第 5 章 数据 EEPROM

目录

本章包括下列主题：

5.1 简介

除了标准的基于闪存的程序存储器和易失性数据 RAM 之外，一些特定的 PIC24F 器件还包含了片上数据 EEPROM。该存储器块使用户可以在非易失性存储单元中存储程序或应用程序信息（如标识、校准常数等），并在需要时方便地重写信息。数据 EEPROM 存储器基于与程序存储器相同的闪存技术，并经过优化，可实现长期数据保存和更高的耐擦写次数。

数据 EEPROM 映射到用户程序存储空间的顶部，最高地址位于程序存储器地址 7FFFFFh 处。EEPROM 的大小取决于具体器件。更多信息，请参见具体器件的数据手册。

EEPROM 按 16 位宽存储器进行组织。每个字都可直接寻址；不同于程序存储器，在奇地址中并不存在 “虚拟字节”。

用于数据EEPROM的编程技术类似于在前面章节中所讨论的用于闪存程序存储器RTSP的技术。闪存和数据 EEPROM 编程操作之间的关键区别在于在每个编程 / 擦除周期中可以烧写或擦除的数据量，并且对数据 EEPROM 进行表写操作时，不会暂停 CPU 的操作。

5.2 控制寄存器

类似于闪存程序存储器，数据 EEPROM 的编程操作使用 3 个非易失性存储器 （Nonvolatile Memory， NVM）控制寄存器进行控制：

- NVMCON：非易失性存储器控制寄存器
- NVMKEY：非易失性存储器密钥寄存器
- NVMADR：非易失性存储器地址寄存器

5.2.1 NVMCON 寄存器

NVMCON 寄存器（寄存器 5-1）是数据 EEPROM 编程 / 擦除操作的主控制寄存器。高字节包含用于启动编程或擦除周期的控制位，以及用于指示操作是否成功执行的标志位。 NVMCOM 的低字节用于配置将执行的 NVM 操作的类型。

NVMCON 寄存器还控制程序存储器的编程 / 擦除操作，如**第 4 章 “程序存储器”**中所述。

5.2.2 NVMKEY 寄存器

NVMKEY 是一个只写寄存器，用于防止数据 EEPROM 存储单元的误写或误擦除。要开始任何编程或擦除序列，必须严格按顺序先执行以下两条指令：

1. 将 55h 写入 NVMKEY。
2. 将 AAh 写入 NVMKEY。

在此序列后，就可以在一个指令周期中写入 NVMCON 寄存器。在多数情况下，用户只需要将 NVMCON 寄存器中的 WR 位置 1 就可以开始编程或擦除周期。在解锁序列中应禁止中断。

MPLAB® C30 C 编译器提供了一个已定义的库过程 （`__builtin_write_NVM`）用来执行解锁序列。例 5-1 显示了如何使用行内汇编来执行解锁序列。

例 5-1： 数据 EEPROM 解锁序列

```
//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");
```


寄存器 5-1： NVMCON：非易失性存储器控制寄存器 （数据 EEPROM 操作）

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

图注：		U = 未实现位，读为 0	
R = 可读位	W = 可写位	S = 只可置 1 位	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15 **WR：**写 （编程或擦除）控制位
1 = 启动数据 EEPROM 的擦除或写周期 （用软件只能将该位置 1，但不能清零）
0 = 写周期完成 （由硬件自动清零）

bit 14 **WREN：**写 （擦除或编程）使能位
1 = 使能擦除或编程操作
0 = 不允许任何操作 （在写 / 擦除操作完成时器件会将该位清零）

bit 13 **WRERR：**闪存错误标志位
1 = 写操作过早终止 （编程操作期间的任何 $\overline{\text{MCLR}}$ 复位或 WDT 复位）
0 = 写操作成功完成

bit 12 **PGMONLY：**仅编程使能位
1 = 不先擦除目标地址内容即执行写操作
0 = 自动写前擦除：在执行写操作之前，自动对目标地址进行擦除操作

bit 11-7 **保留：**用户代码应将 0 写入这些存储单元

bit 6 **擦除：**擦除操作选择位
1 = WR 置 1 时执行擦除操作
0 = WR 置 1 时执行写操作

bit 5 **保留：**用户代码应将 0 写入这些存储单元

bit 4-0 **NVMOP4:NVMOP0：**编程操作命令字节位 (1)
擦除操作 （当 ERASE 位为 1 时）：
11010 = 擦除 8 个字
11001 = 擦除 4 个字
11000 = 擦除 1 个字
100xx = 擦除整个数据 EEPROM
编程操作 （当 ERASE 位为 0 时）：
001xx = 写 1 个字

注 1： 此处仅给出了代表有效的数据 EEPROM 操作的位组合。其他组合或未实现，或是用于闪存程序存储器或器件配置操作。更多信息，请参见具体器件的数据手册。

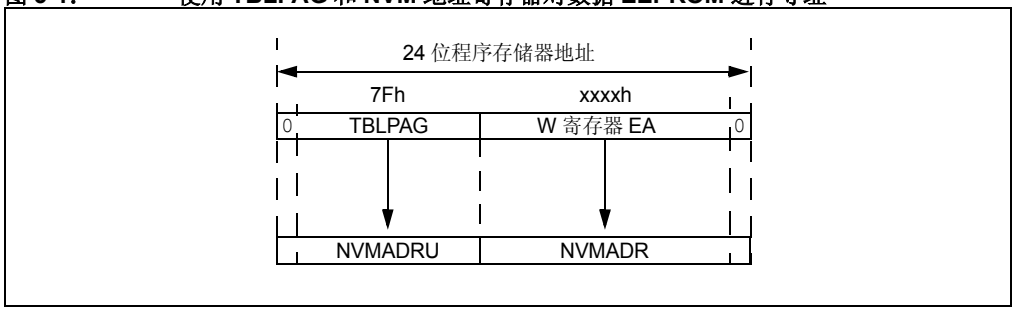
5.2.3 NVM 地址寄存器

类似于闪存程序存储器， NVM 地址寄存器 NVMADRU 和 NVMADR 构成进行数据 EEPROM 操作的选定行或字的 24 位有效地址 （Effective Address， EA）。 NVMADRU 寄存器用于保存 EA 的高 8 位，而 NVMADR 寄存器用于保存 EA 的低 16 位。这两个寄存器不会被映射到 SFR 空间中；相反，它们直接捕捉已执行的上一条表写指令的 EA<23:0>，并选择要擦除的数据 EEPROM 行。图 5-1 显示了用于编程和擦除操作的程序存储器 EA 是如何构成的。

类似于程序存储器操作， NVMADR 的最低有效位 （Least Significant bit， LSb）被限制到偶地址。这是因为数据EEPROM空间中的任何给定地址仅包含程序存储器宽度的低位字；高位字（包括最高位的 “虚拟字节”）是不可用的。这意味着数据 EEPROM 地址的 LSb 将始终为 0。在数据 EEPROM 操作中， NVMADR 值的允许范围由特定器件的 EEPROM 大小决定。

类似地， NVMADRU 的最高有效位 （Most Significant bit， MSb）始终为 0，因为所有地址都位于用户程序空间中。此外，对于数据 EEPROM 操作，TBLPAG 的值（对应于 NVMADRU 的值）还可以固定为 7Fh，因为数据 EEPROM 地址范围将始终位于程序存储空间的最高 64K 页。

图 5-1： 使用 TBLPAG 和 NVM 地址寄存器对数据 EEPROM 进行寻址



5.3 数据 EEPROM 操作

EEPROM 块使用类似于用于程序存储器的表读和表写操作进行访问。对于 EEPROM 操作，不需要 TBLWTH 和 TBLRDH 指令，因为存储器只有 16 位宽。数据 EEPROM 的编程和擦除过程类似于用于闪存程序存储器的过程，只是它们针对快速数据访问进行了优化。对于数据 EEPROM，可以执行以下编程操作：

- 写 1 个字
- 擦除 1、 4 或 8 个字
- 批量擦除整个数据 EEPROM

数据 EEPROM 在整个 VDD 工作范围内正常运行时是可读写的。不同于闪存程序存储器，在 EEPROM 编程或擦除操作期间，不会停止正常的程序执行。

数据 EEPROM 操作使用 NVMCON 和 NVMKEY 寄存器来执行。编程软件负责等待操作完成。软件可通过三种方法来检测 EEPROM 擦除或编程操作是否完成：

- 在软件中查询 WR 位 （NVMCON<15>）。操作完成时 WR 位将被清零。
- 在软件中查询 NVMIF 位 （IFS0<12>）。操作完成时 NVMIF 位将被置 1。
- 允许 NVM 中断。操作完成时 CPU 将会发生中断。进一步的编程操作可以在中断服务程序（Interrupt Service Routine， ISR）中处理。

注： 如果在编程或擦除操作正在进行时，用户尝试读 EEPROM，则会产生意外的结果。

C30 C 编译器包含了一些库过程，用于自动执行表读和表写操作，管理表指针和写缓冲区，以及解锁和启动存储器写序列。这使用户不必用 C 语言为每个应用程序创建汇编器宏或时间要求严格的程序。

在以下几节详细说明的代码示例中使用了这些库过程。对于未使用 C30 编译器库的用户，以下提供了每个过程的大致说明。

5.3.1 单字写入

对数据 EEPROM 进行写操作的总体算法如下：

1. 擦除数据 EEPROM 的一个字：
   - 设置 NVMOP 位以擦除 EEPROM 的一个字 （NVMCON<4:0> = 11000）。
   - 将要擦除的字的地址写入 TBLPAG 和 WREG 寄存器。
   - 清零 NVMIF 状态位并允许 NVM 中断 （可选）。
   - 向 NVMKEY 中写入密钥序列。
   - 将 WR 位置 1 以开始擦除周期。
   - 查询 WR 位，或者等待 NVM 中断。
2. 将数据字写入数据 EEPROM 锁存器。
3. 将数据字编程到 EEPROM 中：
   - 设置 NVMCON 寄存器以烧写 EEPROM 的一个字。
   - 清零 NVMIF 状态位并允许 NVM 中断 （可选）。
   - 向 NVMKEY 中写入密钥序列。
   - 将 WR 位置 1 以开始擦除周期。
   - 查询 WR 位，或者等待 NVM 中断。

表写指令用于将数据写入一个写锁存器。TBLPAG 寄存器中会装入 EEPROM 地址的高 8 位。当执行表写操作时， EEPROM 地址的低 16 位会被自动捕捉到 NVMADR 寄存器中。 NVMADR 寄存器的 LSb 对编程操作没有影响。 NVMCON 寄存器配置为烧写数据 EEPROM 的一个字。

将 WR 控制位 （NVMCON<15>）置 1 会启动编程操作。在将 WR 控制位置 1 之前，必须先向 NVMKEY 寄存器中写入解锁序列。解锁序列需要严格按照所给出的顺序执行，不能有中断 （详情请参见**第 5.2.2 节 “NVMKEY 寄存器”**）。因此，在写入序列之前，应先禁止中断。

例 5-2 给出了典型的写序列，包括擦除和密钥解锁序列。该示例使用了一些 C30 编译器库过程来管理表指针（builtin_tblpage 和 builtin_tbloffset）、解锁序列（builtin_write_NVM）和实际的数据写操作 （builtin_tblwtl）。存储器解锁序列还会将 WR 位置 1 以启动操作，并在完成时返回控制。

例 5-2： 写数据 EEPROM 的单个字

```
// 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 单字擦除

在单字擦除操作中，NVMADRU:NVMADR 寄存器从 TBLPAG 和 WREG 寄存器中装入要擦除的数据 EEPROM 地址。因为访问的是 EEPROM 的一个字，所以 NVMADR 的 LSb 对擦除操作没有影响。 NVMCON 寄存器必须配置为擦除 EEPROM 存储器的一个字。

将 WR 控制位 （NVMCON<15>）置 1 会启动擦除操作。在将 WR 控制位置 1 之前，必须先向 NVMKEY 寄存器中写入解锁或密钥序列。解锁序列需要严格按照所给出的顺序执行，不能有中断（详情请参见**第 5.2.2 节 “NVMKEY 寄存器”**）。因此，在写入序列之前，应先禁止中断。

例 5-3 中给出了典型的擦除序列。该示例及随后的其他擦除示例使用了一些 C 库过程来管理表指针（builtin_tblpage 和 builtin_tbloffset）和擦除页指针（builtin_tblwtl）。存储器解锁序列（builtin_write_NVM）还会将 WR 位置 1 以启动操作，并在完成时返回控制。

例 5-3： 单字擦除

```
// 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 4 字擦除

NVMCON 寄存器配置为擦除 EEPROM 存储器的 4 个字 （或半行）。必须首先初始化指向要擦除的数据 EEPROM 地址的指针。数据 EEPROM 的擦除操作必须在偶地址边界处进行。因此，有效地址的低 3 位对要擦除的存储器没有影响。

将 WR 控制位 （NVMCON<15>）置 1 会启动擦除操作。在将 WR 控制位置 1 之前，应先向 NVMKEY 寄存器中写入解锁序列。解锁序列需要严格按照所给出的顺序执行，不能有中断（详情请参见**第 5.2.2 节 “NVMKEY 寄存器”**）。因此，在写入序列之前，应先禁止中断。

例 5-4： 4 字擦除序列

```
// 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 8 字擦除

NVMCON 寄存器配置为擦除 EEPROM 存储器的一行。 NVMADRU 和 NVMADR 寄存器必须指向要擦除的行。数据 EEPROM 的擦除操作必须在偶地址边界处进行。因此，地址的低 5 位对要擦除的行没有影响。

将 WR 控制位 （NVMCON<15>）置 1 会启动擦除操作。在将 WR 控制位置 1 之前，必须先向 NVMKEY 寄存器中写入解锁序列。解锁序列需要严格按照所给出的顺序执行，不能有中断（详情请参见**第 5.2.2 节 “NVMKEY 寄存器”**）。因此，在写入序列之前，应先禁止中断。

例 5-5： 8 字擦除

```
// 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 数据 EEPROM 批量擦除

NVMCON 寄存器配置为批量擦除整个数据 EEPROM 存储器。因为该操作会影响整个数据 EEPROM，所以不需要配置地址寄存器。

将 WR 控制位 （NVMCON<15>）置 1 会启动擦除操作。在将 WR 控制位置 1 之前，必须先向 NVMKEY 寄存器中写入解锁序列。解锁序列需要严格按照所给出的顺序执行，不能有中断（详情请参见**第 5.2.2 节 “NVMKEY 寄存器”**）。因此，在写入序列之前，应先禁止中断。

在以下批量擦除示例 （例 5-6）中，执行解锁库过程会自动触发擦除过程，因为它还会将 WR 位置 1。

例 5-6： 数据 EEPROM 批量擦除

```
// 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 读取数据 EEPROM

类似于程序存储器操作，表读指令用于从数据 EEPROM 中读取数据。因为 EEPROM 阵列只有 16 位宽，所以只需要 TBLRDL 指令。在例 5-7 中，使用 W0 作为指向数据 EEPROM 地址的指针。结果存放在寄存器 W4 中。

程序空间可视性（Program Space Visibility，PSV）也可以用于读取程序存储器地址空间中的单元。关于 PSV 的更多信息，请参见**第 4 章 “程序存储器”**。

数据EEPROM读操作示例（例5-7）使用了来自C30编译器库的表指针管理（builtin_tblpage 和 builtin_tbloffset）和表读（builtin_tblrdl）过程。

例 5-7： 使用 TBLRD 命令读取数据 EEPROM

```
// 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 寄存器映射

表 5-1 中提供了与 PIC24F 数据 EEPROM 相关的特殊功能寄存器汇总。

表 5-1： 与数据 EEPROM 操作相关的寄存器

寄存器名称	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	所有复位时的状态
TBLPAG	—	—	—	—	—	—	—	—	表存储器页地址寄存器								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

图注： — = 未实现，读为 0； r = 保留，用户代码应将 0 写入这些存储单元

注 1： 所示复位值仅适用于 POR。其他复位状态下的值取决于复位时存储器写操作或擦除操作的状态。

5.5 设计技巧

问 1： *我无法正确对数据 EEPROM 进行编程或擦除操作。我的代码是正确的。会是什么原因呢？*

答：在启动编程或擦除周期时应禁止中断，以确保执行密钥序列时不会发生中断。可以通过使用 DISI 指令，或通过将当前 CPU 优先级升高到 7 来禁止中断。

本章中给出的代码示例使用 DISI 指令来将中断禁止一个指定的指令周期数。临时禁止中断的一种替代方法是，在堆栈中保存当前的 SR 寄存器值，然后将值 00E0h 与 SR 进行逻辑 “或”运算，以强制 IPL<2:0> = 111。

问 2： *是否有可以不使用表指令而读取数据 EEPROM 的简便方式？*

答：数据 EEPROM 被映射到程序存储空间中。可以使用 PSV 来将 EEPROM 区域映射到数据存储空间中。关于 PSV 的更多信息，请参见**第 4 章 “程序存储器”**。

5.6 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为 PIC24F 器件系列而编写的，但其概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。当前与数据 EEPROM 相关的应用笔记有：

标题	应用笔记编号
Emulating Data EEPROM for PIC18 and PIC24 Microcontrollers and dsPIC® Digital Signal Controllers	AN1095

注： 如需获取更多 PIC24F 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

5.7 版本历史

版本 A （2007 年 10 月）

这是本文档的初始版本。

