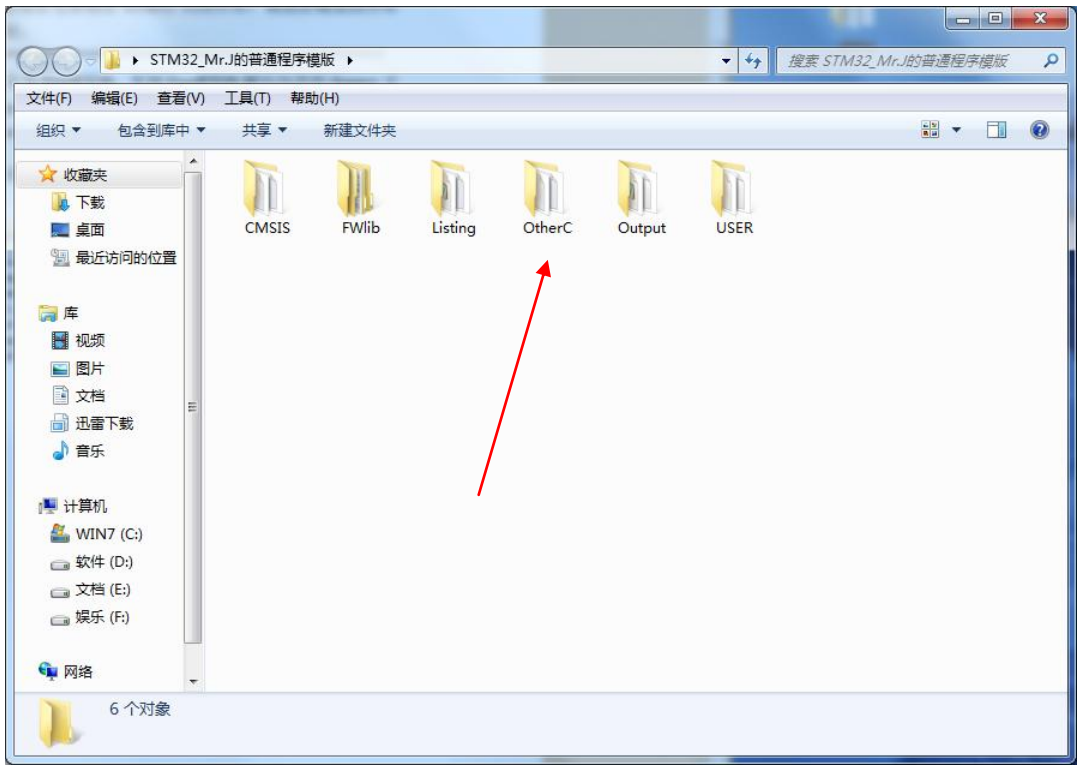
下载 FreeRTOS 的操作系统源代码，可以在官网下（注册比较麻烦），可以直接去我的 CSDN（百度直接搜索：FreeRTOS 系统的源文件，直接官网下载来的），这个版本是 2011 年出来的，还算比较稳定。我的基础程序模版参考了野火的 STM32 的模版，自己修改了，野火资料很多，支持开源。

我的模版（STM32_Mr.J 的普通程序模版）里面已经包含了移植需要的子程序。（ADC_8.c 与 LED.c）

(直接百度搜索：STM32_Mr.J 的普通程序模版

新浪的文件共享也有)为了区别我自己的 UCOS 与 FreeRTOS 模版的。源码和模版都要下载下来。

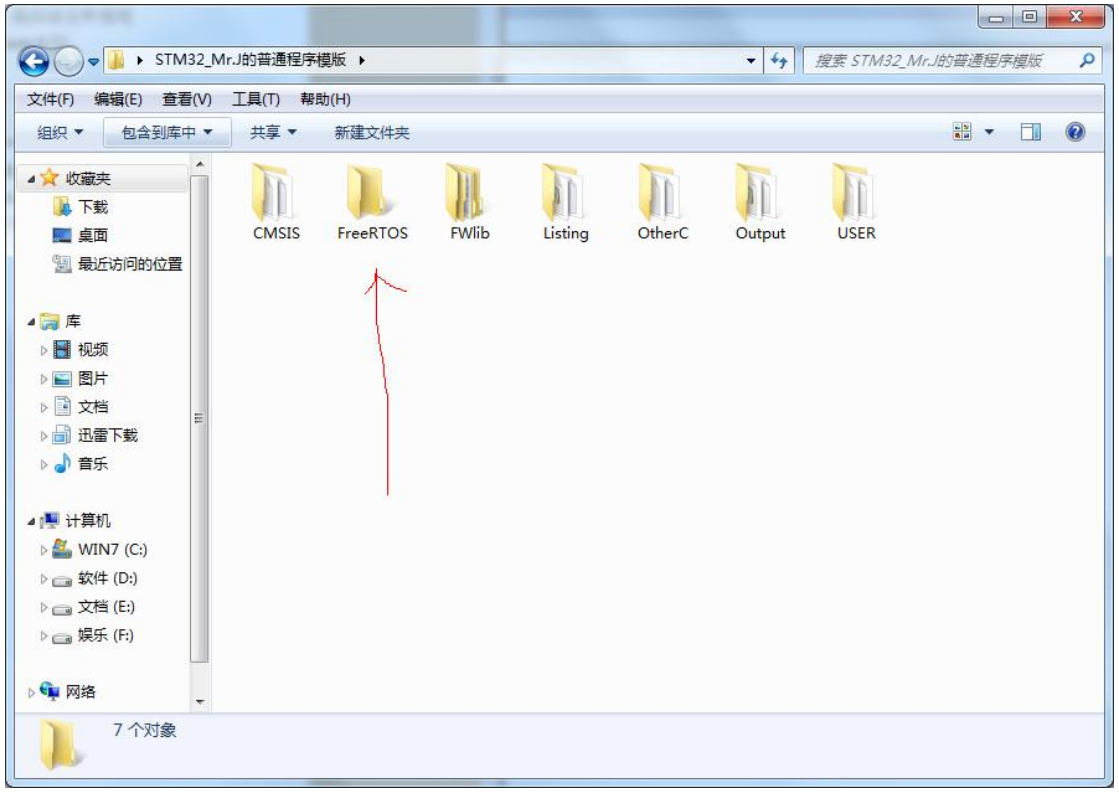
以我的普通程序模版为例一步一步完成 FreeRTOS 的移植。

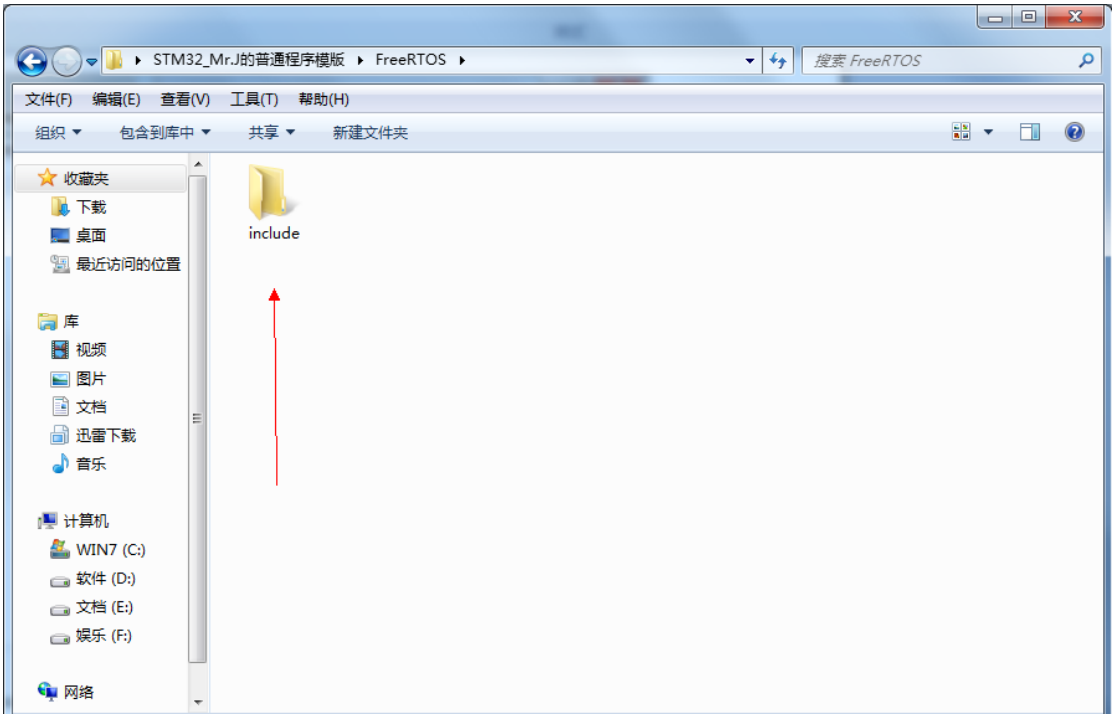


CMSIS,FWLib,Listing,Output,USER 文件夹是移植 STM32 的一些系统文件不用修改,里面为 STM32 的 3.5 库.各个子程序都放在在 OtherC 文件夹里面，加入的 c 文件与 h 文件都在这里。

第二步:开始移植

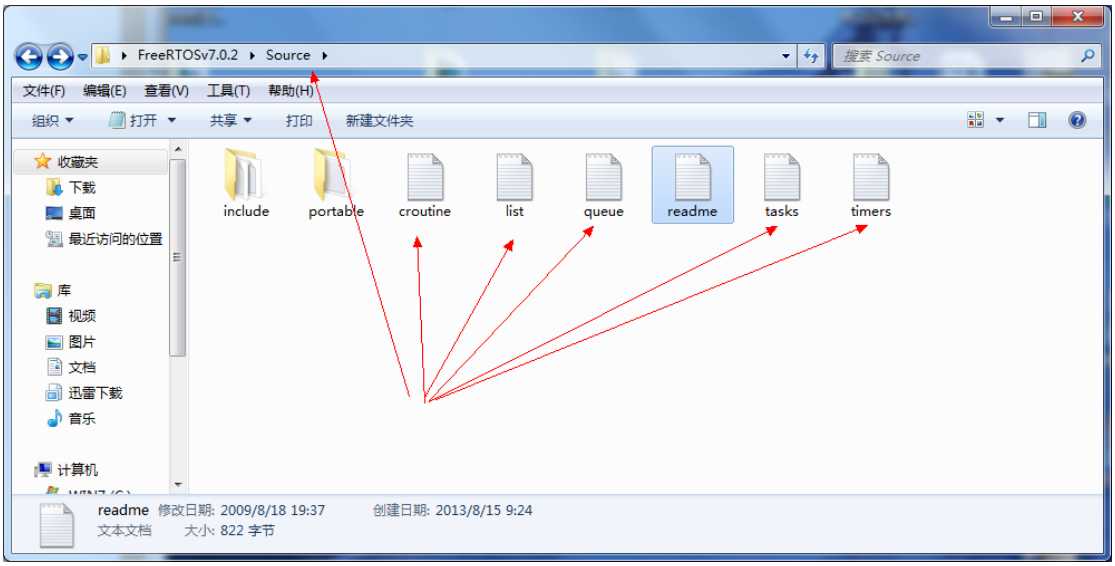
1）新建文件夹：在模版下新建文件夹 FreeRTOS 文件夹用来存放 FreeRTOS 的系统文件，再在 FreeRTOS 的文件夹下新建 include 文件夹 用来存放一些 h 文件，分开存放利于管理。





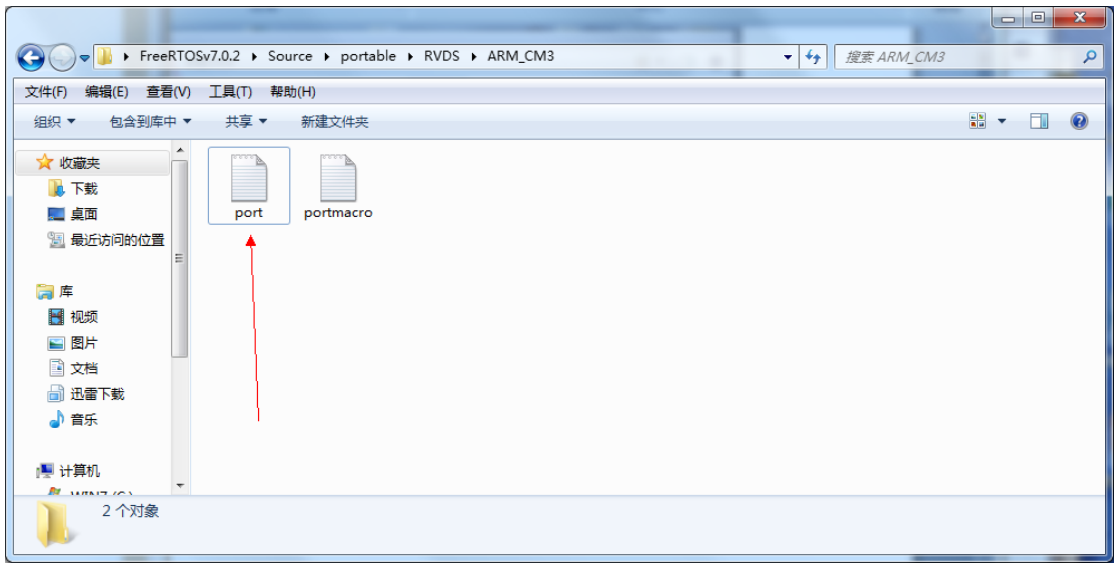
2）拷贝系统文件到指定文件夹（FreeRTOS 文件夹）中：

1. 从 FreeRTOS7.0.2 的源文件中找到 croutine.c, timers.c, list.c, queue.c, tasks.c 这五个源文件，位置为：FreeRTOS7.0.2->Source；拷贝文件至 FreeRTOS 文件夹。

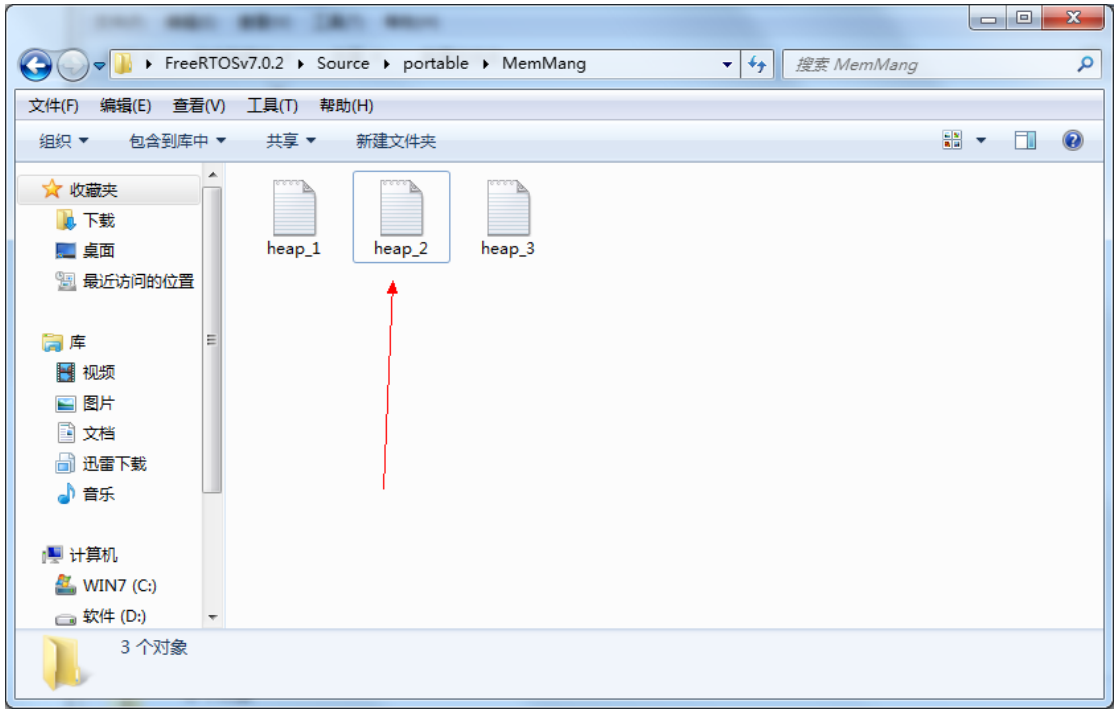


2. 从 FreeRTOS7.0.2 的源文件中找到 port.c 文件，位置 FreeRTOS->Source->portable->RVDS->ARM_CM3->port.c，拷贝至

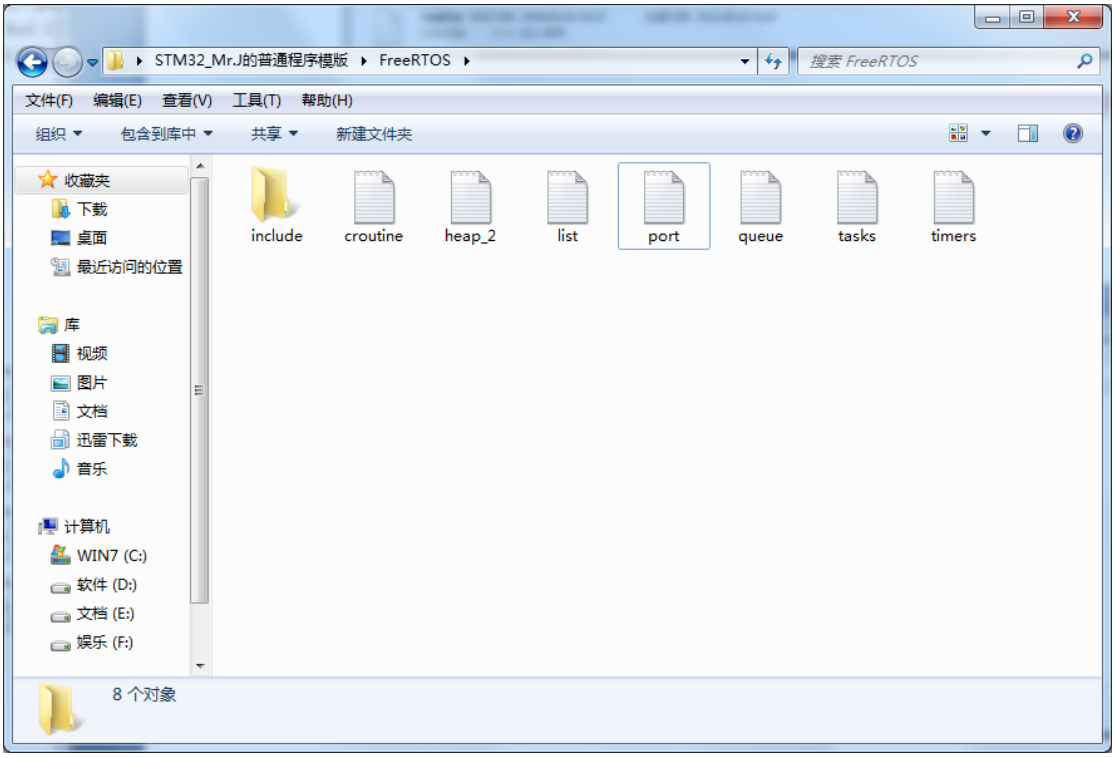
FreeRTOS 文件夹。



3. 从 FreeRTOS7.0.2 的源文件中找到 heap_2.c 文件，位置 Source->portable->MemMang->heap_2.c,拷贝至 FreeRTOS 文件夹。



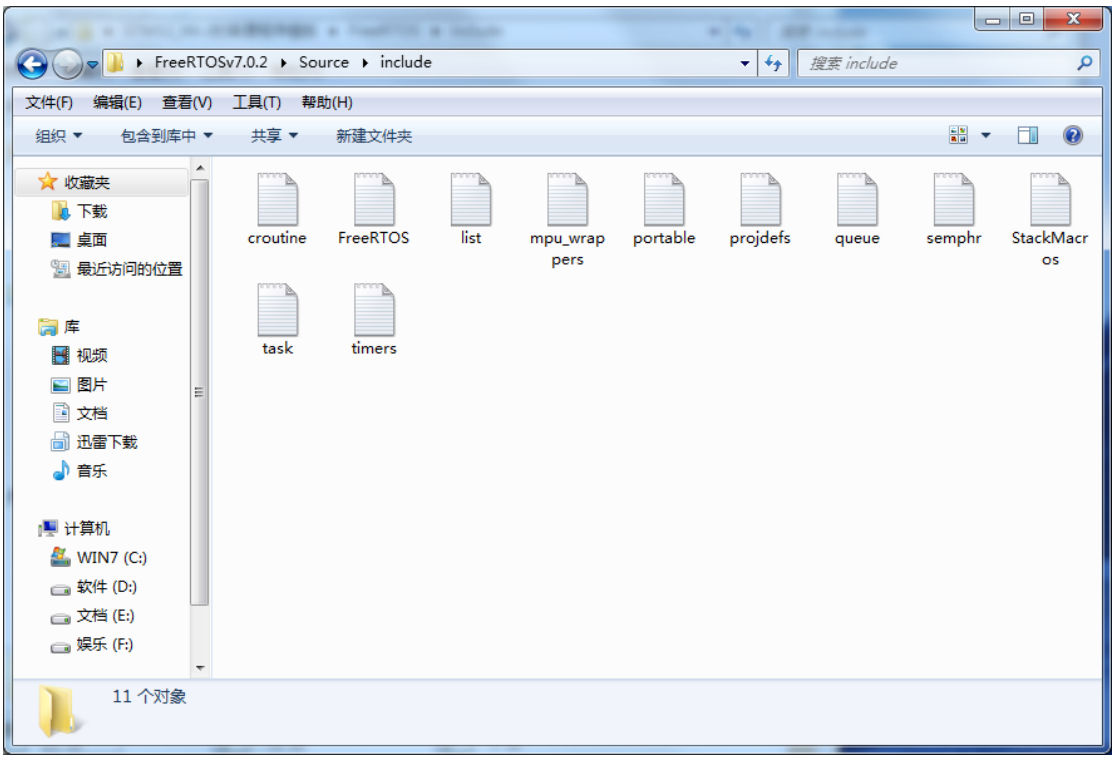
完成移动的 FreeRTOS 文件目录，就这几个文件.



3）拷贝.h 文件到指定文件夹（../FreeRTOS/include 文件夹）中：

1. 拷贝 FreeRTOS7.0.2 的源代码下的 include 文件夹里的全部.h 文件至新建立的 include 文件夹里，位置:

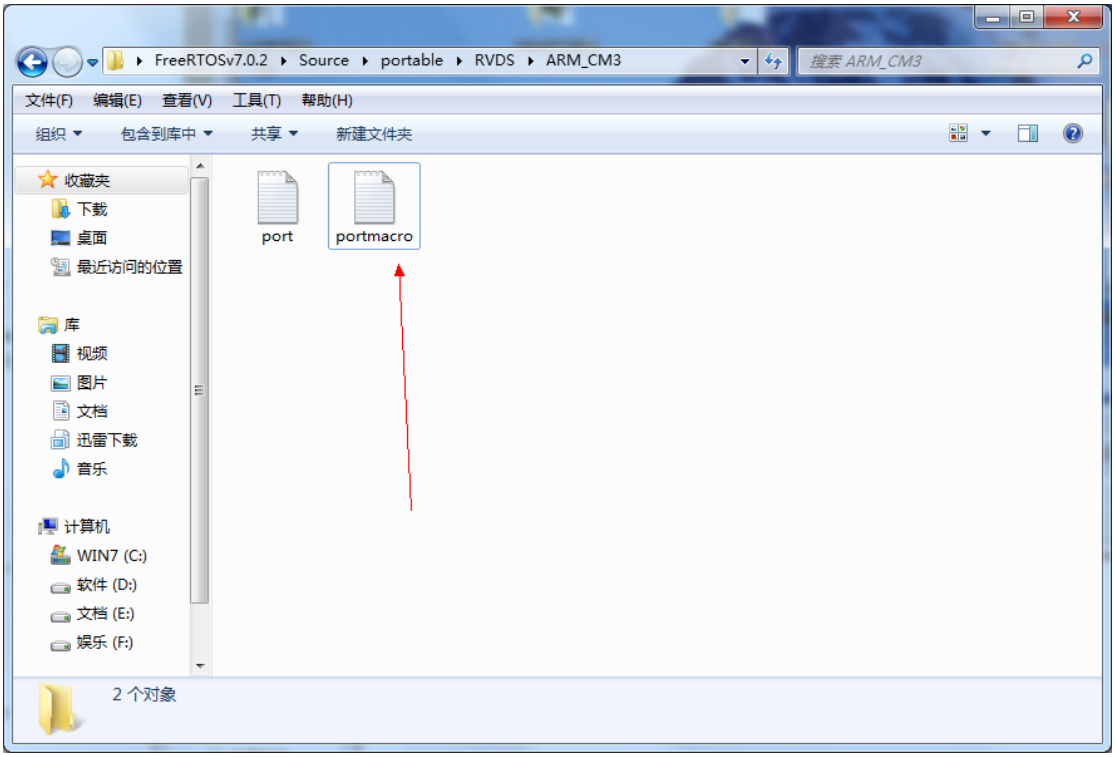
->FreeRTOSv7.0.2->Source->include



2. 从 FreeRTOS7.0.2 的源文件中找到 portmacro.h 文件拷贝至新建立的 include 文件夹里，文件位置

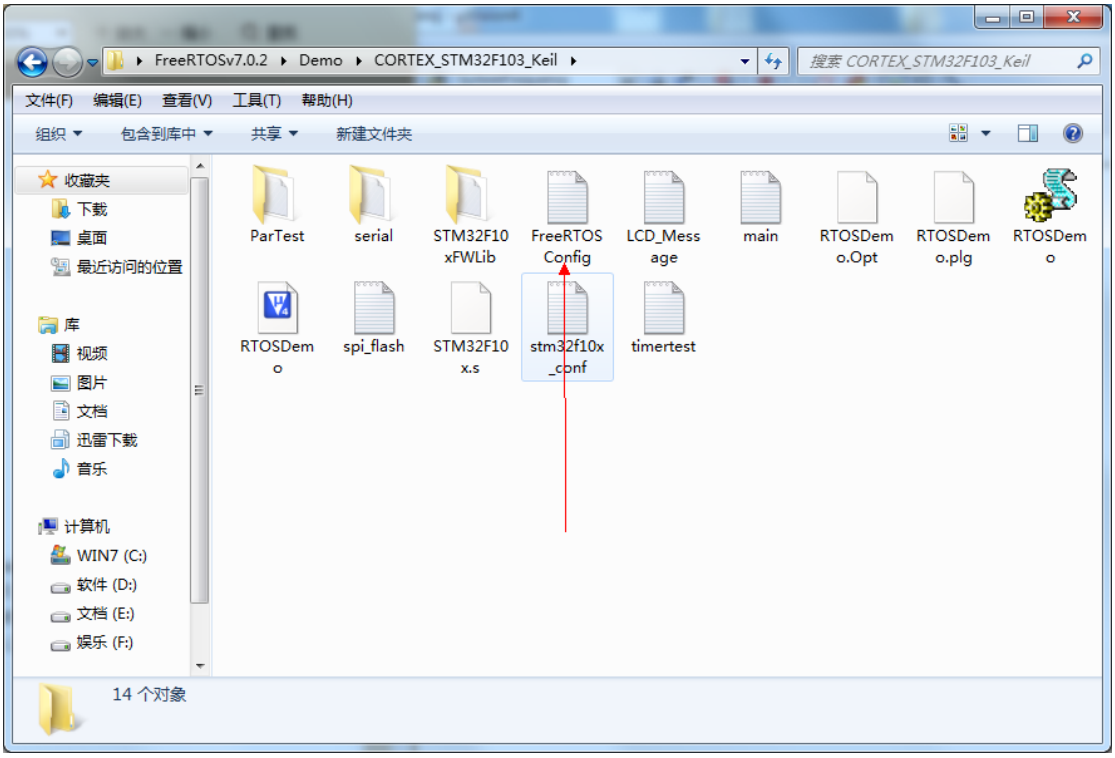
->FreeRTOSv7.0.2->Source->portable->RVDS->ARM_CM3

->portmacro.h

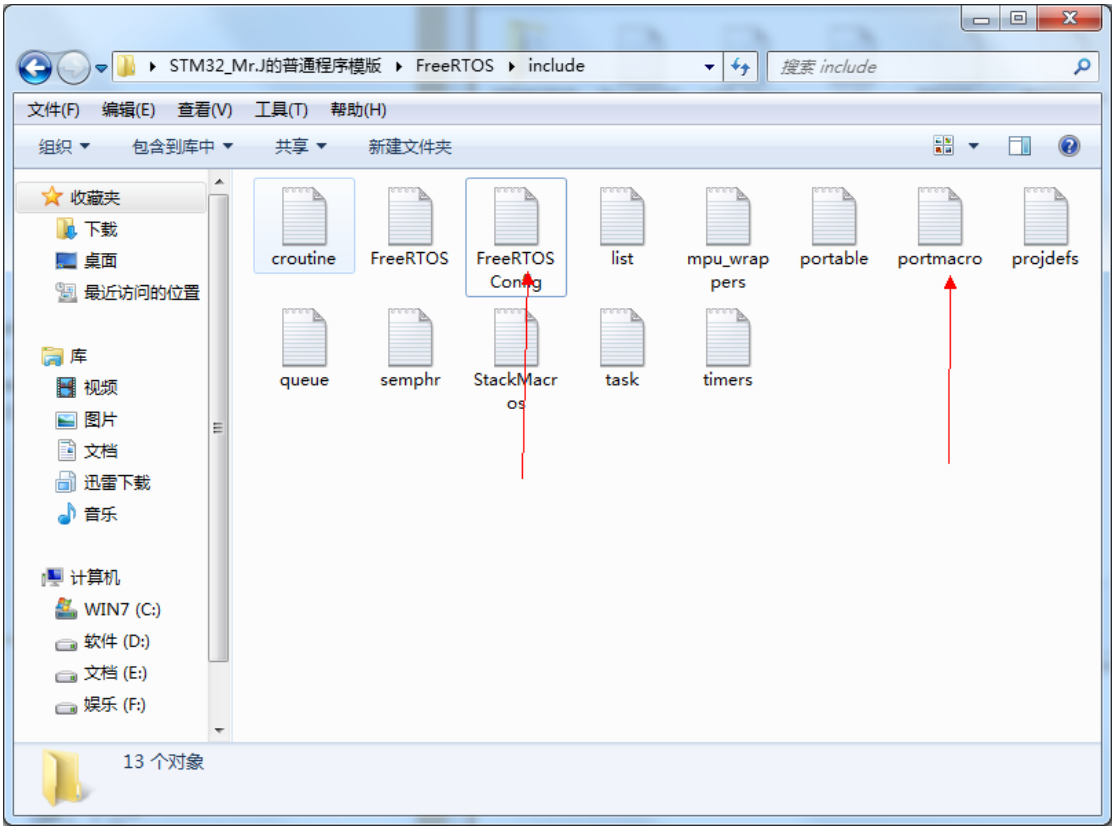


3. 由于我使用的是 Keil-MDK 作为开发工具，使用的是 STM32F103VET6 为目标芯片，所以需要从 FreeRTOS7.0.2 的源文件中找到匹配我这个开发工具与目标芯片的 config.h 的配置文件。这个文件在 Demo 文件夹中，把这个配置文件拷贝到 include 文件夹内，配置文件位置在 CORTEX_STM32F103_Keil 文件夹下：

->FreeRTOSv7.0.2->Demo->CORTEX_STM32F103_Keil->FreeRTOSConfig.h

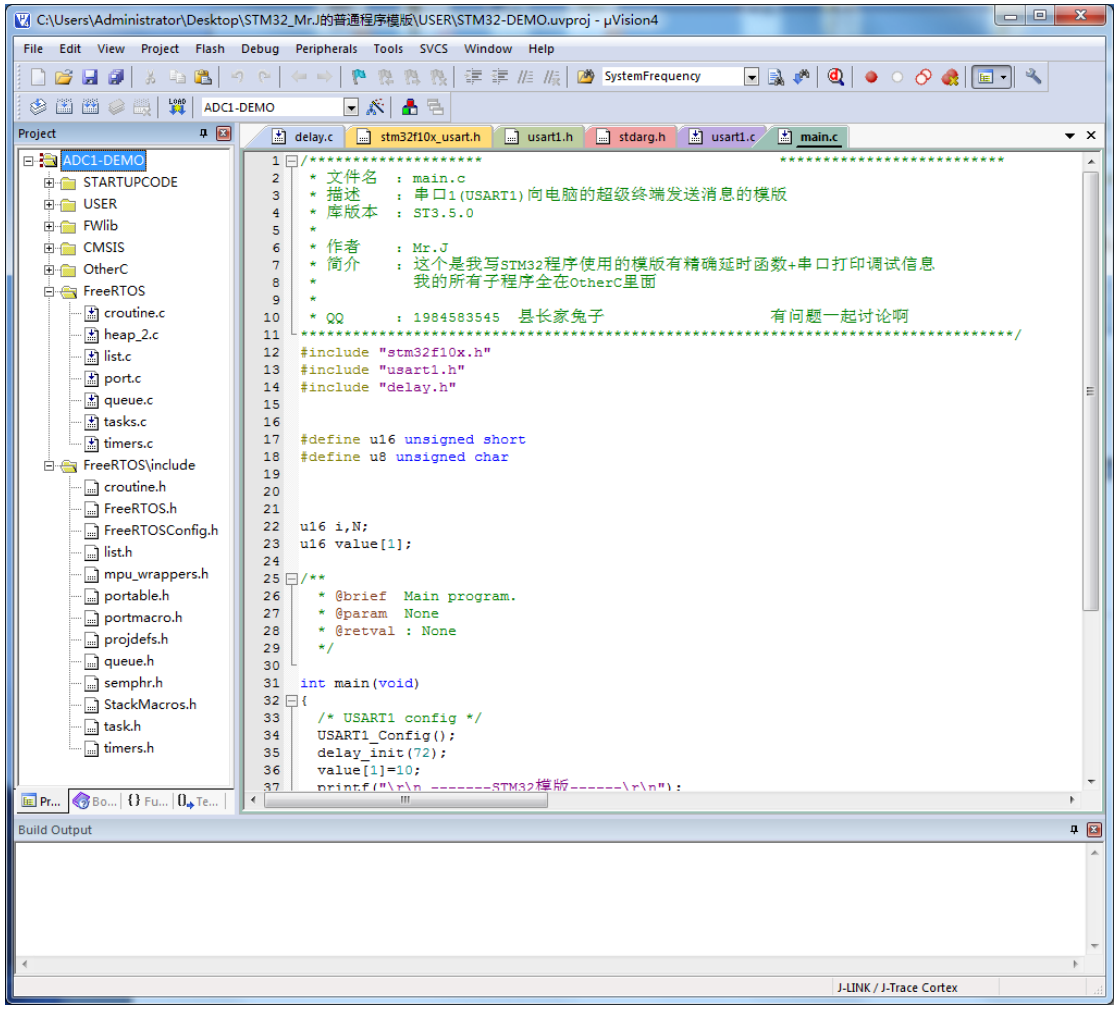


完成移动的 FreeRTOS 下的 include 文件目录，就这几个文件.



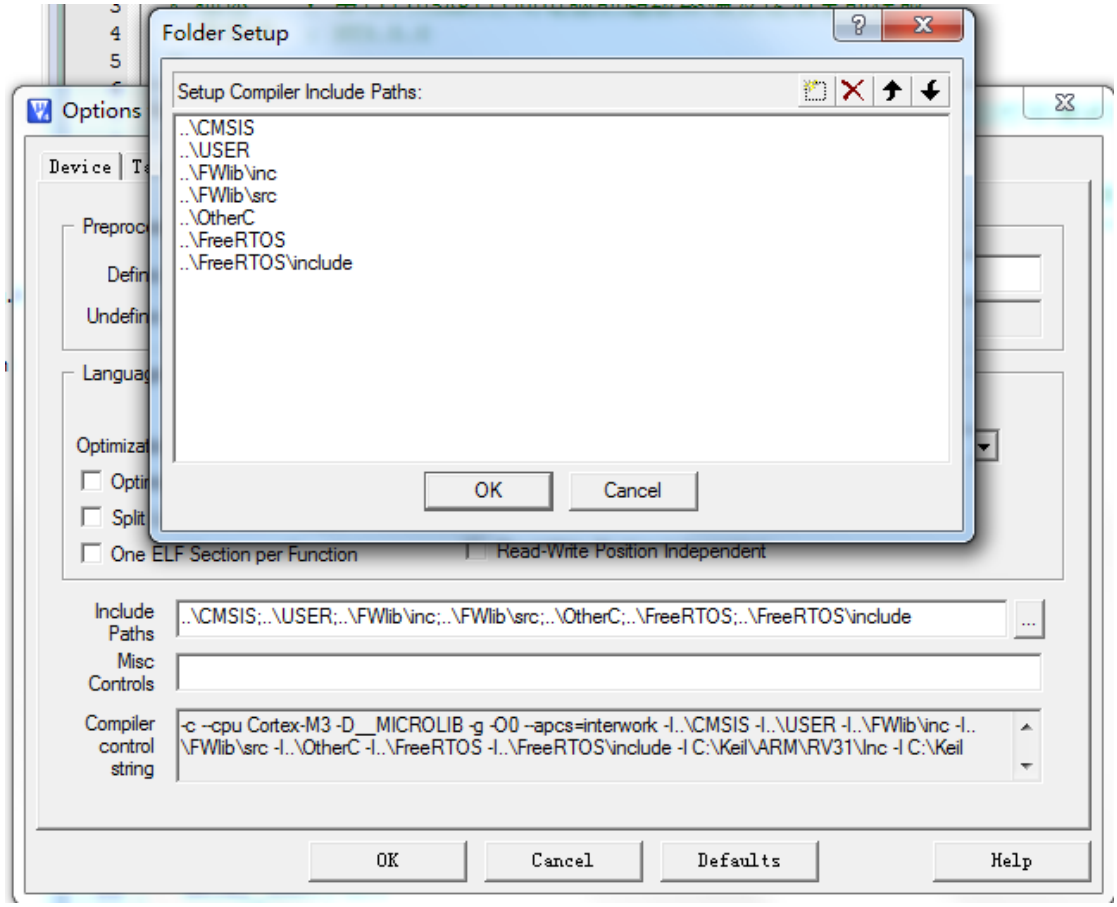
4）在工程中添加各系统文件

在工程中建立 FreeRTOS 文件夹，添加工程模版中 FreeRTOS 的所有文件；再在工程中建立 FreeRTOS\include 文件夹，表示此文件夹的文件在 FreeRTOS 的子文件夹 include 文件夹内，添加 include 文件夹内的.h 文件进去，下图是添加完成的工作空间。



5）修改编译环境与启动代码

菜单栏找到 Project->options for target->C/C++->Include Paths 中加入上述的 include 文件夹，当然 FreeRTOS 文件夹最好也添上，防止遗漏。如下图：



启动文件也需要修改：startup_stm32f10x_hd.s 中等密度的

```
34
35  Stack_Size      EQU     0x00000400
36
37                  AREA    STACK, NOINIT, READWRITE, ALIGN=3
38  Stack_Mem       SPACE   Stack_Size
39  __initial_sp
40
41  ; <h> Heap Configuration
42  ;   <o>  Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
43  ; </h>
44
45  Heap_Size       EQU     0x00000200
46
47                  AREA    HEAP, NOINIT, READWRITE, ALIGN=3
48  __heap_base
49  Heap_Mem        SPACE   Heap_Size
50  __heap_limit
51
52                  PRESERVE8
53                  THUMB
54
55
56  ; Vector Table Mapped to Address 0 at Reset
57                  AREA    RESET, DATA, READONLY
58                  EXPORT  __Vectors
59                  EXPORT  __Vectors_End
60                  EXPORT  __Vectors_Size
61
62  __Vectors       DCD     __initial_sp              ; Top of Stack
63                  DCD     Reset_Handler             ; Reset Handler
64                  DCD     NMI_Handler               ; NMI Handler
65                  DCD     HardFault_Handler         ; Hard Fault Handler
66                  DCD     MemManage_Handler         ; MPU Fault Handler
67                  DCD     BusFault_Handler          ; Bus Fault Handler
68                  DCD     UsageFault_Handler        ; Usage Fault Handler
```

在__heap_limit 下面添加：

```
__heap_limit

                    PRESERVE8

                    THUMB

                    IMPORT xPortPendSVHandler

                    IMPORT xPortSysTickHandler

                    IMPORT vPortSVCHandler
```

另外还有就是这三句代码也得修改

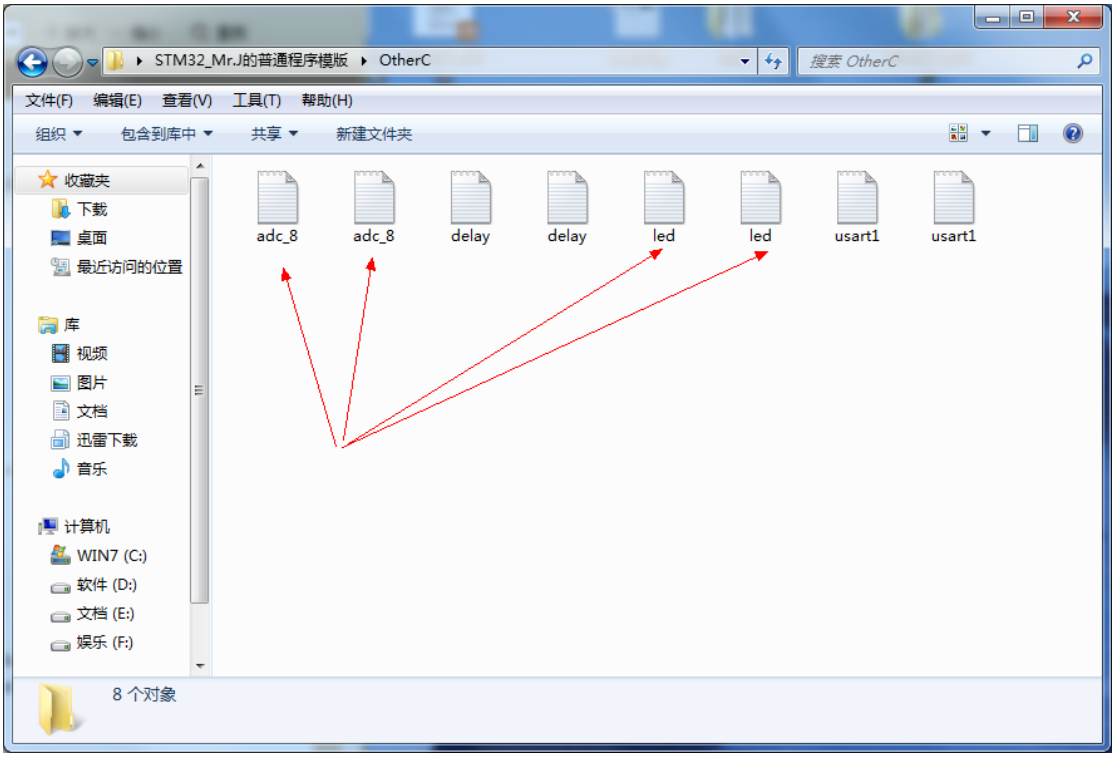
```
DCD SVC_Handler          ->          DCD vPortSVCHandler

DCD PendSV_Handler       ->          DCD xPortPendSVHandler

DCD SysTick_Handler      ->          DCD xPortSysTickHandler
```

```
                EXPORT  __Vectors
                EXPORT  __Vectors_End
                EXPORT  __Vectors_Size

__Vectors       DCD     __initial_sp               ; Top of Stack
                DCD     Reset_Handler              ; Reset Handler
                DCD     NMI_Handler                ; NMI Handler
                DCD     HardFault_Handler          ; Hard Fault Handler
                DCD     MemManage_Handler          ; MPU Fault Handler
                DCD     BusFault_Handler           ; Bus Fault Handler
                DCD     UsageFault_Handler         ; Usage Fault Handler
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     SVC_Handler                ; SVCall Handler
                DCD     DebugMon_Handler           ; Debug Monitor Handler
                DCD     0                          ; Reserved
                DCD     PendSV_Handler             ; PendSV Handler
                DCD     SysTick_Handler            ; SysTick Handler

                ; External Interrupts
                DCD     WWDG_IRQHandler            ; Window Watchdog
                DCD     PVD_IRQHandler             ; PVD through EXTI Line detect
                DCD     TAMPER_IRQHandler          ; Tamper
                DCD     RTC_IRQHandler             ; RTC
                DCD     FLASH_IRQHandler           ; Flash
                DCD     RCC_IRQHandler             ; RCC
                DCD     EXTI0_IRQHandler           ; EXTI Line 0
                DCD     EXTI1_IRQHandler           ; EXTI Line 1
                DCD     EXTI2_IRQHandler           ; EXTI Line 2
                DCD     EXTI3_IRQHandler           ; EXTI Line 3
                DCD     EXTI4_IRQHandler           ; EXTI Line 4
                DCD     DMA1_Channel1_IRQHandler   ; DMA1 Channel 1
                DCD     DMA1_Channel2_IRQHandler   ; DMA1 Channel 2
```

对应修改之后的文件

```

__Vectors       DCD     __initial_sp               ; Top of Stack
                DCD     Reset_Handler              ; Reset Handler
                DCD     NMI_Handler                ; NMI Handler
                DCD     HardFault_Handler          ; Hard Fault Handler
                DCD     MemManage_Handler          ; MPU Fault Handler
                DCD     BusFault_Handler           ; Bus Fault Handler
                DCD     UsageFault_Handler         ; Usage Fault Handler
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                DCD     0                          ; Reserved
                ; DCD     SVC_Handler               ; SVCall Handler
                DCD vPortSVCHandler
                DCD     DebugMon_Handler           ; Debug Monitor Handler
                DCD     0                          ; Reserved
                ; DCD     PendSV_Handler            ; PendSV Handler
                DCD xPortPendSVHandler
                ; DCD     SysTick_Handler           ; SysTick Handler
                DCD xPortSysTickHandler
                ; External Interrupts
                DCD     WWDG_IRQHandler            ; Window Watchdog
                DCD     PVD_IRQHandler             ; PVD through EXTI Line detect
                DCD     TAMPER_IRQHandler          ; Tamper
                DCD     RTC_IRQHandler             ; RTC
                DCD     FLASH_IRQHandler           ; Flash
                DCD     RCC_IRQHandler             ; RCC
                DCD     EXTI0_IRQHandler           ; EXTI Line 0
                DCD     EXTI1_IRQHandler           ; EXTI Line 1
                DCD     EXTI2_IRQHandler           ; EXTI Line 2
                DCD     EXTI3_IRQHandler           ; EXTI Line 3
                DCD     EXTI4_IRQHandler           ; EXTI Line 4
                DCD     DMA1_Channel1_IRQHandler   ; DMA1 Channel 1
                DCD     DMA1_Channel2_IRQHandler   ; DMA1 Channel 2
                DCD     DMA1_Channel3_IRQHandler   ; DMA1 Channel 3
```

6）移植 LED 闪烁程序+AD 的 8 路采集程序

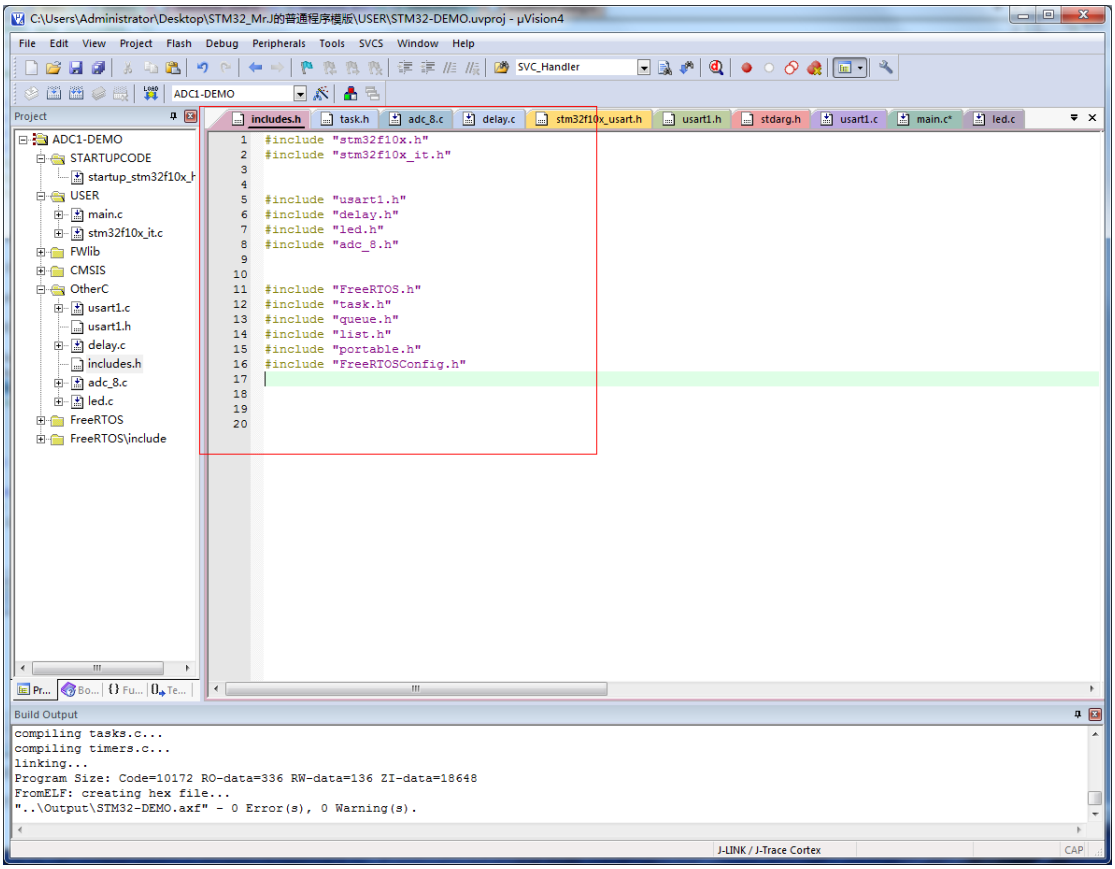


把 LED 闪烁子程序与 adc 的 8 路采集程序拷贝到模版的 OtherC 文件夹，再新建一个 include 文件也同样放在这个 OtherC 文件夹内，工程中添加 includes.h 文件。

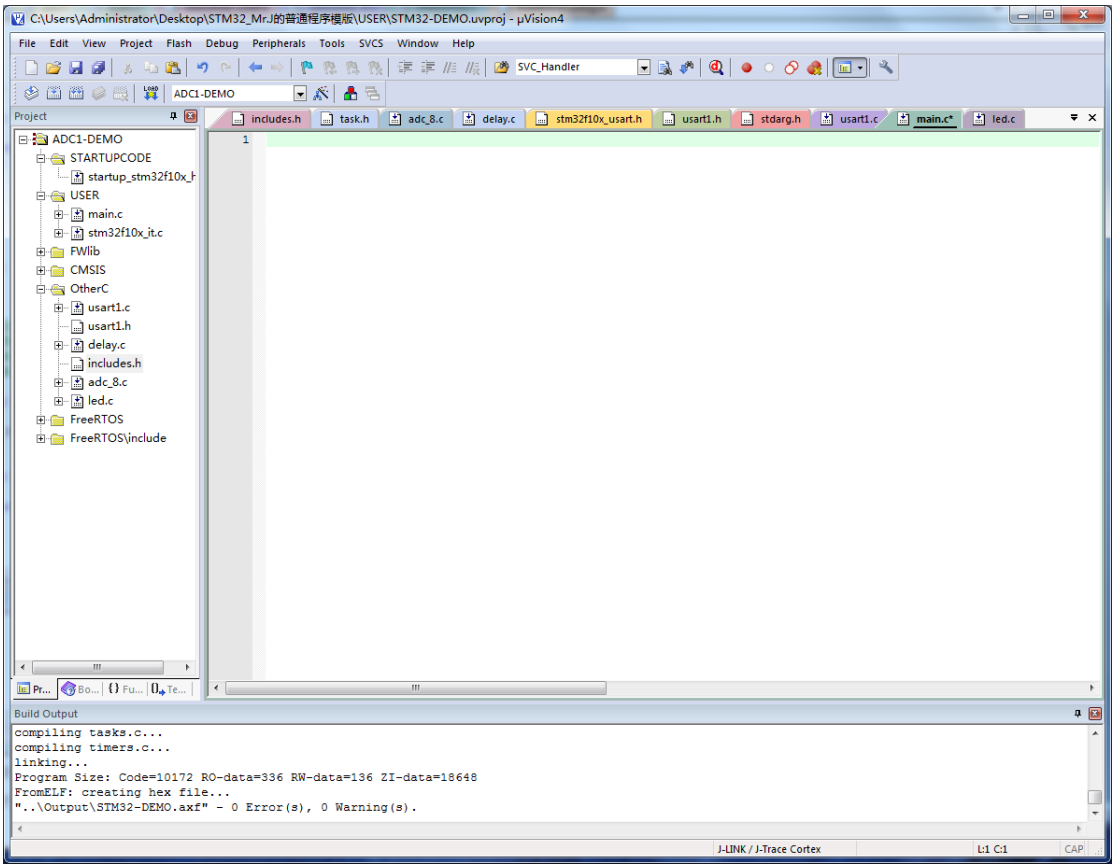
includes.h

```
#include "stm32f10x.h"
#include "stm32f10x_it.h"
#include "usart1.h"
#include "delay.h"
#include "led.h"
#include "FreeRTOS.h"
#include "task.h"
#include "queue.h"
```

```
#include "list.h"

#include "portable.h"

#include "FreeRTOSConfig.h"
```

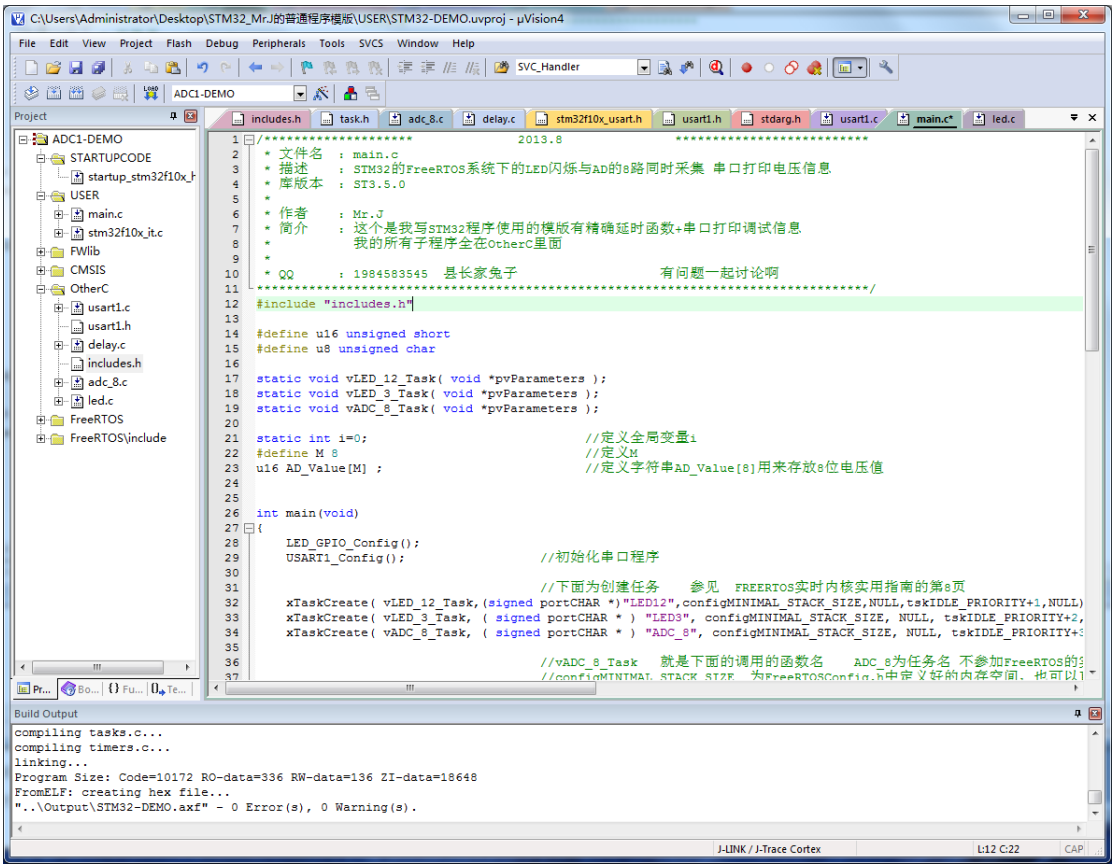


由于我的 LED.c 与 ADC_8.c 文件是可以直接调用的，所以现在只需修改 main.c 文件就可以了，先删除我的原模版内的代码

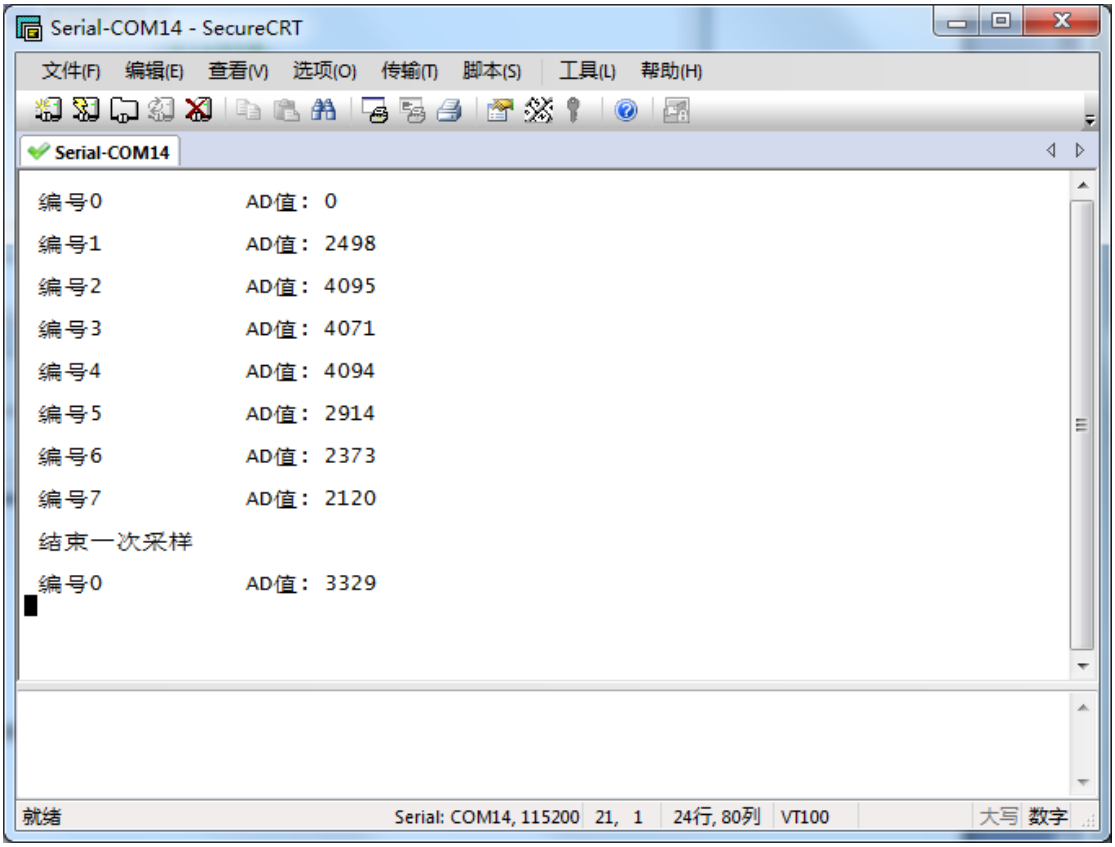


最后修改我自己的 main 文件，完成系统移植。

完成添加的文件信息



串口打印的电压信息：



Main.C 文件的实际代码：

```
/*********************                                    2013.8

**************************

 * 文件名   ：main.c

 * 描述     ：STM32 的 FreeRTOS 系统下的 LED 闪烁与 AD 的 8 路同时采集 串口打印电压信息

 * 库版本   ：ST3.5.0

 *

 * 作者     ：Mr.J

 * 简介     ：这个是我写 STM32 程序使用的模版有精确延时函数+
```

串口打印调试信息

*　　我的所有子程序全在 OtherC 里面

*

* QQ　　：1984583545　县长家兔子

　　有问题一起讨论啊

**********/

```
#include "includes.h"
#define u16 unsigned short
#define u8 unsigned char
static void vLED_12_Task( void *pvParameters );
static void vLED_3_Task( void *pvParameters );
static void vADC_8_Task( void *pvParameters );
int main(void)
{
    LED_GPIO_Config();
    USART1_Config();                    //初始化串口程序
                                                    //
```

下面为　创建任务　　参见 FREERTOS 实时内核实用指南　的第 8 页

(百度文库下载地址 http://wenku.baidu.com/view/58418d21482fb4daa58d4b18.html)

```
    xTaskCreate(        vLED_12_Task,(signed        portCHAR
*)"LED12",configMINIMAL_STACK_SIZE,NULL,tskIDLE_PRIORITY+1
```

```
,NULL);
    xTaskCreate( vLED_3_Task, ( signed portCHAR * ) "LED3",
configMINIMAL_STACK_SIZE, NULL, tskIDLE_PRIORITY+2, NULL );
     xTaskCreate( vADC_8_Task, ( signed portCHAR * ) "ADC_8",
configMINIMAL_STACK_SIZE, NULL, tskIDLE_PRIORITY+3, NULL
```

//vADC_8_Task 就是下面的调用的函数名 ADC_8 为任务名不参加 FreeRTOS 的实际运行

//configMINIMAL_STACK_SIZE 为 FreeRTOSConfig.h 中定义好的内存空间，也可以直接给一个实际值比如 50

//tskIDLE_PRIORITY+1 表示优先级的申请粑 skIDLE_PRIORITY 初始值为 0

```
    vTaskStartScheduler();                  // 启动调度器,任务开
始执行
     return 0;
}
void  vLED_12_Task(void *pvParameters)
{
     while(1)
     {
```

```
            LED1(ON);
            LED2(OFF);
            vTaskDelay(500/portTICK_RATE_MS);          //延时
多少个心跳骤起  延时 500 个毫秒
            LED1(OFF);
            LED2(ON);
            vTaskDelay(500/portTICK_RATE_MS);
      }
}

void  vLED_3_Task(void *pvParameters)
{
      while(1)
      {
            LED3(ON);

            vTaskDelay(500/portTICK_RATE_MS);

            LED3(OFF);
            vTaskDelay(500/portTICK_RATE_MS);
      }
}
```

```
void  vADC_8_Task(void *pvParameters)
                    //AD 的八路采集程序，使用 DMA 方式同时采集
八路
{
      ADC1_Init();
       printf("\r\n **开始采样**\r\n");
       while(1)
       {
         AD_Start();
          vTaskDelay(500/portTICK_RATE_MS);
       }
}
```

在此完成移植

移植好的文件

新浪下载：STM32_Mr.J 的 FreeRTOS 系统采集

CSDN 下载：STM32 的 FreeRTOS 系统多路采集

jiangbigood 新浪博客　　1984583545　县长家兔子