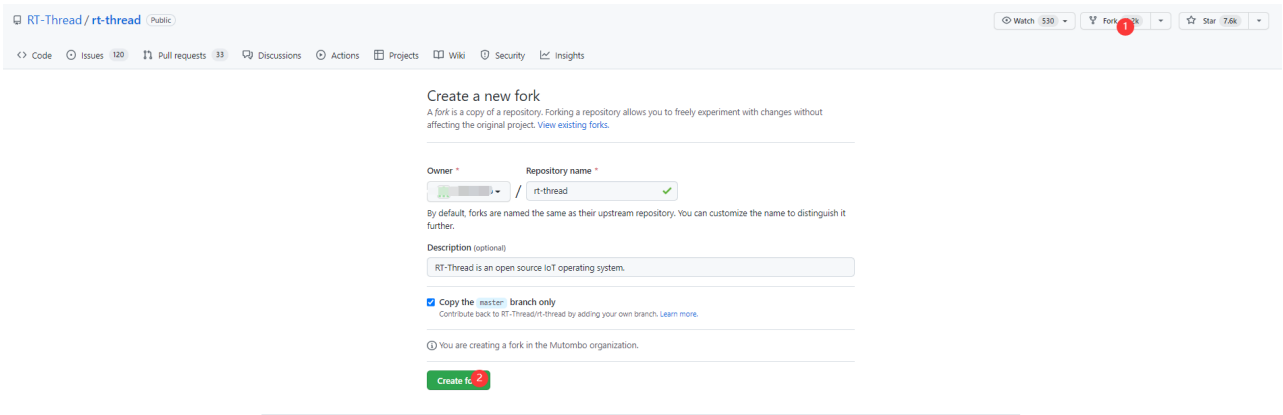
# 1. 背景

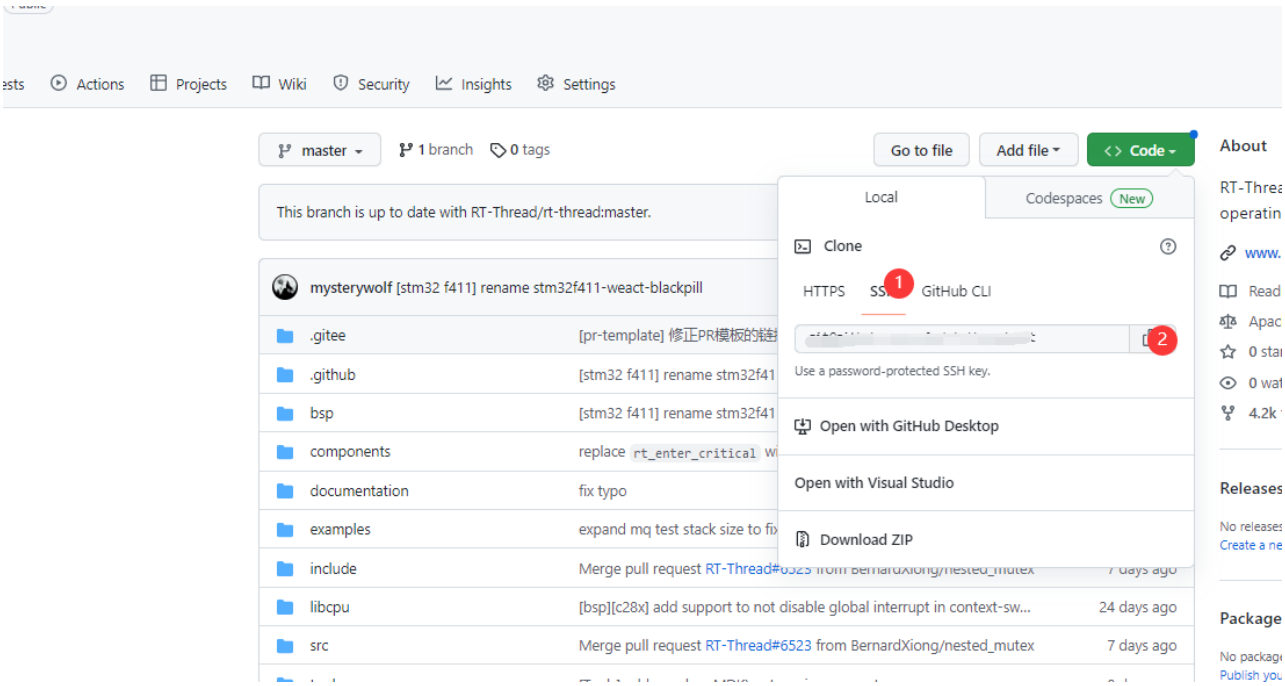
看野火瑞萨新品发布会，Andy Chen提到了瑞萨不同方案之间切换很简单，刚好手头上有一块RA2L1，且有机会拿到启明6M5,因此可以从代码的角度上看瑞萨RA系列平台移植的难易程度。

由于我近几年一直在使用RTT开发，对RTT比较熟悉，刚好使用RA2L1时是在RTT环境下配的，也就顺势想到了先适配RTT，然后逐个驱动打通看差异了。

# 2. 移植步骤

## a. rt-thread代码拉取

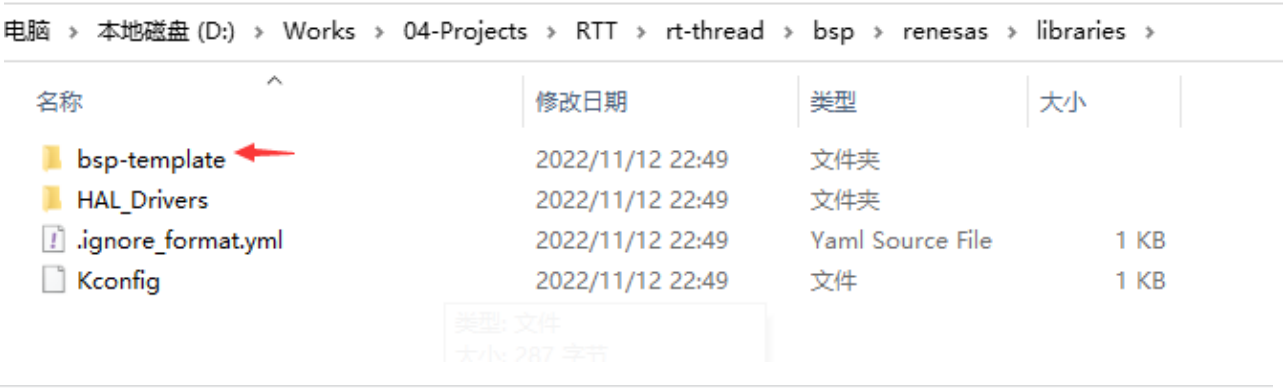


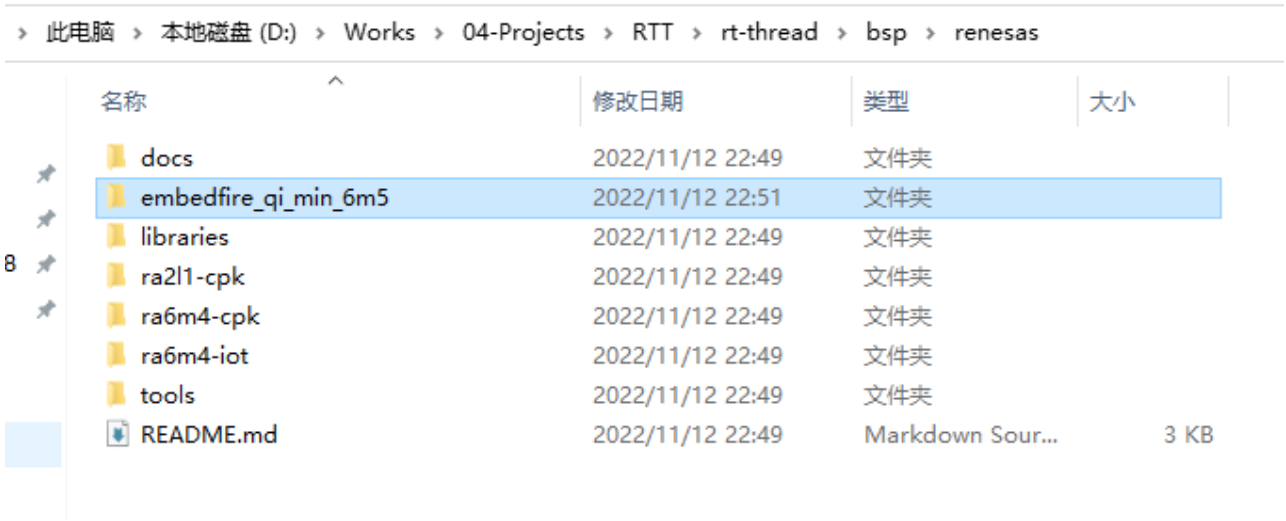




执行以上步骤后，在本地使用git命令将代码拉取下来

## b. 复制bsp模版

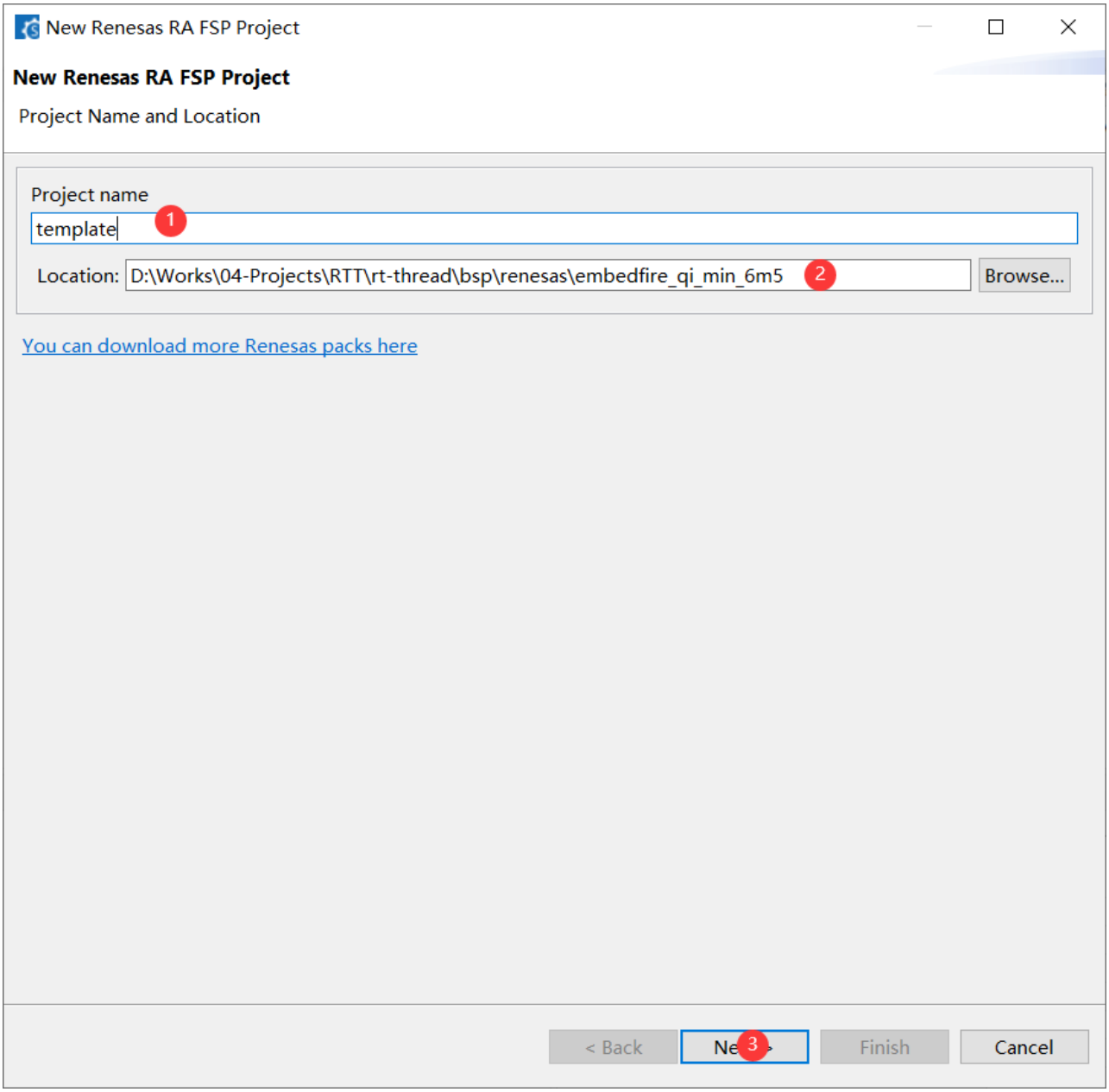
从./bsp/renesas/libraries目录下拷贝 bsp-template文件夹至./bsp/renesas/目录下，并重命名为embedfire_qi_min_6m5（实在想不出更简单的命名了）：

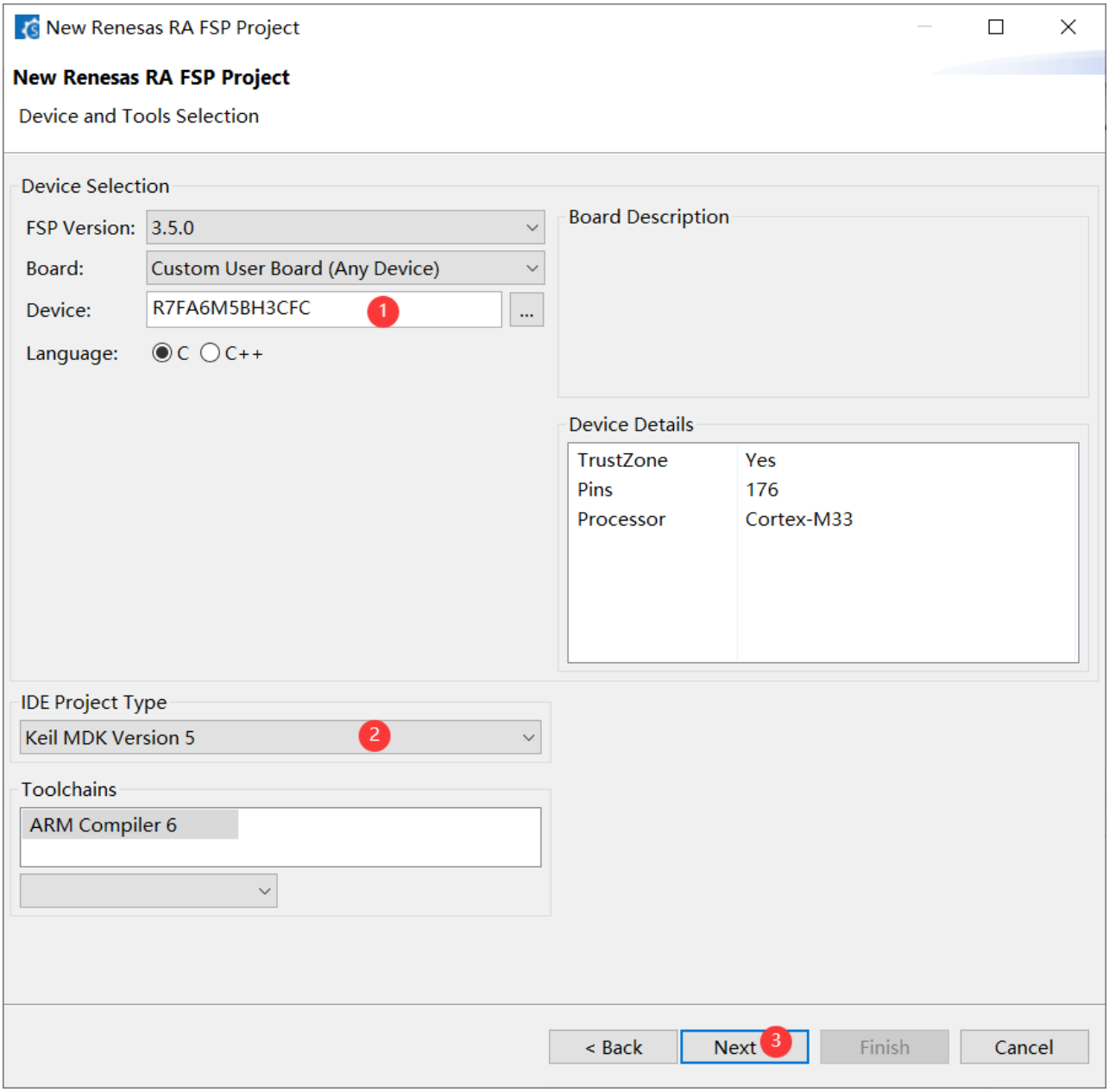


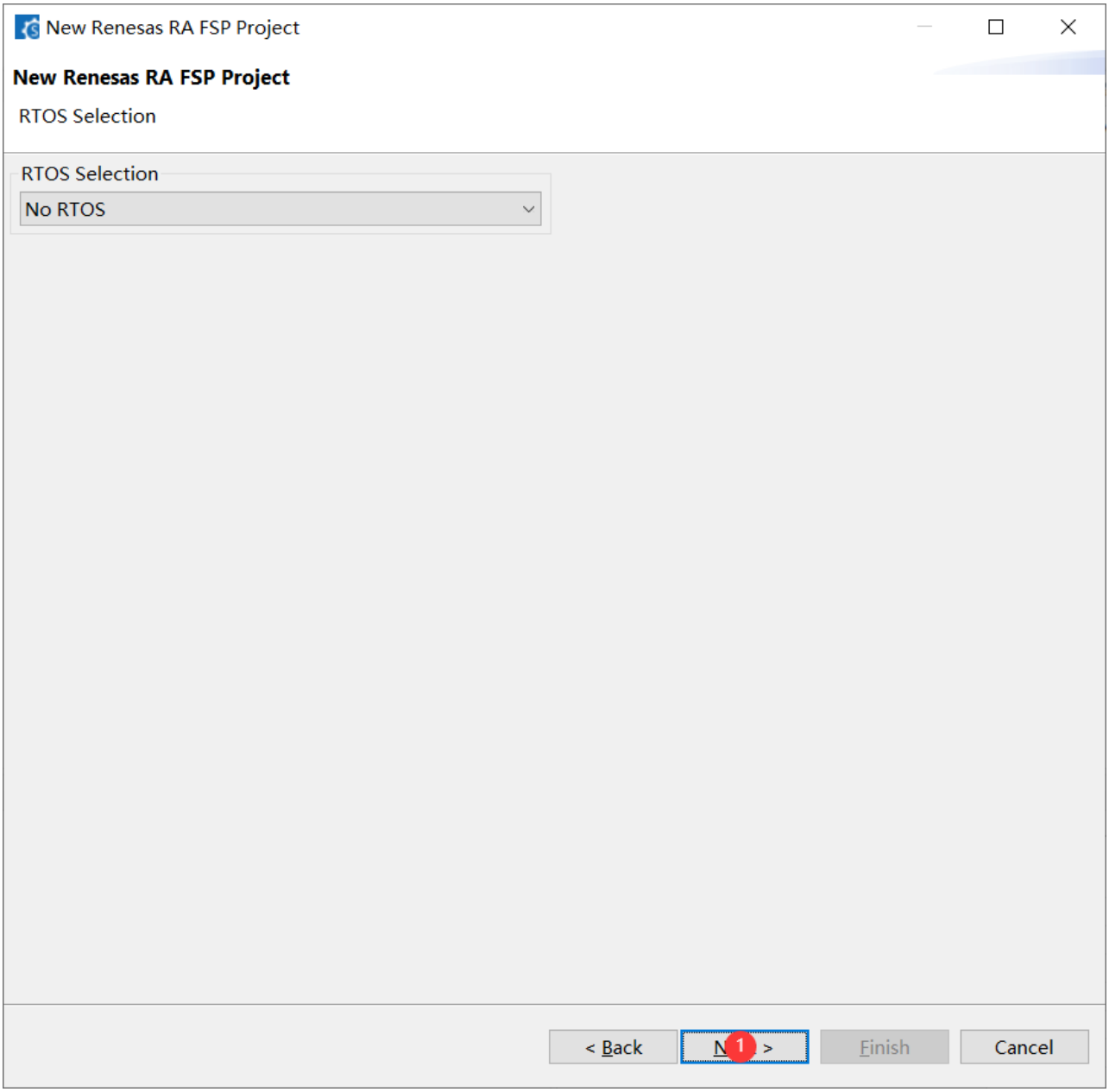



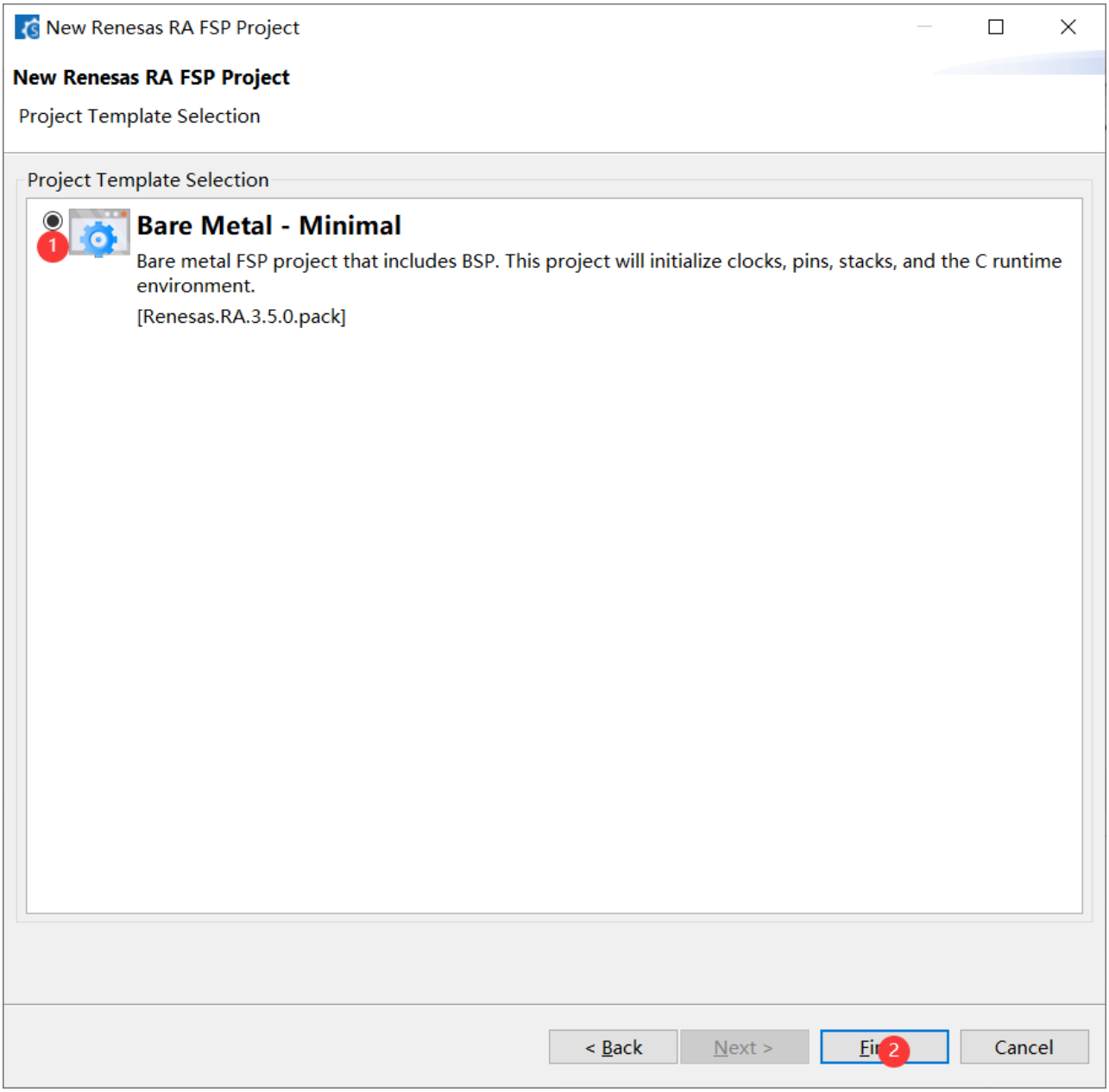

## c. 配置template工程

i. 运行rasc.exe，配置项目参数


New Renesas RA FSP Project
New Renesas RA FSP Project
Project Name and Location
Project name
template
1
Location: D:\Works\04-Projects\RTT\rt-thread\bsp\renesas\embedfire_qi_min_6m5
2
Browse...
You can download more Renesas packs here
< Back
Ne
3
Finish
Cancel

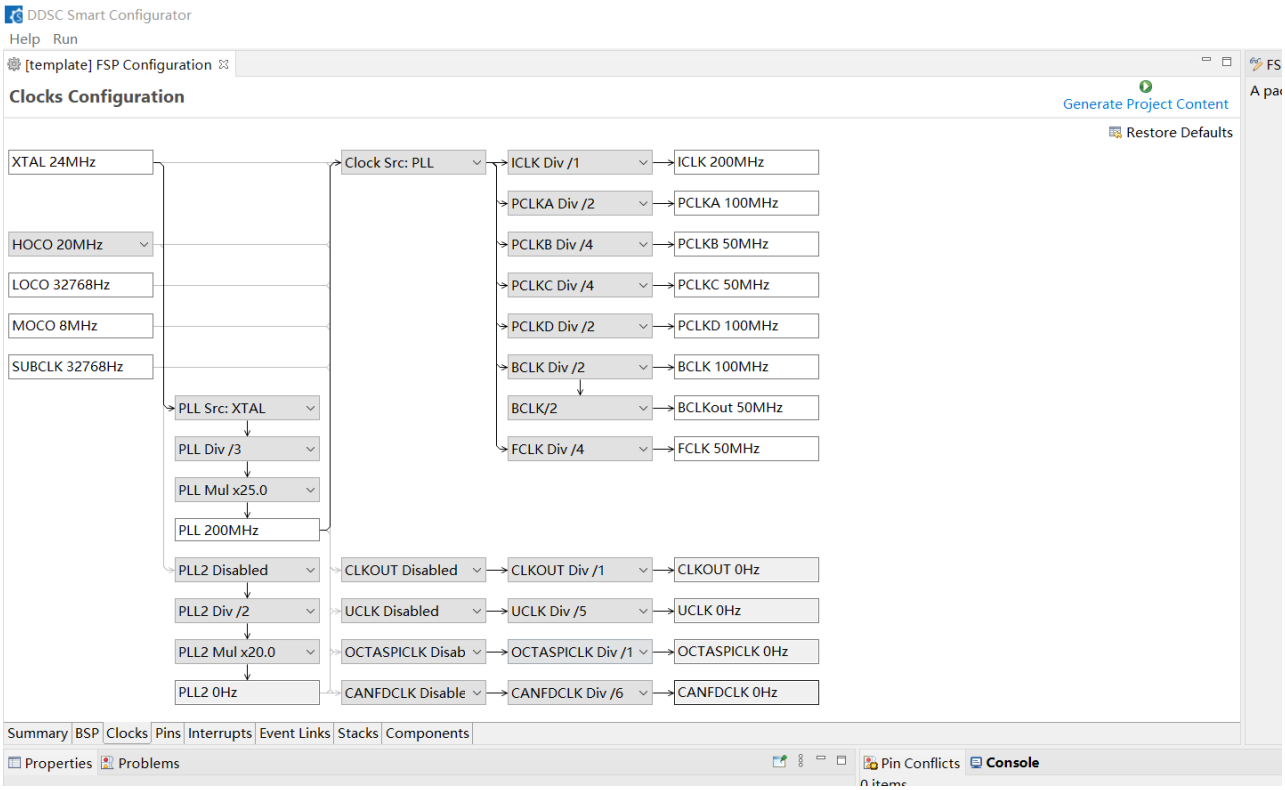

New Renesas RA FSP Project
New Renesas RA FSP Project
Device and Tools Selection
Device Selection
FSP Version: 3.5.0
Board: Custom User Board (Any Device)
Device: R7FA6M5BH3CFC
1
...
Language: C C++
Board Description
Device Details
TrustZone Yes
Pins 176
Processor Cortex-M33
IDE Project Type
Keil MDK Version 5
2
Toolchains
ARM Compiler 6
< Back
Next
3
Finish
Cancel


New Renesas RA FSP Project
New Renesas RA FSP Project
RTOS Selection
RTOS Selection
No RTOS
< Back
N 1 >
Finish
Cancel



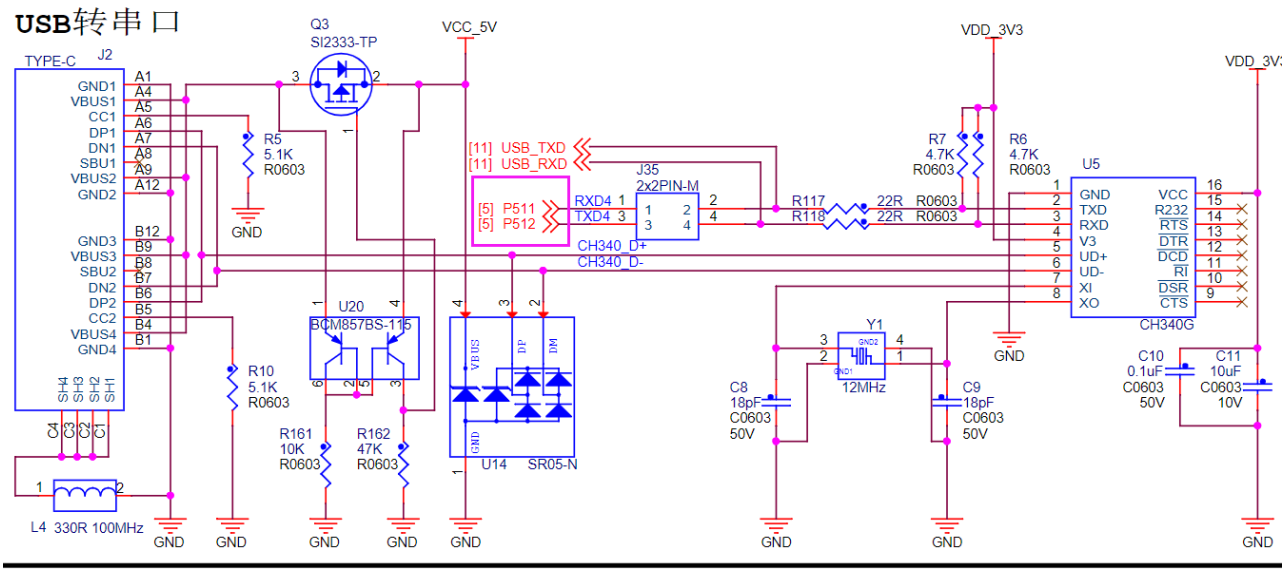

ii. 打开串口

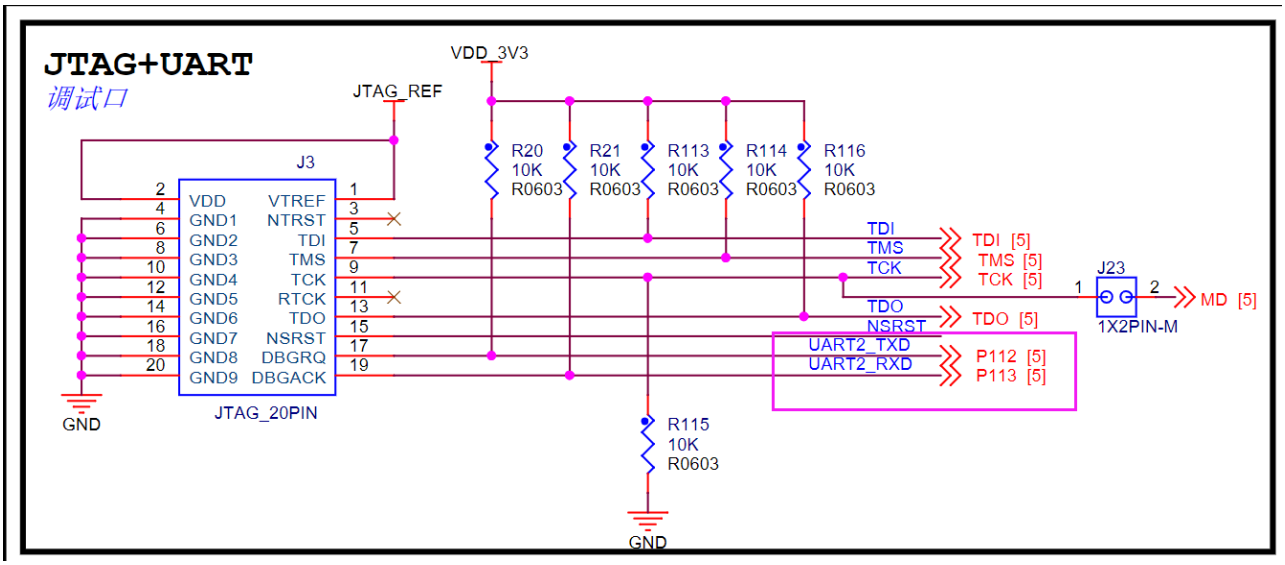
1. 确认晶振和主频（24M和200M）



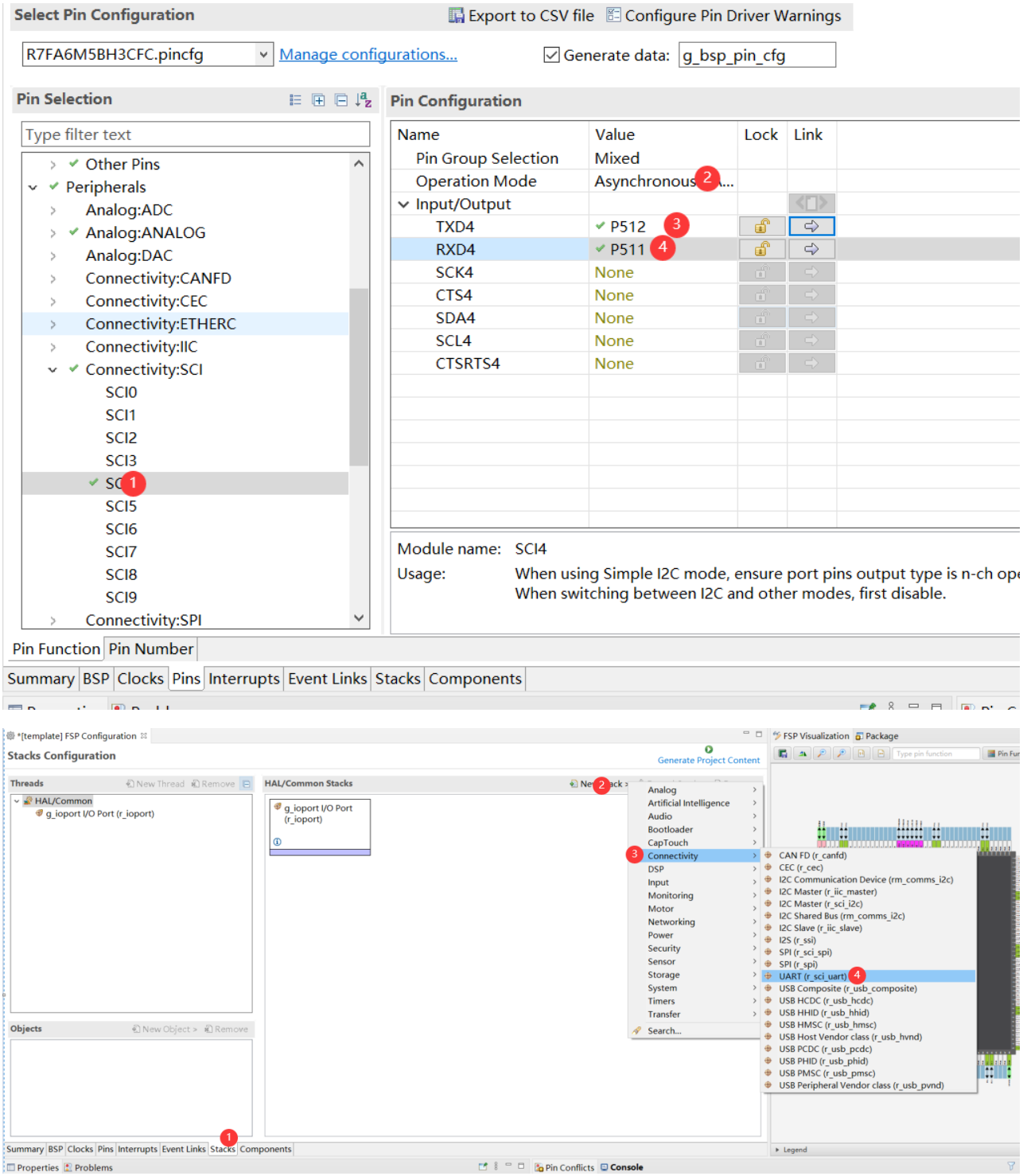

2. 打开串口配置

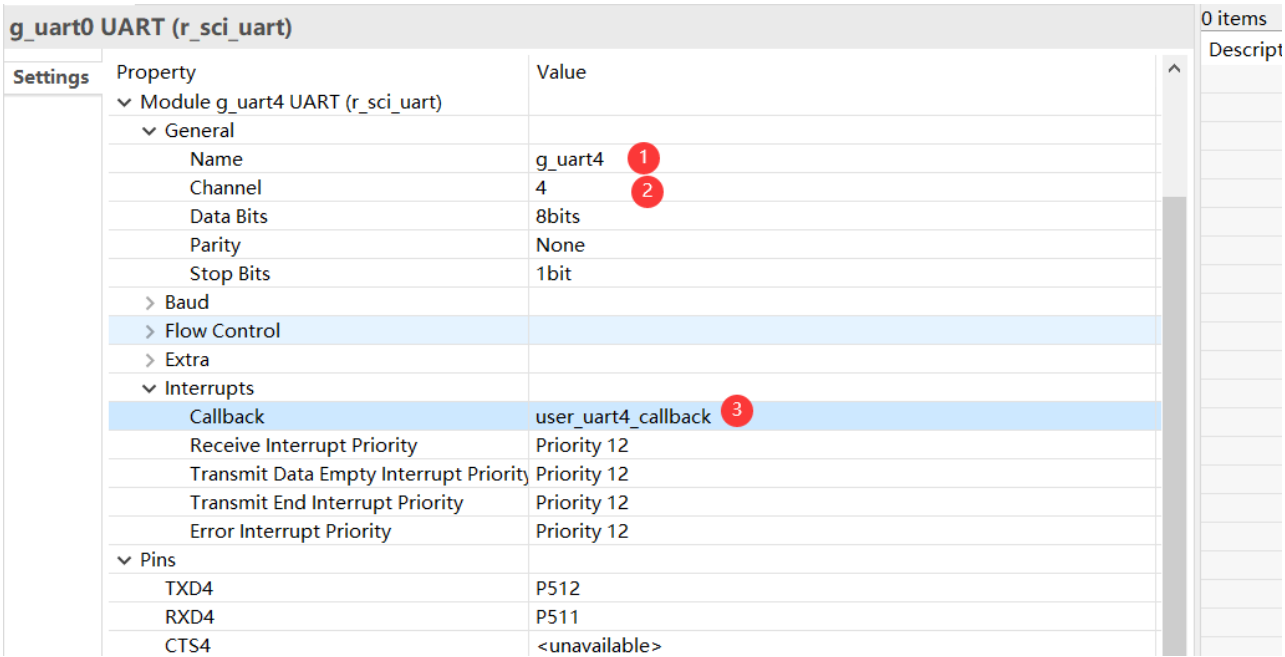
启明6M5上有两个串口引出，如下图：



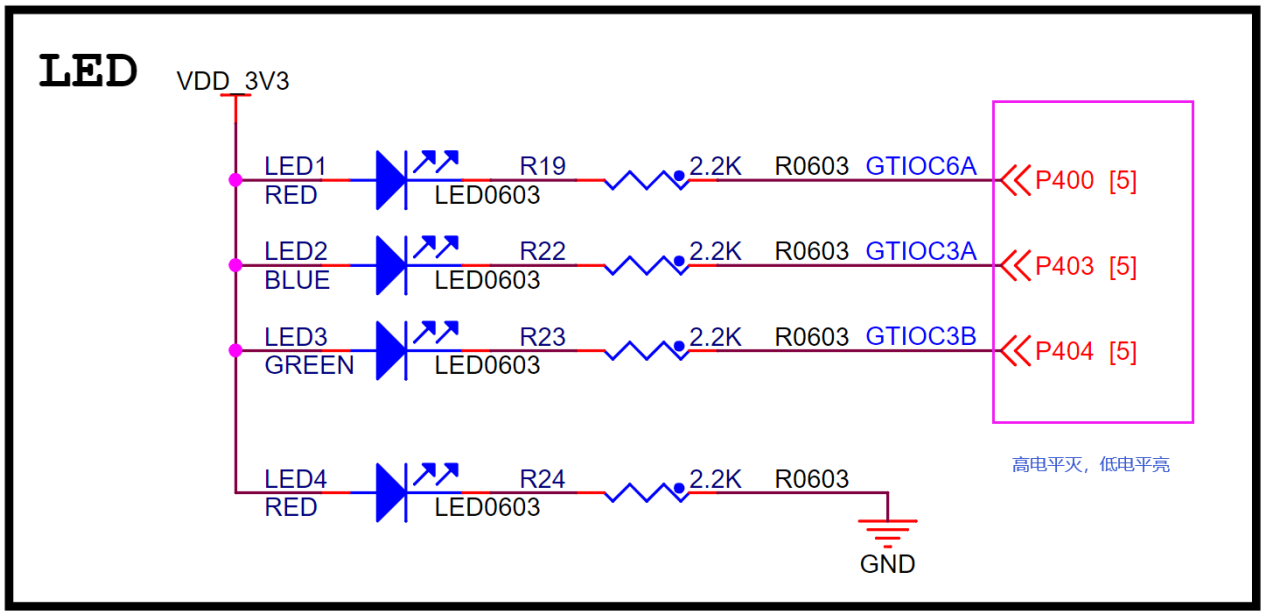
考虑到板卡丝印上明确写了usb口是串口，因此选用usb串口做工程调试串口，具体配置如下：



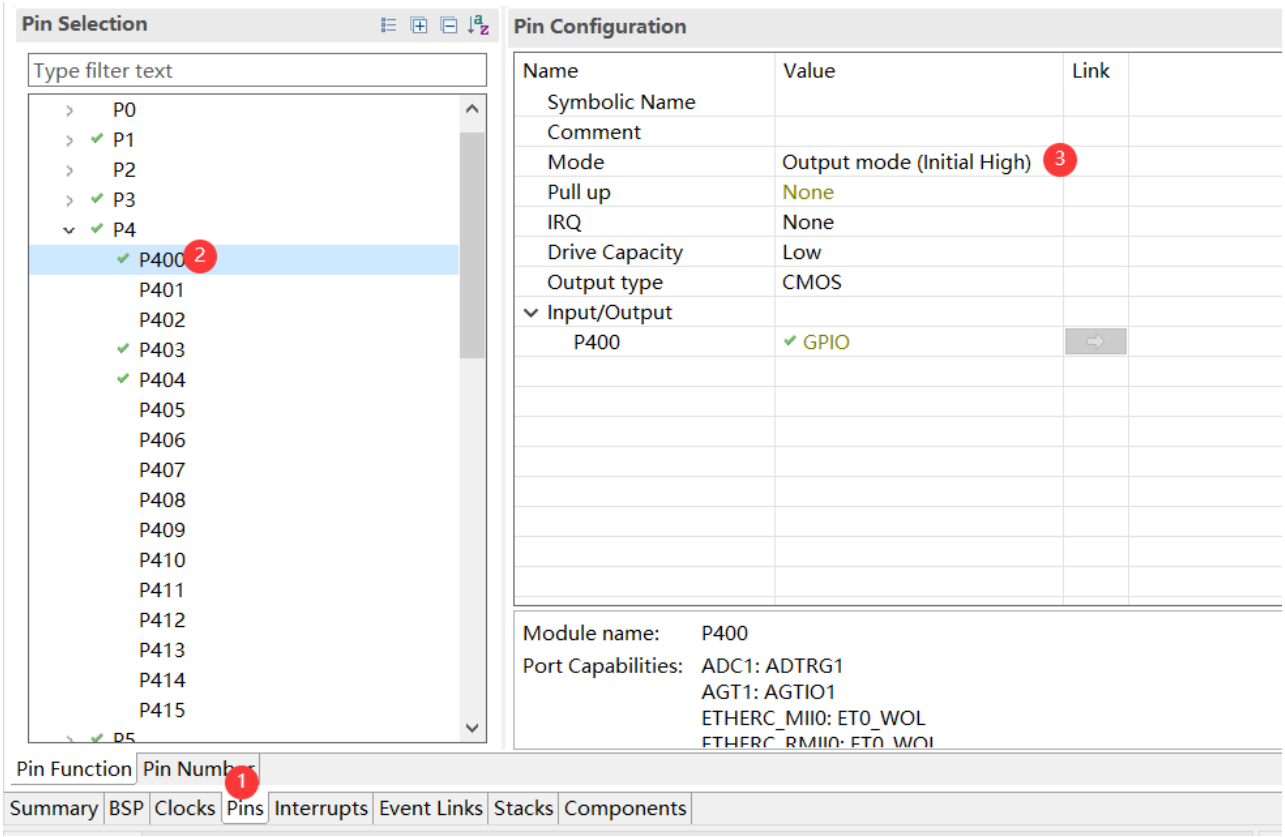



iii. 打开LED灯

1. 启明6M5灯资源




2. rasc配置

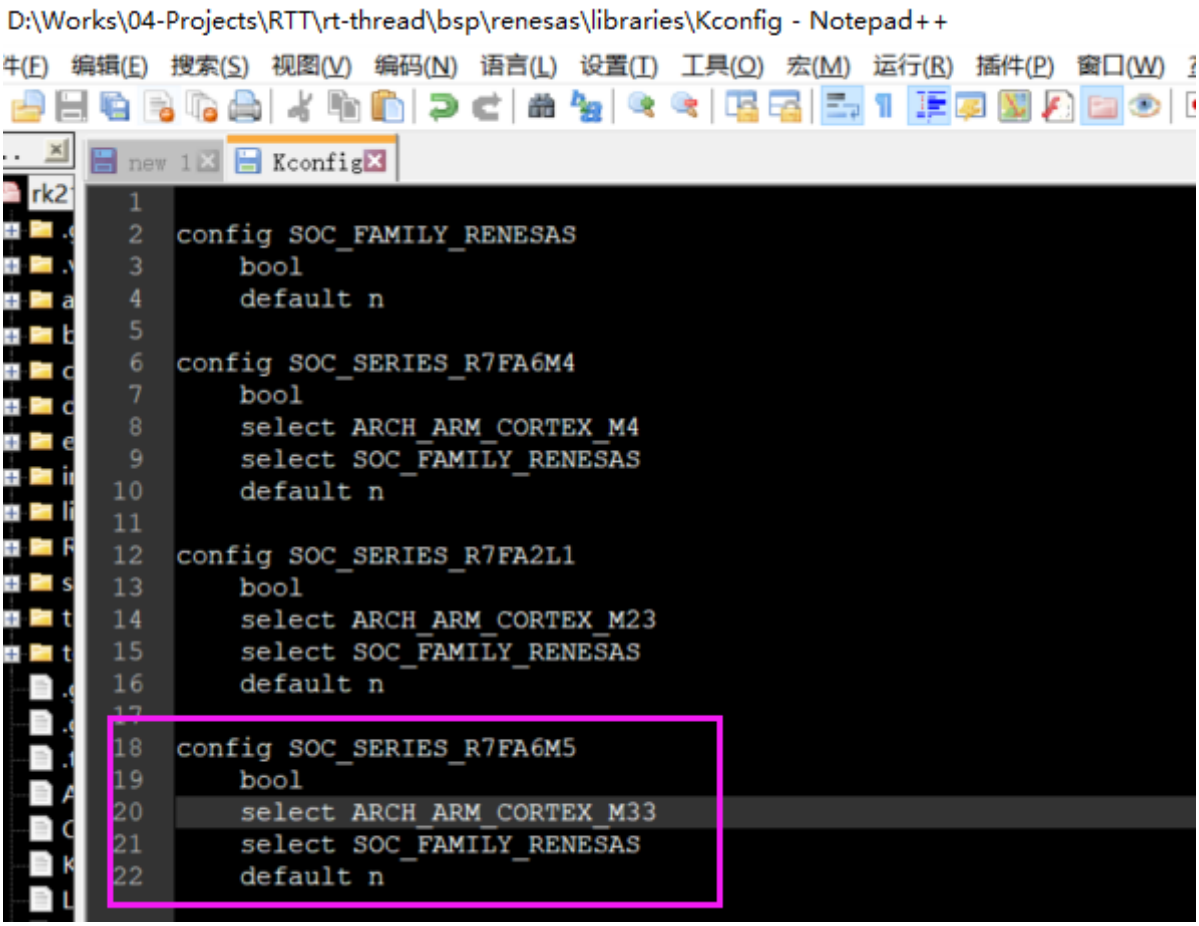



另外两个灯采用同样的方式配置

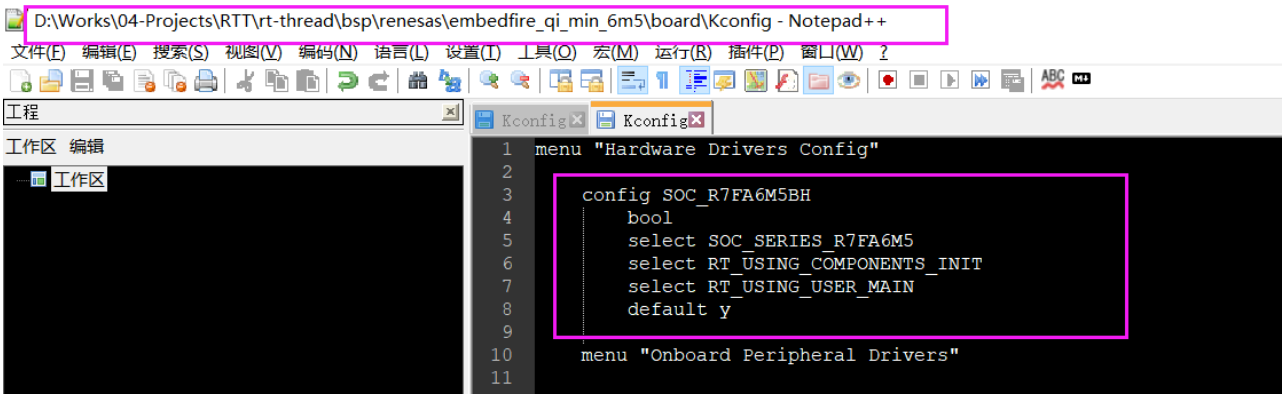
## d. RTT配置

### i. 增加芯片选择

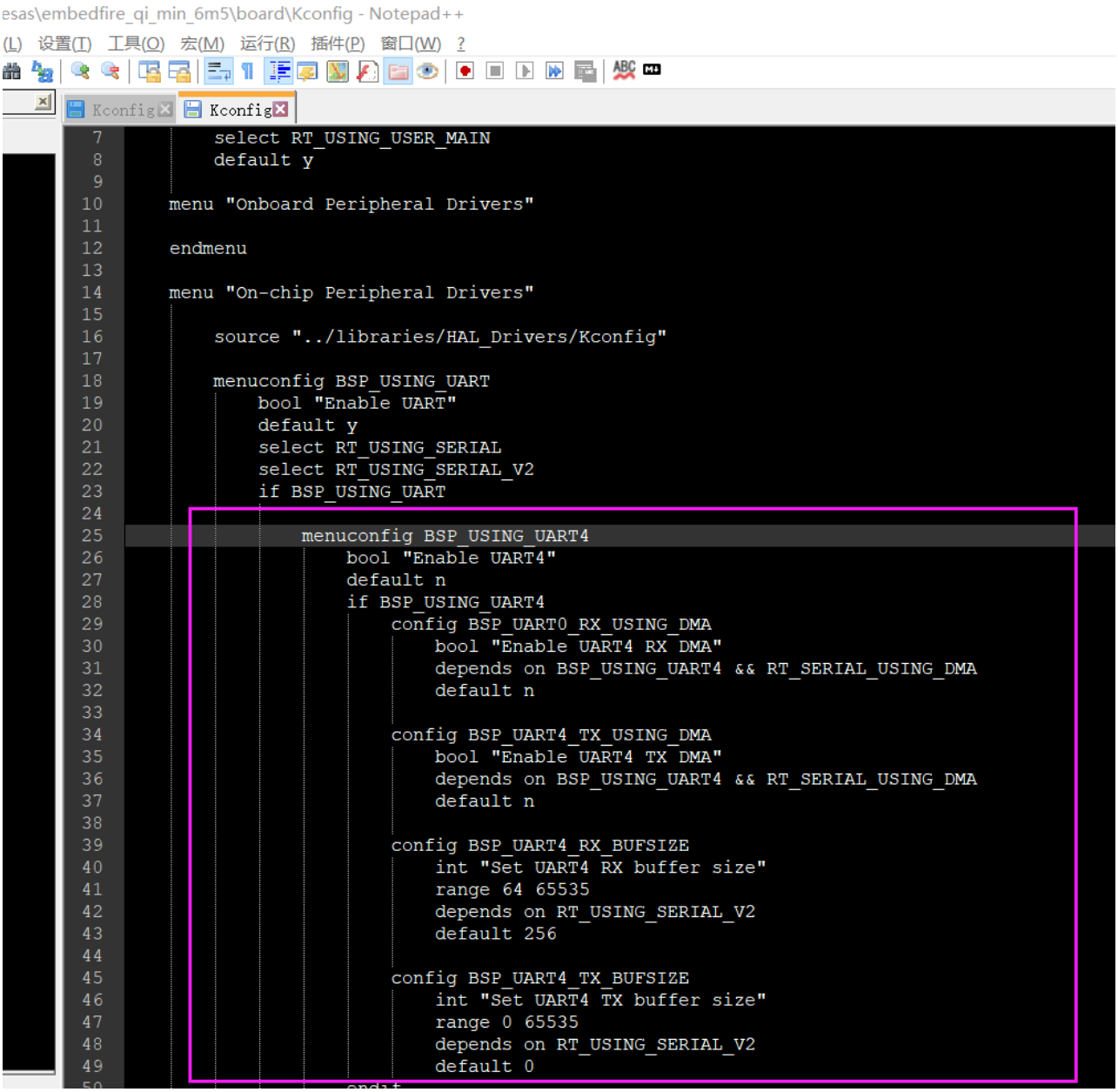
由于之前RTT并无RA6M5芯片配置，因此需手动添加RA6M5芯片配置，具体配置如下：



## ii. 增加驱动选择

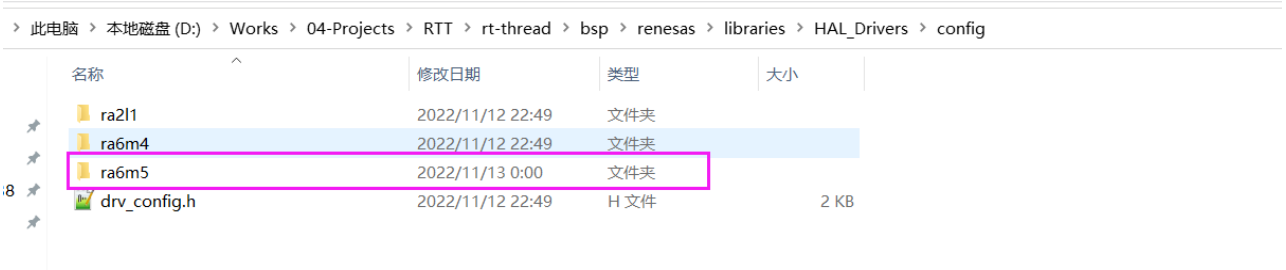



由于上面选的是uart4，因此KConfig配置成uart4

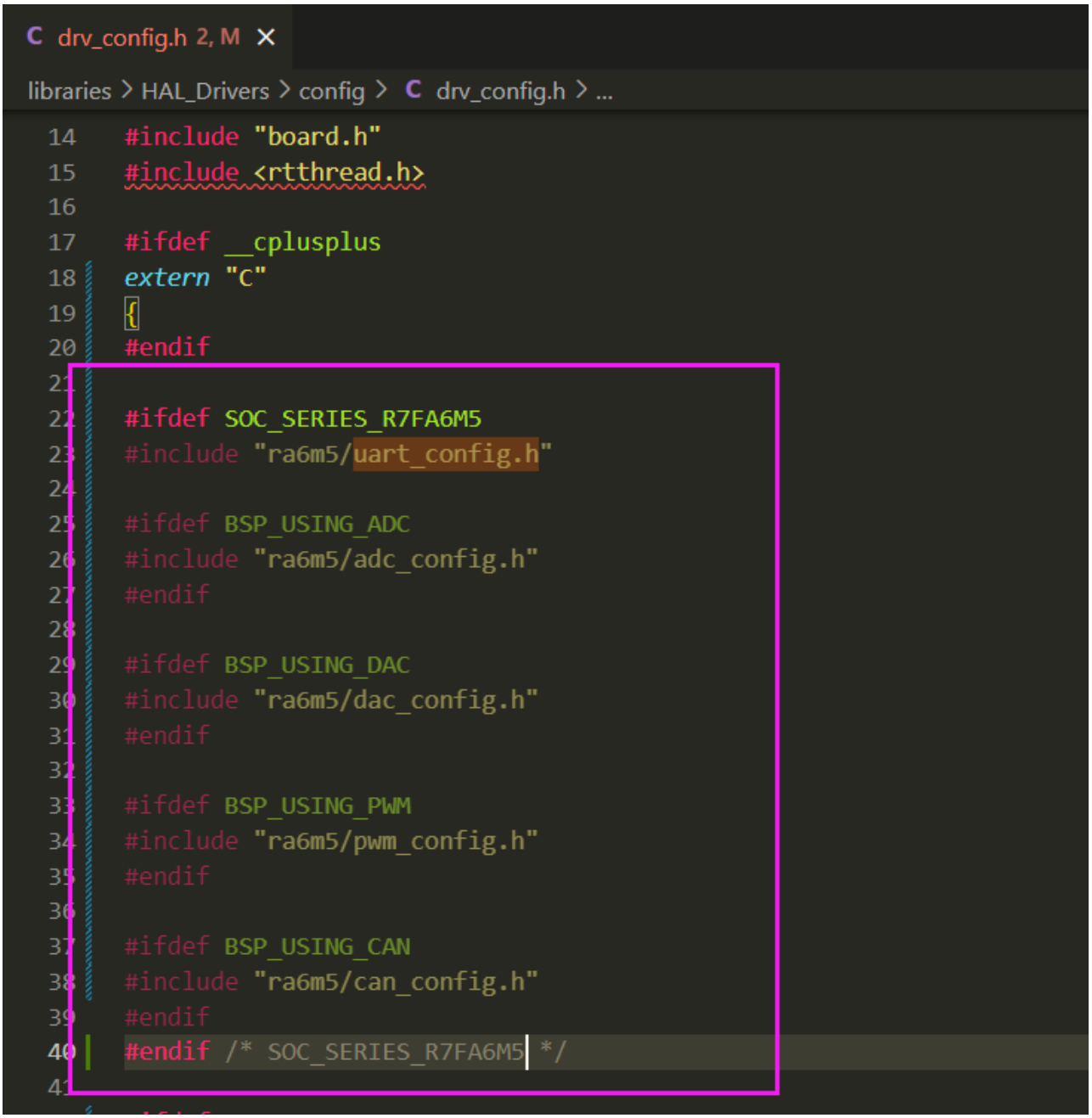



## iii. 增加变量适配层

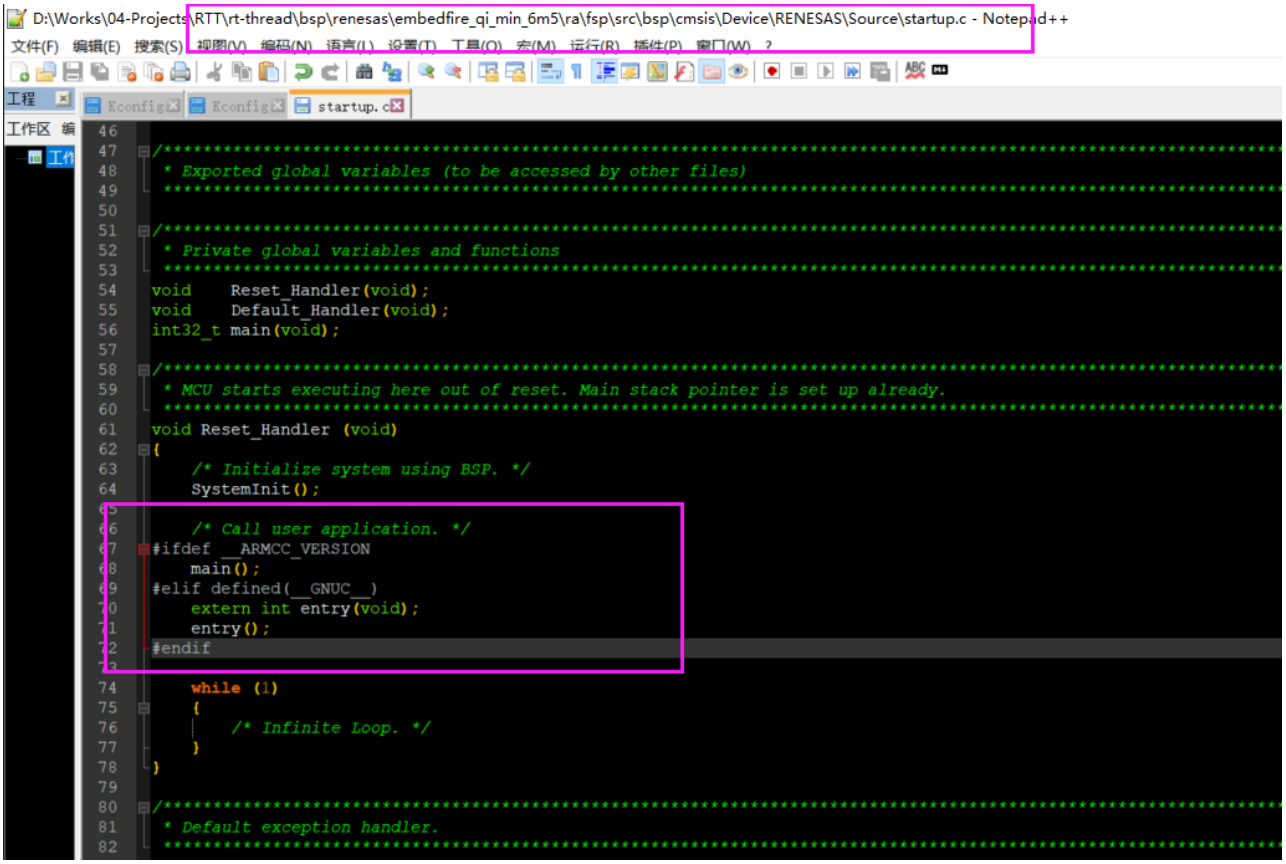
由于RA6M4和RA6M5资源差不多，因此先直接拷贝RA6M4并重命名为RA6M5，之后检查其中的uart部分



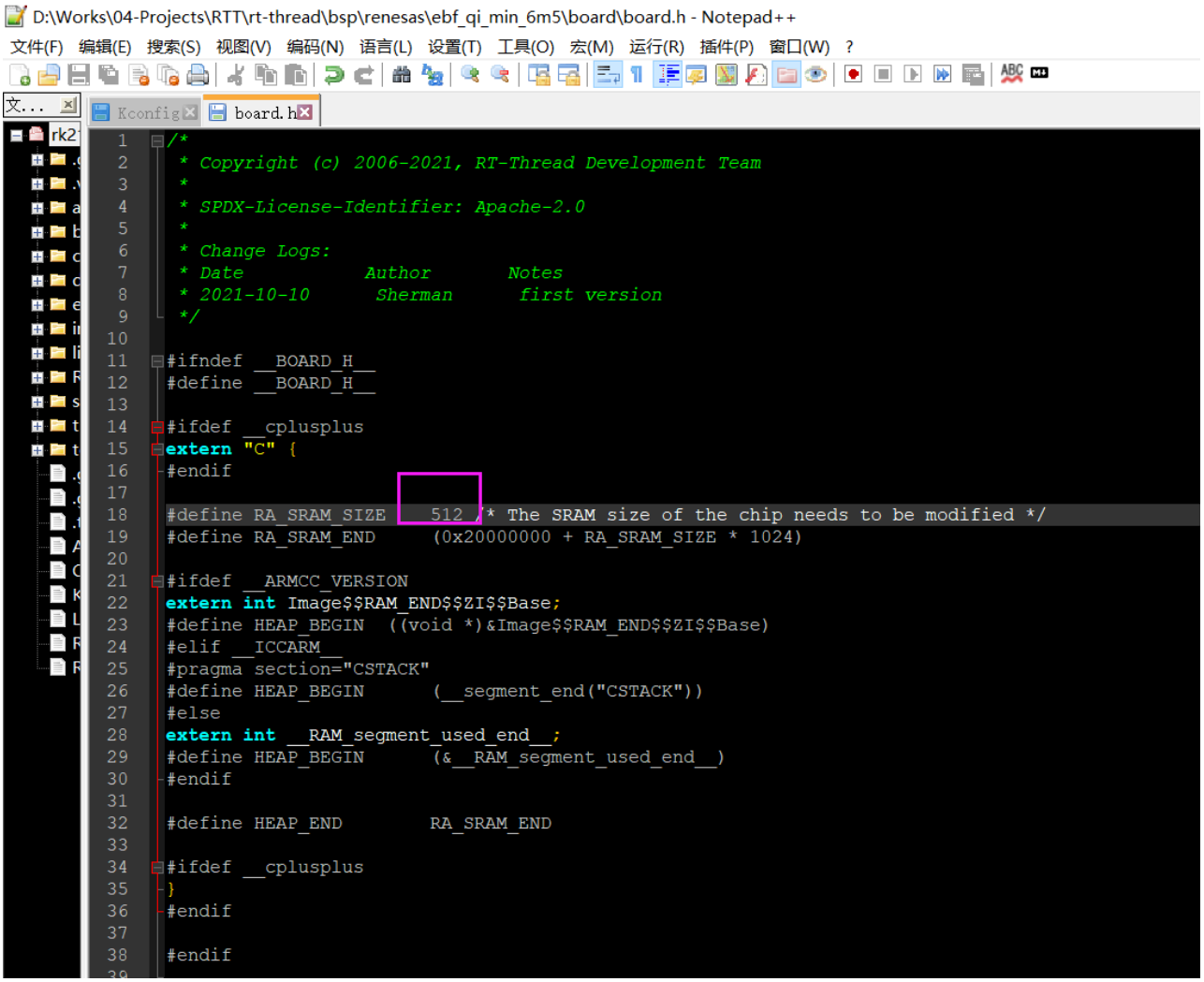

添加完毕后，修改drv_config.h文件




### iv. 修改main入口

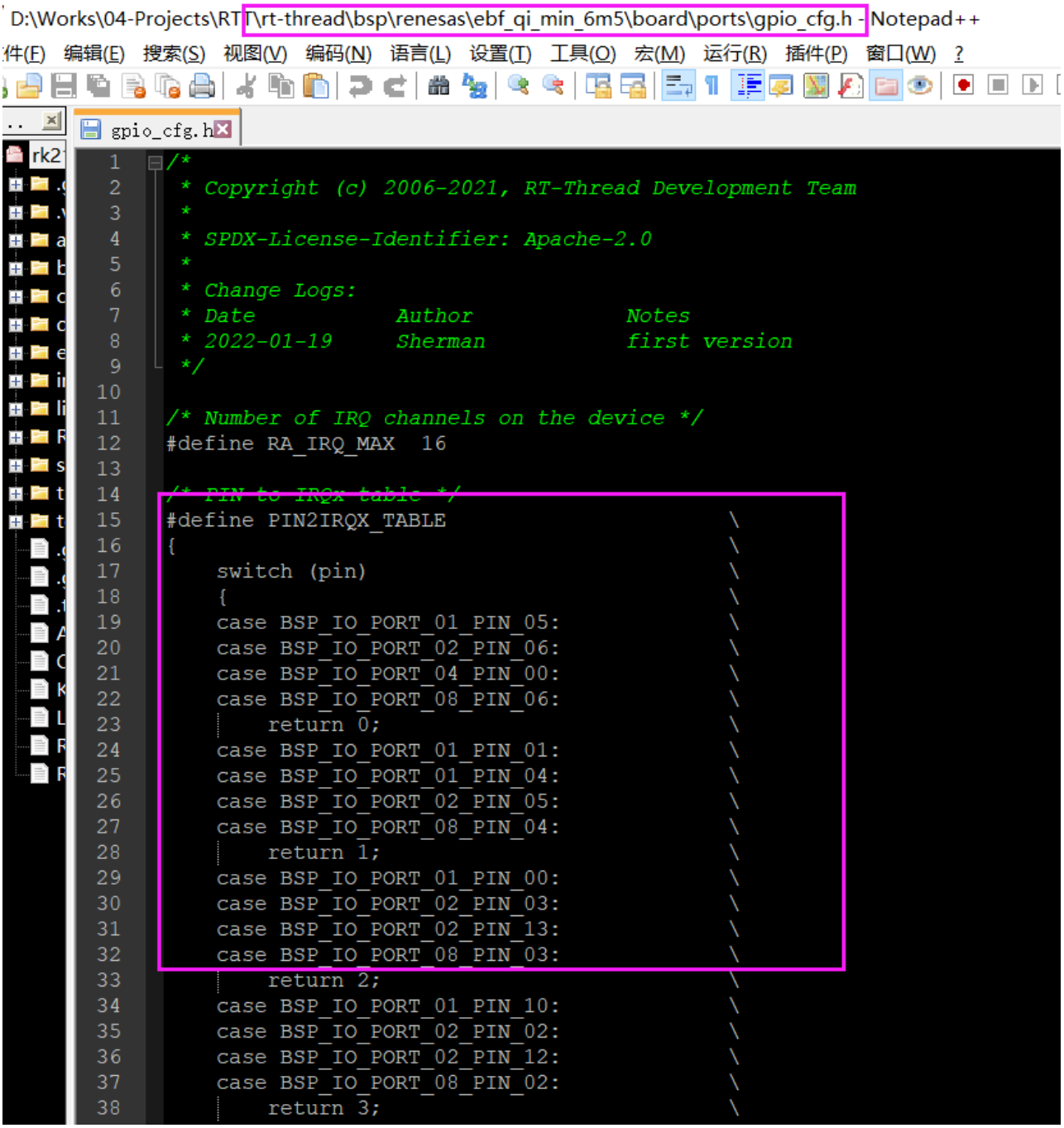



## v. 根据芯片手册修改SRAM大小




RA6M5 SRAM大小为512K，因此设为512K SRAM

### vi. 修改中断表




对比RASC pin中断分配表编写

### e. 修改应用代码（hal_entry.c)

```
/*
 * Copyright (c) 2006-2021, RT-Thread Development Team
 *
 * SPDX-License-Identifier: Apache-2.0
 *
 * Change Logs:
 * Date           Author       Notes
 * 2021-10-10     Sherman      first version
```

```c
 */

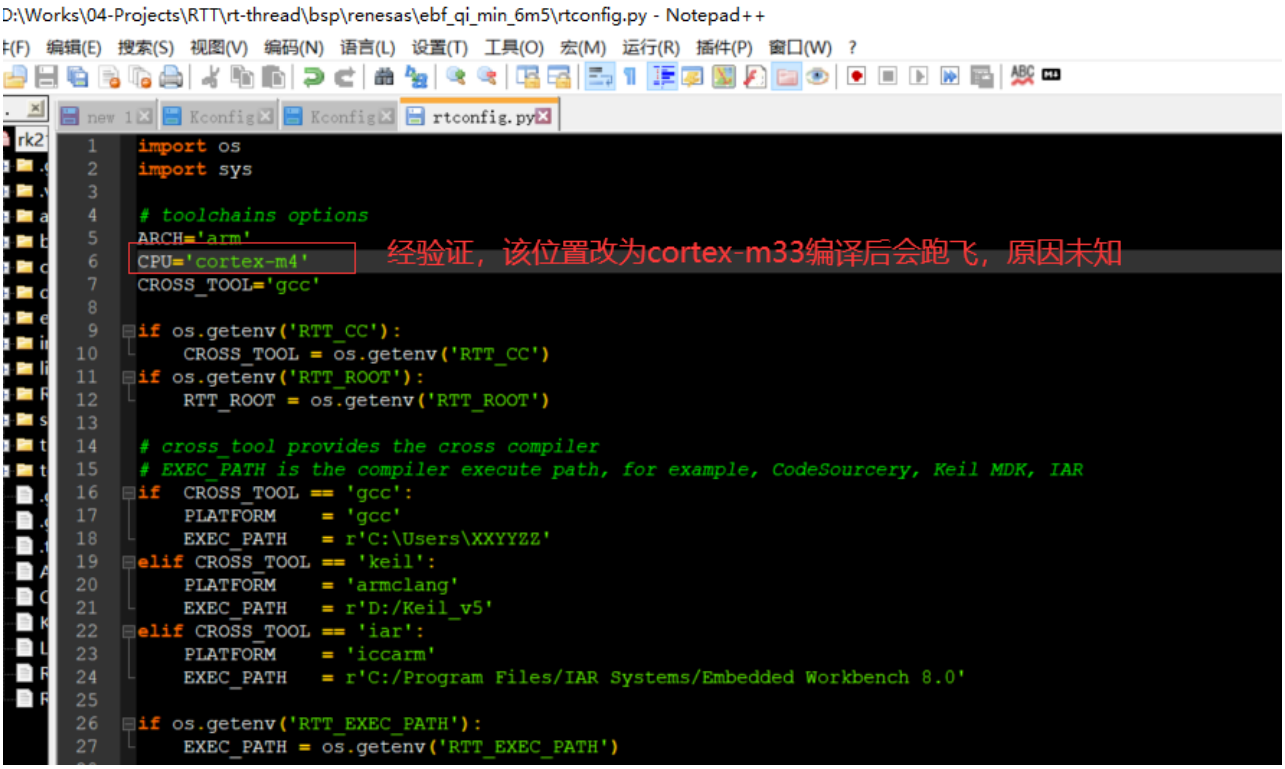
#include <rtthread.h>
#include "hal_data.h"
#include <rtdevice.h>

#define LED_PIN    BSP_IO_PORT_04_PIN_00 /* RED LED pins */

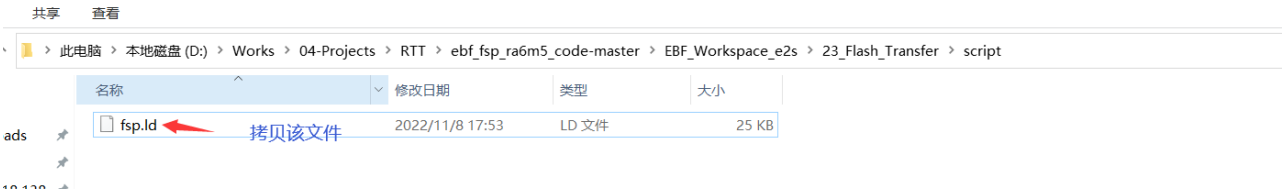
void hal_entry(void)
{
    rt_kprintf("\nHello RT-Thread!\n");

    while (1)
    {
        rt_pin_write(LED_PIN, PIN_HIGH);
        rt_thread_mdelay(500);
        rt_pin_write(LED_PIN, PIN_LOW);
        rt_thread_mdelay(500);
    }
}
```

## f. 修改rtconfig.py




## g. 从野火教程中拷贝fsp.ld到script目录下工具配置

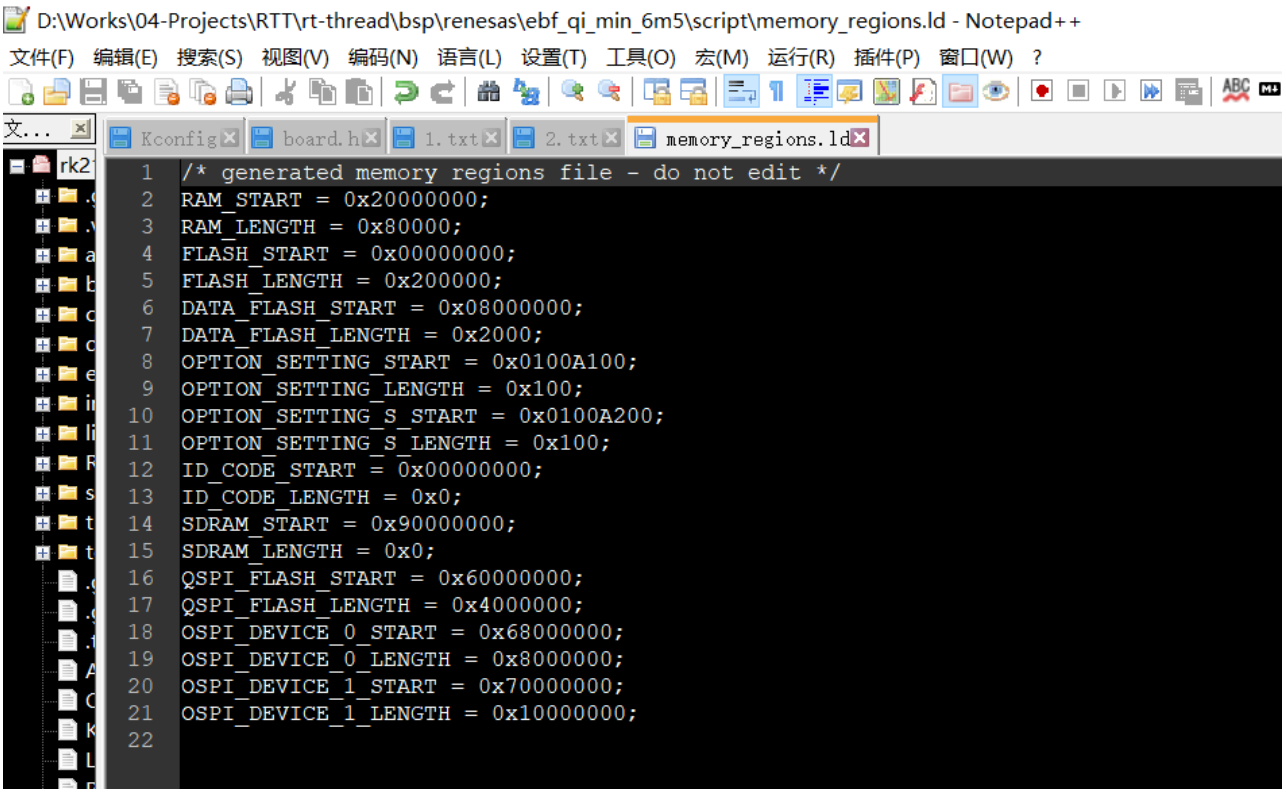



## h. 生成memory_regions.ld文件

对比RA6M4工程，发现memory_regions.ld和memory_regions.scat文件内容相似，可直接用scat文件改过去




因此手动制作该文件，制作结果为：




至此，GCC所需配置已经配置完毕：



## i. 修改链接脚本

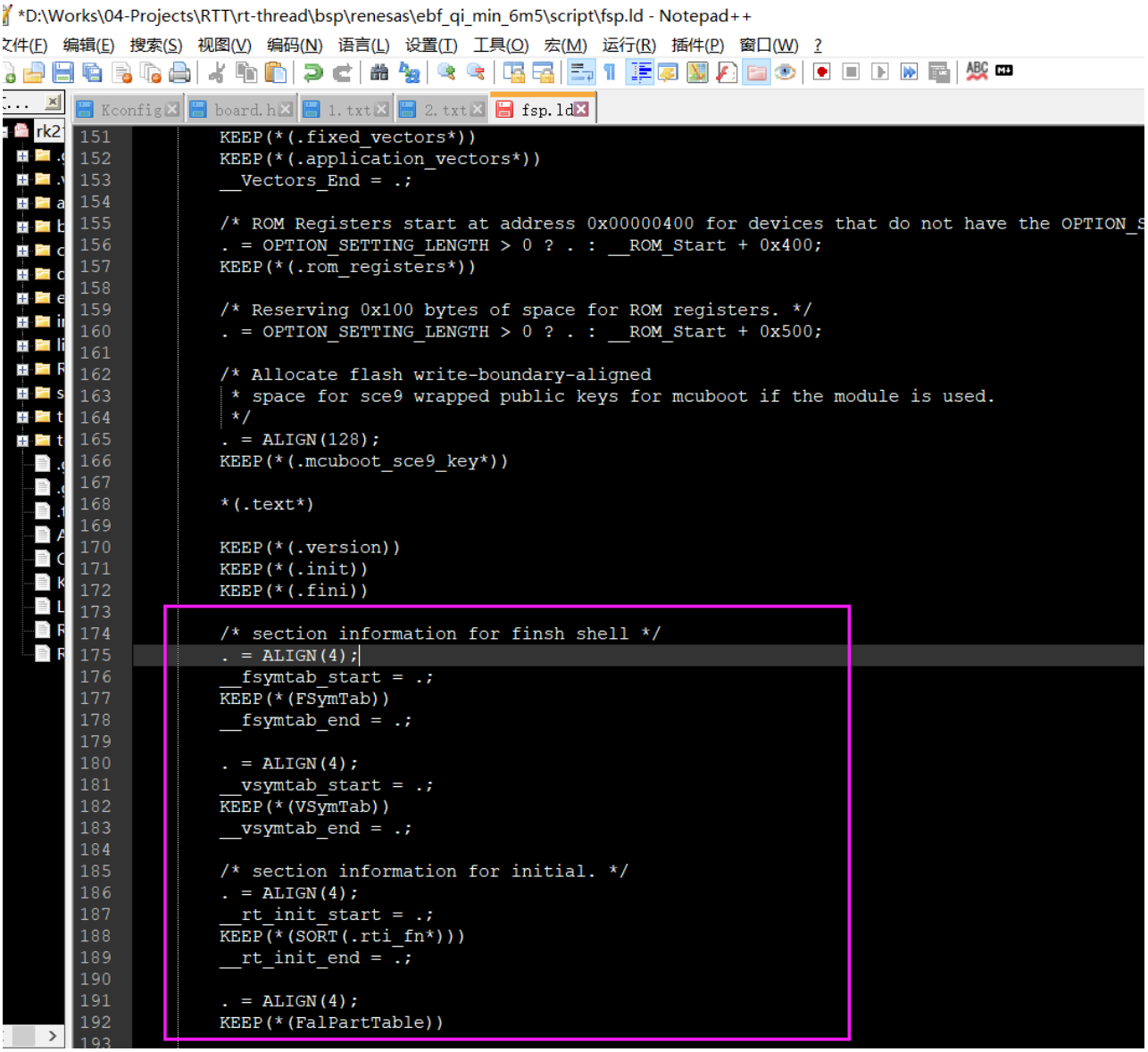
打开fsp.ld，加入以下代码：

```
/* section information for finsh shell */
. = ALIGN(4);
__fsymtab_start = .;
KEEP(*(FSymTab))
__fsymtab_end = .;

. = ALIGN(4);
__vsymtab_start = .;
KEEP(*(VSymTab))
__vsymtab_end = .;

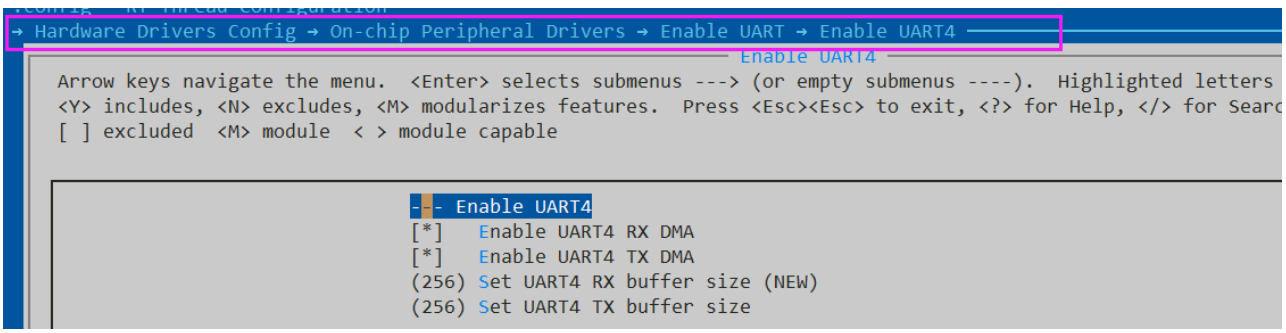
/* section information for initial. */
. = ALIGN(4);
__rt_init_start = .;
KEEP(*(SORT(.rti_fn*)))
__rt_init_end = .;

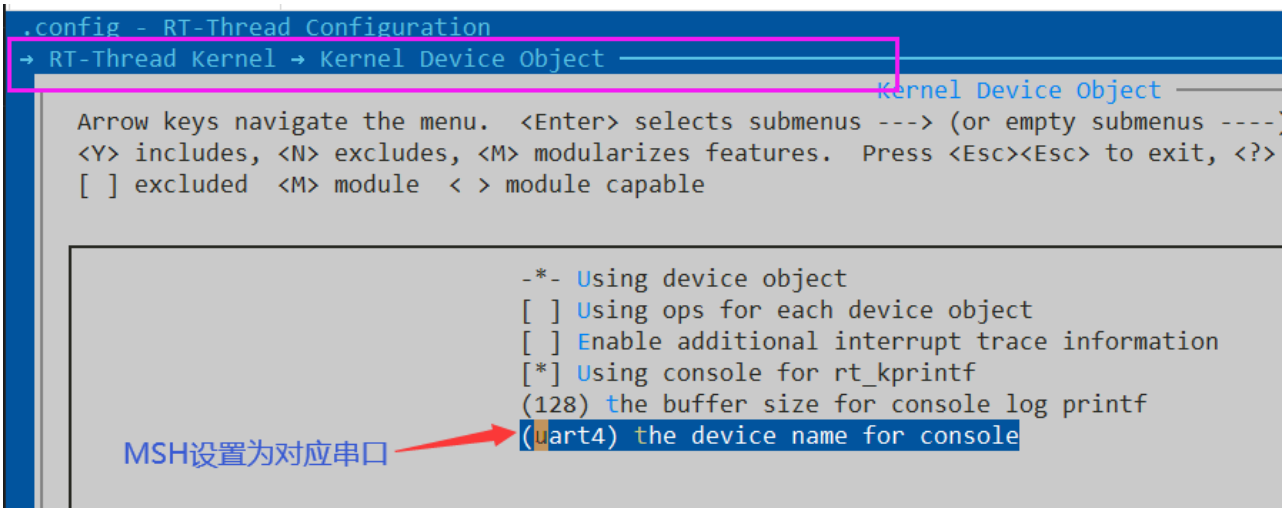
. = ALIGN(4);
KEEP(*(FalPartTable))
```




## j. 运行scons，配置串口和GPIO

在bsp更目录下运行env，并在弹出界面中输入 `menuconfig` 进行配置





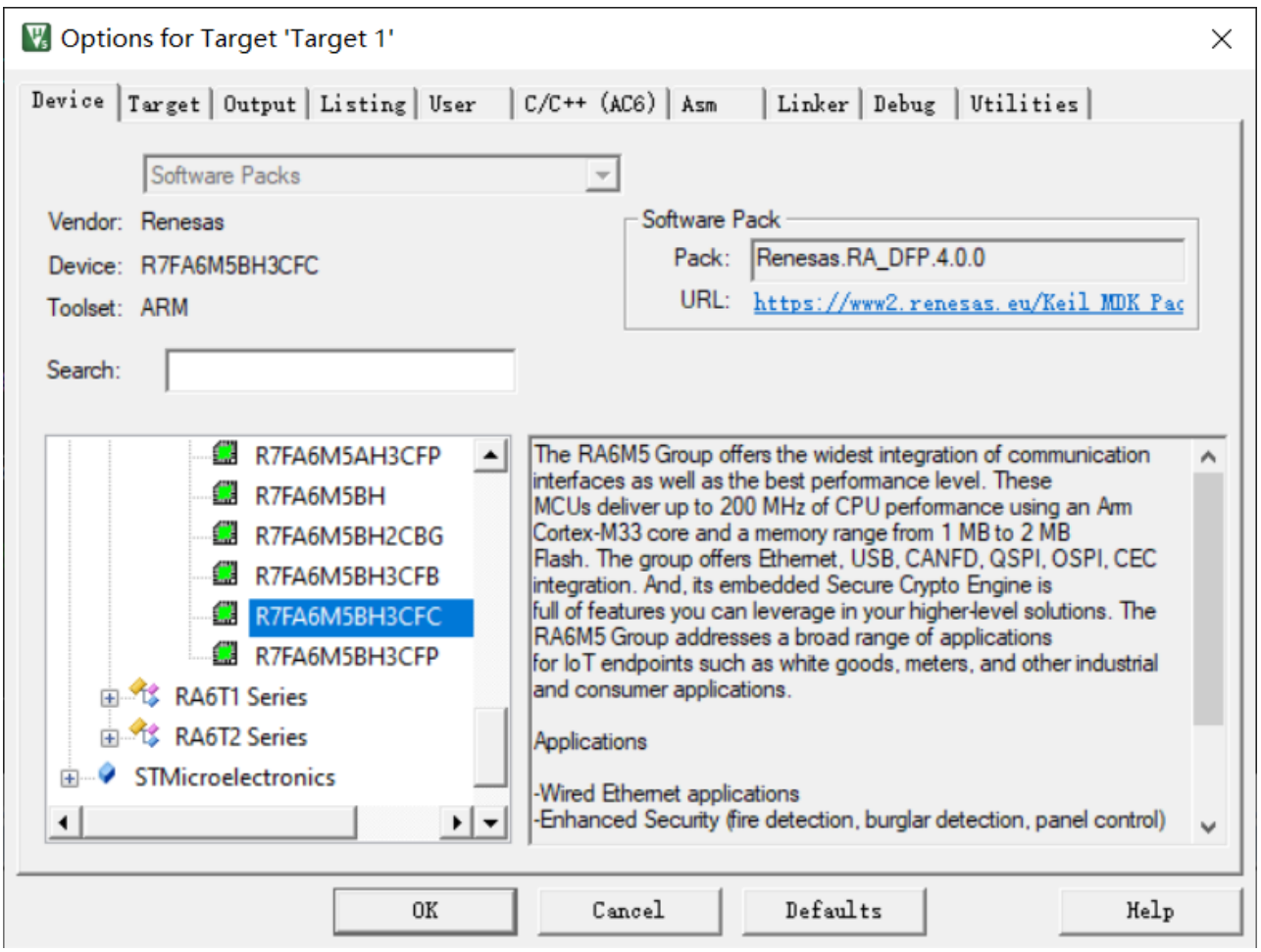

### k. 重新生成工程

在env中运行 `scons --target=mdk5`

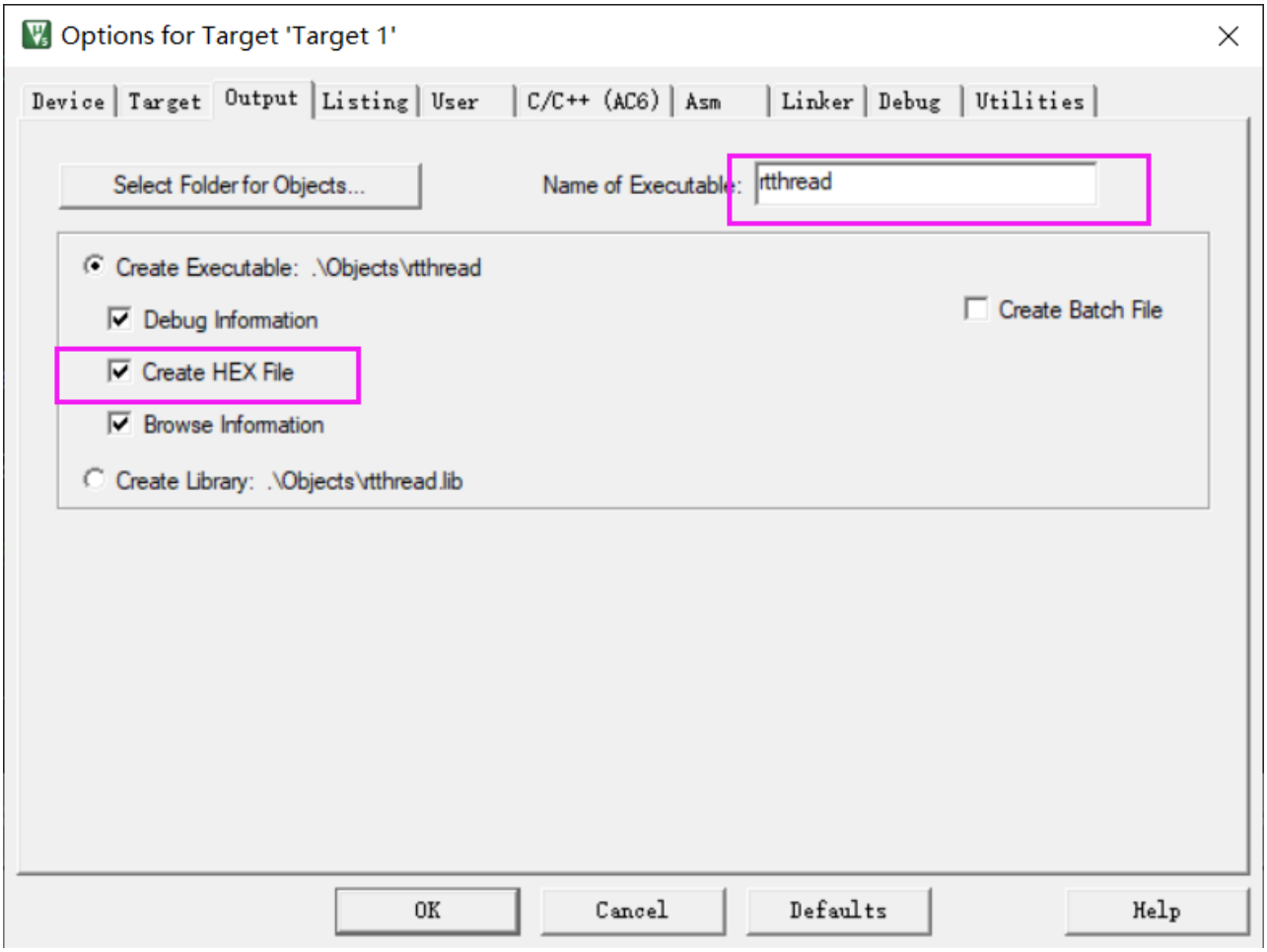
```
user@IT000 D:\Works\04-Projects\RTT\rt-thread\bsp\renesas\ebf_qi_min_6m5
> scons --target=mdk5
scons: Reading SConscript files ...
Keil5 project is generating...
Keil Version: MDK Plus 5.27 (Flex)/ARM Compiler 6.12/armlink [5d624300]
```

## l. 修改工程配置

### i. 芯片配置更改

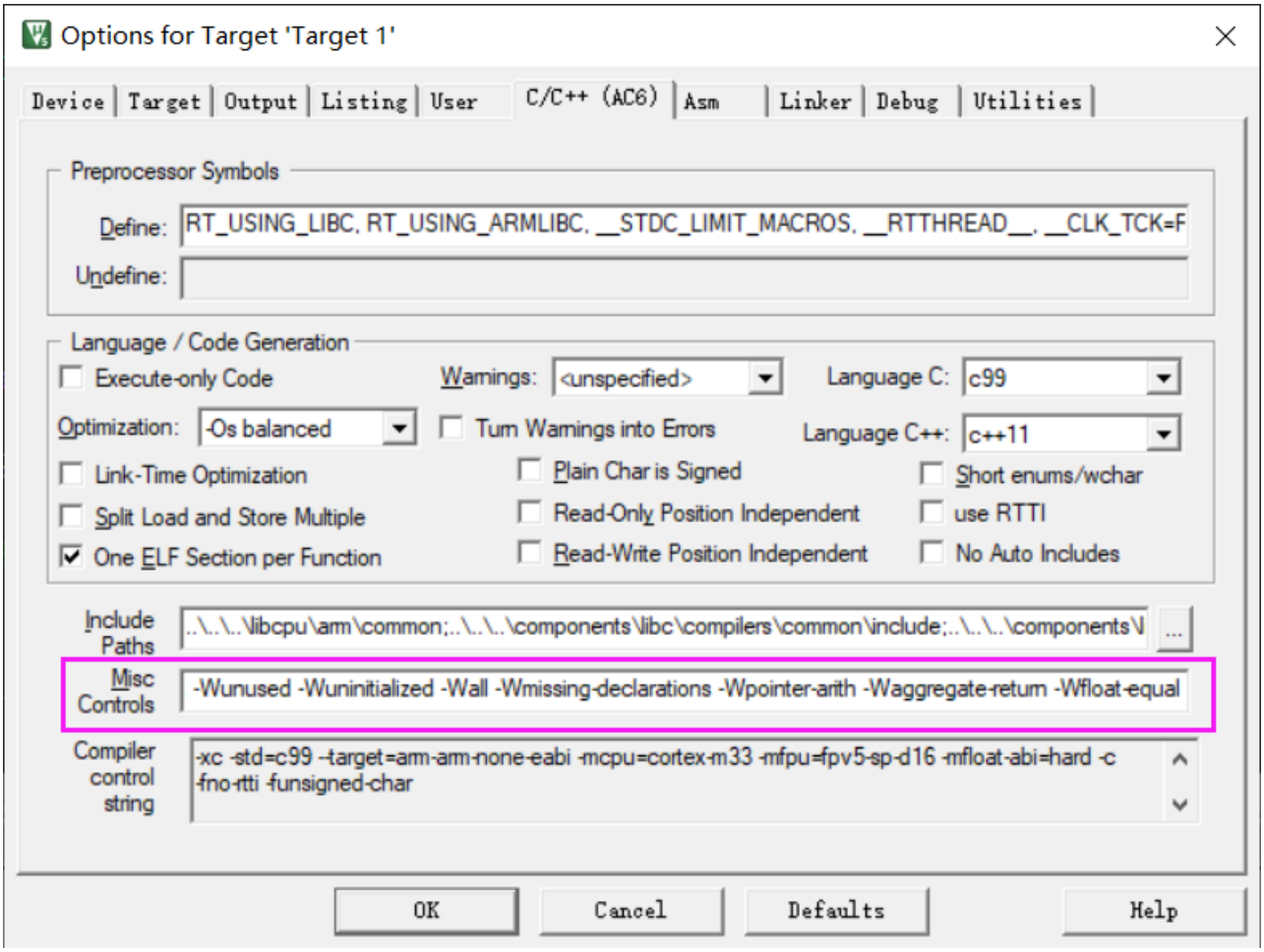


### ii. 修改编译生成的目标文件

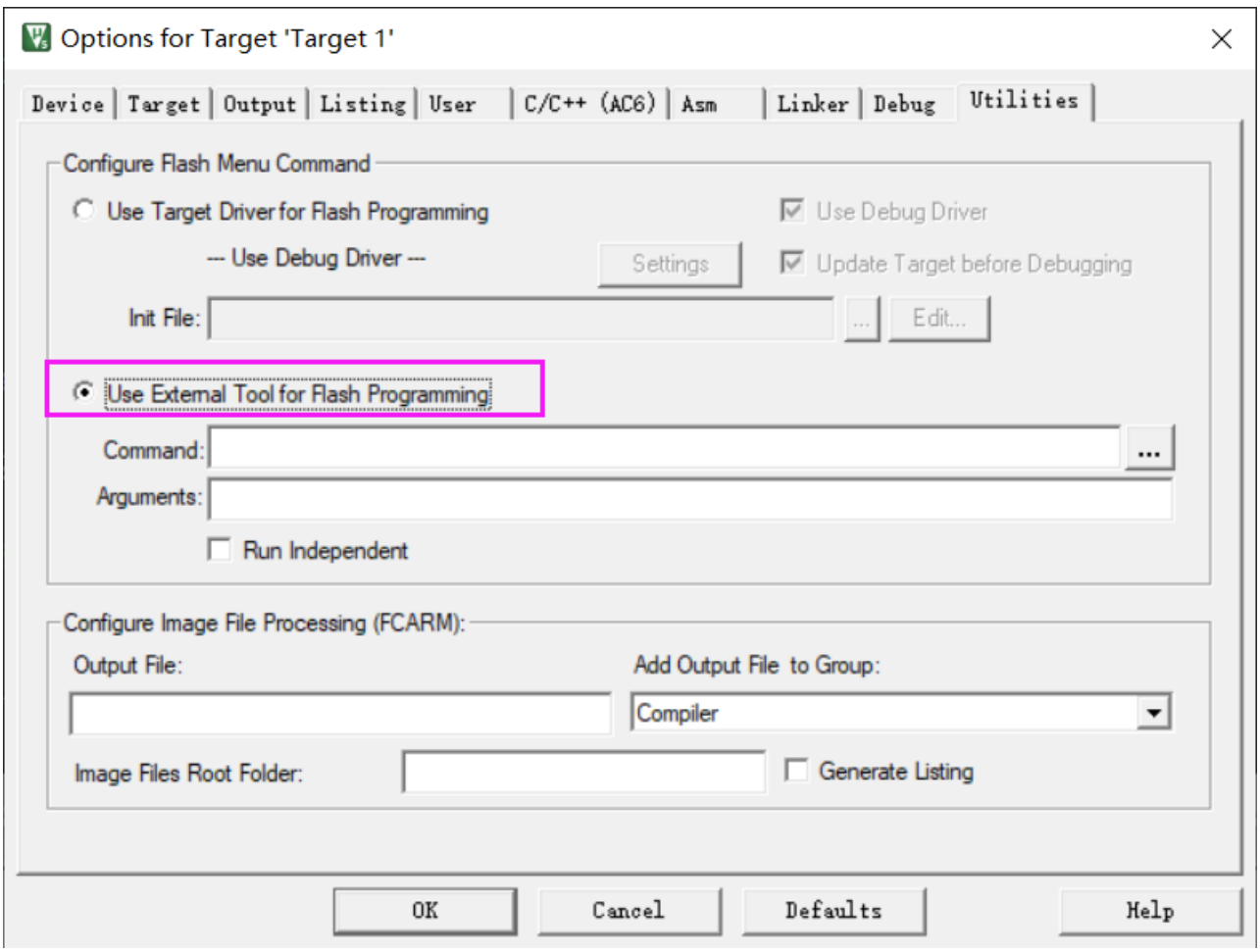



### iii. 修改和关闭部分编译警告

去掉三个参数：“-Wextra“、”-Wconversion“、”-Wshadow”




### iv. 勾选 ‘Use Ecternal Tool for Flash Programming’



## 3. 初步效果

```
 \ | /
- RT -     Thread Operating System
 / | \     5.0.0 build Nov 14 2022 20:43:44
 2006 - 2022 Copyright by RT-Thread team

Hello RT-Thread!
msh >
msh >
msh >
msh >ls
ls: command not found.
msh >lis
list_thread
list_sem
list_event
list_mutex
list_mailbox
list_msgqueue
list_timer
list_device
list
msh >listd
msh >list_d
list_device
msh >list_device
device           type         ref count
-------- -------------------- ----------
uart4    Character Device     2
pin      Pin Device           0
msh >
```

## 4. 注意事项

a. 由于编辑时发现野火所有资料简称都为ebf，个人觉得可能是embedfire的简称，所以将工程名中的embedfire改为了ebf，这会导致步骤截图中出现ebf和embedfire共存的情况，查阅时需注意。

b. 由于个人在配置时是按照将BSP PR至RTT的做法实现的，因此中间添加了几步GCC配置，即 章节2中的g h i 步，若只在keil MDK下使用，这几步可跳过。

c. rtconfig.py中的CPU不能配置成cortex-m33，只能配置cortex-m4原因暂不确定。

## 5. 参考资料

RA系列BSP制作教程.md