AN205551

F²MC-16FX Family, Flash Security

This application note describes the functionality of the Flash Security.

1 Introduction

The 16FX Family of microcontrollers features a Flash Security function. This function ensures that the contents of the Flash memory cannot be read. Instead, when reading the Flash memory, only the data 0x00 is returned for every location.

2 Flash Security

The Flash Security feature protects the content of the Flash memory. In the following,

It is explained how this feature can be used.

2.1 Features

Often, it is desirable to protect the content of the Flash memory from read-out. For this, the 16FX Family MCUs offer the Flash Security feature. When the Flash Security is enabled, Flash memory cannot be read by

- Program activated by external boot vector fetch (modes 0/1/6)
- External parallel Flash memory programmer (mode 7)
- Serial communication mode (mode 2)

However internal user program that has been started in internal vector mode (mode 3) has access to the content of the Flash memory.

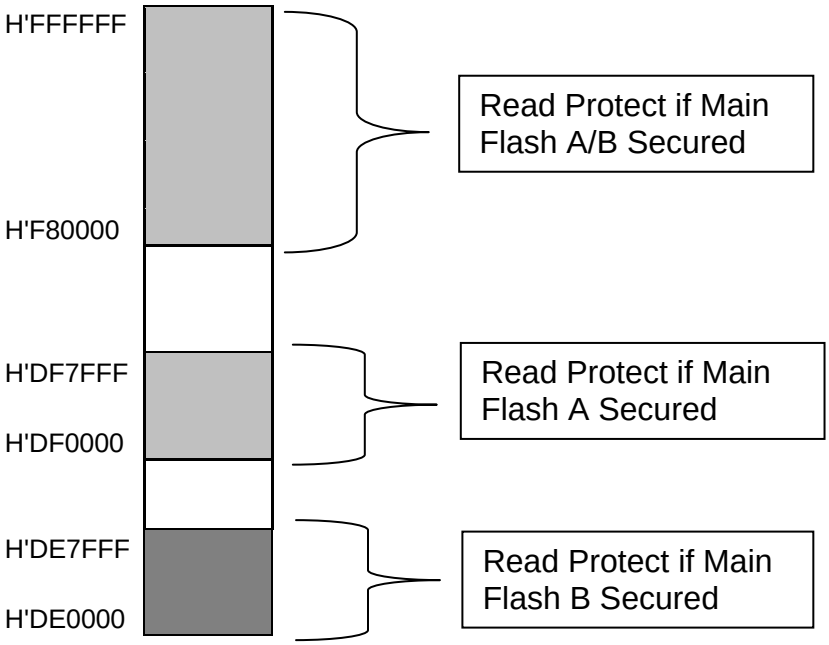
Generally, there are 3 different access levels for the Flash memory:

- Unprotected (Flash Security disabled)
- Flash Security enabled, but can be disabled by an Unlock Key
- Flash Security enabled and cannot be disabled

The default is that the Flash Security is disabled. The Flash Security can be enabled by storing a certain pattern at a specific location. See below for more details. If the Flash Security is enabled, the user can also supply a 16-Byte long Unlock Key that allows disabling the Flash Security. If the supplied Unlock Key is all 0, the Flash Security cannot be disabled at all.

Note: The Flash memory normally allows erasing either the complete memory (“Chip Erase”) or individual sectors (“Sector Erase”). Because the Flash Security is enabled by storing a pattern at a specific location, it would be possible to disable the Flash Security by erasing just the single sector that contains this location. To avoid this, a Sector Erase cannot be performed when the Flash Security is enabled. Only a Chip Erase is allowed. In case of Chip Erase, the Flash memory is erased from the highest memory address to the lowest memory address because of that Flash Security byte is erased at last after erasing entire content of Flash memory.

2.2 Memory Map showing Secured Areas



2.3 Configuration

The Flash Security feature is configured using the ROM Configuration Block. This is done separately for the Main Flash A memory and the Main Flash B memory, if available. The settings can be made manually. Because this is rather tedious, it can be configured easily using the `start.asm` file.

2.3.1 Start.asm Configuration

The configuration block of the Flash Security looks like shown below.

```
;---------------------------------------------------------------------
; 4.12 Flash Security
;---------------------------------------------------------------------
; All settings regarding Flash B are ignored on devices that do not
; have a Flash B.

#set      FLASH_A_SECURITY_ENABLE       OFF ; <<< enable Flash Security for
Flash A
#set      FLASH_B_SECURITY_ENABLE       OFF ; <<< enable Flash Security for
Flash B

; set the Flash Security unlock key (16 bytes)
; all 0: unlock not possible
#set      FLASH_A_UNLOCK_0              0x00
#set      FLASH_A_UNLOCK_1              0x00
#set      FLASH_A_UNLOCK_2              0x00
#set      FLASH_A_UNLOCK_3              0x00
#set      FLASH_A_UNLOCK_4              0x00
#set      FLASH_A_UNLOCK_5              0x00
#set      FLASH_A_UNLOCK_6              0x00
#set      FLASH_A_UNLOCK_7              0x00
#set      FLASH_A_UNLOCK_8              0x00
#set      FLASH_A_UNLOCK_9              0x00
#set      FLASH_A_UNLOCK_10             0x00
#set      FLASH_A_UNLOCK_11             0x00
#set      FLASH_A_UNLOCK_12             0x00
#set      FLASH_A_UNLOCK_13             0x00
#set      FLASH_A_UNLOCK_14             0x00
#set      FLASH_A_UNLOCK_15             0x00

#set      FLASH_B_UNLOCK_0              0x00
#set      FLASH_B_UNLOCK_1              0x00
#set      FLASH_B_UNLOCK_2              0x00
#set      FLASH_B_UNLOCK_3              0x00
#set      FLASH_B_UNLOCK_4              0x00
#set      FLASH_B_UNLOCK_5              0x00
#set      FLASH_B_UNLOCK_6              0x00
#set      FLASH_B_UNLOCK_7              0x00
#set      FLASH_B_UNLOCK_8              0x00
#set      FLASH_B_UNLOCK_9              0x00
#set      FLASH_B_UNLOCK_10             0x00
#set      FLASH_B_UNLOCK_11             0x00
#set      FLASH_B_UNLOCK_12             0x00
#set      FLASH_B_UNLOCK_13             0x00
#set      FLASH_B_UNLOCK_14             0x00
#set      FLASH_B_UNLOCK_15             0x00
```

Enabling the Flash Security for the Main Flash A is as easy as setting:

```
;=====================================================================
; 4.12 Flash Security
;=====================================================================

#set      FLASH_A_SECURITY_ENABLE       ON ; <<< enable Flash Security for Main
Flash A
```

To enable the Flash Security for the Main Flash B requires the following setting:

```
#set      FLASH_B_SECURITY_ENABLE            ON ; <<< enable Flash Security
for Flash B
```

These settings enable the Flash Security. In this case, it is not possible to disable the feature using an Unlock Key. To allow this unlocking, the desired Unlock Key needs to be specified. This can be done separately for the Main Flash A memory and the Main Flash B memory. An example for the Main Flash A memory and Main Flash B memory is shown below:

```
; set the Flash Security unlock key (16 bytes)
; all 0: unlock not possible
#set      FLASH_A_UNLOCK_0              0x01
#set      FLASH_A_UNLOCK_1              0x23
#set      FLASH_A_UNLOCK_2              0x45
#set      FLASH_A_UNLOCK_3              0x67
#set      FLASH_A_UNLOCK_4              0x89
#set      FLASH_A_UNLOCK_5              0xAB
#set      FLASH_A_UNLOCK_6              0xCD
#set      FLASH_A_UNLOCK_7              0xEF
#set      FLASH_A_UNLOCK_8              0x01
#set      FLASH_A_UNLOCK_9              0x23
#set      FLASH_A_UNLOCK_10             0x45
#set      FLASH_A_UNLOCK_11             0x67
#set      FLASH_A_UNLOCK_12             0x89
#set      FLASH_A_UNLOCK_13             0xAB
#set      FLASH_A_UNLOCK_14             0xCD
#set      FLASH_A_UNLOCK_15             0xEF

#set      FLASH_B_UNLOCK_0              0x01
#set      FLASH_B_UNLOCK_1              0x23
#set      FLASH_B_UNLOCK_2              0x45
#set      FLASH_B_UNLOCK_3              0x67
#set      FLASH_B_UNLOCK_4              0x89
#set      FLASH_B_UNLOCK_5              0xAB
#set      FLASH_B_UNLOCK_6              0xCD
#set      FLASH_B_UNLOCK_7              0xEF
#set      FLASH_B_UNLOCK_8              0x01
#set      FLASH_B_UNLOCK_9              0x23
#set      FLASH_B_UNLOCK_10             0x45
#set      FLASH_B_UNLOCK_11             0x67
#set      FLASH_B_UNLOCK_12             0x89
#set      FLASH_B_UNLOCK_13             0xAB
#set      FLASH_B_UNLOCK_14             0xCD
#set      FLASH_B_UNLOCK_15             0xEF
```

2.3.2 Manual Configuration

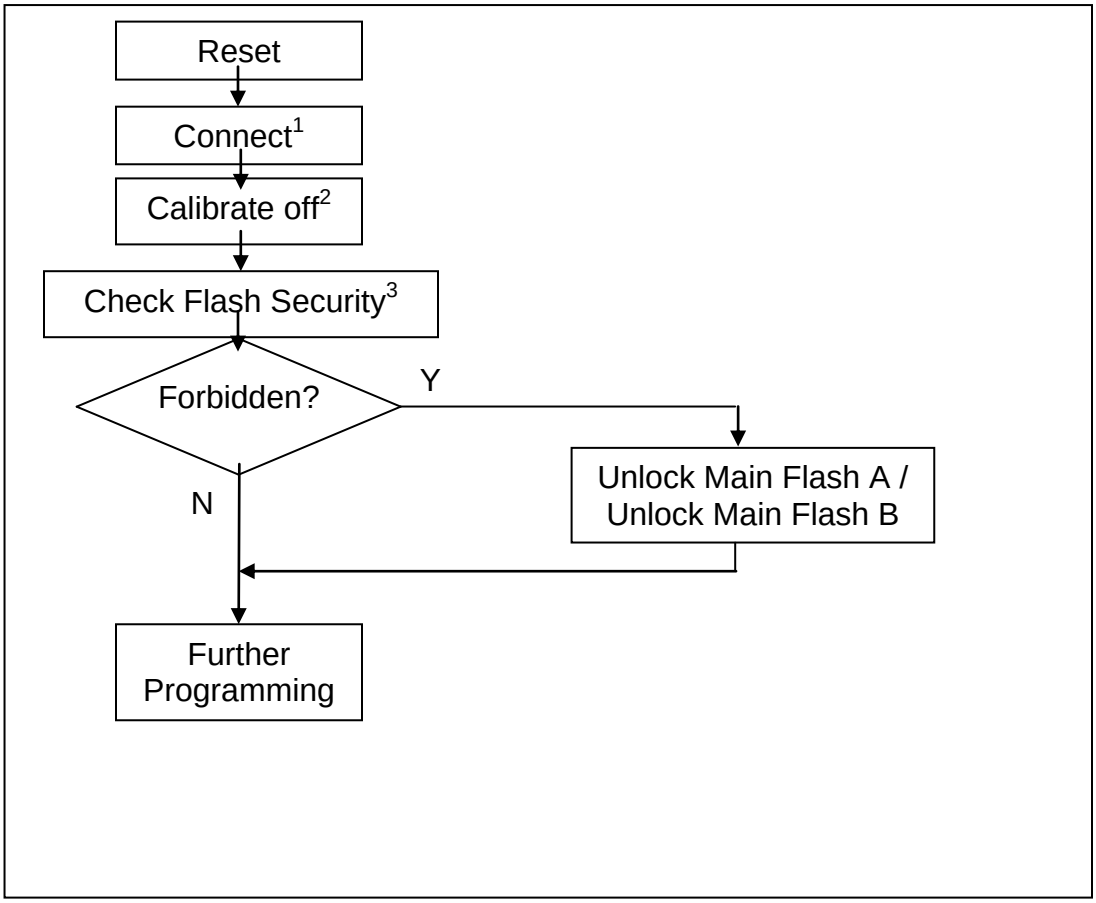
The Flash Security feature can be configured by writing certain patterns at specific Flash memory locations. The table below details of these locations.

	Main Flash A memory	Main Flash B memory
Flash Security Byte	MSBA @ 0xDF0000	MSBB @ 0xDE0000
Enable protection	MSBA = 0x99	MSBB = 0x99
Disable Protection	MSBA = 0x66	MSBB = 0x66
Security Unlock Key location	0xDF0002 – 0xDF0011	0xDE0002 – 0xDE0011

2.4 Unlocking the Flash Security

The Flash Security feature can be unlocked temporarily by communicating with the Boot-ROM. For details refer to the Application Note AN204773 “MCU Flash Programming and Boot-ROM Protocol”.

The following flow chart shows the flow of sequence of communication from a PC to the Boot ROM in order to unlock the Flash Security.



[1,2,3] Further information about these command can be found in AN204773 “MCU Flash Programming and Boot-ROM Protocol”

When the communication is established and Flash memory is secured, the UNLOCK command must be used to unlock it. The format of the command is as follows:

Header		Payload					Check-sum
Cmd	Select	Key 0	Key 1	…	Key 14	Key 15	
0x0A	*0xXX*	*Key0*	*Key1*	…	*Key14*	*Key15*	*0xYY*
PC → MCU							

Select: selects the ROM/Flash module. (0x00: Main ROM/Flash A, 0x01: Main Flash B.)

Key: is the 16-Byte code to be compared against stored key.

Checksum: is calculated with the following equation. It includes all bytes (*n*) except the calibration header[4] (`0x00`, `0x55`):

$$Checksum = 0xFF - \left(\sum_{i=1}^{n} data_i\right) \bmod 0x100 - \left\lfloor \frac{\sum_{i=1}^{n} data_i}{0x100} \right\rfloor - \left\lfloor \frac{\sum_{i=1}^{n} data_i}{0x10000} \right\rfloor$$

The last two summands are correction values, if the overall sum crosses each `0x100` boundary and the `0x10000` boundary.

For the unlocking Main Flash B with Unlock Key as shown in section 2.3.1 checksum can be calculated as follows:

$$Sum = 0x0A + 0x01 + 0x01 + 0x23 + 0x45 + 0x67 + 0x89 + 0xAB + 0xCD + 0xEF + 0x01 + 0x23 + 0x45 + 0x67 + 0x89 + 0xAB + 0xCD + 0xEF$$

$$Checksum = 0xFF - (Sum \mod 0x100) - \left\lfloor \frac{Sum}{0x100} \right\rfloor$$

$$Checksum = 0xFF - (0x78B \mod 0x100) - \left\lfloor \frac{0x78B}{0x100} \right\rfloor$$

$$Checksum = 0xFF - 0x8B - 0x07 = 0x6D$$

The UNLOCK command looks like this:

Header		Payload Key																Check-sum
Cmd	Select	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
0x0A	0x01	0x01	0x23	0x45	0x67	0x89	0xAB	0xCD	0xEF	0x01	0x23	0x45	0x67	0x89	0xAB	0xCD	0xEF	0x6D
PC → MCU																		

Depending on the result, the following response and behavior must be expected:

- The ROM/Flash memory is unlocked successfully:

Response
0x69
MCU → PC

After this response, all other commands are allowed.

- If the ROM/Flash memory cannot be unlocked because the saved Unlock Key is all 0:

Response
0x96
MCU → PC

After this response, serial commands are still handled.

- If the ROM/Flash memory cannot be unlocked because the wrong Unlock Key was transmitted:

Response
0x96
MCU → PC

After this response, no more commands are handled. The MCU must be reset.

[4] Further details about calibration header can be found in application note AN204773 “MCU Flash Programming and Boot-ROM Protocol”

3 Appendix

Further Information

Information about Cypress Microcontrollers can be found on the following Internet page:

http://www.cypress.com/cypress-microcontrollers

3.1 Related Documents

AN204773 - F²MC-16FX, MCU Flash Programming and Boot ROM Protocol

This application note shows flash programming and boot ROM mechanism.

4 Document History

Document Title: AN205551 - F²MC-16FX Family, Flash Security

Document Number: 002-05551

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	NOFL	08/31/2006	Initial release
			10/16/2006	Corrected UNLOCK command
			06/22/2007	Updated Configuration
			08/01/2007	Added memory map, flowchart for unlocking
			10/23/2007	Removed “internal” remark; added Satellite Flash Memory key to 2.3.1
			03/03/2010	Corrected read-value in case of secured Flash memory; updated naming of Flash memory; correct typos
*A	5074682	NOFL	03/07/2016	Migrated Spansion Application Note MCU-AN-300213-E-V15 to Cypress format
*B	5862797	AESATP12	08/24/2017	Updated logo and copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Videos | Blogs | Training | Components

Technical Support

cypress.com/support

All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709

© Cypress Semiconductor Corporation, 2006-2017. This document is the property of Cypress Semiconductor Corporation and its subsidiaries, including Spansion LLC ("Cypress"). This document, including any software or firmware included or referenced in this document ("Software"), is owned by Cypress under the intellectual property laws and treaties of the United States and other countries worldwide. Cypress reserves all rights under such laws and treaties and does not, except as specifically stated in this paragraph, grant any license under its patents, copyrights, trademarks, or other intellectual property rights. If the Software is not accompanied by a license agreement and you do not otherwise have a written agreement with Cypress governing the use of the Software, then Cypress hereby grants you a personal, non-exclusive, nontransferable license (without the right to sublicense) (1) under its copyright rights in the Software (a) for Software provided in source code form, to modify and reproduce the Software solely for use with Cypress hardware products, only internally within your organization, and (b) to distribute the Software in binary code form externally to end users (either directly or indirectly through resellers and distributors), solely for use on Cypress hardware product units, and (2) under those claims of Cypress's patents that are infringed by the Software (as provided by Cypress, unmodified) to make, use, distribute, and import the Software solely for use with Cypress hardware products. Any other use, reproduction, modification, translation, or compilation of the Software is prohibited.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS DOCUMENT OR ANY SOFTWARE OR ACCOMPANYING HARDWARE, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. To the extent permitted by applicable law, Cypress reserves the right to make changes to this document without further notice. Cypress does not assume any liability arising out of the application or use of any product or circuit described in this document. Any information provided in this document, including any sample design information or programming code, is provided only for reference purposes. It is the responsibility of the user of this document to properly design, program, and test the functionality and safety of any application made of this information and any resulting product. Cypress products are not designed, intended, or authorized for use as critical components in systems designed or intended for the operation of weapons, weapons systems, nuclear installations, life-support devices or systems, other medical devices or systems (including resuscitation equipment and surgical implants), pollution control or hazardous substances management, or other uses where the failure of the device or system could cause personal injury, death, or property damage ("Unintended Uses"). A critical component is any component of a device or system whose failure to perform can be reasonably expected to cause the failure of the device or system, or to affect its safety or effectiveness. Cypress is not liable, in whole or in part, and you shall and hereby do release Cypress from any claim, damage, or other liability arising from or related to all Unintended Uses of Cypress products. You shall indemnify and hold Cypress harmless from and against all claims, costs, damages, and other liabilities, including claims for personal injury or death, arising from or related to any Unintended Uses of Cypress products.

Cypress, the Cypress logo, Spansion, the Spansion logo, and combinations thereof, WICED, PSoC, CapSense, EZ-USB, F-RAM, and Traveo are trademarks or registered trademarks of Cypress in the United States and other countries. For a more complete list of Cypress trademarks, visit cypress.com. Other names and brands may be claimed as property of their respective owners.