第 52 章 支持实时更新的闪存

目录

手册的本章包括下列主题：

注：	本系列参考手册章节旨在作为器件数据手册的补充资料，并非适用于所有的 PIC32 器件，适用与否取决于具体的器件型号。 请查询具体器件数据手册 **“闪存程序存储器”** 章节开始处的注释，以查看本文档是否支持您当前使用的器件。 器件数据手册和系列参考手册的章节可从 Microchip 网站下载：http://www.microchip.com。

52.1 简介

本文档描述了在支持在线更新的 PIC32 器件上对闪存进行编程的技术。这些器件包含两个闪存存储区，每个存储区都具有自己的引导闪存（Boot Flash Memory，BFM）分区和闪存程序存储器（Program Flash Memory，PFM）分区，以便存储用户代码或非易失性数据。双存储区功能可使闪存在一个分区中编程，而在另一个分区中执行程序存储器的实时更新。用户可采用三种方法编程该存储器。

- 运行时自编程（Run-Time Self-Programming，RTSP）—— 由用户软件执行
- 在线串行编程（In-Circuit Serial Programming，ICSP™）—— 使用器件的串行数据连接执行，编程速度比 RTSP 快得多
- 增强型联合测试小组编程（Enhanced Joint Test Action Group Programming，EJTAG）—— 使用器件的 EJTAG 端口，通过支持 EJTAG 的编程器执行

本章将介绍 RTSP 技术。《PIC32 闪存编程规范》（DS60001145P_CN）中介绍了 ICSP 和 EJTAG 方法，该文档可从 Microchip 网站（www.microchip.com）下载。

52.1.1 运行时自编程（RTSP）

RTSP 用于需要现场升级固件或非易失性数据存储器的应用。实现现场升级固件功能的软件称为自举程序。闪存中的非易失性数据存储器通常使用 EEPROM 模拟软件实现，包含延长用于数据存储的存储器使用寿命的耗损均衡。所有这些技术的完整示例代码可从 Microchip 网站（www.microchip.com）下载。

52.1.2 双存储区

某些 PIC32 器件提供的双存储区功能使自举程序和 EEPROM 模拟软件获得了显著优势。双闪存存储区可以实现在一个存储区执行代码，而同时擦除或编程另一个存储区，避免了在编程操作时停止 CPU。闪存存储区可以是存储器的别名或映射到存储器中，并选择在启动时自动或手动执行，从而保证应用程序和自举程序软件现场更新的高可靠性。还在一些 PIC32 器件上集成了错误校正代码（Error-Correcting Code，ECC）存储器，用于延长闪存的使用寿命。

52.1.2.1 闪存分区

闪存的每个存储区都分为两个逻辑闪存分区：PFM 和 BFM（一些 PIC32 器件具有两个 BFM 分区）。两个 BFM 分区在启动时指定为这两个存储区的别名。别名的顺序在启动时由每个存储区中指定配置字中存储的闪存序列代码（Flash Sequence Code，FSEQ）决定。该顺序决定 BFM 存储器中的哪些分区将用作器件复位时的启动代码。每个存储区中的 PFM 分区映射到存储器映射的较高和较低区域。从 BFM 执行的启动代码可选择将两个 PFM 分区映射到两个 PFM 区域中。有关闪存的器件存储器映射和可用选项的详细信息，请参见具体器件数据手册中的 **“存储器构成”** 章节。

52.1.3 寻址

PIC32 器件实现两种地址方案：虚拟和物理。虚拟地址专供 CPU 取指、执行指令和访问外设使用。当编程或擦除闪存时，物理地址通常为目标操作的地址。

BFM 上的代码保护由页实现，在复位时使能，且必须在编程任何 BFM 之前禁止。PFM 的代码保护使用水印寄存器实现，在复位时禁止，且初始化时必须在启动代码中配置，以避免意外编程或擦除 PFM。

52.2 控制寄存器

闪存的编程、擦除和写保护操作使用以下非易失性存储器 （Non-Volatile Memory， NVM）控制寄存器控制：

- **NVMCON：编程控制寄存器**

  NVMCON 寄存器是闪存编程 / 擦除操作的控制寄存器。该寄存器用于选择要执行的操作、启动操作，并在操作完成时提供结果状态。

- **NVMKEY：编程解锁寄存器**

  NVMKEY 是用于实现解锁序列的只写寄存器，有助于防止闪存或 EEPROM 存储器的意外写 / 擦除操作或意外修改 NVMSWAP 和写许可设置。

- **NVMADDR：闪存地址寄存器**

  该寄存器用于存储行、四字和字编程以及页擦除的物理目标地址。

- **NVMDATAx：闪存数据寄存器 （x = 0-3）**

  这些寄存器保存闪存字编程操作期间要编程的数据。 NVMDATA3 至 NVMDATA0 用于四字（128 位）编程，只有 NVMDATA0 用于字 （32 位）编程。

- **NVMSRCADDR：源数据地址寄存器**

  当执行行编程操作时，该寄存器用于指向要编程数据的物理地址。

- **NVMPWP：程序闪存写保护寄存器**

  该寄存器用于设置闪存程序存储器分区中最后的写保护页的边界。

- **NVMBWP：闪存引导 （页）写保护寄存器**

  该寄存器用于配置哪些引导闪存分区页受写保护。

- **NVMCON2：闪存编程控制寄存器 2**

  该寄存器用于控制在运行时可否交换引导和闪存程序存储器区域。（见**注**）。

注：	该寄存器不是在所有器件上均可用。请参见具体器件数据手册的**“闪存程序存储器”**章节，以确定可用性。

表 52-1 简要汇总了所有与闪存编程相关的寄存器。该汇总表之后列出了相应的寄存器，并附有详细的说明。

表 52-1： 闪存控制器 SFR 汇总

寄存器名称	位范围	Bit 31/15	Bit 30/14	Bit 29/13	Bit 28/12	Bit 27/11	Bit 26/10	Bit 25/9	Bit 24/8	Bit 23/7	Bit 22/6	Bit 21/5	Bit 20/4	Bit 19/3	Bit 118/2	Bit 17/1	Bit 16/0
NVMCON(1)	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	15:0	WR	WREN	WRERR	LVDERR	—	—	—	—	SWAP(2) PFSWAP(2)	BFSWAP(2)	—	—	NVMOP<3:0>			
NVMKEY	31:16	NVMKEY<31:16>															
	15:0	NVMKEY<15:0>															
NVMADDR(1)	31:16	NVMADDR<31:16>															
	15:0	NVMADDR<15:0>															
NVMDATA3(1)	31:16	NVMDATA<31:16>															
	15:0	NVMDATA<15:0>															
NVMDATA2(1)	31:16	NVMDATA<31:16>															
	15:0	NVMDATA<15:0>															
NVMDATA1(1)	31:16	NVMDATA<31:16>															
	15:0	NVMDATA<15:0>															
NVMDATA0(1)	31:16	NVMDATA<31:16>															
	15:0	NVMDATA<15:0>															
NVM SRCADDR(1)	31:16	NVMSRCADDR<31:16>															
	15:0	NVMSRCADDR<15:0>															
NVMPWP(1)	31:16	PWPULOCK	—	—	—	—	—	—	—	PWP<23:16>							
	15:0	PWP<15:0>															
NVMBWP(1)	31:16	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	15:0	LBWPULOCK	—	—	LBWP4	LBWP3	LBWP2	LBWP1	LBWP0	UBWPULOCK(2)	—	—	UBWP4(2)	UBWP3(2)	UBWP2(2)	UBWP1(2)	UBWP0(2)
NVMCON2(1,2)	31:16	NVMERS<3:0>				—	—	—	—	—	—	—	NVMWS<4:0>				
	15:0	NVMLPRD	—	NVMCRD	NVMVRD	—	—	NVMRETRY<1:0>		SWAPLOCK<1:0>		—	—	—	—	—	—

图注： — = 未实现，读为 0。

注 1： 这些寄存器具有关联的清零、置 1 和取反寄存器，偏移量分别为 0x4、0x8 和 0xC 字节。这些寄存器的命名方式是在关联寄存器的名称末尾附加 CLR、SET 或 INV（例如，NVMCONCLR）。向这些寄存器的任意位位置写入 1 时，会将关联寄存器中的有效位清零、置 1 或取反。对这些寄存器的读操作将被忽略。

2： 该位或寄存器不是在所有器件上均可用。请参见具体器件数据手册的 **“闪存程序存储器”**章节，以确定可用性。

寄存器 52-1： NVMCON：编程控制寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-0, HC	R/W-0	R-0, HS, HC	R-0, HS, HC	U-0	U-0	U-0	U-0
	WR(2)	WREN	WRERR	LVDERR(2)	—	—	—	—
7:0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
	SWAP(3)	BFSWAP(3)	—	—	NVMOP<3:0>			
	PFSWAP(3)							

图注：			
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 31-16 **未实现：**读为 0

bit 15 **WR：**写控制位 (2)

该位不可清零，仅当 WREN = 1 并执行解锁序列时可置 1。
1 = 启动闪存操作
0 = 闪存操作已完成或无效

bit 14 **WREN：**写使能位

1 = 使能对 WR 位的写操作，并禁止对 SWAP 或 PFSWAP 位以及 BFSWAP 和 NVMOP<3:0> 位的写操作
0 = 禁止对 WR 位的写操作，并使能对 SWAP 或 PFSWAP 位以及 BFSWAP 和 NVMOP<3:0> 位的写操作

bit 13 **WRERR：**写错误位 (2)

该位只能通过设置 NVMOP<3:0> 位 = 0000 （NOP）并启动闪存操作清零。
1 = 编程或擦除序列未成功完成
0 = 编程或擦除序列正常完成

bit 12 **LVDERR：**低电压检测错误位 (2)

该位只能通过设置 NVMOP<3:0> 位 = 0000 （NOP）并启动闪存操作清零。
1 = 编程或擦除操作时检测到低电压条件 （如果 WRERR 置 1，则数据可能损坏）
0 = 编程或擦除操作时未发生低电压条件

bit 11-8 **未实现：**读为 0

注 1： 当动态闪存 ECC 配置位 （FECCCON<1:0> （DVCFG0<9:8>）） = 00 时，将始终使能 ECC，导致该操作为 “空操作”（NOP）。对于所有其他 FECCCON<1:0> 位设置，该命令将执行，但是不会写该字的 ECC 位，并在使能动态闪存 ECC （FECCCON<1:0> = 01）时会导致 DED 错误。

2： 该位仅在上电复位 （Power-on reset， POR）时复位，且不受其他复位源的影响。

3： 该位不是在所有器件上均可用。请参见具体器件数据手册的 **“闪存程序存储器”**章节，以确定可用性。

寄存器 52-1： NVMCON：编程控制寄存器 （续）

bit 7 **SWAP**：闪存程序存储区交换控制位 (3)

只有在 WREN = 0 且已执行解锁序列时，该位才可写。
1 = 闪存程序存储区 2 映射到较低映射区域，而闪存程序存储区 1 映射到较高映射区域
0 = 闪存程序存储区 1 映射到较低映射区域，而闪存程序存储区 2 映射到较高映射区域

PFSWAP：闪存程序存储区交换控制位 (3)

只有在 WREN = 0 和 SWAPLOCK = 00 且已执行解锁序列时，该位才可写。
1 = 闪存程序存储区 2 映射到较低映射区域，而闪存程序存储区 1 映射到较高映射区域
0 = 闪存程序存储区 1 映射到较低映射区域，而闪存程序存储区 2 映射到较高映射区域

bit 6 **BFSWAP**：引导闪存存储区别名交换控制位 (3)

只有在 WREN = 0 和 SWAPLOCK = 00 且已执行解锁序列时，该位才可写。
BFSWAP 的复位值通过比较每个引导闪存存储区中用户编程的 BFxSEQ0 字的值确定。
1 = 引导闪存存储区 2 映射到较低引导别名，而引导闪存存储区 1 映射到较高引导别名
0 = 引导闪存存储区 1 映射到较低引导别名，而引导闪存存储区 2 映射到较高引导别名

bit 5-4 **未实现**：读为 0

bit 3-0 **NVMOP<3:0>**：NVM 操作位 (2)

仅当 WREN = 0 时，这些位才可写。
1111 = 保留
•
•
•
1000 = 保留
0111 = 设置擦除操作：擦除所有闪存程序存储器 （所有页必须均无保护， PWP<23:0> = 0x000000）
0110 = 高闪存程序存储器擦除操作：仅擦除闪存程序存储器的较高映射区域 （该区域中的所有页必须均无保护）
0101 = 低闪存程序存储器擦除操作：仅擦除闪存程序存储器的较低映射区域 （该区域中的所有页必须均无保护）
0100 = 页擦除操作：擦除通过 NVMADDR 选择的页 （如果无写保护）
0011 = 行编程操作：对通过 NVMADDR 选择的行进行编程 （如果无写保护）
0010 = 四字 （128 位）编程操作：对通过 NVMADDR 选择的 128 位闪存字进行编程 （如果无写保护）
0001 = 字编程操作：对通过 NVMADDR 选择的字进行编程 （如果无写保护） (1)
0000 = 空操作

注 1： 当动态闪存 ECC 配置位 （FECCCON<1:0> （DVCFG0<9:8>）） = 00 时，将始终使能 ECC，导致该操作为“空操作”（NOP）。对于所有其他 FECCCON<1:0> 位设置，该命令将执行，但是不会写该字的 ECC 位，并在使能动态闪存 ECC （FECCCON<1:0> = 01）时会导致 DED 错误。

2： 该位仅在上电复位 （Power-on reset， POR）时复位，且不受其他复位源的影响。

3： 该位不是在所有器件上均可用。请参见具体器件数据手册的 **“闪存程序存储器”** 章节，以确定可用性。

寄存器 52-2： NVMKEY：编程解锁寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
	NVMKEY<31:24>							
23:16	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
	NVMKEY<23:16>							
15:8	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
	NVMKEY<15:8>							
7:0	W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
	NVMKEY<7:0>							

图注：			
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 31-0 **NVMKEY<31:0>**：解锁寄存器位
这些位是只写位，在读取时读为 0。

注： 该寄存器用作解锁序列的一部分，以防止对 PFM 的意外写操作。

寄存器 52-3： NVMADDR：闪存地址寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMADDR<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMADDR<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMADDR<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMADDR<7:0>							

图注：			
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 31-0 **NVMADDR<31:0>：**闪存地址位

NVMOP<3:0> 选择（见注 1）	闪存地址位
页擦除	地址指定要擦除的页 （根据器件页的大小忽略较低的地址位）
行编程	地址指定要编程的行 （根据器件行的大小忽略较低的地址位）
字编程 （32 位）	地址指定要编程的字 （忽略 NVMADDR<1:0>）
四字编程 （128 位）	地址指定要编程的四字 （128 位）（忽略 NVMADDR<3:0>）

注 1： 对于所有其他 NVMOP<3:0> 位设置，忽略闪存地址。

注： 该寄存器中的这些位只能通过上电复位 （POR）进行复位，且不受其他复位源影响。

寄存器 52-4： NVMDATAx：闪存数据寄存器 （x = 0-3）

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMDATA<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMDATA<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMDATA<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMDATA<7:0>							

图注：			
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 31-0 **NVMDATA<31:0>**：闪存数据位
字编程：将 NVMDATA0 写入 NVMADDR 中定义的目标闪存地址
四字编程：将 NVMDATA3:NVMDATA2:NVMDATA1:NVMDATA0 写入 NVMADDR 中定义的目标闪存地址。NVMDATA0 包含最低有效指令字。

注： 该寄存器中的这些位只能通过上电复位 （POR）进行复位，且不受其他复位源影响。

寄存器 52-5： NVMSRCADDR：源数据地址寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMSRCADDR<31:24>							
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMSRCADDR<23:16>							
15:8	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMSRCADDR<15:8>							
7:0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	NVMSRCADDR<7:0>							

图注：			
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 31-0 **NVMSRCADDR<31:0>**：源数据地址位
当 NVMOP<3:0> 位 （NVMCON<3:0>）设置为执行行编程时，要编程至闪存的数据的系统物理地址。

注： 该寄存器中的这些位只能通过上电复位 （POR）进行复位，且不受其他复位源影响。

寄存器 52-6： NVMPWP：程序闪存写保护寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-1	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	PWPULOCK	—	—	—	—	—	—	—
23:16	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
	PWP<23:16>							
15:8	R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
	PWP<15:8>							
7:0	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
	PWP<7:0>							

图注：			
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 31 **PWPULOCK：**闪存程序存储器页写保护解锁位
1 = 寄存器未锁定，可修改
0 = 寄存器已锁定，不可修改
该位只可清零且只能通过任何复位进行置 1。

bit 30-24 **未实现：**读为 0

bit 23-0 **PWP<23:0>：**闪存程序存储器写保护 （页）地址位
地址 0x1D000000 至 0x1Dxxxxxx 的物理闪存程序存储器受写保护，其中 “xxxxxx” 由 PWP<23:0> 指定。当 PWP<23:0> 值为 0 时，禁止对整个闪存程序存储器写保护。如果指定地址在该页内，那么整个页和地址低于当前页的所有页将受保护。

注： 仅当随后执行 NVMKEY 解锁序列且 PWPULOCK 位置 1 时，才可写该寄存器。

寄存器 52-7： NVMBWP：闪存引导 （页）写保护寄存器

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
23:16	U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
	—	—	—	—	—	—	—	—
15:8	R/W-1	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	LBWPU LOCK	—	—	LBWP4(1)	LBWP3(1)	LBWP2(1)	LBWP1(1)	LBWP0(1)
7:0	R/W-1	r-1	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	UBWPU LOCK(2)	—	—	UBWP4(2,3)	UBWP3(2,3)	UBWP2(2,3)	UBWP1(2,3)	UBWP0(2,3)

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

bit 31-16 **未实现：**读为 0

bit 15 **LBWPULOCK：**低引导别名写保护解锁位
1= LBWPx 位未锁定，可修改
0= LBWPx 位已锁定，不可修改
该位只可清零且只能通过任何复位进行置 1。

bit 14-13 **未实现：**读为 0

bit 12 **LBWP4：**低引导别名 Page 4 写保护位 (1)
1 = 使能写保护
0 = 禁止写保护

bit 11 **LBWP3：**低引导别名 Page 3 写保护位 (1)
1 = 使能写保护
0 = 禁止写保护

bit 10 **LBWP2：**低引导别名 Page 2 写保护位 (1)
1 = 使能写保护
0 = 禁止写保护

bit 9 **LBWP1：**低引导别名 Page 1 写保护位 (1)
1 = 使能写保护
0 = 禁止写保护

bit 8 **LBWP0：**低引导别名 Page 0 写保护位 (1)
1 = 使能写保护
0 = 禁止写保护

bit 7 **UBWPULOCK：**高引导别名写保护解锁位 (3)
1= UBWPx 位未锁定，可修改
0= UBWPx 位已锁定，不可修改
该位只可由用户清零且只能通过任何复位进行置 1。

bit 6 **保留：**该位保留，供开发工具使用

注 1： 仅当随后执行 NVMKEY 解锁序列且 LBWPULOCK 位置 1 时，才可写该位。
2： 仅当随后执行 NVMKEY 解锁序列且 UBWPULOCK 位置 1 时，才可写该位。
3： 该位不是在所有器件上均可用。请参见具体器件数据手册的 **“闪存控制器”**章节，以确定可用性。

寄存器 52-7： NVMBWP：闪存引导 （页）写保护寄存器 （续）

bit 5 **未实现：**读为 0

bit 4 **UBWP4：**高引导别名 Page 4 写保护位 **(2,3)**
1 = 使能写保护
0 = 禁止写保护

bit 3 **UBWP3：**高引导别名 Page 3 写保护位 **(2,3)**
1 = 使能写保护
0 = 禁止写保护

bit 2 **UBWP2：**高引导别名 Page 2 写保护位 **(2,3)**
1 = 使能写保护
0 = 禁止写保护

bit 1 **UBWP1：**高引导别名 Page 1 写保护位 **(2,3)**
1 = 使能写保护
0 = 禁止写保护

bit 0 **UBWP0：**高引导别名 Page 0 写保护位 **(2,3)**
1 = 使能写保护
0 = 禁止写保护

注 1： 仅当随后执行 NVMKEY 解锁序列且 LBWPULOCK 位置 1 时，才可写该位。
2： 仅当随后执行 NVMKEY 解锁序列且 UBWPULOCK 位置 1 时，才可写该位。
3： 该位不是在所有器件上均可用。请参见具体器件数据手册的 **“闪存控制器”**章节，以确定可用性。

寄存器 52-8： NVMCON2：闪存编程控制寄存器 2

位范围	Bit 31/23/15/7	Bit 30/22/14/6	Bit 29/21/13/5	Bit 28/20/12/4	Bit 27/19/11/3	Bit 26/18/10/2	Bit 25/17/9/1	Bit 24/16/8/0
31:24	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
	NVMERS<3:0>				—	—	—	—
23:16	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
	—	—	—	NVMWS<4:0>				
15:8	R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
	NVMLPRD	—	NVMCRD	NVMVRD	—	—	NVMRETRY<1:0>	
7:0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0
	SWAPLOCK<1:0>		—	—	—	—	—	—

图注：	HS = 硬件置 1 位	HC = 硬件清零位	
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 31-28 **NVMERS<3:0>：**闪存存储器擦除重试状态位
软件使用这些位来跟踪发生系统复位 （即，主复位）或欠压复位事件时擦除重试序列的软件状态。硬件不使用这些位。

bit 27-21 **未实现：**读为 0

bit 20-16 **NVMWS<4:0>：**闪存存储器访问等待状态控制位
这些位在 NVMLPRD = 1 或 NVMVRD = 1 时有效，并且只有当满足 NVMKEY 解锁序列时才可修改。
11111 = 31 个等待状态 （总共 32 个 SYSCLK 周期）
11110 = 30 个等待状态 （总共 31 个 SYSCLK 周期）
•
•
•
00010 = 2 个等待状态 （总共 3 个 SYSCLK 周期）
00001 = 1 个等待状态 （总共 2 个 SYSCLK 周期）
00000 = 0 个等待状态 （总共 1 个 SYSCLK 周期）

bit 15 **NVMLPRD：**闪存存储器低功耗读控制位
当 NVMLPRD = 1 时，闪存等待状态控制由 NVMWS<4:0> 位处理。只有当 WR 位 （NVMCON<15>） = 0 且满足 NVMKEY 解锁序列时，才能修改这些位。
1 = 配置闪存用于低功耗读操作 （增加访问时间）
0 = 配置闪存用于低延时读取

bit 14 **未实现：**读为 0

bit 13 **NVMCRD：**闪存存储器逻辑 1 的比较读操作控制位
当在 ECC 闪存系统中使用擦除重试时，必须使能 NVMCRD。只有当满足 NVMKEY 解锁序列时，才能修改这些位。
1 = 使能比较读操作
0 = 禁止比较读操作

bit 12 **NVMVRD：**闪存存储器逻辑 1 的验证读操作控制位
当 NVMVRD = 1 时，对于包含 NVMADDR 的分区，闪存等待状态控制由 NVMWS<4:0> 位处理。只有当 WR 位 （NVMCON<15>） = 0 且满足 NVMKEY 解锁序列时，才能修改这些位。
1 = 选择具有验证读操作的擦除重试序列
0 = 选择不具有验证读操作的单次擦除

bit 11-10 **未实现：**读为 0

注： 该寄存器不是在所有器件上均可用。请参见具体器件数据手册的 **“闪存程序存储器”**章节，以确定可用性。

寄存器 52-8： NVMCON2：闪存编程控制寄存器 2 （续）

bit 9-8 **NVMRETRY<1:0>：**闪存存储器擦除重试控制位
只有当 WR 位 （NVMCON<15>） = 0 时，才能修改该位。
11 = 最后一个重试周期的擦除长度
10 = 第三个重试周期的擦除长度
01 = 最二个重试周期的擦除长度
00 = 第一个重试周期的擦除长度

bit 7-6 **SWAPLOCK<1:0>：**闪存存储器交换锁定控制位
11 = PFSWAP 和 BFSWAP 不可写， SWAPLOCK 不可写
10 = PFSWAP 和 BFSWAP 不可写， SWAPLOCK 可写
01 = PFSWAP 和 BFSWAP 不可写， SWAPLOCK 可写
00 = PFSWAP 和 BFSWAP 可写， SWAPLOCK 可写

bit 5-0 **未实现：**读为 0

注： 该寄存器不是在所有器件上均可用。请参见具体器件数据手册的 **“闪存程序存储器”**章节，以确定可用性。

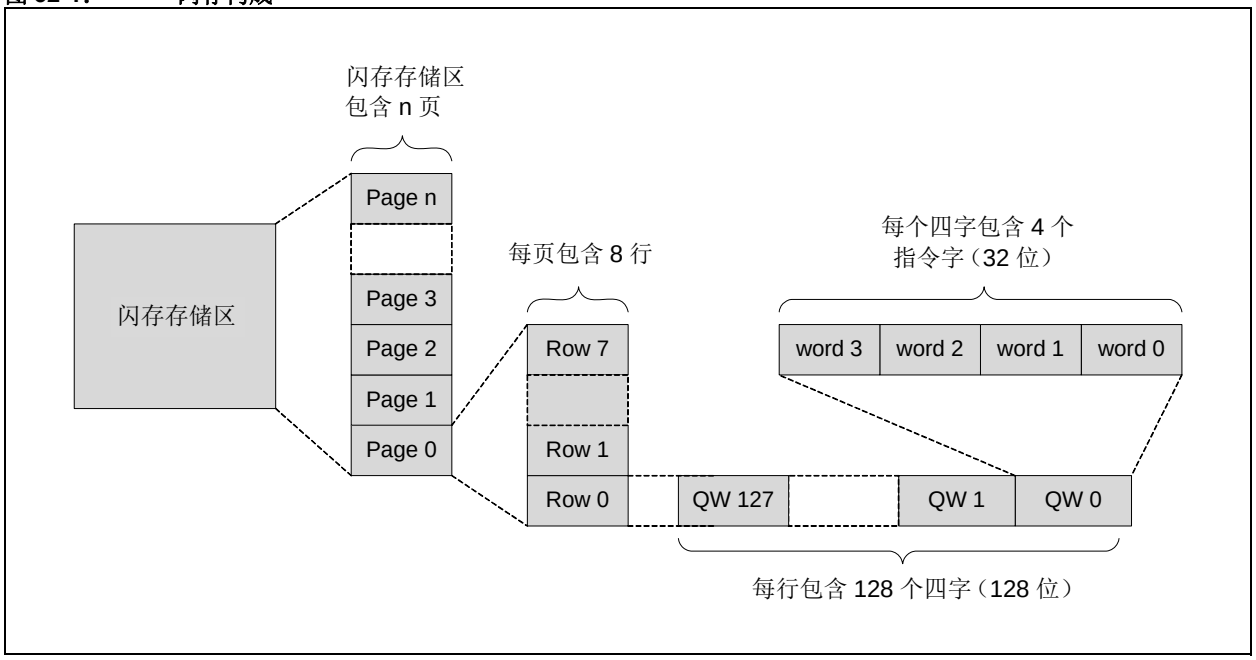
52.3 存储器配置

52.3.1 闪存存储区构成

每个闪存存储区都分为页。页是一次可擦除的最小存储单元。存储器的每页划分为八行。行是一次可编程的最大存储单元。一行由 128 个四字（128 位）组成。每个四字由四个指令字（32 位）组成。闪存存储器可以行、四字 （128 位）或字 （32 位）为单位编程。

注：	页大小因器件而异。请参见具体器件数据手册中的 **“闪存程序存储器”**章节以确定器件的闪存页大小。

图 52-1： 闪存构成



52.3.2 双闪存存储区

在一些 PIC32 器件上，闪存存储器分为两个大小相等的存储区，每个都有 BFM 分区和 PFM 分区。 BFM 和 PFM 分区都可以页递增方式擦除。 PFM 分区可使用一条命令整体擦除。

52.3.3 编程或擦除闪存中执行代码的同一存储区

CPU 不能从作为编程操作目标的同一个闪存存储区中取代码，这个存储区可能是 BFM 或 PFM。当尝试该操作时，CPU 在编程操作进行时将停止（暂停）代码执行。还将包含中断服务程序（及其向量），前提是它们与闪存操作目标位于同一存储区。如果系统软件需要在闪存操作期间执行代码，代码必须存储在不是编程操作目标的系统 RAM 或闪存存储区中。

注：	当启动编程操作时，将继续执行已存储在高速缓存中的指令代码。

52.3.4 在执行代码以外的存储区进行编程或擦除闪存操作

为避免编程操作时 CPU 停止，应保证编程操作的所有目标地址所在的闪存存储区单元中没有正在读取的可执行代码。满足该要求时， CPU 可在任何编程操作期间读取和执行代码 （包含中断服务程序），而不用停止。要实现此操作，应用程序必须将所有可执行代码放在一个存储区的 BFM 和 PFM 分区中，并在另一存储区的 BFM 和 PFM 分区上执行编程操作。

52.4 引导闪存存储器（BFM）分区

52.4.1 BFM 存储区别名

每个闪存存储区的 BFM 分区映射到固定区域和别名区域。映射到别名区域的顺序由器件配置存储器字 BFxSEQ0（不使用 BFxSEQ1-BFxSEQ3）决定。复位时，映射发生在所有指令代码执行之前。在相关的序列字寄存器中拥有较大值的 BFM，将映射至较低的别名区域。如果值相等，Bank 1 将映射至较低的别名区域， Bank 2 和其他 BFM 映射至较高的别名区域。

编程 BFxSEQ0 值时，序列数值存储在低 16 位中，且该值的反码将保存在高 16 位中。举例来说，序列值为 3 时， BFxSEQ0 值为 0xFFFC0003。

图 52-2 说明了 BFxSEQ0 比较操作的影响。不论发生何种情况， Bank 1 和 Bank 2 同样映射到固定的区域， Bank 1 位于较低的固定区域， Bank 2 位于较高的固定区域。别名区域的映射根据 BFxSEQ0 值而变化。该示例展示了具有 80 KB BFM 存储区的器件。

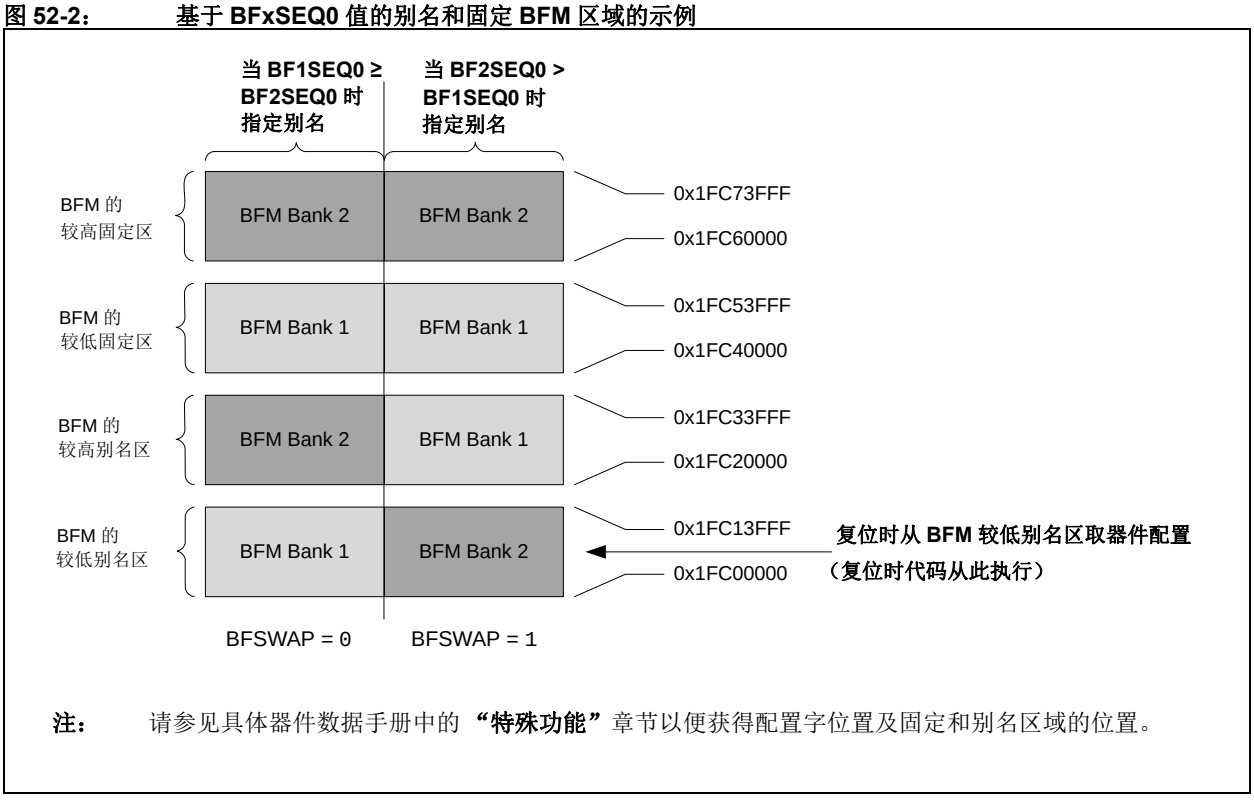
注：	请参见具体器件数据手册中的 **“存储器构成”** 章节，以确定您器件上 BFM 存储区的大小和地址映射。

引导闪存在别名区域中的映射由 BFSWAP 位 （NVMCON<6>）指定。在启动时该位是置 1 还是清零取决于复位时发生的映射。也可在运行时由软件控制该位。请注意，在更改 BFSWAP 之前，必须清零 SWAPLOCK 位 （NVMCON2<7:6>）和 WREN 位 （NVMCON<14>），并且执行 NVMKEY 解锁序列。建议只通过在 BFM 区域执行的代码更改该位。

对于具有 NVMCON2 寄存器的器件，可以通过将 SWAPLOCK 位（NVMCON2<7:6>）设置为除 00 外的任意值来锁定对于引导闪存分区交换的控制 （见**注**）。

注：	请参见具体器件数据手册中的 **“闪存程序存储器”** 章节，以确定 NVMCON2 寄存器的可用性。

图 52-2： 基于 BFxSEQ0 值的别名和固定 BFM 区域的示例



注： 请参见具体器件数据手册中的 **“特殊功能”** 章节以便获得配置字位置及固定和别名区域的位置。

52.4.2 BFM 写保护

在较高和较低别名区域中的页可以使用 NVMBWP 寄存器中的位单独保护。由于这些位对应别名区域引用的页，它们在启动时会受到别名映射的影响。复位时，所有页都处于写保护状态，且必须在 BFM 区域上执行任何编程操作之前禁止。还存在高区域和低区域解锁位（UBWPULOCK（NVMBWP<7>）和 LBWPULOCK（NVMBWP<15>）），这两位可以在复位时置 1，并使用用户软件清零。清零时，不能对此区域中的写保护进行修改。一旦清零，UBWPULOCK 和 LBWPULOCK 位只能通过复位置 1。

只有在随后执行解锁序列时，才能更改 NVMBWP 写保护寄存器。更多信息，请参见**第 52.9 节“NVMKEY 寄存器解锁序列”**。

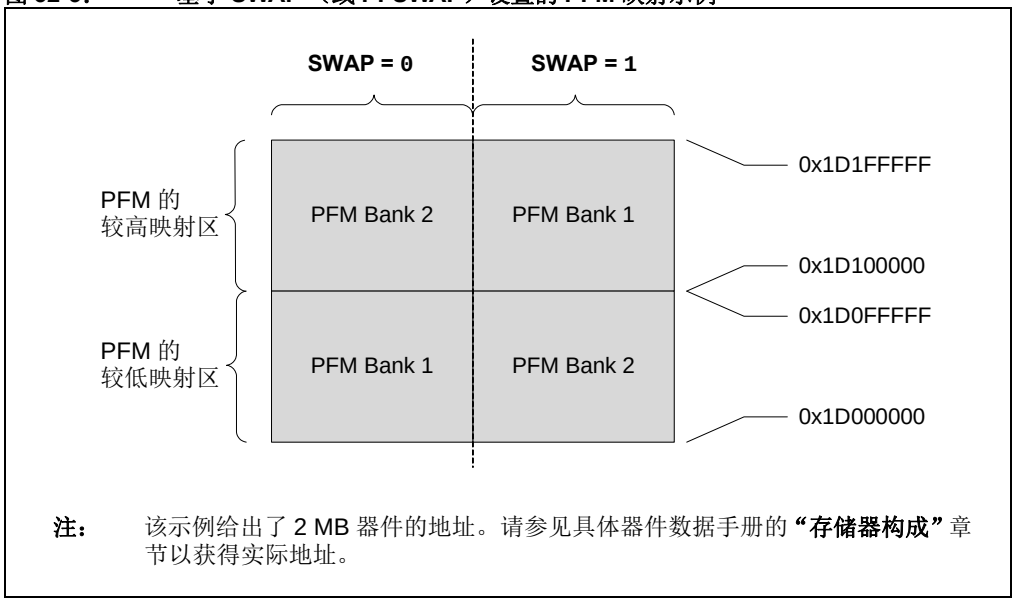
52.5 闪存程序存储器（PFM）分区

52.5.1 PFM 映射至地址空间

每个闪存存储区都有一个相同大小的 PFM 区域。 PFM 存储区至地址空间的映射由 SWAP （或 PFSWAP）控制位（NVMCON<7>）的状态决定，如图 52-3 所示（见**注**）。建议只能通过在 BFM 区域执行的代码更改该位。

注：	请参见具体器件数据手册中的**“闪存程序存储器”**章节，以确定您器件上的是 SWAP 位还是 PFSWAP 位。

图 52-3： 基于 SWAP （或 PFSWAP）设置的 PFM 映射示例



注： 该示例给出了 2 MB 器件的地址。请参见具体器件数据手册的**“存储器构成”**章节以获得实际地址。

当解锁序列首次完成时，才能更改 SWAP （或 PFSWAP）位设置。更多信息，请参见**第 52.9 节“NVMKEY 寄存器解锁序列”**。

对于具有 NVMCON2 寄存器的器件，可以通过将 SWAPLOCK 位（NVMCON2<7:6>）设置为除 00 外的任意值来锁定对于引导闪存分区交换的控制 （见**注**）。

注：	请参见具体器件数据手册中的**“闪存程序存储器”**章节，以确定 NVMCON2 寄存器的可用性。

52.5.2 PFM 写保护

PFM 区域的写保护由使用水印地址的页实现，具体取决于 NVMPWP<23:0> 位。写保护通常在页边界实现；因此，对于一个具有 0x4000 （16 KB）页大小的器件，忽略地址位 bit 0-13，且始终读为 0。当 NVMPWP 等于 0 时，无 PFM 页受写保护。当不等于 0 时，NVMPWP 位通常指向受写保护的最后一页的首地址。所有具有较少地址的页将受写保护。复位时，无 PFM 存储器受写保护。

PFMULOCK （NVMPWP<31>）是解锁位，可在复位时置 1，并可通过用户软件清零。当清零时，不能更改 PFM 的写保护，包括 PFMULOCK 位。只有在随后执行解锁序列时，才能更改 NVMPWP 写保护寄存器。更多信息，请参见**第 52.9 节 “NVMKEY 寄存器解锁序列”**。

52.6 错误校正代码 （ECC）和闪存编程

有些 PIC32 器件具有错误校正代码 （ECC）功能，可以检测和校正错误，从而延长闪存存储器的寿命。该功能已在**第 41 章“带 L1 CPU 高速缓存的器件上的预取模块”**（DS60001183）中进行了详细介绍。

ECC 以 128 位宽的闪存字或四个 32 位指令字组实现。因此，当在采用 ECC 的器件上编程闪存存储器时，编程操作必须是以最低四个指令字或四个指令字组的方式执行。这就是为什么存在四字编程命令及行编程通常编程多个四字的原因。

对于给定的软件应用程序， ECC 可以在任何时候使能 / 禁止，或使用 FECCCON 配置位动态使能。当在任何时候使能 ECC 时，单字 NVMOP 编程命令不起作用，四字是可编程的最小存储单元。当动态禁止或使能 ECC 时，字和四字编程 NVMOP 命令都可工作，使用的编程方法决定 ECC 的处理方式。

对于动态 ECC，如果存储器使用字命令编程，针对该字的 ECC 关闭，并且在读存储器时，不执行错误校正。如果存储器使用四字或行编程命令编程，读取时将针对错误写入或测试 ECC 数据（在需要时校正）。表 52-2 描述了不同的 ECC 情形。

表 52-2： ECC 编程汇总

FECCCON 设置	编程操作			数据读
	字写	四字写	行写	
禁止	允许	允许	允许	ECC 不适用于闪存读操作
使能	不允许	允许	允许	ECC 适用于每个闪存字读操作
动态	允许，但是使用时，将编程的字标志为不使用 ECC	写 ECC 数据并将编程的字标志为使用 ECC	写 ECC 数据并将编程的字标志为使用 ECC	ECC 仅适用于标志为使用 ECC 的字

注： 当使用动态 ECC 时，所有非 ECC 位置必须使用 32 位字编程命令编程，而所有支持 ECC 的位置必须使用 128 位四字或行编程命令编程。ECC 和非 ECC 存储器的分隔必须在偶数个四字边界上（地址位 bit 0-3 等于 0）。

52.7 中断

当完成闪存编程或擦除操作， WR 位由闪存控制器清零时，会产生中断。中断控制器中配置或允许的中断事件将引发 CPU 中断。不论编程或擦除操作的结果是成功还是失败，中断都会发生。唯一的例外是空操作 （NOP）编程操作 （NVMOP = 0），此操作用于手动清零错误标志，完成时不会产生中断事件，但会清零 WR 位。

闪存控制器中断不会持续，因此不需要其他步骤就可以清除中断原因或中断源。仅需要在退出中断服务程序之前清零相应的 IFSx 位即可。

一旦配置中断控制器，闪存事件将导致 CPU 跳转至分配给闪存事件的向量。CPU 将在此向量地址处执行代码。该向量地址处的用户软件将执行所需的操作，然后退出。有关中断、配置方法和向量地址表明细的更多信息，请参见 《PIC32 系列参考手册》中的**第 8 章 “中断”**（DS60001108）和具体器件数据手册中的 **“中断控制器”** 章节。

52.7.1 中断和 CPU 停止

CPU 不能从作为编程操作目标的同一个闪存存储区中取代码，这个存储区可能是 BFM 或 PFM。当尝试该操作时， CPU 在编程操作进行时将停止（暂停）代码执行。CPU 代码执行在编程操作完成后方可恢复，并且此时，任何暂挂的中断 （包括闪存控制器中断）都将优先处理。

注意，已装入处理器高速缓存的代码将继续执行，直至尝试将从同一闪存板上取得的代码或数据用于运行的编程操作时。此时 CPU 停止。

当所有可执行的代码和数据常量位于所有编程操作目标的相对存储区时，闪存事件中断允许系统启动闪存操作，并且开始执行代码并使用中断服务程序标志编程操作完成。使用该技术，设计具有后台闪存编程功能的系统使得闪存存储器实时更新成为可能。

在闪存编程期间，通过在 SRAM 放置任何所需的可执行代码可以避免停止。

52.8 错误检测

NVMCON 寄存器包括两个在编程或擦除期间检测错误情况的位。它们是低电压检测错误 LVDERR 位 （NVMCOM<12>）和写错误 WRERR 位 （NVMCON<13>）。

每次 WR 位 （NVMCON<15>）置 1 时， WRERR 也置 1，以启动编程操作。成功完成闪存操作时清零 WRERR，同时清零 WR，且直至闪存控制器清零之前不可查询 WR。当 WRERR 置 1 时，忽略所有试图启动编程或擦除的操作。 WRERR 必须在开始闪存编程或擦除操作之前清零。

当在编程操作期间发生欠压复位（Brown out Reset，BOR）时，LVDERR 位置 1。清零 LVDERR 位的唯一复位事件是上电复位 （POR）。其他复位类型不会影响 LVDERR 位。当 LVDERR 位置 1 时，忽略所有试图启动编程或擦除的操作。LVDERR 位必须在开始闪存编程或擦除操作之前清零。

通过将 NVMOP 置为 NOP （0x00）启动闪存操作 （WR 置 1），可在软件中手动清零 WRERR 和 LVDERR 位。注意，执行 NVMOP NOP 命令会清零 WRERR、LVDERR 和 WR 位，但是在完成时不产生中断事件。

表 52-3： 编程错误原因和影响

错误原因	编程擦除操作的影响	指示
在编程序列中发生低电压事件。	上一编程或擦除操作可能未完成。	LVDERR = 1， WRERR = 1
在编程时发生非 POR 复位。	中止编程或擦除操作。	WRERR = 1
试图编程或擦除闪存存储器外的页。	未启动擦除或编程操作。	WRERR = 1
试图擦除或编程写保护的 PFM 页。	未启动擦除或编程操作。	WRERR = 1
试图擦除或编程写保护的 BFM 页。	操作发生，但是页未编程或擦除。	WRERR = 0
编程期间总线主器件错误或行编程数据运行错误。	中止编程或擦除操作。	WRERR = 1

52.9 NVMKEY 寄存器解锁序列

由寄存器解锁序列保护重要的寄存器设置，以防发生意外更改时损坏闪存存储器。该功能使用 NVMKEY 寄存器实现。NVMKEY 寄存器是一个用于实现解锁序列的只写寄存器，有助于防止意外写或擦除闪存存储器或意外改变 SWAP （或 PFSWAP）位 （NVMCON<7>）和 BFSWAP 位 （NVMCON<6>），以及写许可设置 （见**注**）。

> **注：** 请参见具体器件数据手册中的**“闪存程序存储器”**章节，以确定您器件上的是 SWAP 位还是 PFSWAP 位。

在某些情况下，操作取决于 WREN 位 （NVMCON<14>）的设置，如表 52-4 所示。

表 52-4： NVMKEY 寄存器解锁和 WREN

操作	WREN 设置	SWAPLOCK 设置 （见注 1）	需要 解锁序列
改变 SWAP 或 PFSWAP （NVMCON<7>）的值	0	00	是
改变 BFSWAP （NVMCON<6>）的值 （见**注 1**）	0	00	是
改变 NVMOP<3:0> （NVMCON<3:0>）的值	0	无关位	否
WR （NVMCON<15>）置 1 启动写或擦除操作	1	无关位	是
改变 NVMPWP 寄存器的任何位域	无关位	无关位	是
改变 NVMBWP 寄存器的任何位域	无关位	无关位	是

注 1： 该位不是在所有器件上均可用。请参见具体器件数据手册上的 **“闪存程序存储器”**章节，以确定您器件上的是 SWAP 位还是 PFSWAP 位。

严格按照所示顺序执行以下步骤，以使能对需要该解锁序列的寄存器的写操作。

1. 将 0x00000000 写入 NVMKEY。
2. 将 0xAA996655 写入 NVMKEY。
3. 将 0x556699AA 写入 NVMKEY。
4. 将值写入需要解锁序列的寄存器。

当在 NVMCON 寄存器中使用解锁序列置 1 或清零位时（如第四步所示），必须执行第 2 至 4 步，且在闪存存储器使用的外设总线上没有任何其他活动。必须禁止与闪存控制器访问同一外设总线的中断和 DMA 传输。请参见具体器件数据手册以确定哪些外设与闪存控制器共享同一外设总线。另外，第 4 步的操作必须是原子操作。在第 4 步中，清零、置 1 和取反寄存器（适用时）可用于目标寄存器。

例 52-1 给出了启动 NVM 操作（NVMOP）命令的 C 语言代码。在特定示例中，WR 位在 NVMCON 寄存器中置 1，因此，必须包含解锁序列。注意， NVMCONSET 寄存器通过一条指令将 WR 位置 1，而不改变寄存器中的其他位。无法使用 NVMCONbits.WR = 1，这是因为该代码行会编译为读 - 修改 - 写序列。

例 52-1：　　启动 NVM 操作 （解锁序列示例）

```
void NVMInitiateOperation(void)
{
    int    int_status;     // storage for current Interrupt Enable state
    int    dma_susp;       // storage for current DMA state

    // Disable Interrupts
    asm volatile("di  %0" : "=r"(int_status));

    // Disable DMA
    if(!(dma_susp=DMACONbits.SUSPEND))
    {
        DMACONSET=_DMACON_SUSPEND_MASK;      // suspend
        while((DMACONbits.DMABUSY));         // wait to be actually suspended
    }

    NVMKEY = 0x0;
    NVMKEY = 0xAA996655;
    NVMKEY = 0x556699AA;
    NVMCONSET = 1 << 15;                     // must be an atomic instruction

    // Restore DMA
    if(!dma_susp)
    {
         DMACONCLR=_DMACON_SUSPEND_MASK;     // resume DMA activity
    }

    // Restore Interrupts
    if(int_status & 0x00000001)
    {
        asm volatile("ei");
    }
}
```

注：　一旦解锁代码已写入 NVMKEY 寄存器，与闪存控制器在同一外设总线的下一个活动将复位锁定。最终，仅可以使用原子操作。使用结构设置寄存器位域会编译为读 - 修改 - 写操作，导致失败。

52.10 字编程

单个操作中可编程的最小数据块是一个 32 位字。要编程的数据必须写入 NVMDATA0 寄存器，字地址必须在启动编程序列之前装入 NVMADDR 寄存器。然后编程 NVMADDR 寄存器指向的物理位置处的指令字。在 32 位字边界上编程；因此，将忽略 NVMADDR 寄存器中的 bit 0 和 bit 1。

编程字后，在再次编程之前必须先擦除，即使将位从擦除的 1 状态改为 0 状态也是如此。

只有目标地址位于未受写保护的页中时，字编程才能成功。编程写保护的 PFM 页将失败，并使 NVMCON寄存器中的WRERR位置1。编程写保护的BFM页将失败，但是不会将WRERR位置1。

编程序列包含以下步骤：

1. 将要编程的 32 位数据写入 NVMDATA0 寄存器。
2. 将要编程的地址装入 NVMADDR 寄存器。
3. 在 NVMCON 寄存器中，设置 WREN 位 = 1，NVMOP 位 = 1。这将定义并使能编程操作。
4. 启动编程操作 （见**第 52.9 节 “NVMKEY 寄存器解锁序列”**）。
5. 监视 NVMCON 寄存器中的 WR 位以标志操作是否完成。
6. 将 NVMCON 寄存器的 WREN 位清零。
7. 相应地检查错误和过程。

例 52-2 给出了字编程的代码，其中的 0x12345678 值编程至 0x1D008000 位置。

例 52-2： 字编程

```
…

// Set up Address and Data Registers
    NVMADDR  = 0x1D008000;     // physical address
    NVMDATA0 = 0x12345678;     // value

// set the operation, assumes WREN = 0
    NVMCONbits.NVMOP = 0x1;    // NVMOP for Word programming

// Enable Flash for write operation and set the NVMOP
    NVMCONbits.WREN = 1;

// Start programming
    NVMInitiateOperation();    // see Example 52-1

// Wait for WR bit to clear
    while(NVMCONbits.WR);

// Disable future Flash Write/Erase operations
    NVMCONbits.WREN = 0;

// Check Error Status
    if(NVMCON & 0x3000)        // mask for WRERR and LVDERR
    {
        // process errors
    }
…
```

52.11 四字编程

四字编程的过程与字编程相同，除使用了所有四个 NVMDATAx 寄存器以外。NVMDATA0 寄存器的值将在地址 NVMADDR 处编程， NVMDATA1 的值将在地址 NVMADDR + 0x4 处编程，NVMDATA2 的值将在地址 NVMADDR + 0x8 处编程， NVMDATA3 的值将在地址 NVMDATA + 0xC 处编程。

四字编程通常在四字边界上执行；因此，忽略 bit 3-0。

只有目标地址在未受写保护的页中时，四字编程才能成功。编程四字后，其中的任何字在再次编程之前必须先擦除，即使将位从擦除的 1 状态改为 0 状态也是如此。

例 52-3 给出了四字编程的代码，其中 0x11111111 值编程至 0x1D008000 位置中， 0x22222222 值编程至 0x1D008004 位置中， 0x33333333 值编程至 0x1D008008 位置中， 0x44444444 值编程至 0x1D00800C 位置中。

例 52-3： 四字编程

```
 …

// Set up Address and Data Registers
    NVMADDR  = 0x1D008000;     // physical address
    NVMDATA0 = 0x11111111;     // value written to 0x1D008000
    NVMDATA1 = 0x22222222;     // value written to 0x1D008004
    NVMDATA2 = 0x33333333;     // value written to 0x1D008008
    NVMDATA3 = 0x44444444;     // value written to 0x1D00800C

// set the operation, assumes WREN = 0
    NVMCONbits.NVMOP = 0x2;    // NVMOP for Quad Word programming

// Enable Flash for write operation and set the NVMOP
    NVMCONbits.WREN = 1;

// Start programming
    NVMInitiateOperation();    // see Example 52-1

// Wait for WR bit to clear
    while(NVMCON & NVMCON_WR);

// Disable future Flash Write/Erase operations
    NVMCONbits.WREN = 0;

// Check Error Status
    if(NVMCON & 0x3000)        // mask for WRERR and LVDERR bits
    {
        // process errors
    }
…
```

52.12 行编程

可编程的最大数据块是行，其大小因器件而异。请参见具体器件数据手册中的 “**闪存程序存储器**”章节，以确定行的大小。

与字和四字编程不同，这两类操作将数据源存储在 SFR 存储器中，而行编程操作将源数据存储在 SRAM 中。 NVMSRCADDR 寄存器指向行编程源数据的物理位置。

与其他非易失性存储器（NVM）编程命令相同， NVMADDR 寄存器指向操作的目标地址。行编程通常发生在行边界；因此，对于指令字行大小为 512 的器件，忽略 NVMADDR 寄存器中的 bit 0-9。

只有目标地址在未受写保护的页中时，行字编程才能成功。编程行后，其中的任何字在再次编程之前必须先擦除，即使将位从擦除的 `1` 状态改为 `0` 状态也是如此。

例52-4给出了用于行编程的代码。阵列`rowbuff`通过数据填充，并编程至物理地址`0x1D008000`处的行。

注：	当为 NVMSRCADDR 寄存器指定值时，必须将其转换为物理地址。

例 52-4： 行编程

```
 …

unsigned int rowbuff[512]; // example is for a 512 word row size.
int x;                     // loop counter

// put some data in the source buffer
    for (x = 0; x < (sizeof(rowbuff) * sizeof (int)); x++)
        ((char *)rowbuff)[x] = x;

// set destination row address
    NVMADDR = 0x1D008000;     // row physical address

// set source address. Must be converted to a physical address.
    NVMSRCADDR = (unsigned int)((int)rowbuff & 0x1FFFFFFF);

// define Flash operation
    NVMCONbits.NVMOP = 0x3;   // NVMOP for Row programming

// Enable Flash Write
    NVMCONbits.WREN = 1;

// commence programming
    NVMInitiateOperation();   // see Example 52-1

// Wait for WR bit to clear
    while(NVMCONbits.WR);

// Disable future Flash Write/Erase operations
    NVMCONbits.WREN = 0;

// Check Error Status
    if(NVMCON & 0x3000)       // mask for WRERR and LVDERR bits
    {
       // process errors
    }
…
```

52.13 页擦除

页擦除用于擦除 PFM 或 BFM 的一页。请参见具体器件数据手册中的**“闪存程序存储器”**章节，以了解器件的页大小。

使用NVMADDR寄存器选择要擦除的页。页通常在页边界擦除；因此，对于指令字页大小为4096的器件，忽略 NVMADDR 寄存器中的 bit 0-11。

只有目标地址在未受写保护的页中时，页擦除才能成功。擦除受写保护的页将失败，并使 NVMCON 寄存器中的 WRERR 位置 1。

例 52-5 给出了在地址 0x1D008000 上擦除单独一页的代码。

例 52-5： 页擦除

```
…

// set destination page address
    NVMADDR = 0x1D008000;     // page physical address

// define Flash operation
    NVMCONbits.NVMOP = 0x4;   // NVMOP for Page Erase

// Enable Flash Write
    NVMCONbits.WREN = 1;

// commence programming
    NVMInitiateOperation();   // see Example 52-1

// Wait for WR bit to clear
    while(NVMCONbits.WR);

// Disable future Flash Write/Erase operations
    NVMCONbits.WREN = 0;

// Check Error Status
    if(NVMCON & 0x3000)       // mask for WRERR and LVDERR bits
    {
       // process errors
    }
…
```

52.13.1 页擦除重试

对于支持该功能的 PIC32 器件，擦除重试是提升闪存分区寿命的一种方法，会在擦除不成功时尝试再次擦除。擦除重试只能用于页擦除。

擦除重试通过在擦除时增加闪存上使用的电压实现。一开始，通过设置 NVMRETRY<1:0> 位（NVMCON2<9:8>）= 00 应用所需的最小电压。如果页擦除不成功，可以通过递增 NVMRETRY<1:0> 位的设置来提升电压。请注意，由于每个闪存页的老化程度不同，可能需要的电压也不一样，因此一个闪存页需要较高电压设定不代表该设定可以应用到所有页。当 NVMRETRY<1:0>位= 11时，使用最大页擦除电压。如果当NVMRETRY<1:0>位= 11时页擦除不成功，则意味着该闪存页或字的内容未擦除，应视为 “无功能性”。

除了常规的页擦除控制，擦除重试还使用 NVMCON2 寄存器中的 NVMWS<4:0>、 NVMLPRD、NVMCRD、NVMVRD 和 NVMRETRY<1:0> 位。在掉电导致的 BOR 事件（而非 POR 事件）中，NVMERS<3:0> 位 （NVMCON2<31:28>）用于软件执行的编程序列。

执行以下步骤以建立页擦除重试：

1. 在 NVMADDR 寄存器中设置要擦除的页地址。
2. 执行写解锁序列。
3. 保存 NVMCON2 寄存器的值。
4. 在 NVMCON2 寄存器中执行以下操作：
   a) 根据需要设置 NVMERS<3:0> 位。
   b) 根据说明设置 NVMWS<4:0> 位。
   c) 将 NVMVRD 位设为 1。
   d) 将 NVMCRD 位设为 1。
   e) 将 NVMRETRY<1:0> 位设为 00。
5. 采用页擦除命令运行解锁序列以启动序列。
6. 等待硬件清零 WR 位 （NVMCON<15>）。
7. 清零 WREN 位 （NVMCON<14>）。
8. 通过 CPU 验证擦除：

   为了缩短验证时间，使用 NVMCRD = 1 对闪存字中的每个位（包括 ECC，如果提供）执行与逻辑 1 的硬件比较。 NVMCRD = 1 只对包含 NVMADDR 的分区起作用。成功的比较将在闪存字 （128 位）的最低地址字处读取 0x00000001 值。该值为比较字。其他所有字为 0x00010000。如果有位为逻辑 0，则该闪存字中的所有字均读为 0x00000000。在两次读操作之间，请记得通过闪存字中的字节数来递增地址。
9. 如果所有的比较字均验证正确，则擦除重写流程完成。直接执行步骤 11。
10. 如果比较字读为0x00000000，则重复执行步骤4至9（最多三次），步骤4将进行如下更改：
    a) 将 NVMRETRY<1:0> 位递增 1。
    b) 保持所有其他位域。
11. 恢复在步骤 3 中保存的 NVMCON2 寄存器的值。

注 1： 当 NVMVRD = 1 时，闪存使用 NVMWS<3:0> 位用于生成由 NVMADDR 选择的分区的闪存访问等待状态。当擦除和验证均完成后，通过软件将 NVMVRD 位写回为 0。

2： 引导页 3 （该页包含 DEVCFGx 值）不支持擦除重试。

例 52-6 提供了使用擦除重试对地址 0x1D008000 进行单页擦除操作的代码。

例 52-6：　　页擦除重试

```
uint32_tsaveNVMCON2;
uint32_t*cmpPtr;
uint8_t erased;
uint8_t tryCount;

// set destination page address
    NVMADDR = 0x1D008000;      // Page physical address

// define flash operation
    NVMCONbits.NVMOP = 0x4;    // NVMOP for Page Erase

// Unlock sequence
    NVMKEY = 0x0;
    NVMKEY = 0xAA996655;
    NVMKEY = 0x556699AA;

// save NVMCON2
    saveNVMCON2 = NVMCON2;

// set up Erase Retry
    NVMCON2bits.NVMERS = 0;    // Stage 0 - SW use only
    NVMCON2bits.NVMVRD = 1;
    NVMCON2bits.NVMCRD = 1;
    NVMCON2bits.NVMRETRY = 0b00;

    tryCount = 0;              // Up to 4 attempts

    do {
        tryCount++;

        // commence programming
        NVMInitiateOperation();

        // Wait for WR bit to clear
        while(NVMCONbits.WR);

        // Turn off WREN
        NVMCONbits.WREN = 0;

        // Check that the page was erased
        erased = 1;
        cmpPtr = (uint32_t *)NVMADDR;
        erased &= (*cmpPtr == 0x00000001);
        cmpPtr++;
        erased &= (*cmpPtr == 0x00010000);
        cmpPtr++;
        erased &= (*cmpPtr == 0x00010000);
        cmpPtr++;
        erased &= (*cmpPtr == 0x00010000);

        if (!erased) {
            // Erase failed. Try with different settings.
            NVMCON2bits.NVMRETRY++;

            NVMCONbits.NVMOP = 0x4;
            NVMCONbits.WREN = 1;
        }
    } while (!erased && (tryCount < 4));

// Restore settings
    NVMCON2 = saveNVMCON2;
```

52.14 闪存程序存储器擦除

闪存程序存储器可以整体擦除或者按存储区擦除。通过执行三个独立的NVMOP命令实现此操作：

- 擦除整个 PFM 区 （两个 PFM 存储区）
- 擦除 PFM 的较高存储区
- 擦除 PFM 的较低存储区

当擦除整个 PFM 区时，代码必须从 BFM 执行。当擦除单个较高或较低的 PFM 存储区时，必须从未擦除的 BFM 或 PFM 存储区中执行代码。

只有正在擦除的存储区的页都不受写保护时，程序闪存存储器擦除操作才能成功。当擦除整个 PFM 区时， PFM 写保护必须完全禁止。

例 52-7 给出了擦除较高的程序闪存存储区的代码。

例 52-7： 程序闪存擦除

```
…

// define Flash operation
    NVMCONbits.NVMOP = 0x6;       // NVMOP for upper bank PFM erase

// Enable Flash Write
    NVMCONbits.WREN = 1;

// commence programming
    NVMInitiateOperation();       // see Example 52-1

// Wait for WR bit to clear
    while(NVMCONbits.WR);

// Disable future Flash Write/Erase operations
    NVMCONbits.WREN = 0;

// Check Error Status
    if(NVMCON & 0x3000)       // mask for WRERR and LVDERR bits
    {
       // process errors
    }
…
```

52.15 节能模式下的操作

闪存控制器在节能模式下不工作。如果在编程时遇到 WAIT 指令，CPU 将停止执行，等待编程操作完成，然后进入节能模式。

52.16 调试模式下的操作

如果处理器在调试模式下暂停执行，编程操作将继续执行。

52.17 各种复位的影响

除上电复位（POR）以外的器件复位将复位 NVMCON 寄存器中的 SWAP（或 PFSWAP）位以及 NVMPWP 和 NVMBWP 寄存器的全部内容（见**注**）。所有其他寄存器的内容在非 POR 复位时保持不变。

注：	请参见具体器件数据手册中的**“闪存程序存储器”**章节，以确定您器件上的是 SWAP 位还是 PFSWAP 位。

所有的闪存控制器寄存器在 POR 时强制进入复位状态。

52.18 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 PIC32 器件系列而编写的，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。当前与支持实时更新的闪存相关的应用笔记包括：

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 PIC32 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

52.19 版本历史

版本 A （2013 年 5 月）

这是本文档的初始版本。

版本 B （2015 年 7 月）

该版本包含以下更新：

- NVMCON 寄存器中的 BOOTSWAP 位重命名为：BFSWAP （见寄存器 52-1）
- 增加了 NVMCON2 寄存器 （见表 52-1 和寄存器 52-8）
- 更新了**第 52.4.1 节 “BFM 存储区别名”**
- 更新了**第 52.5.1 节 “PFM 映射至地址空间”**
- 更新了**第 52.9 节 “NVMKEY 寄存器解锁序列”**
- 更新了**第 52.13.1 节 “页擦除重试”**
- 更新了所有的代码示例 （见例 52-1 至例 52-7）
- 对整篇文档的文字和格式进行了少量更新

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。

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。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949═

ISBN：978-1-5224-0634-1

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 - Karlsruhe
Tel: 49-721-625370

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

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

07/14/15