

# Section 5. Flash Programming

## HIGHLIGHTS

This section of the manual contains the following topics:

5.1	Introduction .....	5-2
5.2	Control Registers .....	5-3
5.3	Run-Time Self-Programming (RTSP) Operation .....	5-10
5.4	Lock-Out Feature .....	5-11
5.5	Word Programming Sequence .....	5-13
5.6	Row Programming Sequence .....	5-14
5.7	Page Erase Sequence .....	5-15
5.8	Program Flash Memory Erase Sequence .....	5-15
5.9	Operation in Power-Saving and Debug Modes .....	5-16
5.10	Effects of Various Resets .....	5-16
5.11	Interrupts .....	5-17
5.12	Related Application Notes .....	5-18
5.13	Revision History .....	5-19

**Note:** This family reference manual section is meant to serve as a complement to device data sheets. Depending on the device variant, this manual section may not apply to all PIC32 devices.

Please consult the note at the beginning of the “**Flash Programming**” chapter in the current device data sheet to check whether this document supports the device you are using.

Device data sheets and family reference manual sections are available for download from the Microchip Worldwide Web site at: <http://www.microchip.com>

## 5.1 INTRODUCTION

This section describes techniques for programming the Flash memory. PIC32 devices contain internal Flash memory for executing user code. There are three methods by which the user can program this memory:

- Run-Time Self-Programming (RTSP) – performed by the user’s software
- In-Circuit Serial Programming™ (ICSP™) – performed using a serial data connection to the device, which allows much faster programming than RTSP
- Enhanced Joint Test Action Group Programming (EJTAG) – performed by an EJTAG-capable programmer, using the EJTAG port of the device

RTSP techniques are described in this chapter. The ICSP and EJTAG methods are described in the “*PIC32MX Flash Programming Specification*” (DS61145), which is available for download from the Microchip web site ([www.microchip.com](http://www.microchip.com)).

## 5.2 CONTROL REGISTERS

Flash program and erase operations are controlled using the following Non-Volatile Memory (NVM) control registers:

- **NVMCON: Programming Control Register**
- **NVMKEY: Programming Unlock Register**
- **NVMADDR: Flash Address Register**
- **NVMDATA: Flash Program Data Register**
- **NVMSRCADDR: Source Data Address Register**

Table 5-1 provides a brief summary of all of the Flash-programming-related registers. Corresponding registers appear after the summary, followed by a detailed description.

**Table 5-1: Flash Controller SFR Summary**

Name		Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
NVMCON <sup>(1)</sup>	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	WR	WREN	WRERR	LVDERR	LVDSTAT	—	—	—
	7:0	—	—	—	—	NVMOP<3:0>			
NVMKEY	31:24	NVMKEY<31:24>							
	23:16	NVMKEY<23:16>							
	15:8	NVMKEY<15:8>							
	7:0	NVMKEY<7:0>							
NVMADDR <sup>(1)</sup>	31:24	NVMADDR<31:24>							
	23:16	NVMADDR<23:16>							
	15:8	NVMADDR<15:8>							
	7:0	NVMADDR<7:0>							
NVMDATA	31:24	NVMDATA<31:24>							
	23:16	NVMDATA<23:16>							
	15:8	NVMDATA<15:8>							
	7:0	NVMDATA<7:0>							
NVMSRCADDR	31:24	NVMSRCADDR<31:24>							
	23:16	NVMSRCADDR<23:16>							
	15:8	NVMSRCADDR<15:8>							
	7:0	NVMSRCADDR<7:0>							

**Legend:** — = unimplemented, read as '0'.

**Note 1:** This register has an associated Clear, Set, and Invert register at an offset of 0x4, 0x8, and 0xC bytes, respectively. These registers have the same name with CLR, SET, or INV appended to the end of the register name (e.g., NVMCONCLR). Writing a '1' to any bit position in these registers will clear, set, or invert valid bits in the associated register. Reads from these registers should be ignored.

## 5.2.1 NVMCON Register

The NVMCON register is the control register for Flash program/erase operations. This register selects whether an erase or program operation can be performed and is used to start the program or erase cycle.

The NVMCON register is shown in [Register 5-1](#). The lower byte of NVMCON configures the type of NVM operation that will be performed. A summary of the NVMCON setup values for various program and erase operations is given in [Table 5-2](#).

**Table 5-2: NVMCON Register Values**

Operation	NVMCON Value
Page Erase	0x8004
Program Word	0x8001
Program Row	0x8003
No Operation (NOP)	0x8000

**Register 5-1: NVMCON: Programming Control Register**

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 31							bit 24

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 23							bit 16

HC, R/W-0	R/W-0	HS, R-0	HS, R-0	HS, HC, R-0	U-0	U-0	U-0
WR	WREN <sup>(1)</sup>	WRERR <sup>(2)</sup>	LVDERR <sup>(2)</sup>	LVDSTAT <sup>(2)</sup>	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	NVMOP<3:0>			
bit 7							bit 0

<b>Legend:</b>	HS = Set by Hardware	HC = Cleared by Hardware
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared      x = Bit is unknown

bit 31-16      **Unimplemented:** Read as '0'

bit 15      **WR:** Write Control bit

This bit is writable when WREN = 1 and the unlock sequence is followed.

1 = Initiate a Flash operation. Hardware clears this bit when the operation completes.

0 = Flash operation complete or inactive

bit 14      **WREN:** Write Enable bit<sup>(1)</sup>

1 = Enable writes to the WR bit and enables the Low-Voltage Detect (LVD) circuit

0 = Disable writes to the WR bit and disables LVD circuit

**Note 1:** This bit is reset by a device Reset.

**2:** This bit is cleared by setting NVMOP<3:0> = 0000, and initiating a Flash operation (i.e., WR).

## Register 5-1: NVMCON: Programming Control Register (Continued)

bit 13 **WRERR:** Write Error bit<sup>(2)</sup>

This bit is read-only and is automatically set by hardware.

- 1 = Program or erase sequence did not complete successfully
- 0 = Program or erase sequence completed normally

bit 12 **LVDERR:** Low-Voltage Detect Error bit (LVD circuit must be enabled)<sup>(2)</sup>

This bit is read-only and is automatically set by hardware.

- 1 = Low-voltage detected (possible data corruption, if WRERR is set)
- 0 = Voltage level is acceptable for programming

bit 11 **LVDSTAT:** Low-Voltage Detect Status bit (LVD circuit must be enabled)<sup>(2)</sup>

This bit is read-only and is automatically set, and cleared by hardware

- 1 = Low-voltage event active
- 0 = Low-voltage event NOT active

bit 10-4 **Unimplemented:** Read as '0'

bit 3-0 **NVMOP<3:0>:** NVM Operation bits

These bits are writable when WREN = 0.

1111 = Reserved

- 
- 
- 

0111 = Reserved

0110 = No operation

0101 = Program Flash (PFM) erase operation: erases PFM, if all pages are not write-protected

0100 = Page erase operation: erases page selected by NVMADDR, if it is not write-protected

0011 = Row program operation: programs row selected by NVMADDR, if it is not write-protected

0010 = No operation

0001 = Word program operation: programs word selected by NVMADDR, if it is not write-protected

0000 = No operation

**Note 1:** This bit is reset by a device Reset.

**2:** This bit is cleared by setting NVMOP<3:0> = 0000, and initiating a Flash operation (i.e., WR).

## 5.2.2 NVMKEY Register

NVMKEY is a write-only register that is used to prevent accidental writes/erasures of Flash or EEPROM memory. To start a programming or an erase sequence, the following steps must be taken in the exact order shown:

1. Write 0xAA996655 to NVMKEY.
2. Write 0x556699AA to NVMKEY.

After this sequence, only the next transaction on the peripheral bus is allowed to write the NVMCON register. In most cases, the user will simply need to set the WR bit in the NVMCON register to start the program or erase cycle. Interrupts should be disabled during the unlock sequence.

**Register 5-2: NVMKEY: Programming Unlock Register**

W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
NVMKEY<31:24>							
bit 31				bit 24			

W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
NVMKEY<23:16>							
bit 23				bit 16			

W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
NVMKEY<15:8>							
bit 15				bit 8			

W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
NVMKEY<7:0>							
bit 7				bit 0			

**Legend:**

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
-n = Value at POR	'1' = Bit is set	'0' = Bit is cleared      x = Bit is unknown

bit 31-0      **NVMKEY<31:0>:** Unlock Register bits  
 These bits are write-only, and read as '0' on any read

**Note:** This register is used as part of the unlock sequence to prevent inadvertent writes to the PFM.

## 5.2.3 NVMADDR Register

The NVM Address register selects the row for Flash memory writes, the address location for word writes, and the page address for Flash memory erase operations.

**Note:** The NVM Address register must be loaded with the physical address of the Flash memory and *not* the virtual address.

**Register 5-3: NVMADDR: Flash Address Register**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMADDR<31:24>							
bit 31				bit 24			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMADDR<23:16>							
bit 23				bit 16			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMADDR<15:8>							
bit 15				bit 8			
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0
NVMADDR<7:0>							
bit 7				bit 0			

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 31-0

**NVMADDR<31:0>:** Flash Address bits

Bulk/Chip/PFM Erase: Address is ignored

Page Erase: Address identifies the page to erase

Row Program: Address identifies the row to program

Word Program: Address identifies the word to program

# PIC32 Family Reference Manual

## 5.2.4 NVMDATA Register

The NVM Data register holds the data to be programmed during Flash Word program operations.

**Register 5-4: NVMDATA: Flash Program Data Register**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMDATA<31:24>							
bit 31				bit 24			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMDATA<23:16>							
bit 23				bit 16			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMDATA<15:8>							
bit 15				bit 8			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMDATA<7:0>							
bit 7				bit 0			

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 31-0      **NVMDATA<31:0>**: Flash Programming Data bits

**Note:** The bits in this register are only reset by a Power-on Reset (POR).



## 5.2.5 NVMSRCADDR Register

The NVM Source Address register selects the source data buffer address in SRAM for performing row programming operations.

**Note:** The address must be word-aligned.

**Register 5-5: NVMSRCADDR: Source Data Address Register**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMSRCADDR<31:24>							
bit 31				bit 24			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMSRCADDR<23:16>							
bit 23				bit 16			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
NVMSRCADDR<15:8>							
bit 15				bit 8			

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0
NVMSRCADDR<7:0>							
bit 7				bit 0			

**Legend:**

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 31-0

**NVMSRCADDR<31:0>:** Source Data Address bits

The system physical address of the data to be programmed into the Flash when the NVMOP<3:0> bits (NVMCON<3:0>) are set to perform row programming.

## 5.3 RUN-TIME SELF-PROGRAMMING (RTSP) OPERATION

Run-Time Self-Programming (RTSP) allows the user code to modify Flash program memory contents. The device Flash memory is divided into two logical Flash partitions: the program Flash memory (PFM), and the boot Flash memory (BFM). The last page in boot Flash memory contains the debug page, which is reserved for use by the debugger tool while debugging.

The program Flash array for the PIC32 device is built up of a series of rows to form a page. Refer to the “**Flash Program Memory**” chapter of the specific device data sheet to determine the row and page sizes. For example, a row may contain 128 32-bit instruction words or 512 bytes. A group of 8 rows compose a page; which, therefore, contains  $8 \times 512 = 4096$  bytes or 1024 instruction words. A page of Flash is the smallest unit of memory that can be erased at a single time. The program Flash array can be programmed in one of two ways:

- Row programming, with 128 instruction words at a time
- Word programming, with 1 instruction word at a time

Performing an RTSP operation while executing (fetching) instructions from program Flash memory, the CPU stalls (waits) until the programming operation is finished. The CPU will not execute any instruction, or respond to interrupts, during this time. If any interrupts occur during the programming cycle, they remain pending until the cycle completes.

Performing an RTSP operation while executing (fetching) instructions from RAM memory, the CPU can continue to execute instructions and respond to interrupts during the programming operation. Any executable code scheduled to execute during the RTSP operation must be placed in RAM memory. This includes the relevant interrupt vector, and the interrupt service routine instructions.

<b>Note:</b>	A minimum VDD requirement for Flash erase and write operations is required. Refer to the “ <b>Electrical Characteristics</b> ” chapter in the specific device data sheet for more information.
--------------	--

### 5.4 LOCK-OUT FEATURE

#### 5.4.1 NVMWREN

A number of mechanisms exists within the device to ensure that inadvertent writes to program Flash do not occur. The WREN bit (NVMCON<14>) should be zero, unless the software intends to write to the program Flash. When WREN = 1, the Flash write control bit, WR (NVMCON<15>), is writable and the Flash LVD circuit is enabled.

#### 5.4.2 NVMKEY

In addition to the write protection provided by the WREN bit, an unlock sequence needs to be performed before the WR bit (NVMCON<15>) can be set. If the WR bit is not set on the next peripheral bus transaction (read or write), WR is locked and the unlock sequence must be restarted.

#### 5.4.3 Unlock Sequence

To unlock Flash operations, perform steps 4 through 8 in order. If sequence is not followed, WR is not set.

1. Suspend or disable all initiators that can access the Peripheral Bus and interrupt the unlock sequence, e.g., DMA and interrupts.
2. Set WREN bit (NVMCON<14>) to allow writes to WR and set NVMOP<3:0> bit (NVMCON<3:0>) to the desired operation with a single store instruction.
3. Wait for LVD to start-up.
4. Load 0xAA996655 to CPU register X.
5. Load 0x556699AA to CPU register Y.
6. Load 0x00008000 to CPU register Z.
7. Store CPU register X to NVMKEY.
8. Store CPU register Y to NVMKEY.
9. Store CPU register Z to NVMCONSET.
10. Wait for WR bit (NVMCON<15>) to be cleared.
11. Clear the WREN bit (NVMCON<14>).
12. Check the WRERR (NVMCON<13>) and LVDERR (NVMCON<12>) bits to ensure that the program/erase sequence completed successfully.

When the WR bit is set, the program/erase sequence starts and the CPU is unable to execute from Flash memory for the duration of the sequence.

## Example 5-1: Unlock Example

```
unsigned int NVMUnlock (unsigned int nvmpop)
{
    unsigned int status;

    // Suspend or Disable all Interrupts
    asm volatile ("di %0" : "=r" (status));

    // Enable Flash Write/Erase Operations and Select
    // Flash operation to perform
    NVMCON = nvmpop;

    // Write Keys
    NVMKEY = 0xAA996655;
    NVMKEY = 0x556699AA;

    // Start the operation using the Set Register
    NVMCONSET = 0x8000;

    // Wait for operation to complete
    while (NVMCON & 0x8000);

    // Restore Interrupts
    if (status & 0x00000001)
        asm volatile ("ei");
    else
        asm volatile ("di");

    // Disable NVM write enable
    NVMCONCLR = 0x0004000;

    // Return WRERR and LVDERR Error Status Bits
    return (NVMCON & 0x3000);
}
```

### 5.5 WORD PROGRAMMING SEQUENCE

The smallest block of data that can be programmed in a single operation is one 32-bit word. The data to be programmed must be written to the NVMDATA register, and the address of the word must be loaded into the NVMADDR register before the programming sequence is initiated. The instruction word at the location pointed to by the NVMADDR register is then programmed.

A program sequence comprises the following steps:

1. Write 32-bit data to be programmed to the NVMDATA register.
2. Load the NVMADDR register with the address to be programmed.
3. Run the unlock sequence using the Word Program command (see [Section 5.4.3 “Unlock Sequence”](#)).

The program sequence completes, and the WR bit (NVMCON<15>) is cleared by hardware.

#### Example 5-2: Word Program Example

```
unsigned int NVMWriteWord (void* address, unsigned int data)
{
    unsigned int res;

    // Load data into NVMDATA register
    NVMDATA = data;

    // Load address to program into NVMADDR register
    NVMADDR = (unsigned int) address;

    // Unlock and Write Word
    res = NVMUnlock (0x4001);

    // Return Result
    return res;
}
```

## 5.6 ROW PROGRAMMING SEQUENCE

The largest block of data that can be programmed is 1 row, which equals to 512 bytes of data (refer to the “**Flash Program Memory**” chapter in the specific device data sheet to determine the row size). The row of data must first be loaded into a buffer in SRAM. The NVMADDR register then points to the Flash address where the Flash controller will start programming the row of data.

**Note:** The Flash controller ignores the sub-row address bits and always starts programming at the beginning of a row.

A row program sequence comprises the following steps:

1. Write the entire row of data to be programmed into system SRAM. The source address must be word-aligned.
2. Set the NVMADDR register with the start address of the Flash row to be programmed.
3. Set the NVMSRCADDR register with the physical source address from step 1.
4. Run the unlock sequence using the Row Program command (see [5.4.3 “Unlock Sequence”](#)).
5. The program sequence completes, and the WR bit (NVMCON<15>) is cleared by hardware.

### Example 5-3: Row Program Example

```
unsigned int NVMWriteRow (void* address, void* data)
{
    unsigned int res;

    // Set NVMADDR to Start Address of row to program
    NVMADDR = (unsigned int) address;

    // Set NVMSRCADDR to the SRAM data buffer Address
    NVMSRCADDR = (unsigned int) data;

    // Unlock and Write Row
    res = NVMUnlock(0x4003);

    // Return Result
    return res;
}
```

### 5.7 PAGE ERASE SEQUENCE

A page erase performs an erase of a single page of either PFM or BFM. Refer to the specific device data sheet for the page size. The page to be erased is selected using the NVMADDR register.

**Note:** The lower bits of the address are ignored in page selection.

A page of Flash can only be erased if its associated page write protection is not enabled.

- All BFM pages are affected by the Boot write protection Configuration bit
- PFM pages are affected by the Program Flash write protection Configuration bits

If in Mission mode, the application must not be executing from the erased page.

A page erase sequence comprises the following steps:

1. Set the NVMADDR register with the address of the page to be erased.
2. Run the unlock sequence using the desired Erase command (see [5.4.3 “Unlock Sequence”](#)).
3. The erase sequence completes and the WR bit (NVMCON<15>) is cleared by hardware.

#### Example 5-4: Page Erase Example

```
unsigned int NVMErasePage(void* address)
{
    unsigned int res;

    // Set NVMADDR to the Start Address of page to erase
    NVMADDR = (unsigned int) address;

    // Unlock and Erase Page
    res = NVMUnlock(0x4004);

    // Return Result
    return res;
}
```

### 5.8 PROGRAM FLASH MEMORY ERASE SEQUENCE

It is possible to erase the entire PFM area. This mode leaves the boot Flash intact and is intended to be used by a field-upgradeable device.

The program Flash can be erased if all pages in the program Flash are not write-protected.

**Note:** The application must *not* be executing from the PFM address range.

A PFM erase sequence comprises the following steps:

1. Run the unlock sequence using the program Flash memory erase command (see [5.4.3 “Unlock Sequence”](#)).
2. The erase sequence completes and the WR bit (NVMCON<15>) is cleared by hardware.

#### Example 5-5: Program Flash Erase Example

```
unsigned int NVMErasePFM(void)
{
    unsigned int res;

    // Unlock and Erase Program Flash
    res = NVMUnlock(0x4005);

    // Return Result
    return res;
}
```

## 5.9 OPERATION IN POWER-SAVING AND DEBUG MODES

### 5.9.1 Operation in Sleep Mode

When a PIC32 device enters Sleep mode, the system clock is disabled. The Flash controller does not function in Sleep mode. If entry into Sleep mode occurs while an NVM operation is in progress, the device will not go into Sleep mode until the NVM operation is complete.

### 5.9.2 Operation in Idle Mode

Idle mode has no effect on the Flash controller module when a programming operation is active. The CPU continues to be stalled until the programming operation completes.

### 5.9.3 Operation in Debug Mode

The Flash controller does not provide debug freeze capability, and therefore, has no effect on the Flash controller module when a programming operation is active. The CPU continues to be stalled until the programming operation completes. Interrupting the normal programming sequence could cause the device to latch-up. The only exception to this is the NVMKEY unlock sequence, which is suspended when in Debug mode, allowing the user to single-step through the unlock sequence.

## 5.10 EFFECTS OF VARIOUS RESETS

### 5.10.1 Device Reset

Only the NVMCON bits for WREN and LVDSTAT are reset on a device Reset. All other SFR bits are only reset by a POR; however, the state of the NVMKEY is reset by a device Reset.

### 5.10.2 Power-on Reset

All Flash controller registers are forced to their reset states upon a POR.

### 5.10.3 Watchdog Timer Reset

All Flash controller registers are unchanged upon a Watchdog Timer Reset.



### 5.11 INTERRUPTS

The Flash controller can generate an interrupt reflecting the events that occur during the programming operations. The following interrupt can be generated:

- Flash Control Event Interrupt (FCEIF)

The interrupt flag must be cleared in software. The Flash controller is enabled as a source of interrupt via the following bit:

- Flash Controller Interrupt Enable (FCEIE)

The interrupt priority-level bits and interrupt subpriority-level bits must also be configured:

- FCEIP
- FCEIS

Refer to **Section 8. “Interrupts”** (DS61108) in the *“PIC32 Family Reference Manual”* for details.

#### 5.11.1 Interrupt Configuration

The Flash controller module has a dedicated interrupt flag bit, FCEIF, and a corresponding interrupt enable/mask bit, FCEIE.

These two bits determine the source of an interrupt and enable or disable an individual interrupt source. All the interrupt sources for a specific Flash controller module share one interrupt vector.

In addition, the FCEIF bit will be set without regard to the state of the corresponding enable bit, and the FCEIF bit can be polled by software if desired.

The FCEIE bit is used to define the behavior of the Vector Interrupt Controller (VIC) when a corresponding FCEIF bit is set. When the corresponding FCEIE bit is clear, the VIC module does not generate a CPU interrupt for the event. If the FCEIE bit is set, the VIC module will generate an interrupt to the CPU when the corresponding FCEIF bit is set (subject to the priority and subpriority as outlined in the following paragraphs).

It is the responsibility of the user’s software routine that services a particular interrupt to clear the appropriate Interrupt Flag bit before the service routine is complete.

The priority of the Flash Controller module can be set independently with the FCEIP<2:0> bits. This priority defines the priority group to which the interrupt source is assigned. The priority groups range from a value of 7 (the highest priority), to a value of 0, which does not generate an interrupt. An interrupt being serviced is preempted by an interrupt in a higher priority group.

The subpriority bits allow setting the priority of a interrupt source within a priority group. The values of the subpriority, FCEIS<1:0>, range from 3 (the highest priority), to 0 the lowest priority. An interrupt with the same priority group but having a higher subpriority value, does not preempt a lower subpriority interrupt that is in progress.

The priority group and subpriority bits allow more than one interrupt source to share the same priority and subpriority. If simultaneous interrupts occur in this configuration, the natural order of the interrupt sources within a priority/subpriority group pair determine the interrupt generated. The natural priority is based on the vector numbers of the interrupt sources. The lower the vector number, the higher the natural priority of the interrupt. Any interrupts that are overridden by natural order generate their respective interrupts based on priority, subpriority, and natural order, after the interrupt flag for the current interrupt is cleared.

After an enabled interrupt is generated, the CPU jumps to the vector assigned to that interrupt. The vector number for the interrupt is the same as the natural order number. Then, the CPU begins executing code at the vector address. The user’s code at this vector address should perform any application-specific operations, clear the FCEIF interrupt flag, and then exit.

For more information on interrupts and the vector address table details, refer to **Section 8. “Interrupts”** (DS61108) in the *“PIC32 Family Reference Manual”* and the **“Interrupt Controller”** chapter of the specific device data sheet.

## 5.12 RELATED APPLICATION NOTES

This section lists application notes that are related to this section of the manual. These application notes may not be written specifically for the PIC32 device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to Flash Programming include the following:

Title	Application Note #
No related application notes at this time.	N/A

<p><b>Note:</b> Please visit the Microchip web site (<a href="http://www.microchip.com">www.microchip.com</a>) for additional application notes and code examples for the PIC32 family of devices.</p>
--

### 5.13 REVISION HISTORY

#### Revision A (September 2007)

This is the initial released version of this document.

#### Revision B (October 2007)

Updated document to remove Confidential status.

#### Revision C (April 2008)

Revised status to Preliminary; Revised U-0 to r-x.

#### Revision D (June 2008)

Revised Register 5-1, bit 14 NVMWREN; Add footnote 1 to Registers 5-12 through 5-14; Add note to Section 5.3; Revise Section 5.4.1; Revised Example 5-1; Change Reserved bits "Maintain as" to "Write".

#### Revision E (December 2010)

This revision includes the following updates:

- Minor updates to the text and formatting have been incorporated throughout the document
- Added Notes 1, 2 and 3, which describe the Clear, Set and Invert registers to the following:
  - Table 5-1: Flash Controller SFR Summary
  - Register 5-1: NVMCON: Programming Control Register<sup>(1,2,3)</sup>
  - Register 5-3: NVMADDR: Flash Address Register<sup>(1,2,3)</sup>
- Removed all Clear, Set and Invert register descriptions
- Removed all Interrupt register references
- Renamed the following NVMCON register bit names were changed throughout the document:
  - NVMWR was renamed to WR
  - NVMWREN was renamed to WREN
  - NVMERR was renamed to WRERR
- Updated the third paragraph and added a new (fourth) paragraph to **5.3 "Run-Time Self-Programming (RTSP) Operation"**
- Updated the unlock Flash operations sequence by adding a new step 3 (see **5.4.3 "Unlock Sequence"**)
- Updated the code in the Unlock Example (see Example 5-1)
- Removed Table 5-3

#### Revision F (July 2012)

This revision includes the following updates:

- Updated Note 1 and removed Notes 2 and 3 from the Flash Controller SFR Register Summary (see [Table 5-1](#))
- Updated NVMCON (see [Register 5-1](#))
- Removed Notes 1, 2, and 3 from NVMADDR (see [Register 5-3](#))
- Updated the second paragraph of **5.3 "Run-Time Self-Programming (RTSP) Operation"**
- Updated the first paragraph of **5.6 "Row Programming Sequence"**
- Updated the first paragraph of **5.7 "Page Erase Sequence"**
- Minor updates to the text and formatting have been incorporated throughout the document

NOTES:

---

**Note the following details of the code protection feature on Microchip devices:**

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

---

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

#### **Trademarks**

The Microchip name and logo, the Microchip logo, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC<sup>32</sup> logo, rPIC and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


FilterLab, Hampshire, HI-TECH C, Linear Active Thermistor, MXDEV, MXLAB, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, dsSPEAK, ECAN, ECONOMONITOR, FanSense, HI-TIDE, In-Circuit Serial Programming, ICSP, Mindi, MiWi, MPASM, MPLAB Certified logo, MPLIB, MPLINK, mTouch, Omniscent Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, REAL ICE, rLAB, Select Mode, Total Endurance, TSHARC, UniWinDriver, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2007-2012, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

ISBN: 978-1-62076-464-0

**QUALITY MANAGEMENT SYSTEM  
CERTIFIED BY DNV  
= ISO/TS 16949 =**

*Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.*

## Worldwide Sales and Service

### AMERICAS

**Corporate Office**  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-792-7200  
Fax: 480-792-7277  
Technical Support:  
<http://www.microchip.com/support>  
Web Address:  
[www.microchip.com](http://www.microchip.com)

**Atlanta**  
Duluth, GA  
Tel: 678-957-9614  
Fax: 678-957-1455

**Boston**  
Westborough, MA  
Tel: 774-760-0087  
Fax: 774-760-0088

**Chicago**  
Itasca, IL  
Tel: 630-285-0071  
Fax: 630-285-0075

**Cleveland**  
Independence, OH  
Tel: 216-447-0464  
Fax: 216-447-0643

**Dallas**  
Addison, TX  
Tel: 972-818-7423  
Fax: 972-818-2924

**Detroit**  
Farmington Hills, MI  
Tel: 248-538-2250  
Fax: 248-538-2260

**Indianapolis**  
Noblesville, IN  
Tel: 317-773-8323  
Fax: 317-773-5453

**Los Angeles**  
Mission Viejo, CA  
Tel: 949-462-9523  
Fax: 949-462-9608

**Santa Clara**  
Santa Clara, CA  
Tel: 408-961-6444  
Fax: 408-961-6445

**Toronto**  
Mississauga, Ontario,  
Canada  
Tel: 905-673-0699  
Fax: 905-673-6509

### ASIA/PACIFIC

**Asia Pacific Office**  
Suites 3707-14, 37th Floor  
Tower 6, The Gateway  
Harbour City, Kowloon  
Hong Kong  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**Australia - Sydney**  
Tel: 61-2-9868-6733  
Fax: 61-2-9868-6755

**China - Beijing**  
Tel: 86-10-8569-7000  
Fax: 86-10-8528-2104

**China - Chengdu**  
Tel: 86-28-8665-5511  
Fax: 86-28-8665-7889

**China - Chongqing**  
Tel: 86-23-8980-9588  
Fax: 86-23-8980-9500

**China - Hangzhou**  
Tel: 86-571-2819-3187  
Fax: 86-571-2819-3189

**China - Hong Kong SAR**  
Tel: 852-2401-1200  
Fax: 852-2401-3431

**China - Nanjing**  
Tel: 86-25-8473-2460  
Fax: 86-25-8473-2470

**China - Qingdao**  
Tel: 86-532-8502-7355  
Fax: 86-532-8502-7205

**China - Shanghai**  
Tel: 86-21-5407-5533  
Fax: 86-21-5407-5066

**China - Shenyang**  
Tel: 86-24-2334-2829  
Fax: 86-24-2334-2393

**China - Shenzhen**  
Tel: 86-755-8203-2660  
Fax: 86-755-8203-1760

**China - Wuhan**  
Tel: 86-27-5980-5300  
Fax: 86-27-5980-5118

**China - Xian**  
Tel: 86-29-8833-7252  
Fax: 86-29-8833-7256

**China - Xiamen**  
Tel: 86-592-2388138  
Fax: 86-592-2388130

**China - Zhuhai**  
Tel: 86-756-3210040  
Fax: 86-756-3210049

### ASIA/PACIFIC

**India - Bangalore**  
Tel: 91-80-3090-4444  
Fax: 91-80-3090-4123

**India - New Delhi**  
Tel: 91-11-4160-8631  
Fax: 91-11-4160-8632

**India - Pune**  
Tel: 91-20-2566-1512  
Fax: 91-20-2566-1513

**Japan - Osaka**  
Tel: 81-66-152-7160  
Fax: 81-66-152-9310

**Japan - Yokohama**  
Tel: 81-45-471-6166  
Fax: 81-45-471-6122

**Korea - Daegu**  
Tel: 82-53-744-4301  
Fax: 82-53-744-4302

**Korea - Seoul**  
Tel: 82-2-554-7200  
Fax: 82-2-558-5932 or  
82-2-558-5934

**Malaysia - Kuala Lumpur**  
Tel: 60-3-6201-9857  
Fax: 60-3-6201-9859

**Malaysia - Penang**  
Tel: 60-4-227-8870  
Fax: 60-4-227-4068

**Philippines - Manila**  
Tel: 63-2-634-9065  
Fax: 63-2-634-9069

**Singapore**  
Tel: 65-6334-8870  
Fax: 65-6334-8850

**Taiwan - Hsin Chu**  
Tel: 886-3-5778-366  
Fax: 886-3-5770-955

**Taiwan - Kaohsiung**  
Tel: 886-7-536-4818  
Fax: 886-7-330-9305

**Taiwan - Taipei**  
Tel: 886-2-2500-6610  
Fax: 886-2-2508-0102

**Thailand - Bangkok**  
Tel: 66-2-694-1351  
Fax: 66-2-694-1350

### EUROPE

**Austria - Wels**  
Tel: 43-7242-2244-39  
Fax: 43-7242-2244-393

**Denmark - Copenhagen**  
Tel: 45-4450-2828  
Fax: 45-4485-2829

**France - Paris**  
Tel: 33-1-69-53-63-20  
Fax: 33-1-69-30-90-79

**Germany - Munich**  
Tel: 49-89-627-144-0  
Fax: 49-89-627-144-44

**Italy - Milan**  
Tel: 39-0331-742611  
Fax: 39-0331-466781

**Netherlands - Drunen**  
Tel: 31-416-690399  
Fax: 31-416-690340

**Spain - Madrid**  
Tel: 34-91-708-08-90  
Fax: 34-91-708-08-91

**UK - Wokingham**  
Tel: 44-118-921-5869  
Fax: 44-118-921-5820