

SDK3 快速入门

文档更新说明

版本	版本更新说明	负责人	校审	发布日期
V1.0	初稿 第一章 搭建开发环境 第二章 开发板介绍 第三章 SDK3 目录 第四章 新建工程 第五章 固件烧录 第六章 点亮 LED 第七章 调试输出 第八章 PEK 工程 第九章 基础 BLE 第十章 OTA 介绍	梁伟衍	耿峻峰	2020/08/05

概述	4
第一章 搭建开发环境.....	4
1、硬件工具.....	4
2、软件工具.....	4
3、安装工具.....	4
第二章 开发板介绍.....	5
1、BX2400_EVK.....	5
第三章 SDK3 目录.....	6
1.1 application.....	6
第四章 新建工程.....	7
1、新建模板.....	7
2、规划工程目录.....	10
3、添加文件.....	11
4、工程配置.....	13
第五章 固件烧录.....	14
1、硬件连接.....	14
2、MKD 直接下载	15
3、J-Flash 下载.....	20
4、常见错误.....	22
第六章 点亮 LED.....	23
1、新建工程.....	23
2、代码编写.....	23
3、实例演示.....	23
第七章 调试输出.....	24
1、串口调试.....	24
2、代码编写.....	25
3、实例演示.....	25
第八章 PEK 工程.....	26
1、指令	26
2、使用方法.....	28
第九章 基础 BLE.....	28
1、新建工程.....	28
2、实例演示.....	28
第十章 OTA 介绍	29
1、新建工程.....	29
2、实例演示.....	29

概述

本文档从搭建开发环境开始，介绍并指导安装开发需要的软硬件环境，然后从零开始新建最简单的裸机工程，逐步编写代码，在此过程中对 SDK3 的目录结构，芯片的烧录调试方式等方面进行阐述，然后介绍基础的蓝牙连接，以及 OTA 方式，让读者逐步了解 BLE 的相关操作，熟悉开发环境以及熟悉软件工具的使用。

文档从无到有，通过实验程序讲解和实验演示，让读者以最短的时间掌握 BlueX 芯片以及 SDK3 的使用方法。

第一章 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
BlueX OTA	BlueX 自研发用于 OTA 的工具

3、安装工具

3.1 安装 MDK

MDK，又称 keil uVision，是 Keil 公司开发的一个集成开发环境，其编译器、调试工具实现与 ARM 器件的完美配合。

网上有很多 MDK 的安装教程，此处不多做介绍，建议 MDK5.23 以上的版本。

3.2 安装 JFlash

JFlash 主要用于读写芯片，查看芯片的调试信息，并非必须安装，读者可以按需安装。

同样网上有很多 JFlash 的安装教程，此处不多做介绍。

3.3 安装 nRF Connect

nRF Connect 为手机端调试 BLE 的工具，可以观察到 BLE 原始的数据，也可以显示 RSSI 曲线，可以同时连接多个从机，通用性和兼容性都非常强。

nRF Connect 的安装只需打开手机应用市场，搜索 nRF Connect 字样，按提示安装即可，与普通 APP 安装方式并没有什么区别。

3.4 安装 Bluex OTA

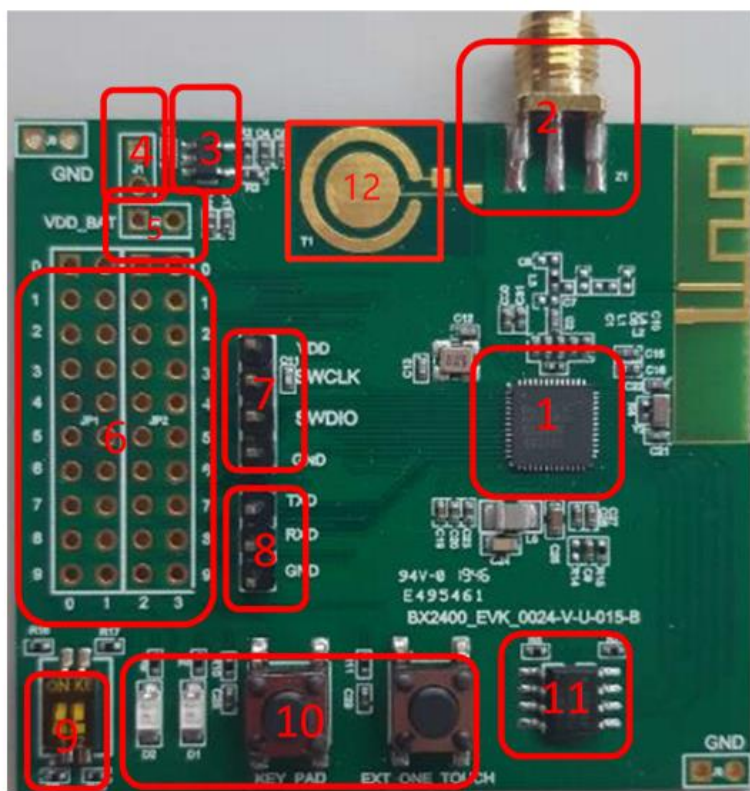
Bluex OTA 工具为联睿微自研的安卓端 APP 工具，目前仅适用于 Android

用户可以通过以下二维码，下载 APP，然后安装即可。



第二章 开发板介绍

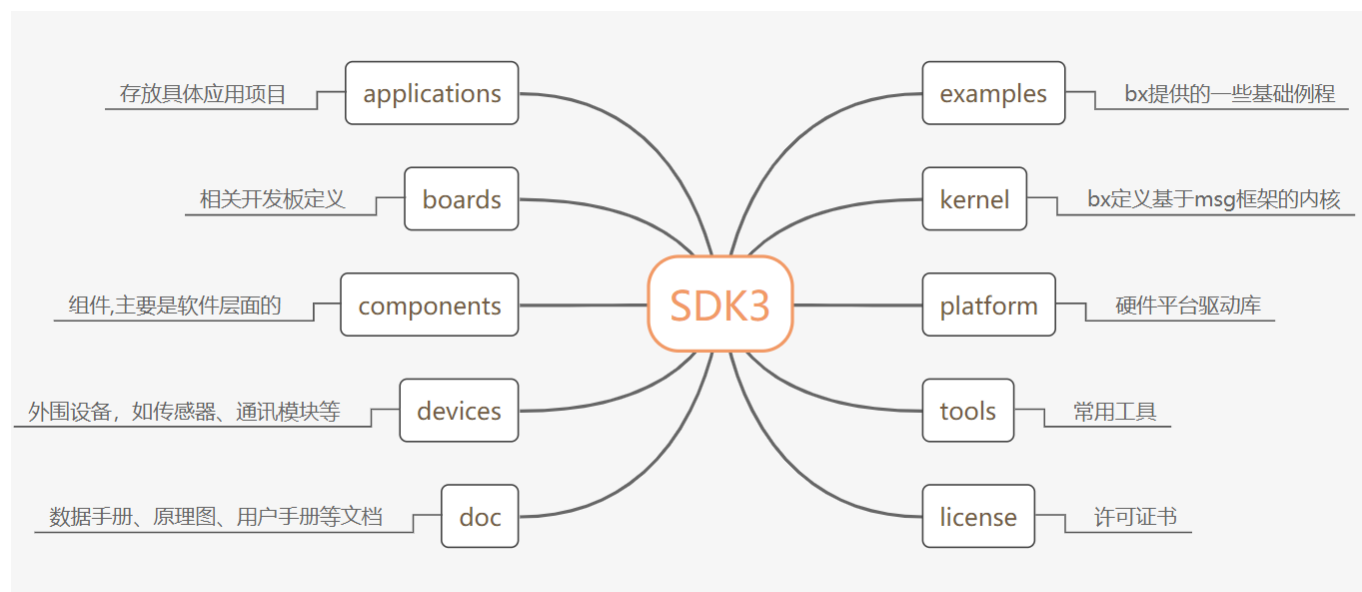
1、BX2400_EVK



- 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。

第三章 SDK3 目录



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

```

bluex
company_name
company_name - 副本
  
```

- 更改企业名称为 bluex001

```

bluex
bluex001
company_name
  
```

- 更改项目名称为 ble_lock

```

< > bluex_sdk_v3.0.0 > applications > bluex001
  
```

名称

```

ble_lock
  
```

```

bluex_sdk_v3.0.0 > applications > bluex001 > ble_lock > project > mdk
  
```

名称

```

ble_lock.uvoptx
ble_lock.uvprojx
  
```

- 最后打开工程项目即可。

第四章 新建工程

本章将从零开始，介绍如何基于 MDK 新建一个可以在 blueX 硬件平台上运行的工程，包括一些必要文件的移动、文件目录说明以及工程目录的介绍。

1、新建模板

1.1 相关文件

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：

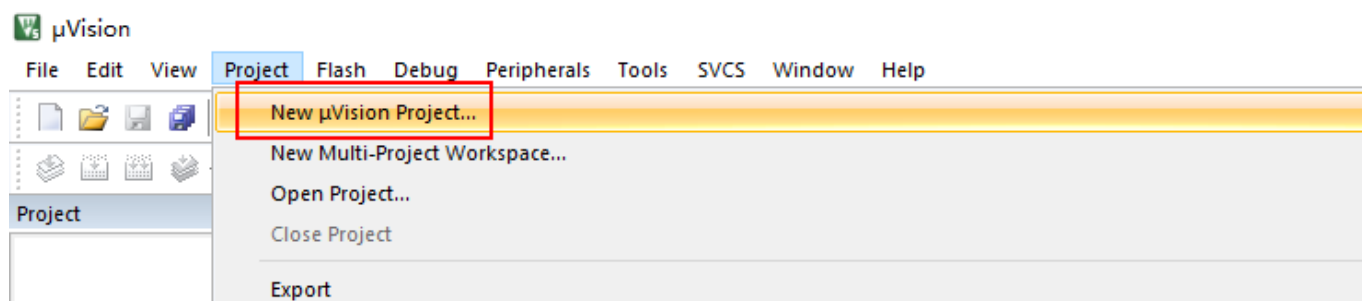
电脑 > ... > KEIL5 > ARM > Flash >				
名称	修改日期	类型	大小	
S29GL064INXZ	2020/3/9 17:55	文件夹		
SP29JL032H	2020/3/9 17:55	文件夹		
AM29F160DB.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29F160DT.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29F320DB.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29F320DBx2.FLX	2015/7/8 16:30	FLX 文件	14 KB	
AM29F320DT.FLX	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	

1.2 文件目录规划

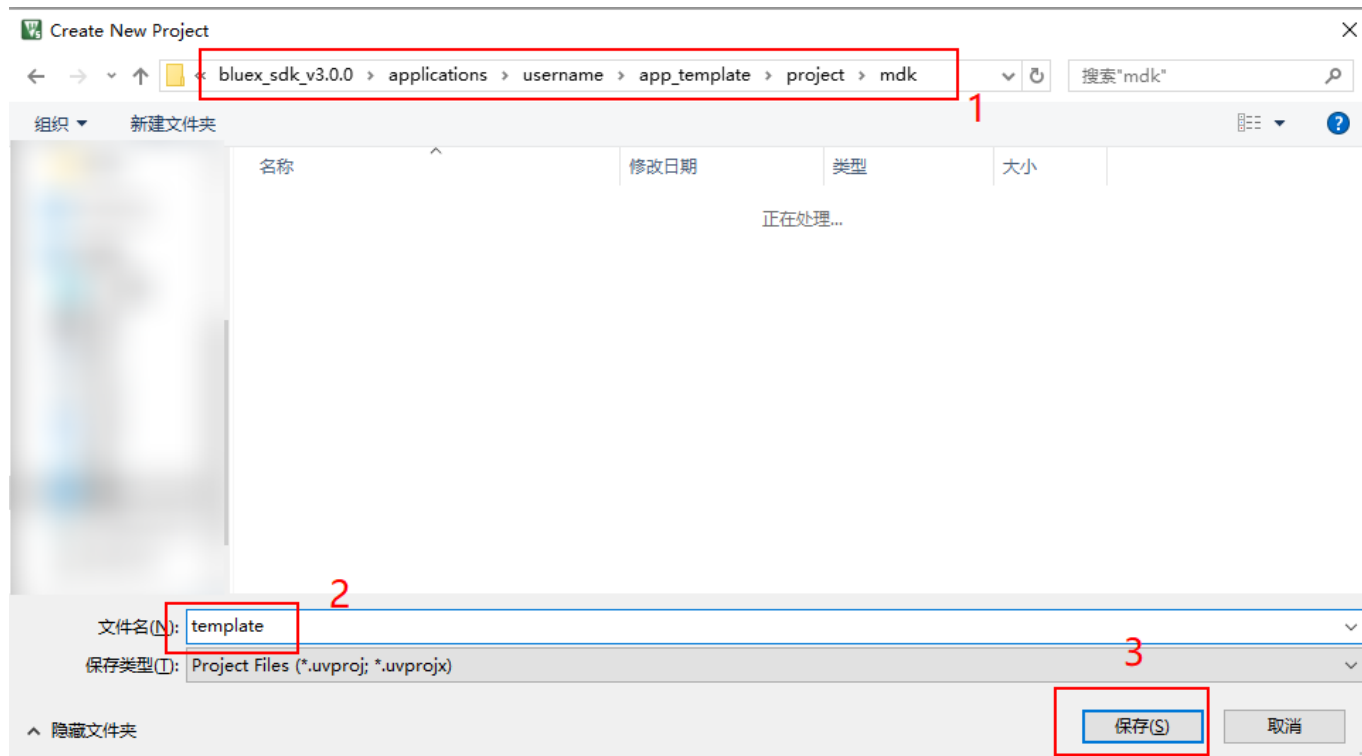
移动完相关文档后，可在任意路径上（建议在 SDK 根目录下的 application/username/app_templat/路径下），新建以下几个文件夹：

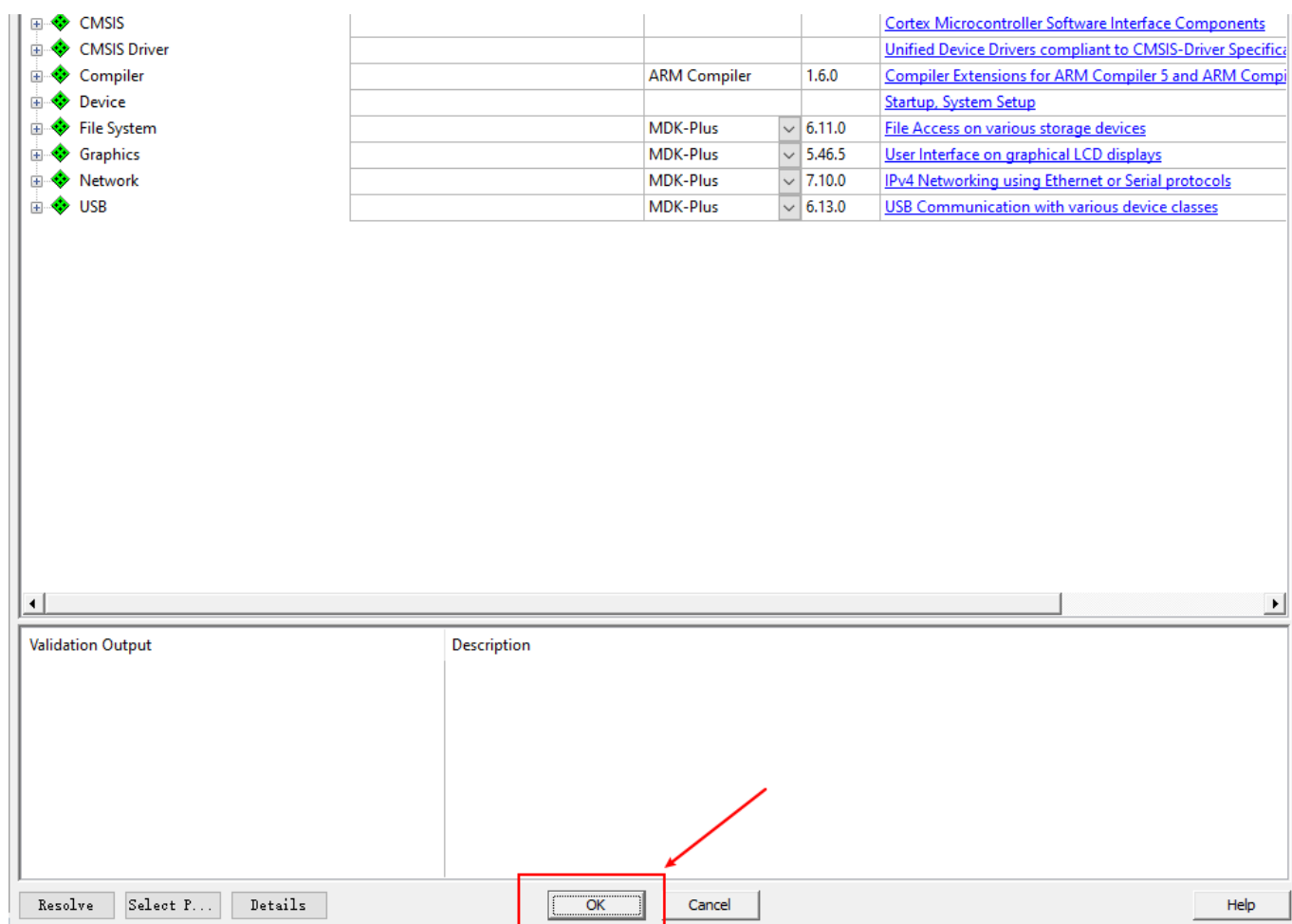
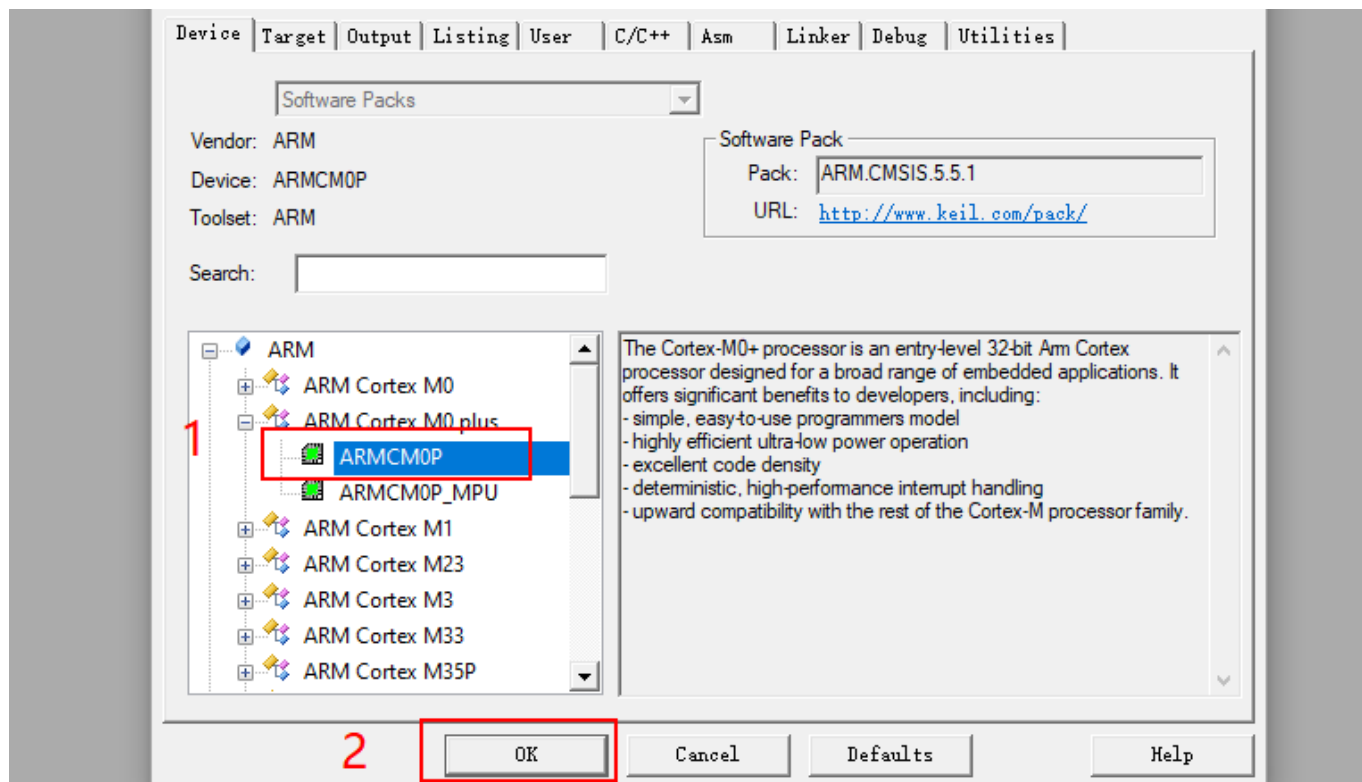


打开 MDK:



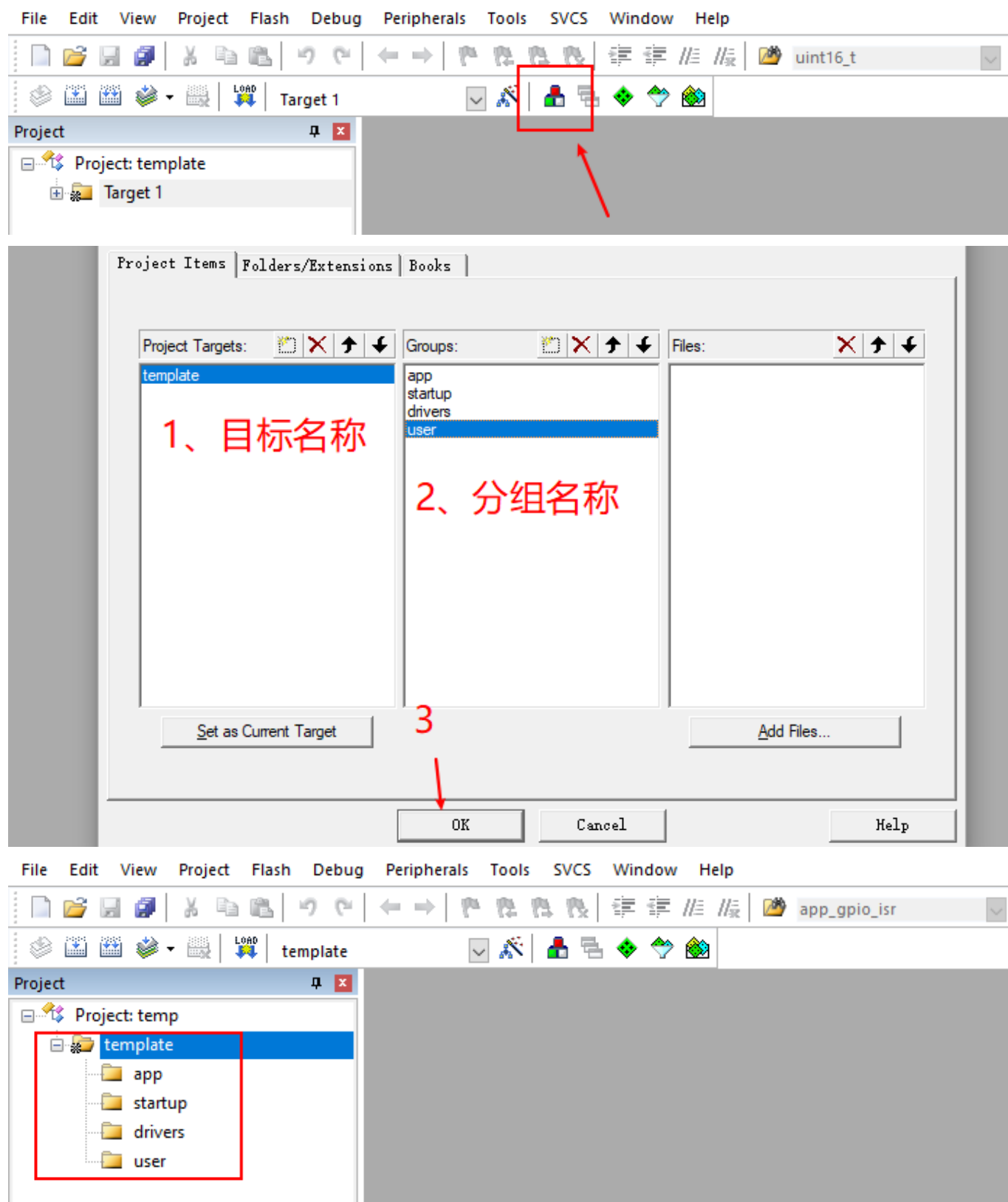
此处工程保存在 sdk 的 application 中, 命名为 template:



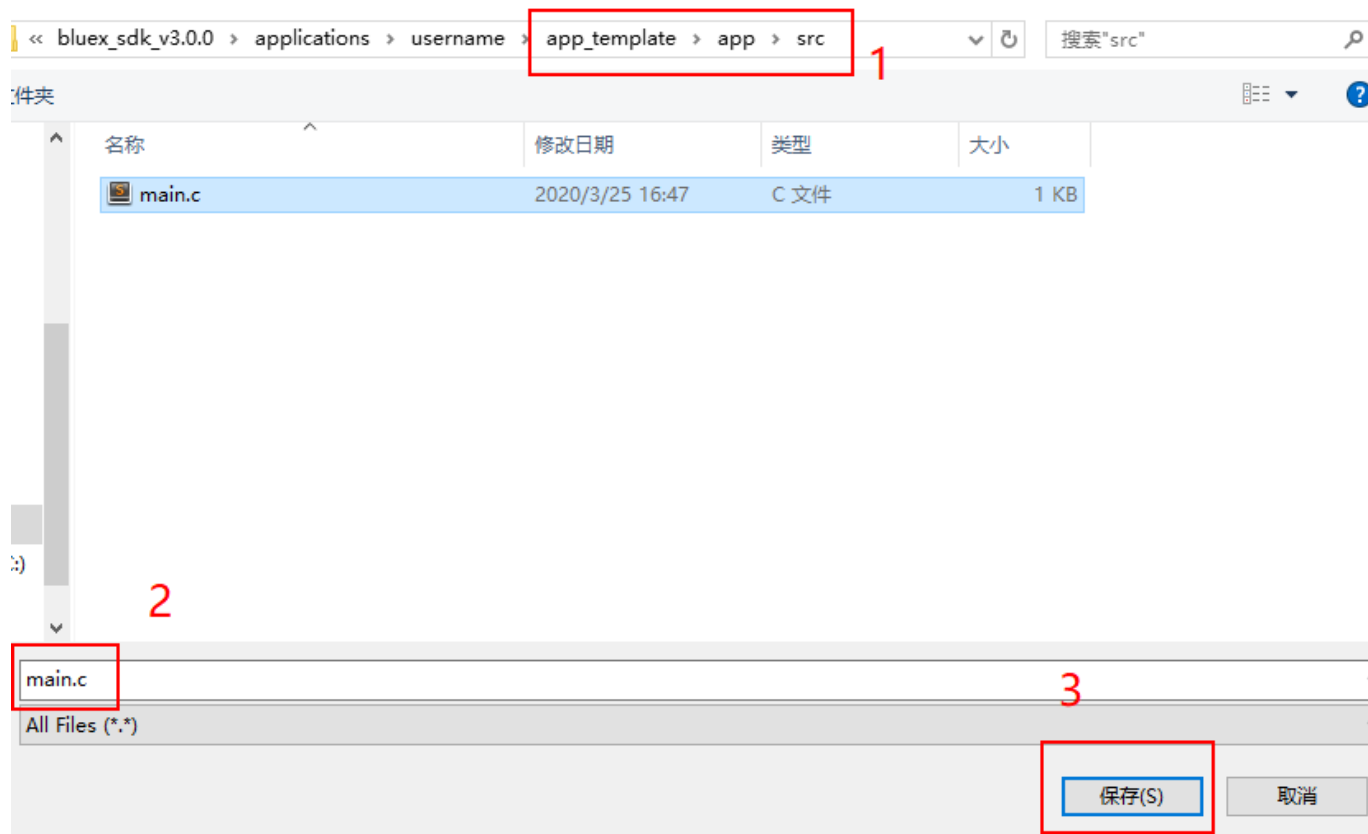


什么都不用选，直接 OK

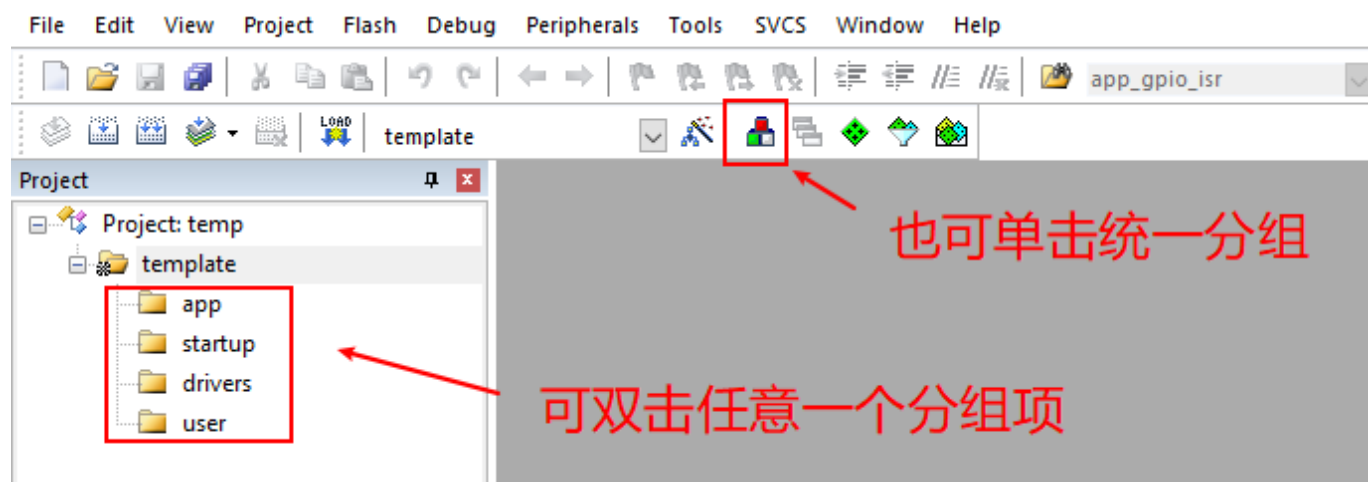
2、规划工程目录

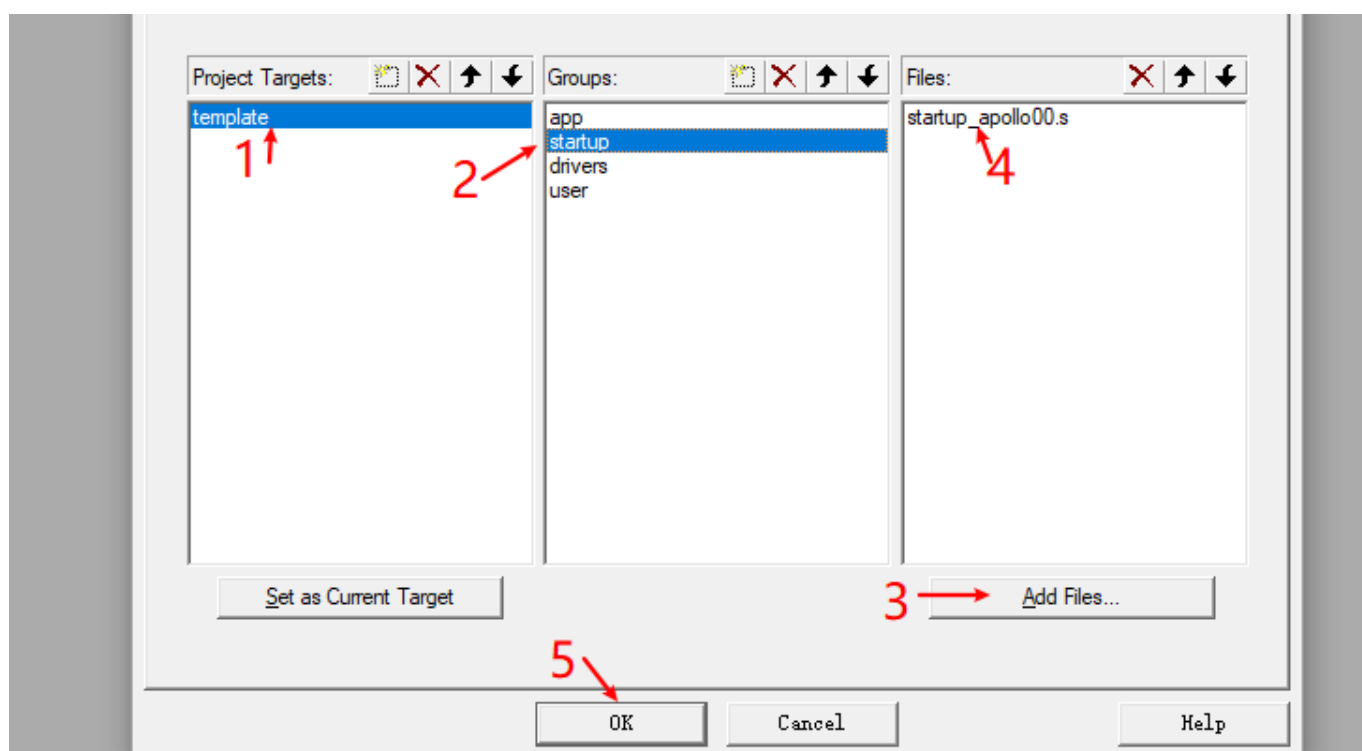
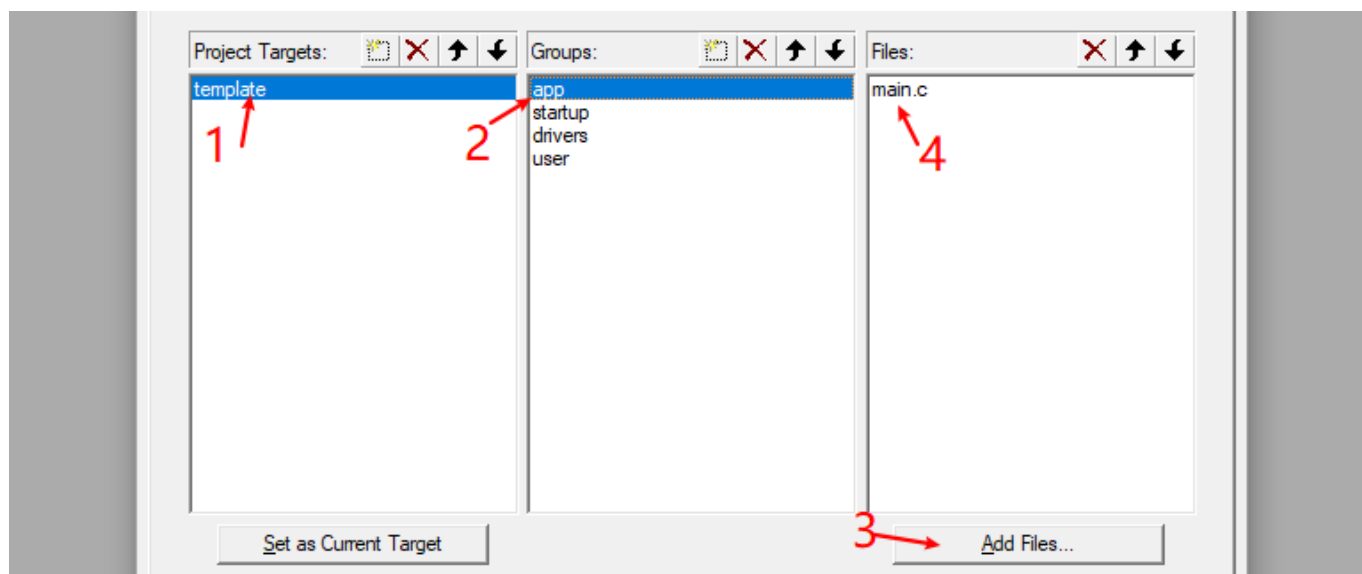


Ctrl+N, 然后 Ctrl+s, 命名为 main.c, 并将其保存在以下目录:



3、添加文件

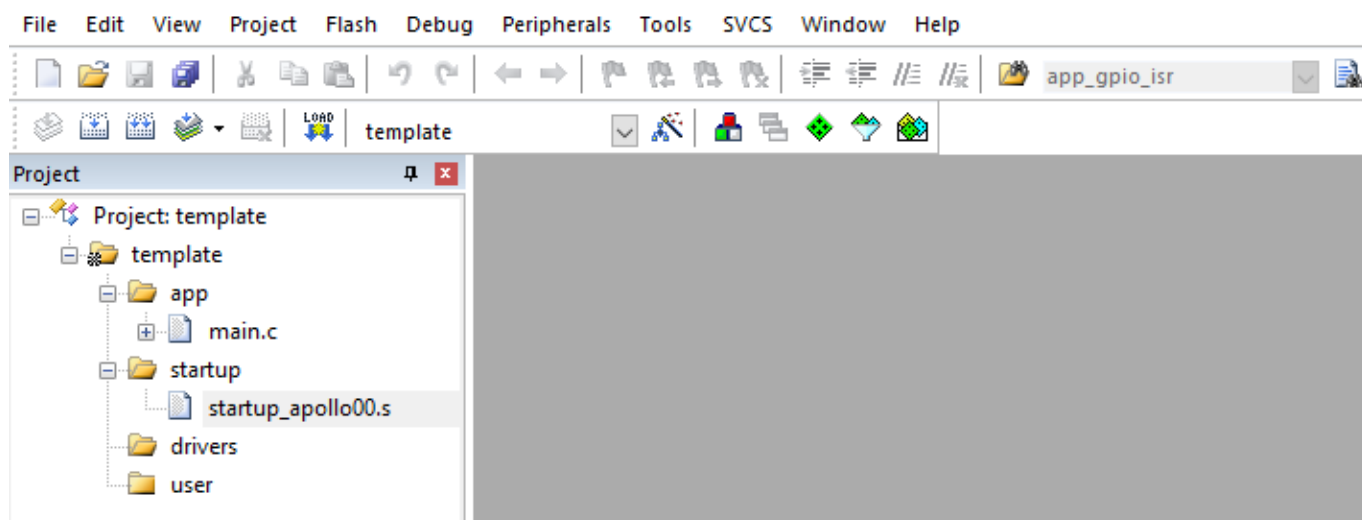




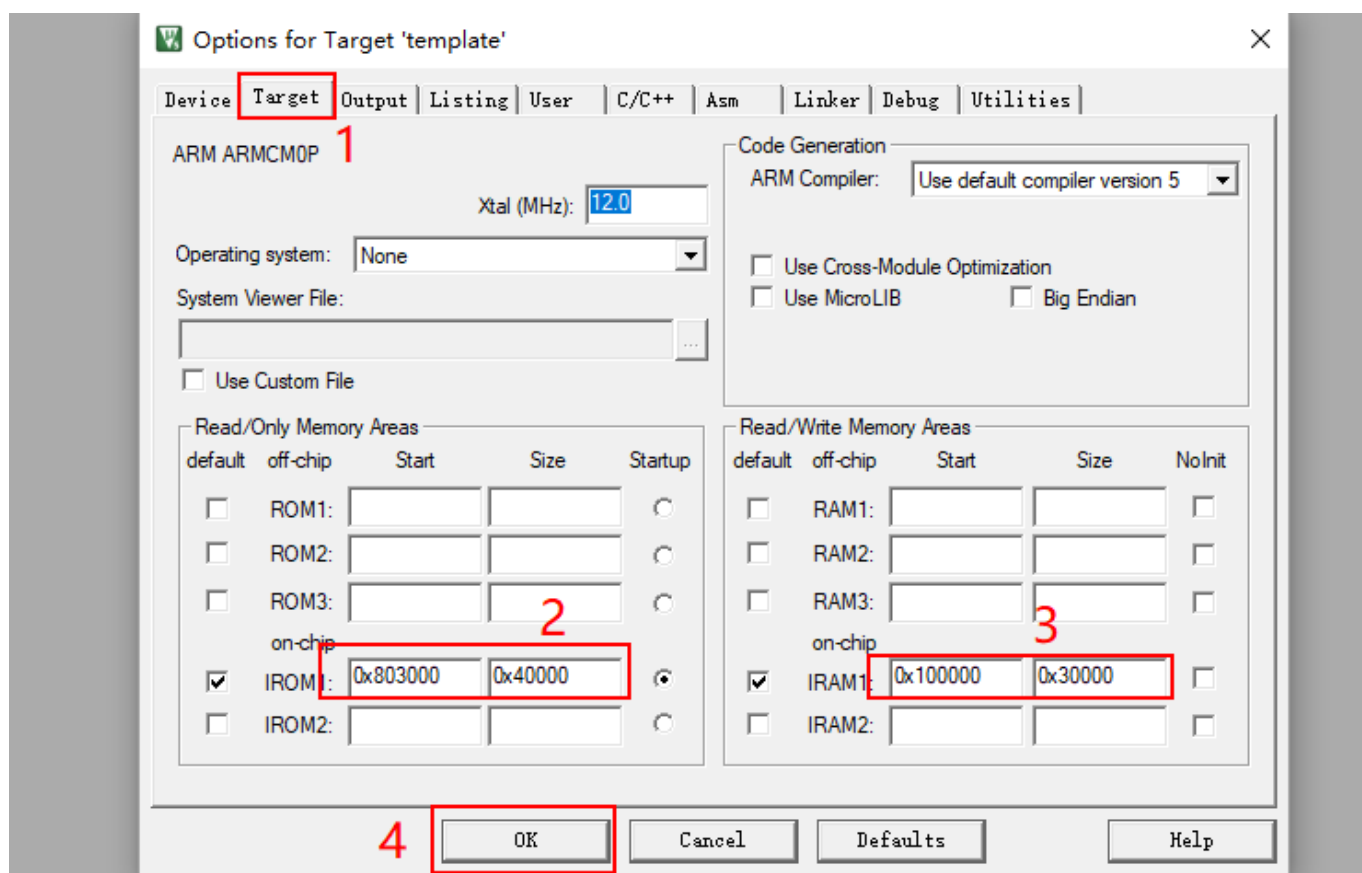
此处使用 apollo00 的启动文件作为演示，不同平台的.s 启动文件在其相对应的路径下：

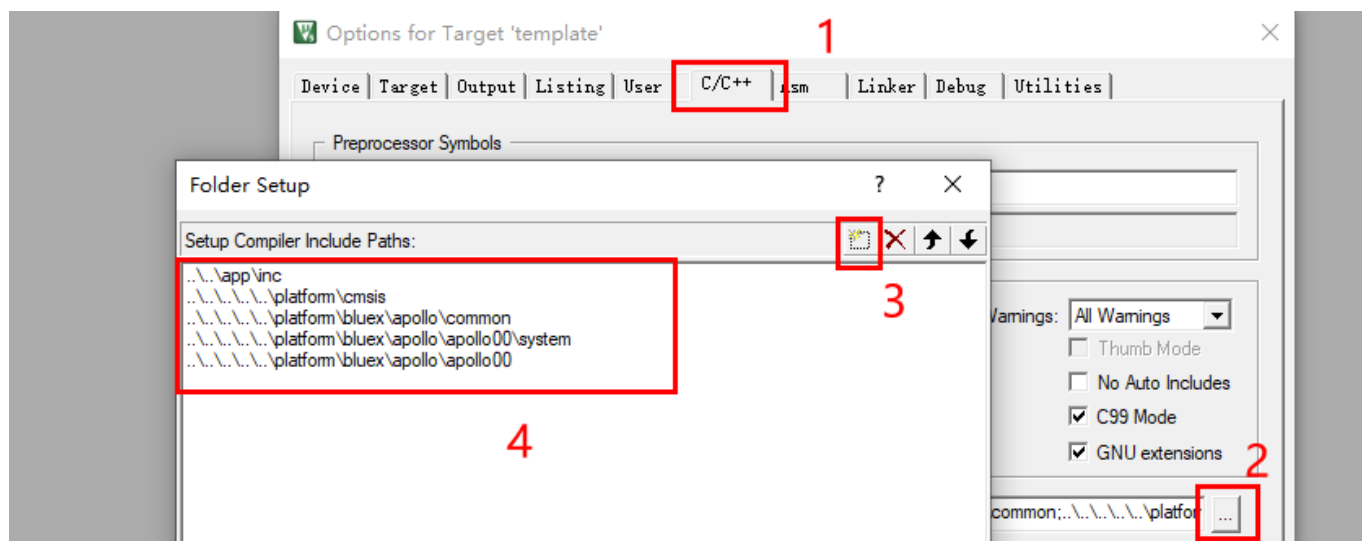


Drivers 目录暂时不需要添加文件。然后点击 OK 即可，添加后文件目录如下：



4、工程配置





点击 OK 即可，然后打开 main.c 输入以下代码：

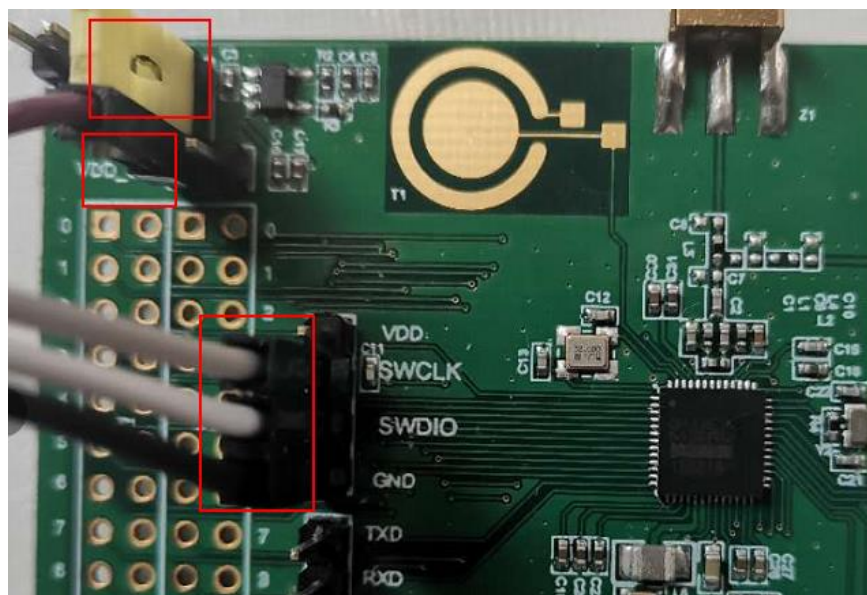
```
1. int main( void )
2. {
3.
4. }
```

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



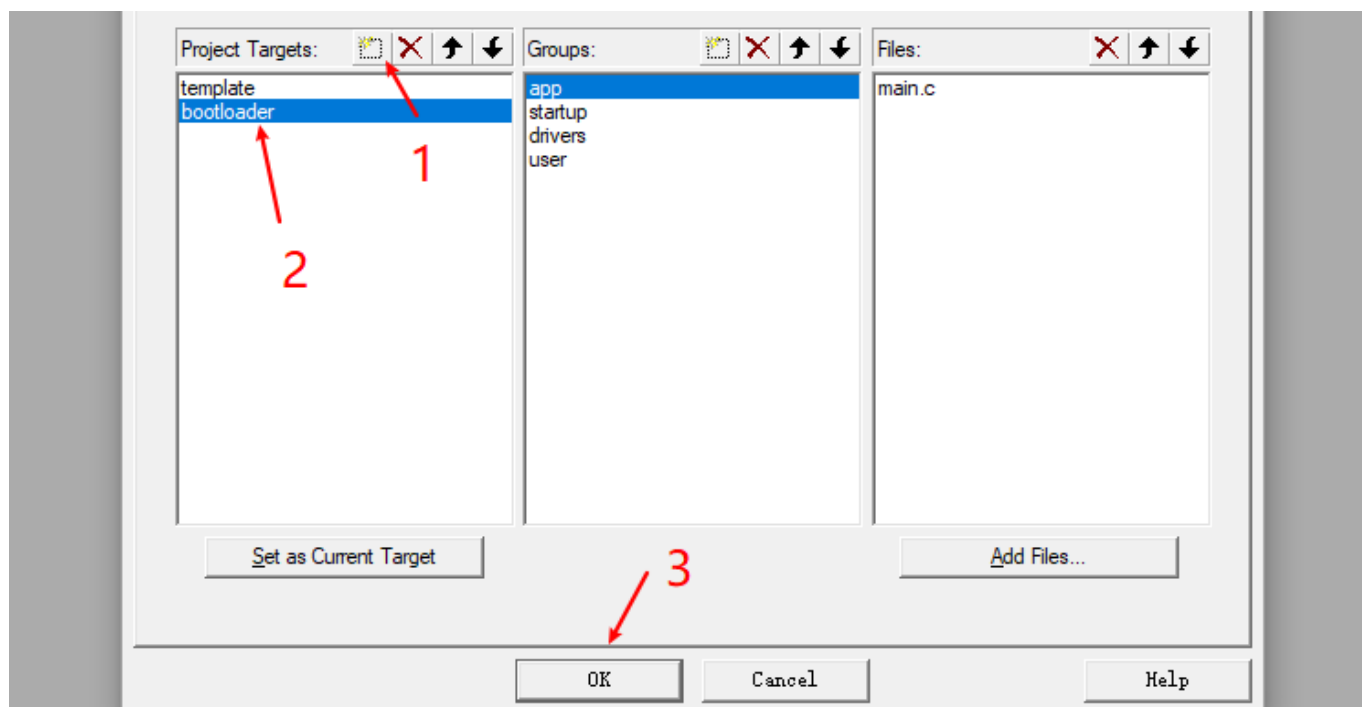


图中紫色-VDD、黑色-GND、白色-SWCLK、灰色-SWDIO。

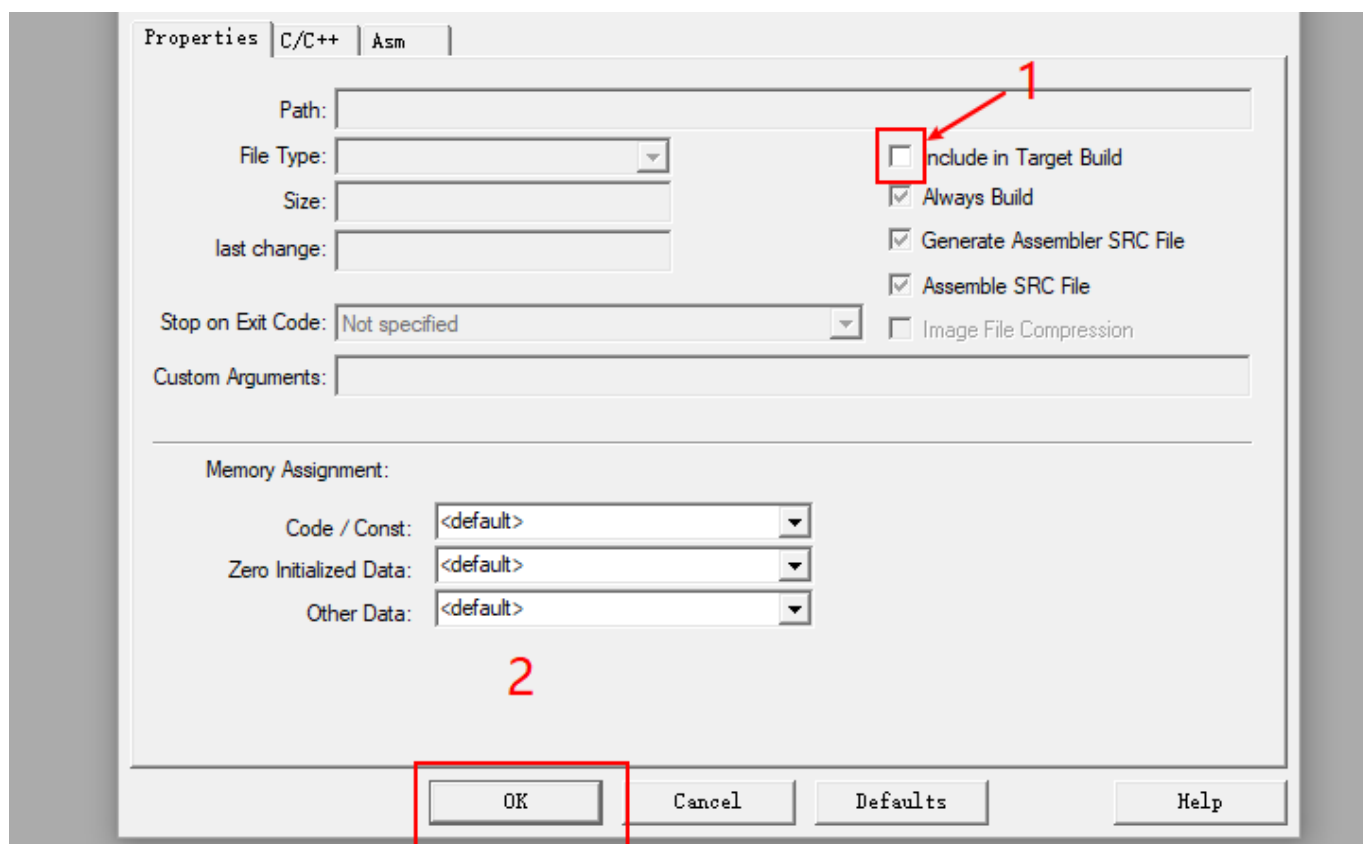
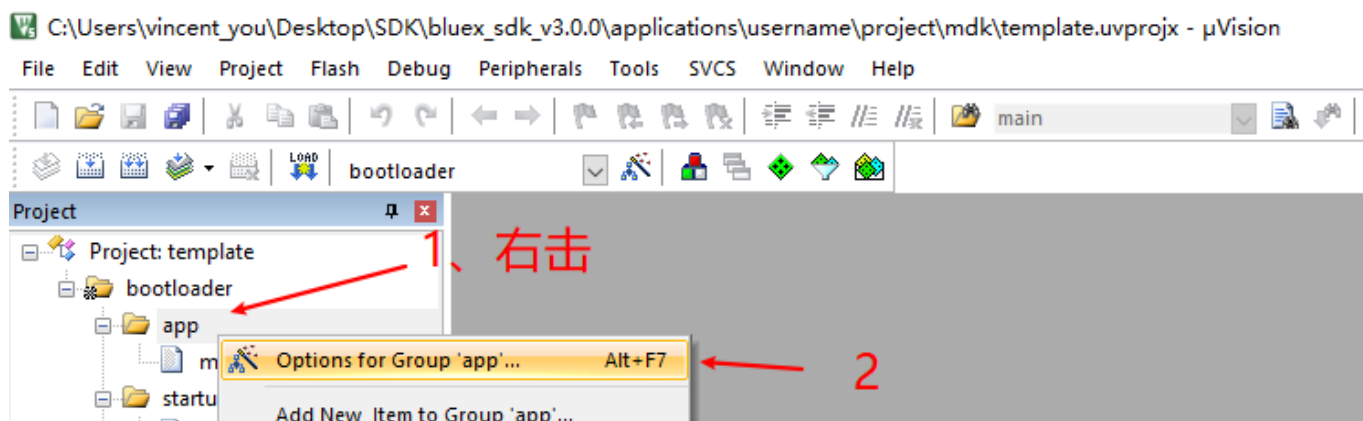
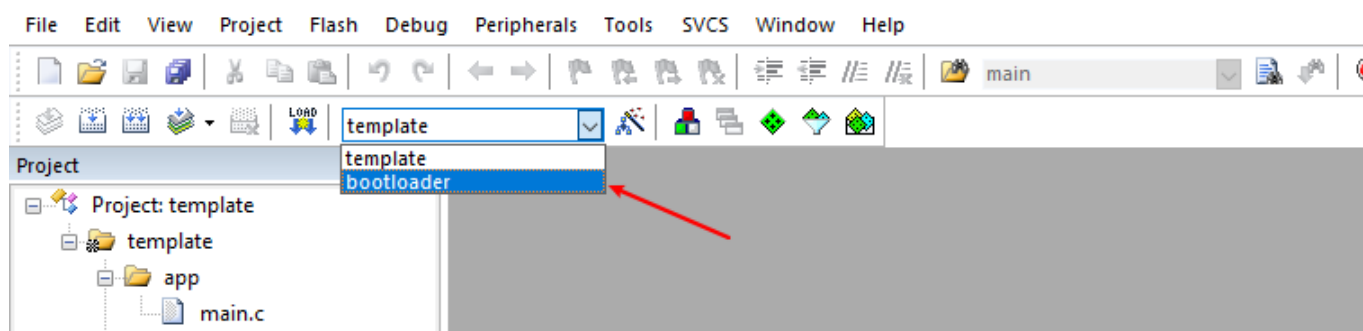
注意:Jlink 的版本不同, VDD 有时 1 引脚, 有时在 2 引脚, 图中接入的为 2 引脚, 用户请自行判断 VDD 接入点, 建议使用外部供电, 然后共地。

2、MKD 直接下载

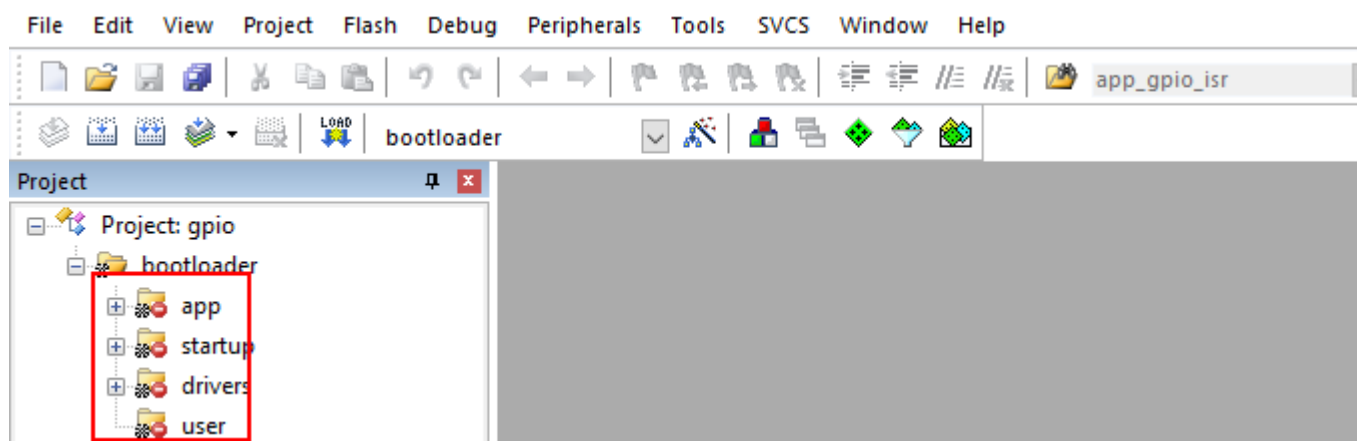
以下载 bootloader 作为演示, 后面的例程都是需要烧录 bootloader 才能正常运行的 (SDK 中 example 的外设例程全都配置好 bootloader 的烧录方式)。首先新建一个目标名称:



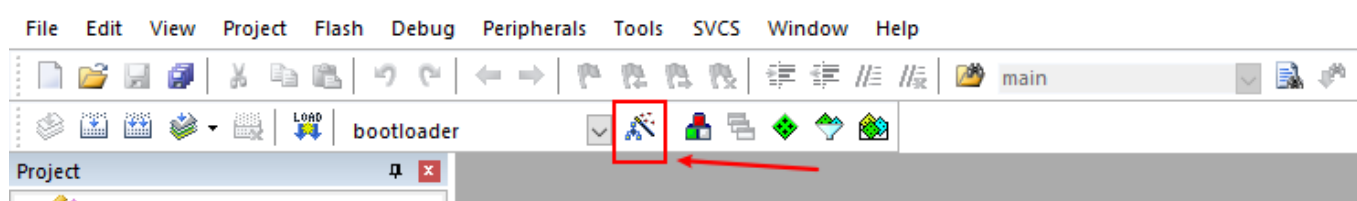
选中 bootloader 目标:



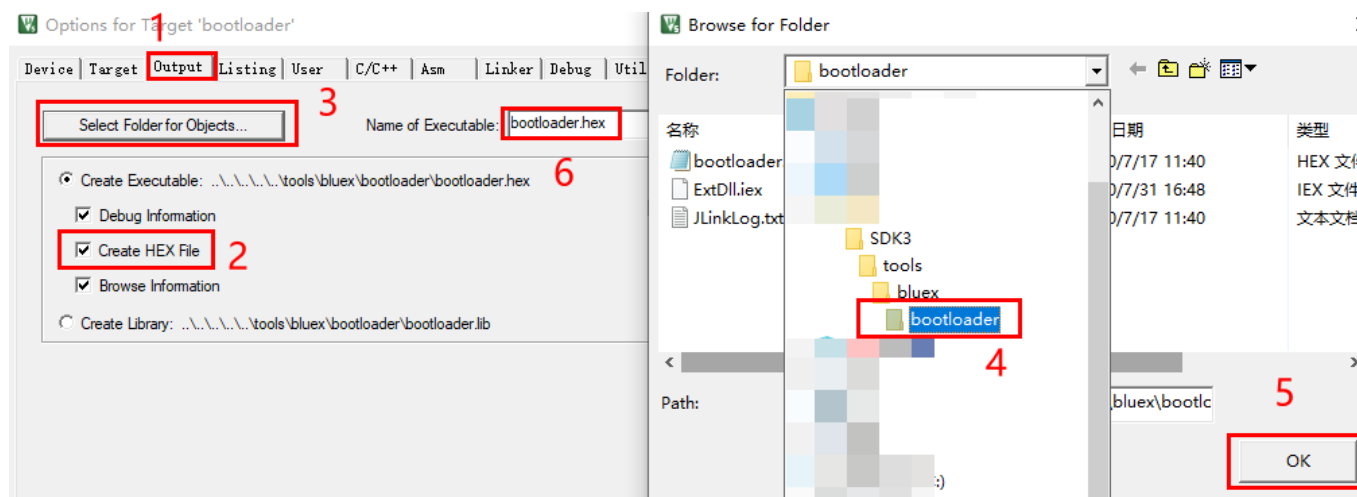
同理去掉所有:



打开配置

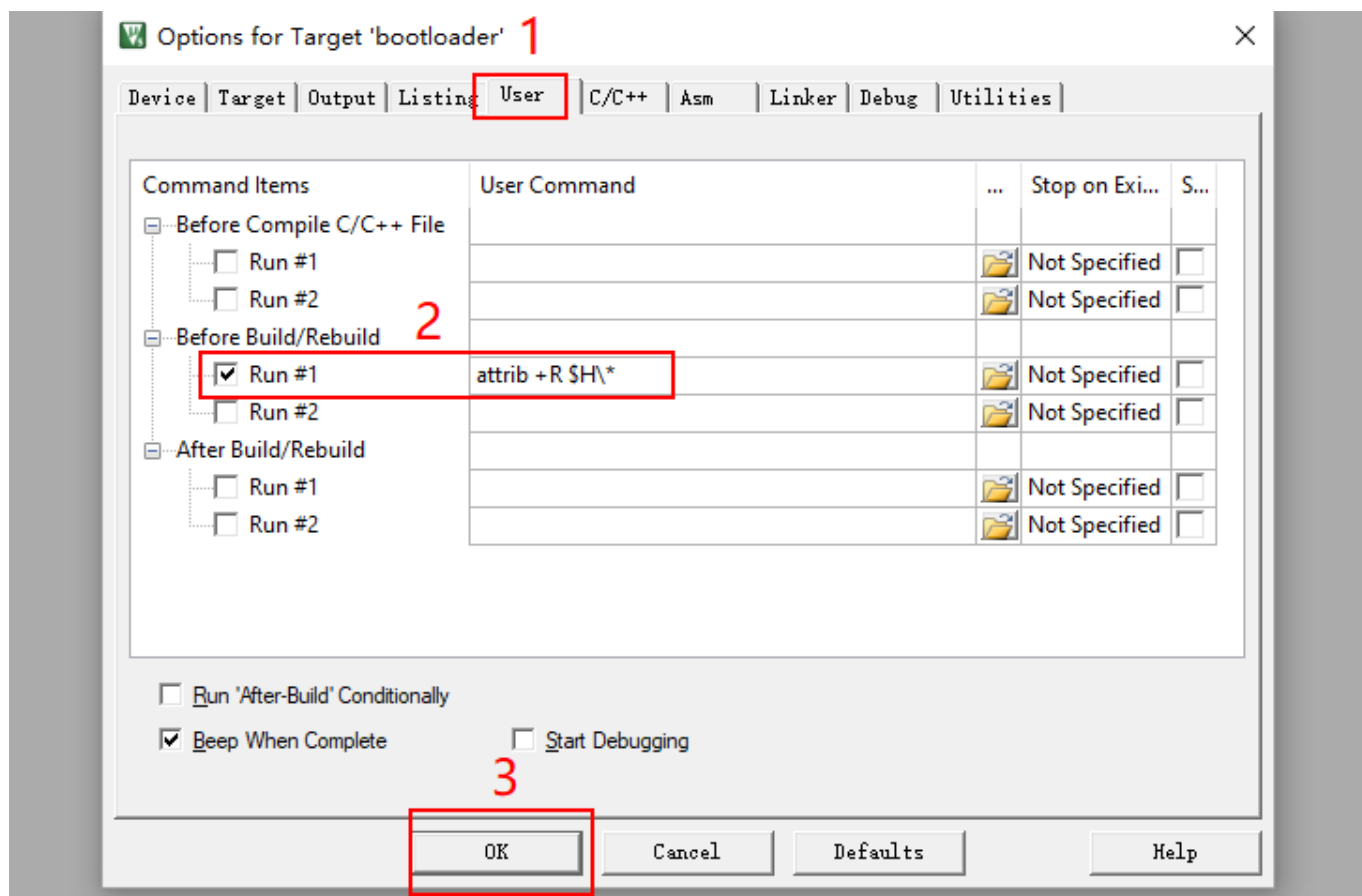


选择工程文件

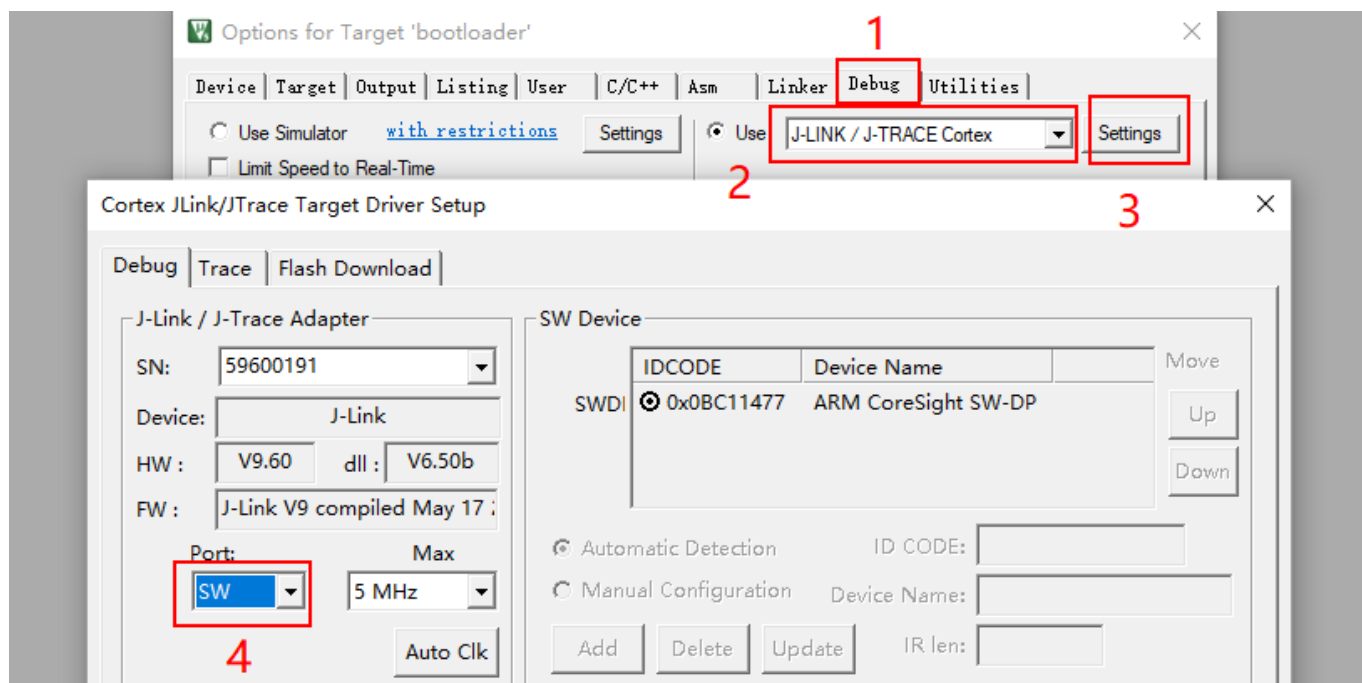


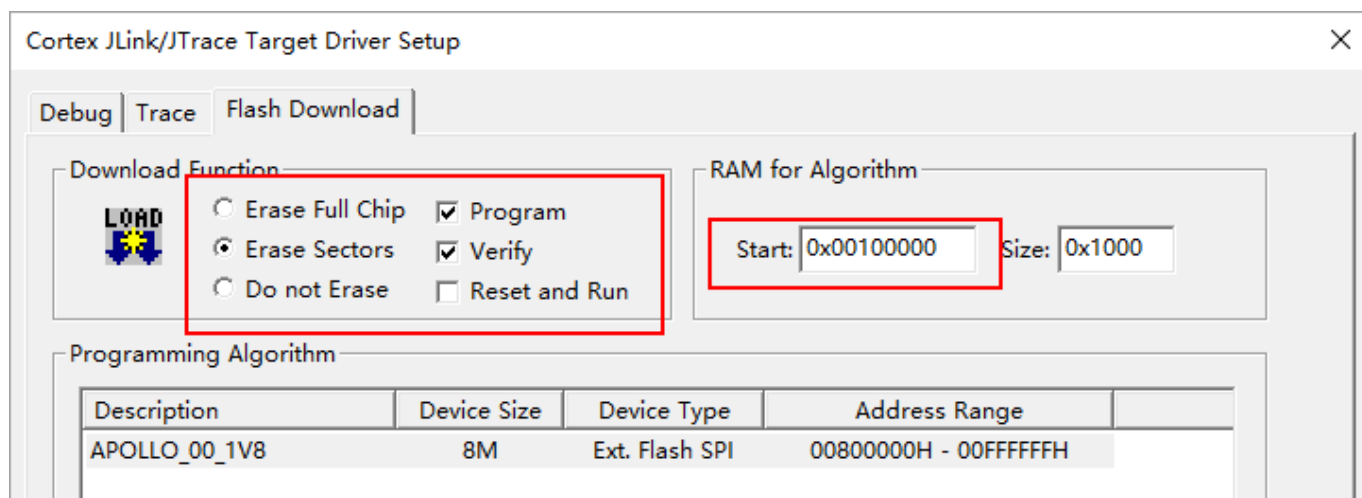
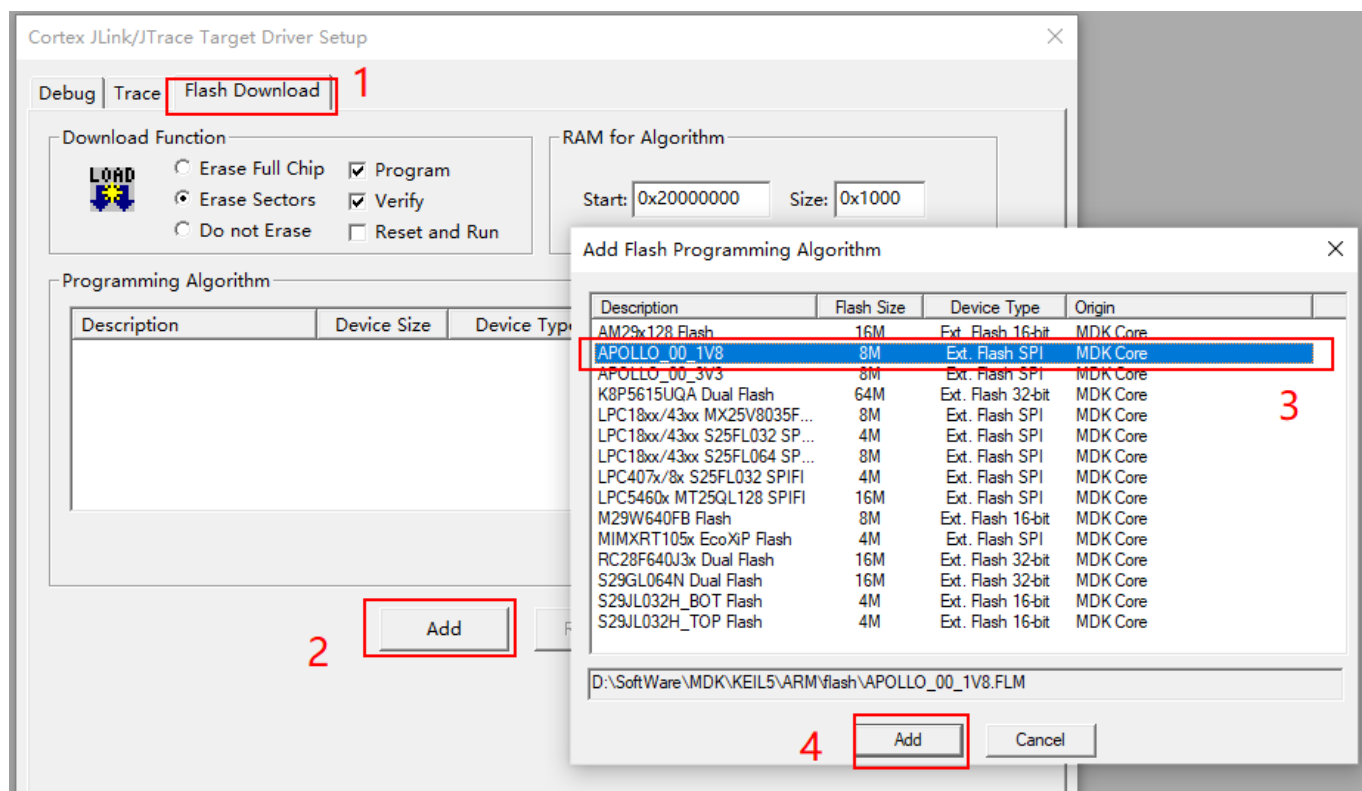
注意图中红框第 4 步，选择的路径必须跟图中所示一致。

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。

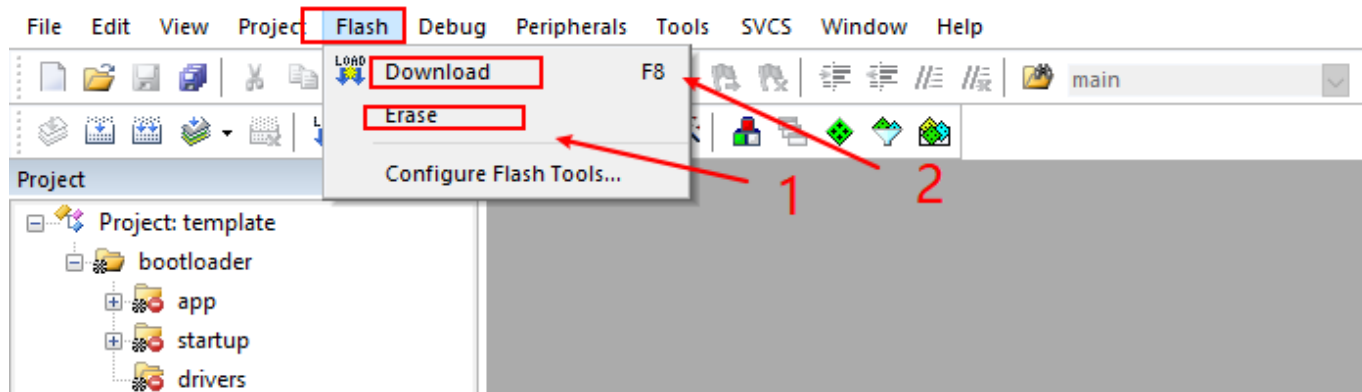


输入此命令 “attrib +R \$H*”





为确保烧录的是此 bootloader，请先擦除芯片，然后再烧录。

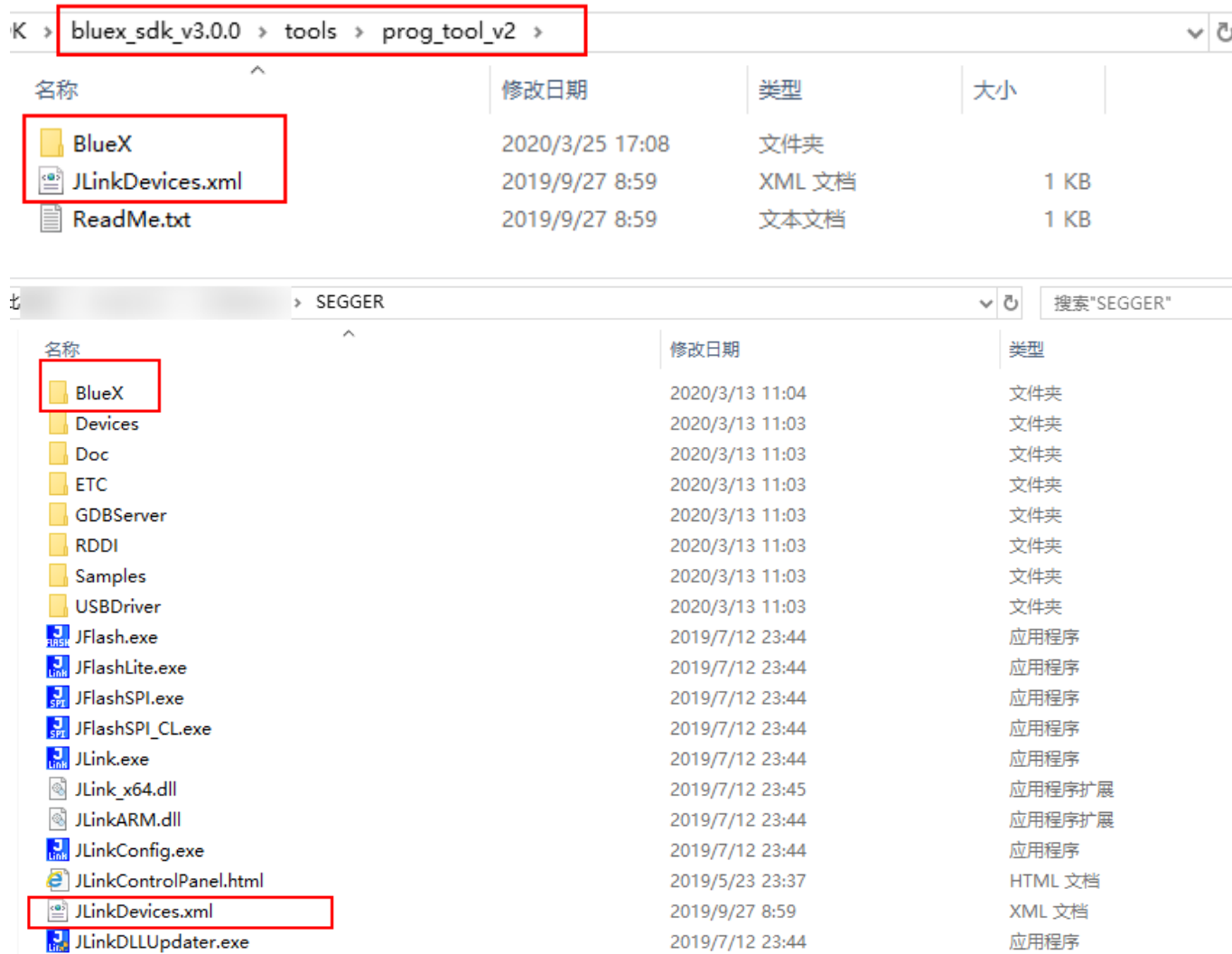


先擦除，后烧写。

3、J-Flash 下载

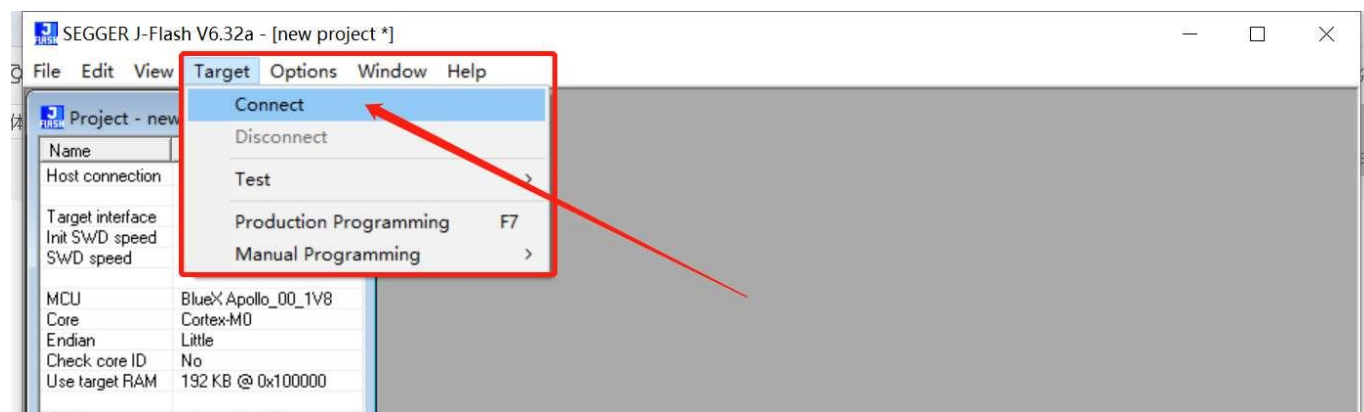
3.1 复制文档

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

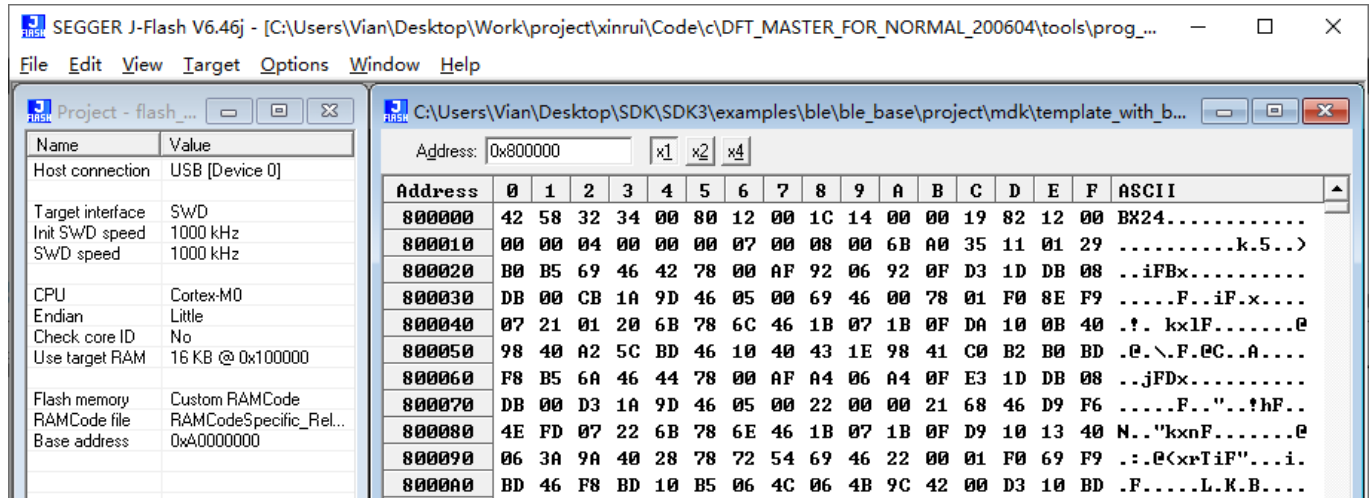


3.2 连接 Jlink

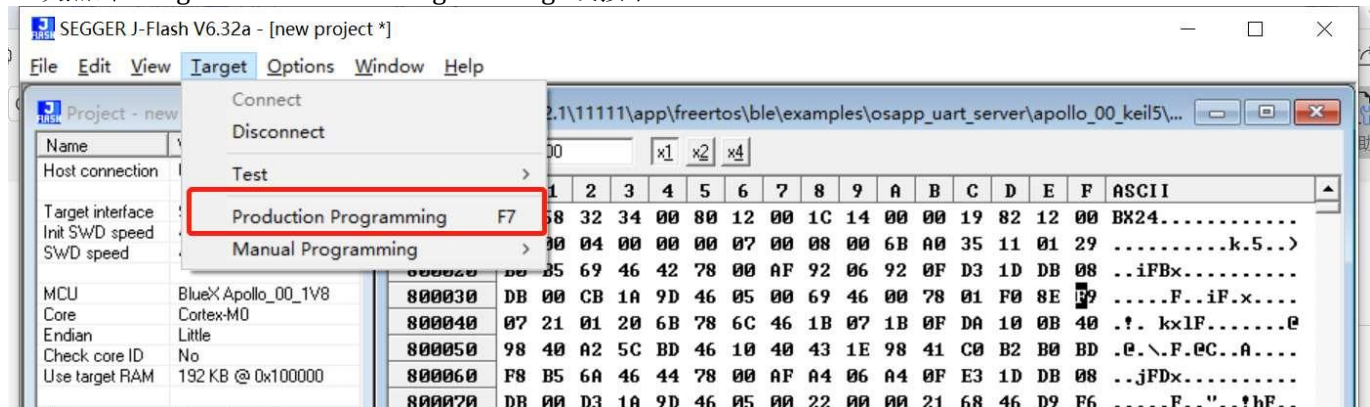
打开 J-Flash，使用 J-Flash 连接开发板，在 J-Flash 界面点击 Target -> Connect



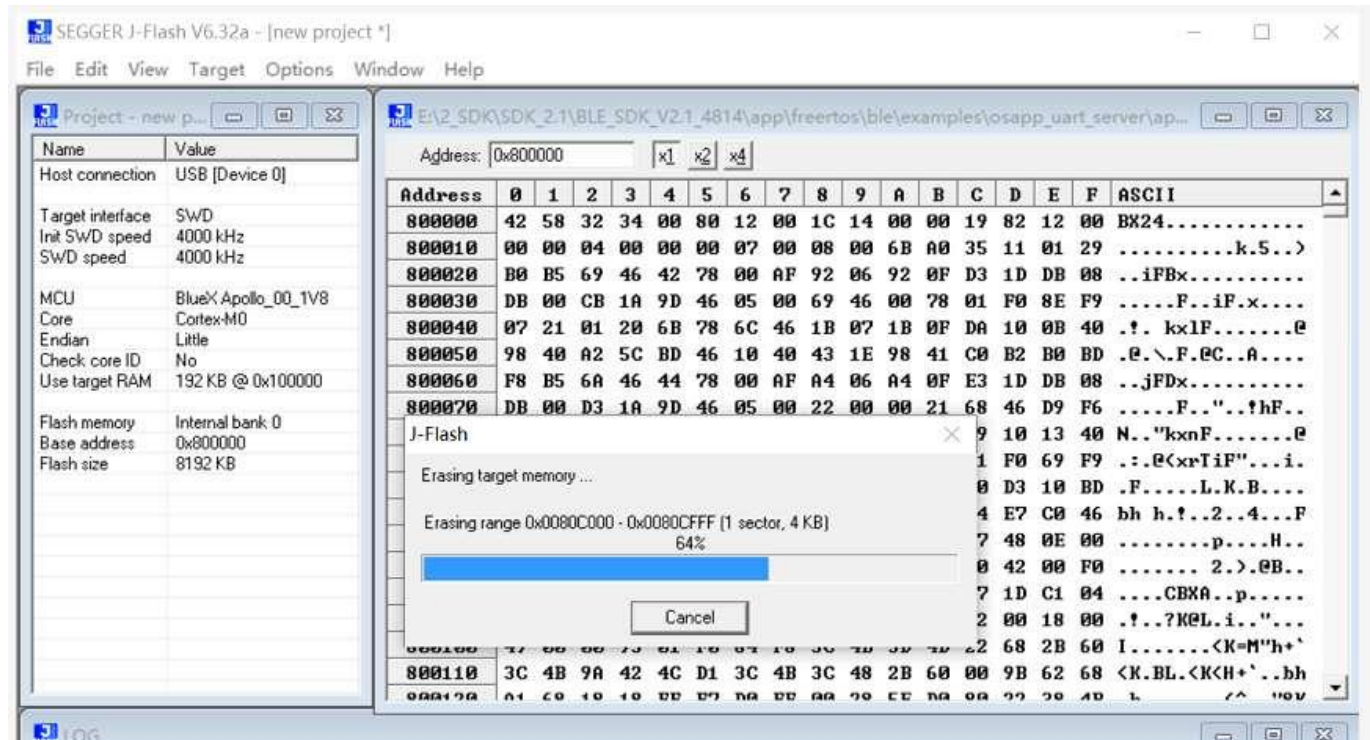
连接成功后，将 hex 文件（如 template_with_bootloader.hex，此文件在\project\mdk\目录下）拖入 J-Flash 软件



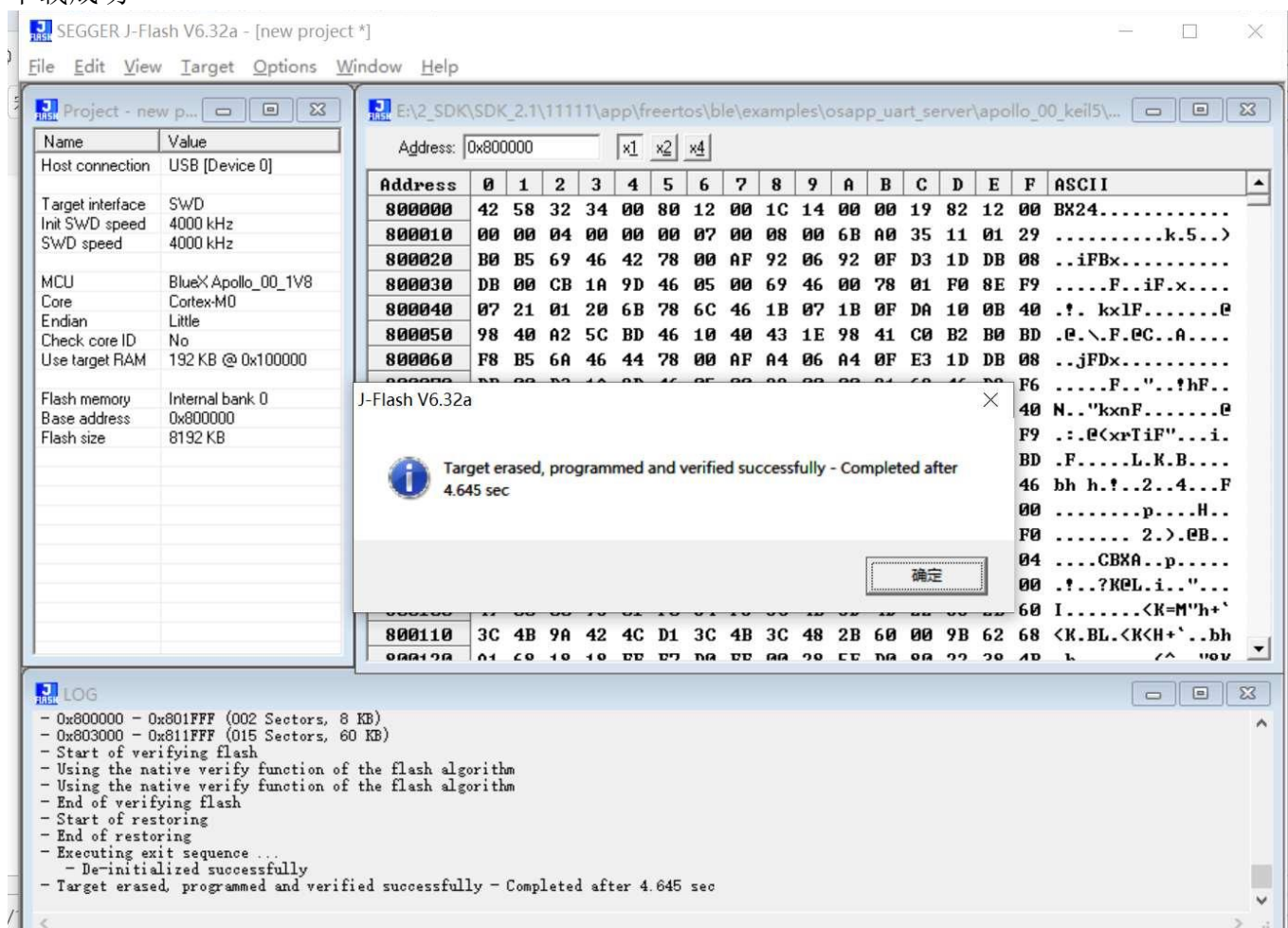
一次点击 Target -> Production Programming 或按下 F7



开始下载:

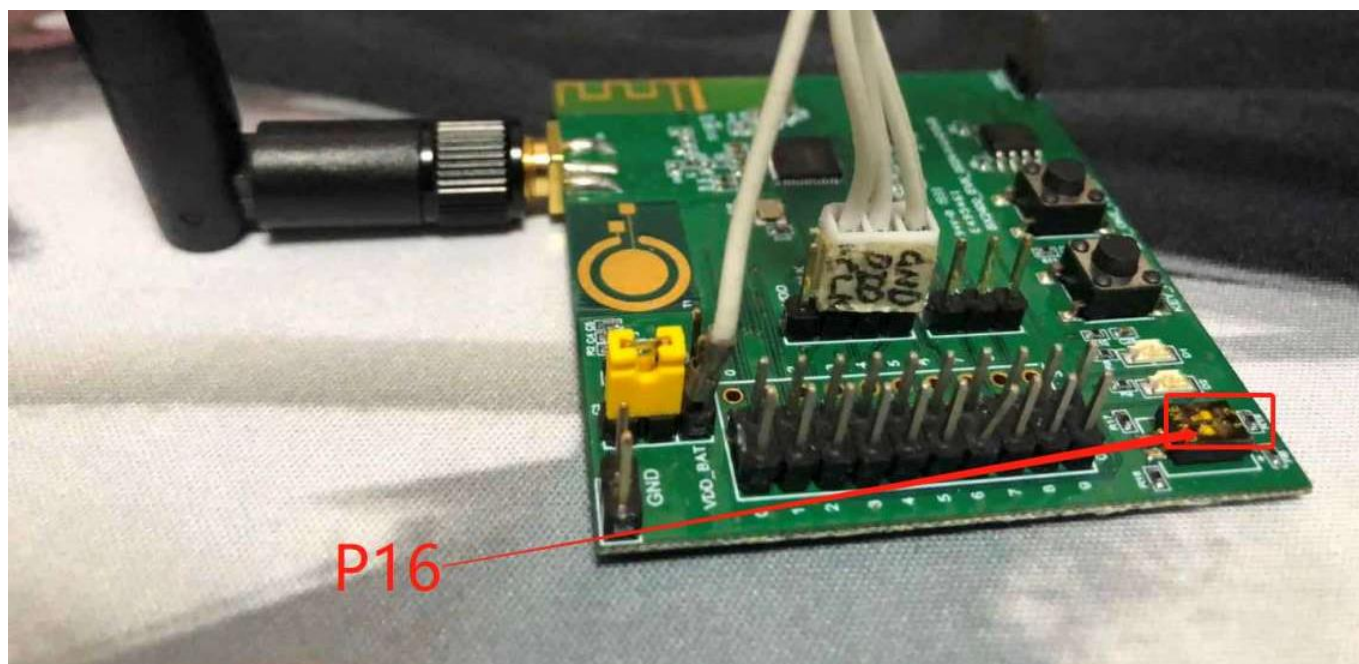


下载成功



4、常见错误

若出现连接失败或烧录失败，将 P16 拉高后重新上电，如下图，重新执行烧录动作



第六章 点亮 LED

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，直接复制一份之前的，将工程名称修改为 gpio 即可。



2、代码编写

```

1. #include "apollo_00_reg.h"
2.
3. void user_delay( uint32_t val )
4. {
5.     for( uint32_t i = 0; i < val; i++ )
6.         for( uint32_t j = 0; j < 5000; j++ );
7. }
8.
9. void output_test( void )
10. {
11.     uint32_t pin_mark = GPIO_PIN_2 | GPIO_PIN_3;
12.     BX_PER->CLKG1 |= PER_CLKG1_SET_GPIO;
13.     BX_GPIOA->DIR |= ( pin_mark );
14.     while( 1 ) {
15.         BX_GPIOA->OD ^= ( pin_mark );
16.         user_delay( 500 );
17.     }
18. }
19. /** -----
20.  * @brief :
21.  * @note :
22.  * @param :
23.  * @retval :
24.  * -----*/
25. int main( void )
26. {
27.
28.     __DMB();
29.     SCB->VTOR = 0x00803000;
30.     __DSB();
31.
32.     output_test()
33.
34.     while( 1 );
35. }

```

3、实例演示

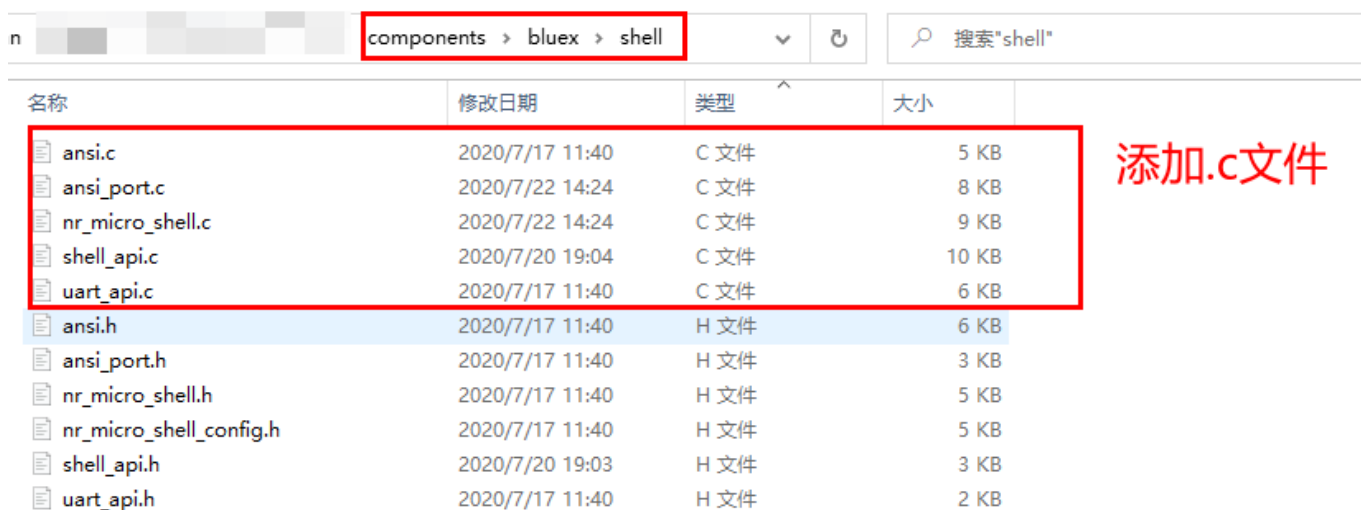
- 按第四章的方式配置工程
- 编译文件
- 按第五章方法连接硬件

- 按第五章方式烧录固件
- 重启开发板，观察此时 P02 引脚对应的 LED 灯是否闪烁

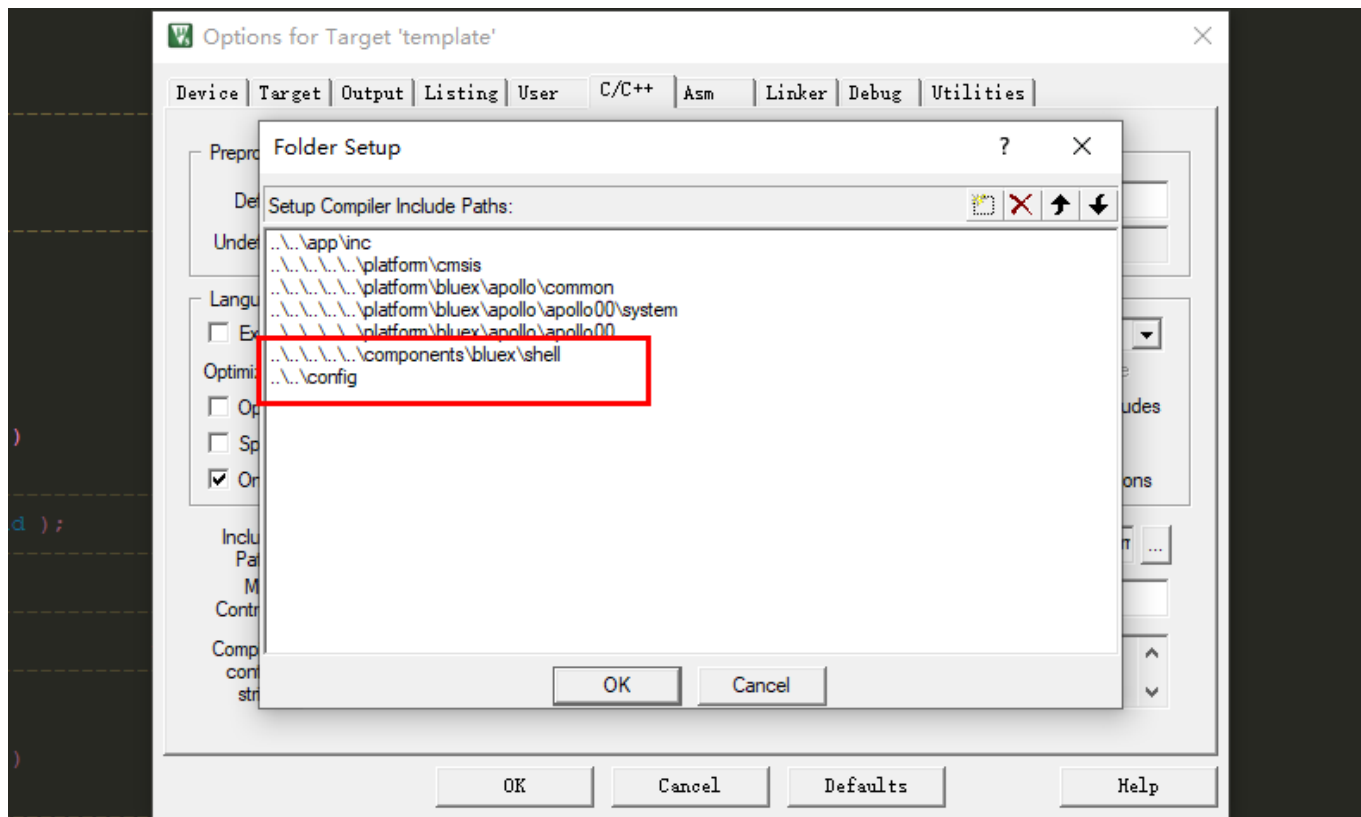
第七章 调试输出

1、串口调试

Bluex 提供一个基于 UART 的 shell 脚本，里面集成了 printf 的重定向，直接添加相关文件即可使用。如图：



添加 include 路径：



2、代码编写

```

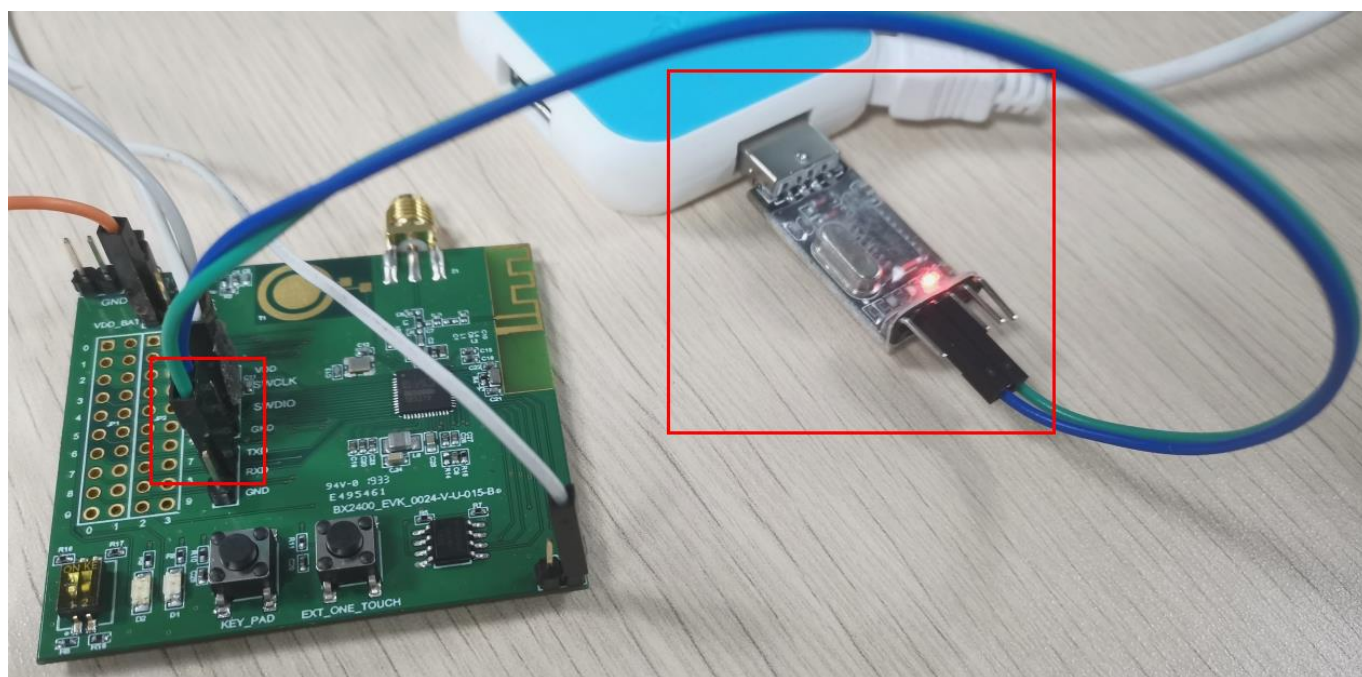
1. #include "apollo_00_reg.h"
2. #include "bx_shell.h"
3.
4. /** -----
5.  * @brief :
6.  * @note :
7.  * @param :
8.  * @retval :
9.  * ----- */
10. int main( void )
11. {
12.     __DMB();
13.     SCB->VTOR = APOLLO_00_VTOR_BASE;
14.     __DSB();
15.
16.     bxsh_init();
17.
18.     while( 1 ) {
19.         bxsh_run();
20.     }
21. }

```

3、实例演示

3.1 硬件连接

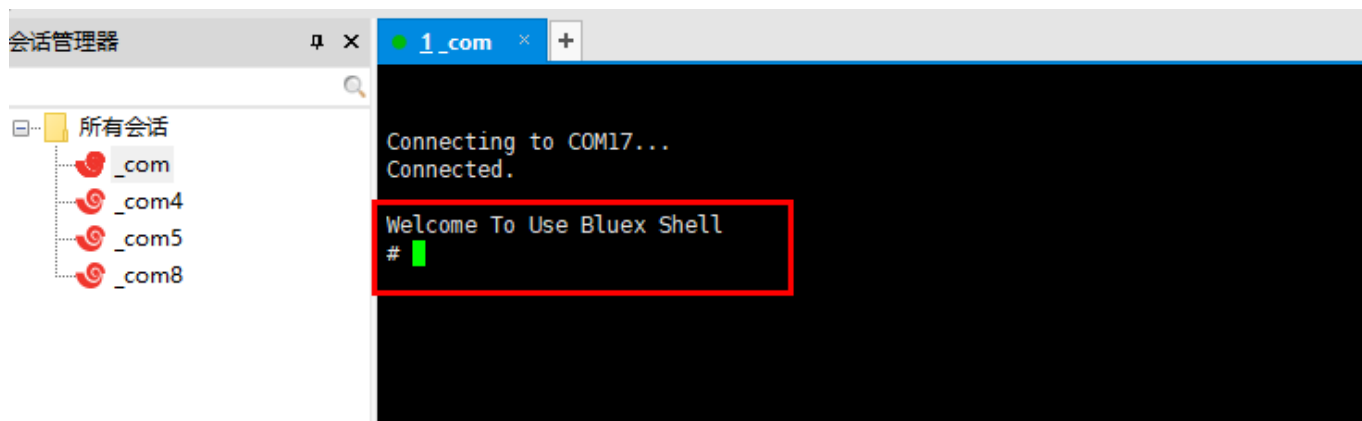
使用 USB 转串口工具，连接设备和电脑，如图：



3.2 演示步骤

- 按第四章的方式配置工程
- 编写代码，然后编译文件
- 按第五章方法连接硬件

- 按第五章方式烧录固件
- 重启开发板，观察此时此时串口是否有打印输出，如下图：



第八章 PEK 工程

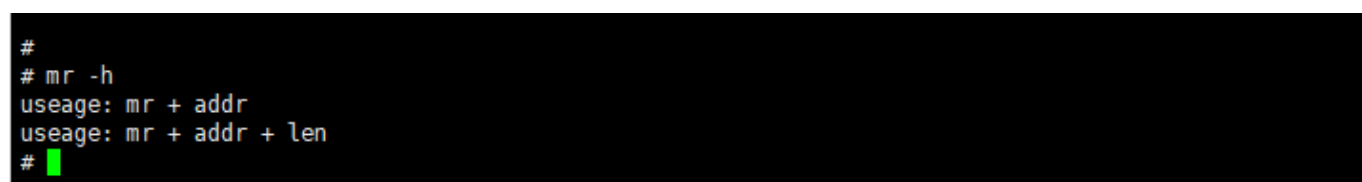
PEK 为（Peripheral Evaluation Kit）的缩写，即外设评估工具，这是基于 uart shell 实现的，uart shell 内置了 3 个指令，mr、mw、reset、ls 指令，分别提供内存读、内存写、复位以及显示当前所有支持的指令操作，所有指令是以空格作为参数分隔符。PEK 工程路径如下：



1、指令

1.1 mr 指令

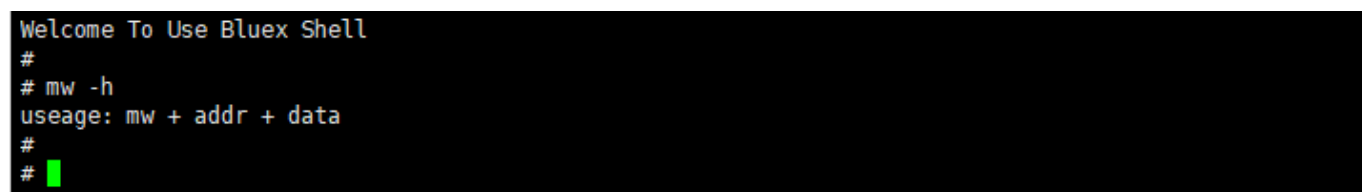
内存读指令，可以通过参数 ‘-h’ 获取帮助信息：



内存读取可以在任何时候对任何数据进行读取，包括寄存器、flash、sram 中的值，addr 的值支持 16 进制和 10 进制数，可以在设备运行时查看寄存器、sram 中的值

1.2 mw 指令

内存写指令，可以通过参数 ‘-h’ 获取帮助信息：



内存写可以对寄存器、sram 中的值进行写操作，addr 和 data 的值支持 16 进制和 10 进制数

1.3 reset 指令

复位指令，设备在接收到此指令后，大概 3s 后，系统会重启



```

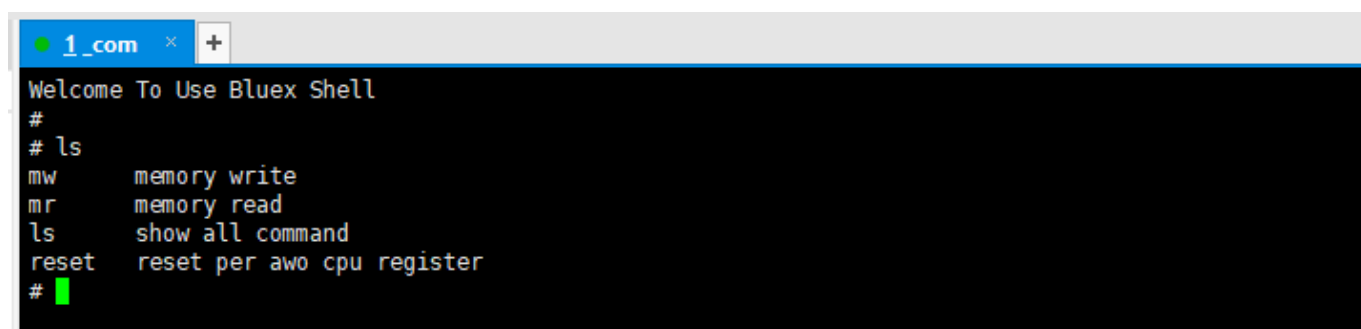
1_com x +
Welcome To Use Bluex Shell
#
# reset
# Welcome To Use Bluex Shell
#

```

执行reset,大概3S后会重新打印welcom

1.4 ls 指令

显示当前支持的所有指令



```

1_com x +
Welcome To Use Bluex Shell
#
# ls
mw      memory write
mr      memory read
ls      show all command
reset   reset per awo cpu register
#

```

1.5 其它指令

UART SHELL 可以添加自己的指令，在 pek 工程中，添加了对外设的基本测试指令，详细可以查看代码。

```

/** -----
 * @brief :
 * @note :
 * @param :
 * @retval :
 * -----*/
int main( void )
{
    init();
    bxsh_init();

    /* add user cmd */
    led_add_shell_cmd();
    gpio_add_shell_cmd();
    pwm_add_shell_cmd();
    tim_add_shell_cmd();
    rtc_add_shell_cmd();
    iic_add_shell_cmd();
    wdt_add_shell_cmd();
    spim_add_shell_cmd();

    iic_lis3dsh_add_shell_cmd();
    /* end of add user cmd */

    while( 1 ) {
        bxsh_run();
    }
}

```

2、使用方法

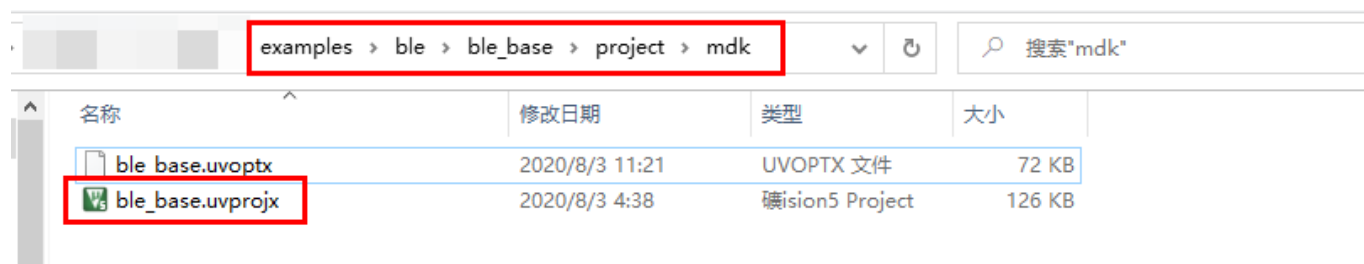
- 将代码烧录到开发板
- 将“\r”作为换行符，换行符可在 nr_micro_shell_cofnfig.h 文件中的第 128 行

```
# ls
mw      memory write
mr      memory read
ls      show all command
reset   reset per awo cpu register
led     led test
gpio    gpio test
pwm     pwm test
tim     timer0 test
rtc     rtc test
iic     iic test
wdt     wdt test
spin    spin test
lis3    lis3 test
#
```

第九章 基础 BLE

1、新建工程

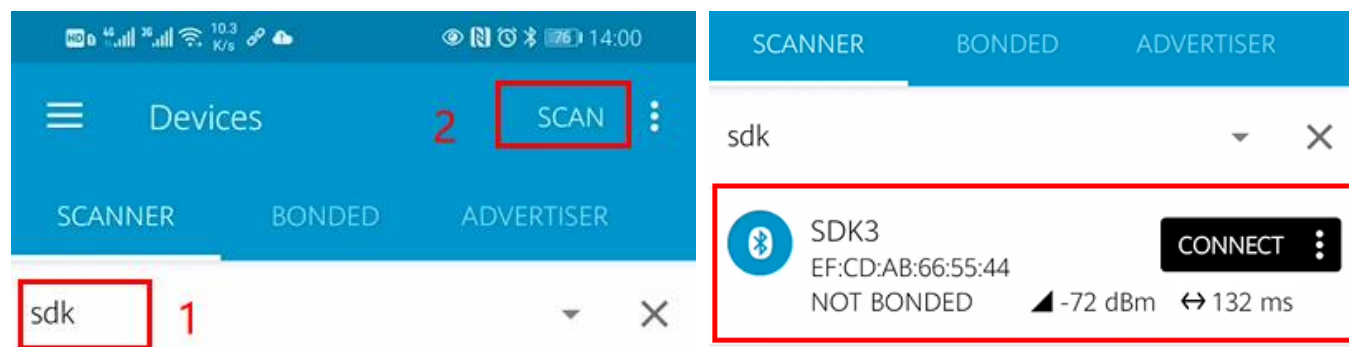
由于新建 BLE 工程要做的配置比较多，此处就不一一详细介绍了，这里直接使用新建好的工程：



2、实例演示

- 编译文件
- 按第五章方法连接硬件
- 按第五章方式烧录固件
- 重启开发板
- 手机打开蓝牙，打开 nRF Connect 软件

如图输入“SDK”，然后扫描，可以扫描到“SDK3”的设备，默认的 MAC 地址为“EF:CD:AB:66:55:44”：



点击“CONNECT”连接：

BONDED	ADVERTISER	SDK3 EF:CD:AB:66:55:44	×
CONNECTED			
NOT BONDED	CLIENT	SERVER	⋮
Generic Access			
UUID: 0x1800			
PRIMARY SERVICE			
1			
Generic Attribute			
UUID: 0x1801			
PRIMARY SERVICE			
2			
Device Information			
UUID: 0x180A			
PRIMARY SERVICE			
3			
Unknown Service			
UUID: 00002600-0000-1000-8000-00805f9b34fb			
PRIMARY SERVICE			
4			
Unknown Service			
UUID: 00006666-d102-e111-9b23-00025b00a5c7			
PRIMARY SERVICE			
5			

可以看到一共又 5 个服务，第 1、2 个为通用服务，第 3 个为设备信息，第 4 个为 blueX 内置的 OTA 服务，第 5 个为自定义服务，可以尝试读写一些参数。

第十章 OTA 介绍

1、新建工程

与第 8 章一样，这里直接使用新建好的工程：

examples > ble > ble_base > project > mdk				▼	↺	🔍 搜索"mdk"
名称	修改日期	类型	大小			
ble_base.uvoptx	2020/8/3 11:21	UVOPTX 文件	72 KB			
ble_base.uvprojx	2020/8/3 4:38	Microvision5 Project	126 KB			

2、实例演示

- 编译文件
- 按第五章方法连接硬件
- 按第五章方式烧录固件

- 重启开发板
- 手机打开蓝牙，打开 OTA 软件，初始界面如图 10.1:
- 输入名称“SDK3”过滤，然后扫描，如图 10.2:
- 直接点击卡片（图 10.3），则会连接设备,并切换到图 10.4

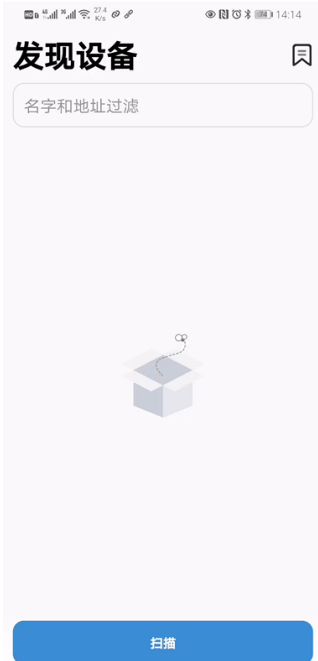


图 10.1

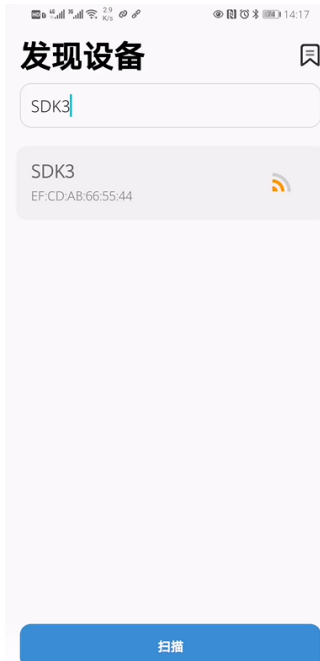


图 10.2



图 10.3



图 10.4

- 选择 OTA 文件路径后点击升级，如图 10.5
- 待测物固件升级成功后，待测物与手机自动断开连接，如图 10.6、10.7、10.8



图 10.5



图 10.6



图 10.7



图 10.8