



Freertos 移植

基于hal库和stm32f407

lts长期支持版本

1.主要的4个：

1. 7个.c 文件
2. 硬件相关的portable文件 及内存管理memmang文件
3. 内核裁剪 配置文件 FreeRTOSconfig.h
4. 头文件 include

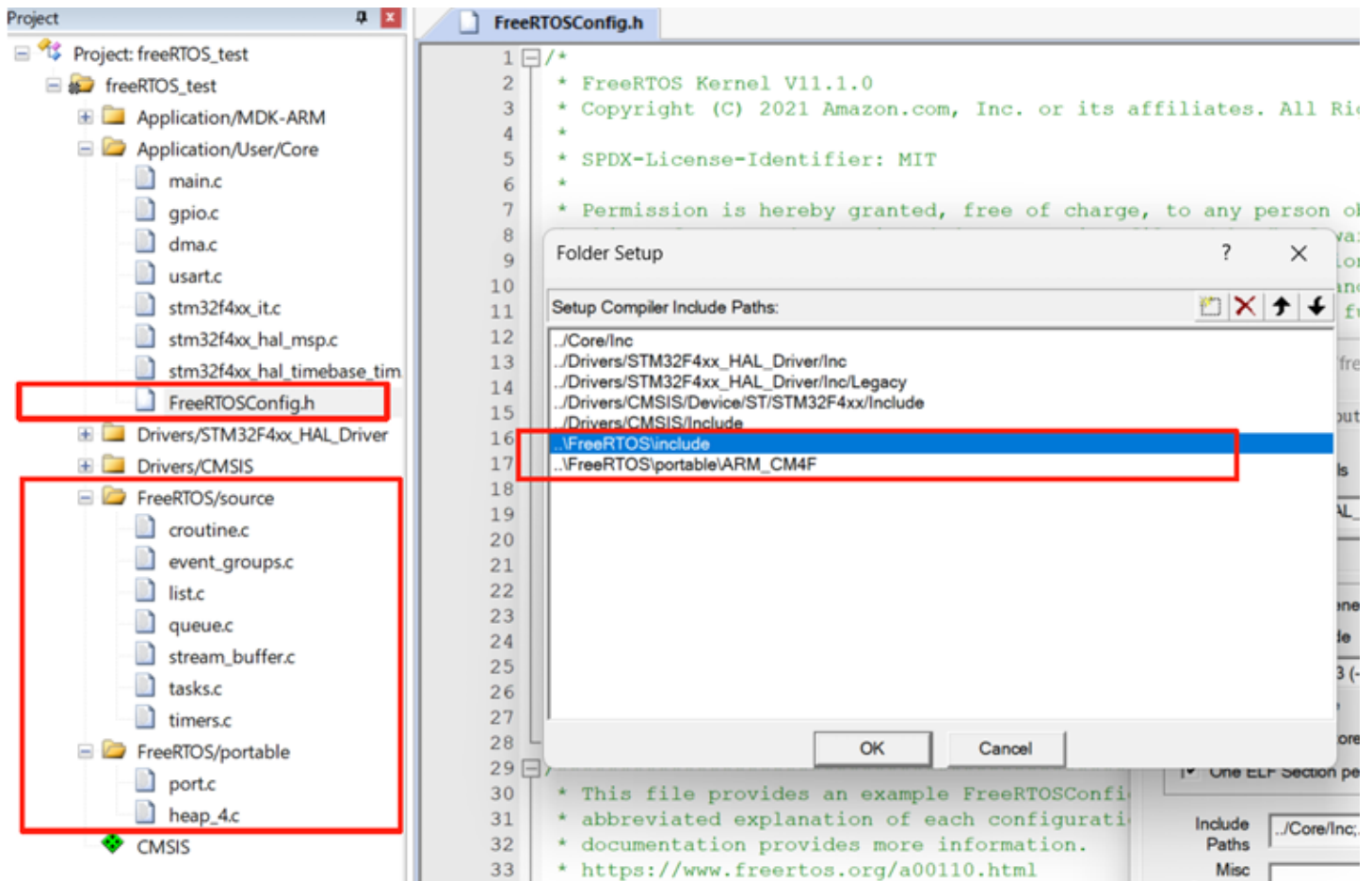
2.新建及复制

生成的cubemx 工程下 新建一个FreeRTOS 文件夹 里面新建两个文件夹

Portable **source**

1. 复制FreeRTOS-Kernel 文件夹中的7个.c文件到新建的source文件夹中
2. 复制RVDS文件夹中的ARM-CM4F文件夹以及上一级目录中的MemMang文件夹（MemMang只保留其中的额heap_4.c ,其余删掉）到新的portable文件夹中
3. 将 FreeRTOS-Kernel 文件夹中的include文件夹整个拷贝到新建的FreeRTOS文 件夹下
4. 将下面的模板FreeRTOSConfig文件拷贝到cubemx生成的user/core/Inc文件夹下（放到哪里都可以但是要方便查看）

5. 接着添加keil相关工程组



6. 修改FreeRTOSConfig.h 等配置文件。

- 修改CPU时钟频率（各芯片不同）
- 滴答计数器改成32位 TICK_TYPE_WIDTH_32_BITS
- 系统调用的中断最高优先级设置为5
- 禁用任务栈溢出的钩子函数
- 将系统tick的频率改为1000，即一秒1000个tick，也就是说时间片是1ms
- 最高优先级改为32，因为STM32支持0-31的优先级
- 然后需要改以下三项；
 - configKERNEL_INTERRUPT_PRIORITY 即 systick 优先级需要改为 最低，改为240(具体依赖cubemx 中的中断优先级分组设置，默认是Group 4)；
 - 然后其余2项保持相同。而后面2项(这2项作用是相同的，均是freeRTOS中断 能管理的最高优先级)的优先级稍比systick中断优先级高就行。
 - 不要把优先级和优先级数值搞混了，中间有一个数值转换，优先级数值计算的公式： 优先级数值=(抢占优先级<< 4)+(子优先级)，而FreeRTOSConfig.h中下面3项 就是需要优先级数值，而不是优先级。

- iv. 由于我们在cubemx中设置的是中断优先级分组group 4即4位全用于抢占优先级，0位用于子优先级，优先级最高是0 最低15。
- v. 240：15 左移4位+子优先级 = (1111 << 4) + 0000 = 11110000 直接写成(15<<4) 也可以
- vi. 192：12 左移4位+子优先级 = (1100 << 4) + 0000 = 11000000 直接写成(12<<4) 也可以，这里的意思相当于我们FreeRTOS能管理的中断范围是12-15

```
#define configKERNEL_INTERRUPT_PRIORITY          240

/* configMAX_SYSCALL_INTERRUPT_PRIORITY sets the interrupt priority
 * FreeRTOS API calls must not be made. Interrupts above this priority
 * disabled, so never delayed by RTOS activity. The default is the
 * highest interrupt priority (0). Not supported by all Cortex-M
 * See https://www.freertos.org/RTOS-Cortex-M3-M4.html
 * ARM Cortex-M devices. */
#define configMAX_SYSCALL_INTERRUPT_PRIORITY    192

/* Another name for configMAX_SYSCALL_INTERRUPT_PRIORITY, used in
 * on the FreeRTOS port. */
#define configMAX_API_CALL_INTERRUPT_PRIORITY  192
```



总结建议

场景	推荐值	说明
NVIC 分组为 Group 4 (4位抢占)	240 (0xF0)	最常用，安全
NVIC 分组为 Group 3 (3位抢占)	224 (0xE0)	保持较低优先级
NVIC 分组为 Group 2 (2位抢占)	192 (0xC0)	适用于中断较多的系统
NVIC 分组为 Group 1 (1位抢占)	128 (0x80)	很少使用，容易冲突
NVIC 分组为 Group 0 (0位抢占)	✗ 不推荐	所有中断平级，无法嵌套

添加必要的3个宏：

```
/******添加必要的3个宏******/  
#define xPortPendSVHandler      PendSV_Handler  
#define vPortSVCHandler         SVC_Handler  
#define INCLUDE_xTaskGetSchedulerState    1
```

接着将stm32f4xx.it.c 中的 SVC 和PendSV的ISR注释掉

```
// void SVC_Handler(void)
// {
//     /* USER CODE BEGIN SVCcall_IRQn 0 */

//     /* USER CODE END SVCcall_IRQn 0 */
//     /* USER CODE BEGIN SVCcall_IRQn 1 */

//     /* USER CODE END SVCcall_IRQn 1 */
// }
```

```
/**
 * @brief This function handles Debug monitor.
 */
```



```
void DebugMon_Handler(void)
```

```
{
```

```
/* USER CODE BEGIN DebugMonitor_IRQn 0 */
```

```
/* USER CODE END DebugMonitor_IRQn 0 */
```

```
/* USER CODE BEGIN DebugMonitor_IRQn 1 */
```

```
/* USER CODE END DebugMonitor_IRQn 1 */
```

```
}
```

```
/**
```

```
 * @brief This function handles Pendable request for system service.
```

```
 */
```

```
// void PendSV_Handler(void)
```

```
// {
```

```
//     /* USER CODE BEGIN PendSV_IRQn 0 */
```

```
//     /* USER CODE END PendSV_IRQn 0 */
```

```
//     /* USER CODE BEGIN PendSV_IRQn 1 */
```

```
//     /* USER CODE END PendSV_IRQn 1 */
```

```
// }
```

在stm32f4xx.it.c 中添加FreeRTOS 和 task头文件，注意FreeRTOS.h一定要在task.h的前面，不然编译会报错

```

#include "main.h"
#include "stm32f4xx_it.h"
/* Private includes -----
/* USER CODE BEGIN Includes */
#include "FreeRTOS.h"
#include "task.h"
/* USER CODE END Includes */

```

还是在这个文件，改SysTick的ISR函数，替换成FreeRTOS的滴答ISR

```

/* Private function prototypes -----
/* USER CODE BEGIN PFP */
extern void xPortSysTickHandler( void );
/* USER CODE END PFP */

```

```

void SysTick_Handler(void)
{
    /* USER CODE BEGIN SysTick_IRQn 0 */
    if(xTaskGetSchedulerState() != taskSCHEDULER_NOT_STARTED){
        xPortSysTickHandler();//替换成FreeRTOS的滴答ISR
    }
    /* USER CODE END SysTick_IRQn 0 */

    /* USER CODE BEGIN SysTick_IRQn 1 */

    /* USER CODE END SysTick_IRQn 1 */
}

```

最后保存以上更改，编译工程

注意：HAL 库本身和FreeRTOS 都默认依赖 systick(且 freeRTOS 默认把 systick 和PendSV 的 ISR 优先级改为最低以防止影响其他中断),可能出现卡死的问题。为了保险起见，可以考虑在sys选择HAL时钟源的时候换成其他的定时器，并且中断优先级设为较高，比如1。这里我选择systick的时基是TIM14，中断优先级改为1。然后systick和PendSV优先级改为最低的15。

Non maskable interrupt	<input checked="" type="checkbox"/>	0	0
Hard fault interrupt	<input checked="" type="checkbox"/>	0	0
Memory management fault	<input checked="" type="checkbox"/>	0	0
Pre-fetch fault, memory access fault	<input checked="" type="checkbox"/>	0	0
Undefined instruction or illegal state	<input checked="" type="checkbox"/>	0	0
System service call via SWI instruction	<input checked="" type="checkbox"/>	0	0
Debug monitor	<input checked="" type="checkbox"/>	0	0
Pendable request for system service	<input checked="" type="checkbox"/>	15	0
System tick timer	<input checked="" type="checkbox"/>	15	0
PVD interrupt through EXTI line 16	<input type="checkbox"/>	0	0
Flash global interrupt	<input type="checkbox"/>	0	0
RCC global interrupt	<input type="checkbox"/>	0	0
USART1 global interrupt	<input checked="" type="checkbox"/>	0	0
Time base: TIM8 trigger and commut...	<input checked="" type="checkbox"/>	1	0
DMA2 stream2 global interrupt	<input checked="" type="checkbox"/>	0	0
FPU global interrupt	<input type="checkbox"/>	0	0

用于freeRTOS



用于hal库时钟

