dsPIC33EPXXXGS70X/80X 系列闪存编程规范

1.0 器件概述

本文档定义了dsPIC33EPXXXGS70X/80X 16位数字信号控制器（Digital Signal Controller， DSC）系列的编程规范。本编程规范仅供为以下器件开发编程支持的人员使用：

- dsPIC33EP64GS708
- dsPIC33EP64GS804
- dsPIC33EP64GS805
- dsPIC33EP64GS806
- dsPIC33EP64GS808
- dsPIC33EP128GS702
- dsPIC33EP128GS704
- dsPIC33EP128GS705
- dsPIC33EP128GS706
- dsPIC33EP128GS708
- dsPIC33EP128GS804
- dsPIC33EP128GS805
- dsPIC33EP128GS806
- dsPIC33EP128GS808

仅使用这些器件的客户应使用已支持器件编程的开发工具。

本文档包含以下主题：

2.0 编程概述

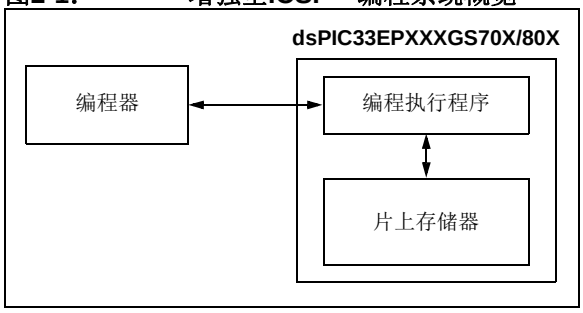
在本编程规范中将讨论两种对器件编程的方法：

- 在线串行编程 （In-Circuit Serial Programming™，ICSP™）
- 增强型在线串行编程

ICSP 编程方法是对器件编程最直接的方法，但也是两种方法中较慢的一种。它提供固有的低级编程功能来擦除、编程和校验器件。

增强型ICSP协议采用一种较快的方法，该方法利用了如图2-1中所示的编程执行程序（Programming Executive，PE）。PE 通过一个小的命令集提供擦除、编程和校验芯片所必需的所有功能。该命令集使得编程器对dsPIC33EPXXXGS70X/80X器件的编程无需处理低级编程协议。

图2-1： 增强型ICSP™编程系统概览



本编程规范分为两个主要部分，分别对上述两种编程方法进行了介绍。**第 3.0 节 “器件编程 ——ICSP”** 介绍了 ICSP 方法。**第 4.0 节“器件编程 —— 增强型 ICSP”** 介绍了增强型 ICSP 方法。

2.1 必需的连接

这些器件需要进行特定的连接，才可进行编程。这些连接包括电源、VCAP、MCLR 和一个编程引脚对（PGDx/PGCx）。表 2-1 对这些连接进行了说明 （欲知引脚说明和电源连接要求，请参见具体器件数据手册）。

2.2 电源要求

所有 dsPIC33EPXXXGS70X/80X 器件都使用标称值为 1.8V 的电压为其内核数字逻辑供电。为了简化系统设计，dsPIC33EPXXXGS70X/80X 系列中的所有器件均包含一个片上稳压器，可使器件内核逻辑在 VDD 下工作。

稳压器通过 VDD 引脚为内核供电。必须在 VCAP 引脚上连接一个低 ESR 的电容 （如陶瓷电容或钽电容）（见表 2-1 和图 2-2）。这可以维持稳压器的稳定性。**第 10.0 节 “交流 / 直流特性和时序要求”**中列出了内核电压和电容的规范。

图 2-2： 片上稳压器连接

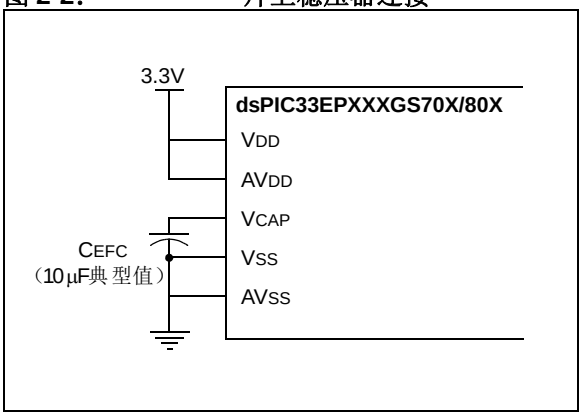


表 2-1： 编程期间使用的引脚

引脚名称	引脚类型	引脚说明
MCLR	I	编程使能
VDD 和 AVDD(1)	P	电源 (1)
VSS 和 AVSS(1)	P	地 (1)
VCAP	P	内部稳压器滤波电容
PGCx	I	编程引脚对：串行时钟
PGDx	I/O	编程引脚对：串行数据

图注： I = 输入 O = 输出 P = 电源

注 1： 所有电源和地引脚必须连接，包括 AVDD 和 AVSS。

2.3 引脚图

图 2-3 至图 2-7 给出了 dsPIC33EPXXXGS70X/80X 系列的引脚图。表 2-1 列出了进行编程所需的引脚，各图中以粗体文字来指示这些引脚。关于完整的引脚说明，请参见相应器件的数据手册。

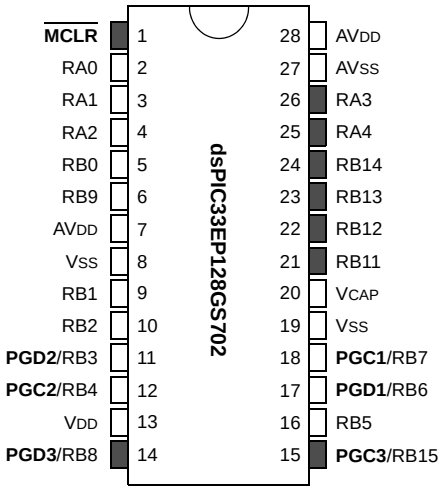
2.3.1 PGCx和PGDx引脚对

dsPIC33EPXXXGS70X/80X 系列中的所有器件均具有 3 对独立的编程引脚，标记为 PGC1/PGD1、PGC2/PGD2 和 PGC3/PGD3。ICSP 或增强型 ICSP 可以使用这些引脚对中的任意一对来进行器件编程。与电源和地引脚不同，对器件进行编程并不需要连接全部 3 个引脚对。但是，编程方法必须使用同一引脚对的两个引脚。

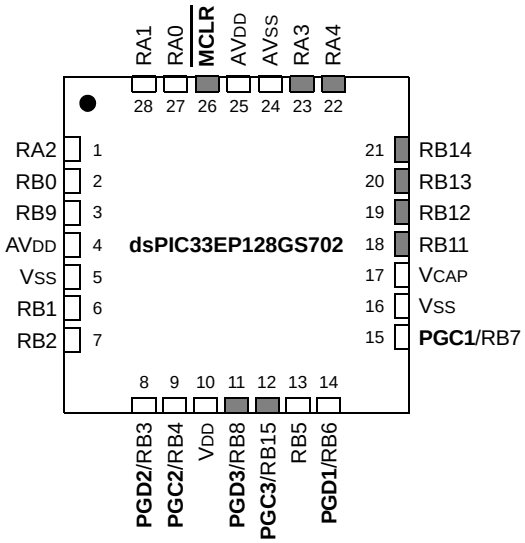
图 2-3： 引脚图

28引脚SOIC

= 引脚最高承受5V电压



28引脚QFN-S和UQFN

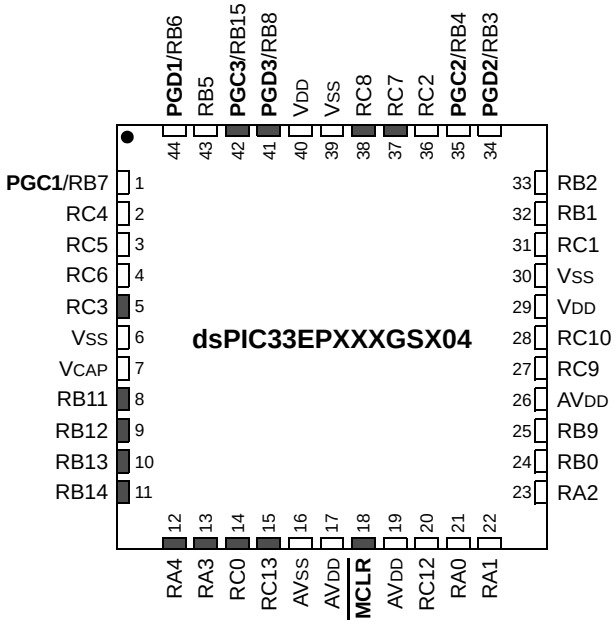


注 1： 粗体表示器件编程中使用的引脚。

图2-4：　　引脚图（续）

44引脚QFN和TQFP

=引脚最高承受5V的电压

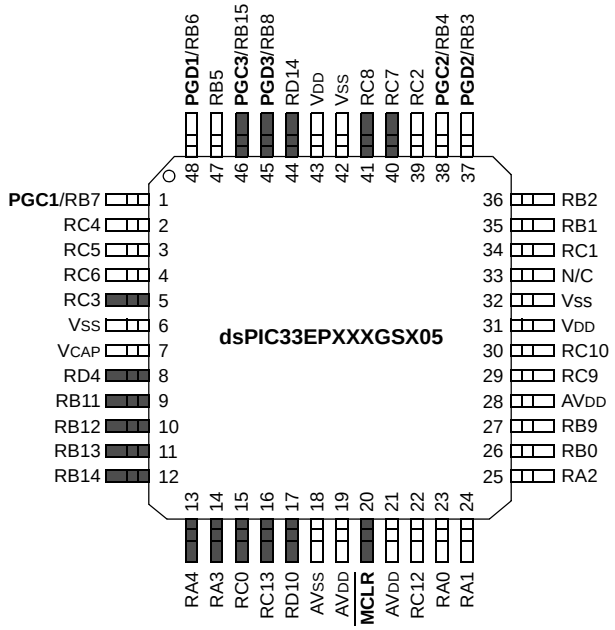


注　　1：　粗体表示器件编程中使用的引脚。

图 2-5：　　引脚图（续）

48引脚TQFP

=引脚最高承受5V的电压

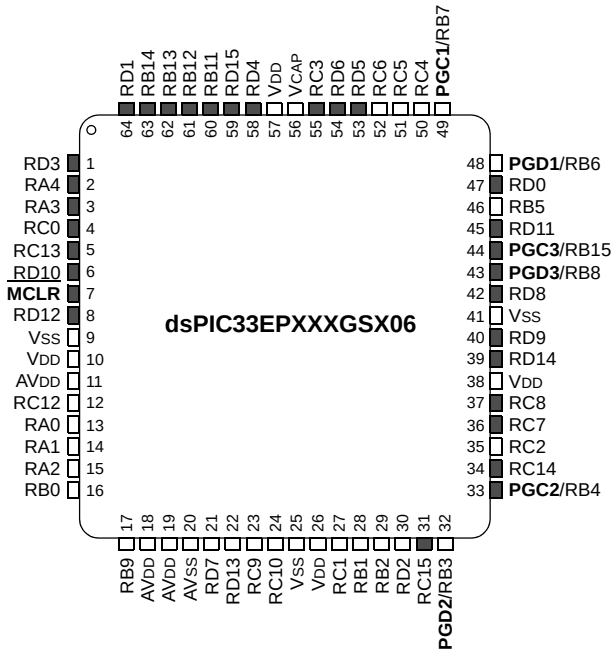


注　　1：　粗体表示器件编程中使用的引脚。

图 2-6：　　引脚图（续）

64引脚TQFP

=引脚最高承受5V的电压

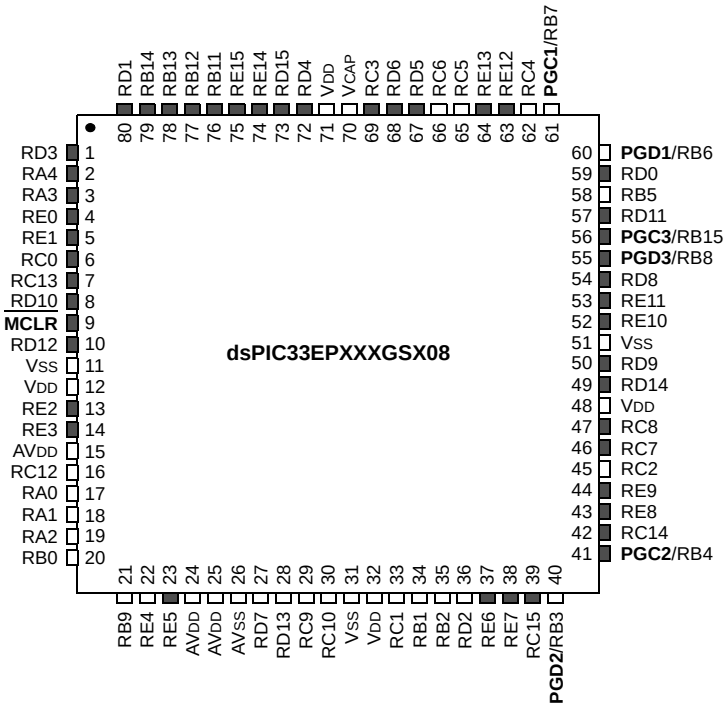


注　1：　粗体表示器件编程中使用的引脚。

图 2-7：　　引脚图（续）

80引脚TQFP

=引脚最高承受5V的电压



注　1：粗体表示器件编程中使用的引脚。

2.4 程序存储器写 / 擦除要求

为了使器件正常工作，必须严格遵守对这些器件的闪存程序存储器进行写 / 擦除的要求。规则如下：在没有擦除存储器中将要写入的任何给定字所在存储页之前，不能对该字进行写入。因此，遵守该规则最简单的方法就是在一个写周期内写入一个编程块中的所有数据。本文档中指定的编程方法就符合这一要求。

注：	一个程序存储字可以在擦除之前被编程两次，但仅限于 (a) 两次编程操作使用相同的数据或 (b) 包含 1 的位被设置为 0，但为 0 的位不能设置为 1。

2.5 存储器映射

程序存储器映射从 000000h 延伸到 FFFFFEh。代码存储区位于存储器映射的起始部分。实现的程序存储器的最后几个存储单元保留用于存储器件配置位。

表 2-2 给出了每款器件中代码存储区的大小以及擦除块的大小和数量。

地址从 800200h 至 800BFEh 的存储单元保留用于存储执行程序代码。该区域存储编程执行程序 （PE）和调试执行程序，用于器件编程。存储器中的该区域不能用于存储用户代码。更多信息，请参见**第 6.0 节 “编程执行程序”**。

地址为 FF0000h 和 FF0002h 的两个存储单元保留用于存储器件 ID 字寄存器。编程器可使用这些位来标识要编程的器件类型。这将在**第 8.0 节 “器件 ID/ 唯一 ID”**中进行介绍。即使应用了代码保护，也可正常读出器件 ID 寄存器。

地址从 800F80h 至 800FFEh 的存储单元为可一次性编程 （One-Time-Programmable， OTP）存储区。用户 OTP 字可用于存储产品信息，如序列号、系统生产日期、生产批号和其他特定于应用的信息。这将在**第 2.6.3 节 “用户 OTP（可一次性编程）存储区”**中进行介绍。

图 2-8 至图 2-11 给出了表 2-2 中列出的器件的通用存储器映射。请参见具体器件数据手册的**“存储器构成”**章节获得确切存储地址。

表 2-2： 单分区闪存代码存储区大小

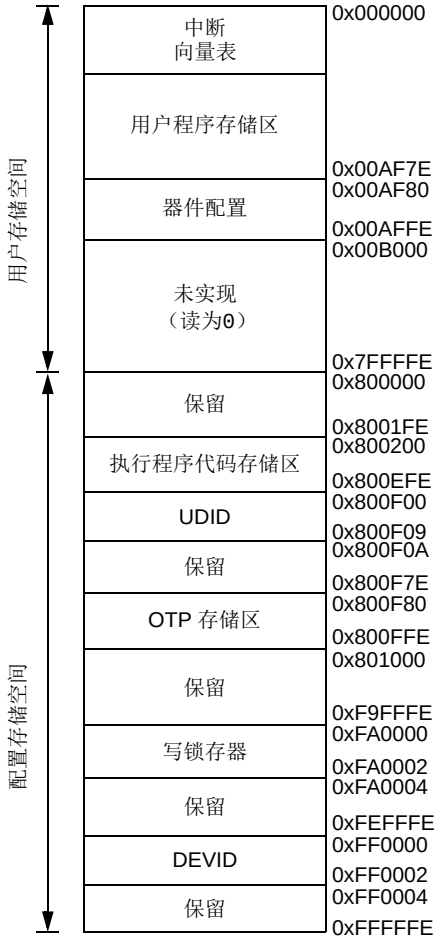
器件系列	用户存储区地址限制（指令字）	擦除块 / 页数 (1)	执行程序存储区地址限制（指令字）
dsPIC33EP64GS708	00AF7Eh (22,464)	44	0x800200-0x800BFE
dsPIC33EP64GS804			
dsPIC33EP64GS805			
dsPIC33EP64GS806			
dsPIC33EP64GS808			
dsPIC33EP128GS702	01577Eh （43,968）	86	
dsPIC33EP128GS704			
dsPIC33EP128GS705			
dsPIC33EP128GS706			
dsPIC33EP128GS708			
dsPIC33EP128GS804			
dsPIC33EP128GS805			
dsPIC33EP128GS806			
dsPIC33EP128GS808			

注 1： 1 个擦除块等于 512 个 24 位指令字。

表 2-3： 配置寄存器地址

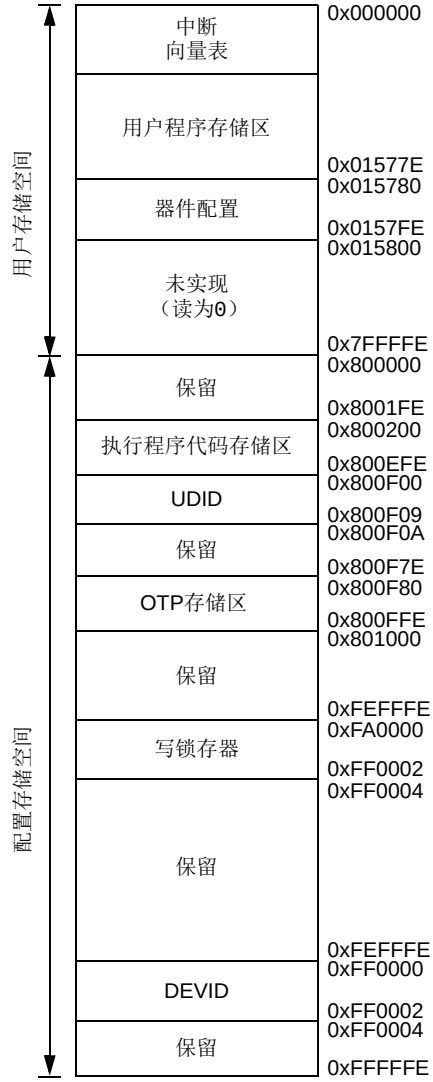
器件	FSEC	FBSLIM	FSIGN	FOSCSEL	FOSC	FWDT	FPOR	FICD	FDEVOPT	FALTREG	FBTSEQ	FBOOT
dsPIC33EP64GS708	00AF80	00AF90	00AF94	00AF98	00AF9C	00AFA0	00AFA4	00AFA8	00AFAC	00AFB0	00AFFC	801000
dsPIC33EP64GS804												
dsPIC33EP64GS805												
dsPIC33EP64GS806												
dsPIC33EP64GS808												
dsPIC33EP128GS702	015780	015790	015794	015798	01579C	0157A0	0157A4	0157A8	0157AC	0157B0	0157FC	801000
dsPIC33EP128GS704												
dsPIC33EP128GS705												
dsPIC33EP128GS706												
dsPIC33EP128GS708												
dsPIC33EP128GS804												
dsPIC33EP128GS805												
dsPIC33EP128GS806												
dsPIC33EP128GS808												

图2-8： dsPIC33EP64GS70X/80X器件的程序存储器映射(1)



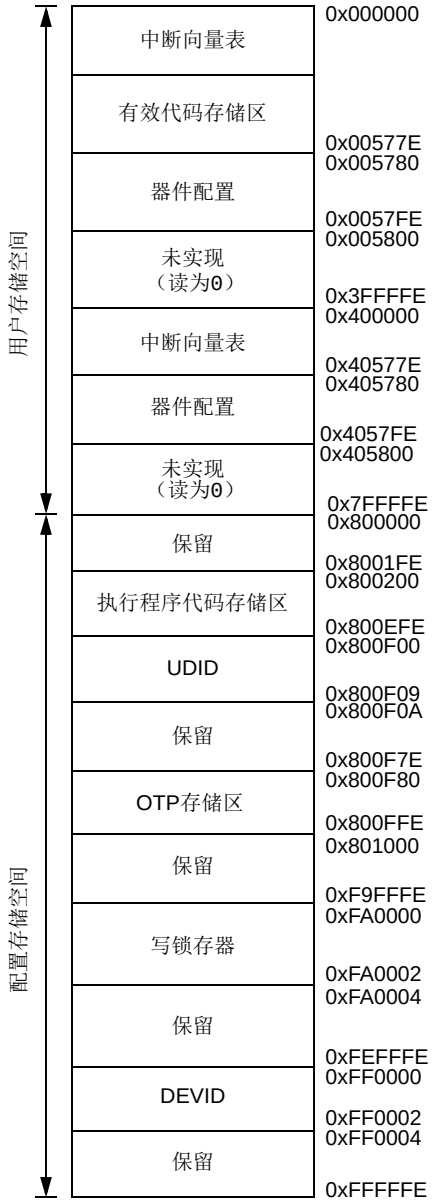
注 1： 存储区未按比例绘制。

图2-9： dsPIC33EP128GS70X/80X器件的程序存储器映射(1)



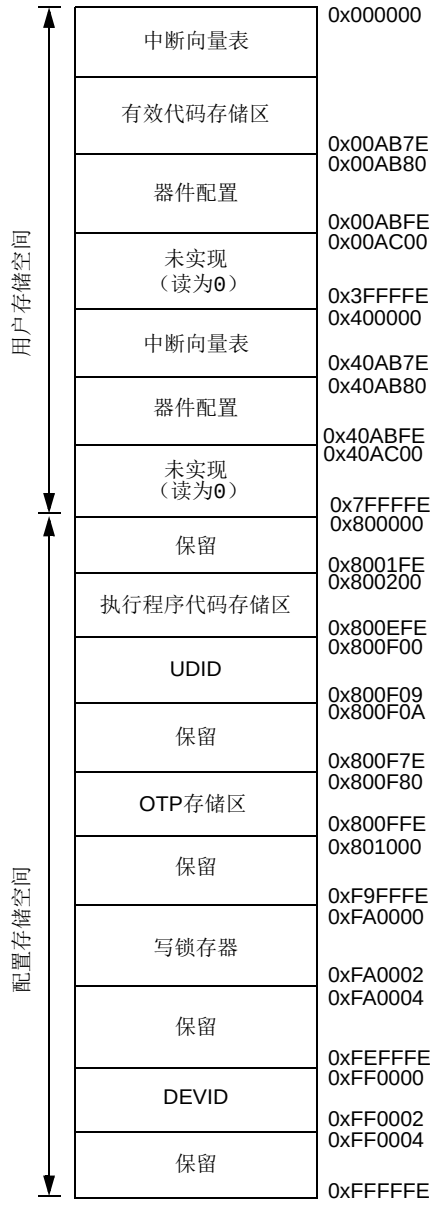
注 1： 存储区未按比例绘制。

图2-10： dsPIC33EP64GS70X/80X器件的程序存储器映射（双分区闪存）(1)



注 1： 存储区未按比例绘制。

图2-11： dsPIC33EP128GS70X/80X器件的程序存储器映射（双分区闪存）(1)



注 1： 存储区未按比例绘制。

2.6 配置位

2.6.1 概述

配置位存储在实现的程序存储器的最后一页中。可以将这些位置 1 或清零来选择各种器件配置。具有两种不同类型的配置位：系统操作位和代码保护位。系统操作位决定系统级组件（如振荡器和看门狗定时器）的上电设置。代码保护位防止程序存储器被读写。

表 2-3 列出了每款器件的配置寄存器地址范围。

表 2-4 给出了配置寄存器映射。请参见具体器件数据手册中的**“特殊功能”**章节了解器件的完整配置字寄存器说明。

2.6.2 代码保护配置位

这些器件实现了由 FSEC 寄存器定义的中等安全功能。引导段（Boot Segment，BS）是权限最高的段，通用段（General Segment，GS）是权限最低的段。整个用户代码存储区可以拆分为 BS 或 GS。这两个段的大小由 BSLIM<12:0> 位决定。段在用户空间内的相对位置不会改变，BS（如果存在）占用紧接在中断向量表（Interrupt Vector Table，IVT）VS 后的存储区，GS 占用紧接在 BS 后的空间，但如果使能了备用中断向量表（Alternate Interrupt Vector Table，AIVT），则 GS 占用紧接在 AIVT VS 后的空间。配置段（Configuration Segment，CS）是用户闪存地址空间中一个很小的段（小于一页，通常只是一行），其中包含 NVM 控制器在复位序列期间装入的所有用户配置数据。

2.6.3 用户 OTP（可一次性编程）存储区

dsPIC33EPXXXGS70X/80X 系列器件提供了 64 字（24 位宽存储器中的 32 个地址单元）的可一次性编程（One-Time-Programmable，OTP）存储区，位于地址 800F80h 至 800FFEh 处。该存储区可用于持久存储在对器件重新编程时不会擦除的特定于应用的信息。这包括许多类型的信息，例如：

- 应用程序检验和
- 代码版本信息
- 产品信息
- 序列号
- 系统生产日期
- 生产批号

可以在任意模式下对用户 OTP 存储区进行编程，包括用户 RTSP 模式，但它无法擦除。芯片擦除不会清除数据。

表 2-4：　配置寄存器映射 (2)

寄存器名称	地址	器件存储器大小（KB）	Bit 23-16	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
FSEC	00AF80	64	—	AIVTDIS	—	—	—	CSS<2:0>			CWRP	GSS<1:0>		GWRP	—	BSEN	BSS<1:0>		BWRP
	015780	128																	
FBSLIM	00AF90	64	—	—	—	—	BSLIM<12:0>												
	015790	128																	
FSIGN	00AF94	64	—	保留 (4)	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
	015794	128																	
FOSCSEL	00AF98	64	—	—	—	—	—	—	—	—	—	IESO	—	—	—	—	FNOSC<2:0>		
	015798	128																	
FOSC	00AF9C	64	—	—	—	—	—	—	—	—	PLLKEN	FCKSM<1:0>		IOL1WAY	—	—	OSCIOFNC	POSCMD<1:0>	
	01579C	128																	
FWDT	00AFA0	64	—	—	—	—	—	—	—	WDTWIN<1:0>		WINDIS	WDTEN<1:0>		WDTPRE	WDTPOST<3:0>			
	0157A0	128																	
FPOR	00AFA4	64	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	保留 (1)
	0157A4	128																	
FICD	00AFA8	64	—	BTSWP	—	—	—	—	—	—	—	保留 (1)	—	JTAGEN	—	—	—	ICS<1:0>	
	0157A8	128																	
FDEVOPT	00AFAC	64	—	—	—	—	—	—	—	—	—	—	DBCC	—	ALTI2C<2:1>		保留 (1)	—	PWMLOCK
	0157AC	128																	
FALTREG	00AFB0	64	—	—	CTXT4<2:0>			保留 (1)	CTXT3<2:0>			保留 (1)	CTXT2<2:0>			—	CTXT1<2:0>		
	0157B0	128																	
FBTSEQ	00AFFC	64	IBSEQ<11:0>					BSEQ<11:0>											
	0157FC	128																	
FBOOT(3)	801000	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	BTMODE<1:0>	

图注：　— = 未实现，读为 1。

注　1：　这些位保留且必须编程为 1。

2：　当工作在双分区闪存模式下时，每个闪存分区将具有专用的配置寄存器。器件复位后，在启动时读取活动分区的配置值，但在发生软交换情况时，将忽略新活动分区的配置设置。

3：　FBOOT 位于配置存储空间中。

4：　该位保留，且必须编程为 0。

表2-5： dsPIC33EPXXXGS70X/80X配置位说明

位域	说明
BSS<1:0>	引导段代码保护级别位 11 = 无代码保护 （BWRP 代码保护除外） 10 = 标准安全性 0x = 高安全性
BSEN	引导段控制位 1 = 禁止引导段 0 = 引导段大小由 BSLIM<12:0> 位决定
BWRP	引导段写保护位 1 = 引导段可写 0 = 引导段被写保护
BSLIM<12:0>	引导段闪存页地址限制位 包含通用段第一个活动页的页地址。要编程的值是页地址的补码，这样，编程更多 0 只会增大引导段大小 （即， 0x1FFD = 2 页或 1024 指令字）。
GSS<1:0>	通用段代码保护级别位 11 = 用户程序存储区不被代码保护 10 = 标准安全性 0x = 高安全性
GWRP	通用段写保护位 1 = 用户程序存储区不被写保护 0 = 用户程序存储区被写保护
CWRP	配置段写保护位 1 = 配置段不被写保护 0 = 配置段被写保护
CSS<2:0>	配置段代码保护级别位 111 = 配置数据不受代码保护 110 = 标准安全性 10x = 增强安全性 0xx = 高安全性
BTSWP	BOOTSWP 指令使能 / 禁止位 1 = 禁止 BOOTSWP 指令 0 = 使能 BOOTSWP 指令
BSEQ<11:0>	引导序列号位 （仅限双分区闪存模式） 用于定义在器件复位后哪个分区将处于活动状态的相对值；包含较低引导号的分区将为活动状态。
IBSEQ<11:0>	引导序列号反码位 （仅限双分区闪存模式） BSEQ<11:0> 的反码；必须由用户计算且写入进行器件编程。如果 BSEQx 和 IBSEQx 不互为反码，则引导序列号视为无效。
AIVTDIS	备用中断向量表位 (1) 1 = 禁止备用中断向量表 0 = 使能备用中断向量表 （前提是 INTCON2<8> = 1）
IESO	双速振荡器启动使能位 1 = 使用 FRC 启动器件，然后自动切换到就绪的用户选择的振荡器源 0 = 使用用户选择的振荡器源启动器件
PWMLOCK	PWM 锁定使能位 1 = 只有在密钥序列之后，才能对某些 PWM 寄存器进行写操作 0 = 无需密钥序列即可对 PWM 寄存器进行写操作

注 1： 必须存在引导段才能使用备用中断向量表。

表2-5： dsPIC33EPXXXGS70X/80X配置位说明（续）

位域	说明
FNOSC<2:0>	初始振荡器源选择位 111 = N 分频快速 RC （Fast RC， FRC）振荡器 （FRCDIVN） 110 = 16 分频快速 RC （FRC）振荡器 101 = 低功耗 RC 振荡器 （LPRC） 100 = 保留；不要使用 011 = 带 PLL 模块的主 （XT、 HS 和 EC）振荡器 010 = 主 （XT、 HS 和 EC）振荡器 001 = 带 PLL 模块的 N 分频快速 RC 振荡器 （FRCPLL） 000 = 快速 RC （FRC）振荡器
FCKSM<1:0>	时钟切换模式位 1x = 禁止时钟切换，禁止故障保护时钟监视器 01 = 使能时钟切换，禁止故障保护时钟监视器 00 = 使能时钟切换和故障保护时钟监视器
IOL1WAY	外设引脚选择配置位 1 = 仅允许一次重新配置 0 = 允许多次重新配置
OSCIOFNC	OSC2 引脚功能位 （XT 和 HS 模式除外） 1 = OSC2 为时钟输出 0 = OSC2 为通用数字 I/O 引脚
POSCMD<1:0>	主振荡器模式选择位 11 = 禁止主振荡器 10 = HS 晶振模式 01 = XT 晶振模式 00 = 外部时钟 （EC）模式
WDTEN<1:0>	看门狗定时器使能位 11 = 始终使能看门狗定时器 （无法禁止 LPRC 振荡器；清零 RCON 寄存器中的 SWDTEN 位将不起作用） 10 = 通过用户软件使能 / 禁止看门狗定时器 （可通过清零 RCON 寄存器中的 SWDTEN 位来禁止 LPRC） 01 = 看门狗定时器仅在器件处于活动状态时使能，在休眠模式下禁止；软件控制在该模式下禁止 00 = 禁止看门狗定时器和 SWDTEN 位
WINDIS	看门狗定时器窗口使能位 1 = 看门狗定时器处于非窗口模式 0 = 看门狗定时器处于窗口模式
PLLKEN	PLL 锁定使能位 1 = 使能 PLL 锁定 0 = 禁止 PLL 锁定
WDTPRE	看门狗定时器预分频比位 1 = 1:128 0 = 1:32
WDTPOST<3:0>	看门狗定时器后分频比位 1111 = 1:32,768 1110 = 1:16,384 • • • 0001 = 1:2 0000 = 1:1

注 1： 必须存在引导段才能使用备用中断向量表。

表2-5： dsPIC33EPXXXGS70X/80X配置位说明（续）

位域	说明
WDTWIN<1:0>	看门狗定时器窗口选择位 11 = WDT 窗口为 WDT 周期的 25% 10 = WDT 窗口为 WDT 周期的 37.5% 01 = WDT 窗口为 WDT 周期的 50% 00 = WDT 窗口为 WDT 周期的 75%
ALTI2C1	I2C1 的备用 I^2C 引脚位 1 = I2C1 被映射到 SDA1/SCL1 引脚 0 = I2C1 被映射到 ASDA1/ASCL1 引脚
ALTI2C2	I2C2 的备用 I^2C 引脚位 1 = I2C2 被映射到 SDA2/SCL2 引脚 0 = I2C2 被映射到 ASDA2/ASCL2 引脚
JTAGEN	JTAG 使能位 1 = 使能 JTAG 0 = 禁止 JTAG
ICS<1:0>	ICD 通信通道选择位 11 = 通过 PGC1 和 PGD1 进行通信 10 = 通过 PGC2 和 PGD2 进行通信 01 = 通过 PGC3 和 PGD3 进行通信 00 = 保留，不要使用
DBCC	DAC 输出交叉连接位 1 = DAC 输出不交叉连接 0 = DACOUT1 和 DACOUT2 互相连接
CTXT1<2:0>	指定备用工作寄存器 1 的中断优先级 （Interrupt Priority Level™， IPL™）的位 111 = 保留 110 = 分配为 IPL 级别 7 101 = 分配为 IPL 级别 6 100 = 分配为 IPL 级别 5 011 = 分配为 IPL 级别 4 010 = 分配为 IPL 级别 3 001 = 分配为 IPL 级别 2 000 = 分配为 IPL 级别 1
CTXT2<2:0>	指定备用工作寄存器 2 的中断优先级 （IPL）的位 111 = 保留 110 = 分配为 IPL 级别 7 101 = 分配为 IPL 级别 6 100 = 分配为 IPL 级别 5 011 = 分配为 IPL 级别 4 010 = 分配为 IPL 级别 3 001 = 分配为 IPL 级别 2 000 = 分配为 IPL 级别 1
CTXT3<2:0>	指定备用工作寄存器 3 的中断优先级 （IPL）的位 111 = 保留 110 = 分配为 IPL 级别 7 101 = 分配为 IPL 级别 6 100 = 分配为 IPL 级别 5 011 = 分配为 IPL 级别 4 010 = 分配为 IPL 级别 3 001 = 分配为 IPL 级别 2 000 = 分配为 IPL 级别 1

注 1： 必须存在引导段才能使用备用中断向量表。

表2-5： dsPIC33EPXXXGS70X/80X配置位说明（续）

位域	说明
CTXT4<2:0>	指定备用工作寄存器 4 的中断优先级 （IPL）的位 111 = 保留 110 = 分配为 IPL 级别 7 101 = 分配为 IPL 级别 6 100 = 分配为 IPL 级别 5 011 = 分配为 IPL 级别 4 010 = 分配为 IPL 级别 3 001 = 分配为 IPL 级别 2 000 = 分配为 IPL 级别 1
BTMODE<1:0>	dsPIC33EPXXXGS70X/80X 引导配置位 11 = 器件工作在单分区闪存模式下 10 = 器件工作在双分区闪存模式下 01 = 器件工作在受保护双分区闪存模式下 00 = 保留，不要使用

注 1： 必须存在引导段才能使用备用中断向量表。

3.0 器件编程 ——ICSP

ICSP 模式是允许您对器件存储器进行读写操作的一种特殊编程协议。 ICSP 模式是对器件编程采用的最直接的方法，通过使用 PGCx 和 PGDx 引脚向器件串行发送控制代码和指令可实现该功能。 ICSP 模式还具有读取执行程序存储区的内容以确定编程执行程序 （PE）是否存在的功能，如果要使用增强型 ICSP 模式，该模式还能够将 PE 写入执行程序存储区。

在 ICSP 模式下，系统时钟总是来自 PGCx 引脚，而无论器件的振荡器配置位的设置如何。所有指令会先串行移入内部缓冲区，然后装入指令寄存器 （Instruction Register， IR）并执行。不会从内部存储器执行取程序操作。一次送入24位指令。PGDx用来移入数据，PGCx既用作串行移位时钟又用作 CPU 执行时钟。

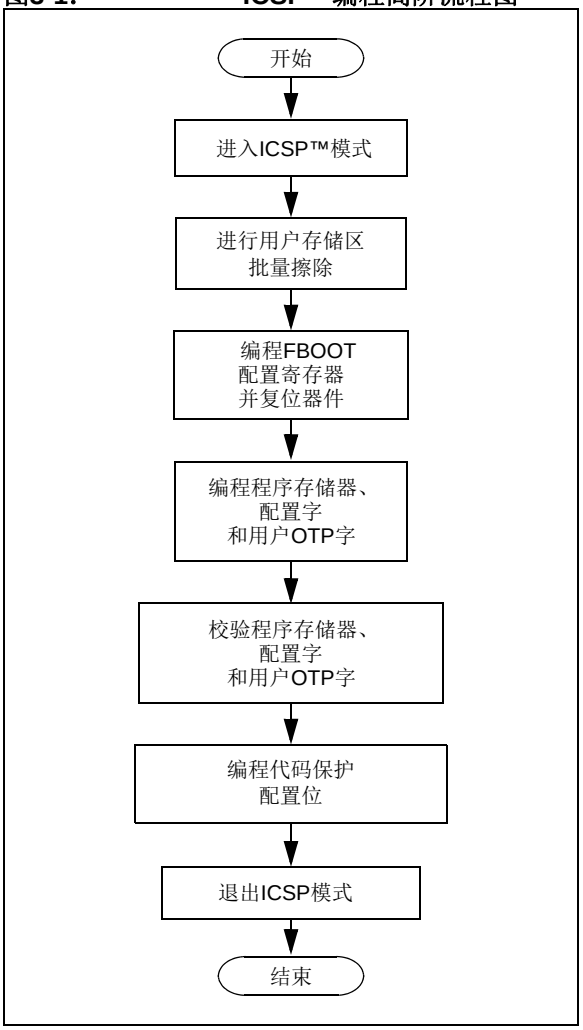
注 **1：** 在 ICSP 操作期间， PGCx 的工作频率不能超过 5 MHz。

2： 在编程时，ICSP 模式比增强型 ICSP 模式慢。

3.1 编程过程概述

图 3-1 高度概括了编程过程。进入 ICSP 模式后，首先执行的操作是对用户程序存储区进行批量擦除。然后，对代码存储区编程，并接着对器件配置位编程。随后，对代码存储区 （包括配置位）进行校验以确保编程成功。最后，如果需要，对代码保护配置位编程。

图3-1： ICSP™编程高阶流程图



3.2 进入 ICSP 模式

如图3-2所示，进入ICSP编程/校验模式需要四个步骤：

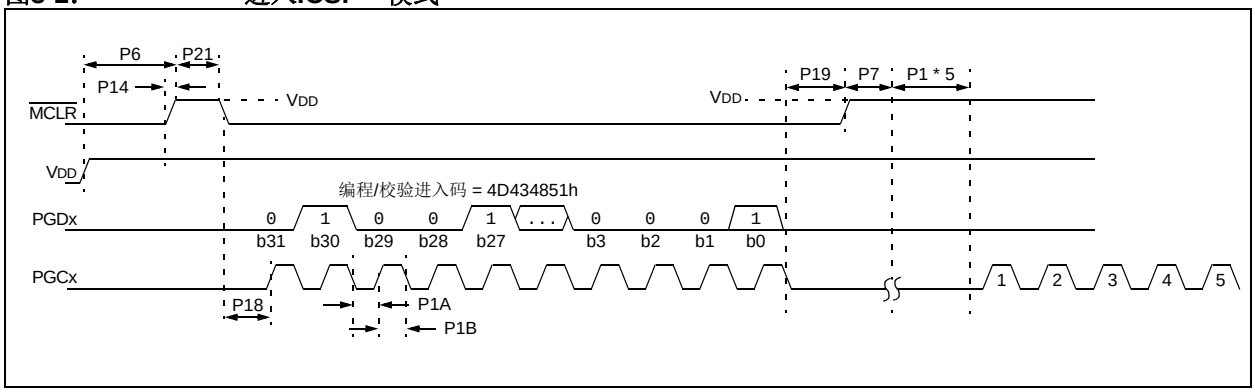
1. MCLR 短暂地驱动为高电平，然后再驱动为低电平 （P21）。
2. 将 32 位的密钥序列随时钟移入到 PGDx 引脚。必须经过至少P18的时间间隔才能将密钥序列移入到 PGDx 引脚。
3. MCLR 在指定周期 P19 期间保持低电平，然后驱动为高电平。
4. 在 P7 + 5 * P1 的延时之后，必须在 PGCx 引脚上产生 5 个时钟脉冲。

注：	如果 MCLR 引脚上存在电容，进入 ICSP 模式的高电平时间可能有所不同。

该密钥序列为一个特定的 32 位形式 0100 1101 0100 0011 0100 1000 0101 0001 （以其十六进制格式 4D434851h 记忆更为简便）。只有在密钥序列有效时，器件才能进入 ICSP 模式。必须首先移入高 4 位的最高有效位 （Most Significant bit， MSb）。

一旦成功进入 ICSP 模式，就能以串行方式访问程序存储器并对其编程。

图3-2： 进入ICSP™模式



3.3 ICSP 工作原理

进入 ICSP 模式后， CPU 为空闲。 CPU 执行由内部状态机控制。使用 PGCx 和 PGDx 引脚随时钟移入 4 位用来命令 CPU 的控制代码 （见表 3-1）。

SIX 控制代码用于发送指令给 CPU 执行，而 REGOUT 控制代码用于通过 VISI 寄存器从器件中读出数据。

表 3-1： ICSP™模 式下的 CPU控 制代码

4 位控制代码	助记符	说明
0000	SIX	移入 24 位指令并执行。
0001	REGOUT	移出 VISI 寄存器。
0010-1111	N/A	保留。

3.3.1 SIX串行指令执行

SIX 控制代码允许执行器件系列的汇编指令。当接收到 SIX 代码时，CPU 暂停 24 个时钟周期，在这段时间指令被移入内部缓冲区。一旦移入指令，状态机就允许其在接下来的 4 个 PGCx 时钟周期内执行。当执行接收到的指令时，状态机会同时移入下一条 4 位命令 （见图 3-3）。

注： PGDx 上的数据位在 PGCx 时钟脉冲的上升沿锁存。

3.3.1.1 SIX 指令执行和正常指令执行之间的差异

使用 SIX ICSP 命令执行指令和正常器件指令执行之间存在一些差异。因此，本规范中的代码示例可能与在正常器件操作期间执行相同操作所需的代码不一致。

差异包括：

- 双字指令需要两个SIX操作来随时钟移入所有必需数据。

双字指令的示例为 GOTO 和 CALL。

- 双周期指令需要两个 SIX 操作来完成。第一个 SIX 操作移入指令并开始执行。第二个 SIX 操作 （应移入 NOP 以避免丢失数据）提供完成指令执行所需的 CPU 时钟。

双周期指令的示例为表读 （TBLRD）和表写 （TBLWT）指令。

- CPU 不会自动停顿来应对流水线变化。当指令修改了某个寄存器， 而紧跟在 CPU 停顿后的指令使用该寄存器来进行间接寻址时，将产生 CPU 停顿。正常操作期间，在读取新数据时，CPU 会强制执行一条 NOP。考虑到这种情况，在使用 ICSP 时，如果要对最近修改过的寄存器进行任何间接引用，应在它的前面加上一条 NOP。

例如，MOV #0x0, W0 后跟 MOV [W0], W1 时，必须在其间插入一条 NOP。

如果双周期指令修改了被间接使用的寄存器，则需要后跟两条 NOP：一条 NOP 用于执行指令的后半部分，另一条 NOP 用于停顿 CPU 来修正流水线。

例如， TBLWTL [W0++], [W1] 应后跟两条 NOP。

- 在 ICSP 指令执行期间，即使并未使用闪存，器件程序计数器（Program Counter，PC）也会继续自动递增。因此， PC 可能会递增到指向无效的存储单元。

无效存储空间的示例是未实现的闪存地址或向量空间（存储单元：0x0 至 0x1FF）。

如果 PC 指向这些存储单元，会导致器件发生复位，可能会中断 ICSP 操作。为了防止这种情况，应通过定期执行指令来复位 PC，使之指向安全空间。实现该目的的最佳方法是执行 “GOTO 0x200”指令。

3.3.2 REGOUT串行指令执行

REGOUT 控制代码允许在 ICSP 模式下从器件中读出数据。它用于通过 PGDx 引脚随时钟移出器件中 VISI 寄存器的内容。接收到 REGOUT 控制代码后，CPU 在 8 个周期内保持空闲。在此之后，还需要额外的 16 个周期将数据移出 （见图 3-4）。

REGOUT 代码的独特之处在于将控制代码发送到器件时， PGDx 引脚为输入引脚。但是，控制代码处理完毕后，移出 VISI 寄存器中数据时 PGDx 引脚就会变成输出引脚。

注 1： VISI 的内容被移出后，器件会将 PGDx 保持为输出，直到接收到下一个时钟的第一个上升沿为止。

2： 数据在 PGCx 下降沿发生变化，在上升沿锁存。对于所有数据的发送，都是先发送最低有效位（Least Significant bit，LSb）。

图3-3： SIX串行指令执行

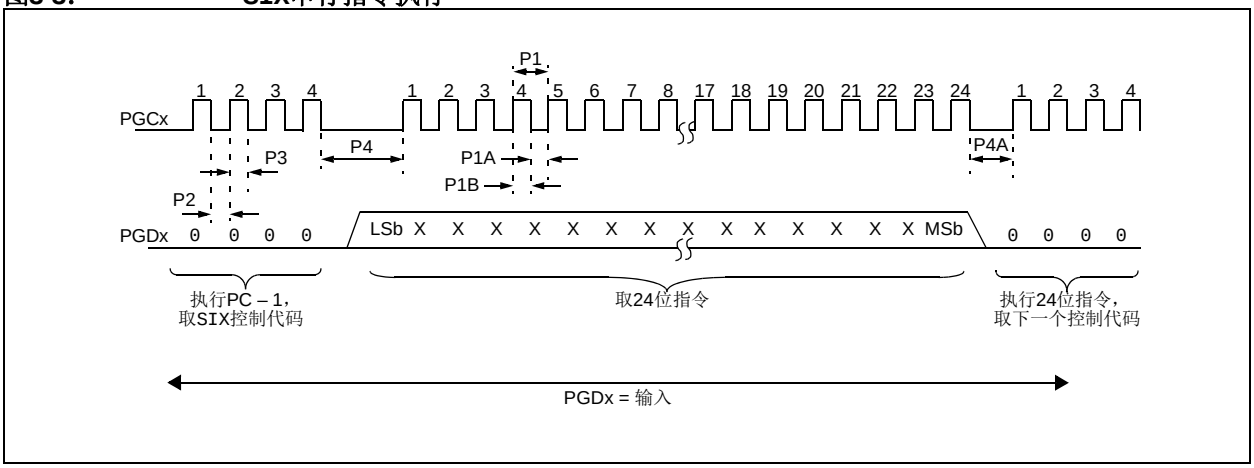
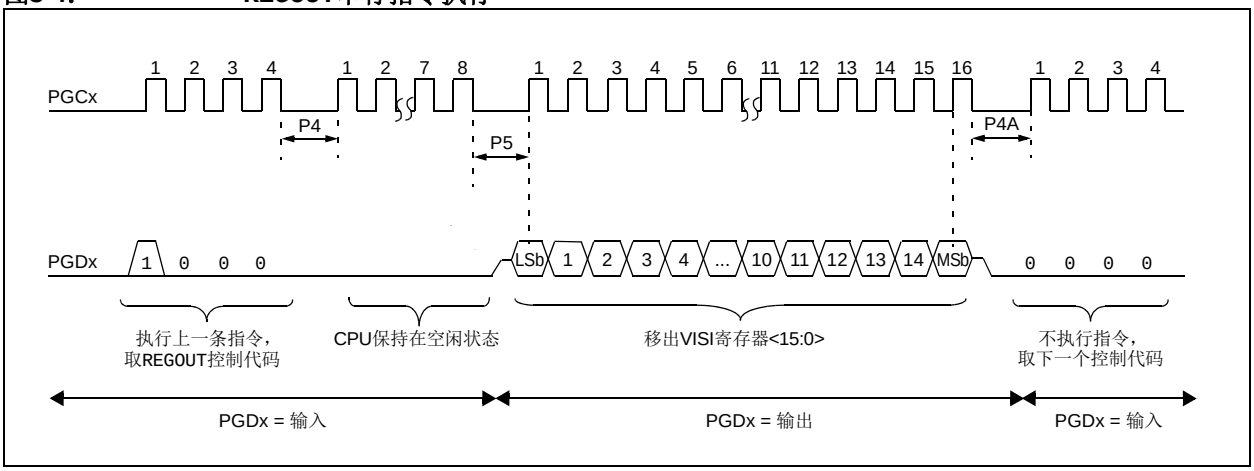


图3-4： REGOUT串行指令执行



3.4 在 ICSP 模式下对闪存编程

3.4.1 编程操作

对闪存执行的写 / 擦除操作是通过 NVMCON 寄存器控制的。编程的执行过程如下：设置 NVMCON 选择擦除操作 （表 3-2）或写操作 （表 3-3）的类型，并通过将 WREN （写使能）位 （NVMCON<14>）和 WR （写控制）位 （NVMCON<15>）置 1 启动编程。

在 ICSP 模式下，所有编程操作都采用自定时方式。在用户将 WR 控制位置 1 与编程操作完成时其被自动清零之间存在一段内部延时。关于与各种编程操作相关的延时的详细信息，请参见**第 10.0 节 “交流 / 直流特性和时序要求”**。

表3-2： NVMCON擦除操作

NVMCON 值	擦除操作
400Eh	仅批量擦除用户存储区 （不擦除器件 ID、执行程序存储区和 OTP 字）。
4003h	擦除程序存储区或执行程序存储区中的一页。
4004h	无效分区存储区擦除操作。

表3-3： NVMCON写操作

NVMCON 值	写操作
4001h	双字编程操作。
4008h	下一条 WR 命令将执行引导模式编程 （写入 FBOOT），然后编程双分区闪存签名 （FSIGN）位。必须先复位器件，新编程模式才会生效。该操作代码仅在 64K 存储器器件上使用。

3.4.2 启动和停止编程周期

为防止意外操作，必须将擦除 / 写启动序列写入 NVMKEY 寄存器以允许进行任何擦除或编程操作。启动编程序列后的两条指令应为 NOP。要启动擦除或写序列，必须完成以下步骤：

1. 将 55h 写入 NVMKEY 寄存器。
2. 将 AAh 写入 NVMKEY 寄存器。
3. 置 1 NVMCON 寄存器中的 WR 位。
4. 执行 3 条 NOP 指令。

所有擦除和写周期都是自定时的。应查询 WR 位以确定擦除或写周期是否已完成。

寄存器 3-1：　　NVMCON：非易失性存储器（ NVM）控制寄存器

R/SO-0(1)	R/W-0(1)	R/W-0(1)	R/W-0	R/C-0	R-0	R/W-0	R/C-0
WR	WREN	WRERR	NVMSIDL(2)	SFTSWP(6)	P2ACTIV(6)	RPDF(8)	URERR(8)
bit 15							bit 8

U-0	U-0	U-0	U-0	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/W-0(1)
—	—	—	—	NVMOP3(3,4)	NVMOP2(3,4)	NVMOP1(3,4)	NVMOP0(3,4)
bit 7							bit 0

图注：	C = 可清零位	SO = 只可置 1 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 = 启动闪存编程或擦除操作。操作是自定时的，一旦操作完成，该位即由硬件清零
0 = 编程或擦除操作完成，并处于停止状态

bit 14　**WREN：**写使能位 (1)
1 = 使能闪存编程 / 擦除操作
0 = 禁止闪存编程 / 擦除操作

bit 13　**WRERR：**写序列错误标志位 (1)
1 = 试图执行不合法的编程 / 擦除序列，或者发生终止 （试图将 WR 位置 1 时该位自动置 1）
0 = 编程 / 擦除操作正常完成

bit 12　**NVMSIDL：**NVM 空闲模式停止控制位 (2)
1 = 闪存稳压器在空闲模式期间进入待机模式
0 = 闪存稳压器在空闲模式期间工作

bit 11　**SFTSWP：**分区软交换状态位 (6)
1 = 已使用 BOOTSWP 指令 （软交换）成功交换分区
0 = 使用 BOOTSWP 指令后等待分区成功交换，或器件复位后根据 FBTSEQ 配置寄存器确定活动分区

bit 10　**P2ACTIV：**分区 2 活动状态位 (6)
1 = 分区 2 闪存映射到活动区域
0 = 分区 1 闪存映射到活动区域

注　1： 这些位只能在 POR 时复位。
2： 若此位置 1，可实现最小功耗 （IIDLE），并且在退出空闲模式时，在闪存开始工作之前会存在一个延时（TVREG）。
3： NVMOP<3:0> 的所有其他组合均未实现。
4： 在任意 NVM 操作正在进行时，执行 PWRSAV 指令会被忽略。
5： 执行该操作期间，会编程 4 字边界上的两个相邻字。
6： 仅在 dsPIC33EPXXXGS70X/80X 器件工作在双分区闪存模式下时可用。对于所有其他器件，这些位保留。
7： 具体引导模式取决于 FBOOT 编程数据的 bit <1:0>：
11 = 单分区闪存模式
10 = 双分区闪存模式
01 = 受保护双分区闪存模式
00 = 保留
8： 在 ICSP™ 模式下不使用。

寄存器 3-1：　NVMCON：非易失性存储器（ NVM）控制寄存器（ 续）

bit 9　**RPDF：**行编程数据格式控制位 (8)
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

bit 8　**URERR：**行编程数据不足错误标志位 (8)
1 = 行编程操作已由于数据不足错误而终止
0 = 未发生数据不足

bit 7-4　**未实现：**读为 0

bit 3-0　**NVMOP<3:0>：**NVM 操作选择位 (1,3,4)
1111 = 保留
1110 = 用户存储区批量擦除操作
1010 = 保留
1001 = 保留
1000 = 双分区闪存模式下的引导存储区双字编程操作 (7)
0101 = 保留
0100 = 非活动分区存储区擦除操作
0011 = 存储器页擦除操作
0010 = 存储器行编程操作 (8)
0001 = 存储器双字编程操作 (5)
0000 = 保留

注　1： 这些位只能在 POR 时复位。
2： 若此位置 1，可实现最小功耗 （IIDLE），并且在退出空闲模式时，在闪存开始工作之前会存在一个延时（TVREG）。
3： NVMOP<3:0> 的所有其他组合均未实现。
4： 在任意 NVM 操作正在进行时，执行 PWRSAV 指令会被忽略。
5： 执行该操作期间，会编程 4 字边界上的两个相邻字。
6： 仅在 dsPIC33EPXXXGS70X/80X 器件工作在双分区闪存模式下时可用。对于所有其他器件，这些位保留。
7： 具体引导模式取决于 FBOOT 编程数据的 bit <1:0>：
11 = 单分区闪存模式
10 = 双分区闪存模式
01 = 受保护双分区闪存模式
00 = 保留
8： 在 ICSP™ 模式下不使用。

3.5 擦除程序存储器

使用批量擦除来擦除整个代码存储区的过程如图 3-5 所示。

图 3-6 显示了擦除代码存储区中的一页的过程。

表 3-4 和表 3-5 分别说明了批量擦除和页擦除的 ICSP 编程过程。

注 1： 在向程序存储器写入任何数据之前必须先将其擦除。

2： 对于页擦除操作，应根据表 3-2 适当修改 NVMCON 的值。NVMADR/U 寄存器应指向要擦除的页中的任意存储单元。

图 3-5： 批量擦除流程图

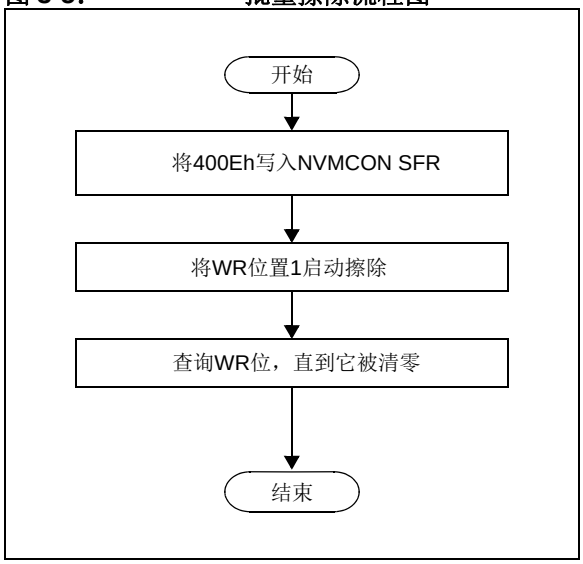


图 3-6： 页擦除流程图

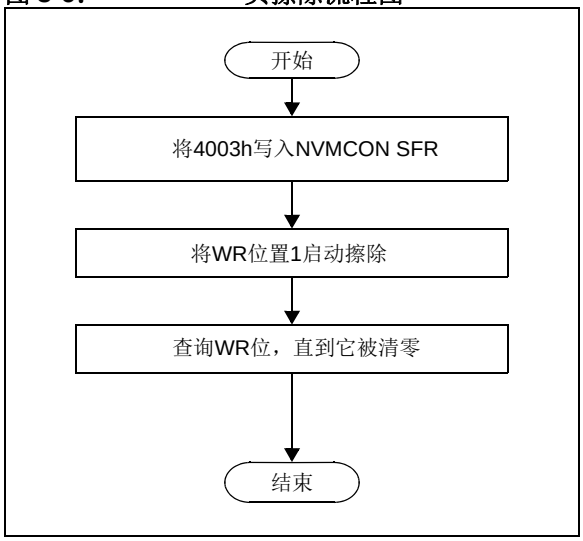


表 3-4：　　批量擦除代码存储区的串行指令执行

命令 （二进制）	数据 （十六进制）	说明
第 1 步：退出复位向量。		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 2 步：设置 NVMCON 寄存器以擦除所有用户程序存储区。		
0000	2400EA	MOV #0x400E, W10
0000	88394A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP
第 3 步：启动擦除周期。		
0000	200551	MOV #0x55, W1
0000	883971	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	883971	MOV W1, NVMKEY
0000	A8E729	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 4 步：产生用于完成用户存储区批量擦除操作的时钟脉冲，直到 WR 位清零为止。		
0000	000000	NOP
0000	803940	MOV NVMCON, W0
0000	000000	NOP
0000	887C40	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
—	—	重复执行直到 WR 位清零为止。

表 3-5：　　擦除代码存储区一页的串行指令执行

命令 （二进制）	数据 （十六进制）	说明
第 1 步：退出复位向量。		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 2 步：设置 NVMADRU/NVMADR 寄存器对以指向要擦除的正确页。		
0000	2xxxx3	MOV #DestinationAddress<15:0>, W3
0000	2xxxx4	MOV #DestinationAddress<23:16>, W4
0000	883953	MOV W3, NVMADR
0000	883964	MOV W4, NVMADRU
第 3 步：设置 NVMCON 寄存器以擦除执行程序存储区的第一页。		
0000	24003A	MOV #0x4003, W10
0000	88394A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP
第 4 步：启动擦除周期。		
0000	200551	MOV #0x55, W1
0000	883971	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	883971	MOV W1, NVMKEY
0000	A8E729	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 5 步：产生用于完成页擦除操作的时钟脉冲，直到 WR 位清零为止。		
0000	000000	NOP
0000	803940	MOV NVMCON, W0
0000	000000	NOP
0000	887C40	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
—	—	重复执行直到 WR 位清零为止。

3.6 编程 FBOOT 配置寄存器

在编程代码存储区、配置寄存器和用户OTP之前，必须先编程FBOOT配置寄存器（位于地址0x801000处），才能将器件配置为双分区闪存模式之一。BTMODE<1:0>位不能写为00（保留）或11（单分区闪存）。必须擦除FBOOT寄存器来设置单分区闪存模式。请参见表3-6了解关于如何写入FBOOT配置寄存器的详细信息。

表3-6： 写FBOOT配置寄存器的串行指令执行

命令（二进制）	数据（十六进制）	说明
第 1 步：退出复位向量。		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 2 步：初始化 TBLPAG 寄存器以写入锁存器。		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
第 3 步：将待编程的下两个配置字装入 W0:W1。		
0000	2xxxx0	MOV #<Config lower word data>, W0
0000	2xxxx1	MOV #<Config upper word data>, W1
第 4 步：设置写指针 （W3）并装载写锁存器。		
0000	EB0030	CLR W6
0000	000000	NOP
0000	BB0B00	TBLWTL W0, [W6]
0000	000000	NOP
0000	000000	NOP
0000	BB9B01	TBLWTH W1, [W6++]
0000	000000	NOP
0000	000000	NOP

表3-6：　　写FBOOT配置寄存器的串行指令执行（续）

命令 （二进制）	数据 （十六进制）	说明
第 5 步：设置 NVMCON 寄存器以编程 FBOOT。		
—	—	; 屏蔽 FBOOT<1:0> = 00 和 FBOOT<1:0> = 11 值， ; 这两个值是保留值或擦除后的默认单分区值， ; 均不可进行编程。 ; 该代码将 WREN （NVMCON<14>）清零，因此不会发生 ; 写操作。
0000	A31000	BTST.C　W0, #1
0000	B08000	ADDC　#0, W0
0000	DD004E	SL　W0, #14, W0
0000	700068	IOR　W0, #0x08, W0
0000	883940	MOV　W0, NVMCON
0000	000000	NOP
0000	000000	NOP
第 6 步：启动写周期。		
0000	200551	MOV　#0x55, W1
0000	883971	MOV　W1, NVMKEY
0000	200AA1	MOV　#0xAA, W1
0000	883971	MOV　W1, NVMKEY
0000	A8E729	BSET　NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 7 步：等待编程操作完成并确认 WR 位清零。		
—	—	在外部计时 “P13”毫秒 （见**第 10.0 节 “交流 / 直流特性和时序要求”**），以便编程操作有足够时间可以完成。
0000	000000	NOP
0000	803940	MOV　NVMCON, W0
0000	000000	NOP
0000	887C40	MOV　W0, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO　0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
—	—	重复执行直到 WR 位清零为止。
第 8 步：复位器件。		

3.7 写代码存储区

图 3-8 概括说明了如何写入代码存储区。

双字写操作每次使用两个指令字来对代码存储区进行编程。两个字装入到位于地址 FA0000h 处的写锁存器，目标地址必须装入 NVMADRU/NVMADR 寄存器对。接下来，通过将 WR 位置 1 来启动写序列。然后，必须通过检查 WR 位来确定序列是否完成。继续执行该过程，直到完成所有数据的编程。表 3-7 给出了 ICSP 编程的详细信息。

装入编程锁存器的数据必须采用打包格式，如图 3-7 所示。

图 3-7： 指令字的打包格式

15	8	7	0
LSW1			
MSB2		MSB1	
LSW2			

LSWx：指令字的低 16 位
MSBx：指令字的最高有效字节

注 1： 当传输的指令字数为奇数时，MSB2 为零，且不能发送 LSW2。

2： 在重新编程双字对中的任意字之前，编程器必须先擦除该字所在的闪存页。

图 3-8： 代码存储区编程流程图

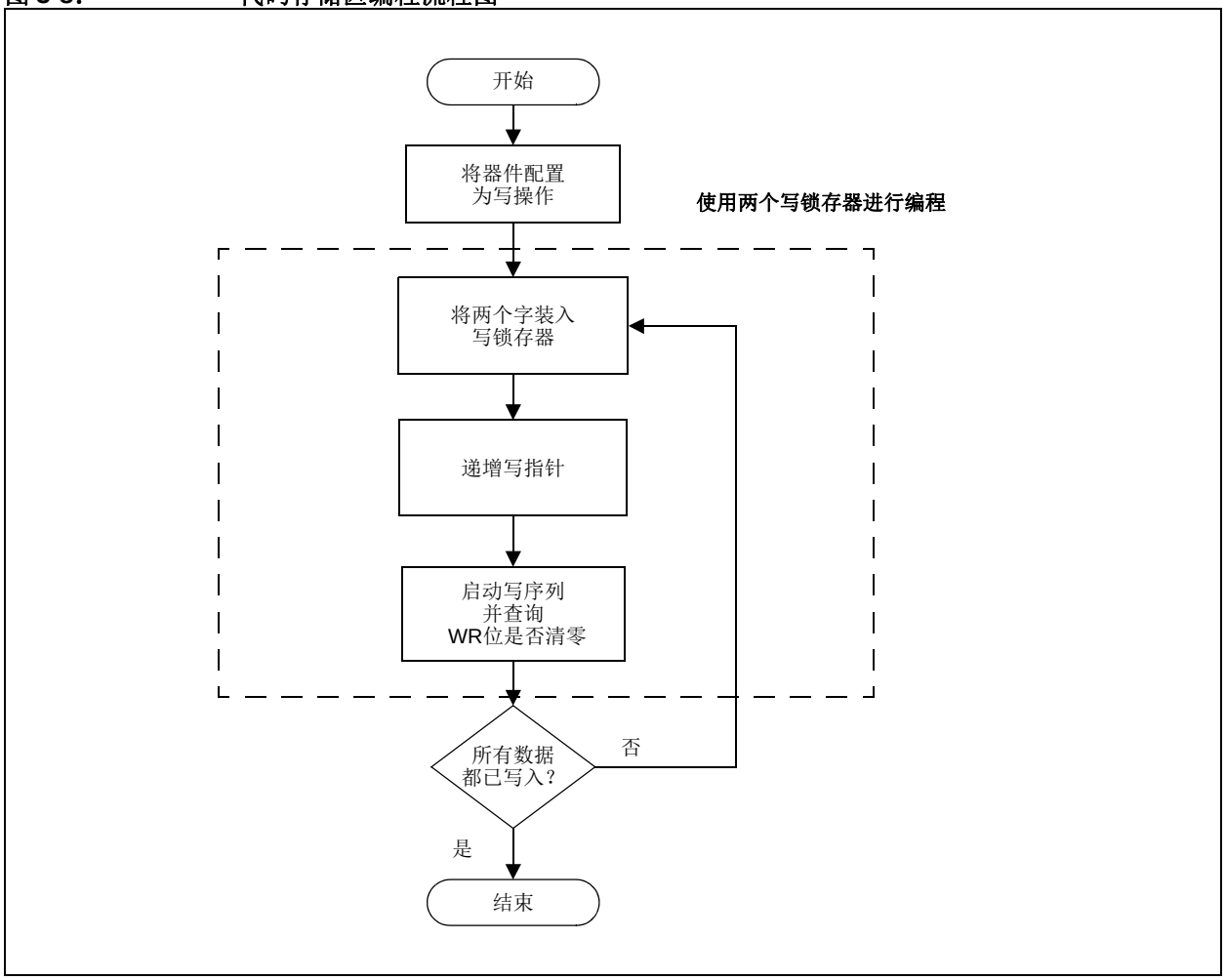


表 3-7：　　代码存储区编程的串行指令执行：双字锁存器写操作

命令 （二进制）	数据 （十六进制）	说明
第 1 步：退出复位向量。		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 2 步：初始化 TBLPAG 寄存器以写入锁存器。		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
第 3 步：将待编程的下 2 个打包指令字装入 W0:W2。		
0000	2xxxx0	MOV #<LSW0>, W0
0000	2xxxx1	MOV #<MSB1:MSB0>, W1
0000	2xxxx2	MOV #<LSW1>, W2
第 4 步：设置读指针 （W6）和写指针 （W7）并装载 （下一组）写锁存器。		
0000	EB0300	CLR W6
0000	000000	NOP
0000	EB0380	CLR W7
0000	000000	NOP
0000	BB0BB6	TBLWTL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	BBDBB6	TBLWTH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BBEBB6	TBLWTH.B [W6++], [++W7]
0000	000000	NOP
0000	000000	NOP
0000	BB0B96	TBLWTL.W [W6], [W7]
0000	000000	NOP
0000	000000	NOP
第 5 步：设置 NVMADRU/NVMADR 寄存器对以指向正确的地址。		
0000	2xxxx3	MOV #DestinationAddress<15:0>, W3
0000	2xxxx4	MOV #DestinationAddress<23:16>, W4
0000	883953	MOV W3, NVMADR
0000	883964	MOV W4, NVMADRU
第 6 步：设置 NVMCON 寄存器来编程 2 个指令字。		
0000	24001A	MOV #0x4001, W10
0000	000000	NOP
0000	88394A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP

表 3-7：　　代码存储区编程的串行指令执行：双字锁存器写操作（续）

命令 （二进制）	数据 （十六进制）	说明
第 7 步：启动写周期。		
0000	200551	MOV　　#0x55, W1
0000	883971	MOV　　W1, NVMKEY
0000	200AA1	MOV　　#0xAA, W1
0000	883971	MOV　　W1, NVMKEY
0000	A8E729	BSET　　NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 8 步：产生用于完成编程操作的时钟脉冲，直到 WR 位清零为止。		
0000	000000	NOP
0000	803940	MOV　　NVMCON, W0
0000	000000	NOP
0000	887C40	MOV　　W0, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO　　0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
—	—	重复执行直到 WR 位清零为止。
第 9 步：重复执行第 3 步至第 8 步，直到完成所有代码存储区的编程。		

3.8 写配置位

写配置位的过程与写代码存储区的过程相似，但是每次只能编程两个 24 位字。

要在编程这些配置位后更改其值，必须先擦除器件（如**第 3.5 节 “擦除程序存储器”**所述），然后再编程为期望值。可以通过在代码保护配置位中写入 0 来使能代码保护。

表 3-6 给出了写配置位的 ICSP 编程详细信息。

要通过在执行写操作后读配置位来校验数据，应在开始将代码保护位编程为 1 以确保校验正常执行。可在校验完成后，通过将字写入相应的配置寄存器将代码保护位编程为 0。

注： 由于每个配置寄存器单元后都跟着一个未用存储单元，因此可以在每个配置寄存器双字对的第二个字中写入 0xFFFFFF。

表 3-8： 写配置字的串行指令执行

命令（二进制）	数据（十六进制）	说明
第 1 步：退出复位向量。		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 2 步：初始化 TBLPAG 寄存器以写入锁存器。		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
第 3 步：将待编程的下两个配置字装入 W0:W1。		
0000	2xxxx0	MOV #<Config1 lower word data>, W0
0000	2xxxx1	MOV #<Config1 upper word data>, W1
0000	2xxxx2	MOV #<Config2 lower word data>, W2
0000	2xxxx3	MOV #<Config2 upper word data>, W3
第 4 步：设置写指针 （W3）并装载写锁存器。		
0000	EB0300	CLR W6
0000	000000	NOP
0000	BB0B00	TBLWTL W0, [W6]
0000	000000	NOP
0000	000000	NOP
0000	BB9B01	TBLWTH W1, [W6++]
0000	000000	NOP
0000	000000	NOP
0000	BB0B02	TBLWTL W2, [W6]
0000	000000	NOP
0000	000000	NOP
0000	BB9B03	TBLWTH W3, [W6++]
0000	000000	NOP
0000	000000	NOP
第 5 步：设置 NVMADRU/NVMADR 寄存器对以指向正确的配置字地址。		
0000	2xxxx4	MOV #DestinationAddress<15:0>, W4
0000	2xxxx5	MOV #DestinationAddress<23:16>, W5
0000	883954	MOV W4, NVMADR
0000	883965	MOV W5, NVMADRU

表 3-8：　　写配置字的串行指令执行（续）

命令 （二进制）	数据 （十六进制）	说明
第 6 步：设置 NVMCON 寄存器来编程 2 个指令字。		
0000	24001A	MOV　　#0x4001, W10
0000	000000	NOP
0000	88394A	MOV　　W10, NVMCON
0000	000000	NOP
0000	000000	NOP
第 7 步：启动写周期。		
0000	200551	MOV　　#0x55, W1
0000	883971	MOV　　W1, NVMKEY
0000	200AA1	MOV　　#0xAA, W1
0000	883971	MOV　　W1, NVMKEY
0000	A8E729	BSET　　NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 8 步：产生用于完成编程操作的时钟脉冲，直到 WR 位清零为止。		
0000	000000	NOP
0000	803940	MOV　　NVMCON, W0
0000	000000	NOP
0000	887C40	MOV　　W0, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO　　0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
—	—	重复执行直到 WR 位清零为止。
第 9 步：重复执行第 3 步至第 8 步，直到所有配置寄存器都被编程为止。		

3.9 写 OTP 字

写OTP字的过程与写代码存储区的过程类似，但是OTP字只能写入一次。无法将 0 编程为 1，但 OTP 字可以从 1 编程为 0。关于用户 OTP 字的位置，请参见图 2-8 至图 2-11。

OTP 存储区必须每次写入一个双字。在写入 OTP 存储区中的任何双字之前，必须先读取它的所有存储单元。只有整个用户 OTP 存储区处于擦除状态时 （即，只有 OTP 存储区中的每个存储单元的值均为 0xFFFFFF 时），才能对 OTP 存储区编程。

3.10 读 OTP 字

读 OTP 字的过程与读代码存储区的过程类似。每次读取多个 OTP 字中的一个。

3.11 读代码存储区

读代码存储区是通过执行一系列 TBLRD 指令和使用 REGOUT 命令随时钟移出数据完成的。

表 3-9 给出了读代码存储区的 ICSP 编程详细步骤。

为了使读时间最短，将使用与 PE 使用的相同数据打包格式。关于数据打包格式的更多详细信息，请参见**第 6.2 节 “编程执行程序命令”**。

表 3-9： 读代码存储区的串行指令执行

命令 （二进制）	数据 （十六进制）	说明
第 1 步：退出复位向量。		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 2 步：为执行 TBLRD 指令初始化 TBLPAG 寄存器和读指针 （W6）。		
0000	200xx0	MOV #<SourceAddress23:16>, W0
0000	8802A0	MOV W0, TBLPAG
0000	2xxxx6	MOV #<SourceAddress15:0>, W6

表 3-9： 读代码存储区的串行指令执行（续）

命令（二进制）	数据（十六进制）	说明
第 3 步：初始化写指针 （W7）并将代码存储区的下 4 个存储单元内容存储到 W0:W5。		
0000	EB0380	CLR W7
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BA0BB6	TBLRDL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP

表 3-9：　　读代码存储区的串行指令执行（续）

命令（二进制）	数据（十六进制）	说明
第 4 步：使用 VISI 寄存器和 REGOUT 命令输出 W0:W5 的内容。		
0000	887C40	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	887C41	MOV W1, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	887C42	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	887C43	MOV W3, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	887C44	MOV W4, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	887C45	MOV W5, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
第 5 步：复位器件的内部 PC。		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 6 步：重复执行第 3 步到第 5 步，直到读完所有需要的代码存储单元为止。		

3.12 读配置寄存器

读配置位的过程与读代码存储区类似。每次读取多个配置字中的一个。

表 3-10 给出了读配置位的 ICSP 编程详细信息。

表 3-10： 读配置字的串行指令执行

命令（二进制）	数据（十六进制）	说明
第 1 步：退出复位向量。		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 2 步：为执行 TBLRD 指令初始化 TBLPAG 寄存器、写指针 （W7）和读指针 （W6）。		
0000	200xx0	MOV #<Address23:16>, W0(1)
0000	20F887	MOV #<VISI>, W7
0000	8802A0	MOV W0, TBLPAG
0000	2xxxx6	MOV #<Address15:0>, W6(1)
第 3 步：存储配置寄存器并发送 VISI 寄存器的内容。		
0000	000000	NOP
0000	BA8B96	TBLRDH [W6], [W7]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	BA0B96	TBLRDL [W6], [W7]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
第 4 步：重复执行第 1 步到第 3 步，直到读完所有配置寄存器为止。		

注 1： 必须在读取任何其他配置字之前，先读取 FBOOT 寄存器。必须通过编程器根据器件的当前分区闪存模式确定配置字地址。

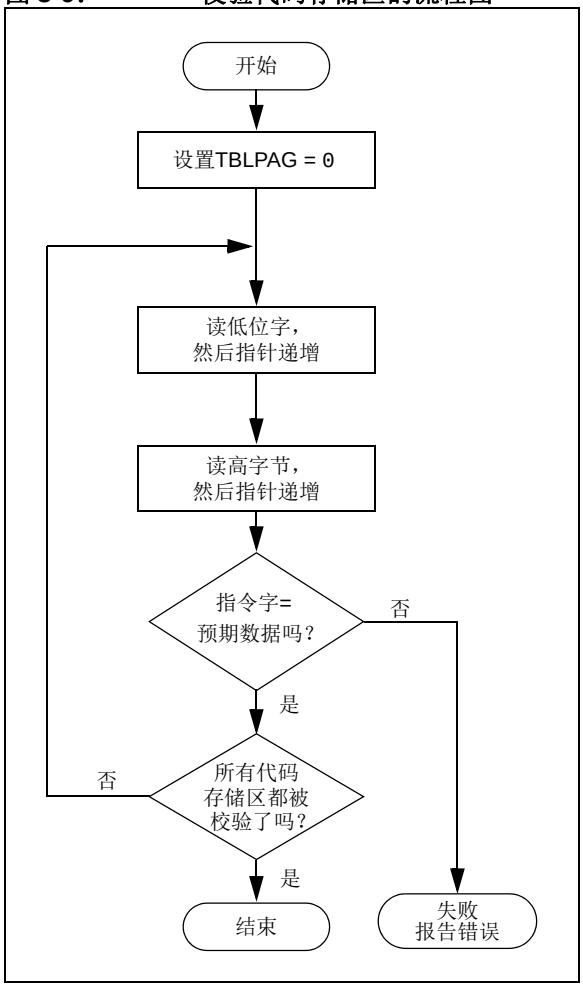
3.13 校验代码存储区和配置字

校验的步骤涉及读回代码存储空间并将读到的值与存储在编程器缓冲区中的副本作比较。使用其余代码校验配置字。

校验过程如图 3-9 所述。读取指令的低位字，然后读取高位字的低字节，并将其与编程器缓冲区中存储的指令作比较。关于读取代码存储区的实现细节，请参见**第 3.11 节 “读代码存储区”**。

注：	由于配置字包含器件代码保护位，如果要使能代码保护，应在写入代码存储区后立即对其进行校验。这是因为如果代码保护位清零后发生了器件复位，将无法读取或校验器件。

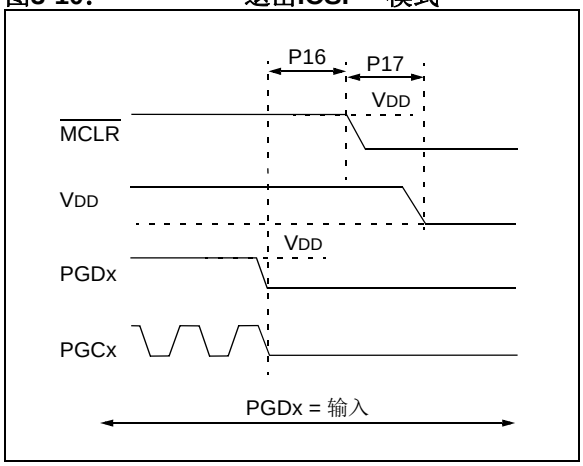
图 3-9： 校验代码存储区的流程图



3.14 退出 ICSP 模式

通过将 VDD 从 $\overline{\text{MCLR}}$ 引脚移除可退出编程 / 校验模式，如图 3-10 所示。退出操作的唯一要求是移除 VDD 要在 PGCx 和 PGDx 引脚上最后一个时钟和编程信号后的 P16 时间间隔后进行。

图3-10： 退出ICSP™模式



4.0 器件编程——增强型ICSP

本节讨论通过增强型 ICSP 和编程执行程序（PE）对器件编程。 PE 存放在执行程序存储区 （独立于代码存储区）中，当进入增强型 ICSP 编程模式时执行编程执行程序。 PE 使用一个简单的命令集和通信协议为编程器（主设备）提供了对 dsPIC33EPXXXGS70X/80X 系列器件进行编程和校验的机制。 PE 可提供以下几项基本功能：

- 读存储器
- 擦除存储器
- 对存储器编程
- 空白检查

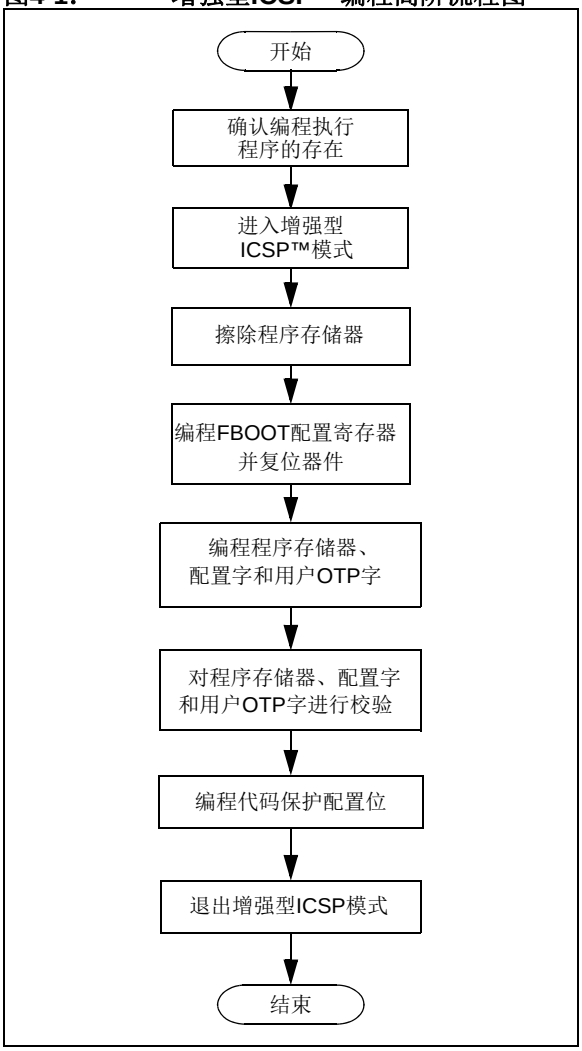
PE 执行擦除、编程和校验器件所需的低级任务。这允许编程器通过发出相应的命令和数据来对器件编程。**第 6.2 节“编程执行程序命令”**提供了每条命令的详细说明。

注：	PE 使用器件的数据 RAM 来存放变量和执行程序。运行 PE 后，数据 RAM 中的内容就无法确定了。

4.1 编程过程概述

图 4-1 高度概括了编程过程。首先，必须确定 PE 是否存放在执行程序存储区中，然后进入增强型ICSP模式。接着擦除程序存储器，并编程和校验程序存储器和配置字。最后，编程代码保护配置位 （如果需要），退出增强型 ICSP 模式。

图4-1： 增强型ICSP™编程高阶流程图



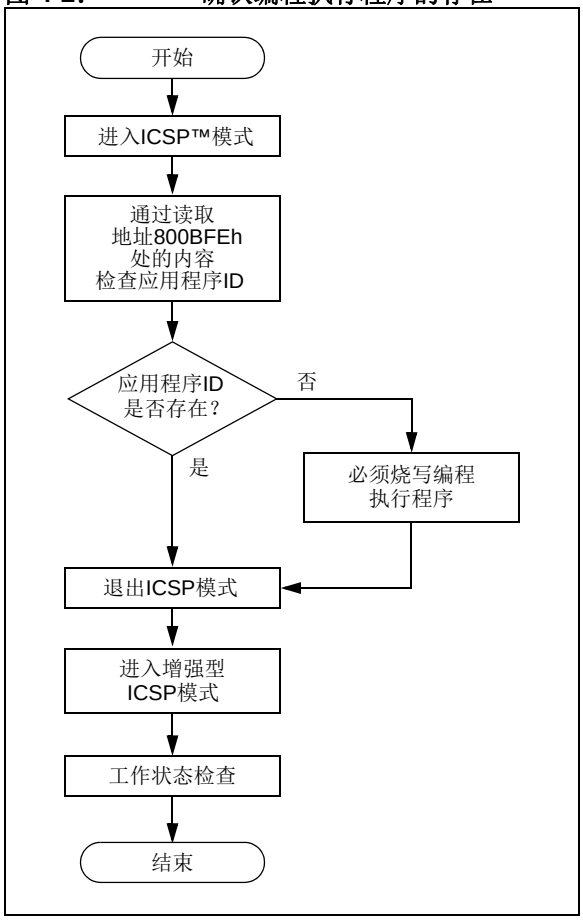
4.2 确认编程执行程序的存在

在开始编程之前，编程人员必须确认 PE 存放在执行程序存储区中。图 4-2 给出了该任务的流程。

首先进入 ICSP 模式，然后读取存储在执行程序存储区中的唯一应用程序 ID 字。如果 PE 已位于执行程序存储区中，则可以读取正确的应用程序 ID 字 0xDF 并正常编程。但是，如果应用程序 ID 字不存在，则必须通过使用**第 5.0 节 “将编程执行程序编程到存储区”**中所述的方法将 PE 烧写到执行程序代码存储区中。

第 3.0 节 “器件编程 ——ICSP”介绍了 ICSP 编程方法。**第 4.3 节 “读应用程序 ID 字”**介绍了在 ICSP 模式下读取应用程序 ID 字的过程。

图 4-2： 确认编程执行程序的存在



4.3 读应用程序 ID 字

应用程序 ID 字存储在执行程序代码存储区内地址 800BFEh 中。要读取该存储单元，必须使用 SIX 控制代码将该程序存储单元中的内容移入 VISI 寄存器。然后，必须使用 REGOUT 控制代码随时钟将 VISI 寄存器的内容移出器件。必须串行发送给器件以执行该操作的相应控制和指令代码如表 4-1 中所示。

编程器随时钟取出应用程序 ID 字后，就必须对其进行检查。如果应用程序 ID 的值为 0xDF，表明 PE 位于执行程序存储区中，可以采用**第 4.0 节 “器件编程 —— 增强型 ICSP”**中所述的机制对器件编程。但是，如果应用程序 ID 为其他值，则表明 PE 不在相应的存储区中；必须在对器件编程前先将编程执行程序装入存储区中。**第 5.0 节 “将编程执行程序编程到存储区”**中介绍了将 PE 装入存储区的过程。

表4-1： 读应用程序ID字的串行指令执行

命令 （二进制）	数据 （十六进制）	说明
第 1 步：退出复位向量。		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 2 步：为执行 TBLRD 指令初始化 TBLPAG 寄存器和读指针 （W0）。		
0000	200800	MOV #0x80, W0
0000	8802A0	MOV W0, TBLPAG
0000	20BFE0	MOV #0xBFE, W0
0000	20F881	MOV #VISI, W1
0000	000000	NOP
0000	BA0890	TBLRDL [W0], [W1]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 3 步：使用 REGOUT 命令输出 VISI 寄存器的内容。		
0001	<VISI>	随时钟移出 VISI 寄存器的内容。

4.4 进入增强型 ICSP 模式

如图 4-3 所示，进入增强型 ICSP 编程 / 校验模式需要三个步骤：

1. 将 MCLR 引脚短暂地驱动为高电平，然后再驱动为低电平。
2. 将 32 位密钥序列随时钟移入到 PGDx 引脚。
3. 然后将MCLR驱动为高电平并保持一段指定的时间。

该密钥序列为一个特定的 32 位形式 0100 1101 0100 0011 0100 1000 0101 0000（以其十六进制格式 4D434850h 记忆更为简便）。只有在密钥序列有效时，器件才能进入编程 / 校验模式。必须首先移入高 4 位的最高有效位 （MSb）。

一旦密钥序列完全移入，就必须将 VDD 施加到 MCLR 并保持该电压，只要需要保持在编程 / 校验模式。必须经过至少 P19、 P7 和 P1 * 5 的时间间隔才能将数据移到 PGDx 引脚。在 P7 之前出现在 PGDx 引脚上的信号被认定为无效。

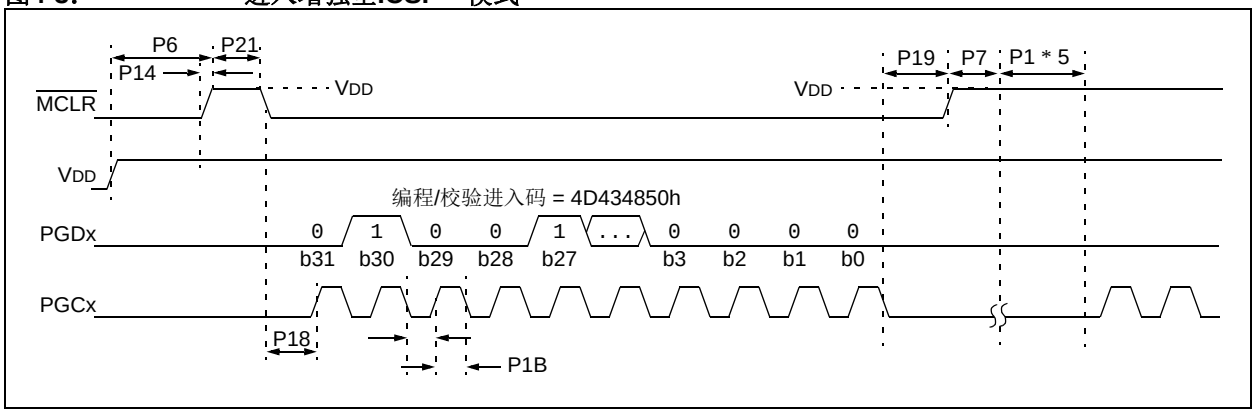
4.5 空白检查

术语“空白检查”的含义是校验器件是否已被成功擦除且没有已编程的存储单元。空白或已擦除的存储单元始终读为 1。

器件 ID 寄存器 （FF0000h:FF0002h）可被空白检查忽略，因为此区域存储了不可擦除的器件信息。另外，所有未实现的存储空间和校准寄存器都应该被空白检查忽略。

QBLANK 命令用于空白检查。该命令通过检测代码存储区，确定代码存储区是否已被擦除。将返回“BLANK”（空白）或“NOT BLANK”（非空白）响应。如果确定器件非空白，则必须在尝试对芯片进行编程前先将其擦除。

图4-3： 进入增强型ICSP™模式



4.6 代码存储区编程

4.6.1 编程方法

有两条命令可用于使用PE来编程代码存储区。PROG2W命令将两个 24 位指令字编程到以指定地址开始的程序存储区并进行校验。另一条速度更快的命令PROGP支持将最多 128 个 24 位指令字编程到以指定地址开始的程序存储器并进行校验。关于这两条命令的完整说明，请参见**第 6.0 节 “编程执行程序”**。

图 4-4 和图 4-5 显示了 PROG2W 和 PROGP 命令的编程方法。在这两个示例中，编程dsPIC33EPXXXGS70X/80X器件的 22464 指令字。

注：	编程自举程序时，建议将自举程序代码编程到代码存储区的第二页（例如，400h 处）中。尝试擦除第一页时，位于起始地址之一（例如， 200h）的自举程序可能会擦除自身。

图 4-4： 双字编程的流程图

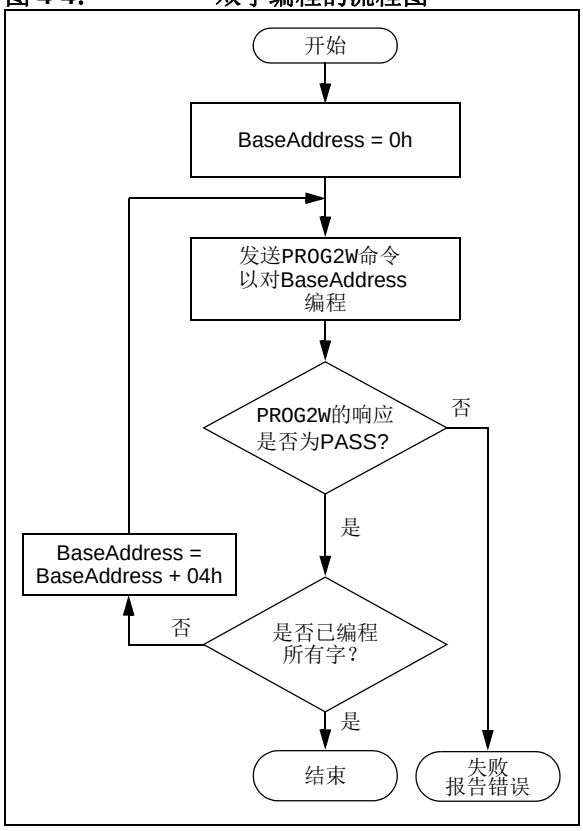
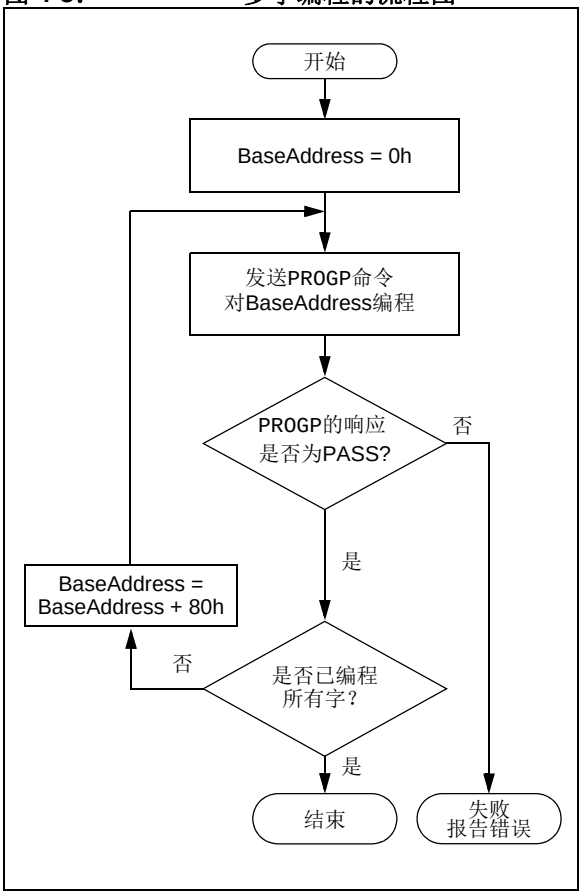


图 4-5： 多字编程的流程图

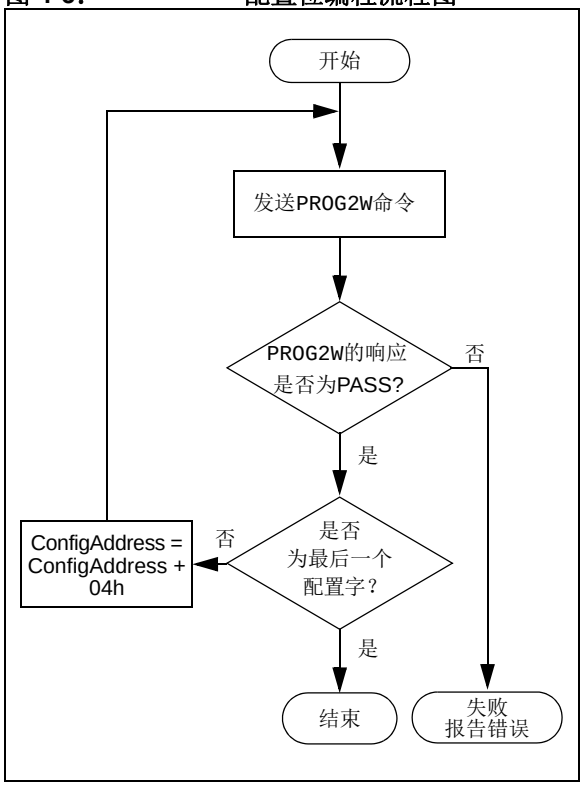


4.7 配置位编程

使用 PROG2W 命令一次编程一个配置位。该命令指定配置数据和地址。编程配置位时，任何未实现位都必须编程为 1。

需要多条 PROG2W 命令来对所有配置位进行编程。配置位编程的流程图如图 4-6 所示。

图 4-6： 配置位编程流程图



4.8 编程校验

一旦代码存储区编程完毕，就可以对存储区的内容进行校验以确保编程成功。校验需要对代码存储区执行读回操作，并将读回的数据与保存在编程器缓冲区中的副本作比较。

READP命令可对所有已编程的代码存储区和配置字执行读回操作。

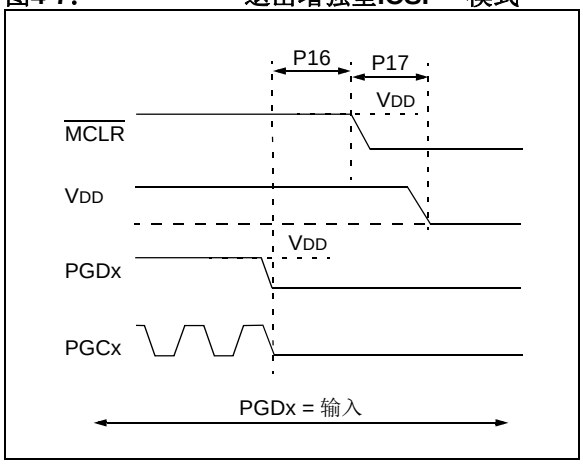
或者，也可以在对整个器件编程完毕后，让编程器通过计算校验和执行校验。

关于计算校验和的更多信息，请参见**第 9.0 节 “校验和计算”**。

4.9 退出增强型 ICSP 模式

通过将 VDD 从 MCLR 引脚移除可退出编程 / 校验模式，如图 4-7 所示。退出操作的唯一要求是移除 VDD 要在 PGCx 和 PGDx 引脚上最后一个时钟和编程信号后的 P16 时间间隔后进行。

图4-7： 退出增强型ICSP™模式



5.0 将编程执行程序编程到存储区

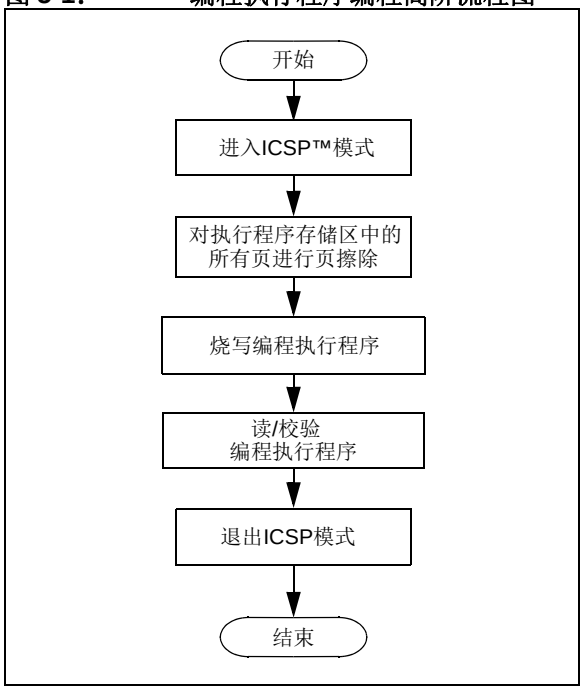
注： 编程执行程序（PE）可从 Microchip 网站（www.microchip.com）上每款器件的页面获取。

5.1 概述

如果已确定 PE 不在执行程序存储区中（如**第 4.2 节“确认编程执行程序的存在”**中所述），则必须将 PE 烧写到执行程序存储区中。

图5-1给出了将PE烧写到执行程序存储区的大概过程。首先，必须进入 ICSP 模式，并擦除执行程序存储区和用户存储区。然后才能烧写和校验PE。最后，退出ICSP模式。

图 5-1： 编程执行程序编程高阶流程图



5.2 擦除执行程序存储区

擦除执行程序存储区每个页的过程与擦除程序存储器的过程类似，如图 5-2 所示。该过程由以下步骤组成：将 NVMCON 设置为 4003h，然后执行编程周期。请注意，也使用该操作擦除程序存储器。

表 5-1 给出了批量擦除存储器的 ICSP 编程过程。

注： 必须始终先擦除 PE 存储区，再对其进行编程，如图 5-1 所述。

图 5-2： 页擦除流程图

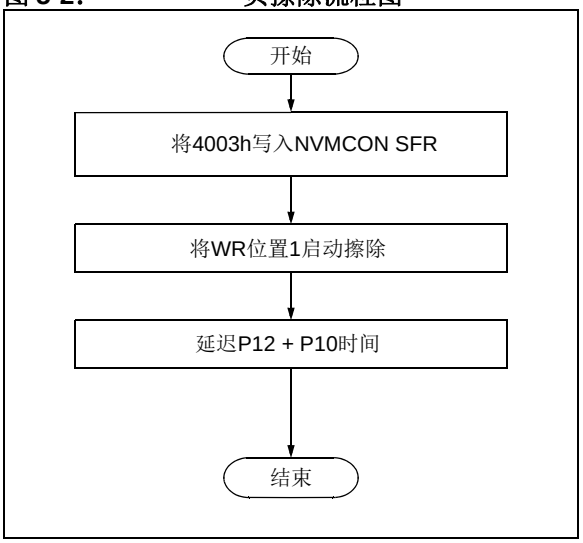


表 5-1：　　擦除执行程序存储区所有页的串行指令执行

命令 （二进制）	数据 （十六进制）	说明
第 1 步：退出复位向量。		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 2 步：设置 NVMADRU/NVMADR 寄存器对以指向要擦除的执行程序存储区的正确页。		
0000	2xxxx3	MOV #DestinationAddress<15:0>, W3
0000	2xxxx4	MOV #DestinationAddress<23:16>, W4
0000	883953	MOV W3, NVMADR
0000	883964	MOV W4, NVMADRU
第 3 步：设置 NVMCON 寄存器以擦除执行程序存储区的第一页。		
0000	24003A	MOV #0x4003, W10
0000	88394A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP
第 4 步：启动擦除周期。		
0000	200551	MOV #0x55, W1
0000	883971	MOV W1, NVMKEY
0000	200AA1	MOV #0xAA, W1
0000	883971	MOV W1, NVMKEY
0000	A8E729	BSET NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 5 步：产生用于完成页擦除操作的时钟脉冲，直到 WR 位清零为止。		
0000	000000	NOP
0000	803940	MOV NVMCON, W0
0000	000000	NOP
0000	887C40	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
—	—	重复执行直到 WR 位清零为止。
第 6 步：重复执行第 2 步到第 5 步，擦除执行程序存储区的所有页。		

5.3 烧写编程执行程序

将 PE 存储到执行程序存储区的过程与对代码存储区的常规编程类似。必须首先擦除执行程序存储区，然后使用双字写操作（两个指令字）进行编程。图 5-3 总结了该方法的控制流程。

表 5-2 给出了 PE 存储区的 ICSP 编程过程。为了使烧写时间最短，将使用与 PE 相同的数据打包格式。关于数据打包格式的更多详细信息，请参见**第 6.2 节“编程执行程序命令”**。

图 5-3： 编程执行程序编程流程图

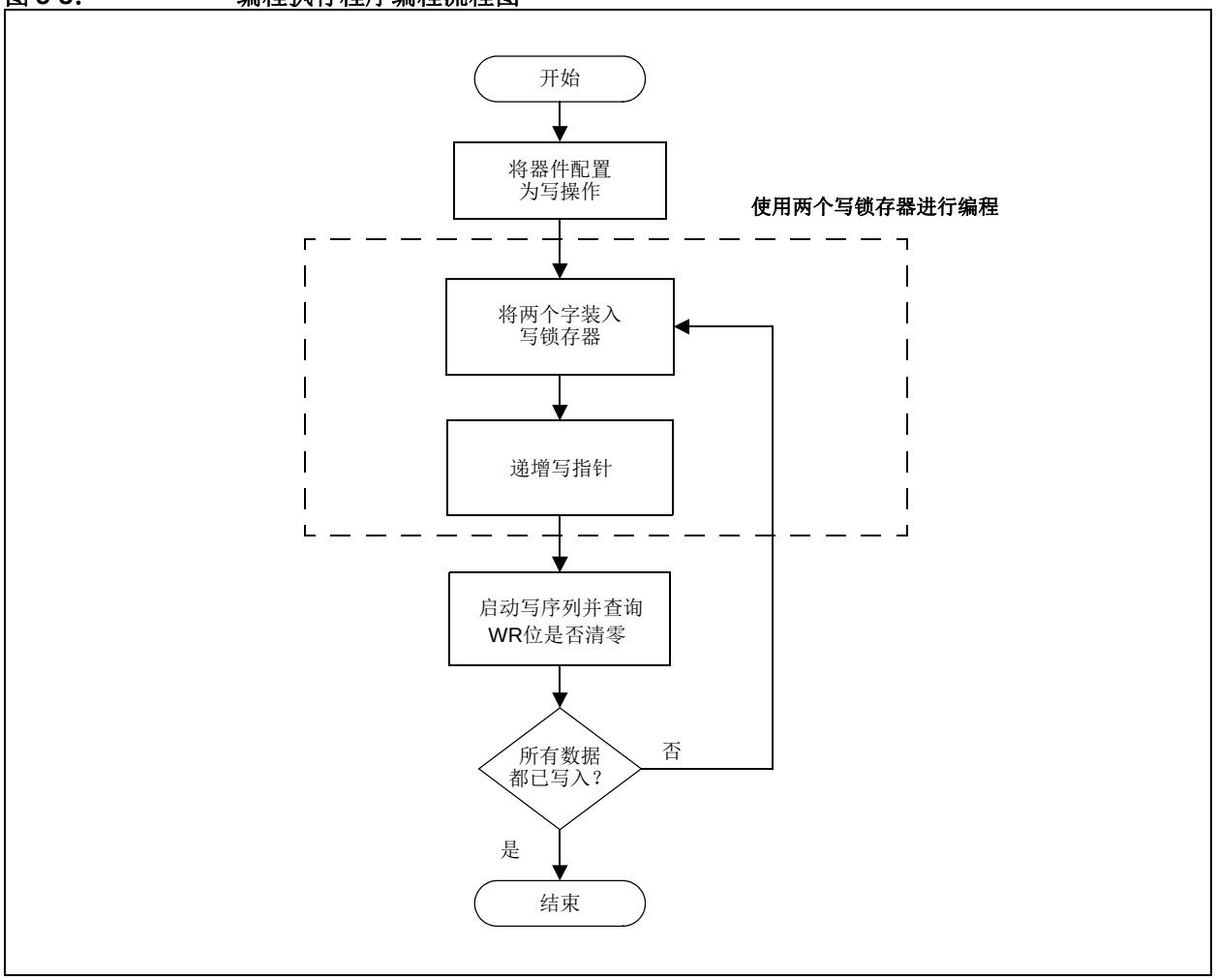


表 5-2：　　烧写编程执行程序（ 双字锁存器写操作）

命令 （二进制）	数据 （十六进制）	说明
第 1 步：退出复位向量。		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 2 步：初始化 TBLPAG 寄存器以写入锁存器。		
0000	200FAC	MOV #0xFA, W12
0000	8802AC	MOV W12, TBLPAG
第 3 步：将下 2 个待编程的打包指令字装入 W0:W2。		
0000	2xxxx0	MOV #<LSW0>, W0
0000	2xxxx1	MOV #<MSB1:MSB0>, W1
0000	2xxxx2	MOV #<LSW1>, W2
第 4 步：设置读指针 （W6）和写指针 （W7）并装载写锁存器。		
0000	EB0300	CLR W6
0000	000000	NOP
0000	EB0380	CLR W7
0000	000000	NOP
0000	BB0BB6	TBLWTL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	BBDBB6	TBLWTH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	BBEBB6	TBLWTH.B [W6++], [++W7]
0000	000000	NOP
0000	000000	NOP
0000	BB0B96	TBLWTL.W [W6], [W7]
0000	000000	NOP
0000	000000	NOP
第 5 步：设置 NVMADRU/NVMADR 寄存器对以指向正确的行。		
0000	2xxxx3	MOV #DestinationAddress<15:0>, W3
0000	2xxxx4	MOV #DestinationAddress<23:16>, W4
0000	883953	MOV W3, NVMADR
0000	883964	MOV W4, NVMADRU
第 6 步：设置 NVMCON 寄存器来烧写 2 个指令字。		
0000	24001A	MOV #0x4001, W10
0000	000000	NOP
0000	88394A	MOV W10, NVMCON
0000	000000	NOP
0000	000000	NOP

表 5-2：　　烧写编程执行程序（ 双字锁存器写操作）（续）

命令 （二进制）	数据 （十六进制）	说明
第 7 步：启动写周期。		
0000	200551	MOV　　#0x55, W1
0000	883971	MOV　　W1, NVMKEY
0000	200AA1	MOV　　#0xAA, W1
0000	883971	MOV　　W1, NVMKEY
0000	A8E729	BSET　　NVMCON, #WR
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 8 步：产生用于完成编程操作的时钟脉冲，直到 WR 位清零为止。		
0000	000000	NOP
0000	803940	MOV　　NVMCON, W0
0000	000000	NOP
0000	887C40	MOV　　W0, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO　　0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
—	—	重复执行直到 WR 位清零为止。
第 9 步：重复执行第 3 步至第 8 步，直到完成所有执行程序存储区的编程。		

5.4 读执行程序存储区

读执行程序存储区是通过执行一系列TBLRD指令和使用 REGOUT 命令随时钟移出数据完成的。

表 5-3 给出了读执行程序存储区的 ICSP 编程的详细步骤。

为了使读时间最短，将使用与 PE 使用的相同数据打包格式。关于数据打包格式的更多详细信息，请参见**第 6.2 节 “编程执行程序命令”**。

表 5-3： 读执行程序存储区的串行指令执行

命令（二进制）	数据（十六进制）	说明
第 1 步：退出复位向量。		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 2 步：为执行 TBLRD 指令初始化 TBLPAG 寄存器和读指针 （W6）。		
0000	200xx0	MOV #<SourceAddress23:16>, W0
0000	8802A0	MOV W0, TBLPAG
0000	2xxxx6	MOV #<SourceAddress15:0>, W6

表 5-3： 读执行程序存储区的串行指令执行（续）

命令（二进制）	数据（十六进制）	说明
第 3 步：初始化写指针 （W7）并将执行程序存储区的下 4 个存储单元内容存储到 W0:W5。		
0000	EB0380	CLR W7
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BA1BB6	TBLRDL [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BA1B96	TBLRDL [W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBB6	TBLRDH.B [W6++], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BADBD6	TBLRDH.B [++W6], [W7++]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	BA0BB6	TBLRDL [W6++], [W7]
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP

表 5-3：　　读执行程序存储区的串行指令执行（续）

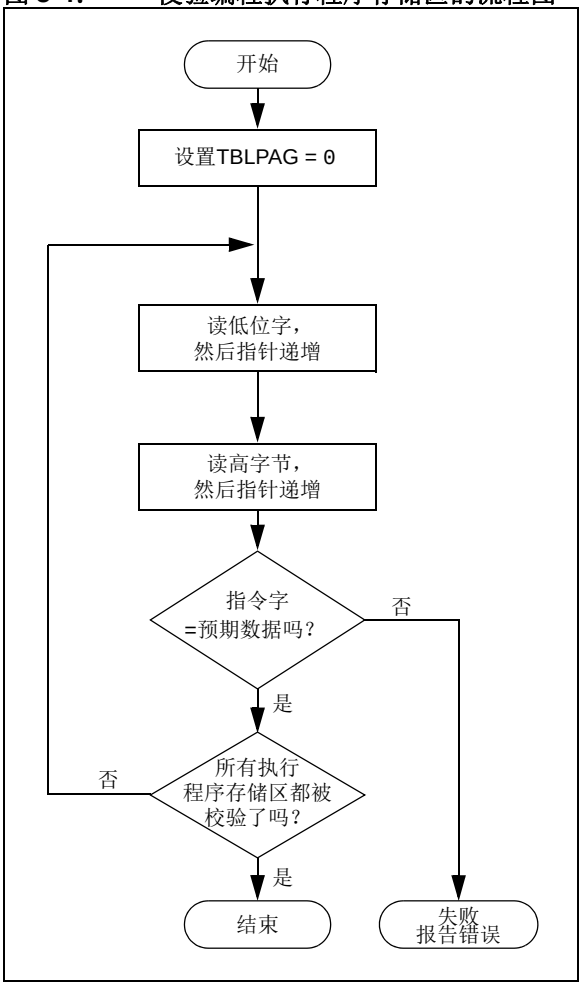
命令（二进制）	数据（十六进制）	说明
第 4 步：使用 VISI 寄存器和 REGOUT 命令输出 W0:W5 的内容。		
0000	887C40	MOV W0, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	887C41	MOV W1, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	887C42	MOV W2, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	887C43	MOV W3, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	887C44	MOV W4, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
0000	887C45	MOV W5, VISI
0000	000000	NOP
0001	<VISI>	随时钟移出 VISI 寄存器的内容。
0000	000000	NOP
第 5 步：复位器件的内部 PC。		
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
0000	040200	GOTO 0x200
0000	000000	NOP
0000	000000	NOP
0000	000000	NOP
第 6 步：重复执行第 3 步到第 5 步，直到读完所有需要的执行程序存储区为止。		

5.5 校验编程执行程序

校验的步骤涉及读回执行程序存储空间并将读到的值与存储在编程器缓冲区中的副本作比较。

校验过程如图 5-4 所述。读取指令的低位字，然后读取高位字的低字节，并将其与编程器缓冲区中存储的指令作比较。关于读取执行程序存储区的实现细节，请参见**第 5.4 节 “读执行程序存储区”**。

图 5-4： 校验编程执行程序存储区的流程图



6.0 编程执行程序

注： 编程执行程序（PE）可从 Microchip 网站（www.microchip.com）上每款器件的页面获取。

6.1 编程执行程序通信

编程器和PE存在主从关系，其中编程器是主编程设备，而 PE 是从设备。

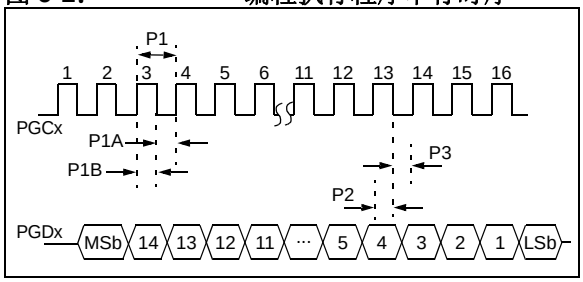
所有通信都是由编程器以命令形式发起的。每次只能将一条命令发送给 PE。而 PE 在接收到命令并对其进行处理后，将只发送一个响应给编程器。**第 6.2 节 “编程执行程序命令”**中介绍了 PE 命令集。**第 6.3 节 “编程执行程序响应”**中介绍了响应集。

6.1.1 通信接口和协议

ICSP/ 增强型 ICSP 接口是使用 PGCx 和 PGDx 引脚实现的双线 SPI 接口。PGCx 引脚用作时钟输入引脚，而时钟源必须由编程器提供。 PGDx 引脚用于向 PE 发送命令数据，以及从其接收响应数据。

注： 对于增强型 ICSP，所有串行数据都在 PGCx 的下降沿发送，在 PGCx 的上升沿锁存。所有数据发送都使用 16 位模式，首先发送最高有效位 （见图 6-1）。

图 6-1： 编程执行程序串行时序



由于使用双线 SPI 接口，且数据发送是双向的，因而可使用一个简单的协议来控制 PGDx 的方向。当编程器完成命令发送时，它会释放 PGDx 线，并允许 PE 将此线驱动为高电平。 PE 将 PGDx 线保持为高电平以指示它正在处理命令。

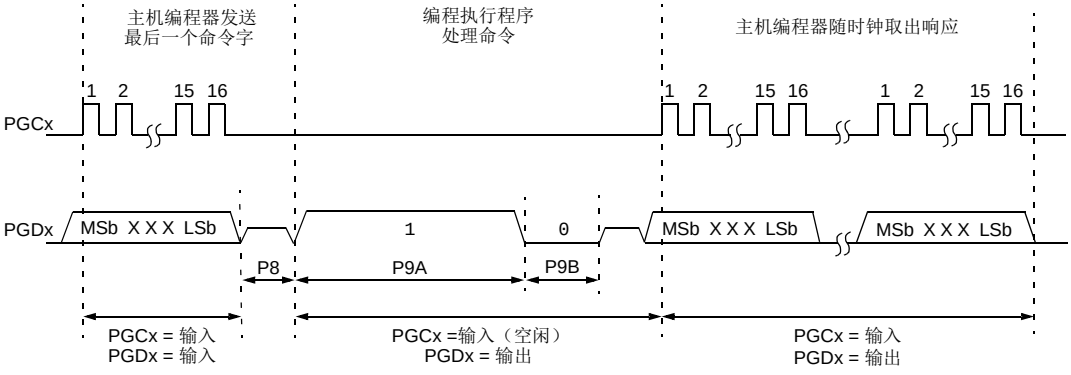
PE 处理完命令之后，会将 PGDx 拉为低电平 （P9B）以通知编程器可以在时钟控制下接收响应。编程器将在最长等待时间 （P9B）后开始接收响应，并且它必须提供必需的时钟脉冲数以从 PE 接收整个响应。

一旦接收到整个响应，编程器应该终止 PGCx 上的时钟直到向PE发送另一条命令为止。该协议如图6-2所示。

6.1.2 SPI速 率

在增强型 ICSP 模式下， dsPIC33EPXXXGS70X/80X 器件使用内部快速 RC（FRC）振荡器作为时钟源工作，其标称频率为 7.3728 MHz。该振荡器频率产生 3.6864 MHz 的有效系统时钟频率。为了确保编程器的时钟速度不至于太快，建议由编程器提供 1.8432 MHz 的时钟。

图6-2： 编程执行程序——编程器通信协议



注： 请参见 表 10-1 了解时序规范。

6.1.3 超时

在向编程器发送响应的过程中， PE 不使用看门狗定时器或超时。如果编程器不遵守**第 6.1.1 节 “通信接口和协议”**中所述的使用 PGCx 的流控制机制，在尝试向编程器发送响应时，PE 可能会表现异常。由于 PE 没有超时，所以编程器必须正确地遵守所述的通信协议。

作为安全措施，编程器应该使用表 6-1 中标识的命令超时。如果命令超时结束，编程器应该将 PE 复位并再次开始对器件编程。

表 6-1： 编程执行程序命令集

操作码	助记符	长度（字节）	超时	说明
0x0	SCHECK	1	1 ms	工作状态检查。
0x1	保留	N/A	N/A	该命令是保留的；它将返回 NACK。
0x2	READP	4	1 ms/ 行	从指定地址开始读取主闪存的 N 个 24 位指令字。
0x3	PROG2W	8	5 ms	对位于代码存储区指定地址处的双指令字进行编程并校验。
0x4	保留	N/A	N/A	该命令是保留的；它将返回 NACK。
0x5	PROGP	195	125 ms	对起始于程序存储器指定地址处的 128 个字进行编程并校验。
0x6	保留	N/A	N/A	该命令是保留的；它将返回 NACK。
0x7	ERASEB	1	125 ms	批量擦除用户存储区。
0x8	保留	N/A	N/A	该命令是保留的；它将返回 NACK。
0x9	ERASEP	3	25 ms	擦除页的命令。
0xA	保留	N/A	N/A	该命令是保留的；它将返回 NACK。
0xB	QVER	1	1 ms	查询 PE 软件版本。
0xC	CRCP	5	1.5s	对指定的存储区范围执行 CRC-16。
0xD	保留	N/A	N/A	该命令是保留的；它将返回 NACK。
0xE	QBLANK	5	1s	查询代码存储区是否为空白。

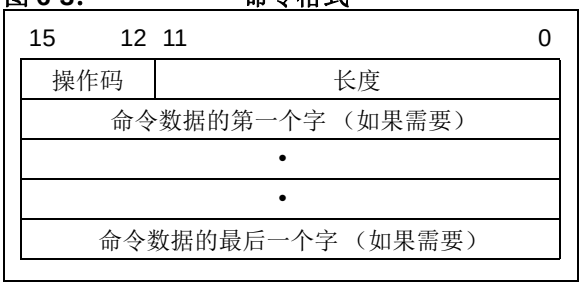
6.2 编程执行程序命令

表 6-1 列出了 PE 命令集。该表包含每条命令的操作码、助记符、长度、超时和说明。在命令说明 （见**第 6.2.4 节 “命令说明”**）中详细说明了每条命令的功能。

6.2.1 命令格式

所有 PE 命令都具有由 16 位头和命令所需的所有数据组成的通用格式（见 图 6-3）。16 位头由用于标识命令的 4 位操作码字段和随后的 12 位命令长度字段组成。

图 6-3： 命令格式



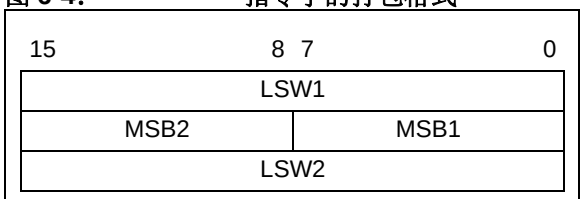
命令操作码必须与命令集中的一条命令匹配。接收到任何与表 6-1 中列出的命令不匹配的命令将返回“NACK”响应 （见**第 6.3.1.1 节 “操作码字段”**）。

由于 SPI 在 16 位模式下工作，所以命令长度以 16 位字表示。 PE 使用命令长度字段来确定要从 SPI 端口读取的字数。如果该字段的值不正确， PE 将无法正确地接收命令。

6.2.2 数据打包格式

当通过 16 位 SPI 接口传输 24 位指令字时，会使用 图 6-4 中所示的格式将这些指令字打包以节省空间。该格式最大限度地降低了通过 SPI 的通信流量，并为 PE 提供了正确对齐以便执行表写操作的数据。

图 6-4： 指令字的打包格式



LSWx：指令字的低 16 位
MSBx：指令字的最高字节

注： 当传输的指令字数为奇数时，MSB2 为零，且不能发送 LSW2。

6.2.3 编程执行程序错误处理

PE 将不应答（NACK）所有不支持的命令。另外，由于 PE 存储空间的限制，所以将不会对包含在编程器命令中的数据进行检查。由编程器负责向 PE 发送带有有效命令参数的命令，否则可能导致编程操作失败。关于错误处理的更多信息，请参见**第 6.3.1.3 节 “QE_Code 字段”**。

6.2.4 命令说明

第 6.2.4.1 节“SCHECK 命令”至**第 6.2.4.10 节“QBLANK 命令”**介绍了 PE 支持的所有命令。

6.2.4.1 SCHECK 命令

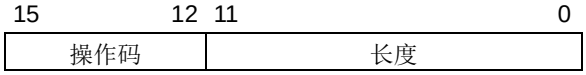


表 6-2 给出了 SCHECK 命令的说明。

表 6-2： 命令说明

字段	说明
操作码	0x0
长度	0x1

SCHECK 命令指示 PE 仅产生响应而不执行其他操作。该命令用作“工作状态检查”以验证 PE 是否正常工作。

预期的响应（双字）：

0x10000

0x0002

注： 该指令不是编程必需的，仅供开发使用。

6.2.4.2 READC 命令

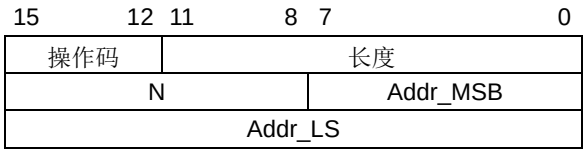


表 6-3 给出了 READC 命令的说明。

表 6-3： 命令说明

字段	说明
操作码	0x1
长度	0x3
N	要读取的 8 位配置寄存器或器件 ID 寄存器的数量（最大为 256）。
Addr_MSB	24 位源地址的 MSB。
Addr_LS	24 位源地址的低 16 位。

READC 命令指示 PE 从由 Addr_MSB 和 Addr_LS 指定的 24 位地址开始读取 N 个配置寄存器或器件 ID 寄存器。该命令只能用于读取 8 位或 16 位数据。

当使用该命令读取配置寄存器时， PE 返回的每个数据字的高字节为 0x00，低字节包含配置寄存器的值。

预期的响应（4 + 3 * (**N – 1**)/2 个字，其中 N 为奇数）：

0x1100

2 + N

配置寄存器或器件 ID 寄存器 1

...

配置寄存器或器件 ID 寄存器 N

注： 读取未实现的存储区将导致 PE 复位。为了防止发生这种情况，请确保只访问在器件中存在的存储单元。

6.2.4.3 READP 命令

15 12	11 8	7 0
操作码	长度	
N		
保留	Addr_MSB	
Addr_LS		

表 6-4 给出了 READP 命令的说明。

表 6-4： 命令说明

字段	说明
操作码	0x2
长度	0x4
N	要读取的 24 位指令数 （最大为 32768）。
保留	0x0
Addr_MSB	24 位源地址的 MSB。
Addr_LS	24 位源地址的低 16 位。

READP 命令指示 PE 从由 Addr_MSB 和 Addr_LS 指定的 24 位地址开始读取代码存储区的 N 个 24 位字。该命令只能用于读取 24 位数据。作为对该命令的响应返回的所有数据使用**第 6.2.2 节 “数据打包格式”**中所述的数据打包格式。

预期的响应 （2 + 3 * N/2 个字，其中 N 为偶数）：

0x1200
2 + 3 * N/2
最低有效程序存储字 1
...
最低有效程序存储字 N

预期的响应（4 + 3 * (**N – 1**)/2 个字，其中 N 为奇数）：

0x1200
4 + 3 * (N – 1)/2
最低有效程序存储字 1
...
程序存储字 N 的 MSB （补零）

注：	读取未实现的存储区将导致PE 复位。为了防止发生这种情况，请确保只访问在器件中存在的存储单元。

6.2.4.4 PROG2W 命令

15 12	11 8	7 0
操作码	长度	
保留	Addr_MSB	
Addr_LS		
DataL_LS		
DataH_MSB	DataL_MSB	
DataH_LS		

表 6-5 给出了 PROG2W 命令的说明。

表 6-5： 命令说明

字段	说明
操作码	0x3
长度	0x6
DataL_MSB	低指令字 24 位数据的 MSB。
DataH_MSB	高指令字 24 位数据的 MSB。
Addr_MSB	24 位目标地址的 MSB。
Addr_LS	24 位目标地址的低 16 位。
DataL_LS	低指令字 24 位数据的低 16 位。
DataH_LS	高指令字 24 位数据的低 16 位。

PROG2W 命令指示 PE 对位于指定存储地址的代码存储区的两个指令字 （6 个字节）进行编程。

当指令字被编程到代码存储区后， PE 将对照命令中的数据对已编程的数据进行校验。

预期的响应 （双字）：

0x1300
0x0002

6.2.4.5 PROGP 命令

15 12	11 8	7 0
操作码	长度	
保留		Addr_MSB
Addr_LS		
D_1		
D_2		
...		
D_N		

表 6-6 给出了 PROGP 命令的说明。

表 6-6： 命令说明

字段	说明
操作码	0x5
长度	0xC3
保留	0x0
Addr_MSB	24 位目标地址的 MSB。
Addr_LS	24 位目标地址的低 16 位。
D_1	16 位数据字 1。
D_2	16 位数据字 2。
...	16 位数据字 3 至 95。
D_96	16 位数据字 96。

PROGP 命令指示 PE 对位于指定存储地址的代码存储区的一行（128 个指令字）进行编程。编程从命令中指定的行地址开始。目标地址应该是 0x80 的倍数。

要编程到存储区中的数据位于命令字 D_1 至 D_96 中，必须使用图 6-4 所示的指令字打包格式对它们进行配置。

在所有数据被编程到代码存储区之后， PE 将对照命令中的数据对已编程的数据进行校验。

预期的响应 （双字）：

0x1500
0x0002

注： 关于代码存储区大小的信息，请参见表2-2。

6.2.4.6 ERASEB 命令

15 12	11 8	7 0
操作码		长度

表 6-7 给出了 ERASEB 命令的说明。

表 6-7： 命令说明

字段	说明
操作码	0x7
长度	0x1

ERASEB 命令指示 PE 对用户闪存执行批量擦除。

预期的响应 （双字）：

0x1700
0x0002

6.2.4.7 ERASEP 命令

15 12	11 8	7 0
操作码	长度	
NUM_PAGES		Addr_MSB
Addr_LS		

表 6-6 给出了 ERASEP 命令的说明。

表 6-8： 命令说明

字段	说明
操作码	0x9
长度	0x3
NUM_PAGES	最多 255 页
Addr_MSB	24 位地址的最高有效字节
Addr_LS	24 位地址的低 16 位

ERASEP 命令指示 PE 对代码存储区执行页擦除 （擦除 [NUM_PAGES] 页）。代码存储区必须按偶数个 512 指令字的地址边界擦除。

预期的响应 （双字）：

0x1900
0x0002

6.2.4.8 QVER 命令

15 12	11 0
操作码	长度

表 6-9 给出了 QVER 命令的说明。

表 6-9： 命令说明

字段	说明
操作码	0xB
长度	0x1

QVER 命令查询存储在测试存储区中的 PE 软件的版本。在响应的 QE_Code 中返回 “版本 . 修订版” 信息，使用一个以下格式的字节：主版本由高半字节表示，修订版本由低半字节表示 （例如， 0x23 表示编程执行程序软件的版本为 2.3）。

预期的响应 （双字）：

0x1BMN （其中 “MN” 代表版本 M.N）
0x0002

6.2.4.9 CRCP 命令

15 12	11 8	7 0
操作码	长度	
保留		Addr_MSB
Addr_LSW		
保留		Size_MSB
Size_LSW		

表 6-11 给出了 CRCP 命令的说明。

表 6-10： 命令说明

字段	说明
操作码	Ch
长度	5h
Addr_MSB	24 位地址的最高有效字节
Addr_LSW	24 位地址的低 16 位
大小	24 位存储单元的数量 （地址范围除以 2）

CRCP命令对指定的存储区范围执行CRC-16。该命令可以代替整片校验。数据以图6-4中所示的打包方法按字节进行移位，先移位最低有效字节（Least SignificantByte，LSB）。

示例：

测试数据“123456789”的 CRC-CCITT-16 将为 29B1h

预期的响应 （3 字）：

QE_Code：0x1C00
长度：0x0003
CRC 值：0xXXXX

6.2.4.10 QBLANK 命令

15 12	11 0
操作码	长度
保留	Size_MSB
Size_LSW	
保留	Addr_MSB
Addr_LSW	

表 6-11 给出了 QBLANK 命令的说明。

表 6-11： 命令说明

字段	说明
操作码	0xE
长度	0x5
大小	要检查的程序存储区的长度 （用 24 位字数表示） + Addr_MS
Addr_MSB	24 位地址的最高有效字节
Addr_LSW	24 位地址的低 16 位

QBLANK 命令查询 PE 以确定代码存储区的内容是否为空白 （包含全 1）。必须在命令中指定要检查的代码存储区的大小。

代码存储区的空白检查始于 [Addr]，并朝着地址更大的方向检查指定数目的指令字。

如果指定的代码存储区为空白，QBLANK 将返回 QE_Code F0h ； 否则， QBLANK 将返回 QE_Code 0Fh。

预期的响应 （空白器件的双字）：

0x1EF0
0x0002

预期的响应 （非空白器件的双字）：

0x1E0F
0x0002

注： QBLANK 命令不会对系统操作配置位进行检查，因为这些位在执行芯片擦除时不会置为 1。

6.3 编程执行程序响应

每接收到一条命令，PE 就会向编程器发送一个响应。该响应指示命令是否已被正确处理。其中包含任何必需的响应数据或错误数据。

表 6-12 列出了 PE 响应集。该表包含每个响应的操作码、助记符和说明。**第 6.3.1 节 “响应格式”**中介绍了响应格式。

表 6-12： 编程执行程序响应集

操作码	助记符	说明
0x1	PASS	命令已成功处理。
0x2	FAIL	命令未成功处理。
0x3	NACK	命令未知。

6.3.1 响应格式

所有 PE 响应都具有由双字响应头和该命令所需的所有数据组成的通用格式。

15 12	11 8	7 0
操作码	Last_Cmd	QE_Code
长度		
D_1 （如果适用）		
...		
D_N （如果适用）		

表 6-13 给出了响应格式的说明。

表 6-13： 响应格式说明

字段	说明
操作码	响应操作码。
Last_Cmd	产生该响应的编程器命令。
QE_Code	查询代码或错误代码。
长度	以 16 位字数表示的响应长度 （包含双字响应头）。
D_1	第一个 16 位数据字 （如果适用）。
D_N	最后一个 16 位数据字 （如果适用）。

6.3.1.1 操作码字段

操作码是响应的第一个字中的一个 4 位字段。操作码指示命令的处理情况 （见表 6-12） 。如果命令已成功处理，响应操作码是 PASS。如果在处理命令过程中发生错误，则响应操作码是 FAIL， QE_Code 将指示失败的原因。如果发送给 PE 的命令未被识别，则 PE 将返回 NACK 响应。

6.3.1.2 Last_Cmd 字段

Last_Cmd 是响应的第一个字中的一个 4 位字段，该字段指示 PE 处理的命令。由于 PE 每次只能处理一条命令，所以从技术角度而言不需要此字段。但是，它可以用来验证 PE 是否正确地接收了编程器发送的命令。

6.3.1.3 QE_Code 字段

QE_Code 是响应的第一个字中的一个字节。该字节用于为查询命令返回数据，为所有其他命令返回错误代码。

当 PE 处理两种查询命令（QBLANK 或 QVER）之一时，返回的操作码总是 PASS，QE_Code 将保存查询响应数据。表 6-14 给出了两种查询的 QE_Code 的格式。

表 6-14： 针对查询命令的 QE_Code

查询	QE_Code
QBLANK	0x0F = 代码存储区非空白 0xF0 = 代码存储区空白
QVER	0xMN，其中 PE 软件版本 = M.N （例如， 0x32 表示软件版本为 3.2）。

当 PE 处理除查询外的任何命令时，QE_Code 代表一个错误代码。表 6-15 给出了支持的错误代码。如果命令已成功处理，返回的 QE_Code 将设置为 0x0，指示在处理命令过程中没有发生错误。如果对 PROGW 命令的编程进行校验时失败，则 QE_Code 将被设置为 0x1。对于所有其他 PE 错误， QE_Code 为 0x02。

表 6-15： 针对非查询命令的 QE_Code

QE_Code	说明
0x0	无错误
0x1	校验失败
0x2	其他错误

6.3.1.4 响应长度

响应长度指示 PE 的响应的长度，以 16 位字数表示。该字段包含双字响应头。

除了对读命令的响应外，对其他命令的响应的长度都只有两个字。

对 READP 命令的响应使用**第 6.2.2 节“数据打包格式”**中所述的指令字打包格式。当读取奇数个程序存储字（N 为奇数）时，对 READP 命令的响应是（3 * (N + 1)/2 + 2）个字。当读取偶数个程序存储字 （N 为偶数）时，对 READP 命令的响应是 （3 *N/2 + 2）个字。

7.0 双分区闪存编程注意事项

dsPIC33EPXXXGS70X/80X 系列器件支持一种单分区闪存模式和两种双分区闪存模式。双分区闪存模式允许通过两个单独的应用程序对器件进行编程，便于引导加载，或允许应用程序在运行时进行编程，而无需暂停 CPU。

器件的引导模式由 FBOOT 配置寄存器中的 BTMODE<1:0> 位决定（见表 7-1）。器件会在复位时自动检查 FBOOT 并确定相应的引导模式。

表 7-1： 引导模式选择

FBOOT<1:0>	引导模式
00	保留
01	受保护双分区闪存模式
10	双分区闪存模式
11	单分区闪存模式 （默认设置）

受保护双分区闪存模式防止对分区 1 执行运行时编程和擦除； ICSP 模式不受影响。

7.1 双分区闪存程序存储器构成

在双分区闪存模式下，器件的存储器均分为两个物理部分，称为分区 1 和分区 2。每个分区都包含自己的程序存储器和配置字。在程序执行期间，只可执行其中一个分区中的代码，该分区为活动分区。另一个非活动分区不可用于执行但可进行编程。

活动分区始终映射到程序存储器地址 000000h，而非活动分区将始终映射到程序存储器地址 400000h。即使用户将代码分区在活动和非活动之间切换，活动分区的地址将仍为 000000h，非活动分区的地址仍为 400000h。

引导序列配置字（FBTSEQ）确定在器件复位后是分区 1 还是分区 2 活动。如果器件工作在双分区闪存模式下，具有较低引导序列号的分区将作为活动分区（FBTSEQ 在单分区模式下不使用）运行。可通过重新编程分区的引导序列号将分区在活动和非活动之间切换，但在执行器件复位前活动分区将不会发生改变。如果两个分区的引导序列号相同或者均损坏，器件将使用分区 1 作为活动分区。如果只有一个引导序列号损坏，器件将使用引导序列号未损坏的分区作为活动分区。

用户还可在运行时使用 BOOTSWP 指令更改活动的分区。BOOTSWP 指令必须先使能才能使用 （位于 FICD 配置字中）。发出 BOOTSWP 指令不会影响在复位后哪个分区将是活动分区。图 7-1 显示了分区 1 和 2 分别在 FBTSEQ 重新编程和 BOOTSWP 执行期间如何在活动和非活动分区之间切换。

P2ACTIV 位（NVMCON<10>）可用于确定哪个物理分区是活动分区。如果 P2ACTIV = 1，分区 2 是活动分区；如果 P2ACTIV = 0，分区 1 是活动分区。

7.2 对双分区闪存的擦除操作

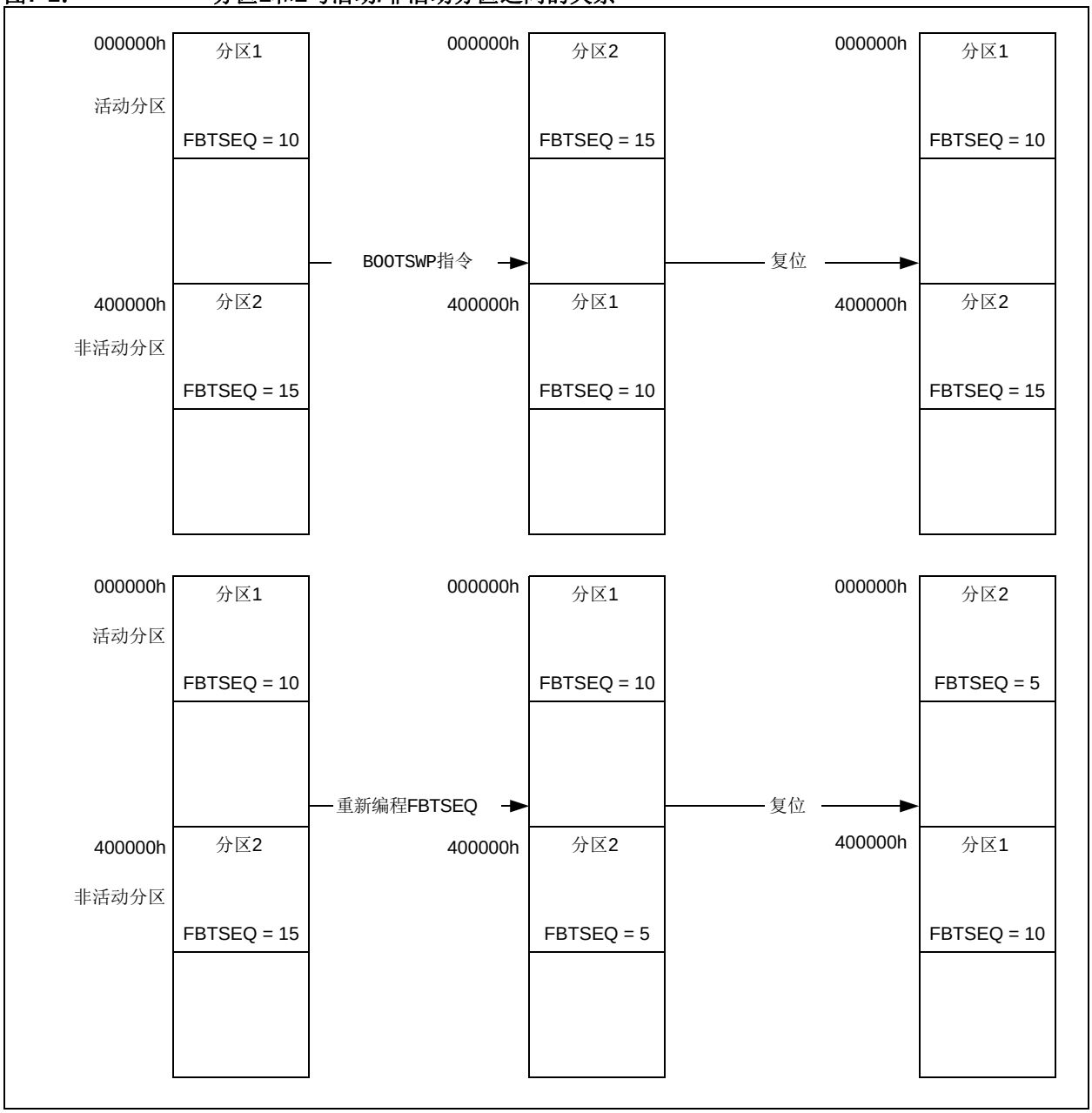
dsPIC33EPXXXGS70X/80X 系列器件支持以下三种擦除操作：批量擦除、非活动分区擦除和页擦除。

批量擦除操作将擦除所有用户存储区，包括闪存配置字和 FBOOT 配置寄存器。这将在器件复位后将器件引导模式恢复为默认的单分区模式。

非活动分区擦除操作可以在运行时从活动分区执行。它将擦除非活动分区中的所有用户存储区和闪存配置字。非活动分区擦除命令仅在器件处于双分区闪存模式之一时有效。

闪存配置字位于每个分区的最后几个存储单元中。可使用批量擦除、非活动分区擦除或页擦除（针对分区的最后一页）来擦除闪存配置字。

图7-1：　　分区1和2与活动/非活动分区之间的关系



7.3 双分区配置字

在双分区模式下，每个分区都有自己的一组闪存配置字。活动分区中的整组配置寄存器用于确定器件的配置。在编程FBTSEQ允许非活动分区成为活动分区并进行复位之前，不使用非活动分区中的配置字。B00TSWP指令不会根据新活动分区的配置字更改器件的有效配置。但是，非活动分区中的一些配置寄存器（FSEC、FBSLIM 和 FSIGN）可用于确定使能活动分区或允许活动分区访问非活动分区的方式。

批量擦除用户存储区将自动编程分区 1 的 FSIGN 寄存器中的保留位（FSIGN<15>）。因此，如果用户将 FBOOT 重新编程为使用双分区闪存模式，则需要手动编程分区 2 的 FSIGN 寄存器中的保留位。

7.4 在双分区闪存模式下编程

以下是使用双分区闪存编程 dsPIC33EPXXXGS70X/80X 器件所需的步骤：

1. 对用户程序存储区执行批量擦除。
2. 重新进入 ICSP 模式，使程序存储器地址映射以已知状态开始。
3. 将 FBOOT 寄存器编程为双分区闪存模式之一。
4. 在地址 0x005400 处执行页擦除。

注： 现在不要复位器件，因为在此阶段复位将需要从步骤 1 重复执行编程序列。

5. 编程地址 0x00AB94 = 0xFF7FFF（相邻字保留为 0xFFFFFF）。
6. 发出器件复位。这将存储区拆分为两个分区。
7. 将第一个应用程序（包括其引导序列号和配置字）编程到位于地址 000000h 处的活动分区。
8. 将第二个应用程序、其引导序列号和配置字编程到位于地址 400000h 处的非活动分区。

8.0 器件 ID/唯一 ID

器件ID存储区可用于确定芯片的型号和生产信息。该存储区只读，并且能在使能代码保护时读取。

表 8-1 列出了每款器件的标识信息。表 8-2 给出了器件 ID 寄存器。

表 8-1： 器件 ID

器件	DEVID	DEVREV	硅片版本
dsPIC33EP64GS708	0x6C03	0x6C00	A0
dsPIC33EP64GS804	0x6C40		
dsPIC33EP64GS805	0x6C60		
dsPIC33EP64GS806	0x6C42		
dsPIC33EP64GS808	0x6C43		
dsPIC33EP128GS702	0x6C11		
dsPIC33EP128GS704	0x6C10		
dsPIC33EP128GS705	0x6C30		
dsPIC33EP128GS706	0x6C12		
dsPIC33EP128GS708	0x6C13		
dsPIC33EP128GS804	0x6C50		
dsPIC33EP128GS805	0x6C70		
dsPIC33EP128GS806	0x6C52		
dsPIC33EP128GS808	0x6C53		

表8-2： 器件ID寄存器

地址	名称	Bit															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FF0000h	DEVID	DEVID 值															
FF0002h	DEVREV	DEVREV 值															

寄存器8-1： JTAG ID寄存器

31 28	27 12	11 0
DEVREV<3:0>	DEVID<15:0>	制造商 ID （053h）
4 位	16 位	12 位

8.1 器件唯一ID（UDID）

在最终制造过程中，会为所有dsPIC33EPXXXGS70X/80X系列器件单独地编码一个器件唯一标识符（Unique Device Identifier，UDID）。批量擦除命令或其他任何用户可使用的方法都无法擦除UDID。在需要对MicrochipTechnology器件进行制造追踪的应用中，可以利用该功能来实现该目的。应用制造商也可以将它用于可能需要唯一标识的任意应用场合，例如：

- 追踪器件
- 唯一序列号
- 唯一安全密钥

UDID 包含 5 个 24 位程序字。这些字组合在一起构成一个唯一的 120 位标识符。 UDID 存储在器件配置空间中的5个只读单元中。表8-3列出了器件标识符字的地址，并给出了其内容。

表8-3： UDID地址

UDID	地址	内容
UDID1	800F00	UDID 字 1
UDID2	800F02	UDID 字 2
UDID3	800F04	UDID 字 3
UDID4	800F06	UDID 字 4
UDID5	800F08	UDID 字 5

9.0 校验和计算

器件校验和大小为 16 位。通过对以下内容求和可以计算校验和：

- 代码存储单元的内容
- 配置字的内容 （除了 FBTSEQ 配置寄存器外）

如果配置位默认为 0 （例如， JTAGEN），则擦除后的状态仍为 1。该位通过器件编程器设置为 0 并在计算校验和时被掩码。因此，术语“空白校验和”在由MPLAB® X IDE 或其他编程器使用时必须考虑被掩码的配置位。

通过将每个存储器地址的所有三个字节相加来对所有存储单元（包括配置字）求和。对于 FSIGN 和 FICD，使用读掩码来忽略保留的位。

计算配置存储器部分的校验和时使用以下掩码：

- FSIGN：FF7FFFh，掩码 bit 15
- FICD：FFFFDFh，掩码 JTAGEN 位 （FICD<5>）

校验和计算跨越程序存储器和配置存储器的整个区域。配置字可能占用总配置存储器的一小部分，但整个地址空间用于校验和计算。

表 9-1： 校验和计算示例（ 单分区闪存）

器件	读代码保护	校验和计算	空白值	0x0 和最后一个代码地址处为 0xAAAAAA 时的值
dsPIC33EP64GS70X	禁止	PROG[0:00AF7Eh] + CFGB[00AF80h:00AFFEh]	0xF463	0xF265
	使能	0x00	0x0000	0x0000
dsPIC33EP128GS70X	禁止	PROG[0:01577Eh] + CFGB[015780h:0157FEh]	0xF863	0xF665
	使能	0x00	0x0000	0x0000

图注： PROG[a:b] = a 至 b （含 a 和 b）的所有程序存储单元的字节和 （代码存储区的全部 3 个字节）
CFGB[c:d] = c 至 d（含 c 和 d）的配置存储单元的字节和（FSIGN 和 FICD 被掩码，而 FBTSEQ 被忽略）

表 9-2： 校验和计算示例（ 双分区闪存）

器件	读代码保护	校验和计算	空白值	0x0 和最后一个代码地址处为 0xAAAAAA 时的值
dsPIC33EP64GS70X	禁止	PROG[0:00577Eh] + CFGB[005780h:0057FEh] + PROG[40000h:40577Eh] + CFGB[40AB80h:40ABFEh]	0xF0C6	0xECCA
	使能	0x00	0x0000	0x0000
dsPIC33EP128GS70X	禁止	PROG[0:00AB7Eh] + CFGB[00AB80h:00ABFEh] + PROG[40000h:40AB7Eh] + CFGB[40AB80h:40ABFEh]	0xF4C6	0xF0CA
	使能	0x00	0x0000	0x0000

图注： PROG[a:b] = a 至 b （含 a 和 b）的所有程序存储单元的字节和 （代码存储区的全部 3 个字节）
CFGB[c:d] = c 至 d（含 c 和 d）的配置存储单元的字节和（FSIGN 和 FICD 被掩码，而 FBTSEQ 被忽略）

10.0 交流/直流特性和时序要求

表 10-1 列出了交流 / 直流特性和时序要求。

表10-1： 交流/直流特性和时序要求

标准工作条件 工作温度：-40°C 至 +85°C。建议在 +25°C 下编程。						
参数编号	**符号**	**特性**	**最小值**	**最大值**	**单位**	**条件**
D111	VDD	编程时的电源电压	3.0	3.6	V	请参见**注 1**
P1	TPGC	串行时钟 （PGCx）周期 （ICSP™）	200	—	ns	
P1	TPGC	串行时钟 （PGCx）周期 （增强型 ICSP）	500	—	ns	
P1A	TPGCL	串行时钟 （PGCx）的低电平时间 （ICSP）	80	—	ns	
P1A	TPGCL	串行时钟 （PGCx）的低电平时间 （增强型 ICSP）	200	—	ns	
P1B	TPGCH	串行时钟 （PGCx）的高电平时间 （ICSP）	80	—	ns	
P1B	TPGCH	串行时钟 （PGCx）的高电平时间 （增强型 ICSP）	200	—	ns	
P2	TSET1	输入数据到串行时钟↓的建立时间	15	—	ns	
P3	THLD1	在 PGCx↓后输入数据的保持时间	15	—	ns	
P4	TDLY1	4 位命令和命令操作数之间的延时	40	—	ns	
P4A	TDLY1A	命令操作数和下一条 4 位命令之间的延时	40	—	ns	
P5	TDLY2	命令的最后一个 PGCx↓到读取数据字时出现的第一个 PGCx↑之间的延时	20	—	ns	
P6	TSET2	VDD↑到 $\overline{\text{MCLR}}$↑的建立时间	100	—	ns	
P7	THLD2	在 $\overline{\text{MCLR}}$↑后输入数据的保持时间	50	—	ms	
P8	TDLY3	命令字节的最后一个 PGCx↓到 PE 将 PGDx 驱动为↑之间的延时	12	—	µs	
P9A	TDLY4	PE 命令处理时间	10	—	µs	
P9B	TDLY5	PE 将 PGDx 驱动为↓到 PE 释放 PGDx 之间的延时	15	23	µs	
P10	TDLY6	编程后 PGCx 处于低电平的时间	400	—	ns	
P11	TDLY7	批量擦除时间	16	24	ms	
P12	TDLY8	页擦除时间	—	—	ms	请参见**注 2**
P13A	TDLY9A	行编程时间	—	—	µs	请参见**注 2** 和**注 3**
P13B	TDLY9B	双字编程时间	—	—	µs	请参见**注 2** 和**注 3**
P14	TR	进入 ICSP 模式的 $\overline{\text{MCLR}}$ 上升时间	—	1.0	µs	
P15	TVALID	PGCx↑后的数据输出有效时间	10	—	ns	
P16	TDLY10	最后一个 PGCx↓和 $\overline{\text{MCLR}}$↓之间的延时	0	—	s	

注 1： 在编程期间还必须将 VDD 施加到 AVDD 引脚。 AVDD 和 AVSS 应该总是分别在 VDD±0.3V 和 VSS±0.3V 以内。

2： 时间取决于 FRC 精度和 FRC 振荡器调节寄存器的值。请参见具体器件数据手册的 **“电气特性”**章节。

3： 该时间适用于程序存储字、配置字和用户 OTP 字。

表10-1： 交流/直流特性和时序要求（续）

标准工作条件 工作温度：-40°C 至 +85°C。建议在 +25°C 下编程。						
参数编号	**符号**	**特性**	**最小值**	**最大值**	**单位**	**条件**
P17	THLD3	$\overline{MCLR}$ ↓ 到 VDD ↓ 的时间	100	—	ns	
P18	TKEY1	从第一个 $\overline{MCLR}$ ↓ 到向 PGDx 输入密钥序列的第一个 PGCx ↑ 之间的延时	1	—	ms	
P19	TKEY2	从向 PGDx 输入密钥序列的最后一个 PGCx ↓ 到第二个 $\overline{MCLR}$ ↑ 之间的延时	25	—	ns	
P21	TMCLRH	$\overline{MCLR}$ 高电平时间	—	500	µs	

注 1： 在编程期间还必须将 VDD 施加到 AVDD 引脚。 AVDD 和 AVSS 应该总是分别在 VDD±0.3V 和 VSS±0.3V 以内。

2： 时间取决于 FRC 精度和 FRC 振荡器调节寄存器的值。请参见具体器件数据手册的 **“电气特性”** 章节。

3： 该时间适用于程序存储字、配置字和用户 OTP 字。

附录 A： 版本历史

版本 A （2016 年 4 月）

为 dsPIC33EPXXXGS70X/80X 器件系列编写的编程规范的初始版本。

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：978-1-5224-1850-4

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

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 - 广州
Tel: 86-20-8755-8029

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

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: 86-21-3326-8000
Fax: 86-21-3326-8021

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

亚太地区

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

台湾地区 - 高雄
Tel: 886-7-213-7830

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

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

法国 France - Saint Cloud
Tel: 33-1-30-60-70-00

德国 Germany - Garching
Tel: 49-8931-9700

德国 Germany - Haan
Tel: 49-2129-3766400

德国 Germany - Heilbronn
Tel: 49-7131-67-3636

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: 47-7289-7561

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

超前信息