Chapter 1

SLAU392F–August 2012–Revised March 2018

Flash Memory Controller

NOTE: This chapter is an excerpt from the *MSP430x5xx and MSP430x6xx Family User's Guide*. The latest version of the full user's guide can be downloaded from http://www.ti.com/lit/pdf/slau208.

This chapter describes the operation of the flash memory controller.

Topic **Page**

1.1 Flash Memory Introduction

The flash memory is byte, word, and long-word addressable and programmable. The flash memory module has an integrated controller that controls programming and erase operations. The module contains three registers, a timing generator, and a voltage generator to supply program and erase voltages. The cumulative high-voltage time must not be exceeded, and each 32-bit word can be written not more than four times (in byte, word, or long word write modes) before another erase cycle (see device-specific data sheet for details).

The flash memory features include:

- Internal programming voltage generation
- Byte, word (2 bytes), and long (4 bytes) programmable
- Ultra-low-power operation
- Segment erase, bank erase (device specific), and mass erase
- Marginal 0 and marginal 1 read modes
- Each bank (device specific) can be erased individually while program execution can proceed in a different flash bank.

NOTE: Bank operations are not supported on all devices. See the device-specific data sheet for banks supported and their respective sizes.

Figure 1-1 shows the block diagram of the flash memory and controller.

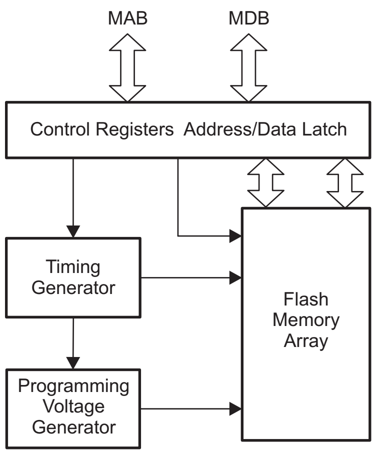
MAB
MDB
Control Registers Address/Data Latch
Timing Generator
Flash Memory Array
Programming Voltage Generator

Figure 1-1. Flash Memory Module Block Diagram

1.2 Flash Memory Segmentation

The flash memory is partitioned into main, information, and BSL memory sections. Bytes, words, 32-byte long-words, or 128-byte blocks can be written to flash memory, but a segment is the smallest size of the flash memory that can be erased. There is no difference in the operation of the memory sections that can be used for storage of code and data. However, the segment size depends on the section:

- Segment size of main memory and BSL memory is 512 bytes
- Segment size of Information memory is 128 bytes

There are four information memory segments, A through D. Each information memory segment can be erased individually.

The bootloader (BSL) memory consists of four segments, A through D. Each BSL memory segment can be erased individually.

The flash main memory is partitioned into 64KB memory banks, which are partitioned into 512-byte segments.

See the device-specific data sheet for availability and address range of each main memory bank.

Figure 1-2 shows the flash segmentation using an example of 256KB of flash that has four banks of 64KB, BSL, and information memory.

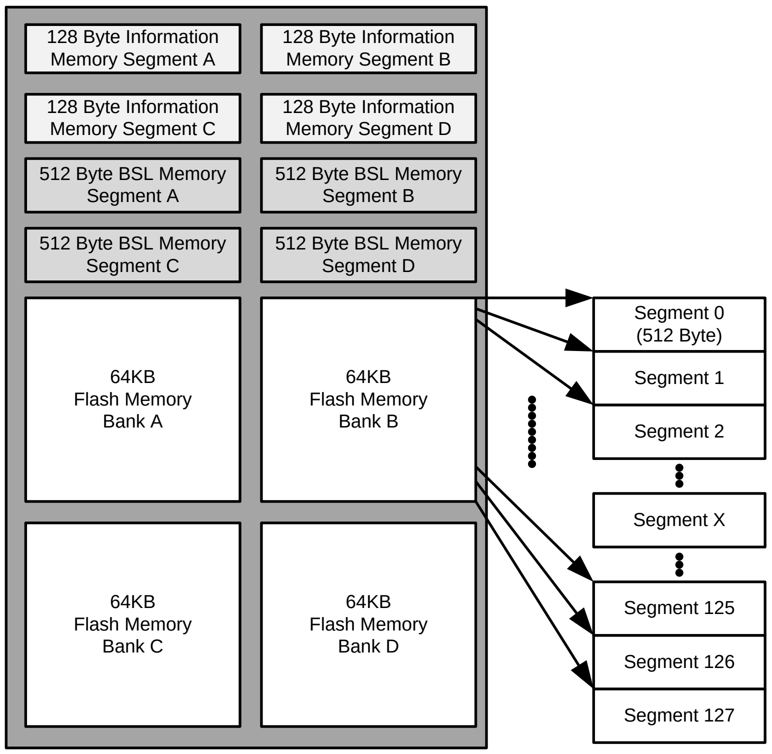
128 Byte Information Memory Segment A
128 Byte Information Memory Segment B
128 Byte Information Memory Segment C
128 Byte Information Memory Segment D
512 Byte BSL Memory Segment A
512 Byte BSL Memory Segment B
512 Byte BSL Memory Segment C
512 Byte BSL Memory Segment D
64KB Flash Memory Bank A
64KB Flash Memory Bank B
64KB Flash Memory Bank C
64KB Flash Memory Bank D
Segment 0 (512 Byte)
Segment 1
Segment 2
Segment X
Segment 125
Segment 126
Segment 127

Figure 1-2. 256KB of Flash Memory Segments Example

1.2.1 Segment A

Segment A of the information memory is locked separately from all other segments with the LOCKA bit. If LOCKA = 1, segment A cannot be written or erased, and all information memory is protected from being segment erased. If LOCKA = 0, segment A can be erased and written like any other flash memory segment.

The state of the LOCKA bit is toggled when a 1 is written to it. Writing a 0 to LOCKA has no effect. This allows existing flash programming routines to be used unchanged.

```
; Unlock Info Memory
   MOV      #FWPW,&FCTL4                ; Clear LOCKINFO, if set
; Unlock SegmentA
   BIT      #LOCKA,&FCTL3               ; Test LOCKA
   JZ       SEGA_UNLOCKED               ; Already unlocked?
   MOV      #FWPW+LOCKA,&FCTL3          ; No, unlock SegmentA
SEGA_UNLOCKED                           ; Yes, continue
; SegmentA is unlocked


; Lock SegmentA
   BIT      #LOCKA,&FCTL3               ; Test LOCKA
   JNZ      SEGA_LOCKED                 ; Already locked?
   MOV      #FWPW+LOCKA,&FCTL3          ; No, lock SegmentA
SEGA_LOCKED                             ; Yes, continue
; SegmentA is locked
; Lock Info Memory
   MOV      #FWPW+LOCKINFO,&FCTL4       ; Set LOCKINFO
```

1.3 Flash Memory Operation

The default mode of the flash memory is read mode. In read mode, the flash memory is not being erased or written, the flash timing generator and voltage generator are off, and the memory operates identically to ROM.

Read and fetch while erase – The flash memory allows execution of a program from flash while a different flash bank is erased. Data reads are also possible from any flash bank not being erased.

NOTE: Read and fetch while erase

The read and fetch while erase feature is available in flash memory configurations where more than one flash bank is available. If there is one flash bank available, holding the complete flash program memory, the read from the program memory and information memory and BSL memory during the erase is not provided. Table 1-1 summarizes which flash operations are supported for devices that support read and fetch while erasing.

Table 1-1. Supported Simultaneous Code Execution and Flash Operations

Flash Operation	Simultaneous Code Execution	
	Within Flash	Within RAM
Bank erase	Supported Executed code must not reside in the bank to be erased	Supported
Segment erase	Not Supported	Supported
Byte, word, long-word write	Not supported	Supported

Flash memory is in-system programmable (ISP) without the need for additional external voltage. The CPU can program the flash memory. The flash memory write and erase modes are selected by the BLKWRT, WRT, MERAS, and ERASE bits and are:

- Byte, word, or long-word (32-bit) write
- Block write
- Segment erase
- Bank erase (only main memory)
- Mass erase (all main memory banks)
- Read during bank erase (except for the one currently read from)

Reading or writing to flash memory while it is busy programming or erasing (page, mass, or bank) from the same bank is prohibited. Any flash erase or programming can be initiated from within flash memory or RAM.

1.3.1 Erasing Flash Memory

The logical value of an erased flash memory bit is 1. Each bit can be programmed from 1 to 0 individually, but to reprogram from 0 to 1 requires an erase cycle. The smallest amount of flash that can be erased is one segment. Table 1-2 lists the three erase modes selected by the ERASE and MERAS bits.

Table 1-2. Erase Modes

MERAS	ERASE	Erase Mode
0	1	Segment erase
1	0	Bank erase (of one bank) selected by the dummy write address[1]
1	1	Mass erase (all memory banks are erased. Information memory A to D and BSL segments A to D are not erased)

[1] Bank operations are not supported on all devices. See the device-specific data sheet for support of bank operations.

1.3.1.1 Erase Cycle

An erase cycle is initiated by a dummy write to the address range of the segment to be erased. The dummy write starts the erase operation and is required for all erase operations including mass erase. Figure 1-3 shows the erase cycle timing. The BUSY bit is set immediately after the dummy write and remains set throughout the erase cycle. BUSY, MERAS, and ERASE are automatically cleared when the cycle completes. No additional dummy write access should be made while the control bits are cleared, otherwise, ACCVIFG is set. The mass erase cycle timing is not dependent on the amount of flash memory present on a device. Erase cycle times are equivalent for all devices.

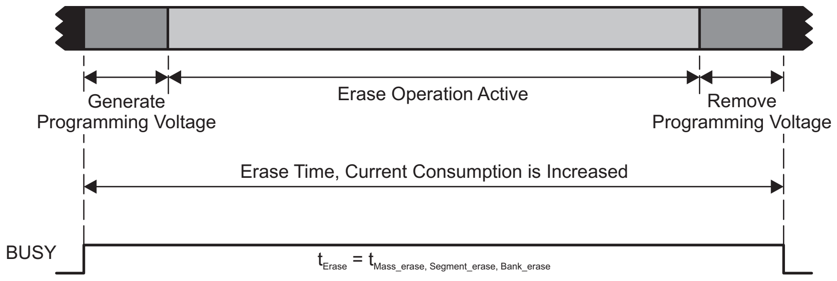
Generate Programming Voltage
Erase Operation Active
Remove Programming Voltage
Erase Time, Current Consumption is Increased
BUSY
$t_{Erase} = t_{Mass\_erase, Segment\_erase, Bank\_erase}$

Figure 1-3. Erase Cycle Timing

1.3.1.2 Erasing Main Memory

The main memory consists of one or more banks. Each bank can be erased individually (bank erase). All main memory banks can be erased in the mass erase mode.

1.3.1.3 Erasing Information Memory or BSL Flash Segments

The information memory A to D and the BSL segments A to D can only be erased in segment erase mode. They are not erased during a bank erase or a mass erase. Erasing is only possible by first clearing the LOCKINFO bit.

1.3.1.4 Initiating Erase From Flash

An erase cycle can be initiated from within flash memory. During a bank erase, code can be executed from flash or RAM. The executed code cannot be located in a bank to be erased.

For any segment erase, the CPU is held until the erase cycle completes regardless of the bank the code resides in. After the segment erase cycle ends, the CPU resumes code execution with the instruction following the dummy write.

When initiating an erase cycle from within flash memory, it is possible to erase the code needed for execution after the erase operation. If this occurs, CPU execution is unpredictable after the erase cycle.

Figure 1-4 shows the flow to initiate an erase from flash.

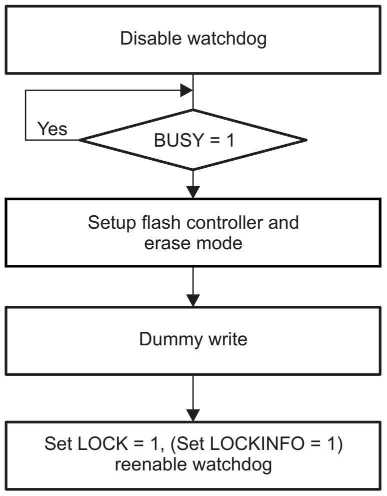
Disable watchdog
Yes
BUSY = 1
Setup flash controller and erase mode
Dummy write
Set LOCK = 1, (Set LOCKINFO = 1) reenable watchdog

Figure 1-4. Erase Cycle From Flash

```
; Segment Erase from flash.
; Assumes Program Memory. Information memory or BSL
; requires LOCKINFO to be cleared as well.
; Assumes ACCVIE = NMIIE = OFIE = 0.
    MOV   #WDTPW+WDTHOLD,&WDTCTL     ; Disable WDT
L1  BIT   #BUSY,&FCTL3               ; Test BUSY
    JNZ   L1                         ; Loop while busy
    MOV   #FWPW,&FCTL3               ; Clear LOCK
    MOV   #FWPW+ERASE,&FCTL1         ; Enable segment erase
    CLR   &0FC10h                    ; Dummy write
L2  BIT   #BUSY,&FCTL3               ; Test BUSY
    JNZ   L2                         ; Loop while busy
    MOV   #FWPW+LOCK,&FCTL3          ; Done, set LOCK
    ...                              ; Re-enable WDT?
```

1.3.1.5 Initiating Erase From RAM

An erase cycle can be initiated from RAM. In this case, the CPU is not held and continues to execute code from RAM. The mass erase (all main memory banks) operation is initiated while executing from RAM. The BUSY bit is used to determine the end of the erase cycle. If the flash is busy completing a bank erase, flash addresses of a different bank can be used to read data or to fetch instructions. While the flash is BUSY, starting an erase cycle or a programming cycle causes an access violation, ACCIFG is set to 1, and the result of the erase operation is unpredictable.

Figure 1-5 shows the flow to initiate an erase of flash from RAM.

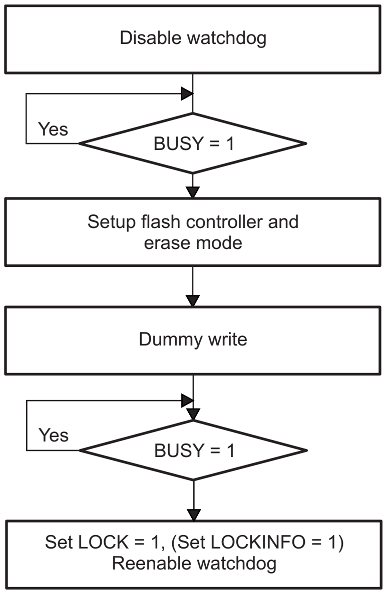
Disable watchdog
Yes
BUSY = 1
Setup flash controller and erase mode
Dummy write
Yes
BUSY = 1
Set LOCK = 1, (Set LOCKINFO = 1)
Reenable watchdog

Figure 1-5. Erase Cycle From RAM

```
; segment Erase from RAM.
; Assumes Program Memory. Information memory or BSL
; requires LOCKINFO to be cleared as well.
; Assumes ACCVIE = NMIIE = OFIE = 0.
    MOV   #WDTPW+WDTHOLD,&WDTCTL      ; Disable WDT
L1  BIT   #BUSY,&FCTL3                ; Test BUSY
    JNZ   L1                          ; Loop while busy
    MOV   #FWPW,&FCTL3                ; Clear LOCK
    MOV   #FWPW+ERASE,&FCTL1          ; Enable page erase
    CLR   &0FC10h                     ; Dummy write
L2  BIT   #BUSY,&FCTL3                ; Test BUSY
    JNZ   L2                          ; Loop while busy
    MOV   #FWPW+LOCK,&FCTL3           ; Done, set LOCK
    ...                               ; Re-enable WDT?
```

1.3.2 Writing Flash Memory

Table 1-3 lists the write modes selected by the WRT and BLKWRT bits.

Table 1-3. Write Modes

BLKWRT	WRT	Write Mode
0	1	Byte or word write
1	0	Long-word write
1	1	Long-word block write

The write modes use a sequence of individual write instructions. Using the long-word write mode is approximately twice as fast as the byte or word mode. Using the long-word block write mode is approximately four times faster than byte or word mode, because the voltage generator remains on for the complete block write, and long-words are written in parallel. Any instruction that modifies a destination can be used to modify a flash location in either byte or word write mode, long-word write mode, or block long-word write mode.

The BUSY bit is set while the write operation is active and cleared when the operation completes. If the write operation is initiated from RAM, the CPU must not access flash while BUSY is set to 1. Otherwise, an access violation occurs, ACCVIFG is set, and the flash write is unpredictable.

1.3.2.1 Byte or Word Write

A byte or word write operation can be initiated from within flash memory or from RAM. When initiating from within flash memory, the CPU is held while the write completes. After the write completes, the CPU resumes code execution with the instruction following the write access. Figure 1-6 shows the byte, word, and long-word write timing. Byte, word, and long-word write times are identical.

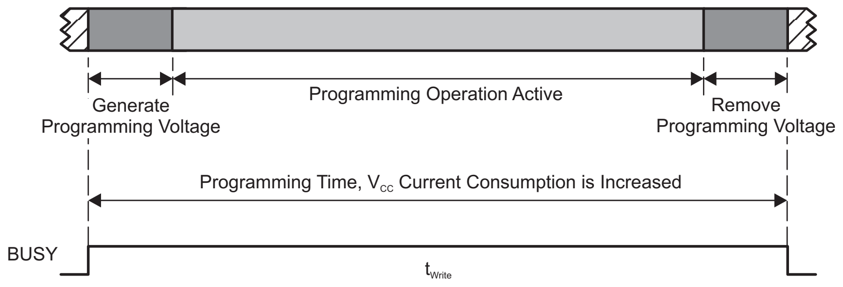
Generate Programming Voltage
Programming Operation Active
Remove Programming Voltage
Programming Time, V_{CC} Current Consumption is Increased
BUSY
t_{Write}

Figure 1-6. Byte, Word, and Long-Word Write Timing

When a byte or word write is executed from RAM, the CPU continues to execute code from RAM. The BUSY bit must be zero before the CPU accesses flash again, otherwise an access violation occurs, ACCVIFG is set, and the write result is unpredictable.

In any write mode, the internally-generated programming voltage is applied to the complete 128-byte block. The cumulative programming time, t_{CPT}, must not be exceeded for any block. Each byte, word, or long-word write adds to the cumulative program time. If the maximum cumulative program time is reached or exceeded, the segment data returns unpredictable results and needs to be erased before further usage. See the device-specific data sheet for specifications.

1.3.2.2 Initiating Byte or Word Write From Flash

Figure 1-7 shows the flow to initiate a byte or word write from flash.

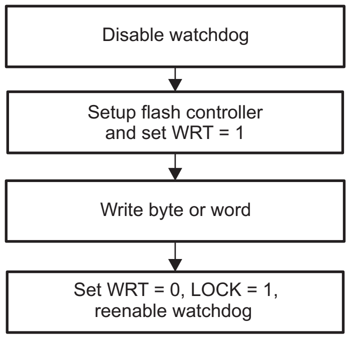
Disable watchdog
Setup flash controller and set WRT = 1
Write byte or word
Set WRT = 0, LOCK = 1, reenable watchdog

Figure 1-7. Initiating a Byte or Word Write From Flash

```
; Byte or word write from flash.
; Assumes 0x0FF1E is already erased
; Assumes ACCVIE = NMIIE = OFIE = 0.
   MOV   #WDTPW+WDTHOLD,&WDTCTL     ; Disable WDT
   MOV   #FWPW,&FCTL3               ; Clear LOCK
   MOV   #FWPW+WRT,&FCTL1           ; Enable write
   MOV   #0123h,&0FF1Eh             ; 0123h -> 0x0FF1E
   MOV   #FWPW,&FCTL1               ; Done. Clear WRT
   MOV   #FWPW+LOCK,&FCTL3          ; Set LOCK
   ...                              ; Re-enable WDT?
```

1.3.2.3 Initiating Byte or Word Write From RAM

Figure 1-8 shows the flow to initiate a byte or word write from RAM.

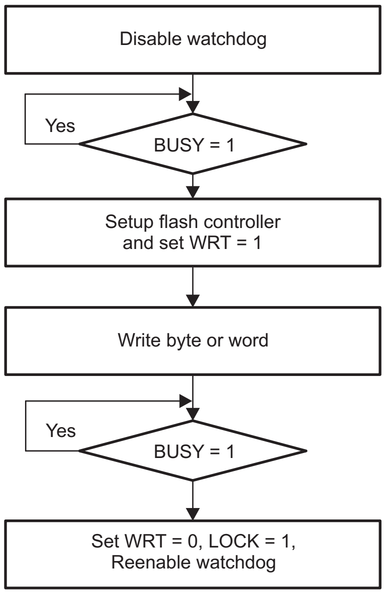
Disable watchdog
Yes
BUSY = 1
Setup flash controller and set WRT = 1
Write byte or word
Yes
BUSY = 1
Set WRT = 0, LOCK = 1, Reenable watchdog

Figure 1-8. Initiating a Byte or Word Write From RAM

```
; Byte or word write from RAM.
; Assumes 0x0FF1E is already erased
; Assumes ACCVIE = NMIIE = OFIE = 0.
    MOV   #WDTPW+WDTHOLD,&WDTCTL     ; Disable WDT
L1  BIT   #BUSY,&FCTL3               ; Test BUSY
    JNZ   L1                         ; Loop while busy
    MOV   #FWPW,&FCTL3               ; Clear LOCK
    MOV   #FWPW+WRT,&FCTL1           ; Enable write
    MOV   #0123h,&0FF1Eh             ; 0123h -> 0x0FF1E
L2  BIT   #BUSY,&FCTL3               ; Test BUSY
    JNZ   L2                         ; Loop while busy
    MOV   #FWPW,&FCTL1               ; Clear WRT
    MOV   #FWPW+LOCK,&FCTL3          ; Set LOCK
    ...                              ; Re-enable WDT?
```

1.3.2.4 Long-Word Write

A long-word write operation can be initiated from within flash memory or from RAM. The BUSY bit is set to 1 after 32 bits are written to the flash controller and the programming cycle starts. When initiating from within flash memory, the CPU is held while the write completes. After the write completes, the CPU resumes code execution with the instruction following the write access. Figure 1-6 shows the long-word write timing.

A long-word consists of four consecutive bytes aligned to at 32-bit address (only the lower two address bits are different). The bytes can be written in any order or any combination of bytes and words. If a byte or word is written more than once, the last data written to the four bytes are stored into the flash memory.

If a write to a flash address outside of the 32-bit address happens before all four bytes are available, the data written so far is discarded, and the latest byte or word written defines the new 32-bit aligned address.

When 32 bits are available, the write cycle is executed. When executing from RAM, the CPU continues to execute code. The BUSY bit must be zero before the CPU accesses flash again, otherwise an access violation occurs, ACCVIFG is set, and the write result is unpredictable.

In long-word write mode, the internally-generated programming voltage is applied to a complete 128-byte block. The cumulative programming time, t_{CPT}, must not be exceeded for any block. Each write adds to the cumulative program time of a segment. If the maximum cumulative program time is reached or exceeded, the segment must be erased. Further programming or using the data returns unpredictable results.

With each write, the amount of time the block is subjected to the programming voltage accumulates. If the cumulative programming time is reached or exceeded, the block must be erased before further programming or use (see the device-specific data sheet for specifications).

1.3.2.5 Initiating Long-Word Write From Flash

Figure 1-9 shows the flow to initiate a long-word write from flash.

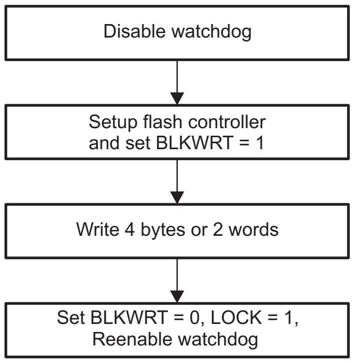
Disable watchdog
Setup flash controller and set BLKWRT = 1
Write 4 bytes or 2 words
Set BLKWRT = 0, LOCK = 1, Reenable watchdog

Figure 1-9. Initiating Long-Word Write From Flash

```
; Long-word write from flash.
; Assumes 0x0FF1C and 0x0FF1E is already erased
; Assumes ACCVIE = NMIIE = OFIE = 0.
   MOV   #WDTPW+WDTHOLD,&WDTCTL     ; Disable WDT
   MOV   #FWPW,&FCTL3               ; Clear LOCK
   MOV   #FWPW+BLKWRT,&FCTL1        ; Enable 2-word write
   MOV   #0123h,&0FF1Ch             ; 0123h -> 0x0FF1C
   MOV   #45676h,&0FF1Eh            ; 04567h -> 0x0FF1E
   MOV   #FWPW,&FCTL1               ; Done. Clear BLKWRT
   MOV   #FWPW+LOCK,&FCTL3          ; Set LOCK
   ...                              ; Re-enable WDT?
```

1.3.2.6 Initiating Long-Word Write From RAM

Figure 1-10 shows the flow to initiate a long-word write from RAM.

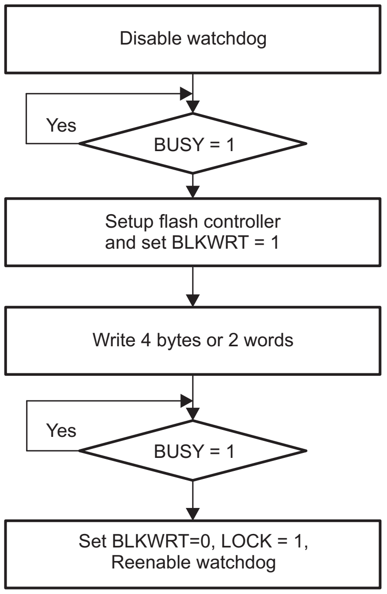
Disable watchdog
Yes
BUSY = 1
Setup flash controller and set BLKWRT = 1
Write 4 bytes or 2 words
Yes
BUSY = 1
Set BLKWRT=0, LOCK = 1, Reenable watchdog

Figure 1-10. Initiating Long-Word Write from RAM

```
; Two 16-bit word writes from RAM.
; Assumes 0x0FF1C and 0x0FF1E is already erased
; Assumes ACCVIE = NMIIE = OFIE = 0.
    MOV   #WDTPW+WDTHOLD,&WDTCTL     ; Disable WDT
L1  BIT   #BUSY,&FCTL3               ; Test BUSY
    JNZ   L1                         ; Loop while busy
    MOV   #FWPW,&FCTL3               ; Clear LOCK
    MOV   #FWPW+BLKWRT,&FCTL1        ; Enable write
    MOV   #0123h,&0FF1Ch             ; 0123h -> 0x0FF1C
    MOV   #4567h,&0FF1Eh             ; 4567h -> 0x0FF1E
L2  BIT   #BUSY,&FCTL3               ; Test BUSY
    JNZ   L2                         ; Loop while busy
    MOV   #FWPW,&FCTL1               ; Clear WRT
    MOV   #FWPW+LOCK,&FCTL3          ; Set LOCK
    ...                              ; Re-enable WDT?
```

1.3.2.7 Block Write

The block write can be used to accelerate the flash write process when many sequential bytes or words need to be programmed. The flash programming voltage remains on for the duration of writing the 128-byte block. The cumulative programming time, t_{CPT}, must not be exceeded for any block during a block write. Only long-word writes are possible using block write mode.

A block write cannot be initiated from within flash memory. The block write must be initiated from RAM. The BUSY bit remains set throughout the duration of the block write. The WAIT bit must be checked between writing four bytes, or two words, to the block. When WAIT is set, then four bytes, or two 16-bit words, of the block can be written. When writing successive blocks, the BLKWRT bit must be cleared after the current block is completed. BLKWRT can be set initiating the next block write after the required flash recovery time given by t_{END}. BUSY is cleared following each block write completion, indicating the next block can be written. Figure 1-11 shows the block write timing. The first long-word write requires $t_{Block,0}$ and the last long-write requires $t_{Block,N}$. All other blocks require $t_{Block,1\text{-}(N-1)}$.

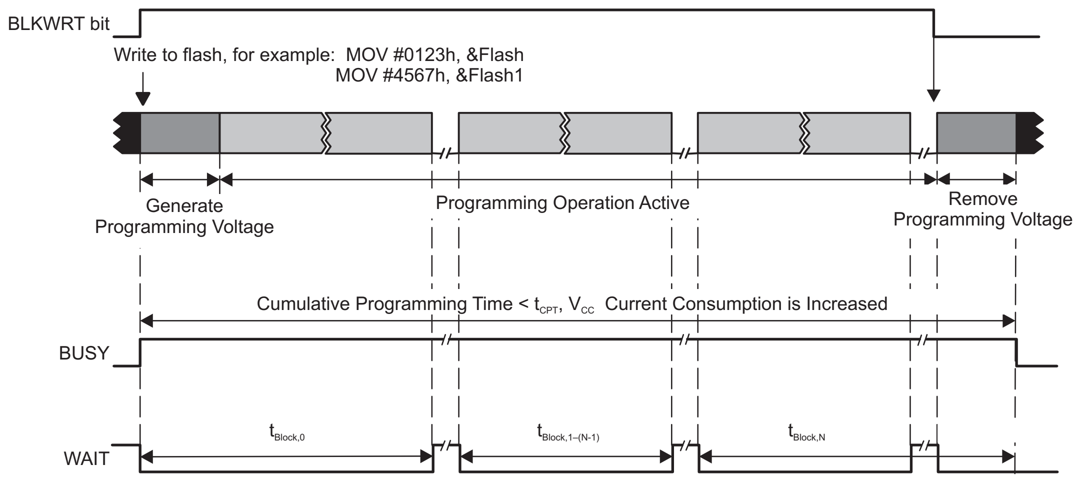
BLKWRT bit
Write to flash, for example: MOV #0123h, &Flash
MOV #4567h, &Flash1
Generate Programming Voltage
Programming Operation Active
Remove Programming Voltage
Cumulative Programming Time < tCPT, VCC Current Consumption is Increased
BUSY
WAIT
tBlock,0
tBlock,1–(N-1)
tBlock,N

Figure 1-11. Block-Write Cycle Timing

1.3.2.8 Block Write Flow and Example

Figure 1-12 and the following code example show the block write flow.

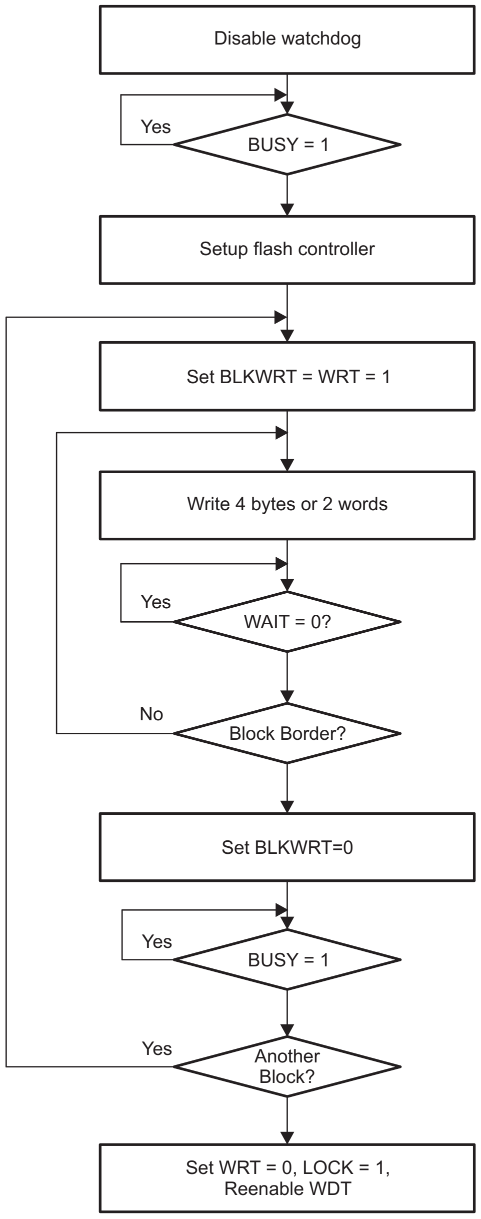

Figure 1-12. Block Write Flow

```
; Write one block starting at 0F000h.
; Must be executed from RAM, Assumes Flash is already erased.
; Assumes ACCVIE = NMIIE = OFIE = 0.
    MOV    #32,R5                     ; Use as write counter
    MOV    #0F000h,R6                 ; Write pointer
    MOV    #WDTPW+WDTHOLD,&WDTCTL     ; Disable WDT
L1  BIT    #BUSY,&FCTL3               ; Test BUSY
    JNZ    L1                         ; Loop while busy
    MOV    #FWPW,&FCTL3               ; Clear LOCK
    MOV    #FWPW+BLKWRT+WRT,&FCTL1    ; Enable block write
L2  MOV    Write_Value1,0(R6)         ; Write 1st location
    MOV    Write_Value2,2(R6)         ; Write 2nd word
L3  BIT    #WAIT,&FCTL3               ; Test WAIT
    JZ     L3                         ; Loop while WAIT=0
    INCD   R6                         ; Point to next words
    INCD   R6                         ; Point to next words
    DEC    R5                         ; Decrement write counter
    JNZ    L2                         ; End of block?
    MOV    #FWPW,&FCTL1               ; Clear WRT, BLKWRT
L4  BIT    #BUSY,&FCTL3               ; Test BUSY
    JNZ    L4                         ; Loop while busy
    MOV    #FWPW+LOCK,&FCTL3          ; Set LOCK
    ...                               ; Re-enable WDT if needed
```

1.3.3 Flash Memory Access During Write or Erase

When a write or an erase operation is initiated from RAM while BUSY = 1, the CPU may not write to any flash location. Otherwise, an access violation occurs, ACCVIFG is set, and the result is unpredictable. ACCVIFG is also set if a Flash write or erase access is attempted without any Flash write or erase mode selected first.

When a write operation is initiated from within flash memory, the CPU continues code execution with the next instruction fetch after the write cycle completed (BUSY = 0).

The op-code 3FFFh is the JMP PC instruction. This causes the CPU to loop until the flash operation is finished. When the operation is finished and BUSY = 0, the flash controller allows the CPU to fetch the op-code and program execution resumes.

Table 1-4 lists the flash access conditions while BUSY = 1.

Table 1-4. Flash Access While Flash is Busy (BUSY = 1)

Flash Operation	Flash Access	WAIT	Result
Bank erase	Read	0	From the erased bank: ACCVIFG = 0. 03FFFh is the value read. From any other flash location: ACCVIFG = 0. Valid read.
	Write	0	ACCVIFG = 1. Write is ignored.
	Instruction fetch	0	From the erased bank: ACCVIFG = 0. CPU fetches 03FFFh. This is the JMP PC instruction. From any other flash location: ACCVIFG = 0. Valid instruction fetch.
Segment erase	Read	0	ACCVIFG = 0: 03FFFh is the value read.
	Write	0	ACCVIFG = 1: Write is ignored.
	Instruction fetch	0	ACCVIFG = 0: CPU fetches 03FFFh. This is the JMP PC instruction.
Word or byte write or long-word write	Read	0	ACCVIFG = 0: 03FFFh is the value read.
	Write	0	ACCVIFG = 1: Write is ignored.
	Instruction fetch	0	ACCVIFG = 0: CPU fetches 03FFFh. This is the JMP PC instruction.
Block write	Any	0	ACCVIFG = 1: LOCK = 1, block write is exited.
	Read	1	ACCVIFG = 0: 03FFFh is the value read.
	Write	1	ACCVIFG = 0: Valid write
	Instruction fetch	1	ACCVIFG = 1: LOCK = 1, block write is exited

Interrupts are automatically disabled during any flash operation.

The watchdog timer (in watchdog mode) should be disabled before a flash erase cycle. A reset aborts the erase and the result is unpredictable. After the erase cycle has completed, the watchdog may be reenabled.

1.3.4 Stopping Write or Erase Cycle

Any write or erase operation can be stopped before its normal completion by setting the emergency exit bit EMEX. Setting the EMEX bit stops the active operation and resets the flash controller. All flash operations cease, the flash returns to read mode, and all bits in the FCTL1 register are reset. The LOCK bit of FCTL3 is set. The result of the intended operation is unpredictable.

1.3.4.1 EMEX With Single Bank Flash Memory

For devices with single bank flash memories, write and erase operations initiated from flash, the CPU is held until the flash operation completes. Therefore it is not possible to perform an emergency exit by the EMEX bit. The emergency exit of write or erase operations initiated from RAM can be performed using the EMEX bit. The BUSY bit is used to determine the end of the emergency exit cycle. The user must ensure that code execution does not continue until the BUSY bit is cleared by the flash controller.

1.3.4.2 EMEX With Multiple Bank Flash Memory

For devices with multiple bank flash memories, write and segment erase operations initiated from flash, regardless of which bank the code resides in, the CPU is held until the flash operation completes. Therefore it is not possible to perform an emergency exit by the EMEX bit. For bank erase, there is a possibility to perform an EMEX if the bank being erased is not where the code resides. The BUSY bit is used to determine the end of the emergency exit cycle. The user must ensure that code execution does not continue until the BUSY bit is cleared by the flash controller.

The emergency exit of write or any erase operations initiated from RAM can be performed using the EMEX bit. The BUSY bit is used to determine the end of the emergency exit cycle. The user must ensure that code execution does not continue until the BUSY bit is cleared by the flash controller.

1.3.5 Checking Flash Memory

The result of a programming cycle of the flash memory can be checked by calculating and storing a checksum (CRC) of parts or the complete flash memory content. The CRC module can be used for this purpose (see the device-specific data sheet). During the runtime of the system, the known checksums can be recalculated and compared with the expected values stored in the flash memory. The program checking the flash memory content is executed in RAM.

To get an early indication of weak memory cells, reading the flash can be done in combination with the device-specific marginal read modes. The marginal read modes are controlled by the FCTL4.MRG0 and FCTL4.MRG1 register bits if available (device specific). During marginal read mode, marginally programmed flash memory bit locations can be detected. One method for identifying such memory locations would be to periodically perform a checksum calculation over a section of flash memory (for example, a flash segment) and repeating this procedure with the marginal read mode enabled. If they do not match, it could indicate an insufficiently programmed flash memory location. It is possible to refresh the affected Flash memory segment by disabling marginal read mode, copying to RAM, erasing the flash segment, and writing back to it from RAM.

The program checking the flash memory contents must be executed from RAM. Executing code from flash automatically disables the marginal read mode. The marginal read modes are controlled by the MRG0 and MRG1 register bits. Setting MRG1 is used to detect insufficiently programmed flash cells containing a "1" (erased bits). Setting MRG0 is used to detect insufficiently programmed flash cells containing a "0" (programmed bits). Only one of these bits should be set at a time. Therefore, a full marginal read check requires two passes of checking the flash memory content’s integrity. During marginal read mode, the flash access speed (MCLK) must be limited to 1 MHz (see the device-specific data sheet).

1.3.6 Configuring and Accessing the Flash Memory Controller

The FCTLx registers are 16-bit password-protected read and write registers. Any read or write access must use word instructions, and write accesses must include the write password 0A5h in the upper byte. Any write to any FCTLx register with a value other than 0A5h in the upper byte is a password violation, sets the KEYV flag, and triggers a PUC system reset. Any read of any FCTLx registers reads 096h in the upper byte.

Any write to FCTL1 during an erase or byte, word, double-word write operation is an access violation and sets ACCVIFG. Writing to FCTL1 is allowed in block write mode when WAIT = 1, but writing to FCTL1 in block write mode when WAIT = 0 is an access violation and sets ACCVIFG.

Any write to FCTL2 (this register is currently not implemented) when BUSY = 1 is an access violation.

Any FCTLx register may be read when BUSY = 1. A read does not cause an access violation.

1.3.7 Flash Memory Controller Interrupts

The flash controller has two interrupt sources, KEYV and ACCVIFG. ACCVIFG is set when an access violation occurs. When the ACCVIE bit is reenabled after a flash write or erase, a set ACCVIFG flag generates an interrupt request. The ACCVIE bit resides in the Special Function Register, SFRIE1 (see the SYS chapter for details). ACCVIFG sources the NMI interrupt vector, so it is not necessary for GIE to be set for ACCVIFG to request an interrupt. ACCVIFG may also be checked by software to determine if an access violation occurred. ACCVIFG must be reset by software.

The password violation flag, KEYV, is set when any of the flash control registers are written with an incorrect password. When this occurs, a PUC is generated immediately, resetting the device.

1.3.8 Programming Flash Memory Devices

There are three options for programming a flash device. All options support in-system programming.

- Program through JTAG (Section 1.3.8.1)
- Program through the BSL (Section 1.3.8.2)
- Program through a custom solution (Section 1.3.8.3)

1.3.8.1 Programming Flash Memory Through JTAG

Devices can be programmed through the JTAG port. The JTAG interface requires four signals (five signals on 20- and 28-pin devices), ground, and optionally VCC and $\overline{\text{RST}}$/NMI.

The JTAG port is protected with a fuse. Blowing the fuse completely disables the JTAG port and is not reversible. Further access to the device through JTAG is not possible. For more details, see the *MSP430 Programming With the JTAG Interface*.

1.3.8.2 Programming Flash Memory Through the BSL

Every flash device contains a BSL. The BSL enables users to read or program the flash memory or RAM using a UART serial interface. Access to the flash memory through the BSL is protected by a 256-bit user-defined password. For more details, see the *MSP430 Programming With the Bootloader (BSL)*.

1.3.8.3 Programming Flash Memory Through Custom Solution

The ability of the MSP430 CPU to write to its own flash memory allows for in-system and external custom programming solutions (see Figure 1-13). The user can choose to provide data through any means available (for example, UART or SPI). User-developed software can receive the data and program the flash memory. Because this type of solution is developed by the user, it can be completely customized to fit the application needs for programming, erasing, or updating the flash memory.

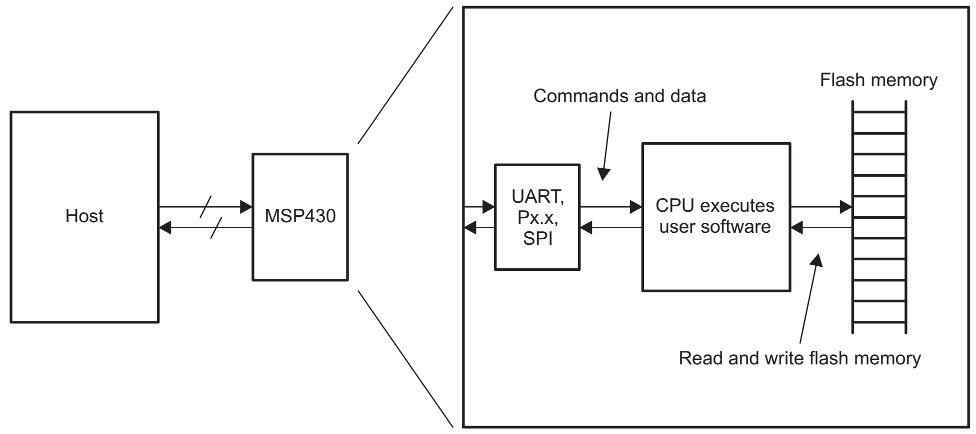

Figure 1-13. User-Developed Programming Solution

1.4 FCTL Registers

The flash memory controller (FCTL) registers are listed in Table 1-5. The base address can be found in the device-specific data sheet. The address offset is given in Table 1-5.

NOTE: All registers have word or byte register access. For a generic register *ANYREG*, the suffix "_L" (*ANYREG_L*) refers to the lower byte of the register (bits 0 through 7). The suffix "_H" (*ANYREG_H*) refers to the upper byte of the register (bits 8 through 15).

Table 1-5. FCTL Registers

Offset	Acronym	Register Name	Type	Access	Reset	Section
00h	FCTL1	Flash Memory Control 1	Read/write	Word	9600h	Section 1.4.1
00h	FCTL1_L		Read/Write	Byte	00h	
01h	FCTL1_H		Read/Write	Byte	96h	
04h	FCTL3	Flash Memory Control 3	Read/write	Word	9658h	Section 1.4.2
04h	FCTL3_L		Read/Write	Byte	58h	
05h	FCTL3_H		Read/Write	Byte	96h	
06h	FCTL4	Flash Memory Control 4	Read/write	Word	9600h	Section 1.4.3
06h	FCTL4_L		Read/Write	Byte	00h	
07h	FCTL4_H		Read/Write	Byte	96h	

1.4.1 FCTL1 Register

Flash Memory Control 1 Register

Figure 1-14. FCTL1 Register

15	14	13	12	11	10	9	8
FRPW/FWPW							
7	6	5	4	3	2	1	0
BLKWRT	WRT	SWRT	Reserved		MERAS	ERASE	Reserved
rw-0	rw-0	rw-0	r-0	r-0	rw-0	rw-0	r-0

Table 1-6. FCTL1 Register Description

Bit	Field	Type	Reset	Description
15-8	FRPW/FWPW	RW	96h	FCTL password. Always read as 096h. Must be written as 0A5h or a PUC is generated.
7	BLKWRT	RW	0h	Block write. BLKWRT and WRT are used together to select the write mode. The values shown below are for BLKWRT-WRT. 0-0 = Reserved 0-1 = Byte or word write 1-0 = Long-word write 1-1 = Long-word block write
6	WRT	RW	0h	Write. BLKWRT and WRT are used together to select the write mode. The values shown below are for BLKWRT-WRT. 0-0 = Reserved 0-1 = Byte or word write 1-0 = Long-word write 1-1 = Long-word block write
5	SWRT	RW	0h	Smart write. If this bit is set, the program time is shortened. The programming quality has to be checked by marginal read modes.
4-3	Reserved	R	0h	Reserved. Always reads as 0.
2	MERAS			Mass erase. MERAS and ERASE are used together to select the erase mode. MERAS and ERASE are automatically reset when EMEX is set or a flash erase operation has completed. The values shown below are for MERAS-ERASE. 0-0 = No erase 0-1 = Segment erase 1-0 = Bank erase (erase of one bank) 1-1 = Mass erase (erase all flash memory banks)
1	ERASE			Erase. MERAS and ERASE are used together to select the erase mode. MERAS and ERASE are automatically reset when EMEX is set or a flash erase operation has completed. The values shown below are for MERAS-ERASE. 0-0 = No erase 0-1 = Segment erase 1-0 = Bank erase (erase of one bank) 1-1 = Mass erase (erase all flash memory banks)
0	Reserved	R	0h	Reserved. Always reads as 0.

1.4.2 FCTL3 Register

Flash Memory Control 3 Register

Figure 1-15. FCTL3 Register

15	14	13	12	11	10	9	8
FRPW/FWPW							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	LOCKA	EMEX	LOCK	WAIT	ACCVIFG	KEYV	BUSY
r-0	rw-1	rw-0	rw-1	r-1	rw-0	rw-(0)	r-0

Table 1-7. FCTL3 Register Description

Bit	Field	Type	Reset	Description
15-8	FRPW/FWPW	RW	96h	FCTLx password. Always read as 096h. Must be written as 0A5h or a PUC is generated.
7	Reserved	R	0h	Reserved. Always reads as 0.
6	LOCKA	RW	1h	Segment A lock. Write a 1 to this bit to change its state. Writing 0 has no effect. 0b = Segment A of the information memory is unlocked and can be written or erased in segment erase mode. 1b = Segment A of the information memory is locked and can not be written or erased in segment erase mode.
5	EMEX	RW	0h	Emergency exit. Setting this bit stops any erase or write operation. The LOCK bit is set. 0b = No emergency exit 1b = Emergency exit
4	LOCK	RW	1h	Lock. This bit unlocks the flash memory for writing or erasing. The LOCK bit can be set any time during a byte or word write or erase operation, and the operation completes normally. In the block write mode, if the LOCK bit is set while BLKWRT = WAIT = 1, BLKWRT and WAIT are reset and the mode ends normally. 0b = Unlocked 1b = Locked
3	WAIT	R	1h	Wait. Indicates the flash memory is being written to. 0b = Flash memory is not ready for the next byte or word write. 1b = Flash memory is ready for the next byte or word write.
2	ACCVIFG	RW	0h	Access violation interrupt flag 0b = No interrupt pending 1b = Interrupt pending
1	KEYV	RW	0h	Flash password violation. This bit indicates an incorrect FCTLx password was written to any flash control register and generates a PUC when set. KEYV must be reset with software. 0b = FCTLx password was written correctly. 1b = FCTLx password was written incorrectly.
0	BUSY	R	0h	Busy. This bit indicates if the flash is currently busy erasing or programming. 0b = Not busy 1b = Busy

1.4.3 FCTL4 Register

Flash Memory Control 4 Register

Figure 1-16. FCTL4 Register

15	14	13	12	11	10	9	8
FRPW/FWPW							
7	6	5	4	3	2	1	0
LOCKINFO	Reserved	MRG1	MRG0	Reserved			VPE
rw-0	r-0	rw-0	rw-0	r-0	r-0	r-0	rw-0

Table 1-8. FCTL4 Register Description

Bit	Field	Type	Reset	Description
15-8	FRPW/FWPW	RW	96h	FCTLx password. Always reads as 096h. Must be written as 0A5h or a PUC is generated.
7	LOCKINFO	RW	0h	Lock information memory. If set, the information memory cannot be erased in segment erase mode and cannot be written to.
6	Reserved	R	0h	Reserved. Always reads as 0.
5	MRG1	RW	0h	Marginal read 1 mode. This bit enables the marginal 1 read mode. The marginal read 1 bit is valid for reads from the flash memory only. During a fetch cycle, the marginal mode is turned off automatically. If both MRG1 and MRG0 are set, MRG1 is active and MRG0 is ignored. 0b = Marginal 1 read mode is disabled. 1b = Marginal 1 read mode is enabled.
4	MRG0	RW	0h	Marginal read 0 mode. This bit enables the marginal 0 read mode. The marginal read 1 bit is valid for reads from the flash memory only. During a fetch cycle, the marginal mode is turned off automatically. If both MRG1 and MRG0 are set, MRG1 is active and MRG0 is ignored. 0b = Marginal 0 read mode is disabled. 1b = Marginal 0 read mode is enabled.
3-1	Reserved	R	0h	Reserved. Always reads as 0.
0	VPE	RW	0h	Voltage changed during program error. This bit is set by hardware and can only be cleared by software. If DVCC changed significantly during programming, this bit is set to indicate an invalid result.

1.4.4 SFRIE1 Register

Interrupt Enable 1 Register

Figure 1-17. SFRIE1 Register

15	14	13	12	11	10	9	8

7	6	5	4	3	2	1	0
		ACCVIE					
		rw-0					

Table 1-9. SFRIE1 Register Description

Bit	Field	Type	Reset	Description
15-6				These bits may be used by other modules (see the device-specific data sheet and the SYS chapter for details).
5	ACCVIE	RW	0h	Flash memory access violation interrupt enable. This bit enables the ACCVIFG interrupt. Because other bits in SFRIE1 may be used for other modules, it is recommended to set or clear this bit using `BIS` or `BIC` instructions, rather than `MOV` or `CLR` instructions. See the SYS chapter for more details. 0b = Interrupt not enabled 1b = Interrupt enabled
4-0				These bits may be used by other modules (see the device-specific data sheet and the SYS chapter for details).