Flash Programming

HIGHLIGHTS

This section of the manual contains the following major topics:

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 dsPIC33/PIC24 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 Website at: http://www.microchip.com

1.0 INTRODUCTION

This section describes the technique for programming Flash program memory. The dsPIC33/PIC24 families of devices have an internal programmable Flash program memory for execution of user code. There are up to three methods to program this memory:

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

RTSP is performed by the application software during execution, while ICSP and EICSP are performed from an external programmer using a serial data connection to the device. ICSP and EICSP allow much faster programming time than RTSP. RTSP techniques are described in **Section 4.0 "Run-Time Self-Programming (RTSP)"**. The ICSP and EICSP protocols are defined in the Programming Specification documents for the respective devices, which can be downloaded from the Microchip website (http://www.microchip.com).

When programming in the C language, several built-in functions are available that facilitate Flash programming. See the *"MPLAB® XC16 C Compiler User's Guide"* (DS50002071) for details regarding built-in functions.

2.0 TABLE INSTRUCTION OPERATION

The table instructions provide the method of transferring data between the Flash program memory space and the data memory space of dsPIC33/PIC24 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 Flash program memory space. `TBLRDL` and `TBLWTL` can access Flash program memory in Word mode 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 Flash program memory in Word or Byte mode. Because the Flash program memory is only 24 bits wide, the `TBLRDH` and `TBLWTH` instructions can address an upper byte of Flash 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.

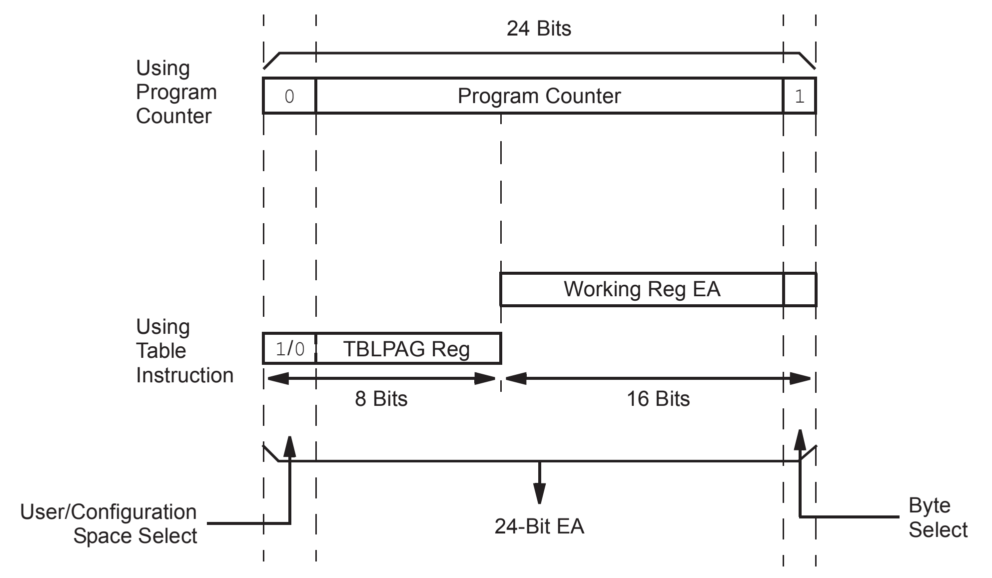
The 24-bit Flash program memory can be regarded as two side-by-side 16-bit spaces, with each space sharing the same address range. Therefore, the `TBLRDL` and `TBLWTL` instructions access the "low" program memory space (PM<15:0>). The `TBLRDH` and `TBLWTH` instructions access the "high" program memory 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 2-1 illustrates how the Flash 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 (PC) is illustrated in Figure 2-1 for reference. The upper 23 bits of the EA are used to select the Flash 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 Flash program memory word is addressed. '1' selects bits<15:8> and '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 Flash 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 Flash program memory data to be written, or the destination for a Flash program memory read. For a table write operation in Byte mode, bits<15:8> of the source Working register are ignored.

Figure 2-1: Addressing for Table Instructions



2.1 Using Table Read Instructions

Table reads require two steps:

1. The Address Pointer is set up using the TBLPAG register and one of the W registers.
2. The Flash program memory contents at the address location may be read.

2.1.1 READ WORD MODE

The code shown in Example 2-1, shows how to read a word of Flash program memory using the table instructions in Word mode.

Example 2-1: 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
```


Example 2-2: Read Word Mode (in C)

```
int addrOffset;
int varWord1;
int varWord2;

TBLPAG = ((PROG_ADDR & 0x7F0000)>>16);
addrOffset = (PROG_ADDR & 0x00FFFE);

varWord1 = __builtin_tblrdl(addrOffset);
varWord2 = __builtin_tblrdh(addrOffset);
```

2.1.2 READ BYTE MODE

The code shown in Example 2-3, shows the post-increment operator on the read of the low byte, which 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 Flash program memory location.

Example 2-3: 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
```

2.1.3 TABLE WRITE LATCHES

Table write instructions do not write directly to the nonvolatile program memory. Instead, the table write instructions load write latches that store the write data. The NVM Address registers must be loaded with the first address where latched data should be written. When all of the write latches have been loaded, the actual memory programming operation is started by executing a special sequence of instructions. During programming, the hardware transfers the data in the write latches to Flash memory.

The write latches always start at address 0xFA0000, and extend through 0xFA0002 for word programming, or through 0xFA00FE for devices which have row programming.

> **Note:** The number of write latches varies by device. Refer to the **"Flash Program Memory"** chapter of the specific device data sheet for the number of available write latches.

3.0 CONTROL REGISTERS

Several Special Function Registers (SFRs) are used to program the Flash program memory erase and write operations: NVMCON, NVMKEY, and the NVM Address registers, NVMADR and NVMADRU.

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 3-1. The lower byte of NVMCON configures the type of NVM operation that will be performed.

3.2 NVMKEY Register

The NVMKEY register (see Register 3-4) is a write-only register used to prevent accidental writes or erasures of the Flash memory. To start a programming or erase sequence, the following steps must be considered:

1. Write 0x55 to NVMKEY.
2. Write 0xAA to NVMKEY.
3. Start the programming write cycle by setting the WR bit (NVMCON<15>).
4. Execute two `NOP` instructions.

After this sequence, a write will be allowed to the NVMCON register for one instruction cycle. The user application needs to set the WR bit (NVMCON<15>) to start the program or erase cycle. Interrupts should be disabled during the unlock sequence. Example 3-1 shows how the unlock sequence is performed.

Example 3-1: NVMKEY Unlock Sequence

```
; if programming, load write latches
...
; Set NVM Address Registers
...
; Disable interrupts < priority 7 for next 5 instructions.
; Assumes no level 7 peripheral interrupts
    DISI    #06
    MOV     #0x55, W0
    MOV     W0, NVMKEY
    MOV     #0xAA, W0
    MOV     W0, NVMKEY
    BSET    NVMCON, #15        ; Start the program/erase cycle
    NOP
    NOP
```

Refer to **Section 4.2 "Flash Programming Operations"** for more programming examples.

3.3 NVM Address Registers

The two NVM Address registers, NVMADRU and NVMADR, when concatenated, form the 24-bit EA of the selected row or word for programming operations. The NVMADRU register is used to hold the upper eight bits of the EA, and the NVMADR register is used to hold the lower 16 bits of the EA. Some devices may refer to these same registers as NVMADRL and NVMADRH.

The NVM Address registers should always point to a double instruction word boundary when performing a double instruction word programming operation, a row boundary when performing a row programming operation or a page boundary when performing a page erase operation.

Register 3-1: NVMCON: Flash Memory Control Register

R/SO-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
WR(1)	WREN(1)	WRERR(1)	NVMSIDL(2)	—	—	RPDF(6)	URERR(6)
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>(3,5)			
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)
1 = Initiates a Flash program memory or erase operation; the operation is self-timed and the bit is cleared by hardware once operation is complete
0 = Program or erase operation is complete and inactive

bit 14 **WREN:** Write Enable bit(1)
1 = Enables Flash program memory/erase operations
0 = Inhibits Flash program memory/erase operations

bit 13 **WRERR:** Write Sequence Error Flag bit(1)
1 = An improper program or erase sequence attempt, or termination has occurred (bit is set automatically on any set attempt of the WR bit)
0 = The program or erase operation completed normally

bit 12 **NVMSIDL:** Stop in Idle Mode bit(2)
1 = Discontinues Flash operation when the device enters Idle mode
0 = Continues Flash operation when the device enters Idle mode

bit 11-10 **Unimplemented:** Read as ‘0’

bit 9 **RPDF:** Row Programming Data Format bit(6)
1 = Row data to be stored in RAM is in compressed format
0 = Row data to be stored in RAM is in uncompressed format

bit 8 **URERR:** Row Programming Data Underrun Error bit(6)
1 = Indicates row programming operation has been terminated
0 = No data underrun error is detected

bit 7-4 **Unimplemented:** Read as ‘0’

Note 1: This bit can only be reset (i.e., cleared) on a Power-on Reset (POR).
2: When exiting Idle mode, there is a power-up delay (T_{VREG}) before Flash program memory becomes operational. Refer to the **“Electrical Characteristics”** chapter of the specific device data sheet for more information.
3: All other combinations of NVMOP<3:0> are unimplemented.
4: This functionality is not available on all devices. Refer to the **“Flash Program Memory”** chapter in the specific device data sheet for available operations.
5: Entry into a power-saving mode after executing a `PWRSAV` instruction is contingent on completion of all pending NVM operations.
6: These bit are only available on devices that support RAM buffered row programming. Refer to the device-specific data sheet for availability.

Register 3-1: NVMCON: Flash Memory Control Register (Continued)

bit 3-0 **NVMOP<3:0>:** NVM Operation Select bits[3,5]

`1111` = Reserved
`1110` = Reserved
`1101` = Bulk erase primary Flash program memory[4]
`1100` = Reserved
`1011` = Reserved
`1010` = Bulk erase auxiliary Flash program memory[4]
`0011` = Memory page erase operation
`0010` = Memory row program operation[4]
`0001` = Memory double-word program operation
`0000` = Program a single Configuration register byte[4]

Note 1: This bit can only be reset (i.e., cleared) on a Power-on Reset (POR).

2: When exiting Idle mode, there is a power-up delay (T_{VREG}) before Flash program memory becomes operational. Refer to the **"Electrical Characteristics"** chapter of the specific device data sheet for more information.

3: All other combinations of NVMOP<3:0> are unimplemented.

4: This functionality is not available on all devices. Refer to the **"Flash Program Memory"** chapter in the specific device data sheet for available operations.

5: Entry into a power-saving mode after executing a `PWRSAV` instruction is contingent on completion of all pending NVM operations.

6: These bit are only available on devices that support RAM buffered row programming. Refer to the device-specific data sheet for availability.

Register 3-2: NVMADRU: Nonvolatile Memory Upper Address Register

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
NVMADRU<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 15-8 **Unimplemented:** Read as ‘0’

bit 7-0 **NVMADRU<7:0>:** Nonvolatile Memory Upper Write Address bits

Selects the upper eight bits of the location to program or erase in Flash program memory. This register may be read or written by the user application.

Register 3-3: NVMADR: Nonvolatile Memory Address Register

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
NVMADR<15:8>							
bit 15							bit 8

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
NVMADR<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 15-0 **NVMADR<15:0>:** Nonvolatile Memory Write Address bits

Selects the 16-bit offset of the location to program or erase in Flash program memory. This register may be read or written by the user application.

> **Note:** The NVM Address register should always point to a double instruction word boundary when performing a double instruction word programming operation, a row boundary when performing a row programming operation or a page boundary when performing a page erase operation.

Register 3-4: 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>:** NVM Key Register (write-only) bits

Note:	Refer to **Section 3.2 “NVMKEY Register”** for NVMKEY register operation.

4.0 RUN-TIME SELF-PROGRAMMING (RTSP)

RTSP allows the user application to modify Flash program memory contents. RTSP is accomplished using the `TBLRD` (table read) and `TBLWT` (table write) instructions, the TBLPAG register, and the NVM Control registers. With RTSP, the user application can erase a single page of Flash memory and program either two instruction words or up to 128 instruction words on certain devices.

4.1 RTSP Operation

The dsPIC33/PIC24 Flash program memory array is organized into erase pages that can contain up to 1024 instructions. The double-word programming option is available in all devices in the dsPIC33/PIC24 families. In addition, certain devices have row programming capability, which allows the programming of up to 128 instruction words at a time.

Programming and erase operations always occur on an even double programming word, row or page boundaries. Refer to the **"Flash Program Memory"** chapter of the specific device data sheet for the availability and sizes of a programming row, and the page size for erasing.

The Flash program memory implements holding buffers, called write latches, that can contain up to 128 instructions of programming data depending on the device. Prior to the actual programming operation, the write data must be loaded into the write latches.

The basic sequence for RTSP is to set up the Table Pointer, TBLPAG register, and then perform a series of `TBLWT` instructions to load the write latches. Programming is performed by setting the control bits in the NVMCON register. The number of `TBLWTL` and `TBLWTH` instructions needed to load the write latches is equal to the number of program words to be written.

Note:	It is recommended that the TBLPAG register be saved prior to modification and restored after use.

CAUTION
On some devices, the Configuration bits are stored in the last page of program Flash user memory space in a section called "Flash Configuration Bytes". With these devices, performing a page erase operation on the last page of program memory erases the Flash Configuration bytes, which enables code protection. Therefore, users should not perform page erase operations on the last page of program memory. This is not a concern when the Configuration bits are stored in Configuration memory space in a section called "Device Configuration Registers". Refer to the Program Memory Map in the **"Memory Organization"** chapter of the specific device data sheet to determine where Configuration bits are located.

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 (refer to the specific device data sheet for timing information). Setting the WR bit (NVMCON<15>) starts the operation. The WR bit is automatically cleared when the operation is finished.

The CPU stalls 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, they will remain pending until the cycle completes.

Some dsPIC33/PIC24 devices may provide auxiliary Flash program memory (refer to the **"Memory Organization"** chapter of specific device data sheet for details), which allows instruction execution without CPU Stalls while user Flash program memory is being erased and/or programmed. Conversely, auxiliary Flash program memory can be programmed without CPU Stalls, as long as code is executed from the user Flash program memory. The NVM interrupt can be used to indicate that the programming operation is complete.

> **Note 1:** If a POR or BOR event occurs while an RTSP erase or programming operation is in progress, the RTSP operation is aborted immediately. The user should execute the RTSP operation again after the device comes out of Reset.
>
> **2:** If an EXTR, SWR, WDTO, TRAPR, CM or IOPUWR Reset event occurs while an RTSP erase or programming operation is in progress, the device will be reset only after the RTSP operation is complete.

4.2.1 RTSP PROGRAMMING ALGORITHM

This section describes RTSP programming, which consists of three major processes.

4.3 Creating a RAM Image of the Data Page to be Modified

Perform these two steps to create a RAM image of the data page to be modified:

1. Read the page of Flash program memory and store it into data RAM as a data "image". The RAM image must be read starting from a page address boundary.
2. Modify the RAM data image as needed.

4.4 Erasing Flash Program Memory

After completing Steps 1 and 2 above, perform the following four steps to erase the Flash program memory page:

1. Set the NVMOP<3:0> bits (NVMCON<3:0>) to erase the page of Flash program memory read from Step 1.
2. Write the starting address of the page to be erased into the NVMADRU and NMVADR registers.
3. With interrupts disabled:
   a) Write the key sequence to the NVMKEY register to enable setting the WR bit (NVMCON<15>).
   b) Set the WR bit; this will start the erase cycle.
   c) Execute two `NOP` instructions.
4. The WR bit is cleared when the erase cycle is complete.

4.5 Programming the Flash Memory Page

The next part of the process is to program the Flash memory page. The Flash memory page is programmed using the data from the image created in Step 1. The data is transferred to the write latches in increments of either double instruction words or rows. All devices have double instruction word programming capability. (Refer to the **"Flash Program Memory"** chapter in the specific device data sheet to determine if, and what type of, row programming is available.) After the write latches are loaded, the programming operation is initiated, which transfers the data from the write latches into Flash memory. This is repeated until the entire page has been programmed.

Repeat the following three steps, starting at the first instruction word of the Flash page and incrementing in steps of either double program words, or instruction rows, until the entire page has been programmed:

1. Load the write latches:
   a) Set the TBLPAG register to point to the location of the write latches.
   b) Load the desired number of latches using pairs of `TBLWTL` and `TBLWTH` instructions:
      - For double-word programming, two pairs of `TBLWTL` and `TBLWTH` instructions are required
      - For row programming, a pair of `TBLWTL` and `TBLWTH` instructions are required for each instruction word row element
2. Initiate the programming operation:
   a) Set the NVMOP<3:0> bits (NVMCON<3:0>) to program either double instruction words or an instruction row, as appropriate.
   b) Write the first address of either the double instruction word or instruction row to be programmed into the NVMADRU and NVMADR registers.
   c) With interrupts disabled:
      - Write the key sequence to the NVMKEY register to enable setting the WR bit (NVMCON<15>)
      - Set the WR bit; this will start the erase cycle
      - Execute two `NOP` instructions
3. The WR bit is cleared when the programming cycle is complete.

Repeat the entire process as needed, to program the desired amount of Flash program memory.

> **Note 1:** The user should remember that the minimum amount of Flash program memory that can be erased using RTSP is a singe erased page. 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.
>
> **3:** On devices with Configuration bytes stored in the last page of Flash, performing a page erase operation on the last page of program memory clears the Configuration bytes, which enables code protection. On these devices, the last page of Flash memory should not be erased.

4.5.1 ERASING ONE PAGE OF FLASH

The code sequence shown in Example 4-1 can be used to erase a page of Flash program memory. The NVMCON register is configured to erase one page of program memory. The NVMADR and NMVADRU registers are loaded with the starting address of the page to be erased. The program memory must be erased at an "even" page address boundary. See the **"Flash Program Memory"** chapter of the specific device data sheet to determine the size of the Flash page.

The erase operation is initiated by writing a special unlock, or key sequence, to the NVMKEY register before setting the WR bit (NVMCON<15>). The unlock sequence needs to be executed in the exact order, as shown in Example 4-1, without interruption; therefore, interrupts should be disabled.

Two `NOP` instructions should be inserted in the code after the erase cycle.

On certain devices, the Configuration bits are stored in the last page of program Flash. With these devices, performing a page erase operation on the last page of program memory erases the Flash Configuration bytes, enabling code protection as a result. Users should not perform page erase operations on the last page of program memory.

Example 4-1: Erasing a Page of Flash Program Memory (in Assembly)

```
; Define the start address of the page to erase
        .equ    PROG_ADDR, 0x022000
; Set up the NVMADR registers to the starting address of the page
        MOV     #tblpage(PROG_ADDR),W0
        MOV     W0,NVMADRU
        MOV     #tbloffset(PROG_ADDR),W0
        MOV     W0,NVMADR
; Set up NVMCON to erase one page of Program Memory
        MOV     #0x4003,W0
        MOV     W0,NVMCON
; Disable interrupts < priority 7 for next 5 instructions
; Assumes no level 7 peripheral interrupts
        DISI    #06
; Write the KEY Sequence
        MOV     #0x55,W0
        MOV     W0,NVMKEY
        MOV     #0xAA,W0
        MOV     W0,NVMKEY
; Start the erase operation
        BSET    NVMCON,#15
; Insert two NOPs after the erase cycle (required)
        NOP
        NOP
```

Example 4-2: Erasing a Page of Flash Program Memory (in C)

```
int targetWriteAddressL;          // bits<15:0>
int targetWriteAddressH;     // bits<22:16>

// Set ERASE, WREN and configure NVMOP for page erase
NVMCON = 0x4003;
// Set target write address
NVMADR = targetWriteAddressL;
NVMADRU = targetWriteAddressH;
__builtin_disi(5);                // Disable interrupts for NVM unlock
__builtin_write_NVM();            // Start write cycle
while(NVMCONbits.WR == 1);
```

4.5.2 LOADING WRITE LATCHES

The write latches are used as a storage mechanism between the user application table writes and the actual programming sequence. During the programming operation, the device will transfer the data from the write latches into Flash memory.

For devices that support row programming, Example 4-3 shows the sequence of instructions that can be used to load 128 write latches (128 instruction words). 128 `TBLWTL` and 128 `TBLWTH` instructions are needed to load the write latches for programming a row of Flash program memory. Refer to the **"Flash Program Memory"** chapter of the specific device data sheet to determine the number of programming latches available on your device.

For devices that do not support row programming, Example 4-4 shows the sequence of instructions that can be used to load two write latches (two instruction words). Two `TBLWTL` and two `TBLWTH` instructions are needed to load the write latches.

> **Note 1:** The code for `Load_Write_Latch_Row` is shown in Example 4-3 and the code for `Load_Write_Latch_Word` is shown in Example 4-4. The code in both of these examples is referred to in subsequent examples.
>
> **2:** Refer to the specific device data sheet for the number of latches.

Example 4-3: Loading Write Latches for Row Programming

```
Load_Write_Latch_Row:
; This example loads 128 write latches
; W2 points to the address of the data to write to the latches
; Set up a pointer to the first latch location to be written
       MOV        #0xFA, W0
       MOV        W0, TBLPAG
       MOV        #0, W1

; Perform the TBLWT instructions to write the latches
; W2 is incremented in the TBLWTH instruction to point to the
; next instruction location
       MOV        #128, W3

    loop:
       TBLWTL.b   [W2++], [W1++]
       TBLWTL.b   [W2++], [W1--]
       TBLWTH.b   [W2++], [W1]
       INC2       W1, W1
       DEC        W3, W3
       BRA        NZ, loop
```

Example 4-4: Loading Write Latches for Double-Word Programming

```
Load_Write_Latch_Word:
; W2 points to the address of the data to write to the latches
; Set up a pointer to the first latch location to be written
       MOV     #0xFA,W0
       MOV     W0,TBLPAG
       MOV     #0,W1

; Perform the TBLWT instructions to write the latches
       TBLWTL  [W2++],[W1]
       TBLWTH  [W2++],[W1++]
       TBLWTL  [W2++],[W1]
       TBLWTH  [W2++],[W1++]
```

4.5.3 SINGLE ROW PROGRAMMING EXAMPLE

The NVMCON register is configured to program one row of Flash program memory. The program operation is initiated by writing a special unlock, or key sequence, to the NVMKEY register before setting the WR bit (NVMCON<15>). The unlock sequence needs to be executed without interruption, and in the exact order, as shown in Example 4-5. Therefore, interrupts should be disabled prior to writing the sequence.

Note:	Not all devices have row programming capability. Refer to the **"Flash Program Memory"** chapter of the specific device data sheet to determine if this option is available.

Two `NOP` instructions should be inserted in the code after the programming cycle.

Example 4-5: Row Programming with Write Latches (in Assembly)

```
; Define the address from where the programming has to start
        .equ    PROG_ADDR,0x022000
; Load the NVMADR register with the starting programming address
        MOV     #tblpage(PROG_ADDR),W9
        MOV     #tbloffset(PROG_ADDR),W8
        MOV     W9,NVMADRU
        MOV     W8,NVMADR
; Setup NVMCON to write 1 row of program memory
        MOV     #0x4002,W0
        MOV     W0,NVMCON
; Load the program memory write latches
        CALL    Load_Write_Latch_Row
; Disable interrupts < priority 7 for next 5 instructions
; Assumes no level 7 peripheral interrupts
        DISI    #06
; Write the KEY sequence
        MOV     #0x55,W0
        MOV     W0,NVMKEY
        MOV     #0xAA,W0
        MOV     W0,NVMKEY
; Start the programming sequence
        BSET    NVMCON,#15
; Insert two NOPs after programming
        NOP
        NOP
```

Example 4-6: Row Programming with Write Latches (in C)

```
int varWordL[64];
int varWordH[64];
int targetWriteAddressL;            // bits<15:0>
int targetWriteAddressH;            // bits<22:16>
int i;

NVMCON = 0x4002;                    // Set WREN and row program mode
TBLPAG = 0xFA;
NVMADR = targetWriteAddressL;       // set target write address
NVMADRU = targetWriteAddressH;

for(i=0; i<=63; i++)                // load write latches with data
{                                   // to be written
__builtin_tblwtl((i * 2), varWordL[i]);
__builtin_tblwth((i * 2), varWordH[i]);
}
__builtin_disi(5);                  // Disable interrupts for NVM unlock sequence
__builtin_write_NVM();              // initiate write
while(NVMCONbits.WR == 1);
```


4.5.4 ROW PROGRAMMING USING THE RAM BUFFER

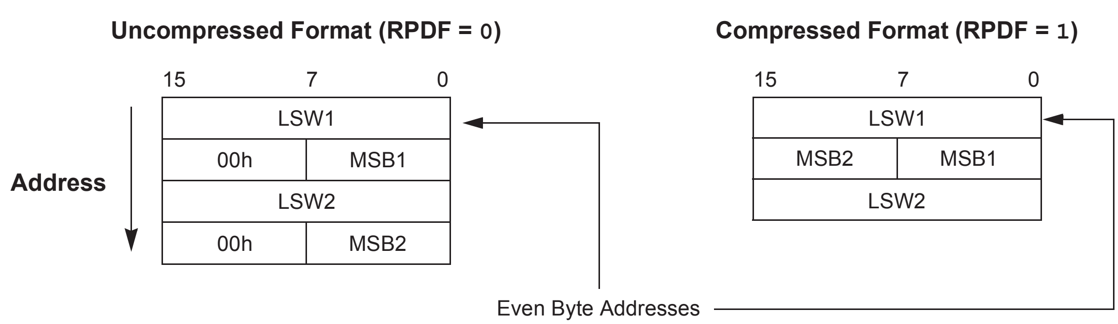
Select dsPIC33 devices permit row programming to be performed directly from a buffer space in data RAM, rather than going through the holding latches to transfer data with `TBLWT` instructions. The location of the RAM buffer is determined by the NVMSRCADR register(s), which are loaded with the data RAM address containing the first word of program data to be written.

Prior to performing the program operation, the buffer space in RAM must be loaded with the row of data to be programmed. The RAM can be loaded in either a compressed (packed) or uncompressed format. Compressed storage uses one data word to store the Most Significant Bytes (MSBs) of two adjacent program data words. The uncompressed format uses two data words for each program data word, with the upper byte of every other word being 00h. Compressed format uses about 3/4 of the space in data RAM as compared to the uncompressed format. Uncompressed format, on the other hand, mimics the structure of the 24-bit program data word, complete with the upper phantom byte. The data format is selected by the RPDF bit (NVMCON<9>). These two formats are shown in Figure 4-1.

Once the RAM buffer is loaded, the Flash Address Pointers, NVMADR and NVMADRU, are loaded with the 24-bit start address of the Flash row to be written. As with programming the write latches, the process is initiated by writing the NVM unlock sequence, followed by setting the WR bit. Once initiated, the device automatically loads the right latches and increments the NVM Address registers until all bytes have been programmed. Example 4-7 shows an example of the process. If NVMSRCADR is set to a value such that a data underrun error condition occurs, the URERR bit (NVMCON<8>) will be set to indicate the condition.

Devices which implement RAM buffer row programming also implement one or two write latches. These are loaded using the `TBLWT` instructions and are used to perform word programming operations.

Figure 4-1: Uncompressed and Compressed Storage Formats for Program Data



Example 4-7: Writing Program Memory from a Data RAM Buffer (in C)

```
int data[128];                    // Data to be programmed in RAM
int targetWriteAddressL;          // bits<15:0>
int targetWriteAddressH;          // bits<22:16>

NVMCON = 0x4002;                  // Row programming
NVMCONbits.RPDF = 0;              // Select uncompressed format
NVMSRCADRL = (int)&data[0];       // Start address of data in RAM
NVMADR = targetWriteAddressL;
NVMADRU = targetWriteAddressH;

__builtin_disi(5);                // Disable interrupts for NVM unlock sequence
__builtin_write_NVM();
while(NVMCONbits.WR == 1);
```


4.5.5 WORD PROGRAMMING

The NVMCON register is configured to program two instruction words of Flash program memory. The program operation is initiated by writing a special unlock, or key sequence, to the NVMKEY register before setting the WR bit (NVMCON<15>). The unlock sequence needs to be executed in the exact order, as shown in Example 4-8, without interruption. Therefore, interrupts should be disabled prior to writing the sequence.

Two `NOP` instructions should be inserted in the code after the programming cycle.

Example 4-8: Two-Word Write (In Assembly)

```
; Define the address from where the programming has to start

.equ PROG_ADDR, 0x022000;
; Load the destination address to be written
       MOV    #tblpage(PROG_ADDR),W9
       MOV    #tbloffset(PROG_ADDR),W8
       MOV    W9,NVMADRU
       MOV    W8,NVMADR;

; Load the two words into the latches
       CALL   Load_Write_Latch_Word
; Setup NVMCON for word programming
       MOV    #0x4001,W0
       MOV    W0,NVMCON
; Disable interrupts < priority 7 for next 5 instructions
; Assumes no level 7 peripheral interrupts
       DISI   #06
; Write the key sequence
       MOV    #0x55,W0
       MOV    W0,NVMKEY
       MOV    #0xAA,W0
       MOV    W0,NVMKEY
; Start the write cycle
       BSET   NVMCON,#15
       NOP
       NOP
```

Example 4-9: Two-Word Write (in C)

```
int varWord1L = 0xXXXX;
int varWord1H = 0x00XX;
int varWord2L = 0xXXXX;
int varWord2H = 0x00XX;
int TargetWriteAddressL;            // bits<15:0>
int TargetWriteAddressH;            // bits<22:16>

NVMCON = 0x4001;                    // Set WREN and word program mode
TBLPAG = 0xFA;                      // write latch upper address
NVMADR = TargetWriteAddressL;       // set target write address
NVMADRU = TargetWriteAddressH;

__builtin_tblwtl(0,varWord1L);      // load write latches
__builtin_tblwth(0,varWord1H);
__builtin_tblwtl(0x2,varWord2L);
__builtin_tblwth(0x2,varWord2H);
__builtin_disi(5);                  // Disable interrupts for NVM unlock sequence
__builtin_write_NVM();              // initiate write
while(NVMCONbits.WR == 1);
```

4.6 Writing to Device Configuration Registers

On certain devices, the Configuration bits are stored in configuration memory space in a section called “Device Configuration Registers”. On other devices, the Configuration bits are stored in the last page of program Flash user memory space in a section called “Flash Configuration Bytes”. With these devices, performing a page erase operation on the last page of program memory erases the Flash Configuration bytes, which enables code protection. Therefore, users should not perform page erase operations on the last page of program memory. Refer to the Program Memory Map in the **“Memory Organization”** chapter of the specific device data sheet to determine where Configuration bits are located.

When the Configuration bits are stored in configuration memory space, RTSP can be used to write to the device Configuration registers, and RTSP allows each Configuration register to be individually rewritten without first performing an erase cycle. Caution must be exercised when writing the Configuration registers since they control critical device operating parameters, such as the system clock source, PLL and WDT enable.

The procedure for programming a device Configuration register is similar to the procedure for programming Flash program memory, except that 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. Refer to **“Device Configuration”** (DS70000618) in the *“dsPIC33/PIC24 Family Reference Manual*” and the **“Special Features”** chapter in the specific device data sheet for a full description of the device Configuration registers.

> **Note 1:** Writing to device Configuration registers is not available in all devices. Refer to the **“Special Features”** chapter in the specific device data sheet to determine the modes that are available according to the device-specific 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 the 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, the user must ensure that the Clock Switching Mode bits (FCKSM<1:0>) in the FOSC register have an initial programmed value of ‘`0`’, prior to performing this RTSP operation.

4.6.1 CONFIGURATION REGISTER WRITE ALGORITHM

The general procedure is as follows:

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 address of the Configuration register to be programmed into the NVMADRU and NVMADR registers.
5. Write the key sequence to the NVMKEY register.
6. Start the write sequence by setting the WR bit (NVMCON<15>).
7. Re-enable interrupts, if needed.

Example 4-10 shows the code sequence that can be used to modify a device Configuration register.

Example 4-10: Configuration Register Write Code

```
; Define the address to be written
       .equ    DestinationAddress, 0xF80000
; Initialize the write pointer for writing to the latches
       MOV     #0x0000, W7
; Initialize TBLPAG register for writing to the latches
       MOV     #0xFA, W12
       MOV     W12, TBLPAG
; Get the new data to write to the configuration register
       MOV     #ConfigValue,W1
; Perform the table write to load the write latch
       TBLWTL  W1,[W7]
; Load the address which is to be programmed
       MOV     #DestinationAddress<15:0>,W2
       MOV     #DestinationAddress<23:16>,W3
       MOV     W3,NVMADRU
       MOV     W2,NVMADR
; Configure NVMCON for a configuration register write
       MOV     #0x4000,W0
       MOV     W0,NVMCON
; Disable interrupts < priority 7 for next 5 instructions
; Assumes no level 7 peripheral interrupts
       DISI    #06
; Write the KEY sequence
       MOV     #0x55,W0
       MOV     W0,NVMKEY
       MOV     #0xAA,W0
       MOV     W0,NVMKEY
; Start the programming sequence
       BSET    NVMCON,#15
; Insert two NOPs after programming
       NOP
       NOP
```

5.0 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	NVMSIDL	—	—	RPDF	URERR	—	—	—	—	NVMOP<3:0>				0000
NVMADRU	—	—	—	—	—	—	—	—	NVMADRU<7:0>								0000
NVMADR	NVMADR<15:0>																0000
NVMKEY	—	—	—	—	—	—	—	—	NVMKEY<7:0>								0000
NVMSRCADR[(1)]	NVMSRCADR<15:0>																0000

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

Note 1: Not available for all devices. Refer to the specific device data sheet for details.

6.0 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 dsPIC33/PIC24 product families, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to Flash Programming are:

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

> **Note:** Please visit the Microchip website (www.microchip.com) for additional Application Notes and code examples for the dsPIC33/PIC24 families of devices.

7.0 REVISION HISTORY

Revision A (August 2009)

This is the initial released version of this document.

Revision B (February 2011)

This revision includes the following updates:

- Examples:
  - Removed Example 5-3 and Example 5-4
  - Updated Example 4-1, Example 4-5 and Example 4-10
  - Any references to `#WR` were updated to `#15` in Example 4-1, Example 4-5 and Example 4-8
  - Updated the following in Example 4-3:
    - Updated the title "Word Programming" to "Loading Write Latches for Row Programming"
    - Any reference to `#ram_image` was updated to `#0xFA`
  - Added Example 4-4
  - Updated the title in Example 4-8
- Notes:
  - Added two notes in **Section 4.2 "Flash Programming Operations"**
  - Updated the note in **Section 4.2.3 "Loading Write Latches"**
  - Added three notes in **Section 4.3 "Writing to Device Configuration Registers"**
  - Added Note 1 in Table 5-1
- Registers:
  - Updated the bit values for NVMOP<3:0>: NVM Operation Select bits in the Flash Memory Control (NVMCON) register (see Register 3-1)
- Sections:
  - Removed sections **5.2.1.4 "Write Word Mode"** and **5.2.1.5 "Write Byte Mode"**
  - Updated **Section 3.0 "Control Registers"**
  - Updated the following in **Section 4.2.6 "Word Programming"**:
    - Changed the section title "Programming One Word of Flash Memory" to "Word Programming"
    - Updated the first paragraph
    - Changed the terms "one word" to "a pair of words" in the second paragraph
  - Added a new Step 1 to **Section 4.3.1 "Configuration Register Write Algorithm"**
- Tables:
  - Updated Table 5-1
- A few references to program memory were updated to Flash program memory
- Other minor updates such as language and formatting updates were incorporated throughout the document

Revision C (June 2011)

This revision includes the following updates:

- Examples:
  - Updated Example 4-1
  - Updated Example 4-8
- Notes:
  - Added a note in **Section 4.1 "RTSP Operation"**
  - Added Note 3 in **Section 4.2 "Flash Programming Operations"**
  - Added Note 3 in **Section 4.2.1 "RTSP Programming Algorithm"**
  - Added a note in **Section 4.2.2 "Erasing One Page of Flash"**
  - Added Note 2 in **Section 4.2.3 "Loading Write Latches"**
- Registers:
  - Updated the bit description for bits 15-0 in the Nonvolatile Memory Address register (see Register 3-3)
- Sections:
  - Updated **Section 4.1 "RTSP Operation"**
  - Updated **Section 4.2.6 "Word Programming"**
- Other minor updates such as language and formatting updates were incorporated throughout the document

Revision D (December 2011)

This revision includes the following updates:

- Updated **Section 2.1.3 "Table Write Latches"**
- Updated **Section 3.2 "NVMKEY Register"**
- Updated the notes in NVMCON: Flash Memory Control Register (see Register 3-1)
- Extensive updates were made throughout **Section 4.0 "Run-Time Self-Programming (RTSP)"**
- Other minor updates such as language and formatting updates were incorporated throughout the document

Revision E (October 2018)

This revision includes the following updates:

- Added Example 2-2, Example 4-2, Example 4-6 and Example 4-9
- Added **Section 4.2.5 "Row Programming Using the RAM Buffer"**
- Updated **Section 1.0 "Introduction"**, **Section 3.3 "NVM Address Registers"**, **Section 4.0 "Run-Time Self-Programming (RTSP)"** and **Section 4.2.4 "Single Row Programming Example"**
- Updated Register 3-1
- Updated Example 4-7
- Updated Table 5-1

NOTES:

ISBN: 978-1-5224-3698-0

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

Austin, TX
Tel: 512-257-3370

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

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

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

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

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

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

Finland - Espoo
Tel: 358-9-4520-820

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

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

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

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

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

Italy - Padova
Tel: 39-049-7625286

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

Norway - Trondheim
Tel: 47-7288-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

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

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820