FM0+
FM3
FM4

SPANSION®

How to setup Flash Security

32-BIT MICROCONTROLLER
FM0+, FM3, and FM4 Families

APPLICATION NOTE

Publication Number FMx_AN706-00089-1v0-E Revision 1.0 Issue Date July 01, 2014

Revision History

Date	Issue
2014-07-01	V1.0; MWi; 1st version

Target products

This application note is described about below products:

Series	Product Number (not included Package suffix)
FM0+	All products
FM3	All products
FM4	All products

Table of Contents

1. Introduction 5
1.1 About Document 5
2. Mechanism of Flash Security 6
2.1 Methods of setting Flash Security 6
3. Setting Flash Security by Project 7
3.1 Procedure 7
3.1.1 Assembly and Linker File for IAR EWARM 7
3.1.2 Assembly File for KEIL µVISION 7
3.1.3 Assembly and Linker File for Atollic TrueStudio 8
3.1.4 Assembly and Linker File for GNU Compiler Environment 9
4. Setting Flash Security by Patching S-Record or HEX File 10
4.1 S-Record File Format 10
4.2 HEX File Format 11
5. Flash Security Sector Overview 12
5.1 FM0+ Devices 12
5.2 FM3 Devices 12
5.3 FM4 Devices 12

1. Introduction

1.1 About Document

This application note describes how to setup the Flash Security for FM0+, FM3, and FM4 devices.

2. Mechanism of Flash Security

The default setting of any FMx-MCU is Flash Security disabled. In this mode the user has full access to the Flash memory via Flash programming tools and JTAG debuggers from outside.

By enabling the Flash security the Flash contents cannot be accessed from outside anymore. Any read out trial results in `0x0000` values. The only way to unlock the Flash Security is to perform a Flash chip erase sequence via a Flash programming tool.

The Flash Security is not controlled by Boot-ROM or software – it is a part of the hardware macro of the Flash memory itself.

The Flash Security is enabled, if a certain data pattern (e.g. `0x0001`) is written to a special Flash sector address.

2.1 Methods of setting Flash Security

Because the Spansion software examples and templates do not contain a Flash Security setting, this application note describes how to enable it.

There are two mechanisms of setting the Flash Security: By special project settings or by patching the S-Record or HEX record file.

In any case a Flash programming tool (serial/USB) has to be used to enable the Flash Security by programming, because the JTAG interface will stop working, if the Flash Security is set.

3. Setting Flash Security by Project

3.1 Procedure

The easiest way is to create a small assembly file with a memory section of the Flash Security sector. In this memory section the Flash Security pattern is defined as a constant.

This assembly file will finally added to the project after debugging phase and just before final project build. IAR's EWARM and GCC tool chains need also a modification of their linker files.

Afterwards use a Flash programming tool (serial/USB) to program the Flash memory with Flash Security enabled.

3.1.1 Assembly and Linker File for IAR EWARM

The following assembly code can be used for enabling the Flash Security. In this example the Flash Security sector address is `0x0010.0000` and the pattern is `0x0001`:

```
    MODULE FLASH_SECURITY
    SECTION .flashsec : CONST (2)

    DC16 0x0001

    END
```

The filename is regardless. It is not necessary to give it the same name as the assembly module name.

Edit the corresponding linker file (**.icf*) and add the following lines:

```
define region FLASH_SECURITY_region = mem:[from 0x00100000 to 0x00100003];
define symbol FLASH_SECURITY_ADDRESS = 0x00100000;
place at address mem:FLASH_SECURITY_ADDRESS { readonly section .flashsec };
```

Assuming the filename is *flashsecurity.s* this section will occur in the mapping file of the linker after build like:

```
"A2":                               0x4
  FLASHSEC  const    0x00100000    0x2  flashsecurity.o [1]
                   - 0x00100002    0x2
```

3.1.2 Assembly File for KEIL µVISION

The following assembly code can be used for enabling the Flash Security. In this example the Flash Security sector address is `0x0010.0000` and the pattern is `0x0001`:

```
                AREA    |.ARM.__at_0x00100000|, DATA, READONLY
                DCB     0x01
                DCB     0x00

                END
```

The filename is regardless.

Assuming the filename is *flash_security.s* this section will occur in the mapping file of the linker as:

```
Load Region LR$$.ARM.__at_0x00100000 (Base: 0x00100000, Size: 0x00000004, Max: 0x00000004, ABSOLUTE)

    Execution Region ER$$.ARM.__at_0x00100000 (Base: 0x00100000, Size: 0x00000004, Max: 0x00000004,
ABSOLUTE, UNINIT)

    Base Addr    Size         Type   Attr      Idx    E Section Name        Object

    0x00100000   0x00000002   Data   RO          83    .ARM.__at_0x00100000  flash_security.o
```

3.1.3 Assembly and Linker File for Atollic TrueStudio

The following assembly code can be used for enabling the Flash Security. In this example the Flash Security sector address is `0x0010.0000` and the pattern is `0x0001`:

```
                .section .flashsecurity, "a"
                .global _flashsecurity
_flashsecurity: .long 0x00000001
```

The filename is regardless.

Edit the corresponding linker file (*.*ld*) and add the highlighted line in the MEMORY definition:

```
/* Specify the memory areas */
MEMORY
{
  FLASH (rx)       : ORIGIN = 0x00000000, LENGTH = 512K
  RAM (xrw)        : ORIGIN = 0x1FFF8000, LENGTH = 64K
  FLASHSEC (r)     : ORIGIN = 0x00100000, LENGTH = 4
  MEMORY_B1 (rx)   : ORIGIN = 0x60000000, LENGTH = 0K
}
```

Add the highlighted lines to the SECTION definitions just before the RAM section definitions:

```
    . = ALIGN(4);
    _edata = .;         /* define a global symbol at data end */
  } >RAM AT> FLASH

  .flashsecurity :
  {
        . = ALIGN(4);
        _flashsecurity = .;
        KEEP(*(.flashsecurity))
        . = ALIGN(4);
  } >FLASHSEC

  /* Uninitialized data section */
  . = ALIGN(4);
```

Assuming the filename is *flashsec.s* this section will occur in the mapping file of the linker as:

```
.flashsecurity  0x00100000        0x4
                0x00100000                . = ALIGN (0x4)
                0x00100000                _flashsecurity = .
 *(.flashsecurity)
 .flashsecurity
                0x00100000        0x4 source/flashsec.o
                0x00100004                . = ALIGN (0x4)
```

3.1.4 Assembly and Linker File for GNU Compiler Environment

Refer to the previous chapter (Atollic TrueStudio).

4. Setting Flash Security by Patching S-Record or HEX File

4.1 S-Record File Format

The S-Record file format lines consist typically of the character ‘S’, the type of the line, the byte count of the data payload of this line, an address, the data payload, and a checksum byte.

The type codes of S-Record lines are:

Code	Description	Number of Address Bytes	Datafield
S0	Block Header	2	Yes
S1	Data Payload	2	Yes
S2	Data Payload	3	Yes
S3	Data Payload	4	Yes
S5	Record Count	2	No
S7	End of Block	4	No
S8	End of Block	3	No
S9	End of Block	2	No

It is a good idea to put the Flash security data before the last S-Record line (S7-9):

Assume the Flash security is enabled by setting 0x0001 to the address 0x0010.0000. Then the additional S-Record line is:

```
S20810000001000000E6
```

Detailed explanation of this line:

	Data Payload with 3 Byte Address	Byte Count	3 Byte Address	Data Payload (Big Endian)	Check Sum
S	2	08	100000	01000000	E6

The checksum is calculated as: Add all bytes after the S*n* type and before the checksum byte itself. Take only the lower byte of the sum and invert the sum:

0x08 + 0x10 + 0x01 = 0x19 (0x00 bytes are skipped in the calculation.)
$\overline{0x19}$ = 0xE6

4.2 HEX File Format

The Intel HEX file format lines for 32 bit architectures consist typically of a colon, the byte count of the data payload of this line, an address, the type of the line, the data payload, and a checksum byte.

The type codes of HEX lines are:

Code	Description
0x00	Data payload
0x01	End of file
0x02	Extended Segment Address
0x03	Start Segment Address
0x04	Extended Linear Address
0x05	Start Linear Address

It is a good idea to put the Flash security data before the last HEX line, which is usually:

```
:00000001FF
```

Assume the Flash security is enabled by setting 0x0001 to the address 0x0010.0000. Then the additional HEX lines are:

```
:020000040010EA
:0400000001000000FB
```

Detailed explanation of these lines:

	Data Count	Dummy Address	Extended Linear Address	Upper 16 Bit of Security Address	Check Sum
:	02	0000	04	0010	EA

	Data Count	Lower 16 Bit of Security Address	Data Line	Security Code Data (Big Endian)	Check Sum
:	04	0000	00	01000000	FB

The checksum is calculated as: Add all bytes after ':' and before the checksum byte itself. Take only the lower byte of the sum. Invert the sum and add 1.

Example for first line:

0x02 + 0x04 + 0x10 = 0x16 (0x00 bytes are skipped in the calculation)
$\overline{0x16}$ = 0xE9
0xE9 + 0x01 = 0xEA

5. Flash Security Sector Overview

The following tables give an overview of the Flash Security sector address and its activation pattern. Refer to the peripheral manual for cross reference between device type and part number.

5.1 FM0+ Devices

Device Type	Flash Security address	Activation pattern
0	–	–
1	0x0010_0000	0x0001

5.2 FM3 Devices

Device Type	Flash Security address	Activation pattern
0	0x0010_0000	0x0001
1	0x0010_0000	0x0001
2	0x0010_0000	0x0001
3	0x0010_0000	0x0001
4	0x0010_0000	0x0001
5	0x0010_0000	0x0001
6	0x0010_0000	0x0001
7	0x0010_0000	0x0001
8	0x0010_0000	0x0001
9	0x0010_0000	0x0001
10	0x0010_0000	0x0001
11	0x0010_0000	0x0001
12	0x00**4**0_0000	0x0001

5.3 FM4 Devices

Device Type	Flash Security address	Activation pattern
0	0x0040_0000	0x0001

FMx_AN706-00089-1v0-E

Spansion • Controller Manual

FM0+, FM3, FM4 Families
32-BIT MICROCONTROLLER
All FM0+, FM3, FM4 Series
How to setup Flash Security

July 2014 Rev. 1.0

Published: Spansion Inc.
Edited: MCU Industrial Application Team

Colophon

The products described in this document are designed, developed and manufactured as contemplated for general use, including without limitation, ordinary industrial use, general office use, personal use, and household use, but are not designed, developed and manufactured as contemplated (1) for any use that includes fatal risks or dangers that, unless extremely high safety is secured, could have a serious effect to the public, and could lead directly to death, personal injury, severe physical damage or other loss (i.e., nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system), or (2) for any use where chance of failure is intolerable (i.e., submersible repeater and artificial satellite). Please note that Spansion will not be liable to you and/or any third party for any claims or damages arising in connection with above-mentioned uses of the products. Any semiconductor devices have an inherent chance of failure. You must protect against injury, damage or loss from such failures by incorporating safety design measures into your facility and equipment such as redundancy, fire protection, and prevention of over-current levels and other abnormal operating conditions. If any products described in this document represent goods or technologies subject to certain restrictions on export under the Foreign Exchange and Foreign Trade Law of Japan, the US Export Administration Regulations or the applicable laws of any other country, the prior authorization by the respective government entity will be required for export of those products.

Trademarks and Notice

The contents of this document are subject to change without notice. This document may contain information on a Spansion product under development by Spansion. Spansion reserves the right to change or discontinue work on any product without notice. The information in this document is provided as is without warranty or guarantee of any kind as to its accuracy, completeness, operability, fitness for particular purpose, merchantability, non-infringement of third-party rights, or any other warranty, express, implied, or statutory. Spansion assumes no liability for any damages of any kind arising out of the use of the information in this document.

Copyright © 2014 Spansion LLC. All rights reserved. Spansion®, the Spansion logo, MirrorBit®, MirrorBit® Eclipse™, ORNAND™ and combinations thereof, are trademarks and registered trademarks of Spansion LLC in the United States and other countries. Other names used are for informational purposes only and may be trademarks of their respective owners.