Section 5. Flash Programming

HIGHLIGHTS

This section of the manual contains the following topics:

5

Flash Programming

> **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 dsPIC33F/PIC24H devices.
>
> Please consult the note at the beginning of the **"Flash Program Memory"** 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 the technique for programming Flash program memory. The dsPIC33F/PIC24H families of devices have an internal programmable Flash memory for executing user codes. There are two methods to program this memory:

- Run-Time Self-Programming (RTSP)
- In-Circuit Serial Programming™ (ICSP™)

This section describes RTSP programming, which is performed by the user-assigned software.

ICSP is performed using a serial data connection to the device and allows faster programming than RTSP. The ICSP protocol is defined in the *"dsPIC33F/PIC24H Flash Programming Specification"* (DS70152), which can be downloaded from the Microchip web site.

5.2 TABLE INSTRUCTION OPERATION

The table instructions provide one method of transferring data between the program memory space and the data memory space of dsPIC33F/PIC24H devices. This section provides a summary of the table instructions used during programming of the Flash program memory. There are four basic table instructions:

- `TBLRDL`: Table Read Low
- `TBLRDH`: Table Read High
- `TBLWTL`: Table Write Low
- `TBLWTH`: Table Write High

The `TBLRDL` instruction is used to read from bits <15:0> of program memory space. The `TBLWTL` instruction is used to write to bits <15:0> of program memory space. `TBLRDL` and `TBLWTL` can access program memory in Word or Byte mode.

The `TBLRDH` and `TBLWTH` instructions are used to read or write to bits <23:16> of program memory space. `TBLRDH` and `TBLWTH` can access program memory in Word or Byte mode. Because the program memory is only 24 bits wide, the `TBLRDH` and `TBLWTH` instructions can address an upper byte of program memory that does not exist. This byte is called the "phantom byte". Any read of the phantom byte will return 0x00. A write to the phantom byte has no effect.
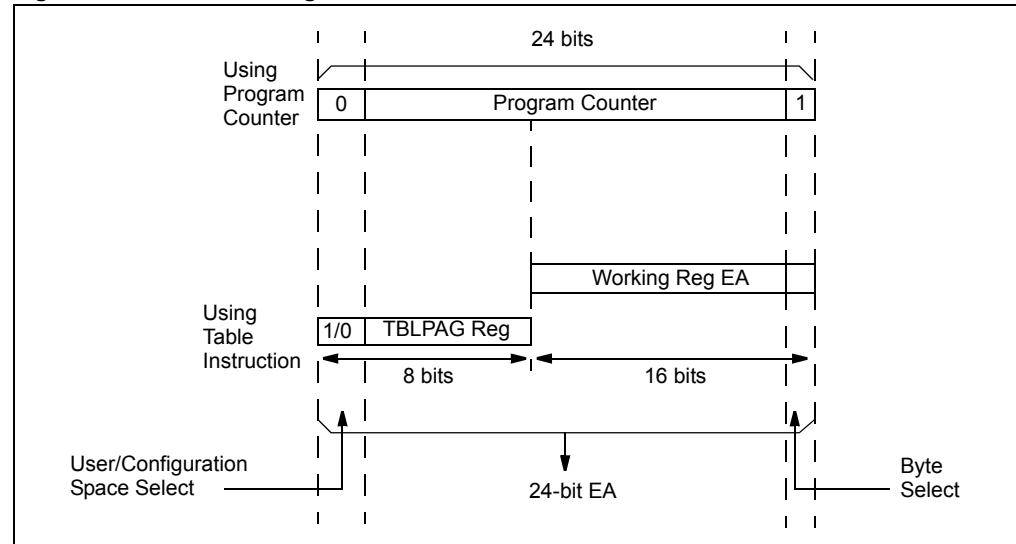
The 24-bit program memory can be regarded as two side-by-side 16-bit spaces, with each space sharing the same address range. Therefore, the `TBLRDL` and `TBLWTL` instructions access the "low" program memory space (PM<15:0>). The `TBLRDH` and `TBLWTH` instructions access the "high" program memory space (PM<31:16>). Any reads or writes to PM<31:24> will access the phantom (unimplemented) byte. When any of the table instructions are used in Byte mode, the Least Significant bit (LSb) of the table address will be used as the byte select bit. The LSb determines which byte in the high or low program memory space is accessed.

Figure 5-1 shows how the program memory is addressed using the table instructions. A 24-bit program memory address is formed using bits <7:0> of the TBLPAG register and the effective address (EA) from a W register specified in the table instruction. The 24-bit program counter is shown in Figure 5-1 for reference. The upper 23 bits of the EA are used to select the program memory location.

For the Byte mode table instructions, the LSb of the W register EA is used to select which byte of the 16-bit program memory word is addressed. A '1' selects bits <15:8>. A '0' selects bits <7:0>. The LSb of the W register EA is ignored for a table instruction in Word mode.

In addition to the program memory address, the table instruction also specifies a W register (or a W register pointer to a memory location) that is the source of the program memory data to be written or the destination for a program memory read. For a table write operation in Byte mode, bits <15:8> of the source working register are ignored.

Figure 5-1:   Addressing for Table Instructions

5.2.1   Using Table Read Instructions

Table reads require two steps:

- An address pointer is set up using the TBLPAG register and one of the W registers
- The program memory contents at the address location may be read

5.2.1.1   READ WORD MODE

The code sequence shown in Example 5-1 shows how to read a word of program memory using the table instructions in Word mode.

Example 5-1:   Read Word Mode

```
; Set up the address pointer to program space
    MOV         #tblpage(PROG_ADDR),W0      ; get table page value
    MOV         W0,TBLPAG                   ; load TBLPAG register
    MOV         #tbloffset(PROG_ADDR),W0    ; load address LS word
; Read the program memory location
    TBLRDH      [W0],W3                     ; Read high byte to W3
    TBLRDL      [W0],W4                     ; Read low word to W4
```

5

Flash Programming

5.2.1.2   READ BYTE MODE

In the code sequence shown in Example 5-2, the post-increment operator on the read of the low byte causes the address in the working register to increment by one. This sets EA<0> to a '1' for access to the middle byte in the third write instruction. The last post-increment sets W0 back to an even address, pointing to the next program memory location.

> **Note:** The `tblpage()` and `tbloffset()` directives are provided by the Microchip assembler for the dsPIC33F/PIC24H. These directives select the appropriate TBLPAG and W register values for the table instruction from a program memory address value. For more information, refer to the *"MPLAB® Assembler Linker and Utilities for PIC24 MCUs and dsPIC® DSCs User's Guide"* (DS51317).

Example 5-2:   Read Byte Mode

```
; Set up the address pointer to program space
    MOV          #tblpage(PROG_ADDR),W0      ; get table page value
    MOV          W0,TBLPAG                   ; load TBLPAG register
    MOV          #tbloffset(PROG_ADDR),W0    ; load address LS word
; Read the program memory location
    TBLRDH.B     [W0],W3                     ; Read high byte to W3
    TBLRDL.B     [W0++],W4                   ; Read low byte to W4
    TBLRDL.B     [W0++],W5                   ; Read middle byte to W5
```

5.2.1.3   TABLE WRITE HOLDING BUFFERS

> **Note:** Some dsPIC33F/PIC24H devices do not have multiple holding buffers. Refer to the **"Flash Program Memory"** chapter in the specific device data sheet for availability.

Table write instructions do not write directly to the non-volatile program memory. Instead, the table write instructions load holding buffers that store the write data. The multiple holding buffers are not memory mapped and can be accessed only using table write instructions. When all the holding buffers are loaded, the memory programming operation is started by executing a special sequence of instructions.

The dsPIC33F/PIC24H Flash program memory is segmented into pages (512 instruction words) and rows (64 instruction words).

The dsPIC33F/PIC24H family of devices support 64 holding registers to program one row of memory at a time. The memory logic decides which set of write buffers to load based on the address value in the table write instruction.

5.2.1.4   WRITE WORD MODE

The sequence shown in Example 5-3 can be used to write a single program memory latch location in Word mode.

Example 5-3:   Writing a Single Program Memory Latch Location in Word Mode

```
; Setup the address pointer to program space
    MOV          #tblpage(PROG_ADDR),W0      ; get table page value
    MOV          W0,TBLPAG                   ; load TBLPAG register
    MOV          #tbloffset(PROG_ADDR),W0    ; load address LS word
; Load write data into W registers
    MOV          #PROG_LOW_WORD,W2
    MOV          #PROG_HI_BYTE,W3
; Perform the table writes to load the latch
    TBLWTL       W2,[W0]
    TBLWTH       W3,[W0++]
```

In Example 5-3, the content of the upper byte of W3 does not matter because this data will be written to the phantom byte location. W0 is post-incremented by two after the second `TBLWTH` instruction in preparation for the write to the next program memory location. For devices without multiple holding buffers, a complete Flash program cycle must be done before writing to the next programming location.

5.2.1.5 WRITE BYTE MODE

The code sequence shown in Example 5-4 can be used to write a single program memory latch location in Byte mode.

Example 5-4: Writing a Single Program Memory Latch Location in Byte Mode

```
; Setup the address pointer to program space
    MOV         #tblpage(PROG_ADDR),W0   ; get table page value
    MOV         W0,TBLPAG                ; load TBLPAG register
    MOV         #tbloffset(PROG_ADDR),W0 ; load address LS word
; Load data into working registers
    MOV         #LOW_BYTE,W2
    MOV         #MID_BYTE,W3
    MOV         #HIGH_BYTE,W4
; Write data to the latch
    TBLWTH.B    W4,[W0]                  ; write high byte
    TBLWTL.B    W2,[W0++]                ; write low byte
    TBLWTL.B    W3,[W0++]                ; write middle byte
```

In Example 5-4, the post-increment operator on the write to the low byte causes the address in W0 to increment by one. This sets EA<0> = 1 for access to the middle byte in the third write instruction. The last post-increment sets W0 back to an even address pointing to the next program memory location. For devices without multiple holding buffers, a complete Flash program cycle must be done before writing to the next programming location.

5

Flash Programming

5.3   CONTROL REGISTERS

Two Special Function Registers (SFRs) are used to program Flash memory erase and write operations: NVMCON and NVMKEY.

The Flash Memory Control register (NVMCON) controls which memory segments are to be erased, erase cycle, fuse/Flash operation, word or row programming, write status, bulk erase and start of programming cycle. For devices without multiple holding buffers, only word programming is allowed.

The Nonvolatile Memory Key register (NVMKEY) is a write-only register that is used for protection from unintended Flash operations. To start a programming or erase sequence, the user must consecutively write 0x55 and 0xAA to the NVMKEY register.

5.3.1   NVMCON Register

The NVMCON register is the primary control register for Flash and program/erase operations. This register selects whether an erase or program operation will be performed and can 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 is performed.

5.3.2   NVMKEY Register

The NVMKEY register (Register 5-2) is a write-only register used to prevent accidental writes or erasures of Flash memory. To start a programming or erase sequence, the following steps must be considered sequentially:

1.   Write 0x55 to NVMKEY.
2.   Write 0xAA to NVMKEY.
3.   Execute two `NOP` instructions.

After this sequence, a write will be allowed to the NVMCON register for one instruction cycle. In most cases, the user-assigned application needs to set the WR bit in the NVMCON register to start the program or erase instruction cycle. Interrupts should be disabled during the key sequence. Example 5-5 shows how the key sequence is performed.

Example 5-5:   NVMKEY Key Sequence

```
    PUSH    SR              ; Disable interrupts, if enabled
    MOV     #0x00E0,W0
    IOR     SR

    MOV     #0x55,W0
    MOV     W0, NVMKEY
    MOV     #0xAA,W0
    MOV     W0, NVMKEY      ; NOP not required
    BSET    NVMCON,#WR      ; Start the program/erase cycle
    NOP
    NOP
    POP     SR              ; Re-enable interrupts
```

For more information, refer to **5.4.2 "Flash Programming Operations"**.

Register 5-1: **NVMCON: Flash Memory Control Register**

R/SO-0[(1)]	R/W-0[(1)]	R/W-0[(1)]	U-0	U-0	U-0	U-0	U-0
WR	WREN	WRERR	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0[(1)]	U-0	U-0	R/W-0[(1)]	R/W-0[(1)]	R/W-0[(1)]	R/W-0[(1)]
—	ERASE	—	—	NVMOP<3:0>[(2,3)]			
bit 7							bit 0

Legend:			
	SO = Settable only bit		
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 15 **WR:** Write Control bit

1 = Initiates a Flash memory program or erase operation. The operation is self-timed and the bit is cleared by hardware after the operation is complete
0 = Program or erase operation is complete and inactive

bit 14 **WREN:** Write Enable bit

1 = Enable Flash program/erase operations
0 = Inhibit Flash program/erase operations

bit 13 **WRERR:** Write Sequence Error Flag bit

1 = An improper program or erase sequence attempt or termination has occurred (bit is set on any set attempt of the WR bit)
0 = The program or erase operation completed normally

bit 12-7 **Unimplemented:** Read as '0'

bit 6 **ERASE:** Erase/Program Enable bit

1 = Perform the erase operation specified by NVMOP<3:0> on the next WR command
0 = Perform the program operation specified by NVMOP<3:0> on the next WR command

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **NVMOP<3:0>:** NVM Operation Select bits[(2,3)]

If ERASE = 1:

1111 = Memory bulk erase operation
1110 = Reserved
1101 = Erase General Segment and FGS Configuration register
1100 = Erase Secure Segment and FSS Configuration register
1011 = Reserved
0011 = No operation
0010 = Memory page erase operation
0001 = No operation
0000 = Erase a single Configuration register byte

If ERASE = 0:

1111 = No operation
1110 = Reserved
1101 = No operation
1100 = No operation
1011 = Reserved
0011 = Memory word program operation
0010 = No operation
0001 = Memory row program operation
0000 = Program a single Configuration register byte

Note 1: These bits can be reset only on a Power-on Reset (POR).
  2: All other combinations of NVMOP<3:0> are unimplemented.
  3: The definition of these bits is device-specific. Refer to the **"Flash Program Memory"** chapter in the specific device data sheet for the definition of the NVMOP<3:0> bits.

5

Flash Programming

Register 5-2: **NVMKEY: Nonvolatile Memory Key Register**

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
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:	SO = Settable only bit		
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 15-8   **Unimplemented:** Read as '0'

bit 7-0   **NVMKEY<7:0>:** Key Register (write-only) bits

5.4 RUN-TIME SELF-PROGRAMMING (RTSP)

RTSP allows the user-assigned application to modify Flash program memory contents. RTSP is accomplished using `TBLRD` (table read) and `TBLWT` (table write) instructions, and the NVM Control registers. With RTSP, the user-assigned application can erase program memory eight rows (64 * 8 = 512 instructions) at a time and can write program memory data a single row (64 instructions) at a time (on devices with multiple holding buffers, or a single instruction simultaneously for devices without multiple holding buffers).

5.4.1 RTSP Operation

The dsPIC33F/PIC24H Flash program memory array is organized into rows of 64 instructions or 192 bytes. RTSP allows the user-assigned application to erase blocks of eight rows (512 instructions) simultaneously and to program 64 instructions at a time (on devices with multiple holding buffers, or a single instruction at a time for devices without multiple holding buffers). The 8-row erase blocks and single-row write blocks are edge-aligned, from the beginning of program memory, on boundaries of 1536 bytes and 192 bytes.

The program memory implements holding buffers on some devices that can contain 64 instructions of programming data. Prior to the programming operation, the write data must be loaded into the buffers. The instruction words loaded must be from a group of 64 boundaries. To successfully program the Flash memory, the address written by the last `TBLWTL` or `TBLWTH` instruction must belong to the row that is intended to be programmed. For devices without multiple holding buffers, single word programming is the only option available.

The basic sequence for RTSP is to set up a Table Pointer, and then perform a series of `TBLWTL` and `TBLWTH` instructions to load the buffers. Programming is performed by setting the control bits in the NVMCON register. A total of 64 `TBLWTL` and `TBLWTH` instructions are required to load the instructions when multiple holding buffers are present. A different sequence is followed on devices without multiple holding buffers. Only one set of `TBLWTL` and `TBLWTH` instructions are required to load a single buffer, and then programming is performed by setting the control bits in the NVMCON register.

All the table write operations are single-word writes (two instruction cycles), because only the buffers are written. A programming cycle is required for programming each word or row.

5.4.2 Flash Programming Operations

A program or erase operation is necessary for programming or erasing the internal Flash program memory in RTSP mode. The program or erase operation is automatically timed by the device (for timing information, refer to the **"Flash Program Memory"** chapter in the specific device data sheet). Setting the WR bit (NVMCON<15>) starts the operation. The WR bit is automatically cleared when the operation is finished.

The CPU stalls (waits) until the programming operation is finished. The CPU will not execute any instructions or respond to interrupts during this time. If any interrupts occur during the programming cycle, it will remain pending until the cycle completes. If a Power-On Reset (POR) or Brown-Out Reset (BOR) occurs, the programming or erase operation will be aborted and user should execute the programming or erase operation again after the reset. If any other reset occurs while executing a programming or erase operation (WDTO, IOPUWR, MCLR, SWR, CM, or TRAPR) the result will remain pending until the RTSP cycle completes.

5

Flash Programming

5.4.2.1   FLASH ROW PROGRAMMING ALGORITHM

Flash row programming is only possible on devices with multiple holding buffers.

The user-assigned application can program one row of Flash program memory (64 instruction words). To perform this, it is necessary to erase a page (512 instruction words) containing the desired row.

The general process is as follows:

1. Read one page (512 instruction words) of Flash program memory and store it into data RAM as a data "image". The RAM image must be read from an even 512-word program memory address boundary.
2. Update the RAM data image with the new program memory data.
3. Erase the Flash program memory page.
    a) Set up the NVMCON register to erase one page of Flash program memory.
    b) Select the page for an erase operation by performing a dummy table write to any location in the page.
    c) Disable interrupts.
    d) Write the key sequence to the NVMKEY register to enable the erase.
    e) Set the WR bit. This will start the erase cycle.
    f) The CPU will stall for the duration of the erase cycle.
    g) The WR bit is cleared when the erase cycle ends.
    h) Re-enable interrupts.
4. Load the row (64 instruction words) of instruction words from RAM into the write buffers using a table write operation.
5. Program the row (64 instruction words) in Flash program memory.
    a) Set up the NVMCON register to program one row of Flash program memory.
    b) Disable interrupts.
    c) Write the key sequence to the NVMKEY register to enable the program cycle.
    d) Set the WR bit. This will start the program cycle.
    e) The CPU will stall for the duration of the program cycle.
    f) The WR bit is cleared by the hardware when the program cycle ends.
    g) Re-enable interrupts.
6. Repeat steps 4 through 6 to program all eight rows in the program memory page.
7. Repeat steps 1 through 7, as needed, to program the desired amount of Flash program memory.

Note 1:	The user should remember that the minimum amount of program memory that can be erased using RTSP is 512 instruction word locations. Therefore, it is important that an image of these locations be stored in general purpose RAM before an erase cycle is initiated.
2:	A row or word in Flash program memory should not be programmed more than twice before being erased.

5.4.2.2   FLASH SINGLE WORD PROGRAMMING ALGORITHM

This algorithm is available for devices with multiple or single holding buffers.

The user-assigned application can program one word of Flash program memory (that is, 1 instruction word).

The general process is as follows:

1. Load the word (1 instruction word) from RAM into the write latch using a table write operation.
2. Program the instruction word in Flash program memory.
    a) Set up the NVMCON register to program one word of Flash program memory.
    b) Disable interrupts.
    c) Write the key sequence to the NVMKEY register to enable the program cycle.
    d) Set the WR bit. This will start the program cycle.
    e) The CPU will stall for the duration of the program cycle.
    f) The WR bit is cleared by the hardware when the program cycle ends.
    g) Re-enable interrupts.
3. Repeat steps 1 and 2 as needed to program the desired amount of Flash program memory.

5.4.2.3   ERASING ONE PAGE OF FLASH

The code sequence shown in Example 5-6 can be used to erase a page (512 instructions) of program memory. The NVMCON register is configured to erase one page of program memory. A dummy table write to any location in the page selects the address of the row to be erased. The program memory must be erased at an "even" 512 instruction word address boundary. Therefore, the 10 Least Significant bits of the table write program memory address have no effect when a page is erased.

The erase operation is initiated by writing a key sequence to the NVMKEY register before setting the WR control bit (NVMCON<15>). The key sequence needs to be executed in the exact order as shown, without interruption. Therefore, interrupts should be disabled prior to writing the sequence.

Two NOP instructions should be inserted in the code at the point where the CPU will resume operation. Finally, interrupts can be enabled (if required).

Example 5-6:   Erasing a Page of Flash Program Memory

```
; Perform dummy table write to the Page to be erased.
        MOV     #tblpage(PROG_ADDR),W0
        MOV     W0,TBLPAG
        MOV     #tbloffset(PROG_ADDR),W0
        TBLWTL  w0,[w0]
; Setup NVMCON to erase one page of Program Memory
        MOV     #0x4042,W0
        MOV     W0,NVMCON
; Disable interrupts while the KEY sequence is written
        PUSH    SR
        MOV     #0x00E0,W0
        IOR     SR
; Write the KEY Sequence
        MOV     #0x55,W0
        MOV     W0,NVMKEY
        MOV     #0xAA,W0
        MOV     W0,NVMKEY
; Start the erase operation
        BSET    NVMCON,#WR
; Insert two NOPs after the erase cycle (required)
        NOP
        NOP
;Re-enable interrupts, if needed
        POP     SR
```

5

Flash Programming

5.4.2.4 LOADING WRITE BUFFERS

This action only applies to devices with multiple holding buffers.

The write buffers are used as a storage mechanism between the user application table writes and the actual row programming sequence. Example 5-7 provides the sequence of instructions that can be used to load 64 write buffers (64 instruction words). Sixty-four TBLWTL and 64 TBLWTH instructions are needed to load the write buffers for programming a row of Flash memory.

The row of 64 instruction words does not necessarily have to be written in sequential order. The 7 Least Significant bits of the table write address determine which of the buffers will be written. However, all 64 instruction words should be written for each programming cycle to overwrite old data.

Table write program memory addresses are captured to select the address of the row to be programmed. The program memory must be programmed at an "even" 64 instruction word address boundary. In effect, the 7 Least Significant bits of the table write operation select the write buffers to program the instruction word within the row. They have no effect when a row is programmed.

> **Note:** The code shown in Example 5-7 is the Load_Write_Latch code that is referred in subsequent examples.

Example 5-7: Loading Write Buffers

```
; Set up a pointer to the first program memory location to be written.
        MOV     #tblpage(PROG_ADDR),W0
        MOV     W0,TBLPAG
        MOV     #tbloffset(PROG_ADDR),W0

; Perform the TBLWT instructions to write the latches
; W0 is incremented in the TBLWTH instruction to point to the
; next instruction location.
        MOV     #64,W3
        MOV     #ram_image,W1
loop:
        TBLWTL  [W1++],[W0]
        TBLWTH  [W1++],[W0++]
        DEC     W3,W3
        BRA     NZ,loop
```

5.4.2.5 SINGLE ROW PROGRAMMING EXAMPLE

This example only applies to devices with multiple holding buffers.

The NVMCON register is configured to program one row of Flash memory. The program operation is initiated by writing a key sequence to the NVMKEY register before setting the WR control bit (NVMCON<15>). The key sequence needs to be executed in the exact order as shown in Example 5-8, without interruption. Therefore, interrupts should be disabled prior to writing the sequence.

Two NOP instructions should be inserted in the code at the point where the CPU will resume operation. Finally, interrupts can be enabled (if required).

Example 5-8: **Single Row Programming**

```
; Setup NVMCON to write 1 row of program memory
        MOV    #0x4001,W0
        MOV    W0,NVMCON
; Load the 64 program memory write latches
        CALL   Load_Write_Latch(1)
; Disable interrupts, if enabled
        PUSH   SR
        MOV    #0x00E0,W0
        IOR    SR
; Write the KEY sequence
        MOV    #0x55,W0
        MOV    W0,NVMKEY
        MOV    #0xAA,W0
        MOV    W0,NVMKEY
; Start the programming sequence
        BSET   NVMCON,#WR
; Insert two NOPs after programming
        NOP
        NOP
; Re-enable interrupts, if required
        POP    SR
```

> **Note:** For more information, see **5.4.2.4 "Loading Write Buffers"**.

5.4.2.6   PROGRAMMING ONE WORD OF FLASH MEMORY

Assuming the user-assigned application has erased the Flash location to be programmed, use table write instructions to write an instruction word (24-bit) into the write latch. The 7 Least Significant bits of the table write operation select the write latch to program the instruction word within the row.

The TBLPAG register is loaded with the 8 Most Significant bits of the Flash address. The 16 Least Significant bits of the Flash address are automatically captured internally when the table write is executed to program the word during program operation.

The NVMCON register is configured to program one word of Flash memory. The program operation is initiated by writing a key sequence to the NVMKEY register before setting the WR control bit (NVMCON<15>). The key sequence needs to be executed in the exact order as shown in Example 5-9, without interruption. Therefore, interrupts should be disabled prior to writing the sequence.

Two NOP instructions should be inserted in the code at the point where the CPU will resume operation. Finally, interrupts can be enabled (if required).

Example 5-9: **Programming One Word of Flash Memory**

```
; Setup a pointer to data Program Memory
        MOV    #tblpage(PROG_ADDR),W0
        MOV    W0,TBLPAG
        MOV    #tbloffset(PROG_ADDR),W0
; Perform the TBLWT instructions to write the latches
        MOV    #LOW_WORD_N,W2
        MOV    #HIGH_BYTE_N,W3
        TBLWTL W2,[W0]
        TBLWTH W3,[W0++]
; Setup NVMCON for programming one word to data Program Memory
        MOV    #0x4003,W0
        MOV    W0,NVMCON
; Disable interrupts while the KEY sequence is written
        PUSH   SR
        MOV    #0x00E0,W0
        IOR    SR
; Write the key sequence
        MOV    #0x55,W0
        MOV    W0,NVMKEY
        MOV    #0xAA,W0
        MOV    W0,NVMKEY
; Start the write cycle
        BSET   NVMCON,#WR
;Re-enable interrupts, if needed
        POP    SR
```

5

Flash Programming

5.4.3 Writing to Device Configuration Registers

Run-Time Self-Programming (RTSP) can be used to write to the device Configuration registers. RTSP allows each Configuration register to be individually rewritten without performing an erase cycle. Caution must be exercised when writing the Configuration registers because they control critical device operating parameters, such as the system clock source, PLL multiplication ratio and WDT enable.

The procedure for programming a device Configuration register is similar to the procedure for programming Flash program memory, except only `TBLWTL` instructions are required. This is because the upper eight bits in each Device Configuration register are unused. Furthermore, bit 23 of the table write address must be set to access the Configuration registers. For more information on device Configuration registers, refer to **Section 25. "Device Configuration"** (DS70194) in the *"dsPIC33F/PIC24H Family Reference Manual"* and the **"Special Features"** chapter of the specific device data sheet.

> **Note 1:** Writing to device Configuration registers is not available in all devices. Please refer to the **"Flash Program Memory"** chapter in the specific device data sheet to determine the modes that are available according to the NVMOP<3:0> bits definition.
>
> **2:** While performing RTSP on device Configuration registers, the device must be operating using the Internal FRC Oscillator (without PLL). If the device is operating from a different clock source, a clock switch to the Internal FRC Oscillator (NOSC<2:0> = `000`) must be performed prior to performing RTSP operation in device Configuration registers.
>
> **3:** If the Primary Oscillator Mode Select bits (POSCMD<1:0>) in the Oscillator Configuration register (FOSC) are being reprogrammed to a new value, ensure that the Clock Switching Mode bits (FCKSM<1:0>) in the FOSC register have an initial programmed value of '`00`', prior to performing this RTSP operation.

5.4.3.1 CONFIGURATION REGISTER WRITE ALGORITHM

1. Write the new configuration value to the table write latch using a `TBLWTL` instruction.
2. Configure NVMCON for a Configuration register write (NVMCON = 0x4000).
3. Disable interrupts, if enabled.
4. Write the key sequence to NVMKEY.
5. Start the write sequence by setting WR (NVMCON<15>).
6. CPU execution will resume when the write is finished.
7. Re-enable interrupts, if needed.

Example 5-10 shows the code sequence to modify a device Configuration register.

Example 5-10: Configuration Register Write Code Example

```
; Set up a pointer to the location to be written.
        MOV     #tblpage(CONFIG_ADDR),W0
        MOV     W0,TBLPAG
        MOV     #tbloffset(CONFIG_ADDR),W0
; Get the new data to write to the configuration register
        MOV     #ConfigValue,W1
; Perform the table write to load the write latch
        TBLWTL  W1,[W0]
; Configure NVMCON for a configuration register write
        MOV     #0x4000,W0
        MOV     W0,NVMCON
; Disable interrupts, if enabled
        PUSH    SR
        MOV     #0x00E0,W0
        IOR     SR
; Write the KEY sequence
        MOV     #0x55,W0
        MOV     W0,NVMKEY
        MOV     #0xAA,W0
        MOV     W0,NVMKEY
; Start the programming sequence
        BSET    NVMCON,#WR
; Insert two NOPs after programming
        NOP
        NOP
; Re-enable interrupts, if required
        POP     SR
```

5.5   REGISTER MAP

A summary of the registers associated with Flash Programming is provided in Table 5-1.

Table 5-1:   Flash Programming Registers

File Name	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	All Resets
NVMCON	WR	WREN	WRERR	—	—	—	—	—	—	ERASE	—	—	NVMOP<3:0>				0000
NVMKEY	—	—	—	—	—	—	—	—	NVMKEY<7:0>								0000

Legend: x = unknown value on Reset, — = unimplemented, read as '0'. Reset values are shown in hexadecimal.

5.6 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 dsPIC33F/PIC24H product families, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Flash Programming module include the following:

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

Note: Please visit the Microchip web site (www.microchip.com) for additional Application Notes and code examples for the dsPIC33F/PIC24H family of devices.

5.7   REVISION HISTORY

Revision A (February 2007)

This is the initial released version of the document.

Revision B (February 2007)

Minor edits throughout the document.

Revision C (December 2009)

This version of the document was created by combining the dsPIC33F and PIC24H Flash Programming family reference manual sections into a single document with the following updates:

- Added a shaded note at the beginning of the section, which provides information on complementary documentation
- Added Note 1 and Note 2 to **5.4.3 "Writing to Device Configuration Registers"**
- Additional minor corrections such as language and formatting updates were incorporated throughout the document

Revision D (April 2010)

This version of the document contains the following updates:

- Added a shaded note to **5.2.1.3 "Table Write Holding Buffers"**
- Added a sentence related to devices without multiple holding buffers to the last paragraph of **5.2.1.4 "Write Word Mode"** and **5.2.1.5 "Write Byte Mode"** and to the second paragraph of **5.3 "Control Registers"**
- Added Note 3 to the Flash Memory Control register (Register 5-1)
- Updated the first paragraph of **5.4 "Run-Time Self-Programming (RTSP)"**
- Updated the second and third paragraphs of **5.4.1 "RTSP Operation"**
- Updated the first paragraph of **5.4.2.1 "Flash Row Programming Algorithm"**
- Added the new section **5.4.2.2 "Flash Single Word Programming Algorithm"**
- Added a shaded note to **5.4.2.4 "Loading Write Buffers"**
- Add a sentence to the first paragraph of **5.4.2.5 "Single Row Programming Example"**
- Added a new Note 1 to the shaded note in **5.4.3 "Writing to Device Configuration Registers"**
- Added the new section **5.5 "Register Map"**

Revision E (April 2012)

This version of the document contains the following updates:

- Updated Example 5-8
- Removed 5.6 "Design Tips"
- Additional minor corrections such as language and formatting updates were incorporated throughout the document

5

Flash Programming

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.

Trademarks

The Microchip name and logo, the Microchip logo, dsPIC, KEELOQ, KEELOQ logo, MPLAB, PIC, PICmicro, PICSTART, PIC^{32} logo, rfPIC 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, Omniscient Code Generation, PICC, PICC-18, PICDEM, PICDEM.net, PICkit, PICtail, REAL ICE, rfLAB, 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.

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.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

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

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

11/29/11