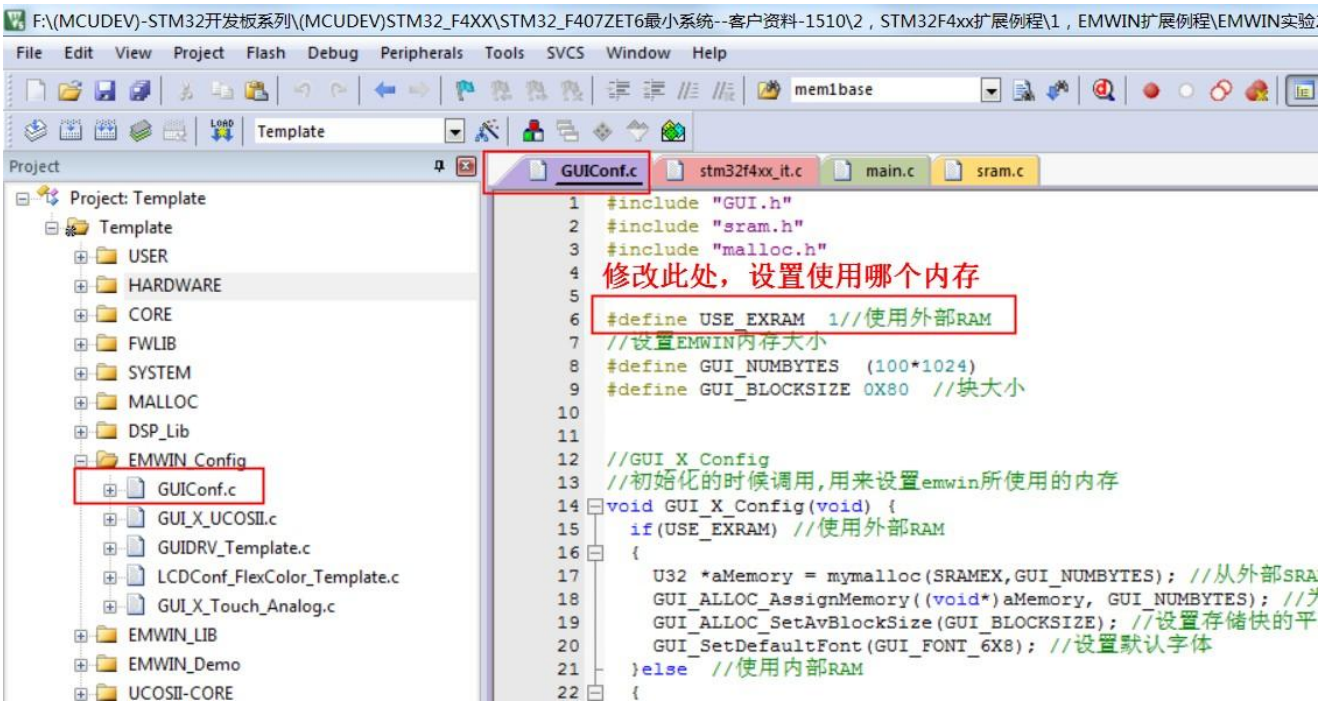
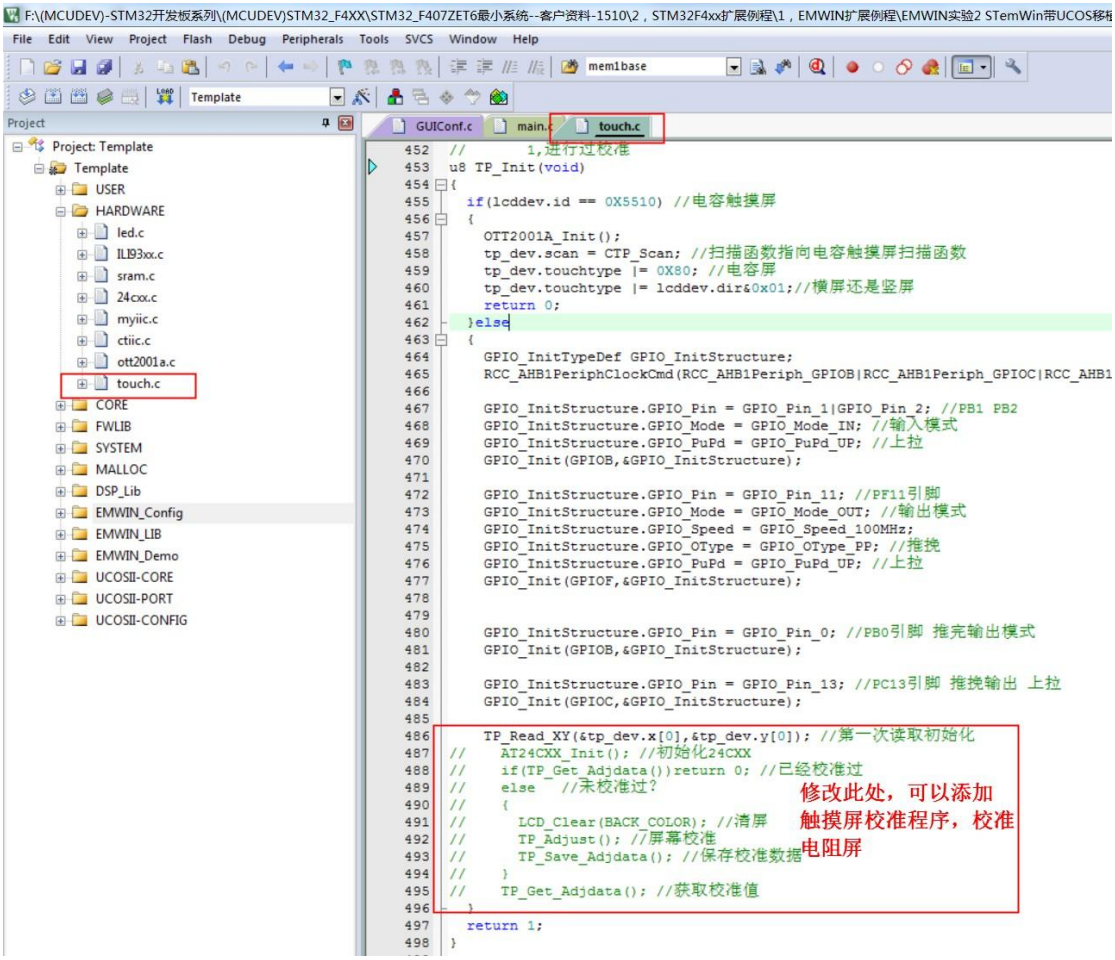
1、使用 STM32F103ZET6 的时候，例程全部使用了外部 SRAM 进行测试，如果板子没有外部 SRAM，那在设置的时候，改成使用内部 RAM 的选项。

修改位置如下：



2、 程序全部去掉触摸屏校准程序，如果需要触摸屏，需要自己将注释掉的校准程序部分添加上去，并且存储校准数据



3、如果程序运行出现异常，请修改内存操作的地址和数据建立的时间周期，因为每个批次的内存，可能通信时间兼

容性不一致。

FSMC_ReadWriteTimingStructure.FSMC_AddressSetupTime = 0x0F; //地址建立时间

FSMC_ReadWriteTimingStructure.FSMC_AddressHoldTime = 0x00; //地址保持时间，A 模式不需要

FSMC_ReadWriteTimingStructure.FSMC_DataSetupTime = 0x18; //数据保持时间

```
//GPIOG引脚复用
GPIO_PinAFConfig(GPIOG,GPIO_PinSource12,GPIO_AF_FSMC);  //PE15 AF12

//FSMC读时序控制寄存器
FSMC_ReadWriteTimingStructure.FSMC_AddressSetupTime = 0X0F; //地址建立时间 16个HCLK =16X(1/168M)=96ns
FSMC_ReadWriteTimingStructure.FSMC_AddressHoldTime = 0x00;  //地址保持时间模式A未用到
FSMC_ReadWriteTimingStructure.FSMC_DataSetupTime = 0x18;    //数据保存时间为25个HCLK 25x(1/168M)=150ns
FSMC_ReadWriteTimingStructure.FSMC_BusTurnAroundDuration = 0x00;
FSMC_ReadWriteTimingStructure.FSMC_CLKDivision = 0x00;
FSMC_ReadWriteTimingStructure.FSMC_DataLatency = 0x00;
FSMC_ReadWriteTimingStructure.FSMC_AccessMode = FSMC_AccessMode_A; //模式A

//FSMC写时序控制寄存器
FSMC_WriteTimingStructure.FSMC_AddressSetupTime = 0x08;  //地址建立时间为8个HCLK 8x(1/168M)=48ns
FSMC_WriteTimingStructure.FSMC_AddressHoldTime = 0x00;   //地址保持时间在模式A未用到
FSMC_WriteTimingStructure.FSMC_DataSetupTime = 0x08;     //应为某些液晶驱动芯片的原因,所以数据保持时间至少为9个HCLK 为9x6=54ns
FSMC_WriteTimingStructure.FSMC_BusTurnAroundDuration = 0x00;
FSMC_WriteTimingStructure.FSMC_CLKDivision = 0x00;
FSMC_WriteTimingStructure.FSMC_DataLatency = 0x00;
FSMC_WriteTimingStructure.FSMC_AccessMode = FSMC_AccessMode_A; //模式A

FSMC_NORSRAMInitStructure.FSMC_Bank = FSMC_Bank1_NORSRAM4;  //NOR/SRAM的Bank4                使用液晶，要初始化成不是存储设备
FSMC_NORSRAMInitStructure.FSMC_DataAddressMux = FSMC_DataAddressMux_Disable; //不复用数据线
FSMC_NORSRAMInitStructure.FSMC_MemoryType = FSMC_MemoryType_NOR;   //接液晶屏，不初始化成SRAM模式
FSMC_NORSRAMInitStructure.FSMC_MemoryDataWidth = FSMC_MemoryDataWidth_16b;  //16位数据宽度
FSMC_NORSRAMInitStructure.FSMC_BurstAccessMode = FSMC_BurstAccessMode_Disable; //是否使能突发访问,仅对同步突发存储器有效,此处未用到
FSMC_NORSRAMInitStructure.FSMC_AsynchronousWait = FSMC_AsynchronousWait_Disable;//是否使能同步传输模式下的等待信号,此处未用到
FSMC_NORSRAMInitStructure.FSMC_WaitSignalPolarity = FSMC_WaitSignalPolarity_Low; //等待信号的极性,仅在突发模式访问下有用
FSMC_NORSRAMInitStructure.FSMC_WrapMode = FSMC_WrapMode_Disable;  //是否使能环路突发模式,此处未用到
FSMC_NORSRAMInitStructure.FSMC_WaitSignalActive = FSMC_WaitSignalActive_BeforeWaitState; //存储器是在等待周期之前的一个时钟周期还是等待
FSMC_NORSRAMInitStructure.FSMC_WriteOperation = FSMC_WriteOperation_Enable; //存储器写使能
FSMC_NORSRAMInitStructure.FSMC_WaitSignal = FSMC_WaitSignal_Disable;   //等待使能位,此处未用到
FSMC_NORSRAMInitStructure.FSMC_ExtendedMode = FSMC_ExtendedMode_Enable; //读写使用不同的时序
FSMC_NORSRAMInitStructure.FSMC_WriteBurst = FSMC_WriteBurst_Disable;  //异步传输期间的等待信号
FSMC_NORSRAMInitStructure.FSMC_ReadWriteTimingStruct = &FSMC_ReadWriteTimingStructure;
FSMC_NORSRAMInitStructure.FSMC_WriteTimingStruct = &FSMC_WriteTimingStructure; //写时序
```

4、 使用液晶屏的时候，液晶屏也适用总线通信， 也可能需要调整以上时间。看初始化过程，初始化内存一般是最后，所以，先设置的会被后设置的改掉。

程序先初始化液晶，所以，FMSC 的总线参数是以后初始化的内存参数为准的。

sram.c | GUIConf.c | main.c | touch.c

```
#define LED0_STK_SIZE        64
//任务堆栈
OS_STK LED0_TASK_STK[LED0_STK_SIZE];
//led0任务
void led0_task(void *pdata);


//EMWINDEMO任务
//设置任务优先级
#define EMWINDEMO_TASK_PRIO      4
//任务堆栈大小
#define EMWINDEMO_STK_SIZE       2048
//任务堆栈
OS_STK EMWINDEMO_TASK_STK[EMWINDEMO_STK_SIZE];
//emwindemo_task任务
void emwindemo_task(void *pdata);


int main(void)
{
   delay_init(168);          //延时初始化
   NVIC_PriorityGroupConfig(NVIC_PriorityGroup_2);    //中断分组配置
   uart_init(115200);        //串口波特率设置
   TFTLCD_Init();          //初始化LCD
   TP_Init();             //初始化触摸屏
   LED_Init();            //LED初始化
   FSMC_SRAM_Init();       //SRAM初始化
   mem_init(SRAMIN);       //内部RAM初始化
   mem_init(SRAMEX);       //外部RAM初始化
   mem_init(SRAMCCM);      //CCM初始化

   OSInit();            //初始化UCOS
   OSTaskCreate(start_task,                    //start_task任务
         (void*)0,                      //参数
         (OS_STK*)&START_TASK_STK[START_STK_SIZE-1], //任务堆栈栈顶
         START_TASK_PRIO);                  //任务优先级
   OSStart();  //开启UCOS
}
```