
Section 5. Flash and EEPROM Programming

HIGHLIGHTS

This section of the manual contains the following topics:

5.1	Introduction	5-2
5.2	Table Instruction Operation	5-2
5.3	Control Registers	5-5
5.4	Run-Time Self-Programming (RTSP)	5-10
5.5	Data EEPROM Programming	5-15
5.6	Design Tips	5-21
5.7	Related Application Notes.....	5-22
5.8	Revision History	5-23

5.1 Introduction

This section describes programming techniques for Flash program memory and data EEPROM memory. The dsPIC30F family of devices contains internal program Flash memory for executing user code. There are two methods by which the user can program this memory:

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

RTSP is performed by the user's software. ICSP is performed using a serial data connection to the device and allows much faster programming times than RTSP. RTSP techniques are described in this chapter. The ICSP protocol is described in the dsPIC30F Programming Specification document, which may be downloaded from the Microchip web site.

The data EEPROM is mapped into the program memory space. The EEPROM is organized as 16-bit wide memory and the memory size can be up to 2K words (4 Kbytes). The amount of EEPROM is device dependent. Refer to the device data sheet for further information.

The programming techniques used for the data EEPROM are similar to those used for Flash program memory RTSP. The key difference between Flash and data EEPROM programming operations is the amount of data that can be programmed or erased during each program/erase cycle.

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 dsPIC30F devices. A summary of the table instructions is provided here since they are used during programming of the Flash program memory and data EEPROM. There are four basic table instructions:

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

The TBLRDL and the TBLWTL instructions are used to read and write to bits <15:0> of program memory space. TBLRDL and TBLWTL can access program memory in Word or Byte mode.

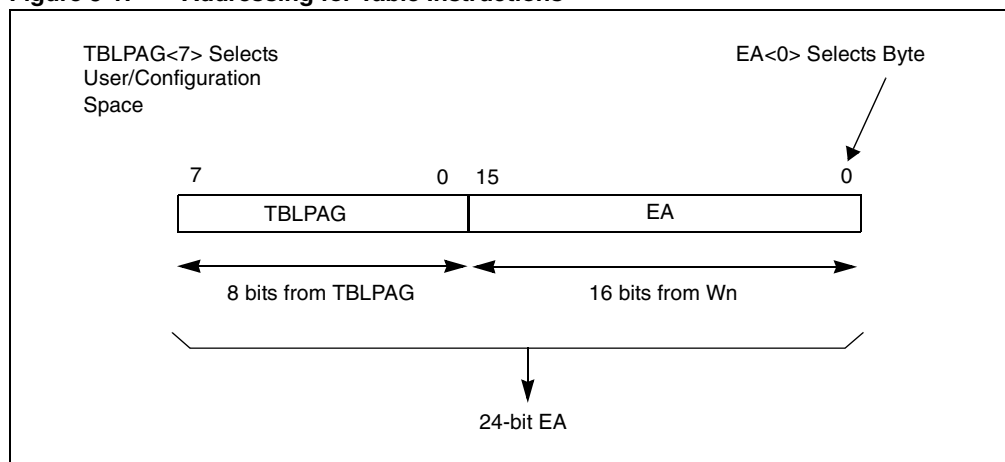
The TBLRDH and TBLWTH instructions are used to read or write to bits <23:16> of program memory space. TBLRDH and TBLWTH can access program memory in Word or Byte mode. Since the program memory is only 24-bits wide, the TBLRDH and TBLWTH instructions have the ability to address an upper byte of program memory that does not exist. This byte is called the 'phantom byte'. Any read of the phantom byte will return 0x00 and a write to the phantom byte has no effect.

Always remember that 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 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 pick 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. First, an address pointer is setup using the TBLPAG register and one of the W registers. Then, the program memory contents at the address location may be read.

5.2.1.1 Read Word Mode

The following code example shows how to read a word of program memory using the table instructions 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
; Read the program memory location
TBLRDH  [W0],W3                  ; Read high byte to W3
TBLRDL  [W0],W4                  ; Read low word to W4
```

5.2.1.2 Read 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
; 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
```

In the code example above, 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 dsPIC30F. These directives select the appropriate TBLPAG and W register values for the table instruction from a program memory address value. Refer to the MPLAB ASM 30, MPLAB LINK30 and Utilities User's Guide (DS51317) for further details.

5.2.2 Using Table Write Instructions

The effect of a table write instruction will depend on the type of memory technology that is present in the device program memory address space. The program memory address space could contain volatile or non-volatile program memory, non-volatile data memory, and an External Bus Interface (EBI). If a table write instruction occurs within the EBI address region, for example, the write data will be placed onto the EBI data lines.

5.2.2.1 Table Write Holding Latches

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

The number of holding latches will determine the maximum memory block size that can be programmed and may vary depending on the type of non-volatile memory and the device variant. For example, the number of holding latches could be different for program memory, data EEPROM memory and Device Configuration registers for a given device.

In general, the program memory is segmented into rows and panels. Each panel will have its own set of table write holding latches. This allows multiple memory panels to be programmed at once, reducing the overall programming time for the device. For each memory panel, there are generally enough holding latches to program one row of memory at a time. The memory logic automatically decides which set of write latches to load based on the address value used in the table write instruction.

Please refer to the specific device data sheet for further details.

5.2.2.2 Write Word Mode

The following sequence can be used to write 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 this example, the contents 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 2, after the second TBLWTH instruction, to prepare for the write to the next program memory location.

5.2.2.3 Write Byte Mode

To write a single program memory latch location in Byte mode, the following code sequence can be used:

```
; 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 the code example above, 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.

5.3 Control Registers

Flash and data EEPROM programming operations are controlled using the following Non-Volatile Memory (NVM) control registers:

- NVMCON: Non-Volatile Memory Control Register
- NVMKEY: Non-Volatile Memory Key Register
- NVMADR: Non-Volatile Memory Address Register

5.3.1 NVMCON Register

The NVMCON register is the primary control register for Flash and EEPROM program/erase operations. This register selects Flash or EEPROM memory, whether an erase or program operation will 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 NVMCOM configures the type of NVM operation that will be performed. For convenience, a summary of NVMCON setup values for various program and erase operations is given in Table 5-1.

Table 5-1: NVMCON Register Values

NVMCON Register Values for RTSP Program and Erase Operations			
Memory Type	Operation	Data Size	NVMCON Value
Flash PM	Erase	1 row (32 instr. words)	0x4041
	Program	1 row (32 instr. words)	0x4001
Data EEPROM	Erase	1 data word	0x4044
		16 data words	0x4045
		Entire EEPROM	0x4046
	Program	1 data word	0x4004
		16 data words	0x4005
Configuration Register	Write ⁽¹⁾	1 config. register	0x4008

Note 1: The Device Configuration registers, except for FG5, may be written to a new value without performing an erase cycle.

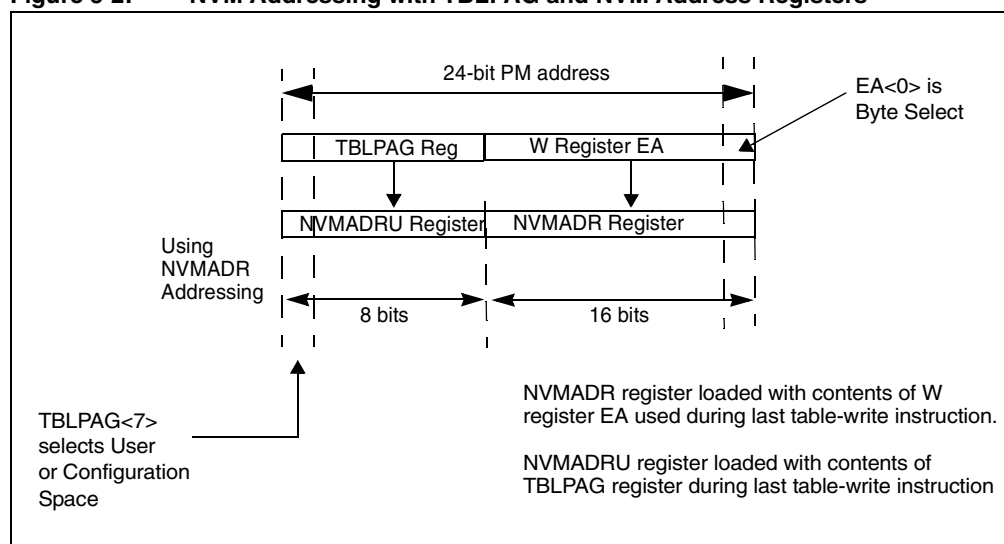
5.3.2 NVM Address Register

There are two NVM Address Registers - NVMADRU and NVMADR. These two registers when concatenated form the 24-bit effective address (EA) of the selected row or word for programming operations. The NVMADRU register is used to hold the upper 8 bits of the EA, while the NVMADR register is used to hold the lower 16 bits of the EA.

The register pair, NVMADRU:NVMADR, capture the EA<23:0> of the last table-write instruction that has been executed and select the row of Flash or EEPROM memory to write/erase. Figure 5-2 shows how the program memory EA is formed for programming and erase operations.

Although the NVMADRU and NVMADR registers are automatically loaded by the table-write instructions, the user can also directly modify their contents before the programming operation begins. A write to these registers will be required prior to an erase operation, because no table-write instructions are required for any erase operation.

Figure 5-2: NVM Addressing with TBLPAG and NVM Address Registers



5.3.3 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 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 will simply need to set the WR bit in the NVMCON register to start the program or erase cycle. Interrupts should be disabled during the unlock sequence. The code example below shows how the unlock sequence is performed:

```

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

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

Refer to **Section 5.4.2 “Flash Programming Operations”** for further programming examples.

Section 5. Flash and EEPROM Programming

Register 5-1: NVMCON: Non-Volatile Memory Control Register

Upper Byte:							
R/S-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
WR	WREN	WRERR	—	—	—	—	—
bit 15							
							bit 8

Lower Byte:							
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PROGOP<7:0>							
bit 7							bit 0

- bit 15 **WR:** Write (Program or Erase) Control bit
 1 = Initiates a data EEPROM or program Flash erase or write cycle
 (the WR bit can be set but not cleared in software)
 0 = Write cycle is complete
- bit 14 **WREN:** Write (Erase or Program) Enable bit
 1 = Enable an erase or program operation
 0 = No operation allowed (Device clears this bit on completion of the write/erase operation)
- bit 13 **WRERR:** Flash Error Flag bit
 1 = A write operation is prematurely terminated (any $\overline{\text{MCLR}}$ or WDT Reset during programming operation)
 0 = The write operation completed successfully
- bit 12-8 **Reserved:** User code should write '0's to these locations
- bit 7-0 **PROGOP<7:0>:** Programming Operation Command Byte bits

Erase Operations:

- 0x41 = Erase 1 row (32 instruction words) of program Flash
 0x44 = Erase 1 data word from data EEPROM
 0x45 = Erase 1 row (16 data words) from data EEPROM
 0x46 = Erase entire data EEPROM

Programming Operations:

- 0x01 = Program 1 row (32 instruction words) into Flash program memory
 0x04 = Program 1 data word into data EEPROM
 0x05 = Program 1 row (16 data words) into data EEPROM
 0x08 = Program 1 data word into device configuration register

Legend:

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

dsPIC30F Family Reference Manual

Register 5-2: NVMADR: Non-Volatile Memory Address Register

Upper Byte:							
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	

Lower Byte:							
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	

bit 15-0 **NVMADR<15:0>**: NV Memory Write Address bits
Selects the location to program or erase in program or data Flash memory.
This register may be read or written by user. This register will contain the address of EA<15:0> of the last table write instruction executed, until written by the user.

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

Section 5. Flash and EEPROM Programming

Register 5-3: NVMADRU: Non-Volatile Memory Upper Address Register

Upper Byte:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15				bit 8			

Lower Byte:							
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			

bit 15-8 **Unimplemented:** Read as '0'

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

Selects the upper 8 bits of the location to program or erase in program or data Flash memory.

This register may be read or written by the user. This register will contain the value of the TBLPAG register when the last table write instruction executed, until written by the user.

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

Register 5-4: NVMKEY: Non-Volatile Memory Key Register

Upper Byte:							
U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15				bit 8			

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

bit 15-8 **Unimplemented:** Read as '0'

bit 7-0 **NVMKEY<7:0>:** Key Register (Write Only) bits

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

5.4 Run-Time Self-Programming (RTSP)

RTSP allows the user code 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 may erase program memory, 32 instructions (96 bytes) at a time and can write program memory data, 32 instructions (96 bytes) at a time.

5.4.1 RTSP Operation

The dsPIC30F Flash program memory is organized into rows and panels. Each row consists of 32 instructions or 96 bytes. The panel size may vary depending on the dsPIC30F device variant. Refer to the device data sheet for further information. Typically, each panel consists of 128 rows, or 4K x 24 instructions. RTSP allows the user to erase one row (32 instructions) at a time and to program 32 instructions at one time.

Each panel of program memory contains write latches that hold 32 instructions of programming data. These latches are not memory mapped. The only way for the user to access the write latches is through the use of table write instructions. Prior to the actual programming operation, the write data must be loaded into the panel write latches with table write instructions. The data to be programmed into the panel is typically loaded in sequential order into the write latches: instruction 0, instruction 1, etc. The instruction words loaded must always be from an 'even' group of four address boundaries (e.g., loading of instructions 3, 4, 5, 6 is not allowed). Another way of stating this requirement is that the starting program memory address of the four instructions must have the 3 LSb's equal to '0'. All 32 write latches must be written during a programming operation to ensure that any old data held in the latches is overwritten.

The basic sequence for RTSP programming is to setup a table pointer, then do a series of `TBLWT` instructions to load the write latches. Programming is performed by setting special bits in the NVMCON register. 32 `TBLWTL` and 32 `TBLWTH` instructions are required to load the four instructions. If multiple, discontinuous regions of program memory need to be programmed, the table pointer should be changed for each region and the next set of write latches written.

All of the table write operations to the Flash program memory take 2 instruction cycles each, because only the table latches are written. The actual programming operation is initiated using the NVMCON register.

5.4.2 Flash Programming Operations

A program/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 and is nominally 2 msec in duration. Setting the WR bit (NVMCON<15>) starts the operation and 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 instruction or respond to interrupts during this time. If any interrupts do occur during the programming cycle, then they will remain pending until the cycle completes.

Section 5. Flash and EEPROM Programming

5.4.2.1 Flash Program Memory Programming Algorithm

The user can erase and program Flash Program Memory by rows (32 instruction words). The general process is as follows:

1. Read one row of program Flash (32 instruction words) and store into data RAM as a data “image”. The RAM image must be read from an even 32-word program memory address boundary.
2. Update the RAM data image with the new program memory data.
3. Erase program Flash row.
 - Setup NVMCON register to erase 1 row of Flash program memory.
 - Write address of row to be erased into NVMADRU and NVMADR registers.
 - Disable interrupts.
 - Write the key sequence to NVMKEY to enable the erase.
 - Set the WR bit. This will begin erase cycle.
 - CPU will stall for the duration of the erase cycle.
 - The WR bit is cleared when erase cycle ends.
 - Re-enable interrupts.
4. Write 32 instruction words of data from RAM into the Flash program memory write latches.
5. Program 32 instruction words into program Flash.
 - Setup NVMCON to program one row of Flash program memory.
 - Disable interrupts.
 - Write the key sequence to NVMKEY to enable the program cycle.
 - Set the WR bit. This will begin the program cycle.
 - CPU will stall for duration of the program cycle.
 - The WR bit is cleared by the hardware when program cycle ends.
 - Re-enable interrupts.
6. Repeat steps 1 through 6, as needed, to program the desired amount of Flash program memory

Note: The user should remember that the minimum amount of program memory that can be modified using RTSP is 32 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. An erase cycle must be performed on any previously written locations before any programming is done.

5.4.2.2 Erasing a Row of Program Memory

The following is a code sequence that can be used to erase a row (32 instructions) of program memory. The NVMCON register is configured to erase one row of program memory. The NVMADRU and NVMADR registers are loaded with the address of the row to be erased. The program memory must be erased at 'even' row boundaries. Therefore, the 6 LSbits of the value written to the NVMADR register have no effect when a row is erased.

The erase operation is initiated by writing a special unlock, or key sequence to the NVMKEY register before setting the WR control bit (NVMCON<15>). The unlock sequence needs to be executed in the exact order 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).

```
; Setup NVMCON to erase one row of Flash program memory
MOV    #0x4041,W0
MOV    W0,NVMCON
; Setup address pointer to row to be ERASED
MOV    #tblpage(PROG_ADDR),W0
MOV    W0,NVMADRU
MOV    #tbloffset(PROG_ADDR),W0
MOV    W0,NVMADR
; 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 erase operation
BSET   NVMCON,#WR
; Insert two NOPs after the erase cycle (required)
NOP
NOP
; Re-enable interrupts, if needed
POP    SR
```

Note: When erasing a row of program memory, the user writes the upper 8 bits of the erase address directly to the NVMADRU and NVMADR registers. Together, the contents of the NVMADRU and NVMADR registers form the complete address of the program memory row to be erased.

The NVMADRU and NVMADR registers specify the address for all Flash erase and program operations. However, these two registers do not have to be directly written by the user for Flash program operations. This is because the table write instructions used to write the program memory data automatically transfers the TBLPAG register contents and the table write address into the NVMADRU and NVMADR registers.

The above code example could be modified to perform a 'dummy' table write operation to capture the program memory erase address.

5.4.2.3 Loading Write Latches

The following is a sequence of instructions that can be used to load the 768-bits of write latches (32 instruction words). 32 TBLWTL and 32 TBLWTH instructions are needed to load the write latches selected by the table pointer.

The TBLPAG register is loaded with the 8 MSbits of the program memory address. The user does not need to write the NVMADRU:NVMADR register-pair for a Flash programming operation. The 24-bits of the program memory address are automatically captured into the NVMADRU:NVMADR register-pair when each table write instruction is executed. The program memory must be programmed at an 'even' 32 instruction word address boundary. In effect, the 6 LSbits of the value captured in the NVMADR register are not used during the programming operation.

The row of 32 instruction words do not necessarily have to be written in sequential order. The 6 LSbits of the table write address determine which of the latches will be written. However, all 32 instruction words should be written for each programming cycle to overwrite old data.

Note: The following code example is the 'Load_Write_Latch' code referred to in subsequent examples.

```
; 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          #LOW_WORD_0,W2
MOV          #HIGH_BYTE_0,W3
TBLWTL      W2,[W0]
TBLWTH      W3,[W0++]      ; 1st_program_word
MOV          #LOW_WORD_1,W2
MOV          #HIGH_BYTE_1,W3
TBLWTL      W2,[W0]
TBLWTH      W3,[W0++]      ; 2nd_program_word
MOV          #LOW_WORD_2,W2
MOV          #HIGH_BYTE_2,W3
TBLWTL      W2,[W0]
TBLWTH      W3,[W0++]      ; 3rd_program_word
MOV          #LOW_WORD_3,W2
MOV          #HIGH_BYTE_3,W3
TBLWTL      W2,[W0]
TBLWTH      W3,[W0++]      ; 4th_program_word
.....
.....
MOV          #LOW_WORD_31,W2
MOV          #HIGH_BYTE_31,W3
TBLWTL      W2,[W0]
TBLWTH      W3,[W0++]      ; 32nd_program_word
```

5.4.2.4 Single Row Programming Example

An example of single row programming code is:

```
; Setup NVMCON to write 1 row of program memory
MOV     #0x4001,W0
MOV     W0,NVMCON

; Load the 32 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 1: See Section 5.4.2.3 “Loading Write Latches”

5.4.3 Writing to Device Configuration Registers

RTSP may be used to write to the Device Configuration registers. RTSP allows each Configuration register, except the FG5, 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 multiplication ratio and WDT enable.

The procedure for programming a Device Configuration register is similar to the procedure for Flash program memory, except that only TBLWTL instructions are required. This is because the upper 8 bits are unused in each Device Configuration register. Furthermore, bit 23 of the table write address must be set to access the Configuration registers. Refer to **Section 24. “Device Configuration”** and the device data sheet for a full description of the Device Configuration registers.

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 = 0x4008).
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.

5.4.3.2 Configuration Register Write Code Example

The following code sequence can be used to modify a Device Configuration register:

```
; 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      #0x4008,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 Data EEPROM Programming

The EEPROM block is accessed using table read and write operations similar to the program memory. The TBLWTH and TBLRDH instructions are not required for EEPROM operations since the memory is only 16-bits wide. The program and erase procedures for the data EEPROM are similar to those used for the Flash program memory, except they are optimized for fast data access. The following programming operations can be performed on the data EEPROM:

- Erase one word
- Erase one row (16 words)
- Erase entire data EEPROM
- Program one word
- Program one row (16 words)

The data EEPROM is readable and writable during normal operation (full VDD operating range). Unlike the Flash program memory, normal program execution is not stopped during an EEPROM program or erase operation.

EEPROM erase and program operations are performed using the NVMCON and NVMKEY registers. The programming software is responsible for waiting for the operation to complete. The software may detect when the EEPROM erase or programming operation is complete by one of three methods:

- Poll the WR bit (NVMCON<15>) in software. The WR bit will be cleared when the operation is complete.
- Poll the NVMIF bit (IFS0<12>) in software. The NVMIF bit will be set when the operation is complete.
- Enable NVM interrupts. The CPU will be interrupted when the operation is complete. Further programming operations can be handled in the ISR.

Note: Unexpected results will be obtained should the user attempt to read the EEPROM while a programming or erase operation is underway.

5.5.1 EEPROM Single Word Programming Algorithm

1. Erase one EEPROM word.
 - Setup NVMCON register to erase one EEPROM word.
 - Write address of word to be erased into NVMADRU, NVMADR registers.
 - Clear NVMIF status bit and enable NVM interrupt (optional).
 - Write the key sequence to NVMKEY.
 - Set the WR bit. This will begin erase cycle.
 - Either poll the WR bit or wait for the NVM interrupt.
2. Write data word into data EEPROM write latch.
3. Program the data word into the EEPROM.
 - Setup the NVMCON register to program one EEPROM word.
 - Clear NVMIF status bit and enable NVM interrupt (optional).
 - Write the key sequence to NVMKEY.
 - Set the WR bit. This will begin the program cycle.
 - Either poll the WR bit or wait for the NVM interrupt.

5.5.2 EEPROM Row Programming Algorithm

If multiple words need to be programmed into the EEPROM, it is quicker to erase and program 16 words (1 row) at a time. The process to program 16 words of EEPROM is:

1. Read one row of data EEPROM (16 words) and store into data RAM as a data “image”. The section of EEPROM to be modified must fall on an even 16-word address boundary.
2. Update the data image with the new data.
3. Erase the EEPROM row.
 - Setup the NVMCON register to erase one row of EEPROM.
 - Write starting address of row to be erased into NUMADRU and NVMADR registers.
 - Clear NVMIF status bit and enable NVM interrupt (optional).
 - Write the key sequence to NVMKEY.
 - Set the WR bit. This will begin the erase cycle.
 - Either poll the WR bit or wait for the NVM interrupt.
4. Write the 16 data words into the data EEPROM write latches.
5. Program a row into data EEPROM.
 - Setup the NVMCON register to program one row of EEPROM.
 - Clear NVMIF status bit and enable NVM interrupt (optional).
 - Write the key sequence to NVMKEY.
 - Set the WR bit. This will begin the program cycle.
 - Either poll the WR bit or wait for the NVM interrupt.

5.5.3 Erasing One Word of Data EEPROM Memory

The NVMADRU and NVMADR registers must be loaded with the data EEPROM address to be erased. Since one word of the EEPROM is accessed, the LSB of the NVMADR has no effect on the erase operation. The NVMCON register must be configured to erase one word of EEPROM memory.

Setting the WR control bit (NVMCON<15>) initiates the erase. A special unlock or key sequence should be written to the NVMKEY register before setting the WR control bit. The unlock sequence needs to be executed in the exact order shown without interruption. Therefore, interrupts should be disabled prior to writing the sequence.

```
; Set up a pointer to the EEPROM location to be erased.
MOV      #tblpage(EE_ADDR),W0
MOV      W0,NVMADRU
MOV      #tbloffset(EE_ADDR),W0
MOV      W0,NVMADR
; Setup NVMCON to erase one word of data EEPROM
MOV      #0x4044,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 cycle
BSET     NVMCON,#WR
; Re-enable interrupts
POP      SR
```

5.5.4 Writing One Word of Data EEPROM Memory

Assuming the user has erased the EEPROM location to be programmed, use a table write instruction to write one write latch. The TBLPAG register is loaded with the 8 MSBs of the EEPROM address. The 16 LSBs of the EEPROM address are automatically captured into the NVMADR register when the table write is executed. The LSB of the NVMADR register has no effect on the programming operation. The NVMCON register is configured to program one word of data EEPROM.

Setting the WR control bit (NVMCON<15>) initiates the programming operation. A special unlock or key sequence should be written to the NVMKEY register before setting the WR control bit. The unlock sequence needs to be executed in the exact order shown without interruption. Therefore, interrupts should be disabled prior to writing the sequence.

```
; Setup a pointer to data EEPROM
MOV      #tblpage(EE_ADDR),W0
MOV      W0,TBLPAG
MOV      #tbloffset(EE_ADDR),W0
; Write data value to holding latch
MOV      EE_DATA,W1
TBLWTL   W1,[ W0]
; NVMADR captures write address from the TBLWTL instruction.
; Setup NVMCON for programming one word to data EEPROM
MOV      #0x4004,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
```

5.5.5 Erasing One Row of Data EEPROM

The NVMCON register is configured to erase one row of EEPROM memory. The NVMADRU and NVMADR registers must point to the row to be erased. The data EEPROM must be erased at even address boundaries. Therefore, the 5 LSBs of the NVMADR register will have no effect on the row that is erased.

Setting the WR control bit (NVMCON<15>) initiates the erase. A special unlock or key sequence should be written to the NVMKEY register before setting the WR control bit. The unlock sequence needs to be executed in the exact order shown without interruption. Therefore, interrupts should be disabled prior to writing the sequence.

```
; Set up a pointer to the EEPROM row to be erased.
MOV      #tblpage (EE_ADDR), W0
MOV      W0, NVMADRU
MOV      #tbloffset (EE_ADDR), W0
MOV      W0, NVMADR
; Setup NVMCON to erase one row of EEPROM
MOV      #0x4045, 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
; Re-enable interrupts, if needed
POP      SR
```

5.5.6 Write One Row of Data EEPROM Memory

To write a row of data EEPROM, all sixteen write latches must be written before the programming sequence is initiated. The TBLPAG register is loaded with the 8 MSbs of the EEPROM address. The 16 LSbs of the EEPROM address are automatically captured into the NVMADR register when each table write is executed. Data EEPROM row programming must occur at even address boundaries, so the 5 LSbs of the NVMADR register have no effect on the row that is programmed.

Setting the WR control bit (NVMCON<15>) initiates the programming operation. A special unlock or key sequence should be written to the NVMKEY register before setting the WR control bit. The unlock sequence needs to be executed in the exact order shown without interruption. Therefore, interrupts should be disabled prior to writing the sequence.

```
; Set up a pointer to the EEPROM row to be programmed.
MOV      #tblpage(EE_ADDR),W0
MOV      W0,TBLPAG
MOV      #tbloffset(EE_ADDR),W0
; Write the data to the programming latches.
MOV      data_ptr,W1      ; Use W1 as pointer to the data.
TBLWTL [W1++],[W0++]      ; Write 1st data word
TBLWTL [W1++],[W0++]      ; Write 2nd data word
TBLWTL [W1++],[W0++]      ; Write 3rd data word
TBLWTL [W1++],[W0++]      ; Write 4th data word
TBLWTL [W1++],[W0++]      ; Write 5th data word
TBLWTL [W1++],[W0++]      ; Write 6th data word
TBLWTL [W1++],[W0++]      ; Write 7th data word
TBLWTL [W1++],[W0++]      ; Write 8th data word
TBLWTL [W1++],[W0++]      ; Write 9th data word
TBLWTL [W1++],[W0++]      ; Write 10th data word
TBLWTL [W1++],[W0++]      ; Write 11th data word
TBLWTL [W1++],[W0++]      ; Write 12th data word
TBLWTL [W1++],[W0++]      ; Write 13th data word
TBLWTL [W1++],[W0++]      ; Write 14th data word
TBLWTL [W1++],[W0++]      ; Write 15th data word
TBLWTL [W1++],[W0++]      ; Write 16th data word
; The NVMADR captures last table access address.
; Setup NVMCON to write one row of EEPROM
MOV      #0x4005,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 programming operation
BSET     NVMCON,#WR
; Re-enable interrupts, if needed
POP      SR
```

Note: Sixteen table write instructions have been used in this code segment to provide clarity in the example. The code segment could be simplified by using a single table write instruction in a REPEAT loop.

5.5.7 Reading the Data EEPROM Memory

A TBLRD instruction reads a word at the current program word address. This example uses W0 as a pointer to data Flash. The result is placed into register W4.

```
; Setup pointer to EEPROM memory
MOV    #tblpage(EE_ADDR),W0
MOV    W0,TBLPAG
MOV    #tbloffset(EE_ADDR),W0
; Read the EEPROM data
TBLRDL [W0],W4
```

Note: Program Space Visibility (PSV) can also be used to read locations in the program memory address space. See **Section 4. “Program Memory”** for further information about PSV.

5.6 Design Tips

Question 1: *I cannot get the device to program or erase properly. My code appears to be correct. What could be the cause?*

Answer: Interrupts should be disabled when a program or erase cycle is initiated to ensure that the key sequence executes without interruption. Interrupts can be disabled by raising the current CPU priority to level 7. The code examples in this chapter disable interrupts by saving the current SR register value on the stack, then ORing the value 0x00E0 with SR to force IPL<2:0> = 111. If no priority level 7 interrupts are enabled, then the DISI instruction provides another method to temporarily disable interrupts, while the key sequence is executed.

Question 2: *What is an easy way to read data EEPROM without using table instructions?*

Answer: The data EEPROM is mapped into the program memory space. PSV can be used to map the EEPROM region into data memory space. See **Section 4. “Program Memory”** for further information about PSV.

5.7 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 dsPIC30F Product Family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Flash and EEPROM Programming module are:

Title	Application Note #
Using the dsPIC30F for Sensorless BLDC Control	AN901

Note: Please visit the Microchip web site (www.microchip.com) for additional Application Notes and code examples for the dsPIC30F Family of devices.

5.8 Revision History

Revision A

This is the initial released revision of this document.

Revision B

This revision incorporates technical content changes for the dsPIC30F Flash and EEPROM Programming module.

Revision C

This revision incorporates all known errata at the time of this document update.

Revision D

This revision incorporates technical content changes for the dsPIC30F Flash and EEPROM Programming module.

NOTES: