Application Report

SPNA148–May 2013

Advanced F021 Flash API Erase/Program Usage

John R. Hall

ABSTRACT

This application report gives the user of the F021 Flash API the ability to write program and erase code in a way that is optimized to their applications.

Contents

List of Figures

List of Tables

1 Introduction

The 65nm Flash Technology used on Hercules devices generally requires the use of the F021 Flash API to do program and erase operations. Since these functions need to cover all possible uses, they are generic, and not the most efficient functions available for all applications. Therefore, this document is designed to allow the user to write erase and program functions that are optimized for their application needs. The F021 Flash API is still required for Flash initialization and bank selection. For recommended erase and program operation flows, see the *F021 Flash API Reference Guide* (SPNU501).

NOTE: ***This document applies to all versions of the F021 Flash API starting with v2.00.00.***

2 Flash Registers Used

In addition to the registers listed in the *F021 Flash Module* chapter in the device-specific technical reference manual, Table 1 lists registers specifically needed for program and erase operations.

Table 1. Additional Flash Control Registers

Offset	Acronym	Register Description	Section
FFF8 7110	FADDR	The absolute address of the data that the CPU would use to access the location.	Section 2.1
FFF8 7120	FWPWRITE0	WPDATA[31: 00]	Section 2.2
FFF8 7124	FWPWRITE1	WPDATA[63:32]	Section 2.2
FFF8 7128	FWPWRITE2	WPDATA[95:64]	Section 2.2
FFF8 712C	FWPWRITE3	WPDATA[127:96]	Section 2.2
FFF8 7130	FWPWRITE4	WPDATA[159:128]	Section 2.2
FFF8 7134	FWPWRITE5	WPDATA[191:160]	Section 2.2
FFF8 7138	FWPWRITE6	WPDATA[223:192]	Section 2.2
FFF8 713C	FWPWRITE7	WPDATA[255:224]	Section 2.2
FFF8 7140	FWPWRITE_ECC	Contains the ECC bits for the FWPWRITE7:0 registers	Section 2.3
FFF8 720C	FSM_COMMAND	The command to be executed	Section 2.4
FFF8 72B4	FSM_EXECUTE	Execute the command in the FSM_COMMAND register	Section 2.5

2.1 Flash Address Register (FADDR)

This register contains the absolute address of the location to be programmed, or area to be erased. For FSM operations, this register can contain the main Flash, data Flash or Customer OTP addresses. The ECC regions at 0xF01x_xxxx or 0xF02x_xxxx are illegal regions because the ECC programs must be done using the address of the data that ECC protects and the FWPWRITE_ECC register. FADDR is used by Program and Erase commands.

The Flash address register is shown in Figure 1 and described in Table 2.

Figure 1. Flash Address Register (FADDR) [offset = FFF8 7110h]

31 … 0
ADDR
RWP

LEGEND: -*n* = value after reset, R=Read, WP=Write in Privilege Mode with FLOCK=any value

Table 2. Flash Address Register (FADDR) Field Descriptions

Bit	Field	Value	Description
31:0	ADDR	CPU address to be operated on by the command	Absolute address of the data that the CPU would use to access the location.

2.2 Flash Wide Programming Write Data Register (FWPWRITE0-7)

The Flash wide programming write data register is shown in Figure 2 through Figure 9 and described in Table 3.

Figure 2. Flash Wide Programming Write Data Register (FWPWRITE0) [offset = FFF8 7120h]

31	0
WPDATA[31:00]	
RWP	

LEGEND: -*n* = value after reset, R=Read, WP=Write in Privilege Mode

Figure 3. Flash Wide Programming Write Data Register (FWPWRITE1) [offset = FFF8 7124h]

31	0
WPDATA[63:32]	
RWP	

LEGEND: -*n* = value after reset, R=Read, WP=Write in Privilege Mode

Figure 4. Flash Wide Programming Write Data Register (FWPWRITE2) [offset = FFF8 7128h]

31	0
WPDATA[95:64]	
RWP	

LEGEND: -*n* = value after reset, R=Read, WP=Write in Privilege Mode

Figure 5. Flash Wide Programming Write Data Register (FWPWRITE3) [offset = FFF8 712Ch]

31	0
WPDATA[127:96]	
RWP	

LEGEND: -*n* = value after reset, R=Read, WP=Write in Privilege Mode

Figure 6. Flash Wide Programming Write Data Register (FWPWRITE4) [offset = FFF8 7130h]

31	0
WPDATA[159:128]	
RWP	

LEGEND: -*n* = value after reset, R=Read, WP=Write in Privilege Mode

Figure 7. Flash Wide Programming Write Data Register (FWPWRITE5) [offset = FFF8 7134h]

31	0
WPDATA[191:160]	
RWP	

LEGEND: -*n* = value after reset, R=Read, WP=Write in Privilege Mode

Figure 8. Flash Wide Programming Write Data Register (FWPWRITE6) [offset = FFF8 7138h]

31	0
WPDATA[223:192]	
RWP	

LEGEND: -*n* = value after reset, R=Read, WP=Write in Privilege Mode

Figure 9. Flash Wide Programming Write Data Register (FWPWRITE7) [offset = FFF8 713Ch]

31	0
WPDATA[255:224]	
RWP	

LEGEND: -*n* = value after reset, R=Read, WP=Write in Privilege Mode

Table 3. Flash Wide Programming Write Data Register (FWPWRITE0) Field Descriptions

Bit	Field	Value	Description
255:0	WPDATA [255:0]	Data to be programmed	This register contains the data to program into the flash memory. Each write will set data bits and another internal ***byte enable bit*** for each byte to be written by the FSM. The FWPWRITE register is set to all ones after writing to the FADDR register or when the FSM completes an operation. For 128-144-bit banks, only FWPWRITE0-3 are used during programming. FWPWRITE4-7 are unused. For 64-72-bit banks, use only FWPWRITE0-1 are used during programming. Unused bits can be written and read but they will not be part of the bank operations. **NOTE:** Do not write to this register while a FSM operation is active.

2.3 Flash Wide Programming Write Data ECC Register (FWPWRITE_ECC)

The Flash wide programming write ECC register is shown in Figure 10 through Figure 12 and described in Table 4.

Figure 10. Flash Wide Programming Write Data ECC Register - 288-Bit Bank (FWPWRITE_ECC) [offset = FFF8 7140h]

31 — 24	23 — 16	15 — 8	7 — 0
ECC for Bytes 7:0	ECC for Bytes 15:8	ECC for Bytes 23:16	ECC for Bytes 31:24
RWP			

LEGEND: -*n* = value after reset, R=Read, WP=Write in Privilege Mode with FLOCK=any value

Figure 11. Flash Wide Programming Write Data ECC Register - 144-Bit Bank (FWPWRITE_ECC) [offset = FFF8 7140h]

31 — 24	23 — 16	15 — 0
ECC for Bytes 7:0	ECC for Bytes 15:8	Reserved
RWP		RWP=FFFF_FFFFh

LEGEND: -*n* = value after reset, R=Read, WP=Write in Privilege Mode with FLOCK=any value

Figure 12. Flash Wide Programming Write Data ECC Register - 72-Bit Bank (FWPWRITE_ECC) [offset = FFF8 7140h]

31 — 24	23 — 0
ECC for Bytes 7:0	Reserved
RWP	RWP=FFFF_FFFFh

LEGEND: -*n* = value after reset, R=Read, WP=Write in Privilege Mode with FLOCK=any value

Table 4. Flash Wide Programming Write Data ECC Register (FWPWRITE_ECC) Field Descriptions

Bit	Field	Value	Description
31:0	ECC [31:0]	ECC Data to be programmed	This register contains the ECC bits for the FWPWRITE7:0 registers. The location of the ECC bits corresponding to the data depends on the device configuration. The three supported configurations are shown above. Unused bits can be written and read but they will not be part of the bank operations. This register is an extension of the FWPWRITE register. Do not write to this register while a FSM operation is active. This register is set to all ones just like the FWPWRITEx registers.

2.4 Flash State Machine Command Register (FSM_COMMAND)

The Flash state machine command register is shown in Figure 13 and described in Table 5.

Figure 13. Flash State Machine Command Register (FSM_COMMAND) [offset = FFF8 720Ch]

31 … 6	5 … 0
Reserved	FSM_CMD
R-0	RWP

LEGEND: -*n* = value after reset, R=Read, WP=Write in Privilege Mode

Table 5. Flash State Machine Command Register (FSM_COMMAND) Field Descriptions

Bit	Field	Value	Description
31:6	Reserved	0	These are reserved bits
5:0	FSM_CMD		Refer to the supported commands listed in Appendix A of *F021 Flash API Reference Guide* (SPNU501). Writes to this register are blocked if the state machine is busy. The only exception is to write the suspend command.

2.5 Flash State Machine Command Execute Register (FSM_EXECUTE)

The Flash state machine command execute register is shown in Figure 14 and described in Table 6.

Figure 14. Flash State Machine Command Execute Register (FSM_EXECUTE) [offset = FFF8 72B4h]

31 … 20	19 … 16
Reserved	SUSPEND_NOW
R-0	RWP-1010

15 … 5	4 … 0
Reserved	FSMEXECUTE
R-0	RWP-01010

LEGEND: -*n* = value after reset, R=Read, WP=Write in Privilege Mode

Table 6. Flash State Machine Command Register (FSM_COMMAND) Field Descriptions

Bit	Field	Value	Description
31:20	Reserved	0	These are reserved bits
19:16	SUSPEND_NOW	0101	Writing a 5 to this register will suspend the Program or Erase sector command. Writes to this register are ignored unless the FSM is busy with one of these commands. This field resets to '1010' when the FSM enters the standby state where it waits for the next command. It will not retain the written value for long. The Bank erase command and every command except those two commands above are not suspendable and will ignore the suspend command. A suspend operation will exit the FSM with the proper setup and hold times for the various modes.
15:5	Reserved	0	These are reserved bits
4:0	FSMEXECUTE	01010	To invoke a FSM command this register must be written with a value of 10101. This register is reset back to 01010 by the FSM upon completion of the command operation. Writes to this register are ignored while the FSM is busy. All others registers must be set up before writing to this command. Writing a 15h to this field is the last step to starting an FSM operation.

3 Example Usage

The following examples show some typically use cases in writing code using the Flash Memory Controller registers. The example code segments make use of the following framework code. For API functions used in these examples, see the *F021 Flash API Reference Guide* (SPNU501). For proper initialization of the device prior to any Flash operations, see the device-specific initialization document. All FMC register writes require the device to be in a privilege state.

```
#include “F021.h”

int main (void)
{
    Fapi_StatusType oReturnCheck = Fapi_Status_Success;
    FwpWriteByteAccessorType  * oFwpWriteByteAccessor    = FWPWRITE_BYTE_ACCESSOR_ADDRESS;
    FwpWriteByteAccessorType  * oFwpWriteEccByteAccessor = FWPWRITE_ECC_BYTE_ACCESSOR_ADDRESS;
    FwpWriteDWordAccessorType * oFwpWriteDWordAccessor   = FWPWRITE_DWORD_ACCESSOR_ADDRESS;

                uint8 au8MainDataBuffer[16] = {0x78, 0x17, 0x19, 0x2E, 0x0A, 0xB9, 0x11, 0x70,
0x5F, 0xC1, 0x9C, 0xFD, 0x54, 0x51, 0xED, 0x86};
    uint32 u32Index;

                /*
                            For proper initialization of the device prior to any Flash
operations,
                            see the device-specific initialization document.

                            Assumes, unless otherwise noted, device has 144bit wide Flash Banks.
    */

    oReturnCheck = Fapi_initializeFlashBanks(180);  /* Example code is assuming operating
frequency of 180 MHz */

    if((oReturnCheck == Fapi_Status_Success) &&
       (FLASH_CONTROL_REGISTER->FmStat.FMSTAT_BITS.Busy != Fapi_Status_FsmBusy))
    {
       oReturnCheck = Fapi_setActiveFlashBank(Fapi_FlashBank0);

       /* Place specific example code here */


       /* Wait for FSM to finish */
       while(FLASH_CONTROL_REGISTER->FmStat.FMSTAT_BITS.Busy == Fapi_Status_FsmBusy);

          /* Check the FSM Status to see if there were no errors */
       if (FLASH_CONTROL_REGISTER->FmStat.u32Register != 0)
       {
           /* Put Error handling code here */
       }
    }
}
```

3.1 Typical Command Execution Flow

Figure 15 shows the typical flow for issuing an erase or program command.

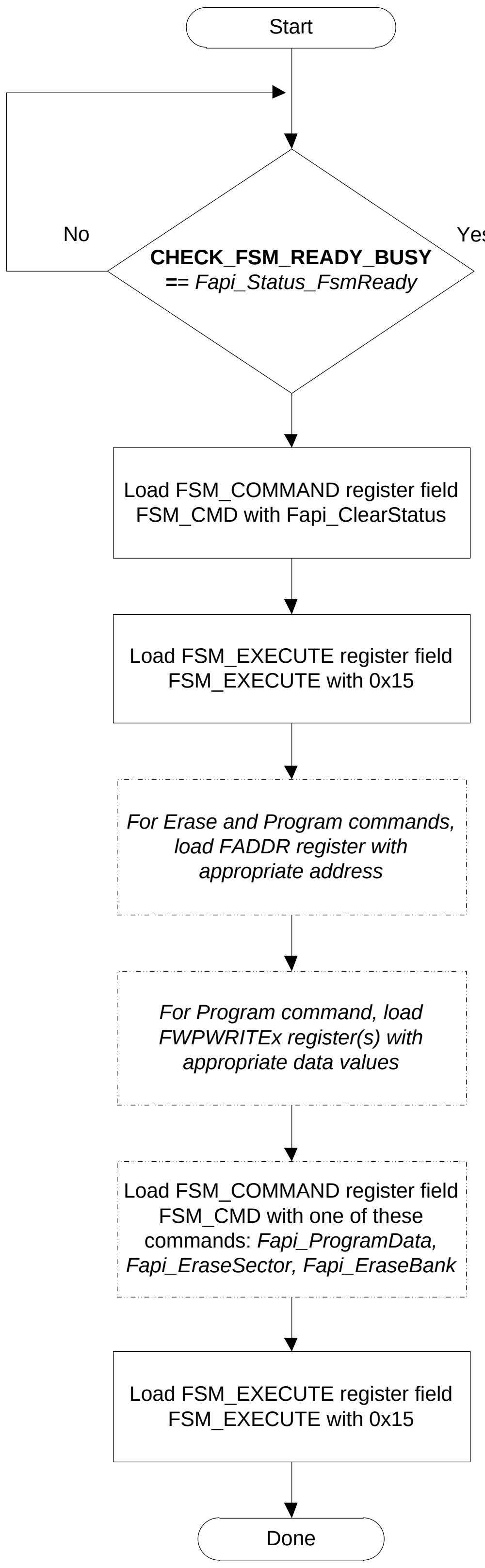
Start
No
Yes
CHECK_FSM_READY_BUSY == *Fapi_Status_FsmReady*
Load FSM_COMMAND register field FSM_CMD with Fapi_ClearStatus
Load FSM_EXECUTE register field FSM_EXECUTE with 0x15
For Erase and Program commands, load FADDR register with appropriate address
For Program command, load FWPWRITEx register(s) with appropriate data values
Load FSM_COMMAND register field FSM_CMD with one of these commands: *Fapi_ProgramData, Fapi_EraseSector, Fapi_EraseBank*
Load FSM_EXECUTE register field FSM_EXECUTE with 0x15
Done

Figure 15. Recommended Command Execution Flow

3.2 Programming Operations

3.2.1 Programming a Single Byte

The following code segement will write a single byte to the address specified.

```
        FLASH_CONTROL_REGISTER->Fbprot.u32Register = 1U;  /* Disable Level 1 Protection */

    /* Enable all sectors of current bank for erase and program.  For EEPROM banks with more
than 16 sectors, this must be 0xFFFF */
    FLASH_CONTROL_REGISTER->Fbse.u32Register = 0xFFFF;

    FLASH_CONTROL_REGISTER->Fbprot.u32Register = 0U;  /* Enable Level 1 Protection */

    /*Unlock FSM registers for writing */
    FLASH_CONTROL_REGISTER->FsmWrEna.u32Register    = 0x5U;

    /* Set command to "Clear the Status Register" */
    FLASH_CONTROL_REGISTER->FsmCommand.FSM_COMMAND_BITS.FSMCMD = Fapi_ClearStatus;
    /* Execute the Clear Status command */
    FLASH_CONTROL_REGISTER->FsmExecute.FSM_EXECUTE_BITS.FSMEXECUTE = 0x15U;

    /* Write address to FADDR register */
    FLASH_CONTROL_REGISTER->Faddr.u32Register = 0x0100U;

             /* Placing byte at address 0x0102 */
    oFwpWriteByteAccessor[2] = 0xA5;

    /* Set command to "Program" */
    FLASH_CONTROL_REGISTER->FsmCommand.FSM_COMMAND_BITS.FSMCMD = Fapi_ProgramData;
    /* Execute the Program command */
    FLASH_CONTROL_REGISTER->FsmExecute.FSM_EXECUTE_BITS.FSMEXECUTE = 0x15U;


    /* re-lock FSM registers to prevent writing */
    FLASH_CONTROL_REGISTER->FsmWrEna.u32Register    = 0x2U;
```

3.2.2 Programming 128-Bit Data and 16-Bit ECC

The following code segement will write 16 bytes starting at the 128-bit aligned address specified.

```
        FLASH_CONTROL_REGISTER->Fbprot.u32Register = 1U;  /* Disable Level 1 Protection */

    /* Enable all sectors of current bank for erase and program.  For EEPROM banks with more
than 16 sectors, this must be 0xFFFF */
    FLASH_CONTROL_REGISTER->Fbse.u32Register = 0xFFFF;

    FLASH_CONTROL_REGISTER->Fbprot.u32Register = 0U;  /* Enable Level 1 Protection */

    /*Unlock FSM registers for writing */
    FLASH_CONTROL_REGISTER->FsmWrEna.u32Register    = 0x5U;

    /* Set command to "Clear the Status Register" */
    FLASH_CONTROL_REGISTER->FsmCommand.FSM_COMMAND_BITS.FSMCMD = Fapi_ClearStatus;
    /* Execute the Clear Status command */
    FLASH_CONTROL_REGISTER->FsmExecute.FSM_EXECUTE_BITS.FSMEXECUTE = 0x15U;

    /* Write address to FADDR register */
    FLASH_CONTROL_REGISTER->Faddr.u32Register = 0x0100U;

             /* Placing bytes at address 0x0100 - 0x010F */
    for(u32Index=0;u32Index<16;u32Index++)
    {
       oFwpWriteByteAccessor[u32Index] = au8MainDataBuffer[u32Index];
    }

    /* Supply the address where ECC is being calculated */
    FLASH_CONTROL_REGISTER->FemuAddr.u32Register  = 0x0100;

#if defined(_LITTLE_ENDIAN)
    /* Supply the lower 32bit word */
    FLASH_CONTROL_REGISTER->FemuDlsw.u32Register  = oFwpWriteDwordAccessor[1];
    /* Supply the upper 32bit word */
    FLASH_CONTROL_REGISTER->FemuDmsw.u32Register  = oFwpWriteDwordAccessor[0];
#else
    /* Supply the upper 32bit word */
    FLASH_CONTROL_REGISTER->FemuDlsw.u32Register  = oFwpWriteDwordAccessor[0];
    /* Supply the lower 32bit word */
    FLASH_CONTROL_REGISTER->FemuDmsw.u32Register  = oFwpWriteDwordAccessor[1];
#endif

    /* Place the Wrapper calculated ECC into FWPWRITE_ECC */
    oFwpWriteEccByteAccessor[EI8(0)] = FLASH_CONTROL_REGISTER->FemuEcc.FEMU_ECC_BITS.EMU_ECC);

    /* Supply the address where ECC is being calculated */
    FLASH_CONTROL_REGISTER->FemuAddr.u32Register  = 0x0108;

#if defined(_LITTLE_ENDIAN)
    /* Supply the lower 32bit word */
    FLASH_CONTROL_REGISTER->FemuDlsw.u32Register  = oFwpWriteDwordAccessor[3];
    /* Supply the upper 32bit word */
    FLASH_CONTROL_REGISTER->FemuDmsw.u32Register  = oFwpWriteDwordAccessor[2];
#else
    /* Supply the upper 32bit word */
    FLASH_CONTROL_REGISTER->FemuDlsw.u32Register  = oFwpWriteDwordAccessor[2];
    /* Supply the lower 32bit word */
    FLASH_CONTROL_REGISTER->FemuDmsw.u32Register  = oFwpWriteDwordAccessor[3];
#endif

    /* Place the Wrapper calculated ECC into FWPWRITE_ECC */
    oFwpWriteEccByteAccessor[EI8(1)] = FLASH_CONTROL_REGISTER->FemuEcc.FEMU_ECC_BITS.EMU_ECC);

    /* Set command to "Program" */
    FLASH_CONTROL_REGISTER->FsmCommand.FSM_COMMAND_BITS.FSMCMD = Fapi_ProgramData;
    /* Execute the Program command */
    FLASH_CONTROL_REGISTER->FsmExecute.FSM_EXECUTE_BITS.FSMEXECUTE = 0x15U;


    /* re-lock FSM registers to prevent writing */
    FLASH_CONTROL_REGISTER->FsmWrEna.u32Register    = 0x2U;
```

3.3 Erasing Operations

3.3.1 Erasing a Single Sector With Bank Erase

The following code segment will erase a single Flash sector using the Bank erase command.

NOTE: The Bank Erase command is not a suspendable operation.

```
       FLASH_CONTROL_REGISTER->Fbprot.u32Register = 1U;  /* Disable Level 1 Protection */

   /* Enable all sectors of current bank for erase and program.  For EEPROM banks with more
than 16 sectors, this must be 0xFFFF */
   FLASH_CONTROL_REGISTER->Fbse.u32Register = 0xFFFF;

   FLASH_CONTROL_REGISTER->Fbprot.u32Register = 0U;  /* Enable Level 1 Protection */

   /*Unlock FSM registers for writing */
   FLASH_CONTROL_REGISTER->FsmWrEna.u32Register    = 0x5U;

   FLASH_CONTROL_REGISTER-
>FsmSector.u32Register = 0xFFFE0000;  /* Disable all sectors except for sector 0 from bank
operation */

   /* Set command to "Clear the Status Register" */
   FLASH_CONTROL_REGISTER->FsmCommand.FSM_COMMAND_BITS.FSMCMD = Fapi_ClearStatus;
   /* Execute the Clear Status command */
   FLASH_CONTROL_REGISTER->FsmExecute.FSM_EXECUTE_BITS.FSMEXECUTE = 0x15U;


   /* Write address to FADDR register.  This address must be within the bank to be erased */
   FLASH_CONTROL_REGISTER->Faddr.u32Register = 0x0100U;

   /* Set command to "Program" */
   FLASH_CONTROL_REGISTER->FsmCommand.FSM_COMMAND_BITS.FSMCMD = Fapi_EraseBank;
   /* Execute the Program command */
   FLASH_CONTROL_REGISTER->FsmExecute.FSM_EXECUTE_BITS.FSMEXECUTE = 0x15U;


   /* re-lock FSM registers to prevent writing */
   FLASH_CONTROL_REGISTER->FsmWrEna.u32Register    = 0x2U;
```

3.3.2 Erasing a Single Sector With Sector Erase

The following code segment will erase a single Flash sector using the Sector erase command.

```
        FLASH_CONTROL_REGISTER->Fbprot.u32Register = 1U;  /* Disable Level 1 Protection */

   /* Enable all sectors of current bank for erase and program.  For EEPROM banks with more
than 16 sectors, this must be 0xFFFF */
   FLASH_CONTROL_REGISTER->Fbse.u32Register = 0xFFFF;

   FLASH_CONTROL_REGISTER->Fbprot.u32Register = 0U;  /* Enable Level 1 Protection */

   /*Unlock FSM registers for writing */
   FLASH_CONTROL_REGISTER->FsmWrEna.u32Register    = 0x5U;

   /* Set command to "Clear the Status Register" */
   FLASH_CONTROL_REGISTER->FsmCommand.FSM_COMMAND_BITS.FSMCMD = Fapi_ClearStatus;
   /* Execute the Clear Status command */
   FLASH_CONTROL_REGISTER->FsmExecute.FSM_EXECUTE_BITS.FSMEXECUTE = 0x15U;


   /* Write address to FADDR register.  This address must be within the bank to be erased */
   FLASH_CONTROL_REGISTER->Faddr.u32Register = 0x0100U;

   /* Set command to "Program" */
   FLASH_CONTROL_REGISTER->FsmCommand.FSM_COMMAND_BITS.FSMCMD = Fapi_EraseSector;
   /* Execute the Program command */
   FLASH_CONTROL_REGISTER->FsmExecute.FSM_EXECUTE_BITS.FSMEXECUTE = 0x15U;


   /* re-lock FSM registers to prevent writing */
   FLASH_CONTROL_REGISTER->FsmWrEna.u32Register    = 0x2U;
```

4 References

F021 Flash API Reference Guide (SPNU501)

Products		**Applications**	
Audio	www.ti.com/audio	Automotive and Transportation	www.ti.com/automotive
Amplifiers	amplifier.ti.com	Communications and Telecom	www.ti.com/communications
Data Converters	dataconverter.ti.com	Computers and Peripherals	www.ti.com/computers
DLP® Products	www.dlp.com	Consumer Electronics	www.ti.com/consumer-apps
DSP	dsp.ti.com	Energy and Lighting	www.ti.com/energy
Clocks and Timers	www.ti.com/clocks	Industrial	www.ti.com/industrial
Interface	interface.ti.com	Medical	www.ti.com/medical
Logic	logic.ti.com	Security	www.ti.com/security
Power Mgmt	power.ti.com	Space, Avionics and Defense	www.ti.com/space-avionics-defense
Microcontrollers	microcontroller.ti.com	Video and Imaging	www.ti.com/video
RFID	www.ti-rfid.com		
OMAP Applications Processors	www.ti.com/omap	**TI E2E Community**	e2e.ti.com
Wireless Connectivity	www.ti.com/wirelessconnectivity		