

FSC_STOS 移植教程

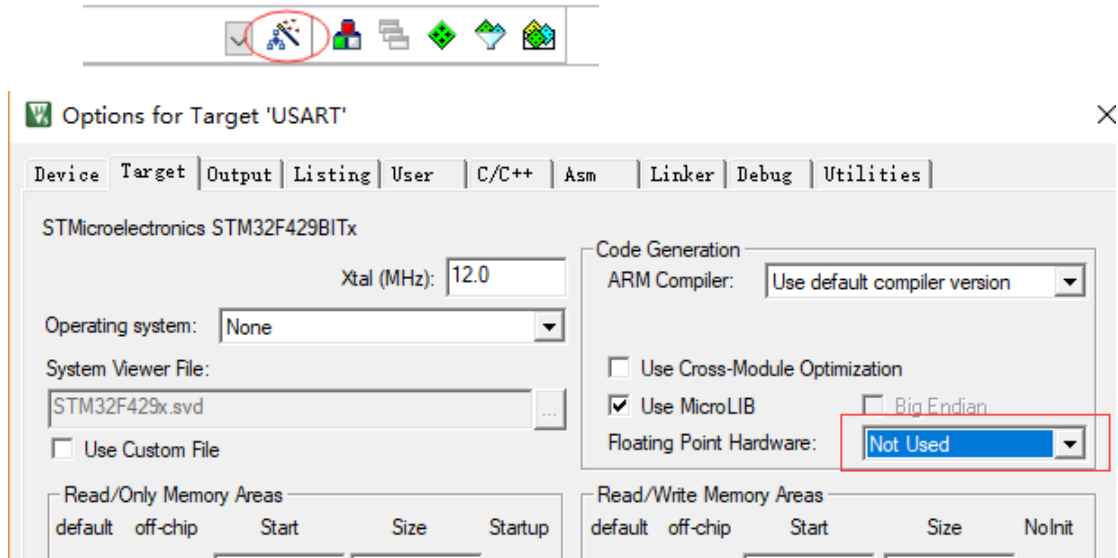
——望穿秋水

在裸机工程中加入 FSC_STOS 内核:

可分为三个内容: <1>stm32fxxx_it.c 修改。<2>main.c 修改。<3>fsc_stos.h 修改。

以 stm32f429 裸机工程加入 FSC_STOS 为例: (F4 中要关闭 FPU)

关闭 FPU 设置(无 FPU 的请忽略):

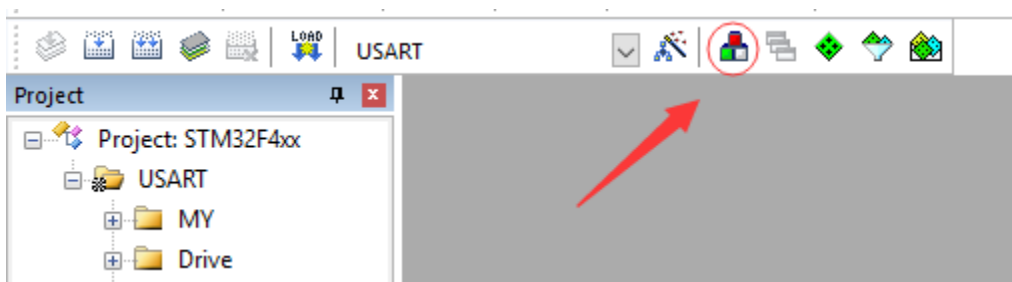


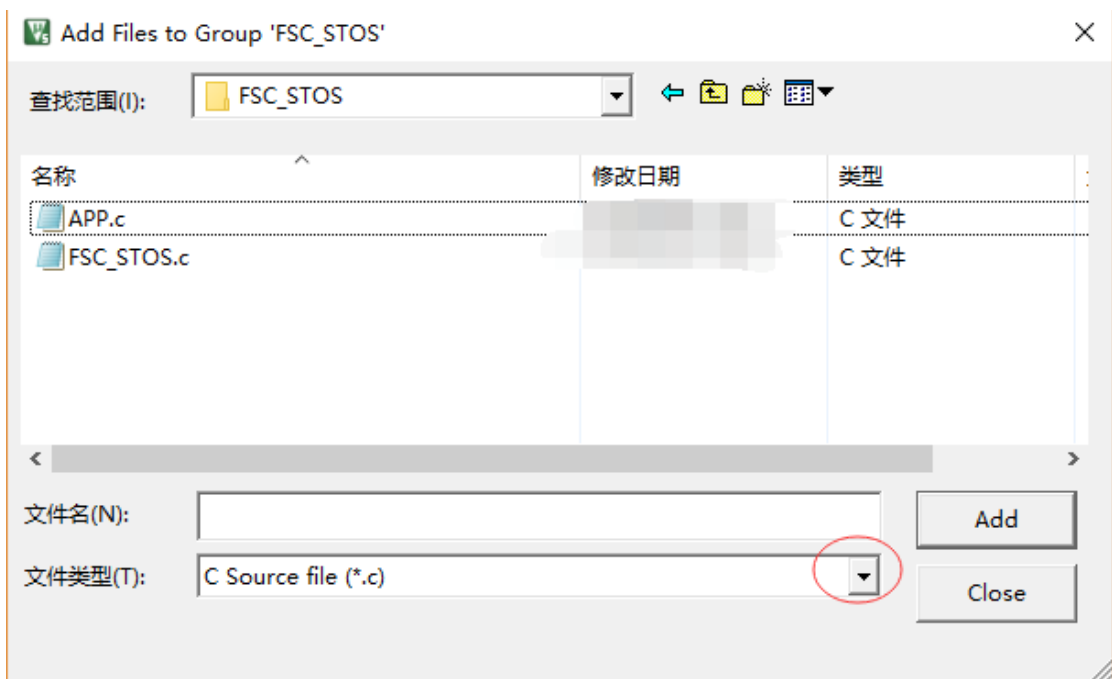
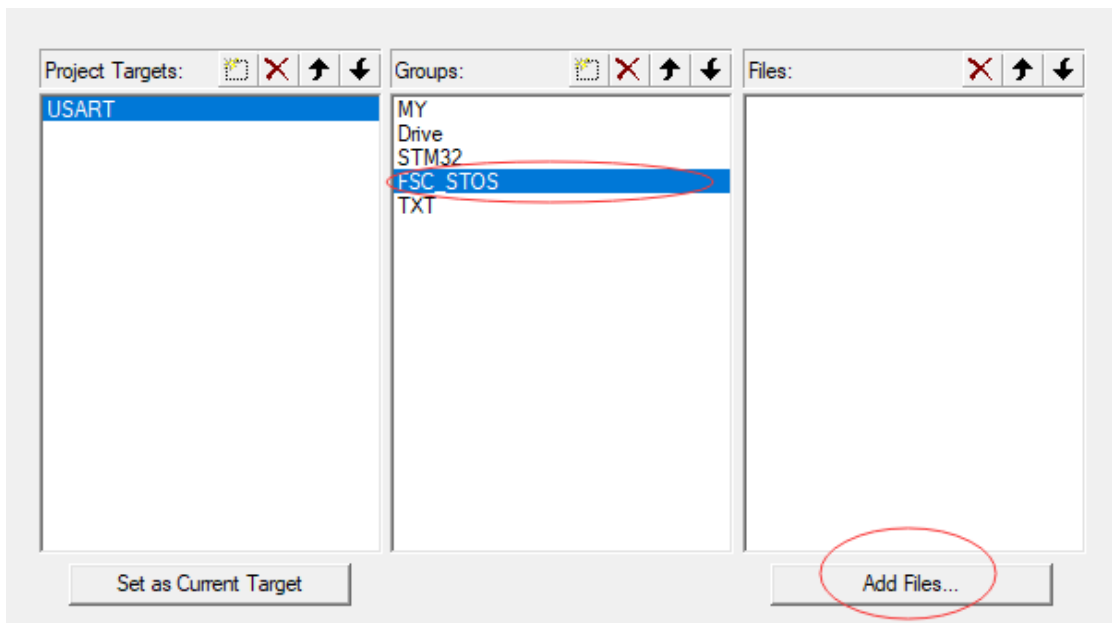
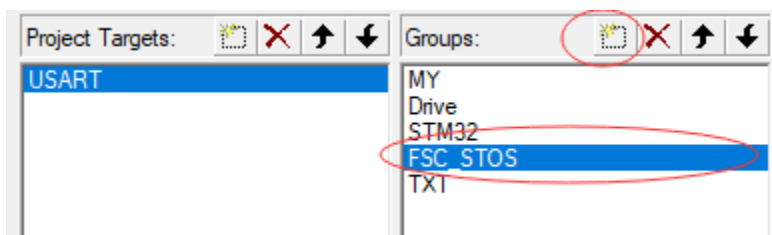
1. 添加文件:

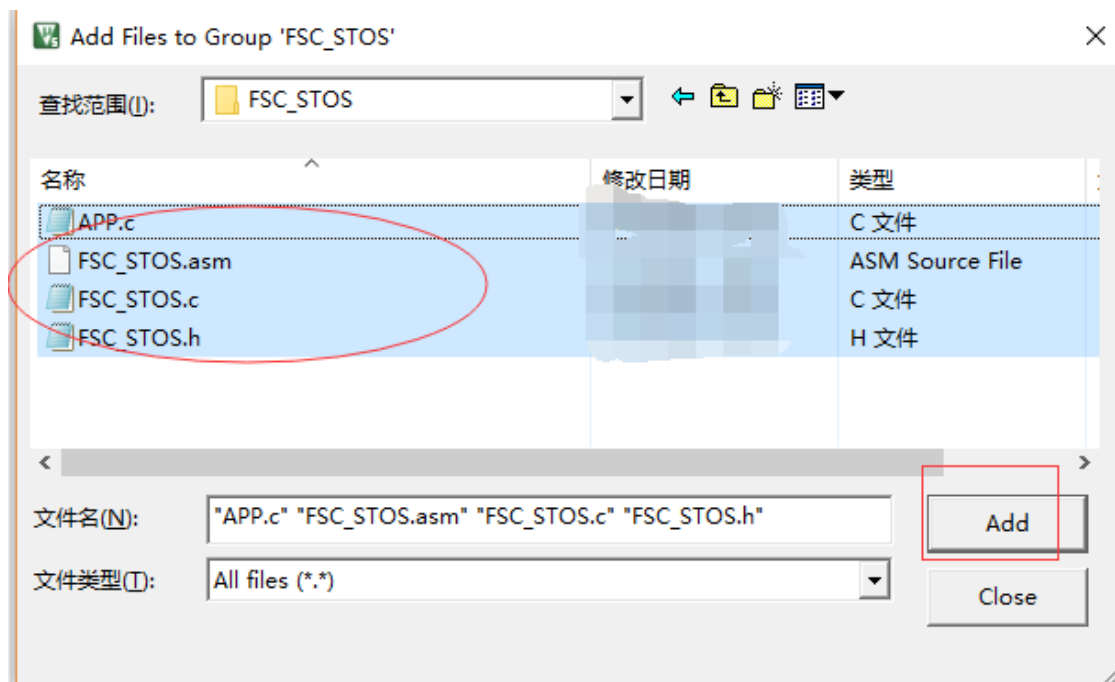
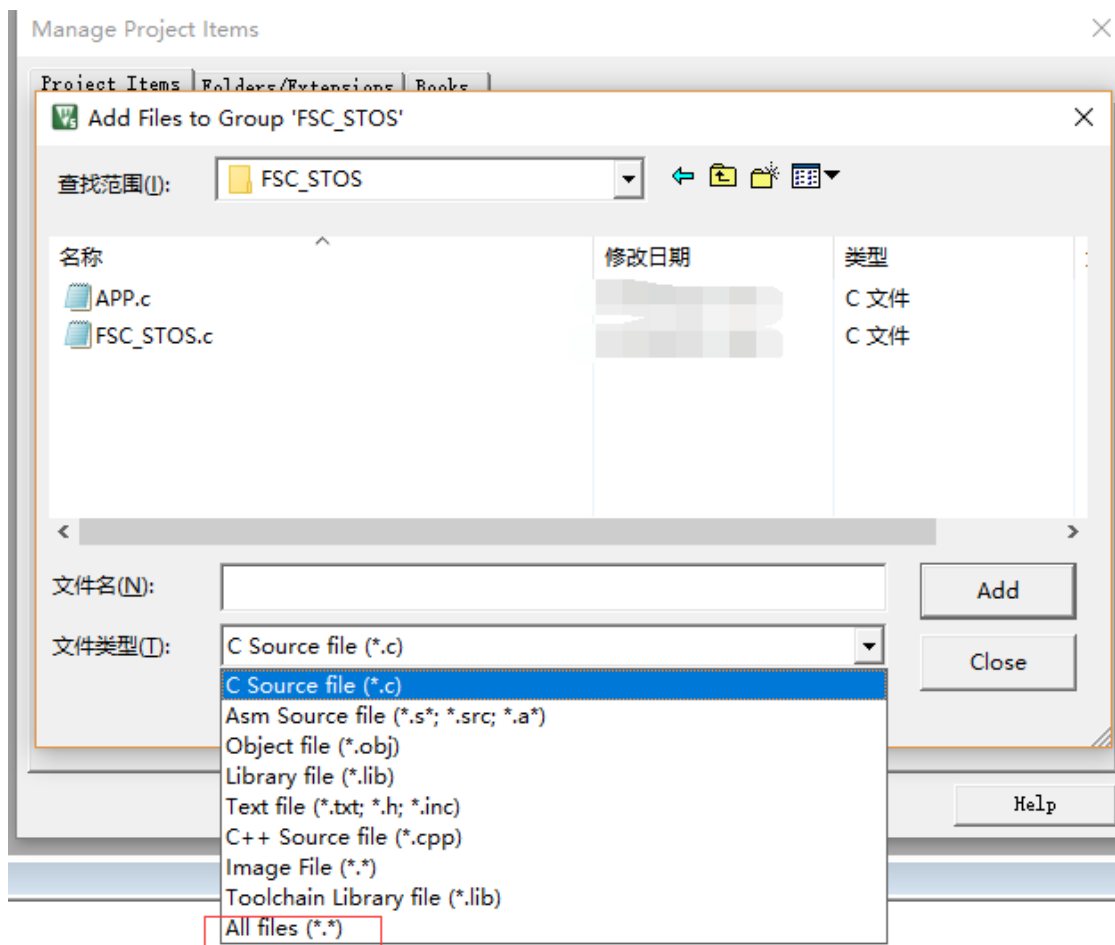
复制 FSC_STOS 内核移植专用文件夹内的 FSC_STOS 文件夹到裸机工程, 如图

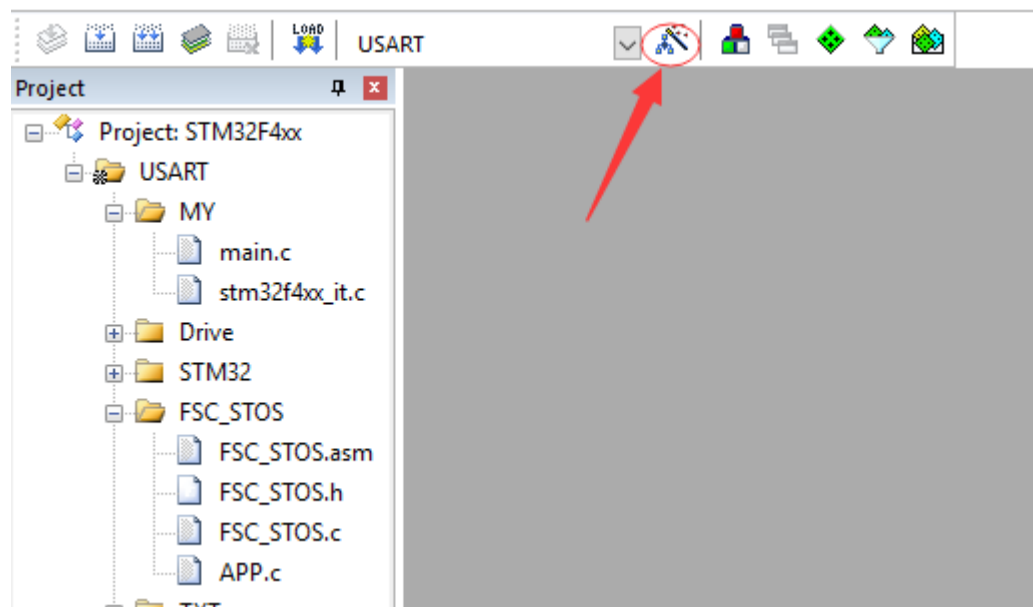
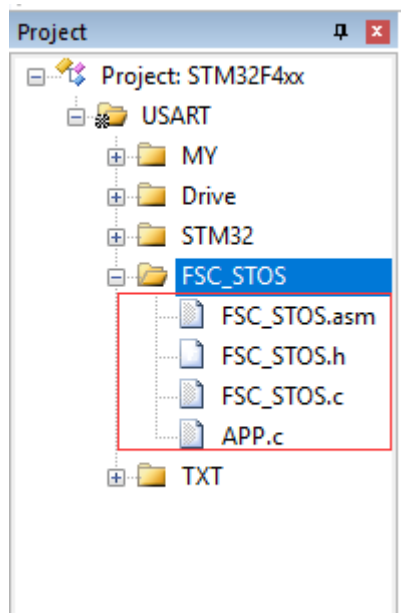


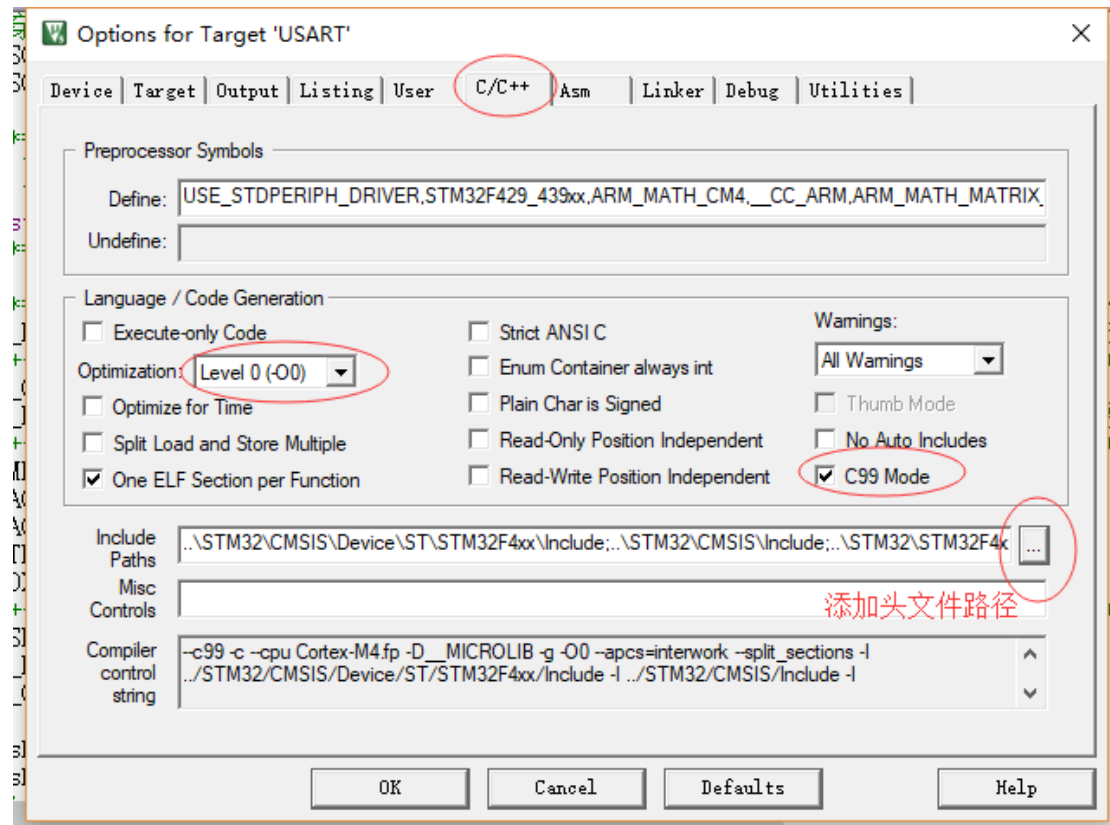
打开 MDK 裸机工程, 在 MDK 工程中添加相关文件和路径。



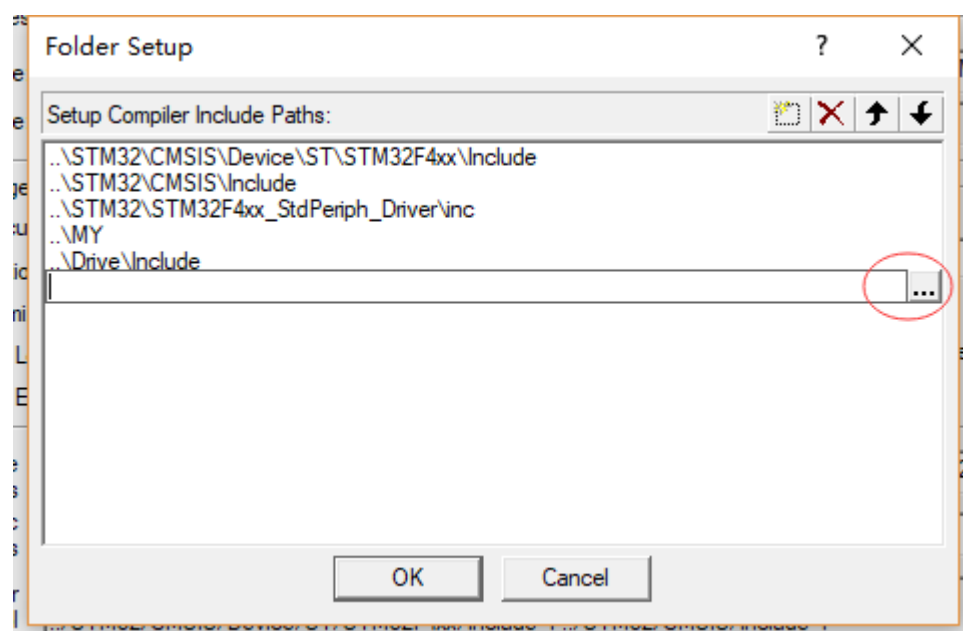
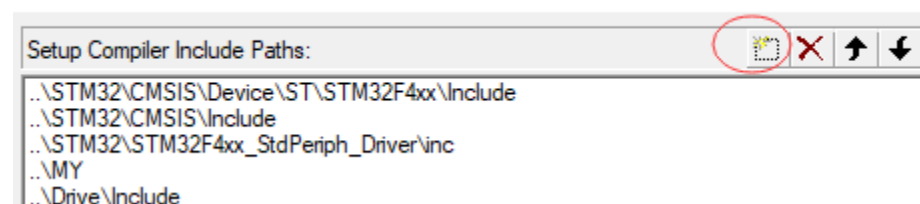


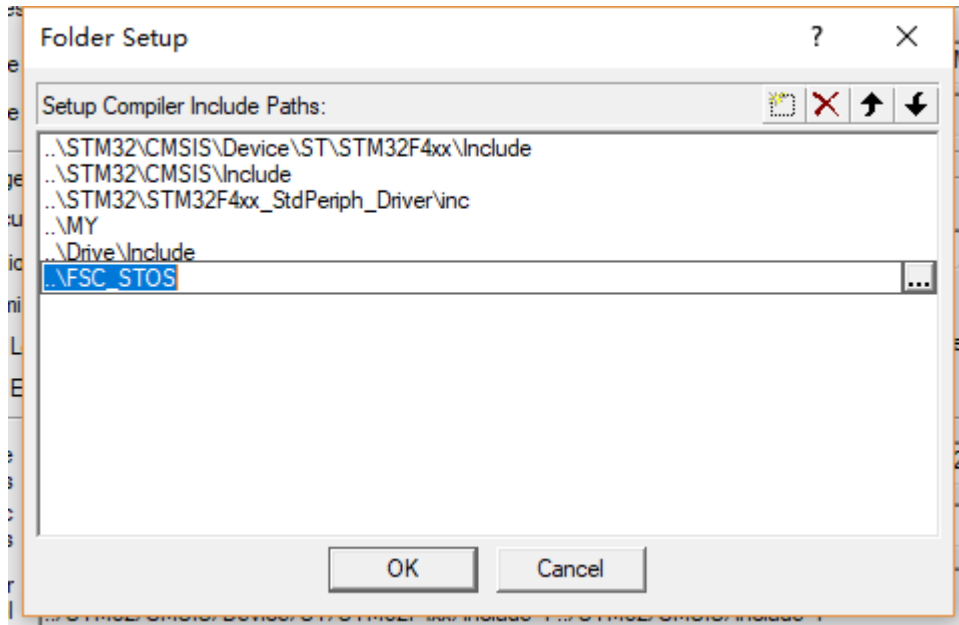






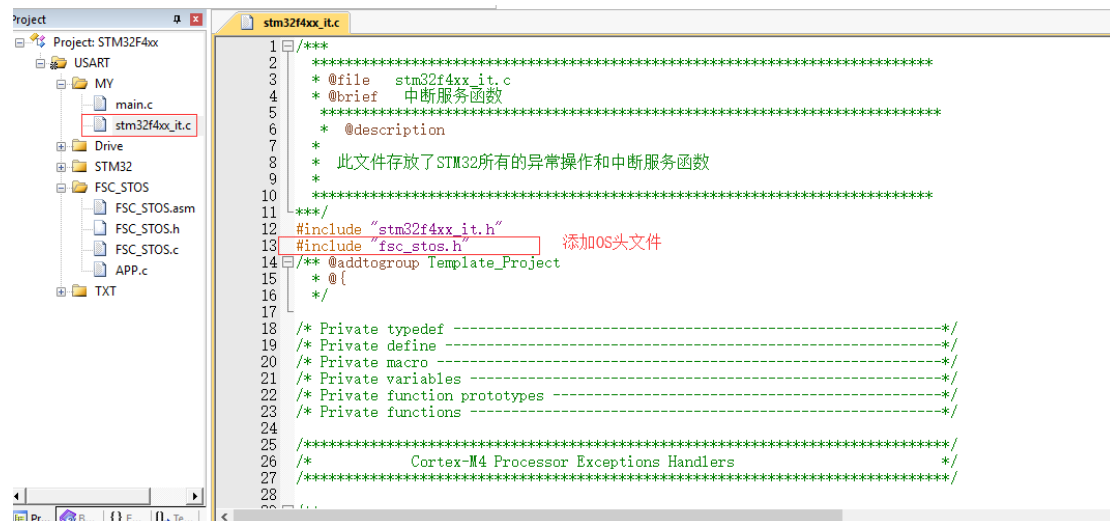
Folder Setup

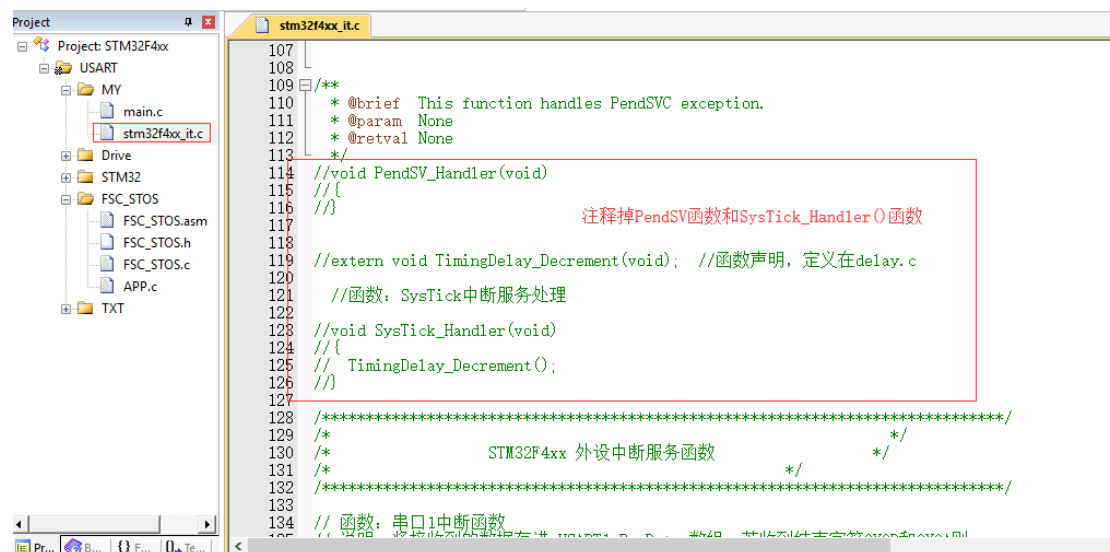




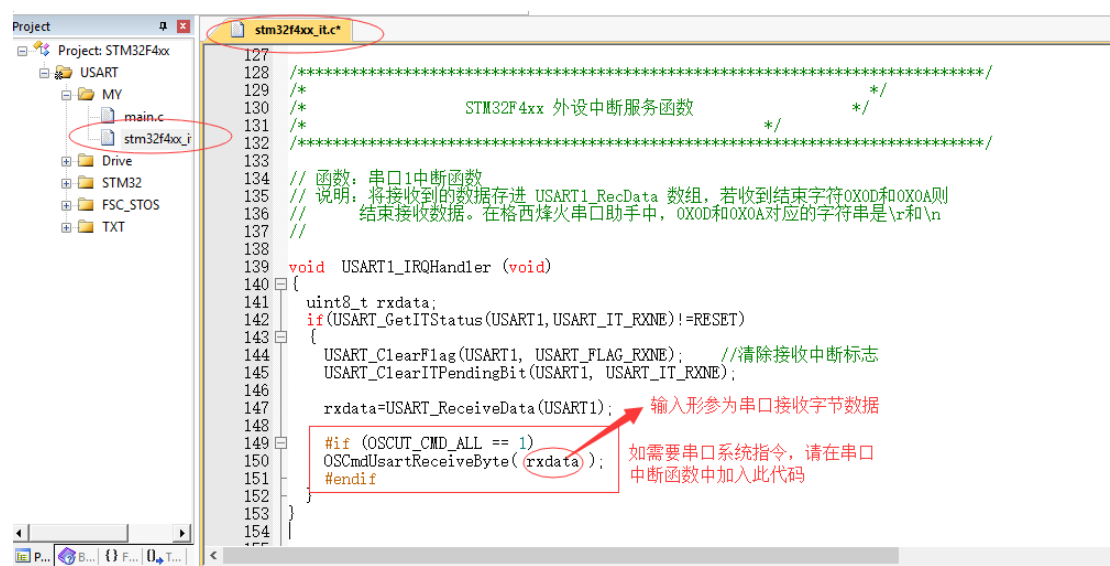
2. 修改 3 个文件:

<1>stm32fxxx_it.c 修改

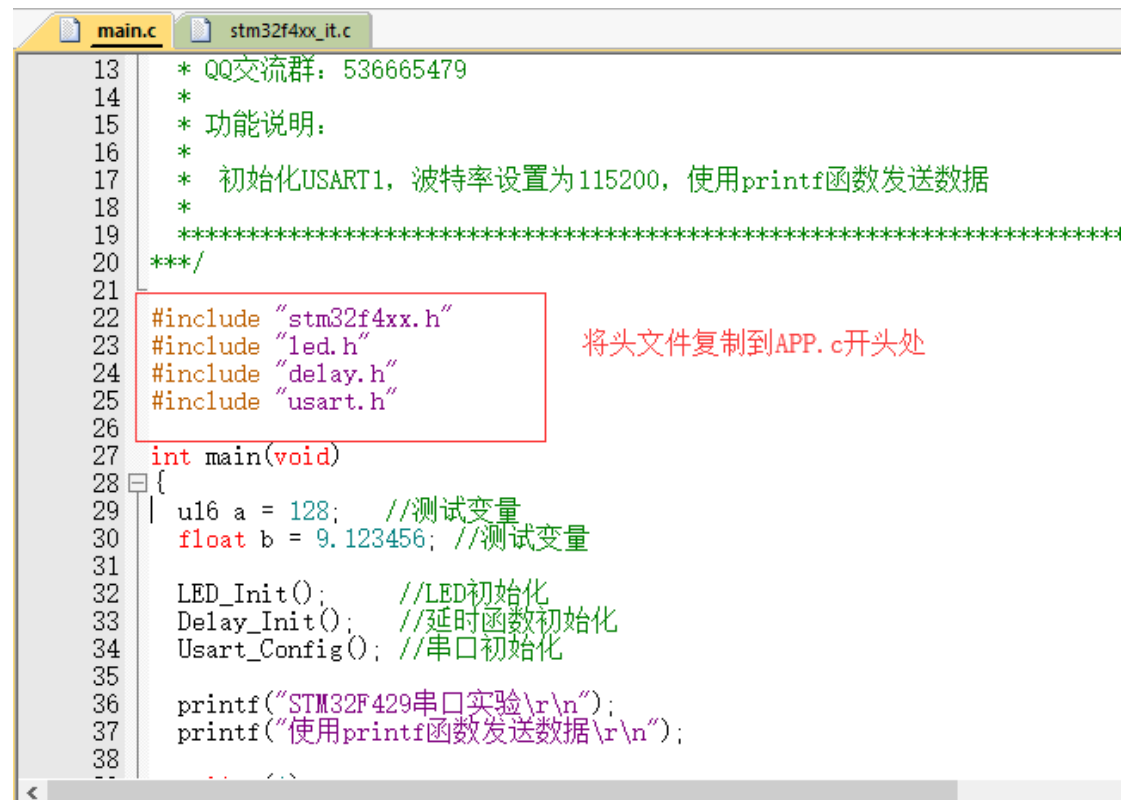




以下指令入口函数可到 **FSC_STOS.h** 中复制。



<2>main.c 修改



```
13  * QQ交流群: 536665479
14  *
15  * 功能说明:
16  *
17  * 初始化USART1, 波特率设置为115200, 使用printf函数发送数据
18  *
19  ****
20  ***/
21
22  #include "stm32f4xx.h"
23  #include "led.h"
24  #include "delay.h"
25  #include "usart.h"
26
27  int main(void)
28  {
29      | ul6 a = 128; //测试变量
30      | float b = 9.123456; //测试变量
31
32      LED_Init(); //LED初始化
33      Delay_Init(); //延时函数初始化
34      Usart_Config(); //串口初始化
35
36      printf("STM32F429串口实验\r\n");
37      printf("使用printf函数发送数据\r\n");
38  }
```

将头文件复制到APP.c开头处


```
APP.c*  main.c
61
62 void Task1(void) //任务1 删除延时有关的代码,
63 { 替换为系统提供的延
64     ul6 a = 128; //测试变量 时函数
65     float b = 9.123456; //测试变量
66
67     LED_Init(); //LED初始化
68     Delay_Init(); //延时函数初始化
69     Usart_Config(); //串口初始化
70
71     printf("STM32F429串口实验\r\n");
72     printf("使用printf函数发送数据\r\n");
73
74     while (1)
75     {
76         LED1_ON; Delay_ms(500); //点亮LED1
77         LED1_OFF; Delay_ms(500); //关闭LED1
78
79         printf("十进制格式: %d\r\n", a);
80         printf("十六进制格式: %x\r\n", a);
81         printf("小数格式: %f\r\n", b);
82     }
83 }
84
85 void Task2(void) //任务2
86 {
87     float a, b, c; //示例代码, 使用时删除
88     a=2.129;
89     b=1.23456;
```

```

APP.c*  main.c
61
62 void Task1(void) //任务1
63 {
64     u16 a = 128; //测试变量
65     float b = 9.123456; //测试变量
66
67     LED_Init(); //LED初始化
68     Usart_Config(); //串口初始化
69
70     printf("STM32F429串口实验\r\n");
71     printf("使用printf函数发送数据\r\n");
72
73     while (1)
74     {
75         LED1_ON; delay_ms(500); //点亮LED1
76         LED1_OFF; delay_ms(500); //关闭LED1
77
78         printf("十进制格式: %d\r\n", a);
79         printf("十六进制格式: %x\r\n", a);
80         printf("小数格式: %f\r\n", b);
81     }
82 }
83
84 void Task2(void) //任务2
85 {
86     float a, b, c; //示例代码, 使用时删除
87     a=2.129;
88     b=4.2346;

```

Project: STM32F4xx

USART

MY

main.c

stm32f4xx_it.c

Drive

led.c

usart.c

delay.c

STM32

for_stnc

APP.c* main.c

```

1 #include "fsc_stos.h"
2 #include "stm32f4xx.h"
3 #include "led.h"
4 #include "delay.h"
5 #include "usart.h"
6
7
8 /******创建任务参数*****/
9 void Task1(void); //任务1声明
10 void Task2(void); //任务2声明
11 void Task3(void); //任务3声明
12 void Task4(void); //任务4声明
13 void Task5(void); //任务5声明

```

从工程中移除delay.c和删除包含头文件

Project: STM32F4xx

USART

MY

main.c

stm32f4xx_it.c

Drive

STM32

FSC_STOS

APP.c

FSC_STOS.asm

FSC_STOS.c

FSC_STOS.h

TXT

APP.c

```

21 __align(8) OS_STK Task2_Stk[Task2_StkSize]; //任务2堆栈
22 __align(8) OS_STK Task3_Stk[Task3_StkSize]; //任务3堆栈
23 __align(8) OS_STK Task4_Stk[Task4_StkSize]; //任务4堆栈
24 __align(8) OS_STK Task5_Stk[Task5_StkSize]; //任务5堆栈
25 /******/
26 void OS_MAIN(void)
27 {
28     /*-----全局初始化区-----*/
29     /*推荐把所有任务都使用到的初始化放在此处, Task独立用到的初始化放在Task内*/
30     Usart_Config(); //串口初始化:9600
31     /*-----*/
32     /******/
33     OSInit(); //系统初始化
34     /******在系统中创建任务*****/
35     OSTaskCreate("Task1", Task1, Task1_Stk, Task1_StkSize, TASK_RUNNING, 1); //OS创建任务1
36     OSTaskCreate("Task2", Task2, Task2_Stk, Task2_StkSize, TASK_RUNNING, 2); //OS创建任务2
37     OSTaskCreate("Task3", Task3, Task3_Stk, Task3_StkSize, TASK_RUNNING, 3); //OS创建任务3
38     OSTaskCreate("Task4", Task4, Task4_Stk, Task4_StkSize, TASK_RUNNING, 4); //OS创建任务4
39     OSTaskCreate("Task5", Task5, Task5_Stk, Task5_StkSize, TASK_RUNNING, 5); //OS创建任务5
40     OSStart(); //OS开始运行
41 }
42
43
44
45
46
47
48

```

将所有任务都有可能用到的初始化放到此处, 串口是所有任务都可能用到的模块

```
stm32f4xx_it.c  main.c  APP.c
18 #define Task5_StkSize 128 //任务5堆栈大小
19
20 __align(8) OS_STK Task1_Stk[Task1_StkSize]; //任务1堆栈
21 __align(8) OS_STK Task2_Stk[Task2_StkSize]; //任务2堆栈
22 __align(8) OS_STK Task3_Stk[Task3_StkSize]; //任务3堆栈
23 __align(8) OS_STK Task4_Stk[Task4_StkSize]; //任务4堆栈
24 __align(8) OS_STK Task5_Stk[Task5_StkSize]; //任务5堆栈
25 /*****
26 void OS_MAIN(void) 将此函数复制到main函数中
27 {
28     /*-----全局初始化区-----*/
29     /*推荐把所有任务都使用到的初始化放在此处，Task独立用到的初始化放在Task内*/
30
31     Usart_Config(); //串口初始化:9600
32
33
34     /*-----*/
35     /*****
36     OSInit(); //系统初始化
37     /*****在系统中创建任务*****
38     OSTaskCreate("Task1", Task1, Task1_Stk, Task1_StkSize, TASK_RUNNING, 1); //OS创建任务1
39     OSTaskCreate("Task2", Task2, Task2_Stk, Task2_StkSize, TASK_RUNNING, 2); //OS创建任务2
40     OSTaskCreate("Task3", Task3, Task3_Stk, Task3_StkSize, TASK_RUNNING, 3); //OS创建任务3
41     OSTaskCreate("Task4", Task4, Task4_Stk, Task4_StkSize, TASK_RUNNING, 4); //OS创建任务4
42     OSTaskCreate("Task5", Task5, Task5_Stk, Task5_StkSize, TASK_RUNNING, 5); //OS创建任务5
43     /*****
44     OSStart(); //OS开始运行
45 }
```

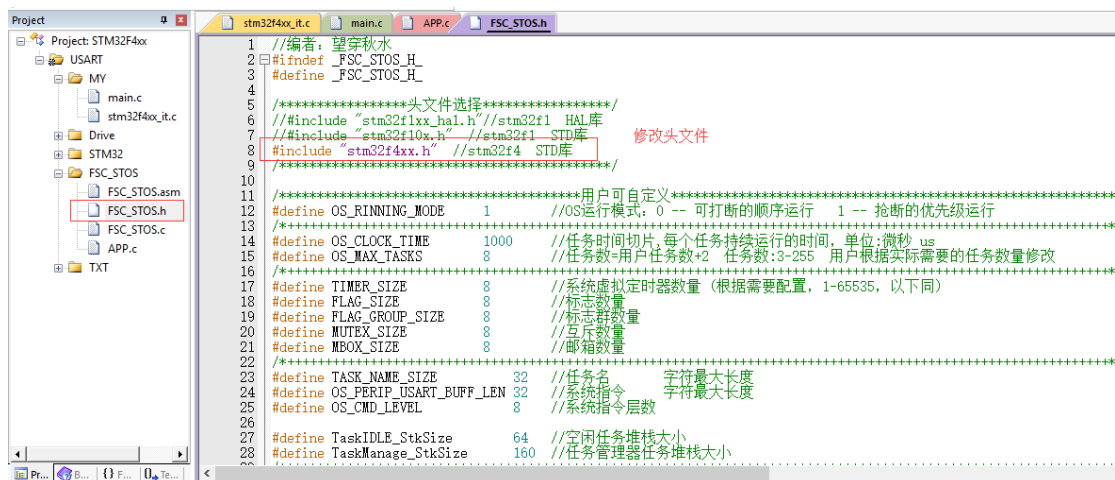
Project

Project: STM32F4xx

- USART
- MY
 - main.c
 - stm32f4xx_it.c
- Drive
- STM32
- FSC_STOS
 - FSC_STOS.asm
 - FSC_STOS.h
 - FSC_STOS.c
- APP.c
- TXT

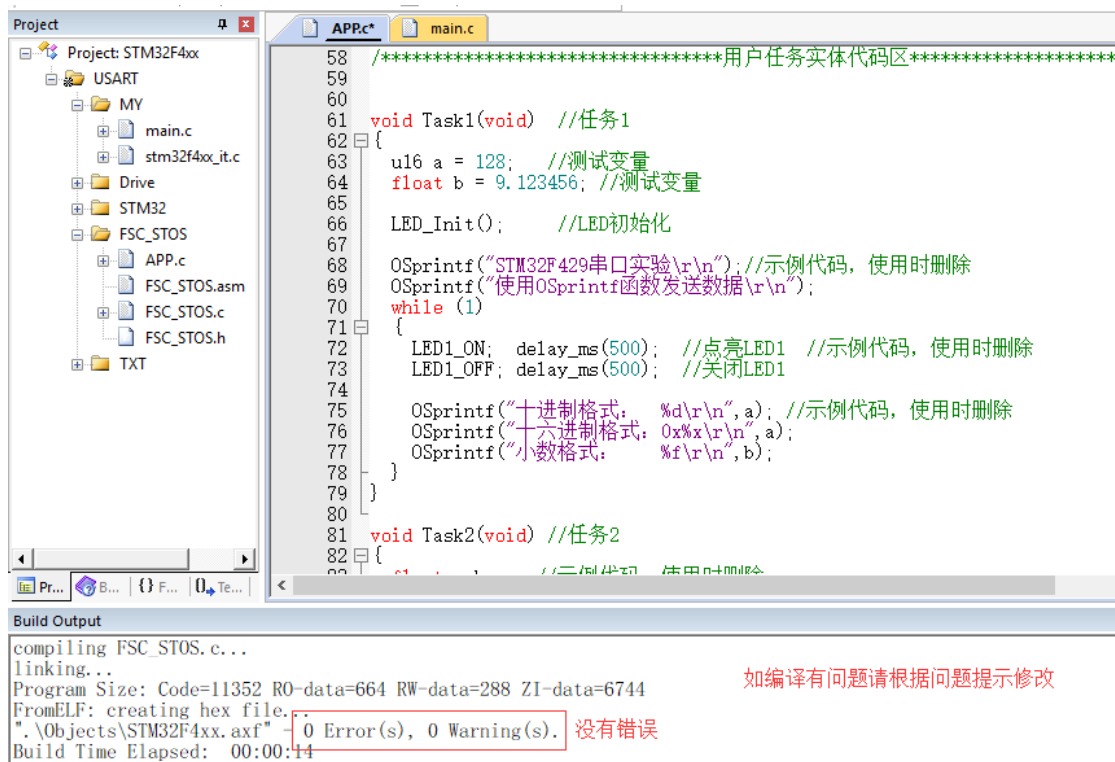
```
stm32f4xx_it.c  main.c  APP.c
1 #include "fsc_stos.h" 包含头文件
2
3 int main(void)
4 {
5     OS_MAIN(); //系统入口函数 删除main函数全
6 } 部内容，只存放
7 //请到APP.c写代码 OS_MAIN() 一个
8 函数
9
10
11
12
```

<3>FSC_STOS.h 修改



```
1 //编者: 望穿秋水
2 #ifndef _FSC_STOS_H_
3 #define _FSC_STOS_H_
4
5 /*****头文件选择*****/
6 #include "stm32f1xx_hal.h" //stm32f1 HAL库
7 #include "stm32f10x.h" //stm32f1 STD库
8 #include "stm32f4xx.h" //stm32f4 STD库
9 /*****
10
11 /*****用户可自定义*****/
12 #define OS_RUNNING_MODE 1 //OS运行模式: 0 -- 可打断的顺序运行 1 -- 抢断的优先级运行
13 /*****
14 #define OS_CLOCK_TIME 1000 //任务时间切片,每个任务持续运行的时间,单位:微秒 us
15 #define OS_MAX_TASKS 8 //任务数=用户任务数+2 任务数:3-255 用户根据实际需要的任务数量修改
16 /*****
17 #define TIMER_SIZE 8 //系统虚拟定时器数量 (根据需要配置, 1-65535, 以下同)
18 #define FLAG_SIZE 8 //标志数量
19 #define FLAG_GROUP_SIZE 8 //标志群数量
20 #define MUTEX_SIZE 8 //互斥数量
21 #define MBOX_SIZE 8 //邮箱数量
22 /*****
23 #define TASK_NAME_SIZE 32 //任务名 字符最大长度
24 #define OS_PERIP_USART_BUFF_LEN 32 //系统指令 字符最大长度
25 #define OS_CMD_LEVEL 8 //系统指令层数
26
27 #define TaskIDLE_StkSize 64 //空闲任务堆栈大小
28 #define TaskManage_StkSize 160 //任务管理器任务堆栈大小
29
```

编译:



```
58 /*****用户任务实体代码区*****/
59
60
61 void Task1(void) //任务1
62 {
63     u16 a = 128; //测试变量
64     float b = 9.123456; //测试变量
65
66     LED_Init(); //LED初始化
67
68     OSprintf("STM32F429串口实验\r\n"); //示例代码,使用时删除
69     OSprintf("使用OSprintf函数发送数据\r\n");
70     while (1)
71     {
72         LED1_ON; delay_ms(500); //点亮LED1 //示例代码,使用时删除
73         LED1_OFF; delay_ms(500); //关闭LED1
74
75         OSprintf("十进制格式: %d\r\n", a); //示例代码,使用时删除
76         OSprintf("十六进制格式: 0x%x\r\n", a);
77         OSprintf("小数格式: %f\r\n", b);
78     }
79
80
81 void Task2(void) //任务2
82 {
83     //示例代码,使用时删除
84
```

Build Output

```
compiling FSC_STOS.c...
linking...
Program Size: Code=11352 RO-data=664 RW-data=288 ZI-data=6744
FromELF: creating hex file.
".\Objects\STM32F4xx.axf" - 0 Error(s), 0 Warning(s). 没有错误
Build Time Elapsed: 00:00:14
```

如编译有问题请根据问题提示修改

ATK
XCOM V2.0

十六进制格式: 0x80
小数格式: 9.123456
*****系统状态信息*****

OS Running Mode: Order
TaskTimeSliceCnt: 4288382
CPU占用率: 7.6% CPU最大占用率: 38.5%

利用率 CPU	使用栈 Used	空闲栈 Free	百分比 per	优先级 prio	任务名 Taskname
92.4%	50	14	78.1%	0	Task_Idle
0.1%	113	47	70.6%	1	Task_Manage
6.4%	93	35	72.7%	1	Task1
0.9%	95	33	74.2%	2	Task2
0.0%	52	76	40.6%	3	Task3
0.0%	52	76	40.6%	4	Task4
0.0%	52	76	40.6%	5	Task5

c= 6.363600

十进制格式: 128
十六进制格式: 0x80
小数格式: 9.123456

串口选择
COM4: USB-SERIAL
波特率 9600
停止位 1
数据位 8
奇偶校验 无
串口操作 关闭串口
保存窗口 清除接收
☐ 16进制显示 ☐ 白底黑字
☐ RTS ☐ DTR
☐ 时间戳(以换行回车断帧)

单条发送 多条发送 协议传输 帮助

cmd/osmanage//

发送

清除发送