Section 5. Data EEPROM

HIGHLIGHTS

This section of the manual contains the following major topics:

5.1 INTRODUCTION

In addition to standard Flash-based program memory and volatile data RAM, certain PIC24F devices also include on-chip data EEPROM. This memory block allows users to store program or application information (such as identification, calibration constants, etc.) in a nonvolatile location, and easily rewrite the information when needed. Data EEPROM memory is based on the same Flash technology as program memory, and has been optimized for both long retention and a higher number of erase/write cycles.

The data EEPROM is mapped to the top of the user program memory space, with the top address at program memory address 7FFFFFh. The size of the EEPROM is device dependent. Refer to the specific device data sheet for further information.

The EEPROM is organized as 16-bit wide memory. Each word is directly addressable; unlike program memory, there are no “phantom bytes” in odd-numbered addresses.

The programming techniques used for the data EEPROM are similar to those used for Flash program memory RTSP, discussed in previous sections. The key differences between Flash and data EEPROM programming operations is the amount of data that can be programmed or erased during each program/erase cycle and that table writes to the data EEPROM do not suspend operation of the CPU.

5.2 CONTROL REGISTERS

Like Flash program memory, data EEPROM programming operations are controlled using the three Nonvolatile Memory (NVM) Control registers:

- NVMCON: Nonvolatile Memory Control Register
- NVMKEY: Nonvolatile Memory Key Register
- NVMADR: Nonvolatile Memory Address Register

5.2.1 NVMCON Register

The NVMCON register (Register 5-1) is the primary control register for data EEPROM program/erase operations. This upper byte contains the control bits used to start the program or erase cycle, and the flag bit to indicate if the operation was successfully performed. The lower byte of NVMCOM configures the type of NVM operation that will be performed.

The NVMCON register also controls programming/erase operations for the program memory, as described in **Section 4. “Program Memory”.**

5.2.2 NVMKEY Register

NVMKEY is a write-only register that is used to prevent accidental writes or erasures of data EEPROM locations. To start any programming or erase sequence, these two instructions must be executed first, in the exact order shown:

1. Write 55h to NVMKEY.
2. Write AAh to NVMKEY.

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 MPLAB® C30 C compiler provides a defined library procedure (`builtin_write_NVM`) to perform the unlock sequence. Example 5-1 shows how the unlock sequence can be performed with in-line assembly.

Example 5-1: Data EEPROM Unlock Sequence

```
//Disable Interrupts For 5 instructions
asm volatile("disi #5");
//Issue Unlock Sequence
asm volatile("mov #0x55, W0   \n"
             "mov W0, NVMKEY  \n"
             "mov #0xAA, W1   \n"
             "mov W1, NVMKEY  \n");
```


Register 5-1: NVMCON: Nonvolatile Memory Control Register (Data EEPROM Operations)

R/S-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
WR	WREN	WRERR	PGMONLY	r	r	r	r
bit 15							bit 8

U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
r	ERASE	r	NVMOP4(1)	NVMOP3(1)	NVMOP2(1)	NVMOP1(1)	NVMOP0(1)
bit 7							bit 0

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

bit 15 **WR:** Write (Program or Erase) Control bit
1 = Initiates a data EEPROM erase or write cycle (can be set but not cleared in software)
0 = Write cycle is complete (cleared automatically by hardware)

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 **PGMONLY:** Program Only Enable bit
1 = Write operation is executed without erasing target address(es) first
0 = Automatic erase-before-write: write operations are preceded automatically by an erase of target address(es)

bit 11-7 **Reserved:** User code should write '0's to these locations

bit 6 **ERASE:** Erase Operation Select bit
1 = Perform an erase operation when WR is set
0 = Perform a write operation when WR is set

bit 5 **Reserved:** User code should write '0's to these locations

bit 4-0 **NVMOP4:NVMOP0:** Programming Operation Command Byte bits(1)

Erase Operations (when ERASE bit is '1'):
11010 = Erase 8 words
11001 = Erase 4 words
11000 = Erase 1 word
100xx = Erase entire data EEPROM

Programming Operations (when ERASE bit is '0'):
001xx = Write 1 word

Note 1: Only those bit combinations that reflect valid data EEPROM operations are shown here. Other combinations are either unimplemented, or are used for Flash program memory or device configuration operations. Please refer to the specific device data sheet for information.

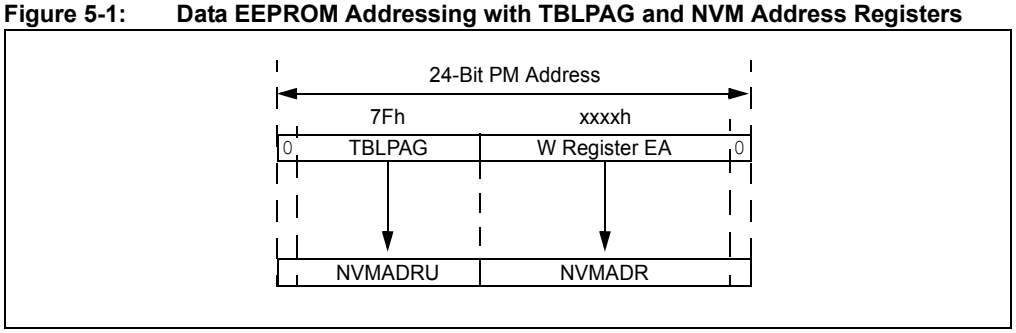
5.2.3 NVM Address Register

As with Flash program memory, the NVM Address Registers, NVMADRU and NVMADR form the 24-bit Effective Address (EA) of the selected row or word for data EEPROM 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. These registers are not mapped into the SFR space; instead, they directly capture the EA<23:0> of the last table write instruction that has been executed and selects the data EEPROM row to erase. Figure 5-1 shows how the program memory EA is formed for programming and erase operations.

Like program memory operations, the Least Significant bit (LSb) of NVMADR is restricted to even addresses. This is because any given address in the data EEPROM space consists of only the lower word of the program memory width; the upper word, including the uppermost "phantom byte", are unavailable. This means that the LSb of a data EEPROM address will always be '0'. The range of allowable values for NVMADR in data EEPROM operations is determined by the size of the particular device's EEPROM.

Similarly, the Most Significant bit (MSb) of NVMADRU is always '0', since all addresses lie in the user program space. Further, the value of TBLPAG (and thus, NVMADRU) can also be fixed at 7Fh for data EEPROM operations, since the data EEPROM address range will always be located in the top 64K page of the program memory space.

Figure 5-1: Data EEPROM Addressing with TBLPAG and NVM Address Registers



5.3 DATA EEPROM OPERATIONS

The EEPROM block is accessed using table read and write operations similar to those used for 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 that they are optimized for fast data access. The following programming operations can be performed on the data EEPROM:

- Write one word
- Erase one, four or eight words
- Bulk erase entire data EEPROM

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

Data EEPROM 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 Interrupt Service Routine (ISR).

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

The C30 C compiler includes library procedures to automatically perform the table read and table write operations, manage the Table Pointer and write buffers, and unlock and initiate memory write sequences. This eliminates the need to create assembler macros or time critical routines in C for each application.

The library procedures are used in the code examples detailed in the following sections. General descriptions of each process are provided for users who are not using the C30 compiler libraries.

5.3.1 Single-Word Write

The overall algorithm for writing to the data EEPROM is as follows:

1. Erase one data EEPROM word:
   - Set the NVMOP bits to erase one EEPROM word (NVMCON<4:0> = `11000`).
   - Write address of word to be erased into the TBLPAG and WREG registers.
   - Clear NVMIF status bit and enable NVM interrupt (optional).
   - Write the key sequence to NVMKEY.
   - Set the WR bit to begin erase cycle.
   - Either poll the WR bit or wait for the NVM interrupt.
2. Write the data word into the data EEPROM latch.
3. Program the data word into the EEPROM:
   - Set up 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 to begin erase cycle.
   - Either poll the WR bit or wait for the NVM interrupt.

A table write instruction is used to write the data to 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. The unlock sequence must 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 (see **Section 5.2.2 "NVMKEY Register"** for details). Therefore, interrupts should be disabled prior to writing the sequence.

A typical write sequence, including the erase and key unlock sequences, is shown in Example 5-2. This example uses C30 compiler library procedures to manage the Table Pointer (`builtin_tblpage` and `builtin_tbloffset`), the unlock sequence (`builtin_write_NVM`) and the actual data write (`builtin_tblwtl`). The memory unlock sequence also sets the WR bit to initiate the operation and returns control when complete.

Example 5-2: Single-Word Write to Data EEPROM

```
// Set up NVMCON to write one word of data EEPROM
NVMCON = 0x4004;

// Set up a pointer to the EEPROM location to be written
__builtin_tblpage(&ee_addr);
offset = __builtin_tbloffset(&ee_addr);

// Write Data Value To Holding Latch
__builtin_tblwtl(offset, data);

// Disable Interrupts For 5 Instructions
asm volatile ("disi #5");

// Issue Unlock Sequence & Start Write Cycle
__builtin_write_NVM();
```

5.3.2 Single-Word Erase

In single-word erase operations, the NVMADRU:NVMADR registers are loaded from the TBLPAG and WREG registers 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. The unlock or key sequence must 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 (see **Section 5.2.2 "NVMKEY Register"** for details). Therefore, interrupts should be disabled prior to writing the sequence.

A typical erase sequence is shown in Example 5-3. This example, and the other erase examples that follow, use C library procedures to manage the Table Pointer (`builtin_tblpage` and `builtin_tbloffset`) and the Erase Page Pointer (`builtin_tblwtl`). The memory unlock sequence (`builtin_write_NVM`) also sets the WR bit to initiate the operation and returns control when complete.

Example 5-3: Single-Word Erase

```
// Set up NVMCON to erase one word of data EEPROM
NVMCON = 0x4044;

// Set up a pointer to the EEPROM location to be erased
__builtin_tblpage(&ee_addr);
offset = __builtin_tbloffset(&ee_addr);
__builtin_tblwtl(offset, offset);

// Disable Interrupts For 5 Instructions
asm volatile ("disi #5");

// Issue Unlock Sequence & Start Write Cycle
__builtin_write_NVM();
```

5.3.3 Four-Word Erase

The NVMCON register is configured to erase 4 words (or one-half row) of EEPROM memory. A pointer to the data EEPROM address to be erased must first be initialized. The data EEPROM must be erased at even address boundaries. Therefore, the 3 LSbs of the Effective Address will have no effect on the memory that is erased.

Setting the WR control bit (NVMCON<15>) initiates the erase. The unlock 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 (see **Section 5.2.2 "NVMKEY Register"** for details). Therefore, interrupts should be disabled prior to writing the sequence.

Example 5-4: Four-Word Erase Sequence

```
// Set up NVMCON to erase four words of data EEPROM
NVMCON = 0x4045;

// Set up a pointer to the EEPROM location to be erased
__builtin_tblpage(&ee_addr);
offset = __builtin_tbloffset(&ee_addr);
__builtin_tblwtl(offset, offset);

// Disable Interrupts For 5 Instructions
asm volatile ("disi #5");

// Issue Unlock Sequence & Start Erase Cycle
__builtin_write_NVM();
```


5.3.4 Eight-Word Erase

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 address will have no effect on the row that is erased.

Setting the WR control bit (NVMCON<15>) initiates the erase. The unlock sequence must 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 (see **Section 5.2.2 "NVMKEY Register"** for details). Therefore, interrupts should be disabled prior to writing the sequence.

Example 5-5: Eight-Word Erase

```
// Set up NVMCON to erase eight words of data EEPROM
NVMCON = 0x4046;

// Set up a pointer to the EEPROM location to be erased
__builtin_tblpage(&ee_addr);
offset = __builtin_tbloffset(&ee_addr);
__builtin_tblwtl(offset, offset);

// Disable Interrupts For 5 Instructions
asm volatile ("disi #5");

// Issue Unlock Sequence & Start Erase Cycle
__builtin_write_NVM();
```

5.3.5 Data EEPROM Bulk Erase

The NVMCON register is configured to bulk erase the entire data EEPROM memory. Because this operation affects the entire data EEPROM, the address registers do not need to be configured.

Setting the WR control bit (NVMCON<15>) initiates the erase. The unlock sequence must 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 (see **Section 5.2.2 "NVMKEY Register"** for details). Therefore, interrupts should be disabled prior to writing the sequence.

In this bulk erase example (Example 5-6), executing the unlock library procedure automatically triggers the erase process, since it also sets the WR bit.

Example 5-6: Data EEPROM Bulk Erase

```
// Set up NVMCON to bulk erase the data EEPROM
NVMCON = 0x4050;

// Disable Interrupts For 5 Instructions
asm volatile ("disi #5");

// Issue Unlock Sequence and Start Erase Cycle
__builtin_write_NVM();
```

5.3.6 Reading the Data EEPROM

As with program memory operations, the table read instruction is used to read data from the data EEPROM. Since the EEPROM array is only 16 bits wide, only the `TBLRDL` instruction is needed. In Example 5-7, W0 is used as a pointer to the data EEPROM address. The result is placed into register, W4.

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.

The data EEPROM read example (Example 5-7) uses the Table Pointer management (`builtin_tblpage` and `builtin_tbloffset`) and table read (`builtin_tblrdl`) procedures from the C30 compiler library.

Example 5-7: Reading the Data EEPROM Using the `TBLRD` Command

```
// Set up a pointer to the EEPROM location to be read
__builtin_tblpage(&ee_addr);
offset = __builtin_tbloffset(&ee_addr);

// Read the EEPROM data
data = __builtin_tblrdl(offset);
```


5.4 REGISTER MAP

A summary of the Special Function Registers associated with the PIC24F Data EEPROM is provided in Table 5-1.

Table 5-1: Registers Associated with Data EEPROM Operation

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
TBLPAG	—	—	—	—	—	—	—	—	Table Memory Page Address Register								0000
NVMCON	WR	WREN	WRERR	PGMONLY	r	r	r	r	r	ERASE	r	NVMOP4	NVMOP3	NVMOP2	NVMOP1	NVMOP0	0000(1)
NVMKEY	—	—	—	—	—	—	—	—	NVMKEY<7:0>								0000

Legend: — = unimplemented, read as '0'; r = reserved, user code should write '0's to these locations

Note 1: Reset value shown is for POR only. Value on other Reset states is dependent on the state of memory write or erase operations at the time of Reset.

5.5 DESIGN TIPS

Question 1: ***I cannot get the data EEPROM 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 using the `DISI` instruction, or by raising the current CPU priority to level 7.

The code examples shown in this chapter disable interrupts for a specified number of instruction cycles with the `DISI` instruction. An alternate method to temporarily disable interrupts is by saving the current SR register value on the stack, then ORing the value 00E0h with SR to force IPL<2:0> = `111`.

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.

5.6 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 PIC24F device family, but the concepts are pertinent and could be used with modification and possible limitations. The current application notes related to the Data EEPROM are:

Title	Application Note #
Emulating Data EEPROM for PIC18 and PIC24 Microcontrollers and dsPIC® Digital Signal Controllers	AN1095

Note: Please visit the Microchip web site (www.microchip.com) for additional application notes and code examples for the PIC24F family of devices.

5.7 REVISION HISTORY

Revision A (October 2007)

This is the initial released revision of this document.

