STM32CubeMX + HAL

前言

我的CSDN博客： **小锋学长生活大爆炸**

紧急避坑

USART

问题1： 打印正常，但是加入接收中断后，开始出bug，最后锁定接收中断挂掉了。
原因： HAL库的串口接收发送函数有bug，就是收发同时进行的时候，会出现锁死的现象。
解决： 需要注释掉 HAL_UART_Receive_IT 和 HAL_UART_Transmit_IT 中的 **__HAL_LOCK(huart)** 函数。或者不要在接收里面，每接收到一个字符就printf一下。

问题2： 在接收中断中使用HAL_UART_Receive_IT（）函数，会导致CR1的RXNEIE 置0，最后一直处于错误状态，无法进行接收。
解决： 注释掉 HAL_UART_Receive_IT 中的 HAL_LOCK(huart) 函数

freertos+fatfs+sdio

问题：没有加freertos时候，sd卡读写正常；加上freertos时候，mout成功，但read等其他操作返回错误3 not ready

解决： sdio和sddma的中断优先级要小于freertos的最小优先级

一些说明

使用`STM32CubeMX`代码生成工具，不用关注底层配置的细节，真舒服。

使用教程：

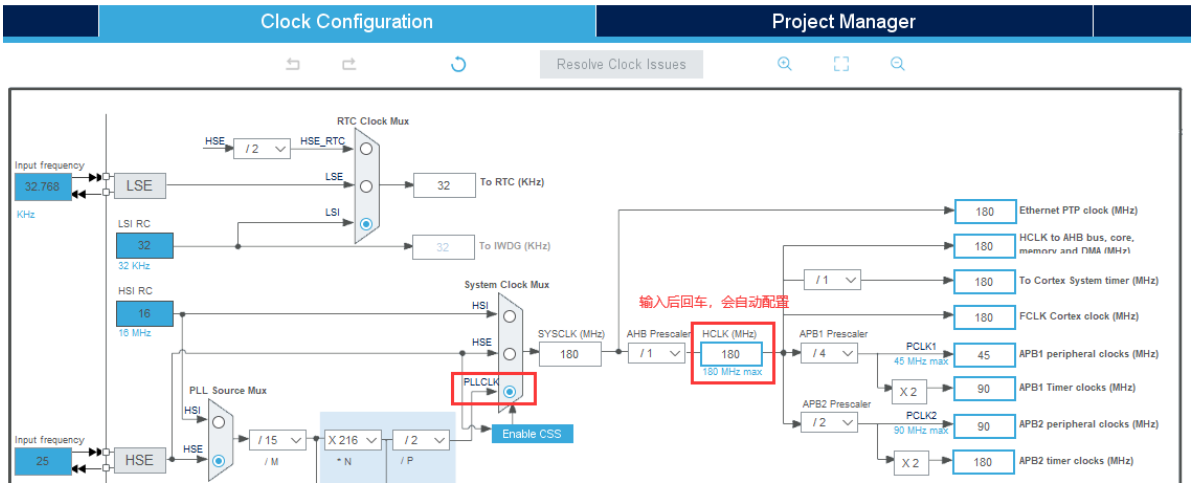
https://sxf1024.lanzoui.com/b09rf2dwj **密码:bgvi**

虽然`Cube+HAL`很舒服，但新手不建议用。最好还是先去学一下标准库怎么用，有个大致概念后，再来学这一套。

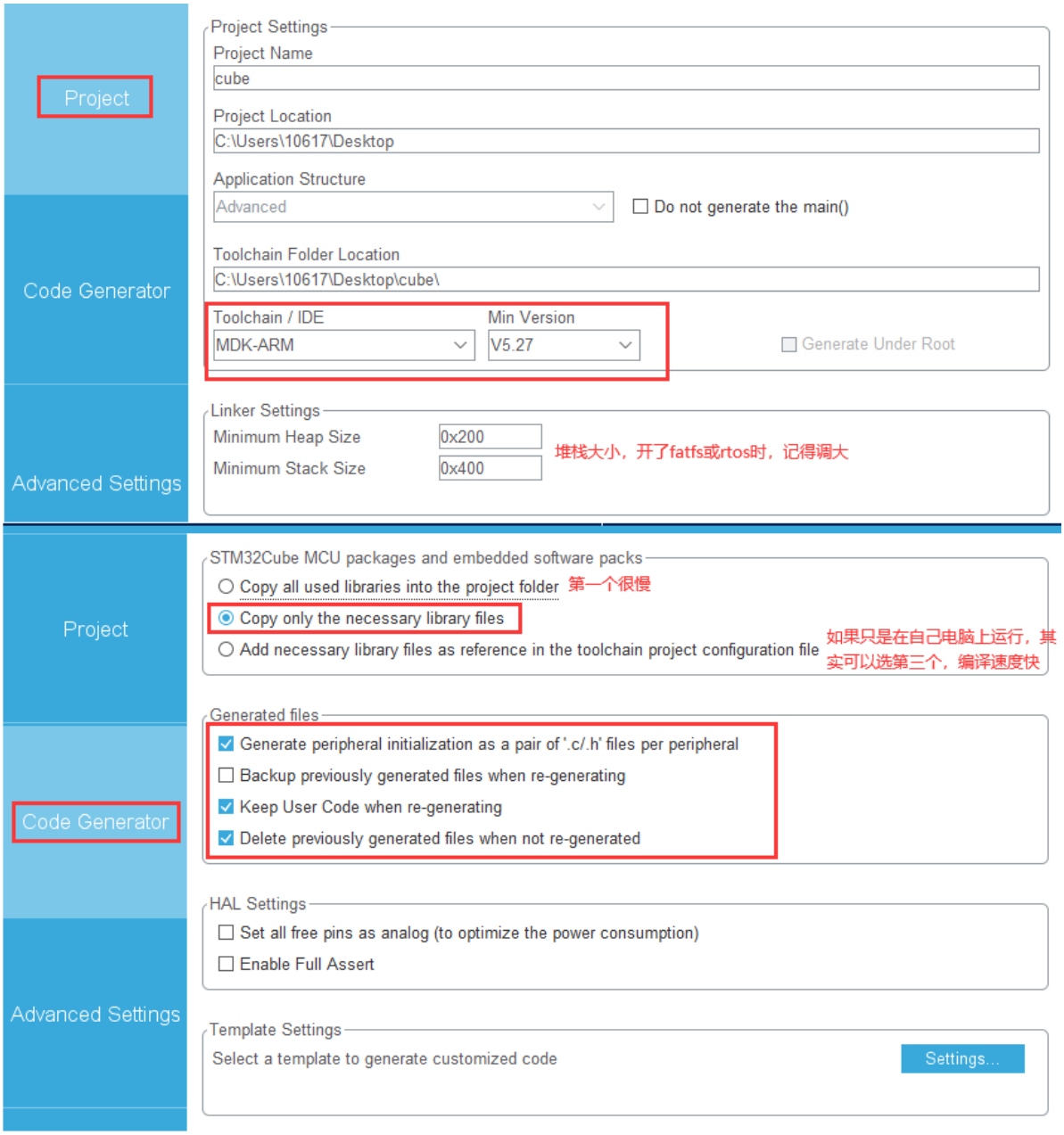
自动化的东西虽好，但一旦出了问题，解决起来也是挺头疼的。

Cube基本使用

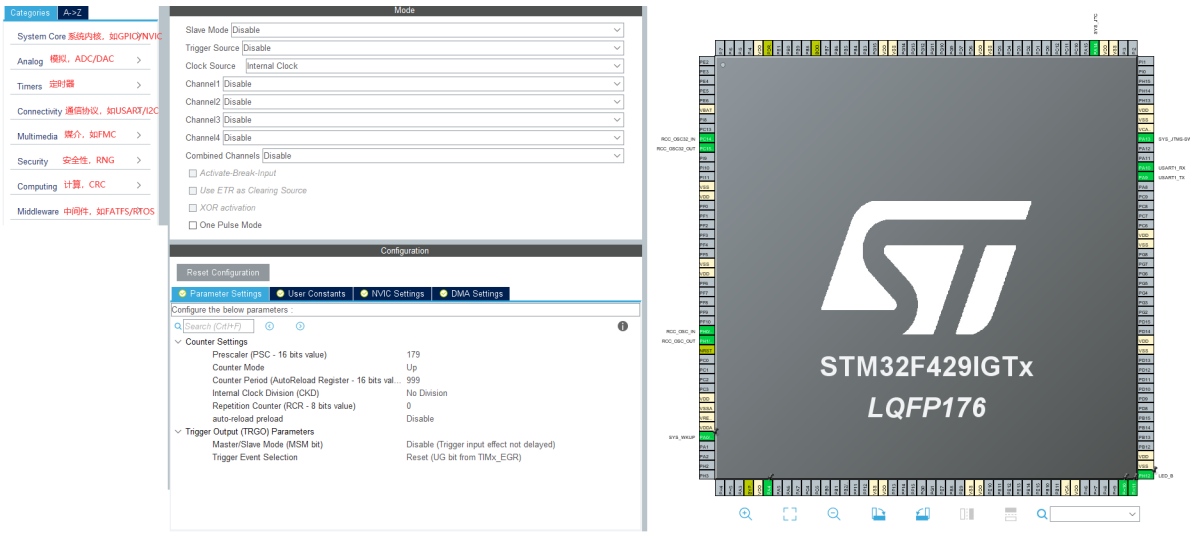
1. 新建工程
2. 选择芯片
3. **Pinout&Configuration**，选择`RCC(HSE：Crystal/Ceramic Resonator)`、`SYS(Debug：Serial Wiire)`
4. **Clock Configuration**，配置时钟树



5. **Project Manager**，配置工程输出项



6. **Pinout&Configuration**，选择功能(若是选`GPIO`相关，可以直接在**Pinout view**选择；若是其他功能，可以在左边**Categories**打开，会自动配置引脚)、设置`Parameter Settings/NVIC`等



7. **GENERATE CODE**，生成工程，用KEIL打开编辑

HAL库函数

- 函数形式：均以`HAL_`开头
- 寻找过程：在驱动文件`stm32f4xx_hal_XXX.c`或其`.h`文件中找函数定义，一般在靠后位置
- 其他说明：
  - `HAL`库并没有把所有的操作都封装成凶数。
  - 对于底层的**寄存器操作**(如读取捕获/比较寄存器)，还有修改外设的某个**配置参数**(如改变输入捕获的极性)，`HAL`库会使用**宏定义**来实现。而且会用`__HAL_`作为这类宏定义的**前缀**。
  - **获取**某个参数，宏定义中一般会有`_GET`；而**设置**某个参数的，宏定义中就会有`_SET`。
  - 在开发过程中，如果遇到**寄存器**级别或者**更小范围**的操作时，可以到该外设的**头文件**中查找，一般都能找到相应的宏定义。
  - `HAL`库函数**第一个参数**一般都是**句柄**(一个包含了当前对象绝大部分状态的结构体)，虽然增加了开销，但是用起来便捷了非常多。

中断回调函数

- 函数形式：`HAL_XXX_XXXCallback()`。
- 寻找过程：中断文件`stm32f4xx_it.c` -> 中断函数`XXX_IRQHandler(void)` -> HAL库中断函数`HAL_XXX_IRQHandler(GPIO_PIN_13)` -> 回调函数`HAL_XXX_XXXCallback()`

外设对应时钟

1. 随便进入一个**外设初始化函数**，如`MX_GPIO_Init()`
2. 随便进入一个**时钟使能函数**，如`__HAL_RCC_GPIOC_CLK_ENABLE()`
3. 随便进入一个**RCC宏定义**，如`RCC_AHB1ENR_GPIOCEN`
4. 或者直接进入`stm32f429xx.h`文件
5. 里面有所有**外设与时钟**对应关系，如`RCC_AHB1ENR_DMA1EN`

配置示例

小编有话说

- 例子源码：

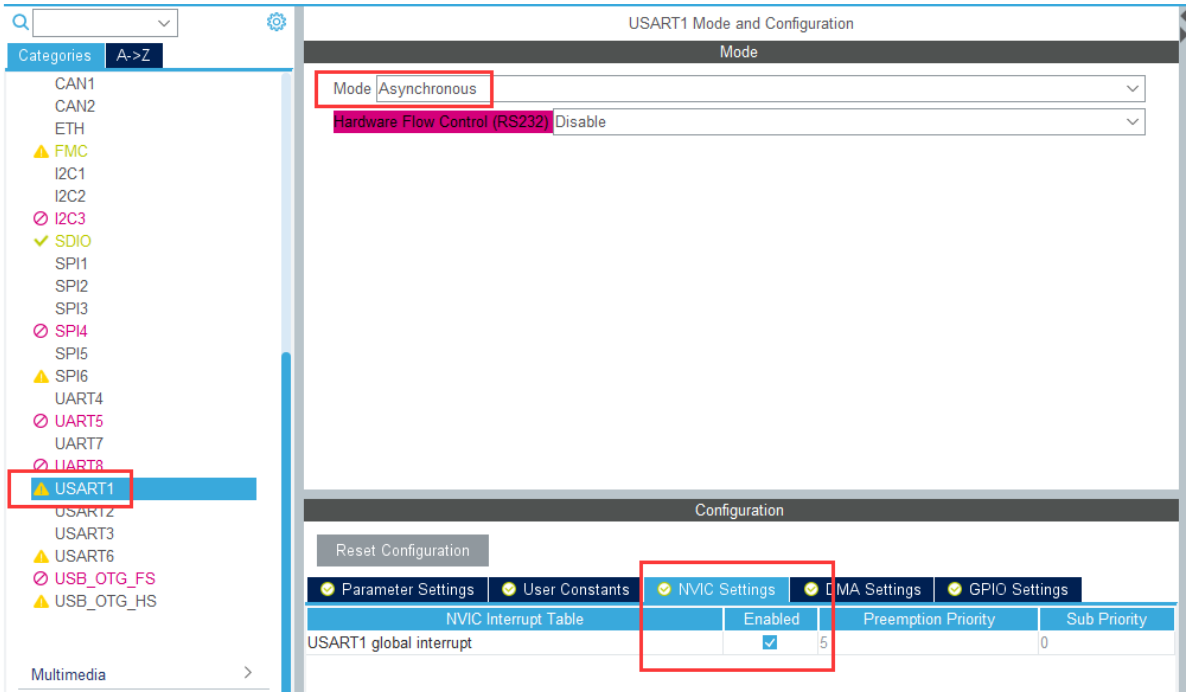
  https://sxf1024.lanzoui.com/b09rf535a 密码:bf5q
- 如果配置过程中，参数不知道怎么设置，可以去标准库例程(如野火、正点原子)中看对应的参数是什么
- Cube软件只是帮你配置了底层，一些初始化代码还是需要自己手动加的，如SDRAM充电初始化、读写函数等
- 以下内容都是基于“**野火F429IGT6挑战者V2开发板**”，其他板子按照原理图改改引脚都能用的

USART

源码链接：

https://sxf1024.lanzoui.com/b09rf535a 密码:bf5q

详细教程网上挺多，配置也简单，只要勾选一下USARTx，再开一下中断就行。



在Keil就比较要注意了。

由于每次接收完，程序内部自动把接收中断关了，所以每次要手动打开。

总的来说，加这几部分：

- `main`函数中，`while`之前：

```
// 使能串口中断接收
HAL_UART_Receive_IT(&huart1, (uint8_t*)&DataTemp_UART1, 1);
```

- 任意位置添加**printf重定向函数**：

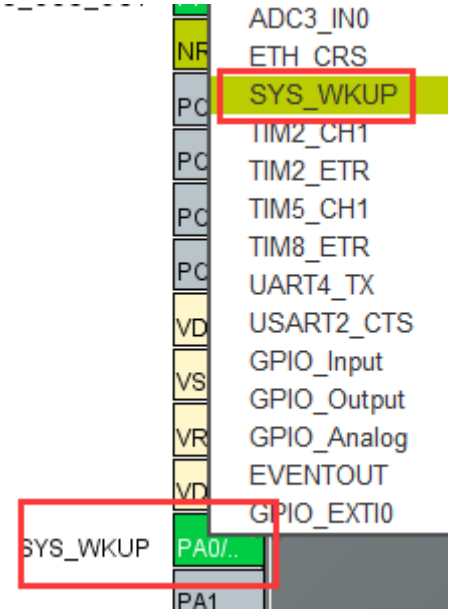
```
#include "stdio.h"
int fputc(int ch, FILE *f){
    HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 0XFF);
    return ch;
}
```

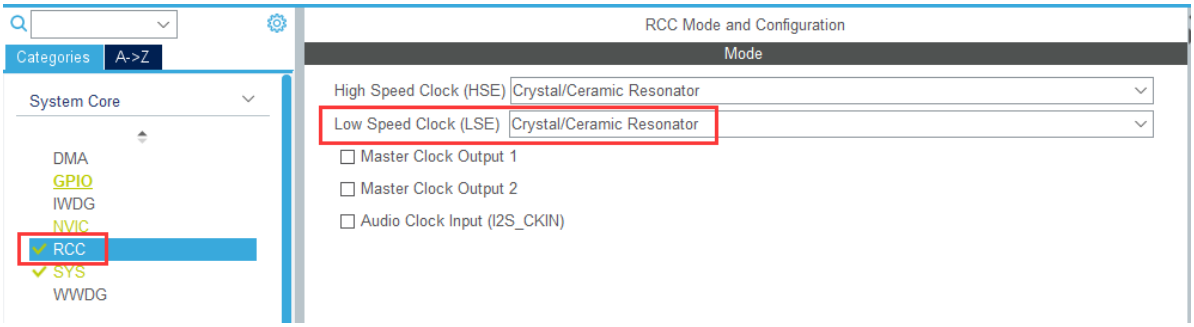
- 任意位置添加**中断回调函数**：

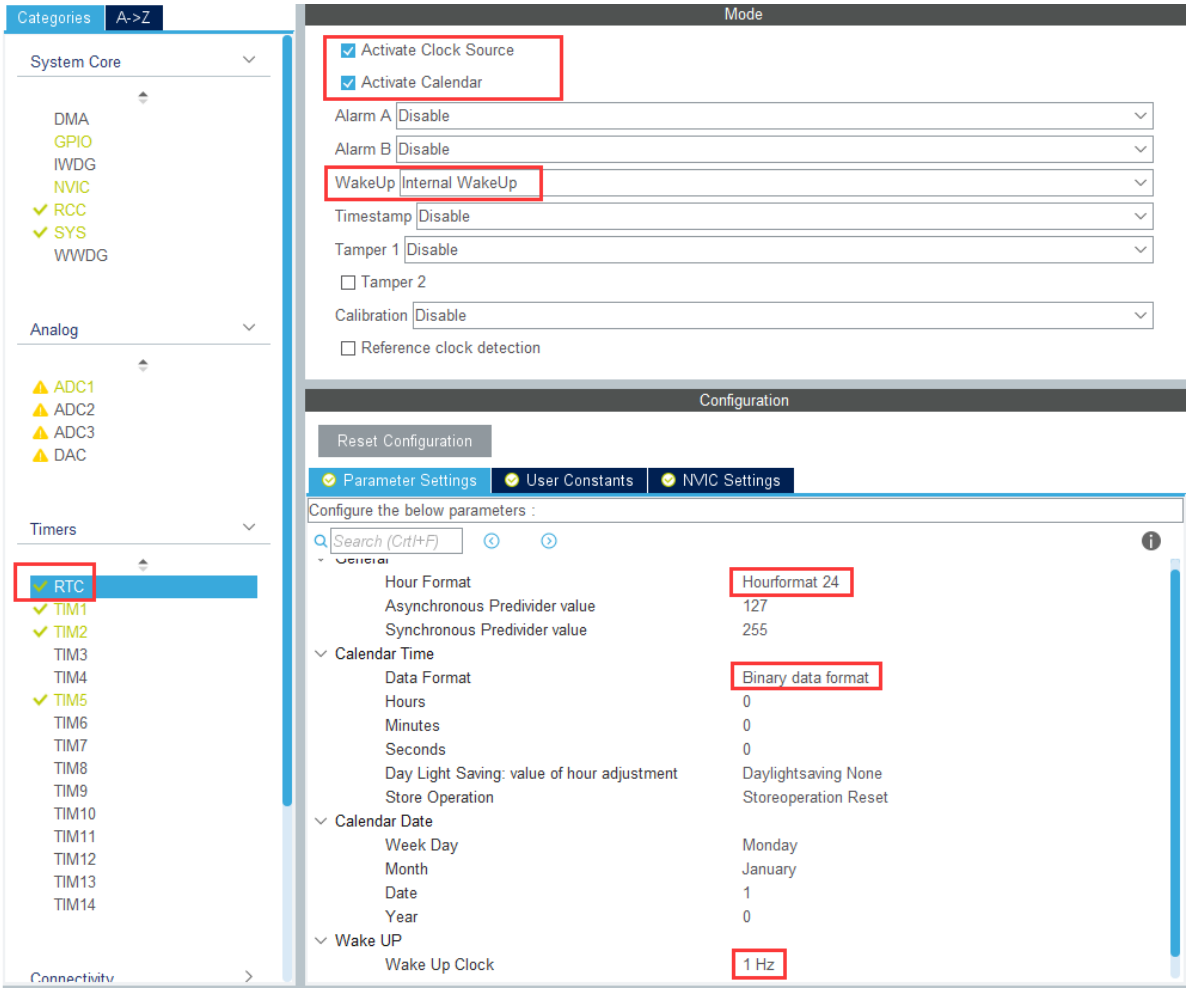
```
 #define UART1BuffLen 200
 extern uint8_t DataBuff_UART1[UART1BuffLen];
 extern uint32_t DataTemp_UART1;
 extern uint16_t DataSTA_UART1;

 uint32_t DataTemp_UART1;
 uint8_t DataBuff_UART1[UART1BuffLen];
 uint16_t DataSTA_UART1;
 void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart)
 {
   if(huart->Instance == USART1){
```

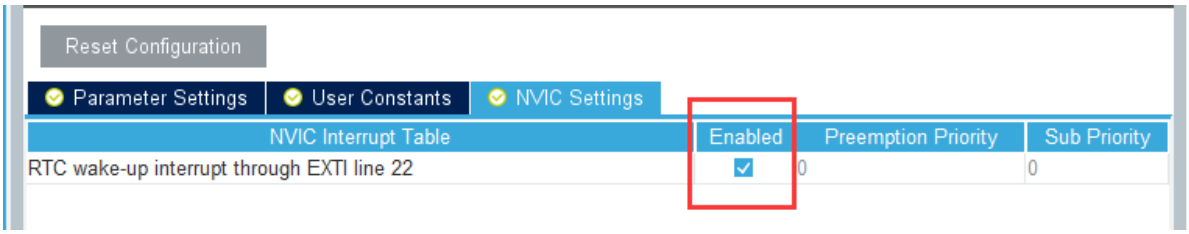
```
        if(DataSTA_UART1 < UART1BuffLen){
            if(DataTemp_UART1 == 0x0A && DataSTA_UART1>0 &&
    DataBuff_UART1[DataSTA_UART1-1]==0X0D){
                printf("USART: %s\r\n", DataBuff_UART1);
                DataSTA_UART1 = 0;
            }
            else{
                if(DataSTA_UART1 == 0){
                    memset(DataBuff_UART1, 0, sizeof(DataBuff_UART1));
                }
                DataBuff_UART1[DataSTA_UART1++] = DataTemp_UART1;
            }
        }
        // 使能串口中断接收
        HAL_UART_Receive_IT(&huart1, (uint8_t*)&DataTemp_UART1, 1);
      }
    }
```

RTC





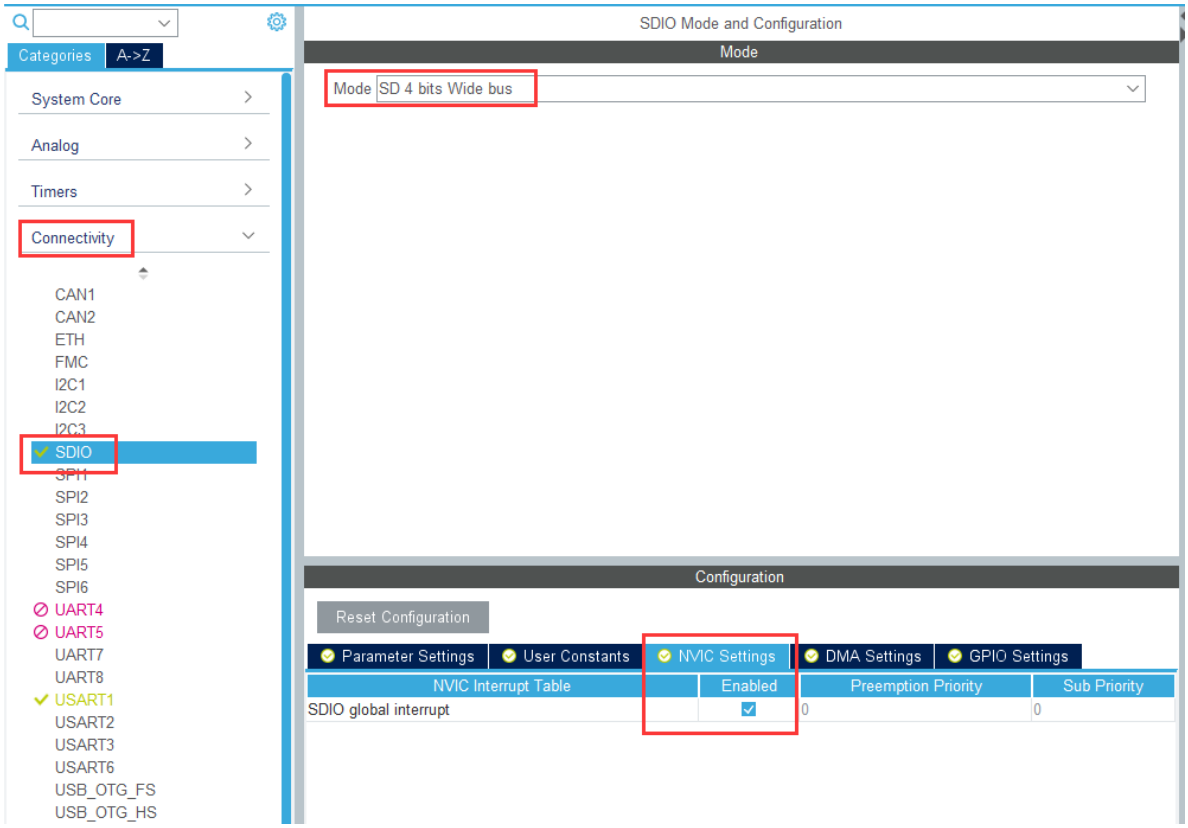




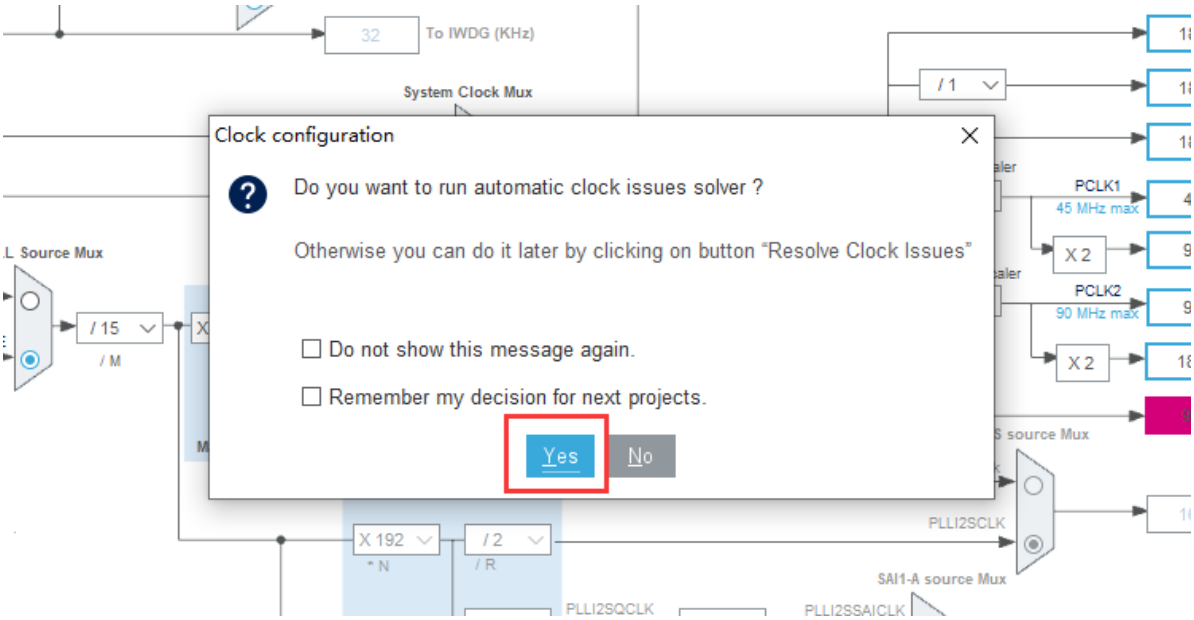

```c
RTC_DateTypeDef sDate;
RTC_TimeTypeDef sTime;
uint8_t second_tmp = 0;


HAL_RTC_GetTime(&hrtc, &sTime, RTC_FORMAT_BIN); // 读取时间
HAL_RTC_GetDate(&hrtc, &sDate, RTC_FORMAT_BIN); // 读取日期
if(second_tmp != sTime.Seconds) { // 读取秒
    second_tmp = sTime.Seconds;
    printf("20%d%d-%d%d-%d%d\r\n",
           sDate.Year/10%10, sDate.Year%10,
           sDate.Month/10%10, sDate.Month%10,
           sDate.Date/10%10, sDate.Date%10);
    printf("%d%d:%d%d:%d%d\r\n",
           sTime.Hours/10%10, sTime.Hours%10,
           sTime.Minutes/10%10, sTime.Minutes%10,
           sTime.Seconds/10%10, sTime.Seconds%10);
}
```

SDIO + FATFS

1. 选择SDIO功能，**Pinout&Clock Configuration**，`Connectivity -> SDIO -> Mode: SD 4bit Wide bus -> 勾选NVIC`



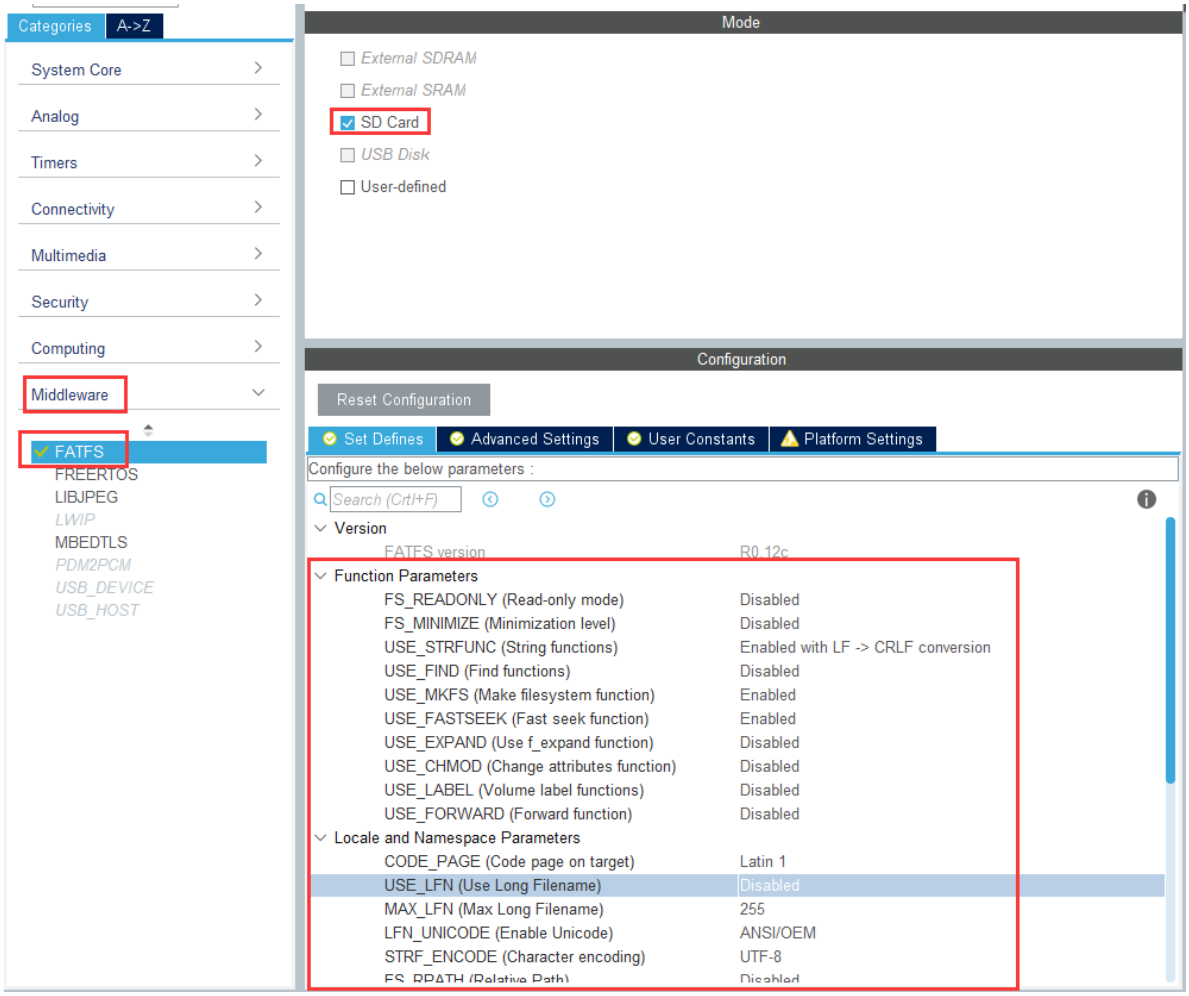
2. 配置SDIO时钟，**Clock Configuration**，SDIO模块输入要求为**48MHz**，系统提示可以自动设置时钟问题，选择`Yes`。SDIO时钟分频系数`CLKDIV`，计算公式为`SDIO_CK=48MHz/(CLKDIV+2)`也可手动修改时钟配置。如果遇到读写问题，可以试着调整到**24MHz**。



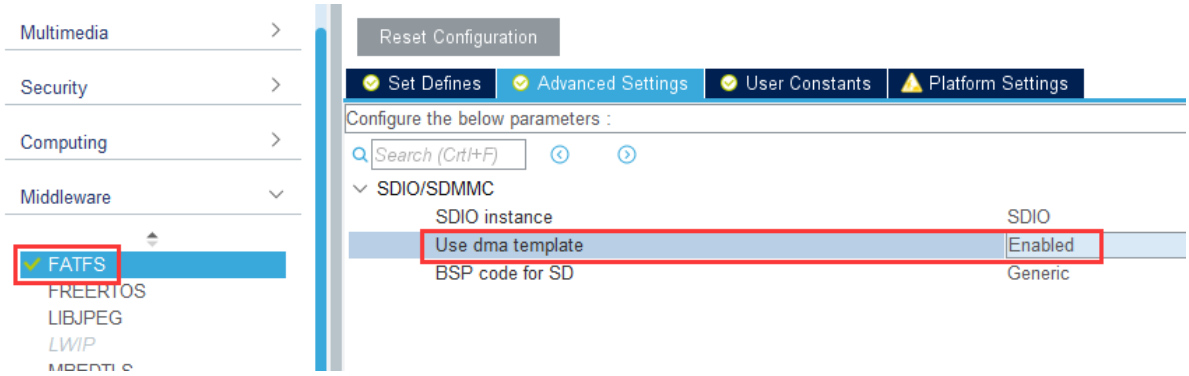
3. **启用文件系统中间件**，**Pinout&Clock Configuration**，`Middleware -> FATFS`，模式选择SD卡，配置文件系统：

   如果要支持中文文件名，则配置`CODE_PAGE`为`Simplified Chinese`

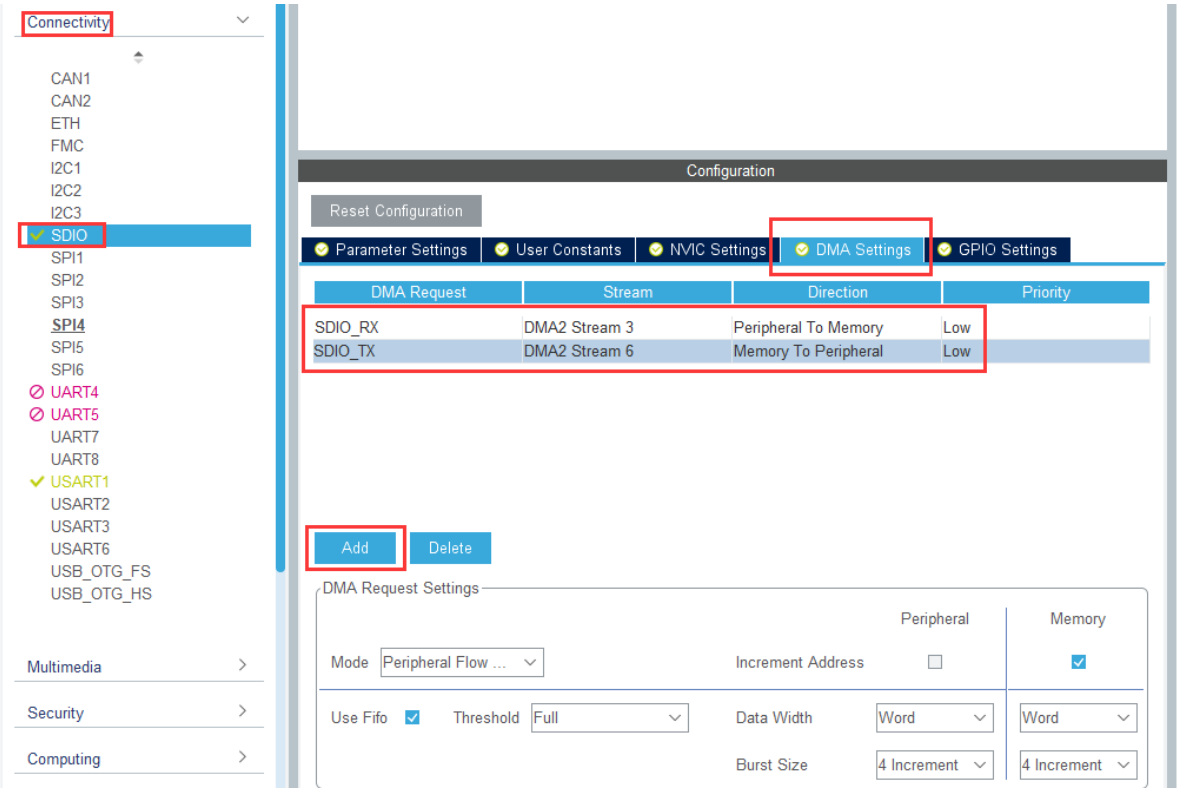
   如果要支持长文件名，则要使能`USE_LEN`



4. 继续上面界面， **Advanced Settings**勾选 `Use dma template`



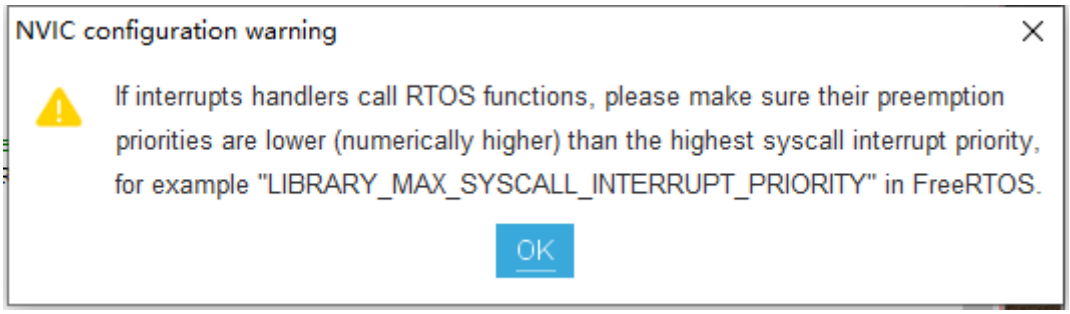
5. 设置DMA传输， **Pinout&Clock Configuration**， `Connectivity -> SDIO-> DMA Settings`

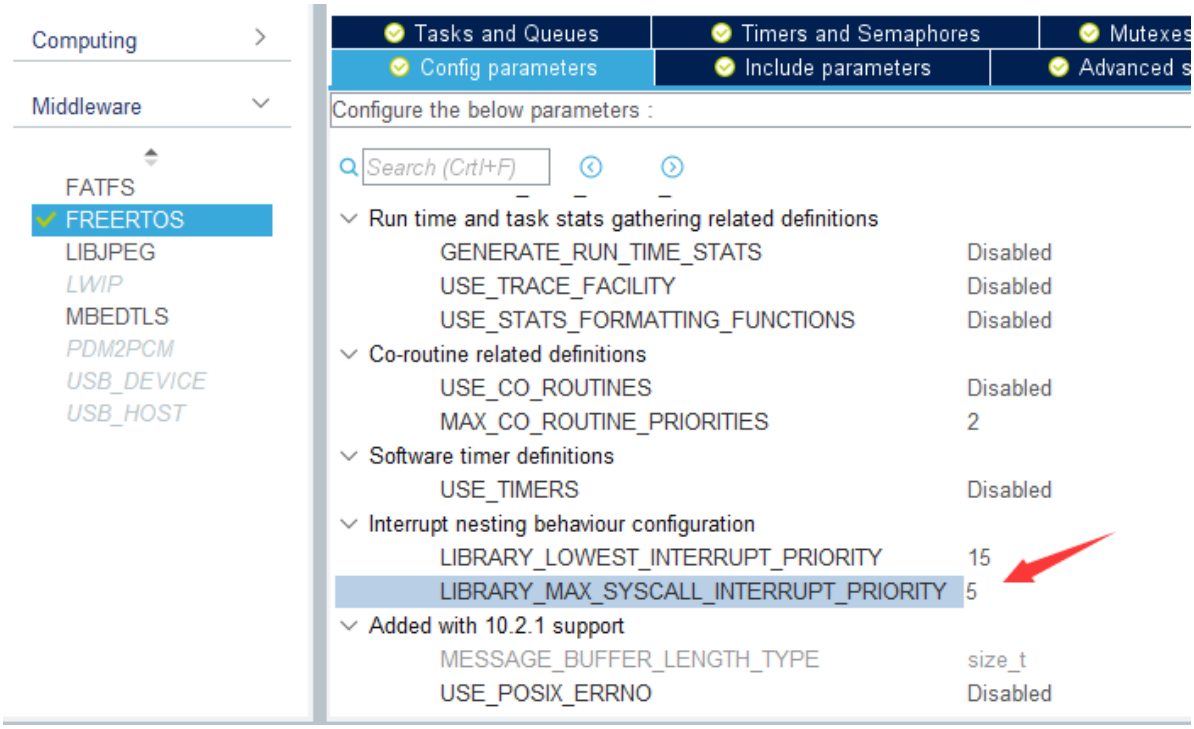


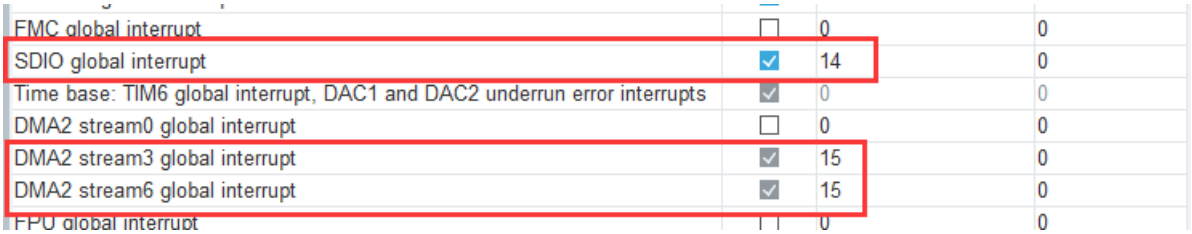
6. 配置NVIC，**Pinout&Clock Configuration**，`System Core -> NVIC`。

注意，SDIO中断优先级**必须高于**DMA2 stream3和DMA2 stream6的中断优先级

紧急避坑！！！如果没有用freertos，那中断优先级设置没啥关系。但如果用了freertos，那SDIO的优先级必须要注意跟freertos区分开来，不能高过他！不然就是mout正常，read等其他操作都返回错误3 not ready。其实当你开启freertos，然后点击NVIC时候，cube会提醒你，要注意函数的中断优先级和freertos优先级的关系。（如果中断处理程序调用RTOS函数，请确保其抢占优先级低于最高的SysCall中断优先级。如FreeRTOS中的“LIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY”）

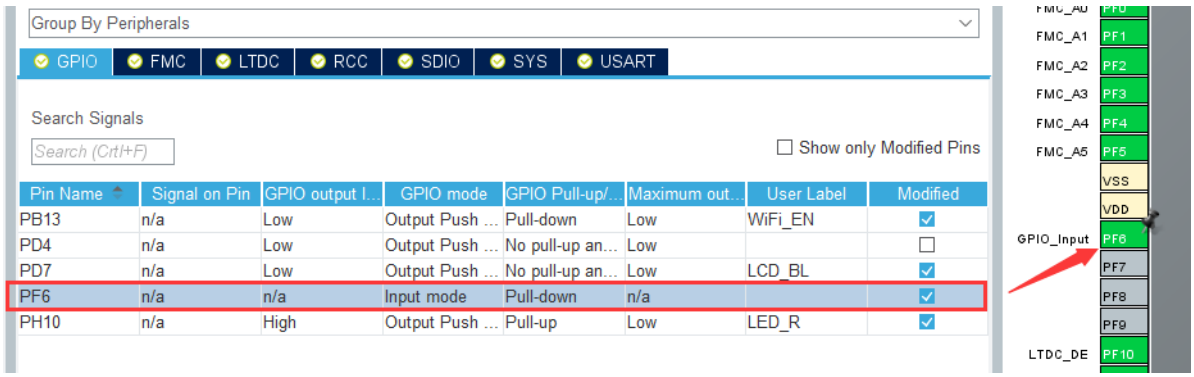


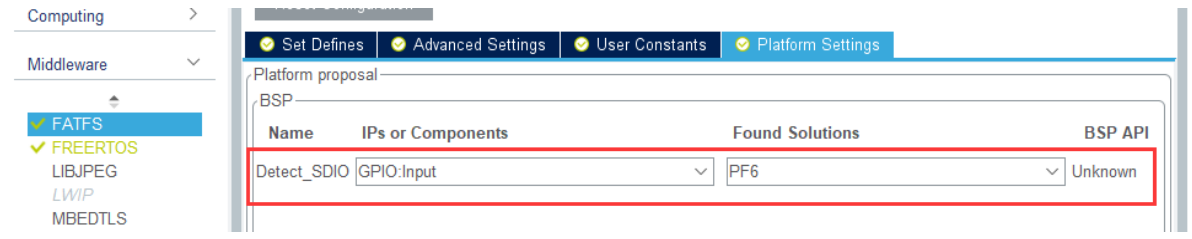




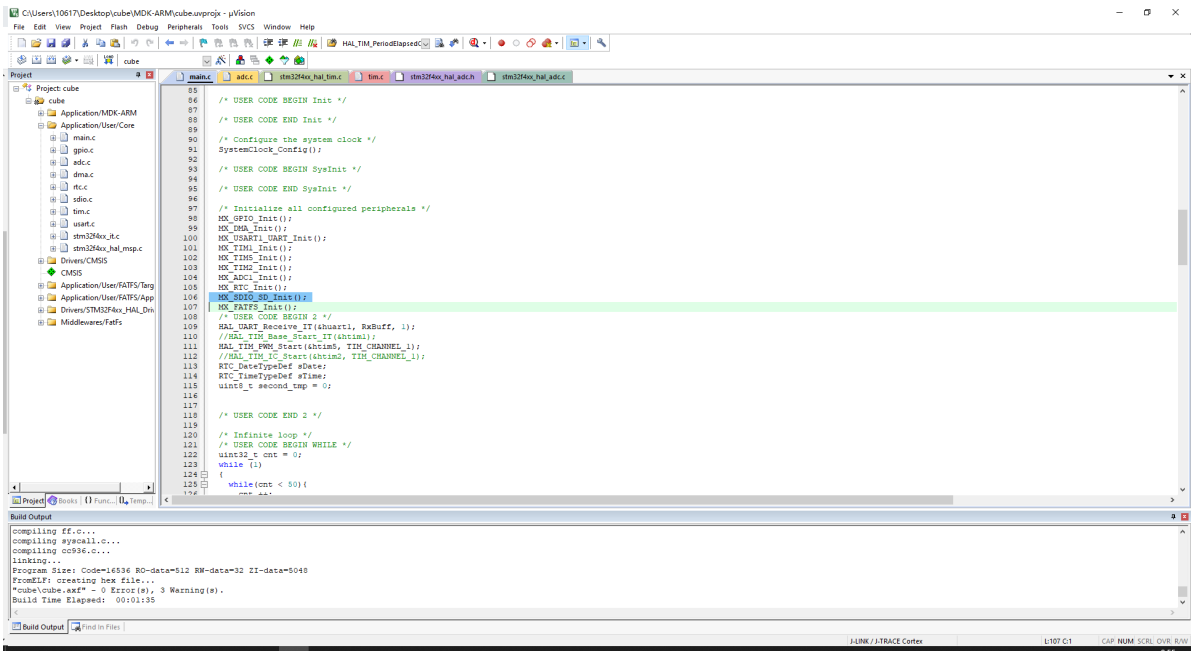
7. 堆栈设置， **Progect Manager -> Project -> Linker Settings**，加大堆栈大小（注意：由于刚才设置长文件名动态缓存存储在堆中，故需要增大堆大小，如果不修改则程序运行时堆会生成溢出，程序进入硬件错误中断(HardFault)，死循环）。

8. 注意这里要找一个闲置的GPIO用作SD卡的检测脚，并拉低。不然open时候可能会一直失败





9. 生成工程， **GENERATE CODE** ， 用KEIL打开



10. 注意，如果是野火的F429开发板，还需要禁用WIFI引脚才行！！！

```
static void BL8782_PDN_INIT(void)
{
  /*定义一个GPIO_InitTypeDef类型的结构体*/
  GPIO_InitTypeDef GPIO_InitStructure;

  RCC_AHB1PeriphClockCmd ( RCC_AHB1Periph_GPIOB, ENABLE);

  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13;
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_OUT;
  GPIO_InitStructure.GPIO_OType = GPIO_OType_PP;
  GPIO_InitStructure.GPIO_PuPd = GPIO_PuPd_DOWN;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_2MHz;
  GPIO_Init(GPIOB, &GPIO_InitStructure);

  GPIO_ResetBits(GPIOB,GPIO_Pin_13);  //禁用WiFi模块
}
```

11. 测试一下

```
UINT bw;
retSD = f_mount(&SDFatFS, SDPath, 0);
if(retSD != FR_OK) {
    printf("Mount Error :%d\r\n", retSD);
}
retSD = f_open(&SDFile, "0:/test.txt", FA_CREATE_ALWAYS | FA_WRITE);
if(retSD != FR_OK){
    printf("Open Error :%d\r\n", retSD);
}
retSD = f_write(&SDFile, "abcde", 5, &bw);
if(retSD != FR_OK){
    printf("Write Error :%d\r\n", retSD);
}
f_close(&SDFile);
retSD = f_open(&SDFile, "0:/test.txt", FA_READ);
char buff[10] = {0};
retSD = f_read(&SDFile, buff, 5, &bw);
if(retSD == FR_OK){
```

```
19        printf("%s\r\n", buff);
20    }
21    f_close(&SDFile);
```

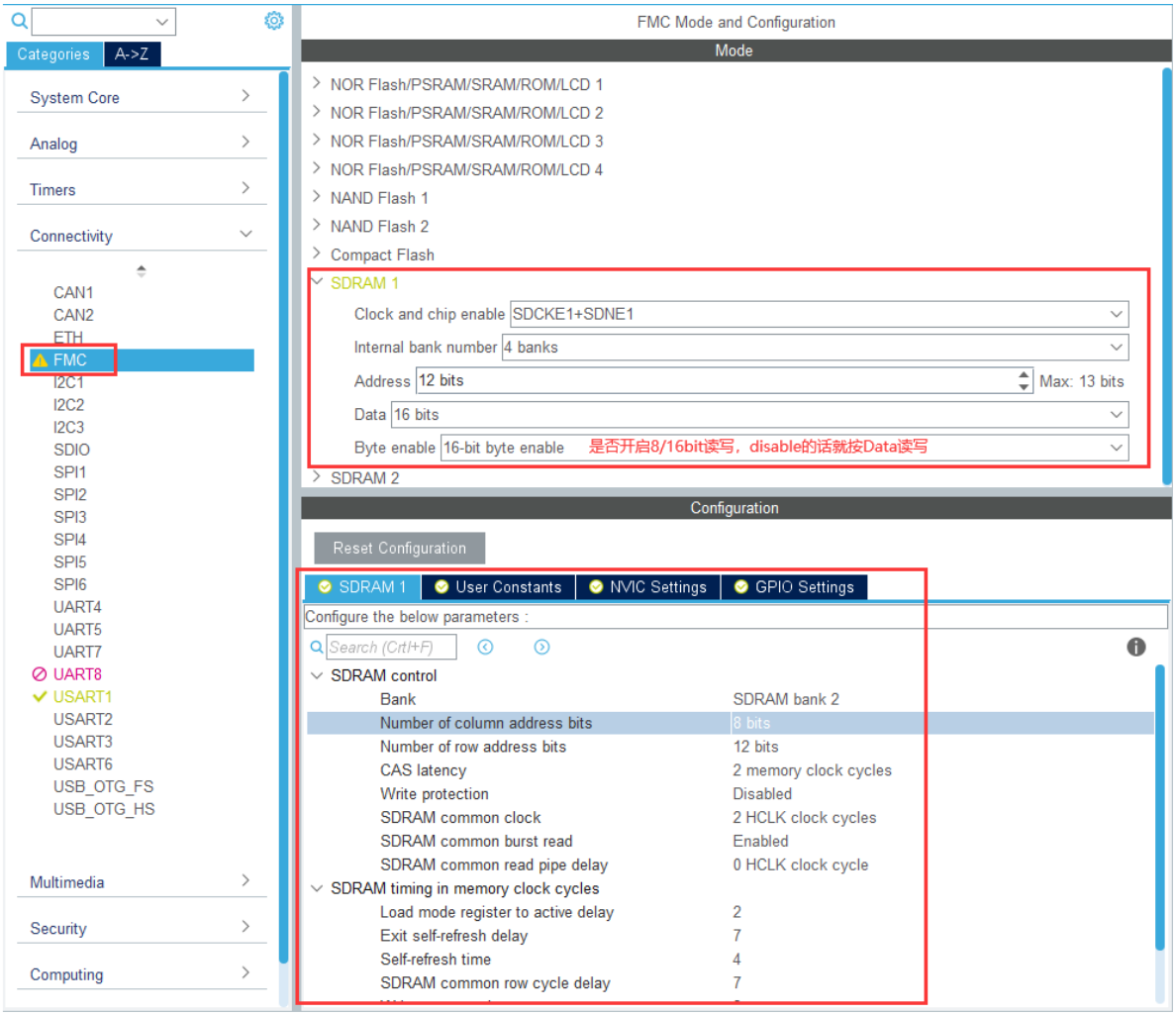
SDRAM

1. 开启FMC功能，**Pinout&Configuration**，`Connectivity -> FMC -> SDRAM2`

   SDRAM1的起始地址为**0XC0000000**，**SDRAM2**的起始地址为**0XD0000000**。

   一般SDRAM都含有**4**个bank。

   Configuration中的参数可从SDRAM的**数据手册**上找到。



各个选项的配置(只做解释，不对应上图)：

- `Clock and chip enable`：**FMC_SDCKE0** 和**FMC_SDCLK0**对应的存储区域1 的地址范围是**0xC000 0000-0xCFFF FFFF**；而**FMC_SDCKE1** 和**FMC_SDCLK1** 对应的存储区域2 的地址范围是**0xD000 0000-0xDFFF FFFF**
- `Bank` 由硬件连接决定需要选择**SDRAM bank 2**
- `Column bit number` 表示列数，**8位**
- `Row bit number` 表示行数，**12位**
- `CAS latency` 表示CAS潜伏期，即上面说的CL，该配置需要与之后的SDRAM模式寄存器的配置相同，这里先配置为**2 memory clock cycles**(对于SDRAM时钟超过133MHz的，则需要配置为3 memory clock cycles)
- `Write protection` 表示写保护，一般配置为**Disabled**
- `SDRAM common clock` 为SDRAM 时钟配置，可选HCLK的2分频\3分频\不使能SDCLK时钟。前面主频配置为216MHz，SDRAM common clock设置为**2分频**，那SDCLK时钟为108MHz，每个时钟周期为

9.25ns

- `SDRAM common burst read` 表示突发读，这里选择使能
- `SDRAM common read pipe delay` 表示CAS潜伏期后延迟多少个时钟在进行读数据，这里选择**0 HCLK clock cycle**
- `Load mode register to active delay` 加载模式寄存器命令和激活或刷新命令之间的延迟，按存储器时钟周期计

SYMBOL	PARAMETER	-5	-6	-7	UNITS
tMRD	LOAD MODE REGISTER command to ACTIVE or REFRESH command	2	2	2	cycle

- `Exit self-refresh delay` 从发出自刷新命令到发出激活命令之间的延迟，按存储器时钟周期数计查数据手册知道其最小值为70ns，由于我们每个时钟周期为9.25ns，所以设为**8** (70÷9.25，向上取整)

		-5		-6		-7		
Symbol	Parameter	Min.	Max.	Min.	Max.	Min.	Max.	Units
tXSR	Exit Self-Refresh to Active Time	60	—	66	—	70	—	ns

- `SDRAM common row cycle delay` 刷新命令和激活命令之间的延迟，以及两个相邻刷新命令之间的延迟， 以存储器时钟周期数表示

  查数据手册知道其最小值为63ns，由于我们每个时钟周期为9.25ns，所以设为**7** (63÷9.25，向上取整)

		-5		-6		-7		
Symbol	Parameter	Min.	Max.	Min.	Max.	Min.	Max.	Units
tRC	Command Period (REF to REF / ACT to ACT)	55	—	60	—	63	—	ns

- `Write recovery time` 写命令和预充电命令之间的延迟，按存储器时钟周期数计

			-5		-6		-7		
Symbol	Parameter		Min.	Max.	Min.	Max.	Min.	Max.	Units
tDPL or tWR	Input Data To Precharge Command Delay time	CAS Latency = 3	2CLK	—	2CLK	—	2CLK	—	ns
		CAS Latency = 2	2CLK	—	2CLK	—	2CLK	—	ns

- `SDRAM common row precharge delay` 预充电命令与其它命令之间的延迟，按存储器时钟周期数计

  查数据手册知道其最小值为15ns，由于我们每个时钟周期为9.25ns，所以设为**2** (15÷9.25，向上取整)

		-5		-6		-7		
Symbol	Parameter	Min.	Max.	Min.	Max.	Min.	Max.	Units
tRP	Command Period (PRE to ACT)	15	—	15	—	15	—	ns

- `Row to column delay` 激活命令与读/写命令之间的延迟，按存储器时钟周期数计

  查数据手册知道其最小值为15ns，由于我们每个时钟周期为9.25ns，所以这里本应该设为**2** (15÷9.25，向上取整)

  但要注意，时序必须满足以下式子：

  TWR ≥ TRAS - TRCD

  TWR ≥ TRC - TRCD - TRP

  其中：TWR = **Write recovery time** = 2

  TRAS = **Self refresh time** = 5

  TRC = **SDRAM common row cycle delay** = 7

  TRP = **SDRAM common row precharge delay** = 2

  TRCD = **Row to column delay**

所以这里Row to column delay应该取3

Symbol	Parameter	-5 Min.	-5 Max.	-6 Min.	-6 Max.	-7 Min.	-7 Max.	Units
tRCD	Active Command To Read / Write Command Delay Time	15	—	15	—	15	—	ns

2. 生成代码， **GENERATE CODE**， 用KEIL打开

```
uint8_t temp[100]__attribute__((at(0xD0000000)));
for(int i=0;i<100;i++){
    temp[i] = i;
}
for(int i=0;i<100;i++){
    printf("%d  ", temp[i]);
}
```

3. 到这里只是借助Cube完成了引脚配置，还需要SDRAM初始化操作和读写函数，可从官方例程里获取，路径：

   C:\Users\10617\STM32Cube\Repository\STM32Cube_FW_F4_V1.25.0\Drivers\BSP\STM32F429I-Discovery\XXX

```
/*****************************SDRAM使能函数*****************************/

/**
  * @brief  对SDRAM芯片进行初始化配置
  * @param  None.
  * @retval None.
  */
static void USER_SDRAM_ENABLE(void)
{
    FMC_SDRAM_CommandTypeDef Command;

  __IO uint32_t tmpmrd =0;

  /* Step 1:  Configure a clock configuration enable command */
  Command.CommandMode             = FMC_SDRAM_CMD_CLK_ENABLE;
  Command.CommandTarget           = FMC_SDRAM_CMD_TARGET_BANK2;
  Command.AutoRefreshNumber       = 1;
  Command.ModeRegisterDefinition  = 0;

  /* Send the command */
  HAL_SDRAM_SendCommand(&hsdram1, &Command, SDRAM_TIMEOUT);

  /* Step 2: Insert 100 us minimum delay */
  /* Inserted delay is equal to 1 ms due to systick time base unit (ms) */
  HAL_Delay(1);

  /* Step 3: Configure a PALL (precharge all) command */
  Command.CommandMode             = FMC_SDRAM_CMD_PALL;
  Command.CommandTarget           = FMC_SDRAM_CMD_TARGET_BANK2;
  Command.AutoRefreshNumber       = 1;
  Command.ModeRegisterDefinition  = 0;

  /* Send the command */
  HAL_SDRAM_SendCommand(&hsdram1, &Command, SDRAM_TIMEOUT);

  /* Step 4: Configure an Auto Refresh command */
  Command.CommandMode             = FMC_SDRAM_CMD_AUTOREFRESH_MODE;
```

```
    Command.CommandTarget           = FMC_SDRAM_CMD_TARGET_BANK2;
    Command.AutoRefreshNumber       = 4;
    Command.ModeRegisterDefinition  = 0;

    /* Send the command */
    HAL_SDRAM_SendCommand(&hsdram1, &Command, SDRAM_TIMEOUT);

    /* Step 5: Program the external memory mode register */
    tmpmrd = (uint32_t)SDRAM_MODEREG_BURST_LENGTH_2          |
                       SDRAM_MODEREG_BURST_TYPE_SEQUENTIAL   |
                       SDRAM_MODEREG_CAS_LATENCY_3           |
                       SDRAM_MODEREG_OPERATING_MODE_STANDARD |
                       SDRAM_MODEREG_WRITEBURST_MODE_SINGLE;

    Command.CommandMode             = FMC_SDRAM_CMD_LOAD_MODE;
    Command.CommandTarget           = FMC_SDRAM_CMD_TARGET_BANK2;
    Command.AutoRefreshNumber       = 1;
    Command.ModeRegisterDefinition  = tmpmrd;

    /* Send the command */
    HAL_SDRAM_SendCommand(&hsdram1, &Command, SDRAM_TIMEOUT);

    /* Step 6: Set the refresh rate counter */
    /* Set the device refresh rate */
    HAL_SDRAM_ProgramRefreshRate(&hsdram1, REFRESH_COUNT);
  }
  /****************************使能函数结束*****************************/
```

或者以我的野火**STM32F429IGT6**的版本SDRAM为**8M**，源码链接：

https://sxf1024.lanzoui.com/b09rf535a 密码:bf5q

添加到工程 `Core` 路径下，然后在KEIL中初始化操作：

(注意这个 `SDRAM_InitSequence();` **不能**加在 `HAL_SDRAM_MspInit()` 后面!!! 因为它这里还没FMC初始化完成!!!! 加在这里是**没有用的**！！！)

```
#include "bsp_sdram.h"
MX_FMC_Init();
SDRAM_InitSequence();
SDRAM_Test();
```

LTDC + DMA2D

务必在上面SDRAM配置成功后，再来搞这个！！！

详细教程看这个： https://zzttzz.gitee.io/blog/posts/7109b92c

但他给的源码还**有点问题**，运行处理没效果。

我提供的源码链接：

https://sxf1024.lanzoui.com/b09rf535a 密码:bf5q

注意：

- 我跟他的配置有点不一样，我的是：
  - Pixel Clock Plparity： Normal Input

    - DMA2D要开一下中断，等级可以不用很高。如果不开的话，有可能会传图时候卡住。
    - LCD-TFT时钟：25MHz
- 层 1 = layer 0，层 2 = layer 1

图 84. 两层与背景混合

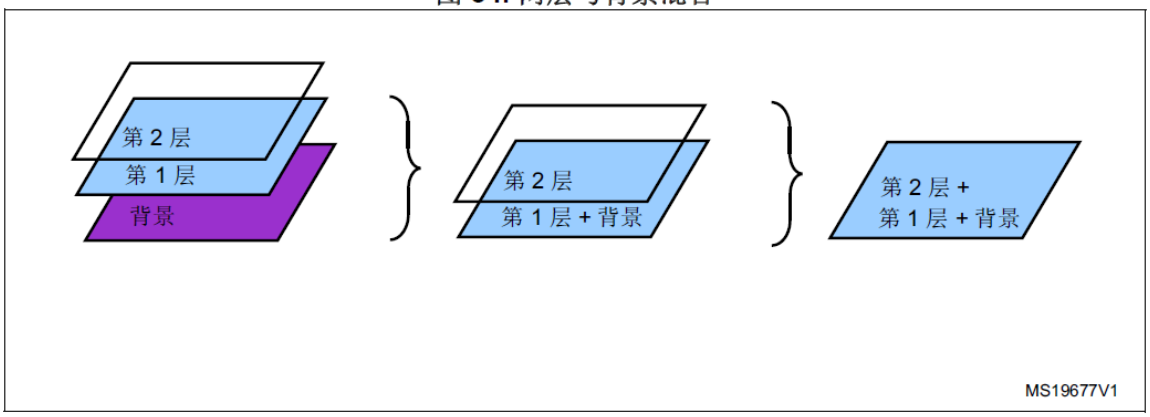
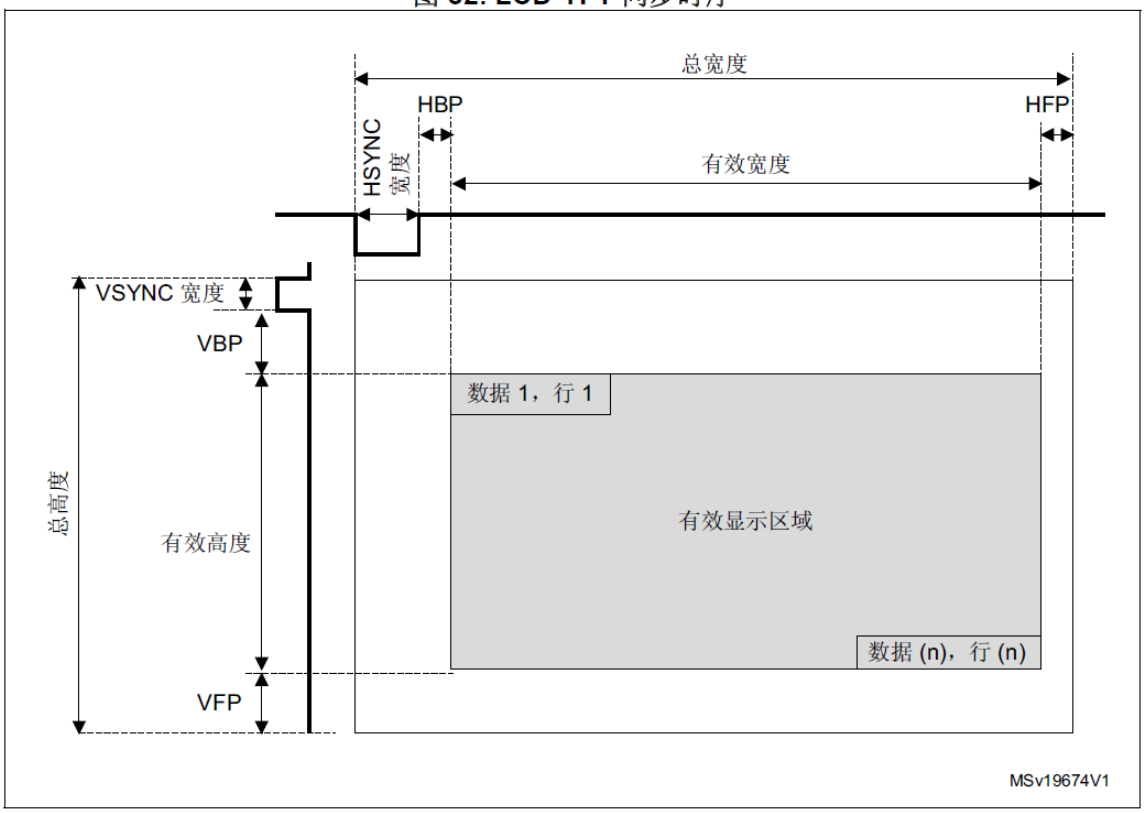


图 82. LCD-TFT 同步时序



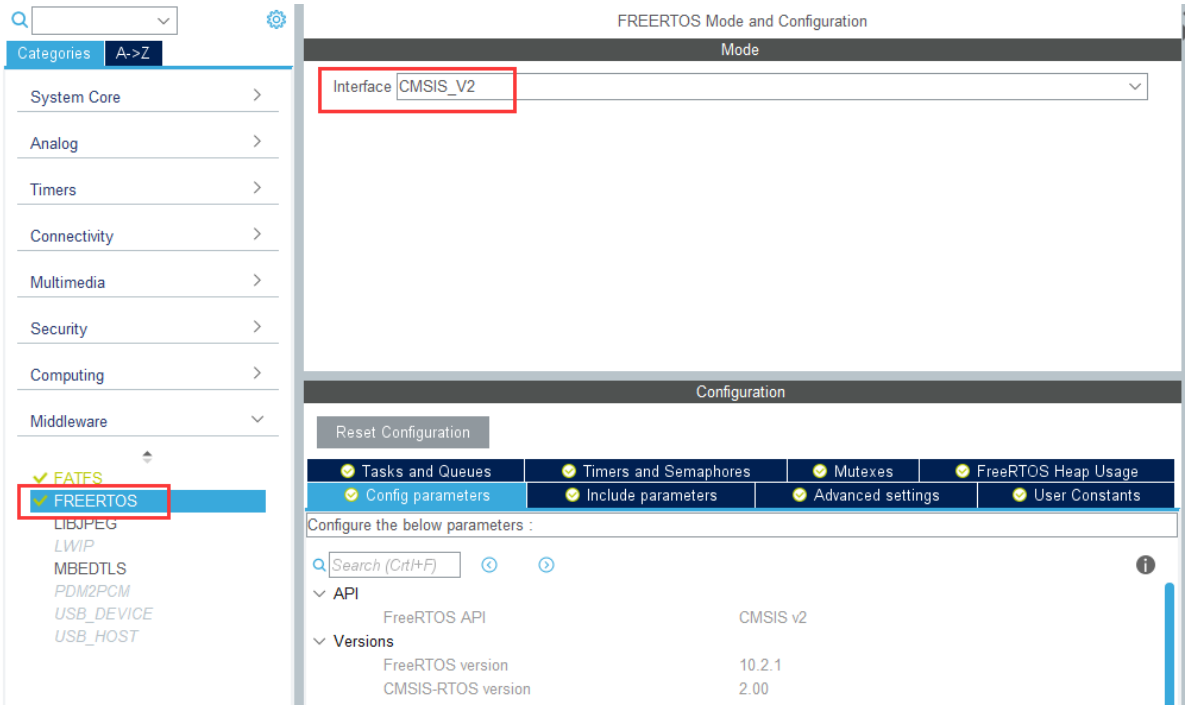
FreeRTOS

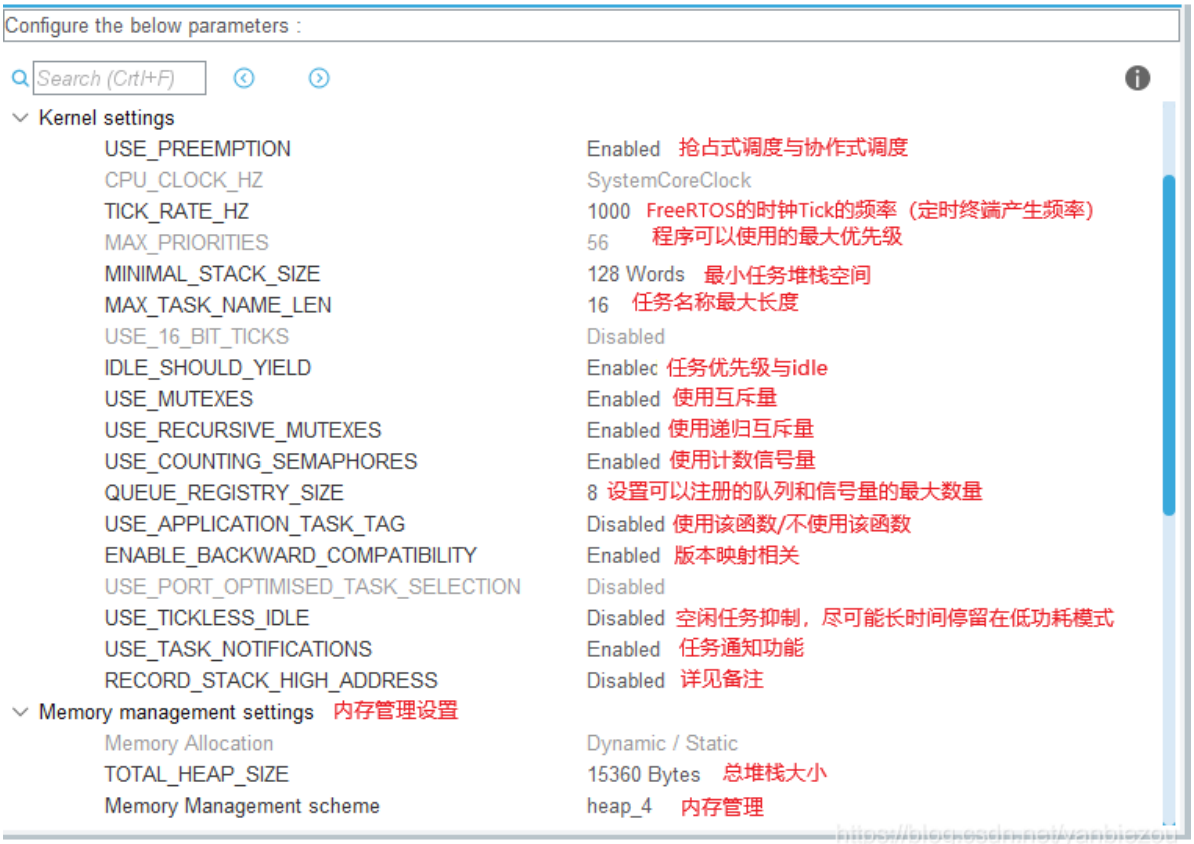
后面要上TouchGFX，这里先加操作系统。

当FreeRTOS遇到FATFS+SDIO时，这里有挺多注意细节的！！！

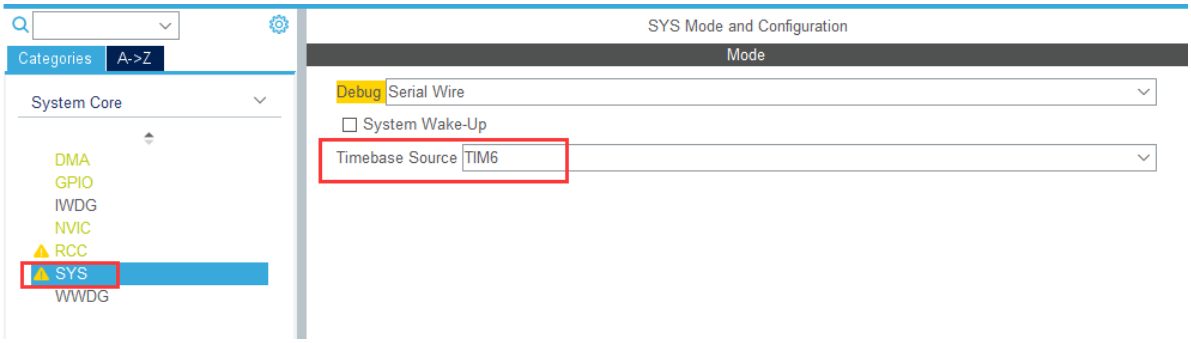
针对初学者，使用STM32CubeMX配置FreeRTOS时，大部分参数默认即可

1. 使能freertos





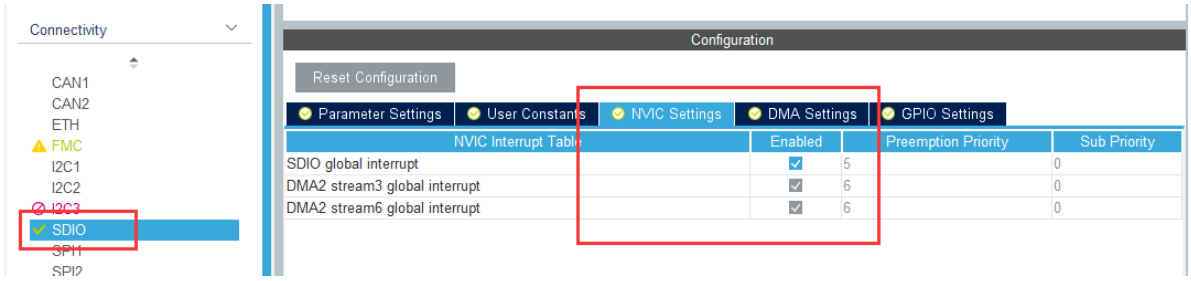
2. 由于freertos工作时会调用systick，为了防止其他进程干扰系统，所以当使用RTOS时，强烈建议使用HAL时基源而不是Systick。HAL时基源可以从SYS下的Pinout选项卡更改。因此更改系统时基源，这里选TIM6



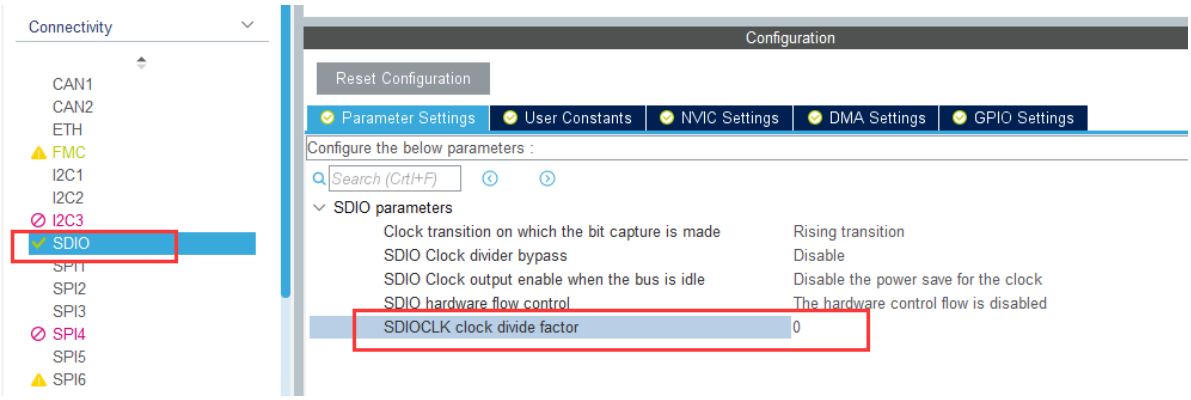
改完之后，注意：中断处理程序调用RTOS函数，请确保它们的优先级比最高的系统调用中断优先级低(数字上高)，例如FreeRTOS中的 `LIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY`

3. 如使用了FreeRTOS，会要求强制使用**DMA模板**的Fatfs，所以**打开DMA通道**，**开中断**，以及**开SDIO中断**是必须的，否则后面配置FATFS无法运行。

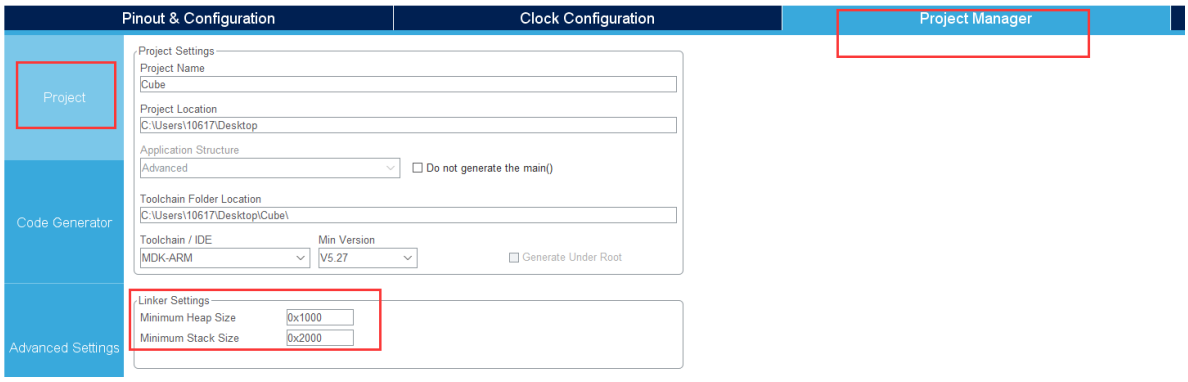
   使能SDIO中断，这里的中断优先级默认不是 `5` 的，而FreeRTOS要求优先级从 `5` 开始



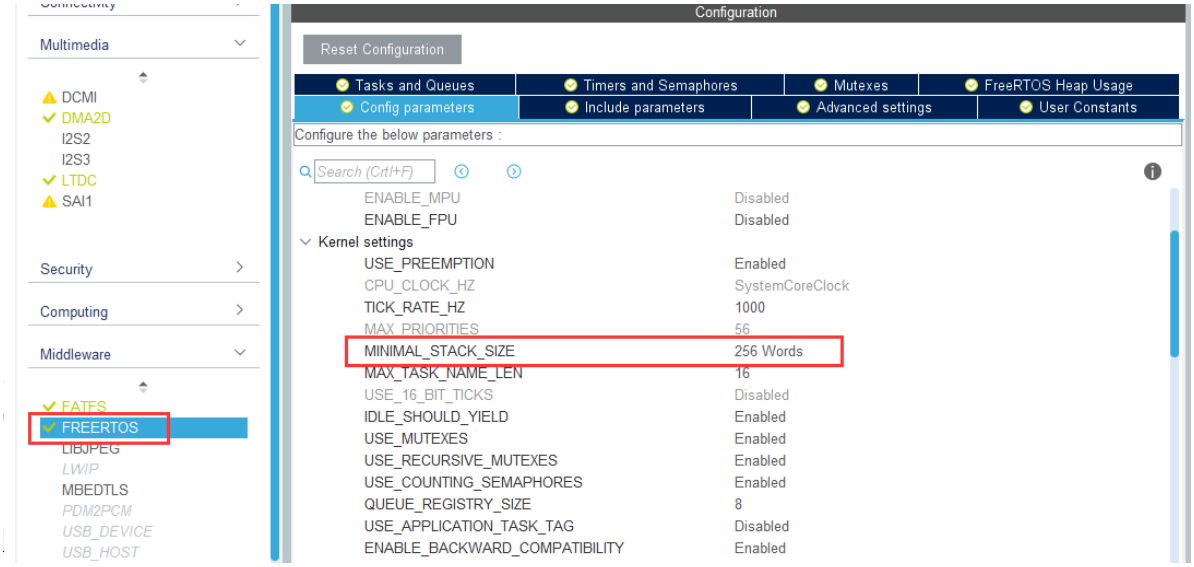
4. 当配置完发现无法mout SD卡，可以尝试加大CLKDIV值



5. 修改 `Project Manager->Project->Linker Setting` 中的**最小堆栈大小**，太小就会无法挂载SD卡或者读写时失败，基本上默认值都是无法正常运行的



6. FreeRTOS基本都是使用默认值，需要增大 `MINIMAL_STACK_SIZE`，默认值是128，使用默认值会造成f_mount直接卡死在内部，这里使用 `256`



7. 生成代码，使用Keil打开。RTOS默认创建了一个 `defaultTask()` ，在 `freertos.c` 文件中
8. 由于SD卡初始化时有检测**读写是否在task任务**中，所以SD读写测试代码需要放到 `defaultTask()` 中

main.c freertos.c

```
107   * @brief  Function implementing the defaultTask thread.
108   * @param  argument: Not used
109   * @retval None
110   */
111 /* USER CODE END Header_StartDefaultTask */
112 void StartDefaultTask(void *argument)
113 {
114   /* USER CODE BEGIN StartDefaultTask */
115
116   UINT bw;
117   retSD = f_mount(&SDFatFS, SDPath, 0);
118   if(retSD != FR_OK) {
119       printf("Mount Error :%d\r\n", retSD);
120   }
121   retSD = f_open(&SDFile, "0:/test.txt", FA_CREATE_ALWAYS | FA_WRITE);
122   if(retSD != FR_OK){
123       printf("Open Error :%d\r\n", retSD);
124   }
125   retSD = f_write(&SDFile, "abcde", 5, &bw);
126   if(retSD != FR_OK){
127       printf("Write Error :%d\r\n", retSD);
128   }
129   f_close(&SDFile);
130   retSD = f_open(&SDFile, "0:/test.txt", FA_READ);
131   char buff[10] = {0};
132   retSD = f_read(&SDFile, buff, 5, &bw);
133   if(retSD == FR_OK){
134       printf("%s\r\n", buff);
135   }else{
136     printf("Read Error: %d\r\n", retSD);
137   }
138   f_close(&SDFile);
```

9. 由于任务调度启动后就不再往下执行，所以把之前的LTDC显示测试代码也可以放到task中，或往前挪一挪

```
106   MX_LTDC_Init();
107   MX_SDIO_SD_Init();
108   MX_FATFS_Init();
109   /* USER CODE BEGIN 2 */
110   SDRAM_InitSequence();
111
112   printf("Start\r\n");
113   LCD_DisplayOn();
114   LCD_SetLayerVisible(1,DISABLE);
115   LCD_SetLayerVisible(0,ENABLE);
116   LCD_SetTransparency(1,0);
117   LCD_SetTransparency(0,255);
118   LCD_SelectLayer(0);
119   LCD_Clear(LCD_COLOR_BLUE);
120   LCD_SetTextColor(LCD_COLOR_RED);
121   LCD_SetBackColor(LCD_COLOR_BLUE);
122   LCD_SetFont(&Font24);
123   LCD_DisplayStringLine(1,(uint8_t *)"ok!!");
124   LCD_DrawCircle(200,200,100);
125
126   // 使能串口中断接收
127   HAL_UART_Receive_IT(&huart1, (uint8_t*)&DataTemp_UART1, 1);
128
129   /* USER CODE END 2 */
130
131   /* Init scheduler */
132   osKernelInitialize();  /* Call init function for freertos objects (in
133   MX_FREERTOS_Init();
134   /* Start scheduler */
135   osKernelStart();
136
```

各种配置初始化

LTDC显示测试

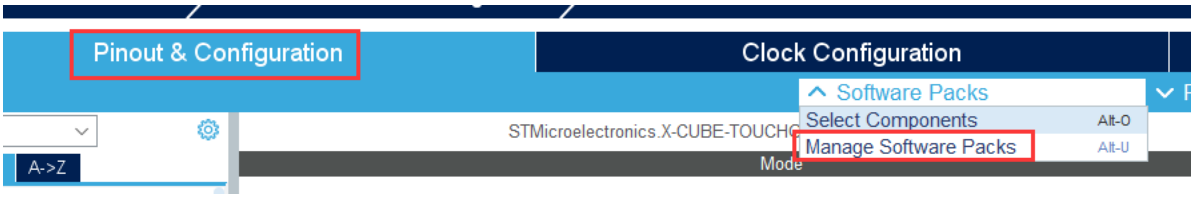
开始任务调度

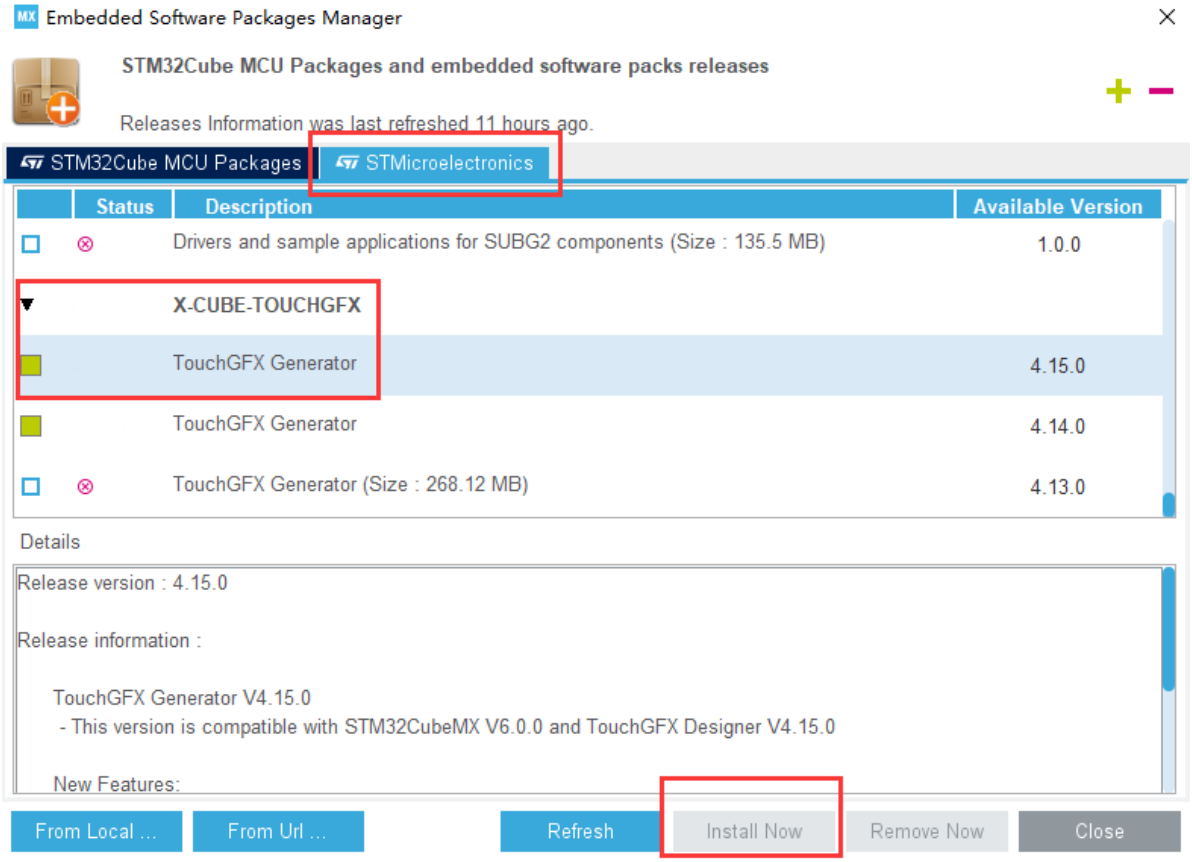
10. 其他的无需改动，运行即可！
11. 发现它创建任务用的是 `osThreadNew`，对 `xTaskCreate` 又进行了封装，省去了繁琐
12. 网上有人说要“**在TIM6中断中，加入临界段保护（或进入中断保护）**”，不知真假，没试

TouchGFX显示

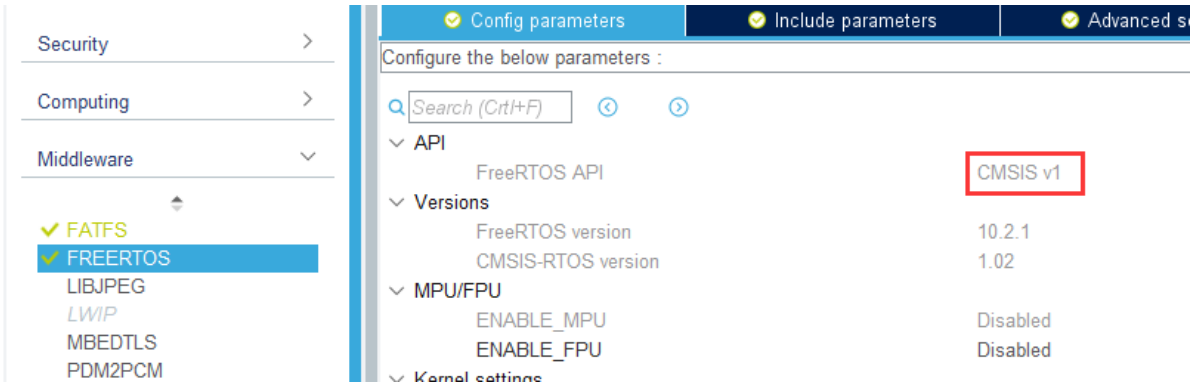
虽然方便，但它是用C++开发的，所以不是特别友好...说白了就是看不懂，不知道怎么去改

1. 先在 `Cube` 里安装 `TouchGFX` 包

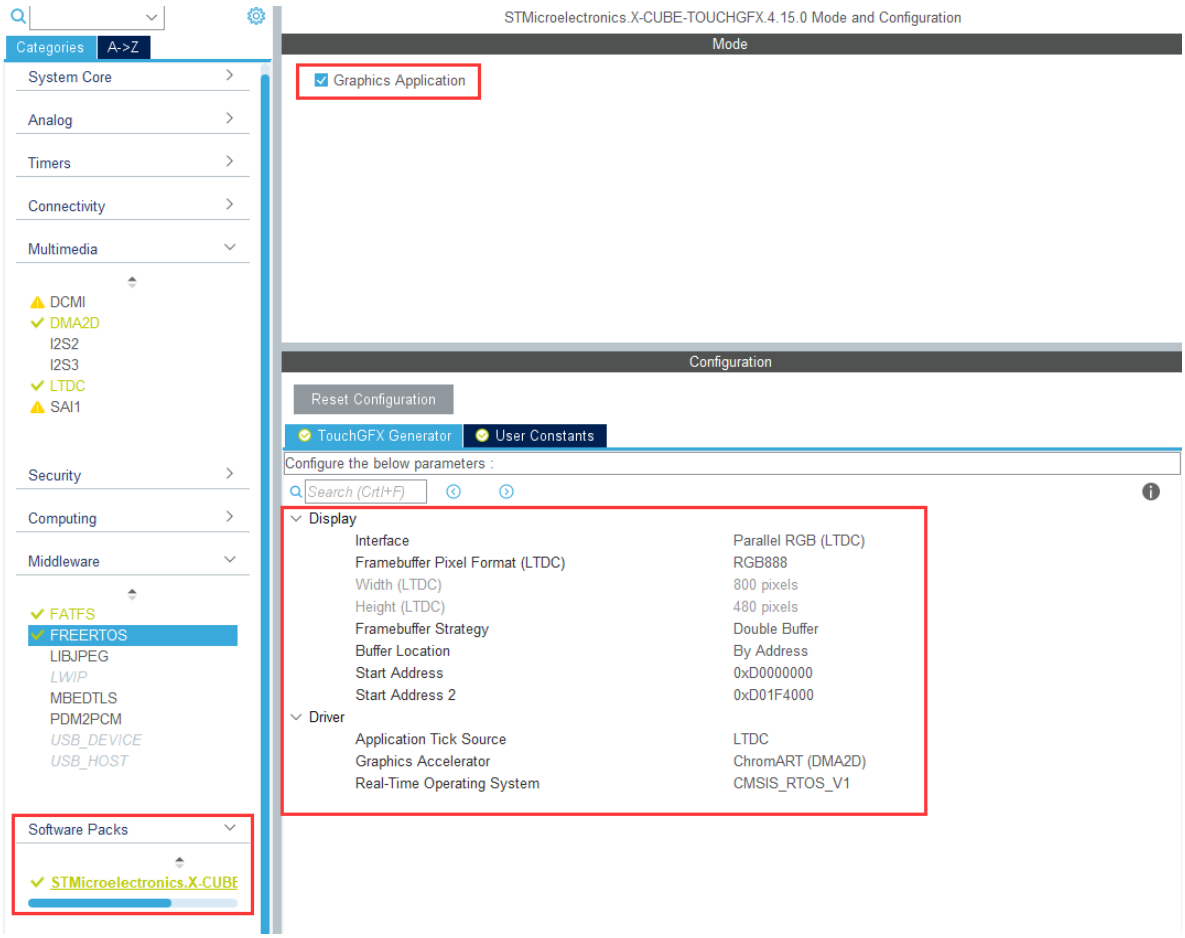




2. 这里要注意，`FreeRTOS`要选`CMSIS V1`版本！！！



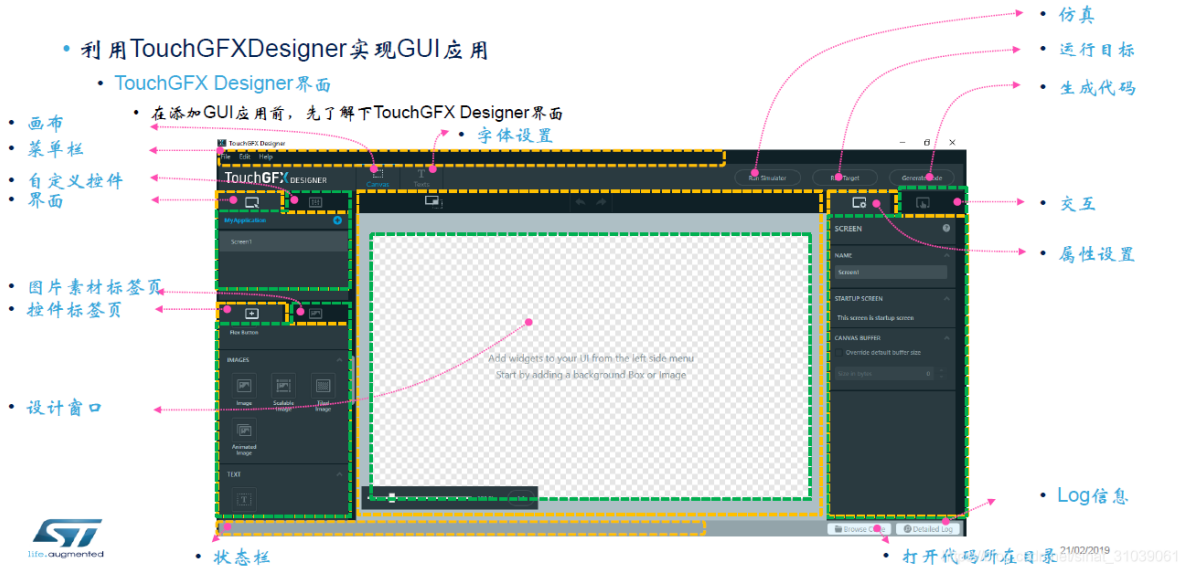
3. 开启`TouchGFX`包，并配置如下

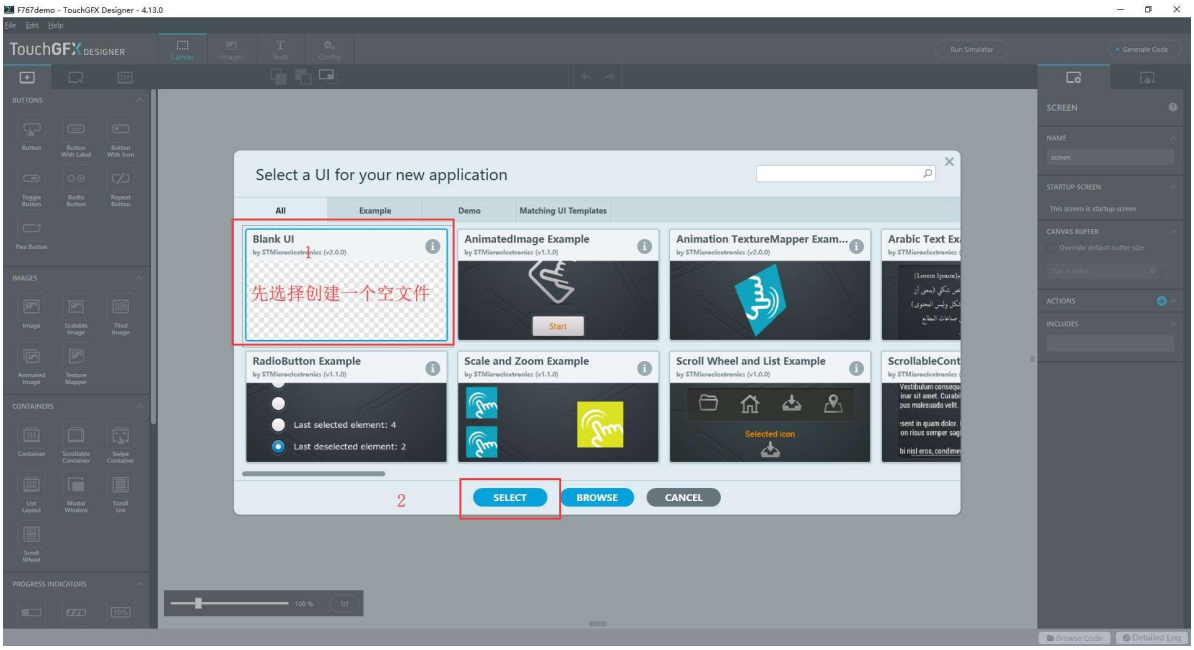


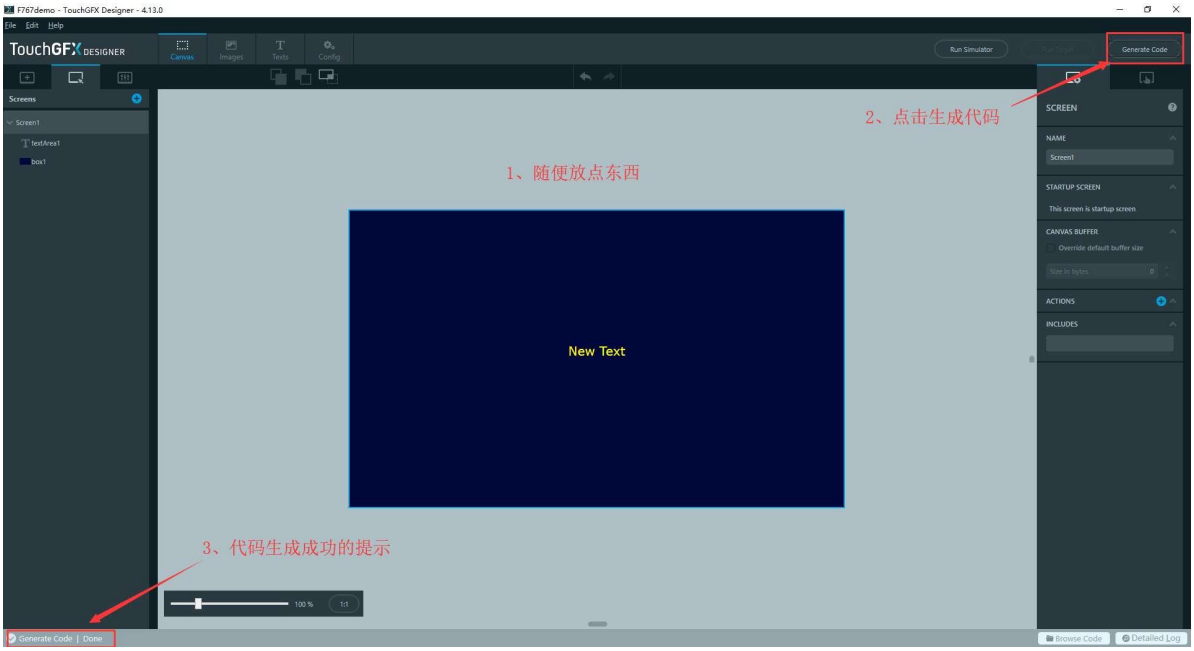
3. 生成工程，但先别用keil打开！！！
4. 到目录下安装TouchGFX Designer软件，它用来绘制界面的

```
C:\Users\10617\STM32Cube\Repository\Packs\STMicroelectronics\X-CUBE-TOUCHGFX\4.15.0\Utilities\PC_Software\TouchGFXDesigner\TouchGFX-4.14.0.msi
```

5. 安装完成后，回到Cube工程目录，会发现多了一个`TouchGFX`文件夹，打开其中的`.touchgfx`文件，绘制界面



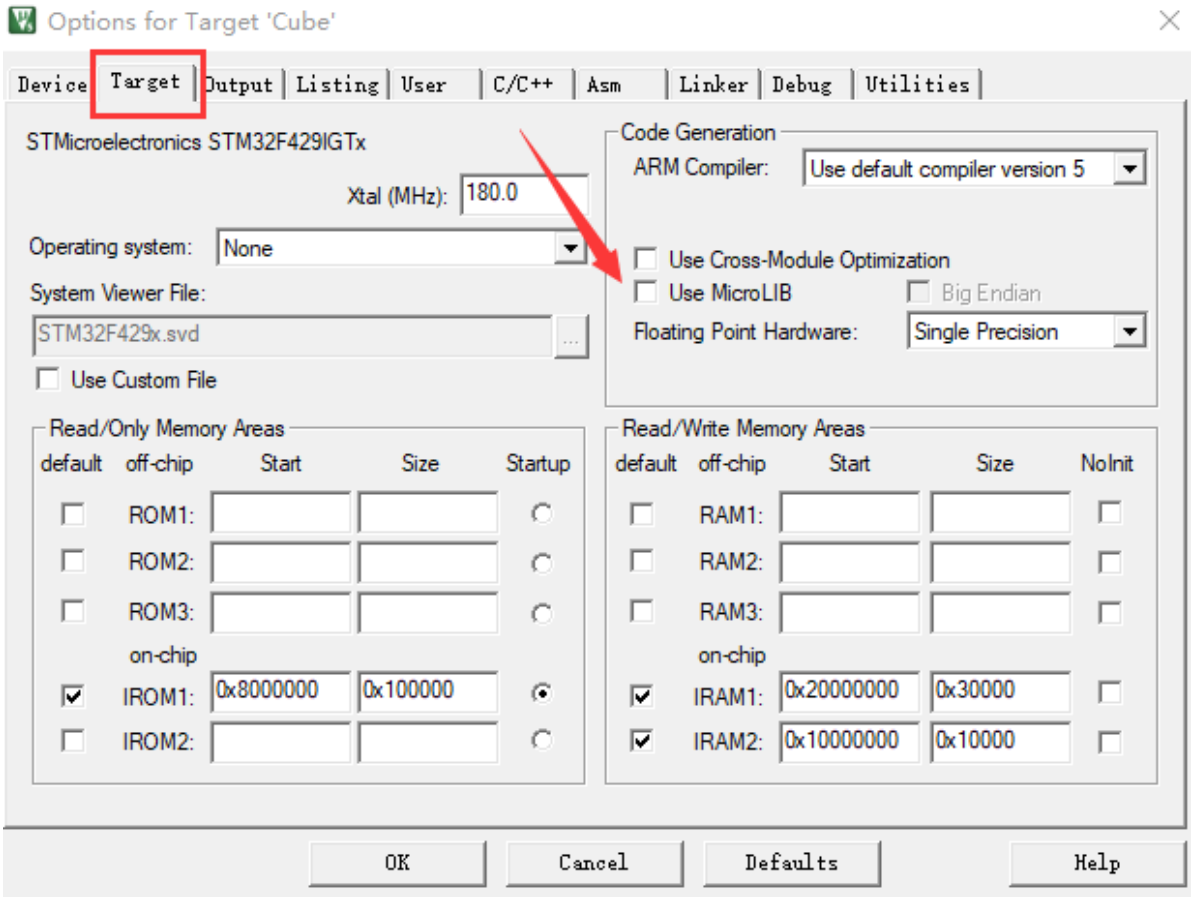




6. 然后就可以打开keil，添加以下内容

```
#include "app_touchgfx.h"

// 开启LCD
LCD_DisplayOn();
LCD_SetLayerVisible(1,DISABLE);
LCD_SetLayerVisible(0,ENABLE);
LCD_SetTransparency(1,0);
LCD_SetTransparency(0,255);
LCD_SelectLayer(0);
// 显示TouchGFX内容
MX_TouchGFX_Process();
```

7. 这里要注意！由于 `MicroLib` **不支持** `C++`，所以需要在keil里**取消勾选**！！！



8. 但 `printf` 函数又**需要** `MicroLib` 支持，所以需要**添加**以下函数，**否则会开机无法运行**！！！

```c
#if 1
#pragma import(__use_no_semihosting)
//标准库需要的支持函数
struct __FILE
{
    int handle;
};

FILE __stdout;
//定义_sys_exit()以避免使用半主机模式
void _sys_exit(int x)
{
    x = x;
}
void _ttywrch(int ch)
{
    ch = ch;
}
//重定义fputc函数
int fputc(int ch, FILE *f)
{
    HAL_UART_Transmit(&huart1, (uint8_t*)&ch, 1, 0XFF);
    return ch;
}
#endif
```

9. 完成以上内容后，编译、烧录即可

LittleVGL

奈何不会C++，只能另谋出路，LittltVGL设计的界面似乎还挺好看的，而且用C编写，兼容C++，更新很活跃。EMWIN风格类似window XP ， littlevGL风格类似android 。移植很简单（并没有多简单）。

官网github： https://github.com/lvgl/lvgl

官方文档： https://docs.lvgl.io/latest/en/html

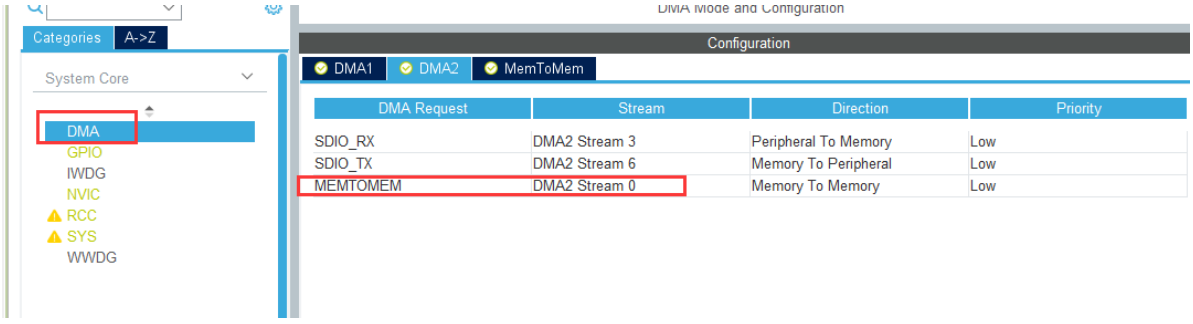
官方推荐方法学习路线:

- 点击在线演示以查看LVGL的运行情况(3分钟)
- 阅读文档的介绍页面(5分钟)
- 熟悉Quick overview页面的基础知识(15分钟)
- 设置模拟器(10分钟)
- 尝试一些例子
- 移植LVGL到一个板子。请参阅移植指南或准备使用项目
- 阅读概述页，更好地了解库(2-3小时)
- 查看小部件的文档，了解它们的特性和用法
- 如果你有问题可以去论坛
- 阅读贡献指南，了解如何帮助改进LVGL(15分钟)

1、 教程可交叉参考以下这几篇，取长补短吧：

- csdn移植教程： https://blog.csdn.net/t01051/article/details/108748462
- 微雪移植教程： https://www.waveshare.net/study/article-964-1.html
- No space解决方案： https://blog.csdn.net/qq_36075612/article/details/107671669
- csdn移植教程： https://blog.csdn.net/zcy_cyril/article/details/107457371
- 放SRAM/SDRAM： https://blog.csdn.net/qq_41543888/article/details/106532577

2、在Cube里开一个MTM的DMA（或者不开它，直接用DMA2D）



3、生成工程，修改`disp_flush`函数

```
/***********************
 *  STATIC VARIABLES
 ***********************/
static __IO uint16_t * my_fb = (__IO uint16_t*) (0xD0000000);

static DMA_HandleTypeDef  DmaHandle;
static int32_t            x1_flush;
//static int32_t            y1_flush;
static int32_t            x2_flush;
static int32_t            y2_fill;
static int32_t            y_fill_act;
static const lv_color_t * buf_to_flush;
static lv_disp_t *our_disp = NULL;

```

```c
/*********************
 *      INCLUDES
 *********************/
#include "lv_port_disp.h"
#include "bsp_lcd.h"
#include "dma2d.h"
#include "stm32f4xx_hal_dma.h"
#include "dma.h"

static void disp_flush(lv_disp_drv_t * disp_drv, const lv_area_t * area, lv_color_t * color_p)
{
//    int32_t x;
//    int32_t y;
//    for(y = area->y1; y <= area->y2; y++) {
//        for(x = area->x1; x <= area->x2; x++) {
//            /* Put a pixel to the display. For example: */
//            /* put_px(x, y, *color_p)*/
////                        LCD_FillRect_C(area->x1, area->y1, area->x2-area->x1, area->y2-area->y1, (uint32_t)color_p);
////                        LCD_DrawPixel(x, y, (uint32_t)color_p->full);
//                    LCD_FillRect_C(x, y, 1, 1, (uint32_t)color_p->full);
//            color_p++;
//        }
//    }

    int32_t x1 = area->x1;
    int32_t x2 = area->x2;
    int32_t y1 = area->y1;
    int32_t y2 = area->y2;
    /*Return if the area is out the screen*/
    if(x2 < 0) return;
    if(y2 < 0) return;
    if(x1 > LV_HOR_RES_MAX - 1) return;
    if(y1 > LV_VER_RES_MAX - 1) return;
    /*Truncate the area to the screen*/
    int32_t act_x1 = x1 < 0 ? 0 : x1;
    int32_t act_y1 = y1 < 0 ? 0 : y1;
    int32_t act_x2 = x2 > LV_HOR_RES_MAX - 1 ? LV_HOR_RES_MAX - 1 : x2;
    int32_t act_y2 = y2 > LV_VER_RES_MAX - 1 ? LV_VER_RES_MAX - 1 : y2;
    x1_flush = act_x1;
    //      y1_flush = act_y1;
    x2_flush = act_x2;
    y2_fill = act_y2;
    y_fill_act = act_y1;
    buf_to_flush = color;
    HAL_StatusTypeDef err;
    uint32_t length = (x2_flush - x1_flush + 1);
    #if LV_COLOR_DEPTH == 24 || LV_COLOR_DEPTH == 32
    length *= 2; /* STM32 DMA uses 16-bit chunks so multiply by 2 for 32-bit color */
    #endif
    err = HAL_DMA_Start_IT(&hdma_memtomem_dma2_stream0,(uint32_t)buf_to_flush, (uint32_t)&my_fb[y_fill_act * LV_HOR_RES_MAX + x1_flush], length);
    if(err != HAL_OK) {
        printf("disp_flush %d\r\n",err);
        while(1);   /*Halt on error*/
    }
```

```
69
70      lv_disp_flush_ready(disp_drv);
71  }
```

注意！！！

微雪这个函数**有点问题**，如果遇到显示**不正确**的时候，建议改成以下这个试试：

```
1   int32_t x1 = area->x1;
2   int32_t x2 = area->x2;
3   int32_t y1 = area->y1;
4   int32_t y2 = area->y2;
5   /*Return if the area is out the screen*/
6   if(x2 < 0) return;
7   if(y2 < 0) return;
8   if(x1 > LV_HOR_RES_MAX - 1) return;
9   if(y1 > LV_VER_RES_MAX - 1) return;
10  /*Truncate the area to the screen*/
11  int32_t act_x1 = x1 < 0 ? 0 : x1;
12  int32_t act_y1 = y1 < 0 ? 0 : y1;
13  int32_t act_x2 = x2 > LV_HOR_RES_MAX - 1 ? LV_HOR_RES_MAX - 1 : x2;
14  int32_t act_y2 = y2 > LV_VER_RES_MAX - 1 ? LV_VER_RES_MAX - 1 : y2;
15
16
17  for(int32_t y = act_y1; y <= act_y2; y++) {
18      for(int32_t x = act_x1; x <= act_x2; x++) {
19          /* Put a pixel to the display. For example: */
20          /* put_px(x, y, *color_p)*/
21          my_fb[y*LV_HOR_RES_MAX+x] = (uint32_t)color_p->full;
22          color_p++;
23      }
24  }
```

4、按着上面教程，把littlvgl的显存地址改为SDRAM的

5、由于用作时基的TIM6的中断时间是`100ms`，所以我们可以新开一个定时器如TIM7，设置它的中断时间为`1~5ms`。但好像是需要手动启动定时器的：

```
1   HAL_TIM_Base_Start_IT(&htim7);
```

6、keil测试：

```
1   lv_init();
2   lv_port_disp_init();
3
4   // 开启LCD
5   LCD_DisplayOn();
6   LCD_SetLayerVisible(1,DISABLE);
7   LCD_SetLayerVisible(0,ENABLE);
8   LCD_SetTransparency(1,0);
9   LCD_SetTransparency(0,255);
10  LCD_SelectLayer(0);
11  LCD_Clear(LCD_COLOR_BLUE);
12
13  /*Create a Label on the currently active screen*/
14  lv_obj_t * label1 =  lv_label_create(lv_scr_act(), NULL);
15  /*Modify the Label's text*/
16  lv_label_set_text(label1, "Hello world!");
17  /* Align the Label to the center
```

```
      * NULL means align on parent (which is the screen now)
      * 0, 0 at the end means an x, y offset after alignment*/
  lv_obj_align(label1, NULL, LV_ALIGN_CENTER, 0, 0);

  #include "bsp_lcd.h"
  static void disp_flush(lv_disp_drv_t * disp_drv, const lv_area_t * area,
  lv_color_t * color_p)
  {
    int32_t x;
    int32_t y;
      for(y = area->y1; y <= area->y2; y++) {
        for(x = area->x1; x <= area->x2; x++) {
            LCD_DrawPixel(x, y, (uint32_t)color_p->full);
            color_p++;
        }
      }
      lv_disp_flush_ready(disp_drv);
  }
```

当然，我的源码链接（只有显示部分，触摸目前没用到）：

https://sxf1024.lanzoui.com/b09rf535a 密码:bf5q

注意！！！：
一定要**先执行**初始化`lv_init(); lv_port_disp_init();`，再做其他ui操作，不然会死活不显示出来！！！

附，我的一些理解：

- display是buff区，screen是一整个界面，界面里可以放控件和窗口win，窗口里还能放控件，一个控件可能有多个part
- 创建 `screen`： `lv_obj_t* screen1 = lv_obj_create(NULL, NULL);`
- 创建 `window`： `lv_obj_t* win = lv_win_create(screen1, NULL);`
- 创建 `label`： `lv_obj_t * label1 =  lv_label_create(win, NULL);`
- `label` 中设置text： lv_label_set_text(label1, "Hello world!");`
- 居中：`lv_obj_align(label1, NULL, LV_ALIGN_CENTER, 0, 0);`
- `screen` 切换：`lv_scr_load(screen1);`
- 添加 `style`：

```
  static lv_style_t loading_style;
  lv_obj_t* loading_label = lv_label_create(lv_scr_act(), NULL);
  lv_label_set_text(loading_label, "Loading...");
  lv_obj_align(loading_label, NULL, LV_ALIGN_CENTER, 0, 0);
  lv_style_init(&loading_style);
  lv_style_set_text_color(&loading_style, LV_STATE_DEFAULT, LV_COLOR_BLUE);
  lv_style_set_text_font(&loading_style,LV_STATE_DEFAULT, &lv_font_montserrat_24);
  lv_obj_add_style(loading_label, LV_OBJ_PART_MAIN, &loading_style);
```

显示图片

C数组形式

图片转C文件链接： https://littlevgl.com/image-to-c-array

- 首先我们需要将文件加入到我们的工程树中
- 然后在需要的地方声明一下就可以了，可以用下面两种方式：

```
extern const lv_img_t my_image_name;
LV_IMG_DECLARE(my_image_name);
```

- 我们再来看看转换出来的图片文件的一些信息，包括宽度，高度，大小等。就在我们的刚下载的文件最后就可以看到一个结构体，如下：

```
const lv_img_dsc_t WaveShare_LOGO = {
  .header.always_zero = 0,
  .header.w = 287,
  .header.h = 81,
  .data_size = 11728,
  .header.cf = LV_IMG_CF_INDEXED_4BIT,
  .data = WaveShare_LOGO_map,
};
```

- 使用示例：

```
// 先声明一下外部图片结构体
LV_IMG_DECLARE(WaveShare_LOGO)
// 创建一个图片
lv_obj_t * img1 = lv_img_create(lv_scr_act(), NULL);
// 将数组内容放入
lv_img_set_src(img1, &WaveShare_LOGO);
// 图片在屏幕居中
lv_obj_align(img1, NULL, LV_ALIGN_CENTER, 0, -20);
```

canvas画图

```
// 声明一个buff
static lv_color_t buffer[LV_CANVAS_BUF_SIZE_TRUE_COLOR(48, 48)];
// 创建canvas
lv_obj_t* canvas = lv_canvas_create(lv_scr_act(), NULL);
// 关联canvas与buff
lv_canvas_set_buffer(canvas, buffer, 48, 48, LV_IMG_CF_TRUE_COLOR);
// 背景涂色
```

```
lv_canvas_fill_bg(canvas, LV_COLOR_BLUE, LV_OPA_50);
// 在画布上画点
lv_color_t c0;
c0.full = 0;
uint32_t x;
uint32_t y;
for( y = 10; y < 30; y++) {
    for( x = 5; x < 20; x++) {
        // 这里的x,y都是相对父元素而言
        lv_canvas_set_px(canvas, x, y, c0);
    }
}
```

文件系统

1. 下载以下链接中的几个文件： https://github.com/lvgl/lv_fs_if
2. 将下载的.c/.h添加到工程中
3. 添加这些行到 `lv_conf.h`：

```
/*File system interface*/
#define LV_USE_FS_IF    1
#if LV_USE_FS_IF
#  define LV_FS_IF_FATFS    '\0'     // 'S'
#  define LV_FS_IF_PC       '\0'
#endif  /*LV_USE_FS_IF*/
```

4. 通过将 `'\0'` 更改为要用于该驱动器的字母来启用所需的接口。如 `'S'` 表示FATFS的SD卡
5. 调用 `lv_fs_if_init()` 来注册启用的接口
6. 使用 `lv_fs_fatfs.c` 中提供的函数完成操作
7. 初始化图像，需要以下回调:
    - open
    - close
    - read
    - seek
    - tell
8. 使用示例：

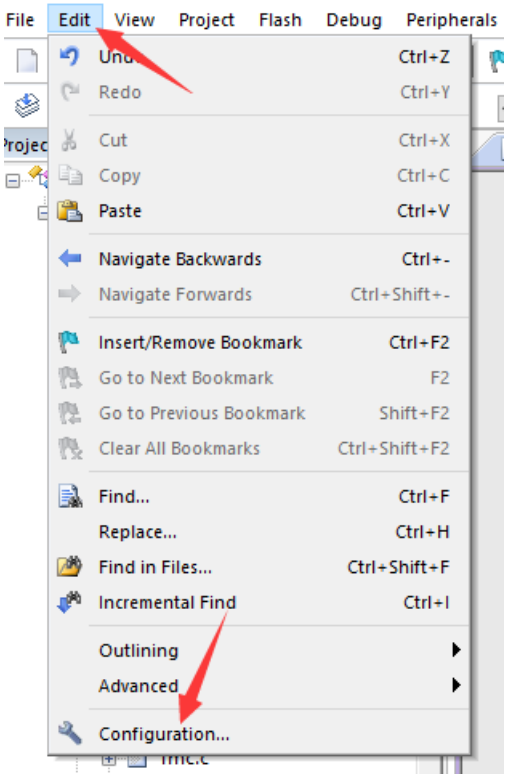
```
// 挂载SD卡
retSD = f_mount(&SDFatFS, SDPath, 0);
// 文件系统初始化
lv_fs_if_init();
// 创建一个图像
lv_obj_t *icon = lv_img_create(lv_scr_act(), NULL);
// SD卡文件绑定到图像
lv_img_set_src(icon, "S:0.bin");
// 居中显示
lv_obj_align(icon, NULL, LV_ALIGN_CENTER, 0, 0);
```

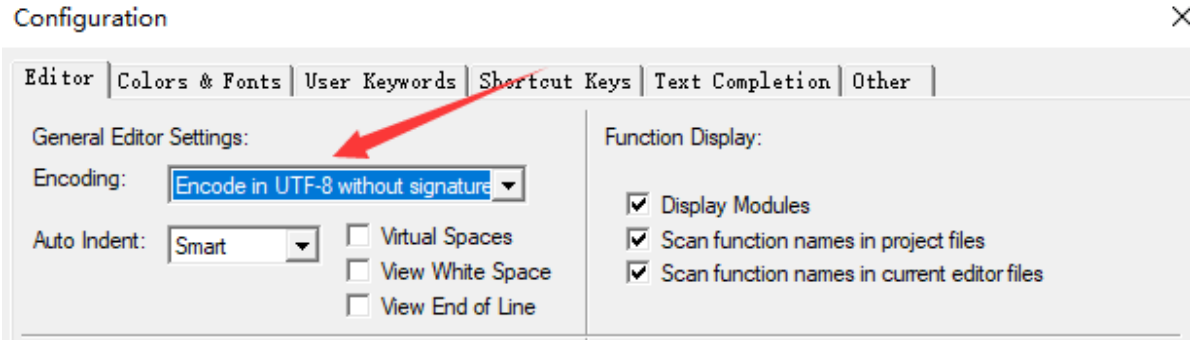
显示中文

教程看这篇： http://www.lfly.xyz/forum.php?mod=viewthread&tid=21

LvglFontTool V0.4： http://www.lfly.xyz/forum.php?mod=viewthread&tid=11&extra=page%3D1

注意keil工程必须是UTF8编码！





生成的myFont.c下，有个函数需要替换一下（读SD卡方式）：

```
static uint8_t *__user_font_getdata(int offset, int size){
//如字模保存在SPI FLASH, SPIFLASH_Read(__g_font_buf,offset,size);
//如字模已加载到SDRAM,直接返回偏移地址即可如:return (uint8_t*)(sdram_fontddr+offset);
    uint32_t br;
    if( f_open(&SDFile, (const TCHAR*)"0:/myFont.bin", FA_READ) != FR_OK )
    {
        printf("myFont.bin open failed\r\n");
    }
    else
    {
        if( f_lseek(&SDFile, (FSIZE_t)offset) != FR_OK )
        {
            printf("myFont.bin lseek failed\r\n");
        }
        if( f_read(&SDFile, __g_font_buf, (UINT)size, (UINT*)&br) != FR_OK )
        {
            printf("myFont.bin lseek failed\r\n");
        }

        //    printf("offset:%d\t size:%d\t __g_font_buf:%s\r\n", offset, size, __g_font_buf);
        f_close(&SDFile);
    }
    return __g_font_buf;
```

```
24    }
```

调用示例：

```
1   static lv_style_t date_style;
2   lv_style_init(&date_style);
3   lv_obj_t* date_label2 = lv_label_create(lv_scr_act(), NULL);
4   lv_label_set_text(date_label2, "123y呀");
5   lv_style_set_text_color(&date_style, LV_STATE_DEFAULT, LV_COLOR_RED);
6   lv_style_set_text_font(&date_style, LV_STATE_DEFAULT, &myFont);
7   lv_obj_add_style(date_label2, LV_OBJ_PART_MAIN, &date_style);
8   lv_obj_align(date_label2, NULL, LV_ALIGN_IN_TOP_LEFT, 10, 10);
9   lv_obj_set_pos(date_label2, 10, 10);
```

待补充...