

联睿微 SDK_V3.0 使用手册

文档修改记录

版本	日期	修改内容	作者

第一章 SDK 简介

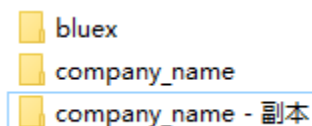
1、目录结构



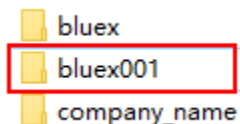
1.1 application

此文件夹下，起码有两个文件夹，**bluex** 和 **company_name**，其中 **bluex** 为本公司定义的一些应用，**company_name** 为通用模板，用户如需定义自己的项目，可以直接复制文件夹，然后更改相关文件名即可。如一个名为 **bluex001** 的企业，新建一个蓝牙锁的项目，可以按以下步骤快速新建项目：

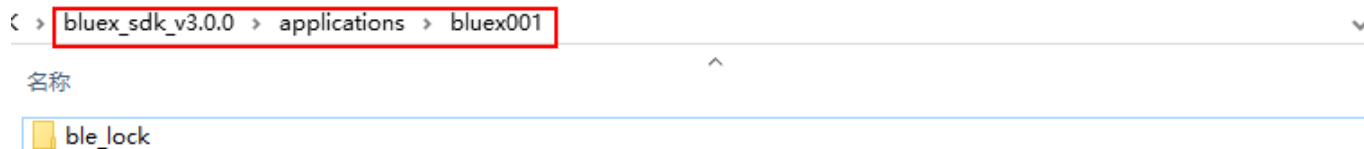
- 复制 **company_name** 文件夹

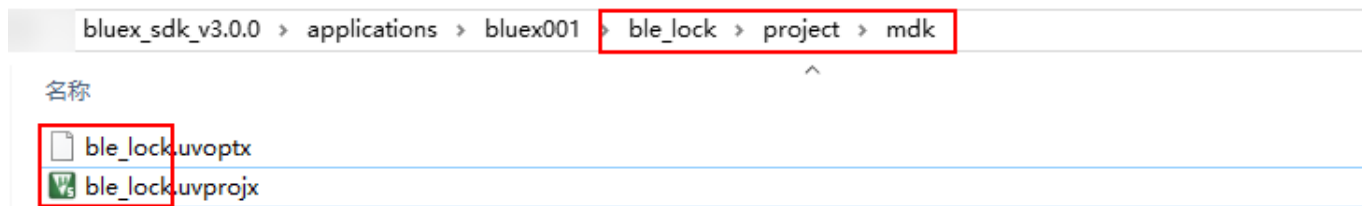


- 更改企业名称为 **bluex001**



- 更改项目名称为 **ble_lock**





- 最后打开工程项目即可。

第二章 搭建开发环境

1、硬件工具

名称	描述
BX2400_EVK 开发板	开发板
Jlink 仿真器	仿真和下载程序
杜邦线、跳线帽	连接开发板和仿真器

2、软件工具

名称	描述
win7/win10	计算机系统
mdk 集成开发环境	BlueX 芯片开发环境
J-Flash 软件	读写芯片，查看调试信息，非必须
nRF Connect	手机端软件，用来扫描、连接、读写设备等

3、安装工具

3.1 安装 MDK

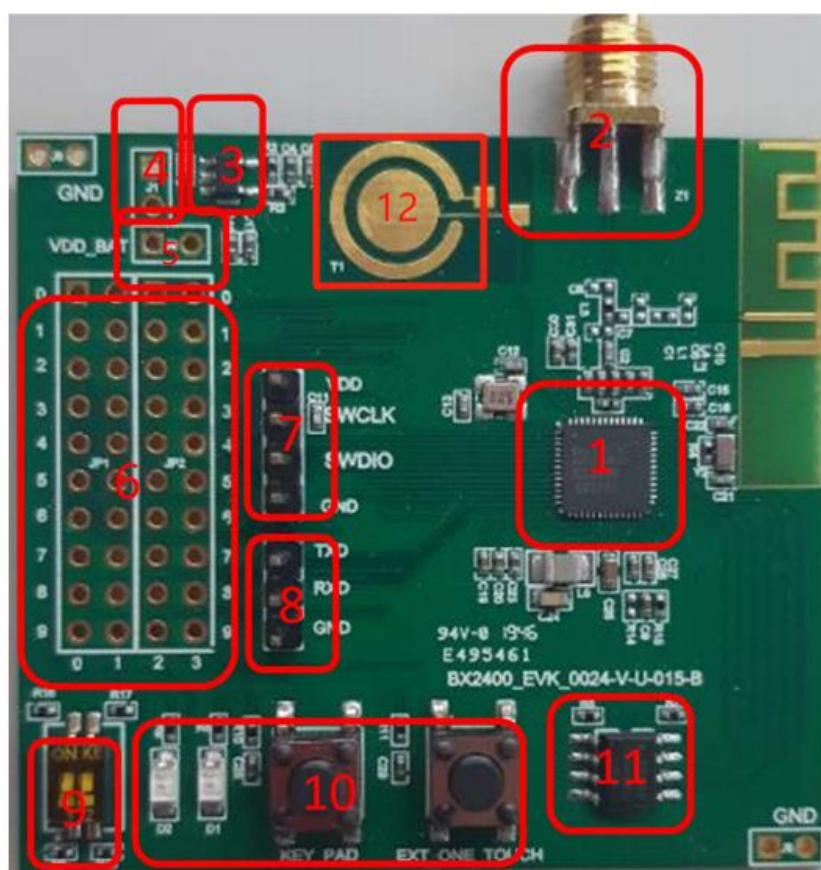
3.2 安装 JFlash

3.3 安装手机端 APP

第三章 开发板介绍

1、BX2400_EVK

1.1 功能特点



- 1、RF01(QFN52)
- 2、外接天线座
- 3、板载3.3V LDO
- 4、跳线端子
- 5、电池供电端
- 6、IO引出脚
- 7、SWD调试口
- 8、串口接口
- 9、boot启动配置
- 10、按键和LED
- 11、SPI Flash芯片
- 12、触摸按键

注意：在第9个boot启动配置中，左边的开关（P16）用于选择SPI Flash启动还是UART启动，下拨（接GND）则正常从SPI Flash启动，上拨一般用于强制烧录芯片。右边的开关（P23）用于适配SPI Flash芯片的供电电压，EVK开发板保持下拨接地即可。

1.2 硬件原理

1.2.1 按钮

1.2.2 触摸按钮

1.2.3 LED

1.2.4 拨码开关

1.2.5 天线

1.2.6 Jlink 接口

1.2.7 串口接口

1.2.8 电源接口

1.2.9 IO 拓展

第四章 新建工程

1、新建模板

1.1 事前准备

1.1.1 把文件移动到相关路径

打开 SDK 工具箱，找到 prog_tool_v2:

< > bluex_sdk_v3.0.0 > tools > bluex > prog_tool_v2				
名称	修改日期	类型	大小	
BlueX	2020/3/25 17:08	文件夹		
JLinkDevices.xml	2019/9/27 8:59	XML 文档	1 KB	
ReadMe.txt	2019/9/27 8:59	文本文档	1 KB	

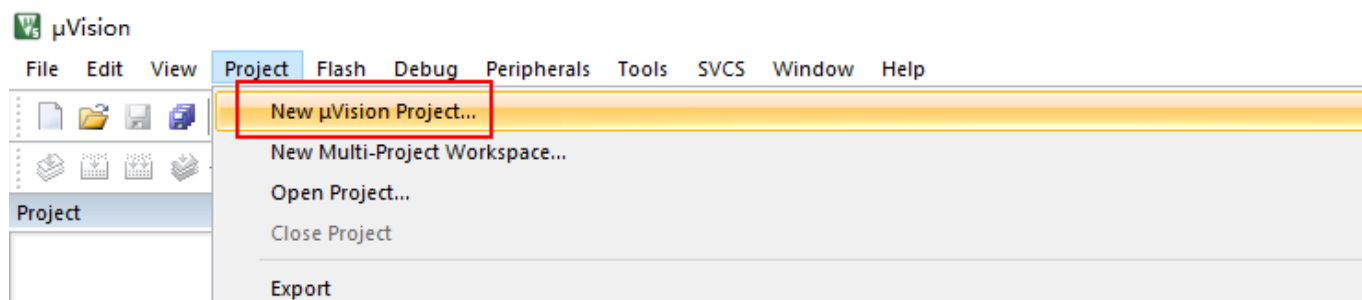
将 BlueX 文件夹中的 APOLLO_00_1V8.FLM 和 APOLLO_00_3V3.FLM 拷贝到 Keil_v5 安装目录下的 ARM/Flash 目录中，如下图：

电脑 > > > KEIL5 > ARM > Flash >				
名称	修改日期	类型	大小	
S29GL064NX2	2020/3/9 17:55	文件夹		
SP29JL032H	2020/3/9 17:55	文件夹		
AM29F160DB.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29F160DT.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29F320DB.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29F320DBx2.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29F320DT.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29F320DTx2.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29x033.FLX	2015/7/8 16:30	FLX 文件	13 KB	
AM29x128.FLM	2015/7/8 16:30	FLM 文件	13 KB	
AM29x128.FLX	2015/7/8 16:30	FLX 文件	13 KB	
AM29x800BB.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29x800BBx2.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29x800BT.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29x800BTx2.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29x800DB.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29x800DBx2.FLX	2015/7/8 16:30	FLX 文件	14 KB	
APOLLO_00_1V8.FLM	2019/9/27 8:59	FLM 文件	90 KB	
APOLLO_00_3V3.FLM	2019/9/27 8:59	FLM 文件	90 KB	
FlashOS.h	2018/3/5 19:30	H 文件	4 KB	
K8P5615UQA_x2.FLM	2015/7/8 16:30	FLM 文件	11 KB	
LPC18xx43xx_MX25V8035F.FLM	2017/11/6 19:30	FLM 文件	112 KB	
LPC18xx43xx_S25FL032.FLM	2015/7/8 16:30	FLM 文件	77 KB	
LPC18xx43xx_S25FL064.FLM	2015/7/8 16:30	FLM 文件	77 KB	
LPC18xx43xx_S25FL102.FLM	2015/7/8 16:30	FLM 文件	77 KB	

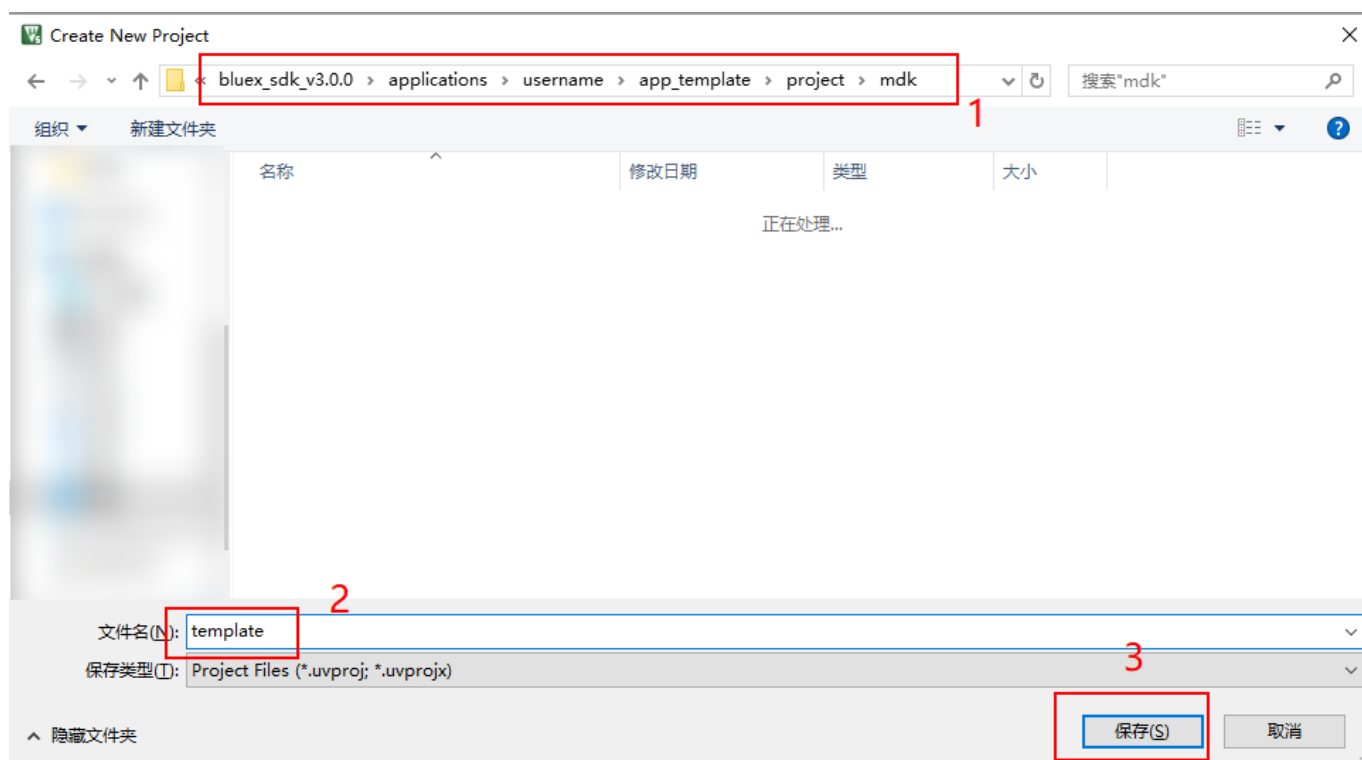
1.1.2 文件目录规划

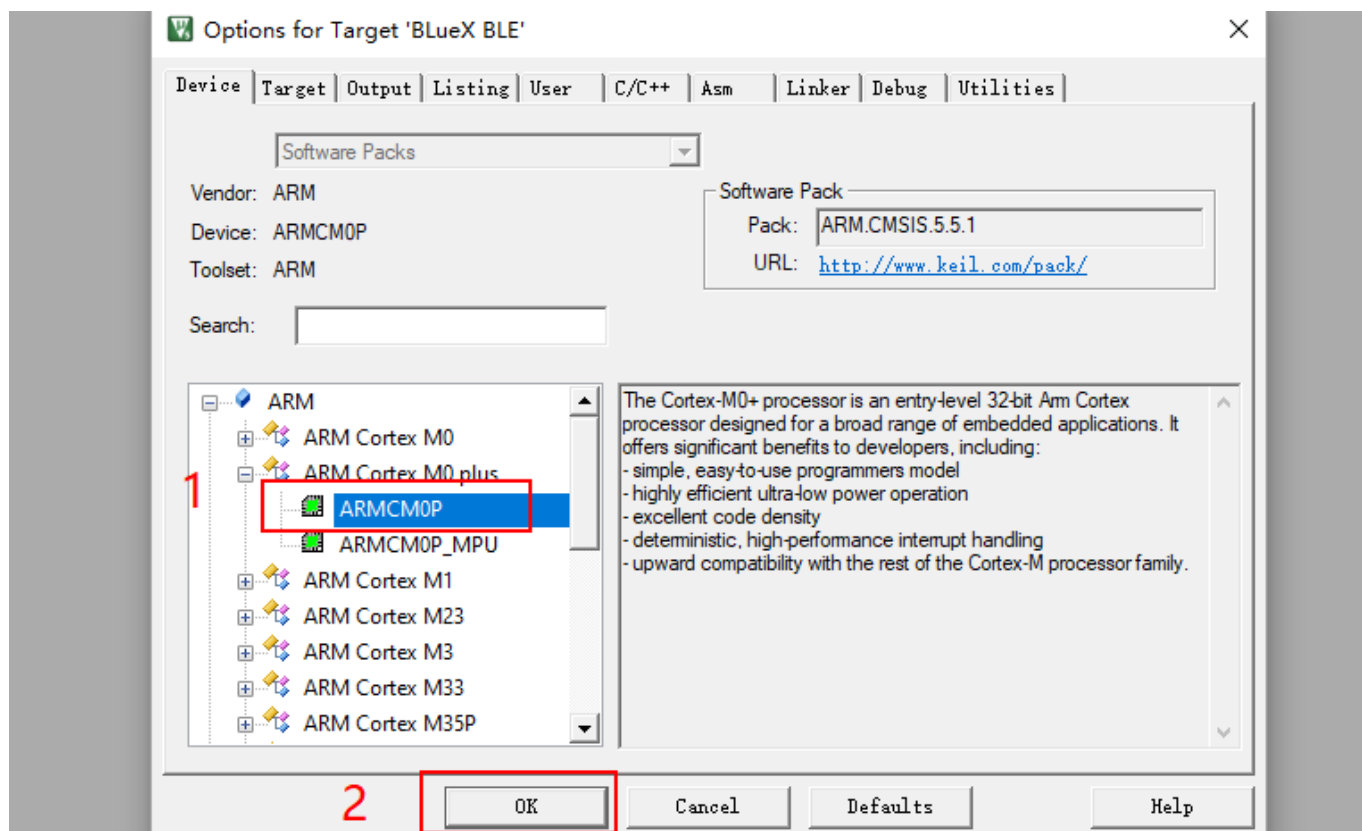


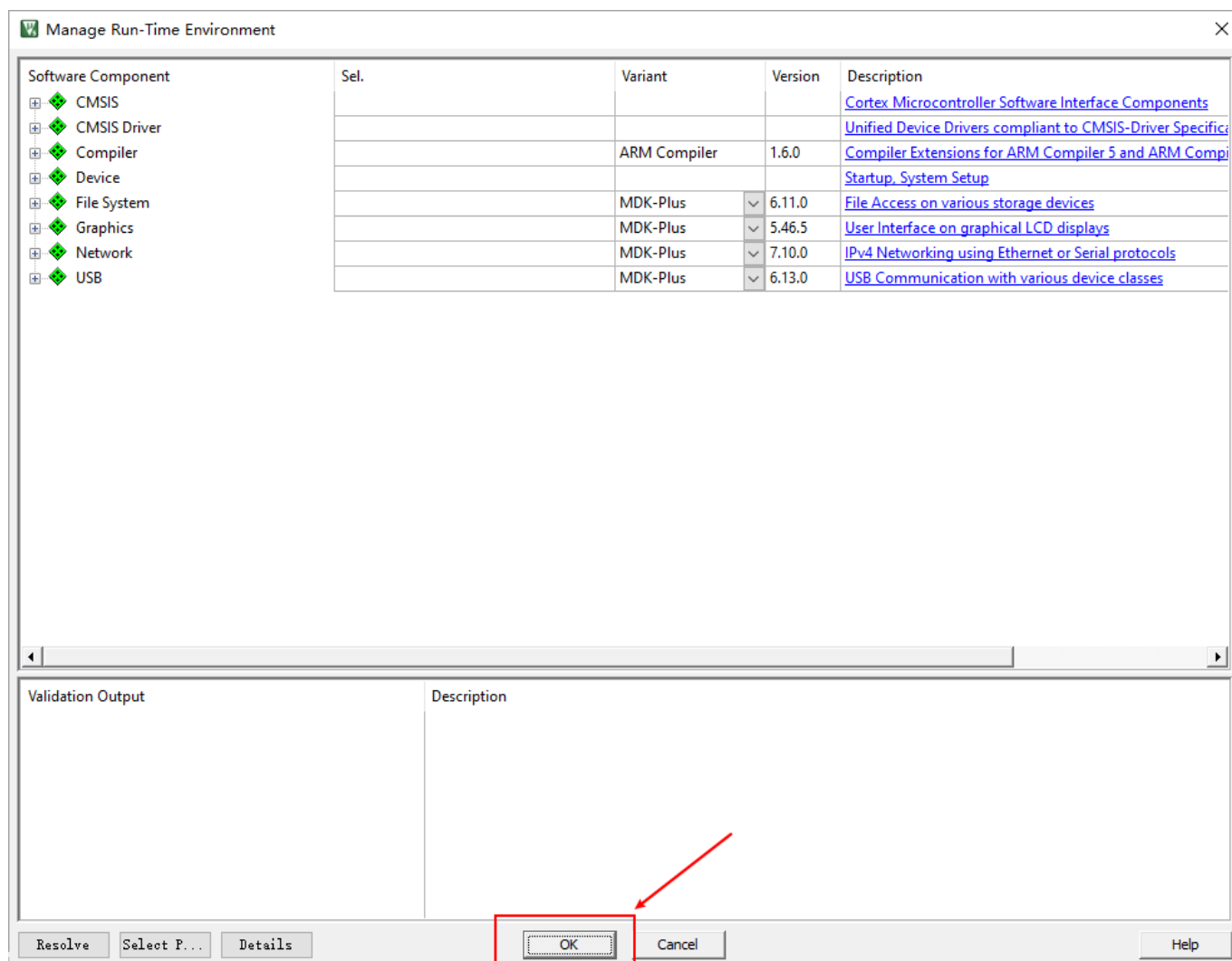
打开 MDK:



此处工程保存在 sdk 的 application 中, 命名为 template:

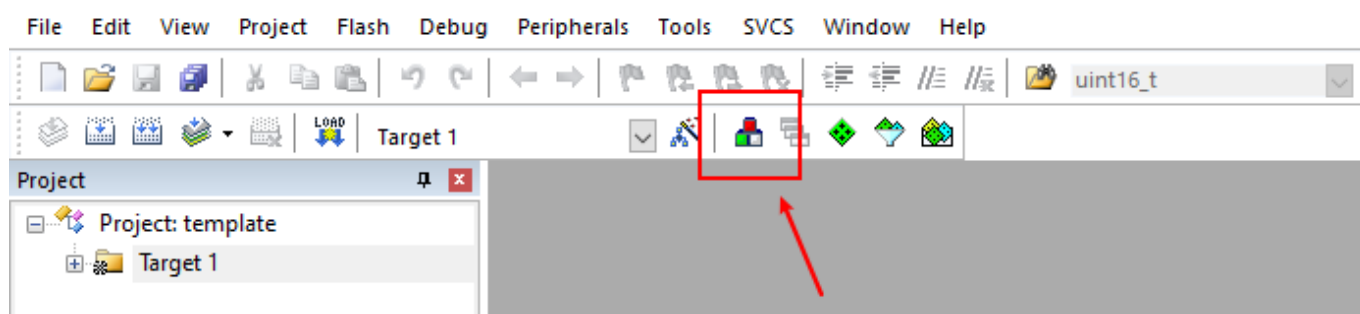


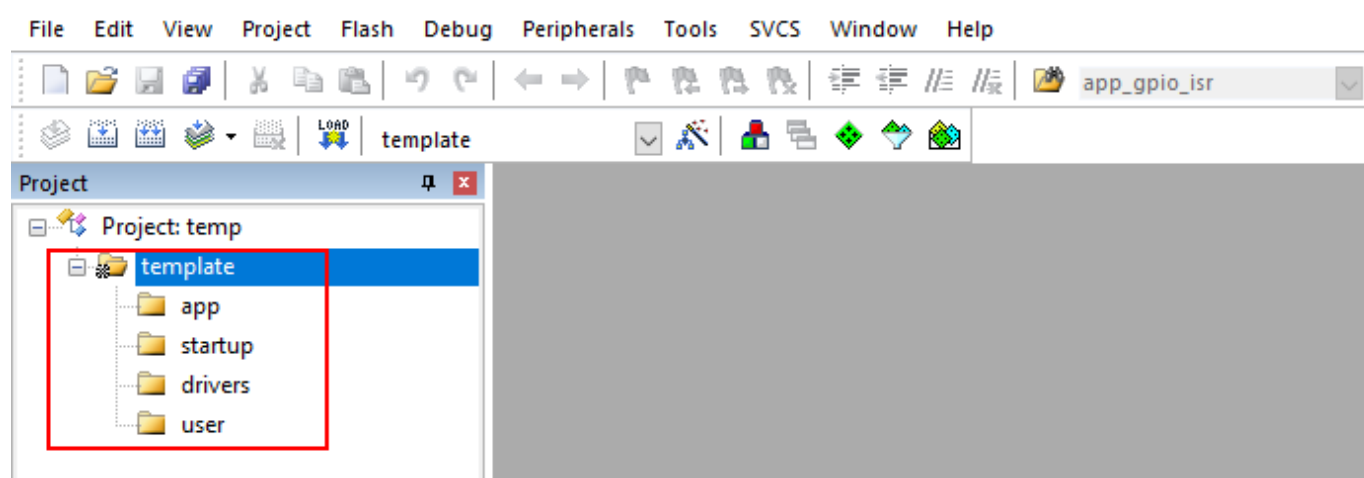
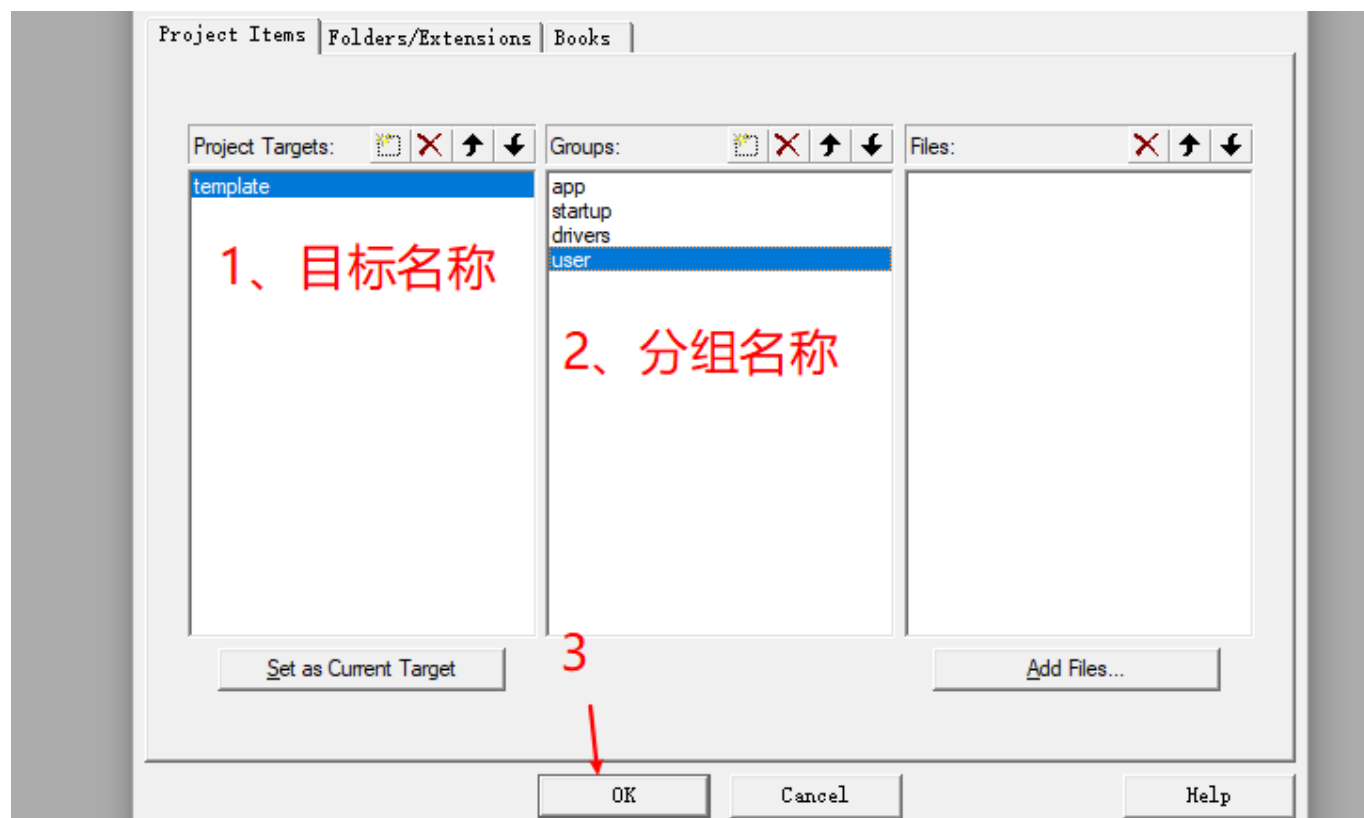




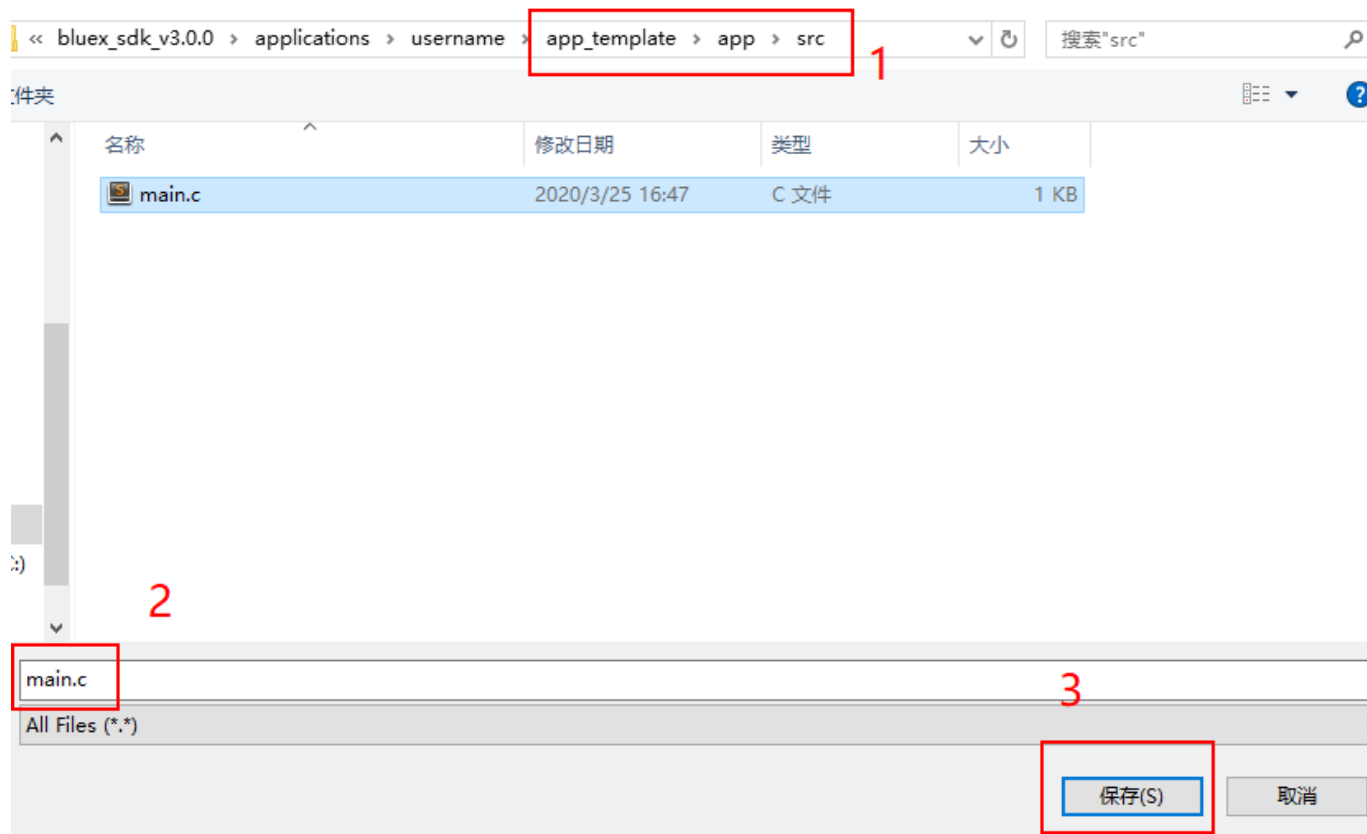
什么都不用选，直接 OK

2、规划工程目录

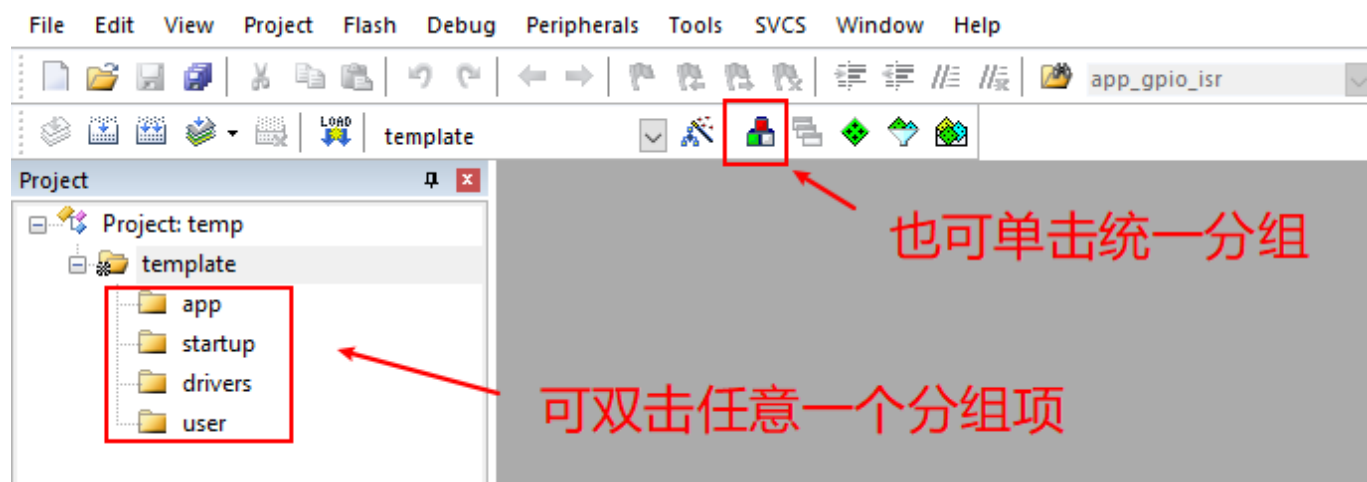


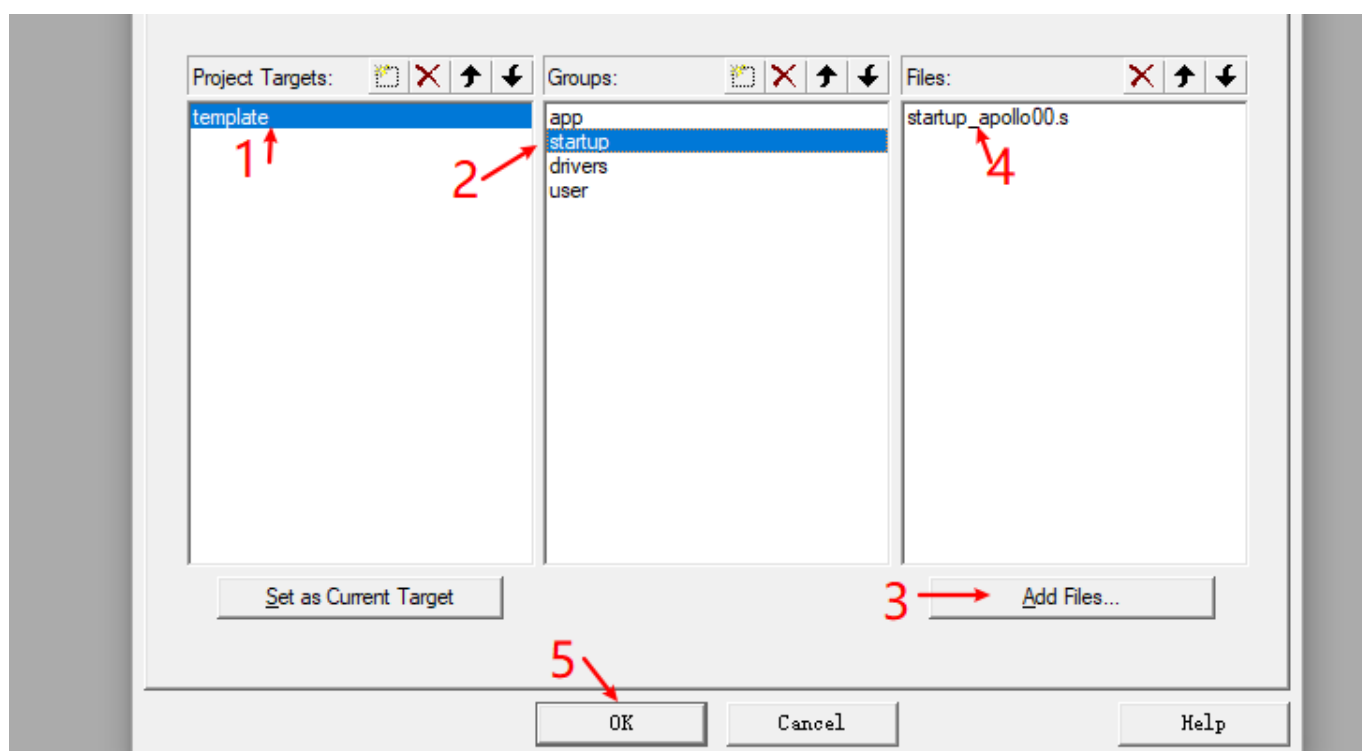
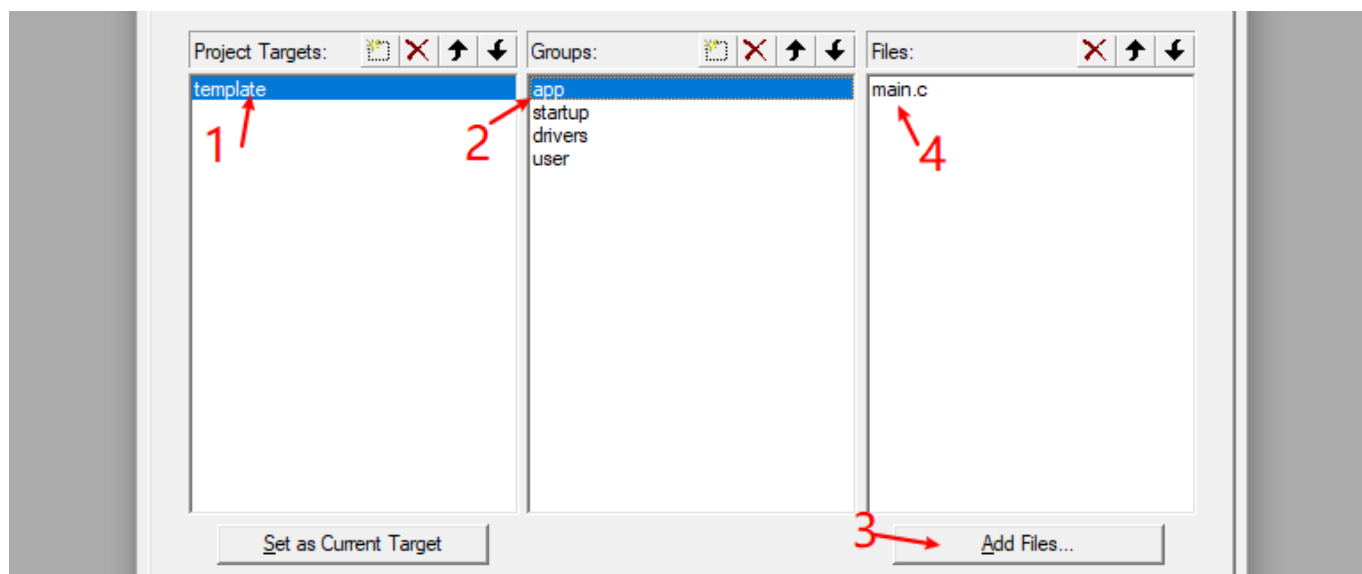


Ctrl+N, 然后 Ctrl+s, 命名为 main.c, 并将其保存在以下目录:



3、添加文件

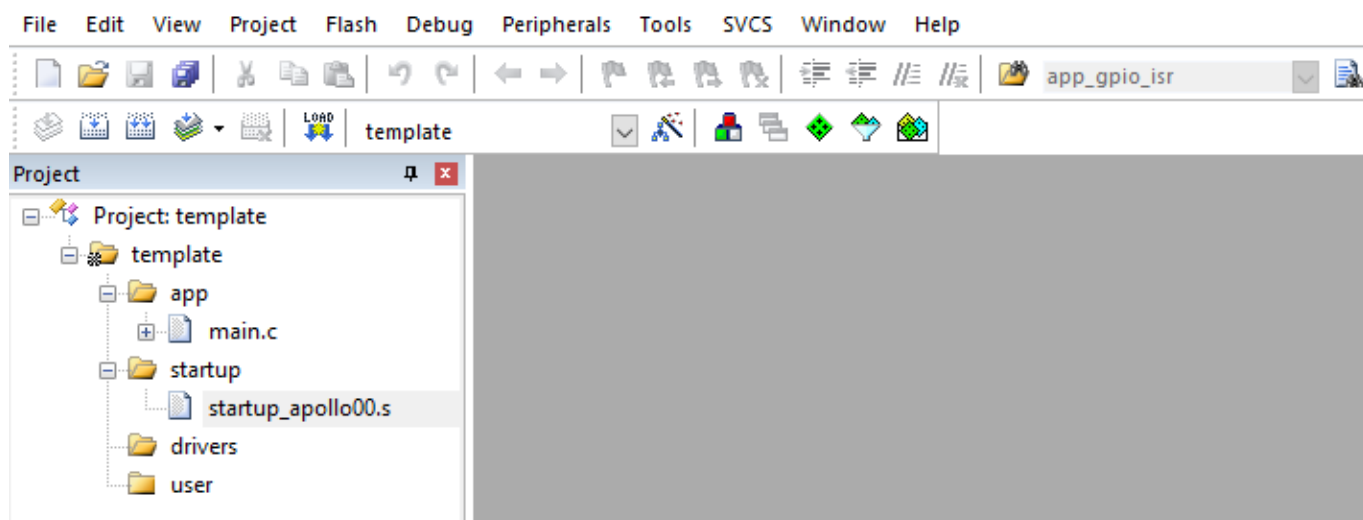




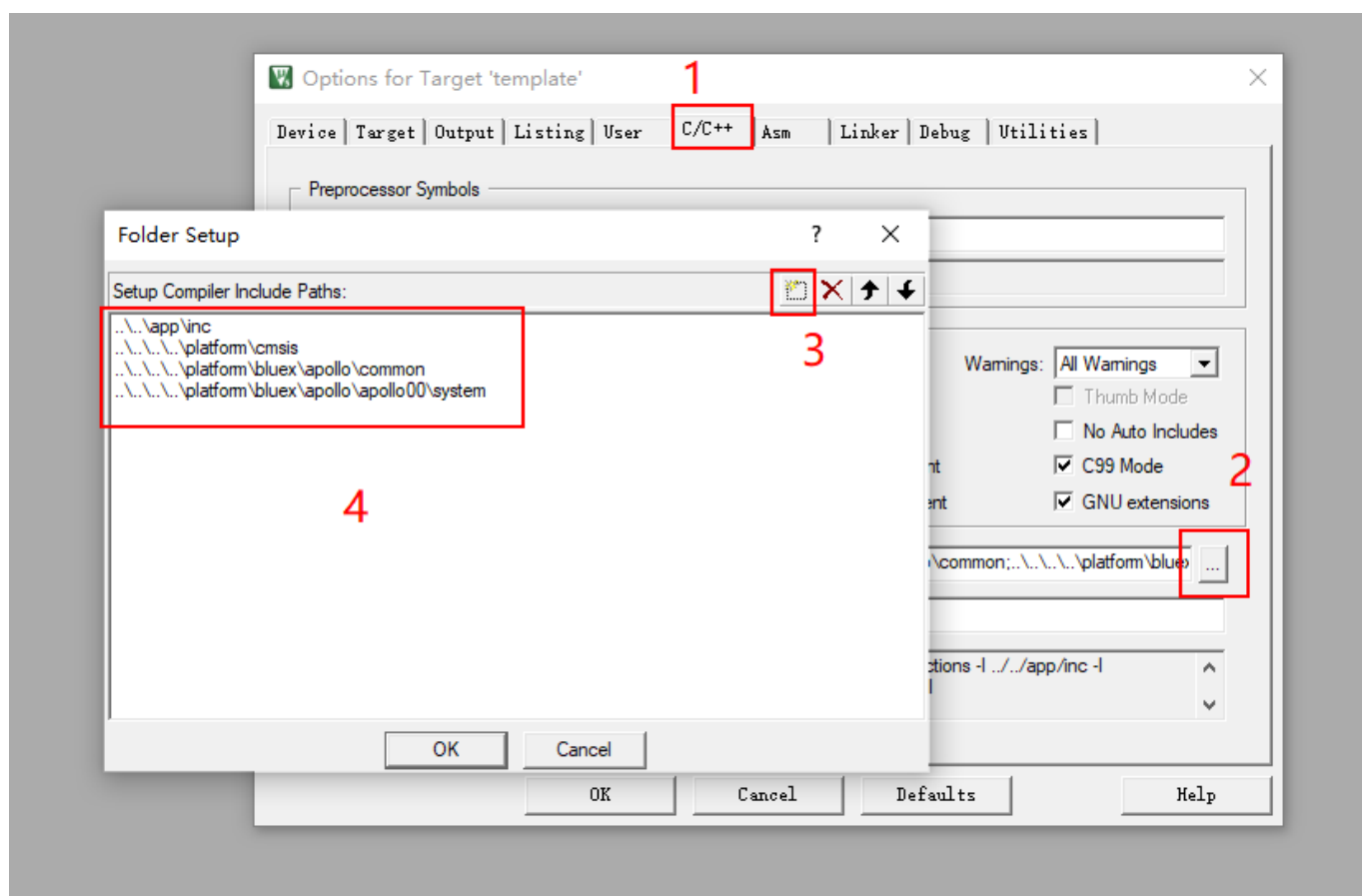
起处使用 apollo00 的启动文件作为演示，不同平台的.s 启动文件在其相对应的路径下：

bluex_sdk_v3.0.0 > platform > bluex > apollo > apollo00 > system				
名称	修改日期	类型	大小	
 apollo00_reg.h	2020/3/27 16:09	H 文件	69 KB	
 startup_apollo00.s	2020/3/27 16:33	S 文件	8 KB	

Drivers 目录暂时不需要添加文件。然后点击 OK 即可，添加后文件目录如下：



4、工程配置



点击 OK 即可，然后打开 main.c 输入以下代码：

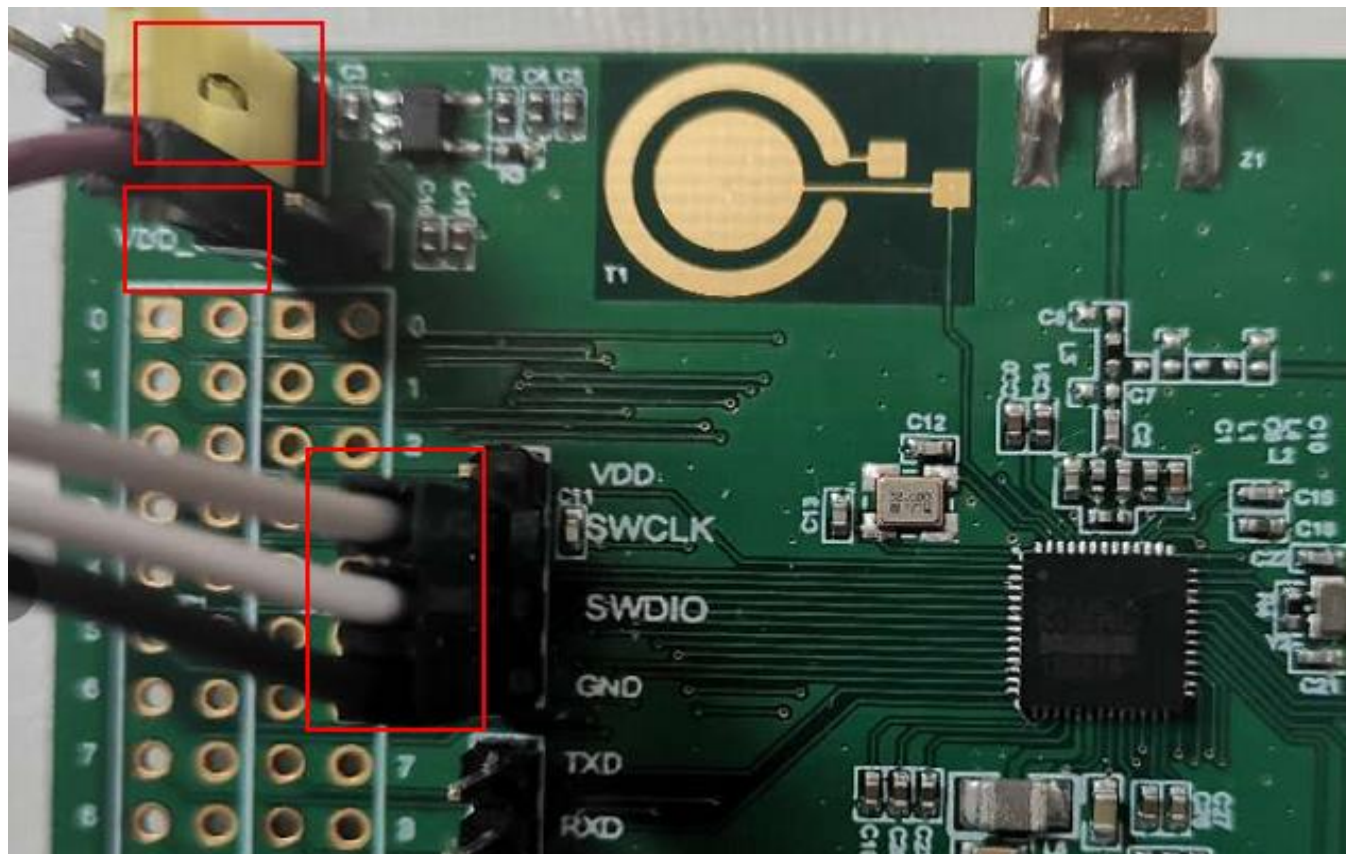
```
1. int main( void )
2. {
3.
4. }
```

可正常通过编译：

```
Build started: Project: template
*** Using Compiler 'V5.06 update 6 (build 750)', folder: 'D:\SoftWare\MDK\KEIL5\ARM\ARMCC\Bin'
Build target 'template'
compiling main.c...
linking...
Program Size: Code=344 RO-data=156 RW-data=0 ZI-data=4448
".\Objects\template.axf" - 0 Error(s), 0 Warning(s).
Build Time Elapsed: 00:00:00
```

第五章 固件烧录

1、硬件连接



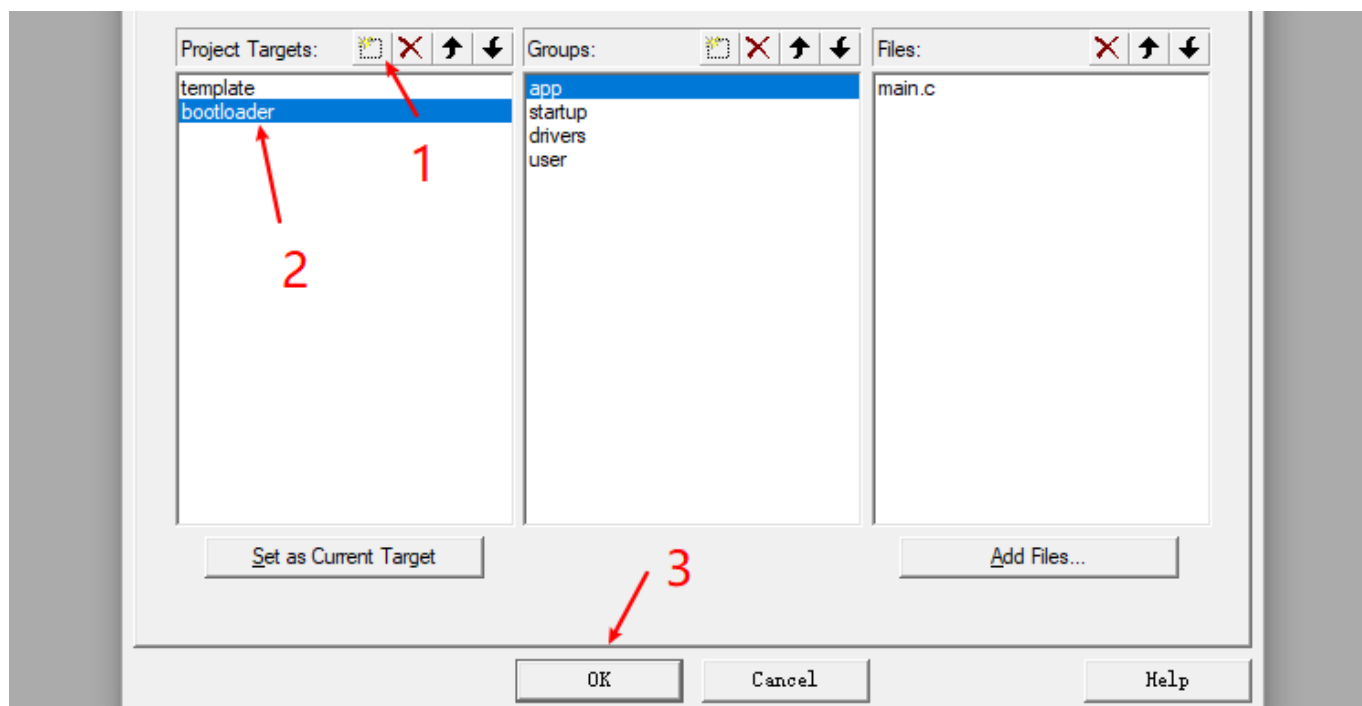


图中紫色-VDD、黑色-GND、白色-SWCLK、灰色-SWDIO。

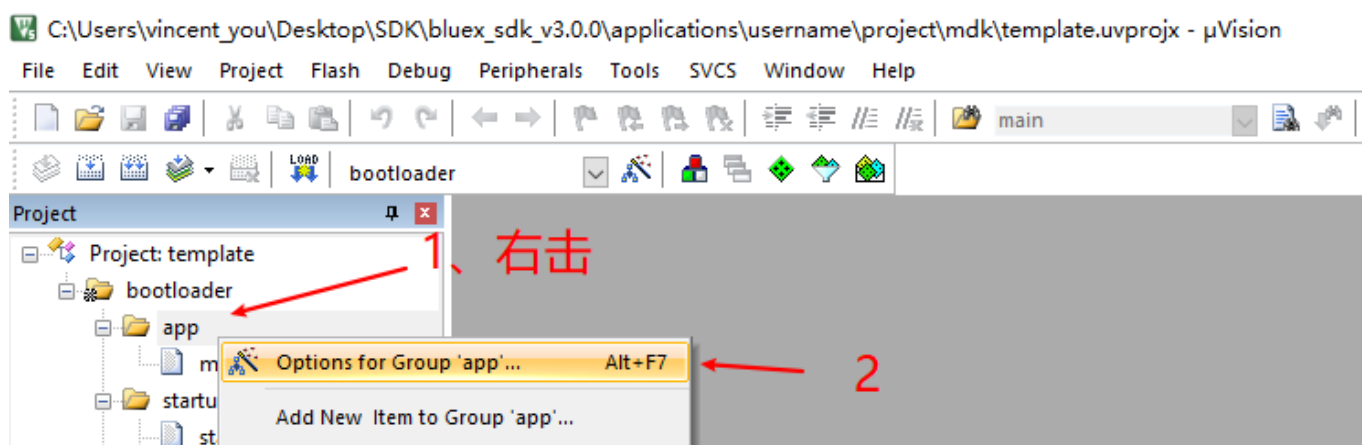
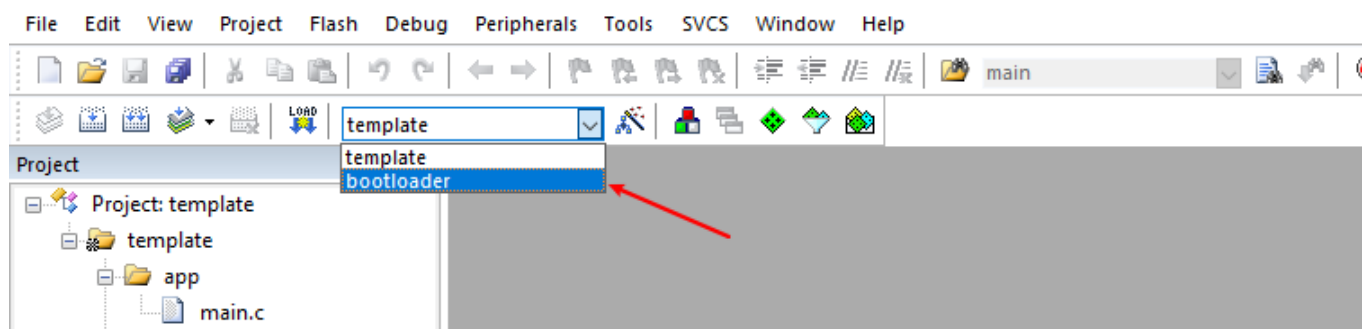
注意:Jlink 的版本不同，VDD 有时 1 引脚，有时在 2 引脚，图中接入的为 2 引脚，用户请自行判断 VDD 接入点，建议使用外部供电，然后共地。

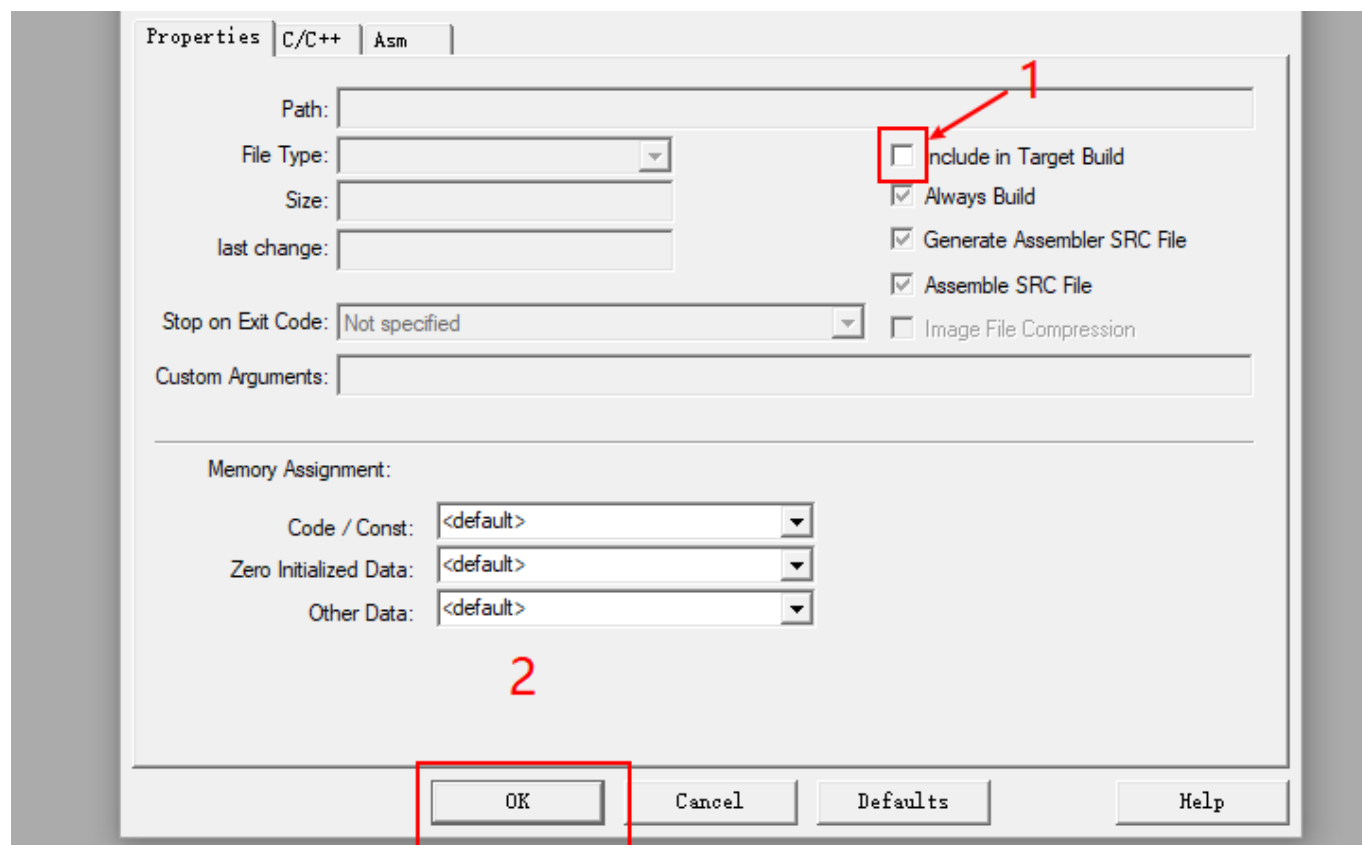
2、MKD 直接下载

以下载 bootloader 作为演示，因为后面的例程都是需要烧录 bootloader 才能正常运行的（SDK 中 example 的外设例程全都配置好 bootloader 的烧录方式）。首先新建一个目标名称：

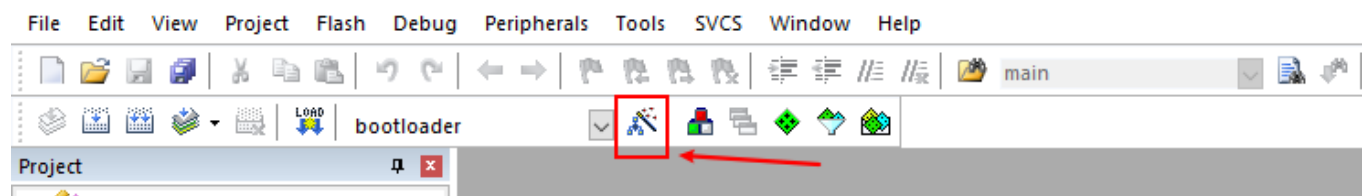
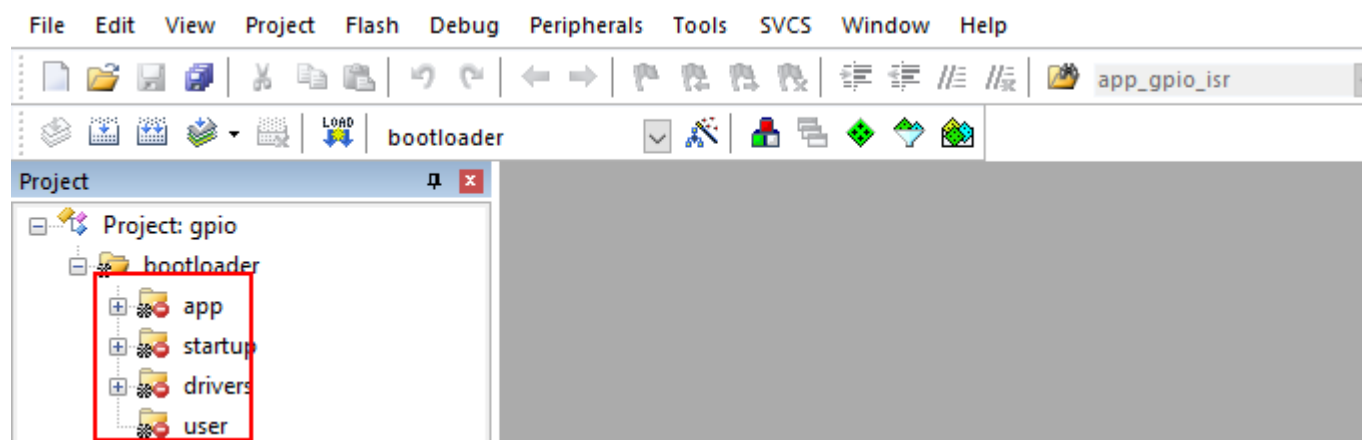


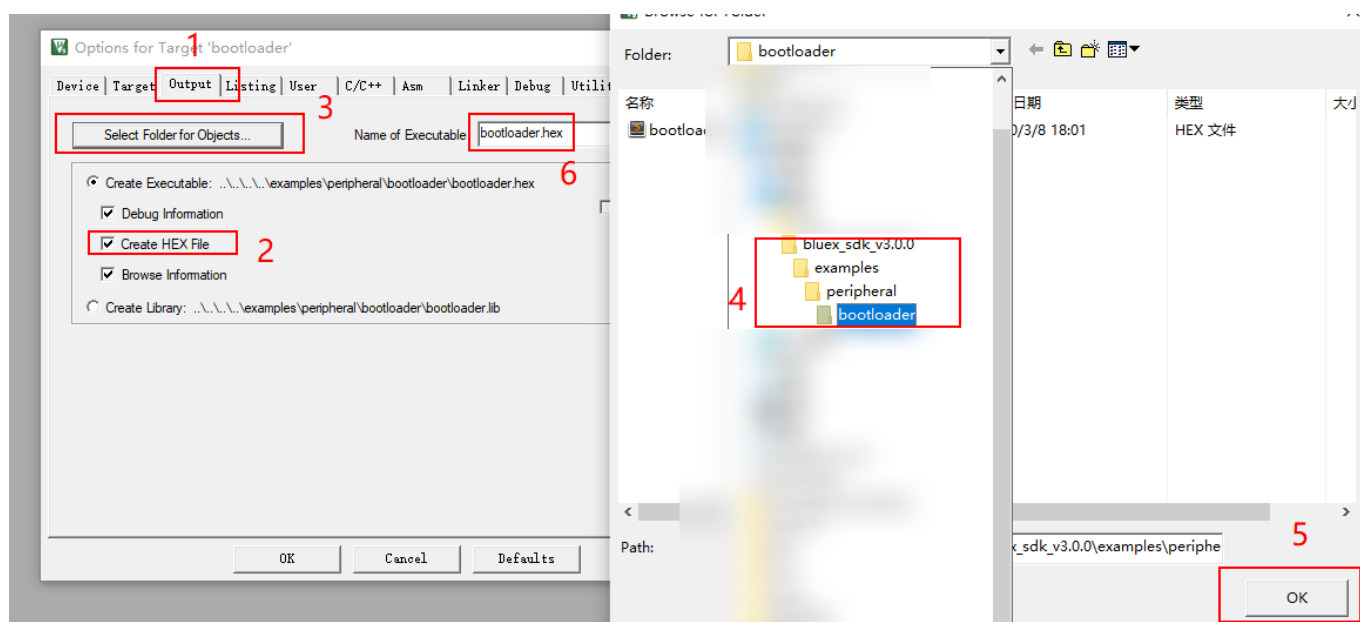
选中 bootloader 目标:





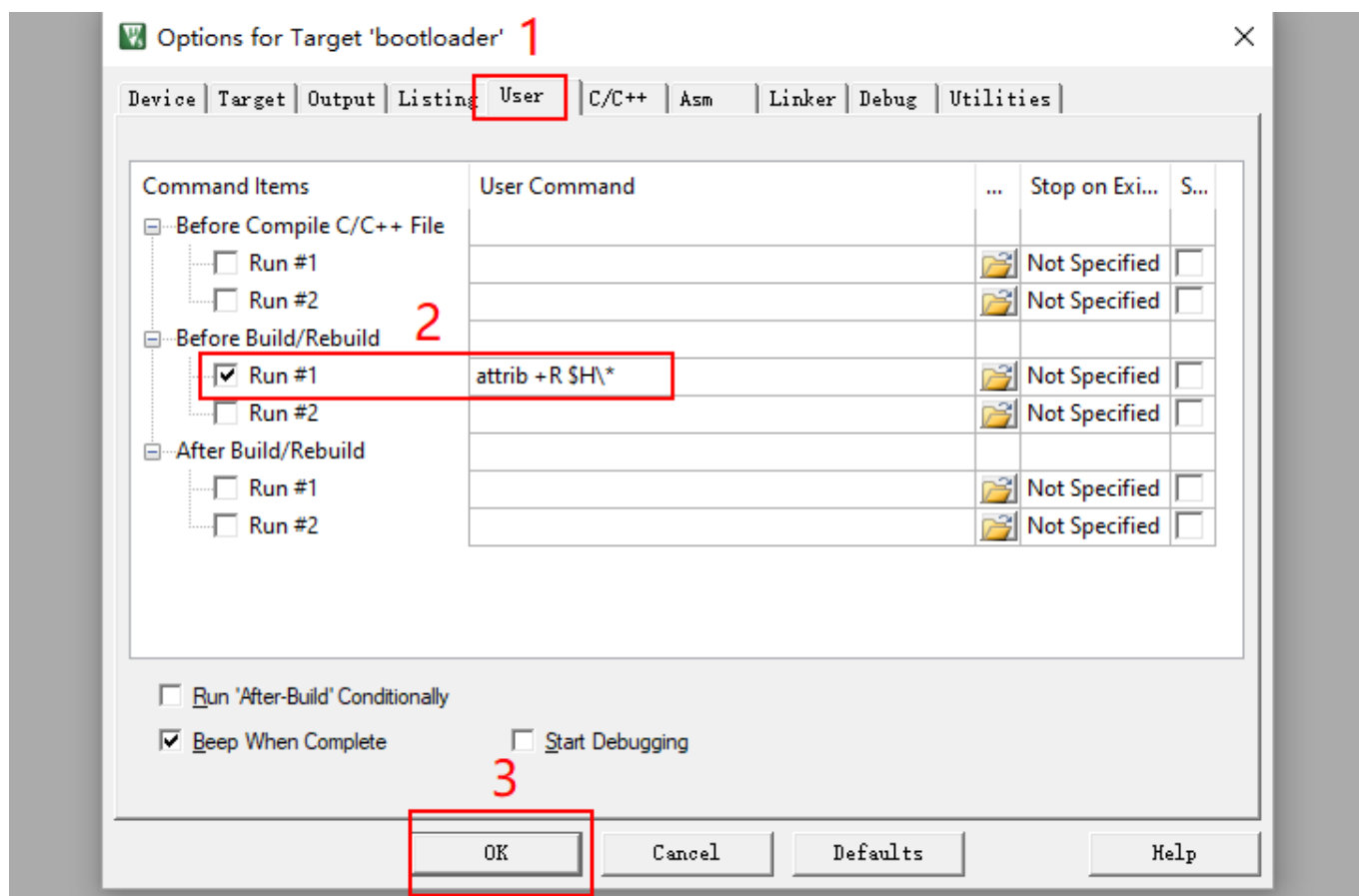
同理去掉所有：



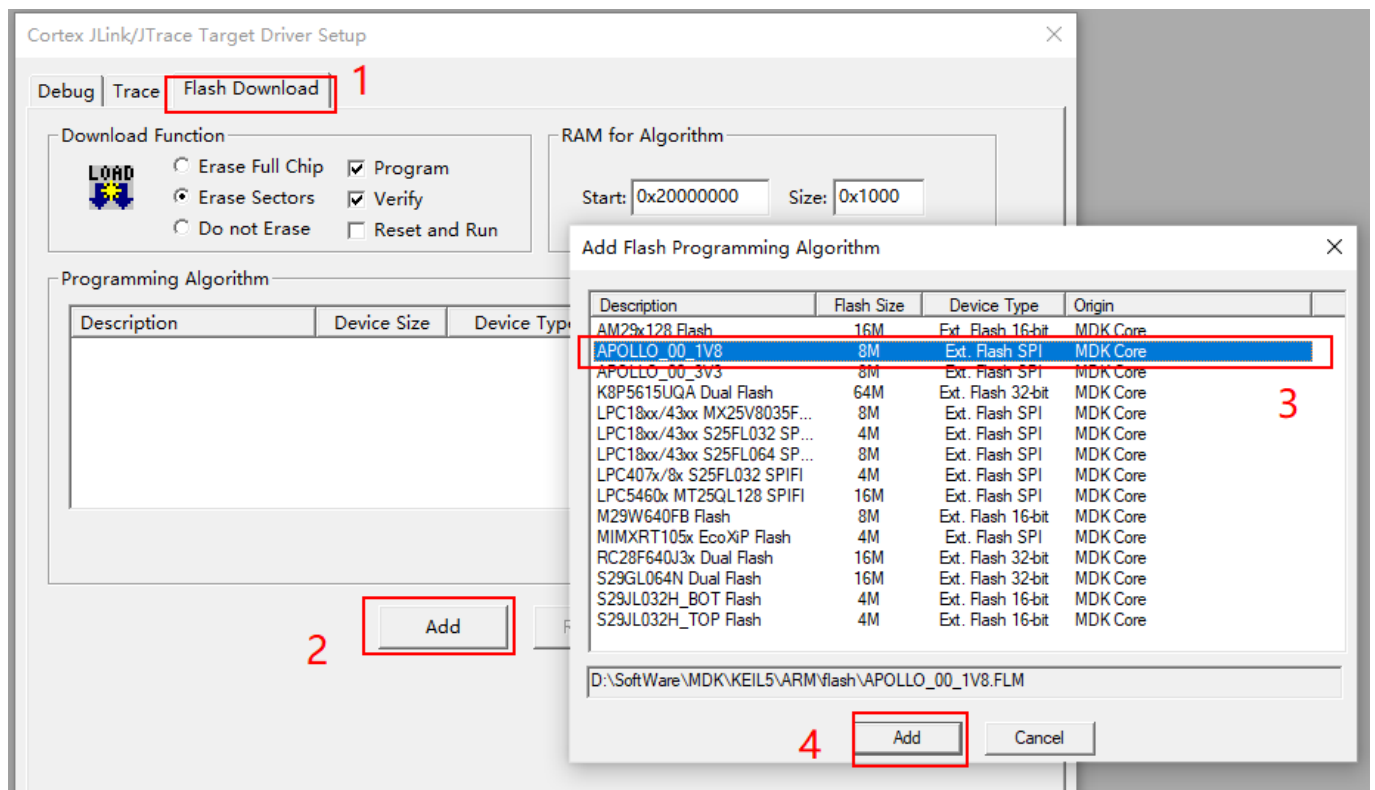
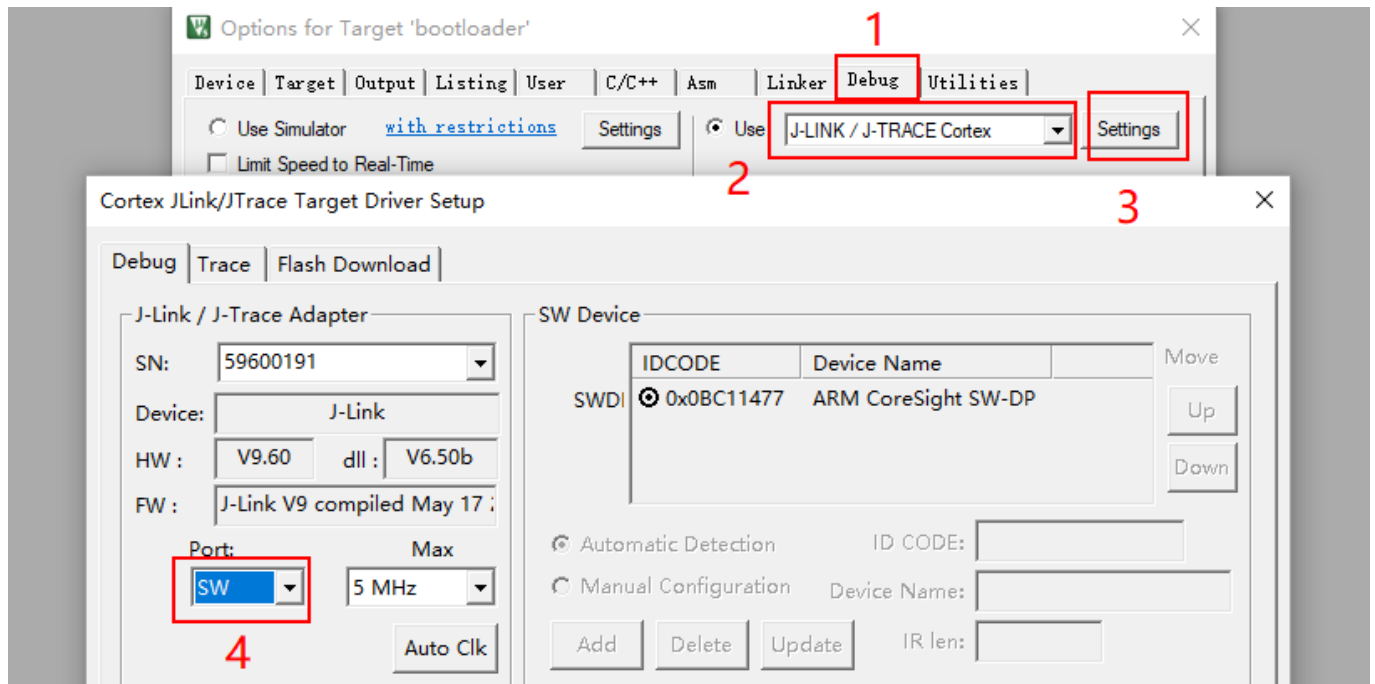


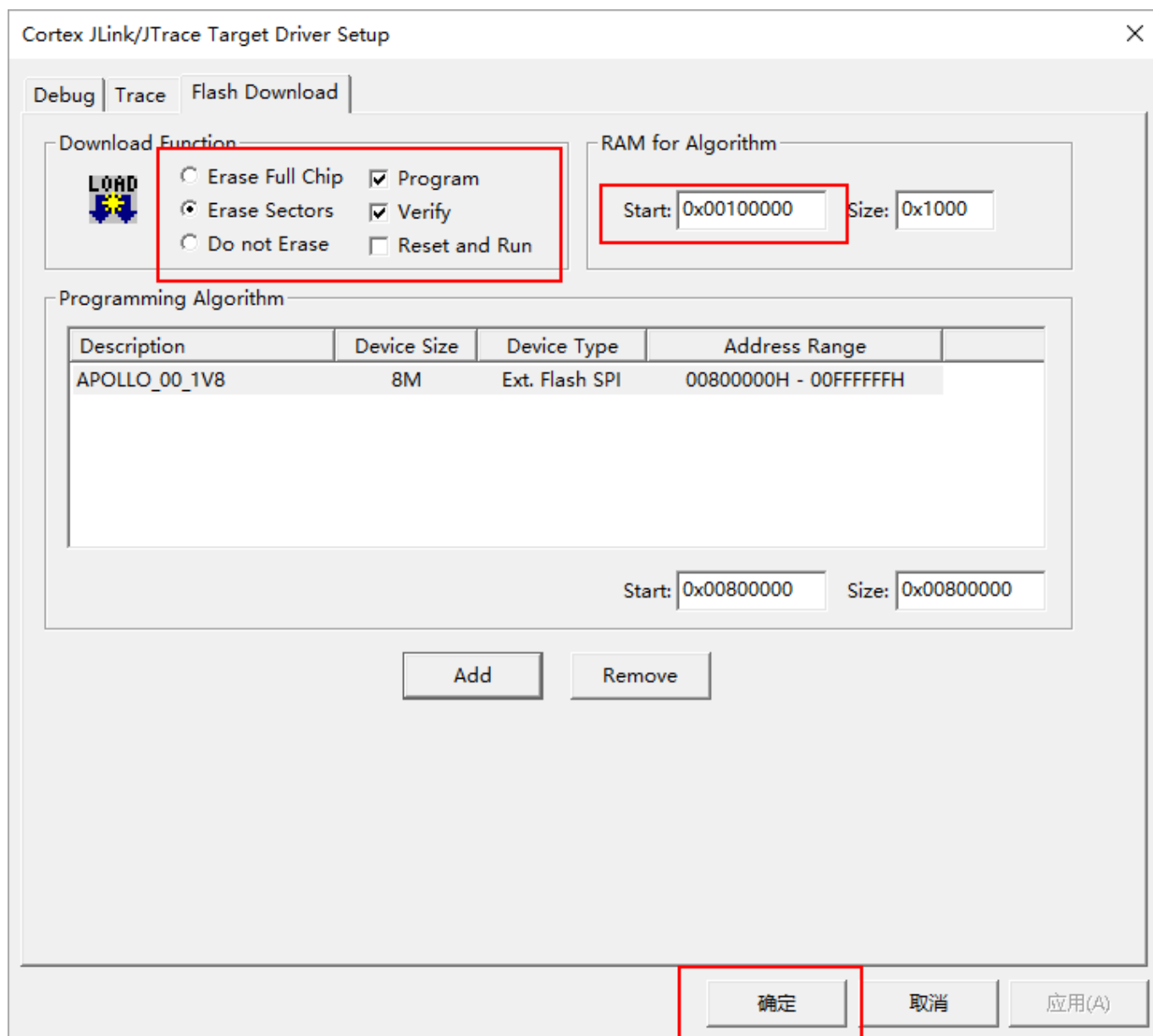
注意图中红框第 4 步，选择的路径必须跟图中所示一致。

注意图中红框第 6 步，写入的名字必须跟 bootloader 文件夹中 hex 文件的文件名一致，注意把后缀.hex 也写上。

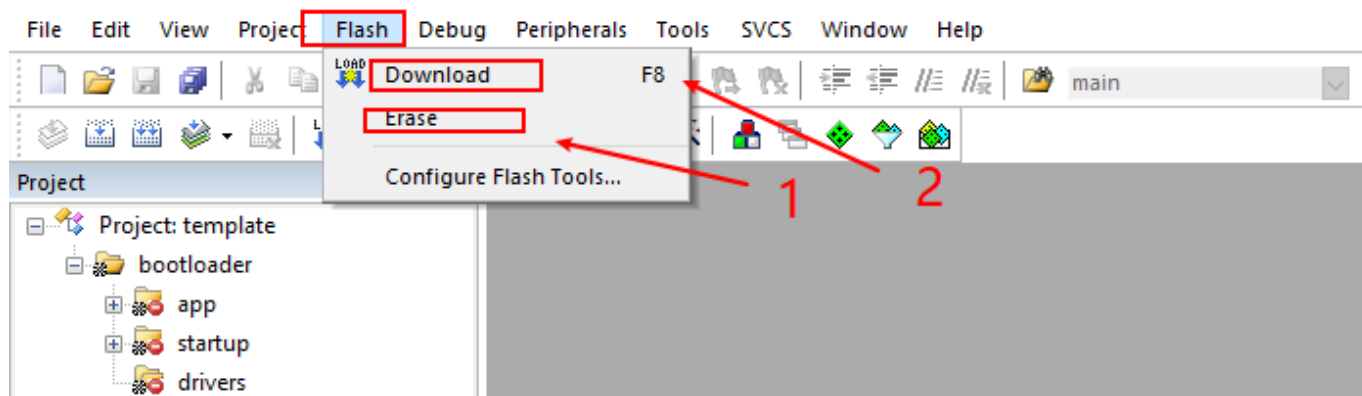


输入此命令 “attrib +R \$H*”





为确保烧录的是此 bootloader，前线擦除芯片，然后再烧录。



先擦除，后烧写。

3、J-Flash 下载

把目录下的 BlueX 文件夹和 JLinkDevices.xml 复制到 JLink 目录下，如图所示：

K > **bluex_sdk_v3.0.0 > tools > prog_tool_v2 >**

名称	修改日期	类型	大小
BlueX	2020/3/25 17:08	文件夹	
JLinkDevices.xml	2019/9/27 8:59	XML 文档	1 KB
ReadMe.txt	2019/9/27 8:59	文本文档	1 KB

目 录 下 的

北 > SEGGER

名称	修改日期	类型
BlueX	2020/3/13 11:04	文件夹
Devices	2020/3/13 11:03	文件夹
Doc	2020/3/13 11:03	文件夹
ETC	2020/3/13 11:03	文件夹
GDBServer	2020/3/13 11:03	文件夹
RDDI	2020/3/13 11:03	文件夹
Samples	2020/3/13 11:03	文件夹
USBDriver	2020/3/13 11:03	文件夹
JFlash.exe	2019/7/12 23:44	应用程序
JFlashLite.exe	2019/7/12 23:44	应用程序
JFlashSPI.exe	2019/7/12 23:44	应用程序
JFlashSPI_CL.exe	2019/7/12 23:44	应用程序
JLink.exe	2019/7/12 23:44	应用程序
JLink_x64.dll	2019/7/12 23:45	应用程序扩展
JLinkARM.dll	2019/7/12 23:44	应用程序扩展
JLinkConfig.exe	2019/7/12 23:44	应用程序
JLinkControlPanel.html	2019/5/23 23:37	HTML 文档
JLinkDevices.xml	2019/9/27 8:59	XML 文档
JLinkDLLUpdater.exe	2019/7/12 23:44	应用程序
JLinkGDBServer.exe	2019/7/12 23:44	应用程序

4、常见错误

第六章 点亮 LED

如果还没看第四、第五章节，建议先查看第四、第五章节内容。

1、新建工程

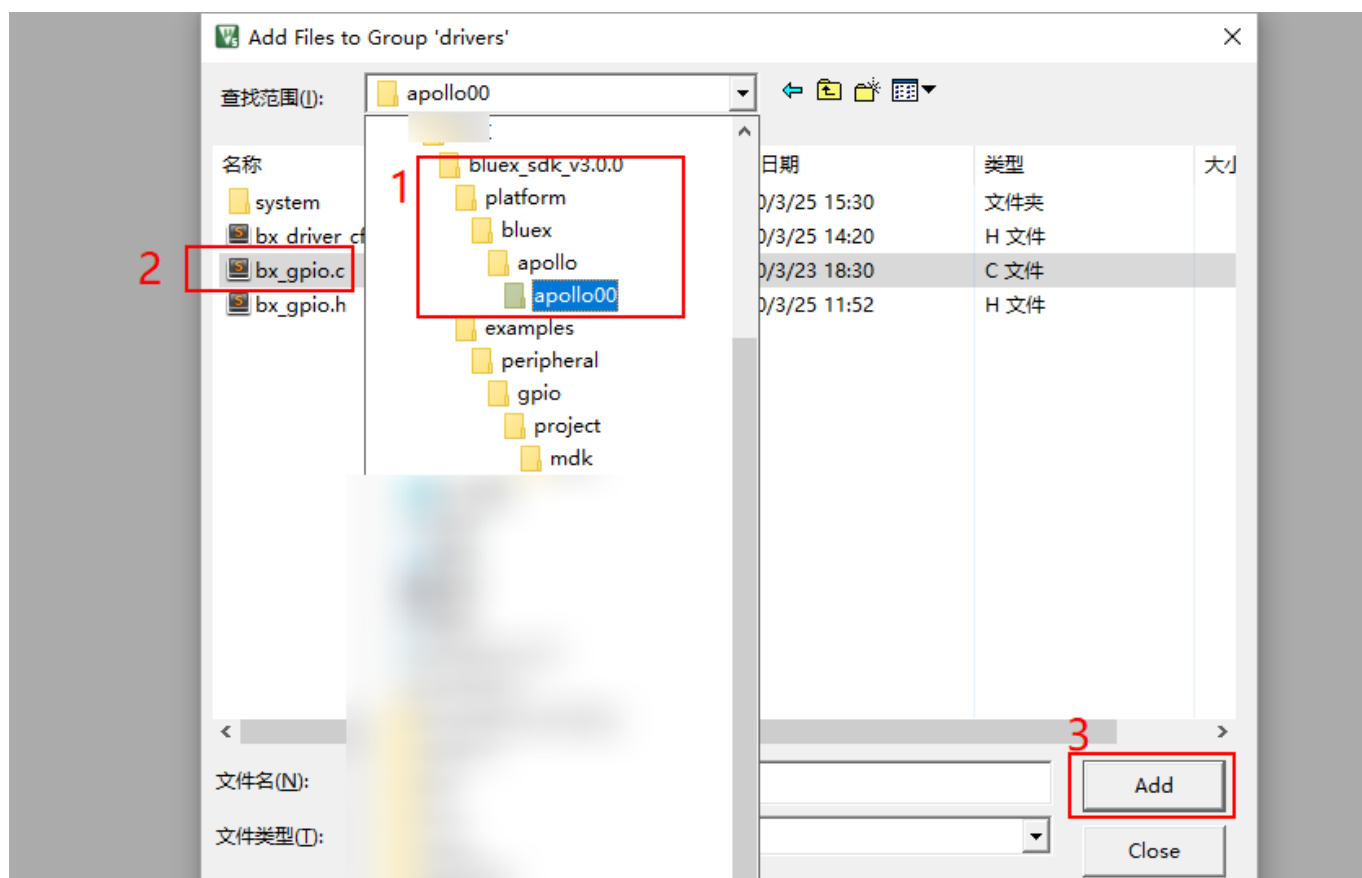
新建工程与第 4 章基本一致，只是路径跟名称稍微修改了一下，直接复制一份，将工程名称修改为 gpio 即可。

SDK > bluex_sdk_v3.0.0 > examples > peripheral > **gpio** > project > mdk

名称	修改日期	类型	大小
gpio.uvoptx	2020/3/25 18:04	UVOPTX 文件	11 KB
gpio.uvprojx	2020/3/25 18:04	Keil uVision5 Project	36 KB

修改名称即可

添加 GPIO 文件：



2、代码编写

```

1. void user_delay( uint32_t val )
2. {
3.     for( uint32_t i = 0; i < val; i++ )
4.         for( uint32_t j = 0; j < 5000; j++ );
5. }
6.
7. void output_test( void )
8. {
9.     uint32_t pin_mark = GPIO_PIN_2 | GPIO_PIN_3;
10.    gpio_clk_set();
11.    gpio_pin_cfg_output( pin_mark );
12.    while( 1 ) {
13.        gpio_pin_toggle( pin_mark );
14.        user_delay( 500 );
15.    }
16. }
17. /** -----
18.  * @brief :
19.  * @note :
20.  * @param :
21.  * @retval :
22.  * -----*/
23. int main( void )
24. {
25.     /* peripheral init */
26.
27.     /* board */

```

```

28.
29.  /* component */
30.
31.  /* device */
32.
33.  /* user */
34.
35.  __DMB();
36.  SCB->VTOR = 0x00803000;
37.  __DSB();
38.
39.  output_test()
40.
41.  while( 1 );
42. }

```

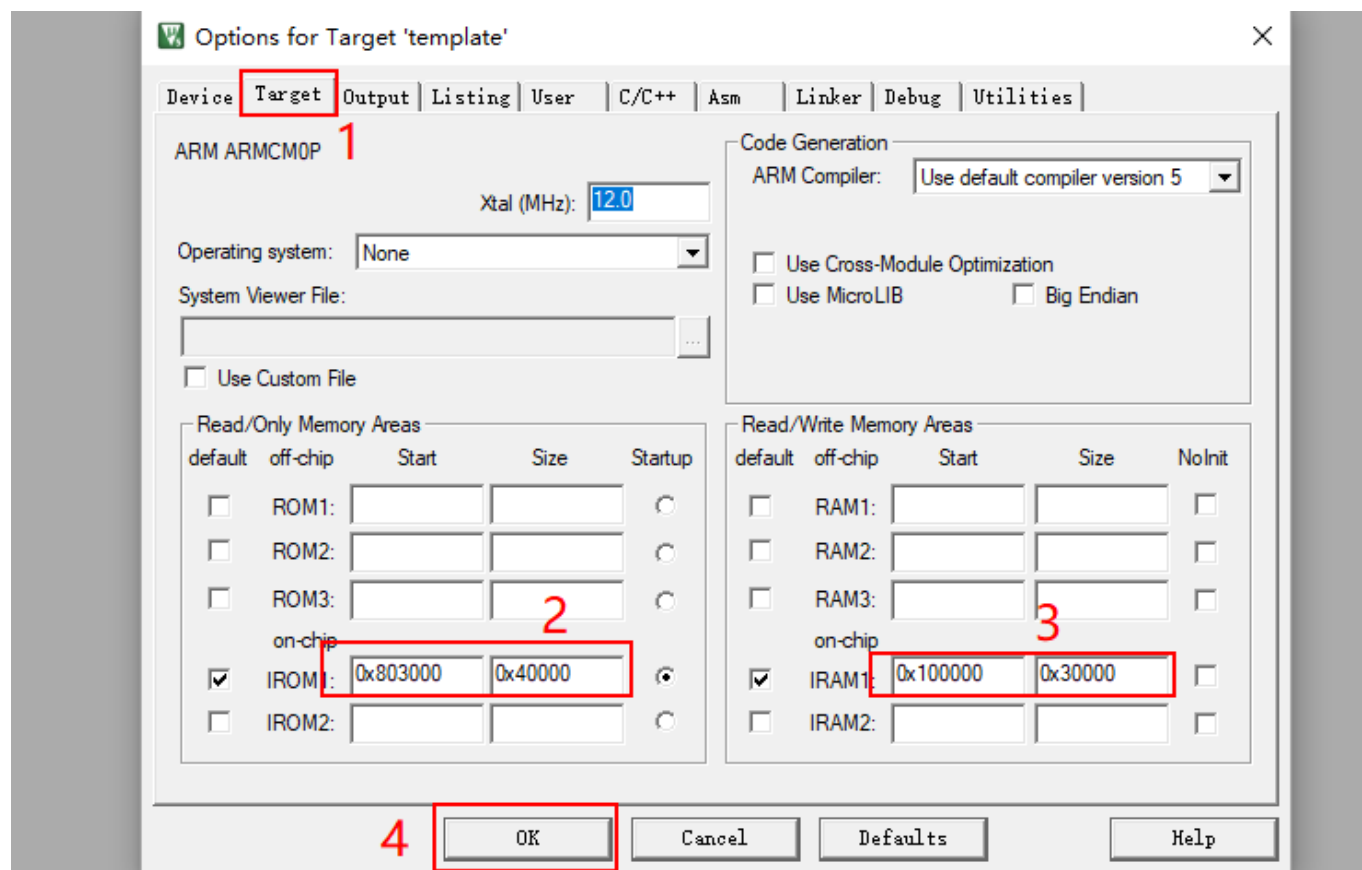
此处只需编写初级演示代码，具体 API 及寄存器放在后 4、5、6 小结上，详细可参考 example 例程。

3、实例演示

3.1 硬件连接

开发板正常连接 Jlink。

3.2 软件配置



3.3 演示步骤

- 打开 SDK 中的 gpio 的例程

> bluemicro_sdk_v3.0.0 > examples > peripheral > gpio > project > mdk

名称

gpio.uvoptx
gpio.uvprojx

- 编译文件
- 按本章硬件连接与软件配置要求正确操作
- 按第五章方式烧录固件
- 重启开发板，观察实验现象与代码是否符合

4、芯片 GPIO 介绍

4.1 寄存器

5、API 简介

原型	
参数	
功能	

第七章 调试输出

1、新建工程

2、代码编写

3、实例演示

3.1 硬件连接

3.2 软件配置

3.3 演示步骤

第八章 BLE 连接

1、BLE 连接流程

2、硬件设计

3、代码编写

4、实例演示

5、API 简介

原型	
----	--

参数	
功能	
响应事件	
触发消息	

只需说明复杂函数即可，功能函数需要简单说明调用函数后会触发什么动作/事件/消息，主要针对 CEVA 类。

6、相关文档

只需说明相关文档名称即可，如果可以最好具体到文档章节。

第九章 BLE 数据交换

- 1、BLE 数据交换方式
- 2、硬件设计
- 3、代码编写
- 4、实例演示
- 5、API 简介

原型	
参数	
功能	
响应事件	
触发消息	

只需说明复杂函数即可，功能函数需要简单说明调用函数后会触发什么动作/事件/消息，主要针对 CEVA 类。

6、相关文档

只需说明相关文档名称即可，如果可以最好具体到文档章节。

第十章 BLE 从机

- 1、基础知识
- 2、硬件设计
- 3、代码编写
- 4、实例演示
- 5、API 简介

原型	
参数	
功能	
响应事件	
触发消息	

只需说明复杂函数即可，功能函数需要简单说明调用函数后会触发什么动作/事件/消息，主要针对 CEVA 类。

6、相关文档

只需说明相关文档名称即可，如果可以最好具体到文档章节。

第十一章 BLE 主机

1、基础知识

2、硬件设计

3、代码编写

4、实例演示

5、API 简介

原型	
参数	
功能	
响应事件	
触发消息	

只需说明复杂函数即可，功能函数需要简单说明调用函数后会触发什么动作/事件/消息，主要针对 CEVA 类。

6、相关文档

只需说明相关文档名称即可，如果可以最好具体到文档章节。

第十二章 BLE 主从一体

1、基础知识

2、硬件设计

3、代码编写

4、实例演示

5、API 简介

原型	
参数	
功能	
响应事件	
触发消息	

只需说明复杂函数即可，功能函数需要简单说明调用函数后会触发什么动作/事件/消息，主要针对 CEVA 类。

6、相关文档

只需说明相关文档名称即可，如果可以最好具体到文档章节。

第十三章 BLE 组网 MESH

1、基础知识

2、硬件设计

3、代码编写

4、实例演示

5、API 简介

原型	
参数	
功能	
响应事件	
触发消息	

只需说明复杂函数即可，功能函数需要简单说明调用函数后会触发什么动作/事件/消息，主要针对 CEVA 类。

6、相关文档

只需说明相关文档名称即可，如果可以最好具体到文档章节。