PIC24FJ64GA104 系列
数据手册

采用 nanoWatt XLP 技术的
28/44 引脚 16 位
通用闪存单片机

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham 的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2002 认证。公司在 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

PIC24FJ64GA104 系列

采用 nanoWatt XLP 技术的 28/44 引脚 16 位通用闪存单片机

功耗管理模式：

- 可选择的功耗管理模式采用 nanoWatt XLP 技术实现极低功耗：
  - 深度休眠模式支持近乎完全掉电 （典型值为 20 nA，使能 RTCC 或 WDT 时为 500 nA），并可通过外部触发唤醒，或通过可编程 WDT 或 RTCC 闹钟自唤醒
  - 对于深度休眠模式使用极低功耗 DSBOR，对于所有其他模式使用 LPBOR
  - 休眠模式会关闭外设和内核，可显著降低功耗和快速唤醒
  - 空闲模式会关闭 CPU 和外设，可显著降低功耗（典型值低至 4.5 μA）
  - 打盹模式支持 CPU 时钟以低于外设的速度运行
  - 备用时钟模式支持在运行时切换为较低的时钟速度，可在运行模式期间选择性地降低功耗 （典型值低至 15 μA）

高性能 CPU：

- 改进型哈佛架构
- 32 MHz 时最高 16 MIPS 工作速度
- 8 MHz 内部振荡器具有：
  - 4 倍频 PLL 选项
  - 多个分频选项
- 17 位 x 17 位单周期硬件小数 / 整数乘法器
- 32 位 /16 位硬件除法器
- 16 x 16 位工作寄存器阵列
- 优化的 C 编译器指令集架构：
  - 76 条基本指令
  - 灵活的寻址模式
- 可寻址最大 12 MB 的线性程序存储空间
- 可寻址最大 64 KB 的线性数据存储空间
- 两个地址发生单元，分别用于数据存储器的读和写寻址

单片机特性：

- 工作电压范围为 2.0V 至 3.6V
- 可在软件控制下自编程
- 可承受 5.5V 输入电压 （仅对于数字引脚）
- 所有 I/O 引脚上的高灌 / 拉电流 （18 mA/18 mA）

单片机特性 （续）：

- 闪存程序存储器：
  - 最少可耐受 10,000 次擦 / 写
  - 最少 20 年数据保存时间
  - 可选择的写保护边界
- 故障保护时钟监视器操作：
  - 检测时钟故障并切换到片上 FRC 振荡器
- 片上 2.5V 稳压器
- 上电复位 （Power-on Reset， POR）、上电延时定时器（Power-up Timer， PWRT）和振荡器起振定时器（Oscillator Start-up Timer， OST）
- 两个灵活的看门狗定时器 （Watchdog Timer， WDT），用于可靠工作：
  - 标准可编程 WDT 用于正常工作模式
  - 极低功耗 WDT 用于深度休眠模式 （可编程周期为 2 ms 至 26 天）
- 在线串行编程（In-Circuit Serial Programming™，ICSP™）和通过两个引脚进行的在线调试（In-Circuit Debug，ICD）
- JTAG 边界扫描支持

模拟特性：

- 最多 13 路通道的 10 位模数（Analog-to-Digital，A/D）转换器：
  - 转换速率为 500 ksps
  - 在休眠和空闲期间可以进行转换
- 3 个具有可编程输入 / 输出配置的模拟比较器
- 充电时间测量单元 （Charge Time Measurement Unit，CTMU）：
  - 支持触摸屏和电容式开关的电容触摸传感
  - 提供高分辨率的时间测量和简单的温度检测

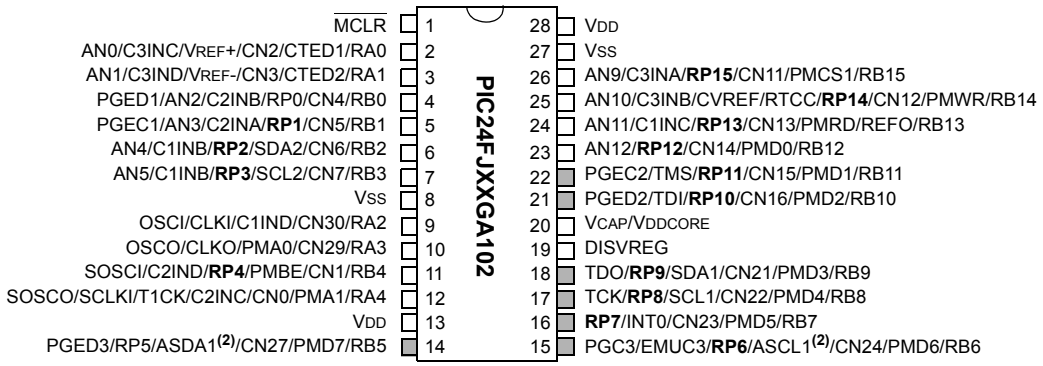
PIC24FJ 系列器件	引脚	程序存储器（字节数）	SRAM（字节数）	可重映射的外设						I²C™	10 位 A/D（通道数）	比较器	PMP/PSP	RTCC	CTMU
				可重映射的引脚	16 位定时器	捕捉输入	比较 /PWM 输出	带 IrDA® 的 UART	SPI						
32GA102	28	32K	8K	16	5	5	5	2	2	2	10	3	有	有	有
64GA102	28	64K	8K	16	5	5	5	2	2	2	10	3	有	有	有
32GA104	44	32K	8K	26	5	5	5	2	2	2	13	3	有	有	有
64GA104	44	64K	8K	26	5	5	5	2	2	2	13	3	有	有	有

外设特性：

- 外设引脚选择：
  - 允许许多外设进行独立的 I/O 映射
  - 最多 26 个可用引脚 （44 引脚器件）
  - 连续的硬件完整性检查和安全互锁以防止无意中更改配置
- 8 位并行主端口 （Parallel Master Port， PMP）：
  - 最多 16 位复用寻址，在 44 引脚器件上最多具有 11 个专用地址引脚
  - 控制线上的可编程极性
  - 支持传统并行从端口
- 硬件实时时钟/日历（Real-Time Clock/Calendar，RTCC）：
  - 提供时钟、日历和闹钟功能
  - 即使处于深度休眠模式也可工作
- 两个带有8级FIFO缓冲区的3线/4线SPI模块（支持4帧模式）
- 两个 I²C™ 模块，支持多主器件 / 从模式和 7 位 /10 位寻址
- 两个 UART 模块：
  - 支持 RS-485、 RS-232 和 LIN/J2602
  - 用于 IrDA® 的片上硬件编码器 / 解码器
  - 接收到启动位时自动唤醒
  - 自动波特率检测 （Auto-Baud Detect， ABD）
  - 4 级深 FIFO 缓冲区
- 5 个带有可编程预分频器的 16 位定时器 / 计数器
- 5 个 16 位捕捉输入，每个均具有一个专用时基
- 5 个 16 位比较 /PWM 输出，每个均具有一个专用时基
- 32 位可编程循环冗余校验（Cyclic Redundancy Check，CRC）发生器
- 数字 I/O 引脚上的可配置漏极开路输出
- 最多 3 个外部中断源

引脚图

28 引脚 SPDIP 和 SOIC(1)



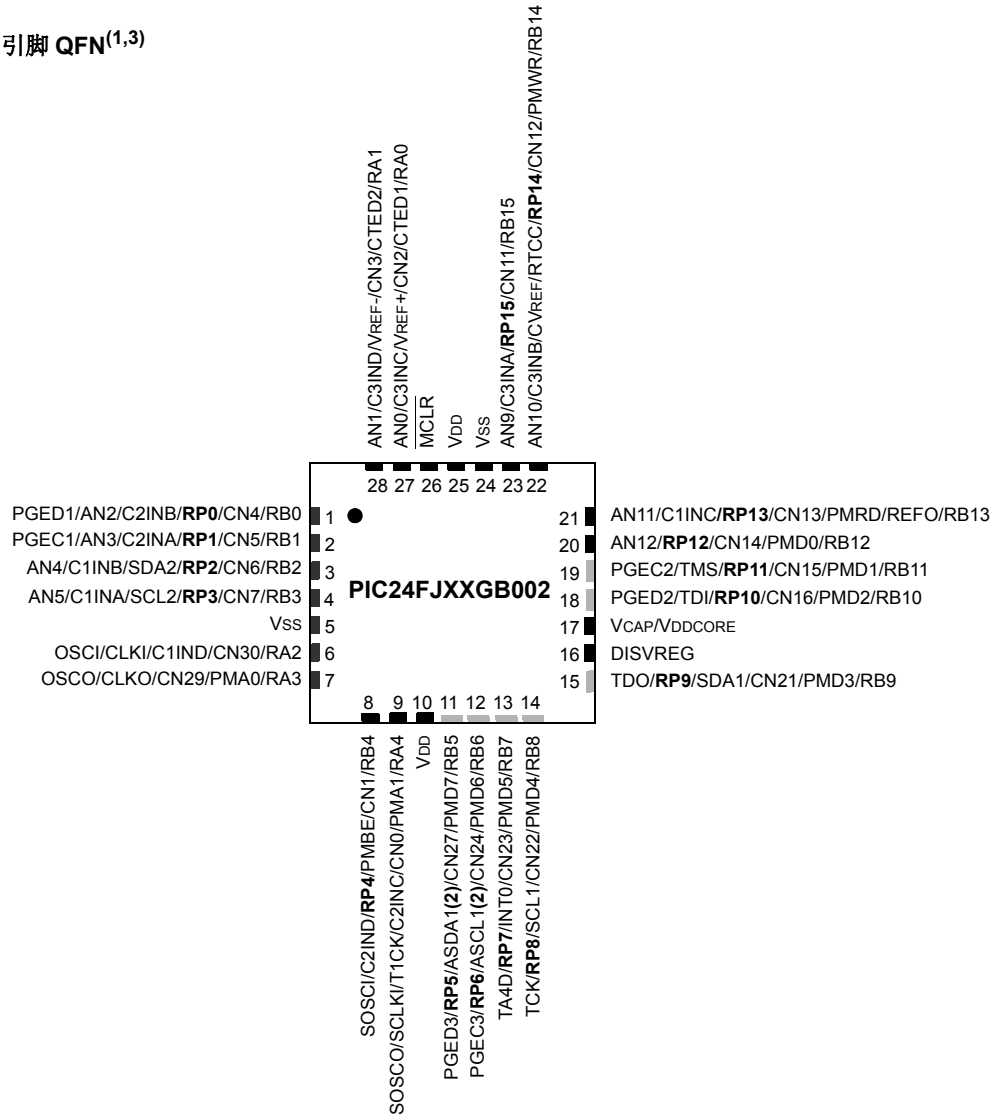
图注： **RPn** 表示可重映射的外设引脚。

注 1： 灰色阴影表示可承受 5.5V 的输入引脚。

2： 当 I2C1SEL 位置 1 时，相应的引脚分别用作 SDA1 和 SCL1。

引脚图

28 引脚 QFN(1,3)



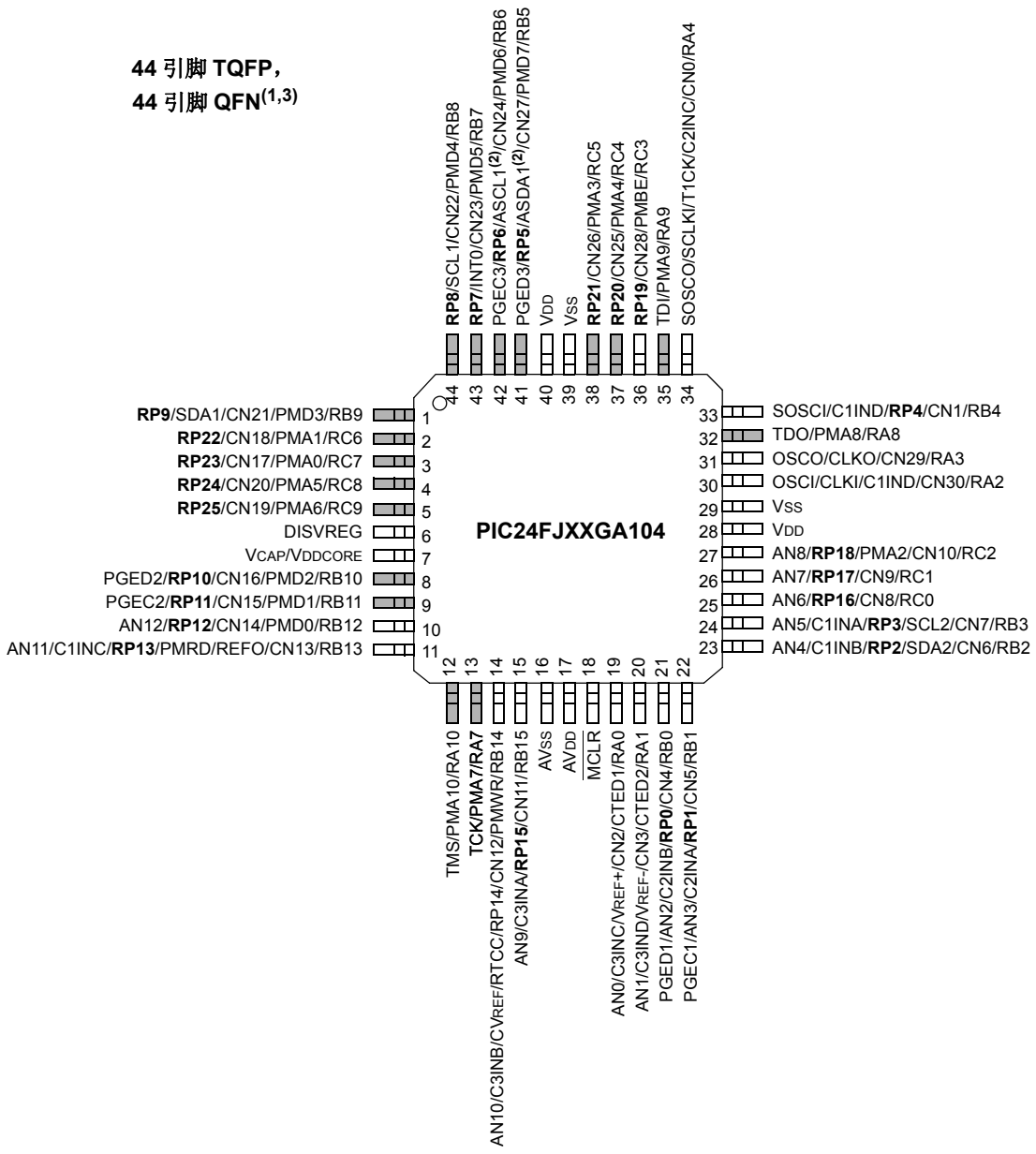
图注： **RPn** 表示可重映射的外设引脚。

注 1： 灰色阴影表示可承受 5.5V 的输入引脚。

2： 当 I2C1SEL 位置 1 时，相应的引脚分别用作 SDA1 和 SCL1。

3： QFN 器件上的底板应连接到 Vss。

引脚图



图注： **RPn** 表示可重映射的外设引脚。

注 1： 灰色阴影表示可承受 5.5V 的输入引脚。

2： 当 I2C1SEL 位置 1 时，相应的引脚分别用作 SDA1 和 SCL1。

3： QFN 器件上的底板应连接到 Vss。

目录

致客户

我们旨在提供最佳文档供客户正确使用 Microchip 产品。为此，我们将不断改进出版物的内容和质量，使之更好地满足您的要求。出版物的质量将随新文档及更新版本的推出而得到提升。

如果您对本出版物有任何问题和建议，请通过电子邮件联系我公司 TRC 经理，电子邮件地址为 **CTRC@microchip.com**，或将本数据手册后附的 **《读者反馈表》**传真到 86-21-5407 5066。我们期待您的反馈。

最新数据手册

欲获得本数据手册的最新版本，请查询我公司的网站：

http://www.microchip.com

查看数据手册中任意一页下边角处的文献编号即可确定其版本。文献编号中数字串后的字母是版本号，例如：DS30000A 是 DS30000 的 A 版本。

勘误表

现有器件可能带有一份勘误表，描述了实际运行与数据手册中记载内容之间存在的细微差异以及建议的变通方法。一旦我们了解到器件 / 文档存在某些差异时，就会发布勘误表。勘误表上将注明其所适用的硅片版本和文件版本。

欲了解某一器件是否存在勘误表，请通过以下方式之一查询：

- Microchip 网站 http://www.microchip.com
- 当地 Microchip 销售办事处 （见最后一页）

在联络销售办事处时，请说明您所使用的器件型号、硅片版本和数据手册版本 （包括文献编号）。

客户通知系统

欲及时获知 Microchip 产品的最新信息，请到我公司网站 **www.microchip.com** 上注册。

1.0 器件概述

本文档包含以下器件的具体信息：

- PIC24FJ32GA102
- PIC24FJ32GA104
- PIC24FJ64GA102
- PIC24FJ64GA104

PIC24FJ64GA104 系列器件提供了扩展的外设功能部件，并为那些 8 位平台无法满足其需求但又不要求使用数字信号处理器的高性能应用提供了一种新的选择。

1.1 内核特性

1.1.1 16 位架构

所有 PIC24F 器件的核心都是 16 位改进型哈佛架构，第一次引入该架构的就是 Microchip 的 dsPIC® 数字信号控制器。 PIC24F CPU 内核提供了众多增强功能，例如：

- 16位数据路径和24位地址路径，能在数据空间和存储空间之间传送信息
- 最大 12 MB（程序空间）和 64 KB（数据空间）的线性寻址
- 16 个寄存器组成的工作寄存器阵列，支持内置软件堆栈
- 17 位 x 17 位硬件乘法器，支持整数数学运算
- 硬件支持 32 位 /16 位的除法运算
- 指令集支持多种寻址模式，并针对高级语言（如 C 语言）进行了优化
- 工作性能最高可达 16 MIPS

1.1.2 节能技术

PIC24FJ64GA104 系列中的所有器件都具有一系列能在工作时显著降低功耗的功能。主要包括以下几项：

- **动态时钟切换：**在器件工作过程中，器件时钟可在软件控制下切换为 Timer1 时钟源或内部低功耗 RC 振荡器，允许用户将节能理念融入到软件设计中去。
- **打盹模式操作：**当那些对时序敏感的应用（如串行通信）要求外设不间断地工作时，可有选择地降低 CPU 时钟速度，从而可在不丢失时钟的前提下进一步节约功耗。
- **基于指令的节能模式：**有三种基于指令的节能模式：
  - 空闲模式——内核关闭，而外设仍然工作。
  - 休眠模式——内核和需要使用系统时钟的外设关闭，而使用自身时钟或由其他器件提供时钟的外设仍然工作。
  - 深度休眠模式——内核、外设（RTCC 和 DSWDT 除外）、闪存和 SRAM 均关闭，以最大程度节约电流来延长便携式应用的电池寿命。

1.1.3 振荡器选项和特性

PIC24FJ64GA104 系列中的所有器件均提供 5 个不同的振荡器选项，使用户在开发应用硬件时有很大的选择范围。这些选项包括：

- 两种晶振模式，使用晶振或陶瓷谐振器。
- 两种外部时钟模式，提供 2 分频时钟输出选项。
- 一个标称输出值为 8 MHz 的快速内部振荡器（Fast Internal Oscillator， FRC），可在软件控制下被分频，从而使时钟速度可低至 31 kHz。
- 一个锁相环 （Phase Lock Loop， PLL）倍频器，可在外部振荡器模式和采用 FRC 振荡器的情况下使用，从而使时钟速度最高可达 32 MHz。
- 具有固定 31 kHz 输出的独立低功耗内部 RC 振荡器（LPRC），可为对时序不敏感的应用提供低功耗时钟选项。

内部振荡器模块还为故障保护时钟监视器提供了一个稳定的参考源。故障保护时钟监视器不断地监视主时钟源，将之与内部振荡器提供的参考信号作比较。一旦发生时钟故障，允许控制器将时钟源切换到内部振荡器，继续保持低速工作或安全地关闭应用。

1.1.4 易于移植

无论存储器容量如何，所有器件均共享同一组丰富外设，使得应用程序可在升级时很方便地移植。整个系列使用相同的引脚配置方案也有助于向更大型器件的移植。

PIC24F 系列器件的引脚同 dsPIC33 系列器件的引脚是兼容的，并与 PIC18 和 dsPIC30 器件的引脚配置方案部分兼容。这样全部采用 Microchip 器件，就可将应用从相对简单的功能顺利扩展到强大和复杂的功能。

1.2 其他特性

- **外设引脚选择：**外设引脚选择功能允许大部分的数字外设被映射到一组固定的数字 I/O 引脚。用户可独立地将许多数字外设之一的输入和 / 或输出映射到其中的任一 I/O 引脚。
- **通信：**PIC24FJ64GA104 系列集成了一些串行通信外设以满足一系列的应用要求。有两个独立的I^2C™模块支持主模式和从模式下的操作。通过外设引脚选择（Peripheral Pin Select， PPS）功能，器件还具有两个带内置IrDA®编码器/解码器的独立UART以及两个 SPI 模块。
- **模拟特性：**PIC24FJ64GA104 系列的所有成员都包括一个10位A/D转换器模块和一个三比较器模块。A/D 模块实现了可编程采集时间，允许选择通道立即开始转换而无需等待采样周期结束，同时也提高了采样速度。比较器模块包括 3 个模拟比较器，它们可配置为多种工作模式。
- **CTMU 接口：**该模块为精确时间测量和脉冲产生提供了一种便捷的方法，同时也可以作为电容传感器的接口。
- **并行主 / 增强型并行从端口：**可以将一个通用 I/O 端口重新配置为用于增强型并行数据通信。在该模式下，可以将端口配置为工作在主模式或从模式下。在主模式下支持 8 位和 16 位数据传输，并具有最多 12 条外部地址线。
- **实时时钟 / 日历：**该模块通过硬件实现了带有闹钟功能的全功能时钟和日历，从而释放了定时器资源和程序存储空间供核心应用使用。

1.3 系列中各器件的详细说明

PIC24FJ64GA104 系列中的器件有 28 引脚和 44 引脚两种封装形式。图 1-1 给出了所有器件的一般框图。

这些器件在以下几个方面存在差异：

- 闪存程序存储器：
  - PIC24FJ32GA1 器件——32 KB
  - PIC24FJ64GA1 器件——64 KB
- 可用的 I/O 引脚和端口数：
  - 28 引脚器件——2 个端口 21 个引脚
  - 44 引脚器件——3 个端口 35 个引脚
- 可用的电平变化中断通知（Interrupt-on-Change Notification， ICN）输入：
  - 28 引脚器件——21
  - 44 引脚器件——31
- 可用的可重映射引脚数：
  - 28 引脚器件——16 个引脚
  - 44 引脚器件——26 个引脚
- 可用的 PMP 地址引脚数：
  - 28 引脚器件——3 个引脚
  - 44 引脚器件——12 个引脚
- 可用的 A/D 输入通道数：
  - 28 引脚器件——10 个引脚
  - 44 引脚器件——13 个引脚

本系列器件的所有其他功能都是相同的。表 1-1 汇总了这些功能。

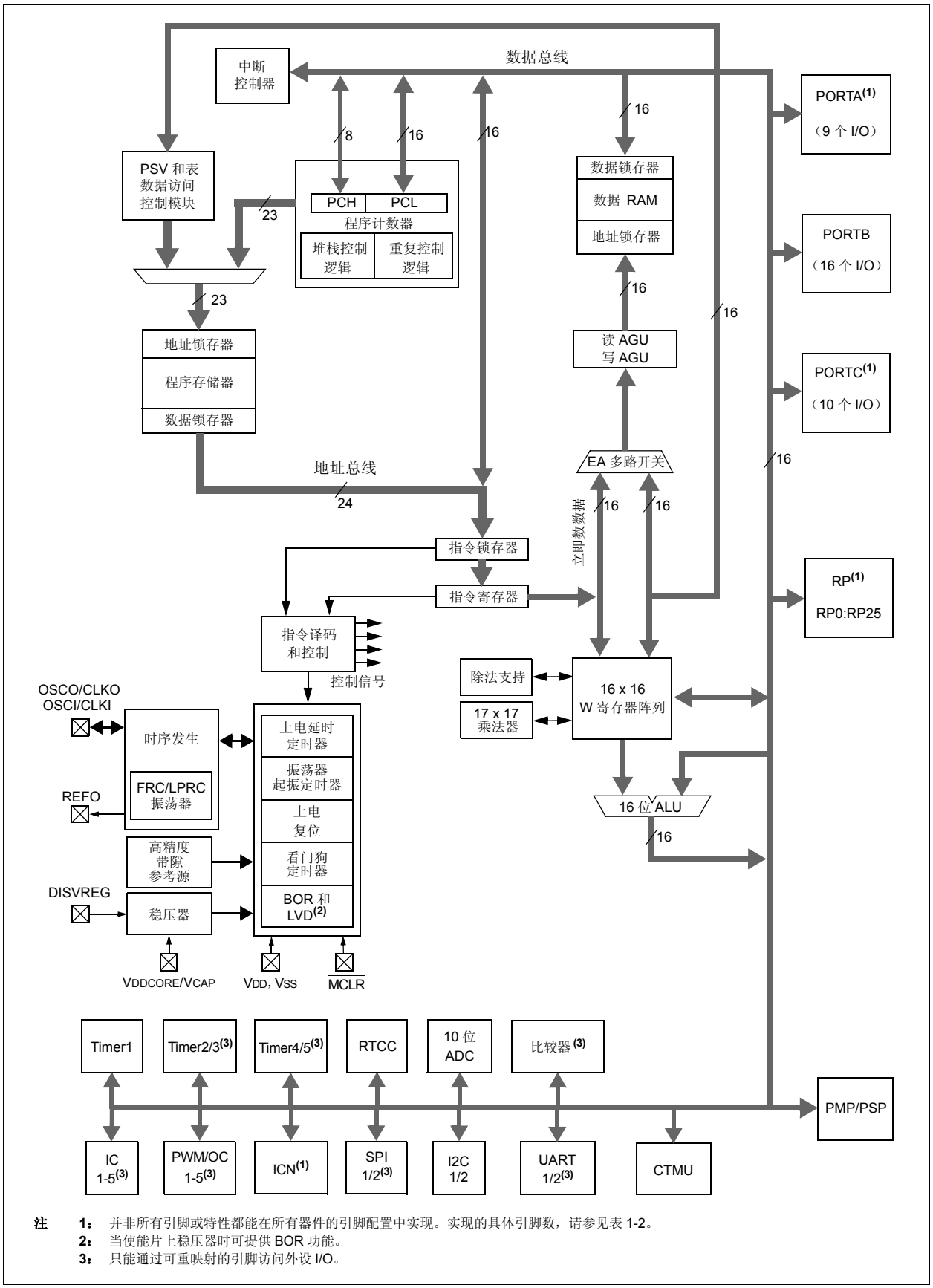
PIC24FJ64GA104 系列器件上可用的引脚功能列表如表 1-2 所示，按功能名称排序。注意该表只显示了各个外设功能所使用的引脚，而没有显示同一引脚上多种功能的复用方式。在本数据手册开始部分的引脚图中提供了相关信息。复用的功能按功能的优先级排列，最前面的是优先级最高的外设功能。

表 1-1： PIC24FJ64GA104 系列的器件特性

特性	PIC24FJ32GA102	PIC24FJ64GA102	PIC24FJ32GA104	PIC24FJ64GA104
工作频率	DC – 32 MHz			
程序存储器 （字节）	32K	64K	32K	64K
程序存储器 （指令）	11,008	22,016	11,008	22,016
数据存储器 （字节）	8,192			
中断源 （软向量 /NMI 陷阱）	45 （41/4）			
I/O 端口	端口 A 和 B		端口 A、 B 和 C	
I/O 引脚总数	21		35	
可重映射的引脚	16		26	
定时器：				
总数 （16 位）	5(1)			
32 位 （由一对 16 位定时器组成）	2			
输入捕捉通道	5(1)			
输出比较 /PWM 通道	5(1)			
输入电平变化通知中断	21		31	
串行通信：				
UART	2(1)			
SPI （3 线 /4 线）	2(1)			
I^2C™	2			
并行通信 （PMP/PSP）	有			
JTAG 边界扫描	有			
10 位模数转换模块 （输入通道）	10		13	
模拟比较器	3			
CTMU 接口	有			
复位 （和延时）	POR、 BOR、 RESET 指令、 $\overline{\text{MCLR}}$、 WDT ； 非法操作码、 REPEAT 指令、硬件陷阱和配置字不匹配 （PWRT、 OST 和 PLL 锁定）			
指令集	76 条基本指令，多种寻址模式			
封装	28 引脚 QFN、 SOIC 和 SPDIP		44 引脚 QFN 和 TQFP	

注 1： 可通过可重映射的引脚访问外设。

图 1-1：　　PIC24FJ64GA104 系列一般框图



注　**1：** 并非所有引脚或特性都能在所有器件的引脚配置中实现。实现的具体引脚数，请参见表 1-2。
2： 当使能片上稳压器时可提供 BOR 功能。
3： 只能通过可重映射的引脚访问外设 I/O。

表 1-2： PIC24FJ64GA104 系列引脚说明

功能	引脚编号			I/O	输入缓冲器	说明
	28 引脚 SPDIP/ SOIC	28 引脚 QFN	44 引脚 QFN/ TQFP			
AN0	2	27	19	I	ANA	A/D 模拟输入。
AN1	3	28	20	I	ANA	
AN2	4	1	21	I	ANA	
AN3	5	2	22	I	ANA	
AN4	6	3	23	I	ANA	
AN5	7	4	24	I	ANA	
AN6	—	—	25	I	ANA	
AN7	—	—	26	I	ANA	
AN8	—	—	27	I	ANA	
AN9	26	23	15	I	ANA	
AN10	25	22	14	I	ANA	
AN11	24	21	11	I	ANA	
AN12	23	20	10	I	ANA	
ASCL1	15	12	42	I/O	I^2C	备用 I2C1 同步串行时钟输入 / 输出。
ASDA1	14	11	41	I/O	I^2C	备用 I2C1 同步串行数据输入 / 输出。
AV_{DD}	—	—	17	P	—	模拟模块的正电源。
AV_{SS}	—	—	16	P	—	模拟模块的参考地。
C1INA	7	4	24	I	ANA	比较器 1 的输入 A。
C1INB	6	3	23	I	ANA	比较器 1 的输入 B。
C1INC	24	21	11	I	ANA	比较器 1 的输入 C。
C1IND	9	6	30	I	ANA	比较器 1 的输入 D。
C2INA	5	2	22	I	ANA	比较器 2 的输入 A。
C2INB	4	1	21	I	ANA	比较器 2 的输入 B。
C2INC	12	9	34	I	ANA	比较器 2 的输入 C。
C2IND	11	8	33	I	ANA	比较器 2 的输入 D。
C3INA	26	23	15	I	ANA	比较器 3 的输入 A。
C3INB	25	22	14	I	ANA	比较器 3 的输入 B。
C3INC	2	27	19	I	ANA	比较器 3 的输入 C。
C3IND	3	28	20	I	ANA	比较器 3 的输入 D。
CLKI	9	6	30	I	ANA	主时钟输入连接。
CLKO	10	7	31	O	—	系统时钟输出。

图注： TTL = TTL 输入缓冲器　　ST = 施密特触发器输入缓冲器
ANA = 模拟电平输入 / 输出　　I^2C™ = I^2C/SMBus 输入缓冲器

表 1-2： PIC24FJ64GA104 系列引脚说明 （续）

功能	引脚编号			I/O	输入缓冲器	说明
	28 引脚 SPDIP/ SOIC	28 引脚 QFN	44 引脚 QFN/ TQFP			
CN0	12	9	34	I	ST	电平变化中断输入。
CN1	11	8	33	I	ST	
CN2	2	27	19	I	ST	
CN3	3	28	20	I	ST	
CN4	4	1	21	I	ST	
CN5	5	2	22	I	ST	
CN6	6	3	23	I	ST	
CN7	7	4	24	I	ST	
CN8	—	—	25	I	ST	
CN9	—	—	26	I	ST	
CN10	—	—	27	I	ST	
CN11	26	23	15	I	ST	
CN12	25	22	14	I	ST	
CN13	24	21	11	I	ST	
CN14	23	20	10	I	ST	
CN15	22	19	9	I	ST	
CN16	21	18	8	I	ST	
CN17	—	—	3	I	ST	
CN18	—	—	2	I	ST	
CN19	—	—	5	I	ST	
CN20	—	—	4	I	ST	
CN21	18	15	1	I	ST	
CN22	17	14	44	I	ST	
CN23	16	13	43	I	ST	
CN24	15	12	42	I	ST	
CN25	—	—	37	I	ST	
CN26	—	—	38	I	ST	
CN27	14	11	41	I	ST	
CN28	—	—	36	I	ST	
CN29	10	7	31	I	ST	
CN30	9	6	30	I	ST	
CTED1	2	27	19	I	ANA	CTMU 外部边沿输入 1。
CTED2	3	28	20	I	ANA	CTMU 外部边沿输入 2。
CVREF	25	22	14	O	—	比较器参考电压输出。
DISVREG	19	16	6	I	ST	稳压器禁止。

图注： TTL = TTL 输入缓冲器　　ST = 施密特触发器输入缓冲器
ANA = 模拟电平输入 / 输出　　I^2C™ = I^2C/SMBus 输入缓冲器

表 1-2： PIC24FJ64GA104 系列引脚说明 （续）

功能	引脚编号			I/O	输入缓冲器	说明
	28 引脚 SPDIP/ SOIC	28 引脚 QFN	44 引脚 QFN/ TQFP			
INT0	16	13	43	I	ST	外部中断输入。
MCLR	1	26	18	I	ST	主复位 （器件复位）输入。此线变为低电平，引起复位。
OSCI	9	6	30	I	ANA	主振荡器输入连接。
OSCO	10	7	31	O	ANA	主振荡器输出连接。
PGEC1	5	2	22	I/O	ST	在线调试器 / 仿真器 /ICSP™ 编程时钟。
PGED1	4	1	21	I/O	ST	在线调试器 / 仿真器 /ICSP 编程数据。
PGEC2	22	19	9	I/O	ST	在线调试器 / 仿真器 /ICSP 编程时钟。
PGED2	21	18	8	I/O	ST	在线调试器 / 仿真器 /ICSP 编程数据。
PGEC3	15	12	42	I/O	ST	在线调试器 / 仿真器 /ICSP 编程时钟。
PGED3	14	11	41	I/O	ST	在线调试器 / 仿真器 /ICSP 编程数据。
PMA0	10	7	3	I/O	ST	并行主端口地址 bit 0 的输入 （带缓冲的从模式）和输出 （主模式）。
PMA1	12	9	2	I/O	ST	并行主端口地址 bit 1 的输入 （带缓冲的从模式）和输出 （主模式）。
PMA2	—	—	27	O	—	并行主端口地址 （解复用的主模式）。
PMA3	—	—	38	O	—	
PMA4	—	—	37	O	—	
PMA5	—	—	4	O	—	
PMA6	—	—	5	O	—	
PMA7	—	—	13	O	—	
PMA8	—	—	32	O	—	
PMA9	—	—	35	O	—	
PMA10	—	—	12	O	—	
PMCS1	26	23	15	I/O	ST/TTL	并行主端口片选 1 选通 / 地址 bit 15。
PMBE	11	8	36	O	—	并行主端口字节使能选通。
PMD0	23	20	10	I/O	ST/TTL	并行主端口数据 （解复用的主模式）或地址 / 数据 （复用的主模式）。
PMD1	22	19	9	I/O	ST/TTL	
PMD2	21	18	8	I/O	ST/TTL	
PMD3	18	15	1	I/O	ST/TTL	
PMD4	17	14	44	I/O	ST/TTL	
PMD5	16	13	43	I/O	ST/TTL	
PMD6	15	12	42	I/O	ST/TTL	
PMD7	14	11	41	I/O	ST/TTL	
PMRD	24	21	11	O	—	并行主端口读选通。
PMWR	25	22	14	O	—	并行主端口写选通。

图注： TTL = TTL 输入缓冲器　　ST = 施密特触发器输入缓冲器
ANA = 模拟电平输入 / 输出　　I^2C™ = I^2C/SMBus 输入缓冲器

表 1-2： PIC24FJ64GA104 系列引脚说明 （续）

功能	引脚编号			I/O	输入缓冲器	说明
	28 引脚 SPDIP/ SOIC	28 引脚 QFN	44 引脚 QFN/ TQFP			
RA0	2	27	19	I/O	ST	PORTA 数字 I/O。
RA1	3	28	20	I/O	ST	
RA2	9	6	30	I/O	ST	
RA3	10	7	31	I/O	ST	
RA4	12	9	34	I/O	ST	
RA7	—	—	13	I/O	ST	
RA8	—	—	32	I/O	ST	
RA9	—	—	35	I/O	ST	
RA10	—	—	12	I/O	ST	
RB0	4	1	21	I/O	ST	PORTB 数字 I/O。
RB1	5	2	22	I/O	ST	
RB2	6	3	23	I/O	ST	
RB3	7	4	24	I/O	ST	
RB4	11	8	33	I/O	ST	
RB5	14	11	41	I/O	ST	
RB6	15	12	42	I/O	ST	
RB7	16	13	43	I/O	ST	
RB8	17	14	44	I/O	ST	
RB9	18	15	1	I/O	ST	
RB10	21	18	8	I/O	ST	
RB11	22	19	9	I/O	ST	
RB12	23	20	10	I/O	ST	
RB13	24	21	11	I/O	ST	
RB14	25	22	14	I/O	ST	
RB15	26	23	15	I/O	ST	
RC0	—	—	25	I/O	ST	PORTC 数字 I/O。
RC1	—	—	26	I/O	ST	
RC2	—	—	27	I/O	ST	
RC3	—	—	36	I/O	ST	
RC4	—	—	37	I/O	ST	
RC5	—	—	38	I/O	ST	
RC6	—	—	2	I/O	ST	
RC7	—	—	3	I/O	ST	
RC8	—	—	4	I/O	ST	
RC9	—	—	5	I/O	ST	
REFO	24	21	11	O	—	参考时钟输出。

图注： TTL = TTL 输入缓冲器　　ST = 施密特触发器输入缓冲器
ANA = 模拟电平输入 / 输出　　I^2C™ = I^2C/SMBus 输入缓冲器

表 1-2： PIC24FJ64GA104 系列引脚说明 （续）

功能	引脚编号			I/O	输入缓冲器	说明
	28 引脚 SPDIP/ SOIC	28 引脚 QFN	44 引脚 QFN/ TQFP			
RP0	4	1	21	I/O	ST	可重映射的外设 （输入或输出）。
RP1	5	2	22	I/O	ST	
RP2	6	3	23	I/O	ST	
RP3	7	4	24	I/O	ST	
RP4	11	8	33	I/O	ST	
RP5	2	27	19	I/O	ST	
RP6	3	28	20	I/O	ST	
RP7	16	13	43	I/O	ST	
RP8	17	14	44	I/O	ST	
RP9	18	15	1	I/O	ST	
RP10	21	18	8	I/O	ST	
RP11	22	19	9	I/O	ST	
RP12	23	20	10	I/O	ST	
RP13	24	21	11	I/O	ST	
RP14	25	22	14	I/O	ST	
RP15	26	23	15	I/O	ST	
RP16	—	—	25	I/O	ST	
RP17	—	—	26	I/O	ST	
RP18	—	—	27	I/O	ST	
RP19	—	—	36	I/O	ST	
RP20	—	—	37	I/O	ST	
RP21	—	—	38	I/O	ST	
RP22	—	—	2	I/O	ST	
RP23	—	—	3	I/O	ST	
RP24	—	—	4	I/O	ST	
RP25	—	—	5	I/O	ST	
RTCC	25	22	14	O	—	实时时钟闹钟 / 秒脉冲输出。
SCL1	17	14	44	I/O	I^2C	I2C1 同步串行时钟输入 / 输出。
SCL2	7	4	24	I/O	I^2C	I2C2 同步串行时钟输入 / 输出。
SDA1	18	15	1	I/O	I^2C	I2C1 数据输入 / 输出。
SDA2	6	3	23	I/O	I^2C	I2C2 数据输入 / 输出。
SOSCI	11	8	33	I	ANA	辅助振荡器 /Timer1 时钟输入。
SOSCO	12	9	34	O	ANA	辅助振荡器 /Timer1 时钟输出。
T1CK	12	9	34	I	ST	Timer1 时钟输入。
TCK	17	14	13	I	ST	JTAG 测试时钟 / 编程时钟输入。
TDI	16	13	35	I	ST	JTAG 测试数据 / 编程数据输入。
TDO	18	15	32	O	—	JTAG 测试数据输出。
TMS	14	11	12	I	ST	JTAG 测试模式选择输入。

图注： TTL = TTL 输入缓冲器　　ST = 施密特触发器输入缓冲器
ANA = 模拟电平输入 / 输出　　I^2C™ = I^2C/SMBus 输入缓冲器

表 1-2： PIC24FJ64GA104 系列引脚说明 （续）

功能	引脚编号			I/O	输入缓冲器	说明
	28 引脚 SPDIP/ SOIC	28 引脚 QFN	44 引脚 QFN/ TQFP			
VCAP	20	17	7	P	—	外部滤波电容连接 （稳压器使能）。
VDD	13, 28	10, 25	28, 40	P	—	外设数字逻辑和 I/O 引脚的正电源。
VDDCORE	20	17	7	P	—	单片机内核逻辑的正电源 （稳压器禁止）。
VREF-	3	28	20	I	ANA	A/D 和比较器参考电压 （低电压）输入。
VREF+	2	27	19	I	ANA	A/D 和比较器参考电压 （高电压）输入。
VSS	8, 27	5, 24	29, 39	P	—	逻辑和 I/O 引脚的参考地。

图注： TTL = TTL 输入缓冲器　　ST = 施密特触发器输入缓冲器
ANA = 模拟电平输入 / 输出　　I^2C™ = I^2C/SMBus 输入缓冲器

2.0 16 位单片机入门指南

2.1 基本连接要求

在开始使用 PIC24FJ64GA104 系列 16 位单片机进行开发之前，需要注意最低限度的器件引脚连接要求。

必须始终连接以下引脚：

- 所有 VDD 和 VSS 引脚（见**第 2.2 节“电源引脚”**）
- 所有 AVDD 和 AVSS 引脚（不论是否使用模拟器件功能）（见**第 2.2 节“电源引脚”**）
- MCLR 引脚（见**第 2.3 节“主复位（MCLR）引脚”**）
- ENVREG/DISVREG 和 VCAP/VDDCORE 引脚（仅限 PIC24FJ 器件）（见**第 2.4 节“稳压器引脚（ENVREG/DISVREG 和 VCAP/VDDCORE）”**）

如果在最终应用中使用了以下引脚，则也必须连接它们：

- PGECx/PGEDx 引脚，用于进行在线串行编程（ICSP™）和调试（见**第 2.5 节“ICSP 引脚”**）
- OSCI 和 OSCO 引脚（使用外部振荡器源时）（见**第 2.6 节“外部振荡器引脚”**）

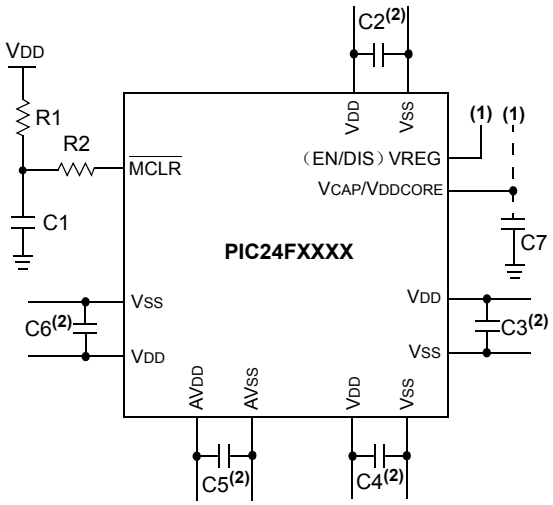
此外，可能还需要连接以下引脚：

- VREF+/VREF- 引脚（在实现模拟模块的外部参考电压时使用）

注：	不论是否使用任何模拟模块，都必须始终连接 AVDD 和 AVSS 引脚。

图 2-1 中显示了最低限度的连接要求。

图 2-1： 建议的最低限度连接



关键（所有值均为建议值）：

C1 至 C6：0.1 μF， 20V 陶瓷电容

C7：10 μF， 16V 钽电容或陶瓷电容

R1：10 kΩ

R2：100Ω 至 470Ω

注 1： 关于 ENVREG/DISVREG 引脚连接的说明，请参见**第2.4节“稳压器引脚（ENVREG/DISVREG 和 VCAP/VDDCORE）”**。

2： 所给出的示例针对的是具有 5 个 VDD/VSS 和 AVDD/AVSS 引脚对的 PIC24F 器件。其他器件的引脚对可能增多或减少，请相应地调整去耦电容的数量。

2.2 电源引脚

2.2.1 去耦电容

在每对电源引脚（例如，VDD、VSS、AVDD 和 AVSS）上，需要使用去耦电容。

使用去耦电容时，需要考虑以下标准：

- **电容的类型和电容值：**建议使用参数为 0.1 μF（100 nF）、 10-20V 的电容。电容应为低 ESR 元件，谐振频率处于 200 MHz 和更高范围。建议使用陶瓷电容。
- **在印制电路板上的布置：**去耦电容应尽可能靠近引脚。建议将电容与器件放置在电路板的同一面。如果空间受到限制，可以使用过孔将电容放置在 PCB 的另一层，但请确保从引脚到电容的走线长度不超出 0.25 英寸 （6 毫米）。
- **高频噪声处理：**如果电路板遇到高频噪声（频率高于数十 MHz），则另外添加一个陶瓷电容，与上述去耦电容并联。第二个电容的电容值可以介于 0.001 μF 至 0.01 μF 之间。请将第二个电容放置在靠近每个主去耦电容的位置。在高速电路设计中，需要考虑让 10 对电容尽可能靠近电源和接地引脚（例如， 0.1 μF 电容与 0.001 μF 电容并联）。
- **最大程度提高性能：**对于从电源电路开始的电路板布线，需要将电源和回路走线先连接到去耦电容，然后再与器件引脚连接。这可以确保去耦电容是电源链中的第一个元件。同等重要的是尽可能减小电容和电源引脚之间的走线长度，从而降低 PCB 走线电感。

2.2.2 槽路电容

对于电源走线长度超出 6 英寸的电路板，建议对集成电路（包括单片机）使用槽路电容来提供本地电源。槽路电容的电容值应根据连接电源与器件的走线电阻和应用中的器件的最大电流确定。也就是说，选择的槽路电容需要能够承受器件电压骤降的情况。典型值的范围为 4.7 μF 至 47 μF。

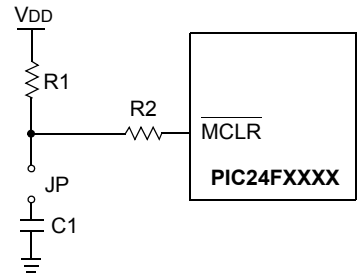
2.3 主复位（$\overline{\text{MCLR}}$）引脚

$\overline{\text{MCLR}}$ 引脚提供两种特定的器件功能：器件复位，以及器件编程和调试。如果最终应用中不需要进行编程和调试，则只需直接连接 VDD 即可。添加其他元件来帮助提高应用抵抗由于电压骤降导致误复位的能力，可能会是有好处的。图 2-1 给出了一种典型配置。根据应用的需求，还可以实现其他电路设计。

在编程和调试过程中，必须考虑到引脚上可能会增加的电阻和电容。器件编程器和调试器会驱动 $\overline{\text{MCLR}}$ 引脚。因此，特定电平 （VIH 和 VIL）和快速信号切换一定不能受到影响。所以，需要根据应用和 PCB 需求来调整 R1 和 C1 的具体值。例如，在编程和调试操作期间，建议通过使用跳线将电容C1与$\overline{\text{MCLR}}$引脚隔离（图2-2）。对于正常的运行时操作，可以将跳线放回原处。

与 $\overline{\text{MCLR}}$ 引脚关联的所有元件都应放置在引脚 0.25 英寸（6 毫米）范围内。

图 2-2： $\overline{\text{MCLR}}$ 引脚连接示例



注 1： 建议 R1 ≤ 10 kΩ。建议的起始值为 10 kΩ。请确保满足 $\overline{\text{MCLR}}$ 引脚 VIH 和 VIL 规范。

2： R2 ≤ 470Ω 将限制任何电流从外部电容 C 流入 $\overline{\text{MCLR}}$，以避免由于静电放电（Electrostatic Discharge， ESD）或电过载（Electrical Overstress，EOS）导致 $\overline{\text{MCLR}}$ 引脚损坏。请确保满足 $\overline{\text{MCLR}}$ 引脚 VIH 和 VIL 规范。

2.4 稳压器引脚 （ENVREG/DISVREG 和 VCAP/VDDCORE）

注： 本节仅适用于带有片上稳压器的 PIC24FJ 器件。

片上稳压器使能 / 禁止引脚 （ENVREG 或 DISVREG，取决于器件系列）必须总是直接与电源连接或直接接地。具体的连接取决于是否使用稳压器：

- 对于 ENVREG，与 VDD 连接可以使能稳压器，接地可以禁止稳压器
- 对于 DISVREG，接地可以使能稳压器，与 VDD 连接可以禁止稳压器

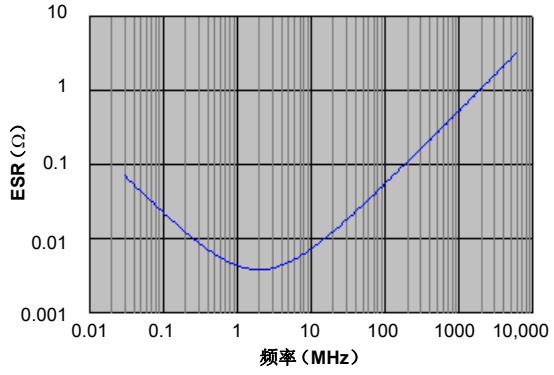
关于连接和使用片上稳压器的详细信息，请参见**第 25.2 节 “片上稳压器”**。

使能稳压器时，在 VCAP/VDDCORE 引脚上需要有低 ESR（< 5Ω）电容，以稳定稳压器的输出电压。VCAP/VDDCORE 引脚一定不能与 VDD 连接，并且必须使用 10 µF 的电容接地。可以使用陶瓷电容或钽电容。一个合适的示例就是 Murata GRM21BF50J106ZE01 （10 µF， 6.3V），或等效电容。设计人员可以根据图 2-3 来评估候选器件的 ESR 等效值。

该电容的位置应靠近 VCAP/VDDCORE。建议走线长度不要超出 0.25 英寸 （6 毫米）。更多信息，请参见**第 28.0 节 “电气特性”**。

禁止稳压器时， VCAP/VDDCORE 引脚必须与电压为 VDDCORE 的电源连接。关于 VDD 和 VDDCORE 的信息，请参见**第 28.0 节 “电气特性”**。

图 2-3： 所建议 VCAP 的频率与 ESR 性能



注： 所显示的是对应于 Murata GRM21BF50J106ZE01 的数据。在 25°C、 0V 直流偏置条件下测量。

2.5 ICSP 引脚

PGECx 和 PGEDx 引脚用于进行在线串行编程（ICSP）和调试。建议尽可能减小 ICSP 连接器与器件 ICSP 引脚之间的走线长度。如果 ICSP 连接器会遇到 ESD 事件，则建议添加一个串联电阻，电阻值介于几十欧姆的范围，不要超出 100Ω。

建议不要在 PGECx 和 PGEDx 引脚上连接上拉电阻、串联二极管和电容，因为它们会影响与器件的编程器 / 调试器通信。如果应用需要此类分立元件，则在编程和调试期间应将它们从电路中取出。或者，请参见相应器件闪存编程规范中的交流 / 直流特性与时序要求信息，了解关于容性负载限制、引脚输入高电压 （VIH）和输入低电压 （VIL）要求的信息。

对于器件仿真，请确保烧写到器件中的 “通信通道选择”（即，PGECx/PGEDx 引脚）符合 ICSP 到 MPLAB® ICD 2、 MPLAB ICD 3 或 MPLAB REAL ICE™ 仿真器的物理连接。

关于 ICD 2、 ICD 3 和 REAL ICE 仿真器连接要求的更多信息，请参见 Microchip 网站上提供的以下文档。

- 《MPLAB® ICD 2 在线调试器用户指南》（DS51331C_CN）
- *“Using MPLAB® ICD 2”*（海报）（DS51265）
- *“MPLAB® ICD 2 Design Advisory”*（DS51566）
- *“Using MPLAB® ICD 3”*（海报）（DS51765）
- *“MPLAB® ICD 3 Design Advisory”*（DS51764）
- 《MPLAB® REAL ICE™ 在线仿真器用户指南》（DS51616A_CN）
- *“Using MPLAB® REAL ICE™ In-Circuit Emulator”*（海报）（DS51749）

2.6 外部振荡器引脚

许多单片机都有至少两个振荡器可供选择：高频主振荡器和低频辅助振荡器（详细信息，请参见**第 8.0 节“振荡器配置”**）。

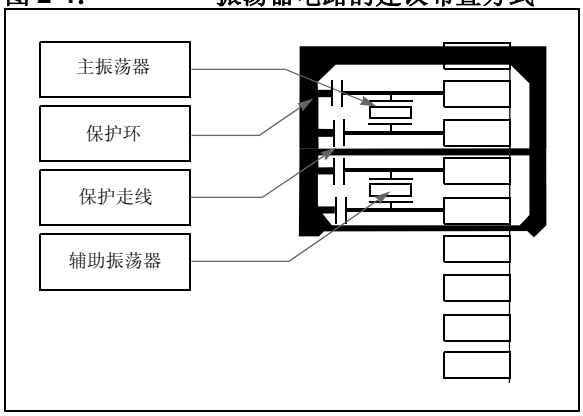
振荡器电路与器件应放置在电路板的同一面。请将振荡器电路放置在靠近相应振荡器引脚的位置，电路元件与引脚之间的距离不要超出 0.5 英寸（12 毫米）。负载电容应靠近振荡器自身，位于电路板的同一面。

在振荡器电路周围请设置接地灌铜区，以将它与周围电路隔离。接地灌铜区应与 MCU 地直接连接。不要在接地灌铜区内安排任何信号走线或电源走线。此外，如果使用双面电路板，则对于晶振所在位置，请避免在电路板另一面有任何走线。图2-4给出了一个建议的布线图。

关于振荡器电路的其他信息和设计指南，请参见 Microchip 公司网站（www.microchip.com）上提供的以下应用笔记：

- AN826，*“Crystal Oscillator Basics and Crystal Selection for rfPIC® and PICmicro® Devices”*
- AN849，*“Basic PICmicro® Oscillator Design”*
- AN943，*“Practical PICmicro® Oscillator Analysis and Design”*
- AN949，*“Making Your Oscillator Work”*

图 2-4：　　振荡器电路的建议布置方式



2.7 ICSP 操作期间的模拟和数字引脚配置

如果选择使用 MPLAB ICD 2、ICD 3 或 REAL ICE 仿真器作为调试器，则它们会自动将 AD1PCFGL 寄存器中的所有位置 1，从而将所有 A/D 输入引脚（ANx）初始化为“数字”引脚。

禁止用户应用程序固件清零该寄存器中对应于由 MPLAB ICD 2、ICD 3 或 REAL ICE 仿真器初始化的 A/D 引脚的位；否则，调试器和器件之间将会产生通信错误。

如果在调试会话期间，应用程序需要使用一些特定 A/D 引脚作为模拟输入引脚，则用户应用程序必须在ADC模块初始化期间，清零 AD1PCFGL 寄存器中的相应位。

使用 MPLAB ICD 2、ICD 3 或 REAL ICE 仿真器作为编程器时，用户应用程序固件必须正确地配置 AD1PCFGL 寄存器。该寄存器的自动初始化仅在调试器操作期间执行。未能正确配置寄存器将导致所有 A/D 引脚被识别为模拟输入引脚，从而导致端口值读为逻辑 0，这可能会影响用户应用程序功能。

2.8 未用 I/O

未用I/O引脚应配置为输出，并驱动为逻辑低电平状态。或者，将未用引脚通过一个 1 kΩ 至 10 kΩ 的电阻与 VSS 连接，并将输出驱动为逻辑低电平。

3.0 CPU

注：	本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 2 章“CPU”**（DS39703A_CN）。

PIC24F CPU 采用 16 位（数据）的改进型哈佛架构，具有增强指令集以及带有长度可变操作码字段的24位指令字。程序计数器（Program Counter，PC）为 23 位宽，可以寻址最大 4M 指令字的用户程序存储空间。单周期指令预取机制可帮助维持吞吐量，并使指令的执行具有预测性。除了改变程序流的指令、双字传送（MOV.D）指令和表指令以外，所有指令都在单个周期内执行。使用 REPEAT 指令可以支持无开销的程序循环结构，该指令在任何时间都可以被中断。

PIC24F 器件在编程模型中有 16 个 16 位工作寄存器。每个工作寄存器都可以充当数据、地址或地址偏移寄存器。第 16 个工作寄存器（W15）作为软件堆栈指针工作，用于中断和调用。

可以选择将数据存储空间的高 32 KB 映射到由 8 位程序空间可视性页地址（Program Space Visibility Page Address，PSVPAG）寄存器定义的任何 16K 字边界内的程序空间内。程序空间到数据空间的映射功能让任何指令都能像访问数据空间一样访问程序空间。

指令集架构（Instruction Set Architecture，ISA）与 PIC18 相比有了显著的提升，但仍保持了一定程度的向后兼容性。该架构直接支持或通过简单的宏支持所有的 PIC18 指令和寻址模式。对编译器执行效率的需求也促使了对 ISA 的许多改进。

内核支持固有（无操作数）寻址、相对寻址、立即数寻址、存储器直接寻址及其他三组寻址模式。所有模式都支持寄存器直接寻址和各种寄存器间接寻址模式。每组都提供了最多 7 种寻址模式。指令根据其功能要求，与预定义的寻址模式相关联。

对于大多数指令，内核能在每个指令周期内执行一次数据（或程序数据）存储器读操作、一次工作寄存器（数据）读操作、一次数据存储器写操作和一次程序（指令）存储器读操作。因此可以支持三个操作数的指令，使三个操作数的运算（即，A + B = C）能在单个周期内执行。

内核中包括一个高速 17 位 x 17 位乘法器，显著提高了内核的运算能力和吞吐量。乘法器支持有符号、无符号和混合模式的 16 位 x 16 位或 8 位 x 8 位整数乘法。所有的乘法指令都在单个周期内执行。

已对 16 位 ALU 进行了改进使其具备一个支持整数除法的硬件，该硬件支持迭代的不可撤消的除法算法。它与 REPEAT 指令循环机制和迭代除法指令一起工作，支持 32 位（或 16 位）除以 16 位有符号和无符号整数的除法运算。所有除法运算都需要 19 个周期来完成，但可以在任何周期边界被中断。

PIC24F 具有向量异常机制，带有最多 8 个不可屏蔽陷阱源和最多 118 个中断源。可以为每个中断源分配 7 个优先级之一。

图 3-1 给出了 CPU 的框图。

3.1 编程模型

图 3-2 给出了 PIC24F 的编程模型。编程模型中的所有寄存器都是存储器映射的，并且可以由指令直接操作。表 3-1 中提供了对每个寄存器的说明。所有与编程模型相关的寄存器都是存储器映射的。

图 3-1：　　PIC24F CPU 内核框图

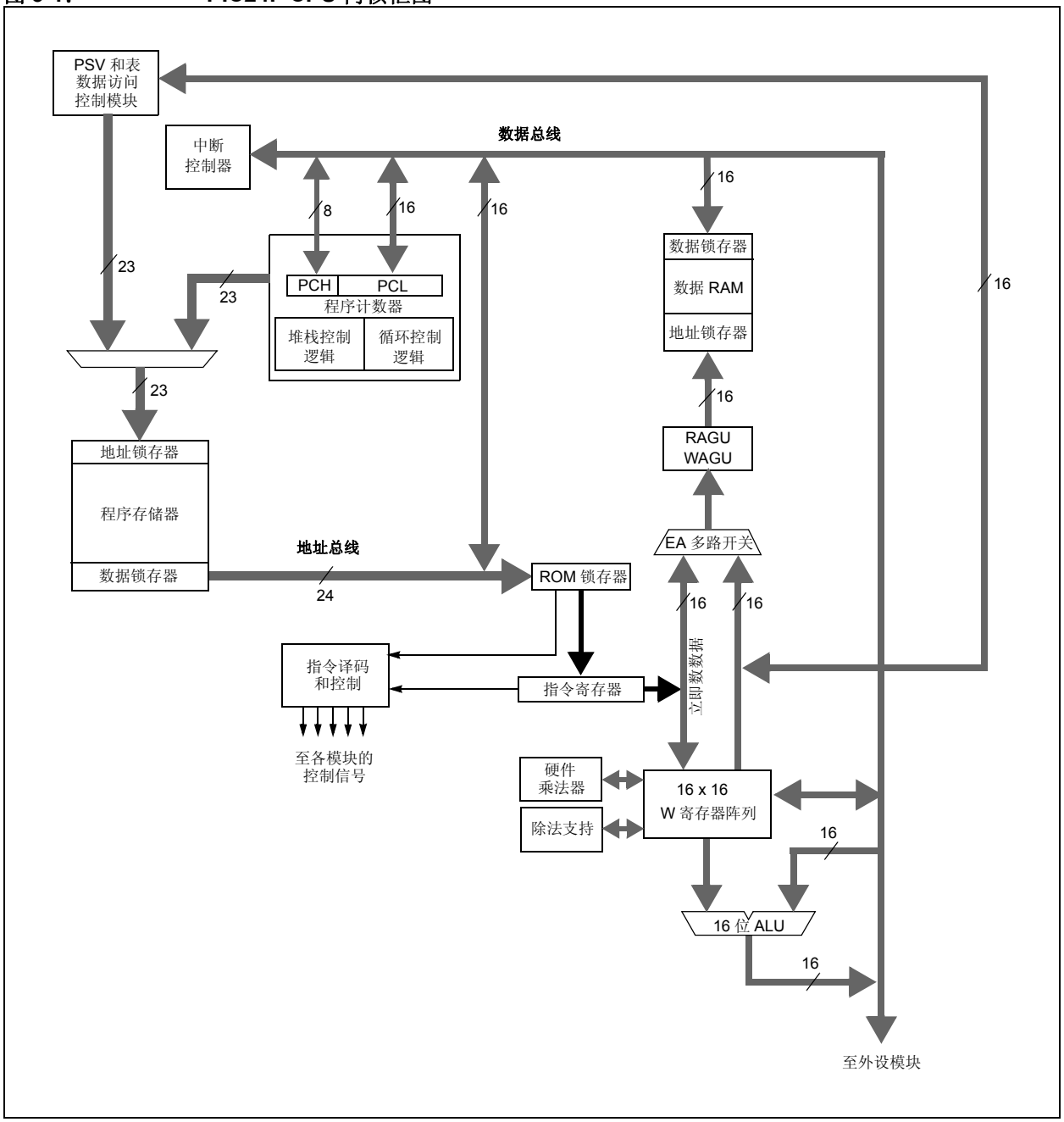
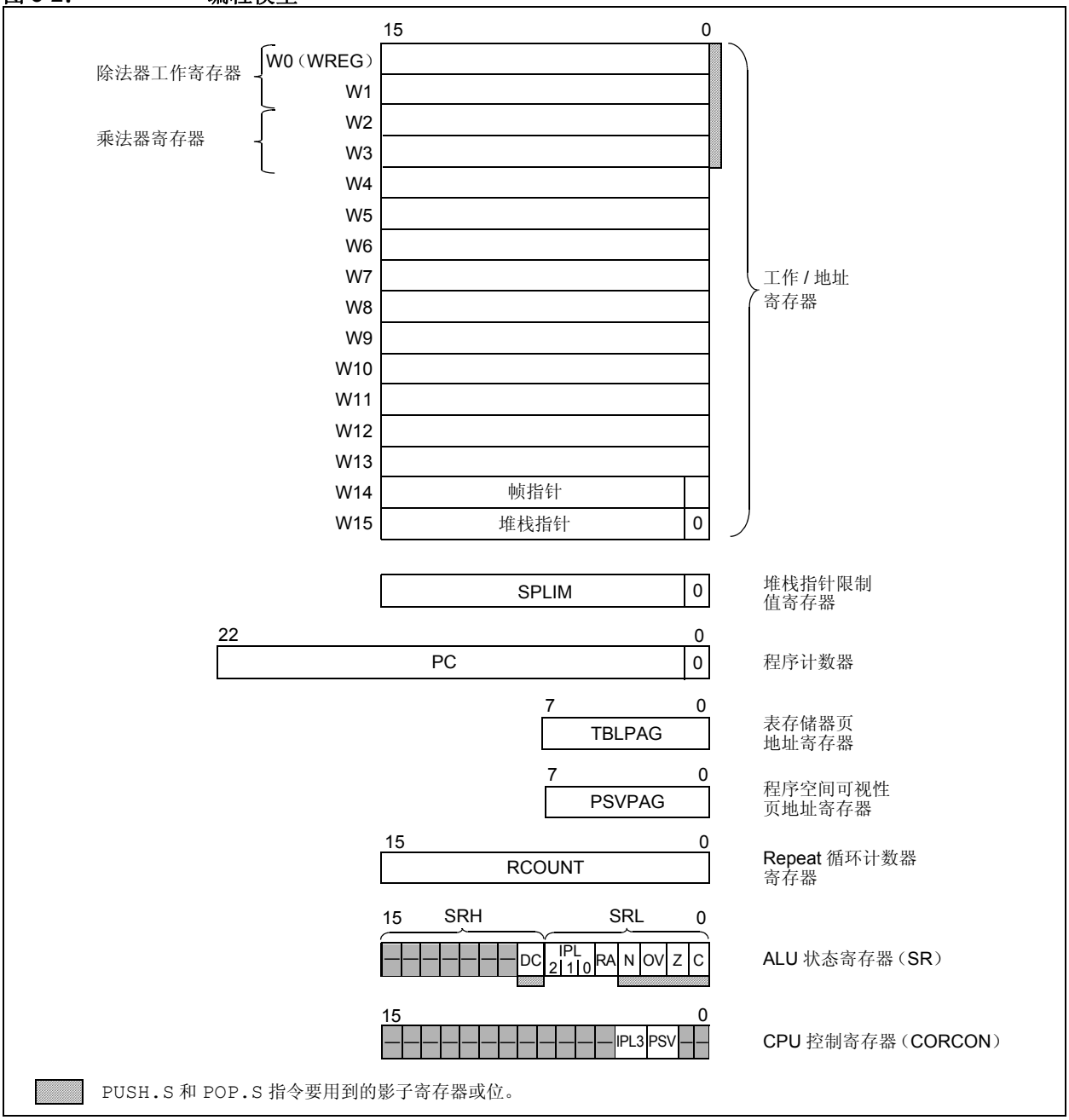


表 3-1：　　CPU 内核寄存器

寄存器名称	说明
W0 到 W15	工作寄存器阵列
PC	23 位程序计数器
SR	ALU 状态寄存器
SPLIM	堆栈指针限制值寄存器
TBLPAG	表存储器页地址寄存器
PSVPAG	程序空间可视性页地址寄存器
RCOUNT	Repeat 循环计数器寄存器
CORCON	CPU 控制寄存器

图 3-2：　　编程模型



3.2 CPU 控制寄存器

寄存器 3-1： SR：ALU 状态寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	DC
bit 15							bit 8

R/W-0(1)	R/W-0(1)	R/W-0(1)	R-0	R/W-0	R/W-0	R/W-0	R/W-0
IPL2(2)	IPL1(2)	IPL0(2)	RA	N	OV	Z	C
bit 7							bit 0

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

bit 15-9 **未实现：**读为 0

bit 8 **DC：**ALU 半进位 / 借位标志位
1 = 结果的第 4 个低位 （对于字节大小的数据）或第 8 个低位 （对于字大小的数据）发生了进位
0 = 结果的第 4 个或第 8 个低位未发生进位

bit 7-5 **IPL<2:0>：**CPU 中断优先级状态位 (1,2)
111 = CPU 中断优先级为 7 （15）；禁止用户中断
110 = CPU 中断优先级为 6 （14）
101 = CPU 中断优先级为 5 （13）
100 = CPU 中断优先级为 4 （12）
011 = CPU 中断优先级为 3 （11）
010 = CPU 中断优先级为 2 （10）
001 = CPU 中断优先级为 1 （9）
000 = CPU 中断优先级为 0 （8）

bit 4 **RA：**REPEAT 循环活动位
1 = 正在进行 REPEAT 循环
0 = 不在进行 REPEAT 循环

bit 3 **N：**ALU 负标志位
1 = 结果为负
0 = 结果为非负 （零或正值）

bit 2 **OV：**ALU 溢出标志位
1 = 有符号 （二进制补码）算术运算中发生溢出 （本次运算）
0 = 未发生溢出

bit 1 **Z：**ALU 全零标志位
1 = 影响 Z 位的任何运算在过去某时已将该位置 1
0 = 影响 Z 位的最近一次运算已将该位清零 （即运算结果非零）

bit 0 **C：**ALU 进位 / $\overline{\text{借位}}$标志位
1 = 结果的最高有效位发生了进位
0 = 结果的最高有效位未发生进位

注 1： 当 NSTDIS （INTCON1<15>） = 1 时， IPL 状态位是只读位。
2： IPL 状态位与 IPL3 位 （CORCON<3>）组合形成 CPU 中断优先级 （Interrupt Priority Level， IPL）。当 IPL3 = 1 时，括号中的值表示 IPL。

寄存器 3-2：　　CORCON：CPU 控制寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	R/C-0	R/W-0	U-0	U-0
—	—	—	—	IPL3(1)	PSV	—	—
bit 7							bit 0

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

bit 15-4　**未实现：**读为 0

bit 3　**IPL3：**CPU 中断优先级状态位 (1)
1 = CPU 中断优先级大于 7
0 = CPU 中断优先级等于或小于 7

bit 2　**PSV：**数据空间中程序空间可视性使能位
1 = 程序空间在数据空间中可视
0 = 程序空间在数据空间中不可视

bit 1-0　**未实现：**读为 0

注　1： 当 IPL3 = 1 时，禁止用户中断。

3.3　算术逻辑单元（ALU）

PIC24F ALU 为 16 位宽，并能进行加法、减法、移位和逻辑运算。除非另外声明，算术运算一般采用二进制补码方式进行。根据不同的运算，ALU 可能会影响 SR 寄存器中的进位标志位（C）、全零标志位（Z）、负标志位（N）、溢出标志位（OV）和半进位标志位（DC）的值。在减法运算中，C 和 DC 状态位分别作为借位位和半借位位。

根据所使用的指令模式，ALU 可执行 8 位或 16 位运算。根据指令的寻址模式，ALU 运算的数据可以来自 W 寄存器阵列或数据存储器。同样，ALU 的输出数据可被写入 W 寄存器阵列或数据存储单元。

PIC24F CPU 融入了对乘法和除法的硬件支持。它带有专用的硬件乘法器以及支持 16 位除数除法的硬件。

3.3.1　乘法器

ALU 包含一个高速 17 位 x 17 位乘法器。它支持以下几种无符号、有符号或混合符号乘法运算模式：

1. 16 位 x 16 位有符号
2. 16 位 x 16 位无符号
3. 16 位有符号 x 5 位（立即数）无符号
4. 16 位无符号 x 16 位无符号
5. 16 位无符号 x 5 位（立即数）无符号
6. 16 位无符号 x 16 位有符号
7. 8 位无符号 x 8 位无符号

3.3.2 除法器

除法模块支持具有以下数据长度的有符号和无符号整数除法运算：

1. 32 位有符号 /16 位有符号除法
2. 32 位无符号 /16 位无符号除法
3. 16 位有符号 /16 位有符号除法
4. 16 位无符号 /16 位无符号除法

所有除法指令的商都被放在 W0 中，余数放在 W1 中。16 位有符号和无符号 DIV 指令可为 16 位除数指定任一 W 寄存器（Wn），为 32 位被除数指定任意两个连续的 W 寄存器（W(m + 1):Wm）。除法运算中处理除数的每一位需要一个周期，因此 32 位 /16 位和 16 位 /16 位指令的执行周期数相同。

3.3.3 多位移位支持

PIC24F ALU 支持单位和单周期、多位算术和逻辑移位。多位移位使用移位寄存器模块实现，能够在单个周期内执行最多 15 位的算术右移或最多 15 位的算术左移。所有的多位移位指令都只支持操作数源寄存器和结果目标寄存器的寄存器直接寻址模式。

下面的表 3-2 中提供了使用移位操作的指令的完整汇总。

表 3-2： 使用单位和多位移位操作的指令

指令	说明
ASR	将源寄存器算术右移一位或多位。
SL	将源寄存器左移一位或多位。
LSR	将源寄存器逻辑右移一位或多位。

4.0 存储器构成

作为哈佛架构器件， PIC24F 单片机具有独立的程序和数据存储空间以及总线。这一架构同时还允许在代码执行过程中从数据空间直接访问程序存储器。

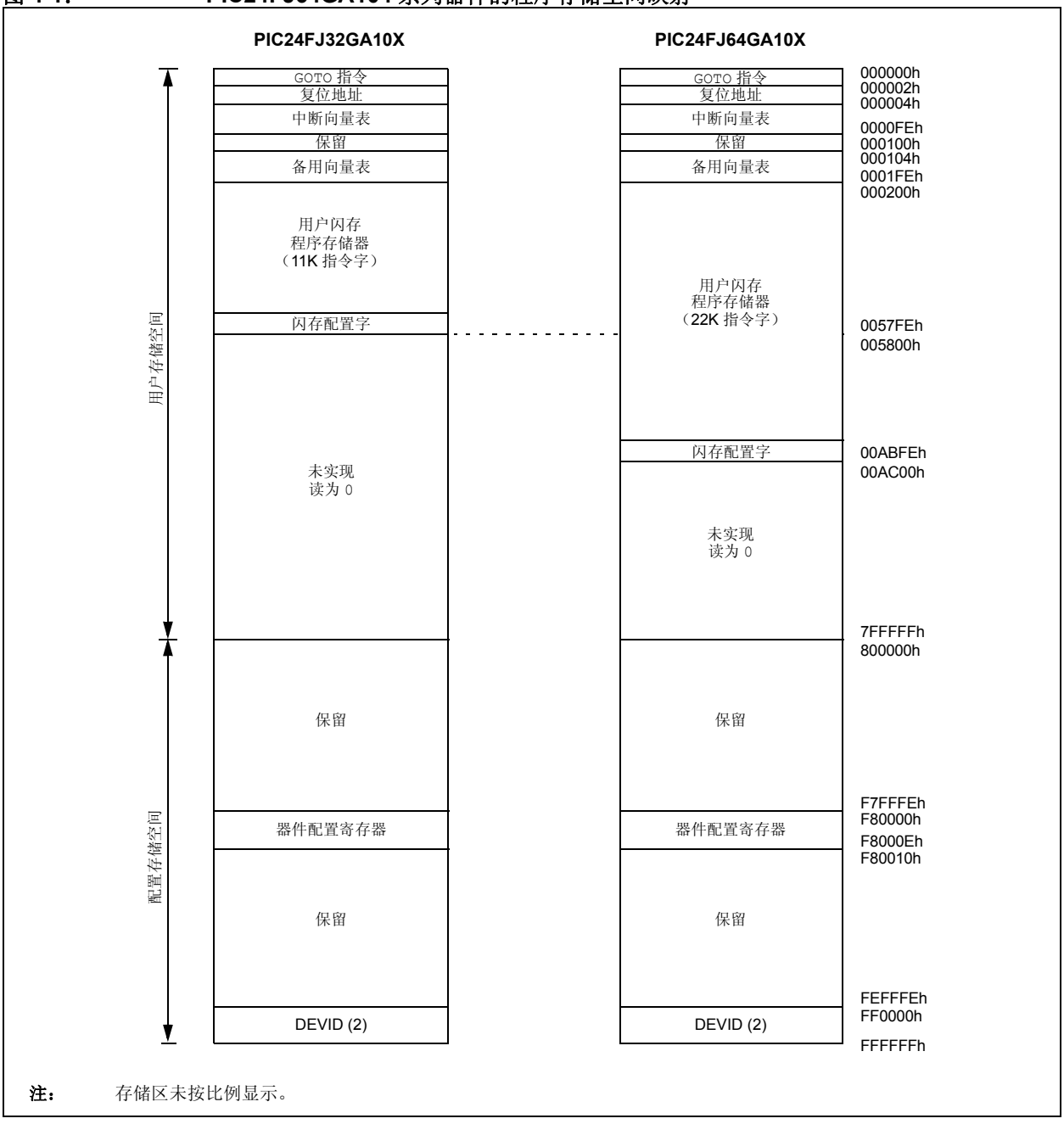
4.1 程序地址空间

PIC24FJ64GA104 系列器件的程序地址存储空间可存储 4M 个指令字。可通过由程序执行过程中 23 位程序计数器 （PC）、表操作或数据空间重映射得到的 24 位值寻址这一空间，如**第 4.3 节 “程序存储空间与数据存储空间的接口”**中所述。

用户只能访问程序存储空间的低半地址部分（地址范围为 000000h 至 7FFFFFh）。使用 TBLRD/TBLWT 指令时，情况有所不同，这两条指令使用 TBLPAG<7> 以允许访问配置存储空间中的配置位和器件 ID。

图4-1给出了PIC24FJ64GA104系列器件的存储器映射。

图 4-1： PIC24FJ64GA104 系列器件的程序存储空间映射



注： 存储区未按比例显示。

4.1.1 程序存储器构成

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

程序存储器地址始终在低位字处按字对齐，并且在代码执行过程中地址将递增或递减 2。这种寻址模式与数据存储空间寻址兼容，且为访问程序存储空间中的数据提供了可能。

4.1.2 存储器硬编码向量

所有 PIC24F 器件中从 00000h 到 000200h 之间的地址空间都是保留的，用来存储硬编码的程序执行向量。提供了一个硬件复位向量将代码执行从器件复位时 PC 的默认值重新定位到代码实际起始处。用户可在地址 000000h 处编写一条 GOTO 指令以将代码的实际起始地址设置为 000002h。

PIC24F 器件也具有两个中断向量表，地址分别为从 000004h 到 0000FFh 和 000100h 到 0001FFh。这两个向量表允许使用不同的中断服务程序（Interrupt Service Routine，ISR）处理每个器件中断源。关于中断向量表更详细的讨论，请参见**第 7.1 节 “中断向量表”**。

4.1.3 闪存配置字

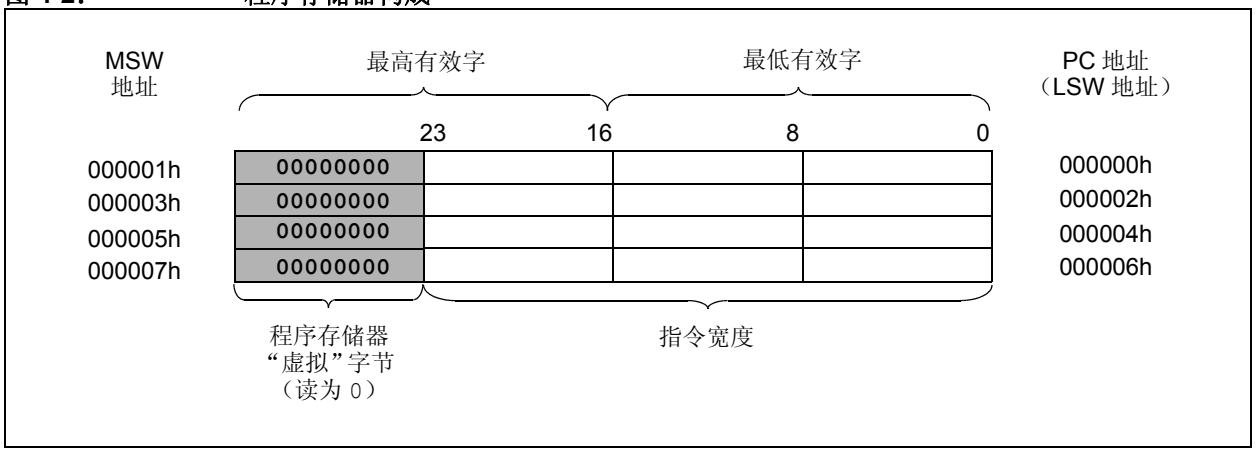
在 PIC24FJ64GA104 系列器件中，保留片上程序存储器的开始 4 个字用于配置信息。器件复位时，该配置信息被复制到相应的配置寄存器中。PIC24FJ64GA104 系列中器件的闪存配置字的地址如表 4-1 所示。图 4-1 给出了闪存配置字以及其他存储器向量在存储器映射中的位置。

程序存储器中的配置字为紧凑的格式。实际配置位被映射到配置存储空间的几个不同的寄存器中。它们在闪存配置字中的顺序并不反映它们在配置空间中的相应顺序。**第 25.1 节 “配置位”**中提供了关于器件配置字的更多详细信息。

表 4-1： PIC24FJ64GA104 系列器件的闪存配置字

器件	程序存储器（字）	配置字地址
PIC24FJ32GB0	11,008	0057F8h: 0057FEh
PIC24FJ64GB0	22,016	00ABF8h: 00ABFEh

图 4-2： 程序存储器构成



4.2 数据地址空间

PIC24F 内核具有独立的 16 位宽数据存储空间，可将其作为一个线性空间寻址。使用两个地址发生单元（Address Generation Unit，AGU）分别对数据空间执行读写操作。数据存储空间映射如图 4-3 所示。

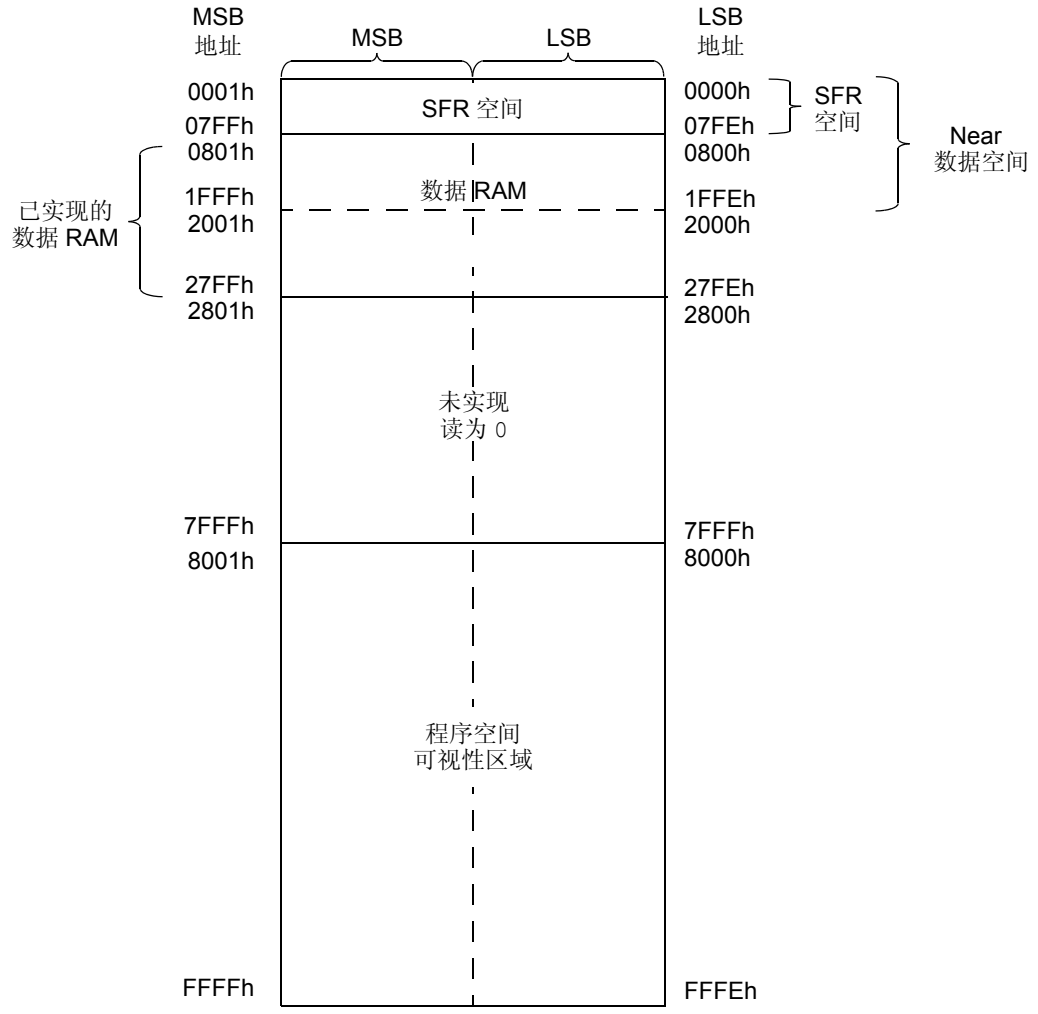
数据存储空间中的所有有效地址（Effective Address，EA）均为 16 位宽，并且指向数据空间内的字节。这种构成方式使得数据空间的地址范围为 64 KB 即 32K 字。数据存储空间的低半地址部分（即当 EA<15> = 0 时）用作实现的存储单元，而高半地址部分（EA<15> = 1）则保留为程序空间可视性区域（见**第 4.3.3 节 “使用程序空间可视性读程序存储器中的数据”**）。

PIC24FJ64GA104 系列器件实现了总共 16 KB 的数据存储器。如果 EA 指向了该区域以外的存储单元，则将返回一个全零的字或字节。

4.2.1 数据空间宽度

数据存储空间由可字节寻址的 16 位宽的块构成。在数据存储器和寄存器中的数据是以 16 位字为单位对齐的，但所有数据空间的 EA 都被解析为字节。每个字的最低有效字节（LSB）部分具有偶地址，而最高有效字节（MSB）部分则具有奇地址。

图 4-3： PIC24FJ64GA104 系列器件的数据存储空间映射



注： 数据存储区未按比例显示。

4.2.2 数据存储器构成和对齐方式

为维持与 PIC® 器件的向后兼容性和提高数据存储空间的使用效率，PIC24F 指令集同时支持字和字节操作。字节访问会在内部对按字对齐的存储空间的所有有效地址计算进行调整。例如，对于执行后修改寄存器间接寻址模式 [Ws++] 的结果，字节操作时，内核将其识别为值 Ws + 1；而字操作时，内核将其识别为值 Ws + 2。

使用任何 EA 的 LSb 来确定要选取的字节，数据字节读操作将读取包含该字节的整个字。选定的字节被放在数据路径的 LSB 处。也就是说，数据存储器和寄存器被组织为两个并行的字节宽的实体，它们共享（字）地址译码，但写入线相互独立。数据字节写操作只写入阵列或寄存器中与字节地址匹配的那一侧。

所有字访问必须按偶地址对齐。不支持不对齐的字数据取操作，因此在进行混合字节和字操作或从 8 位 MCU 代码移植时，必须要小心。如果试图进行不对齐的读或写操作，将产生地址错误陷阱。如果在读操作时产生错误，正在执行的指令将完成；而如果在写操作时产生错误，指令仍将执行，但不会进行写入。无论是哪种情况都将产生陷阱，从而允许系统和 / 或用户检查地址错误发生之前的机器状态。

所有装入 W 寄存器的字节都将被装入最低有效字节（LSB）。最高有效字节不变。

提供了一条符号扩展（SE）指令，允许用户将 8 位有符号数据转换为 16 位有符号值。或者，对于 16 位无符号数据，用户可以通过在适当地址处执行一条零扩展（ZE）指令清零任何 W 寄存器的 MSB。

尽管大多数指令能够对字或字节大小的数据进行操作，但必须要注意的是，某些指令仅适用于字操作。

4.2.3 NEAR 数据空间

在 0000h 和 1FFFh 之间的 8 KB 的区域被称为 Near 数据空间。可以使用所有存储器直接寻址指令中的 13 位绝对地址字段直接寻址这一空间中的存储单元。剩余的数据空间通过间接寻址访问。此外，还可以使用 MOV 指令寻址整个数据空间，支持使用 16 位地址字段的存储器直接寻址。

4.2.4 SFR 空间

Near 数据空间的前 2 KB 单元（从 0000h 到 07FFh）主要被特殊功能寄存器（Special Function Register，SFR）占用。PIC24F 的内核和外设模块使用这些寄存器来控制器件的工作。

SFR 被分配给受其控制的模块，通常一个模块会使用一组 SFR。许多 SFR 空间包含未用的地址单元，它们读为 0。显示实际实现的 SFR 所在地址的 SFR 空间的框图如表 4-2 所示。每个已实现区域指示一个 32 字节区域，其中至少有一个地址实现为SFR。表 4-3至表 4-26 给出了已实现 SFR 及其地址的完整列表。

表 4-2： SFR 数据空间的已实现区域

	SFR 空间地址							
	xx00	**xx20**	**xx40**	**xx60**	**xx80**	**xxA0**	**xxC0**	**xxE0**
000h	内核			ICN	中断			—
100h	定时器		捕捉		比较			—
200h	I^2C™	UART	SPI	—	—	—	I/O	
300h	A/D	A/D/CTMU	—	—	—	—	—	—
400h	—	—	—	—	—	—	—	—
500h	—	—	—	—	—	—	—	—
600h	PMP	RTCC	CRC/ 比较器	比较器	PPS			—
700h	—	—	系统 /DS	NVM/PMD	—	—	—	—

图注： — = 该存储块中未实现的 SFR

表 4-3：　　CPU 内核寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
WREG0	0000	工作寄存器 0																0000
WREG1	0002	工作寄存器 1																0000
WREG2	0004	工作寄存器 2																0000
WREG3	0006	工作寄存器 3																0000
WREG4	0008	工作寄存器 4																0000
WREG5	000A	工作寄存器 5																0000
WREG6	000C	工作寄存器 6																0000
WREG7	000E	工作寄存器 7																0000
WREG8	0010	工作寄存器 8																0000
WREG9	0012	工作寄存器 9																0000
WREG10	0014	工作寄存器 10																0000
WREG11	0016	工作寄存器 11																0000
WREG12	0018	工作寄存器 12																0000
WREG13	001A	工作寄存器 13																0000
WREG14	001C	工作寄存器 14																0000
WREG15	001E	工作寄存器 15																0800
SPLIM	0020	堆栈指针限制值寄存器																xxxx
PCL	002E	程序计数器低位字寄存器																0000
PCH	0030	—	—	—	—	—	—	—	—	程序计数器寄存器的高字节								0000
TBLPAG	0032	—	—	—	—	—	—	—	—	表存储器页地址寄存器								0000
PSVPAG	0034	—	—	—	—	—	—	—	—	程序空间可视性页地址寄存器								0000
RCOUNT	0036	Repeat 循环计数器寄存器																xxxx
SR	0042	—	—	—	—	—	—	—	DC	IPL2	IPL1	IPL0	RA	N	OV	Z	C	0000
CORCON	0044	—	—	—	—	—	—	—	—	—	—	—	—	IPL3	PSV	—	—	0000
DISICNT	0052	—	—	禁止中断计数器寄存器														xxxx

图注：　　**— =** 未实现，读为 0。复位值以十六进制显示。

表 4-4： ICN 寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
CNEN1	0060	CN15IE	CN14IE	CN13IE	CN12IE	CN11IE	CN10IE(1)	CN9IE(1)	CN8IE(1)	CN7IE	CN6IE	CN5IE	CN4IE	CN3IE	CN2IE	CN1IE	CN0IE	0000
CNEN2	0062	—	CN30IE	CN29IE	CN28IE(1)	CN27IE	CN26IE(1)	CN25IE(1)	CN24IE	CN23IE	CN22IE	CN21IE	CN20IE(1)	CN19IE(1)	CN18IE(1)	CN17IE(1)	CN16IE	0000
CNPU1	0068	CN15PUE	CN14PUE	CN13PUE	CN12PUE	CN11PUE	CN10PUE(1)	CN9PUE(1)	CN8PUE(1)	CN7PUE	CN6PUE	CN5PUE	CN4PUE	CN3PUE	CN2PUE	CN1PUE	CN0PUE	0000
CNPU2	006A	—	CN30PUE	CN29PUE	CN28PUE(1)	CN27PUE	CN26PUE(1)	CN25PUE(1)	CN24PUE	CN23PUE	CN22PUE	CN21PUE	CN20PUE(1)	CN19PUE(1)	CN18PUE(1)	CN17PUE(1)	CN16PUE	0000

图注： — = 未实现，读为 0。复位值以十六进制显示。

注 1： 在 28 引脚器件上未实现；读为 0。

表 4-5：　　中断控制器寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
INTCON1	0080	NSTDIS	—	—	—	—	—	—	—	—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—	0000
INTCON2	0082	ALTIVT	DISI	—	—	—	—	—	—	—	—	—	—	—	INT2EP	INT1EP	INT0EP	0000
IFS0	0084	—	—	AD1IF	U1TXIF	U1RXIF	SPI1IF	SPF1IF	T3IF	T2IF	OC2IF	IC2IF	—	T1IF	OC1IF	IC1IF	INT0IF	0000
IFS1	0086	U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	—	—	—	—	INT1IF	CNIF	CMIF	MI2C1IF	SI2C1IF	0000
IFS2	0088	—	—	PMPIF	—	—	—	OC5IF	—	IC5IF	IC4IF	IC3IF	—	—	—	SPI2IF	SPF2IF	0000
IFS3	008A	—	RTCIF	—	—	—	—	—	—	—	—	—	—	—	MI2C2IF	SI2C2IF	—	0000
IFS4	008C	—	—	CTMUIF	—	—	—	—	LVDIF	—	—	—	—	CRCIF	U2ERIF	U1ERIF	—	0000
IFS5	008E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
IEC0	0094	—	—	AD1IE	U1TXIE	U1RXIE	SPI1IE	SPF1IE	T3IE	T2IE	OC2IE	IC2IE	—	T1IE	OC1IE	IC1IE	INT0IE	0000
IEC1	0096	U2TXIE	U2RXIE	INT2IE	T5IE	T4IE	OC4IE	OC3IE	—	—	—	—	INT1IE	CNIE	CMIE	MI2C1IE	SI2C1IE	0000
IEC2	0098	—	—	PMPIE	—	—	—	OC5IE	—	IC5IE	IC4IE	IC3IE	—	—	—	SPI2IE	SPF2IE	0000
IEC3	009A	—	RTCIE	—	—	—	—	—	—	—	—	—	—	—	MI2C2IE	SI2C2IE	—	0000
IEC4	009C	—	—	CTMUIE	—	—	—	—	LVDIE	—	—	—	—	CRCIE	U2ERIE	U1ERIE	—	0000
IEC5	009E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0000
IPC0	00A4	—	T1IP2	T1IP1	T1IP0	—	OC1IP2	OC1IP1	OC1IP0	—	IC1IP2	IC1IP1	IC1IP0	—	INT0IP2	INT0IP1	INT0IP0	4444
IPC1	00A6	—	T2IP2	T2IP1	T2IP0	—	OC2IP2	OC2IP1	OC2IP0	—	IC2IP2	IC2IP1	IC2IP0	—	—	—	—	4440
IPC2	00A8	—	U1RXIP2	U1RXIP1	U1RXIP0	—	SPI1IP2	SPI1IP1	SPI1IP0	—	SPF1IP2	SPF1IP1	SPF1IP0	—	T3IP2	T3IP1	T3IP0	4444
IPC3	00AA	—	—	—	—	—	—	—	—	—	AD1IP2	AD1IP1	AD1IP0	—	U1TXIP2	U1TXIP1	U1TXIP0	0044
IPC4	00AC	—	CNIP2	CNIP1	CNIP0	—	CMIP2	CMIP1	CMIP0	—	MI2C1IP2	MI2C1IP1	MI2C1IP0	—	SI2C1IP2	SI2C1IP1	SI2C1IP0	4444
IPC5	00AE	—	—	—	—	—	—	—	—	—	—	—	—	—	INT1IP2	INT1IP1	INT1IP0	0004
IPC6	00B0	—	T4IP2	T4IP1	T4IP0	—	OC4IP2	OC4IP1	OC4IP0	—	OC3IP2	OC3IP1	OC3IP0	—	—	—	—	4440
IPC7	00B2	—	U2TXIP2	U2TXIP1	U2TXIP0	—	U2RXIP2	U2RXIP1	U2RXIP0	—	INT2IP2	INT2IP1	INT2IP0	—	T5IP2	T5IP1	T5IP0	4444
IPC8	00B4	—	—	—	—	—	—	—	—	—	SPI2IP2	SPI2IP1	SPI2IP0	—	SPF2IP2	SPF2IP1	SPF2IP0	0044
IPC9	00B6	—	IC5IP2	IC5IP1	IC5IP0	—	IC4IP2	IC4IP1	IC4IP0	—	IC3IP2	IC3IP1	IC3IP0	—	—	—	—	4440
IPC10	00B8	—	—	—	—	—	—	—	—	—	OC5IP2	OC5IP1	OC5IP0	—	—	—	—	0040
IPC11	00BA	—	—	—	—	—	—	—	—	—	PMPIP2	PMPIP1	PMPIP0	—	—	—	—	0040
IPC12	00BC	—	—	—	—	—	MI2C2IP2	MI2C2IP1	MI2C2IP0	—	SI2C2IP2	SI2C2IP1	SI2C2IP0	—	—	—	—	0440
IPC15	00C2	—	—	—	—	—	RTCIP2	RTCIP1	RTCIP0	—	—	—	—	—	—	—	—	0400
IPC16	00C4	—	CRCIP2	CRCIP1	CRCIP0	—	U2ERIP2	U2ERIP1	U2ERIP0	—	U1ERIP2	U1ERIP1	U1ERIP0	—	—	—	—	4440
IPC18	00C8	—	—	—	—	—	—	—	—	—	—	—	—	—	LVDIP2	LVDIP1	LVDIP0	0004
IPC19	00CA	—	—	—	—	—	—	—	—	—	CTMUIP2	CTMUIP1	CTMUIP0	—	—	—	—	0040
IPC21	00CE	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	0400
INTTREG	00E0	CPUIRQ	—	VHOLD	—	ILR3	ILR2	ILR1	ILR0	—	VECNUM6	VECNUM5	VECNUM4	VECNUM3	VECNUM2	VECNUM1	VECNUM0	0000

图注：　　**— =** 未实现，读为 0。复位值以十六进制显示。

表 4-6：　　定时器寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
TMR1	0100	Timer1 寄存器																0000
PR1	0102	Timer1 周期寄存器																FFFF
T1CON	0104	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	TSYNC	TCS	—	0000
TMR2	0106	Timer2 寄存器																0000
TMR3HLD	0108	Timer3 保持寄存器（仅适用于 32 位定时器操作）																0000
TMR3	010A	Timer3 寄存器																0000
PR2	010C	Timer2 周期寄存器																FFFF
PR3	010E	Timer3 周期寄存器																FFFF
T2CON	0110	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T32	—	TCS	—	0000
T3CON	0112	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	—	TCS	—	0000
TMR4	0114	Timer4 寄存器																0000
TMR5HLD	0116	Timer5 保持寄存器（仅适用于 32 位定时器操作）																0000
TMR5	0118	Timer5 寄存器																0000
PR4	011A	Timer4 周期寄存器																FFFF
PR5	011C	Timer5 周期寄存器																FFFF
T4CON	011E	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	T32	—	TCS	—	0000
T5CON	0120	TON	—	TSIDL	—	—	—	—	—	—	TGATE	TCKPS1	TCKPS0	—	—	TCS	—	0000

图注：　　**— =** 未实现，读为 0。复位值以十六进制显示。

表 4-7：　　输入捕捉寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
IC1CON1	0140	—	—	ICSIDL	ICTSEL2	ICTSEL1	ICTSEL0	—	—	—	ICI1	ICI0	ICOV	ICBNE	ICM2	ICM1	ICM0	0000
IC1CON2	0142	—	—	—	—	—	—	—	IC32	ICTRIG	TRIGSTAT	—	SYNCSEL4	SYNCSEL3	SYNCSEL2	SYNCSEL1	SYNCSEL0	000D
IC1BUF	0144	输入捕捉 1 缓冲寄存器																0000
IC1TMR	0146	定时器值 1 寄存器																xxxx
IC2CON1	0148	—	—	ICSIDL	ICTSEL2	ICTSEL1	ICTSEL0	—	—	—	ICI1	ICI0	ICOV	ICBNE	ICM2	ICM1	ICM0	0000
IC2CON2	014A	—	—	—	—	—	—	—	IC32	ICTRIG	TRIGSTAT	—	SYNCSEL4	SYNCSEL3	SYNCSEL2	SYNCSEL1	SYNCSEL0	000D
IC2BUF	014C	输入捕捉 2 缓冲寄存器																0000
IC2TMR	014E	定时器值 2 寄存器																xxxx
IC3CON1	0150	—	—	ICSIDL	ICTSEL2	ICTSEL1	ICTSEL0	—	—	—	ICI1	ICI0	ICOV	ICBNE	ICM2	ICM1	ICM0	0000
IC3CON2	0152	—	—	—	—	—	—	—	IC32	ICTRIG	TRIGSTAT	—	SYNCSEL4	SYNCSEL3	SYNCSEL2	SYNCSEL1	SYNCSEL0	000D
IC3BUF	0154	输入捕捉 3 缓冲寄存器																0000
IC3TMR	0156	定时器值 3 寄存器																xxxx
IC4CON1	0158	—	—	ICSIDL	ICTSEL2	ICTSEL1	ICTSEL0	—	—	—	ICI1	ICI0	ICOV	ICBNE	ICM2	ICM1	ICM0	0000
IC4CON2	015A	—	—	—	—	—	—	—	IC32	ICTRIG	TRIGSTAT	—	SYNCSEL4	SYNCSEL3	SYNCSEL2	SYNCSEL1	SYNCSEL0	000D
IC4BUF	015C	输入捕捉 4 缓冲寄存器																0000
IC4TMR	015E	定时器值 4 寄存器																xxxx
IC5CON1	0160	—	—	ICSIDL	ICTSEL2	ICTSEL1	ICTSEL0	—	—	—	ICI1	ICI0	ICOV	ICBNE	ICM2	ICM1	ICM0	0000
IC5CON2	0162	—	—	—	—	—	—	—	IC32	ICTRIG	TRIGSTAT	—	SYNCSEL4	SYNCSEL3	SYNCSEL2	SYNCSEL1	SYNCSEL0	000D
IC5BUF	0164	输入捕捉 5 缓冲寄存器																0000
IC5TMR	0166	定时器值 5 寄存器																xxxx

图注：　　**—** = 未实现，读为 0。复位值以十六进制显示。

表 4-8：　　输出比较寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
OC1CON1	0190	—	—	OCSIDL	OCTSEL2	OCTSEL1	OCTSEL0	ENFLT2	ENFLT1	ENFLT0	OCFLT2	OCFLT1	OCFLT0	TRIGMODE	OCM2	OCM1	OCM0	0000
OC1CON2	0192	FLTMD	FLTOUT	FLTTRIEN	OCINV	—	DCB1	DCB0	OC32	OCTRIG	TRIGSTAT	OCTRIS	SYNCSEL4	SYNCSEL3	SYNCSEL2	SYNCSEL1	SYNCSEL0	000C
OC1RS	0194	输出比较 1 辅助寄存器																0000
OC1R	0196	输出比较 1 寄存器																0000
OC1TMR	0198	定时器值 1 寄存器																xxxx
OC2CON1	019A	—	—	OCSIDL	OCTSEL2	OCTSEL1	OCTSEL0	ENFLT2	ENFLT1	ENFLT0	OCFLT2	OCFLT1	OCFLT0	TRIGMODE	OCM2	OCM1	OCM0	0000
OC2CON2	019C	FLTMD	FLTOUT	FLTTRIEN	OCINV	—	DCB1	DCB0	OC32	OCTRIG	TRIGSTAT	OCTRIS	SYNCSEL4	SYNCSEL3	SYNCSEL2	SYNCSEL1	SYNCSEL0	000C
OC2RS	019E	输出比较 2 辅助寄存器																0000
OC2R	01A0	输出比较 2 寄存器																0000
OC2TMR	01A2	定时器值 2 寄存器																xxxx
OC3CON1	01A4	—	—	OCSIDL	OCTSEL2	OCTSEL1	OCTSEL0	ENFLT2	ENFLT1	ENFLT0	OCFLT2	OCFLT1	OCFLT0	TRIGMODE	OCM2	OCM1	OCM0	0000
OC3CON2	01A6	FLTMD	FLTOUT	FLTTRIEN	OCINV	—	DCB1	DCB0	OC32	OCTRIG	TRIGSTAT	OCTRIS	SYNCSEL4	SYNCSEL3	SYNCSEL2	SYNCSEL1	SYNCSEL0	000C
OC3RS	01A8	输出比较 3 辅助寄存器																0000
OC3R	01AA	输出比较 3 寄存器																0000
OC3TMR	01AC	定时器值 3 寄存器																xxxx
OC4CON1	01AE	—	—	OCSIDL	OCTSEL2	OCTSEL1	OCTSEL0	ENFLT2	ENFLT1	ENFLT0	OCFLT2	OCFLT1	OCFLT0	TRIGMODE	OCM2	OCM1	OCM0	0000
OC4CON2	01B0	FLTMD	FLTOUT	FLTTRIEN	OCINV	—	DCB1	DCB0	OC32	OCTRIG	TRIGSTAT	OCTRIS	SYNCSEL4	SYNCSEL3	SYNCSEL2	SYNCSEL1	SYNCSEL0	000C
OC4RS	01B2	输出比较 4 辅助寄存器																0000
OC4R	01B4	输出比较 4 寄存器																0000
OC4TMR	01B6	定时器值 4 寄存器																xxxx
OC5CON1	01B8	—	—	OCSIDL	OCTSEL2	OCTSEL1	OCTSEL0	ENFLT2	ENFLT1	ENFLT0	OCFLT2	OCFLT1	OCFLT0	TRIGMODE	OCM2	OCM1	OCM0	0000
OC5CON2	01BA	FLTMD	FLTOUT	FLTTRIEN	OCINV	—	DCB1	DCB0	OC32	OCTRIG	TRIGSTAT	OCTRIS	SYNCSEL4	SYNCSEL3	SYNCSEL2	SYNCSEL1	SYNCSEL0	000C
OC5RS	01BC	输出比较 5 辅助寄存器																0000
OC5R	01BE	输出比较 5 寄存器																0000
OC5TMR	01C0	定时器值 5 寄存器																xxxx

图注：　　— = 未实现，读为 0。复位值以十六进制显示。

表 4-9：　　I²C™ 寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
I2C1RCV	0200	—	—	—	—	—	—	—	—	接收寄存器								0000
I2C1TRN	0202	—	—	—	—	—	—	—	—	发送寄存器								00FF
I2C1BRG	0204	—	—	—	—	—	—	—	波特率发生器寄存器									0000
I2C1CON	0206	I2CEN	—	I2CSIDL	SCLREL	IPMIEN	A10M	DISSLW	SMEN	GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	1000
I2C1STAT	0208	ACKSTAT	TRSTAT	—	—	—	BCL	GCSTAT	ADD10	IWCOL	I2COV	D/$\overline{A}$	P	S	R/$\overline{W}$	RBF	TBF	0000
I2C1ADD	020A	—	—	—	—	—	—	地址寄存器										0000
I2C1MSK	020C	—	—	—	—	—	—	地址掩码寄存器										0000
I2C2RCV	0210	—	—	—	—	—	—	—	—	接收寄存器								0000
I2C2TRN	0212	—	—	—	—	—	—	—	—	发送寄存器								00FF
I2C2BRG	0214	—	—	—	—	—	—	—	波特率发生器寄存器									0000
I2C2CON	0216	I2CEN	—	I2CSIDL	SCLREL	IPMIEN	A10M	DISSLW	SMEN	GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	1000
I2C2STAT	0218	ACKSTAT	TRSTAT	—	—	—	BCL	GCSTAT	ADD10	IWCOL	I2COV	D/$\overline{A}$	P	S	R/$\overline{W}$	RBF	TBF	0000
I2C2ADD	021A	—	—	—	—	—	—	地址寄存器										0000
I2C2MSK	021C	—	—	—	—	—	—	地址掩码寄存器										0000

图注：　　**— =** 未实现，读为 0。复位值以十六进制显示。

表 4-10：　　UART 寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
U1MODE	0220	UARTEN	—	USIDL	IREN	RTSMD	—	UEN1	UEN0	WAKE	LPBACK	ABAUD	RXINV	BRGH	PDSEL1	PDSEL0	STSEL	0000
U1STA	0222	UTXISEL1	UTXINV	UTXISEL0	—	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA	0110
U1TXREG	0224	—	—	—	—	—	—	—	发送寄存器									xxxx
U1RXREG	0226	—	—	—	—	—	—	—	接收寄存器									0000
U1BRG	0228	波特率发生器预分频器寄存器																0000
U2MODE	0230	UARTEN	—	USIDL	IREN	RTSMD	—	UEN1	UEN0	WAKE	LPBACK	ABAUD	RXINV	BRGH	PDSEL1	PDSEL0	STSEL	0000
U2STA	0232	UTXISEL1	UTXINV	UTXISEL0	—	UTXBRK	UTXEN	UTXBF	TRMT	URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA	0110
U2TXREG	0234	—	—	—	—	—	—	—	发送寄存器									xxxx
U2RXREG	0236	—	—	—	—	—	—	—	接收寄存器									0000
U2BRG	0238	波特率发生器预分频器寄存器																0000

图注：　　**— =** 未实现，读为 0。复位值以十六进制显示。

表 4-11： SPI 寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
SPI1STAT	0240	SPIEN	—	SPISIDL	—	—	SPIBEC2	SPIBEC1	SPIBEC0	SRMPT	SPIROV	SRXMPT	SISEL2	SISEL1	SISEL0	SPITBF	SPIRBF	0000
SPI1CON1	0242	—	—	—	DISSCK	DISSDO	MODE16	SMP	CKE	SSEN	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0	0000
SPI1CON2	0244	FRMEN	SPIFSD	SPIFPOL	—	—	—	—	—	—	—	—	—	—	—	SPIFE	SPIBEN	0000
SPI1BUF	0248	发送和接收缓冲区																0000
SPI2STAT	0260	SPIEN	—	SPISIDL	—	—	SPIBEC2	SPIBEC1	SPIBEC0	SRMPT	SPIROV	SRXMPT	SISEL2	SISEL1	SISEL0	SPITBF	SPIRBF	0000
SPI2CON1	0262	—	—	—	DISSCK	DISSDO	MODE16	SMP	CKE	SSEN	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0	0000
SPI2CON2	0264	FRMEN	SPIFSD	SPIFPOL	—	—	—	—	—	—	—	—	—	—	—	SPIFE	SPIBEN	0000
SPI2BUF	0268	发送和接收缓冲区																0000

图注： — = 未实现，读为 0。复位值以十六进制显示。

表 4-12： PORTA 寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10(1)	Bit 9(1)	Bit 8(1)	Bit 7(1)	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
TRISA	02C0	—	—	—	—	—	TRISA10	TRISA9	TRISA8	TRISA7	—	—	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	079F
PORTA	02C2	—	—	—	—	—	RA10	RA9	RA8	RA7	—	—	RA4	RA3	RA2	RA1	RA0	xxxx
LATA	02C4	—	—	—	—	—	LATA10	LATA9	LATA8	LATA7	—	—	LATA4	LATA3	LATA2	LATA1	LATA0	xxxx
ODCA	02C6	—	—	—	—	—	ODA10	ODA9	ODA8	ODA7	—	—	ODA4	ODA3	ODA2	ODA1	ODA0	0000

图注： — = 未实现，读为 0。复位值以十六进制显示。显示的复位值针对 44 引脚器件。

注 1： 这些位在 28 引脚器件上未实现；读为 0。

表 4-13： PORTB 寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
TRISB	02C8	TRISB15	TRISB14	TRISB13	TRISB12	TRISB11	TRISB10	TRISB9	TRISB8	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	EFBF
PORTB	02CA	RB15	RB14	RB13	RB12	RB11	RB10	RB9	RB8	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx
LATB	02CC	LATB15	LATB14	LATB13	LATB12	LATB11	LATB10	LATB9	LATB8	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	xxxx
ODCB	02CE	ODB15	ODB14	ODB13	ODB12	ODB11	ODB10	ODB9	ODB8	ODB7	ODB6	ODB5	ODB4	ODB3	ODB2	ODB1	ODB0	0000

图注： — = 未实现，读为 0。复位值以十六进制显示。

表 4-14： PORTC 寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9(1)	Bit 8(1)	Bit 7(1)	Bit 6(1)	Bit 5(1)	Bit 4(1)	Bit 3(1)	Bit 2(1)	Bit 1(1)	Bit 0(1)	所有复位时的状态
TRISC	02D0	—	—	—	—	—	—	TRISC9	TRISC8	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	03FF
PORTC	02D2	—	—	—	—	—	—	RC9	RC8	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx
LATC	02D4	—	—	—	—	—	—	LATC9	LATC8	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	xxxx
ODCC	02D6	—	—	—	—	—	—	ODC9	ODC8	ODC7	ODC6	ODC5	ODC4	ODC3	ODC2	ODC1	ODC0	0000

图注： — = 未实现，读为 0。复位值以十六进制显示。显示的复位值针对 44 引脚器件。

注 1： 这些位在 28 引脚器件上未实现；读为 0。

表 4-15：　　焊盘配置寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
PADCFG1	02FC	—	—	—	—	—	—	—	—	—	—	—	—	—	RTSECSEL1	RTSECSEL0	PMPTTL	0000

图注：　— = 未实现，读为 0。复位值以十六进制显示。

表 4-16：　　ADC 寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
ADC1BUF0	0300	ADC 数据缓冲区 0																xxxx
ADC1BUF1	0302	ADC 数据缓冲区 1																xxxx
ADC1BUF2	0304	ADC 数据缓冲区 2																xxxx
ADC1BUF3	0306	ADC 数据缓冲区 3																xxxx
ADC1BUF4	0308	ADC 数据缓冲区 4																xxxx
ADC1BUF5	030A	ADC 数据缓冲区 5																xxxx
ADC1BUF6	030C	ADC 数据缓冲区 6																xxxx
ADC1BUF7	030E	ADC 数据缓冲区 7																xxxx
ADC1BUF8	0310	ADC 数据缓冲区 8																xxxx
ADC1BUF9	0312	ADC 数据缓冲区 9																xxxx
ADC1BUFA	0314	ADC 数据缓冲区 10																xxxx
ADC1BUFB	0316	ADC 数据缓冲区 11																xxxx
ADC1BUFC	0318	ADC 数据缓冲区 12																xxxx
ADC1BUFD	031A	ADC 数据缓冲区 13																xxxx
ADC1BUFE	031C	ADC 数据缓冲区 14																xxxx
ADC1BUFF	031E	ADC 数据缓冲区 15																xxxx
AD1CON1	0320	ADON	—	ADSIDL	—	—	—	FORM1	FORM0	SSRC2	SSRC1	SSRC0	—	—	ASAM	SAMP	DONE	0000
AD1CON2	0322	VCFG2	VCFG1	VCFG0	r	—	CSCNA	—	—	BUFS	—	SMPI3	SMPI2	SMPI1	SMPI0	BUFM	ALTS	0000
AD1CON3	0324	ADRC	r	r	SAMC4	SAMC3	SAMC2	SAMC1	SAMC0	ADCS7	ADCS6	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0	0000
AD1CHS	0328	CH0NB	—	—	CH0SB4	CH0SB3	CH0SB2	CH0SB1	CH0SB0	CH0NA	—	—	CH0SA4	CH0SA3	CH0SA2	CH0SA1	CH0SA0	0000
AD1PCFG	032C	PCFG15	PCFG14	PCFG13	PCFG12(1)	PCFG11	PCFG10	PCFG9	PCFG8(1)	PCFG7(1)	PCFG6(1)	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	0000
AD1CSSL	0330	CSSL15	CSSL14	CSSL13	CSSL12(1)	CSSL11	CSSL10	CSSL9	CSSL8(1)	CSSL7(1)	CSSL6(1)	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0	0000

图注：　— = 未实现，读为 0； r = 保留，保持为 0。复位值以十六进制显示。

注　1：　这些位在 28 引脚器件上不可用；读为 0。

表 4-17：　　CTMU 寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
CTMUCON	033C	CTMUEN	—	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	CTTRIG	EDG2POL	EDG2SEL1	EDG2SEL0	EDG1POL	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT	0000
CTMUICON	033E	ITRIM5	ITRIM4	ITRIM3	ITRIM2	ITRIM1	ITRIM0	IRNG1	IRNG0	—	—	—	—	—	—	—	—	0000

图注：　— = 未实现，读为 0。复位值以十六进制显示。

表 4-18：　并行主 / 从端口寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
PMCON	0600	PMPEN	—	PSIDL	ADRMUX1	ADRMUX0	PTBEEN	PTWREN	PTRDEN	CSF1	CSF0	ALP	—	CS1P	BEP	WRSP	RDSP	0000
PMMODE	0602	BUSY	IRQM1	IRQM0	INCM1	INCM0	MODE16	MODE1	MODE0	WAITB1	WAITB0	WAITM3	WAITM2	WAITM1	WAITM0	WAITE1	WAITE0	0000
PMADDR	0604	—	CS1	—	—	—	ADDR10(1)	ADDR9(1)	ADDR8(1)	ADDR7(1)	ADDR6(1)	ADDR5(1)	ADDR4(1)	ADDR3(1)	ADDR2(1)	ADDR1	ADDR0	0000
PMDOUT1		并行端口数据输出寄存器 1（缓冲区 0 和 1）																0000
PMDOUT2	0606	并行端口数据输出寄存器 2（缓冲区 2 和 3）																0000
PMDIN1	0608	并行端口数据输入寄存器 1（缓冲区 0 和 1）																0000
PMDIN2	060A	并行端口数据输入寄存器 2（缓冲区 2 和 3）																0000
PMAEN	060C	—	PTEN14	—	—	—	PTEN10(1)	PTEN9(1)	PTEN8(1)	PTEN7(1)	PTEN6(1)	PTEN5(1)	PTEN4(1)	PTEN3(1)	PTEN2(1)	PTEN1	PTEN0	0000
PMSTAT	060E	IBF	IBOV	—	—	IB3F	IB2F	IB1F	IB0F	OBE	OBUF	—	—	OB3E	OB2E	OB1E	OB0E	0000

图注：　— = 未实现，读为 0。复位值以十六进制显示。

注　1：　这些位在 28 引脚器件上不可用；读为 0。

表 4-19：　实时时钟和日历寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
ALRMVAL	0620	基于 ALRMPTR<1:0> 的闹钟值寄存器窗口																xxxx
ALCFGRPT	0622	ALRMEN	CHIME	AMASK3	AMASK2	AMASK1	AMASK0	ALRMPTR1	ALRMPTR0	ARPT7	ARPT6	ARPT5	ARPT4	ARPT3	ARPT2	ARPT1	ARPT0	0000
RTCVAL	0624	基于 RTCPTR<1:0> 的 RTCC 值寄存器窗口																xxxx
RCFGCAL	0626	RTCEN	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE	RTCPTR1	RTCPTR0	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	xxxx

图注：　— = 未实现，读为 0。复位值以十六进制显示。

表 4-20：　CRC 寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
CRCCON1	0640	CRCEN	—	CSIDL	VWORD4	VWORD3	VWORD2	VWORD1	VWORD0	CRCFUL	CRCMPT	CRCISEL	CRCGO	LENDIAN	—	—	—	0000
CRCCON2	0642	—	—	—	DWIDTH4	DWIDTH3	DWIDTH2	DWIDTH1	DWIDTH0	—	—	—	PLEN4	PLEN3	PLEN2	PLEN1	PLEN0	0000
CRCXORL	0644	X15	X14	X13	X12	X11	X10	X9	X8	X7	X6	X5	X4	X3	X2	X1	—	0000
CRCXORH	0646	X31	X30	X29	X28	X27	X26	X25	X24	X23	X22	X21	X20	X19	X18	X17	X16	0000
CRCDATL	0648	CRC 数据输入寄存器的低位字																xxxx
CRCDATH	064A	CRC 数据输入寄存器的高位字																xxxx
CRCWDATL	064C	CRC 结果寄存器的低位字																xxxx
CRCWDATH	064E	CRC 结果寄存器的高位字																xxxx

图注：　— = 未实现，读为 0。复位值以十六进制显示。

表 4-21：　比较器寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
CMSTAT	0650	CMIDL	—	—	—	—	C3EVT	C2EVT	C1EVT	—	—	—	—	—	C3OUT	C2OUT	C1OUT	0000
CVRCON	0652	—	—	—	—	—	CVREFP	CVREFM1	CVREFM0	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000
CM1CON	0654	CEN	COE	CPOL	—	—	—	CEVT	COUT	EVPOL1	EVPOL0	—	CREF	—	—	CCH1	CCH0	0000
CM2CON	065C	CEN	COE	CPOL	—	—	—	CEVT	COUT	EVPOL1	EVPOL0	—	CREF	—	—	CCH1	CCH0	0000
CM3CON	0664	CEN	COE	CPOL	—	—	—	CEVT	COUT	EVPOL1	EVPOL0	—	CREF	—	—	CCH1	CCH0	0000

图注：　— = 未实现，读为 0。复位值以十六进制显示。

表 4-22：　外设引脚选择寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
RPINR0	0680	—	—	—	INT1R4	INT1R3	INT1R2	INT1R1	INT1R0	—	—	—	—	—	—	—	—	1F00
RPINR1	0682	—	—	—	—	—	—	—	—	—	—	—	INT2R4	INT2R3	INT2R2	INT2R1	INT2R0	001F
RPINR3	0686	—	—	—	T3CKR4	T3CKR3	T3CKR2	T3CKR1	T3CKR0	—	—	—	T2CKR4	T2CKR3	T2CKR2	T2CKR1	T2CKR0	1F1F
RPINR4	0688	—	—	—	T5CKR4	T5CKR3	T5CKR2	T5CKR1	T5CKR0	—	—	—	T4CKR4	T4CKR3	T4CKR2	T4CKR1	T4CKR0	1F1F
RPINR7	068E	—	—	—	IC2R4	IC2R3	IC2R2	IC2R1	IC2R0	—	—	—	IC1R4	IC1R3	IC1R2	IC1R1	IC1R0	1F1F
RPINR8	0690	—	—	—	IC4R4	IC4R3	IC4R2	IC4R1	IC4R0	—	—	—	IC3R4	IC3R3	IC3R2	IC3R1	IC3R0	1F1F
RPINR9	0692	—	—	—	—	—	—	—	—	—	—	—	IC5R4	IC5R3	IC5R2	IC5R1	IC5R0	001F
RPINR11	0696	—	—	—	OCFBR4	OCFBR3	OCFBR2	OCFBR1	OCFBR0	—	—	—	OCFAR4	OCFAR3	OCFAR2	OCFAR1	OCFAR0	1F1F
RPINR18	06A4	—	—	—	U1CTSR4	U1CTSR3	U1CTSR2	U1CTSR1	U1CTSR0	—	—	—	U1RXR4	U1RXR3	U1RXR2	U1RXR1	U1RXR0	1F1F
RPINR19	06A6	—	—	—	U2CTSR4	U2CTSR3	U2CTSR2	U2CTSR1	U2CTSR0	—	—	—	U2RXR4	U2RXR3	U2RXR2	U2RXR1	U2RXR0	1F1F
RPINR20	06A8	—	—	—	SCK1R4	SCK1R3	SCK1R2	SCK1R1	SCK1R0	—	—	—	SDI1R4	SDI1R3	SDI1R2	SDI1R1	SDI1R0	1F1F
RPINR21	06AA	—	—	—	—	—	—	—	—	—	—	—	SS1R4	SS1R3	SS1R2	SS1R1	SS1R0	001F
RPINR22	06AC	—	—	—	SCK2R4	SCK2R3	SCK2R2	SCK2R1	SCK2R0	—	—	—	SDI2R4	SDI2R3	SDI2R2	SDI2R1	SDI2R0	1F1F
RPINR23	06AE	—	—	—	—	—	—	—	—	—	—	—	SS2R4	SS2R3	SS2R2	SS2R1	SS2R0	001F
RPOR0	06C0	—	—	—	RP1R4	RP1R3	RP1R2	RP1R1	RP1R0	—	—	—	RP0R4	RP0R3	RP0R2	RP0R1	RP0R0	0000
RPOR1	06C2	—	—	—	RP3R4	RP3R3	RP3R2	RP3R1	RP3R0	—	—	—	RP2R4	RP2R3	RP2R2	RP2R1	RP2R0	0000
RPOR2	06C4	—	—	—	RP5R4	RP5R3	RP5R2	RP5R1	RP5R0	—	—	—	RP4R4	RP4R3	RP4R2	RP4R1	RP4R0	0000
RPOR3	06C6	—	—	—	RP7R4	RP7R3	RP7R2	RP7R1	RP7R0	—	—	—	RP6R4	RP6R3	RP6R2	RP6R1	RP6R0	0000
RPOR4	06C8	—	—	—	RP9R4	RP9R3	RP9R2	RP9R1	RP9R0	—	—	—	RP8R4	RP8R3	RP8R2	RP8R1	RP8R0	0000
RPOR5	06CA	—	—	—	RP11R4	RP11R3	RP11R2	RP11R1	RP11R0	—	—	—	RP10R4	RP10R3	RP10R2	RP10R1	RP10R0	0000
RPOR6	06CC	—	—	—	RP13R4	RP13R3	RP13R2	RP13R1	RP13R0	—	—	—	RP12R4	RP12R3	RP12R2	RP12R1	RP12R0	0000
RPOR7	06CE	—	—	—	RP15R4	RP15R3	RP15R2	RP15R1	RP15R0	—	—	—	RP14R4	RP14R3	RP14R2	RP14R1	RP14R0	0000
RPOR8(1)	06D0	—	—	—	RP17R4	RP17R3	RP17R2	RP17R1	RP17R0	—	—	—	RP16R4	RP16R3	RP16R2	RP16R1	RP16R0	0000
RPOR9(1)	06D2	—	—	—	RP19R4	RP19R3	RP19R2	RP19R1	RP19R0	—	—	—	RP18R4	RP18R3	RP18R2	RP18R1	RP18R0	0000
RPOR10(1)	06D4	—	—	—	RP21R4	RP21R3	RP21R2	RP21R1	RP21R0	—	—	—	RP20R4	RP20R3	RP20R2	RP20R1	RP20R0	0000
RPOR11(1)	06D6	—	—	—	RP23R4	RP23R3	RP23R2	RP23R1	RP23R0	—	—	—	RP22R4	RP22R3	RP22R2	RP22R1	RP22R0	0000
RPOR12(1)	06D8	—	—	—	RP25R4	RP25R3	RP25R2	RP25R1	RP25R0	—	—	—	RP24R4	RP24R3	RP24R2	RP24R1	RP24R0	0000

图注：　— = 未实现，读为 0。复位值以十六进制显示。

注　1：　这些寄存器在 28 引脚器件上未实现；读为 0。

表 4-23：　系统寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
RCON	0740	TRAPR	IOPUWR	—	—	—	DPSLP	CM	PMSLP	EXTR	SWR	SWDTEN	WDTO	SLEEP	IDLE	BOR	POR	注 1
OSCCON	0742	—	COSC2	COSC1	COSC0	—	NOSC2	NOSC1	NOSC0	CLKLOCK	IOLOCK	LOCK	—	CF	POSCEN	SOSCEN	OSWEN	注 2
CLKDIV	0744	ROI	DOZE2	DOZE1	DOZE0	DOZEN	RCDIV2	RCDIV1	RCDIV0	CPDIV1	CPDIV0	PLLEN	—	—	—	—	—	0100
OSCTUN	0748	—	—	—	—	—	—	—	—	—	—	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0	0000
REFOCON	074E	ROEN	—	ROSSLP	ROSEL	RODIV3	RODIV2	RODIV1	RODIV0	—	—	—	—	—	—	—	—	0000

图注：　— = 未实现，读为 0。复位值以十六进制显示。

注　1：　RCON 寄存器的复位值取决于复位事件的类型。更多信息，请参见**第 6.0 节 “复位”**。

2：　OSCCON 寄存器的复位值取决于复位事件的类型和器件配置。更多信息，请参见**第 8.0 节 “振荡器配置”**。

表 4-24：　深度休眠寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态 (1)
DSCON	758	DSEN	—	—	—	—	—	—	—	—	—	—	—	—	—	DSBOR	RELEASE	0000
DSWAKE	075A	—	—	—	—	—	—	—	DSINT0	DSFLT	—	—	DSWDT	DSRTC	DSMCLR	—	DSPOR	0001
DSGPR0	075C	深度休眠通用寄存器 0																0000
DSGPR1	075E	深度休眠通用寄存器 1																0000

图注：　— = 未实现，读为 0。复位值以十六进制显示。

注　1：　深度休眠寄存器仅在 VDD POR 事件时复位。

表 4-25：　NVM 寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
NVMCON	0760	WR	WREN	WRERR	—	—	—	—	—	—	ERASE	—	—	NVMOP3	NVMOP2	NVMOP1	NVMOP0	0000 (1)
NVMKEY	0766	—	—	—	—	—	—	—	—	NVMKEY 寄存器 <7:0>								0000

图注：　— = 未实现，读为 0。复位值以十六进制显示。

注　1：　显示的复位值仅适用于 POR。其他复位状态下的值取决于复位时对存储器执行写或擦除操作的状态。

表 4-26：　PMD 寄存器映射

寄存器名称	地址	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态
PMD1	0770	T5MD	T4MD	T3MD	T2MD	T1MD	—	—	—	I2C1MD	U2MD	U1MD	SPI2MD	SPI1MD	—	—	ADC1MD	0000
PMD2	0772	—	—	—	IC5MD	IC4MD	IC3MD	IC2MD	IC1MD	—	—	—	OC5MD	OC4MD	OC3MD	OC2MD	OC1MD	0000
PMD3	0774	—	—	—	—	—	CMPMD	RTCCMD	PMPMD	CRCMD	—	—	—	—	—	I2C2MD	—	0000
PMD4	0776	—	—	—	—	—	—	—	—	—	—	—	—	REFOMD	CTMUMD	LVDMD	—	0000

图注：　— = 未实现，读为 0。复位值以十六进制显示。

4.2.5 软件堆栈

除了用作工作寄存器外，PIC24F 器件中的 W15 寄存器也可用作软件堆栈指针。指针总是指向第一个可用的空字，并且从低地址向高地址方向增长。它在弹出堆栈之前递减，而在压入堆栈之后递增，如图 4-4 所示。注意，对于执行任何 CALL 指令时的 PC 压栈，在压入堆栈之前，PC的MSB要进行零扩展，从而确保MSB始终清零。

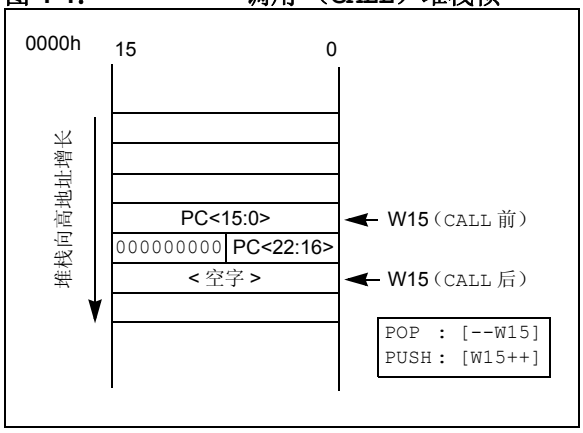
注：	在异常处理期间，在将 PC 压入堆栈之前，要先将 PC 的 MSB 与 SRL 寄存器组合在一起。

堆栈指针限制值 （SPLIM）寄存器与堆栈指针相关联，它设置堆栈上边界的值。 SPLIM 在复位时不会被初始化。与堆栈指针的情况一样， SPLIM<0> 被强制为 0，因为所有的堆栈操作必须是字对齐的。每当使用W15作为源指针或目标指针产生 EA 时，有效地址会与 SPLIM 中的值进行比较。如果堆栈指针 （W15）的内容与 SPLIM 寄存器的内容相等，则会执行压栈操作而不产生堆栈错误陷阱，但在随后的压栈操作时将会产生堆栈错误陷阱。因此，例如如果想要在堆栈增长超过 RAM 中的地址 2000h 时产生堆栈错误陷阱，则需要用值 1FFEh 来初始化 SPLIM。

类似地，当堆栈指针地址小于 0800h 时，就会产生堆栈指针下溢（堆栈错误）陷阱。这可防止堆栈进入特殊功能寄存器 （SFR）空间。

在对 SPLIM 寄存器进行写操作之后，不应紧跟着使用 W15 进行间接读操作的指令。

图 4-4： 调用 （CALL）堆栈帧



4.3 程序存储空间与数据存储空间的接口

PIC24F 架构采用 24 位宽的程序空间和 16 位宽的数据空间。该架构也是一种改进型哈佛结构，这意味着数据也能存放在程序空间内。要成功使用该数据，在访问数据时必须确保这两种存储空间中的信息是对齐的。

除了正常执行外， PIC24F 架构还提供了两种可在操作过程中访问程序空间的方法：

- 使用表指令访问程序空间中任意位置的各个字节或字
- 将程序空间的一部分重映射到数据空间（程序空间可视性）

表指令允许应用程序读写程序存储器的一小块区域。这一功能对于访问需要随时更新的数据表来说非常理想。也可通过表操作访问一个程序字的所有字节。重映射方法允许应用程序访问一大块数据，但只限于读操作，它非常适合于在一个大的静态数据表中进行查找；这一方法只能访问程序字的最低有效字。

4.3.1 对程序空间进行寻址

由于数据空间和程序空间的地址范围分别为 16 位和 24 位，因此需要一个从 16 位数据寄存器创建一个 23 位或 24 位程序地址的方法。方法取决于所采用的接口方式。

对于表操作，使用 8 位的表存储器页地址 （TBLPAG）寄存器定义程序空间内一个 32K 字的区域。这与 16 位 EA 组合形成了一个完整的 24 位程序空间地址。在这种地址形式下， TBLPAG 的最高有效位用来决定操作是发生在用户存储区（TBLPAG<7> = 0）中还是配置存储区（TBLPAG<7> = 1）中。

对于重映射操作，使用 8 位的程序空间可视性页地址（PSVPAG）寄存器定义程序空间中的 16K 字页。当 EA 的最高有效位为 1 时， PSVPAG 与 EA 的低 15 位组合形成一个 23 位的程序空间地址。与表操作不同，重映射操作被严格限制在用户存储区中。

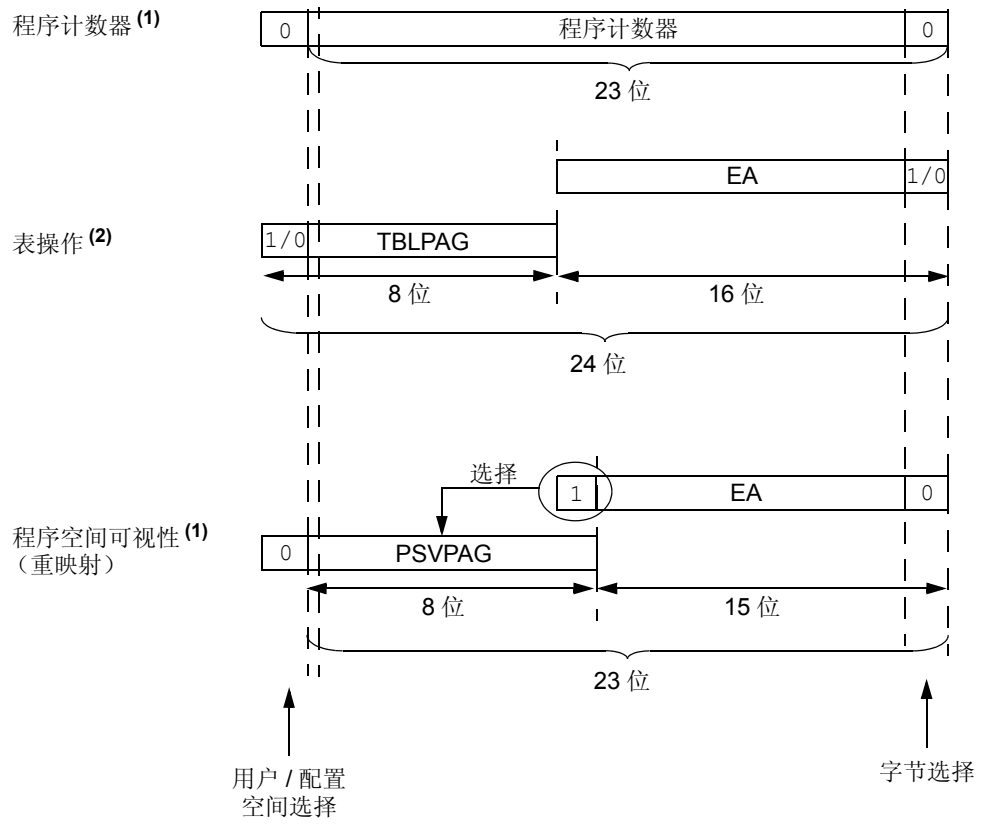
表 4-27 和图 4-5 显示了如何通过表操作和重映射访问来从数据 EA 创建程序 EA。本文中，P<23:0> 指的是程序空间字；而 D<15:0> 指的是数据空间字。

表 4-27：　　程序空间地址构成

访问类型	访问空间	程序空间地址				
		<23>	<22:16>	<15>	<14:1>	<0>
指令访问（代码执行）	用户	0	PC<22:1>			0
		0xx xxxx xxxx xxxx xxxx xxx0				
TBLRD/TBLWT（读 / 写字节或字）	用户	TBLPAG<7:0>		数据 EA<15:0>		
		0xxx xxxx		xxxx xxxx xxxx xxxx		
	配置	TBLPAG<7:0>		数据 EA<15:0>		
		1xxx xxxx		xxxx xxxx xxxx xxxx		
程序空间可视性（块重映射 / 读）	用户	0	PSVPAG<7:0>		数据 EA<14:0>(1)	
		0	xxxx xxxx		xxx xxxx xxxx xxxx	

注　1： 在这种情况下，数据 EA<15> 始终为 1，但并不用它来计算程序空间地址。地址的 bit 15 为 PSVPAG<0>。

图 4-5：　　访问程序空间内数据的地址生成方式



注　1： 程序空间地址的 LSb 始终固定为 0，从而确保程序和数据空间中的数据是字对齐的。

2： 表操作不要求字对齐。允许在配置存储空间中执行表读操作。

4.3.2 使用表指令访问程序存储器中的数据

TBLRDL 和 TBLWTL 指令提供了直接读或写程序空间内任何地址的低位字的方法，无需通过数据空间。TBLRDH 和 TBLWTH 指令是可以将一个程序空间字的高 8 位作为数据读写的唯一方法。

对于每个连续的 24 位程序字，PC 的递增量为 2。这使得程序存储器地址能够直接映射到数据空间地址。于是，程序存储器可以看作是两个 16 位字宽的地址空间，它们并排放置，具有相同的地址范围。TBLRDL 和 TBLWTL 访问包含最低有效数据字的空间，而 TBLRDH 和 TBLWTH 则访问包含最高数据字节的空间。

提供了两条表指令来对程序空间执行字节或字（16 位）大小的数据读写。读和写都可以采用字节或字操作的形式。

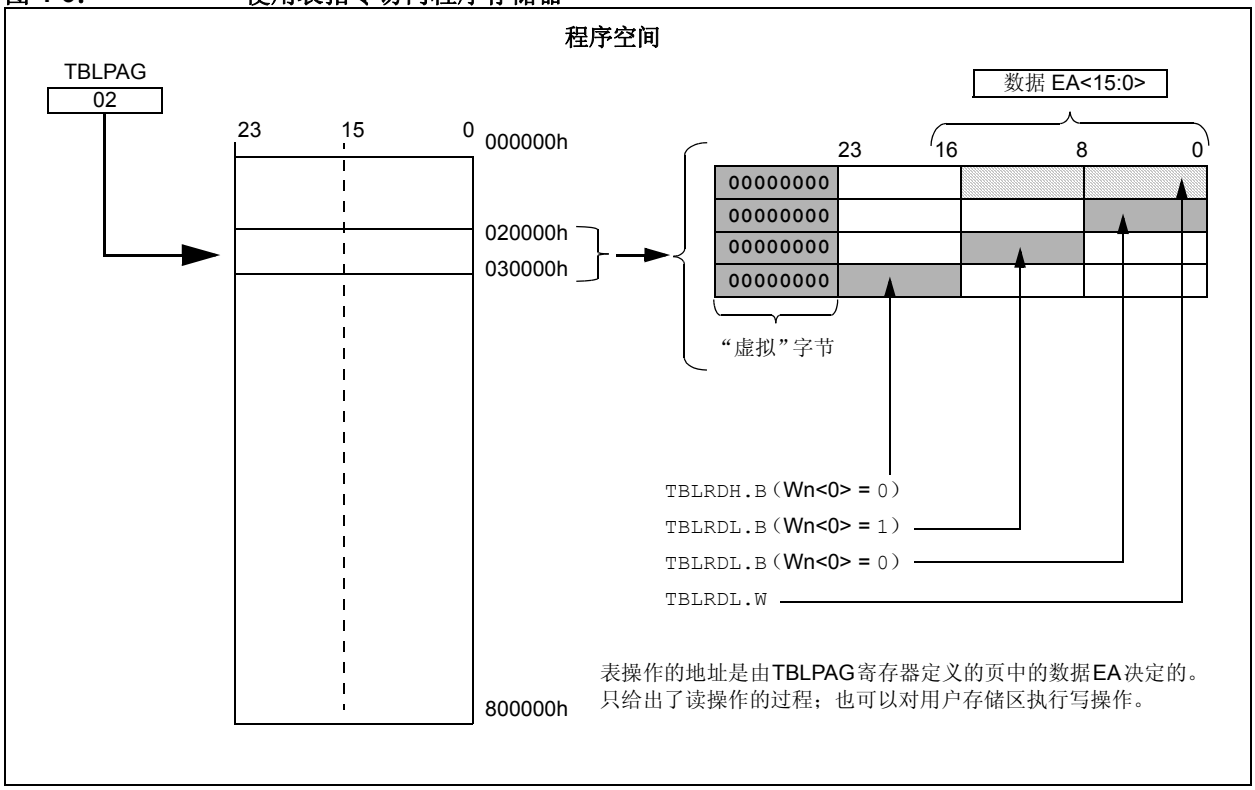
1. TBLRDL（表读低位字）：在字模式下，该指令将程序空间地址的低位字（P<15:0>）映射到数据地址（D<15:0>）中。
   在字节模式下，低位程序字的高字节或低字节被映射到数据地址的低字节中。当字节选择位为 1 时映射高字节；当字节选择位为 0 时映射低字节。
2. TBLRDH（表读高位字）：在字模式下，该指令将程序地址的整个高位字（P<23:16>）映射到数据地址中。注意，D<15:8> 为“虚拟”字节，它始终为 0。
   在字节模式下，该指令将程序字的高字节或低字节映射到数据地址的 D<7:0> 中，就如同 TBLRDL 指令。注意，当选择高位“虚拟”字节（字节选择位 = 1）时，数据将始终为 0。

表指令 TBLWTH 和 TBLWTL 以类似的方式向程序空间地址写入各字节或字。**第 5.0 节“闪存程序存储器”**对这两条指令的详细操作给出了说明。

对于所有的表操作，要访问程序存储空间的哪个区域是由表存储器页地址寄存器（TBLPAG）决定的。TBLPAG 可寻址器件的整个程序存储空间，包括用户和配置空间。当 TBLPAG<7> = 0 时，表页位于用户存储空间中。当 TBLPAG<7> = 1 时，表页位于配置存储空间中。

注：	仅表读操作可在配置存储空间中执行，且只能在已实现区域中执行，例如器件 ID。不允许表写操作。

图 4-6：　使用表指令访问程序存储器



4.3.3 使用程序空间可视性读程序存储器中的数据

可选择将数据空间的高 32 KB 映射到程序空间中的任何 16K 字页中。这提供了通过数据空间对存储的常量数据的透明访问，而无需使用特殊指令（即 TBLRDL/H）。

如果数据空间 EA 的最高有效位（MSb）为 1，并且程序空间可视性使能（方法是将 CPU 内核控制寄存器中的 PSV 位（CORCON<2>）置 1），就能通过数据空间访问程序空间。由程序空间可视性页地址寄存器（PSVPAG）决定要被映射到数据空间中的程序存储空间的位置。这一 8 位的寄存器定义程序空间中 256 个可能的 16K 字页中的任意一个。实际上，PSVPAG 用作程序存储地址的高 8 位，而 EA 的 15 位则用作地址的低位。注意，对于每个程序存储字，PC 都将递增 2，数据空间地址的低15位将直接映射到相应程序空间地址的低 15 位。

将数据读入该区域的指令，需要增加一个额外的指令周期，因为这类指令需要对程序存储器执行两次数据取操作。

尽管大于或等于8000h的每个数据空间地址直接映射到对应的程序存储器地址（见图 4-7），但只使用 24 位程序字的低 16 位来存放数据。任何用作数据存储空间的程序空间存储单元的高 8 位都应编程为 1111 1111 或 0000 0000，以强制为一条 NOP 指令，从而避免了可能出现意外执行这一区域内代码的情况。

注： 在表读 / 写期间，暂时禁止 PSV 访问。

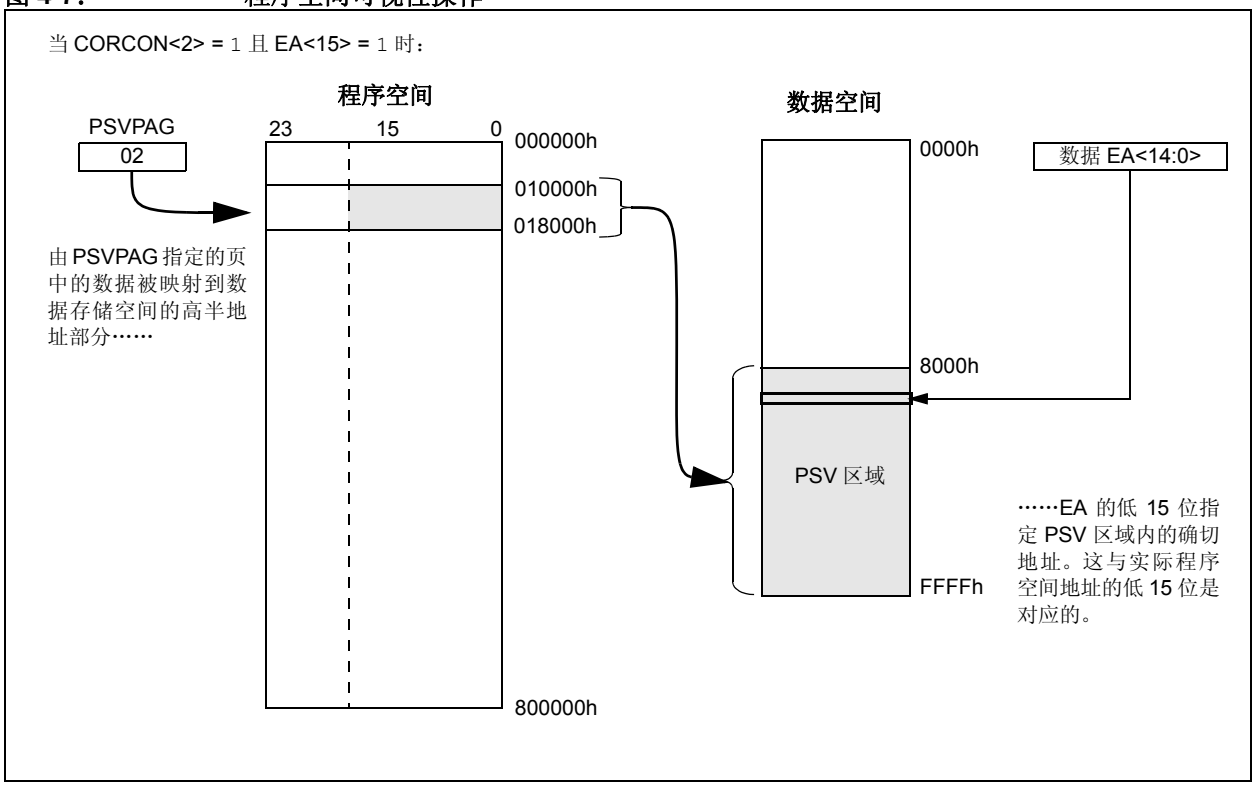
对于使用 PSV 而又在 REPEAT 循环外执行的操作，MOV 和 MOV.D 指令除了规定的执行时间之外，还需要一个额外的指令周期。其他所有指令都需要在规定的执行时间之外额外增加两个指令周期。

对于使用 PSV 而又在 REPEAT 循环内执行的操作，下列情况，除了规定的指令执行时间之外，还需要两个额外的指令周期：

- 在第一次迭代中执行的指令
- 在最后一次迭代中执行的指令
- 由于中断而退出循环之前执行的指令
- 中断得到处理后再次进入循环时执行的指令

REPEAT 循环的所有其他各次迭代，都允许使用 PSV 访问数据的指令在一个周期内执行。

图 4-7： 程序空间可视性操作



5.0 闪存程序存储器

注：	本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 4 章“程序存储器”**（DS39715A_CN）。

PIC24FJ64GA104 系列器件包含用于存储和执行应用程序代码的内部闪存程序存储器。当依靠 2.35V 的 VDD 工作时，该存储器可读、可写、可擦除。（如果禁止稳压器， VDDCORE 必须为 2.25V。）

闪存有以下 4 种编程方式：

- 在线串行编程 （ICSP™）
- 运行时自编程 （Run-Time Self-Programming，RTSP）
- JTAG
- 增强型在线串行编程 （增强型 ICSP）

ICSP 允许在最终的应用电路中对 PIC24FJ64GA104 系列器件进行串行编程。只需要使用 5 根线就可以完成编程，其中编程时钟线和编程数据线（分别命名为 PGECx 和 PGEDx）各一根，其余 3 根分别是电源线 （VDD）、接地线 （VSS）和主复位线 （MCLR）。这允许用户在生产电路板时使用未编程器件，仅在产品交付之前才对单片机进行编程，从而可以使用最新版本的固件或者定制固件进行编程。

使用 TBLRD（表读）和 TBLWT（表写）指令来实现 RTSP。使用 RTSP，用户可以一次将 64 条指令 （192 字节）的数据块写入程序存储器，也可以一次擦除 512 条指令 （1536 字节）。

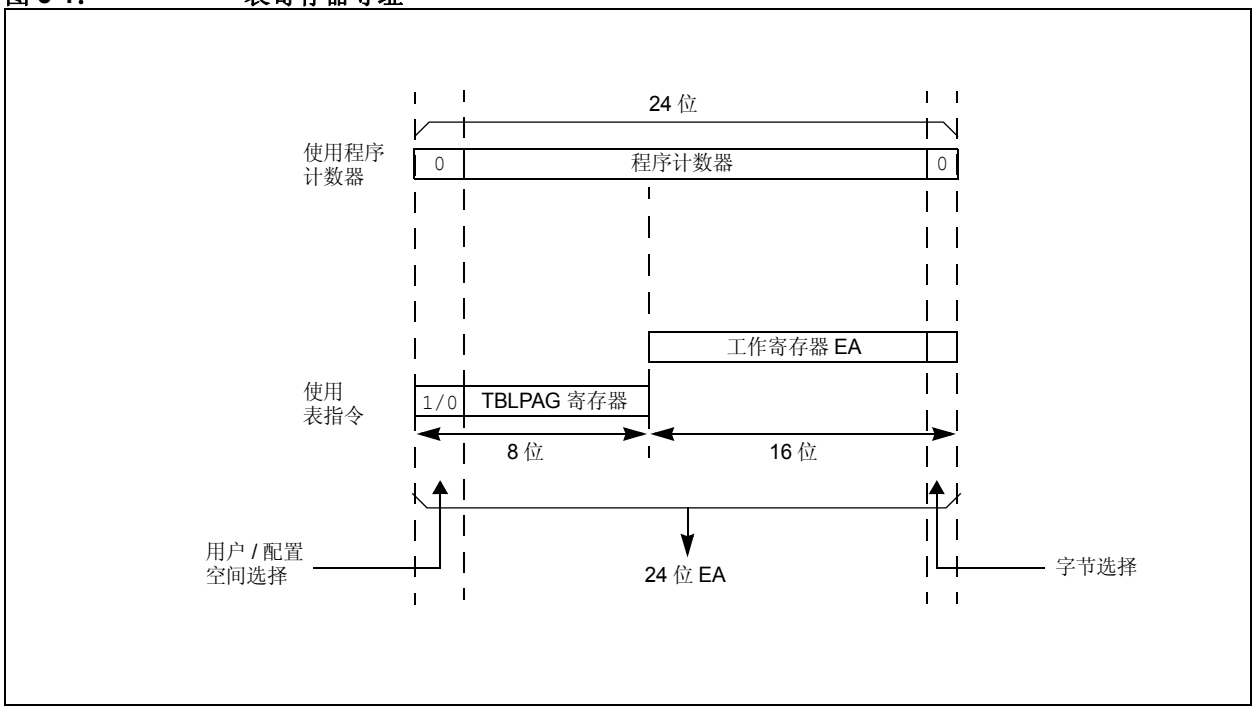
5.1 表指令和闪存编程

闪存的所有编程都是通过表读和表写指令完成的，与使用的编程方法无关。这些指令允许器件在正常工作模式下通过数据存储器直接读写程序存储空间。程序存储器中 24 位目标地址由 TBLPAG<7:0> 位和表指令中指定 W 寄存器中的有效地址 （EA）组成，如图 5-1 所示。

TBLRDL 和 TBLWTL 指令用于读或写程序存储器的 bit<15:0>。TBLRDL 和 TBLWTL 能以字或字节模式访问程序存储器。

TBLRDH 和 TBLWTH 指令用于读或写程序存储器的 bit<23:16>。 TBLRDH 和 TBLWTH 同样能以字或字节模式访问程序存储器。

图 5-1： 表寄存器寻址



5.2 RTSP 工作原理

PIC24F 闪存程序存储器阵列以 64 条指令或 192 字节的行为单位构成。RTSP 允许用户一次擦除 8 行（512 条指令）的块以及一次编程一行。它还可以编程单字。

8行擦除块和单行写入块都是边沿对齐的，从程序存储器起始地址开始，分别到 1536 字节边界和 192 字节边界。

当使用 TBLWT 指令将数据写入程序存储器时，数据并未直接写入存储器。而是将使用表写指令写入的数据存储在保持锁存器中，直到执行编程序列。

可以执行任意数量的 TBLWT 指令，写操作都将成功执行。但是，需要 64 条 TBLWT 指令来写整行存储器。

要确保在写操作期间没有数据被改动，应将所有未使用的地址编程为FFFFFFh。这是因为保持锁存器复位为未知状态，因此如果地址处于复位状态，就可能改写未被重写的行的单元。

RTSP 编程的基本步骤是先建立一个表指针，然后执行一系列 TBLWT 指令来装载缓冲区。通过设置 NVMCON 寄存器中的控制位来执行编程。

可按任何顺序装入数据，且在执行写操作之前可以对保持寄存器进行多次写操作。但后续写操作将覆盖先前的所有写操作。

注：	不建议对同一个存储单元进行多次写操作而不进行擦除操作。

因为只写缓冲区，因此所有表写操作都是单字写操作（2 个指令周期）。编程每行都需要一个编程周期。

5.3 JTAG 操作

PIC24F 系列支持 JTAG 编程和边界扫描。边界扫描可以通过校验引脚到 PCB 的连通性改进制造工艺。编程是通过支持串行向量格式（Serial Vector Format，SVF）的工业标准 JTAG 编程器来执行的。

5.4 增强型在线串行编程

增强型在线串行编程使用片内自举程序（称为编程执行程序）来管理编程过程。使用 SPI 数据帧格式，编程执行程序能够擦除、编程和校验程序存储器。关于增强型 ICSP 的更多信息，请参见器件编程规范。

5.5 控制寄存器

有两个 SFR 用于读写闪存程序存储器：NVMCON 和 NVMKEY。

NVMCON 寄存器（寄存器 5-1）控制要擦除的块、要编程的存储器类型以及编程周期的开始时间。

NVMKEY 是一个只写寄存器，用于写保护。要启动编程或擦除序列，用户必须将 55h 和 AAh 连续写入 NVMKEY 寄存器。更多详细信息，请参见**第 5.6 节“编程操作”**。

5.6 编程操作

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

寄存器 5-1：　　NVMCON：闪存控制寄存器

R/SO-0, HC(1)	R/W-0(1)	R/W-0, HS(1)	U-0	U-0	U-0	U-0	U-0
WR	WREN	WRERR	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0(1)	U-0	U-0	R/W-0(1)	R/W-0(1)	R/W-0(1)	R/W-0(1)
—	ERASE	—	—	NVMOP3(2)	NVMOP2(2)	NVMOP1(2)	NVMOP0(2)
bit 7							bit 0

图注：	SO = 只可置 1 位	HC = 硬件清零位	HS = 硬件置 1 位
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15　**WR：**写控制位 (1)
1 = 启动闪存编程或擦除操作。操作是自定时的，一旦该操作完成，该位即由硬件清零。
0 = 编程或擦除操作完成，并处于停止状态

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

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

bit 12-7　**未实现：**读为 0

bit 6　**ERASE：**擦除 / 编程使能位 (1)
1 = 在下一条 WR 命令时执行 NVMOP<3:0> 指定的擦除操作
0 = 在下一条 WR 命令时执行 NVMOP<3:0> 指定的编程操作

bit 5-4　**未实现：**读为 0

bit 3-0　**NVMOP<3:0>：**NVM 操作选择位 (1,2)
1111 = 存储器批量擦除操作 （ERASE = 1）或无操作 （ERASE = 0） (3)
0011 = 存储器字编程操作 （ERASE = 0）或无操作 （ERASE = 1）
0010 = 存储器页擦除操作 （ERASE = 1）或无操作 （ERASE = 0）
0001 = 存储器行编程操作 （ERASE = 0）或无操作 （ERASE = 1）

注　1： 这些位只能在 POR 时复位。
2： NVMOP<3:0> 的所有其他组合均未实现。
3： 仅在 ICSP™ 模式下可用。请参见器件编程规范。

5.6.1 闪存程序存储器的编程算法

用户一次可编程闪存程序存储器的一行。要实现该操作，必须擦除包含该行在内的一个 8 行大小的块。一般过程如下：

1. 读取程序存储器的 8 行（512 条指令），并将其存储在数据 RAM 中。
2. 用所需的新数据更新 RAM 中的程序数据。
3. 擦除块（见例 5-1）：
   a) 将 NVMOP 位（NVMCON<3:0>）设置为 0010，以配置为块擦除操作。将 ERASE（NVMCON<6>）和 WREN（NVMCON<14>）位置 1。
   b) 将要擦除块的起始地址写入 TBLPAG 和 W 寄存器。
   c) 将 55h 写入 NVMKEY。
   d) 将 AAh 写入 NVMKEY。
   e) 将 WR 位（NVMCON<15>）置 1。擦除周期开始，CPU 暂停，等待擦除周期完成。当擦除操作完成时，WR 位会被自动清零。
4. 将数据 RAM 中的前 64 条指令写入程序存储器缓冲区（见例 5-1）。
5. 将程序块写入闪存：
   a) 将 NVMOP 位设置为 0001，以配置为行编程操作。将 ERASE 位清零，将 WREN 位置 1。
   b) 将 55h 写入 NVMKEY。
   c) 将 AAh 写入 NVMKEY。
   d) 将 WR 位置 1。编程周期开始，CPU 暂停，等待写周期完成。当闪存写操作完成时，WR 位会被自动清零。
6. 将 TBLPAG 中的值递增 1，使用数据 RAM 中下一个 64 条指令的块重复步骤 4 和 5，直到将所有 512 条指令写回闪存。

为防止意外操作，必须向 NVMKEY 写入启动序列，用于允许执行任何擦除或编程操作。在执行编程命令后，用户必须等待一段编程时间，直到编程操作完成。紧跟编程启动序列后面的两条指令应为 NOP，如例 5-5 所示。

例 5-1： 擦除程序存储器块（汇编语言代码）

```
; Set up NVMCON for block erase operation
        MOV     #0x4042, W0                 ;
        MOV     W0, NVMCON                  ; Initialize NVMCON
; Init pointer to row to be ERASED
        MOV     #tblpage(PROG_ADDR), W0     ;
        MOV     W0, TBLPAG                  ; Initialize PM Page Boundary SFR
        MOV     #tbloffset(PROG_ADDR), W0   ; Initialize in-page EA[15:0] pointer
        TBLWTL  W0, [W0]                    ; Set base address of erase block
        DISI    #5                          ; Block all interrupts with priority <7
                                            ; for next 5 instructions
        MOV     #0x55, W0
        MOV     W0, NVMKEY                  ; Write the 55 key
        MOV     #0xAA, W1                   ;
        MOV     W1, NVMKEY                  ; Write the AA key
        BSET    NVMCON, #WR                 ; Start the erase sequence
        NOP                                 ; Insert two NOPs after the erase
        NOP                                 ; command is asserted
```

例 5-2： 擦除程序存储器块 （C 语言代码）

```
// C example using MPLAB C30
    unsigned long progAddr = 0xXXXXXX;       // Address of row to write
    unsigned int offset;

//Set up pointer to the first memory location to be written
    TBLPAG = progAddr>>16;                   // Initialize PM Page Boundary SFR
    offset = progAddr & 0xFFFF;              // Initialize lower word of address

    __builtin_tblwtl(offset, 0x0000);        // Set base address of erase block
                                             // with dummy latch write

    NVMCON = 0x4042;                         // Initialize NVMCON

    asm("DISI #5");                          // Block all interrupts with priority <7
                                             // for next 5 instructions
    __builtin_write_NVM();                   // C30 function to perform unlock
                                             // sequence and set WR
```

例 5-3： 装载写缓冲区 （汇编语言代码）

```
; Set up NVMCON for row programming operations
        MOV     #0x4001, W0                  ;
        MOV     W0, NVMCON                   ; Initialize NVMCON
; Set up a pointer to the first program memory location to be written
; program memory selected, and writes enabled
        MOV     #0x0000, W0                  ;
        MOV     W0, TBLPAG                   ; Initialize PM Page Boundary SFR
        MOV     #0x6000, W0                  ; An example program memory address
; Perform the TBLWT instructions to write the latches
; 0th_program_word
        MOV     #LOW_WORD_0, W2              ;
        MOV     #HIGH_BYTE_0, W3             ;
        TBLWTL  W2, [W0]                     ; Write PM low word into program latch
        TBLWTH  W3, [W0++]                   ; Write PM high byte into program latch
; 1st_program_word
        MOV     #LOW_WORD_1, W2              ;
        MOV     #HIGH_BYTE_1, W3             ;
        TBLWTL  W2, [W0]                     ; Write PM low word into program latch
        TBLWTH  W3, [W0++]                   ; Write PM high byte into program latch
;  2nd_program_word
        MOV     #LOW_WORD_2, W2              ;
        MOV     #HIGH_BYTE_2, W3             ;
        TBLWTL  W2, [W0]                     ; Write PM low word into program latch
        TBLWTH  W3, [W0++]                   ; Write PM high byte into program latch
        •
        •
        •
; 63rd_program_word
        MOV     #LOW_WORD_31, W2             ;
        MOV     #HIGH_BYTE_31, W3            ;
        TBLWTL  W2, [W0]                     ; Write PM low word into program latch
        TBLWTH  W3, [W0]                     ; Write PM high byte into program latch
```

例 5-4：　　装载写缓冲区 （C 语言代码）

```
// C example using MPLAB C30

    #define NUM_INSTRUCTION_PER_ROW 64
    unsigned int offset;
    unsigned int i;
    unsigned long progAddr = 0xXXXXXX;                  // Address of row to write
    unsigned int progData[2*NUM_INSTRUCTION_PER_ROW];   // Buffer of data to write

//Set up NVMCON for row programming
    NVMCON = 0x4001;                                    // Initialize NVMCON

//Set up pointer to the first memory location to be written
    TBLPAG = progAddr>>16;                              // Initialize PM Page Boundary SFR
    offset = progAddr & 0xFFFF;                         // Initialize lower word of address

//Perform TBLWT instructions to write necessary number of latches
for(i=0; i < 2*NUM_INSTRUCTION_PER_ROW; i++)
    {
        __builtin_tblwtl(offset, progData[i++]);        // Write to address low word
        __builtin_tblwth(offset, progData[i]);          // Write to upper byte
        offset = offset + 2;                            // Increment address
    }
```

例 5-5：　　启动编程序列 （汇编语言代码）

```
        DISI    #5                      ; Block all interrupts with priority <7
                                        ; for next 5 instructions
        MOV     #0x55, W0
        MOV     W0, NVMKEY              ; Write the 55 key
        MOV     #0xAA, W1               ;
        MOV     W1, NVMKEY              ; Write the AA key
        BSET    NVMCON, #WR             ; Start the erase sequence
        NOP                             ;
        NOP                             ;
        BTSC    NVMCON, #15             ; and wait for it to be
        BRA     $-2                     ; completed
```

例 5-6：　　启动编程序列 （C 语言代码）

```
// C example using MPLAB C30

asm("DISI #5");           // Block all interrupts with priority < 7
                          // for next 5 instructions

__builtin_write_NVM();    // Perform unlock sequence and set WR
```

5.6.2 编程闪存程序存储器的一个单字

如果已擦除了一个闪存单元，则可用表写指令对该单元进行编程以将一个指令字 （24 位）写入写锁存器。将闪存地址的 8 个最高有效字节装入 TBLPAG 寄存器。TBLWTL 和 TBLWTH 指令将所需数据写入写锁存器，并指定要写入的程序存储器地址的低16位。要将NVMCON寄存器配置为字写操作，应将NVMOP位（NVMCON<3:0>）设置为 0011。通过执行解锁序列并将 WR 位置 1 来执行该写操作 （见例 5-7）。

例 5-7： 编程闪存程序存储器的一个单字 （汇编语言代码）

```
; Setup a pointer to data Program Memory
    MOV     #tblpage(PROG_ADDR), W0       ;
    MOV     W0, TBLPAG                    ;Initialize PM Page Boundary SFR
    MOV     #tbloffset(PROG_ADDR), W0     ;Initialize a register with program memory address

    MOV     #LOW_WORD, W2                 ;
    MOV     #HIGH_BYTE, W3                ;
    TBLWTL  W2, [W0]                      ; Write PM low word into program latch
    TBLWTH  W3, [W0++]                    ; Write PM high byte into program latch

; Setup NVMCON for programming one word to data Program Memory
    MOV     #0x4003, W0                   ;
    MOV     W0, NVMCON                    ; Set NVMOP bits to 0011

    DISI    #5                            ; Disable interrupts while the KEY sequence is written
    MOV     #0x55, W0                     ; Write the key sequence
    MOV     W0, NVMKEY
    MOV     #0xAA, W0
    MOV     W0, NVMKEY
    BSET    NVMCON, #WR                   ; Start the write cycle
    NOP                                   ; Insert two NOPs after the erase
    NOP                                   ; Command is asserted
```

例 5-8： 编程闪存程序存储器的一个单字 （C 语言代码）

```
// C example using MPLAB C30

    unsigned int offset;
    unsigned long progAddr = 0xXXXXXX;          // Address of word to program
    unsigned int progDataL = 0xXXXX;            // Data to program lower word
    unsigned char progDataH = 0xXX;             // Data to program upper byte

//Set up NVMCON for word programming
    NVMCON = 0x4003;                            // Initialize NVMCON

//Set up pointer to the first memory location to be written
    TBLPAG = progAddr>>16;                      // Initialize PM Page Boundary SFR
    offset = progAddr & 0xFFFF;                 // Initialize lower word of address

//Perform TBLWT instructions to write latches
        __builtin_tblwtl(offset, progDataL);    // Write to address low word
        __builtin_tblwth(offset, progDataH);    // Write to upper byte
        asm(“DISI #5”);                         // Block interrupts with priority < 7
                                                // for next 5 instructions
        __builtin_write_NVM();                  // C30 function to perform unlock
                                                // sequence and set WR
```

注：

6.0 复位

注： 本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 7 章“复位”**（DS39712A_CN）。

复位模块结合了所有复位源并控制器件的主复位信号 SYSRST。下面列出了器件的复位源：

- POR：上电复位
- MCLR：引脚复位
- SWR：RESET 指令
- WDT：看门狗定时器复位
- BOR：欠压复位
- CM：配置不匹配复位
- TRAPR：陷阱冲突复位
- IOPUWR：非法操作码复位
- UWR：未初始化的 W 寄存器复位

图 6-1 给出了复位模块的简化框图。

任何有效的复位源都将使 SYSRST 信号有效。许多与 CPU 和外设相关的寄存器均会被强制为已知的复位状态。大多数寄存器都不受复位影响；它们的状态在POR 时未知，而在所有其他复位时不变。

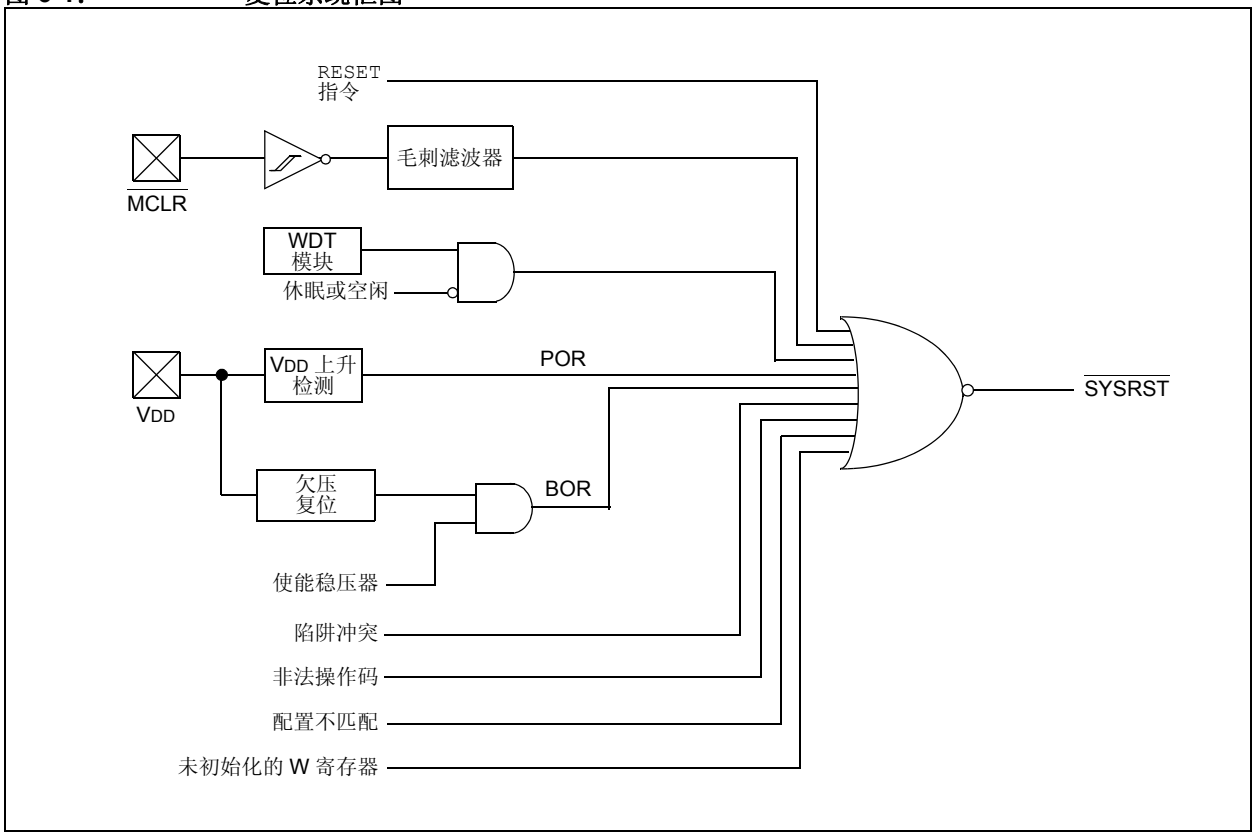
注： 如需了解寄存器复位状态的信息，请参见本手册中的特定外设或 CPU 章节。

任何类型的器件复位都会将RCON寄存器中相应的状态位置 1，以指示复位类型（见寄存器 6-1）。上电复位将清零除 BOR 和 POR 位（RCON<1:0>）之外的所有位，BOR 和 POR 位在 POR 时被置 1。用户可以在代码执行过程中的任何时间置 1 或清零任意位。RCON 寄存器中的位仅用作状态位。用软件将特定的复位状态位置 1 不会导致器件发生复位。

RCON寄存器还包含与看门狗定时器和器件节能状态相关的其他位。本数据手册的其他章节中将讨论这些位的功能。

注： RCON 寄存器中的状态位应该在被读取后清零，这样在器件复位后 RCON 寄存器的下一个值才有意义。

图 6-1： 复位系统框图



寄存器 6-1：　　RCON：复位控制寄存器 (1)

R/W-0	R/W-0	U-0	U-0	U-0	R/CO-0, HS	R/W-0	R/W-0
TRAPR	IOPUWR	—	—	—	DPSLP	CM	PMSLP
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0,	R/W-0	R/W-1	R/W-1
EXTR	SWR	SWDTEN(2)	WDTO	SLEEP	IDLE	BOR	POR
bit 7							bit 0

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

bit 15　**TRAPR**：陷阱复位标志位
1 = 发生了陷阱冲突复位
0 = 未发生陷阱冲突复位

bit 14　**IOPUWR**：非法操作码或访问未初始化的 W 寄存器复位标志位
1 = 检测到非法操作码、非法地址模式或将未初始化的 W 寄存器用作地址指针而导致复位
0 = 未发生非法操作码或未初始化的 W 寄存器复位

bit 13-11　**未实现**：读为 0

bit 10　**DPSLP**：深度休眠模式标志位
1 = 发生了深度休眠
0 = 未发生深度休眠

bit 9　**CM**：配置字不匹配复位标志位
1 = 发生了配置字不匹配复位
0 = 未发生配置字不匹配复位

bit 8　**PMSLP**：程序存储器休眠期间电源位
1 = 程序存储器偏置电压在休眠期间保持供电
0 = 程序存储器偏置电压在休眠期间掉电，稳压器进入待机模式

bit 7　**EXTR**：外部复位（$\overline{\text{MCLR}}$）引脚位
1 = 发生了主复位（引脚）复位
0 = 未发生主复位（引脚）复位

bit 6　**SWR**：软件复位（指令）标志位
1 = 执行了 RESET 指令
0 = 未执行 RESET 指令

bit 5　**SWDTEN**：软件使能 / 禁止 WDT 位 (2)
1 = 使能 WDT
0 = 禁止 WDT

bit 4　**WDTO**：看门狗定时器超时标志位
1 = 发生了 WDT 超时
0 = 未发生 WDT 超时

bit 3　**SLEEP**：从休眠状态唤醒标志位
1 = 器件处于休眠模式
0 = 器件不处于休眠模式

bit 2　**IDLE**：从空闲状态唤醒标志位
1 = 器件处于空闲模式
0 = 器件不处于空闲模式

注　1： 所有复位状态位都可以用软件置 1 或清零。用软件将这些位中的某一位置 1 不会导致器件复位。
2： 如果 FWDTEN 配置位为 1（未编程），则 WDT 始终使能，而与 SWDTEN 位的设置无关。

寄存器 6-1：　　RCON：复位控制寄存器 (1)（续）

bit 1　　**BOR**：欠压复位标志位
1 = 发生了欠压复位。注意 BOR 在上电复位后也将置 1。
0 = 未发生欠压复位

bit 0　　**POR**：上电复位标志位
1 = 发生了上电复位
0 = 未发生上电复位

注　1： 所有复位状态位都可以用软件置 1 或清零。用软件将这些位中的某一位置 1 不会导致器件复位。
2： 如果 FWDTEN 配置位为 1（未编程），则 WDT 始终使能，而与 SWDTEN 位的设置无关。

表 6-1：　　复位标志位操作

标志位	置 1 事件	清零事件
TRAPR（RCON<15>）	陷阱冲突事件	POR
IOPUWR（RCON<14>）	非法操作码或访问了未初始化的 W 寄存器	POR
CM（RCON<9>）	配置不匹配复位	POR
EXTR（RCON<7>）	$\overline{\text{MCLR}}$ 复位	POR
SWR（RCON<6>）	RESET 指令	POR
WDTO（RCON<4>）	WDT 超时	PWRSAV 指令和 POR
SLEEP（RCON<3>）	PWRSAV #SLEEP 指令	POR
IDLE（RCON<2>）	PWRSAV #IDLE 指令	POR
BOR（RCON<1>）	POR 和 BOR	—
POR（RCON<0>）	POR	—
DPSLP（RCON<10>）	PWRSAV #SLEEP 指令且 DSCON <DSEN> 置 1	POR

注：　所有复位标志位均可由用户软件置 1 或清零。

6.1　复位时的时钟源选择

如果使能了时钟切换，器件复位时的系统时钟源选择如表 6-2 中所示。如果禁止了时钟切换，则总是根据振荡器配置位选择系统时钟源。更多详细信息，请参见**第 8.0 节“振荡器配置”**。

表 6-2：　　不同复位类型的振荡器选择（使能时钟切换）

复位类型	确定时钟源的方式
POR	FNOSC 配置位（CW2<10:8>）
BOR	
$\overline{\text{MCLR}}$	COSC 控制位（OSCCON<14:12>）
WDTO	
SWR	

6.2　器件复位时间

表 6-3 总结了各种类型器件复位的复位时间。注意，在 POR 和 PWRT 延时结束后会释放系统复位信号 SYSRST。

器件实际开始执行代码的时间还取决于系统振荡器延时，它包括振荡器起振定时器（OST）延时和 PLL 锁定时间。OST 和 PLL 锁定时间与相应的 SYSRST 延时同时发生。

FSCM 延时决定在 $\overline{\text{SYSRST}}$ 信号释放后 FSCM 开始监视系统时钟源的时间。

表 6-3： 各种器件复位的复位延时

复位类型	时钟源	$\overline{\text{SYSRST}}$ 延时	系统时钟延时	注
POR(6)	EC	TPOR + TRST + TPWRT	—	**1, 2, 3, 8**
	FRC 和 FRCDIV	TPOR + TRST + TPWRT	TFRC	**1, 2, 3, 4, 7, 8**
	LPRC	TPOR + TRST + TPWRT	TLPRC	**1, 2, 3, 4, 8**
	ECPLL	TPOR + TRST + TPWRT	TLOCK	**1, 2, 3, 5, 8**
	FRCPLL	TPOR + TRST + TPWRT	TFRC + TLOCK	**1, 2, 3, 4, 5, 7, 8**
	XT、 HS 和 SOSC	TPOR + TRST + TPWRT	TOST	**1, 2, 3, 6, 8**
	XTPLL 和 HSPLL	TPOR + TRST + TPWRT	TOST + TLOCK	**1, 2, 3, 5, 6, 8**
BOR	EC	TRST + TPWRT	—	**2, 3, 8**
	FRC 和 FRCDIV	TRST + TPWRT	TFRC	**2, 3, 4, 7, 8**
	LPRC	TRST + TPWRT	TLPRC	**2, 3, 4, 8**
	ECPLL	TRST + TPWRT	TLOCK	**2, 3, 5, 8**
	FRCPLL	TRST + TPWRT	TFRC + TLOCK	**2, 3, 4, 5, 7, 8**
	XT、 HS 和 SOSC	TRST + TPWRT	TOST	**2, 3, 6, 8**
	XTPLL 和 HSPLL	TRST + TPWRT	TFRC + TLOCK	**2, 3, 4, 5, 8**
所有其他复位	任何时钟	TRST	—	**2, 8**

注 1： TPOR = 上电复位延时。

2： TRST = 内部状态复位时间。

3： 如果禁止稳压器 （DISVREG 连接到 VDD），则 TPWRT = 64 ms 标称值。

4： TFRC 和 TLPRC = RC 振荡器起振时间。

5： TLOCK = PLL 锁定时间。

6： TOST = 振荡器起振定时器 （OST）延时。 10 位计数器等待 1024 个振荡周期后，才将振荡器时钟释放给系统使用。

7： 如果使能了双速启动，则无论选择何种主振荡器，器件都使用 FRC 启动，并且在这种情况下， FRC 起振时间有效。

8： TRST = 配置设置时间。

注： 关于工作频率和时序规范的详细信息，请参见**第 28.0 节 “电气特性”**。

6.2.1 POR 和长振荡器起振时间

振荡器起振电路及其相关的延时定时器与上电时发生的器件复位延时没有关系。某些晶振电路（尤其是低频晶振）的起振时间会相对较长。因此，在 SYSRST 被释放后，可能会发生以下一种或多种情况：

- 振荡电路未起振。
- 振荡器起振定时器尚未超时 （如果使用了晶振）。
- PLL 未实现锁定 （如果使用了 PLL）。

在有效时钟源供系统使用前，器件不会开始执行代码。因此，如果必须确定复位延时，必须考虑到振荡器和 PLL 起振延时。

6.2.2 故障保护时钟监视器 （FSCM）和器件复位

如果使能了 FSCM，它将在 $\overline{\text{SYSRST}}$ 释放后开始监视系统时钟源。如果此时没有可用的有效时钟源，器件将自动切换为 FRC 振荡器，并且用户可以在陷阱服务程序（Trap Service Routine， TSR）中切换为所需的晶振。

6.3 特殊功能寄存器的复位状态

大多数与 PIC24F CPU 和外设相关的特殊功能寄存器（SFR）会在器件复位时复位为某个特定值。 SFR 是按其外设或 CPU 功能分组的，其复位值在本手册的相应章节有说明。

除了 4 个寄存器外，其他 SFR 的复位值都与复位类型无关。复位控制寄存器 RCON 的复位值取决于器件复位的类型。振荡器控制寄存器 OSCCON 的复位值取决于复位类型和闪存配置字 2（CW2）中 FNOSC 位的编程值；请参见表 6-2。 RCFGCAL 和 NVMCON 寄存器只受 POR 影响。

6.4 深度休眠 BOR （DSBOR）

深度休眠 BOR 是功耗极低的 BOR 电路，在器件处于深度休眠模式时使用。由于电流消耗很低，因此精度会因情况而异。

DSBOR 跳变点约为 2.0V。通过配置 CW4（DSBOREN）= 1 来使能 DSBOR。 DSBOR 会重新激活 POR，以确保当 V_{DD} 降到低于 POR 门限值时器件会发生复位。

注：

7.0 中断控制器

注： 本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 8 章“中断”**（DS39707A_CN）。

PIC24F 中断控制器将诸多外设中断请求信号缩减为一个送往 PIC24F CPU 的中断请求信号。它具有以下特性：

- 最多 8 个处理器异常和软件陷阱
- 7 个可由用户选择的优先级
- 最多 118 个向量的中断向量表 （Interrupt Vector Table， IVT）
- 每个中断或异常源对应一个唯一的向量
- 在指定的用户优先级内具有固定的优先级
- 用于支持调试功能的备用中断向量表 （Alternate Interrupt Vector Table， AIVT）
- 固定的中断进入和返回延时

7.1 中断向量表

中断向量表 （IVT）如图 7-1 所示。 IVT 位于程序存储器中，起始单元地址是 000004h。IVT 包含 126 个向量，由 8 个不可屏蔽陷阱向量和最多 118 个中断源组成。一般来说，每个中断源都有自己的中断向量。每个中断向量都包含一个 24 位宽的地址。每个中断向量单元中设定的值是其相关的中断服务程序 （ISR）的起始地址。

中断向量根据它们的自然优先级区分优先次序；也就是说每个中断向量的优先级与其在向量表中的位置有关。如果其他方面都相同，较低地址的中断向量具有较高的自然优先级。例如，与向量 0 相关的中断比任何其他向量地址的中断具有更高的自然优先级。

PIC24FJ64GA104 系列器件实现了不可屏蔽陷阱和唯一中断。表 7-1 和表 7-2 对此做了总结。

7.1.1 备用中断向量表

备用中断向量表 （AIVT）位于 IVT 之后，如图 7-1 所示。由 ALTIVT 控制位 （INTCON2<15>）控制对 AIVT 的访问。如果 ALTIVT 位置 1，则所有的中断和异常处理都将使用备用向量，而非默认向量。备用向量与默认向量的组织方式相同。

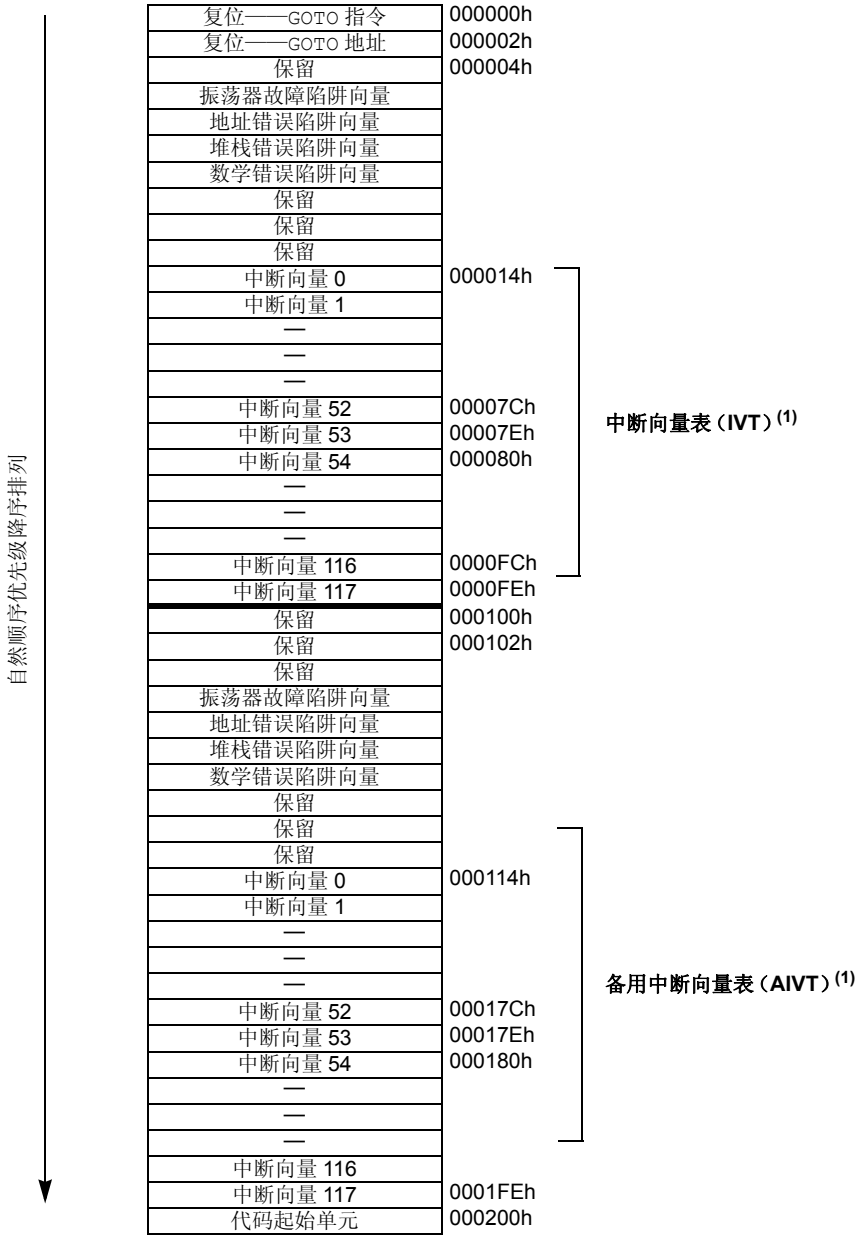
AIVT 通过提供一种不需要将中断向量再编程就可以在应用程序和支持环境之间切换的方法，来支持仿真和调试功能。此特性也支持运行时在不同应用程序之间切换以便评估各种不同的软件算法。如果不需要 AIVT，则应该用 IVT 中使用的相同地址编程 AIVT。

7.2 复位过程

器件复位不是真正的异常，因为复位过程中并不涉及到中断控制器。作为对复位的响应， PIC24F 器件清零其寄存器，同时强制PC为零。然后单片机从地址000000h处开始执行程序。用户可以在复位地址中写入 GOTO 指令，将程序执行重定向到相应的启动程序。

注： 应该使用包含 RESET 指令的默认中断处理程序的入口地址编程 IVT 和 AIVT 中所有未实现或未使用的向量单元。

图 7-1：　　PIC24F 中断向量表



注　1： 请参见表 7-2 了解中断向量列表。

表 7-1：　　陷阱向量详细信息

向量编号	IVT 地址	AIVT 地址	陷阱源
0	000004h	000104h	保留
1	000006h	000106h	振荡器故障
2	000008h	000108h	地址错误
3	00000Ah	00010Ah	堆栈错误
4	00000Ch	00010Ch	数学错误
5	00000Eh	00010Eh	保留
6	000010h	000110h	保留
7	000012h	000112h	保留

表 7-2：　　已实现的中断向量

中断源	向量编号	IVT 地址	AIVT 地址	中断位的位置		
				标志位	允许位	优先级位
ADC1 转换完成	13	00002Eh	00012Eh	IFS0<13>	IEC0<13>	IPC3<6:4>
比较器事件	18	000038h	000138h	IFS1<2>	IEC1<2>	IPC4<10:8>
CRC 发生器	67	00009Ah	00019Ah	IFS4<3>	IEC4<3>	IPC16<14:12>
CTMU 事件	77	0000AEh	0001AEh	IFS4<13>	IEC4<13>	IPC19<6:4>
外部中断 0	0	000014h	000114h	IFS0<0>	IEC0<0>	IPC0<2:0>
外部中断 1	20	00003Ch	00013Ch	IFS1<4>	IEC1<4>	IPC5<2:0>
外部中断 2	29	00004Eh	00014Eh	IFS1<13>	IEC1<13>	IPC7<6:4>
I2C1 主事件	17	000036h	000136h	IFS1<1>	IEC1<1>	IPC4<6:4>
I2C1 从事件	16	000034h	000134h	IFS1<0>	IEC1<0>	IPC4<2:0>
I2C2 主事件	50	000078h	000178h	IFS3<2>	IEC3<2>	IPC12<10:8>
I2C2 从事件	49	000076h	000176h	IFS3<1>	IEC3<1>	IPC12<6:4>
输入捕捉 1	1	000016h	000116h	IFS0<1>	IEC0<1>	IPC0<6:4>
输入捕捉 2	5	00001Eh	00011Eh	IFS0<5>	IEC0<5>	IPC1<6:4>
输入捕捉 3	37	00005Eh	00015Eh	IFS2<5>	IEC2<5>	IPC9<6:4>
输入捕捉 4	38	000060h	000160h	IFS2<6>	IEC2<6>	IPC9<10:8>
输入捕捉 5	39	000062h	000162h	IFS2<7>	IEC2<7>	IPC9<14:12>
输入电平变化通知	19	00003Ah	00013Ah	IFS1<3>	IEC1<3>	IPC4<14:12>
LVD 低压检测	72	0000A4h	0001A4h	IFS4<8>	IEC4<8>	IPC18<2:0>
输出比较 1	2	000018h	000118h	IFS0<2>	IEC0<2>	IPC0<10:8>
输出比较 2	6	000020h	000120h	IFS0<6>	IEC0<6>	IPC1<10:8>
输出比较 3	25	000046h	000146h	IFS1<9>	IEC1<9>	IPC6<6:4>
输出比较 4	26	000048h	000148h	IFS1<10>	IEC1<10>	IPC6<10:8>
输出比较 5	41	000066h	000166h	IFS2<9>	IEC2<9>	IPC10<6:4>
并行主端口	45	00006Eh	00016Eh	IFS2<13>	IEC2<13>	IPC11<6:4>
实时时钟 / 日历	62	000090h	000190h	IFS3<14>	IEC3<14>	IPC15<10:8>
SPI1 错误	9	000026h	000126h	IFS0<9>	IEC0<9>	IPC2<6:4>
SPI1 事件	10	000028h	000128h	IFS0<10>	IEC0<10>	IPC2<10:8>
SPI2 错误	32	000054h	000154h	IFS2<0>	IEC2<0>	IPC8<2:0>
SPI2 事件	33	000056h	000156h	IFS2<1>	IEC2<1>	IPC8<6:4>
Timer1	3	00001Ah	00011Ah	IFS0<3>	IEC0<3>	IPC0<14:12>
Timer2	7	000022h	000122h	IFS0<7>	IEC0<7>	IPC1<14:12>
Timer3	8	000024h	000124h	IFS0<8>	IEC0<8>	IPC2<2:0>
Timer4	27	00004Ah	00014Ah	IFS1<11>	IEC1<11>	IPC6<14:12>
Timer5	28	00004Ch	00014Ch	IFS1<12>	IEC1<12>	IPC7<2:0>
UART1 错误	65	000096h	000196h	IFS4<1>	IEC4<1>	IPC16<6:4>
UART1 接收器	11	00002Ah	00012Ah	IFS0<11>	IEC0<11>	IPC2<14:12>
UART1 发送器	12	00002Ch	00012Ch	IFS0<12>	IEC0<12>	IPC3<2:0>
UART2 错误	66	000098h	000198h	IFS4<2>	IEC4<2>	IPC16<10:8>
UART2 接收器	30	000050h	000150h	IFS1<14>	IEC1<14>	IPC7<10:8>
UART2 发送器	31	000052h	000152h	IFS1<15>	IEC1<15>	IPC7<14:12>

7.3 中断控制和状态寄存器

PIC24FJ64GA104 系列器件实现了以下用于中断控制器的寄存器：

- INTCON1
- INTCON2
- IFS0 至 IFS4
- IEC0 至 IEC4
- IPC0 至 IPC20 （IPC13、 IPC14 和 IPC17 除外）
- INTTREG

INTCON1 和 INTCON2 控制全局中断。INTCON1 包含中断嵌套禁止 （NSTDIS）位以及处理器陷阱源的控制和状态标志。 INTCON2 寄存器控制外部中断请求信号行为以及备用中断向量表的使用。

IFSx 寄存器包含所有中断请求标志。每个中断源都有一个状态位，由各自的外设或外部信号置 1，而由软件清零。

IECx寄存器包含所有中断允许位。这些控制位用于单独允许外设或外部信号中断。

IPCx寄存器用于设置每个中断源的中断优先级。可以为每个用户中断源分配 8 个优先级之一。

中断源按表 7-2 中的向量编号顺序分配给 IFSx、 IECx 和 IPCx 寄存器。例如， INT0 （外部中断 0）具有一个向量编号，自然顺序优先级为 0。所以 INT0IF 状态位在 IFS0<0>中，INT0IE 允许位在IEC0<0>中，INT0IP<2:0> 优先级位在 IPC0 最初的位置 （IPC0<2:0>）中。

尽管两个 CPU 控制寄存器都不是中断控制硬件的特定组成部分，但它们仍包含控制中断功能的位。ALU 状态寄存器 （SR）包含 IPL<2:0> 位 （SR<7:5>）；这些位用于指示当前 CPU 中断优先级。用户可以通过写 IPL 位来更改当前 CPU 优先级。

CORCON 寄存器包含 IPL3 位，这个位与 IPL<2:0> 位一起指示当前 CPU 优先级。 IPL3 是只读位，所以用户软件不能屏蔽陷阱事件。

中断控制器具有中断控制器测试寄存器 （INTTREG），该寄存器显示中断控制器的状态。当产生中断请求时，相关的向量编号和新的中断优先级被锁存到 INTTREG 中。

如果一个通用 ISR 用于多个向量（例如，在自举应用中使用ISR重映射时），该信息可用来确定具体的中断源。它还可以用来检查处于 ISR 时，是否有另一个中断正在等待处理。

在下面各页中的寄存器 7-1 到寄存器 7-32 说明了所有的中断寄存器。

寄存器 7-1：　SR：ALU 状态寄存器 （在 CPU 中）

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R-0
—	—	—	—	—	—	—	DC(1)
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
IPL2(2,3)	IPL1(2,3)	IPL0(2,3)	RA(1)	N(1)	OV(1)	Z(1)	C(1)
bit 7							bit 0

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

bit 7-5　**IPL<2:0>**：CPU 中断优先级状态位 (2,3)

111 = CPU 中断优先级为 7 （15）。禁止用户中断。
110 = CPU 中断优先级为 6 （14）
101 = CPU 中断优先级为 5 （13）
100 = CPU 中断优先级为 4 （12）
011 = CPU 中断优先级为 3 （11）
010 = CPU 中断优先级为 2 （10）
001 = CPU 中断优先级为 1 （9）
000 = CPU 中断优先级为 0 （8）

注　1： 请参见寄存器 3-1 了解其他位的说明，它们并不是专用于中断控制功能。
2： IPL 位与 IPL3 位 （CORCON<3>）组合形成 CPU 中断优先级。如果 IPL3 = 1，那么括号中的值表示中断优先级。
3： 当 NSTDIS （INTCON1<15>） = 1 时， IPL 状态位是只读位。

寄存器 7-2：　CORCON：CPU 控制寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	R/C-0	R/W-0	U-0	U-0
—	—	—	—	IPL3(2)	PSV(1)	—	—
bit 7							bit 0

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

bit 3　**IPL3**：CPU 中断优先级状态位 (2)

1 = CPU 中断优先级大于 7
0 = CPU 中断优先级等于或小于 7

注　1： 请参见寄存器 3-2 了解其他位的说明，它们并不是专用于中断控制功能。
2： IPL3 位与 IPL<2:0> 位 （SR<7:5>）组合形成 CPU 中断优先级。

寄存器 7-3：　　INTCON1：中断控制寄存器 1

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
NSTDIS	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
—	—	—	MATHERR	ADDRERR	STKERR	OSCFAIL	—
bit 7							bit 0

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

bit 15　**NSTDIS**：中断嵌套禁止位
1 = 禁止中断嵌套
0 = 使能中断嵌套

bit 14-5　**未实现**：读为 0

bit 4　**MATHERR**：算术错误陷阱状态位
1 = 发生了溢出陷阱
0 = 未发生溢出陷阱

bit 3　**ADDRERR**：地址错误陷阱状态位
1 = 发生了地址错误陷阱
0 = 未发生地址错误陷阱

bit 2　**STKERR**：堆栈错误陷阱状态位
1 = 发生了堆栈错误陷阱
0 = 未发生堆栈错误陷阱

bit 1　**OSCFAIL**：振荡器故障陷阱状态位
1 = 发生了振荡器故障陷阱
0 = 未发生振荡器故障陷阱

bit 0　**未实现**：读为 0

寄存器 7-4：　　INTCON2：中断控制寄存器 2

R/W-0	R-0	U-0	U-0	U-0	U-0	U-0	U-0
ALTIVT	DISI	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	INT2EP	INT1EP	INT0EP
bit 7							bit 0

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

bit 15　**ALTIVT**：备用中断向量表使能位
1 = 使用备用中断向量表
0 = 使用标准 （默认）向量表

bit 14　**DISI**：DISI 指令状态位
1 = 执行了 DISI 指令
0 = 未执行 DISI 指令

bit 13-3　**未实现**：读为 0

bit 2　**INT2EP**：外部中断 2 边沿检测极性选择位
1 = 下降沿中断
0 = 上升沿中断

bit 1　**INT1EP**：外部中断 1 边沿检测极性选择位
1 = 下降沿中断
0 = 上升沿中断

bit 0　**INT0EP**：外部中断 0 边沿检测极性选择位
1 = 下降沿中断
0 = 上升沿中断

寄存器 7-5：　　IFS0：中断标志状态寄存器 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	AD1IF	U1TXIF	U1RXIF	SPI1IF	SPF1IF	T3IF
bit 15							bit 8

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
T2IF	OC2IF	IC2IF	—	T1IF	OC1IF	IC1IF	INT0IF
bit 7							bit 0

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

bit 15-14　**未实现：**读为 0

bit 13　**AD1IF：**A/D 转换完成中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 12　**U1TXIF：**UART1 发送器中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 11　**U1RXIF：**UART1 接收器中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 10　**SPI1IF：**SPI1 事件中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 9　**SPF1IF：**SPI1 故障中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 8　**T3IF：**Timer3 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 7　**T2IF：**Timer2 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 6　**OC2IF：**输出比较通道 2 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 5　**IC2IF：**输入捕捉通道 2 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 4　**未实现：**读为 0

bit 3　**T1IF：**Timer1 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 2　**OC1IF：**输出比较通道 1 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 1　**IC1IF：**输入捕捉通道 1 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 0　**INT0IF：**外部中断 0 标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

寄存器 7-6：　　IFS1：中断标志状态寄存器 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
U2TXIF	U2RXIF	INT2IF	T5IF	T4IF	OC4IF	OC3IF	—
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT1IF	CNIF	CMIF	MI2C1IF	SI2C1IF
bit 7							bit 0

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

bit 15　**U2TXIF**：UART2 发送器中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 14　**U2RXIF**：UART2 接收器中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 13　**INT2IF**：外部中断 2 标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 12　**T5IF**：Timer5 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 11　**T4IF**：Timer4 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 10　**OC4IF**：输出比较通道 4 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 9　**OC3IF**：输出比较通道 3 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 8-5　**未实现**：读为 0

bit 4　**INT1IF**：外部中断 1 标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 3　**CNIF**：输入电平变化通知中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 2　**CMIF**：比较器中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 1　**MI2C1IF**：I2C1 主事件中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 0　**SI2C1IF**：I2C1 从事件中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

寄存器 7-7：　　IFS2：中断标志状态寄存器 2

U-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0	U-0
—	—	PMPIF	—	—	—	OC5IF	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
IC5IF	IC4IF	IC3IF	—	—	—	SPI2IF	SPF2IF
bit 7							bit 0

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

bit 15-14　**未实现：**读为 0

bit 13　**PMPIF：**并行主端口中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 12-10　**未实现：**读为 0

bit 9　**OC5IF：**输出比较通道 5 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 8　**未实现：**读为 0

bit 7　**IC5IF：**输入捕捉通道 5 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 6　**IC4IF：**输入捕捉通道 4 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 5　**IC3IF：**输入捕捉通道 3 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 4-2　**未实现：**读为 0

bit 1　**SPI2IF：**SPI2 事件中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 0　**SPF2IF：**SPI2 故障中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

寄存器 7-8：　　IFS3：中断标志状态寄存器 3

U-0	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0
—	RTCIF	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	R/W-0,	R/W-0	U-0
—	—	—	—	—	MI2C2IF	SI2C2IF	—
bit 7							bit 0

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

bit 15　**未实现：**读为 0

bit 14　**RTCIF：**实时时钟 / 日历中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 13-3　**未实现：**读为 0

bit 2　**MI2C2IF：**I2C2 主事件中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 1　**SI2C2IF：**I2C2 从事件中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 0　**未实现：**读为 0

寄存器 7-9：　　IFS4：中断标志状态寄存器 4

U-0	U-0	R/W-0	U-0	U-0	U-0	U-0	R/W-0
—	—	CTMUIF	—	—	—	—	LVDIF
bit 15							bit 8

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	U-0
—	—	—	—	CRCIF	U2ERIF	U1ERIF	—
bit 7							bit 0

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

bit 15-14　**未实现：**读为 0

bit 13　**CTMUIF：**CTMU 中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 12-9　**未实现：**读为 0

bit 8　**LVDIF：**低压检测中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 7-4　**未实现：**读为 0

bit 3　**CRCIF：**CRC 发生器中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 2　**U2ERIF：**UART2 错误中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 1　**U1ERIF：**UART1 错误中断标志状态位
1 = 产生了中断请求
0 = 未产生中断请求

bit 0　**未实现：**读为 0

寄存器 7-10：　　IEC0：中断允许控制寄存器 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	AD1IE	U1TXIE	U1RXIE	SPI1IE	SPF1IE	T3IE
bit 15							bit 8

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
T2IE	OC2IE	IC2IE	—	T1IE	OC1IE	IC1IE	INT0IE
bit 7							bit 0

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

bit 15-14　**未实现：**读为 0

bit 13　**AD1IE：**A/D 转换完成中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 12　**U1TXIE：**UART1 发送器中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 11　**U1RXIE：**UART1 接收器中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 10　**SPI1IE：**SPI1 传输完成中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 9　**SPF1IE：**SPI1 故障中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 8　**T3IE：**Timer3 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 7　**T2IE：**Timer2 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 6　**OC2IE：**输出比较通道 2 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 5　**IC2IE：**输入捕捉通道 2 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 4　**未实现：**读为 0

bit 3　**T1IE：**Timer1 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 2　**OC1IE：**输出比较通道 1 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 1　**IC1IE：**输入捕捉通道 1 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 0　**INT0IE：**外部中断 0 允许位
1 = 允许中断请求
0 = 禁止中断请求

寄存器 7-11： IEC1：中断允许控制寄存器 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
U2TXIE	U2RXIE	INT2IE(1)	T5IE	T4IE	OC4IE	OC3IE	—
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	INT1IE(1)	CNIE	CMIE	MI2C1IE	SI2C1IE
bit 7							bit 0

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

bit 15 **U2TXIE**：UART2 发送器中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 14 **U2RXIE**：UART2 接收器中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 13 **INT2IE**：外部中断 2 允许位 (1)
1 = 允许中断请求
0 = 禁止中断请求

bit 12 **T5IE**：Timer5 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 11 **T4IE**：Timer4 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 10 **OC4IE**：输出比较通道 4 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 9 **OC3IE**：输出比较通道 3 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 8-5 **未实现**：读为 0

bit 4 **INT1IE**：外部中断 1 允许位 (1)
1 = 允许中断请求
0 = 禁止中断请求

bit 3 **CNIE**：输入电平变化通知中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 2 **CMIE**：比较器中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 1 **MI2C1IE**：I2C1 主事件中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 0 **SI2C1IE**：I2C1 从事件中断允许位
1 = 允许中断请求
0 = 禁止中断请求

注 1： 如果允许外部中断，中断输入还必须配置给可用的 RPn 或 PRIx 引脚。更多信息，请参见**第 10.4 节“外设引脚选择（PPS）”**。

寄存器 7-12：　　IEC2：中断允许控制寄存器 2

U-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0	U-0
—	—	PMPIE	—	—	—	OC5IE	—
bit 15							bit 8

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
IC5IE	IC4IE	IC3IE	—	—	—	SPI2IE	SPF2IE
bit 7							bit 0

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

bit 15-14　**未实现：**读为 0

bit 13　**PMPIE：**并行主端口中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 12-10　**未实现：**读为 0

bit 9　**OC5IE：**输出比较通道 5 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 8　**未实现：**读为 0

bit 7　**IC5IE：**输入捕捉通道 5 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 6　**IC4IE：**输入捕捉通道 4 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 5　**IC3IE：**输入捕捉通道 3 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 4-2　**未实现：**读为 0

bit 1　**SPI2IE：**SPI2 事件中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 0　**SPF2IE：**SPI2 故障中断允许位
1 = 允许中断请求
0 = 禁止中断请求

寄存器 7-13：　　IEC3：中断允许控制寄存器 3

U-0	R/W-0	U-0	U-0	U-0	U-0	U-0	U-0
—	RTCIE	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	U-0
—	—	—	—	—	MI2C2IE	SI2C2IE	—
bit 7							bit 0

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

bit 15　**未实现：**读为 0

bit 14　**RTCIE：**实时时钟 / 日历中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 13-3　**未实现：**读为 0

bit 2　**MI2C2IE：**I2C2 主事件中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 1　**SI2C2IE：**I2C2 从事件中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 0　**未实现：**读为 0

寄存器 7-14：　　IEC4：中断允许控制寄存器 4

U-0	U-0	R/W-0	U-0	U-0	U-0	U-0	R/W-0
—	—	CTMUIE	—	—	—	—	LVDIE
bit 15							bit 8

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	U-0
—	—	—	—	CRCIE	U2ERIE	U1ERIE	—
bit 7							bit 0

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

bit 15-14　**未实现：**读为 0

bit 13　**CTMUIE：**CTMU 中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 12-9　**未实现：**读为 0

bit 8　**LVDIE：**低压检测中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 7-4　**未实现：**读为 0

bit 3　**CRCIE：**CRC 发生器中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 2　**U2ERIE：**UART2 错误中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 1　**U1ERIE：**UART1 错误中断允许位
1 = 允许中断请求
0 = 禁止中断请求

bit 0　**未实现：**读为 0

寄存器 7-15：　　IPC0：中断优先级控制寄存器 0

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	T1IP2	T1IP1	T1IP0	—	OC1IP2	OC1IP1	OC1IP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	IC1IP2	IC1IP1	IC1IP0	—	INT0IP2	INT0IP1	INT0IP0
bit 7							bit 0

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

bit 15　**未实现：**读为 0

bit 14-12　**T1IP<2:0>：**Timer1 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 11　**未实现：**读为 0

bit 10-8　**OC1IP<2:0>：**输出比较通道 1 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 7　**未实现：**读为 0

bit 6-4　**IC1IP<2:0>：**输入捕捉通道 1 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 3　**未实现：**读为 0

bit 2-0　**INT0IP<2:0>：**外部中断 0 优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

寄存器 7-16：　　IPC1：中断优先级控制寄存器 1

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	T2IP2	T2IP1	T2IP0	—	OC2IP2	OC2IP1	OC2IP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	IC2IP2	IC2IP1	IC2IP0	—	—	—	—
bit 7							bit 0

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

bit 15　**未实现：**读为 0

bit 14-12　**T2IP<2:0>：**Timer2 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 11　**未实现：**读为 0

bit 10-8　**OC2IP<2:0>：**输出比较通道 2 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 7　**未实现：**读为 0

bit 6-4　**IC2IP<2:0>：**输入捕捉通道 2 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 3-0　**未实现：**读为 0

寄存器 7-17：　　IPC2：中断优先级控制寄存器 2

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	U1RXIP2	U1RXIP1	U1RXIP0	—	SPI1IP2	SPI1IP1	SPI1IP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	SPF1IP2	SPF1IP1	SPF1IP0	—	T3IP2	T3IP1	T3IP0
bit 7							bit 0

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

bit 15　**未实现：**读为 0

bit 14-12　**U1RXIP<2:0>：**UART1 接收器中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 11　**未实现：**读为 0

bit 10-8　**SPI1IP<2:0>：**SPI1 事件中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 7　**未实现：**读为 0

bit 6-4　**SPF1IP<2:0>：**SPI1 故障中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 3　**未实现：**读为 0

bit 2-0　**T3IP<2:0>：**Timer3 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

寄存器 7-18：　　IPC3：中断优先级控制寄存器 3

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	AD1IP2	AD1IP1	AD1IP0	—	U1TXIP2	U1TXIP1	U1TXIP0
bit 7							bit 0

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

bit 15-7　**未实现：**读为 0

bit 6-4　**AD1IP<2:0>：**A/D 转换完成中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 3　**未实现：**读为 0

bit 2-0　**U1TXIP<2:0>：**UART1 发送器中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

寄存器 7-19：　　IPC4：中断优先级控制寄存器 4

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	CNIP2	CNIP1	CNIP0	—	CMIP2	CMIP1	CMIP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	MI2C1IP2	MI2C1IP1	MI2C1IP0	—	SI2C1IP2	SI2C1IP1	SI2C1IP0
bit 7							bit 0

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

bit 15　**未实现：**读为 0

bit 14-12　**CNIP<2:0>：**输入电平变化通知中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 11　**未实现：**读为 0

bit 10-8　**CMIP<2:0>：**比较器中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 7　**未实现：**读为 0

bit 6-4　**MI2C1IP<2:0>：**I2C1 主事件中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 3　**未实现：**读为 0

bit 2-0　**SI2C1IP<2:0>：**I2C1 从事件中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

寄存器 7-20：　　IPC5：中断优先级控制寄存器 5

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R/W-0
—	—	—	—	—	INT1IP2	INT1IP1	INT1IP0
bit 7							bit 0

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

bit 15-3　**未实现：**读为 0

bit 2-0　**INT1IP<2:0>：**外部中断 1 优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

寄存器 7-21：　　IPC6：中断优先级控制寄存器 6

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	T4IP2	T4IP1	T4IP0	—	OC4IP2	OC4IP1	OC4IP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	OC3IP2	OC3IP1	OC3IP0	—	—	—	—
bit 7							bit 0

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

bit 15　**未实现：**读为 0

bit 14-12　**T4IP<2:0>：**Timer4 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 11　**未实现：**读为 0

bit 10-8　**OC4IP<2:0>：**输出比较通道 4 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 7　**未实现：**读为 0

bit 6-4　**OC3IP<2:0>：**输出比较通道 3 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 3-0　**未实现：**读为 0

寄存器 7-22：　　IPC7：中断优先级控制寄存器 7

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	U2TXIP2	U2TXIP1	U2TXIP0	—	U2RXIP2	U2RXIP1	U2RXIP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	INT2IP2	INT2IP1	INT2IP0	—	T5IP2	T5IP1	T5IP0
bit 7							bit 0

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

bit 15　**未实现：**读为 0

bit 14-12　**U2TXIP<2:0>：**UART2 发送器中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 11　**未实现：**读为 0

bit 10-8　**U2RXIP<2:0>：**UART2 接收器中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 7　**未实现：**读为 0

bit 6-4　**INT2IP<2:0>：**外部中断 2 优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 3　**未实现：**读为 0

bit 2-0　**T5IP<2:0>：**Timer5 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

寄存器 7-23：　　IPC8：中断优先级控制寄存器 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	SPI2IP2	SPI2IP1	SPI2IP0	—	SPF2IP2	SPF2IP1	SPF2IP0
bit 7							bit 0

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

bit 15-7　**未实现：**读为 0

bit 6-4　**SPI2IP<2:0>：**SPI2 事件中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 3　**未实现：**读为 0

bit 2-0　**SPF2IP<2:0>：**SPI2 故障中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

寄存器 7-24：　　IPC9：中断优先级控制寄存器 9

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	IC5IP2	IC5IP1	IC5IP0	—	IC4IP2	IC4IP1	IC4IP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	IC3IP2	IC3IP1	IC3IP0	—	—	—	—
bit 7							bit 0

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

bit 15　**未实现：**读为 0

bit 14-12　**IC5IP<2:0>：**输入捕捉通道 5 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 11　**未实现：**读为 0

bit 10-8　**IC4IP<2:0>：**输入捕捉通道 4 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 7　**未实现：**读为 0

bit 6-4　**IC3IP<2:0>：**输入捕捉通道 3 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 3-0　**未实现：**读为 0

寄存器 7-25：　　IPC10：中断优先级控制寄存器 10

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	OC5IP2	OC5IP1	OC5IP0	—	—	—	—
bit 7							bit 0

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

bit 15-7　**未实现：**读为 0

bit 6-4　**OC5IP<2:0>：**输出比较通道 5 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 3-0　**未实现：**读为 0

寄存器 7-26：　　IPC11：中断优先级控制寄存器 11

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	PMPIP2	PMPIP1	PMPIP0	—	—	—	—
bit 7							bit 0

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

bit 15-7　**未实现：**读为 0

bit 6-4　**PMPIP<2:0>：**并行主端口中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 3-0　**未实现：**读为 0

寄存器 7-27： IPC12：中断优先级控制寄存器 12

U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R/W-0
—	—	—	—	—	MI2C2IP2	MI2C2IP1	MI2C2IP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	SI2C2IP2	SI2C2IP1	SI2C2IP0	—	—	—	—
bit 7							bit 0

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

bit 15-11 **未实现：**读为 0

bit 10-8 **MI2C2IP<2:0>：**I2C2 主事件中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 7 **未实现：**读为 0

bit 6-4 **SI2C2IP<2:0>：**I2C2 从事件中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 3-0 **未实现：**读为 0

寄存器 7-28：　　IPC15：中断优先级控制寄存器 15

U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R/W-0
—	—	—	—	—	RTCIP2	RTCIP1	RTCIP0
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

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

bit 15-11　**未实现：**读为 0

bit 10-8　**RTCIP<2:0>：**实时时钟 / 日历中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 7-0　**未实现：**读为 0

寄存器 7-29：　　IPC16：中断优先级控制寄存器 16

U-0	R/W-1	R/W-0	R/W-0	U-0	R/W-1	R/W-0	R/W-0
—	CRCIP2	CRCIP1	CRCIP0	—	U2ERIP2	U2ERIP1	U2ERIP0
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	U1ERIP2	U1ERIP1	U1ERIP0	—	—	—	—
bit 7							bit 0

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

bit 15　**未实现：**读为 0

bit 14-12　**CRCIP<2:0>：**CRC 发生器错误中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 11　**未实现：**读为 0

bit 10-8　**U2ERIP<2:0>：**UART2 错误中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 7　**未实现：**读为 0

bit 6-4　**U1ERIP<2:0>：**UART1 错误中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 3-0　**未实现：**读为 0

寄存器 7-30：　　IPC18：中断优先级控制寄存器 18

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R/W-0
—	—	—	—	—	LVDIP2	LVDIP1	LVDIP0
bit 7							bit 0

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

bit 15-3　**未实现：**读为 0

bit 2-0　**LVDIP<2:0>：**低压检测中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

寄存器 7-31：　　IPC19：中断优先级控制寄存器 19

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	R/W-1	R/W-0	R/W-0	U-0	U-0	U-0	U-0
—	CTMUIP2	CTMUIP1	CTMUIP0	—	—	—	—
bit 7							bit 0

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

bit 15-7　**未实现：**读为 0

bit 6-4　**CTMUIP<2:0>：**CTMU 中断优先级位
111 = 中断优先级为 7（最高优先级中断）
•
•
•
001 = 中断优先级为 1
000 = 禁止中断源

bit 3-0　**未实现：**读为 0

寄存器 7-32：　　INTTREG：中断控制和状态寄存器

R-0	U-0	R/W-0	U-0	R-0	R-0	R-0	R-0
CPUIRQ	—	VHOLD	—	ILR3	ILR2	ILR1	ILR0
bit 15							bit 8

U-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
—	VECNUM6	VECNUM5	VECNUM4	VECNUM3	VECNUM2	VECNUM1	VECNUM0
bit 7							bit 0

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

bit 15　**CPUIRQ**：中断控制器 CPU 中断请求位
1 = 产生了中断请求，但 CPU 尚未响应；当 CPU 优先级高于中断优先级时会发生这种情况
0 = 没有未响应的中断请求

bit 14　未实现：读为 0

bit 13　**VHOLD**：向量编号捕捉配置位
1 = VECNUM 位包含优先级最高的待处理中断的向量值
0 = VECNUM 位包含上次响应的中断的向量值 （即，即使有其他中断待处理，捕捉的仍然是上次产生的优先级高于 CPU 的中断）

bit 12　未实现：读为 0

bit 11-8　**ILR<3:0>**：新的 CPU 中断优先级位
1111 = CPU 中断优先级为 15
•
•
•
0001 = CPU 中断优先级为 1
0000 = CPU 中断优先级为 0

bit 7　未实现：读为 0

bit 6-0　**VECNUM<6:0>**：待处理中断向量 ID 位 （待处理中断的向量编号为 VECNUM + 8）
0111111 = 待处理中断的向量编号为 135
•
•
•
0000001 = 待处理中断的向量编号为 9
0000000 = 待处理中断的向量编号为 8

7.4 中断设置过程

7.4.1 初始化

要配置中断源：

1. 如果不需要嵌套中断，则将 NSTDIS 控制位（INTCON1<15>）置 1。
2. 通过写相应 IPCx 寄存器中的控制位为中断源选择由用户分配的优先级。优先级将取决于具体的应用和中断源类型。如果不需要多个优先级，则可以将所有允许中断源的 IPCx 寄存器控制位编程为相同的非零值。

注：	在器件复位时，IPCx 寄存器被初始化，为所有用户中断源分配优先级 4。

3. 将相应 IFSx 寄存器中与外设相关的中断标志状态位清零。
4. 通过将相应 IECx 寄存器中与中断源相关的中断允许控制位置 1 来允许中断源。

7.4.2 中断服务程序

如何声明ISR以及怎样使用正确的向量地址初始化IVT，取决于编程语言（即，C 语言或汇编语言）和用于开发应用程序的语言开发工具包。一般情况下，用户必须将相应 IFSx 寄存器中与 ISR 处理的中断源相对应的中断标志清零。否则，在退出 ISR 程序后应用程序将立即再次进入 ISR。如果 ISR 用汇编语言编码，则必须使用 RETFIE 指令结束 ISR，以便将保存的 PC 值、SRL 值和原先的 CPU 优先级弹出堆栈。

7.4.3 陷阱服务程序

除了必须清零 INTCON1 寄存器中相应的陷阱状态标志来避免重新进入陷阱服务程序（TSR）之外，TSR 使用与 ISR 类似的方式编写。

7.4.4 中断禁止

可以使用以下步骤禁止所有用户中断：

1. 使用 PUSH 指令将当前的 SR 值压入软件堆栈。
2. 通过将值 OEh 与 SRL 进行逻辑或运算来强制将 CPU 的优先级设置为 7。

要允许用户中断，则可以使用POP指令恢复先前的SR值。

注意，只能禁止优先级小于等于 7 的用户中断。不能禁止陷阱源 （优先级为 8-15）。

使用 DISI 指令可以方便地将优先级为 1-6 的中断禁止一段固定的时间。 DISI 指令不能禁止优先级为 7 的中断源。

注：

8.0 振荡器配置

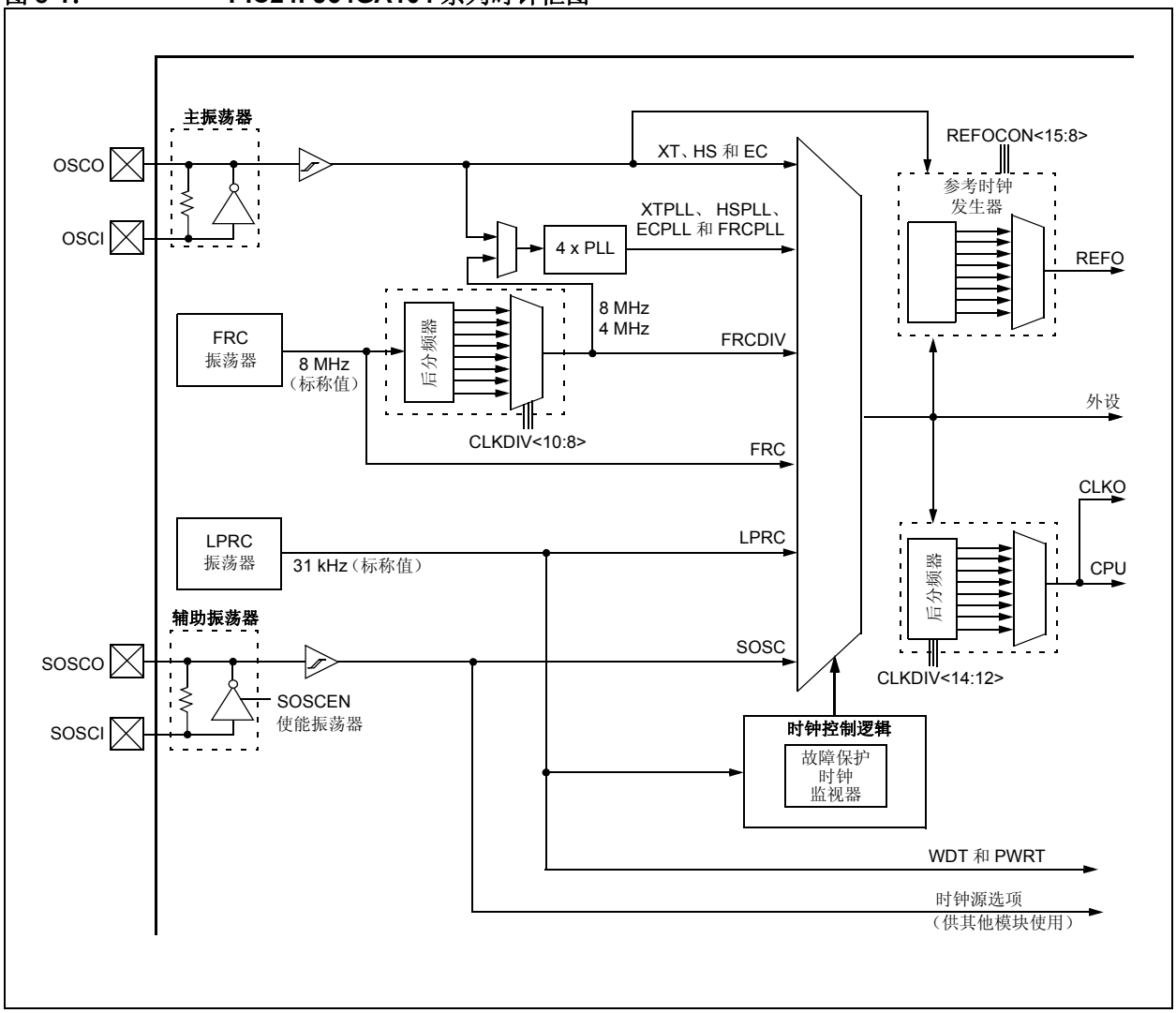
注： 本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 6 章“振荡器”**（DS39700A_CN）。

PIC24FJ64GA104 系列器件的振荡器系统具有以下特性：

- 可选择4个外部和内部振荡器作为时钟源，提供11种不同的时钟模式
- 具有片上 4x PLL，可基于所选的内部和外部振荡器源升高内部工作频率
- 可通过软件控制在各种时钟源之间切换
- 可通过软件控制预分频器有选择地为 CPU 提供时钟，以节省系统功耗
- 具有故障保护时钟监视器（Fail-Safe Clock Monitor，FSCM），可检测时钟故障，以使应用安全地恢复或关闭
- 用于同步外部硬件的单独和独立配置的系统时钟输出

图 8-1 给出了振荡器系统的简化框图。

图 8-1： PIC24FJ64GA104 系列时钟框图



8.1 CPU 时钟机制

系统时钟源可以由以下 4 种之一提供：

- OSCI 和 OSCO 引脚上的主振荡器 （POSC）
- SOSCI 和 SOSCO 引脚上的辅助振荡器（SOSC）
- 内部快速 RC （FRC）振荡器
- 内部低功耗 RC （LPRC）振荡器

主振荡器和 FRC 源可以选择使用内部 4x PLL。FRC 时钟源的频率可选择通过可编程时钟分频器降低。选定的时钟源将产生处理器和外设的时钟源。

处理器时钟源需进行二分频，以产生内部指令周期时钟 FCY。在本文档中，指令周期时钟也表示为 FOSC/2。内部指令周期时钟 FOSC/2 可以在 OSCO I/O 引脚提供，用于主振荡器的一些工作模式。

8.2 POR 时的初始配置

在发生器件上电复位事件时使用的振荡器源 （以及工作模式）使用配置位设置进行选择。振荡器配置位在程序存储器的配置寄存器中进行设置 （更多详细信息，请参见 **第 25.1 节“配置位”**）。主振荡器配置位 POSCMD<1:0>（配置字2<1:0>）和初始振荡器选择配置位FNOSC<2:0>（配置字 2<10:8>）用于选择在上电复位时使用的振荡器源。默认情况下 （未编程时）将选择带后分频器的 FRC 主振荡器 （FRCDIV）。通过编程这些位，可以选择辅助振荡器或一个内部振荡器。

这些配置位使用户可以选择多种时钟模式，如表8-1所示。

8.2.1 时钟切换模式配置位

FCKSM 配置位（配置字 2<7:6>）一起用于配置器件时钟切换和故障保护时钟监视器 （FSCM） 。只有将 FCKSM1 编程 （0）时，才会使能时钟切换。只有同时将 FCKSM<1:0> 位编程 （00）时，才会使能 FSCM。

表 8-1： 时钟选择的配置位值

振荡器模式	振荡器源	POSCMD1: POSCMD0	FNOSC2: FNOSC0	注
带后分频器的快速 RC 振荡器 （FRCDIV）	内部	11	111	**1, 2**
（保留）	内部	xx	110	**1**
低功耗 RC 振荡器 （LPRC）	内部	11	101	**1**
辅助 （Timer1）振荡器 （SOSC）	辅助	11	100	**1**
带 PLL 模块的主振荡器 （XT） （XTPLL）	主	01	011	
带 PLL 模块的主振荡器 （EC） （ECPLL）	主	00	011	
主振荡器 （HS）	主	10	010	
主振荡器 （XT）	主	01	010	
主振荡器 （EC）	主	00	010	
带 PLL 模块的快速 RC 振荡器 （FRCPLL）	内部	11	001	**1**
快速 RC 振荡器 （FRC）	内部	11	000	**1**

注 1： OSCO 引脚功能由 OSCIOFCN 配置位决定。

2： 这是未编程 （已擦除）器件的默认振荡器模式。

8.3 控制寄存器

振荡器的操作由 3 个特殊功能寄存器控制：

- OSCCON
- CLKDIV
- OSCTUN

OSCCON 寄存器 （寄存器 8-1）是振荡器的主控制寄存器。它监视时钟源并控制时钟源切换。

CLKDIV 寄存器 （寄存器 8-2）控制与打盹模式相关的功能，以及 FRC 振荡器的后分频器。

OSCTUN 寄存器 （寄存器 8-3）使用户可以对 FRC 振荡器在大约 ±12% 的范围内进行微调。每个位的递增或递减都会将 FRC 振荡器的出厂校准频率改变一个固定的量。

寄存器 8-1： OSCCON：振荡器控制寄存器

U-0	R-0	R-0	R-0	U-0	R/W-x(1)	R/W-x(1)	R/W-x(1)
—	COSC2	COSC1	COSC0	—	NOSC2	NOSC1	NOSC0
bit 15							bit 8

R/SO-0	R/W-0	R-0(3)	U-0	R/CO-0	R/W-0	R/W-0	R/W-0
CLKLOCK	IOLOCK(2)	LOCK	—	CF	POSCEN	SOSCEN	OSWEN
bit 7							bit 0

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

bit 15 **未实现：**读为 0

bit 14-12 **COSC<2:0>：**当前振荡器选择位
111 = 带后分频器的快速 RC 振荡器 （FRCDIV）
110 = 保留
101 = 低功耗 RC 振荡器 （LPRC）
100 = 辅助振荡器 （SOSC）
011 = 带 PLL 模块的主振荡器 （XTPLL、 HSPLL 和 ECPLL）
010 = 主振荡器 （XT、 HS 和 EC）
001 = 带后分频器和 PLL 模块的快速 RC 振荡器 （FRCPLL）
000 = 快速 RC 振荡器 （FRC）

bit 11 **未实现：**读为 0

bit 10-8 **NOSC<2:0>：**新振荡器选择位 (1)
111 = 带后分频器的快速 RC 振荡器 （FRCDIV）
110 = 保留
101 = 低功耗 RC 振荡器 （LPRC）
100 = 辅助振荡器 （SOSC）
011 = 带 PLL 模块的主振荡器 （XTPLL、 HSPLL 和 ECPLL）
010 = 主振荡器 （XT、 HS 和 EC）
001 = 带后分频器和 PLL 模块的快速 RC 振荡器 （FRCPLL）
000 = 快速 RC 振荡器 （FRC）

注 1： 这些位的复位值由 FNOSC 配置位决定。

2： 只能在执行解锁序列后更改 IOLOCK 位的状态。此外，如果 IOL1WAY 配置位为 1，一旦 IOLOCK 位置 1，就不能清零。

3： 在任意有效时钟切换期间，或者每当选择了非 PLL 时钟模式时，也复位为 0。

寄存器 8-1：　　OSCCON：振荡器控制寄存器 （续）

bit 7　**CLKLOCK**：时钟选择锁定使能位
如果使能 FSCM （FCKSM1 = 1）：
1 = 时钟和 PLL 选择被锁定
0 = 时钟和 PLL 选择未锁定，可以通过将 OSWEN 位置 1 进行修改
如果禁止 FSCM （FCKSM1 = 0）：
时钟和 PLL 选择始终未锁定，可以通过将 OSWEN 位置 1 进行修改。

bit 6　**IOLOCK**：I/O 锁定使能位 **(2)**
1 = I/O 锁定工作
0 = I/O 锁定不工作

bit 5　**LOCK**：PLL 锁定状态位 **(3)**
1 = PLL 模块处于锁定状态，或 PLL 模块起振定时器延时结束
0 = PLL 模块处于失锁状态， PLL 起振定时器正在运行或 PLL 被禁止

bit 4　**未实现**：读为 0

bit 3　**CF**：时钟故障检测位
1 = FSCM 检测到时钟故障
0 = 未检测到时钟故障

bit 2　**POSCEN**：主振荡器休眠使能位
1 = 主振荡器在休眠模式下继续工作
0 = 主振荡器在休眠模式下被禁止

bit 1　**SOSCEN**：32 kHz 辅助振荡器 （SOSC）使能位
1 = 使能辅助振荡器
0 = 禁止辅助振荡器

bit 0　**OSWEN**：振荡器切换使能位
1 = 启动振荡器切换，切换为由 NOSC<2:0> 位指定的时钟源
0 = 振荡器切换完成

注　1： 这些位的复位值由 FNOSC 配置位决定。
2： 只能在执行解锁序列后更改 IOLOCK 位的状态。此外，如果 IOL1WAY 配置位为 1，一旦 IOLOCK 位置 1，就不能清零。
3： 在任意有效时钟切换期间，或者每当选择了非 PLL 时钟模式时，也复位为 0。

寄存器 8-2：　　CLKDIV：时钟分频比寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
ROI	DOZE2	DOZE1	DOZE0	DOZEN(1)	RCDIV2	RCDIV1	RCDIV0
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

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

bit 15　**ROI**：中断恢复位
1 = 发生中断时清零 DOZEN 位，并将 CPU 和外设的时钟比复位为 1:1
0 = 中断对 DOZEN 位没有影响

bit 14-12　**DOZE<2:0>**：CPU 和外设的时钟比选择位
111 = 1:128
110 = 1:64
101 = 1:32
100 = 1:16
011 = 1:8
010 = 1:4
001 = 1:2
000 = 1:1

bit 11　**DOZEN**：打盹使能位 (1)
1 = DOZE<2:0> 位指定 CPU 和外设的时钟比
0 = CPU 和外设的时钟比设置为 1:1

bit 10-8　**RCDIV<2:0>**：FRC 后分频比选择位
111 = 31.25 kHz （256 分频）
110 = 125 kHz （64 分频）
101 = 250 kHz （32 分频）
100 = 500 kHz （16 分频）
011 = 1 MHz （8 分频）
010 = 2 MHz （4 分频）
001 = 4 MHz （2 分频）
000 = 8 MHz （1 分频）

bit 7-0　**未实现**：读为 0

注　1：该位在 ROI 位置 1 和发生中断时自动清零。

寄存器 8-3：　　OSCTUN：FRC 振荡器调节寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	TUN5(1)	TUN4(1)	TUN3(1)	TUN2(1)	TUN1(1)	TUN0(1)
bit 7							bit 0

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

bit 15-6　**未实现：**读为 0

bit 5-0　**TUN<5:0>：**FRC 振荡器调节位

011111 = 最大频率偏移
011110 =
•
•
•
000001 =
000000 = 中心频率，振荡器以出厂校准频率工作
111111 =
•
•
•
100001 =
100000 = 最小频率偏移

注　1： TUN<5:0> 的递增或递减不能在 FRC 调节范围内同步更改 FRC 频率，并且频率的变化也可能不是单调的。

8.4　时钟切换工作原理

在软件控制下，应用可以在任何时候在四个时钟源（POSC、SOSC、FRC 和 LPRC）之间自由切换，几乎没有什么限制。为限制这种灵活性可能产生的负面影响，PIC24F 器件在切换过程中带有安全锁定。

注：　主振荡器模式有三种不同的子模式（XT、HS 和 EC），它们由 POSCMDx 配置位决定。在应用中可以用软件实现从主振荡器模式切换到其他模式，或从其他模式切换到主振荡器模式，但不能在不对器件进行再编程的情况下在主振荡器模式的不同子模式之间进行切换。

8.4.1　使能时钟切换

要使能时钟切换，CW2 中的 FCKSM 配置位必须编程为 00。（更多详细信息，请参见**第 25.1 节 “配置位”**。）如果 FCKSM 配置位未被编程 （1x），则时钟切换功能和故障保护时钟监视器功能被禁止。这是默认设置。

当时钟切换被禁止时，NOSCx控制位（OSCCON<10:8>）不控制时钟选择。但是，COSCx 位（OSCCON<14:12>）反映由 FNOSCx 配置位选择的时钟源。

在时钟切换被禁止时，OSWEN 控制位（OSCCON<0>）不起作用。它始终保持为 0。

8.4.2 振荡器切换过程

执行时钟切换需要以下基本过程：

1. 如果需要，读 COSCx 位 （OSCCON<14:12>）以确定当前的振荡器源。
2. 执行解锁序列以允许写入 OSCCON 寄存器的高字节。
3. 将适当的值写入 NOSCx 位（OSCCON<10:8>）选择新振荡器源。
4. 执行解锁序列以允许写入 OSCCON 寄存器的低字节。
5. 将 OSWEN 位置 1 以启动振荡器切换。

一旦基本过程完成，系统时钟硬件将自动进行如下响应：

1. 时钟切换硬件将 NOSCx 位的新值与 COSCx 位进行比较。如果它们相同，则时钟切换是冗余操作。在这种情况下， OSWEN 位自动清零，时钟切换中止。
2. 如果启动了有效的时钟切换，则 LOCK （OSCCON<5>）和 CF （OSCCON<3>）位清零。
3. 如果新振荡器现在不在运行，则硬件会将其启动。如果必须要启动晶振，则硬件将等待到 OST 延时结束。如果新的振荡器源使用 PLL，则硬件将等待到检测到 PLL 锁定 （LOCK = 1）。
4. 硬件会等待新时钟源的 10 个时钟周期，然后执行时钟切换。
5. 硬件清零 OSWEN 位表示时钟切换成功。此外，NOSCx 位的值被传送到 COSCx 位。
6. 此时旧时钟源被关闭，LPRC（如果WDT或FSCM使能）或 SOSC（如果 SOSCEN 保持置 1）除外。

注 1： 在整个时钟切换过程中，处理器将继续执行代码。对时序敏感的代码不应在此时执行。

2： 不允许直接在使能 PLL 的任何主振荡器模式和 FRCPLL 模式之间进行时钟切换。这适用于任何方向下的时钟切换。在这些情况下，应用必须首先切换到 FRC 模式将其作为两个 PLL 模式之间的过渡时钟源。

时钟切换的建议代码序列如下：

1. 禁止在 OSCCON 寄存器解锁和写序列期间中断。
2. 在两条连续的指令中将 78h 和 9Ah 写入 OSCCON<15:8>，以执行 OSCCON 高字节的解锁序列。
3. 在紧接解锁序列之后的指令中将新的振荡器源写入 NOSCx 位。
4. 在两条连续的指令中将 46h 和 57h 写入 OSCCON<7:0>，以执行 OSCCON 低字节的解锁序列。
5. 在紧接解锁序列之后的指令中将OSWEN位置1。
6. 继续执行对时钟不敏感的代码 （可选）。
7. 调用合适的软件延时 （周期计数），以使选定的振荡器和 / 或 PLL 启动并稳定。
8. 检查 OSWEN 是否为 0。如果为 0，则说明切换成功。如果 OSWEN 仍然置 1，则检查 LOCK 位以确定故障的原因。

例 8-1 中显示了解锁 OSCCON 寄存器和启动时钟切换的核心序列。

例 8-1： 时钟切换的基本代码序列

```
;Place the new oscillator selection in W0
;OSCCONH (high byte) Unlock Sequence
MOV         #OSCCONH, w1
MOV         #0x78, w2
MOV         #0x9A, w3
MOV.b       w2, [w1]
MOV.b       w3, [w1]
;Set new oscillator selection
MOV.b       WREG, OSCCONH
;OSCCONL (low byte) unlock sequence
MOV         #OSCCONL, w1
MOV         #0x46, w2
MOV         #0x57, w3
MOV.b       w2, [w1]
MOV.b       w3, [w1]
;Start oscillator switch operation
BSET        OSCCON,#0
```

8.5 辅助振荡器 （SOSC）

8.5.1 基本 SOSC 操作

要使用辅助振荡器，PIC24FJ64GA104 系列器件不一定要将 SOSCEN 位置 1。任何需要 SOSC 的模块 （例如 RTCC、Timer1 或 DSWDT），在需要时钟信号时都会自动开启 SOSC。但是 SOSC 的起振时间很长。为了避免对于外设启动产生延时，可以使用 SOSCEN 位手动启动 SOSC。

要使用辅助振荡器，SOSCSEL<1:0> 位（CW3<9:8>）必须配置为处于振荡器模式——11或01。将SOSCSEL设置为 00 时， SOSC 引脚配置为数字模式，从而使能引脚上的数字 I/O 功能。如果 SOSC 配置为处于任一振荡器模式，则数字功能将不可用。

8.5.2 低功耗 SOSC 操作

根据器件配置，辅助振荡器可以在两种不同的功耗级别下工作。在低功耗模式下，振荡器以低驱动能力、低功耗状态工作。默认情况下，振荡器使用较高的驱动能力，因而会产生较多功耗。辅助振荡器模式配置位 SOSCSEL<1:0> （CW3<9:8>）决定振荡器的功耗模式。将 SOSCSEL 位编程为 01 即可选择低功耗工作。

该模式的驱动能力较低，使 SOSC 对于噪声较为敏感，需要较长的起振时间。使用低功耗模式时，必须小心处理 SOSC 电路的设计和布线，以确保振荡器正确起振和振荡。

8.5.3 外部 （数字）时钟模式 （SCLKI）

SOSC 还可以配置为使用外部的 32 kHz 时钟源运行，而不是使用内部振荡器。在该模式 （也称为数字模式）下，在 SCLKI 引脚上提供的时钟源用于为所有配置为使用辅助振荡器的模块提供时钟。在该模式下，晶振驱动电路被禁止， SOSCEN 位 （OSCCON<1>）不起任何作用。

8.5.4 SOSC 布线注意事项

由于低引脚数器件 （例如 PIC24FJ64GA104 系列中的那些器件）所存在的引脚排列限制， SOSC 比其他 PIC24F 器件更易受噪声影响。除非正确地处理 SOSC 电路的设计和布线，否则这种外部噪声可能会影响振荡器周期的精度。

通常，晶振电路连线应尽可能短。使用接地环或地平面包围晶振电路也是很好的做法。关于晶振电路设计的更多信息，请参见 《PIC24F 系列参考手册》的**第 6 章 “振荡器”**（DS39700A_CN）。以下 Microchip 应用笔记也提供了更多信息：

- *AN826，“Crystal Oscillator Basics and Crystal Selection for rfPIC® and PICmicro® Devices”* （DS00826）
- *AN849，“Basic PICmicro® Oscillator Design”* （DS00849）。

8.6 参考时钟输出

除了某些振荡器模式中可用的 CLKO 输出 （FOSC/2）外，PIC24FJ64GA104 系列器件中的器件时钟也可以配置为向端口引脚提供参考时钟输出信号。该功能在所有振荡器配置中都可用，允许用户选择更大范围的时钟分频比来驱动应用中的外部器件。

该参考时钟输出由 REFOCON 寄存器（寄存器 8-4）控制。将 ROEN 位 （REFOCON<15>）置 1，将使时钟信号在 REFO 引脚上可用。RODIV 位（REFOCON<11:8>）允许选择 16 种不同的时钟分频比。

ROSSLP 和 ROSEL 位 （REFOCON<13:12>）控制休眠模式下参考输出的可用性。ROSEL 位决定用 OSC1 和 OSC2 上的振荡器还是当前系统时钟源作为参考时钟输出。ROSSLP 位决定器件处于休眠模式时 REFO 上的参考时钟源是否可用。

要在休眠模式下使用参考时钟输出， ROSSLP 和 ROSEL 位都必须置 1。器件时钟也必须配置为主模式之一（EC、HS 或 XT）；否则，如果 POSCEN 位没有同时置 1，OSC1 和 OSC2 上的振荡器将在器件进入休眠模式后掉电。清零 ROSEL 位允许参考输出频率在任何时钟切换期间随着系统时钟的改变而改变。

寄存器 8-4：　REFOCON：参考振荡器控制寄存器

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ROEN	—	ROSSLP	ROSEL	RODIV3	RODIV2	RODIV1	RODIV0
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

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

bit 15　**ROEN**：参考振荡器输出使能位
1 = 在 REFO 引脚上使能参考振荡器
0 = 禁止参考振荡器

bit 14　**未实现**：读为 0

bit 13　**ROSSLP**：参考振荡器输出在休眠模式下停止的位
1 = 参考振荡器在休眠模式下继续运行
0 = 参考振荡器在休眠模式下被禁止

bit 12　**ROSEL**：参考振荡器源选择位
1 = 主振荡器用作基本时钟。注意，必须使用 FOSC<2:0> 位使能晶振；在休眠模式下，晶振会保持工作。
0 = 系统时钟用作基本时钟；基本时钟反映器件的任何时钟切换

bit 11-8　**RODIV<3:0>**：参考振荡器分频比选择位
1111 = 基本时钟值被 32,768 分频
1110 = 基本时钟值被 16,384 分频
1101 = 基本时钟值被 8,192 分频
1100 = 基本时钟值被 4,096 分频
1011 = 基本时钟值被 2,048 分频
1010 = 基本时钟值被 1,024 分频
1001 = 基本时钟值被 512 分频
1000 = 基本时钟值被 256 分频
0111 = 基本时钟值被 128 分频
0110 = 基本时钟值被 64 分频
0101 = 基本时钟值被 32 分频
0100 = 基本时钟值被 16 分频
0011 = 基本时钟值被 8 分频
0010 = 基本时钟值被 4 分频
0001 = 基本时钟值被 2 分频
0000 = 基本时钟值

bit 7-0　**未实现**：读为 0

注：

9.0 节能特性

注：	本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 39 章“深度休眠的节能特性”**（DS39727A_CN）。

PIC24FJ64GA104 系列器件提供了管理功耗的功能，该功能是通过有选择地管理 CPU 和外设的时钟源来实现的。一般来说，较低的时钟频率和减少时钟源驱动电路的数目可降低功耗。所有 PIC24F 器件可通过以下四种不同的方式管理功耗：

- 时钟频率
- 基于指令的休眠、空闲和深度休眠模式
- 软件控制的打盹模式
- 用软件有选择地进行外设控制

可以组合使用这些方法从而在保证关键应用功能（如对于时序敏感的通信）的情况下有选择地调节应用的功耗。

9.1 时钟频率和时钟切换

PIC24F器件提供的时钟频率范围较宽，用户可根据应用需要进行选择。如果未锁定系统时钟配置，用户只需更改 NOSC 位即可选择低功耗或高精度振荡器。在工作期间更改系统时钟的过程以及相应的限制，已在**第 8.0 节“振荡器配置”**中进行了更详细的讨论。

9.2 基于指令的节能模式

PIC24F 器件有两种特殊的节能模式，通过执行特殊的 PWRSAV 指令可以进入这两种模式。休眠模式下时钟停止工作并暂停所有代码执行；空闲模式下 CPU 暂停工作并暂停代码执行，但是允许外设模块继续工作。深度休眠模式下时钟停止工作，暂停代码执行，所有外设（RTCC 和 DSWDT 除外）停止工作。它还将冻结 I/O 状态并且不对 SRAM 和闪存供电。

PWRSAV 指令的汇编语法如例 9-1 所示。

注：	SLEEP_MODE 和 IDLE_MODE 是在所选器件的汇编头文件中定义的常量。

在被允许的中断产生、 WDT 超时或器件复位时，器件会退出休眠和空闲模式。器件退出这两种模式的过程称为“唤醒”。

9.2.1 休眠模式

休眠模式具有以下特征：

- 系统时钟源关闭。如果使用了片上振荡器，也要关闭它。
- 如果没有 I/O 引脚拉电流，则器件电流消耗将降至最低。
- I/O 引脚方向和状态被冻结。
- 因为系统时钟源被禁止，所以故障保护时钟监视器在休眠模式下不工作。
- 如果将LPRC作为时钟源的WDT或RTCC被使能，则 LPRC 时钟将在休眠模式下继续运行。
- 如果 WDT 被使能，则在进入休眠模式之前被自动清零。
- 有些器件功能或外设可能在休眠模式下继续工作。包括 I/O 端口上的输入电平变化通知功能或使用外部时钟输入的外设。任何需要使用系统时钟源来工作的外设在休眠模式下都将被禁止。

当发生以下任何事件时，器件将被从休眠模式唤醒：

- 产生任何被单独允许的中断
- 任何形式的器件复位
- WDT 超时

从休眠模式唤醒时，处理器将使用在进入休眠模式时处于工作状态的时钟源重新开始工作。

例 9-1： PWRSAV 指令语法

```
PWRSAV     #SLEEP_MODE    ; Put the device into SLEEP mode
PWRSAV     #IDLE_MODE     ; Put the device into IDLE mode
BSET       DSCON, #DSEN   ; Enable Deep Sleep
PWRSAV     #SLEEP_MODE    ; Put the device into Deep SLEEP mode
```

9.2.2 空闲模式

空闲模式具有以下特征：

- CPU 将停止执行指令。
- WDT 被自动清零。
- 系统时钟源保持工作状态。默认情况下，所有外设模块将继续使用系统时钟源正常工作，也可以有选择地禁止它们（见**第 9.4 节“选择性外设模块控制”**）。
- 如果WDT或FSCM被使能，则LPRC也将保持工作状态。

当发生以下任何事件时，器件将被从空闲模式唤醒：

- 产生任何被单独允许的中断
- 任何器件复位
- WDT 超时

从空闲模式唤醒时，重新为 CPU 提供时钟，且立即从 PWRSAV 指令之后的下一条指令或 ISR 中的第一条指令开始执行。

9.2.3 在节能指令执行期间的中断

在 PWRSAV 指令执行期间（深度休眠除外）发生的中断都将延迟到进入休眠或空闲模式后才产生，并导致器件从休眠或空闲模式中唤醒。

9.2.4 深度休眠模式

在 PIC24FJ64GA104 系列器件中，深度休眠模式用于实现最低程度的功耗，而无需使用外部开关来完全关闭器件的所有电源。进入深度休眠模式完全由软件控制。从深度休眠模式退出可以通过以下任意事件触发：

- POR 事件
- $\overline{\text{MCLR}}$ 事件
- RTCC 闹钟 （如果有 RTCC）
- 外部中断 0
- 深度休眠看门狗定时器 （DSWDT）超时

在深度休眠模式下，可以让器件实时时钟和日历（RTCC）保持运行，从而不会失去时钟周期。

器件具有专用的深度休眠欠压复位 （DSBOR）和深度休眠看门狗定时器 （DSWDT）复位电路，用于监视电压和超时事件。DSBOR 和 DSWDT 独立于用于其他功耗管理模式（休眠、空闲和打盹）的标准 BOR 和 WDT。

注： 因为深度休眠模式是通过关闭片上 VDDCORE 稳压器来使单片机掉电，所以仅当使能内部稳压器工作时才可使用深度休眠功能。

9.2.4.1 进入深度休眠模式

进入深度休眠模式的方式为：将 DSCON 寄存器中的 DSEN 位置 1，然后在 1 到 3 个指令周期内执行一条休眠指令（PWRSAV #SLEEP_MODE），以最大程度降低意外进入深度休眠的机会。

如果未在 3 个指令周期内执行 PWRSAV 命令，硬件会将 DSEN 位清零，在进入深度休眠模式之前，软件必须重新将该位置 1。退出深度休眠模式时， DSEN 位也将自动清零。

注： 要在从深度休眠唤醒之后重新进入深度休眠模式，则在清零 RELEASE 位之后，必须经过至少 3 个 TCY 的延时。

进入深度休眠模式的序列如下：

1. 如果应用程序需要深度休眠 WDT，则使能它并配置它的时钟源 （详细信息，请参见**第 9.2.4.7 节“深度休眠 WDT”**）。
2. 如果应用程序需要深度休眠 BOR，则通过编程 DSBOREN 配置位 （CW4<6>）来使能它。
3. 如果应用程序需要通过RTCC闹钟从深度休眠唤醒，则使能并配置 RTCC 模块 （更多信息，请参见**第 19.0 节“实时时钟和日历（RTCC）”**）。
4. 如果需要，通过将其写入 DSGPR0 和 DSGPR1 寄存器来保护应用程序中重要的现场数据 （可选）。
5. 通过将 DSEN 位（DSCON<15>）置 1 使能深度休眠模式。
6. 通过立即发出PWRSAV #0指令进入深度休眠模式。

任何时候 DSEN 位置 1 时， DSWAKE 寄存器中的所有位均自动清零。

9.2.4.2 进入深度休眠模式时的特殊情形

进入深度休眠模式时，有一些情况下，在将 DSEN 位置 1 和执行 PWRSAV 指令之间需要有一定的延时。它们通常可以归纳为三种情形：

1. 情形 (1)：使用外部唤醒源 （INT0），或者使用 RTCC
2. 情形 (2)：使用可临时禁止的应用程序级别中断
3. 情形 (3)：使用必须监视的中断

在第一种情形中，应用程序需要在 INT0 引脚置为有效或发生 RTCC 中断时从深度休眠唤醒。在这种情形中，必须插入三条 NOP 指令，以便在器件进入深度休眠模式之后正确同步对于异步 INT0 中断的检测。如果应用程序不使用 INT0 或 RTCC 唤醒，则 NOP 指令是可有可无的。

在第二种情形中，应用程序还使用了可以简单忽略的中断。对于这些应用程序，在执行 NOP 指令期间发生的中断事件可能导致执行 ISR。这意味着，在返回到代码之前，经过的时间将超出三个指令周期， DSEN 位将会被清零。要防止错过深度休眠，可以在进入深度休眠模式之前临时禁止中断。调用 DISI 指令 4 个周期即足以防止中断对于进入深度休眠的干扰。

在第三种情形中，即使在将 DSEN 置 1 和执行 PWRSAV 指令之间的时间间隔中，也不能简单地忽略中断，而必须不断地进行中断检测。对于这些情况，可以禁止中断并测试中断条件，并根据需要跳过 PWRSAV 指令。中断测试可以通过检查 CPUIRQ 位 （INTTREG<15>）的状态来实现；如果有未响应中断等待处理，该位会置 1。如果在执行 PWRSAV 指令之前 CPUIRQ 置 1，则跳过该指令。此时， DISI 指令到期 （距离执行该指令的时间超出 4 个指令周期），应用程序将跳转到相应的 ISR。在应用程序返回时，它可以尝试重新进入深度休眠模式或执行一些其他系统功能。在两种情况下，在 PWRSAV 指令之后，应用程序都必须具有一些功能代码，用于处理 PWRSAV指令被跳过，器件未进入深度休眠模式的情况。

例 9-2 给出了处理这些情况的示例。在这些情况中，建议使用汇编或行内 C 程序，以确保代码在所需的周期数内执行完毕。

例 9-2： 处理进入深度休眠的特殊情形

```
// Case 1:simplest delay scenario
//
asm("bset  DSCON, #15");
asm("nop");
asm("nop");
asm("nop");
asm("pwrsav #0");
//
// Case 2:interrupts disabled
//
asm("disi #4");
asm("bset  DSCON, #15");
asm("nop");
asm("nop");
asm("nop");
asm("pwrsav #0");
//
// Case 3:interrupts disabled with
// interrupt testing
//
asm("disi #4");
asm("bset  DSCON, #15");
asm("nop");
asm("nop");
asm("btss  INTTREG, #15");
asm("pwrsav #0");
// continue with application code here
//
```

9.2.4.3 退出深度休眠模式

深度休眠模式在发生以下任一事件时退出：

- VDD 电源发生 POR 事件。如果没有 DSBOR 电路用于重新激活 VDD 电源 POR 电路，则外部 VDD 电源必须降到 POR 电路的自然激活电压。
- DSWDT超时。当DSWDT定时器发生超时时，器件退出深度休眠模式。
- RTCC 闹钟 （如果 RTCEN = 1）。
- MCLR 引脚置为有效 （0）。
- INT0引脚置为有效（如果在进入深度休眠之前允许了中断）。极性配置用于确定引脚的有效电平 （0 或 1），这将导致从深度休眠模式退出。处于深度休眠模式时，需要在 INT0 引脚上发生电平变化时才会从深度休眠模式退出。

注：	在进入深度休眠模式时的所有待处理中断都会被清除。

退出深度休眠模式相当于器件上电复位 （POR），器件通常不会保持原有状态。这种情况的例外情况包括 RTCC （如果有，它在唤醒后将继续保持工作）、DSGPRx 寄存器和 DSWDT。

从退出深度休眠模式到 POR 序列完成之前的唤醒事件都会被忽略，不会被捕捉到 DSWAKE 寄存器中。

退出深度休眠模式的序列如下：

1. 在发生唤醒事件之后，器件会退出深度休眠并执行 POR。DSEN 位被自动清零。代码将在复位向量处继续执行。
2. 要确定器件先前是否从深度休眠模式退出，可以读取深度休眠位 DPSLP （RCON<10>）。如果上次是从深度休眠模式退出，该位将置 1。如果该位置 1，则将它清零。
3. 通过读取 DSWAKE 寄存器来确定唤醒源。
4. 通过读取 DSBOR 位 （DSCON<1>）确定在深度休眠模式期间是否发生了 DSBOR 事件。
5. 如果应用程序现场数据已保存，则通过DSGPR0 和 DSGPR1 寄存器回读它们。
6. 清零 RELEASE 位 （DSCON<0>）。

9.2.4.4 深度休眠唤醒时间

因为从深度休眠唤醒会产生POR，所以深度休眠唤醒时间与器件 POR 时间相同。此外，因为内部稳压器被关闭，所以 VCAP 上的电压可能下降，具体取决于器件休眠的时间。如果 VCAP 降至低于 2V，则还需要额外的唤醒时间，以便稳压器对 VCAP 进行充电。

深度休眠唤醒时间在**第 28.0 节 “电气特性”**中规定为 TDSWU。该规定代表最坏情况下的唤醒时间，包括全部 POR 复位时间（包括 TPOR 和 TRST），以及用于对 VCAP 上已放电至 0V 的 10 μF 电容进行完全充电的时间。如果 VCAP 未放电，则唤醒会明显快得多。

9.2.4.5 使用 DSGPR0/DSGPR1 寄存器保存现场数据

因为退出深度休眠模式会产生POR，所以大多数特殊功能寄存器将复位为其默认的 POR 值。此外，因为在深度休眠模式下不会对 VDDCORE 供电，所以退出该模式时，数据 RAM 中的信息可能丢失。

需要在进入深度休眠之前保存关键数据的应用程序可以使用深度休眠通用寄存器 DSGPR0 和 DSGPR1，或者数据 EEPROM （如果可用）。不同于其他 SFR，器件处于深度休眠模式时，这些寄存器的内容会被保存。退出深度休眠模式之后，软件可以通过读取寄存器并清零 RELEASE 位 （DSCON<0>）来恢复数据。

9.2.4.6 深度休眠期间的 I/O 引脚

在深度休眠期间，通用I/O引脚将保持它们原先的状态，辅助振荡器 （SOSC）将保持运行 （如果使能）。在进入深度休眠模式之前配置为输入 （TRIS 位置 1）的引脚在深度休眠期间将保持高阻态。在进入深度休眠模式之前配置为输出 （TRIS 位清零）的引脚在深度休眠期间将保持为输出引脚。处于该模式时，它们继续驱动由进入深度休眠模式时其相应的 LAT 位决定的输出电平。

器件被唤醒之后，所有 I/O 引脚将继续维持它们原先的状态，即使器件已经完成 POR 序列，并再次执行应用程序代码。配置为输入的引脚在深度休眠期间将保持高阻态，配置为输出的引脚将继续驱动其原先的值。唤醒之后，TRIS 和 LAT 寄存器，以及 SOSCEN 位（OSCCON<1>）会复位。如果固件修改了其中任意位或寄存器，则 I/O 不会立即切换为最新配置的状态。在固件清零 RELEASE 位（DSCON<0>）之后，I/O 引脚将被“释放”。这会导致 I/O 引脚切换为由它们相应的 TRIS 和 LAT 位值配置的状态。

这意味着，如果要在唤醒之后保持 SOSC 运行，则需要在清零 RELEASE 之前先将 SOSCEN 位置 1。

如果使能了深度休眠 BOR （DSBOR），并且在深度休眠期间发生 DSBOR 或真正的 POR 事件，I/O 引脚将立即被释放，类似于清零 RELEASE 位。所有原先的状态信息都将丢失，包括通用寄存器 DSGPR0 和 DSGPR1 的内容。

如果在深度休眠期间发生 $\overline{\text{MCLR}}$ 复位事件，DSGPRx、DSCON 和 DSWAKE 寄存器将保持有效，RELEASE 位将保持置 1。SOSC 的状态也会保持不变。但是，I/O 引脚将复位为它们的 $\overline{\text{MCLR}}$ 复位状态。因为 RELEASE 仍然置 1，所以 SOSCEN 位 （OSCCON<1>）的变化直到 RELEASE 位清零之后才会生效。

在所有其他深度休眠唤醒情形中，应用固件必须清零 RELEASE 位，以重新配置 I/O 引脚。

9.2.4.7 深度休眠 WDT

要在深度休眠模式下使能 DSWDT，应编程配置位 DSWDTEN （CW4<7>）。要使 DSWDT 正常工作，需要使能器件看门狗定时器（WDT）。进入深度休眠模式时，会自动复位 DSWDT。

DSWDT 时钟源通过 DSWDTOSC 配置位（CW4<4>）进行选择。后分频比选项可以通过DSWDTPS<3:0>配置位 （CW4<3:0>）进行设定。可实现的最小超时周期为 2.1 ms，最大超时周期为 25.7 天。关于 CW4 配置寄存器和 DSWDT 配置选项的更多详细信息，请参见**第 25.0 节 “特殊功能”**。

9.2.4.8 在深度休眠模式下切换时钟

RTCC 和 DSWDT 都可以使用 SOSC 或 LPRC 时钟源。这使 RTCC 和 DSWDT 可以在无需同时使能 LPRC 和 SOSC 的情况下运行，从而降低功耗。

RTCC 使用 LPRC 时钟源运行时，RTCC 的精度会降低大约 5-10%。如果要求 RTCC 保持高精度，则它必须使用 SOSC 时钟源。 RTCC 时钟源通过 RTCOSC 配置位（CW4<5>）进行选择。

在某些情况下，当进入深度休眠模式时， DSWDT 的时钟源可能已关闭。在这种情况下，会自动开启时钟源（如果 DSWDT 已使能），而无需软件干预。但是，这会使 DSWDT 计数器的启动产生延时。在使用 SOSC 作为时钟源时，为了避免这种延时，应用程序可在进入深度休眠模式之前激活 SOSC。

9.2.4.9 检查和清除深度休眠状态

在进入深度休眠模式时，状态位 DPSLP（RCON<10>）会置 1，必须用软件清零。

在上电时，软件应读取该状态位，以确定是否是由于从深度休眠模式退出而发生复位，如果该位置 1，则清零该位。在 DPSLP 和 POR 位状态的 4 种可能组合中，可以考虑三种情形：

- DPSLP和POR位均清零。在这种情况下，复位是由于退出深度休眠模式之外的某个其他事件而发生的。
- DPSLP位清零，但POR位置1。这是正常的上电复位。
- DPSLP 和 POR 位均置 1。这表示发生了以下情况：进入深度休眠模式、器件发生掉电，然后从深度休眠模式退出。

9.2.4.10 上电复位 （POR）

器件通过监视 VDD 电压来产生 POR。因为退出深度休眠在功能上类似于 POR，所以应使用**第 9.2.4.9 节“检查和清除深度休眠状态”**中所述的技术来区分深度休眠和真正的 POR 事件。

发生真正的 POR 时，包括所有深度休眠逻辑（深度休眠寄存器、 RTCC 和 DSWDT 等）在内的整个器件将复位。

9.2.4.11 深度休眠序列汇总

为了进行回顾，以下列出了调用和退出深度休眠模式所涉及的必需步骤：

1. 器件退出复位并开始执行它的应用程序代码。
2. 如果需要 DSWDT 功能，则编程相应的配置位。
3. 为 DSWDT 和 RTCC 选择相应的时钟（可选）。
4. 使能并配置 RTCC （可选）。
5. 将现场数据写入 DSGPRx 寄存器 （可选）。
6. 允许 INT0 中断 （可选）。
7. 将 DSCON 寄存器中的 DSEN 位置 1。
8. 通过发出 `PWRSV #SLEEP_MODE` 命令进入深度休眠模式。
9. 器件在发生唤醒事件时退出深度休眠。
10. DSEN 位被自动清零。
11. 读取并清零 RCON 中的 DPSLP 状态位，以及 DSWAKE 状态位。
12. 读取 DSGPRx 寄存器 （可选）。
13. 完成所有与状态相关的配置之后，清零 RELEASE 位。
14. 应用程序恢复正常运行。

寄存器 9-1：　　DSCON：深度休眠控制寄存器

R/W-0, HC	U-0	U-0	U-0	U-0	U-0	U-0	U-0
DSEN(1)	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0, HCS	R/C-0, HS
—	—	—	—	—	—	DSBOR(1,2,3)	RELEASE(1,2)
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	C = 可清零位	U = 未实现，读为 0
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知
HC = 硬件清零位	HS = 硬件置 1 位	HCS = 硬件清零 / 置 1 位	

bit 15　**DSEN**：深度休眠使能位 (1)
1 = 在下一条指令执行 PWRSAV #0 时器件进入深度休眠模式
0 = 在执行 PWRSAV #0 时器件进入正常休眠模式

bit 14-2　**未实现**：读为 0

bit 1　**DSBOR**：深度休眠 BOR 事件状态位 (1,2,3)
1 = DSBOR 先前在工作，并且在深度休眠期间检测到 BOR 事件
0 = DSBOR 先前被禁止，或者先前在工作，但在深度休眠期间未检测到 BOR 事件

bit 0　**RELEASE**：I/O 引脚状态深度休眠释放位 (1,2)
1 = 退出深度休眠之后， I/O 引脚和 SOSC 维持它们的状态，无论它们的 LAT 和 TRIS 配置如何
0 = I/O 引脚和 SOSC 从它们的深度休眠状态释放。引脚状态受 LAT 和 TRIS 的配置以及 SOSCEN 位控制。

注　1： 只有在深度休眠模式之外发生 POR 事件时，这些位才会复位。
2： 只有对于初始上电时发生的 POR，复位值才为 0；对于深度休眠 POR，复位值为 1。
3： 它只是一个状态位； DSBOR 事件不会导致从深度休眠唤醒。

寄存器 9-2：　　DSWAKE：深度休眠唤醒源寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0, HS
—	—	—	—	—	—	—	DSINT0(1)
bit 15							bit 8

R/W-0, HS	U-0	U-0	R/W-0, HS	R/W-0, HS	R/W-0, HS	U-0	R/W-0, HS
DSFLT(1)	—	—	DSWDT(1)	DSRTC(1)	DSMCLR(1)	—	DSPOR(2)
bit 7							bit 0

图注：	HS = 可由硬件置 1 的位		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15-9　**未实现：**读为 0

bit 8　**DSINT0：**电平变化中断位 (1)
1 = 外部中断 0 在深度休眠期间被置为有效
0 = 外部中断 0 在深度休眠期间未被置为有效

bit 7　**DSFLT：**深度休眠故障检测位 (1)
1 = 在深度休眠期间发生了故障，并且某些深度休眠配置设置可能已被破坏
0 = 在深度休眠期间未检测到故障

bit 6-5　**未实现：**读为 0

bit 4　**DSWDT：**深度休眠看门狗定时器超时位 (1)
1 = 深度休眠看门狗定时器在深度休眠期间超时
0 = 深度休眠看门狗定时器在深度休眠期间未超时

bit 3　**DSRTC：**实时时钟和日历闹钟位 (1)
1 = 实时时钟和日历在深度休眠期间触发了一次闹钟
0 = 实时时钟和日历在深度休眠期间未触发闹钟

bit 2　**DSMCLR：**深度休眠 $\overline{\text{MCLR}}$ 事件位 (1)
1 = $\overline{\text{MCLR}}$ 引脚在深度休眠模式期间被置为有效
0 = $\overline{\text{MCLR}}$ 引脚在深度休眠模式期间未被置为有效

bit 1　**未实现：**读为 0

bit 0　**DSPOR：**上电复位事件位 (2)
1 = VDD 电源 POR 电路先前在工作，并且检测到 POR 事件
0 = VDD 电源 POR 电路先前不工作，或者先前在工作，但未检测到 POR 事件

注　1：只有器件处于深度休眠模式时，该位才能置 1。
2：该位可以在深度休眠模式之外置 1。

9.3 打盹模式

通常，更改时钟速度和使用某种节能模式是降低功耗的首选策略。但在有些情况下可能不可行。例如，某些应用可能必须保持不间断的同步通信，即便在它不执行任何其他操作时也不例外。降低系统时钟速度可能会带来通信错误，而使用节能模式可能会完全停止通信。

打盹模式是另一种简单有效的节能方法，它可以在器件仍然执行代码的情况下降低功耗。在该模式下，系统时钟以相同的时钟源和相同的速度继续工作。外设模块时钟速度保持不变，但 CPU 时钟速度降低了。保持这两个时钟域同步，可以保持外设访问 SFR 的能力，同时 CPU 以较慢的速度执行代码。

通过将 DOZEN 位（CLKDIV<11>）置 1 使能打盹模式。外设与内核的时钟速度之比是由 DOZE<2:0> 位（CLKDIV<14:12>）决定的。有八种可能的配置，从 1:1 到 1:128，其中 1:1 是默认设置。

在事件驱动的应用中，使用打盹模式有选择地降低功耗是可行的。这样就可以实现不间断地运行对时序敏感的功能（如同步通信），而 CPU 保持空闲等待事件调用中断服务程序。通过将 ROI 位（CLKDIV<15>）置 1，可以使器件在产生中断时自动返回到全速 CPU 工作模式。默认情况下，中断事件对打盹模式工作没有影响。

9.4 选择性外设模块控制

空闲和打盹模式使用户可通过降低或停止 CPU 时钟显著降低功耗。即便如此，外设模块仍然保持有时钟提供，因此会有功耗。有些应用可能存在这些模式无法提供的需求：即将功率资源分配给 CPU 处理，而使外设功耗最低。

PIC24F 器件通过允许有选择地禁止外设模块以降低或消除其功耗来满足这种需求。这可以通过两个控制位来实现：

- 外设使能位，一般命名为“XXXEN”，位于模块的主控制 SFR 中。
- 外设模块禁止（PMD）位，一般命名为“XXXMD”，位于相应的 PMD 控制寄存器中。

这两个位在使能或禁止其相关模块方面具有相似的功能。将模块的 PMD 位置 1 会禁止该模块的所有时钟源，从而将其功耗降至绝对最低。在该状态下，与外设相关的控制寄存器和状态寄存器也被禁止，因此写入那些寄存器不会有影响，且读取值无效。许多外设模块都有一个对应的 PMD 位。

相反，通过清零模块的 XXXEN 位以禁止其功能来禁止该模块，会保留其寄存器的读写功能。这也会降低功耗，但不如将 PMD 位置 1 降低得多。大多数外设模块都有一个使能位；输入捕捉、输出比较和 RTCC 则例外。

要实现更多可选择的节能，可在器件进入空闲模式后有选择地禁止外设模块。这可通过通用名称格式为“XXXIDL”的控制位实现。默认情况下，所有能在空闲模式下工作的模块都适用。使用空闲模式禁止功能使功耗在空闲模式下得到进一步降低，从而增强了对功耗要求异常严格的应用的节能功能。

注：

10.0 I/O 端口

注：	本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 12 章“带外设引脚选择（PPS）的I/O端口”**（DS39711A_CN）。

所有器件引脚 （VDD、 VSS、 $\overline{\text{MCLR}}$ 和 OSCI/CLKI 除外）均由外设和并行 I/O 端口所共用。所有 I/O 输入端口都为施密特触发器输入，提高了抗噪声能力。

10.1 并行 I/O （PIO）端口

与某个外设共用一个引脚的并行I/O端口通常服从于该外设。外设的输出缓冲器数据和控制信号提供给一对多路开关。这对多路开关用于选择I/O引脚的输出数据和控制信号是来自外设还是相应的端口。该逻辑电路同时会阻止“环回进入”（loop through），即一个端口的数字输出可以驱动共用同一个引脚的外设的输入。图 10-1 显示了端口是如何与其他外设复用的以及相关的 I/O 引脚。

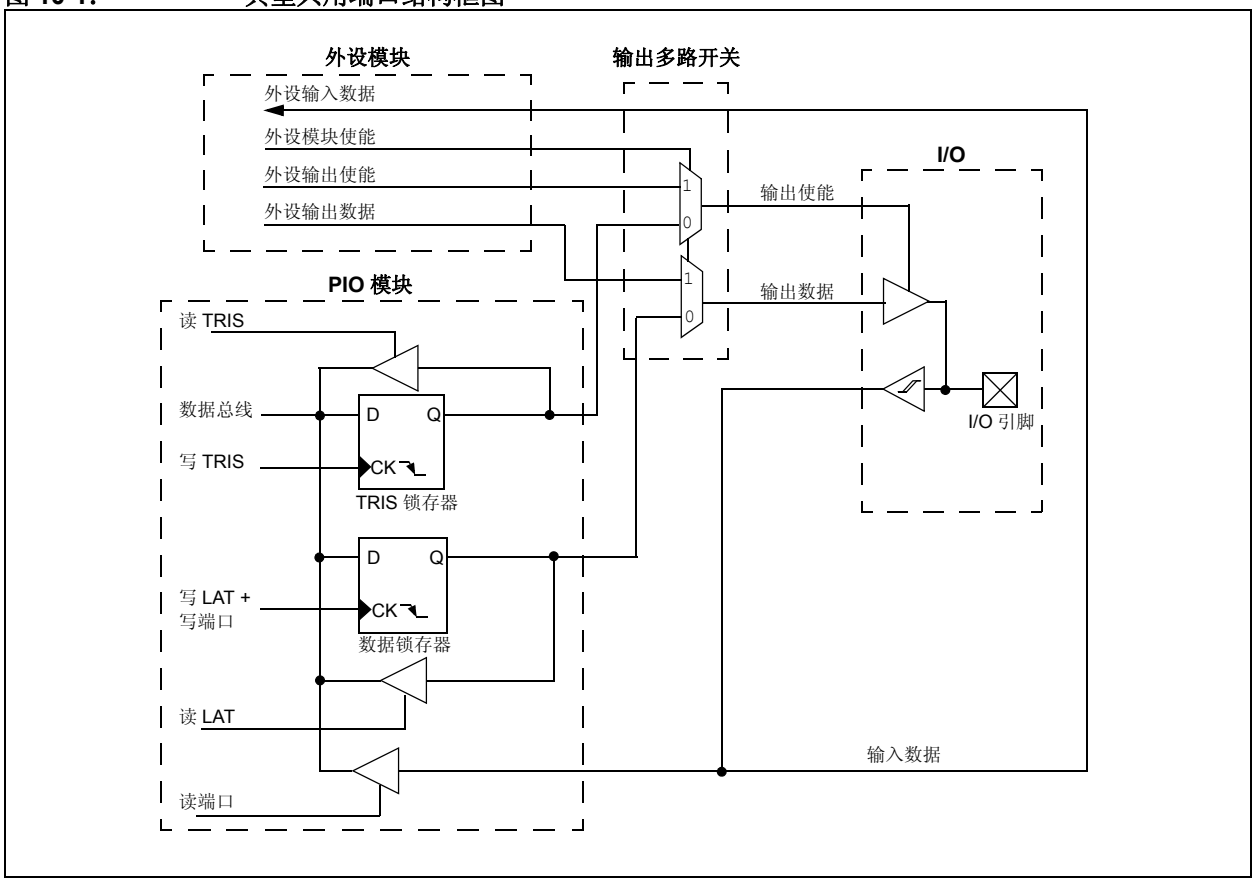
如果外设使能，并且外设正在驱动相关引脚时，将禁止该引脚用作通用输出引脚。可以读该 I/O 引脚，但并行端口位的输出驱动器将被禁止。如果使能某外设但该外设没有驱动相应的引脚，则该引脚可由一个端口驱动。

所有端口引脚都有 3 个寄存器与其作为数字 I/O 时的操作直接相关。数据方向寄存器 （TRIS）决定引脚是输入还是输出。如果数据方向位为 1，则引脚为输入。复位后，所有端口引脚均定义为输入。读输出锁存寄存器（LAT）时，读到的是锁存器中的值；写输出锁存寄存器时，写入的是锁存器。读端口 （PORT）时，读到的是端口引脚的值；而写端口引脚时，写入的是锁存器。

对于特定器件无效的任何位及其相关的数据和控制寄存器都将被禁止。这意味着对应的 LAT 和 TRIS 寄存器以及端口引脚都将读为零。

当端口引脚与另一个外设共用或与定义为仅输入的功能共用时，它将被视为专用端口，因为没有任何其他竞争的输出源。

图 10-1： 典型共用端口结构框图



10.1.1 漏极开路配置

除 PORT、LAT 和 TRIS 寄存器用于数据控制外，每个端口引脚也可被单独地配置为数字输出或漏极开路输出。这是由与每个端口相关的漏极开路控制寄存器 ODCx 控制的。将其中的任何位置 1 即可将相应的引脚配置为漏极开路输出。

这种漏极开路特性允许通过使用外部上拉电阻，在所需的任意仅用作数字功能的引脚上产生高于 VDD（如 5V）的输出电平。允许的最大漏极开路电压与最大 VIH 规范相同。

10.2 配置模拟端口引脚

AD1PCFGL 和 TRIS 寄存器用于控制 A/D 端口引脚的操作。如果要将端口引脚设置为模拟输入，则对应的 TRIS 位也必须置 1。如果将 TRIS 位清零（输出），则数字输出电平（VOH 或 VOL）将被转换。

当读取端口寄存器时，所有配置为模拟输入通道的引脚均读为零（低电平）。

配置为数字输入的引脚将不会对模拟输入进行转换。对于任何定义为数字输入的引脚（包括 ANx 引脚），加在引脚上的模拟电压可能导致输入缓冲器消耗的电流超出器件规范。

10.2.1 I/O 端口写 / 读时序

端口方向改变或端口写操作与同一端口的读操作之间需要一个指令周期。这通常通过一条 NOP 指令来实现（例 10-1）。

10.2.2 模拟输入引脚和电压注意事项

用作器件输入的引脚的耐压能力取决于引脚的输入功能。仅用作数字输入的引脚能够承受最高 5.5V 的直流电压，这个电压值是数字逻辑电路的典型耐压值。而具有模拟输入功能的引脚只能承受最高为 VDD 的电压值。应避免在这些引脚上施加超过 VDD 的电压。

表 10-1 汇总了输入电压能力。更多详细信息，请参见**第 28.0 节 “电气特性”**。

表 10-1： 输入耐压

端口或引脚	可承受的最高输入电压	说明
PORTA<4:0>	VDD	只能承受最高为VDD的输入电压。
PORTB<15:12>		
PORTB<4:0>		
PORTC<3:0>(1)		
PORTA<10:7>(1)	5.5V	可承受高于VDD的输入电压，可用于大部分标准逻辑电路。
PORTB<11:7>		
PORTB<6:5>		
PORTC<9:4>(1)		

注 1： 在 28 引脚器件上不可用。

例 10-1： 端口写 / 读示例

```
MOV   0xFF00, W0          ; Configure PORTB<15:8> as inputs
MOV   W0, TRISB           ; and PORTB<7:0> as outputs
NOP                       ; Delay 1 cycle
BTSS  PORTB, #13          ; Next Instruction
```

10.3 输入电平变化通知

I/O端口的输入电平变化通知功能允许PIC24FJ64GA104系列器件在选定输入引脚的状态变化（Change-Of-State，COF）时，向处理器发出中断请求。即使在时钟被禁止的休眠模式下，该特性也可检测输入状态改变。根据器件的引脚数，最多可以选择（允许）29 个外部输入在状态发生变化时产生中断请求。

CNEN1 和 CNEN2 寄存器包含每个 CN 输入引脚的中断允许控制位。将其中任一位置 1 将允许相应引脚的 CN 中断。

每个 CN 引脚都有一个与之相连的弱上拉电路。弱上拉电路为与之相连的引脚提供电流源。当连接了按钮或键盘设备时，不再需要使用外部电阻。使用 CNPU1 和 CNPU2 寄存器（用于上拉）可分别使能上拉电路。每个 CN 引脚都有独立的上拉控制位。将控制位置 1 可使能相应引脚的弱上拉功能。

当选择内部上拉电路时，引脚上拉为 V_{DD} – 0.7V（典型值）。请确保在使能内部上拉电路时没有任何外部上拉源，因为电压差会引起电流回路。

注：	只要端口引脚被配置为数字输出，电平变化通知引脚的上拉电路应始终被禁止。

10.4 外设引脚选择（PPS）

通用器件的主要挑战是提供尽可能多的外设功能部件，同时将其与 I/O 引脚功能的冲突降到最小。在需要多个外设复用一个引脚的应用中，要在应用程序代码中进行变通比较困难，换句话说彻底重新设计可能是唯一的选择。

外设引脚选择功能提供了这些选择的替代方法，使得用户可以在较宽的 I/O 引脚范围内选择和配置外设功能部件。通过增加特定器件上可用的引脚排列选项，用户可以让单片机更好地适合他们的整个应用，而不是通过修改应用来适应器件。

外设引脚选择功能对固定的一部分数字 I/O 引脚进行操作。用户可以将许多数字外设的输入和 / 或输出独立地映射到这些 I/O 引脚中的任何一个。外设引脚选择通过软件来执行，通常不需要对器件进行再编程。一旦建立外设引脚选择，就同时包含了硬件保护，以防止对外设映射的意外或错误更改。

10.4.1 可用的引脚

外设引脚选择功能可在最多25个引脚的范围内使用，取决于特定器件及其引脚数。支持外设引脚选择功能的引脚在它们的引脚全称中包含名称“RPn”，其中“n”是可重映射的引脚编号。

请参见表 1-2 了解每种封装中引脚排列选项的汇总。

10.4.2 可用的外设

外设引脚选择管理的外设都是仅数字功能的外设。这些外设包括一般串行通信（UART 和 SPI）、通用定时器时钟输入、与定时器相关的外设（输入捕捉和输出比较）以及外部中断输入。还包括比较器模块的输出，因为这些是离散的数字信号。

外设引脚选择不适用于I^2C™电平变化通知输入、RTCC 闹钟输出或带模拟输入的外设。

有引脚选择和无引脚选择外设之间的主要差异在于引脚选择外设与默认的 I/O 引脚无关。必须始终在使用外设前将其分配给特定的 I/O 引脚。相反，无引脚选择外设始终在默认引脚上可用，假设该外设工作且与其他外设没有冲突。

10.4.2.1 外设引脚选择功能优先级

引脚可选择的外设输出（例如，OC 和 UART 发送）的优先级高于该引脚的任何通用数字功能（例如，PMP 和端口 I/O）。同一引脚上的专用数字输出（例如 USB 功能）的优先级高于 PPS 输出。本数据手册开头提供的引脚图按优先级顺序列出了外设输出。在确定特定引脚的优先级时，可参见这些图。

与具有固定外设的器件不同，引脚可选择的外设输入绝不会全权拥有引脚。引脚的输出缓冲器由引脚的 TRIS 位设置或固定外设控制。如果将引脚配置为数字模式，则 PPS 输入将正确工作（读取输入）。如果使能同一引脚的模拟功能，则将禁止引脚可选择的输入。

10.4.3 控制外设引脚选择

外设引脚选择功能由两组特殊功能寄存器控制：一组映射外设输入，另一组映射外设输出。因为它们是分别控制的，所以可以不受限制地将特定外设的输入和输出（如果外设同时具有输入和输出）配置在任何可选择的功能引脚上。

外设与外设可选择引脚之间的关系用两种不同的方式进行处理，取决于被映射的是输入还是输出。

10.4.3.1 输入映射

外设引脚选择选项的输入在外设基础上进行映射；即，与外设相关的控制寄存器指示要被映射的引脚。RPINRx 寄存器用于配置外设输入映射（见寄存器 10-1 至寄存器 10-14）。每个寄存器包含最多两组 5 位位域，每组都与引脚可选择外设之一相关。用适当的 6 位值编程给定外设的位域，会将具有对应值的 RPn 引脚映射到该外设。对于任何给定的器件，任何位域的值的有效范围与器件所支持的外设引脚选择选项的最大数目相对应。

表 10-2： 可选择的输入源 （将输入映射到功能）(1)

输入名称	功能名称	寄存器	功能映射位
外部中断 1	INT1	RPINR0	INT1R<5:0>
外部中断 2	INT2	RPINR1	INT2R<5:0>
输入捕捉 1	IC1	RPINR7	IC1R<5:0>
输入捕捉 2	IC2	RPINR7	IC2R<5:0>
输入捕捉 3	IC3	RPINR8	IC3R<5:0>
输入捕捉 4	IC4	RPINR8	IC4R<5:0>
输入捕捉 5	IC5	RPINR9	IC5R<5:0>
输出比较故障 A	OCFA	RPINR11	OCFAR<5:0>
输出比较故障 B	OCFB	RPINR11	OCFBR<5:0>
SPI1 时钟输入	SCK1IN	RPINR20	SCK1R<5:0>
SPI1 数据输入	SDI1	RPINR20	SDI1R<5:0>
SPI1 从选择输入	SS1IN	RPINR21	SS1R<5:0>
SPI2 时钟输入	SCK2IN	RPINR22	SCK2R<5:0>
SPI2 数据输入	SDI2	RPINR22	SDI2R<5:0>
SPI2 从选择输入	SS2IN	RPINR23	SS2R<5:0>
Timer2 外部时钟	T2CK	RPINR3	T2CKR<5:0>
Timer3 外部时钟	T3CK	RPINR3	T3CKR<5:0>
Timer4 外部时钟	T4CK	RPINR4	T4CKR<5:0>
Timer5 外部时钟	T5CK	RPINR4	T5CKR<5:0>
UART1 允许发送	$\overline{\text{U1CTS}}$	RPINR18	U1CTSR<5:0>
UART1 接收	U1RX	RPINR18	U1RXR<5:0>
UART2 允许发送	$\overline{\text{U2CTS}}$	RPINR19	U2CTSR<5:0>
UART2 接收	U2RX	RPINR19	U2RXR<5:0>

注 1： 除非另外声明，否则所有输入都使用施密特触发器输入缓冲器。

10.4.3.2 输出映射

与输入相比，外设引脚选择选项的输出在引脚基础上进行映射。在这种情况下，与特定引脚相关的控制寄存器指示要被映射的外设输出。 RPORx 寄存器用于控制输出映射。每个寄存器包含最多两组 5 位位域，每组都与一个 RPn 引脚相关（见寄存器 10-15 至寄存器 10-27）。

位域的值与外设之一相对应，并且该外设的输出被映射到引脚 （见表 10-3）。

输出映射的外设列表也包含 000000 的空值，这是由于映射技术造成的。这允许任何给定的引脚保持与任何引脚可选择外设的输出之间的断开状态。

表 10-3： 可选择的输出源 （将功能映射到输出）

输出功能编号 (1)	功能	输出名称
0	NULL(2)	空
1	C1OUT	比较器 1 的输出
2	C2OUT	比较器 2 的输出
3	U1TX	UART1 发送
4	$\overline{\text{U1RTS}}$(3)	UART1 请求发送
5	U2TX	UART2 发送
6	$\overline{\text{U2RTS}}$(3)	UART2 请求发送
7	SDO1	SPI1 数据输出
8	SCK1OUT	SPI1 时钟输出
9	SS1OUT	SPI1 从选择输出
10	SDO2	SPI2 数据输出
11	SCK2OUT	SPI2 时钟输出
12	SS2OUT	SPI2 从选择输出
18	OC1	输出比较 1
19	OC2	输出比较 2
20	OC3	输出比较 3
21	OC4	输出比较 4
22	OC5	输出比较 5
23-28	（未用）	NC
29	CTPLS	CTMU 输出脉冲
30	C3OUT	比较器 3 的输出
31	（未用）	NC

注 1： 用所列的值设置 RPORx 寄存器会将输出功能分配给相关的 RPn 引脚。

2： 在器件复位时将 NULL 功能分配给所有 RPn 输出，并禁止 RPn 输出功能。

3： IrDA® BCLK 功能使用该输出。

10.4.3.3 映射限制

外设引脚选择的控制机制相当灵活。除了防止两个物理引脚配置为同一功能输入或两个功能输出配置为同一引脚导致的信号争用的系统模块，没有硬件强制的锁定。这种灵活性达到以下程度：允许一个输入驱动多个外设，或一个功能输出驱动多个输出引脚。

10.4.3.4 PIC24FJ64GA1 系列器件的 PPS 映射例外

虽然 PPS 寄存器允许最多 32 个可重映射的引脚，但在 44 引脚器件上只实现了最多 26 个引脚（RP0 至 RP25）。在 28 引脚器件上， RP15 以上的所有可重映射引脚均未实现。

10.4.4 控制配置更改

由于可以在运行时更改外设的重映射，因此需要对外设重映射设置一些限制条件以防意外更改配置。 PIC24F 器件具有 3 个功能可防止对外设映射的更改：

- 控制寄存器锁定序列
- 连续状态监视
- 配置位重映射锁定

10.4.4.1 控制寄存器锁定

在正常工作时，不允许写入 RPINRx 和 RPORx 寄存器。尝试写入操作看似正常执行，但实际上寄存器的内容保持不变。要更改这些寄存器，必须用硬件进行解锁。寄存器锁定由 IOLOCK 位（OSCCON<6>）控制。将 IOLOCK 置 1 可防止对控制寄存器的写操作；将 IOLOCK 清零则允许写操作。

要置 1 或清零 IOLOCK，必须执行特定的命令序列：

1. 将 46h 写入 OSCCON<7:0>。
2. 将 57h 写入 OSCCON<7:0>。
3. 通过一次操作将 IOLOCK 清零 （或置 1）。

与振荡器的 LOCK 位的类似序列不同，IOLOCK 会保持一种状态直到被更改。这允许对所有的外设引脚选择这样进行配置：在对所有控制寄存器的更新后紧跟一个解锁序列，然后用第二个锁定序列进行锁定。

10.4.4.2 连续状态监视

除了防止直接写操作， RPINRx 和 RPORx 寄存器的内容一直由影子寄存器通过硬件进行监视。如果任何寄存器发生了意外更改 （例如 ESD 或其他外部事件引起的干扰），将会触发配置不匹配复位。

10.4.4.3 配置位引脚选择锁定

为了进一步确保安全，可以将器件配置为防止对 RPINRx 和 RPORx 寄存器进行多于一次写会话。IOL1WAY（CW2<4>）配置位会阻止 IOLOCK 位在置 1 后被清零。如果 IOLOCK 保持置 1，则不会执行寄存器解锁过程，且不能写入外设引脚选择控制寄存器。清零该位并重新使能外设映射的唯一方法是执行器件复位。

在默认 （未编程）状态下， IOL1WAY 被置 1，将用户限制为只能进行一次写会话。对 IOL1WAY 编程可允许用户（通过对解锁序列的正确使用）对外设引脚选择寄存器不受限制的访问。

10.4.5 外设引脚选择的注意事项

在应用设计中使用控制外设引脚选择功能有一些可能被忽略的注意事项。对于几个只能作为可重映射外设的常见外设尤其如此。

主要的注意事项是在器件的默认（复位）状态下，外设引脚选择在默认引脚上不可用。由于所有 RPINRx 寄存器复位为 11111，所有 RPORx 寄存器复位为 00000，因此所有外设引脚选择输入连接到 Vss，而所有外设引脚选择输出处于断开状态。

注： 在将外设引脚输出连接到 RP31 时，器件上不必具有 RP31，因为寄存器复位就会实现这一设置。

这种情况要求用户在执行任何其他应用程序代码前，必须用适当的外设配置初始化器件。由于 IOLOCK 位在解锁状态下复位，因此在器件复位结束后不必执行解锁序列。然而，基于应用安全考虑，在写入控制寄存器后最好将 IOLOCK 置 1 并锁定配置。

由于解锁序列对时序有严格要求，它必须作为汇编语言程序以与更改振荡器配置相同的方式执行。如果应用程序是用 C 语言或其他高级语言编写的，则解锁序列应通过写行内汇编代码来执行。

选择配置需要查看所有外设引脚选择及其引脚分配，尤其是那些不会在应用中使用的外设。在所有情况下，必须完全禁止未用的引脚可选择外设。未用的外设应将它们的输入分配给未用的 RPn 引脚功能。带有未用 RPn 功能的 I/O 引脚应被配置为空外设输出。

外设到特定引脚的分配不会自动执行引脚的I/O电路的任何其他配置。理论上，这意味着将引脚可选择输出加到引脚，当驱动输出时，引脚可能会意外驱动现有的外设输入。用户必须熟悉共用同一个可重映射引脚的其他固定外设的行为，了解何时使能或禁止它们。为安全起见，共用同一个引脚的固定数字外设在不使用时应被禁止。

根据这些概念，配置特定外设的可重映射引脚不会自动开启该外设功能。必须将外设特别配置为工作并使能，好像是连接到固定引脚一样。这部分在应用程序代码中的位置（紧跟器件复位和外设配置，或在主应用程序内）取决于外设及其在应用中的使用。

最后的注意事项是，外设引脚选择功能既不会改写模拟输入，也不会将带模拟功能的引脚重新配置为数字 I/O。如果器件复位时引脚被配置为模拟输入，则使用外设引脚选择时必须明确将其重新配置为数字 I/O。

例 10-2 给出了使用 UART1 进行带流控制的双向通信的配置。使用了以下输入和输出功能：

- 输入功能：U1RX 和 $\overline{\text{U1CTS}}$
- 输出功能：U1TX 和 $\overline{\text{U1RTS}}$

例 10-2： 配置 UART1 输入和输出功能

```
// Unlock Registers
asm volatile    ("MOV   #OSCCON, w1\n"
                 "MOV    #0x46, w2\n"
                 "MOV    #0x57, w3\n"
                 "MOV.b  w2, [w1]\n"
                 "MOV.b  w3, [w1]\n"
                 "BCLR   OSCCON,#6");

// Configure Input Functions (Table 9-1))
    // Assign U1RX To Pin RP0
    RPINR18bits.U1RXR = 0;

    // Assign U1CTS To Pin RP1
    RPINR18bits.U1CTSR = 1;

// Configure Output Functions (Table 9-2)
    // Assign U1TX To Pin RP2
    RPOR1bits.RP2R = 3;

    // Assign U1RTS To Pin RP3
    RPOR1bits.RP3R = 4;

// Lock Registers
asm volatile    ("MOV   #OSCCON, w1\n"
                    "MOV    #0x46, w2\n"
                    "MOV    #0x57, w3\n"
                    "MOV.b  w2, [w1]\n"
                    "MOV.b  w3, [w1]\n"
                    "BSET   OSCCON, #6");
```

10.4.6 外设引脚选择寄存器

PIC24FJ64GA104 系列器件共实现了 27 个寄存器用于可重映射的外设配置：

- 14 个输入可重映射外设寄存器
- 13 个输出可重映射外设寄存器

注： 仅在 IOLOCK（OSCCON<6>）= 0 时才能改变输入和输出寄存器的值。具体的命令序列请参见**第10.4.4.1节“控制寄存器锁定”**。

寄存器 10-1： RPINR0：外设引脚选择输入寄存器 0

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	INT1R4	INT1R3	INT1R2	INT1R1	INT1R0
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **INT1R<4:0>：**将外部中断 1 （INT1）分配给对应 RPn 或 RPIn 引脚的位

bit 7-0 **未实现：**读为 0

寄存器 10-2： RPINR1：外设引脚选择输入寄存器 1

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	INT2R4	INT2R3	INT2R2	INT2R1	INT2R0
bit 7							bit 0

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

bit 15-5 **未实现：**读为 0

bit 4-0 **INT2R<4:0>：**将外部中断 2 （INT2）分配给对应 RPn 或 RPIn 引脚的位

寄存器 10-3： RPINR3：外设引脚选择输入寄存器 3

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	T3CKR4	T3CKR3	T3CKR2	T3CKR1	T3CKR0
bit 15							bit 8

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	T2CKR4	T2CKR3	T2CKR2	T2CKR1	T2CKR0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **T3CKR<4:0>：**将 Timer3 外部时钟 （T3CK）分配给对应 RPn 或 RPIn 引脚的位

bit 7-5 **未实现：**读为 0

bit 4-0 **T2CKR<4:0>：**将 Timer2 外部时钟 （T2CK）分配给对应 RPn 或 RPIn 引脚的位

寄存器 10-4： RPINR4：外设引脚选择输入寄存器 4

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	T5CKR4	T5CKR3	T5CKR2	T5CKR1	T5CKR0
bit 15							bit 8

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	T4CKR4	T4CKR3	T4CKR2	T4CKR1	T4CKR0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **T5CKR<4:0>：**将 Timer5 外部时钟 （T5CK）分配给对应 RPn 或 RPIn 引脚的位

bit 7-5 **未实现：**读为 0

bit 4-0 **T4CKR<4:0>：**将 Timer4 外部时钟 （T4CK）分配给对应 RPn 或 RPIn 引脚的位

寄存器 10-5：　RPINR7：外设引脚选择输入寄存器 7

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	IC2R4	IC2R3	IC2R2	IC2R1	IC2R0
bit 15							bit 8

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	IC1R4	IC1R3	IC1R2	IC1R1	IC1R0
bit 7							bit 0

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

bit 15-13　**未实现：**读为 0

bit 12-8　**IC2R<4:0>：**将输入捕捉 2 （IC2）分配给对应 RPn 或 RPIn 引脚的位

bit 7-5　**未实现：**读为 0

bit 4-0　**IC1R<4:0>：**将输入捕捉 1 （IC1）分配给对应 RPn 或 RPIn 引脚的位

寄存器 10-6：　RPINR8：外设引脚选择输入寄存器 8

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	IC4R4	IC4R3	IC4R2	IC4R1	IC4R0
bit 15							bit 8

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	IC3R4	IC3R3	IC3R2	IC3R1	IC3R0
bit 7							bit 0

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

bit 15-13　**未实现：**读为 0

bit 12-8　**IC4R<4:0>：**将输入捕捉 4 （IC4）分配给对应 RPn 或 RPIn 引脚的位

bit 7-5　**未实现：**读为 0

bit 4-0　**IC3R<4:0>：**将输入捕捉 3 （IC3）分配给对应 RPn 或 RPIn 引脚的位

寄存器 10-7： RPINR9：外设引脚选择输入寄存器 9

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	IC5R4	IC5R3	IC5R2	IC5R1	IC5R0
bit 7							bit 0

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

bit 15-5 **未实现：**读为 0

bit 4-0 **IC5R<4:0>：**将输入捕捉 5 （IC5）分配给对应 RPn 或 RPIn 引脚的位

寄存器 10-8： RPINR11：外设引脚选择输入寄存器 11

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	OCFBR4	OCFBR3	OCFBR2	OCFBR1	OCFBR0
bit 15							bit 8

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	OCFAR4	OCFAR3	OCFAR2	OCFAR1	OCFAR0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **OCFBR<4:0>：**将输出比较故障 B （OCFB）分配给对应 RPn 或 RPIn 引脚的位

bit 7-5 **未实现：**读为 0

bit 4-0 **OCFAR<4:0>：**将输出比较故障 A （OCFA）分配给对应 RPn 或 RPIn 引脚的位

寄存器 10-9： RPINR18：外设引脚选择输入寄存器 18

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	U1CTSR4	U1CTSR3	U1CTSR2	U1CTSR1	U1CTSR0
bit 15							bit 8

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	U1RXR4	U1RXR3	U1RXR2	U1RXR1	U1RXR0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **U1CTSR<4:0>：**将 UART1 允许发送 （$\overline{\text{U1CTS}}$）分配给对应 RPn 或 RPIn 引脚的位

bit 7-5 **未实现：**读为 0

bit 4-0 **U1RXR<4:0>：**将 UART1 接收 （U1RX）分配给对应 RPn 或 RPIn 引脚的位

寄存器 10-10： RPINR19：外设引脚选择输入寄存器 19

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	U2CTSR4	U2CTSR3	U2CTSR2	U2CTSR1	U2CTSR0
bit 15							bit 8

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	U2RXR4	U2RXR3	U2RXR2	U2RXR1	U2RXR0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **U2CTSR<4:0>：**将 UART2 允许发送 （$\overline{\text{U2CTS}}$）分配给对应 RPn 或 RPIn 引脚的位

bit 7-5 **未实现：**读为 0

bit 4-0 **U2RXR<4:0>：**将 UART2 接收 （U2RX）分配给对应 RPn 或 RPIn 引脚的位

寄存器 10-11： RPINR20：外设引脚选择输入寄存器 20

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	SCK1R4	SCK1R3	SCK1R2	SCK1R1	SCK1R0
bit 15							bit 8

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	SDI1R4	SDI1R3	SDI1R2	SDI1R1	SDI1R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **SCK1R<4:0>：**将 SPI1 时钟输入 （SCK1IN）分配给对应 RPn 或 RPIn 引脚的位

bit 7-5 **未实现：**读为 0

bit 4-0 **SDI1R<4:0>：**将 SPI1 数据输入 （SDI1）分配给对应 RPn 或 RPIn 引脚的位

寄存器 10-12： RPINR21：外设引脚选择输入寄存器 21

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	SS1R4	SS1R3	SS1R2	SS1R1	SS1R0
bit 7							bit 0

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

bit 15-5 **未实现：**读为 0

bit 4-0 **SS1R<4:0>：**将 SPI1 从选择输入 （SS1IN）分配给对应 RPn 或 RPIn 引脚的位

寄存器 10-13： RPINR22：外设引脚选择输入寄存器 22

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	SCK2R4	SCK2R3	SCK2R2	SCK2R1	SCK2R0
bit 15							bit 8

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	SDI2R4	SDI2R3	SDI2R2	SDI2R1	SDI2R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **SCK2R<4:0>：**将 SPI2 时钟输入 （SCK2IN）分配给对应 RPn 或 RPIn 引脚的位

bit 7-5 **未实现：**读为 0

bit 4-0 **SDI2R<4:0>：**将 SPI2 数据输入 （SDI2）分配给对应 RPn 或 RPIn 引脚的位

寄存器 10-14： RPINR23：外设引脚选择输入寄存器 23

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	SS2R4	SS2R3	SS2R2	SS2R1	SS2R0
bit 7							bit 0

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

bit 15-5 **未实现：**读为 0

bit 4-0 **SS2R<4:0>：**将 SPI2 从选择输入 （SS2IN）分配给对应 RPn 或 RPIn 引脚的位

寄存器 10-15： RPOR0：外设引脚选择输出寄存器 0

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP1R4	RP1R3	RP1R2	RP1R1	RP1R0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP0R4	RP0R3	RP0R2	RP0R1	RP0R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **RP1R<4:0>：**RP1 输出引脚映射位
将外设输出编号 n 分配给引脚 RP1 （请参见表 10-3 了解外设功能编号）。

bit 7-5 **未实现：**读为 0

bit 4-0 **RP0R<4:0>：**RP0 输出引脚映射位
将外设输出编号 n 分配给引脚 RP0 （请参见表 10-3 了解外设功能编号）。

寄存器 10-16： RPOR1：外设引脚选择输出寄存器 1

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP3R4	RP3R3	RP3R2	RP3R1	RP3R0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP2R4	RP2R3	RP2R2	RP2R1	RP2R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **RP3R<4:0>：**RP3 输出引脚映射位
将外设输出编号 n 分配给引脚 RP3 （请参见表 10-3 了解外设功能编号）。

bit 7-5 **未实现：**读为 0

bit 4-0 **RP2R<4:0>：**RP2 输出引脚映射位
将外设输出编号 n 分配给引脚 RP2 （请参见表 10-3 了解外设功能编号）。

寄存器 10-17： RPOR2：外设引脚选择输出寄存器 2

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP5R4	RP5R3	RP5R2	RP5R1	RP5R0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP4R4	RP4R3	RP4R2	RP4R1	RP4R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **RP5R<4:0>：**RP5 输出引脚映射位 **(1)**
将外设输出编号 n 分配给引脚 RP5 （请参见表 10-3 了解外设功能编号）。

bit 7-5 **未实现：**读为 0

bit 4-0 **RP4R<4:0>：**RP4 输出引脚映射位
将外设输出编号 n 分配给引脚 RP4 （请参见表 10-3 了解外设功能编号）。

寄存器 10-18： RPOR3：外设引脚选择输出寄存器 3

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP7R4	RP7R3	RP7R2	RP7R1	RP7R0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP6R4	RP6R3	RP6R2	RP6R1	RP6R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **RP7R<4:0>：**RP7 输出引脚映射位
将外设输出编号 n 分配给引脚 RP7 （请参见表 10-3 了解外设功能编号）。

bit 7-5 **未实现：**读为 0

bit 4-0 **RP6R<4:0>：**RP6 输出引脚映射位
将外设输出编号 n 分配给引脚 RP6 （请参见表 10-3 了解外设功能编号）。

寄存器 10-19： RPOR4：外设引脚选择输出寄存器 4

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP9R4	RP9R3	RP9R2	RP9R1	RP9R0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP8R4	RP8R3	RP8R2	RP8R1	RP8R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **RP9R<4:0>：**RP9 输出引脚映射位
将外设输出编号 n 分配给引脚 RP9 （请参见表 10-3 了解外设功能编号）。

bit 7-5 **未实现：**读为 0

bit 4-0 **RP8R<4:0>：**RP8 输出引脚映射位
将外设输出编号 n 分配给引脚 RP8 （请参见表 10-3 了解外设功能编号）。

寄存器 10-20： RPOR5：外设引脚选择输出寄存器 5

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP11R4	RP11R3	RP11R2	RP11R1	RP11R0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP10R4	RP10R3	RP10R2	RP10R1	RP10R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **RP11R<4:0>：**RP11 输出引脚映射位
将外设输出编号 n 分配给引脚 RP11 （请参见表 10-3 了解外设功能编号）。

bit 7-5 **未实现：**读为 0

bit 4-0 **RP10R<4:0>：**RP10 输出引脚映射位
将外设输出编号 n 分配给引脚 RP10 （请参见表 10-3 了解外设功能编号）。

寄存器 10-21： RPOR6：外设引脚选择输出寄存器 6

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP13R4	RP13R3	RP13R2	RP13R1	RP13R0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP12R4	RP12R3	RP12R2	RP12R1	RP12R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **RP13R<4:0>：**RP13 输出引脚映射位
将外设输出编号 n 分配给引脚 RP13 （请参见表 10-3 了解外设功能编号）。

bit 7-5 **未实现：**读为 0

bit 4-0 **RP12R<4:0>：**RP12 输出引脚映射位
将外设输出编号 n 分配给引脚 RP12 （请参见表 10-3 了解外设功能编号）。

寄存器 10-22： RPOR7：外设引脚选择输出寄存器 7

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP15R4	RP15R3	RP15R2	RP15R1	RP15R0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP14R4	RP14R3	RP14R2	RP14R1	RP14R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **RP15R<4:0>：**RP15 输出引脚映射位
将外设输出编号 n 分配给引脚 RP15 （请参见表 10-3 了解外设功能编号）。

bit 7-5 **未实现：**读为 0

bit 4-0 **RP14R<4:0>：**RP14 输出引脚映射位
将外设输出编号 n 分配给引脚 RP14 （请参见表 10-3 了解外设功能编号）。

寄存器 10-23： RPOR8：外设引脚选择输出寄存器 8(1)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP17R4	RP17R3	RP17R2	RP17R1	RP17R0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP16R4	RP16R3	RP16R2	RP16R1	RP16R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **RP17R<4:0>：**RP17 输出引脚映射位
将外设输出编号 n 分配给引脚 RP17（请参见表 10-3 了解外设功能编号）。

bit 7-5 **未实现：**读为 0

bit 4-0 **RP16R<4:0>：**RP16 输出引脚映射位
将外设输出编号 n 分配给引脚 RP16（请参见表 10-3 了解外设功能编号）。

注 1： 该寄存器在 28 引脚器件上未实现；所有位均读为 0。

寄存器 10-24： RPOR9：外设引脚选择输出寄存器 9(1)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP19R4	RP19R3	RP19R2	RP19R1	RP19R0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP18R4	RP18R3	RP18R2	RP18R1	RP18R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **RP19R<4:0>：**RP19 输出引脚映射位
将外设输出编号 n 分配给引脚 RP19（请参见表 10-3 了解外设功能编号）。

bit 7-5 **未实现：**读为 0

bit 4-0 **RP18R<4:0>：**RP18 输出引脚映射位
将外设输出编号 n 分配给引脚 RP18（请参见表 10-3 了解外设功能编号）。

注 1： 该寄存器在 28 引脚器件上未实现；所有位均读为 0。

寄存器 10-25： RPOR10：外设引脚选择输出寄存器 10(1)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP21R4	RP21R3	RP21R2	RP21R1	RP21R0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP20R4	RP20R3	RP20R2	RP20R1	RP20R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **RP21R<4:0>：**RP21 输出引脚映射位
将外设输出编号 n 分配给引脚 RP21 （请参见表 10-3 了解外设功能编号）。

bit 7-5 **未实现：**读为 0

bit 4-0 **RP20R<4:0>：**RP20 输出引脚映射位
将外设输出编号 n 分配给引脚 RP20 （请参见表 10-3 了解外设功能编号）。

注 1： 该寄存器在 28 引脚器件上未实现；所有位均读为 0。

寄存器 10-26： RPOR11：外设引脚选择输出寄存器 11(1)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP23R4	RP23R3	RP23R2	RP23R1	RP23R0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP22R4	RP22R3	RP22R2	RP22R1	RP22R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **RP23R<4:0>：**RP23 输出引脚映射位
将外设输出编号 n 分配给引脚 RP23 （请参见表 10-3 了解外设功能编号）。

bit 7-5 **未实现：**读为 0

bit 4-0 **RP22R<4:0>：**RP22 输出引脚映射位
将外设输出编号 n 分配给引脚 RP22 （请参见表 10-3 了解外设功能编号）。

注 1： 该寄存器在 28 引脚器件上未实现；所有位均读为 0。

寄存器 10-27： RPOR12：外设引脚选择输出寄存器 12(1)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP25R4	RP25R3	RP25R2	RP25R1	RP25R0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	RP24R4	RP24R3	RP24R2	RP24R1	RP24R0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **RP25R<5:0>：**RP25 输出引脚映射位
将外设输出编号 n 分配给引脚 RP25（请参见表 10-3 了解外设功能编号）。

bit 7-5 **未实现：**读为 0

bit 4-0 **RP24R<5:0>：**RP24 输出引脚映射位
将外设输出编号 n 分配给引脚 RP24（请参见表 10-3 了解外设功能编号）。

注 1： 该寄存器在 28 引脚器件上未实现；所有位均读为 0。

注：

11.0 TIMER1

注：	本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 14 章 “定时器”**（DS39704A_CN）。

Timer1 模块是一个 16 位定时器，可作为实时时钟（Real-Time Clock，RTC）的时间计数器，或作为自由运行的间隔定时器 / 计数器。Timer1 可在以下三种模式下工作：

- 16 位定时器
- 16 位同步计数器
- 16 位异步计数器

Timer1 还支持以下功能：

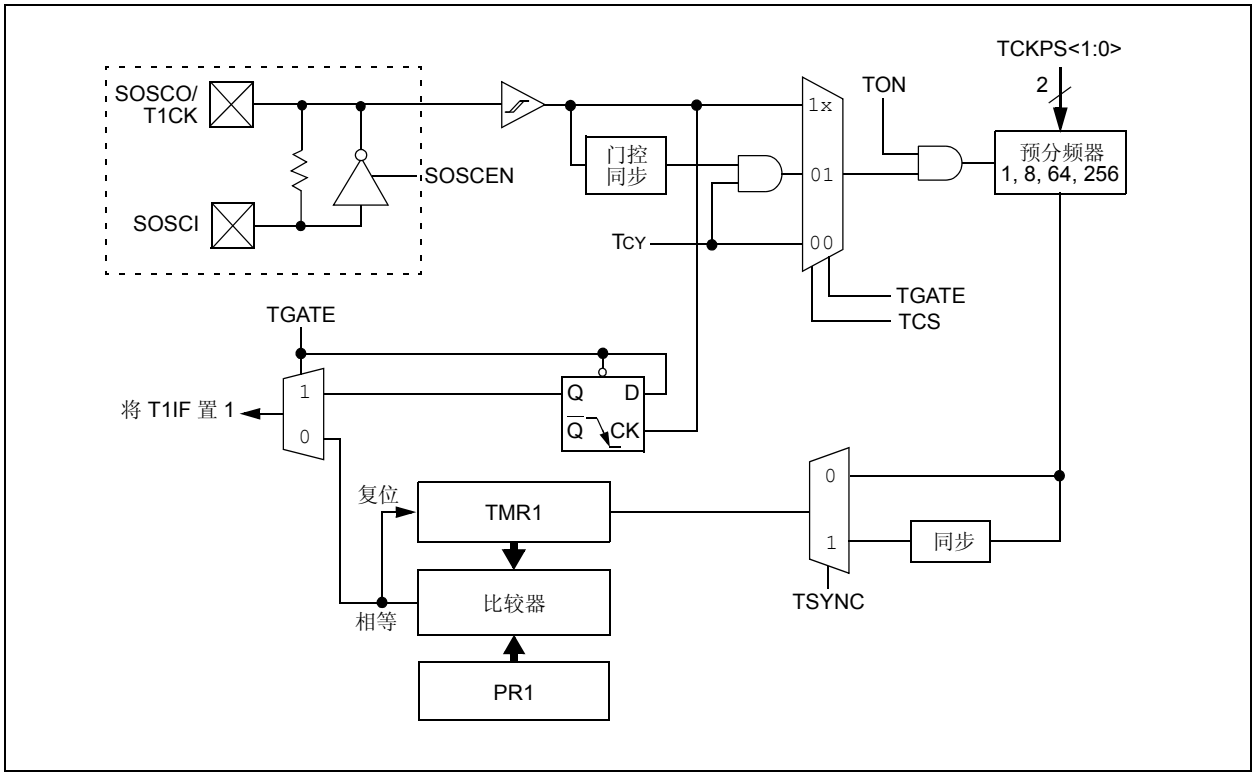
- 定时器门控操作
- 可选择的预分频比设置
- 在 CPU 空闲和休眠模式期间的定时器工作
- 在 16 位周期寄存器匹配时或外部门控信号的下降沿产生中断

图 11-1 给出了 16 位定时器模块的框图。

配置 Timer1 的操作：

1. 将 TON 位置 1 （= 1）。
2. 使用 TCKPS<1:0> 位选择定时器预分频比。
3. 使用 TCS 和 TGATE 位设置时钟和门控模式。
4. 将TSYNC位置1或清零来配置同步或异步操作。
5. 将定时器的周期值装入 PR1 寄存器。
6. 如果需要中断，将中断允许位 T1IE 置 1。使用优先级位 T1IP<2:0> 来设置中断优先级。

图 11-1： 16 位 TIMER1 模块框图



寄存器 11-1： T1CON：TIMER1 控制寄存器 (1)

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0
—	TGATE	TCKPS1	TCKPS0	—	TSYNC	TCS	—
bit 7							bit 0

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

bit 15 **TON**：Timer1 使能位
1 = 启动 16 位 Timer1
0 = 停止 16 位 Timer1

bit 14 **未实现**：读为 0

bit 13 **TSIDL**：空闲模式停止位
1 = 当器件进入空闲模式时，模块停止工作
0 = 在空闲模式下模块继续工作

bit 12-7 **未实现**：读为 0

bit 6 **TGATE**：Timer1 门控时间累加使能位
当 TCS = 1 时：
该位为无关位。
当 TCS = 0 时：
1 = 使能门控时间累加
0 = 禁止门控时间累加

bit 5-4 **TCKPS<1:0>**：Timer1 输入时钟预分频比选择位
11 = 1:256
10 = 1:64
01 = 1:8
00 = 1:1

bit 3 **未实现**：读为 0

bit 2 **TSYNC**：Timer1 外部时钟输入同步选择位
当 TCS = 1 时：
1 = 同步外部时钟输入
0 = 不同步外部时钟输入
当 TCS = 0 时：
该位为无关位。

bit 1 **TCS**：Timer1 时钟源选择位
1 = 来自 T1CK 引脚的外部时钟 （上升沿触发计数）
0 = 内部时钟 （Fosc/2）

bit 0 **未实现**：读为 0

注 1： 在定时器运行 （TON = 1）时更改 TxCON 的值会导致定时器预分频计数器复位，因此不建议这样做。

12.0 TIMER2/3 和 TIMER4/5

注：	本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 14 章“定时器”**（DS39704A_CN）。

Timer2/3 和 Timer4/5 模块为 32 位定时器，也可被配置为 4 个具有可选工作模式的独立 16 位定时器。

作为 32 位定时器，Timer2/3 和 Timer4/5 各具有三种工作模式：

- 具有所有 16 位工作模式 （异步计数器模式除外）的两个独立的 16 位定时器
- 单个 32 位定时器
- 单个 32 位同步计数器

Timer2/3 还支持以下功能：

- 定时器门控操作
- 可选择的预分频比设置
- 空闲和休眠模式期间的定时器工作
- 在 32 位周期寄存器匹配时产生中断
- ADC 事件触发功能 （仅限 Timer4/5）

所有 4 个 16 位定时器都能单独用作同步定时器或计数器。它们也提供上面所列的功能，但 ADC 事件触发功能除外；这仅由Timer5实现。通过设置T2CON、T3CON、T4CON 和 T5CON 寄存器中的相应位来确定工作模式和要被使能的功能。 T2CON 和 T4CON 在寄存器 12-1 中作了一般介绍； T3CON和T5CON如寄存器12-2所示。

作为 32 位定时器 / 计数器工作时，Timer2 和 Timer4 是 32 位定时器的最低有效字，而 Timer3 和 Timer5 是最高有效字。

注：	对于 32 位工作， T3CON 和 T5CON 控制位将被忽略。设置和控制只使用 T2CON 和 T4CON 控制位。 32 位定时器模块采用 Timer2 和 Timer4 时钟和门控输入，但产生中断时会将 Timer3 或 Timer5 中断标志位置 1。

要将 Timer2/3 或 Timer4/5 配置为 32 位工作：

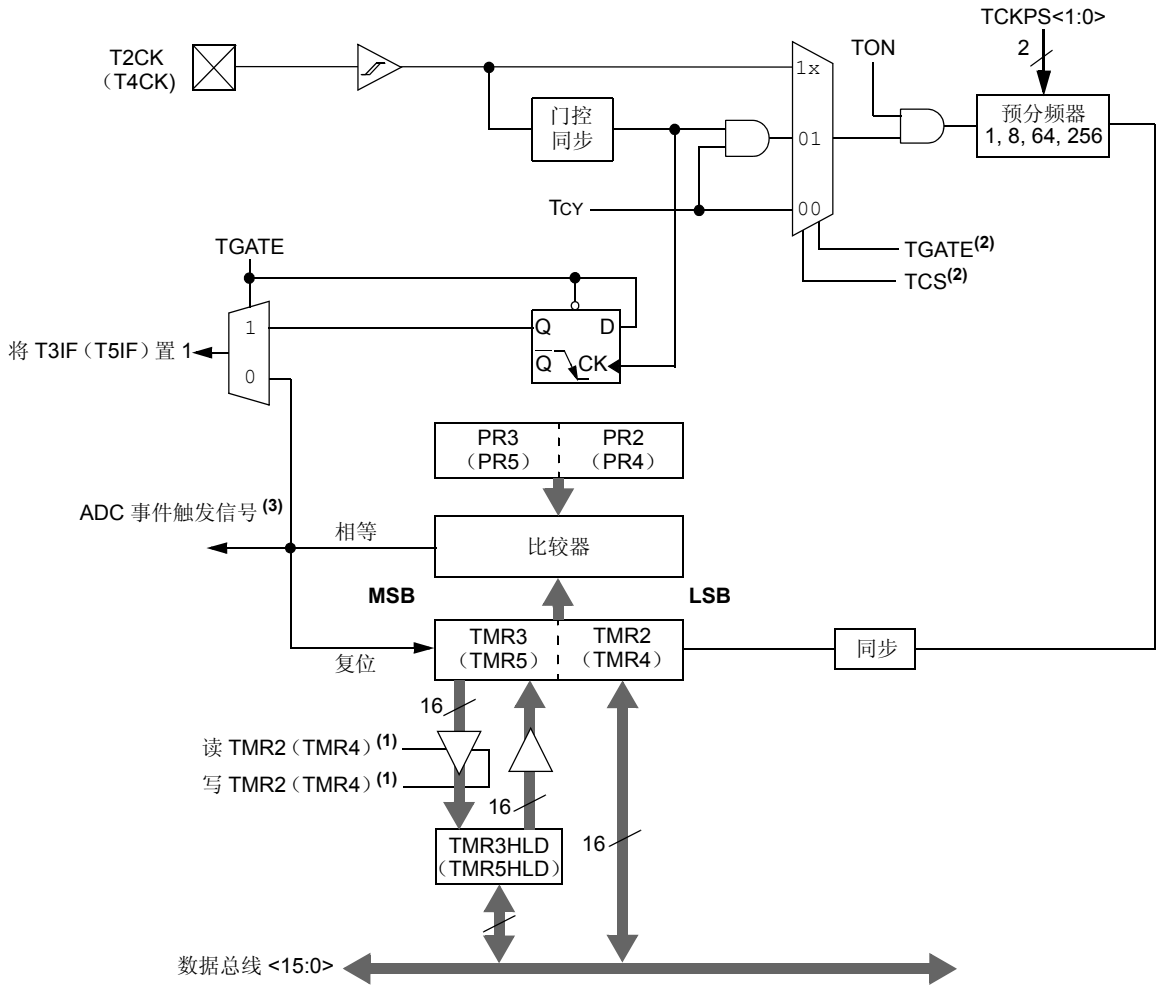
1. 将T32位置1（T2CON<3>或T4CON<3> = 1）。
2. 使用 TCKPS<1:0> 位为 Timer2 或 Timer4 选择预分频比。
3. 使用 TCS 和 TGATE 位设置时钟和门控模式。如果 TCS 被设置为外部时钟，则 RPINRx（TxCK）必须配置为可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。
4. 装入定时器的周期值。PR3（或 PR5）将包含值的最高有效字，而 PR2 （或 PR4）包含最低有效字。
5. 如果需要中断，将中断允许位 T3IE 或 T5IE 置 1；使用中断优先级位T3IP<2:0>或T5IP<2:0>来设置中断优先级。注意，Timer2或Timer4控制定时器，而中断则表现为 Timer3 或 Timer5 中断。
6. 将 TON 位置 1 （= 1）。

任意时刻定时器的值被存储在寄存器对 TMR3:TMR2（或 TMR5:TMR4）中。TMR3（TMR5）总是包含计数值的最高有效字，而 TMR2（TMR4）包含最低有效字。

要将任一定时器配置为独立的 16 位定时器：

1. 清零与该定时器对应的 T32 位（Timer2 和 Timer3 的T2CON<3>或Timer4和Timer5的T4CON<3>）。
2. 使用 TCKPS<1:0> 位选择定时器预分频比。
3. 使用 TCS 和 TGATE 位设置时钟和门控模式。更多信息，请参见**第 10.4 节“外设引脚选择（PPS）”**。
4. 将定时器的周期值装入 PRx 寄存器。
5. 如果需要中断，将中断允许位 TxIE 置 1 ；使用中断优先级位 TxIP<2:0> 来设置中断优先级。
6. 将 TON 位置 1 （TxCON<15> = 1）。

图 12-1：　　TIMER2/3 和 TIMER4/5（32 位）框图

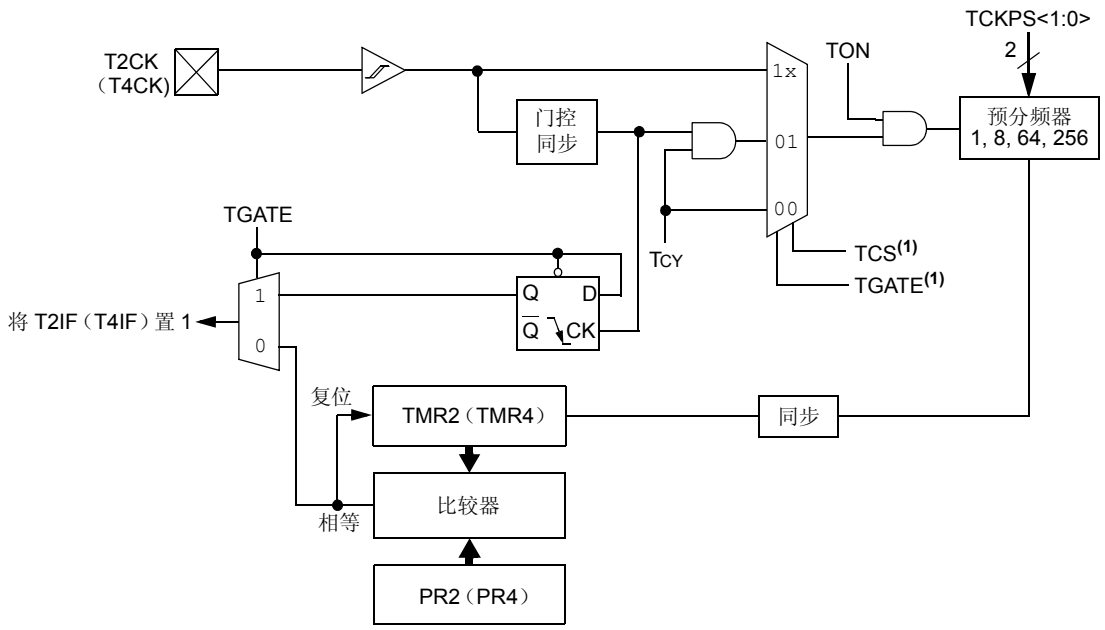


注　1： 必须将 32 位定时器配置位 T32 置 1，以使定时器工作在 32 位定时器 / 计数器模式下。所有控制位分别在 T2CON 和 T4CON 寄存器中。

2： 定时器时钟输入必须在使用前分配给可用的 RPn 引脚。更多信息，请参见**第 10.4 节“外设引脚选择（PPS）”**。

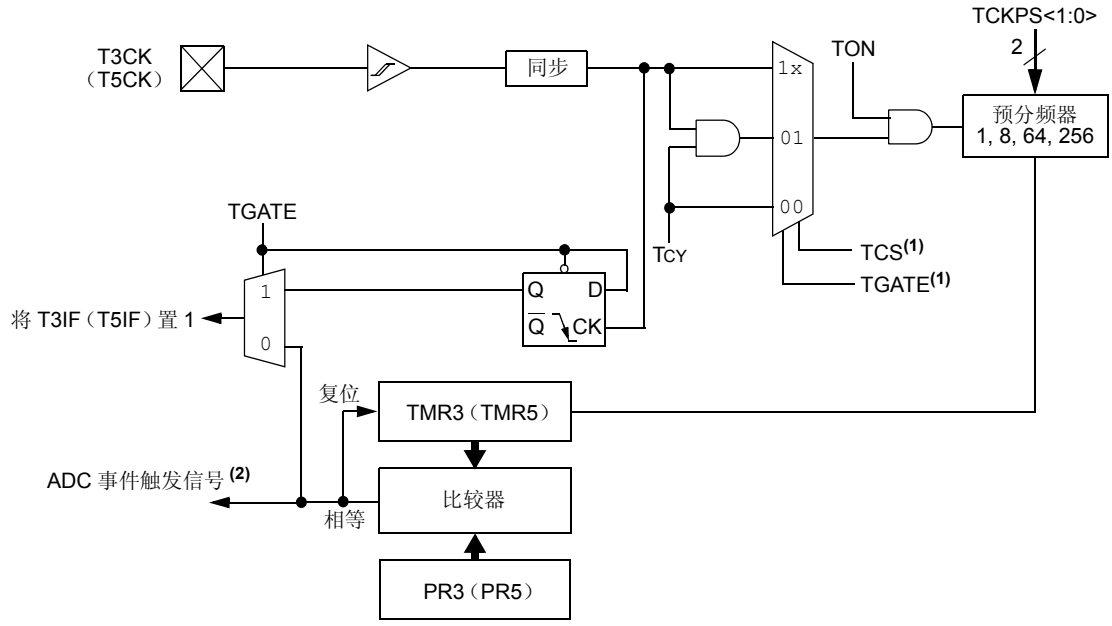
3： 只有 32 位模式下的 Timer 2/3 和 16 位模式下的 Timer3 可以产生 ADC 事件触发信号。

图 12-2：　　TIMER2 和 TIMER4 （16 位同步）框图



注　　1： 定时器时钟输入必须在使用前分配给可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。

图 12-3：　　TIMER3 和 TIMER5 （16 位异步）框图



注　　1： 定时器时钟输入必须在使用前分配给可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。
2： 只有 Timer3 可以产生 ADC 事件触发信号。

寄存器 12-1： TxCON：TIMER2 和 TIMER4 控制寄存器 (3)

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON	—	TSIDL	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	U-0
—	TGATE	TCKPS1	TCKPS0	T32(1)	—	TCS(2)	—
bit 7							bit 0

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

bit 15 **TON**：Timerx 使能位
当 TxCON<3> = 1 时：
1 = 启动 32 位 Timerx/y
0 = 停止 32 位 Timerx/y
当 TxCON<3> = 0 时：
1 = 启动 16 位 Timerx
0 = 停止 16 位 Timerx

bit 14 **未实现**：读为 0

bit 13 **TSIDL**：空闲模式停止位
1 = 当器件进入空闲模式时，模块停止工作
0 = 在空闲模式下模块继续工作

bit 12-7 **未实现**：读为 0

bit 6 **TGATE**：Timerx 门控时间累加使能位
当 TCS = 1 时：
该位为无关位。
当 TCS = 0 时：
1 = 使能门控时间累加
0 = 禁止门控时间累加

bit 5-4 **TCKPS<1:0>**：Timerx 输入时钟预分频比选择位
11 = 1:256
10 = 1:64
01 = 1:8
00 = 1:1

bit 3 **T32**：32 位定时器模式选择位 (1)
1 = Timerx 和 Timery 形成一个 32 位定时器
0 = Timerx 和 Timery 作为两个 16 位定时器
在 32 位模式下， T3CON 控制位不影响 32 位定时器的工作。

bit 2 **未实现**：读为 0

bit 1 **TCS**：Timerx 时钟源选择位 (2)
1 = 来自 TxCK 引脚的外部时钟 （上升沿触发计数）
0 = 内部时钟 （FOSC/2）

bit 0 **未实现**：读为 0

注 1： 在 32 位模式下， T3CON 或 T5CON 控制位不影响 32 位定时器的工作。
2： 如果 TCS = 1，则 RPINRx（TxCK）必须配置为可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。
3： 在定时器运行 （TON = 1）时更改 TxCON 的值会导致定时器预分频计数器复位，因此不建议这样做。

寄存器 12-2： TyCON：TIMER3 和 TIMER5 控制寄存器 (3)

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
TON(1)	—	TSIDL(1)	—	—	—	—	—
bit 15							bit 8

U-0	R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	U-0
—	TGATE(1)	TCKPS1(1)	TCKPS0(1)	—	—	TCS(1,2)	—
bit 7							bit 0

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

bit 15 **TON：**Timery 使能位 (1)
1 = 启动 16 位 Timery
0 = 停止 16 位 Timery

bit 14 **未实现：**读为 0

bit 13 **TSIDL：**空闲模式停止位 (1)
1 = 当器件进入空闲模式时，模块停止工作
0 = 在空闲模式下模块继续工作

bit 12-7 **未实现：**读为 0

bit 6 **TGATE：**Timery 门控时间累加使能位 (1)
当 TCS = 1 时：
该位为无关位。
当 TCS = 0 时：
1 = 使能门控时间累加
0 = 禁止门控时间累加

bit 5-4 **TCKPS<1:0>：**Timery 输入时钟预分频比选择位 (1)
11 = 1:256
10 = 1:64
01 = 1:8
00 = 1:1

bit 3-2 **未实现：**读为 0

bit 1 **TCS：**Timery 时钟源选择位 (1,2)
1 = 来自 TyCK 引脚的外部时钟 （上升沿触发计数）
0 = 内部时钟 （FOSC/2）

bit 0 **未实现：**读为 0

注 1： 当使能 32 位工作 （T2CON<3> 或 T4CON<3> = 1）时，这些位对 Timery 的工作没有影响；所有定时器功能都通过 T2CON 和 T4CON 进行设置。

2： 如果 TCS = 1，则 RPINRx （TxCK）必须配置为可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。

3： 在定时器运行 （TON = 1）时更改 TyCON 的值会导致定时器预分频计数器复位，因此不建议这样做。

注：

13.0 带专用定时器的输入捕捉

注：	本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 34 章“带专用定时器的输入捕捉”**（DS39722A_CN）。

PIC24FJ64GA104 系列中的器件都具有 9 个独立的输入捕捉模块。每个模块都提供了大量配置和工作选项，用于捕捉外部脉冲事件和产生中断。

输入捕捉模块的主要特性包括：

- 通过级联两个相邻的模块，可通过硬件配置为所有 32 位工作模式
- 输出比较操作有同步和触发两种模式，最多有 30 个用户可选择的触发 / 同步源可供使用
- 用于捕捉和保持几个事件的定时器值的 4 级 FIFO 缓冲区
- 产生可配置中断
- 每个模块最多有 6 种时钟源可供使用，可驱动一个单独的内部 16 位计数器

该模块通过两个寄存器 ICxCON1（寄存器 13-1）和 ICxCON2（寄存器 13-2）进行控制。图 13-1 给出了该模块的一般框图。

13.1 一般工作模式

13.1.1 同步和触发模式

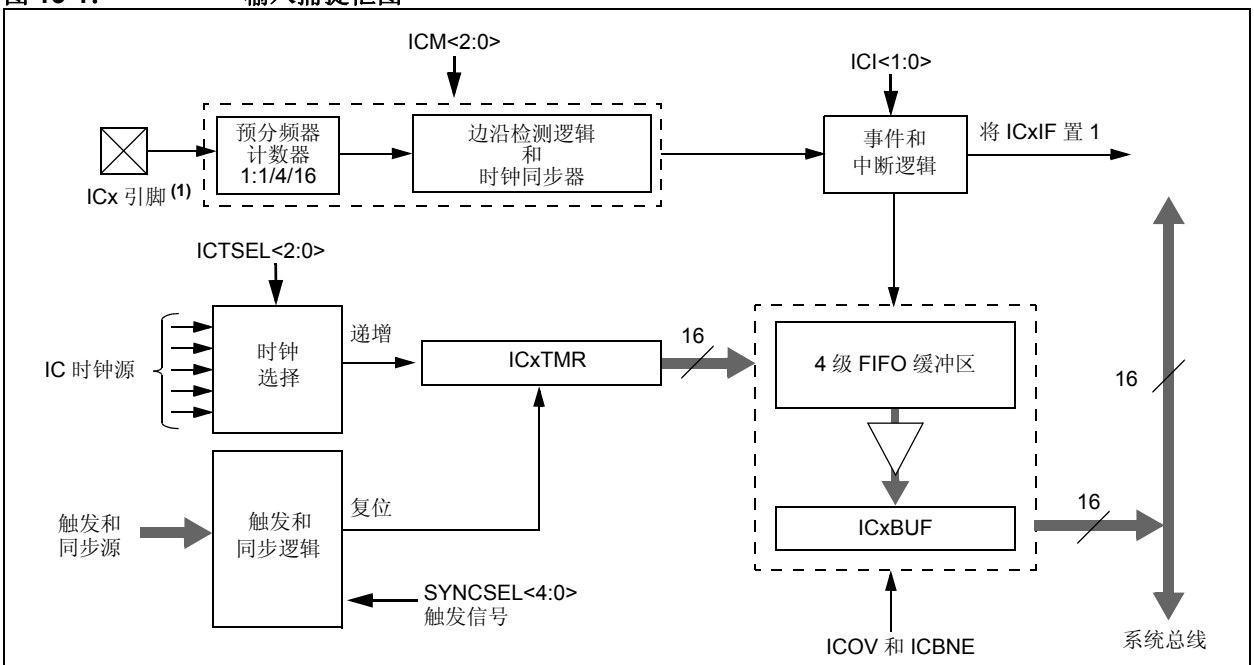
默认情况下，输入捕捉模块工作于自由运行模式。内部 16 位计数器 ICxTMR 连续递增计数，每次溢出时从 FFFFh 折回 0000h，其周期与选定的外部时钟源同步。发生捕捉事件时，内部计数器当前的 16 位值会被写入 FIFO 缓冲区。

在同步模式下，只要选定的时钟源被使能，模块就开始捕捉 ICx 引脚上的事件。只要选定的同步源发生事件，内部计数器就会复位。在触发模式下，模块在允许内部计数器运行之前会等待来自另一个内部模块的同步事件。

通过将 SYNCSEL 位设置为 00000 并清零 ICTRIG 位（ICxCON2<7>），可选择工作在标准的自由运行模式下。只要 SYNCSEL 位设置为除 00000 外的任何值，就可以选择同步和触发模式。可通过 ICTRIG 位选择同步或触发模式；将该位置 1 选择触发模式操作。在两种模式下， SYNCSEL 位都可决定同步 / 触发源。

当 SYNCSEL 位设置为 00000 且 ICTRIG 置 1 时，模块工作在软件触发模式下。在这种情况下，通过手动置 1 TRIGSTAT 位 （ICxCON2<6>）启动捕捉操作。

图 13-1： 输入捕捉框图



注 1： ICx 输入必须在使用前分配给可用的 RPn 引脚。更多信息，请参见**第 10.4 节“外设引脚选择 （PPS）”**。

13.1.2 级联（32 位）模式

默认情况下，每个模块都以其自身的 16 位定时器独立运行。要提高分辨率，可将相邻的偶数和奇数模块配置为一个 32 位模块。（例如，模块 1 和模块 2 配对，模块 3 和模块 4 配对等。）奇数编号的模块（ICx）为 32 位寄存器对提供低 16 位，而偶数编号的模块（ICy）提供高 16 位。 ICx 寄存器的折回操作会引起其相应的 ICy 寄存器递增 1。

对于这两个模块，级联操作都通过将 IC32 位（ICxCON2<8>）置 1 在硬件内完成配置。

13.2 捕捉操作

输入捕捉模块可配置为在 ICx 的上升沿或所有跳变处捕捉定时器值并产生中断。捕捉也可配置为在所有上升沿或仅在某些（每 4 个或 16 个）上升沿发生。可单独配置中断，使之在每个事件发生时，或者在一系列事件发生之后产生。

要为捕捉操作设置模块：

1. 将 ICx 输入配置为可用外设引脚选择引脚之一。
2. 如果要使用同步模式，在开始前禁止同步源。
3. 确保通过读 ICxBUF 删除了 FIFO 中的所有先前数据，直到 ICBNE 位（ICxCON1<3>）清零。
4. 设置 SYNCSEL 位（ICxCON2<4:0>）来获取所需的同步 / 触发源。
5. 设置 ICTSEL 位（ICxCON1<12:10>）来获取所需的时钟源。
6. 设置 ICI 位（ICxCON1<6:5>）来获取所需的中断频率。
7. 选择同步或触发模式操作：
   a) 确认 SYNCSEL 位未设置为 00000。
   b) 对于同步模式，清零 ICTRIG 位（ICxCON2<7>）。
   c) 对于触发模式，将 ICTRIG 置 1，将 TRIGSTAT 位（ICxCON2<6>）清零。
8. 设置 ICM 位（ICxCON1<2:0>）来获取所需的工作模式。
9. 使能选定的触发 / 同步源。

对于 32 位级联操作，设置步骤略有不同：

1. 将两个模块的 IC32 位（ICyCON2<8> 和 ICxCON2<8>）置 1，首先使能偶数编号的模块。这可以确保两个模块同时启动。
2. 将两个模块的 ICTSEL 和 SYNCSEL 位置 1，选择相同的同步 / 触发源以及时基源。先设置偶数模块，再设置奇数模块。两个模块必须使用相同的 ICTSEL 和 SYNCSEL 设置。
3. 将偶数模块的 ICTRIG 位（ICyCON2<7>）清零；这会强制该模块以与奇数模块同步的模式运行，与其触发设置无关。
4. 使用奇数模块的 ICI 位（ICxCON1<6:5>）来设置所需的中断频率。
5. 使用奇数模块的 ICTRIG 位（ICxCON2<7>）来配置触发或同步模式操作。

> **注：** 对于同步模式操作，使能同步源作为最后一个步骤。两个输入捕捉模块都将保持复位状态，直到使能同步源。

6. 使用奇数模块的 ICM 位（ICxCON1<2:0>）来设置所需的捕捉模式。

当使能时基和触发 / 同步源时，该模块为捕捉事件做好了准备。当 ICBNE 位（ICxCON1<3>）变为置 1 时，FIFO 中至少有一个捕捉值。读 FIFO 中的输入捕捉值，直到 ICBNE 清零。

对于 32 位操作，读 ICxBUF 和 ICyBUF（ICxBUF 用于 lsw， ICyBUF 用于 msw）获取完整的 32 位定时器值。当奇数模块的 ICBNE 位（ICxCON1<3>）变为置 1 时，FIFO 缓冲区中至少有一个捕捉值。持续读缓冲寄存器，直到 ICBNE 清零（由硬件自动执行）。

寄存器 13-1： ICxCON1：输入捕捉 x 控制寄存器 1

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0
—	—	ICSIDL	ICTSEL2	ICTSEL1	ICTSEL0	—	—
bit 15							bit 8

U-0	R/W-0	R/W-0	R-0, HCS	R-0, HCS	R/W-0	R/W-0	R/W-0
—	ICI1	ICI0	ICOV	ICBNE	ICM2(1)	ICM1(1)	ICM0(1)
bit 7							bit 0

图注：	HCS = 硬件清零 / 置 1 位		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15-14 **未实现：**读为 0

bit 13 **ICSIDL：**输入捕捉 x 模块在空闲模式下停止的控制位
1 = 在 CPU 空闲模式下输入捕捉模块停止工作
0 = 在 CPU 空闲模式下输入捕捉模块继续工作

bit 12-10 **ICTSEL<2:0>：**输入捕捉定时器选择位
111 = 系统时钟 （FOSC/2）
110 = 保留
101 = 保留
100 = Timer1
011 = Timer5
010 = Timer4
001 = Timer2
000 = Timer3

bit 9-7 **未实现：**读为 0

bit 6-5 **ICI<1:0>：**每次中断的捕捉次数选择位
11 = 每 4 次捕捉事件中断一次
10 = 每 3 次捕捉事件中断一次
01 = 每 2 次捕捉事件中断一次
00 = 每次捕捉事件中断一次

bit 4 **ICOV：**输入捕捉 x 溢出状态标志位 （只读）
1 = 发生了输入捕捉溢出
0 = 未发生输入捕捉溢出

bit 3 **ICBNE：**输入捕捉 x 缓冲区空状态位 （只读）
1 = 输入捕捉缓冲区非空，至少可以再读一个捕捉值
0 = 输入捕捉缓冲区为空

bit 2-0 **ICM<2:0>：**输入捕捉模式选择位 (1)
111 = 中断模式：仅当器件处于休眠或空闲模式时，输入捕捉可用作中断引脚（只检测上升沿，所有其他控制位都不适用）
110 = 未使用 （禁止模块）
101 = 预分频器捕捉模式：每 16 个上升沿捕捉一次
100 = 预分频器捕捉模式：每 4 个上升沿捕捉一次
011 = 简单捕捉模式：每个上升沿捕捉一次
010 = 简单捕捉模式：每个下降沿捕捉一次
001 = 边沿检测捕捉模式：每个边沿 （上升沿和下降沿）捕捉一次； ICI<1:0> 位不控制该模式下的中断产生
000 = 输入捕捉模块关闭

注 1： ICx 输入也必须配置给可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。

寄存器 13-2：　　ICxCON2：输入捕捉 x 控制寄存器 2

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	IC32
bit 15							bit 8

R/W-0	R/W-0, HS	U-0	R/W-0	R/W-1	R/W-1	R/W-0	R/W-1
ICTRIG	TRIGSTAT	—	SYNCSEL4	SYNCSEL3	SYNCSEL2	SYNCSEL1	SYNCSEL0
bit 7							bit 0

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

bit 15-9　**未实现：**读为 0

bit 8　**IC32：**级联两个 IC 模块的使能位 （32 位操作）
1 = ICx 和 ICy 级联成一个 32 位模块 （两个模块中的该位都必须置 1）
0 = ICx 单独用作一个 16 位模块

bit 7　**ICTRIG：**ICx 触发 / 同步选择位
1 = ICx 由 SYNCSELx 位指定的触发源触发
0 = ICx 由 SYNCSELx 位指定的同步源同步

bit 6　**TRIGSTAT：**定时器触发状态位
1 = 定时器源已触发并正在运行 （可以由硬件置 1，或使用软件置 1）
0 = 定时器源未触发并保持清零

bit 5　**未实现：**读为 0

bit 4-0　**SYNCSEL<4:0>：**触发 / 同步源选择位
11111 = 保留
11110 = 保留
11101 = 保留
11100 = CTMU(1)
11011 = A/D(1)
11010 = 比较器 3(1)
11001 = 比较器 2(1)
11000 = 比较器 1(1)
10111 = 输入捕捉 4
10110 = 输入捕捉 3
10101 = 输入捕捉 2
10100 = 输入捕捉 1
10011 = 保留
10010 = 保留
1000x = 保留
01111 = Timer5
01110 = Timer4
01101 = Timer3
01100 = Timer2
01011 = Timer1
01010 = 输入捕捉 5
01001 = 保留
01000 = 保留
00111 = 保留
00110 = 保留
00101 = 输出比较 5
00100 = 输出比较 4
00011 = 输出比较 3
00010 = 输出比较 2
00001 = 输出比较 1
00000 = 不与任何其他模块同步

注　1： 仅将这些输入作为触发源，从不用作同步源。

14.0 带专用定时器的输出比较

注：	本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 35 章“带专用定时器的输出比较”**（DS39723A_CN）。

PIC24FJ64GA104 系列中的器件都具有 9 个独立的输出比较模块。其中每一个模块都提供了大量配置和工作选项，以在发生器件内部事件时产生连续的脉冲，并可为驱动功率应用产生脉宽调制 （Pulse-Width Modulated，PWM）波形。

输出比较模块的主要特性包括：

- 通过级联两个相邻的模块，可通过硬件配置为所有 32 位工作模式
- 输出比较操作有同步和触发两种模式，最多有30个用户可选择的触发 / 同步源可供使用
- 两个单独的周期寄存器（一个主寄存器 OCxR 和一个辅助寄存器 OCxRS）为产生脉宽可变的脉冲提供了更大的灵活性
- 配置在发生输出事件时产生单脉冲或连续脉冲，或者产生连续的 PWM 波形
- 每个模块最多有 6 种时钟源可供使用，可驱动一个单独的内部 16 位计数器

14.1 一般工作模式

14.1.1 同步和触发模式

默认情况下，输出比较模块工作于自由运行模式。内部 16 位计数器 OCxTMR 连续递增计数，每次溢出时从 FFFFh 折回 0000h，其周期与选定的外部时钟源同步。内部计数器的值与周期寄存器的值每匹配一次，就会发生一次比较或 PWM 事件。

在同步模式下，只要选定的时钟源被使能，该模块就启动比较或 PWM 操作。只要选定的同步源发生事件，该模块的内部计数器就会复位。在触发模式下，该模块在允许计数器运行之前会等待来自另一个内部模块的同步事件。

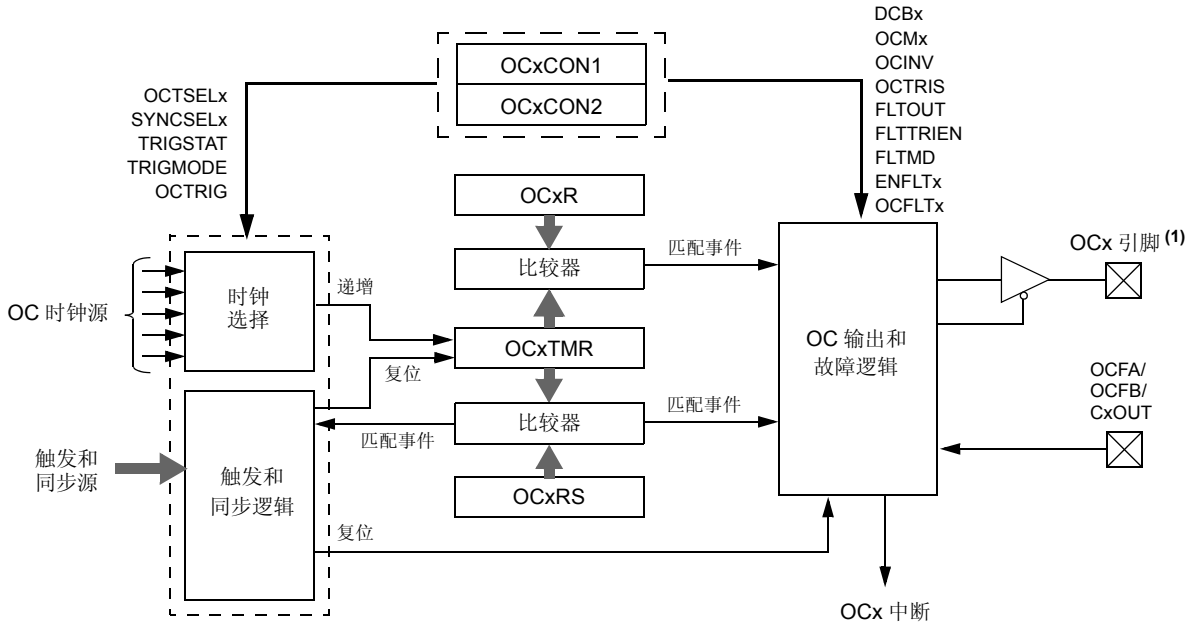
默认情况下或 SYNCSEL 位 （OCxCON2<4:0>）设置为 00000 时，选择自由运行模式。SYNCSEL 位设置为除 00000 外的任何值时，选择同步或触发模式。可通过 OCTRIG 位 （OCxCON2<7>）选择同步或触发模式；将该位置 1 选择触发模式操作。在两种模式下，SYNCSEL 位都可决定同步 / 触发源。

14.1.2 级联 （32 位）模式

默认情况下，每个模块都以其自身的一组 16 位定时器和占空比寄存器独立运行。要提高分辨率，可将相邻的偶数和奇数模块配置为一个 32 位模块。（例如，模块 1 和模块 2 配对，模块 3 和模块 4 配对等。）奇数编号的模块（OCx）为 32 位寄存器对提供低 16 位，而偶数编号的模块（OCy）提供高 16 位。OCx 寄存器的折回操作会引起其相应的 OCy 寄存器递增 1。

对于这两个模块，级联操作都通过将 OC32 位（OCxCON2<8>）置 1 在硬件内完成配置。

图 14-1：　　输出比较框图 （16 位模式）



注　　1： OCx 输出必须在使用前分配给可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。

14.2 比较操作

在比较模式下（图 14-1），可以配置输出比较模块以产生单事件脉冲或连续脉冲；也可以在每次定时器事件时重复翻转输出引脚。

要为比较操作设置模块：

1. 将OCx输出配置为可用的外设引脚选择引脚之一。
2. 计算 OCxR 和 （对于双比较模式） OCxRS 占空比寄存器所需的值：
   a) 确定指令时钟周期时间。考虑定时器源（如果使用了一个定时器源）的外部时钟频率和定时器预分频比的设置。
   b) 计算从定时器起始值 （0000h）到输出脉冲的上升沿所需的时间。
   c) 根据所需的脉冲宽度和到脉冲上升沿的时间，计算出现脉冲下降沿的时间。
3. 将上升沿的值写入 OCxR，将下降沿的值写入 OCxRS。
4. 将定时器周期寄存器PRy的值设置为等于或大于 OCxRS 中的值。
5. 对于触发模式操作，将 OCTRIG 置 1 可使能触发模式。将 TRIGMODE 置 1 或清零可配置触发操作，将 TRIGSTAT 置 1 或清零可选择硬件或软件触发。对于同步模式，可将 OCTRIG 清零。
6. 设置 SYNCSEL<4:0> 位可配置触发或同步源。如果要求自由运行定时器操作，可将 SYNCSEL 位设置为 00000 （无同步 / 触发源）。
7. 使用 OCTSEL<2:0> 位可选择时基源。如果需要，将使能比较时基进行计数的选定定时器的 TON 位置1。同步模式操作在时基使能时启动；而触发模式操作在发生了一个触发源事件后才启动。
8. 将 OCM<2:0> 位设置为适当的比较操作（= 0xx）。

对于 32 位级联操作，以下步骤也是必需的：

1. 将两个寄存器（OCyCON2<8> 和 OCxCON2<8>）的 OC32 位置 1。先使能偶数编号的模块，以确保奇偶模块同时启动。
2. 将偶数模块 （OCyCON2）的 OCTRIG 位清零，这样模块将在同步模式下运行。
3. 对 OCy 所需的输出和故障设置进行配置。
4. 通过将 OCTRIS 位清零强制 OCx 的输出引脚为输出状态。
5. 如果需要触发模式操作，通过 OCTRIG （OCxCON2<7>）、TRIGSTAT （OCxCON2<6>）和 SYNCSEL （OCxCON2<4:0>）位对 OCx 中的触发选项进行配置。
6. 先为 OCy 配置所需的比较或 PWM 工作模式（通过设置 OCM<2:0>），再为 OCx 进行此配置。

根据选择的输出模式，模块将使 OCx 引脚保持在默认状态，并在 OCxR 和定时器的数据匹配时强行翻转引脚的状态。在双比较模式下，当 OCxRS 与定时器的数据匹配时 OCx 将被强制回到默认状态。 OCxIF 中断标志在 OCxR 匹配（单比较模式下）和每次 OCxRS 匹配（双比较模式下）时被置 1。

单事件脉冲事件只发生一次，但是可以通过简单地重写 OCxCON1 寄存器的值即可重复。连续脉冲事件会无限地持续进行直到被终止。

14.3 脉宽调制 （PWM）模式

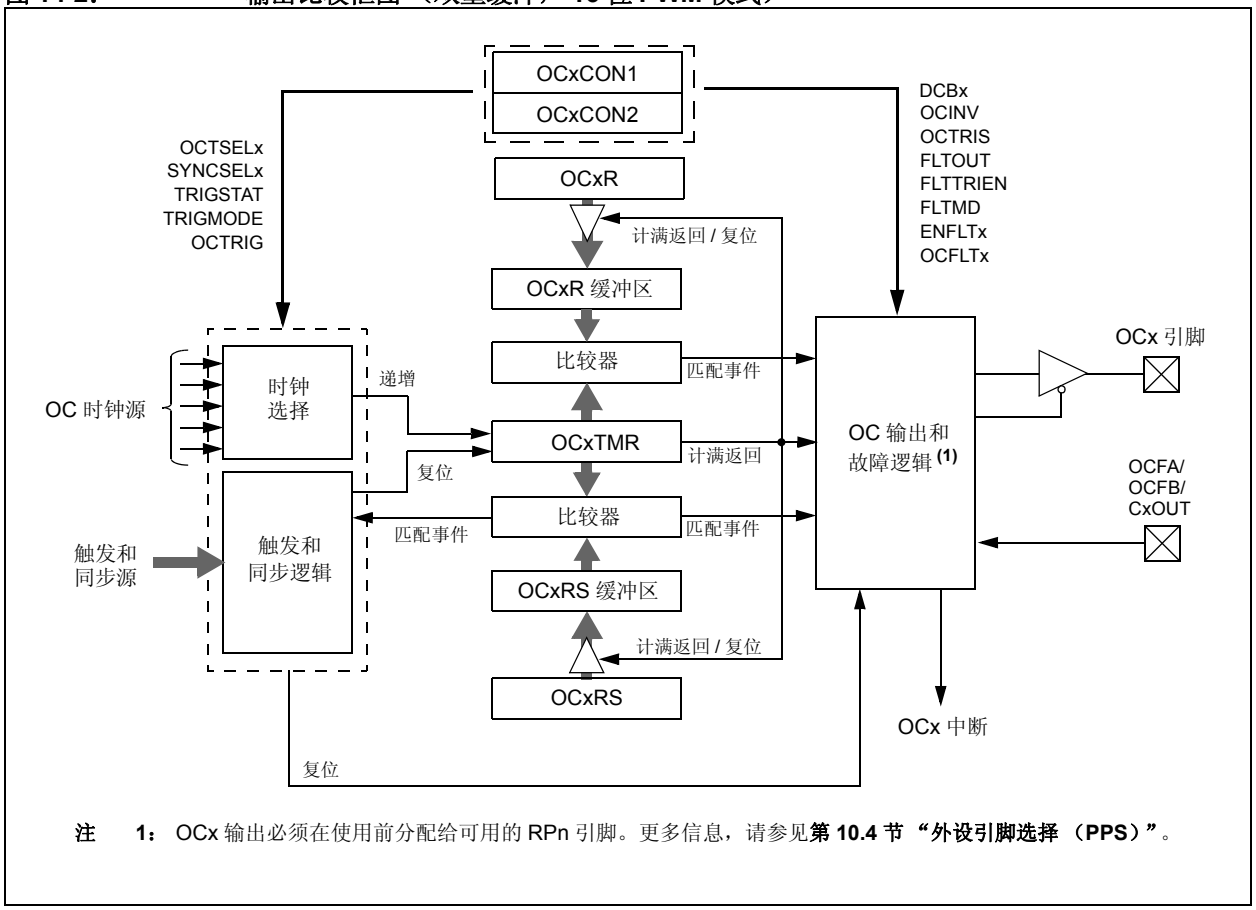
在 PWM 模式下，输出比较模块可配置为产生边沿对齐或中心对齐的脉冲波形。所有 PWM 操作都是双重缓冲的（缓冲寄存器在模块内部，并且不映射到SFR空间）。

要将输出比较模块配置为 PWM 操作：

1. 将OCx输出配置为可用的外设引脚选择引脚之一。
2. 计算所需的占空比并将其装入 OCxR 寄存器中。
3. 计算所需的周期并将其装入 OCxRS 寄存器中。
4. 通过将0x1F写入SYNCSEL<4:0>（OCxCON2<4:0>）并将 0 写入 OCTRIG （OCxCON2<7>）来选择当前的 OCx 作为同步源。
5. 通过写 OCTSEL2<2:0> （OCxCON<12:10>）位来选择时钟源。
6. 如果需要，允许定时器和输出比较模块的中断。使用 PWM 故障引脚时需要输出比较中断。
7. 在 OCM<2:0>（OCxCON1<2:0>）位中选择所需的 PWM 模式。
8. 如果选择定时器作为时钟源，则设置 TMRy 的预分频值并通过将 TON （TxCON<15>）位置 1 来使能时基。

注： 本外设包含的输入和输出功能可能需要通过外设引脚选择功能进行配置。更多信息，请参见**第 10.4 节 “外设引脚选择（PPS）”**。

图 14-2： 输出比较框图 （双重缓冲， 16 位 PWM 模式）



注 1： OCx 输出必须在使用前分配给可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。

14.3.1 PWM 周期

PWM 周期可通过写入 PRy（定时器周期寄存器）来指定。可以使用公式 14-1 来计算 PWM 周期。

公式 14-1： 计算 PWM 周期 (1)

PWM 周期 = [(PRy) + 1] • TCY •（定时器预分频值）

其中： PWM 频率 = 1/[PWM 周期]

注 1： 基于 TCY = TOSC * 2，打盹模式和 PLL 被禁止。

注： 如果 PRy 的值为 N，则会使 PWM 周期为 N + 1 个时基计数周期。例如，如果写入 PRy 寄存器的值为 7，则将产生由 8 个时基周期组成的 PWM 周期。

14.3.2 PWM 占空比

PWM 占空比通过写 OCxRS 和 OCxR 寄存器来指定。在任何时间都可以对 OCxRS 和 OCxR 寄存器进行写操作，但是在 PRy 和 TMRy 发生匹配 （即周期完成）前占空比值不会被锁存。这可以为 PWM 占空比提供双重缓冲，对于 PWM 的无毛刺操作是极其重要的。

PWM 占空比有一些重要的边界参数，包括：

- 如果 OCxR、OCxRS 和 PRy 中都装入 0000h，则 OCx 引脚将保持低电平 （占空比为 0%）。
- 如果 OCxRS 的值大于 PRy 的值，则引脚将保持高电平 （占空比为 100%）。

请参见例 14-1 了解 PWM 模式时序的详细信息。表 14-1 和表 14-2 给出了器件分别以 4 MIPS 和 10 MIPS 工作时，所对应的 PWM 频率和分辨率的示例。

公式 14-2： 计算最大 PWM 分辨率 (1)

$$\text{最大 PWM 分辨率（位）} = \frac{\log_{10}\left(\frac{F_{CY}}{F_{PWM} \bullet (\text{定时器预分频值})}\right)}{\log_{10}(2)} \text{ 位}$$

注 1： 基于 FCY = FOSC/2，打盹模式和 PLL 被禁止。

例 14-1： PWM 周期和占空比计算 (1)

1. 确定对应于所需的 PWM 频率为 52.08 kHz 的定时器周期寄存器的值，其中 FOSC = 8 MHz，带 PLL （器件时钟速率为 32 MHz）且 Timer2 预分频比设置为 1:1。

   TCY = 2 * TOSC = 62.5 ns

   PWM 周期 = 1/PWM 频率 = 1/52.08 kHz = 19.2 µs

   PWM 周期 = (PR2 + 1) • TCY • (Timer2 预分频值)

   19.2 µs = (PR2 + 1) • 62.5 ns • 1

   PR2 = 306

2. 在 PWM 频率为 52.08 kHz 且器件时钟速率为 32 MHz 时，计算占空比的最大分辨率。

   PWM 分辨率 = $\log_{10}(F_{CY}/F_{PWM})/\log_{10}2$) 位

   = ($\log_{10}$(16 MHz/52.08 kHz)/$\log_{10}$2) 位

   = 8.3 位

注 1： 基于 TCY = 2 * TOSC，打盹模式和 PLL 被禁止。

14.4 亚周期分辨率

通过 DCB 位（OCxCON2<10:9>）可实现优于一个指令周期的分辨率。使用这些位时，它们可以延迟由匹配事件产生的下降沿，延时等于指令周期的一定比例。

例如，设置 DCB<1:0> = 10 时，下降沿将在发生匹配事件的指令周期的中间产生，而不是在开始处产生。在 OCM<2:0> = 001 时，不能使用这些位。在模块工作于 PWM 模式（OCM<2:0> = 110 或 111）时，DCB 位将进行双重缓冲。

DCB 位用于与系统时钟相同的时钟源。在使用使能了预分频器的定时器时，DCB 位产生的下降沿延时将以系统时钟周期作为参考，而不是以定时器周期作为参考。

表 14-1： 4 MIPS （FCY = 4 MHz）时的 PWM 频率和分辨率示例 (1)

PWM 频率	7.6 Hz	61 Hz	122 Hz	977 Hz	3.9 kHz	31.3 kHz	125 kHz
定时器预分频比	8	1	1	1	1	1	1
周期寄存器的值	FFFFh	FFFFh	7FFFh	0FFFh	03FFh	007Fh	001Fh
分辨率（位）	16	16	15	12	10	7	5

注 1： 基于 FCY = FOSC/2，打盹模式和 PLL 被禁止。

表 14-2： 16 MIPS （FCY = 16 MHz）时的 PWM 频率和分辨率示例 (1)

PWM 频率	30.5 Hz	244 Hz	488 Hz	3.9 kHz	15.6 kHz	125 kHz	500 kHz
定时器预分频比	8	1	1	1	1	1	1
周期寄存器的值	FFFFh	FFFFh	7FFFh	0FFFh	03FFh	007Fh	001Fh
分辨率（位）	16	16	15	12	10	7	5

注 1： 基于 FCY = FOSC/2，打盹模式和 PLL 被禁止。

寄存器 14-1： OCxCON1：输出比较 x 控制寄存器 1

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	OCSIDL	OCTSEL2	OCTSEL1	OCTSEL0	ENFLT2(2)	ENFLT1
bit 15							bit 8

R/W-0	R/W-0, HCS	R/W-0, HCS	R/W-0, HCS	R/W-0	R/W-0	R/W-0	R/W-0
ENFLT0	OCFLT2	OCFLT1	OCFLT0	TRIGMODE	OCM2(1)	OCM1(1)	OCM0(1)
bit 7							bit 0

图注：	HCS = 硬件清零 / 置 1 位		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15-14 **未实现：**读为 0

bit 13 **OCSIDL：**在空闲模式下停止输出比较 x 的控制位
1 = 输出比较 x 在 CPU 空闲模式下停止工作
0 = 输出比较 x 在 CPU 空闲模式下继续工作

bit 12-10 **OCTSEL<2:0>：**输出比较 x 定时器选择位
111 = 系统时钟
110 = 保留
101 = 保留
100 = Timer1
011 = Timer5
010 = Timer4
001 = Timer3
000 = Timer2

bit 9 **ENFLT2：**比较器故障输入使能位 (2)
1 = 使能比较器故障输入
0 = 禁止比较器故障输入

bit 8 **ENFLT1：**OCFB 故障输入使能位
1 = 使能 OCFB 故障输入
0 = 禁止 OCFB 故障输入

bit 7 **ENFLT0：**OCFA 故障输入使能位
1 = 使能 OCFA 故障输入
0 = 禁止 OCFA 故障输入

bit 6 **OCFLT2：**PWM 比较器故障条件状态位 (2)
1 = 产生了 PWM 比较器故障条件 （该位只能由硬件清零）
0 = 未产生 PWM 比较器故障条件 （仅当 OCM<2:0> = 111 时，才使用该位）

bit 5 **OCFLT1：**PWM OCFB 故障条件状态位
1 = 产生了 PWM OCFB 故障条件 （该位只能由硬件清零）
0 = 未产生 PWM OCFB 故障条件 （仅当 OCM<2:0> = 111 时，才使用该位）

bit 4 **OCFLT0：**PWM OCFA 故障条件状态位
1 = 产生了 PWM OCFA 故障条件 （该位只能由硬件清零）
0 = 未产生 PWM OCFA 故障条件 （仅当 OCM<2:0> = 111 时，才使用该位）

bit 3 **TRIGMODE：**触发状态模式选择位
1 = TRIGSTAT （OCxCON2<6>）在 OCxRS = OCxTMR 时或者在软件中清零
0 = TRIGSTAT 只能用软件清零

注 1： OCx 输出也必须配置给可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。
2： 用于故障输入的比较器模块随 OCx 模块不同而不同。 OC1 和 OC2 使用比较器 1； OC3 和 OC4 使用比较器 2；OC5 使用比较器 3。

寄存器 14-1： OCxCON1：输出比较 x 控制寄存器 1 （续）

bit 2-0 **OCM<2:0>**：输出比较 x 模式选择位 **(1)**

111 = 在 OCx 上产生中心对齐的 PWM 模式
110 = 在 OCx 上产生边沿对齐的 PWM 模式
101 = 双比较连续脉冲模式：初始化 OCx 引脚为低电平，在连续周期内 OCxR 与 OCxRS 接连匹配时，持续翻转 OCx 状态
100 = 双比较单事件模式：初始化 OCx 引脚为低电平，在一个周期内 OCxR 与 OCxRS 匹配时，翻转 OCx 状态
011 = 单比较连续脉冲模式：比较事件持续翻转 OCx 引脚
010 = 单比较单事件模式：初始化 OCx 引脚为高电平，比较事件强制 OCx 引脚为低电平
001 = 单比较单事件模式：初始化 OCx 引脚为低电平，比较事件强制 OCx 引脚为高电平
000 = 禁止输出比较通道

注 1： OCx 输出也必须配置给可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。

2： 用于故障输入的比较器模块随 OCx 模块不同而不同。 OC1 和 OC2 使用比较器 1； OC3 和 OC4 使用比较器 2；OC5 使用比较器 3。

寄存器 14-2： OCxCON2：输出比较 x 控制寄存器 2

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
FLTMD	FLTOUT	FLTTRIEN	OCINV	—	DCB1(3)	DCB0(3)	OC32
bit 15							bit 8

R/W-0	R/W-0, HS	R/W-0	R/W-0	R/W-1	R/W-1	R/W-0	R/W-0
OCTRIG	TRIGSTAT	OCTRIS	SYNCSEL4	SYNCSEL3	SYNCSEL2	SYNCSEL1	SYNCSEL0
bit 7							bit 0

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

bit 15 **FLTMD**：故障模式选择位
1 = 故障模式将保持到故障源消除，并且对应的 OCFLT0 位用软件清零
0 = 故障模式将保持到故障源消除，并且新的 PWM 周期开始

bit 14 **FLTOUT**：故障输出位
1 = PWM 输出在发生故障时被驱动为高电平
0 = PWM 输出在发生故障时被驱动为低电平

bit 13 **FLTTRIEN**：故障输出状态选择位
1 = 引脚在发生故障条件时强制为输出
0 = 引脚 I/O 状态不受故障影响

bit 12 **OCINV**：OCMP 反相位
1 = OCx 输出反相
0 = OCx 输出不反相

bit 11 **未实现**：读为 0

bit 10-9 **DCB<1:0>**：OC 脉宽最低有效位 (3)
11 = 将 OCx 下降沿延迟 3/4 个指令周期
10 = 将 OCx 下降沿延迟 1/2 个指令周期
01 = 将 OCx 下降沿延迟 1/4 个指令周期
00 = OCx 下降沿在指令周期开始处产生

bit 8 **OC32**：级联两个 OC 模块的使能位 （32 位操作）
1 = 使能级联模块操作
0 = 禁止级联模块操作

bit 7 **OCTRIG**：OCx 触发 / 同步选择位
1 = OCx 由 SYNCSELx 位指定的触发源触发
0 = OCx 由 SYNCSELx 位指定的同步源同步

bit 6 **TRIGSTAT**：定时器触发状态位
1 = 定时器源已触发并正在运行
0 = 定时器源未触发并保持清零

bit 5 **OCTRIS**：OCx 输出引脚方向选择位
1 = OCx 引脚为三态
0 = OCx 引脚上连接了输出比较外设 x

注 1：绝不要通过选择该模式或其他相当的 SYNCSEL 设置将 OC 模块用作其自身的触发源。
2：仅将这些输入作为触发源，从不用作同步源。
3：当 OCINV = 1 时，这些位会影响上升沿。当 OCM 位 （OCxCON1<1:0>） = 001 时，这些位不起作用。

寄存器 14-2： OCxCON2：输出比较 x 控制寄存器 2 （续）

bit 4-0 **SYNCSEL<4:0>**：触发 / 同步源选择位

11111 = 该 OC 模块 (1)
11110 = 保留
11101 = 保留
11100 = CTMU(2)
11011 = A/D(2)
11010 = 比较器 3(2)
11001 = 比较器 2(2)
11000 = 比较器 1(2)
10111 = 输入捕捉 4(2)
10110 = 输入捕捉 3(2)
10101 = 输入捕捉 2(2)
10100 = 输入捕捉 1(2)
100xx = 保留
01111 = Timer5
01110 = Timer4
01101 = Timer3
01100 = Timer2
01011 = Timer1
01010 = 输入捕捉 5(2)
01001 = 保留
01000 = 保留
00111 = 保留
00110 = 保留
00101 = 输出比较 5(1)
00100 = 输出比较 4(1)
00011 = 输出比较 3(1)
00010 = 输出比较 2(1)
00001 = 输出比较 1(1)
00000 = 不与任何其他模块同步

注 1： 绝不要通过选择该模式或其他相当的 SYNCSEL 设置将 OC 模块用作其自身的触发源。
2： 仅将这些输入作为触发源，从不用作同步源。
3： 当 OCINV = 1 时，这些位会影响上升沿。当 OCM 位 （OCxCON1<1:0>） = 001 时，这些位不起作用。

15.0 串行外设接口（SPI）

注： 本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 23 章 “串行外设接口（SPI）”**（DS39699A_CN）。

串行外设接口（Serial Peripheral Interface，SPI）模块是用于同其他外设或单片机进行通信的同步串行接口。这些外设器件可以是串行 EEPROM、移位寄存器、显示驱动器和 A/D 转换器等。SPI 模块与 Motorola® 的 SPI 和 SIOP 接口兼容。PIC24FJ64GA104 系列的所有器件都包含 3 个 SPI 模块。

该模块支持在两种缓冲模式下工作。在标准模式下，数据通过单个串行缓冲区移动。在增强型缓冲模式下，数据通过一个 8 级深 FIFO 缓冲区移动。

注： 在标准或增强型缓冲模式下，都不要在 SPIxBUF 寄存器上执行读 - 修改 - 写操作（如针对位的指令）。

工作于主或从模式时，模块还支持基本的帧 SPI 协议。共支持 4 种帧 SPI 配置。

SPI 串行接口由 4 个引脚组成：

- SDIx：串行数据输入
- SDOx：串行数据输出
- SCKx：移位时钟输入或输出
- $\overline{SSx}$：低电平有效从选择或帧同步 I/O 脉冲

SPI 模块可以配置为使用 2、3 或 4 个引脚工作。在 3 引脚模式下，不使用 $\overline{SSx}$。在 2 引脚模式下，SDOx 和 $\overline{SSx}$ 均不使用。

标准和增强型模式下的模块框图如图15-1和图15-2所示。

注： 在本章中，SPI 模块统称为 SPIx，或分别称为 SPI1、SPI2 或 SPI3。特殊功能寄存器也使用类似的符号表示。例如，SPIxCON1 和 SPIxCON2 指 3 个 SPI 模块中任何一个的控制寄存器。

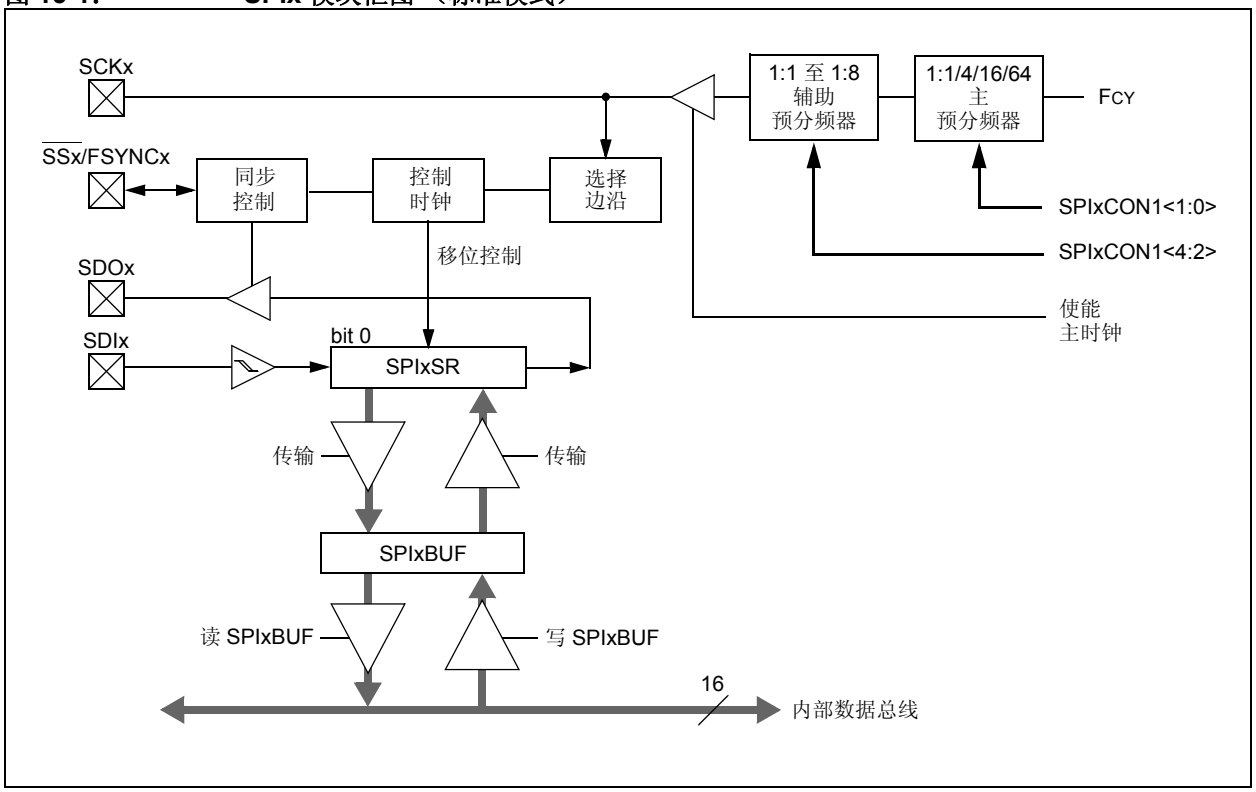
要将 SPI 模块设置为工作于标准主模式：

1. 如果使用中断：
   a) 将相应 IFS 寄存器中的 SPIxIF 位清零。
   b) 将相应 IEC 寄存器中的 SPIxIE 位置 1。
   c) 通过写相应 IPC 寄存器中的 SPIxIP 位来设置中断优先级。
2. 将所需的设置写入 SPIxCON1 和 SPIxCON2 寄存器，且 MSTEN （SPIxCON1<5>） = 1。
3. 将 SPIROV 位 （SPIxSTAT<6>）清零。
4. 通过将 SPIEN 位 （SPIxSTAT<15>）置 1 使能 SPI 工作。
5. 将待发送数据写入 SPIxBUF 寄存器。发送 （和接收）在数据写入 SPIxBUF 寄存器时立即开始。

要将 SPI 模块设置为工作于标准从模式：

1. 将 SPIxBUF 寄存器清零。
2. 如果使用中断：
   a) 将相应 IFS 寄存器中的 SPIxIF 位清零。
   b) 将相应 IEC 寄存器中的 SPIxIE 位置 1。
   c) 通过写相应 IPC 寄存器中的 SPIxIP 位来设置中断优先级。
3. 将所需的设置写入 SPIxCON1 和 SPIxCON2 寄存器，且 MSTEN （SPIxCON1<5>） = 0。
4. 将 SMP 位清零。
5. 如果 CKE 位 （SPIxCON1<8>）置 1，则必须将 SSEN位（SPIxCON1<7>）置1来使能SSx引脚。
6. 将 SPIROV 位 （SPIxSTAT<6>）清零。
7. 通过将 SPIEN 位 （SPIxSTAT<15>）置 1 使能 SPI 工作。

图 15-1： SPIx 模块框图 （标准模式）



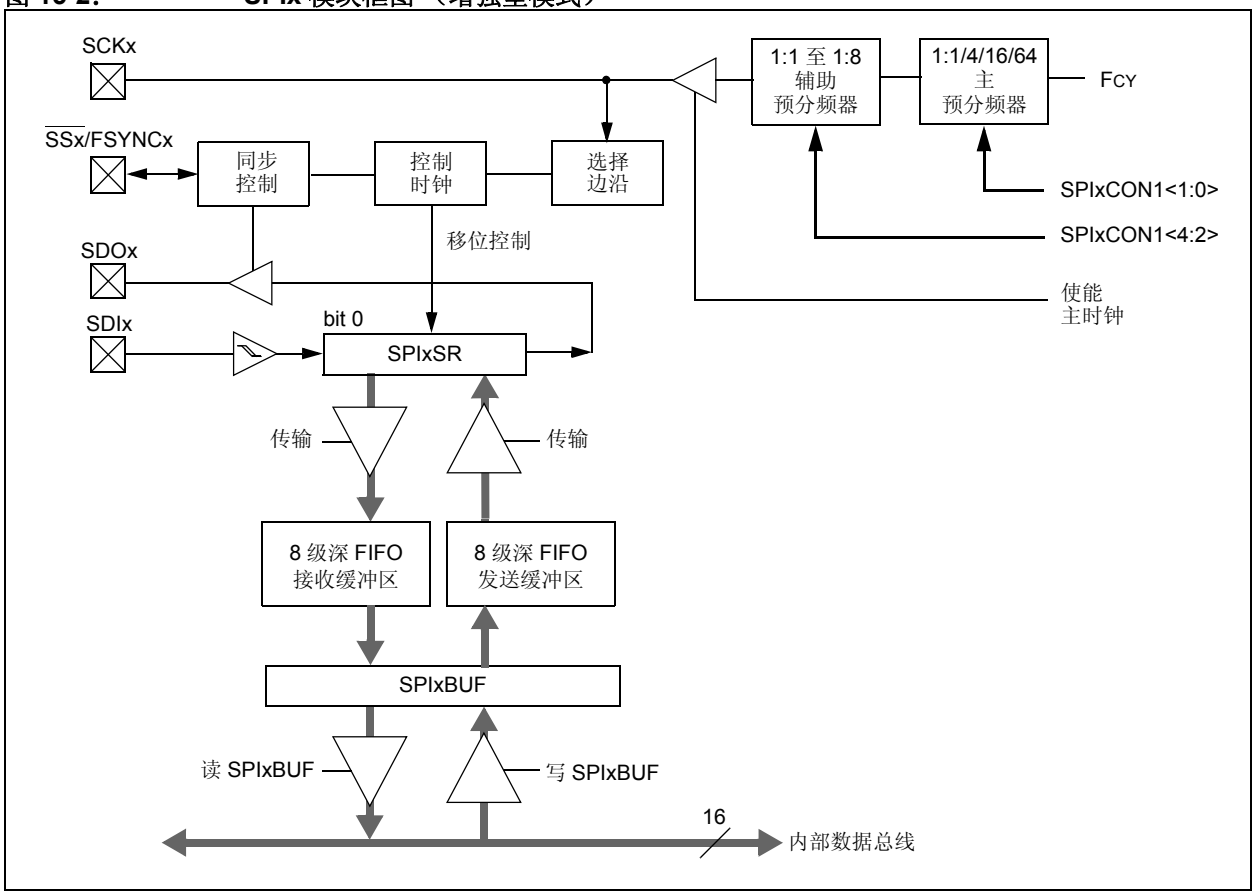
要将 SPI 模块设置为工作于增强型缓冲主模式：

1. 如果使用中断：
   a) 将相应 IFS 寄存器中的 SPIxIF 位清零。
   b) 将相应 IEC 寄存器中的 SPIxIE 位置 1。
   c) 写入相应 IPC 寄存器中的 SPIxIP 位。
2. 将所需的设置写入 SPIxCON1 和 SPIxCON2 寄存器，且 MSTEN （SPIxCON1<5>） = 1。
3. 将 SPIROV 位 （SPIxSTAT<6>）清零。
4. 通过将 SPIBEN 位 （SPIxCON2<0>）置 1 选择增强型缓冲模式。
5. 通过将 SPIEN 位 （SPIxSTAT<15>）置 1 使能 SPI 工作。
6. 将待发送数据写入 SPIxBUF 寄存器。发送 （和接收）在数据写入 SPIxBUF 寄存器时立即开始。

要将 SPI 模块设置为工作于增强型缓冲从模式：

1. 将 SPIxBUF 寄存器清零。
2. 如果使用中断：
   a) 将相应 IFS 寄存器中的 SPIxIF 位清零。
   b) 将相应 IEC 寄存器中的 SPIxIE 位置 1。
   c) 通过写相应 IPC 寄存器中的 SPIxIP 位来设置中断优先级。
3. 将所需的设置写入 SPIxCON1 和 SPIxCON2 寄存器，且 MSTEN （SPIxCON1<5>） = 0。
4. 将 SMP 位清零。
5. 如果 CKE 位置 1，则必须将 SSEN 位置 1 来使能 SSx 引脚。
6. 将 SPIROV 位 （SPIxSTAT<6>）清零。
7. 通过将 SPIBEN 位 （SPIxCON2<0>）置 1 选择增强型缓冲模式。
8. 通过将 SPIEN 位 （SPIxSTAT<15>）置 1 使能 SPI 工作。

图 15-2： SPIx 模块框图 （增强型模式）



寄存器 15-1： SPIxSTAT：SPIx 状态和控制寄存器

R/W-0	U-0	R/W-0	U-0	U-0	R-0	R-0	R-0
SPIEN(1)	—	SPISIDL	—	—	SPIBEC2	SPIBEC1	SPIBEC0
bit 15							bit 8

R-0	R/C-0, HS	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0
SRMPT	SPIROV	SRXMPT	SISEL2	SISEL1	SISEL0	SPITBF	SPIRBF
bit 7							bit 0

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

bit 15 **SPIEN**：SPIx 使能位 (1)
1 = 使能模块并将 SCKx、 SDOx、 SDIx 和 $\overline{SSx}$ 配置为串口引脚
0 = 禁止模块

bit 14 **未实现**：读为 0

bit 13 **SPISIDL**：空闲模式停止位
1 = 当器件进入空闲模式时，模块停止工作
0 = 在空闲模式下模块继续工作

bit 12-11 **未实现**：读为 0

bit 10-8 **SPIBEC<2:0>**：SPIx 缓冲区元素计数位 （在增强型缓冲模式下有效）
主模式：
SPI 传输等待的数目。
从模式：
SPI 传输未读的数目。

bit 7 **SRMPT**：移位寄存器 （SPIxSR）空位 （在增强型缓冲模式下有效）
1 = SPIx 移位寄存器为空，等待发送或接收
0 = SPIx 移位寄存器非空

bit 6 **SPIROV**：接收溢出标志位
1 = 一个新字节 / 字已完全接收并丢弃。用户软件尚未读取 SPIxBUF 寄存器中的先前数据。
0 = 未发生溢出

bit 5 **SRXMPT**：接收 FIFO 空位 （在增强型缓冲模式下有效）
1 = 接收 FIFO 为空
0 = 接收 FIFO 非空

bit 4-2 **SISEL<2:0>**：SPIx 缓冲区中断模式位 （在增强型缓冲模式下有效）
111 = 当 SPIx 发送缓冲区已满时产生中断 （SPITBF 位置 1）
110 = 当最后一位移入 SPIxSR 时产生中断；此时发送 FIFO 为空
101 = 当最后一位移出 SPIxSR 时产生中断；此时发送完成
100 = 当一个数据移入 SPIxSR 时产生中断；此时发送 FIFO 有一个空位
011 = 当 SPIx 接收缓冲区已满时产生中断 （SPIRBF 位置 1）
010 = 当 SPIx 接收缓冲区为 3/4 满或更满时产生中断
001 = 当接收缓冲区中有数据时产生中断 （SRMPT 位置 1）
000 = 当接收缓冲区中的最后数据被读取时产生中断；此时缓冲区为空 （SRXMPT 位置 1）

注 1： 如果 SPIEN = 1，这些功能必须在使用前分配给可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。

寄存器 15-1： SPIxSTAT：SPIx 状态和控制寄存器 （续）

bit 1 **SPITBF**：SPIx 发送缓冲区满状态位

1 = 发送尚未开始； SPIxTXB 为满
0 = 发送已开始； SPIxTXB 为空

在标准缓冲模式下：
当 CPU 通过写 SPIxBUF 存储单元装入 SPIxTXB 时，该位由硬件自动置 1。当 SPIx 模块将数据从 SPIxTXB 传输到 SPIxSR 时，该位由硬件自动清零。

在增强型缓冲模式下：
当 CPU 通过写 SPIxBUF 存储单元装入最后的可用缓冲单元时，该位由硬件自动置 1。当有缓冲单元可用于 CPU 写操作时，该位由硬件自动清零。

bit 0 **SPIRBF**：SPIx 接收缓冲区满状态位

1 = 接收完成， SPIxRXB 为满
0 = 接收未完成， SPIxRXB 为空

在标准缓冲模式下：
当 SPIx 将数据从 SPIxSR 传输到 SPIxRXB 时，该位由硬件自动置 1。当内核通过读 SPIxBUF 存储单元读 SPIxRXB 时，该位由硬件自动清零。

在增强型缓冲模式下：
当 SPIx 通过将数据从 SPIxSR 传送到缓冲区填充最后未读的缓冲单元时，该位由硬件自动置 1。当有缓冲单元可用于从 SPIxSR 进行传输时，该位由硬件自动清零。

注 1： 如果 SPIEN = 1，这些功能必须在使用前分配给可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。

寄存器 15-2： SPIxCON1：SPIx 控制寄存器 1

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	DISSCK(1)	DISSDO(2)	MODE16	SMP	CKE(3)
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SSEN(4)	CKP	MSTEN	SPRE2	SPRE1	SPRE0	PPRE1	PPRE0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12 **DISSCK：**禁止 SCKx 引脚位 （仅限 SPI 主模式）(1)
1 = 禁止内部 SPI 时钟；引脚用作 I/O
0 = 使能内部 SPI 时钟

bit 11 **DISSDO：**禁止 SDOx 引脚位 (2)
1 = 模块不使用 SDOx 引脚；引脚用作 I/O
0 = SDOx 引脚由模块控制

bit 10 **MODE16：**字 / 字节通信选择位
1 = 采用字宽 （16 位）通信
0 = 采用字节宽 （8 位）通信

bit 9 **SMP：**SPIx 数据输入采样阶段位
主模式：
1 = 在数据输出时间的末端采样输入数据
0 = 在数据输出时间的中间采样输入数据
从模式：
当在从模式下使用 SPIx 时，必须将 SMP 清零。

bit 8 **CKE：**SPIx 时钟边沿选择位 (3)
1 = 串行输出数据在时钟从工作状态转变为空闲状态时变化 （见 bit 6）
0 = 串行输出数据在时钟从空闲状态转变为工作状态时变化 （见 bit 6）

bit 7 **SSEN：**从选择使能 （从模式）位 (4)
1 = $\overline{SSx}$ 引脚用于从模式
0 = 模块不使用 $\overline{SSx}$ 引脚；引脚由端口功能控制

bit 6 **CKP：**时钟极性选择位
1 = 空闲状态时时钟信号为高电平；工作状态时为低电平
0 = 空闲状态时时钟信号为低电平；工作状态时为高电平

bit 5 **MSTEN：**主模式使能位
1 = 主模式
0 = 从模式

注 1： 如果 DISSCK = 0， SCKx 必须配置为可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择（PPS）”**。

2： 如果 DISSDO = 0， SDOx 必须配置为可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择（PPS）”**。

3： 在帧 SPI 模式下不使用 CKE 位。在帧 SPI 模式 （FRMEN = 1）下，用户应将该位编程为 0。

4： 如果 SSEN = 1，$\overline{SSx}$ 必须配置为可用的 RPn 引脚。更多信息，请参见**第 10.4 节“外设引脚选择（PPS）”**。

寄存器 15-2： SPIxCON1：SPIx 控制寄存器 1 （续）

bit 4-2 **SPRE<2:0>：**辅助预分频比位 （主模式）
111 = 辅助预分频比 1:1
110 = 辅助预分频比 2:1
...
000 = 辅助预分频比 8:1

bit 1-0 **PPRE<1:0>：**主预分频比位 （主模式）
11 = 主预分频比 1:1
10 = 主预分频比 4:1
01 = 主预分频比 16:1
00 = 主预分频比 64:1

注 1： 如果 DISSCK = 0， SCKx 必须配置为可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择（PPS）”**。

2： 如果 DISSDO = 0， SDOx 必须配置为可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择（PPS）”**。

3： 在帧 SPI 模式下不使用 CKE 位。在帧 SPI 模式 （FRMEN = 1）下，用户应将该位编程为 0。

4： 如果 SSEN = 1，$\overline{SSx}$ 必须配置为可用的 RPn 引脚。更多信息，请参见**第 10.4 节“外设引脚选择（PPS）”**。

寄存器 15-3： SPIxCON2：SPIx 控制寄存器 2

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
FRMEN	SPIFSD	SPIFPOL	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	SPIFE	SPIBEN
bit 7							bit 0

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

bit 15 **FRMEN：**帧 SPIx 支持位
1 = 使能帧 SPIx 支持
0 = 禁止帧 SPIx 支持

bit 14 **SPIFSD：**$\overline{SSx}$ 引脚帧同步脉冲方向控制位
1 = 帧同步脉冲输入 （从器件）
0 = 帧同步脉冲输出 （主器件）

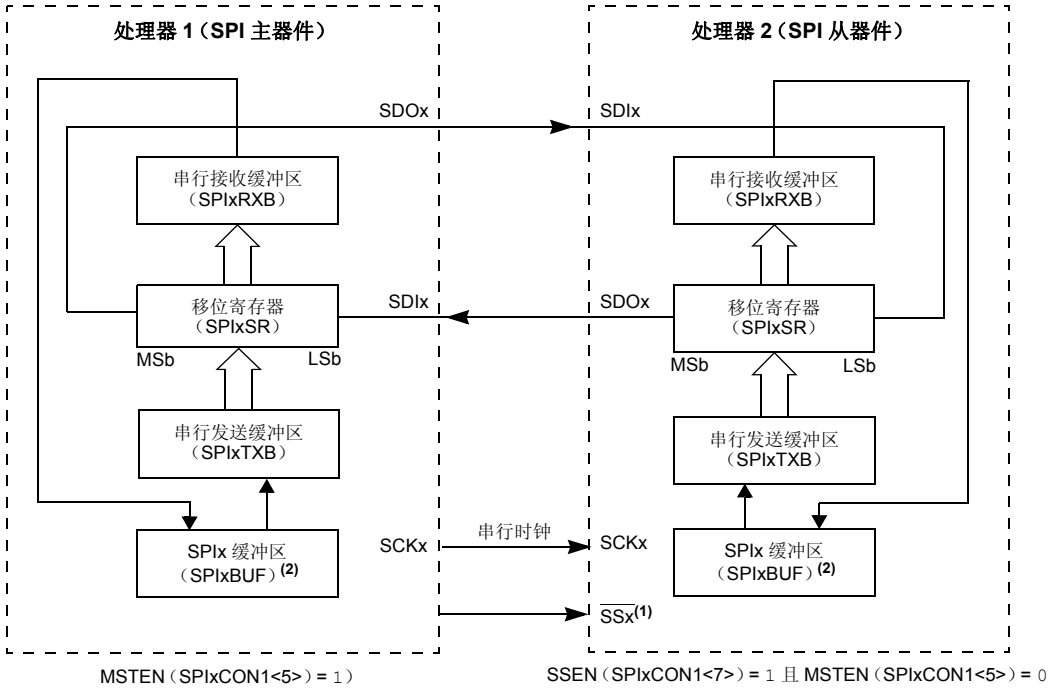
bit 13 **SPIFPOL：**帧同步脉冲极性位 （仅限帧模式）
1 = 帧同步脉冲为高电平有效
0 = 帧同步脉冲为低电平有效

bit 12-2 **未实现：**读为 0

bit 1 **SPIFE：**帧同步脉冲边沿选择位
1 = 帧同步脉冲与第一个位时钟一致
0 = 帧同步脉冲比第一个位时钟提前

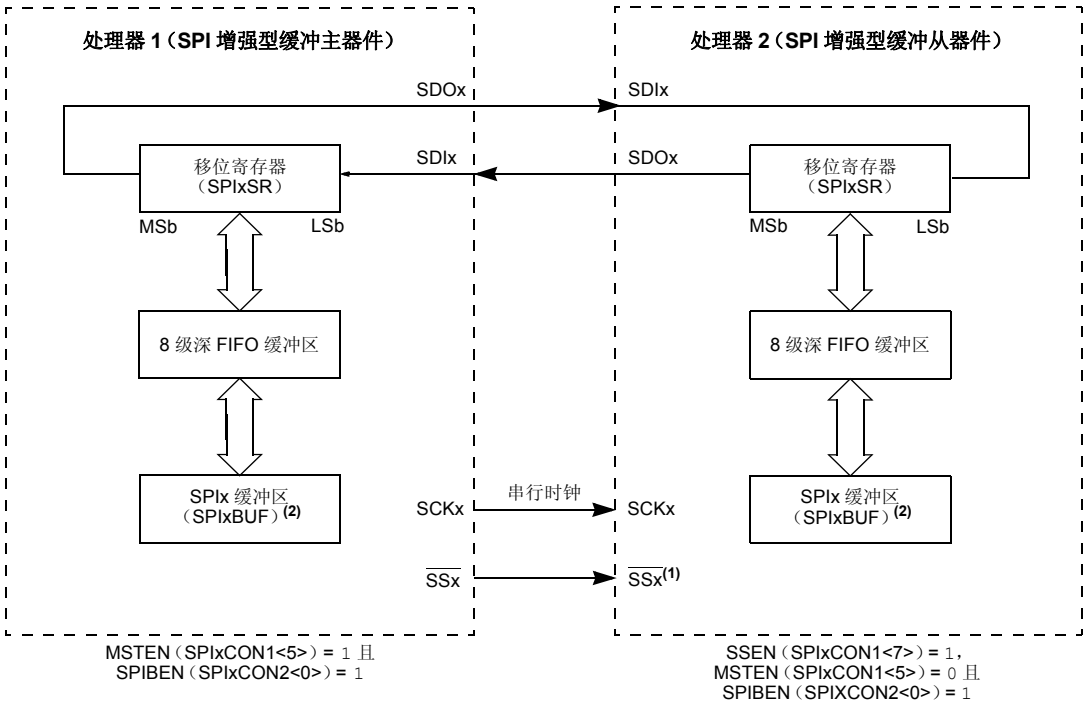
bit 0 **SPIBEN：**增强型缓冲区使能位
1 = 使能增强型缓冲区
0 = 禁止增强型缓冲区 （传统模式）

图 15-3：　　SPI 主 / 从连接 （标准模式）



注　1： 可选择在从模式下使用 $\overline{SSx}$ 引脚。
2： 用户必须向 SPIxBUF 写入发送数据，从 SPIxBUF 读取接收数据。 SPIxTXB 和 SPIxRXB 寄存器被存储器映射到 SPIxBUF。

图 15-4：　　SPI 主 / 从连接 （增强型缓冲模式）



注　1： 可选择在从模式下使用 $\overline{SSx}$ 引脚。
2： 用户必须向 SPIxBUF 写入发送数据，从 SPIxBUF 读取接收数据。SPIxTXB 和 SPIxRXB 寄存器被存储器映射到 SPIxBUF。

图 15-5：　　SPI 主 - 帧主模式连接图

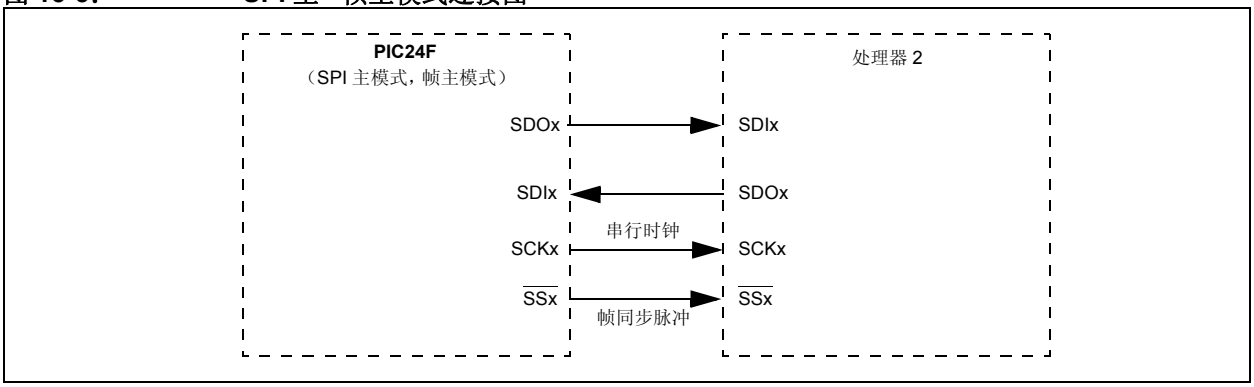


图 15-6：　　SPI 主 - 帧从模式连接图

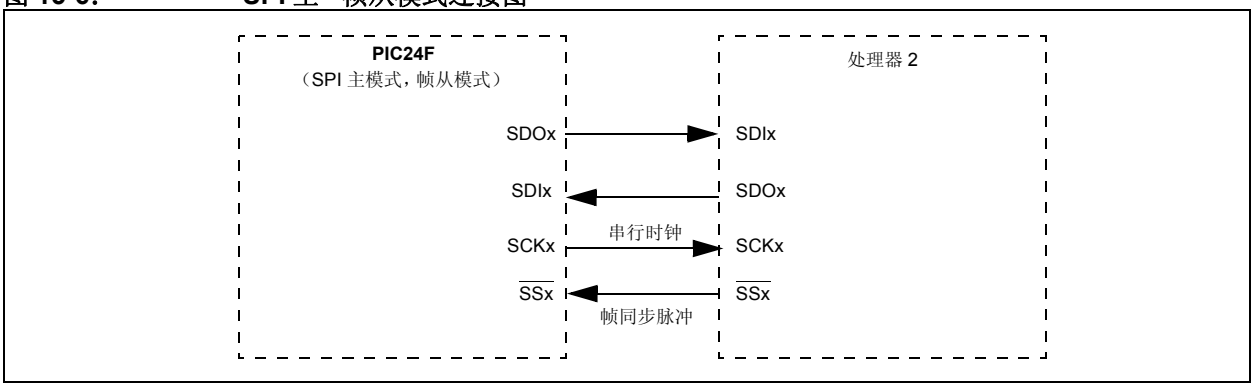


图 15-7：　　SPI 从 - 帧主模式连接图

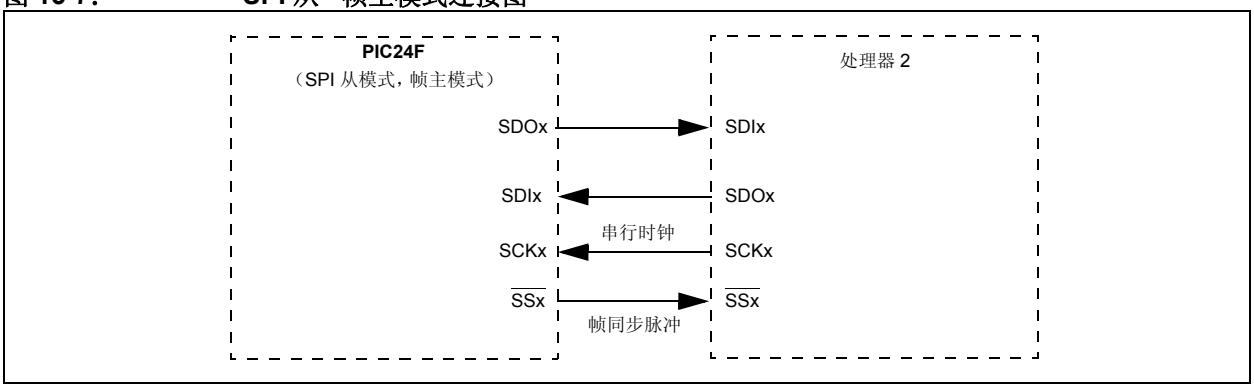
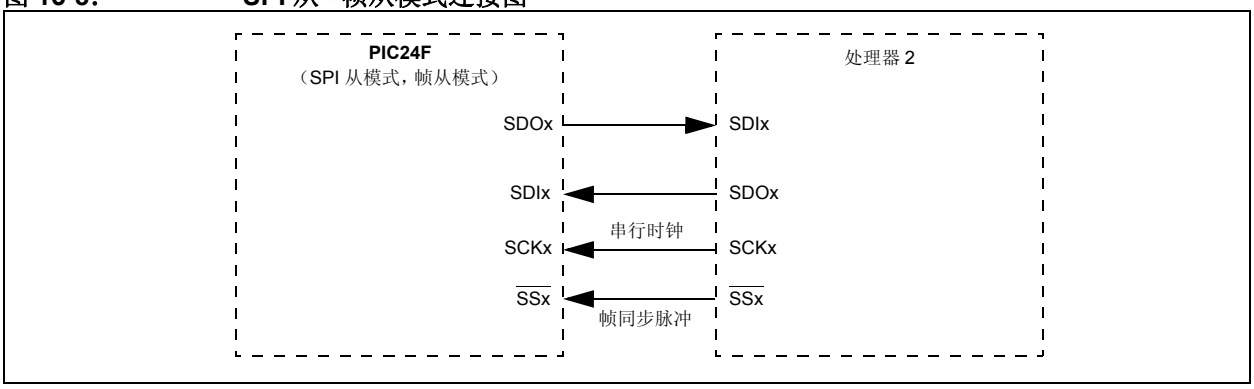


图 15-8：　　SPI 从 - 帧从模式连接图



公式 15-1：　　器件工作频率和 SPI 时钟速度之间的关系 (1)

$$FSCK = \frac{FCY}{\text{主预分频比} * \text{辅助预分频比}}$$

注　1： 基于 FCY = FOSC/2，打盹模式和 PLL 被禁止。

表 15-1：　　示例 SCK 频率 (1,2)

FCY = 16 MHz		**辅助预分频比设置**				
		1:1	**2:1**	**4:1**	**6:1**	**8:1**
主预分频比设置	1:1	无效	8000	4000	2667	2000
	4:1	4000	2000	1000	667	500
	16:1	1000	500	250	167	125
	64:1	250	125	63	42	31
FCY = 5 MHz						
主预分频比设置	1:1	5000	2500	1250	833	625
	4:1	1250	625	313	208	156
	16:1	313	156	78	52	39
	64:1	78	39	20	13	10

注　1： 基于 FCY = FOSC/2，打盹模式和 PLL 被禁止。

2： 表中 SCKx 频率的单位为 kHz。

16.0 I²C™

注： 本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F系列参考手册》的**第24章“I²C™”**（DS39702A_CN）。

I²C 模块是用于同其他外设或单片机器件进行通信的串行接口。这些外设可以是串行 EEPROM、显示驱动器和 A/D 转换器等。

I²C 模块支持以下特性：

- 主器件和从器件逻辑相互独立
- 7 位和 10 位器件地址
- I²C 协议中所定义的广播呼叫地址
- 时钟延长功能，为处理器响应从器件数据请求提供延时
- 100 kHz 和 400 kHz 总线规范
- 可配置的地址掩码
- 多主机模式以防仲裁时报文丢失
- 总线转发器模式，允许作为从器件接收所有报文，与地址无关
- 自动 SCL

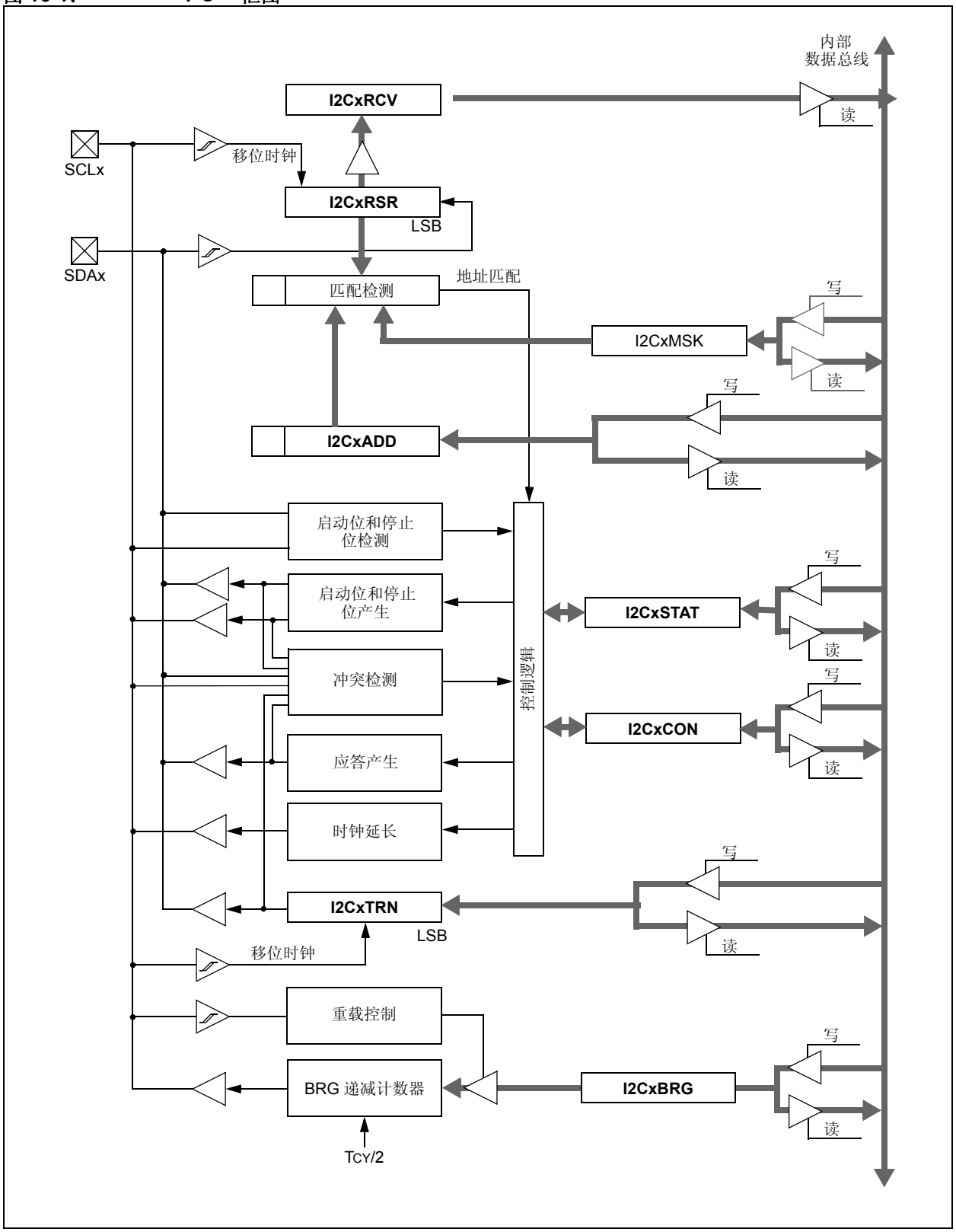
图 16-1 给出了该模块的框图。

16.1 在单主机环境下作为主器件进行通信

在主模式下发送报文的详细信息取决于用于与器件通信的通信协议。通常，该事件的序列如下所示：

1. 在 SDAx 和 SCLx 上发出一个启动条件。
2. 向从器件发送一个带写指示的I²C器件地址字节。
3. 等待并验证来自从器件的应答。
4. 向从器件发送第一个数据字节（有时称为命令）。
5. 等待并验证来自从器件的应答。
6. 向从器件发送串行存储器地址低字节。
7. 重复步骤 4 和 5，直到发送完所有数据字节。
8. 在 SDAx 和 SCLx 上发出一个重复启动条件。
9. 向从器件发送一个带读指示的器件地址字节。
10. 等待并验证来自从器件的应答。
11. 使能主器件接收，以接收串行存储器数据。
12. 在数据字节接收结束时产生ACK或NACK条件。
13. 在 SDAx 和 SCLx 上产生一个停止条件。

图 16-1：　　I²C™ 框图



16.2 设置作为总线主器件工作时的波特率

要计算波特率发生器（Baud Rate Generator， BRG）的重载值，可使用公式 16-1。

公式 16-1： 计算波特率重载值 (1,2)

$$FSCL = \frac{FCY}{I2CxBRG + 1 + \frac{FCY}{10,000,000}}$$

或

$$I2CxBRG = \left(\frac{FCY}{FSCL} - \frac{FCY}{10,000,000}\right) - 1$$

注 1： 基于 FCY = FOSC/2，打盹模式和 PLL 被禁止。

2： 这些时钟频率仅供参考。实际的时钟频率由多个系统级参数决定。应该在目标应用中测量实际的时钟频率。

16.3 从地址掩码

I2CxMSK 寄存器（寄存器 16-3）将 7 位和 10 位寻址模式下地址中的某些位指定为“无关位”。将 I2CxMSK 寄存器中某个特定位置 1（= 1），不论相应的地址位的值是 0 还是 1，工作在从模式下的模块都会作出响应。例如，当将 I2CxMSK 设置为 00100000 时，工作在从模式下的模块将检测两个地址 0000000 和 0100000。

要使能地址掩码，必须通过将IPMIEN位（I2CxCON<11>）清零来禁止智能外设管理接口（Intelligent Peripheral Management Interface， IPMI）。

注： 新修改的 I²C™ 协议使得表 16-2 中的地址保留，而且在从模式下不会应答。这包括包含任何这些地址的任何地址掩码设置。

表 16-1： I²C™ 时钟速率 (1,2)

必需的系统 FSCL	FCY	I2CxBRG 值		实际 FSCL
		（十进制）	（十六进制）	
100 kHz	16 MHz	157	9D	100 kHz
100 kHz	8 MHz	78	4E	100 kHz
100 kHz	4 MHz	39	27	99 kHz
400 kHz	16 MHz	37	25	404 kHz
400 kHz	8 MHz	18	12	404 kHz
400 kHz	4 MHz	9	9	385 kHz
400 kHz	2 MHz	4	4	385 kHz
1 MHz	16 MHz	13	D	1.026 MHz
1 MHz	8 MHz	6	6	1.026 MHz
1 MHz	4 MHz	3	3	0.909 MHz

注 1： 基于 FCY = FOSC/2，打盹模式和 PLL 被禁止。

2： 这些时钟频率仅供参考。实际的时钟频率由多个系统级参数决定。应该在目标应用中测量实际的时钟频率。

表 16-2： I²C™ 保留的地址 (1)

从器件地址	R/$\overline{W}$ 位	说明
0000 000	0	广播呼叫地址 (2)
0000 000	1	启动字节
0000 001	x	Cbus 地址
0000 010	x	保留
0000 011	x	保留
0000 1xx	x	HS 模式主机码
1111 1xx	x	保留
1111 0xx	x	10 位从地址高字节 (3)

注 1： 这里所列的地址位将永远不会导致地址匹配，与地址掩码设置无关。

2： 仅当 GCEN = 1 时才会应答地址。

3： 只有 10 位寻址模式下的高字节才会与该地址发生匹配。

寄存器 16-1：　　I2CxCON：I2Cx 控制寄存器

R/W-0	U-0	R/W-0	R/W-1, HC	R/W-0	R/W-0	R/W-0	R/W-0
I2CEN	—	I2CSIDL	SCLREL	IPMIEN	A10M	DISSLW	SMEN
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0, HC	R/W-0, HC	R/W-0, HC	R/W-0, HC	R/W-0, HC
GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

图注：	HC = 硬件清零位		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15　**I2CEN**：I2Cx 使能位
1 = 使能 I2Cx 模块，并将 SDAx 和 SCLx 引脚配置为串口引脚
0 = 禁止 I2Cx 模块。所有 I^2C 引脚由端口功能控制。

bit 14　**未实现**：读为 0

bit 13　**I2CSIDL**：空闲模式停止位
1 = 当器件进入空闲模式时，模块停止工作
0 = 在空闲模式下模块继续工作

bit 12　**SCLREL**：SCLx 释放控制位 （作为 I^2C 从器件工作时）
1 = 释放 SCLx 时钟
0 = 保持 SCLx 时钟为低电平 （时钟延长）
如果 STREN = 1：
该位可读可写（即软件可以写入 0 来启动时钟延长或写入 1 来释放时钟）。在从器件发送开始时由硬件清零。在从器件接收结束时由硬件清零。
如果 STREN = 0：
该位可读且可被置 1 （即软件只能写入 1 来释放时钟）。在从器件发送开始时由硬件清零。

bit 11　**IPMIEN**：智能平台管理接口 （IPMI）使能位
1 = 使能 IPMI 支持模式；应答所有地址
0 = 禁止 IPMI 模式

bit 10　**A10M**：10 位从器件寻址位
1 = I2CxADD 为 10 位从器件地址
0 = I2CxADD 为 7 位从器件地址

bit 9　**DISSLW**：禁止压摆率控制位
1 = 禁止压摆率控制
0 = 使能压摆率控制

bit 8　**SMEN**：SMBus 输入电平位
1 = 使能符合 SMBus 规范的 I/O 引脚门限值
0 = 禁止 SMBus 输入门限值

bit 7　**GCEN**：广播呼叫使能位 （作为 I^2C 从器件工作时）
1 = 允许在 I2CxRSR 接收到广播呼叫地址时产生中断 （已使能模块接收）
0 = 禁止广播呼叫地址

bit 6　**STREN**：SCLx 时钟延长使能位 （作为 I^2C 从器件工作时）
与 SCLREL 位配合使用。
1 = 使能软件或接收时钟延长
0 = 禁止软件或接收时钟延长

寄存器 16-1： I2CxCON：I2Cx 控制寄存器 （续）

bit 5 **ACKDT**：应答数据位 （作为 I²C 主器件工作时。适用于主器件接收过程。）
当软件启动应答序列时将发送的值。
1 = 在应答时发送 NACK
0 = 在应答时发送 ACK

bit 4 **ACKEN**：应答序列使能位 （作为 I²C 主器件工作时。适用于主器件接收过程。）
1 = 在 SDAx 和 SCLx 引脚上发出应答序列，并发送 ACKDT 数据位。在主器件应答序列结束时由硬件清零。
0 = 应答序列不在进行中

bit 3 **RCEN**：接收使能位 （作为 I²C 主器件工作时）
1 = 使能 I²C 接收模式。在主器件接收完数据字节的第 8 位时由硬件清零。
0 = 接收序列不在进行中

bit 2 **PEN**：停止条件使能位 （作为 I²C 主器件工作时）
1 = 在 SDAx 和 SCLx 引脚上发出停止条件。在主器件停止序列结束时由硬件清零。
0 = 停止条件不在进行中

bit 1 **RSEN**：重复启动条件使能位 （作为 I²C 主器件工作时）
1 = 在 SDAx 和 SCLx 引脚上发出重复启动条件。在主器件重复启动序列结束时由硬件清零。
0 = 重复启动条件不在进行中

bit 0 **SEN**：启动条件使能位 （作为 I²C 主器件工作时）
1 = 在 SDAx 和 SCLx 引脚上发出启动条件。在主器件启动序列结束时由硬件清零。
0 = 启动条件不在进行中

寄存器 16-2： I2CxSTAT：I2Cx 状态寄存器

R-0, HSC	R-0, HSC	U-0	U-0	U-0	R/C-0, HS	R-0, HSC	R-0, HSC
ACKSTAT	TRSTAT	—	—	—	BCL	GCSTAT	ADD10
bit 15							bit 8

R/C-0, HS	R/C-0, HS	R-0, HSC	R/C-0, HSC	R/C-0, HSC	R-0, HSC	R-0, HSC	R-0, HSC
IWCOL	I2COV	D/$\overline{A}$	P	S	R/$\overline{W}$	RBF	TBF
bit 7							bit 0

图注：	C = 可清零位	HS = 硬件置 1 位	HSC = 硬件置 1/ 清零位
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15 **ACKSTAT**：应答状态位
1 = 最后检测到 NACK
0 = 最后检测到 ACK
在应答结束时由硬件置 1 或清零。

bit 14 **TRSTAT**：发送状态位 （作为 I^2C 主器件工作时。适用于主器件发送操作。）
1 = 主器件正在进行发送 （8 位 + ACK）
0 = 主器件不在进行发送
在主器件发送开始时由硬件置 1。在从器件应答结束时由硬件清零。

bit 13-11 **未实现**：读为 0

bit 10 **BCL**：主器件总线冲突检测位
1 = 主器件工作期间检测到了总线冲突
0 = 未发生冲突
在检测到总线冲突时由硬件置 1。

bit 9 **GCSTAT**：广播呼叫状态位
1 = 接收到广播呼叫地址
0 = 未接收到广播呼叫地址
当地址与广播呼叫地址匹配时由硬件置 1。在检测到停止条件时由硬件清零。

bit 8 **ADD10**：10 位地址状态位
1 = 10 位地址匹配
0 = 10 位地址不匹配
在与匹配的 10 位地址的第 2 个字节匹配时由硬件置 1。在检测到停止条件时由硬件清零。

bit 7 **IWCOL**：写冲突检测位
1 = 因为 I^2C 模块忙，尝试写 I2CxTRN 寄存器失败
0 = 未发生冲突
当总线忙时写 I2CxTRN 会使硬件将该位置 1 （由软件清零）。

bit 6 **I2COV**：接收溢出标志位
1 = 当 I2CxRCV 寄存器仍然保存原先的字节时接收到了新字节
0 = 未溢出
尝试将数据从 I2CxRSR 传输到 I2CxRCV 时由硬件置 1 （由软件清零）。

bit 5 **D/$\overline{A}$**：数据 / $\overline{\text{地址}}$位 （作为 I^2C 从器件工作时）
1 = 表示上次接收的字节为数据
0 = 表示上次接收的字节为器件地址
在器件地址匹配时由硬件清零。传输完成或接收到从器件字节后由硬件置 1。

寄存器 16-2： I2CxSTAT：I2Cx 状态寄存器 （续）

bit 4 **P：**停止位
1 = 表示上次检测到停止位
0 = 上次未检测到停止位
当检测到启动、重复启动或停止条件时由硬件置 1 或清零。

bit 3 **S：**启动位
1 = 表示上次检测到启动 （或重复启动）位
0 = 上次未检测到启动位
当检测到启动、重复启动或停止条件时由硬件置 1 或清零。

bit 2 **R/$\overline{W}$：**读 / $\overline{写}$信息位 （作为 I^2C 从器件工作时）
1 = 读——表示数据自从器件输出
0 = 写——表示数据输入到从器件
接收到 I^2C 器件地址字节后由硬件置 1 或清零。

bit 1 **RBF：**接收缓冲区满状态位
1 = 接收完成， I2CxRCV 为满
0 = 接收未完成， I2CxRCV 为空
用接收到的字节写 I2CxRCV 时由硬件置 1。用软件读 I2CxRCV 时由硬件清零。

bit 0 **TBF：**发送缓冲区满状态位
1 = 发送正在进行中， I2CxTRN 为满
0 = 发送完成， I2CxTRN 为空
用软件写 I2CxTRN 时由硬件置 1。数据发送完成时由硬件清零。

寄存器 16-3：　　I2CxMSK：I2Cx 从模式地址掩码寄存器

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	AMSK9	AMSK8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
AMSK7	AMSK6	AMSK5	AMSK4	AMSK3	AMSK2	AMSK1	AMSK0
bit 7							bit 0

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

bit 15-10　**未实现：**读为 0

bit 9-0　**AMSK<9:0>：**地址中 bit x 的掩码选择位

1 = 使能输入报文的地址中 bit x 的掩码；在此位置上不需要位匹配
0 = 禁止 bit x 的掩码；在此位置上需要位匹配

17.0 通用异步收发器 （UART）

注： 本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 21 章“UART”**（DS39708A_CN）。

通用异步收发器 （Universal Asynchronous Receiver Transmitter，UART）模块是 PIC24F 器件系列提供的串行 I/O 模块之一。 UART 是可以与外设 （例如个人计算机、LIN/J2602、RS-232 和 RS-485 接口）通信的全双工异步系统。模块还通过 UxCTS 和 UxRTS 引脚支持硬件流控制选项，其中还包括 IrDA® 编码器和解码器。

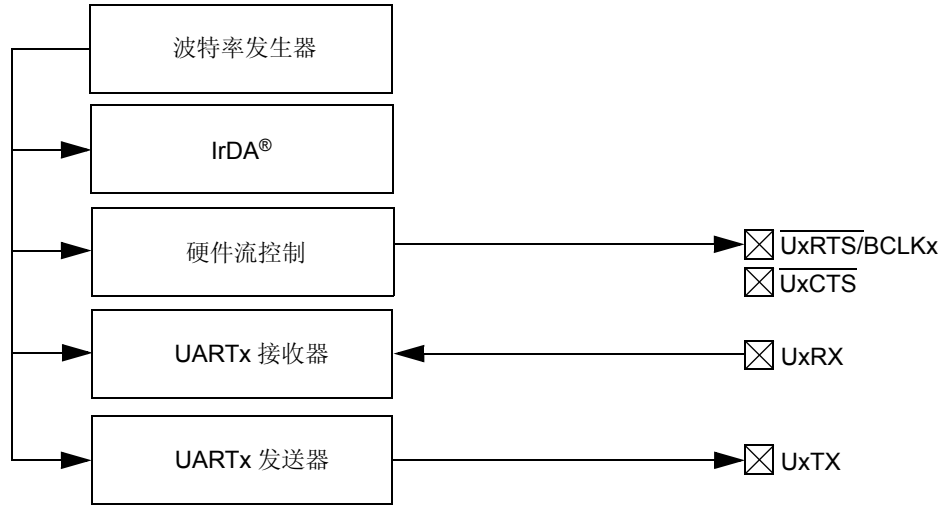
UART 模块的主要特性有：

- 通过UxTX和UxRX引脚进行全双工8位或9位数据传输
- 偶校验、奇校验或无奇偶校验选项（对于 8 位数据）
- 一个或两个停止位
- 通过 UxCTS 和 UxRTS 引脚支持硬件流控制选项
- 完全集成的波特率发生器，具有 16 位预分频器
- 当器件工作在 16 MIPS 时，波特率范围从 1 Mbps 到 15 Mbps
- 4 级深度先进先出 （First-In-First-Out， FIFO）发送数据缓冲区
- 4 级深度 FIFO 接收数据缓冲区
- 奇偶校验、帧和缓冲区溢出错误检测
- 支持带地址检测的 9 位模式 （第 9 位 = 1）
- 发送和接收中断
- 用于诊断支持的环回模式
- 支持同步和间隔字符
- 支持自动波特率检测
- IrDA 编码器和解码器逻辑
- 用于 IrDA 支持的 16 倍波特率时钟输出

图 17-1 给出了 UART 的简化框图。 UART 模块由以下至关重要的硬件组件组成：

- 波特率发生器
- 异步发送器
- 异步接收器

图 17-1： UART 简化框图



注： UART 输入和输出都必须在使用前分配给可用的 RPn 引脚。更多信息，请参见**第 10.4 节“外设引脚选择 （PPS）”**。

17.1 UART 波特率发生器 （BRG）

UART 模块包含一个专用的 16 位波特率发生器。 UxBRG 寄存器控制一个自由运行的16位定时器的周期。公式17-1 给出了 BRGH = 0 时计算波特率的公式。

公式 17-1： UART 波特率 （BRGH = 0） (1,2)

$$\text{波特率} = \frac{FCY}{16 \bullet (UxBRG + 1)}$$

$$UxBRG = \frac{FCY}{16 \bullet \text{波特率}} - 1$$

注 1： FCY 表示指令周期时钟频率 （FOSC/2）。
2： 基于 FCY = FOSC/2，打盹模式和 PLL 被禁止。

例 17-1 给出了如下条件下的波特率误差计算：

- FCY = 4 MHz
- 目标波特率 = 9600

最大可能波特率（BRGH = 0）是FCY/16（当UxBRG = 0 时），最小可能波特率是 FCY/(16 * 65536)。

公式 17-2 给出了 BRGH = 1 时计算波特率的公式。

公式 17-2： UART 波特率 （BRGH = 1） (1,2)

$$\text{波特率} = \frac{FCY}{4 \bullet (UxBRG + 1)}$$

$$UxBRG = \frac{FCY}{4 \bullet \text{波特率}} - 1$$

注 1： FCY 表示指令周期时钟频率。
2： 基于 FCY = FOSC/2，打盹模式和 PLL 被禁止。

最大可能波特率（BRGH = 1）是 FCY/4（当 UxBRG = 0 时），最小可能波特率是 FCY/(4 * 65536)。

向 UxBRG 寄存器写入新值会使 BRG 定时器复位 （清零）。这可以确保BRG无需等待定时器溢出就可以产生新的波特率。

例 17-1： 波特率误差计算 （BRGH = 0） (1)

目标波特率 = FCY/(16 (UxBRG + 1))

求解 UxBRG 值：

UxBRG = ((FCY/ 目标波特率)/16) – 1
UxBRG = ((4000000/9600)/16) – 1
UxBRG = 25

计算波特率 = 4000000/(16 (25 + 1))
= 9615

误差 = (计算波特率 – 目标波特率)/ 目标波特率
= (9615 – 9600)/9600
= 0.16%

注 1： 基于 FCY = FOSC/2，打盹模式和 PLL 被禁止。

17.2 8 位数据模式下的发送

1. 设置 UART：
   a) 将适当的值写入数据位、奇偶校验位和停止位。
   b) 将适当的波特率值写入 UxBRG 寄存器。
   c) 设置发送和接收中断允许位和优先级位。
2. 使能 UART。
3. 将UTXEN位置1（置1后两个周期产生发送中断）。
4. 将数据字节写入 UxTXREG 字的低字节。该值将被立即传输到发送移位寄存器 （Transmit Shift Register，TSR），且在波特率时钟的下一个上升沿开始移出串行比特流。
5. 或者，当 UTXEN = 0 时，数据字节也可被发送，且随后用户可将 UTXEN 置 1。由于波特率时钟将从清零状态启动，这将立即开始发送串行比特流。
6. 中断控制位 UTXISELx 的设置决定何时产生发送中断。

17.3 9 位数据模式下的发送

1. 设置 UART （如**第 17.2 节 “8 位数据模式下的发送”**中所述）。
2. 使能 UART。
3. 将 UTXEN 位置 1 （产生发送中断）。
4. 仅向 UxTXREG 写入一个 16 位的值。
5. 向 UxTXREG 写入一个字可触发 9 位数据向 TSR 的传输。串行比特流将会在波特率时钟的第一个上升沿开始移出。
6. 中断控制位 UTXISELx 的设置决定何时产生发送中断。

17.4 间隔和同步发送序列

以下序列将发送一个报文帧头，包括一个间隔字符和其后的一个自动波特率同步字节。

1. 将 UART 配置为所需的模式。
2. 将 UTXEN 和 UTXBRK 置 1 以设置间隔字符。
3. 将一个无效字符装入 UxTXREG 以启动发送（值被忽略）。
4. 向 UxTXREG 写入 “55h”；将同步字符装入发送 FIFO 中。
5. 间隔字符发送后，硬件会将 UTXBRK 位复位。然后开始发送同步字符。

17.5 8 位或 9 位数据模式下的接收

1. 设置 UART （如**第 17.2 节 “8 位数据模式下的发送”**中所述）。
2. 使能 UART。
3. 当接收到一个或多个数据字符时，将会根据中断控制位 URXISELx 的设置产生接收中断。
4. 读OERR位以确定是否发生了溢出错误。OERR位必须用软件复位。
5. 读 UxRXREG。

读取UxRXREG字符的行为会将下一个字符传送到接收 FIFO 的顶部，其中包含一组新的 PERR 和 FERR 值。

17.6 $\overline{\text{UxCTS}}$ 和 $\overline{\text{UxRTS}}$ 控制引脚的操作

UARTx 允许发送 （$\overline{\text{UxCTS}}$）和请求发送 （$\overline{\text{UxRTS}}$）是两个与 UART 模块相关、由硬件控制的引脚。这两个引脚允许 UART 运行在单工模式和流控制模式下。它们用于控制数据终端设备（Data Terminal Equipment，DTE）之间的发送和接收。UxMODE 寄存器中的 UEN<1:0> 位用来配置这两个引脚。

17.7 红外支持

UART 模块提供两种类型的红外 UART 支持：一种是 IrDA 时钟输出，用于支持外部 IrDA 编码器和解码器（传统模块支持）；另一种是完全实现的IrDA编码器和解码器。注意，由于 IrDA 模式需要 16 倍波特率时钟，它们仅在 BRGH 位 （UxMODE<3>）为 0 时才能工作。

17.7.1 用于外部 IRDA 支持的 IRDA 时钟输出

为了支持外部 IrDA 编码器和解码器，可将 BCLKx 引脚（和 $\overline{\text{UxRTS}}$ 引脚相同）配置为产生 16 倍波特率时钟。当使能了 UART 模块且 UEN<1:0> = 11 时，BCLKx 引脚将输出 16 倍波特率时钟。它可用于支持 IrDA 编解码器芯片。

17.7.2 内置 IRDA 编码器和解码器

UART 模块在其内部完全实现了 IrDA 编码器和解码器。内置 IrDA 编码器和解码器的功能可通过 IREN 位（UxMODE<12>）来使能。当使能（IREN = 1）时，接收引脚 （UxRX）可作为红外接收器的输入引脚。发送引脚 （UxTX）可作为红外发送器的输出引脚。

寄存器 17-1： UxMODE：UARTx 模式寄存器

R/W-0	U-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
UARTEN(1)	—	USIDL	IREN(2)	RTSMD	—	UEN1	UEN0
bit 15							bit 8

R/W-0, HC	R/W-0	R/W-0, HC	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WAKE	LPBACK	ABAUD	RXINV	BRGH	PDSEL1	PDSEL0	STSEL
bit 7							bit 0

图注：	HC = 硬件清零位		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15 **UARTEN**：UARTx 使能位 (1)
1 = 使能 UARTx； UARTx 根据 UEN<1:0> 的定义控制所有 UARTx 引脚
0 = 禁止 UARTx；由端口锁存器控制所有 UARTx 引脚；此时 UARTx 的功耗最小

bit 14 **未实现**：读为 0

bit 13 **USIDL**：空闲模式停止位
1 = 当器件进入空闲模式时，模块停止工作
0 = 在空闲模式下模块继续工作

bit 12 **IREN**：IrDA® 编码器和解码器使能位 (2)
1 = 使能 IrDA 编码器和解码器
0 = 禁止 IrDA 编码器和解码器

bit 11 **RTSMD**：$\overline{\text{UxRTS}}$ 引脚模式选择位
1 = $\overline{\text{UxRTS}}$ 引脚处于单工模式
0 = $\overline{\text{UxRTS}}$ 引脚处于流控制模式

bit 10 **未实现**：读为 0

bit 9-8 **UEN<1:0>**：UARTx 使能位
11 = 使能并使用 UxTX、 UxRX 和 BCLKx 引脚； $\overline{\text{UxCTS}}$ 引脚由端口锁存器控制
10 = 使能并使用 UxTX、 UxRX、 $\overline{\text{UxCTS}}$ 和 $\overline{\text{UxRTS}}$ 引脚
01 = 使能并使用 UxTX、 UxRX 和 $\overline{\text{UxRTS}}$ 引脚； $\overline{\text{UxCTS}}$ 引脚由端口锁存器控制
00 = 使能并使用 UxTX 和 UxRX 引脚； $\overline{\text{UxCTS}}$ 和 $\overline{\text{UxRTS}}$/BCLKx 引脚由端口锁存器控制

bit 7 **WAKE**：在休眠模式下检测到启动位时唤醒的使能位
1 = UARTx 将继续采样 UxRX 引脚；在出现下降沿时产生中断；在出现下一个上升沿时由硬件清零该位
0 = 禁止唤醒

bit 6 **LPBACK**：UARTx 环回模式选择位
1 = 使能环回模式
0 = 禁止环回模式

bit 5 **ABAUD**：自动波特率使能位
1 = 使能对下一个字符的波特率测量——需要接收同步字段 （55h）；完成时由硬件清零
0 = 禁止波特率测量或测量已完成

bit 4 **RXINV**：接收极性翻转位
1 = UxRX 的空闲状态为 0
0 = UxRX 的空闲状态为 1

注 1： 如果 UARTEN = 1，外设输入和输出必须配置给可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。

2： 此功能只能在 16 倍 BRG 模式 （BRGH = 0）下使用。

寄存器 17-1： UxMODE：UARTx 模式寄存器 （续）

bit 3 **BRGH**：高波特率使能位
1 = 高速模式 （每位 4 个 BRG 时钟周期）
0 = 标准模式 （每位 16 个 BRG 时钟周期）

bit 2-1 **PDSEL<1:0>**：奇偶校验和数据选择位
11 = 9 位数据，无奇偶校验
10 = 8 位数据，奇校验
01 = 8 位数据，偶校验
00 = 8 位数据，无奇偶校验

bit 0 **STSEL**：停止位选择位
1 = 2 个停止位
0 = 1 个停止位

注 1： 如果 UARTEN = 1，外设输入和输出必须配置给可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。

2： 此功能只能在 16 倍 BRG 模式 （BRGH = 0）下使用。

寄存器 17-2： UxSTA：UARTx 状态和控制寄存器

R/W-0	R/W-0	R/W-0	U-0	R/W-0, HC	R/W-0	R-0	R-1
UTXISEL1	UTXINV(1)	UTXISEL0	—	UTXBRK	UTXEN(2)	UTXBF	TRMT
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R-1	R-0	R-0	R/C-0	R-0
URXISEL1	URXISEL0	ADDEN	RIDLE	PERR	FERR	OERR	URXDA
bit 7							bit 0

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

bit 15,13 **UTXISEL<1:0>：**发送中断模式选择位
11 = 保留；不要使用
10 = 当一个字符被传输到发送移位寄存器 （TSR）导致发送缓冲区为空时，产生中断
01 = 当最后一个字符被移出发送移位寄存器；所有发送操作执行完毕时产生中断
00 = 当一个字符被传输到发送移位寄存器 （这意味着发送缓冲区中至少还有一个字符）时产生中断

bit 14 **UTXINV：**IrDA® 编码器发送极性翻转位 (1)
IREN = 0：
1 = UxTX 的空闲状态为 0
0 = UxTX 的空闲状态为 1
IREN = 1：
1 = UxTX 的空闲状态为 1
0 = UxTX 的空闲状态为 0

bit 12 **未实现：**读为 0

bit 11 **UTXBRK：**发送间隔位
1 = 在下次发送时发出同步间隔字符——启动位，后跟 12 个 0 位，然后是停止位；完成时由硬件清零
0 = 禁止或已完成同步间隔字符的发送

bit 10 **UTXEN：**发送使能位 (2)
1 = 使能发送， UARTx 控制 UxTX 引脚
0 = 禁止发送，中止所有等待的发送，缓冲区被复位；由端口控制 UxTX 引脚

bit 9 **UTXBF：**发送缓冲区满状态位 （只读）
1 = 发送缓冲区满
0 = 发送缓冲区未满；至少还可写入一个或多个字符

bit 8 **TRMT：**发送移位寄存器空位 （只读）
1 = 发送移位寄存器为空，同时发送缓冲区为空 （上一次发送已完成）
0 = 发送移位寄存器非空，发送在进行中或在发送缓冲区中排队

bit 7-6 **URXISEL<1:0>：**接收中断模式选择位
11 = 当 RSR 传输使接收缓冲区为满时 （即有 4 个数据字符），中断标志位置 1
10 = 当 RSR 传输使接收缓冲区 3/4 满时 （即有 3 个数据字符），中断标志位置 1
0x = 当接收到任何一个字符且将字符从 RSR 传输到接收缓冲区时，中断标志位置 1 ；接收缓冲区中有一个或多个字符

注 1： 仅当使能了 IrDA 编码器 （IREN = 1）时，该位的值才影响模块的发送属性。
2： 如果 UARTEN = 1，外设输入和输出必须配置给可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。

寄存器 17-2： UxSTA：UARTx 状态和控制寄存器 （续）

bit 5 **ADDEN**：地址字符检测位 （接收数据的 bit 8 = 1）
1 = 使能地址检测模式。如果没有选择 9 位模式，这个控制位将无效。
0 = 禁止地址检测模式

bit 4 **RIDLE**：接收器空闲位 （只读）
1 = 接收器空闲
0 = 接收器工作

bit 3 **PERR**：奇偶校验错误状态位 （只读）
1 = 检测到当前字符 （接收 FIFO 顶部的字符）的奇偶校验错误
0 = 未检测到奇偶校验错误

bit 2 **FERR**：帧错误状态位 （只读）
1 = 检测到当前字符 （接收 FIFO 顶部的字符）的帧错误
0 = 未检测到帧错误

bit 1 **OERR**：接收缓冲区溢出错误状态位 （清零 / 只读）
1 = 接收缓冲区已溢出
0 = 接收缓冲器未溢出。清零原来置 1 的 OERR 位（1 → 0 的跳变）将使接收缓冲区和 RSR 复位为空状态

bit 0 **URXDA**：接收缓冲区中是否有数据标志位 （只读）
1 = 接收缓冲区中有数据，有至少一个或多个字符可被读取
0 = 接收缓冲区为空

注 1： 仅当使能了 IrDA 编码器 （IREN = 1）时，该位的值才影响模块的发送属性。

2： 如果 UARTEN = 1，外设输入和输出必须配置给可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。

注：

18.0 并行主端口（PMP）

注： 本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 13 章“并行主端口（PMP）”**（DS39713A_CN）。

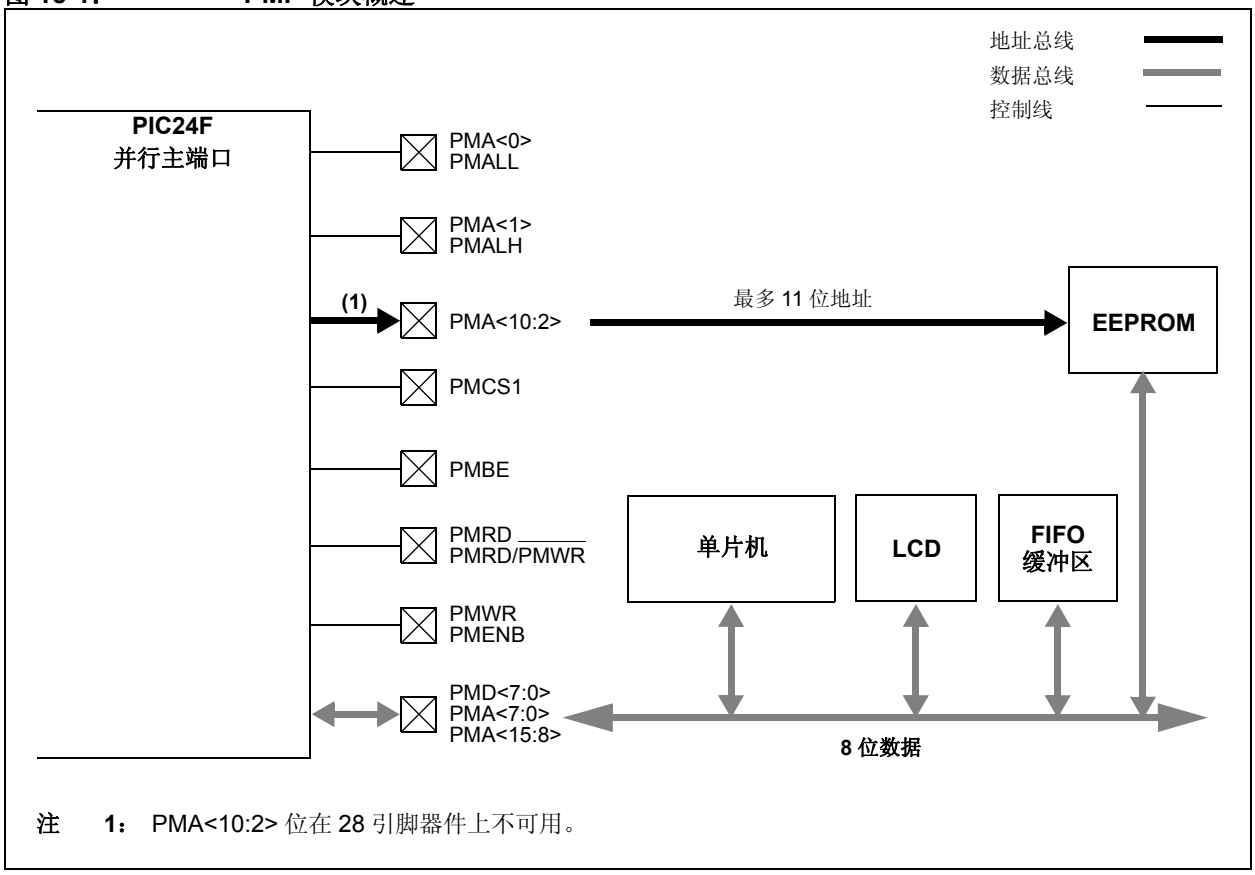
并行主端口（PMP）模块是专为与多种并行器件，如通信外设、LCD、外部存储器和单片机等通信而设计的 8 位并行 I/O 模块。由于并行外设的接口差异很大，因此 PMP 具有高度可配置性。

注： PMP 的一些引脚在 PIC24FJ64GA1 系列器件上不存在。请参见具体器件的引脚排列来确定可用的引脚。

PMP 模块的主要特性包括：

- 最多 16 条可编程地址线
- 1 条片选线
- 可编程选通选项：
  - 独立的读和写选通，或；
  - 带使能选通的读 / $\overline{\text{写}}$选通
- 地址自动递增 / 自动递减
- 可编程地址 / 数据复用
- 可编程控制信号的极性
- 支持传统并行从端口
- 支持增强型并行从端口：
  - 地址支持
  - 4 字节深的自动递增缓冲区
- 可编程等待状态
- 可选择的输入电平

图 18-1： PMP 模块概述



注 1： PMA<10:2> 位在 28 引脚器件上不可用。

寄存器 18-1：　PMCON：并行端口控制寄存器

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PMPEN	—	PSIDL	ADRMUX1(1)	ADRMUX0(1)	PTBEEN	PTWREN	PTRDEN
bit 15							bit 8

R/W-0	R/W-0	R/W-0(2)	U-0	R/W-0(2)	R/W-0	R/W-0	R/W-0
CSF1	CSF0	ALP	—	CS1P	BEP	WRSP	RDSP
bit 7							bit 0

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

bit 15　**PMPEN**：并行主端口使能位
1 = 使能 PMP
0 = 禁止 PMP，不执行任何片外访问

bit 14　**未实现**：读为 0

bit 13　**PSIDL**：空闲模式停止位
1 = 当器件进入空闲模式时，模块停止工作
0 = 在空闲模式下模块继续工作

bit 12-11　**ADRMUX<1:0>**：地址 / 数据复用选择位 (1)
11 = 保留
10 = 地址的所有 16 位在 PMD<7:0> 引脚上复用
01 = 地址的低 8 位在 PMD<7:0> 引脚上复用；高 3 位在 PMA<10:8> 上复用
00 = 地址和数据使用独立的引脚

bit 10　**PTBEEN**：字节使能端口使能位 （16 位主模式）
1 = 使能 PMBE 端口
0 = 禁止 PMBE 端口

bit 9　**PTWREN**：写使能选通端口使能位
1 = 使能 PMWR/PMENB 端口
0 = 禁止 PMWR/PMENB 端口

bit 8　**PTRDEN**：读 / 写选通端口使能位
1 = 使能 PMRD/$\overline{\text{PMWR}}$ 端口
0 = 禁止 PMRD/PMWR 端口

bit 7-6　**CSF<1:0>**：片选功能位
11 = 保留
10 = PMCS1 用作片选
01 = 保留
00 = 保留

bit 5　**ALP**：地址锁存器极性位 (2)
1 = 高电平有效 （$\overline{\text{PMALL}}$ 和 $\overline{\text{PMALH}}$）
0 = 低电平有效 （PMALL 和 PMALH）

bit 4　**未实现**：读为 0

bit 3　**CS1P**：片选 1 极性位 (2)
1 = 高电平有效 （$\overline{\text{PMCS1}}$/PMCS1）
0 = 低电平有效 （PMCS1/$\overline{\text{PMCS1}}$）

注　1：PMA<10:2> 位在 28 引脚器件上不可用。
2：这些位在相应引脚用作地址线时无效。

寄存器 18-1： PMCON：并行端口控制寄存器 （续）

bit 2 **BEP**：字节使能极性位
1 = 字节使能高电平有效 （PMBE）
0 = 字节使能低电平有效 （PMBE）

bit 1 **WRSP**：写选通极性位
对于从模式和主模式 2 （PMMODE<9:8> = 00、 01 和 10）：
1 = 写选通高电平有效 （PMWR）
0 = 写选通低电平有效 （PMWR）
对于主模式 1 （PMMODE<9:8> = 11）：
1 = 使能选通高电平有效 （PMENB）
0 = 使能选通低电平有效 （PMENB）

bit 0 **RDSP**：读选通极性位
对于从模式和主模式 2 （PMMODE<9:8> = 00、 01 和 10）：
1 = 读选通高电平有效 （PMRD）
0 = 读选通低电平有效 （PMRD）
对于主模式 1 （PMMODE<9:8> = 11）：
1 = 读 / 写选通高电平有效 （PMRD/PMWR）
0 = 读 / 写选通低电平有效 （PMRD/PMWR）

注 1： PMA<10:2> 位在 28 引脚器件上不可用。
2： 这些位在相应引脚用作地址线时无效。

寄存器 18-2：　　PMMODE：并行端口模式寄存器

R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUSY	IRQM1	IRQM0	INCM1	INCM0	MODE16	MODE1	MODE0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WAITB1(1)	WAITB0(1)	WAITM3	WAITM2	WAITM1	WAITM0	WAITE1(1)	WAITE0(1)
bit 7							bit 0

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

bit 15　**BUSY**：忙位 （仅适用于主模式）
1 = 端口忙 （当处理器停止工作时无用）
0 = 端口不忙

bit 14-13　**IRQM<1:0>**：中断请求模式位
11 = 当读取读缓冲区 3 或写入写缓冲区 3 时产生中断 （缓冲 PSP 模式），
　　或在 PMA<1:0> = 11 时执行读或写操作时产生中断 （仅适用于可寻址 PSP 模式）
10 = 不产生中断，处理器停止工作
01 = 在读 / 写周期结束时产生中断
00 = 不产生中断

bit 12-11　**INCM<1:0>**：递增模式位
11 = PSP 读和写缓冲区自动递增 （仅适用于传统 PSP 模式）
10 = 每个读 / 写周期 ADDR<10:0> 递减 1
01 = 每个读 / 写周期 ADDR<10:0> 递增 1
00 = 无地址递增或递减

bit 10　**MODE16**：8/16 位模式位
1 = 16 位模式：数据寄存器为 16 位；对数据寄存器执行读或写操作将调用两次 8 位传输
0 = 8 位模式：数据寄存器为 8 位；对数据寄存器执行读或写操作将调用一次 8 位传输

bit 9-8　**MODE<1:0>**：并行端口模式选择位
11 = 主模式 1 （PMCS1、 PMRD/$\overline{\text{PMWR}}$、 PMENB、 PMBE、 PMA<x:0> 和 PMD<7:0>）
10 = 主模式 2 （PMCS1、 $\overline{\text{PMRD}}$、 $\overline{\text{PMWR}}$、 $\overline{\text{PMBE}}$、 PMA<x:0> 和 PMD<7:0>）
01 = 增强型 PSP 控制信号 （$\overline{\text{PMRD}}$、 $\overline{\text{PMWR}}$、 $\overline{\text{PMCS1}}$、 PMD<7:0> 和 PMA<1:0>）
00 = 传统并行从端口控制信号 （$\overline{\text{PMRD}}$、 $\overline{\text{PMWR}}$、 $\overline{\text{PMCS1}}$ 和 PMD<7:0>）

bit 7-6　**WAITB<1:0>**：从数据建立到执行读 / 写的等待状态配置位 (1)
11 = 数据等待时间为 4 个 TCY ； 地址复用时间为 4 个 TCY
10 = 数据等待时间为 3 个 TCY ； 地址复用时间为 3 个 TCY
01 = 数据等待时间为 2 个 TCY ； 地址复用时间为 2 个 TCY
00 = 数据等待时间为 1 个 TCY ； 地址复用时间为 1 个 TCY

bit 5-2　**WAITM<3:0>**：从执行读操作到字节使能选通的等待状态配置位
1111 = 额外等待 15 个 TCY
...
0001 = 额外等待 1 个 TCY
0000 = 无额外等待周期 （强制操作在 1 个 TCY 内完成）

bit 1-0　**WAITE<1:0>**：选通后数据保持的等待状态配置位 (1)
11 = 等待 4 个 TCY
10 = 等待 3 个 TCY
01 = 等待 2 个 TCY
00 = 等待 1 个 TCY

注　1： 只要 WAITM<3:0> = 0000， WAITB 和 WAITE 位就可被忽略。

寄存器 18-3：　PMADDR：并行端口地址寄存器

U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	CS1	—	—	—	ADDR10(1)	ADDR9(1)	ADDR8(1)
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADDR7(1)	ADDR6(1)	ADDR5(1)	ADDR4(1)	ADDR3(1)	ADDR2(1)	ADDR1(1)	ADDR0(1)
bit 7							bit 0

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

bit 15　**未实现：**读为 0

bit 14　**CS1：**片选 1 位
1 = 片选 1 有效
0 = 片选 1 无效

bit 13-11　**未实现：**读为 0

bit 10-0　**ADDR<10:0>：**并行端口目标地址位 (1)

注　1： PMA<10:2> 位在 28 引脚器件上不可用。

寄存器 18-4：　PMAEN：并行端口使能寄存器

U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	PTEN14	—	—	—	PTEN10(1)	PTEN9(1)	PTEN8(1)
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PTEN7(1)	PTEN6(1)	PTEN5(1)	PTEN4(1)	PTEN3(1)	PTEN2(1)	PTEN1	PTEN0
bit 7							bit 0

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

bit 15　**未实现：**读为 0

bit 14　**PTEN14：**PMCS1 选通使能位
1 = PMCS1 用作片选
0 = PMCS1 引脚用作端口 I/O

bit 13-11　**未实现：**读为 0

bit 10-2　**PTEN<10:2>：**PMP 地址端口使能位 (1)
1 = PMA<10:2> 用作 PMP 地址线
0 = PMA<10:2> 用作端口 I/O

bit 1-0　**PTEN<1:0>：**PMALH/PMALL 选通使能位
1 = PMA1 和 PMA0 用作 PMA<1:0> 或 PMALH 和 PMALL
0 = PMA1 和 PMA0 引脚用作端口 I/O

注　1： PMA<10:2> 位在 28 引脚器件上不可用。

寄存器 18-5：　　PMSTAT：并行端口状态寄存器

R-0	R/W-0, HS	U-0	U-0	R-0	R-0	R-0	R-0
IBF	IBOV	—	—	IB3F	IB2F	IB1F	IB0F
bit 15							bit 8

R-1	R/W-0, HS	U-0	U-0	R-1	R-1	R-1	R-1
OBE	OBUF	—	—	OB3E	OB2E	OB1E	OB0E
bit 7							bit 0

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

bit 15　**IBF**：输入缓冲区满状态位
1 = 所有可写的输入缓冲寄存器均已满
0 = 部分或所有可写的输入缓冲寄存器为空

bit 14　**IBOV**：输入缓冲区溢出状态位
1 = 尝试对已满的输入字节寄存器执行写操作 （必须用软件清零）
0 = 未发生溢出

bit 13-12　**未实现**：读为 0

bit 11-8　**IB3F:IB0F**：输入缓冲区 x 状态满位
1 = 输入缓冲区包含尚未读取的数据 （读缓冲区将清零该位）
0 = 输入缓冲区不包含任何未读数据

bit 7　**OBE**：输出缓冲区空状态位
1 = 所有可读的输出缓冲寄存器均为空
0 = 部分或所有可读的输出缓冲寄存器已满

bit 6　**OBUF**：输出缓冲区下溢状态位
1 = 对空输出字节寄存器执行读操作 （必须用软件清零）
0 = 未发生下溢

bit 5-4　**未实现**：读为 0

bit 3-0　**OB3E:OB0E**：输出缓冲区 x 空状态位
1 = 输出缓冲区为空 （向缓冲区写数据将清零该位）
0 = 输出缓冲区包含尚未发送的数据

寄存器 18-6：　PADCFG1：焊盘配置控制寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	RTSECSEL1(1)	RTSECSEL0(1)	PMPTTL
bit 7							bit 0

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

bit 15-3　**未实现：**读为 0

bit 2-1　**RTSECSEL<1:0>：**RTCC 秒时钟输出选择位 (1)

11 = 保留；不要使用
10 = 选择 RTCC 引脚输出 RTCC 源时钟 （时钟可以是 LPRC 或 SOSC，取决于闪存配置位 RTCOSC 位 （CW4<5>）的设置）
01 = 选择 RTCC 引脚输出 RTCC 秒时钟
00 = 选择 RTCC 引脚输出 RTCC 闹钟脉冲

bit 0　**PMPTTL：**PMP 模块 TTL 输入缓冲器选择位

1 = PMP 模块使用 TTL 输入缓冲器
0 = PMP 模块使用施密特触发器输入缓冲器

注　1： 要使能实际 RTCC 输出，需要将 RTCOE （RCFGCAL<10>）位置 1。

图 18-2：　　传统并行从端口示例

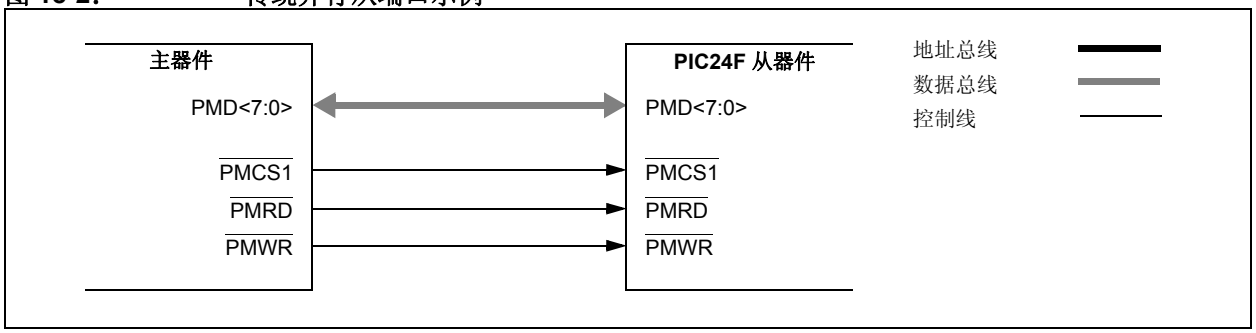


图 18-3：　　可寻址的并行从端口示例

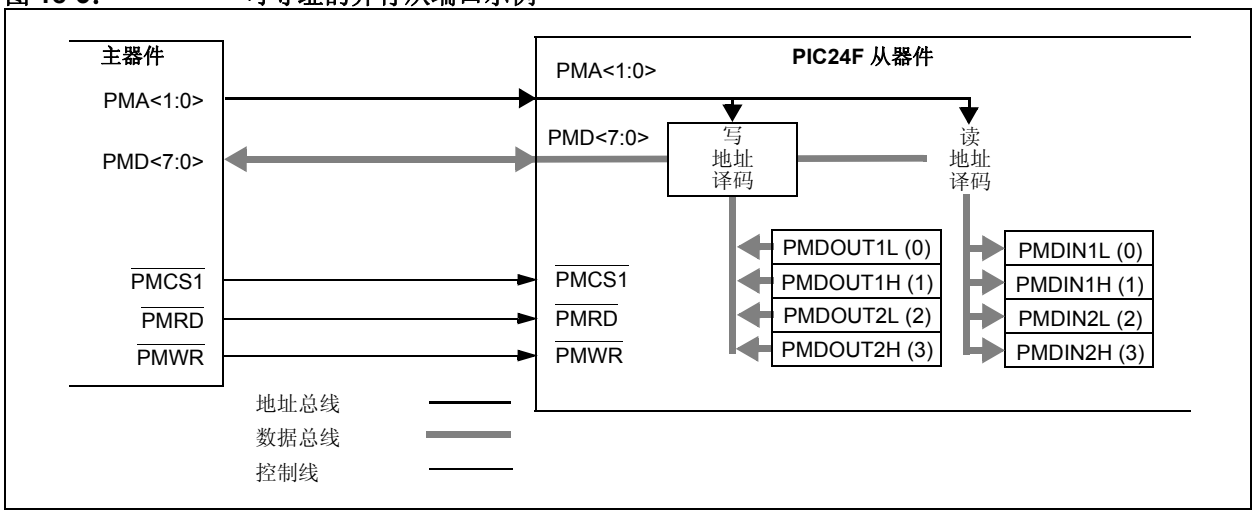


表 18-1：　　从模式地址解析

PMA<1:0>	输出寄存器 （缓冲区）	输入寄存器 （缓冲区）
00	PMDOUT1<7:0> (0)	PMDIN1<7:0> (0)
01	PMDOUT1<15:8> (1)	PMDIN1<15:8> (1)
10	PMDOUT2<7:0> (2)	PMDIN2<7:0> (2)
11	PMDOUT2<15:8> (3)	PMDIN2<15:8> (3)

图 18-4：　　主模式，解复用的寻址 （独立的读和写选通，单片选）

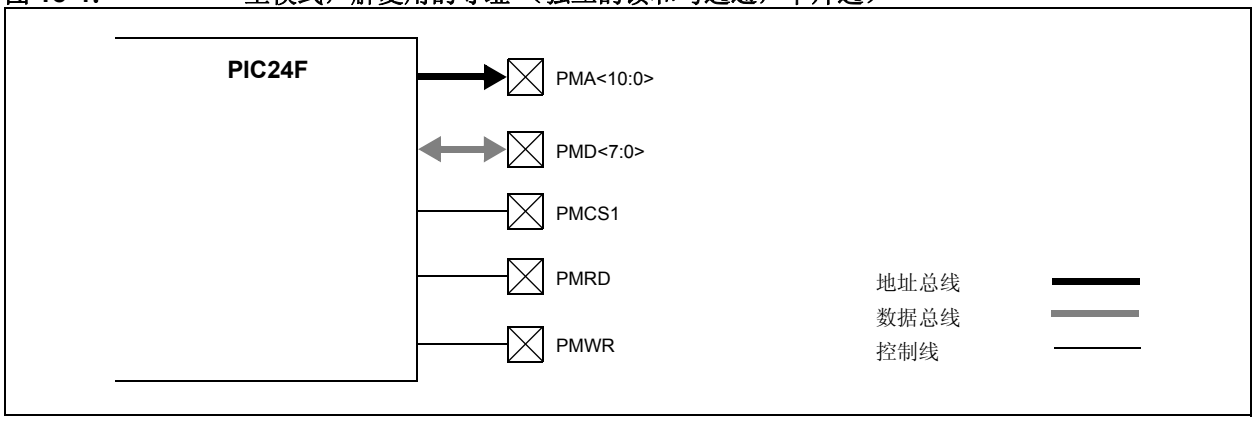


图 18-5：　　主模式，部分复用的寻址 （独立的读和写选通，单片选）

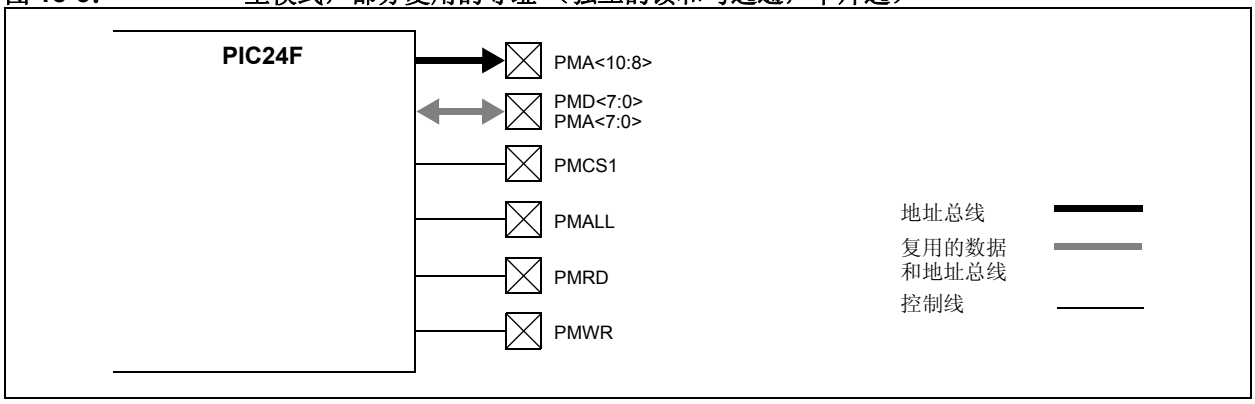


图 18-6：　　主模式，完全复用的寻址 （独立的读和写选通，单片选）

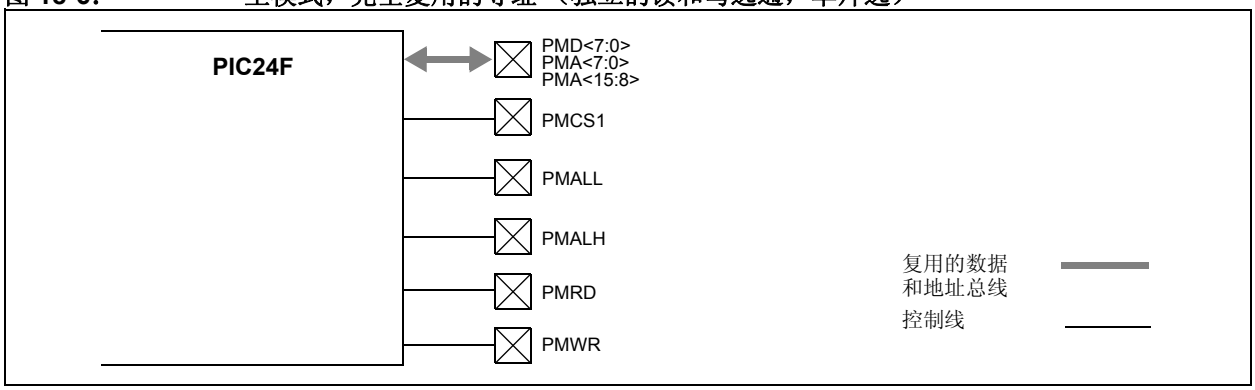


图 18-7：　　复用寻址应用示例

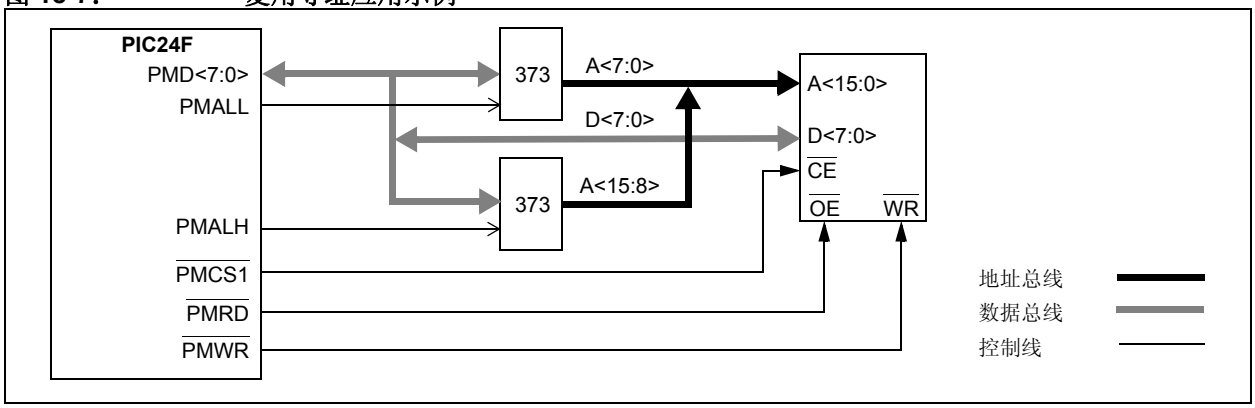


图 18-8：　　部分复用寻址应用示例

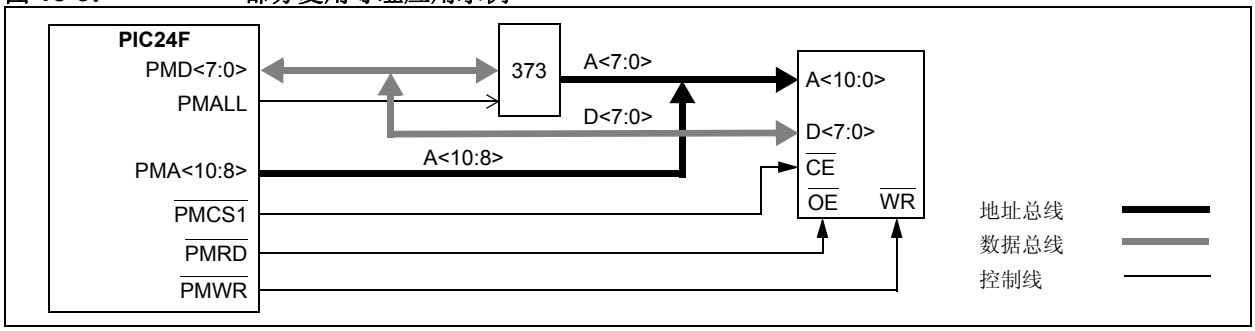


图 18-9：　　8 位地址和数据复用的应用示例

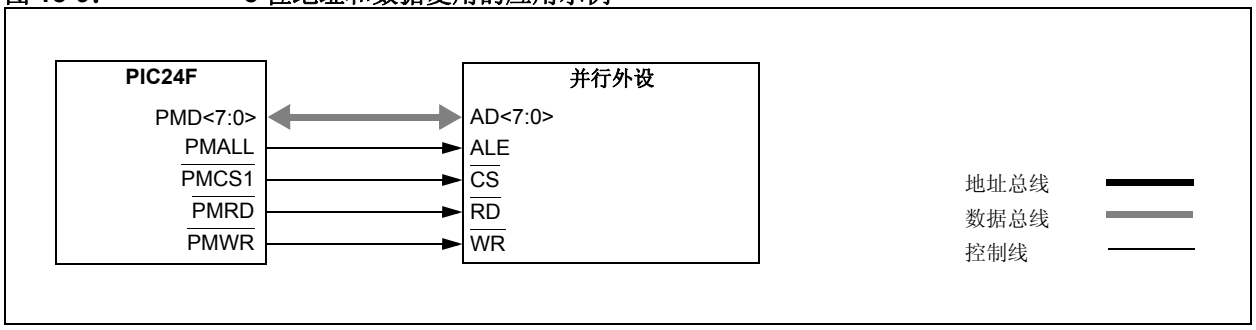


图 18-10：　　并行 EEPROM 示例 （最多 11 位地址， 8 位数据）

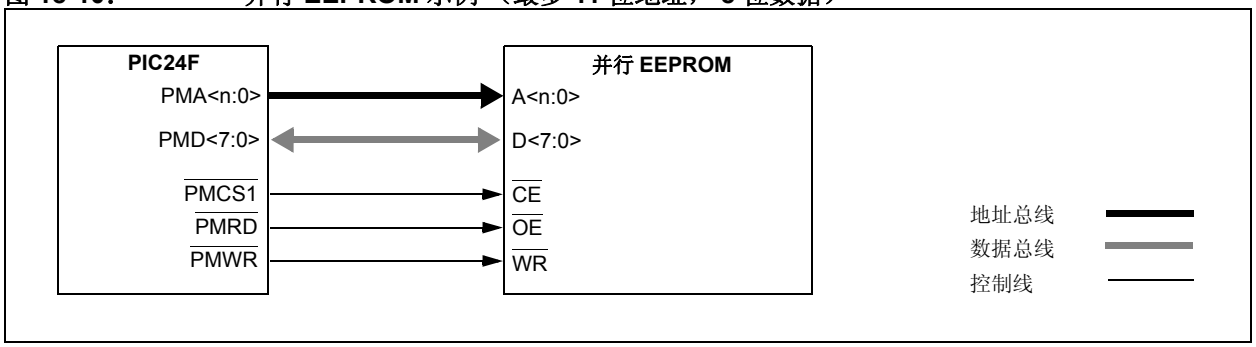


图 18-11：　　并行 EEPROM 示例 （最多 11 位地址， 16 位数据）

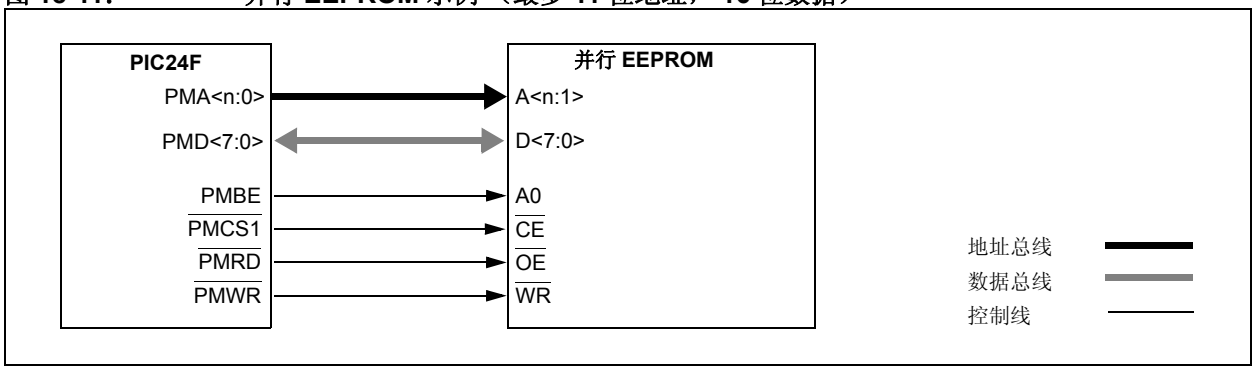
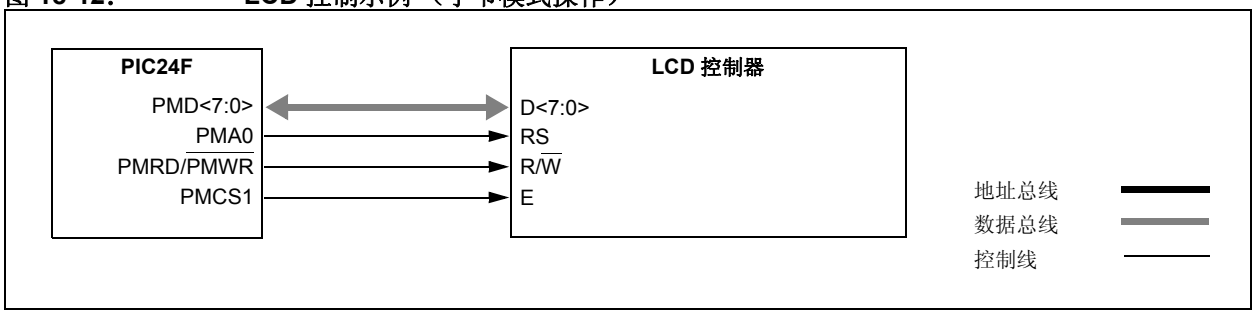


图 18-12：　　LCD 控制示例 （字节模式操作）



19.0 实时时钟和日历 （RTCC）

注：	本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 29 章“实时时钟和日历 （RTCC）”**（DS39696A_CN）。

RTCC 向用户提供了可进行校准的实时时钟和日历（RTCC）功能。

RTCC 模块的主要特性包括：

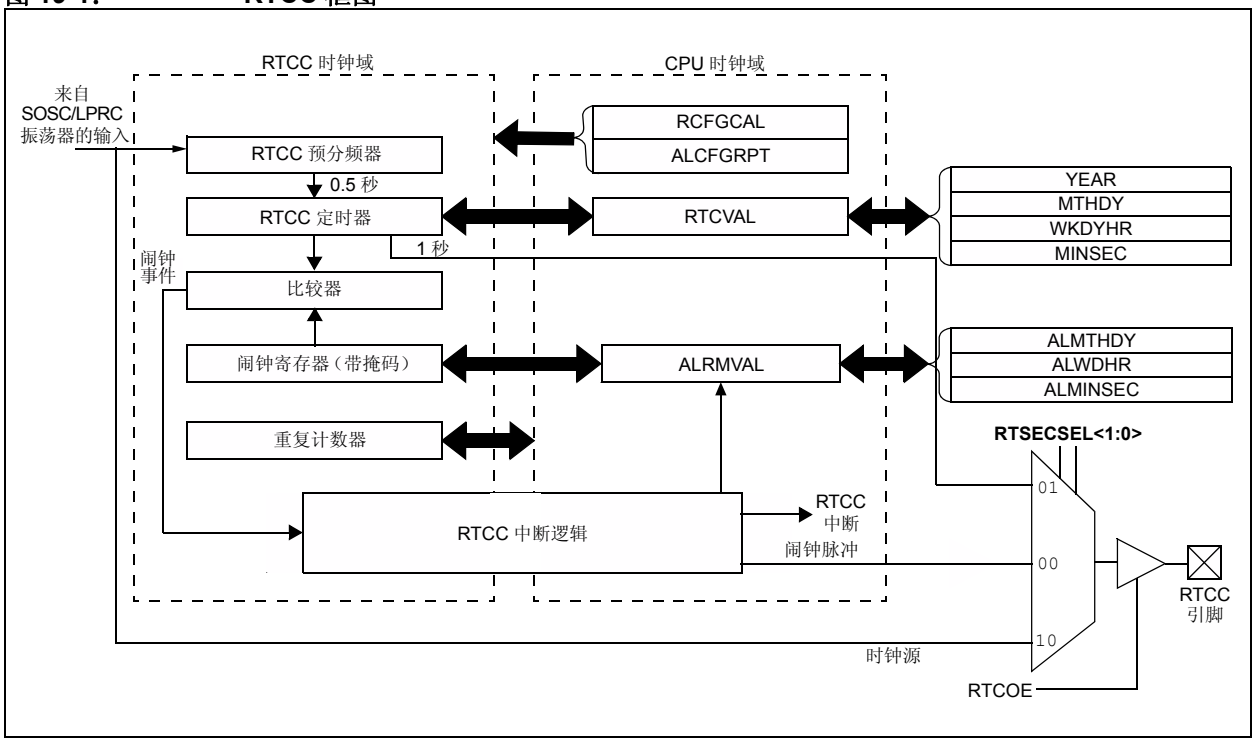
- 可在深度休眠模式下工作
- 可选择的时钟源
- 使用 24 小时格式提供小时、分钟和秒钟
- 可分辨半秒的时长
- 提供日历——星期、日期、月和年
- 闹钟可配置为半秒、1 秒、10 秒、1 分钟、10 分钟、1 小时、 1 天、 1 周、 1 个月或 1 年
- 闹钟使用递减计数器进行重复
- 闹钟具有无限重复响铃
- 年份为 2000 至 2099，带闰年修正
- BCD 格式以减少软件开销
- 为长期电池工作进行了优化
- 32.768 kHz 时钟晶振的用户校准 / 带周期性自动调整的 32K INTRC 频率

19.1 RTCC 源时钟

用户可以选择 SOSC 晶振或 LPRC 低功耗内部振荡器作为 RTCC 模块的参考时钟。可通过 RTCOSC（CW4<5>）配置位进行配置。这为用户提供了一个基于系统总需求来权衡系统成本、精度和功耗的选择。

当器件保持在 $\overline{\text{MCLR}}$ 复位状态时， SOSC 和 RTCC 都将保持运行，并且在 $\overline{\text{MCLR}}$ 无效后仍将继续运行。

图 19-1： RTCC 框图



19.2 RTCC 模块寄存器

RTCC 模块寄存器可分为以下三类：

- RTCC 控制寄存器
- RTCC 值寄存器
- 闹钟值寄存器

19.2.1 寄存器映射

为限制寄存器接口，RTCC 定时器和闹钟定时器寄存器通过相应的寄存器指针访问。RTCC 值寄存器窗口（RTCVALH 和 RTCVALL）使用 RTCPTR 位（RCFGCAL<9:8>）选择所需的定时器寄存器对（见表 19-1）。

通过写 RTCVALH 字节，RTCC 指针值（RTCPTR<1:0> 位）递减 1，直到达到 00。一旦达到 00，分钟和秒值可通过 RTCVALH 和 RTCVALL 访问，直到手动更改指针值。

表 19-1： RTCVAL 寄存器映射

RTCPTR<1:0>	RTCC 值寄存器窗口	
	RTCVAL<15:8>	RTCVAL<7:0>
00	分钟	秒
01	星期	小时
10	月	日
11	—	年

闹钟值寄存器窗口（ALRMVALH 和 ALRMVALL）使用 ALRMPTR 位（ALCFGRPT<9:8>）选择所需的闹钟寄存器对（见表 19-2）。

通过写 ALRMVALH 字节，闹钟指针值（ALRMPTR<1:0> 位）递减 1，直到达到 00。一旦达到 00，ALRMMIN 和 ALRMSEC 值可通过 ALRMVALH 和 ALRMVALL 访问，直到手动更改指针值。

表 19-2： ALRMVAL 寄存器映射

ALRMPTR <1:0>	闹钟值寄存器窗口	
	ALRMVAL<15:8>	ALRMVAL<7:0>
00	ALRMMIN	ALRMSEC
01	ALRMWD	ALRMHR
10	ALRMMNTH	ALRMDAY
11	—	—

考虑到 16 位内核并不区分 8 位和 16 位读操作，用户要注意，读 ALRMVALH 或 ALRMVALL 字节时都会使 ALRMPTR<1:0> 值递减 1。同样的规律也适用于 RTCVALH 或 RTCVALL 字节，读取它们时会使 RTCPTR<1:0> 递减 1。

注： 这只适用于读操作，不适用于写操作。

19.2.2 写锁定

为执行对任何 RTCC 定时器寄存器的写入，RTCWREN 位（RCFGCAL<13>）都必须置 1（见例 19-1）。

注： 为避免意外写入定时器，建议其他任何时候 RTCWREN 位（RCFGCAL<13>）都保持清零。要将 RTCWREN 位置 1，在 55h/AA 序列和 RTCWREN 置 1 之间只允许一个指令周期的时间窗；因此，建议遵循例 19-1 中的代码示例。

19.2.3 选择 RTCC 时钟源

RTCC 模块的时钟源可通过闪存配置位 RTCOSC（CW4<5>）进行选择。当将该位设为 1 时，辅助振荡器（SOSC）用作参考时钟；当将该位设为 0 时，LPRC 用作参考时钟。

例 19-1： 将 RTCWREN 位置 1

```
asm volatile("push w7");
asm volatile("push w8");
asm volatile("disi #5");
asm volatile("mov #0x55, w7");
asm volatile("mov w7, _NVMKEY");
asm volatile("mov #0xAA, w8");
asm volatile("mov w8, _NVMKEY");
asm volatile("bset _RCFGCAL, #13");   //set the RTCWREN bit
asm volatile("pop w8");
asm volatile("pop w7");
```

19.2.4 RTCC 控制寄存器

寄存器 19-1： RCFGCAL：RTCC 校准和配置寄存器 (1)

R/W-0	U-0	R/W-0	R-0, HSC	R-0, HSC	R/W-0	R/W-0	R/W-0
RTCEN(2)	—	RTCWREN	RTCSYNC	HALFSEC(3)	RTCOE	RTCPTR1	RTCPTR0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0
bit 7							bit 0

图注：	HSC = 硬件置 1/ 清零位		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15 **RTCEN**：RTCC 使能位 (2)
1 = 使能 RTCC 模块
0 = 禁止 RTCC 模块

bit 14 **未实现**：读为 0

bit 13 **RTCWREN**：RTCC 值寄存器写使能位
1 = RTCVALH 和 RTCVALL 寄存器可由用户写入
0 = RTCVALH 和 RTCVALL 寄存器已锁定，不可由用户写入

bit 12 **RTCSYNC**：RTCC 值寄存器读同步位
1 = 由于计满返回，RTCVALH、RTCVALL 和 ALCFGRPT 寄存器在读操作过程中可能改变，从而导致读取的数据无效。如果两次读取寄存器得到的数据相同，可认为数据是有效的。
0 = RTCVALH、RTCVALL 或 ALCFGRPT 寄存器在读取时无需考虑计满返回

bit 11 **HALFSEC**：半秒状态位 (3)
1 = 一秒的后半秒
0 = 一秒的前半秒

bit 10 **RTCOE**：RTCC 输出使能位
1 = 使能 RTCC 输出
0 = 禁止 RTCC 输出

bit 9-8 **RTCPTR<1:0>**：RTCC 值寄存器窗口指针位
读取 RTCVALH 和 RTCVALL 寄存器时，指向相应的 RTCC 值寄存器。每当读或写 RTCVALH 时 RTCPTR<1:0> 的值就递减 1，直到达到 00。
RTCVAL<15:8>：
00 = 分钟数
01 = 星期
10 = 月
11 = 保留
RTCVAL<7:0>：
00 = 秒数
01 = 小时数
10 = 日
11 = 年

注 1： RCFGCAL 寄存器只受 POR 的影响。
2： 仅当 RTCWREN = 1 时才允许写入 RTCEN 位。
3： 该位是只读位；写入 MINSEC 寄存器的低半部分时，它被清零。

寄存器 19-1： RCFGCAL：RTCC 校准和配置寄存器 (1)（续）

bit 7-0 **CAL<7:0>**：RTC 漂移校准位
01111111 = 最大正向调整；每分钟增加 508 个 RTC 时钟脉冲
.
.
.
01111111 = 最小正向调整；每分钟增加 4 个 RTC 时钟脉冲
00000000 = 无调整
11111111 = 最小负向调整；每分钟减少 4 个 RTC 时钟脉冲
.
.
.
10000000 = 最大负向调整；每分钟减少 512 个 RTC 时钟脉冲

注 1： RCFGCAL 寄存器只受 POR 的影响。
2： 仅当 RTCWREN = 1 时才允许写入 RTCEN 位。
3： 该位是只读位；写入 MINSEC 寄存器的低半部分时，它被清零。

寄存器 19-2： PADCFG1：焊盘配置控制寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	RTSECSEL1(1)	RTSECSEL0(1)	PMPTTL
bit 7							bit 0

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

bit 15-3 **未实现**：读为 0
bit 2-1 **RTSECSEL<1:0>**：RTCC 秒时钟输出选择位 (1)
11 = 保留；不要使用
10 = 选择 RTCC 引脚输出 RTCC 源时钟（时钟可以是 LPRC 或 SOSC，取决于 RTCOSC 位（CW4<5>）的设置）
01 = 选择 RTCC 引脚输出 RTCC 秒时钟
00 = 选择 RTCC 引脚输出 RTCC 闹钟脉冲
bit 0 **PMPTTL**：PMP 模块 TTL 输入缓冲器选择位
1 = PMP 模块使用 TTL 输入缓冲器
0 = PMP 模块使用施密特触发器输入缓冲器

注 1： 要使能实际 RTCC 输出，需要将 RTCOE （RCFGCAL<10>）位置 1。

寄存器 19-3：　　ALCFGRPT：闹钟配置寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ALRMEN	CHIME	AMASK3	AMASK2	AMASK1	AMASK0	ALRMPTR1	ALRMPTR0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ARPT7	ARPT6	ARPT5	ARPT4	ARPT3	ARPT2	ARPT1	ARPT0
bit 7							bit 0

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

bit 15　**ALRMEN：**闹钟使能位

1 = 使能闹钟 （每当 ARPT<7:0> = 00h 且 CHIME = 0，发生闹钟事件后都自动清零）
0 = 禁止闹钟

bit 14　**CHIME：**响铃 （chime）使能位

1 = 使能响铃； ARPT<7:0> 位允许从 00h 返回到 FFh
0 = 禁止响铃； ARPT<7:0> 位到达 00h 就停止

bit 13-10　**AMASK<3:0>：**闹钟掩码配置位

0000 = 每半秒
0001 = 每秒
0010 = 每 10 秒
0011 = 每分钟
0100 = 每 10 分钟
0101 = 每小时
0110 = 一天一次
0111 = 一周一次
1000 = 一月一次
1001 = 一年一次 （配置在 2 月 29 日除外，这种情况下每 4 年一次）
101x = 保留；不要使用
11xx = 保留；不要使用

bit 9-8　**ALRMPTR<1:0>：**闹钟值寄存器窗口指针位

读取 ALRMVALH 和 ALRMVALL 寄存器时，指向相应的闹钟值寄存器。每当读或写 ALRMVALH 时 ALRMPTR<1:0> 的值就递减 1，直到达到 00。

ALRMVAL<15:8>：
00 = ALRMMIN
01 = ALRMWD
10 = ALRMMNTH
11 = 未实现

ALRMVAL<7:0>：
00 = ALRMSEC
01 = ALRMHR
10 = ALRMDAY
11 = 未实现

bit 7-0　**ARPT<7:0>：**闹钟重复计数器值位

11111111 = 闹钟将再重复 255 次
.
.
.
00000000 = 闹钟将不再重复
每当发生闹钟事件时计数器就递减 1 ；除非 CHIME = 1，否则计数器不能从 00h 返回到 FFh。

19.2.5 RTCVAL 寄存器映射

寄存器 19-4： YEAR：年值寄存器 (1)

U-0, HSC	U-0, HSC	U-0, HSC	U-0, HSC	U-0, HSC	U-0, HSC	U-0, HSC	U-0, HSC
—	—	—	—	—	—	—	—
bit 15							bit 8

R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC
YRTEN3	YRTEN2	YRTEN1	YRTEN0	YRONE3	YRONE2	YRONE1	YRONE0
bit 7							bit 0

图注：	HSC = 硬件置 1/ 清零位		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15-8 **未实现：**读为 0

bit 7-4 **YRTEN<3:0>：**年份的十位数的二十进制码 （Binary Coded Decimal， BCD）值
值为 0 到 9。

bit 3-0 **YRONE<3:0>：**年份的个位数的 BCD 值
值为 0 到 9。

注 1： 仅当 RTCWREN = 1 时才允许写入 YEAR 寄存器。

寄存器 19-5： MTHDY：月和日值寄存器 (1)

U-0, HSC	U-0, HSC	U-0, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC
—	—	—	MTHTEN0	MTHONE3	MTHONE2	MTHONE1	MTHONE0
bit 15							bit 8

U-0, HSC	U-0, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC
—	—	DAYTEN1	DAYTEN0	DAYONE3	DAYONE2	DAYONE1	DAYONE0
bit 7							bit 0

图注：	HSC = 硬件置 1/ 清零位		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15-13 **未实现：**读为 0

bit 12 **MTHTEN0：**月份的十位数的 BCD 值
值为 0 或 1。

bit 11-8 **MTHONE<3:0>：**月份的个位数的 BCD 值
值为 0 到 9。

bit 7-6 **未实现：**读为 0

bit 5-4 **DAYTEN<1:0>：**日的十位数的 BCD 值
值为 0 到 3。

bit 3-0 **DAYONE<3:0>：**日的个位数的 BCD 值
值为 0 到 9。

注 1： 仅当 RTCWREN = 1 时才允许写入该寄存器。

寄存器 19-6： WKDYHR：星期和小时值寄存器 (1)

U-0, HSC	U-0, HSC	U-0, HSC	U-0, HSC	U-0, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC
—	—	—	—	—	WDAY2	WDAY1	WDAY0
bit 15							bit 8

U-0, HSC	U-0, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC
—	—	HRTEN1	HRTEN0	HRONE3	HRONE2	HRONE1	HRONE0
bit 7							bit 0

图注：	HSC = 硬件置 1/ 清零位		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15-11 **未实现：**读为 0

bit 10-8 **WDAY<2:0>：**星期的 BCD 值
值为 0 到 6。

bit 7-6 **未实现：**读为 0

bit 5-4 **HRTEN<1:0>：**小时的十位数的 BCD 值
值为 0 到 2。

bit 3-0 **HRONE<3:0>：**小时的个位数的 BCD 值
值为 0 到 9。

注 1： 仅当 RTCWREN = 1 时才允许写入该寄存器。

寄存器 19-7： MINSEC：分钟和秒值寄存器

U-0, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC
—	MINTEN2	MINTEN1	MINTEN0	MINONE3	MINONE2	MINONE1	MINONE0
bit 15							bit 8

U-0, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC	R/W-x, HSC
—	SECTEN2	SECTEN1	SECTEN0	SECONE3	SECONE2	SECONE1	SECONE0
bit 7							bit 0

图注：	HSC = 硬件置 1/ 清零位		
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15 **未实现：**读为 0

bit 14-12 **MINTEN<2:0>：**分钟的十位数的 BCD 值
值为 0 到 5。

bit 11-8 **MINONE<3:0>：**分钟的个位数的 BCD 值
值为 0 到 9。

bit 7 **未实现：**读为 0

bit 6-4 **SECTEN<2:0>：**秒的十位数的 BCD 值
值为 0 到 5。

bit 3-0 **SECONE<3:0>：**秒的个位数的 BCD 值
值为 0 到 9。

19.2.6 ALRMVAL 寄存器映射

寄存器 19-8： ALMTHDY：闹钟月和日值寄存器 (1)

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	MTHTEN0	MTHONE3	MTHONE2	MTHONE1	MTHONE0
bit 15							bit 8

U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	DAYTEN1	DAYTEN0	DAYONE3	DAYONE2	DAYONE1	DAYONE0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12 **MTHTEN0：**月份的十位数的 BCD 值
值为 0 或 1。

bit 11-8 **MTHONE<3:0>：**月份的个位数的 BCD 值
值为 0 到 9。

bit 7-6 **未实现：**读为 0

bit 5-4 **DAYTEN<1:0>：**日的十位数的 BCD 值
值为 0 到 3。

bit 3-0 **DAYONE<3:0>：**日的个位数的 BCD 值
值为 0 到 9。

注 1： 仅当 RTCWREN = 1 时才允许写入该寄存器。

寄存器 19-9： ALWDHR：闹钟星期和小时值寄存器 (1)

U-0	U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x
—	—	—	—	—	WDAY2	WDAY1	WDAY0
bit 15							bit 8

U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	HRTEN1	HRTEN0	HRONE3	HRONE2	HRONE1	HRONE0
bit 7							bit 0

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

bit 15-11 **未实现：**读为 0

bit 10-8 **WDAY<2:0>：**星期的 BCD 值
值为 0 到 6。

bit 7-6 **未实现：**读为 0

bit 5-4 **HRTEN<1:0>：**小时的十位数的 BCD 值
值为 0 到 2。

bit 3-0 **HRONE<3:0>：**小时的个位数的 BCD 值
值为 0 到 9。

注 1： 仅当 RTCWREN = 1 时才允许写入该寄存器。

寄存器 19-10： ALMINSEC：闹钟分钟和秒值寄存器

U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	MINTEN2	MINTEN1	MINTEN0	MINONE3	MINONE2	MINONE1	MINONE0
bit 15							bit 8

U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	SECTEN2	SECTEN1	SECTEN0	SECONE3	SECONE2	SECONE1	SECONE0
bit 7							bit 0

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

bit 15 **未实现：**读为 0

bit 14-12 **MINTEN<2:0>：**分钟的十位数的 BCD 值
值为 0 到 5。

bit 11-8 **MINONE<3:0>：**分钟的个位数的 BCD 值
值为 0 到 9。

bit 7 **未实现：**读为 0

bit 6-4 **SECTEN<2:0>：**秒的十位数的 BCD 值
值为 0 到 5。

bit 3-0 **SECONE<3:0>：**秒的个位数的 BCD 值
值为 0 到 9。

19.3 校准

实时晶振输入可用周期性自动调节功能校准。正确校准后，RTCC 可提供小于每月 3 秒的误差。为实现该校准，需要找到误差时钟脉冲数并将该值存储到RCFGCAL 寄存器的低半部分。装入 RCFGCAL 低半部分的 8 位有符号值乘以4，每分钟从RTCC定时器中加上或减去一次。关于 RTCC 校准，请参见以下步骤：

1. 使用器件上的其他定时器资源；用户必须找出 32.768 kHz 晶振的误差。
2. 知道误差后，必须将它转换为每分钟误差时钟脉冲数。
3. a) 如果振荡器快于理想频率 （从步骤 2 得出负的结果），RCFGCAL 寄存器值必须为负。这会导致每分钟从定时器计数器中减去指定的时钟脉冲数。

   b) 如果振荡器慢于理想频率 （从步骤 2 得出正的结果），RCFGCAL 寄存器值必须为正。这会导致每分钟从定时器计数器中加上指定的时钟脉冲数。

将每分钟误差时钟数除以 4 得到正确的校准值，并将正确的值装入 RCFGCAL 寄存器。（校准值中每 1 位递增会加上或减去 4 个脉冲。）

公式 19-1：

（理想频率 † – 测得频率）* 60 = 每分钟误差时钟数

† 理想频率 = 32,768 Hz

只有当定时器关闭或紧接秒脉冲的上升沿时，才会发生对 RCFGCAL 寄存器低半部分的写操作。

注：	是否在误差值中包含由于温度和晶振老化造成的漂移，由用户自行决定。

19.4 闹钟

- 可在半秒到一年的范围内配置
- 使用 ALRMEN 位 （ALCFGRPT<15>）使能
- 有一次性闹钟和重复闹钟选项可用

19.4.1 配置闹钟

闹钟功能用 ALRMEN 位使能。发出闹钟后该位清零。只能在 ALRMEN = 0 时对 ALRMVAL 执行写操作。

闹钟的间隔时间通过AMASK位（ALCFGRPT<13:10>）选择，如图 19-2 所示。这些位决定了要触发闹钟，闹钟的哪些位以及多少位必须和时钟值匹配。

也可以配置闹钟使之根据预先配置的间隔时间重复。闹钟使能后发生的总次数存储在 ARPT<7:0> 位（ALCFGRPT<7:0>）中。当 ARPT 位的值等于 00h 且 CHIME 位（ALCFGRPT<14>）清零时，重复功能被禁止，只发生单次闹钟。通过将 FFh 装入 ARPT<7:0>，闹钟可重复最多 255 次。

每个闹钟发出后， ARPT 位的值都递减 1。值达到 00h 后，将最后一次发出闹钟，此后 ALRMEN 位将自动清零，闹钟将关闭。

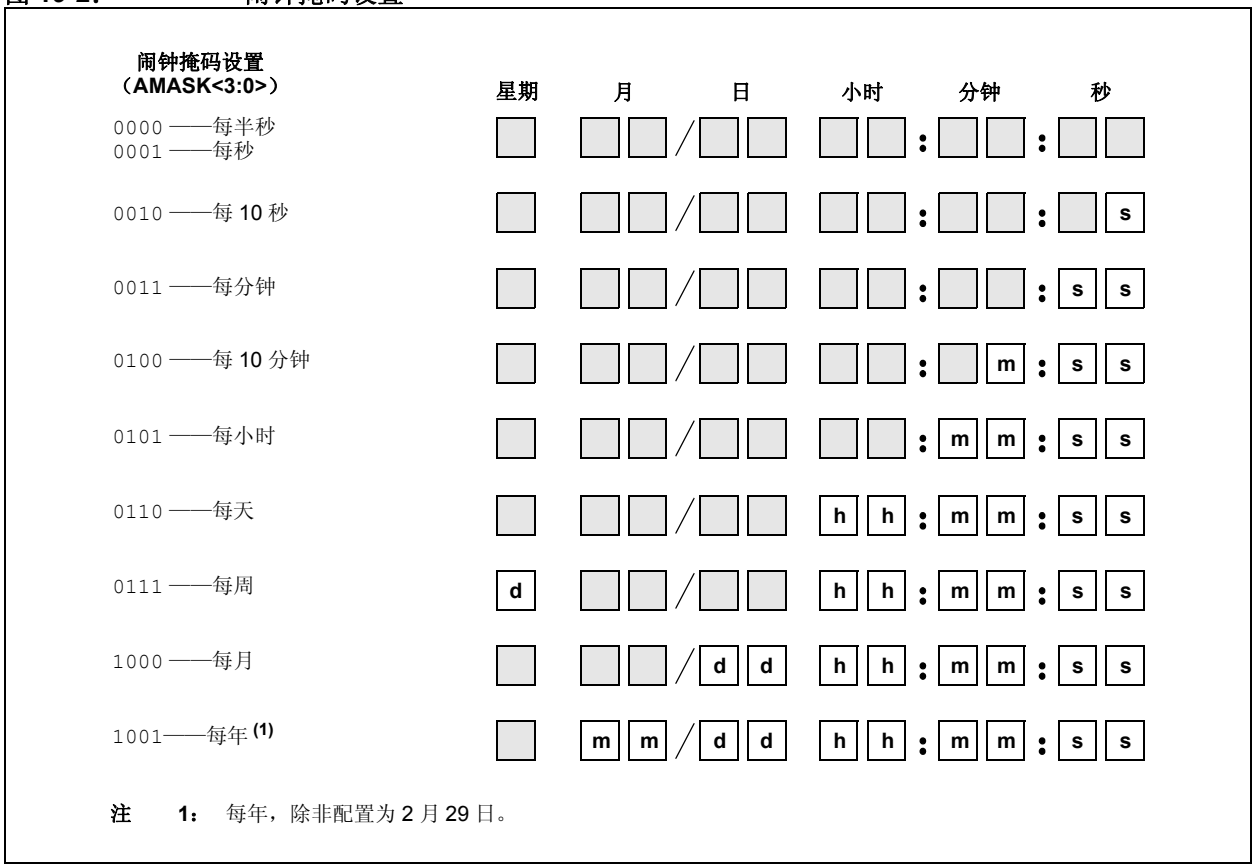
如果 CHIME 位 = 1，闹钟可能不断重复。当 CHIME 置 1 时，ARPT 位的值达到 00h 时不会禁止闹钟，而是返回到 FFh，继续无限计数。

19.4.2 闹钟中断

每个闹钟事件发生时，都会产生中断。此外会提供闹钟脉冲输出，其频率是闹钟频率的一半。该输出完全和 RTCC 时钟同步，可用作其他外设的触发时钟。

注：	使能闹钟 （ALRMEN = 1）时，更改除 RCFGCAL 和ALCFGRPT 寄存器以外的任何寄存器以及 CHIME 位，都会导致误闹钟事件，进而导致错误的闹钟中断。为避免误闹钟事件，只应在禁止闹钟（ALRMEN = 0）时更改定时器和闹钟值。建议在 RTCSYNC = 0 时更改 ALCFGRPT 寄存器和 CHIME 位。

图 19-2：　　闹钟掩码设置



注　　1： 每年，除非配置为 2 月 29 日。

注：

20.0 32 位可编程循环冗余校验 （CRC） 发生器

注： 本数据手册总结了PIC24F系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 30 章 “可编程循环冗余校验 （CRC）”**（DS39714A_CN）。

可编程 CRC 发生器提供了一种由硬件实现的方法，为各种联网和安防应用快速生成校验和。它具有以下特性：

- 用户可编程 CRC 多项式方程，最多 32 位
- 可编程移位方向 （小尾数或大尾数）
- 独立的数据和多项式长度
- 可配置的中断输出
- 数据 FIFO

图 20-1 给出了 CRC 发生器的简化框图。图 20-2 给出了 CRC 移位引擎的简化版本。

图 20-1： CRC 框图

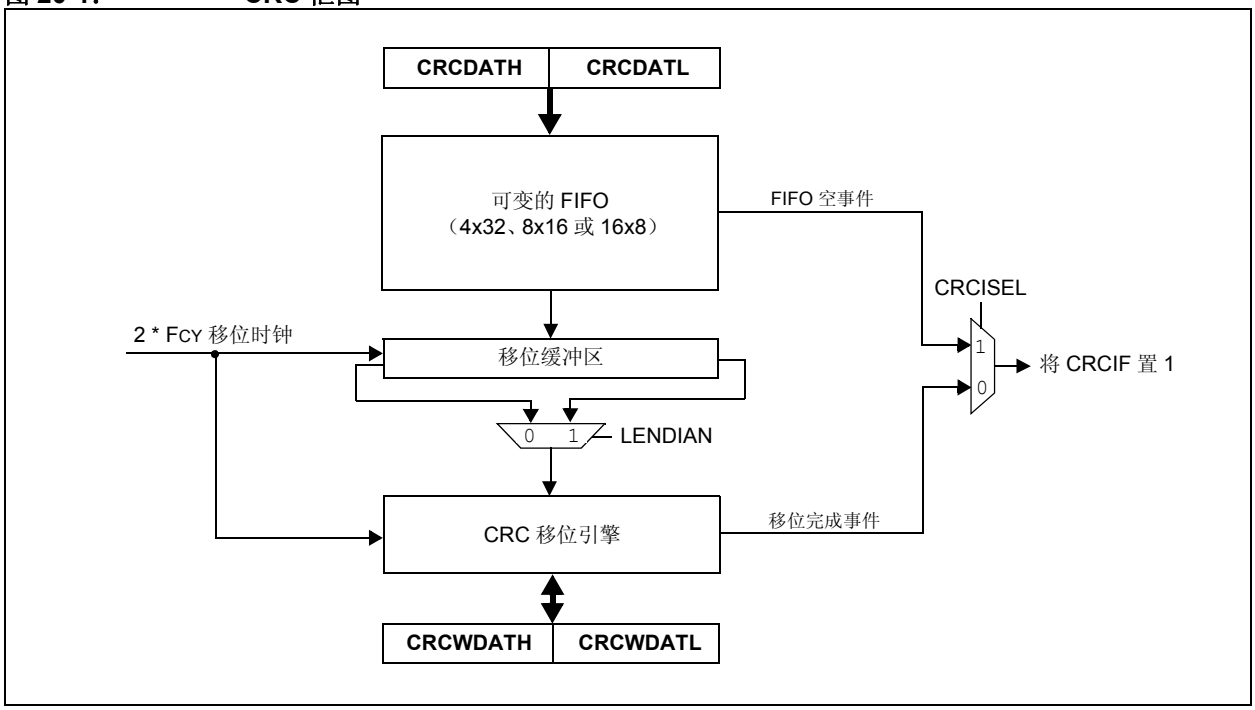
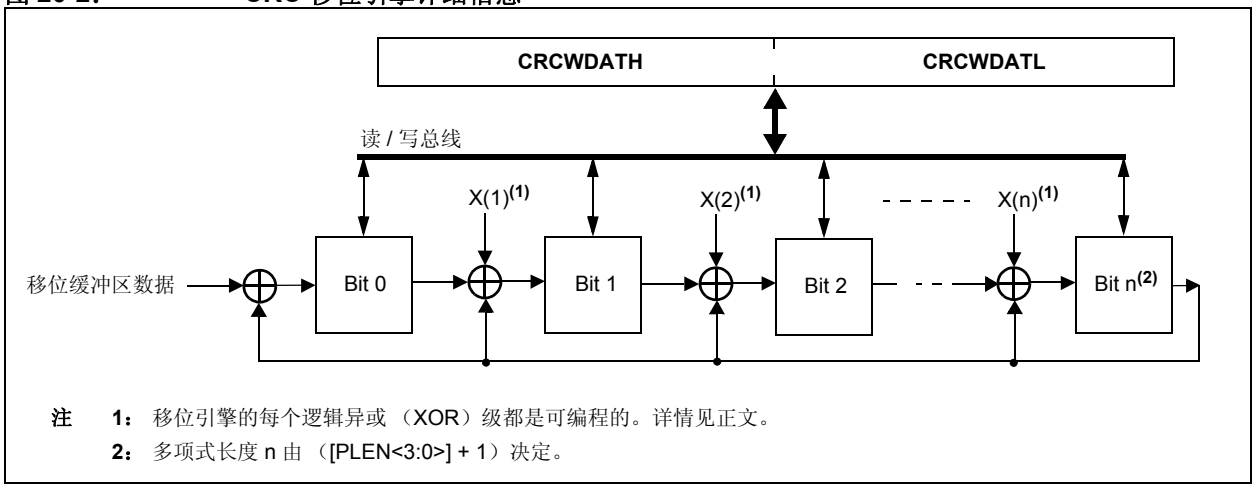


图 20-2： CRC 移位引擎详细信息



注 1： 移位引擎的每个逻辑异或 （XOR）级都是可编程的。详情见正文。

2： 多项式长度 n 由 （[PLEN<3:0>] + 1）决定。

20.1 用户接口

20.1.1 多项式接口

CRC 模块最多可使用 32 位，编程最高 32 阶的 CRC 多项式。多项式长度代表方程中的最高指数，它通过 PLEN<4:0> 位 （CRCCON2<4:0>）进行选择。

CRCXORL 和 CRCXORH 寄存器控制在方程中包含哪些指数项。将特定位置 1 时，方程中将包含相应的指数项；从功能角度来说，将特定位置 1 时，在 CRC 引擎中将对相应位进行异或运算。清零该位将会禁止异或运算。

例如，假设有两个 CRC 多项式，一个是 16 位方程，另一个是 32 位方程：

$$x16 + x12 + x5 + 1$$

和

$$x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1$$

在 CRC 发生器中编程这两个多项式，应按照表 20-1 所示来设置寄存器位。

可以注意到，相应的一些位设置为 1，指示要在方程中使用它们（例如，X26 和 X23）。bit 0 对于方程是必需的，总是进行异或运算；所以，X0 是无关位。对于长度为 N 的多项式，无论第 *N* 位的设置如何，总是假定使用该位。因此，对于长度为 32 的多项式，在 CRCxOR 寄存器中没有第 32 位。

20.1.2 数据接口

模块具有可处理可变数据宽度的 FIFO。输入数据宽度可以使用 DWIDTH<4:0> 位（CRCCON2<12:8>）配置为 1 和 32 位之间的任意值。当数据宽度大于 15 时，FIFO 为 4 字深。当 DWIDTH 值介于 15 和 8 之间时，FIFO 为 8 字深。当 DWIDTH 值小于 8 时， FIFO 为 16 字深。

首先必须将要计算 CRC 的数据写入 FIFO。即使数据宽度小于 8，可写入 FIFO 的最小数据元素也是一个字节。例如，如果 DWIDTH 值为 5，那么数据宽度为 DWIDTH + 1（即 6）。数据以整个字节的形式写入；模块会忽略两个未用的高位。

数据写入 CRCDAT 寄存器的 MSb （即，由数据宽度定义的MSb）之后，VWORD<4:0>位（CRCCON1<12:8>）的值会递增 1。例如，如果 DWIDTH 值为 24，当写入 CRCDATH 的 bit 7 时， VWORD 位会递增 1。因此，必须总是先写入 CRCDATL，然后再写入 CRCDATH。

当 CRCGO 位置 1，并且 VWORD 的值大于 0 时，CRC 引擎会开始对数据进行移位。每个字从 FIFO 复制到缓冲寄存器中，复制之后 VWORD 会递减 1。然后数据从缓冲区移出。CRC 引擎会不断以每个指令周期 2 位的速率进行移位，直到 VWORD 值变为 0。这意味着，对于给定的数据宽度，需要等于数据宽度一半的指令数完成每个字的计算。例如，对于 32 位的单字数据，需要 16 个周期来计算 CRC。

当 VWORD 值达到对应于 DWIDTH 配置值的最大值（4、8 或 16）时， CRCFUL 位会置 1。当 VWORD 值变为 0 时，CRCMPT 位会置 1。每当 CRCEN 为 0 时，FIFO 会清空， VWORD<4:0> 位会设置为 00000。

在写入 CRCDAT 之后，必须至少经过一个指令周期才可以读 VWORD 位。

表 20-1： 16 和 32 位多项式的 CRC 设置示例

CRC 控制位	位值	
	16 位多项式	32 位多项式
PLEN<4:0>	01111	11111
X<31:16>	0000 0000 0000 000x	0000 0100 1100 0001
X<15:0>	0001 0000 0010 000x	0001 1101 1011 011x

20.1.3 数据移位方向

LENDIAN 位（CRCCON1<3>）用于控制移位方向。默认情况下， CRC 从 MSb 开始通过引擎进行数据移位。将 LENDIAN 置 1（= 1）时， CRC 会从 LSb 开始进行数据移位。通过该设置，可以更好地与各种通信方案进行集成，并且消除在软件中反转位顺序的开销。注意，该设置仅改变移入引擎的数据的方向。CRC 计算的结果仍然是正常的 CRC 结果，不是反转的 CRC 结果。

20.1.4 中断操作

模块可以在两种条件下产生中断，可由用户配置。

如果 CRCISEL 为 0，则在 VWORD<4:0> 位的值从 1 变为 0 时产生中断。如果 CRCISEL 为 1，则在 CRC 运算完成，模块将 CRCGO 位设置为 0 之后产生中断。手动将 CRCGO 设置为 0 不会产生中断。

20.1.5 典型操作

使用模块进行典型的 CRC 计算：

1. 将 CRCEN 位置 1 以使能模块。
2. 将模块配置为所需的操作：
   a) 使用 CRCXORL 和 CRCXORH 寄存器，以及 PLEN<4:0> 位编程所需的多项式
   b) 使用 DWIDTH 和 LENDIAN 位配置数据宽度和移位方向
   c) 使用 CRCISEL 位选择所需的中断模式
3. 通过写入 CRCDATL 和 CRCDATH 寄存器在 FIFO 中预先装入数据，直到 CRCFUL 位置 1（即没有剩余数据）
4. 通过将 00h 写入 CRCWDATL 和 CRCWDATH 来清除旧结果。此外，也可以将 CRCWDAT 保留不变，以继续先前暂停的计算。
5. 将 CRCGO 位置 1 以启动计算。
6. 在有空间可用时，将剩余的数据写入 FIFO。
7. 在计算完成时， CRCGO 会自动清零。如果 CRCISEL = 1，则会产生中断。
8. 读取 CRCWDATL 和 CRCWDATH，获取计算结果。

20.2 寄存器

有 8 个寄存器与该模块关联：

- CRCCON1
- CRCCON2
- CRCXORL
- CRCXORH
- CRCDATL
- CRCDATH
- CRCWDATL
- CRCWDATH

CRCCON1 和 CRCCON2 寄存器（寄存器 20-1 和寄存器 20-2）用于控制模块操作和配置各种设置。CRCXOR 寄存器（寄存器 20-3 和寄存器 20-4）用于选择要在 CRC 方程中使用的多项式项。 CRCDAT 和 CRCWDAT 寄存器各自均为一个寄存器对，分别用作双字输入数据和 CRC 处理输出的缓冲区。

寄存器 20-1：　　CRCCON1：CRC 控制寄存器 1

R/W-0	U-0	R/W-0	R-0	R-0	R-0	R-0	R-0
CRCEN	—	CSIDL	VWORD4	VWORD3	VWORD2	VWORD1	VWORD0
bit 15							bit 8

R-0, HCS	R-1, HCS	R/W-0	R/W-0, HC	R/W-0	U-0	U-0	U-0
CRCFUL	CRCMPT	CRCISEL	CRCGO	LENDIAN	—	—	—
bit 7							bit 0

图注：	HC = 硬件清零位	HCS = 硬件清零 / 置 1 位	
R = 可读位	W = 可写位	U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 15　**CRCEN**：CRC 使能位
1 = 使能模块
0 = 使能模块。所有状态机、指针和 CRCWDAT/CRCDAT 均复位；其他 SFR 不复位。

bit 14　**未实现**：读为 0

bit 13　**CSIDL**：CRC 空闲模式停止位
1 = 当器件进入空闲模式时，模块停止工作
0 = 在空闲模式下模块继续工作

bit 12-8　**VWORD<4:0>**：指针值位
表示 FIFO 中有效字的个数。 PLEN<3:0> > 7 时最大值是 8， PLEN<3:0> ≤ 7 时最大值是 16。

bit 7　**CRCFUL**：FIFO 满位
1 = FIFO 已满
0 = FIFO 未满

bit 6　**CRCMPT**：FIFO 空位
1 = FIFO 为空
0 = FIFO 非空

bit 5　**CRCISEL**：CRC 中断选择位
1 = 在 FIFO 为空时产生中断； CRC 计算未完成
0 = 在移位完成且 CRCWDAT 结果就绪时产生中断

bit 4　**CRCGO**：启动 CRC 位
1 = 启动 CRC 串行移位器
0 = CRC 串行移位器关闭

bit 3　**LENDIAN**：数据移位方向选择位
1 = 数据字从 LSb 开始移入 CRC （小尾数）
0 = 数据字从 MSb 开始移入 CRC （大尾数）

bit 2-0　**未实现**：读为 0

寄存器 20-2： CRCCON2：CRC 控制寄存器 2

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	DWIDTH4	DWIDTH3	DWIDTH2	DWIDTH1	DWIDTH0
bit 15							bit 8

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	PLEN4	PLEN3	PLEN2	PLEN1	PLEN0
bit 7							bit 0

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

bit 15-13 **未实现：**读为 0

bit 12-8 **DWIDTH<4:0>：**数据宽度选择位
定义数据字的宽度 （数据字宽度 = （DWIDTH<4:0>）+ 1）。

bit 7-5 **未实现：**读为 0

bit 4-0 **PLEN<4:0>：**多项式长度选择位
定义 CRC 多项式的长度 （多项式长度 = （PLEN<4:0>）+ 1）。

寄存器 20-3： CRCXORL：CRC 异或多项式寄存器，低字节

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
X15	X14	X13	X12	X11	X10	X9	X8
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
X7	X6	X5	X4	X3	X2	X1	—
bit 7							bit 0

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

bit 15-1 **X<15:1>：**多项式的项 X^n 的异或使能位

bit 0 **未实现：**读为 0

寄存器 20-4：　　CRCXORH：CRC 异或多项式寄存器，高字节

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
X31	X30	X29	X28	X27	X26	X25	X24
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
X23	X22	X21	X20	X19	X18	X17	X16
bit 7							bit 0

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

bit 15-0　　**X<31:16>：**多项式的项 X^n 的异或使能位

21.0 10 位高速 A/D 转换器

注： 本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 17 章 “10 位 A/D 转换器”**（DS39705A_CN）。

10 位 A/D 转换器具有以下主要特性：

- 逐次逼近（Successive Approximation，SAR）转换
- 转换速度最高可达 500 ksps
- 13 个模拟输入引脚
- 外部参考电压输入引脚
- 内部带隙参考输入
- 自动通道扫描模式
- 可选择转换触发源
- 16 字的转换结果缓冲区
- 可选择缓冲区填充模式
- 4 个结果对齐选项
- 可在 CPU 休眠和空闲模式下工作

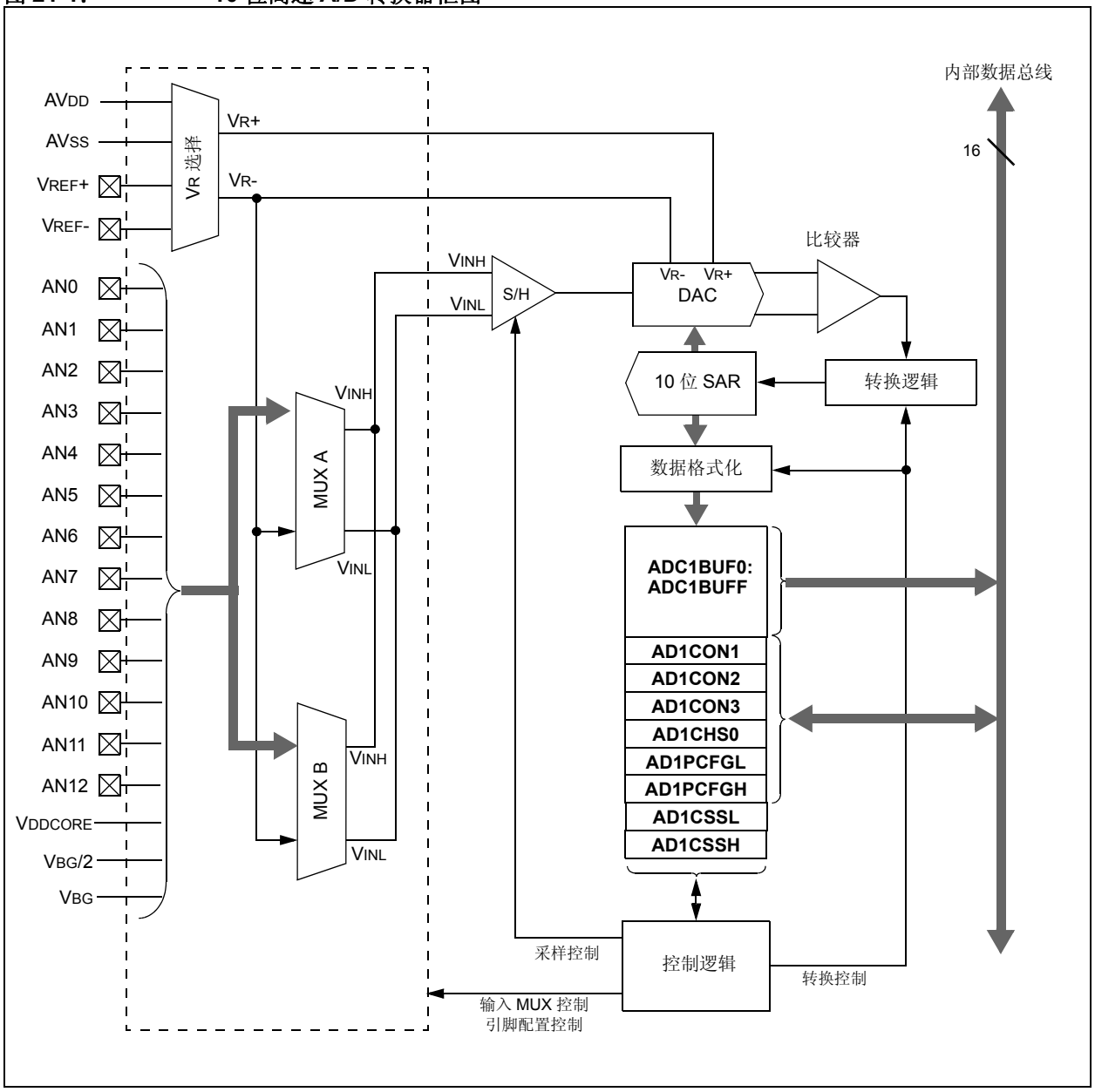
在所有 PIC24FJ64GA104 系列器件上，10 位 A/D 转换器有 13 个模拟输入引脚，指定为从 AN0 至 AN12。此外，还有两个模拟输入引脚用于连接外部参考电压（VREF+ 和 VREF-）。这两个参考电压输入可以与其他模拟输入引脚共用。

图 21-1 给出了 A/D 转换器的框图。

要执行 A/D 转换：

1. 配置 A/D 模块：
   a) 将端口引脚配置为模拟输入和 / 或选择带隙参考输入（AD1PCFGL<15:0> 和 AD1PCFGH<1:0>）。
   b) 选择参考电压源以匹配模拟输入的预期范围（AD1CON2<15:13>）。
   c) 选择模拟转换时钟以使期望的数据速率与处理器时钟匹配（AD1CON3<7:0>）。
   d) 选择适当的采样 / 转换序列（AD1CON1<7:5> 和 AD1CON3<12:8>）。
   e) 选择转换结果在缓冲区中的存储方式（AD1CON1<9:8>）。
   f) 选择中断频率（AD1CON2<5:2>）。
   g) 开启 A/D 模块（AD1CON1<15>）。
2. 配置 A/D 中断（如果需要）：
   a) 清零 AD1IF 位。
   b) 选择 A/D 中断优先级。

图 21-1：　　10 位高速 A/D 转换器框图



寄存器 21-1：　AD1CON1：A/D 控制寄存器 1

R/W-0	U-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0
ADON(1)	—	ADSIDL	—	—	—	FORM1	FORM0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0, HCS	R/C-0, HCS
SSRC2	SSRC1	SSRC0	—	—	ASAM	SAMP	DONE
bit 7							bit 0

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

bit 15　**ADON**：A/D 工作模式位 (1)
1 = A/D 转换器模块正在工作
0 = A/D 转换器关闭

bit 14　**未实现**：读为 0

bit 13　**ADSIDL**：空闲模式停止位
1 = 当器件进入空闲模式时，模块停止工作
0 = 在空闲模式下模块继续工作

bit 12-10　**未实现**：读为 0

bit 9-8　**FORM<1:0>**：数据输出格式位
11 = 有符号小数 （sddd dddd dd00 0000）
10 = 小数 （dddd dddd dd00 0000）
01 = 有符号整数 （ssss sssd dddd dddd）
00 = 整数 （0000 00dd dddd dddd）

bit 7-5　**SSRC<2:0>**：转换触发源选择位
111 = 由内部计数器结束采样并启动转换 （自动转换）
110 = 由 CTMU 事件结束采样并启动转换
101 = 保留
100 = 由 Timer5 比较结束采样并启动转换
011 = 保留
010 = 由 Timer3 比较结束采样并启动转换
001 = 由 INT0 引脚的有效跳变结束采样并启动转换
000 = 由清零 SAMP 位结束采样并启动转换

bit 4-3　**未实现**：读为 0

bit 2　**ASAM**：A/D 采样自动启动位
1 = 最后一次转换结束后立即开始采样； SAMP 位自动置 1
0 = SAMP 位置 1 时开始采样

bit 1　**SAMP**：A/D 采样使能位
1 = A/D 采样 / 保持放大器正在对输入进行采样
0 = A/D 采样 / 保持放大器保持采样结果

bit 0　**DONE**：A/D 转换状态位
1 = A/D 转换已完成
0 = A/D 转换未完成

注　1： 一旦 ADON 位清零，ADC1BUFx 寄存器将不能保留其中的值。在禁止模块之前，从缓冲区中读出转换值。

寄存器 21-2： AD1CON2：A/D 控制寄存器 2

R/W-0	R/W-0	R/W-0	r-0	U-0	R/W-0	U-0	U-0
VCFG2	VCFG1	VCFG0	r	—	CSCNA	—	—
bit 15							bit 8

R-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BUFS	—	SMPI3	SMPI2	SMPI1	SMPI0	BUFM	ALTS
bit 7							bit 0

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

bit 15-13 **VCFG<2:0>**：参考电压配置位

VCFG<2:0>	VR+	VR-
000	AVDD	AVSS
001	外部 VREF+ 引脚	AVSS
010	AVDD	外部 VREF- 引脚
011	外部 VREF+ 引脚	外部 VREF- 引脚
1xx	AVDD	AVSS

bit 12 **保留**：保持为 0

bit 11 **未实现**：读为 0

bit 10 **CSCNA**：MUX A 输入多路开关选择的 CH0+ S/H 输入的扫描输入设置位
1 = 扫描输入
0 = 不扫描输入

bit 9-8 **未实现**：读为 0

bit 7 **BUFS**：缓冲区填充状态位 （仅当 BUFM = 1 时有效）
1 = A/D 当前正在填充缓冲区 08-0F ； 用户应访问 00-07 中的数据
0 = A/D 当前正在填充缓冲区 00-07 ； 用户应访问 08-0F 中的数据

bit 6 **未实现**：读为 0

bit 5-2 **SMPI<3:0>**：选择每次中断的采样 / 转换序列数的位
1111 = 每完成 16 个采样 / 转换序列时产生中断
1110 = 每完成 15 个采样 / 转换序列时产生中断
.....
0001 = 每完成 2 个采样 / 转换序列时产生中断
0000 = 每完成 1 个采样 / 转换序列时产生中断

bit 1 **BUFM**：缓冲区模式选择位
1 = 缓冲区配置为两个 8 字缓冲区 （ADC1BUFn<15:8> 和 ADC1BUFn<7:0>）
0 = 缓冲区配置为一个 16 字缓冲区 （ADC1BUFn<15:0>）

bit 0 **ALTS**：交替输入采样模式选择位
1 = 对于第一次采样，使用 MUX A 输入多路开关设置，然后对于所有后续采样，在 MUX B 和 MUX A 输入多路开关设置之间交替
0 = 始终使用 MUX A 输入多路开关设置

寄存器 21-3： AD1CON3：A/D 控制寄存器 3

R/W-0	r-0	r-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADRC	r	r	SAMC4	SAMC3	SAMC2	SAMC1	SAMC0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADCS7	ADCS6	ADCS5	ADCS4	ADCS3	ADCS2	ADCS1	ADCS0
bit 7							bit 0

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

bit 15 **ADRC**：A/D 转换时钟源位
1 = A/D 内部 RC 时钟
0 = 时钟由系统时钟产生

bit 14-13 **保留**：保持为 0

bit 12-8 **SAMC<4:0>**：自动采样时间位
11111 = 31 TAD
. . . . .
00001 = 1 TAD
00000 = 0 TAD （不推荐）

bit 7-0 **ADCS<7:0>**：A/D 转换时钟选择位
11111111 至 01000000 = 保留
. . . . .
00111111 = 64 • TCY
. . . . .
00000001 = 2 • TCY
00000000 = TCY

寄存器 21-4： AD1CHS：A/D 输入选择寄存器

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CH0NB	—	—	CH0SB4(1,2)	CH0SB3(1,2)	CH0SB2(1,2)	CH0SB1(1,2)	CH0SB0(1,2)
bit 15							bit 8

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CH0NA	—	—	CH0SA4	CH0SA3	CH0SA2	CH0SA1	CH0SA0
bit 7							bit 0

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

bit 15 **CH0NB**：MUX B 多路开关设置的通道 0 的反相端输入选择位
1 = 通道 0 的反相端输入为 AN1
0 = 通道 0 的反相端输入为 VR-

bit 14-13 **未实现**：读为 0

bit 12-8 **CH0SB<4:0>**：MUX B 多路开关设置的通道 0 的同相端输入选择位 (1,2)
11111 = 通道 0 的同相端输入保留为仅用于 CTMU(3)
1xxxx = 未实现；不要使用
01111 = 通道 0 的同相端输入为内部带隙参考 （VBG）
01110 = 通道 0 的同相端输入为 VBG/2
01101 = 通道 0 的同相端输入为稳压器输出 （VDDCORE）
01100 = 通道 0 的同相端输入为 AN12
01011 = 通道 0 的同相端输入为 AN11
01010 = 通道 0 的同相端输入为 AN10
01001 = 通道 0 的同相端输入为 AN9
01000 = 通道 0 的同相端输入为 AN8
00111 = 通道 0 的同相端输入为 AN7
00110 = 通道 0 的同相端输入为 AN6
00101 = 通道 0 的同相端输入为 AN5
00100 = 通道 0 的同相端输入为 AN4
00011 = 通道 0 的同相端输入为 AN3
00010 = 通道 0 的同相端输入为 AN2
00001 = 通道 0 的同相端输入为 AN1
00000 = 通道 0 的同相端输入为 AN0

bit 7 **CH0NA**：MUX A 多路开关设置的通道 0 的反相端输入选择位
1 = 通道 0 的反相端输入为 AN1
0 = 通道 0 的反相端输入为 VR-

bit 6-5 **未实现**：读为 0

bit 4-0 **CH0SA<4:0>**：MUX A 多路开关设置的通道 0 的同相端输入选择位
实现的组合与 CH0SB<4:0> （见上）相同。

注 1： 未在此处显示的组合未实现；不要使用。
2： 模拟通道 AN6、 AN7、 AN8 和 AN12 在 28 引脚器件上不可用；不要使用。
3： 通过选择该内部通道，使 CTMU 模块可以利用 A/D 转换器的采样和保持电容（CAD）来实现最小时间测量。

寄存器 21-5： AD1PCFG：A/D 端口配置寄存器

R/W-0	R/W-0	R/W-0	R/W-0(1)	R/W-0	R/W-0	R/W-0	R/W-0(1)
PCFG15	PCFG14	PCFG13	PCFG12	PCFG11	PCFG10	PCFG9	PCFG8
bit 15							bit 8

R/W-0(1)	R/W-0(1)	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PCFG7	PCFG6	PCFG5	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

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

bit 15 **PCFG15**：A/D 输入带隙参考使能位
1 = 禁止内部带隙 （V_{BG}）参考通道
0 = 使能内部带隙参考通道

bit 14 **PCFG14**：A/D 输入半带隙参考使能位
1 = 禁止内部半带隙 （$V_{BG}/2$）参考通道
0 = 使能内部半带隙参考通道

bit 13 **PCFG13**：A/D 输入稳压器输出参考使能位
1 = 禁止内部稳压器输出 （V_{DDCORE}）参考通道
0 = 使能内部稳压器输出参考通道

bit 12-0 **PCFG<12:0>**：模拟输入引脚配置控制位 (1)
1 = 对应模拟通道的引脚配置为数字模式；使能 I/O 端口读操作
0 = 引脚配置为模拟模式；禁止 I/O 端口读操作， A/D 采样引脚电压

注 1： 模拟通道 AN6、 AN7、 AN8 和 AN12 在 28 引脚器件上不可用；保留相应的位置 1。

寄存器 21-6： AD1CSSL：A/D 输入扫描选择寄存器

R/W-0	R/W-0	R/W-0	R/W-0(1)	R/W-0	R/W-0	R/W-0	R/W-0
CSSL15	CSSL14	CSSL13	CSSL12	CSSL11	CSSL10	CSSL9	CSSL8(1)
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CSSL7	CSSL6	CSSL5	CSSL4	CSSL3	CSSL2	CSSL1	CSSL0
bit 7							bit 0

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

bit 15 **CSSL15**：A/D 输入带隙扫描使能位
1 = 使能对内部带隙 （VBG）通道进行输入扫描
0 = 禁止对模拟通道进行输入扫描

bit 14 **CSSL14**：A/D 输入半带隙扫描使能位
1 = 使能对内部半带隙 （VBG/2）通道进行输入扫描
0 = 禁止对模拟通道进行输入扫描

bit 13 **CSSL13**：A/D 输入稳压器输出扫描使能位
1 = 使能对内部稳压器输出 （VDDCORE）进行输入扫描
0 = 禁止对模拟通道进行输入扫描

bit 12-0 **CSSL<12:0>**：A/D 输入引脚扫描选择位 (1)
1 = 为输入扫描选择的对应模拟通道
0 = 被输入扫描忽略的模拟通道

注 1： 模拟通道 AN6、 AN7、 AN8 和 AN12 在 28 引脚器件上不可用；保留相应的位清零。

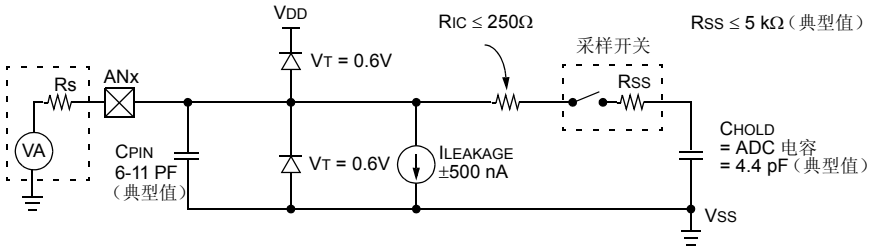
公式 21-1：　　A/D 转换时钟周期 (1)

$$ADCS = \frac{TAD}{TCY} - 1$$

$$TAD = TCY \bullet (ADCS + 1)$$

注　1： 基于 TCY = 2 * TOSC，打盹模式和 PLL 被禁止。

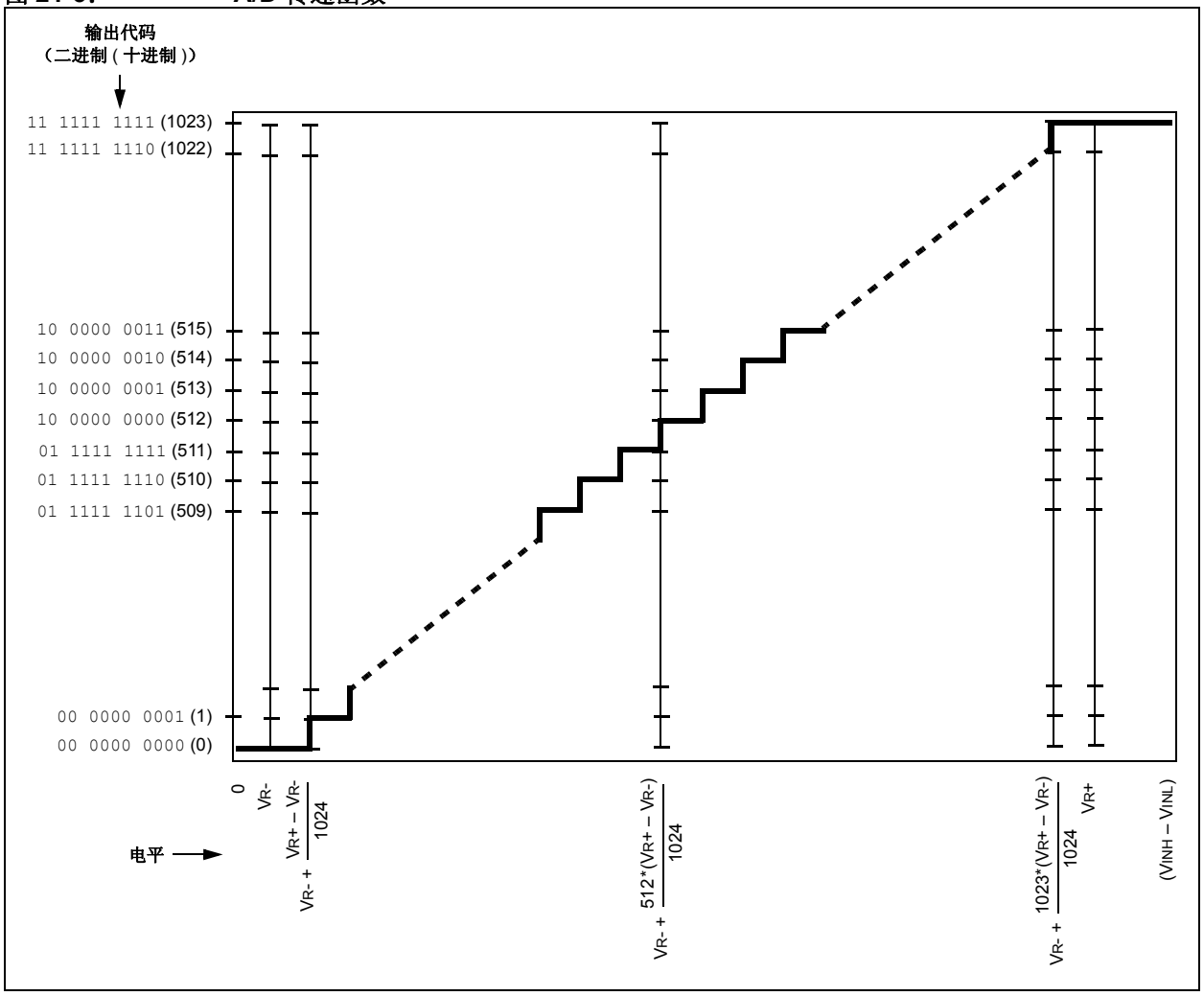
图 21-2：　　10 位 A/D 转换器模拟输入模型



图注：		
	CPIN	= 输入电容
	VT	= 门限电压
	ILEAKAGE	= 由各连接点在引脚上产生的泄漏电流
	RIC	= 片内走线等效电阻
	RSS	= 采样开关电阻
	CHOLD	= 采样 / 保持电容（来自 DAC）

注： CPIN 值取决于器件的封装，未经测试。如果 Rs ≤ 5 kΩ， CPIN 的影响可忽略。

图 21-3：　　A/D 传递函数



22.0 三比较器模块

注： 本数据手册总结了 PIC24F 器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 19 章“比较器模块”**（DS39710A_CN）。

三比较器模块提供了三个双输入的比较器。至比较器的输入可以配置为使用 4 个外部模拟输入之一，而且参考电压输入可以来自参考电压发生器或带隙参考。

比较器输出可能直接连接到 CxOUT 引脚。当相应的 COE 等于 1 时，I/O 引脚逻辑使比较器的不同步输出能够在引脚上使用。

图 22-1 给出了该模块的简化框图。图 22-2 给出了各种可能的比较器配置的框图。

每个比较器都有自己的控制寄存器 CMxCON（寄存器 22-1），用于使能和配置其操作。CMSTAT 寄存器（寄存器 22-2）提供了所有 3 个比较器的输出和事件状态。

图 22-1： 三比较器模块框图

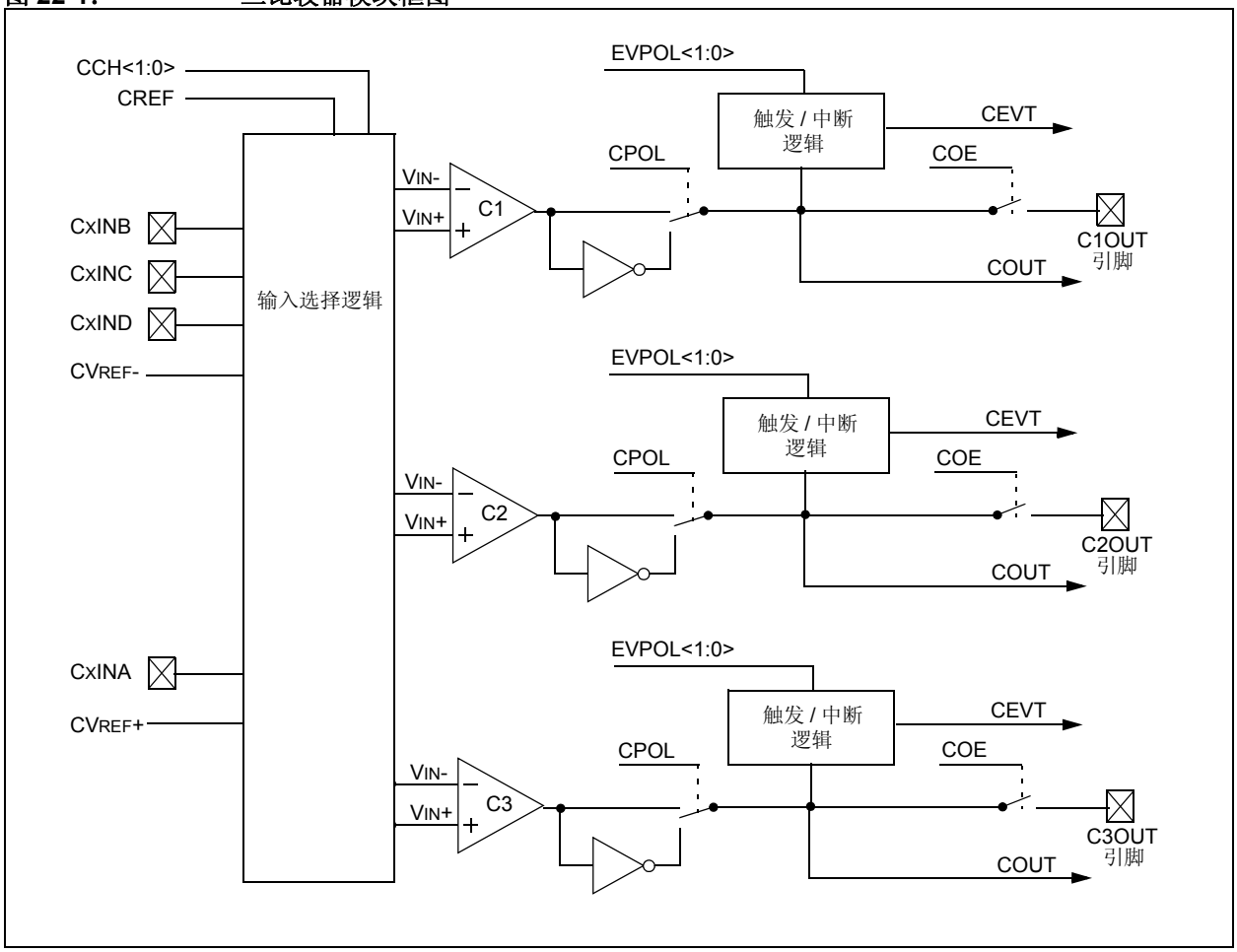
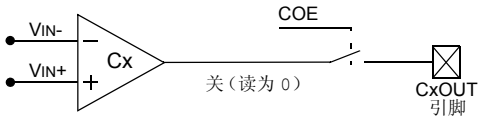
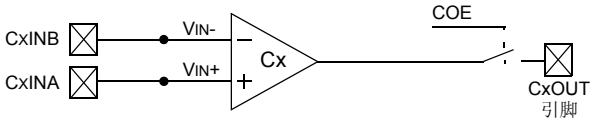


图 22-2：　　各种比较器配置

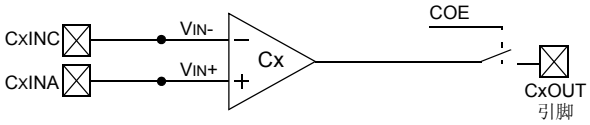
比较器关闭
CEN = 0，CREF = x，CCH<1:0> = xx



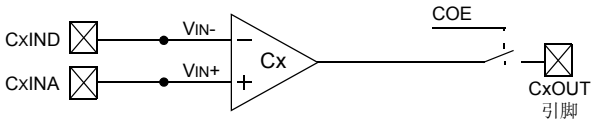
比较器 CxINB > CxINA 比较
CEN = 1，CREF = 0，CCH<1:0> = 00



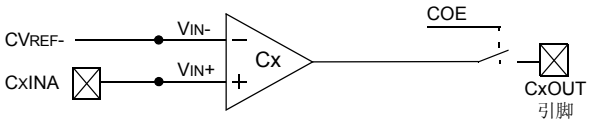
比较器 CxINC > CxINA 比较
CEN = 1，CREF = 0，CCH<1:0> = 01



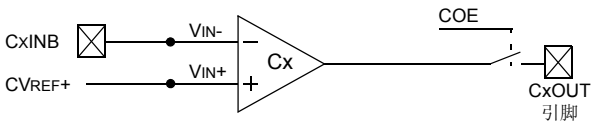
比较器 CxIND > CxINA 比较
CEN = 1，CREF = 0，CCH<1:0> = 10



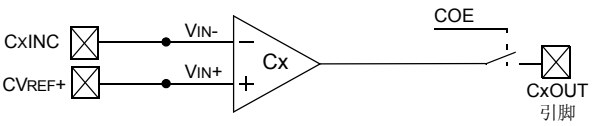
比较器 CVREF- > CxINA 比较
CEN = 1，CREF = 0，CCH<1:0> = 11



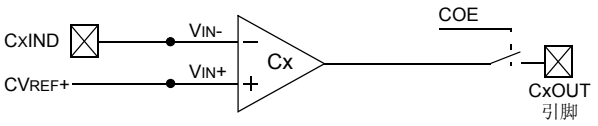
比较器 CxINB > CVREF+ 比较
CEN = 1，CREF = 1，CCH<1:0> = 00



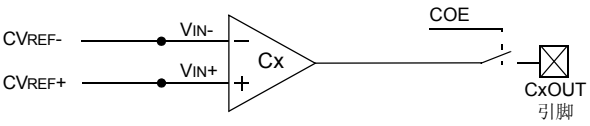
比较器 CxINC > CVREF+ 比较
CEN = 1，CREF = 1，CCH<1:0> = 01



比较器 CxIND > CVREF+ 比较
CEN = 1，CREF = 1，CCH<1:0> = 10



比较器 CVREF- > CVREF+ 比较
CEN = 1，CREF = 1，CCH<1:0> = 11



寄存器 22-1：　CMxCON：比较器 x 控制寄存器 （比较器 1 至 3）

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R-0
CEN	COE	CPOL	—	—	—	CEVT	COUT
bit 15							bit 8

R/W-0	R/W-0	U-0	R/W-0	U-0	U-0	R/W-0	R/W-0
EVPOL1	EVPOL0	—	CREF	—	—	CCH1	CCH0
bit 7							bit 0

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

bit 15　**CEN：**比较器使能位
1 = 使能比较器
0 = 禁止比较器

bit 14　**COE：**比较器输出使能位
1 = 比较器输出出现在 CxOUT 引脚上
0 = 比较器输出仅在内部有效

bit 13　**CPOL：**比较器输出极性选择位
1 = 比较器输出反相
0 = 比较器输出不反相

bit 12-10　**未实现：**读为 0

bit 9　**CEVT：**比较器事件位
1 = 发生了由 EVPOL<1:0> 定义的比较器事件；后续的触发和中断被禁止，直到该位被清零
0 = 未发生比较器事件

bit 8　**COUT：**比较器输出位
当 CPOL = 0 时：
1 = VIN+ > VIN-
0 = VIN+ < VIN-
当 CPOL = 1 时：
1 = VIN+ < VIN-
0 = VIN+ > VIN-

bit 7-6　**EVPOL<1:0>：**触发 / 事件 / 中断极性选择位
11 = 在比较器输出发生任何变化时产生触发 / 事件 / 中断 （当 CEVT = 0 时）
10 = 在比较器输出发生跳变时产生触发 / 事件 / 中断：
　如果 CPOL = 0 （极性不反相）：
　仅限从高至低跳变。
　如果 CPOL = 1 （极性反相）：
　仅限从低至高跳变。
01 = 在比较器输出发生跳变时产生触发 / 事件 / 中断：
　如果 CPOL = 0 （极性不反相）：
　仅限从低至高跳变。
　如果 CPOL = 1 （极性反相）：
　仅限从高至低跳变。
00 = 禁止产生触发 / 事件 / 中断

bit 5　**未实现：**读为 0

寄存器 22-1： CMxCON：比较器 x 控制寄存器 （比较器 1 至 3）（续）

bit 4 **CREF：**比较器参考电压选择位 （同相输入）
1 = 同相输入端连接到内部 CVREF+ 输入参考电压
0 = 同相输入端连接到 CxINA 引脚

bit 3-2 **未实现：**读为 0

bit 1-0 **CCH<1:0>：**比较器通道选择位
11 = 比较器的反相输入端连接到 CVREF- 输入参考电压
10 = 比较器的反相输入端连接到 CxIND 引脚
01 = 比较器的反相输入端连接到 CxINC 引脚
00 = 比较器的反相输入端连接到 CxINB 引脚

寄存器 22-2： CMSTAT：比较器模块状态寄存器

R/W-0	U-0	U-0	U-0	U-0	R-0	R-0	R-0
CMIDL	—	—	—	—	C3EVT	C2EVT	C1EVT
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	R-0	R-0	R-0
—	—	—	—	—	C3OUT	C2OUT	C1OUT
bit 7							bit 0

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

bit 15 **CMIDL：**比较器空闲模式停止位
1 = 器件进入空闲模式后所有比较器停止工作
0 = 在空闲模式下所有已使能的比较器继续工作

bit 14-11 **未实现：**读为 0

bit 10 **C3EVT：**比较器 3 事件状态位 （只读）
显示比较器 3 的当前事件状态 （CM3CON<9>）。

bit 9 **C2EVT：**比较器 2 事件状态位 （只读）
显示比较器 2 的当前事件状态 （CM2CON<9>）。

bit 8 **C1EVT：**比较器 1 事件状态位 （只读）
显示比较器 1 的当前事件状态 （CM1CON<9>）。

bit 7-3 **未实现：**读为 0

bit 2 **C3OUT：**比较器 3 输出状态位 （只读）
显示比较器 3 的当前输出状态 （CM3CON<8>）。

bit 1 **C2OUT：**比较器 2 输出状态位 （只读）
显示比较器 2 的当前输出状态 （CM2CON<8>）。

bit 0 **C1OUT：**比较器 1 输出状态位 （只读）
显示比较器 1 的当前输出状态 （CM1CON<8>）。

23.0 比较器参考电压

注：	本数据手册总结了 PIC24F 系列器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 20 章 “比较器参考电压模块”**（DS39709A_CN）。

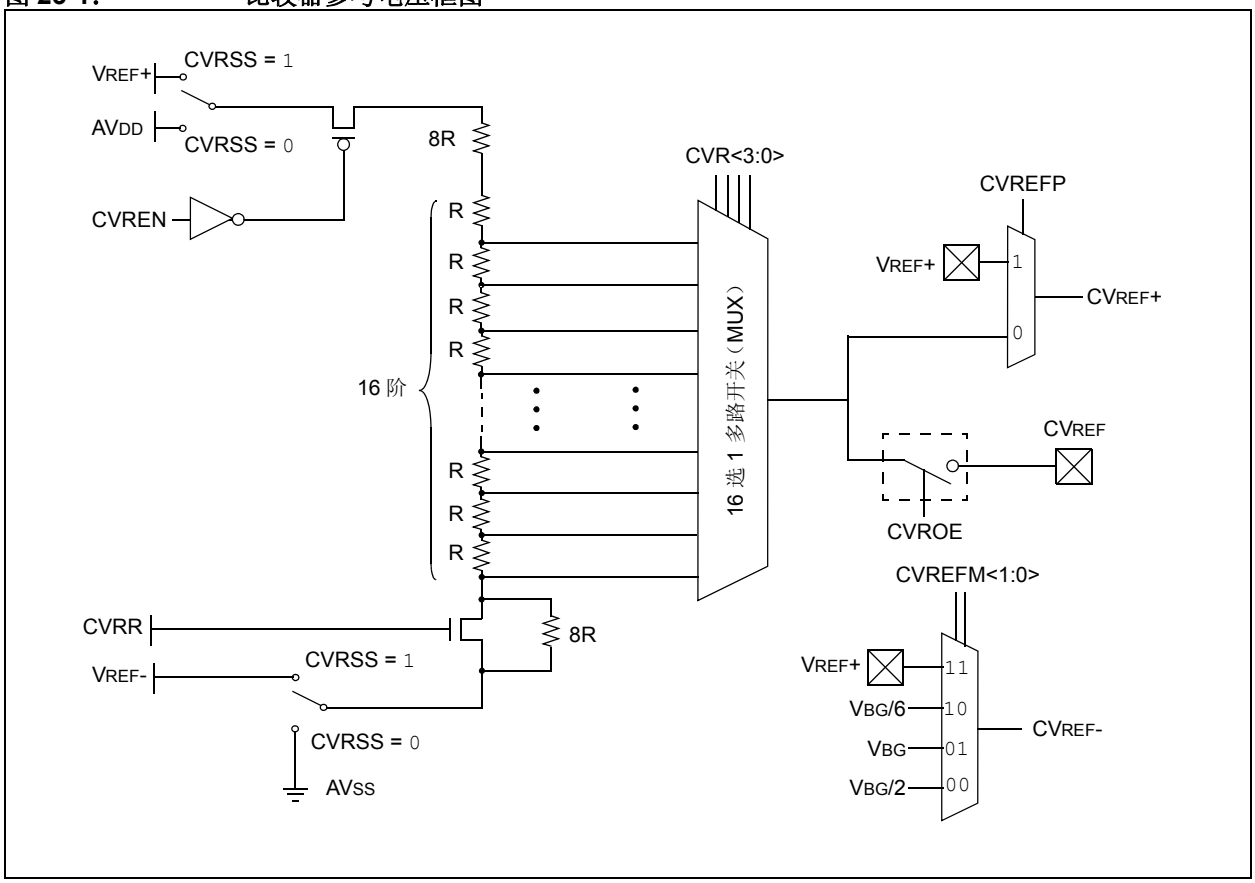
23.1 配置比较器参考电压

参考电压模块是通过 CVRCON 寄存器（寄存器 23-1）来控制的。比较器参考电压模块提供两种范围的输出电压，每种范围都具有 16 个不同的电压值。通过 CVRR 位（CVRCON<5>）选择输出电压的范围。这两种范围的主要区别在于其电压值之间的步长不同（其中一种范围可提供较高的分辨率），该步长通过 CVREF 选择位（CVR<3:0>）选择。

比较器参考电压模块的电压源可以来自 VDD 和 VSS，也可以来自外部 VREF+ 和 VREF-。电压源通过 CVRSS 位（CVRCON<4>）选择。

在更改 CVREF 输出值时，必须考虑比较器参考电压的稳定时间。

图 23-1： 比较器参考电压框图



寄存器 23-1：　　CVRCON：比较器参考电压控制寄存器

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
—	—	—	—	—	CVREFP	CVREFM1	CVREFM0
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0
bit 7							bit 0

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

bit 15-11　**未实现：**读为 0

bit 10　**CVREFP：**CVREF+ 参考输出选择位
1 = 使用 VREF+ 输入引脚作为比较器的 CVREF+ 参考输出
0 = 使用比较器参考电压模块产生的输出作为比较器的 CVREF+ 参考输出

bit 9-8　**CVREFM<1:0>：**CVREF- 参考输出选择位
11 = 使用 VREF+ 输入引脚作为比较器的 CVREF- 参考输出
10 = 使用 VBG/6 作为比较器的 CVREF- 参考输出
01 = 使用 VBG 作为比较器的 CVREF- 参考输出
00 = 使用 VBG/2 作为比较器的 CVREF- 参考输出

bit 7　**CVREN：**比较器参考电压使能位
1 = CVREF 电路上电
0 = CVREF 电路断电

bit 6　**CVROE：**比较器 VREF 输出使能位
1 = CVREF 电压从 CVREF 引脚输出
0 = CVREF 电压从 CVREF 引脚断开

bit 5　**CVRR：**比较器 VREF 范围选择位
1 = CVRSRC 范围应从 0 到 0.625 CVRSRC，步长为 CVRSRC/24
0 = CVRSRC 范围应从 0.25 到 0.719 CVRSRC，步长为 CVRSRC/32

bit 4　**CVRSS：**比较器 VREF 源选择位
1 = 比较器参考电压源， CVRSRC = VREF+ – VREF-
0 = 比较器参考电压源， CVRSRC = AVDD – AVSS

bit 3-0　**CVR<3:0>：**比较器 VREF 值选择 （0 ≤ CVR<3:0> ≤ 15）位
当 CVRR = 1 时：
CVREF = (CVR<3:0>/24) • (CVRSRC)
当 CVRR = 0 时：
CVREF = 1/4 • (CVRSRC) + (CVR<3:0>/32) • (CVRSRC)

24.0 充电时间测量单元（CTMU）

注： 本数据手册总结了 PIC24F 器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的**第 11 章“充电时间测量单元（CTMU）”**（DS39724A_CN）。

充电时间测量单元是一个灵活的模拟模块，它提供脉冲源之间的精确时间差测量，以及异步脉冲生成。它的主要特性包括：

- 4 个边沿输入触发源
- 每个边沿源的极性控制
- 边沿顺序控制
- 控制对边沿的响应
- 1 纳秒的时间测量分辨率
- 适合测量电容的精确电流源

CTMU 可与其他片上模拟模块一起，用于精确测量时间、电容以及电容的相对变化，或生成独立于系统时钟的输出脉冲。CTMU 模块是与电容式传感器接口的理想选择。

CTMU 通过两个寄存器 CTMUCON 和 CTMUICON 进行控制。CTMUCON 使能该模块，并控制边沿源选择、边沿源极性选择和边沿顺序。 CTMUICON 寄存器控制电流源的选择和微调。

24.1 测量电容

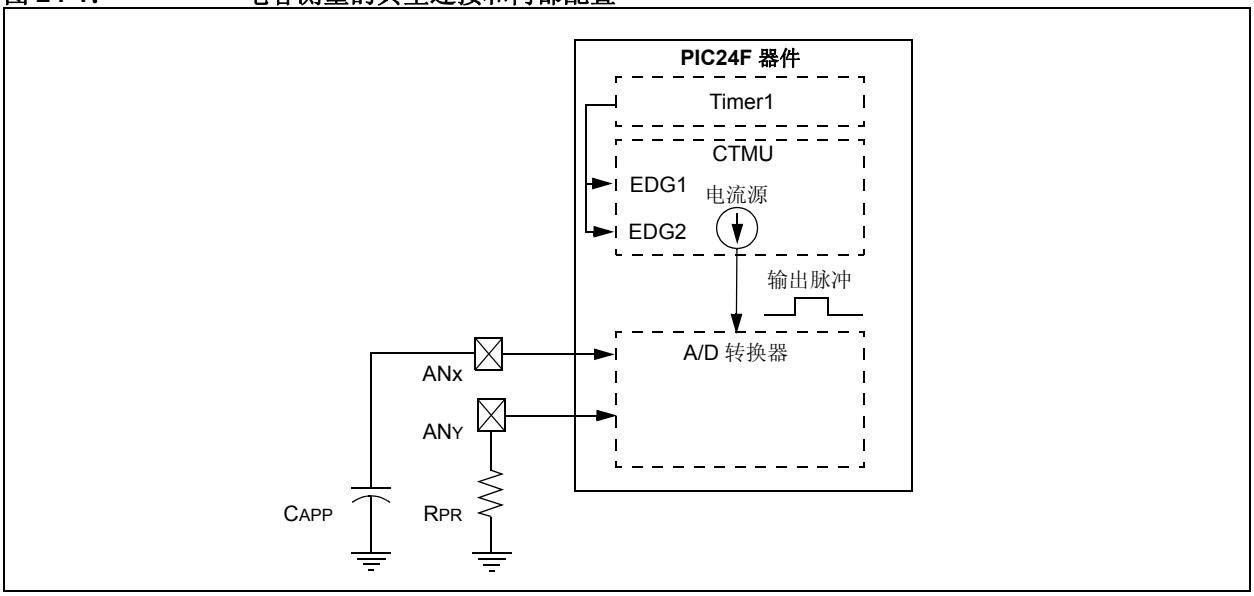
CTMU 模块通过生成宽度和两个独立输入通道上的边沿事件间的时间差相等的输出脉冲，来测量电容。两个输入通道的脉冲边沿事件都可以从 4 个来源选择：两个内部外设模块（OC1 和 Timer1）和两个外部引脚（CTEDG1 和 CTEDG2）。该脉冲和该模块的精确电流源一起使用，可根据以下关系计算电容：

$$i = C \bullet \frac{dV}{dT}$$

测量电容时，A/D 转换器在 CTMU 输出脉冲结束后对其某个输入通道上的外部电容（CAPP）进行采样。一个高精度电阻（RPR）对第二个 A/D 通道上的电流源进行校准。脉冲结束后，转换器将测量电容上的电压。电容的实际计算在软件中由应用程序执行。

图 24-1 显示了用于电容测量的外部连接，以及该应用中 CTMU 和 A/D 模块关系如何。该示例还显示了来自 Timer1 的边沿事件，但也可能存在使用外部边沿源的其他配置。用 CTMU 模块测量电容和时间的详细讨论在《PIC24F 系列参考手册》中提供。

图 24-1： 电容测量的典型连接和内部配置



24.2 测量时间

对脉冲宽度的时间测量也可类似地执行，需要使用 A/D 模块的内部电容（CAD）和用于校准电流的高精度电阻。图 24-2 显示了用于时间测量的外部连接，以及该应用中 CTMU 和 A/D 模块关系如何。该示例还显示了来自外部 CTEDG 引脚的两个边沿事件，但也可能存在使用内部边沿源的其他配置。为实现最小时间测量，可选择内部 A/D 通道 31，CH0Sx <4:0>= 11111。这可以最大程度降低其他情况下使用输入引脚时具有的杂散电容，从而将总电容保持为 A/D 转换器自身的电容 （4-5 pF）。用 CTMU 模块测量电容和时间的详细讨论在《PIC24F 系列参考手册》中提供。

24.3 脉冲产生和延时

CTMU 模块也可产生边沿与器件的系统时钟异步的输出脉冲。更明确地说，它可以产生距离模块边沿事件输入有可编程延时的脉冲。

通过将 TGEN 位 （CTMUCON<12>）置 1 配置该模块在产生脉冲前应用延时时，内部电流源连接到比较器 2 的 B 输入。电容 （CDELAY）连接到比较器 2 的引脚 C2INB，比较器参考电压 CVREF 连接到 C2INA 将其配置为特定跳变点。检测到边沿事件时，该模块开始对 CDELAY 充电。 CDELAY 充电达到 CVREF 跳变点时，在 CTPLS 上输出脉冲。脉冲延时的长度由 CDELAY 和 CVREF 跳变点的值决定。

图 24-3 显示了脉冲产生的外部连接，以及所需的不同模拟模块间的关系。CTEDG1 显示为输入脉冲源时，其他选项可用。用 CTMU 模块产生脉冲的详细讨论在《PIC24F 系列参考手册》中提供。

图 24-2： 时间测量的典型连接和内部配置

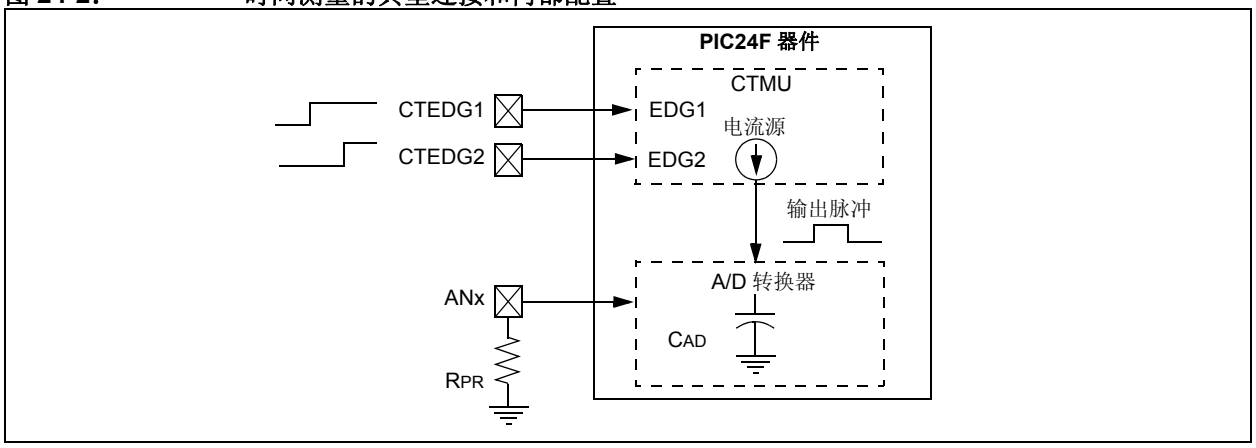
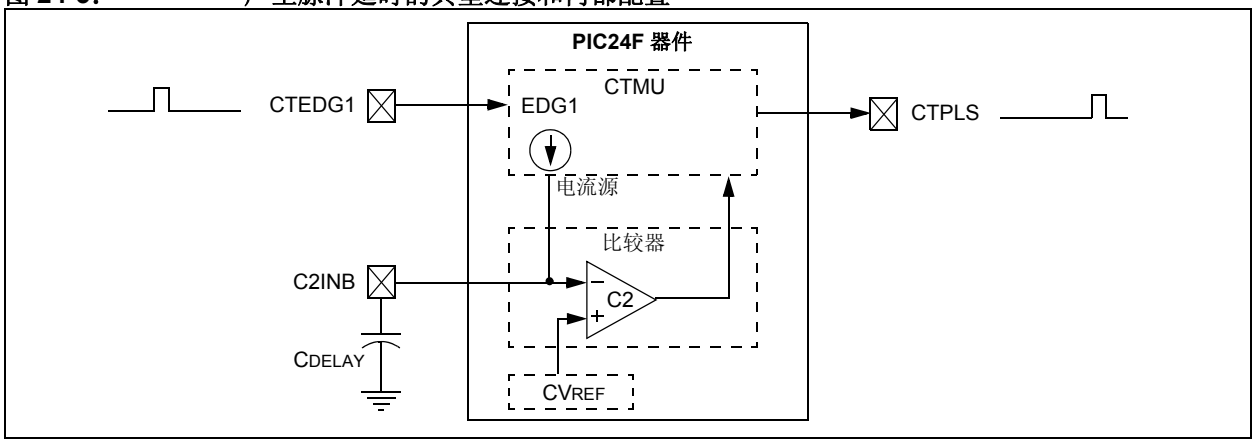


图 24-3： 产生脉冲延时的典型连接和内部配置



寄存器 24-1： CTMUCON：CTMU 控制寄存器

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CTMUEN	—	CTMUSIDL	TGEN(1)	EDGEN	EDGSEQEN	IDISSEN	CTTRIG
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
EDG2POL	EDG2SEL1	EDG2SEL0	EDG1POL	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT
bit 7							bit 0

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

bit 15 **CTMUEN**：CTMU 使能位
1 = 使能模块
0 = 禁止模块

bit 14 **未实现**：读为 0

bit 13 **CTMUSIDL**：空闲模式停止位
1 = 当器件进入空闲模式时，模块停止工作
0 = 在空闲模式下模块继续工作

bit 12 **TGEN**：延时产生使能位 (1)
1 = 使能边沿延时产生
0 = 禁止边沿延时产生

bit 11 **EDGEN**：边沿使能位
1 = 未阻止边沿
0 = 阻止边沿

bit 10 **EDGSEQEN**：边沿顺序使能位
1 = 边沿 1 事件必须在边沿 2 事件发生前发生
0 = 无需边沿顺序

bit 9 **IDISSEN**：模拟电流源控制位
1 = 模拟电流源输出接地
0 = 模拟电流源输出未接地

bit 8 **CTTRIG**：触发器控制位
1 = 使能触发器输出
0 = 禁止触发器输出

bit 7 **EDG2POL**：边沿 2 极性选择位
1 = 边沿 2 设定为正边沿响应
0 = 边沿 2 设定为负边沿响应

bit 6-5 **EDG2SEL<1:0>**：边沿 2 源选择位
11 = CTED1 引脚
10 = CTED2 引脚
01 = OC1 模块
00 = Timer1 模块

bit 4 **EDG1POL**：边沿 1 极性选择位
1 = 边沿 1 设定为正边沿响应
0 = 边沿 1 设定为负边沿响应

注 1： 如果 TGEN = 1，外设输入和输出必须配置给可用的 RPn 引脚。更多信息，请参见**第 10.4 节“外设引脚选择（PPS）”**。

寄存器 24-1： CTMUCON：CTMU 控制寄存器 （续）

bit 3-2 **EDG1SEL<1:0>**：边沿 1 源选择位
11 = CTED1 引脚
10 = CTED2 引脚
01 = OC1 模块
00 = Timer1 模块

bit 1 **EDG2STAT**：边沿 2 状态位
1 = 已发生边沿 2 事件
0 = 未发生边沿 2 事件

bit 0 **EDG1STAT**：边沿 1 状态位
1 = 已发生边沿 1 事件
0 = 未发生边沿 1 事件

注 1： 如果 TGEN = 1，外设输入和输出必须配置给可用的 RPn 引脚。更多信息，请参见**第 10.4 节 “外设引脚选择 （PPS）”**。

寄存器 24-2： CTMUICON：CTMU 电流控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ITRIM5	ITRIM4	ITRIM3	ITRIM2	ITRIM1	ITRIM0	IRNG1	IRNG0
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

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

bit 15-10 **ITRIM<5:0>**：电流源微调位
011111 = 对标称电流的最大正向调整
011110
.....
000001 = 对标称电流的最小正向调整
000000 = IRNG<1:0> 指定的标称电流输出
111111 = 对标称电流的最小负向调整
.....
100010
100001 = 对标称电流的最大负向调整

bit 9-8 **IRNG<1:0>**：电流源范围选择位
11 = 100 × 基本电流
10 = 10 × 基本电流
01 = 基本电流 （标称值为 0.55 μA）
00 = 禁止电流源

bit 7-0 **未实现**：读为 0

25.0 特殊功能

注： 本数据手册总结了 PIC24F 器件的特性。但是不应把本手册当作无所不包的参考手册来使用。更多信息，请参见《PIC24F 系列参考手册》的以下章节：

- **第 9 章 “看门狗定时器 （WDT）”**（DS39697A_CN）
- **第 32 章 “高级器件集成”**（DS39719A_CN）
- **第 33 章 “编程和诊断”**（DS39716A_CN）

PIC24FJ64GA104 系列器件具有几项特殊的功能旨在最大限度地提高应用的灵活性和可靠性，并通过减少外部元件的使用将成本降至最低。提供的特殊功能包括：

- 灵活的配置
- 看门狗定时器 （WDT）
- 代码保护
- JTAG 边界扫描接口
- 在线串行编程
- 在线仿真

25.1 配置位

可以通过对配置位编程（读为 0）或不编程（读为 1）来选择不同的器件配置。这些配置位被映射到程序存储器以 F80000h 开始的单元中。从寄存器 25-1 到寄存器 25-6 详细说明了各配置位的功能。

注意，地址F80000h超出了用户程序存储空间。事实上，它属于只能使用表读和表写访问的配置存储空间（800000h-FFFFFFh）。

25.1.1 配置 PIC24FJ64GA104 系列器件的注意事项

在PIC24FJ64GA104 系列器件中，配置字节以易失性存储方式实现。这就意味着在器件每次上电时都必须对配置数据进行编程。配置数据存储在片上程序存储空间顶部的 3 个字中，这些字被称为闪存配置字。表 25-1 显示了它们的具体位置。这些是实际器件配置位的紧凑表现形式，这些配置位实际分布在配置空间中的多个位置中。器件复位期间，配置数据会自动从闪存配置字装入到相应的配置寄存器中。

注： 所有类型的器件复位都会重新装入配置数据。

当为这些器件创建应用程序时，用户应该总是为配置数据特别分配闪存配置字地址，以确保当编译代码时程序代码不会存储到该地址。

程序存储器中的所有闪存配置字的高字节应该总为 1111 1111。这使得当远程事件意外执行这些存储单元时将其作为 NOP 指令来执行。由于没有在相应的存储单元中实现这些配置位，因此向这些存储单元写入 1 不会影响器件工作。

注： 在程序存储器的最后页上执行页擦除操作会清零闪存配置字，从而使能代码保护。因此，用户应避免在程序存储器的最后页上执行页擦除操作。

表 25-1： PIC24FJ64GA104 系列器件的闪存配置字位置

器件	配置字地址			
	1	2	3	4
PIC24FJXXGA102	57FEh	57FCh	57FAh	57F8h
PIC24FJXXGA104	ABFEh	ABFCh	ABFAh	ABF8h

寄存器 25-1：　　CW1：闪存配置字 1

U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1
—	—	—	—	—	—	—	—
bit 23							bit 16

r-x	R/PO-1	R/PO-1	R/PO-1	R/PO-1	U-1	R/PO-1	R/PO-1
r	JTAGEN(1)	GCP	GWRP	$\overline{\text{DEBUG}}$	—	ICS1	ICS0
bit 15							bit 8

R/PO-1	R/PO-1	U-1	R/PO-1	R/PO-1	R/PO-1	R/PO-1	R/PO-1
FWDTEN	WINDIS	—	FWPSA	WDTPS3	WDTPS2	WDTPS1	WDTPS0
bit 7							bit 0

图注：	r = 保留位		
R = 可读位	PO = 一次编程位	U = 未实现位，读为 0	
-n = 未对器件编程时的值		1 = 置 1	0 = 清零

bit 23-16　**未实现：**读为 1

bit 15　**保留：**未知值；编程为 0

bit 14　**JTAGEN：**JTAG 端口使能位 (1)
1 = 使能 JTAG 端口
0 = 禁止 JTAG 端口

bit 13　**GCP：**通用段程序存储器代码保护位
1 = 禁止代码保护
0 = 对整个程序存储空间使能代码保护

bit 12　**GWRP：**通用段代码闪存写保护位
1 = 允许写程序存储器
0 = 禁止写程序存储器

bit 11　**$\overline{\text{DEBUG}}$：**后台调试器使能位
1 = 器件复位到工作模式
0 = 器件复位到调试模式

bit 10　**未实现：**读为 1

bit 9-8　**ICS<1:0>：**仿真器引脚位置选择位
11 = 仿真器功能与 PGEC1/PGED1 共用
10 = 仿真器功能与 PGEC2/PGED2 共用
01 = 仿真器功能与 PGEC3/PGED3 共用
00 = 保留；不要使用

bit 7　**FWDTEN：**看门狗定时器使能位
1 = 使能看门狗定时器
0 = 禁止看门狗定时器

bit 6　**WINDIS：**窗口看门狗定时器禁止位
1 = 使能标准看门狗定时器
0 = 使能窗口看门狗定时器； FWDTEN 必须为 1

bit 5　**未实现：**读为 1

bit 4　**FWPSA：**WDT 预分频比选择位
1 = 预分频比为 1:128
0 = 预分频比为 1:32

注　1： 仅能使用在线串行编程（ICSP™）修改 JTAGEN 位。在通过 JTAG 接口对器件进行编程时，不能修改该位。

寄存器 25-1：　　CW1：闪存配置字 1 （续）

bit 3-0　　**WDTPS<3:0>**：看门狗定时器后分频比选择位
1111 = 1:32,768
1110 = 1:16,384
1101 = 1:8,192
1100 = 1:4,096
1011 = 1:2,048
1010 = 1:1,024
1001 = 1:512
1000 = 1:256
0111 = 1:128
0110 = 1:64
0101 = 1:32
0100 = 1:16
0011 = 1:8
0010 = 1:4
0001 = 1:2
0000 = 1:1

注　　1： 仅能使用在线串行编程（ICSP™）修改 JTAGEN 位。在通过 JTAG 接口对器件进行编程时，不能修改该位。

寄存器 25-2： CW2：闪存配置字 2

U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1
—	—	—	—	—	—	—	—
bit 23							bit 16

R/PO-1	U-1	U-1	U-1	U-1	R/PO-1	R/PO-1	R/PO-1
IESO	—	—	—	—	FNOSC2	FNOSC1	FNOSC0
bit 15							bit 8

R/PO-1	R/PO-1	R/PO-1	R/PO-1	U-1	R/PO-1	R/PO-1	R/PO-1
FCKSM1	FCKSM0	OSCIOFCN	IOL1WAY	—	I2C1SEL	POSCMD1	POSCMD0
bit 7							bit 0

图注：		
R = 可读位	PO = 一次编程位	U = 未实现位，读为 0
-n = 未对器件编程时的值	1 = 置 1	0 = 清零

bit 23-16 **未实现：**读为 1

bit 15 **IESO：**内 / 外部切换位
1 = 使能 IESO 模式 （双速启动）
0 = 禁止 IESO 模式 （双速启动）

bit 14-11 **未实现：**读为 1

bit 10-8 **FNOSC<2:0>：**初始振荡器选择位
111 = 带后分频器的快速 RC 振荡器 （FRCDIV）
110 = 保留
101 = 低功耗 RC 振荡器 （LPRC）
100 = 辅助振荡器 （SOSC）
011 = 带 PLL 模块的主振荡器 （XTPLL、 HSPLL 和 ECPLL）
010 = 主振荡器 （XT、 HS 和 EC）
001 = 带后分频器和 PLL 模块的快速 RC 振荡器 （FRCPLL）
000 = 快速 RC 振荡器 （FRC）

bit 7-6 **FCKSM<1:0>：**时钟切换和故障保护时钟监视器配置位
1x = 禁止时钟切换和故障保护时钟监视器
01 = 使能时钟切换，禁止故障保护时钟监视器
00 = 使能时钟切换和故障保护时钟监视器

bit 5 **OSCIOFCN：**OSCO 引脚配置位
如果 POSCMD<1:0> = 11 或 00：
1 = OSCO/CLKO/RA3 用作 CLKO （FOSC/2）
0 = OSCO/CLKO/RA3 用作端口 I/O （RC15）
如果 POSCMD<1:0> = 10 或 01：
OSCIOFCN 对 OSCO/CLKO/RA3 没有影响。

bit 4 **IOL1WAY：**IOLOCK 一次置 1 使能位
1 = IOLOCK 位 （OSCCON<6>）可以置 1 一次，前提是已经完成解锁序列。一旦被置 1，就不能再次写入外设引脚选择寄存器。
0 = 根据需要将 IOLOCK 位置 1 或清零，前提是已经完成解锁序列

bit 3 **未实现：**读为 1

bit 2 **I2C1SEL：**I2C1 引脚选择位
1 = 使用默认 SCL1/SDA1 引脚
0 = 使用备用 SCL1/SDA1 引脚

bit 1-0 **POSCMD<1:0>：**主振荡器配置位
11 = 禁止主振荡器
10 = 选择 HS 振荡器模式
01 = 选择 XT 振荡器模式
00 = 选择 EC 振荡器模式

寄存器 25-3： CW3：闪存配置字 3

U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1
—	—	—	—	—	—	—	—
bit 23							bit 16

R/PO-1	R/PO-1	R/PO-1	U-1	R/PO-1	R/PO-1	R/PO-1	R/PO-1
WPEND	WPCFG	WPDIS	—	WUTSEL1	WUTSEL0	SOSCSEL1(1)	SOSCSEL0(1)
bit 15							bit 8

U-1	U-1	R/PO-1	R/PO-1	R/PO-1	R/PO-1	R/PO-1	R/PO-1
—	—	WPFP5	WPFP4	WPFP3	WPFP2	WPFP1	WPFP0
bit 7							bit 0

图注：			
R = 可读位	PO = 一次编程位	U = 未实现位，读为 0	
-n = 未对器件编程时的值		1 = 置 1	0 = 清零

bit 23-16 **未实现：**读为 1

bit 15 **WPEND：**段写保护结束页选择位
1 = 受保护代码段的下边界位于程序存储器的底部（000000h）；上边界是由 WPFP<8:0> 指定的代码页
0 = 受保护代码段的上边界位于程序存储器的最后页；下边界是由 WPFP<8:0> 指定的代码页

bit 14 **WPCFG：**配置字代码页保护选择位
1 = 最后页 （位于程序存储器的顶部）和闪存配置字未受保护
0 = 最后页和闪存配置字受代码保护

bit 13 **WPDIS：**段写保护禁止位
1 = 禁止段代码保护
0 = 使能段代码保护；受保护的段由 WPEND、 WPCFG 和 WPFPx 配置位定义

bit 12 **未实现：**读为 1

bit 11-10 **WUTSEL<1:0>：**稳压器待机模式唤醒时间选择位
11 = 使用默认稳压器启动时间
01 = 使用快速稳压器启动时间
x0 = 保留；不要使用

bit 9-8 **SOSCSEL<1:0>：**辅助振荡器功耗模式选择位 (1)
11 = SOSC 引脚处于默认 （高驱动能力）振荡器模式
01 = SOSC 引脚处于低功耗 （低驱动能力）振荡器模式
00 = SOSC 引脚具有数字 I/O 功能 （RA4 和 RB4）；可以使用 SCLKI
10 = 保留

bit 7-6 **未实现：**读为 1

bit 5-0 **WPFP5:WPFP0：**受保护的代码段边界页位
指定受保护代码段的边界为 512 指令页，从程序存储器底部的页 9 开始。
如果 WPEND = 1：
指定代码页的结束地址为代码段的上边界。
如果 WPEND = 0：
指定代码页的起始地址为代码段的下边界。

注 1： SOSCI 和 SOSCO 引脚上的数字功能仅在配置为数字 I/O 模式 （00）时可用。

寄存器 25-4：　　CW4：闪存配置字 4

U-1	U-1	U-1	U-1	U-1	U-1	U-1	U-1
—	—	—	—	—	—	—	—
bit 23							bit 16

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

R/PO-1	R/PO-1	R/PO-1	R/PO-1	R/PO-1	R/PO-1	R/PO-1	R/PO-1
DSWDTEN	DSBOREN	RTCOSC	DSWDTOSC	DSWDTPS3	DSWDTPS2	DSWDTPS1	DSWDTPS0
bit 7							bit 0

图注：			
R = 可读位	PO = 一次编程位	U = 未实现位，读为 0	
-n = 未对器件编程时的值		1 = 置 1	0 = 清零

bit 23-8　**未实现：**读为 1

bit 7　**DSWDTEN：**深度休眠看门狗定时器使能位
1 = 使能 DSWDT
0 = 禁止 DSWDT

bit 6　**DSBOREN：**深度休眠 BOR 使能位
1 = 在深度休眠模式下使能 BOR
0 = 在深度休眠模式下禁止 BOR （不影响休眠模式）

bit 5　**RTCOSC：**RTCC 参考时钟选择位
1 = RTCC 使用 SOSC 作为参考时钟
0 = RTCC 使用 LPRC 作为参考时钟

bit 4　**DSWDTOSC：**DSWDT 参考时钟选择位
1 = DSWDT 使用 LPRC 作为参考时钟
0 = DSWDT 使用 SOSC 作为参考时钟

bit 3-0　**DSWDTPS<3:0>：**DSWDT 后分频比选择位
DSWDT 预分频比为 32；这产生一个约为 1 ms 的基本时间单位。
1111 = 1:2,147,483,648 （25.7 天）
1110 = 1:536,870,912 （6.4 天）
1101 = 1:134,217,728 （38.5 小时）
1100 = 1:33,554,432 （9.6 小时）
1011 = 1:8,388,608 （2.4 小时）
1010 = 1:2,097,152 （36 分钟）
1001 = 1:524,288 （9 分钟）
1000 = 1:131,072 （135 秒）
0111 = 1:32,768 （34 秒）
0110 = 1:8,192 （8.5 秒）
0101 = 1:2,048 （2.1 秒）
0100 = 1:512 （528 ms）
0011 = 1:128 （132 ms）
0010 = 1:32 （33 ms）
0001 = 1:8 （8.3 ms）
0000 = 1:2 （2.1 ms）

寄存器 25-5：　DEVID：器件 ID 寄存器

U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23							bit 16

R	R	R	R	R	R	R	R
FAMID7	FAMID6	FAMID5	FAMID4	FAMID3	FAMID2	FAMID1	FAMID0
bit 15							bit 8

R	R	R	R	R	R	R	R
DEV7	DEV6	DEV5	DEV4	DEV3	DEV2	DEV1	DEV0
bit 7							bit 0

图注：	R = 只读位	U = 未实现位

bit 23-16　**未实现：**读为 1

bit 15-8　**FAMID<7:0>：**器件系列标识符位
01000010 = PIC24FJ64GA104 系列

bit 7-0　**DEV<7:0>：**各个器件的标识符位
00000010 = PIC24FJ32GA102
00000110 = PIC24FJ64GA102
00001010 = PIC24FJ32GA104
00001110 = PIC24FJ64GA104

寄存器 25-6：　DEVREV：器件版本寄存器

U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 23							bit 16

U	U	U	U	U	U	U	U
—	—	—	—	—	—	—	—
bit 15							bit 8

U	U	U	U	R	R	R	R
—	—	—	—	REV3	REV2	REV1	REV0
bit 7							bit 0

图注：	R = 只读位	U = 未实现位

bit 23-4　**未实现：**读为 0

bit 3-0　**REV<3:0>：**次要版本标识符位
编码器件的版本号 （仅采用连续编号；无主要 / 次要字段）。

25.2 片上稳压器

所有 PIC24FJ64GA104 系列器件使用标称值为 2.5V 的电压为其内核数字逻辑供电。对于需要工作在一个更高的典型电压值如 3.3V 的设计中，这可能会产生冲突。PIC24FJ64GA104 系列的所有器件均包含一个片上稳压器，可使器件内核逻辑运行在 VDD 下。

由 DISVREG 引脚控制稳压器。将 VSS 连接到该引脚将使能稳压器，然后稳压器通过其他 VDD 引脚向内核供电。当使能稳压器时，必须将一个低 ESR 电容（如陶瓷电容）连接到 VDDCORE/VCAP 引脚（图 25-1）。这有利于维持稳压器的稳定性。**第 28.1 节 “直流特性”**中提供了该滤波电容（CEFC）的推荐值。

如果 DISVREG 与 VDD 相连，则禁止稳压器。在这种情况下，必须通过 VDDCORE/VCAP 引脚对器件内核逻辑单独提供标称值为 2.5V 的电压，从而使 I/O 引脚可以具有较高的电压（通常为 3.3V）。另外，VDDCORE/VCAP 和 VDD 引脚可以连接在一起，使器件工作在较低的标称电压下。请参见图 25-1 了解可能的配置。

25.2.1 稳压器跟踪模式和低压检测

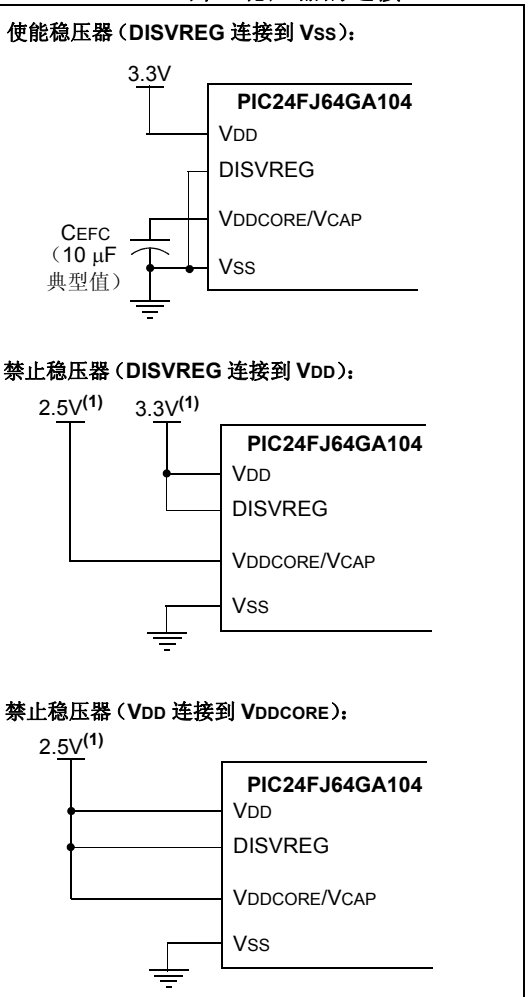
当片上稳压器被使能时，它可以为数字内核逻辑提供标称值为 2.5V 的恒定电压。

稳压器可提供从约为 2.5V 到器件所能承受的最高 VDDMAX 的 VDD 范围。该稳压器无法将低于 2.5V 的 VDD 电压提高。为防止出现“欠压”情况，当电压对于稳压器过低时，稳压器进入跟踪模式。在跟踪模式下，稳压器输出跟随 VDD，但通常比 VDD 低 100 mV。

器件进入跟踪模式后，不可能再继续以全速工作。为提供有关器件何时进入跟踪模式的信息，片上稳压器包含一个简单的低压检测电路。当 VDD 降到全速工作电压以下时，电路会将低压检测中断标志 LVDIF（IFS4<8>）置 1。这可被用于产生中断并使应用进入低功耗工作模式，或触发受控关闭。

低压检测仅在稳压器使能的情况下可用。

图 25-1： 片上稳压器的连接



注 1： 这些为典型的工作电压值。请参见**第 28.1 节 “直流特性”**了解 VDD 和 VDDCORE 的全部工作电压范围。

25.2.2 片上稳压器和 POR

当使能稳压器时，需等待大约 10 μs 才能产生输出。这段时间被称为 TPM，在此期间禁止代码执行。每次器件在掉电（包括休眠模式）之后恢复工作时都会产生 TPM 延时。TPM 由 PMSLP 位（RCON<8>）和 WUTSEL 配置位（CW3<11:10>）的设置决定。

注：	关于 TPM 的更多信息，请参见**第 28.0 节 “电气特性”**。

如果禁止了稳压器，将自动使能独立的上电延时定时器（PWRT）。在器件启动时，PWRT 会增加一段 64 ms 的固定标称延时（仅 POR 或 BOR）。

在禁止稳压器的情况下，从休眠模式唤醒时，TPM 用于决定唤醒时间。在禁止稳压器的情况下工作时，将 PMSLP 位置 1 会缩短器件唤醒时间。

25.2.3 片上稳压器和 BOR

当使能片上稳压器时， PIC24FJ64GA104 系列器件也会有一个简单的欠压保护功能。如果向稳压器提供的电压不足以维持跟踪电压，那么稳压器复位电路将产生欠压复位。 BOR 标志位 （RCON<1>）会捕捉该事件。欠压电压规范在《PIC24F 系列参考手册》的**第 7 章“复位”**（DS39712A_CN）中提供。

25.2.4 上电要求

片上稳压器是为了满足器件的上电要求而设计的。如果应用不使用稳压器，那就必须严格遵守上电条件。在上电时， VDDCORE 决不能超出 VDD 0.3V 以上。

注：	更多信息，请参见**第 28.0 节“电气特性”**。

25.2.5 稳压器待机模式

当使能片上稳压器时，它总是消耗略大于 IDD/IPD 的电流，器件处于休眠模式时也是如此，尽管此时内核数字逻辑并不需要耗能。对于能源紧张的应用，为了节省更多的能源，只要器件通过关闭闪存程序存储器的电源进入休眠模式，稳压器就会自动进入待机模式。该功能由 PMSLP 位 （RCON<8>）控制。默认情况下，该位被清零，这将使能待机模式。

对于 PIC24FJ64GA104 系列器件，将稳压器从待机模式唤醒所需的时间由 WUTSEL<1:0> 配置位 （CW3<11:10>）控制。所有器件的默认唤醒时间为 190 μs，这是一种传统的模式，目的是与较旧的 PIC24F 器件的唤醒时间匹配。

实现的 WUTSEL 配置位可提供快速唤醒选项。当 WUTSEL<1:0> = 01 时，稳压器唤醒时间为 TPM， 10 μs。

当稳压器的待机模式关闭 （PMSLP = 1）时，闪存程序存储器在休眠模式下保持通电状态。这使器件可立即被唤醒，无需等待 TPM。但当 PMSLP 置 1 时，休眠模式下的功耗约比允许稳压器进入待机模式时高 40 μA。

25.3 看门狗定时器 （WDT）

对于 PIC24FJ64GA104 系列器件，WDT 由 LPRC 振荡器驱动。当使能 WDT 时，时钟源也将被使能。

由 LPRC 提供的 WDT 时钟源的频率标称值为 31 kHz。此信号输入给可配置为 5 位 （32 分频）或 7 位 （128 分频）工作的预分频器。预分频比由 FWPSA 配置位设置。使用 31 kHz 输入，预分频器在 5 位模式下将产生 1 ms 的标称 WDT 超时周期 （TWDT），在 7 位模式下产生的超时周期为 4 ms。

分频比可变的后分频器对 WDT 预分频器的输出进行分频，以获得范围较宽的超时周期。后分频比由 WDTPS<3:0> 配置位 （CW1<3:0>）控制，配置位允许选择 16 种设置，从 1:1 到 1:32,768。使用预分频器和后分频器，可以使超时周期的范围扩展到 1 ms 至 131 秒。

WDT、预分频器和后分频器在以下条件下复位：

- 任何器件复位时
- 在完成时钟切换时，无论时钟切换是由软件 （即，在更改 NOSC 位后将 OSWEN 位置 1）引起还是由硬件 （即，故障保护时钟监视器）引起
- 当执行 PWRSAV 指令时 （即，进入休眠或空闲模式）
- 当器件退出休眠或空闲模式恢复正常工作时
- 当在正常执行过程中执行 CLRWDT 指令时

如果使能了 WDT，它将在休眠或空闲模式下继续运行。当发生 WDT 超时时，将唤醒器件并且将从执行 PWRSAV 指令处继续执行代码。唤醒器件后，需要用软件将对应的 SLEEP 或 IDLE 位 （RCON<3:2>）清零。

WDT 标志位 WDTO（RCON<4>）不会在 WDT 超时后自动清零。要检测后面的 WDT 事件，必须用软件将该标志位清零。

注：	执行 CLRWDT 和 PWRSAV 指令会将预分频器和后分频器的计数值清零。

25.3.1 窗口操作

看门狗定时器有一种可选的固定窗口工作模式。在该窗口模式下， CLRWDT 指令只能在设定的 WDT 周期的后 1/4 复位 WDT。在该窗口前执行的 CLRWDT 指令会导致 WDT 复位；这类似于 WDT 超时。

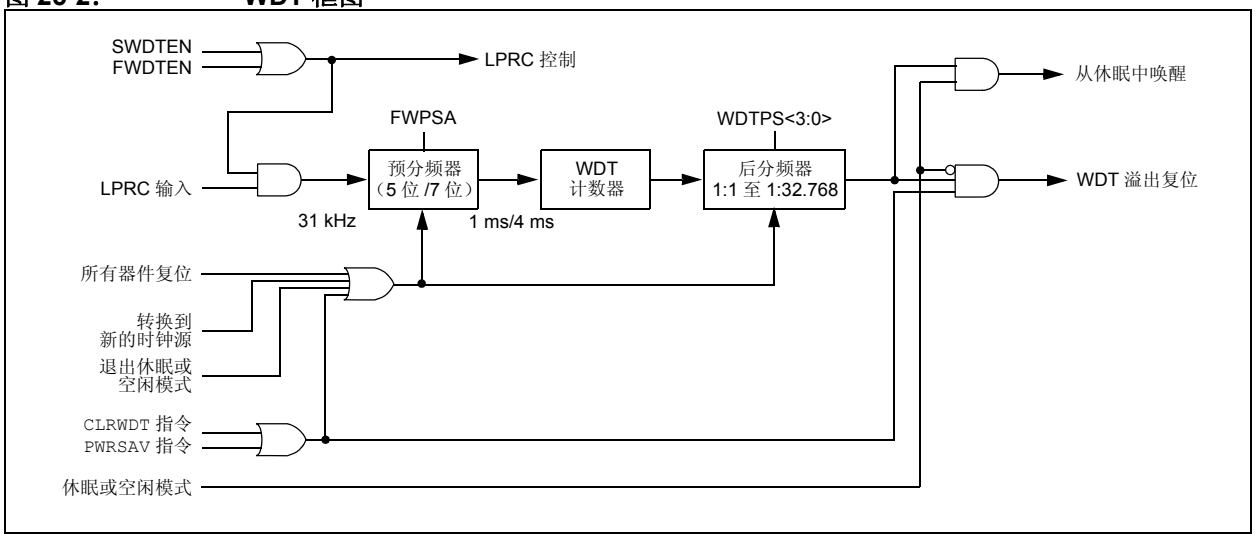
窗口 WDT 模式通过将 WINDIS 配置位 （CW1<6>）编程为 0 来使能。

25.3.2 控制寄存器

WDT 通过 FWDTEN 配置位使能或禁止。当 FWDTEN 配置位置 1 时， WDT 始终是使能的。

当 FWDTEN 配置位被编程为 0 时，可以选择用软件控制 WDT。用软件通过将 SWDTEN 控制位（RCON<5>）置 1 来使能 WDT。任何器件复位都会导致 SWDTEN 控制位清零。 WDT 软件选项允许用户在关键代码段使能 WDT 并在非关键代码段禁止 WDT，从而最大限度地降低功耗。

图 25-2： WDT 框图



25.4 深度休眠看门狗定时器 （DSWDT）

PIC24FJ64GA104 系列器件具有 WDT 模块和 DSWDT 模块。 DSWDT 模块在器件处于深度休眠模式时运行（如果使能），由 SOSC 或 LPRC 振荡器驱动。时钟源通过 DSWDTOSC （CW4<4>）配置位进行选择。

通过选择相应的后分频比， DSWDT 的超时周期可以配置为 2.1 ms 至 25.7 天。后分频比可以通过配置位 DSWDTPS<3:0> （CW4<3:0>）进行选择。当使能 DSWDT 时，时钟源也将被使能。 DSWDT 是可以将器件从深度休眠模式中唤醒的唤醒源之一。

25.5 程序校验和代码保护

PIC24FJ64GA104 系列器件提供了两种补充方法来防止应用程序代码被改写和擦除。这两种方法还可以防止器件配置在运行过程中被意外更改。

25.5.1 通用段保护

对于 PIC24FJ64GA104 系列中的所有器件，片上程序存储空间被当作一个单独的块，称为通用段 （General Segment， GS）。配置位 GCP 控制该块的代码保护。该位阻止外部对程序存储空间的读和写操作。但对正常的执行模式没有直接影响。

写保护由配置字中的 GWRP 位控制。当 GWRP 被编程为 0 时，禁止在内部写和擦除程序存储器。

25.5.2 代码段保护

除了全局通用段保护外，可以单独保护程序存储空间的独立子区域以防止对其进行写和擦除操作。该区域可以有很多用途（例如自举应用程序），需要单独对这些区域中的代码块进行保护，以防其被擦除或改写。不同于常见的引导区，用户可在程序空间内任何位置定位 PIC24FJ64GA104 系列器件中的任意大小的特定受保护段。

代码段保护通过在写或擦除地址位于指定范围内时禁止 NVM安全互锁，以此向程序存储器的指定区域提供了新增的保护等级。它不会改写由 GCP 或 GWR 位控制的通用段保护。例如，如果使能了 GCP 和 GWRP，那么使能程序存储器低半部分的段代码保护不会取消高半部分的通用段保护。

受保护代码段的大小和类型由配置字 3 中的 WPFPx、WPEND、WPCFG 和 WPDIS 位进行配置。通过对 WPDIS 位进行编程（= 0）来使能代码段保护。WPFP 位通过指定受保护段的起始或结束地址的 512 字的代码页确定被保护段的大小。指定区域也被包括在内，因此该页也将被保护。

WPEND 位决定被保护段是否将程序空间的顶部或底部作为边界。对 WPEND 进行编程（= 0）会将程序存储器的底部（000000h）设置为被保护段的下边界。不对 WPEND 进行编程（= 1）会保护从指定页到已实现程序存储器的最后页的存储区域，包括配置字单元。

独立位 WPCFG 用于单独保护程序空间的最后页，包括闪存配置字。对 WPCFG 进行编程（= 0）会保护最后页，与其他位的设置无关。在需要对存储器底部的代码段和闪存配置字同时进行写保护的情况下，这可能有用。

段代码保护的各种选项如表 25-2 所示。

25.5.3 配置寄存器保护

有两种方法可以保护配置寄存器，使其免遭意外改写或读取。主要保护方法与保护 RP 寄存器的方法相同——影子寄存器包含一个补充值，持续将该值与实际值进行比较。

为防范不可预见事件造成损害，由于单独的存储单元故障（如 ESD 事件）引起的配置位更改将导致奇偶校验错误并触发器件复位。

配置寄存器的数据来自于程序存储器中的闪存配置字。当 GCP 位置 1 时，也将保护器件配置的源数据。即使未使能通用段保护，也可以通过使用适当的代码段保护设置来保护器件配置。

表 25-2： 段代码保护配置选项

段配置位			代码段的写 / 擦除保护
WPDIS	**WPEND**	**WPCFG**	
1	x	1	未使能其他保护；所有程序存储器保护由 GCP 和 GWRP 配置
1	x	0	最后代码页受保护，包括闪存配置字
0	1	0	保护从由 WPFP<5:0> 定义的代码页的首地址，到实现的程序存储器的尾地址（包括在内），包括闪存配置字
0	0	0	保护从地址 000000h 到由 WPFP<5:0> 定义的代码页的尾地址（包括在内）
0	1	1	保护从由 WPFP<5:0> 定义的代码页的首地址，到实现的程序存储器的尾地址（包括在内），包括闪存配置字
0	0	1	保护从由 WPFP<5:0> 定义的代码页的首地址，到实现的程序存储器的尾地址（包括在内）

25.6 JTAG 接口

PIC24FJ64GA104 系列器件实现了 JTAG 接口，以支持边界扫描器件测试。

25.7 在线串行编程

可以在最终的应用电路中对 PIC24FJ64GA104 系列单片机进行串行编程。只需要 5 根线即可实现这一操作，其中时钟线（PGECx）、数据线（PGEDx）各一根，其余 3 根分别是电源线、接地线和编程电压线。这允许用户在生产电路板时使用未编程器件，仅在产品交付之前才对单片机进行编程，从而可以使用最新版本的固件或者定制固件进行编程。

25.8 在线调试器

当选择 MPLAB® ICD 2 作为调试器时，在线调试功能会被使能。该功能允许与 MPLAB IDE 配合使用进行简单的调试。通过 PGECx（仿真 / 调试时钟）和 PGEDx（仿真 / 调试数据）引脚控制调制功能。

要使用器件的在线调试功能，在设计中必须实现 $\overline{\text{MCLR}}$、VDD、VSS 以及由 ICS 配置位指定的 PGECx/PGEDx 引脚对的 ICSP 连接。此外，当使能该功能时，某些资源就不能用于一般用途了。这些资源包括数据 RAM 的前 80 字节和两个 I/O 引脚。

26.0 开发支持

一系列软件及硬件开发工具对PIC® 单片机和dsPIC® 数字信号控制器提供支持：

- 集成开发环境
  - MPLAB® IDE 软件
- 编译器 / 汇编器 / 链接器
  - 适用于各种器件系列的 MPLAB C 编译器
  - 适用于各种器件系列的 HI-TECH C 编译器
  - MPASM™ 汇编器
  - MPLINK™ 目标链接器 / MPLIB™ 目标库管理器
  - 适用于各种器件系列的 MPLAB 汇编器 / 链接器 / 库管理器
- 模拟器
  - MPLAB SIM 软件模拟器
- 仿真器
  - MPLAB REAL ICE™ 在线仿真器
- 在线调试器
  - MPLAB ICD 3
  - PICkit™ 3 Debug Express
- 器件编程器
  - PICkit™ 2 编程器
  - MPLAB PM3 器件编程器
- 低成本演示 / 开发板、评估工具包及入门工具包

26.1 MPLAB 集成开发环境软件

MPLAB IDE 软件为 8/16/32 位单片机市场提供了前所未有的易于使用的软件开发平台。 MPLAB IDE 是基于 Windows® 操作系统的应用软件，包括：

- 一个包含所有调试工具的图形界面
  - 模拟器
  - 编程器 （单独销售）
  - 在线仿真器 （单独销售）
  - 在线调试器 （单独销售）
- 具有彩色上下文代码显示的全功能编辑器
- 多项目管理器
- 内容可直接编辑的可定制式数据窗口
- 高级源代码调试
- 鼠标停留在变量上进行查看的功能
- 将变量从源代码窗口拖放到 Watch （观察）窗口
- 丰富的在线帮助
- 集成了可选的第三方工具，如 IAR C 编译器

MPLAB IDE 可以让您：

- 编辑源文件 （C 语言或汇编语言）
- 点击一次即可完成编译或汇编，并将代码下载到仿真器和模拟器工具中 （自动更新所有项目信息）
- 可使用如下各项进行调试：
  - 源文件 （C 语言或汇编语言）
  - 混合 C 语言和汇编语言
  - 机器码

MPLAB IDE 在单个开发范例中支持使用多种调试工具，包括从成本效益高的模拟器到低成本的在线调试器，再到全功能的仿真器。这样缩短了用户升级到更加灵活而功能强大的工具时的学习时间。

26.2 适用于各种器件系列的 MPLAB C 编译器

MPLAB C 编译器代码开发系统是完全的 ANSI C 编译器，适用于 Microchip 的 PIC18、PIC24 和 PIC32 系列单片机及 dsPIC30 和 dsPIC33 系列数字信号控制器。这些编译器提供强大的集成功能和出众的代码优化能力，且使用方便。

为便于源代码调试，编译器提供针对 MPLAB IDE 调试器优化的符号信息。

26.3 适用于各种器件系列的 HI-TECH C 编译器

HI-TECH C 编译器代码开发系统是完全的 ANSI C 编译器，适用于 Microchip 的 PIC 系列单片机及 dsPIC 系列数字信号控制器。这些编译器提供强大的集成功能和全知代码生成能力，且使用方便。

为便于源代码调试，编译器提供针对 MPLAB IDE 调试器优化的符号信息。

编译器包括一个宏汇编器、链接器、预处理程序和单步驱动程序，可以在多种平台上运行。

26.4 MPASM 汇编器

MPASM 汇编器是全功能通用宏汇编器，适用于 PIC10/12/16/18 MCU。

MPASM 汇编器可生成用于 MPLINK 目标链接器的可重定位目标文件、Intel® 标准 HEX 文件、详细描述存储器使用状况和符号参考的 MAP 文件、包含源代码行及生成机器码的绝对 LST 文件以及用于调试的 COFF 文件。

MPASM 汇编器具有如下特性：

- 集成在 MPLAB IDE 项目中
- 用户定义的宏可简化汇编代码
- 对多用途源文件进行条件汇编
- 允许完全控制汇编过程的指令

26.5 MPLINK 目标链接器 / MPLIB 目标库管理器

MPLINK 目标链接器包含了由 MPASM 汇编器、MPLAB C18 C 编译器产生的可重定位目标。通过使用链接器脚本中的指令，它还可链接预编译库中的可重定位目标。

MPLIB目标库管理器管理预编译代码库文件的创建和修改。当从源文件调用库中的一段子程序时，只有包含此子程序的模块被链接到应用程序。这样可使大型库在许多不同应用中被高效地利用。

目标链接器 / 库管理器具有如下特性：

- 高效地连接单个的库而不是许多小文件
- 通过将相关的模块组合在一起来增强代码的可维护性
- 只要列出、替换、删除和抽取模块，便可灵活地创建库

26.6 适用于各种器件系列的 MPLAB 汇编器、链接器和库管理器

MPLAB 汇编器为 PIC24、PIC32 和 dsPIC 器件从符号汇编语言生成可重定位机器码。 MPLAB C 编译器使用该汇编器生成目标文件。汇编器产生可重定位目标文件之后，可将这些目标文件存档，或与其他可重定位目标文件和存档链接以生成可执行文件。该汇编器有如下显著特性：

- 支持整个器件指令集
- 支持定点数据和浮点数据
- 命令行界面
- 丰富的指令集
- 灵活的宏语言
- MPLAB IDE 兼容性

26.7 MPLAB SIM 软件模拟器

MPLAB SIM 软件模拟器通过在指令级对 PIC MCU 和 dsPIC® DSC 进行模拟，可在 PC 主机环境下进行代码开发。对于任何给定的指令，都可以对数据区进行检查或修改，并通过一个全面的激励控制器来施加激励。可以将各寄存器记录在文件中，以便进行进一步的运行时分析。跟踪缓冲区和逻辑分析器的显示使软件模拟器还能记录和跟踪程序的执行、 I/O 的动作、大部分的外设及内部寄存器。

MPLAB SIM 软件模拟器完全支持使用 MPLAB C 编译器以及 MPASM 和 MPLAB 汇编器的符号调试。该软件模拟器可用于在硬件实验室环境外灵活地开发和调试代码，是一款完美且经济的软件开发工具。

26.8 MPLAB REAL ICE 在线仿真器系统

MPLAB REAL ICE 在线仿真器系统是 Microchip 针对其闪存 DSC 和 MCU 器件而推出的新一代高速仿真器。结合 MPLAB 集成开发环境（IDE）所具有的易于使用且功能强大的图形用户界面，该仿真器可对 PIC® 闪存 MCU 和 dsPIC® 闪存 DSC 进行调试和编程。IDE 是随每个工具包一起提供的。

该仿真器通过高速 USB 2.0 接口与设计工程师的 PC 相连，并利用与在线调试器系统兼容的连接器（RJ11）或新型抗噪声、高速低压差分信号（LVDS）互连电缆（CAT5）与目标板相连。

可通过 MPLAB IDE 下载将来版本的固件，对该仿真器进行现场升级。在即将推出的 MPLAB IDE 版本中，会支持许多新器件，还将增加一些新特性。在同类仿真器中，MPLAB REAL ICE 的优势十分明显：低成本、全速仿真、运行时变量查看、跟踪分析、复杂断点、耐用的探针接口及较长（长达 3 米）的互连电缆。

26.9 MPLAB ICD 3 在线调试器系统

MPLAB ICD 3 在线调试器系统是 Microchip 成本效益最高的高速硬件调试器 / 编程器，适用于 Microchip 闪存数字信号控制器（DSC）和单片机（MCU）器件。结合 MPLAB 集成开发环境（IDE）所具有的功能强大但易于使用的图形用户界面，该调试器可对 PIC® 闪存单片机和 dsPIC® DSC 进行调试和编程。

MPLAB ICD 3 在线调试器通过高速 USB 2.0 接口与设计工程师的PC相连，并利用与MPLAB ICD 2或MPLAB REAL ICE 系统兼容的连接器（RJ-11）与目标板相连。MPLAB ICD 3 支持所有 MPLAB ICD 2 转接器。

26.10 PICkit 3 在线调试器 / 编程器及 PICkit 3 Debug Express

结合 MPLAB 集成开发环境（IDE）所具有的功能强大的图形用户界面，MPLAB PICkit 3 可对 PIC® 闪存单片机和 dsPIC® 数字信号控制器进行调试和编程，且价位较低。MPLAB PICkit 3 通过全速 USB 接口与设计工程师的 PC 相连，并利用 Microchip 调试（RJ-11）连接器（与 MPLAB ICD 3 和 MPLAB REAL ICE 兼容）与目标板相连。连接器使用两个器件 I/O 引脚和复位线来实现在线调试和在线串行编程。

PICkit 3 Debug Express 包括 PICkit 3、演示板和单片机、连接电缆和光盘（内含用户指南、课程、教程、编译器和 MPLAB IDE 软件）。

26.11 PICkit 2 开发编程器 / 调试器及 PICkit 2 Debug Express

PICkit™ 2 开发编程器 / 调试器是一款低成本开发工具，具有易于使用的界面，适用于对 Microchip 的闪存系列单片机进行编程和调试。这一全功能的 Windows® 编程界面支持低档（PIC10F、PIC12F5xx 和 PIC16F5xx）、中档（PIC12F6xx 和 PIC16F）、PIC18F、PIC24、dsPIC30、dsPIC33 和 PIC32 系列的 8 位、16 位及 32 位单片机，以及许多 Microchip 串行 EEPROM 产品。结合 Microchip 功能强大的 MPLAB 集成开发环境（IDE），PICkit 2 可对大多数 PIC® 单片机进行在线调试。即使 PIC 单片机已嵌入应用，在线调试功能仍可以运行、暂停和单步执行程序。在断点处暂停时，可以检查和修改文件寄存器。

PICkit 2 Debug Express 包括 PICkit 2、演示板和单片机、连接电缆和光盘（内含用户指南、课程、教程、编译器和 MPLAB IDE 软件）。

26.12 MPLAB PM3 器件编程器

MPLAB PM3 器件编程器是一款符合 CE 规范的通用器件编程器，在 VDDMIN 和 VDDMAX 点对其可编程电压进行校验以确保可靠性最高。它有一个用来显示菜单和错误消息的大 LCD 显示器（128 x 64），以及一个支持各种封装类型的可拆卸模块化插槽装置。编程器标准配置中带有一根 ICSP™ 电缆。在单机模式下，MPLAB PM3 器件编程器不必与 PC 相连即可对 PIC 器件进行读取、校验和编程。在该模式下它还可设置代码保护。MPLAB PM3 通过 RS-232 或 USB 电缆连接到 PC 主机上。MPLAB PM3 具备高速通信能力以及优化算法，可对具有大存储器的器件进行快速编程。它还包含了 MMC 卡，用于文件存储及数据应用。

26.13 演示 / 开发板、评估工具包及入门工具包

有许多演示、开发和评估板可用于各种 PIC MCU 和 dsPIC DSC，实现对全功能系统的快速应用开发。大多数的演示、开发和评估板都有实验布线区，供用户添加定制电路；还有应用固件和源代码，用于检查和修改。

这些板支持多种功能部件，包括 LED、温度传感器、开关、扬声器、RS-232 接口、LCD 显示器、电位计和附加 EEPROM 存储器。

演示和开发板可用于教学环境，在实验布线区设计定制电路，从而掌握各种单片机应用。

除了 PICDEM™ 和 dsPICDEM™ 演示 / 开发板系列电路外，Microchip 还有一系列评估工具包和演示软件，适用于模拟滤波器设计、KEELOQ® 数据安全产品 IC、CAN、IrDA®、PowerSmart 电池管理、SEEVAL® 评估系统、Σ–Δ ADC、流速传感器，等等。

同时还提供入门工具包，其中包含体验指定器件功能所需的所有软硬件。通常提供单个应用以及调试功能，都包含在一块电路板上。

有关演示、开发和评估工具包的完整列表，请访问 Microchip 网站（www.microchip.com）。

27.0 指令集汇总

注： 本章简要概述了 PIC24F 指令集架构，但是不应将其当作无所不包的参考手册来使用。

与以前的 PIC® MCU 指令集相比，PIC24F 指令集添加了许多增强功能，同时保持了易于从以前 PIC MCU 指令集移植的特点。大部分指令的长度为一个程序存储字。只有三条指令需要两个程序存储单元。

每一条单字指令长 24 位，分为一个指定指令类型的 8 位操作码和进一步指定指令操作的一个或多个操作数。指令集是高度正交的，分为 4 个基本类别：

- 针对字或字节的操作
- 针对位的操作
- 立即数操作
- 控制操作

表 27-1 给出了在说明指令时使用的通用符号。表 27-2 是 PIC24F 指令集的汇总，它列出了所有指令，以及每条指令影响的状态标志位。

大多数针对字或字节的 W 寄存器指令 （包括桶形移位指令）有三个操作数：

- 第一个源操作数，通常是寄存器Wb，不带任何地址修改量
- 第二个源操作数，通常是寄存器Ws，带或不带地址修改量
- 运算结果的目标寄存器，通常是寄存器Wd，带或不带地址修改量

不过，针对字或字节的文件寄存器指令只有两个操作数：

- 文件寄存器，由 f 值指定
- 目标寄存器，可以是文件寄存器 f 或 W0 寄存器（用 WREG 表示）

大多数位操作类指令 （包括简单的循环 / 移位指令）有两个操作数：

- W 寄存器 （带或不带地址修改量）或文件寄存器（由 Ws 或 f 的值指定）
- W 寄存器或文件寄存器中的位（由立即数值指定，或者由寄存器 Wb 的内容间接指定）

涉及数据传送的立即数指令，可以使用下列操作数：

- 要被装入到 W 寄存器或文件寄存器中的立即数（由 k 值指定）
- 要装入立即数的 W 寄存器或文件寄存器（由 Wb 或 f 指定）

然而，涉及算术或逻辑运算的立即数指令，使用如下的操作数：

- 第一个源操作数是寄存器 Wb，不带任何地址修改量
- 第二个源操作数是立即数
- 操作结果的目标寄存器（仅在与第一个源操作数不同时）通常是寄存器 Wd （带或不带地址修改量）

控制指令可以使用下列操作数：

- 程序存储器中的地址
- 表读和表写指令的模式

除了某些双字指令外，所有指令都是单字指令；双字指令之所以是双字长的 （48 位），是因为要用 48 位来提供所需信息。在第二个字中，高 8 位全为 0。如果指令自身把第二个字当作一条指令来执行的话，它将作为一条 NOP 指令来执行。

大多数单字长指令都在一个指令周期内执行，除非条件测试结果为真或者指令执行结果改变了程序计数器。对于上述两种特殊情况，指令执行需要两个指令周期，在第二个指令周期中执行一条 NOP 指令。值得注意的例外是 BRA（无条件 / 计算跳转）、间接 CALL/GOTO、所有的表读和表写以及 RETURN/RETFIE 指令，它们是单字长指令，但执行需要两个或三个周期。

某些可能涉及到跳过后续指令的指令，如果要执行跳过的话，可能需要两个或三个周期，这取决于被跳过的指令是单字还是双字指令。此外，双字传送需要两个周期。双字指令执行需要两个指令周期。

表 27-1： 操作码说明中使用的符号

字段	说明
#text	表示由 text 定义的立即数
(text)	表示 text 的内容
[text]	表示由 text 寻址的存储单元
{ }	可选字段或操作
<n:m>	寄存器位域
.b	字节模式选择
.d	双字模式选择
.S	影子寄存器选择
.w	字模式选择 （默认）
bit4	4 位位选择字段 （用于字寻址指令） ∈ {0...15}
C, DC, N, OV, Z	MCU 状态位：进位、半进位、负、溢出和全零标志位
Expr	绝对地址、标号或表达式 （由链接器解析）
f	文件寄存器地址 ∈ {0000h...1FFFh}
lit1	1 位无符号立即数 ∈ {0,1}
lit4	4 位无符号立即数 ∈ {0...15}
lit5	5 位无符号立即数 ∈ {0...31}
lit8	8 位无符号立即数 ∈ {0...255}
lit10	10 位无符号立即数，对于字节模式， ∈ {0...255}；对于字模式， ∈ {0:1023}
lit14	14 位无符号立即数 ∈ {0...16383}
lit16	16 位无符号立即数 ∈ {0...65535}
lit23	23 位无符号立即数 ∈ {0...8388607}； LSB 必须为 0
None	字段无需内容，可为空
PC	程序计数器
Slit10	10 位有符号立即数 ∈ {-512...511}
Slit16	16 位有符号立即数 ∈ {-32768...32767}
Slit6	6 位有符号立即数 ∈ {-16...16}
Wb	基本 W 寄存器 ∈ {W0..W15}
Wd	目标 W 寄存器 ∈ {Wd, [Wd], [Wd++], [Wd--], [++Wd], [--Wd]}
Wdo	目标 W 寄存器 ∈ {Wnd, [Wnd], [Wnd++], [Wnd--], [++Wnd], [--Wnd], [Wnd+Wb]}
Wm,Wn	被除数和除数工作寄存器对 （直接寻址）
Wn	16 个工作寄存器之一 ∈ {W0..W15}
Wnd	16 个目标工作寄存器之一 ∈ {W0..W15}
Wns	16 个源工作寄存器之一 ∈ {W0..W15}
WREG	W0 （文件寄存器指令中使用的工作寄存器）
Ws	源 W 寄存器 ∈ {Ws, [Ws], [Ws++], [Ws--], [++Ws], [--Ws]}
Wso	源 W 寄存器 ∈ {Wns, [Wns], [Wns++], [Wns--], [++Wns], [--Wns], [Wns+Wb]}

表 27-2： 指令集概述

汇编助记符	汇编语法		说明	字数	周期数	影响的状态标志
ADD	ADD	f	f = f + WREG	1	1	C, DC, N, OV, Z
	ADD	f,WREG	WREG = f + WREG	1	1	C, DC, N, OV, Z
	ADD	#lit10,Wn	Wd = lit10 + Wd	1	1	C, DC, N, OV, Z
	ADD	Wb,Ws,Wd	Wd = Wb + Ws	1	1	C, DC, N, OV, Z
	ADD	Wb,#lit5,Wd	Wd = Wb + lit5	1	1	C, DC, N, OV, Z
ADDC	ADDC	f	f = f + WREG + (C)	1	1	C, DC, N, OV, Z
	ADDC	f,WREG	WREG = f + WREG + (C)	1	1	C, DC, N, OV, Z
	ADDC	#lit10,Wn	Wd = lit10 + Wd + (C)	1	1	C, DC, N, OV, Z
	ADDC	Wb,Ws,Wd	Wd = Wb + Ws + (C)	1	1	C, DC, N, OV, Z
	ADDC	Wb,#lit5,Wd	Wd = Wb + lit5 + (C)	1	1	C, DC, N, OV, Z
AND	AND	f	f = f .AND.WREG	1	1	N, Z
	AND	f,WREG	WREG = f .AND.WREG	1	1	N, Z
	AND	#lit10,Wn	Wd = lit10 .AND.Wd	1	1	N, Z
	AND	Wb,Ws,Wd	Wd = Wb .AND.Ws	1	1	N, Z
	AND	Wb,#lit5,Wd	Wd = Wb .AND. lit5	1	1	N, Z
ASR	ASR	f	f = 算术右移 f	1	1	C, N, OV, Z
	ASR	f,WREG	WREG = 算术右移 f	1	1	C, N, OV, Z
	ASR	Ws,Wd	Wd = 算术右移 Ws	1	1	C, N, OV, Z
	ASR	Wb,Wns,Wnd	Wnd = 将 Wb 算术右移 Wns 位	1	1	N, Z
	ASR	Wb,#lit5,Wnd	Wnd = 将 Wb 算术右移 lit5 位	1	1	N, Z
BCLR	BCLR	f,#bit4	将 f 中的指定位清零	1	1	无
	BCLR	Ws,#bit4	将 Ws 中的指定位清零	1	1	无
BRA	BRA	C,Expr	如果有进位则跳转	1	1（2）	无
	BRA	GE,Expr	如果大于或等于则跳转	1	1（2）	无
	BRA	GEU,Expr	如果无符号大于或等于则跳转	1	1（2）	无
	BRA	GT,Expr	如果大于则跳转	1	1（2）	无
	BRA	GTU,Expr	如果无符号大于则跳转	1	1（2）	无
	BRA	LE,Expr	如果小于或等于则跳转	1	1（2）	无
	BRA	LEU,Expr	如果无符号小于或等于则跳转	1	1（2）	无
	BRA	LT,Expr	如果小于则跳转	1	1（2）	无
	BRA	LTU,Expr	如果无符号小于则跳转	1	1（2）	无
	BRA	N,Expr	如果为负则跳转	1	1（2）	无
	BRA	NC,Expr	如果没有进位则跳转	1	1（2）	无
	BRA	NN,Expr	如果不为负则跳转	1	1（2）	无
	BRA	NOV,Expr	如果不溢出则跳转	1	1（2）	无
	BRA	NZ,Expr	如果不为零则跳转	1	1（2）	无
	BRA	OV,Expr	如果溢出则跳转	1	1（2）	无
	BRA	Expr	无条件跳转	1	2	无
	BRA	Z,Expr	如果为零则跳转	1	1（2）	无
	BRA	Wn	计算跳转	1	2	无
BSET	BSET	f,#bit4	将 f 中的指定位置 1	1	1	无
	BSET	Ws,#bit4	将 Ws 中的指定位置 1	1	1	无
BSW	BSW.C	Ws,Wb	将 C 位内容写入 Ws<Wb>	1	1	无
	BSW.Z	Ws,Wb	将 Z 位内容写入 Ws<Wb>	1	1	无
BTG	BTG	f,#bit4	将 f 中的指定位翻转	1	1	无
	BTG	Ws,#bit4	将 Ws 中的指定位翻转	1	1	无
BTSC	BTSC	f,#bit4	对 f 中的指定位进行测试，如果为零则跳过	1	1（2 或 3）	无
	BTSC	Ws,#bit4	对 Ws 中的指定位进行测试，如果为零则跳过	1	1（2 或 3）	无

表 27-2： 指令集概述 （续）

汇编助记符	汇编语法	说明	字数	周期数	影响的状态标志
BTSS	BTSS f,#bit4	对 f 中的指定位进行测试，如果为 1 则跳过	1	1（2 或 3）	无
	BTSS Ws,#bit4	对 Ws 中的指定位进行测试，如果为 1 则跳过	1	1（2 或 3）	无
BTST	BTST f,#bit4	对 f 中的指定位进行测试	1	1	Z
	BTST.C Ws,#bit4	对 Ws 中的指定位进行测试，并将其值存储到 C	1	1	C
	BTST.Z Ws,#bit4	对 Ws 中的指定位进行测试，并将其反码存储到 Z	1	1	Z
	BTST.C Ws,Wb	对 Ws<Wb> 位进行测试，并将其值存储到 C	1	1	C
	BTST.Z Ws,Wb	对 Ws<Wb> 位进行测试，并将其反码存储到 Z	1	1	Z
BTSTS	BTSTS f,#bit4	对 f 中的指定位进行测试，并将 f 中的该位置 1	1	1	Z
	BTSTS.C Ws,#bit4	对 Ws 中的指定位进行测试，并将其值存储到 C，然后将 Ws 中的该位置 1	1	1	C
	BTSTS.Z Ws,#bit4	对 Ws 中的指定位进行测试，并将其反码存储到 Z，然后将 Ws 中的该位置 1	1	1	Z
CALL	CALL lit23	调用子程序	2	2	无
	CALL Wn	间接调用子程序	1	2	无
CLR	CLR f	f = 0x0000	1	1	无
	CLR WREG	WREG = 0x0000	1	1	无
	CLR Ws	Ws = 0x0000	1	1	无
CLRWDT	CLRWDT	将看门狗定时器清零	1	1	WDTO, Sleep
COM	COM f	$f = \overline{f}$	1	1	N, Z
	COM f,WREG	$WREG = \overline{f}$	1	1	N, Z
	COM Ws,Wd	$Wd = \overline{Ws}$	1	1	N, Z
CP	CP f	比较 f 和 WREG	1	1	C, DC, N, OV, Z
	CP Wb,#lit5	比较 Wb 和 lit5	1	1	C, DC, N, OV, Z
	CP Wb,Ws	比较 Wb 和 Ws （Wb – Ws）	1	1	C, DC, N, OV, Z
CP0	CP0 f	比较 f 和 0x0000	1	1	C, DC, N, OV, Z
	CP0 Ws	比较 Ws 和 0x0000	1	1	C, DC, N, OV, Z
CPB	CPB f	带借位比较 f 和 WREG	1	1	C, DC, N, OV, Z
	CPB Wb,#lit5	带借位比较 Wb 和 lit5	1	1	C, DC, N, OV, Z
	CPB Wb,Ws	带借位比较 Wb 和 Ws （$Wb – Ws – \overline{C}$）	1	1	C, DC, N, OV, Z
CPSEQ	CPSEQ Wb,Wn	比较 Wb 和 Wn，如果相等则跳过	1	1（2 或 3）	无
CPSGT	CPSGT Wb,Wn	比较 Wb 和 Wn，如果大于则跳过	1	1（2 或 3）	无
CPSLT	CPSLT Wb,Wn	比较 Wb 和 Wn，如果小于则跳过	1	1（2 或 3）	无
CPSNE	CPSNE Wb,Wn	比较 Wb 和 Wn，如果不相等则跳过	1	1（2 或 3）	无
DAW	DAW.B Wn	Wn = 十进制调整 Wn	1	1	C
DEC	DEC f	f = f – 1	1	1	C, DC, N, OV, Z
	DEC f,WREG	WREG = f – 1	1	1	C, DC, N, OV, Z
	DEC Ws,Wd	Wd = Ws – 1	1	1	C, DC, N, OV, Z
DEC2	DEC2 f	f = f – 2	1	1	C, DC, N, OV, Z
	DEC2 f,WREG	WREG = f – 2	1	1	C, DC, N, OV, Z
	DEC2 Ws,Wd	Wd = Ws – 2	1	1	C, DC, N, OV, Z
DISI	DISI #lit14	在 k 个指令周期内禁止中断	1	1	无
DIV	DIV.SW Wm,Wn	有符号 16/16 位整数除法	1	18	N, Z, C, OV
	DIV.SD Wm,Wn	有符号 32/16 位整数除法	1	18	N, Z, C, OV
	DIV.UW Wm,Wn	无符号 16/16 位整数除法	1	18	N, Z, C, OV
	DIV.UD Wm,Wn	无符号 32/16 位整数除法	1	18	N, Z, C, OV
EXCH	EXCH Wns,Wnd	交换 Wns 和 Wnd 的内容	1	1	无
FF1L	FF1L Ws,Wnd	从左边 （MSb）查找第一个 1	1	1	C
FF1R	FF1R Ws,Wnd	从右边 （LSb）查找第一个 1	1	1	C

表 27-2： 指令集概述 （续）

汇编助记符	汇编语法		说明	字数	周期数	影响的状态标志
GOTO	GOTO	Expr	转移到地址	2	2	无
	GOTO	Wn	间接转移到地址	1	2	无
INC	INC	f	f = f + 1	1	1	C, DC, N, OV, Z
	INC	f,WREG	WREG = f + 1	1	1	C, DC, N, OV, Z
	INC	Ws,Wd	Wd = Ws + 1	1	1	C, DC, N, OV, Z
INC2	INC2	f	f = f + 2	1	1	C, DC, N, OV, Z
	INC2	f,WREG	WREG = f + 2	1	1	C, DC, N, OV, Z
	INC2	Ws,Wd	Wd = Ws + 2	1	1	C, DC, N, OV, Z
IOR	IOR	f	f = f .IOR.WREG	1	1	N, Z
	IOR	f,WREG	WREG = f .IOR.WREG	1	1	N, Z
	IOR	#lit10,Wn	Wd = lit10 .IOR.Wd	1	1	N, Z
	IOR	Wb,Ws,Wd	Wd = Wb .IOR.Ws	1	1	N, Z
	IOR	Wb,#lit5,Wd	Wd = Wb .IOR. lit5	1	1	N, Z
LNK	LNK	#lit14	分配堆栈帧	1	1	无
LSR	LSR	f	f = 逻辑右移 f	1	1	C, N, OV, Z
	LSR	f,WREG	WREG = 逻辑右移 f	1	1	C, N, OV, Z
	LSR	Ws,Wd	Wd = 逻辑右移 Ws	1	1	C, N, OV, Z
	LSR	Wb,Wns,Wnd	Wnd = 将 Wb 逻辑右移 Wns 位	1	1	N, Z
	LSR	Wb,#lit5,Wnd	Wnd = 将 Wb 逻辑右移 lit5 位	1	1	N, Z
MOV	MOV	f,Wn	将 f 中的内容送入 Wn	1	1	无
	MOV	[Wns+Slit10],Wnd	将 [Wns + Slit10] 中的内容送入 Wnd	1	1	无
	MOV	f	将 f 中的内容送入目标寄存器	1	1	N, Z
	MOV	f,WREG	将 f 中的内容送入 WREG	1	1	N, Z
	MOV	#lit16,Wn	将 16 位立即数送入 Wn	1	1	无
	MOV.b	#lit8,Wn	将 8 位立即数送入 Wn	1	1	无
	MOV	Wn,f	将 Wn 中的内容送入 f	1	1	无
	MOV	Wns,[Wns+Slit10]	将 Wns 中的内容送入 [Wns + Slit10]	1	1	
	MOV	Wso,Wdo	将 Ws 中的内容送入 Wd	1	1	无
	MOV	WREG,f	将 WREG 中的内容送入 f	1	1	N, Z
	MOV.D	Wns,Wd	将 W(ns):W(ns + 1) 中的双字内容送入 Wd	1	2	无
	MOV.D	Ws,Wnd	将 Ws 中的双字内容送入 W(nd + 1):W(nd)	1	2	无
MUL	MUL.SS	Wb,Ws,Wnd	{Wnd + 1, Wnd} = Signed(Wb) * Signed(Ws)	1	1	无
	MUL.SU	Wb,Ws,Wnd	{Wnd + 1, Wnd} = Signed(Wb) * Unsigned(Ws)	1	1	无
	MUL.US	Wb,Ws,Wnd	{Wnd + 1, Wnd} = Unsigned(Wb) * Signed(Ws)	1	1	无
	MUL.UU	Wb,Ws,Wnd	{Wnd + 1, Wnd} = Unsigned(Wb) * Unsigned(Ws)	1	1	无
	MUL.SU	Wb,#lit5,Wnd	{Wnd + 1, Wnd} = Signed(Wb) * Unsigned(lit5)	1	1	无
	MUL.UU	Wb,#lit5,Wnd	{Wnd + 1, Wnd} = Unsigned(Wb) * Unsigned(lit5)	1	1	无
	MUL	f	W3:W2 = f * WREG	1	1	无
NEG	NEG	f	$f = \overline{f} + 1$	1	1	C, DC, N, OV, Z
	NEG	f,WREG	$WREG = \overline{f} + 1$	1	1	C, DC, N, OV, Z
	NEG	Ws,Wd	$Wd = \overline{Ws} + 1$	1	1	C, DC, N, OV, Z
NOP	NOP		空操作	1	1	无
	NOPR		空操作	1	1	无
POP	POP	f	将栈顶 （TOS）的内容弹出到 f	1	1	无
	POP	Wdo	将栈顶 （TOS）的内容弹出到 Wdo	1	1	无
	POP.D	Wnd	将栈顶 （TOS）的内容弹出到 W(nd):W(nd + 1)	1	2	无
	POP.S		将影子寄存器的内容弹出到主寄存器	1	1	全部
PUSH	PUSH	f	将 f 的内容压入栈顶 （TOS）	1	1	无
	PUSH	Wso	将 Wso 的内容压入栈顶 （TOS）	1	1	无
	PUSH.D	Wns	将 W(ns):W(ns + 1) 的双字内容压入栈顶 （TOS）	1	2	无
	PUSH.S		将主寄存器中的内容压入影子寄存器	1	1	无

表 27-2： 指令集概述 （续）

汇编助记符	汇编语法		说明	字数	周期数	影响的状态标志
PWRSAV	PWRSAV	#lit1	进入休眠或空闲模式	1	1	WDTO, Sleep
RCALL	RCALL	Expr	相对调用	1	2	无
	RCALL	Wn	计算调用	1	2	无
REPEAT	REPEAT	#lit14	将下一条指令重复执行 lit14 + 1 次	1	1	无
	REPEAT	Wn	将下一条指令重复执行 (Wn) + 1 次	1	1	无
RESET	RESET		软件器件复位	1	1	无
RETFIE	RETFIE		从中断返回	1	3（2）	无
RETLW	RETLW	#lit10,Wn	返回并将立即数存入 Wn	1	3（2）	无
RETURN	RETURN		从子程序返回	1	3（2）	无
RLC	RLC	f	f = 对 f 执行带进位的循环左移	1	1	C, N, Z
	RLC	f,WREG	WREG = 对 f 执行带进位的循环左移	1	1	C, N, Z
	RLC	Ws,Wd	Wd = 对 Ws 执行带进位的循环左移	1	1	C, N, Z
RLNC	RLNC	f	f = 循环左移 f（不带进位）	1	1	N, Z
	RLNC	f,WREG	WREG = 循环左移 f（不带进位）	1	1	N, Z
	RLNC	Ws,Wd	Wd = 循环左移 Ws（不带进位）	1	1	N, Z
RRC	RRC	f	f = 对 f 执行带进位的循环右移	1	1	C, N, Z
	RRC	f,WREG	WREG = 对 f 执行带进位的循环右移	1	1	C, N, Z
	RRC	Ws,Wd	Wd = 对 Ws 执行带进位的循环右移	1	1	C, N, Z
RRNC	RRNC	f	f = 循环右移 f（不带进位）	1	1	N, Z
	RRNC	f,WREG	WREG = 循环右移 f（不带进位）	1	1	N, Z
	RRNC	Ws,Wd	Wd = 循环右移 Ws（不带进位）	1	1	N, Z
SE	SE	Ws,Wnd	Wnd = 符号扩展后的 Ws	1	1	C, N, Z
SETM	SETM	f	f = FFFFh	1	1	无
	SETM	WREG	WREG = FFFFh	1	1	无
	SETM	Ws	Ws = FFFFh	1	1	无
SL	SL	f	f = 左移 f	1	1	C, N, OV, Z
	SL	f,WREG	WREG = 左移 f	1	1	C, N, OV, Z
	SL	Ws,Wd	Wd = 左移 Ws	1	1	C, N, OV, Z
	SL	Wb,Wns,Wnd	Wnd = 将 Wb 左移 Wns 位	1	1	N, Z
	SL	Wb,#lit5,Wnd	Wnd = 将 Wb 左移 lit5 位	1	1	N, Z
SUB	SUB	f	f = f – WREG	1	1	C, DC, N, OV, Z
	SUB	f,WREG	WREG = f – WREG	1	1	C, DC, N, OV, Z
	SUB	#lit10,Wn	Wn = Wn – lit10	1	1	C, DC, N, OV, Z
	SUB	Wb,Ws,Wd	Wd = Wb – Ws	1	1	C, DC, N, OV, Z
	SUB	Wb,#lit5,Wd	Wd = Wb – lit5	1	1	C, DC, N, OV, Z
SUBB	SUBB	f	f = f – WREG – ($\overline{C}$)	1	1	C, DC, N, OV, Z
	SUBB	f,WREG	WREG = f – WREG – ($\overline{C}$)	1	1	C, DC, N, OV, Z
	SUBB	#lit10,Wn	Wn = Wn – lit10 – ($\overline{C}$)	1	1	C, DC, N, OV, Z
	SUBB	Wb,Ws,Wd	Wd = Wb – Ws – ($\overline{C}$)	1	1	C, DC, N, OV, Z
	SUBB	Wb,#lit5,Wd	Wd = Wb – lit5 – ($\overline{C}$)	1	1	C, DC, N, OV, Z
SUBR	SUBR	f	f = WREG – f	1	1	C, DC, N, OV, Z
	SUBR	f,WREG	WREG = WREG – f	1	1	C, DC, N, OV, Z
	SUBR	Wb,Ws,Wd	Wd = Ws – Wb	1	1	C, DC, N, OV, Z
	SUBR	Wb,#lit5,Wd	Wd = lit5 – Wb	1	1	C, DC, N, OV, Z
SUBBR	SUBBR	f	f = WREG – f – ($\overline{C}$)	1	1	C, DC, N, OV, Z
	SUBBR	f,WREG	WREG = WREG – f – ($\overline{C}$)	1	1	C, DC, N, OV, Z
	SUBBR	Wb,Ws,Wd	Wd = Ws – Wb – ($\overline{C}$)	1	1	C, DC, N, OV, Z
	SUBBR	Wb,#lit5,Wd	Wd = lit5 – Wb – ($\overline{C}$)	1	1	C, DC, N, OV, Z
SWAP	SWAP.b	Wn	Wn = 将 Wn 的前后两个半字节交换	1	1	无
	SWAP	Wn	Wn = 将 Wn 的两个字节相交换	1	1	无

表 27-2： 指令集概述 （续）

汇编助记符	汇编语法		说明	字数	周期数	影响的状态标志
TBLRDH	TBLRDH	Ws,Wd	将程序存储器中某个单元的 bit<23:16> 读入 Wd<7:0>	1	2	无
TBLRDL	TBLRDL	Ws,Wd	将程序存储器中某个单元的 bit<15:0> 读入 Wd	1	2	无
TBLWTH	TBLWTH	Ws,Wd	将 Ws<7:0> 写入程序存储器中某个单元的 bit<23:16>	1	2	无
TBLWTL	TBLWTL	Ws,Wd	将 Ws 写入程序存储器中某个单元的 bit<15:0>	1	2	无
ULNK	ULNK		释放堆栈帧	1	1	无
XOR	XOR	f	f = f .XOR.WREG	1	1	N, Z
	XOR	f,WREG	WREG = f .XOR.WREG	1	1	N, Z
	XOR	#lit10,Wn	Wd = lit10 .XOR.Wd	1	1	N, Z
	XOR	Wb,Ws,Wd	Wd = Wb .XOR.Ws	1	1	N, Z
	XOR	Wb,#lit5,Wd	Wd = Wb .XOR. lit5	1	1	N, Z
ZE	ZE	Ws,Wnd	Wnd = 零扩展后的 Ws	1	1	C, Z, N

注：

28.0 电气特性

本章将对 PIC24FJ64GA104 系列的电气特性进行概括介绍。其余信息在本文档的将来版本中提供。

下面列出了 PIC24FJ64GA104 系列器件的绝对最大值。器件长时间工作在最大值条件下可能会影响其可靠性。我们不建议使器件在或超过本规范指定的最大值条件下运行。

绝对最大值 (†)

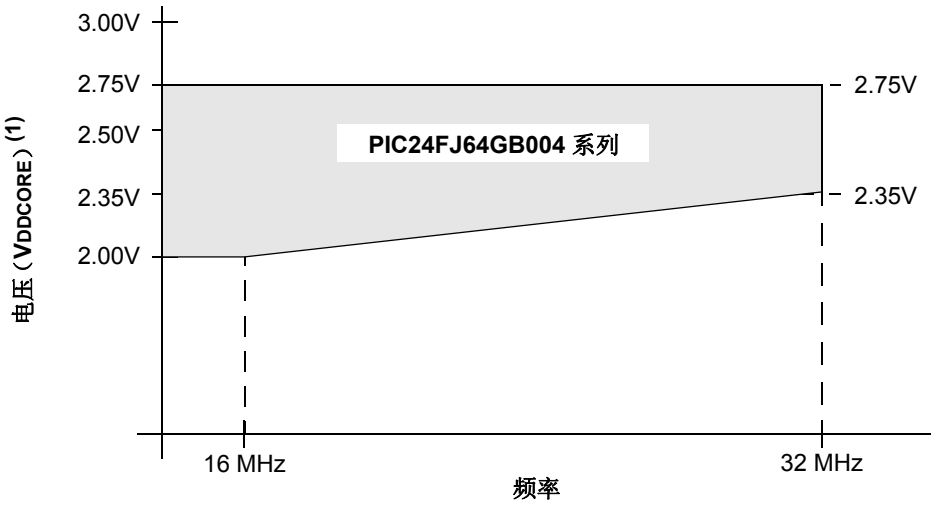
环境温度 ..... -40°C 至 +100°C
储存温度 ..... -65°C 至 +150°C
VDD 引脚相对于 VSS 的电压 ..... -0.3V 至 +4.0V
任一模拟 / 数字引脚和 $\overline{MCLR}$ 引脚相对于 VSS 的电压 ..... -0.3V 至 （VDD + 0.3V）
任一仅用作数字功能的引脚相对于 VSS 的电压 ..... -0.3V 至 +6.0V
VDDCORE 引脚相对于 VSS 的电压 ..... -0.3V 至 +3.0V
流出 VSS 引脚的最大电流 ..... 300 mA
流入 VDD 引脚的最大电流 **（注 1）** ..... 250 mA
任一 I/O 引脚的最大灌电流 ..... 25 mA
任一 I/O 引脚的最大拉电流 ..... 25 mA
所有端口的最大灌电流 ..... 200 mA
所有端口的最大拉电流 **（注 1）** ..... 200 mA

注 1： 允许的最大电流由器件最大功耗决定 （见表 28-1）。

† 注：如果器件工作条件超过上述 “绝对最大值”，可能引起器件永久性损坏。这仅是极限参数，我们不建议器件工作在极限值甚至超过上述极限值。器件长时间工作在极限条件下可能会影响其可靠性。

28.1 直流特性

图 28-1： PIC24FJ64GA104 系列电压——频率关系图 （工业级）



对于 16 MHz 和 32 MHz 之间的频率， FMAX = (45.7 MHz/V) * (VDDCORE – 2V) + 16 MHz。

注 1： 当稳压器被禁止时， VDD 和 VDDCORE 必须保持在 VDDCORE ≤ VDD ≤ 3.6V。

表 28-1： 温度工作条件

参数	符号	最小值	典型值	最大值	单位
PIC24FJ64GB004 系列：					
工作结温范围	TJ	-40	—	+125	°C
工作环境温度范围	TA	-40	—	+85	°C
功耗： 芯片内部功耗： PINT = VDD x (IDD – Σ IOH) I/O 引脚功耗： PI/O = Σ ({VDD – VOH} x IOH) + Σ (VOL x IOL)	PD	PINT + PI/O			W
最大允许功耗	PDMAX	(TJ – TA)/θJA			W

表 28-2： 温度封装特性

特性	符号	典型值	最大值	单位	注
封装热阻， 300 mil SOIC	θJA	49	—	°C/W	（注 1）
封装热阻， 6x6x0.9 mm QFN	θJA	33.7	—	°C/W	（注 1）
封装热阻， 8x8x1 mm QFN	θJA	28	—	°C/W	（注 1）
封装热阻， 10x10x1 mm TQFP	θJA	39.3	—	°C/W	（注 1）

注 1： 通过封装模拟获得结点与环境的热阻值 θJA。

表 28-3： 直流特性：温度和电压规范

直流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）				
参数编号	**符号**	**特性**	**最小值**	**典型值 (1)**	**最大值**	**单位**	**条件**
工作电压							
DC10	供电电压						
	VDD		2.2	—	3.6	V	使能稳压器
	VDD		VDDCORE	—	3.6	V	禁止稳压器
	VDDCORE		2.0	—	2.75	V	禁止稳压器
DC12	VDR	RAM 数据保持电压 (2)	1.5	—	—	V	
DC16	VPOR	确保内部上电复位信号的 VDD 启动电压	—	VSS	—	V	
DC17	SVDD	确保内部上电复位信号的 VDD 上升速率	0.05	—	—	V/ms	0-3.3V/0.1s 0-2.5V/60 ms
DC18	VBOR	欠压复位电压	—	2.05	—	V	

注 1： 除非另外声明，否则 “典型值”栏中的数据均为 3.3V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

2： 这是在不丢失 RAM 数据的前提下， VDD 的下限值。

表 28-4： 直流特性：工作电流 （IDD）

直流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）			
参数编号	典型值 (1)	最大值	单位	条件		
工作电流 （IDD） (2)						
DC21	0.24	0.395	mA	-40°C	2.0V(3)	0.5 MIPS
DC21a	0.25	0.395	mA	+25°C		
DC21b	0.25	0.395	mA	+85°C		
DC21c	0.44	0.78	mA	-40°C	3.3V(4)	
DC21d	0.41	0.78	mA	+25°C		
DC21e	0.41	0.78	mA	+85°C		
DC20	0.5	0.75	mA	-40°C	2.0V(3)	1 MIPS
DC20a	0.5	0.75	mA	+25°C		
DC20b	0.5	0.75	mA	+85°C		
DC20d	0.75	1.4	mA	-40°C	3.3V(4)	
DC20e	0.75	1.4	mA	+25°C		
DC20f	0.75	1.4	mA	+85°C		
DC23	2.0	3.0	mA	-40°C	2.0V(3)	4 MIPS
DC23a	2.0	3.0	mA	+25°C		
DC23b	2.0	3.0	mA	+85°C		
DC23d	2.9	4.2	mA	-40°C	3.3V(4)	
DC23e	2.9	4.2	mA	+25°C		
DC23f	2.9	4.2	mA	+85°C		
DC24	10.5	15.5	mA	-40°C	2.5V(3)	16 MIPS
DC24a	10.5	15.5	mA	+25°C		
DC24b	10.5	15.5	mA	+85°C		
DC24d	11.3	15.5	mA	-40°C	3.3V(4)	
DC24e	11.3	15.5	mA	+25°C		
DC24f	11.3	15.5	mA	+85°C		
DC31	15.0	18.0	μA	-40°C	2.0V(3)	LPRC （31 kHz）
DC31a	15.0	19.0	μA	+25°C		
DC31b	20.0	36.0	μA	+85°C		
DC31d	57.0	120.0	μA	-40°C	3.3V(4)	
DC31e	57.0	125.0	μA	+25°C		
DC31f	95.0	160.0	μA	+85°C		

注 1： 除非另外声明，否则 “典型值”栏中的数据均为 3.3V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

2： 供电电流主要受工作电压和频率的影响。其他因素，如 I/O 引脚负载和开关速率、振荡器类型、内部代码执行模式和温度也会对电流消耗产生影响。所有 IDD 测量值的测试条件为：OSCI 使用轨到轨外部方波驱动。所有 I/O 引脚配置为输入且被上拉至 VDD。
$\overline{MCLR}$ = VDD； WDT 和 FSCM 被禁止。CPU、SRAM、程序存储器和数据存储器处于工作状态。没有外设模块正在工作，所有的外设模块禁止 （Peripheral Module Disable， PMD）位被置 1。

3： 禁止片上稳压器 （DISVREG 连接到 VDD）。

4： 使能片上稳压器 （DISVREG 连接到 VSS） 。使能低压检测 （Low-Voltage Detect， LVD）和欠压检测 （Brown-out Detect， BOD）。

表 28-5：　直流特性：空闲电流 （IIDLE）

直流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度　-40°C ≤ TA ≤ +85°C （工业级）			
参数编号	典型值 (1)	最大值	单位	条件		
空闲电流 （IIDLE） (2)						
DC41	67	100	μA	-40°C	2.0V(3)	0.5 MIPS
DC41a	68	100	μA	+25°C		
DC41b	74	100	μA	+85°C		
DC41c	166	265	μA	-40°C	3.3V(4)	
DC41d	167	265	μA	+25°C		
DC41e	177	265	μA	+85°C		
DC40	125	180	μA	-40°C	2.0V(3)	1 MIPS
DC40a	125	180	μA	+25°C		
DC40b	125	180	μA	+85°C		
DC40d	210	350	μA	-40°C	3.3V(4)	
DC40e	210	350	μA	+25°C		
DC40f	210	350	μA	+85°C		
DC43	0.5	0.6	mA	-40°C	2.0V(3)	4 MIPS
DC43a	0.5	0.6	mA	+25°C		
DC43b	0.5	0.6	mA	+85°C		
DC43d	0.75	0.95	mA	-40°C	3.3V(4)	
DC43e	0.75	0.95	mA	+25°C		
DC43f	0.75	0.95	mA	+85°C		
DC47	2.6	3.3	mA	-40°C	2.5V(3)	16 MIPS
DC47a	2.6	3.3	mA	+25°C		
DC47b	2.6	3.3	mA	+85°C		
DC47c	2.9	3.5	mA	-40°C	3.3V(4)	
DC47d	2.9	3.5	mA	+25°C		
DC47e	2.9	3.5	mA	+85°C		
DC50	0.8	1.0	mA	-40°C	2.0V(3)	FRC （4 MIPS）
DC50a	0.8	1.0	mA	+25°C		
DC50b	0.8	1.0	mA	+85°C		
DC50d	1.1	1.3	mA	-40°C	3.3V(4)	
DC50e	1.1	1.3	mA	+25°C		
DC50f	1.1	1.3	mA	+85°C		
DC51	2.4	8.0	μA	-40°C	2.0V(3)	LPRC （31 kHz）
DC51a	2.2	8.0	μA	+25°C		
DC51b	7.2	21.0	μA	+85°C		
DC51d	38	55	μA	-40°C	3.3V(4)	
DC51e	44	60	μA	+25°C		
DC51f	70	100	μA	+85°C		

注　1： 除非另外声明，否则 “典型值”栏中的数据均为 3.3V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

2： 基本 IIDLE 电流是在内核不工作且 OSCI 使用轨到轨外部方波驱动的条件下进行测量的。所有 I/O 引脚配置为输入且被上拉至 VDD。MCLR = VDD； WDT 和 FSCM 被禁止。没有外设模块正在工作，所有的外设模块禁止（PMD）位被置 1。

3： 禁止片上稳压器 （DISVREG 连接到 VDD）。

4： 使能片上稳压器 （DISVREG 连接到 VSS）。使能低压检测 （LVD）和欠压检测 （BOD）。

表 28-6： 直流特性：掉电基本电流 （IPD）

直流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）			
参数编号	典型值 (1)	最大值	单位	条件		
掉电电流 （IPD） (2)						
DC60	0.05	1.0	μA	-40°C	2.0V(3)	基本掉电电流 (5)
DC60a	0.2	1.0	μA	+25°C		
DC60i	2.0	6.5	μA	+60°C		
DC60b	3.5	12.0	μA	+85°C		
DC60c	0.1	1.0	μA	-40°C	2.5V(3)	
DC60d	0.4	1.0	μA	+25°C		
DC60j	2.5	15	μA	+60°C		
DC60e	4.2	25	μA	+85°C		
DC60f	3.3	9.0	μA	-40°C	3.3V(4)	
DC60g	3.3	10.0	μA	+25°C		
DC60k	5.0	20.0	μA	+60°C		
DC60h	7.0	30.0	μA	+85°C		
DC70c	.003	0.2	μA	-40°C	2.5V(4)	深度休眠时的基本电流
DC70d	0.02	0.2	μA	+25°C		
DC70j	0.2	0.35	μA	+60°C		
DC70e	.51	1.5	μA	+85°C		
DC70f	.01	0.3	μA	-40°C	3.3V(4)	
DC70g	0.04	0.3	μA	+25°C		
DC70k	0.2	0.5	μA	+60°C		
DC70h	.71	2.0	μA	+85°C		

注 1： 除非另外声明，否则 “典型值”栏中的数据均为 3.3V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

2： 基本 IPD 是在器件处于休眠模式（所有外设和时钟都关闭）时进行测量的。所有 I/O 引脚配置为输入且被上拉到高电平。 WDT 等外设也都被关闭， PMSLP 位清零，所有未用外设的外设模块禁止 （PMD）位置 1。

3： 禁止片上稳压器 （DISVREG 连接到 VDD）。

4： 使能片上稳压器 （DISVREG 连接到 VSS）。使能低压检测 （LVD）和欠压检测 （BOD）。

5： Δ 电流为当模块使能时额外消耗的电流。此电流应被加到基本 IPD 电流。

表 28-7： 直流特性：掉电外设模块 Δ 电流 （IPD）

直流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）			
参数编号	典型值 (1)	最大值	单位	条件		
Δ 掉电电流 （IPD）：PMD 位置 1， PMSLP 位为 0(2)						
DC61	0.2	0.7	μA	-40°C	2.0V(3)	31 kHz LPRC 振荡器 + RTCC、WDT、 DSWDT 或 Timer1：ΔILPRC(5)
DC61a	0.2	0.7	μA	+25°C		
DC61i	0.2	0.7	μA	+60°C		
DC61b	0.23	0.7	μA	+85°C		
DC61c	0.25	0.9	μA	-40°C	2.5V(3)	
DC61d	0.25	0.9	μA	+25°C		
DC61j	0.25	0.9	μA	+60°C		
DC61e	0.28	0.9	μA	+85°C		
DC61f	0.6	1.5	μA	-40°C	3.3V(4)	
DC61g	0.6	1.5	μA	+25°C		
DC61k	0.6	1.5	μA	+60°C		
DC61h	0.8	1.5	μA	+85°C		
DC62	0.5	1.0	μA	-40°C	2.0V(3)	低驱动能力， 32 kHz 晶振 + RTCC、 DSWDT 或 Timer1：ΔISOSC ；SOSCSEL = 01
DC62a	0.5	1.0	μA	+25°C		
DC62i	0.5	1.0	μA	+60°C		
DC62b	0.5	1.3	μA	+85°C		
DC62c	0.7	1.5	μA	-40°C	2.5V(3)	
DC62d	0.7	1.5	μA	+25°C		
DC62j	0.7	1.5	μA	+60°C		
DC62e	0.7	1.8	μA	+85°C		
DC62f	1.5	2.0	μA	-40°C	3.3V(4)	
DC62g	1.5	2.0	μA	+25°C		
DC62k	1.5	2.0	μA	+60°C		
DC62h	1.5	2.5	μA	+85°C		

注 1： 除非另外声明，否则 “典型值”栏中的数据均为 3.3V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

2： 外设 IPD 是在器件处于休眠模式 （所有外设和时钟都关闭）时进行测量的。所有 I/O 引脚配置为输入且被上拉到高电平。只有正在进行测量的外设或时钟是使能的。PMSLP 位清零，所有未用外设的外设模块禁止位 （PMD）置 1。

3： 禁止片上稳压器 （DISVREG 连接到 VDD）。

4： 使能片上稳压器 （DISVREG 连接到 VSS）。使能低压检测 （LVD）和欠压检测 （BOD）。

5： Δ 电流为当模块使能时额外消耗的电流。此电流应被加到基本 IPD 电流。

表 28-7： 直流特性：掉电外设模块 Δ 电流 （IPD）（续）

直流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）			
参数编号	典型值 (1)	最大值	单位	条件		
Δ 掉电电流 （IPD）：PMD 位置 1， PMSLP 位为 0 (2)						
DC63	1.8	2.3	μA	-40°C	2.0V (3)	32 kHz 晶振 + RTCC、DSWDT 或 Timer1：ΔISOSC； SOSCSEL = 11 (5)
DC63a	1.8	2.7	μA	+25°C		
DC63i	1.8	3.0	μA	+60°C		
DC63b	1.8	3.0	μA	+85°C		
DC63c	2	2.7	μA	-40°C	2.5V (3)	
DC63d	2	2.9	μA	+25°C		
DC63j	2	3.2	μA	+60°C		
DC63e	2	3.5	μA	+85°C		
DC63f	2.25	3.0	μA	-40°C	3.3V (4)	
DC63g	2.25	3.0	μA	+25°C		
DC63k	2.25	3.3	μA	+60°C		
DC63h	2.25	3.5	μA	+85°C		
DC71c	.001	0.25	μA	-40°C	2.5V (4)	深度休眠 BOR：ΔIDSBOR
DC71d	.03	0.25	μA	+25°C		
DC71j	0.05	0.60	μA	+60°C		
DC71e	.08	2.0	μA	+85°C		
DC71f	.001	0.50	μA	-40°C	3.3V (4)	
DC71g	.03	0.50	μA	+25°C		
DC71k	0.05	0.75	μA	+60°C		
DC71h	.08	2.50	μA	+85°C		

注 1： 除非另外声明，否则 “典型值”栏中的数据均为 3.3V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

2： 外设 IPD 是在器件处于休眠模式 （所有外设和时钟都关闭）时进行测量的。所有 I/O 引脚配置为输入且被上拉到高电平。只有正在进行测量的外设或时钟是使能的。PMSLP 位清零，所有未用外设的外设模块禁止位 （PMD）置 1。

3： 禁止片上稳压器 （DISVREG 连接到 VDD）。

4： 使能片上稳压器 （DISVREG 连接到 VSS）。使能低压检测 （LVD）和欠压检测 （BOD）。

5： Δ 电流为当模块使能时额外消耗的电流。此电流应被加到基本 IPD 电流。

表 28-8：　　直流特性：I/O 引脚输入规范

直流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）				
参数编号	符号	特性	最小值	典型值 (1)	最大值	单位	条件
	VIL	**输入低电压 (4)**					
DI10		I/O 引脚，带 ST 缓冲器	VSS	—	0.2 VDD	V	
DI11		I/O 引脚，带 TTL 缓冲器	VSS	—	0.15 VDD	V	
DI15		$\overline{MCLR}$	VSS	—	0.2 VDD	V	
DI16		OSC1 （XT 模式）	VSS	—	0.2 VDD	V	
DI17		OSC1 （HS 模式）	VSS	—	0.2 VDD	V	
DI18		I/O 引脚，带 I^2C™ 缓冲器：	VSS	—	0.3 VDD	V	
DI19		I/O 引脚，带 SMBus 缓冲器：	VSS	—	0.8	V	使能 SMBus
	VIH	**输入高电压 (4)**					
DI20		I/O 引脚，带 ST 缓冲器：					
		带模拟功能，	0.8 VDD	—	VDD	V	
		仅数字功能	0.8 VDD	—	5.5	V	
DI21		I/O 引脚，带 TTL 缓冲器：					
		带模拟功能，	0.25 VDD + 0.8	—	VDD	V	
		仅数字功能	0.25 VDD + 0.8	—	5.5	V	
DI25		$\overline{MCLR}$	0.8 VDD	—	VDD	V	
DI26		OSC1 （XT 模式）	0.7 VDD	—	VDD	V	
DI27		OSC1 （HS 模式）	0.7 VDD	—	VDD	V	
DI28		I/O 引脚，带 I^2C 缓冲器：					
		带模拟功能，	0.7 VDD	—	VDD	V	
		仅数字功能	0.7 VDD	—	5.5	V	
DI29		I/O 引脚，带 SMBus 缓冲器：					2.5V ≤ VPIN ≤ VDD
		带模拟功能，	2.1	—	VDD	V	
		仅数字功能	2.1	—	5.5	V	
DI30	ICNPU	**CNx 上拉电流**	50	250	400	μA	VDD = 3.3V， VPIN = VSS
	IIL	**输入泄漏电流 (2,3)**					
DI50		I/O 端口	—	—	±50	nA	VSS ≤ VPIN ≤ VDD， 引脚处于高阻态
DI51		模拟输入引脚	—	—	±50	nA	VSS ≤ VPIN ≤ VDD， 引脚处于高阻态
DI55		$\overline{MCLR}$	—	—	±50	nA	VSS ≤ VPIN ≤ VDD
DI56		OSC1	—	—	±50	nA	VSS ≤ VPIN ≤ VDD， XT 和 HS 模式

注　1： 除非另外声明，否则 “典型值”栏中的数据均为 3.3V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

2： $\overline{MCLR}$ 引脚上的泄漏电流主要取决于所施加的电压。规定电压为正常工作条件下的电压。在不同的输入电压下可能测得更高的泄漏电流。

3： 负电流定义为引脚的拉电流。

4： 请参见表 1-2 了解 I/O 引脚缓冲器类型。

表 28-9： 直流特性：I/O 引脚输出规范

直流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）				
参数编号	符号	特性	最小值	典型值 (1)	最大值	单位	条件
DO10	VOL	输出低电压 I/O 端口	— —	— —	0.4 0.4	V V	IOL = 8.5 mA， VDD = 3.6V IOL = 5.0 mA， VDD = 2.0V
DO20	VOH	输出高电压 I/O 端口	3.0 2.4 1.65 1.4	— — — —	— — — —	V V V V	IOH = -3.0 mA， VDD = 3.6V IOH = -6.0 mA， VDD = 3.6V IOH = -1.0 mA， VDD = 2.0V IOH = -3.0 mA， VDD = 2.0V

注 1： 除非另外声明，否则“典型值”栏中的数据均为3.3V和25°C条件下的值。这些参数仅供设计参考，未经测试。

表 28-10： 直流特性：程序存储器

直流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）				
参数编号	符号	特性	最小值	典型值 (1)	最大值	单位	条件
D130	EP	单元耐擦写能力	10,000	—	—	E/W	-40°C 至 +85°C
D131	VPR	读操作时的 VDD	VMIN	—	3.6	V	VMIN = 最小工作电压
D132A D132B	VPEW	自定时写的供电电压 VDDCORE VDD	2.25 2.35	— —	3.6 3.6	V V	
D133A	TIW	自定时写周期时间	—	3	—	ms	
D133B	TIE	自定时页擦除时间	40	—	—	ms	
D134	TRETD	特性保持时间	20	—	—	年	假设没有违反其他规范
D135	IDDP	编程时的供电电流	—	7	—	mA	

注 1： 除非另外声明，否则 “典型值”栏中的数据均为 3.3V 和 25°C 条件下的值。

表 28-11： 比较器规范

工作条件：2.0V < VDD < 3.6V， -40°C < TA < +85°C （除非另外声明）							
参数编号	**符号**	**特性**	**最小值**	**典型值**	**最大值**	**单位**	**备注**
D300	VIOFF	输入失调电压 *	—	20	40	mV	
D301	VICM	输入共模电压 *	0	—	VDD	V	
D302	CMRR	共模抑制比 *	55	—	—	dB	
300	TRESP	响应时间 *(1)	—	150	400	ns	
301	TMC2OV	比较器模式变为输出有效的时间 *	—	—	10	µs	

* 参数为特性值，未经测试。

注 1： 响应时间是在比较器的一个输入端电压为 (VDD – 1.5)/2，而另一个输入端从 VSS 跳变到 VDD 时测得的。

表 28-12： 比较器参考电压规范

工作条件：2.0V < VDD < 3.6V， -40°C < TA < +85°C （除非另外声明）							
参数编号	**符号**	**特性**	**最小值**	**典型值**	**最大值**	**单位**	**备注**
VRD310	CVRES	分辨率	VDD/24	—	VDD/32	LSb	
VRD311	CVRAA	绝对精度	—	—	AVDD –1.5	LSb	
VRD312	CVRUR	单位电阻值 （R）	—	2k	—	Ω	
VR310	TSET	稳定时间 (1)	—	—	10	µs	

注 1： 稳定时间是在 CVRR = 1 并且 CVR<3:0> 位从 0000 跳变到 1111 时测得的。

表 28-13： 内部稳压器规范

工作条件：-40°C < TA < +85°C （除非另外声明）							
参数编号	**符号**	**特性**	**最小值**	**典型值**	**最大值**	**单位**	**备注**
	VBG	带隙参考电压	1.14	1.2	1.26	V	
	TBG	带隙参考电压启动时间	—	1	—	ms	
	VRGOUT	稳压器输出电压	2.35	2.5	2.75	V	
	CEFC	外部滤波电容值	4.7	10	—	µF	建议串联电阻 < 3Ω； 必须 < 5Ω。

28.2 交流特性和时序参数

本节包含的信息说明了 PIC24FJ64GA104 系列器件的交流特性和时序参数。

表 28-14： 温度和电压规范——交流

交流特性	**标准工作条件：2.0V 至 3.6V （除非另外声明）** 工作温度 -40°C ≤ TA ≤ +85°C （工业级） 工作电压 VDD 范围如**第 28.1 节 “直流特性”**中所述。

图 28-2： 器件时序规范的负载条件

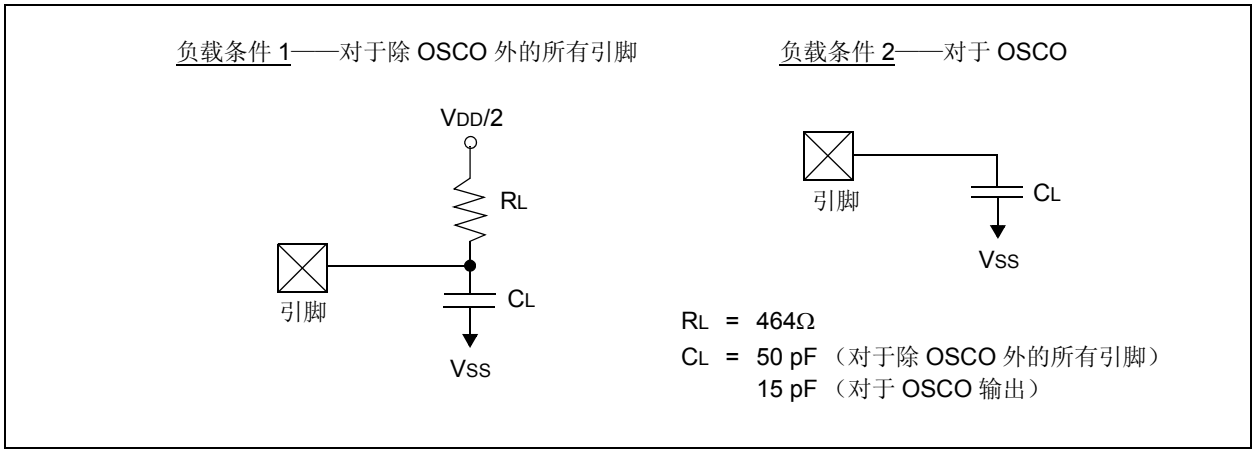


表 28-15： 输出引脚上的容性负载要求

参数编号	符号	特性	最小值	典型值 (1)	最大值	单位	条件
DO50	COSC2	OSCO/CLKO 引脚	—	—	15	pF	当外部时钟用于驱动 OSCI 时处于 XT 和 HS 模式下。
DO56	CIO	所有 I/O 引脚和 OSCO	—	—	50	pF	EC 模式。
DO58	CB	SCLx 和 SDAx	—	—	400	pF	在 I^2C™ 模式下。

注 1： 除非另外声明，否则“典型值”栏中的数据均为3.3V和25°C条件下的值。这些参数仅供设计参考，未经测试。

图 28-3：　外部时钟时序

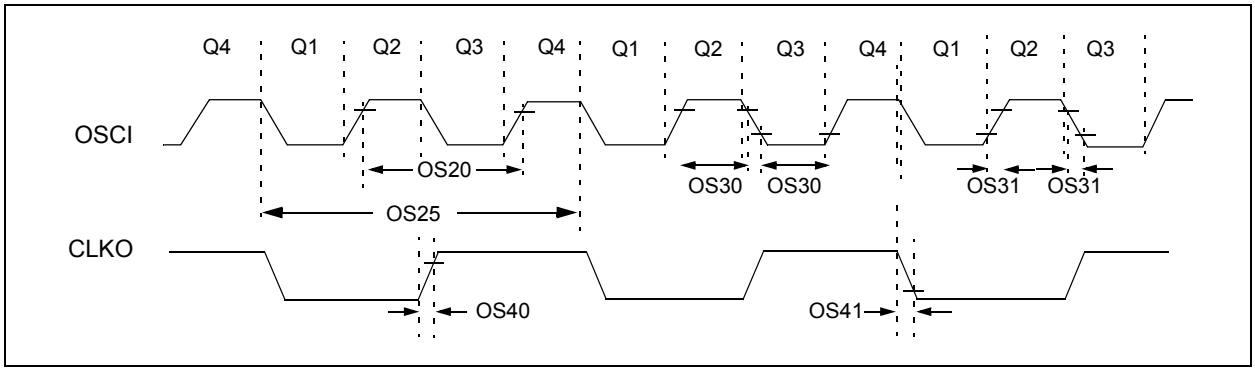


表 28-16：　外部时钟时序要求

交流特性			标准工作条件：2.5V 至 3.6V （除非另外声明） 工作温度　-40°C ≤ TA ≤ +85°C （工业级）				
参数编号	**符号**	**特性**	**最小值**	**典型值 (1)**	**最大值**	**单位**	**条件**
OS10	FOSC	外部 CLKI 频率 （外部时钟仅允许运行于 EC 模式）	DC 4	— —	32 48	MHz MHz	EC ECPLL
		振荡器频率	3 4 10 12 31	— — — — —	10 8 32 32 33	MHz MHz MHz MHz kHz	XT XTPLL HS HSPLL SOSC
OS20	TOSC	TOSC = 1/FOSC	—	—	—	—	FOSC 值见参数 OS10
OS25	TCY	指令周期 (2)	62.5	—	DC	ns	
OS30	TosL，TosH	外部时钟输入 （OSCI）高电平或低电平时间	0.45 x TOSC	—	—	ns	EC
OS31	TosR，TosF	外部时钟输入 （OSCI）上升或下降时间	—	—	20	ns	EC
OS40	TckR	CLKO 上升时间 (3)	—	6	10	ns	
OS41	TckF	CLKO 下降时间 (3)	—	6	10	ns	

注　1： 除非另外声明，否则 “典型值”栏中的数据均为 3.3V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

2： 指令周期 （TCY）等于输入振荡器时基周期的两倍。所有规定值均为基于针对特定振荡器类型，器件在标准工作条件下执行代码时的特性数据。超出这些规定的限定值，可能导致振荡器运行不稳定和 / 或导致电流消耗超出预期值。所有器件在测试 “最小”值时，都在 OSCI/CLKI 引脚连接了外部时钟。当使用了外部时钟输入时，所有器件的 “最大”周期时间限制为 “DC”（无时钟）。

3： 测量在 EC 模式下进行。在 OSCO 引脚上测量 CLKO 信号。CLKO 在 Q1-Q2 周期（1/2 TCY）中为低电平，在 Q3-Q4 周期 （1/2 TCY）中为高电平。

表 28-17： PLL 时钟时序规范 （VDD = 2.0V 至 3.6V）

交流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）				
参数编号	符号	特性 (1)	最小值	典型值 (2)	最大值	单位	条件
OS50	FPLLI	PLL 输入频率范围	3 3	— —	8 6	MHz MHz	ECPLL、 HSPLL 和 XTPLL 模式， -40°C ≤ TA ≤ +85°C ECPLL、 HSPLL 和 XTPLL 模式， -40°C ≤ TA ≤ +125°C
OS51	FSYS	PLL 输出频率范围	8 8	— —	32 24	MHz MHz	-40°C ≤ TA ≤ +85°C -40°C ≤ TA ≤ +125°C
OS52	TLOCK	PLL 起振时间（锁定时间）	—	—	2	ms	
OS53	DCLK	CLKO 稳定性（抗抖动性）	-2	1	2	%	在 100 ms 时间段内测量

注 1： 这些参数为特性值，但生产时未经测试。

2： 除非另外声明，否则“典型值”栏中的数据均为3.3V和25°C条件下的值。这些参数仅供设计参考，未经测试。

表 28-18： 内部 RC 振荡器规范

交流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）				
参数编号	符号	特性 (1)	最小值	典型值	最大值	单位	条件
	TFRC	FRC 起振时间	—	15	—	μs	
	TLPRC	LPRC 起振时间	—	500	—	μs	

表 28-19： 内部 RC 振荡器精度

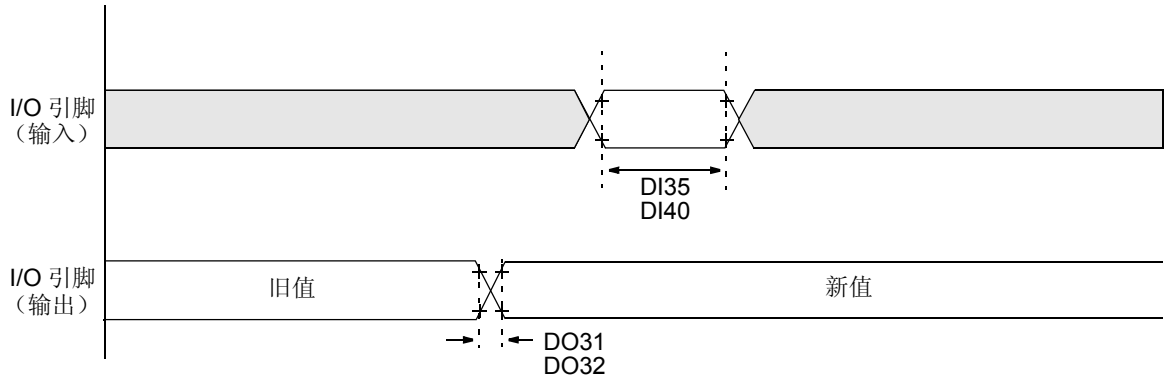
交流特性		标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）				
参数编号	特性	最小值	典型值	最大值	单位	条件
F20	频率为 8 MHz 时的 FRC 精度 (1,3)	-1.25	±0.25	1.0	%	-40°C ≤ TA ≤ +85°C， 3.0V ≤ VDD ≤ 3.6V
F21	频率为 31 kHz 时的 LPRC 精度 (2)	-15	—	15	%	-40°C ≤ TA ≤ +85°C， 3.0V ≤ VDD ≤ 3.6V

注 1： 频率在 25°C 和 3.3V 条件下校准。 OSCTUN 位可用来补偿温度漂移。

2： LPRC 频率将随 VDD 的变化而变化。

3： 为获得该精度，加在单片机封装上的物理应力 （例如，通过弯曲 PCB）必须保持最小。

图 28-4：　　CLKO 和 I/O 时序特性



注：负载条件请参见图 28-2。

表 28-20：　　CLKO 和 I/O 时序要求

交流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度　-40°C ≤ TA ≤ +85°C （工业级）				
参数编号	符号	特性	最小值	典型值 (1)	最大值	单位	条件
DO31	TIOR	端口输出上升时间	—	10	25	ns	
DO32	TIOF	端口输出下降时间	—	10	25	ns	
DI35	TINP	INTx 引脚高电平或低电平时间 （输出）	20	—	—	ns	
DI40	TRBP	CNx 高电平或低电平时间（输入）	2	—	—	TCY	

注　1： 除非另外声明，否则 “典型值”栏中的数据均为 3.3V 和 25°C 条件下的值。

表 28-21：　　复位、上电延时定时器和欠压复位时序要求

交流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度　-40°C ≤ TA ≤ +85°C （工业级）				
参数编号	符号	特性	最小值	典型值 (1)	最大值	单位	条件
SY10	TmcL	$\overline{\text{MCLR}}$ 脉冲宽度 （低电平）	2	—	—	μs	
SY11	TPWRT	上电延时定时器周期	—	64	—	ms	
SY12	TPOR	上电复位延时	—	2	—	μs	
SY13	TIOZ	自 $\overline{\text{MCLR}}$ 低电平或看门狗定时器复位起 I/O 处于高阻态的时间	—	—	100	ns	
SY25	TBOR	欠压复位脉冲宽度	1	—	—	μs	VDD ≤ VBOR
	TRST	内部状态复位时间	—	50	—	μs	
	TDSWU	从深度休眠唤醒的时间	—	200	—	μs	基于在 VCAP 下对 10 μF 电容完全放电。包括 TPOR 和 TRST。
	TPM		— —	10 190	— —	μs μs	从休眠唤醒， PMSLP = 0 且 WUTSEL<1:0> = 11

注　1： 除非另外声明，否则 “典型值”栏中的数据均为 3.3V 和 25°C 条件下的值。

表 28-22： ADC 模块规范

交流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C				
参数编号	**符号**	**特性**	**最小值**	**典型值**	**最大值**	**单位**	**条件**
器件电源							
AD01	AVDD	模块电源 VDD	VDD – 0.3 和 2.0 中的 较大值	—	VDD + 0.3 和 3.6 中的 较小值	V	
AD02	AVSS	模块电源 VSS	VSS – 0.3	—	VSS + 0.3	V	
参考输入							
AD05	VREFH	参考电压高电压	AVSS + 1.7	—	AVDD	V	
AD06	VREFL	参考电压低电压	AVSS	—	AVDD – 1.7	V	
AD07	VREF	绝对参考电压	AVSS – 0.3	—	AVDD + 0.3	V	
模拟输入							
AD10	VINH-VINL	满量程输入范围	VREFL	—	VREFH	V	（注 2）
AD11	VIN	绝对输入电压	AVSS – 0.3	—	AVDD + 0.3	V	
AD12	VINL	绝对 VINL 输入电压	AVSS – 0.3	—	AVDD/2	V	
AD13	—	泄漏电流	—	±0.001	±0.610	μA	VINL = AVSS = VREFL = 0V， AVDD = VREFH = 3V， 信号源阻抗 = 2.5 kΩ
AD17	RIN	模拟信号源的推荐阻抗	—	—	2.5K	Ω	10 位
ADC 精度							
AD20b	NR	分辨率	—	10	—	位	
AD21b	INL	积分非线性误差	—	±1	<±2	LSb	VINL = AVSS = VREFL = 0V， AVDD = VREFH = 3V
AD22b	DNL	微分非线性误差	—	±0.5	<±1.25	LSb	VINL = AVSS = VREFL = 0V， AVDD = VREFH = 3V
AD23b	GERR	增益误差	—	±1	±3	LSb	VINL = AVSS = VREFL = 0V， AVDD = VREFH = 3V
AD24b	EOFF	失调误差	—	±1	±2	LSb	VINL = AVSS = VREFL = 0V， AVDD = VREFH = 3V
AD25b	—	单调性 (1)	—	—	—	—	保证

注 1： ADC 转换结果不会因输入电压的增加而减小，并且不会丢失编码。

2： 测量采用外部 VREF+ 和 VREF- 用作 ADC 参考电压。

表 28-23： ADC 转换时序要求 (1)

交流特性			标准工作条件：2.0V 至 3.6V （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C				
参数编号	符号	特性	最小值	典型值	最大值	单位	条件
时钟参数							
AD50	TAD	ADC 时钟周期	75	—	—	ns	TCY = 75 ns， AD1CON3 处于默认状态
AD51	tRC	ADC 内部 RC 振荡器周期	—	250	—	ns	
转换速率							
AD55	tCONV	转换时间	—	12	—	TAD	
AD56	FCNV	吞吐率	—	—	500	ksps	AVDD > 2.7V
AD57	tSAMP	采样时间	—	1	—	TAD	
时钟参数							
AD61	tPSS	从采样位 （SAMP）置 1 到采样启动的时间	2	—	3	TAD	

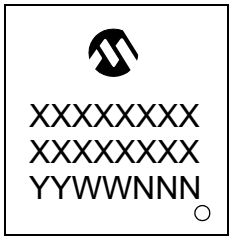
注 1： 因为采样电容最终将释放电荷，因此低于 10 kHz 的时钟频率可能影响线性性能，尤其是在温度较高时。

注：

29.0 封装信息

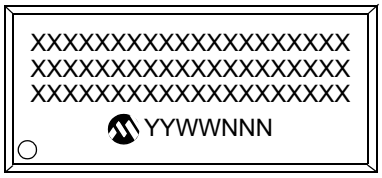
29.1 封装标识信息

28 引脚 QFN

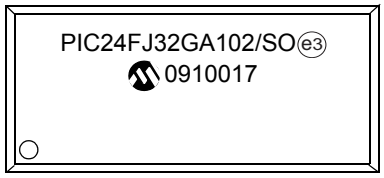


示例

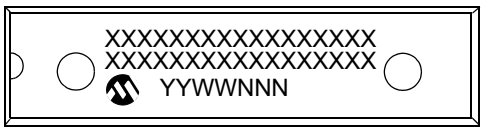
28 引脚 SOIC（0.300 英寸）



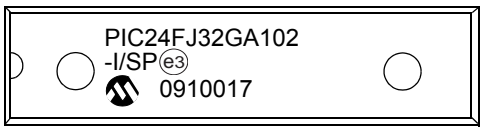
示例



28 引脚 SPDIP

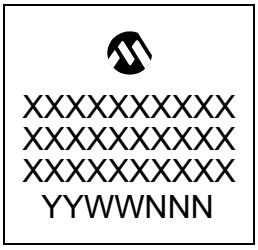


示例



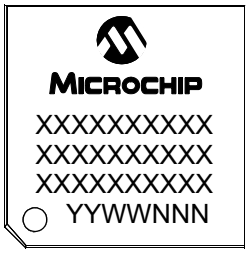
图注：	XX...X	客户指定信息
	Y	年份代码 （日历年的最后一位数字）
	YY	年份代码 （日历年的最后两位数字）
	WW	星期代码 （一月一日的星期代码为 “01”）
	NNN	以字母数字排序的追踪代码
	(e3)	雾锡 （Matte Tin， Sn）的 JEDEC 无铅标志
	*	本封装为无铅封装。 JEDEC 无铅标志 （(e3)）标示于此种封装的外包装上。
注：	Microchip 元器件编号如果无法在同一行内完整标注，将换行标出，因此会限制表示客户指定信息的字符数。	

44 引脚 QFN

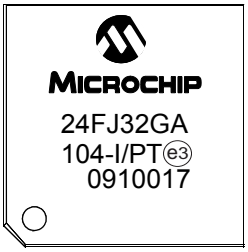


示例

44 引脚 TQFP



示例

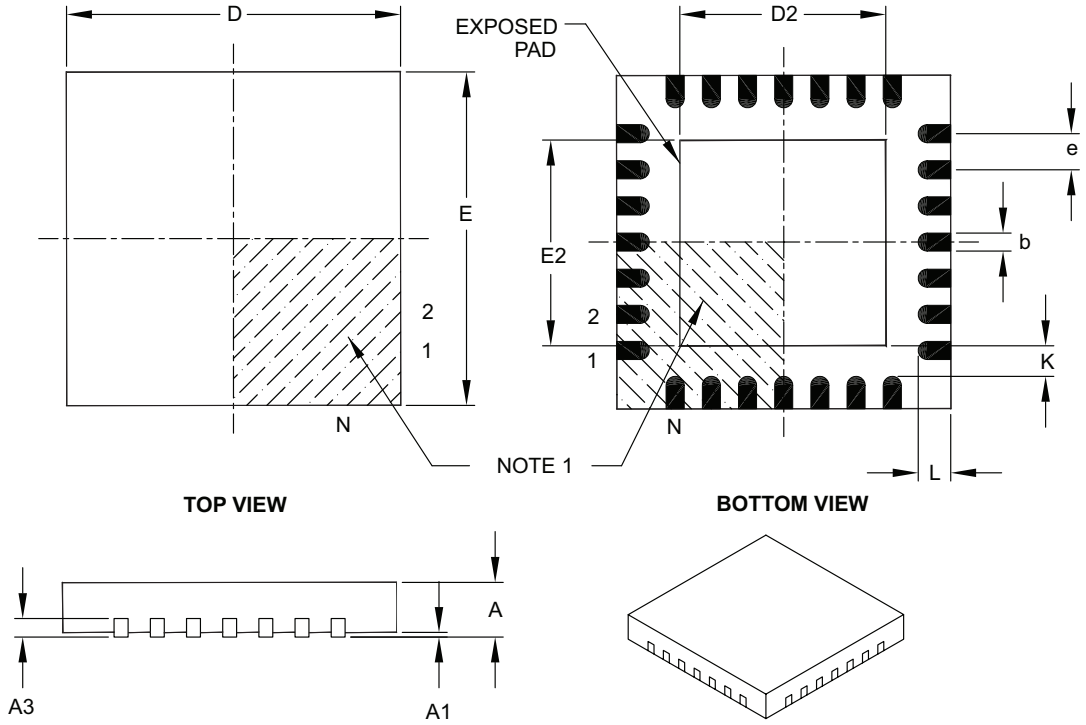


29.2 封装详细信息

以下部分将介绍各种封装的技术细节。

28引脚塑封正方扁平无脚封装（ML）——主体6x6 mm [QFN]，触点长度为0.55 mm

注： 最新封装图请至http://www.microchip.com/packaging查看Microchip封装规范。



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		0.65 BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Contact Thickness	A3		0.20 REF	
Overall Width	E		6.00 BSC	
Exposed Pad Width	E2	3.65	3.70	4.20
Overall Length	D		6.00 BSC	
Exposed Pad Length	D2	3.65	3.70	4.20
Contact Width	b	0.23	0.30	0.35
Contact Length	L	0.50	0.55	0.70
Contact-to-Exposed Pad	K	0.20	–	–

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.

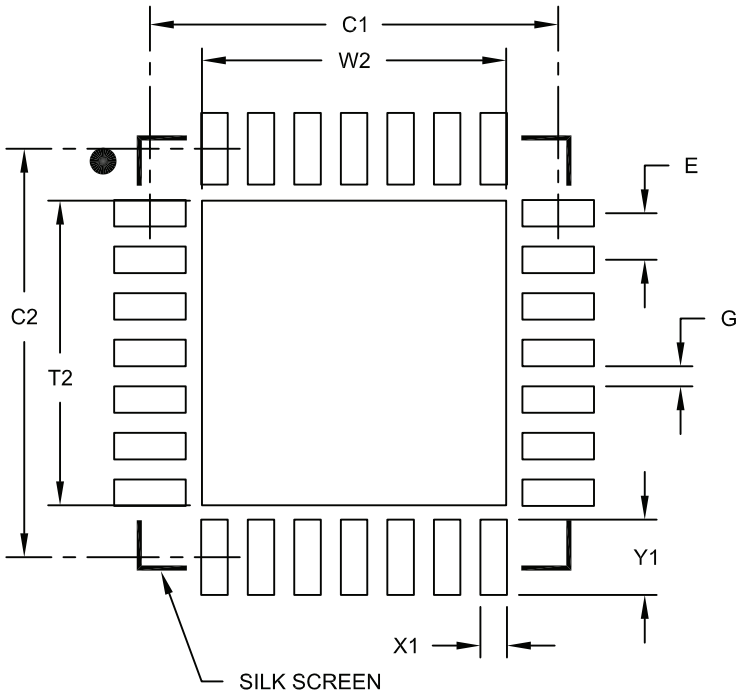
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-105B

28引脚塑封正方扁平无脚封装（ML）——主体6x6 mm [QFN]，触点长度为0.55 mm

注： 最新封装图请至http://www.microchip.com/packaging查看Microchip封装规范。



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.65 BSC	
Optional Center Pad Width	W2			4.25
Optional Center Pad Length	T2			4.25
Contact Pad Spacing	C1		5.70	
Contact Pad Spacing	C2		5.70	
Contact Pad Width (X28)	X1			0.37
Contact Pad Length (X28)	Y1			1.00
Distance Between Pads	G	0.20		

Notes:

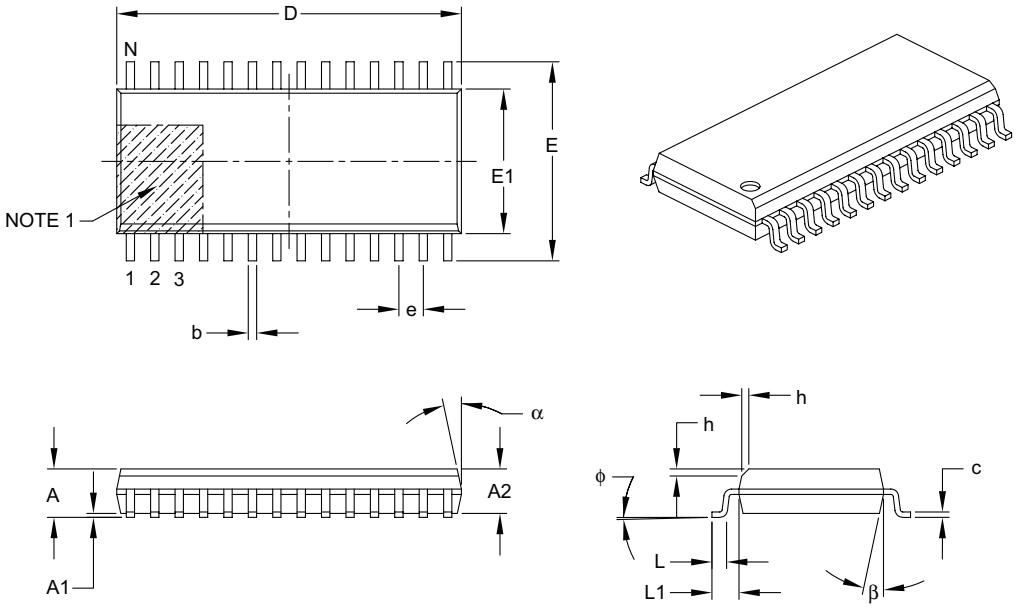
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2105A

28引脚塑封宽条小外形封装（SO）——主体7.50 mm [SOIC]

注： 最新封装图请至http://www.microchip.com/packaging查看Microchip封装规范。



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		1.27 BSC	
Overall Height	A	–	–	2.65
Molded Package Thickness	A2	2.05	–	–
Standoff §	A1	0.10	–	0.30
Overall Width	E		10.30 BSC	
Molded Package Width	E1		7.50 BSC	
Overall Length	D		17.90 BSC	
Chamfer (optional)	h	0.25	–	0.75
Foot Length	L	0.40	–	1.27
Footprint	L1		1.40 REF	
Foot Angle Top	φ	0°	–	8°
Lead Thickness	c	0.18	–	0.33
Lead Width	b	0.31	–	0.51
Mold Draft Angle Top	α	5°	–	15°
Mold Draft Angle Bottom	β	5°	–	15°

Notes:

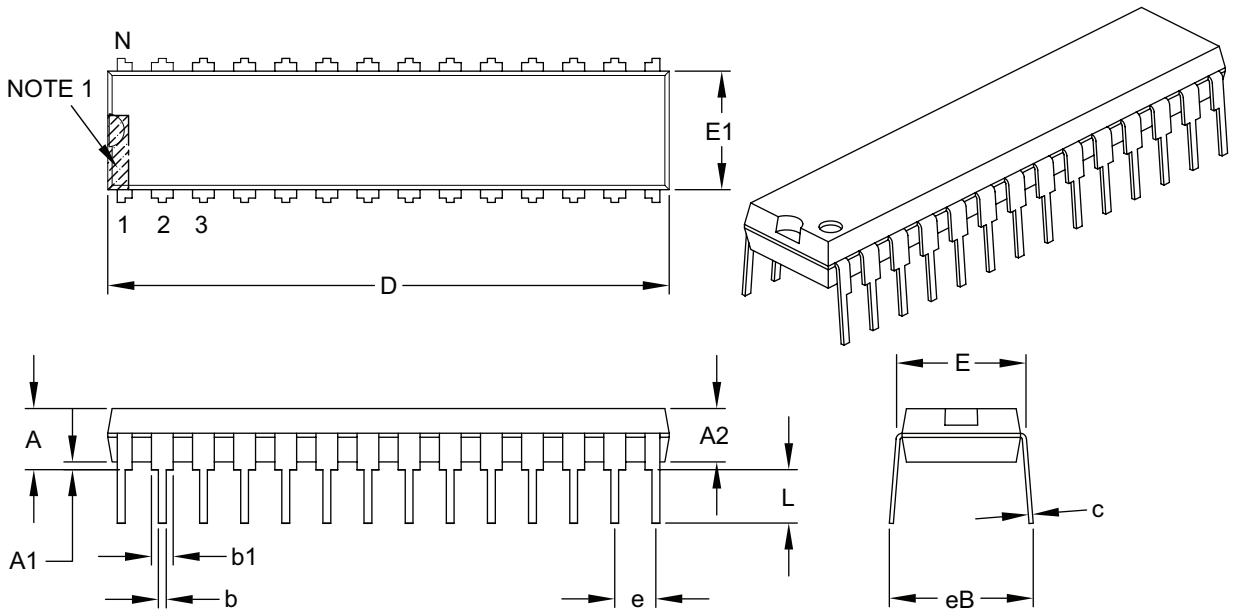
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.15 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-052B

28引脚窄型塑封双列直插式封装（SP）——主体300 mil [SPDIP]

注： 最新封装图请至http://www.microchip.com/packaging查看Microchip封装规范。



Units		INCHES		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		.100 BSC	
Top to Seating Plane	A	–	–	.200
Molded Package Thickness	A2	.120	.135	.150
Base to Seating Plane	A1	.015	–	–
Shoulder to Shoulder Width	E	.290	.310	.335
Molded Package Width	E1	.240	.285	.295
Overall Length	D	1.345	1.365	1.400
Tip to Seating Plane	L	.110	.130	.150
Lead Thickness	c	.008	.010	.015
Upper Lead Width	b1	.040	.050	.070
Lower Lead Width	b	.014	.018	.022
Overall Row Spacing §	eB	–	–	.430

Notes:

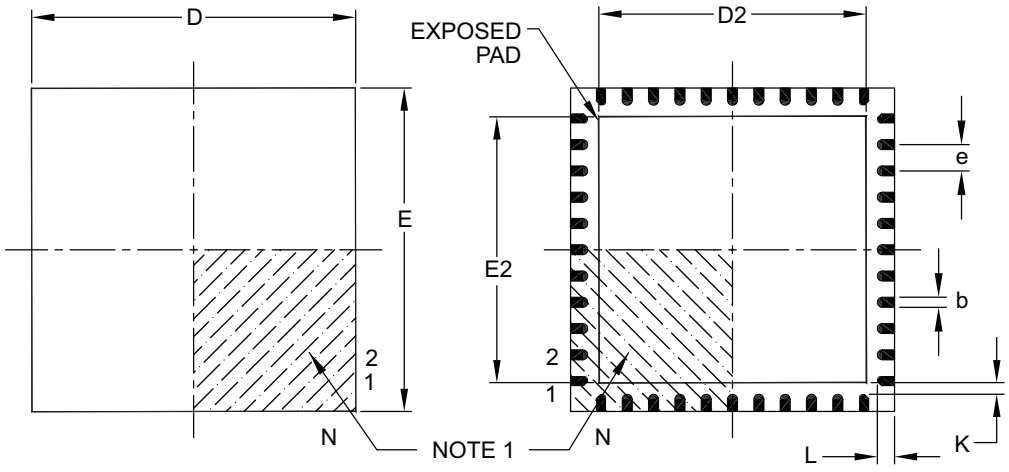
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

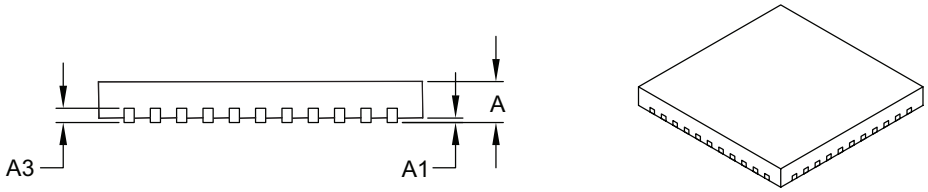
Microchip Technology Drawing C04-070B

44引脚塑封正方扁平无脚封装（ML）——主体8x8 mm [QFN]

注： 最新封装图请至http://www.microchip.com/packaging查看Microchip封装规范。



TOP VIEW **BOTTOM VIEW**



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		44	
Pitch	e		0.65 BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Contact Thickness	A3		0.20 REF	
Overall Width	E		8.00 BSC	
Exposed Pad Width	E2	6.30	6.45	6.80
Overall Length	D		8.00 BSC	
Exposed Pad Length	D2	6.30	6.45	6.80
Contact Width	b	0.25	0.30	0.38
Contact Length	L	0.30	0.40	0.50
Contact-to-Exposed Pad	K	0.20	–	–

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated.
3. Dimensioning and tolerancing per ASME Y14.5M.

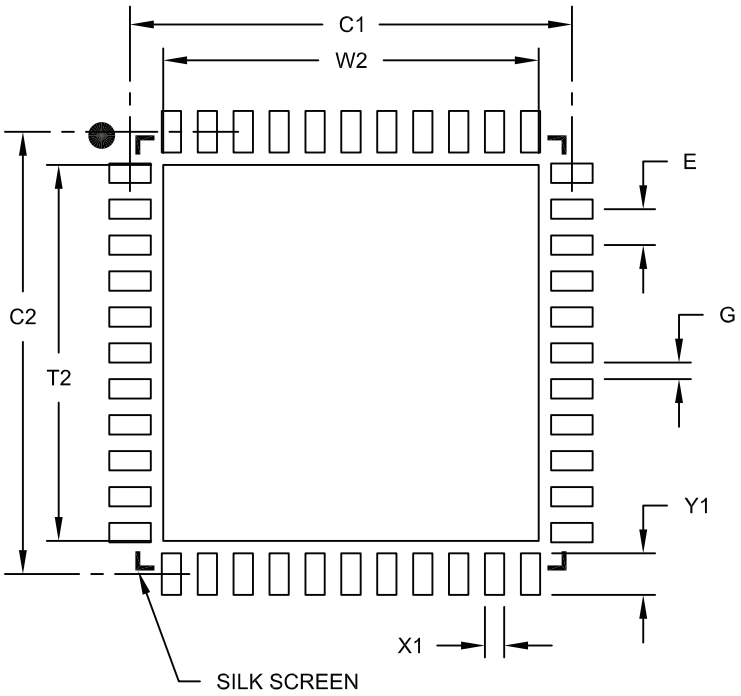
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-103B

44引脚塑封正方扁平无脚封装（ML）——主体8x8 mm [QFN]

注： 最新封装图请至http://www.microchip.com/packaging查看Microchip封装规范。



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.65 BSC	
Optional Center Pad Width	W2			6.80
Optional Center Pad Length	T2			6.80
Contact Pad Spacing	C1		8.00	
Contact Pad Spacing	C2		8.00	
Contact Pad Width (X44)	X1			0.35
Contact Pad Length (X44)	Y1			0.80
Distance Between Pads	G	0.25		

Notes:

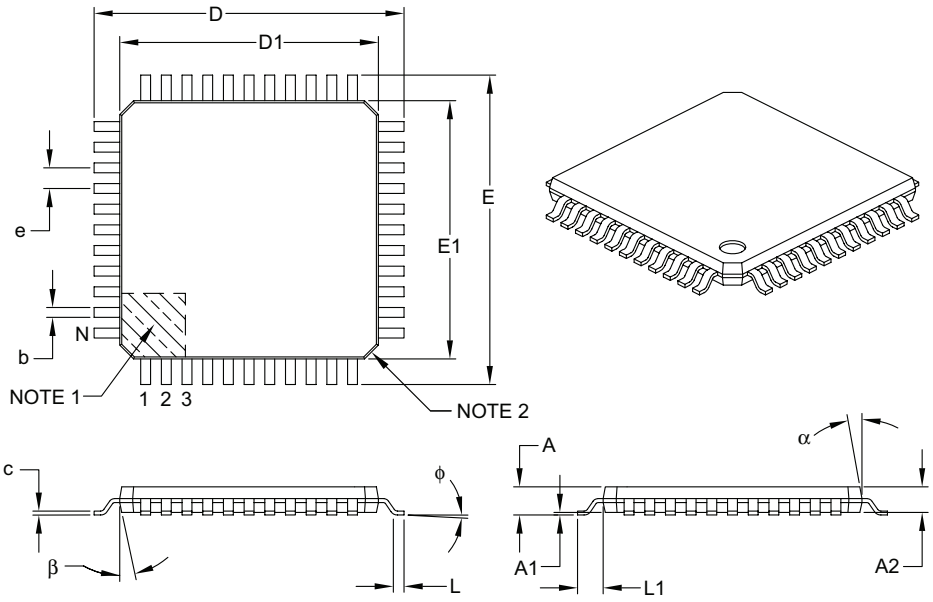
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2103A

44引脚塑封薄型正方扁平封装（PT）——主体10x10x1 mm，2.00 mm [TQFP]

注： 最新封装图请至http://www.microchip.com/packaging查看Microchip封装规范。



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Leads	N		44	
Lead Pitch	e		0.80 BSC	
Overall Height	A	–	–	1.20
Molded Package Thickness	A2	0.95	1.00	1.05
Standoff	A1	0.05	–	0.15
Foot Length	L	0.45	0.60	0.75
Footprint	L1		1.00 REF	
Foot Angle	φ	0°	3.5°	7°
Overall Width	E		12.00 BSC	
Overall Length	D		12.00 BSC	
Molded Package Width	E1		10.00 BSC	
Molded Package Length	D1		10.00 BSC	
Lead Thickness	c	0.09	–	0.20
Lead Width	b	0.30	0.37	0.45
Mold Draft Angle Top	α	11°	12°	13°
Mold Draft Angle Bottom	β	11°	12°	13°

Notes:

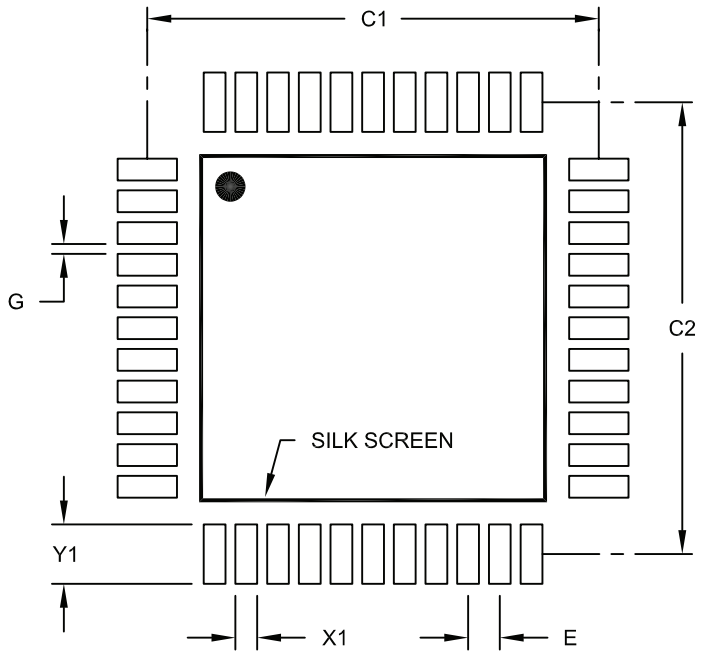
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-076B

44引脚塑封薄型正方扁平封装（PT）——主体10x10x1 mm，2.00 mm [TQFP]

注： 最新封装图请至http://www.microchip.com/packaging查看Microchip封装规范。



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.80 BSC	
Contact Pad Spacing	C1		11.40	
Contact Pad Spacing	C2		11.40	
Contact Pad Width (X44)	X1			0.55
Contact Pad Length (X44)	Y1			1.50
Distance Between Pads	G	0.25		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2076A

附录 A： 版本历史

版本 A （2009 年 8 月）

PIC24FJ64GA104 系列器件的原始数据手册。

版本 B （2009 年 10 月）

更正了**第 10.3 节 “输入电平变化通知”**关于 ICN 输入数量和下拉可用性的内容。

更新了**第 10.4.2 节 “可用的外设”**，从表 10-2 中删除了 Timer1 时钟输入。

更新了**第 28.1 节 “直流特性”**，如下：

- 在表29-4和29-5中增加了以0.5 MIPS工作时，对于 IDD 和 IIDLE 的规定。
- 更新了表29-4，修改了 1 MIP 和 4 MIPS 的最大 IDD 规定。
- 对 ΔIPD 电流（32 kHz，SOSCEL = 11）重新进行编号，从 DC62*n* 更改为 DC63*n*。

注：

索引

J

K

M

N

P

R

注：

MICROCHIP 网站

Microchip 网站（www.microchip.com）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。只要使用常用的因特网浏览器即可访问。网站提供以下信息：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及存档软件
- **一般技术支持**——常见问题 （FAQ）、技术支持请求、在线讨论组以及 Microchip 顾问计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、 Microchip 销售办事处、代理商以及工厂代表列表

变更通知客户服务

Microchip 的客户通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请登录 Microchip 网站 www.microchip.com，点击“变更通知客户 （Customer Change Notification）”服务后按照注册说明完成注册。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师 （FAE）
- 技术支持

客户应联系其代理商、代表或应用工程师 （FAE）寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过http://support.microchip.com获得网上技术支持。

读者反馈表

我们努力为您提供最佳文档，以确保您能够成功使用 Microchip 产品。如果您对文档的组织、条理性、主题及其他有助于提高文档质量的方面有任何意见或建议，请填写本反馈表并传真给我公司 TRC 经理，传真号码为 86-21-5407-5066。请填写以下信息，并从下面各方面提出您对本文档的意见。

致： TRC 经理　　总页数 _____

关于： 读者反馈

发自： 姓名 _____

公司 _____

地址 _____

国家 / 省份 / 城市 / 邮编 _____

电话: (_____) _____ - _____　　传真: (_____) _____ - _____

应用（选填）：

您希望收到回复吗？是 ____ 否____

器件：PIC24FJ64GB004 系列　　文献编号：DS39951B_CN

问题：

1. 本文档中哪些部分最有特色？

2. 本文档是否满足了您的软硬件开发要求？如何满足的？

3. 您认为本文档的组织结构便于理解吗？如果不便于理解，那么问题何在？

4. 您认为本文档应该添加哪些内容以改善其结构和主题？

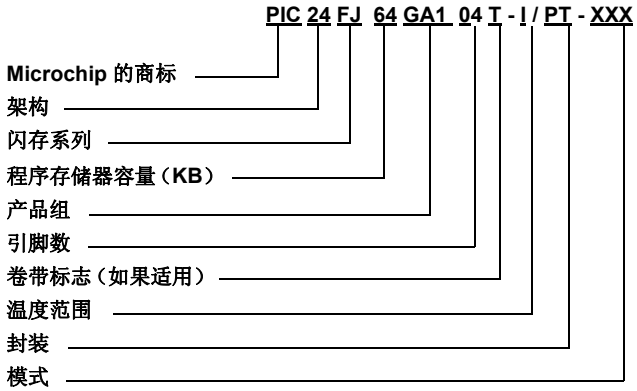
5. 您认为本文档中可以删减哪些内容，而又不会影响整体使用效果？

6. 本文档中是否存在错误或误导信息？如果存在，请指出是什么信息及其具体页数。

7. 您认为本文档还有哪些方面有待改进？

产品标识体系

欲订货或获取价格、交货等信息，请与我公司生产厂或各销售办事处联系。



架构	24	= 不带 DSP 的 16 位改进型哈佛架构
闪存系列	FJ	= 闪存程序存储器
产品组	GA1	= 通用单片机
引脚数	02 04	= 28 引脚 = 44 引脚
温度范围	I	= -40°C 至 +85°C （工业级）
封装	ML PT SO SP	= 28 引脚（6x6 mm）或 44 引脚（8x8 mm）QFN（正方扁平封装） = 44 引脚（10x10x1 mm）TQFP（薄型正方扁平封装） = 28 引脚 （7.50 mm 宽条） SOIC （小外形封装） = 28 引脚（300 mil）SPDIP（窄型塑封双列直插式封装）
模式	三位 QTP、 SQTP、代码或特殊要求 （空白为其他情况） ES	= 工程样片

示例：

a) PIC24FJ64GA104-I/PT：具有 USB On-The-Go 的 PIC24F 器件、64 KB 程序存储器、44 引脚、工业级温度和 TQFP 封装。

b) PIC24FJ32GA102-I/ML：具有 USB On-The-Go 的 PIC24F 器件、32 KB 程序存储器、28 引脚、工业级温度和 QFN 封装。

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://support.microchip.com
网址：www.microchip.com

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

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

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

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

Fax: 216-447-0643

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

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

科科莫 Kokomo
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

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

圣克拉拉 Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

加拿大多伦多 Toronto
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

亚太总部 Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200

Fax: 852-2401-3431

中国 – 北京
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

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

中国 – 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

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

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

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

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

中国 – 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

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

中国 – 厦门
Tel: 86-592-238-8138

Fax: 86-592-238-8130

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

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

台湾地区 – 高雄
Tel: 886-7-536-4818
Fax: 886-7-536-4803

台湾地区 – 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

台湾地区 – 新竹
Tel: 886-3-6578-300
Fax: 886-3-6578-370

亚太地区

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

印度 India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4080

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

印度 India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

日本 Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

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

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

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

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

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

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

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

欧洲

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

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

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

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

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

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

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

英国 UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

03/26/09