1386 基于涂鸦平台 OTA 升级

一、TM52eF1386介绍

TM52eF1386 有 64K 的 Flash，4K Byte 的 INFO 区以及 4K 的 SRAM。Flash 可分为 128 页，每页 0.5K;INFO 区分为 8 页，每页 0.5K(其中 INFO1~4 为系统保留，INFO5~8 可供用户擦除和写入)。

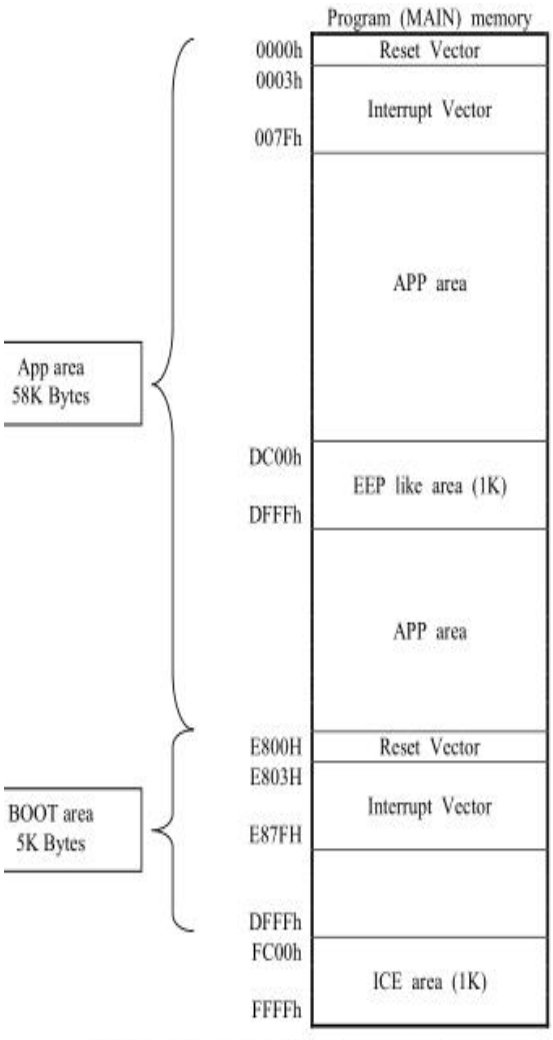
该芯片为了能更好的实现 OTA 升级，再设计之初将 Flash 存储器分为 BOOT 区与 APP 区，BOOT 区存放被保护的更新程序代码，APP 区则存放可被改写的程序代码。操作 SFR RSTV 寄存器设置复位后的复位/中断向量（软件复位、WDT 复位或引脚复位）。通过配置 CFGWH[1:0]寄存器(在 keil 里配置)设置是否将 flash 划分为整个 APP 区还是 BOOT 与 APP 区并且 BOOT 支持三种配置 0K、5K 与 7K。

CFGWH[1:0]=00b, 未设置 BOOT 区域，则上电复位/中断向量为 0000h。

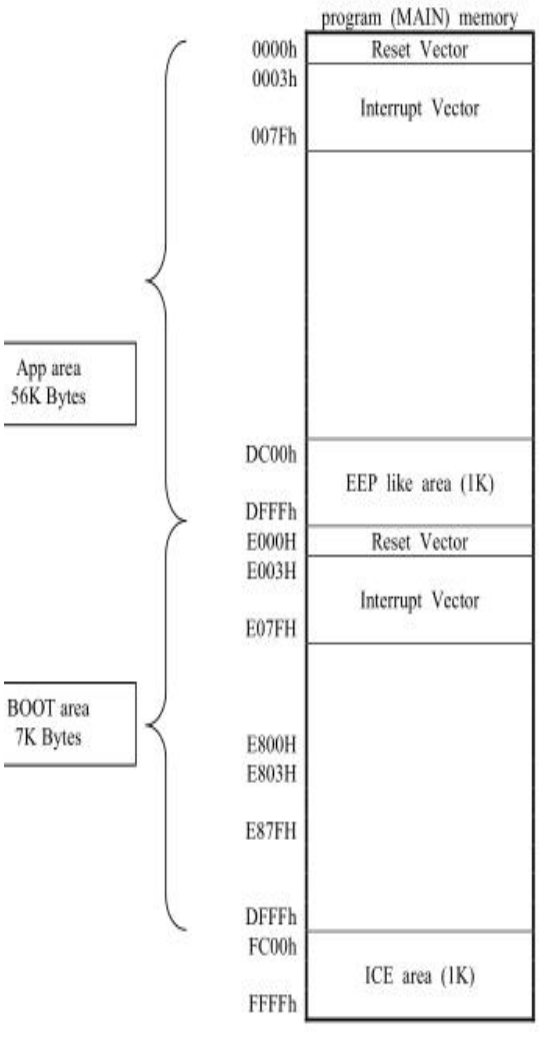
CFGWH[1:0]=10b, 设置 7K BOOT 区域，则上电复位/中断向量为 E000h。

CFGWH[1:0]=11b, 设置 5K BOOT 区域，则上电复位/中断向量为 E800h。

BOOT 区与 APP 区的中断入口位置是独立，所以 BOOT 与 APP 工程配置是不一样的（参考下面的 keil 工程配置章节）。



64K Bytes Program (MAIN) memory with 5K boot area



64K Bytes Program (MAIN) memory with 7K boot area

写/擦除范围	類 EEP 區(MAIN)		APP 區 (when Boot area 7K)		APP 區 (when Boot area 5K)		信息存储器 INFO 5~8	
	写	擦除	写	擦除	写	擦除	写	擦除
	DC00h-DFFFh		0000h-DBFFh		0000h-DBFFh E000h-E7FFh		0800h-0FFFh	
SWCMD 设置	-		65, A7 僅在 Boot 区 E000~FFFFh 時使能生效		65, A7 僅在 Boot 区 E800~FFFFh 時使能生效		-	
IAPWE 设置	E2, 4C	E2, BA	E2, 4C	E2, BA	E2, 4C	E2, BA	A1, 4C	A1, BA

IAP 写/擦除使能条件

芯片支持 IAP 功能，**IAP 写入时（Flash 区、INFO 区、EEprom 区），必须先擦除再写入字节。 擦除后，每个地址只能写入一次。** 只提供页面擦除和字节写入功能。**在写入 IAP 之前，用户应先关闭 LVR。**

IAP 写入通过“MOVX @DPTR, A”指令即可实现。写入时，**VCC 电压必须 > 2.5V**。

当 IAP 写/擦除时，SFR IAPWE/SWCMD 需要遵循以下设置
SFR IAPWE
写入 E2h 和 4Ch 以启用 MAIN（类 EEP / APP）区域字节写入
写入 E2h 和 BAh 以启用 MAIN（类 EEP / APP）区域页擦除
写入 A1h 和 4Ch 使能 INFO5~8 区域字节写入
写入 A1h 和 BAh 使能 INFO5~8 区域页擦除
详细的操作，可参考例程里 iap.c 文件

特别注意：APP 区的 flash 只能在 BOOT 区才有权限擦除与写入。

二、涂鸦智能

涂鸦智能是物联网开发平台服务商,涂鸦提供能够智连万物的云平台,打造互联互通的开发标准,连接品牌,OEM 厂商,开发者,零售商和各行业的智能化需求。

为了调试方便，涂鸦自己开发了涂鸦模组调试助手，涂鸦模组调试助手是一个集成了云模组通讯协议的串口调试工具，常用于 MCU 代码开发方案的开发调试。模组调试助手集成了包括 Wi-Fi、蓝牙、Zigbee、NB-IoT 等云模组常用串口协议，既可以模拟模组验证 MCU 代码逻辑，也可以模拟 MCU 调试配网功能。

MCU 模拟模式

模组调试助手模拟 MCU，电脑通过串口工具接涂鸦模组串口，模组上电后会发起初始化数据交互，助手会模拟 MCU 自动回复正确数据。此模式可以使用涂鸦智能生活 App 给模组直接配网，查看正确的数据上报下发格式。拿到模组后，可以：

1、使用此模式体验全部收据交互流程，提高开发效率。

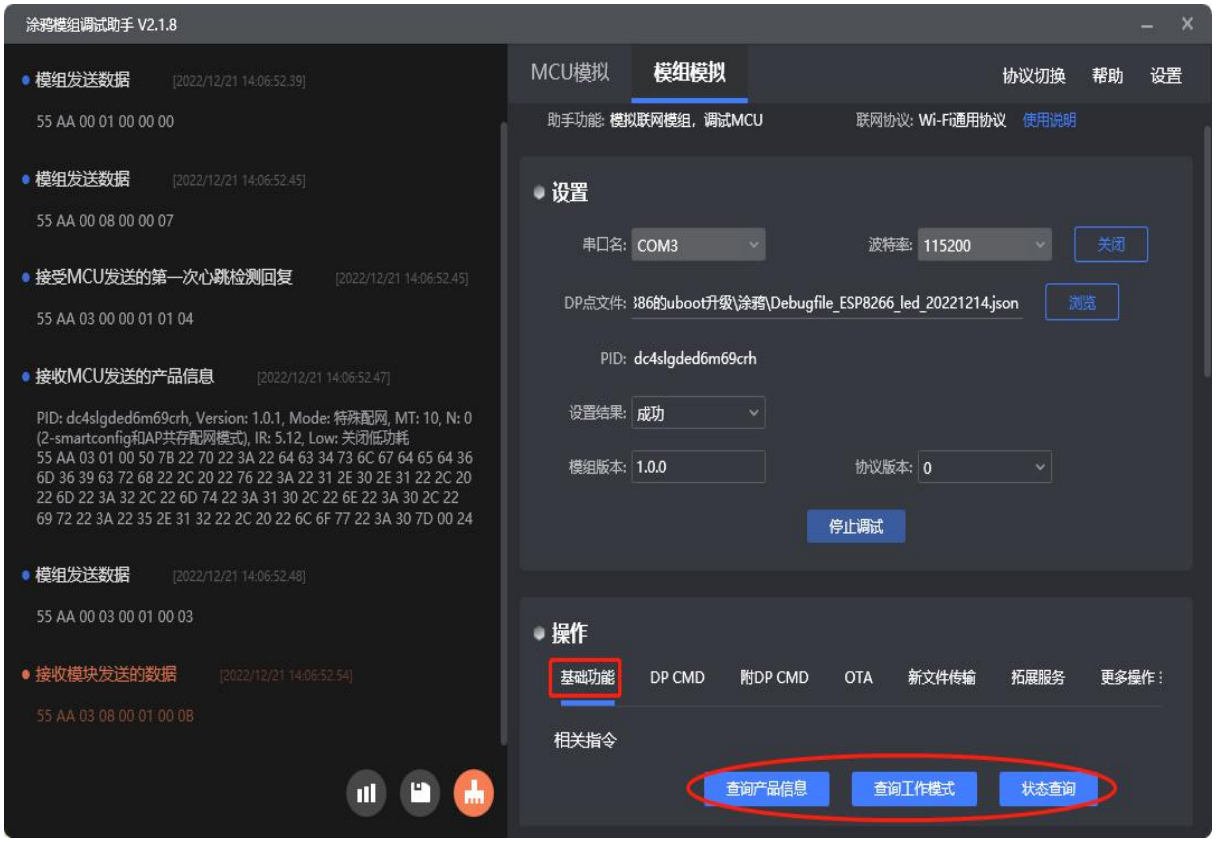
2、调试过程遇到问题，也可通过此模式对比验证，确定问题归属。

模组模拟模式

模组调试助手模拟涂鸦模组，电脑通过串口工具接入 MCU 串口，助手自动发起初始化数据交互，检测 MCU 回复数据是否正确。此模式通常用于代码移植完毕后测试 MCU 协议代码是否正确，助手对于错误数据会有对应的提示，接实际模组前先通过此模式校验，可减少后期出现问题概率。

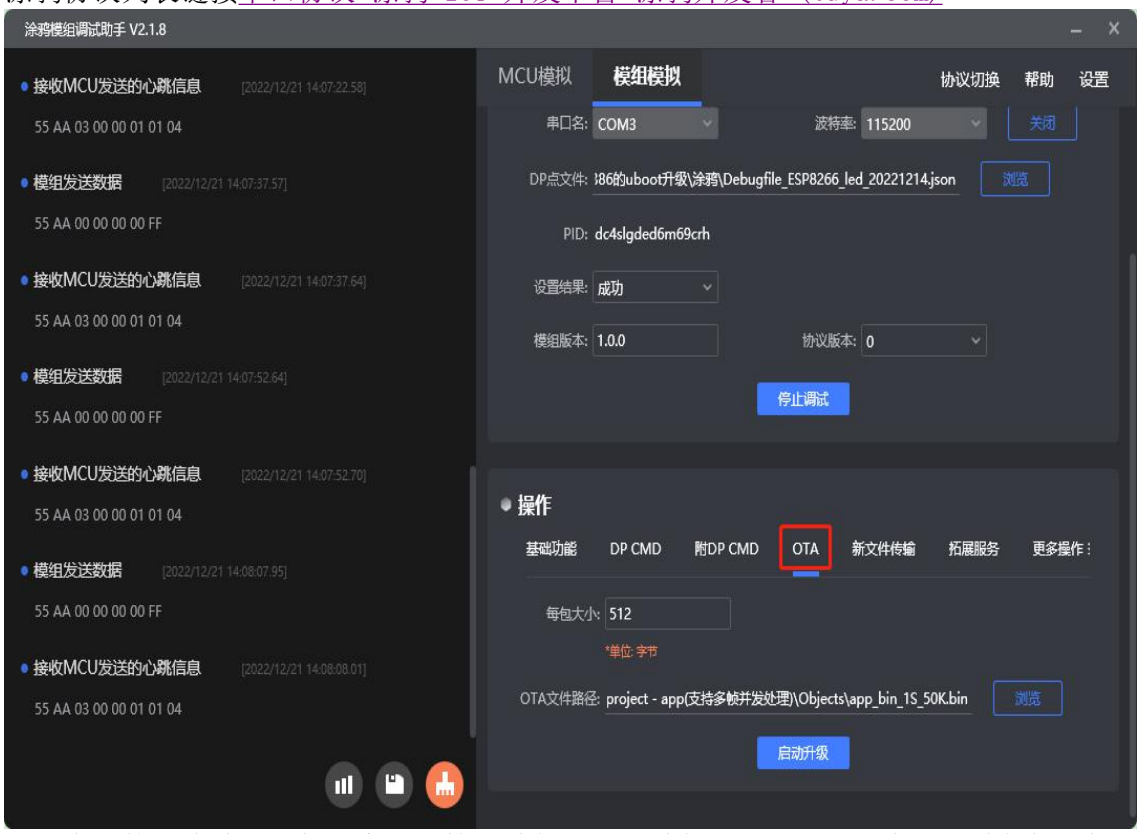
该模式没有联网功能，只能用来验证 MCU 串口协议收发正确性。

详细的使用方法，参考该链接[模组调试助手-涂鸦 IoT 开发平台-涂鸦开发者(tuya.com)]



本次我们使用的是模组模拟模式（调试助手充当 wifi 模块的角色），基于 wifi 通用协议去开发，使用的是 uart 通信。使用涂鸦助手，需要一个 json 文件才能正常使用，该文件需要在涂鸦平台搭建一个产品信息便会生成（本例程先提供自己的 json 文件，方便

客户验证一些基础功能指令）。
基础功能指令：
1、心跳包
2、查询产品信息
3、查询工作模式
4、状态查询
5、...
涂鸦协议列表链接串口协议-涂鸦 IoT 开发平台-涂鸦开发者（tuya.com）



本次使用涂鸦助手想要实现的核心功能是 OTA 功能，按照上图选中 OTA 功能框，加载需要 OTA 的文件(我们使用的是 bin 文件),最后点击启动升级按键触发升级功能。

点击启动升级，涂鸦会先发送升级启动指令（命令字 0x0A）,用于询问 MCU 每包帧的大小，之后便会发送 OTA 文件。

升级启动（升级包大小通知）

升级启动方式含自动及手动升级。当处于自动升级时，模组检测云端 MCU 有更新版本固件，则自动启动与 MCU 升级包交互流程。当处于手动升级时，通过 App 确定，模组才启动与 MCU 升级包交互流程

模组发送

字段	字节数	说明
帧头	2	0x55aa
版本	1	0x00
命令字	1	0x0a
数据长度	2	0x0004
数据	4	固件包字节数，unsigned int，大端
校验和	1	从帧头开始，按字节求和，得出的结果对 256 求余

示例： `55 aa 00 0a 00 04 00 00 68 00 75`

表示固件包长度 26624，即 26KB。

MCU 返回

字段	字节数	说明
帧头	2	0x55aa
版本	1	0x03
命令字	1	0x0a
数据长度	2	0x0001
数据	1	升级包分包传输大小： • 0x00：默认 256byte（兼容旧固件） • 0x01：512byte • 0x02：1024byte
校验和	1	从帧头开始，按字节求和，得出的结果对 256 求余

示例： `55 aa 03 0a 00 01 00 0d`

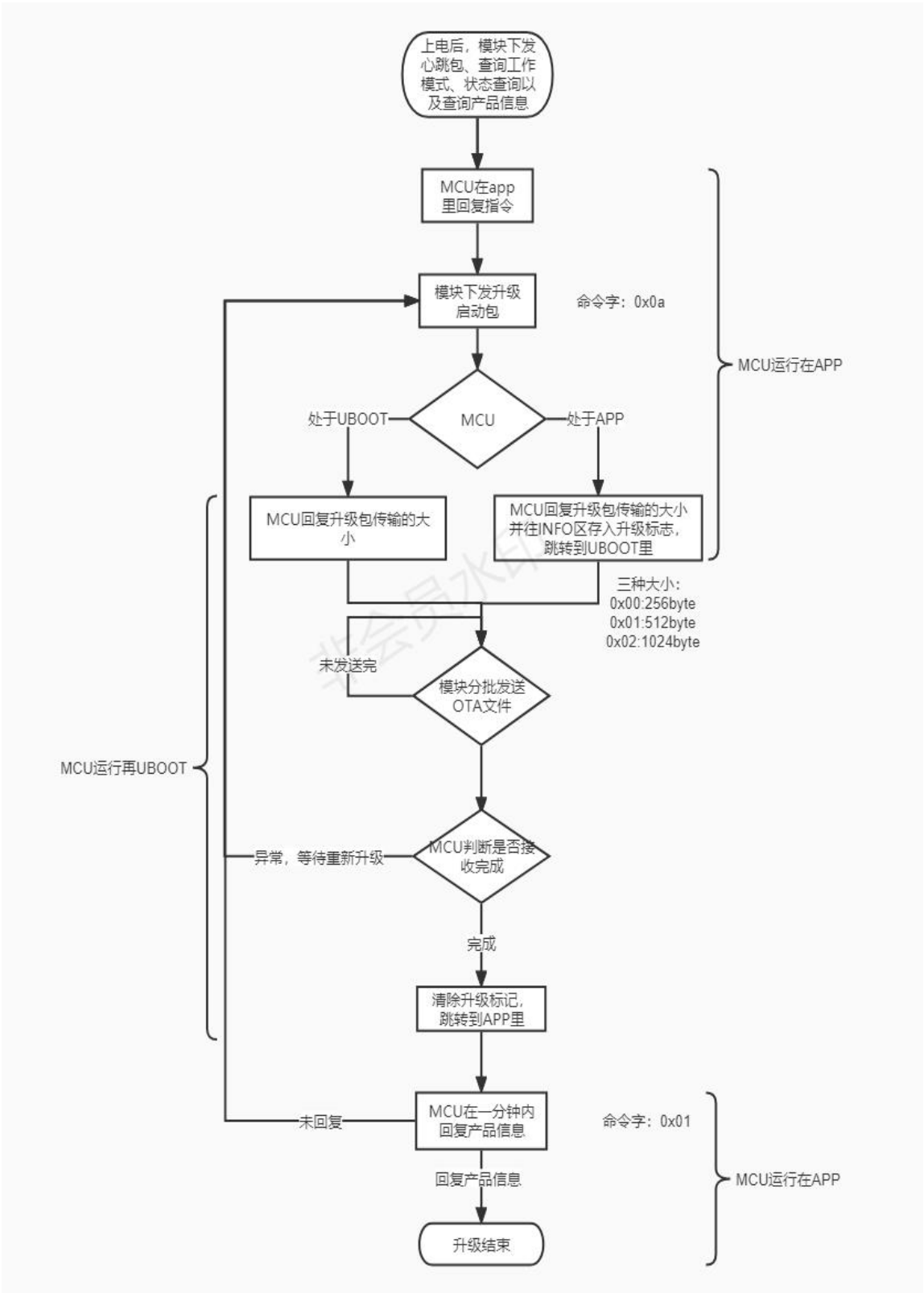
字段	字节数	说明
帧头	2	0x55aa
版本	1	0x00
命令字	1	0x0b
数据长度	2	0x0004+N
数据	4+N	• data[0]–data[3]：固定为包偏移 • data[4]–data[n]：数据包内容
校验和	1	从帧头开始，按字节求和，得出的结果对 256 求余

示例:

若要升级的文件大小 530Byte，(最后一包数据可不回复)

- 第一包数据，包偏移为 0x00000000，数据包长度为 256
  `55 aa 00 0b 01 04 00000000 xx…xx XX`
- 第二包数据，包偏移为 0x00000100，数据包长度为 256
  `55 aa 00 0b 01 04 00000100 xx…xx XX`
- 倒数第 2 包数据，包偏移为 0x00000200，数据包长度为 18
  `55 aa 00 0b 00 16 00000200 xx…xx XX`
- 最后一包，包偏移为 0x00000212，数据包长度为 0
  `55 aa 00 0b 00 04 00000212 xx...xx XX`

三、OTA流程图


上电后，模块下发心跳包、查询工作模式、状态查询以及查询产品信息
MCU在app里回复指令
模块下发升级启动包
命令字：0x0a
MCU运行在APP
MCU
处于UBOOT
处于APP
MCU回复升级包传输的大小
MCU回复升级包传输的大小并往INFO区存入升级标志，跳转到UBOOT里
三种大小：
0x00:256byte
0x01:512byte
0x02:1024byte
未发送完
模块分批发送OTA文件
MCU运行再UBOOT
异常，等待重新升级
MCU判断是否接收完成
完成
清除升级标记，跳转到APP里
未回复
MCU在一分钟内回复产品信息
命令字：0x01
回复产品信息
MCU运行在APP
升级结束

四、Uboot与app例程

uboot 上电，先读取存储在 INFO5 区域里面的标识，用于判断更新 APP 程序还是跳转到 APP。INFO 区域 5~8 都可以使用，目前例程使用的是 INFO5(标识是 APP 工程里判断并写入)。采用三个字节作为标识，如果是 0x55、0xaa、0x5a 表示更新 APP，其他表示跳转到 APP。

app 主要是运行客户自己程序与检测是否要升级。如果要升级将 0x55、0xaa、0x5a 写入到 INFO5 区域，设好 BOOTV 再热复位就可以切换到 uboot 里。

bit user_write_INFO_byte(unsigned int addr, unsigned char *buf, unsigned int len)

INFO 写函数，函数自带写入与读回对比校验，带返回值（0 ：正常 1： 异常）

user_read_INFO_byte(unsigned int addr, unsigned char *buf, unsigned int len)

INFO 读函数

bit user_write_APP_byte(unsigned int addr, unsigned char *buf, unsigned int len)

APP 写函数，函数自带写入与读回对比校验（CRC 校验），带返回值（0 ：正常 1： 异常）。函数内添加了页擦除动作，通过判断地址是不是页的首地址进行擦除。建议每次存储的长度 256 字节或者 512 字节，最大长度不能超过 512 字节

Uboot 与 APP 跳转函数

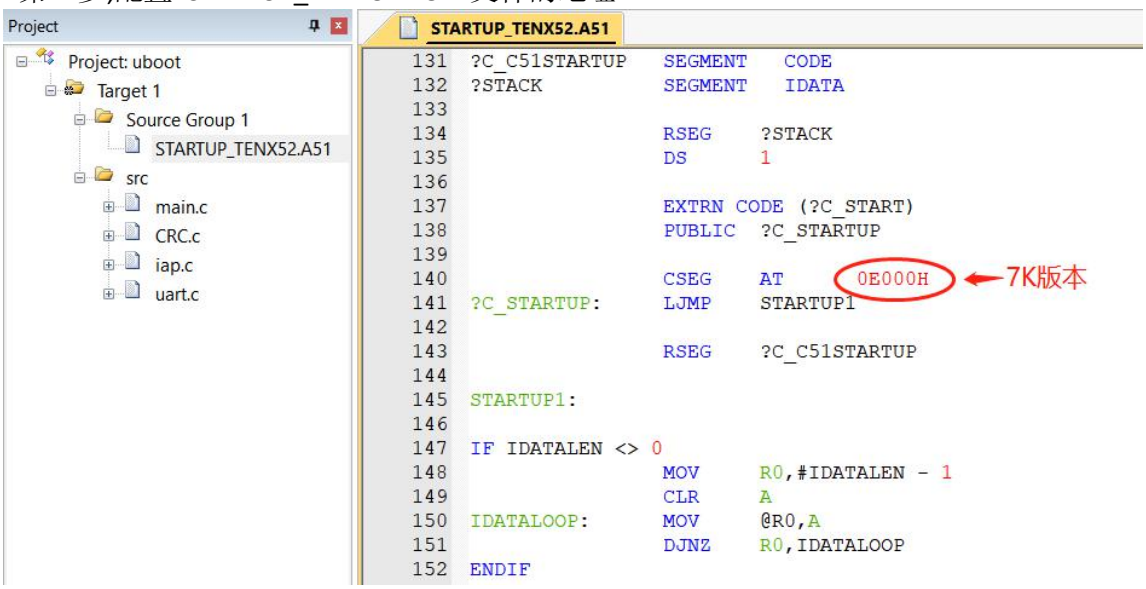
```
EA = 0;         //关闭中断
BOOTV = 0x04;   //设置热复位后跳转到uboot
SWCMD = 0x56;   //软件复位
```

```
EA = 0;         //关闭中断
BOOTV = 0x00;   //设置热复位后跳转到app
SWCMD = 0x56;   //软件复位
```

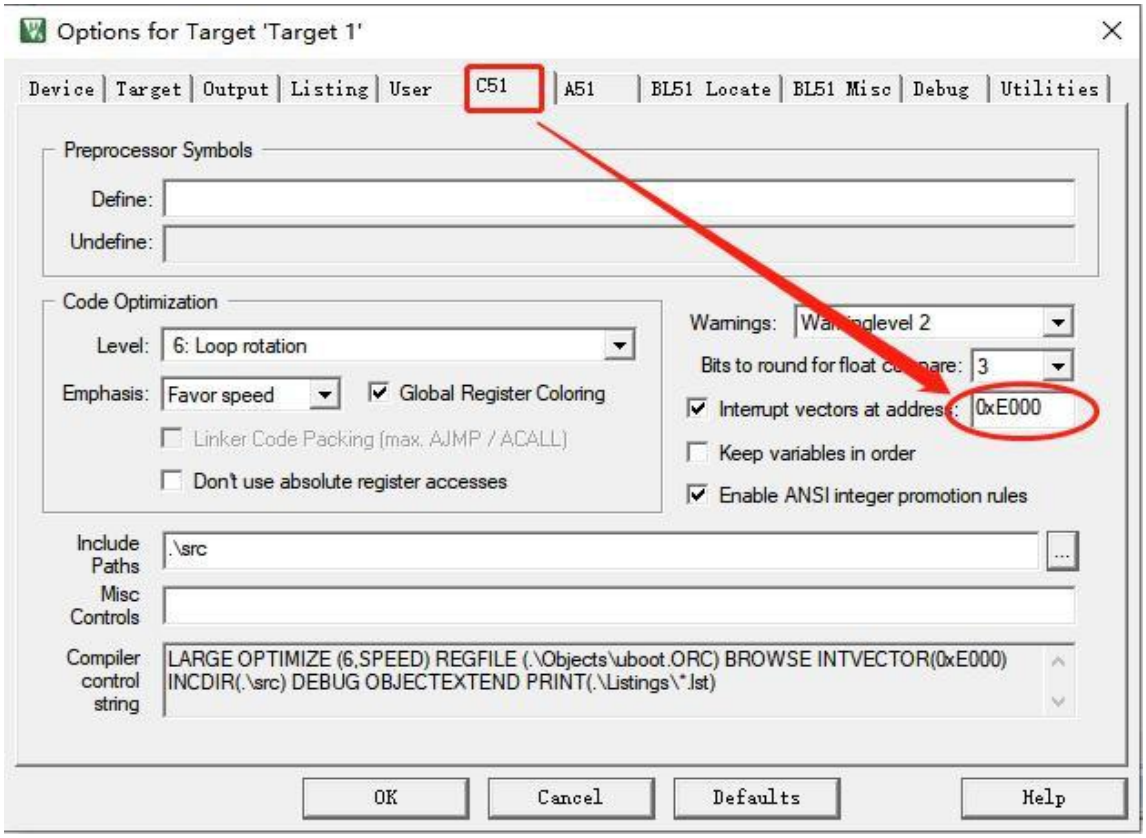
五、Keil工程配置

1、uboot 工程配置（7K 版本）

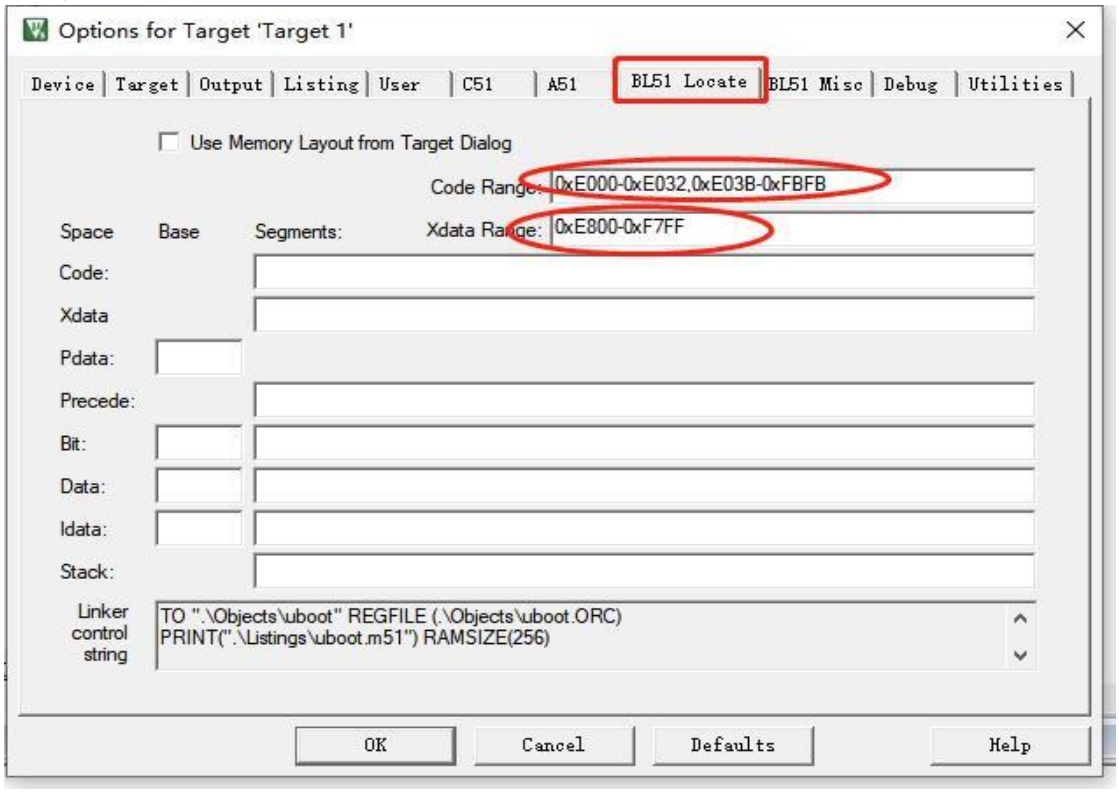
第一步,配置 STARTUP_TENX52.A51 文件的地址



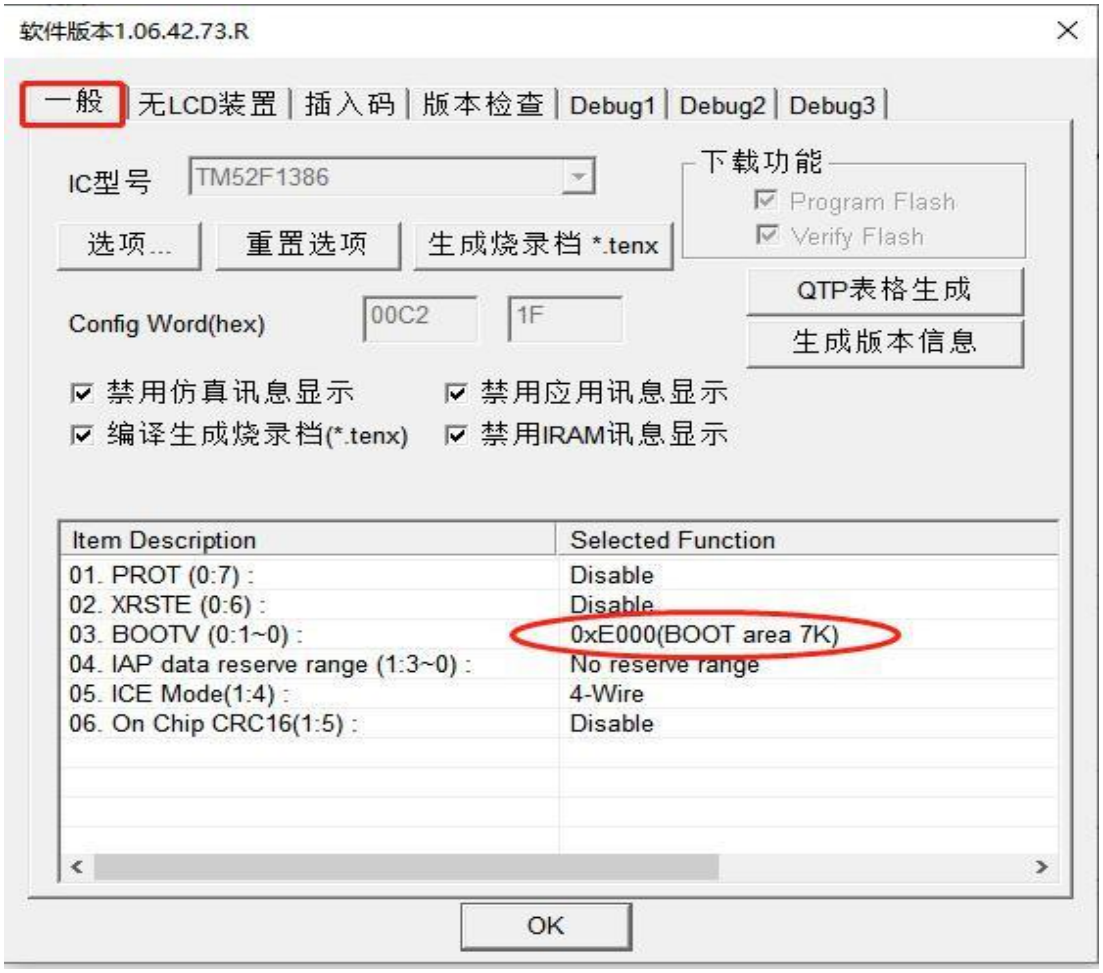
第二步，配置中断入口地址



第三步，配置 code 范围与 xdata 的范围

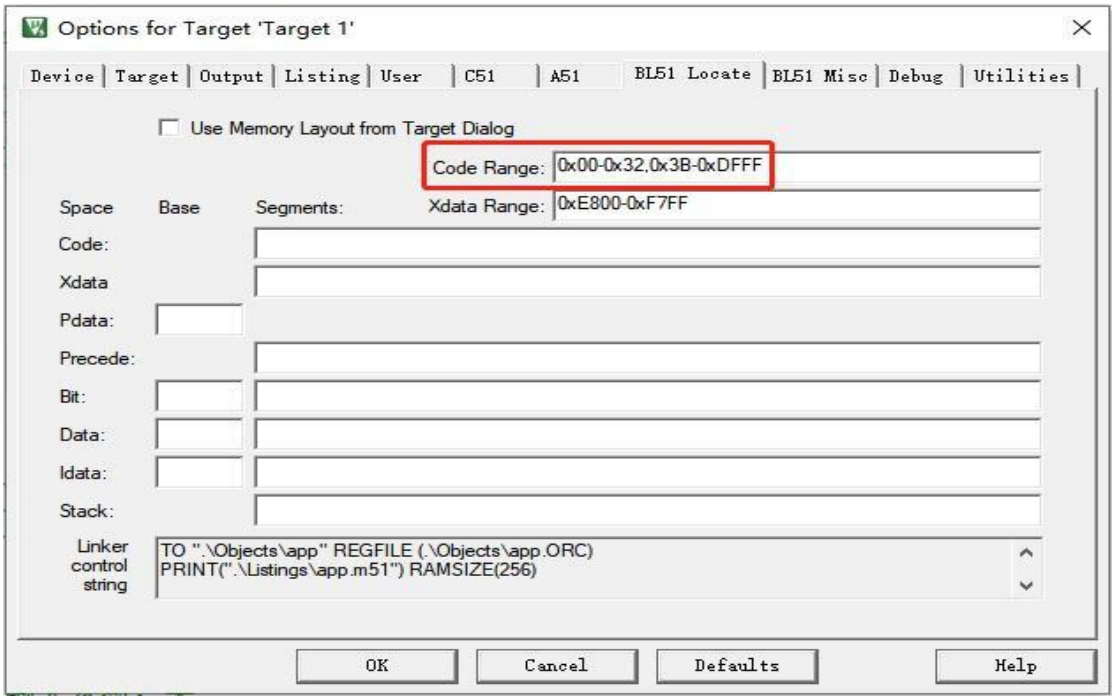


第四步，配置 uboot 为 7K



2、APP 工程配置（7K 版本）

第一步,配置 code 范围与 xdata 的范围



第二步，配置程序为APP

六、注意事项

1、本例程是 uboot 7K 例程，信息存储器 INFO5 本例程用于 uboot 与 app 信息交互
2、APP 的 flash 区必须在 uboot 里才有权限擦除与写入
3、IAP 写入时（Flash 区、INFO 区、EEprom 区），必须先擦除再写入字节。 擦除后，每个地址只能写入一次。在写入 IAP 之前，必须先关闭 LVR 且 VCC 电压吗> 2.5V。
4、如果要使能读保护，只能在 uboot 里开启有效，app 里开与关无效。