

32 位 微控制器

HC32F460 系列的嵌入式 FLASH

适用对象

系列	产品型号
HC32F460	HC32F460JEU A
	HC32F460JETA
	HC32F460KEUA
	HC32F460KETA
	HC32F460PETB

目 录

1	摘要	3
2	FLASH 简介	3
3	HC32F460 系列的 FLASH	4
3.1	简介.....	4
3.2	说明.....	4
3.2.1	寄存器介绍.....	4
3.2.2	工作流程介绍	5
3.2.3	一次性可编程字节（OTP）	11
3.2.4	引导交换	11
4	样例代码	13
4.1	代码介绍	13
4.2	代码运行	14
5	版本信息 & 联系方式	15

1 摘要

本篇应用笔记主要介绍如何使用 HC32F460 系列芯片的嵌入式 FLASH 读写数据。

2 FLASH 简介

什么是 FLASH?

FLASH 接口通过 FLASH ICODE、DCODE、MCODE 总线对 FLASH 进行访问，该接口可对 FLASH 执行编程、擦除和全擦除操作；通过缓存机制加速代码执行。

FLASH 的重要特征?

FLASH 读、编程、擦除和全擦除操作，支持引导交换，安全保护和数据加密。

3 HC32F460 系列的 FLASH

3.1 简介

FLASH 接口通过 FLASH ICODE、DCODE、MCODE 总线对 FLASH 进行访问，该接口可对 FLASH 执行编程、擦除和全擦除操作；通过指令预取和缓存机制加速代码执行。

3.2 说明

FLASH 读、编程、扇区擦除和全擦除操作。

CODE 总线 16Bytes 预取值，I_CODE 和 D-CODE 总线上共享 64 个缓存（128bit 宽）。

支持 FLASH 低功耗读。

支持引导交换。

支持安全保护及数据加密。

容量为 512Kbytes（其中有 32bytes 为功能保留位），分为 64 个扇区，每个扇区为 8KBytes。

编程单位为 4Bytes，擦除单位为 8KBytes。

128bit 宽数据读取。

OTP（One Time Program）区域共 1020Bytes，分为 960Bytes 数据区，并配有 60Bytes 的锁存区。

3.2.1 寄存器介绍

- 1) EFM_FAPRT：访问 EFM 寄存器保护寄存器。
- 2) EFM_FSTP：FLASH 停止寄存器。
- 3) EFM_FRMC：读模式寄存器。可配置插入等待周期，缓存功能、预取指功能等。
- 4) EFM_FWMC：擦写模式寄存器。配置编程擦除模式。
- 5) EFM_FSR：状态寄存器。查看 FLASH 状态，结束标志、错误标志等。
- 6) EFM_FSCLR：状态清除寄存器。
- 7) EFM_FITE：中断许可寄存器。配置操作结束或错误中断许可。
- 8) EFM_FSWP：引导交换状态寄存器。通过该寄存器可判断程序复位后从扇区 0 还是扇区 1 启动。

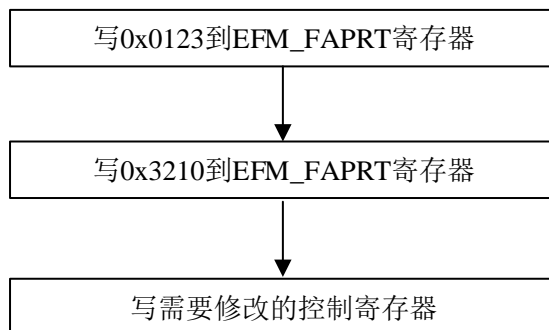
- 9) EFM_FPMTSW: FLASH 窗口保护起始地址寄存器。
- 10) EFM_FPMTEW: FLASH 窗口保护结束地址寄存器。
- 11) EFM_UQID1~3: Unique ID 寄存器。

3.2.2 工作流程介绍

3.2.2.1 寄存器解除保护和写保护

本模块的寄存器受 EFM_FAPRT 寄存器保护，当处于保护状态，屏蔽普通的写操作。

解除保护的步骤如下图：



在解除保护的状态下，对 EFM_FAPRT 寄存器写任意值，EFM 寄存器再次进入保护状态。

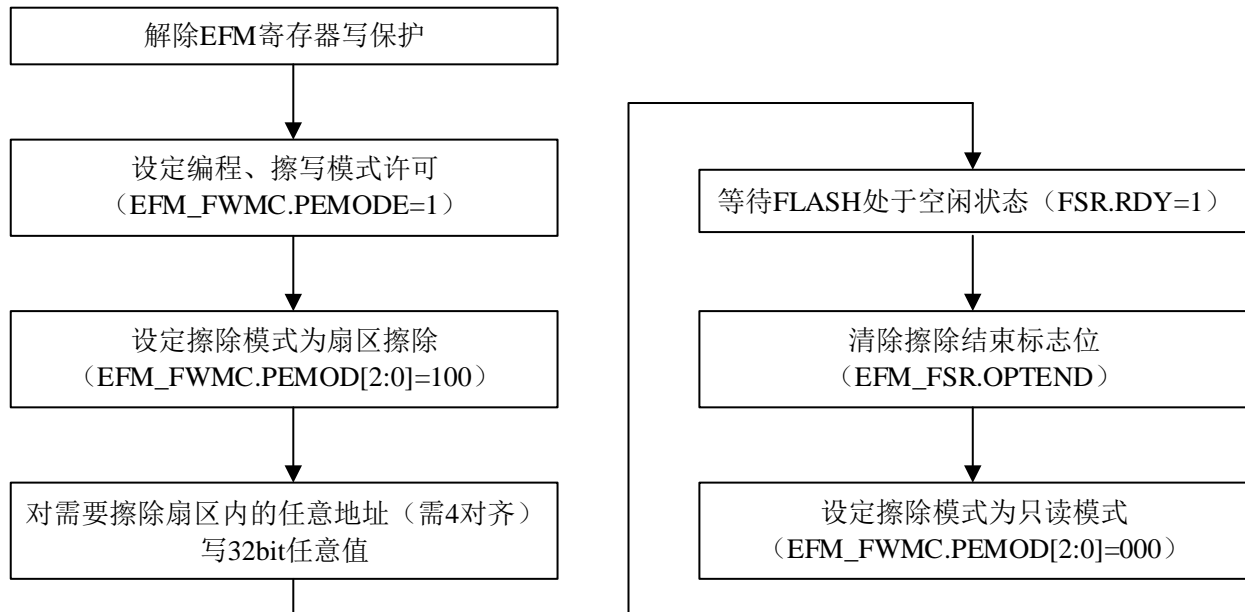
注意：

- 在实际应用中，发现寄存器值未写入成功，应首先检查 EFM 寄存器访问保护是否有效，保护有效时，EFM_FAPRT 寄存器值读出的为 0x00000000。

3.2.2.2 扇区擦除

EFM 提供扇区擦除和全擦除两种擦除方式。对 FLASH 进行扇区擦除操作后，该扇区内地址（8Kbytes 空间）数据刷新为全 1。

扇区擦除的设定步骤如下：



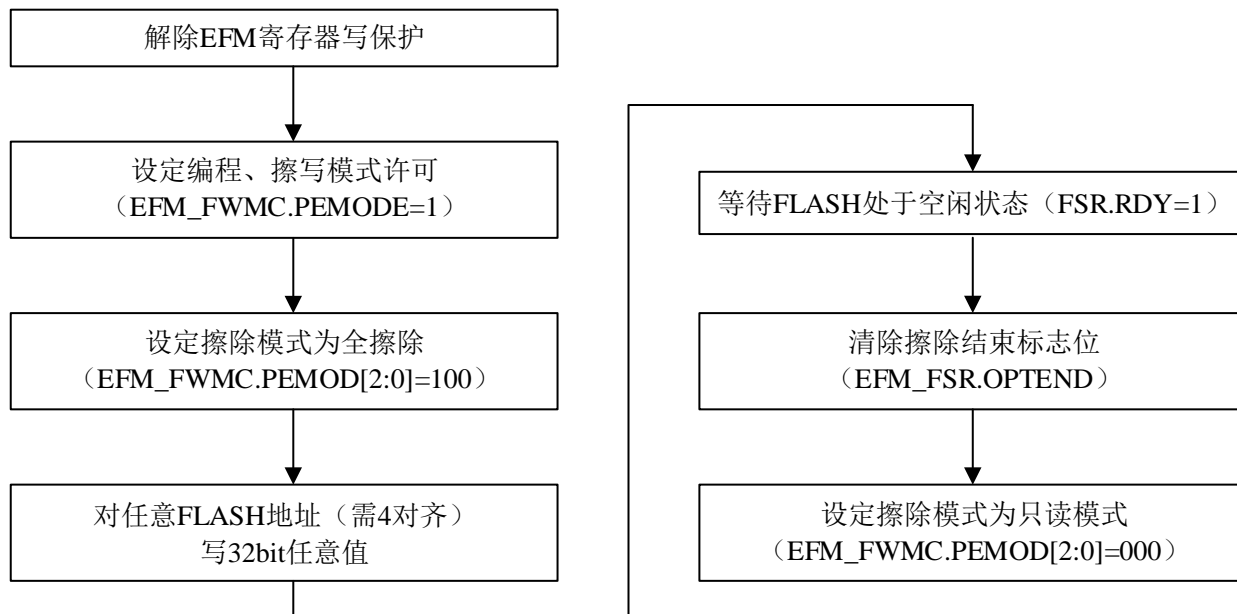
注意：

- EFM_FWMC.PEMODE 是对 EFM_FWMC.PEMOD[2:0]设定的许可；
- 擦除完成后需把 EFM_FWMC.PEMOD[2:0]设定为只读模式，预防对 FLASH 误操作，导致整个扇区擦除。

3.2.2.3 全擦除

EFM 提供扇区擦除和全擦除两种擦除方式。对 FLASH 进行全擦除操作后整个 FLASH 区域所有地址数据刷新为全 1。

全擦除的设定步骤如下：

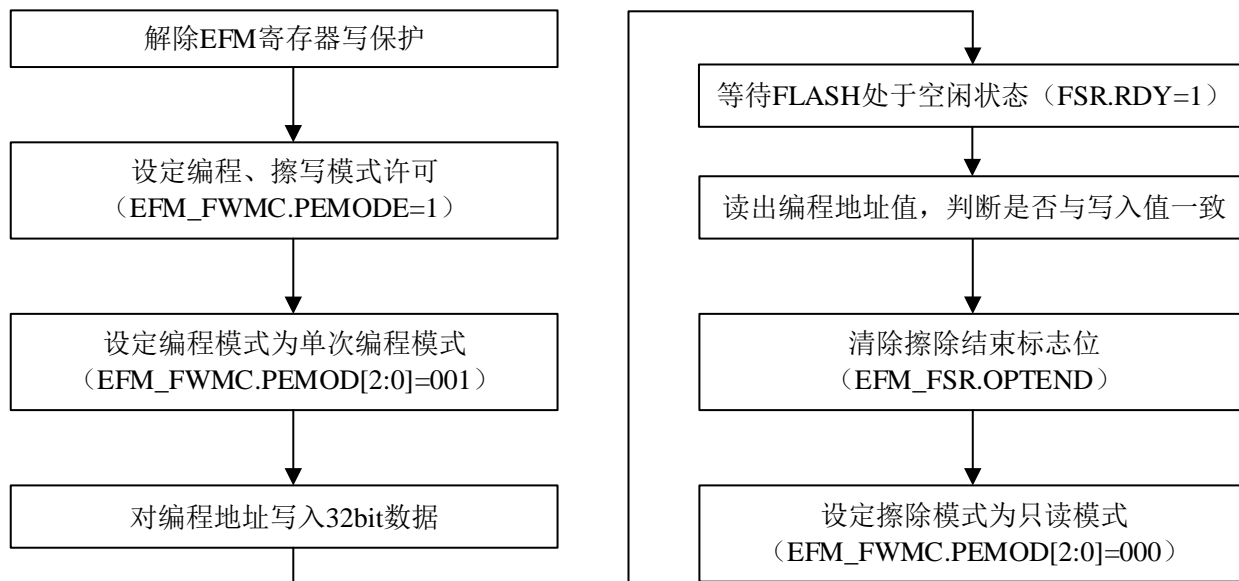


注意：

- 擦除完成后需把 EFM_FWMC.PEMOD[2:0] 设定为只读模式，预防对 FLASH 误操作，导致整个扇区擦除。

3.2.2.4 单次编程无回读

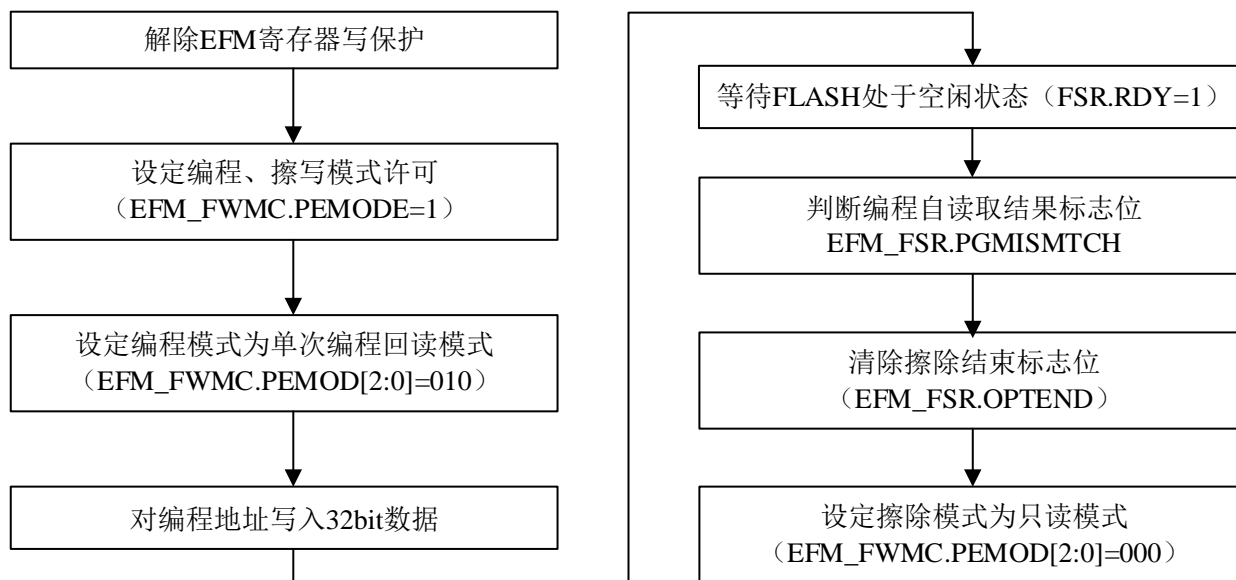
单编程无回读模式设定步骤如下：



3.2.2.5 单次编程回读

单编程回读模式是指编程结束后自动读取编程地址并和写入数据对比，输出判断一致标志位 EFM_FSR.PGMISMTCH。

单编程回读模式设定步骤如下：



注意：

- EFM_FSR.PGMISMTCH 为 0，表示编程成功，为 1 表示该 FLASH 地址已遭破坏，永废弃。

3.2.2.6 连续编程

当连续对 FLASH 地址进行编程时，推荐使用连续编程模式。连续编程模式比单编程模式可以节约时间 50% 以上。连续编程模式时，频率不能低于 12MHz。

连续编程模式和单编程模式事件对比如下表所示：

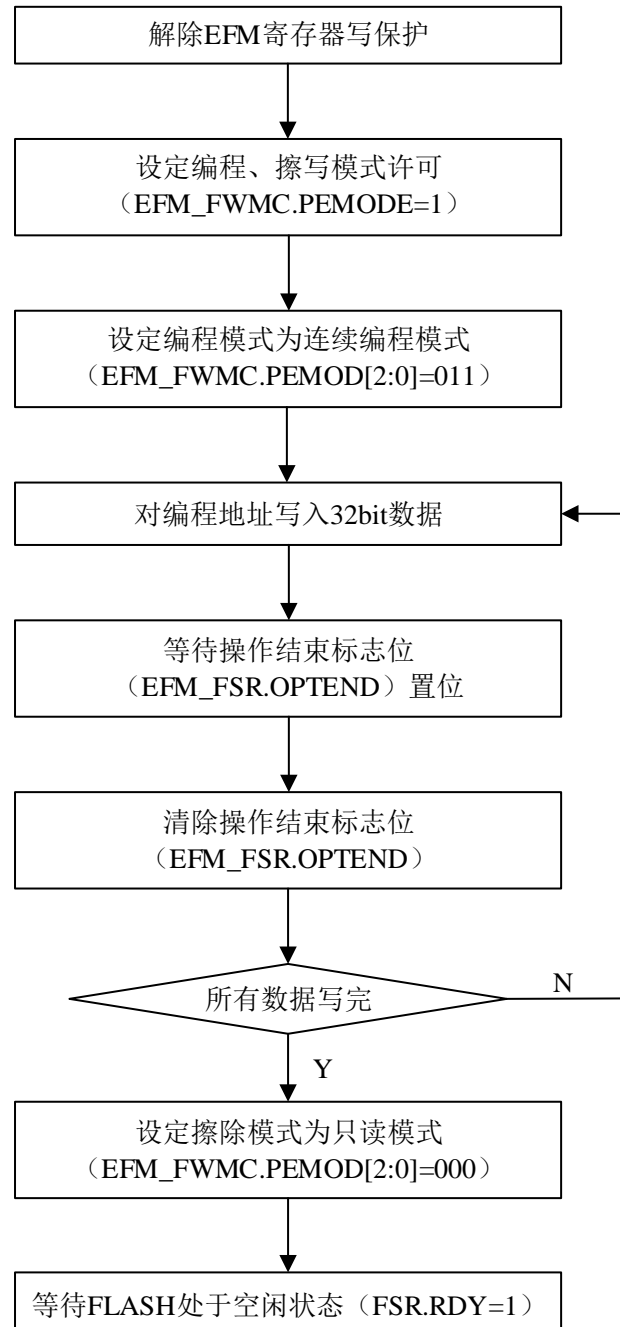
符号	参数	条件	最小值	典型值	最大值	单位
Tprog	字编程时间	单编程模式	$43+2*Thclk$	$48+4*Thclk$	$53+6*Thclk$	μs
	字编程时间	连续编程模式	$12+2*Thclk$	$14+4*Thclk$	$16+6*Thclk$	μs
Terase	块擦除时间	-	$16+2*Thclk$	$18+4*Thclk$	$20+6*Thclk$	ms
Tmas	全擦除时间	-	$16+2*Thclk$	$18+4*Thclk$	$20+6*Thclk$	ms

注：Thclk 为 CPU 时钟的 1 周期

连续编程设定步骤如下：

注意：

- 在 FLASH 连续编程期间，如果对 FLASH 进行读操作，将会读到不定值。
- 连续编程模式中对任意 FLASH 地址发起 dummy read 前，需关闭缓存使能功能，防止不定数据误入缓存池；同时，此次 dummy read 会产生读冲突置位（即 EFM_FSR.RDCOLERR 置位）。



3.2.2.7 总线保持/释放功能

通过设定寄存器 EFM_FWMC.BUSHLDCCTL 位，可设定 FLASH 编程、擦除期间，总线处于保持还是释放状态。

FLASH 编程、擦除指令在 FLASH 上执行时，该控制位必须设定为 0。擦除指令在 FLASH 意外空间（例如 RAM）执行时，可根据需求自由设定。

当 EFM_FWMC.BUSHLDCCTL 为 1（即 FLASH 编程、擦除期间，总线释放状态）时，在编程（连续编程除外）、擦除结束前（EFM_FSR.RDY=1）对 FLASH 的读写访问将会被保护，标志位 EFM_FSR.BUSCOLERR 位置位。

3.2.3 一次性可编程字节（OTP）

OTP（One Time Program）区域分为 15 个 64 字节的数据块，每块数据对应一个 4Bytes 的锁存地址。详见用户手册表 9-3。

锁存地址用于锁存对应的数据块。锁存地址数据全为 1 时，对应的 OTP 区域数据块可以编程；当锁存地址数据全为 0 时，对应的 OTP 区域数据不可编程。

所有 OTP 数据块和锁存地址均无法擦除。

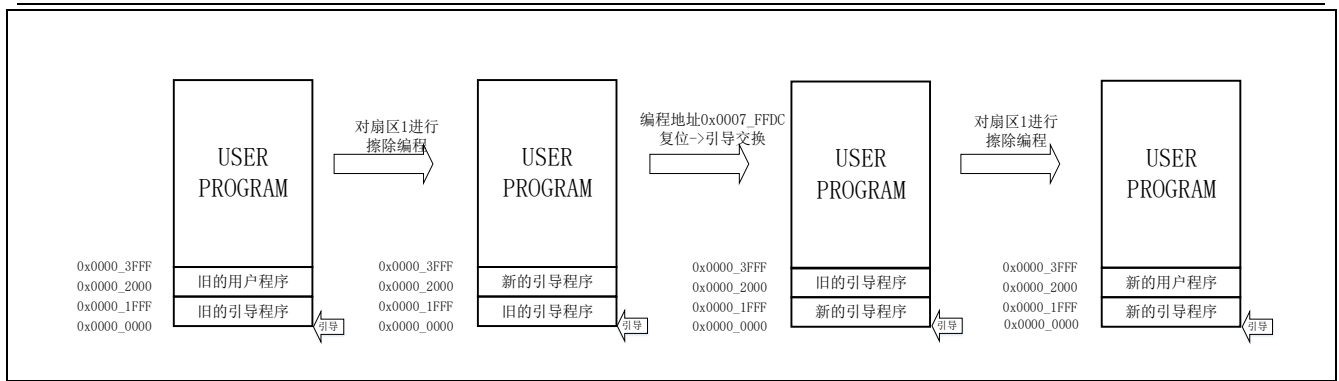
OTP 区域的编程已封装在 flashloader 中，用户可以直接操作 otp 对应地址。具体样例请参考 EFM 模块下的 efm_otp 样例代码。

3.2.4 引导交换

EFM 提供引导交换功能，用户在升级引导程序的时候，对扇区 0（0x00000000~0x00001FFF）进行擦写，如擦写时遇到不可期的意外（掉电、复位），有可能会整个芯片不能正常启动。引导交换功能可以避免这种情况。

在对扇区 0 擦除前预先把新的引导程序写入扇区 1（0x00002000~0x00003FFF），然后对 EFM 地址 0x0007FFDC 进行编程数据 0xFFFF4321，通过端子复位，实现 CPU 从扇区 1 启动新的引导程序，此时，再对扇区 0 进行擦除，重新编程用户程序。

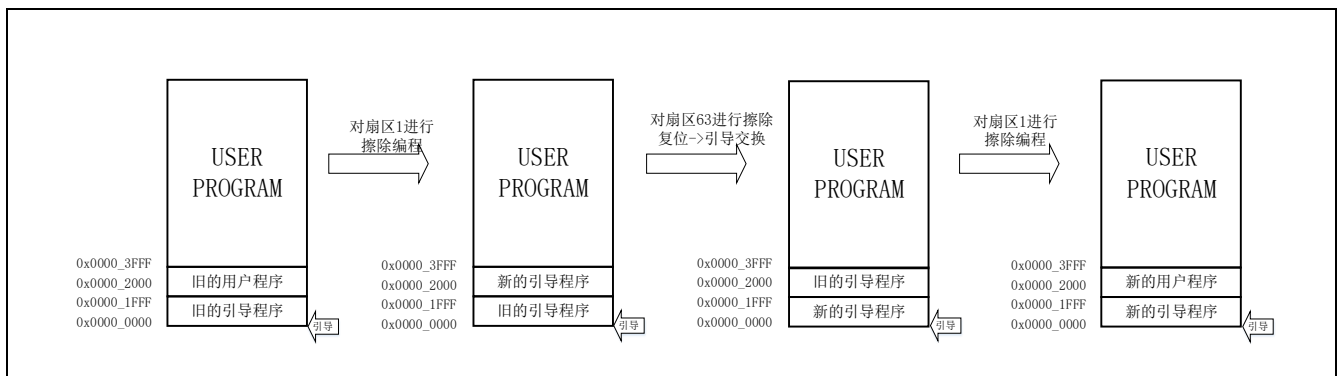
启动引导交换的流程如下图：



再次升级启动引导程序时，由于保存启动扇区交换信息的地址 0x0007FFDC 已经被编程过了（用户可以根据读 FLASH 地址或者 EFM_FSWP 寄存器判断是否使用过启动交换功能，EFM_FSWP.FSWP = 0，表明扇区 0 和扇区 1 已经交换，复位后从扇区 1 启动），需对扇区 63（0x0007E000~0x0007FFFF）进行扇区擦除后再进行启动程序的升级。

在对扇区 0 进行擦除前，预先将新的引导程序写入扇区 1，然后对扇区 63 进行擦除，通过端子复位，CPU 就从扇区 1 启动新的引导程序，此时再对扇区 0 进行擦除，重新编程新的引导程序。

操作流程如下图：



引导交换的样例具体可参考 EFM 模块下 efm_switch 样例代码。

4 样例代码

4.1 代码介绍

用户可根据上述的工作流程编写自己的代码来学习验证该模块，也可以直接通过华大半导体的网站下载到设备驱动库（Device Driver Library, DDL）的样例代码并使用其中的 FLASH 的样例进行验证。

以下部分简要介绍本 AN 基于 DDL 的 FLASH 模块样例 efm_simple 代码所涉及的各项配置。

- 1) 解除 FLASH 寄存器保护：

```
/* Unlock EFM. */  
EFM_Unlock();
```

- 2) 使能 FLASH：

```
/* Enable flash. */  
EFM_FlashCmd(Enable);
```

- 3) 等待 FLASH ready：

```
/* Wait flash ready. */  
while(Set != EFM_GetFlagStatus(EFM_FLAG_RDY));
```

- 4) 扇区擦除：

```
/* Erase sector 62. */  
EFM_SectorErase(FLASH_SECTOR62_ADRR)
```

- 5) 编程 FLASH：

```
u32Addr = FLASH_SECTOR62_ADRR;  
for(i = 0; i < 10; i++)  
{  
    EFM_SingleProgram(u32Addr,u32TestData);  
    u32Addr += 4;  
}
```

- 6) 锁 FLASH 寄存器：

```
/* Lock EFM. */  
EFM_Lock();
```

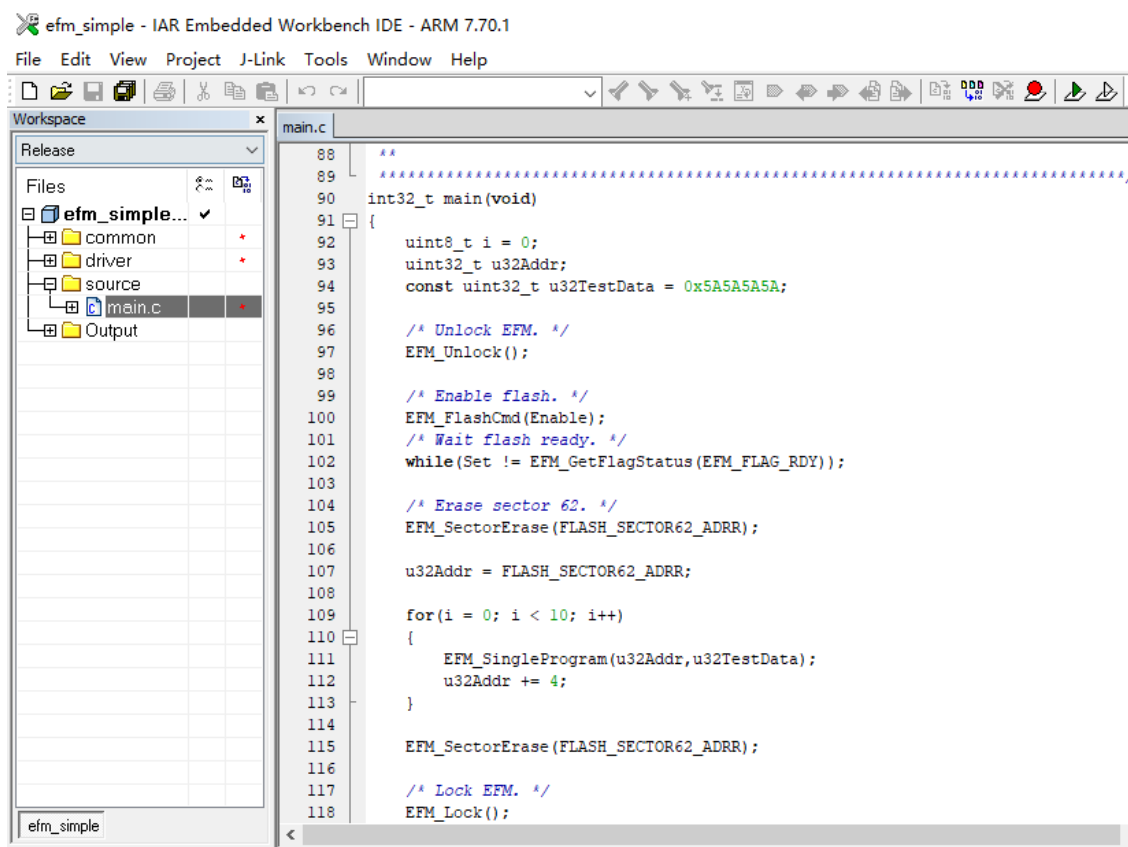
4.2 代码运行



用户可以通过华大半导体的网站下载到 HC32F460 的 DDL 的样例代码（efm_simple），并配合评估用板（EV-HC32F460-LQFP100-050-V1.1）运行相关代码学习使用 FLASH 模块。

以下部分主要介绍如何在评估板上运行 FLASH 样例代码并观察结果：

- 确认安装正确的 IAR EWARM v7.7 工具（请从 IAR 官方网站下载相应的安装包，并参考用户手册进行安装）。
- 从华大半导体网站下载 HC32F460 DDL 代码。
- 下载并运行 efm\efm_simple\中的工程文件：

1) 打开 efm_simple \工程，并打开‘main.c’如下视图：



- 2) 点击  重新编译整个项目。
- 3) 点击  将代码下载到评估板上，分别在 105 行、115 行代码处打断点，全速运行。
- 4) 运行第一个断点处，单步运行，查看 memory，可观察到扇区 62 全为 1，继续全速运行到下一个断点，该地址被写入预期值。基础单步运行，写入值被擦除。

5 版本信息 & 联系方式

日期	版本	修改记录
2019/3/15	Rev1.0	初版发布



如果您在购买与使用过程中有任何意见或建议，请随时与我们联系。

Email: mcu@hdsc.com.cn

网址: <http://www.hdsc.com.cn/mcu.htm>

通信地址: 上海市张江高科园区碧波路 572 弄 39 号

邮编: 201203

