深圳富芯电子

TX 系列芯片常见问题解答（FAQ）

Rev. V2.0

修订记录

日期	版本	描　　述	修订人
2026-9-21	V2.0	TX 系列 FAQ 合并	LPC
2025-10-21	V1.9	优化页眉页脚	CYH
2023-03-09	V1.8	添加问题 12	LDS
2022-11-16	V1.7	修改问题 5 的内容描述	HLW
2022-07-28	V1.6	修改为 TX8C101x，统一了 TX8C1010 和 TX8C1011	HLW
2022-05-24	V1.5	更新问题 11	HLW
2022-01-15	V1.4	更新问题 8 和问题 9	HLW、LDS
2021-11-19	V1.3	更新问题 6 和问题 7	LDS
2021-09-14	V1.2	添加问题 4 和问题 5	HLW、LDS
2021-09-03	V1.1	添加 TX8C101x 烧录口使用为 GPIO 的 FAQ	HLW
2020-08-01	V1.0	初始版本	HLW

目 录

1. TX 系列入门-手册 5
1.1. TX 系列入门手册-资料获取 5
1.2. TX 系列入门手册-开发环境安装 6
1.3. TX 系列入门手册-开发工具 13
1.4. TX 系列入门手册-仿真操作 17
2. TX 系列通用 FAQ 20
2.1. 挂大电容在板烧录失败 20
2.2. 新建 keil 工程 23
2.3. 烧录口 HCK、HDA 使用为 GPIO 的限制条件 32
2.4. 烧录口使用为 GPIO 后下载异常问题 35
2.5. 烧录方式 36
2.6. 读取 CodeCRC 的功能说明 40
2.7. Sleep 模式下使用外部晶振实现循环计时 43
2.8. 代码空间扩展 44
2.9. 使能加密功能后程序运行异常 50
2.10. 减小功耗将不使用的 IO 设为模拟模式 51
2.11. 函数 delay_ms 不准确的问题 52
2.12. Flash 的空间分配 53
2.13. Loader 功能 62
2.14. 在 TXPROG 量产模式下烧录异常问题 69
2.15. 将 Flash 函数加载到 RAM 上运行 72
3. TX8C101x-FAQ 78
3.1. TXPROG 在板烧录时报错 ER03 或 ER15 错误 78
4. TX8C126x-FAQ 80
4.1. TX8C126x 系列滚码功能 80

4.2. TX8C126x 系列软件加密功能 82
4.3. TX8C126x 系列使用 6K Loader 的配置说明 84
5. TX8W701x-FAQ 86
5.1. TX8W701x 系列 RF 模块 TX 发送的数据包错误率高问题 86
5.2. TX8W701x 系列 RF 模块 TRX 发送接收过程中数据包未发送或接收不到数据问题 .88
5.3. TX8W701x 系列 RF 模块 TRX 发送接收过程中数据包某 1byte 数据错误问题 89
5.4. TX8W701x 系列 RF 模块 RX 接收过程中添加异常处理 90

1. TX 系列入门-手册

1.1. TX 系列入门手册-资料获取

1.1.1. 官网资料获取

相对应资料可以到官网下载（FXMCU.COM）下载。进入网页后选择对应芯片的开发资料，下载对应的固件、手册、上位机以及 SDK 等等，如下图所示：



1.1.2. 向销售或者 FAE 工程师询问

不同芯片型号对应的技术资料存在差异，咨询时请说明具体芯片型号，告知所需资料。

1.2. TX 系列入门手册-开发环境安装

1.2.1. 开发软件

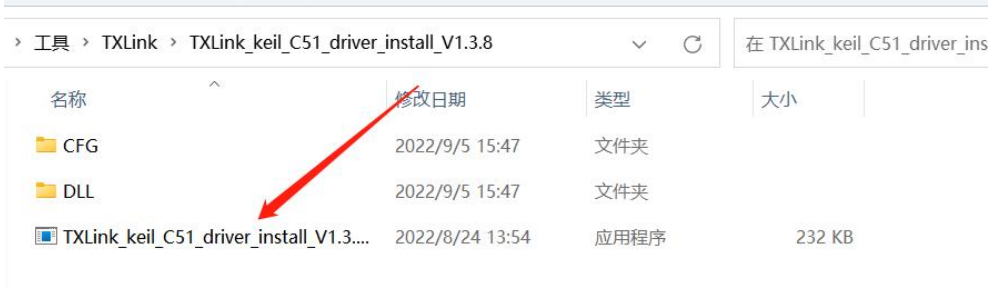
我司 MCU 系列统一使用 Keil uVision5 进行开发，32 位系列使用 ARM 版本，8 位系列使用 C51 版本。由于版本问题，我司不提供该软件安装包，需要自行前往对应官网进行下载。

1.2.2. 固件安装

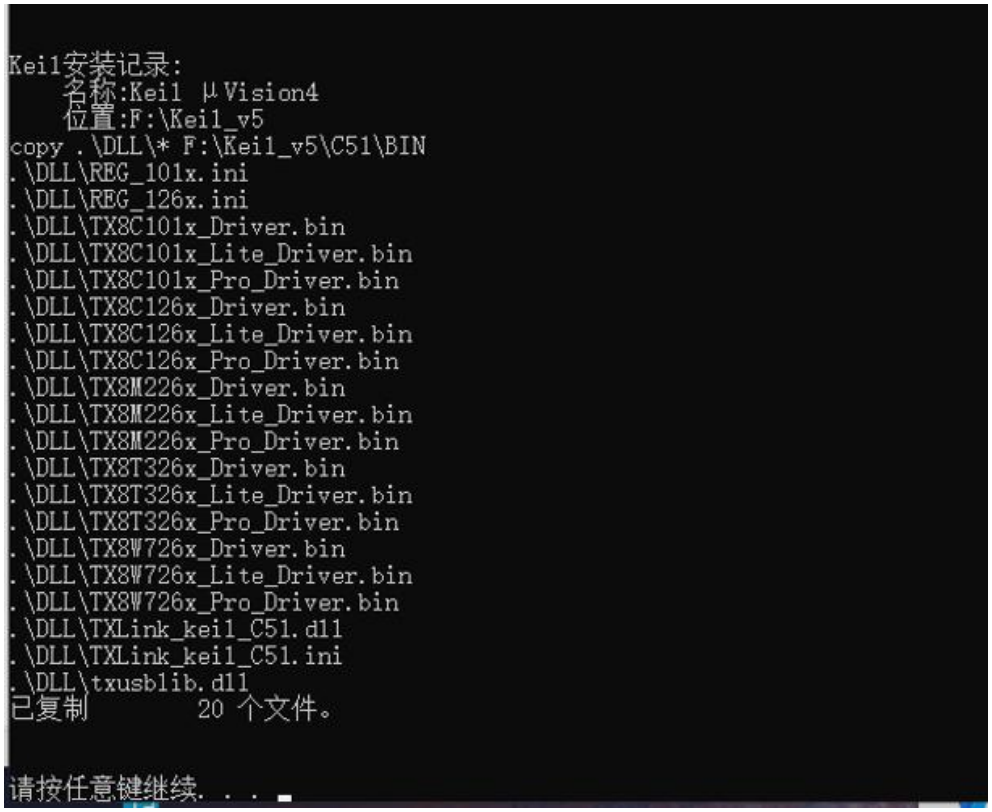
官网下载对应工具的资料包。



下载后打开文件夹运行“TXLink_keil_C51_driver_install_xxx.exe”，待安装完毕即可。



安装完毕后未看到“xxx 另一个程序正在使用此文件，进程无法访问”的提示，则认为是安装插件成功，安装失败请截图联系 FAE 工程师进行处理，如图所示：



如安装提示“找不到安装路径”等错误提示，请从如下几个方面先排查定位下问题

- 请确认电脑是否有安装好 keil C51(注意：TX 系列芯片及插件不支持 keil MDK）->安装 keil C51（建议安装在默认 C 盘路径），并关闭所有 keil 工程，重新安装插件。

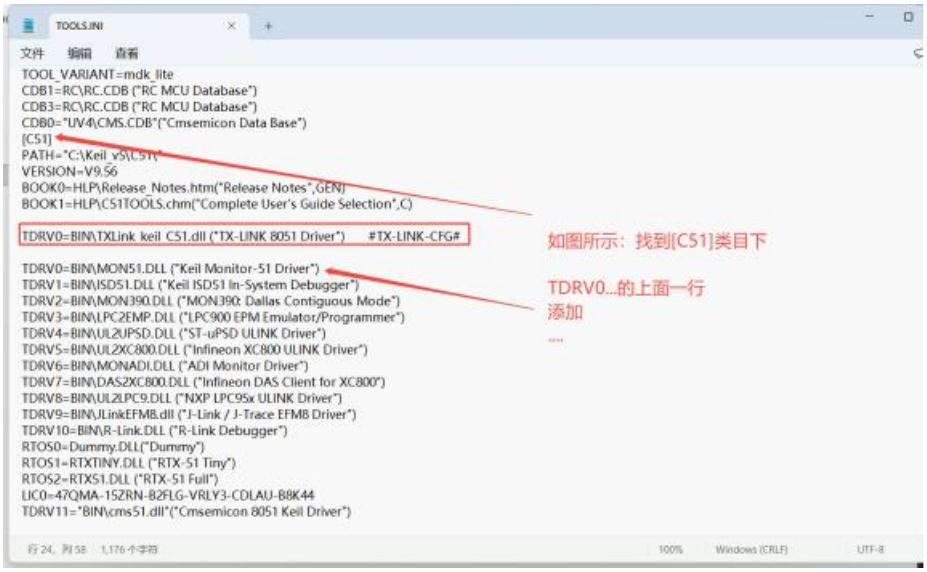
- 请确认 keil C51 是否安装在 C 盘外或自定义的路径下->建议重新安装在默认 C 盘路径下，或参考“手动安装 keil 插件指引”进行安装。

- 如果确认安装好 keil C51 并在默认 C 盘路径->参考“手动安装 keil 插件指引”进行安装，不行建议换台电脑试下。

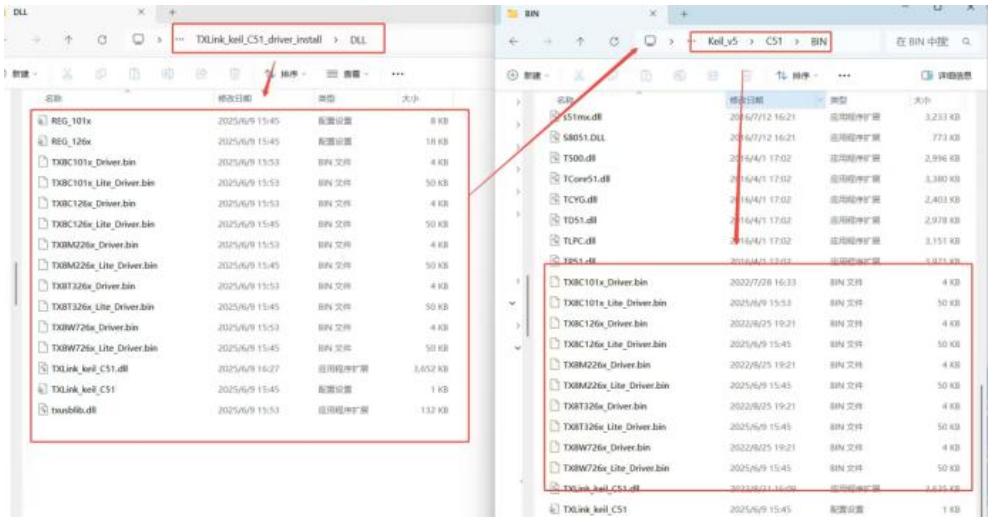
手动安装 keil 插件指引

1. 找到 keil 安装目录下的“..Keil_v5\TOOLS.INI”文件，用记事本打开，找到[C51]下，TDRVO...的上面一行，添加如下字符：

TDRV0=BIN\TXLink_keil_C51.dll("TX-LINK 8051 Driver") #TX-LINK-CFG#



2. 打开安装包/DLL 文件夹，复制全部文件，打开 keil 安装目录找到/C51/BIN，粘贴全部文件。



1.2.3. 安装 CH340 驱动

TXLink 连接到电脑，识别不到 TXLink 的 COM 口，在设备管理器中看到如图所示，需要安装 CH340 驱动。



如图所示，以管理员权限打开“TXLink_CH340_driver_V1.0.0.SETUP.EXE”，点击安装按钮，等待安装结果，安装完成如图所示：

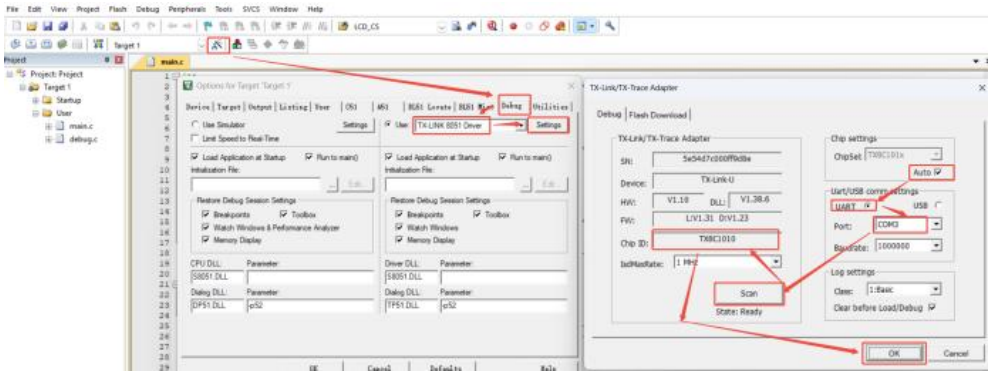


1.2.4. 设置 keil 界面

Debug

Debug 界面用于显示 TXLink 的信息以及一些仿真的配置，如图所示：

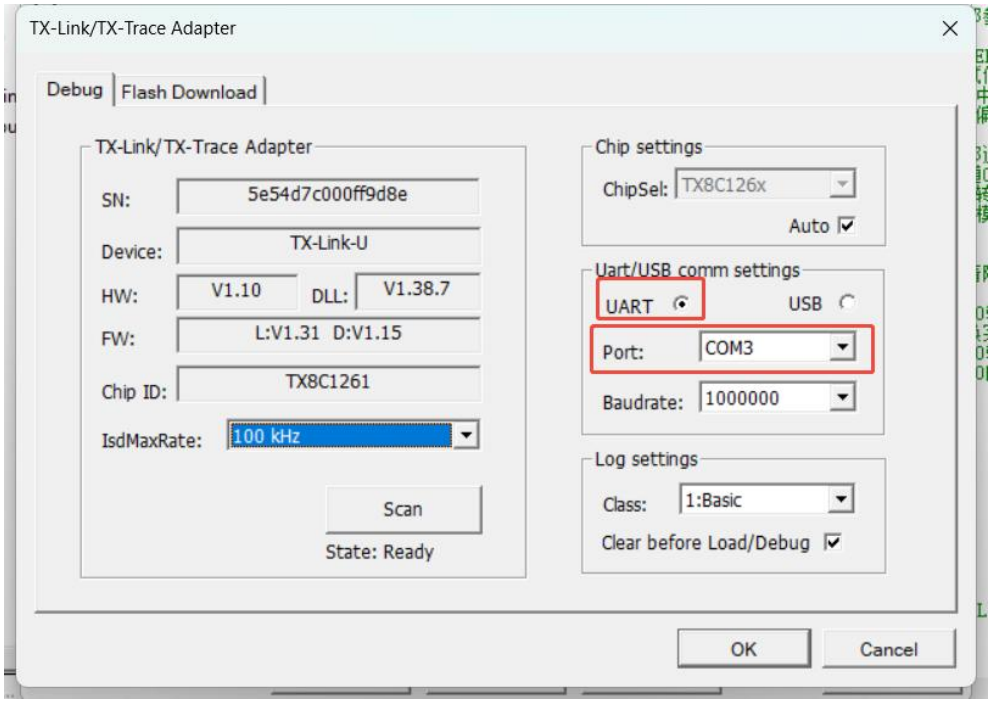
先点击 Options for Target 'Target'页面，点击 Debug，如图所示：



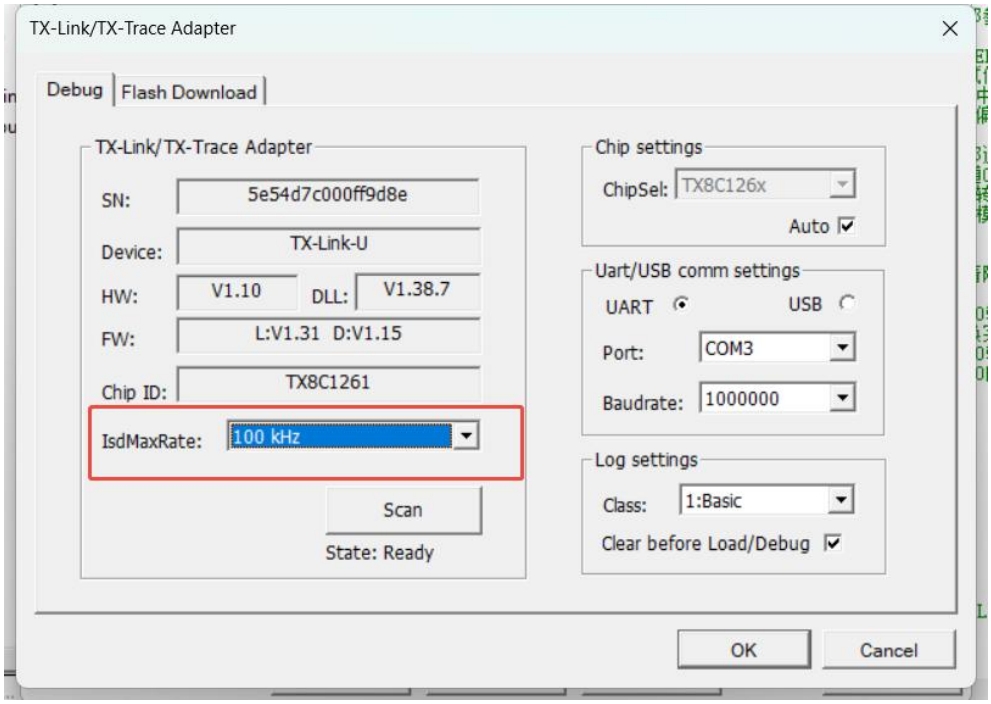
注意：如果打开此页面，Chip ID 没有出现目标芯片，请按照下面的指引逐一排查

- 硬件线路连接，或外围异常-》使用 demo 板或单芯片的空板试下，或万用表测量通断。

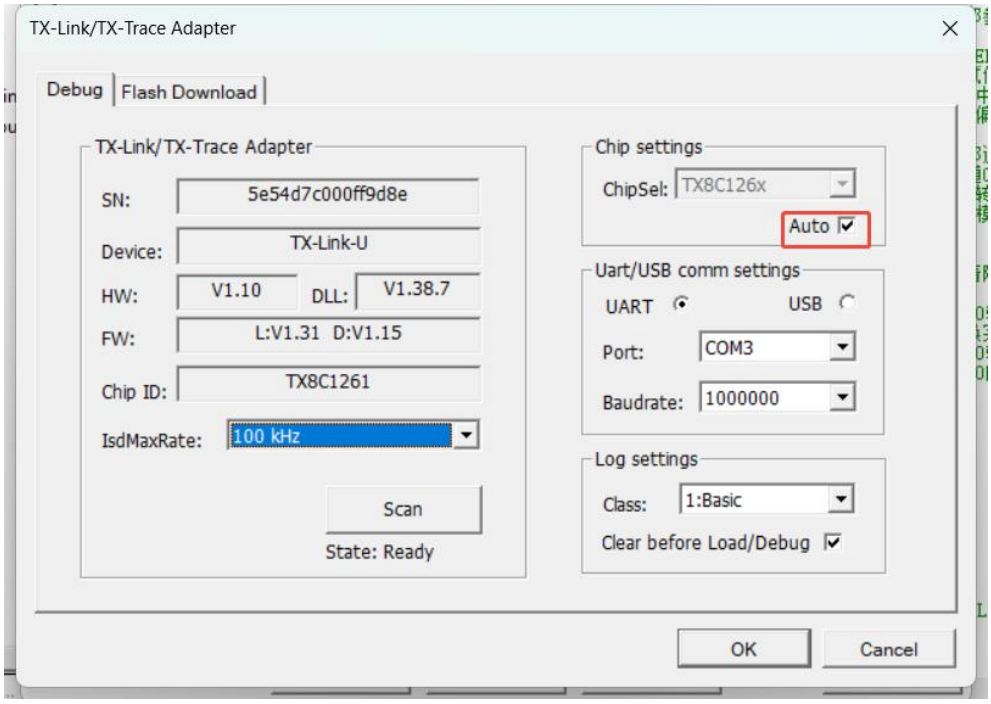
- 配置“Debug -> Uart comm settings -> Port”端口不是 TXLink 的端口-》选 UART,拔插 USB 确认选择正确端口



- 配置“Debug -> TX-Link/TX-Trace Adapter -> IsdMaxRate”的速率太高-》选 100K 速率试下，正常建议选择 1M



- 配置“Debug -> Chip settings -> ChipSel”型号与目标芯片型号不对应-》勾选 Auto



Chip settings

1. ChipSel：芯片型号选择建议勾选 Auto，此型号必须与目标芯片的型号的一致，否则会导致代码下载失败。

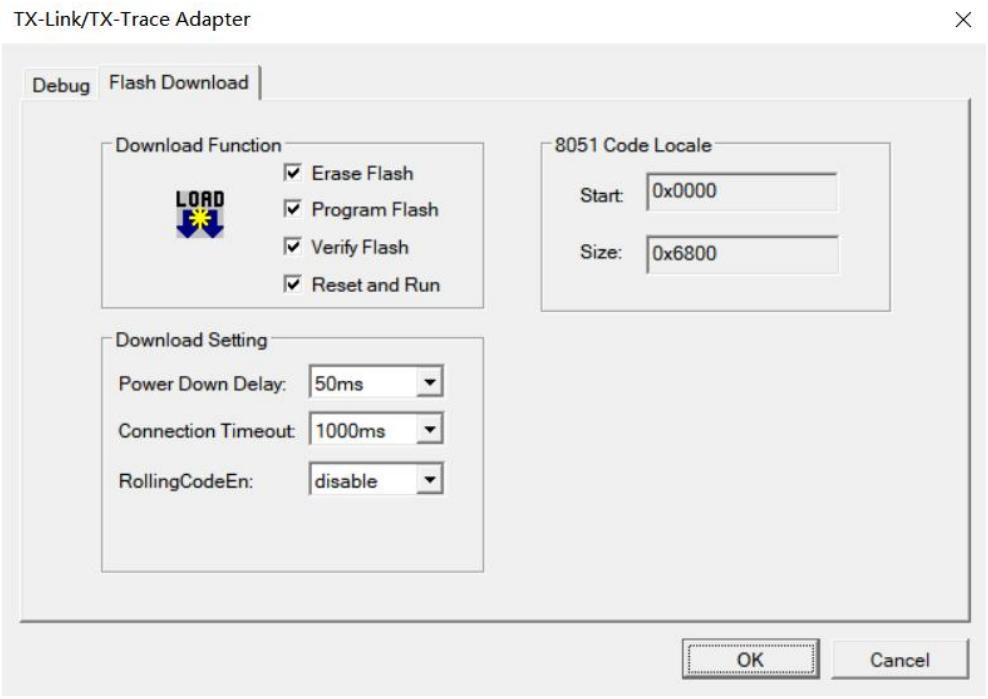
Uart comm settings

1. Port：TXLink 在 PC 端的端口号，一定要选择对应的端口号；

2. Baudrate：PC 与 TXLink 的通信速率，建议选择 1M。

Flash Download

Flash Download 界面用于配置“LOAD”程序时需要走的流程以及显示 Flash 的信息，如图所示：



Download Function

LOAD 快捷键的动作（优先级：Erase Flash > Program Flash > Verify Flash > Reset and Run）：

1. Erase Flash：勾选后，点击 LOAD 快捷键会触发全片擦除 Flash 功能；
2. Program Flash：勾选后，点击 LOAD 快捷键会触发更新代码到 Flash 功能；
3. Verify Flash：勾选后，点击 LOAD 快捷键会触发下载 Flash 后，进行校验功能；
4. Reset and Run：勾选后，点 LOAD 快捷键后，在下载 Flash 后，自动复位运行程序。

1.3. TX 系列入门手册-开发工具

1.3.1. TXLink 调试器

适用型号：TX8C101X、TX8C126X、TX8M226X、TX8W7262、TX8W7011

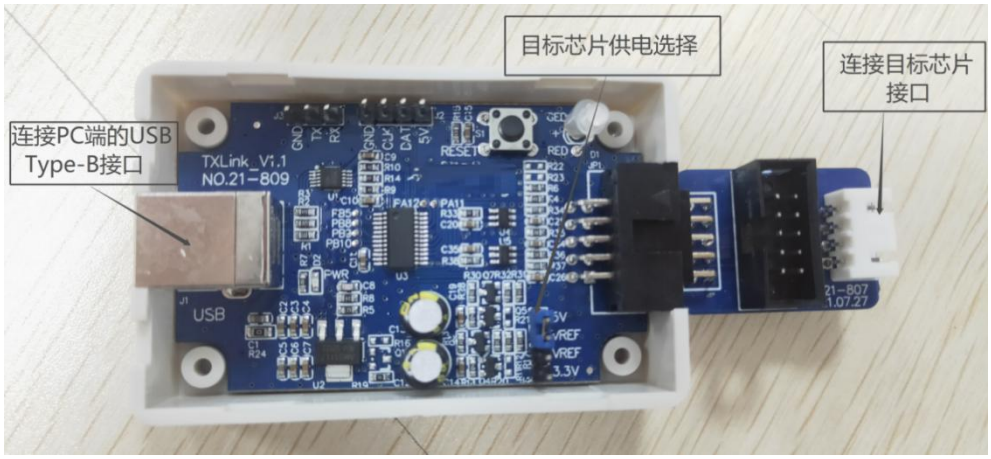
功能特性：

- 支持 keil 在线烧录、在线仿真；
- 支持 TXprogrammer.exe 专用软件，在线烧录。

J4 为通信的电平选择，连接情况可分为：

- 跳冒选择连接“5V”和“VREF”时，HCK 和 HDA 引脚输出 5V 电平；
- 跳冒选择连接“3.3V”和“VREF”时，HCK 和 HDA 引脚输出 3.3V 电平；
- 跳冒选择不连接时，参考电平由外部提供，HCK 和 HDA 引脚输出 VREF 电平。





1.3.2. TX-PROG 烧录器

TX-PROG 是一款带 LED 显示，集多种模式为一体的量产工具，适用于 TX 系列芯片。

适用型号：TX8C101X、TX8C126X、TX8M226X、TX8W7262、TX8W7011

功能特性：

- 支持 TXprogrammer.exe 专用软件；
- 支持脱机烧录、机台烧录。



外部功能模块介绍

- 5V 电源和开关：控制 DC-5V 供电；
- 蜂鸣器：烧录状态声音指示；
- 模式和拨码开关：负责选择烧写模式，模式切换在复位或重新上电后生效；
- 烧写按键：手动烧录模式时使用按键触发烧写；
- LED 指示灯：烧录状态 LED 指示；
- 烧写器 USB 接口：和 PC 连接更新烧写器代码和目标代码；
- 数码管：烧录状态数码管指示；
- 20Pin 牛角座：20Pin 烧录接口。

拨码功能介绍

表 3-1 烧录器拨码功能表

拨码开关	模式	备注
0000	手动烧录	正常烧录流程
0001	自动烧录	正常烧录流程
0010	空片校验	检测芯片是否为空片
0011	芯片擦除	芯片全片擦除
0100	机台烧写	连接机台烧录
0101	芯片校验	校验芯片代码
0110	串口升级	单线串口离线升级
0111	扩展烧录模式	部分芯片不支持
1000	读 CRC 模式	读取芯片代码 CRC 码
1001	机台校验模式	使用机台做芯片校验
1001-1110	保留	
1111	测试模式	芯片 short 测试



读 CRC 或芯片校验功能

- TX-PROG 量产模式下烧录的芯片用“1000”(读 CRC 模式)

- TX-PROG 调试模式下烧录的芯片用“0101”

(芯片校验模式)



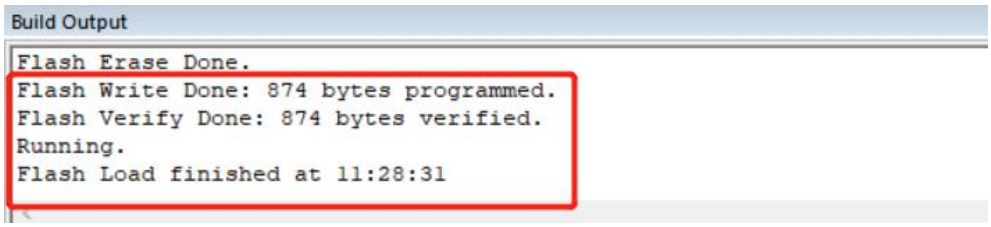
1.4. TX 系列入门手册-仿真操作

1.4.1. Download 程序

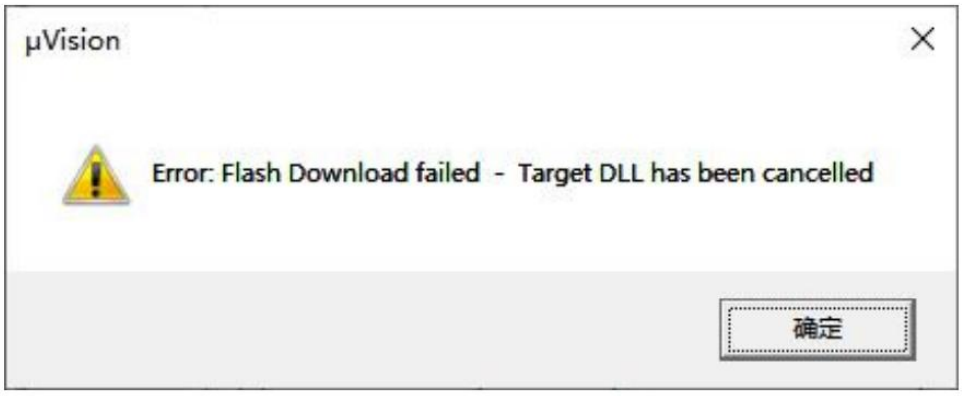
当程序编译通过后，点击快捷图标“LOAD”或者按下快捷键“F8”，如图所示：



完成程序的代码烧录，烧录过程与“Flash Download -> Download Function”配置相关，并在“Build Output”窗口输出相应的信息，如图所示：

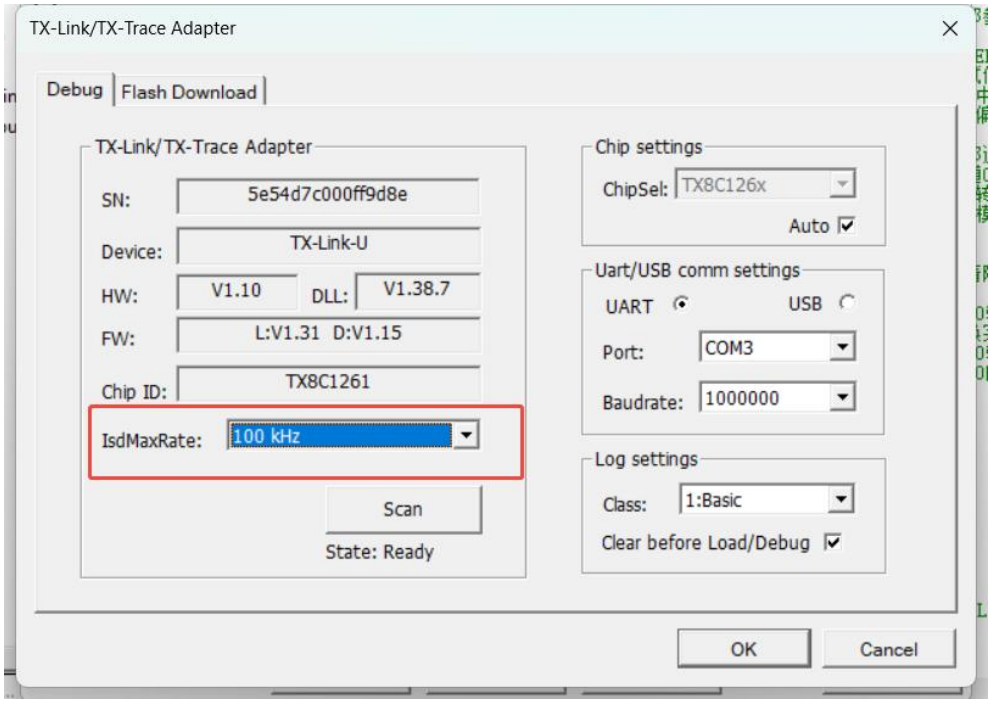


当 Download 失败时，会输出错误的提示信息，如图所示：

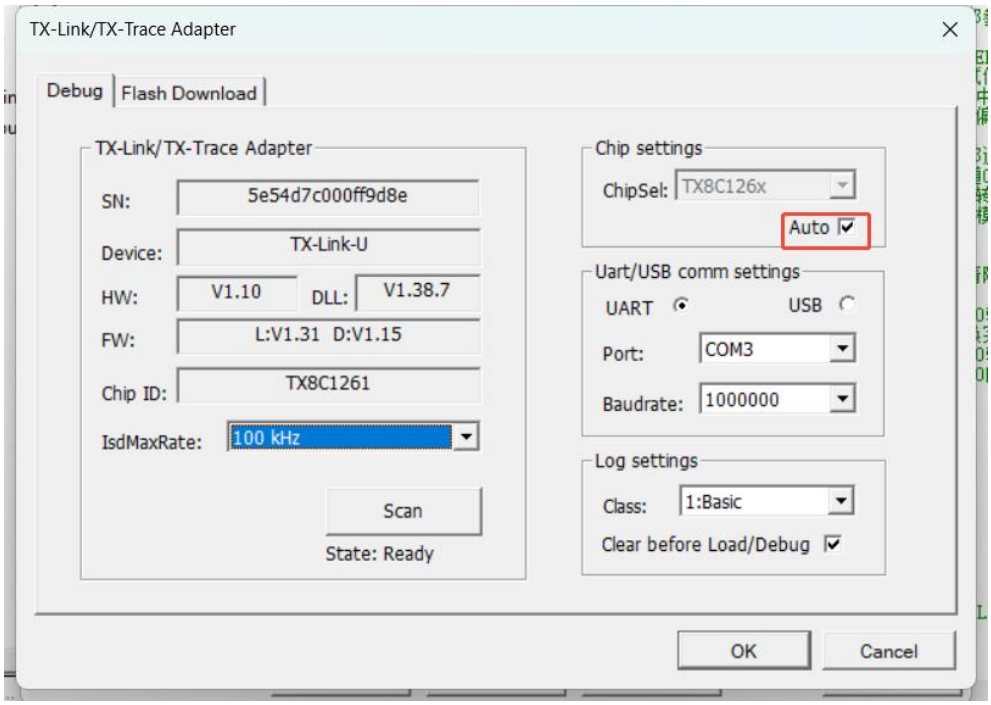


Download 失败的原因，可能出现情况如下：

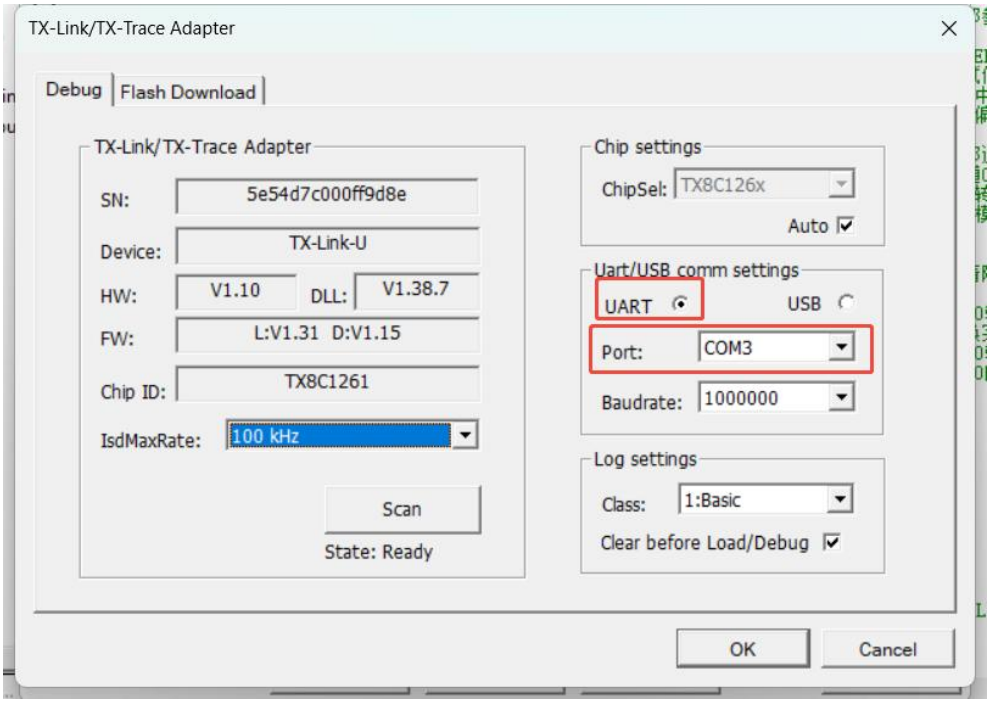
- 硬件线路连接异常-》使用 demo 板或单芯片的空板试下，或万用表测量通断。
- 配置“Debug -> TX-Link/TX-Trace Adapter -> IsdMaxRate”的速率太高-》降低速率



- 配置“Debug -> Chip settings -> ChipSel”型号与目标芯片型号不对应-》勾选 Auto



- 配置“Debug -> Uart comm settings -> Port”端口不是 TXLink 的端口-》拔插 USB 确认选择正确端口



1.4.2. 芯片不匹配

当选择下载的芯片型号与目标芯片的型号不一致时，Keil 的界面上弹出以下的提示框，此时用户需要留意配置界面中选择的芯片型号和实际方案板上的型号是否一致。



2. TX 系列通用 FAQ

2.1. 挂大电容在板烧录失败

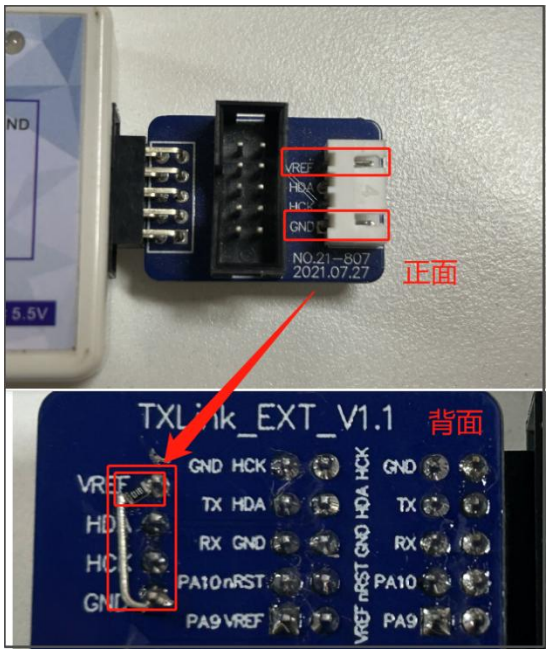
2.1.1. 问题现象

TX8C101x/TX8C126x/TX8M226x/TX8W701x 系列烧录挂有大电容（例如 220uF）板子出现烧录失败。

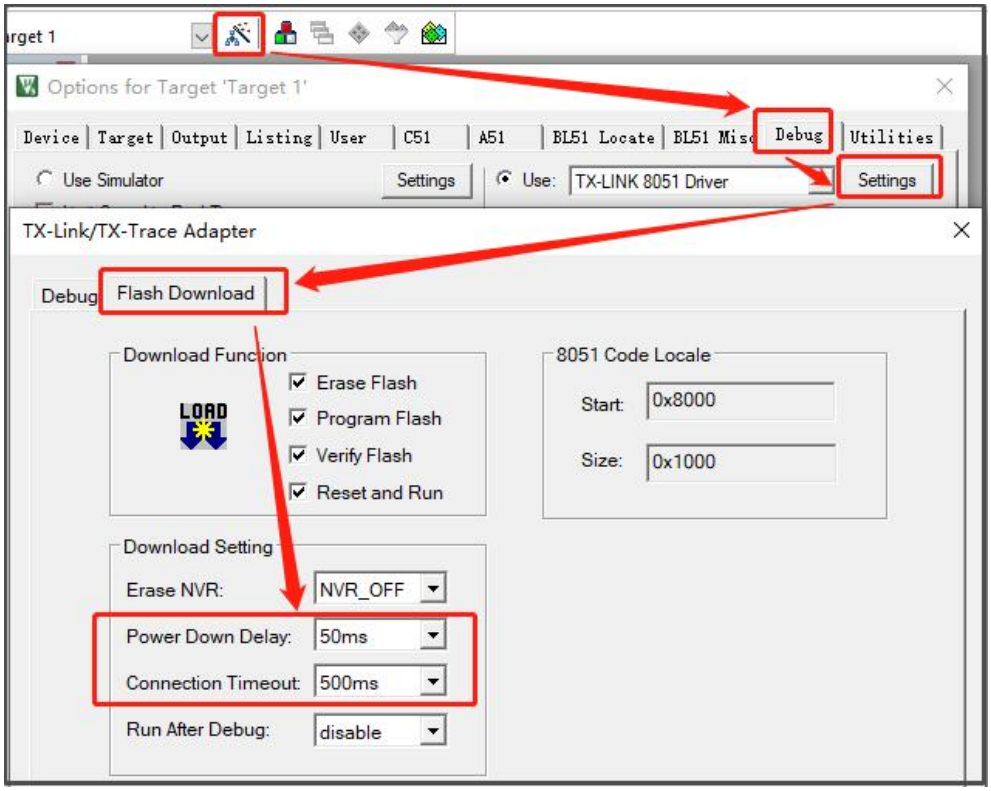
2.1.2. 解决方案

使用 TXLink 调试器下载失败：

1. 在 TXLink 转接板上的 VCC 与 GND 之间加一个 1~0.5K 的电阻(电阻越小，放电越快)，目的是加快电容放电的时间；

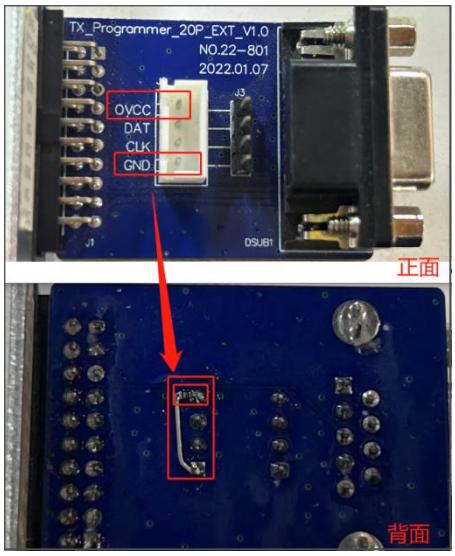


2. 需修改下图配置，加大“Power Down Delay”(掉电等待时间)、“Connection Timeout”（连接超时时间）的时间。

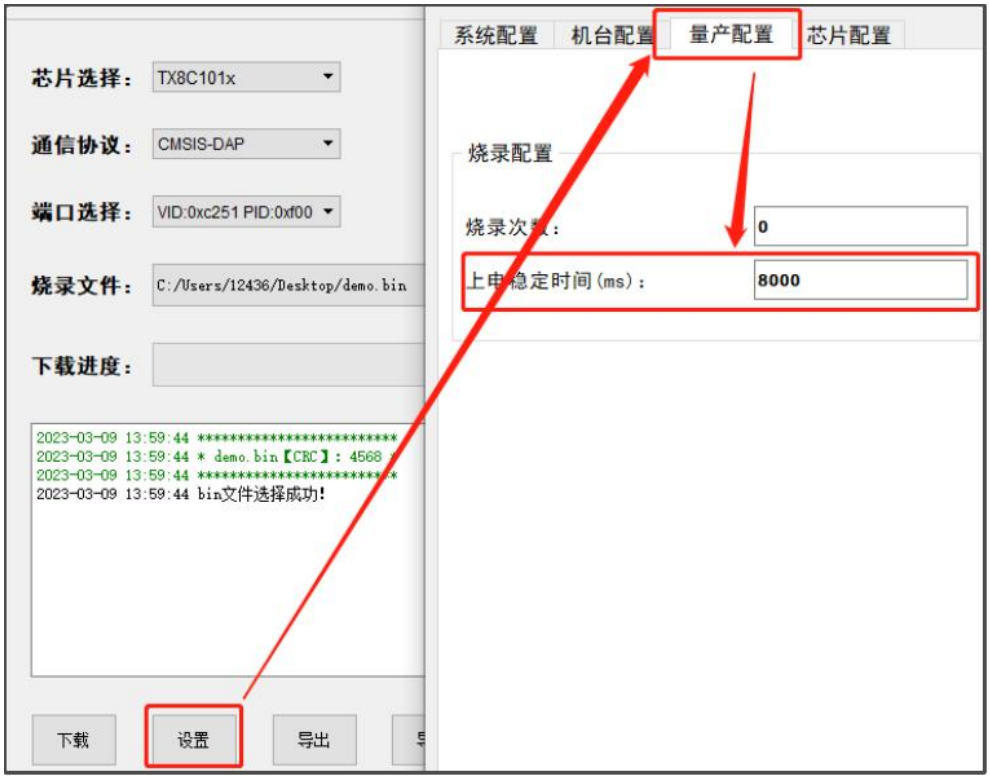


使用 TX-PROG 量产烧录器下载失败：

1. 在 TX-PROG 量产烧录器转接板上的 VCC 与 GND 之间加一个 1~0.5K 的电阻(电阻越小，放电越快)，目的是加快电容放电的时间；



2. 需修改下图配置，加大“上电稳定时间”，此配置在实际中是连接超时时间，此配置的值除以 8 为掉电等待时间（上位机软件需要 V2.1.36 版本以上）。

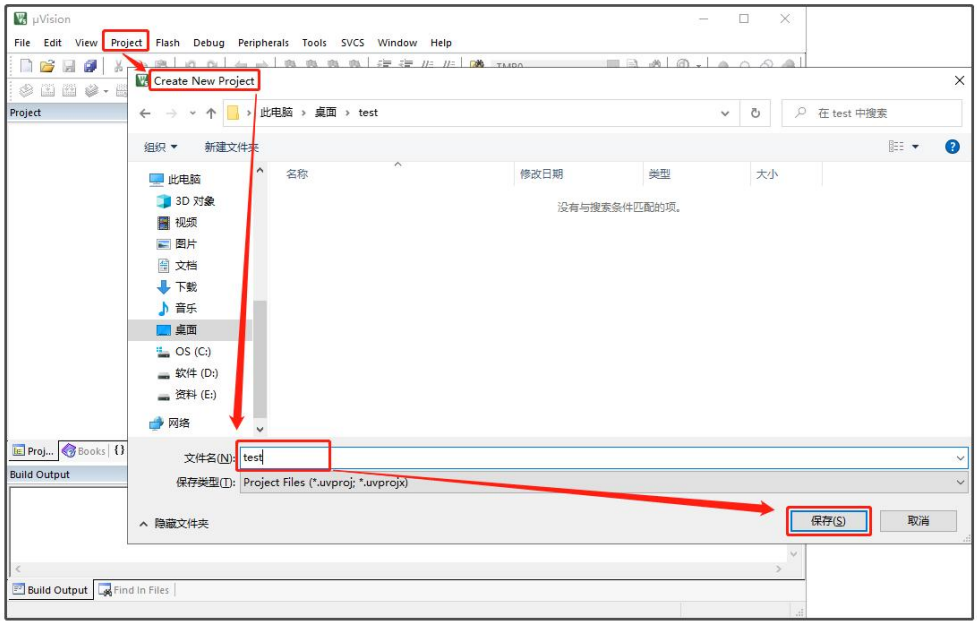

芯片选择：
TX8C101x
通信协议：
CMSIS-DAP
端口选择：
VID:0xc251 PID:0xf00
烧录文件：
C:/Users/12436/Desktop/demo.bin
下载进度：
2023-03-09 13:59:44 **********
2023-03-09 13:59:44 * demo.bin【CRC】: 4568
2023-03-09 13:59:44 **********
2023-03-09 13:59:44 bin文件选择成功!
下载
设置
导出
系统配置
机台配置
量产配置
芯片配置
烧录配置
烧录次数：
0
上电稳定时间(ms)：
8000

2.2. 新建 keil 工程

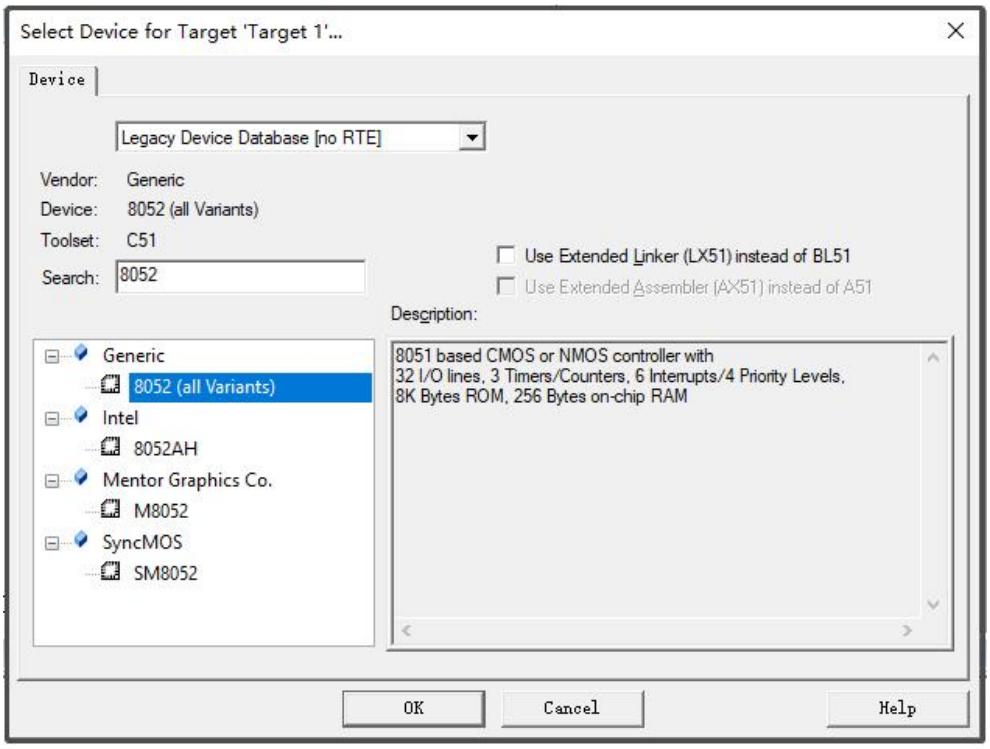
2.2.1. 解决方案

通用步骤（1~3）：

1. 打开 IDE 软件，在工具栏中选择“Project”->“New uVision Project”->“Create New Project”，在该界面中，选择工程要保存的路径，并设置工程的名字，最后点击“保存”。图示中，新建一个“test”的示例工程。



2. 在“Select Device for Target ‘Target 1’”的配置界面中，选择“8052”，点击“OK”。



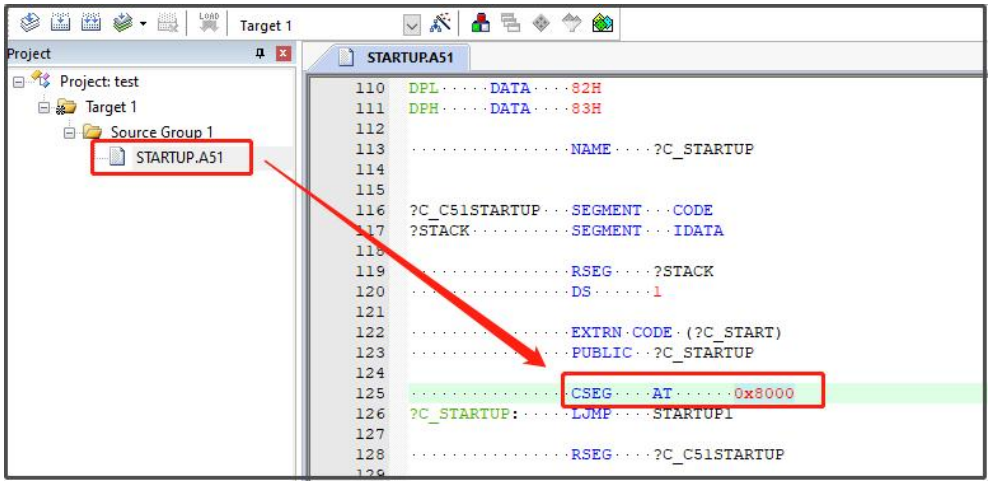
3. 弹出以下选择框，点击“是”。



步骤 4：STARTUP.A51 配置（因系列而异）

TX8C101x/TX8W701x：

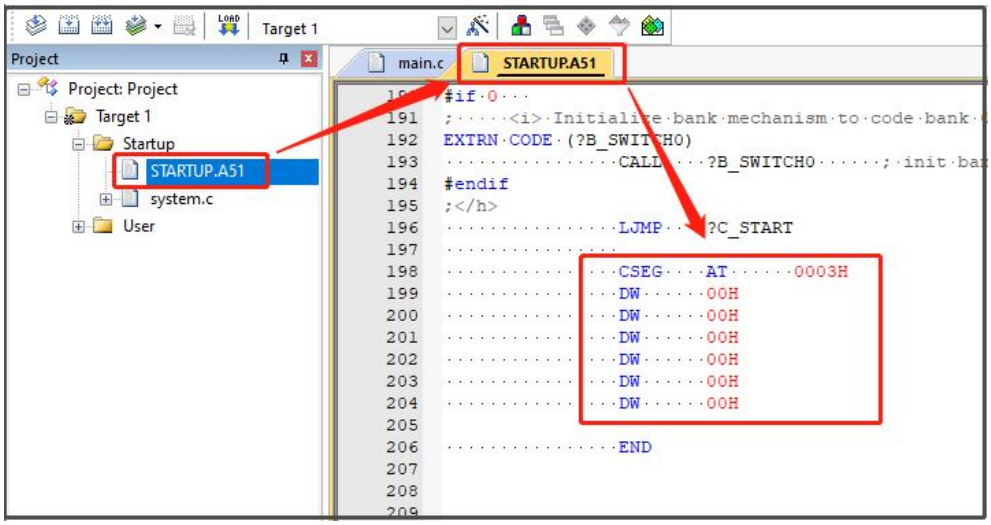
新建工程中，包含了“STARTUP.A51”文件。点击该文件，并找到“?C_STARTUP:”的段地址，将“CSEG AT 0x0”修改为“CSEG AT 0x8000”，因为 TX8C101x/TX8W701x 的 Code 区地址从 0x8000 开始。



TX8C126x/TX8M226x：

点击该文件，在文件末尾处，找到“LJMP ?C_START”和“END”的位置，在中间插入以下汇编程序。

```
CSEG    AT      0003H
DW      00H
DW      00H
DW      00H
DW      00H
DW      00H
DW      00H
```



步骤 5：添加工程文件（因系列而异）

TX8C101x/TX8W701x：

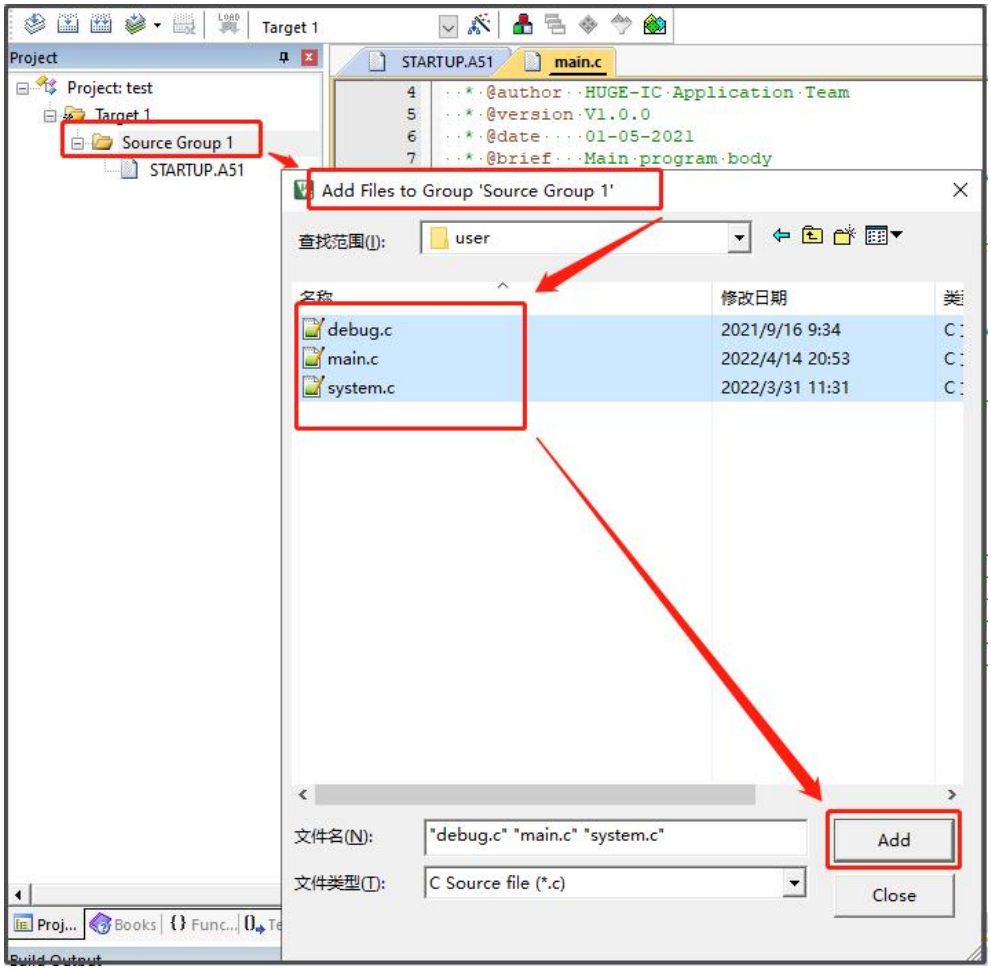
工程中添加 TX8C101x/TX8W701x 的相关文件，这些文件可以从对应系列的标准 SDK 中获取，其中文件“tx8c1010.h”是必要的且内容不可修改，“main.c”和“system.c”文件可以选择性整理合并。但是“main.c”中的函数 main（）、“system.c”中的函数 system_init()是必要的，可以整

合，但是不可删除。

TX8C126x/TX8M226x：

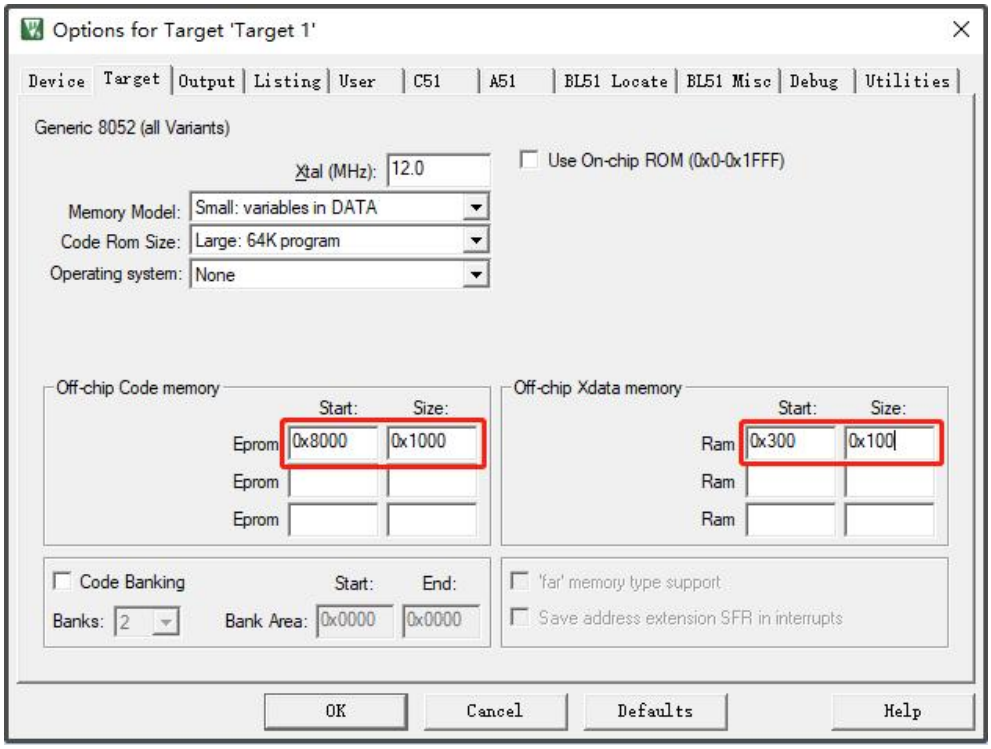
工程中添加对应系列的相关文件，这些文件可以从对应系列的标准 SDK 中获取，其中文件“tx8c126x.h”（TX8M226x 为“TX8M226x.h”）是必要的且内容不可修改，“main.c”和“system.c”文件可以选择性整理合并。但是“main.c”中的函数 main（）、“system.c”中的函数 system_init()是必要的，可以整合，但是不可删除。工程中模块的头文件建议一起添加，如果用户想自己依照 SPEC 配置寄存器，也可以不添加。

点击左边工作栏“Project”->“Source Group 1”右键->“ADD Exist Files To Group”。图示中，是将标准工程中的 3 个 C 文件添加到 test 工程中，最后点击“ADD”。

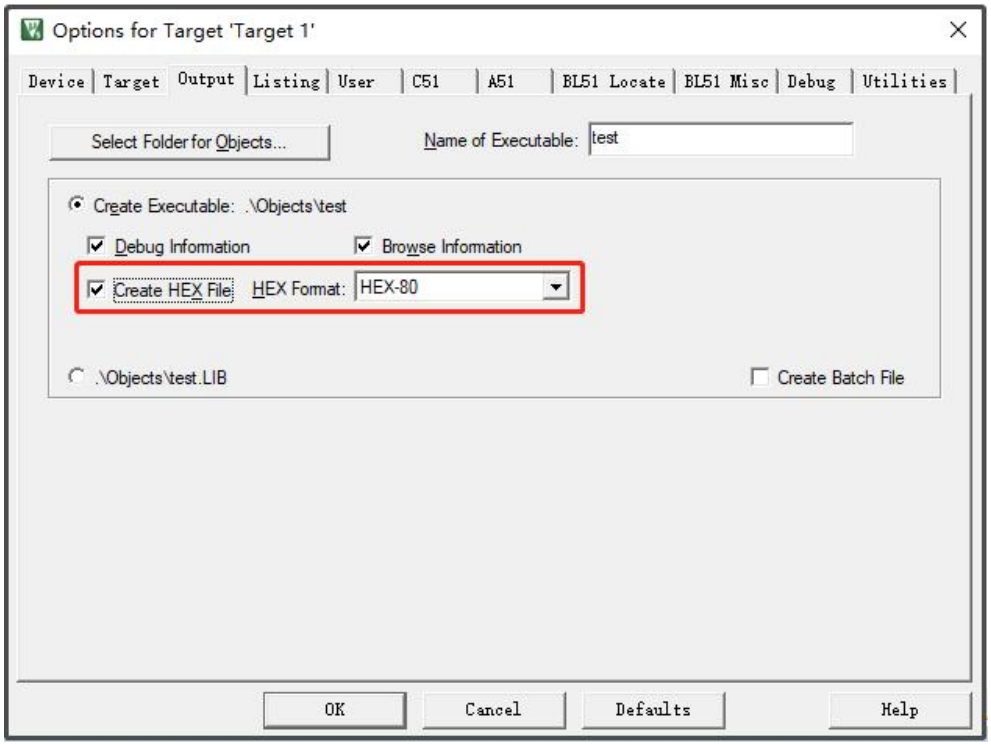


步骤 6：设置工程配置

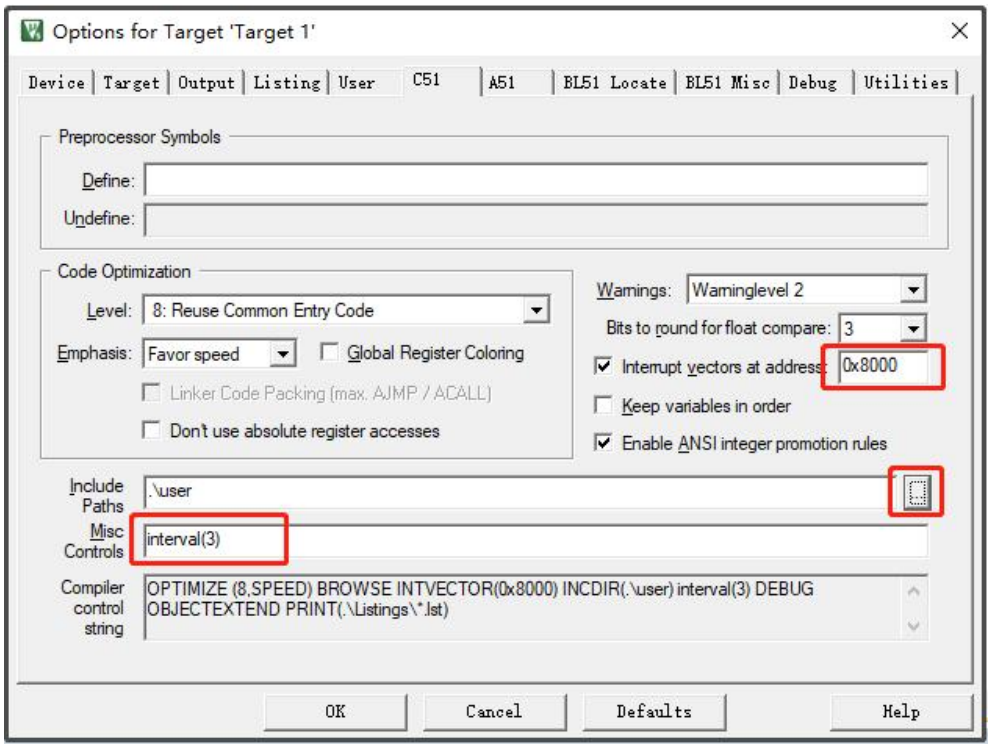
6.1、“Target”配置界面，设置 Code 和 Xdata 的起始地址和长度，如下图。



6.2、“Output”配置界面，勾选“Create HEX File”。

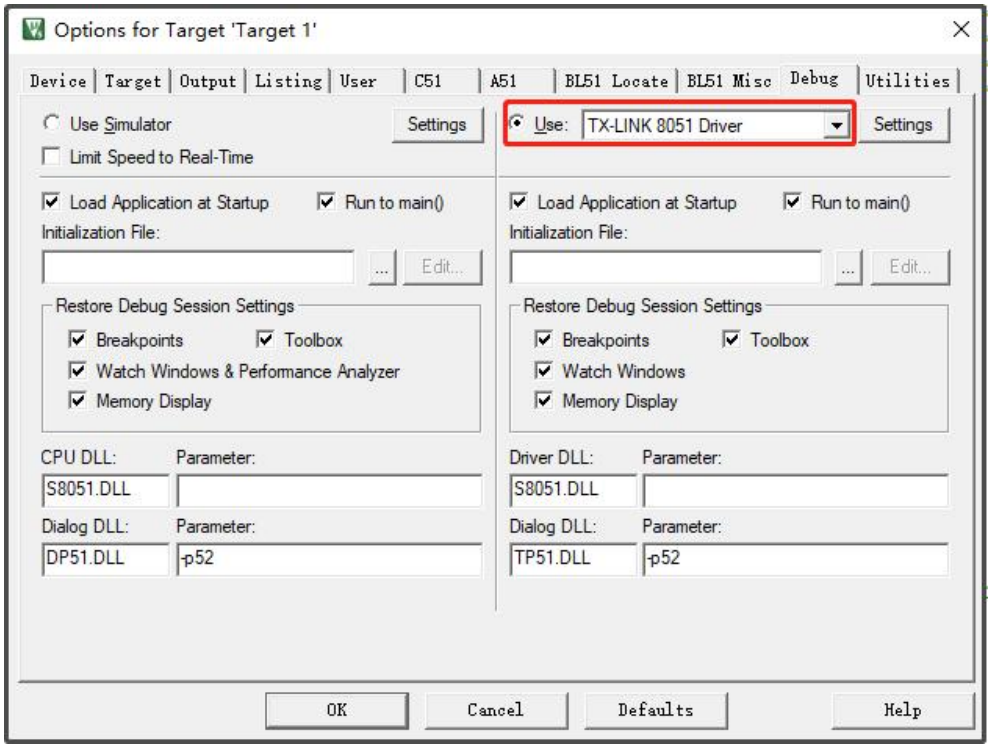


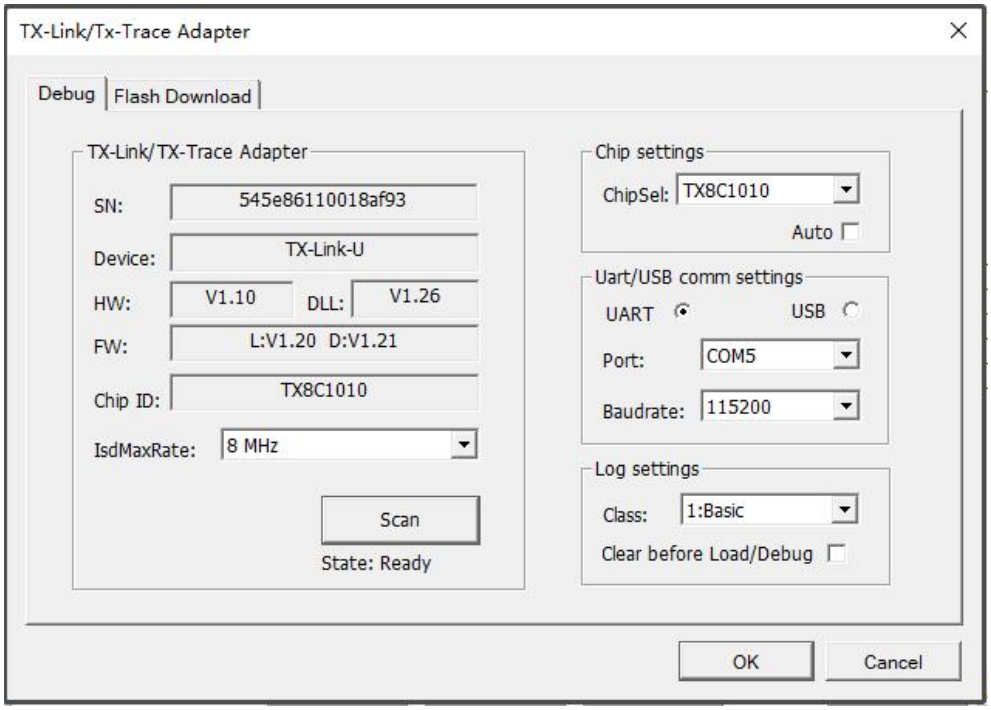
6.3、“C51”配置界面设置中断的入口地址、工程头文件的路径（这个根据工程中调用头文件的路径进行添加，跟工程的目录结构相关）、中断入口函数的位宽。

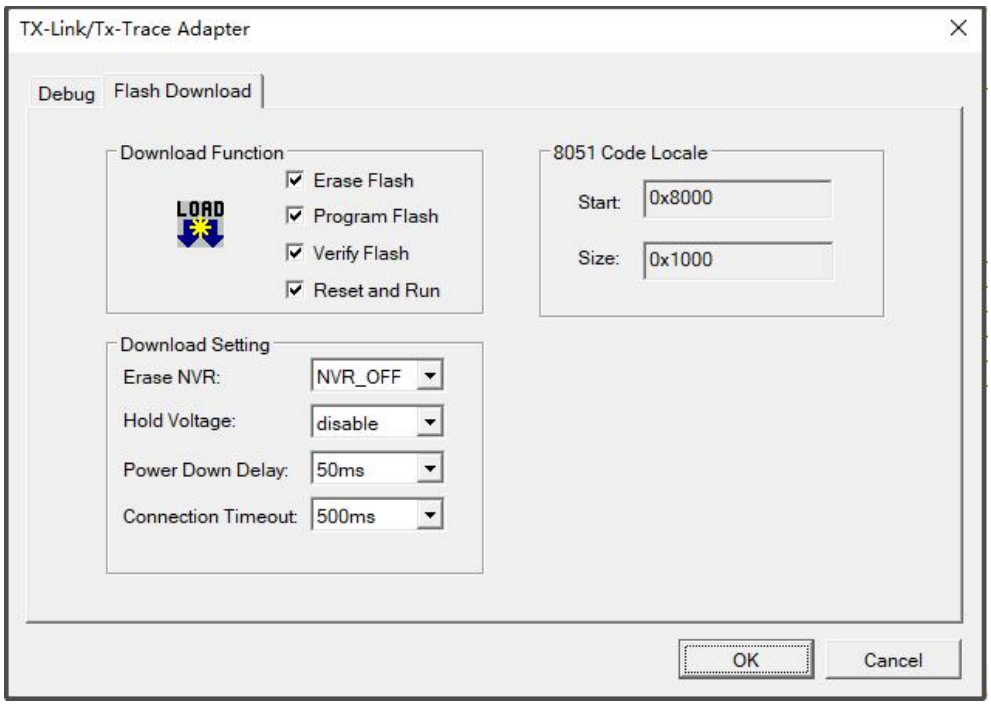


6.4、“Debug”配置界面，选择“TX-LINK 8051 Driver”，点击“Settings”，设置 TXLink 的配置信息。

TX8C101x/TX8W701x 配置界面如下：

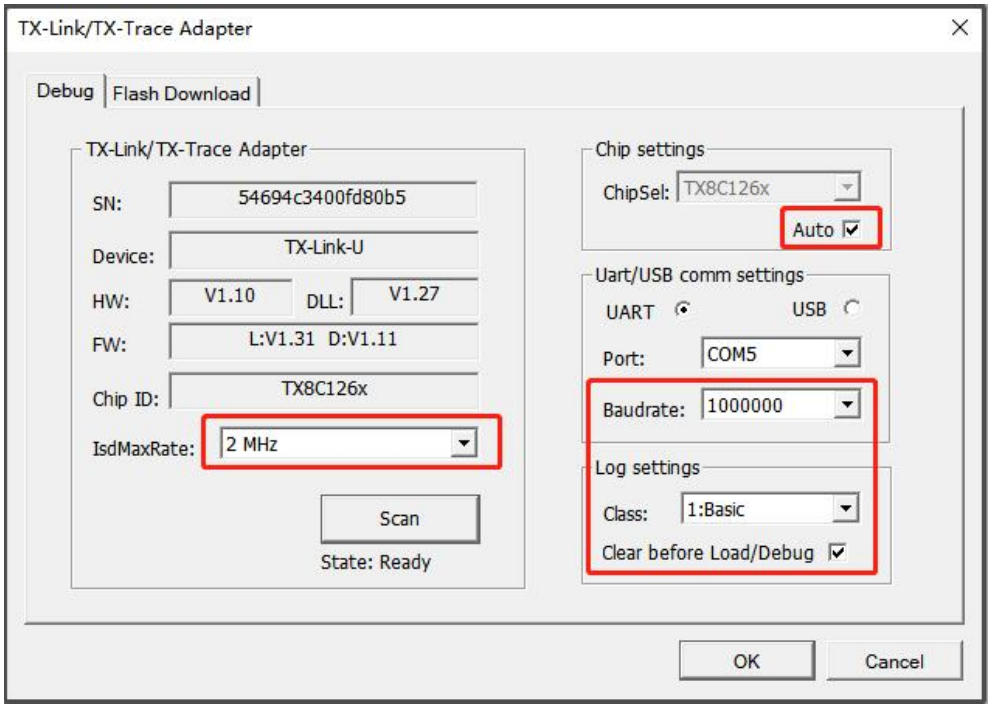




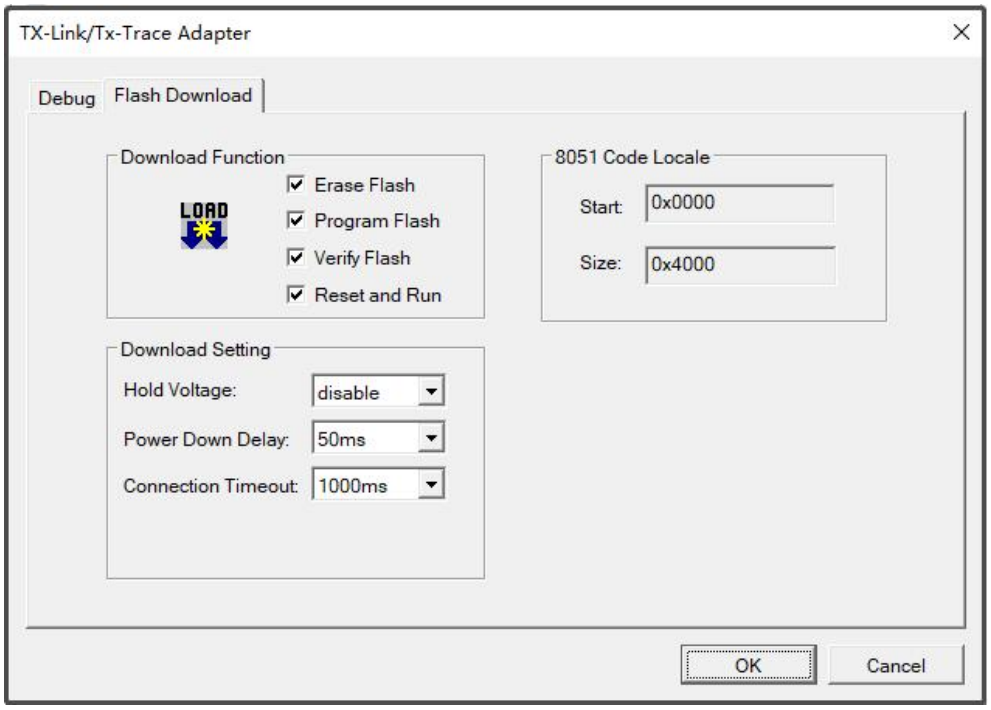


TX8C126x/TX8M226x：

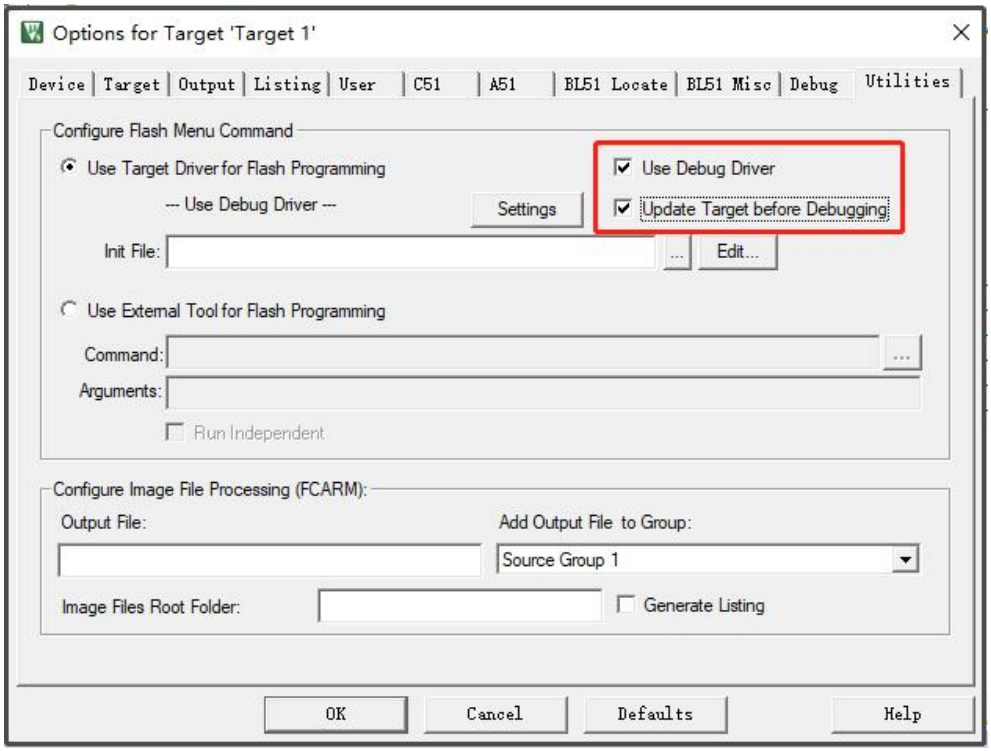
点击“Settings”后，弹出以下选择框，请按照以下配置进行选择，并点击“Scan”，检查“Chip ID”中的数据是不是跟实际选择的芯片对应上的。



点击“Flash Download”选择框，将对应的配置修改为以下：



6.5、“Utilites”配置界面，将以下两个配置勾选上。


Options for Target 'Target 1'
Device
Target
Output
Listing
User
C51
A51
BL51 Locate
BL51 Misc
Debug
Utilities
Configure Flash Menu Command
Use Target Driver for Flash Programming
--- Use Debug Driver ---
Settings
Use Debug Driver
Update Target before Debugging
Init File:
Edit...
Use External Tool for Flash Programming
Command:
Arguments:
Run Independent
Configure Image File Processing (FCARM):
Output File:
Add Output File to Group:
Source Group 1
Image Files Root Folder:
Generate Listing
OK
Cancel
Defaults
Help

7. 点击编译下载即可。

2.3. 烧录口 HCK、HDA 使用为 GPIO 的限制条件

2.3.1. 问题现象

TX8C101x/TX8C126x/TX8M226x/TX8W701x 系列的烧录口 HCK/HDA 引脚使用为 GPIO 有哪些限制条件？

2.3.2. 解决方案

TX8C101x/TX8C126x/TX8M226x/TX8W701x 系列的烧录口（HCK、HDA）可以作为 GPIO，但是为了烧录稳定性，使用时有限制条件。

各系列烧录口引脚如下：

系列	HCK 引脚	HDA 引脚
TX8C101x / TX8W701x	P10	P07
TX8C126x / TX8M226x	P31	P17

HDA 引脚限制条件（各系列通用）：

外围的硬件电路，在引脚不工作默认状态下，必须满足：

- 不能有大负载；
- 不建议外挂电容（超过 102 会导致烧录失败）。

HCK 引脚限制条件（各系列通用）：

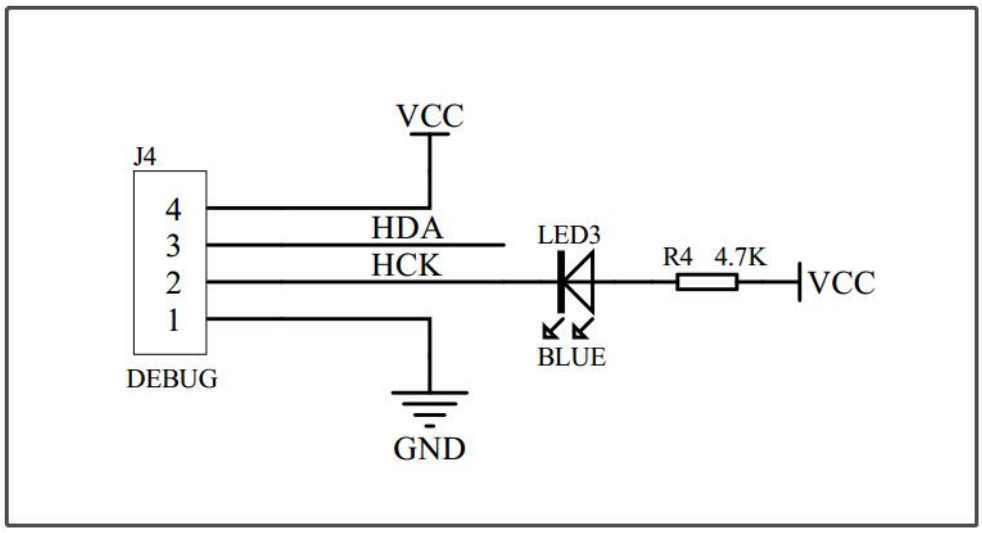
引脚 HCK 的外围硬件，在引脚不工作的默认状态下，必须满足：

- 不能有大负载；
- 不建议外挂电容（超过 102 会导致烧录失败）。

TX8C101x/TX8W701x 系列特有限制：

对于 TX8C1010 系列，HCK（P10）使用还有以下限制条件：

- 适用于驱动 LED（参考电路如下图所示）；

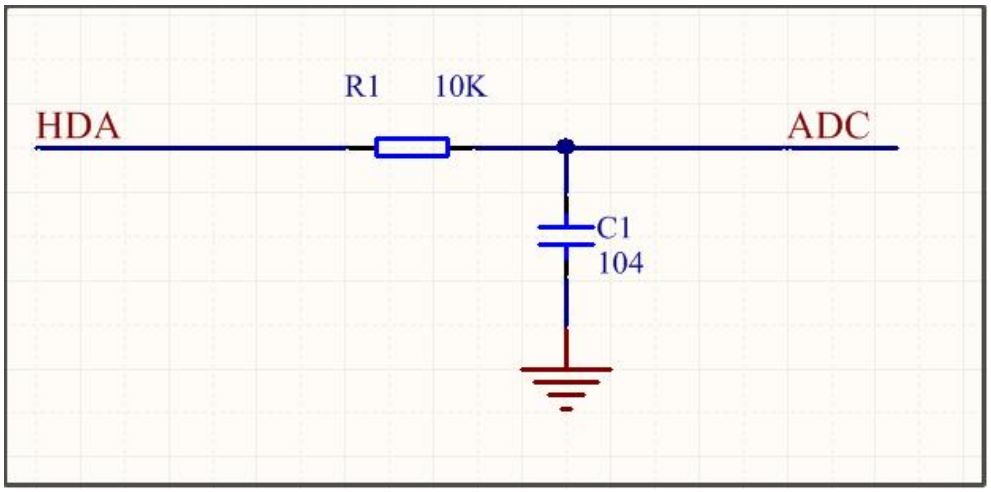


具体的使用实例，请参考 SDK 目录下的 Examples\DEBUG\Debug_To_GPIO 或者 Examples\DEBUG\Debug_To_GPIO_BIWP 的工程实例；如果方案中 ADC 使用了外部参考电压，请参考 Examples\DEBUG\Debug_To_GPIO_ADC_XREF。

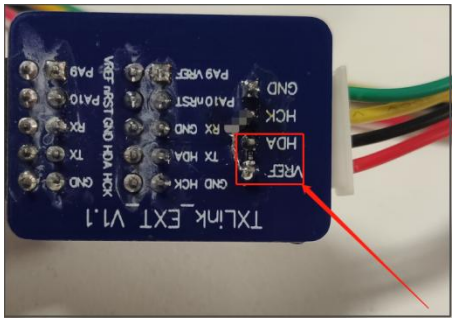
TX8C126x/TX8M226x 系列特有限制：

如果应用方案使用到了 ADC，必须满足：

- 在 HDA 与 ADC 滤波电容之间串联 10K 电阻，如图所示：



- 烧录器转接板上 VREF 与 HDA 之间串联一个 1K 电阻，如图所示：



具体的使用实例，请参考 SDK 目录下的 Examples\DEBUG\Debug_To_GPIO 或者 Examples\DEBUG\Debug_To_GPIO_BIWP 的工程实例。

注意：如果电路不满足以上描述的条件，会出现下载异常的情况，请用户留意。

2.4. 烧录口使用为 GPIO 后下载异常问题

2.4.1. 问题现象

TX8C101x/TX8C126x/TX8M226x/TX8W701x 系列的烧录口（HCK、HDA）使用为 GPIO 后，发现下载异常的问题。

2.4.2. 解决方案

TX8C101x/TX8C126x/TX8M226x/TX8W701x 系列的烧录口（HCK、HDA）使用为 GPIO 后，出现下载异常，有以下几种可能：

- 请优先确认烧录的模式，是“掉电烧录+GPIO”还是“带电烧录+GPIO”，再确认是否使用对应的示例工程，最后确认烧录配置是否选择正确，具体请参考“TX8C101x/TX8C126x/TX8M226x/TX8W701x 系列的烧录方式”；

- 请检查 HDA 的使用限制条件是否满足；

- 请检查 HCK 的使用限制条件是否满足；

- HCK、HDA 烧录口不建议使用电容（102 以上电容会导致烧录失败）；

- 请检查烧录器/调试器与 HCK、HDA 的连接是否有下拉电阻，默认使用 10K；

- 使用调试器下载出现下载异常，请确认芯片是否使用烧录器打开加密功能；

- 这种使用场景，请注意烧录方式。如果烧录模式选择为“掉电烧录+GPIO”，那么板子/芯片只能由烧录器/调试器供电，板子/芯片不能独立供电，否则会出现下载异常的情况。

2.5. 烧录方式

2.5.1. 问题现象

TX8C101x/TX8C126x/TX8M226x/TX8W701x 系列的烧录方式？

解决方案

TX8C101x/TX8C126x/TX8M226x/TX8W701x 的烧录方式分为三种，分别是正常烧录、【掉电烧录+GPIO】和【带电烧录+GPIO】。

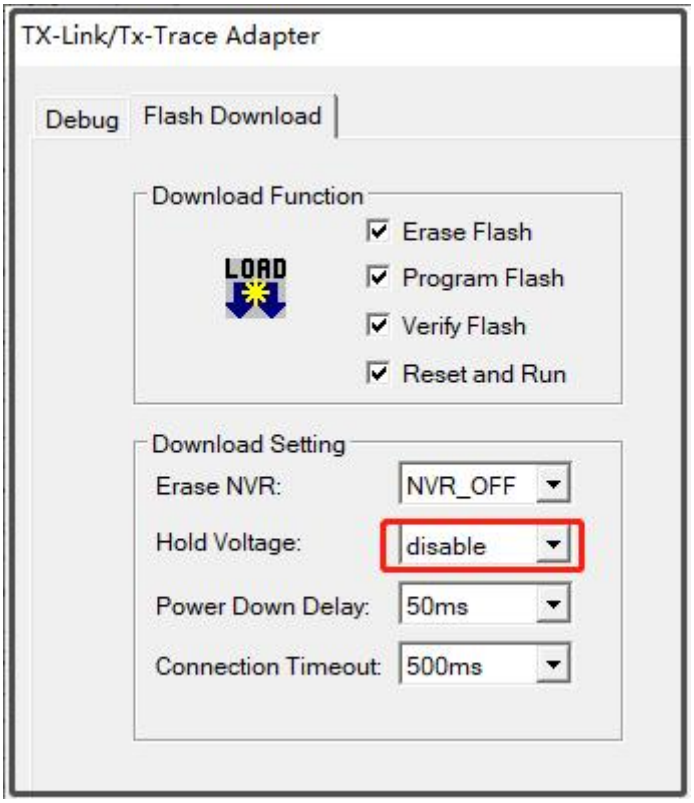
各系列烧录口使用为 GPIO 时，可用的 GPIO 数量不同，具体如下：

序号	烧录方式	HDA/HCK 功能	VCC 掉电	GPIO 数量（TX8C101x/TX8W701x）	GPIO 数量（TX8C126x/TX8M226x）	限制
1	正常烧录	烧录功能	不需要	12	20	无
2	掉电烧录+GPIO	烧录功能+GPIO	需要	14	22	有
3	带电烧录+GPIO	烧录功能+GPIO	不需要	14	22	有

表 1 - 各系列烧录方式与 GPIO 数量列表

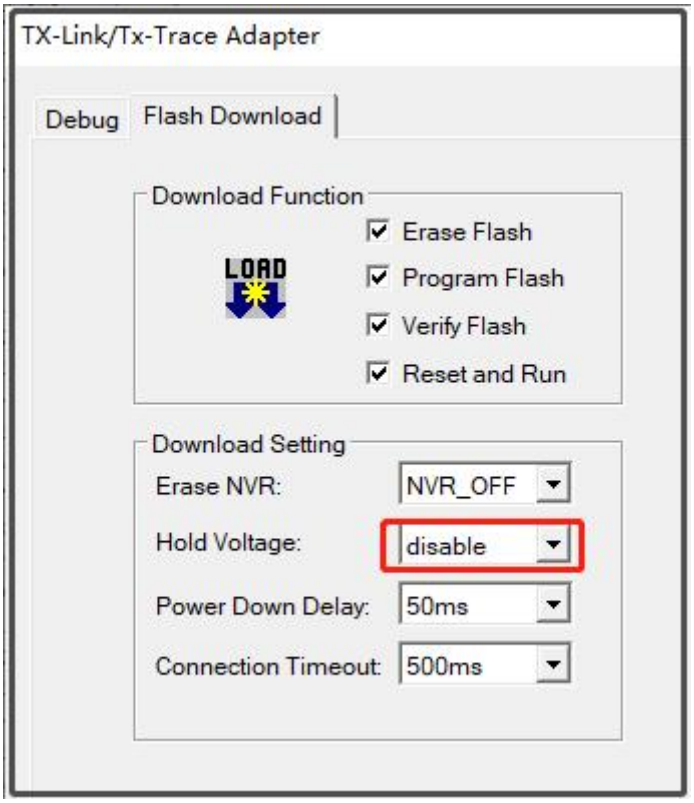
正常烧录方式：

- 烧录口只用于烧录程序或者调试程序，不用于其他功能；
- 烧录时支持 VCC 不掉电烧录；
- Keil IDE 中 “Options for Target”->“Debug”->“Settings”->“Flash Download”->“Download Setting”->“Hold Voltage”选项设置为“disable”。



掉电烧录+GPIO 方式：

- 烧录口即用于烧录功能，又用于 GPIO 功能，但是使用上有限制：
- 烧录时 VCC 需要掉电；
- 需要使用特殊的示例功能，该方式的烧录需要依赖软件配合，各系列对应的示例工程路径如下：
- TX8C101x 系列：TX8C101x_SDK_V1.1.0_20210903\DEBUG\Debug_To_GPIO
- TX8C126x 系列：TX8C126x_SDK\DEBUG\Debug_To_GPIO
- TX8M226x 系列：TX8M226x_SDK\DEBUG\Debug_To_GPIO
- TX8W701x 系列：TX8W701x_SDK_V1.1.0_20210903\DEBUG\Debug_To_GPIO
- Keil IDE 中 “Options for Target”->“Debug”->“Settings”->“Flash Download”->“Download Setting”->“Hold Voltage”选项设置为“disable”。



带电烧录+GPIO 方式：

- 烧录口即用于烧录功能，又用于 GPIO 功能，但是使用上有限制：
- 烧录时不需要 VCC 掉电；
- 需要使用特殊的示例功能，该方式的烧录需要依赖软件配合，各系列对应的示例工程路径如下：

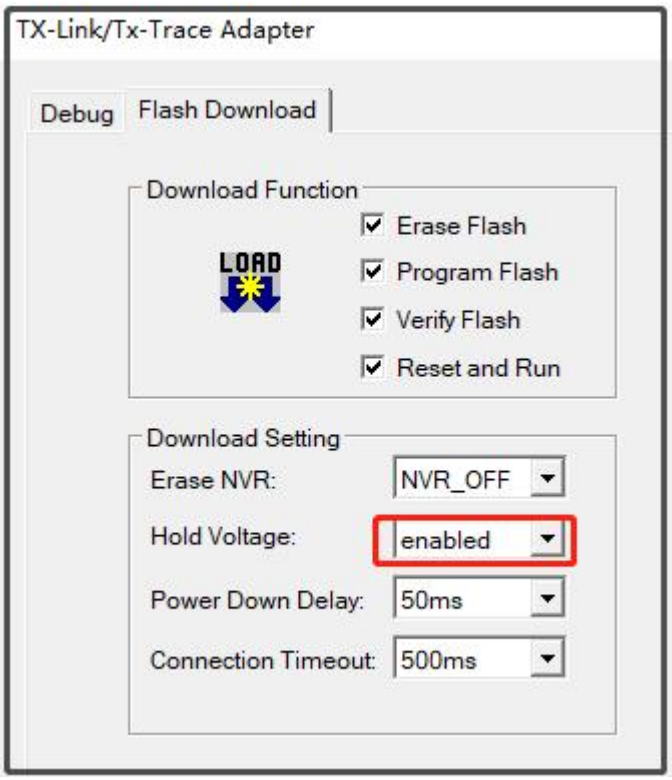
• TX8C101x 系列：TX8C101x_SDK_V1.1.0_20210903\DEBUG\Debug_To_GPIO_BIWP

• TX8C126x 系列：TX8C126x_SDK\DEBUG\Debug_To_GPIO_BIWP

• TX8M226x 系列：TX8M226x_SDK\DEBUG\Debug_To_GPIO_BIWP

• TX8W701x 系列：TX8W701x_SDK_V1.1.0_20210903\DEBUG\Debug_To_GPIO_BIWP

• Keil IDE 中 “Options for Target”->“Debug”->“Settings”->“Flash Download”->“Download Setting”->“Hold Voltage”选项设置为“enable”。


TX-Link/Tx-Trace Adapter
Debug
Flash Download
Download Function
LOAD
Erase Flash
Program Flash
Verify Flash
Reset and Run
Download Setting
Erase NVR:
NVR_OFF
Hold Voltage:
enabled
Power Down Delay:
50ms
Connection Timeout:
500ms

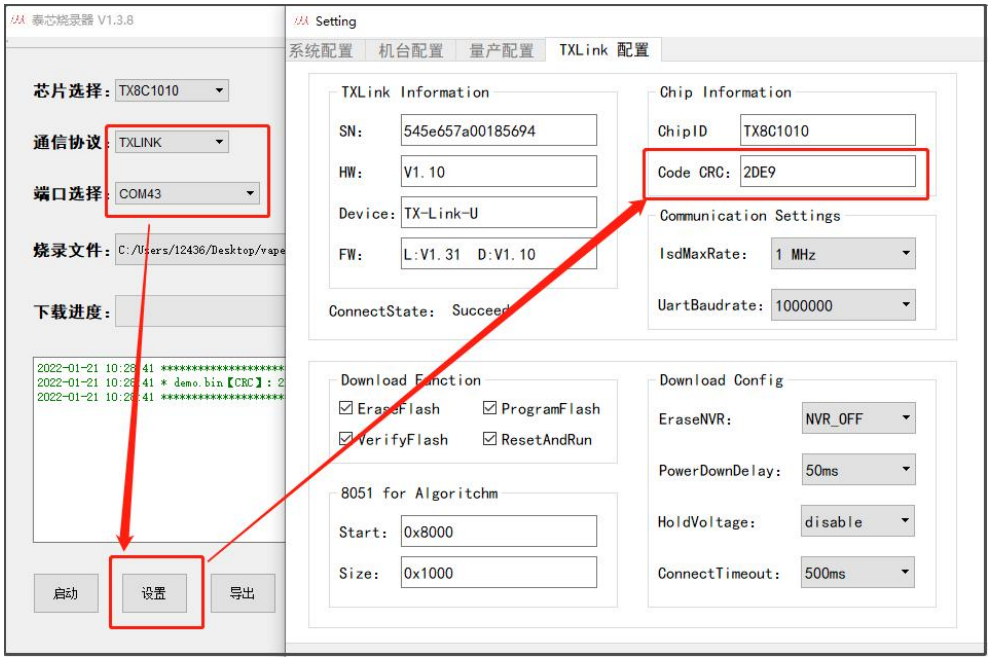
2.6. 读取 CodeCRC 的功能说明

2.6.1. 解决方案

步骤一：读取方式

方式 1：使用 TXLink 通过 PC 工具 TXProgrammer

TXProgrammer 在 TXLink 模式下的设置界面中，对目标芯片 Code CRC 的读取（需要使用 TX-PROG 烧录器用量产模式烧录过的芯片才能读取到 CRC 数值），如图所示。



方式 2：烧录器 TXPROG

- TX-PROG 量产模式下烧录的芯片用“1000”(读 CRC 模式)
- TX-PROG 调试模式下烧录的芯片用“0101”



问题解答

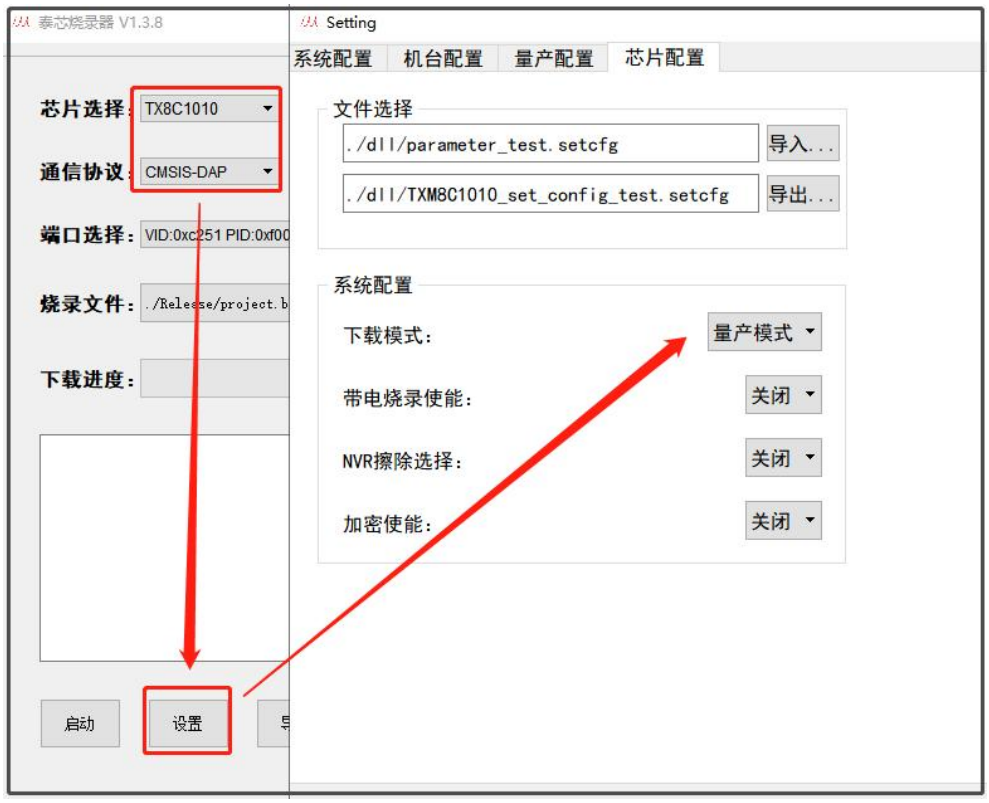
问题： 用户使用 IDE 下载后，发现读出来的 bin/hex 文件 Code CRC 与下载不一致？

回答： 请按照以下步骤检查：

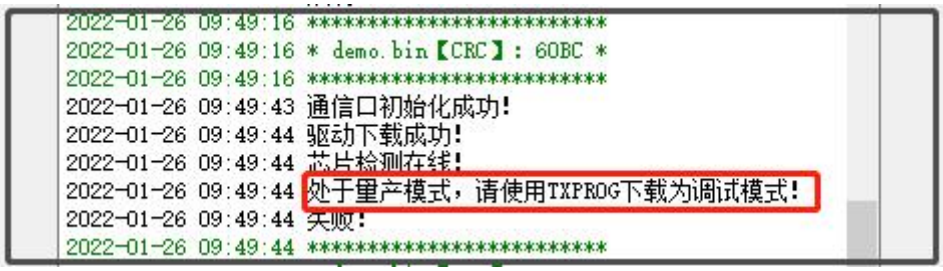
1. 优先检查 TXProgrammer 和 TXLink_keil_C51_driver_install，请按照步骤 2 的“更新说明”操作。如果 TXLink_keil_C51_driver_install 版本低于对应系列要求版本（TX8C101x/TX8W701x 为 V1.2.3，TX8C126x/TX8M226x 为 V1.2.6）时，下载时不会进行烧录模式检测的，所以大概率出现上述情况；

2. Bin/hex 的 Code CRC 的更新功能，只有使用 TXPROG 烧录器的“量产模式”进行烧录后，才会同步更新 Code CRC，其他下载方式均不更新，步骤 3 的使用说明有显著标记。

问题： 用户想要下载的 bin/hex 文件要有 Code CRC 功能，则需要在 PC 工具 TXProgrammer 对 TXPROG 进行烧录时，将烧录模式设置为“量产模式”。注意：只有通过 TXPROG 的“量产模式”烧录的才会更新 Code CRC，其他情况均不更新。



回答： 使用“量产模式”烧录后，如果用户还想继续开发，则需要将“量产模式”烧录为“调试模式”才能使用 IDE 进行下载和 TXProgrammer 的 TXLink 方式才能支持下载，不然会出现以下提醒。





出现以上两种提示，则需要使用 TXPROG 重新烧录为“调试模式”。

2.7. Sleep 模式下使用外部晶振实现循环计时

2.7.1. 问题现象

TX8C101x/TX8W701x 系列使用外部晶振实现循环计时的同时进出 Sleep 模式。在 Sleep 模式下，Timer2 模块时钟需要选择外部晶振才能正常计时，清除中断标志位时系统时钟与 Timer2 模块时钟同步才能清除中断标志位，这其中就涉及到系统时钟在外部晶振与内部高速 32M 之间来回切换，会影响 PWM、ADC 等模块的正常工作。

2.7.2. 解决方案

芯片 TX8C101x/TX8W701x 不仅实现外部晶振实现循环计时的同时进出 Sleep 模式，并且不影响 PWM、ADC 等模块的正常工作，现将芯片分为三种模式：正常模式、低性能模式、SLEEP 模式。

正常模式：

系统时钟保持内部高速 32M 不变，Timer2 模块时钟选择系统时钟，计数源选择外部晶振，此时系统时钟和 Timer2 模块时钟都处于内部高速 32M 下，可直接进行中断标志位的清除；此模式下 PWM、ADC 等模块都可正常工作；

低性能模式：

Timer2 模块时钟选择外部晶振，系统时钟选择内部高速 32M，清除标志位需要将系统时钟切换到外部晶振、Timer2 模块时钟选择系统时钟（此处的系统时钟指的是 Timer2 模块时钟，切换不会影响计时），保证系统时钟与 TIMER2 模块时钟保持同步才能进行清除标志位；此模式下系统时钟会有切换，适合做一些简单逻辑处理，也可使用 ADC，使用 ADC 时需要关闭总中断，目的防止 Timer2 清中断标志位时切换系统时钟影响 ADC，此模式下的 ADC 精度没有正常模式时那么高，在 2%以内；

SLEEP 模式：

此模式为芯片标准的 Sleep 模式。

具体的使用实例，请参考 SDK 目录下的 Examples\SLEEP\SLEEP_CYCLE_TIMER 的工程实例。当需要芯片进入低功耗模式时，调用 sleep_config 函数后进入低性能模式，低性能模式和 SLEEP 模式会来回切换；当需要芯片正常工作时，从低性能模式中切换到正常模式，需调用 sleep_wake_up_config 函数。此示例需要外接晶振才能正常工作。

注意： 调用一次 sleep_wake_up_config 函数循环计时延迟 4 个晶振时钟周期；调用一次 sleep_config 函数循环计时偏快 2 个晶振时钟周期；

总结： Timer2 模块时钟需与系统时钟同步的情况下才能清除中断标志位。

2.8. 代码空间扩展

2.8.1. 问题现象

TX8C101x/TX8W701x 系列在开发过程中，代码量超出 4k Byte，想要扩展代码空间到(4k+128)Byte 或者(4k+256)Byte。

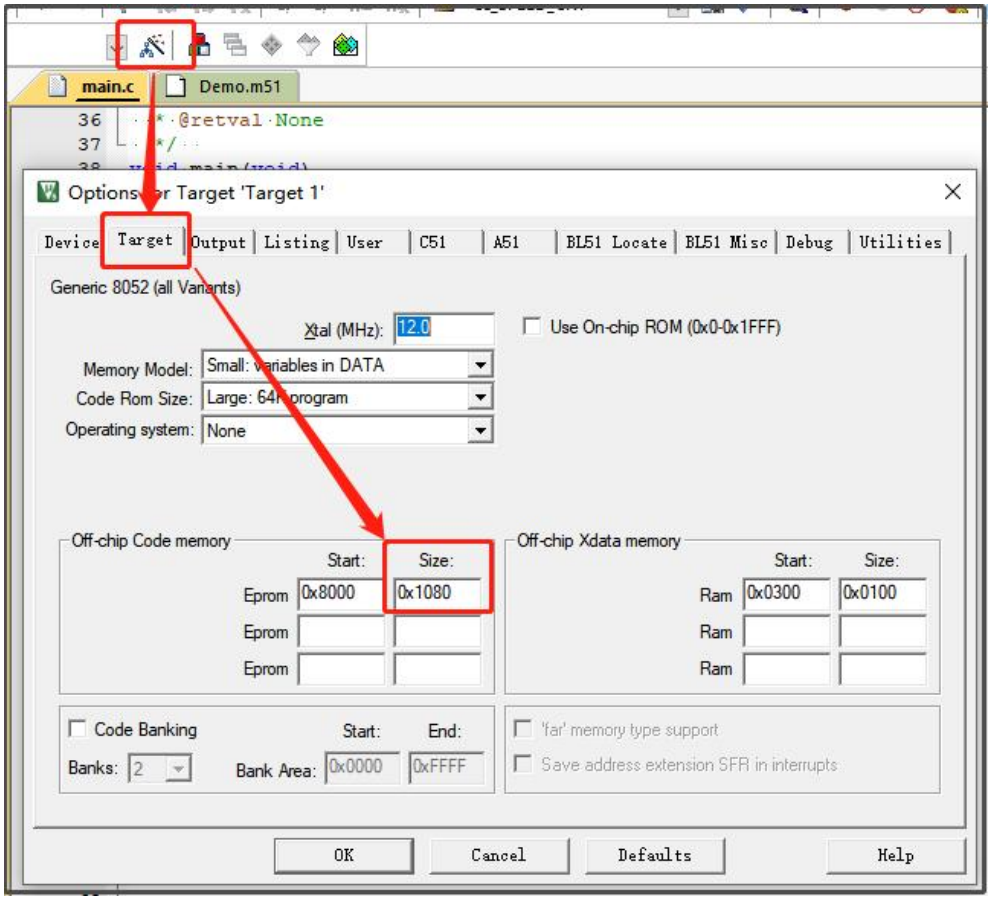
2.8.2. 解决方案

TX8C101x/TX8W701x 系列有 4K Byte 的 Flash 用于存储用户代码，256Byte 类 EEPROM（分为两个扇区：NVR0、NVR1，各 128Byte），当用户的代码量超出 4k Byte 时，可将 256Byte 类 EEPROM 存放代码，但有一定的限制：

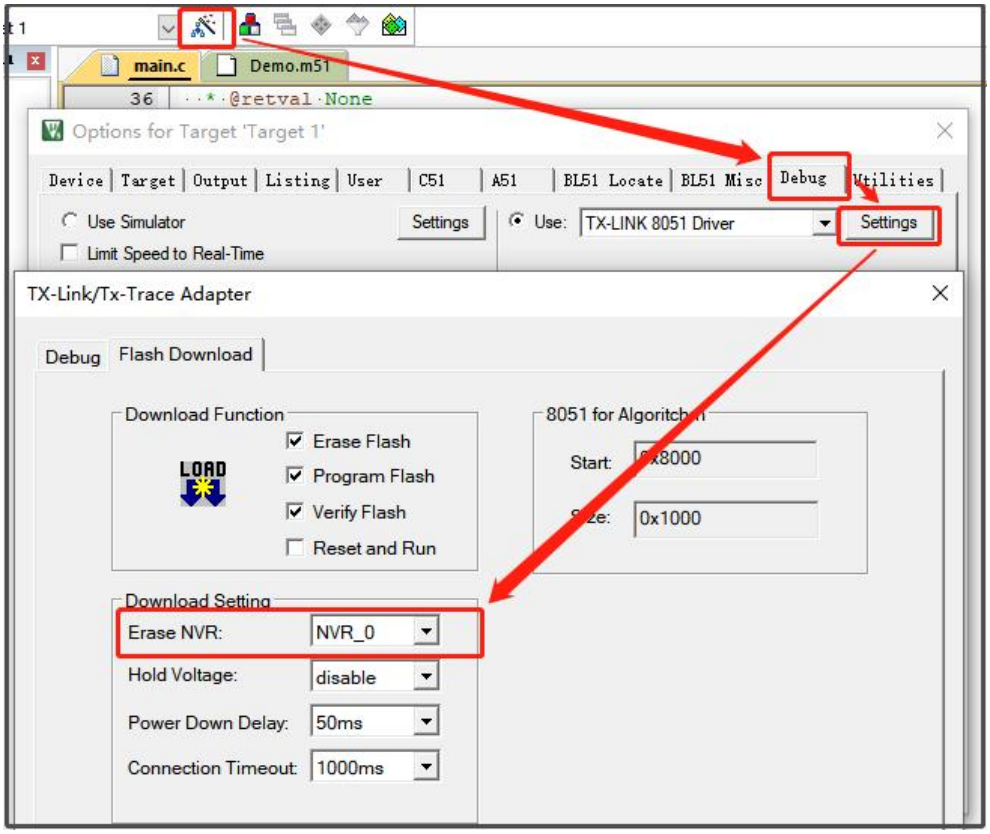
- 代码量扩展到(4k+128)Byte 时，类 EEPROM 只能使用 128Byte；
- 代码量扩展到(4k+256)Byte 时，不能使用类 EEPROM。

使用 128Byte 做类 EEPROM 的情况下，代码空间扩展到 4k+128Byte：

1. Keil IDE 中“Options for Target”->“Target”->“Off-chip Code memory”->“size”编辑框修改为“0x1080”，如下图所示：



2. Keil IDE 中 “Options for Target”->“Debug”->“Settings”->“Flash Download”->“Download Setting”->“Erase NVR”选项设置为“NVR_0”，如下图所示：



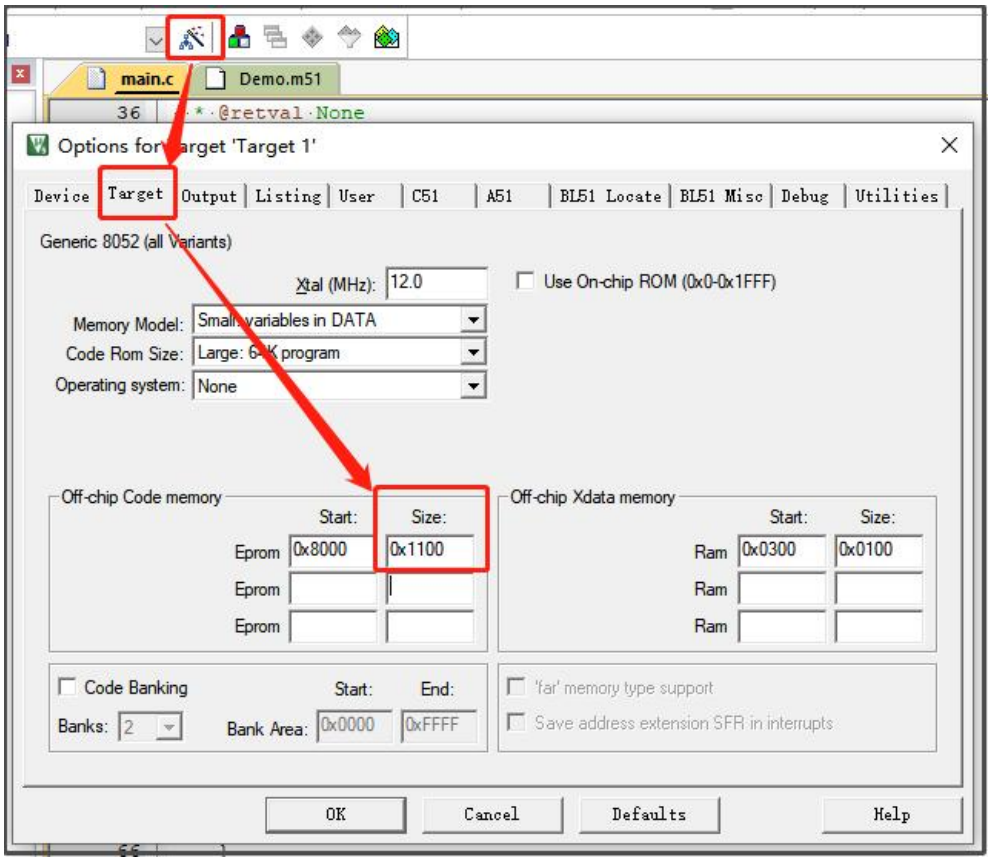
3. Keil IDE 编译下载程序即可。

4. 使用量产烧录器烧录时，PC 端需要在“设置”->“芯片设置”->“系统设置”->“NVR 擦除选择”选项中选择“NVR0”，如下图所示：

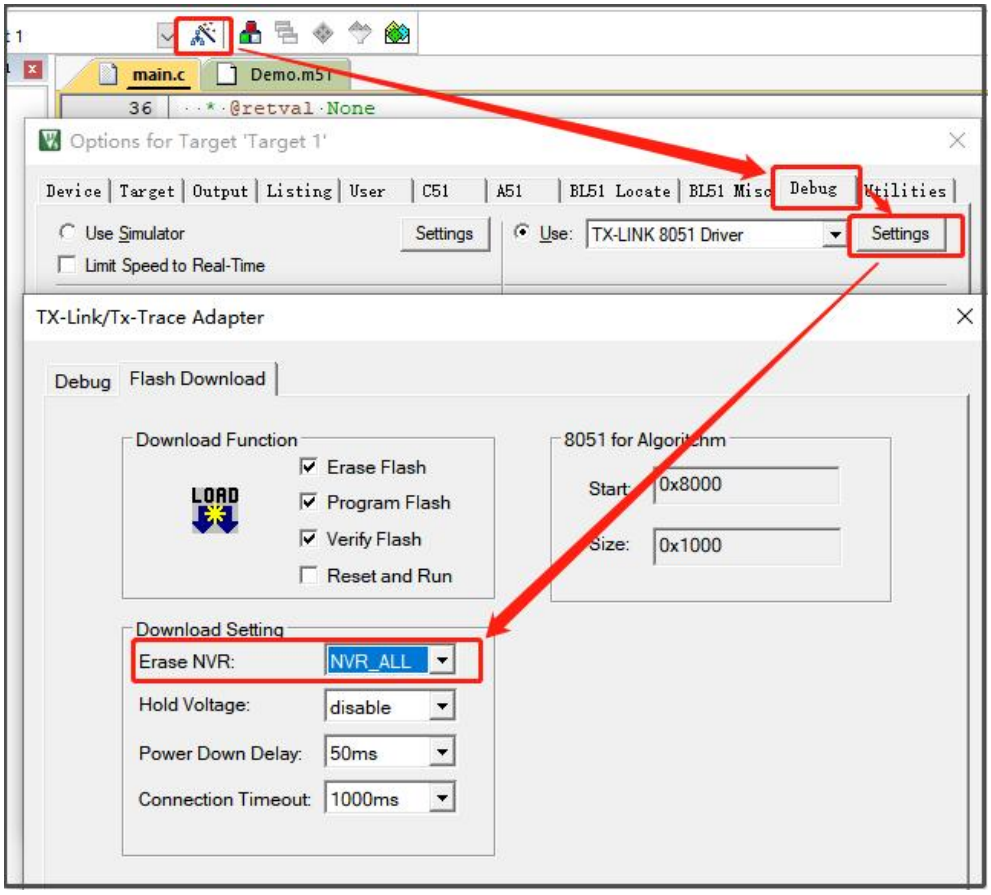


不使用类 EEPROM 的情况下，代码空间扩展到 4k+256Byte：

1. Keil IDE 中“Options for Target”->“Target”->“Off-chip Code memory”->“size”编辑框修改为“0x1100”，如下图所示;



2. Keil IDE 中“Options for Target”->“Debug”->“Settings”->“Flash Download”->“Download Setting”->“Erase NVR”选项设置为“NVR_ALL”，如下图所示;



3. Keil IDE 编译下载程序即可。

4. 使用量产烧录器烧录时，PC 端需要在“设置”->“芯片设置”->“系统设置”->“NVR 擦除选择”选项中选择“全部”，如下图所示：


启动
设置
导出
导入
帮助
芯片配置
系统配置
导入...
test.setcfg
导出...
关闭
关闭
关闭
Debug IO选择：
CLK:P00, DAT:P01
MCLR IO选择：
关闭
NVR擦除选择：
全部
加密使能：
打开
FLASH更新使能：
打开
带电烧录使能：
关闭

2.9. 使能加密功能后程序运行异常

2.9.1. 问题现象

TX8C101x/TX8W701x 系列在量产烧录时，在 PC 端系统配置界面上使能加密功能，再通过烧录器烧录程序后，出现功能异常的现象。

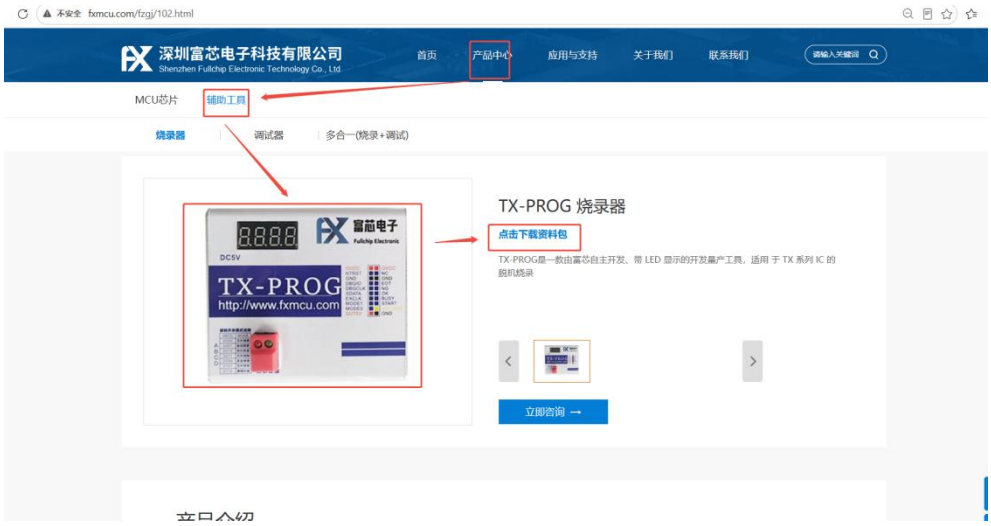
2.9.2. 解决方案

TX8C101x/TX8W701x 系列模式建议使用调试模式 说明：

- 建议量产烧录器使用调试模式（不加密）烧录：使用烧录量产模式受到电源波动干扰会有几率会导致芯片出厂数据被擦写，建议使用调试模式（不加密）烧录就不会有这类不良。

- 现烧录器安全，没必要量产烧录模式：我们现在烧录器协议都是私有的不开放的，烧录器不支持烧录文件回读。不存在像 ST，AVR...很多品牌协议是开放的，烧录器也支持回读烧录文件。

- 现烧录上位机软件已升级 http://fxmcu.com，并在界面功能上已添加了相关功能提示 。



2.10. 减小功耗将不使用的 IO 设为模拟模式

2.10.1. 问题现象

TX8C101x/TX8W701x 系列有多种封装芯片，因封装差异有些 IO 引脚未引出来，这些引脚在芯片内部处于输入浮空状态，会存在漏电情况。

2.10.2. 解决方案

在“system_init()”里打开“__SET_ALL_IO_ANALOG_MODE”宏，这个宏是将所有 IO 的初始状态设为模拟模式，在二次开发时不使用到的 IO 保持模拟模式可减小功耗。

不使用到的 IO 包括：

- 因封装差异未引出来的 IO；
- 封装上存在的 IO 但开发项目中未使用到。

如下图所示：

```
/**
  *@brief  System initialization function
  *@param  None
  *@retval None
  */
 void system_init(void)
{
    u8 hrcsc_data = HRCSC_STEP_VAL & 0x7f;    //获取HRC时钟细调值

    WDT_KEY    = 0xDD;                        //Close WDT

    FLASH_TIM0 = 0x55;
    FLASH_TIM1 = 0x54;                        //FLASH 16M 访问速度

    AIP_CON0   = 0x80 | hrcsc_data;           //使能 hirc
    CLK_CON0   = 0x03;                        //选择 hirc clk
    CLK_CON2   = 0x00;                        //系统时钟不分频
    CLK_CON6   = 0x1F;                        //FLASH烧写时钟32分频：1M

    delay_ms(1);                              //该延迟不可删除，保证烧录稳定性

//    __SET_ALL_IO_ANALOG_MODE;               //所有IO的初始状态设为模拟模式，在二次开发时不使用到的IO保持模拟模式可减小功耗
}
```

2.11. 函数 delay_ms 不准确的问题

2.11.1. 问题现象

TX8C101x/TX8W701x 系列的 SDK 中函数 delay_ms（）使用时，修改工程中代码后，重新编译下载，会出现延迟时间改变的情况。

2.11.2. 解决方案

参考示例芯片 TX8C101x/TX8W701x SDK 中函数 delay_ms（）是属于软件延迟，该函数本身的延迟时间是非准确的，会存在一定的误差。

但是如果出现修改工程后，重新编译下载会让延迟时间出现改变，则有可能是以下情况：

1. 中断打断：

delay_ms（）是软件延迟，优先级低于任何中断函数，一旦中断触发，delay_ms（）中的程序会被打断，进而导致延迟的时间出现改变，具体改变的时间，由打断的中断函数所决定。

2. delay_ms（）奇偶地址：

芯片 TX8C101x/TX8W701x 中由于 Flash 的特性，属于 16 Bit 位宽。而 delay_ms（）中的函数会有跳转指令（JB、JNB、JZ、JNZ、LCALL 等）所对应地址值，如果为偶地址，Flash 能按照 16 bit 对齐的位宽取值，如果是奇地址则是非对齐取值。因为 delay_ms（）存在多次跳转，会累积误差。因此会导致以上情况。

总结： 所以如果需要精准的延迟，请使用定时器方式。

2.12. Flash 的空间分配

2.12.1. 解决方案

TX8C126x/TX8M226x 系列 Flash 空间可规划为 EEPROM 区、Rolling Code 区、Code CRC 空间。

其中 EEPROM 区和 Rolling Code 区是可选择的，用户可以根据具体的产品需求，使能/关闭这两个功能。当这两个功能被关闭后，其 Flash 空间是可以被使用为 Code 区的。EEPROM 默认的空间大小为 256 Bytes，Rolling Code 的空间大小为 8 Bytes。

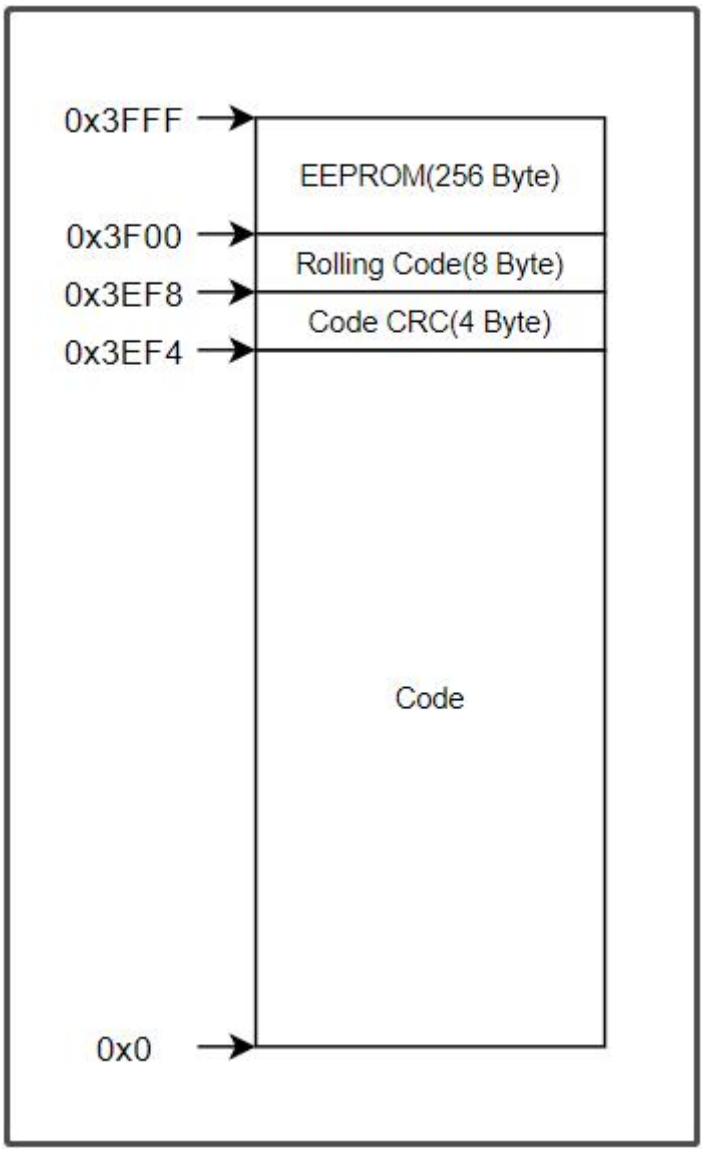
而 Code CRC 区是不可选择的，一定存在，空间大小为 4 Bytes。

在 TXProgrammer 的 PC 工具中，对应滚码功能和 EEPROM 功能的使能位，如下图所示：

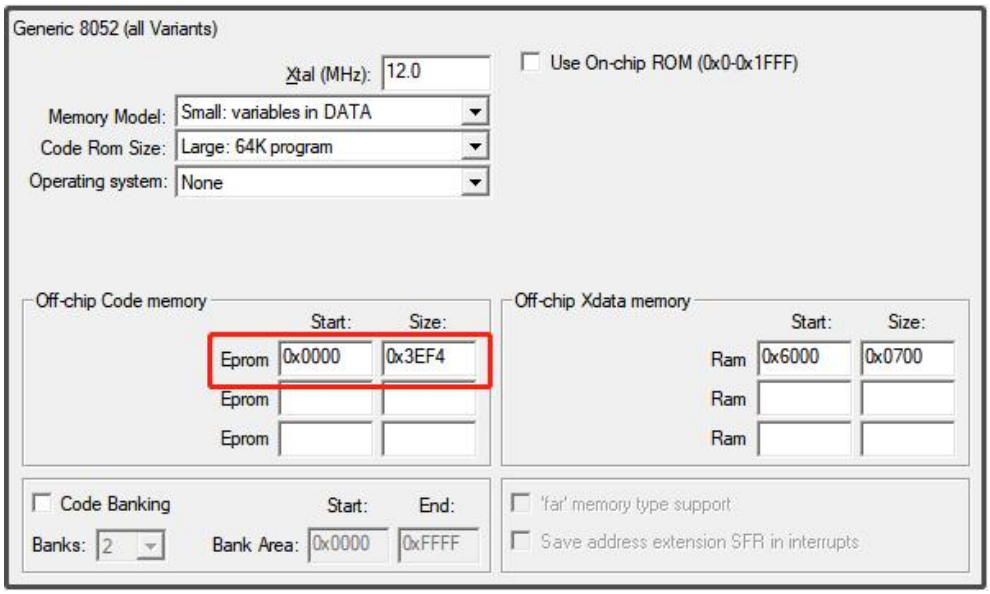


情况一：使能 EEPROM 和使能 Rolling Code：

Flash 的空间分配如下：

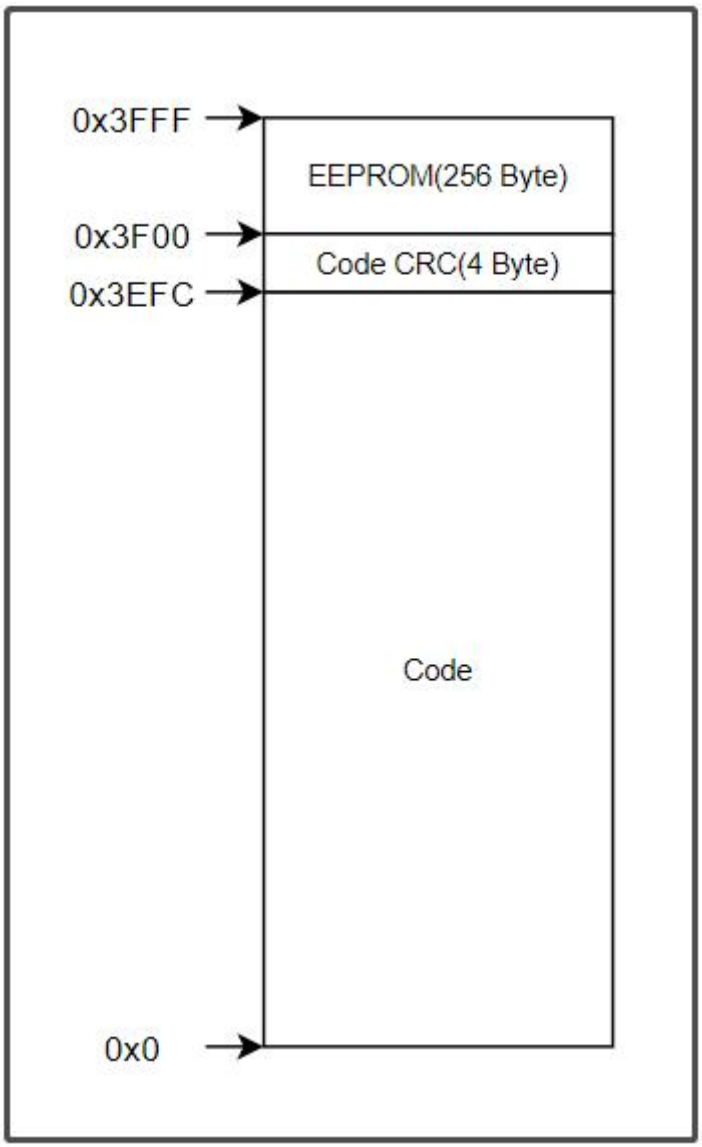


该情况下，Code 区的代码空间为 0x3EF4，所以标准 SDK 的示例工程中配置的 Code 大小则为 0x3EF4。对应查看开发 IDE 的设置，如下：

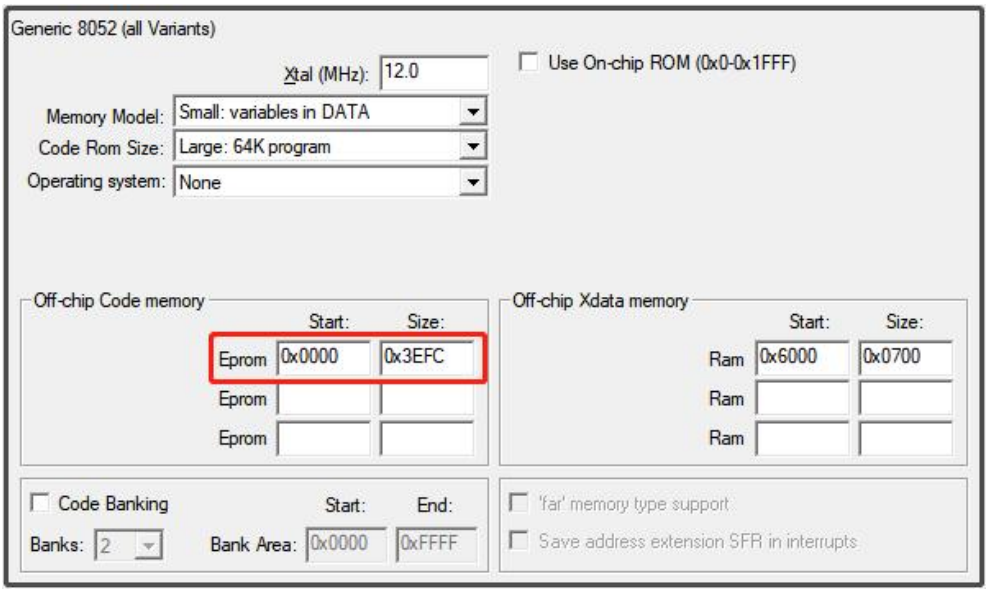


情况二：使能 EEPROM 和关闭 Rolling Code：

Flash 的空间分配如下：

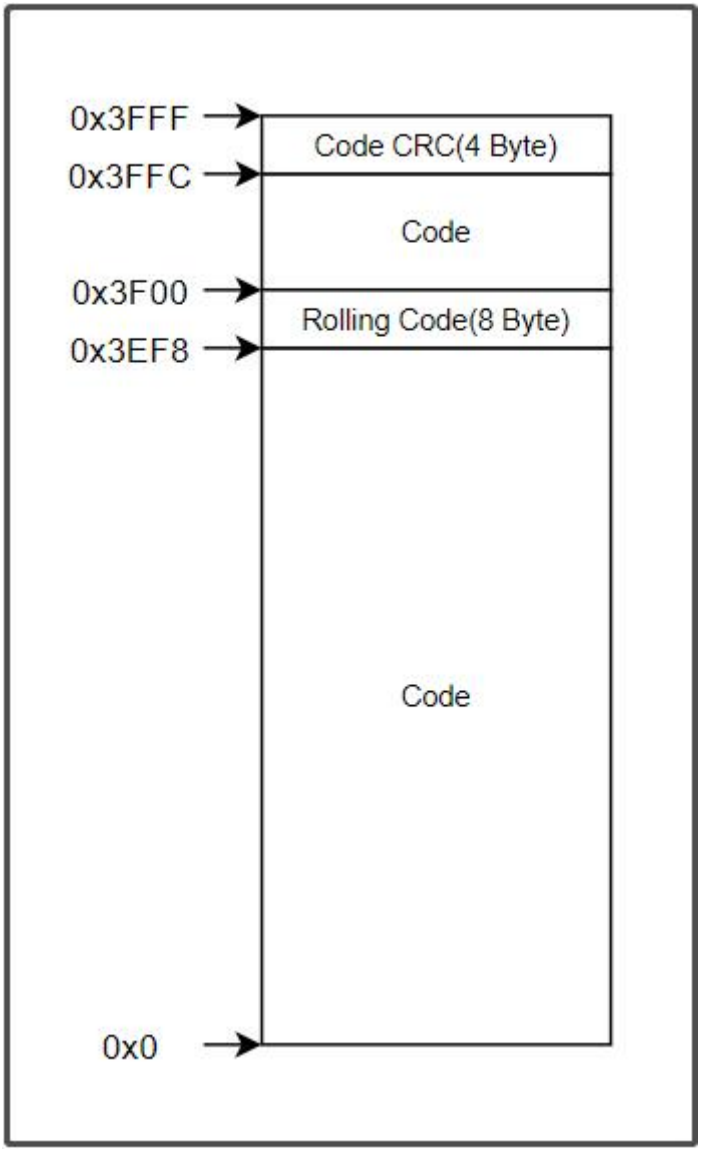


该情况下，Code 区的代码空间为 0x3EFC，所以 SDK 的示例工程中配置的 Code 大小则为 0x3EFC。对应查看开发 IDE 的设置，如下：



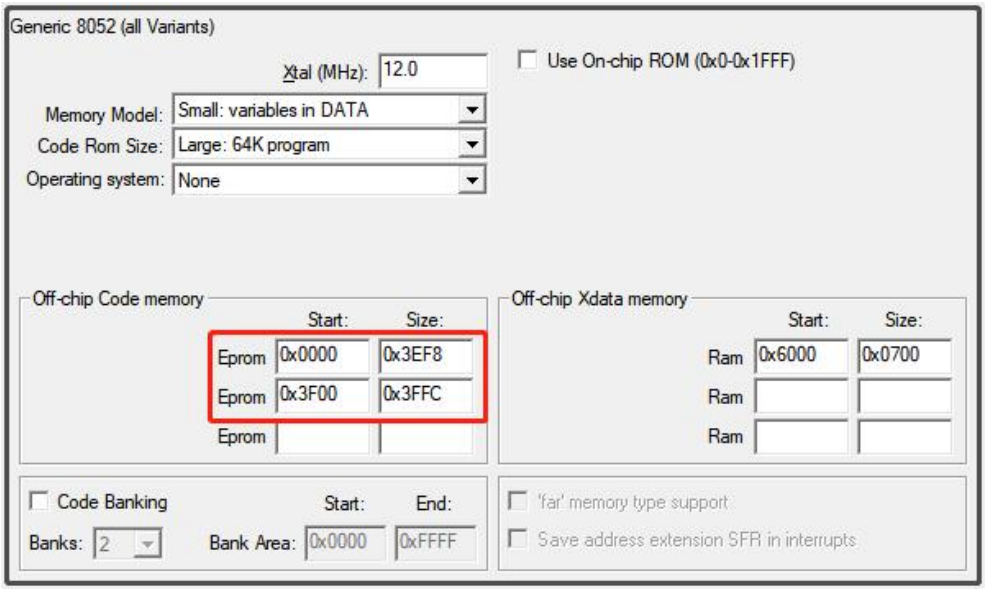
情况三：关闭 EEPROM 和使能 Rolling Code：

Flash 的空间分配如下：



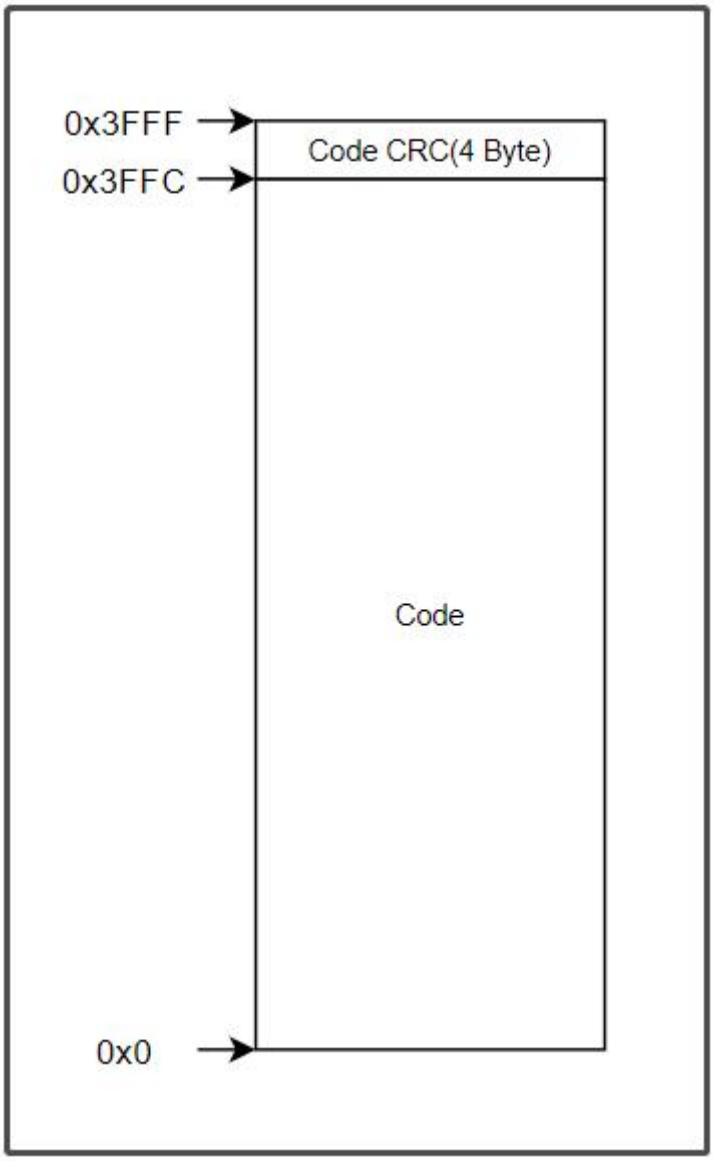
该情况下，Code 区的代码空间为 0x3FF4，所以 SDK 的示例工程中配置的 Code 大小则为 0x3FF4。

但是从图中的分配可以看出，code 被切割为 2 部分，所以在开发时，需要注意配置。对应查看开发 IDE 的设置，如下：

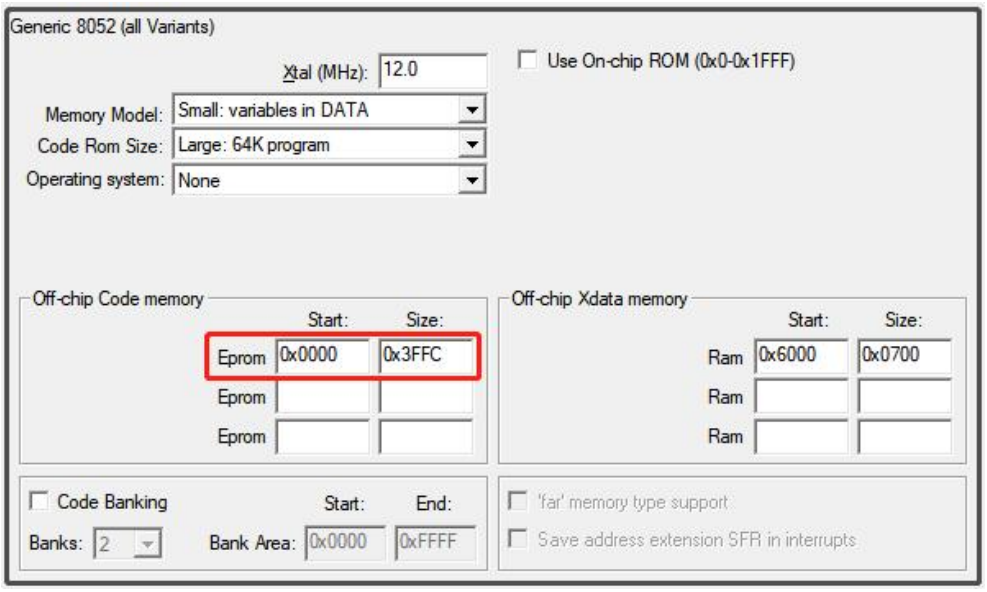


情况四：关闭 EEPROM 和关闭 Rolling Code：

Flash 的空间分配如下：



该情况下，Code 区的代码空间为 0x3FFC，所以 SDK 的示例工程中配置的 Code 大小则为 0x3FFC。对应查看开发 IDE 的设置，如下：


Generic 8052 (all Variants)
Xtal (MHz): 12.0
Use On-chip ROM (0x0-0x1FFF)
Memory Model: Small: variables in DATA
Code Rom Size: Large: 64K program
Operating system: None
Off-chip Code memory
Start:
Size:
Eprom 0x0000 0x3FFC
Eprom
Eprom
Off-chip Xdata memory
Start:
Size:
Ram 0x6000 0x0700
Ram
Ram
Code Banking
Start:
End:
Banks: 2
Bank Area: 0x0000 0xFFFF
'far' memory type support
Save address extension SFR in interrupts

2.13. Loader 功能

2.13.1. 解决方案

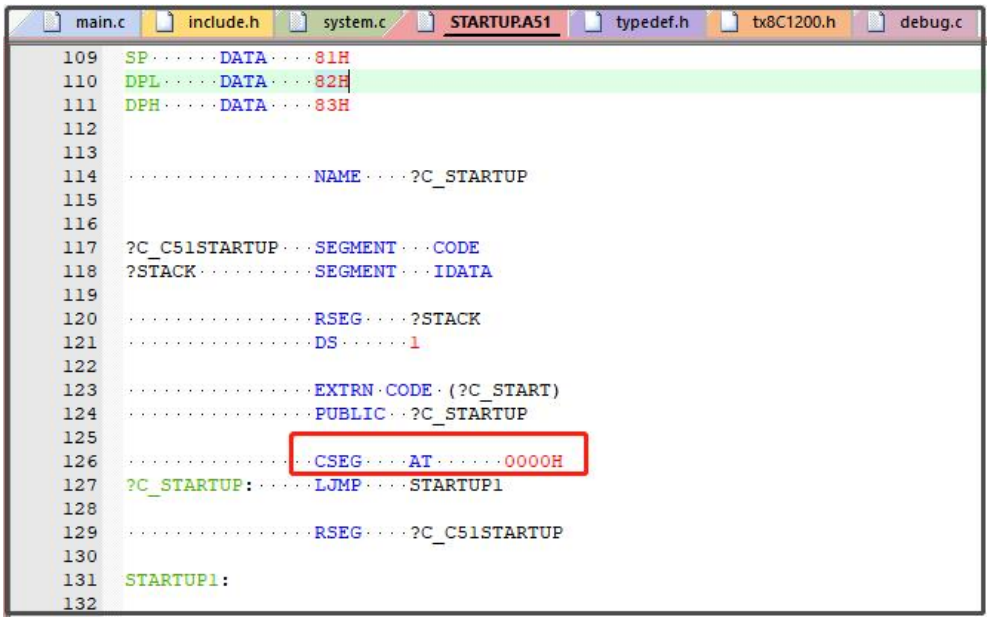
TX8C126x/TX8M226x 系列支持在程序中划分 Loader 的功能。

Loader 空间可以划分为 0、2K、4K 和 6K，其中 0K 表示没有 Loader。

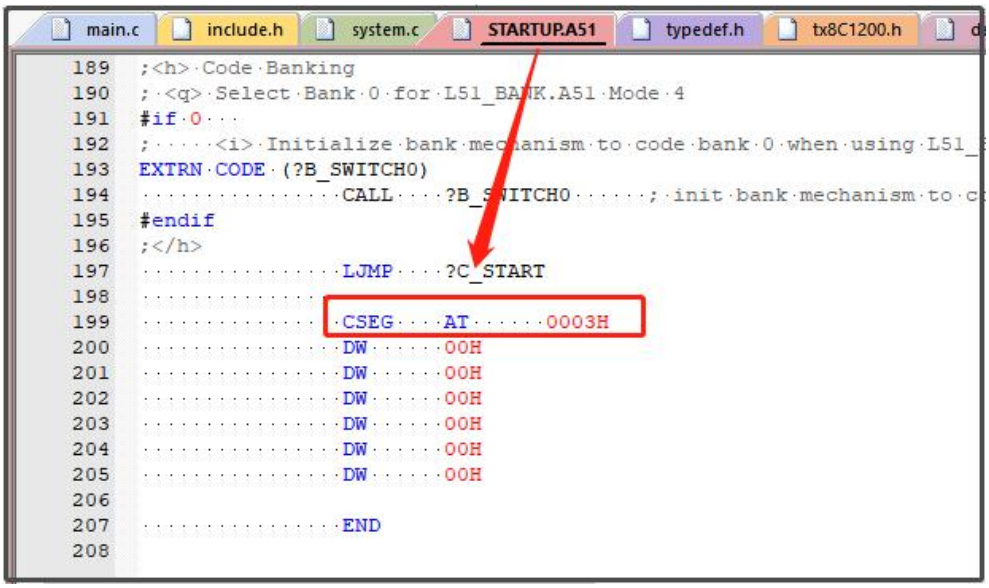
Loader 和 Main 的工程配置是不一样的，需要在工程上做区分，具体参考示例工程“Loader\Loader_Project”和“Loader\Main_Project”。

调试阶段：

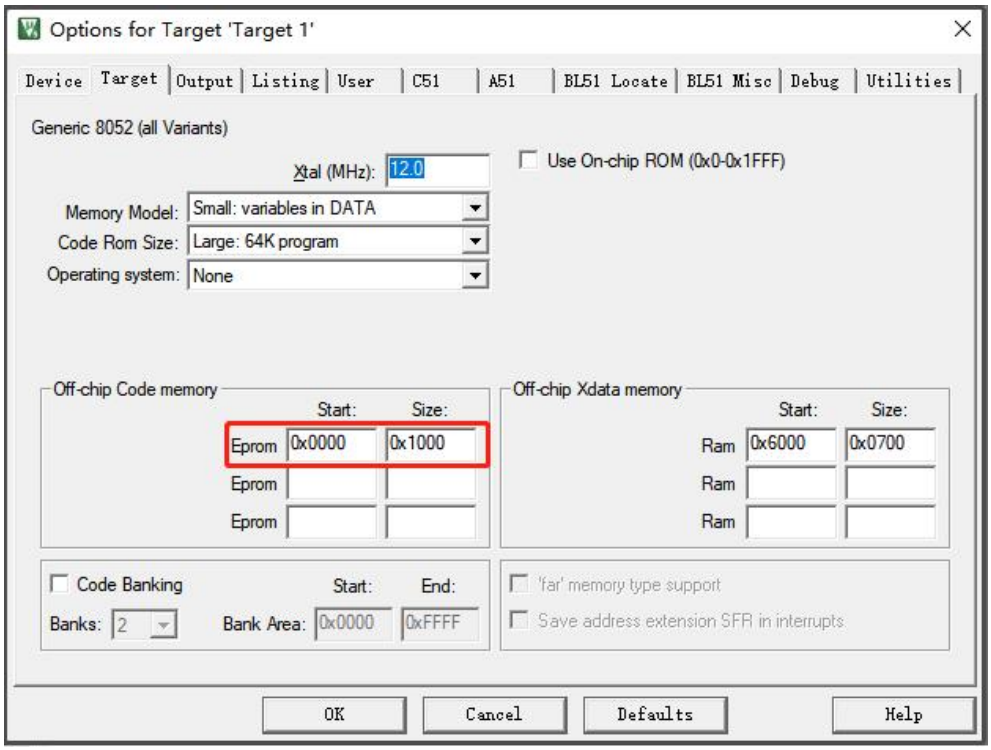
Loader 的工程需要留意以下配置：



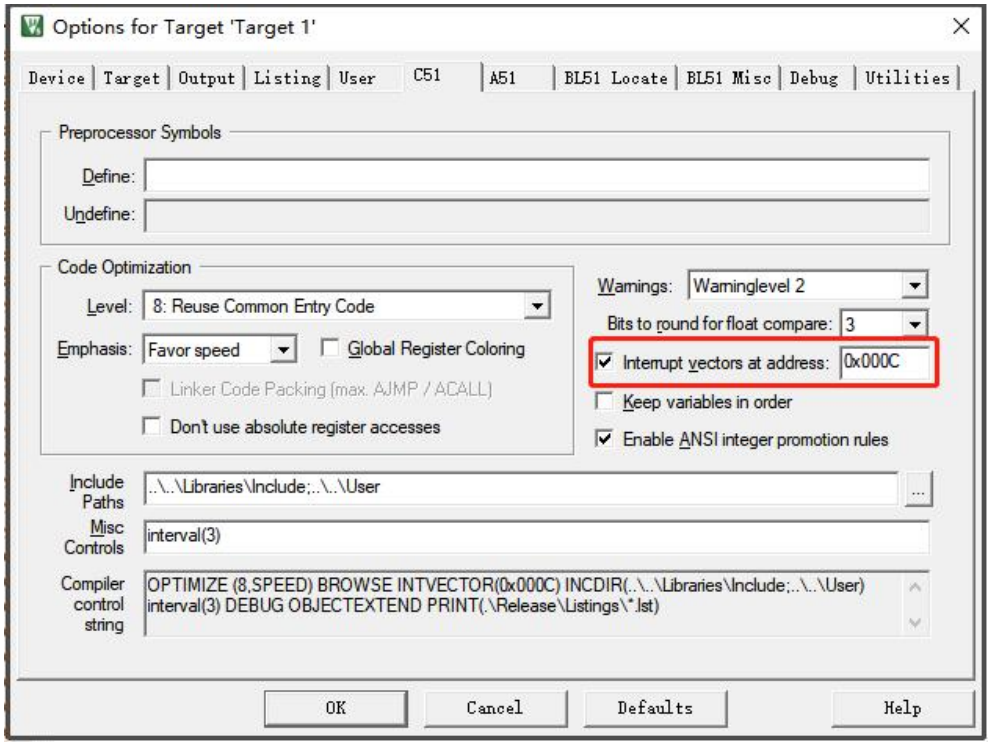
启动地址配置为“0000H”。



这里的地址要改为“0003H”。



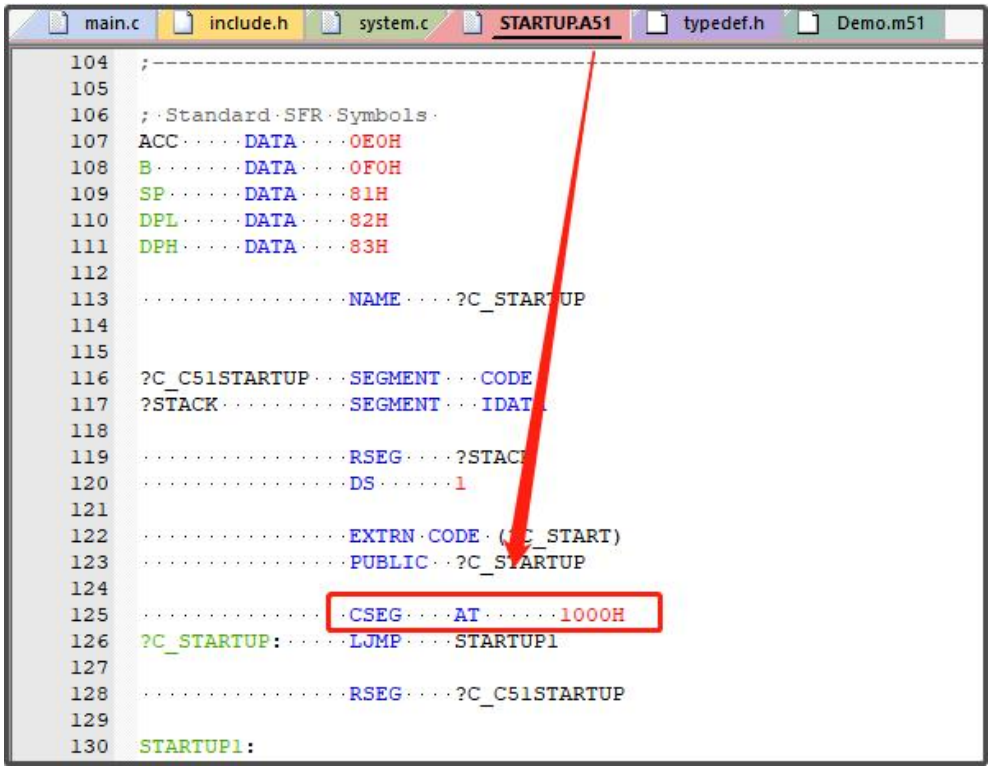
Code 空间的大小需要根据 Loader 的空间大小配置，图中默认是将 Loader 的空间规划为 4K。



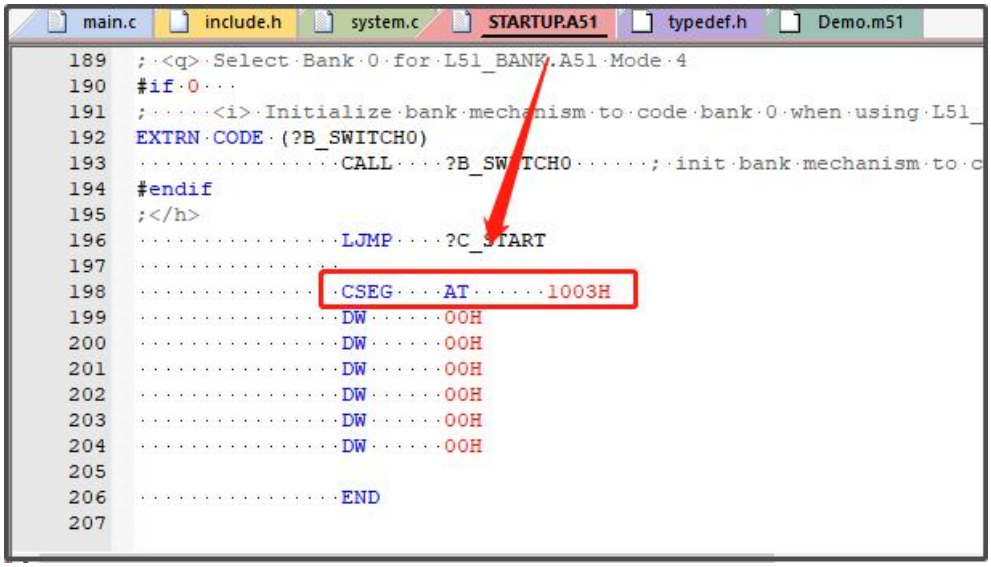
配置 Loader 工程的中断向量表起始地址为“0x000C”。

其余的功能请参考“Loader\Loader_Project”中的程序。

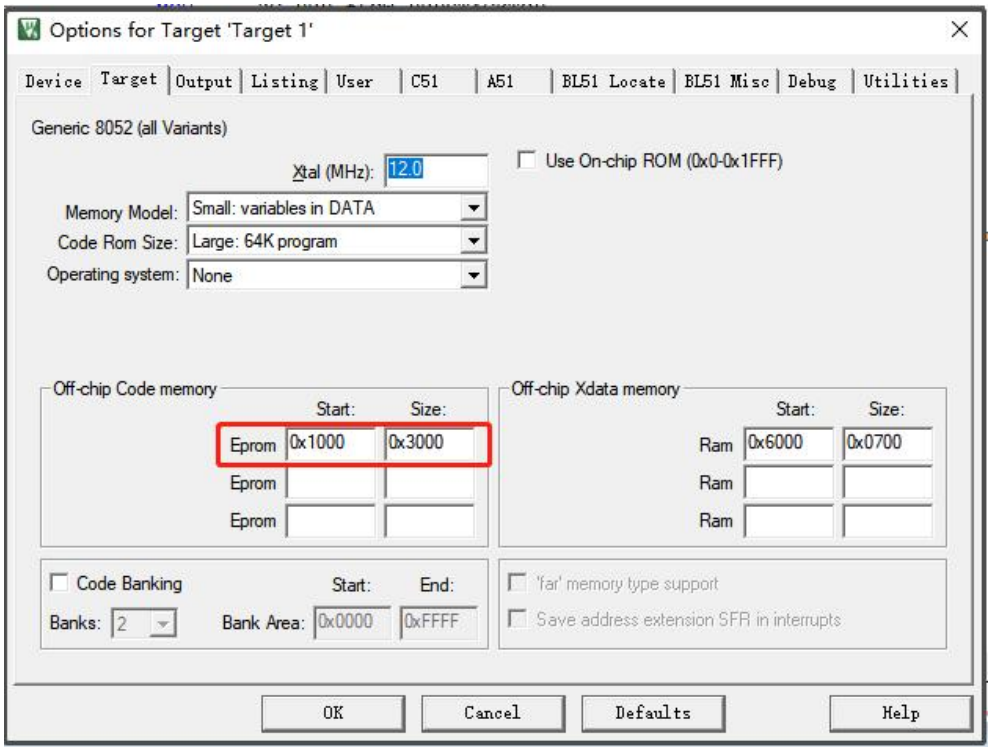
Main 的工程需要留意以下配置：



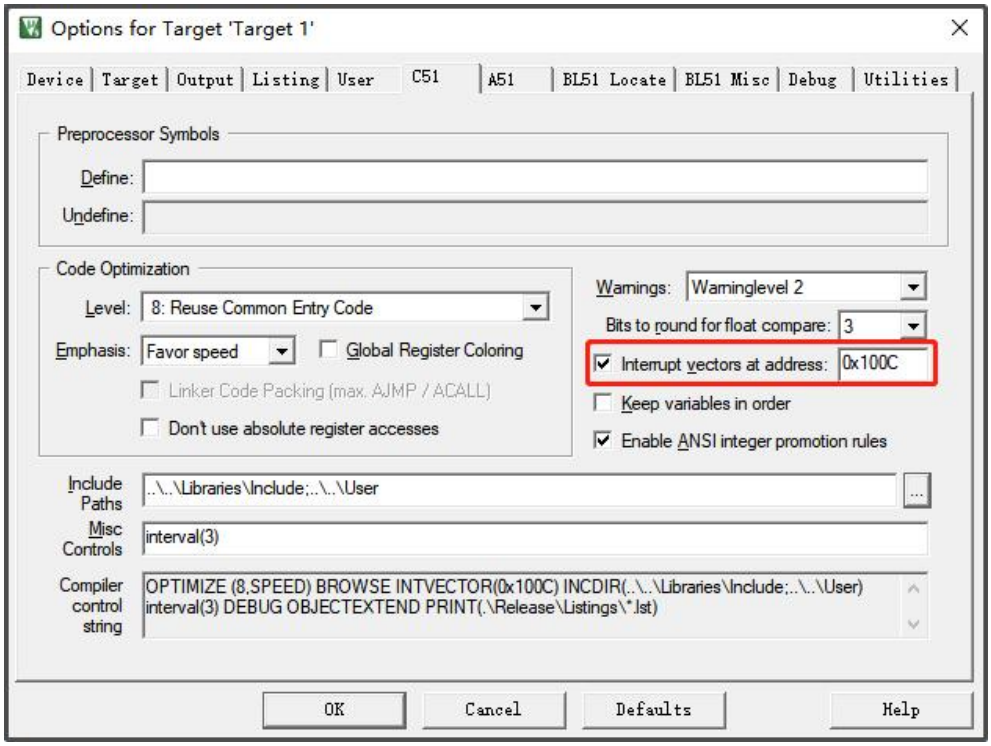
启动地址配置为“1000H”。



这里的地址要改为“1003H”。



Main 工程中 code 空间的大小需要根据 Loader 的空间大小配置，图中默认是将 loader 的空间规划为 4K，所以 Main 工程中的 code 起始地址则从 0x1000 开始。



配置 Main 工程的中断向量表起始地址为“0x100C”。

其余的功能请参考“Loader\Main_Project”中的程序。

量产阶段：

1. 在量产阶段时，需要将 PC 工具 TXProgrammer 升级到 V2.1.24 版本及以上；



根据应用中设置的 Loader 大小，上述工程默认是 4K，这里选择“0~4K”。

2. 注意“Code 保护选择”的配置，建议配置项按照 loader 的大小进行配置，切记不可涵盖到 Main 区域空间，不然 Main 的程序区无法更新程序。


系统配置
机台配置
量产配置
芯片配置
文件选择
./dll/parameter_test.setcfg
导入...
./dll/TXM8C1010_set_config_test.setcfg
导出...
系统配置
下载模式：
量产模式
Code加密选择：
关闭
Code保护选择：
（EEPROM使能）
0~4K
FLASH BOOT选择：
0~4K
下载完成，复位使能：
关闭
MCLR IO选择：
关闭

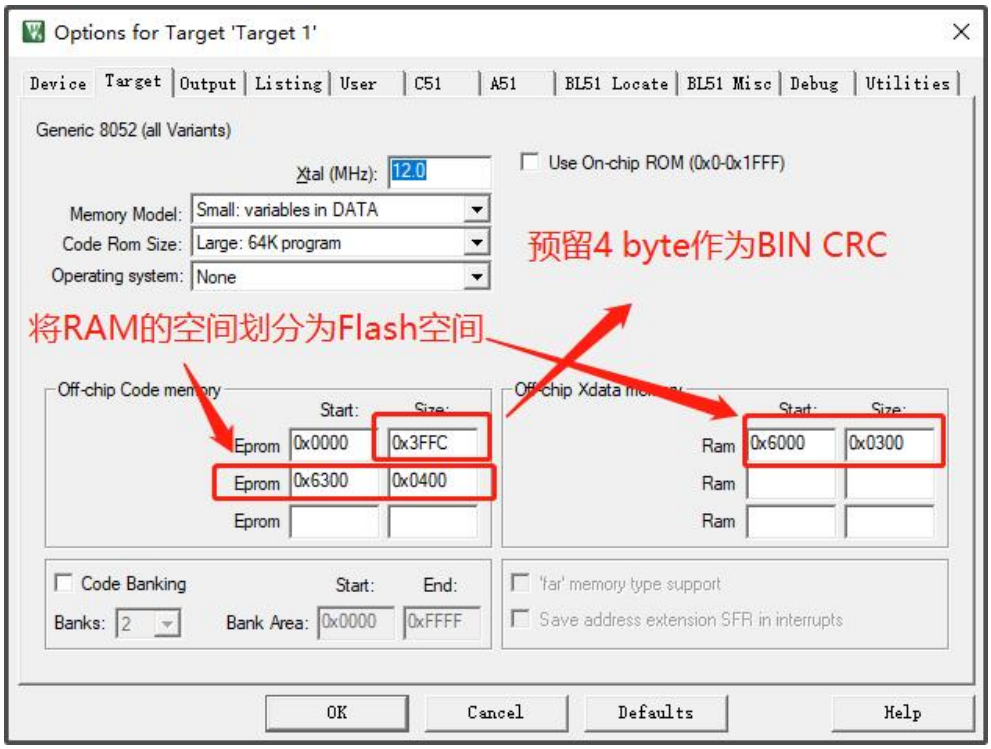
2.14. 在 TXPROG 量产模式下烧录异常问题

2.14.1. 解决方案

TX8C126x/TX8M226x 系列，在上述功能 7 在 TXPROG 的“量产模式”下，烧录异常的问题。

通过以下更新修改：

1. 程序空间需要预留末尾的 4 byte 给 BIN CRC，配置如下：



2. 将 BinScript.exe 工具的版本更新到 V1.0.5 版本，如何确认版本？

按照以下操作：

2.1、打开批处理文件“make.bat”，在末尾加上命令“pause”，保存点击批处理“make.bat”运行，则会出现以下界面。

make.bat

```
SET FormerFileHex=Demo.hex
SET FormerFileM51=Demo.m51
SET FormerFileBin=demo.bin

if exist %FormerFileHex% (
    del /q /f Demo.hex
)
if exist %FormerFileM51% (
    del /q /f Demo.m51
)
copy .\Objects\Demo.hex Demo.hex
copy .\Listings\Demo.m51 Demo.m51

BinScript.exe BinScript.BinScript

setlocal enabledelayedexpansion
for /f "delims=" %%i in ('dir /b %FormerFileBin%') do (
    SET fileLen=%%~zi
    echo !fileLen!
    
    if !fileLen! GTR 16384 (
        BinScript.exe FlashToRam.BinScript
        echo FlashToRAM:!fileLen!
    ) else ( echo Flash:!fileLen! )
)

pause
```

```
C:\WINDOWS\system32\cmd.exe
已复制         1 个文件。
已复制         1 个文件。
BinScript.exe v1.0.5
Successfully output a BIN file: demo.bin
25360
BinScript.exe v1.0.5
Successfully output a BIN file: demo.bin
Successfully output a HEX file: demo.hex
FlashToRAM:25360
请按任意键继续. . .
```

2.2、修改配置文件“FlashToRam.BinScript”，将 bin 和 hex 文件的输出大小设置为 0x3FFC，需要空出 4 byte 作为 BIN CRC。

```
1  memset(ibuf, 0);
2  read(bin, Demo.bin);
3  remap(v_ax, 0, v_lx);
4  remap(6300, 3300, 400);
5  memmode(obuf, 8, big);
6  outrange(0, 3FFC, 10);
7  write(bin, Demo.bin);
8  outrange(0, 3FFC, 10);
9  write(hex, Demo.hex);
10
```

3. 将烧录工具 TXProgrammer 升级到 V2.1.22 版本。

2.15. 将 Flash 函数加载到 RAM 上运行

2.15.1. 解决方案

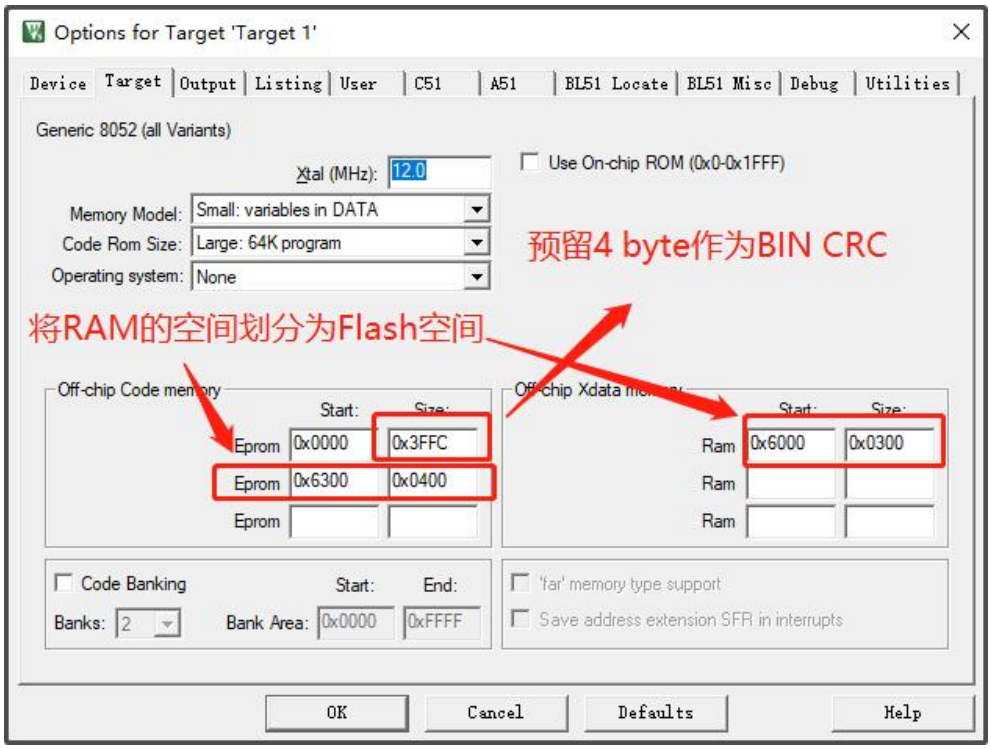
在 TX8C126x/TX8M226x 系列的 SDK 中打开示例工程“Flash_To_SRAM_Run”。

将 Flash 的函数加载到 RAM 中，首先需要将函数编译在 RAM 的地址上，这样才能避免因为 AJMP\LJMP 指令导致程序在 RAM 运行后，又跳转回 Flash 空间。

示例工程中，将函数 add_func_inram(u8 cef1, u8 cef2)和函数 dec_func_inram(u8 cef1, u8 cef2)加载到 RAM 中运行。

步骤一：将函数编译到 RAM 中。

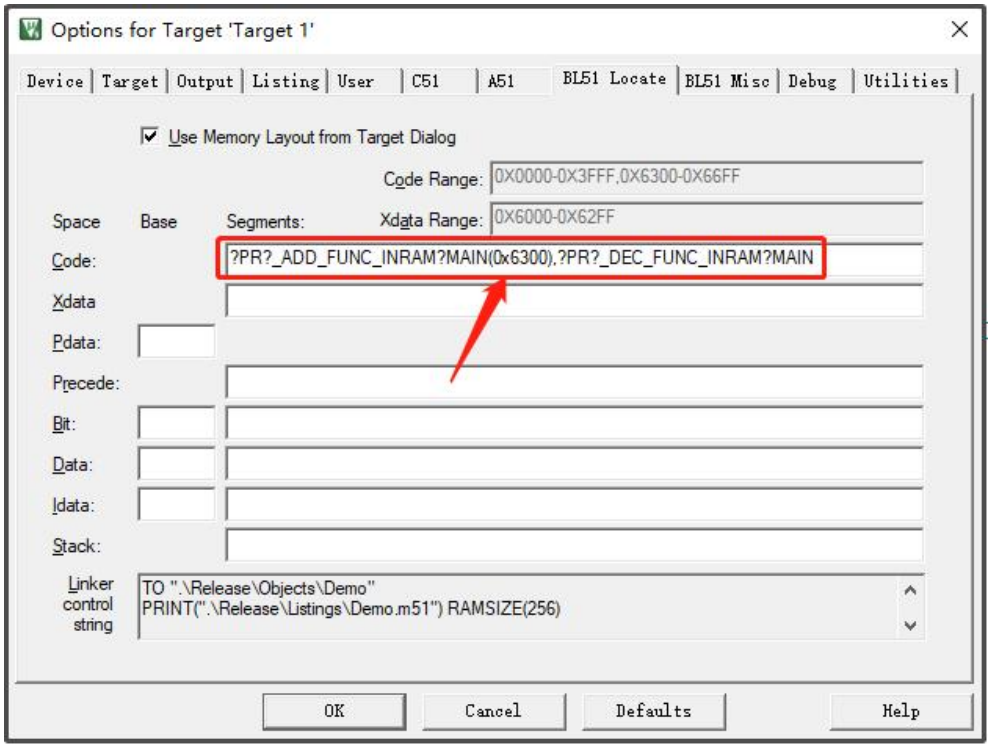
（1）划分 RAM 空间为 Flash 空间：



TX8C126x/TX8M226x 系列的 RAM 空间大小为 0x6000~0x66FF 之间，示例中是将 0x6000~0x62FF 划分为 xdata 空间，将 0x6300~0x66FF 划分为 Flash 空间。这里需要注意，Code 空间需要设置为 0x3FFC，需要在末尾预留 4 byte 作为 bin 文件的 CRC。

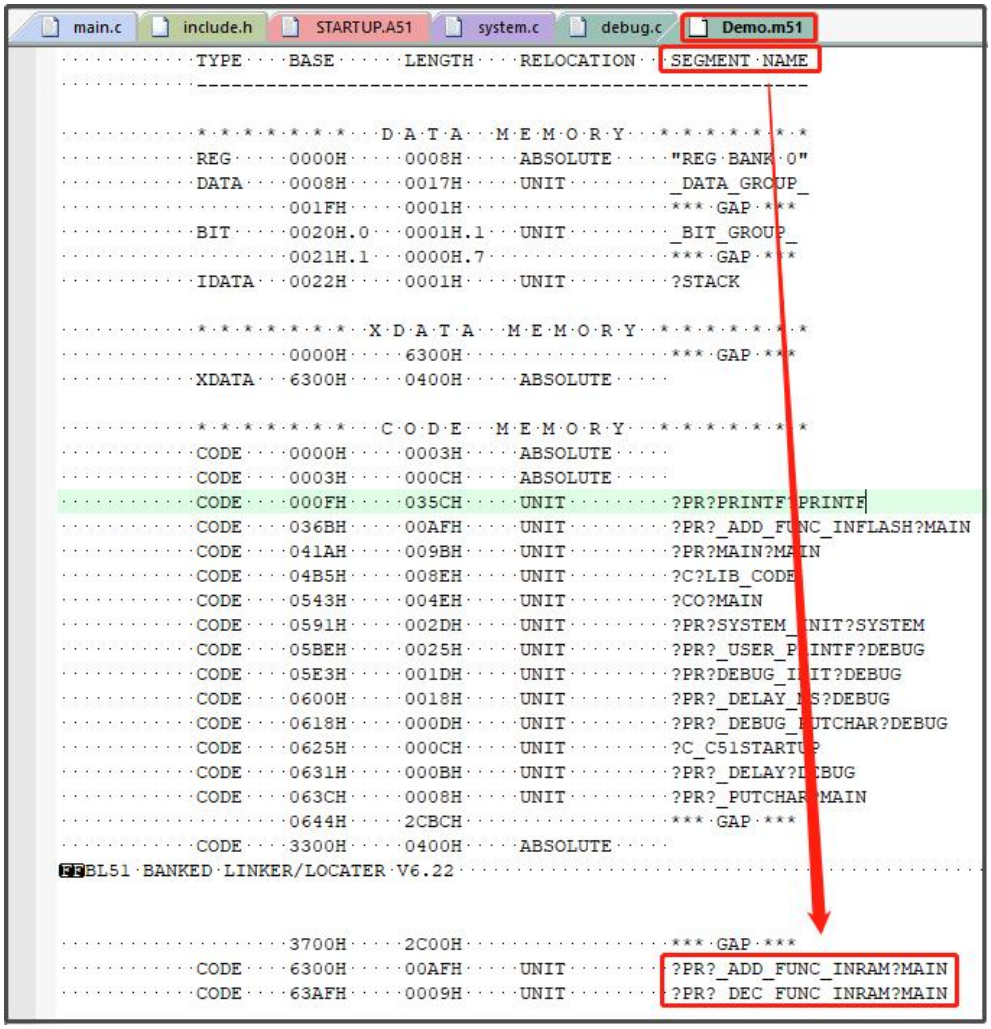
注意：将 RAM 分配为 Flash 时，地址需要 0x100 对齐，长度需要 0x100 对齐。

（2）将要加载 RAM 的函数指定编译到 RAM 中：



在文件“Demo.m51”中找到要函数编译后的标识 SEGMENT NAME，两个函数的标识符分别为：

- add_func_inram（?PR?_ADD_FUNC_INRAM?MAIN）
- dec_func_inram（?PR?_DEC_FUNC_INRAM?MAIN）



找到函数对应的 SEGMENT NAME 后，将其设置到上方配置图中，每加一个段，使用“，”进行间隔，只需要第一个段指定 RAM 的地址即可。

步骤二：程序上操作

（1）RAM 空间和对应 Flash 的储存空间声明：

以下是 RAM 空间和 Flash 空间的变量声明，其中：

- FLASH_TO_RAM_LEN：是指 RAM 作为函数运行的空间大小，与步骤一中划分 RAM 空间的配置保持一致，设置为 0x400，按照 0x100 对齐；

- sram_func_xdata：是 RAM 作为函数运行的 buf，在这里声明，是为了方便将数据从 Flash 加载到 RAM 的操作，注意后面需要指定地址，该地址也是与步骤一中划分 RAM 空间的配置保持一致，设置为 0x6300，地址按照 0x100 对齐；

- sram_func_code：是 RAM 作为函数时，将数据缓存到 FLASH 中，该操作会在下载时执行，但是工程需要提前声明空间。需要注意的该 buf 也是需要指定地址空间的，该地址

空间是会将 0x6000~0x66FF 映射到 0x3000~0x36FF 中。所以如果 RAM 作为函数的起始的地址空间为 0x6300，则映射的 FLASH 中为 0x3300。

```
//该工程的使用请参考FAQ文档
#define FLASH_TO_RAM_LEN            (0x400)
xdata u8 sram_func_xdata[FLASH_TO_RAM_LEN] _at_ 0x6300;      //将函数放到RAM中的buf
code u8 sram_func_code[FLASH_TO_RAM_LEN] _at_ 0x3300;        //函数存在在Flash中的buf，该buf是预留的，
```

（2）将 Flash 中的函数加载到 RAM 中：

```
//将Flash中的程序加载到xdata的中
//flash_addr：是指函数的起始地址，可以根据文件"Demo"查找函数的地址
//xram_addr：是指加载到RAM的地址，这里是指xdata空间，0x6000~0x6700之间
//len：是指要加载的长度大小
#define LOADER_FLASH_TO_XRAM(flash_addr, xram_addr, len)                        \
do {                                                                            \
    u16 i;                                                                      \
    for(i=0; i<(len); i++) {                                                    \
        *(u8 volatile xdata *)((xram_addr) + i) = *(u8 code *)((flash_addr) + i); \
    }                                                                           \
} while(0)
```

```
void main(void)
{
    s16 add_res, dec_res;

    system_init();

    //看门狗默认打开，复位时间2s
    WDT_KEY = WDT_KEY_VAL(0xDD);                //关闭看门狗（如需配置看门狗请查看"WDT\WDT_Reset"示例）

    //将函数从Flash加载到RAM中
    LOADER_FLASH_TO_XRAM((u16)sram_func_code, (u16)sram_func_xdata, FLASH_TO_RAM_LEN);
```

程序运行后，需要将 Flash 的函数加载到 RAM 中。然后函数就正常调用即可，不需要其他额外的操作。

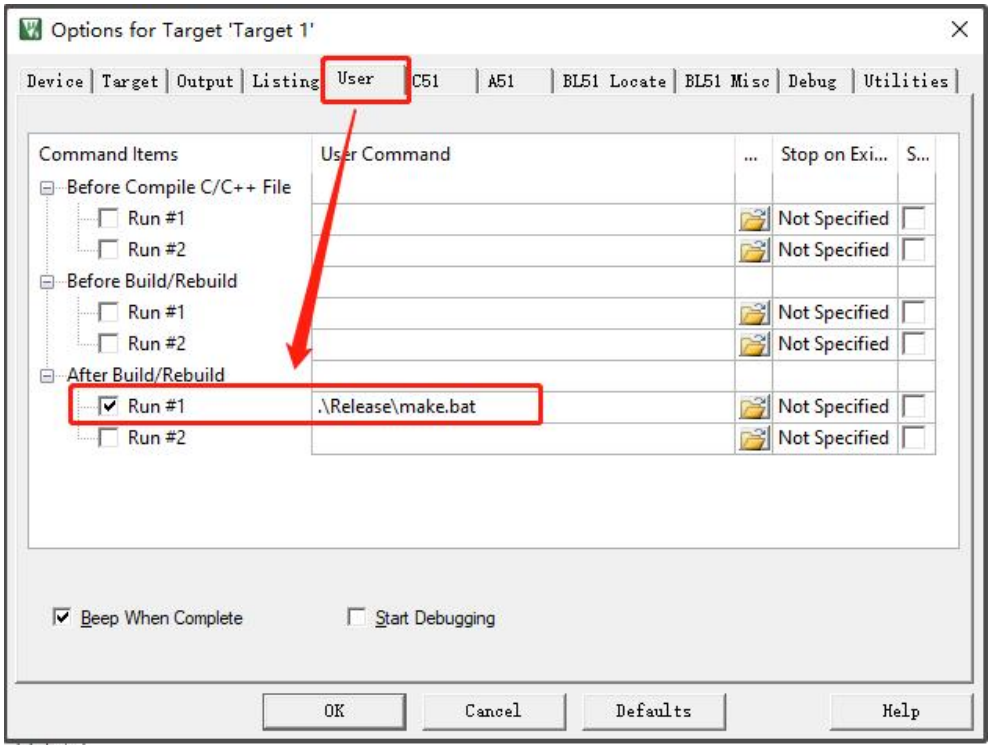
注意： 要支持上述描述的功能，需要将 TXLink 的驱动升级到“TXLink_keil_C51_driver_install_V1.2.9”及以上版本才能实现上述功能。

步骤三：烧录器 TXProgrammer 的烧录文件

（1）因为将 RAM 的空间变成 Flash 空间，所以编译后的 hex 文件的 Flash 数据一定会超过 0x4000 的地址，所以需要将文件进行转换。

打开“Flash_To_SRAM_Run”示例工程，进入到“Project\Keil\Release”的目录下，则会与比其他的示例工程多了 BinScript.BinScript、FlashToRam.BinScript、BinScript.exe 和 make.bat 的文件，如果是移植到其他工程则需要将这几个文件移植过去。

在编辑界面中打开设置，将 make.bat 的批处理添加到编译完成的调用配置中。



如果不在上述配置中添加，用户也可以手动运行 make.bat。

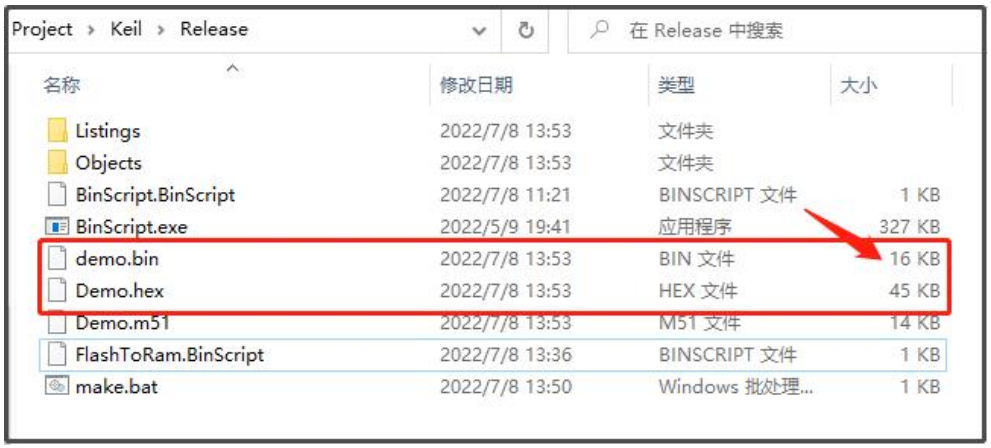
（2）批处理的修改。

同样的如果步骤一中的配置修改了，则批处理的配置文件也需要同步修改，打开文件 FlashToRam.BinScript，在下图中对应的位置修改即可。

```
1  memset(ibuf, 0);
2  read(bin, Demo.bin);
3  remap(v_ax, 0, v_lx);
4  remap(6300, 3300, 400);
5  memmode(obuf, 8, big);
6  outrange(0, 3FFC, 10);
7  write(bin, Demo.bin);
8  outrange(0, 3FFC, 10);
9  write(hex, Demo.hex);
10
```

（3）最后注意，编译出来的 bin 文件一定是小于等于 16K-4，如果大于 16K-4 使用烧录器下

载会异常报错。使用 hex 和 bin 文件都可以进行下载。



3. TX8C101x-FAQ

3.1. TXPROG 在板烧录时报错 ER03 或 ER15 错误

3.1.1. 问题现象

TX8C101x 使用 TXPROG 烧录器在板烧录时部分芯片烧录失败，数码管显示 ER03 或 ER15 错误

ER03:芯片连接失败(烧写器连接不到芯片)

ER15:芯片 NVR 数据配置异常(NVR 在配置芯片信息时，出现异常)

3.1.2. 解决方案

ER03:芯片连接失败(烧写器连接不到芯片)

- 检查芯片和烧录器的连线是否短路或者断路；
- 检测烧写器的 OVCC 跳线帽端子是否被拔除,烧录器需要 5v 供电；
- 检测芯片是否被供电（如果芯片中有关闭 ISD 的操作，则此时烧录芯片端不能供电，烧写器需要掉电复位）
- ISD 通信速率过高导致芯片连接失败，需要降低 ISD 通信速率，可以在上位机配置界面修改默认为 1000khz；把上电稳定时间加大





- 去掉板子上的大电容,缩短烧录器的线长(200mm 以内)。

ER15:芯片 NVR 数据配置异常(NVR 在配置芯片信息时，出现异常)

- 建议 TXPROG 设置为调试模式烧录：
- 使用量产模式会操作 NVR 区，烧录器如受到电源干扰有几率导致芯片异常。
- 以住案列：客户使用量产模式，在板烧录出现约 2%的不良，后改为调试模式烧录验证未再发现类似不良。
- 现烧录器安全，没必要量产烧录模式（及加密）：
- ST，AVR...很多品牌协议是开放的，烧录器也支持回读烧录文件。
- 我们现在烧录器协议都是私有的不开放的，烧录器不支持烧录文件回读。
- 现烧录上位机软件已升级：
- 在界面功能上已添加了相关功能提示 ，建议客户使用调试模式。
- 请从富芯官网下载新版软件 http://fxmcu.com/fzgj/102.html，

4. TX8C126x-FAQ

4.1. TX8C126x 系列滚码功能

4.1.1. 解决方案

TX8C126x 系列支持烧录器滚码的烧录配置功能。注意滚码的烧录只能在烧录器中完成，如果客户烧录滚码后，还需要进行调试，要注意在使用 TXProgrammer 时，将“下载模式”设置为调试模式，再用编辑环境继续开发。

配置 TXProgrammer：

1. 烧录工具 TXProgrammer 选择“TX8C126x”，打开“设置”，即会出现以下界面；

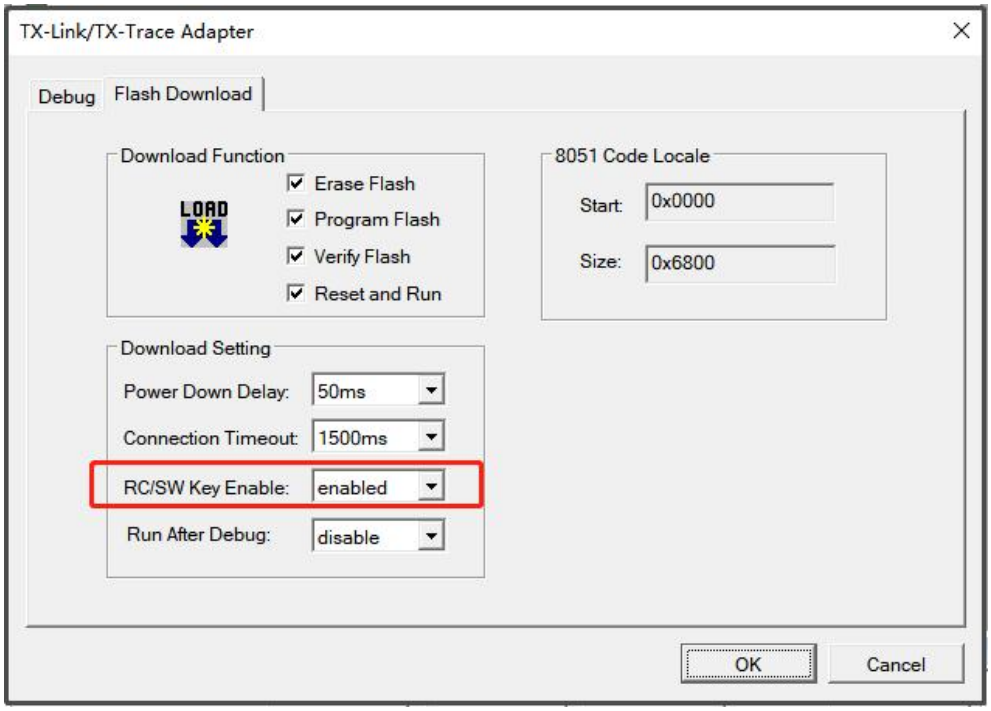


2. 勾选“滚码使能”，其中地址的值是固定的，不可修改；初始值和累加的步进值，用户可以根据产品而定。

配置开发工程：

1. 打开 TX8C126x 的软件开发包，打开文件目录下的“Examples\UUID\RollingCode”工程，该工程就是获取芯片滚码的示例；

2. 打开编辑工具设置选项，将“TX-Link/TX-Trace Adapter”->“Flash Download”->“RC/SW Key Enable”选择“enable”。没有勾选的话，使用 TXProgrammer 烧录后的滚码会被擦除掉。


TX-Link/TX-Trace Adapter
Debug
Flash Download
Download Function
LOAD
Erase Flash
Program Flash
Verify Flash
Reset and Run
8051 Code Locale
Start:
0x0000
Size:
0x6800
Download Setting
Power Down Delay:
50ms
Connection Timeout:
1500ms
RC/SW Key Enable:
enabled
Run After Debug:
disable
OK
Cancel

4.2. TX8C126x 系列软件加密功能

4.2.1. 解决方案

TX8C126x 系列支持烧录器烧录软件密钥功能。注意软件密钥的烧录只能在烧录器中完成，如果客户烧录软件密钥后，还需要进行调试，要注意在使用 TXProgrammer 时，将“下载模式”设置为调试模式，再用编辑环境继续开发。

配置 TXProgrammer：

1. 烧录工具 TXProgrammer 选择“TX8C126x”，打开“设置”，即会出现以下界面：



2. 勾选“密钥设置使能”，其中地址的值是固定的，不可修改；

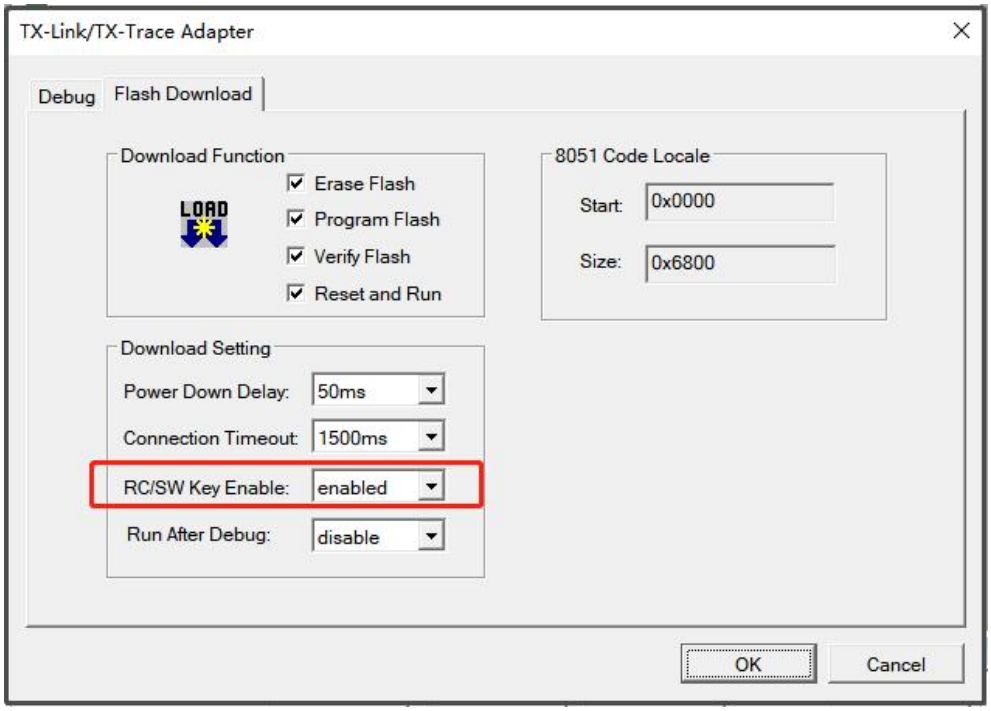
3. “模式”一共有三种，分别是默认模式、CRC8 模式和 CRC32 模式：

- 默认模式：将 UUID 与界面配置的“初值”进行累加操作；
- CRC8 模式：将 UUID 进行 CRC8 的运算，其中 CRC8 的 Poly 值可配置，即为“初值”；
- CRC32 模式：将 UUID 进行 CRC32 的运算，其中 CRC32 的 Poly 值可配置，即为“初值”。

配置开发工程：

1. 打开 TX8C126x 的软件开发包，打开文件目录下的“Examples\UUID\SW_Encryption”工程，该工程就是获取芯片软件密钥和校验软件密钥的示例；

2. 打开编辑工具设置选项，将“TX-Link/TX-Trace Adapter”->“Flash Download”->“RC/SW Key Enable”选择“enable”。没有勾选的话，使用 TXProgrammer 烧录后的软件密钥会被擦除掉。


TX-Link/TX-Trace Adapter
Debug
Flash Download
Download Function
Erase Flash
Program Flash
Verify Flash
Reset and Run
8051 Code Locale
Start:
0x0000
Size:
0x6800
Download Setting
Power Down Delay:
50ms
Connection Timeout:
1500ms
RC/SW Key Enable:
enabled
Run After Debug:
disable
OK
Cancel

4.3. TX8C126x 系列使用 6K Loader 的配置说明

4.3.1. 问题现象

将 loader 配置为 6K 时，出现中断入口函数无法对应上的问题。

将 FLASH_BOOTCON[0:1]=0x3 时，而 DPCFG[6:7]中没有对应的“中断起始地址为 0x1800C”的配置。

17.5.13. **FLASH_BOOTCON**

Addr = 0xBE（SFR）

Bit(s)	Name	Description	R/W	Reset
7: 2	BOOTKEY	**输入 0x15，将强制从 0 地址执行程序，输入 0x3A，将强制从 BOOTADDR 配置的区执行程序**	WO	0x0
1: 0	BOOTADDR	**跳转到代码区的地址** 0x0：默认从 0 开始 0x1：从 2K 开始 0x2：从 4K 开始 0x3：从 6K 开始	RW	0x0

Note: 配置完该寄存器后，需要执行软复位才可以进行代码区跳转。

2.6. **数据指针控制寄存器（DPCFG）**

Addr = 0x86（SFR）

Bit(s)	Name	Description	R/W	Reset
7: 6	IA	**中断地址起始区域选择** 0x0：中断起始地址为 0x000C 0x1：中断起始地址为 0x80C 0x2：中断起始地址为 0x100C 0x3：中断起始地址为 0x600C	RW	0x0

4.3.2. 解决方案

在将 loader 配置为 6K 时（即 FLASH_BOOTCON[0:1]=0x3），中断的入口配置对应配置为 DPCFG[6:7]=0x3，只是相对于 4K loader 来说，多一步将中断的入口地址程序复制到 SRAM 中（即 0x6000 的地址）。

1. 对于 Loader 工程来说，与“9 - TX8C126x 系列 Loader 功能”步骤一样；

2. 而 Main 工程，就只需要在程序使用中断之前，添加一段将中断的入口地址复制到 SRAM 的操作。程序如下：

```
/**
  * @brief  Main program.
  * @param  None
  * @retval None
  */
void main(void)
{
    u16 i;
    u8 xdata *p_data8;
    u8 code *p_cdata8;

    // 看门狗默认打开，复位时间2s
//    WDT_KEY = WDT_KEY_VAL(0xDD);              //  关闭看门狗 (如需配置看门狗请查看"WDT\WDT_Reset"示例)

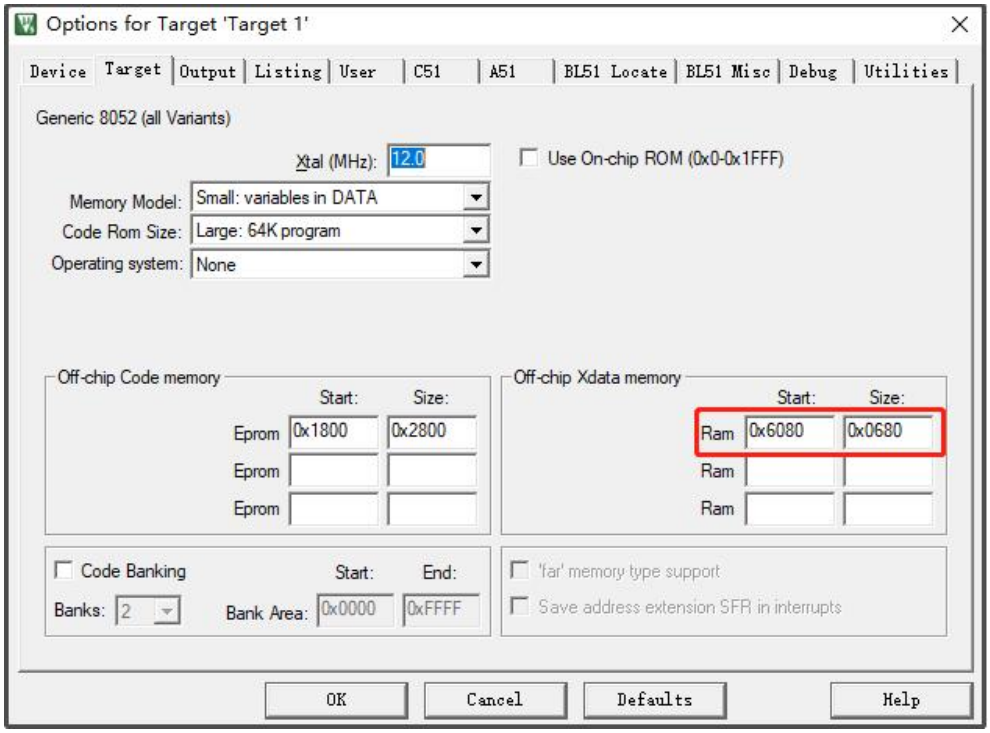
    system_init();

    // !!!注意：如果Loader的大小选择为6K时，则需要将Flash中的中断入口程序复制到RAM中
    // !!!注意：在这个操作需要在使能中断之前使用
    p_data8 = (u8 xdata *)0x6000;
    p_cdata8 = (u8 code *)0x1800;
    for(i=0; i<128; i++) {
        *p_data8 = *p_cdata8;
        p_data8++;
        p_cdata8++;
    }

    // 则将中断的起始位置选择为0x1000的区域
    DPCFG &= 0x3F;
    DPCFG |= 0xC0;
```

建议在函数 system_init()之后马上使用。

另外由于中断入口函数需要放在 SRAM 中，所以 0x6000 的地址需要预留 0x80 的空间。



注意： 以上列出的配置是与"Loader 4K"有差异的地址，其余的配置还是要跟着"9 - TX8C126x 系列 Loader 功能"描述的步骤配置的。"Loader 6K"该配置的地方都得修改。

5. TX8W701x-FAQ

5.1. TX8W701x 系列 RF 模块 TX 发送的数据包错误率高问题

5.1.1. 问题现象

在数据的发送阶段，给 CE 信号的过程中使用延时函数导致的数据发送错包率高现象。

5.1.2. 解决方案

在给 CE 信号的时候，在 CE_HIGH 的情况下，如果不往 TX FIFO 写数据，TX 也会发射出去，发送数据是随机的，而且此时的 TX FIFO 无法清除。所以在接收转发送后，需要在 10us 内往 TX FIFO 内写 payload 或者将 CE 重新拉低。

为了解决这个问题，可以通过等待数据发送完成时产生的 Pending 位，来让数据的发送更合理、更高效。在实际的测试中发现，等待 Pending 的方式在数据的发送过程中，发送数据包的准确率在屏蔽房几乎可以达到 99.9%，外部环境中的数据准确率也可达 98%以上。

添加 while 等待标志位的方法，实现发送的包不会出现错包，如图所示。

```
void ble_tx(u8 cnt)
{
    u8 i;
    rf_spi_write_byte(REG_BANK0_CONFIG, 0xfa); // PTX
    rf_spi_write_byte(REG_BANK0_BLE_CTL, 0x03); // adv mode
    tx_payload[5] = crc8_payload(tx_payload+2, 3);
//  while(1) {
        // 发送端：只有在有数据需要发送时才给CE信号，其他时间CE都是拉低的；
        // 如果需要切换到接收的话，那么在接收状态下CE要一直为高
        for(i=0; i<cnt; i++) {
            rf_spi_write_buf(REG_BANK0_TX_HEADER, tx_payload, 2);
            rf_send_pack(CMD_W_TX_PAYLOAD, tx_payload+2, sizeof(tx_payload)-2);
            rf_spi_write_cmd(CMD_NOP);
            rf_ce_high();
            while(!(rf_spi_read_byte(REG_BANK0_STATUS_EXT) & TX_DS));
            rf_ce_low();
            rf_flush_tx();
            rf_clear_all_irq();
        }
```

这样的发包方式，是适合发包的方法，同时也是稳定的发包方法，包的具体数据如图所示。下图的时间说明的是发送数据包的间隔时间，而不是数据包的发送时间。

P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI	FCS
	+697				Type	TxAdd	RxAdd	PDU-Length		12 34		(dBm)	
65	=51108	0x25	0x8E89BED6	ADV_IND	0	0	0	10	0x112233445566	01 DE	0x30E0BF	-45	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI	FCS
	+696				Type	TxAdd	RxAdd	PDU-Length		12 34		(dBm)	
66	=51804	0x25	0x8E89BED6	ADV_IND	0	0	0	10	0x112233445566	01 DE	0x30E0BF	-45	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI	FCS
	+695				Type	TxAdd	RxAdd	PDU-Length		12 34		(dBm)	
67	=52499	0x25	0x8E89BED6	ADV_IND	0	0	0	10	0x112233445566	01 DE	0x30E0BF	-45	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI	FCS
	+873				Type	TxAdd	RxAdd	PDU-Length		12 34		(dBm)	
69	=53890	0x25	0x8E89BED6	ADV_IND	0	0	0	10	0x112233445566	01 DE	0x30E0BF	-45	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI	FCS
	+697				Type	TxAdd	RxAdd	PDU-Length		12 34		(dBm)	
70	=54587	0x25	0x8E89BED6	ADV_IND	0	0	0	10	0x112233445566	01 DE	0x30E0BF	-45	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI	FCS
	+695				Type	TxAdd	RxAdd	PDU-Length		12 34		(dBm)	
71	=55282	0x25	0x8E89BED6	ADV_IND	0	0	0	10	0x112233445566	01 DE	0x30E0BF	-45	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI	FCS
	+635				Type	TxAdd	RxAdd	PDU-Length		12 34		(dBm)	
73	=57189	0x25	0x8E89BED6	ADV_IND	0	0	0	10	0x112233445566	01 DE	0x30E0BF	-45	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI	FCS
	+697				Type	TxAdd	RxAdd	PDU-Length		12 34		(dBm)	
74	=57886	0x25	0x8E89BED6	ADV_IND	0	0	0	10	0x112233445566	01 DE	0x30E0BF	-45	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI	FCS
	+696				Type	TxAdd	RxAdd	PDU-Length		12 34		(dBm)	
75	=58582	0x25	0x8E89BED6	ADV_IND	0	0	0	10	0x112233445566	01 DE	0x30E0BF	-45	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI	FCS
	+695				Type	TxAdd	RxAdd	PDU-Length		12 34		(dBm)	
76	=59277	0x25	0x8E89BED6	ADV_IND	0	0	0	10	0x112233445566	01 DE	0x30E0BF	-45	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI	FCS
	+696				Type	TxAdd	RxAdd	PDU-Length		12 34		(dBm)	
77	=59973	0x25	0x8E89BED6	ADV_IND	0	0	0	10	0x112233445566	01 DE	0x30E0BF	-45	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI	FCS
	+870				Type	TxAdd	RxAdd	PDU-Length		12 34		(dBm)	
79	=61364	0x25	0x8E89BED6	ADV_IND	0	0	0	10	0x112233445566	01 DE	0x30E0BF	-45	OK
P.nbr.	Time (us)	Channel	Access Address	Adv PDU Type	Adv PDU Header				AdvA	AdvData	CRC	RSSI	FCS
	+697				Type	TxAdd	RxAdd	PDU-Length		12 34		(dBm)	
80	=62061	0x25	0x8E89BED6	ADV_IND	0	0	0	10	0x112233445566	01 DE	0x30E0BF	-45	OK

5.2. TX8W701x 系列 RF 模块 TRX 发送接收过程中数据包未发送或接收不到数据问题

5.2.1. 问题现象

在使用 RF 模块时 TRX 发送接收切换过程中，出现发送一次数据后就无法发送数据了，或者在接收一次数据后就接收不到数据。

5.2.2. 解决方案

这类问题的出现很大可能就是 CE 拉高拉低的问题导致的。在发送端切换至接收端的过程中，通常如果忘记将 CE 拉高，就会出现接收不到数据的现象。这其实并非接收不到数据，而是在 CE 拉高的时候，才是数据的接收阶段。

而数据的接收端再切换为发送端的时候，需要将 CE 拉低，才能更改寄存器由接收端切换为发送端，然后发送数据。

5.3. TX8W701x 系列 RF 模块 TRX 发送接收过程中数据包某 1byte 数据错误问题

5.3.1. 问题现象

在数据的发送过程中，快速移动时发送数据可能会导致发送的数据出现错误，再加上外部环境的干扰，导致发送的数据包被接收时存在错包现象。

5.3.2. 解决方案

上述问题在实际的应用中几乎无法避免。所以在数据的发送端和接收端通过采用 CRC8 校验来保证数据发送和接收的准确性。

5.4. TX8W701x 系列 RF 模块 RX 接收过程中添加异常处理

5.4.1. 问题现象

接收端在很长时间处于接收状态下，出现了无法接收到发送端数据现象。

5.4.2. 解决方案

通过加上异常处理，来让 RF 模块一段固定接收时间复位，短时间未接收到数据的情况下，进行一次睡眠唤醒操作，一段时间未收到数据也进行一次初始化复位。这样可以保证接收端始终可以保持高灵敏的接收状态。