

---

## TI-MSPM0G3507-SDK 工程模板文件使用教程

本项目为电赛指定开发板 TI-MSPM0G3507 工程模板文件。

主要功能是：使用 KIEL5+VSCODE 配合嵌入式操作系统（FreeRTOS/RT-Thread）  
优雅的开发 TI-MSPM0G3507 单片机

。

TI-MSPM0G3507-SDK 版本：**mspm0\_sdk\_2\_05\_00\_05**

TI-SYSCONFIG 图形配置工具版本：**sysconfig-1.24.0\_4150**

使用代码编译环境为：**KEIL5 MDK-Arm-v5.39**。（KEIL5 必须高于 v5.38a 版本才能编译 TI-M0 系列芯片）。

项目内容包括：

- （1） 一个没有使用操作系统的 TI-MSPM0G3507 工程模板文件。
- （2） 一个使用 FreeRTOS 操作系统的 TI-MSPM0G3507 工程模板文件。
- （3） 一个使用 RT-Thread 操作系统的 TI-MSPM0G3507 工程模板文件。

项目开发工具配合使用：VSCODE、KEIL5、SSCOM 串口助手、VOFA+上位机。

调试器可用包括：有线 JLink、有线 DAPLink 以及-无线 DAPLink。

其中有线 JLink 下载最稳定。

不允许使用 ST-Link，否则会锁芯片。

JLink 推荐：[STM32 下载器 Jlink OB ARM 仿真调试器 SWD 下载器-淘宝网](#)。

本项目主要目的是：舍弃裸机开发，配合操作系统开发 TI-MSPM0G3507 单片机，花了点时间为 TI-MSPM0G3507 移植并适配了 FreeRTOS/RT-Thread 操作系统。

下面是使用教程：

在使用本工程模板前，请务必学习：

- （1）开发板要求介绍：

[天猛星入门手册 | 立创开发板技术文档中心](#)

- （2）下载 TI 相关工具以及环境配置：

[天猛星入门手册 | 立创开发板技术文档中心](#)

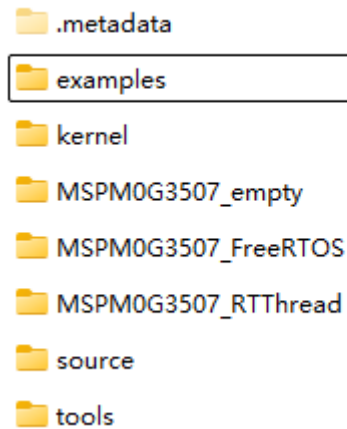
（3）学会修改 KEIL5 的源文件、头文件、用户命令、第三方库文件链接的设置：

[3. 点亮第一个灯 | 立创开发板技术文档中心](#)

---

MSPM0G3507 项目模板文件包含内容如下：

mspm0\_sdk\_2\_05\_00\_05



**Examples:** 官方例程（分为有 FreeRTOS 版本和无 FreeRTOS 版本、该实例中的 FreeRTOS 版本无法使用，问题不知，后续是本人重新移植版本）

**Kernel:** TI-M0 内核驱动库

**Source:** 第三方库（包括 DSP）、TI 库文件（包括各个外设的源代码、实现类似 HAL 库）

**Tools:** 一些 IAR、KIEL 的工具

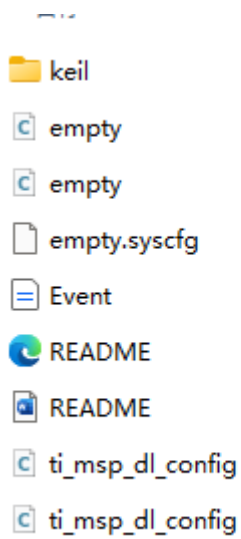
其中 Examples 可以删除、Kernel 和 Source 不允许删。

- （1）MSPM0G3507\_empty：一个没有使用操作系统的 TI-MSPM0G3507 工程模板文件。
- （2）MSPM0G3507\_FreeRTOS：一个使用 FreeRTOS 操作系统的 TI-MSPM0G3507 工程模板文件。
- （3）MSPM0G3507\_RT-Thread：一个使用 RT-Thread 操作系统的 TI-MSPM0G3507 工程模板文件。

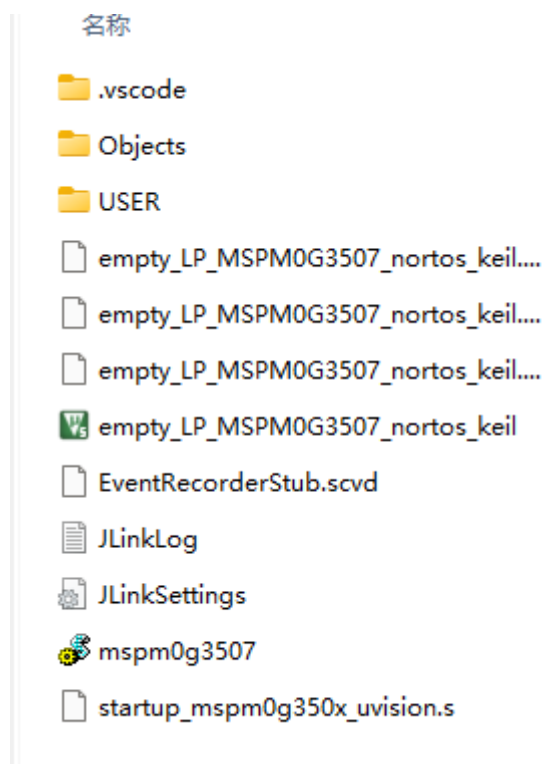
下面是各个模板文件的介绍与使用教程：

---

## MSPM0G3507\_empty

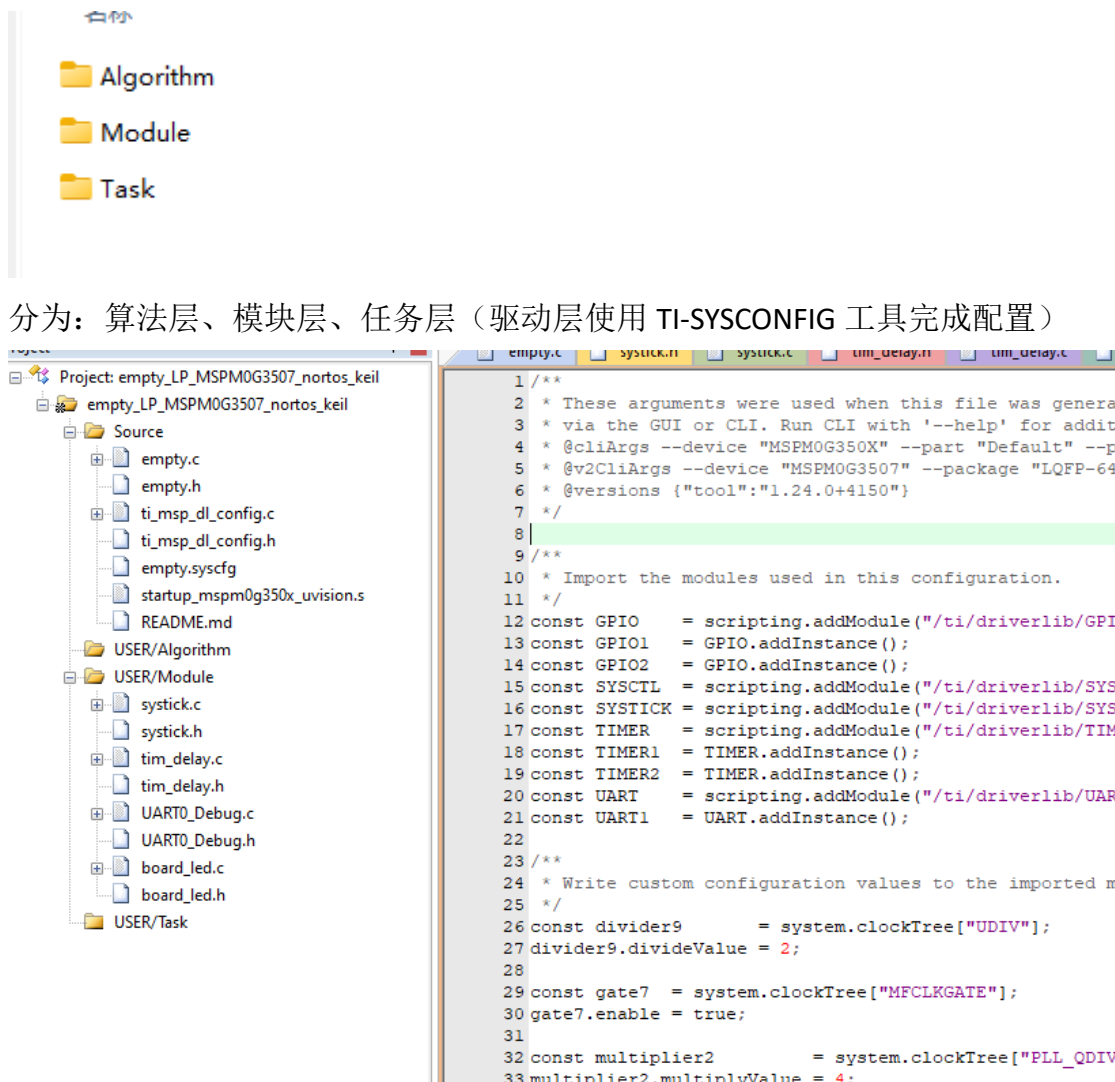


Empty: main 主函数文件



Objects: KEIL 编译产生的编译文件

USER: 用户代码放置文件



KIEL 打开项目，代码层级如上。

介绍：

- (1) 使用 systick 系统时钟节拍实现高精度延时与程序运行时间测量
- (2) 使用 TIMA0 和 TIMA1 实现两个不同的延时函数
- (3) UART0 调试串口
- (4) 天猛星板载 LED

这里需要注意！！

使用了 SYSTICK 系统时钟，就不能使用官方给的

```
#define delay_cycles(cycles) DL_Common_delayCycles(cycles)
```

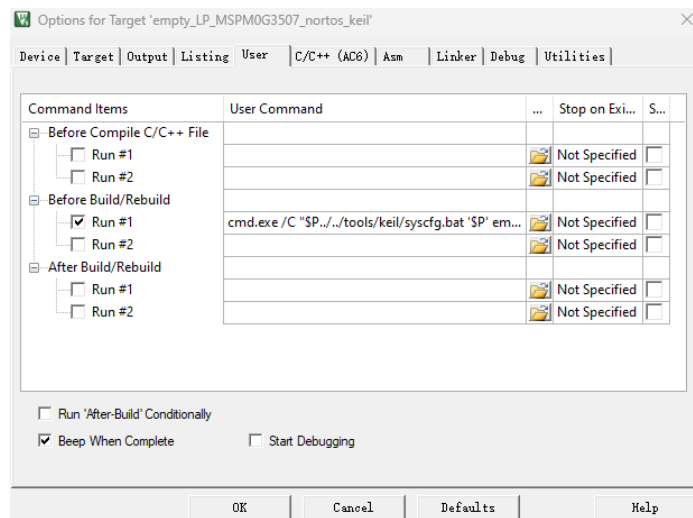
延时函数。原因如下：

查看库源码发现该函数实现，是使用汇编语言通过修改 SYSTICK 系统时钟的 R0 寄存器实现的。

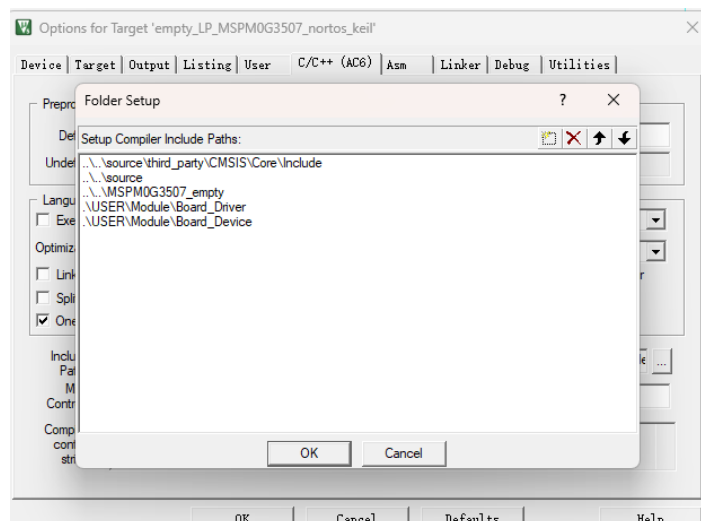
使用 SYSCONFIG 工具配置 SYSTICK 后会与该函数产生冲突。

KIEL 工程的源文件、头文件、用户命令、第三方库文件链接的设置如下：

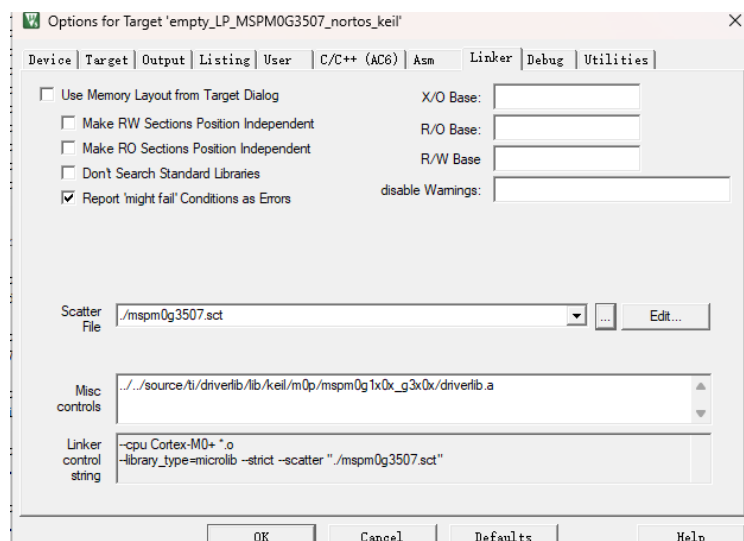
### (1) 编译前命令使用



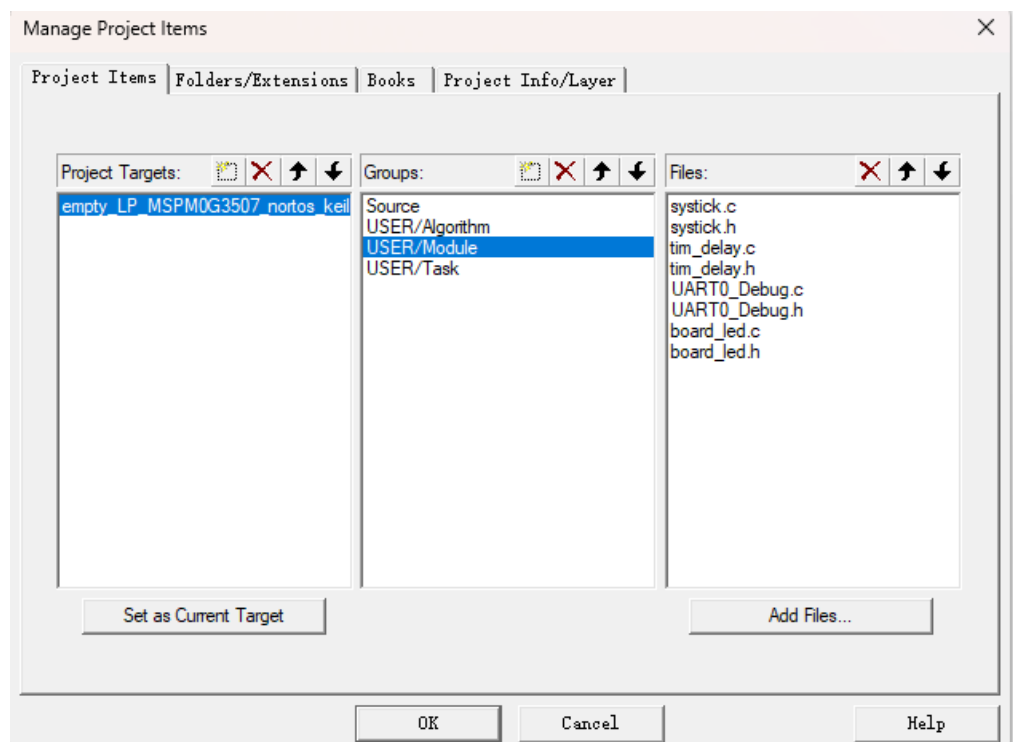
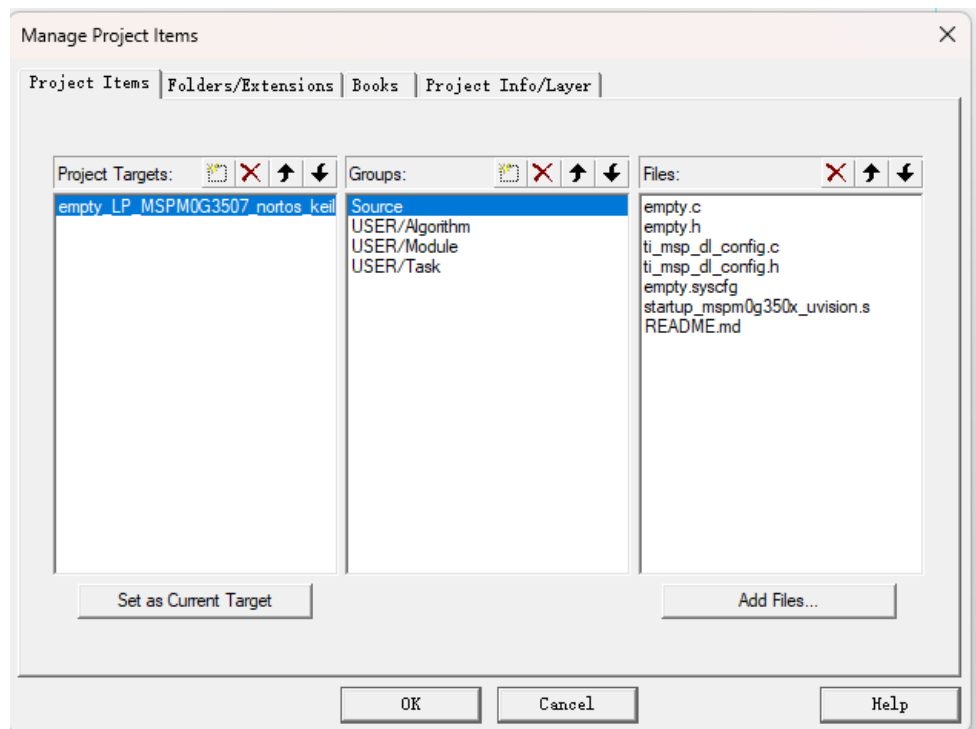
### (2) 添加头文件路径



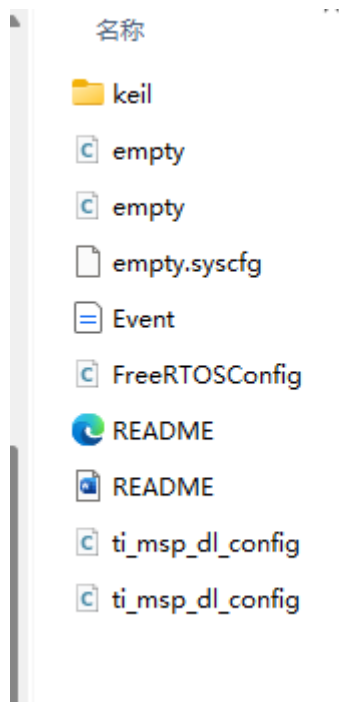
### (3) 链接第三方库



#### (4) 添加源文件路径



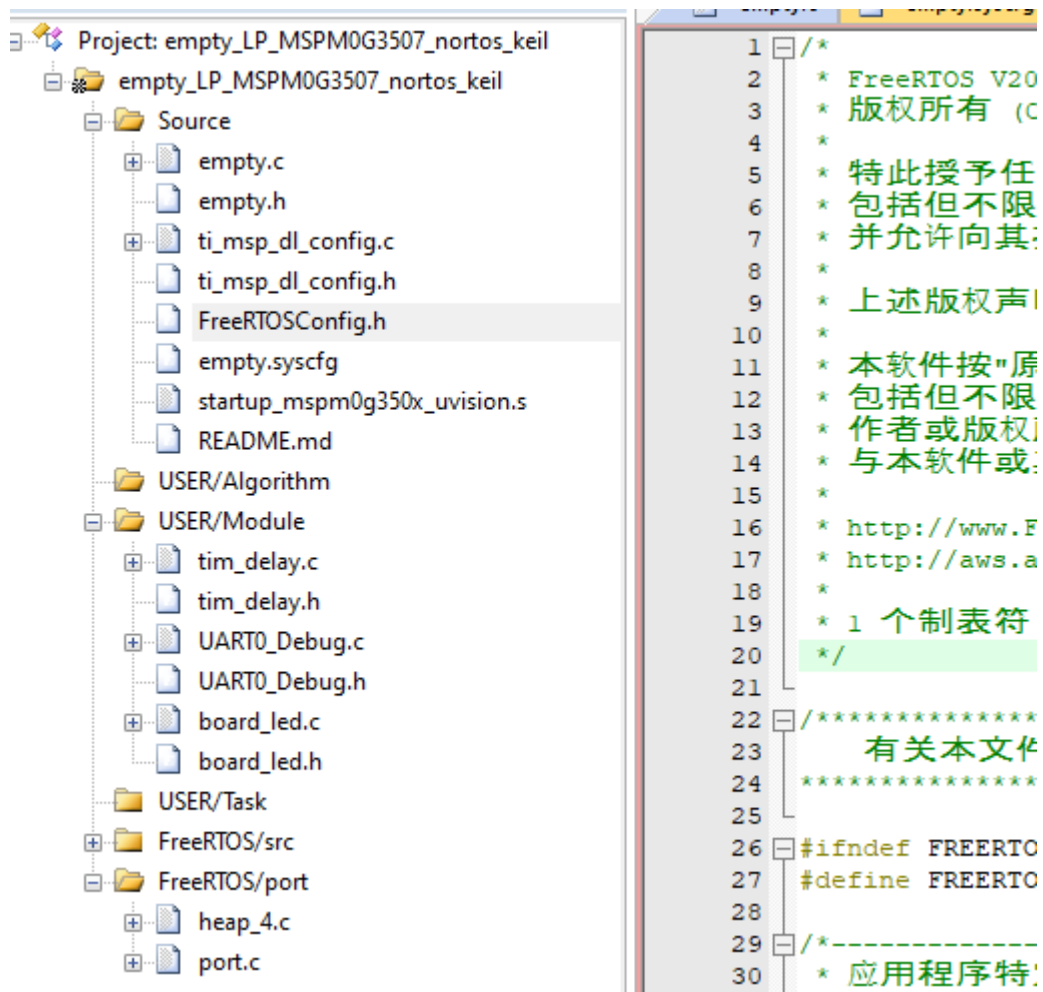
## MSPM0G3507\_FreeRTOS



多一个 FreeRTOSConfig 的操作系统配置文件。

名称	
.vscode	2
FreeRTOS	2
Objects	2
USER	2
empty_LP_MSPM0G3507_nortos_keil....	2
empty_LP_MSPM0G3507_nortos_keil....	2
empty_LP_MSPM0G3507_nortos_keil....	2
empty_LP_MSPM0G3507_nortos_keil	2
EventRecorderStub.scvd	2
JLinkLog	2
JLinkSettings	2
mspm0g3507	2
startup_mspm0g350x_uvision.s	2

多一个 FreeRTOS 操作系统内核源码。与官方实例配置不同，官方例子总有报错。



KIEL 打开项目，代码层级如上。

介绍：

- (5) 使用 **systick** 系统时钟作为 **FreeRTOS** 操作系统的时钟心跳
- (6) 使用 **TIMA0** 和 **TIMA1** 实现两个不同的延时函数
- (7) **UART0** 调试串口
- (8) 天猛星板载 LED

这里需要注意！！

使用了 **SYSTICK** 系统时钟，就不能使用官方给的

```
#define delay_cycles(cycles) DL_Common_delayCycles(cycles)
```

延时函数。

！！不允许再任何地方配置 **SYSTICK** 系统时钟有关的东西！！

因为该时钟在 **FreeRTOS** 中被使用。

然后依旧是设置 **KIEL** 工程的源文件、头文件、用户命令、第三方库文件链接，同上。



## MSPM0G3507\_RT-Thread

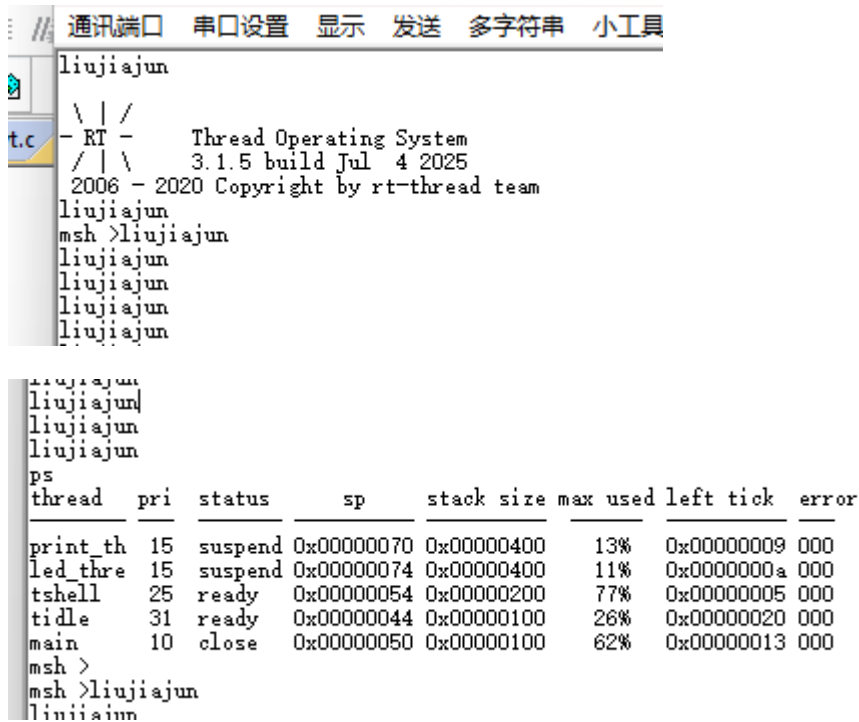
**!! 这里请注意!!**

使用 RT-Thread 操作系统，需要安装官方 KIEL 支持包，我们只使用 RTTHREAD 的内核文件，不使用 RTTHREAD 的设备驱动框架文件，下载安装教程如下：

### (1) 使用 MDK 移植

**!! 还需要注意!!**

(2) 在本项目中配置了 RTTHREAD-NANO 系统，并且移植设置了 RTT 的控制台以及 FinSH 命令行工具，实现效果如下：



```

// 通讯端口 串口设置 显示 发送 多字符串 小工具
liujiajun
- RT -      Thread Operating System
/ | \      3.1.5 build Jul  4 2025
2006 - 2020 Copyright by rt-thread team
liujiajun
msh >liujiajun
liujiajun
liujiajun
liujiajun
liujiajun
liujiajun

liujiajun
liujiajun|
liujiajun
liujiajun
ps
thread pri status sp stack size max used left tick error
print_th 15 suspend 0x00000070 0x00000400 13% 0x00000009 000
led_thre 15 suspend 0x00000074 0x00000400 11% 0x0000000a 000
tshell 25 ready 0x00000054 0x00000200 77% 0x00000005 000
tidle 31 ready 0x00000044 0x00000100 26% 0x00000020 000
main 10 close 0x00000050 0x00000100 62% 0x00000013 000
msh >
msh >liujiajun
liujiajun
```

使用 ps 命令查看各个线程状态。

**!! 还需要注意!!**

### (3) KIEL 编译问题。

与之前的配置不同，RT-Thread 在 KIEL 中必须设置-O2 级别的优化如下：

具体原因未知。

Options for Target 'empty\_LP\_MSPM0G3507\_nortos\_keil'

Device | Target | Output | Listing | User | C/C++ (AC6) | Asm | Linker | Debug | Utilities

Preprocessor Symbols

Define:

Undefine:

Language / Code Generation

☐ Execute-only Code      Warnings:       Language C:

Optimization:       ☐ Turn Warnings into Errors      Language C++:

☐ Link-Time Optimization      ☐ Plain Char is Signed      ☒ Short enums/wchar

☐ Split Load and Store Multiple      ☐ Read-Only Position Independent      ☐ use RTTI

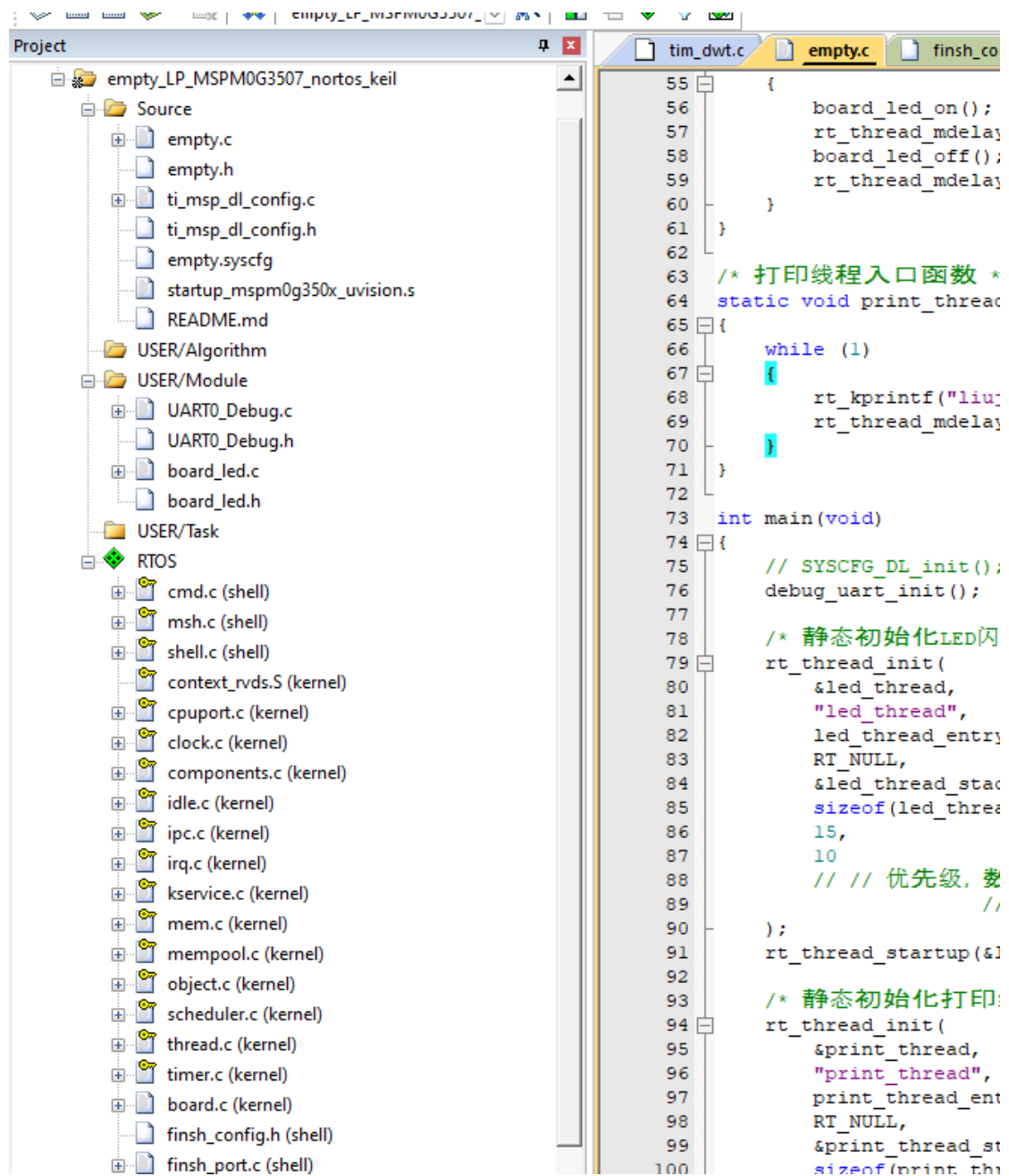
☒ One ELF Section per Function      ☐ Read-Write Position Independent      ☐ No Auto Includes

Include Paths:  ...

Misc Controls:

Compiler control string:

OK      Cancel      Defaults      Help



KIEL 打开项目，代码层级如上。

介绍：

- (9) 使用 **sys tick** 系统时钟作为 **RT-Thread** 操作系统的时钟心跳
- (10) 使用 **TIMAO** 和 **TIMA1** 实现两个不同的延时函数
- (11) **UART0** 调试串口
- (12) 天猛星板载 LED

这里需要注意!!

使用了 **SYSTICK** 系统时钟，就不能使用官方给的

```
#define delay_cycles(cycles) DL_Common_delayCycles(cycles)
```

---

延时函数。

!! 不允许再任何地方配置 **SYSTICK** 系统时钟有关的东西!!  
因为该时钟在 **FreeRTOS** 中被使用。

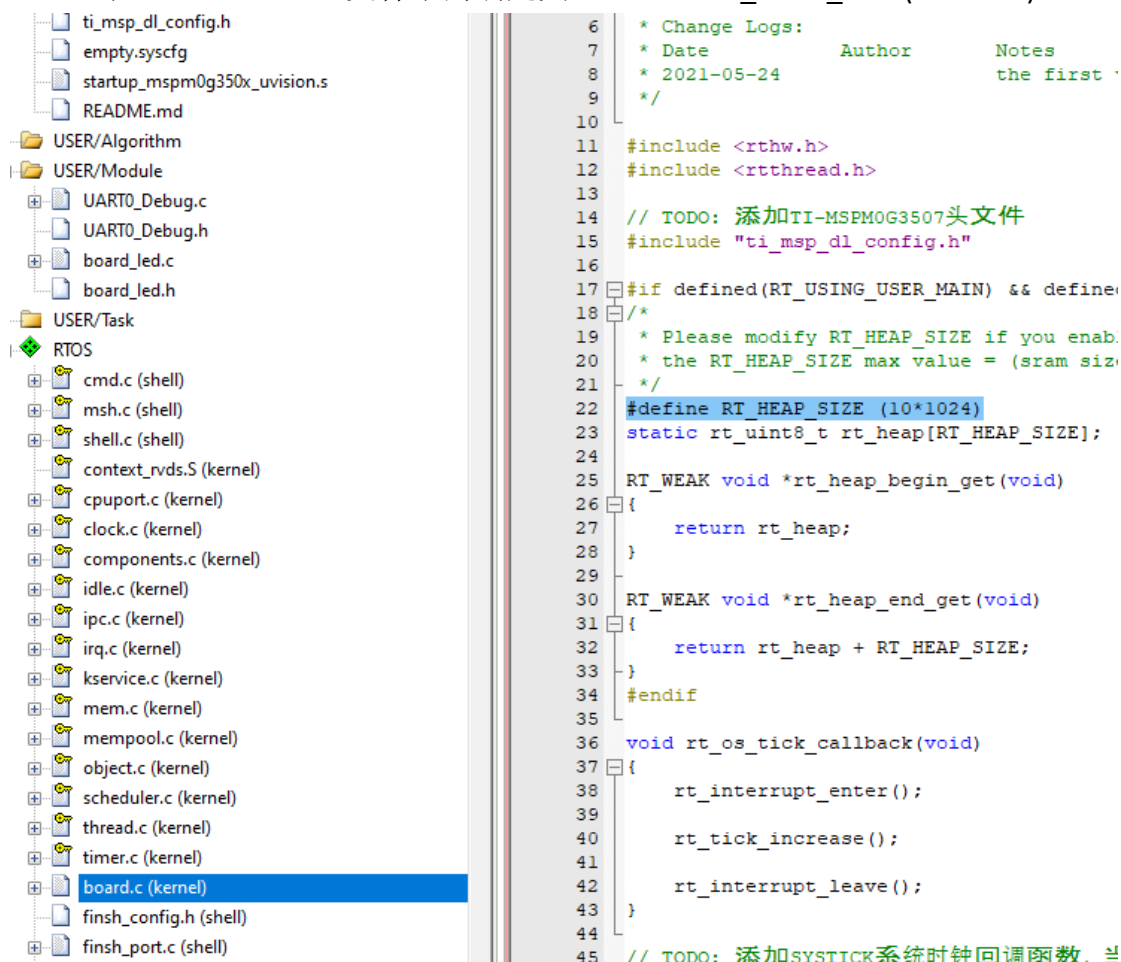
然后依旧是设置 **KIEL** 工程的源文件、头文件、用户命令、第三方库文件链接，同上。

必须注意，RTThread 这里的 **SYSTICK** 是通过 **TI-SYSTICK** 配置工具实现，与 **FreeRTOS** 手写寄存器配置实现不同。

方式类似使用 **STM32CUBEMAX** 对 **RTThread** 配置系统时钟心跳。

## 动态线程内存堆栈配置

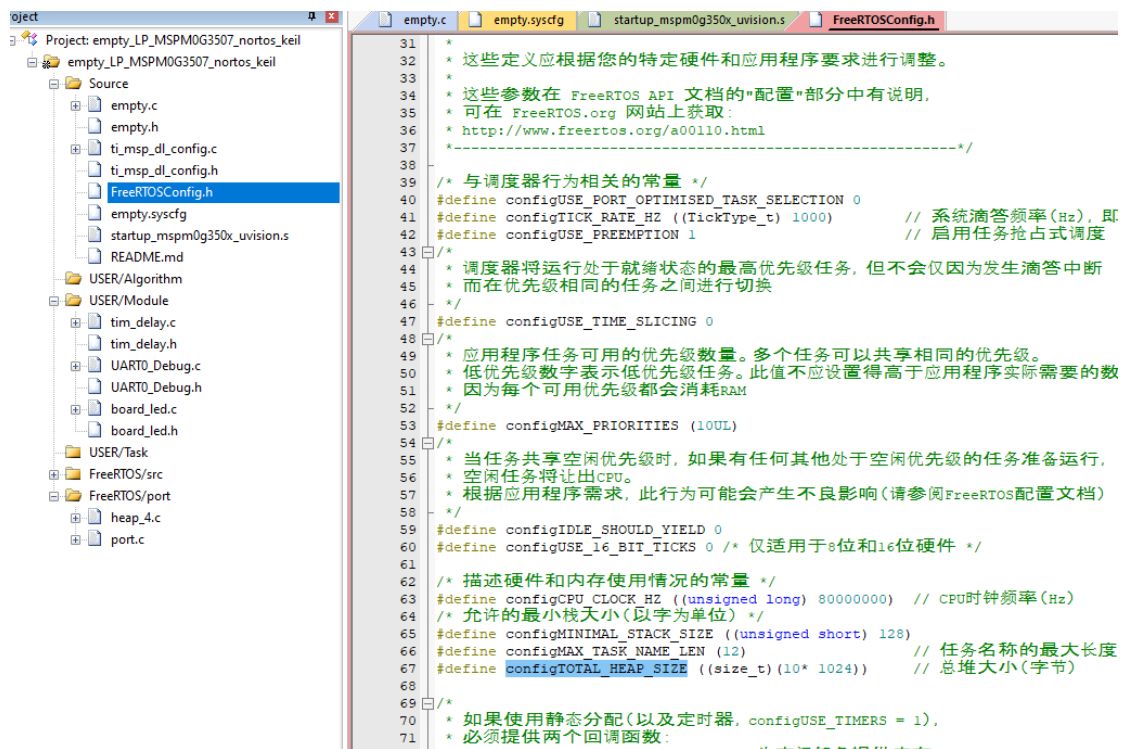
- (1) RTThread 修改动态创建线程任务的总堆栈内存 (RAM 占用的大小), 在 RTOS->Board.c 文件中的宏定义: `#define RT_HEAP_SIZE (10*1024)`



有什么用?

- (2) 总所周知, 使用操作系统动态创建线程任务时系统需要分配堆栈大小给线程任务, 所有动态创建的任务堆栈都是从 `#define RT_HEAP_SIZE (10*1024)` 中扣除。
- (3) 这里提前给了 10kb 的 RAM 给 RT-Thread 用于线程的动态创建。
- (4) 如果你使用静态创建线程, 就可以调小这个宏定义分配的 RAM 大小, 因为静态线程的堆栈大小不从这里扣除, 而是额外通过手动配置从另外的 RAM 区扣除。
- (5) 在 RAM 较小的单片机上, 比如 M0G3507 只有 32kb, 推荐使用静态创建线程, 手动管理内存大小。

FreeRTOS 同理，动态线程堆栈大小修改宏定义如下：



```
31 *
32 * 这些定义应根据您的特定硬件和应用程序要求进行调整。
33 *
34 * 这些参数在 FreeRTOS API 文档的"配置"部分中有说明,
35 * 可在 FreeRTOS.org 网站上获取:
36 * http://www.freertos.org/a00110.html
37 * -----*/
38
39 /* 与调度器行为相关的常量 */
40 #define configUSE_PORT_OPTIMISED_TASK_SELECTION 0
41 #define configTICK_RATE_HZ ((TickType_t) 1000) // 系统滴答频率(Hz), 即
42 #define configUSE_PREEMPTION 1 // 启用任务抢占式调度
43
44 /*
45 * 调度器将运行处于就绪状态的最高优先级任务, 但不会仅因为发生滴答中断
46 * 而在优先级相同的任务之间进行切换
47 */
48 #define configUSE_TIME_SLICING 0
49
50 /*
51 * 应用程序任务可用的优先级数量。多个任务可以共享相同的优先级。
52 * 低优先级数字表示低优先级任务。此值不应设置得高于应用程序实际需要的数
53 * 因为每个可用优先级都会消耗RAM
54 */
55 #define configMAX_PRIORITIES (10UL)
56
57 /*
58 * 当任务共享空闲优先级时, 如果有任何其他处于空闲优先级的任务准备运行,
59 * 空闲任务将让出CPU。
60 * 根据应用程序需求, 此行为可能会产生不良影响(请参阅FreeRTOS配置文档)
61 */
62 #define configIDLE_SHOULD_YIELD 0
63 #define configUSE_16_BIT_TICKS 0 /* 仅适用于8位和16位硬件 */
64
65 /* 描述硬件和内存使用情况的常量 */
66 #define configCPU_CLOCK_HZ ((unsigned long) 80000000) // CPU时钟频率(Hz)
67 /* 允许的最小栈大小(以字为单位) */
68 #define configMINIMAL_STACK_SIZE ((unsigned short) 128)
69 #define configMAX_TASK_NAME_LEN (12) // 任务名称的最大长度
70 #define configTOTAL_HEAP_SIZE ((size_t) (10* 1024)) // 总堆大小(字节)
71
72 /*
73 * 如果使用静态分配(以及定时器, configUSE_TIMERS = 1),
74 * 必须提供两个回调函数:
75 */
```