

密级状态: 绝密() 秘密() 内部资料(✓) 公开()

文档编号: (芯片型号) - ASR6505 (英文、数字)

ASR6505 应用开发 DEMO 说明

文件状态: [✓] 正在修改 [] 正式发布	当前版本:	V0.1
	作者:	Ruilin Hao
	启动日期:	2019-04-16
	审核:	
	完成日期:	2019-04-20

翱捷科技（上海）有限公司

ASR Microelectronics Co., Ltd

(版本所有, 翻版必究)

版本历史

版本号	修改日期	作 者	修 改 说 明
V0.1	2019.04.20	Ruilin Hao	Initial Version

Table of Contents

1	概述.....	5
2	准备.....	6
2.1	硬件准备	6
2.2	开发环境安装	7
2.3	SDK 获取.....	7
2.4	联网准备（联网通信需要）	7
2.4.1	节点信息申请	7
2.4.2	网关配置	7
3	DEMO 程序说明	8
3.1	Demo 简介.....	8
3.2	代码结构	8
3.3	代码流程	8
3.4	LoRaWan 配置与修改说明	9
3.4.1	三元组修改	9
3.4.2	入网方式修改	9
3.4.3	ClassC 修改	9
3.4.4	频道掩码修改	10
3.4.5	Mac 参数修改	10
3.4.6	Region 修改	10
3.5	外设使用说明	11
3.5.1	GPIO	11
3.5.2	UART	12
3.5.3	I2C	12
3.5.4	SPI	12
3.5.5	Timer.....	12
4	软件编译与烧录.....	13
4.1	IAR 环境开发.....	13
4.1.1	打开工程	13
4.1.2	编译	13
4.1.3	调试	13
4.2	STVD 环境开发.....	14
4.2.1	打开工程	14
4.2.2	编译	15
4.2.3	调试	15
4.3	UART 升级	17
4.3.1	准备	17
4.3.2	设置 Bootloader Check.....	17
4.3.3	升级固件	21

5	Q&A.....	29
5.1	如何修改 SDK 支持 TCXO 晶振?	29

ASR Confidential

1 概述

本文档主要对 ASR6506 SDK 中的 classA demo 程序进行说明，方便客户在 ASR6505 上进行应用程序的二次开发。

ASR Confidential

2 准备

2.1 硬件准备

LoRa 节点必需硬件列表如下：

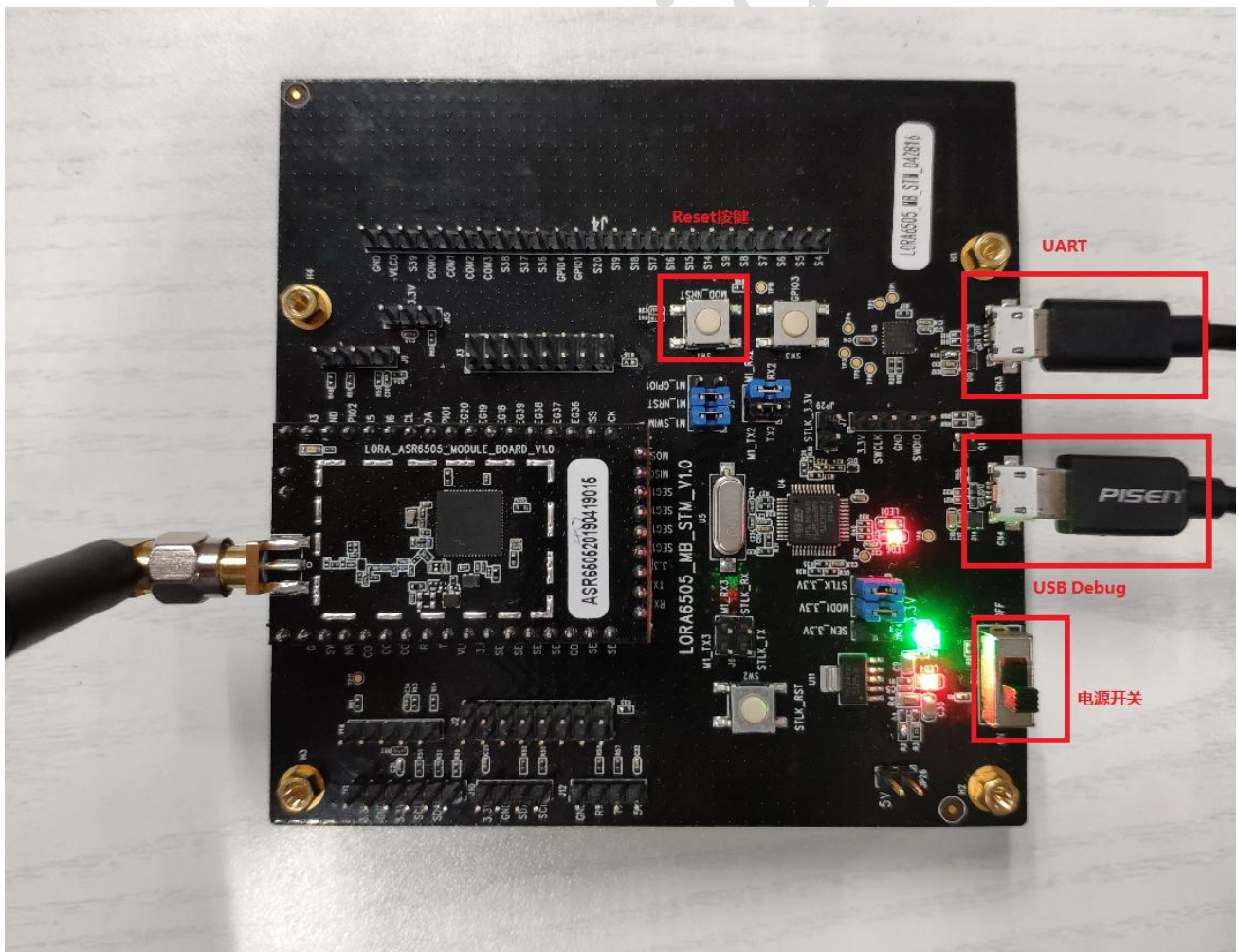
- 1) ASR6505 LoRa 子板 1 个
- 2) ASR6505 LoRa 母板 1 个
- 3) 天线 1 根
- 4) USB 线 2 根
- 5) PC 机 1 台

另外，针对 LoRaWan 通信还需要以下硬件：

- 1) LoRaWan 网关及对应的服务器

针对 LoRa 点对点通信还需要以下硬件：

- 1) 另外一套 LoRa 节点硬件



硬件连接图

2.2 开发环境安装

请参考开发环境搭建文档，进行开发环境的安装。

2.3 SDK 获取

请咨询 ASR 技术支持人员进行获取。

2.4 联网准备（联网通信需要）

2.4.1 节点信息申请

节点设备的配置信息需要从服务器申请，通常 OTAA 设备需要 DEVEUI，APPEUI 和 APPKEY 等信息，ABP 设备需要 DEVADDR，NWKSKEY 和 APPSKEY 等信息。

2.4.2 网关配置

节点连接网关时，需要网关的信道配置信息，具体配置请咨询网关提供商，。

3 Demo 程序说明

3.1 Demo 简介

ASR6505 SDK 主要包含三个样例工程，位于 Projects 目录下：

1) ClassA 示例工程

lorawan 工程来自 Semtech 的 classA 示例程序，主要为标准 LoRaWan 的通信提供示例。

2) pingpong 示例工程

pingpong 工程来自 Semtech 的代码，主要为点对点通信提供示例；

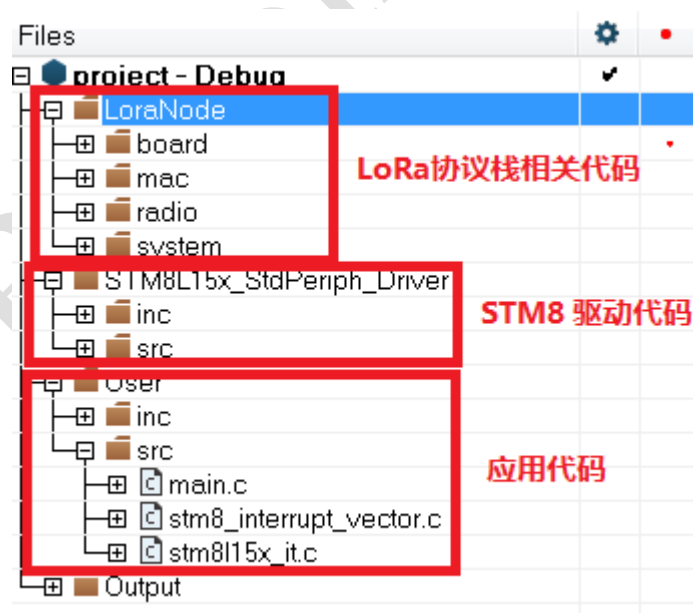
3) ClassC 示例工程

描述如何进行 LoRaWan ClassC 的配置和通信。

3.2 代码结构

如下图所示，ClassA 工程主要分为下面几个部分：LoRa 协议栈相关代码，STM 驱动代码和应用代码。

其中 LoraNode 目录下为 LoRa 协议栈相关代码，其主体为 Semtech 的 LoRaMac-Node 部分代码，其中添加了 ASR6505 的目录；STM88L15x_StdPeriph_Driver 目录下为 STM8 驱动部分代码；User 目录下为应用相关代码。

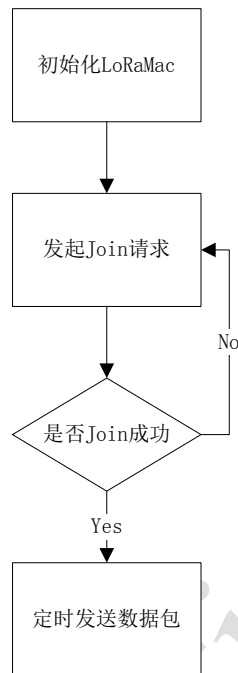


3.3 代码流程

Demo 代码中的流程比较简单，主要流程如下：

1) 初始化 LoRaMac 协议栈；

- 2) 根据设置的三元组信息，发起 Join 请求；
- 3) Join 成功后，起定时器，定时发送 LoRa 数据包；
- 4) 如果 Join 不成功，则继续发送 Join 请求。



3.4 LoRaWan 配置与修改说明

3.4.1 三元组修改

Demo 程序中，三元组信息可以在 LoRaMac-node\src\apps\LoRaMac\classA\ASR6505\Commissioning.h 中进行修改。

3.4.2 入网方式修改

Demo 程序中，入网方式可以在 LoRaMac-node\src\apps\LoRaMac\classA\ASR6505\Commissioning.h 中进行修改。OVER_THE_AIR_ACTIVATION 为 1，使用 OTAA 的入网方式。OVER_THE_AIR_ACTIVATION 为 0，则使用 ABP 的入网方式。

3.4.3 ClassC 修改

Demo 程序中默认是 Class A 类型，如需修改为 Class C 类型，请在 LoRaMacInitialization 之后，调用 LoRaMacMibSetRequestConfirm 函数进行修改。示例如下，具体可参考 ClassC 工程：

```

MibRequestConfirm_t mibReq;
mibReq.Type = MIB_DEVICE_CLASS;
mibReq.Param.Class = CLASS_C;

```

```
LoRaMacMibSetRequestConfirm(&mibReq);
```

3.4.4 频道掩码修改

Demo 程序中默认的频道设置是 0-7，用户可以调用 LoRaMacMibSetRequestConfirm 函数进行修改，具体可参考 lwan_dev_params_update 函数。

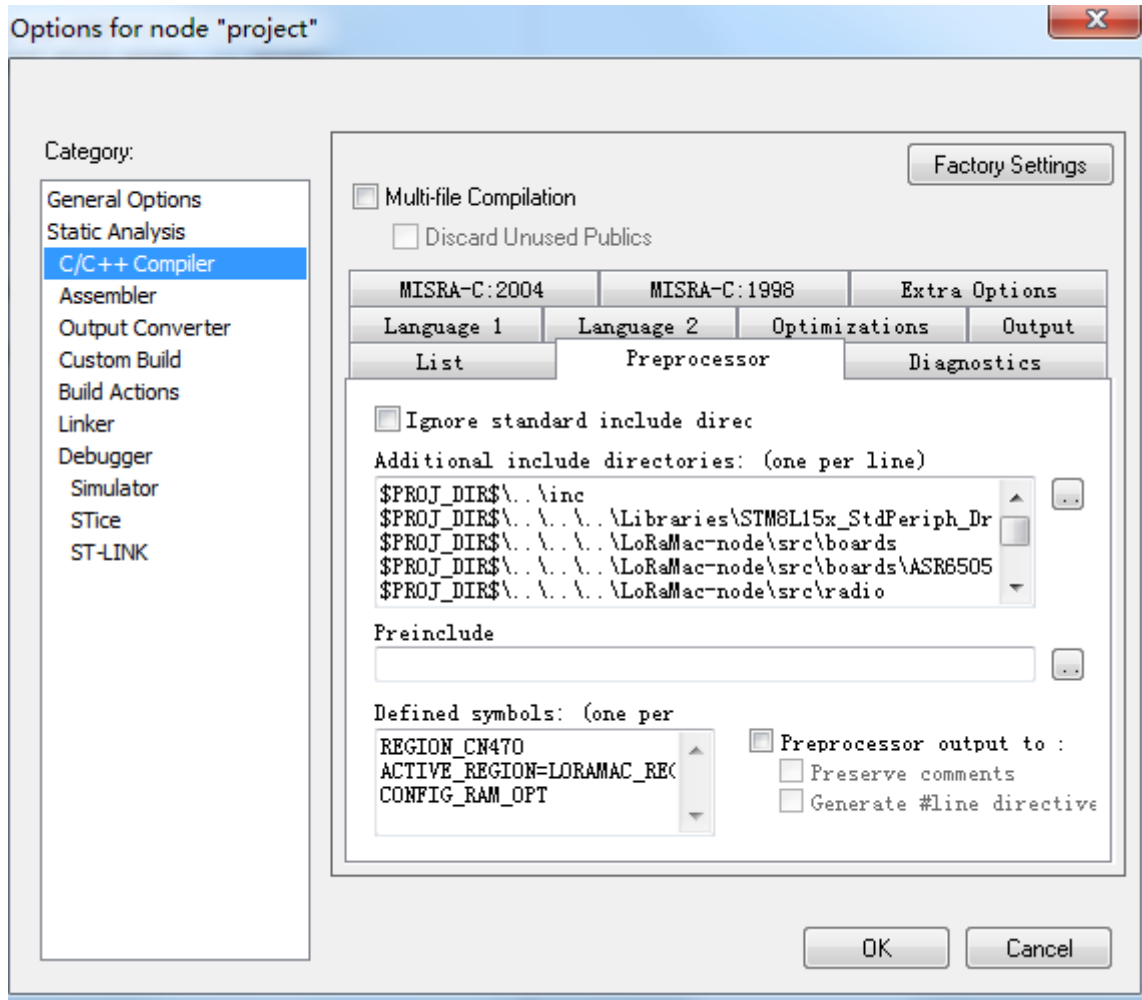
3.4.5 Mac 参数修改

其他 Mac 参数的修改，可以直接调用 LoRaMacMibSetRequestConfirm 函数。

3.4.6 Region 修改

修改 Region 时主要需要下面修改：

- 1) 去掉 RegionCN470.c 和 RegionCN470.h
- 2) 添加对应 Region 的源文件，Region 文件位于 LoRaMac-node\src\mac\region 目录下
- 3) 去掉宏 CONFIG_RAM_OPT
- 4) 修改宏 REGION_CN470 为对应 Region 的宏，在 Region.c 或者 Region.h 中可以找到所有支持的 Region 宏
- 5) 修改宏 ACTIVE_REGION 为对应 Region，参见 LoRaMac.h 中的结构体 LoRaMacRegion_t



3.5 外设使用说明

3.5.1 GPIO

GPIO 初始化

```
void GpioInit( Gpio_t *obj, PinNames pin, PinModes mode, PinConfigs config, PinTypes type, uint32_t value );
```

GPIO 读

```
uint32_t GpioRead( Gpio_t *obj );
```

GPIO 写

```
void GpioWrite( Gpio_t *obj, uint32_t value );
```

GPIO 中断

```
void GpioSetInterrupt( Gpio_t *obj, IrqModes irqMode, IrqPriorities irqPriority, GpioIrqHandler *irqHandler );
```

3.5.2 UART

UART 的使用可以参考 STM8 的示例代码（https://www.st.com/content/st_com/en/products/embedded-software/mcu-mpu-embedded-software/stm8-embedded-software/stsw-stm8016.html），其中 STM8L15x-16x-05x-AL31-L_StdPeriph_Lib\Project\STM8L15x_StdPeriph_Examples\USART 目录下有 UART 的代码示例。

3.5.3 I2C

I2C 的使用可以参考 STM8 的示例代码（https://www.st.com/content/st_com/en/products/embedded-software/mcu-mpu-embedded-software/stm8-embedded-software/stsw-stm8016.html），其中 STM8L15x-16x-05x-AL31-L_StdPeriph_Lib\Project\STM8L15x_StdPeriph_Examples\I2C 目录下面下有 I2C 的代码示例。

3.5.4 SPI

SPI 的使用可以参考 STM8 的示例代码（https://www.st.com/content/st_com/en/products/embedded-software/mcu-mpu-embedded-software/stm8-embedded-software/stsw-stm8016.html），其中 STM8L15x-16x-05x-AL31-L_StdPeriph_Lib\Project\STM8L15x_StdPeriph_Examples\SPI 目录下面有 SPI 的示例代码。

3.5.5 Timer

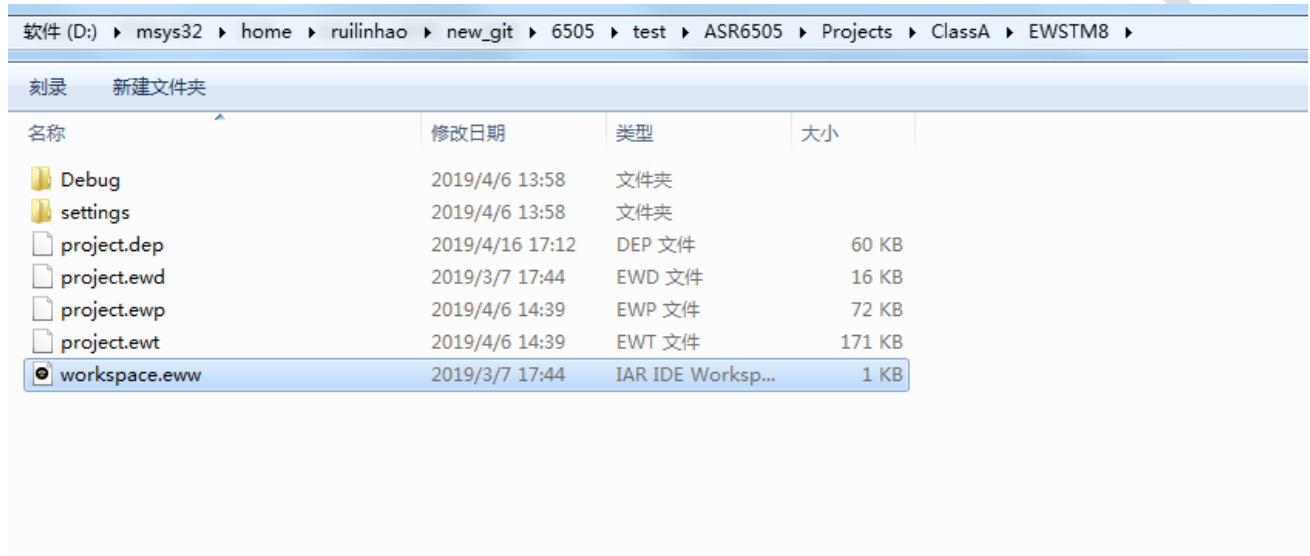
Timer 的使用可以直接调用 LoRaMac-node\src\syste\timer.h 中的接口。具体可以参考 classA 工程中的 TxNextPacketTimer

4 软件编译与烧录

4.1 IAR 环境开发

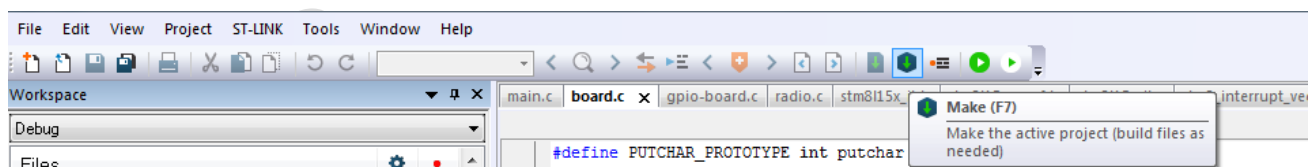
4.1.1 打开工程

首先在 ASR6505\Projects\ClassA\EWSTM8 目录下打开.eww 的工程文件。



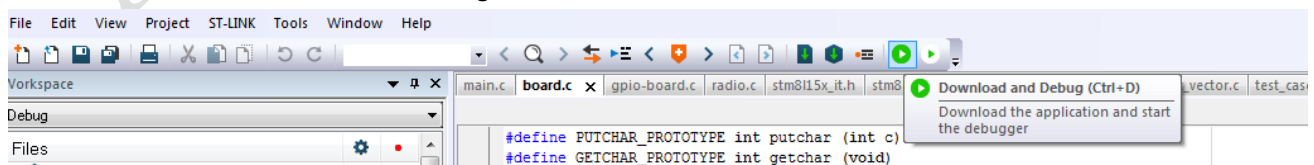
4.1.2 编译

在工具栏选择“Make”图标进行编译

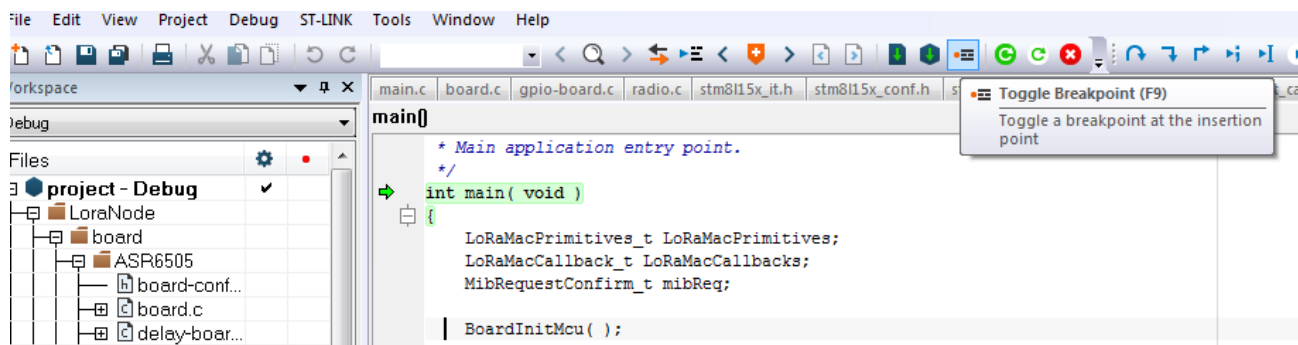


4.1.3 调试

在工具栏选择“Download and Debug”下载程序并开始调试



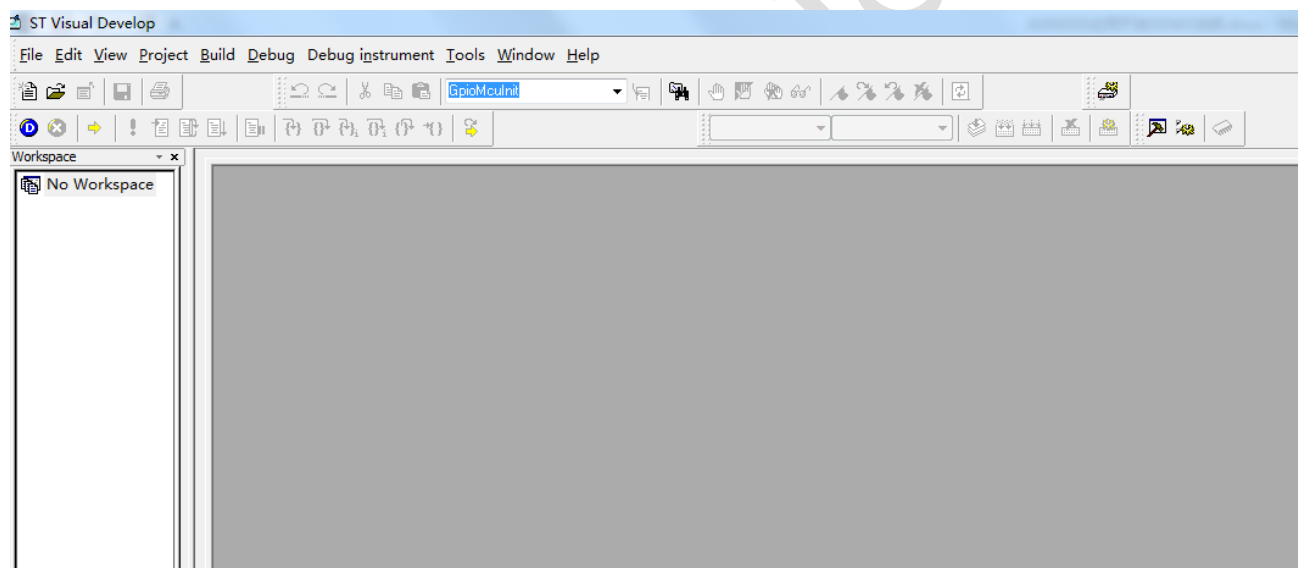
点“Toggle Breakpoint”添加/去除断点。



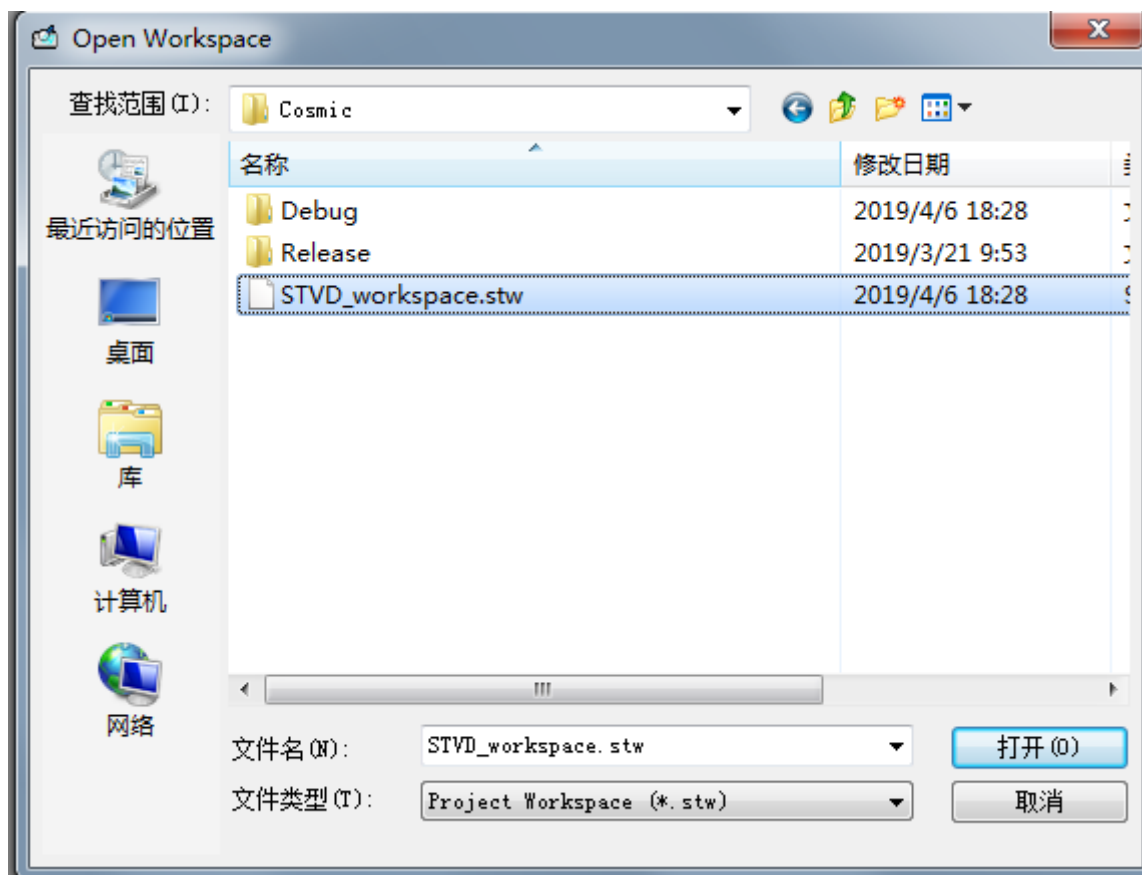
4.2 STVD 环境开发

4.2.1 打开工程

STVD 不支持直接双击工程文件进行打开，可以先打开 STVD 程序。

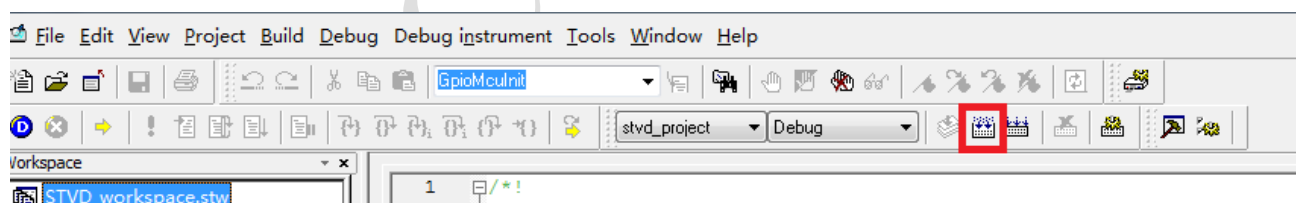


然后在 File->Open Workspace 里找到 STVD 工程，并打开



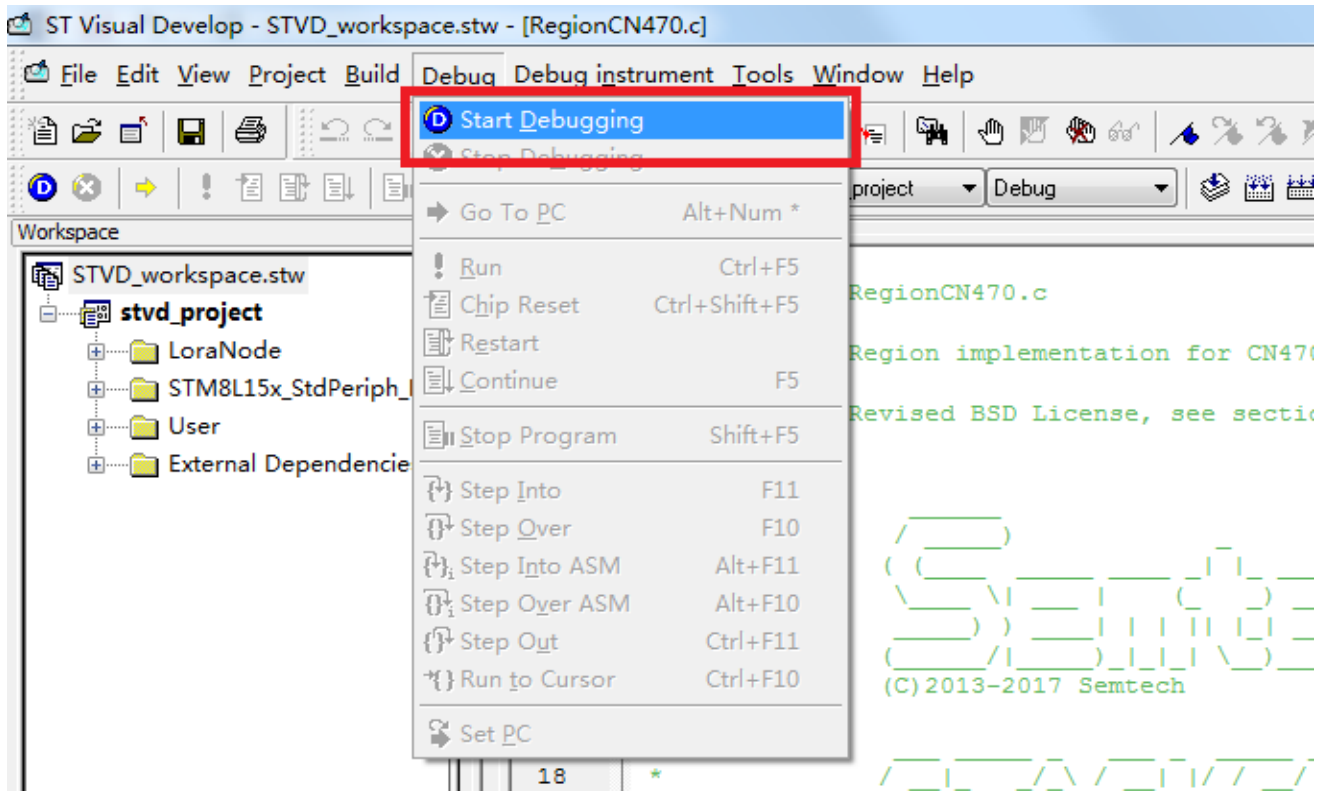
4.2.2 编译

在工具栏找到编译图标，点击进行编译。

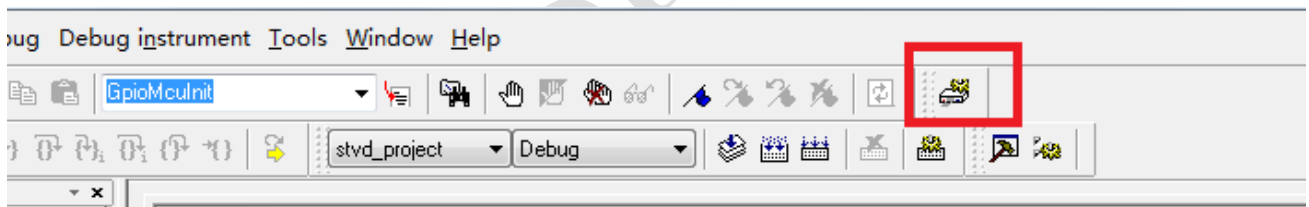


4.2.3 调试

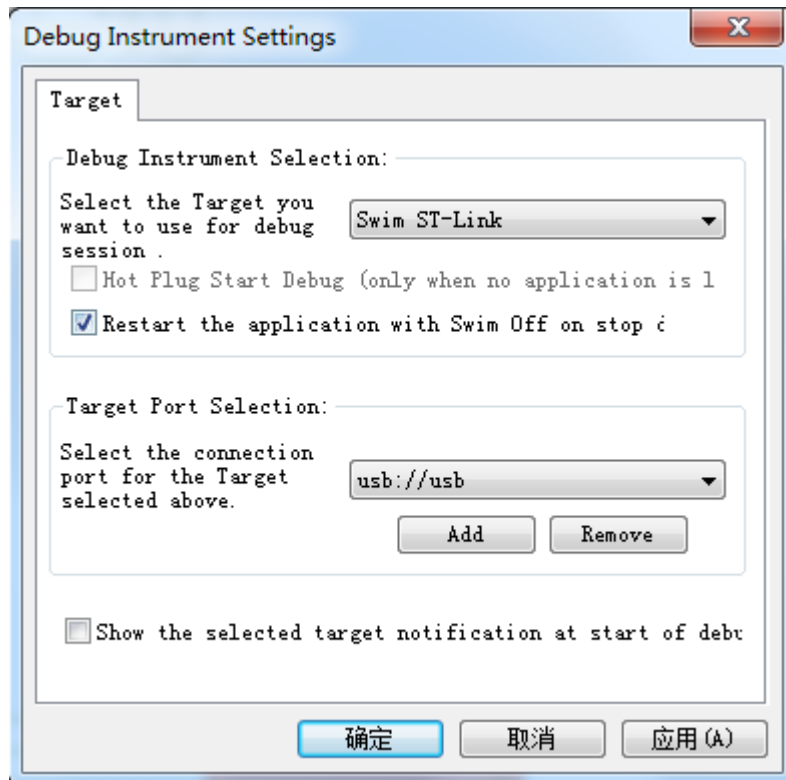
在菜单中寻找 Debug->Start Debugging，点击开始下载和调试。



第一次使用可能会遇到，设置 target 的提示，首先在工具栏找到 Target Setting 的图标



然后选择“Swim ST-Link”，点击应用和确认



4.3 UART 升级

这里主要介绍使用 STM8L152 自带的 bootloader 进行升级的操作，如有其他需求，可考虑根据 ST 的文档自行开发 bootloader。

4.3.1 准备

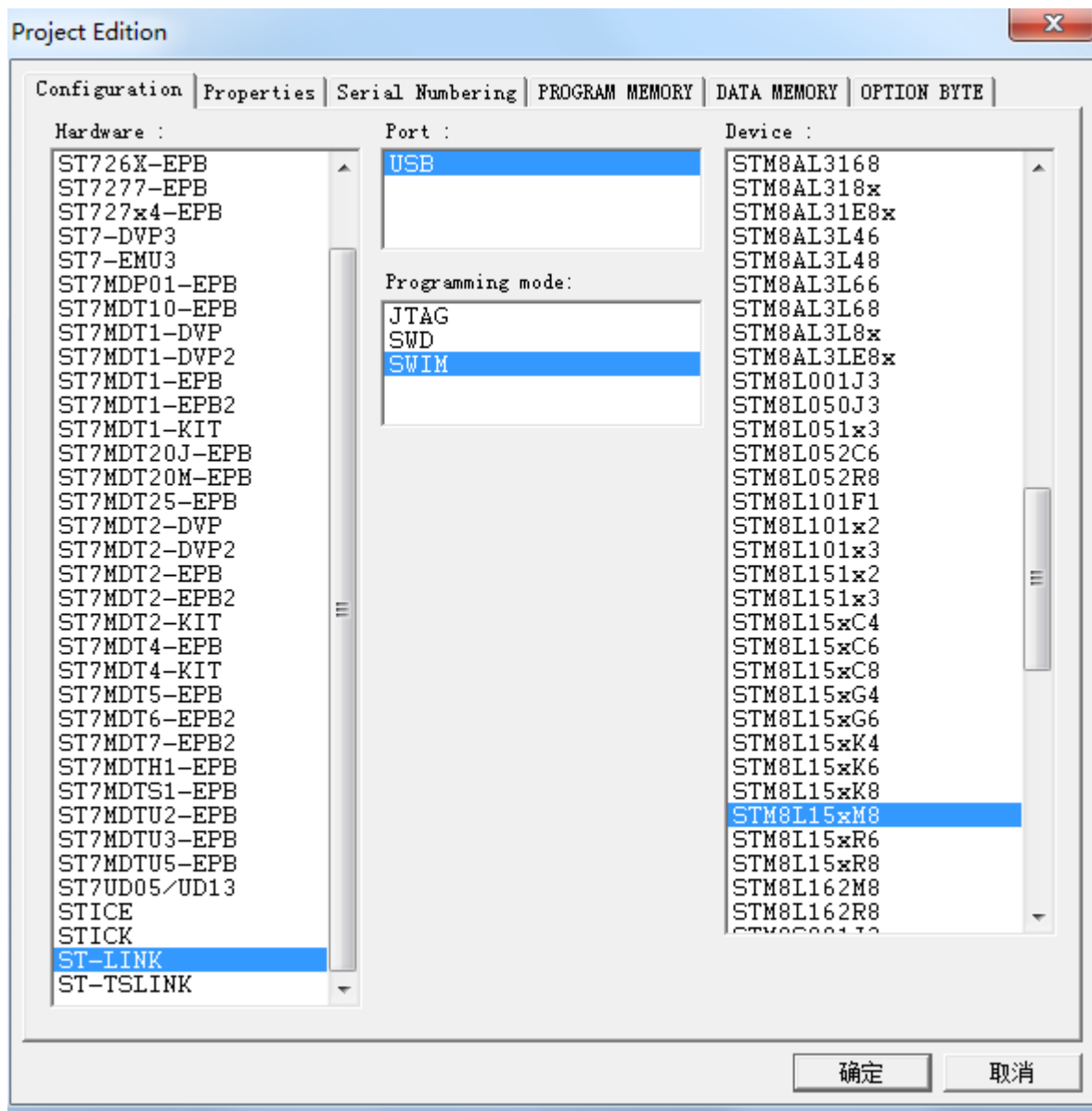
- 1) 安装 STM FlashLoader Demonstrator (https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm8-software-development-tools/stm8-programmers/flasher-stm8.html)
- 2) 安装 STVP (安装 STVD 时会同时安装)

4.3.2 设置 Bootloader Check

设置 bootloader check 可以通过 STVP 设置，也可以通过代码设置，下面就两种情况分别进行描述。

- 1) 使用 STVP 设置

打开 STVP 程序，并新建 STM8L152x8 的工程



打开 Option Byte 页面。

The screenshot shows the STM8L15xM8 configuration tool interface. The left pane displays the project configuration and memory status. The right pane shows the option byte configuration table. The bottom status bar has tabs for PROGRAM MEMORY, DATA MEMORY, and OPTION BYTE, with the latter being selected and highlighted with a red box.

PROJECT:
FileName: stm8l.stp

CONFIGURATION: *
Hardware: ST-LINK
Programming mode: SWIM
Device name: STM8L15xM8
Port: USB

PROGRAM MEMORY status:
[0x008000 - 0x017FFF]
No File
Not programmed
Memory checksum: 0x0

DATA MEMORY status:
[0x001000 - 0x0017FF]
No File
Not programmed
Memory checksum: 0x0

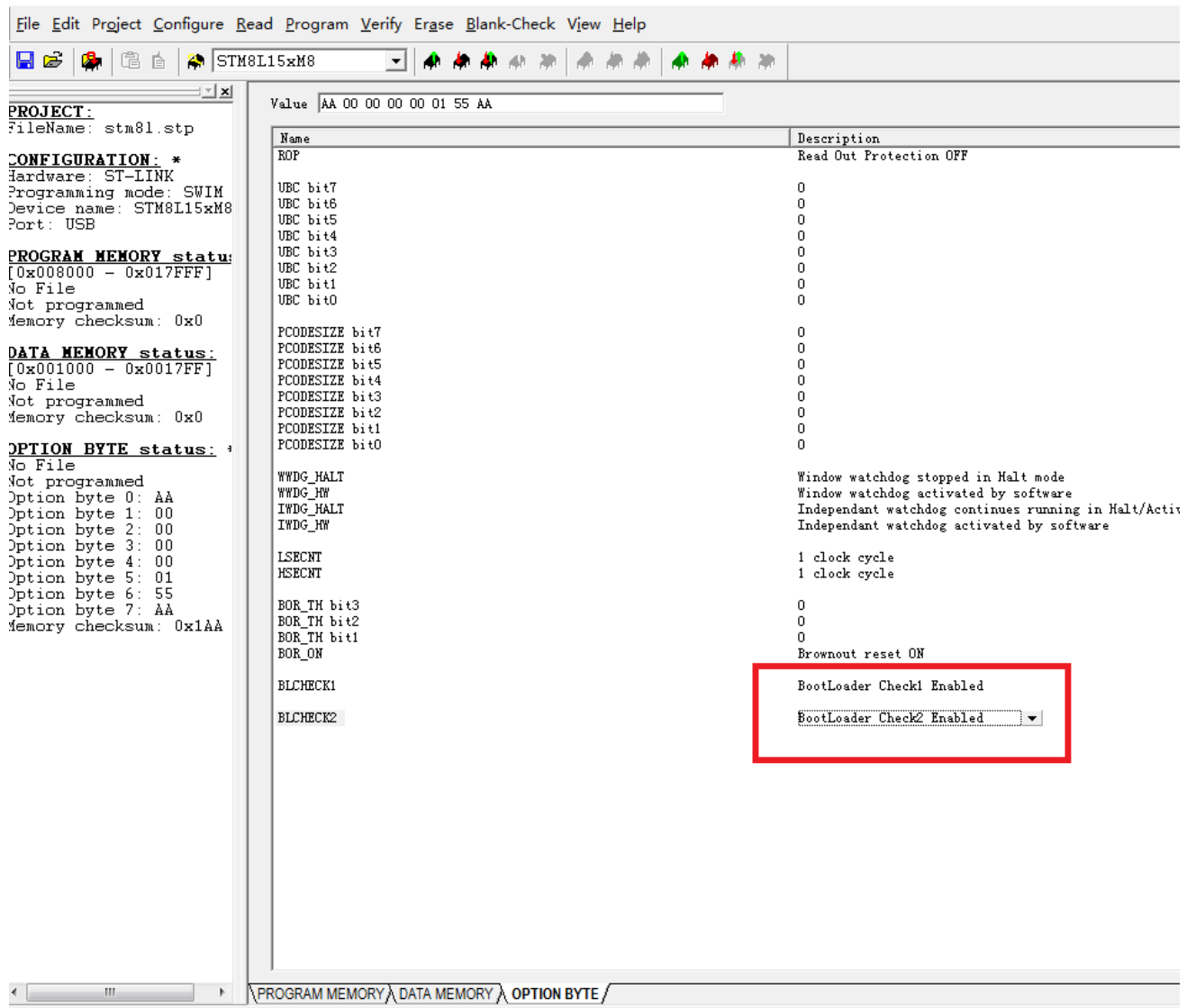
OPTION BYTE status:
No File
Not programmed
Option byte 0: AA
Option byte 1: 00
Option byte 2: 00
Option byte 3: 00
Option byte 4: 00
Option byte 5: 01
Option byte 6: 00
Option byte 7: 00
Memory checksum: 0xAB

Value: AA 00 00 00 01 00 00

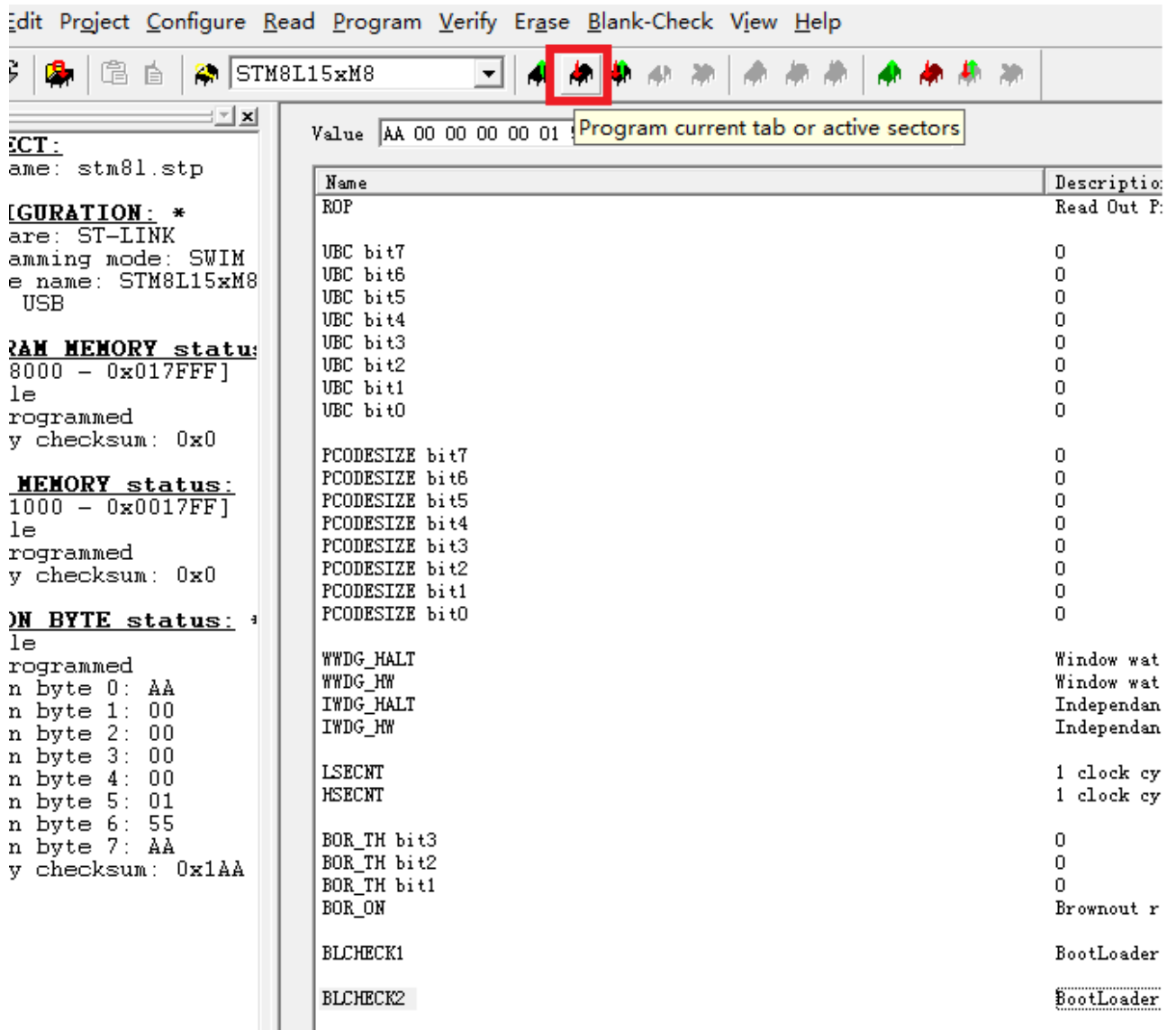
Name	Description
ROP	Read Out Protection OFF
UBC bit7	0
UBC bit6	0
UBC bit5	0
UBC bit4	0
UBC bit3	0
UBC bit2	0
UBC bit1	0
UBC bit0	0
PCODESIZE bit7	0
PCODESIZE bit6	0
PCODESIZE bit5	0
PCODESIZE bit4	0
PCODESIZE bit3	0
PCODESIZE bit2	0
PCODESIZE bit1	0
PCODESIZE bit0	0
WWDG_HALT	Window watchdog stopped in Halt mode
WWDG_HW	Window watchdog activated by software
IWDG_HALT	Independent watchdog continues running
IWDG_HW	Independent watchdog activated by software
LSECNT	1 clock cycle
HSECNT	1 clock cycle
BOR_TH bit3	0
BOR_TH bit2	0
BOR_TH bit1	0
BOR_ON	Brownout reset ON
BLCHECK1	BootLoader Check1 Disabled
BLCHECK2	BootLoader Check2 Disabled

PROGRAM MEMORY / DATA MEMORY / **OPTION BYTE**

修改 BLCHECK1 和 BLCHECK2 为 enabled



修改后，点击写入。



2) 代码设置

使用代码设置，可以参考下面代码。

```
FLASH_SetProgrammingTime(FLASH_ProgramTime_Standard);
FLASH_Unlock(FLASH_MemType_Data);

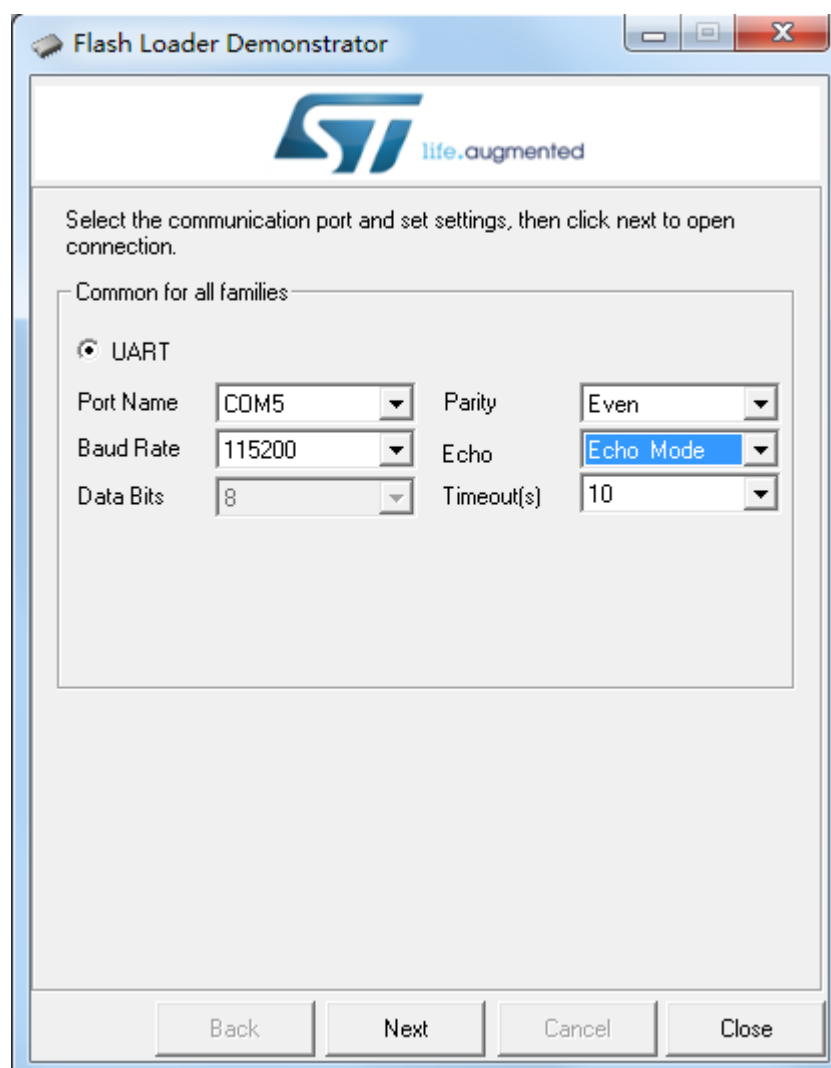
FLASH_ProgramOptionByte(0x480B,0x55); //bootloader enable
FLASH_ProgramOptionByte(0x480C,0xAA); //bootloader enable

FLASH_Lock(FLASH_MemType_Data);
```

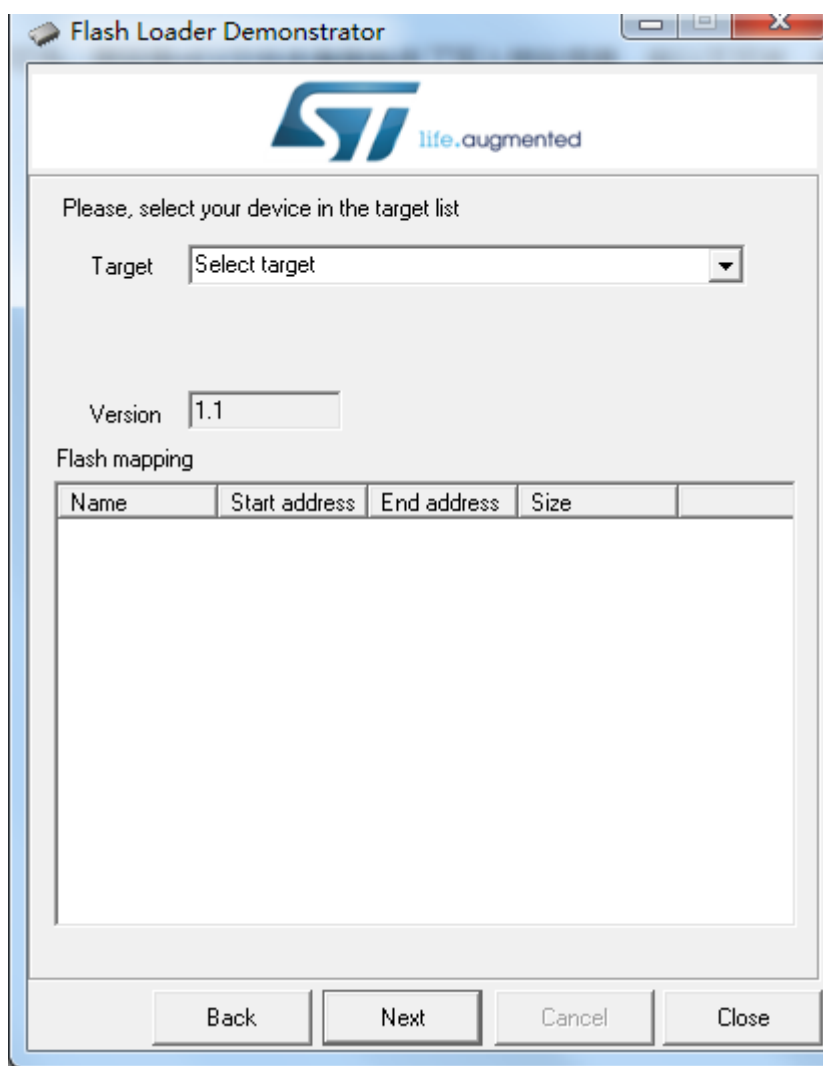
4.3.3 升级固件

1) 打开 Flash Loader Demonstrator，注意串口号不要太大（100 以上可能会有问题），另外设置 Echo

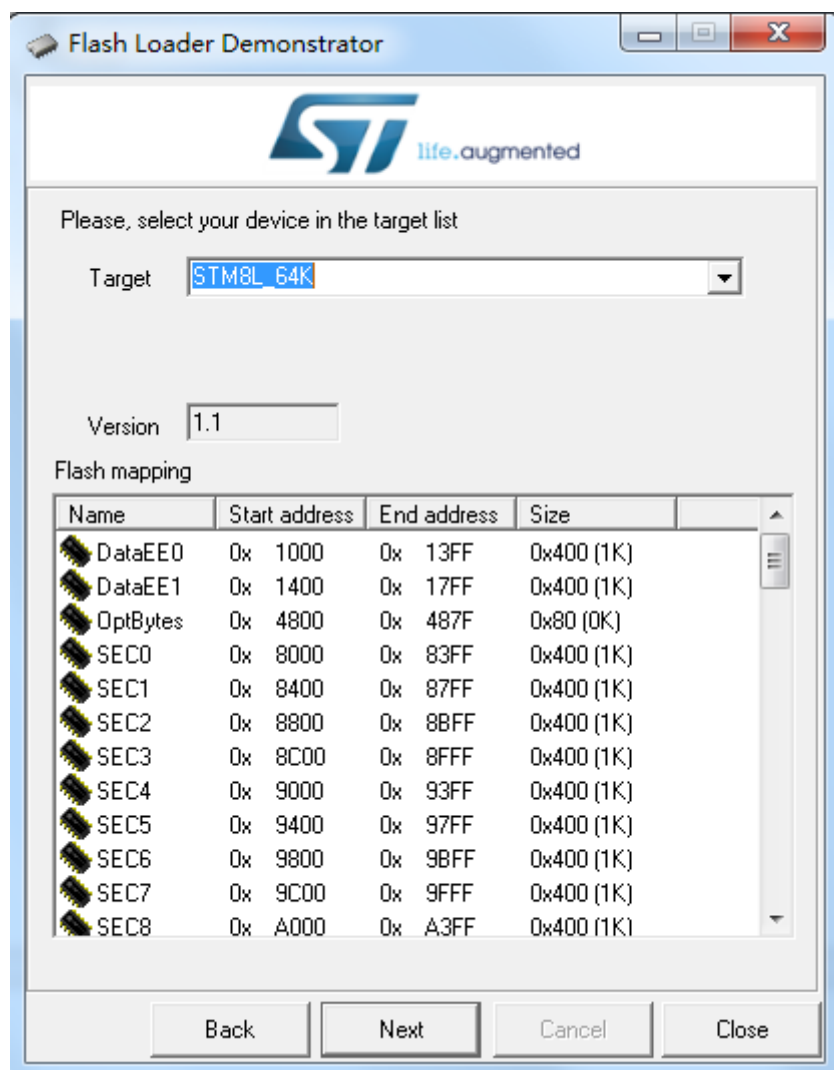
Mode。



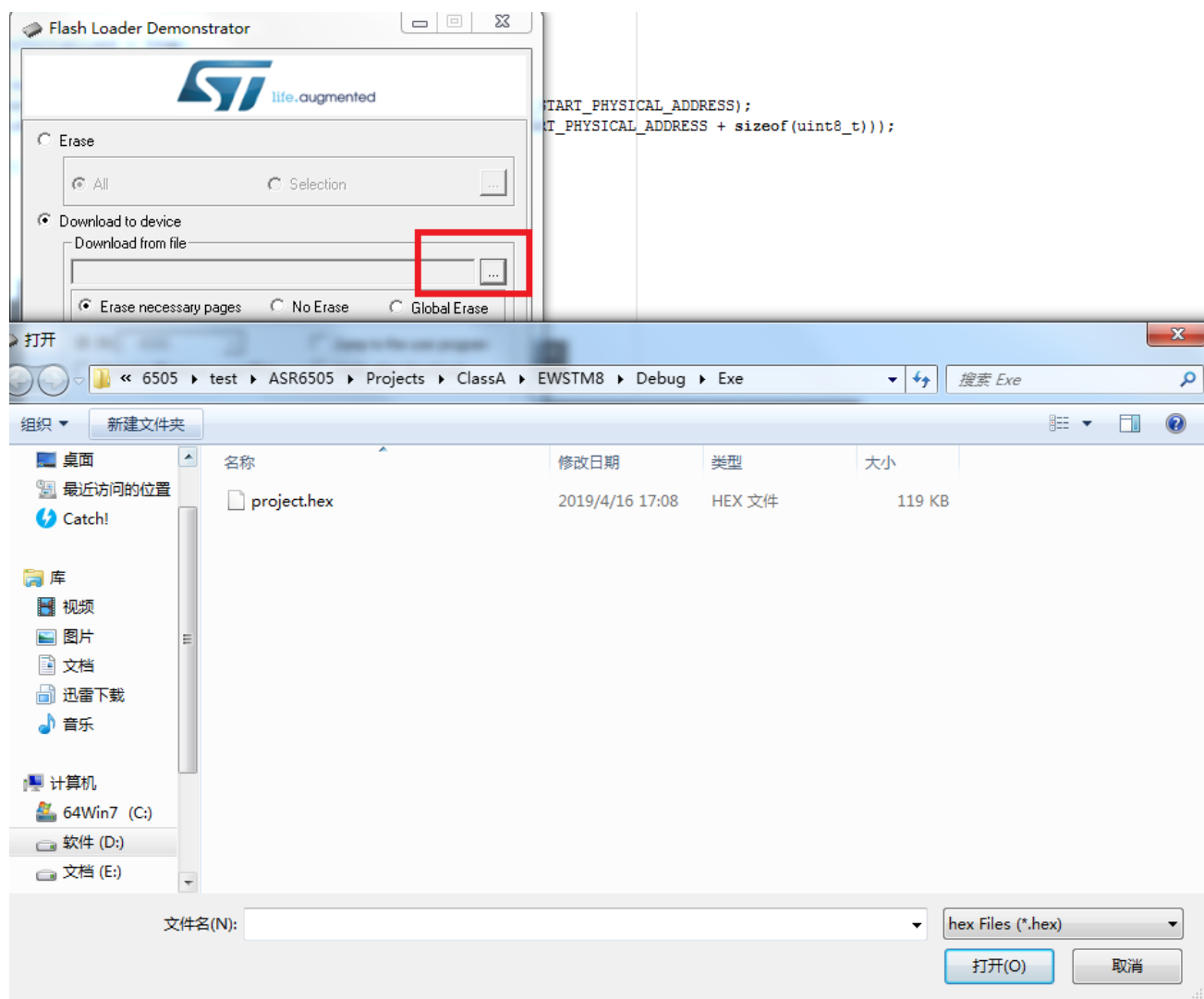
2) 复位 ASR6505, 并在 1s 内点 Next



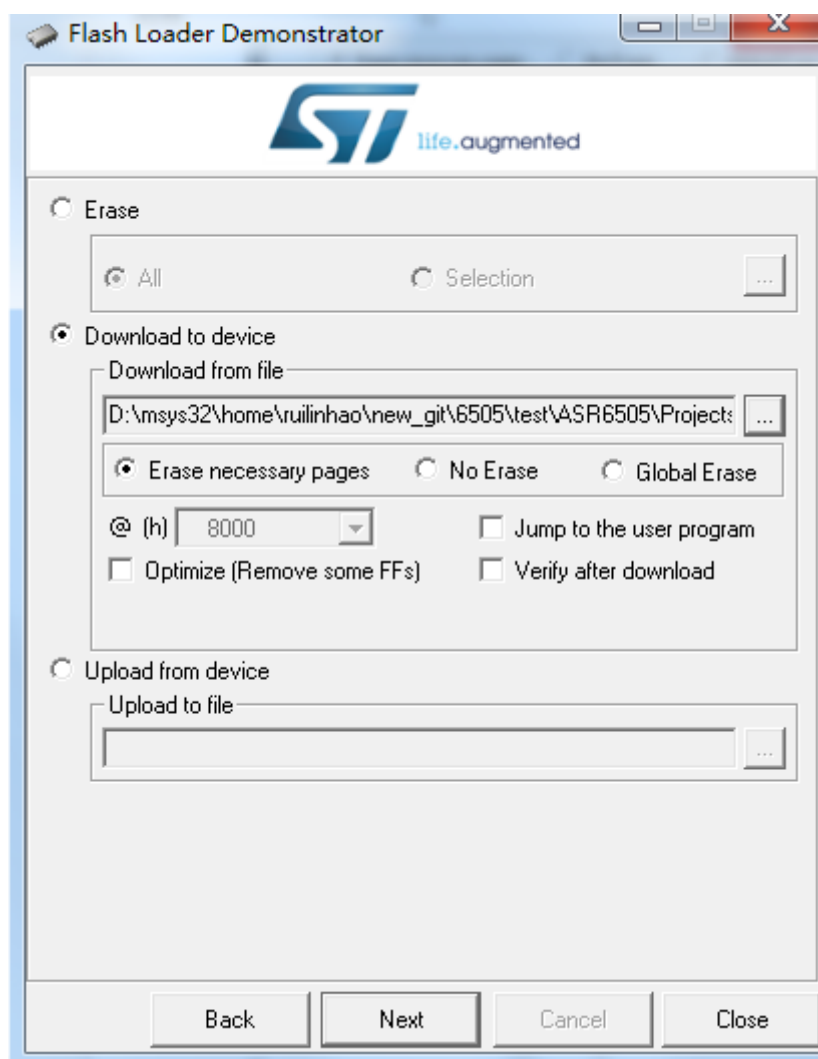
3) 选择 Target 为 STM8L_64K, 这里注意不是 STM8_64K



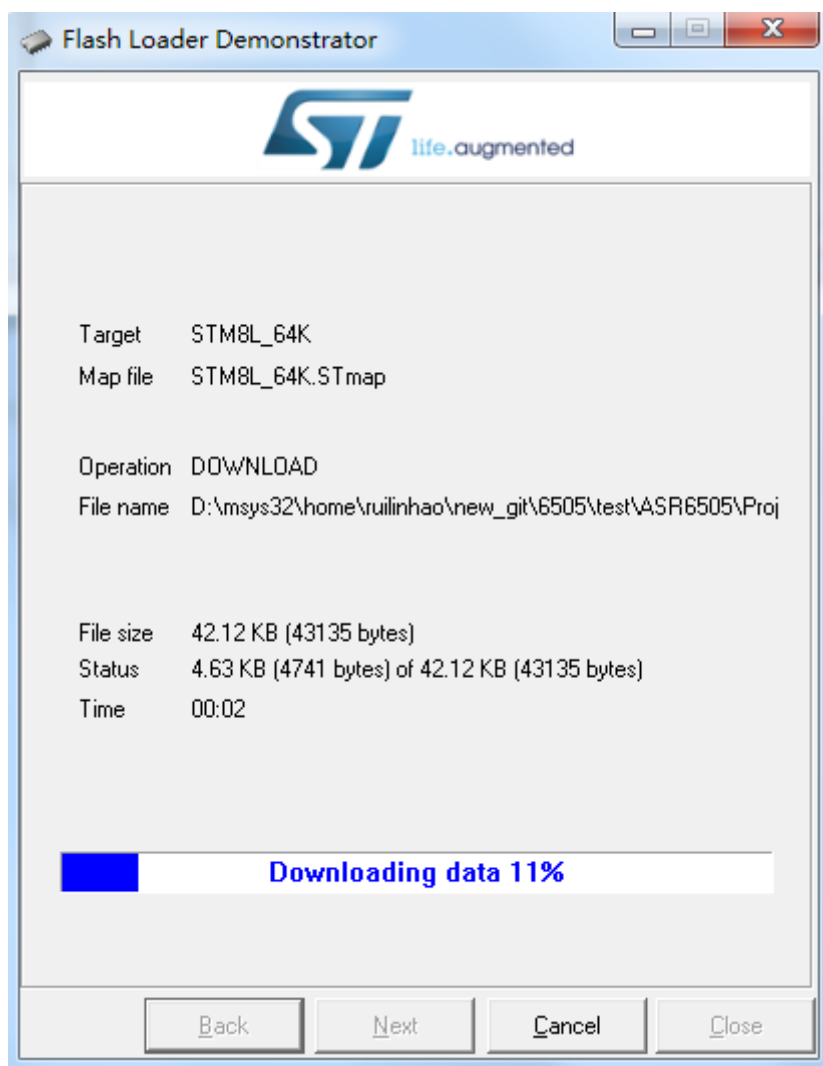
找到待升级文件，支持的文件格式为 hex, bin 和 s19。



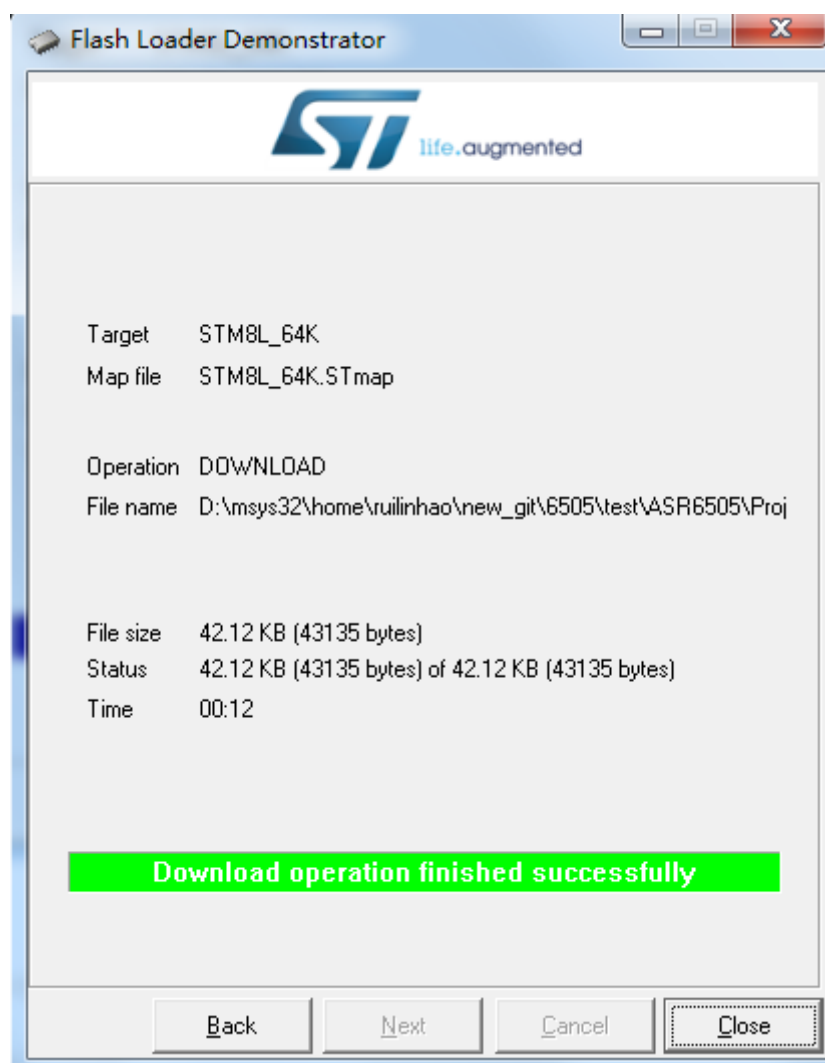
点击 Next。



点击 **Next** 开始升级。



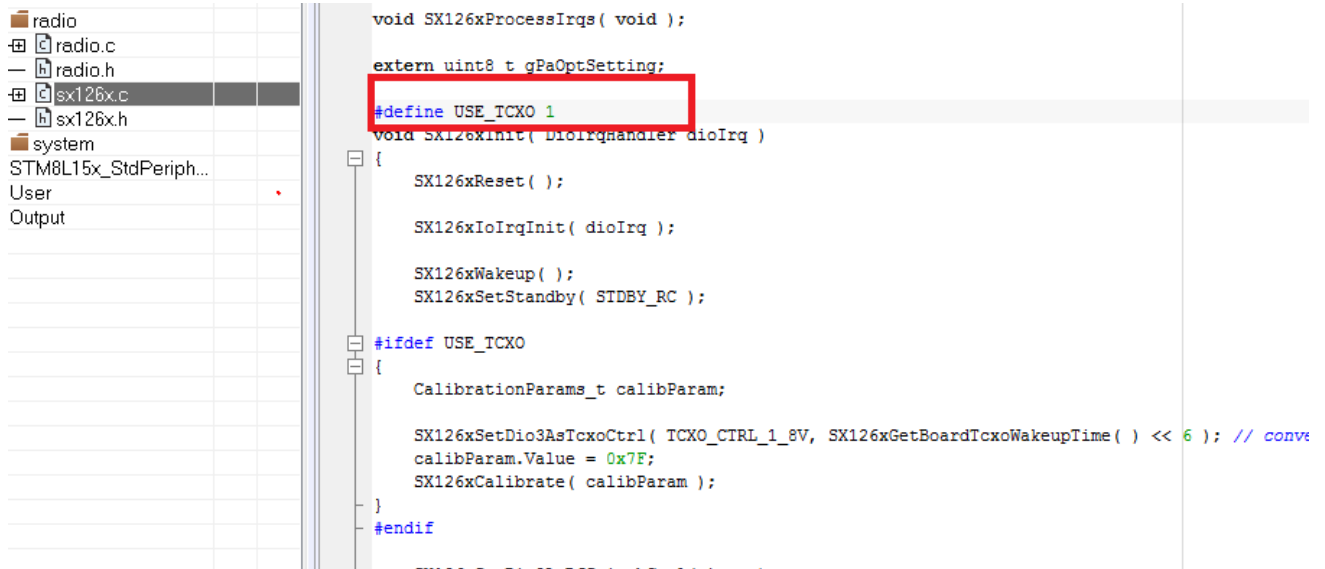
最后升级成功。



5 Q&A

5.1 如何修改 SDK 支持 TCXO 晶振？

请在 `sx126x.c` 中打开 `USE_TCXO` 宏定义



The screenshot shows an IDE with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with folders 'radio', 'system', and 'STM8L15x_StdPeriph...', and files 'radio.c', 'radio.h', 'sx126x.c', 'sx126x.h', 'User', and 'Output'. The 'sx126x.c' file is selected. The code editor shows the following code:

```
void SX126xProcessIrqs( void );  
  
extern uint8_t gPaOptSetting;  
  
#define USE_TCXO 1  
  
void SX126xInit( DioIrqHandler dioIrq )  
{  
    SX126xReset( );  
  
    SX126xIoIrqInit( dioIrq );  
  
    SX126xWakeup( );  
    SX126xSetStandby( STDBY_RC );  
  
    #ifdef USE_TCXO  
    {  
        CalibrationParams_t calibParam;  
  
        SX126xSetDio3AsTcxoCtrl( TCXO_CTRL_1_8V, SX126xGetBoardTcxoWakeupTime( ) << 6 ); // conv  
        calibParam.Value = 0x7F;  
        SX126xCalibrate( calibParam );  
    }  
    #endif  
}
```