NV32F100x - FLASH 保护机制配置

在 IAR 环境下：

一、首先，用包中的 icf 文件对之前的 icf 进行替换，改动如下：

```
define symbol __ICFEDIT_intvec_start__ = 0x00000000;   //PflaSh启动
/*-Memory Regions-*/
define symbol __ICFEDIT_region_ROM_start__ = 0x00000000;
define symbol __ICFEDIT_region_ROM_end__   = 0x00020000;
define symbol __ICFEDIT_region_RAM_start__ = 0x1ffffc10;
define symbol __ICFEDIT_region_RAM_end__   = 0x20000000;
define symbol __region_FlashConfig_start   = 0x00000400;  //定义FlashConfig字段的首地址为0x400
/*-Sizes-*/
define symbol __ICFEDIT_size_cstack__ = 0x400;
define symbol __ICFEDIT_size_heap__   = 0x200;

define symbol __region_RAM2_start__ = 0x20000000;
define symbol __region_RAM2_end__   = 0x20001800;
define symbol __ICFEDIT_FlashConfig_start__ = __region_FlashConfig_start;

define exported symbol __VECTOR_TABLE = 0x00000000;
define exported symbol __VECTOR_RAM = __ICFEDIT_region_RAM_start__ - 0x410;

define exported symbol __BOOT_STACK_ADDRESS = __region_RAM2_end__ - 8;
/**** End of ICF editor section. ###ICF###*/
define symbol __code_start__ = 0x00000410;

define memory mem with size = 4G;
define region ROM_region   = mem:[from __ICFEDIT_region_ROM_start__   to __ICFEDIT_region_ROM_end__];
define region RAM_region   = mem:[from __ICFEDIT_region_RAM_start__   to __ICFEDIT_region_RAM_end__]| mem:[from __region_RAM2_start__ to __region_RAM2_end__];
define block CSTACK    with alignment = 8, size = __ICFEDIT_size_cstack__   { };
define block HEAP      with alignment = 8, size = __ICFEDIT_size_heap__     { };

initialize manually { readwrite };
initialize manually { section .data};
initialize manually { section .textrw };
do not initialize  { section .noinit };

define block CodeRelocate { section .textrw_init };
define block CodeRelocateRam { section .textrw };

place at address mem:__ICFEDIT_intvec_start__ { readonly section .intvec };
place at address mem:__code_start__ { readonly section .noinit };
place at address mem:__ICFEDIT_FlashConfig_start__ { readonly section .flashConfig };

place in ROM_region   { readonly, block CodeRelocate};

place in RAM_region   { readwrite, block CodeRelocateRam,
                        block CSTACK, block HEAP };
```

二、在 main.c 中定义数组

```
const char  FlashConfig[16] @ ".flashConfig" =

{

  0xff,0xff,0xff,0xff,

  0xff,0xff,0xff,0xff,

  0xff,0xff,0xff,0xff,

  0xff,0xff,0xfe,0xfe   //红色标记处为 0xfe 则为正常状态，为 0xfc 则为 flash 上锁

};
```

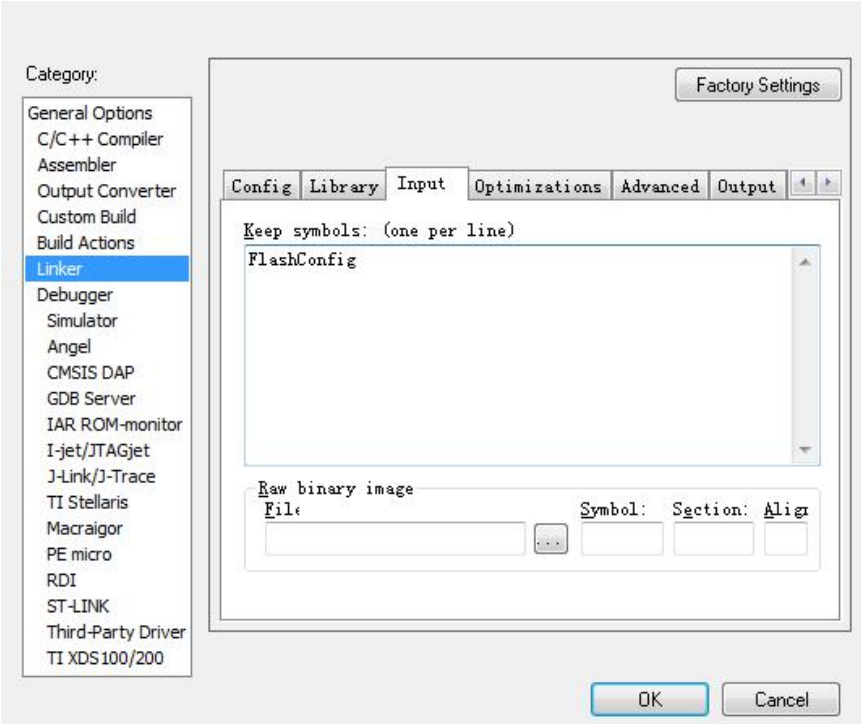
```
/*****************************************************************************
 *
 * 实验名称：FLASH-LOCK
 * 实验平台：NV32开发板
 * 板载芯片：NV32F100FL64E
 * 实验效果：该例阐述了如何在NV32F100x上加锁FLASH
 *
 *
*****************************************************************************/

#include "common.h"
#include "flash.h"
#include "sysinit.h"


int main (void);

const char  FlashConfig[16] @ ".flashConfig" =
{
0xff,0xff,0xff,0xff,
0xff,0xff,0xff,0xff,
0xff,0xff,0xff,0xff,
0xff,0xff,0xfc,0xfe
};
```

三、在工程配置中，定义如下标识



四、在工程 vectors.c 中注释如下红框的代码

```
#ifndef KEIL
#ifndef USE_BOOTLOADER
#ifdef KEIL
const uint32_t  __flash_config[] __attribute__((at(0x400))) =
#elif (defined(__GNUC__))
const uint32_t __flash_config[] __attribute__ ((section(".cfmconfig"))) =
#else
//#pragma location = 0x400
//__root const uint32_t  __flash_config[] = //@ ".intvec" =
#endif
//{
//    CONFIG_1,
//    CONFIG_2,
//    CONFIG_3,
//    CONFIG_4,
//};
#endif
#endif
```

五、编译-FLASH 加锁成功

在 MAP 文件中查看 0x400 地址

```
EFM_LaunchCMD           0x1ffffc11   0x34  Code  Gb  flash.o [1]
FBE_to_FBELP            0x00000c79    0xe  Code  Gb  ics.o [1]
FBI_to_FBILP            0x00000c65    0xe  Code  Gb  ics.o [1]
FEI_to_FBE              0x00000b05   0x76  Code  Gb  ics.o [1]
FEI_to_FBE_OSC          0x00000b7d   0x80  Code  Gb  ics.o [1]
FEI_to_FBI              0x00000aa9   0x5c  Code  Gb  ics.o [1]
FEI_to_FEE              0x00000a51   0x58  Code  Gb  ics.o [1]
FEI_to_FEE_OSC          0x00000bfd   0x5c  Code  Gb  ics.o [1]
FlashConfig             0x00000400   0x10  Data  Gb  Flash_demo.o [1]
Flash_EraseSector       0x0000121f   0x30  Code  Gb  flash.o [1]
Flash_Init              0x000010c9   0x30  Code  Gb  flash.o [1]
Flash_Program           0x000010f9   0xb6  Code  Gb  flash.o [1]
Flash_Program1LongWord  0x000011af   0x2c  Code  Gb  flash.o [1]
Flash_Program2LongWords
```

在 MDK 环境下:

startup_NV32.s 中的 FSEC 为 0XFE 时则为正常状态

修改 FSEC 为 0XFF,0xFD,0XFC 则为 FLASH 读保护。

```
173  FSEC            EQU     0xFE
174  ;   </h>
175  ;   <h> Flash Option Register (FOPT)
176  FOPT            EQU     0xFE
177  ;   </h>
178                  IF      :LNOT::DEF:RAM_TARGET
179                  AREA    |.ARM.__at_0x400|, CODE, READONLY
180                  DCB     BackDoorK0, BackDoorK1, BackDoorK2, BackDoorK3
181                  DCB     BackDoorK4, BackDoorK5, BackDoorK6, BackDoorK7
182                  DCB     0xFF,       0xFF,       0xFF,       0xFF
183                  DCB     EEPROT,      FPROT,     FSEC,       FOPT
184                  ENDIF
185
186                  AREA    |.text|, CODE, READONLY
```

在生成 bin 文件后:

修改 0x40E 地址的字节为 0xFC,0xFD,0xFF 则为 FLASH 读保护

注：建议在开发过程中不要使能读保护，否则需要在再次下载前对芯片进行解锁操作，才可以正常下载调试。