AN205569

F²MC-8FX Family, MB95200H/210H Series FF70 Flash Usage

This application note describes how to use the dual operation flash (FF70) in the MB95F264K.

Contents

1 Introduction

This application note describes how to use the dual operation flash (FF70) in the MB95F264K.

2 Dual Operation Flash Structure

This chapter introduces the structure of FF70 flash.

2.1 General

With the Dual Operation Flash it is possible to erase and program/read flash sectors during code execution in the other flash bank.

2.2 Flash Memory

The flash memory in the MB95F264 is divided into two banks, which are divided into sectors as follows:

	Flash memory	CPU address
Lower bank	SA0 (2KB)	B000H B7FFH
	SA1 (2KB)	B800H BFFFH
Higher bank	SA2 (16KB)	C000H FFFFH

3 Dual Operation Flash Handling

This chapter introduces the handling for dual operation flash.

3.1 Flash Programming

Write data to given address of FLASH memory.

3.1.1 Command Sequence

Following sequence is needed for writing data to a sector:

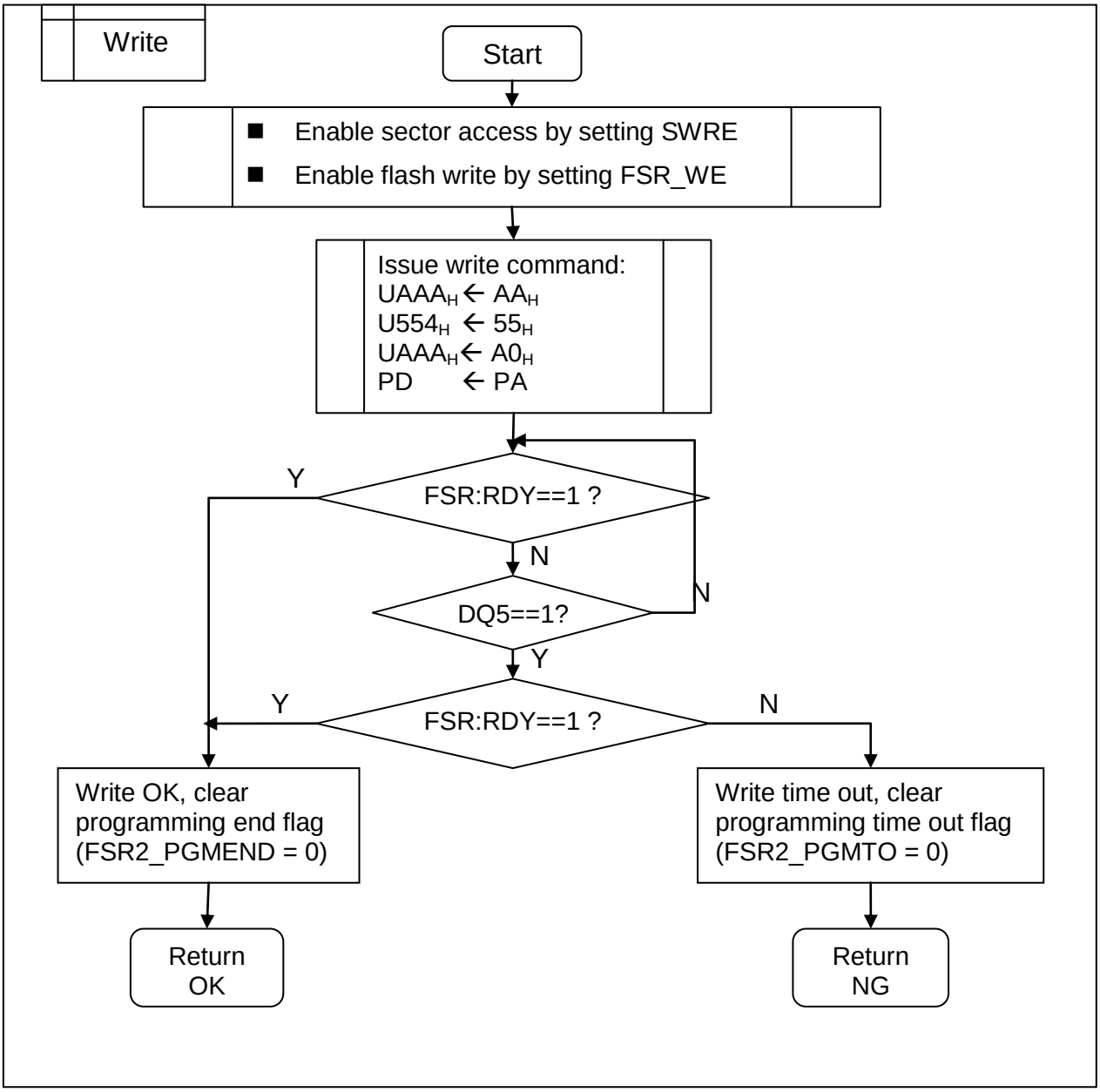
$UAAA_H$ ← AA_H

$U554_H$ ← 55_H

$UAAA_H$ ← $A0_H$

PD ← PA

- PA : Write address
- PD : Write data
- U : Upper 4 bits same as PA



3.1.2 Time Consumption

A typical flash byte write operation may cost about 21uS.

3.1.3 Sample Code

Below sample code shows how to write a data (0x5A) to address (0xB000).

```
_Write:
        SETB        0x73:0              ; Sector 0 (B000H~B7FFH) enable access
        SETB        0x72:1              ; FSR_WE = 1, write enable
        MOV         0xBAAA,     #0xAA
        MOV         0xB554,     #0x55
        MOV         0xBAAA,     #0xA0
        MOV         0xB000,     #0x5A   ; PD → PA
        NOP
        NOP
WrtieLoop:
        BBS         0x72:4,     Write_OK    ; FSR_RDY = 1 → Write OK
        MOV         A,@EP
        AND         A,#0x20             ; Check time out flag (DQ5) ?
        BZ          WriteLoop
        BBS         0x72:4,     Write_OK    ; Check again.
        ; Write Time Out
        CLRB        0x71:4              ; Clear programming time out flag
        MOV         A,      #NG         ; Return NG
        JMP         Write_End
Write_OK:
        CLRB        0x71:6              ; Clear programming end flag
        MOV         A,      #OK         ; Return OK
Write_End
        RET
```

Note: After the issue of a write command, wait for those two machine clock cycles to elapse (e.g. inserting NOP twice) before reading.

3.2 Flash Sector Erase

Erase only one or a number of given FLASH memory sectors

3.2.1 Command Sequence

Following sequence is needed for erasing a sector:

$UAAA_H \leftarrow AA_H$

$X554_H \leftarrow 55_H$

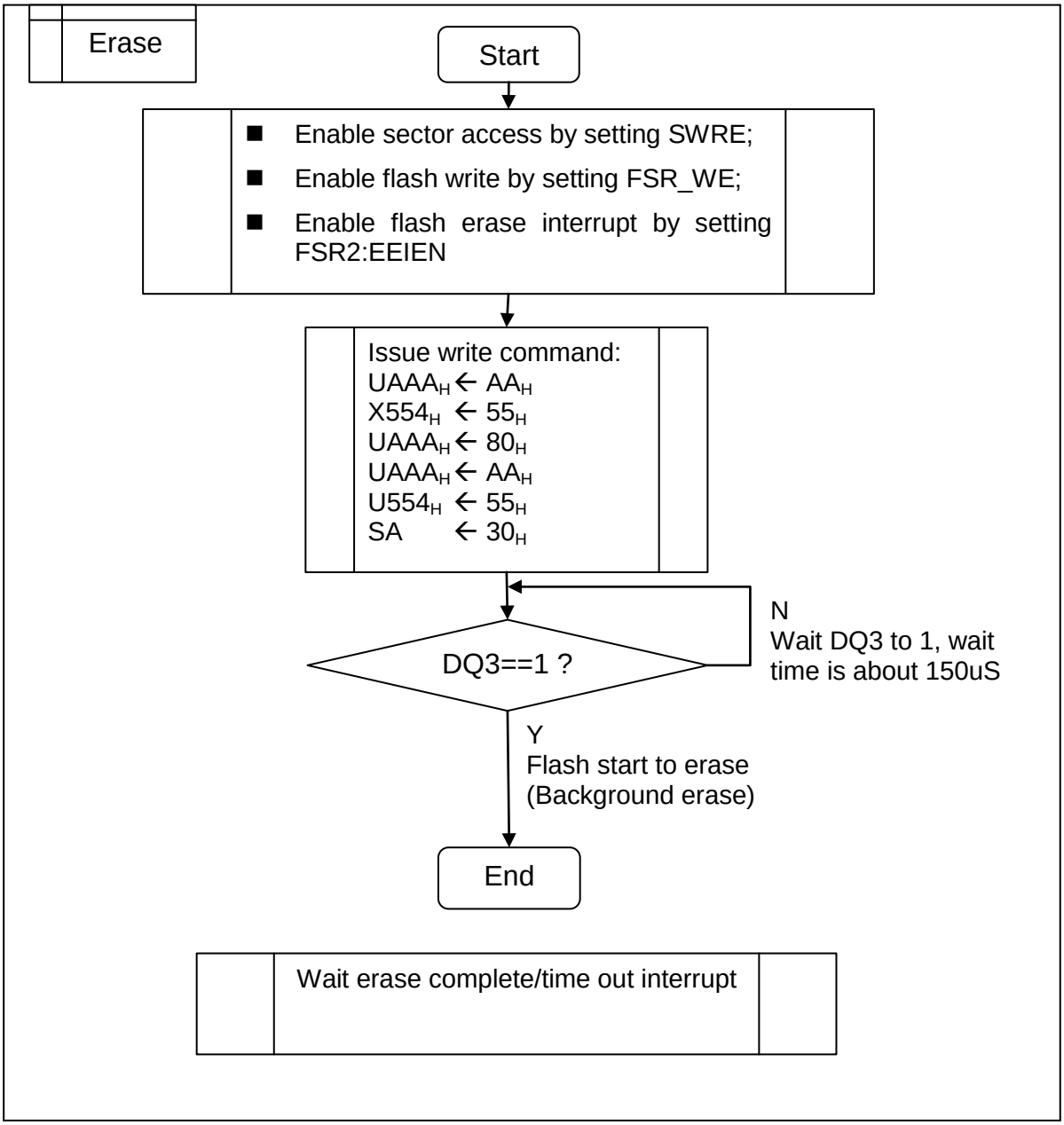
$UAAA_H \leftarrow 80_H$

$UAAA_H \leftarrow AA_H$

$U554_H \leftarrow 55_H$

$SA \leftarrow 30_H$

- SA : Sector address (specify arbitrary one address in sector)
- X : Arbitrary address
- U : Upper 4 bits same as PA


Erase
Start
Enable sector access by setting SWRE;
Enable flash write by setting FSR_WE;
Enable flash erase interrupt by setting FSR2:EEIEN
Issue write command:
UAAA$_H$ ← AA$_H$
X554$_H$ ← 55$_H$
UAAA$_H$ ← 80$_H$
UAAA$_H$ ← AA$_H$
U554$_H$ ← 55$_H$
SA ← 30$_H$
DQ3==1 ?
N
Wait DQ3 to 1, wait time is about 150uS
Y
Flash start to erase (Background erase)
End
Wait erase complete/time out interrupt

3.2.2 Time Consumption

A typical flash sector erase operation may cost about 500mS.

3.2.3 Sample Code

Blow sample code shows how to erase sector 0 (0xB000).

```
_Erase:
        SETB    0x73:0              ; Sector 0 (B000H~B7FFH) enable access
        SETB    0x72:1              ; FSR_WE = 1, write enable
        SETB    0x71:3              ; FSR2_EEIEN = 1, enable erase interrupt
        MOV     0xBAAA,     #0xAA
        MOV     0xB554,     #0x55
        MOV     0xBAAA,     #0x80
        MOV     0xBAAA,     #0xAA
        MOV     0xB554,     #0x55
        MOV     0xB000,     #0x30   ; 30 → SA
        NOP
        NOP
CHECK_DQ3:
        MOV     A,      0xB000
        AND     A,      #0x08
        BZ      CHECK_DQ3           ; wait DQ3 = 1, indicate flash start to erase
        …
        ; Background erase start
```

Note:

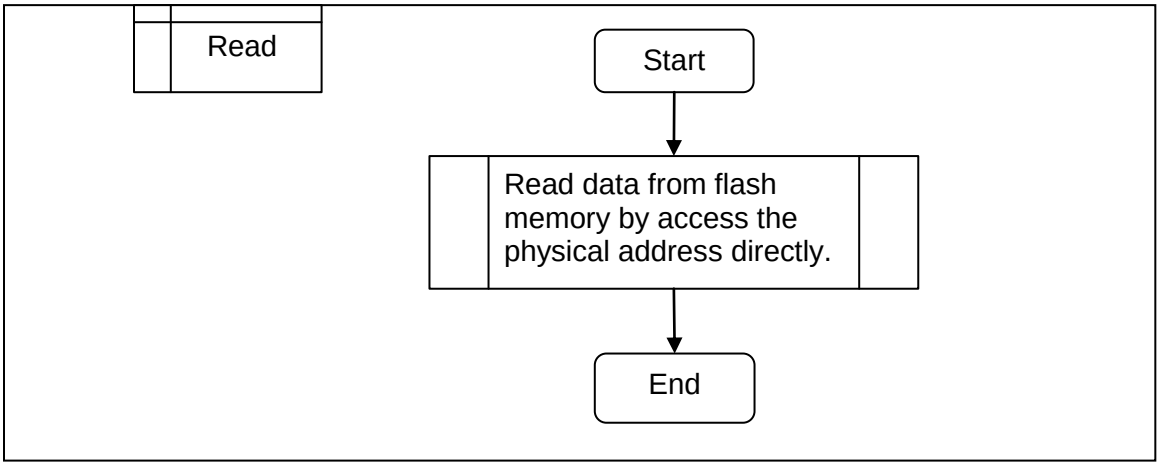
1. After the issue of an erase command, wait for those two machine clock cycles to elapse (e.g. inserting NOP twice) before reading
2. After the issue of an erase command, wait flash memory really start erasing (DQ3 = 1), the wait time is about 150uS.

3.3 Flash Read

Read one data from given address from FLASH memory.

3.3.1 Command Sequence

Read operation can be realized by accessing flash memory directly if another sector is not programming/erasing.



3.3.2 Time Consumption

The time consumption of read operation is determined by the machine clock.

3.3.3 Sample Code

Below sample code shows how to read a data from 0xB000.

```
_Read:
        MOV   A,      0xB000        ; Read data from 0xB000
        RET   A                     ; Return read data
```

3.4 Flash Erase Suspend

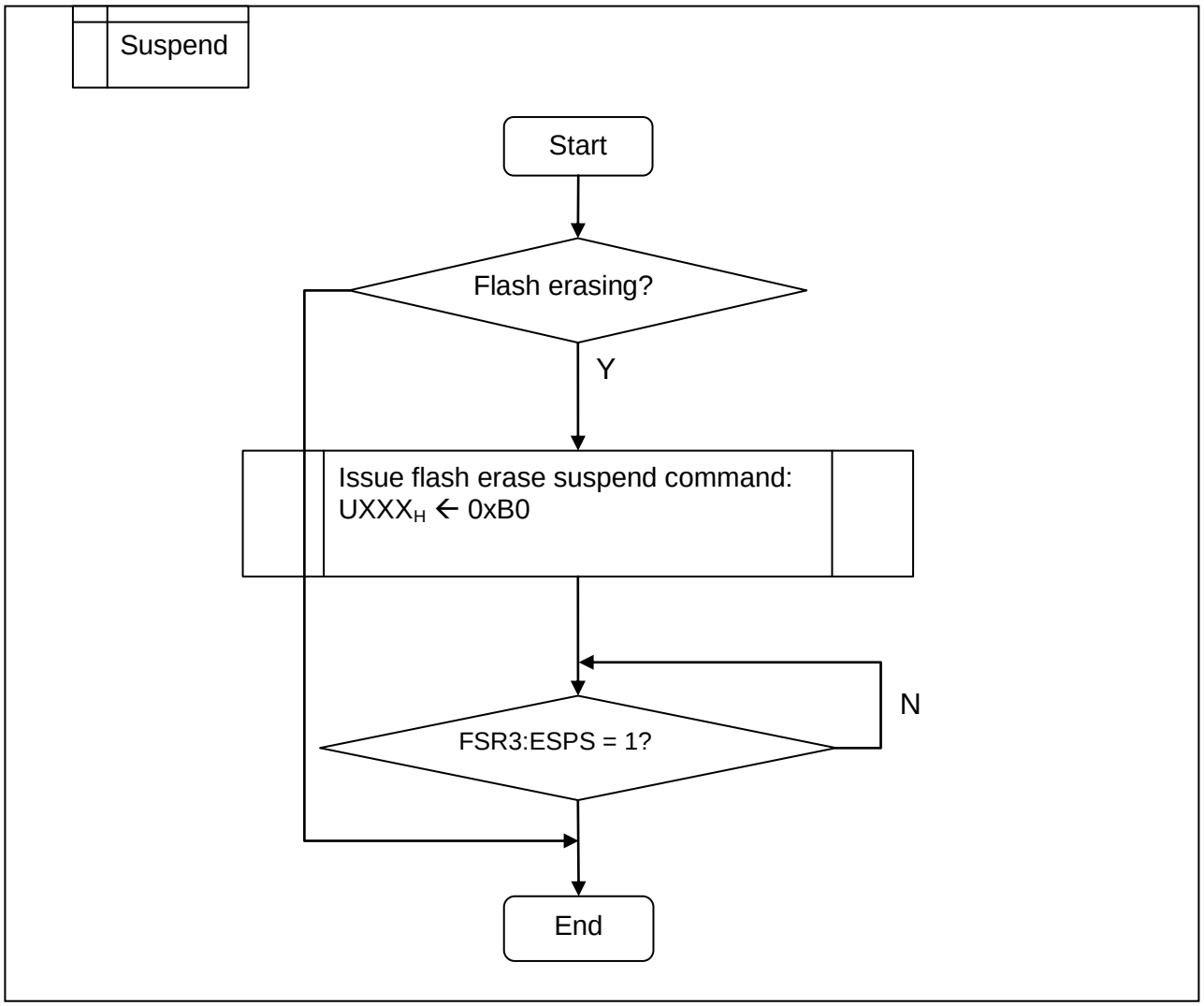
Suspend currently erasing sector to realize read/write operation in another sector.

3.4.1 Command Sequence

Following sequence is needed for suspending currently erasing sector:

$UXXX_H \leftarrow B0_H$

- X : Arbitrary address
- U : Upper 4 bits same as PA



3.4.2 Time Consumption

One bit of the flash status register (FSR3_ESPS) may indicate flash erase suspend operation if perform successfully while the suspend command is issued. Wait FSR3_ESPS to 1 is necessary after the command is sent; the wait time is about 2uS.

3.4.3 Sample Code

Please refer to Section 3.5.3 for details.

3.5 Flash Erase Resume

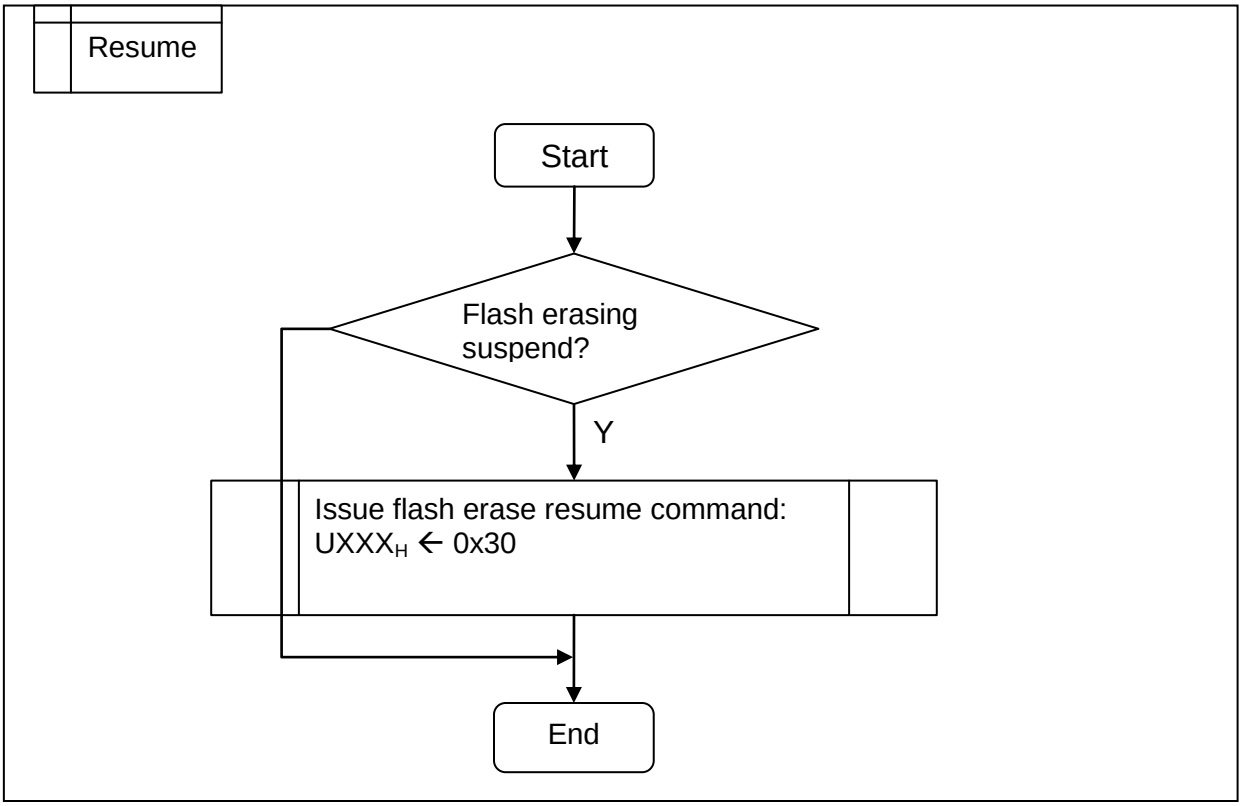
Resume a sector erase command which was previously suspended.

3.5.1 Command Sequence

Following sequence is needed for resuming currently suspended sector:

$UXXX_H \leftarrow 30_H$

- X : Arbitrary address
- U : Upper 4 bits same as PA



3.5.2 Time Consumption

Flash erase will be performed again immediately while the resume command is issued if there was previously suspended.

3.5.3 Sample Code

Below sample code shows the background erase → suspend → write → read → resume operation flow.

```
        …
        CALL   _Erase                   ; Background erase (sector 1: 0xB800~0xBFFF)
        ; Suspend current erasing sector
Suspend:
        MOV    0xB800,          #0xB0   ; Send erase suspend command to erasing flash
        NOP
        NOP
Suspend_Wait:
        BBC    0x74:3,          Suspend_Wait          ; Wait sector suspend flag to 1
        ;Write a data to 0xB800 (sector 0: 0xB000 ~ 0xB7FF)
        CALL   _Write
        ; Read a data from 0xB800
        CALL   _Read
        ; Resume previously suspended erase
Resume:
        MOV    0xB800,          #0x30   ; Send erase resume command to suspended flash
        …
```

Additional Information:

For more Information on MB95200 series products, visit the following website:

http://www.cypress.com/MB95200

4 Document History

Document Title: AN205569 - F²MC-8FX Family, MB95200H/210H Series FF70 Flash Usage

Document Number: 002-05569

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	-	CHMA	09/17/2009	Initial release
*A	5267476	CHMA	10/06/2016	Migrated Spansion Application Note MCU-AN- 500049-E-10 to Cypress format
*B	5843287	AESATMP9	08/03/2017	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

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