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 RTSP Operation 5-17
5.4 Lock-Out Feature 5-18
5.5 Word Programming Sequence 5-20
5.6 Row Programming Sequence 5-21
5.7 Page Erase Sequence 5-22
5.8 Program Flash Memory Erase Sequence 5-23
5.9 Operation in Power-Saving and DEBUG Modes 5-24
5.10 Effects of Various Resets 5-24
5.11 Interrupts 5-25
5.12 Related Application Notes 5-27
5.13 Revision History 5-28

5.1 INTRODUCTION

This section describes techniques for programming the Flash memory. PIC32MX 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, allows much faster programming than RTSP
- EJTAG Programming – 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 Programming Specification document, which can be downloaded from the Microchip web site at www.microchip.com.

5.2 CONTROL REGISTERS

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

- NVMCON: Nonvolatile Memory Control Register

  NVMCONCLR, NVMCONSET, NVMCONINV: Atomic Bit Manipulation, Write-only Registers for NVMCON

- NVMKEY: Nonvolatile Memory Key Register
- NVMADDR: Nonvolatile Memory Address Register

  NVMADDRCLR, NVMADDRSET, NVMADDRINV: Atomic Bit Manipulation, Write-only Registers for NVMADDR

- NVMDATA: Nonvolatile Memory Data Register
- NVMSRCADDR: Nonvolatile Memory SRAM Source Address Register
- IFSx: Interrupt Flag Status Registers

  IFSxCLR, IFSxSET, IFSxINV: Atomic Bit Manipulation, Write-only Registers for IFSx

- IECx: Interrupt Enable Control Registers

  IECxCLR, IECxSET, IECxINV: Atomic Bit Manipulation, Write-only Registers for IECx

- IPCx: Interrupt Priority Control Registers
- IPCxCLR, IPCxSET, IPCxINV: Atomic Bit Manipulation, Write-only Registers for IPCx

The following table provides a brief summary of all the Flash-programming-related registers. Corresponding registers appear after the summary, followed by a detailed description of each register.

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	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	NVMWR	NVMWREN	NVMERR	LVDERR	LVDSTAT	—	—	—
	7:0	—	—	—	—	NVMOP<3:0>			
NVMCONCLR	31:0	Write clears selected bits in NVMCON, read yields undefined value							
NVMCONSET	31:0	Write sets selected bits in NVMCON, read yields undefined value							
NVMCONINV	31:0	Write inverts selected bits in NVMCON, read yields undefined value							
NVMKEY	31:24	NVMKEY<31:24>							
	23:16	NVMKEY<23:16>							
	15:8	NVMKEY<15:8>							
	7:0	NVMKEY<7:0>							
NVMADDR	31:24	NVMADDR<31:24>							
	23:16	NVMADDR<23:16>							
	15:8	NVMADDR<15:8>							
	7:0	NVMADDR<7:0>							
NVMADDRCLR	31:0	Write clears selected bits in NVMADDR, read yields undefined value							
NVMADDRSET	31:0	Write sets selected bits in NVMADDR, read yields undefined value							
NVMADDRINV	31:0	Write inverts selected bits in NVMADDR, read yields undefined value							

Table 5-1: Flash Controller SFR Summary (Continued)

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
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>							

Table 5-2: Flash Controller Interrupt SFR Summary

IFS1	31:24	—	—	—	—	—	—	USBIF	FCEIF
	23:16	—	—	—	—	DMA3IF	DMA2IF	DMA1IF	DMA0IF
	15:8	RTCCIF	FSCMIF	I2C2MIF	I2C2SIF	I2C2BIF	U2TXIF	U2RXIF	U2EIF
	7:0	SPI2RXIF	SPI2TXIF	SPI2EIF	CMP2IF	CMP1IF	PMPIF	AD1IF	CNIF
IFS1CLR	31:0	Write clears the selected bits in IFS1, read yields undefined value							
IFS1SET	31:0	Write sets the selected bits in IFS1, read yields undefined value							
IFS1INV	31:0	Write inverts the selected bits in IFS1, read yields undefined value							
IEC1	31:24	—	—	—	—	—	—	USBIE	FCEIE
	23:16	—	—	—	—	DMA3IE	DMA2IE	DMA1IE	DMA0IE
	15:8	RTCCIE	FSCMIE	I2C2MIE	I2C2SIE	I2C2BIE	U2TXIE	U2RXIE	U2EIE
	7:0	SPI2RXIE	SPI2TXIE	SPI2EIE	CMP2IE	CMP1IE	PMPIE	AD1IE	CNIE
IEC1CLR	31:0	Write clears the selected bits in IEC1, read yields undefined value							
IEC1SET	31:0	Write sets the selected bits in IEC1, read yields undefined value							
IEC1INV	31:0	Write inverts the selected bits in IEC1, read yields undefined value							
IPC11	31:24	—	—	—	—	—	—	—	—
	23:16	—	—	—	—	—	—	—	—
	15:8	—	—	—	—	—	—	—	—
	7:0	—	—	—	FCEIP<2:0>			FCEIS<1:0>	
IPC11CLR	31:0	Write clears the selected bits in IPC11, read yields undefined value							
IPC11SET	31:0	Write sets the selected bits in IPC11, read yields undefined value							
IPC11INV	31:0	Write inverts the selected bits in IPC11, read yields undefined value							

Register 5-1: NVMCON: Programming Control Register

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—	—	—	—	—	—
bit 23							bit 16

R/W-0	R/W-0	R-0	R-0	R-x	r-x	r-x	r-x
NVMWR	NVMWREN	NVMERR	LVDERR	LVDSTAT	—	—	—
bit 15							bit 8

r-x	r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	NVMOP3	NVMOP2	NVMOP1	NVMOP0
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-16 **Reserved:** Write ‘0’; ignore read

bit 15 **NVMWR:** Write Control bit
This bit is writable when NVMWREN = 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 **NVMWREN:** Write Enable bit
1 = Enable writes to NVMWR bit and enables LVD circuit
0 = Disable writes to NVMWR bit and disables LVD circuit
Note: This is the only bit in this register reset by a device Reset.

bit 13 **NVMERR:** Write Error bit
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
Note: Cleared by setting NVMOP==0000b, and initiating a Flash operation (i.e., NVMWR).

bit 12 **LVDERR:** Low Voltage Detect Error Bit (LVD circuit must be enabled)
This bit is read-only and is automatically set by hardware
1 = Low voltage detected (possible data corruption, if NMVERR is set)
0 = Voltage level is acceptable for programming
Note: Cleared by setting NVMOP==0000b, and initiating a Flash operation (i.e., NVMWR).

bit 11 **LVDSTAT:** Low-Voltage Detect Status bit (LVD circuit must be enabled)
This bit is read-only and is automatically set, and cleared, by hardware
1 = Low voltage event active
0 = Low voltage event NOT active
Note: Cleared by setting NVMOP==0000b, and initiating a Flash operation (i.e., NVMWR).

bit 10-4 **Reserved:** Write ‘0’; ignore read

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

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

These bits are writeable when NVMWREN = 1 and the unlock sequence is followed

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

Register 5-2: NVMCONCLR: Programming Control Clear Register

Write clears selected bits in NVMCON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in NVMCON**

A write of '1' in one or more bit positions clears the corresponding bit(s) in NVMCON register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `NVMCONCLR = 0x00008001` will clear bits 15 and 0 in NVMCON register.

Register 5-3: NVMCONSET: Programming Control Set Register

Write sets selected bits in NVMCON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Sets selected bits in NVMCON**

A write of '1' in one or more bit positions sets the corresponding bit(s) in NVMCON register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `NVMCONSET = 0x00008001` will set bits 15 and 0 in NVMCON register.

Register 5-4: NVMCONINV: Programming Control Invert Register

Write inverts selected bits in NVMCON, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in NVMCON**

A write of '1' in one or more bit positions inverts the corresponding bit(s) in NVMCON register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: `NVMCONINV = 0x00008001` will invert bits 15 and 0 in NVMCON register.

Register 5-5: 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	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = 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.

Register 5-6: 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-x	r-x
NVMADDR<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-3 **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

bit 2-0 **Reserved:** Write ‘0’; ignore read

Register 5-7: NVMADDRCLR: Flash Address Clear Register

Write clears selected bits in NVMADDR, read yields undefined value	
bit 31	bit 0

bit 31-0 **Clears selected bits in NVMADDR**

A write of '1' in one or more bit positions clears the corresponding bit(s) in NVMADDR register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: NVMADDRCLR = 0x00008001 will clear bits 15 and 0 in NVMADDR register.

Register 5-8: NVMADDRSET: Flash Address Set Register

Write sets selected bits in NVMADDR, read yields undefined value	
bit 31	bit 0

bit 31-0 **Sets selected bits in NVMADDR**

A write of '1' in one or more bit positions sets the corresponding bit(s) in NVMADDR register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: NVMADDRSET = 0x00008001 will set bits 15 and 0 in NVMADDR register.

Register 5-9: NVMADDRINV: Flash Address Invert Register

Write inverts selected bits in NVMADDR, read yields undefined value	
bit 31	bit 0

bit 31-0 **Inverts selected bits in NVMADDR**

A write of '1' in one or more bit positions inverts the corresponding bit(s) in NVMADDR register and does not affect unimplemented or read-only bits. A write of '0' will not affect the register.

Example: NVMADDRINV = 0x00008001 will invert bits 15 and 0 in NVMADDR register.

Register 5-10: 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	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

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

Note: These bits are only reset by a Power-on Reset (POR).

Register 5-11: 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-x	r-x
NVMSRCADDR<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	P = Programmable bit	r = Reserved bit
U = Unimplemented bit	-n = Bit Value at POR: (‘0’, ‘1’, x = Unknown)		

bit 31-3 **NVMSRCADDR<31:0>:** Source Data Address bits
The system physical address of the data to be programmed into the Flash when NVMCON.NVMOP is set to perform row programming

bit 2-0 **Reserved:** Write ‘0’; ignore read

Register 5-12: IFS1: Interrupt Flag Status Register 1(1)

r-x	r-x	r-x	r-x	r-x	r-x	R/W-0	R/W-0
—	—	—	—	—	—	USBIF	FCEIF
bit 31							bit 24

r-x	r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	DMA3IF	DMA2IF	DMA1IF	DMA0IF
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
RTCCIF	FSCMIF	I2C2MIF	I2C2SIF	I2C2BIF	U2TXIF	U2RXIF	U2EIF
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
SPI2RXIF	SPI2TXIF	SPI2EIF	CMP2IF	CMP1IF	PMPIF	AD1IF	CNIF
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
HC = Hardware clear	HS = Hardware set	C = Clearable by software	
-n = Bit Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 24 **FCEIF:** Flash Control Event Interrupt Flag bit

1 = Interrupt request has occurred
0 = No interrupt request has occurred

Note 1: Shaded bit names in this Interrupt register control other PIC32MX peripherals and are not related to the NVM.

Register 5-13: IEC1: Interrupt Enable Control Register 1(1)

r-x	r-x	r-x	r-x	r-x	r-x	R/W-0	R/W-0
—	—	—	—	—	—	USBIE	FCEIE
bit 31							bit 24

r-x	r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	DMA3IE	DMA2IE	DMA1IE	DMA0IE
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
RTCCIE	FSCMIE	I2C2MIE	I2C2SIE	I2C2BIE	U2TXIE	U2RXIE	U2EIE
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
SPI2RXIE	SPI2TXIE	SPI2EIE	CMP2IE	CMP1IE	PMPIE	AD1IE	CNIE
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
HC = Hardware clear	HS = Hardware set	C = Clearable by software	
-n = Bit Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 24 **FCEIE:** Flash Control Event Interrupt Enable bit

1 = Interrupt is enabled
0 = Interrupt is disabled

Note 1: Shaded bit names in this Interrupt register control other PIC32MX peripherals and are not related to the NVM.

Register 5-14: IPC11: Interrupt Priority Control Register 11(1)

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—					
bit 31							bit 24

r-x	r-x	r-x	r-x	r-x	r-x	r-x	r-x
—	—	—					
bit 23							bit 16

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	USBIP<2:0>			USBIS<1:0>	
bit 15							bit 8

r-x	r-x	r-x	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FCEIP<2:0>			FCEIS<1:0>	
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
HC = Hardware clear	HS = Hardware set	C = Clearable by software	
-n = Bit Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

bit 4-2 **FCEIP<2:0>:** Flash Control Event INterrupt Priority bits
- 111 = Interrupt Priority is 7
- 110 = Interrupt Priority is 6
- 101 = Interrupt Priority is 5
- 100 = Interrupt Priority is 4
- 011 = Interrupt Priority is 3
- 010 = Interrupt Priority is 2
- 001 = Interrupt Priority is 1
- 000 = Interrupt is disabled

bit 1-0 **FCEIS<1:0>:** Flash Control Event Interrupt Subpriority bits
- 11 = Interrupt Subpriority is 3
- 10 = Interrupt Subpriority is 2
- 01 = Interrupt Subpriority is 1
- 00 = Interrupt Subpriority is 0

Note 1: Shaded bit names in this Interrupt register control other PIC32MX peripherals and are not related to the NVM.

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 5-3.

Table 5-3: NVMCON Register Values

Operation	NVMCON Value
Page Erase	0x8004
Program Word	0x8001
Program Row	0x8003
NOP	0x8000

5.2.2 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.

5.2.3 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 NVMWR bit in the NVMCON register to start the program or erase cycle. Interrupts should be disabled during the unlock sequence.

5.2.4 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.

5.3 RTSP OPERATION

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 PIC32MX device is built up of a series of rows. A row contains 128 32-bit instruction words or 512 bytes. A group of 8 rows compose a page; which, therefore, contains 8 × 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.

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.

Note:	A minimum V_{DD} requirement for Flash erase and write operations is required. Refer to the specific device data sheet for further details.

5.4 LOCK-OUT FEATURE

5.4.1 NVMWREN

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

5.4.2 NVMKEY

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

5.4.3 Unlock Sequence

To unlock Flash operations, steps 3 through 8 below must be performed exactly in order. If the sequence is not followed exactly, NVMWR 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 NVMWREN (NVMCON<14>) to allow writes to NVMWR and set NVMOP<3:0> (NVMCON<3:0>) to the desired operation with a single store instruction.
3. Load 0xAA996655 to CPU register X.
4. Load 0x556699AA to CPU register Y.
5. Load 0x00008000 to CPU register Z.
6. Store CPU register X to NVMKEY.
7. Store CPU register Y to NVMKEY.
8. Store CPU register Z to NVMCONSET.
9. Wait for NVMWR (NVMCON<15>) bit to be clean.
10. Clear the NVMWREN (NVMCON<14>) bit.
11. Check the NVMERR (NVMCON<13>) and LVDERR (NVMCON<12>) bits to ensure that the program/erase sequence completed successfully.

When the NVMWR 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 nvmop)
{
    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 = nvmop;

    // 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”);

    // Return NVMERR 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 NVMADDR 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 NVMWR (NVMCON<15>) bit 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 equates to 512 bytes of data. 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 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 **Section 5.4.3 “Unlock Sequence”**).
5. The program sequence completes, and the NVMWR (NVMCON<15>) bit 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, which equates to 4096 bytes. 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 **Section 5.4.3 “Unlock Sequence”**).

The erase sequence completes and the NVMWR (NVMCON<15>) bit 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 **Section 5.4.3 “Unlock Sequence”**)

The erase sequence completes and the NVMWR (NVMCON<15>) bit 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

Note:	In this manual, a distinction is made between a power mode as it is used in a specific module, and a power mode as it is used by the device, e.g., Sleep mode of the Comparator and SLEEP mode of the CPU. To indicate which type of power mode is intended, uppercase and lowercase letters (Sleep, Idle, Debug) signify a module power mode, and all uppercase letters (SLEEP, IDLE, DEBUG) signify a device power mode.

5.9.1 Operation in SLEEP Mode

When the PIC32MX 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, then the device will not go into sleep 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.

Note:	The FRZ bit is readable and writable only when the CPU is executing in Debug Exception mode. In all other modes, the FRZ bit reads as '0'. If FRZ bit is changed during DEBUG mode, the new value does not take effect until the current Debug Exception mode is exited and re-entered. During the Debug Exception mode, the FRZ bit reads the state of the peripheral when entering DEBUG mode.

5.10 EFFECTS OF VARIOUS RESETS

5.10.1 Device Reset

Only the NVMCON bits for NVMWREN and LVDSTAT are reset on a device Reset. All other SFR bits are only reset by POR. The state of the NUMKEY is reset by a device Reset however.

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 has the ability to generate an interrupt reflecting the events that occur during the programming operations. The following interrupt can be generated:

Flash Control Event Interrupt FCEIF (IFS1<24>

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

- FCEIE (IE1<24>)

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

- FCEIP (IPC11<2:0> and FCEIS (IPC11<1:0>)

Refer to **Section 8. "Interrupts"** in this manual for details.

5.11.1 Interrupt Configuration

The Flash Controller module has the following dedicated interrupt flag bit.

- FCEIF

The Flash Controller module also has the following corresponding interrupt enable/mask bit:

- FCEIE

The bits determine the source of an interrupt and enable or disable an individual interrupt source. Note that all the interrupt sources for a specific Flash Controller module share just one interrupt vector.

Note that the FCEIF bit will be set without regard to the state of the corresponding enable bit, and the IF 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 IE bit is clear the VIC module does not generate a CPU interrupt for the event. If the IE bit is set, the VIC module will generate an interrupt to the CPU when the corresponding IF 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. Refer to **Section 8. "Interrupts"** in this manual for the vector address table details and more information on interrupts.

Table 5-4: UART Interrupt Vectors for Various Offsets with EBASE = 0x8000:0000

Interrupt	Vector/ Natural Order	IRQ Number	Vector Address IntCtl.VS = 0x01	Vector Address IntCtl.VS = 0x02	Vector Address IntCtl.VS = 0x04	Vector Address IntCtl.VS = 0x08	Vector Address IntCtl.VS = 0x10
FCE	43	56	80000760	80000CC0	80001780	80002D00	80005800

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 PIC32MX device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Flash module include the following:

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

Note: Please visit the Microchip web site (www.microchip.com) for additional application notes and code examples for the PIC32MX family of devices.

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 5-12-5-14; Add note to Section 5.3; Revise Section 5.4.1; Revised Example 5-1; Change Reserved bits “Maintain as” to “Write”.

