Freescale Semiconductor
Application Note

Document Number: AN3521
Rev. 0, 09/2007

Using the ColdFire Flash Module with the MCF521x ColdFire Microcontroller

by: Paolo Alcantara
RTAC Americas

1 Introduction

This document is a quick reference to get the FLASH module configured for the MCF521x microcontroller. It includes basic functional description and configuration options to give the user a better understanding of how the flash module works. This application note provides an example that demonstrates how to configure the flash module for the MCF521x microcontroller. The examples mentioned may be modified to suit the specific needs for any application.

2 MCF521x Flash Features

The ColdFire FLASH module (CFM) is a nonvolatile memory (NVM) module for integration with a CPU. The CFM provides 256 Kbytes of 32-bit flash memory serving as electrically erasable and programmable, nonvolatile memory. The flash memory is ideal for programming and data storage for single-chip applications, allowing field reprogramming without requiring external programming voltage sources.

Contents

freescale semiconductor

2.1 Description

Main features include the following:

- 256 Kbytes of 32-bit flash memory (the MCF5211 has only 128 Kbytes).
- Automated program, erase, and verify operations
- Single power supply for program and erase operations
- Software programmable interrupts on command completion, access violations, or protection violations
- Fast page erase operation
- Fast word program operation
- Protection scheme to prevent the flash memory of accidental programming or erasing
- Access restriction control for supervisor/user and data/instruction operations
- Security feature to prevent unauthorized access to the flash memory

3 FLASH Module Explanation

This document demonstrates how to configure and use the FLASH module in an easy way. It allows programmers to include it quickly in their projects. The example code shows how to write a longword to the flash memory using the M5213EVB board. The code also includes a routine that copies a function from ROM to RAM and executes it in the volatile memory. The ROM cannot be modified while there is code reading from there.

Internally, the flash is divided in the following ways:

- Logical block — there is one block, (256 KBytes in 5212 and 5213, 128 KBytes in 5211). It is not possible to read from any block while the same block is being erased, programmed, or verified
- Physical block — there are two physical blocks inside each logical block
- Sector — portion of a logical block. There are 32 sectors in an entire logical block, each one 8 Kbytes, also known as a logical sector)
- Logical Page — each page is 2 Kbytes

The entire logical block is divided in sectors according to Figure 1. Each one can be programmed to:

- Protect against program and erase
- Declare as supervisor address space only
- Place as data or data and instruction address space

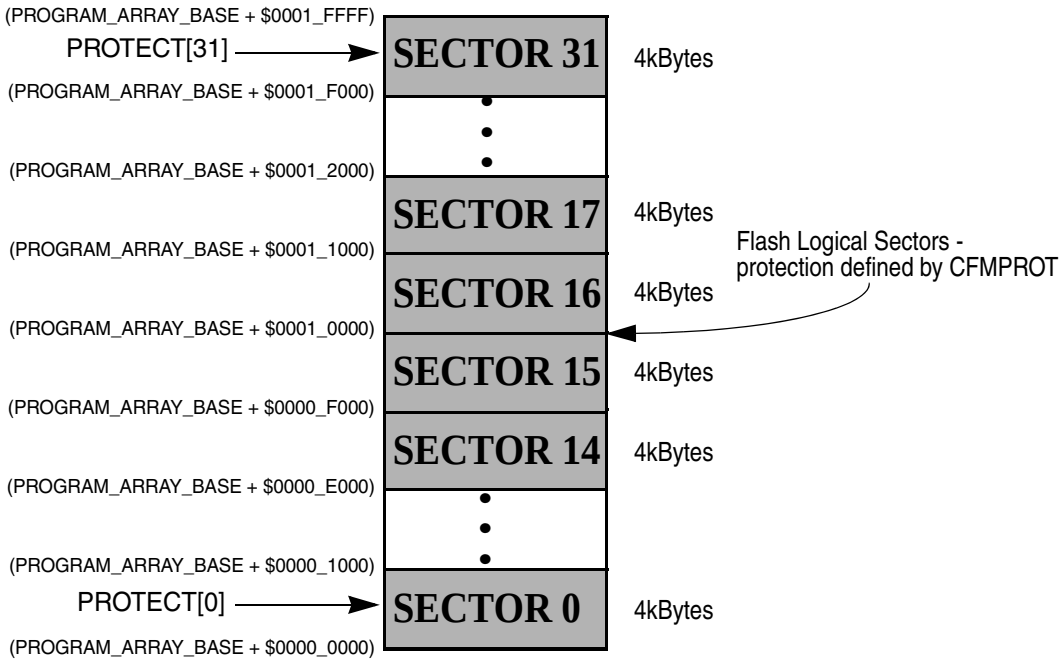

Figure 1. Sector Division of FLASH

The following actions can be used to modify flash during runtime:

- Word Program: program a 32-bit word
- Page Erase: erase a flash logical page
- Mass Erase: erase the entire flash memory. All flash memory protection must be disabled
- Blank Check: verify that the entire flash memory is erased
- Page Erase Verify: verifies that a flash logical page is erased

3.1 Example: Case Study for the CFM Module

After programming code in ROM, it shows messages through a serial port (UART0), and a menu of some longwords for testing purposes to write into flash:

```
[0] 0xdeadbeef
[1] 0xcafebabe
[2] 0xbabadada
[3] 0xdecafbad
[4] 0xfeedface
[5] 0xdeadc0de
[6] 0xabadbabe
[7] 0xfeedbabe
[8] 0x00000000
[9] 0xFFFFFFFF
```

After this, the serial terminal waits for pressing associated number with longword (see longwords above), and the program does the following:

- Read flash address 0x6000: to check whether the last write was successful
- Execute an erase page (2 Kbytes) starting in flash address 0x6000: it writes in flash during runtime and only clears bits but does not set them. An erase has to be executed to allow a new longword in that space by clearing bits only
- Write flash address 0x6000. The word is verified in next reading space

After writing the word in flash, if code is downloaded to the flash, the M5213EVB board can receive a power-on-reset to check if longword write selection is executed successfully. In another case, the starting menu appears again.

3.2 Example: CFM Clock Configuration

Equation 1 shows the formula applied to maintain flash frequency within $150kHz \le f_{CFM} \le 200kHz$, as internal flash specifications states.

$$f_{CFM} = \frac{f_{CORE}}{(2 * DIV[5:0] * (1 + (7 * PRODIV8)))}$$

Clock Frequency ***Eqn. 1***

- CFMCLKD [DIV[5:0]]: chosen such Equation 1 is valid
- CFMCLKD[PRODIV8]: set in case fCORE is greater than 12.8 MHz

3.3 Example: Configuring CFM Module

After a longword is selected Section 3.1, “Example: Case Study for the CFM Module,” the program asks for another option. The most important part of the code is related to copy a function from ROM to RAM and then executed in RAM. In any case longword selected is deleted.

There are two ways to copy a function from ROM to RAM:

1. Compiler independent code (code named CFMSoftwareDEMO):
   a) Declares a variable that stores the entire function from the ROM. This is reserved space in the RAM to be modified during runtime
   b) Declares a function pointer in the RAM memory to point to the variable that stores the function in the RAM. A cast from char to function pointer is necessary.
   c) Copies entire function space from the ROM to the variable in the RAM using memcopy from stdio.h library. This has to be stored only one time.
   d) Executes a function that points to a variable that stores a function in the RAM. This way it executes the function in the RAM, and after returns to the flash to continue executing code there.
   e) If only one function is copied from ROM to RAM, the lcf file does not need to be modified in this case.
2. CodeWarrior stile (code named CFMSofwareDEMO-LCF):
   a) Declares a memory segment inside the ROM.lcf file from CodeWarrior
   b) After that it declares a segment that holds the function in the RAM memory

c) From the code, it declares a section with the same name as the segment

d) Writes your function inside this section

4 Register Configuration

This section describes the registers configuration that affects the CFM module and the code lines used for this purpose.

4.1 CFM Clock Divider Register

The following code lines are used to configure the CFMCLKD, Figure 2, to assign the CFM frequency.

```
/* Select FLASH frequency to 200 kHz. Section 15.4.3.1 MCF5213RM */
MCF_CFM_CFMCLKD = MCF_CFM_CFMCLKD_DIV(25)
                  | MCF_CFM_CFMCLKD_PRDIV8;.
```

IPSBAR Access: User read/write

Offset: 0x1D_0002 (CFMCLKD)

	7	6	5	4	3	2	1	0
R	DIVLD	PRDIV8	DIV					
W								
Reset:	0	0	0	0	0	0	0	0

Figure 2. CFM Clock Divider Register

4.2 CFM Module Configuration Register

To configure the CFMMCR, the next code lines are used, the changes performed in the register are shown in Figure 3.

```
/* clear the lock bit */
MCF_CFM_CFMMCR &= ~MCF_CFM_CFMMCR_LOCK;
```

IPSBAR Access: User read/write

Offset: 0x1D_0000 (CFMMCR)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	0	0	0	0	LOCK	PVIE	AEIE	CBEIE	CCIE	KEYACC	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3. CFM Module Configuration Register

4.3 CFM Clock Select Register

To write the values shown in Figure 4, Figure 5, and Figure 6, in registers, the program uses these lines:

```
/* clear sector protection registers if set from reset */
MCF_CFM_CFMPROT = 0x0;
MCF_CFM_CFMSACC = 0x0;
MCF_CFM_CFMDACC = 0x0;
```

IPSBAR Access: User read/write

Offset: 0x1D_0010 (CFMPROT)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PROTECT															
Reset	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PROTECT															
Reset	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]

1 Reset state loaded from flash configuration field during reset.

Figure 4. CFM Protection Register

IPSBAR Access: User read/write

Offset: 0x1D_0014 (CFMSACC)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SUPV															
Reset	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SUPV															
Reset	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]

1 Reset state loaded from flash configuration field during reset.

Figure 5. CFM Supervisor Access Register

IPSBAR Access: User read/write

Offset: 0x1D_0018 (CFMDACC)

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DACC															
Reset	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DACC															
Reset	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]	F[1]

1 Reset state loaded from flash configuration field during reset.

Figure 6. CFM Data Access Register

5 Configuration Summary

The following steps are needed to modify the flash content. Reading flash memory can be performed directly without any CFM register configuration.

1. Clear the CFMMCR[LOCK] bit
2. Set the CFMCLKD with the calculated value
3. Clear registers: CMFPROT, CFMSACC, CFMDACC
4. Run the following steps from the RAM
5. Wait until CFMUSTAT[CBEIF] flag is set to continue
6. Write a longword from 0x0400 0000 plus IPSBAR in addresses divisible by 4 and within flash boundaries. The memory space to use is 64 k @ 32 bits in MCF5212 and MCF5213. It is 32 k @ 32 bits in MCF5211. This longword value is useless, but mandatory if you want to execute an erase operation
7. Write in the MCFCMD register the command code desired
8. Set the CFMUSTAT[CBEIF] bit to clear flag
9. Wait until the CFMUSTAT[CCIF] flag is set to continue
10. If you want to write more longwords, then repeat step 4 down

6 Configuration Notes

The following details are considered important when configuring and using the CFM:

- Only 32-bit write operations are allowed to the flash memory space. Byte and halfword write operations to the flash memory space result in a cycle termination transfer error.
- Avoid writing longword or erasing page to areas used by code.
- It is not possible to read from any block while the same logical block is being erased, programmed, or verified.
- Always maintain flash write frequency around $150kHz \leq f_{CFM} \leq 200kHz$
- Program and erase command execution time increases proportionally with the period of FCLK.
- As active commands are immediately aborted when the MCU enters stop mode, it is strongly recommended that the user does not execute the stop instruction during program and erase operations.
- After erasing a longword, using page or mass erase, the value is 0xFFFFFFFF. The programmer can clear bits in the same longword, but cannot set bits, until an erase command is executed previously.
- Security information that allows the MCU to prevent intrusive access to the flash memory is stored in the flash configuration field. The flash configuration field is composed of 24 bytes of reserved memory space within the flash memory, which contains information that determines the CFM protection and access restriction scheme out of reset. This area is between FLASHBAR+0x400 to FLASHBAR+0x417. This restriction is included at the bottom of the file vectors.s in the software example.

7 Conclusion

The CFM module is a non-volatile-memory that is not only used as code storage. It can also be used as external event storage, keeping ADC measures, or an external event to be used later, even if the microcontroller is turned off for a while. All this information can be stored or changed during runtime.

7.1 Considerations and References

Find the newest software updates and configuration files for the MCF521x on the Freescale Semiconductor home site www.freescale.com.

This application note considers MCF5211, MCF5212 and MCF5213 devices.

The M5213EVB development board employed CFM software demo.

For more information on FLASH module refer to MCF5213 ColdFire Integrated Microcontroller Reference Manual. Rev 2.0 at www.freescale.com

The CFMSoftwareDemo software was developed in CodeWarrior for ColdFire v6.4.

Download the source files for CFMSoftwareDemo software (CFMSoftwareDemo.zip) from www.freescale.com.

THIS PAGE IS INTENTIONALLY BLANK

How to Reach Us:

Home Page:
www.freescale.com

Web Support:
http://www.freescale.com/support

USA/Europe or Locations Not Listed:
Freescale Semiconductor, Inc.
Technical Information Center, EL516
2100 East Elliot Road
Tempe, Arizona 85284
+1-800-521-6274 or +1-480-768-2130
www.freescale.com/support

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
www.freescale.com/support

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

Document Number: AN3521
Rev. 0
09/2007

freescale™
semiconductor