双分区闪存程序存储器

目录

本章包括下列主题：

1.0 简介 2
2.0 程序存储器架构 2
3.0 程序存储器闪存分区操作 7
4.0 闪存编程 13
5.0 程序空间可视性和扩展数据空间 （PSV 和 EDS） 25
6.0 寄存器映射 26
7.0 相关应用笔记 27
8.0 版本历史 28

1.0 简介

所有 PIC24 和 dsPIC33 器件均具有用于执行用户代码的内部可编程闪存阵列。高耐用性闪存阵列同时具有保持时间长和读 / 写周期数高的特性，可以在代码开发和存储中提供极大的灵活性。

该版本的闪存程序存储器增加了以下新特性：

- 双分区闪存操作，可以支持稳健的自举程序系统和以故障保护的形式存储应用程序代码，并具有用以增强代码安全性的选项
- 即时更新操作，可以在修改或完全擦除非活动代码段的同时继续执行主应用程序
- 直接从数据 RAM 空间中对闪存阵列进行运行时编程，并可选择压缩数据 RAM 映像

2.0 程序存储器架构

PIC24 和 dsPIC33 器件可寻址 4M x 24 位的程序存储器地址空间，如图 2-1 所示。程序存储器映射均等地划分为用户程序空间 （000000h 至 7FFFFFh）和配置 （或测试）存储空间 （800000h 至 FFFFFFh）。

用户程序空间包含复位向量、中断向量表（Interrupt Vector Table， IVT）和程序存储器。有三种方法可以访问程序空间。

1. 23 位程序计数器 （Program Counter， PC）。
2. 表读 （TBLRD）和表写 （TBLWT）指令。
3. 通过将程序存储器的 32 KB 段映射到数据存储器地址空间。

所实现的程序存储器可以进一步划分为向量区域（其中包含复位和中断向量）和代码区域（其中还包含闪存配置数据）。访问未实现的用户程序空间区域（即，高于所实现程序存储器的上边界）将导致地址错误陷阱。

2.1 向量区域

向量区域从程序存储空间起始处 000000h 开始。它包含主复位向量、硬件陷阱向量和所有已实现硬件中断的中断向量表 （IVT）。

由于架构差异和 IVT 的大小，向量区域在不同器件系列中会占用不同的存储器容量。对于 PIC24 器件，向量区域延伸至 0000FEh。对于 dsPIC33 器件，向量区域延伸至 0001FEh。图 2-2 说明了不同器件的IVT之间的差异。无论器件系列如何，硬件中断向量总是从000014h以中断向量0开始。

向量区域大致对应于 CodeGuard™ 安全实现中的向量段（Vector Segment， VS）。根据安全配置，向量区域可视为引导段（Boot Segment，BS）或通用段（General Segment，GS）的一部分。

2.1.1 备用中断向量表

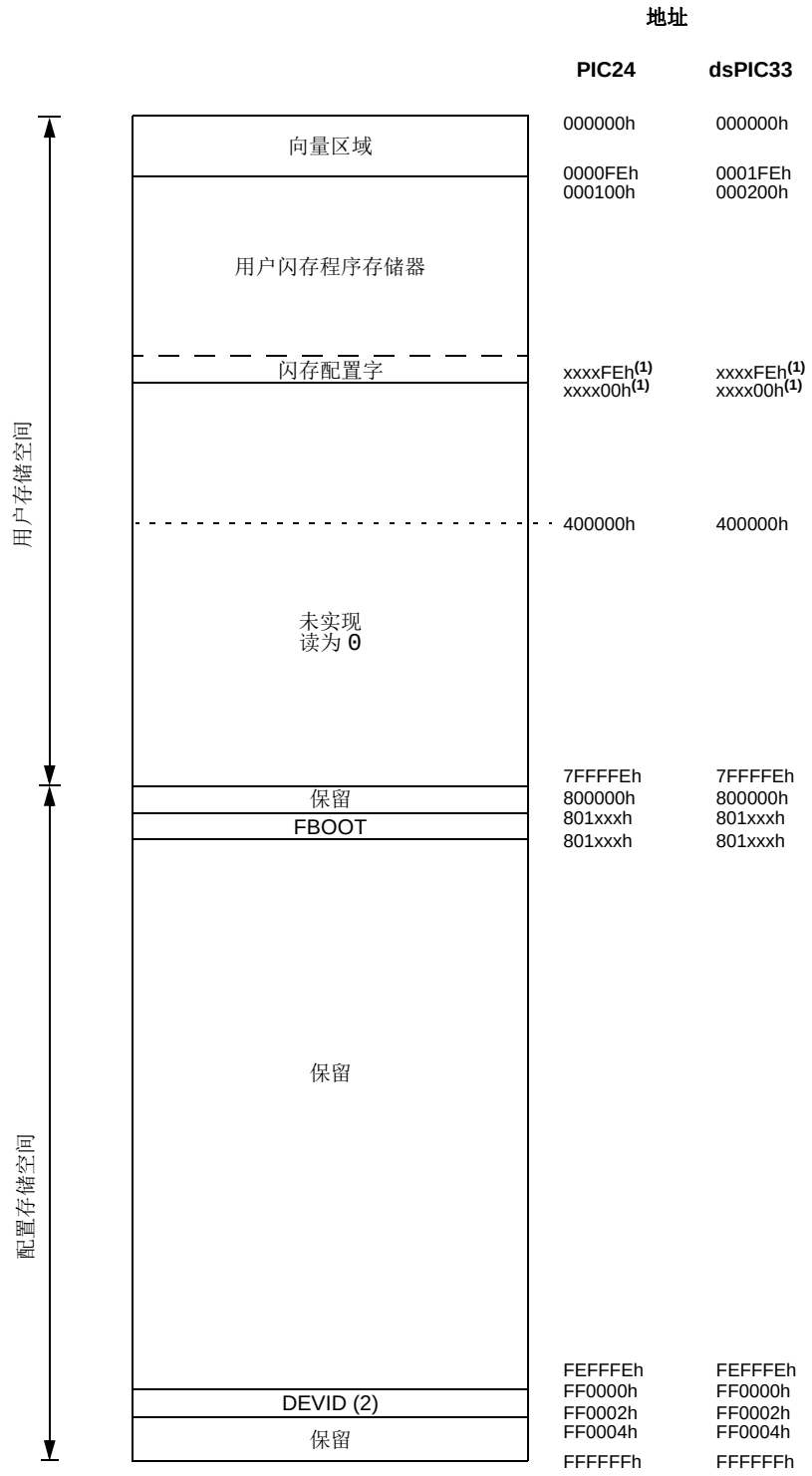
所有 dsPIC33 和 PIC24 器件均可实现备用 IVT （AIVT）， AIVT 可以在高安全性代码应用程序中使用，用于备用异常处理。与这些系列中的先前器件不同，AIVT 并不是永久性地分配在程序存储器中的固定地址范围内。相反， AIVT 仅在以下情况下存在：

- 为大小至少为两页的引导段配置了 CodeGuard 安全性 （通过 FBSLIM 配置寄存器设置），并且
- 通过将 AIVTDIS 配置位编程为 0 使能 AIVT。

当使能 AIVT 时，它位于从 BS 最后一页起始处开始的地址范围；每个向量均位于相对于页边界具有固定偏移量的位置。 AIVT 的总大小和内容 （即，向量顺序）是 IVT 的镜像。

© 2014-2015 Microchip Technology Inc.

图 2-1：　dsPIC33 和 PIC24 器件的默认程序空间存储器映射



图注：　存储区未按比例显示。

注 1：　确切边界地址由所实现程序存储器的大小决定。

图 2-2： 向量区域详细信息

PIC24 器件

中断向量表	复位 ——GOTO 指令	000000
	复位 ——GOTO 地址	000002
	振荡器故障陷阱向量	000004
	地址错误陷阱向量	
	堆栈错误陷阱向量	
	数学错误陷阱向量	
	...	
	中断向量 0	000014
	中断向量 1	000016
	...	
	中断向量 52	00007C
	中断向量 53	00007E
	中断向量 54	000080
	...	
	中断向量 116	0000FC
	中断向量 117	0000FE
备用中断向量表（可选）	保留	BOA + 00
	保留	BOA + 02
	保留	BOA + 04
	振荡器故障陷阱向量	
	地址错误陷阱向量	
	堆栈错误陷阱向量	
	数学错误陷阱向量	
	...	
	中断向量 0	BOA + 14
	中断向量 1	BOA + 16
	...	
	中断向量 52	BOA + 7C
	中断向量 53	BOA + 7E
	中断向量 54	BOA + 80
	...	
	中断向量 116	BOA + FC
	中断向量 117	BOA + FE

dsPIC33 器件

中断向量表	复位 ——GOTO 指令	000000
	复位 ——GOTO 地址	000002
	振荡器故障陷阱向量	000004
	地址错误陷阱向量	
	硬陷阱向量	
	堆栈错误陷阱向量	
	数学错误陷阱向量	
	...	
	中断向量 0	000014
	中断向量 1	000016
	...	
	中断向量 116	0000FC
	中断向量 117	0000FE
	中断向量 118	000100
	...	
	中断向量 244	0001FC
	中断向量 245	0001FE
备用中断向量表（可选）	保留	BOA + 00
	保留	BOA + 02
	振荡器故障陷阱向量	BOA + 04
	地址错误陷阱向量	
	硬陷阱向量	
	堆栈错误陷阱向量	
	数学错误陷阱向量	
	...	
	中断向量 0	BOA + 14
	中断向量 1	BOA + 16
	...	
	中断向量 116	BOA + 0FC
	中断向量 117	BOA + 0FE
	中断向量 118	BOA + 100
	...	
	中断向量 244	BOA + 1FC
	中断向量 245	BOA +1FE

图注： BOA = 偏移量基址，引导段最后一页的起始地址。地址以十六进制显示。

注： 所示的向量区域构成是给定架构的默认构成。具体器件可能会有所不同。关于特定于器件的详细信息，请参见器件数据手册。

© 2014-2015 Microchip Technology Inc.

2.2 代码区域

代码区域是包含用户应用程序代码的用户程序存储器区域。它从向量区域的末尾延伸至闪存配置字开始。如果实现了引导段，则引导段从向量区域的末尾开始，并延伸一个预定范围。不处于引导段中的代码区域部分对应于 CodeGuard 安全系统中的通用段 （GS）。

除了所实现存储器末尾的闪存配置字，整个区域都可用于应用程序代码，如下文所述。

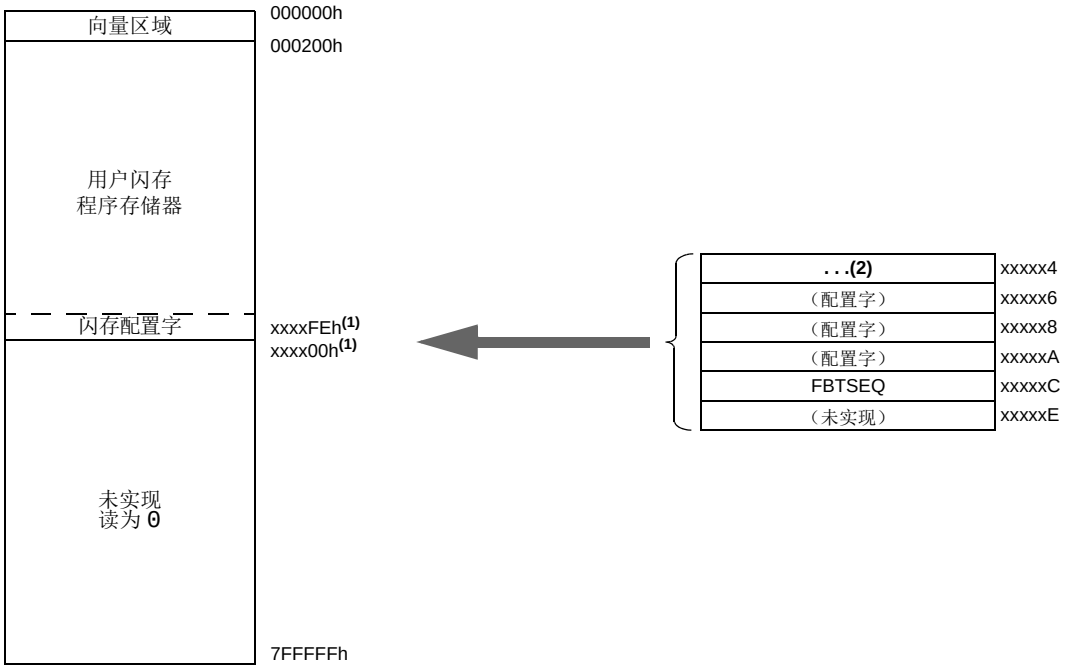
2.2.1 闪存配置数据

所实现闪存程序存储器末尾的区域 （通常为最后一行）保留用于闪存配置数据。在器件复位时，该配置信息会被复制到相应的器件配置寄存器中，这些寄存器是用户不可访问的。器件配置数据只能通过在闪存配置字中编程所需的值来设定。

在器件架构之间和相同架构内的器件系列之间，配置位的数量、顺序和构成会不同。一些器件的配置位由 16 位配置字构成，它们通常按照功能进行分组。其他器件的配置位由可单独寻址配置字节构成。图 2-3 按配置字的构成形式显示了该区域。关于特定于系列的信息，请参见器件数据手册。

对于具有双分区功能的器件，FBTSEQ 配置字通常为倒数第二个配置字，位于所实现程序存储器的末尾。

图 2-3： 闪存配置字



图注： 存储区未按比例显示。

注 1： 确切边界地址由所实现程序存储器的大小决定。

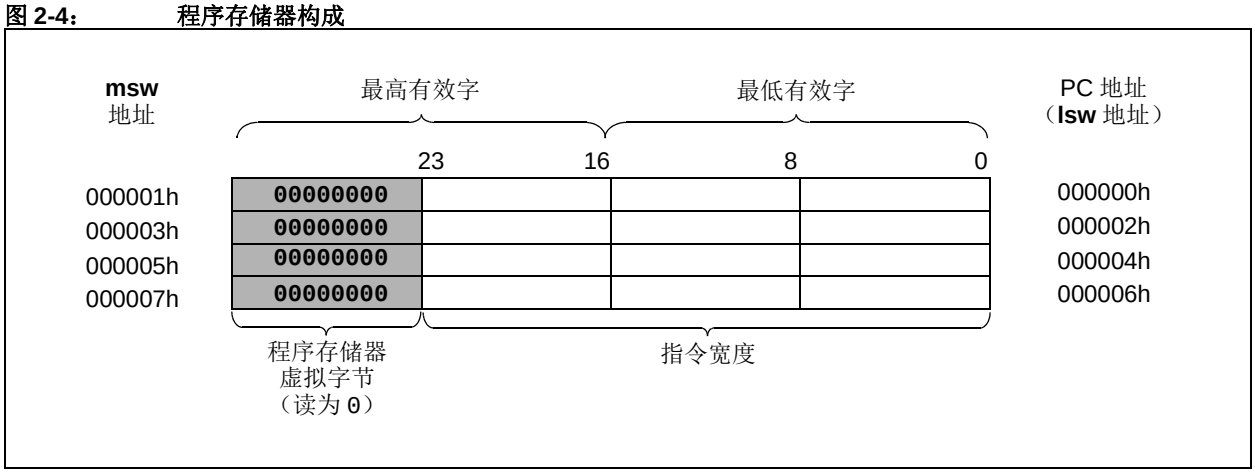
2： 图中所示为典型情况。所实现的闪存配置字 / 字节的确切数量取决于具体器件和架构。一些单元可能不包含配置数据。所实现配置字或字节的确切地址由所实现程序存储器的大小决定。

2.3 存储器构成

程序存储空间由可字寻址的区块构成。虽然它被视为 24 位宽，但将程序存储器的每个地址视作一个低位字和一个高位字的组合更加合理，其中高位字的高字节部分未实现。低位字的地址始终为偶数，而高位字的地址为奇数（图 2-4）。

程序存储器地址和 PC 总是在低位字处进行字对齐（即，最低位（LSb）总是为 0）。在代码执行过程中，地址按 2 递增或递减。

图 2-4： 程序存储器构成



2.3.1 寻址程序存储器

对于正常代码执行，执行的有效地址（Effective Address，EA）由程序计数器（PC）提供。PC 为 23 位宽，允许直接访问用户程序空间中的任意单元。PC<0> 固定为 0，以保持程序指令对齐。PC 通过递增 PC<1> 来递增到下一个连续地址，从而使 PC 值增 2。

对于表读和表写操作，通过将 W 寄存器的 16 位地址与 TBLPAG 寄存器的 8 位地址串联来构造 EA。这使表操作同时可以访问用户和配置空间。**第 4.2.1 节 " 表操作的地址生成 "** 详细介绍了表操作的地址生成。

对于扩展数据空间（Extended Data Space，EDS）和程序空间可视性（Program Space Visibility，PSV）操作，通过将 W 寄存器的低 15 位与 DSRPAG/DSWPAG 寄存器（dsPIC33）或 PSVPAG 寄存器（PIC24F）的 8 位地址串联来构造 EA。《dsPIC33/PIC24 系列参考手册》中 **" 数据存储器 "**（dsPIC33 为 DS70595， PIC24 为 DS39717）对扩展数据空间和程序空间可视性操作进行了介绍。

© 2014-2015 Microchip Technology Inc.

3.0 程序存储器闪存分区操作

对于具有双分区闪存功能的器件，双分区程序存储器模式通过对 FBOOT 配置字中的 BTMODE<1:0> 位编程来进行选择。与其他配置字不同的是， FBOOT 位于配置存储空间，远离所有其他闪存配置寄存器。其确切地址特定于架构（即， PIC24 或 dsPIC33），可能因器件系列而异。表 3-1 列出了可能的闪存分区选项，以下几节将对它们进行介绍。

当第一次通过在线串行编程 （In-Circuit Serial Programming™， ICSP™）对器件进行编程时，编程器应通过编程 FBOOT 来正确设置器件闪存分区模式。请注意，在运行时无法使用运行时自编程（Run-Time Self-Programming， RTSP）来重新编程 FBOOT。必须通过编程器在 ICSP 模式下配置 FBOOT 位。这是因为从标准分区模式到双分区模式，闪存配置字的位置会改变，这可能导致意外的器件操作。

表 3-1： 闪存分区选项

BTMODE<1:0>	分区选项
11	标准模式 （单分区，默认设置）
10	双分区模式
01	受保护双分区模式
00	特权双分区模式 (1)

注 1： 并非在所有双分区器件上实现。

3.1 标准 （单分区）模式

标准模式也称为单分区或标准分区模式，是程序存储器的默认工作模式。当 BTMODEx 配置位为 11 （它们的未编程配置）时会选择该模式。它也是唯一一种所有先前 dsPIC33 和 PIC24 器件均可用的程序存储器工作模式。

在标准模式下，整个用户程序存储器映射为一整块连续的存储空间，范围为从 000000h 到所实现闪存的上边界。例如，对于具有 256 KB 闪存的器件，程序存储器地址范围为 000000h 至 02AFFFh，高于该范围的地址是未实现的。所实现的整个存储器范围 （不包括用于复位向量、IVT 和闪存配置字的保留空间）均可供用户应用程序使用。在具有段代码安全功能的器件中，还可实现引导段。

3.2 双分区模式

当 BTMODEx 配置位编程为 11 之外的值时，器件将以三种双分区模式之一工作。在所有这些模式下，所实现的闪存对称地分为两个区域：活动分区 （从 000000h 开始）和非活动分区 （从 400000h 开始）。对于前面示例中的器件， 256 KB 闪存将以各为 128 KB 的两个区域的形式实现，范围为地址 000000h 至 0157FFh 和 400000h 至 4157FFh。这两个区域之间的地址是未实现的 （见图 3-1）。

在双分区模式下，可以向器件中编程写入两个独立的应用程序，每个应用程序编程到两个闪存分区 （称为分区 1 和分区 2）之一中。在初始化该器件时，两个分区之一会被动态地映射到活动分区并执行。另一个被映射到非活动分区，用于程序存储器操作。分区到活动或非活动分区的分配是由代码签名（称为引导序列号）自动决定的。在运行时，代码分区可以在软件控制下于活动和非活动分区之间交换。

双分区模式允许活动分区的应用程序访问（但不执行）非活动分区中的程序数据，或重新编程非活动分区。对非活动分区中的闪存进行写操作不需要 CPU 在发生闪存写操作时停顿。这可以用于实现即时更新功能，即关键控制功能的执行或时序敏感的通信可以与应用程序更新同时发生。某些双分区模式对该过程具有一些额外的限制，以帮助确保代码的安全和操作的稳健性。但代码映射到非活动分区时无法执行。分区可以交换，但只有在活动分区中的代码才能执行。

3.2.1 双分区模式

最简单的双分区模式对于从活动分区到分区 1 或 2 中代码的操作没有任何限制。对于不同分区中的代码段之间相互操作的所有限制均由增强型安全功能的配置决定。

3.2.2 受保护双分区模式

受保护双分区模式会保护默认代码段（分区 1），阻止任何闪存写操作或擦除操作。这使用户可以在分区 1 中存储故障保护备份映像，从而实现 “ 出厂默认 ” 模式。

使用受保护双分区模式时，当分区 1 处于非活动分区时，无法通过闪存操作写入或擦除它。如果分区 1 还通过配置位设置进行写保护，则任何时候都不能擦除或写入它。相反，分区 2 可以由来自任一分区的操作进行擦除或写入。

这使用户可以在分区 1 中放入故障保护自举程序以及故障保护备份代码映像。然后，在闪存更新发生故障时，可以默认执行该代码映像，并使用它来重写分区 2。

3.2.3 特权双分区模式

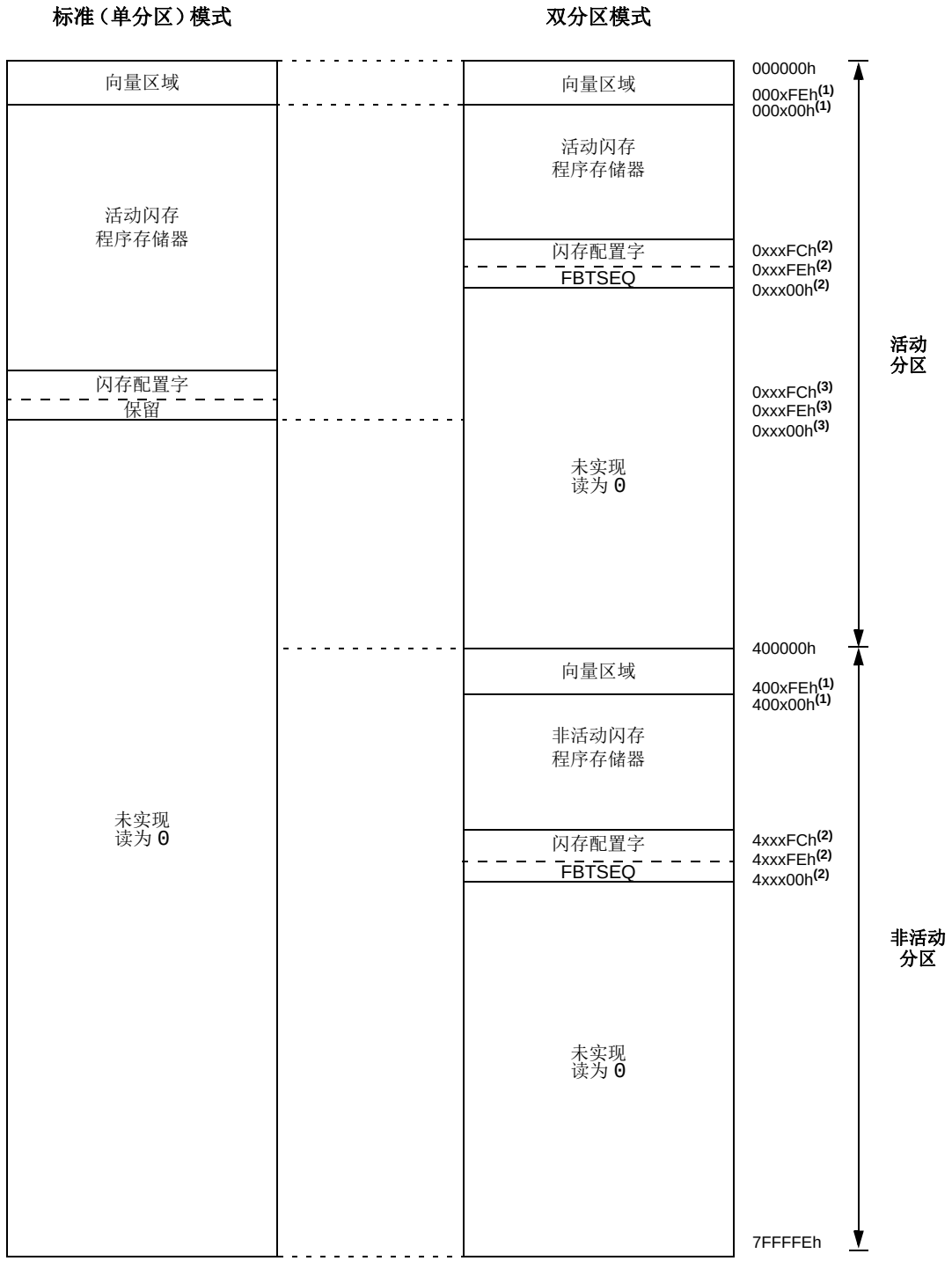
特权双分区模式实现了额外的安全保护措施，用于处理如下情况：应用程序可能具有由不同作者编写的代码段，需要更高的安全级别来保护其中一个段的知识产权。例如，某个系统的大部分代码是由硬件的应用程序开发人员编写的，但包含了专有的第三方库。该模式设计为与选定器件中的增强型安全功能配合工作，可以有选择地保护程序存储空间中的不同代码段。

特权双分区模式与标准双分区模式的不同之处在于增加了对两个分区的 BSLIMx 配置位的特殊保护。这种保护可以有效地锁定这些位，防止更改引导段和通用段的大小。借助适当的安全设置，可以确保不会在运行时更改或意外地读取这两个段。

特权双分区模式并未在所有具有双分区功能的器件上实现。详情请参见具体器件的数据手册。

© 2014-2015 Microchip Technology Inc.

图 3-1： 标准和双分区用户存储空间映射



图注： 存储区未按比例显示。

注 1： 对于 PIC24 器件，默认向量区域边界为 000100h，对于 dsPIC33 器件为 000200h。

2： 双分区模式下的存储器边界值是标准分区模式下值的一半。

3： 确切程序存储器边界由所实现程序存储器的大小决定。

3.2.4 选择代码分区

在双分区模式下，有两种方法可以决定哪个分区将被映射到活动分区并执行：引导序列号和 BOOTSWP 指令。P2ACTIV 位（NVMCON<10>）用于确定哪个物理分区为活动分区。如果 P2ACTIV = 1，分区 2 为有效分区；如果 P2ACTIV = 0，分区 1 为有效分区。

引导序列号是一个 12 位值，用于在器件复位时自动决定活动分区。每个分区应具有唯一的引导序列号，它存储在 FBTSEQ 闪存配置字中。

BOOTSWP 指令用于在不复位器件的情况下交换活动和非活动分区。

3.2.4.1 引导序列号

12 位引导序列号存储在 FBTSEQ 闪存配置字中，总是位于用户程序存储器的最后一个单元，处于其他闪存配置字之上（见图 3-2）。与其他配置寄存器（仅使用程序存储器字的低 16 位）不同的是，FBTSEQ 为完整的 24 位宽。在正常工作条件下，每个分区应具有不同的 FBTSEQ 值。不使用双分区模式时，FBTSEQ 的值将被忽略。

引导序列号分为两个部分存储：位域 BSEQx（FBTSEQ<11:0>）中存储实际值，IBSEQx 位域（FBTSEQ <23:12>）中存储该值的二进制补码。在器件复位后读取引导序列号时，会自动比较 BSEQx 和 IBSEQx 的值。如果这两个值不互补，则引导序列号会被视为无效。互补值不是由硬件自动创建的，在编程时硬件也不会进行校验。应用程序必须计算并编程写入适当的值。

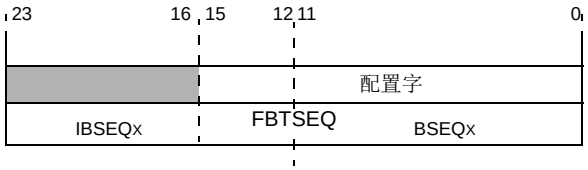
在器件复位时，会对两个分区中的引导序列号进行比较。具有较小 BSEQx 值的分区即为被映射到活动分区的分区，其代码会被执行。无论哪个引导序列号较小，如果其中一个引导序列号无效，器件将选择具有有效引导序列号的分区作为活动分区。如果两个引导序列号均无效，默认情况下将选择分区 1 作为活动分区。

在运行时可以交换分区，方法是重新编程非活动分区的引导序列号，使之具有较小的值。当执行复位时，具有较小值的分区现在会变为活动分区。该方法在以下情况下使用：对非活动分区进行了更新，并在复位之后映射为活动分区。

由于 FBTSEQ 所处的位置，可以从校验和或闪存程序存储器的其他校验中方便地排除它。由于 FBTSEQ 值可能需要在运行时确定（基于另一个分区的 BSEQx），所以通常不能将它包含在校验和中，例如 CRC。

图 3-3 上方的序列显示了在更改引导序列号并执行器件复位时代码分区之间的关系。

图 3-2： FBTSEQ 与其他配置字的关系（仅对于双分区模式）



图注： 阴影区域是未实现的，总是读为 0。地址反映的是所实现程序存储器的上边界处的相对位置。

注： FBTSEQ 和其他配置字并不一定邻近；此处所示的关系是为了进行比较。

© 2014-2015 Microchip Technology Inc.

3.2.4.2 BOOTSWP 指令

BOOTSWP 指令是 PIC24 和 dsPIC33 指令集的扩展。通过它可以在无需器件复位的情况下在活动和非活动分区之间交换代码段，从而支持代码即时更新。使用 BOOTSWP 指令进行的分区交换称为“软交换”。要执行 BOOTSWP 指令，必须清零配置位 BTSWP（FIDC<25>）。若在 BTSWP 置 1 的情况下，尝试执行 BOOTSWP 指令，将会导致执行 NOP 指令。

BOOTSWP 指令必须总是紧跟着一条写入 PC 的单字指令（例如，GOTO W、CALL W 或 BRA W）；该指令的目标必须为处于当前地址 32 KB 范围内的地址。在执行时，活动和非活动分区会交换位置， PC 会跳转到指向由 GOTO 指令在新活动分区中指定的位置。

注：	如果从使用 LNK 指令创建了新堆栈帧的函数内执行 BOOTSWP 指令，则必须在 BOOTSWP 之后使用 CALL，而不是 GOTO；否则，器件将产生堆栈错误陷阱。

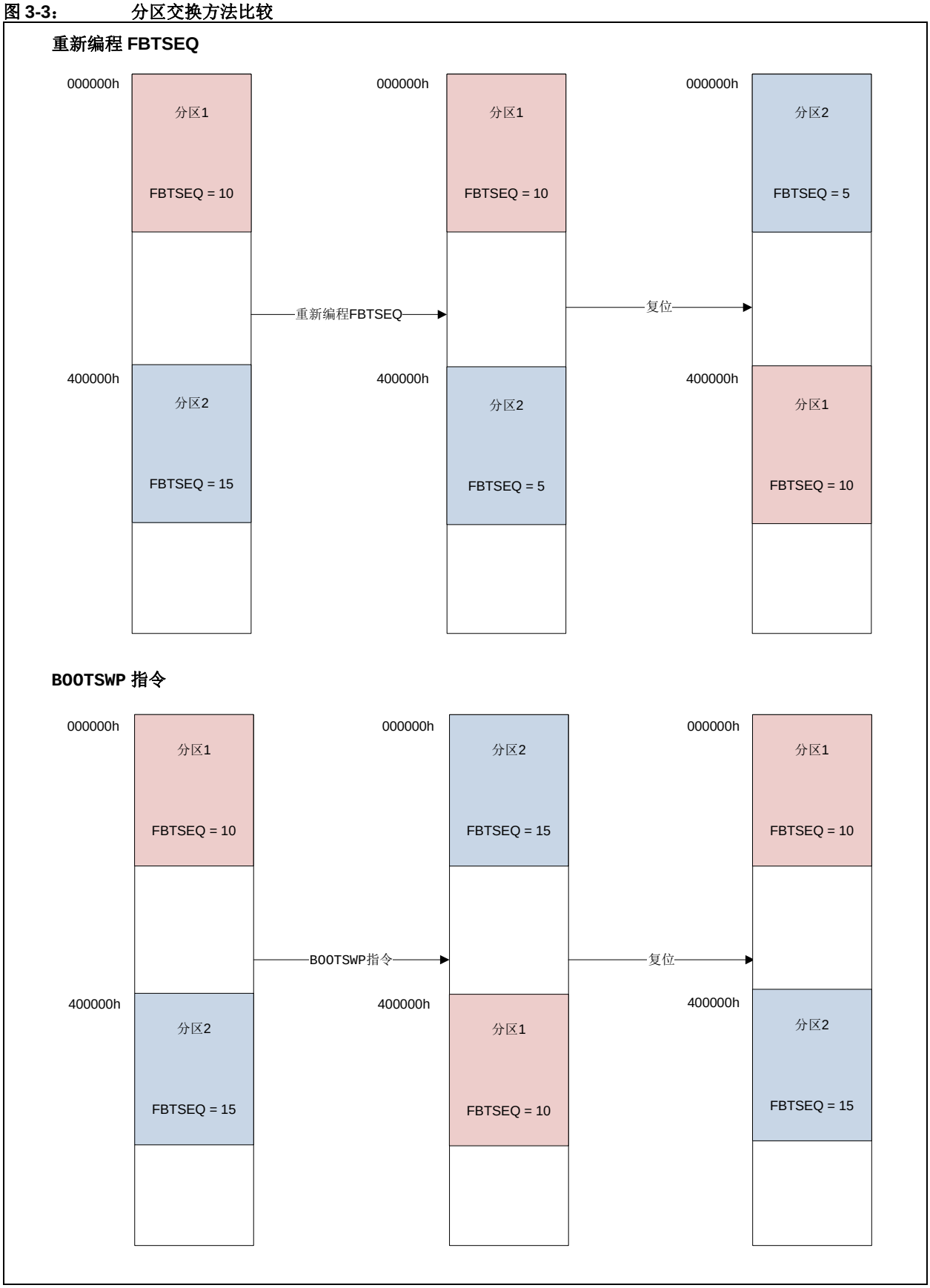
执行 BOOTSWP 指令后，SFTSWP 位（NVMCON<11>）会置 1。该位向固件指示 BOOTSWP 指令已正确发生，当前活动分区是通过 BOOTSWP 而不是通过器件复位进入的。还可以读取状态位 P2ACTIV （NVMCON<10>）以验证哪个分区处于活动状态。

在分区交换后，所有先前使能的外设和允许的中断将保持原有状态，注意到这一点很重要。此外，RAM和堆栈会在交换后维持它们的状态。强烈建议使用软交换的应用程序跳转到一个重新初始化器件的程序，以确保应用程序按预期方式继续运行。

为了确保操作的稳健性，在执行 BOOTSWP 指令之前需要执行标准 NVM 解锁序列 （在两个顺序步骤中向 NVMKEY 寄存器中写入 55h 和 AAh；更多信息，请参见**第 4.1 节“寄存器”**）。执行解锁序列之前还需要禁止中断，这一点非常重要。如果不执行解锁序列，则 BOOTSWP 将作为强制 NOP 执行。 BOOTSWP 后的 GOTO 指令仍然会执行，导致 PC 跳转到当前工作分区中的相应位置。类似地， BOOTSWP 在标准引导模式下没有任何作用。

图 3-3 下方的序列显示了在执行 BOOTSWP 指令时分区之间的关系。请注意，BOOTSWP 分区变化是暂时的；在随后的器件复位后，具有较小引导序列号的分区会被重新指定为活动分区。

图 3-3： 分区交换方法比较



© 2014-2015 Microchip Technology Inc.

4.0 闪存编程

PIC24 和 dsPIC33 器件可以通过三种方法之一进行编程：

- 运行时自编程 （RTSP）
- 在线串行编程 （ICSP™）
- 增强型在线串行编程 （EICSP）

RTSP 由应用程序软件在执行期间进行，而 ICSP 和 EICSP 则从使用串行数据连接与器件相连的外部编程器中执行。 ICSP 和 EICSP 支持比 RTSP 要快得多的编程速度。

本节将描述 RTSP 技术。相应器件的编程规范文档中定义了 ICSP 和 EICSP 协议，这些文档可从 Microchip 网站 （www.microchip.com）下载。

4.1 寄存器

编程操作通过 6 个寄存器进行控制。NVMCON 和 NVMKEY 寄存器用于使能和选择所有操作。其余 4 个寄存器用于定义数据和地址指针。

注：	并非所有器件都实现了数据 RAM 缓冲区编程。更多信息，请参见具体器件的数据手册。

4.1.1 控制寄存器

NVMCON 寄存器（寄存器 4-1）用于控制所有闪存编程操作。NVMOP<3:0> 位（NVMCOM<3:0>）用于选择要执行的特定写操作或擦除操作。 WR 位 （NVMCOM<15>）用于触发相应的操作；它会保持置 1，直到该操作完成，然后由硬件清零。 WREN 位 （NVMCOM<14>）用于使能或禁止写操作和擦除操作。当 WREN 清零时，无法通过将 WR 位置 1 来触发操作。

NVMKEY 是一个只写寄存器，用来防止意外写入或擦除闪存；仅实现了低字节。要启动编程或擦除序列，需要对寄存器执行包含两个写操作的 “ 解锁 ” 序列：

1. 将 55h 写入 NVMKEY<7:0>。
2. 将 AAh 写入 NVMKEY<7:0>。

在此序列后，允许在一个指令周期中写入 NVMCON 寄存器。

4.1.2 地址寄存器

NVMADRL 和 NVMADRH 寄存器用于定义写操作的起始地址指针。两种类型的程序存储器写操作 （基于锁存器和 RAM 缓冲）均使用这些寄存器来设置目标地址。

使用 RAM 缓冲编程时，NVMSRCADRL 和 NVMSRCADRH 寄存器定义源数据在数据 RAM 中的起始地址。在具有扩展数据空间 （EDS）的器件上， NVMSRCADRH 寄存器用于指向扩展数据空间存储器中的地址。

寄存器 4-1： NVMCON：闪存编程控制寄存器

R/S-0[1]	R/C-0	R/C-0	R/W-0	R/C-0	R-0	R/W-0	R/C-0
WR	WREN	WRERR	NVMPIDL[5]	SFTSWP	P2ACTIV	RPDF[2]	URERR[2]
bit 15							bit 8

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	NVMOP3	NVMOP2	NVMOP1	NVMOP0
bit 7							bit 0

图注：	S = 只可置 1 位	C = 只可清零位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15 **WR：**写控制位 [1]
1 = 启动闪存编程 / 擦除操作
0 = 编程 / 擦除操作完成，并处于停止状态

bit 14 **WREN：**编程 / 擦除使能位
1 = 允许编程 / 擦除周期
0 = 禁止编程 / 擦除存储器或熔丝元件

bit 13 **WRERR：**序列错误标志位
1 = 发生了错误的编程 / 擦除终止，或选择了未实现的编程操作
0 = 编程或擦除操作正在进行中，已正常完成或已经开始

bit 12 **NVMPIDL：**NVM 空闲掉电使能位 [5]
1 = 当器件进入空闲模式时，停止为闪存阵列供电
0 = 当器件进入空闲模式时，在待机模式下保持为闪存阵列供电

bit 11 **SFTSWP：**软交换状态位
当 BTMODE<1:0> = 10 或 0x 时：
1 = 已使用 BOOTSWP 指令成功地交换分区
0 = 正在等待使用 BOOTSWP 指令成功地交换分区
当 BTMODE<1:0> = 11 时：
未实现，读为 0。

bit 10 **P2ACTIV：**双分区活动分区状态位
当 BTMODE<1:0> = 10 或 0x 时：
1 = 分区 2 闪存为活动分区
0 = 分区 1 闪存为活动分区
当 BTMODE<1:0> = 11 时：
未实现，读为 0。

bit 9 **RPDF：**RAM 缓冲区行编程数据格式控制位 [2]
1 = 行数据以压缩格式存储在 RAM 中
0 = 行数据以未压缩格式存储在 RAM 中

注 1： 发生欠压复位 （Brown-out Reset， BOR）时该位也会复位。
2： RAM 缓冲区行操作并不是在所有器件上均可用；在这些情况下，这些位是未实现的，读为 0。
3： 选择这些选项会使 WRERR 位置 1 并使 WR 位清零。
4： 双字编程操作需要两个在 4 字边界处对齐的相邻指令字 （各 24 位）。
5： 仅在特定器件中实现。欲知详情，请参见具体器件的数据手册。

© 2014-2015 Microchip Technology Inc.

寄存器 4-1： NVMCON：闪存编程控制寄存器 （续）

bit 8 **URERR**：RAM 缓冲区行编程数据不足错误标志位 (2)
1 = 编程操作已由于数据不足错误而终止
0 = 未检测到数据不足错误

bit 7-4 **未实现**：读为 0

bit 3-0 **NVMOP<3:0>**：NVM 操作选择位 （通过 WR 的下一次置 1 启动）
1xxx = 保留 (3)
011x = 保留 (3)
0101 = 保留 (3)
0100 = 非活动分区擦除操作 （在标准分区模式下为保留选项）
0011 = 页擦除操作
0010 = 行编程操作
0001 = 双字编程操作 (4)
0000 = 保留 (3)

注 1： 发生欠压复位 （Brown-out Reset， BOR）时该位也会复位。
2： RAM 缓冲区行操作并不是在所有器件上均可用；在这些情况下，这些位是未实现的，读为 0。
3： 选择这些选项会使 WRERR 位置 1 并使 WR 位清零。
4： 双字编程操作需要两个在 4 字边界处对齐的相邻指令字 （各 24 位）。
5： 仅在特定器件中实现。欲知详情，请参见具体器件的数据手册。

4.2 表操作指令

表指令提供了一种在 PIC24 和 dsPIC33 器件的程序存储空间和数据存储空间之间传输数据的方法。本节提供了对闪存程序存储器编程期间使用的表指令汇总。有 4 条基本的表指令：

- TBLRDL：表读低位字
- TBLRDH：表读高位字
- TBLWTL：表写低位字
- TBLWTH：表写高位字

TBLRDL 和 TBLWTL 指令用于读写程序存储空间的 bit <15:0>。 TBLRDL 和 TBLWTL 可以以字或字节模式访问程序存储器。

TBLRDH 和 TBLWTH 指令用于读或写程序存储空间的 bit <23:16>。TBLRDH 和 TBLWTH 同样能以字或字节模式访问程序存储器。因为程序存储器只有 24 位宽，所以 TBLRDH 和 TBLWTH 指令能寻址程序存储器中并不存在的高字节。该字节被称为“虚拟字节”。读取虚拟字节将总是返回00h，而对其进行写操作则不起作用。

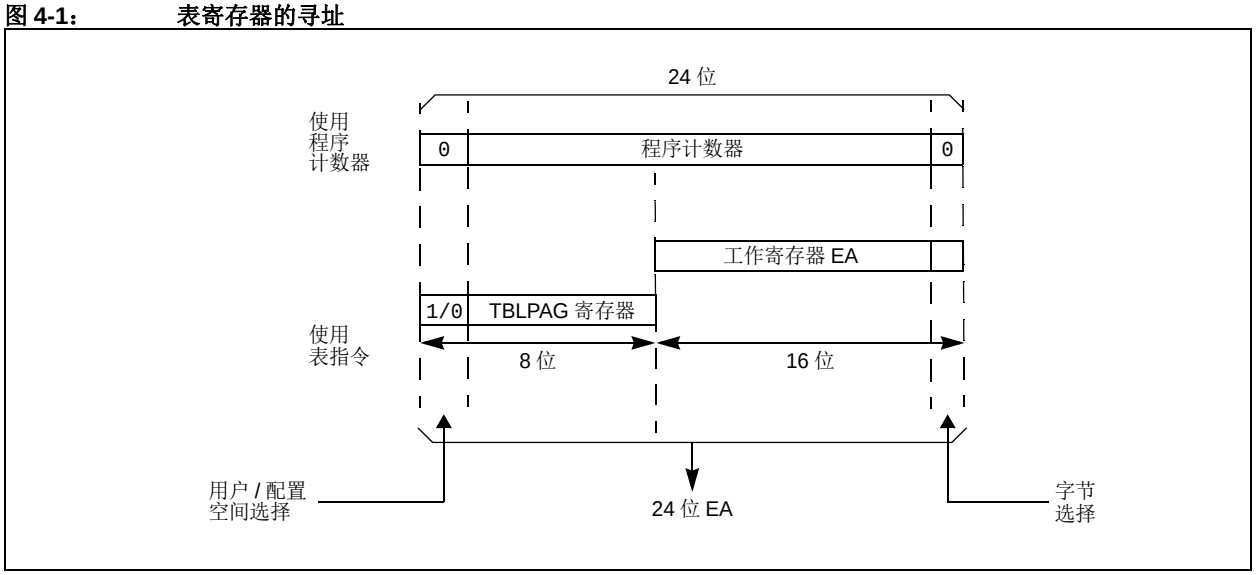
4.2.1 表操作的地址生成

24 位程序存储器可以视作两个并排放置的 16 位空间，它们共享同一个地址范围。因此， TBLRDL 和 TBLWTL 指令可访问 “ 低 ” 程序存储空间 （PM<15:0>）。 TBLRDH 和 TBLWTH 指令可访问 “ 高 ” 程序存储空间 （PM<31:16>）。对 PM<31:24> 的任何读或写操作将访问虚拟 （未实现）字节。当在字节模式下使用任何表指令时，表地址的 LSb 将被用作字节选择位。LSb 决定将访问高或低程序存储空间的哪个字节。

图 4-1 显示了如何使用表指令寻址程序存储器。 24 位程序存储器地址由 TBLPAG<7:0> 位以及表指令指定的 W 寄存器中的有效地址（EA）组成（为了供以参考，图中显示了 24 位程序计数器）。EA 的高 23 位用于选择程序存储单元。对于字节模式的表指令，W 寄存器 EA 的 LSb 用于选择 16 位程序储存字中要寻址的字节。“1” 选择 bit <15:8>， “0” 选择 bit <7:0>。 W 寄存器 EA 的 LSb 在字模式下的表指令中会被忽略。

除了指定程序存储器地址外，表指令还指定作为要写入的程序存储器来源或要读取的程序存储器目标的 W 寄存器 （或指向存储单元的 W 寄存器指针）。对于字节模式下的表写操作，源工作寄存器的 bit <15:8> 会被忽略。

图 4-1： 表寄存器的寻址

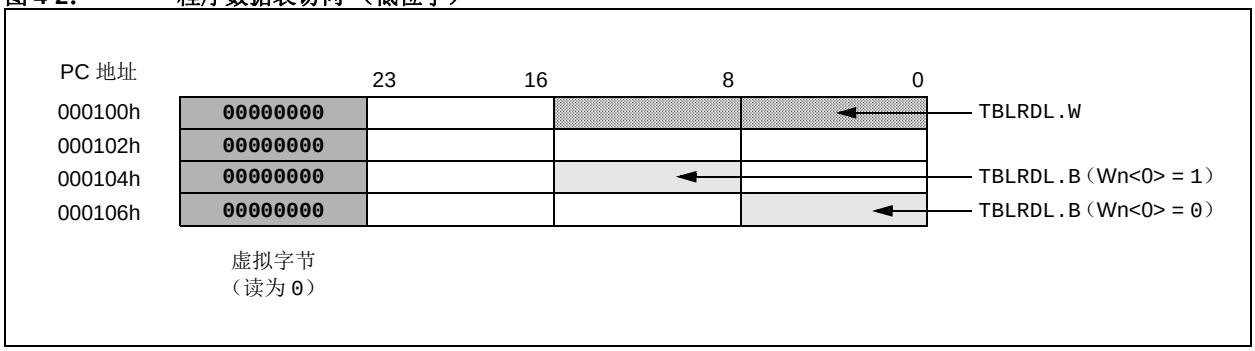


© 2014-2015 Microchip Technology Inc.

4.2.2 低位字访问

TBLRDL 和 TBLWTL 指令用于访问程序存储器数据的低 16 位。对于以字宽为单位的表访问，W 寄存器地址的 LSb 被忽略。对于以字节宽为单位的访问，W 寄存器地址的 LSb 决定读哪个字节。图 4-2 说明了通过 TBLRDL 和 TBLWTL 指令访问的程序存储器数据区域。

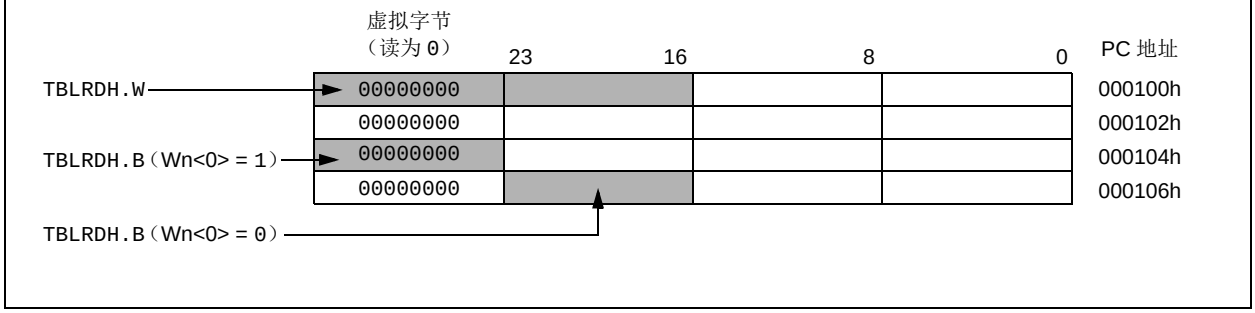
图 4-2： 程序数据表访问 （低位字）



4.2.3 高位字访问

TBLRDH 和 TBLWTH 指令用于访问程序存储器数据的高 8 位。为了正交性，这些指令也支持字或字节访问模式，但是程序存储器数据的高字节将总是返回 0，如图 4-3 所示。

图 4-3： 程序数据表访问 （高位字）



4.2.4 程序存储器中的数据存储

假设对于大多数应用，高字节 （PM<23:16>）将不会用于存储数据，使程序存储器有 16 位宽供数据存储。建议将程序数据的高字节编程为 NOP （00h 或 FFh）或作为非法操作码 （3Fh）值，以防止器件意外执行所存储的数据。 TBLRDH 和 TBLWTH 指令主要用于阵列编程 / 校验和那些要求压缩数据存储的应用。

4.2.5 程序存储器位行为

闪存程序存储器中的位只能从 1 编程为 0，可以随后擦除为 1。尝试通过编程序列将某个位置 1 没有任何作用。

4.2.6 使用表读指令

表读需要两个步骤。首先，使用 TBLPAG 寄存器和一个 W 寄存器建立一个地址指针。然后就可以读取地址单元的程序存储器内容了。

例 4-1 和例 4-2 中的代码示例说明了如何使用字模式下的表指令读取程序存储器的一个字。

例 4-1： 读字模式 （汇编语言）

```
; Set up the address pointer to program space
MOV     #tblpage(PROG_ADDR),W0        ; get table page value
MOV     W0,TBLPAG                     ; load TBLPAG register
MOV     #tbloffset(PROG_ADDR),W0      ; load address LS word
; Perform the table writes to load the latch
TBLRDL  [W0],W2
TBLRDH  [W0],W3
```

例 4-2： 读字模式 （C 语言）

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

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

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

注： Microchip 的 dsPIC33 和 PIC24 器件的汇编器提供了 `tblpage()` 和 `tbloffset()` 伪指令。这两条伪指令会从程序存储器地址值中为表指令选择适当的 TBLPAG 和 W 寄存器值。更多信息，请参见 《MPLAB® ASM30、 MPLAB LINK30 和实用程序用户指南》（DS51317F_CN）。

© 2014-2015 Microchip Technology Inc.

4.2.7 表写保持锁存器

表写指令不会直接写入闪存程序阵列。相反，这些指令会导致要编程的数据先装入保持锁存器。这些锁存器通过存储器映射的方式映射在配置存储空间中（通常从 FA0000h 开始），并且只能使用表写指令访问。当所有保持锁存器都被装入后，通过执行一个特殊的指令序列即可启动实际的存储器编程操作。

不同器件基于特定的程序阵列设计（即，行编程大小和行编程算法）实现了不同数量的保持锁存器。更多详细信息，请参见具体器件的数据手册和 / 或编程规范。

4.2.7.1 执行双字写操作

使用一对 TBLWTH 和 TBLWTL 指令一次执行两个字的字写操作。例 4-3 或例 4-4 （等效 C 代码）中的任一代码序列均可用于通过字写模式写入两个要编程到闪存中的程序存储器锁存器单元。

例 4-3： 双字写操作示例 （汇编语言）

```
; Set up the address pointer to 1st write latch
MOV     0xFA,W0                                ; get table page value
MOV     W0,TBLPAG                              ; load TBLPAG register
MOV     0x0,W0                                 ; load address LS word
; Load write data into W registers
MOV     #PROG_LOW_WORD_1,W2
MOV     #PROG_HI_BYTE_1,W3
MOV     #PROG_LOW_WORD_2,W4
MOV     #PROG_HI_BYTE_2,W5
; Perform the table writes to load the latch
TBLWTL  W2,[W0]
TBLWTH  W3,[W0++]
TBLWTL  W4,[W0]
TBLWTH  W5,[W0++]
```

例 4-4： 双字写操作示例 （C 语言）

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

NVMCON = 0x4001;                    // Set WREN and word program mode
TBLPAG = 0xFA;                      // write latch upper address
addrOffset = (PROG_ADDR & 0x00FFFE);  // ensure address is properly aligned
NVMADRL = TargetWriteAddressL;      // set target write address
NVMADRH = TargetWriteAddressH;

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

4.3 运行时自编程 （RTSP）

RTSP 允许用户代码修改闪存程序存储器的内容。 RTSP 是通过使用 TBLRD （表读）、 TBLWT （表写）指令和 NVM 控制寄存器实现的。 PIC24 和 dsPIC33 器件支持以下闪存编程操作：

- 闪存页擦除
- 行编程 （基于锁存器或基于 RAM）
- 字编程

通过 RTSP 进行的闪存编程以存储器块（称为行）或两个闪存字的方式执行。在编程之前，必须先擦除存储器单元。擦除操作以存储器块 （称为页，由多行组成）的方式执行。行的大小会因器件而异；关于详细信息，请参见器件数据手册。通常情况下，对于 dsPIC33 和 PIC24 器件，一页定义为 8 行。本文档使用每行 64 条指令 （每页 512 条指令）的示例。

4.3.1 使用写保持锁存器进行行编程

如**第 4.2.7 节 “ 表写保持锁存器 ”** 中所述，实现了基于锁存器的行编程的器件具有一些保持锁存器，其中包含编程数据。在实际编程操作前，必须通过 TBLWT 指令将待写数据顺序装入锁存器。在执行行写操作时，指令字必须以一个完整行的形式装入锁存器。

RTSP 编程的基本步骤是先建立一个表指针，然后执行一系列 TBLWT 指令来装载缓冲区。通过设置 NVMCON 寄存器中的控制位来执行编程。例如，在具有 64 个指令行的器件上，一个编程周期将包含 64 条 TBLWTL 和 64 条 TBLWTH 指令来装入写锁存器，后面跟随一个解锁 NVMCON 和将 WR 位置 1 的编程序列。例 4-5 给出了该过程的一个示例。

例 4-5： 使用写锁存器进行行编程 （C 语言）

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

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

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


© 2014-2015 Microchip Technology Inc.

4.3.2 使用 RAM 缓冲区进行行编程

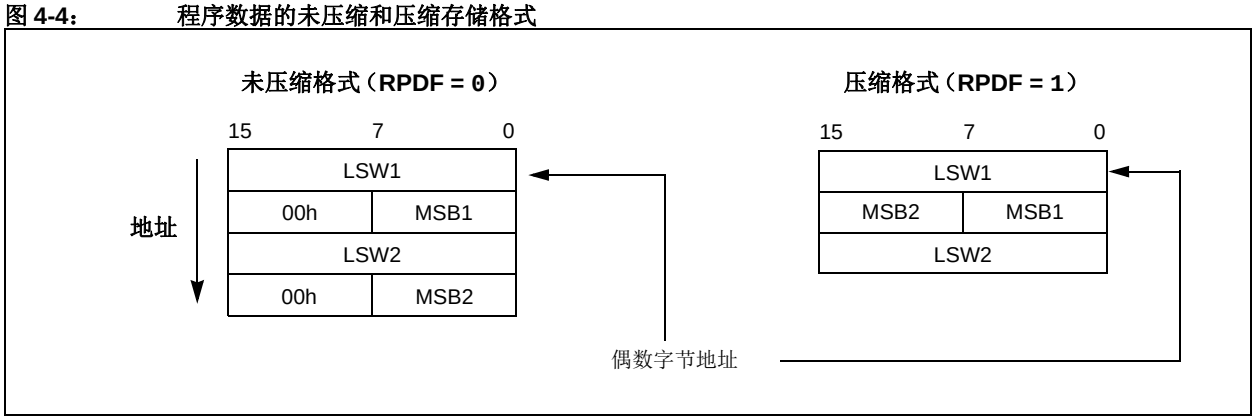
选定的 dsPIC33 和 PIC24 器件允许直接从数据 RAM 中的缓冲区空间中执行行编程，而不是使用 TBLWT 指令通过保持锁存器来传输数据。 RAM 缓冲区的位置由 NVMSRCADR 寄存器决定，该寄存器需要装入包含待写程序数据第一个字的数据 RAM 地址。

在执行编程操作之前，必须先在 RAM 中的缓冲区空间中装入要编程的数据行。可以使用压缩（打包）或未压缩格式来装入 RAM。压缩存储使用一个数据字来存储两个相邻程序数据字的高字节（MSB）。未压缩格式则对于每个程序数据字使用两个数据字，每隔一个字的高字节为 00h。与未压缩格式相比，压缩格式使用数据 RAM 中大约 3/4 的空间。另一方面，未压缩格式会模仿 24 位程序数据字的结构，补足高位虚拟字节。数据格式通过 RPDF 位 （NVMCON<9>）进行选择。图 4-4 给出了这两种格式。

装入 RAM 缓冲区后，闪存地址指针 NVMADRL 和 NVMADRH 中需要装入待写闪存行的 24 位起始地址。与编程写锁存器一样，该过程通过写入 NVM 解锁序列然后将 WR 位置 1 来启动。启动之后，器件会自动装入适当的锁存器并递增 NVM 地址寄存器，直到所有字节都已编程为止。例 4-6 给出了该过程的一个示例。若设置 NVMSRCADR 导致数据下溢错误条件发生， URERR 位 （NVMCON<8>）将置 1 以指示该条件。

实现了 RAM 缓冲区行编程的器件还实现了一个或两个写锁存器。这些锁存器使用 TBLWT 指令来装入数据，并用于执行字编程操作。

图 4-4： 程序数据的未压缩和压缩存储格式



例 4-6： 通过数据 RAM 缓冲区写入程序存储器 （C 语言）

```
int data[64];                       // Data to be programmed in RAM
int targetWriteAddressL;            // bits<15:0>
int targetWriteAddressH;            // bits<22:16>

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

__builtin_disi(5);                  // Disable interrupts for NVM unlock sequence
__builtin_write_NVM();
```

4.4 通用闪存编程算法

闪存编程操作使用以下非易失性存储器 （Nonvolatile Memory， NVM）控制寄存器进行控制：

- NVMCON
- NVMKEY
- NVMADRL/H
- NVMSRCADRL/H （部分器件）

在RTSP模式下，对内部闪存进行编程或擦除需要执行完整的编程序列。将WR位（NVMCON<15>）置 1 启动操作，当操作完成时 WR 位会自动清零。

对活动分区执行闪存编程操作 （特别是在标准分区模式下）时， CPU 会发生停顿，直到操作完成为止。对非活动分区进行编程时， CPU 可以无需停顿而继续工作。以下几节概述了发生 CPU 停顿和不发生停顿的编程算法。

4.4.1 擦除程序存储器 （活动分区）

1. 通过将 NVMOPx 位 （NVMCOM<3:0>）设置为 `0011` 来配置为进行页擦除操作，并将 WREN 位 （NVMCOM<14>）置 1。
2. 将要擦除块的起始地址写入 NVMADRL/H 寄存器。
3. 禁止中断。
4. 将 55h 写入 NVMKEY。
5. 将 AAh 写入 NVMKEY。
6. 通过将 WR 位 （NVMCOM <15>）置 1 来启动擦除周期。
7. 执行两条 `NOP` 指令。

当擦除操作完成时， WR 位会被自动清零。

4.4.2 行编程 （活动分区，标准分区模式）

用户可以一次对闪存的一行进行编程。要实现该操作，必须擦除包含该行在内的页。

对活动分区进行行编程的一般过程为：

1. 读取程序存储器的 8 行 （512 条指令），并存储在数据 RAM 中。
2. 用所需的新数据更新 RAM 中的程序数据。
3. 擦除该区块：
   a) 通过将 NVMOPx 位 （NVMCOM<3:0>）设置为 `0011` 来配置为进行页擦除操作，并将 WREN 位 （NVMCOM<14>）置 1。
   b) 将要擦除块的起始地址写入 NVMADRL/H 寄存器。
   c) 将 55h 写入 NVMKEY。
   d) 将 AAh 写入 NVMKEY。
   e) 将 WR 位 （NVMCOM <15>）置 1。擦除周期开始， CPU 暂停，等待擦除周期完成。当擦除操作完成时， WR 位会被自动清零。
4. 将数据 RAM 中的前 64 条指令写入程序存储器缓冲区 （见**第 4.2.7 节 “ 表写保持锁存器 ”**）或向 NVMSRCADR 寄存器写入 RAM 中存储数据的起始地址。
5. 将程序块写入闪存：
   a) 通过将 NVMOPx 位设置为 `0010` 来配置为行编程操作，并将 WREN 位置 1。
   b) 将 55h 写入 NVMKEY。
   c) 将 AAh 写入 NVMKEY。
   d) 将 WR 位置 1。编程周期开始，CPU 暂停，等待写周期完成。写入闪存结束时，WR 位会自动清零。
6. 递增 NVMADRL/H 中的地址，使用数据 RAM 中下一个 64 条指令的块重复步骤 4 和 5，直到将所有 512 条指令写回闪存。

© 2014-2015 Microchip Technology Inc.

> **注：** 在写周期或擦除周期期间，并非所有器件都会发生 CPU 停顿。要避免停顿，建议避免让应用程序读取或写入正在擦除或写入的行。

为防止意外操作，在执行任何擦除或编程操作之前，需要先向 NVMKEY 写入启动序列。在执行编程命令后，用户必须等待一段编程时间，直到编程操作完成。编程序列开始后紧跟的两条指令应该为 NOP 指令。

> **注 1：** 行、块和保持锁存器的数目可能随器件不同而有所不同；实际数目以及闪存编程的完整参考代码，请参见具体器件的数据手册。
>
> **2：** 对于具有单个保持锁存器的器件，必须通过字编程来写入闪存程序存储器。

4.4.3 编程非活动分区中的页 （双分区模式）

在双分区模式下编程需要注意一些特殊事项。由于对非活动分区进行编程时，CPU 能够继续执行指令，所以不会发生 CPU 停顿。

在双分区模式之一下擦除和重新编程一个数据页的算法如下：

1. 擦除该区块：
   a) 将 NVMOPx 位 （NVMCOM<3:0>）设置为 0011 以配置页擦除。
   b) 将 WREN 位 （NVMCOM<14>）置 1。
   c) 将要擦除块的起始地址写入 NVMADR 寄存器，作为页地址。
   d) 将 55h 写入 NVMKEY。
   e) 将 AAh 写入 NVMKEY。
   f) 将 WR 位 （NVMCOM <15>）置 1。擦除周期开始， CPU 将保持运行。
   g) 当擦除操作完成时，WR 位将自动清零，NVM 写完成中断标志（NVMIF）将会出现。
2. 通过填充 RAM 缓冲区来准备要编程的数据；或者，使用 TBLWT 指令向锁存器中装入存储器第一行的数据 （64 条指令）。
3. 编程该区块：
   a) 将 NVMOPx 位 （NVMCON<3:0>）设置为 0010，以配置为行编程操作。
   b) 将 WREN 位 （NVMCON<14>）置 1。
   c) 将要写入块的起始地址写入 NVMADR 寄存器，作为行起始地址。
   d) 将 55h 写入 NVMKEY。
   e) 将 AAh 写入 NVMKEY。
   f) 将 WR 位 （NVMCOM <15>）置 1。写周期开始， CPU 将保持运行。
   g) 当擦除操作完成时，WR 位将自动清零，NVM 写完成中断标志（NVMIF）将会出现。
4. 重复步骤 2 和步骤 3 来编程已擦除页中剩余的每个数据行。

4.4.4 编程整个非活动分区 （双分区模式）

完全更新非活动分区中的代码：

1. 擦除非活动分区：
   a) 将 NVMOPx 位 （NVMCOM<3:0>）设置为 0100 以配置非活动分区擦除。
   b) 将 WREN 位 （NVMCOM<14>）置 1。
   c) 将 55h 写入 NVMKEY。
   d) 将 AAh 写入 NVMKEY。
   e) 将 WR 位（NVMCOM <15>）置 1。擦除周期开始，CPU 将在该周期期间保持运行。当擦除操作完成时，WR 位将自动清零，NVM 写完成中断标志（NVMIF）将会出现。
2. 使用页写操作写入非活动分区的每一页，如**第 4.4.3 节 “ 编程非活动分区中的页 （双分区模式） ”** 所述。
3. 校验写入的数据。一种建议方法是对要写入的数据执行 CRC，并校验完整分区上的 CRC 值，以确保数据被正确写入。

4.4.5 使用自举程序更新活动分区

1. 擦除并编程整个非活动分区，如**第 4.4.4 节 “ 编程整个非活动分区 （双分区模式） ”** 所述。
2. 读取活动分区的 FBTSEQ 配置寄存器。
3. 将值减 1 并写入非活动分区的 FBTSEQ。
4. 强制分区交换：
   a) 如果不需要考虑 CPU 停顿的问题，则执行器件复位。由于非活动分区具有较小的引导序列号，它将在复位之后变为活动分区。
   b) 如果 CPU 停顿是不可接受的，则执行 BOOTSWP 指令。

© 2014-2015 Microchip Technology Inc.

5.0 程序空间可视性和扩展数据空间 （PSV 和 EDS）

对于所有 dsPIC33 和 PIC24 器件，可以使用表指令（见**第 4.2 节 " 表操作指令 "**）来访问程序存储空间中的数据。在一次只需读取或写入数据的一个字节或字时，这非常有用。此外，也可以将程序存储空间 16K 字的页映射到数据地址空间的高 32 KB。这可以有效地扩展空间数据，超出其常规的 64 KB 寻址限制，以及在无需使用表指令的情况下进行透明访问。

所有 dsPIC33 和 PIC24 器件都能够将所实现程序存储空间中的任意页映射到数据空间中。这种功能称为程序空间可视性 （PSV）。

一些器件通过以存储器映射的方式将一些特定外设映射到特定的虚拟程序存储器页范围来扩展 PSV。该功能对于具有高数据吞吐率要求的外设特别有用，例如高级图形控制器。PSV 的这种扩展称为扩展数据空间 （EDS）。

PSV 和 EDS 是作为数据存储器的功能实现的。对于 dsPIC33 和 PIC24 器件，它们的实现方式不同。关于详细说明，请参见 《dsPIC33/PIC24 系列参考手册》中的 **" 数据存储器 "** （dsPIC33，DS70595）和 / 或 **" 数据存储器 "** （PIC24， DS39717）。

5.1 PSV 和指令停顿

关于使用 PSV 产生的指令停顿的更多信息，请参见 《dsPIC33/PIC24 系列参考手册》中的 **"dsPIC33E 增强型 CPU"** （DS70005158）。

6.0 寄存器映射

表 6-1 汇总了与双分区闪存程序存储器相关联的 SFR。

表 6-1：与闪存程序存储器相关的特殊功能寄存器 (1)

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	所有复位时的值 (2)
TBLPAG	—	—	—	—	—	—	—	—	表页地址指针								0000
NVMCON	WR	WREN	WRERR	NVMPIDL	SFTSWP	P2ACTIV	RPDF	URERR	—	—	—	—	NVMOP3	NVMOP2	NVMOP1	NVMOP0	0000
NVMKEY	—	—	—	—	—	—	—	—	NVMKEY<7:0>								0000
NVMSRCADRL	数据 RAM 编程缓冲区起始地址																0000
NVMSRCADRH	数据 RAM 编程缓冲区地址（仅适用于 EDS 操作）																0000
NVMADRL	闪存程序存储器目标地址，低字节（ADDR<15:0>）																0000
NVMADRH	—	—	—	—	—	—	—	—	闪存程序存储器目标地址，高字节（ADDR<23:16>）								0000

图注： — = 未实现，读为 0。复位值以十六进制显示。

注 1： 关于具体存储器映射的详细信息，请参见器件数据手册。

2： 显示的复位值仅适用于 POR。其他复位状态下的值取决于复位时对存储器执行写或擦除操作的状态。

© 2014-2015 Microchip Technology Inc.

7.0 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PIC24或dsPIC33产品系列而编写的，但其概念是相近的，通过适当修改并受到一定限制即可使用。

当前与双分区闪存程序存储器相关的应用笔记有：

标题	应用笔记编号
目前没有相关的应用笔记。	N/A

注： 如需获取更多PIC24和dsPIC33系列器件的应用笔记和代码示例，请访问Microchip网站（www.microchip.com）。

8.0 版本历史

版本 A （2014 年 3 月）

本文档的初始版本。

版本 B （2015 年 2 月）

将标题和正文中的“双引导闪存程序存储器”全局修改为“双分区闪存程序存储器”或“双分区闪存”。

© 2014-2015 Microchip Technology Inc.

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指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，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。这样做的人极可能侵犯了知识产权。
- 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。

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc. 及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc. 的英文原版文档。

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。**Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。** Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和 / 或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

Microchip 的名称和徽标组合、 Microchip 徽标、 dsPIC、 FlashFlex、 flexPWR、 JukeBlox、 KEELOQ、 KEELOQ 徽标、 Kleer、 LANCheck、 MediaLB、 MOST、 MOST 徽标、 MPLAB、 OptoLyzer、 PIC、 PICSTART、 PIC32 徽标、 RightTouch、 SpyNIC、 SST、 SST 徽标、 SuperFlash 及 UNI/O 均为 Microchip Technology Inc. 在美国和其他国家或地区的注册商标。

The Embedded Control Solutions Company 和 mTouch 为 Microchip Technology Inc. 在美国的注册商标。

Analog-for-the-Digital Age、BodyCom、chipKIT、chipKIT 徽标、 CodeGuard、 dsPICDEM、 dsPICDEM.net、 ECAN、 In-Circuit Serial Programming、 ICSP、 Inter-Chip Connectivity、 KleerNet、 KleerNet 徽标、 MiWi、 MPASM、 MPF、 MPLAB Certified 徽标、 MPLIB、 MPLINK、 MultiTRAK、 NetDetach、 Omniscient Code Generation、 PICDEM、 PICDEM.net、 PICkit、 PICtail、 RightTouch 徽标、 REAL ICE、 SQI、 Serial Quad I/O、 Total Endurance、 TSHARC、 USBCheck、 VariSense、 ViewSpan、 WiperLock、 Wireless DNA 和 ZENA 均为 Microchip Technology Inc. 在美国和其他国家或地区的商标。

SQTP 为 Microchip Technology Inc. 在美国的服务标记。

Silicon Storage Technology 为 Microchip Technology Inc. 在除美国外的国家或地区的注册商标。

GestIC 为 Microchip Technology Inc. 的子公司 Microchip Technology Germany II GmbH & Co. & KG 在除美国外的国家或地区的注册商标。

在此提及的所有其他商标均为各持有公司所有。

© 2014-2015, Microchip Technology Inc. 版权所有。

ISBN：978-1-63277-432-3

QUALITY MANAGEMENT SYSTEM CERTIFIED BY DNV

═ ISO/TS 16949 ═

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2009 认证。 Microchip 的 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品严格遵守公司的质量体系流程。此外， Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

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

克里夫兰 Cleveland
Independence, OH
Tel: 1-216-447-0464
Fax: 1-216-447-0643

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

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

纽约 New York, NY
Tel: 1-631-435-6000

圣何塞 San Jose, CA
Tel: 1-408-735-9110

加拿大多伦多 Toronto
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

中国 - 北京
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

中国 - 成都
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

中国 - 重庆
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

中国 - 东莞
Tel: 86-769-8702-9880

中国 - 杭州
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

中国 - 香港特别行政区
Tel: 852-2943-5100
Fax: 852-2401-3431

中国 - 南京
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

中国 - 青岛
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

中国 - 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 - 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 - 深圳
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

中国 - 武汉
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

中国 - 西安
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

中国 - 厦门
Tel: 86-592-238-8138
Fax: 86-592-238-8130

中国 - 珠海
Tel: 86-756-321-0040
Fax: 86-756-321-0049

亚太地区

台湾地区 - 高雄
Tel: 886-7-213-7828

台湾地区 - 台北
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

台湾地区 - 新竹
Tel: 886-3-5778-366
Fax: 886-3-5770-955

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

印度 India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

印度 India - Pune
Tel: 91-20-3019-1500

日本 Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

日本 Japan - Tokyo
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

韩国 Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

韩国 Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

菲律宾 Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

新加坡 Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

泰国 Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

欧洲

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

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

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

德国 Germany - Dusseldorf
Tel: 49-2129-3766400

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

德国 Germany - Pforzheim
Tel: 49-7231-424750

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

意大利 Italy - Venice
Tel: 39-049-7625286

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

波兰 Poland - Warsaw
Tel: 48-22-3325737

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

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

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

01/27/15

© 2014-2015 Microchip Technology Inc.