PIC32 闪存编程规范

1.0 器件概述

本文档定义了PIC32系列32位单片机的闪存编程规范。

本编程规范旨在指导开发人员使用外部编程器工具。对于正在开发PIC32器件应用程序的客户应使用已提供器件编程支持的开发工具。

主要讨论的主题包括：

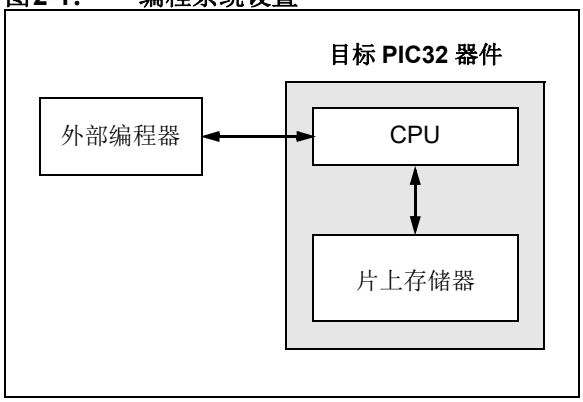
- **第1.0节“器件概述”**
- **第2.0节“编程概述”**
- **第3.0节“编程步骤”**
- **第4.0节“连接至器件”**
- **第5.0节“EJTAG与ICSP”**
- **第6.0节“伪操作”**
- **第7.0节“进入2线增强型ICSP模式”**
- **第8.0节“检查器件状态”**
- **第9.0节“擦除器件”**
- **第10.0节“进入串行执行模式”**
- **第11.0节“下载编程执行程序（PE）”**
- **第12.0节“下载数据块”**
- **第13.0节“启动页擦除”**
- **第14.0节“启动闪存行写入”**
- **第15.0节“校验器件存储器”**
- **第16.0节“退出编程模式”**
- **第17.0节“编程执行程序”**
- **第18.0节“校验和”**
- **第19.0节“配置存储器和器件ID”**
- **第20.0节“TAP控制器”**
- **第21.0节“交流/直流特性和时序要求”**
- **附录A：“PIC32闪存映射”**
- **附录B：“Hex文件格式”**
- **附录C：“器件ID”**
- **附录D：“版本历史”**

2.0 编程概述

在开发编程工具时，需要了解目标器件的内部闪存编程操作以及用于控制闪存编程的特殊功能寄存器（Special Function Register，SFR），因为这些操作和寄存器将由外部编程工具及其软件使用。这些操作和控制寄存器在具体器件数据手册的**“闪存程序存储器”**章节和《PIC32系列参考手册》的相关章节中进行了说明。强烈建议将这些文档与本编程规范结合使用。

外部工具编程设置由外部编程器工具和目标PIC32器件组成。图2-1给出了一个典型的编程设置。编程器工具负责执行所需的编程步骤并完成编程操作。

图2-1： 编程系统设置



2.1 具有双闪存分区和双引导区域的器件

PIC32MKXXXXXXD/E/F/K/L/M和PIC32MZ系列器件集成了多个用于器件现场（自）编程的实用功能。这些功能包括具有双引导区域的双闪存分区、用于引导区域的别名机制（允许在启动时自动选择引导代码）和用于程序闪存的分区交换功能。两个闪存分区及其相关的引导区域均可分别擦除和编程。关于这些功能的详细说明，请参见请参见《PIC32系列参考手册》的**第48章“存储器构成和权限”**（DS60001214A_CN）。

用于生产编程的开发工具将不用考虑大部分特性，以下除外：

- 确保SWAP位（NVMCON<7>）设置正确。默认设置为0，无分区交换。在生成用于编程工具的源文件时，开发工具应假设为默认设置。
- 正确处理校验和计算中的引导存储器别名。别名部分将与固定部分一致。关于别名区域校验和计算的更多信息，请参见**第18.0节“校验和”**。
- 对于PIC32MK器件，尝试擦除闪存页面失败并需要重试时，请使用擦除/重试功能。更多信息，请参见**第13.0节“启动页擦除”**。

2.2 编程接口

所有PIC32器件都为外部编程器工具提供了两种物理接口：

- 2线在线串行编程（In-Circuit Serial Programming™，ICSP™）
- 4线联合测试工作组（Joint Test Action Group，JTAG）

更多信息，请参见**第4.0节“连接至器件”**。

这两种方法都可以使用可下载的编程执行程序（Programming Executive，PE）。PE可从目标器件RAM中执行并对编程器隐藏器件编程详细信息。它还可以免除与数据传输关联的开销并提高总数据吞吐量。Microchip开发了可用于任意外部编程器的PE（更多信息，请参见**第17.0节“编程执行程序”**）。

第3.0节“编程步骤”介绍了一些高级编程步骤，并随后给出了每个步骤的简要说明。有关详细说明，请参见本文档的相应章节。

有关编程命令、EJTAG和直流规范的更多信息，请参见以下章节：

- **第19.0节“配置存储器和器件ID”**
- **第20.0节“TAP控制器”**
- **第21.0节“交流/直流特性和时序要求”**

2.3 增强型JTAG（EJTAG）

2线和4线接口使用EJTAG协议来与编程器交换数据。虽然本文档根据需要提供了该协议的工作描述，但还是建议高级用户访问Imagination Technologies Limited网站（www.imgtec.com）以获取更多信息。

2.4 数据大小

数据大小定义如下：

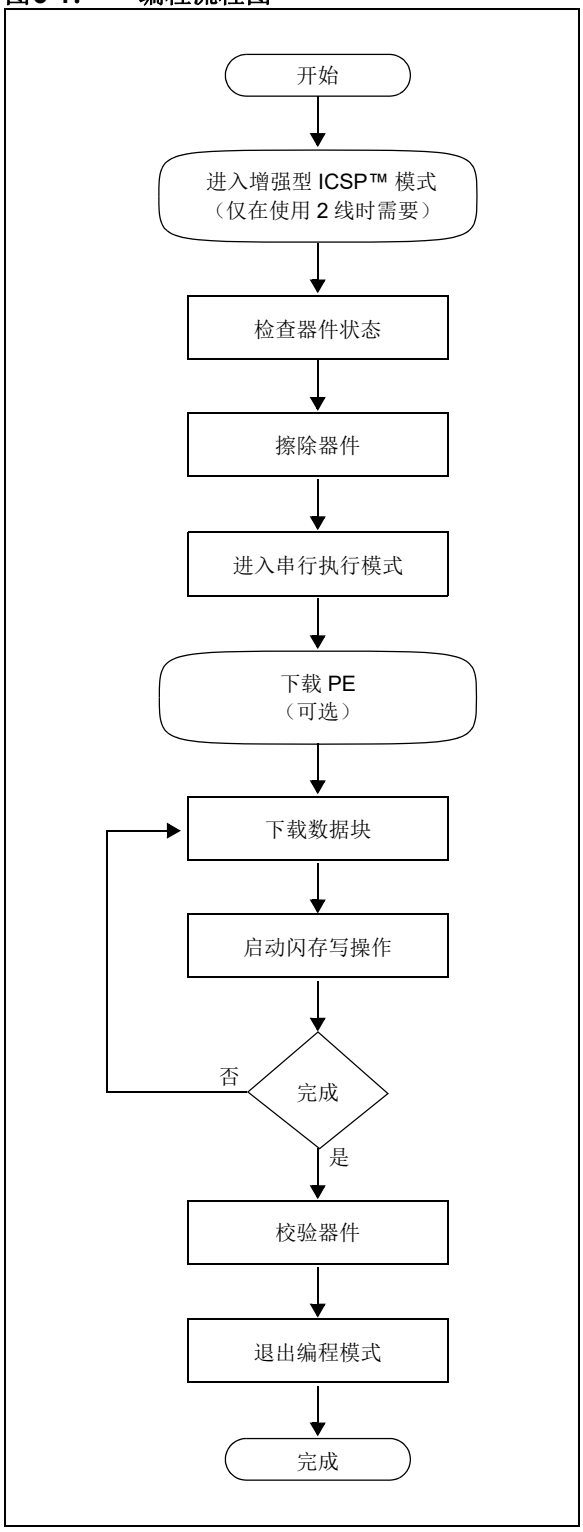
- 一个字：32位
- 半字：16位
- 四分之一字：8位
- 一个字节：8位

© 2014-2020 Microchip Technology Inc.

3.0 编程步骤

所有工具编程器都必须执行一系列常用步骤，与实际使用的方法无关。图3-1显示了用于PIC32器件编程的一系列步骤。

图3-1： 编程流程图



下面按顺序列出了编程步骤，并简要介绍了每个步骤。有关各个步骤的更多详细信息，请参见后续章节。

1. 连接到目标器件。

   为确保成功编程，必须将所有必要的引脚都连接到对应的信号。更多信息，请参见**第4.0节“连接至器件”**。

2. 将目标器件置于编程模式。

   对于2线编程方法，必须在执行任何其他步骤之前将目标器件置于特殊的编程模式（增强型ICSP™）下。

   注： 对于4线编程方法，第2步不适用。

   更多信息，请参见**第7.0节“进入2线增强型ICSP模式”**。

3. 检查器件的状态。

   检查器件的状态，确保它准备好接收来自编程器的信息。

   更多信息，请参见**第8.0节“检查器件状态”**。

4. 擦除目标器件。

   如果器件中的目标存储块为非空白，或器件处于代码保护状态，则必须在对任何新数据进行编程之前执行擦除步骤。

   更多信息，请参见**第9.0节“擦除器件”**。

5. 进入编程模式。

   验证器件是否受代码保护，并引导TAP控制器以开始向/从PIC32 CPU发送/接收数据。

   更多信息，请参见**第10.0节“进入串行执行模式”**。

6. 下载编程执行程序（PE）。

   PE是下载到目标器件RAM中的一小段可执行代码。它将接收实际数据并对其进行编程。

注：	如果使用的编程方法不需要PE，则第6步不适用。

   更多信息，请参见**第11.0节“下载编程执行程序（PE）”**。

7. 下载要编程的数据块。

   不管是否需要PE，所有方法都必须将所需编程数据下载到RAM中的存储块中。

   更多信息，请参见**第12.0节“下载数据块”**。

8. 启动闪存写入。

   将每个数据块下载到RAM中后，必须启动编程序列，以将其编程到目标器件的闪存中。

   更多信息，请参见**第14.0节“启动闪存行写入”**。

9. 重复第 7 步和第 8 步，直到所有数据块都已下载并编程。

10. 校验程序存储器。

    对所有编程数据和配置位进行编程之后，应回读目标器件存储器，验证内容是否匹配。

    更多信息，请参见**第15.0节“校验器件存储器”**。

11. 退出编程模式。

    只有对目标器件断电并重新上电，或者执行退出编程序列之后，新编程的数据才会生效。

    更多信息，请参见**第16.0节“退出编程模式”**。

© 2014-2020 Microchip Technology Inc.

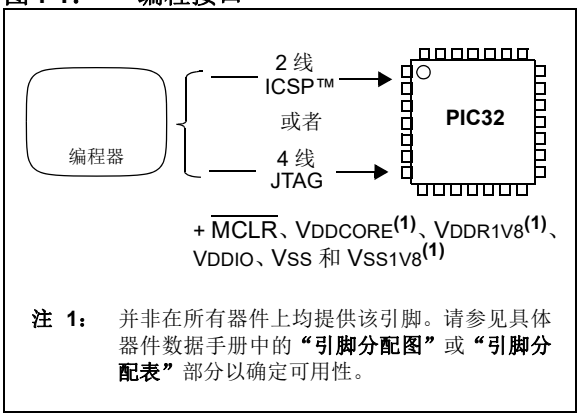
4.0 连接至器件

PIC32系列提供了两种可供选择的物理接口，用于进行连接和存储器内容编程，请参见图4-1。对于所有编程接口，必须为目标器件供电，并且必须连接全部所需信号。此外，还必须使能接口，如4线JTAG接口通过其配置位使能，2线ICSP接口则通过特殊初始化序列使能。

在默认情况下，出厂的空白器件会使能JTAG接口。

有关如何使能ICSP，请参见**第7.0节“进入2线增强型ICSP模式”**。

图4-1： 编程接口



注 1： 并非在所有器件上均提供该引脚。请参见具体器件数据手册中的**“引脚分配图”**或**“引脚分配表”**部分以确定可用性。

4.1 4线接口

其中一个可用的接口是4线JTAG（IEEE 1149.1）端口。表4-1列出了所需引脚连接。该接口使用以下四条通信线路来与要编程的PIC32器件交换数据：

- 测试时钟输入（Test Clock Input，TCK）
- 测试模式选择输入（Test Mode Select Input，TMS）
- 测试数据输入（Test Data Input，TDI）
- 测试数据输出（Test Data Output，TDO）

有关信号与器件引脚的连接，请参见具体器件的数据手册。

4.1.1 测试时钟输入（TCK）

TCK是通过指令或选定数据寄存器控制TAP控制器更新和数据移位的时钟。TCK在频率和相位上都不依赖于处理器时钟。

4.1.2 测试模式选择输入（TMS）

TMS是TAP控制器的控制信号。该信号在TCK的上升沿进行采样。

4.1.3 测试数据输入（TDI）

TDI是指令或选定数据寄存器的测试数据输入。对于某些TAP控制器状态，该信号在TCK的上升沿进行采样。

4.1.4 测试数据输出（TDO）

TDO是指令或数据寄存器的测试数据输出。该信号在TCK的下降沿发生变化。只有在移出数据时才会驱动TDO，否则TDO处于三态。

表4-1： 4线接口引脚

器件引脚名称	引脚类型	引脚说明
$\overline{\text{MCLR}}$	I	编程使能
ENVREG(2)	I	片上稳压器使能
VDD、VDDIO、VDDCORE(2)、VDDR1V8(2)、VBAT(2)和AVDD(1)	P	电源
VSS、VSS1V8(2)和AVSS(1)	P	地
VCAP(2)	P	CPU逻辑滤波电容连接
TDI	I	测试数据输入
TDO	O	测试数据输出
TCK	I	测试时钟
TMS	I	测试模式状态

图注： I = 输入　　O = 输出　　P = 电源

注 1： 所有电源和接地引脚都必须连接，包括模拟电源（AVDD）和模拟地（AVSS）。

2： 并非在所有器件上均提供该引脚。请参见具体器件数据手册中的**“引脚分配图”**或**“引脚分配表”**部分以确定可用性。

4.2 2线接口

另一个可用的接口是2线ICSP端口。表4-2列出了所需引脚连接。该接口使用以下两条通信线路来与要编程的PIC32器件交换数据：

- 串行编程时钟（PGECx）
- 串行编程数据（PGEDx）

下面两节将介绍这些信号。有关信号与芯片引脚的连接，请参见具体器件的数据手册。

4.2.1 串行编程时钟（PGECx）

PGECx是通过指令或选定数据寄存器控制TAP控制器更新和数据移位的时钟。PGECx在频率和相位上都不依赖于处理器时钟。

4.2.2 串行编程数据（PGEDx）

PGEDx是指令或选定数据寄存器的数据输入/输出，也是TAP控制器的控制信号。对于某些TAP控制器状态，该信号在PGECx的下降沿进行采样。

表4-2： 2线接口引脚

器件 引脚名称	编程器 引脚名称	引脚类型	引脚说明
$\overline{MCLR}$	MCLR	P	编程使能
ENVREG(2)	N/A	I	片上稳压器使能
VDD、VDDIO、VBAT(2)和AVDD(1)	VDD	P	电源
VDDCORE(2)和VDDR1V8(2)	N/A	P	DDR接口的电源
VSS、VSS1V8(2)和AVSS(1)	VSS	P	地
VCAP(2)	N/A	P	CPU逻辑滤波电容连接
PGECx	PGEC	I	主编程引脚对：串行时钟
PGEDx	PGED	I/O	主编程引脚对：串行数据

图注： I = 输入　　O = 输出　　P = 电源

注 1： 所有电源和接地引脚都必须连接，包括模拟电源（AVDD）和模拟地（AVSS）。

2： 并非在所有器件上均提供该引脚。请参见具体器件数据手册中的“**引脚分配图**”或“**引脚分配表**”部分以确定可用性。

© 2014-2020 Microchip Technology Inc.

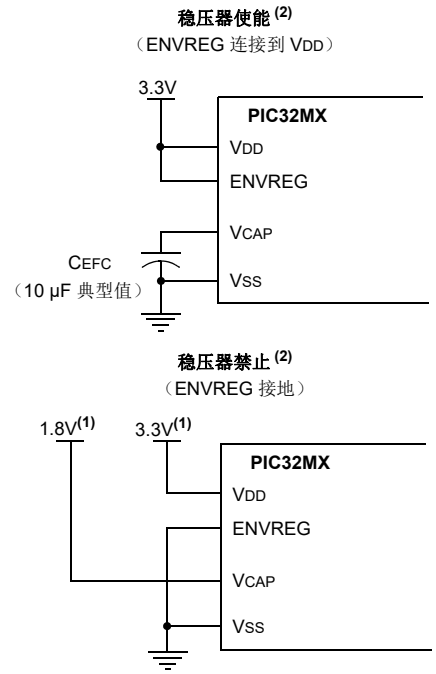
4.3 PIC32MX 电源要求

PIC32MX 系列器件均采用双电源设计。其中一个为内核供电，另一个为外设和 I/O 引脚供电。所有器件都包含一个用于低压内核电源的片上稳压器，因此无需附加外部稳压器。片上稳压器有三种实现方式：

- 第一个版本具有一个内部稳压器，可通过ENVREG引脚禁止。禁止时，必须使用外部电源为内核供电。如果使能，必须将一个低ESR滤波电容连接到VCAP引脚（见图4-2）。
- 第二个版本具有一个不能禁止的内部稳压器。必须始终将一个低ESR滤波电容连接到VCAP引脚。
- 第三个版本具有一个不能被禁止的内部稳压器，且无需滤波电容。

有关器件的电源要求，请参见**第21.0节“交流/直流特性和时序要求”**和具体器件数据手册中的**“电气特性”**章节。

图4-2： 内部稳压器使能/禁止选项



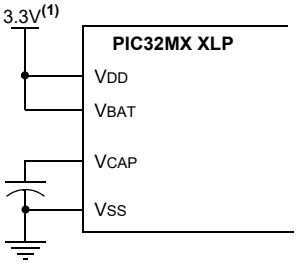
注 1： 这些是典型的工作电压值。有关 VDD 和 VCAP 的完整工作范围，请参见**第21.0节“交流/直流特性和时序要求”**。

2： 并非所有器件都支持稳压器使能和稳压器禁止模式。请参见具体器件的数据手册，以确定可用性。

4.4 具有VBAT引脚的PIC32MX 电源要求

PIC32MX 系列中的某些器件提供了 VBAT 引脚，可在编程期间将此引脚连接到VDD电源。请参见图4-3。

图4-3： 具有VBAT引脚的PIC32MX 电源连接



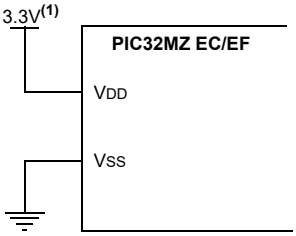
注 1： 这是典型的工作电压值。有关 VDD 的完整工作范围，请参见**第21.0节“交流/直流特性和时序要求”**。

4.5 PIC32MZ EC和PIC32MZ EF 电源要求

PIC32MZ EC和PIC32MZ EF 系列器件与PIC32MX器件一样，同样采用双电源设计。但是，内部稳压器不需要外部滤波电容，并且没有相应的VCAP或ENVREG 引脚。请参见图4-4。

有关器件的电源要求，请参见**第21.0节“交流/直流特性和时序要求”**和具体器件数据手册中的**“电气特性”**章节。

图4-4： PIC32MZ EC/EF 电源连接



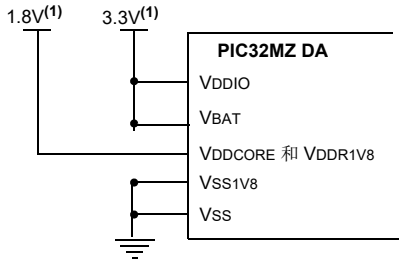
注 1： 这是典型的工作电压值。有关 VDD 的完整工作范围，请参见**第21.0节“交流/直流特性和时序要求”**。

4.6 PIC32MZ DA 电源要求

PIC32MZ DA 系列器件均采用四电源设计。其中两个电源与 PIC32MZ EC 和 PIC32MZ EF 电源相同。第三个电源用于 DDR 存储器接口，需要 1.8V 电压。第四个电源用于 VBAT 引脚，但可连接到 VDD 电源。请参见图 4-5。

有关器件的电源要求，请参见**第 21.0 节“交流/直流特性和时序要求”**和具体器件数据手册中的**“电气特性”**章节。

图 4-5： PIC32MZ DA 电源连接

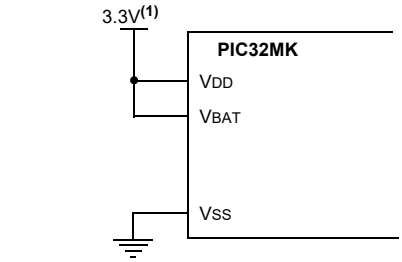


注 1： 这些是典型的工作电压值。有关 VDDIO、VBAT、VDDCORE 和 VDDR1V8 的完整工作范围，请参见**第 21.0 节“交流/直流特性和时序要求”**。

4.7 PIC32MK 电源要求

PIC32MK 系列器件均采用三电源设计。其中两个电源与 PIC32MZ EC 和 PIC32MZ EF 电源相同。第三个电源用于 VBAT 引脚，但可连接到 VDD 电源。请参见图 4-6。

图 4-6： PIC32MK 电源连接



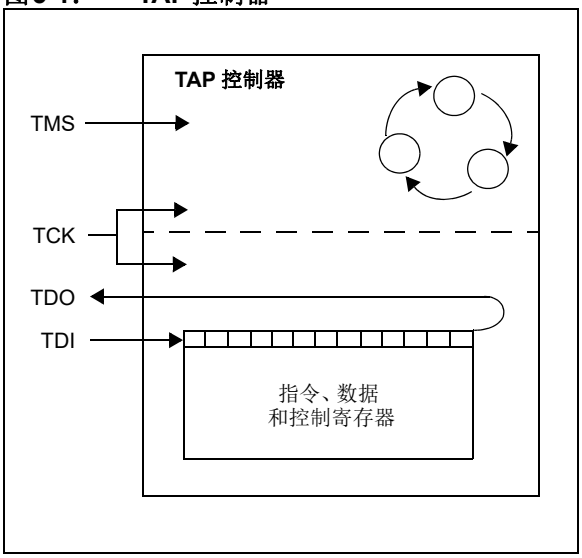
注 1： 这些是典型的工作电压值。有关 VDD 和 VBAT 的完整工作范围，请参见**第 21.0 节“交流/直流特性和时序要求”**。

5.0 EJTAG 与 ICSP

通过 CPU 内核中的 EJTAG 模块完成编程。EJTAG 与一组完整的 JTAG 引脚连接，或者与精简的 ICSP 模式 2 线转 4 线 EJTAG 接口连接。在两种模式下，PIC32 闪存的编程都是通过 ETAP 控制器完成。TAP 控制器使用 TMS 引脚来确定是否应该在 TDI 和 TDO 之间的移位路径中访问指令或数据寄存器（见图 5-1）。

用于编程的基本 EJTAG 概念是使用一个称为 DMSEG 的特殊存储区（0xFF200000 到 0xFF2FFFFF），该区域只有在处理器以调试模式运行时才可用。所有指令以串行方式移入内部缓冲区，然后装入指令寄存器并由 CPU 执行。指令按 32 位一组的形式送入 ETAP 状态机。

图 5-1： TAP 控制器

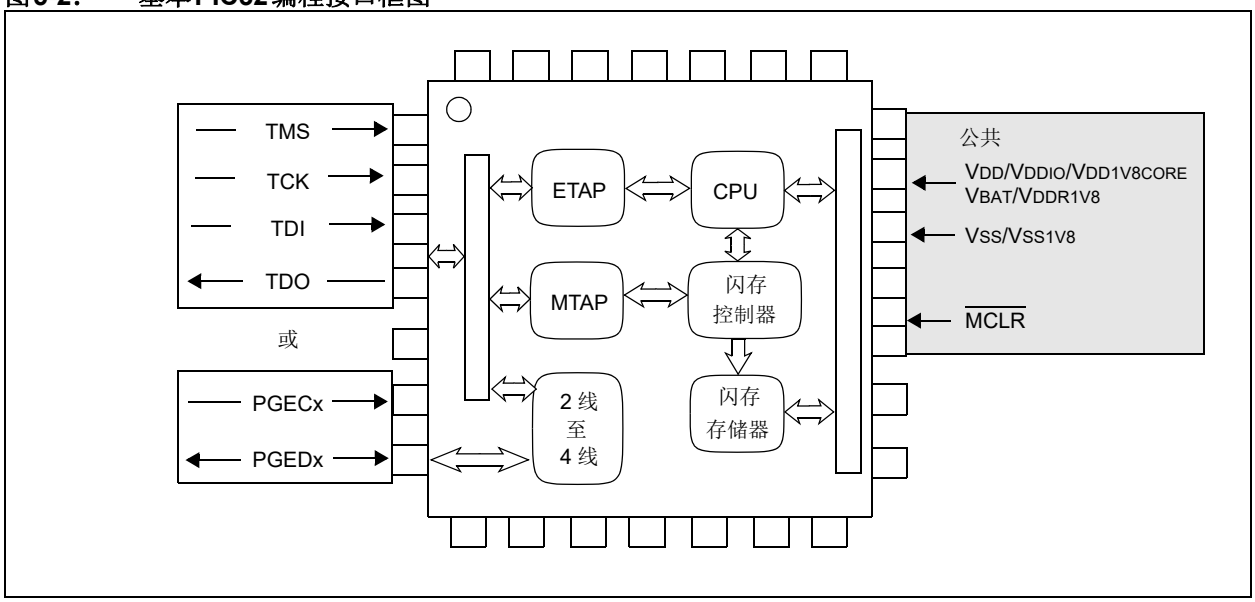


© 2014-2020 Microchip Technology Inc.

5.1 编程接口

图5-2显示了PIC32器件中的基本编程接口。后面几节将介绍各个接口模块。

图5-2： 基本PIC32编程接口框图



5.1.1 ETAP

此模块以串行方式将指令和数据馈送到CPU中。

5.1.2 MTAP

除了EJTAG TAP（ETAP）控制器，PIC32器件还使用另一个专有TAP控制器来执行其他操作。Microchip TAP（MTAP）控制器支持与编程相关的两条指令：MTAP_COMMAND和TAP切换指令。有关完整的指令列表，请参见表20-1。MTAP_COMMAND指令为JTAG探针提供了通过其数据寄存器将命令发送到器件的机制。

编程器发送命令的方式是：通过SendCommand伪操作移入MTAP_COMMAND指令，然后通过XferData伪操作发送MTAP_COMMAND DR命令。有关具体命令，请参见表20-2。

探针无需为移入数据寄存器的每条命令都发出MTAP_COMMAND指令。

5.1.3 2线转4线

此模块可将2线ICSP接口转换为4线JTAG接口。

5.1.4 CPU

CPU通过内部振荡器以8 MHz的速率执行指令。

5.1.5 闪存控制器

闪存控制器用于控制器件上闪存的擦除和编程。

5.1.6 闪存

PIC32器件闪存分为两个逻辑闪存分区，包括引导闪存（Boot Flash Memory，BFM）和程序闪存（Program Flash Memory，PFM）。BFM从地址0x1FC00000开始，PFM则从地址0x1D000000开始。每个闪存分区都可分成多页，页表示可以擦除的最小存储块。页大小可为256字（1024字节）、1024字（4096字节）或4096字（16,384字节），具体取决于器件。行大小表示使用行编程命令编程的字数。一页内始终有8行；因此，页大小为256、1024和4096字的器件的行大小分别为32、128和512字。表5-1列出了每个器件系列的PFM、BFM、行和页大小。

BFM的存储单元保留用作器件配置寄存器；更多信息，请参见**第19.0节“配置存储器和器件ID”**。

表5-1：　代码存储区大小

PIC32器件	行大小（字）	页大小（字）	引导闪存地址（字节）（见注1）	编程执行程序（见注2和注3）
PIC32MX 110/120/130/150/170 210/220/230/350/270（仅限28/36/44引脚器件）	32	256	0x1FC00000-0x1FC00BFF（3 KB）	RIPE_11_aabbcc.hex
PIC32MX 120/130/150/170/230/250/270/530/550/570（仅限64/100引脚器件）				
PIC32MX 15X/17X/25X/27X（仅限28/44引脚器件）			0x1FC00000-0x1FC02FFF（12 KB）	
PIC32MX 330/350/370/430/450/470	128	1024	0x1FC00000-0x1FC02FFF（12 KB）	RIPE_06_aabbcc.hex
PIC32MX 320/340/360/420/440/460				
PIC32MX 534/564/664/764				
PIC32MX 575/675/695/795				
PIC32MK 0512/1024XXD/E/F/K/L/M	128	1024	0x1FC00000-0x1FC04FFF（20 KB） 0x1FC20000-0x1FC24FFF（20 KB）	RIPE_15a_aabbcc.hex
PIC32MK 0256/0512XXG/H	128	1024	0x1FC00000-0x1FC04FFF（20 KB）	RIPE_15a_aabbcc.hex
PIC32MZ 05XX/10XX/20XX	512	4096	0x1FC00000-0x1FC13FFF（80 KB） 0x1FC20000-0x1FC33FFF（80 KB）	RIPE_15_aabbcc.hex

注 1： 闪存程序存储器的地址范围取决于程序闪存大小，具体如下所示：

- 0x1D000000-0x1D003FFF（16 KB）
- 0x1D000000-0x1D007FFF（32 KB）
- 0x1D000000-0x1D00FFFF（64 KB）
- 0x1D000000-0x1D01FFFF（128 KB）
- 0x1D000000-0x1D03FFFF（256 KB）
- 0x1D000000-0x1D07FFFF（512 KB）
- 0x1D000000-0x1D0FFFFF（1024 KB）
- 0x1D000000-0x1D1FFFFF（2048 KB）

并非每个系列都支持所有闪存程序存储器大小。

2： 编程执行程序可以从Microchip网站的相关产品页面获取，也可能位于以下MPLAB® X IDE安装文件夹中：
…\Microchip\MPLABX\<版本>\mplab_ide\mplablibs\modules\ext\REALICE.jar
…\Microchip\MPLABX\<版本>\mplab_ide\mplablibs\modules\ext\ICD3.jar
…\Microchip\MPLABX\<版本>\mplab_ide\mplablibs\modules\ext\PICKIT3.jar

3： 文件名的最后几个字符aabbcc因文件版本而异。

© 2014-2020 Microchip Technology Inc.

5.2 4线JTAG详细信息

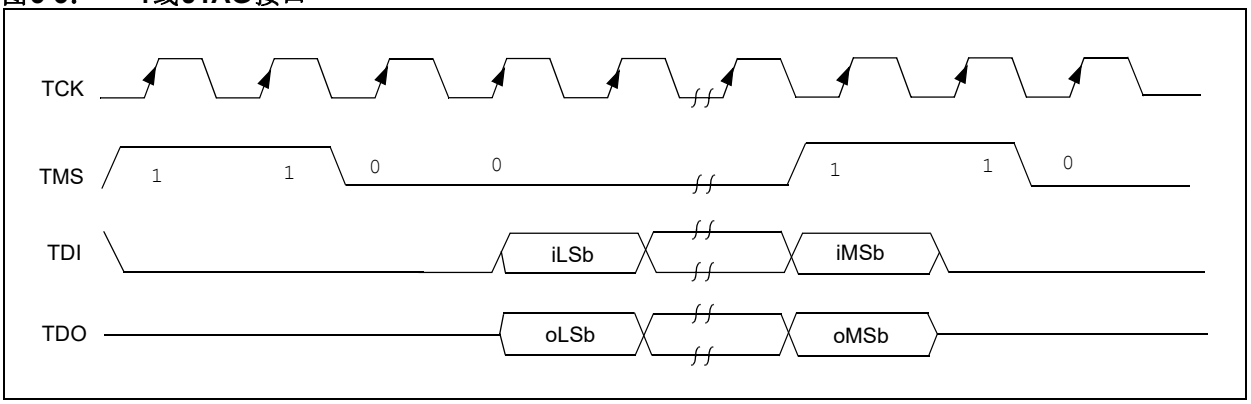
4线接口使用标准JTAG（IEEE 1149.1-2001）接口信号。

- TCK：测试时钟——驱动数据输入/输出
- TMS：测试模式选择——选择工作模式
- TDI：测试数据输入——数据移入器件
- TDO：测试数据输出——数据移出器件

由于只有一条数据线可用，因此必须采用串行传输协议（如SPI）。时钟输入位于TCK引脚。通过TMS引脚逐位操控状态机来执行配置。在TDI和TDO引脚上每个TCK时钟脉冲输入和输出一位数据。可以装载不同的指令模式，以读取芯片ID或操控芯片功能。

传送到TDI的数据必须在TCK上升沿之前的芯片特定建立时间内以及TCK上升沿之后的保持时间内保持有效。TDO数据在TCK下降沿之后的芯片特定时间内保持有效（见图5-3）。

图5-3： 4线JTAG接口



5.3 2线ICSP详细信息

在ICSP模式下，2线ICSP信号以时分复用的方式移入2线转4线模块。然后，2线转4线模块将转换信号，使其相当于TAP控制器的4线JTAG端口。下面是支持的两种工作模式：

- 4相ICSP
- 2相ICSP

5.3.1 4相ICSP

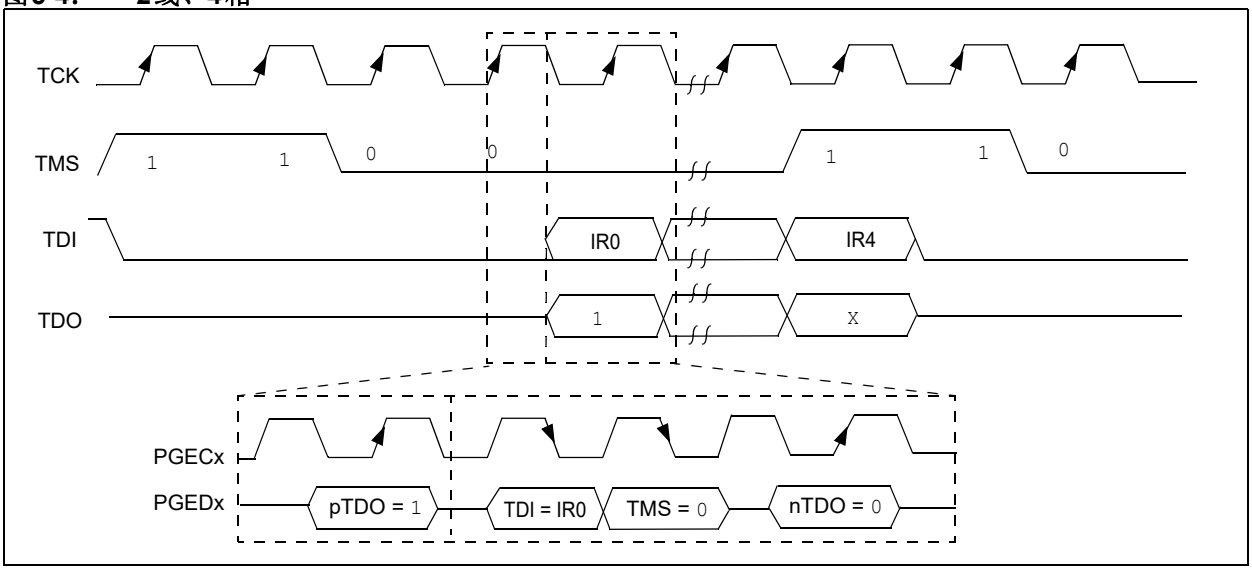
在4相ICSP模式下，TDI、TDO和TMS器件引脚在四个时钟内复用到PGEDx，请参见图5-4。先移入最低有效位（Least Significant bit，LSb）；TDI和TMS在PGECx的下降沿进行采样，而TDO则在PGECx的下降沿进行驱动。读取和写入数据传输都使用4相ICSP模式。

5.3.2 2相ICSP

在2相ICSP模式下，TMS和TDI器件引脚在两个时钟内复用到PGEDx，请参见图5-5。先移入LSb；TDI和TMS在PGECx的下降沿进行采样。此模式下不提供TDO输出。2相ICSP模式旨在加快2线只写事务的速度。

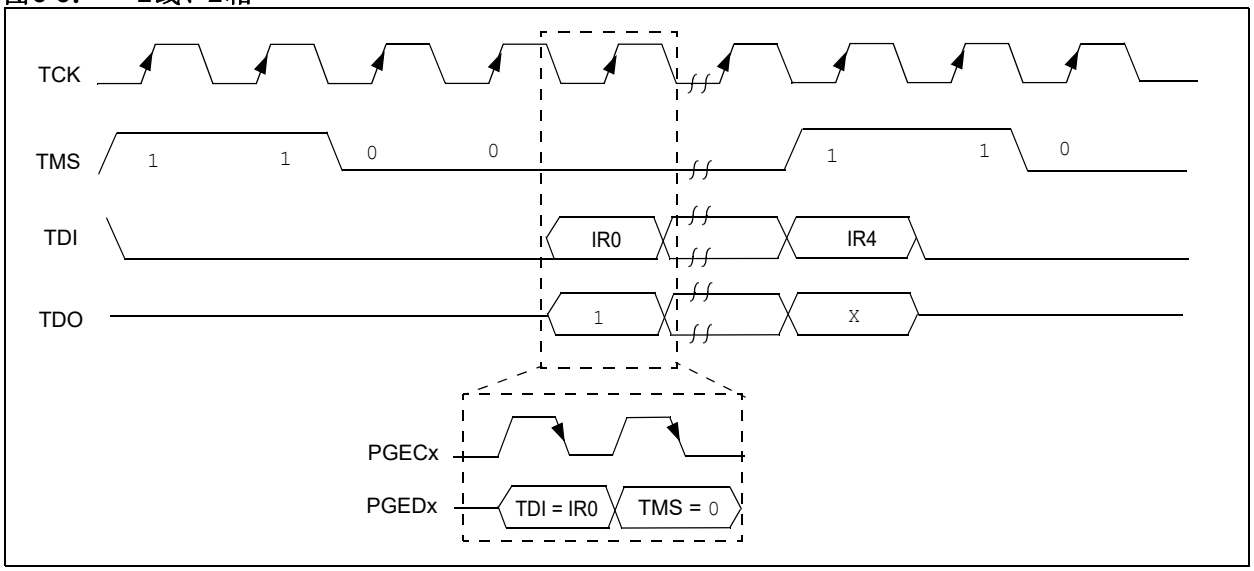
注：	只有到下一个数据包的第一个时钟时，才会实际执行数据包。要进入2线、2相ICSP模式，TDOEN位（DDPCON<0>或CFGCON<0>）必须设为0。

图5-4： 2线、4相



© 2014-2020 Microchip Technology Inc.

图5-5： 2线、2相



5.3.3 同步

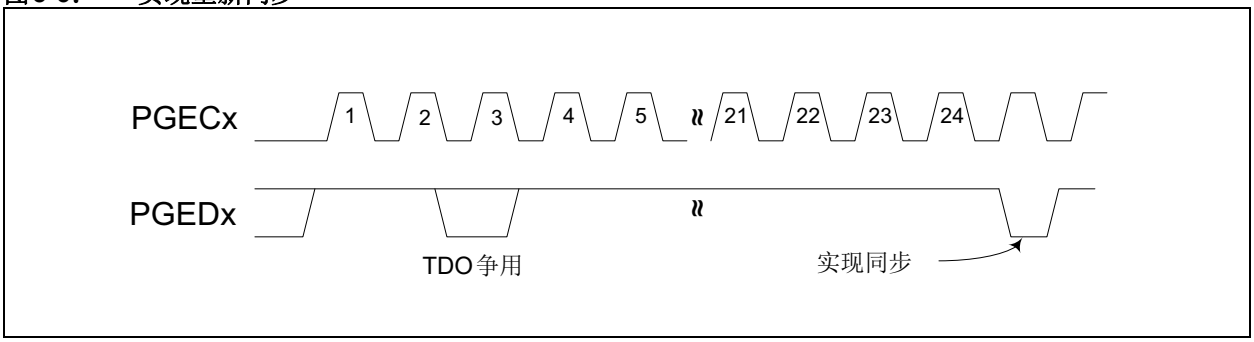
如果所连接的编程器失去同步，则某些PIC32器件可复位内部EJTAG状态机。PGCx信号上存在噪声时，可能会发生这种情况。

要实现重新同步，PGEDx引脚在24个PGECx时钟周期内保持高电平。这将强制五个TMS事件进入EJTAG控制器，并将EJTAG状态机置于测试空闲复位状态。有关如何实现重新同步的示例，请参见图5-6。

将PGEDx引脚置为高电平时，该引脚上可能会发生争用，因为器件可能会在在线仿真器驱动输入期间尝试将TDO输出驱动到该引脚上。这种情况最长只会持续一个周期，因为TMS高电平会使EJTAG状态机提前退出移位IR或移位DR状态。

不支持在2线、2相模式下进行同步。

图5-6： 实现重新同步



6.0 伪操作

为简化编程详细信息的说明，所有操作都将使用伪操作进行说明。伪代码描述中使用了多个函数。这些函数用于提高伪代码的可读性、抽象化特定于实现的行为，或者两者兼有。使用伪操作解析参数时，将使用以下语法：

- `5'h0x03`——发送3的5位十六进制值
- `6'b011111`——发送31的6位二进制值

本节定义了这些函数，包括以下操作：

- **SetMode** (mode)
- **`SendCommand`** (command)
- oData = **XferData** (iData)
- oData = **XferFastData** (iData)
- oData = **`XferInstruction`** (instruction)

6.1 `SetMode`伪操作

格式：
SetMode (mode)

目的：
将EJTAG状态机设置为特定状态。

说明：
模式的值随时钟移入器件的TMS信号引脚。TDI设置为0，TDO会被忽略。

限制：
无。

示例：
SetMode (`6'b011111`)

图6-1： SetMode 4线

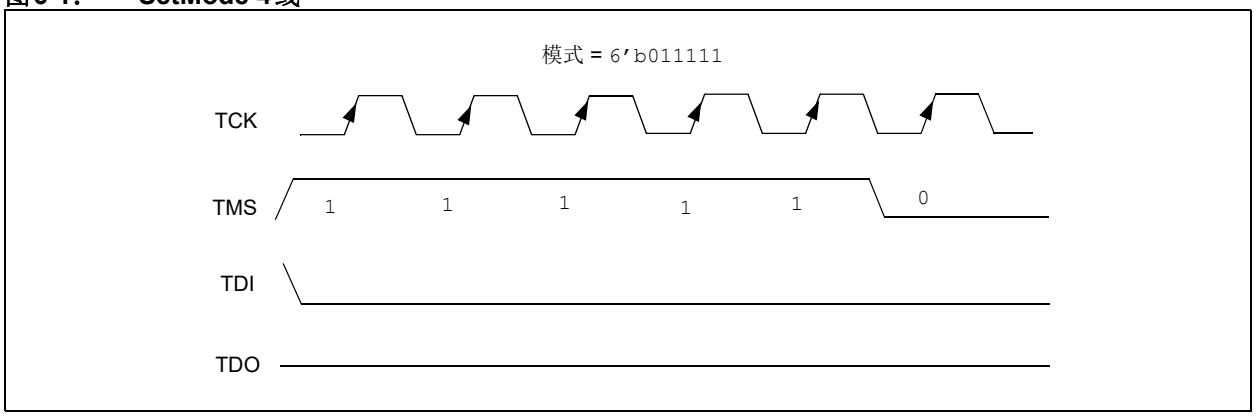
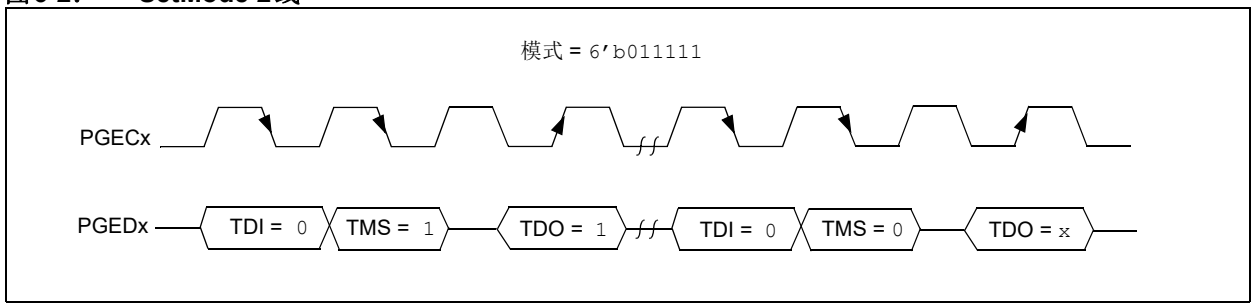


图6-2： SetMode 2线



© 2014-2020 Microchip Technology Inc.

6.2 SendCommand伪操作

格式：

SendCommand (command)

目的：

发送命令以选择特定TAP寄存器。

说明（按顺序）：

1. 将TMS头部移入器件，以选择移位IR状态。
2. 信号TMS保持为低电平时，将命令移入器件的TDI引脚。
3. 在将TMS设置为高电平时，移入命令的最后一位，即最高有效位（Most Significant bit，MSb）。
4. 将TMS尾部移入器件的TMS引脚，使TAP控制器恢复为运行/测试空闲状态。

限制：

无。

示例：

SendCommand (5'h0x07)

图6-3： SendCommand 4线

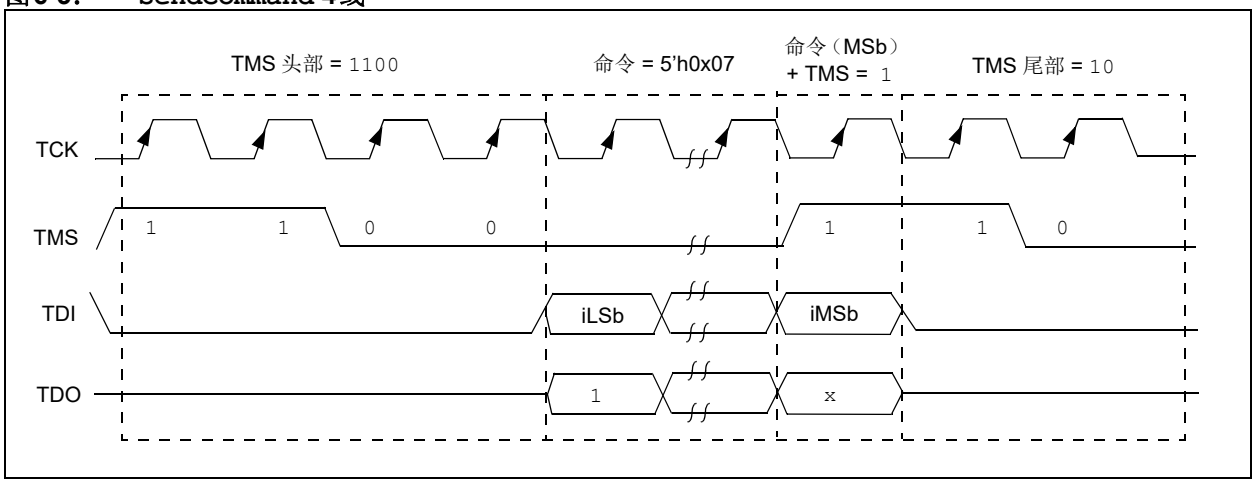
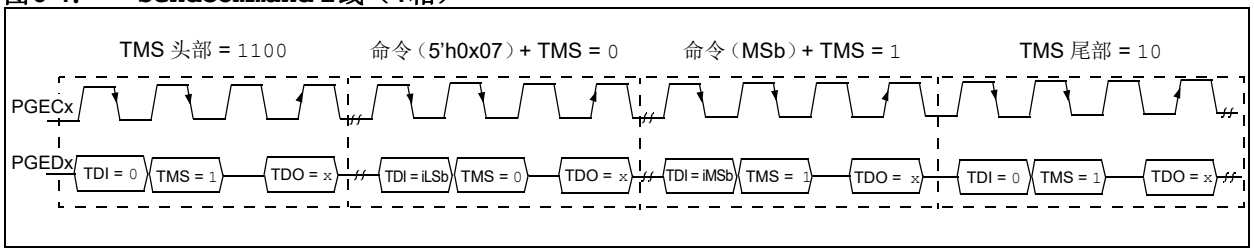


图6-4： SendCommand 2线（4相）



6.3 XferData伪操作

格式：
oData = **XferData** (iData)

目的：
将数据移入和移出命令选择的寄存器。

说明（按顺序）：
1. 将TMS头部移入器件，以选择移位DR状态。
2. 信号 TMS 保持为低电平时，将数据移入 / 移出器件的TDI/TDO引脚。
3. 在将 TMS 设置为高电平时，移入 / 移出数据的最后一位MSb。
4. 将 TMS 尾部移入器件的 TMS 引脚，使 TAP 控制器恢复为运行 / 测试空闲状态。

限制：
无。

示例：
oData = XferData (32'h0x12)

图6-5： XferData 4线

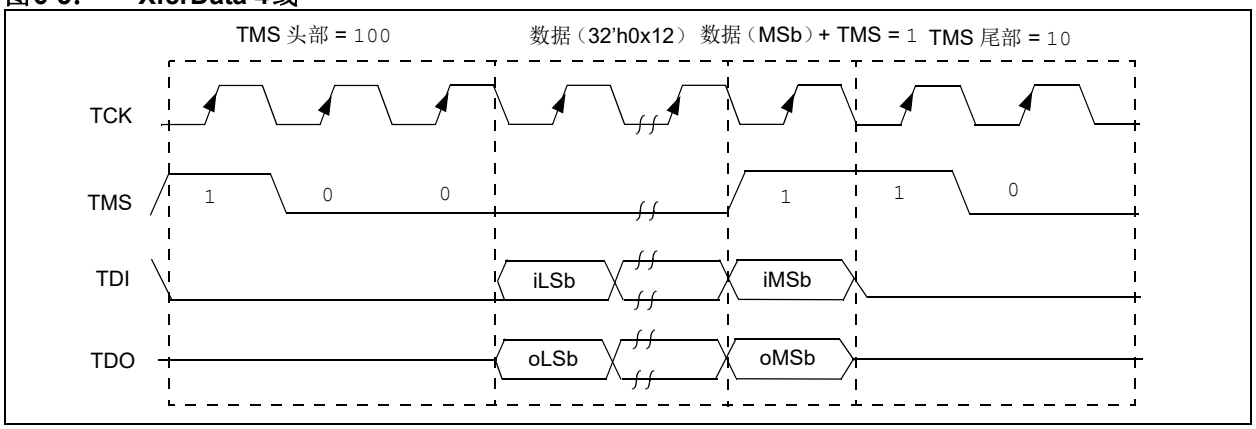
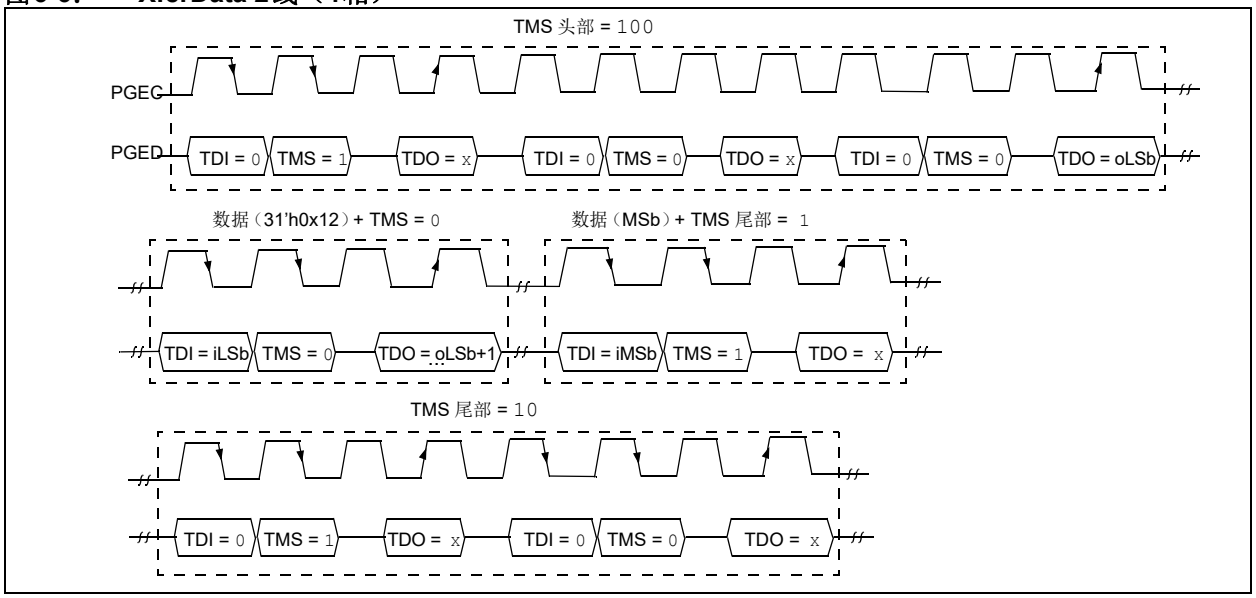


图6-6： XferData 2线（4相）



© 2014-2020 Microchip Technology Inc.

6.4 XferFastData伪操作

格式：
oData = **XferFastData** (iData)

目的：
将32位数据快速输入/输出器件。

说明（按顺序）：

1. 将TMS头部移入器件，以选择移位DR状态。

> **注：** 对于2线（4相）——在最后一个时钟，移入TMS头部的同时，oPrAcc位从TDO移出。如果oPrAcc的值不是1，则必须重复整个操作。

2. 移入PrAcc位的输入值，即0。

> **注：** 对于2线（4相）——在此操作期间，TDO将成为输出数据的LSb。移入输入数据剩余的31位，并移出输出数据的31位。对于输入数据的最后一位，将设置TMS尾部 = 1。

3. 移入TMS尾部 = 10，使TAP控制器恢复为运行/测试空闲状态。

限制：
必须先发送SendCommand (ETAP_FASTDATA)，以选择Fastdata寄存器，如示例6-1中所示。有关命令的详细说明，请参见表20-4。

> **注：** 仅在与PE通信时，才使用2相XferData。更多信息，请参见**第17.0节“编程执行程序”**。

示例6-1： SendCommand

```
// Select the Fastdata Register
SendCommand(ETAP_FASTDATA)
// Send/Receive 32-bit Data
oData = XferFastData(32'h0x12)
```

图6-7： XferFastData 4线

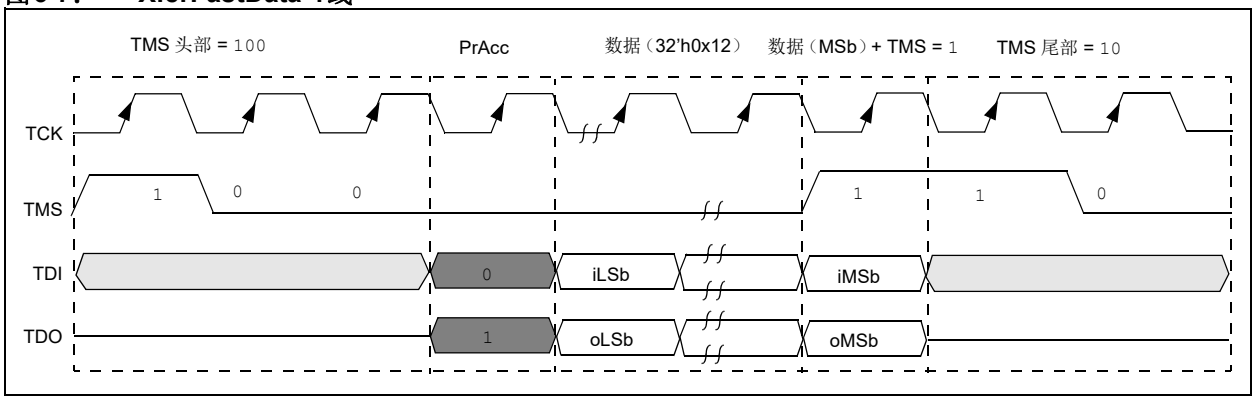


图6-8： XferFastData 2线（2相）

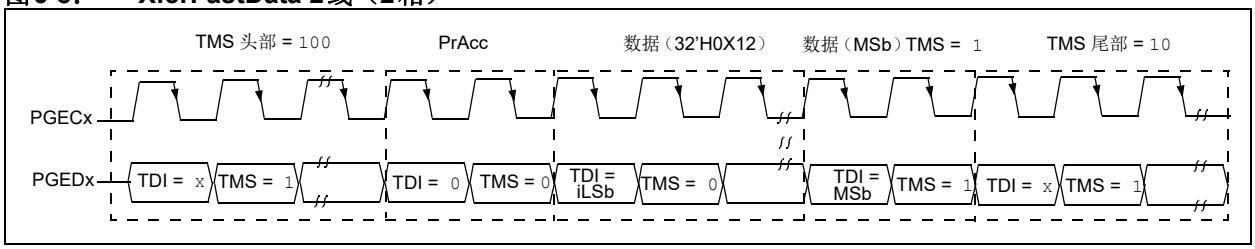
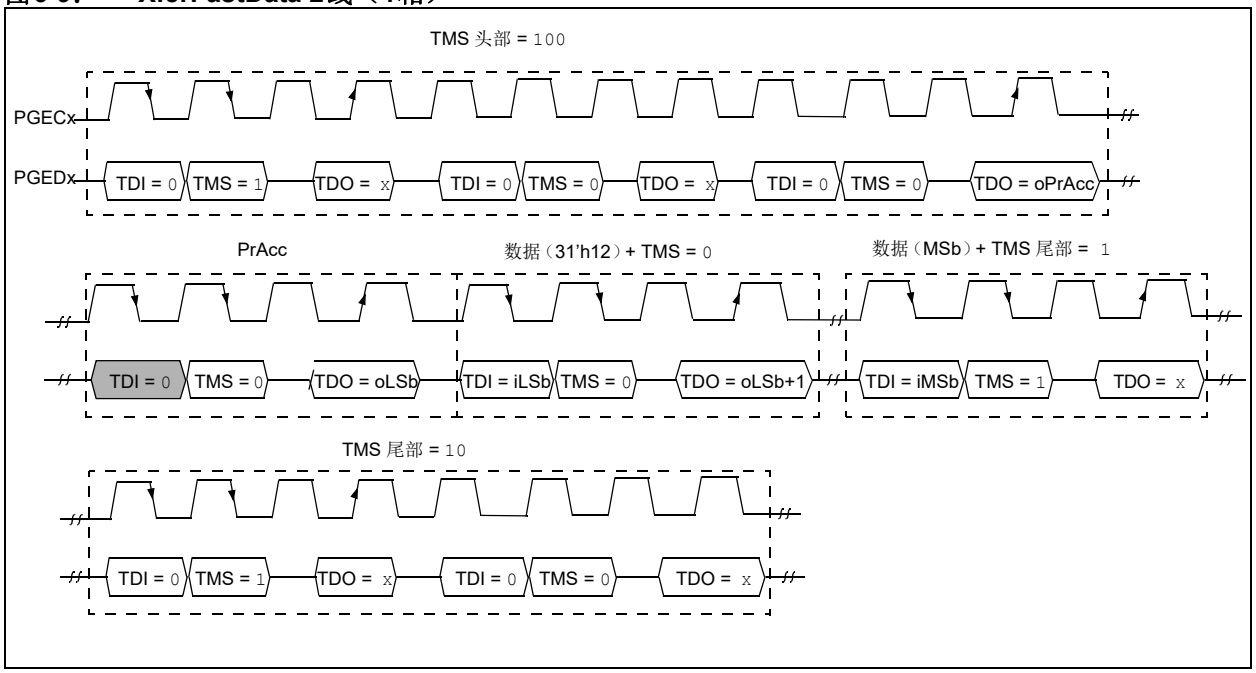


图6-9： XferFastData 2线（4相）



© 2014-2020 Microchip Technology Inc.

6.5 XferInstruction伪操作

格式：

XferInstruction (instruction)

目的：

发送用于器件执行的32位数据。

说明：

将指令移入器件，然后由CPU执行。

限制：

器件必须处于调试模式。

示例6-2： XferInstruction

```
XferInstruction (instruction)
{
    // Select Control Register
    SendCommand(ETAP_CONTROL);
    // Wait until CPU is ready
    // Check if Processor Access bit (bit 18) is set
    do {
        controlVal = XferData(32'h0x0004C000);
    } while( PrAcc(contorlVal<18>) is not '1' );

    // Select Data Register
    SendCommand(ETAP_DATA);

    // Send the instruction
    XferData(instruction);

    // Tell CPU to execute instruction
    SendCommand(ETAP_CONTROL);
    XferData(32'h0x0000C000);
}
```

6.6 ReadFromAddress伪操作

格式：

oData = ReadFromAddress (address)

目的：

将32位数据发送到器件存储器。

说明：

从存储器中“address”参数指定的地址读取32位数据。

限制：

器件必须处于调试模式。

示例6-3： ReadFromAddress（适用于PIC32MX、PIC32MZ和PIC32MK器件）

```
ReadFromAddress (address)
{

// Load Fast Data register address to s3
instruction = 0x3c130000;
instruction |= (0xff200000>>16)&0x0000ffff;
XferInstruction(instruction); // lui s3, <FAST_DATA_REG_ADDRESS(31:16)> - set address of fast
data register

// Load memory address to be read into t0
instruction = 0x3c080000;
instruction |= (address>>16)&0x0000ffff;
XferInstruction(instruction); // lui t0, <DATA_ADDRESS(31:16)> - set address of data
instruction = 0x35080000;
instruction |= (address&0x0000ffff);
XferInstruction(instruction); // ori t0, <DATA_ADDRESS(15:0)> - set address of data

// Read data
XferInstruction(0x8d090000); // lw  t1, 0(t0)

// Store data into Fast Data register
XferInstruction(0xae690000); // sw  t1, 0(s3) - store data to fast data register
XferInstruction(0); // nop

// Shift out the data
SendCommand(ETAP_FASTDATA);
oData = XferFastData(32'h0x00000000);

return oData;

}
```


© 2014-2020 Microchip Technology Inc.

6.7 Synchronize伪操作

格式：

```
Synchronize ()
```

目的：

将EJTAG状态机复位为测试空闲复位状态。

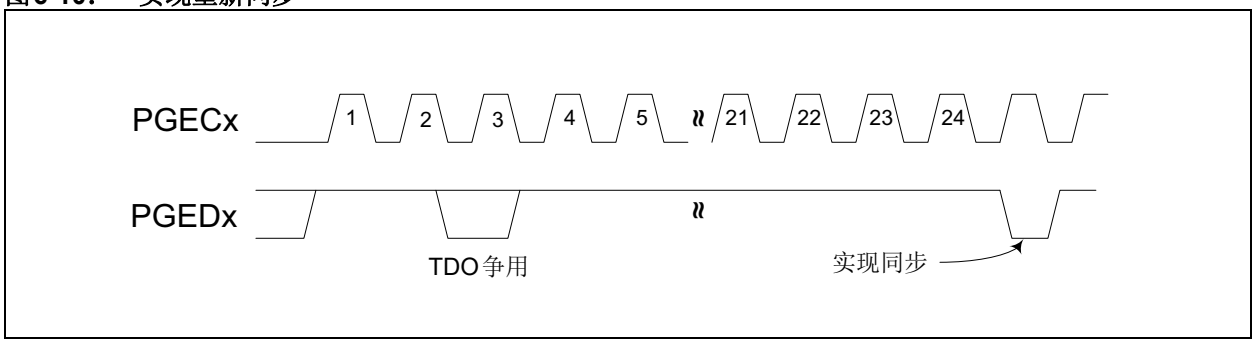
说明：

PGEDx信号在24个PGECx时钟周期内保持高电平。
所有其他信号将被忽略。

限制：

无。

图6-10： 实现重新同步



7.0 进入2线增强型ICSP模式

要使用2线PGEDx和PGECx引脚进行编程，必须将其使能。请注意，可以使用特定器件上任何一对可用的编程引脚，但是引脚必须成对使用。PGED1必须与PGEC1一起使用，以此类推。

注： 如果使用的是4线JTAG接口，则不必执行以下步骤。

进入2线增强型ICSP模式需要执行以下步骤：

1. 将$\overline{MCLR}$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PGEDx引脚。
3. $\overline{MCLR}$引脚随后将MCLR驱动为高电平并保持一段指定的时间。

有关时序要求，请参见**第21.0节“交流/直流特性和时序要求”**。

施加到$\overline{MCLR}$引脚的编程电压为VIH，对于PIC32器件，该电压实际就是VDD。对于VIH，没有最短保持时间要求。在移除VIH后，必须经过至少P18时间间隔才能将密钥序列移入PGEDx引脚。

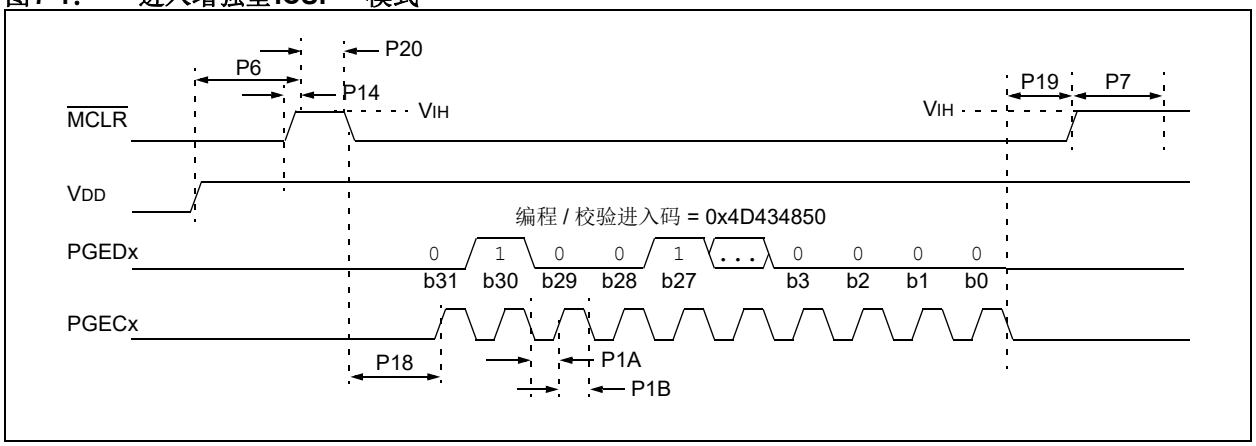
该密钥序列为一个特定的32位形式：0100 1101 0100 0011 0100 1000 0101 0000（首字母缩写词“MCHP”，使用ASCII表示）。只有在密钥序列有效时，器件才能进入编程/校验模式。首先必须移入高4位的MSb。

一旦密钥序列完全移入，就必须将VIH加到$\overline{MCLR}$引脚并在2线增强型ICSP接口下始终保持该电压。必须经过至少P19和P7的时间间隔才能将数据移入PGEDx引脚。在P7之前出现在PGEDx引脚上的信号被解析为无效。

成功进入后，即可执行后续章节中所述的编程操作。在2线增强型ICSP模式下，所有未使用的I/O都被置于高阻态。

注： 进入ICSP模式后将器件置于复位状态，以阻止执行指令。要解除复位状态，必须发出MCHP_DE_ASSERT_RST命令。

图7-1： 进入增强型ICSP™模式

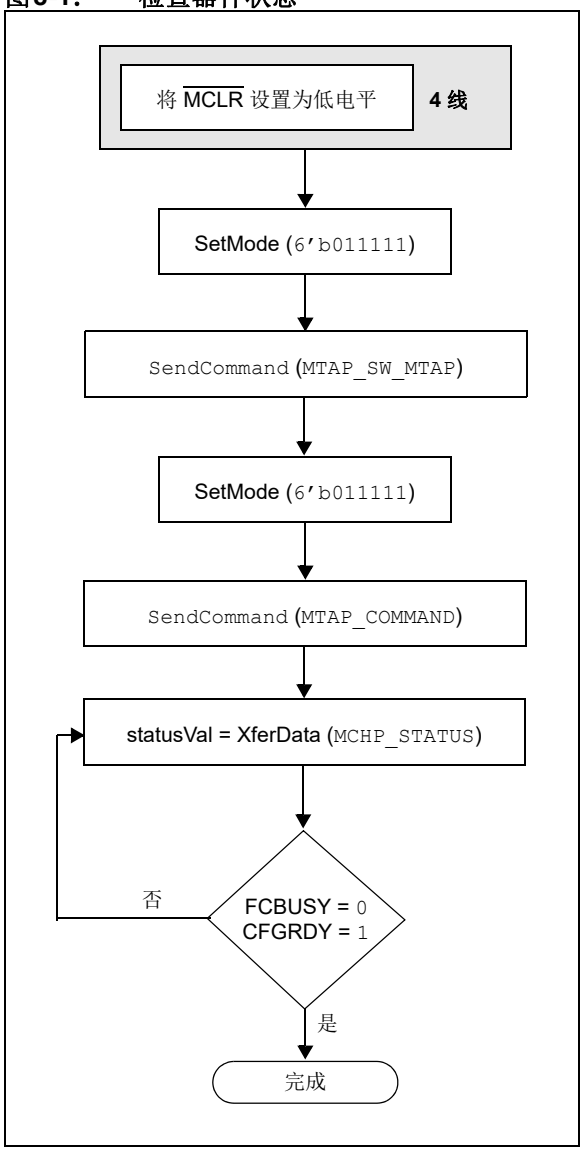


© 2014-2020 Microchip Technology Inc.

8.0 检查器件状态

在对器件进行编程之前，编程器必须先检查器件状态，以确保其准备好接收信息。

图8-1： 检查器件状态



8.1 4线接口

用于进入4线JTAG编程模式的设置序列应在将$\overline{\text{MCLR}}$引脚置为有效时执行。进入此编程模式后，可释放$\overline{\text{MCLR}}$引脚，以允许处理器执行指令或驱动端口。

要使用4线接口检查器件状态，需执行以下步骤：

1. 将$\overline{\text{MCLR}}$引脚设置为低电平。
2. 通过SetMode (6'b011111)强制芯片TAP控制器进入运行测试/空闲状态。
3. SendCommand (MTAP_SW_MTAP)。
4. 通过SetMode (6'b011111)强制芯片TAP控制器进入运行测试/空闲状态。
5. SendCommand (MTAP_COMMAND)。
6. statusVal = XferData (MCHP_STATUS)。
7. 如果CFGRDY（statusVal<3>）不为1且FCBUSY（statusVal<2>）不为0，则转到第5步。

注： 如果使用的是4线接口，则必须存在配置字所选择的振荡器源，才能访问闪存。在未编程器件中，振荡器源是内部FRC，可通过它访问闪存。如果已将配置字重新编程选择外部振荡器源，则必须存在该振荡器源，才能访问闪存。有关使用配置字设置选择振荡器的详细信息，请参见特定器件数据手册中的**“特殊功能”**章节。

8.2 2线接口

要使用2线接口检查器件状态，需要执行以下步骤：

1. 通过SetMode (6'b011111)强制芯片TAP控制器进入运行测试/空闲状态。
2. SendCommand (MTAP_SW_MTAP)。
3. 通过SetMode (6'b011111)强制芯片TAP控制器进入运行测试/空闲状态。
4. SendCommand (MTAP_COMMAND)。
5. statusVal = XferData (MCHP_STATUS)。
6. 如果CFGRDY（statusVal<3>）不为1且FCBUSY（statusVal<2>）不为0，则转到第4步。

注： 如果CFGRDY和FCBUSY位未在10 ms内进入正确状态，则说明未正确执行序列或者器件已损坏。

9.0 擦除器件

在对器件进行编程前，必须先将其擦除。擦除操作会向闪存中写入全1，使其准备好编程新的一组数据。擦除器件后，可以通过执行“空白检查”操作进行校验。更多信息，请参见**第9.1节“空白检查”**。

擦除程序存储器（程序存储区、引导存储区和配置存储区）的步骤包括选择MTAP和发送MCHP_ERASE命令。编程器必须通过读取和校验MCHP_STATUS值中的位，等待擦除操作完成。图9-1显示了执行全片擦除的过程。

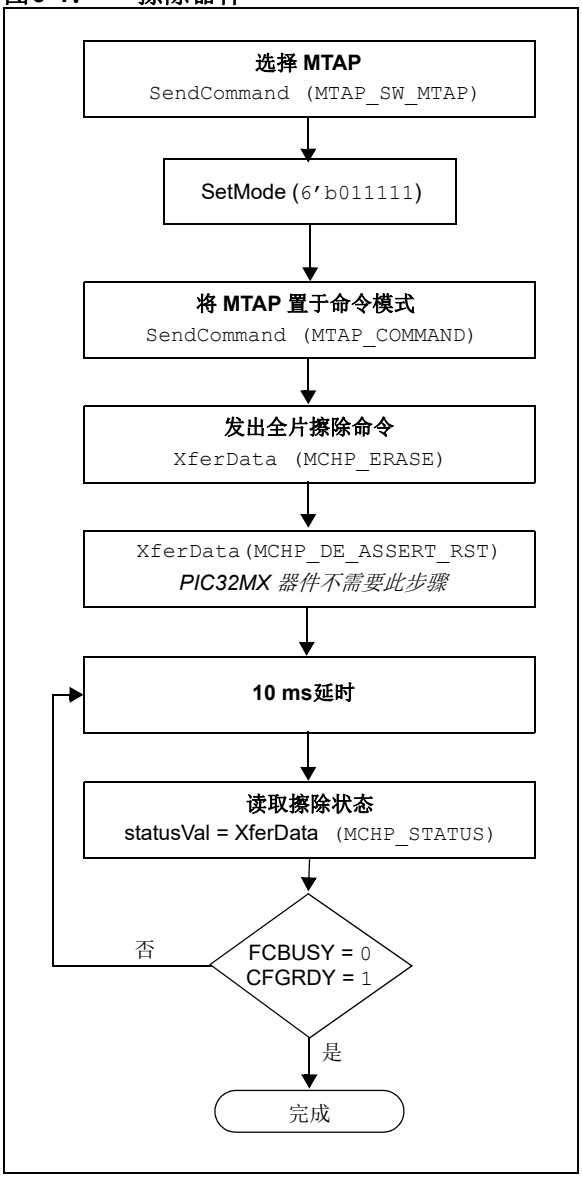
注： 器件ID存储单元为只读，无法擦除。因此，全片擦除对这些存储单元没有影响。

要擦除目标器件，需要完成以下步骤：

1. SendCommand (MTAP_SW_MTAP)。
2. SetMode (6'b011111)。
3. SendCommand (MTAP_COMMAND)。
4. XferData (MCHP_ERASE)。
5. XferData (MCHP_DE_ASSERT_RST)。*PIC32MX器件不需要此步骤。*
6. 延时10 ms。
7. statusVal = XferData (MCHP_STATUS)。
8. 如果CFGRDY（statusVal<3>）不为1且FCBUSY（statusVal<2>）不为0，则转到第5步。

注： 全片擦除操作是自定时操作。如果未在指定的全片擦除时间内将FCBUSY和CFGRDY位正确置1，则说明未正确执行序列或者器件已损坏。

图9-1： 擦除器件



9.1 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。

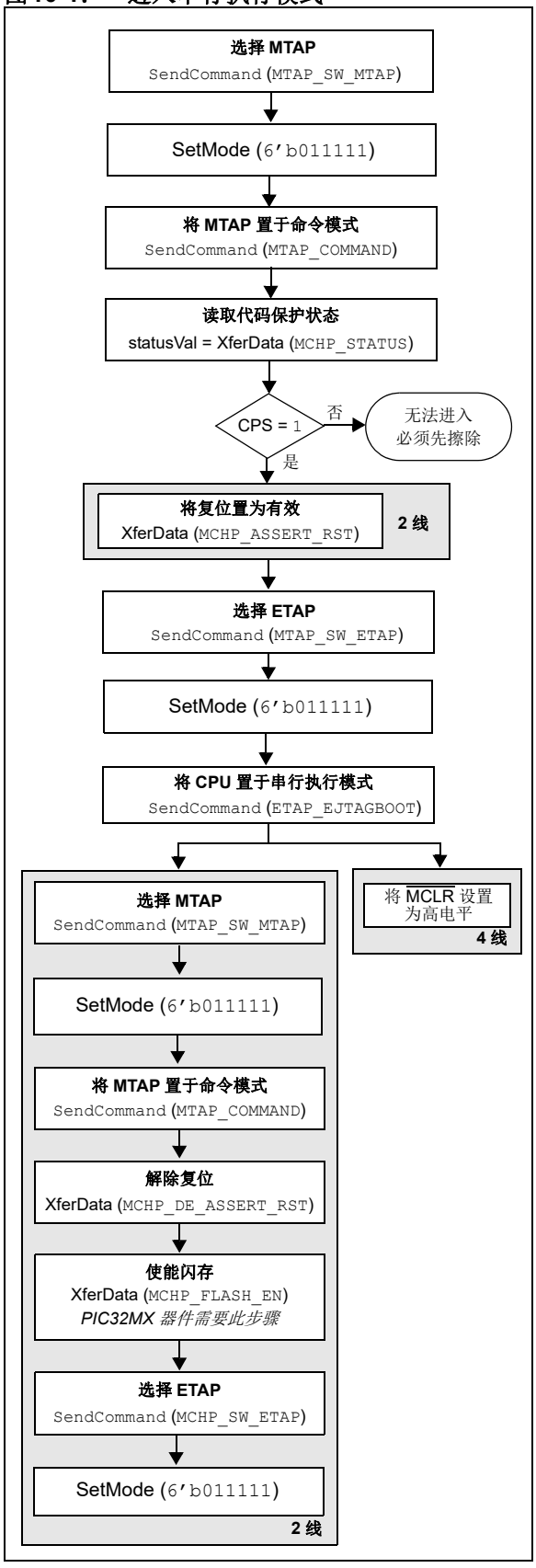
器件配置寄存器将被空白检查忽略。另外，所有未实现的存储空间都应该被空白检查忽略。

© 2014-2020 Microchip Technology Inc.

10.0 进入串行执行模式

在对器件进行编程之前，必须先将器件置于串行执行模式。进入串行执行模式的过程包括验证器件是否受代码保护。如果器件受代码保护，则必须执行全片擦除操作。有关详细信息，请参见**第9.0节“擦除器件”**。

图10-1： 进入串行执行模式



10.1 4线接口

进入串行执行模式需要执行以下步骤：

注： 假定$\overline{\text{MCLR}}$引脚在前面的“检查器件状态”步骤（见图8-1）中已驱动为低电平。

1. SendCommand (MTAP_SW_MTAP)。
2. SetMode (6'b011111)。
3. SendCommand (MTAP_COMMAND)。
4. statusVal = XferData (MCHP_STATUS)。
5. 如果CPS（statusVal<7>）不为1，则必须先擦除器件。
6. SendCommand (MTAP_SW_ETAP)。
7. SetMode (6'b011111)。
8. SendCommand (ETAP_EJTAGBOOT)。
9. 将$\overline{\text{MCLR}}$引脚设置为高电平。

10.2 2线接口

进入串行执行模式需要执行以下步骤：

1. SendCommand (MTAP_SW_MTAP)。
2. SetMode (6'b011111)。
3. SendCommand (MTAP_COMMAND)。
4. statusVal = XferData (MCHP_STATUS)。
5. 如果CPS（statusVal<7>）不为1，则必须先擦除器件。
6. XferData (MCHP_ASSERT_RST)。
7. SendCommand (MTAP_SW_ETAP)。
8. SetMode (6'b011111)。
9. SendCommand (ETAP_EJTAGBOOT)。
10. SendCommand (MTAP_SW_MTAP)。
11. SetMode (6'b011111)。
12. SendCommand (MTAP_COMMAND)。
13. XferData (MCHP_DE_ASSERT_RST)。
14. XferData (MCHP_FLASH_ENABLE)。*PIC32MX系列器件需要此步骤。*
15. SendCommand (MTAP_SW_ETAP)。
16. SetMode (6'b011111)。

© 2014-2020 Microchip Technology Inc.

11.0 下载编程执行程序（PE）

PE驻留在RAM存储器中，由CPU执行以对器件编程。PE使用一个简单的命令集和通信协议为编程器提供了对PIC32器件进行编程和校验的机制。PE可提供以下几项基本功能：

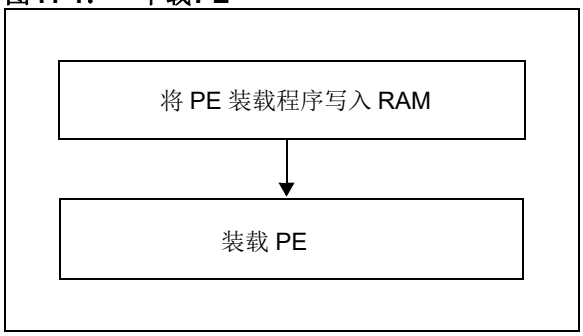
- 读存储器
- 擦除存储器
- 程序存储器
- 空白检查
- 读取执行程序固件版本
- 获取闪存存储单元的循环冗余校验（Cyclic Redundancy Check，CRC）

PE执行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17.2节“PE命令集”**提供了每条命令的详细说明。

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。

将PE装入数据RAM中之后，可以使用表17-1中所示的命令集对PIC32系列进行编程。

图11-1： 下载PE



将PE装入存储器是一个分为两步的过程：

1. 将PE装载程序装入数据RAM。（PE装载程序将PE二进制文件装入数据RAM的适当位置，在完成后跳转到编程执行程序，并开始执行它。）
2. 将PE二进制代码馈送到PE装载程序。

表11-1列出了下载PE所需的步骤。

表11-1： 下载PE操作码

操作	操作数

步骤1： 仅限PIC32MX器件：将BMXCON初始化为0x1F0040。PIC32内核执行的指令序列为：
`lui a0,0xbf88`
`ori a0,a0,0x2000 /* address of BMXCON */`
`lui a1,0x1f`
`ori a1,a1,0x40 /* a1 has 0x1f0040 */`
`sw a1,0(a0) /* BMXCON initialized */`	

XferInstruction	0x3c04bf88
XferInstruction	0x34842000
XferInstruction	0x3c05001f
XferInstruction	0x34a50040
XferInstruction	0xac850000

步骤2： 仅限PIC32MX器件：将BMXDKPBA初始化为0x800。PIC32内核执行的指令序列为：
`li a1,0x800`
`sw a1,16(a0)`	

XferInstruction	0x34050800
XferInstruction	0xac850010

步骤3： 仅限PIC32MX器件：将BMXDUDBA和BMXDUPBA初始化为BMXDRMSZ的值。PIC32内核执行的指令序列为：
`lw a1,64(a0) /* load BMXDMSZ */`
`sw a1,32(a0)`
`sw a1,48(a0)`	

XferInstruction	0x8C850040
XferInstruction	0xac850020
XferInstruction	0xac850030

步骤4： 设置PE的PIC32 RAM地址。PIC32内核执行的指令序列为：
`lui a0,0xa000`
`ori a0,a0,0x800`	

XferInstruction	0x3c04a000
XferInstruction	0x34840800

步骤5： 装载PE_Loader。重复此步骤（步骤5），直到整个PE_Loader都装入PIC32存储器中。在操作数字段中，“<PE_loader hi++>”表示表11-2中所示的PE装载程序操作码的MSb 31到16。同样，“<PE_loader lo++>”表示表11-2中所示的PE装载程序操作码的LSb 15到0。符号“++”表示当连续执行这些操作时，将从表11-2中所示的LPE装载程序的操作码列表中传输新字。PIC32内核执行的指令序列为：
`lui a2, <PE_loader hi++>`
`ori a2,a2, <PE_loader lo++>`
`sw a2,0(a0)`
`addiu a0,a0,4`	

XferInstruction	(0x3c06 <PE_loader hi++>)

表11-1： 下载PE操作码（续）

操作	操作数
XferInstruction	(0x34c6 <PE_loader lo++>)
XferInstruction	0xac860000
XferInstruction	0x24840004
步骤6： 跳转到PE_Loader。PIC32内核执行的指令序列为： lui t9,0xa000 ori t9,t9,0x800 jr t9 nop	
XferInstruction	0x3c19a000
XferInstruction	0x37390800
XferInstruction	0x03200008
XferInstruction	0x00000000
步骤7： 使用 PE_Loader 装载 PE。重复执行此步骤（第 7 步）的最后一个指令，直到整个PE 都装入PIC32 存储器中。在此步骤中，将提供一个Intel® Hex格式的PE 文件，您每次需要解析多个32位字并将其传输到PIC32 存储器中（见**附录B：“Hex文件格式”**）。PIC32执行的指令序列如表11-2中所示。	
SendCommand	ETAP_FASTDATA
XferFastData	PE_ADDRESS (Address of PE program block from PE Hex file)
XferFastData	PE_SIZE (Number of 32-bit words of the program block from PE Hex file)
XferFastData	PE software op code from PE Hex file (PE Instructions)
步骤8： 跳转到PE。幻数（0xDEAD0000）指示PE_Loader，PE 已完全装入存储器中。当PE_Loader 发现幻数时，将跳转到PE。	
XferFastData	0x00000000
XferFastData	0xDEAD0000

表11-2： PE装载程序操作码

操作码	指令
0x3c07dead	lui a3, 0xdead
0x3c06ff20	lui a2, 0xff20
0x3c05ff20	lui a1, 0xff20
	here1:
0x8cc40000	lw a0, 0 (a2)
0x8cc30000	lw v1, 0 (a2)
0x1067000b	beq v1, a3, <here3>
0x00000000	nop
0x1060fffb	beqz v1, <here1>
0x00000000	nop
	here2:
0x8ca20000	lw v0, 0 (a1)
0x2463ffff	addiu v1, v1, -1
0xac820000	sw v0, 0 (a0)

表11-2： PE装载程序操作码

操作码	指令
0x24840004	addiu a0, a0, 4
0x1460fffb	bnez v1, <here2>
0x00000000	nop
0x1000fff3	b <here1>
0x00000000	nop
	here3:
0x3c02a000	lui v0, 0xa000
0x34420900	ori v0, v0, 0x900
0x00400008	jr v0
0x00000000	nop

© 2014-2020 Microchip Technology Inc.

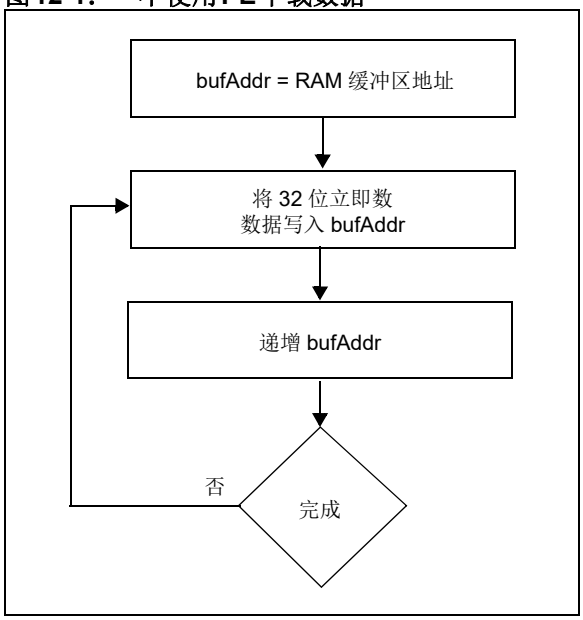
12.0 下载数据块

要将数据块编程到PIC32器件中，必须将其装入SRAM中。

12.1 不使用PE

要在不使用PE的情况下对存储块进行编程，必须先将数据块写入RAM。此方法要求编程器传输带有嵌入（立即）数据的实际机器指令，以将数据块写入器件的内部RAM存储器。

图12-1： 不使用PE下载数据



下载数据块需要执行以下步骤：

1. `XferInstruction` (op code)。
2. 重复第1步，直到最后一条指令传输到CPU中。

表12-1： 下载数据操作码

操作码	指令
步骤1： 将SRAM基址初始化为0xA0000000。	
3c10a000	lui s0, 0xA000;
步骤2： 将要编程的整行数据写入系统SRAM。	

3c08<DATA>
3508<DATA>
ae08<OFFSET>
	lui t0, <DATA(31:16)>;
	ori t0, t0, <DATA(15:0)>;
	sw t0, <OFFSET>(s0);
	// OFFSET increments by 4

步骤3： 重复第2步，直到装入一行数据。	

12.2 使用PE

使用PE时，将通过两条命令处理**第12.0节“下载数据块”**和**第14.0节“启动闪存行写入”**中的步骤：`ROW_PROGRAM`和`PROGRAM`。

`ROW_PROGRAM`命令可对单行闪存数据进行编程，`PROGRAM`命令则可以对多行闪存数据进行编程。这两条命令在**第17.0节“编程执行程序”**中有详细说明。

13.0 启动页擦除

可擦除单个页，而非擦除全部闪存。这种情况下不使用PE。

PIC32MK系列器件可以通过提高用于执行擦除的内部电压来对页执行擦除重试。

表13-1： 页擦除操作码

操作码	指令
步骤1： 所有PIC32器件：初始化常量。将寄存器a1、a2和a3分别设置为WREN = 1或NVMOP<3:0> = 0100、WR = 1且WREN = 1。寄存器s1和s2设置为解锁数据值，且s0初始化为0。	
34054004	ori a1, $0,0x4004
34068000	ori a2,$0,0x8000
34074000	ori a3,$0,0x4000
3c11aa99	lui s1,0xaa99
36316655	ori s1,s1,0x6655
3c125566	lui s2,0x5566
365299aa	ori s2,s2,0x99aa
3c100000	lui s0,0x0000
步骤2： 仅限PIC32MX系列器件：将寄存器a0设置为NVM寄存器的基址（0xBF80_F400）。	
3c04bf80	lui a0,0xbf80
3484f400	ori a0,a0,0xf400
步骤3： 仅限PIC32MK和PIC32MZ系列器件：将寄存器a0设置为NVM寄存器的基址（0xBF80_0600）。寄存器s3设置为用于禁止NVMBPB中写保护的值。	
3c04b480	lui a0,0xbf80
34840600	ori a0,a0,0x0600
34138080	ori s3,$0,0x8080
步骤4： 仅限PIC32MK和PIC32MZ系列器件：解锁并禁止引导闪存写保护。	
ac910010	sw s1,16(a0)
ac920010	sw s2,16(a0)
ac930090	sw s3,144(a0)
00000000	nop
步骤5： 仅限PIC32MK系列器件：保存NVMCON2的内容。	
8c9400a0	lw s4,160(a0)
步骤6： 仅限PIC32MK系列器件：设置初始编程电压并使能页测试（需要解锁）。	
36953000	ori s5,s4,0x3000
32b5fcff	andi s5,s5,0xFCFF
	here3:
ac910010	sw s1,16(a0)
ac920010	sw s2,16(a0)
ac860008	sw a2,8(a0)
ac9500a0	sw s5,160(a0)

© 2014-2020 Microchip Technology Inc.

表13-1： 页擦除操作码（续）

操作码	指令
步骤7： 所有PIC32器件：使用要擦除的闪存页的地址设置NVMADDR寄存器。	
3c08<ADDR>	lui t0,<FLASH_PAGE_ADDR(31:16)>
3508<ADDR>	ori t0,t0,<FLASH_PAGE_ADDR(15:0)>
ac880020	sw t0,32(a0)
步骤8： 所有PIC32器件：设置NVMCON寄存器进行写保护。	
ac850000	sw a1,0(a0)
	delay (6 us)
步骤9： 仅限PIC32MX器件：轮询LVDSTAT寄存器。	
	here1:
8c880000	lw t0,0(a0)
31080800	andi t0,t0,0x0800
1500fffd	bne t0,$0,here1
00000000	nop
步骤10： 所有PIC32器件：解锁NVMCON寄存器并启动写操作。	
ac910010	sw s1,16(a0)
ac920010	sw s2,16(a0)
ac860008	sw a2,8(a0)
步骤11： 所有PIC32器件：循环执行直到WR位（NVMCON<15>）清零为止。	
	here2:
8c880000	lw t0,0(a0)
01064024	and t0,t0,a2
1500fffd	bne t0,$0,here2
00000000	nop
步骤12： 所有PIC32器件：WR位（NVMCON<15>）清零后至少等待500 ns，然后再写入任意NVM寄存器。这需要在执行过程中插入延时。编程工具和编程执行程序使用FRC 8 MHz时钟。因此，四条NOP指令等于500 ns（见**注1**）。	
00000000	nop
00000000	nop
00000000	nop
00000000	nop
步骤13： 所有PIC32器件：清零WREN位（NVMCON<14>）。	
ac870004	sw a3,4(a0)

表13-1： 页擦除操作码（续）

操作码	指令
步骤14： 仅限 PIC32MK 系列器件：检查页面中的所有数据是否均已擦除。如果未全部擦除，请调整电压，然后重试。如果尝试了所有电压，则表示擦除失败，此时将转至错误程序。	
ac870004	sw a3, 4(a0)
20171000	addi s7, $0, 4096
00005020	add t2, $0, $0
8c880020	lw t0, 32(a0)
01194020	add t0, t0, t9
	here5:
8d090000	lw t1, 0(t0)
15200005	bne t1, $0, here6
214a0010	addi t2, t2, 16
11570009	beq t2, s7, here7
00000000	nop
1000fffa	beq $0, $0, here5
21080010	addi t0, t0, 16
	here6:
22b50100	addi s5, s5, 256
32b60300	andi s6, s5, 768
16c0ffde	bne s6, $0, here3
00000000	nop
10000005	beq $0, $0, err_proc
00000000	nop
步骤15： 仅限PIC32MK系列器件：恢复NVMCON2寄存器。	
	here7:
ac9400a0	sw s4,160(a0)
步骤16： 所有PIC32器件：检查WRERR位（NVMCON<13>），确保程序序列已成功完成。如果发生错误，则转至错误处理程序。	
8c880000	lw t0,0(a0)
30082000	andi t0,t0,0x2000
1500<ERR_PROC>	bne t0,$0,<err_proc_offset>
00000000	nop

注 1： 要在运行时在用户应用程序中对闪存进行编程，建议使用以下代码：

```
while(NVMCON.WR) // waitfor WR bit(NVMCON<15>) to clear
{};
{
unsigned int start_count = _CP0_GET_COUNT();
unsigned int total_count = (.00000025 * SYSCLK); //count for 500 ns and CPU
frequency in MHz
while ((_CP0_GET_COUNT()- start_count) < total_count);
}
```


© 2014-2020 Microchip Technology Inc.

14.0 启动闪存行写入

注： 某些PIC32器件具有可用的ECC存储器。使用ECC功能时，必须使用四个32位字对齐的方式以四个32位字为一组来编程闪存。如果以动态方式使用ECC，则由编程方法确定何时使用该功能。对于使用单字编程命令编程的字，未使能ECC。对于以四字为一组（使用四字或行编程命令）来编程的字，使能ECC。如果未遵循上述方法，则可能会在运行时导致ECC DED错误。有关ECC使用和配置的详细信息，请参见具体器件的数据手册。

一旦将一行数据下载到器件的SRAM中，就必须启动编程序列来将数据块写入闪存中。

有关启动闪存行写入的操作码和指令，请参见表14-1。

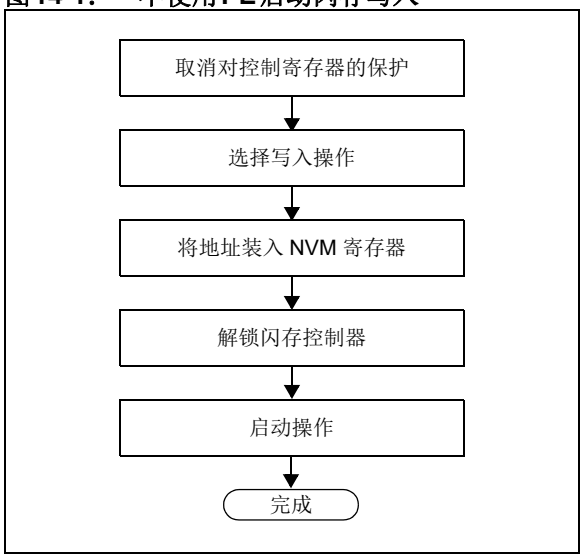
14.1 使用PE

使用PE时，数据会立即从SRAM写入闪存中。不需要进一步的操作。

14.2 不使用PE

对闪存执行的写入操作是通过NVMCON寄存器控制的。编程的执行过程如下：设置NVMCON选择写入操作的类型，并通过将WR控制位（NVMCON<15>）置1启动编程序列。

图14-1： 不使用PE启动闪存写入



在闪存写入过程（见表14-1）中，使用行编程方法对闪存进行编程，因为这通常是最便捷的方法。此外，还提供了双字和四字编程方法，根据您的应用，可能需要使用此方法。有关更多信息，请参见具体器件数据手册中的**“闪存程序存储器”**章节和《PIC32系列参考手册》的相关章节。

启动闪存写入需要执行以下步骤：

1. `XferInstruction` (op code)。
2. 重复第1步，直到最后一条指令传输到CPU中。

表14-1： 启动闪存行写入操作码

操作码	指令
步骤1： 所有PIC32器件：初始化常量。将寄存器a1、a2和a3分别设置为WREN = 1或NVMOP<3:0> = 0011、WR = 1且WREN = 1。寄存器s1和s2设置为解锁数据值，且s0初始化为0。	
34054003	ori a1,$0,0x4003
34068000	ori a2,$0,0x8000
34074000	ori a3,$0,0x4000
3c11aa99	lui s1,0xaa99
36316655	ori s1,s1,0x6655
3c125566	lui s2,0x5566
365299aa	ori s2,s2,0x99aa
3c100000	lui s0,0x0000
步骤2： 仅限PIC32MX器件：将寄存器a0设置为NVM寄存器的基址（0xBF80_F400）。	
3c04bf80	lui a0,0xbf80
3484f400	ori a0,a0,0xf400
步骤2： 仅限PIC32MK和PIC32MZ系列器件：将寄存器a0设置为NVM寄存器的基址（0xBF80_0600）。寄存器s3设置为用于禁止NVMBPB中写保护的值。	
3c04bf80	lui a0,0xbf80
34840600	ori a0,a0,0x0600
34138080	ori s3,$0,0x8080
步骤3： 仅限PIC32MK和PIC32MZ系列器件：解锁并禁止引导闪存写保护。	
ac910010	sw s1,16(a0)
ac920010	sw s2,16(a0)
ac930090	sw s3,144(a0)
00000000	nop
步骤4： 所有PIC32器件：使用要编程的闪存行的地址设置NVMADDR寄存器。	
3c08<ADDR>	lui t0,<FLASH_ROW_ADDR(31:16)>
3508<ADDR>	ori t0,t0,<FLASH_ROW_ADDR(15:0)>
ac880020	sw t0,32(a0)
步骤5： 仅限PIC32MX器件：使用物理源SRAM地址（偏移为64）设置NVMSRCADDR寄存器。	
3c10<ADDR>	lui s0, <RAM_ADDR(31:16)>
3610<ADDR>	ori s0,s0,<RAM_ADDR(15:0)>
ac900040	sw s0,64(a0)
步骤5： 仅限PIC32MK和PIC32MZ系列器件：使用物理源SRAM地址（偏移为112）设置NVMSRCADDR寄存器。	
3c10<ADDR>	lui s0, <RAM_ADDR(31:16)>
3610<ADDR>	ori s0,s0,<RAM_ADDR(15:0)>
ac900070	sw s0,112(a0)
步骤6： 所有PIC32器件：设置NVMCON寄存器进行写保护。	
ac850000	sw a1,0(a0)
	delay (6 µs)

© 2014-2020 Microchip Technology Inc.

表14-1： 启动闪存行写入操作码（续）

操作码	指令
步骤7： 仅限PIC32MX器件：轮询LVDSTAT寄存器。	
	here1:
8C880000	lw t0,0(a0)
31080800	andi t0,t0,0x0800
1500fffd	bne t0,$0,here1
00000000	nop
步骤8： 所有PIC32器件：解锁NVMCON寄存器并启动写操作。	
ac910010	sw s1,16(a0)
ac920010	sw s2,16(a0)
ac860008	sw a2,8(a0)
步骤9： 所有PIC32器件：循环执行直到WR位（NVMCON<15>）清零为止。	
	here2:
8c880000	lw t0,0(a0)
01064024	and t0,t0,a2
1500fffd	bne t0,$0,here2
00000000	nop
步骤10： 所有PIC32器件：WR位（NVMCON<15>）清零后至少等待500 ns，然后再写入任意NVM寄存器。这需要在执行过程中插入延时。编程工具和编程执行程序使用FRC 8 MHz时钟。因此，四条NOP指令等于500 ns（见**注1**）。	
00000000	nop
00000000	nop
00000000	nop
00000000	nop
步骤11： 所有PIC32器件：清零WREN位（NVMCONM<14>）。	
ac870004	sw a3,4(a0)
步骤12： 所有PIC32器件：检查WRERR位（NVMCON<13>），以确保程序序列已成功完成。如果发生错误，则转至错误处理程序。	
8c880000	lw t0,0(a0)
30082000	andi t0,zero,0x2000
1500<ERR_PROC>	bne t0, $0, <err_proc_offset>
00000000	nop

注 1： 要在运行时在用户应用程序中对闪存进行编程，建议使用以下代码：

```
while(NVMCON.WR)  //Wait for WR bit (NVMCON<15>) to clear
{};
{
unsigned int start_count = _CP0_GET_COUNT();
unsigned int total_count = (.00000025 * SYSCLK); //count for 500 ns and CPU
frequency in MHz
while ((_CP0_GET_COUNT()- start_count) < total_count);
}
```

15.0 校验器件存储器

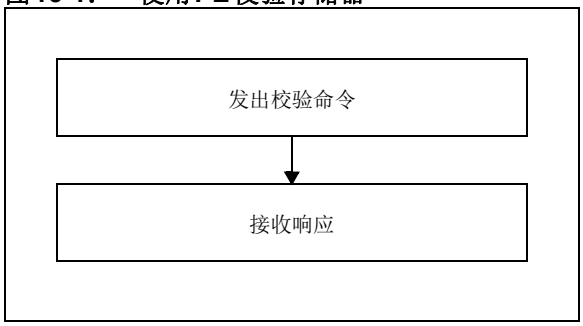
校验的步骤涉及读回代码存储空间并将读到的值与存储在编程器缓冲区中的副本作比较。使用其余代码校验配置寄存器。

注： 由于配置寄存器包含器件代码保护位，因此应在写入后立即对代码存储区进行校验（如果使能代码保护）。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。

15.1 使用PE校验存储器

使用GET_CRC命令执行存储器校验。表17-2列出了操作码和指令。

图15-1： 使用PE校验存储器



使用PE校验存储器需要执行以下步骤：

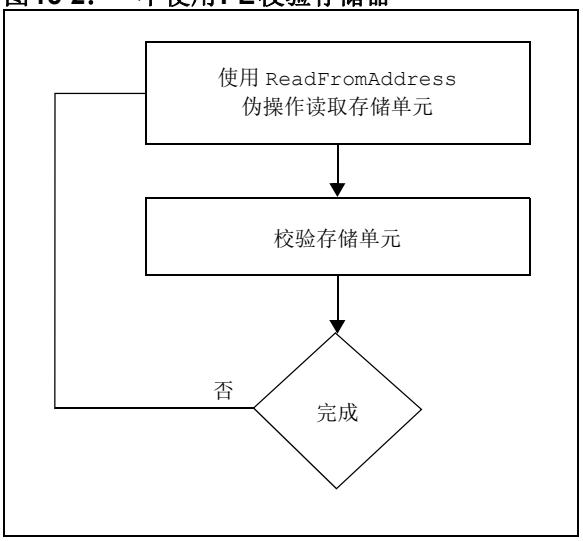
1. XferFastData (GET_CRC)。
2. XferFastData (start_Address)。
3. XferFastData (length)。
4. valCkSum = XferFastData (32'h0x00)。

校验valCkSum是否与编程器缓冲区中保存的副本的校验和相匹配。

15.2 不使用PE校验存储器

读取闪存是通过从Fastdata寄存器执行一系列读访问来执行的。表20-4显示了EJTAG编程详细信息，包括用于执行处理器访问操作的地址和操作码数据。

图15-2： 不使用PE校验存储器



校验存储器需要执行以下步骤：

1. XferInstruction (op code)。
2. 重复第1步，直到最后一条指令传输到CPU中。
3. 校验valRead是否与编程器缓冲区中保存的副本相匹配。
4. 对每个存储单元重复第1-3步。

表15-1： 校验器件操作码

操作码	指令
步骤1： 初始化一些常量。	
3c13ff20	lui s3, 0xFF20
步骤2： 读取存储单元。	
3c08<ADDR> 3508<ADDR>	lui t0,<FLASH_WORD_ADDR(31:16)> ori t0, t0, <FLASH_WORD_ADDR(15:0)>
步骤3： 写入Fastdata存储单元。	
8d090000 ae690000	lw t1, 0(t0) sw t1, 0(s3)
步骤4： 从Fastdata寄存器0xFF200000读取数据。	
步骤5： 重复第2-4步，直到读取完所有配置存储单元。	

© 2014-2020 Microchip Technology Inc.

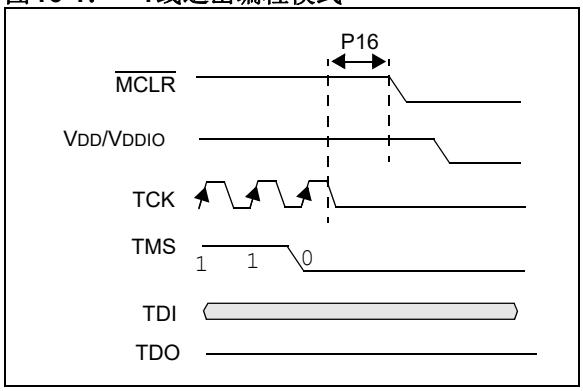
16.0 退出编程模式

对器件进行编程后，必须使器件退出编程模式，以开始正确执行其新的程序存储器内容。

16.1 4线接口

通过将VIH从$\overline{\text{MCLR}}$引脚移除，可退出编程模式，如图16-1所示。退出操作的惟一要求是在最后一个时钟和编程信号之后的P16时间间隔后移除VIH。

图16-1： 4线退出编程模式



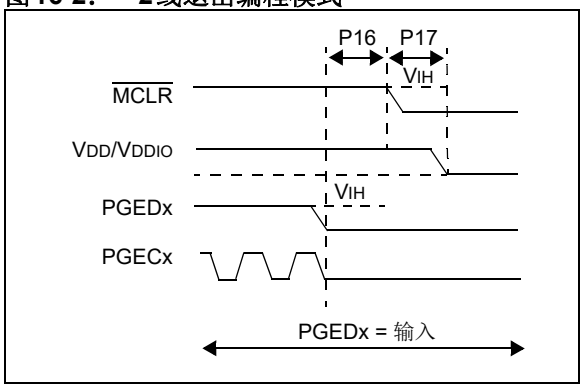
退出编程模式需要执行以下步骤：

1. SetMode (5'b11111)。
2. 将$\overline{\text{MCLR}}$引脚置为有效。
3. 断开电源（如果器件已通电）。

16.2 2线接口

通过将VIH从$\overline{\text{MCLR}}$引脚移除，可退出编程模式，如图16-2所示。退出操作的惟一要求是在PGECx和PGEDx引脚上最后一个时钟和编程信号之后的P16时间间隔后移除VIH。

图16-2： 2线退出编程模式



退出编程模式需要执行以下步骤：

1. SetMode (5'b11111)。
2. 将$\overline{\text{MCLR}}$引脚置为有效。
3. 在PGECx上发出时钟脉冲。
4. 断开电源（如果器件已通电）。

17.0 编程执行程序

注： 编程执行程序（PE）随附于您安装的MPLAB X IDE中。要下载器件对应的PE文件，请访问 Microchip 网站（www.microchip.com）上的相关产品页面。

17.1 PE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。PE一次只能接收一个命令。相应地，在接收并处理命令后，PE将向编程器发送一个响应。

17.1.1 2线ICSP EJTAG速率

在增强型ICSP模式下，PIC32系列器件使用内部快速RC振荡器工作，其标称频率为8 MHz。

17.1.2 通信概述

编程器和PE使用EJTAG地址、数据和Fastdata寄存器进行通信。尤其是，编程器使用Fastdata寄存器将命令和数据传输到PE。编程器使用地址和数据寄存器接收来自PE的响应。接收响应的伪操作如下面的GetPEResponse伪操作所示：

格式：

```
response = GetPEResponse()
```

目的：

使编程器从PE接收32位响应值。

示例17-1： GetPEResponse示例

```
WORD GetPEResponse()
{
        WORD response;

// Wait until CPU is ready
SendCommand(ETAP_CONTROL);

// Check if Proc. Access bit (bit 18) is set
do {
      controlVal=XferData(32'h0x0004C000 );
} while( PrAcc(contorlVal<18>) is not '1' );

// Select Data Register
SendCommand(ETAP_DATA);

// Receive Response
response = XferData(0);

// Tell CPU to execute instruction
SendCommand(ETAP_CONTROL);
XferData(32'h0x0000C000);

// return 32-bit response
return response;
}
```

编程器和PE之间的典型通信序列如表17-1中所示。

序列在编程器将命令和可选附加数据发送到PE时开始，然后PE会执行命令。

PE完成命令执行时，会将响应发送回编程器。

响应可以有多个。例如，如果编程器发送READ命令，响应将包含读取的数据。

表17-1： PE的通信序列

操作	操作数
步骤1： 将命令和可选数据从编程器发送到PE。	
XferFastData	(Command | data len)
XferFastData..	optional data..
步骤2： 编程器读取来自PE的响应。	
GetPEResponse	response
GetPEResponse...	response...

© 2014-2020 Microchip Technology Inc.

17.2 PE命令集

表17-2提供了PE命令集的详细信息，如每条命令的操作码、助记符和简短说明。**第17.2.3节“ROW_PROGRAM命令”**到**第17.2.14节“CHANGE_CFG命令”**详细说明了每条命令的功能。

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，其中包含所有必需的响应数据或错误数据。

17.2.1 命令格式

所有PE命令都具有由32位头和命令所需的所有数据组成的通用格式（见图17-1）。32位头由用于标识命令的16位操作码字段以及16位命令操作数字段组成。操作数字段的使用因命令而异。

注： 有些命令没有操作数信息；但是，必须发送操作数字段，且编程执行程序将忽略该数据。

图17-1： 命令格式

操作码字段中的命令必须与表17-2中所列命令集中的一个命令相匹配。如果接收到的命令与列表中的任何命令都不匹配，将返回NACK响应，如表17-3中所示。

PE使用命令操作数字段来确定要读取或写入的字节数。如果该字段的值不正确，PE将无法正确地接收命令。

表17-2： PE命令集

操作码	助记符	说明
0x0	ROW_PROGRAM(1)	对指定地址处的一行闪存进行编程。
0x1	READ	从指定地址开始读取存储器的N个32位字（N < 65,536）。
0x2	PROGRAM	从指定地址开始对闪存进行编程。
0x3	WORD_PROGRAM(3)	对指定地址处的一行闪存进行编程。
0x4	CHIP_ERASE	对整个芯片进行芯片擦除。
0x5	PAGE_ERASE	从指定地址擦除代码存储区页。
0x6	BLANK_CHECK	代码空白检查。
0x7	EXEC_VERSION	读取PE软件版本。
0x8	GET_CRC	获取闪存的CRC。
0x9	PROGRAM_CLUSTER	将指定的字节数编程到指定地址。
0xA	GET_DEVICEID	返回器件的硬件ID。
0xB	CHANGE_CFG(2)	由探针用来设置PE的各种配置设置。
0xC	GET_CHECKSUM	获取闪存的校验和。
0xD	QUAD_WORD_PGRM(4)	对指定地址处闪存的4个字进行编程。

注 1： 有关每个器件的行大小，请参见表5-1。

2： 该命令在PIC32MX1XX/2XX器件中不可用。

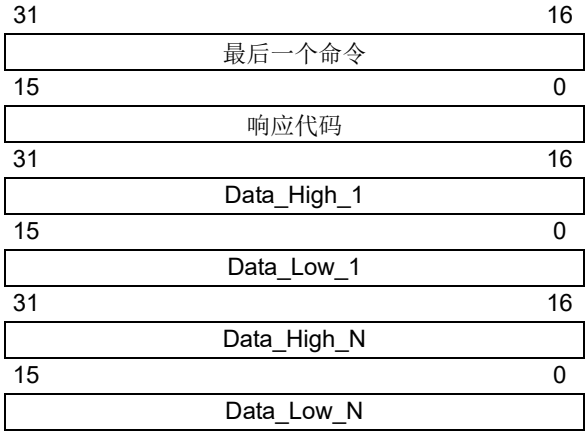
3： 对于具有ECC功能的PIC32MZ系列器件，WORD_PROGRAM命令不会产生ECC奇偶校验位。在使能了ECC的条件下读取使用WORD_PROGRAM命令编程的存储单元会导致DED故障。

4： 该命令仅在PIC32MK和PIC32MZ系列器件上可用。

17.2.2 响应格式

PE响应集如表17-3中所示。所有PE响应都具有由32位头和响应所需的所有数据组成的通用格式（见图17-2）。

图17-2： 响应格式



17.2.2.1 Last_Cmd字段

Last_Cmd是响应的第一个字中的一个16位字段，该字段指示PE处理的命令。它可以用来校验PE是否正确地接收了编程器发送的命令。

17.2.2.2 响应码

响应码指示最后一个命令是成功还是失败，或者命令是否为无法识别的值。响应码的值如表17-3中所示。

表17-3： 响应值

操作码	助记符	说明
0x0	PASS	命令已成功处理
0x2	FAIL	命令未成功处理
0x3	NACK	命令未知

17.2.2.3 可选数据

对于某些命令（如读取），响应头后面可能会跟随可选数据。可选数据的32位字数量取决于最后一个命令操作及其参数。

17.2.3 ROW_PROGRAM命令

ROW_PROGRAM命令可指示PE对指定地址处的一行数据进行编程。

要编程到存储器中的数据位于命令字Data_1至Data_N中，必须使用表17-4给出的指令字打包格式对它们进行打包（此命令需要整行数据）。

图17-3： ROW_PROGRAM命令

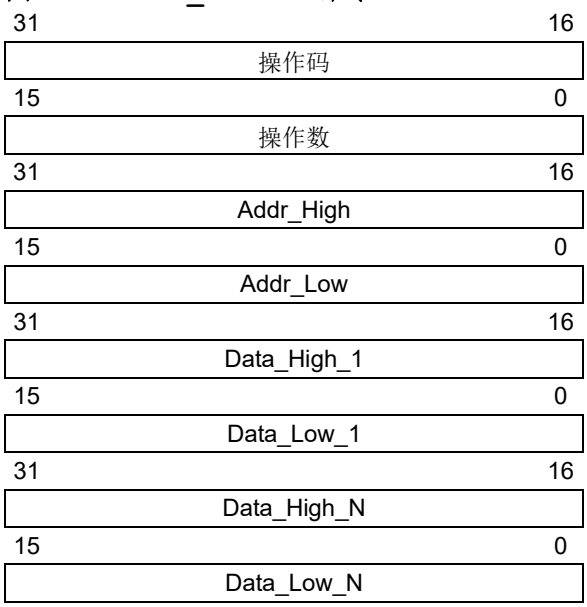
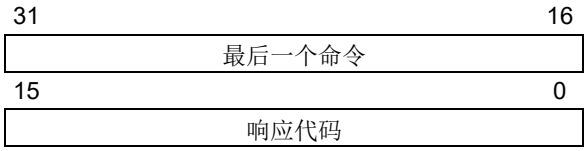


表17-4： ROW_PROGRAM格式

字段	说明
操作码	0x0
操作数	未使用
Addr_High	32位目标地址的高16位
Addr_Low	32位目标地址的低16位
Data_High_1	高16位数据字1
Data_Low_1	低16位数据字1
Data_High_N	高16位数据字2至N
Data_Low_N	低16位数据字2至N

预期的响应（单字）：

图17-4： ROW_PROGRAM响应



© 2014-2020 Microchip Technology Inc.

17.2.4 READ命令

READ命令指示PE从存储器中读取。操作数字段中指定的32位字数从Addr_Low和Addr_High字段指定的32位地址开始。此命令可用于读取闪存和配置字。作为对该命令的响应返回的所有数据都使用表17-5中所述的数据打包格式。

图17-5： READ命令

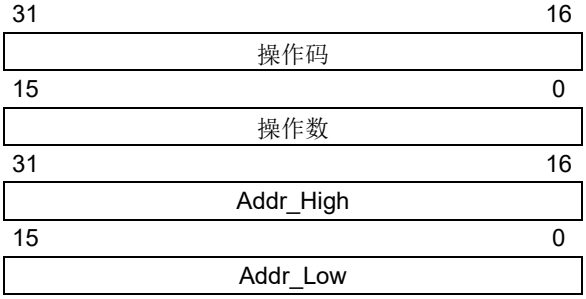


表17-5： READ格式

字段	说明
操作码	0x1
操作数	要读取的32位字的数量（N）（最大为65,535）
Addr_Low	32位源地址的低16位
Addr_High	32位源地址的高16位

预期响应：

图17-6： READ响应

注： 读取未实现的存储器将导致PE复位。确保只访问特定器件上存在的存储单元。

17.2.5 PROGRAM命令

PROGRAM命令指示PE从Addr_Low和Addr_High字段指定的32位地址开始对闪存（包括配置字）进行编程。一个32位长度字段指定要编程的字节数。

该地址必须与闪存行大小边界对齐，且长度必须是闪存行大小的倍数。有关要编程器件的合适行大小，请参见表5-1。

图17-7： PROGRAM命令

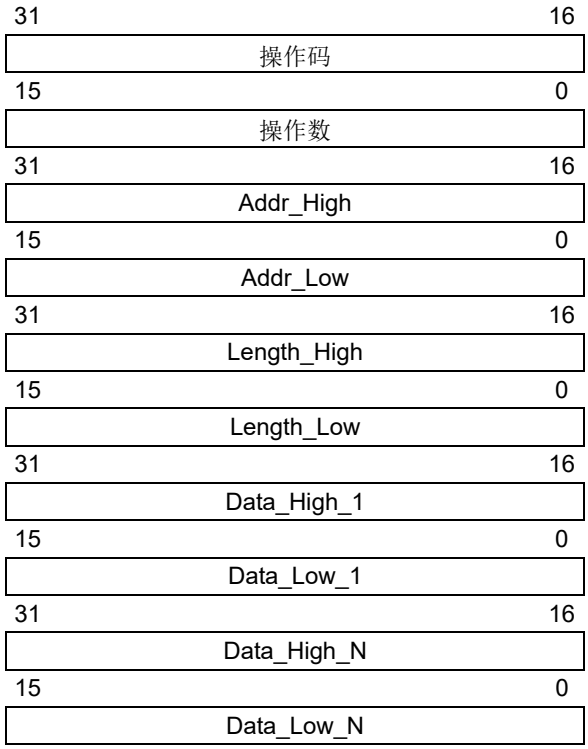


表17-6： PROGRAM格式

字段	说明
操作码	0x2
操作数	未使用
Addr_Low	32位目标地址的低16位
Addr_High	32位目标地址的高16位
Length_Low	长度的低16位
Length_High	长度的高16位
Data_Low_N	低16位数据字2至N
Data_High_N	高16位数据字2至N

以下是三个编程情形：

- 要编程的数据长度等于单个闪存行的大小
- 要编程的数据长度等于两个闪存行的大小
- 要编程的数据长度大于两个闪存行的大小

当数据长度等于512字节时，PE从探针接收512字节的数据块，并立即将此命令的响应发送回探针。

PE将对其接收的每行数据做出响应。如果命令的数据长度等于单行，将生成单个PE响应。如果数据长度等于两行，PE将等待，以接收这两行数据，然后为每个数据行发送连续响应。如果数据长度大于两行，PE将在接收前两行数据后发送第一行的响应。接收后续数据行包后将发送后续响应。响应将比数据滞后一行。接收最后一行数据时，PE将发出倒数第二个数据行的连续响应，后面跟随最后一行的响应。

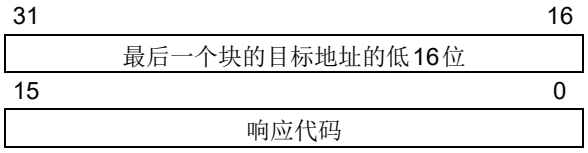
如果PE在任何块的编程中遇到错误，它将向探针发送故障状态，并中止 PROGRAM 命令。接收到故障状态时，探针必须停止发送数据。PE将不会处理此命令来自探针的任何其他数据。该过程如图17-9中所示。

注： 如果PROGRAM命令失败，则编程器应使用READ命令从闪存中读取失败的行。然后，编程器应将从闪存接收的行与其本地副本逐字比较，以确定闪存编程失败的地址。

此命令的响应与其他命令的响应稍有不同。响应的16个MSb包含目标地址的16个LSb，其中最后一个块已编程。这有助于探针和PE在发送和接收数据及响应时保持正确同步。

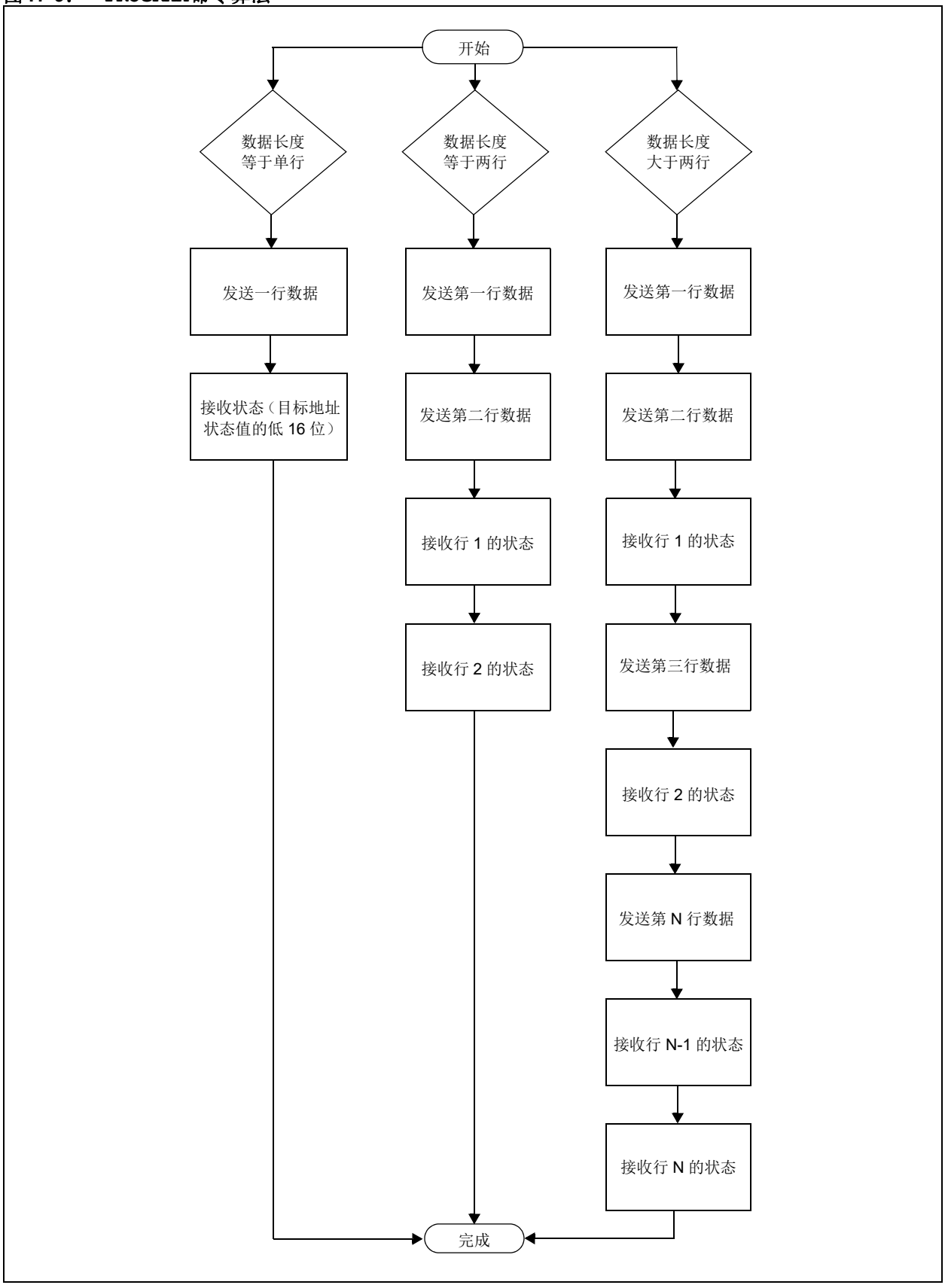
预期的响应（单字）：

图17-8： PROGRAM响应



© 2014-2020 Microchip Technology Inc.

图17-9： PROGRAM命令算法



17.2.6 WORD_PROGRAM命令

WORD_PROGRAM命令可指示PE对指定地址处的一个32位字数据进行编程。

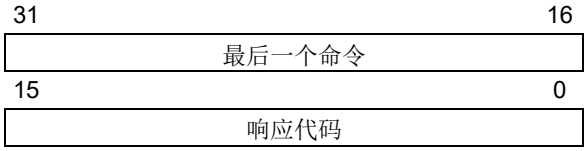
图17-10： WORD_PROGRAM命令

表17-7： WORD_PROGRAM格式

字段	说明
操作码	0x3
操作数	未使用
Addr_High	32位目标地址的高16位
Addr_Low	32位目标地址的低16位
Data_High	高16位数据字
Data_Low	低16位数据字

预期的响应（单字）：

图17-11： WORD_PROGRAM响应



17.2.7 CHIP_ERASE命令

CHIP_ERASE命令擦除整个芯片，包括配置块。在执行擦除后，整个闪存将包含0xFFFFFFFF。

图17-12： CHIP_ERASE命令

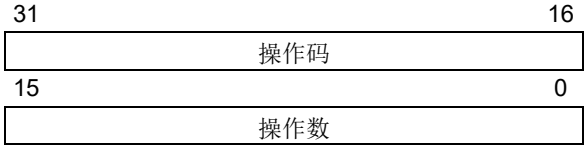
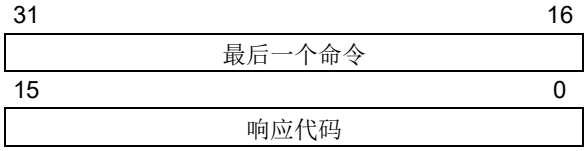


表17-8： CHIP_ERASE格式

字段	说明
操作码	0x4
操作数	未使用
Addr_Low	32位目标地址的低16位
Addr_High	32位目标地址的高16位

预期的响应（单字）：

图17-13： CHIP_ERASE响应



© 2014-2020 Microchip Technology Inc.

17.2.8 PAGE_ERASE命令

PAGE_ERASE命令从指定基址擦除代码存储区的指定页数。指定的基址必须为0x400或0x100的倍数，具体取决于器件。

在执行擦除后，代码存储区的所有目标字都将包含0xFFFFFFFF。

图17-14： PAGE_ERASE命令

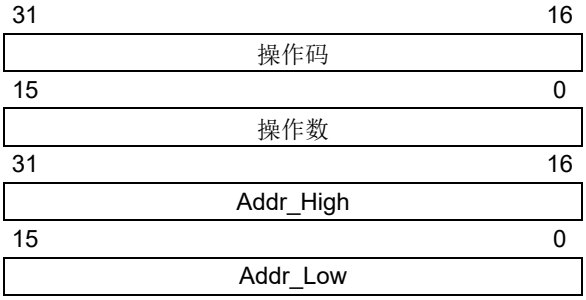
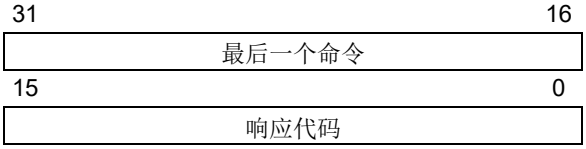


表17-9： PAGE_ERASE格式

字段	说明
操作码	0x5
操作数	要擦除的页数
Addr_Low	32位目标地址的低16位
Addr_High	32位目标地址的高16位

预期的响应（单字）：

图17-15： PAGE_ERASE响应



17.2.9 BLANK_CHECK命令

BLANK_CHECK命令查询PE，以确定代码存储区和代码保护配置位（GCP和GWRP）的内容是否为空白（包含全1）。

图17-16： BLANK_CHECK命令

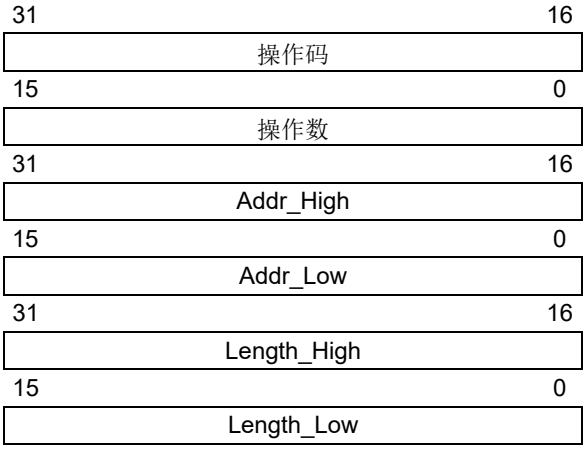
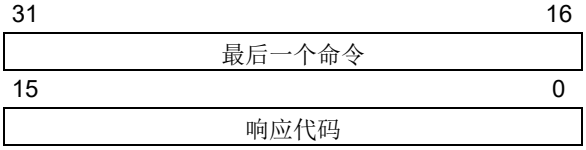


表17-10： BLANK_CHECK格式

字段	说明
操作码	0x6
操作数	未使用
地址	空白检查的起始地址
长度	要检查的程序存储单元数（以字节为单位）

预期的响应（空白器件为单字）：

图17-17： BLANK_CHECK响应



17.2.10 EXEC_VERSION命令

EXEC_VERSION查询RAM中存储的PE软件的版本。

图17-18： EXEC_VERSION命令

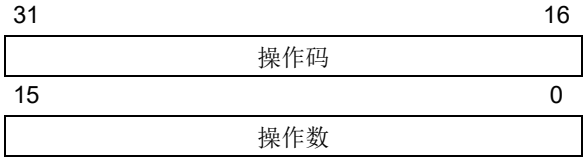
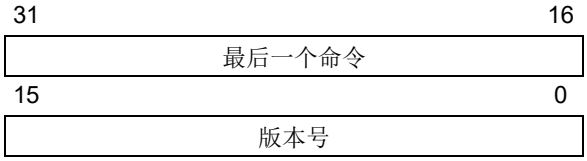


表17-11： EXEC_VERSION格式

字段	说明
操作码	0x7
操作数	未使用

预期的响应（单字）：

图17-19： EXEC_VERSION响应



17.2.11 GET_CRC命令

GET_CRC使用查表方法计算从指定地址开始指定长度的缓冲区的CRC。CRC详细信息如下：

- CRC-CCITT，16位
- 多项式：X^16+X^12+X^5+1，十六进制0x00011021
- 种子：0xFFFF
- 最高有效字节（Most Significant Byte，MSB）先移入

注 1： 在响应中，只有CRC的最低16位是有效的。

2： PE将自动确定硬件CRC是否可用并默认使用它。PIC32MX1XX/2XX器件不使用硬件CRC。

图17-20： GET_CRC命令

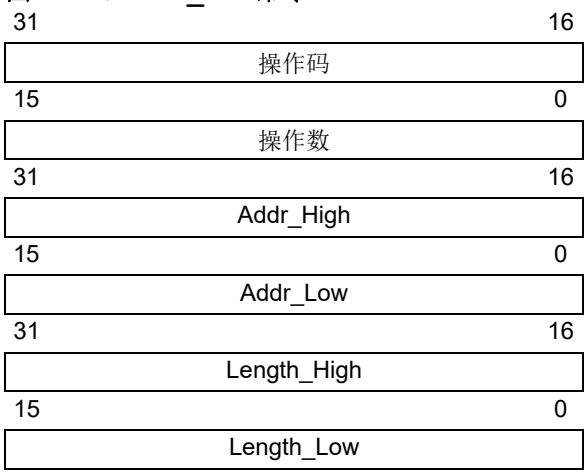


表17-12： GET_CRC格式

字段	说明
操作码	0x8
操作数	未使用
地址	计算CRC的起始地址
长度	进行CRC计算的缓冲区的长度（以字节数为单位）

预期的响应（双字）：

图17-21： GET_CRC响应

© 2014-2020 Microchip Technology Inc.

17.2.12 PROGRAM_CLUSTER命令

PROGRAM_CLUSTER将指定的字节数编程到指定地址。该地址必须按32位对齐，且字节数必须是32位字的倍数。

图17-22： PROGRAM_CLUSTER命令

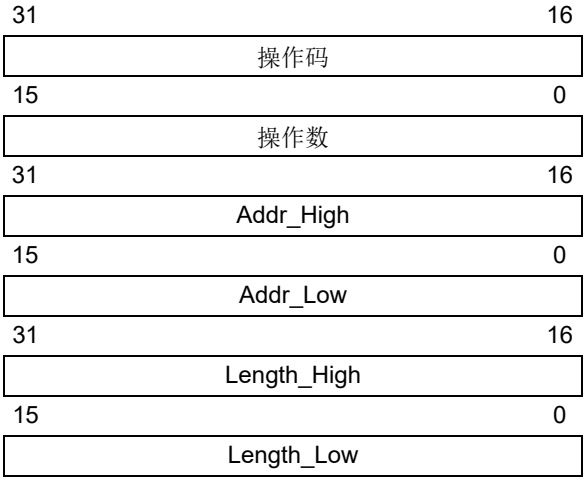


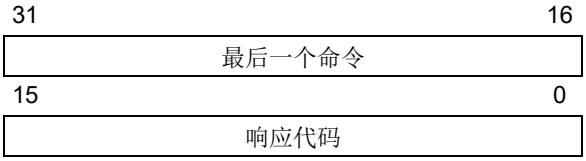
表17-13： PROGRAM_CLUSTER格式

字段	说明
操作码	0x9
操作数	未使用
地址	编程的起始地址
长度	要编程的存储区长度（以字节数为单位）

注： 如果PROGRAM_CLUSTER命令失败，则编程器应使用READ命令从闪存中读取失败的行。然后，编程器应将从闪存接收的行与其本地副本逐字比较，以确定闪存编程失败的地址。

预期的响应（单字）：

图17-23： PROGRAM_CLUSTER响应



17.2.13 GET_DEVICEID命令

GET_DEVICEID命令返回器件的硬件ID。

图17-24： GET_DEVICEID命令

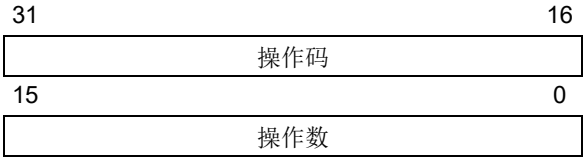


表17-14： GET_DEVICEID格式

字段	说明
操作码	0xA
操作数	未使用

预期的响应（单字）：

图17-25： GET_DEVICEID响应

17.2.14 CHANGE_CFG命令

CHANGE_CFG由探针用来设置PE的各种配置设置。目前，单个配置设置确定PE应使用以下哪种计算方法：

- 软件CRC计算方法
- 硬件计算方法

图17-26： CHANGE_CFG命令

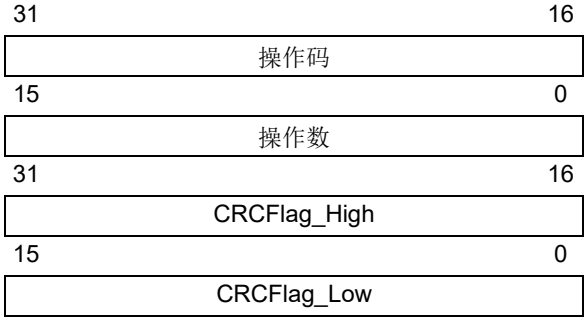
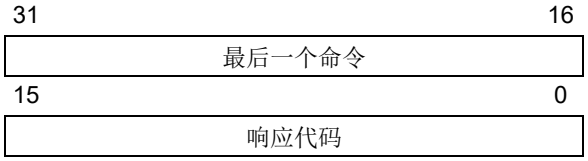


表17-15： CHANGE_CFG格式

字段	说明
操作码	0xB
操作数	未使用
CRCFlag	如果值为0，则PE使用软件CRC计算方法。 如果值为1，则PE使用硬件CRC单元来计算CRC。

预期的响应（单字）：

图17-27： CHANGE_CFG响应



注： CHANGE_CFG命令在PIC32MX1XX/2XX器件中不可用。

17.2.15 GET_CHECKSUM命令

GET_CHECKSUM返回所有字节的总和，从地址参数开始一直到长度参数。结果为一个32位字。

图17-28： CHANGE_CFG命令

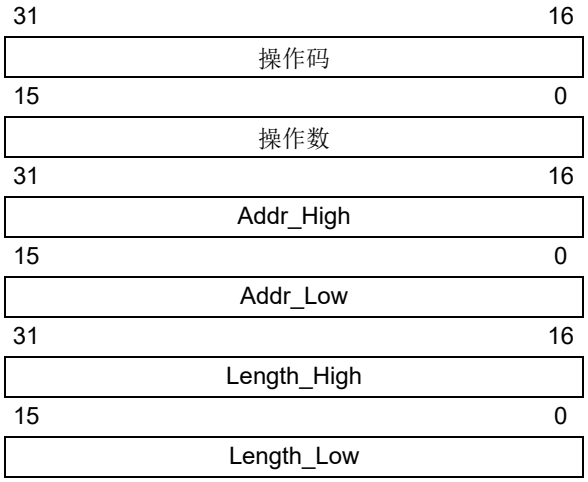


表17-16： GET_CHECKSUM格式

字段	说明
操作码	0x0C
操作数	未使用
Addr_High	用于计算校验和的数据的32位起始地址的高16位。
Addr_Low	用于计算校验和的数据的32位起始地址的低16位。
Length_High	用于计算校验和的数据的32位长度的高16位（以字节为单位）。
Length_Low	用于计算校验和的数据的32位长度的低16位（以字节为单位）。

预期的响应（单字）：

图17-29： GET_CHECKSUM响应

© 2014-2020 Microchip Technology Inc.

17.2.16 QUAD_WORD_PROGRAM命令

QUAD_WORD_PROGRAM指示PE对指定地址处的四个32位字进行编程。该地址必须是一个对齐的四字边界（bit 0-1必须为0）。如果不是，该命令将返回FAIL响应值，且不对任何数据进行编程。

图17-30： QUAD_WORD_PROGRAM命令

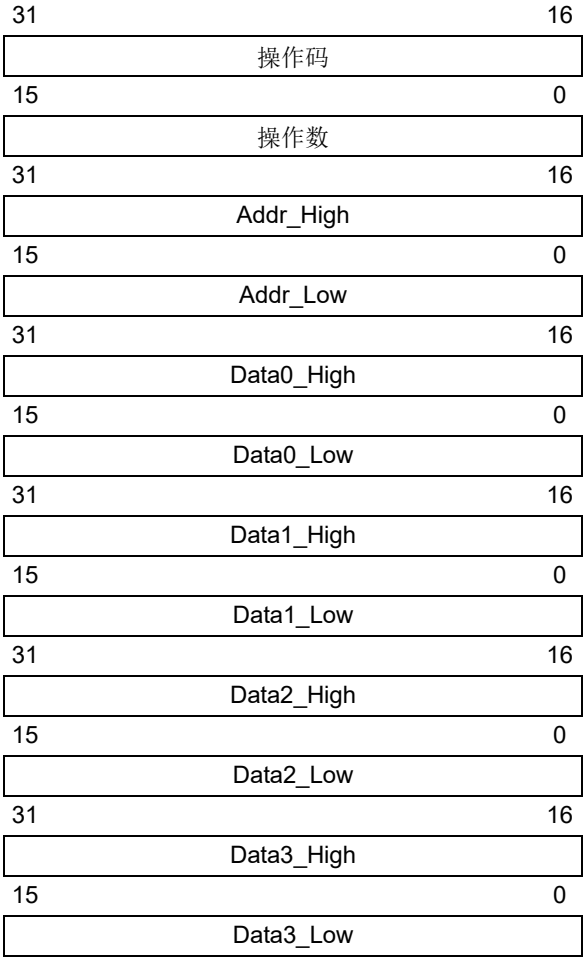


表17-17： QUAD_WORD_PROGRAM格式

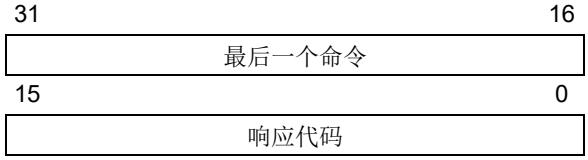
字段	说明
操作码	0x0D
操作数	未使用
Addr_High	32位起始地址的高16位。
Addr_Low	32位起始地址的低16位。
Data0_High	数据字0的高16位。
Data0_Low	数据字0的低16位。
Data1_High	数据字1的高16位。
Data1_Low	数据字1的低16位。

表17-17： QUAD_WORD_PROGRAM格式

Data2_High	数据字2的高16位。
Data2_Low	数据字2的低16位。
Data3_High	数据字3的高16位。
Data3_Low	数据字3的低16位。

预期的响应（单字）：

图17-31： QUAD_WORD_PROGRAM响应



18.0 校验和

18.1 原理

校验和的计算方式为：对程序闪存、引导闪存（除器件配置字外）、带适用掩码的器件ID寄存器和带适用掩码的器件配置字中的所有字节（8位量）进行32位求和。然后，计算该和的二进制补码。这个最终的32位数就是校验和。

注： PIC32MKXXXXGPK/MCM/GPG/MCJXXX器件使用CRC32校验和。有关CRC32校验和的更多信息，请参见文档“Readme for MPLABX IDE.htm”的“校验和更改”一章，此文档可在<MPLABX安装路径>\ <MPLABX版本>\docs下找到。

18.2 掩码值

器件配置的掩码值是通过将所有未实现位设置为0，并将所有已实现位设置为1计算得出。

例如，寄存器18-1显示了PIC32MX360F512L器件的DEVCFG0寄存器。此寄存器的掩码值为：

mask_value_devcfg0 = 0x110FF00B

寄存器18-1： PIC32MX360F512L的DEVCFG0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-0	r-1	r-1	R/P-1	r-1	r-1	r-1	R/P-1
	—	—	—	CP	—	—	—	BWP
23:16	r-1	r-1	r-1	r-1	R/P-1	R/P-1	R/P-1	R/P-1
	—	—	—	—	PWP19	PWP18	PWP17	PWP16
15:8	R/P-1	R/P-1	R/P-1	R/P-1	r-1	r-1	r-1	r-1
	PWP15	PWP14	PWP13	PWP12	—	—	—	—
7:0	r-1	r-1	r-1	r-1	R/P-1	r-1	R/P-1	R/P-1
	—	—	—	—	ICESEL	—	DEBUG<1:0>	

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

© 2014-2020 Microchip Technology Inc.

表18-1列出了要在器件校验和计算中使用的器件配置寄存器和器件ID寄存器的掩码值（针对PIC32MX、PIC32MZ和PIC32MKXXXXGPD/GPE/MCFXXX器件）。PIC32MKXXXXGPK/MCM/GPG/MCJXXX 器件使用CRC32校验和。有关CRC32校验和的更多信息，请参见文档“Readme for MPLABX IDE.htm”的“校验和更改”一章，此文档可在<MPLABX安装路径>\<MPLABX版本>\docs下找到。

表18-1： 器件配置寄存器的掩码值（目前支持的PIC32MX、PIC32MZ和PIC32MKXXXXGPD/GPE/MCFXXX器件）

器件系列	闪存大小（KB）	DEVCFG0	DEVCFG1	DEVCFG2	DEVCFG3	DEVCFG4	DEVID
PIC32MX110/120/130/150F0xx PIC32MX150F128 （仅限28/36/44引脚器件）	16、32、64和128	0x1100FC1F	0x03DFF7A7	0x00070077	0xF000FFFF	—	0x0FFFFFFF
PIC32MX130F128/256 PIC32MX150F256 （仅限28/36/44引脚器件）	16、32、64和128	0x1100FC1F	0x03DFF7A7	0x00070077	0xF0000000	—	0x0FFFFFFF
PIC32MX210/220/230/250（仅限28/36/44引脚器件）	16、32、64和128	0x1100FC1F	0x03DFF7A7	0x00078777	0xF0000000	—	0x0FFFFFFF
PIC32MX15X/17X（仅限28/44引脚器件）	128和256	0x1187F01F	0x03FFF7A7	0xFFB700F7	0x30C00000	—	0x0FFFFFFF
PIC32MX25X/27X（仅限28/44引脚器件）	128和256	0x1187F01F	0x03FFF7A7	0xFFB787F7	0x70C00000	—	0x0FFFFFFF
PIC32MX320/340/360	32、64、128、256和512	0x110FF00B	0x009FF7A7	0x00070077	0x0000FFFF	—	0x000FF000
PIC32MX420/440/460	32、64、128、256和512	0x110FF00B	0x009FF7A7	0x00078777	0x0000FFFF	—	0x000FF000
PIC32MX110/120/130/150F0xx PIC32MX150F128 PIC32MX170F256 （仅限64/100引脚器件）	64、128、256和512	0x110FFC1F	0x03DFF7A7	0x00070077	0xF000FFFF	—	0x0FFFFFFF
PIC32MX130F128/256 PIC32MX150F256 PIC32MX170F512 （仅限64/100引脚器件）	64、128、256和512	0x110FFC1F	0x03DFF7A7	0x00070077	0xF0000000	—	0x0FFFFFFF
PIC32MX230F0xx PIC32MX250F128 PIC32MX270F256 （仅限64/100引脚器件）	64、128、256和512	0x110FFC1F	0x03DFF7A7	0x00078777	0xF000FFFF	—	0x0FFFFFFF

注 1： 仅适用于PIC32MZ DA系列器件。

表18-1： 器件配置寄存器的掩码值（目前支持的PIC32MX、PIC32MZ和PIC32MKXXXXGPD/GPE/MCFXXX器件）（续）

器件系列	闪存大小（KB）	DEVCFG0	DEVCFG1	DEVCFG2	DEVCFG3	DEVCFG4	DEVID
PIC32MX230F128 PIC32MX230F256 PIC32MX250F256 PIC32MX270F512 PIC32MX530 PIC32MX550 PIC32MX570 （仅限64/100引脚器件）	64、128、256和512	0x110FFC1F	0x03DFF7A7	0x00078777	0xF0000000	—	0x0FFFFFFF
PIC32MX330/350/370	64、128、256和512	0x110FF01F	0x03DFF7A7	0x00070077	0x3007FFFF	—	0x0FFFFFFF
PIC32MX430/450/470	64、128、256和512	0x110FF01F	0x03DFF7A7	0x00078777	0xF007FFFF	—	0x0FFFFFFF
PIC32MX534/564	64和128	0x110FF00F	0x009FF7A7	0x00078777	0xC407FFFF	—	0x0FFFF000
PIC32MX664	64和128	0x110FF00F	0x009FF7A7	0x00078777	0xC307FFFF	—	0x0FFFF000
PIC32MK0512/1024XXD/E/F	512和1024	0x7FFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFF0000	—	0x0FFFFFFF
PIC32MX764	128	0x110FF00F	0x009FF7A7	0x00078777	0xC707FFFF	—	0x0FFFF000
PIC32MX170F256（仅限28/36/44引脚器件）	256	0x1107FC1F	0x03DFF7A7	0x00070077	0xF000FFFF	—	0x0FFFFFFF
PIC32MX170F512（仅限28/36/44引脚器件）	256	0x1107FC1F	0x03DFF7A7	0x00070077	0xF0000000	—	0x0FFFFFFF
PIC32MX270F256（仅限28/36/44引脚器件）	256	0x1107FC1F	0x03DFF7A7	0x00078777	0xF000FFFF	—	0x0FFFFFFF
PIC32MX270F512（仅限28/36/44引脚器件）	256	0x1107FC1F	0x03DFF7A7	0x00078777	0xF0000000	—	0x0FFFFFFF
PIC32MZ05XX/10XX/20XX	512、1024和2048	0x7FFFFFFF	0xFFFFFFFF	0xFFFFFFFF	0xFFFF0000	0xFFFFFFFF（**注1**）	0x0FFFFFFF
PIC32MX575	256和512	0x110FF00F	0x009FF7A7	0x00078777	0xC407FFFF	—	0x000FF000
PIC32MX675/695	256和512	0x110FF00F	0x009FF7A7	0x00078777	0xC307FFFF	—	0x000FF000
PIC32MX775/795	256和512	0x110FF00F	0x009FF7A7	0x00078777	0xC707FFFF	—	0x000FF000

注 1： 仅适用于PIC32MZ DA系列器件。

© 2014-2020 Microchip Technology Inc.

18.3 算法

图18-1给出了计算PIC32器件校验和的高级算法的示例，说明了一种得出校验和的方法。这仅仅是一个说明如何实现实际计算的示例，最终使用的方法由软件开发人员自行决定。

如前所述，PIC32校验和的计算方式为：对程序闪存、引导闪存（除器件配置字外）、带适用掩码的器件ID寄存器和带适用掩码的器件配置字中的所有字节（8位量）进行32位求和。

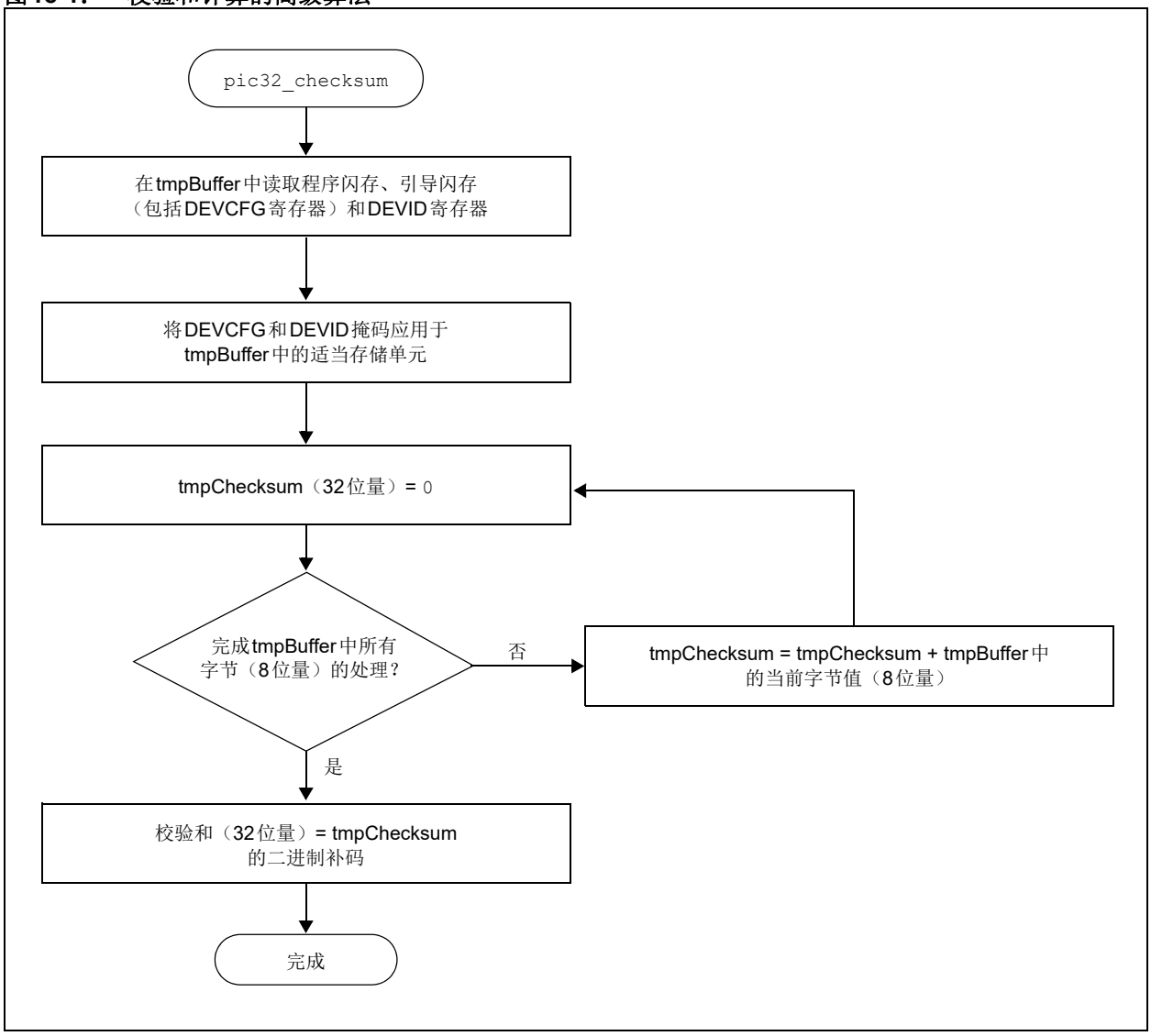
然后，计算该和的二进制补码。这个最终的32位数就是校验和。

器件配置和器件ID寄存器的掩码值的计算方法如前面的**第18.2节“掩码值”**中所述。

在将器件配置寄存器值的字节加到校验和之前，需要使用适当掩码值来执行这些寄存器值的算术与运算。

同样，在将器件ID寄存器的字节加到校验和之前，需要使用适当掩码值来执行该寄存器的算术与运算，更多信息请参见**第19.0节“配置存储器和器件ID”**。

图18-1： 校验和计算的高级算法



公式18-1给出了用于计算PIC32器件校验和的公式。

公式18-1： 校验和公式

$$校验和 = 二进制补码\ (PF + BF + DCR + DIR)$$

其中，

PF = 程序闪存中所有字节的32位求和

BF = 引导闪存（器件配置寄存器除外）中所有字节的32位求和（见**注1**）

$$DCR = \sum_{X=0}^{y} 字节的32位求和(MASK_{DEVCFGX}\ \&\ DEVCFGx)$$

其中，

y = 3（针对PIC32MX、PIC32MKXXXXGPD/GPE/MCFXXX、PIC32MZ EC和PIC32MZ EF系列器件）

y = 4（针对所有其他PIC32MZ系列器件）

$$DIR = 字节的32位求和(MASK_{DEVID}\ \&\ DEVID)$$

$MASK_{DEVCFGX}$ = 表18-1中的掩码值

$MASK_{DEVID}$ = 表18-1中的掩码值（**注2**）

$DEVCP$ = 字节的32位求和（$MASK_{DEVCP}$和$DEVCP$）

其中，

$MASK_{DEVCP}$ = 0x10000000（针对PIC32MKXXXXGPD/GPE/MCFXXX）

注 1： 对于PIC32MZ系列器件，不对0x1FCxFF00到0x1FCxFFFF范围内的引导闪存求和，因为这些存储单元包含器件配置和CP值。对于PIC32MKXXXXGPD/GPE/MCFXXX系列器件，不对0x1FC03F00至0x1FC03FFF范围内的引导闪存求和。

2： 对于PIC32MZ和PIC32MKXXXXGPD/GPE/MCFXXX系列器件，在MPLAB X IDE中计算出的校验和仅使用主DEVCFGx寄存器。不会对备用或第二个引导闪存（如果有）寄存器进行计算。

© 2014-2020 Microchip Technology Inc.

18.4 校验和计算示例

以下五节说明了如何使用公式18-1计算PIC32MX360F512L器件的校验和。

对此校验和计算示例进行了以下假定：

- 程序闪存和引导闪存处于已擦除状态（所有字节为0xFF）
- 器件配置处于器件的默认状态（未进行任何配置更改）

需要单独计算公式右侧的每一项（PF、BF、DCR和DIR）。得出值后，就可以计算校验和的最终值。

18.4.1 计算校验和公式中的“PF”

程序闪存的大小为512 KB，相当于524288字节。由于假定程序闪存处于已擦除状态，因此PF的值可通过以下计算求得：

PF = 0xFF + 0xFF + … 524288次

PF = 0x7F80000（32位数）

18.4.2 计算校验和公式中的“BF”

引导闪存的大小为12 KB，相当于12288字节。但是，最后16字节是器件配置寄存器，需单独处理。因此，我们在此步骤中考虑的引导闪存中的字节数为12272。由于假定引导闪存处于已擦除状态，因此“BF”的值可通过以下计算求得：

BF = 0xFF + 0xFF + … 12272次

BF = 0x002FC010（32位数）

18.4.3 计算校验和公式中的“DCR”

由于器件配置寄存器处于其默认状态，因此通过相应DEVCFG寄存器的值（如PIC32内核所读取）、其各自的掩码值、应用掩码得出的值和字节的32位求和（全部如表18-2中所示）可以得到字节的32位求和的总和。

从表18-2可知，“DCR”值为：

DCR = 0x000003D6（32位数）

表18-2： DCR计算示例

寄存器	POR默认值	掩码	POR默认值和掩码	字节的32位求和
DEVCFG0	0x7FFFFFFF	0x110FF00B	0x110FF00B	0x0000011B
DEVCFG1	0xFFFFFFFF	0x009FF7A7	0x009FF7A7	0x0000023D
DEVCFG2	0xFFFFFFFF	0x00070077	0x00070077	0x0000007E
DEVCFG3	0xFFFFFFFF	0x00000000	0x00000000	0x00000000
			字节的32位求和的总和 =	**0x000003D6**

18.4.4 计算校验和公式中的“DIR”

器件ID寄存器的值、其掩码值、应用掩码得出的值和字节的32位求和如表18-3中所示。

从表18-3可知，“DIR”值为：

DIR = 0x00000083（32位数）

表18-3： DIR计算示例

寄存器	POR默认值	掩码	POR默认值和掩码	字节的32位求和
DEVID	0x00938053	0x000FF000	0x00038000	0x00000083

18.4.5 完成PIC32校验和计算

在前面几节中得出的值（PF、BF、DCR和DIR）用于计算校验和值。执行前面几节中得出的PF、BF、DCR和DIR的32位求和，然后将其存储在一个称为*temp*的变量中，如示例18-1中所示。

示例18-1： 校验和计算过程

1. 首先，*temp* = PF + BF + DCR + DIR，即：
   temp = 0x7F80000 + 0x002FC010 + 0x000003D6 + 0x00000083
2. 将全部四个值相加可得到*temp*等于0x0827C469
3. 最后，*temp*的二进制补码为校验和：
   校验和 =（*temp*）的二进制补码，即校验和 =（（*temp*）的二进制反码）+ 1，得到0xF7D83B97

18.4.6 器件处于代码保护状态时的校验和值

由于器件配置字在PIC32器件处于代码保护状态时不可读取，因此对于所有器件而言校验和值均为零。

© 2014-2020 Microchip Technology Inc.

19.0 配置存储器和器件ID

PIC32器件具有多个功能，旨在最大程度提高应用灵活性和可靠性，并通过消除外部元件最大限度降低成本。这些功能可通过每个器件的特定配置位进行配置。

有关可用功能、配置位和器件ID寄存器的完整列表，请参见特定器件数据手册中的**“特殊功能”**章节。

有关特定PIC32MX、PIC32MZ或PIC32MK系列器件的器件ID，请参见**附录C：“器件ID”**。

有关特定器件的最新芯片版本和版本ID，请参见相关的“系列芯片勘误表和数据手册澄清”。这些文档可从Microchip网站下载：访问http://www.microchip.com/PIC32，然后导航至：*Documentation* > *Errata*（文档 > 勘误表）。

19.1 器件配置

在PIC32器件中，配置字可在执行任何代码之前选择在器件复位时设置的各种器件配置。这些值位于引导闪存（BFM）的高地址存储单元，且由于它们是程序存储器的一部分，因此将与可执行代码和程序常量一起包含在编程文件中。表19-1至表19-4中列出了这些配置字的名称和存储单元。

此外，对于分别具有双引导和双分区闪存的PIC32MZ和PIC32MK系列器件，表19-3和表19-4还包含用于此类器件的配置字。有关双引导区域的详细说明，请参见《PIC32系列参考手册》中的**第48章“存储器构成和权限”**（DS60001214）。

表19-1： DEVCFG存储单元（仅针对15X/17X/25X/27X和PIC32MX3XX/4XX/5XX/6XX/7XX器件）

配置字寄存器	物理地址
DEVCFG0	0x1FC02FFC
DEVCFG1	0x1FC02FF8
DEVCFG2	0x1FC02FF4
DEVCFG3	0x1FC02FF0

表19-2： DEVCFG存储单元（仅针对28/36/44引脚PIC32MX1XX/2XX和64/100引脚PIC32MX1XX/2XX/5XX器件）

配置字寄存器	物理地址
DEVCFG0	0x1FC00BFC
DEVCFG1	0x1FC00BF8
DEVCFG2	0x1FC00BF4
DEVCFG3	0x1FC00BF0

发生上电复位（Power-on Reset，POR）或任何复位时，会将配置字从引导闪存复制到其对应的配置寄存器。配置位只能编程为0（未编程状态 = 1）。

在编程期间，配置字最多可以编程两次（针对PIC32MX器件）和一次（针对PIC32MZ和PIC32MK器件），之后就必须执行页擦除操作。

在对配置字进行编程后，器件复位将导致向配置寄存器中装载新值。因此，编程器应在器件校验之前对配置字进行编程。最后一个步骤是对代码保护配置字进行编程。

这些配置字可确定振荡器源。如果使用2线增强型ICSP模式将忽略配置字，且器件将总是使用FRC；但是，在4线模式中，情况并非如此。如果在复位后器件中并不存在的配置字选择了振荡器源，则编程器将无法在器件复位后对器件执行闪存操作。有关使用配置字选择振荡器的详细信息，请参见特定器件数据手册中的**“特殊功能”**章节。

表19-3： PIC32MZ系列器件的配置字存储单元

配置字寄存器（见注1）	寄存器物理地址			
	固定引导区域1	固定引导区域2	活动引导别名区域（见注2）	非活动引导别名区域（见注2）
引导序列号	0x1FC4FFF0	0x1FC6FFF0	0x1FC0FFF0	0x1FC2FFF0
代码保护	0x1FC4FFD0	0x1FC6FFD0	0x1FC0FFD0	0x1FC2FFD0
DEVCFG0	0x1FC4FFCC	0x1FC6FFCC	0x1FC0FFCC	0x1FC2FFCC
DEVCFG1	0x1FC4FFC8	0x1FC6FFC8	0x1FC0FFC8	0x1FC2FFC8
DEVCFG2	0x1FC4FFC4	0x1FC6FFC4	0x1FC0FFC4	0x1FC2FFC4
DEVCFG3	0x1FC4FFC0	0x1FC6FFC0	0x1FC0FFC0	0x1FC2FFC0
DEVCFG4（见**注3**）	0x1FC4FFBC	0x1FC6FFBC	0x1FC0FFBC	0x1FC2FFBC
备用引导序列号	0x1FC4FF70	0x1FC6FF70	0x1FC0FF70	0x1FC2FF70
备用代码保护	0x1FC4FF50	0x1FC6FF50	0x1FC0FF50	0x1FC2FF50
备用DEVCFG0	0x1FC4FF4C	0x1FC6FF4C	0x1FC0FF4C	0x1FC2FF4C
备用DEVCFG1	0x1FC4FF48	0x1FC6FF48	0x1FC0FF48	0x1FC2FF48
备用DEVCFG2	0x1FC4FF44	0x1FC6FF44	0x1FC0FF44	0x1FC2FF44
备用DEVCFG3	0x1FC4FF40	0x1FC6FF40	0x1FC0FF40	0x1FC2FF40
备用DEVCFG4（见**注3**）	0x1FC4FF3C	0x1FC6FF3C	0x1FC0FF3C	0x1FC2FF3C

注 1： 0x1FCxFF00-0x1FCxFFFF 存储器区域中的所有值都应使用 QUAD_WORD_PROGRAM 命令进行编程，以确保正确的ECC配置。有关详细信息，请参见**第17.2.16节“QUAD_WORD_PROGRAM命令”**。

2： 对于固定区域1有效而固定区域2无效的未编程器件，假设存在活动/非活动引导别名选择。有关双引导区域的详细说明，请参见《PIC32系列参考手册》中的**第48章“存储器构成和权限”**（DS60001214A_CN）。

3： 这些配置字仅在PIC32MZ DA系列器件中可用。

表19-4： PIC32MKXXXXXXD/E/FXX系列器件的配置字存储单元

配置字寄存器（见注1）	寄存器物理地址			
	固定引导区域1	固定引导区域2	活动引导别名区域（见注2）	非活动引导别名区域（见注2）
引导序列号	0x1FC43FF0	0x1FC63FF0	0x1FC03FF0	0x1FC23FF0
代码保护	0x1FC43FD0	0x1FC63FD0	0x1FC03FD0	0x1FC23FD0
DEVCFG0	0x1FC43FCC	0x1FC63FCC	0x1FC03FCC	0x1FC23FCC
DEVCFG1	0x1FC43FC8	0x1FC63FC8	0x1FC03FC8	0x1FC23FC8
DEVCFG2	0x1FC43FC4	0x1FC63FC4	0x1FC03FC4	0x1FC23FC4
DEVCFG3	0x1FC43FC0	0x1FC63FC0	0x1FC03FC0	0x1FC23FC0

注 1： 如果器件具有ECC存储器，则应使用QUAD_WORD_PROGRAM命令对以下各配置字组进行编程：

- 引导序列号（单个四字编程操作）
- 代码保护（单个四字编程操作）
- DEVCFG3、DEVCFG2、DEVCFG1和DEVCFG0（单个四字编程操作）

2： 对于固定区域1有效而固定区域2无效的未编程器件，假设存在活动/非活动引导别名选择。关于这些功能的详细说明，请参见《PIC32系列参考手册》的**第48章“存储器构成和权限”**（DS60001214A_CN）。

© 2014-2020 Microchip Technology Inc.

表19-5： PIC32MKXXXXXXH/G/J/K/L/MXX系列器件的配置字存储单元

配置字寄存器（见注1）	寄存器物理地址			
	固定引导区域1	固定引导区域2	活动引导别名区域（见注2和注3）	非活动引导别名区域（见注2和注3）
引导序列号	0x1FC43FF0	0x1FC63FF0	0x1FC03FF0	0x1FC23FF0
代码保护	0x1FC43FD0	0x1FC63FD0	0x1FC03FD0	0x1FC23FD0
DEVCFG0	0x1FC43FCC	0x1FC63FCC	0x1FC03FCC	0x1FC23FCC
DEVCFG1	0x1FC43FC8	0x1FC63FC8	0x1FC03FC8	0x1FC23FC8
DEVCFG2	0x1FC43FC4	0x1FC63FC4	0x1FC03FC4	0x1FC23FC4
DEVCFG3	0x1FC43FC0	0x1FC63FC0	0x1FC03FC0	0x1FC23FC0
DEVCFG4	0x1FC43FBC			
备用引导序列号	0x1FC43F70	0x1FC63F70	0x1FC03F70	0x1FC23F70
备用代码保护	0x1FC43F50	0x1FC63F50	0x1FC03F50	0x1FC23F50
备用DEVCFG0	0x1FC43F4C	0x1FC63F4C	0x1FC03F4C	0x1FC23F4C
备用DEVCFG1	0x1FC43F48	0x1FC63F48	0x1FC03F48	0x1FC23F48
备用DEVCFG2	0x1FC43F44	0x1FC63F44	0x1FC03F44	0x1FC23F44
备用DEVCFG3	0x1FC43F40	0x1FC63F40	0x1FC03F40	0x1FC23F40
备用DEVCFG4	0x1FC43F3C	0x1FC63F3C	0x1FC03F3C	0x1FC23F3C

注 1： 如果器件具有ECC存储器，则应使用QUAD_WORD_PROGRAM命令对以下各配置字组进行编程：

- 引导序列号（单个四字编程操作）
- 代码保护（单个四字编程操作）
- DEVCFG3、DEVCFG2、DEVCFG1和DEVCFG0（单个四字编程操作）

2： 0x1FCxFF00-0x1FCxFFFF存储器区域中的所有值都应使用QUAD_WORD_PROGRAM命令进行编程，以确保正确的ECC配置。有关详细信息，请参见**第17.2.16节“QUAD_WORD_PROGRAM命令”**。

3： 对于固定区域1有效而固定区域2无效的未编程器件，假设存在活动/非活动引导别名选择。有关双引导区域的详细说明，请参见**第48章“存储器构成和权限”**（DS60001214A_CN）。

19.1.1 配置寄存器保护

为了避免在代码执行期间配置被意外更改，所有的可编程配置位只可被写入一次。在上电周期内对一个位进行初始编程之后就不能再次写该位了。更改器件配置需要更改引导闪存中的配置数据，并对器件掉电再上电。

为确保128位数据的完整性，将连续比较每个配置位及其存储的补码。如果检测到不匹配，则会生成配置不匹配复位信号，从而导致器件复位。

19.2 器件代码保护位（CP）

PIC32系列器件具有代码保护功能，使能后可防止外部编程器件读取闪存。使能代码保护之后，只能使用全片擦除命令（MCHP_ERASE）擦除器件来禁止此功能。

对选择使用代码保护的器件进行编程时，必须在使能代码保护之前对编程器件执行校验。使能代码保护应该是编程过程的最后一步。代码保护使能位的位置因器件而异。有关详细信息，请参见特定器件数据手册中的**“特殊功能”**章节。

注：	使能代码保护之后，无法再读取闪存，而只能使用全片擦除命令（MCHP_ERASE）通过外部编程器禁止闪存。

19.3 程序写保护位（PWP）

PIC32系列器件具有写保护功能，可防止在程序执行期间擦除或写入指定的引导和程序闪存区域。

在PIC32MX器件中，写保护通过器件配置字在配置存储器中实现，而在PIC32MZ和PIC32MK系列器件中，此功能通过闪存控制器中的SFR实现。

当通过器件配置字实现写保护时，写保护寄存器只有在所有引导和程序闪存编程后才能写入。有关详细信息，请参见具体器件数据手册中的**“特殊功能”**章节。

如果通过SFR实现写保护，则在编程闪存区域前，可能需要通过外部编程器在器件初始化期间执行一些步骤。有关详细信息，请参见具体器件数据手册中的**“闪存程序存储器”**章节。

© 2014-2020 Microchip Technology Inc.

20.0 TAP控制器

表20-1： MCHP TAP指令

命令	值	说明
MTAP_COMMAND	5'h0x07	TDI和TDO连接到MCHP命令移位寄存器（见表20-2）
MTAP_SW_MTAP	5'h0x04	将TAP控制器切换为MCHP TAP控制器
MTAP_SW_ETAP	5'h0x05	将TAP控制器切换为EJTAG TAP控制器
MTAP_IDCODE	5'h0x01	选择芯片标识数据寄存器

20.1 Microchip TAP控制器（MTAP）

20.1.1 MTAP_COMMAND指令

MTAP_COMMAND选择MCHP命令移位寄存器。有关可用命令，请参见表20-2。

20.1.1.1 MCHP_STATUS指令

MCHP_STATUS返回Microchip TAP控制器的8位状态值。表20-3提供返回的状态值的格式。

20.1.1.2 MCHP_ASSERT_RST指令

MCHP_ASSERT_RST执行持久性器件复位。它类似于将$\overline{\text{MCLR}}$引脚置为有效并保持有效。其关联的状态位为DEVRST。

20.1.1.3 MCHP_DE_ASSERT_RST指令

MCHP_DE_ASSERT_RST取消持久性器件复位。它类似于将$\overline{\text{MCLR}}$引脚置为无效。其关联的状态位为DEVRST。

20.1.1.4 MCHP_ERASE指令

MCHP_ERASE执行全片擦除操作。CHIP_ERASE命令将一个请求闪存控制器执行擦除的内部位置1。一旦控制器变为繁忙状态（通过FCBUSY（状态位）指示），内部位将清零。

20.1.1.5 MCHP_FLASH_ENABLE指令

MCHP_FLASH_ENABLE将FAEN位置1，该位控制处理器对闪存的访问。FAEN位的状态在同名字段中返回。如果CPS = 0，则该命令无效。此命令需要一个NOP来完成。

注： PIC32MZ和PIC32MK系列器件不需要此命令。

20.1.1.6 MCHP_FLASH_DISABLE指令

MCHP_FLASH_DISABLE将FAEN位清零，该位控制处理器对闪存的访问。FAEN位的状态在同名字段中返回。如果CPS = 0，则该命令无效。此命令需要一个NOP来完成。

注： PIC32MZ和PIC32MK系列器件不需要此命令。

20.1.2 MTAP_SW_MTAP指令

MTAP_SW_MTAP将TAP指令集切换为MCHP TAP指令集。

其中每条命令都应后跟一个SetMode (6'b011111)，强制芯片TAP控制器进入运行测试/空闲状态。

20.1.3 MTAP_SW_ETAP指令

MTAP_SW_ETAP将TAP指令集有效切换为EJTAG TAP指令集。为此，它会将EJTAG TAP控制器保持在运行测试/空闲状态，直到MCHP TAP控制器对MTAP_SW_ETAP指令进行解码。

其中每条命令都应后跟一个SetMode (6'b011111)，强制芯片TAP控制器进入运行测试/空闲状态。

20.1.4 MTAP_IDCODE指令

MTAP_IDCODE返回存储在DEVID寄存器中的值。

表20-2： MTAP_COMMAND DR命令

命令	值	说明
MCHP_STATUS	8'h0x00	NOP和返回状态。
MCHP_ASSERT_RST	8'h0xD1	请求复位控制器将器件复位置为有效。
MCHP_DE_ASSERT_RST	8'h0xD0	取消器件复位请求，如果没有其他请求复位的源（如$\overline{\text{MCLR}}$），这会使复位控制器将器件复位置为无效。
MCHP_ERASE	8'h0xFC	使闪存控制器执行全片擦除。
MCHP_FLASH_ENABLE(1)	8'h0xFE	使能取数据并装入闪存（从处理器中）。
MCHP_FLASH_DISABLE(1)	8'h0xFD	禁止取数据并装入闪存（从处理器中）。

注 1： PIC32MK和PIC32MZ系列器件不需要此命令。

表20-3： MCHP状态值

位范围	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
7:0	CPS	0	NVMERR(1)	0	CFGRDY	FCBUSY	FAEN(2)	DEVRST

bit 7 **CPS：**代码保护状态位
1 = 器件不受代码保护
0 = 器件受代码保护

bit 6 **未实现：**读为0

bit 5 **NVMERR：**NVMCON状态位(1)
1 = 在NVM操作期间发生错误
0 = 在NVM操作期间未发生错误

bit 4 **未实现：**读为0

bit 3 **CFGRDY：**代码保护状态位
1 = 配置已读取且CP为有效
0 = 配置未读取

bit 2 **FCBUSY：**闪存控制器忙状态位
1 = 闪存控制器正忙（正在擦除）
0 = 闪存控制器不忙（擦除未开始或已完成）

bit 1 **FAEN：**闪存访问使能位(2)
该位反映CFGCON.FAEN的状态。
1 = 使能闪存访问
0 = 禁止闪存访问（例如，阻止处理器访问）

bit 0 **DEVRST：**器件复位状态位
1 = 器件复位有效
0 = 器件复位无效

注 1： PIC32MX320/340/360/420/440/460器件中未实现该位。
2： PIC32MK和PIC32MZ系列器件中未实现该位。

表20-4： EJTAG TAP指令

命令	值	说明
ETAP_ADDRESS	5'h0x08	选择地址寄存器。
ETAP_DATA	5'h0x09	选择数据寄存器。
ETAP_CONTROL	5'h0x0A	选择EJTAG控制寄存器。
ETAP_EJTAGBOOT	5'h0x0C	将EjtagBrk、ProbEn和ProbTrap设置为1，作为复位值。
ETAP_FASTDATA	5'h0x0E	选择数据寄存器和Fastdata寄存器。

© 2014-2020 Microchip Technology Inc.

20.2 EJTAG TAP控制器

20.2.1 ETAP_ADDRESS命令

ETAP_ADDRESS选择地址寄存器。只读地址寄存器提供用于处理器访问的地址。如果处理器访问处于待处理状态，则在该寄存器中读取的值为有效，否则该值处于未定义状态。

该寄存器的两个或三个最低有效字节（Least Significant Byte，LSB）与EJTAG控制寄存器的Psz域一起用于指示待处理的处理器访问传输的大小和数据位置。这些位并非直接取自装载/存储所引用的地址。

20.2.2 ETAP_DATA命令

ETAP_DATA选择数据寄存器。读/写数据寄存器用于处理器访问期间的操作码和数据传输。只有当处理器写访问处于待处理状态，在该寄存器中读取的值才为有效，在这种情况下，数据寄存器将保留存储的值。只有当处理器将在之后完成待处理读访问时，才会使用写入到数据寄存器中的值，在这种情况下，写入的数据值为取出或装载的值。此行为暗示数据寄存器并非是一个之前写入的值可以在之后读取的存储单元。

20.2.3 ETAP_CONTROL命令

ETAP_CONTROL选择控制寄存器。EJTAG控制寄存器（EJTAG Control Register，ECR）处理以下各项操作：处理器复位、软复位指示、调试模式指示、访问开始、完成和大小，以及读/写指示。ECR还提供以下功能：

- 控制调试向量地址和处理的处理器访问的指示
- 可发出调试中断请求
- 指示处理器的低功耗模式
- 支持依赖于实现的处理器和外设复位

20.2.3.1 EJTAG控制寄存器（ECR）

除非发生了复位，否则将不会在“更新-DR”状态中更新/写入EJTAG控制寄存器（见寄存器20-1），也就是说Rocc（bit 31）已经为0，或将被同时写入为0。此条件可确保在复位后正确处理处理器访问。

在处理器被从处理器时钟域中移除若干个TCK周期后，处理器的复位可通过TCK域中的Rocc位进行指示，以便能够在两个时钟域之间实现正确同步。

除非定义了其他行为，否则在寄存器中可读/写（R/W）的位将在后续读取时返回其写入的值。

内部同步可确保写入的值为紧接之后的读取进行更新，即使TAP控制器采取从“更新-DR”到“捕捉-DR”状态的最短路径时也是如此。

寄存器20-1： ECR：EJTAG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	R/W-0	R-0	R-0	U-0	U-0	U-0	U-0	U-0
	Rocc	Psz<1:0>		—	—	—	—	—
23:16	R-0	R-0	R-0	R/W-0	R-0	R/W-0	U-0	R/W-0
	VPED	Doze	Halt	PerRst	PrnW	PrACC	—	PrRst
15:8	R/W-0	R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0
	ProbEn	ProbTrap	—	EjtagBrk	—	—	—	—
7:0	U-0	U-0	U-0	U-0	R-0	U-0	U-0	U-0
	—	—	—	—	DM	—	—	—

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 31-29 请参见**注1**

bit 28-24 **未实现：**读为0

bit 23-19 请参见**注1**

bit 18 **PrACC：**待处理的处理器访问和控制位
此位指示待处理的处理器访问，并控制待处理的处理器访问的完成。写入0可完成待处理的处理器访问。写入1则将忽略。成功的FASTDATA访问会将此位清零。
1 = 待处理的处理器访问
0 = 无待处理的预处理器访问

bit 17 **未实现：**读为0

bit 16 请参见**注1**

bit 15 **ProbEn：**处理器访问处理控制位
此位控制探针通过处理处理器访问而处理DMSEG段访问的位置。
1 = 探针处理处理器访问
0 = 探针不处理处理器访问

bit 14 **ProbTrap：**调试异常向量地址控制位
此位控制调试异常向量的地址。
1 = 0xFF200200
0 = 0xBFC00480

bit 13 **未实现：**读为0

bit 12 **EjtagBrk：**调试中断异常请求位
此位可在写为1时向处理器请求调试中断异常。写入0将被忽略。
1 = 处于待处理状态调试中断异常请求
0 = 未处于待处理状态调试中断异常请求

bit 11-4 **未实现：**读为0

bit 3 请参见**注1**

bit 2-0 **未实现：**读为0

注 1： 有关这些位的说明，请访问Imagination Technologies Limited网站（www.imgtec.com）。

© 2014-2020 Microchip Technology Inc.

20.2.4 ETAP_EJTAGBOOT命令

ETAP_EJTAGBOOT命令使处理器在复位后从调试异常向量取代码。这使编程器可以将指令发送到处理器执行，而不是使处理器从正常的复位向量取指令。在设置内部EJTAGBOOT指示之后是EjtagBrk、ProbTrap和ProbE的复位值。

如果EJTAGBOOT指令已给出，且内部EJTAGBOOT指示处于有效状态，则这三个位的复位值将被置1（1）；否则复位值将被清零（0）。

这些位置1的结果为：

- 如将EjtagBrk置1，则将在紧接处理器从EJTAGBOOT指令复位后请求调试中断异常
- 将从EJTAG存储区执行调试处理程序，因为ProbTrap被置1，指示EJTAG存储区中的调试向量位于0xFF200200
- ProbEn被置1将指示处理器访问的处理

完成此配置后，将发生中断异常，且处理器将在0xFF200200从DMSEG取处理程序。在ProbEn置1后，处理器将等待探针提供指令。

20.2.5 ETAP_FASTDATA命令

ETAP_FASTDATA命令提供了一种可在处理器和探针之间快速传输数据的机制。Fastdata寄存器的宽度为一位。在快速数据访问期间，将写入和读取Fastdata寄存器（即移入一位，并移出一位）。在快速数据访问期间，移入的Fastdata寄存器值可指定是否应完成快速数据访问。移出的值是一个标志，用于指示快速数据访问是否成功（如果请求完成）。FASTDATA 访问用于DMSEG段（探针上）和目标存储器（处理器上）之间的高效块传输。“上传”定义为一个处理器从目标存储器装载并存储到DMSEG段的序列。“下载”是指一个处理器从DMSEG段装载并存储到目标存储器的序列。“Fastdata区域”指定可用于上传和下载的DMSEG段地址的合法范围（0xFF200000到0xFF20000F）。数据和Fastdata寄存器（使用FASTDATA指令选择）可高效地完成Fastdata区域的访问。

在Fastdata上传和下载期间，处理器将在访问Fastdata区域时暂停。PrAcc（处理器访问待处理位）将变为1，指示需要探针才能完成访问。上传和下载访问的尝试方式都是，移入一个零SPrAcc值（以请求完成访问）并移出SPrAcc，以查看尝试是否成功（即存在一个待处理访问并且使用了合法的Fastdata区域地址）。

下载也将移入用于满足从DMSEG段Fastdata区域装载的数据，同时上传将移出存储到DMSEG段Fastdata区域的数据。

如前文所述，以下两个条件必须为真，Fastdata访问才能成功：

- PrAcc必须为1（即必须存在待处理的处理器访问）
- Fastdata操作必须使用DMSEG段（0xFF200000至0xFF20000F）中的有效Fastdata区域地址

21.0 交流/直流特性和时序要求

表21-1： 交流/直流特性和时序要求

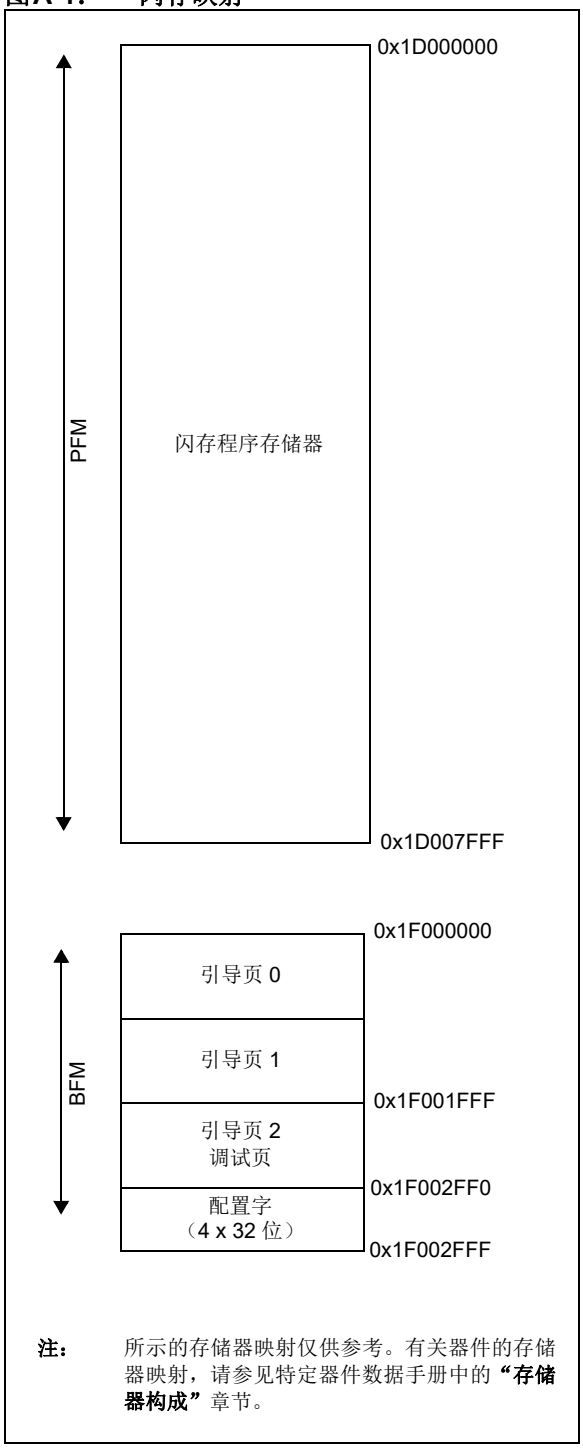
标准工作条件 工作温度：0°C至+70°C。建议在+25°C下编程。						
参数编号	**符号**	**特性**	**最小值**	**最大值**	**单位**	**条件**
D111	VDDIO	编程时的电源电压	—	—	V	请参见**注1**
D112a	VDDCORE	编程时的内核电源电压	—	—	V	请参见**注1**
D112b	VDDR1V8	编程时的DDR SDRAM电源电压	—	—	V	请参见**注1**
D113	IDDP	编程时的电源电流	—	—	mA	请参见**注1**
D114	IPEAK	启动时的瞬时峰值电流	—	—	mA	请参见**注1**
D115a	IDDCORE	编程时的内核电源电流	—	—	mA	请参见**注1**
D115b	IDDR1V8P	编程时的DDR SDRAM电源电流	—	—	mA	请参见**注1**
D116	VDDVBAT	编程时的VBAT电源电压	—	—	V	请参见**注1**
D117	IDDVBAT	编程时的VBAT电源电流	—	—	mA	请参见**注1**
D031	VIL	输入低电压	—	—	V	请参见**注1**
D041	VIH	输入高电压	—	—	V	请参见**注1**
D080	VOL	输出低电压	—	—	V	请参见**注1**
D090	VOH	输出高电压	—	—	V	请参见**注1**
D012	CIO	I/O引脚（PGEDx）上的容性负载	—	—	pF	请参见**注1**
D013	CF	VCAP上的滤波电容值	—	—	µF	请参见**注1**
P1	TPGC	串行时钟（PGECx）周期	100	—	ns	—
P1A	TPGCL	串行时钟（PGECx）的低电平时间	40	—	ns	—
P1B	TPGCH	串行时钟（PGECx）的高电平时间	40	—	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	500	—	ns	—
P9a	TDLY4	PE命令处理时间	40	—	µs	—
P9b	TDLY5	PE将PGEDx驱动为↓到PE释放PGEDx之间的延时	15	—	µs	—
P11	TDLY7	全片擦除时间	—	—	ms	请参见**注1**
P12	TDLY8	页擦除时间	—	—	ms	请参见**注1**
P13	TDLY9	行编程时间	—	—	ms	请参见**注1**
P14	TR	进入ICSP™模式的$\overline{\text{MCLR}}$上升时间	—	1.0	µs	—
P15	TVALID	PGECx↑后的数据输出有效时间	10	—	ns	—
P16	TDLY8	最后一个PGECx↓到$\overline{\text{MCLR}}$↓之间的延时	0	—	s	—
P17	THLD3	$\overline{\text{MCLR}}$↓到VDD↓的时间	—	100	ns	—
P18	TKEY1	从第一个$\overline{\text{MCLR}}$↓到向PGEDx输入密钥序列的第一个PGECx↑之间的延时	40	—	ns	—
P19	TKEY2	从向PGEDx输入密钥序列的最后一个PGECx↓到第二个$\overline{\text{MCLR}}$↑之间的延时	40	—	ns	—
P20	TMCLRH	$\overline{\text{MCLR}}$高电平时间	—	500	µs	—

注 1： 有关此参数的最大值和最小值，请参见具体器件数据手册中的**“电气特性”**章节。

© 2014-2020 Microchip Technology Inc.

附录A： PIC32闪存映射

图A-1： 闪存映射



注： 所示的存储器映射仅供参考。有关器件的存储器映射，请参见特定器件数据手册中的**“存储器构成”**章节。

附录B： HEX文件格式

闪存编程器可处理Microchip开发工具所使用的标准十六进制（hex）文件。支持的格式为Intel® HEX32格式（INHX32）。有关hex文件格式的更多信息，请参见《MPASM™汇编器、MPLINK™目标链接器、MPLIB™目标库管理器用户指南》（DS33014L_CN）中的**第1.7.5节“Hex文件格式”**。

hex文件的基本格式如下：

```
:BBAAAATTHHHH...HHHHCC
```

每条数据记录以9个字符的前缀开始，并总是以2个字符的校验和结尾。无论格式如何，所有记录均以“:”开始。各个元素如下所述。

- BB——是一个两位的十六进制字节计数，表示出现在行中的数据字节数。将此数值除以二就可以得到每行的字数。
- AAAA——是一个四位的十六进制地址，表示数据记录的开始地址。格式为先高字节，后低字节。
- TT——是一个两位记录类型，对于数据记录为00，对于文件结束记录为01，对于延长的地址记录为04。
- HHHH——是一个四位的十六进制数据字。格式为先低字节，后高字节。在TT后将有BB/2个数据字。
- CC——2位数字的十六进制校验和，它是行记录中所有先前字节之和的二进制补码。

由于Intel hex文件格式是针对字节的，而16位程序计数器不是，所以程序存储器段需要特殊的处理。每个24位程序字都通过插入所谓的“虚拟字节”而扩展为32位。每个程序存储器地址都乘以2来获得字节地址。

例如，在程序存储器中位于0x100的段在hex文件中将表示为0x200。

产生的hex文件将具有以下内容：

```
:020000040000fa
:040200003322110096
:00000001FF
```

数据记录（第2行）的装入地址为0200，而源代码指定的地址为0x100。数据以“小尾数法”格式表示，最低有效字节先出现，虚拟字节最后出现，位于校验和之前。

附录C： 器件ID

表C-1： PIC32MX320/340/360/440/460系列器件ID

器件编号	器件ID（见注1）
PIC32MX360F512L	0x0938053
PIC32MX360F256L	0x0934053
PIC32MX340F128L	0x092D053
PIC32MX320F128L	0x092A053
PIC32MX340F512H	0x0916053
PIC32MX340F256H	0x0912053
PIC32MX340F128H	0x090D053
PIC32MX320F128H	0x090A053
PIC32MX320F064H	0x0906053
PIC32MX320F032H	0x0902053
PIC32MX460F512L	0x0978053
PIC32MX460F256L	0x0974053
PIC32MX440F128L	0x096D053
PIC32MX440F256H	0x0952053
PIC32MX440F512H	0x0956053
PIC32MX440F128H	0x094D053
PIC32MX420F032H	0x0942053

表C-2： PIC32MX575/675/695/775/795系列器件ID

器件编号	器件ID（见注1）
PIC32MX575F256H	0x04317053
PIC32MX675F256H	0x0430B053
PIC32MX775F256H	0x04303053
PIC32MX575F512H	0x04309053
PIC32MX675F512H	0x0430C053
PIC32MX695F512H	0x04325053
PIC32MX775F512H	0x0430D053
PIC32MX795F512H	0x0430E053
PIC32MX575F256L	0x04333053
PIC32MX675F256L	0x04305053
PIC32MX775F256L	0x04312053
PIC32MX575F512L	0x0430F053
PIC32MX675F512L	0x04311053
PIC32MX695F512L	0x04341053
PIC32MX775F512L	0x04307053
PIC32MX795F512L	0x04307053

表C-3： PIC32MX534/564/664/764系列器件ID

器件编号	器件ID（见注1）
PIC32MX534F064H	0x04400053
PIC32MX564F064H	0x04401053
PIC32MX564F128H	0x04403053
PIC32MX664F064H	0x04405053
PIC32MX664F128H	0x04407053
PIC32MX764F128H	0x0440B053
PIC32MX534F064L	0x0440C053
PIC32MX564F064L	0x0440D053
PIC32MX564F128L	0x0440F053
PIC32MX664F064L	0x04411053
PIC32MX664F128L	0x04413053
PIC32MX764F128L	0x04417053

表C-4： PIC32MX1XX/2XX 28/36/44引脚系列器件ID

器件编号	器件ID（见注1）
PIC32MX110F016B	0x04A07053
PIC32MX110F016C	0x04A09053
PIC32MX110F016D	0x04A0B053
PIC32MX210F016B	0x04A01053
PIC32MX210F016C	0x04A03053
PIC32MX210F016D	0x04A05053
PIC32MX120F032B	0x04A06053
PIC32MX120F032C	0x04A08053
PIC32MX120F032D	0x04A0A053
PIC32MX220F032B	0x04A00053
PIC32MX220F032C	0x04A02053
PIC32MX220F032D	0x04A04053
PIC32MX130F064B	0x04D07053
PIC32MX130F064C	0x04D09053
PIC32MX130F064D	0x04D0B053
PIC32MX230F064B	0x04D01053
PIC32MX230F064C	0x04D03053
PIC32MX230F064D	0x04D05053
PIC32MX150F128B	0x04D06053
PIC32MX150F128C	0x04D08053
PIC32MX150F128D	0x04D0A053
PIC32MX250F128B	0x04D00053
PIC32MX250F128C	0x04D02053
PIC32MX250F128D	0x04D04053
PIC32MX170F256B	0x06610053
PIC32MX170F256D	0x0661A053
PIC32MX270F256B	0x06600053
PIC32MX270F256D	0x0660A053
PIC32MX270F256DB	0x0660C053
PIC32MX130F256B	0x06703053
PIC32MX130F256D	0x06705053
PIC32MX230F256B	0x06700053
PIC32MX230F256D	0x06702053

© 2014-2020 Microchip Technology Inc.

表C-5： PIC32MX330/350/370/430/450/470系列器件ID

器件编号	器件ID（见注1）
PIC32MX330F064H	0x05600053
PIC32MX330F064L	0x05601053
PIC32MX350F256H	0x05704053
PIC32MX350F256L	0x05705053
PIC32MX430F064H	0x05602053
PIC32MX430F064L	0x05603053
PIC32MX450F256H	0x05706053
PIC32MX450F256L	0x05707053
PIC32MX350F128H	0x0570C053
PIC32MX350F128L	0x0570D053
PIC32MX450F128H	0x0570E053
PIC32MX450F128L	0x0570F053
PIC32MX370F512H	0x05808053
PIC32MX370F512L	0x05809053
PIC32MX470F512H	0x0580A053
PIC32MX470F512L	0x0580B053
PIC32MX450F256HB	0x05710053
PIC32MX470F512LB	0x05811053

表C-6： PIC32MZ嵌入式连接（EC）系列器件ID

器件编号	器件ID（见注1）
PIC32MZ1024ECG064	0x05103053
PIC32MZ1024ECH064	0x05108053
PIC32MZ1024ECM064	0x05130053
PIC32MZ2048ECG064	0x05104053
PIC32MZ2048ECH064	0x05109053
PIC32MZ2048ECM064	0x05131053
PIC32MZ1024ECG100	0x0510D053
PIC32MZ1024ECH100	0x05112053
PIC32MZ1024ECM100	0x0513A053
PIC32MZ2048ECG100	0x0510E053
PIC32MZ2048ECH100	0x05113053
PIC32MZ2048ECM100	0x0513B053
PIC32MZ1024ECG124	0x05117053
PIC32MZ1024ECH124	0x0511C053
PIC32MZ1024ECM124	0x05144053
PIC32MZ2048ECG124	0x05118053
PIC32MZ2048ECH124	0x0511D053
PIC32MZ2048ECM124	0x05145053
PIC32MZ1024ECG144	0x05121053
PIC32MZ1024ECH144	0x05126053
PIC32MZ1024ECM144	0x0514E053
PIC32MZ2048ECG144	0x05122053
PIC32MZ2048ECH144	0x05127053
PIC32MZ2048ECM144	0x0514F053

表C-7： PIC32MX1XX/2XX/5XX 64/100引脚系列器件ID

器件编号	器件ID（见注1）
PIC32MX150F256H	0x06A10053
PIC32MX150F256L	0x06A11053
PIC32MX170F512H	0x06A30053
PIC32MX170F512L	0x06A31053
PIC32MX250F256H	0x06A12053
PIC32MX250F256L	0x06A13053
PIC32MX270F512H	0x06A32053
PIC32MX270F512L	0x06A33053
PIC32MX550F256H	0x06A14053
PIC32MX550F256L	0x06A15053
PIC32MX570F512H	0x06A34053
PIC32MX570F512L	0x06A35053
PIC32MX120F064H	0x06A50053
PIC32MX130F128H	0x06A00053
PIC32MX130F128L	0x06A01053
PIC32MX230F128H	0x06A02053
PIC32MX230F128L	0x06A03053
PIC32MX530F128H	0x06A04053
PIC32MX530F128L	0x06A05053

表C-8： 带FPU的PIC32MZ嵌入式连接（EF）系列器件ID

器件编号	器件ID（见注1）
PIC32MZ0512EFE064	0x07201053
PIC32MZ0512EFF064	0x07206053
PIC32MZ0512EFK064	0x0722E053
PIC32MZ1024EFE064	0x07202053
PIC32MZ1024EFF064	0x07207053
PIC32MZ1024EFK064	0x0722F053
PIC32MZ1024EFG064	0x07203053
PIC32MZ1024EFH064	0x07208053
PIC32MZ1024EFM064	0x07230053
PIC32MZ2048EFG064	0x07204053
PIC32MZ2048EFH064	0x07209053
PIC32MZ2048EFM064	0x07231053
PIC32MZ0512EFE100	0x0720B053
PIC32MZ0512EFF100	0x07210053
PIC32MZ0512EFK100	0x07238053
PIC32MZ1024EFE100	0x0720C053
PIC32MZ1024EFF100	0x07211053
PIC32MZ1024EFK100	0x07239053
PIC32MZ1024EFG100	0x0720D053
PIC32MZ1024EFH100	0x07212053
PIC32MZ1024EFM100	0x0723A053
PIC32MZ2048EFG100	0x0720E053
PIC32MZ2048EFH100	0x07213053
PIC32MZ2048EFM100	0x0723B053
PIC32MZ0512EFE124	0x07215053
PIC32MZ0512EFF124	0x0721A053
PIC32MZ0512EFK124	0x07242053
PIC32MZ1024EFE124	0x07216053
PIC32MZ1024EFF124	0x0721B053
PIC32MZ1024EFK124	0x07243053
PIC32MZ1024EFG124	0x07217053
PIC32MZ1024EFH124	0x0721C053
PIC32MZ1024EFM124	0x07244053
PIC32MZ2048EFG124	0x07218053
PIC32MZ2048EFH124	0x0721D053
PIC32MZ2048EFM124	0x07245053
PIC32MZ0512EFE144	0x0721F053
PIC32MZ0512EFF144	0x07224053
PIC32MZ0512EFK144	0x0724C053
PIC32MZ1024EFE144	0x07220053
PIC32MZ1024EFF144	0x07225053
PIC32MZ1024EFK144	0x0724D053
PIC32MZ1024EFG144	0x07221053
PIC32MZ1024EFH144	0x07226053
PIC32MZ1024EFM144	0x0724E053
PIC32MZ2048EFG144	0x07222053
PIC32MZ2048EFH144	0x07227053
PIC32MZ2048EFM144	0x0724F053

表C-9： PIC32MZ图形（DA）系列器件ID

器件编号	器件ID（见注1）
PIC32MZ1025DAA169	0x05F0C053
PIC32MZ1025DAB169	0x05F0D053
PIC32MZ1064DAA169	0x05F0F053
PIC32MZ1064DAB169	0x05F10053
PIC32MZ2025DAA169	0x05F15053
PIC32MZ2025DAB169	0x05F16053
PIC32MZ2064DAA169	0x05F18053
PIC32MZ2064DAB169	0x05F19053
PIC32MZ1025DAG169	0x05F42053
PIC32MZ1025DAH169	0x05F43053
PIC32MZ1064DAG169	0x05F45053
PIC32MZ1064DAH169	0x05F46053
PIC32MZ2025DAG169	0x05F4B053
PIC32MZ2025DAH169	0x05F4C053
PIC32MZ2064DAG169	0x05F4E053
PIC32MZ2064DAH169	0x05F4F053
PIC32MZ1025DAA176	0x05F78053
PIC32MZ1025DAB176	0x05F79053
PIC32MZ1064DAA176	0x05F7B053
PIC32MZ1064DAB176	0x05F7C053
PIC32MZ2025DAA176	0x05F81053
PIC32MZ2025DAB176	0x05F82053
PIC32MZ2064DAA176	0x05F84053
PIC32MZ2064DAB176	0x05F85053
PIC32MZ1025DAG176	0x05FAE053
PIC32MZ1025DAH176	0x05FAF053
PIC32MZ1064DAG176	0x05FB1053
PIC32MZ1064DAH176	0x05FB2053
PIC32MZ2025DAG176	0x05FB7053
PIC32MZ2025DAH176	0x05FB8053
PIC32MZ2064DAG176	0x05FBA053
PIC32MZ2064DAH176	0x05FBB053
PIC32MZ1025DAA288	0x05F5D053
PIC32MZ1025DAB288	0x05F5E053
PIC32MZ1064DAA288	0x05F60053
PIC32MZ1064DAB288	0x05F61053
PIC32MZ2025DAA288	0x05F66053
PIC32MZ2025DAB288	0x05F67053
PIC32MZ2064DAA288	0x05F69053
PIC32MZ2064DAB288	0x05F6A053
PIC32MZ1025DAK169	0x08A0C053
PIC32MZ1025DAL169	0x08A0D053
PIC32MZ1064DAK169	0x08A0F053
PIC32MZ1064DAL169	0x08A10053
PIC32MZ2025DAK169	0x08A15053
PIC32MZ2025DAL169	0x08A16053
PIC32MZ2064DAK169	0x08A18053
PIC32MZ2064DAL169	0x08A19053

© 2014-2020 Microchip Technology Inc.

表C-9： PIC32MZ图形（DA）系列器件ID

器件编号	器件ID（见注1）
PIC32MZ1025DAR169	0x08A42053
PIC32MZ1025DAS169	0x08A43053
PIC32MZ1064DAR169	0x08A45053
PIC32MZ1064DAS169	0x08A46053
PIC32MZ2025DAR169	0x08A4B053
PIC32MZ2025DAS169	0x08A4C053
PIC32MZ2064DAR169	0x08A4E053
PIC32MZ2064DAS169	0x08A4F053
PIC32MZ1025DAK176	0x08A78053
PIC32MZ1025DAL176	0x08A79053
PIC32MZ1064DAK176	0x08A7B053
PIC32MZ1064DAL176	0x08A7C053
PIC32MZ2025DAK176	0x08A81053
PIC32MZ2025DAL176	0x08A82053
PIC32MZ2064DAK176	0x08A84053
PIC32MZ2064DAL176	0x08A85053
PIC32MZ1025DAR176	0x08AAE053
PIC32MZ1025DAS176	0x08AAF053
PIC32MZ1064DAR176	0x08AB1053
PIC32MZ1064DAS176	0x08AB2053
PIC32MZ2025DAR176	0x08AB7053
PIC32MZ2025DAS176	0x08AB8053
PIC32MZ2064DAR176	0x08ABA053
PIC32MZ2064DAS176	0x08ABB053

表C-10： PIC32MX1XX/2XX 28/44引脚XLP系列器件ID

器件编号	器件ID（见注1）
PIC32MX154F128B	0x07800053
PIC32MX154F128D	0x07804053
PIC32MX155F128B	0x07808053
PIC32MX155F128D	0x0780C053
PIC32MX174F256B	0x07801053
PIC32MX174F256D	0x07805053
PIC32MX175F256B	0x07809053
PIC32MX175F256D	0x0780D053
PIC32MX254F128B	0x07802053
PIC32MX254F128D	0x07806053
PIC32MX255F128B	0x0780A053
PIC32MX255F128D	0x0780E053
PIC32MX274F256B	0x07803053
PIC32MX274F256D	0x07807053
PIC32MX275F256B	0x0780B053
PIC32MX275F256D	0x0780F053

表C-11： PIC32MK通用和电机控制（GP/MC）系列器件ID

器件编号	器件ID（见注1）
PIC32MK1024MCF100	0x06201053
PIC32MK1024MCF064	0x06202053
PIC32MK0512MCF100	0x06204053
PIC32MK0512MCF064	0x06205053
PIC32MK1024GPE100	0x06207053
PIC32MK1024GPE064	0x06208053
PIC32MK0512GPE100	0x0620A053
PIC32MK0512GPE064	0x0620B053
PIC32MK1024GPD100	0x0620D053
PIC32MK1024GPD064	0x0620E053
PIC32MK0512GPD100	0x06210053
PIC32MK0512GPD064	0x06211053

表C-12： 带ECC的PIC32MK通用和电机控制（GP/MC）闪存系列器件ID

器件编号	器件ID（见注1）
PIC32MK1024MCM100	0x08B01053
PIC32MK1024MCM064	0x08B02053
PIC32MK0512MCM100	0x08B04053
PIC32MK0512MCM064	0x08B05053
PIC32MK1024GPL100	0x08B07053
PIC32MK1024GPL064	0x08B08053
PIC32MK0512GPL100	0x08B0A053
PIC32MK0512GPL064	0x08B0B053
PIC32MK1024GPK100	0x08B0D053
PIC32MK1024GPK064	0x08B0E053
PIC32MK0512GPK100	0x08B10053
PIC32MK0512GPK064	0x08B11053
PIC32MK0512MCJ064	0x06300053
PIC32MK0512MCJ048	0x06301053
PIC32MK0512MCJ040	0x06302053
PIC32MK0256MCJ064	0x06304053
PIC32MK0256MCJ048	0x06305053
PIC32MK0256MCJ040	0x06306053
PIC32MK0512GPH064	0x06308053
PIC32MK0512GPH048	0x06309053
PIC32MK0512GPH040	0x0630A053
PIC32MK0256GPH064	0x0630C053
PIC32MK0256GPH048	0x0630D053
PIC32MK0256GPH040	0x0630E053
PIC32MK0512GPG064	0x06318053
PIC32MK0512GPG048	0x06319053
PIC32MK0512GPG040	0x0631A053
PIC32MK0256GPG064	0x0631C053
PIC32MK0256GPG048	0x0631D053
PIC32MK0256GPG040	0x0631E053

注 1： 32位器件ID的前4位表示芯片版本。

© 2014-2020 Microchip Technology Inc.

附录D： 版本历史

版本A（2007年8月）

本文档的初始版本。

版本B（2008年2月）

无版本更新记录。

版本C（2008年4月）

无版本更新记录。

版本D（2008年5月）

无版本更新记录。

版本E（2009年7月）

此文档版本包含以下新增内容和更新：

- 对整篇文档的样式和格式进行了少量更改
- 添加了以下器件：
  - PIC32MX565F256H
  - PIC32MX575F512H
  - PIC32MX675F512H
  - PIC32MX795F512H
  - PIC32MX575F512L
  - PIC32MX675F512L
  - PIC32MX795F512L
- 在图7-1中，将$\overline{\text{MCLR}}$脉冲线更新为指示高电平有效（P20）
- 更新了表11-1的步骤7，以说明在步骤中需要重复*最后一*条指令
- 更新了表13-1中的以下指令：
  - 步骤1中的第7条、第9条和第11条指令
  - 步骤2中的所有指令
  - 步骤3中的第1条指令
  - 步骤4中的第3条指令
- 在表17-1中添加了以下器件：
  - PIC32MX565F256H
  - PIC32MX575F512H
  - PIC32MX575F512L
  - PIC32MX675F512H
  - PIC32MX675F512L
  - PIC32MX795F512H
  - PIC32MX795F512L
- 更新了表17-2中的地址值

版本E（2009年7月）（续）

- 在表17-5中添加了以下器件：
  - PIC32MX565F256H
  - PIC32MX575F512H
  - PIC32MX675F512H
  - PIC32MX795F512H
  - PIC32MX575F512L
  - PIC32MX675F512L
  - PIC32MX795F512L
- 在“DEVCFG器件配置字汇总”和“DEVCFG3：器件配置字3”中添加了注1-3和以下位（见表18-1和寄存器）：
  - FVBUSIO
  - FUSBIDIO
  - FCANIO
  - FETHIO
  - FMIIEN
  - FPBDIV<1:0>
  - FJTAGEN
- 更新了DEVID汇总（见表18-1）
- 在“DEVCFG0：器件配置字0”中更新了ICESEL位说明并添加了FJTAGEN位（见寄存器16-1）
- 更新了“DEVID：器件和版本ID寄存器”
- 添加了“器件ID和版本”表（表18-4）
- 在表20-1中添加了$\overline{\text{MCLR}}$高电平时间（参数P20）
- 添加了**附录B：“Hex文件格式”**和**附录D：“版本历史”**

版本F（2010年4月）

此文档版本包含以下新增内容和更新：

- 进行了以下全局位名称更改：
  - NVMWR重命名为WR
  - NVMWREN重命名为WREN
  - NVMERR重命名为WRERR
  - FVBUSIO重命名为FVBUSONIO
  - FUPLLEN重命名为UPLLEN
  - FUPLLIDIV重命名为UPLLIDIV
  - POSCMD重命名为POSCMOD
- 更新了**第2.0节“编程概述”**第4段中的PIC32MX系列数据手册引用
- 更新了**第5.2.2节“2相ICSP”**中的注释
- 更新了启动闪存行写操作码和指令（见表13-1中的步骤4、步骤5和步骤6）

版本F（2010年4月）（续）

- 添加了以下器件：
  - PIC32MX534F064H
  - PIC32MX534F064L
  - PIC32MX564F064H
  - PIC32MX564F064L
  - PIC32MX564F128H
  - PIC32MX564F128L
  - PIC32MX575F256L
  - PIC32MX664F064H
  - PIC32MX664F064L
  - PIC32MX664F128H
  - PIC32MX664F128L
  - PIC32MX675F256H
  - PIC32MX675F256L
  - PIC32MX695F512H
  - PIC32MX605F512L
  - PIC32MX764F128H
  - PIC32MX764F128L
  - PIC32MX775F256H
  - PIC32MX775F256L
  - PIC32MX775F512H
  - PIC32MX775F512L

版本G（2010年8月）

此版本文档包含以下更新：

- 更新了“表11-1：下载PE”中的步骤3
- 对整篇文档的格式和文字进行了少量修正

版本H（2011年4月）

此文档版本包含以下新增内容和更新：

- 对整篇文档的格式和文字进行了少量修正
- 添加了以下器件：
  - PIC32MX110F016B
  - PIC32MX110F016C
  - PIC32MX110F016D
  - PIC32MX120F032B
  - PIC32MX120F032C
  - PIC32MX120F032D
  - PIC32MX210F016B
  - PIC32MX210F016C
  - PIC32MX210F016D
  - PIC32MX220F032B
  - PIC32MX220F032C
  - PIC32MX220F032D
- 在表17-1中添加了以下两行：
  - PIC32MX1X0
  - PIC32MX2X0
- 添加了新的小节：**第17.4.6节“器件受代码保护时的校验和值”**
- 删除了寄存器18-1至寄存器18-5
- 删除了表17-2
- 删除了第17.5节“PIC32器件的校验和”及其小节
- 删除了“闪存程序存储器的写保护范围”表（之前的表18-4）
- 添加了“DEVCFG存储单元（仅针对PIC32MX1X0和PIC32MX20X器件）”（见表18-3）
- 在**第18.0节“配置存储器和器件ID”**中，删除了表18-1，并更新了“表18-2：DEVID汇总”作为表18-1
- 在MCHP状态值表中添加了NVMERR位（见表19-3）
- 在表18-4中添加了以下芯片版本和版本ID：
  - 0x5——B6版本
  - 0x1——A1版本
- 在闪存映射中添加了一条注（见图A-1）
- 添加了**附录C：“闪存程序存储器数据手册澄清”**

版本J（2011年8月）

注：	本文档的版本历史从版本H直接跳到版本J，这是为了避免混淆大写字母“I”（EY）与小写字母“l”（EL）。

此版本包含以下更新：

- 将所有VCORE/VCAP修改为VCAP
- 更新了**第2.0节“编程概述”**的第四段
- 删除了“2线接口引脚”表中的“编程器引脚名称”列，并更新了$\overline{\text{MCLR}}$的引脚类型（见表4-2）
- 向“代码存储区大小”表（见表5-1）和“器件ID和版本”表（见表18-4）添加了以下新器件：
  - PIC32MX130F064B
  - PIC32MX130F064C
  - PIC32MX130F064D
  - PIC32MX150F128B
  - PIC32MX150F128C
  - PIC32MX150F128D
  - PIC32MX230F064B
  - PIC32MX230F064C
  - PIC32MX230F064D
  - PIC32MX250F128B
  - PIC32MX250F128C
  - PIC32MX250F128D
- 向“代码存储区大小”表添加了“行大小”和“页大小”列（见表5-1）

© 2014-2020 Microchip Technology Inc.

版本J（2011年8月）（续）

- 更新了“进入增强型ICSP模式”图中的PGCx信号（见图7-1）
- 更新了“擦除器件框图”（见图9-1）
- 在**第9.0节“擦除器件”**中向擦除目标器件流程中添加了新的步骤4
- 更新了“2线退出测试模式”图中的$\overline{\text{MCLR}}$信号（见图15-2）
- 在“PE命令集”表中更新了以下命令并修改了注2（见表16-2）：
  - PROGRAM_CLUSTER
  - GET_DEVICEID
  - CHANGE_CFG
- 向**第16.2.11节“GET_CRC命令”**中添加了注2
- 在“PROGRAM_CLUSTER格式”表中更新了地址和长度说明（见表16-13）
- 在“CHANGE_CFG响应”图后添加了注释（见图16-27）
- 在表17-1中更新了所有PIC32MX1XX和所有PIC32MX2XX器件中DEVCFG0和DEVCFG1的值
- 对“交流/直流特性和时序要求”表进行了如下修改（见表20-1）：
  - 更新了参数D111（V_{DD}）中的最小值
  - 添加了参数D114（I_{PEAK}）
  - 删除了参数P2、P3、P4、P4A、P5、P8和P10
- 删除了附录C：“闪存程序存储器数据手册澄清”
- 对整篇文档的文字和格式进行了少量更新

版本K（2012年7月）

此版本包含以下更新：

- 将所有PGC和PGD分别更改为：PGEC和PGED
- 更新了**第1.0节“器件概述”**，添加了该文档中的主要章节列表
- 添加了**第2.3节“数据大小”**
- 更新了**第4.0节“连接到器件”**
- 向“片上稳压器连接图”添加了注2（见图4-2）
- 向4线和2线接口引脚表添加了注2（见表4-1和表4-2）
- 更新了**第7.0节“进入2线增强型ICSP模式”**
- 更新了“进入串行执行模式”图（见图10-1）
- 更新了**第10.2节“2线接口”**中的步骤11
- 更新了**第12.2节“使用PE”**
- 更新了“启动闪存行写操作码”表中的步骤 3（见表13-1）
- 更新了“校验器件操作码”表中的步骤 1（见表14-1）
- 更新了**第15.1节“4线接口”**和**第15.2节“2线接口”**中的时间间隔
- 在**第16.0节“编程执行程序”**中添加了PE位置相关的注释
- 在整个**第16.2节“PE命令集”**中添加了操作数字段
- 更新了PROGRAM命令算法（见图16-9）
- 对所有PIC32MX1XX和PIC32MX2XX器件更新了掩码值，对所有器件更新了DEVCFG3（见表17-1）
- 更新了DCR值（见**第17.4.3节“计算校验和公式中的DCR”**和表17-2）
- 更新了校验和计算过程（见示例17-1）
- 向“代码存储区大小”表（见表 5-1）和“器件 ID和版本”表（见表18-4）中添加了以下新器件：
  - PIC32MX420F032H
  - PIC32MX330F064H
  - PIC32MX330F064L
  - PIC32MX430F064H
  - PIC32MX430F064L
  - PIC32MX340F128H
  - PIC32MX340F128L
  - PIC32MX350F128H
  - PIC32MX350F128L
  - PIC32MX350F256H
  - PIC32MX350F256L
  - PIC32MX440F128H
  - PIC32MX440F128L
  - PIC32MX450F128H
  - PIC32MX450F128L
  - PIC32MX440F256H
  - PIC32MX450F256H
  - PIC32MX450F256L
  - PIC32MX460F256L
  - PIC32MX340F512H
  - PIC32MX360F512H
  - PIC32MX370F512H
  - PIC32MX370F512L
  - PIC32MX440F512H
  - PIC32MX460F512L
  - PIC32MX470F512H
  - PIC32MX470F512L
- 向**第18.2节“器件代码保护位（CP）”**添加了一条注
- 添加了EJTAG控制寄存器（见寄存器19-1）
- 更新了**第19.2.4节“ETAP_EJTAGBOOT命令”**
- 更新了交流/直流特性和时序要求（见表20-1）：
  - 删除了参数D112
  - 将注1和2替换为新注1
  - 更新了参数 D111、D113、D114、D031、D041、D080、D090、D012、D013、P11、P12和P13
- 对整篇文档的文字和格式进行了少量更新

版本L（2013年1月）

此版本包含以下更新：

- 新增或更新了以下章节：
  - **第2.1节“具有双闪存分区和双引导区域的器件”**（新）
  - **第4.3节“电源要求”**
  - **第13.0节“启动闪存行写入”**
  - **第16.1.1节“2线ICSP EJTAG频率”**
- 更新了器件配置寄存器掩码值（见表17-1）
- 向“代码存储区大小”表和“器件ID和版本”表添加了以下器件（见表5-1和表18-4）：
  - PIC32MZ0256ECE064
  - PIC32MZ0256ECE100
  - PIC32MZ0256ECE124
  - PIC32MZ0256ECE144
  - PIC32MZ0256ECF064
  - PIC32MZ0256ECF100
  - PIC32MZ0256ECF124
  - PIC32MZ0256ECF144
  - PIC32MZ0512ECE064
  - PIC32MZ0512ECE100
  - PIC32MZ0512ECE124
  - PIC32MZ0512ECE144
  - PIC32MZ0512ECF064
  - PIC32MZ0512ECF100
  - PIC32MZ0512ECF124
  - PIC32MZ0512ECF144
  - PIC32MZ1024ECE064
  - PIC32MZ1024ECE100
  - PIC32MZ1024ECE124
  - PIC32MZ1024ECE144
  - PIC32MZ1024ECF064
  - PIC32MZ1024ECF100
  - PIC32MZ1024ECF124
  - PIC32MZ1024ECF144
  - PIC32MZ1024ECG064
  - PIC32MZ1024ECG100
  - PIC32MZ1024ECG124
  - PIC32MZ1024ECG144
  - PIC32MZ1024ECH064
  - PIC32MZ1024ECH100
  - PIC32MZ1024ECH124
  - PIC32MZ1024ECH144
  - PIC32MZ2048ECG064
  - PIC32MZ2048ECG100
  - PIC32MZ2048ECG124
  - PIC32MZ2048ECG144
  - PIC32MZ2048ECH064
  - PIC32MZ2048ECH100
  - PIC32MZ2048ECH124
  - PIC32MZ2048ECH144
- 向“PE命令集”表中添加了注3、注4、GET_CHECKSUM和QUAD_WORD_PRGM命令（见表16-2）
- 添加了**第16.2.15节“GET_CHECKSUM命令”**
- 添加了**第16.2.16节“QUAD_WORD_PROGRAM命令”**
- 更新了DEVCFG存储单元中的全部地址（见表18-1和表18-2）
- 向PIC32MZ EC系列器件添加了配置字存储单元（见表18-3）
- 更新了**第18.2节“器件代码保护位（CP）”**
- 更新了**第18.3节“程序写保护位（PWP）”**
- 在整个文档中，将所有“测试模式”更新为“编程模式”
- 对整篇文档的文字和格式进行了少量更新

版本M（2013年9月）

此版本包含以下更新：

- 将所有MIPS Technologies Inc.和www.mips.com分别更新为Imagination Technologies Limited和www.imgtec.com
- 更新了**第2.0节“编程概述”**
- 更新了**第5.1.6节“闪存”**中的最后一段
- 更新了“代码存储区大小”表并添加了注3（见表5-1）
- 更新了“擦除器件”流程图（见图9-1）
- 更新了表11-1中的步骤1、2、3和5
- 在**第13.2节“不使用PE”**中添加了一个段落
- 更新了表13-1中的步骤2、3和5
- 更新了表16-17中的操作码说明
- 更新了“器件配置掩码值”表（见表17-1）
- 删除了**第17.3节“算法”**第四段中的第一句话
- 更新了“器件ID和版本”表（见表18-4）

版本N（2014年4月）

此版本包含以下更新：

- 更新了**“表4-1“4线接口引脚”中的注2**
- 更新了**“表4-2“2线接口引脚”中的注2**
- 更新了**第9.0节“擦除器件”**第5步中的延时值
- 在“器件ID和版本”表（见表18-4）中，更新了“版本ID”和“芯片版本”列表，并添加了以下新器件：
  - PIC32MX170F256B
  - PIC32MX170F256D
  - PIC32MX270F256B
  - PIC32MX270F256D
  - PIC32MX330F064H
  - PIC32MX330F064L
  - PIC32MX350F128H
  - PIC32MX350F128L
  - PIC32MX350F256H
  - PIC32MX350F256L
  - PIC32MX430F064H
  - PIC32MX430F064L
  - PIC32MX450F128H
  - PIC32MX450F128L
  - PIC32MX450F256H
  - PIC32MX450F256L

© 2014-2020 Microchip Technology Inc.

版本P（2014年10月）

注： 本文档的版本历史从版本N直接跳到版本P，这是为了避免混淆大写字母“O”（OH）与数字零“0”。

本文当包含以下更新：

- 更新了**表5-1：“代码存储区大小”**，以包含PIC32MK器件信息
- 更新了**表18-1：“器件配置寄存器的掩码值（目前支持的PIC32MX、PIC32MZ和PIC32MKxxxxGPD/GPE/MCFxxx器件）”**，以包含PIC32MK器件信息
- 删除了原始表18-4：“器件ID和版本”，该信息现在可从当前的系列芯片勘误表中获得
- 新增了**表19-4：“PIC32MK系列器件的配置字存储单元”**

版本Q（2015年7月）

此版本包含以下更新：

- 新增了**第14.0节“启动闪存行写入”**
- 更新了**表18-1：“目前支持的PIC32MX、PIC32MZ和PIC32MK器件的器件配置寄存器掩码值”**，以包含DEVCFG4
- 更新了**公式18-1：“校验和公式”**
- 更新了**表 19-3：“PIC32MZ 系列器件的配置字存储单元”**，以包含DEVCFG4
- 对整篇文档的文字和格式进行了少量更新

版本R（2016年4月）

此版本包含以下更新：

- 更新了**图4-1：“编程接口”**
- 更新了**表4-1：“4线接口”**
- 更新了**表4-2：“2线接口”**
- 更新了**图4-4：“PIC32MZ EC/EF电源连接”**
- 更新了**图4-5：“PIC32MZ DA电源连接”**
- 更新了**表5-1：“代码存储区大小”**
- 更新了**图5-2：“基本PIC32编程接口框图”**
- 更新了**图16-1：“4线退出编程模式”**
- 更新了**图16-2：“2线退出编程模式”**
- 向**表 21-1：“交流 / 直流特性和时序要求”**中添加了参数D112（V_{DD1V8}）和D115（I_{DD1V8P}）
- 更新了**第4.3节“PIC32MX电源要求”**
- 添加了**第4.4节“具有V_{BAT}引脚的电源要求”**
- 添加了**第4.5节“PIC32MZ EC和PIC32MZ EF电源要求”**
- 添加了**第4.6节“PIC32MZ DA电源要求”**
- 添加了**第4.7节“PIC32MK电源要求”**
- 添加了**第5.3.3节“同步”**
- 添加了**第6.7节“Synchronize伪操作”**
- 更新了**第8.1节“4线接口”**
- 更新了**第8.2节“2线接口”**
- 更新了**表13-1：“页擦除操作码”**
- 更新了**表18-1：“器件配置寄存器的掩码值（目前支持的PIC32MX、PIC32MZ和PIC32MK器件）”**
- 更新了“校验和公式”中的注1（见公式18-1）

版本S（2016年9月）

此版本包含以下更新：

- 更新了“编程接口图”（见图4-1）
- 更新了“4线接口引脚表”（见表4-1）
- 更新了“2线接口引脚表”（见表4-2）
- 更新了“PIC32MZ DA电源连接”图（见图4-5）
- 更新了“基本PIC32编程接口框图”（见图5-2）
- 更新了**第5.3.2节“2相ICSP”**中的注
- 更新了“交流/直流特性和时序要求”（见表21-1）
- 添加了器件ID（见**附录C：“器件ID”**中的表C-1至表C-11）

版本T（2017年5月）

此版本包含以下更新：

- 更新了表20-1、表4-1和表4-2
- 更新了图4-1、图4-2和图5-2
- 添加了表C-12
- 对整篇文档的文字和格式进行了少量更新

版本U（2017年7月）

此版本包含以下更新：

- 更新了PIC32MK器件（见**第2.1节“具有双闪存分区和双引导区域的器件”**）
- 更新了“代码存储区大小”中的PIC32MK器件（见表5-1）
- 更新了“器件配置寄存器的掩码值（当前支持的PIC32MX、PIC32MZ和PIC32MK器件）”中的PIC32MK器件（见表18-1）
- 更新了“PIC32MK系列器件的配置字存储单元”（见表19-4）
- 添加了**表19-5：“PIC32MKXXXXXXH/G/J/K/L/MXX系列器件的配置字存储单元”**
- 更新了“PIC32MK通用和电机控制（GP/MC）系列器件ID”（见表C-11）
- 此外，还对整篇文档的文本和格式进行了少量更新

版本V（2018年7月）

此版本包含以下更新：

- 在表18-1中：将PIC32MK0256/0512XXG/H重命名为PIC32MK0256/0512XXH/G/J。此外，还更新了PIC32MK0512/1024XXK/L/M和PIC32MK0512/1024XXH/G/J的器件配置寄存器掩码值
- 在表19-5中：将系列名称从PIC32MKXXXXXXG/H/K/L/MXX重命名为PIC32MKXXXXXXH/G/J/K/L/MXX
- 在表C-11中：删除了对PIC32MKXXXXMCM/GPL/GPKXXX和PIC32MKXXXXMCH/GPGXXX的引用
- 在表C-12中：重新编排了PIC32MKXXXX/MCM/GPL/GPKXXX器件。添加了PIC32MKXXXXMCJ/GPH/GPGXXX器件
- 表C-1至表C-12中添加了对版本ID的注释引用，用于识别32位器件ID内的版本ID字段。

版本W（2018年10月）

此版本包含以下更新：

- 在表13-1和表14-1下添加了注释
- 表C-13的内容已移至表C-9

版本X（2019年11月）

此版本包含以下更新：

- 向**18.1“原理”**下添加了适用于PIC32MKXXXXGPD/GPE/MCFXXX器件的新注释
- 更新了**表18-1：“器件配置寄存器的掩码值（目前支持的PIC32MX、PIC32MZ和PIC32MKxxxxGPD/GPE/MCFxxx器件）”**中的PIC32MKXXXXGPD/GPE/MCFXXX器件信息
- 更新了**公式18-1：“校验和公式”**中的PIC32MKXXXXGPD/ GPE/MCFXXX器件信息

© 2014-2020 Microchip Technology Inc.

请注意以下有关 Microchip 器件代码保护功能的要点：

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

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

本出版物中提供的信息仅仅是为方便您使用 Microchip 产品或使用这些产品来进行设计。本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。

Microchip “按原样”提供这些信息。 Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担保，包括但不限于针对非侵权性、适销性和特定用途的适用性的暗示担保，或针对其使用情况、质量或性能的担保。

在任何情况下，对于因这些信息或使用这些信息而产生的任何间接的、特殊的、惩罚性的、偶然的或间接的损失、损害或任何类型的开销， Microchip 概不承担任何责任，即使 Microchip 已被告知可能发生损害或损害可以预见。在法律允许的最大范围内，对于因这些信息或使用这些信息而产生的所有索赔， Microchip 在任何情况下所承担的全部责任均不超出您为获得这些信息向 Microchip 直接支付的金额 （如有）。如果将 Microchip 器件用于生命维持和 / 或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切损害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任。除非另外声明，在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

有关 Microchip 质量管理体系的更多信息，请访问 www.microchip.com/quality。

商标

Microchip 的名称和徽标组合、 Microchip 徽标、 Adaptec、AnyRate、AVR、AVR 徽标、AVR Freaks、BesTime、BitCloud、chipKIT、 chipKIT 徽标、 CryptoMemory、 CryptoRF、 dsPIC、FlashFlex、 flexPWR、 HELDO、 IGLOO、 JukeBlox、 KeeLoq、Kleer、 LANCheck、 LinkMD、 maXStylus、 maXTouch、MediaLB、 megaAVR、 Microsemi、 Microsemi 徽标、 MOST、MOST 徽标、 MPLAB、 OptoLyzer、 PackeTime、 PIC、picoPower、 PICSTART、 PIC32 徽标、 PolarFire、 Prochip Designer、 QTouch、 SAM-BA、 SenGenuity、 SpyNIC、 SST、SST 徽标、 SuperFlash、 Symmetricom、 SyncServer、Tachyon、 TimeSource、 tinyAVR、 UNI/O、 Vectron 及 XMEGA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的注册商标。

AgileSwitch、 APT、 ClockWorks、 The Embedded Control Solutions Company、 EtherSynch、 FlashTec、 Hyper Speed Control、 HyperLight Load、 IntelliMOS、 Libero、 motorBench、mTouch、 Powermite 3、 Precision Edge、 ProASIC、 ProASIC Plus、 ProASIC Plus 徽标、 Quiet-Wire、 SmartFusion、SyncWorld、 Temux、 TimeCesium、 TimeHub、 TimePictra、TimeProvider、 WinPath 和 ZL 均为 Microchip Technology Incorporated 在美国的注册商标。

Adjacent Key Suppression、 AKS、 Analog-for-the-Digital Age、Any Capacitor、 AnyIn、 AnyOut、 Augmented Switching、BlueSky、 BodyCom、 CodeGuard、 CryptoAuthentication、CryptoAutomotive、 CryptoCompanion、 CryptoController、dsPICDEM、 dsPICDEM.net、 Dynamic Average Matching、DAM、 ECAN、 Espresso T1S、 EtherGREEN、 IdealBridge、 In-Circuit Serial Programming、 ICSP、 INICnet、 Intelligent Paralleling、 Inter-Chip Connectivity、 JitterBlocker、 maxCrypto、maxView、 memBrain、 Mindi、 MiWi、 MPASM、 MPF、 MPLAB Certified 徽标、 MPLIB、 MPLINK、 MultiTRAK、 NetDetach、Omniscient Code Generation、 PICDEM、 PICDEM.net、 PICkit、PICtail、 PowerSmart、 PureSilicon、 QMatrix、 REAL ICE、Ripple Blocker、 RTAX、 RTG4、 SAM-ICE、 Serial Quad I/O、simpleMAP、 SimpliPHY、 SmartBuffer、 SMART-I.S.、 storClad、SQI、 SuperSwitcher、 SuperSwitcher II、 Switchtec、SynchroPHY、 Total Endurance、 TSHARC、 USBCheck、VariSense、 VectorBlox、 VeriPHY、 ViewSpan、 WiperLock、XpressConnect 和 ZENA 均为 Microchip Technology Incorporated 在美国和其他国家或地区的商标。

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

Adaptec 徽标、 Frequency on Demand、 Silicon Storage Technology 和 Symmcom 均为 Microchip Technology Inc. 在除美国外的国家或地区的注册商标。

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

在此提及的所有其他商标均为各持有公司所有。

© 2014-2020, Microchip Technology Incorporated 版权所有。

ISBN：978-1-5224-7033-5

全球销售及服务网点

美洲

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

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

奥斯汀 Austin, TX
Tel: 1-512-257-3370

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

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

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

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

休斯敦 Houston, TX
Tel: 1-281-894-5983

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

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

罗利 Raleigh, NC
Tel: 1-919-844-7510

纽约 New York, NY
Tel: 1-631-435-6000

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

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

亚太地区

中国 - 北京
Tel: 86-10-8569-7000

中国 - 成都
Tel: 86-28-8665-5511

中国 - 重庆
Tel: 86-23-8980-9588

中国 - 东莞
Tel: 86-769-8702-9880

中国 - 广州
Tel: 86-20-8755-8029

中国 - 杭州
Tel: 86-571-8792-8115

中国 - 南京
Tel: 86-25-8473-2460

中国 - 青岛
Tel: 86-532-8502-7355

中国 - 上海
Tel: 86-21-3326-8000

中国 - 沈阳
Tel: 86-24-2334-2829

中国 - 深圳
Tel: 86-755-8864-2200

中国 - 苏州
Tel: 86-186-6233-1526

中国 - 武汉
Tel: 86-27-5980-5300

中国 - 西安
Tel: 86-29-8833-7252

中国 - 厦门
Tel: 86-592-238-8138

中国 - 香港特别行政区
Tel: 852-2943-5100

中国 - 珠海
Tel: 86-756-321-0040

台湾地区 - 高雄
Tel: 886-7-213-7830

台湾地区 - 台北
Tel: 886-2-2508-8600

台湾地区 - 新竹
Tel: 886-3-577-8366

亚太地区

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

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

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

印度 India - Pune
Tel: 91-20-4121-0141

日本 Japan - Osaka
Tel: 81-6-6152-7160

日本 Japan - Tokyo
Tel: 81-3-6880-3770

韩国 Korea - Daegu
Tel: 82-53-744-4301

韩国 Korea - Seoul
Tel: 82-2-554-7200

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

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

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

新加坡 Singapore
Tel: 65-6334-8870

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

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

欧洲

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

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

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

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

德国 Germany - Garching
Tel: 49-8931-9700

德国 Germany - Haan
Tel: 49-2129-3766400

德国 Germany - Heilbronn
Tel: 49-7131-72400

德国 Germany - Karlsruhe
Tel: 49-721-625370

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

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

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

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

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

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

挪威 Norway - Trondheim
Tel: 47-7288-4388

波兰 Poland - Warsaw
Tel: 48-22-3325737

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

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

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

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

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

© 2014-2020 Microchip Technology Inc.