# AN1095

# 仿真 PIC18 和 PIC24 单片机及 dsPIC® 数字信号控制器的数据 EEPROM

*作者：* *David Otten、Stephen Cowden、Pradeep Budagutta、 Priyabrata Sinha 和 Harish Agari*
*Microchip Technology Inc.*

## 简介

Microchip 对其产品线进行了扩展，其中包括大量具有良好性价比但不具有内部数据 EEPROM 的 PIC® 单片机 （MCU）。

许多应用使用表读和表写操作将非易失性信息存储到闪存程序存储器中。然而，那些需要频繁更新这些数据的应用场合对 MCU/ 数字信号控制器 （Digital Signal Controller，DSC）器件耐用性的要求可能比闪存存储器更高。

有一种解决方法是选择使用外部串行 EEPROM。不过这对于那些成本敏感或受引脚限制的应用来说可能也并不适用。

该应用笔记介绍用于解决上述问题的第三种选择。这种算法的特点是采用一个类似于内部数据 EEPROM 的接口，最高可将现有程序存储器的耐用性提升为原来的 500 倍。

**注：** 使用这一解决方案时，器件必须具有字写入或双字写入功能。有关验证是否支持此特性的信息，请参见具体器件的数据手册。

## 术语定义

页 – 擦除操作影响的最小程序存储容量。

行 – 编程操作影响的最大程序存储容量。

擦 / 写周期 – 程序存储器可被擦除和写入的次数。

耐用性 – 表明最大擦 / 写次数和相关条件的规范参数。

数据保持时间 – 表明数据能够保存在闪存程序存储器中的最短时间及相关条件的规范参数。

有效耐用性 – 采用高效编程算法提升后的仿真数据 EEPROM 的耐用性。

当前 （激活）页 – 数据 EEPROM 仿真算法正在进行读写操作的程序存储器页。

合并页 – 合并程序执行完后新的当前页。

页状态 – 位于存储数据 EEPROM 仿真状态的当前页开头处的程序存储器地址。PIC18器件使用两个地址而 PIC24/dsPIC33 器件使用一个地址。

## 工作原理

本应用笔记中使用的算法可支持对总数最多为 255 个地址倍数的多个数据 EEPROM 进行仿真，且地址是可选的，使用单个地址空间，范围为 0 至仿真数据 EEPROM 总数减 1 （见下注）。

例如：如果应用共需要 260 个 EEPROM 地址，则需要多个 EEPROM 存储区。用户可以配置为使用 2 个 EEPROM 存储区，每个存储区 130 个 EEPROM 数据。可用的总 EEPROM 地址范围为 0 至 259。

**注：** PIC18 和 PIC24/dsPIC33 实现支持多个 EEPROM 存储区。每个 EEPROM 最多可以有 255 个地址。因此，地址总数为 0 至 N x 255 -1，其中 N = EEPROM 存储区数量。

PIC18 实现支持 8 位数据和多个 EEPROM 存储区；PIC24/dsPIC33 实现支持 16 位数据和多个 EEPROM 存储区。由于程序存储器的结构不同，因此 8 位数据和 16 位数据实现方式具有不同的仿真数据 EEPROM 信息存储方式。有关这些格式的信息，可分别参见表 1 和表 2。

**表 1： 程序存储器中 PIC18 数据 EEPROM 采用的信息格式**

| Bit 15-8 | Bit 7-0 |
|---|---|
| 数据 EE 的数据 | 数据 EE 的地址 |

**表 2： 程序存储器中 PIC24/dsPIC33 数据 EEPROM 采用的信息格式**

| Bit 23-16 | Bit 15-0 |
|---|---|
| 数据 EE 的地址 | 数据 EE 的数据 |

该算法利用了 PIC MCU 具有的对程序存储器中的一个单元进行自编程的能力。对于 PIC18 来说，对于这一地址执行的操作是一个 8 位操作；而对于 PIC24/dsPIC33 来说，则由正在写入的地址是奇数还是偶数地址来决定其是 8 位操作还是 16 位操作。

## MPLAB® 代码配置器 16 位数据 EEPROM 仿真支持

基于本应用笔记中的 PIC24/dsPIC 固件实现，Microchip 增加了 MPLAB 代码配置器 16 位数据 EEPROM 模块（MCC 16位 DEE）支持。这有助于简化固件配置和设置。关于下载位置，请参见 **“ 软件 ”** 部分。

MCC 16 位 DEE 版本与最初的应用笔记实现之间的一些明显差异：

**文件名：**

`dee.c`（相当于本文档中引用的 “`DEE Emulation 16-bit.c`”）。

`dee.h`（相当于本文档中引用的 “`DEE Emulation 16-bit.h`”）。

**功能名：**

`DEE_Init`、`DEE_Read` 和 `DEE_Write`（相当于本文档中引用的 `DataEEInit`、`DataEERead` 和 `DataEEWrite`）。

**注 1：** 有关程序存储器构成的更多信息，可参见具体器件的数据手册。

**2：** 在 dsPIC33 和 PIC24 器件中，EEPROM 仿真算法提供了使用辅助闪存程序存储器代替主闪存程序存储器的选项。用户可以通过取消注释掉头文件 “`DEE Emulation 16-bit.h`” 中的以下行来选择该选项：
```
#define __AUXFLASH 1
```

**3：** 在包含闪存错误校正编码（Error Correction Coding， ECC）功能的所有 dsPIC33 和 PIC24 器件中，用户必须取消注释掉头文件 “`DEE Emulation 16-bit.h`” 中的以下行：
```
#define __HAS_ECC
```
在所有这些器件中，数据EEPROM仿真软件不会使用每个双字对中的第2个指令字。例如，如果闪存地址0x2004用于数据EEPROM仿真，则不会使用闪存地址0x2006。

在MCC 16位DEE版本中自动处理此更改。

**4：** PIC24/dsPIC33 实现采用 PSV 存储器映射，因此支持的 EEPROM 地址总数有限。为了提高总有效耐用性，可能需要减少 EEPROM 地址的总数。

## PIC24/dsPIC33 案例

为了便于了解算法的工作原理，本部分介绍了 PIC24/dsPIC33 器件的简单案例。

在每个 EEPROM 存储区的第一页初始化后，首个地址预留给页状态信息。该信息表明某一个页是处于激活状态还是已超出限制，以及已对其完成了多少次擦 / 写操作。用户不能直接对这一信息进行访问，但算法使用这一信息来寻找可用页以及对状态标志进行更新。初始化后，首页被指定为激活页。

在该例中，单个写操作执行后，数据 0x0202 被存储到数据 EEPROM 的地址 0x2 中。如表 3 所示，这一数据信息被存储在页内第一个可用的地址单元中。随着越来越多的写操作被执行，算法继续以类似的方式写入信息，如表 4 至表 6 所示。

在该例中，数据 EEPROM 地址 0x7 被写入 0x0707，0x2 被更新为 0x2222，地址 0xA 被写入 0x0A0A。

在表 7 中，将 0x7777 写入地址 0x7 改写页的最后一个地址单元。由于当前激活页已满，数据 EEPROM 信息将移至下一可用页。这一新页称为合并页。合并程序执行这项任务。对每一个数据 EEPROM 地址来说，由于仅仅需要最新数据，所以写入的信息量减小。

数据被移走后，这一页被指定为当前页。如果当前页已递增超过程序存储器中所有已分配页，那么擦 / 写次数会相应递增，如表 8 所示。现在可以通过写操作将更多信息存储到存储器页中。

PIC18 算法的工作方式与上述算法类似，但它并不为页状态信息预留一个程序存储器地址单元，而是预留两个。而且，它存储的是 8 位数据，而非 16 位。

**表 3：写数据 EEPROM（ 0x0202， 2）**

| 页地址 | 数据 EE 的地址 | 数据 EE 的数据 |
|---|---|---|
| 页 + 0 | 页状态 [23:16] | 0x0000 |
| 页 + 2 | **2** | **0x0202** |
| 页 + 4 | 0xFF | 0xFFFF |
| 页 + 6 | 0xFF | 0xFFFF |
| 页 + 8 | 0xFF | 0xFFFF |
| | • • • | |
| 页 + 1022 | 0xFF | 0xFFFF |

**表 4：写数据 EEPROM（ 0x0707， 7）**

| 页地址 | 数据 EE 的地址 | 数据 EE 的数据 |
|---|---|---|
| 页 + 0 | 页状态 [23:16] | 0x0001 |
| 页 + 2 | **2** | 0x0202 |
| 页 + 4 | **7** | **0x0707** |
| 页 + 6 | 0xFF | 0xFFFF |
| 页 + 8 | 0xFF | 0xFFFF |
| | • • • | |
| 页 + 1022 | 0xFF | 0xFFFF |

**表 5：写数据 EEPROM（ 0x2222， 2）**

| 页地址 | 数据 EE 的地址 | 数据 EE 的数据 |
|---|---|---|
| 页 + 0 | 页状态 [23:16] | 0x0000 |
| 页 + 2 | 2 | 0x0202 |
| 页 + 4 | 7 | 0x0707 |
| 页 + 6 | **2** | **0x2222** |
| 页 + 8 | 0xFF | 0xFFFF |
| | • • • | |
| 页 + 1022 | 0xFF | 0xFFFF |

**表 6：写数据 EEPROM（ 0x0A0A， 0xA）**

| 页地址 | 数据 EE 的地址 | 数据 EE 的数据 |
|---|---|---|
| 页 + 0 | 页状态 [23:16] | 0x0000 |
| 页 + 2 | 2 | 0x0202 |
| 页 + 4 | 7 | 0x0707 |
| 页 + 6 | 2 | 0x2222 |
| 页 + 8 | **0xA** | **0x0A0A** |
| | • • • | |
| 页 + 1022 | 0xFF | 0xFFFF |

**表 7：写数据 EEPROM（ 0x7777， 7）**

| 页地址 | 数据 EE 的地址 | 数据 EE 的数据 |
|---|---|---|
| 页 + 0 | 页状态 [23:16] | 0x0000 |
| 页 + 2 | 2 | 0x0202 |
| 页 + 4 | 7 | 0x0707 |
| 页 + 6 | 2 | 0x2222 |
| 页 + 8 | 0xA | 0x0A0A |
| | • • • | |
| 页 + 1022 | **7** | **0x7777** |

**表 8：合并操作后的页**

| 页地址 | 数据 EE 的地址 | 数据 EE 的数据 |
|---|---|---|
| 页 + 0 | 页状态 [23:16] | 0x0001 |
| 页 + 4 | **2** | **0x2222** |
| 页 + 6 | **7** | **0x7777** |
| 页 + 2 | **0xA** | **0x0A0A** |
| 页 + 8 | **0xFF** | **0xFFFF** |
| | • • • | |
| 页 + 1022 | **0xFF** | **0xFFFF** |

由于在擦除页之前页内每个地址单元只被写入了一次，因而对于页来说只耗费了一个擦 / 写周期。因此，该算法成倍地提高了仿真数据 EEPROM 的有效耐用性。

只有当最新信息被写入至下一个可用页并被成功校验后，以前填充的页才可被擦除。经过这一过程，信息总是被存储到非易失性存储器，从而使意外断电造成的损失降到最小。

由于程序存储器页的填充是按从头到尾顺序进行的，因此该算法假定最新的数据 EEPROM 信息处在最接近页末的位置。为简化读操作，当寻找指定数据 EEPROM 地址时，搜索操作是从当前程序存储器页的末尾向页首进行的。

当找到与给定地址匹配的第一个地址时，会返回其中的数据。如果地址无法找到，那么返回值为全 “1”—— 即 0xFF 或 0xFFFF—— 仿真对独立数据 EEPROM 中未写入地址单元执行操作的结果。

## 状态标志

状态标志用来指示仿真过程中是否有错误或警告事件发生。通过数据 EEPROM 标志寄存器可对这些指示标志进行访问；所有标志皆为高有效。

| **注：** | 所有 EEPROM 存储区都会影响相同的状态标志。 |
|---|---|

这些状态位和返回值定义如下：

- addrNotFound （0xFF/0xFFFF） – 对以前未写入的数据 EEPROM 地址单元进行读操作。
- expiredPage（0x1）– 程序存储器的擦 / 写周期计数值超出用户定义的限制。该算法将试图执行写操作。
- packBeforePageFull （0x2） – 在激活页未满之前调用合并子程序。合并子程序试图将最新数据 EEPROM 信息移至合并页，即使激活页未满。
- packBeforeInit（0x3）– 合并子程序在初始化程序执行之前被执行。合并操作被中止。
- packSkipped（0x4）– 对页的写入操作超出了页边界。这可能是由于合并程序没有正确执行造成的。合并操作被中止。
- illegalAddress （0x5） – 试图对地址大小等于或大于数据 EEPROM 容量的地址单元进行写 / 读操作。读 / 写操作被中止。
- pageCorrupt （0x6） – 页状态信息被破坏。当前操作被中止。
- writeError （0x7） – 写入程序存储器中的数据未能通过校验。当前操作被中止。

不同状态标志在严重程度和如何对其处理方面是不同的。信息性状态标志可能在正常处理过程中出现，仅需通过相关宏清除该标志即可对其进行处理。这些标志包括：addrNotFound、packBeforePageFull 和 illegalAddress 标志。

警告状态标志指示相应限定条件已被超出，但操作仍在继续的情况。这包括 expiredPage 状态标志。尽管这一标志置 1，算法仍将试图处理读写请求，但在每一操作后该标志都将置 1。

最严重的标志是系统错误状态标志。这表明数据 EEPROM 中的数据完整性受损或在非法条件被解除之前算法不能继续执行。这些状态标志包括 packBeforeInit、 pageCorrupt 和 writeError。

为避免 packBeforeInit 事件发生，应在执行所有其他仿真程序之前启动初始化程序`DataEEInit`或`DEE_Init`（MCC 16 位 DEE）。由于该程序对仿真过程的当前状态进行访问，因此只有当需要时它才起作用。因此，在数据 EEPROM 的仿真过程中，它随时都能被调用。

pageCorrupt 和 writeError 标志指的是写操作未能通过校验，当前操作被中止。如果这种情况发生，数据 EEPROM 信息的完整性就会受到损害。此时不应再继续进一步的仿真操作。惟一解决办法是擦除程序存储器中预留给数据 EEPROM 仿真的所有页，并对其进行初始化。

通过宏可获取和清除状态标志值。状态标志只能由用户来清除。标志值不会影响操作，但它能指示操作是否顺利完成。

**例1： 宏命名规则示例**

```
宏：“Getx”“Setx y”
x = 标志名称
y = 赋予标志的值
```

使用 8 位字符 `dataEEFlags.val` 可通过一次操作读或清除所有标志。

## 页状态

每一个程序存储器页都会为页状态预留空间 —— 对于 PIC18 单片机，应预留头两个字的地址空间，而对于 PIC24/dsPIC33 实现则预留头一个字的地址空间。页状态包含了页的信息，页是已超过限制还是处于激活状态以及对其执行擦 / 写的次数。

**注：** 带有自举程序的应用不要修改用于数据 EEPROM 仿真的任何页。

这些状态值不能直接由用户使用，而是通过算法来监测和控制页信息。有关 PIC24/dsPIC33 和 PIC18 页状态信息的格式，请参见寄存器 1、寄存器 2 和寄存器 3。

**寄存器1： PIC24/dsPIC33算法的页状态**

| U-1 | U-1 | U-1 | R-1 | R-1 | R-1 | U-1 | U-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | Page Expired | Page Current | Page Available | — | — |
| bit 23 | | | | | | | bit 16 |

| R-1 | R-1 | R-1 | R-1 | R-1 | R-1 | R-1 | R-1 |
|---|---|---|---|---|---|---|---|
| Page Erase/Write Count | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R-1 | R-1 | R-1 | R-1 | R-1 | R-1 | R-1 | R-1 |
|---|---|---|---|---|---|---|---|
| Page Erase/Write Count | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| **图注：** | | |
|---|---|---|
| R = 保留位 | U = 未用位，读为 “1” | |
| -n = 初始化之前的值 | 1 = 该位处于擦除状态 | 0 = 该位处于编程状态 |

bit 23-21 **未用：**读为“1”

bit 20 **Page Expired**
1 = 页未超出限制
0 = 页已超出限制

bit 19 **Page Current**
1 = 非当前页
0 = 当前页

bit 18 **Page Available**
1 = 页可用
0 = 页不可用

bit 17-16 **未用：**读为“1”

bit 15-0 **Page Erase/Write Count：**页擦 / 写次数

**寄存器2：　　PIC18算法的页状态（页起始）**

| U-1 | U-1 | U-1 | U-1 | U-1 | U-1 | U-1 | U-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | — | — | — |
| bit 15 | | | | | | | bit 8 |

| U-1 | U-1 | U-1 | U-1 | U-1 | R-1 | R-1 | R-1 |
|---|---|---|---|---|---|---|---|
| — | — | — | — | — | Page Expired | Page Current | Page Available |
| bit 7 | | | | | | | bit 0 |

| 图注： | | |
|---|---|---|
| R = 保留位 | U = 未用位，读为 “1” | |
| -n = 初始化之前的值 | 1 = 该位处于擦除状态 | 0 = 该位处于编程状态 |

bit 15-3　**未实现：**读为“1”

bit 2　**Page Expired**
1 = 页未超出限制
0 = 页已超出限制

bit 1　**Page Current**
1 = 非当前页
0 = 当前页

bit 0　**Page Available**
1 = 页可用
0 = 页不可用

**寄存器3：　　PIC18算法的页状态（页起始 + 2）**

| R-1 | R-1 | R-1 | R-1 | R-1 | R-1 | R-1 | R-1 |
|---|---|---|---|---|---|---|---|
| Page Erase/Write Count | | | | | | | |
| bit 15 | | | | | | | bit 8 |

| R-1 | R-1 | R-1 | R-1 | R-1 | R-1 | R-1 | R-1 |
|---|---|---|---|---|---|---|---|
| Page Erase/Write Count | | | | | | | |
| bit 7 | | | | | | | bit 0 |

| 图注： | | |
|---|---|---|
| R = 保留位 | U = 未用位，读为 “1” | |
| -n = 初始化之前的值 | 1 = 该位处于擦除状态 | 0 = 该位处于编程状态 |

bit 15-0　**Page Erase/Write Count：**页擦 / 写次数

## 初始化操作

必须在其他所有的数据 EEPROM 操作发生之前调用初始化程序 `DataEEInit` 或 `DEE_Init`（MCC 16 位 DEE）；这可初始化所有 EEPROM 存储区。如果初始化程序确认程序存储器还没有进行仿真初始化，它就会找到程序存储器中第一个分配页，并对其状态信息进行初始化。随后，在需要时调用读写功能。

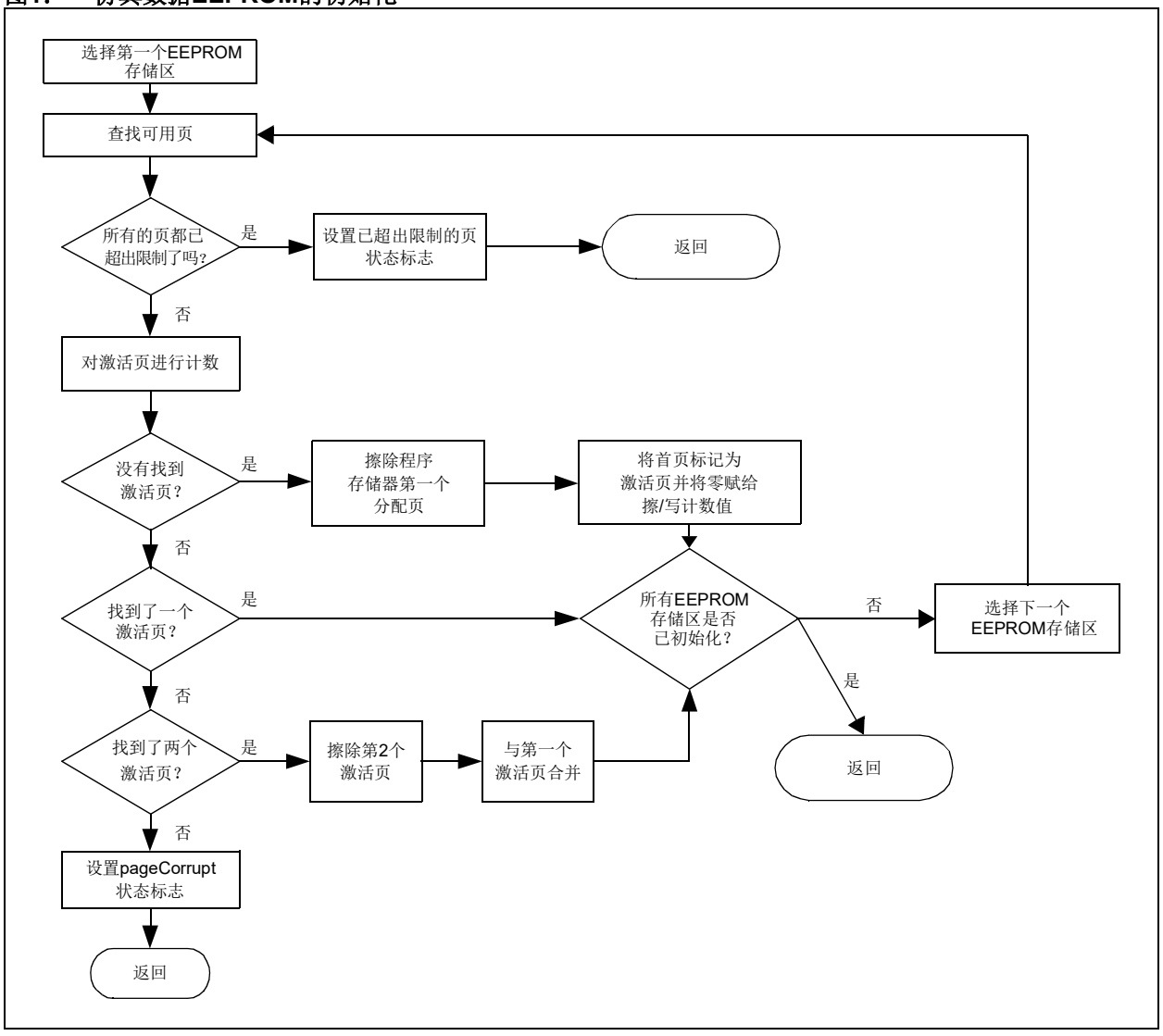
初始化程序也能确定数据 EEPROM 仿真是否已经开始进行。如果是，那么下列三种情形之一可能发生：

- 如果只发现了一个激活页，初始化程序就认为发生了一次复位。这时将不执行任何操作，程序正常退出。复位过程中所有已处于激活状态的读 / 写操作都应在复位完成后重新执行。
- 如果发现了两个当前页，初始化子程序则认为是在合并过程中出现了一次意外复位。子程序将擦除第二个激活页并调用合并程序重新更新直至结束。
- 如果初始化程序发现激活页超过两个，则认为程序存储器已受到应用程序代码的破坏，同时将 pageCorrupt 标志置 1。

对页状态位、RCON 和 NVMCON 寄存器（PIC24/dsPIC33）或 EECON1 寄存器（PIC18）的状态进行监控是重要的。这样，应用程序会对复位和电源电压变化作出正确的响应。

初始化程序的流程图如图 1 所示。

**图1： 仿真数据EEPROM的初始化**



## 读操作

DataEERead 或 DEE_Read（MCC 16 位 DEE）函数用于获取数据EEPROM的信息。它将返回与数据EEPROM地址有关的数据。如果指定的地址等于或大于指定数据EEPROM 的容量，则将设置 illegalAddress 标志并返回全 “1”。这个返回值模拟了专用数据 EEPROM 的反应，即未写入地址返回一擦除值。

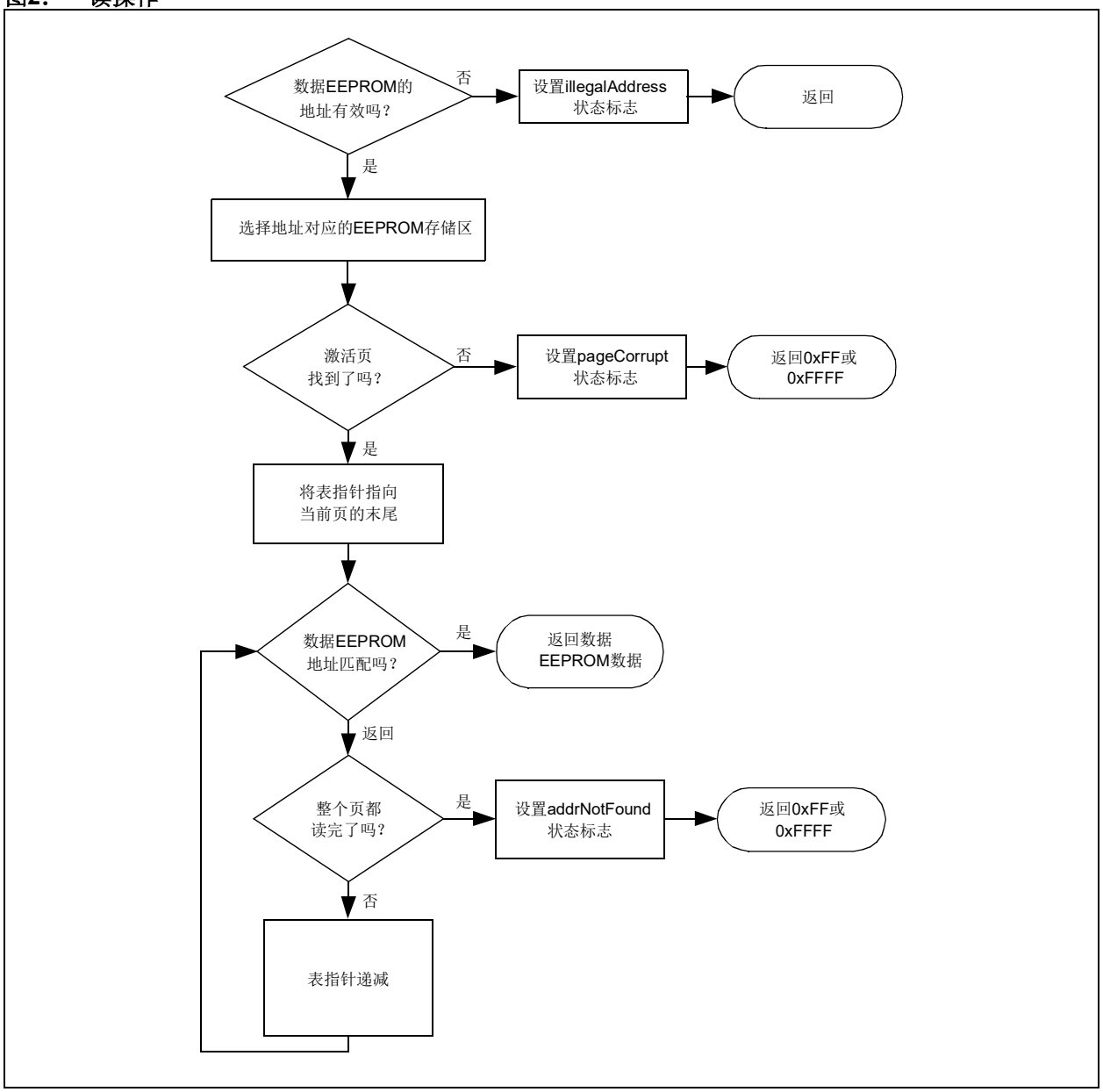
随后，读操作程序开始搜索与该地址对应的 EEPROM 存储区激活页。一旦搜索到，程序就从页尾开始在激活页中寻找匹配的地址。有关数据 EEPROM 信息在程序存储器中是如何存放的详细信息，请参见表 1 和表 2。

由于页是顺序填充的，所以当逆向搜索时，首先找到的是最新的 EEPROM 信息。一旦被搜索到，程序就会返回与数据 EEPROM 地址相关的数据 EEPROM 数据值。

如果没有找到激活页，则设置 pageCorrupt 标志。

读操作的流程图如图 2 所示。

**图2： 读操作**



## 写操作

应用程序使用 DataEEWrite 或 DEE_Write（MCC 16 位 DEE）函数对被仿真的数据 EEPROM 进行写操作。像读函数一样，它也校验数据 EEPROM 地址是否在 0 和仿真数据 EEPROM 的容量减一之间。如果所提供的地址未实现，则将设置 illegalAddress 标志而写操作将被中止。

随后，写操作程序开始搜索与该地址对应的 EEPROM 存储区激活页。找到激活页后，开始执行读操作。为了使擦 / 写次数最小，数值只有在改变时才被重新写入。

如果没有找到激活页，就设置 pageCorrupt 标志并返回一个非零值。

对激活页的正向搜索将返回到下一个可用地址的偏移量。如果下一个可用地址等于或大于页的上一个地址，那么程序就会设置 packSkipped 标志且写操作将被中止。否则，数据 EEPROM 的信息就会被写到页的下一个可用地址。

如果信息没有通过校验，将会设置 writeError 标志并返回一个非零值。用户可以尝试重写数据或根据需要对其作出响应。

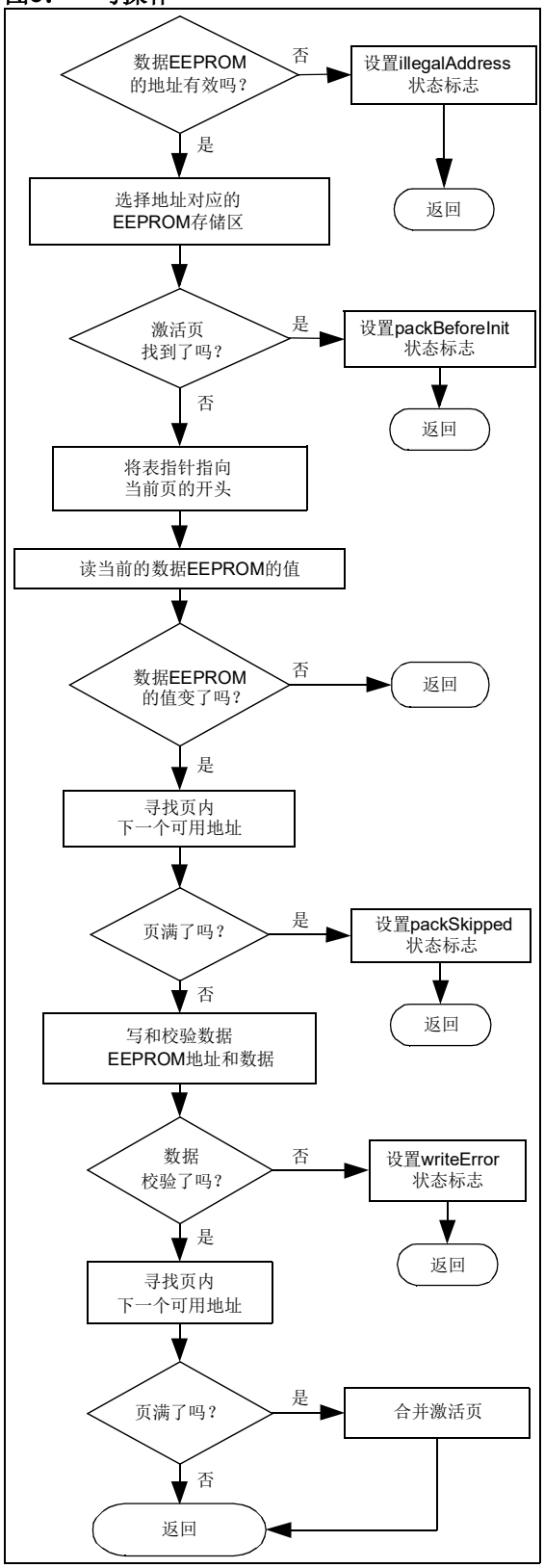
算法会在激活页为下一次写操作保留至少一个可用的地址单元。在对写操作进行成功校验后，如果没有保留可用地址单元，那么将调用合并程序。

当写程序成功地执行完后，将返回一个零值。

写操作的流程图如图 3 所示。

> **注：** 在包含闪存错误校正编码（ECC）功能的所有 PIC24/dsPIC33 器件，每个双字对中的第 2 个指令字不会被 DataEEWrite 或 DEE_Write（MCC 16 位 DEE）函数修改（但会在闪存页被擦除时被擦除）。

**图3： 写操作**



## 合并操作

合并程序 PackEE 或 DEE_Pack（MCC 16 位 DEE，非公共函数）既可在当前页被填满后由 DataEEWrite 或 DEE_Write（MCC 16 位 DEE）调用，也可由初始化程序 DataEEInit 或 DEE_Init（MCC 16 位 DEE）调用，以对用于数据 EEPROM 仿真的程序存储器进行初始化。它还可以由用户调用，这对时间敏感的应用非常有利。由于该子程序执行了多种导致 CPU 暂停的闪存操作，因此可在应用较方便的时间进行执行。这样做的缺点是使用了无需写入的程序存储单元从而导致有效耐用性降低。

函数首先将读取程序存储器中每个已分配页的 Page Current 状态位，已帮助仿真过程查找已填充的页。如果未找到已填充的页，则合并程序认为合并函数是在初始化程序存储器之前被调用的。随后，将设置 packBeforeInit 标志且合并操作将被中止。

最新的数据 EEPROM 信息需要使用一个新页来写入。这一页称为合并页。通常总是将程序存储器内的下一页指定为合并页。如果所有的可用页都已达到了用户指定的擦 / 写限制，将设置 expiredPage 标志，程序将继续进行合并操作。

只有当合并页变为分配给数据 EEPROM 存储器的首页时，擦 / 写计数器才会加一。从这一点上来说，每一页都具有相同数目的擦 / 写周期。如果擦 / 写计数器超过了规定的限制，则会通过设定页状态寄存器中的 Page Expired 位将页状态标记为已超出限制。

由于程序存储器的页数和擦 / 写周期限定值都是在编译时定义的，因而所有页将会按顺序依次超出限制。通过读函数，在激活页中对每一个已定义的数据 EEPROM 地址进行搜索。将要写入的信息写入程序存储器的写锁存器。在写锁存器中的一行被填满后，将进行数据写操作直到所有信息都被存储到合并数组中。如果写锁存器的最后一行没有被填满，则在编程之前将该锁存器的剩余位统统写为“1”。

如果激活页没有满，则认为用户调用了该程序。于是，将设置packBeforePageFull 状态标志，子程序将继续进行编程操作。

在所有数据被写入到合并页之后，当前页数据将被读取并与合并页数据进行比较。如果不匹配，则将设置 writeError 标志并在函数退出时返回错误代码。页状态信息也将被写入并进行校验。如果校验成功，激活页将被擦除，合并页将被指定为新的激活页。

合并函数返回零值表明合并子程序正常结束。

合并操作的流程图如图 4 所示。

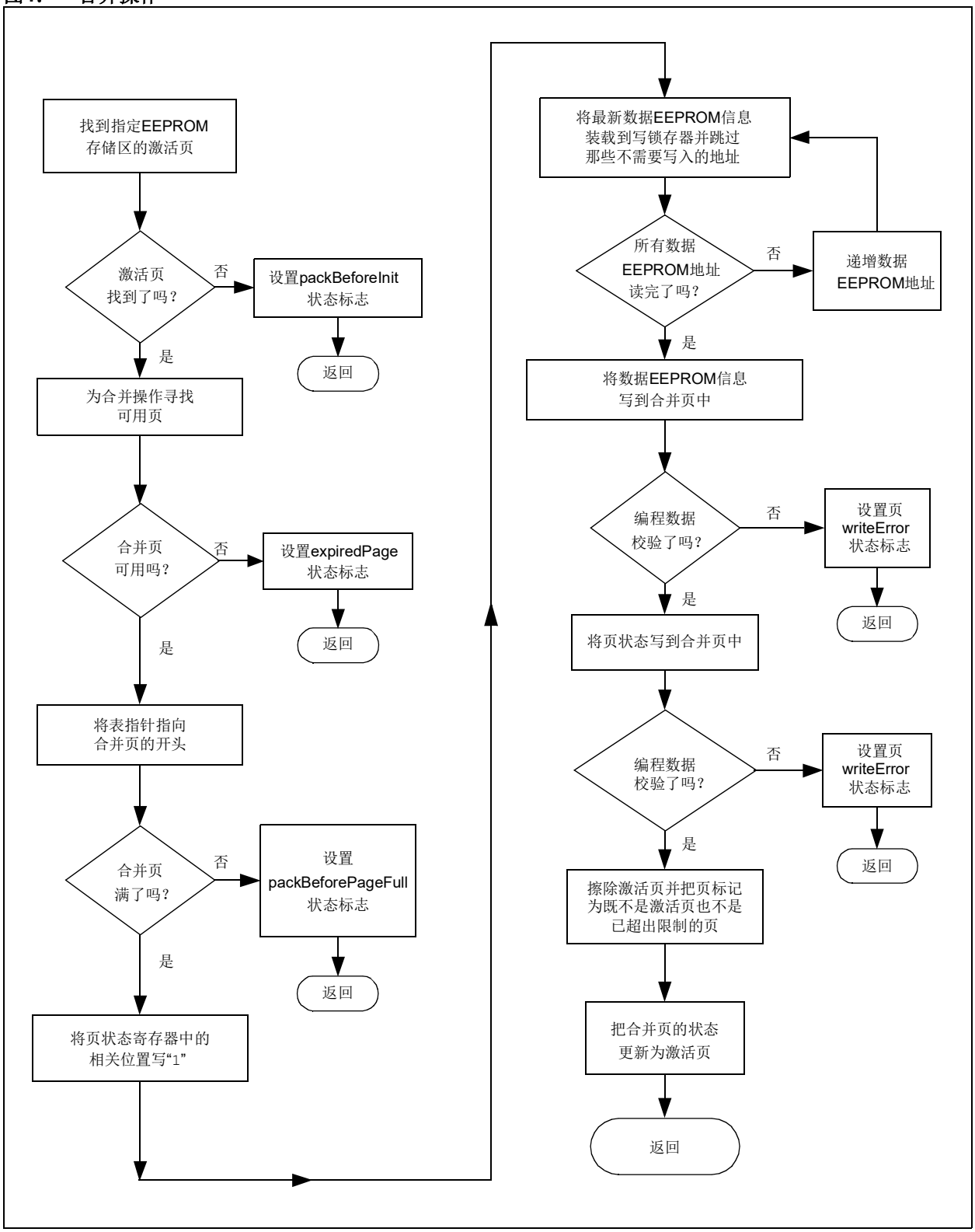
**注 1：** 数据 EEPROM 的最大容量不得超过 N x 255，其中 N = EEPROM 存储区数量。

**2：** 在 PIC24/dsPIC33 器件中，双字写操作用于编程合并页。在 PIC24/dsPIC33 器件中，行写操作用于编程合并页。

**3：** 对于 PIC24/dsPIC33，选择 255 作为每个存储区数据 EEPROM 的最大容量适用于大多数器件。

对于没有ECC且每个闪存页容量为256个或更少指令字的器件，以及有ECC且每个闪存页容量为512个或更少指令字的器件，使用的每个存储区数据EEPROM的最大容量应小于255，以便在最坏情况下DEE合并操作后至少有一半页可用（即128）。

**图4： 合并操作**


找到指定EEPROM
存储区的激活页
激活页
找到了吗？
否
设置packBeforeInit
状态标志
返回
是
为合并操作寻找
可用页
合并页
可用吗？
否
设置expiredPage
状态标志
返回
是
将表指针指向
合并页的开头
合并页
满了吗？
否
设置
packBeforePageFull
状态标志
返回
是
将页状态寄存器中的
相关位置写“1”
将最新数据EEPROM信息
装载到写锁存器并跳过
那些不需要写入的地址
所有数据
EEPROM地址
读完了吗？
否
递增数据
EEPROM地址
是
将数据EEPROM信息
写到合并页中
编程数据
校验了吗？
否
设置页
writeError
状态标志
返回
是
将页状态写到合并页中
编程数据
校验了吗？
否
设置页
writeError
状态标志
返回
是
擦除激活页并把页标记
为既不是激活页也不是
已超出限制的页
把合并页的状态
更新为激活页
返回

## 性能

### 有效耐用性

确定有效耐用性的计算并不容易，因为它由很多因素决定。传统上，耐用性是指一个地址能被安全烧写的次数。鉴于一些不同的原因，这个定义并不适用于仿真数据 EEPROM。

首先，对数据 EEPROM 的一个地址烧写五次并不意味着整个存储器件经历了五个擦 / 写周期。从程序存储器的角度看，五次烧写被写到了程序存储器的五个不同的地址上。在填充满整个页并调用合并子程序之前，这五次烧写并没有消耗额外的耐用性周期。

其次，计算有效耐用性并非简单地将程序存储器每页的容量与仿真数据 EEPROM 的容量相乘。这是因为并非整页都能被仿真用到。页状态信息存储在页首，对于 16 位算法，它占用一个程序存储器地址；对于 8 位算法，则占用两个程序存储器地址。此外，在合并子程序执行之后将占用更多的地址，这取决于写入了多少个数据 EEPROM 地址。因此，对一个地址烧写一次会对耐用性产生很大影响，因为每合并一次数组就会少一个地址。

基于上述讨论，可以得出一个简化公式（见公式 1）用于计算总有效耐用性。有关术语方面的更多信息，请参见 **“ 术语定义 ”**。

以 PIC24FJ128GA010 为例，该单片机的程序存储器每页含字 512 个。在 16 位算法中预留了一个地址用于保存页状态。

公式 2 提供了一个计算公式，其中程序存储器内有两页，仿真数据 EEPROM 的地址是 10 个，耐用性限制参数采用典型值。

平均有效耐用性是用总有效耐用性除以仿真数据 EEPROM 存储区的容量得出的，但这与实际情况有出入。这里假设每一个数据 EEPROM 地址以同一速率更新。但在大多数应用中，这种假设是不正确的。某些数据，如校准数据或用户信息可能很少更新，而传感器信息却被写得较为频繁。写操作较为频繁的地址会对程序存储器耐用性产生较大的影响。因此，写操作在数据 EEPROM 地址上的分布情况会对有效耐用性产生很大影响。可对每一地址分配一个比率因子以使计算更加精确，但这仍仅是个近似结果。因为在应用的生命周期内，很难预测每个地址的写入频率。

**注：** EEPROM 存储区被视为不同的 EEPROM，每个存储区都有其自己的有效耐用性。

**公式1：有效耐用性**

$$\textit{总有效耐用性} = (\textit{页容量} - \textit{页状态容量} - \textit{一个数据EEPROM存储区的容量}) \times \textit{页数} \times \textit{耐用性限制参数}$$

**公式2：有效耐用性计算示例**

$$\textit{总有效耐用性} = (512 - 1 - 10) \times 2 \times 1000 = 1002000\textit{个周期}$$

### CPU 暂停

在对程序存储器操作期间， CPU 保持暂停直到写操作结束为止。算法在 DataEEWrite 或 DEE_Write（MCC 16 位 DEE）程序中执行程序存储器写操作。当该程序被调用时， CPU 暂停时间的长短取决于是否对页进行填充。

如果写操作没有针对页进行，CPU暂停时间将会较短—大约是写入一个字的时间量。如果写操作针对当前页进行，暂停时间将会长一些。这是因为合并程序可能将针对合并页执行多行或双字编程操作并对激活页执行擦除操作。

GetNextAvailCount 函数可用来确定当前页被填满的程度以及合并程序被调用之前可对当前页执行的写入次数。该函数返回当前页内到下一个可用地址的偏移量。这个偏移量的数值范围介于 2 和页容量的两倍之间。如果需要的话，可在当前页填满之前调用合并程序。提前执行合并操作有助于将CPU暂停时间的影响降到最小。

**注：** 有关程序存储器工作时序的更多信息，可参见具体器件的数据手册。

## 应用

用户可使用以下任意步骤清单来实现应用中的数据 EEPROM 仿真：

- **PIC18 仿真步骤清单**
- **PIC24/dsPIC33 仿真步骤清单**

无论是 8 位还是 16 位算法都需要占用约 2.7 KB 的程序存储空间。这尚不包括程序存储器中预留给存放数据EEPROM信息的空间。同时这两种算法还需要占用约82字节的数据存储空间。

存储数据 EEPROM 信息的程序存储器是使用一个二维数组进行分配的。数组大小取决于器件的页擦除大小和为仿真预留页的个数。对于 16 位实现方案，编译器会在编译时自动将数组存储在程序存储器下一个可用页的起始位置。对于 8 位实现方案，用户必须指定可用页的起始地址。如果这个地址不与页边界对齐，就会产生编译错误。该数组用来确定要进行表操作的程序存储器地址。如果指定的程序存储器地址范围不可用，就会产生编译错误。

**注：** 对于 PIC18FXXJ 和许多 16 位器件（见具体器件的数据手册来了解更多详细信息），程序存储器的最后一页用于存放配置字信息。它不能被分配用于数据 EEPROM 仿真。

仿真数据 EEPROM 的容量不会对代码大小和数据存储器需求产生显著影响。

这些算法不仅可通过不同配置方法来满足不同类型器件的要求，而且还能满足特定耐用性的要求（见公式 1 和公式 2）。如果需要更强的耐用性，就需要分配更多的页给程序存储器。可采用的一种方式是将擦 / 写限制设置为耐用性参数的额定值而非最小值。以上这些选项可通过改变相关头文件中的常数进行方便的选择。

## PIC18 仿真步骤清单

对NoFilDee.inc文件进行以下修改。用户可通过查阅具体器件数据手册来了解程序存储器的信息。

1. 在 EMULATION_PAGES_START_ADDRESS 中指定仿真页的起始地址。
2. 在 DATA_EE_BANKS 中指定所需的 EEPROM 存储区数量。最大数受器件存储器限制。
3. 在NUM_DATA_EE_PAGES中指定每个EEPROM存储区的程序存储器页数目。指定的最小数目为两页。如果少于两页就会产生编译错误。
4. 在DATA_EE_SIZE中指定所需数据EEPROM的容量。最大是 255 （0xFE） 。如果指定的数据EEPROM 的容量超过 255，就会产生编译错误。
5. 校验ERASE、PROGRAM_ROW和PROGRAM_ WORD操作码的值。
6. 在 NUMBER_OF_INSTRUCTIONS_IN_PAGE 中（以指令条数）指定擦除的最小页容量（典型值为 512）
7. 在 NUMBER_OF_INSTRUCTIONS_IN_ROW 中（以指令条数）指定最大的编程容量（典型值为 64）。
8. 在 ERASE_WRITE_CYCLE_MAX 中选择对每个地址执行擦/写操作的最大次数；最大数为65,535。如果超过这个限额，就会产生编译错误。
9. 在对仿真数据EEPROM执行任何其他操作之前，应首先调用 DataEEInit 函数。
10. 在项目中添加以下文件：
    - DEE Emulation 8-bit.c
    - GenericTypeDefs.h
    - DEE Emulation 8-bit.h
    - NoFilDee.asm
    - NoFilDee.inc

## PIC24/dsPIC33 仿真步骤清单

使用 MCC 16 位 DEE 版本时，无需执行以下步骤清单。

对 DEE Emulation 16-bit.h. 文件进行以下修改。用户可通过查阅具体器件数据手册来了解程序存储器的信息。

1. 在 DATA_EE_BANKS 中指定所需的 EEPROM 存储区数量；最大数受器件存储器限制。
2. 在 DATA_EE_SIZE 中指定每个存储区所需数据 EEPROM 的容量。最大是 255 （0xFE）。如果指定的数据 EEPROM 的容量超过 255，就会产生编译错误。
3. 在NUM_DATA_EE_PAGES中指定每个EEPROM存储区的程序存储器页数目。指定的最小数目为两页。如果少于两页就会产生编译错误。
4. 校验ERASE、PROGRAM_ROW和PROGRAM_ WORD操作码的值。
5. 在 NUMBER_OF_INSTRUCTIONS_IN_PAGE 中（以指令条数）指定擦除的页容量。有关验证器件页擦除大小的信息，请参见具体器件的数据手册。
6. 在 NUMBER_OF_INSTRUCTIONS_IN_ROW 中指定最大编程容量（以指令条数）（dsPIC33E/PIC24E 器件为 128，其他器件系列为 64）。
7. 在 ERASE_WRITE_CYCLE_MAX 中选择最大擦 / 写次数。最大值为 65,535。如果超过这个限额，就会产生编译错误。
8. 在包含闪存 ECC 功能的所有 dsPIC33 和 PIC24 器件中（查看具体器件的数据手册以确定 ECC 是否存在），取消注释掉头文件 “DEE Emulation 16-bit.h” 中的以下行：
   #define __HAS_ECC。
9. 对于 dsPIC33E/PIC24E 器件，注释或取消注释 _AUXFLASH 预处理器定义，以便分别使用主闪存程序存储器或辅助闪存程序存储器。
10. 在对仿真数据 EEPROM 执行任何其他操作之前，应首先调用 DataEEInit 函数。
11. 在项目中添加以下文件：
    - DEE Emulation 16-bit.c
    - DEE Emulation 16-bit.h
    - Flash Operation.s

**注 1：** 地址总数为 DATA_EE_BANKS x DATA_EE_SIZE，范围为 0 至（DATA_EE_BANKS x DATA_EE_SIZE）– 1。

**2：** 在PIC24/dsPIC33器件系列中，如果用户选择了辅助闪存程序存储器选项（即定义了 _AUXFLASH 常数），则 DATA_EE_BANKS x NUM_DATA_EE_PAGES 不得超过 7。

**3：** 在包含闪存 ECC 功能的所有 16 位器件中，数据 EEPROM 仿真所使用的闪存大小是不包含 ECC 的器件所使用闪存的两倍。

## 软件

表 9 列出了用于创建 PIC24/dsPIC33 和 PIC18 解决方案的软件工具和版本信息。也可以使用后续版本的软件工具，但需经过测试以判断其与相应应用的兼容性。

**表 9：不同解决方案中使用的软件工具**

| 软件工具 | 版本 |
|---|---|
| MPLAB® X IDE：www.microchip.com/mplabx | 3.65 或更高版本 |
| MPLAB XC16 C 编译器：www.microchip.com/xc16 | 1.30 或更高版本 |
| MPLAB XC8 C 编译器：www.microchip.com/xc8 | 1.42 或更高版本 |

**表 10：适用于 PIC24/dsPIC33的其他工具**

| 软件工具 | 版本 |
|---|---|
| MPLAB® 代码配置器 （MCC）插件：www.microchip.com/mcc | 4.0.1 或更高版本 |
| MCC 16 位数据 EEPROM 仿真库：https://www.microchip.com/SWLibraryWeb/product.aspx?product=16-Bit_Data_EEPROM_Emulation | 1.0.4 或更高版本 |

用户可登录 Microchip 网站 （www.microchip.com）下载最新源代码和开发工具。有关本应用笔记的最新信息，请阅读软件附带的相关自述文件。

## 结论

对于那些需要高耐用性非易失性数据存储器且同时对成本又较为敏感的应用来说，仿真数据 EEPROM 提供了一种有效的解决方案。适合采用 Microchip 的高性价比 MCU 和 DSC 器件的应用可采用空白程序存储器并将非易失性数据存储器的耐用性提高至原来的 500 倍以上。用户通过选择程序存储器页数、仿真数据 EEPROM 容量和擦 / 写次数限制值来定制“有效耐用性”。通过这一灵活算法，用户可在应用中使用高耐用性的数据 EEPROM。

## 版本历史

### 版本 A （2007 年 4 月）

这是本文档的初始版本。

### 版本 B （2008 年 4 月）

增加了 PIC24/dsPIC33 的多存储器支持。

### 版本 C （2009 年 10 月）

增加了 PIC18 的多存储器支持。

### 版本 D （2011 年 5 月）

该版本包括以下更新：

- 注：
  - 在**“工作原理”**中添加了注释2
  - 在**“页状态”**中更新了注释
  - 在**“PIC24/dsPIC33仿真步骤清单”**中添加了注释2
- 寄存器：
  - 在“寄存器1”中更新了标题
- 部分：
  - 将标题**“PIC24/dsPIC33F/dsPIC33E应用场合”**更新为**“PIC24/dsPIC33案例示例”**
  - 更新了**“PIC24/dsPIC33案例示例”**中的第二段
  - 更新了标题**“PIC24/dsPIC33仿真步骤清单”**
  - 在**“PIC24/dsPIC33仿真步骤清单”**中更新了步骤5、步骤6和步骤10，并添加了步骤8
  - 更新了**“结论”**
- 表：
  - 更新了表9
- 将整个文档中PIC24/dsPIC33F、PIC24和dsPIC33F的所有引用都更新为 PIC24/dsPIC33F/dsPIC33E
- 对闪存的所有引用都更新为闪存程序存储器
- 对主闪存的所有引用都更新为主闪存程序存储器
- 对辅助闪存的所有引用都更新为辅助闪存程序存储器
- 将对文本和格式的细微更改整合到整个文档中

### 版本 E （2018 年 2 月）

该版本包括以下更新：

- 注：
  - 在**“简介”**中更新了注释
  - 在**“工作原理”**中添加了注释3
  - 在**“写操作”**中添加了注释
  - 在**“合并操作”**中添加了注释2
  - 在**“应用”**中更新了注释
  - 在**“PIC24/dsPIC33仿真步骤清单”**中添加了注释3
- 部分：
  - 更新了**“CPU暂停”**的第二段
  - 更新了**“应用”**的第二段
  - 在**“PIC24/dsPIC33仿真步骤清单”**中修改了步骤5和步骤11（之前的步骤10），并添加了步骤8
- 表
  - 更新了表9

### 版本 F （2020 年 12 月）

该版本包括以下更新：

- 对器件的所有引用都更改为 PIC24/dsPIC33。
- 部分：
  - 更新了**“工作原理”**部分。
  - 增加了新部分**“MPLAB®代码配置器16位数据EEPROM仿真支持”**。
  - 更新了**“状态标志”**部分。
  - 更新了**“初始化操作”**部分。
  - 更新了**“读操作”**部分。
  - 更新了**“写操作”**部分。
  - 更新了**“合并操作”**部分。
  - 更新了**“CPU暂停”**部分。
  - 更新了**“PIC18仿真步骤清单”**部分。
  - 更新了**“PIC24/dsPIC33仿真步骤清单”**部分。
- 注：
  - 更新了注释3，并将注释4添加至**“工作原理”**部分。
  - 在**“合并操作”**部分中添加了注释3。
- 表：
  - 更新了表9。
  - 增加了表10。

注：

**请注意以下有关 Microchip 器件代码保护功能的要点：**

- Microchip 的产品均达到 Microchip 数据手册中所述的技术规范。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品非常安全。
- 目前，仍存在着用恶意、甚至是非法的方法来试图破坏代码保护功能的行为。我们确信，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这种试图破坏代码保护功能的行为极可能侵犯 Microchip 的知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。



ISBN：978-1-5224-9411-9

# 全球销售及服务网点


## 美洲

**公司总部 Corporate Office**
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://www.microchip.com/support
网址：www.microchip.com

**亚特兰大 Atlanta**
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

**奥斯汀 Austin, TX**
Tel: 1-512-257-3370

**波士顿 Boston**
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

**芝加哥 Chicago**
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

**达拉斯 Dallas**
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

**底特律 Detroit**
Novi, MI
Tel: 1-248-848-4000

**休斯敦 Houston, TX**
Tel: 1-281-894-5983

**印第安纳波利斯 Indianapolis**
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453
Tel: 1-317-536-2380

**洛杉矶 Los Angeles**
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608
Tel: 1-951-273-7800

**罗利 Raleigh, NC**
Tel: 1-919-844-7510

**纽约 New York, NY**
Tel: 1-631-435-6000

**圣何塞 San Jose, CA**
Tel: 1-408-735-9110
Tel: 1-408-436-4270

**加拿大多伦多 Toronto**
Tel: 1-905-695-1980
Fax: 1-905-695-2078

## 亚太地区

**中国 - 北京**
Tel: 86-10-8569-7000

**中国 - 成都**
Tel: 86-28-8665-5511

**中国 - 重庆**
Tel: 86-23-8980-9588

**中国 - 东莞**
Tel: 86-769-8702-9880

**中国 - 广州**
Tel: 86-20-8755-8029

**中国 - 杭州**
Tel: 86-571-8792-8115

**中国 - 南京**
Tel: 86-25-8473-2460

**中国 - 青岛**
Tel: 86-532-8502-7355

**中国 - 上海**
Tel: 86-21-3326-8000

**中国 - 沈阳**
Tel: 86-24-2334-2829

**中国 - 深圳**
Tel: 86-755-8864-2200

**中国 - 苏州**
Tel: 86-186-6233-1526

**中国 - 武汉**
Tel: 86-27-5980-5300

**中国 - 西安**
Tel: 86-29-8833-7252

**中国 - 厦门**
Tel: 86-592-238-8138

**中国 - 香港特别行政区**
Tel: 852-2943-5100

**中国 - 珠海**
Tel: 86-756-321-0040

**台湾地区 - 高雄**
Tel: 886-7-213-7830

**台湾地区 - 台北**
Tel: 886-2-2508-8600

**台湾地区 - 新竹**
Tel: 886-3-577-8366

## 亚太地区

**澳大利亚 Australia - Sydney**
Tel: 61-2-9868-6733

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

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

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

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

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

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

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

**马来西亚 Malaysia - Kuala Lumpur**
Tel: 60-3-7651-7906

**马来西亚 Malaysia - Penang**
Tel: 60-4-227-8870

**菲律宾 Philippines - Manila**
Tel: 63-2-634-9065

**新加坡 Singapore**
Tel: 65-6334-8870

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

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

## 欧洲

**奥地利 Austria - Wels**
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

**丹麦 Denmark - Copenhagen**
Tel: 45-4485-5910
Fax: 45-4485-2829

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

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

**德国 Germany - Garching**
Tel: 49-8931-9700

**德国 Germany - Haan**
Tel: 49-2129-3766400

**德国 Germany - Heilbronn**
Tel: 49-7131-72400

**德国 Germany - Karlsruhe**
Tel: 49-721-625370

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

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

**以色列 Israel - Ra'anana**
Tel: 972-9-744-7705

**意大利 Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

**意大利 Italy - Padova**
Tel: 39-049-7625286

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

**挪威 Norway - Trondheim**
Tel: 47-7288-4388

**波兰 Poland - Warsaw**
Tel: 48-22-3325737

**罗马尼亚 Romania - Bucharest**
Tel: 40-21-407-87-50

**西班牙 Spain - Madrid**
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

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

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

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