文档编号：AN_100

上海东软载波微电子有限公司

应用笔记

分区加密下的仿真

版权所有©上海东软载波微电子有限公司

修订历史

版本	修订日期	修改概要
V1.0	2018-03-06	初版

地　址：中国上海市龙漕路 299 号天华信息科技园 2A 楼 5 层
邮　编：200235
E-mail：support@essemi.com
电　话：+86-21-60910333
传　真：+86-21-60914991
网　址：http://www.essemi.com/

版权所有©
上海东软载波微电子有限公司
本资料内容为上海东软载波微电子有限公司在现有数据资料基础上慎重且力求准确无误编制而成，本资料中所记载的实例以正确的使用方法和标准操作为前提，使用方在应用该等实例时请充分考虑外部诸条件，上海东软载波微电子有限公司不担保或确认该等实例在使用方的适用性、适当性或完整性，上海东软载波微电子有限公司亦不对使用方因使用本资料所有内容而可能或已经带来的风险或后果承担任何法律责任。基于使本资料的内容更加完善等原因，上海东软载波微电子有限公司保留未经预告的修改权。使用方如需获得最新的产品信息，请随时用上述联系方式与上海东软载波微电子有限公司联系。

目 录

内容目录

第 1 章 概述 4
1.1 应用场景及适用芯片 4
1.2 与调用 LIB 库的区别 4
第 2 章 操作说明 5
2.1 示例工程功能说明 5
2.2 程序烧录方法 5
2.3 Lib 编程注意事项 7
2.4 User 编程注意事项 9

版权所有©上海东软载波微电子有限公司 http://www.essemi.com

第1章 概述

1.1 应用场景及适用芯片

该功能的一个典型应用场景即提供客户二次开发，客户可以事先在芯片的指定地址烧入一段代码，该代码用于另一个客户进一步的应用开发，做仿真调试时，用户工程不需要包含库一块编译烧录，只需要运行自己的应用代码即可。

该方案只适用于含有分区加密功能的芯片。

1.2 与调用LIB库的区别

-调用 LIB 库时，工程中需要包含 LIB 库文件，仿真时需要一起编译烧录。

-分区加密仿真的工程只需要包含一个 api 接口的文件，独立编译烧录自己的应用程序。

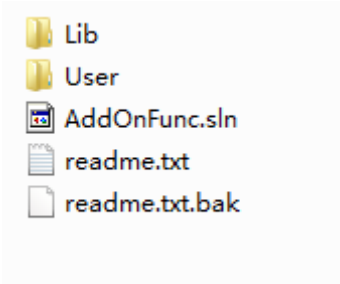
版权所有©上海东软载波微电子有限公司 http://www.essemi.com

第2章 操作说明

2.1 示例工程功能说明

以下说明将结合示例项目《二次开发 Demo》一起讲解，该工程基于 ES7P2023 开发。

该项目包含两个工程，一个 User 程序及一个 Lib 程序



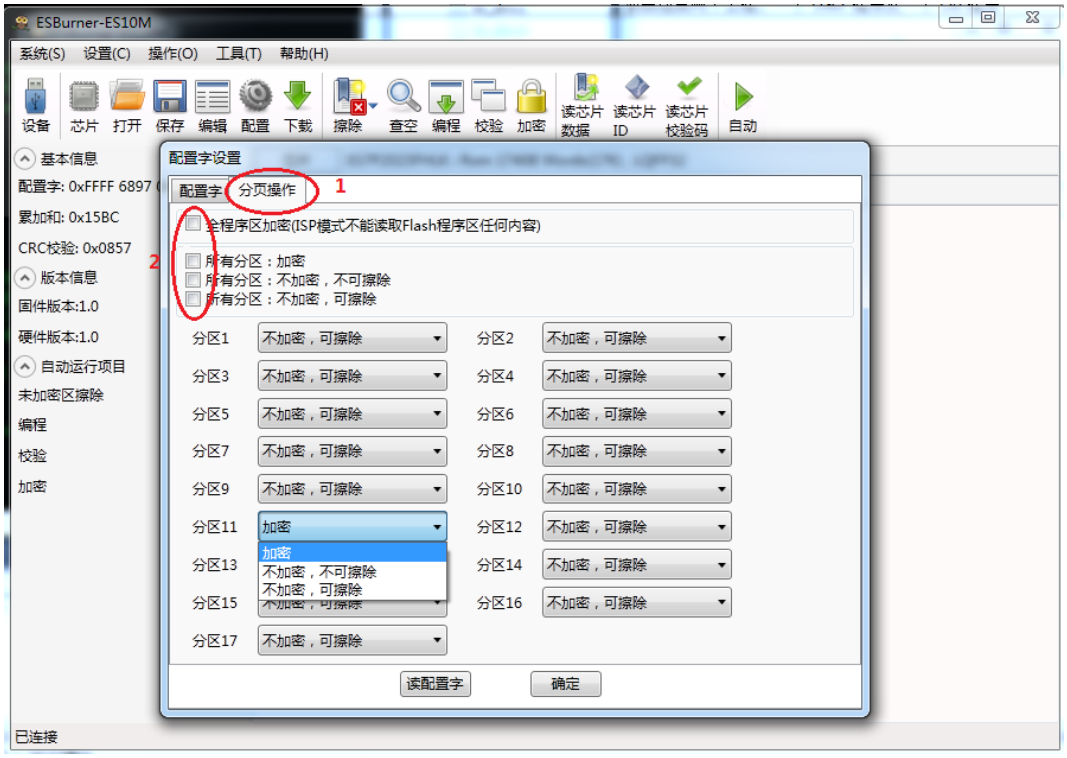
Lib 程序用于提前烧入芯片，之后 User 程序会通过跳转的方法调用 Lib 程序中的函数。

在 Lib 程序中以加法 void Add()函数和减法 void Sub()函数作为范例，演示编程方法。

通过在 User 程序中调用这两个函数，观察变量的变化，来评估调用的效果。

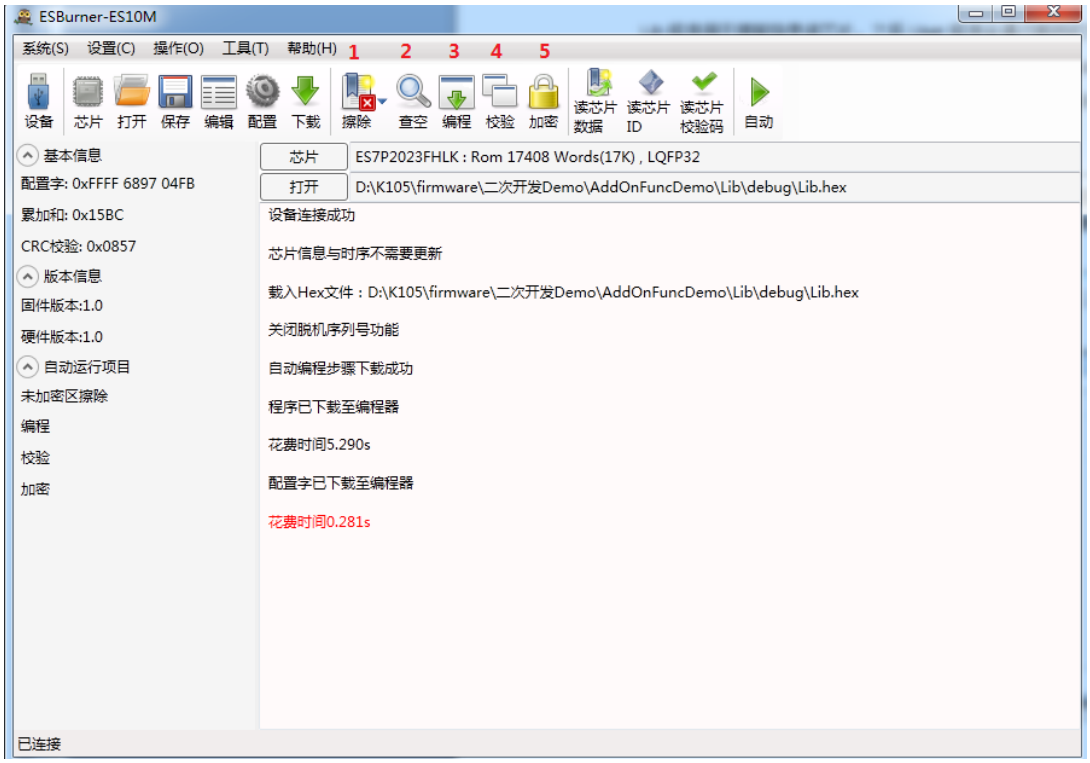
2.2 程序烧录方法

该示例中，Lib 通过“ES10M 编程工具”进行烧录。在“配置字设置”->“分页操作”界面，将所有勾选去掉，在本例中，Lib 工程起始地址为 0x2800，因此，设计为将分区 11 和分区 12 选择“加密”属性。用户根据 Lib 程序最终编译结果，判断需要加密的 ROM 分区。

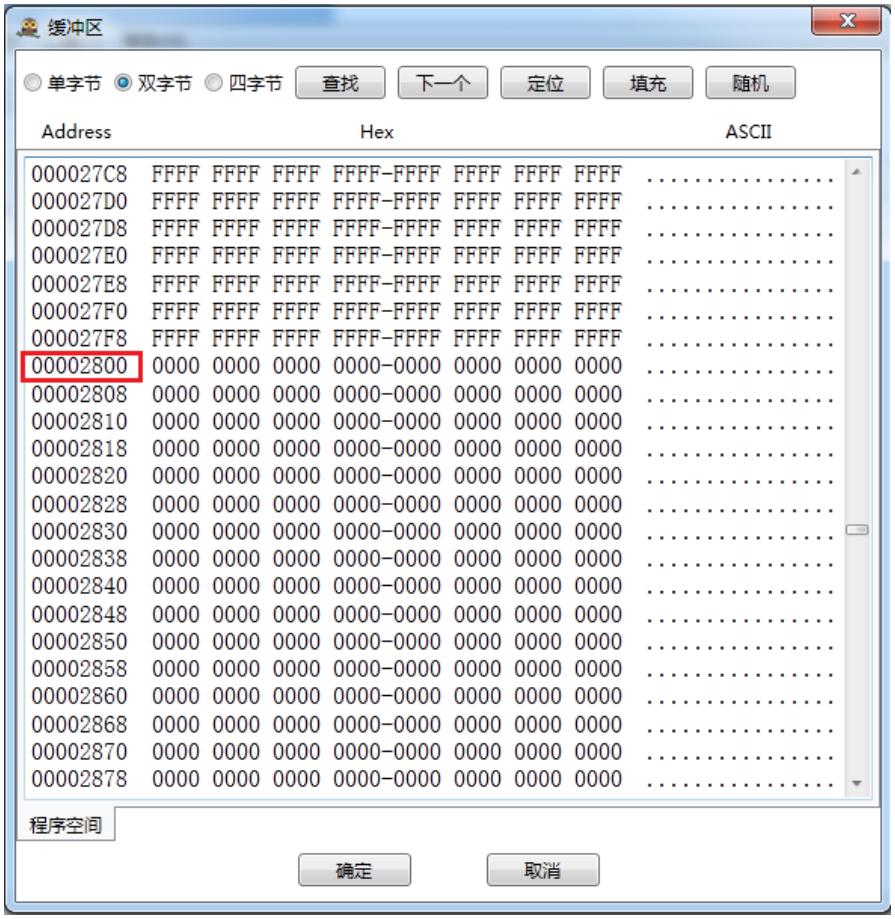


之后按照下图数字所标，逐个点击按钮，完成烧录。

版权所有©上海东软载波微电子有限公司 http://www.essemi.com



完成加密后，再次读取芯片数据，所有加密区域均显示 0。



到这步为止，所有操作结束，之后用户即可在主程序中进行仿真编程。

版权所有©上海东软载波微电子有限公司 http://www.essemi.com

2.3 Lib编程注意事项

1）修改 startup.asm 文件，确定库函数的起始地址；

```
EXTERN  _main

APP_ENTRY EQU 0x2800

_STARTUP    CSEG 0X00
    AJMP    APP_ENTRY
    NOP
    NOP
    PUSH
    AJMP    APP_ENTRY + 4

   END
```

2）根据起始地址，修改 xxx.lkr 文件中每个分区的 PROTECTED 属性，在示例工程中，库函数从 0x2800 开始编程，因此从 0x2800 开始的 PROTECTED 属性均需修改为 0；

```
 7   ROM     NAME = SEGMENT0      START = 0000     END = 07FF     PROTECTED = 1
 8   ROM     NAME = SEGMENT1      START = 0800     END = 0FFF     PROTECTED = 1
 9   ROM     NAME = SEGMENT2      START = 1000     END = 17FF     PROTECTED = 1
10   ROM     NAME = SEGMENT3      START = 1800     END = 1FFF     PROTECTED = 1
11   ROM     NAME = SEGMENT4      START = 2000     END = 27FF     PROTECTED = 1
12   ROM     NAME = SEGMENT5      START = 2800     END = 2FFF     PROTECTED = 0
13   ROM     NAME = SEGMENT6      START = 3000     END = 37FF     PROTECTED = 0
14   ROM     NAME = SEGMENT7      START = 3800     END = 3FFF     PROTECTED = 0
15   ROM     NAME = SEGMENT8      START = 4000     END = 43FF     PROTECTED = 0
```

3）在 main.c 文件中，定义了一个数组，用来分配给 User 程序的 RAM，避免编译冲突。在示例工程中，从地址 0x0000 开始，预留 11 个 section 给 User 程序；

```
static volatile unsigned char section0 arrUser[1408]    @ 0x0 ;
```

注：预留给 Lib 的空间要避免与堆栈空间冲突，用户需要通过芯片数据手册，确定 RAM 中预留堆栈空间大小。

4）所有参数在声明时均需要固定地址，可以参见 api.h；

5）库函数提供入口地址，供 User 程序调用，所有 Lib 程序中的函数均由该函数跳转访问；

版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
/*************************************
/所有API函数的入口函数，在User程序
*************************************
void ESLIB(void)  @0x2E00
{
    switch(index)
    {
        case 0:
            __asm
            {
                EXTERN _ADD
                goto _ADD
            }

            break;


        default:break;
    }
}

void ESLIB_INT(void)  @0x2F00
{
    switch(index)
    {

        case 1:
            __asm
            {
                EXTERN _SUB
                goto _SUB
            }

            break;

        default:break;
    }
}
```

6）在编译过程中，不同函数会产生 RAM 空间重叠，因此需要将主程序与中断程序中的函数，包括入口地址函数也需要分别在各自线程中调用，本示例中，ADD()函数设计为主函数需要调用的程序，而 SUB()函数设计为在中断函数中需要调用的程序，因此需要编写为如下图所示，上图分为两个入口地址作跳转也是同理；

版权所有©上海东软载波微电子有限公司 http://www.essemi.com

```
void isr_handler(void) interrupt
{
    ESLIB_INT();
    SUB();
}

void main(void)
{
/**********************************
/为了保证编译时程序不被优化，所有
**********************************/

    ADD();
    ESLIB();
}
```

2.4 User编程注意事项

1）User 程序建议提供一个 api 的头文件给用户，以方便程序的调用。

```
#define ADD(a1,a2)   add1 = a1;                    \
                     add2 = a2;                    \
                     index = 0;                    \
                     ESLIB();                      \



#define SUB(b1,b2)   sub1 = b1;                    \
                     sub2 = b2;                    \
                     index = 1;                    \
                     ESLIB_INT();                  \
```

2）在 main.c 文件中，定义了一个数组，用来分配给 Lib 程序的 RAM，避免编译冲突。

```
static volatile unsigned char section11 arrLib[128] @ 0x0580 ;
```


版权所有©上海东软载波微电子有限公司 http://www.essemi.com