PIC18F87J90 系列
数据手册

采用纳瓦技术、
带 LCD 驱动器的
64/80 引脚高性能单片机

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。

ISBN：978-1-61341-577-1

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-16949:2009 认证。 Microchip 的 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品严格遵守公司的质量体系流程。此外， Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

PIC18F87J90 系列

采用纳瓦技术、带 LCD 驱动器的 64/80 引脚高性能单片机

LCD 驱动器和键盘接口特性：

- 直接驱动 LCD 面板能力：
  - 处于休眠模式时仍可驱动 LCD 面板
- 最多 48 个段和 192 像素，可由软件选择
- 可编程 LCD 定时模块：
  - 多个 LCD 定时源
  - 最多 4 个公共时钟：静态、1/2、1/3 或 1/4 多路复用
  - 静态、 1/2 或 1/3 偏置配置
- 用于对比度控制的片上 LCD 升压稳压器
- 用于容性触摸传感的充电时间测量单元 （Charge Time Measurement Unit， CTMU）
- 用于阻性触摸传感的 ADC

低功耗特性：

- 功耗管理模式：
  - 运行：CPU 工作，外设工作
  - 空闲：CPU 不工作，外设工作
  - 休眠：CPU 不工作，外设不工作
- 双速振荡器启动

灵活的振荡器结构：

- 两种晶振模式，频率为 4-25 MHz
- 两种外部时钟模式，频率最高为 48 MHz
- 4 倍频锁相环 （Phase Lock Loop， PLL）
- 带 PLL 的内部振荡器模块：
  - 8个可由用户选择的频率：从31.25 kHz到8 MHz
- 辅助振荡器使用 Timer1 （工作频率为 32 kHz）
- 故障保护时钟监视器 （Fail-Safe Clock Monitor，FSCM）：
  - 当外设时钟发生故障时可使器件安全关闭

外设特点：

- 高灌 / 拉电流：25 mA/25 mA（PORTB 和 PORTC）
- 最多 4 个外部中断
- 4 个 8 位 /16 位定时器 / 计数器模块
- 两个捕捉 / 比较 /PWM（Capture/Compare/PWM，CCP）模块
- 主同步串行口 （Master Synchronous Serial Port，MSSP）模块，支持两种工作模式：
  - 3 线 /4 线 SPI （支持所有 4 种 SPI 模式）
  - I^2C™ 主 / 从模式
- 一个可寻址的 USART 模块
- 一个增强型可寻址的 USART 模块：
  - 支持 LIN/J2602
  - 接收到启动位和间隔字符时自动唤醒
  - 自动波特率检测 （Auto-Baud Detect， ABD）
- 最多 12 路通道的 10 位 A/D 转换器：
  - 自动采集
  - 可在休眠模式下进行转换
- 两个模拟比较器
- 用于比较器的可编程参考电压
- 硬件实时时钟和日历（Real-Time Clock and Calendar，RTCC），提供时钟、日历和闹钟功能
- 充电时间测量单元 （CTMU）：
  - 电容测量
  - 时间测量，分辨率典型值为 1 ns

单片机特性：

- 闪存程序存储器可承受 10,000 次擦写 （典型值）
- 闪存保存时间：最少 20 年
- 可在软件控制下自编程
- 用于数据 EEPROM 仿真器的闪存程序存储器的字写能力

器件	闪存程序存储器（字节）	SRAM 数据存储器（字节）	I/O	LCD（像素）	8/16 位定时器	CCP	MSSP		EUSART AUSART	10 位 A/D（通道）	比较器	BOR/LVD	RTCC	CTMU
							SPI	主 I^2C™						
PIC18F66J90	64K	3,923	51	132	1/3	2	有	有	1/1	12	2	有	有	有
PIC18F67J90	128K	3,923	51	132	1/3	2	有	有	1/1	12	2	有	有	有
PIC18F86J90	64K	3,923	67	192	1/3	2	有	有	1/1	12	2	有	有	有
PIC18F87J90	128K	3,923	67	192	1/3	2	有	有	1/1	12	2	有	有	有

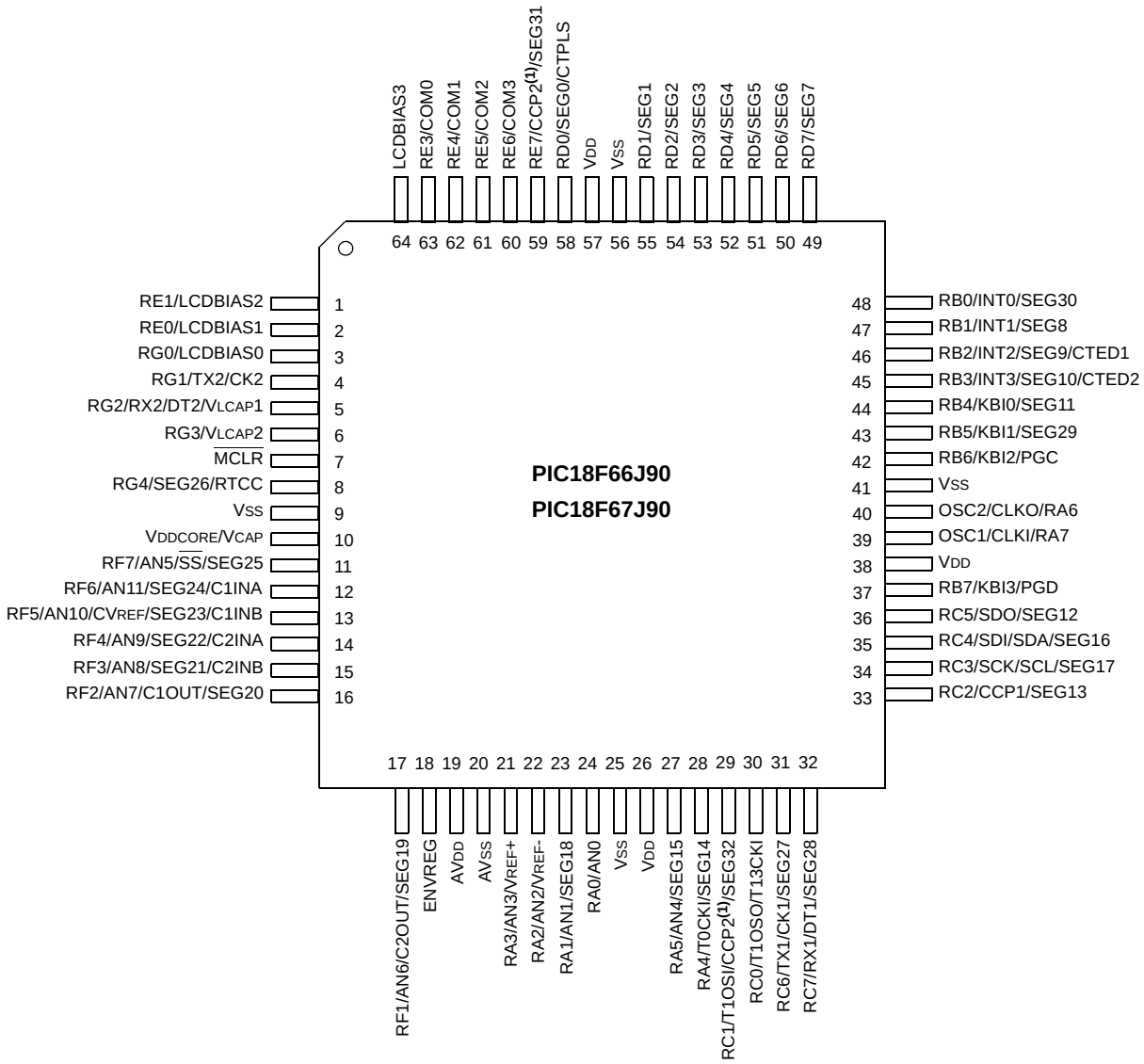
单片机特性（续）：

- 中断优先级
- 8 x 8 单周期硬件乘法器
- 扩展型看门狗定时器（Watchdog Timer，WDT）：
  - 可编程周期从 4 ms 到 131s
- 通过两个引脚进行在线串行编程（In-Circuit Serial Programming™， ICSP™）
- 通过两个引脚进行在线调试
- 工作电压范围：2.0V 至 3.6V
- 可承受 5.5V 输入电压（仅对于数字引脚）
- 用于串行通信以及 CCP 引脚的可选择漏极开路配置可驱动最高 5V 输出
- 片上 2.5V 稳压器

引脚图 ——PIC18F6XJ90

64 引脚 TQFP

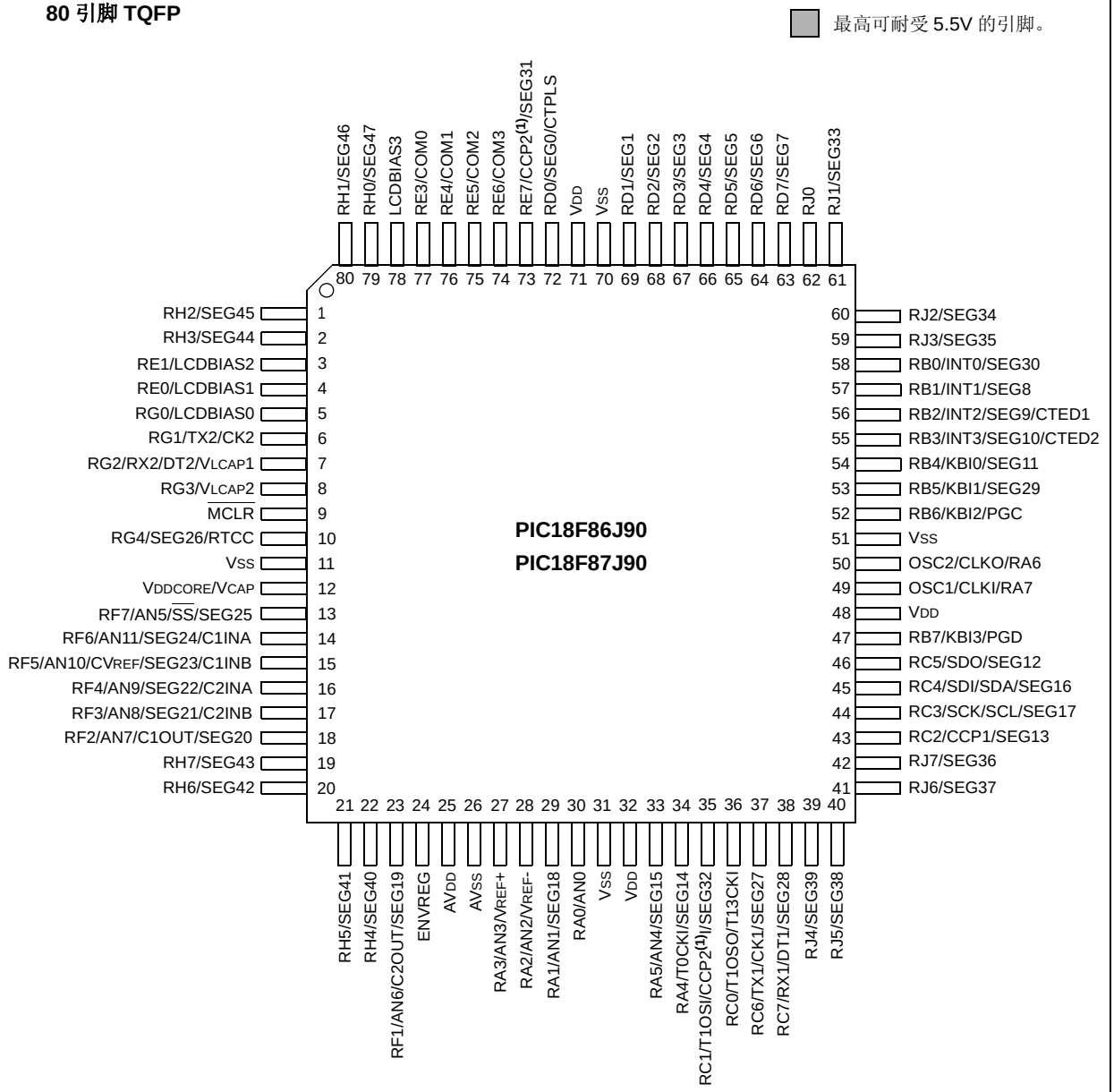
最高可耐受 5.5V 的引脚。



注 1： CCP2 引脚位置取决于 CCP2MX 位设置。

引脚图 ——PIC18F8XJ90

80 引脚 TQFP



注 1： CCP2 引脚位置取决于 CCP2MX 位设置。

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

本文档涉及以下器件的具体信息：

- PIC18F66J90
- PIC18F67J90
- PIC18F86J90
- PIC18F87J90

本系列继承了所有 PIC18 单片机的传统优点，即具有出色的计算性能、丰富的功能集，除此之外，还带有一个多功能片上 LCD 驱动器，同时价位极具竞争力。这些特点使得 PIC18F87J90 系列成为许多高性能，尤其是那些价格作为首要考虑因素的应用的理想选择。

1.1 内核特性

1.1.1 纳瓦技术

PIC18F87J90系列的所有器件具有一系列能在工作时显著降低功耗的功能。主要包含以下几项：

- **备用运行模式：**通过将 Timer1 或内部 RC 振荡器作为单片机时钟源，可使代码执行时的功耗降低大约 90%。
- **多种空闲模式：**控制器还可在其 CPU 内核禁止而外设仍然工作的情况下工作。处于这些状态时，功耗能降得更低，只有正常工作时的 4%。
- **动态模式切换：**在器件工作期间可由用户代码调用功耗管理模式，允许用户将节能的理念融入到他们的应用软件设计中。

1.1.2 振荡器选项和特性

PIC18F87J90系列的所有器件可提供6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，提供四分频时钟输出选项。
- 一个锁相环 （PLL）倍频器，可在外部振荡器模式下使用，可使时钟速度最高达到 40 MHz。 PLL 还可以与内部振荡器一起使用。
- 一个内部振荡器模块，它提供一个 8 MHz 的时钟源（精度为 ±2%）和一个 INTRC 时钟源（振荡频率大约为 31 kHz，温度和 V_{DD} 变化时频率保持稳定），以及用户可选择的 6 种时钟频率 （从 125 kHz 到 4 MHz），因此共有 8 种时钟频率可供选择。此选项可以空出两个振荡器引脚作为额外的通用 I/O 引脚。

内部振荡器模块还提供了一个稳定的参考源，从而帮助器件增加了以下功能以使器件更可靠地工作：

- **故障保护时钟监视器：**该功能持续监视主时钟源，将其与内部振荡器提供的参考信号作比较。如果时钟发生了故障，控制器会将时钟源切换到内部振荡器，使器件可继续低速工作或安全地关闭应用。
- **双速启动：**该功能允许在上电复位或从休眠模式唤醒时将内部振荡器用作时钟源，直到主时钟源可用为止。

1.1.3 存储器选项

PIC18F87J90 系列为应用程序代码提供了充足的空间，代码空间从 64 KB 到 128 KB。程序存储器的闪存单元经评测最多可承受 10,000 次擦写。在不刷新的情况下，数据保存时间保守地估计在 20 年以上。

闪存程序存储器是可读写的。在正常工作期间，PIC18F87J90 系列还为动态应用数据提供了充足的空间，最大 3,923 字节的数据 RAM。

1.1.4 扩展指令集

PIC18F87J90 系列在 PIC18 指令集的基础上实现了可选择的扩展，添加了 8 条新指令和一个变址寻址模式。此扩展可以使用一个器件配置选项使能，它是为优化原来采用高级语言（如 C）开发的可重入应用程序代码而专门设计的。

1.1.5 移植方便

无论存储器容量如何，所有器件都共享相同的丰富外设，使应用的扩展和升级工作变得轻松而容易。

整个系列的引脚排列设计一致也有助于向下一代引脚更多的器件移植。在64引脚器件间、80引脚器件间移植，甚至是从64引脚器件向80引脚器件的移植都是可以的。

PIC18F87J90系列的引脚还与其他PIC18系列器件（如PIC18F8720 和 PIC18F8722、PIC18F85J11 以及带有 LCD 驱动器的 PIC18F8490 和 PIC18F85J90 系列单片机）的引脚兼容。这为不同应用的发展拓展了空间，使开发人员能在保留相同功能集的同时可在 Microchip PIC18 系列中选择不同价位的器件。

1.2 LCD 驱动器

片上LCD驱动器包括许多功能，使得在低功耗应用中显示的集成更容易。这些包括一个带有电荷泵的集成稳压器，允许用软件控制对比度并且可在高于器件 VDD 下进行显示操作。

1.3 其他特性

- **通信：**PIC18F87J90系列具有一系列串行通信外设，包括一个可寻址的 USART、一个独立的支持 LIN/J2602 规范 1.2 的增强型 USART 以及一个支持 SPI 和 I^2C™（主和从）工作模式的主 SSP 模块。
- **CCP 模块：**本系列的所有器件都具有两个捕捉 / 比较 /PWM（CCP）模块。在同一时间，最多可以使用 4 种不同时基来执行几项不同的操作。
- **10 位 A/D 转换器：**该模块具备可编程采集时间，从而不必在选择通道和启动转换之间等待一个采样周期，因而减少了代码开销。
- **充电时间测量单元（CTMU）：**CTMU 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。

  CTMU 可与其他片上模拟模块一起，用于精确测量时间、电容、电容的相对变化，或生成独立于系统时钟的输出脉冲。
- **扩展型看门狗定时器（WDT）：**该增强型版本融入了一个 16 位预分频器，可以提供在不同工作电压和温度下保持稳定的扩展超时范围。超时周期请参见**第 28.0 节 “ 电气特性 ”**。
- **实时时钟和日历（RTCC）模块：**RTCC 模块是为需要长时间维持精确时间的应用设计的，无需或很少需要 CPU 干预。

  该模块是百年时钟和日历，能自动检测闰年。时钟范围从 2000 年 1 月 1 日 00:00:00（午夜）到 2099 年 12 月 31 日 23:59:59。

1.4 系列中各产品的详细说明

PIC18F87J90 系列器件提供 64 引脚和 80 引脚封装形式。图 1-1 和图 1-2 分别为这两类器件的框图。

这些器件在以下四个方面存在差异：

1. 闪存程序存储器 （两种容量， PIC18FX6J90 器件为 64 KB， PIC18FX7J90 器件为 128 KB）。
2. 数据 RAM （PIC18FX6J90 和 PIC18FX7J90 器件均为 3,923 字节 RAM）。
3. I/O 端口数 （PIC18F6XJ90 器件有 7 个双向端口， PIC18F8XJ90 器件有 9 个双向端口）。
4. LCD 像素：64 引脚器件可驱动 132 像素（33 个 SEG x 4 个 COM），而 80 引脚器件可驱动 192 像素 （48 个 SEG x 4 个 COM）。

本系列器件的所有其他功能都是相同的。表1-1 和表1-2 汇总了这些功能。

表 1-3 和表 1-4 列出了所有器件的引脚排列。

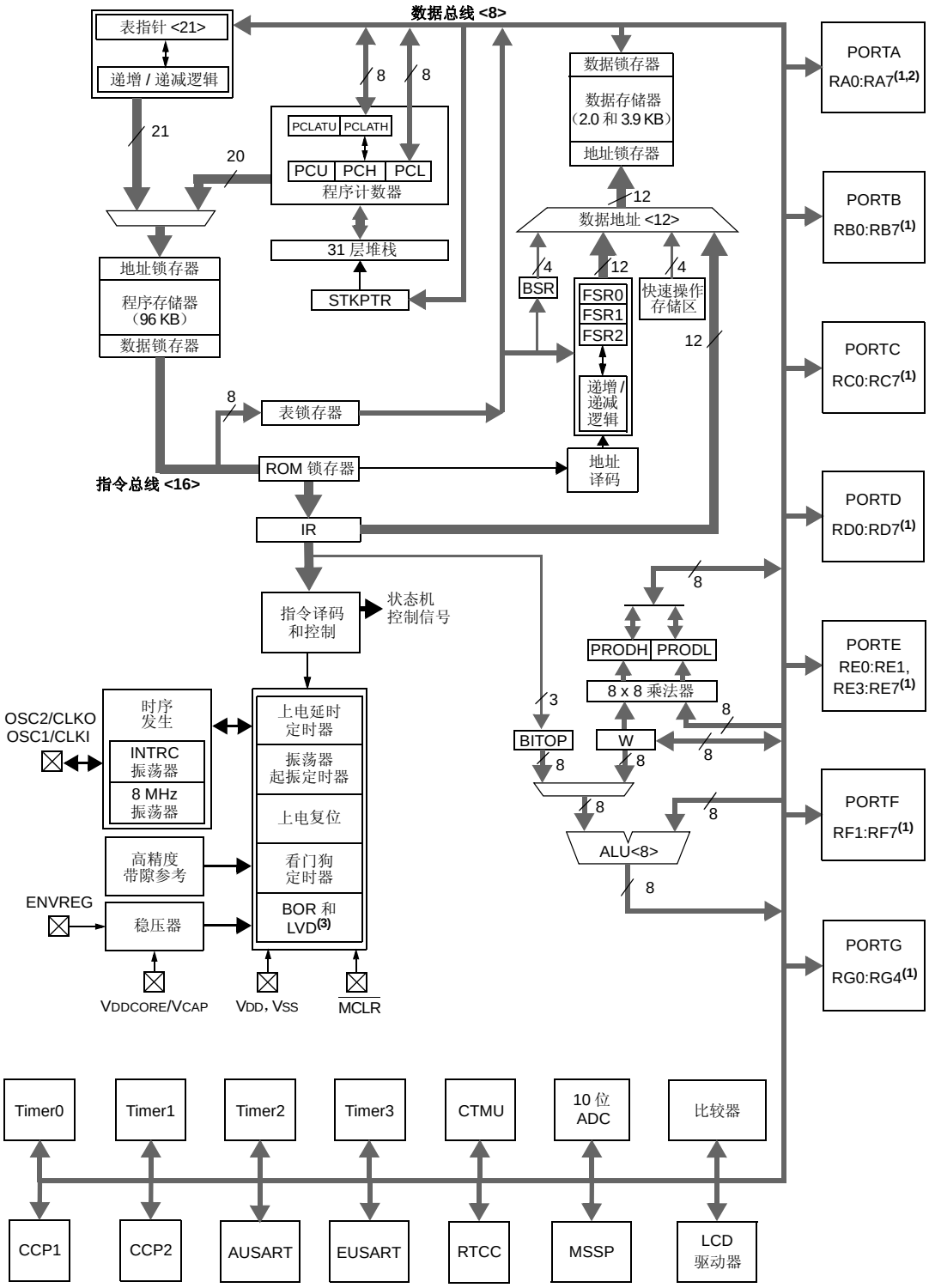
表 1-1： PIC18F6XJ90 器件特性 （64 引脚器件）

特性	PIC18F66J90	PIC18F67J90
工作频率	DC – 48 MHz	
程序存储器 （字节）	64K	128K
程序存储器 （指令）	32,768	65,536
数据存储器 （字节）	3,923	3,923
中断源	29	
I/O 端口	端口 A, B, C, D, E, F, G	
LCD 驱动器 （可以驱动的像素数量）	132 （33 个 SEG x 4 个 COM）	
定时器	4	
比较器	2	
CTMU	有	
RTCC	有	
捕捉 / 比较 /PWM 模块	2	
串行通信	MSSP、可寻址的 USART 和增强型 USART	
10 位模数转换模块	12 路输入通道	
复位 （和延时）	POR、 BOR、 RESET 指令、堆栈满、堆栈下溢、 $\overline{\text{MCLR}}$ 和 WDT （PWRT 和 OST）	
指令集	75 条指令，使能扩展指令集后总共为 83 条指令	
封装	64 引脚 TQFP	

表 1-2： PIC18F8XJ90 器件特性 （80 引脚器件）

特性	PIC18F86J90	PIC18F87J90
工作频率	DC – 48 MHz	
程序存储器 （字节）	64K	128K
程序存储器 （指令）	32,768	65,536
数据存储器 （字节）	3,923	3,923
中断源	29	
I/O 端口	端口 A, B, C, D, E, F, G, H, J	
LCD 驱动器 （可以驱动的像素数量）	192 （48 个 SEG x 4 个 COM）	
定时器	4	
比较器	2	
CTMU	有	
RTCC	有	
捕捉 / 比较 /PWM 模块	2	
串行通信	MSSP、可寻址的 USART 和增强型 USART	
10 位模数转换模块	12 路输入通道	
复位 （和延时）	POR、 BOR、 RESET 指令、堆栈满、堆栈下溢、 $\overline{\text{MCLR}}$ 和 WDT （PWRT 和 OST）	
指令集	75 条指令，使能扩展指令集后总共为 83 条指令	
封装	80 引脚 TQFP	

图 1-1：　　PIC18F6XJ90 （64 引脚）框图

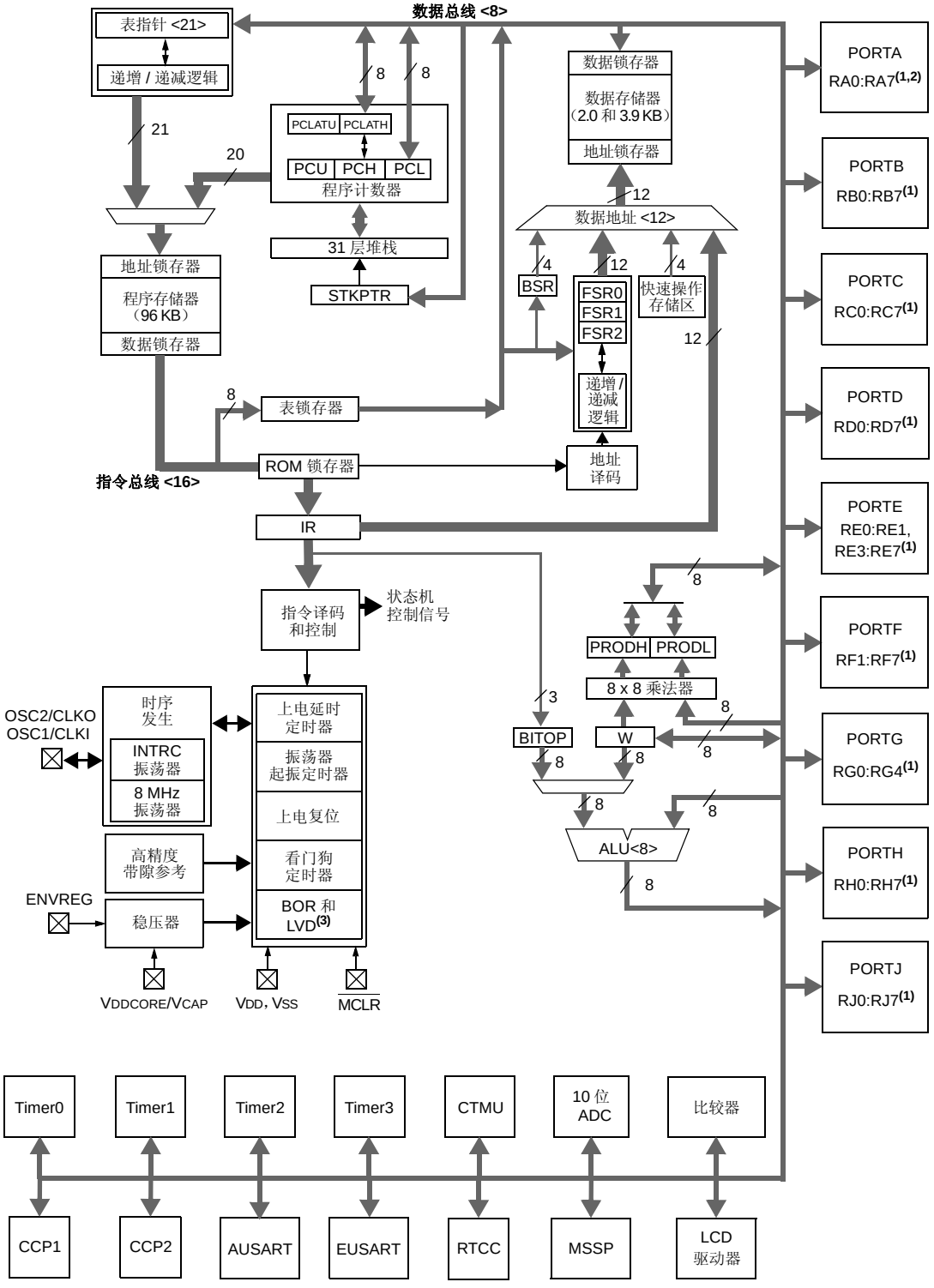


注　**1：** 关于 I/O 端口引脚的说明，请参见表 1-3。

2： RA6 和 RA7 在某些振荡器模式中只能用作数字 I/O。更多信息，请参见**第 3.0 节 “ 振荡器配置 ”**。

3： 欠压复位和低压检测功能在使能片上稳压器时提供。

图 1-2：　　PIC18F8XJ90 （80 引脚）框图



注　　1： 关于 I/O 端口引脚的说明，请参见表 1-3。

2： RA6 和 RA7 在某些振荡器模式中只能用作数字 I/O。更多信息，请参见**第 3.0 节 “ 振荡器配置 ”**。

3： 欠压复位和低压检测功能在使能片上稳压器时提供。

表 1-3： PIC18F6XJ90 引脚说明

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
$\overline{MCLR}$	7	I	ST	主复位 （输入）或编程电压 （输入）。此引脚为低电平有效的器件复位输入端。
OSC1/CLKI/RA7	39			晶振或外部时钟输入。
OSC1		I	CMOS	晶振输入。
CLKI		I	CMOS	外部时钟源输入。总是与 OSC1 引脚功能相关联。（见相关的 OSC1/CLKI 和 OSC2/CLKO 引脚信息。）
RA7		I/O	TTL	通用 I/O 引脚。
OSC2/CLKO/RA6	40			晶振或时钟输出。
OSC2		O	—	晶振输出。在晶振模式下，该引脚与晶振或谐振器相连。
CLKO		O	—	在EC模式下，OSC2引脚输出CLKO信号，其频率是OSC1引脚上信号频率的 1/4，该频率等于指令周期的倒数。
RA6		I/O	TTL	通用 I/O 引脚。
				PORTA 是双向 I/O 端口。
RA0/AN0	24			
RA0		I/O	TTL	数字 I/O。
AN0		I	Analog	模拟输入 0。
RA1/AN1/SEG18	23			
RA1		I/O	TTL	数字 I/O。
AN1		I	Analog	模拟输入 1。
SEG18		O	Analog	LCD 的 SEG18 输出。
RA2/AN2/V_{REF-}	22			
RA2		I/O	TTL	数字 I/O。
AN2		I	Analog	模拟输入 2。
V_{REF-}		I	Analog	A/D 参考电压 （低电压）输入。
RA3/AN3/V_{REF+}	21			
RA3		I/O	TTL	数字 I/O。
AN3		I	Analog	模拟输入 3。
V_{REF+}		I	Analog	A/D 参考电压 （高电压）输入。
RA4/T0CKI/SEG14	28			
RA4		I/O	ST	数字 I/O。
T0CKI		I	ST	Timer0 外部时钟输入。
SEG14		O	Analog	LCD 的 SEG14 输出。
RA5/AN4/SEG15	27			
RA5		I/O	TTL	数字 I/O。
AN4		I	Analog	模拟输入 4。
SEG15		O	Analog	LCD 的 SEG15 输出。
RA6				见 OSC2/CLKO/RA6 引脚信息。
RA7				见 OSC1/CLKI/RA7 引脚信息。

图注： TTL = TTL 兼容输入
ST = CMOS 电平的施密特触发器输入
I = 输入
P = 电源
I^2C™ = I^2C/SMBus
CMOS = CMOS 兼容输入或输出
Analog = 模拟输入
O = 输出
OD = 漏极开路 （没有 P 型二极管连接到 V_{DD}）

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。
2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-3： PIC18F6XJ90 引脚说明 （续）

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
				PORTB 是双向 I/O 端口。PORTB 的所有输入端都可软件编程为内部弱上拉。
RB0/INT0/SEG30	48			
RB0		I/O	TTL	数字 I/O。
INT0		I	ST	外部中断 0。
SEG30		O	Analog	LCD 的 SEG30 输出。
RB1/INT1/SEG8	47			
RB1		I/O	TTL	数字 I/O。
INT1		I	ST	外部中断 1。
SEG8		O	Analog	LCD 的 SEG8 输出。
RB2/INT2/SEG9/CTED1	46			
RB2		I/O	TTL	数字 I/O。
INT2		I	ST	外部中断 2。
CTED1		I	ST	CTMU 边沿 1 输入。
SEG9		O	Analog	LCD 的 SEG9 输出。
RB3/INT3/SEG10/CTED2	45			
RB3		I/O	TTL	数字 I/O。
INT3		I	ST	外部中断 3。
SEG10		O	Analog	LCD 的 SEG10 输出。
CTED2		I	ST	CTMU 边沿 2 输入。
RB4/KBI0/SEG11	44			
RB4		I/O	TTL	数字 I/O。
KBI0		I	TTL	电平变化中断引脚。
SEG11		O	Analog	LCD 的 SEG11 输出。
RB5/KBI1/SEG29	43			
RB5		I/O	TTL	数字 I/O。
KBI1		I	TTL	电平变化中断引脚。
SEG29		O	Analog	LCD 的 SEG29 输出。
RB6/KBI2/PGC	42			
RB6		I/O	TTL	数字 I/O。
KBI2		I	TTL	电平变化中断引脚。
PGC		I/O	ST	在线调试器和 ICSP™ 编程时钟引脚。
RB7/KBI3/PGD	37			
RB7		I/O	TTL	数字 I/O。
KBI3		I	TTL	电平变化中断引脚。
PGD		I/O	ST	在线调试器和 ICSP 编程数据引脚。

图注：
TTL = TTL 兼容输入
ST = CMOS 电平的施密特触发器输入
I = 输入
P = 电源
I^2C™ = I^2C/SMBus
CMOS = CMOS 兼容输入或输出
Analog = 模拟输入
O = 输出
OD = 漏极开路 （没有 P 型二极管连接到 V_{DD}）

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。
2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-3： PIC18F6XJ90 引脚说明 （续）

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
				PORTC 是双向 I/O 端口。
RC0/T1OSO/T13CKI	30			
RC0		I/O	ST	数字 I/O。
T1OSO		O	—	Timer1 振荡器输出。
T13CKI		I	ST	Timer1/Timer3 外部时钟输入。
RC1/T1OSI/CCP2/SEG32	29			
RC1		I/O	ST	数字 I/O。
T1OSI		I	CMOS	Timer1 振荡器输入。
CCP2(1)		I/O	ST	捕捉 2 输入 / 比较 2 输出 /PWM2 输出。
SEG32		O	Analog	LCD 的 SEG32 输出。
RC2/CCP1/SEG13	33			
RC2		I/O	ST	数字 I/O。
CCP1		I/O	ST	捕捉 1 输入 / 比较 1 输出 /PWM1 输出。
SEG13		O	Analog	LCD 的 SEG13 输出。
RC3/SCK/SCL/SEG17	34			
RC3		I/O	ST	数字 I/O。
SCK		I/O	ST	SPI 模式的同步串行时钟输入 / 输出。
SCL		I/O	I^2C	I^2C™ 模式的同步串行时钟输入 / 输出。
SEG17		O	Analog	LCD 的 SEG17 输出。
RC4/SDI/SDA/SEG16	35			
RC4		I/O	ST	数字 I/O。
SDI		I	ST	SPI 数据输入。
SDA		I/O	I^2C	I^2C 数据 I/O。
SEG16		O	Analog	LCD 的 SEG16 输出。
RC5/SDO/SEG12	36			
RC5		I/O	ST	数字 I/O。
SDO		O	—	SPI 数据输出。
SEG12		O	Analog	LCD 的 SEG12 输出。
RC6/TX1/CK1/SEG27	31			
RC6		I/O	ST	数字 I/O。
TX1		O	—	EUSART 异步发送。
CK1		I/O	ST	EUSART 同步时钟 （见相关的 RX1/DT1 引脚信息）。
SEG27		O	Analog	LCD 的 SEG27 输出。
RC7/RX1/DT1/SEG28	32			
RC7		I/O	ST	数字 I/O。
RX1		I	ST	EUSART 异步接收。
DT1		I/O	ST	EUSART 同步数据 （见相关的 TX1/CK1 引脚信息）。
SEG28		O	Analog	LCD 的 SEG28 输出。

图注： TTL = TTL 兼容输入
ST = CMOS 电平的施密特触发器输入
I = 输入
P = 电源
I^2C™ = I^2C/SMBus
CMOS = CMOS 兼容输入或输出
Analog = 模拟输入
O = 输出
OD = 漏极开路 （没有 P 型二极管连接到 V_{DD}）

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。
2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-3： PIC18F6XJ90 引脚说明 （续）

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
				PORTD 是双向 I/O 端口。
RD0/SEG0/CTPLS	58			
RD0		I/O	ST	数字 I/O。
SEG0		O	Analog	LCD 的 SEG0 输出。
CTPLS		O	—	CTMU 脉冲发生器输出。
RD1/SEG1	55			
RD1		I/O	ST	数字 I/O。
SEG1		O	Analog	LCD 的 SEG1 输出。
RD2/SEG2	54			
RD2		I/O	ST	数字 I/O。
SEG2		O	Analog	LCD 的 SEG2 输出。
RD3/SEG3	53			
RD3		I/O	ST	数字 I/O。
SEG3		O	Analog	LCD 的 SEG3 输出。
RD4/SEG4	52			
RD4		I/O	ST	数字 I/O。
SEG4		O	Analog	LCD 的 SEG4 输出。
RD5/SEG5	51			
RD5		I/O	ST	数字 I/O。
SEG5		O	Analog	LCD 的 SEG5 输出。
RD6/SEG6	50			
RD6		I/O	ST	数字 I/O。
SEG6		O	Analog	LCD 的 SEG6 输出。
RD7/SEG7	49			
RD7		I/O	ST	数字 I/O。
SEG7		O	Analog	LCD 的 SEG7 输出。

图注： TTL = TTL 兼容输入
ST = CMOS 电平的施密特触发器输入
I = 输入
P = 电源
I^2C™ = I^2C/SMBus
CMOS = CMOS 兼容输入或输出
Analog = 模拟输入
O = 输出
OD = 漏极开路 （没有 P 型二极管连接到 V_{DD}）

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。
2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-3： PIC18F6XJ90 引脚说明 （续）

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
				PORTE 是双向 I/O 端口。
RE0/LCDBIAS1	2			
RE0		I/O	ST	数字 I/O。
LCDBIAS1		I	Analog	LCD 的 BIAS1 输入。
RE1/LCDBIAS2	1			
RE1		I/O	ST	数字 I/O。
LCDBIAS2		I	Analog	LCD 的 BIAS2 输入。
LCDBIAS3	64	I	Analog	LCD 的 BIAS3 输入。
RE3/COM0	63			
RE3		I/O	ST	数字 I/O。
COM0		O	Analog	LCD 的 COM0 输出。
RE4/COM1	62			
RE4		I/O	ST	数字 I/O。
COM1		O	Analog	LCD 的 COM1 输出。
RE5/COM2	61			
RE5		I/O	ST	数字 I/O。
COM2		O	Analog	LCD 的 COM2 输出。
RE6/COM3	60			
RE6		I/O	ST	数字 I/O。
COM3		O	Analog	LCD 的 COM3 输出。
RE7/CCP2/SEG31	59			
RE7		I/O	ST	数字 I/O。
CCP2(2)		I/O	ST	捕捉 2 输入 / 比较 2 输出 /PWM2 输出。
SEG31		O	Analog	LCD 的 SEG31 输出。

图注： TTL = TTL 兼容输入；ST = CMOS 电平的施密特触发器输入；I = 输入；P = 电源；I^2C™ = I^2C/SMBus；CMOS = CMOS 兼容输入或输出；Analog = 模拟输入；O = 输出；OD = 漏极开路 （没有 P 型二极管连接到 V_{DD}）

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。

2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-3： PIC18F6XJ90 引脚说明 （续）

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
				PORTF 是双向 I/O 端口。
RF1/AN6/C2OUT/SEG19	17			
RF1		I/O	ST	数字 I/O。
AN6		I	Analog	模拟输入 6。
C2OUT		O	—	比较器 2 输出。
SEG19		O	Analog	LCD 的 SEG19 输出。
RF2/AN7/C1OUT/SEG20	16			
RF2		I/O	ST	数字 I/O。
AN7		I	Analog	模拟输入 7。
C1OUT		O	—	比较器 1 输出。
SEG20		O	Analog	LCD 的 SEG20 输出。
RF3/AN8/SEG21/C2INB	15			
RF3		I/O	ST	数字 I/O。
AN8		I	Analog	模拟输入 8。
SEG21		O	Analog	LCD 的 SEG21 输出。
C2INB		I	Analog	比较器 2 的输入 B。
RF4/AN9/SEG22/C2INA	14			
RF4		I/O	ST	数字 I/O。
AN9		I	Analog	模拟输入 9。
SEG22		O	Analog	LCD 的 SEG22 输出。
C2INA		I	Analog	比较器 2 的输入 A。
RF5/AN10/CVREF/ SEG23/C1INB	13			
RF5		I/O	ST	数字 I/O。
AN10		I	Analog	模拟输入 10。
CVREF		O	Analog	比较器参考电压输出。
SEG23		O	Analog	LCD 的 SEG23 输出。
C1INB		I	Analog	比较器 1 的输入 B。
RF6/AN11/SEG24/C1INA	12			
RF6		I/O	ST	数字 I/O。
AN11		I	Analog	模拟输入 11。
SEG24		O	Analog	LCD 的 SEG24 输出。
C1INA		I	Analog	比较器 1 的输入 A。
RF7/AN5/$\overline{SS}$/SEG25	11			
RF7		I/O	ST	数字 I/O。
AN5		O	Analog	模拟输入 5。
$\overline{SS}$		I	TTL	SPI 从选择输入。
SEG25		O	Analog	LCD 的 SEG25 输出。

图注： TTL = TTL 兼容输入 CMOS = CMOS 兼容输入或输出
ST = CMOS 电平的施密特触发器输入 Analog = 模拟输入
I = 输入 O = 输出
P = 电源 OD = 漏极开路 （没有 P 型二极管连接到 VDD）
I^2C™ = I^2C/SMBus

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。
2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-3： PIC18F6XJ90 引脚说明 （续）

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
				PORTG 是双向 I/O 端口。
RG0/LCDBIAS0	3			
RG0		I/O	ST	数字 I/O。
LCDBIAS0		I	Analog	LCD 的 BIAS0 输入。
RG1/TX2/CK2	4			
RG1		I/O	ST	数字 I/O。
TX2		O	—	AUSART 异步发送。
CK2		I/O	ST	AUSART 同步时钟 （见相关的 RX2/DT2 引脚信息）。
RG2/RX2/DT2/VLCAP1	5			
RG2		I/O	ST	数字 I/O。
RX2		I	ST	AUSART 异步接收。
DT2		I/O	ST	AUSART 同步数据 （见相关的 TX2/CK2 引脚信息）。
VLCAP1		I	Analog	LCD 电荷泵电容输入。
RG3/VLCAP2	6			
RG3		I/O	ST	数字 I/O。
VLCAP2		I	Analog	LCD 电荷泵电容输入。
RG4/SEG26/RTCC	8			
RG4		I/O	ST	数字 I/O。
SEG26		O	Analog	LCD 的 SEG26 输出。
RTCC		O	—	RTCC 输出。
VSS	9, 25, 41, 56	P	—	逻辑和 I/O 引脚的参考地。
VDD	26, 38, 57	P	—	逻辑和 I/O 引脚的正电源。
AVSS	20	P	—	模拟模块的参考地。
AVDD	19	P	—	模拟模块的正电源。
ENVREG	18	I	ST	片上稳压器使能端。
VDDCORE/VCAP	10			内核逻辑电源或外部滤波电容连接。
VDDCORE		P	—	单片机内核逻辑的正电源 （禁止稳压器）。
VCAP		P	—	外部滤波电容连接 （使能稳压器）。

图注： TTL = TTL 兼容输入
ST = CMOS 电平的施密特触发器输入
I = 输入
P = 电源
I^2C™ = I^2C/SMBus
CMOS = CMOS 兼容输入或输出
Analog = 模拟输入
O = 输出
OD = 漏极开路 （没有 P 型二极管连接到 VDD）

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。
2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-4： PIC18F8XJ90 引脚说明

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
$\overline{MCLR}$	9	I	ST	主复位 （输入）或编程电压 （输入）。此引脚为低电平有效的器件复位输入端。
OSC1/CLKI/RA7	49			晶振或外部时钟输入。
OSC1		I	CMOS	晶振输入。
CLKI		I	CMOS	外部时钟源输入。总是与 OSC1 引脚功能相关联。（见相关的 OSC1/CLKI 和 OSC2/CLKO 引脚信息。）
RA7		I/O	TTL	通用 I/O 引脚。
OSC2/CLKO/RA6	50			晶振或时钟输出。
OSC2		O	—	晶振输出。在晶振模式下，该引脚与晶振或谐振器相连。
CLKO		O	—	在EC模式下，OSC2引脚输出CLKO信号，其频率是OSC1引脚上信号频率的 1/4，该频率等于指令周期的倒数。
RA6		I/O	TTL	通用 I/O 引脚。
				PORTA 是双向 I/O 端口。
RA0/AN0	30			
RA0		I/O	TTL	数字 I/O。
AN0		I	Analog	模拟输入 0。
RA1/AN1/SEG18	29			
RA1		I/O	TTL	数字 I/O。
AN1		I	Analog	模拟输入 1。
SEG18		O	Analog	LCD 的 SEG18 输出。
RA2/AN2/VREF-	28			
RA2		I/O	TTL	数字 I/O。
AN2		I	Analog	模拟输入 2。
VREF-		I	Analog	A/D 参考电压 （低电压）输入。
RA3/AN3/VREF+	27			
RA3		I/O	TTL	数字 I/O。
AN3		I	Analog	模拟输入 3。
VREF+		I	Analog	A/D 参考电压 （高电压）输入。
RA4/T0CKI/SEG14	34			
RA4		I/O	ST	数字 I/O。
T0CKI		I	ST	Timer0 外部时钟输入。
SEG14		O	Analog	LCD 的 SEG14 输出。
RA5/AN4/SEG15	33			
RA5		I/O	TTL	数字 I/O。
AN4		I	Analog	模拟输入 4。
SEG15		O	Analog	LCD 的 SEG15 输出。
RA6				见 OSC2/CLKO/RA6 引脚信息。
RA7				见 OSC1/CLKI/RA7 引脚信息。

图注： TTL = TTL 兼容输入
ST = CMOS 电平的施密特触发器输入
I = 输入
P = 电源
I^2C™ = I^2C/SMBus
CMOS = CMOS 兼容输入或输出
Analog = 模拟输入
O = 输出
OD = 漏极开路 （没有 P 型二极管连接到 VDD）

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。
2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-4： PIC18F8XJ90 引脚说明 （续）

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
				PORTB 是双向 I/O 端口。PORTB 的所有输入端都可软件编程为内部弱上拉。
RB0/INT0/SEG30	58			
RB0		I/O	TTL	数字 I/O。
INT0		I	ST	外部中断 0。
SEG30		O	Analog	LCD 的 SEG30 输出。
RB1/INT1/SEG8	57			
RB1		I/O	TTL	数字 I/O。
INT1		I	ST	外部中断 1。
SEG8		O	Analog	LCD 的 SEG8 输出。
RB2/INT2/SEG9/CTED1	56			
RB2		I/O	TTL	数字 I/O。
INT2		I	ST	外部中断 2。
SEG9		O	Analog	LCD 的 SEG9 输出。
CTED1		I	ST	CTMU 边沿 1 输入。
RB3/INT3/SEG10/CTED2	55			
RB3		I/O	TTL	数字 I/O。
INT3		I	ST	外部中断 3。
SEG10		O	Analog	LCD 的 SEG10 输出。
CTED2		I	ST	CTMU 边沿 2 输入。
RB4/KBI0/SEG11	54			
RB4		I/O	TTL	数字 I/O。
KBI0		I	TTL	电平变化中断引脚。
SEG11		O	Analog	LCD 的 SEG11 输出。
RB5/KBI1/SEG29	53			
RB5		I/O	TTL	数字 I/O。
KBI1		I	TTL	电平变化中断引脚。
SEG29		O	Analog	LCD 的 SEG29 输出。
RB6/KBI2/PGC	52			
RB6		I/O	TTL	数字 I/O。
KBI2		I	TTL	电平变化中断引脚。
PGC		I/O	ST	在线调试器和 ICSP™ 编程时钟引脚。
RB7/KBI3/PGD	47			
RB7		I/O	TTL	数字 I/O。
KBI3		I	TTL	电平变化中断引脚。
PGD		I/O	ST	在线调试器和 ICSP 编程数据引脚。

图注： TTL = TTL 兼容输入 CMOS = CMOS 兼容输入或输出
ST = CMOS 电平的施密特触发器输入 Analog = 模拟输入
I = 输入 O = 输出
P = 电源 OD = 漏极开路 （没有 P 型二极管连接到 V_{DD}）
I^2C™ = I^2C/SMBus

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。
2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-4： PIC18F8XJ90 引脚说明 （续）

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
				PORTC 是双向 I/O 端口。
RC0/T1OSO/T13CKI	36			
RC0		I/O	ST	数字 I/O。
T1OSO		O	—	Timer1 振荡器输出。
T13CKI		I	ST	Timer1/Timer3 外部时钟输入。
RC1/T1OSI/CCP2/SEG32	35			
RC1		I/O	ST	数字 I/O。
T1OSI		I	CMOS	Timer1 振荡器输入。
CCP2(1)		I/O	ST	捕捉 2 输入 / 比较 2 输出 /PWM2 输出。
SEG32		O	Analog	LCD 的 SEG32 输出。
RC2/CCP1/SEG13	43			
RC2		I/O	ST	数字 I/O。
CCP1		I/O	ST	捕捉 1 输入 / 比较 1 输出 /PWM1 输出。
SEG13		O	Analog	LCD 的 SEG13 输出。
RC3/SCK/SCL/SEG17	44			
RC3		I/O	ST	数字 I/O。
SCK		I/O	ST	SPI 模式的同步串行时钟输入 / 输出。
SCL		I/O	I^2C	I^2C™ 模式的同步串行时钟输入 / 输出。
SEG17		O	Analog	LCD 的 SEG17 输出。
RC4/SDI/SDA/SEG16	45			
RC4		I/O	ST	数字 I/O。
SDI		I	ST	SPI 数据输入。
SDA		I/O	I^2C	I^2C 数据 I/O。
SEG16		O	Analog	LCD 的 SEG16 输出。
RC5/SDO/SEG12	46			
RC5		I/O	ST	数字 I/O。
SDO		O	—	SPI 数据输出。
SEG12		O	Analog	LCD 的 SEG12 输出。
RC6/TX1/CK1/SEG27	37			
RC6		I/O	ST	数字 I/O。
TX1		O	—	EUSART 异步发送。
CK1		I/O	ST	EUSART 同步时钟 （见相关的 RX1/DT1 引脚信息）。
SEG27		O	Analog	LCD 的 SEG27 输出。
RC7/RX1/DT1/SEG28	38			
RC7		I/O	ST	数字 I/O。
RX1		I	ST	EUSART 异步接收。
DT1		I/O	ST	EUSART 同步数据 （见相关的 TX1/CK1 引脚信息）。
SEG28		O	Analog	LCD 的 SEG28 输出。

图注：
TTL = TTL 兼容输入
ST = CMOS 电平的施密特触发器输入
I = 输入
P = 电源
I^2C™ = I^2C/SMBus
CMOS = CMOS 兼容输入或输出
Analog = 模拟输入
O = 输出
OD = 漏极开路 （没有 P 型二极管连接到 V_{DD}）

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。
2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-4： PIC18F8XJ90 引脚说明 （续）

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
				PORTD 是双向 I/O 端口。
RD0/SEG0/CTPLS	72			
RD0		I/O	ST	数字 I/O。
SEG0		O	Analog	LCD 的 SEG0 输出。
CTPLS		O	ST	CTMU 脉冲发生器输出。
RD1/SEG1	69			
RD1		I/O	ST	数字 I/O。
SEG1		O	Analog	LCD 的 SEG1 输出。
RD2/SEG2	68			
RD2		I/O	ST	数字 I/O。
SEG2		O	Analog	LCD 的 SEG2 输出。
RD3/SEG3	67			
RD3		I/O	ST	数字 I/O。
SEG3		O	Analog	LCD 的 SEG3 输出。
RD4/SEG4	66			
RD4		I/O	ST	数字 I/O。
SEG4		O	Analog	LCD 的 SEG4 输出。
RD5/SEG5	65			
RD5		I/O	ST	数字 I/O。
SEG5		O	Analog	LCD 的 SEG5 输出。
RD6/SEG6	64			
RD6		I/O	ST	数字 I/O。
SEG6		O	Analog	LCD 的 SEG6 输出。
RD7/SEG7	63			
RD7		I/O	ST	数字 I/O。
SEG7		O	Analog	LCD 的 SEG7 输出。

图注： TTL = TTL 兼容输入
ST = CMOS 电平的施密特触发器输入
I = 输入
P = 电源
I^2C™ = I^2C/SMBus
CMOS = CMOS 兼容输入或输出
Analog = 模拟输入
O = 输出
OD = 漏极开路 （没有 P 型二极管连接到 V_{DD}）

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。
2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-4： PIC18F8XJ90 引脚说明 （续）

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
				PORTE 是双向 I/O 端口。
RE0/LCDBIAS1	4			
RE0		I/O	ST	数字 I/O。
LCDBIAS1		I	Analog	LCD 的 BIAS1 输入。
RE1/LCDBIAS2	3			
RE1		I/O	ST	数字 I/O。
LCDBIAS2		I	Analog	LCD 的 BIAS2 输入。
LCDBIAS3	78	I	Analog	LCD 的 BIAS3 输入。
RE3/COM0	77			
RE3		I/O	ST	数字 I/O。
COM0		O	Analog	LCD 的 COM0 输出。
RE4/COM1	76			
RE4		I/O	ST	数字 I/O。
COM1		O	Analog	LCD 的 COM1 输出。
RE5/COM2	75			
RE5		I/O	ST	数字 I/O。
COM2		O	Analog	LCD 的 COM2 输出。
RE6/COM3	74			
RE6		I/O	ST	数字 I/O。
COM3		O	Analog	LCD 的 COM3 输出。
RE7/CCP2/SEG31	73			
RE7		I/O	ST	数字 I/O。
CCP2(2)		I/O	ST	捕捉 2 输入 / 比较 2 输出 /PWM2 输出。
SEG31		O	Analog	LCD 的 SEG31 输出。

图注： TTL = TTL 兼容输入
ST = CMOS 电平的施密特触发器输入
I = 输入
P = 电源
I^2C™ = I^2C/SMBus
CMOS = CMOS 兼容输入或输出
Analog = 模拟输入
O = 输出
OD = 漏极开路 （没有 P 型二极管连接到 V_{DD}）

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。
2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-4： PIC18F8XJ90 引脚说明 （续）

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
				PORTF 是双向 I/O 端口。
RF1/AN6/C2OUT/SEG19	23			
RF1		I/O	ST	数字 I/O。
AN6		I	Analog	模拟输入 6。
C2OUT		O	—	比较器 2 输出。
SEG19		O	Analog	LCD 的 SEG19 输出。
RF2/AN7/C1OUT/SEG20	18			
RF2		I/O	ST	数字 I/O。
AN7		I	Analog	模拟输入 7。
C1OUT		O	—	比较器 1 输出。
SEG20		O	Analog	LCD 的 SEG20 输出。
RF3/AN8/SEG21/C2INB	17			
RF3		I/O	ST	数字 I/O。
AN8		I	Analog	模拟输入 8。
SEG21		O	Analog	LCD 的 SEG21 输出。
C2INB		I	Analog	比较器 2 的输入 B。
RF4/AN9/SEG22/C2INA	16			
RF4		I/O	ST	数字 I/O。
AN9		I	Analog	模拟输入 9。
SEG22		O	Analog	LCD 的 SEG22 输出。
C2INA		I	Analog	比较器 2 的输入 A。
RF5/AN10/CVREF/ SEG23/C1INB	15			
RF5		I/O	ST	数字 I/O。
AN10		I	Analog	模拟输入 10。
CVREF		O	Analog	比较器参考电压输出。
SEG23		O	Analog	LCD 的 SEG23 输出。
C1INB		I	Analog	比较器 1 的输入 B。
RF6/AN11/SEG24/C1INA	14			
RF6		I/O	ST	数字 I/O。
AN11		I	Analog	模拟输入 11。
SEG24		O	Analog	LCD 的 SEG24 输出。
C1INA		I	Analog	比较器 1 的输入 A。
RF7/AN5/$\overline{SS}$/SEG25	13			
RF7		I/O	ST	数字 I/O。
AN5		O	Analog	模拟输入 5。
$\overline{SS}$		I	TTL	SPI 从选择输入。
SEG25		O	Analog	LCD 的 SEG25 输出。

图注： TTL = TTL 兼容输入；ST = CMOS 电平的施密特触发器输入；I = 输入；P = 电源；I^2C™ = I^2C/SMBus；CMOS = CMOS 兼容输入或输出；Analog = 模拟输入；O = 输出；OD = 漏极开路 （没有 P 型二极管连接到 VDD）

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。

2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-4： PIC18F8XJ90 引脚说明 （续）

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
				PORTG 是双向 I/O 端口。
RG0/LCDBIAS0	5			
RG0		I/O	ST	数字 I/O。
LCDBIAS0		I	Analog	LCD 的 BIAS0 输入。
RG1/TX2/CK2	6			
RG1		I/O	ST	数字 I/O。
TX2		O	—	AUSART 异步发送。
CK2		I/O	ST	AUSART 同步时钟 （见相关的 RX2/DT2 引脚信息）。
RG2/RX2/DT2/VLCAP1	7			
RG2		I/O	ST	数字 I/O。
RX2		I	ST	AUSART 异步接收。
DT2		I/O	ST	AUSART 同步数据 （见相关的 TX2/CK2 引脚信息）。
VLCAP1		I	Analog	LCD 电荷泵电容输入。
RG3/VLCAP2	8			
RG3		I/O	ST	数字 I/O。
VLCAP2		I	Analog	LCD 电荷泵电容输入。
RG4/SEG26/RTCC	10			
RG4		I/O	ST	数字 I/O。
SEG26		O	Analog	LCD 的 SEG26 输出。
RTCC		O	—	RTCC 输出。

图注： TTL = TTL 兼容输入
ST = CMOS 电平的施密特触发器输入
I = 输入
P = 电源
I^2C™ = I^2C/SMBus
CMOS = CMOS 兼容输入或输出
Analog = 模拟输入
O = 输出
OD = 漏极开路 （没有 P 型二极管连接到 VDD）

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。
2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-4： PIC18F8XJ90 引脚说明 （续）

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
				PORTH 是双向 I/O 端口。
RH0/SEG47	79			
RH0		I/O	ST	数字 I/O。
SEG47		O	Analog	LCD 的 SEG47 输出。
RH1/SEG46	80			
RH1		I/O	ST	数字 I/O。
SEG46		O	Analog	LCD 的 SEG46 输出。
RH2/SEG45	1			
RH2		I/O	ST	数字 I/O。
SEG45		O	Analog	LCD 的 SEG45 输出。
RH3/SEG44	2			
RH3		I/O	ST	数字 I/O。
SEG44		O	Analog	LCD 的 SEG44 输出。
RH4/SEG40	22			
RH4		I/O	ST	数字 I/O。
SEG40		O	Analog	LCD 的 SEG40 输出。
RH5/SEG41	21			
RH5		I/O	ST	数字 I/O。
SEG41		O	Analog	LCD 的 SEG41 输出。
RH6/SEG42	20			
RH6		I/O	ST	数字 I/O。
SEG42		O	Analog	LCD 的 SEG42 输出。
RH7/SEG43	19			
RH7		I/O	ST	数字 I/O。
SEG43		O	Analog	LCD 的 SEG43 输出。

图注： TTL = TTL 兼容输入
ST = CMOS 电平的施密特触发器输入
I = 输入
P = 电源
I^2C™ = I^2C/SMBus
CMOS = CMOS 兼容输入或输出
Analog = 模拟输入
O = 输出
OD = 漏极开路 （没有 P 型二极管连接到 V_{DD}）

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。
2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 1-4： PIC18F8XJ90 引脚说明 （续）

引脚名称	引脚编号 TQFP	引脚类型	缓冲器类型	说明
				PORTJ 是双向 I/O 端口。
RJ0	62	I/O	ST	数字 I/O。
RJ1/SEG33	61			
RJ1		I/O	ST	数字 I/O。
SEG33		O	Analog	LCD 的 SEG33 输出。
RJ2/SEG34	60			
RJ2		I/O	ST	数字 I/O。
SEG34		O	Analog	LCD 的 SEG34 输出。
RJ3/SEG35	59			
RJ3		I/O	ST	数字 I/O。
SEG35		O	Analog	LCD 的 SEG35 输出。
RJ4/SEG39	39			
RJ4		I/O	ST	数字 I/O。
SEG39		O	Analog	LCD 的 SEG39 输出。
RJ5/SEG38	40			
RJ5		I/O	ST	数字 I/O
SEG38		O	Analog	LCD 的 SEG38 输出。
RJ6/SEG37	41			
RJ6		I/O	ST	数字 I/O。
SEG37		O	Analog	LCD 的 SEG37 输出。
RJ7/SEG36	42			
RJ7		I/O	ST	数字 I/O。
SEG36		O	Analog	LCD 的 SEG36 输出。
VSS	11, 31, 51, 70	P	—	逻辑和 I/O 引脚的参考地。
VDD	32, 48, 71	P	—	逻辑和 I/O 引脚的正电源。
AVSS	26	P	—	模拟模块的参考地。
AVDD	25	P	—	模拟模块的正电源。
ENVREG	24	I	ST	片上稳压器使能端。
VDDCORE/VCAP	12			内核逻辑电源或外部滤波电容连接。
VDDCORE		P	—	单片机内核逻辑的正电源 （禁止稳压器）。
VCAP		P	—	外部滤波电容连接 （使能稳压器）。

图注： TTL = TTL 兼容输入
ST = CMOS 电平的施密特触发器输入
I = 输入
P = 电源
I^2C™ = I^2C/SMBus
CMOS = CMOS 兼容输入或输出
Analog = 模拟输入
O = 输出
OD = 漏极开路 （没有 P 型二极管连接到 VDD）

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。
2： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

注：

2.0 PIC18FJ 系列单片的入门指南

2.1 基本连接要求

在着手使用 PIC18F87J90 系列 8 位单片机进行开发之前必须至少注意一些器件引脚的连接。

以下引脚必须始终连接：

- 所有 VDD 和 VSS 引脚 （见**第 2.2 节 “ 电源引脚 ”**）
- 所有 AVDD 和 AVSS 引脚 （不管是否使用模拟器件功能）（见**第 2.2 节 “ 电源引脚 ”**）
- $\overline{\text{MCLR}}$ 引脚 （见**第 2.3 节 “ 主复位 （MCLR）引脚 ”**）
- ENVREG/DISVREG （如果实现）和 VCAP/VDDCORE 引脚 （见**第 2.4 节 “ 稳压器引脚 （ENVREG 和 VCAP/VDDCORE） ”**）

如果在最终应用中使用了以下引脚，那么也必须连接这些引脚：

- 用于在线串行编程 （ICSP™）和调试的 PGC/PGD 引脚 （见**第 2.5 节 “ICSP 引脚 ”**）
- 使用外部振荡器源时用到的 OSCI 和 OSCO 引脚（见**第 2.6 节 “ 外部振荡器引脚 ”**）

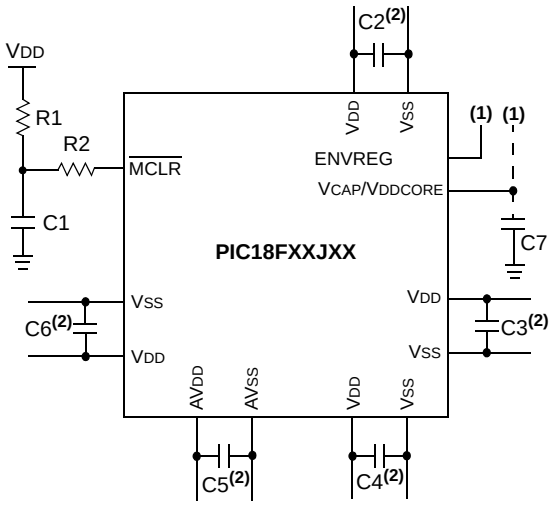
此外，还可能需要以下引脚：

- 模拟模块使用外部参考电压时需要的 VREF+/VREF- 引脚

注：	不管是否使用了任何模拟模块， AVDD 和 AVSS 引脚必须始终连接。

图 2-1 给出了最起码的连接要求。

图 2-1： 建议的最少连接



要点 （所有值都是建议值）：

C1 至 C6：0.1 µF、 20V 的陶瓷电容

C7：10 µF、 6.3V 或更高的钽电容或陶瓷电容

R1：10 kΩ

R2：100Ω 至 470Ω

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

2： 在此给出的示例针对的是具有 5 个 VDD/VSS 和 AVDD/AVSS 引脚对的 PIC18F 器件。其他器件具有的引脚对数可能稍有不同；应相应的调整去耦电容数。

2.2 电源引脚

2.2.1 去耦电容

需要在每对电源引脚（例如 VDD 和 VSS 以及 AVDD 和 AVSS）上使用去耦电容。

使用去耦电容时需考虑以下条件：

- **电容的值和类型：**建议采用 0.1 µF（100 nF）、10-20V 的电容。应使用谐振频率在 200 MHz 及更高范围内的低 ESR 电容。建议使用陶瓷电容。
- **印刷电路板上的放置：**去耦电容应尽可能靠近相应的引脚放置。建议将电容放到电路板上与器件相同的一侧。如果空间有限，可使用过孔将电容放到 PCB 的另一层上；但是，需要确保从引脚到电容的走线长度不超过 0.25 英寸（6 mm）。
- **处理高频噪声：**如果电路板产生高达几十兆赫兹的高频噪声，请在上述去耦电容旁并联另一个陶瓷电容。该电容值的范围为 0.01 µF 至 0.001 µF。请将这个电容靠近每个主去耦电容放置。在高速电路设计中，请考虑在尽可能靠近电源和接地引脚的地方布置 10 对电容（例如，一个 0.1 µF 的电容与一个 0.001 µF 的电容并联构成一对）。
- **最大程度提高性能：**在从电源电路开始布置电路板时，请首先将电源和回路走线连接到去耦电容，然后再连接到器件引脚。这可以确保去耦电容在电源链中处于第一位。保持电容和电源引脚之间走线长度尽可能短也同样重要，因为这可以减少 PCB 的走线感抗。

2.2.2 槽路电容

在电源走线长度大于 6 英寸的电路板上，建议在包含单片机的集成电路中使用槽路电容，以提供本地电源。槽路电容的值应根据以下因素确定：连接电源和器件的走线的电阻值以及应用中器件消耗的最大电流。也就是说，选择槽路电容使之满足器件的可接受的电压骤降要求。典型值范围为 4.7 µF 至 47 µF。

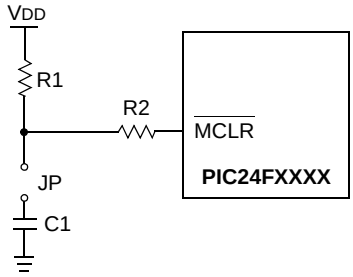
2.3 主复位（$\overline{\text{MCLR}}$）引脚

$\overline{\text{MCLR}}$ 引脚提供了两个特殊的器件功能：器件复位以及器件编程和调试功能。如果最终应用中不需要编程和调试功能，只需要将该引脚直接连接到 VDD。为帮助增加应用的电阻以避免因电压骤降而意外复位，可能需要添加其他元件。图 2-1 给出了典型的配置。可根据应用的要求实现其他电路设计。

在编程和调试期间，必须考虑添加到引脚的电阻和电容。器件编程器和调试器驱动 $\overline{\text{MCLR}}$ 引脚。因此，不允许对特殊电压电平（VIH 和 VIL）以及快速信号转换造成不良影响。这就需要根据应用和 PCB 的要求调整 R1 和 C1 的具体值。例如，建议在编程和调试操作期间通过使用跳线将电容 C1 和 $\overline{\text{MCLR}}$ 引脚隔离（图 2-2）。正常的运行时操作不需要这样的跳线操作。

与 $\overline{\text{MCLR}}$ 引脚关联的任何元件应放置在距离该引脚0.25 英寸（6 mm）的范围内。

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



注　1： 建议 R1 ≤ 10 kΩ。建议起始值为 10 kΩ。确保满足 $\overline{\text{MCLR}}$ 引脚 VIH 和 VIL 规范的要求。

2： R2 ≤ 470Ω 会限制任何电流从外部电容 C 流入 $\overline{\text{MCLR}}$，从而避免由于静电放电（Electrostatic Discharge，ESD）或过度电应力（Electrical Overstress，EOS）导致 $\overline{\text{MCLR}}$ 引脚损坏。确保满足 $\overline{\text{MCLR}}$ 引脚 VIH 和 VIL 规范的要求。

2.4 稳压器引脚（ENVREG 和 VCAP/VDDCORE）

片内稳压器使能引脚 ENVREG 必须始终直接连接电源电压引脚或接地引脚。将 ENVREG 连接到 VDD 使能稳压器，将其接地则禁止稳压器。关于连接和使用片内稳压器的详细信息，请参见**第 25.3 节 “ 片上稳压器 ”**。

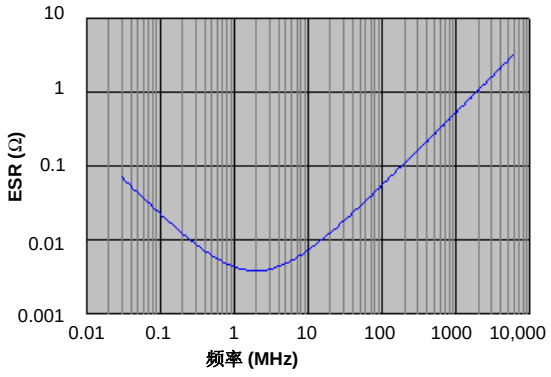
使能了稳压器时，需要在 VCAP/VDDCORE 引脚上连接一个低 ESR（<5Ω）的电容以稳定稳压器输出电压。VCAP/VDDCORE 引脚不得与 VDD 相连，而是必须使用一个 10 µF 的电容接地。电容类型可以是陶瓷电容或钽电容。Murata GRM21BF50J106ZE01（10 µF，6.3 V）或同等规格的电容就很合适。设计人员可使用图 2-3 来评估侯选器件的 ESR。

建议走线长度不要超过 0.25 英寸（6 mm）。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 的电源相连。关于 VDD 和 VDDCORE 的信息，请参见**第 28.0 节 “ 电气特性 ”**。

请注意，一些低引脚数的 “LF” 版 PIC18FJ 器件（例如 PIC18LF45J10）没有 ENVREG 引脚。这些器件中的稳压器是永久禁止的。必须始终在 VDDCORE 引脚上供电才能工作。

图 2-3： 建议 VCAP 的频率和 ESR 性能



注： 给出的 Murata GRM21BF50J106ZE01 数据是在温度为 25°C 时，直流偏压为 0V 时测得的。

2.5 ICSP 引脚

PGC/PGD 引脚用于在线串行编程（ICSP™）和调试。建议保持器件上的 ICSP 连接器和 ICSP 引脚之间的走线长度尽可能短。如果预期 ICSP 连接器上会发生 ESD 事件，建议使用一个串联电阻，且电阻值在几十欧姆范围内，不要超过 100Ω。

建议不要在 PGEC 和 PGED 引脚上使用上拉电阻、串联二极管和电容，因为它们会干扰编程器 / 调试器与器件之间的通信。如果应用需要这类分立元件，应在编程和调试期间将它们从电路中除去。或者，参阅相关器件闪存编程规范中的交流 / 直流特性和时序要求信息，以了解关于容性负载限制以及引脚高输入电压（VIH）和低输入电压（VIL）要求的信息。

对于器件仿真，请确保编程到器件中的 “ 通信通道选择 ”（即 PGCx/PGDx 引脚）与 ICSP 到 Microchip 的调试器 / 仿真器工具的物理连接一致。

更多关于可用 Microchip 开发工具的连接要求的信息，请参见**第 27.0 节 “ 开发支持 ”**。

2.6 外部振荡器引脚

许多单片机可以至少有两个振荡器：一个高频主振荡器和一个低频辅助振荡器（详细信息，请参见**第 3.0 节“振荡器配置”**）。

振荡器电路应放到电路板上与器件相同的一侧。将振荡器电路靠近相关振荡器引脚放置，且电路元件与引脚之间的距离不要超过 0.5 英寸（12 mm）。负载电容应在电路板的同一侧挨着振荡器本身放置。

在振荡器电路周围设置接地灌铜区，将其与周围电路隔离。接地灌铜区应直接连接到 MCU 地。不要在接地灌铜区内使用任何信号线或电源线。而且，如果使用双面的电路板，请避免在电路板上放置晶振的位置的另一面布线。

图 2-4 给出了一些布线建议。直插式封装可以采用可完全容纳振荡器引脚的单面布线来处理。对于引脚排列紧密的器件，单面布局则可能无法始终完全地容纳所有引脚和元件。一种适合的解决方案是将含有保护走线的部分连接到反面的接地层。在所有情形中，保护走线都必须回接到地。

在规划应用的走线和 I/O 分配时，需要确保相邻端口引脚和其他邻近振荡器的信号是无害的（即无高频，无短暂上升和下降，以及无其他类似噪声）。

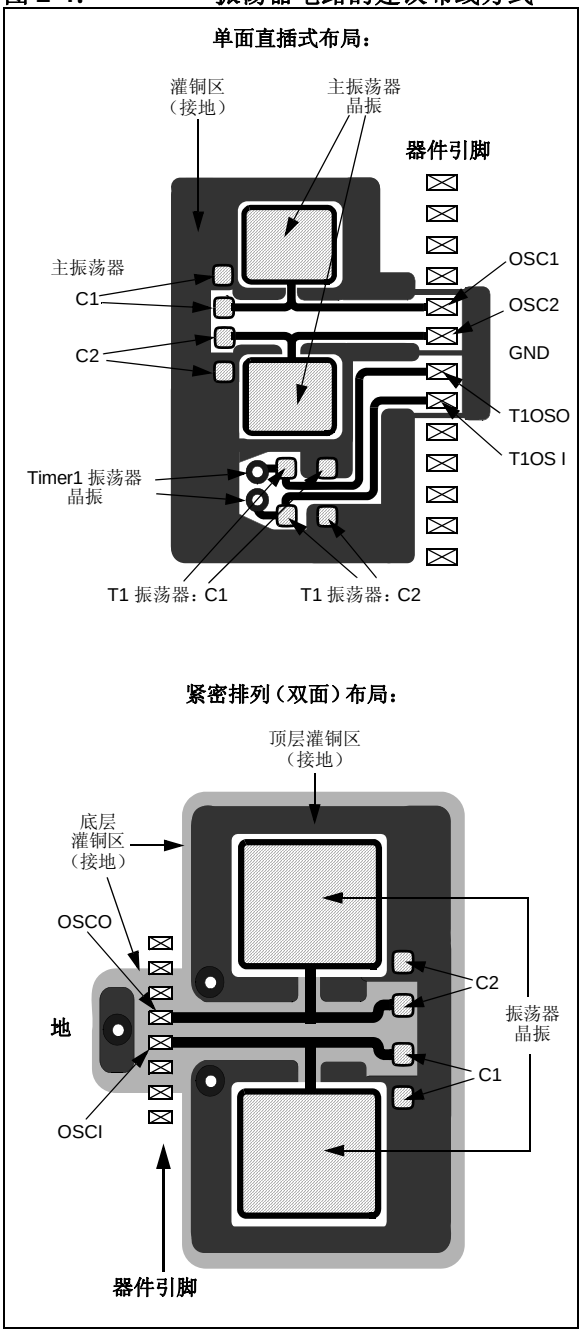
关于振荡器电路的其他信息和设计指导，请参见我公司网站（www.microchip.com）上提供的以下应用笔记：

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

2.7 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。

图 2-4： 振荡器电路的建议布线方式



3.0 振荡器配置

3.1 振荡器类型

PIC18F87J90 系列器件可以在 8 种不同的振荡器模式下工作：

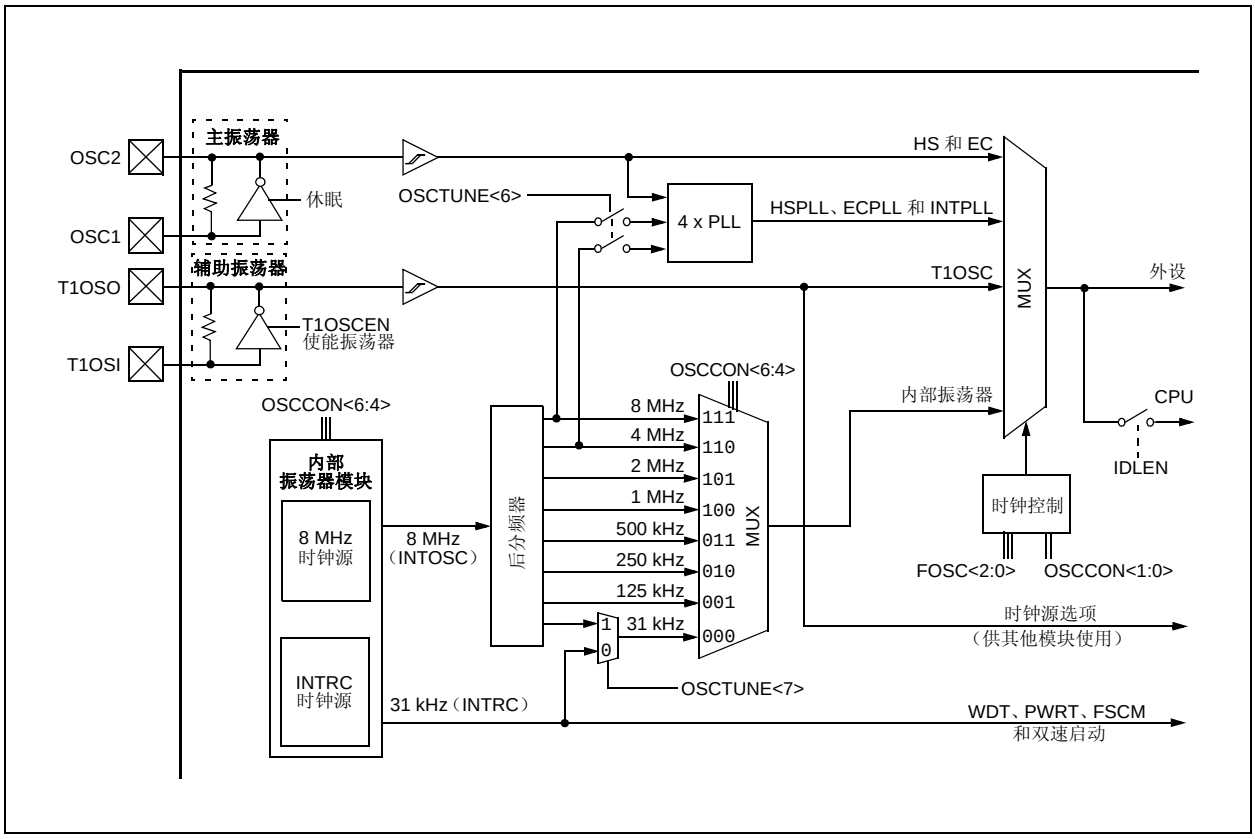
1. ECPLL　OSC1/OSC2 作为主振荡器；使能 PLL 的 ECPLL 振荡器，RA6 用作 CLKO 引脚
2. EC　OSC1/OSC2 作为主振荡器；带 FOSC/4 输出的外部时钟
3. HSPLL　OSC1/OSC2 作为主振荡器；带软件 PLL 控制的高速晶振 / 谐振器
4. HS　OSC1/OSC2 作为主振荡器；高速晶振 / 谐振器
5. INTPLL1　带软件 PLL 控制的内部振荡器模块，RA6 引脚输出 FOSC/4 信号，RA7 用作 I/O 引脚
6. INTIO1　内部振荡器模块，RA6 引脚输出 FOSC/4 信号， RA7 用作 I/O 引脚
7. INTPLL2　带软件 PLL 控制的内部振荡器模块，RA6 和 RA7 均用作 I/O 引脚
8. INTIO2　内部振荡器模块，RA6 和 RA7 均用作 I/O 引脚

用户可以通过编程 FOSC<2:0> 配置位来选择所有这些模式。

此外，PIC18F87J90 系列器件可以在软件控制下或在某些条件下自动在不同时钟源之间进行切换。通过实时管理器件时钟速度而无需复位应用，进一步节省了功耗。

图 3-1 显示了 PIC18F87J90 系列器件的时钟源。

图 3-1：　PIC18F87J90 系列时钟框图



3.2 控制寄存器

OSCCON 寄存器（寄存器 3-1）控制器件时钟操作的主要方面。它选择要使用的振荡器类型、要调用的功耗管理模式以及 INTOSC 时钟源的输出频率。它还提供振荡器的状态。

OSCTUNE 寄存器（寄存器 3-2）控制内部振荡器模块的调节和操作。它还实现了 PLLEN 位，该位用于控制锁相环（PLL）的操作（见**第 3.4.3 节 “PLL 倍频器 ”**）。

寄存器 3-1： OSCCON：振荡器控制寄存器

R/W-0	R/W-1	R/W-1	R/W-0	R(1)	R-0	R/W-0	R/W-0
IDLEN	IRCF2(2)	IRCF1(2)	IRCF0(2)	OSTS	IOFS	SCS1(4)	SCS0(4)
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 **IDLEN：**空闲使能位
1 = 执行 SLEEP 指令后器件进入空闲模式
0 = 执行 SLEEP 指令后器件进入休眠模式

bit 6-4 **IRCF<2:0>：**INTOSC 时钟源频率选择位 (2)
111 = 8 MHz （由 INTOSC 直接驱动时钟）
110 = 4 MHz （默认值）
101 = 2 MHz
100 = 1 MHz
011 = 500 kHz
010 = 250 kHz
001 = 125 kHz
000 = 31 kHz （来自 INTOSC/256 或 INTRC） (3)

bit 3 **OSTS：**振荡器起振定时器延时状态位 (1)
1 = 振荡器起振定时器 （Oscillator Start-up Timer， OST）延时已结束；主振荡器正在运行
0 = 振荡器起振定时器 （OST）延时正在进行；主振荡器尚未就绪

bit 2 **IOFS：**INTOSC 频率稳定位
1 = 快速 RC 振荡器频率稳定
0 = 快速 RC 振荡器频率不稳定

bit 1-0 **SCS<1:0>：**系统时钟选择位 (4)
11 = 内部振荡器模块
10 = 主振荡器
01 = Timer1 振荡器
00 = 默认的主振荡器 （由 FOSC<2:0> 配置位定义）

注 1： 复位状态取决于 IESO 配置位的状态。
2： 如果由内部振荡器提供器件时钟，修改这些位将导致立即进行时钟频率切换。
3： 时钟源由 INTSRC 位 （OSCTUNE<7>）选择，请参见上文。
4： 修改这些位将导致立即进行时钟源切换。

寄存器 3-2：　　OSCTUNE：振荡器调节寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INTSRC	PLLEN	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0
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　　**INTSRC：**内部振荡器低频时钟源选择位
1 = 来自 8 MHz INTOSC 时钟源的 31.25 kHz 器件时钟 （使能 256 分频）
0 = 来自 INTRC 31 kHz 振荡器的 31 kHz 器件时钟

bit 6　　**PLLEN：**倍频器 PLL 使能位
1 = 使能 PLL
0 = 禁止 PLL

bit 5-0　　**TUN<5:0>：**快速 RC 振荡器 （INTOSC）频率调节位
011111 = 最高频率
•　　•
•　　•
000001
000000 = 中心频率。快速 RC 振荡器以已校准的频率运行。
111111
•　　•
•　　•
100000 = 最低频率

3.3　时钟源与振荡器切换

基本上，PIC18F87J90系列器件都有3种独立的时钟源：

- 主振荡器
- 辅助振荡器
- 内部振荡器

主振荡器可认为是主要的器件振荡器。这些是指与OSC1 和 OSC2 引脚连接的任何外部振荡器，包括外部晶振和谐振器模式以及外部时钟模式。如果通过FOSC<2:0> 配置位选择，则内部振荡器模块 （31 kHz INTRC 或 8 MHz INTOSC 时钟源）可认为是主振荡器。特定的模式由 FOSC 配置位定义。这些模式的详细信息将在**第 3.4 节 “ 外部振荡器模式 ”** 中进行介绍。

辅助振荡器是指那些不与OSC1或OSC2引脚连接的外部时钟源。即使在控制器处于功耗管理模式时，这些时钟源仍可继续工作。 PIC18F87J90 系列器件将 Timer1 振荡器作为辅助振荡器源。此振荡器（在所有功耗管理模式中）通常是实时时钟 （Real-Time Clock， RTC）等功能的时基。将在**第 12.0 节 “Timer1 模块 ”** 中详细讨论 Timer1 振荡器。

除了在某些情况下作为主时钟源之外，**内部振荡器**还可以作为功耗管理模式的时钟源。INTRC 时钟源也可作为几种特殊功能部件（例如WDT和故障保护时钟监视器）的时钟源。在**第 3.5 节 “ 内部振荡器模块 ”** 中对内部振荡器模块进行了更详细的讨论。

PIC18F87J90 系列包含了允许器件时钟源从主振荡器（由器件配置选择）切换到备用时钟源之一的功能。当使能备用时钟源时，可以使用多种功耗管理工作模式。

3.3.1 时钟源选择

系统时钟选择位 SCS<1:0>（OSCCON<1:0>）用于选择时钟源。可用的时钟源包括主时钟（由 FOSC<2:0> 配置位定义）、辅助时钟（Timer1 振荡器）和内部振荡器。当写入一个或多个位之后，接着是一段很短的时钟转换间隔，然后时钟源会改变。

OSTS（OSCCON<3>）和 T1RUN（T1CON<6>）位指出当前提供器件时钟的是哪一个时钟源。OSTS 位置 1 表明振荡器起振定时器（OST）已超时且主时钟在主时钟模式下提供器件时钟。T1RUN 位置 1 表明 Timer1 振荡器正在辅助时钟模式下提供器件时钟。在功耗管理模式下，任何时候这些位中只有一位会置 1。如果这些位都没有置 1，则表示当前时钟源是 INTRC，或内部振荡器刚刚起振而尚未稳定。

IDLEN 位决定当执行 SLEEP 指令时器件是进入休眠模式还是某个空闲模式。

第 4.0 节“功耗管理模式”更详细地讨论了 OSCCON 寄存器中标志位和控制位的使用。

注 1： 要选择辅助时钟源，必须使能 Timer1 振荡器。通过将Timer1控制寄存器中的T1OSCEN 位（T1CON<3>）置 1，可以使能 Timer1 振荡器。如果未使能 Timer1 振荡器，则在执行 SLEEP 指令时选择辅助时钟源的任何尝试都会被忽略。

2： 建议在Timer1振荡器稳定工作之后再执行 SLEEP 指令，否则当 Timer1 振荡器起振时可能会发生很长的延时。

3.3.1.1 系统时钟选择和器件复位

发生所有形式的复位时 SCS 位都会被清零，这意味着 FOSC<2:0> 配置位定义的主振荡器用作器件复位时的主时钟源。这可以是内部振荡器模块本身，或是其他主时钟源（HS、EC、HSPLL、ECPLL1/2 或 INTPLL1/2）之一。

在内部振荡器模块（无 PLL）作为复位时默认时钟的情况下，快速 RC 振荡器（INTOSC）将被用作器件时钟源。它将从 1 MHz 开始起振，这是对应于 IRCF<2:0> 位复位值（100）的后分频比选择。

不管选择了哪个主振荡器，INTRC 总是会在器件上电时被使能。它将作为时钟源直到器件已从存储器中装入了它的配置值。此时 FOSC 配置位被读取并选择了振荡器的工作模式。

请注意，主时钟源或内部振荡器在任何给定时间都将有两种可能的 SCS<1:0> 位设置选项。

3.3.2 振荡器转换

PIC18F87J90 系列器件包含在时钟源切换时防止时钟产生“毛刺”的电路。在切换时钟时，器件时钟会有短暂的停顿。该停顿的时间长度是旧时钟源的两个周期加上新时钟源的三到四个周期的和。此公式假设新时钟源是稳定的。

第 4.1.2 节“进入功耗管理模式”详细讨论了时钟转换。

3.4 外部振荡器模式

3.4.1 晶振 / 陶瓷谐振器 （HS 模式）

在 HS 或 HSPLL 振荡器模式下，将晶振或陶瓷谐振器连接在 OSC1 和 OSC2 引脚之间来产生振荡信号。图 3-2 显示了引脚连接方式。

振荡器的设计要求使用平行切割的晶体。

注：	使用顺序切割的晶体，可能会使振荡器产生的频率超出晶体制造厂商所给的规范。

表 3-1： 陶瓷谐振器的电容选择

使用的典型电容值：			
模式	**频率**	**OSC1**	**OSC2**
HS	8.0 MHz 16.0 MHz	27 pF 22 pF	27 pF 22 pF

上述电容值仅供设计参考。

要达到理想的振荡器工作状况，可能需要不同的电容值。用户应当在应用要求的 VDD 和温度范围下测试振荡器的性能。请参见以下应用笔记以获取振荡器具体信息：

- AN588， *"PIC® Microcontroller Oscillator Design Guide"*
- AN826， *"Crystal Oscillator Basics and Crystal Selection for rfPIC® and PIC® Devices"*
- AN849， *"Basic PIC® Oscillator Design"*
- AN943， *"Practical PIC® Oscillator Analysis and Design"*
- AN949， *"Making Your Oscillator Work"*

更多信息，请参见表 3-2 下方的 " 注 "。

表 3-2： 晶振的电容选择

振荡器类型	晶振频率	已测试的典型电容值：	
		C1	**C2**
HS	4 MHz	27 pF	27 pF
	8 MHz	22 pF	22 pF
	20 MHz	15 pF	15 pF

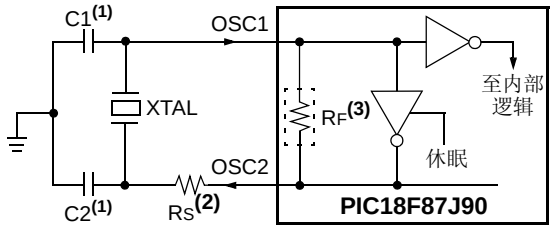
上述电容值仅供设计参考。

要达到理想的振荡器工作状况，可能需要不同的电容值。用户应当在应用要求的 VDD 和温度范围下测试振荡器的性能。

请参见表 3-1 中列出的 Microchip 应用笔记以获取振荡器的相关信息。更多信息，请参见本表下方的 " 注 "。

注	**1：** 电容值越大，振荡器的稳定性越高，但同时起振时间也越长。
	2： 因为每种谐振器 / 晶振都有其自身特性，用户应当向谐振器 / 晶振制造厂商询问外部元件的适当值。
	3： 可能需要使用电阻 RS 以避免对低驱动规格的晶体造成过驱动。
	4： 应始终验证振荡器在应用要求的 VDD 和温度范围下的性能。

图 3-2： 晶振 / 陶瓷谐振器工作原理（HS 或 HSPLL 配置）



注 **1：** 关于C1和C2的初始值，请参见表3-1和表3-2。

2： 对于 AT 条形切割的晶体，可能需要串联一个电阻 （RS）。

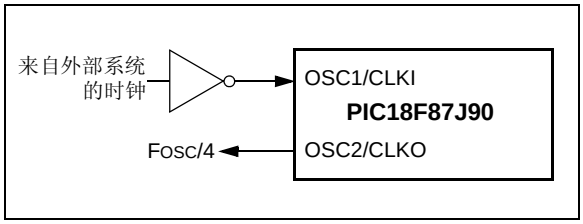
3： RF 的值随选定的振荡器模式变化。

3.4.2 外部时钟输入 （EC 模式）

EC 和 ECPLL 振荡器模式要求 OSC1 引脚与一个外部时钟源相连。在上电复位后或从休眠模式退出后，不需要振荡器起振时间。

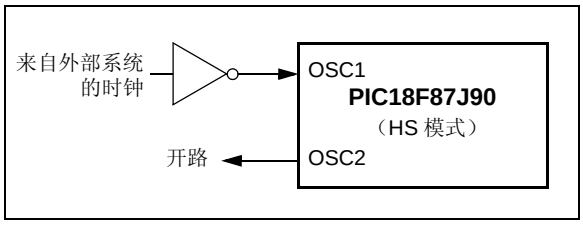
在 EC 振荡器模式下，由 OSC2 引脚输出振荡器频率的 4 分频信号。此信号可用于测试或同步其他逻辑。图 3-3 显示了 EC 振荡器模式的引脚连接方式。

图 3-3： 外部时钟输入工作原理（EC 配置）



如图 3-4 所示，在 HS 模式下， OSC1 引脚也可以连接外部时钟源。在此配置中，OSC2 上无法得到 4 分频输出信号。此配置中的电流消耗稍高于 EC 模式，因为内部振荡器的反馈电路将被使能（在 EC 模式下，反馈电路被禁止）。

图 3-4： 外部时钟输入工作原理（HS 振荡器配置）



3.4.3 PLL 倍频器

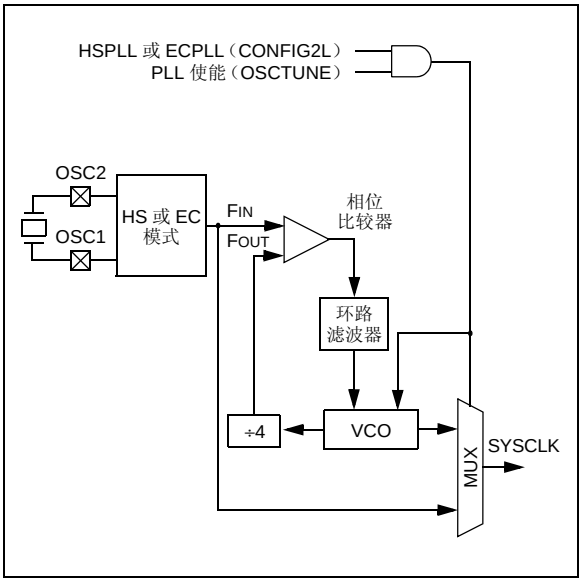
如果用户希望使用低频振荡器电路或通过晶振将器件频率调节至其最高额定频率，可以选择使用锁相环（PLL）电路。对于担心高频晶振引起 EMI 或需要内部振荡器提供高速时钟的用户而言，这样做可能有用。

3.4.3.1 HSPLL 和 ECPLL 模式

HSPLL 和 ECPLL 模式可以以外部振荡源频率的 4 倍运行器件，从而最高频率可达 40 MHz。

通过将 FOSC<2:0> 配置位编程为 111（对于 ECPLL）或 101（对于 HSPLL），可以使能 PLL。此外，还必须将 PLLEN 位 （OSCTUNE<6>）置 1。清零 PLLEN 会禁止 PLL，与所选的振荡器配置无关。它还允许用软件更灵活地控制应用的时钟速度。

图 3-5： PLL 框图



3.4.3.2 PLL 和 INTOSC

当内部振荡器模块配置为主时钟源时，内部振荡器模块也可以使用 PLL。在此配置下，用软件使能 PLL 并产生最高为 32 MHz 的时钟输出。**第 3.5.2 节 “INTPLL 模式 ”** 描述了带 PLL 的 INTOSC 的工作原理。

3.5 内部振荡器模块

PIC18F87J90 系列器件包含可产生两种不同时钟信号的内部振荡器模块；这两种信号均可充当单片机的时钟源，从而避免在 OSC1 和 / 或 OSC2 引脚上使用外部振荡器电路。

主输出是快速 RC 振荡器或 INTOSC，这是一个 8 MHz 的时钟源，可以用于直接驱动器件时钟。它还驱动一个后分频器，该分频器可提供从 31 kHz 到 4 MHz 的时钟频率。当选择了 125 kHz 到 8 MHz 的时钟频率时，就会使能 INTOSC。当选择了 31 kHz 的时钟频率时，根据 INTSRC 位 （OSCTUNE<7>）的设置，还能使能 INTOSC 输出。

另一个时钟源是内部 RC 振荡器 （INTRC），它提供了标称值为 31 kHz 的输出。如果选择 INTRC 作为器件的时钟源，它就会被使能；当使能以下任一功能时，也将自动使能 INTRC：

- 上电延时定时器
- 故障保护时钟监视器
- 看门狗定时器
- 双速启动

第 25.0 节 “CPU 的特殊功能 ” 将详细讨论以上功能。

通过配置 OSCCON 寄存器的 IRCF 位，可以选择时钟源频率 （INTOSC 直接频率、INTOSC 后分频频率或 INTRC 直接频率）。器件复位时的默认频率为 4 MHz。

3.5.1 INTIO 模式

使用内部振荡器作为时钟源可以避免使用最多两个外部振荡器引脚，从而可将它们用作数字 I/O。有两种不同的振荡器配置 （由 FOSC 配置位决定）可用：

- 在 INTIO1 模式下， OSC2 引脚输出 FOSC/4，而 OSC1充当RA7（见图3-6），用于数字输入和输出。
- 在 INTIO2 模式下， OSC1 充当 RA7， OSC2 充当 RA6 （见图 3-7），两者都用于数字输入和输出。

图 3-6： INTIO1 振荡器模式

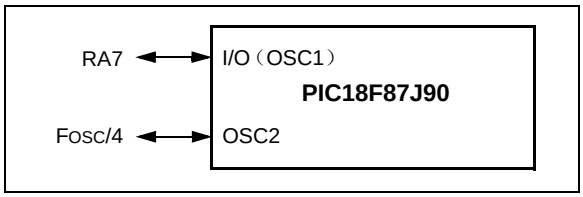
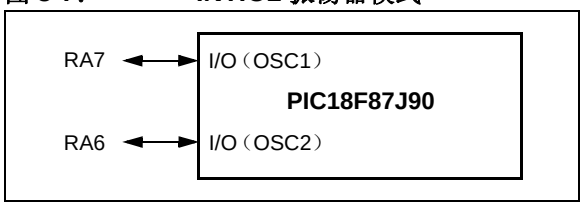


图 3-7： INTIO2 振荡器模式



3.5.2 INTPLL 模式

内部振荡器模块可以通过使用 4 倍频锁相环 （PLL）来产生比一般内部振荡器源所能产生的时钟速度更快的器件时钟速度。当使能时，PLL 可产生 16 MHz 或 32 MHz 的时钟速度。

PLL操作通过软件控制。控制位PLLEN（OSCTUNE<6>）用于使能或禁止其操作。仅当器件被配置为使用INTPLL模式之一作为主时钟源 （FOSC<2:0> = 011 或 001）时， INTOSC 才可使用 PLL。此外，仅当选定的输出频率为 4 MHz 或 8 MHz（OSCCON<6:4> = 111 或 110）时， PLL 才会工作。

与 INTIO 模式一样，有两种不同的 INTPLL 模式可用：

- 在 INTPLL1 模式下， OSC2 引脚输出 FOSC/4，而 OSC1 充当 RA7，用于数字输入和输出。从外部看，这与 INTIO1 完全相同 （图 3-6）。
- 在 INTPLL2 模式下，OSC1 充当 RA7，OSC2 充当 RA6，两者都用于数字输入和输出。从外部看，这与 INTIO2 完全相同 （图 3-7）。

3.5.3 内部振荡器输出频率和调节

出厂时已校准了内部振荡器模块使之能够产生8 MHz的INTOSC 输出频率。可以通过写 OSCTUNE 寄存器（寄存器 3-2）中的 TUN<5:0>（OSCTUNE<5:0>），在用户应用中进行调整。

当修改了 OSCTUNE 寄存器后，INTOSC 的频率将变为新的频率。振荡器将在 1 ms 内稳定下来。在此变化期间，代码会继续执行，不会有任何迹象表明时钟频率发生了改变。

INTRC 振荡器的工作独立于 INTOSC 时钟源。电压和温度变化导致的 INTOSC 变化并不一定会使 INTRC 变化，反之亦然。INTRC 的频率不受 OSCTUNE 的影响。

3.5.4 INTOSC 频率漂移

INTOSC 频率可能会随着 V_{DD} 电压或温度的改变而发生漂移，这一点可能会以各种方式影响控制器的运行。通过修改 OSCTUNE 寄存器的值可以调节 INTOSC 的频率。根据不同器件，这可能不会对 INTRC 时钟源频率造成影响。

调节 INTOSC 需要了解何时调节、调节的方向以及在某些情况下的调整量。这里给出了三种补偿技术。

3.5.4.1 用 EUSART 进行补偿

当 EUSART 开始产生帧错误，或者在异步模式下接收数据有错误时可能需要进行调节。帧错误表示器件时钟的频率太高。要对此进行调整，可以减小 OSCTUNE 中的值来降低时钟频率。另一方面，数据中有错误可能表明时钟速度太低。要进行补偿，可以增大 OSCTUNE 中的值来提高时钟频率。

3.5.4.2 用定时器进行补偿

此技术是将器件时钟的速度与某一个参考时钟进行比较。可能要用到两个定时器；一个由外设时钟提供时钟源，而另一个由一个固定的参考源（如 Timer1 振荡器）提供时钟源。

两个定时器都被清零，但由参考源提供时钟信号的定时器产生中断。当发生中断时，使用内部时钟源的定时器值被读取且两个定时器均被清零。如果使用内部时钟源的定时器的值比期望值大很多，则表示内部振荡器模块运行过快。要对此进行调整，需减小 OSCTUNE 寄存器中的值。

3.5.4.3 在捕捉模式下用 CCP 模块进行补偿

CCP 模块可以使用由内部振荡器模块提供时钟信号的独立运行 Timer1（或 Timer3）和已知周期的外部事件（即交流电源频率）。在 CCPRxH:CCPRxL 寄存器中捕捉并记录第一个事件的时间以备以后使用。当第二个事件导致捕捉时，要用第二个事件的时间减去第一个事件的时间。由于外部事件的周期是已知的，因此可以计算事件之间的时间差。

如果测得的时间比计算得到的时间大很多，则表示内部振荡器模块运行过快。要进行补偿，需减小 OSCTUNE 寄存器中的值。如果测得的时间比计算得到的时间小很多，则表示内部振荡器模块运行过慢。要进行补偿，需增大 OSCTUNE 寄存器中的值。

3.6 功耗管理模式对各种时钟源的影响

当选取 PRI_IDLE 模式时，指定的主振荡器会继续运行而不中断。对于所有其他功耗管理模式，使用 OSC1 引脚的振荡器会被禁止。OSC1 引脚（以及由振荡器使用的 OSC2 引脚）将会停止振荡。

在辅助时钟模式（SEC_RUN和SEC_IDLE）下，Timer1 振荡器作为器件时钟源工作。如果需要，Timer1 振荡器也可以运行在所有功耗管理模式下为 Timer1 或 Timer3 提供时钟。

在 RC_RUN 和 RC_IDLE 模式下，由内部振荡器提供器件时钟源。无论是哪种功耗管理模式，31 kHz 的 INTRC 输出均可被直接用来提供时钟并且可被使能来支持多种特殊的功能部件 （关于 WDT、故障保护时钟监视器和双速启动的更多信息，请参见**第 25.2 节 “ 看门狗定时器（WDT）”** 到**第 25.5 节 “ 故障保护时钟监视器 ”**）。

如果选择了休眠模式，所有的时钟源都会被停止。因为休眠模式切断了所有晶体管的开关电流，休眠模式能实现最低的器件电流消耗 （仅泄漏电流）。

在休眠期间使能任何片上功能都将增加休眠时的电流消耗。要支持 WDT 工作，需要使能 INTRC。Timer1 振荡器可以用来为实时时钟（RTC）提供时钟源。不需要器件时钟源的其他功能部件也可以工作（即，MSSP 从器件、 INTx 引脚和其他等）。在**第 28.2 节 “ 直流特性：PIC18F87J90 系列的掉电电流和供电电流（工业级）”** 中列出了可能显著增加电流消耗的外设。

3.7 上电延时

由两个定时器控制上电延时，这样大多数应用都无需外接复位电路。上电延时可以确保在器件电源稳定（常规环境下）和主时钟稳定工作之前器件保持在复位状态。关于上电延时的更多信息，请参见**第 5.6 节 “ 上电延时定时器 （PWRT） ”**。

第一个定时器是上电延时定时器 （Power-up Timer，PWRT），在上电时它提供了固定的延迟时间（表 28-11 中的参数 33）；它总是使能的。

第二个定时器是振荡器起振定时器 （OST），用于在晶振稳定前使芯片保持在复位状态（HS 模式）。 OST 在计数 1024 个振荡周期后允许振荡器为器件提供时钟。

POR 之后有一个 TCSD 间隔的延时 （表 28-11 中的参数 38），在此延时期间控制器为执行指令做准备。

表 3-3： 休眠模式下 OSC1 和 OSC2 引脚的状态

振荡器模式	OSC1 引脚	OSC2 引脚
EC 和 ECPLL	悬空，由外部时钟驱动	处于逻辑低电平 （时钟 /4 输出）
HS 和 HSPLL	反馈反相器被禁止，处于静止电压值	反馈反相器被禁止，处于静止电压值
INTOSC 和 INTPLL1/2	I/O 引脚 RA6，方向由 TRISA<6> 控制	I/O 引脚 RA6，方向由 TRISA<7> 控制

注： 关于由休眠和 $\overline{\text{MCLR}}$ 复位引起的延时，请参见**第 5.0 节 “ 复位 ”**。

注：

4.0 功耗管理模式

PIC18F87J90 系列器件提供了只需通过管理 CPU 和外设的时钟源就可以管理功耗的功能。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。为了在应用中管理功耗，提供了三种主要的工作模式：

- 运行模式
- 空闲模式
- 休眠模式

这些模式定义了需要为器件的哪些部分提供时钟以及时钟的速度。运行模式和空闲模式可以使用三种时钟源（主时钟源、辅助时钟源或内部振荡器）中的任意一种；而休眠模式则不使用时钟源。

功耗管理模式包括几个由早期的 PIC® 器件提供的节省功耗的功能。其中之一就是其他 PIC18 器件也提供的时钟切换功能，该功能允许使用 Timer1 振荡器代替主振荡器。节省功耗的功能还包括所有 PIC 器件都提供的休眠模式，在该模式下，器件所有的时钟都停止。

4.1 选择功耗管理模式

选择功耗管理模式之前需要先做出两个决定：是否为 CPU 提供时钟源以及选择何种时钟源。IDLEN 位（OSCCON<7>）控制是否为 CPU 提供时钟源，而 SCS<1:0> 位 （OSCCON<1:0>）选择时钟源。表 4-1 总结了各个模式下的位设置、时钟源和受影响的模块。

4.1.1 时钟源

SCS<1:0> 位允许为功耗管理模式在三个时钟源中任选其一。它们是：

- 主时钟，由 FOSC<2:0> 配置位定义
- 辅助时钟 （Timer1 振荡器）
- 内部振荡器

4.1.2 进入功耗管理模式

可以通过装载 OSCCON 寄存器从一种功耗管理模式切换到另一种功耗管理模式。SCS<1:0> 位选择时钟源并确定使用运行模式还是空闲模式。更改这些位会导致立即切换到一个新的时钟源 （假定新时钟源正在运行）。此切换可能会引起时钟转换延时。**第 4.1.3 节 “ 时钟转换和状态指示 ”** 及其后续章节将会讨论这些问题。

执行SLEEP指令可以触发进入功耗管理空闲模式或休眠模式。最后实际进入哪个模式由 IDLEN 位的状态决定。

更改功耗管理模式并不总是要求设置所有这些位，而是取决于当前的模式和将要切换到的模式。通过在发出 SLEEP 指令之前更改振荡器选择位或更改 IDLEN 位可完成多种模式转换。如果已经正确配置了 IDLEN 位，可能只需执行 SLEEP 指令就可切换到所需的模式。

表 4-1： 功耗管理模式

模式	OSCCON 位		模块时钟		可用时钟和振荡器源
	IDLEN<7>(1)	SCS<1:0>	CPU	外设	
休眠	0	N/A	关闭	关闭	无 —— 所有时钟被禁止
PRI_RUN	N/A	10	提供时钟	提供时钟	主时钟 ——HS、 EC、 HSPLL 和 ECPLL；这是正常的全功耗执行模式
SEC_RUN	N/A	01	提供时钟	提供时钟	辅助时钟 ——Timer1 振荡器
RC_RUN	N/A	11	提供时钟	提供时钟	内部振荡器
PRI_IDLE	1	10	关闭	提供时钟	主时钟 ——HS、 EC、 HSPLL 和 ECPLL
SEC_IDLE	1	01	关闭	提供时钟	辅助时钟 ——Timer1 振荡器
RC_IDLE	1	11	关闭	提供时钟	内部振荡器

注 1： IDLEN 在执行 SLEEP 指令时反映其值。

4.1.3 时钟转换和状态指示

在两个时钟源之间进行转换所需的时间长度是旧时钟源的两个周期与新时钟源的三到四个周期的和。此公式假设新时钟源是稳定的。

以下两位用于指明当前的时钟源及其状态：OSTS（OSCCON<3>）和 T1RUN （T1CON<6>）。一般来说，当处于给定的功耗管理模式时，这两个位中只有一个位会置 1。当 OSTS 位置 1 时，表明由主时钟提供器件时钟。当 T1RUN 位置 1 时，表明由 Timer1 振荡器提供时钟源。如果这些位均不置 1，则由 INTRC 为器件提供时钟信号。

注：	执行 SLEEP 指令并不一定会将器件置于休眠模式。它只是作为触发条件，让器件进入休眠模式或一种空闲模式，具体何种模式由 IDLEN 位的设置决定。

4.1.4 多条 SLEEP 命令

使用 SLEEP 指令调用功耗管理模式时，具体进入何种模式在该指令执行那一刻由 IDLEN 位的设置决定。如果执行了另一条 SLEEP 指令，器件将进入由此时 IDLEN 位指定的功耗管理模式。如果 IDLEN 位已更改，器件将进入由新的设置指定的新的功耗管理模式。

4.2 运行模式

在运行模式下，内核和外设的时钟都是激活的。这些运行模式之间的区别就在于时钟源的不同。

4.2.1 PRI_RUN 模式

PRI_RUN 模式是单片机的正常全功耗执行模式。除非使能了双速启动（详情请参见**第 25.4 节 “ 双速启动 ”**），该模式也是器件复位后的默认模式。在此模式下，OSTS 位置 1 （见**第 3.2 节 “ 控制寄存器 ”**）。

4.2.2 SEC_RUN 模式

SEC_RUN 模式与其他 PIC18 器件提供的 “ 时钟切换 ” 功能兼容。在此模式下， CPU 和外设将 Timer1 振荡器作为时钟源。这允许用户在使用高精度时钟源的情况下仍可获得较低的功耗。

通过将 SCS<1:0> 位设置为 01 可以进入 SEC_RUN 模式。器件时钟源被切换到 Timer1 振荡器 （见图 4-1），主振荡器被关闭， T1RUN 位（T1CON<6>）被置 1 并且 OSTS 位被清零。

注：	Timer1 振荡器应该在进入 SEC_RUN 模式之前就已经运行了。如果 SCS<1:0> 位被设置为 01 时， T1OSCEN 位未置 1，则不会进入 SEC_RUN 模式。如果使能了 Timer1 振荡器，但它尚未运行，器件时钟将会延时直到该振荡器起振。在这种情况下，最初的振荡器运行很不稳定，可能会导致无法预料的结果。

在从 SEC_RUN 模式转换到 PRI_RUN 模式时，在主时钟启动过程中外设和 CPU 继续使用 Timer1 振荡器作为时钟源。当主时钟就绪以后，时钟切换回主时钟（见图 4-2）。当时钟切换完成后， T1RUN 位被清零，OSTS 位被置 1 并且由主时钟提供器件时钟。这种唤醒不会影响 IDLEN 和 SCS 位； Timer1 振荡器继续运行。

图 4-1： 进入 SEC_RUN 模式的转换时序

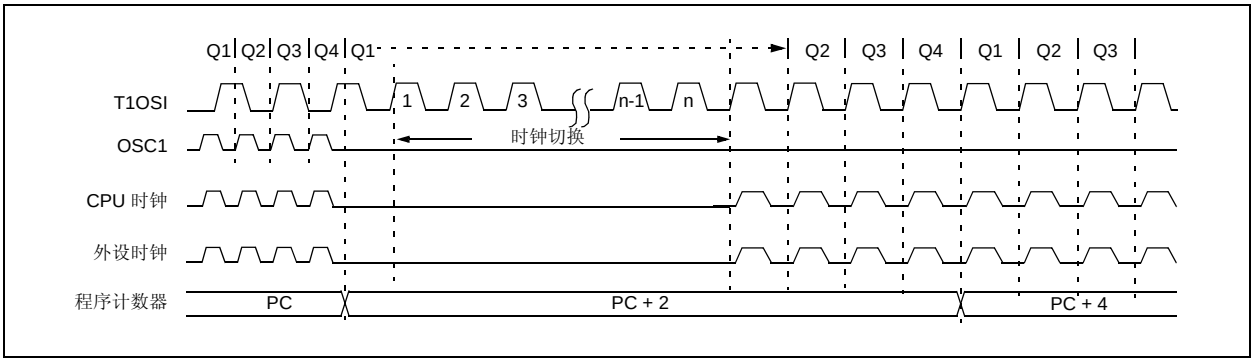
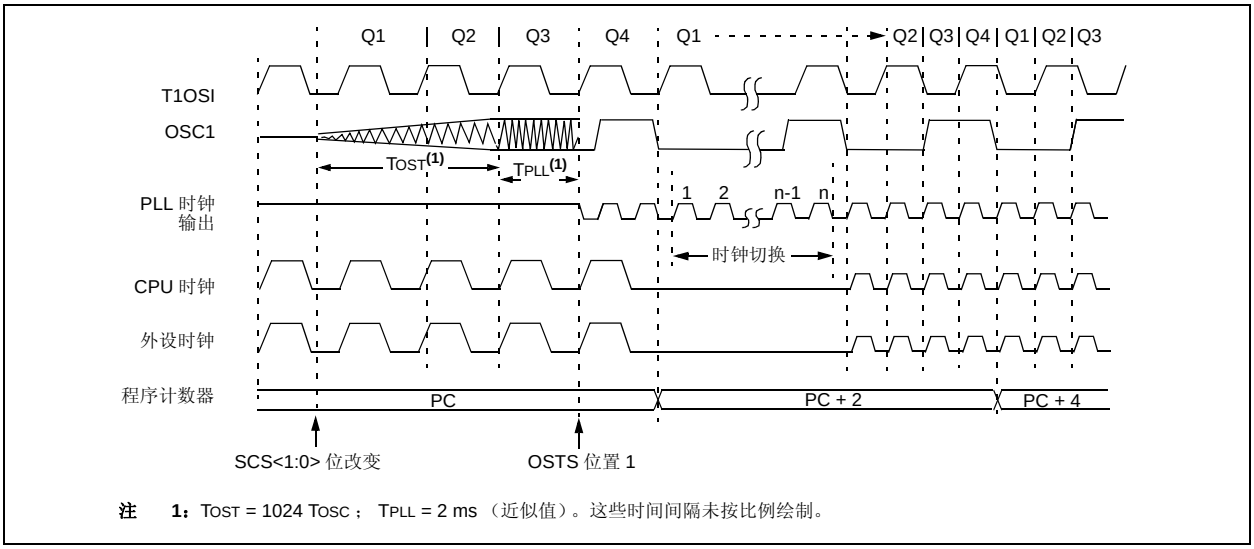


图 4-2： 从 SEC_RUN 模式切换到 PRI_RUN 模式的转换时序 （HSPLL）



注 1： TOST = 1024 TOSC； TPLL = 2 ms （近似值）。这些时间间隔未按比例绘制。

4.2.3 RC_RUN 模式

在 RC_RUN 模式下，内部振荡器作为 CPU 和外设的时钟源；主时钟关闭。此模式是在代码执行期间所有运行模式中最节省功耗的模式。它非常适用于对定时精度要求不高或者不是一直需要高速时钟的应用。

通过将 SCS 位设置为 11 可以进入此模式。当将时钟源切换到 INTRC 时（见图 4-3），主振荡器将被关闭并且 OSTS 位被清零。

在从 RC_RUN 模式转换到 PRI_RUN 模式期间，在主时钟处于启动状态时，器件将继续使用 INTRC 作为时钟源。当主时钟就绪以后，时钟切换到主时钟（见图4-4）。当时钟切换完成后， OSTS 位被置 1 并且由主时钟提供器件时钟。这种切换不会影响 IDLEN 和 SCS 位。如果使能了 WDT 或故障保护时钟监视器， INTRC 时钟源将继续运行。

图 4-3： 到 RC_RUN 模式的转换时序

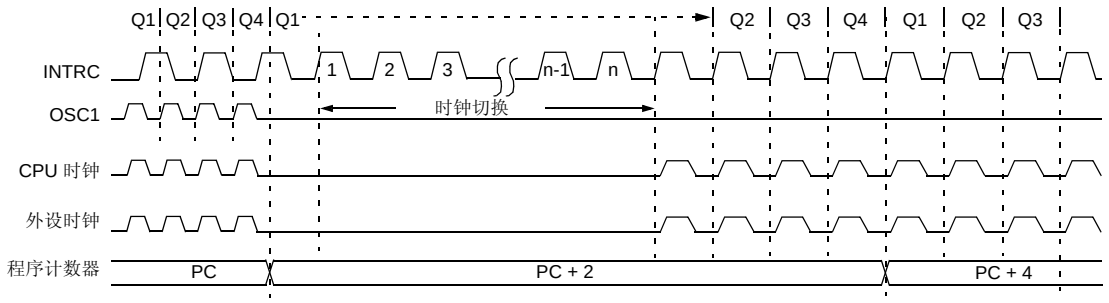
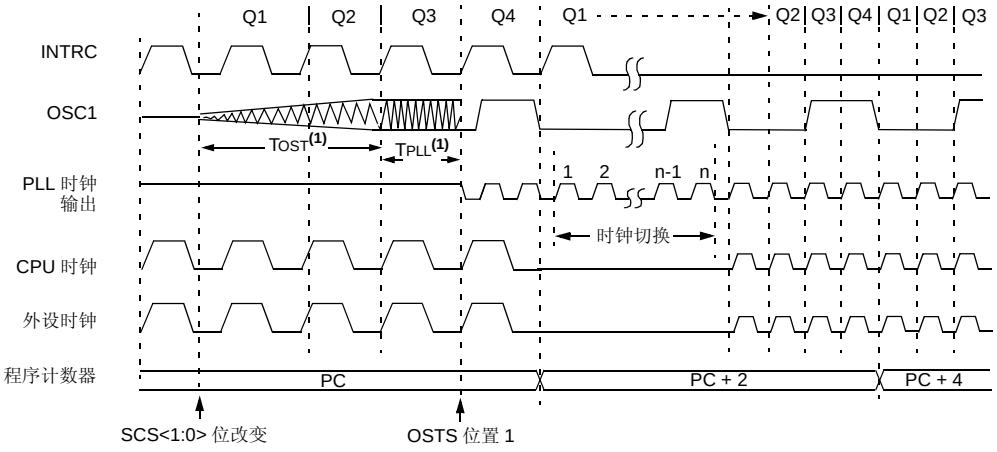


图 4-4： 从 RC_RUN 模式切换到 PRI_RUN 模式的转换时序



注 1： TOST = 1024 TOSC； TPLL = 2 ms （近似值）。这些时间间隔未按比例绘制。

4.3 休眠模式

功耗管理休眠模式和所有其他 PIC 器件提供的传统休眠模式相同。通过清零 IDLEN 位（器件复位时的默认状态）并执行 SLEEP 指令即可进入此模式。这将关闭选定的振荡器（图 4-5），并将所有的时钟源状态位清零。

从任何其他模式进入休眠模式不需要切换时钟。这是因为单片机一旦进入休眠模式就不需要时钟了。如果选择了WDT，INTRC时钟源将继续工作。如果使能了Timer1振荡器，它也将继续运行。

当在休眠模式下发生唤醒事件（通过中断、复位或 WDT 超时）时，在时钟源（通过 SCS<1:0> 位选择）就绪之前器件将没有时钟源（见图 4-6），或者如果使能了双速启动或故障保护时钟监视器，它将使用内部振荡器作为时钟源（见**第 25.0 节 “CPU 的特殊功能 ”**）。在这两种情况下，当由主时钟提供器件时钟时， OSTS 位将置 1。唤醒不会影响 IDLEN 和 SCS 位。

4.4 空闲模式

空闲模式允许在外设继续工作的同时有选择地关闭控制器的CPU。选择特定的空闲模式允许用户进一步管理功耗。

如果在执行 SLEEP 指令时， IDLEN 位被置 1，外设将使用由 SCS<1:0> 位选择的时钟源；而 CPU 没有时钟源。时钟源状态位不受影响。将 IDLEN 置 1 并执行 SLEEP 指令可以从给定的运行模式快速切换到相应的空闲模式。

如果选择了 WDT， INTRC 时钟源将继续工作。如果使能了 Timer1 振荡器，它也将继续运行。

由于 CPU 不在执行指令，器件只能通过中断、 WDT 超时或复位从空闲模式退出。当发生唤醒事件时， CPU 会在其准备执行代码前延时一个 TCSD 间隔（表 28-11 中的参数 38）。当 CPU 开始执行代码时，它将沿用当前空闲模式所使用的时钟源。例如，当从 RC_IDLE 模式唤醒时，将使用内部振荡器模块为 CPU 和外设提供时钟（即 RC_RUN 模式）。唤醒不会影响 IDLEN 和 SCS 位。

当处于任何空闲模式或休眠模式下时，WDT 超时会导致 WDT 唤醒并进入当前由 SCS<1:0> 位指定的运行模式。

图 4-5： 进入休眠模式的转换时序

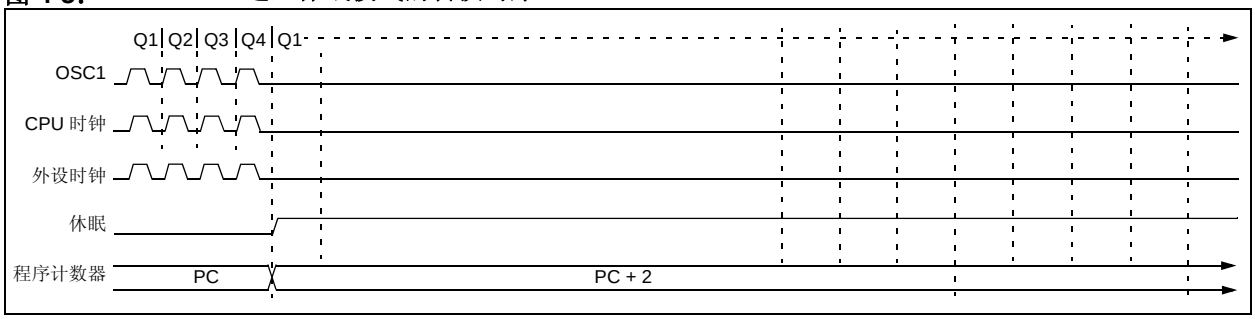
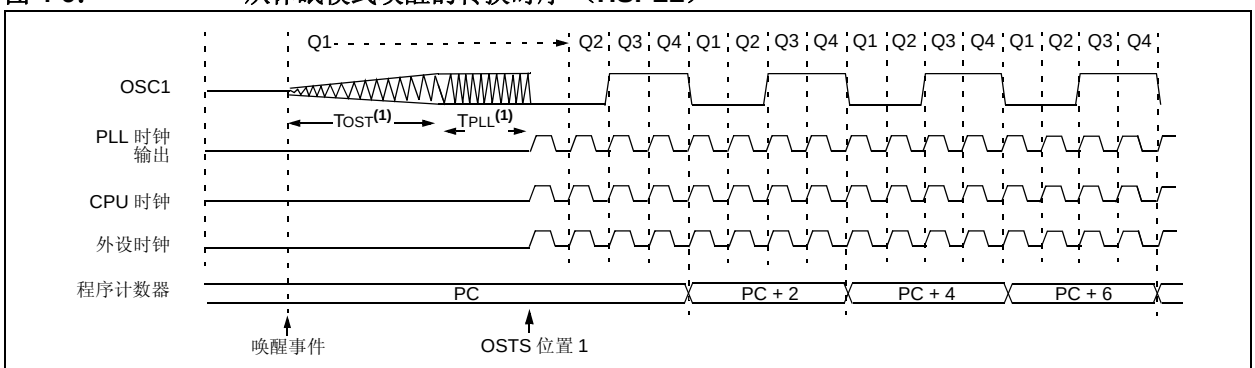


图 4-6： 从休眠模式唤醒的转换时序 （HSPLL）



注 1： TOST = 1024 TOSC； TPLL = 2 ms （近似值）。这些时间间隔未按比例绘制。

4.4.1 PRI_IDLE 模式

在三种低功耗空闲模式中，只有该模式不会禁止主器件时钟。由于时钟源不需要“ 预热 ”或是从其他振荡器转换过来，对于时序敏感的应用，选用此模式可以使用较精确的主时钟源以最快的速度恢复器件工作。

可以通过将 IDLEN 位置 1 并执行 SLEEP 指令以实现从 PRI_RUN 模式进入 PRI_IDLE 模式。如果器件处于另一种运行模式，首先将 IDLEN 位置 1，然后将 SCS 位设置为 10 并执行 SLEEP。虽然 CPU 已被禁止，但外设仍可继续使用由 FOSC<1:0> 配置位指定的主时钟源为其提供时钟信号。 OSTS 位保持置 1 （见图 4-7）。

当发生唤醒事件时，由主时钟源为 CPU 提供时钟。在唤醒事件和代码执行开始之间需要一个TCSD间隔的延时。该延时用来让 CPU 做好执行指令的准备。在唤醒之后，OSTS 位保持置 1 状态。这种唤醒不会影响 IDLEN 和 SCS 位 （见图 4-8）。

4.4.2 SEC_IDLE 模式

在 SEC_IDLE 模式下， CPU 被禁止，但外设继续将 Timer1 振荡器作为时钟源。可以通过将 IDLEN 位置 1 并执行 SLEEP 指令从 SEC_RUN 模式进入此模式。如果器件处于另一种运行模式，首先将 IDLEN 置 1，然后将 SCS<1:0> 设置为 01 并执行 SLEEP。当时钟源切换到 Timer1 振荡器时，主振荡器被关闭， OSTS 位被清零并且 T1RUN 位被置 1。

当唤醒事件发生时，外设继续将 Timer1 振荡器作为时钟源。唤醒事件发生后经过一个 TCSD 时间间隔， CPU 开始执行代码并使用 Timer1 振荡器作为其时钟源。这种唤醒不会影响 IDLEN 和 SCS 位。Timer1 振荡器继续运行 （见图 4-8）。

注：	Timer1 振荡器应该在进入 SEC_IDLE 模式之前就已经运行了。如果执行 SLEEP 指令时，T1OSCEN 位未置 1，就会忽略 SLEEP 指令并且不会进入 SEC_IDLE 模式。如果使能了 Timer1 振荡器，但它尚未运行，外设时钟将会延时直到该振荡器起振。在这种情况下，最初的振荡器运行很不稳定，可能会导致无法预料的结果。

图 4-7： 进入空闲模式的转换时序

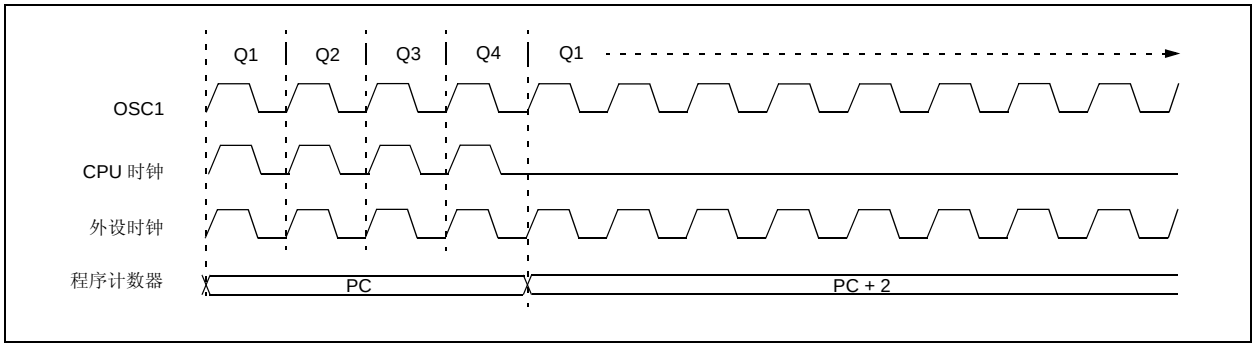
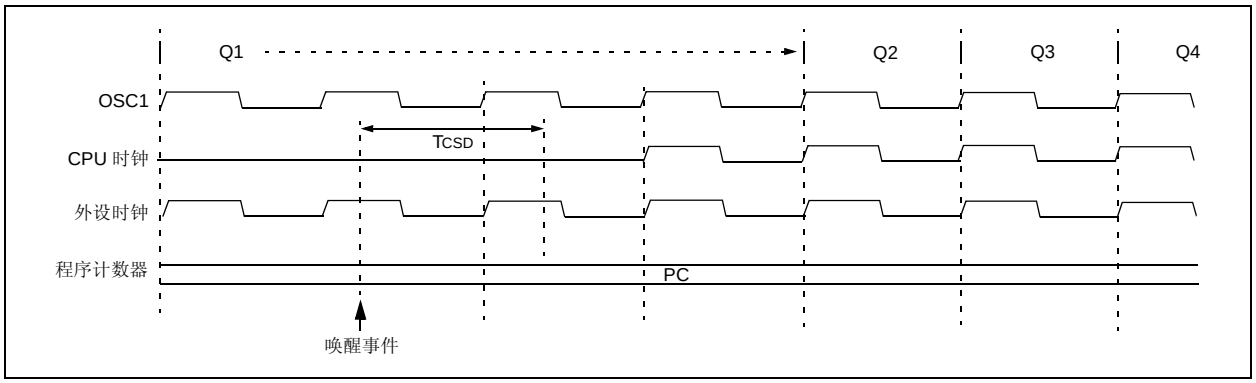


图 4-8： 从空闲模式唤醒进入运行模式的转换时序



4.4.3 RC_IDLE 模式

在 RC_IDLE 模式下，CPU 被禁止，但仍继续由内部振荡器为外设提供时钟。该模式允许在空闲期间对功耗进行控制。

可以通过将IDLEN位置1并执行SLEEP指令从RC_RUN模式进入此模式。如果器件处于另一种运行模式，首先将 IDLEN 置 1，然后清零 SCS 位并执行 SLEEP。当时钟源切换到 INTRC 时，主振荡器被关闭， OSTS 位被清零。

当唤醒事件发生时，外设继续将 INTOSC 作为时钟源。唤醒事件发生后经过一个TCSD延时，CPU使用INTOSC作为时钟源开始执行代码。唤醒不会影响IDLEN和SCS位。如果使能了 WDT 或故障保护时钟监视器，INTOSC 时钟源将继续运行。

4.5 退出空闲和休眠模式

由中断、复位或 WDT 超时触发从休眠模式或任何空闲模式的退出。本节将讨论从功耗管理模式退出的触发方式。在每种功耗管理模式章节中我们已经讨论过其时钟子系统的操作（见**第 4.2 节 “ 运行模式 ”**、**第 4.3 节 “ 休眠模式 ”** 和**第 4.4 节 “ 空闲模式 ”**）。

4.5.1 通过中断退出

任何可用的中断源都可导致器件从空闲模式或休眠模式退出到运行模式。要使能此功能，必须通过将对应 INTCON 或 PIE 寄存器中的中断源允许位置 1 来允许中断源。当相应的中断标志位置 1 时，触发退出操作。

在通过中断从空闲或休眠模式退出时，如果 GIE/GIEH 位 （INTCON<7>）置 1，程序将跳转到中断向量处执行代码。否则，代码将继续执行，不进行跳转 （见**第 9.0 节 “ 中断 ”**）。

唤醒事件之后需要一个固定的 TCSD 间隔的延时，器件才会退出休眠和空闲模式。CPU 需要此延时来准备执行代码。在延时后的第一个时钟周期重新开始执行指令。

4.5.2 通过 WDT 超时退出

根据 WDT 超时发生时器件所处的不同功耗管理模式会引发不同的操作。

如果器件不在执行代码 （所有空闲模式和休眠模式），超时将导致从功耗管理模式退出 （见**第 4.2 节 “ 运行模式 ”** 和**第 4.3 节 “ 休眠模式 ”**）。如果器件正在执行代码（所有运行模式），超时将导致 WDT 复位（见**第 25.2 节 “ 看门狗定时器 （WDT） ”**）。

看门狗定时器和后分频器可由以下任一事件清零：

- 执行 SLEEP 或 CLRWDT 指令
- 当前选定的时钟源失效 （如果使能了故障保护时钟监视器）

4.5.3 通过复位退出

通过复位退出空闲或休眠模式会自动强制器件使用 INTRC 运行。

4.5.4 在没有振荡器起振延时的情况下退出

从某些功耗管理模式退出不需要 OST 延时。有以下两种情形：

- 主时钟源不停止的 PRI_IDLE 模式
- 主时钟源为 EC 或 ECPLL 模式

在这些情况下，主时钟源不需要振荡器起振延时，因为它已经在运行（PRI_IDLE），或者它本来就不需要振荡器起振延时（EC）。但是，当器件退出休眠和空闲模式时，在唤醒事件之后仍然需要一个固定的 TCSD 间隔的延时，以便让 CPU 准备好执行代码。在延时后的第一个时钟周期重新开始执行指令。

注：

5.0 复位

PIC18F87J90 系列器件有以下几种不同类型的复位：

- 上电复位 （Power-on Reset， POR）
- 正常工作期间的 $\overline{\text{MCLR}}$ 复位
- 功耗管理模式下的 $\overline{\text{MCLR}}$ 复位
- 看门狗定时器 （WDT）复位 （执行程序期间）
- 欠压复位 （Brown-out Reset， BOR）
- 配置不匹配（Configuration Mismatch，CM）复位
- RESET 指令
- 堆栈满复位
- 堆栈下溢复位

本节讨论了由 $\overline{\text{MCLR}}$、 POR 和 BOR 产生的复位，并涉及各种起振定时器的工作方式。堆栈复位事件将在**第 6.1.4.4 节 “ 堆栈满和下溢复位 ”** 中讨论。 WDT 复位将在**第 25.2 节 “ 看门狗定时器（WDT） ”** 中讨论。

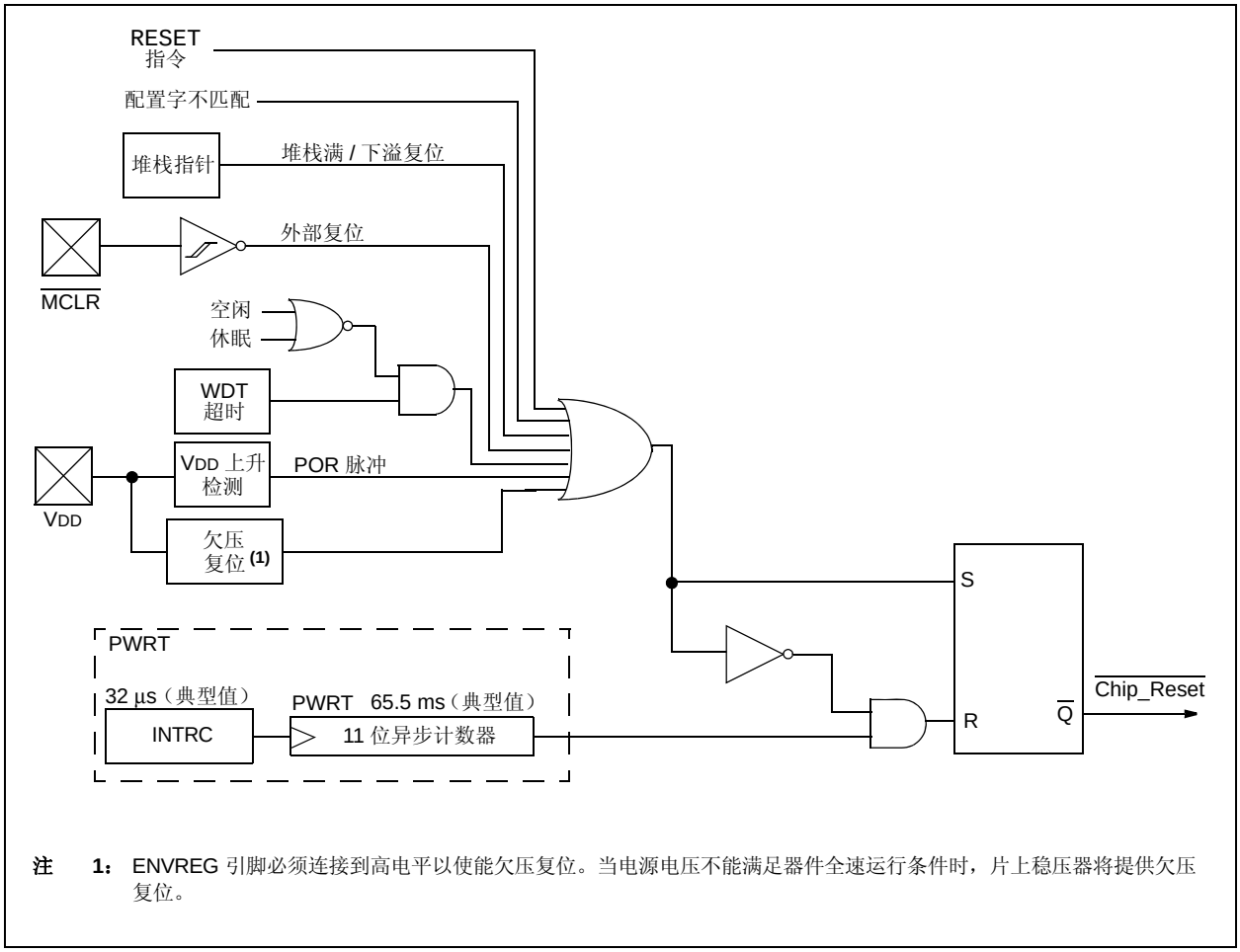
图 5-1 给出了片上复位电路的简化框图。

5.1 RCON 寄存器

通过 RCON 寄存器 （寄存器 5-1）跟踪器件复位事件。该寄存器的低5位表明是否已经发生了特定的复位事件。在大多数情况下，只能通过事件将这些位置 1，而且必须在事件发生后由应用程序将它们清零。需要读取所有这些标志位来确定刚发生的复位的类型。在**第 5.7 节 “ 寄存器的复位状态 ”** 中对此进行了更详细的说明。

RCON 寄存器还有设置中断优先级的控制位（IPEN）。在**第 9.0 节 “ 中断 ”** 中讨论了中断优先级。

图 5-1： 片上复位电路的简化框图



注 1： ENVREG 引脚必须连接到高电平以使能欠压复位。当电源电压不能满足器件全速运行条件时，片上稳压器将提供欠压复位。

寄存器 5-1：　　RCON：复位控制寄存器

R/W-0	U-0	R/W-1	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	$\overline{CM}$	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
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　　**IPEN**：中断优先级使能位
1 = 使能中断优先级
0 = 禁止中断优先级 （PIC16XXXX 兼容模式）

bit 6　　**未实现**：读为 0

bit 5　　**$\overline{CM}$**：配置不匹配标志位
1 = 未发生配置不匹配
0 = 发生了配置不匹配 （发生配置不匹配复位后必须用软件置 1）

bit 4　　**$\overline{RI}$**：RESET 指令标志位
1 = 未执行 RESET 指令 （只能由固件置 1）
0 = 执行了 RESET 指令，导致器件复位 （发生欠压复位后必须用软件置 1）

bit 3　　**$\overline{TO}$**：看门狗超时标志位
1 = 通过上电、 CLRWDT 指令或 SLEEP 指令置 1
0 = 发生了 WDT 超时

bit 2　　**$\overline{PD}$**：掉电检测标志位
1 = 通过上电或 CLRWDT 指令置 1
0 = 通过执行 SLEEP 指令置 1

bit 1　　**$\overline{POR}$**：上电复位状态位
1 = 未发生上电复位 （只能由固件置 1）
0 = 发生了上电复位 （发生上电复位后必须用软件置 1）

bit 0　　**$\overline{BOR}$**：欠压复位状态位
1 = 未发生欠压复位 （只能由固件置 1）
0 = 发生了欠压复位 （发生欠压复位后必须用软件置 1）

注　1： 建议在检测到上电复位后，将 $\overline{POR}$ 位置 1，以便继续检测后续的上电复位。
2： 如果禁止了片上稳压器， $\overline{BOR}$ 则始终保持为 0。更多信息，请参见**第 5.4.1 节 “ 检测 BOR”**。
3： 当$\overline{BOR}$为0并且$\overline{POR}$为1时（假定在上电复位之后立即用软件将$\overline{POR}$设为1），可以说已发生了欠压复位。

5.2 主复位（$\overline{\text{MCLR}}$）

$\overline{\text{MCLR}}$ 引脚提供了触发器件外部硬复位的方法。将该引脚拉低可以产生复位信号。PIC18 扩展型单片机器件在 $\overline{\text{MCLR}}$ 复位路径上有一个噪声滤波器，该滤波器可以检测并滤除小的干扰脉冲。

任何内部复位，包括 WDT 复位，均不能将 $\overline{\text{MCLR}}$ 引脚驱动为低电平。

5.3 上电复位（POR）

只要当 VDD 上升到高于某个门限时，就会在片上产生上电复位条件。这使得 VDD 达到满足器件正常工作的数值时，器件会以初始化状态启动。

为了利用 POR 电路，需要将 $\overline{\text{MCLR}}$ 引脚通过一个电阻（阻值范围为 1 kΩ 到 10 kΩ）连接到 VDD。这样可以省去产生上电复位延时通常所需的外部 RC 元件。VDD 的最小上升速率已指定（参数 D004）。上升速率缓慢的情况，请参见图 5-2。

当器件开始正常工作（即，退出复位状态）时，器件的工作参数（电压、频率和温度等）必须得到满足，以确保其正常工作。如果不满足这些条件，那么器件必须保持在复位状态，直到满足工作条件为止。

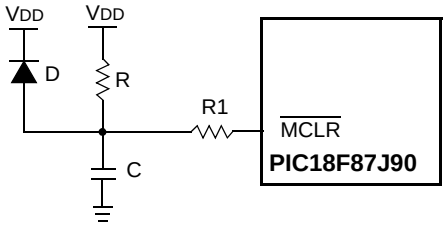
上电复位事件由 $\overline{\text{POR}}$ 位（RCON<1>）捕捉。每当发生上电复位时，该位的状态就会被设为 0；任何其他复位事件均不能改变它。任何硬件事件均不能将 $\overline{\text{POR}}$ 复位为 1。要捕捉多个事件，用户必须在任何上电复位之后用软件手动将该位复位为 1。

5.4 欠压复位（BOR）

当内部稳压器被使能时（ENVREG 引脚连接到 VDD），PIC18F87J90 系列器件就具备了简单的 BOR 功能。当稳压器输出到器件内核的电压接近于器件无法以全速运行的电压时，稳压器将触发欠压复位。在 VDD 上升到稳压器输出电平足够使器件可以全速运行前，BOR 电路保持器件处于复位状态。

一旦发生 BOR，上电延时定时器就将芯片保持在复位状态，持续时间为 TPWRT（参数 33）。如果在上电延时定时器运行过程中，VDD 电压降到全速运行的门限值以下，芯片将重新回到欠压复位状态并且初始化上电延时定时器。一旦 VDD 电压上升到稳压器输出足够全速运行时，上电延时定时器将重新执行延时。

图 5-2：　外部上电复位电路（VDD 缓慢上电的情况）



注　1： 仅当 VDD 上电速率过慢时才需要外部上电复位电路。二极管 D 有助于在 VDD 掉电时使电容迅速放电。

2： 建议 R < 40 kΩ，确保电阻 R 两端压降符合器件的电气规范。

3： R1 ≥ 1 kΩ 将限制任何电流从外部电容 C 流入 MCLR，以避免由于静电放电（Electrostatic Discharge，ESD）或电过载（Electrical Overstress，EOS）导致MCLR/VPP引脚损坏。

5.4.1 检测 BOR

在发生欠压复位或上电复位事件时，$\overline{\text{BOR}}$ 位总是复位为 0。因此只通过读 $\overline{\text{BOR}}$ 位的状态很难确定是否发生了欠压复位事件。更可靠的方法是同时检查 $\overline{\text{POR}}$ 和 $\overline{\text{BOR}}$ 的状态。假定在发生任何上电复位事件后，$\overline{\text{POR}}$ 位被立即用软件复位为 1。如果 $\overline{\text{BOR}}$ 为 0 同时 $\overline{\text{POR}}$ 为 1，那么就可以断定已经发生了欠压复位事件。

如果禁止稳压器，也会禁止欠压复位功能。在这种情况下，不能使用 $\overline{\text{BOR}}$ 位来确定欠压复位事件。上电复位事件仍会将 $\overline{\text{BOR}}$ 位清零。

5.5 配置不匹配（CM）

配置不匹配（CM）复位旨在用于检测随机存储器损坏事件，并尝试从事件中恢复。这些事件包括静电放电（ESD）事件，这种事件会导致整个器件中大范围的单个位单元内容改变，并导致灾难性的故障。

在 PIC18FXXJ 闪存器件中，在工作期间会持续监视器件配置寄存器（位于配置存储空间中），即将它们的值与互补的影子寄存器进行比较。如果在两组寄存器之间检测到不匹配，则会自动发生 CM 复位。这些事件由 CM 位（RCON<5>）捕捉。每当发生 CM 事件时，该位的状态就会被设为 0。任何其他复位事件均不能改变它。

5.6 上电延时定时器 （PWRT）

PIC18F87J90 系列器件具有片上上电延时定时器（PWRT）以帮助稳定上电复位过程。PWRT 总是使能的。其主要功能是确保在代码执行之前，器件的电压是稳定的。

PIC18F87J90 系列器件的上电延时定时器（PWRT）是一个 11 位计数器，它使用 INTRC 时钟源作为时钟输入。该定时器可产生大约 2048 x 32 µs = 65.6 ms 的时间间隔。 PWRT 计数期间，器件保持在复位状态。

上电延时时间取决于 INTRC 时钟，并且由于温度和工艺的不同，不同器件的上电延时时间也将各不相同。详情请参见直流参数 33。

5.6.1 延时时序

如果使能了 PWRT，则在 POR 脉冲被清零后，启动 PWRT 延时。总延迟时间将取决于 PWRT 的状态。图 5-3、图 5-4、图 5-5 和图 5-6 说明了在使能上电延时定时器时的延时时序。

由于延时是由 POR 脉冲触发的，因此如果 $\overline{\text{MCLR}}$ 保持足够长时间的低电平， PWRT 将结束。将 $\overline{\text{MCLR}}$ 电平拉高后器件将立即开始执行程序 （图 5-5）。这对于测试或同步多个并行工作的 PIC18FXXXX 器件来说是非常有用的。

图 5-3： 上电延时时序 （$\overline{\text{MCLR}}$ 连接到 VDD， VDD 电压上升时间 < TPWRT）

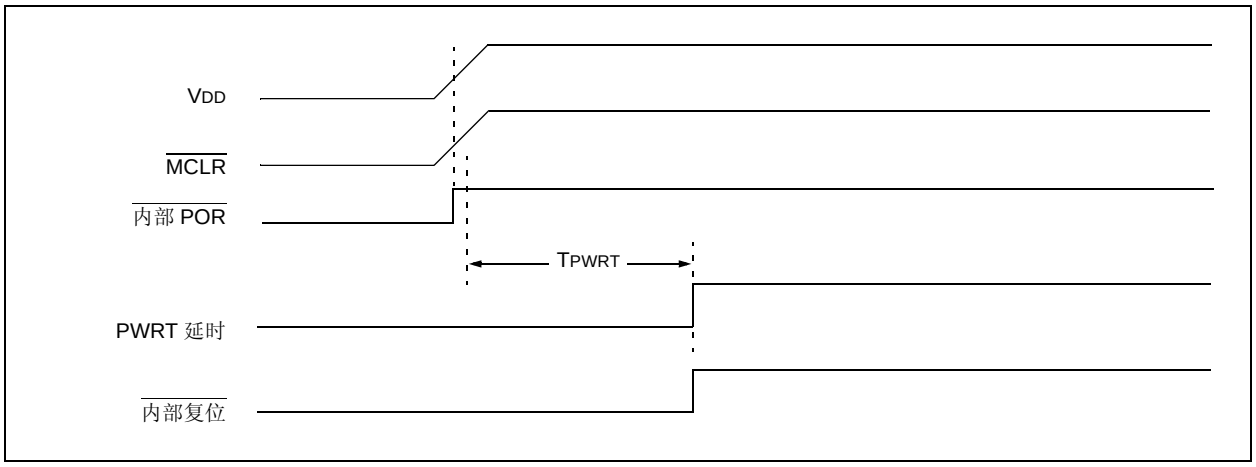


图 5-4： 上电延时时序 （$\overline{\text{MCLR}}$ 未连接到 VDD）：情形 1

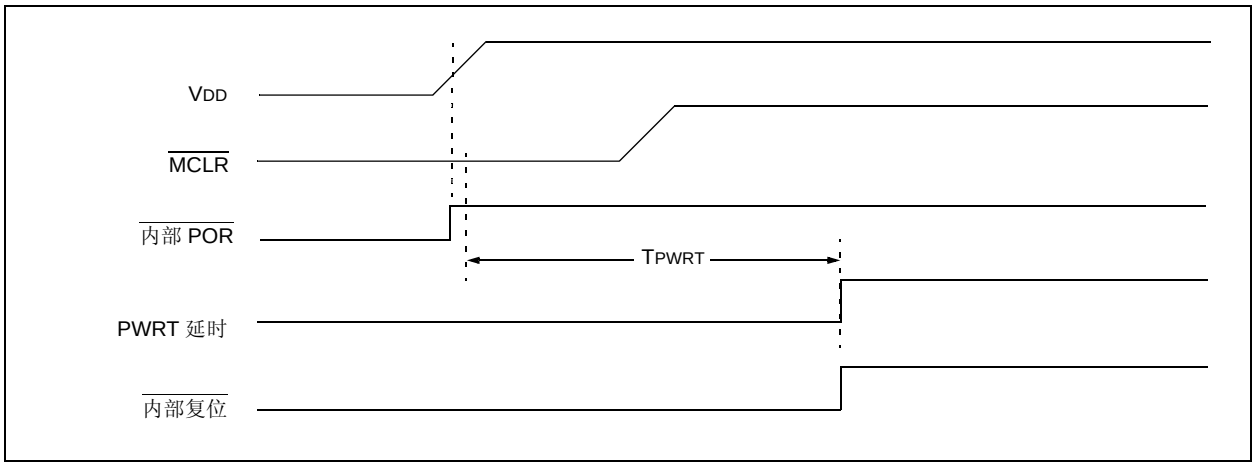


图 5-5：　　上电延时时序（$\overline{\text{MCLR}}$ 未连接到 VDD）：情形 2

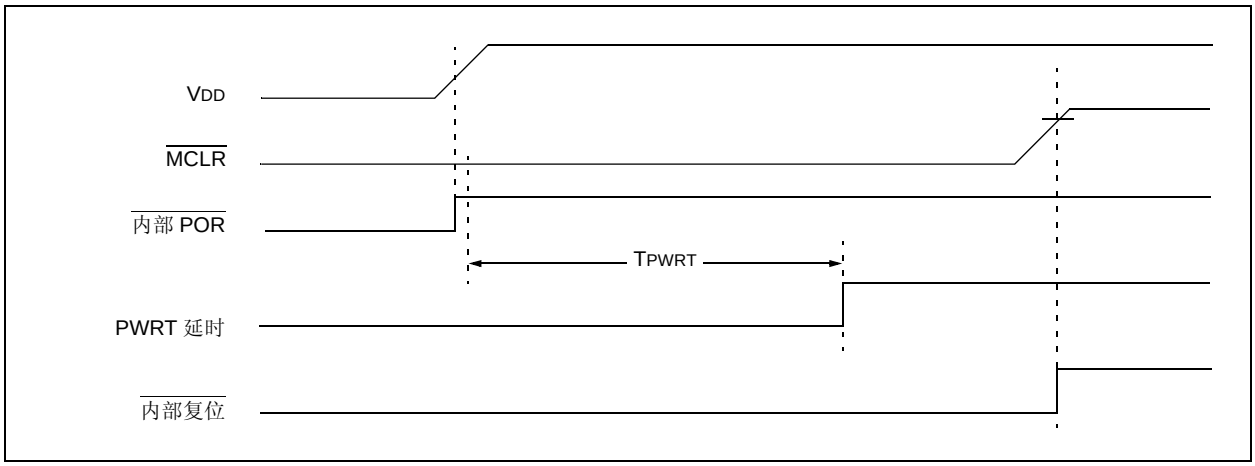
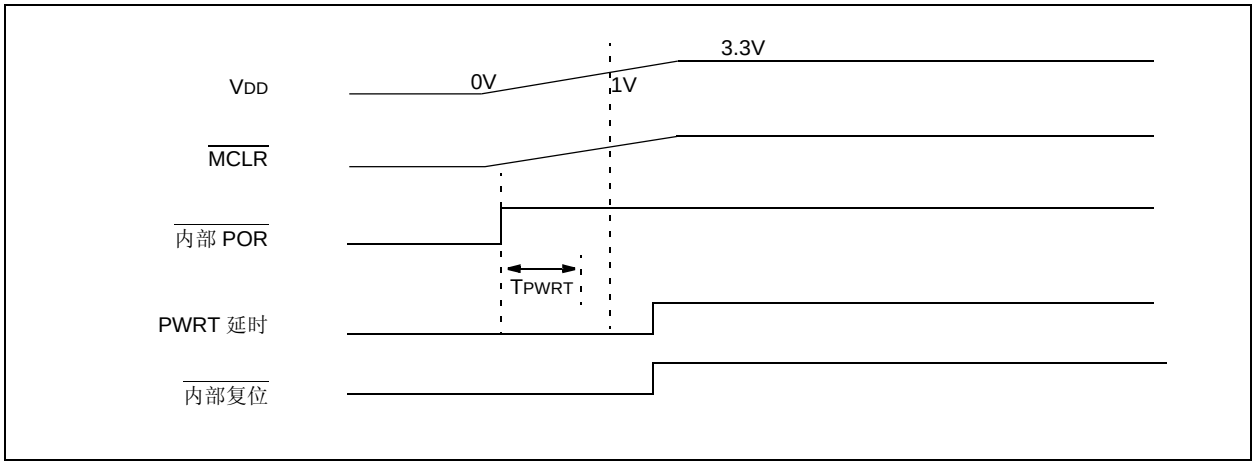


图 5-6：　　缓慢上升时间（$\overline{\text{MCLR}}$ 连接到 VDD，VDD 电压上升时间 > TPWRT）



5.7 寄存器的复位状态

大多数寄存器不受复位的影响。在 POR 时这些寄存器的状态不确定，而在其他复位时它们的状态不变。而所有其他寄存器则根据不同的复位类型被强制为 “ 复位状态 ”。

大多数寄存器不受 WDT 唤醒的影响，因为这被视为是正常工作的恢复。如表 5-1 所示，RCON 寄存器中的状态位 （RI、TO、PD、POR 和 BOR）在不同的复位情形下会分别被置 1 或清零。可在软件中使用这些位判断复位的性质。

表 5-2 描述了所有特殊功能寄存器的复位状态。可以将这些复位状态分类为上电和欠压复位、主复位、WDT 复位以及 WDT 唤醒。

表 5-1： RCON 寄存器的状态位、含义以及初始化状态

条件	程序计数器 (1)	RCON 寄存器					STKPTR 寄存器	
		RI	TO	PD	POR	BOR	STKFUL	STKUNF
上电复位	0000h	1	1	1	0	0	0	0
RESET 指令	0000h	0	u	u	u	u	u	u
欠压复位	0000h	1	1	1	u	0	u	u
功耗管理运行模式下的 MCLR 复位	0000h	u	1	u	u	u	u	u
功耗管理空闲模式和休眠模式下的 MCLR 复位	0000h	u	1	0	u	u	u	u
全功耗或功耗管理运行模式下的 WDT 超时	0000h	u	0	u	u	u	u	u
全功耗执行期间的 MCLR 复位	0000h	u	u	u	u	u	u	u
堆栈满复位 （STVREN = 1）	0000h	u	u	u	u	u	1	u
堆栈下溢复位 （STVREN = 1）	0000h	u	u	u	u	u	u	1
堆栈下溢错误 （不是真正的复位， STVREN = 0）	0000h	u	u	u	u	u	u	1
功耗管理空闲模式或休眠模式下的 WDT 超时	PC + 2	u	0	0	u	u	u	u
通过中断从功耗管理模式退出	PC + 2	u	u	0	u	u	u	u

图注： u = 不变

注 1： 当器件被中断唤醒且 GIEH 或 GIEL 位被置 1 时， PC 装入中断向量 （0008h 或 0018h）。

表 5-2： 所有寄存器的初始化状态

寄存器	适用器件		上电复位， 欠压复位	MCLR 复位 WDT 复位 RESET 指令 堆栈复位	通过 WDT 或 中断唤醒器件
TOSU	PIC18F6XJ90	PIC18F8XJ90	---0 0000	---0 0000	---0 uuuu(1)
TOSH	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu(1)
TOSL	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu(1)
STKPTR	PIC18F6XJ90	PIC18F8XJ90	uu-0 0000	00-0 0000	uu-u uuuu(1)
PCLATU	PIC18F6XJ90	PIC18F8XJ90	---0 0000	---0 0000	---u uuuu
PCLATH	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
PCL	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	PC + 2(2)
TBLPTRU	PIC18F6XJ90	PIC18F8XJ90	--00 0000	--00 0000	--uu uuuu
TBLPTRH	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
TBLPTRL	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
TABLAT	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
PRODH	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODL	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
INTCON	PIC18F6XJ90	PIC18F8XJ90	0000 000x	0000 000u	uuuu uuuu(3)
INTCON2	PIC18F6XJ90	PIC18F8XJ90	1111 1111	1111 1111	uuuu uuuu(3)
INTCON3	PIC18F6XJ90	PIC18F8XJ90	1100 0000	1100 0000	uuuu uuuu(3)
INDF0	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A
POSTINC0	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A
POSTDEC0	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A
PREINC0	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A
PLUSW0	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A
FSR0H	PIC18F6XJ90	PIC18F8XJ90	---- xxxx	---- uuuu	---- uuuu
FSR0L	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF1	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A
POSTINC1	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A
POSTDEC1	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A
PREINC1	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A
PLUSW1	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A

图注： u = 不变， x = 未知， - = 未实现位 （读为 0）， q = 值取决于具体条件。
阴影单元表示不适用于指定器件的状态。

注 1： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时，用 PC 的当前值更新 TOSU、TOSH 和 TOSL。将 STKPTR 修改为指向硬件堆栈的下一个存储单元。

2： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时， PC 装入中断向量 （0008h 或 0018h）。

3： INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。

4： 具体条件下的复位值，请参见表 5-1。

5： 根据所选的振荡器模式使能 PORTA、LATA 和 TRISA 中的 bit 6 和 bit 7。如果未被使能为 PORTA 引脚，则它们将被禁止并读为 0。

表 5-2： 所有寄存器的初始化状态 （续）

寄存器	适用器件		上电复位， 欠压复位	$\overline{\text{MCLR}}$ 复位 WDT 复位 RESET 指令 堆栈复位	通过 WDT 或 中断唤醒器件
FSR1H	PIC18F6XJ90	PIC18F8XJ90	---- xxxx	---- uuuu	---- uuuu
FSR1L	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
BSR	PIC18F6XJ90	PIC18F8XJ90	---- 0000	---- 0000	---- uuuu
INDF2	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A
POSTINC2	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A
POSTDEC2	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A
PREINC2	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A
PLUSW2	PIC18F6XJ90	PIC18F8XJ90	N/A	N/A	N/A
FSR2H	PIC18F6XJ90	PIC18F8XJ90	---- xxxx	---- uuuu	---- uuuu
FSR2L	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	PIC18F6XJ90	PIC18F8XJ90	---x xxxx	---u uuuu	---u uuuu
TMR0H	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
TMR0L	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	PIC18F6XJ90	PIC18F8XJ90	1111 1111	1111 1111	uuuu uuuu
OSCCON	PIC18F6XJ90	PIC18F8XJ90	0110 q000	0110 q000	uuuu quuu
LCDREG	PIC18F6XJ90	PIC18F8XJ90	-011 1100	-011 1000	-uuu uuuu
WDTCON	PIC18F6XJ90	PIC18F8XJ90	0--- ---0	0--- ---0	u--- ---u
RCON(4)	PIC18F6XJ90	PIC18F8XJ90	0-11 11q0	0-0q qquu	u-uu qquu
TMR1H	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	PIC18F6XJ90	PIC18F8XJ90	0000 0000	u0uu uuuu	uuuu uuuu
TMR2	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
PR2	PIC18F6XJ90	PIC18F8XJ90	1111 1111	1111 1111	1111 1111
T2CON	PIC18F6XJ90	PIC18F8XJ90	-000 0000	-000 0000	-uuu uuuu
SSPBUF	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPADD	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
SSPCON1	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
SSPCON2	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu

图注： u = 不变， x = 未知， - = 未实现位 （读为 0）， q = 值取决于具体条件。
阴影单元表示不适用于指定器件的状态。

注 1： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时，用 PC 的当前值更新 TOSU、TOSH 和 TOSL。将 STKPTR 修改为指向硬件堆栈的下一个存储单元。

2： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时， PC 装入中断向量 （0008h 或 0018h）。

3： INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。

4： 具体条件下的复位值，请参见表 5-1。

5： 根据所选的振荡器模式使能 PORTA、LATA 和 TRISA 中的 bit 6 和 bit 7。如果未被使能为 PORTA 引脚，则它们将被禁止并读为 0。

表 5-2： 所有寄存器的初始化状态 （续）

寄存器	适用器件		上电复位， 欠压复位	MCLR 复位 WDT 复位 RESET 指令 堆栈复位	通过 WDT 或 中断唤醒器件
ADRESH	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	PIC18F6XJ90	PIC18F8XJ90	0-00 0000	0-00 0000	u-uu uuuu
ADCON1	PIC18F6XJ90	PIC18F8XJ90	0-00 0000	0-00 0000	u-uu uuuu
ADCON2	PIC18F6XJ90	PIC18F8XJ90	0-00 0000	0-00 0000	u-uu uuuu
LCDDATA4	PIC18F6XJ90	PIC18F8XJ90	---- ---x	---- ---u	---- ---u
LCDDATA4	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA3	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA2	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA1	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA0	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDSE5	PIC18F6XJ90	PIC18F8XJ90	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE4	PIC18F6XJ90	PIC18F8XJ90	---- ---0	---- ---u	---- ---u
LCDSE4	PIC18F6XJ90	PIC18F8XJ90	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE3	PIC18F6XJ90	PIC18F8XJ90	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE2	PIC18F6XJ90	PIC18F8XJ90	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE1	PIC18F6XJ90	PIC18F8XJ90	0000 0000	uuuu uuuu	uuuu uuuu
CVRCON	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
CMCON	PIC18F6XJ90	PIC18F8XJ90	0000 0111	0000 0111	uuuu uuuu
TMR3H	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
T3CON	PIC18F6XJ90	PIC18F8XJ90	0000 0000	uuuu uuuu	uuuu uuuu
SPBRG1	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
RCREG1	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
TXREG1	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
TXSTA1	PIC18F6XJ90	PIC18F8XJ90	0000 0010	0000 0010	uuuu uuuu
RCSTA1	PIC18F6XJ90	PIC18F8XJ90	0000 000x	0000 000x	uuuu uuuu
LCDPS	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
LCDSE0	PIC18F6XJ90	PIC18F8XJ90	0000 0000	uuuu uuuu	uuuu uuuu
LCDCON	PIC18F6XJ90	PIC18F8XJ90	000- 0000	000- 0000	uuu- uuuu
EECON2	PIC18F6XJ90	PIC18F8XJ90	---- ----	---- ----	---- ----
EECON1	PIC18F6XJ90	PIC18F8XJ90	---0 x00-	---0 u00-	---0 u00-

图注： u = 不变， x = 未知， - = 未实现位 （读为 0）， q = 值取决于具体条件。
阴影单元表示不适用于指定器件的状态。

注 1： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时，用 PC 的当前值更新 TOSU、TOSH 和 TOSL。将 STKPTR 修改为指向硬件堆栈的下一个存储单元。

2： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时， PC 装入中断向量 （0008h 或 0018h）。

3： INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。

4： 具体条件下的复位值，请参见表 5-1。

5： 根据所选的振荡器模式使能 PORTA、LATA 和 TRISA 中的 bit 6 和 bit 7。如果未被使能为 PORTA 引脚，则它们将被禁止并读为 0。

表 5-2： 所有寄存器的初始化状态 （续）

寄存器	适用器件		上电复位，欠压复位	MCLR 复位 WDT 复位 RESET 指令 堆栈复位	通过 WDT 或中断唤醒器件
IPR3	PIC18F6XJ90	PIC18F8XJ90	-111 1111	-111 1111	-uuu 1111
PIR3	PIC18F6XJ90	PIC18F8XJ90	-000 0000	-000 0000	-uuu 0000(3)
PIE3	PIC18F6XJ90	PIC18F8XJ90	-000 0000	-000 0000	-uuu 0000
IPR2	PIC18F6XJ90	PIC18F8XJ90	11-- 111-	11-- 111-	uu-- uuu-
PIR2	PIC18F6XJ90	PIC18F8XJ90	00-- 000-	00-- 000-	uu-- uuu-(3)
PIE2	PIC18F6XJ90	PIC18F8XJ90	00-- 000-	00-- 000-	uu-- uuu-
IPR1	PIC18F6XJ90	PIC18F8XJ90	-111 1-11	-111 1-11	-uuu u-uu
PIR1	PIC18F6XJ90	PIC18F8XJ90	-000 0-00	-000 0-00	-uuu u-uu(3)
PIE1	PIC18F6XJ90	PIC18F8XJ90	-000 0-00	-000 0-00	-uuu u-uu
OSCTUNE	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
TRISJ	PIC18F6XJ90	PIC18F8XJ90	1111 1111	1111 1111	uuuu uuuu
TRISH	PIC18F6XJ90	PIC18F8XJ90	1111 1111	1111 1111	uuuu uuuu
TRISG	PIC18F6XJ90	PIC18F8XJ90	0001 1111	0001 1111	uuuu uuuu
TRISF	PIC18F6XJ90	PIC18F8XJ90	1111 111-	1111 111-	uuuu uuu-
TRISE	PIC18F6XJ90	PIC18F8XJ90	1111 1-11	1111 1-11	uuuu u-uu
TRISD	PIC18F6XJ90	PIC18F8XJ90	1111 1111	1111 1111	uuuu uuuu
TRISC	PIC18F6XJ90	PIC18F8XJ90	1111 1111	1111 1111	uuuu uuuu
TRISB	PIC18F6XJ90	PIC18F8XJ90	1111 1111	1111 1111	uuuu uuuu
TRISA(5)	PIC18F6XJ90	PIC18F8XJ90	1111 1111(5)	1111 1111(5)	uuuu uuuu(5)
LATJ	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATH	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATG	PIC18F6XJ90	PIC18F8XJ90	00-x xxxx	00-u uuuu	uu-u uuuu
LATF	PIC18F6XJ90	PIC18F8XJ90	xxxx xxx-	uuuu uuu-	uuuu uuu-
LATE	PIC18F6XJ90	PIC18F8XJ90	xxxx x-xx	uuuu u-uu	uuuu u-uu
LATD	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATC	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATB	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATA(5)	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx(5)	uuuu uuuu(5)	uuuu uuuu(5)
PORTJ	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTH	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTG	PIC18F6XJ90	PIC18F8XJ90	000x xxxx	000u uuuu	000u uuuu
PORTF	PIC18F6XJ90	PIC18F8XJ90	xxxx xxx-	uuuu uuu-	uuuu uuu-

图注： u = 不变， x = 未知， - = 未实现位 （读为 0）， q = 值取决于具体条件。
阴影单元表示不适用于指定器件的状态。

注 1： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时，用 PC 的当前值更新 TOSU、TOSH 和 TOSL。将 STKPTR 修改为指向硬件堆栈的下一个存储单元。

2： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时， PC 装入中断向量 （0008h 或 0018h）。

3： INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。

4： 具体条件下的复位值，请参见表 5-1。

5： 根据所选的振荡器模式使能 PORTA、LATA 和 TRISA 中的 bit 6 和 bit 7。如果未被使能为 PORTA 引脚，则它们将被禁止并读为 0。

表 5-2： 所有寄存器的初始化状态 （续）

寄存器	适用器件		上电复位， 欠压复位	MCLR 复位 WDT 复位 RESET 指令 堆栈复位	通过 WDT 或 中断唤醒器件
PORTE	PIC18F6XJ90	PIC18F8XJ90	xxxx x-xx	uuuu u-uu	uuuu u-uu
PORTD	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTB	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA(5)	PIC18F6XJ90	PIC18F8XJ90	xx0x 0000(5)	uu0u 0000(5)	uuuu uuuu(5)
SPBRGH1	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
BAUDCON1	PIC18F6XJ90	PIC18F8XJ90	0100 0-00	0100 0-00	uuuu u-uu
LCDDATA23	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA22	PIC18F6XJ90	PIC18F8XJ90	---- ---x	---- ---u	---- ---u
LCDDATA22	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA21	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA20	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA19	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA18	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA17	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA16	PIC18F6XJ90	PIC18F8XJ90	---- ---x	---- ---u	---- ---u
LCDDATA16	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA15	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA14	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA13	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA12	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA11	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA10	PIC18F6XJ90	PIC18F8XJ90	---- ---x	---- ---u	---- ---u
LCDDATA10	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA9	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA8	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA7	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA6	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
LCDDATA5	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1H	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	PIC18F6XJ90	PIC18F8XJ90	--00 0000	--00 0000	--uu uuuu
CCPR2H	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu

图注： u = 不变， x = 未知， - = 未实现位 （读为 0）， q = 值取决于具体条件。
阴影单元表示不适用于指定器件的状态。

注 1： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时，用 PC 的当前值更新 TOSU、TOSH 和 TOSL。将 STKPTR 修改为指向硬件堆栈的下一个存储单元。

2： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时， PC 装入中断向量 （0008h 或 0018h）。

3： INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。

4： 具体条件下的复位值，请参见表 5-1。

5： 根据所选的振荡器模式使能 PORTA、LATA 和 TRISA 中的 bit 6 和 bit 7。如果未被使能为 PORTA 引脚，则它们将被禁止并读为 0。

表 5-2： 所有寄存器的初始化状态 （续）

寄存器	适用器件		上电复位， 欠压复位	MCLR 复位 WDT 复位 RESET 指令 堆栈复位	通过 WDT 或 中断唤醒器件
CCPR2L	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP2CON	PIC18F6XJ90	PIC18F8XJ90	--00 0000	--00 0000	--uu uuuu
SPBRG2	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
RCREG2	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
TXREG2	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
TXSTA2	PIC18F6XJ90	PIC18F8XJ90	0000 -010	0000 -010	uuuu -uuu
RCSTA2	PIC18F6XJ90	PIC18F8XJ90	0000 000x	0000 000x	uuuu uuuu
RTCCFG	PIC18F6XJ90	PIC18F8XJ90	0-00 0000	0-00 0000	u-uu uuuu
RTCCAL	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
RTCVALH	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
RTCVALL	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
ALRMCFG	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
ALRMRPT	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
ALRMVALH	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
ALRMVALL	PIC18F6XJ90	PIC18F8XJ90	xxxx xxxx	uuuu uuuu	uuuu uuuu
CTMUCONH	PIC18F6XJ90	PIC18F8XJ90	0-00 0000	0-00 0000	u-uu uuuu
CTMUCONL	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
CTMUICON	PIC18F6XJ90	PIC18F8XJ90	0000 0000	0000 0000	uuuu uuuu
PADCFG1	PIC18F6XJ90	PIC18F8XJ90	---- -00-	---- -00-	---- -uu-

图注： u = 不变， x = 未知， - = 未实现位 （读为 0）， q = 值取决于具体条件。
阴影单元表示不适用于指定器件的状态。

注 1： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时，用 PC 的当前值更新 TOSU、TOSH 和 TOSL。将 STKPTR 修改为指向硬件堆栈的下一个存储单元。

2： 当器件被中断唤醒且 GIEL 或 GIEH 位被置 1 时， PC 装入中断向量 （0008h 或 0018h）。

3： INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。

4： 具体条件下的复位值，请参见表 5-1。

5： 根据所选的振荡器模式使能 PORTA、LATA 和 TRISA 中的 bit 6 和 bit 7。如果未被使能为 PORTA 引脚，则它们将被禁止并读为 0。

6.0 存储器构成

PIC18 闪存单片机器件有两种类型的存储器：

- 程序存储器
- 数据 RAM

由于是哈佛架构的器件，数据和程序存储器使用不同的总线，因而可同时访问这两种存储空间。

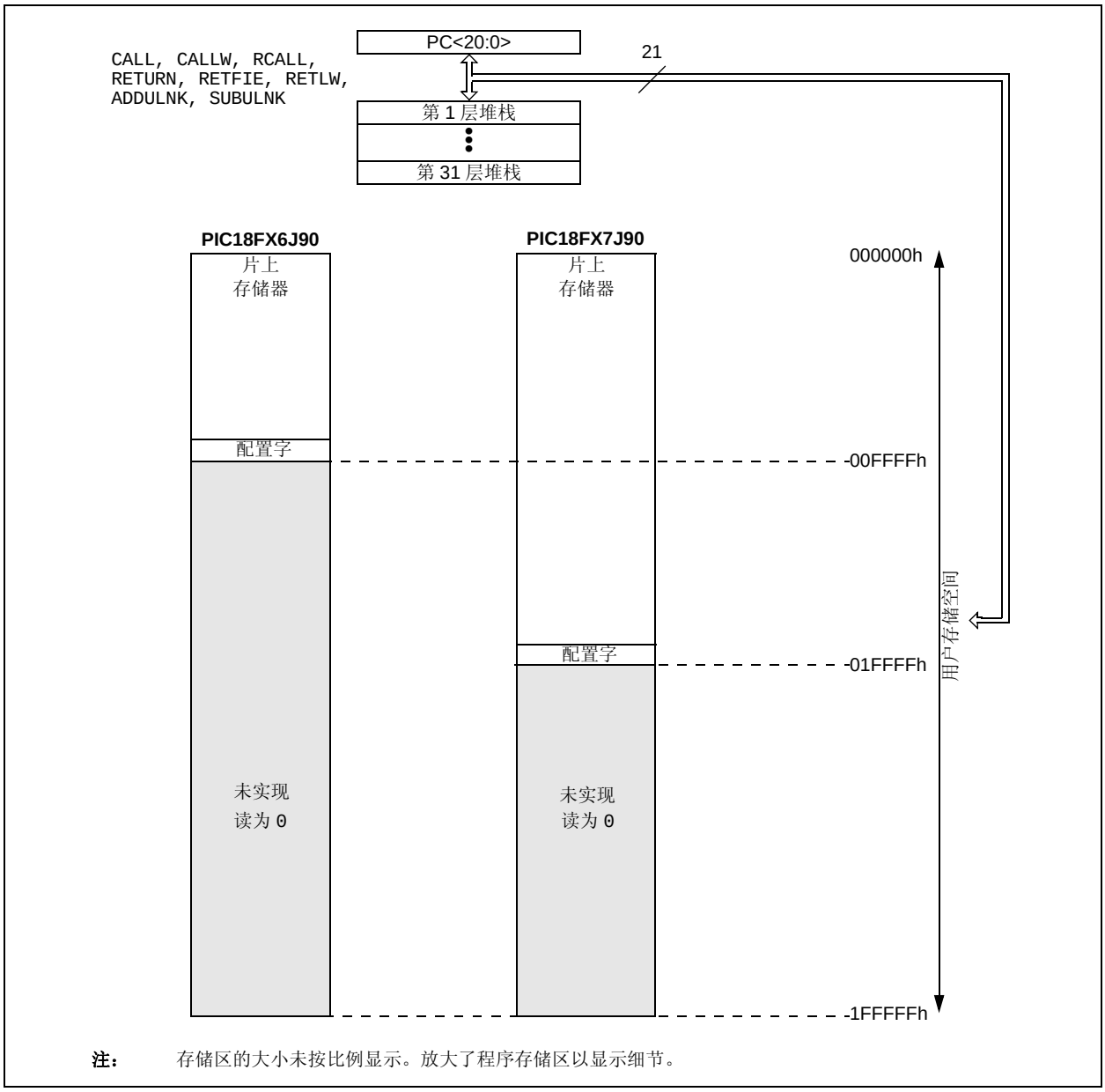
第 7.0 节 “ 闪存程序存储器 ” 提供了关于闪存程序存储器操作的更多详细信息。

6.1 程序存储器构成

PIC18 单片机具有 21 位程序计数器，可以对 2 MB 的程序存储空间进行寻址。访问物理实现存储器的上边界和 2 MB 地址之间的存储单元会返回全 0（NOP 指令）。

整个 PIC18F87J90 系列提供了各种容量的片上闪存程序存储器，从 64 KB（最多 16,384 条单字指令）到 128 KB（65,536 条单字指令）。图 6-1 给出了本系列中各器件的程序存储器映射。

图 6-1： PIC18F87J90 系列器件的存储器映射



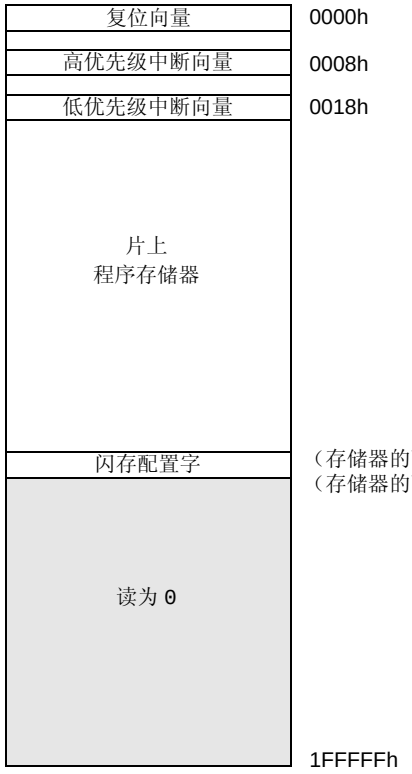
注： 存储区的大小未按比例显示。放大了程序存储区以显示细节。

6.1.1 存储器硬编码向量

所有的 PIC18 器件在它们的程序存储空间内共有 3 个硬编码的返回向量。复位向量地址是在器件发生任何复位时程序计数器返回到的默认值；它位于 0000h。

PIC18 器件还有两个中断向量地址，用于处理高优先级和低优先级中断。高优先级中断向量位于 0008h，低优先级中断向量位于 0018h。它们在程序存储器映射中的相对位置如图 6-2 所示。

图 6-2： PIC18F87J90 系列器件的硬编码向量和配置字单元



图注： （存储器的顶部）代表片上程序存储空间的上边界（关于特定器件的具体值，请参见图 6-1）。阴影区域表示未实现的存储区。存储区未按比例显示。

6.1.2 闪存配置字

由于 PIC18F87J90 系列器件没有固定的配置存储器，所以保留片上程序存储器顶部的 4 个字来保存配置信息。复位时，该配置信息被复制到配置寄存器。

配置字按数字编号顺序存储在程序存储单元中，从最低地址开始存放 CONFIG1 的低字节，到 CONFIG4 的高字节结束。对于这些器件，只使用从 CONFIG1 到 CONFIG3 的配置字；保留 CONFIG4。PIC18F87J90 系列器件的闪存配置字的实际地址如表 6-1 所示。图 6-2 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。

第 25.1 节 “ 配置位 ” 中提供了有关器件配置字的更多详细信息。

表 6-1： PIC18F87J90 系列器件的闪存配置字

器件	程序存储器（KB）	配置字地址
PIC18F66J90	64	FFF8h 到 FFFFh
PIC18F86J90		
PIC18F67J90	128	1FFF8h 到 1FFFFh
PIC18F87J90		

6.1.3 程序计数器

程序计数器（Program Counter，PC）指定欲取出执行的指令的地址。PC 为 21 位宽，保存在三个不同的 8 位寄存器中。存储低字节的寄存器称为 PCL 寄存器，该寄存器可读写。存储高字节的寄存器，即 PCH 寄存器，存储 PC<15:8> 位；该寄存器不可直接读写。更新 PCH 寄存器的操作是通过 PCLATH 寄存器实现的。存储最高字节的寄存器称为 PCU。该寄存器存储 PC<20:16> 位；它也不能直接读写。更新 PCU 寄存器的操作是通过 PCLATU 寄存器实现的。

PCLATH 和 PCLATU 的内容通过执行写 PCL 的任何操作被传送到程序计数器。同样，程序计数器的两个高字节通过读 PCL 的操作被传送到 PCLATH 和 PCLATU。这对于 PC 的计算偏移量很有用处（见**第 6.1.6.1 节 “ 计算 GOTO”**）。

PC 是按字节寻址程序存储器的。为了防止 PC 不能与字指令对齐，需要将 PCL 的最低有效位固定取值为 0。PC 每次加 2 来寻址程序存储器中的顺序指令。

CALL、RCALL、GOTO 和程序跳转指令直接写入程序计数器。对于这些指令，PCLATH 和 PCLATU 的内容不会传送到程序计数器。

6.1.4 返回地址堆栈

返回地址堆栈允许最多 31 个程序调用和中断的任意组合。当执行 CALL、 RCALL 指令或响应中断时， PC 值会被压入该堆栈。而在执行 RETURN, RETLW 或 RETFIE 指令（如果使能了扩展指令集，则还包括 ADDULNK 和 SUBULNK 指令）时，PC 值会从堆栈弹出。PCLATU 和 PCLATH 不受 RETURN 或 CALL 指令的影响。

通过 21 位的 RAM 和一个 5 位的堆栈指针 STKPTR 来实现 31 字的堆栈操作。堆栈既不占用程序存储空间，也不占用数据存储空间。堆栈指针是可读写的，并且通过栈顶的特殊功能寄存器可以读写栈顶地址。也可以使用这些寄存器将数据压入堆栈或者从堆栈弹出。

执行 CALL 类型的指令时，产生进栈操作：首先堆栈指针加 1，并且将 PC（PC 已经指向 CALL 后下一条指令）的内容写入堆栈指针所指向的地址单元。执行 RETURN 类型的指令时，产生出栈操作：STKPTR 所指向的地址单元的内容会被传送给 PC，然后堆栈指针减 1。

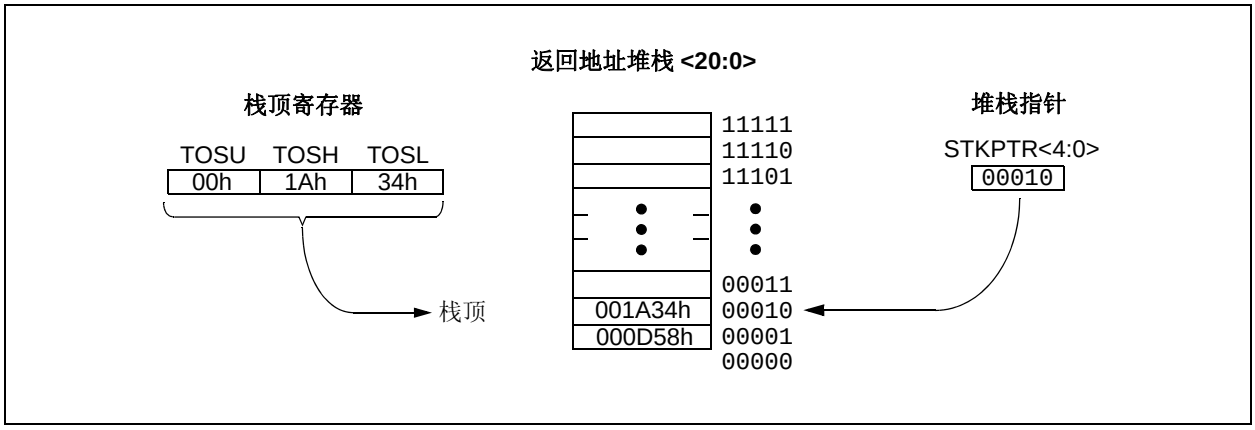
所有复位后，堆栈指针均会初始化为 00000。堆栈指针值 00000 不指向任何 RAM 单元；它仅仅是一个复位值。状态位表明堆栈是已满、上溢还是下溢。

6.1.4.1 访问栈顶

只可读写返回地址堆栈的栈顶（Top-of-Stack， TOS）。有三个寄存器 TOSU:TOSH:TOSL 用于保存由 STKPTR 寄存器所指向的堆栈单元的内容（图 6-3）。这可以让用户在必要时实现软件堆栈。在 CALL、RCALL 或响应中断（如果使能了扩展指令集，则还包括ADDULNK和SUBULNK 指令）后，软件可以通过读取 TOSU:TOSH:TOSL 寄存器来读取压入堆栈的值。这些值可以被存放到由用户定义的软件堆栈。返回时，软件将这些值存回 TOSU:TOSH:TOSL 并执行返回。

为防止意外的堆栈操作，访问堆栈时用户必须禁止全局中断允许位。

图 6-3： 返回地址堆栈和相关的寄存器



6.1.4.2 返回堆栈指针 （STKPTR）

STKPTR 寄存器（寄存器 6-1）包含堆栈指针值、STKFUL（堆栈满）状态位和 STKUNF（堆栈下溢）状态位。堆栈指针值可为 0 到 31 范围内的值。向堆栈压入值前，堆栈指针加 1；而从堆栈弹出值后，堆栈指针减 1。复位时，堆栈指针值为零。用户可以读写堆栈指针的值。实时操作系统（Real-Time Operating System，RTOS）可以利用此特性对返回堆栈进行维护。

向堆栈压入 PC 值 31 次（且没有值从堆栈弹出）后，STKFUL 位置 1。通过软件或 POR 将 STKFUL 位清零。

由 STVREN （堆栈溢出复位使能）配置位的状态决定堆栈满时将执行的操作。（关于器件配置位的说明，请参见 **第 25.1 节 “ 配置位 ”**。）如果 STVREN 位置 1（默认），第 31 次压栈将把（PC + 2）值压入堆栈，从而将 STKFUL 位置 1 并复位器件。STKFUL 位将保持置 1，而堆栈指针将被清零。

如果 STVREN 清零，第 31 次压栈时 STKFUL 位将会置 1，堆栈指针值递增到 31。后续的压栈操作都不会覆盖第 31 次进栈的值，并且 STKPTR 将保持为 31。

当堆栈弹出次数足够卸空堆栈时，下一次出栈会向 PC 返回一个零值，并将 STKUNF 位置 1，而堆栈指针则保持为零。STKUNF 位将保持置 1，直到由软件清零或发生 POR 为止。

注：	下溢时，将零值返回给 PC，会使程序指向复位向量，此时可以验证堆栈状态并采取相应的操作。这与复位不同，因为下溢时 SFR 的内容不受影响。

6.1.4.3 PUSH 和 POP 指令

由于栈顶是可以读写的，因此将值压入堆栈或从堆栈弹出而不影响程序的正常执行是非常理想的。PIC18 指令集包含两条指令 PUSH 和 POP，使用这两条指令可在软件控制下对 TOS 执行操作。然后就可以修改 TOSU、TOSH 和 TOSL，将数据或返回地址压入堆栈。

PUSH 指令将当前的 PC 值压入堆栈。执行该指令会使堆栈指针加 1 并将当前的 PC 值装入堆栈。

POP 指令通过将堆栈指针减 1 来放弃当前的 TOS 值。然后前一个入栈的值就成为了 TOS 值。

寄存器 6-1： STKPTR：堆栈指针寄存器

R/C-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STKFUL(1)	STKUNF(1)	—	SP4	SP3	SP2	SP1	SP0
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 7 **STKFUL：**堆栈满标志位 (1)
1 = 堆栈满或溢出
0 = 堆栈未满也未溢出

bit 6 **STKUNF：**堆栈下溢标志位 (1)
1 = 发生了堆栈下溢
0 = 未发生堆栈下溢

bit 5 **未实现：**读为 0

bit 4-0 **SP<4:0>：**堆栈指针地址位

注 1： 通过用户软件或 POR 清零 bit 7 和 bit 6。

6.1.4.4 堆栈满和下溢复位

通过将配置寄存器 1L 中的 STVREN 位置 1，来使能在堆栈溢出或下溢时的器件复位。当 STVREN 置 1 时，堆栈满或堆栈下溢状态会将相应的 STKFUL 或 STKUNF 位置 1，然后使器件复位。当 STVREN 清零时，堆栈满或堆栈下溢状态会将相应的 STKFUL 或 STKUNF 位置 1，但不会使器件复位。只能通过用户软件或上电复位将 STKFUL 或 STKUNF 位清零。

6.1.5 快速寄存器堆栈

为 STATUS、WREG 和 BSR 寄存器提供的快速寄存器堆栈具有从中断“快速返回”的功能。此堆栈只有一层深且不可读写。当处理器转入中断向量处执行时，它装入对应寄存器的当前值。所有中断源都会将值压入堆栈寄存器。如果使用 RETFIE 和 FAST 指令从中断返回，这些寄存器中的值就会被装回工作寄存器。

如果同时允许低优先级中断和高优先级中断，从低优先级中断返回时，无法可靠地使用堆栈寄存器。如果在处理低优先级中断时，发生了高优先级中断，则低优先级中断存储在堆栈寄存器中的值将被覆盖。在这种情况下，在处理低优先级中断时用户必须用软件保存关键寄存器的值。

如果未使用中断优先级，所有中断都可以使用快速寄存器堆栈从中断返回。如果没有使用中断，快速寄存器堆栈可以用于在子程序调用结束时恢复 STATUS、WREG 和 BSR 寄存器。要在子程序调用中使用快速寄存器堆栈，必须执行 CALL label, FAST 指令，将 STATUS、WREG 和 BSR 寄存器的内容存入快速寄存器堆栈。然后执行 RETURN, FAST 指令，从快速寄存器堆栈恢复这些寄存器。

例 6-1 给出了一个在子程序调用和返回期间使用快速寄存器堆栈的源代码示例。

例 6-1： 快速寄存器堆栈代码示例

```
CALL SUB1, FAST    ;STATUS, WREG, BSR
                   ;SAVED IN FAST REGISTER
                   ;STACK
       •
       •

SUB1   •
       •
   RETURN FAST     ;RESTORE VALUES SAVED
                   ;IN FAST REGISTER STACK
```

6.1.6 程序存储器中的查找表

有的编程场合可能需要在程序存储器中创建数据结构或查找表。对于 PIC18 器件，可以用两种方式实现查找表：

- 计算 GOTO
- 表读

6.1.6.1 计算 GOTO

计算 GOTO 是通过向程序计数器加一个偏移量来实现的。例 6-2 中给出了一个示例。

可以使用 ADDWF PCL 指令和一组 RETLW nn 指令创建一个查找表。在调用该表前，会先将查找表中的偏移量装入 W 寄存器。被调用子程序的第一条指令应该是 ADDWF PCL 指令。接下去执行的一条是 RETLW nn 指令，它将值 nn 返回给调用函数。

偏移量值（WREG 中）指定程序计数器应该增加的字节数，其值应该为 2 的倍数（LSb = 0）。

在这种方式中，每个指令单元只能存储一个数据字节，并且要求返回地址堆栈还有空闲单元。

例 6-2： 使用偏移量值的计算 GOTO

```
        MOVF   OFFSET, W
        CALL   TABLE
ORG     nn00h
TABLE   ADDWF  PCL
        RETLW  nnh
        RETLW  nnh
        RETLW  nnh
        .
        .
        .
```

6.1.6.2 表读

有一种更好的方法可以将数据存储在程序存储器中，这种方法允许在每个指令单元存储 2 个字节的数据。

编程时，每个程序字可以存储 2 个字节的查找表数据。表指针（TBLPTR）指定字节地址，而表锁存器（TABLAT）则存储从程序存储器读取的数据。一次只能从程序存储器读取一个字节。

第 7.1 节“表读与表写”将进一步讨论表读操作。

6.2 PIC18 指令周期

6.2.1 时钟机制

来自内部或外部时钟源的单片机时钟输入都将在内部被四分频以产生四个互不重叠的正交时钟信号（Q1、Q2、Q3 和 Q4）。在单片机内部，程序计数器在每个 Q1 递增；在 Q4 期间，从程序存储器取指令并将指令锁存到指令寄存器（Instruction Register， IR）中。指令的译码和执行在下一个 Q1 到 Q4 周期完成。图 6-4 所示为时钟和指令执行流程。

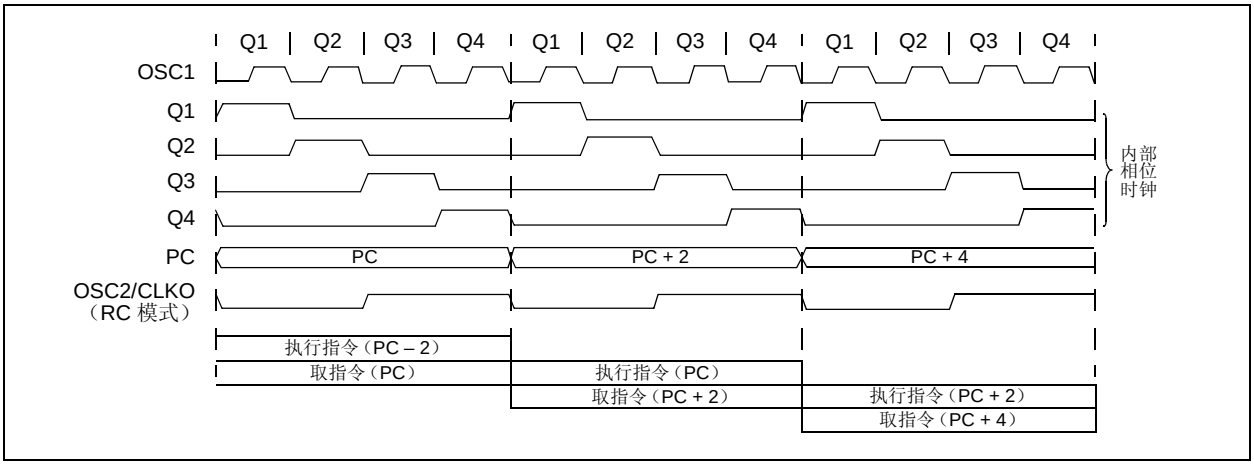
6.2.2 指令流 / 流水线

一个指令周期由 Q1 到 Q4 四个 Q 周期组成。取指令和执行指令是以流水线方式进行的，用一个指令周期来取指令，而用另一个指令周期译码和执行指令。但由于是流水线操作，所以每条指令的等效执行时间都是一个指令周期。如果某条指令改变了程序计数器（如 GOTO），则需要两个指令周期才能完成该指令 （例 6-3）。

取指周期中：程序计数器（PC）在 Q1 周期递增 1，开始取指令。

指令执行周期中：在 Q1 周期，将所取指令锁存到指令寄存器（IR）。在随后的 Q2、Q3 和 Q4 周期中译码并执行该指令。其中读数据存储器 （读操作数）发生在 Q2 周期，写操作 （写目标寄存器）发生在 Q4 周期。

图 6-4： 时钟 / 指令周期



例 6-3： 指令流水线流程

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOVLW 55h	取指 1	执行 1				
2. MOVWF PORTB		取指 2	执行 2			
3. BRA SUB_1			取指 3	执行 3		
4. BSF PORTA, BIT3 (Forced NOP)				取指 4	清除 （NOP）	
5. Instruction @ address SUB_1					取指 SUB_1	执行 SUB_1

除程序跳转指令外，所有的指令都是单周期指令。由于程序跳转指令将导致一条已取指令从流水线清除，与此同时取新指令，然后执行指令，所以程序跳转指令需要两个周期。

6.2.3 程序存储器中的指令

程序存储器按字节寻址。指令以 2 字节或 4 字节的形式存储在程序存储器中。指令字的最低有效字节 （Least Significant Byte， LSB）始终存储在地址为偶数的程序存储单元中 （LSb = 0）。要保证与指令边界对齐， PC 必须以 2 为单位递增，并且 LSb 总是读为 0（见**第 6.1.3 节 “ 程序计数器 ”**）。

图 6-5 给出了指令字存储在程序存储器中的一个示例。

CALL 和 GOTO 指令在指令中嵌入了程序存储器的绝对地址。由于指令总是按字边界存储，因而指令所包含的数据为一个字地址。字地址会写入 PC<20:1>，用于访问程序存储器中的目标字节地址。图 6-5 中的指令 2 给出了指令 GOTO 0006h 在程序存储器中的译码过程。程序跳转指令也采取同样的方式对相对地址偏移量进行译码。存储在跳转指令中的偏移量代表单字指令数，PC 将以此作为偏移量跳转到指定的地址单元。**第 26.0 节 “ 指令集汇总 ”** 提供了指令集的更多详细信息。

图 6-5： 程序存储器中的指令

```
                                                  LSB = 1     LSB = 0     字地址
                                                                          ↓
                  程序存储器
                  字节单元 →                       |           |           |
                                                  |           |           | 000000h
                                                  |           |           | 000002h
                                                  |           |           | 000004h
                                                  |           |           | 000006h
指令 1：          MOVLW     055h                  | 0Fh       | 55h       | 000008h
指令 2：          GOTO      0006h                 | EFh       | 03h       | 00000Ah
                                                  | F0h       | 00h       | 00000Ch
指令 3：          MOVFF     123h, 456h            | C1h       | 23h       | 00000Eh
                                                  | F4h       | 56h       | 000010h
                                                  |           |           | 000012h
                                                  |           |           | 000014h
```

6.2.4 双字指令

标准的 PIC18 指令集有 4 条双字指令：CALL、 MOVFF、 GOTO 和 LSFR。这些指令第二个字的高 4 位 （MSb）均为 1111 ； 其他 12 位是立即数数据，通常为一个数据存储器地址。

指令的高 4 位为 1111，用于代表一条特殊形式的 NOP 指令。指令执行的正确顺序为：执行完第一个字之后立即按顺序访问并使用第二个字中的数据。如果由于某些原因跳过了第一个字而自动执行指令的第二个字，那么将作为一条 NOP 指令执行。如果双字指令跟在修改 PC 的条件指令后，就有必要执行此操作。例 6-4 给出了它的执行过程。

> **注：** 关于扩展指令集中的双字指令信息，请参见**第 6.5 节 “ 程序存储器和扩展指令集 ”**。

例 6-4： 双字指令

情形 1：

目标代码	源代码		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; No, skip this word
1111 0100 0101 0110			; Execute this word as a NOP
0010 0100 0000 0000	ADDWF	REG3	; continue code

情形 2：

目标代码	源代码		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; Yes, execute this word
1111 0100 0101 0110			; 2nd word of instruction
0010 0100 0000 0000	ADDWF	REG3	; continue code

6.3 数据存储器构成

注：	当使能了 PIC18 扩展指令集时，数据存储器某些方面的操作会有所改变。更多信息，请参见**第 6.6 节 “ 数据存储器和扩展指令集 ”**。

PIC18 器件中的数据存储器是用静态 RAM 实现的。在数据存储器中，每个寄存器有 12 位地址，允许数据存储器实现为最大 4,096 个字节。存储空间最多被分为 16 个存储区，每个存储区包含 256 个字节。 PIC18FX6J90 和 PIC18FX7J90 器件实现了全部 16 个完整的存储区，总计 4 KB。图 6-6 和图 6-7 给出了器件的数据存储器构成。

数据存储器由特殊功能寄存器（Special Function Register，SFR）和通用寄存器 （General Purpose Register， GPR）组成。 SFR 用于单片机和外设功能模块的控制和状态指示， GPR 则用于用户应用程序的数据存储和中间结果暂存。任何未实现存储单元均读为 0。

这样的指令集和架构支持跨所有存储区的操作。可以通过直接、间接或变址寻址模式访问整个数据存储器。本章后面的部分将讨论寻址模式。

为确保能在一个周期中访问常用寄存器 （某些 SFR 和 GPR）， PIC18 器件实现了一个快速操作存储区。该存储区是一个 256 字节的存储空间，它可实现对某些 SFR 和 GPR Bank 0 的低地址单元的快速访问，而无需使用 BSR。**第 6.3.2 节 “ 快速操作存储区 ”** 提供了对于快速操作 RAM 的详细说明。

6.3.1 存储区选择寄存器

容量较大的数据存储器需要高效的寻址机制，以便对所有地址进行快速访问。理想状况下，这意味着不必为每次读写操作提供完整地址。 PIC18 器件是使用 RAM 存储区机制实现快速访问的。这种机制将存储空间分成连续的 16 个 256 字节的存储区。根据不同的指令，可以通过完整的 12 位地址，或通过 8 位的低字节地址和 4 位存储区指针直接寻址每个存储单元。

PIC18 指令集中的大部分指令都使用存储区指针，也就是存储区选择寄存器（Bank Select Register，BSR）。 SFR 保存单元地址的高 4 位，而指令本身则包括单元地址的低 8 位。只使用 BSR 的低 4 位（BSR<3:0>），不使用高 4 位；高 4 位始终读为 0 且不能被写入。可以通过使用 MOVLB 指令直接装入 BSR。

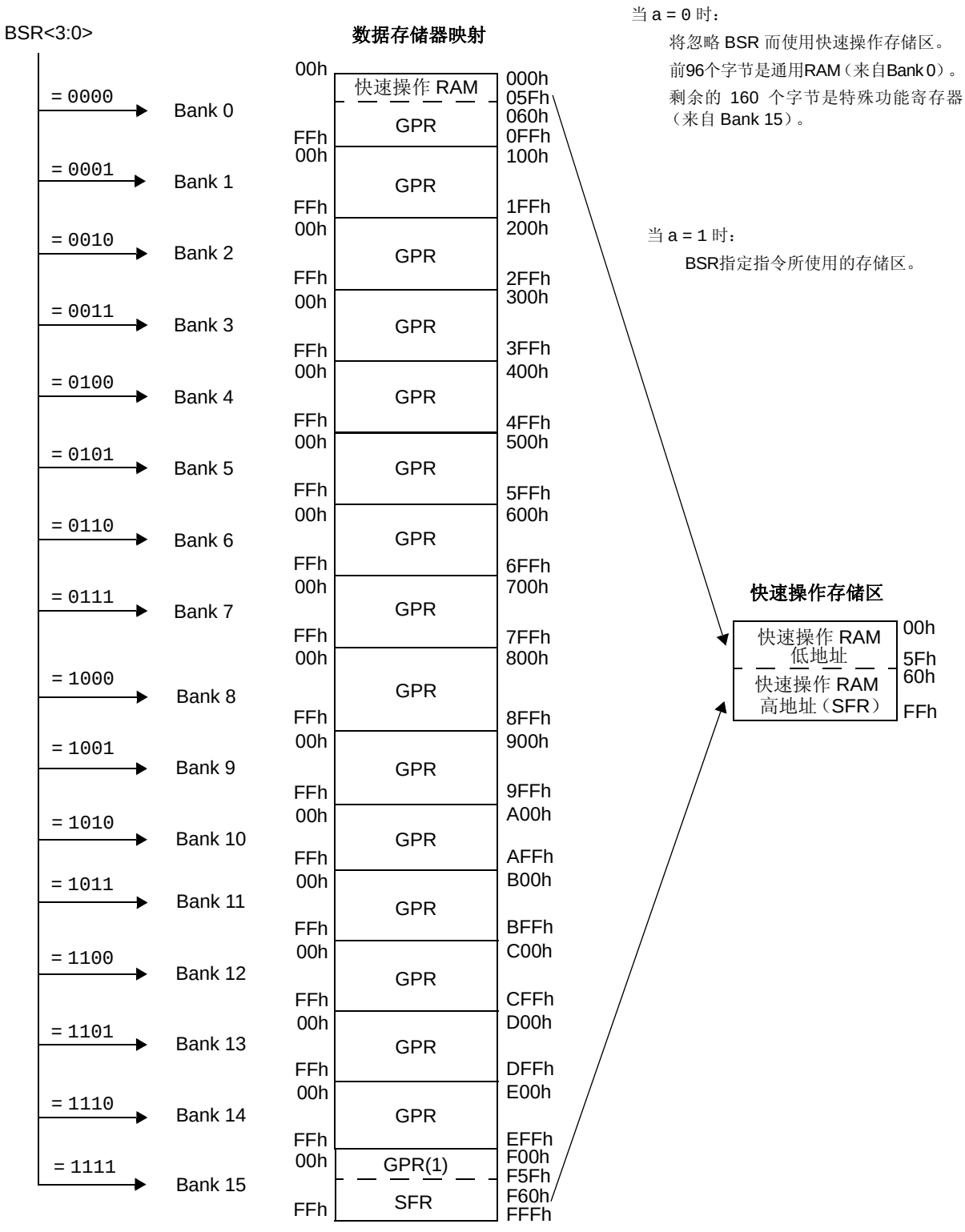
BSR 的值代表数据存储器中的存储区，指令中的 8 位指向存储区中的存储单元，可以将它看作距离存储区下边界的偏移量。图 6-7 显示了 BSR 的值与数据存储器中的存储区之间的关系。

由于最多可有 16 个寄存器共享同一个低位地址，用户必须非常小心以确保在执行数据读或写之前选择了正确的存储区。例如，当 BSR 为 0Fh 时将程序数据写入地址为 F9h 的 8 位地址单元，将导致程序计数器被复位。

当选择存储区时，只有已实现的存储区才可以读写。对未实现存储区进行的写操作将被忽略，而读这些存储区会返回 0。虽然是这样， STATUS 寄存器仍然会受到影响，好像操作是成功的。图 6-6 中的数据存储器映射指出了已实现的存储区。

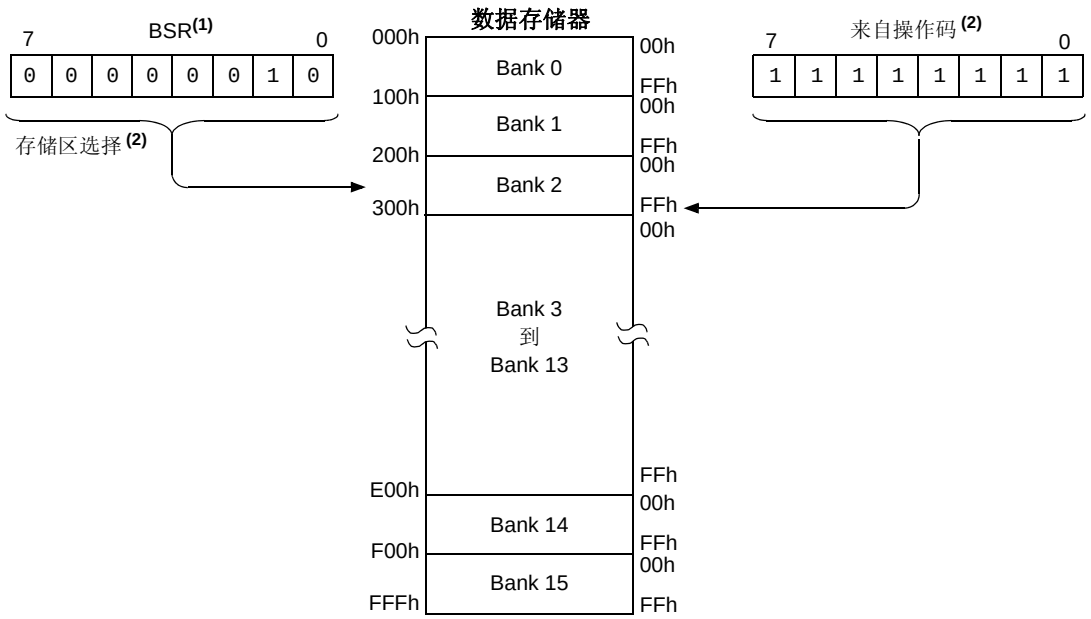
在核心 PIC18 指令集中，只有 MOVFF 指令指定源寄存器和目标寄存器的完整 12 位地址。该指令在执行时完全忽略 BSR。所有其他指令仅包含作为操作数的低位地址，而且必须使用 BSR 或快速操作存储区来寻址目标寄存器。

图 6-6：　　PIC18FX6J90 和 PIC18FX7J90 器件的数据存储器映射



注　1： SFR 还使用 F5Ah 到 F5Fh 的地址，但它们不是快速操作 RAM 的一部分。用户必须始终使用完整的地址或装入正确的 BSR 值，以访问这些寄存器。

图 6-7：　　使用存储区选择寄存器 （直接寻址）



注　　1： 指令的快速操作 RAM 位可用于强制忽略选定的存储区（BSR<3:0>）直接寻址快速操作存储区的寄存器。

2： MOVFF 指令在指令中嵌入了完整的 12 位地址。

6.3.2　　快速操作存储区

使用 BSR 和嵌入的 8 位地址，用户可以寻址数据存储器的整个空间，但这同时也意味着用户必须始终确保选择了正确的存储区。否则，可能会从错误的单元读取数据或将数据写入错误的单元。如果本来是向 GPR 进行写操作，却将结果写入了 SFR，后果是非常严重的。但是在每次对数据存储器进行读或写操作时验证和 / 或更改 BSR 会严重影响工作效率。

为了提高访问大多数常用数据存储单元的效率，现为数据存储器配置了快速操作存储区，这样可以允许用户访问被映射的存储区而无需指定 BSR。快速操作存储区由 Bank 0 的前 96 个字节（00h-5Fh）和 Bank 15 的后 160 个字节（60h-FFh）组成。地址较低的部分被称为 “ 快速操作 RAM”，由 GPR 组成。地址较高的部分则被映射为器件的 SFR。这两个区域被连续地映射到快速操作存储区并且可以用一个 8 位地址进行线性寻址 （图 6-6）。

包括快速操作 RAM 位（指令中的 “a” 参数）的核心 PIC18 指令使用快速操作存储区。当 “a” 等于 1 时，指令使用 BSR 和包含在操作码中的 8 位地址对数据存储器寻址。当 “a” 为 0 时，强制指令使用快速操作存储区地址映射，此时完全忽略 BSR 的当前值。

此 “ 强制 ” 寻址方式可使指令在一个周期内对数据地址进行操作，而不需要首先更新 BSR。这意味着用户可以更高效地对 8 位地址为 60h 或以上的 SFR 进行取值和操作。地址为 60h 以下的快速操作 RAM 非常适合于存储那些用户可能需要快速访问的数据值，如直接计算结果或常用程序变量。快速操作 RAM 也可实现更加快速和高效的现场保护和变量切换代码。

使能扩展指令集 （XINST 配置位 = 1）时的快速操作存储区的映射略有不同。在**第 6.6.3 节 “ 在立即数变址模式下映射快速操作存储区 ”** 中对此进行了更详细的讨论。

6.3.3　　通用寄存器文件

PIC18 器件在 GPR 区中划分了一部分存储区。这部分存储区为数据 RAM，所有指令均可使用它。GPR 区从 Bank 0 的底部 （地址 000h）开始向上延伸直到 SFR 区的底部。上电复位不会初始化 GPR，并且其他复位也不会改变其内容。

6.3.4 特殊功能寄存器

特殊功能寄存器（SFR）是 CPU 和外设模块用来控制所需器件操作的寄存器。这些寄存器以静态 RAM 的形式实现。SFR 从数据存储器的顶部（FFFh）开始向下，它占用了 Bank 15 上面部分的单元空间（F60h 到 FFFh）。表 6-2 和表 6-3 列出了这些寄存器。

可以将 SFR 归类为两组：与“内核”器件功能（ALU、复位和中断）相关的寄存器和与外设功能相关的寄存器。复位和中断寄存器在相关的章节中进行讨论，本章后面的部分将对 ALU 的状态寄存器进行说明。与外设操作相关的寄存器将在该外设的章节中进行说明。

SFR 通常分布在功能受其控制的外设中。未使用的 SFR 单元是未实现的，读为 0。

表 6-2： PIC18F87J90 系列器件的特殊功能寄存器映射

地址	名称	地址	名称	地址	名称	地址	名称	地址	名称	地址	名称
FFFh	TOSU	FDFh	INDF2(1)	FBFh	LCDDATA4(3)	F9Fh	IPR1	F7Fh	SPBRGH1	F5Fh	RTCCFG
FFEh	TOSH	FDEh	POSTINC2(1)	FBEh	LCDDATA3	F9Eh	PIR1	F7Eh	BAUDCON1	F5Eh	RTCCAL
FFDh	TOSL	FDDh	POSTDEC2(1)	FBDh	LCDDATA2	F9Dh	PIE1	F7Dh	LCDDATA23(3)	F5Dh	RTCVALH
FFCh	STKPTR	FDCh	PREINC2(1)	FBCh	LCDDATA1	F9Ch	—(2)	F7Ch	LCDDATA22(3)	F5Ch	RTCVALL
FFBh	PCLATU	FDBh	PLUSW2(1)	FBBh	LCDDATA0	F9Bh	OSCTUNE	F7Bh	LCDDATA21	F5Bh	ALRMCFG
FFAh	PCLATH	FDAh	FSR2H	FBAh	LCDSE5(3)	F9Ah	TRISJ(3)	F7Ah	LCDDATA20	F5Ah	ALRMRPT
FF9h	PCL	FD9h	FSR2L	FB9h	LCDSE4(3)	F99h	TRISH(3)	F79h	LCDDATA19	F59h	ALRMVALH
FF8h	TBLPTRU	FD8h	STATUS	FB8h	LCDSE3	F98h	TRISG	F78h	LCDDATA18	F58h	ALRMVALL
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	LCDSE2	F97h	TRISF	F77h	LCDDATA17(3)	F57h	CTMUCONH
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	LCDSE1	F96h	TRISE	F76h	LCDDATA16(3)	F56h	CTMUCONL
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD	F75h	LCDDATA15	F55h	CTMUICON
FF4h	PRODH	FD4h	—(2)	FB4h	CMCON	F94h	TRISC	F74h	LCDDATA14	F54h	PADCFG1
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB	F73h	LCDDATA13		
FF2h	INTCON	FD2h	LCDREG	FB2h	TMR3L	F92h	TRISA	F72h	LCDDATA12		
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	LATJ(3)	F71h	LCDDATA11(3)		
FF0h	INTCON3	FD0h	RCON	FB0h	—(2)	F90h	LATH(3)	F70h	LCDDATA10(3)		
FEFh	INDF0(1)	FCFh	TMR1H	FAFh	SPBRG1	F8Fh	LATG	F6Fh	LCDDATA9		
FEEh	POSTINC0(1)	FCEh	TMR1L	FAEh	RCREG1	F8Eh	LATF	F6Eh	LCDDATA8		
FEDh	POSTDEC0(1)	FCDh	T1CON	FADh	TXREG1	F8Dh	LATE	F6Dh	LCDDATA7		
FECh	PREINC0(1)	FCCh	TMR2	FACh	TXSTA1	F8Ch	LATD	F6Ch	LCDDATA6		
FEBh	PLUSW0(1)	FCBh	PR2	FABh	RCSTA1	F8Bh	LATC	F6Bh	LCDDATA5(3)		
FEAh	FSR0H	FCAh	T2CON	FAAh	LCDPS	F8Ah	LATB	F6Ah	CCPR1H		
FE9h	FSR0L	FC9h	SSPBUF	FA9h	LCDSE0	F89h	LATA	F69h	CCPR1L		
FE8h	WREG	FC8h	SSPADD	FA8h	LCDCON	F88h	PORTJ(3)	F68h	CCP1CON		
FE7h	INDF1(1)	FC7h	SSPSTAT	FA7h	EECON2	F87h	PORTH(3)	F67h	CCPR2H		
FE6h	POSTINC1(1)	FC6h	SSPCON1	FA6h	EECON1	F86h	PORTG	F66h	CCPR2L		
FE5h	POSTDEC1(1)	FC5h	SSPCON2	FA5h	IPR3	F85h	PORTF	F65h	CCP2CON		
FE4h	PREINC1(1)	FC4h	ADRESH	FA4h	PIR3	F84h	PORTE	F64h	SPBRG2		
FE3h	PLUSW1(1)	FC3h	ADRESL	FA3h	PIE3	F83h	PORTD	F63h	RCREG2		
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC	F62h	TXREG2		
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB	F61h	TXSTA2		
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA	F60h	RCSTA2		

注 **1：** 这不是实际存在的寄存器。
2： 未实现的寄存器，读为 0。
3： 该寄存器在 PIC18F6XJ90 器件上不可用。

表 6-3： PIC18F87J90 系列寄存器文件汇总

寄存器名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR/BOR 时的值	详情请见（页）
TOSU	—	—	—	栈顶最高字节 （TOS<20:16>）					---0 0000	59, 67
TOSH	栈顶寄存器高字节 （TOS<15:8>）								0000 0000	59, 67
TOSL	栈顶寄存器低字节 （TOS<7:0>）								0000 0000	59, 67
STKPTR	STKFUL	STKUNF	—	返回堆栈指针					uu-0 0000	59, 68
PCLATU	—	—	bit 21(1)	PC<20:16> 的保持寄存器					---0 0000	59, 67
PCLATH	PC<15:8> 的保持寄存器								0000 0000	59, 67
PCL	PC 低字节 （PC<7:0>）								0000 0000	59, 67
TBLPTRU	—	—	bit 21	程序存储器表指针最高字节 （TBLPTR<20:16>）					--00 0000	59, 92
TBLPTRH	程序存储器表指针高字节 （TBLPTR<15:8>）								0000 0000	59, 92
TBLPTRL	程序存储器表指针低字节 （TBLPTR<7:0>）								0000 0000	59, 92
TABLAT	程序存储器表锁存器								0000 0000	59, 92
PRODH	乘积寄存器的高字节								xxxx xxxx	59, 99
PRODL	乘积寄存器的低字节								xxxx xxxx	59, 99
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	59, 103
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP	1111 1111	59, 104
INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF	1100 0000	59, 105
INDF0	使用 FSR0 的内容寻址数据存储器 ——FSR0 的值不变 （不是实际存在的寄存器）								N/A	59, 83
POSTINC0	使用 FSR0 的内容寻址数据存储器 ——FSR0 的值后递增 （不是实际存在的寄存器）								N/A	59, 84
POSTDEC0	使用 FSR0 的内容寻址数据存储器 ——FSR0 的值后递减 （不是实际存在的寄存器）								N/A	59, 84
PREINC0	使用 FSR0 的内容寻址数据存储器 ——FSR0 的值预递增 （不是实际存在的寄存器）								N/A	59, 84
PLUSW0	使用 FSR0 的内容寻址数据存储器 ——FSR0 的值预递增 （不是实际存在的寄存器）， FSR0 偏移量的值由 W 寄存器提供								N/A	59, 84
FSR0H	—	—	—	—	间接数据存储器地址指针 0 的高字节				---- xxxx	59, 83
FSR0L	间接数据存储器地址指针 0 的低字节								xxxx xxxx	59, 83
WREG	工作寄存器								xxxx xxxx	59
INDF1	使用 FSR1 的内容寻址数据存储器 ——FSR1 的值不变 （不是实际存在的寄存器）								N/A	59, 83
POSTINC1	使用 FSR1 的内容寻址数据存储器 ——FSR1 的值后递增 （不是实际存在的寄存器）								N/A	59, 84
POSTDEC1	使用 FSR1 的内容寻址数据存储器 ——FSR1 的值后递减 （不是实际存在的寄存器）								N/A	59, 84
PREINC1	使用 FSR1 的内容寻址数据存储器 ——FSR1 的值预递增 （不是实际存在的寄存器）								N/A	59, 84
PLUSW1	使用 FSR1 的内容寻址数据存储器 ——FSR1 的值预递增 （不是实际存在的寄存器）， FSR1 偏移量的值由 W 寄存器提供								N/A	59, 84
FSR1H	—	—	—	—	间接数据存储器地址指针 1 的高字节				---- xxxx	60, 83
FSR1L	间接数据存储器地址指针 1 的低字节								xxxx xxxx	60, 83
BSR	—	—	—	—	存储区选择寄存器				---- 0000	60, 72
INDF2	使用 FSR2 的内容寻址数据存储器 ——FSR2 的值不变 （不是实际存在的寄存器）								N/A	60, 83
POSTINC2	使用 FSR2 的内容寻址数据存储器 ——FSR2 的值后递增 （不是实际存在的寄存器）								N/A	60, 84
POSTDEC2	使用 FSR2 的内容寻址数据存储器 ——FSR2 的值后递减 （不是实际存在的寄存器）								N/A	60, 84
PREINC2	使用 FSR2 的内容寻址数据存储器 ——FSR2 的值预递增 （不是实际存在的寄存器）								N/A	60, 84
PLUSW2	使用 FSR2 的内容寻址数据存储器 ——FSR2 的值预递增 （不是实际存在的寄存器）， FSR2 偏移量的值由 W 寄存器提供								N/A	60, 84
FSR2H	—	—	—	—	间接数据存储器地址指针 2 的高字节				---- xxxx	60, 83
FSR2L	间接数据存储器地址指针 2 的低字节								xxxx xxxx	60, 83
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx	60, 81

图注： x = 未知， u = 不变， — = 未实现， q = 值取决于具体条件， r = 保留，不要修改

注 1： PC 的 bit 21 仅在测试模式和串行编程模式下可用。

2： 这些寄存器和 / 或位仅在 80 引脚器件上可用；在其他器件上，它们未实现并读为 0。所示为 80 引脚器件的复位状态。

3： 当 MSSP 模块工作在 I^2C™ 从模式下时，这些位的备用名称和定义。详情请参见**第 18.4.3.2 节 “ 地址掩码 ”**。

4： 只有在特定的振荡器配置中才可使用 PLLEN 位；否则，它被禁止并读为 0。详情请参见**第 3.4.3 节 “PLL 倍频器 ”**。

5： 仅当内部振荡器被选为默认时钟源 （FOSC2 配置位 = 0） 时， RA<7:6> 及其相关的锁存和方向位才能被配置为端口引脚；否则，这些位将被禁止且读为 0。

表 6-3： PIC18F87J90 系列寄存器文件汇总 （续）

寄存器名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR/BOR 时的值	详情请见（页）
TMR0H	Timer0 高字节								0000 0000	60, 141
TMR0L	Timer0 低字节								xxxx xxxx	60, 141
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	60, 141
OSCCON	IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOFS	SCS1	SCS0	0110 q000	36, 60
LCDREG	—	CPEN	BIAS2	BIAS1	BIAS0	MODE13	CKSEL1	CKSEL0	-011 1100	60, 189
WDTCON	REGSLP	—	—	—	—	—	—	SWDTEN	0--- ---0	60, 332
RCON	IPEN	—	$\overline{CM}$	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	0-11 11q0	54, 60
TMR1H	Timer1 寄存器的高字节								xxxx xxxx	60, 147
TMR1L	Timer1 寄存器的低字节								xxxx xxxx	60, 147
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	0000 0000	60, 143
TMR2	Timer2 寄存器								0000 0000	60, 150
PR2	Timer2 周期寄存器								1111 1111	60, 150
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	60, 149
SSPBUF	MSSP 接收缓冲区 / 发送寄存器								xxxx xxxx	60, 219, 254
SSPADD	I^2C™ 从模式下的 MSSP 地址寄存器。 I^2C 主模式下的 MSSP1 波特率重载寄存器。								0000 0000	60, 254
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	60, 212, 221
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	60, 213, 222
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	60, 223, 224
	GCEN	ACKSTAT	ADMSK5(3)	ADMSK4(3)	ADMSK3(3)	ADMSK2(3)	ADMSK1(3)	SEN		
ADRESH	A/D 结果寄存器的高字节								xxxx xxxx	61, 297
ADRESL	A/D 结果寄存器的低字节								xxxx xxxx	61, 297
ADCON0	ADCAL	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	ADON	0-00 0000	61, 289
ADCON1	TRIGSEL	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	0-00 0000	61, 290
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	0-00 0000	61, 291
LCDDATA4	S39C0(2)	S38C0(2)	S37C0(2)	S36C0(2)	S35C0(2)	S34C0(2)	S33C0(2)	S32C0	xxxx xxxx	61, 187
LCDDATA3	S31C0	S30C0	S29C0	S28C0	S27C0	S26C0	S25C0	S24C0	xxxx xxxx	61, 187
LCDDATA2	S23C0	S22C0	S21C0	S20C0	S19C0	S18C0	S17C0	S16C0	xxxx xxxx	61, 187
LCDDATA1	S15C0	S14C0	S13C0	S12C0	S11C0	S10C0	S09C0	S08C0	xxxx xxxx	61, 187
LCDDATA0	S07C0	S06C0	S05C0	S04C0	S03C0	S02C0	S01C0	S00C0	xxxx xxxx	61, 187
LCDSE5(2)	SE47	SE46	SE45	SE44	SE43	SE42	SE41	SE40	0000 0000	61, 187
LCDSE4	SE39(2)	SE38(2)	S37(2)	SE36(2)	SE35(2)	SE34(2)	SE33(2)	SE32	0000 0000	61, 187
LCDSE3	SE31	SE30	SE29	SE28	SE27	SE26	SE25	SE24	0000 0000	61, 187
LCDSE2	SE23	SE22	SE21	SE20	SE19	SE18	SE17	SE16	0000 0000	61, 187
LCDSE1	SE15	SE14	SE13	SE12	SE11	SE10	SE09	SE08	0000 0000	61, 187
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000 0000	61, 305
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0111	61, 299
TMR3H	Timer3 寄存器的高字节								xxxx xxxx	61, 153
TMR3L	Timer3 寄存器的低字节								xxxx xxxx	61, 153
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{T3SYNC}$	TMR3CS	TMR3ON	0000 0000	61, 151

图注： x = 未知， u = 不变， — = 未实现， q = 值取决于具体条件， r = 保留，不要修改

注 1： PC 的 bit 21 仅在测试模式和串行编程模式下可用。

2： 这些寄存器和 / 或位仅在 80 引脚器件上可用；在其他器件上，它们未实现并读为 0。所示为 80 引脚器件的复位状态。

3： 当 MSSP 模块工作在 I^2C™ 从模式下时，这些位的备用名称和定义。详情请参见**第 18.4.3.2 节 “ 地址掩码 ”**。

4： 只有在特定的振荡器配置中才可使用 PLLEN 位；否则，它被禁止并读为 0。详情请参见**第 3.4.3 节 “PLL 倍频器 ”**。

5： 仅当内部振荡器被选为默认时钟源 （FOSC2 配置位 = 0）时， RA<7:6> 及其相关的锁存和方向位才能被配置为端口引脚；否则，这些位将被禁止且读为 0。

表 6-3： PIC18F87J90 系列寄存器文件汇总 （续）

寄存器名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR/BOR 时的值	详情请见（页）
SPBRG1	EUSART 波特率发生器的低字节								0000 0000	61, 259
RCREG1	EUSART 接收寄存器								0000 0000	61, 267
TXREG1	EUSART 发送寄存器								0000 0000	61, 265
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	61, 256
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	61, 257
LCDPS	WFT	BIASMD	LCDA	WA	LP3	LP2	LP1	LP0	0000 0000	61, 185
LCDSE0	SE07	SE06	SE05	SE04	SE03	SE02	SE01	SE00	0000 0000	61, 186
LCDCON	LCDEN	SLPEN	WERR	—	CS1	CS0	LMUX1	LMUX0	000- 0000	61, 184
EECON2	EEPROM 控制寄存器 2（不是实际存在的寄存器）								---- ----	61, 90
EECON1	—	—	WPROG	FREE	WRERR	WREN	WR	—	--00 x00-	61, 90
IPR3	—	LCDIP	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	RTCCIP	-111 1111	62, 114
PIR3	—	LCDIF	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	RTCCIF	-000 0000	62, 108
PIE3	—	LCDIE	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	RTCCIE	-000 0000	62, 111
IPR2	OSCFIP	CMIP	—	—	BCLIP	LVDIP	TMR3IP	—	11-- 111-	62, 113
PIR2	OSCFIF	CMIF	—	—	BCLIF	LVDIF	TMR3IF	—	00-- 000-	62, 107
PIE2	OSCFIE	CMIE	—	—	BCLIE	LVDIE	TMR3IE	—	00-- 000-	62, 110
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	-111 1-11	62, 112
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	-000 0-00	62, 106
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	-000 0-00	62, 109
OSCTUNE	INTSRC	PLLEN(4)	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0	0000 0000	37, 62
TRISJ(2)	TRISJ7	TRISJ6	TRISJ5	TRISJ4	TRISJ3	TRISJ2	TRISJ1	TRISJ0	1111 1111	62, 138
TRISH(2)	TRISH7	TRISH6	TRISH5	TRISH4	TRISH3	TRISH2	TRISH1	TRISH0	1111 1111	62, 136
TRISG	SPIOD	CCP2OD	CCP1OD	TRISG4	TRISG3	TRISG2	TRISG1	TRISG0	0001 1111	62, 134
TRISF	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	—	1111 111-	62, 132
TRISE	TRISE7	TRISE6	TRISE5	TRISE4	TRISE3	—	TRISE1	TRISE0	1111 1-11	62, 129
TRISD	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	1111 1111	62, 127
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	1111 1111	62, 125
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	1111 1111	62, 122
TRISA	TRISA7(5)	TRISA6(5)	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	1111 1111	62, 119
LATJ(2)	LATJ7	LATJ6	LATJ5	LATJ4	LATJ3	LATJ2	LATJ1	LATJ0	xxxx xxxx	62, 138
LATH(2)	LATH7	LATH6	LATH5	LATH4	LATH3	LATH2	LATH1	LATH0	xxxx xxxx	62, 136
LATG	U2OD	U1OD	—	LATG4	LATG3	LATG2	LATG1	LATG0	00-x xxxx	62, 134
LATF	LATF7	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	—	xxxx xxx-	62, 132
LATE	LATE7	LATE6	LATE5	LATE4	LATE3	—	LATE1	LATE0	xxxx x-xx	62, 129
LATD	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0	xxxx xxxx	62, 127
LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	xxxx xxxx	62, 125
LATB	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	xxxx xxxx	62, 122
LATA	LATA7(5)	LATA6(5)	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0	xxxx xxxx	62, 119

图注： x = 未知， u = 不变， — = 未实现， q = 值取决于具体条件， r = 保留，不要修改

注 1： PC 的 bit 21 仅在测试模式和串行编程模式下可用。

2： 这些寄存器和 / 或位仅在 80 引脚器件上可用；在其他器件上，它们未实现并读为 0。所示为 80 引脚器件的复位状态。

3： 当 MSSP 模块工作在 I^2C™ 从模式下时，这些位的备用名称和定义。详情请参见**第 18.4.3.2 节 “ 地址掩码 ”**。

4： 只有在特定的振荡器配置中才可使用 PLLEN 位；否则，它被禁止并读为 0。详情请参见**第 3.4.3 节 “PLL 倍频器 ”**。

5： 仅当内部振荡器被选为默认时钟源 （FOSC2 配置位 = 0）时， RA<7:6> 及其相关的锁存和方向位才能被配置为端口引脚；否则，这些位将被禁止且读为 0。

表 6-3： PIC18F87J90 系列寄存器文件汇总 （续）

寄存器名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR/BOR 时的值	详情请见（页）
PORTJ(2)	RJ7	RJ6	RJ5	RJ4	RJ3	RJ2	RJ1	RJ0	xxxx xxxx	62, 138
PORTH(2)	RH7	RH6	RH5	RH4	RH3	RH2	RH1	RH0	xxxx xxxx	62, 136
PORTG	RDPU	REPU	RJPU(2)	RG4	RG3	RG2	RG1	RG0	000x xxxx	62, 134
PORTF	RF7	RF6	RF5	RF4	RF3	RF2	RF1	—	xxxx xxx-	62, 132
PORTE	RE7	RE6	RE5	RE4	RE3	—	RE1	RE0	xxxx x-xx	63, 129
PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	63, 127
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	63, 125
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	63, 122
PORTA	RA7(5)	RA6(5)	RA5	RA4	RA3	RA2	RA1	RA0	xx0x 0000	63, 119
SPBRGH1	EUSART 波特率发生器的高字节								0000 0000	63, 259
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	0100 0-00	63, 258
LCDDATA23(2)	S47C3	S46C3	S45C3	S44C3	S43C3	S42C3	S41C3	S40C3	xxxx xxxx	63, 187
LCDDATA22	S39C3(2)	S38C3(2)	S37C3(2)	S36C3(2)	S35C3(2)	S34C3(2)	S33C3(2)	S32C3	xxxx xxxx	63, 187
LCDDATA21	S31C3	S30C3	S29C3	S28C3	S27C3	S26C3	S25C3	S24C3	xxxx xxxx	63, 187
LCDDATA20	S23C3	S22C3	S21C3	S20C3	S19C3	S18C3	S17C3	S16C3	xxxx xxxx	63, 187
LCDDATA19	S15C3	S14C3	S13C3	S12C3	S11C3	S10C3	S09C3	S08C3	xxxx xxxx	63, 187
LCDDATA18	S07C3	S06C3	S05C3	S04C3	S03C3	S02C3	S01C3	S00C3	xxxx xxxx	63, 187
LCDDATA17(2)	S47C2	S46C2	S45C2	S44C2	S43C2	S42C2	S41C2	S40C2	xxxx xxxx	63, 187
LCDDATA16	S39C2(2)	S38C2(2)	S37C2(2)	S36C2(2)	S35C2(2)	S34C2(2)	S33C2(2)	S32C2	xxxx xxxx	63, 187
LCDDATA15	S31C2	S30C2	S29C2	S28C2	S27C2	S26C2	S25C2	S24C2	xxxx xxxx	63, 187
LCDDATA14	S23C2	S22C2	S21C2	S20C2	S19C2	S18C2	S17C2	S16C2	xxxx xxxx	63, 187
LCDDATA13	S15C2	S14C2	S13C2	S12C2	S11C2	S10C2	S09C2	S08C2	xxxx xxxx	63, 187
LCDDATA12	S07C2	S06C2	S05C2	S04C2	S03C2	S02C2	S01C2	S00C2	xxxx xxxx	63, 187
LCDDATA11(2)	S47C1	S46C1	S45C1	S44C1	S43C1	S42C1	S41C1	S40C1	xxxx xxxx	63, 187
LCDDATA10	S39C1(2)	S38C1(2)	S37C1(2)	S36C1(2)	S35C1(2)	S34C1(2)	S33C1(2)	S32C1	xxxx xxxx	63, 187
LCDDATA9	S31C1	S30C1	S29C1	S28C1	S27C1	S26C1	S25C1	S24C1	xxxx xxxx	63, 187
LCDDATA8	S23C1	S22C1	S21C1	S20C1	S19C1	S18C1	S17C1	S16C1	xxxx xxxx	63, 187
LCDDATA7	S15C1	S14C1	S13C1	S12C1	S11C1	S10C1	S09C1	S08C1	xxxx xxxx	63, 187
LCDDATA6	S07C1	S06C1	S05C1	S04C1	S03C1	S02C1	S01C1	S00C1	xxxx xxxx	63, 187
LCDDATA5(2)	S47C0	S46C0	S45C0	S44C0	S43C0	S42C0	S41C0	S40C0	xxxx xxxx	63, 187
CCPR1H	捕捉 / 比较 /PWM 寄存器 1 的高字节								xxxx xxxx	63, 174
CCPR1L	捕捉 / 比较 /PWM 寄存器 1 的低字节								xxxx xxxx	63, 174
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	63, 173
CCPR2H	捕捉 / 比较 /PWM 寄存器 2 的高字节								xxxx xxxx	63, 174
CCPR2L	捕捉 / 比较 /PWM 寄存器 2 的低字节								xxxx xxxx	64, 174
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	64, 173
SPBRG2	AUSART 波特率发生器寄存器								0000 0000	64, 278
RCREG2	AUSART 接收寄存器								0000 0000	64, 283
TXREG2	AUSART 发送寄存器								0000 0000	64, 281
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	64, 276
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	64, 277
RTCCFG	RTCEN	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE	RTCPTR1	RTCPTR0	0-00 0000	64, 157
RTCCAL	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	0000 0000	64, 158
RTCVALH	RTCC 值寄存器窗口的高字节，基于 RTCPTR<1:0>								xxxx xxxx	64, 160

图注： x = 未知， u = 不变， — = 未实现， q = 值取决于具体条件， r = 保留，不要修改

注 1： PC 的 bit 21 仅在测试模式和串行编程模式下可用。

2： 这些寄存器和 / 或位仅在 80 引脚器件上可用；在其他器件上，它们未实现并读为 0。所示为 80 引脚器件的复位状态。

3： 当 MSSP 模块工作在 I^2C™ 从模式下时，这些位的备用名称和定义。详情请参见**第 18.4.3.2 节 “ 地址掩码 ”**。

4： 只有在特定的振荡器配置中才可使用 PLLEN 位；否则，它被禁止并读为 0。详情请参见**第 3.4.3 节 “PLL 倍频器 ”**。

5： 仅当内部振荡器被选为默认时钟源 （FOSC2 配置位 = 0）时， RA<7:6> 及其相关的锁存和方向位才能被配置为端口引脚；否则，这些位将被禁止且读为 0。

表 6-3： PIC18F87J90 系列寄存器文件汇总 （续）

寄存器名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR/BOR 时的值	详情请见（页）
RTCVALL	RTCC 值寄存器窗口的低字节，基于 RTCPTR<1:0>								xxxx xxxx	64, 160
ALRMCFG	ALRMEN	CHIME	AMASK3	AMASK2	AMASK1	AMASK0	ALRMPTR1	ALRMPTR0	0000 0000	64, 159
ALRMRPT	ARPT7	ARPT6	ARPT5	ARPT4	ARPT3	ARPT2	ARPT1	ARPT0	0000 0000	64, 160
ALRMVALH	闹钟值寄存器窗口的高字节，基于 ALRMPTR<1:0>								xxxx xxxx	64, 163
ALRMVALL	闹钟值寄存器窗口的低字节，基于 ALRMPTR<1:0>								xxxx xxxx	64, 163
CTMUCONH	CTMUEN	—	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	CTTRIG	0-00 0000	64, 321
CTMUCONL	EDG2POL	EDG2SEL1	EDG2SEL0	EDG1POL	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT	0000 0000	64, 322
CTMUICON	ITRIM5	ITRIM4	ITRIM3	ITRIM2	ITRIM1	ITRIM0	IRNG1	IRNG0	0000 0000	64, 323
PADCFG1	—	—	—	—	—	RTSECSEL1	RTSECSEL0	—	---- -00-	64, 158

图注： x = 未知， u = 不变， — = 未实现， q = 值取决于具体条件， r = 保留，不要修改

注 1： PC 的 bit 21 仅在测试模式和串行编程模式下可用。

2： 这些寄存器和 / 或位仅在 80 引脚器件上可用；在其他器件上，它们未实现并读为 0。所示为 80 引脚器件的复位状态。

3： 当 MSSP 模块工作在 I^2C™ 从模式下时，这些位的备用名称和定义。详情请参见**第 18.4.3.2 节 “ 地址掩码 ”**。

4： 只有在特定的振荡器配置中才可使用 PLLEN 位；否则，它被禁止并读为 0。详情请参见**第 3.4.3 节 “PLL 倍频器 ”**。

5： 仅当内部振荡器被选为默认时钟源 （FOSC2 配置位 = 0）时， RA<7:6> 及其相关的锁存和方向位才能被配置为端口引脚；否则，这些位将被禁止且读为 0。

6.3.5 STATUS 寄存器

如寄存器 6-2 所示，STATUS 寄存器包含 ALU 的算术运算状态。和任何其他寄存器一样，STATUS 寄存器可以作为任何指令的操作数。如果一条影响 Z、DC、C、OV 或 N 位的指令以 STATUS 寄存器作为目标寄存器，则会禁止对这 5 位进行写操作。

这些位根据器件逻辑被置 1 或清零。因此，当执行一条把 STATUS 寄存器作为目标寄存器的指令后，运行结果可能与预想的不同。例如，CLRF STATUS 将 Z 位置 1 并保持其余位不变。然后读 STATUS 寄存器将得到“000u u1uu”。因此，建议仅使用 BCF、BSF、SWAPF、MOVFF 和 MOVWF 指令来改变 STATUS 寄存器，因为这些指令不会影响 STATUS 寄存器中的 Z、C、DC、OV 或 N 位。

关于其他不会影响状态位的指令，请参见表26-2和表26-3中的指令集汇总。

注：	在减法运算中，C 和 DC 位分别作为借位位和半借位位。

寄存器 6-2： STATUS 寄存器

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC(1)	C(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 7-5 **未实现：**读为 0

bit 4 **N：**负标志位
此位用于有符号的算术运算（以二进制补码方式进行）。它表明结果是否为负（ALU MSB = 1）。
1 = 结果为负
0 = 结果为正

bit 3 **OV：**溢出标志位
此位用于有符号的算术运算（以二进制补码方式进行）。它表明运算结果溢出了 7 位二进制数的范围，溢出导致符号位（bit 7）发生改变。
1 = 有符号算术运算中发生溢出（本次算术运算）
0 = 未发生溢出

bit 2 **Z：**全零标志位
1 = 算术运算或逻辑运算的结果为零
0 = 算术运算或逻辑运算的结果不为零

bit 1 **DC：**半进位 / 借位位 (1)
对于 ADDWF、ADDLW、SUBLW 和 SUBWF 指令：
1 = 结果的第 4 个低位发生了进位
0 = 结果的第 4 个低位未发生进位

bit 0 **C：**进位 / 借位位 (2)
对于 ADDWF、ADDLW、SUBLW 和 SUBWF 指令：
1 = 结果的最高位发生了进位
0 = 结果的最高位未发生进位

注 1： 对于借位，极性是相反的。减法是通过加上第二个操作数的二进制补码来执行的。对于移位指令（RRF 和 RLF），此位来自源寄存器的 bit 4 或 bit 3。

2： 对于借位，极性是相反的。减法是通过加上第二个操作数的二进制补码来执行的。对于移位指令（RRF 和 RLF），此位来自源寄存器的最高位或最低位。

6.4 数据寻址模式

注：	当使能 PIC18 扩展指令集时，核心 PIC18 指令集中某些指令的执行方式会发生改变。更多信息，请参见**第 6.6 节 “ 数据存储器和扩展指令集 ”**。

程序存储器只能用一种方式寻址 （通过程序计数器），而数据存储空间可用多种方式寻址。大部分指令的寻址模式都是固定的。其他指令可能使用最多三种模式，根据它们所使用的操作数和是否使能了扩展指令集而定。

这些寻址模式为：

- 固有寻址
- 立即数寻址
- 直接寻址
- 间接寻址

当使能了扩展指令集（XINST 配置位 = 1）时，还可使用另外一种寻址模式，即立即数变址寻址模式。**第 6.6.1 节 “ 使用立即数偏移量进行变址寻址 ”**将更详细讨论它的操作。

6.4.1 固有寻址和立即数寻址

很多 PIC18 控制指令根本不需要任何参数；执行这些指令要么对整个器件造成影响，要么仅隐式地针对一个寄存器进行操作。此寻址模式就是固有寻址。例如指令 SLEEP、 RESET 和 DAW。

其他指令的工作方式与此类似，但需要操作码中有其他显式的参数。由于需要一些立即数作为参数，这种寻址模式被称为立即数寻址。例如 ADDLW 和 MOVLW，它们分别向 W 寄存器加或移入立即数值。其他立即数寻址指令，例如 CALL 和 GOTO，它们包括一个 20 位的程序存储器地址。

6.4.2 直接寻址

直接寻址在操作码中指定操作的全部或部分源地址和/或目标地址。这些选项由指令附带的参数指定。

在核心 PIC18 指令集中，针对位和针对字节的指令默认情况下使用直接寻址。所有这些指令都包含某个 8 位的立即数地址作为其最低有效字节。此地址指定数据RAM 的某个存储区中寄存器的地址 （**第 6.3.3 节 “ 通用寄存器文件 ”**）或快速操作存储区 （**第 6.3.2 节 “ 快速操作存储区 ”**）中作为指令数据源的单元地址。

快速操作 RAM 位 “a” 决定地址的解析方式。当 “a” 为 1 时， BSR （**第 6.3.1 节 “ 存储区选择寄存器 ”**）的内容将和指令中的直接地址一起用于确定寄存器的完整 12 位地址。当 “a” 为 0 时，此直接地址将被解析为快速操作存储区中的一个寄存器。使用快速操作 RAM 的寻址模式有时也被称为直接强制寻址模式。

有几条指令，例如 MOVFF，在操作码中包含完整的 12 位地址（源地址或目标地址）。在这些情况下， BSR 被完全忽略。

保存操作结果的目标寄存器由目标位 “d” 确定。当 “d” 为 1 时，结果被存回源寄存器并覆盖原来的内容。当 “d” 为 0 时，结果被存储在 W 寄存器中。没有 “d” 参数的指令的目标寄存器隐含在指令中，这些指令的目标寄存器是正在操作的目标寄存器或 W 寄存器。

6.4.3 间接寻址

间接寻址允许用户访问数据存储器中的单元而无需在指令中给出一个固定的地址。这种寻址模式是通过使用文件选择寄存器 （File Select Register， FSR）作为指向要读写单元的指针实现的。由于FSR 本身作为特殊功能寄存器位于 RAM 中，因此也可在程序控制中直接操作它们。这使得FSR 对于在数据存储器中实现诸如表和数组等数据结构时非常有用。

也可以使用间接文件操作数 （Indirect File Operand， INDF）进行寄存器间接寻址。这种操作允许自动递增、递减或偏移指针，从而自动操作指针的值。它通过使用循环提高代码执行效率，如例 6-5 所示的清零整个 RAM 存储区的操作。它还允许用户在数据存储器中执行变址寻址和其他针对程序存储器的堆栈指针操作。

例 6-5： 如何使用间接寻址清零 RAM （BANK 1）

```
        LFSR    FSR0, 100h ;
NEXT    CLRF    POSTINC0   ; Clear INDF
                           ; register then
                           ; inc pointer
        BTFSS   FSR0H, 1   ; All done with
                           ; Bank1?
        BRA     NEXT       ; NO, clear next
CONTINUE                   ; YES, continue
```

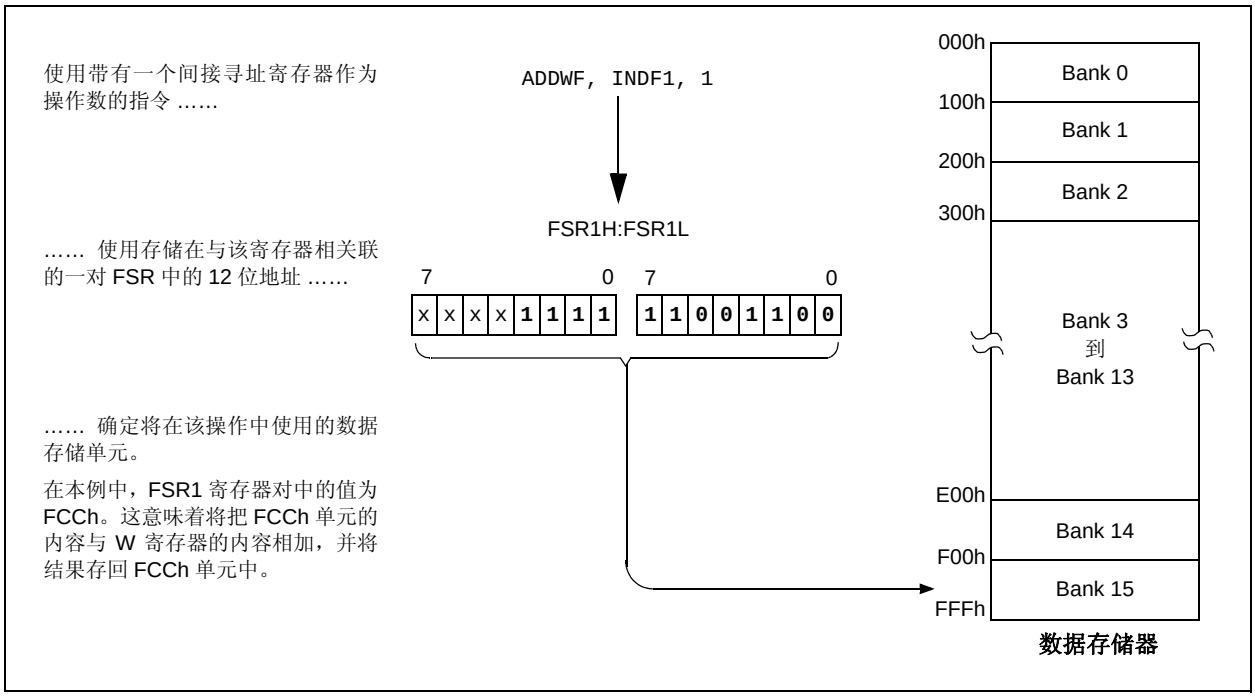
6.4.3.1 FSR 寄存器和 INDF 操作数

间接寻址的核心是三组寄存器：FSR0、FSR1 和 FSR2。每组寄存器都含有一对 8 位寄存器：FSRnH 和 FSRnL。FSRnH 寄存器的高 4 位未使用，因此每对 FSR 只保存一个 12 位值，可通过这个值线性寻址数据存储器的整个空间。因此，FSR 寄存器对被用作数据存储器的地址指针。

间接寻址是通过一组间接文件操作数（INDF0 到 INDF2）完成的。这些操作数可被看作“虚拟”寄存器：它们被映射到 SFR 空间而不是物理实现的。对特定的 INDF 寄存器执行读或写操作实际上访问的是与之对应的一对 FSR 寄存器。例如，读 INDF1 就是读 FSR1H:FSR1L 指向地址中的数据。使用 INDF 寄存器作为操作数的指令实际上使用相应 FSR 的内容作为指向指令目标地址的指针。INDF 操作数只是使用指针的一种简便方法。

由于间接寻址使用完整的 12 位地址，因此没有必要进行数据 RAM 分区。所以 BSR 的当前内容和快速操作 RAM 位对于确定目标地址没有影响。

图 6-8： 间接寻址



6.4.3.2 FSR 寄存器和 POSTINC、POSTDEC、 PREINC 以及 PLUSW

除了 INDF 操作数之外，每对 FSR 寄存器还有 4 个额外的间接操作数。和 INDF 一样，它们也都是不能直接读写的“虚拟”寄存器。访问这些寄存器实际上访问的是与之相关的一对 FSR 寄存器，并对其所存储的值进行特定的操作。这些寄存器是：

- POSTDEC：访问 FSR 的值，然后将它自动减 1
- POSTINC：访问 FSR 的值，然后将它自动加 1
- PREINC：将 FSR 的值加 1，然后在操作中使用该值
- PLUSW：将 W 寄存器中的有符号值（从 -127 到 128）与 FSR 中的值相加，并在操作中使用得到的新值

在本文中，访问 INDF 寄存器使用相关 FSR 寄存器中的值（不会更改此值）。同样，访问 PLUSW 寄存器是将 W 寄存器中的值作为 FSR 的偏移量；该操作不会改变 W 或 FSR 中的值。访问其他虚拟寄存器均会更改 FSR 寄存器的值。

使用 POSTDEC、 POSTINC 和 PREINC 对 FSR 进行操作会影响整对寄存器：即， FSRnL 寄存器从 FFh 溢出到 00h 并向 FSRnH 寄存器进位。但这些操作的结果不会更改 STATUS 寄存器中的任何标志位（如 Z、N 和 OV 等）。

PLUSW 寄存器可用于在数据存储空间中实现变址寻址。通过操作 W 寄存器中的值，用户可以访问相对当前指针地址有固定偏移量的地址单元。在某些应用中，该功能可用于在数据存储器内部实现某些强大的程序控制结构，如软件堆栈。

6.4.3.3 通过 FSR 对其他 FSR 进行操作

在某些特殊情况下，间接寻址操作以其他 FSR 或虚拟寄存器作为目标。例如，使用 FSR 指向一个虚拟寄存器会导致操作不成功。假设如下特殊情况：FSR0H:FSR0L 寄存器保存的是 INDF1 的地址 FE7h。尝试使用 INDF0 作为操作数读取 INDF1 的值，将返回 00h。尝试使用 INDF0 作为操作数写入 INDF1，将会导致执行一条 NOP 指令。

另一方面，使用虚拟寄存器对一对 FSR 寄存器进行写操作可能会产生与预期不同的结果。在这些情况下，会将值写入一对 FSR 寄存器，但 FSR 不会递增或递减。因此，写入 INDF2 或 POSTDEC2 时会把同样的值写入 FSR2H:FSR2L 寄存器。

由于 FSR 是映射到 SFR 空间中的物理寄存器，所以可以通过所有直接寻址来操作它们。用户在使用这些寄存器时应该特别小心，尤其是在代码使用间接寻址时。

同样，通常允许通过间接寻址对所有其他 SFR 进行操作。用户在进行此类操作时应该特别小心，以免不小心更改设置从而影响器件操作。

6.5 程序存储器和扩展指令集

程序存储器的操作不受扩展指令集的影响。

使能扩展指令集会将 5 条额外的双字命令添加到现有的 PIC18 指令集中：即 ADDFSR、CALLW、MOVSF、MOVSS 和 SUBFSR。这些指令如**第 6.2.4 节“双字指令”**中所述执行。

6.6 数据存储器和扩展指令集

使能 PIC18 扩展指令集（XINST 配置位 = 1）显著改变了数据存储器及其寻址的某些方面。特别是，许多核心 PIC18 指令使用快速操作存储区的方式有所不同。这是由于扩展指令集引入了对数据存储空间的新的寻址模式。该模式还会更改使用 FSR2 及其相关操作数进行间接寻址的方式。

同样需要了解哪些部分保持不变。数据存储空间的大小及其线性寻址模式都不会改变。SFR 映射也保持不变。核心PIC18指令也仍然以直接和间接寻址模式进行操作；固有和立即数寻址指令操作照旧。FSR0 和 FSR1 的间接寻址模式也保持不变。

6.6.1 使用立即数偏移量进行变址寻址

使能 PIC18 扩展指令集将更改使用 FSR2 寄存器对及其相关文件操作数进行间接寻址的方式。在适当的条件下，使用快速操作存储区的指令（即绝大多数针对位和针对字节的指令）可以利用指令中的偏移量来执行变址寻址。这种特殊的寻址模式被称为使用立即数偏移量的变址寻址或立即数变址寻址模式。

使用扩展指令集时，这种寻址模式有如下要求：

- 强制使用快速操作存储区 （a = 0）；且
- 文件地址参数要小于或等于 5Fh。

在这些条件下，指令的文件地址不会被解析为地址的低字节（在直接寻址中和 BSR 一起使用），或快速操作存储区中的 8 位地址，而是被解析为由 FSR2 指定的地址指针的偏移量。将该偏移量与 FSR2 的内容相加以获取操作的目标地址。

6.6.2 受立即数变址寻址模式影响的指令

任何使用直接寻址模式的核心PIC18指令均会受到立即数变址寻址模式的潜在影响，包括所有针对字节和针对位的指令，或标准 PIC18 指令集中几乎一半的指令。只有使用固有或立即数寻址模式的指令不受影响。

此外，如果针对字节和针对位的指令不使用快速操作存储区 （快速操作 RAM 位为 1）或包含 60h 以上的文件地址，它们也不受影响。符合这些条件的指令会像以前一样执行。图 6-9 显示了当使能扩展指令集时，各种寻址模式之间的对比。

那些想要在立即数变址寻址模式中使用针对位或针对字节的指令的用户，应该注意此模式下汇编语法的改变。在**第 26.2.1 节 “ 扩展指令的语法 ”** 中对此进行了更详细的说明。

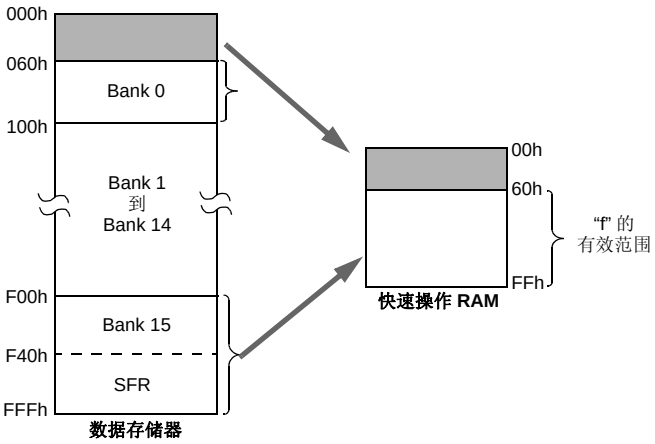
图 6-9：　　针对位和针对字节的指令的寻址模式对比 （使能了扩展指令集）

示例指令：ADDWF, f, d, a（操作码：0010 01da ffff ffff）

当 a = 0 且 f ≥ 60h 时：

此指令以直接强制模式执行。“f” 被解析为快速操作 RAM 中 060h 到 FFFh 之间的单元地址，该地址也是数据存储器的 F60h 到 FFFh （Bank 15）。

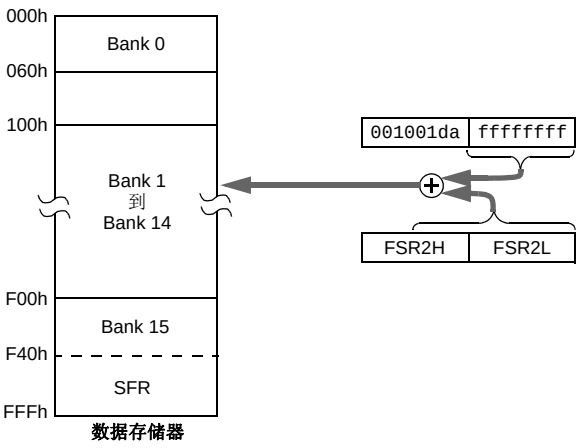
不可用此模式寻址地址低于 060h 的单元。



当 a = 0 且 f ≤ 5Fh 时：

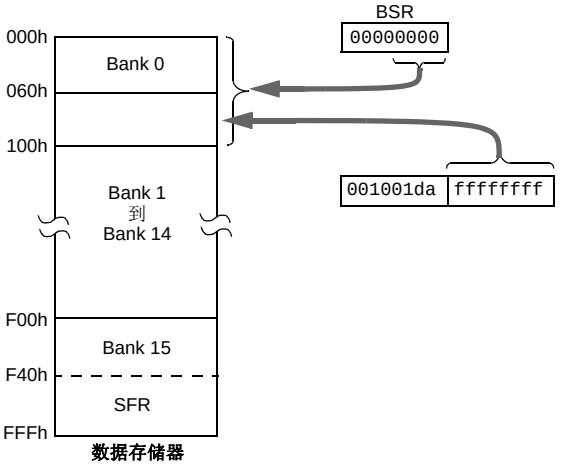
此指令以立即数变址寻址模式执行。 “f” 被解析为 FSR2 中地址值的偏移量。将这两个值相加可以得到指令的目标寄存器的地址。此地址可以在数据存储空间的任何地方。

注意在此模式中，正确的语法如下：
ADDWF [k], d
其中 “k” 就是 “f”。



当 a = 1（f 可为任何值）时：

指令以直接寻址模式 （也称为直接长地址寻址模式）执行。 “f” 被解析为数据存储空间的16个存储区之一中的一个单元地址。存储区由存储区选择寄存器 （BSR）指定。此地址可以位于数据存储空间的任何已实现存储区中。



6.6.3 在立即数变址模式下映射快速操作存储区

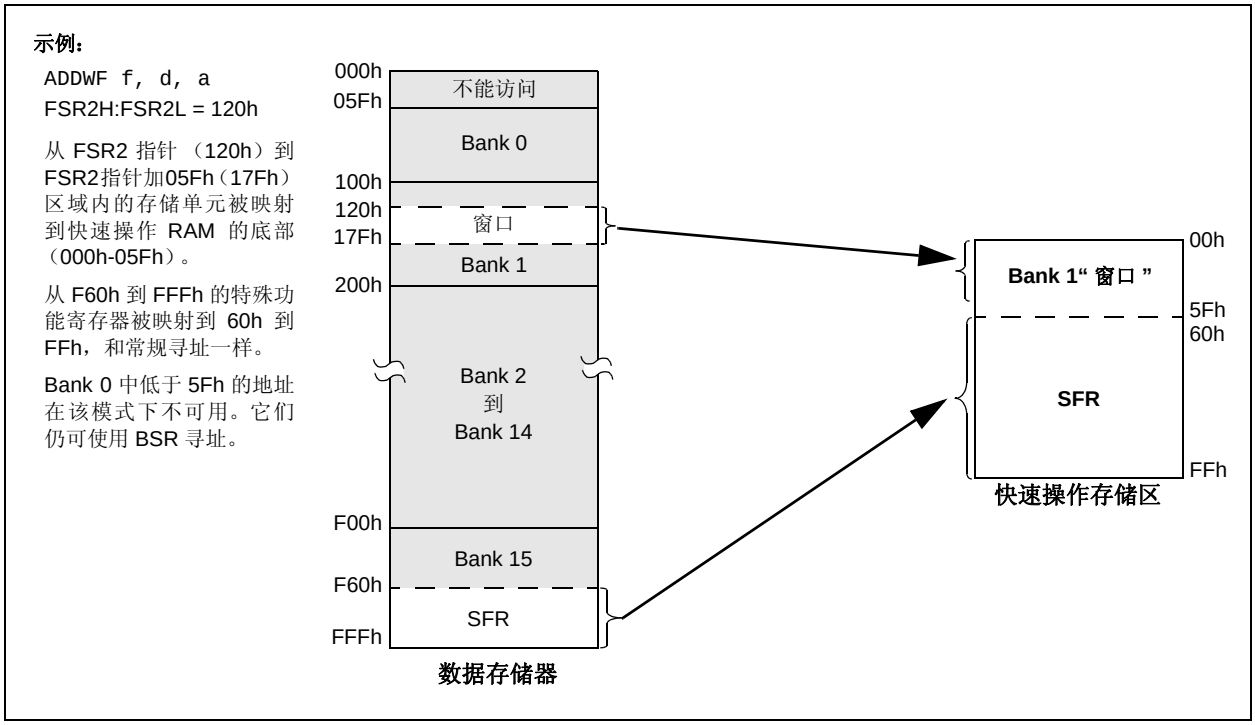
使用立即数变址寻址模式能有效改变快速操作RAM低地址单元（00h 到 5Fh）的映射方式。此模式映射 Bank 0 的内容和由用户定义的、可位于数据存储空间中任何地方的“窗口”内容，而不仅仅映射 Bank 0 底部的内容。FSR2 的值定义映射到窗口的地址的下边界，而上边界则由 FSR2 加 95（5Fh）决定。地址为 5Fh 以上的快速操作 RAM 的映射方法如前所述（见**第 6.3.2 节“快速操作存储区”**）。图 6-10 显示了在此寻址模式下重映射的快速操作存储区示例。

快速操作存储区的重映射*仅*适用于立即数变址寻址模式。使用 BSR（快速操作 RAM 位为 1）的操作和以前一样继续使用直接寻址模式。任何明确使用间接文件操作数（包括 FSR2）的间接或变址寻址操作都将像标准间接寻址一样操作。任何使用快速操作存储区（另外包括大于 05Fh 的寄存器地址）的指令都将使用直接寻址和常规的快速操作存储区映射。

6.6.4 立即数变址模式中的 BSR

尽管使能扩展指令集时会重映射快速操作存储区，但 BSR 的操作保持不变。使用 BSR 选择数据存储区的直接寻址操作方式和以前描述的相同。

图 6-10：　使用立即数变址寻址模式重映射快速操作存储区



注：

7.0 闪存程序存储器

在整个 VDD 范围内，正常操作期间，闪存程序存储器都是可读写、可擦除的。

读程序存储器时，每次读取一个字节。写程序存储器时，每次写一个 64 字节或 2 字节的块。擦除程序存储器时，每次擦除一个 1024 字节的块。用户代码不能执行批量擦除操作。

在擦写程序存储器时，系统会停止取指令直到操作完成。擦写期间不能访问程序存储器，因此也就无法执行代码。由内部编程定时器来终止程序存储器的擦写操作。

写入程序存储器的值不一定非要是有效指令。执行存储无效指令的程序存储单元会导致执行 NOP。

7.1 表读与表写

为了读写程序存储器，有两个操作可供处理器在程序存储空间和数据 RAM 之间传送字节：

- 表读 （TBLRD）
- 表写 （TBLWT）

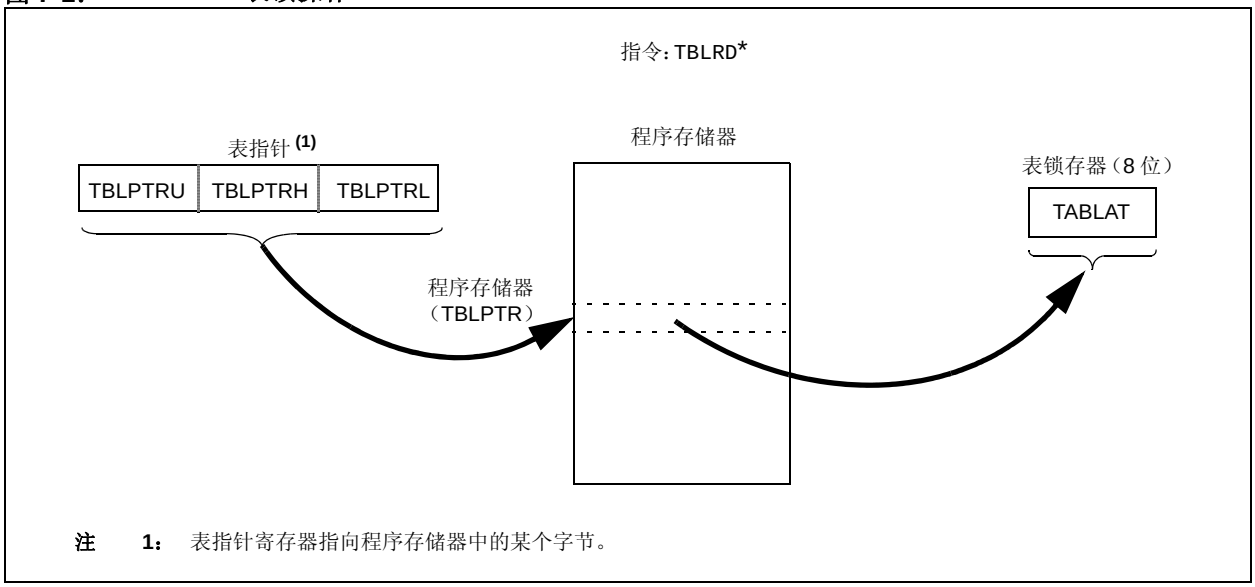
程序存储空间为 16 位宽，而数据 RAM 空间为 8 位宽。表读和表写操作通过一个 8 位寄存器 （TABLAT）在这两个存储空间之间传送数据。

表读操作从程序存储器获取数据并将其放入数据 RAM 空间。图 7-1 显示了程序存储器和数据 RAM 之间的一次表读操作。

表写操作将数据存储空间中的数据存储到程序存储器的保持寄存器中。**第 7.5 节 “ 写闪存程序存储器 ”** 详细介绍了将保持寄存器的内容写入程序存储器的过程。图7-2 显示了程序存储器和数据 RAM 之间的一次表写操作。

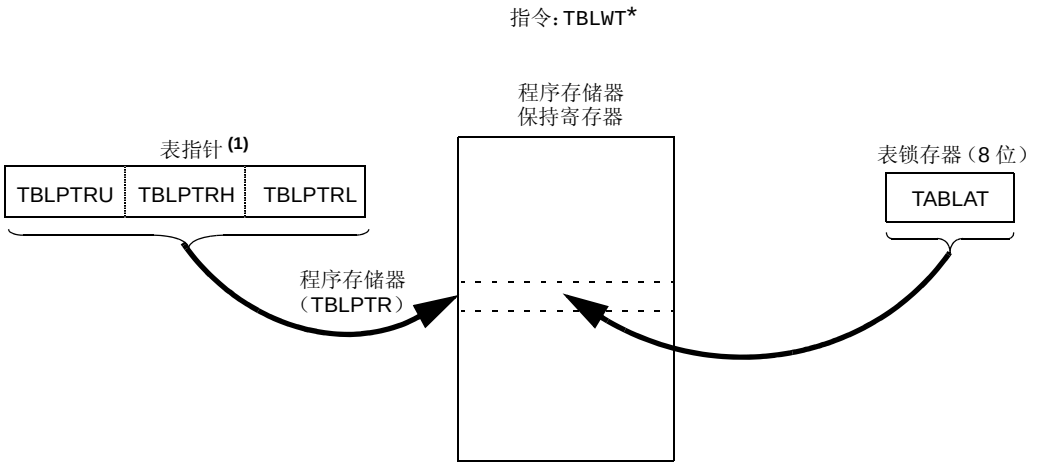
表操作以字节为单位。包含数据而非程序指令的表块不需要按字对齐。因此，表块可以在任何字节地址开始和结束。如果使用表写操作向程序存储器写入可执行代码，程序指令必须按字对齐。

图 7-1： 表读操作



注 1： 表指针寄存器指向程序存储器中的某个字节。

图 7-2：　　表写操作



注　1： 表指针实际指向 64 个保持寄存器之一，其地址由 TBLPTRL<5:0> 决定。**第 7.5 节 “ 写闪存程序存储器 ”** 中将讨论向程序存储器阵列写入数据的物理过程。

7.2　控制寄存器

TBLRD 和 TBLWT 指令要用到几个控制寄存器。这些寄存器包括：

- EECON1 寄存器
- EECON2 寄存器
- TABLAT 寄存器
- TBLPTR 寄存器

7.2.1　EECON1 和 EECON2 寄存器

EECON1 寄存器（寄存器 7-1）是存储器访问的控制寄存器。EECON2 寄存器不是实际存在的寄存器，专用于存储器的擦写操作。读 EECON2 将得到全 0。

如果 WPROG 位置 1，则执行 WR 命令时，允许用户按单个字 （2 个字节）编程。如果该位清零，则执行 WR 命令时，将编程 64 字节的块。

当 FREE 位置 1 时，允许对程序存储器进行擦除操作，擦除操作由下一条 WR 命令启动。当 FREE 清零时，则仅使能写操作。

当 WREN 位置 1 时，允许进行写操作。上电时，WREN 位被清零。WRERR 位在 WR 位置 1 时由硬件置 1，在内部编程定时器超时、写操作结束时被清零。

注：	在正常操作期间，WRERR 读为 1。这表明写操作被复位提早终止或进行了不合法的写操作。

WR 控制位用于启动写操作。用软件只能将该位置 1 而无法清零。在写操作完成后，由硬件将其清零。

寄存器 7-1：　　EECON1：EEPROM 控制寄存器 1

U-0	U-0	R/W-0	R/W-0	R/W-x	R/W-0	R/S-0	U-0
—	—	WPROG	FREE	WRERR(1)	WREN	WR	—
bit 7							bit 0

图注：	S = 可置 1 位 （无法用软件清零）		
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 7-6　**未实现：**读为 0

bit 5　**WPROG：**单字宽编程位
1 = 在执行下一条 WR 命令时编程 2 个字节
0 = 在执行下一条 WR 命令时编程 64 个字节

bit 4　**FREE：**闪存擦除使能位
1 = 在下一条 WR 命令时执行擦除操作 （擦除操作完成后清零）
0 = 仅执行写操作

bit 3　**WRERR：**闪存程序存储器错误标志位 (1)
1 = 写操作提早终止 （由于正常操作中自定时编程期间的任何复位，或不合法的写操作）
0 = 写操作完成

bit 2　**WREN：**闪存程序存储器写使能位
1 = 允许对闪存程序存储器的写周期
0 = 禁止对闪存程序存储器的写周期

bit 1　**WR：**写控制位
1 = 启动程序存储器的擦写周期
（操作是自定时的，一旦写操作完成，该位即由硬件清零。用软件只能将 WR 位置 1（不能清零）。）
0 = 写周期完成

bit 0　**未实现：**读为 0

注　1： 当发生 WRERR 错误时， EEPGD 和 CFGS 位不会被清零。这样可以跟踪错误条件。

7.2.2 表锁存寄存器 （TABLAT）

表锁存器（TABLAT）是映射到 SFR 空间的一个 8 位寄存器。表锁存器用于在程序存储器和数据 RAM 之间传输数据时保存 8 位数据。

7.2.3 表指针寄存器 （TBLPTR）

表指针 （TBLPTR）寄存器在程序存储器中以字节为单位进行寻址。TBLPTR 由 3 个 SFR 寄存器组成：表指针最高字节、表指针高字节和表指针低字节 （TBLPTRU:TBLPTRH:TBLPTRL）。这 3 个寄存器合起来组成一个 22 位宽的指针。其中低 21 位允许器件寻址最大 2 MB 程序存储空间。第 22 位则允许访问器件 ID、用户 ID 和配置位。

TBLRD 和 TBLWT 指令要使用表指针寄存器 TBLPTR。这些指令可以基于表操作以 4 种方法之一更新 TBLPTR。表 7-1 列出了这些操作。这些操作只会影响 TBLPTR 的低 21 位。

7.2.4 表指针边界

TBLPTR 用于读、写和擦除闪存程序存储器。

当执行 TBLRD 时，TBLPTR 的所有 22 位决定将程序存储器的哪个字节读入 TABLAT。

当执行 TBLWT 时，表指针寄存器的低 7 位 （TBLPTR<6:0>）决定要写入程序存储器的哪个保持寄存器 （共有 64 个）。当程序存储器的定时写入 （通过 WR 位）开始时，TBLPTR 的高 12 位（TBLPTR<21:10>）将决定要写入哪个程序存储器块（每块 1024 字节）。更多详细信息，请参见**第 7.5 节 “ 写闪存程序存储器 ”**。

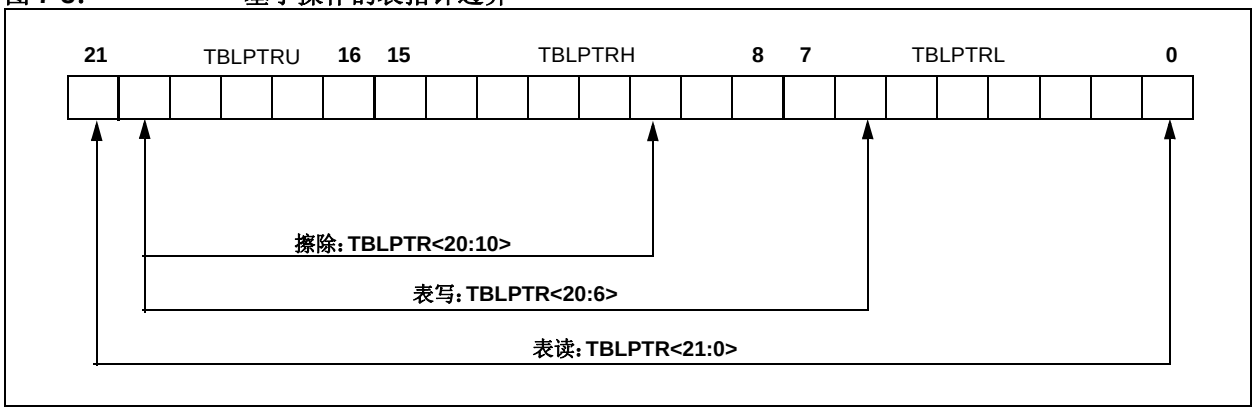
当执行擦除程序存储器时，表指针寄存器的高 12 位指向将要擦除的 1024 字节块。最低有效位被忽略。

图 7-3 说明了基于闪存程序存储器操作的 TBLPTR 相关边界。

表 7-1： 执行 TBLRD 和 TBLWT 指令的表指针操作

示例	表指针操作
TBLRD* TBLWT*	不修改 TBLPTR
TBLRD*+ TBLWT*+	TBLPTR 在读 / 写后递增
TBLRD*- TBLWT*-	TBLPTR 在读 / 写后递减
TBLRD+* TBLWT+*	TBLPTR 在读 / 写前递增

图 7-3： 基于操作的表指针边界



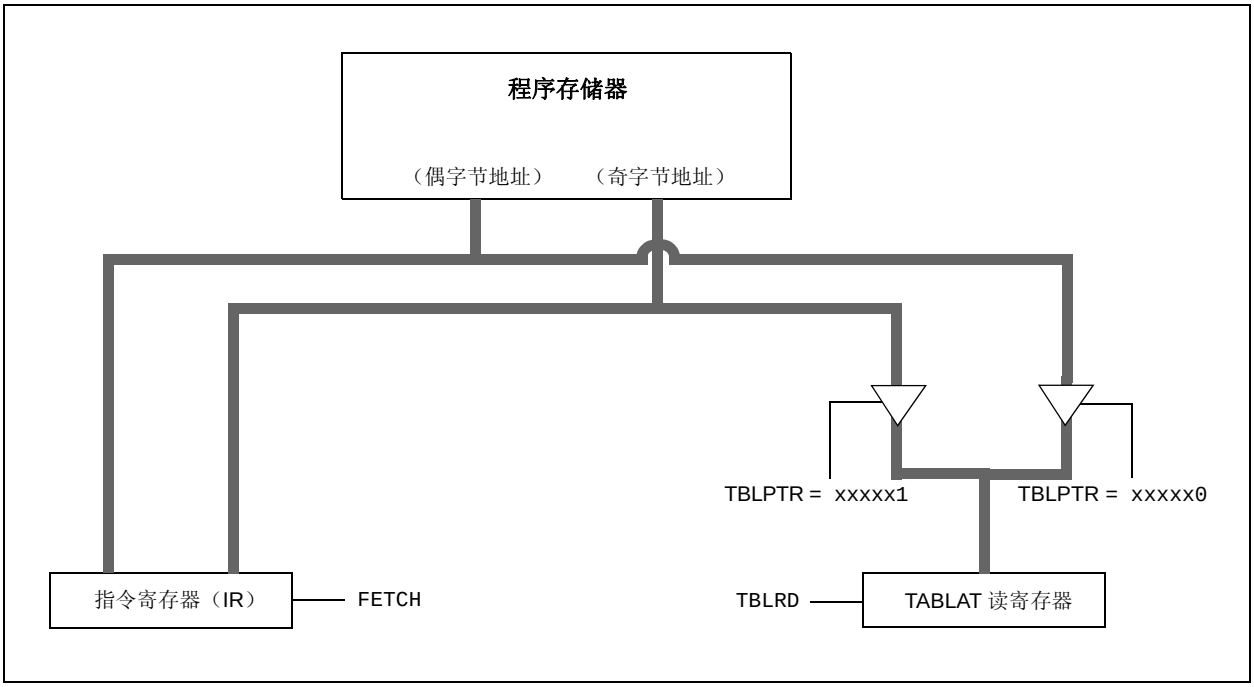
7.3 读闪存程序存储器

TBLRD 指令用于从程序存储器读取数据并放入数据 RAM。表读操作每次从程序存储器读取一个字节。

TBLPTR 指向程序存储空间的某个字节地址。执行 TBLRD 指令将把指向的字节装入 TABLAT。此外，还可以自动修改 TBLPTR 以进行下一次表读操作。

内部程序存储器通常以字为单位进行组织。由地址的最低有效位来选择字的高字节或低字节。图 7-4 显示了内部程序存储器和 TABLAT 之间的接口。

图 7-4：　　读闪存程序存储器



例 7-1：　　读闪存程序存储器的一个字

```
            MOVLW     CODE_ADDR_UPPER          ; Load TBLPTR with the base
            MOVWF     TBLPTRU                  ; address of the word
            MOVLW     CODE_ADDR_HIGH
            MOVWF     TBLPTRH
            MOVLW     CODE_ADDR_LOW
            MOVWF     TBLPTRL
READ_WORD
            TBLRD*+                            ; read into TABLAT and increment
            MOVF      TABLAT, W                ; get data
            MOVWF     WORD_EVEN
            TBLRD*+                            ; read into TABLAT and increment
            MOVF      TABLAT, W                ; get data
            MOVWF     WORD_ODD
```

7.4 擦除闪存程序存储器

最小擦除块大小为 512 个字或 1024 字节。只有通过使用外部编程器，或通过 ICSP 控制，才能够批量擦除更大的程序存储器块。闪存阵列不支持字擦除。

当单片机自身启动一个擦除序列时，会擦除一个 1024 字节的程序存储器块。 TBLPTR<21:10> 的高 12 位指向要擦除的块。 TBLPTR<9:0> 位被忽略。

擦除操作由 EECON1 寄存器控制。WREN 位必须被置 1 以使能写操作。FREE 位被置 1 以选择擦除操作。为了安全起见，必须使用 EECON2 的写启动序列。

擦除内部闪存必须执行长写操作。在长写周期中，指令暂停执行。由内部编程定时器终止长写操作。

7.4.1 闪存程序存储器擦除序列

擦除内部程序存储器块的过程如下：

1. 将要擦除的地址装入表指针寄存器。
2. 将 WREN 和 FREE 位（EECON1<2,4>）置 1 以使能擦除操作。
3. 禁止中断。
4. 将 55h 写入 EECON2。
5. 将 0AAh 写入 EECON2。
6. 将 WR 位置 1。这将开始擦除周期。
7. CPU 在擦除期间（Tiw，见参数 D133B）将会停止工作。
8. 重新允许中断。

例 7-2： 擦除闪存程序存储器

```
                MOVLW   CODE_ADDR_UPPER       ; load TBLPTR with the base
                MOVWF   TBLPTRU               ; address of the memory block
                MOVLW   CODE_ADDR_HIGH
                MOVWF   TBLPTRH
                MOVLW   CODE_ADDR_LOW
                MOVWF   TBLPTRL
        ERASE
                BSF     EECON1, WREN
                BSF     EECON1, FREE          ; enable Erase operation
                BCF     INTCON, GIE           ; disable interrupts
                MOVLW   55h
必需的序列      MOVWF   EECON2                ; write 55h
                MOVLW   0AAh
                MOVWF   EECON2                ; write 0AAh
                BSF     EECON1, WR            ; start erase (CPU stall)
                BSF     INTCON, GIE           ; re-enable interrupts
```

7.5 写闪存程序存储器

编程块大小为 32 个字或 64 字节。还支持一次编程一个字或 2 字节。

在内部使用表写指令将需要写入闪存的内容装入保持寄存器中。表写操作使用 64 个保持寄存器进行编程。

由于表锁存器（TABLAT）只是单字节寄存器，所以每次编程操作 TBLWT 指令可能需要执行 64 次（如果 WPROG = 0）。因为只写保持寄存器，所以所有的表写操作实际上都是短写。更新 64 个保持寄存器后，必须写 EECON1 寄存器，以便启动长写周期开始编程操作。

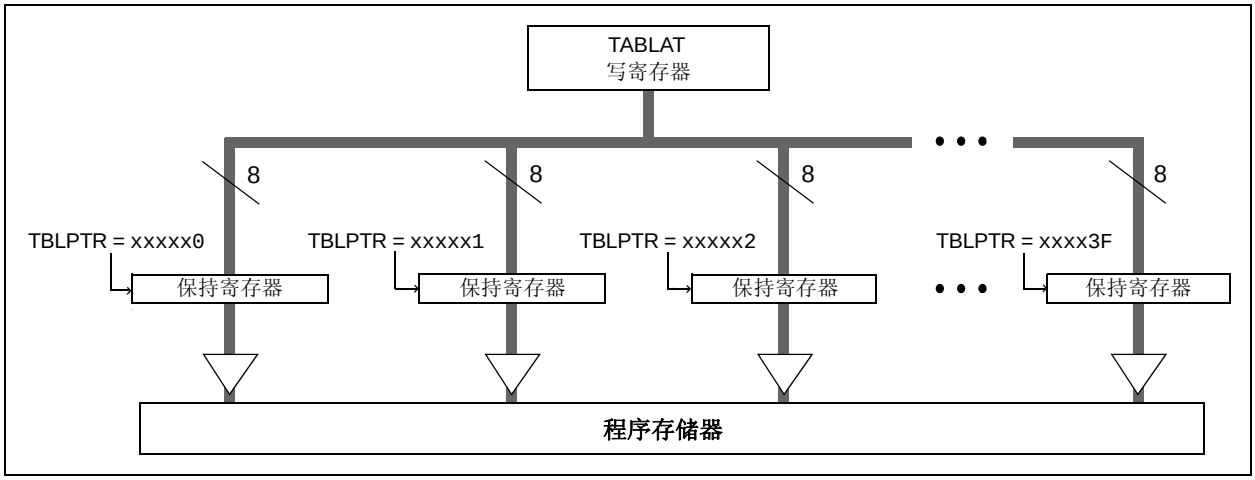
对内部闪存编程要求使用长写操作。在长写周期中，指令暂停执行。由内部编程定时器终止长写操作。

由片上定时器控制写入的时间。写入 / 擦除电压由片上的电荷泵产生，该电荷泵可以在器件的电压范围内工作。

注 1： 与早期的PIC18闪存器件不同，PIC18F87J90 系列器件在写操作发生后并不会复位保持寄存器。必须在编程序列之前清零或改写保持寄存器。

2： 为了确保程序存储单元的耐擦写次数，在两次擦除操作之间不应对每个闪存字节编程多于一次。在第二次尝试修改目标单元内容之前，必须对其所在单元执行擦除操作或整个存储器的批量擦除。

图 7-5： 对闪存程序存储器的表写操作



7.5.1 闪存程序存储器写序列

对内部程序存储单元编程的过程如下：

1. 将 1024 字节读入 RAM。
2. 必要时更新 RAM 中的数据值。
3. 将要擦除的地址装入表指针寄存器。
4. 执行擦除过程。
5. 将要写入的第一个字节的地址装入表指针寄存器，表指针递减 1。
6. 通过自动递增将 64 个字节写入保持寄存器。
7. 将 WREN 位（EECON1<2>）置 1 以使能字节写操作。
8. 禁止中断。
9. 将 55h 写入 EECON2。
10. 将 0AAh 写入 EECON2。
11. 将 WR 位置 1。这将开始写周期。
12. CPU 在写入期间（Tiw，参数 D133A）将会停止工作。
13. 重新允许中断。
14. 重复步骤 6 到 13，直到将所有 1024 个字节都写入程序存储器中。
15. 校验存储器（表读）。

下页中的例 7-3 给出了所需代码的示例。

注： 在将 WR 位置 1 前，表指针地址必须处于保持寄存器中的 64 字节预期地址范围内。

例 7-3：　写闪存程序存储器

```
                    MOVLW   CODE_ADDR_UPPER     ; Load TBLPTR with the base address
                    MOVWF   TBLPTRU             ; of the memory block, minus 1
                    MOVLW   CODE_ADDR_HIGH
                    MOVWF   TBLPTRH
                    MOVLW   CODE_ADDR_LOW
                    MOVWF   TBLPTRL
ERASE_BLOCK
                    BSF     EECON1, WREN        ; enable write to memory
                    BSF     EECON1, FREE        ; enable Erase operation
                    BCF     INTCON, GIE         ; disable interrupts
                    MOVLW   55h
                    MOVWF   EECON2              ; write 55h
                    MOVLW   0AAh
                    MOVWF   EECON2              ; write 0AAh
                    BSF     EECON1, WR          ; start erase (CPU stall)
                    BSF     INTCON, GIE         ; re-enable interrupts
                    MOVLW   D'16'
                    MOVWF   WRITE_COUNTER       ; Need to write 16 blocks of 64 to write
                                                ; one erase block of 1024
RESTART_BUFFER
                    MOVLW   D'64'
                    MOVWF   COUNTER
                    MOVLW   BUFFER_ADDR_HIGH    ; point to buffer
                    MOVWF   FSR0H
                    MOVLW   BUFFER_ADDR_LOW
                    MOVWF   FSR0L
FILL_BUFFER
                    ...                         ; read the new data from I2C, SPI,
                                                ; USART, etc.
WRITE_BUFFER
                    MOVLW   D’64                ; number of bytes in holding register
                    MOVWF   COUNTER
WRITE_BYTE_TO_HREGS
                    MOVFF   POSTINC0, WREG      ; get low byte of buffer data
                    MOVWF   TABLAT              ; present data to table latch
                    TBLWT+*                     ; write data, perform a short write
                                                ; to internal TBLWT holding register.
                    DECFSZ  COUNTER             ; loop until buffers are full
                    BRA     WRITE_BYTE_TO_HREGS
PROGRAM_MEMORY
                    BSF     EECON1, WREN        ; enable write to memory
                    BCF     INTCON, GIE         ; disable interrupts
                    MOVLW   55h
                    MOVWF   EECON2              ; write 55h
    必需的序列      MOVLW   0AAh
                    MOVWF   EECON2              ; write 0AAh
                    BSF     EECON1, WR          ; start program (CPU stall)
                    BSF     INTCON, GIE         ; re-enable interrupts
                    BCF     EECON1, WREN        ; disable write to memory

                    DECFSZ  WRITE_COUNTER       ; done with one write cycle
                    BRA     RESTART_BUFFER      ; if not done replacing the erase block
```

7.5.2 闪存程序存储器写序列 （字编程）

PIC18F87J90系列器件具有允许编程单个字（2个字节）的功能。当 WPROG 位置 1 时，该功能使能。如果存储单元已经被擦除，需要通过以下序列来使能该功能：

1. 向表指针寄存器中装入要写入的数据的地址。
2. 将 2 个字节写入保持寄存器并执行表写指令。
3. 将 WPROG 置 1 以使能单字写操作。
4. 将 WREN 置 1 以使能写存储器。
5. 禁止中断。
6. 将 55h 写入 EECON2。
7. 将 0AAh 写入 EECON2。
8. 将 WR 位置 1。这将开始写周期。
9. CPU 在写入期间（Tiw，见参数 D133A）将会停止工作。
10. 重新允许中断。

例 7-4： 单字写闪存程序存储器

```
                MOVLW    CODE_ADDR_UPPER      ; Load TBLPTR with the base address
                MOVWF    TBLPTRU
                MOVLW    CODE_ADDR_HIGH
                MOVWF    TBLPTRH
                MOVLW    CODE_ADDR_LOW
                MOVWF    TBLPTRL

                MOVLW    DATA0
                MOVWF    TABLAT
                TBLWT*+
                MOVLW    DATA1
                MOVWF    TABLAT
                TBLWT*


PROGRAM_MEMORY
                BSF     EECON1, WPROG        ; enable single word write
                BSF     EECON1, WREN         ; enable write to memory
                BCF     INTCON, GIE          ; disable interrupts
                MOVLW   55h
必需的序列      MOVWF   EECON2               ; write 55h
                MOVLW   0AAh
                MOVWF   EECON2               ; write 0AAh
                BSF     EECON1, WR           ; start program (CPU stall)
                BSF     INTCON, GIE          ; re-enable interrupts
                BCF     EECON1, WPROG        ; disable single word write
                BCF     EECON1, WREN         ; disable write to memory
```

7.5.3 写校验

根据具体应用，将写入存储器的值对照原始值进行校验是一个很好的编程习惯。在应用中，如果某些位的写次数接近规定极限值，就应该进行写校验。

7.5.4 意外终止写操作

如果由于意外事件（如掉电或意外复位）终止了写操作，应该对刚刚编程的存储单元进行校验，如有必要，还要重新进行编程。如果写操作在正常操作期间因 $\overline{\text{MCLR}}$ 复位或 WDT 超时复位而中断，用户可以检查 WRERR 位以确定是否需要重写该单元。

7.6 代码保护期间闪存程序存储器的操作

关于闪存程序存储器代码保护的详细信息，请参见 **第 25.6 节 “ 程序校验和代码保护 ”**。

表 7-2： 与闪存程序存储器相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
TBLPTRU	—	—	bit 21	程序存储器表指针最高字节 （TBLPTR<20:16>）					59
TBPLTRH	程序存储器表指针高字节 （TBLPTR<15:8>）								59
TBLPTRL	程序存储器表指针低字节 （TBLPTR<7:0>）								59
TABLAT	程序存储器表锁存器								59
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
EECON2	EEPROM 控制寄存器 2 （不是实际存在的寄存器）								61
EECON1	—	—	WPROG	FREE	WRERR	WREN	WR	—	61

图注： — = 未实现，读为 0。闪存程序存储器访问期间不使用阴影单元。

8.0 8 x 8 硬件乘法器

8.1 简介

所有 PIC18 器件均包含一个 8 x 8 硬件乘法器（是 ALU 的一部分）。该乘法器可执行无符号运算并产生一个16位运算结果，该结果存储在一对乘积寄存器PRODH:PRODL 中。该乘法器执行的运算不会影响 STATUS 寄存器中的任何标志。

通过硬件执行乘法运算只需要一个指令周期。硬件乘法器具有更高的计算吞吐量并缩短了乘法算法的代码长度，从而可在许多先前仅能使用数字信号处理器的应用中使用 PIC18 器件。表 8-1 给出了硬件和软件乘法运算的比较，包括所需存储器空间和执行时间。

8.2 工作原理

例 8-1 给出了一个 8 x 8 无符号乘法运算的指令序列。当已在 WREG 寄存器中装入了一个参数时，实现该运算仅需一条指令。

例 8-2 给出了一个 8 x 8 有符号乘法运算的指令序列。要弄清参数的符号位，必须检查每个参数对的最高有效位（Most Significant bit， MSb），并做相应的减法。

例 8-1： 8 x 8 无符号乘法程序

```
MOVF    ARG1, W     ;
MULWF   ARG2        ; ARG1 * ARG2 ->
                    ; PRODH:PRODL
```

例 8-2： 8 x 8 有符号乘法程序

```
MOVF    ARG1, W
MULWF   ARG2        ; ARG1 * ARG2 ->
                    ; PRODH:PRODL
BTFSC   ARG2, SB    ; Test Sign Bit
SUBWF   PRODH, F    ; PRODH = PRODH
                    ;         - ARG1
MOVF    ARG2, W
BTFSC   ARG1, SB    ; Test Sign Bit
SUBWF   PRODH, F    ; PRODH = PRODH
                    ;         - ARG2
```

表 8-1： 各种乘法运算的性能比较

程序	乘法实现方法	程序存储器（字）	周期数（最大）	时间		
				48 MHz 时	10 MHz 时	4 MHz 时
8 x 8 无符号	无硬件乘法	13	69	5.7 µs	27.6 µs	69 µs
	硬件乘法	1	1	83.3 ns	400 ns	1 µs
8 x 8 有符号	无硬件乘法	33	91	7.5 µs	36.4 µs	91 µs
	硬件乘法	6	6	500 ns	2.4 µs	6 µs
16 x 16 无符号	无硬件乘法	21	242	20.1 µs	96.8 µs	242 µs
	硬件乘法	28	28	2.3 µs	11.2 µs	28 µs
16 x 16 有符号	无硬件乘法	52	254	21.6 µs	102.6 µs	254 µs
	硬件乘法	35	40	3.3 µs	16.0 µs	40 µs

例 8-3 给出了一个 16 x 16 无符号乘法运算的指令序列。公式 8-1 为所使用的算法。32 位结果存储在 4 个寄存器（RES3:RES0）中。

公式 8-1： 16 x 16 无符号乘法算法

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L)
\end{aligned}
$$

例 8-3： 16 x 16 无符号乘法程序

```
    MOVF    ARG1L, W
    MULWF   ARG2L           ; ARG1L * ARG2L ->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES1     ;
    MOVFF   PRODL, RES0     ;
;
    MOVF    ARG1H, W
    MULWF   ARG2H           ; ARG1H * ARG2H ->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES3     ;
    MOVFF   PRODL, RES2     ;
;
    MOVF    ARG1L, W
    MULWF   ARG2H           ; ARG1L * ARG2H ->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
;
    MOVF    ARG1H, W        ;
    MULWF   ARG2L           ; ARG1H * ARG2L ->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
```

例 8-4 给出了一个 16 x 16 有符号乘法运算的指令序列。公式 8-2 为所使用的算法。32 位结果存储在 4 个寄存器（RES3:RES0）中。要弄清参数的符号位，必须检查每个参数对的 MSb，并做相应的减法。

公式 8-2： 16 x 16 有符号乘法算法

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L) + \\
&\quad (-1 \bullet ARG2H<7> \bullet ARG1H:ARG1L \bullet 2^{16}) + \\
&\quad (-1 \bullet ARG1H<7> \bullet ARG2H:ARG2L \bullet 2^{16})
\end{aligned}
$$

例 8-4： 16 x 16 有符号乘法程序

```
   MOVF    ARG1L, W
   MULWF   ARG2L           ; ARG1L * ARG2L ->
                           ; PRODH:PRODL
   MOVFF   PRODH, RES1     ;
   MOVFF   PRODL, RES0     ;
;
   MOVF    ARG1H, W
   MULWF   ARG2H           ; ARG1H * ARG2H ->
                           ; PRODH:PRODL
   MOVFF   PRODH, RES3     ;
   MOVFF   PRODL, RES2     ;
;
   MOVF    ARG1L, W
   MULWF   ARG2H           ; ARG1L * ARG2H ->
                           ; PRODH:PRODL
   MOVF    PRODL, W        ;
   ADDWF   RES1, F         ; Add cross
   MOVF    PRODH, W        ; products
   ADDWFC  RES2, F         ;
   CLRF    WREG            ;
   ADDWFC  RES3, F         ;
;
   MOVF    ARG1H, W        ;
   MULWF   ARG2L           ; ARG1H * ARG2L ->
                           ; PRODH:PRODL
   MOVF    PRODL, W        ;
   ADDWF   RES1, F         ; Add cross
   MOVF    PRODH, W        ; products
   ADDWFC  RES2, F         ;
   CLRF    WREG            ;
   ADDWFC  RES3, F         ;
;
   BTFSS   ARG2H, 7        ; ARG2H:ARG2L neg?
   BRA     SIGN_ARG1       ; no, check ARG1
   MOVF    ARG1L, W        ;
   SUBWF   RES2            ;
   MOVF    ARG1H, W        ;
   SUBWFB  RES3
;
SIGN_ARG1
   BTFSS   ARG1H, 7        ; ARG1H:ARG1L neg?
   BRA     CONT_CODE       ; no, done
   MOVF    ARG2L, W        ;
   SUBWF   RES2            ;
   MOVF    ARG2H, W        ;
   SUBWFB  RES3
;
CONT_CODE
   :
```

9.0 中断

PIC18F87J90 系列器件具有多个中断源及中断优先级功能，该功能可以给大多数中断源分配高优先级或者低优先级。高优先级中断向量位于 0008h，低优先级中断向量位于 0018h。高优先级中断事件可以中断正在处理的低优先级中断。

有 13 个寄存器用于控制中断操作。这些寄存器是：

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1、 PIR2 和 PIR3
- PIE1、 PIE2 和 PIE3
- IPR1、 IPR2 和 IPR3

建议使用 MPLAB® IDE 提供的 Microchip 头文件命名这些寄存器中的位。这使得汇编器 / 编译器能够自动存放指定寄存器中的这些位。

通常，中断源有 3 个位用于控制其操作。这些位分别是：

- **标志位**表明发生了中断事件
- **允许位**允许程序跳转到中断向量地址处执行 （当标志位置 1 时）
- **优先级位**用于选择高优先级还是低优先级

通过将 IPEN 位 （RCON<7>）置 1，可使能中断优先级功能。当使能中断优先级时，有2个全局中断允许位。将 GIEH 位 （INTCON<7>）置 1，可允许所有优先级位置 1（高优先级）的中断。将 GIEL 位（INTCON<6>）置 1，可允许所有优先级位清零 （低优先级）的中断。当中断标志位、允许位及相应的全局中断允许位均被置 1 时，中断将根据优先级位的设置立即跳转到地址 0008h 或 0018h。也可以通过设置相应的中断允许位来禁止单个中断。

当 IPEN 位清零 （默认状态）时，便会禁止中断优先级功能，此时中断是与 PIC® 中档器件兼容的。在兼容模式下，各个中断源的中断优先级位不起作用。INTCON<6> 是 PEIE 位，用于允许 / 禁止所有的外设中断源。 INTCON<7> 是 GIE 位，用于允许 / 禁止所有中断源。在兼容模式下，所有中断均跳转到地址 0008h。

当响应中断时，全局中断允许位被清零以禁止后续中断。清零后的 IPEN 位就是 GIE 位。如果使用了中断优先级，这个位就是 GIEH 位或者 GIEL 位。高优先级中断源会中断低优先级中断。在处理高优先级中断时，低优先级中断将不被处理。

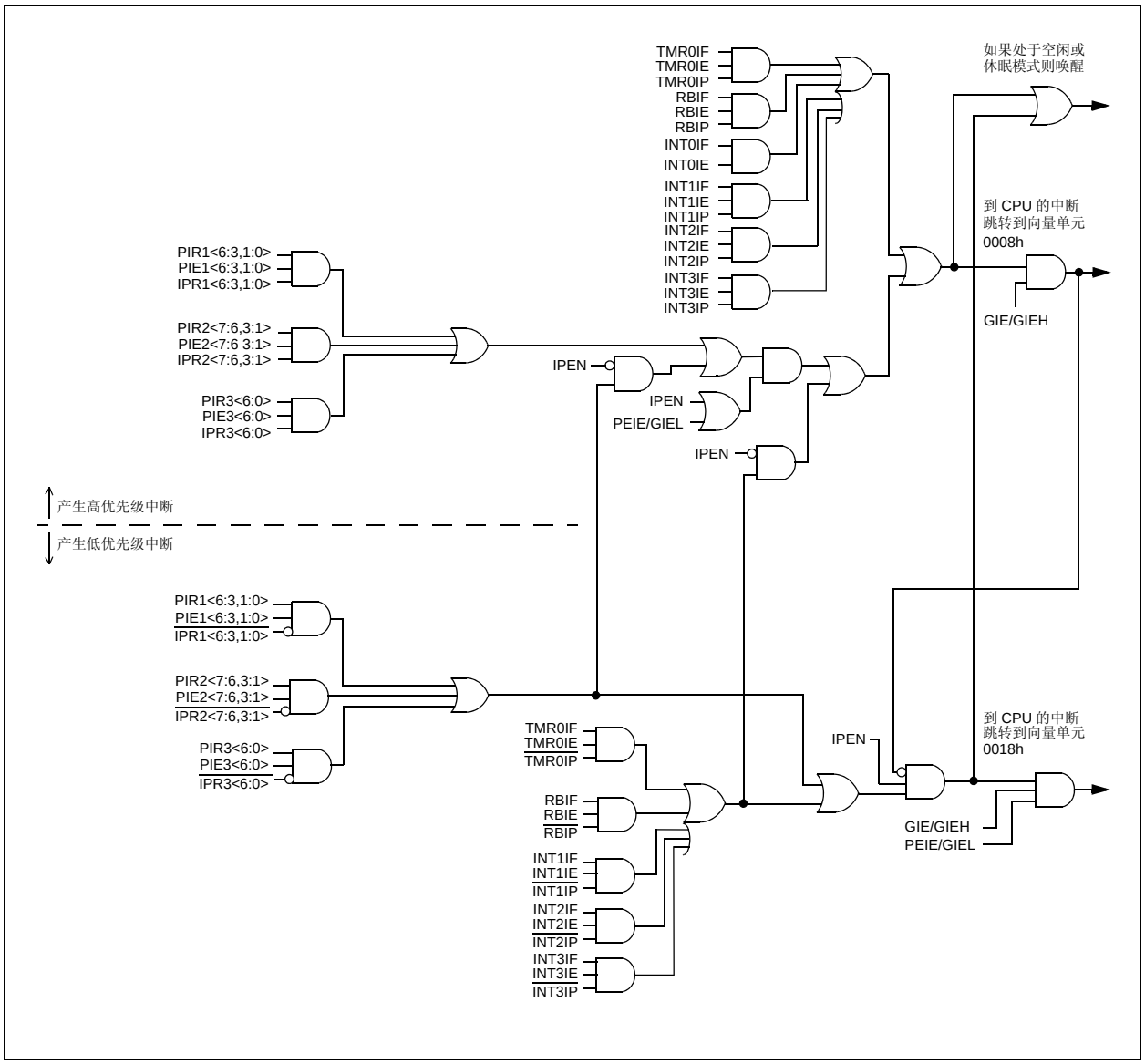
返回地址被压入堆栈，中断向量地址（0008h 或 0018h）被装入 PC。只要在中断服务程序中，就可以通过查询中断标志位来确定中断源。在重新允许中断前，必须用软件将中断标志位清零，以避免中断递归。

执行 “ 从中断返回 ” 指令 RETFIE 将退出中断程序，同时将 GIE 位（若使用中断优先级则为 GIEH 或 GIEL 位）置 1，从而重新允许中断。

对于外部中断事件，例如 INTx 引脚中断或者 PORTB 输入电平变化中断，中断响应延时将会是 3 到 4 个指令周期。对于单周期或双周期指令，中断响应延时完全相同。各中断标志位的置 1 不受对应的中断允许位和 GIE 位状态的影响。

注：	当允许**任何**中断时，不要使用 MOVFF 指令修改中断控制寄存器。否则可能导致单片机操作出错。

图 9-1：　　PIC18F87J90 系列中断逻辑



9.1 INTCON 寄存器

INTCON 寄存器是可读写的寄存器，包含各个中断允许位、优先级位和标志位。

注： 当中断条件产生时，不管相应的中断允许位或全局中断允许位的状态如何，中断标志位都将置 1。用户软件应在允许一个中断前，先将相应的中断标志位清零。这样做允许用软件查询中断标志位。

寄存器 9-1： INTCON：中断控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF(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 **GIE/GIEH**：全局中断允许位
当 IPEN = 0 时：
1 = 允许所有未被屏蔽的中断
0 = 禁止所有中断
当 IPEN = 1 时：
1 = 允许所有高优先级中断
0 = 禁止所有中断

bit 6 **PEIE/GIEL**：外设中断允许位
当 IPEN = 0 时：
1 = 允许所有未被屏蔽的外设中断
0 = 禁止所有外设中断
当 IPEN = 1 时：
1 = 允许所有低优先级的外设中断
0 = 禁止所有低优先级的外设中断

bit 5 **TMR0IE**：TMR0 溢出中断允许位
1 = 允许 TMR0 溢出中断
0 = 禁止 TMR0 溢出中断

bit 4 **INT0IE**：INT0 外部中断允许位
1 = 允许 INT0 外部中断
0 = 禁止 INT0 外部中断

bit 3 **RBIE**：RB 端口电平变化中断允许位
1 = 允许 RB 端口电平变化中断
0 = 禁止 RB 端口电平变化中断

bit 2 **TMR0IF**：TMR0 溢出中断标志位
1 = TMR0 寄存器已溢出 （必须用软件清零）
0 = TMR0 寄存器未溢出

bit 1 **INT0IF**：INT0 外部中断标志位
1 = 发生了 INT0 外部中断 （必须用软件清零）
0 = 未发生 INT0 外部中断

bit 0 **RBIF**：RB 端口电平变化中断标志位 (1)
1 = RB<7:4> 引脚中至少有一个引脚的电平状态发生了改变 （必须用软件清零）
0 = 所有 RB<7:4> 引脚的电平状态都没有改变

注 1： 不匹配条件将继续把该位置 1。读取 PORTB，然后额外等待一个指令周期，将结束不匹配条件，并允许将该位清零。

寄存器 9-2：　　INTCON2：中断控制寄存器 2

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP
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　　**$\overline{\text{RBPU}}$**：PORTB 上拉使能位
1 = 禁止所有 PORTB 上拉
0 = 根据各端口锁存值使能 PORTB 上拉

bit 6　　**INTEDG0**：外部中断 0 边沿选择位
1 = 上升沿触发中断
0 = 下降沿触发中断

bit 5　　**INTEDG1**：外部中断 1 边沿选择位
1 = 上升沿触发中断
0 = 下降沿触发中断

bit 4　　**INTEDG2**：外部中断 2 边沿选择位
1 = 上升沿触发中断
0 = 下降沿触发中断

bit 3　　**INTEDG3**：外部中断 3 边沿选择位
1 = 上升沿触发中断
0 = 下降沿触发中断

bit 2　　**TMR0IP**：TMR0 溢出中断优先级位
1 = 高优先级
0 = 低优先级

bit 1　　**INT3IP**：INT3 外部中断优先级位
1 = 高优先级
0 = 低优先级

bit 0　　**RBIP**：RB 端口电平变化中断优先级位
1 = 高优先级
0 = 低优先级

注：　当中断条件产生时，不管相应的中断允许位或全局中断允许位的状态如何，中断标志位都将置 1。用户软件应在允许一个中断前，先将相应的中断标志位清零。这样做允许用软件查询中断标志位。

寄存器 9-3：　　INTCON3：中断控制寄存器 3

R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF
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　**INT2IP：**INT2 外部中断优先级位
1 = 高优先级
0 = 低优先级

bit 6　**INT1IP：**INT1 外部中断优先级位
1 = 高优先级
0 = 低优先级

bit 5　**INT3IE：**INT3 外部中断允许位
1 = 允许 INT3 外部中断
0 = 禁止 INT3 外部中断

bit 4　**INT2IE：**INT2 外部中断允许位
1 = 允许 INT2 外部中断
0 = 禁止 INT2 外部中断

bit 3　**INT1IE：**INT1 外部中断允许位
1 = 允许 INT1 外部中断
0 = 禁止 INT1 外部中断

bit 2　**INT3IF：**INT3 外部中断标志位
1 = 发生了 INT3 外部中断 （必须用软件清零）
0 = 未发生 INT3 外部中断

bit 1　**INT2IF：**INT2 外部中断标志位
1 = 发生了 INT2 外部中断 （必须用软件清零）
0 = 未发生 INT2 外部中断

bit 0　**INT1IF：**INT1 外部中断标志位
1 = 发生了 INT1 外部中断 （必须用软件清零）
0 = 未发生 INT1 外部中断

注：　当中断条件产生时，不管相应的中断允许位或全局中断允许位的状态如何，中断标志位都将置 1。用户软件应在允许一个中断前，先将相应的中断标志位清零。这样做允许用软件查询中断标志位。

9.2 PIR 寄存器

PIR 寄存器包含各外设中断的标志位。根据外设中断源的数量，有三个外设中断请求（标志）寄存器（PIR1、PIR2 和 PIR3）。

注 1： 当中断条件产生时，不管相应的中断允许位或全局中断允许位 GIE（INTCON<7>）的状态如何，中断标志位都将置 1。

2： 用户软件应在允许中断前和处理完中断后，将相应的中断标志位清零。

寄存器 9-4： PIR1：外设中断请求（标志）寄存器 1

U-0	R/W-0	R-0	R-0	R/W-0	U-0	R/W-0	R/W-0
—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF
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 **未实现：** 读为 0

bit 6 **ADIF：** A/D 转换器中断标志位
1 = A/D 转换已完成（必须用软件清零）
0 = A/D 转换未完成

bit 5 **RC1IF：** EUSART 接收中断标志位
1 = EUSART 接收缓冲区 RCREG1 已满（读取 RCREG1 时清零）
0 = EUSART 接收缓冲区为空

bit 4 **TX1IF：** EUSART 发送中断标志位
1 = EUSART 发送缓冲区 TXREG1 为空（写入 TXREG1 时清零）
0 = EUSART 发送缓冲区已满

bit 3 **SSPIF：** 主同步串口中断标志位
1 = 发送 / 接收已完成（必须用软件清零）
0 = 等待发送 / 接收

bit 2 **未实现：** 读为 0

bit 1 **TMR2IF：** TMR2 与 PR2 匹配中断标志位
1 = TMR2 与 PR2 发生匹配（必须用软件清零）
0 = TMR2 与 PR2 未发生匹配

bit 0 **TMR1IF：** TMR1 溢出中断标志位
1 = TMR1 寄存器已溢出（必须用软件清零）
0 = TMR1 寄存器未溢出

寄存器 9-5：　　PIR2：外设中断请求 （标志）寄存器 2

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	U-0
OSCFIF	CMIF	—	—	BCLIF	LVDIF	TMR3IF	—
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　**OSCFIF：**振荡器故障中断标志位
1 = 器件振荡器发生故障，改由 INTOSC 作为时钟输入 （必须用软件清零）
0 = 器件时钟正常工作

bit 6　**CMIF：**比较器中断标志位
1 = 比较器输入已改变 （必须用软件清零）
0 = 比较器输入未改变

bit 5-4　**未实现：**读为 0

bit 3　**BCLIF：**总线冲突中断标志位
1 = 发生了总线冲突 （必须用软件清零）
0 = 未发生总线冲突

bit 2　**LVDIF：**低压检测中断标志位
1 = 发生了低压条件 （必须用软件清零）
0 = 器件电压高于稳压器的低压跳变点

bit 1　**TMR3IF：**TMR3 溢出中断标志位
1 = TMR3 寄存器已溢出 （必须用软件清零）
0 = TMR3 寄存器未溢出

bit 0　**未实现：**读为 0

寄存器 9-6：　　PIR3：外设中断请求 （标志）寄存器 3

U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	LCDIF	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	RTCCIF
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　　**未实现：**读为 0

bit 6　　**LCDIF：**LCD 中断标志位 （当选择非静态模式下的 B 型波形时有效）
1 = 输出所有 COM 的 LCD 数据 （必须用软件清零）
0 = 尚未输出所有 COM 的 LCD 数据

bit 5　　**RC2IF：**AUSART 接收中断标志位
1 = AUSART 接收缓冲区 RCREG2 已满 （读取 RCREG2 时清零）
0 = AUSART 接收缓冲区为空

bit 4　　**TX2IF：**AUSART 发送中断标志位
1 = AUSART 发送缓冲区 TXREG2 为空 （写入 TXREG2 时清零）
0 = AUSART 发送缓冲区已满

bit 3　　**CTMUIF：**CTMU 中断标志位
1 = 发生了 CTMU 中断 （必须用软件清零）
0 = 未发生 CTMU 中断

bit 2　　**CCP2IF：**CCP2 中断标志位
捕捉模式：
1 = 发生了 TMR1/TMR3 寄存器捕捉 （必须用软件清零）
0 = 未发生 TMR1/TMR3 寄存器捕捉
比较模式：
1 = 发生了 TMR1/TMR3 寄存器的比较匹配 （必须用软件清零）
0 = 未发生 TMR1/TMR3 寄存器的比较匹配
PWM 模式：
在此模式下未使用。

bit 1　　**CCP1IF：**CCP1 中断标志位
捕捉模式：
1 = 发生了 TMR1/TMR3 寄存器捕捉 （必须用软件清零）
0 = 未发生 TMR1/TMR3 寄存器捕捉
比较模式：
1 = 发生了 TMR1/TMR3 寄存器的比较匹配 （必须用软件清零）
0 = 未发生 TMR1/TMR3 寄存器的比较匹配
PWM 模式：
在此模式下未使用。

bit 0　　**RTCCIF：**RTCC 中断标志位
1 = 发生了 RTCC 中断 （必须用软件清零）
0 = 未发生 RTCC 中断

9.3 PIE 寄存器

PIE 寄存器包含各外设中断的允许位。根据外设中断源的数量，有三个外设中断允许寄存器（PIE1、PIE2 和 PIE3）。当 IPEN = 0 时，要允许任一外设中断，必须将 PEIE 位置 1。

寄存器 9-7： PIE1：外设中断允许寄存器 1

U-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE
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 **未实现：**读为 0

bit 6 **ADIE：**A/D 转换器中断允许位
1 = 允许 A/D 中断
0 = 禁止 A/D 中断

bit 5 **RC1IE：**EUSART 接收中断允许位
1 = 允许 EUSART 接收中断
0 = 禁止 EUSART 接收中断

bit 4 **TX1IE：**EUSART 发送中断允许位
1 = 允许 EUSART 发送中断
0 = 禁止 EUSART 发送中断

bit 3 **SSPIE：**主同步串口中断允许位
1 = 允许 MSSP 中断
0 = 禁止 MSSP 中断

bit 2 **未实现：**读为 0

bit 1 **TMR2IE：**TMR2 与 PR2 匹配中断允许位
1 = 允许 TMR2 与 PR2 匹配中断
0 = 禁止 TMR2 与 PR2 匹配中断

bit 0 **TMR1IE：**TMR1 溢出中断允许位
1 = 允许 TMR1 溢出中断
0 = 禁止 TMR1 溢出中断

寄存器 9-8：　　PIE2：外设中断允许寄存器 2

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	U-0
OSCFIE	CMIE	—	—	BCLIE	LVDIE	TMR3IE	—
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　　**OSCFIE**：振荡器故障中断允许位
1 = 允许
0 = 禁止

bit 6　　**CMIE**：比较器中断允许位
1 = 允许
0 = 禁止

bit 5-4　　**未实现**：读为 0

bit 3　　**BCLIE**：总线冲突中断允许位
1 = 允许
0 = 禁止

bit 2　　**LVDIE**：低压检测中断允许位
1 = 允许
0 = 禁止

bit 1　　**TMR3IE**：TMR3 溢出中断允许位
1 = 允许
0 = 禁止

bit 0　　**未实现**：读为 0

寄存器 9-9：　　PIE3：外设中断允许寄存器 3

U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
—	LCDIE	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	RTCCIE
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　　**未实现：**读为 0

bit 6　　**LCDIE：**LCD 中断允许位 （当选择非静态模式下的 B 型波形时有效）
1 = 允许
0 = 禁止

bit 5　　**RC2IE：**AUSART 接收中断允许位
1 = 允许
0 = 禁止

bit 4　　**TX2IE：**AUSART 发送中断允许位
1 = 允许
0 = 禁止

bit 3　　**CTMUIE：**CTMU 中断允许位
1 = 允许
0 = 禁止

bit 2　　**CCP2IE：**CCP2 中断允许位
1 = 允许
0 = 禁止

bit 1　　**CCP1IE：**CCP1 中断允许位
1 = 允许 CCP1 中断
0 = 禁止 CCP1 中断

bit 0　　**RTCCIE：**RTCC 中断允许位
1 = 允许
0 = 禁止

9.4 IPR 寄存器

IPR 寄存器包含各外设中断的优先级位。根据外设中断源的数量，有三个外设中断优先级寄存器（IPR1、IPR2 和 IPR3）。使用优先级位时，要求将中断优先级使能（IPEN）位置 1。

寄存器 9-10： IPR1：外设中断优先级寄存器 1

U-0	R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	R/W-1
—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP
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 **未实现：**读为 0

bit 6 **ADIP：**A/D 转换器中断优先级位
1 = 高优先级
0 = 低优先级

bit 5 **RC1IP：**EUSART 接收中断优先级位
1 = 高优先级
0 = 低优先级

bit 4 **TX1IP：**EUSART 发送中断优先级位
1 = 高优先级
0 = 低优先级

bit 3 **SSPIP：**主同步串口中断优先级位
1 = 高优先级
0 = 低优先级

bit 2 **未实现：**读为 0

bit 1 **TMR2IP：**TMR2 与 PR2 匹配中断优先级位
1 = 高优先级
0 = 低优先级

bit 0 **TMR1IP：**TMR1 溢出中断优先级位
1 = 高优先级
0 = 低优先级

寄存器 9-11：　　IPR2：外设中断优先级寄存器 2

R/W-1	R/W-1	U-0	U-0	R/W-1	R/W-1	R/W-1	U-0
OSCFIP	CMIP	—	—	BCLIP	LVDIP	TMR3IP	—
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　　**OSCFIP：**振荡器故障中断优先级位
1 = 高优先级
0 = 低优先级

bit 6　　**CMIP：**比较器中断优先级位
1 = 高优先级
0 = 低优先级

bit 5-4　　**未实现：**读为 0

bit 3　　**BCLIP：**总线冲突中断优先级位
1 = 高优先级
0 = 低优先级

bit 2　　**LVDIP：**低压检测中断优先级位
1 = 高优先级
0 = 低优先级

bit 1　　**TMR3IP：**TMR3 溢出中断优先级位
1 = 高优先级
0 = 低优先级

bit 0　　**未实现：**读为 0

寄存器 9-12：　　IPR3：外设中断优先级寄存器 3

U-0	R/W-1	R-1	R-1	R/W-1	R/W-1	R/W-1	R/W-1
—	LCDIP	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	RTCCIP
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　　**未实现：**读为 0

bit 6　　**LCDIP：**LCD 中断优先级位 （当选择非静态模式下的 B 型波形时有效）
1 = 高优先级
0 = 低优先级

bit 5　　**RC2IP：**AUSART 接收中断优先级位
1 = 高优先级
0 = 低优先级

bit 4　　**TX2IP：**AUSART 发送中断优先级位
1 = 高优先级
0 = 低优先级

bit 3　　**CTMUIP：**CTMU 中断优先级位
1 = 高优先级
0 = 低优先级

bit　　**CCP2IP：**CCP2 中断优先级位
1 = 高优先级
0 = 低优先级

bit　　**CCP1IP：**CCP1 中断优先级位
1 = 高优先级
0 = 低优先级

bit 0　　**RTCCIP：**RTCC 中断优先级位
1 = 高优先级
0 = 低优先级

9.5 RCON 寄存器

RCON寄存器中包含的标志位可用来确定器件上次复位或者从空闲模式或休眠模式唤醒的原因。RCON 还包含一个可使能中断优先级的 IPEN 位。

寄存器 9-13： RCON：复位控制寄存器

R/W-0	U-0	R/W-1	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	$\overline{CM}$	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
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 **IPEN**：中断优先级使能位
1 = 使能中断优先级
0 = 禁止中断优先级 （PIC16CXXX 兼容模式）

bit 6 **未实现**：读为 0

bit 5 **$\overline{CM}$**：配置不匹配标志位
1 = 未发生配置不匹配复位
0 = 发生了配置不匹配复位 （随后必须用软件置 1。）

bit 4 **$\overline{RI}$**：RESET 指令标志位
位操作的详细信息，请参见寄存器 5-1。

bit 3 **$\overline{TO}$**：看门狗定时器超时标志位
位操作的详细信息，请参见寄存器 5-1。

bit 2 **$\overline{PD}$**：掉电检测标志位
位操作的详细信息，请参见寄存器 5-1。

bit 1 **$\overline{POR}$**：上电复位状态位
位操作的详细信息，请参见寄存器 5-1。

bit 0 **$\overline{BOR}$**：欠压复位状态位
位操作的详细信息，请参见寄存器 5-1。

9.6 INTx 引脚中断

RB0/INT0、RB1/INT1、RB2/INT2 和 RB3/INT3 引脚上的外部中断都是边沿触发的。如果 INTCON2 寄存器中相应的 INTEDGx 位被置 1 （= 1），则为上升沿触发；如果该位被清零，则为下降沿触发。当 RBx/INTx 引脚上出现一个有效边沿时，相应的标志位 INTxIF 被置 1。通过清零相应的允许位 INTxIE，可禁止该中断。在重新允许该中断前，必须在中断服务程序中先用软件将中断标志位 INTxIF 清零。

如果 INTxIE 位在进入功耗管理模式前被置 1，则所有的外部中断（INT0、INT1、INT2 和 INT3）均能将处理器从功耗管理模式唤醒。如果全局中断允许位 GIE 被置 1，则处理器将在被唤醒之后跳转到中断向量处执行。

INT1、 INT2 和 INT3 的中断优先级由中断优先级位 INT1IP（INTCON3<6>）、INT2IP（INTCON3<7>）和 INT3IP （INTCON2<1>）的值决定。没有与 INT0 相关的优先级位。 INT0 始终是一个高优先级的中断源。

9.7 TMR0 中断

在 8 位模式（默认模式）下， TMR0 寄存器的溢出（FFh → 00h）会将 TMR0IF 标志位置 1。在 16 位模式下，TMR0H:TMR0L 寄存器对的溢出（FFFFh → 0000h）会将 TMR0IF 标志位置 1。可以通过置 1/ 清零中断允许位 TMR0IE（INTCON<5>）来允许 / 禁止该中断。Timer0 的中断优先级由中断优先级位 TMR0IP （INTCON2<2>）的值决定。欲进一步了解 Timer0 模块的详细信息，请参见**第 11.0 节 “Timer0 模块 ”**。

9.8 PORTB 电平变化中断

PORTB<7:4> 上的输入电平变化会将标志位 RBIF（INTCON<0>）置 1。可以通过置 1/ 清零中断允许位 RBIE（INTCON<3>）来允许 / 禁止该中断。 PORTB 电平变化中断的优先级由中断优先级位 RBIP （INTCON2<0>）的值决定。

9.9 中断的现场保护

在中断期间， PC 的返回地址被保存在堆栈中。另外，WREG、STATUS 和 BSR 寄存器的值被压入快速返回堆栈。如果未使用从中断快速返回功能 （见**第 6.3 节 “ 数据存储器构成 ”**），那么用户可能需要在进入中断服务程序（Interrupt Service Routine，ISR）前，保存 WREG、STATUS 和 BSR 寄存器的值。根据用户的具体应用，还可能需要保存其他寄存器的值。例 9-1 在执行中断服务程序期间，保存并恢复 WREG、 STATUS 和 BSR 寄存器的值。

例 9-1： 将 STATUS、 WREG 和 BSR 寄存器的值保存在 RAM 中

```
MOVWF   W_TEMP                  ; W_TEMP is in virtual bank
MOVFF   STATUS, STATUS_TEMP     ; STATUS_TEMP located anywhere
MOVFF   BSR, BSR_TEMP           ; BSR_TMEP located anywhere
;
; USER ISR CODE
;
MOVFF   BSR_TEMP, BSR           ; Restore BSR
MOVF    W_TEMP, W               ; Restore WREG
MOVFF   STATUS_TEMP, STATUS     ; Restore STATUS
```

10.0 I/O 端口

根据选定的器件和使能的功能，最多有 9 个端口可供使用。I/O 端口的一些引脚与器件上外设功能复用。通常而言，当某个外设使能时，其相关引脚可能不能用作通用 I/O 引脚。

每个端口都有三个存储器映射的寄存器与其操作相关：

- TRIS 寄存器 （数据方向寄存器）
- PORT 寄存器 （读取器件引脚的电平）
- LAT 寄存器 （输出锁存寄存器）

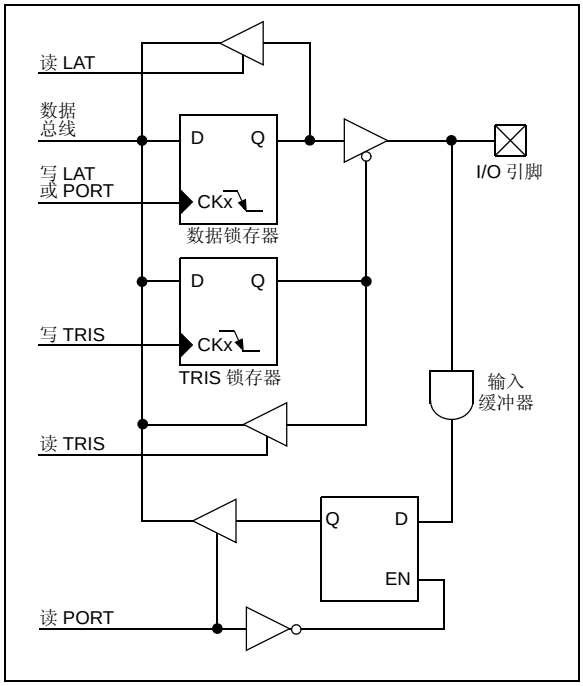
读 PORT 寄存器将读出引脚的当前状态，而写 PORT 寄存器则是将数据写入输出锁存 （LAT）寄存器。

将 TRIS 某位置 1 （= 1）时，会将 PORT 的相应引脚设为输入 （即，使相应的输出驱动器呈高阻态）。将 TRIS 某位清零（= 0）时，会将 PORT 的相应引脚设为输出 （即，将相应 LAT 位的内容输出到所选引脚）。

在对 I/O 引脚驱动值进行读 - 修改 - 写操作时会用到输出锁存器 （LAT 寄存器）。对 LAT 寄存器执行读 - 修改 - 写操作将读写 PORT 寄存器的锁存输出值。

图 10-1 给出了通用 I/O 端口的简化模型，没有给出与其他外设的接口。

图 10-1： 通用 I/O 端口的工作原理



10.1 I/O 端口引脚功能

在开发应用程序时，必须考虑到端口引脚的能力。某些引脚上的输出驱动能力比其他引脚要高。同样，某些引脚可以承受高于 VDD 的输入电压。

10.1.1 输入引脚和电压注意事项

用作器件输入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<7:0>	VDD	只能承受 VDD 以下的输入电压。
PORTC<1:0>		
PORTF<7:1>		
PORTG<3:2>		
PORTB<7:0>	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。
PORTC<7:2>		
PORTD<7:0>		
PORTE<7:3>		
PORTG<4:1>		
PORTH<7:0>(1)		
PORTJ<7:0>(1)		

注 1： 在 PIC18F6XJ90 器件上不可用。

10.1.2 引脚输出驱动能力

用作数字 I/O 时，各引脚组的输出引脚驱动能力是不同的，这样可以满足各种不同应用的需求。通常，按驱动能力可划分为三类输出引脚。

PORTB 和 PORTC 以及 PORTA<7:6> 是为驱动较高的电流负载（例如 LED）而设计的。PORTD、PORTE 和 PORTJ 也能驱动 LED，但只是指那些电流要求较小的 LED。PORTF、PORTG 和 PORTH 以及 PORTA<5:0> 具有最低的驱动能力，但能够驱动具有高输入阻抗的常规数字电路负载。不管引脚位于哪个端口，LCD 段或公共模式中的所有输出引脚都具有足够的输出能力可直接驱动显示屏。

表 10-2 汇总了端口的输出能力。更多详细信息，请参见 **第 28.0 节 “ 电气特性 ”** 中的 **“ 绝对最大额定值 ”**。

表 10-2：　　各个端口的输出驱动能力

低	中	高
PORTA<5:0>	PORTD	PORTA<7:6>
PORTF	PORTE	PORTB
PORTG	PORTJ(1)	PORTC
PORTH(1)		

注　1： 在 PIC18F6XJ90 器件上不可用。

10.1.3　上拉配置

4 个 I/O 端口（PORTB、PORTD、PORTE 和 PORTJ）实现了所有引脚的可配置弱上拉。这些内部上拉可使悬空数字输入信号上拉至一固定的电平而无需使用外部电阻。

可通过每个端口的单个位使能上拉功能：对于 PORTB 是 $\overline{\text{RBPU}}$（INTCON2<7>），对于其他端口为 RDPU、REPU 和 PJPU（PORTG<7:5>）。

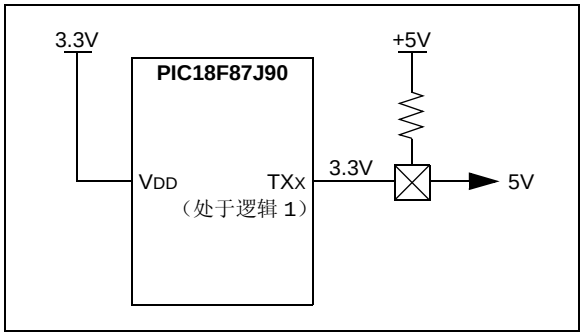
10.1.4　漏极开路输出

几个外设的输出引脚还配备了一个可配置的漏极开路输出选项。这使外设可以与工作在较高电压下的外部数字逻辑通信，而无需使用电平转换器。

漏极开路选项在与 USART、MSSP 模块（在 SPI 模式下）和 CCP 模块的数据和时钟输出相关的端口引脚上实现。通过将 TRISG 和 LATG 中相应模块的漏极开路控制位置 1，可以有选择地使能该选项。在**第 10.4 节 “PORTC、TRISC 和 LATC 寄存器”**、**第 10.6 节 “PORTE、TRISE 和 LATE 寄存器”** 和**第 10.8 节 “PORTG、TRISG 和 LATG 寄存器”** 中对其配置进行了更详细的讨论。

当需要漏极开路选项时，输出引脚也必须通过用户提供的外部上拉电阻连接到较高电压，最高为 5V（图 10-2）。当输出数字逻辑高电平信号时，它被上拉至更高电压。

图 10-2：　　使用漏极开路输出（以 USART 为例）



10.2　PORTA、TRISA 和 LATA 寄存器

PORTA 是一个 8 位宽的双向端口，对应的数据方向和输出锁存寄存器是 TRISA 和 LATA。

RA4/T0CKI 是施密特触发器输入引脚。所有其他 PORTA 引脚都采用 TTL 输入电平和全 CMOS 输出驱动器。

RA4 引脚与 Timer0 时钟输入和 LCD 段驱动之一复用。RA5 和 RA<3:0> 与 A/D 转换器的模拟输入复用。

通过将 ADCON1 寄存器中的 PCFG<3:0> 控制位清零或置 1，可将模拟输入选作 A/D 转换器输入。相应的 TRISA 位控制着这些引脚的方向，即使它们被用作模拟输入。当引脚用作模拟输入时，用户必须确保 TRISA 寄存器中的相应位保持置 1。

注：　RA5 和 RA<3:0> 在任何复位时被配置为模拟输入并读为 0。RA4 被配置为数字输入。

OSC2/CLKO/RA6 和 OSC1/CLKI/RA7 通常用作外部（主）振荡器电路（HS 振荡器模式），或外部时钟输入和输出（EC 振荡器模式）的外部电路连接引脚。在这些情况下，RA6 和 RA7 不能用作数字 I/O，并且其相应的 TRIS 和 LAT 位读为 0。当器件被配置为使用 INTOSC 或 INTRC 作为默认振荡器模式（FOSC2 配置位为 0）时，RA6 和 RA7 会被自动配置为数字 I/O；振荡器和时钟输入 / 时钟输出功能被禁止。

RA1、RA4 和 RA5 与 LCD 段驱动复用，由 LCDSE1 和 LCDSE2 寄存器中的位控制。仅当禁止 LCD 段时，才能使用 I/O 端口功能。

例 10-1：　　初始化 PORTA

```
CLRF    PORTA   ; Initialize PORTA by
                ; clearing output latches
CLRF    LATA    ; Alternate method to
                ; clear output data latches
MOVLW   07h     ; Configure A/D
MOVWF   ADCON1  ; for digital inputs
MOVLW   0BFh    ; Value used to initialize
                ; data direction
MOVWF   TRISA   ; Set RA<7, 5:0> as inputs,
                ; RA<6> as output
```

表 10-3： PORTA 功能

引脚名称	功能	TRIS 设置	I/O	I/O 类型	说明
RA0/AN0	RA0	0	O	DIG	LATA<0> 数据输出；不受模拟输入影响。
		1	I	TTL	PORTA<0> 数据输入；当使能模拟输入时被禁止。
	AN0	1	I	ANA	A/D 输入通道 0。 POR 时的默认输入配置；不影响数字输出。
RA1/AN1/SEG18	RA1	0	O	DIG	LATA<1> 数据输出；不受模拟输入影响。
		1	I	TTL	PORTA<1> 数据输入；当使能模拟输入时被禁止。
	AN1	1	I	ANA	A/D 输入通道 1。 POR 时的默认输入配置；不影响数字输出。
	SEG18	x	O	ANA	LCD 段 18 的输出；禁止所有其他引脚功能。
RA2/AN2/VREF-	RA2	0	O	DIG	LATA<2> 数据输出；不受模拟输入影响。
		1	I	TTL	PORTA<2> 数据输入；当使能模拟功能时被禁止。
	AN2	1	I	ANA	A/D 输入通道 2。 POR 时的默认输入配置。
	VREF-	1	I	ANA	A/D 和比较器低参考电压输入。
RA3/AN3/VREF+	RA3	0	O	DIG	LATA<3> 数据输出；不受模拟输入影响。
		1	I	TTL	PORTA<3> 数据输入；当使能模拟输入时被禁止。
	AN3	1	I	ANA	A/D 输入通道 3。 POR 时的默认输入配置。
	VREF+	1	I	ANA	A/D 和比较器高参考电压输入。
RA4/T0CKI/ SEG14	RA4	0	O	DIG	LATA<4> 数据输出。
		1	I	ST	PORTA<4> 数据输入。 POR 时的默认配置。
	T0CKI	x	I	ST	Timer0 的时钟输入。
	SEG14	x	O	ANA	LCD 段 14 的输出；禁止所有其他引脚功能。
RA5/AN4/SEG15	RA5	0	O	DIG	LATA<5> 数据输出；不受模拟输入影响。
		1	I	TTL	PORTA<5> 数据输入；当使能模拟输入时被禁止。
	AN4	1	I	ANA	A/D 输入通道 4。 POR 时的默认配置。
	SEG15	x	O	ANA	LCD 段 15 的输出；禁止所有其他引脚功能。
OSC2/CLKO/RA6	OSC2	x	O	ANA	主振荡器反馈输出连接 （HS 和 HSPLL 模式）。
	CLKO	x	O	DIG	系统周期时钟输出 （Fosc/4）（EC 和 ECPLL 模式）。
	RA6	0	O	DIG	LATA<6> 数据输出；当 FOSC2 配置位置 1 时被禁止。
		1	I	TTL	PORTA<6> 数据输入；当 FOSC2 配置位置 1 时被禁止。
OSC1/CLKI/RA7	OSC1	x	I	ANA	主振荡器输入连接 （HS 和 HSPLL 模式）。
	CLKI	x	I	ANA	主外部时钟源输入 （EC 和 ECPLL 模式）。
	RA7	0	O	DIG	LATA<7> 数据输出；当 FOSC2 配置位置 1 时被禁止。
		1	I	TTL	PORTA<7> 数据输入；当 FOSC2 配置位置 1 时被禁止。

图注： O = 输出， I = 输入， ANA = 模拟信号， DIG = 数字输出， ST = 施密特触发器缓冲器输入，
TTL = TTL 缓冲器输入， x = 无关位 （TRIS 位不影响端口方向或在此可忽略）。

表 10-4： 与 PORTA 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
PORTA	RA7(1)	RA6(1)	RA5	RA4	RA3	RA2	RA1	RA0	63
LATA	LATA7(1)	LATA6(1)	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0	62
TRISA	TRISA7(1)	TRISA6(1)	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	62
ADCON1	TRIGSEL	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	61
LCDSE1	SE15	SE14	SE13	SE12	SE11	SE10	SE09	SE08	61
LCDSE2	SE23	SE22	SE21	SE20	SE19	SE18	SE17	SE16	61

图注： — = 未实现，读为 0。 PORTA 不使用阴影单元。

注 1： 这些位根据所选的振荡器模式被使能。如果未被使能为 PORTA 引脚，则它们将被禁止并读为 x。

10.3 PORTB、 TRISB 和 LATB 寄存器

PORTB 是一个 8 位宽的双向端口，对应的数据方向和输出锁存寄存器是 TRISB 和 LATB。PORTB 上的所有引脚都是仅数字功能引脚，可以承受最高 5.5V 的电压。

例 10-2： 初始化 PORTB

```
CLRF    PORTB   ; Initialize PORTB by
                ; clearing output
                ; data latches
CLRF    LATB    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISB   ; Set RB<3:0> as inputs
                ; RB<5:4> as outputs
                ; RB<7:6> as inputs
```

每个 PORTB 引脚都具有内部弱上拉功能。一个控制位即可接通所有上拉。这是通过清零$\overline{\text{RBPU}}$位（INTCON2<7>）实现的。当端口引脚被配置为输出时，其弱上拉功能会自动关闭。上电复位会禁止上拉功能。

PORTB 的 4 个引脚 （RB<7:4>）具有电平变化中断功能。仅当将这些引脚配置为输入时，才能产生此中断（即当 RB<7:4> 中的任何一个引脚被配置为输出时，该引脚将不再具有电平变化中断功能） 。将输入引脚（RB<7:4>）上的电平与 PORTB 上次读入锁存器的旧值进行比较。对 RB<7:4> 上的 “ 不匹配 ” 输出进行逻辑或运算，产生 RB 端口电平变化中断，并将标志位 RBIF（INTCON<0>）置 1。

该中断可将器件从功耗管理模式唤醒。用户可用以下方式在中断服务程序中清除该中断：

a) 读或写 PORTB（MOVFF (ANY), PORTB 指令除外）。这将结束不匹配条件。
b) 等待一个指令周期 （例如执行一条 NOP 指令）。
c) 清零标志位 RBIF。

不匹配条件将继续把标志位 RBIF 置 1。读 PORTB 将结束不匹配条件并允许在经过一个 TCY 的延时后，将标志位 RBIF 清零。

建议使用电平变化中断功能实现按键唤醒操作，以及那些仅将 PORTB 用于电平变化中断功能的操作。在使用电平变化中断功能时，建议不要查询 PORTB 的状态。

RB<3:2> 复用为 CTMU 边沿输入。

RB<5:0>还与LCD段驱动复用，由LCDSE1和LCDSE3寄存器中的位控制。仅当禁止 LCD 段时，才能使用 I/O 端口功能。

表 10-5： PORTB 功能

引脚名称	功能	TRIS 设置	I/O	I/O 类型	说明
RB0/INT0/SEG30	RB0	0	O	DIG	LATB<0> 数据输出。
		1	I	TTL	PORTB<0> 数据输入；当 $\overline{\text{RBPU}}$ 位清零时使能弱上拉。
	INT0	1	I	ST	外部中断 0 输入。
	SEG30	x	O	ANA	LCD 段 30 的输出；禁止所有其他引脚功能。
RB1/INT1/SEG8	RB1	0	O	DIG	LATB<1> 数据输出。
		1	I	TTL	PORTB<1> 数据输入；当 $\overline{\text{RBPU}}$ 位清零时使能弱上拉。
	INT1	1	I	ST	外部中断 1 输入。
	SEG8	x	O	ANA	LCD 段 8 的输出；禁止所有其他引脚功能。
RB2/INT2/SEG9/ CTED1	RB2	0	O	DIG	LATB<2> 数据输出。
		1	I	TTL	PORTB<2> 数据输入；当 $\overline{\text{RBPU}}$ 位清零时使能弱上拉。
	INT2	1	I	ST	外部中断 2 输入。
	SEG9	x	O	ANA	LCD 段 9 的输出；禁止所有其他引脚功能。
	CTED1	x	I	ST	CTMU 边沿 1 输入。
RB3/INT3/SEG10/ CTED2	RB3	0	O	DIG	LATB<3> 数据输出。
		1	I	TTL	PORTB<3> 数据输入；当 $\overline{\text{RBPU}}$ 位清零时使能弱上拉。
	INT3	1	I	ST	外部中断 3 输入。
	SEG10	x	O	ANA	LCD 段 10 的输出；禁止所有其他引脚功能。
	CTED2	x	I	ST	CTMU 边沿 2 输入。
RB4/KBI0/SEG11	RB4	0	O	DIG	LATB<4> 数据输出。
		1	I	TTL	PORTB<4> 数据输入；当 $\overline{\text{RBPU}}$ 位清零时使能弱上拉。
	KBI0	1	I	TTL	引脚电平变化中断。
	SEG11	x	O	ANA	LCD 段 11 的输出；禁止所有其他引脚功能。
RB5/KBI1/SEG29	RB5	0	O	DIG	LATB<5> 数据输出。
		1	I	TTL	PORTB<5> 数据输入；当 $\overline{\text{RBPU}}$ 位清零时使能弱上拉。
	KBI1	1	I	TTL	引脚电平变化中断。
	SEG29	x	O	ANA	LCD 段 29 的输出；禁止所有其他引脚功能。
RB6/KBI2/PGC	RB6	0	O	DIG	LATB<6> 数据输出。
		1	I	TTL	PORTB<6> 数据输入；当 $\overline{\text{RBPU}}$ 位清零时使能弱上拉。
	KBI2	1	I	TTL	引脚电平变化中断。
	PGC	x	I	ST	供 ICSP 和 ICD 操作使用的串行执行 （ICSP）时钟输入。
RB7/KBI3/PGD	RB7	0	O	DIG	LATB<7> 数据输出。
		1	I	TTL	PORTB<7> 数据输入；当 $\overline{\text{RBPU}}$ 位清零时使能弱上拉。
	KBI3	1	I	TTL	引脚电平变化中断。
	PGD	x	O	DIG	供 ICSP 和 ICD 操作使用的串行执行数据输出。
		x	I	ST	供 ICSP 和 ICD 操作使用的串行执行数据输入。

图注： O = 输出， I = 输入， ANA = 模拟信号， DIG = 数字输出， ST = 施密特触发器缓冲器输入，
TTL = TTL 缓冲器输入， x = 无关位 （TRIS 位不影响端口方向或在此可忽略）。

表 10-6：　　与 PORTB 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值 所在页
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	63
LATB	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0	62
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	62
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
INTCON2	$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP	59
INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF	59
LCDSE1	SE15	SE14	SE13	SE12	SE11	SE10	SE09	SE08	61
LCDSE3	SE31	SE30	SE29	SE28	SE27	SE26	SE25	SE24	61

图注：　　PORTB 不使用阴影单元。

10.4 PORTC、 TRISC 和 LATC 寄存器

PORTC 是一个 8 位宽的双向端口，对应的数据方向和输出锁存寄存器是 TRISC 和 LATC。仅 PORTC 引脚 RC2 到 RC7 能作为仅数字功能引脚并且可以承受最高 5.5V 的输入电压。

PORTC 与 CCP、 MSSP 和 EUSART 外设功能复用（表 10-7）。这些引脚配有施密特触发器输入缓冲器。无论何时，只要这些功能有效，CCP、SPI 和 EUSART 的引脚就可配置为漏极开路输出。通过将 SPIOD、CCPxOD 和 U1OD 控制位（分别为 TRISG<7:5> 和 LATG<6>）置 1 选择漏极开路配置。

RC1 通常被配置为 CCP2 模块的默认外设引脚。CCP2 的分配由配置位 CCP2MX（默认状态，CCP2MX = 1）控制。

当使能外设功能时，应小心定义每个 PORTC 引脚的 TRIS 位。有些外设会改写 TRIS 位的设置，将引脚定义为输出引脚或输入引脚。用户应该查阅相应的外设章节来正确设置 TRIS 位。

注：	这些引脚在任何器件复位时都被配置为数字输入引脚。

外设对引脚的改写会影响 TRISC 寄存器的内容。尽管外设可能会改写一个或多个引脚，读 TRISC 总是会返回其当前的内容。

RC<7:1> 引脚与 LCD 段驱动复用，由 LCDSE1、LCDSE2、LCDSE3 和 LCDSE4 寄存器中的位控制。仅当禁止 LCD 段时，才能使用 I/O 端口功能。

例 10-3： 初始化 PORTC

```
CLRF    PORTC   ; Initialize PORTC by
                ; clearing output
                ; data latches
CLRF    LATC    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISC   ; Set RC<3:0> as inputs
                ; RC<5:4> as outputs
                ; RC<7:6> as inputs
```

表 10-7： PORTC 功能

引脚名称	功能	TRIS 设置	I/O	I/O 类型	说明
RC0/T1OSO/ T13CKI	RC0	0	O	DIG	LATC<0> 数据输出。
		1	I	ST	PORTC<0> 数据输入。
	T1OSO	x	O	ANA	Timer1 振荡器输出；当使能 Timer1 振荡器时被使能。禁止数字 I/O 和 LCD 段驱动器。
	T13CKI	1	I	ST	Timer1/Timer3 计数器输入。
RC1/T1OSI/ CCP2/SEG32	RC1	0	O	DIG	LATC<1> 数据输出。
		1	I	ST	PORTC<1> 数据输入。
	T1OSI	x	I	ANA	Timer1 振荡器输入。
	CCP2(1)	0	O	DIG	CCP2 比较 /PWM 输出。
		1	I	ST	CCP2 捕捉输入。
	SEG32	x	O	ANA	LCD 段 32 的输出；禁止所有其他引脚功能。
RC2/CCP1/ SEG13	RC2	0	O	DIG	LATC<2> 数据输出。
		1	I	ST	PORTC<2> 数据输入。
	CCP1	0	O	DIG	CCP1 比较 /PWM 输出；优先于端口数据。
		1	I	ST	CCP1 捕捉输入。
	SEG13	x	O	ANA	LCD 段 13 的输出；禁止所有其他引脚功能。
RC3/SCK/SCL/ SEG17	RC3	0	O	DIG	LATC<3> 数据输出。
		1	I	ST	PORTC<3> 数据输入。
	SCK	0	O	DIG	SPI 时钟输出 （MSSP 模块）；优先于端口数据。
		1	I	ST	SPI 时钟输入 （MSSP 模块）。
	SCL	0	O	DIG	I^2C™ 时钟输出 （MSSP 模块）；优先于端口数据。
		1	I	I2C	I^2C 时钟输入 （MSSP 模块）；输入类型取决于模块设置。
	SEG17	x	O	ANA	LCD 段 17 的输出；禁止所有其他引脚功能。
RC4/SDI/SDA/ SEG16	RC4	0	O	DIG	LATC<4> 数据输出。
		1	I	ST	PORTC<4> 数据输入。
	SDI		I	ST	SPI 数据输入 （MSSP 模块）。
	SDA	1	O	DIG	I^2C 数据输出 （MSSP 模块）；优先于端口数据。
		1	I	I2C	I^2C 数据输入 （MSSP 模块）；输入类型取决于模块设置。
	SEG16	x	O	ANA	LCD 段 16 的输出；禁止所有其他引脚功能。
RC5/SDO/ SEG12	RC5	0	O	DIG	LATC<5> 数据输出。
		1	I	ST	PORTC<5> 数据输入。
	SDO	0	O	DIG	SPI 数据输出 （MSSP 模块）。
	SEG12	x	O	ANA	LCD 段 12 的输出；禁止所有其他引脚功能。
RC6/TX1/CK1/ SEG27	RC6	0	O	DIG	LATC<6> 数据输出。
		1	I	ST	PORTC<6> 数据输入。
	TX1	1	O	DIG	同步串行数据输出 （EUSART 模块）；优先于端口数据。
	CK1	1	O	DIG	同步串行数据输入 （EUSART 模块）；用户必须将其配置为输入。
		1	I	ST	同步串行时钟输入 （EUSART 模块）。
	SEG27	x	O	ANA	LCD 段 27 的输出；禁止所有其他引脚功能。
RC7/RX1/DT1/ SEG28	RC7	0	O	DIG	LATC<7> 数据输出。
		1	I	ST	PORTC<7> 数据输入。
	RX1	1	I	ST	异步串行接收数据输入 （EUSART 模块）。
	DT1	1	O	DIG	同步串行数据输出 （EUSART 模块）；优先于端口数据。
		1	I	ST	同步串行数据输入 （EUSART 模块）；用户必须将其配置为输入。
	SEG28	x	O	ANA	LCD 段 28 的输出；禁止所有其他引脚功能。

图注： O = 输出， I = 输入， ANA = 模拟信号， DIG = 数字输出， ST = 施密特触发器缓冲器输入， TTL = TTL 缓冲器输入， I2C = I^2C/SMBus 缓冲器输入， x = 无关位 （TRIS 位不影响端口方向或在此可忽略）。

注 1： 当 CCP2MX 配置位置 1 时 CCP2 的默认分配。

表 10-8：　　与 PORTC 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	63
LATC	LATC7	LATBC6	LATC5	LATCB4	LATC3	LATC2	LATC1	LATC0	62
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	62
LATG	U2OD	U1OD	—	LATG4	LATG3	LATG2	LATG1	LATG0	62
TRISG	SPIOD	CCP2OD	CCP1OD	TRISG4	TRISG3	TRISG2	TRISG1	TRISG0	62
LCDSE1	SE15	SE14	SE13	SE12	SE11	SE10	SE09	SE08	61
LCDSE2	SE23	SE22	SE21	SE20	SE19	SE18	SE17	SE16	61
LCDSE3	SE31	SE30	SE29	SE28	SE27	SE26	SE25	SE24	61
LCDSE4	SE39(1)	SE38(1)	SE37(1)	SE36(1)	SE35(1)	SE34(1)	SE33(1)	SE32	61

图注：　　PORTC 不使用阴影单元。

注　1：　在 PIC18F6XJ90 器件上未实现，读为 0。

10.5 PORTD、 TRISD 和 LATD 寄存器

PORTD 是一个 8 位宽的双向端口，对应的数据方向和输出锁存寄存器是 TRISD 和 LATD。PORTD 上的所有引脚都是仅数字功能引脚，并且可以承受最高 5.5V 的电压。

PORTD 上的所有引脚都配有施密特触发器输入缓冲器。每个引脚都可被单独配置为输入或输出。

注： 这些引脚在任何器件复位时都被配置为数字输入引脚。

PORTD 的每个引脚都具有内部弱上拉功能。通过单个控制位可以关闭所有上拉。可以通过清零 RDPU 位（PORTG<7>）来实现。当端口引脚被配置为输出时，其弱上拉功能会自动关闭。在所有器件复位时上拉被禁止。

PORTD 的所有引脚均与 LCD 段驱动复用，由 LCDSE0 寄存器中的位控制。RD0与CTMU脉冲发生器输出复用。仅当禁止 LCD 段时，才能使用 I/O 端口功能。

例 10-4： 初始化 PORTD

```
CLRF    PORTD   ; Initialize PORTD by
                 ; clearing output
                 ; data latches
CLRF    LATD    ; Alternate method
                 ; to clear output
                 ; data latches
MOVLW   0CFh    ; Value used to
                 ; initialize data
                 ; direction
MOVWF   TRISD   ; Set RD<3:0> as inputs
                 ; RD<5:4> as outputs
                 ; RD<7:6> as inputs
```

表 10-9： PORTD 功能

引脚名称	功能	TRIS 设置	I/O	I/O 类型	说明
RD0/SEG0/ CTPLS	RD0	0	O	DIG	LATD<0> 数据输出。
		1	I	ST	PORTD<0> 数据输入。
	SEG0	x	O	ANA	LCD 段 0 的输出；禁止所有其他引脚功能。
	CTPLS	x	O	DIG	CTMU 脉冲发生器输出。
RD1/SEG1	RD1	0	O	DIG	LATD<1> 数据输出。
		1	I	ST	PORTD<1> 数据输入。
	SEG1	x	O	ANA	LCD 段 1 的输出；禁止所有其他引脚功能。
RD2/SEG2	RD2	0	O	DIG	LATD<2> 数据输出。
		1	I	ST	PORTD<2> 数据输入。
	SEG2	x	O	ANA	LCD 段 2 的输出；禁止所有其他引脚功能。
RD3/SEG3	RD3	0	O	DIG	LATD<3> 数据输出。
		1	I	ST	PORTD<3> 数据输入。
	SEG3	x	O	ANA	LCD 段 3 的输出；禁止所有其他引脚功能。
RD4/SEG4	RD4	0	O	DIG	LATD<4> 数据输出。
		1	I	ST	PORTD<4> 数据输入。
	SEG4	x	O	ANA	LCD 段 4 的输出；禁止所有其他引脚功能。
RD5/SEG5	RD5	0	O	DIG	LATD<5> 数据输出。
		1	I	ST	PORTD<5> 数据输入。
	SEG5	x	O	ANA	LCD 段 5 的输出；禁止所有其他引脚功能。
RD6/SEG6	RD6	0	O	DIG	LATD<6> 数据输出。
		1	I	ST	PORTD<6> 数据输入。
	SEG6	x	O	ANA	LCD 段 6 的输出；禁止所有其他引脚功能。
RD7/SEG7	RD7	0	O	DIG	LATD<7> 数据输出。
		1	I	ST	PORTD<7> 数据输入。
	SEG7	x	I	ANA	LCD 段 7 的输出；禁止所有其他引脚功能。

图注： O = 输出， I = 输入， ANA = 模拟信号， DIG = 数字输出， ST = 施密特触发器缓冲器输入，
x = 无关位 （TRIS 位不影响端口方向或在此可忽略）。

表 10-10： 与 PORTD 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	63
LATD	LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0	62
TRISD	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	62
PORTG	RDPU	REPU	RJPU(1)	RG4	RG3	RG2	RG1	RG0	62
LCDSE0	SE07	SE06	SE05	SE04	SE03	SE02	SE01	SE00	61

图注： PORTD 不使用阴影单元。

注 1： 在 PIC18F6XJ90 器件上未实现，读为 0。

10.6 PORTE、 TRISE 和 LATE 寄存器

PORTE 是一个 7 位宽的双向端口，对应的数据方向和输出锁存寄存器是 TRISE 和 LATE。PORTE 上的所有引脚都是仅数字功能引脚，并且可以承受最高 5.5V 的电压。

PORTE 上的所有引脚都配有施密特触发器输入缓冲器。每个引脚都可被单独配置为输入或输出。当 RE7 引脚上的 CCP2 有效时，该引脚还可配置为漏极开路输出。通过将 CCP2OD 控制位 （TRISG<6>）置 1 选择漏极开路配置。

注：	这些引脚在任何器件复位时都被配置为数字输入引脚。

PORTE 的每个引脚都具有内部弱上拉功能。通过单个控制位可以关闭所有上拉。可以通过清零 REPU 位（PORTG<6>）来实现。当端口引脚被配置为输出时，其弱上拉功能会自动关闭。发生任何器件复位时，上拉功能会被禁止。

RE<6:3> 引脚与 LCD 公共端驱动复用。根据哪些公共端有效，I/O 端口功能仅在某些 PORTE 引脚上可用。配置由 LMUX<1:0> 控制位 （LCDCON<1:0>）决定。表 10-11 汇总了可用性。

表 10-11： 不同 LCD 驱动配置中可用的 PORTE 引脚

LCDCON <1:0>	有效的 LCD 公共端	可用于 I/O 的 PORTE
00	COM0	RE6、 RE5 和 RE4
01	COM0 和 COM1	RE6 和 RE5
10	COM0、 COM1 和 COM2	RE6
11	全部 （COM0 到 COM3）	无

RE1 和 RE0 引脚与 LCDBIAS2 和 LCDBIAS1 的功能复用。当需要产生 LCD 偏置时 （即，器件被连接到外部 LCD 的任何应用），这些引脚不能用作数字 I/O。

注：	在该器件中，其他 PIC18F 器件的 RE2 对应的引脚具有 LCDBIAS3 的功能。它不能用作数字 I/O。

RE7 与 LCD 段驱动 （SEG31）复用，由 LCDSE3<7> 位控制。仅当禁止该段时，才能使用 I/O 端口功能。

RE7 还可以配置为 CCP2 模块的备用外设引脚。这是通过将 CCP2MX 配置位清零实现的。

例 10-5： 初始化 PORTE

```
CLRF    PORTE   ; Initialize PORTE by
                ; clearing output
                ; data latches
CLRF    LATE    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   03h     ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISE   ; Set RE<1:0> as inputs
                ; RE<7:2> as outputs
```

表 10-12： PORTE 功能

引脚名称	功能	TRIS 设置	I/O	I/O 类型	说明
RE0/LCDBIAS1	RE0	0	O	DIG	LATE<0> 数据输出。
		1	I	ST	PORTE<0> 数据输入。
	LCDBIAS1	—	I	ANA	LCD 模块偏置电压输入。
RE1/LCDBIAS2	RE1	0	O	DIG	LATE<1> 数据输出。
		1	I	ST	PORTE<1> 数据输入。
	LCDBIAS2	—	I	ANA	LCD 模块偏置电压输入。
RE3/COM0	RE3	0	O	DIG	LATE<3> 数据输出。
		1	I	ST	PORTE<3> 数据输入。
	COM0	x	O	ANA	LCD 公共端 0 的输出；禁止所有其他输出。
RE4/COM1	RE4	0	O	DIG	LATE<4> 数据输出。
		1	I	ST	PORTE<4> 数据输入。
	COM1	x	O	ANA	LCD 公共端 1 的输出；禁止所有其他输出。
RE5/COM2	RE5	0	O	DIG	LATE<5> 数据输出。
		1	I	ST	PORTE<5> 数据输入。
	COM2	x	O	ANA	LCD 公共端 2 的输出；禁止所有其他输出。
RE6/COM3	RE6	0	O	DIG	LATE<6> 数据输出。
		1	I	ST	PORTE<6> 数据输入。
	COM3	x	O	ANA	LCD 公共端 3 的输出；禁止所有其他输出。
RE7/CCP2/ SEG31	RE7	0	O	DIG	LATE<7> 数据输出。
		1	I	ST	PORTE<7> 数据输入。
	CCP2(1)	0	O	DIG	CCP2 比较 /PWM 输出；优先于端口数据。
		1	I	ST	CCP2 捕捉输入。
	SEG31	x	O	ANA	LCD 的段 31 模拟输出；禁止数字输出。

图注： O = 输出， I = 输入， ANA = 模拟信号， DIG = 数字输出， ST = 施密特触发器缓冲器输入，
x = 无关位（TRIS 位不影响端口方向或在此可忽略）。

注 1： 当 CCP2MX 配置位清零时 CCP2 的备用分配。

表 10-13： 与 PORTE 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
PORTE	RE7	RE6	RE5	RE4	RE3	—	RE1	RE0	63
LATE	LATE7	LATE6	LATE5	LATE4	LATE3	—	LATE1	LATE0	62
TRISE	TRISE7	TRISE6	TRISE5	TRISE4	TRISE3	—	TRISE1	TRISE0	62
PORTG	RDPU	REPU	RJPU(1)	RG4	RG3	RG2	RG1	RG0	62
TRISG	SPIOD	CCP2OD	CCP1OD	TRISG4	TRISG3	TRISG2	TRISG1	TRISG0	62
LCDCON	LCDEN	SLPEN	WERR	—	CS1	CS0	LMUX1	LMUX0	61
LCDSE3	SE31	SE30	SE29	SE28	SE27	SE26	SE25	SE24	61

图注： PORTE 不使用阴影单元。

注 1： 在 PIC18F6XJ90 器件上未实现，读为 0。

10.7 PORTF、 LATF 和 TRISF 寄存器

PORTF 是一个 7 位宽的双向端口，对应的数据方向和输出锁存寄存器是 TRISF 和 LATF。PORTF 上的所有引脚都配有施密特触发器输入缓冲器。每个引脚都可被单独配置为输入或输出。

PORTF 与模拟外设功能以及 LCD 段复用。可以通过设置 CMCON 寄存器中的相应位，将 RF1 到 RF6 引脚用作比较器输入或输出。要将 RF<6:3> 用作数字输入，还必须关闭比较器。

注 1： 器件复位时，引脚 RF<6:1> 被配置为模拟输入并读为 0。

2： 要将 PORTF 配置为数字 I/O，可以关闭比较器并设置 ADCON1 的值。

PORTF 还与 LCD 段驱动复用，由 LCDSE2 和 LCDSE3 寄存器中的位控制。仅当禁止该段时，才能使用 I/O 端口功能。

例 10-6： 初始化 PORTF

```
CLRF    PORTF   ; Initialize PORTF by
                ; clearing output
                ; data latches
CLRF    LATF    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   07h     ;
MOVWF   CMCON   ; Turn off comparators
MOVLW   0Fh     ;
MOVWF   ADCON1  ; Set PORTF as digital I/O
MOVLW   0CEh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISF   ; Set RF3:RF1 as inputs
                ; RF5:RF4 as outputs
                ; RF7:RF6 as inputs
```

表 10-14： PORTF 功能

引脚名称	功能	TRIS 设置	I/O	I/O 类型	说明
RF1/AN6/C2OUT/ SEG19	RF1	0	O	DIG	LATF<1> 数据输出；不受模拟输入影响。
		1	I	ST	PORTF<1> 数据输入；当使能模拟输入时被禁止。
	AN6	1	I	ANA	A/D 输入通道 6。 POR 时的默认配置。
	C2OUT	0	O	DIG	比较器 2 的输出；优先于端口数据。
	SEG19	x	O	ANA	LCD 段 19 的输出；禁止所有其他引脚功能。
RF2/AN7/C1OUT/ SEG20	RF2	0	O	DIG	LATF<2> 数据输出；不受模拟输入影响。
		1	I	ST	PORTF<2> 数据输入；当使能模拟输入时被禁止。
	AN7	1	I	ANA	A/D 输入通道 7。 POR 时的默认配置。
	C1OUT	0	O	DIG	比较器 1 的输出；优先于端口数据。
	SEG20	x	O	ANA	LCD 段 20 的输出；禁止所有其他引脚功能。
RF3/AN8/SEG21/ C2INB	RF3	0	O	DIG	LATF<3> 数据输出；不受模拟输入影响。
		1	I	ST	PORTF<3> 数据输入；当使能模拟输入时被禁止。
	AN8	1	I	ANA	A/D 输入通道 8 和比较器 C2+ 输入。 POR 时的默认输入配置；不受模拟输出影响。
	SEG21	x	O	ANA	LCD 段 21 的输出；禁止所有其他引脚功能。
	C2INB	1	I	ANA	比较器 2 的输入 B。
RF4/AN9/SEG22/ C2INA	RF4	0	O	DIG	LATF<4> 数据输出；不受模拟输入影响。
		1	I	ST	PORTF<4> 数据输入；当使能模拟输入时被禁止。
	AN9	1	I	ANA	A/D 输入通道 9 和比较器 C2- 输入。 POR 时的默认输入配置；不影响数字输出。
	SEG22	x	O	ANA	LCD 段 22 的输出；禁止所有其他引脚功能。
	C2INA	1	I	ANA	比较器 2 的输入 A。
RF5/AN10/CVREF/ SEG23/C1INB	RF5	0	O	DIG	LATF<5>数据输出；不受模拟输入影响。当使能CVREF输出时被禁止。
		1	I	ST	PORTF<5> 数据输入；当使能模拟输入时被禁止。当使能 CVREF 输出时被禁止。
	AN10	1	I	ANA	A/D 输入通道 10 和比较器 C1+ 输入。 POR 时的默认输入配置。
	CVREF	x	O	ANA	比较器参考电压输出。使能该功能将禁止数字 I/O。
	SEG23	x	O	ANA	LCD 段 23 的输出；禁止所有其他引脚功能。
	C1INB	1	I	ANA	比较器 1 的输入 B。
RF6/AN11/SEG24/ C1INA	RF6	0	O	DIG	LATF<6> 数据输出；不受模拟输入影响。
		1	I	ST	PORTF<6> 数据输入；当使能模拟输入时被禁止。
	AN11	1	I	ANA	A/D 输入通道 11 和比较器 C1- 输入。POR 时的默认输入配置；不影响数字输出。
	SEG24	x	O	ANA	LCD 段 24 的输出；禁止所有其他引脚功能。
	C1INA	1	I	ANA	比较器 1 的输入 A。
RF7/AN5/$\overline{SS}$/ SEG25	RF7	0	O	DIG	LATF<7> 数据输出；不受模拟输入影响。
		1	I	ST	PORTF<7> 数据输入；当使能模拟输入时被禁止。
	AN5	1	I	ANA	A/D 输入通道 5。 POR 时的默认配置。
	$\overline{SS}$	1	I	TTL	MSSP 模块的从选择输入。
	SEG25	x	O	ANA	LCD 段 25 的输出；禁止所有其他引脚功能。

图注： O = 输出， I = 输入， ANA = 模拟信号， DIG = 数字输出， ST = 施密特触发器缓冲器输入，
TTL = TTL 缓冲器输入， x = 无关位 （TRIS 位不影响端口方向或在此可忽略）。

表 10-15：　　与 PORTF 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
PORTF	RF7	RF6	RF5	RF4	RF3	RF2	RF1	—	62
LATF	LATF7	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	—	62
TRISF	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	—	62
ADCON1	TRIGSEL	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	61
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	61
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	61
LCDSE2	SE23	SE22	SE21	SE20	SE19	SE18	SE17	SE16	61
LCDSE3	SE31	SE30	SE29	SE28	SE27	SE26	SE25	SE24	61

图注：　　— = 未实现，读为 0。 PORTF 不使用阴影单元。

10.8 PORTG、TRISG 和 LATG 寄存器

PORTG 是一个 5 位宽的双向端口，对应的数据方向和输出锁存寄存器是 TRISG 和 LATG。PORTG 上的所有引脚都是仅数字功能引脚，并且可以承受最高 5.5V 的电压。

PORTG 与 AUSART 和 LCD 功能复用（表 10-16）。当用作 I/O 时，所有 PORTG 引脚都配有施密特触发器输入缓冲器。当 AUSART 有效时， RG1 引脚还可配置为漏极开路输出。通过将 U2OD 控制位（LATG<7>）置 1 选择漏极开路配置。

RG4 与 LCD 段驱动复用，由 LCDSE2 寄存器中的位控制，并作为 RTCC 引脚。仅当禁止该段时，才能使用 I/O 端口功能。

RG3 和 RG2 与 LCD 电荷泵的 VLCAP 引脚复用， RG0 与 LCDBIAS0 偏置电压输入复用。当这些引脚用于产生 LCD 偏置， I/O 和其他功能不可用。

当使能外设功能时，应小心定义每个 PORTG 引脚的 TRIS 位。有些外设会改写 TRIS 位的设置，将引脚定义为输出引脚或输入引脚。用户应该查阅相应的外设章节来正确设置 TRIS 位。引脚改写值不装入 TRIS 寄存器中。这将允许对 TRIS 寄存器执行读 - 修改 - 写操作而无需担心外设的改写。

尽管端口本身只有 5 位宽，PORTG<7:5> 位仍然可以控制与 PORTD、PORTE 和 PORTJ 相关的 I/O 端口上的弱上拉。将这些位清零可使能各自端口的上拉功能。所有器件复位时均默认禁止所有的上拉功能。

TRISG 和 LATG 大多数相应位实现为 CCP1、 CCP2、SPI（TRISG<7:5>）以及 USART（LATG<7:6>）的漏极开路控制位。通过将这些位置 1 可将对应外设的输出引脚配置为漏极开路操作。未实现 LATG<5>。

例 10-7： 初始化 PORTG

```
CLRF    PORTG   ; Initialize PORTG by
                ; clearing output
                ; data latches
CLRF    LATG    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   04h     ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISG   ; Set RG1:RG0 as outputs
                ; RG2 as input
                ; RG4:RG3 as inputs
```

表 10-16： PORTG 功能

引脚名称	功能	TRIS 设置	I/O	I/O 类型	说明
RG0/LCDBIAS0	RG0	0	O	DIG	LATG<0> 数据输出。
		1	I	ST	PORTG<0> 数据输入。
	LCDBIAS0	x	I	ANA	LCD 模块偏置电压输入。
RG1/TX2/CK2	RG1	0	O	DIG	LATG<1> 数据输出。
		1	I	ST	PORTG<1> 数据输入。
	TX2	1	O	DIG	同步串行数据输出（AUSART 模块）；优先于端口数据。
	CK2	1	O	DIG	同步串行数据输入（AUSART 模块）；用户必须将其配置为输入。
		1	I	ST	同步串行时钟输入（AUSART 模块）。
RG2/RX2/DT2/ VLCAP1	RG2	0	O	DIG	LATG<2> 数据输出。
		1	I	ST	PORTG<2> 数据输入。
	RX2	1	I	ST	异步串行接收数据输入（ASART 模块）。
	DT2	1	O	DIG	同步串行数据输出（AUSART 模块）；优先于端口数据。
		1	I	ST	同步串行数据输入（AUSART 模块）；用户必须将其配置为输入。
	VLCAP1	x	I	ANA	LCD 电荷泵电容输入。
RG3/VLCAP2	RG3	0	O	DIG	LATG<3> 数据输出。
		1	I	ST	PORTG<3> 数据输入。
	VLCAP2	x	I	ANA	LCD 电荷泵电容输入。
RG4/SEG26/ RTCC	RG4	0	O	DIG	LATG<4> 数据输出。
		1	I	ST	PORTG<4> 数据输入。
	SEG26	x	O	ANA	LCD 段 26 的输出；禁止所有其他引脚功能。
	RTCC	x	O	DIG	RTCC 输出。

图注： O = 输出， I = 输入， ANA = 模拟信号， DIG = 数字输出， ST = 施密特触发器缓冲器输入，
x = 无关位（TRIS 位不影响端口方向或在此可忽略）。

表 10-17： 与 PORTG 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
PORTG	RDPU	REPU	RJPU(1)	RG4	RG3	RG2	RG1	RG0	62
LATG	U2OD	U1OD	—	LATG4	LATG3	LATG2	LATG1	LATG0	62
TRISG	SPIOD	CCP2OD	CCP1OD	TRISG4	TRISG3	TRISG2	TRISG1	TRISG0	62
LCDSE3	SE31	SE30	SE29	SE28	SE27	SE26	SE25	SE24	61

图注： — = 未实现，读为 0。 PORTG 不使用阴影单元。
注 1： 在 PIC18F6XJ90 器件上未实现，读为 0。

10.9 PORTH、 LATH 和 TRISH 寄存器

注： PORTH 仅在 PIC18F8XJ90 器件上可用。

PORTH 是一个 8 位宽的双向 I/O 端口，对应的数据方向和输出锁存寄存器是 TRISH 和 LATH。所有引脚都是仅数字功能引脚，并且可以承受最高 5.5V 的电压。

PORTH 上的所有引脚都配有施密特触发器输入缓冲器。每个引脚都可被单独配置为输入或输出。

所有 PORTH 引脚与 LCD 段驱动复用，由 LCDSE5 寄存器控制。仅当禁止该段时，才能使用 I/O 端口功能。

例 10-8： 初始化 PORTH

```
CLRF    PORTH       ; Initialize PORTH by
                    ; clearing output
                    ; data latches
CLRF    LATH        ; Alternate method
                    ; to clear output
                    ; data latches
MOVLW   0Fh         ; Configure PORTH as
MOVWF   ADCON1      ; digital I/O
MOVLW   0CFh        ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISH       ; Set RH3:RH0 as inputs
                    ; RH5:RH4 as outputs
                    ; RH7:RH6 as inputs
```

表 10-18： PORTH 功能

引脚名称	功能	TRIS 设置	I/O	I/O 类型	说明
RH0/SEG47	RH0	0	O	DIG	LATH<0> 数据输出。
		1	I	ST	PORTH<0> 数据输入。
	SEG47	x	O	ANA	LCD 段 47 的输出；禁止所有其他引脚功能。
RH1/SEG46	RH1	0	O	DIG	LATH<1> 数据输出。
		1	I	ST	PORTH<1> 数据输入。
	SEG46	x	O	ANA	LCD 段 46 的输出；禁止所有其他引脚功能。
RH2/SEG45	RH2	0	O	DIG	LATH<2> 数据输出。
		1	I	ST	PORTH<2> 数据输入。
	SEG45	x	O	ANA	LCD 段 45 的输出；禁止所有其他引脚功能。
RH3/SEG44	RH3	0	O	DIG	LATH<3> 数据输出。
		1	I	ST	PORTH<3> 数据输入。
	SEG44	x	O	ANA	LCD 段 44 的输出；禁止所有其他引脚功能。
RH4/SEG40	RH4	0	O	DIG	LATH<4> 数据输出。
		1	I	ST	PORTH<4> 数据输入。
	SEG40	x	O	ANA	LCD 段 40 的输出；禁止所有其他引脚功能。
RH5/SEG41	RH5	0	O	DIG	LATH<5> 数据输出。
		1	I	ST	PORTH<5> 数据输入。
	SEG41	x	O	ANA	LCD 段 41 的输出；禁止所有其他引脚功能。
RH6/SEG42	RH6	0	O	DIG	LATH<6> 数据输出。
		1	I	ST	PORTH<6> 数据输入。
	SEG42	x	O	ANA	LCD 段 42 的输出；禁止所有其他引脚功能。
RH7/SEG43	RH7	0	O	DIG	LATH<7> 数据输出。
		1	I	ST	PORTH<7> 数据输入。
	SEG43	x	O	ANA	LCD 段 43 的输出；禁止所有其他引脚功能。

图注： O = 输出， I = 输入， ANA = 模拟信号， DIG = 数字输出， ST = 施密特触发器缓冲器输入，
x = 无关位 （TRIS 位不影响端口方向或在此可忽略）。

表 10-19： 与 PORTH 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
PORTH	RH7	RH6	RH5	RH4	RH3	RH2	RH1	RH0	62
LATH	LATH7	LATH6	LATH5	LATH4	LATH3	LATH2	LATH1	LATH0	62
TRISH	TRISH7	TRISH6	TRISH5	TRISH4	TRISH3	TRISH2	TRISH1	TRISH0	62
LCDSE5	SE47	SE46	SE45	SE44	SE43	SE42	SE41	SE40	61

10.10 PORTJ、 TRISJ 和 LATJ 寄存器

注： PORTJ 仅在 PIC18F8XJ90 器件上可用。

PORTJ 是一个 8 位宽的双向端口，对应的数据方向和输出锁存寄存器是 TRISJ 和 LATJ。PORTJ 上的所有引脚都是仅数字功能引脚，并且可以承受最高 5.5V 的电压。

PORTJ 上的所有引脚都配有施密特触发器输入缓冲器。每个引脚都可被单独配置为输入或输出。

注： 这些引脚在任何器件复位时都被配置为数字输入引脚。

除 RJ0 外的所有 PORTJ 引脚均与 LCD 段驱动复用，由 LCDSE4 寄存器控制。仅当禁止这些段时，才能使用这些引脚的 I/O 端口功能。

PORTJ 的每个引脚都具有内部弱上拉。提供上拉是为了保证上电时外扩存储器接口输入为一个已知状态。可通过单个控制位关闭所有上拉。可以通过清零 RJPU 位（PORTG<5>）来实现。当端口引脚被配置为输出时，其弱上拉功能会自动关闭。发生任何器件复位时，上拉功能会被禁止。

例 10-9： 初始化 PORTJ

```
CLRF    PORTJ   ; Initialize PORTJ by
                ; clearing output latches
CLRF    LATJ    ; Alternate method
                ; to clear output latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISJ   ; Set RJ3:RJ0 as inputs
                ; RJ5:RJ4 as output
                ; RJ7:RJ6 as inputs
```

表 10-20： PORTJ 功能

引脚名称	功能	TRIS 设置	I/O	I/O 类型	说明
RJ0	RJ0	0	O	DIG	LATJ<0> 数据输出。
		1	I	ST	PORTJ<0> 数据输入。
RJ1/SEG33	RJ1	0	O	DIG	LATJ<1> 数据输出。
		1	I	ST	PORTJ<1> 数据输入。
	SEG33	x	O	ANA	LCD 段 33 的输出；禁止所有其他引脚功能。
RJ2/SEG34	RJ2	0	O	DIG	LATJ<2> 数据输出。
		1	I	ST	PORTJ<2> 数据输入。
	SEG34	x	O	ANA	LCD 段 34 的输出；禁止所有其他引脚功能。
RJ3/SEG35	RJ3	0	O	DIG	LATJ<3> 数据输出。
		1	I	ST	PORTJ<3> 数据输入。
	SEG35	x	O	ANA	LCD 段 35 的输出；禁止所有其他引脚功能。
RJ4/SEG39	RJ4	0	O	DIG	LATJ<4> 数据输出。
		1	I	ST	PORTJ<4> 数据输入。
	SEG39	x	O	ANA	LCD 段 39 的输出；禁止所有其他引脚功能。
RJ5/SEG38	RJ5	0	O	DIG	LATJ<5> 数据输出。
		1	I	ST	PORTJ<5> 数据输入。
	SEG38	x	O	ANA	LCD 段 38 的输出；禁止所有其他引脚功能。
RJ6/SEG37	RJ6	0	O	DIG	LATJ<6> 数据输出。
		1	I	ST	PORTJ<6> 数据输入。
	SEG37	x	O	ANA	LCD 段 37 的输出；禁止所有其他引脚功能。
RJ7/SEG36	RJ7	0	O	DIG	LATJ<7> 数据输出。
		1	I	ST	PORTJ<7> 数据输入。
	SEG36	x	O	ANA	LCD 段 36 的输出；禁止所有其他引脚功能。

图注： O = 输出， I = 输入， ANA = 模拟信号， DIG = 数字输出， ST = 施密特触发器缓冲器输入，
x = 无关位 （TRIS 位不影响端口方向或在此可忽略）。

表 10-21： 与 PORTJ 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
PORTJ	RJ7	RJ6	RJ5	RJ4	RJ3	RJ2	RJ1	RJ0	62
LATJ	LATJ7	LATJ6	LATJ5	LATJ4	LATJ3	LATJ2	LATJ1	LATJ0	62
TRISJ	TRISJ7	TRISJ6	TRISJ5	TRISJ4	TRISJ3	TRISJ2	TRISJ1	TRISJ0	62
PORTG	RDPU	REPU	RJPU(1)	RG4	RG3	RG2	RG1	RG0	62
LCDSE4	SE39	SE38	SE37	SE36	SE35	SE34	SE33	SE32	61

图注： PORTJ 不使用阴影单元。

注 1： 在 PIC18F6XJ90 器件上未实现，读为 0。

11.0 TIMER0 模块

Timer0 模块具有以下特性：

- 可由软件选择作为 8 位或 16 位定时器 / 计数器
- 可读写寄存器
- 专用的 8 位软件可编程预分频器
- 可选的时钟源 （内部或外部）
- 外部时钟的边沿选择
- 溢出时产生中断

T0CON 寄存器 （寄存器 11-1）控制该模块操作的所有方面，包括预分频比的选择；它是可读写的。

图 11-1 给出了 8 位模式下 Timer0 模块的简化框图。图 11-2 给出了 16 位模式下 Timer0 模块的简化框图。

寄存器 11-1： T0CON：TIMER0 控制寄存器

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
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 **TMR0ON**：Timer0 开 / 关控制位
1 = 使能 Timer0
0 = 停止 Timer0

bit 6 **T08BIT**：Timer0 8 位 /16 位控制位
1 = Timer0 被配置为 8 位定时器 / 计数器
0 = Timer0 被配置为 16 位定时器 / 计数器

bit 5 **T0CS**：Timer0 时钟源选择位
1 = T0CKI 引脚上的电平跳变
0 = 内部指令周期时钟 （CLKO）

bit 4 **T0SE**：Timer0 时钟源边沿选择位
1 = 在 T0CKI 引脚信号从高至低跳变时，递增计数
0 = 在 T0CKI 引脚信号从低至高跳变时，递增计数

bit 3 **PSA**：Timer0 预分频器分配位
1 = 未分配 Timer0 预分频器。 Timer0 时钟输入不经预分频器分频。
0 = 已分配 Timer0 预分频器。 Timer0 时钟输入来自预分频器的输出。

bit 2-0 **T0PS<2:0>**：Timer0 预分频比选择位
111 = 1:256 预分频比
110 = 1:128 预分频比
101 = 1:64 预分频比
100 = 1:32 预分频比
011 = 1:16 预分频比
010 = 1:8 预分频比
001 = 1:4 预分频比
000 = 1:2 预分频比

11.1 Timer0 工作原理

Timer0 既可用作定时器也可用作计数器。可以通过 T0CS 位（T0CON<5>）来选择模式。在定时器模式下（T0CS = 0），该模块在每个时钟周期都会递增（默认情况下），除非选择了其他预分频值（见**第 11.3 节 “预分频器”**）。在对 TMR0 寄存器执行写操作之后的两个指令周期内 Timer0 禁止递增。用户可通过将调整值写入 TMR0 寄存器来解决这一问题。

通过将 T0CS 位置 1（= 1）选择计数器模式。在该模式下，Timer0 可在 RA4/T0CKI 引脚信号的每个上升沿或下降沿递增。递增边沿由 Timer0 时钟源边沿选择位 T0SE（T0CON<4>）决定。清零该位即选择上升沿。下面讨论外部时钟输入的限制条件。

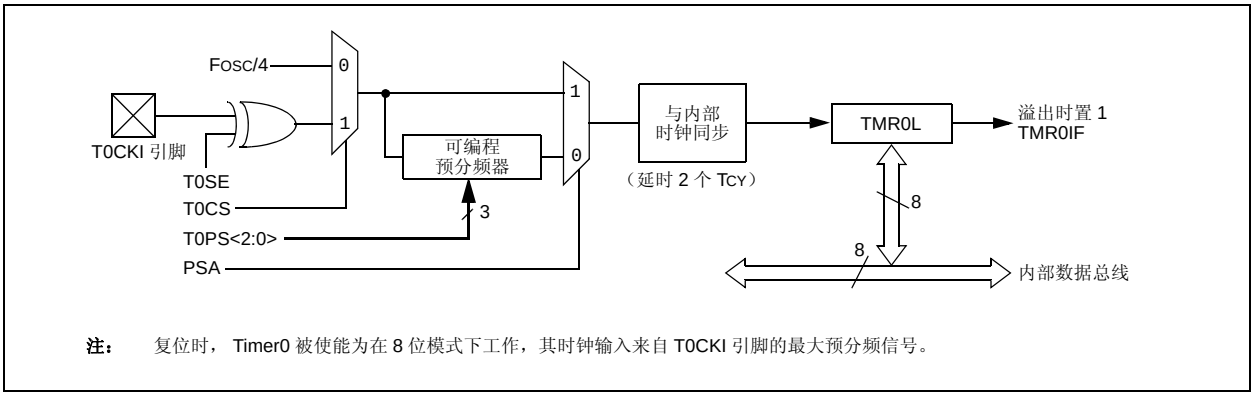
可以使用外部时钟源来驱动 Timer0。但是，必须满足一定要求，以确保外部时钟和内部相位时钟（TOSC）保持同步。在同步之后，定时器 / 计数器需要一定的延时才开始递增。

11.2 16 位模式下 Timer0 的读写操作

TMR0H 并不是 16 位模式下 Timer0 的实际高字节，而是 Timer0 实际高字节的缓存形式，不可以被直接读写（见图 11-2）。在读 TMR0L 时使用 Timer0 高字节的内容更新 TMR0H。这种方式使用户可以读取 Timer0 的全部 16 位，而不需要验证高字节和低字节读取的有效性。由于高字节和低字节连续读取之间的计满返回，可能会产生无效读取。

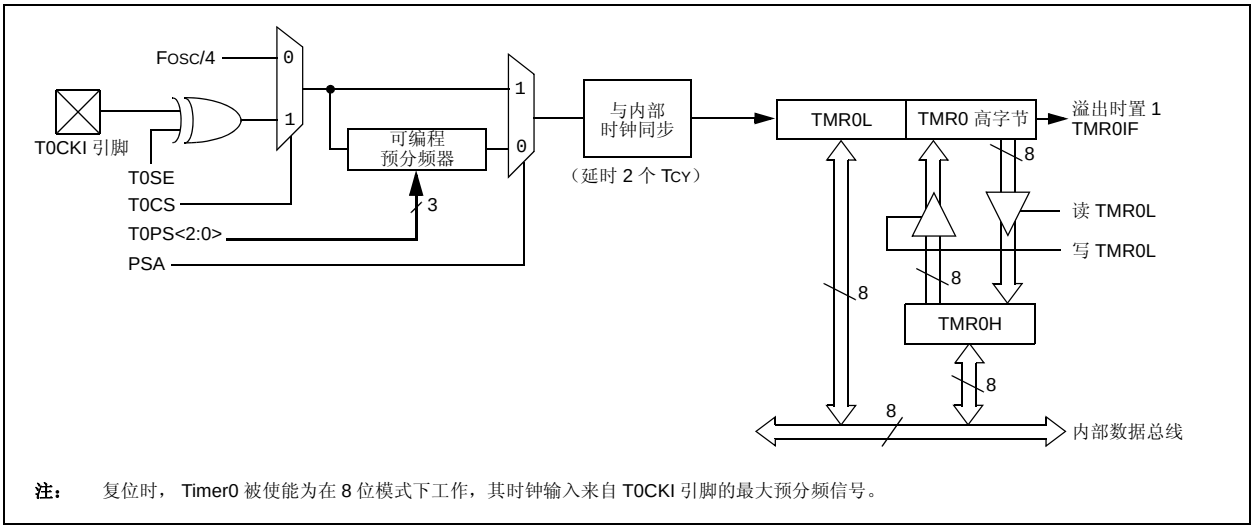
同样，写入 Timer0 的高字节也必须通过 TMR0H 缓冲寄存器来操作。在写入 TMR0L 的同时，使用 TMR0H 的内容更新 Timer0 的高字节。这样一次就可以完成 Timer0 全部 16 位的更新。

图 11-1： TIMER0 框图（8 位模式）



注： 复位时，Timer0 被使能为在 8 位模式下工作，其时钟输入来自 T0CKI 引脚的最大预分频信号。

图 11-2： TIMER0 框图（16 位模式）



注： 复位时，Timer0 被使能为在 8 位模式下工作，其时钟输入来自 T0CKI 引脚的最大预分频信号。

11.3 预分频器

Timer0 模块的预分频器为一个 8 位计数器。该预分频器不可直接读写。通过 PSA 和 T0PS<2:0> 位（T0CON <3:0>）进行预分频器的分配和设定预分频比。

将 PSA 位清零可将预分频器分配给 Timer0 模块。预分频值可以在 1:2 到 1:256 之间进行选择，以 2 的整数次幂递增。

如果将预分频器分配给 Timer0 模块，所有写入 TMR0 寄存器的指令（例如，`CLRF TMR0`、`MOVWF TMR0` 和 `BSF TMR0` 等）都会将预分频器的计数值清零。

注：	如果将预分频器分配给 Timer0，写入 TMR0 会将预分频器的计数值清零，但不会改变预分频器的分配。

11.3.1 切换预分频器的分配

预分频器的分配完全由软件控制，并且在程序执行期间可以随时更改。

11.4 Timer0 中断

8 位模式下 TMR0 寄存器从 FFh 溢出到 00h，或 16 位模式下 TMR0 从 FFFFh 溢出到 0000h 时，将产生 TMR0 中断。这种溢出会将 TMR0IF 标志位置 1。可以通过清零 TMR0IE 位 （INTCON<5>）来屏蔽该中断。在重新允许该中断前，必须在中断服务程序中用软件清零 TMR0IF 位。

由于 Timer0 在休眠模式下是关闭的，所以 TMR0 中断无法将处理器从休眠状态唤醒。

表 11-1： 与 TIMER0 相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
TMR0L	Timer0 寄存器的低字节								60
TMR0H	Timer0 寄存器的高字节								60
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	60
TRISA	TRISA7(1)	TRISA6(1)	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	62

图注： — = 未实现，读为 0。 Timer0 不使用阴影单元。

注 1： 仅当内部振荡器被选为默认时钟源 （FOSC2 配置位 = 0）时， RA<7:6> 及其相关的锁存和方向位才能被配置为端口引脚；否则，这些位将被禁止且读为 0。

注：

12.0 TIMER1 模块

Timer1 定时器 / 计数器模块具有以下特性：

- 可由软件选择作为 16 位定时器或计数器
- 可读写的 8 位寄存器 （TMR1H 和 TMR1L）
- 可选择器件时钟或 Timer1 内部振荡器作为时钟源（内部或外部）
- 溢出时产生中断
- 由 CCP 特殊事件触发信号复位
- 器件时钟状态标志位 （T1RUN）

图 12-1 给出了 Timer1 模块的简化框图。图 12-2 给出了此模块在读写模式下的工作原理框图。

此模块自身带有低功耗振荡器，可提供额外的时钟选项。Timer1 振荡器也可作为单片机处于节能状态时的低功耗时钟源。

仅需极少外部元件和代码开销，Timer1 就可为应用提供实时时钟 （RTC）功能。

Timer1 由 T1CON 控制寄存器 （寄存器 12-1）控制。该寄存器还包含 Timer1 振荡器使能位 （T1OSCEN）。可以通过将控制位 TMR1ON （T1CON<0>）置 1 或清零来使能或禁止 Timer1。

寄存器 12-1： T1CON：TIMER1 控制寄存器

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
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 **RD16**：16 位读 / 写模式使能位
1 = 使能 Timer1 通过一次 16 位操作进行寄存器读 / 写
0 = 使能 Timer1 通过两次 8 位操作进行寄存器读 / 写

bit 6 **T1RUN**：Timer1 系统时钟状态位
1 = 器件时钟由 Timer1 振荡器产生
0 = 器件时钟由另一个时钟源产生

bit 5-4 **T1CKPS<1:0>**：Timer1 输入时钟预分频比选择位
11 = 1:8 预分频比
10 = 1:4 预分频比
01 = 1:2 预分频比
00 = 1:1 预分频比

bit 3 **T1OSCEN**：Timer1 振荡器使能位
1 = 使能 Timer1 振荡器
0 = 关闭 Timer1 振荡器
关闭振荡器的反相器和反馈电阻以减少功耗。

bit 2 **$\overline{\text{T1SYNC}}$**：Timer1 外部时钟输入同步选择位
当 TMR1CS = 1 时：
1 = 不同步外部时钟输入
0 = 同步外部时钟输入
当 TMR1CS = 0 时：
该位被忽略。当 TMR1CS = 0 时， Timer1 使用内部时钟。

bit 1 **TMR1CS**：Timer1 时钟源选择位
1 = 使用 RC0/T1OSO/T13CKI 引脚上的外部时钟 （上升沿计数）
0 = 内部时钟 （FOSC/4）

bit 0 **TMR1ON**：Timer1 使能位
1 = 使能 Timer1
0 = 停止 Timer1

12.1 Timer1 工作原理

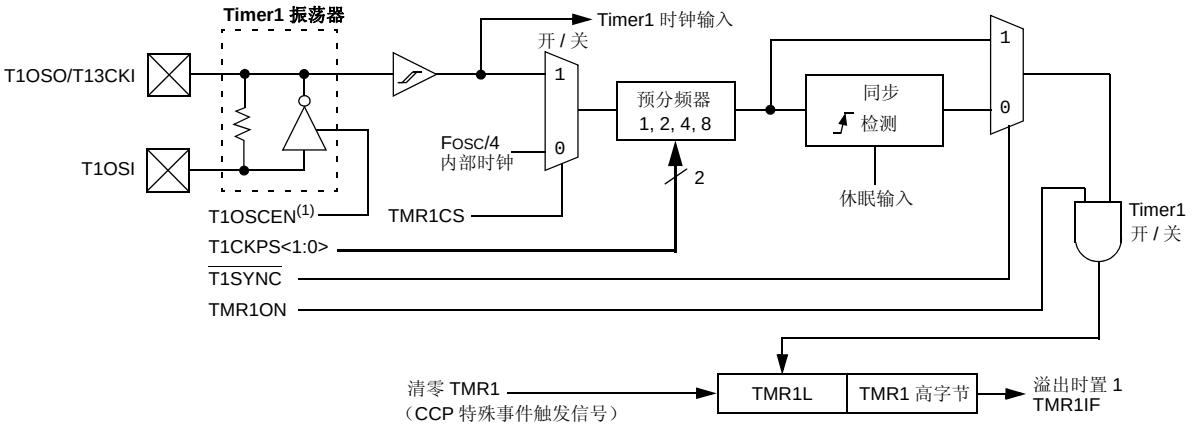
Timer1 可工作在以下模式之一：

- 定时器
- 同步计数器
- 异步计数器

工作模式由时钟选择位 TMR1CS（T1CON<1>）决定。当 TMR1CS 清零 （= 0）时， Timer1 在每个内部指令周期（Fosc/4）递增。当该位置 1 时，Timer1 在 Timer1 外部时钟输入信号或 Timer1 振荡器输出信号 （如果使能）的每个上升沿递增。

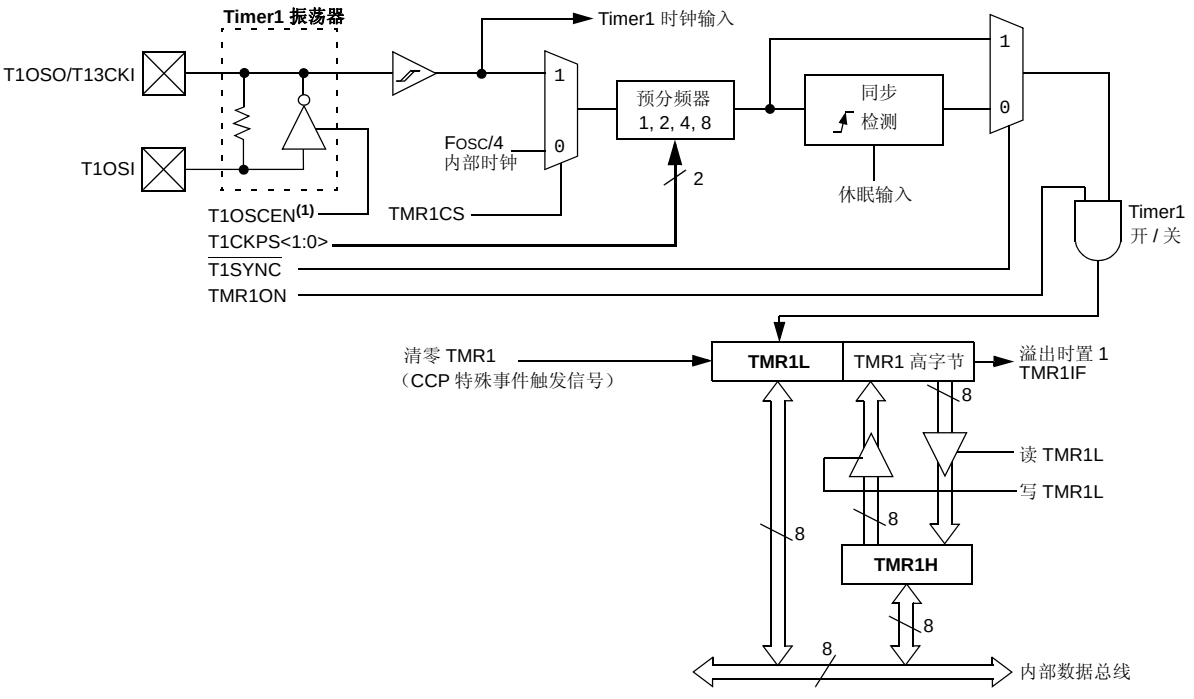
当使能 Timer1 时，RC1/T1OSI 和 RC0/T1OSO/T13CKI 引脚变为输入引脚。这意味着 TRISC<1:0> 的值将被忽略并且这些引脚将读为 0。

图 12-1： TIMER1 框图 （8 位模式）



注 1：当使能位 T1OSCEN 清零时，将关闭振荡器的反相器和反馈电阻以减少功耗。

图 12-2： TIMER1 框图 （16 位读 / 写模式）



注 1：当使能位 T1OSCEN 清零时，将关闭振荡器的反相器和反馈电阻以减少功耗。

12.2 Timer1 的 16 位读 / 写模式

可将 Timer1 配置为 16 位读写模式 （见图 12-2）。当 RD16 控制位 （T1CON<7>）置 1 时， TMR1H 的地址被映射到 Timer1 的高字节缓冲寄存器。读 TMR1L 将把 Timer1的高字节的内容装入Timer1高字节缓冲寄存器。这种方式使用户可以精确地读取 Timer1 的全部 16 位，而不需要像先读高字节再读低字节那样，由于两次读取之间可能存在计满返回，而不得不验证读取的有效性。

写 Timer1 的高字节也必须通过 TMR1H 缓冲寄存器进行。在写入 TMR1L 的同时，使用 TMR1H 的内容更新 Timer1 的高字节。这样允许用户将所有 16 位一次写入 Timer1 的高字节和低字节。

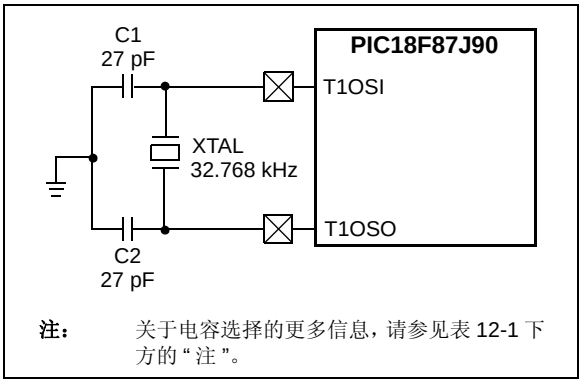
在该模式下不能直接读写 Timer1 的高字节。所有读写都必须通过 Timer1 高字节缓冲寄存器来进行。写入 TMR1H 不会清零 Timer1 预分频器。只有在写 TMR1L 时才会清零该预分频器。

12.3 Timer1 振荡器

片上晶振电路连接在 T1OSI（输入）引脚和 T1OSO（放大器输出）引脚之间。可以通过将 Timer1 振荡器使能位 T1OSCEN （T1CON<3>）置 1 来使能该振荡器电路。该振荡器电路是一种低功耗电路，它采用了额定振荡频率为 32 kHz 的晶振。在所有功耗管理模式下都可继续运行。图 12-3 所示为典型的 LP 振荡器电路。表 12-1 给出了供 Timer1 振荡器选择的电容值。

用户必须提供软件延时来确保 Timer1 振荡器的正常起振。

图 12-3： TIMER1 LP振荡器的外部元件



注： 关于电容选择的更多信息，请参见表 12-1 下方的 “ 注 ”。

表 12-1： TIMER1 振荡器的电容选择 (2,3,4)

振荡器类型	频率	C1	C2
LP	32.768 kHz	27 pF(1)	27 pF(1)

注 1： Microchip 建议仅将这些值作为验证振荡器电路的起始点。

2： 电容越大，振荡器越稳定，但起振时间越长。

3： 因为每种谐振器 / 晶振都有其自身特性，用户应当向谐振器 / 晶振制造厂商询问外部元件的适当值。

4： 上述电容值仅供设计参考。

12.3.1 使用 TIMER1 作为时钟源

在功耗管理模式下也可以将Timer1振荡器用作时钟源。通过将系统时钟选择位 SCS<1:0> （OSCCON<1:0>）设置为 01，器件可以切换到 SEC_RUN 模式； CPU 和外设都可以用 Timer1 振荡器作为时钟源。如果 IDLEN 位 （OSCCON<7>）被清零并且执行了 SLEEP 指令，器件将进入 SEC_IDLE 模式。更多详细信息，请参见**第 4.0 节 “ 功耗管理模式 ”**。

无论何时将 Timer1 振荡器用作时钟源， Timer1 系统时钟状态标志位 T1RUN （T1CON<6>）均会置 1。这可用于确定控制器的当前时钟模式。该位也可指示故障保护时钟监视器当前正使用的时钟源。如果使能了时钟监视器并且Timer1振荡器在提供时钟信号时发生了故障，查询T1RUN位可以确定时钟源是Timer1振荡器还是其他时钟源。

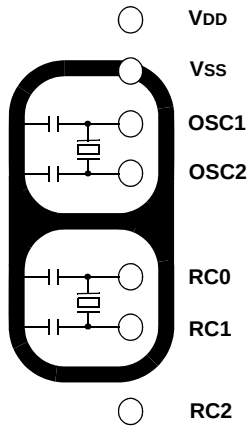
12.3.2 TIMER1 振荡器布线注意事项

Timer1 振荡器电路在工作期间仅消耗极小的电流。鉴于此振荡器的低功耗特性，它对附近变化较快的信号比较敏感。

如图 12-3 所示，振荡器电路应该尽可能靠近单片机。除了 VSS 或 VDD 外，在该振荡器电路边界内不应有其他电路通过。

对于单面 PCB，如果必须要在该振荡器附近布高速电路（如输出比较模式或 PWM 模式下的 CCP1 引脚，或使用 OSC2 引脚的主振荡器），那么在该振荡器电路周围布接地保护环（如图 12-4 所示），或再加一个地平面可能会有帮助。

图 12-4： 带有接地保护环的振荡器电路



注： 未按比例绘制。

12.4 Timer1 中断

TMR1 寄存器对 （TMR1H:TMR1L）从 0000h 递增到 FFFFh，然后计满返回到 0000h 重新开始计数。如果允许了 Timer1 中断，则溢出时会产生 Timer1 中断，锁存到中断标志位 TMR1IF （PIR1<0>）中。可以通过将 Timer1 中断允许位 TMR1IE（PIE1<0>）置 1 或清零来允许或禁止该中断。

12.5 使用 CCP 特殊事件触发信号复位 Timer1

如果 CCP1 或 CCP2 配置为在比较模式 （CCPxM <3:0> = 1011）下使用 Timer1 并产生特殊事件触发信号，该信号将复位 Timer1。如果使能了 A/D 模块，来自 CCP2 的触发信号还将启动 A/D 转换 （更多信息，请参见**第 16.3.4 节 “ 特殊事件触发器 ”**）。

要使用这一功能，必须将模块配置为定时器或同步计数器。在这种情况下，CCPRxH:CCPRxL 寄存器对实际上变成了 Timer1 的周期寄存器。

如果 Timer1 在异步计数器模式下运行，复位操作可能不起作用。

如果对Timer1的写操作和特殊事件触发信号同时发生，则写操作优先。

注：	CCPx模块产生的特殊事件触发信号不会将 TMR1IF 中断标志位 （PIR1<0>）置 1。

12.6 使用 Timer1 作为实时时钟

为 Timer1 外接一个 LP 振荡器 （如**第 12.3 节 “Timer1 振荡器 ”** 中所述），可以允许用户在他们的应用中包含 RTC 功能。只需通过一个提供精确时基的廉价时钟晶振以及几行计算时间的应用代码就可实现这一功能。当器件在休眠模式下工作并使用电池或超级电容作为电源时，可省去另外的 RTC 器件和备用电池。

应用代码程序 RTCisr（如例 12-1 所示），演示了使用中断服务程序以 1 秒的间隔递增计数器的简单方法。将 TMR1 寄存器对的值递增直至溢出将触发中断并调用中断服务程序，该程序会使秒计数器递增 1，其他的分钟和小时计数器则会在前面的计数器溢出时递增 1。

由于寄存器对为 16 位宽，采用 32.768 kHz 时钟源时寄存器需要 2 秒递增计数到溢出。要强制在所需的 1 秒时间间隔溢出，必须预先加载它。最简单的方法是使用 BSF 指令将 TMR1H 的最高有效位置 1。请注意决不要预先加载或改变 TMR1L 寄存器，这样做可能会引起多个周期的累积错误。

要使此方法精确，Timer1 必须工作于异步模式且必须允许 Timer1 溢出中断（PIE1<0> = 1），如程序 RTCinit 所示。同时 Timer1 振荡器也必须被使能并始终运行。

例 12-1：　使用 TIMER1 中断服务程序实现实时时钟

```
RTCinit
            MOVLW   80h             ; Preload TMR1 register pair
            MOVWF   TMR1H           ; for 1 second overflow
            CLRF    TMR1L
            MOVLW   b’00001111’     ; Configure for external clock,
            MOVWF   T1CON           ; Asynchronous operation, external oscillator
            CLRF    secs            ; Initialize timekeeping registers
            CLRF    mins            ;
            MOVLW   .12
            MOVWF   hours
            BSF     PIE1, TMR1IE    ; Enable Timer1 interrupt
            RETURN
RTCisr
            BSF     TMR1H, 7        ; Preload for 1 sec overflow
            BCF     PIR1, TMR1IF    ; Clear interrupt flag
            INCF    secs, F         ; Increment seconds
            MOVLW   .59             ; 60 seconds elapsed?
            CPFSGT  secs
            RETURN                  ; No, done
            CLRF    secs            ; Clear seconds
            INCF    mins, F         ; Increment minutes
            MOVLW   .59             ; 60 minutes elapsed?
            CPFSGT  mins
            RETURN                  ; No, done
            CLRF    mins            ; clear minutes
            INCF    hours, F        ; Increment hours
            MOVLW   .23             ; 24 hours elapsed?
            CPFSGT  hours
            RETURN                  ; No, done
            CLRF    hours           ; Reset hours
            RETURN                  ; Done
```

表 12-2：　与 TIMER1 作为定时器 / 计数器相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	62
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	62
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	62
TMR1L	Timer1 寄存器的低字节								60
TMR1H	Timer1 寄存器的高字节								60
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	60

图注：　Timer1 模块不使用阴影单元。

注：

13.0 TIMER2 模块

Timer2 模块具有以下特性：

- 8 位定时器和周期寄存器（分别为 TMR2 和 PR2）
- 可读写 （以上两个寄存器）
- 可软件编程的预分频比 （1:1、 1:4 和 1:16）
- 可软件编程的后分频比 （1:1 到 1:16）
- TMR2 与 PR2 匹配时产生中断
- 可选择用作 MSSP 模块的移位时钟

此模块由 T2CON 寄存器（寄存器 13-1）控制，此寄存器使能或禁止定时器并配置预分频比和后分频比。可以通过清零控制位 TMR2ON（T2CON<2>）关闭 Timer2，以实现功耗最小。

图 13-1 给出了此模块的简化框图。

13.1 Timer2 工作原理

在正常工作模式下，TMR2 从 00h 开始，每个时钟周期（FOSC/4）递增 1。4 位计数器 / 预分频器提供了对时钟输入不分频、4 分频和 16 分频三种预分频选项，可通过预分频比控制位 T2CKPS<1:0> （T2CON<1:0>）进行选择。在每个时钟周期，TMR2 的值都会与周期寄存器 PR2 中的值进行比较。当两个值匹配时，由比较器产生匹配信号作为定时器的输出。此信号也会使 TMR2 的值在下一个周期复位为 00h，并驱动输出计数器 / 后分频器 （见**第 13.2 节 “Timer2 中断 ”**）。

TMR2 和 PR2 寄存器均可直接读写。在任何器件复位时，TMR2 寄存器都会清零，而 PR2 寄存器则初始化为 FFh。预分频器和后分频器计数器均会在发生以下事件时清零：

- 对 TMR2 寄存器进行写操作
- 对 T2CON 寄存器进行写操作
- 任何器件复位（上电复位、$\overline{\text{MCLR}}$ 复位、看门狗定时器复位或欠压复位）

写 T2CON 时 TMR2 不会清零。

寄存器 13-1： T2CON：TIMER2 控制寄存器

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
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 **未实现：**读为 0

bit 6-3 **T2OUTPS<3:0>：**Timer2 输出后分频比选择位
0000 = 1:1 后分频比
0001 = 1:2 后分频比
•
•
•
1111 = 1:16 后分频比

bit 2 **TMR2ON：**Timer2 使能位
1 = 使能 Timer2
0 = 关闭 Timer2

bit 1-0 **T2CKPS<1:0>：**Timer2 时钟预分频比选择位
00 = 预分频比为 1
01 = 预分频比为 4
1x = 预分频比为 16

13.2 Timer2 中断

Timer2 也可以产生可选的器件中断。 Timer2 输出信号（TMR2 与 PR2 匹配时）为 4 位输出计数器 / 后分频器提供输入。此计数器产生 TMR2 匹配中断，对应的中断标志位为 TMR2IF （PIR1<1>）。可以通过将 TMR2 匹配中断允许位 TMR2IE（PIE1<1>）置 1 来允许此中断。

可以通过后分频比控制位 T2OUTPS<3:0>（T2CON<6:3>）在 16 个后分频比选项 （从 1:1 到 1:16）中选择其一。

13.3 Timer2 输出

TMR2 的不经分频的输出主要用于 CCP 模块，它用作 CCP 模块在 PWM 模式下工作时的时基。

还可选择将 Timer2 用作 MSSP 模块在 SPI 模式下工作时的移位时钟源。**第 18.0 节 “ 主同步串行口（MSSP）模块 ”** 中提供了更多信息。

图 13-1： TIMER2 框图

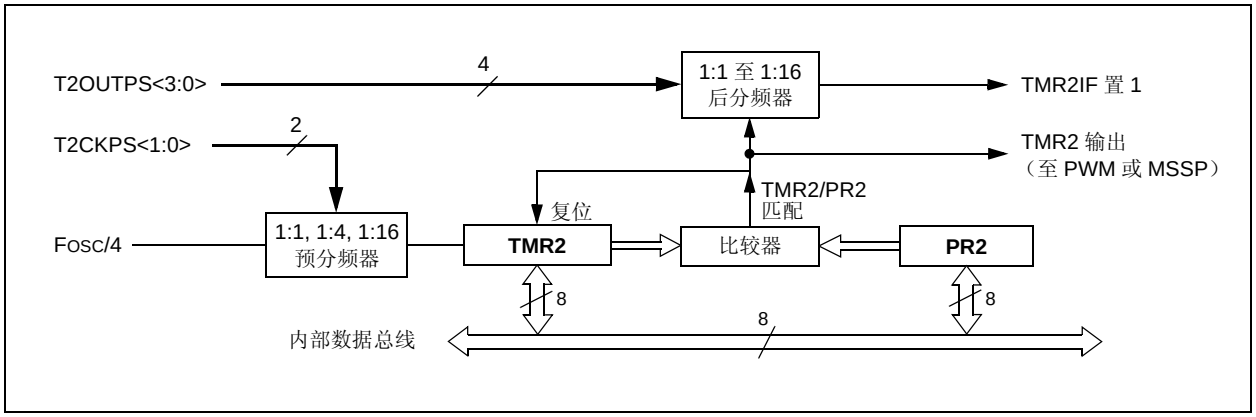


表 13-1： 与 TIMER2 作为定时器 / 计数器相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	62
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	62
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	62
TMR2	Timer2 寄存器								60
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	60
PR2	Timer2 周期寄存器								60

图注： — = 未实现，读为 0。 Timer2 模块不使用阴影单元。

14.0 TIMER3 模块

Timer3 模块定时器 / 计数器具有以下特性：

- 可由软件选择作为 16 位定时器或计数器
- 可读写的 8 位寄存器 （TMR3H 和 TMR3L）
- 可选择器件时钟或 Timer1 内部振荡器作为时钟源（内部或外部）
- 溢出时产生中断
- 由 CCP 特殊事件触发信号复位

图 14-1 给出了 Timer3 模块的简化框图。图 14-2 给出了此模块在读写模式下的工作原理框图。

Timer3 模块是通过 T3CON 寄存器 （寄存器 14-1）来控制的。它还可以为 CCP 模块选择时钟源。更多信息，请参见**第 16.2.2 节 “Timer1/Timer3 模式选择 ”**。

寄存器 14-1： T3CON：TIMER3 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
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
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 **RD16**：16 位读 / 写模式使能位
1 = 使能 Timer3 通过一次 16 位操作进行寄存器读 / 写
0 = 使能 Timer3 通过两次 8 位操作进行寄存器读 / 写

bit 6,3 **T3CCP<2:1>**：CCPx 的 Timer3 和 Timer1 使能位
1x = Timer3 是两个 CCP 模块的捕捉 / 比较时钟源
01 = Timer3 是 CCP2 的捕捉 / 比较时钟源；
Timer1 是 CCP1 的捕捉 / 比较时钟源
00 = Timer1 是两个 CCP 模块的捕捉 / 比较时钟源

bit 5-4 **T3CKPS<1:0>**：Timer3 输入时钟预分频比选择位
11 = 1:8 预分频比
10 = 1:4 预分频比
01 = 1:2 预分频比
00 = 1:1 预分频比

bit 2 **$\overline{\text{T3SYNC}}$**：Timer3 外部时钟输入同步控制位
（不适用于器件时钟来自 Timer1/Timer3 的场合。）
当 TMR3CS = 1 时：
1 = 不同步外部时钟输入
0 = 同步外部时钟输入
当 TMR3CS = 0 时：
该位被忽略。当 TMR3CS = 0 时， Timer3 使用内部时钟。

bit 1 **TMR3CS**：Timer3 时钟源选择位
1 = 使用 Timer1 振荡器或 T13CKI 引脚信号作为外部时钟输入（在第一个下降沿之后的上升沿开始计数）
0 = 内部时钟 （FOSC/4）

bit 0 **TMR3ON**：Timer3 使能位
1 = 使能 Timer3
0 = 停止 Timer3

14.1 Timer3 工作原理

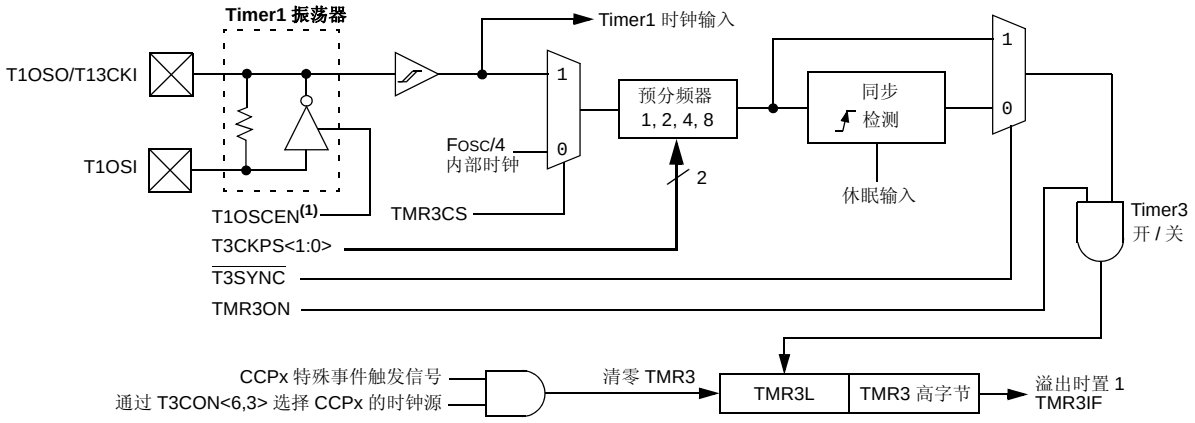
Timer3 可工作在以下三种模式之一：

- 定时器
- 同步计数器
- 异步计数器

工作模式由时钟选择位 TMR3CS（T3CON<1>）决定。当 TMR3CS 清零 （= 0）时， Timer3 在每个内部指令周期（FOSC/4）递增。当该位置 1 时，Timer3 在 Timer1 外部时钟输入信号或 Timer1 振荡器输出信号 （如果使能）的每个上升沿递增。

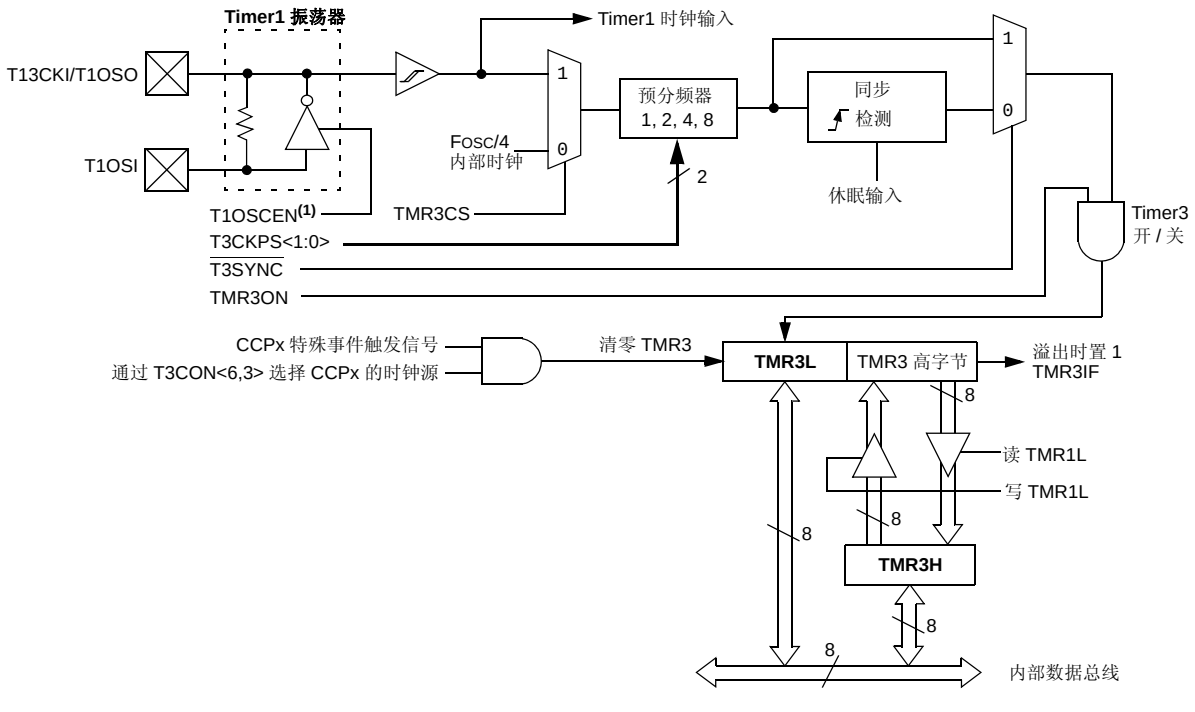
当使能 Timer1 时，RC1/T1OSI 和 RC0/T1OSO/T13CKI 引脚变为输入引脚。这意味着 TRISC<1:0> 的值将被忽略并且这些引脚将读为 0。

图 14-1： TIMER3 框图 （8 位模式）



注 1：当使能位 T1OSCEN 清零时，将关闭振荡器的反相器和反馈电阻以减少功耗。

图 14-2： TIMER3 框图 （16 位读 / 写模式）



注 1：当使能位 T1OSCEN 清零时，将关闭振荡器的反相器和反馈电阻以减少功耗。

14.2 Timer3 16 位读 / 写模式

可将 Timer3 配置为 16 位读写模式 （见图 14-2）。当 RD16 控制位 （T3CON<7>）置 1 时， TMR3H 的地址被映射到 Timer3 的高字节缓冲寄存器。读 TMR3L 将把 Timer3的高字节的内容装入Timer3高字节缓冲寄存器。这种方式使用户可以精确地读取 Timer3 的全部 16 位，而不需要像先读高字节再读低字节那样，由于两次读取之间可能存在计满返回，而不得不验证读取的有效性。

写 Timer3 的高字节也必须通过 TMR3H 缓冲寄存器进行。在写入 TMR3L 的同时，使用 TMR3H 的内容更新 Timer3 的高字节。这样允许用户将所有 16 位一次写入 Timer3 的高字节和低字节。

在该模式下不能直接读写 Timer3 的高字节。所有读写都必须通过 Timer3 高字节缓冲寄存器来进行。

写入 TMR3H 不会清零 Timer3 预分频器。只有在写 TMR3L 时才会清零该预分频器。

14.3 使用 Timer1 振荡器作为 Timer3 的时钟源

Timer1 内部振荡器可用作 Timer3 的时钟源。通过将 T1OSCEN 位 （T1CON<3>）置 1 可使能 Timer1 振荡器。要将它用作 Timer3 的时钟源，还必须将 TMR3CS 位置 1。如前文所述，这样做也会将 Timer3 配置为在振荡器源的每个上升沿递增。

在**第 12.0 节 “Timer1 模块 ”** 中对 Timer1 振荡器进行了说明。

14.4 Timer3 中断

TMR3 寄存器对 （TMR3H:TMR3L）从 0000h 递增到 FFFFh，然后计满返回到 0000h 重新开始计数。如果允许了 Timer3 中断，则溢出时会产生 Timer3 中断，锁存到中断标志位 TMR3IF （PIR2<1>）中。可以通过将 Timer3 中断允许位 TMR3IE（PIE2<1>）置 1 或清零来允许或禁止该中断。

14.5 使用 CCP 特殊事件触发信号复位 Timer3

如果 CCP1 或 CCP2 配置为在比较模式 （CCPxM <3:0> = 1011）下使用 Timer3 并产生特殊事件触发信号，该信号将复位 Timer3。如果使能了 A/D 模块，来自 CCP2 的触发信号还将启动 A/D 转换 （更多信息，请参见**第 16.3.4 节 “ 特殊事件触发器 ”**）。

要使用这一功能，必须将模块配置为定时器或同步计数器。在这种情况下，CCPRxH:CCPRxL 这对寄存器实际上变成了 Timer3 的周期寄存器。

如果 Timer3 在异步计数器模式下运行，复位操作可能不起作用。

如果 Timer3 的写操作和特殊事件触发同时发生，则写操作优先。

注：	CCPx 模块产生的特殊事件触发信号不会将 TMR3IF 中断标志位（PIR2<1>）置 1。

表 14-1： 与 TIMER3 作为定时器 / 计数器相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR2	OSCFIF	CMIF	—	—	BCLIF	LVDIF	TMR3IF	—	62
PIE2	OSCFIE	CMIE	—	—	BCLIE	LVDIE	TMR3IE	—	62
IPR2	OSCFIP	CMIP	—	—	BCLIP	LVDIP	TMR3IP	—	62
TMR3L	Timer3 寄存器的低字节								61
TMR3H	Timer3 寄存器的高字节								61
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	60
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	61

图注： — = 未实现，读为 0。 Timer3 模块不使用阴影单元。

注：

15.0 实时时钟和日历 （RTCC）

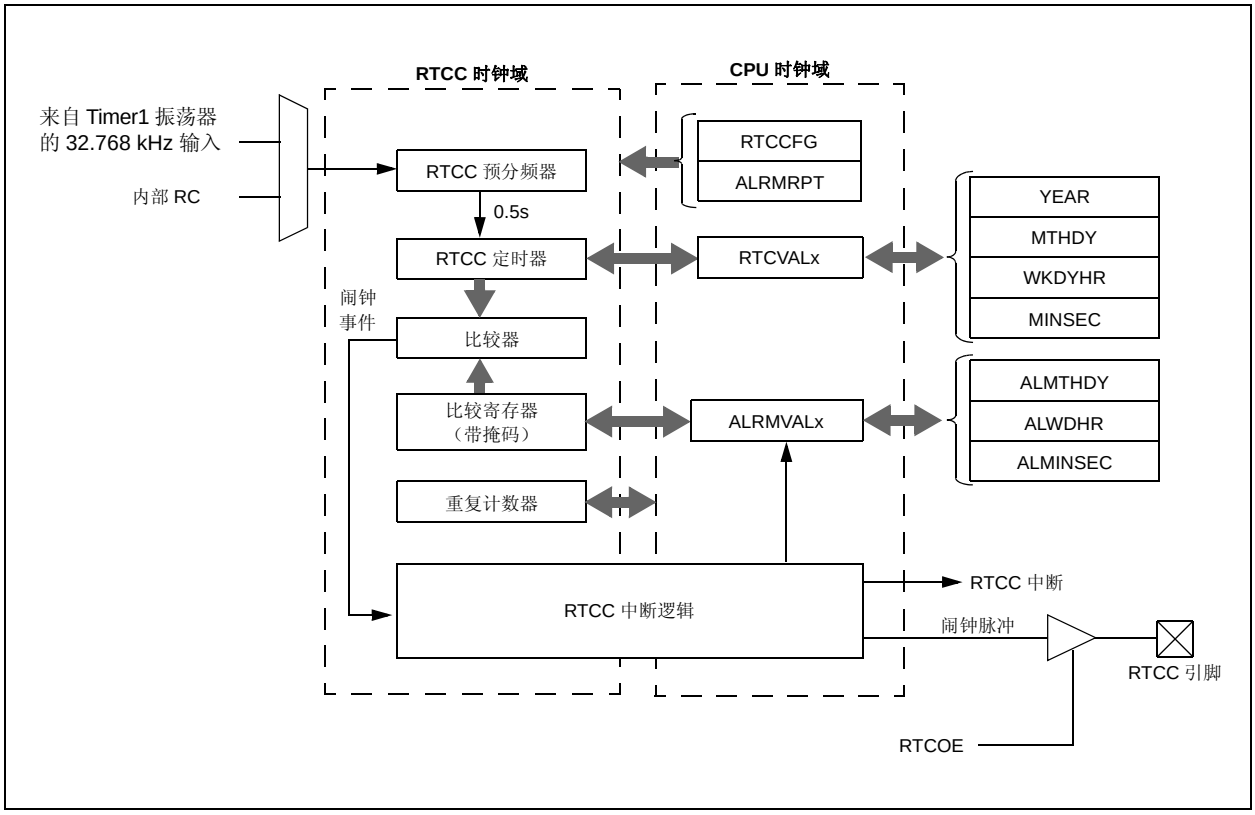
实时时钟和日历 （RTCC）模块的主要特性有：

- 时间：小时、分钟和秒钟
- 24 小时格式 （军用时间）
- 日历：星期、日期、月和年
- 闹钟可配置
- 年份范围：2000 至 2099
- 闰年修正
- 用于获得紧凑固件的 BCD 格式
- 为低功耗工作进行了优化
- 带自动调节的用户校准
- 校准范围：每月 ±2.64 秒误差
- 要求：外部 32.768 kHz 时钟晶振
- RTCC 引脚上的闹钟脉冲或秒时钟输出

RTCC 模块是为需要长时间维持精确时间的应用设计的，无需或很少需要 CPU 干预。该模块为低功耗使用进行了优化，以便在跟踪时间时延长电池寿命。

该模块是百年时钟和日历，能自动检测闰年。时钟范围为从 2000 年 1 月 1 日 00:00:00（午夜）到 2099 年 12 月 31 日 23:59:59。小时数采用 24 小时（军用时间）格式计量。该时钟提供一秒的时间粒度，用户可看到半秒的时间间隔。

图 15-1： RTCC 框图



15.1 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

- RTCCFG
- RTCCAL
- PADCFG1
- ALRMCFG
- ALRMRPT

RTCC 值寄存器

- RTCVALH 和 RTCVALL—— 可访问以下寄存器：
  - YEAR
  - MONTH
  - DAY
  - WEEKDAY
  - HOUR
  - MINUTE
  - SECOND

闹钟值寄存器

- ALRMVALH 和 ALRMVALL—— 可访问以下寄存器：
  - ALRMMNTH
  - ALRMDAY
  - ALRMWD
  - ALRMHR
  - ALRMMIN
  - ALRMSEC

注： RTCVALH 和 RTCVALL 寄存器可通过 RTCRPT<1:0> 进行访问。ALRMVALH 和 ALRMVALL 可通过 ALRMPTR<1:0> 进行访问。

15.1.1 RTCC 控制寄存器

寄存器 15-1： RTCCFG：RTCC 配置寄存器 (1)

R/W-0	U-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0
RTCEN(2)	—	RTCWREN	RTCSYNC	HALFSEC(3)	RTCOE	RTCPTR1	RTCPTR0
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 **RTCEN：**RTCC 使能位 (2)
1 = 使能 RTCC 模块
0 = 禁止 RTCC 模块

bit 6 **未实现：**读为 0

bit 5 **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 4 **RTCSYNC：**RTCC 值寄存器读同步位
1 = 由于计满返回，RTCVALH、RTCVALL 和 ALRMRPT 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 和 ALCFGRPT 寄存器在读取时无需考虑计满返回

bit 3 **HALFSEC：**半秒状态位 (3)
1 = 一秒的后半秒
0 = 一秒的前半秒

bit 2 **RTCOE：**RTCC 输出使能位
1 = 使能 RTCC 时钟输出
0 = 禁止 RTCC 时钟输出

bit 1-0 **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<7:0> 时 RTCPTR<1:0> 的值就递减 1，直到达到 00。
RTCVALH：
00 = 分钟数
01 = 星期
10 = 月
11 = 保留
RTCVALL：
00 = 秒数
01 = 小时数
10 = 日
11 = 年

注 1： RTCCFG 寄存器只受 POR 的影响。对于除 POR 外的所有其他复位， RTCC 将在器件处于复位状态时继续运行。
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 15-2：　　RTCCAL：RTCC 校准寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0
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 7-0　　**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
.
.
.
00000001 = 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

寄存器 15-3：　　PADCFG1：焊盘配置寄存器

U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0	U-0
—	—	—	—	—	RTSECSEL1(1)	RTSECSEL0(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 7-3　　**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 （引脚可以是 INTOSC 或 Timer1 振荡器，取决于 RTCOSC （CONFIG3L<1>）位的设置）(2)
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　　**未实现：**读为 0

注　1： 要使能实际 RTCC 输出， RTCOE 位 （RTCCFG<2>）必须置 1。
2： 如果 Timer1 振荡器作为 RTCC 的时钟源，应将 T1OSCEN 位置 1 （T1CON<3> = 1）。

寄存器 15-4： ALRMCFG：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 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 **ALRMEN：**闹钟使能位
1 = 使能闹钟 （每当 ARPT<7:0> = 00 且 CHIME = 0 时，发生闹钟事件后都自动清零）
0 = 禁止闹钟

bit 6 **CHIME：**响铃 （Chime）使能位
1 = 使能响铃； ALRMPTR<1:0> 位允许从 00h 返回到 FFh
0 = 禁止响铃； ALRMPTR<1:0> 到达 00h 就停止

bit 5-2 **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 日时除外，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 1-0 **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。

ALRMVALH：
00 = ALRMMIN
01 = ALRMWD
10 = ALRMMNTH
11 = 未实现

ALRMVALL：
00 = ALRMSEC
01 = ALRMHR
10 = ALRMDAY
11 = 未实现

寄存器 15-5：　　ALRMRPT：闹钟重复寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
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 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。

15.1.2　RTCVALH 和 RTCVALL 寄存器映射

寄存器 15-6：　　保留的寄存器

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 7-0　　**未实现：**读为 0

寄存器 15-7：　　YEAR：年份值寄存器 (1)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
YRTEN3	YRTEN2	YRTEN1	YRTEN0	YRONE3	YRONE2	YRONE1	YRONE0
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 7-4　　**YRTEN<3:0>：**年份的十位数的二 - 十进制码 （Binary Coded Decimal， BCD）值位
值为 0 到 9。

bit 3-0　　**YRONE<3:0>：**年份的个位数的 BCD 值位
值为 0 到 9。

注　1： 仅当 RTCWREN = 1 时才允许写入 YEAR 寄存器。

寄存器 15-8：　　MONTH：月值寄存器 (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 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　　**未实现：**读为 0

bit 4　　**MTHTEN0：**月份的十位数的 BCD 值位
值为 0 或 1。

bit 3-0　　**MTHONE<3:0>：**月份的个位数的 BCD 值位
值为 0 到 9。

注　1： 仅当 RTCWREN = 1 时才允许写入该寄存器。

寄存器 15-9：　　DAY：日值寄存器 (1)

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 7-6　　**未实现：**读为 0

bit 5-4　　**DAYTEN<1:0>：**日的十位数的 BCD 值位
值为 0 到 3。

bit 3-0　　**DAYONE<3:0>：**日的个位数的 BCD 值位
值为 0 到 9。

注　1： 仅当 RTCWREN = 1 时允许写入该寄存器。

寄存器 15-10：　　WEEKDAY：星期值寄存器 (1)

U-0	U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x
—	—	—	—	—	WDAY2	WDAY1	WDAY0
bit 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 7-3　　**未实现：**读为 0

bit 2-0　　**WDAY<2:0>：**星期的 BCD 值位
值为 0 到 6。

注　1： 仅当 RTCWREN = 1 时才允许写入该寄存器。

寄存器 15-11： HOUR：小时值寄存器 (1)

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 7-6 **未实现：**读为 0

bit 5-4 **HRTEN<1:0>：**小时的十位数的 BCD 值位
值为 0 到 2。

bit 3-0 **HRONE<3:0>：**小时的个位数的 BCD 值位
值为 0 到 9。

注 1： 仅当 RTCWREN = 1 时才允许写入该寄存器。

寄存器 15-12： MINUTE：分钟值寄存器

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 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 **未实现：**读为 0

bit 6-4 **MINTEN<2:0>：**分钟的十位数的 BCD 值位
值为 0 到 5。

bit 3-0 **MINONE<3:0>：**分钟的个位数的 BCD 值位
值为 0 到 9。

寄存器 15-13： SECOND：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
—	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 7 **未实现：**读为 0

bit 6-4 **SECTEN<2:0>：** 秒的十位数的 BCD 值位
值为 0 到 5。

bit 3-0 **SECONE<3:0>：** 秒的个位数的 BCD 值位
值为 0 到 9。

15.1.3 ALRMVALH 和 ALRMVALL 寄存器映射

寄存器 15-14： ALRMMNTH：闹钟月值寄存器 (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 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 **未实现：**读为 0

bit 4 **MTHTEN0：**月份的十位数的 BCD 值位
值为 0 或 1。

bit 3-0 **MTHONE<3:0>：**月份的个位数的 BCD 值位
值为 0 到 9。

注 1： 仅当 RTCWREN = 1 时才允许写入该寄存器。

寄存器 15-15： ALRMDAY：闹钟日值寄存器 (1)

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 7-6 **未实现：**读为 0

bit 5-4 **DAYTEN<1:0>：**日的十位数的 BCD 值位
值为 0 到 3。

bit 3-0 **DAYONE<3:0>：**日的个位数的 BCD 值位
值为 0 到 9。

注 1： 仅当 RTCWREN = 1 时才允许写入该寄存器。

寄存器 15-16： ALRMWD：闹钟星期值寄存器 (1)

U-0	U-0	U-0	U-0	U-0	R/W-x	R/W-x	R/W-x
—	—	—	—	—	WDAY2	WDAY1	WDAY0
bit 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 7-3 **未实现：**读为 0

bit 2-0 **WDAY<2:0>：**星期的 BCD 值位
值为 0 到 6。

注 1： 仅当 RTCWREN = 1 时才允许写入该寄存器。

寄存器 15-17： ALRMHR：闹钟小时值寄存器 (1)

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 7-6 **未实现：**读为 0

bit 5-4 **HRTEN<1:0>：**小时的十位数的 BCD 值位
值为 0 到 2。

bit 3-0 **HRONE<3:0>：**小时的个位数的 BCD 值位
值为 0 到 9。

注 1： 仅当 RTCWREN = 1 时才允许写入该寄存器。

寄存器 15-18： ALRMMIN：闹钟分钟值寄存器

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 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 **未实现：**读为 0

bit 6-4 **MINTEN<2:0>：**分钟的十位数的 BCD 值位
值为 0 到 5。

bit 3-0 **MINONE<3:0>：**分钟的个位数的 BCD 值位
值为 0 到 9。

寄存器 15-19： ALRMSEC：闹钟秒值寄存器

U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	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 7 **未实现：**读为 0

bit 6-4 **SECTEN<2:0>：**秒的十位数的 BCD 值位
值为 0 到 5。

bit 3-0 **SECONE<3:0>：**秒的个位数的 BCD 值位
值为 0 到 9。

15.1.4 RTCEN 位写

RTCWREN = 0 时尝试写入 RTCEN 位将被忽略。RTCWREN必须置1，然后才会发生对RTCEN的写操作。

像 RTCEN 位一样， RTCVALH 和 RTCVALL 寄存器只能在 RTCWREN = 1 时写入。RTCWREN = 0 时写入这些寄存器将被忽略。

15.2 工作原理

15.2.1 寄存器接口

RTCC和闹钟值的寄存器接口是用二-十进制码（BCD）格式实现的。这就在使用该模块时简化了固件，因为每个数字都包含在它自己的 4 位值中了（见图 15-2 和图 15-3）。

图 15-2： 定时器位格式

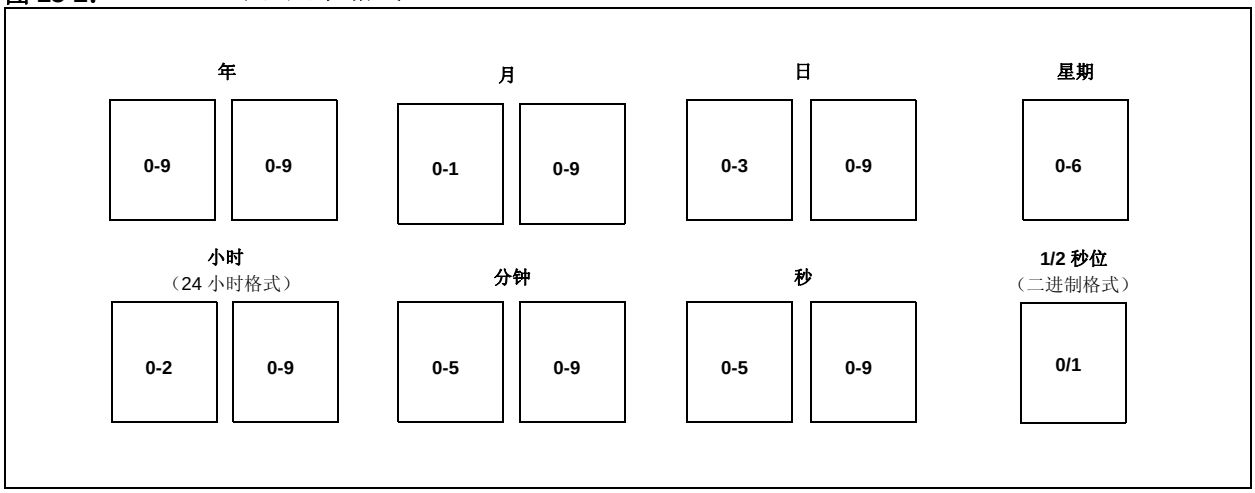
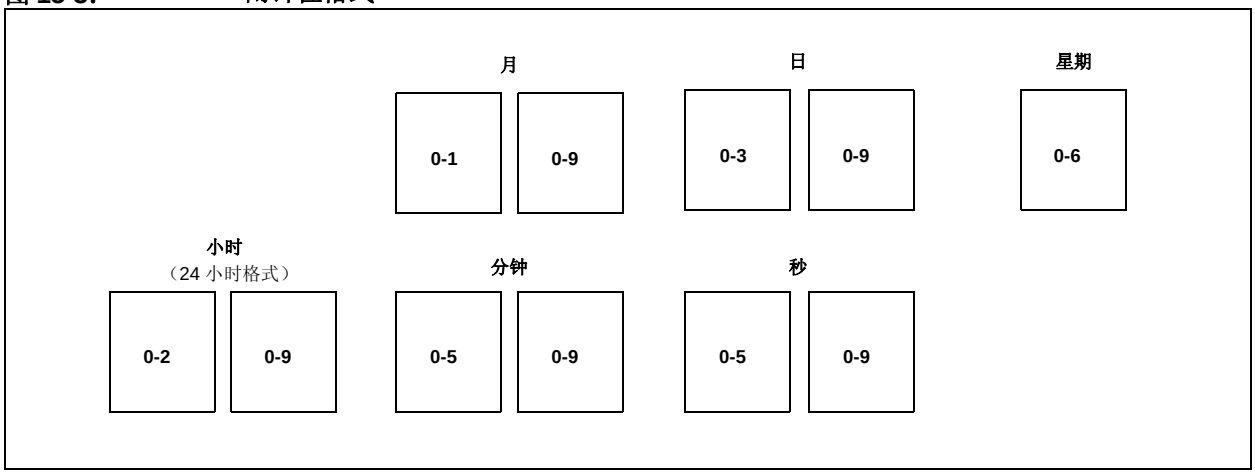


图 15-3： 闹钟位格式

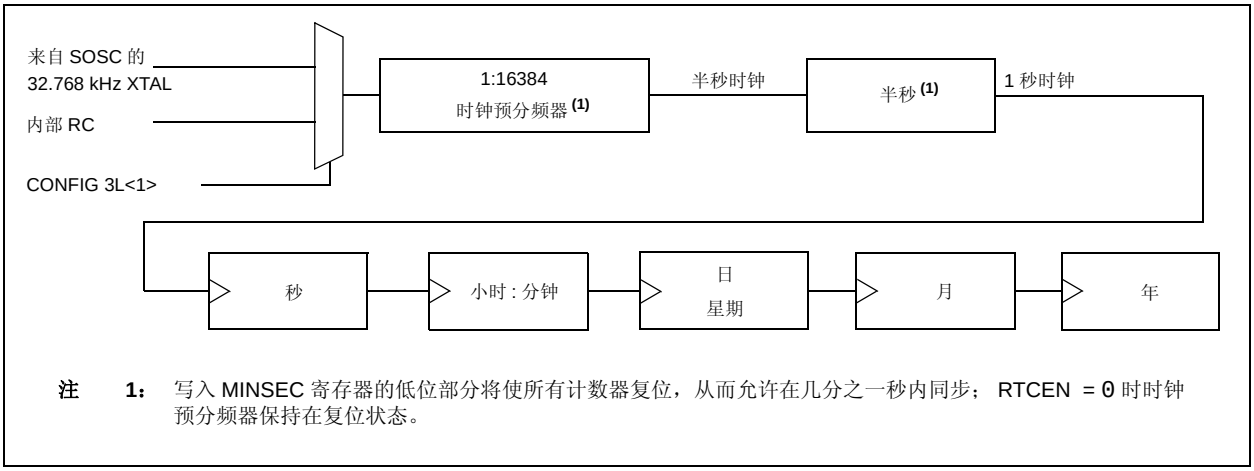


15.2.2 时钟源

正如之前提到的，RTCC 模块应由 32.768 kHz 的外部实时时钟晶振提供时钟源，但也可以由内部振荡器提供时钟源。RTCC 时钟选择由 RTCOSC 位（CONFIG3L<1>）决定。

晶振校准可通过该模块实现，产生的误差为每月 3 秒或更小。（更多详细信息，请参见**第 15.2.9 节 “ 校准 ”**。）

图 15-4： 时钟源复用



注 1： 写入 MINSEC 寄存器的低位部分将使所有计数器复位，从而允许在几分之一秒内同步； RTCEN = 0 时时钟预分频器保持在复位状态。

15.2.2.1 实时时钟使能

RTCC 模块可由外部 32.768 kHz 晶振（Timer1 振荡器）或内部 RC 振荡器提供时钟源，这通过 CONFIG3L<1> 选择。

如果使用了外部时钟，应通过将 T1OSCEN 位置 1（T1CON<3> = 1）使能 Timer1 振荡器。如果由 INTRC 提供时钟，可通过 PADCFG 寄存器中的 RTSECSEL<1:0> 位将 INTRC 时钟输出到 RTCC 引脚。

15.2.3 进位规则

本节说明发生计满返回时，会影响哪些定时器值。

- 日时间：从 23:59:59 到 00:00:00，向日字段进位
- 月：从 12/31 到 01/01，向年字段进位
- 星期：从 6 到 0，无进位 （见表 15-1）
- 年进位：从 99 到 00 ；这也超越 RTCC 的使用

关于日到月的计满返回设定，请参见表 15-2。

考虑到以下值是 BCD 格式，进位到 BCD 的高位将在计数为 10 时发生，而不是在计数为 16 时发生（SECONDS、MINUTES、HOURS、WEEKDAY、DAYS 和 MONTHS）。

表 15-1： 星期设定

星期	
星期日	0
星期一	1
星期二	2
星期三	3
星期四	4
星期五	5
星期六	6

表 15-2： 日到月计满返回设定

月	最大日字段
01 （一月）	31
02 （二月）	28 或 29 (1)
03 （三月）	31
04 （四月）	30
05 （五月）	31
06 （六月）	30
07 （七月）	31
08 （八月）	31
09 （九月）	30
10 （十月）	31
11 （十一月）	30
12 （十二月）	31

注 1： 请参见**第 15.2.4 节 “ 闰年 ”**。

15.2.4 闰年

由于 RTCC 模块的年份范围是 2000 到 2099，闰年是通过以上范围内的年份能否被 4 整除来确定的。闰年中唯一受影响的月份是二月。

二月在闰年中有 29 天，而其他年份中二月是 28 天。

15.2.5 一般功能

所有包含秒或更大时间值的定时器寄存器都可写。用户只需将要写入的年、月、日、小时、分钟和秒通过寄存器指针写入定时器寄存器，就可以配置时间（见**第 15.2.8 节“寄存器映射”**）。

随后定时器就会用新写入的值从所需起点继续计数。

通过将 RTCEN 位（RTCCFG<7>）置 1 使能 RTCC。如果定时器在调节这些寄存器时是使能的，则定时器仍将继续递增。但是，任何时候只要写入MINSEC寄存器，两个定时器预分频器都将复位为 0。这就允许在几分之一秒内同步。

定时器寄存器的更新周期与 CPU 执行写指令的周期相同。用户必须确保 RTCEN = 1 时，更新的寄存器不会同时递增。这可以通过几种方式实现：

- 检查 RTCSYNC 位（RTCCFG<4>）
- 检查前面可能发生进位的位
- 在秒脉冲（或闹钟中断）后立即更新寄存器

用户可看到计数器的半秒字段。该值是只读的，只能通过写入 SECONDS 寄存器的低半部分复位。

15.2.6 寄存器读写安全窗口

RTCSYNC 位指示可安全读写 RTCC 时钟域寄存器、而无需担心计满返回的时间窗。当 RTCSYNC = 0 时，CPU 可安全访问这些寄存器。

无论 RTCSYNC = 1 还是 0，用户都应采用固件方法确保数据读取没有发生在计满返回边界，从而导致无效或部分读取。该固件方法由读取每个寄存器两遍，然后比较两个值组成。如果两个值匹配，则没有发生计满返回。

15.2.7 写锁定

为执行对任何 RTCC 定时器寄存器的写入，RTCWREN 位（RTCCFG<5>）必须置 1。

为避免意外写入 RTCC 定时器寄存器，建议在不写入该寄存器的其他任何时候 RTCWREN 位（RTCCFG<5>）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，建议用户遵循例 15-1 中的代码示例。

例 15-1：　将 RTCWREN 位置 1

```
movlw   0x55
movwf   EECON2
movlw   0xAA
movwf   EECON2
bsf     RTCCFG,RTCWREN
```

15.2.8 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（RTCCFG <1:0>）选择所需的定时器寄存器对。

通过读或写 RTCVALH 寄存器，RTCC 指针值（RTCPTR <1:0>）递减 1，直到达到 00。一旦达到 00，MINUTES 和 SECONDS 值可通过 RTCVALH 和 RTCVALL 访问，直到手动更改指针值。

表 15-3：　RTCVALH 和 RTCVALL 寄存器映射

RTCPTR<1:0>	RTCC 值寄存器窗口	
	RTCVALH	RTCVALL
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（ALRMCFG<1:0>）选择所需的闹钟寄存器对。

通过读或写 ALRMVALH 寄存器，闹钟指针值 ALRMPTR <1:0> 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 15-4： ALRMVAL 寄存器映射

ALRMPTR<1:0>	闹钟值寄存器窗口	
	ALRMVALH	ALRMVALL
00	ALRMMIN	ALRMSEC
01	ALRMWD	ALRMHR
10	ALRMMNTH	ALRMDAY
11	—	—

15.2.9 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 RTCCAL 寄存器的低半部分。装入 RTCCAL 的 8 位有符号值乘以 4，每分钟一次将其加到 RTCC 定时器中或从 RTCC 定时器中减去。

校准 RTCC 模块：

1. 使用器件上的其他定时器资源找出32.768 kHz晶振的误差。
2. 转换每分钟误差时钟脉冲数 （见公式 15-1）。

公式 15-1： 转换误差时钟脉冲

(理想频率 （32,758） – 测得频率) * 60 = 每分钟误差时钟数

- 如果振荡器*快*于理想频率 （从步骤 2 得出的负的结果）， RCFGCALL 寄存器值必须为负。这会导致每分钟从定时器计数器减去指定的时钟脉冲数。
- 如果振荡器*慢*于理想频率（从步骤 2 得出的正的结果），RCFGCALL 寄存器值必须为正。这会导致每分钟在定时器计数器上加上指定的时钟脉冲数。

3. 将正确的值装入 RTCCAL 寄存器。

只有当定时器关闭或紧接秒脉冲的上升沿时，才会发生对 RTCCAL 寄存器的写操作。

注： 在确定晶振误差值时，是否包含由温度或晶振老化所造成的漂移引起的初始误差，由用户自行决定。

15.3 闹钟

闹钟功能和特性如下所示：

- 可在半秒到一年的范围内配置
- 使用 ALRMEN 位（寄存器 15-4 的 ALRMCFG<7>）使能
- 提供一次性闹钟和重复闹钟选项

15.3.1 配置闹钟

闹钟功能用 ALRMEN 位使能。

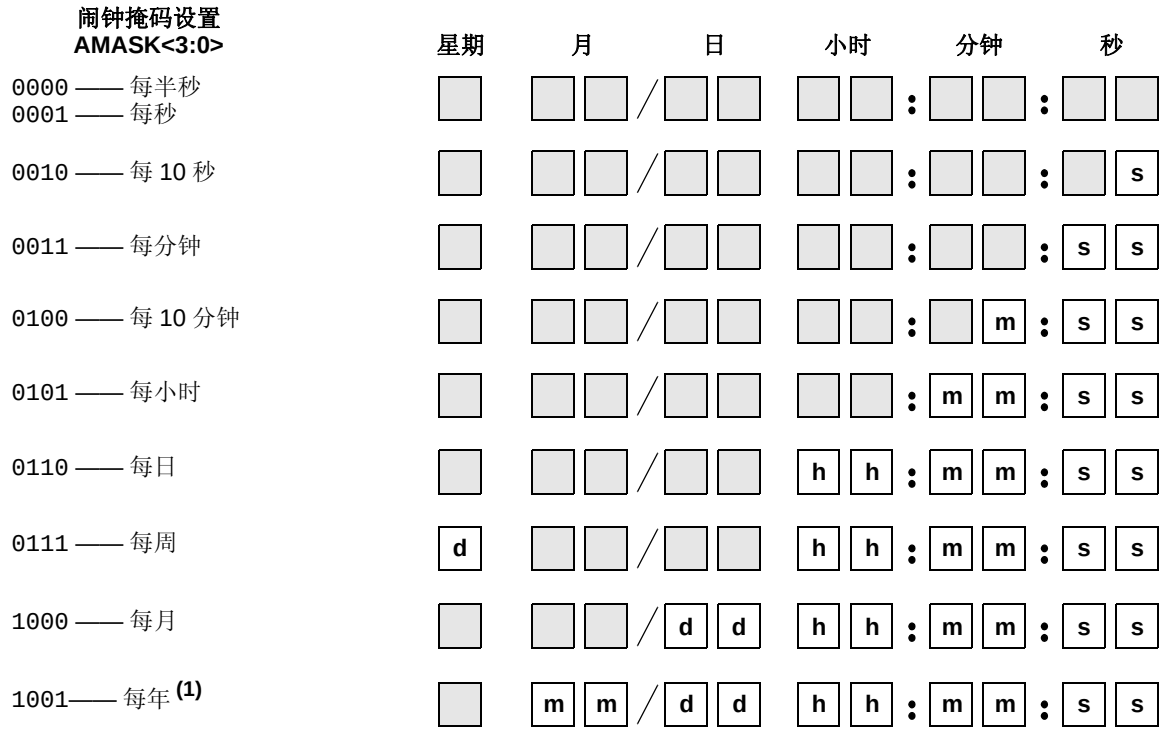
发出闹钟后该位清零。如果 CHIME 位 = 1 或者 ALRMRPT ≠ 0，则不会清零该位。

闹钟的间隔时间选择通过 ALRMCFG （AMASK<3:0>）位配置 （见图 15-5）。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。使能闹钟后发生的总次数存储在 ALRMRPT 寄存器中。

注： 使能闹钟 （ALRMEN = 1）时，更改除 RTCCAL、ALRMCFG 和 ALRMRPT 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。为避免这种情况，只应在禁止闹钟（ALRMEN = 0）时更改定时器值和闹钟值。建议在 RTCSYNC = 0 时更改 ALRMCFG 和 ALRMRPT 寄存器以及 CHIME 位。

图 15-5：　　闹钟掩码设置



注　1： 每年，除非配置为 2 月 29 日。

当 ALRMCFG = 00 且 CHIME 位 = 0（ALRMCFG<6>）时，重复功能被禁止，只发生单次闹钟。通过将 FFh 装入 ALRMRPT 寄存器，闹钟可重复最多 255 次。

每个闹钟发出后， ALRMRPT 寄存器都递减 1。一旦寄存器达到 00，将最后一次发出闹钟。

最后一次发出闹钟后 ALRMEN 位自动清零，闹钟关闭。如果 CHIME 位 = 1，闹钟可能不断重复。

当 CHIME = 1 时，ALRMRPT 寄存器达到 00 时不会禁止闹钟，而是计满返回到 FF，继续无限计数。

15.3.2　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。

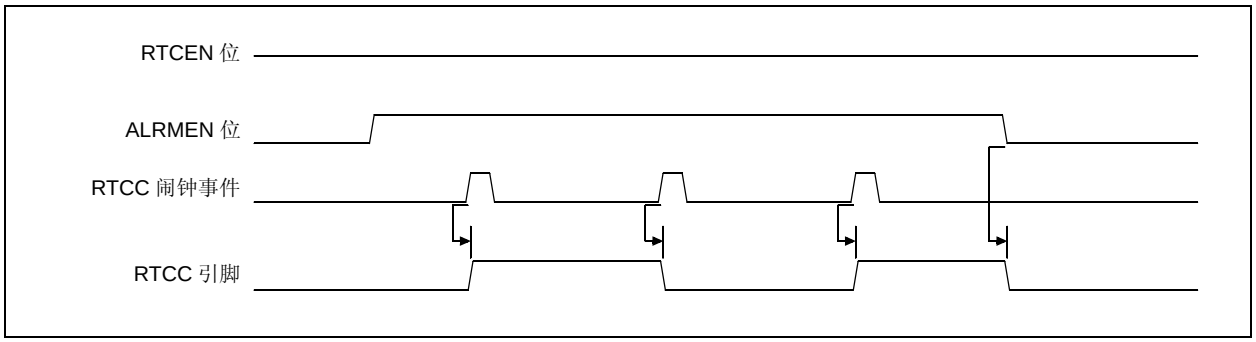
闹钟脉冲输出完全与 RTCC 时钟同步，可用作其他外设的触发时钟。可在 RTCC 引脚上输出闹钟脉冲。输出脉冲是占空比为 50% 的时钟，频率为闹钟事件频率的一半 （见图 15-6）。

RTCC 引脚也能输出秒时钟。用户可在 RTCC 模块生成的闹钟脉冲或秒时钟输出之间选择。

RTSECSEL<1:0> 位 （PADCFG1<2:1>）在以下两个输出间选择：

- 闹钟脉冲 ——RTSECSEL<1:0> = 00
- 秒时钟 ——RTSECSEL<1:0> = 01

图 15-6：　　定时器脉冲的产生



15.4　休眠模式

定时器和闹钟在休眠模式下也继续工作。闹钟的工作不受休眠影响，因为闹钟事件总能唤醒 CPU。

空闲模式不影响定时器或闹钟的工作。

15.5　复位

15.5.1　　器件复位

发生器件复位时， ALCFGRPT 寄存器强制进入其复位状态，导致闹钟被禁止（如果复位前闹钟是使能的）。如果 RTCC 是使能的，发生基本器件复位后将继续工作。

15.5.2　　上电复位（POR）

RTCCFG 和 ALRMRPT 寄存器只在 POR 时复位。器件退出 POR 状态后，时钟寄存器应重新装入所需值。

定时器预分频值只能通过写入 SECONDS 寄存器复位。器件复位不会影响预分频器。

15.6 寄存器映射

表 15-5、表 15-6 和表 15-7 总结了与 RTCC 模块相关的寄存器。

表 15-5： RTCC 控制寄存器

寄存器名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态所在页
RTCCFG	RTCEN	—	RTCWREN	RTCSYNC	HALFSEC	RTCOE	RTCPTR1	RTCPTR0	64
RTCCAL	CAL7	CAL6	CAL5	CAL4	CAL3	CAL2	CAL1	CAL0	64
PADCFG1	—	—	—	—	—	RTSECSEL1	RTSECSEL0	—	64
ALRMCFG	ALRMEN	CHIME	AMASK3	AMASK2	AMASK1	AMASK0	ALRMPTR1	ALRMPTR0	64
ALRMRPT	ARPT7	ARPT6	ARPT5	ARPT4	ARPT3	ARPT2	ARPT1	ARPT0	64

图注： — = 未实现，读为 0。 80 引脚器件的复位值以十六进制显示。

表 15-6： RTCC 值寄存器

寄存器名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态所在页
RTCVALH	RTCC 值高字节寄存器窗口，基于 RTCPTR<1:0>								64
RTCVALL	RTCC 值低字节寄存器窗口，基于 RTCPTR<1:0>								64

图注： — = 未实现，读为 0。 80 引脚器件的复位值以十六进制显示。

表 15-7： 闹钟值寄存器

寄存器名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	所有复位时的状态所在页
ALRMVALH	闹钟值高字节寄存器窗口，基于 ALRMPTR<1:0>								64
ALRMVALL	闹钟值低字节寄存器窗口，基于 ALRMPTR<1:0>								64

图注： — = 未实现，读为 0。 80 引脚器件的复位值以十六进制显示。

注：

16.0 捕捉 / 比较 /PWM （CCP）模块

PIC18F87J90 系列器件具有两个捕捉 / 比较 /PWM （Capture / Compare / PWM， CCP）模块，分别命名为 CCP1 和 CCP2。两个模块均可实现标准的捕捉、比较和脉宽调制 （Pulse-Width Modulation， PWM）模式。

每个 CCP 模块包含两个 8 位寄存器，可用作两个 8 位捕捉寄存器、两个 8 位比较寄存器或两个 PWM 主 / 从占空比寄存器。为避免混淆，以下所有的 CCP 模块操作描述均针对 CCP2，但同样适用于 CCP1。

寄存器 16-1： CCPxCON：CCPx 控制寄存器 （CCP1 和 CCP2 模块）

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
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 7-6 **未实现：**读为 0

bit 5-4 **DCxB<1:0>：**CCPx 模块的 PWM 占空比 bit 1 和 bit 0

捕捉模式：
未使用。

比较模式：
未使用。

PWM 模式：
这两位是 10 位 PWM 占空比的低 2 位（bit 1 和 bit 0）。占空比的高 8 位（DCx<9:2>）在 CCPRxL 中。

bit 3-0 **CCPxM<3:0>：**CCPx 模块模式选择位

0000 = 禁止捕捉 / 比较 /PWM （复位 CCPx 模块）
0001 = 保留
0010 = 比较模式，匹配时输出电平翻转 （CCPxIF 位置 1）
0011 = 保留
0100 = 捕捉模式，每个下降沿
0101 = 捕捉模式，每个上升沿
0110 = 捕捉模式，每 4 个上升沿
0111 = 捕捉模式，每 16 个上升沿
1000 = 比较模式：初始化 CCPx 引脚为低电平，比较匹配时强制 CCPx 引脚为高电平 （CCPxIF 位置 1）
1001 = 比较模式：初始化 CCPx 引脚为高电平，比较匹配时强制 CCPx 引脚为低电平 （CCPxIF 位置 1）
1010 = 比较模式：比较匹配时产生软件中断 （CCPxIF 位置 1， CCPx 引脚反映 I/O 状态）
1011 = 比较模式：当 CCPx 发生匹配时产生特殊事件触发信号、复位定时器或启动 A/D 转换 （CCPxIF 位置 1）**(1)**
11xx = PWM 模式

注 1： 如果 CCPxM<3:0> = 1011，在 CCP1 发生匹配时将只复位定时器而不启动 A/D 转换。

16.1 CCP 模块配置

每个捕捉 / 比较 /PWM 模块均与一个控制寄存器（通常为 CCPxCON）和一个数据寄存器（CCPRx）相关联。数据寄存器由两个 8 位寄存器组成：CCPRxL（低字节）和CCPRxH（高字节）。所有寄存器都是可读写的。

16.1.1 CCP 模块和定时器资源

CCP 模块根据选定的模式使用 Timer1、Timer2 或 Timer3。模块在捕捉或比较模式下使用 Timer1 和 Timer3，而在 PWM 模式下使用 Timer2。

表 16-1： CCP 模式 —— 定时器资源

CCP 模式	定时器资源
捕捉	Timer1 或 Timer3
比较	Timer1 或 Timer3
PWM	Timer2

要将哪个特定的定时器分配给一个 CCP 模块由 T3CON 寄存器（寄存器 14-1）中的 “CCP 的定时器 ” 使能位决定。如果将两个 CCP 模块配置为同时工作在相同的模式（捕捉 / 比较或 PWM）下，那么这两个模块可同时被激活并可共享相同的定时器资源。表 16-2 总结了这两个模块间的相互关系。

根据选定的配置，最多可以有 4 个定时器同时有效，具有相同配置（捕捉 / 比较或 PWM）的模块共用定时器资源。图 16-1 给出了可能的配置情况。

16.1.2 漏极开路输出选项

当在输出模式下工作时（即处于比较或 PWM 模式下），可以选择将 CCPx 引脚的驱动器配置为漏极开路输出。此功能使引脚上的电压可通过外部上拉电阻上拉至较高的电压，并且无需额外的电平转换器就可使输出与外部电路进行通信。

漏极开路输出选项由 CCP2OD 和 CCP1OD 位（TRISG <6:5>）控制。通过将相应的位置 1 可将对应模块的引脚配置为漏极开路操作。

16.1.3 CCP2 引脚分配

可根据器件配置改变 CCP2（捕捉输入、比较和 PWM 输出）的引脚分配。CCP2MX 配置位决定哪个引脚将与 CCP2 复用。默认情况下，CCP2 引脚被分配给 RC1（CCP2MX = 1）。如果清零该配置位，CCP2 将与 RE7 复用。

改变 CCP2 的引脚分配并不会自动改变对端口引脚配置的任何要求。无论其引脚的分配如何，用户必须始终确保与 CCP2 操作相对应的 TRIS 寄存器配置正确。

图 16-1： CCP 和定时器互连配置

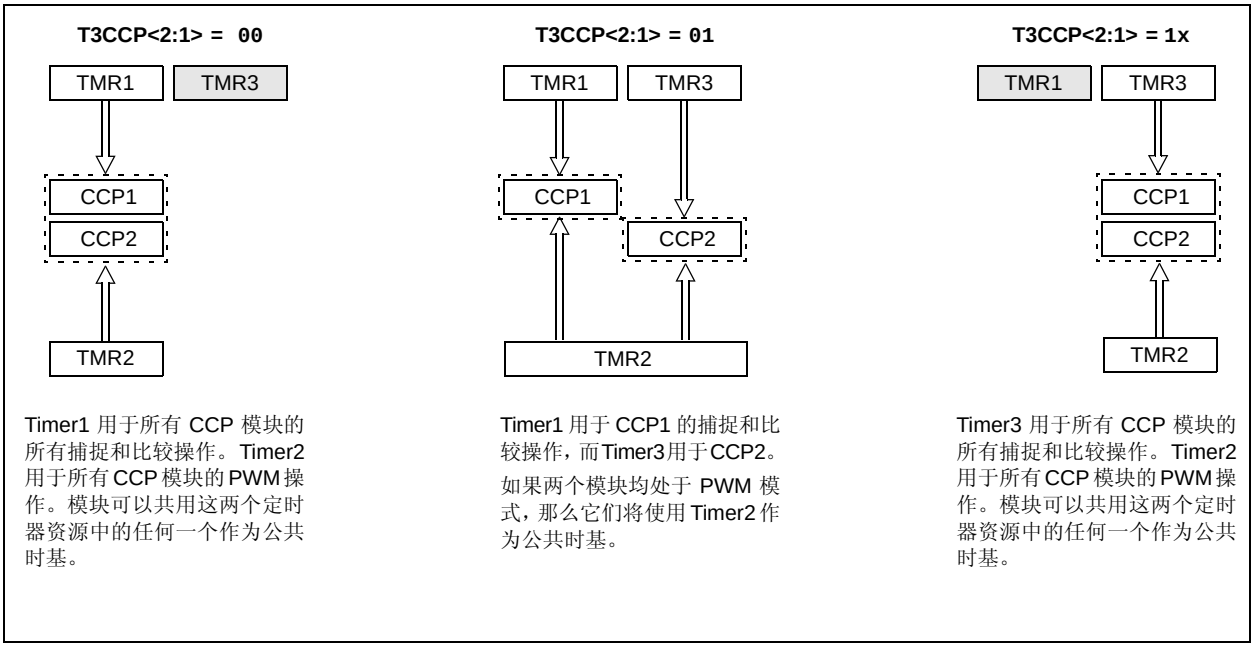


表 16-2： CCP1 和 CCP2 在使用定时器资源方面的相互关系

CCP1 模式	CCP2 模式	相互关系
捕捉	捕捉	每个模块都可用 TMR1 或 TMR3 作为时基。每个 CCP 的时基也可以不同。
捕捉	比较	可将 CCP2 配置为产生特殊事件触发信号用以复位 TMR1 或 TMR3 （取决于所使用的时基）。也可用于在发生触发事件时自动触发 A/D 转换。如果 CCP1 使用与 CCP2 相同的定时器作为时基，上述操作可能会对 CCP1 产生影响。
比较	捕捉	可将 CCP1 配置为产生特殊事件触发信号用以复位 TMR1 或 TMR3 （取决于所使用的时基）。如果 CCP2 使用与 CCP1 相同的定时器作为时基，上述操作可能会对 CCP2 产生影响。
比较	比较	每个模块均可配置为产生特殊事件触发信号用以复位时基。 CCP2 触发事件还可自动触发 A/D 转换。如果两个模块使用相同的时基，可能会发生冲突。
捕捉	PWM	无
比较	PWM	无
PWM	捕捉	无
PWM	比较	无
PWM	PWM	两个 PWM 具有相同的频率和更新速率 （TMR2 中断）。

16.2 捕捉模式

在捕捉模式下，当在 CCP2 引脚（RC1 或 RE7 引脚，取决于器件配置）上发生事件时，CCPR2H:CCPR2L 寄存器对捕捉 TMR1 或 TMR3 寄存器的 16 位值。事件定义为以下情况之一：

- 每个下降沿
- 每个上升沿
- 每 4 个上升沿
- 每 16 个上升沿

事件由模式选择位 CCP2M<3:0>（CCP2CON<3:0>）选择。当完成一次捕捉时，中断请求标志位 CCP2IF（PIR3<2>）置 1；它必须用软件清零。如果在读取寄存器 CCPR2 值之前发生了另一次捕捉，那么原来的捕捉值会被新的捕捉值覆盖。

16.2.1 CCP 引脚配置

在捕捉模式下，应通过将相应的 TRIS 方向位置 1 将 CCPx 引脚配置为输入。

注：	如果 RC1/CCP2 或 RE7/CCP2 引脚被配置为输出，则对端口的写操作可能产生捕捉条件。

16.2.2 TIMER1/TIMER3 模式选择

用于捕捉功能的定时器（Timer1 和 / 或 Timer3）必须运行在定时器模式或同步计数器模式下。在异步计数器模式下，可能无法进行捕捉操作。可在 T3CON 寄存器中选择用于每个 CCP 模块的定时器（见**第 16.1.1 节 “CCP 模块和定时器资源 ”**）。

16.2.3 软件中断

当捕捉模式改变时，可能会产生错误的捕捉中断。用户应该保持 CCP2IE 位（PIE3<2>）清零以避免错误中断，并且还应该在工作模式改变后清零标志位 CCP2IF。

16.2.4 CCP 预分频器

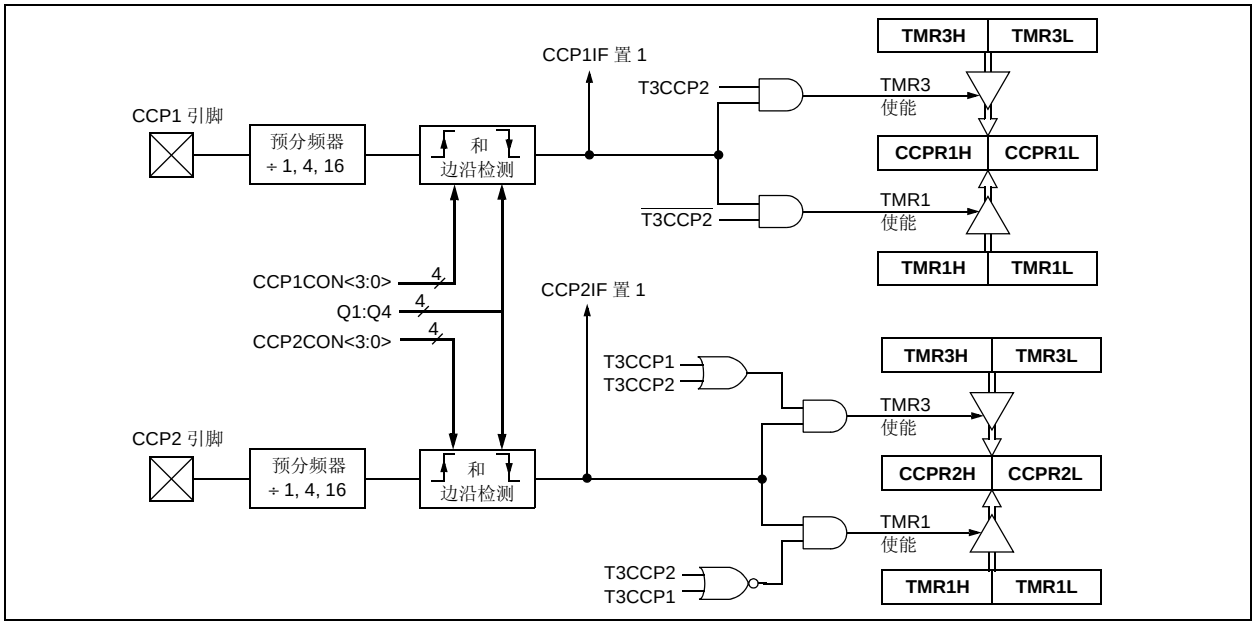
在捕捉模式下有 4 种预分频比设置。它们作为工作模式的一部分由模式选择位（CCP2M<3:0>）指定。每当关闭 CCP 模块，或者 CCP 模块不处于捕捉模式时，预分频器计数器就会被清零。这意味着任何复位都会将预分频器计数器清零。

在两个捕捉预分频比之间切换可能会产生中断。而且，预分频器计数器不会被清零；因此第一次捕捉可能来自于一个非零的预分频器。例 16-1 给出了切换捕捉预分频比时建议采用的方法。这个示例使预分频器计数器清零且不会产生错误中断。

例 16-1： 改变捕捉预分频比

```
CLRF   CCP2CON      ; Turn CCP module off
MOVLW  NEW_CAPT_PS  ; Load WREG with the
                    ; new prescaler mode
                    ; value and CCP ON
MOVWF  CCP2CON      ; Load CCP2CON with
                    ; this value
```

图 16-2： 捕捉模式工作原理框图



16.3 比较模式

在比较模式下，16 位 CCPR2 寄存器的值不断与 TMR1 或 TMR3 寄存器对的值作比较。当两者匹配时， CCP2 引脚将会：

- 驱动为高电平
- 驱动为低电平
- 电平翻转（高电平变为低电平或低电平变为高电平）
- 保持不变 （即反映 I/O 锁存器的状态）

引脚动作取决于模式选择位（CCP2M<3:0>）的值。同时，中断标志位 CCP2IF 置 1。

16.3.1 CCP 引脚配置

用户必须通过将相应的 TRIS 位清零，将 CCPx 引脚配置为输出。

注：	清零 CCP2CON 寄存器会将 RC1 或 RE7 比较输出锁存器（取决于器件配置）强制为默认的低电平。这不是 PORTC 或 PORTE I/O 数据锁存器。

16.3.2 TIMER1/TIMER3 模式选择

如果 CCP 模块使用比较功能，则 Timer1 和 / 或 Timer3 必须运行在定时器模式或同步计数器模式下。在异步计数器模式下，可能无法进行比较操作。

16.3.3 软件中断模式

当选择了 “ 产生软件中断 ” 模式（CCP2M<3:0> = 1010）时，CCP2 引脚不受影响。如果中断被允许，将仅产生一个 CCP 中断并将 CCP2IE 位置 1。

16.3.4 特殊事件触发器

两个 CCP 模块均配备了一个特殊事件触发器。在比较模式下可产生内部硬件信号以触发其他模块动作。通过选择 “ 比较特殊事件触发 ” 模式（CCP2M<3:0> = 1011），使能特殊事件触发器。

对于任何一个 CCP 模块，无论当前使用哪个定时器资源作为模块的时基，特殊事件触发器将把对应的定时寄存器对复位。这样 CCPRx 寄存器可用作两个定时器中任一定时器的可编程周期寄存器。

CCP2 的特殊事件触发信号还能启动 A/D 转换。要实现此功能，必须首先使能 A/D 转换器。

注：	CCP1 特殊事件触发信号只复位 Timer1/Timer3，即使在使能了 A/D 转换器时也不能启动 A/D 转换。

图 16-3： 比较模式工作原理框图

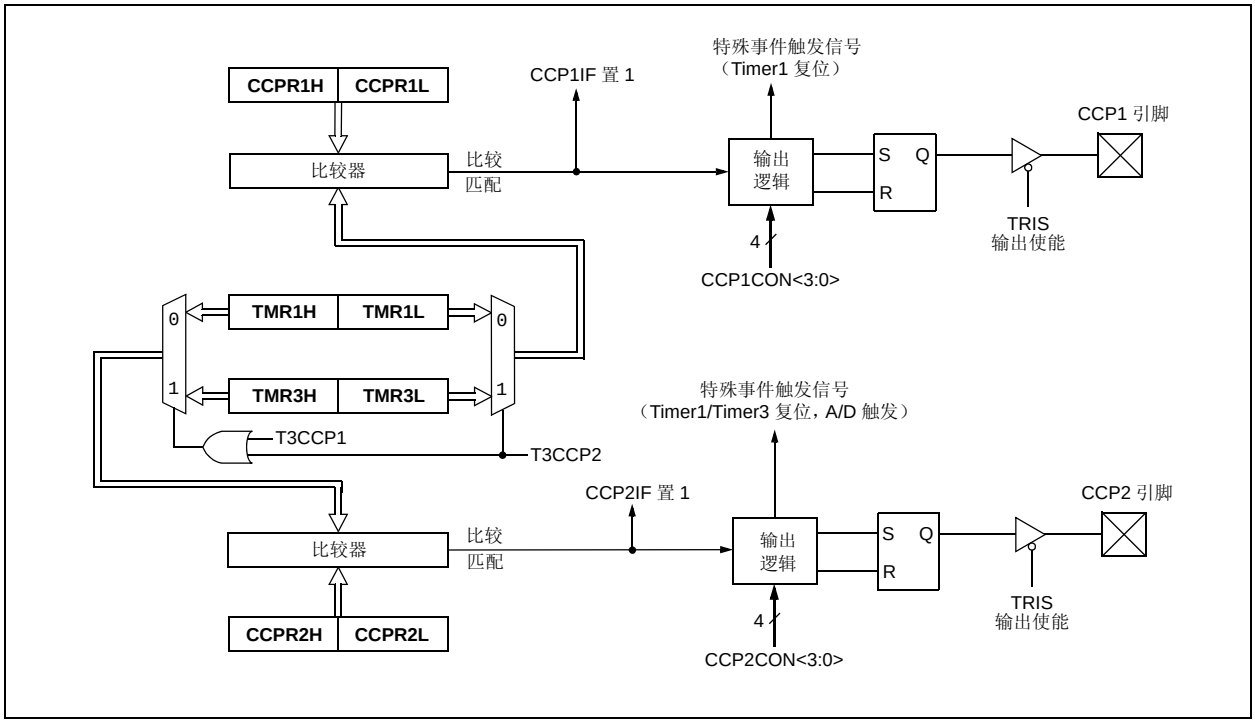


表 16-3：　　与捕捉、比较、 TIMER1 和 TIMER3 相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
RCON	IPEN	—	$\overline{CM}$	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	60
PIR3	—	LCDIF	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	RTCCIF	62
PIE3	—	LCDIE	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	RTCCIE	62
IPR3	—	LCDIP	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	RTCCIP	62
PIR2	OSCFIF	CMIF	—	—	BCLIF	LVDIF	TMR3IF	—	62
PIE2	OSCFIE	CMIE	—	—	BCLIE	LVDIE	TMR3IE	—	62
IPR2	OSCFIP	CMIP	—	—	BCLIP	LVDIP	TMR3IP	—	62
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	62
TRISE	TRISE7	TRISE6	TRISE5	TRISE4	TRISE3	—	TRISE1	TRISE0	62
TRISG	SPIOD	CCP2OD	CCP1OD	TRISG4	TRISG3	TRISG2	TRISG1	TRISG0	62
TMR1L	Timer1 寄存器的低字节								60
TMR1H	Timer1 寄存器的高字节								60
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	60
TMR3H	Timer3 寄存器的高字节								61
TMR3L	Timer3 寄存器的低字节								61
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{T3SYNC}$	TMR3CS	TMR3ON	61
CCPR1L	捕捉 / 比较 /PWM 寄存器 1 的低字节								63
CCPR1H	捕捉 / 比较 /PWM 寄存器 1 的高字节								63
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	63
CCPR2L	捕捉 / 比较 /PWM 寄存器 2 的低字节								64
CCPR2H	捕捉 / 比较 /PWM 寄存器 2 的高字节								63
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	64

图注：　　— = 未实现，读为 0。捕捉 / 比较、 Timer1 或 Timer3 不使用阴影单元。

16.4 PWM 模式

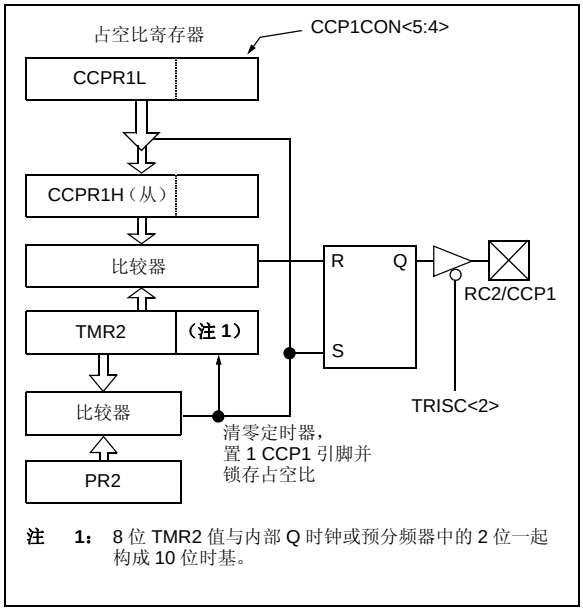
在脉宽调制 （PWM）模式下， CCP2 引脚会产生最高 10 位分辨率的 PWM 输出信号。由于 CCP2 引脚与 PORTC 或 PORTE 数据锁存器复用，必须清零相应的 TRIS 位才能使 CCP2 引脚成为输出引脚。

注：	清零 CCP2CON 寄存器会将 RC1 或 RE7 输出锁存器 （取决于器件配置）强制为默认的低电平。这不是PORTC或PORTE I/O数据锁存器。

图 16-4 给出了 PWM 模式下 CCP1 模块的简化框图。

关于如何设置 CCP 模块使之工作于 PWM 模式的详细步骤，请参见**第 16.4.3 节 “ 设置 PWM 操作 ”**。

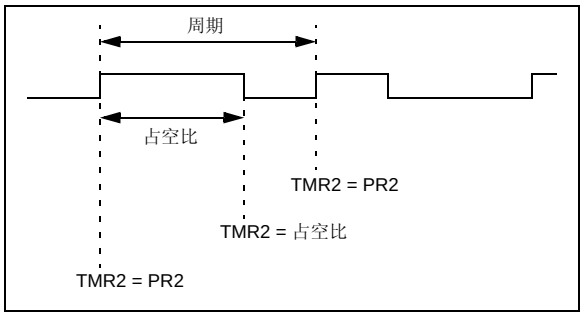
图 16-4： 简化的 PWM 框图



注 1： 8 位 TMR2 值与内部 Q 时钟或预分频器中的 2 位一起构成 10 位时基。

PWM 输出（图 16-5）有一个时基（周期）和一段输出保持为高电平的时间 （占空比）。 PWM 的频率是周期的倒数 （1/ 周期）。

图 16-5： PWM 输出



16.4.1 PWM 周期

可通过写 PR2 寄存器指定 PWM 周期。PWM 周期可由以下公式计算：

公式 16-1：

$$\text{PWM 周期} = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (\text{TMR2 预分频值})$$

PWM 频率定义为 1/[PWM 周期]。

当 TMR2 中的值与 PR2 中的值相等时，在下一个递增周期将发生以下 3 个事件：

- TMR2 被清零
- CCP2引脚置1（例外情况：如果PWM占空比 = 0%，CCP2 引脚将不会置 1）
- PWM 占空比从 CCPR2L 锁存到 CCPR2H

注：	在确定 PWM 频率时不会用到 Timer2 后分频比 （见**第 13.0 节 “Timer2 模块 ”**）。后分频器可用不同于PWM输出频率的频率进行数据更新。

16.4.2 PWM 占空比

通过写入 CCPR2L 寄存器和 CCP2CON<5:4> 位来指定 PWM 占空比。分辨率最高可达 10 位。CCPR2L 包含高 8 位而 CCP2CON<5:4> 包含低 2 位。这 10 位值由 CCPR2L:CCP2CON<5:4> 表示。以下公式用于计算 PWM 的占空比（用时间来表示）：

公式 16-2：

PWM 占空比 = (CCPR2L:CCP2CON<5:4>) •
TOSC • (TMR2 预分频值)

可以在任何时候写入 CCPR2L 和 CCP2CON<5:4>，但是在 PR2 和 TMR2 发生匹配（即周期结束）前占空比值不会被锁存到 CCPR2H 中。在 PWM 模式下，CCPR2H 是只读寄存器。

CCPR2H 寄存器和一个 2 位的内部锁存器用于给 PWM 占空比提供双重缓冲。这种双重缓冲结构非常重要，它可以避免在 PWM 操作中产生毛刺。

当 CCPR2H 和 2 位锁存值与 TMR2（以及内部 2 位 Q 时钟或 TMR2 预分频值的 2 位）匹配时，CCP2 引脚被清零。

在给定 PWM 频率的情况下，最大的 PWM 分辨率（位）由以下公式给出：

公式 16-3：

$$\text{PWM 分辨率（最大）} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ 位}$$

注： 如果 PWM 占空比的值大于 PWM 周期，则 CCP2 引脚将不会被清零。

表 16-4： 40 MHz 时 PWM 频率和分辨率示例

PWM 频率	2.44 kHz	9.77 kHz	39.06 kHz	156.25 kHz	312.50 kHz	416.67 kHz
定时器预分频值（1、4 和 16）	16	4	1	1	1	1
PR2 值	FFh	FFh	FFh	3Fh	1Fh	17h
最大分辨率（位）	14	12	10	8	7	6.58

16.4.3 设置 PWM 操作

当配置 CCP 模块的 PWM 操作时，可采用以下步骤：

1. 通过写 PR2 寄存器设置 PWM 周期。
2. 通过写 CCPR2L 寄存器和 CCP2CON<5:4> 位设置 PWM 占空比。
3. 通过清零相应的 TRIS 位将 CCP2 引脚设为输出引脚。
4. 通过写 T2CON 设置 TMR2 预分频值并随后使能 Timer2。
5. 配置 CCP2 模块使之工作于 PWM 模式。

表 16-5： 与 PWM 和 TIMER2 相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
RCON	IPEN	—	$\overline{CM}$	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	60
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	62
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	62
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	62
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	62
TRISE	TRISE7	TRISE6	TRISE5	TRISE4	TRISE3	—	TRISE1	TRISE0	62
TRISG	SPIOD	CCP2OD	CCP1OD	TRISG4	TRISG3	TRISG2	TRISG1	TRISG0	62
TMR2	Timer2 寄存器								60
PR2	Timer2 周期寄存器								60
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	60
CCPR1L	捕捉 / 比较 /PWM 寄存器 1 的低字节								63
CCPR1H	捕捉 / 比较 /PWM 寄存器 1 的高字节								63
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	63
CCPR2L	捕捉 / 比较 /PWM 寄存器 2 的低字节								64
CCPR2H	捕捉 / 比较 /PWM 寄存器 2 的高字节								63
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	64

图注： — = 未实现，读为 0。 PWM 或 Timer2 不使用阴影单元。

注：

17.0 液晶显示（LCD）驱动模块

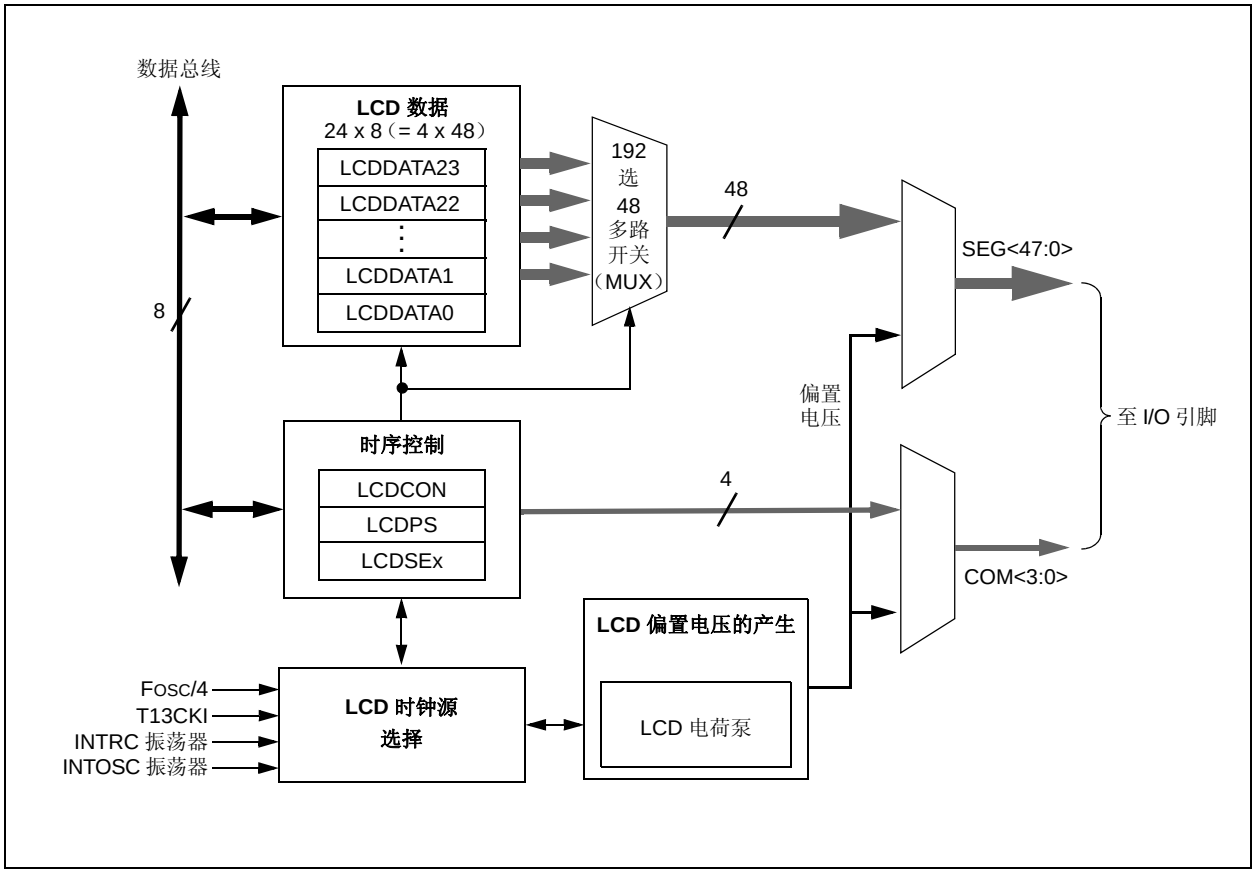
液晶显示（Liquid Crystal Display，LCD）驱动模块产生时序控制驱动静态或复用的 LCD 面板，还可以控制 LCD 像素数据。在 PIC18F8XJ90 器件中，模块可以驱动最多 192 像素（4 个公共端、48 段）的面板；在 PIC18F6XJ90 器件中，模块可以驱动最多 132 像素（4 个公共端、33 段）的面板。

LCD 驱动模块支持以下功能：

- LCD 面板的直接驱动
- 带有专用电荷泵的片上偏置电压生成电路，支持一系列的固定和可变的偏置选项
- 最多 4 个公共端，4 种复用模式
- 最多48段（PIC18F8XJ90器件）或33段（PIC18F6XJ90器件）
- 带有可选预分频比的 3 个 LCD 时钟源，还有第 4 个时钟源可用于 LCD 电荷泵

图 17-1 给出了该模块的简化框图。

图 17-1： LCD 驱动模块框图



17.1 LCD 寄存器

LCD 驱动模块具有 33 个寄存器：

- LCD 控制寄存器 （LCDCON）
- LCD 相位寄存器 （LCDPS）
- LCDREG 寄存器 （LCD 稳压器控制）
- 6 个 LCD 段使能寄存器 （LCDSE5:LCDSE0）
- 24 个 LCD 数据寄存器 （LCDDATA23:LCDDATA0）

17.1.1 LCD 控制寄存器

LCDCON 寄存器 （如寄存器 17-1 所示）控制模块的整体操作。完成模块的配置之后，就可使用 LCDEN 位（LCDCON<7>）使能或禁止 LCD 模块。通过清零 SLPEN 位（LCDCON<6>）可使 LCD 面板在休眠模式下继续工作。

LCDPS 寄存器（如寄存器 17-2 所示）用于配置 LCD 时钟源预分频比和波形 （A 型或 B 型）。关于这些特性的详细信息，请参见**第 17.2 节 “LCD 时钟源 ”**、**第 17.3 节 “LCD 偏置电压的产生 ”** 和**第 17.8 节 “LCD 波形产生 ”**。

在**第 17.3 节 “LCD 偏置电压的产生 ”** 中介绍了 LCDREG 寄存器。

LCD 段使能寄存器 （LCDSEx）用于配置端口引脚的功能。通过置 1 特定段的段使能位，将该引脚配置为 LCD 驱动引脚。寄存器 17-3 所示为 LCDSE 寄存器原型。共有 6 个 LCDSE 寄存器（LCDSE5:LCDSE0），如表 17-1 中所示。

寄存器 17-1： LCDCON：LCD 控制寄存器

R/W-0	R/W-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
LCDEN	SLPEN	WERR	—	CS1	CS0	LMUX1	LMUX0
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 7 **LCDEN：** LCD 驱动使能位
1 = 使能 LCD 驱动模块
0 = 禁止 LCD 驱动模块

bit 6 **SLPEN：** 休眠模式下 LCD 驱动使能位
1 = 休眠模式下禁止 LCD 驱动模块
0 = 休眠模式下使能 LCD 驱动模块

bit 5 **WERR：** LCD 写失败错误位
1 = 当 LCDPS<4> = 0 时写 LCDDATAx 寄存器 （必须用软件清零）
0 = 无 LCD 写错误

bit 4 **未实现：** 读为 0

bit 3-2 **CS<1:0>：** 时钟源选择位
1x = INTRC （31 kHz）
01 = T13CKI （Timer1）
00 = 系统时钟 （FOSC/4）

bit 1-0 **LMUX<1:0>：** 公共端选择位

LMUX<1:0>	复用类型	最大像素数：		偏置类型
		PIC18F6XJ90	PIC18F8XJ90	
00	静态 （COM0）	33	48	静态
01	1/2 （COM1:COM0）	66	96	1/2 或 1/3
10	1/3 （COM2:COM0）	99	144	1/2 或 1/3
11	1/4 （COM3:COM0）	132	192	1/3

寄存器 17-2：　　LCDPS：LCD 相位寄存器

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
WFT	BIASMD	LCDA	WA	LP3	LP2	LP1	LP0
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　**WFT**：波形选择位
1 = B 型波形 （在每一帧边界改变相位）
0 = A 型波形 （在每一公共端类型内改变相位）

bit 6　**BIASMD**：偏置模式选择位
当 LMUX<1:0> = 00 时：
0 = 静态偏置模式 （不要将该位置 1）
当 LMUX<1:0> = 01 或 10 时：
1 = 1/2 偏置模式
0 = 1/3 偏置模式
当 LMUX<1:0> = 11 时：
0 = 1/3 偏置模式 （不要将该位置 1）

bit 5　**LCDA**：LCD 有效状态位
1 = LCD 驱动模块有效
0 = LCD 驱动模块无效

bit 4　**WA**：LCD 写允许状态位
1 = 允许写入 LCDDATAx 寄存器
0 = 禁止写入 LCDDATAx 寄存器

bit 3-0　**LP<3:0>**：LCD 预分频比选择位
1111 = 1:16
1110 = 1:15
1101 = 1:14
1100 = 1:13
1011 = 1:12
1010 = 1:11
1001 = 1:10
1000 = 1:9
0111 = 1:8
0110 = 1:7
0101 = 1:6
0100 = 1:5
0011 = 1:4
0010 = 1:3
0001 = 1:2
0000 = 1:1

寄存器 17-3： LCDSEx：LCD 段使能寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SE(n + 7)	SE(n + 6)	SE(n + 5)	SE(n + 4)	SE(n + 3)	SE(n + 2)	SE(n + 1)	SE(n)
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 7-0 **SEG(n + 7):SEG(n)：**段使能位
对于 LCDSE0：n = 0
对于 LCDSE1：n = 8
对于 LCDSE2：n = 16
对于 LCDSE3：n = 24
对于 LCDSE4：n = 32
对于 LCDSE5：n = 40
1 = 使能引脚的段功能，禁止其数字 I/O 功能
0 = 使能引脚的 I/O 功能

表 17-1： LCDSE 寄存器和相关段

寄存器	段
LCDSE0	7:0
LCDSE1	15:8
LCDSE2	23:16
LCDSE3	31:24
LCDSE4(1)	39:32
LCDSE5(2)	47:40

注 1： LCDSE4<7:1>（SEG<39:33>）在 PIC18F6XJ90 器件中未实现。
2： LCDSE5 在 PIC18F6XJ90 器件中未实现。

17.1.2 LCD 数据寄存器

一旦为LCD面板初始化了模块，LCDDATA23:LCDDATA0寄存器中的各位就会被清零或置 1，分别代表透明或不透明像素。特定的段和公共端信号使用了一组特定的 LCDDATA 寄存器。寄存器中每一位均表示特定段同特定公共端间的唯一组合。

LCDDATA 位命名约定为 “SxxCy”，其中 “xx” 表示段号， “y” 表示公共端号。表 17-2 总结了两者间的关系。寄存器 17-4 为 LCDDATA 寄存器原型。

注：	在64引脚器件中，写入寄存器LCDDATA5、LCDDATA11、LCDDATA17和LCDDATA23不会影响任何像素的状态。

寄存器 17-4： LCDDATAx：LCD 数据寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
S(n + 7)Cy	S(n + 6)Cy	S(n + 5)Cy	S(n + 4)Cy	S(n + 3)Cy	S(n + 2)Cy	S(n + 1)Cy	S(n)Cy
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 7-0 **S(n + 7)Cy:S(n)Cy：**像素点亮位

对于 LCDDATA0 到 LCDDATA5：n = (8x)， y = 0

对于 LCDDATA6 到 LCDDATA11：n = (8(x – 6))， y = 1

对于 LCDDATA12 到 LCDDATA17：n = (8(x – 12))， y = 2

对于 LCDDATA18 到 LCDDATA23：n = (8(x – 18))， y = 3

1 = 点亮像素 （不透明）

0 = 不点亮像素 （透明）

表 17-2： LCDDATA 寄存器位和段与公共端组合的对应关系

段	公共端			
	0	**1**	**2**	**3**
0 到 7	LCDDATA0	LCDDATA6	LCDDATA12	LCDDATA18
	S00C0:S07C0	S00C1:S07C1	S00C2:S07C2	S00C3:S07C3
8 到 15	LCDDATA1	LCDDATA7	LCDDATA13	LCDDATA19
	S08C0:S15C0	S08C1:S15C1	S08C2:S15C2	S08C0:S15C3
16 到 23	LCDDATA2	LCDDATA8	LCDDATA14	LCDDATA20
	S16C0:S23C0	S16C1:S23C1	S16C2:S23C2	S16C3:S23C3
24 到 31	LCDDATA3	LCDDATA9	LCDDATA15	LCDDATA21
	S24C0:S31C0	S24C1:S31C1	S24C2:S31C2	S24C3:S31C3
32 到 39	LCDDATA4(1)	LCDDATA10(1)	LCDDATA16(1)	LCDDATA22(1)
	S32C0:S39C0	S32C1:S39C1	S32C2:S39C2	S32C3:S39C3
40 到 47	LCDDATA5(2)	LCDDATA11(2)	LCDDATA17(2)	LCDDATA23(2)
	S40C0:S47C0	S40C1:S47C1	S40C2:S47C2	S40C3:S47C3

注 1： 这些寄存器的 bit <7:1> 在 PIC18F6XJ90 器件中未实现。这些寄存器 （SEG32Cy）的 bit 0 始终实现。

2： 这些寄存器在 PIC18F6XJ90 器件上未实现。

17.2 LCD 时钟源

LCD 驱动模块可以使用 3 种时钟源来生成其内部时钟：

- 系统时钟 （FOSC/4）
- Timer1 振荡器
- INTRC 时钟源

不论选择何种时钟源，LCD 时钟发生器都会使用可配置的 32 分频 /8192 分频后分频器来生成标称值约为 1 kHz 的基准频率。时钟源选择和后分频比配置取决于时钟源选择位 CS<1:0> （LCDCON<3:2>）。

可以使用另一个可编程预分频器从 1 kHz 基准频率获得 LCD 帧频率。可以使用 LP<3:0> 位 （LCDPS<3:0>）将预分频比配置为 16 个选项之一，范围从 1:1 到 1:16。

波形产生的正确时序由LMUX<1:0>位（LCDCON<1:0>）设置。这些位决定所使用的公共端复用模式，并按需要对 LCD 时钟源分频。它们还决定环形计数器（用于控制 LCD 公共端的打开或关闭）的配置。

17.2.1 LCD 稳压器时钟源

除了用于 LCD 时序的时钟源外， LCD 电荷泵还需要一个独立的标称值为 31 kHz 的时钟。这由 LCD 时钟源的另一分支提供。

电荷泵时钟可以使用 Timer1 振荡器或 INTRC 时钟源，以及 8 MHz INTOSC 时钟源（经过预分频器进行 256 分频后） 。电荷泵时钟源通过 CKSEL<1:0> 位（LCDREG<1:0>）进行配置。

17.2.2 时钟源注意事项

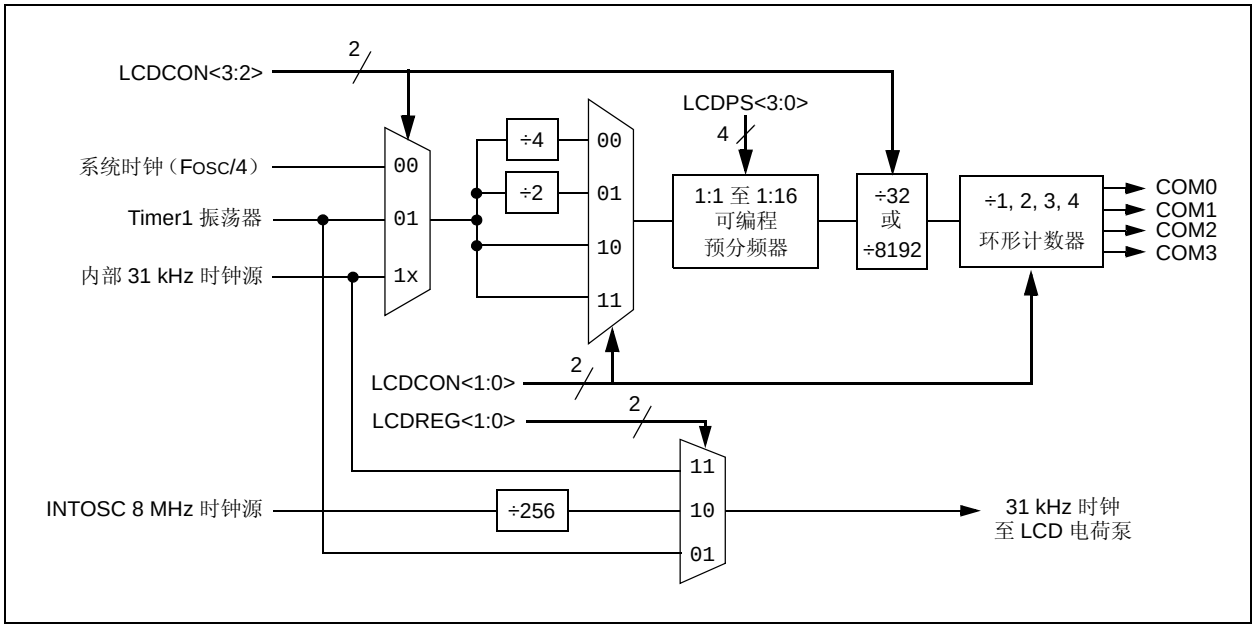
当使用系统时钟作为LCD时钟源时，假设系统时钟频率为标称值32 MHz（FOSC/4频率为8 MHz）。由于FOSC/4时钟选择的预分频比选项固定为 8192 分频，如果系统时钟速度不是 32 MHz，则其所产生的帧频率和刷新速率与本章中所讨论的不同。用户在设计显示应用时应谨记这一点。

当器件处于休眠模式时，可以使用 Timer1 和 INTRC 源作为 LCD 时钟源。要使用 Timer1 振荡器，需要将T1OSCEN位（T1CON<3>）置1。选择Timer1或INTRC作为 LCD 时钟源不会自动激活这些时钟源。

同样，选择 INTOSC 作为电荷泵时钟源不会自动打开振荡器。要使用 INTOSC，必须通过 FOSC2 配置位将其选择为系统时钟源。

如果使用 Timer1 作为器件的时钟源（用作 LCD 时钟源或其他用途），则 LCD 的段 32 变为不可用。

图 17-2： LCD 时钟的产生



17.3 LCD 偏置电压的产生

LCD驱动模块能够使用最少量的外部元件产生LCD操作所需的偏置电压。这包括根据 LCD 所需的不同偏置类型产生所需的不同电压的能力。通过使用片上 LCD 稳压器，驱动模块还可以提供高于和低于单片机 VDD 的偏置电压。

17.3.1 LCD 偏置类型

根据产生的用于控制段和公共端的波形，PIC18F87J90 系列器件支持 3 种偏置类型：

- 静态 （两种离散电平）
- 1/2 偏置 （三种离散电平）
- 1/3 偏置 （四种离散电平）

在**第 17.8 节 “LCD 波形产生 ”** 中对使用不同波形来驱动 LCD 进行了更详细的讨论。

17.3.2 LCD 稳压器

LCD 稳压器的目的在于为 LCD 提供合适的偏置电压和良好的对比度，而与 VDD 电压无关。该模块包含一个电荷泵和内部参考电压。稳压器可以通过使用外部元件将偏置电压提高到 VDD 以上。也可以在低于 VDD 的某个固定电压下进行显示操作。用户也可以选择禁止该稳压器，通过外部电阻网络来产生偏置电压。

LCD 稳压器由 LCDREG 寄存器控制 （寄存器 17-5）。可以使用 CKSEL<1:0> 位使能或禁止该稳压器，同时可以使用 CPEN 位有选择地使能电荷泵。使能稳压器时，MODE13 位用于选择偏置类型。 LCD 偏置电压峰值为 LCDBIAS3 和 LCDBIAS0 之间的电位差，由 BIAS 位进行配置。

寄存器 17-5： LCDREG：稳压器控制寄存器

U-0	RW-0	RW-1	RW-1	RW-1	RW-1	RW-0	RW-0
—	CPEN	BIAS2	BIAS1	BIAS0	MODE13	CKSEL1	CKSEL0
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 **未实现：** 读为 0

bit 6 **CPEN：** LCD 电荷泵使能位
1 = 使能电荷泵，最高 LCD 偏置电压为 3.6V
0 = 禁止电荷泵，最高 LCD 偏置电压为 AVDD

bit 5-3 **BIAS<2:0>：** 稳压器电压输出控制位
111 = 峰值为 3.60V （在 LCDBIAS0 上的偏移为 0V）
110 = 峰值为 3.47V （在 LCDBIAS0 上的偏移为 0.13V）
101 = 峰值为 3.34V （在 LCDBIAS0 上的偏移为 0.26V）
100 = 峰值为 3.21V （在 LCDBIAS0 上的偏移为 0.39V）
011 = 峰值为 3.08V （在 LCDBIAS0 上的偏移为 0.52V）
010 = 峰值为 2.95V （在 LCDBIAS0 上的偏移为 0.65V）
001 = 峰值为 2.82V （在 LCDBIAS0 上的偏移为 0.78V）
000 = 峰值为 2.69V （在 LCDBIAS0 上的偏移为 0.91V）

bit 2 **MODE13：** 1/3 LCD 偏置使能位
1 = 稳压器输出支持 1/3 LCD 偏置模式
0 = 稳压器输出支持静态 LCD 偏置模式

bit 1-0 **CKSEL<1:0>：** 稳压器时钟源选择位
11 = INTRC
10 = INTOSC 8 MHz 时钟源
01 = Timer1 振荡器
00 = 禁止 LCD 稳压器

17.3.3 偏置配置

PIC18F87J90 系列器件有 4 种不同的电路配置可用于产生 LCD 偏置：

- M0：带升压电路的稳压器
- M1：不带升压电路的稳压器
- M2：带软件对比度控制的梯形电阻网络
- M3：带硬件对比度控制的梯形电阻网络

17.3.3.1 M0 （带升压电路的稳压器）

在 M0 操作中，使能了 LCD 电荷泵功能。这使稳压器可以产生最高 +3.6V 的 LCD 驱动电压（在 LCDBIAS3 引脚测得）。

M0 在 VLCAP1 和 VLCAP2 之间连接了一个反激电容，并为 LCDBIAS0 到 LCDBIAS3 连接了滤波电容，以实现所需的电压升压 （图 17-3） 。输出电压 （VBIAS）是 LCDBIAS3 和 LCDBIAS0 之间的电位差。它由调节 LCDBIAS0 和 VSS 之间偏移的 BIAS<2:0> 位进行设置。反激电容 （CFLY）充当 LCD 负载较大时的电荷存储元件。该模式在要求 LCD 电压比单片机的 VDD 高的情况下很有用。它同样允许通过改变 BIAS 位的值调节偏置电压，从而实现对显示对比度的软件控制。

M0 支持静态和 1/3 偏置类型。1/3 偏置电压是自动生成的，但必须用软件配置。

通过选择一个有效的稳压器时钟源（CKSEL<1:0> 设置为除 00 外的任何值）并将 CPEN 位置 1，可以使能 M0。如果需要静态偏置类型，必须将 MODE13 位清零。

17.3.3.2 M1 （不带升压电路的稳压器）

M1 操作和 M0 类似，但不使用 LCD 电荷泵。它能够提供的最大 VBIAS 电压为直接提供给 LCDBIAS3 的电压。在希望 VDD 永远不会下降到保证 LCD 维持足够对比度的电压之下的应用中，可以使用该偏置类型。外部元件的连接和 M0 的非常类似，不同之处在于 LCDBIAS3 必须直接连接到 VDD （图 17-3）。

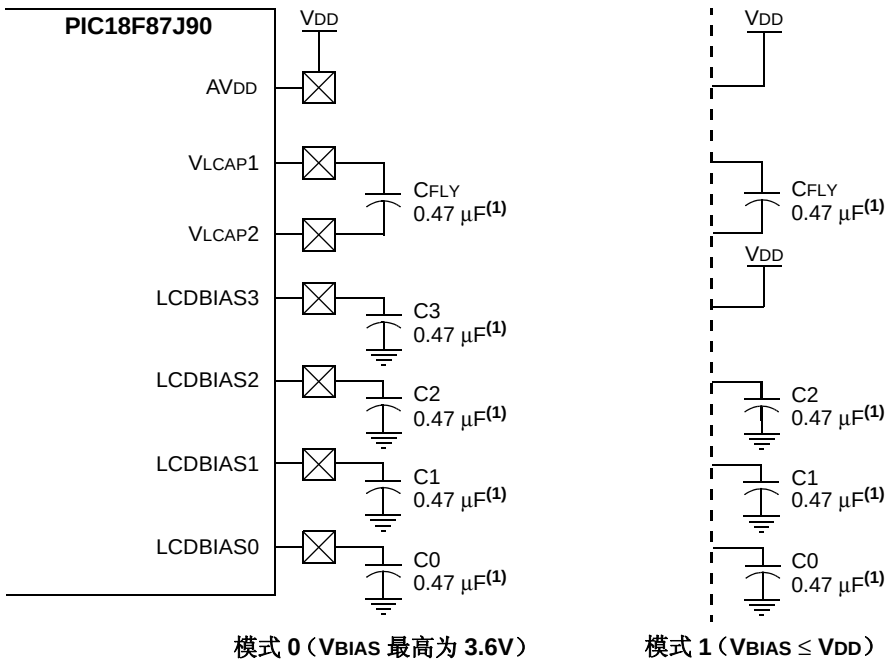
注：	当器件工作在模式 0 或模式 1 时进入休眠状态，应确保偏置电容彻底放电，以使休眠电流最小。

通过改变 VBIAS， BIAS<2:0> 位仍然可用于用软件调节对比度。和 M0 一样，改变这些位将更改 LCDBIAS0 和 VSS 之间的偏移。在 M1 中，这反映在 LCDBIAS0 和 LCDBIAS3 之间的电压变化。因此，如果 VDD 发生改变，VBIAS 也将改变；而在 M0 中， VBIAS 的电压是不变的。

和 M0 一样，M1 支持静态和 1/3 偏置类型。1/3 偏置电压是自动生成的，但必须用软件配置。

通过选择一个有效的稳压器时钟源（CKSEL<1:0> 设置为除 00 外的任何值）并清零 CPEN 位，可以使能 M1。如果需要 1/3 偏置类型，还要将 MODE13 位置 1。

图 17-3： M0 和 M1 配置的 LCD 稳压器连接



注 1： 这些值仅供设计参考，设计人员应根据实际 LCD 规格对应用进行优化。

17.3.3.3 M2（带软件对比度控制的梯形电阻网络）

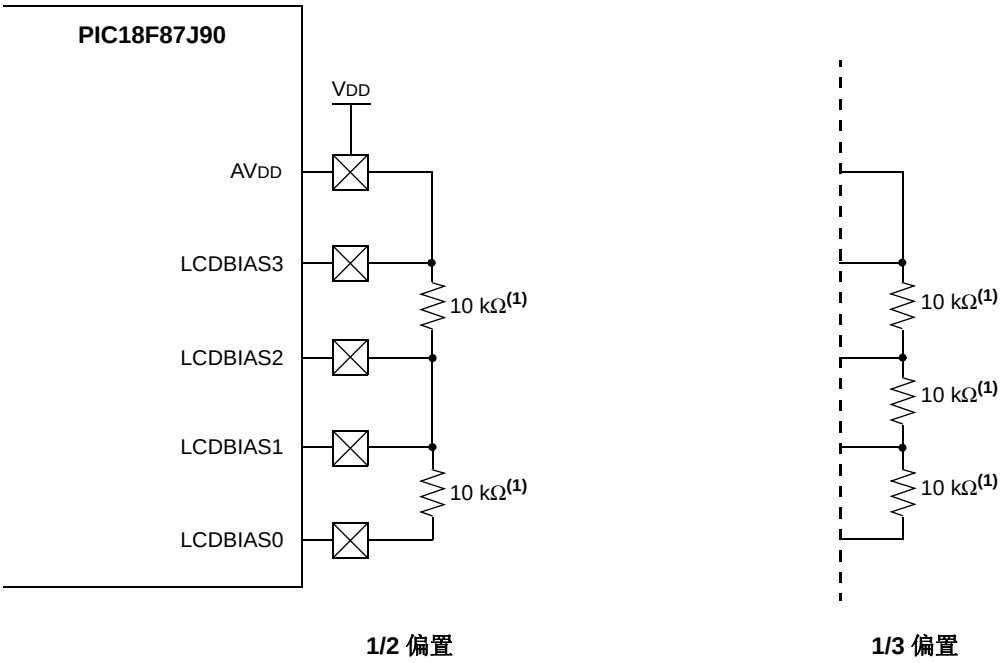
M2 操作同样使用 LCD 稳压器，但禁止电荷泵。稳压器的内部参考电压保持为有效，以实现对比度调节。该类型用于LCD的电流需求超出稳压器电荷泵能力的情况。

在此配置中，LCD 偏置电压是通过连接到LCDBIAS0 和 LCDBIAS3 的外部电阻分压器产生的，该分压器顶部连接到 VDD（图 17-4）。梯形底部的电势取决于 LCD 稳压器的参考电压（内部连接到 LCDBIAS0）。偏置类型取决于 LCDBIAS 引脚的电压，该电压受梯形电阻网络的配置控制。使用 M2 的大部分应用将使用 1/3 或 1/2 偏置类型。虽然同样可以使用静态偏置，但其提供的对比度范围十分有限，并与其他偏置产生模式相比要消耗额外的电流。

和 M1 一样，可以使用 LCDBIAS 位控制对比度，这受到供给器件的 VDD 电压的限制。此外，由于在 VLCAP1 和 VLCAP2 之间不需要电容，因此这些引脚可以用作数字 I/O 端口 RG2 和 RG3。

通过清零 CKSEL<1:0> 位并将 CPEN 位置 1，可以选择 M2。

图 17-4： M2 配置的梯形电阻网络连接



引脚处的偏置电平	偏置类型	
	1/2 偏置	1/3 偏置
LCDBIAS0	（内部低参考电压）	（内部低参考电压）
LCDBIAS1	1/2 VBIAS	1/3 VBIAS
LCDBIAS2	1/2 VBIAS	2/3 VBIAS
LCDBIAS3	VBIAS（最大 AVDD）	VBIAS（最大 AVDD）

注 1： 这些值仅供设计参考，设计人员应根据实际 LCD 规格对应用进行优化。

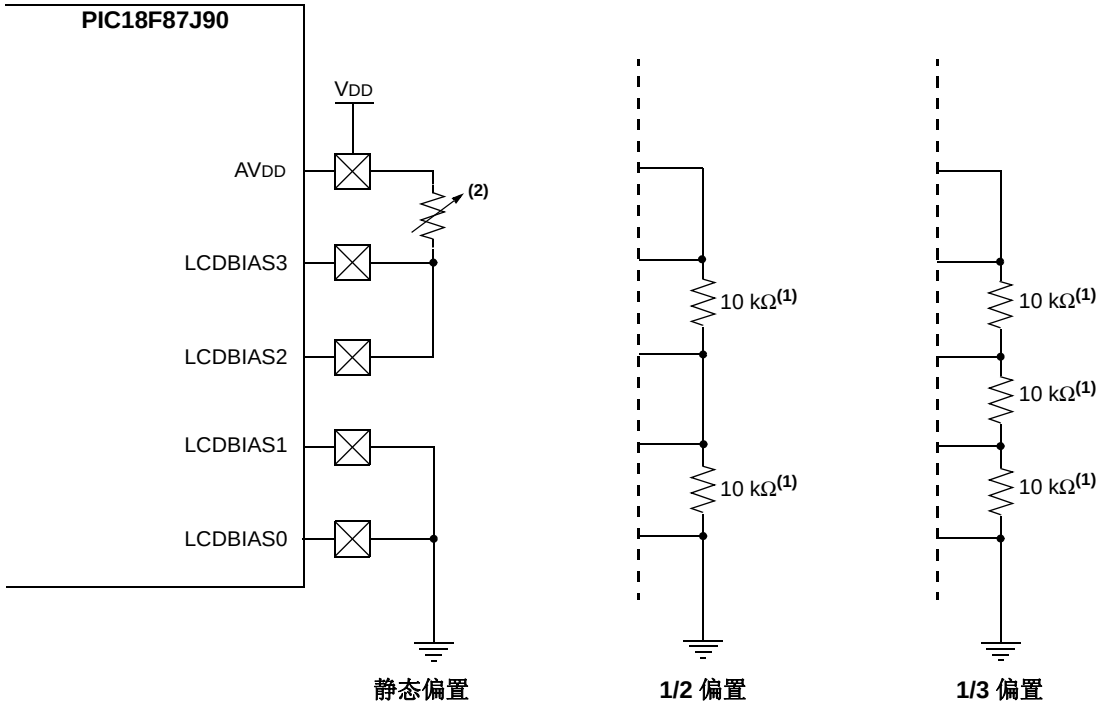
17.3.3.4 M3 （硬件对比度控制）

在 M3 中，LCD 稳压器是完全禁止的。和 M2 一样，LCD 偏置电压连接到 AVDD 并使用外部分压器产生。不同之处在于内部参考电压也被禁止，并且梯形网络的底部接地（VSS），如图 17-5 所示。电阻值以及 VSS 和 VDD 之间的电压差值决定对比度的范围，不能使用软件进行调节。此配置同样用于 LCD 的电流需求超出电荷泵能力并且不需要软件对比度控制的情况。

根据所需的偏置类型，要在某些或全部引脚之间连接电阻。也可以在 LCDBIAS3 和 VDD 之间接一个电位器，来进行硬件控制的对比度调节。

通过清零 CKSEL<1:0> 和 CPEN 位，可以选择 M3。

图 17-5： M3 配置的梯形电阻网络连接



引脚处的偏置电平	偏置类型		
	静态	1/2 偏置	1/3 偏置
LCDBIAS0	AVSS	AVSS	AVSS
LCDBIAS1	AVSS	1/2 AVDD	1/3 AVDD
LCDBIAS2	AVDD	1/2 AVDD	2/3 AVDD
LCDBIAS3	AVDD	AVDD	AVDD

注 1： 这些值仅供设计参考，设计人员应根据实际 LCD 规格对应用进行优化。

2： 用于手动调节对比度的电位器是可选的，可以不用。

17.3.4 LCD 电荷泵设计注意事项

如果应用设计中使用LCD稳压器并使能了电荷泵，用户必须始终同时考虑显示器的动态电流和 RMS （静态）电流要求，以及电荷泵可以提供的电流。动态和静态电流都可以通过公式 17-1 确定：

公式 17-1：

$$I = C \times \frac{dV}{dT}$$

对于动态电流，C 表示连接到 LCDBIAS3 和 LCDBIAS2 的电容值。变量 dV 为 LCD 显示器上电压切换过程中 C2 和 C3 上允许的压降； dT 为产生时钟脉冲后瞬态电流的持续时间。实际设计中，可以假定 C 为 0.047 µF， dV 为 0.1V， dT 为 1 µs。这将产生持续 1 µs 的 4.7 mA 的动态电流。

RMS 电流值取决于 CFLY 的值（公式中的 C）、VLCAP1 和 VLCAP2 之间的电压值（公式中的 dV），以及稳压器时钟周期 （TPER）（公式中的 dT）。假定 CFLY 值为 0.047 µF，CFLY 上的电压为 1.02V，TPER 值为 30 µs，则最大理论静态电流为 1.8 mA。由于电荷泵必须为 5 个电容充电，最大电流变为 360 µA。对于一个假定效率为 50% 的实际电路来说，产生的实际电流为 180 µA。

用户应将计算出的电流大小与 LCD 的需求相比较。 dV 和 dT 的值是由器件设计相对固定的，然而可以改变 CFLY 的值和 LCDBIAS 引脚上的电容值来增大或减小电流。应该始终在实际电路中评估任何改变对应用的影响。

17.4 LCD 复用类型

LCD 驱动模块可以被配置为 4 种复用类型：

- 静态 （仅使用 COM0）
- 1/2 复用 （使用 COM0 和 COM1）
- 1/3 复用 （使用 COM0、 COM1 和 COM2）
- 1/4 复用 （使用 COM0、COM1、COM2 和 COM3）

所使用的有效公共端数目由 LMUX<1:0> 位（LCDCON <1:0>）配置，这两个位决定 PORTE<6:4> 引脚的功能（详情请参见表 17-3）。如果引脚配置为 COM 驱动，则将禁止端口 I/O 功能，并改写该引脚的 TRIS 设置。

注：	上电复位时， LMUX<1:0> 位为 00。

表 17-3： PORTE<6:4> 功能

LMUX<1:0>	PORTE<6>	PORTE<5>	PORTE<4>
00	数字 I/O	数字 I/O	数字 I/O
01	数字 I/O	数字 I/O	COM1 驱动
10	数字 I/O	COM2 驱动	COM1 驱动
11	COM3 驱动	COM2 驱动	COM1 驱动

17.5 段使能

LCDSEx寄存器用于选择每个段引脚的引脚功能。置1某位将相应引脚配置为段驱动。 LCDSEx 寄存器不改写TRIS位设置，因此必须将该引脚的TRIS位配置为输入。

注：	上电复位时，这些引脚被配置为数字 I/O。

17.6 像素控制

LCDDATAx 寄存器中包含用于定义像素状态的位。每一位只定义一个像素。

表 17-2 给出了 LCDDATAx 寄存器中每一位与各个公共端和段信号间的相互关系。没有用于显示的LCD像素地址可用作通用 RAM。

17.7 LCD 帧频率

COM 和 SEG 输出改变的速率称为 LCD 帧频率。帧频率由 LP<3:0> 位（LCDPS<3:0>）进行设置，并受所使用的复用模式影响。复用模式、LP 位设置和帧速率之间的关系如表 17-4 和表 17-5 所示。

表 17-4： 帧频率计算公式

复用模式	帧频率 （Hz）
静态	时钟源 /(4 x 1 x (LP<3:0> + 1))
1/2	时钟源 /(2 x 2 x (LP<3:0> + 1))
1/3	时钟源 /(1 x 3 x (LP<3:0> + 1))
1/4	时钟源 /(1 x 4 x (LP<3:0> + 1))

表 17-5： LP 预分频器设置对应的近似帧频率 （单位：Hz）

LP<3:0>	复用模式			
	静态	1/2	1/3	1/4
1	125	125	167	125
2	83	83	111	83
3	62	62	83	62
4	50	50	67	50
5	42	42	56	42
6	36	36	48	36
7	31	31	42	31

17.8 LCD 波形产生

LCD 波形产生基于如下理论：不透明像素上的净交流电压应该为最大值而透明像素上的净交流电压应该为最小值。任何像素上的净直流电压应该为零。

COM 信号表示每个公共端的时间片，而 SEG 中包含像素数据。像素信号 （COM-SEG）中不包含直流分量，并且只可取两个 rms 值之一。高 rms 值会产生不透明像素而低 rms 值产生透明像素。

随着公共端数量的增加，两个 rms 值间的判别比逐渐减小。判别比表示显示器可具有的最大对比度。

可以用两种类型波形驱动 LCD：A 型和 B 型。在 A 型波形中，相位在每个公共端类型中改变；然而在 B 型波形中，相位在每个帧边界上改变。这样， A 型波形在单帧上维持 0 VDC 而 B 型波形则需要两个帧。

注 1： 如果要通过将LCD休眠位（SLPEN）置1（LCDCON<6> 为 1）启用功耗管理休眠模式，则必须格外小心，因为只有当所有像素上的 VDC 为 0 时才可执行休眠。

2： 当LCD时钟源为系统时钟时，不论SLPEN位设置如何，只要单片机进入休眠模式，LCD 模块就会进入休眠模式。因此，在启用休眠模式时，应始终注意查看所有像素上的 VDC 是否为 0。

图 17-6 至图 17-16 给出了 A 型和 B 型波形在静态、1/2 复用、 1/3 复用和 1/4 复用驱动时的波形图。

图 17-6：　　在静态驱动时的 A/B 型波形

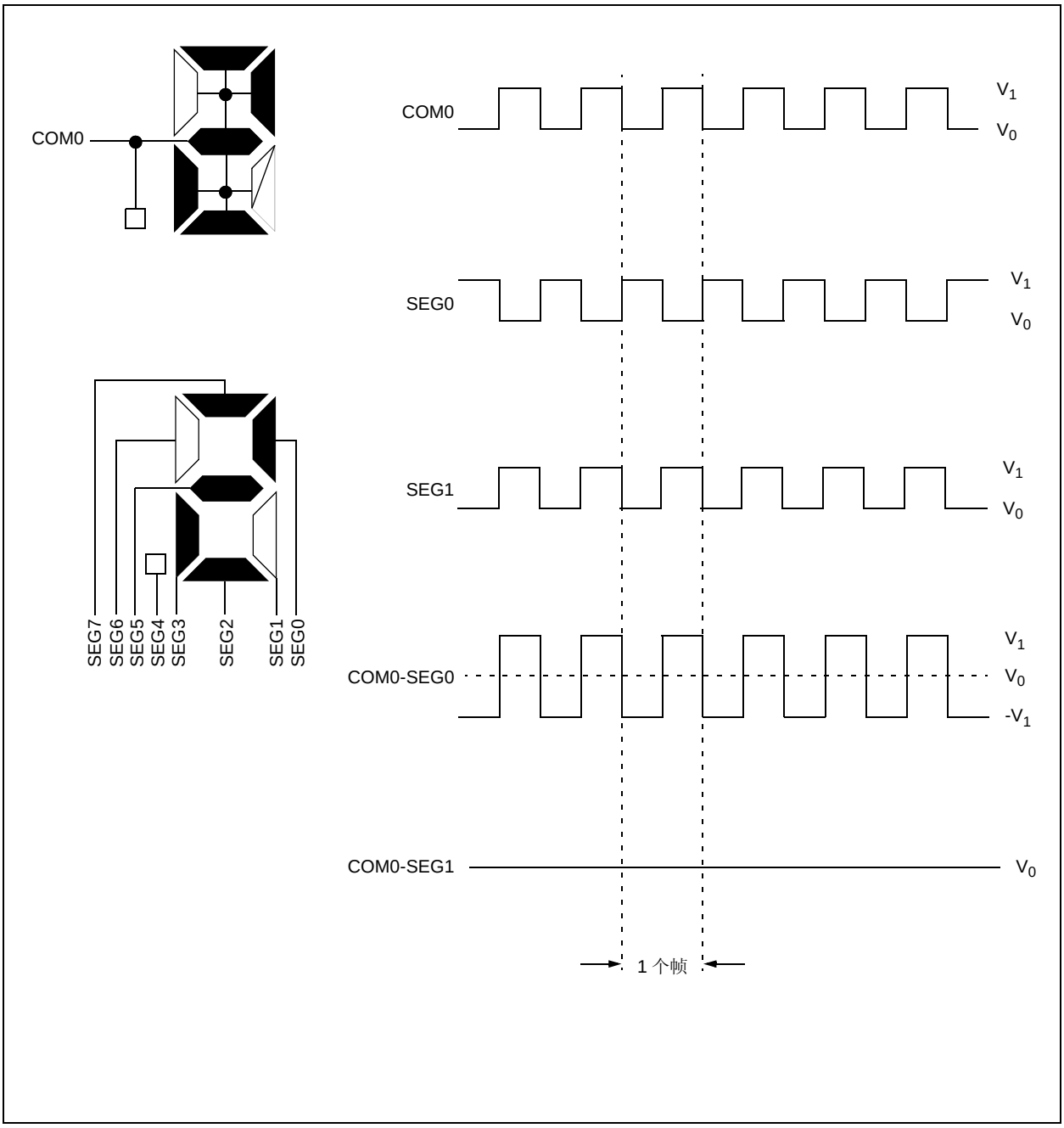


图 17-7：　　在 1/2 复用、 1/2 偏置驱动时的 A 型波形

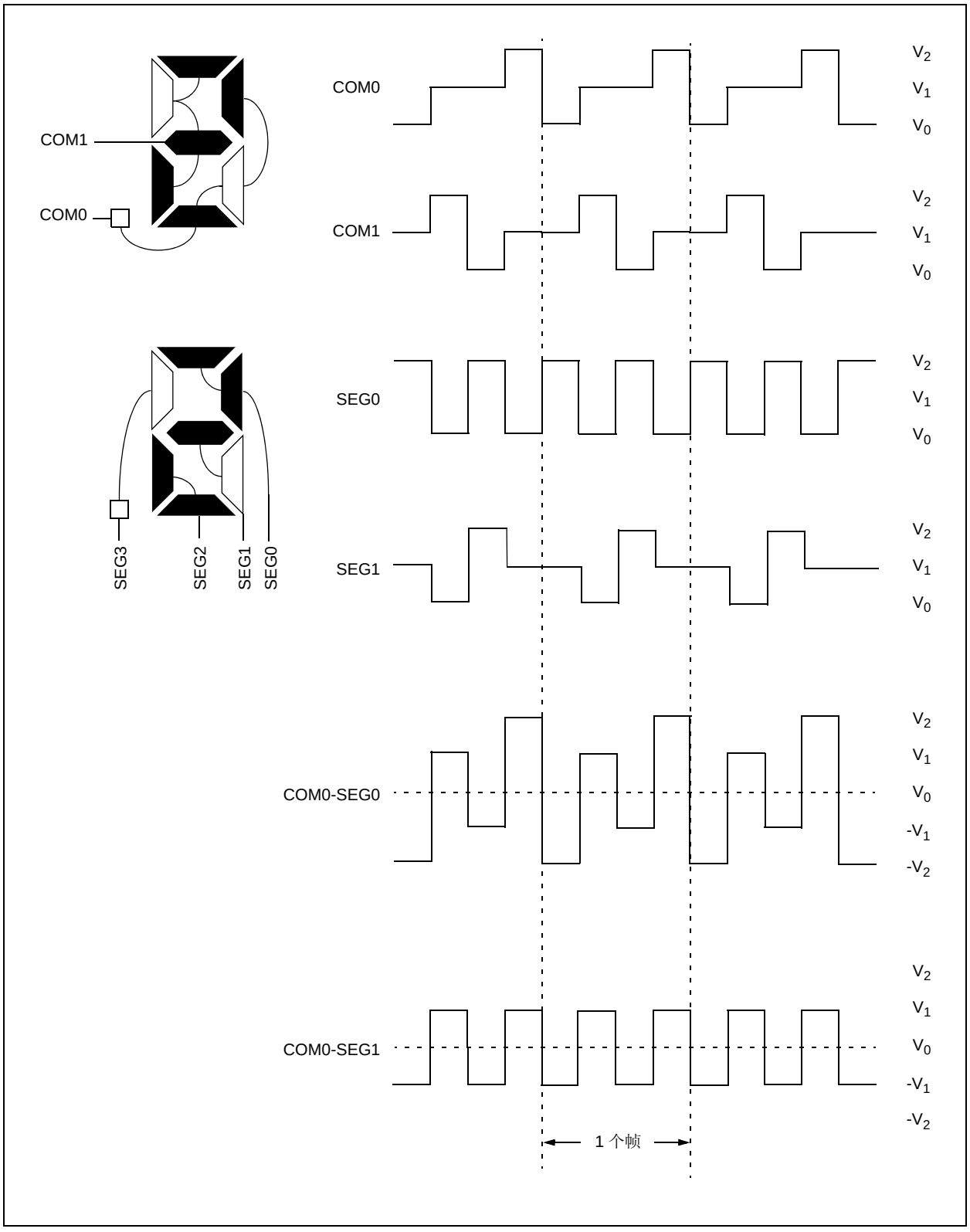


图 17-8：　　在 1/2 复用、 1/2 偏置驱动时的 B 型波形

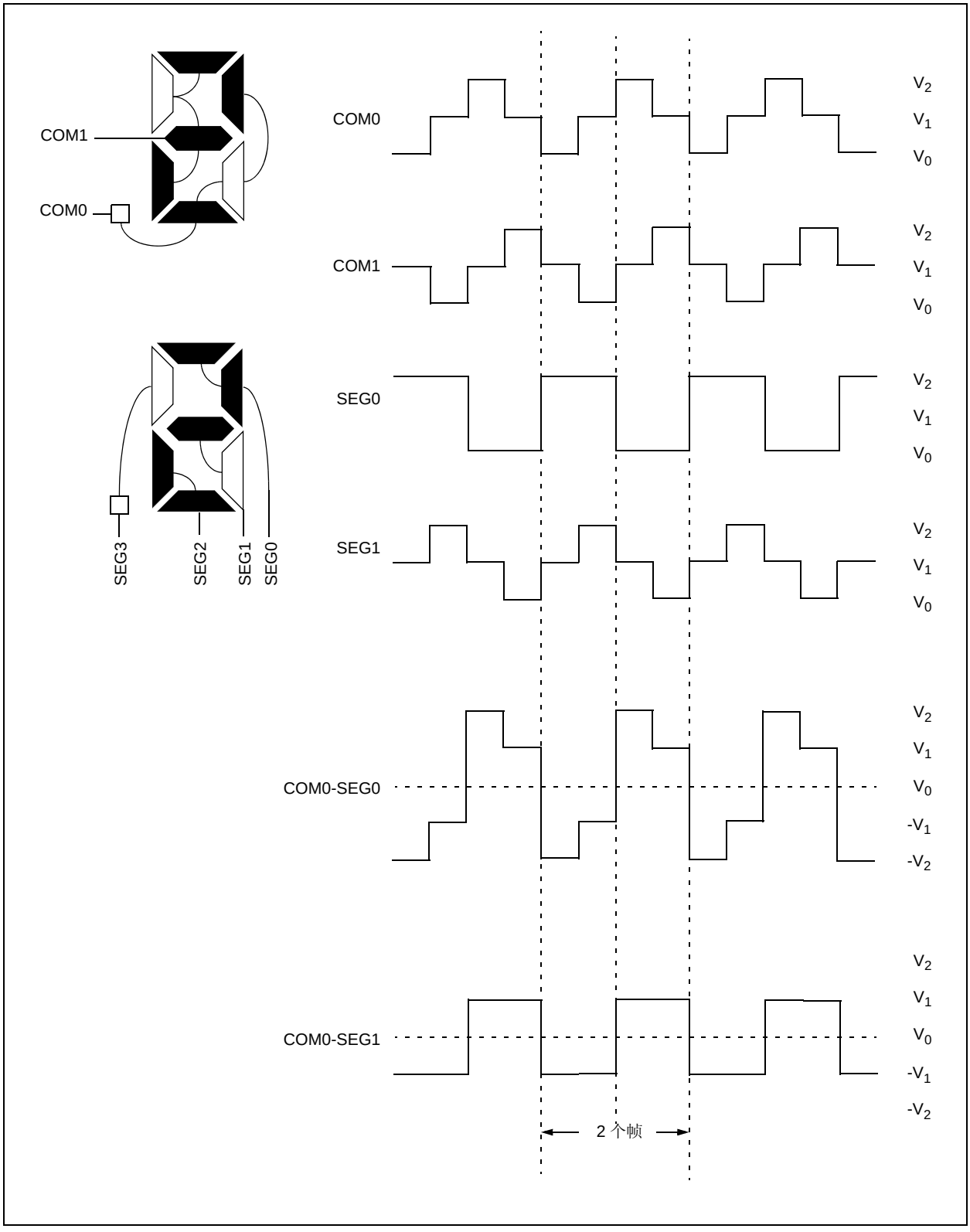


图 17-9：　　在 1/2 复用、 1/3 偏置驱动时的 A 型波形

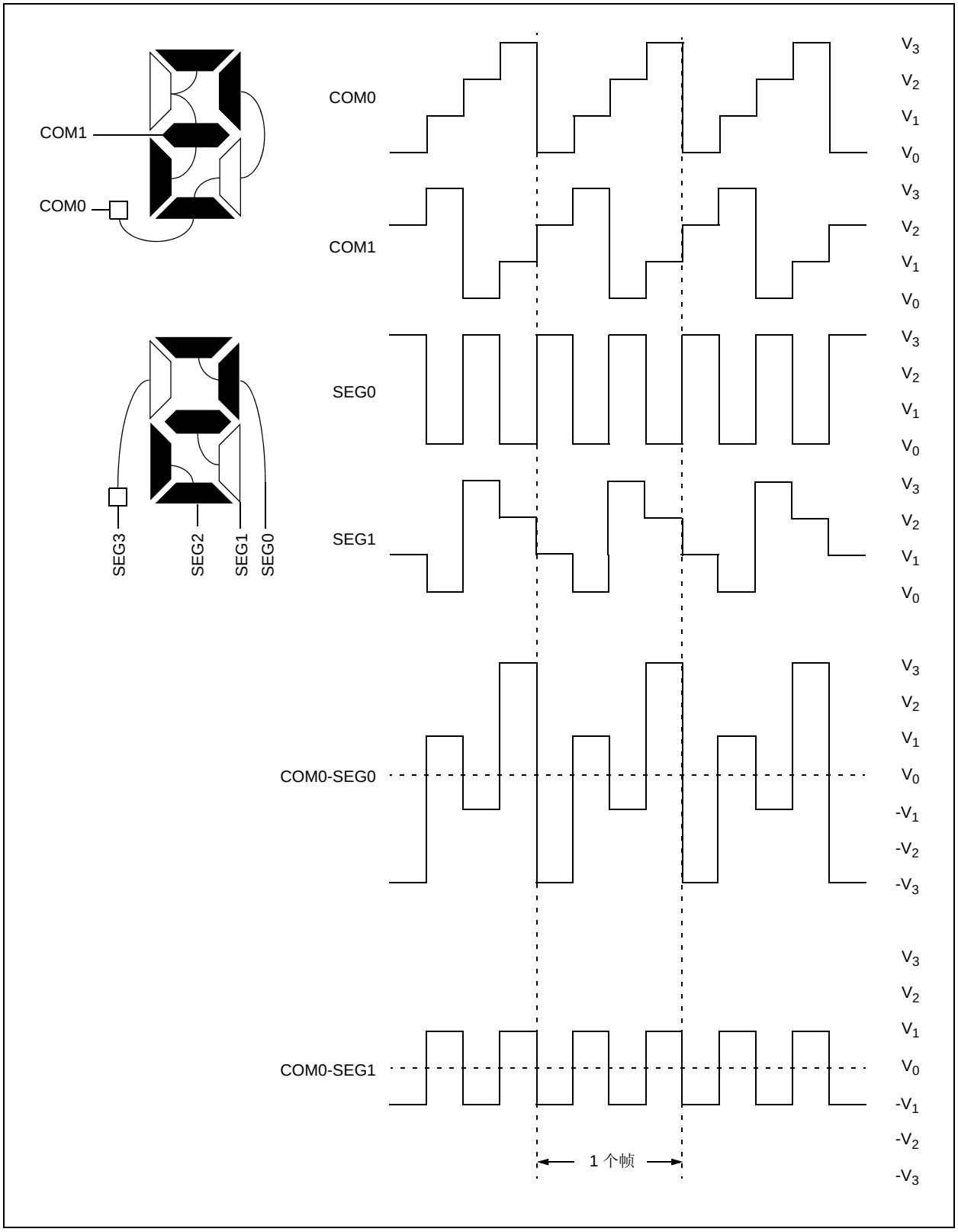


图 17-10：　　在 1/2 复用、 1/3 偏置驱动时的 B 型波形

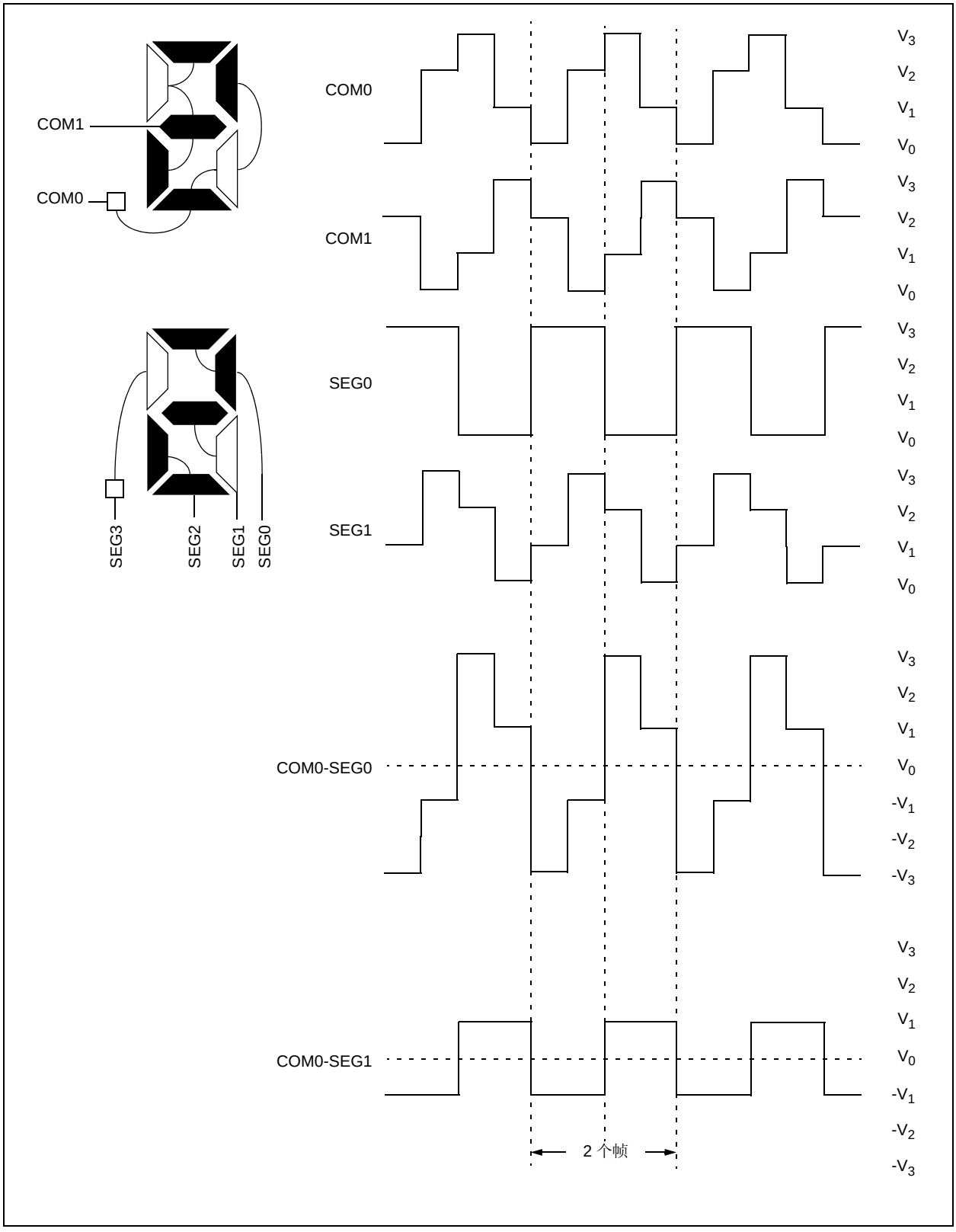


图 17-11：　　在 1/3 复用、 1/2 偏置驱动时的 A 型波形

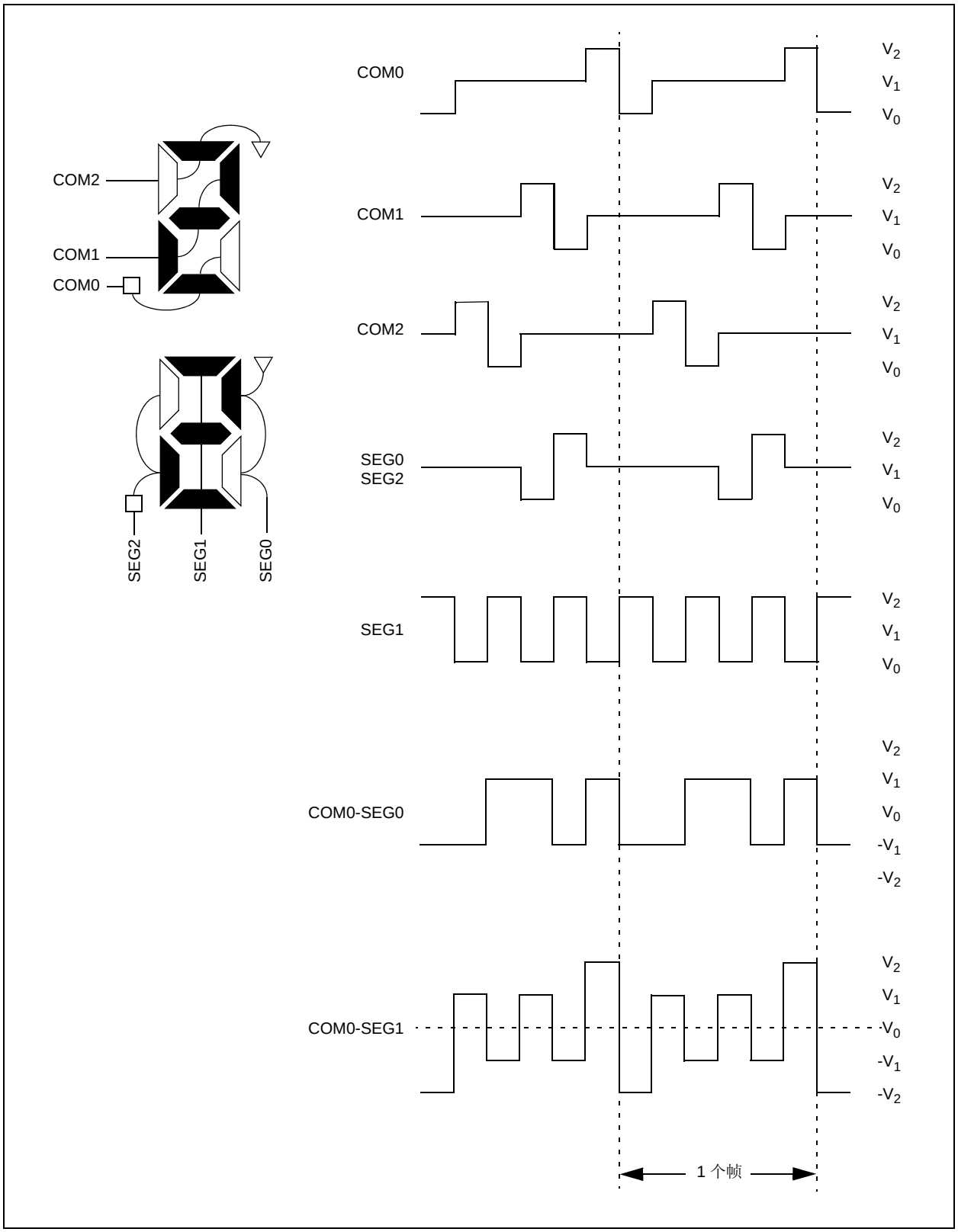


图 17-12：　　在 1/3 复用、 1/2 偏置驱动时的 B 型波形

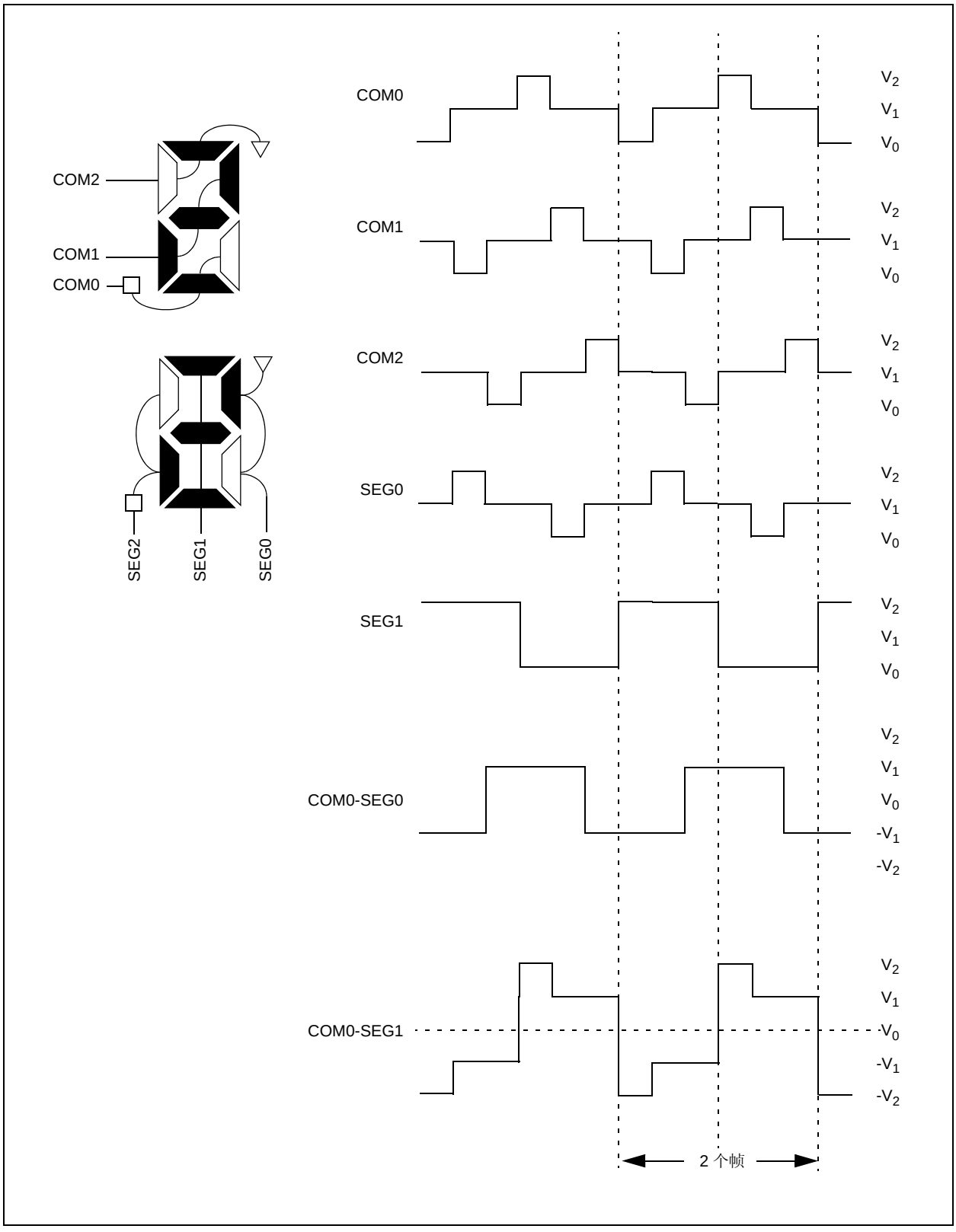


图 17-13：　　在 1/3 复用、 1/3 偏置驱动时的 A 型波形

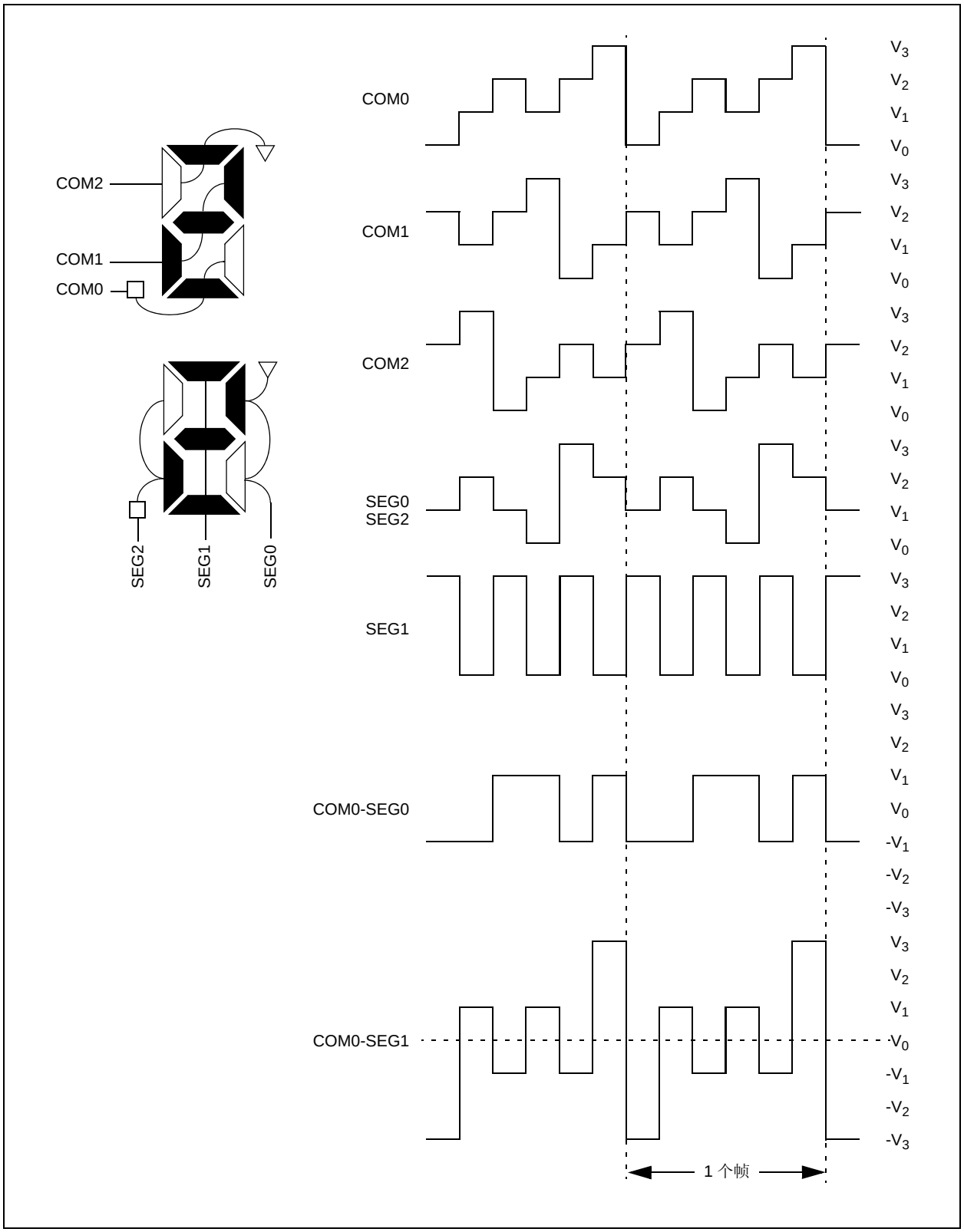


图 17-14：　　在 1/3 复用、 1/3 偏置驱动时的 B 型波形

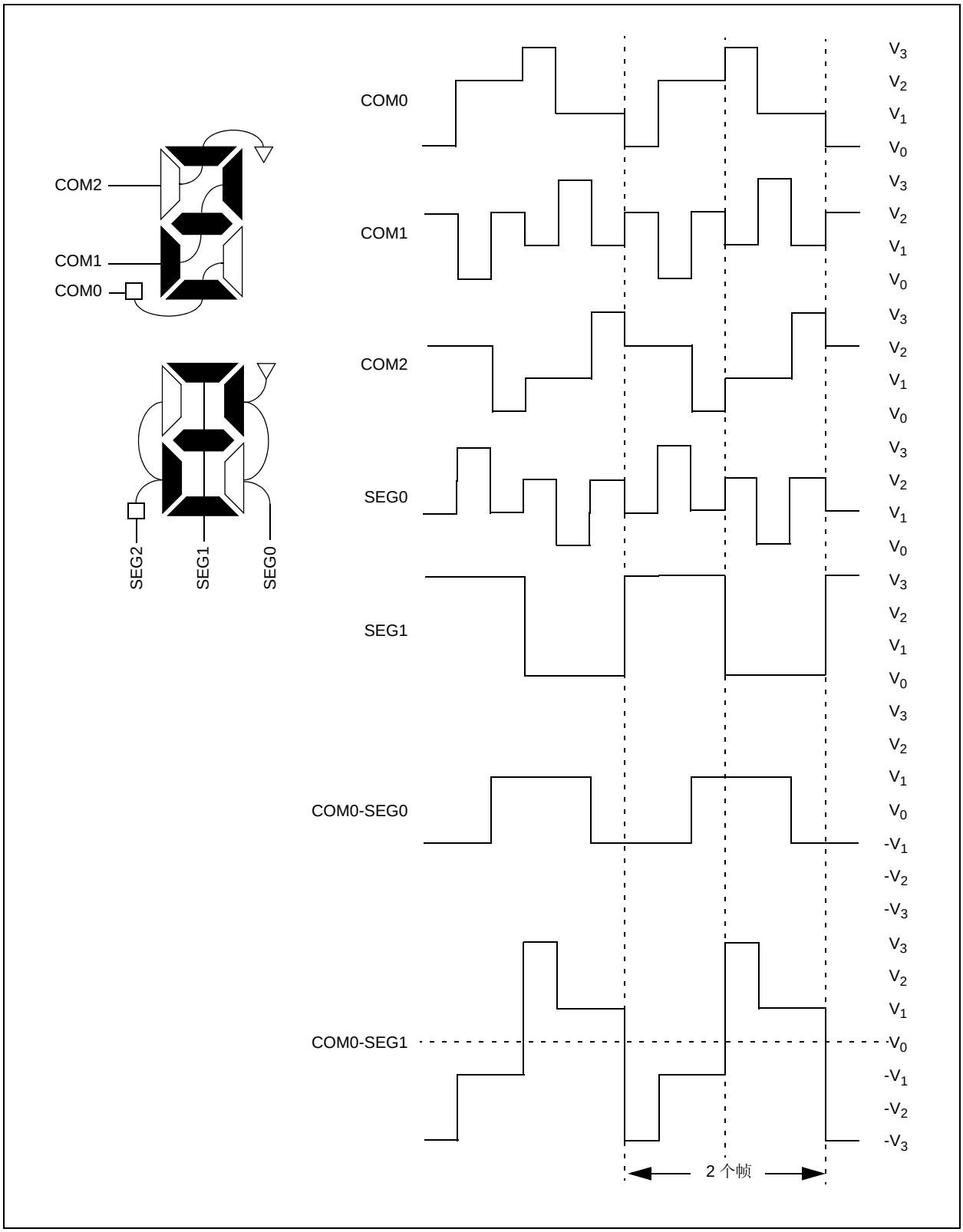


图 17-15：　　在 1/4 复用、 1/3 偏置驱动时的 A 型波形

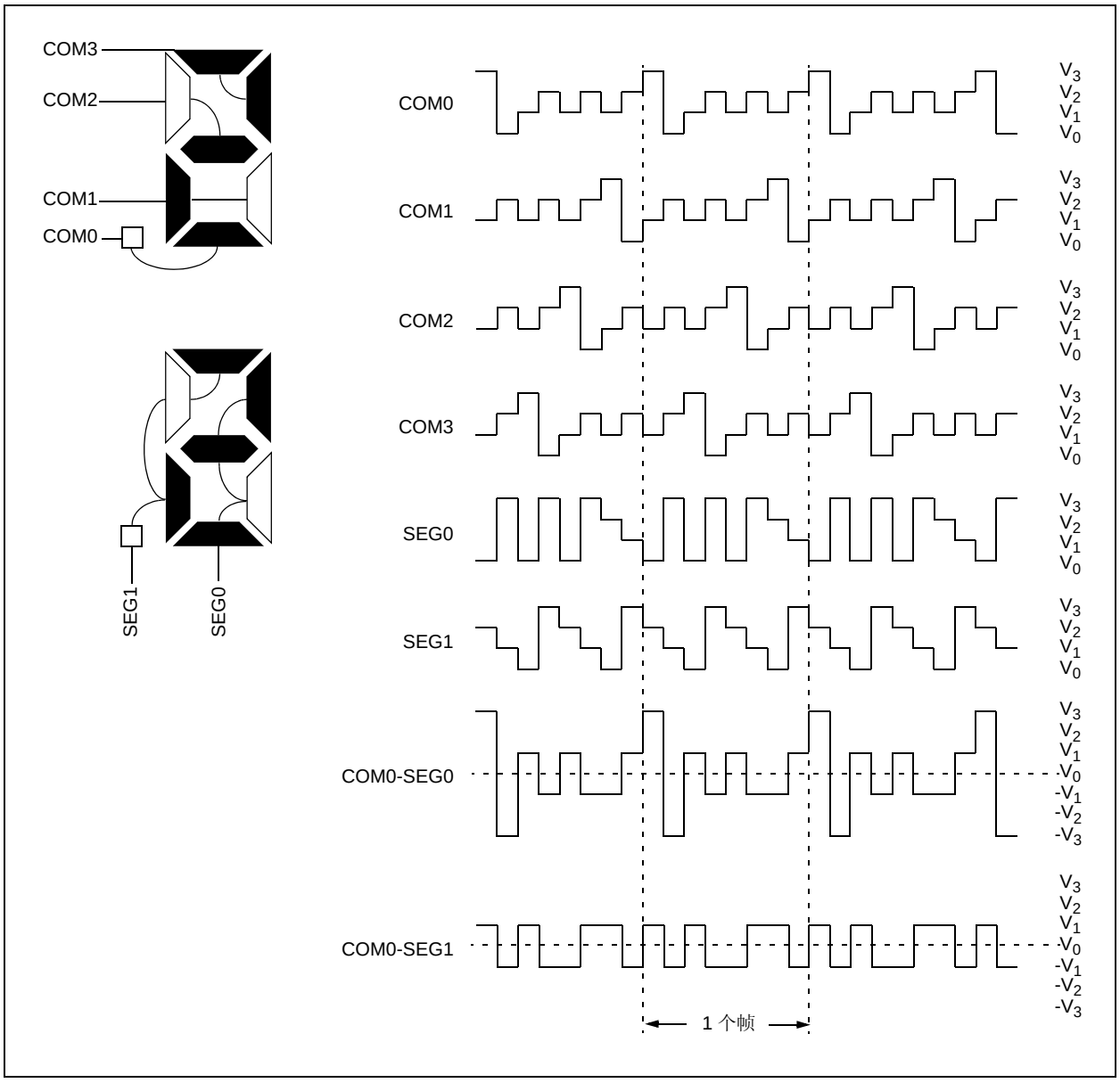
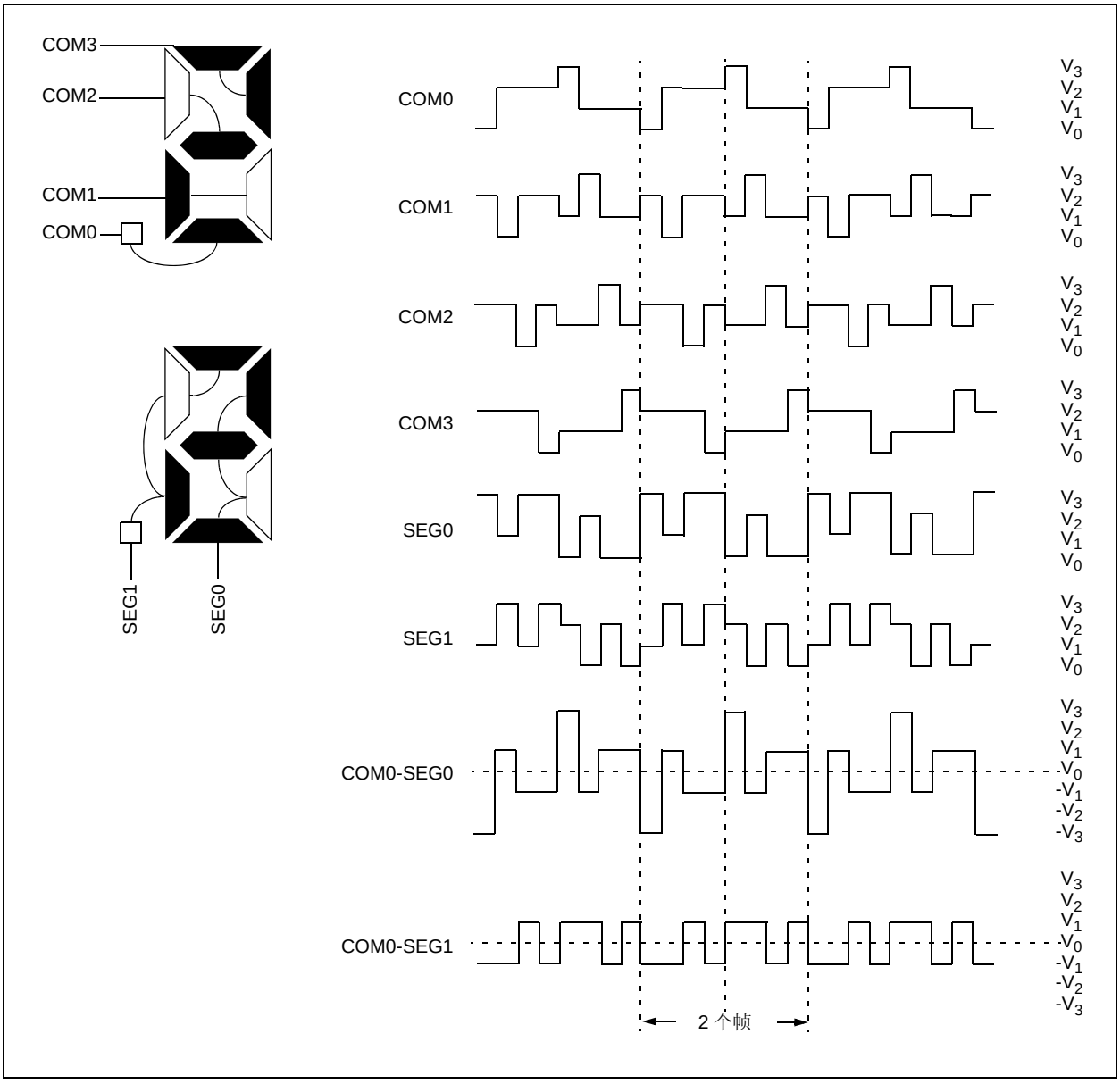


图 17-16：　　在 1/4 复用、 1/3 偏置驱动时的 B 型波形



17.9 LCD 中断

LCD 时序发生提供了一个中断，该中断用于定义 LCD 的帧时序。该中断用于协调在一个新帧开始时写入像素数据，在帧边界处写像素数据可使图像过渡更清晰。该中断还可用于同步 LCD 和外部事件。例如，与外部段驱动的接口，可与 LCD 帧的段数据更新同步。

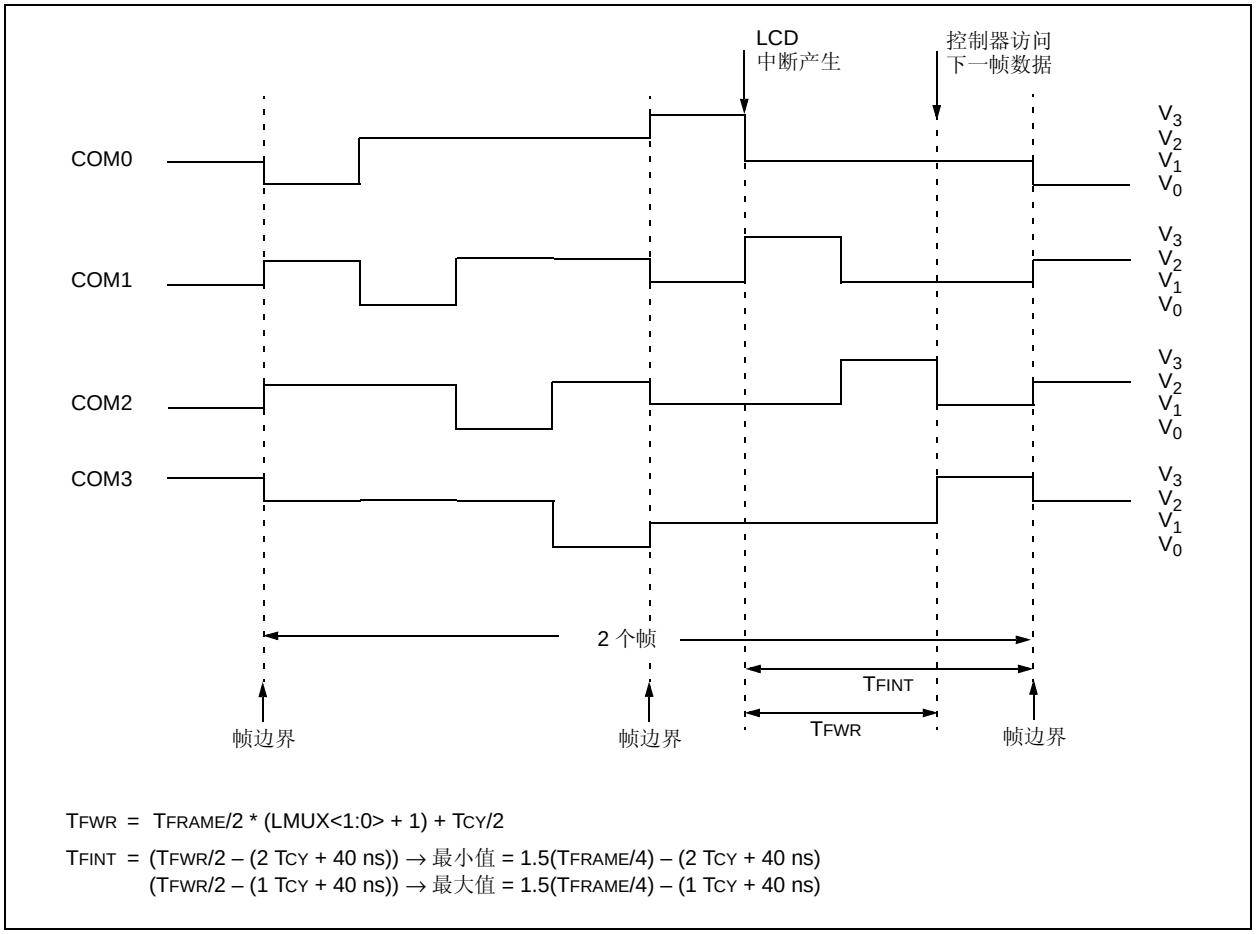
一个新帧定义为开始于 COM0 公共端信号的起始边界。在 LCD 控制器完成对帧所需的像素数据访问后，将立即产生中断。中断发生在帧边界（TFINT）前的某一固定时间，如图 17-17 所示。在中断发生的 TFWR 时间后，LCD 控制器将开始访问下一帧数据。新数据必须在 TFWR 内写入，因为之后 LCD 控制器将开始访问下一帧数据。

当 LCD 驱动器采用 B 型波形且 LMUX<1:0> 位不为 00 时，需要处理一些其他问题。由于需要用两帧来维持像素上的直流电压为零，因此在此期间像素数据要保持不变。一旦像素数据发生改变，奇数帧波形和偶数帧波形不再互补，在面板中会引入一个直流分量。因此，当使用 B 型波形时，用户必须同步 LCD 像素更新，使之发生在帧中断后的子帧中。

在 B 型波形时要保证正确的写入时序，中断只能发生在完整的相位间隔内。当禁止写入时，如果用户试图进行写操作， WERR 位 （LCDCON<5>）将被置 1。

注：	当选择 A 型波形和选择不带复用 （静态）的 B 型波形时，不会产生中断。

图 17-17： 1/4 占空比驱动时波形和中断时序示例



TFWR = TFRAME/2 * (LMUX<1:0> + 1) + TCY/2

TFINT = (TFWR/2 – (2 TCY + 40 ns)) → 最小值 = 1.5(TFRAME/4) – (2 TCY + 40 ns)
(TFWR/2 – (1 TCY + 40 ns)) → 最大值 = 1.5(TFRAME/4) – (1 TCY + 40 ns)

17.10 休眠期间的操作

LCD 模块可以在休眠模式下工作。模式选择由 SLPEN 位（LCDCON<6>）控制。将 SLPEN 位置 1 允许 LCD 模块进入休眠模式。清零 SLPEN 位允许模块在休眠模式下继续工作。

如果执行了 SLEEP 指令且 SLPEN = 1，则 LCD 模块将中止所有功能，进入极低电流消耗模式。模块将立即停止工作并在段和公共线上都输出最小LCD驱动电压，如图 17-18 所示。

为确保没有直流分量引入面板， SLEEP 指令应紧跟在 LCD 帧边界后执行。可用 LCD 中断判定帧边界。延时的计算请参见**第 17.9 节 “LCD 中断 ”** 中的公式。

如果执行了 SLEEP 指令且 SLPEN = 0，则模块将继续显示 LCDDATA 寄存器的当前内容。要使模块在休眠模式下继续工作，时钟源必须为 Timer1 振荡器或内部振荡器（INTRC 或 INTOSC 作为默认系统时钟）之一。在休眠模式下，LCD 数据不能改变。在此模式下，LCD 模块电流消耗并未降低，然而器件的整体消耗将因内核和其他外设功能的关闭而降低。

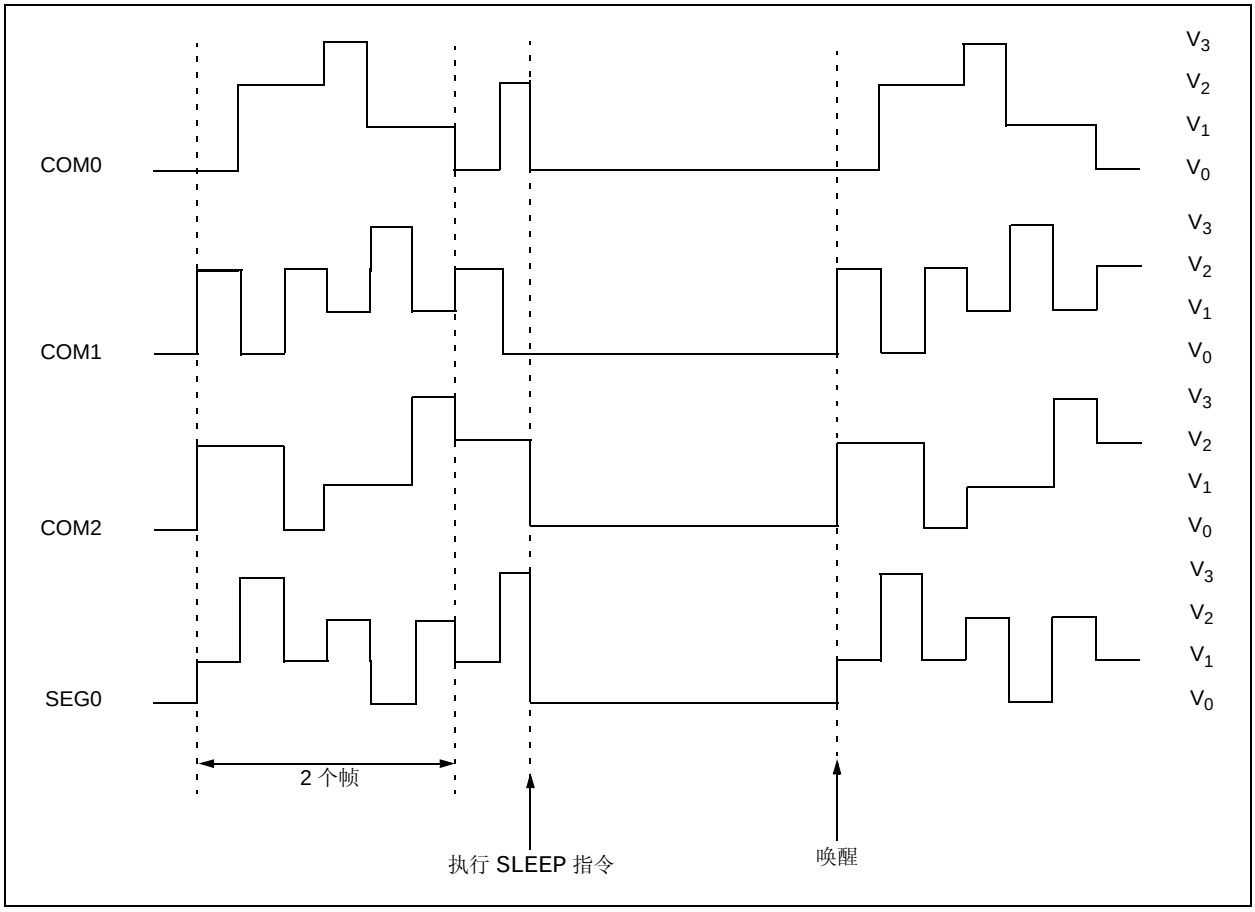
如果选择了系统时钟并将模块配置为非休眠模式，则模块将忽略 SLPEN 位并立即停止工作。段和公共端上输出的为最小 LCD 驱动电压。

17.10.1 在休眠模式下使用 LCD 稳压器

与使用模式 3（梯形电阻网络）偏置的应用相比，在休眠模式下，使用LCD稳压器来产生偏置的应用可能无法获得同等程度的功耗降低。在模式 0（其中电荷泵有效）操作中尤其如此。

如果使用模式 0、模式 1 或模式 2 来产生偏置，则软件对比度控制将不可用。

图 17-18： 当 SLPEN = 1 或 CS<1:0> = 00 时进入 / 退出休眠模式



17.11 配置 LCD 模块

以下是配置 LCD 模块的步骤：

1. 使用 LP<3:0> 位（LCDPS<3:0>）选择帧时钟预分频比。
2. 使用LCDSEx寄存器将相应的引脚配置为段驱动引脚。
3. 使用 TRISx 寄存器将相应的引脚配置为输入。
4. 使用 LCDCON 寄存器配置 LCD 模块：
   - 复用和偏置模式 （LMUX<1:0>）
   - 时钟源 （CS<1:0>）
   - 休眠模式 （SLPEN）
5. 将初始值写入像素数据寄存器 LCDDATA0 至 LCDDATA23。
6. 配置 LCD 稳压器：
   a) 如果要使用 M2 或 M3 偏置配置，通过将 CKSEL<1:0>（LCDREG<1:0>）设置为 00 来关闭稳压器。置 1 或清零 CPEN 位（LCDREG <6>）来相应选择模式 2 或模式 3。
   b) 如果要使用 M0 或 M1 产生偏置：
      - 使用 BIAS<2:0> 位 （LCDREG<5:3>）设置 V_{BIAS} 电平。
      - 置1或清零CPEN位来使能或禁止电荷泵。
      - 置 1 或清零 MODE13 位（LCDREG<2>）来选择偏置模式。
      - 使用 CKSEL<1:0> 位选择稳压器时钟源。
7. 清零 LCD 中断标志位 LCDIF （PIR3<6>），如有必要，通过将 LCDIE 位 （PIE3<6>）置 1 来允许中断。
8. 通过将 LCDEN 位 （LCDCON<7>）置 1，使能 LCD 模块。

表 17-6：　　与 LCD 操作相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR3	—	LCDIF	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	RTCCIF	62
PIE3	—	LCDIE	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	RTCCIE	62
IPR3	—	LCDIP	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	RTCCIP	62
RCON	IPEN	—	$\overline{CM}$	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	60
LCDDATA23(1)	S47C3	S46C3	S45C3	S44C3	S43C3	S42C3	S41C3	S40C3	63
LCDDATA22	S39C3(1)	S38C3(1)	S37C3(1)	S36C3(1)	S35C3(1)	S34C3(1)	S33C3(1)	S32C3	63
LCDDATA21	S31C3	S30C3	S29C3	S28C3	S27C3	S26C3	S25C3	S24C3	63
LCDDATA20	S23C3	S22C3	S21C3	S20C3	S19C3	S18C3	S17C3	S16C3	63
LCDDATA19	S15C3	S14C3	S13C3	S12C3	S11C3	S10C3	S09C3	S08C3	63
LCDDATA18	S07C3	S06C3	S05C3	S04C3	S03C3	S02C3	S01C3	S00C3	63
LCDDATA17(1)	S47C2	S46C2	S45C2	S44C2	S43C2	S42C2	S41C2	S40C2	63
LCDDATA16	S39C2(1)	S38C2(1)	S37C2(1)	S36C2(1)	S35C2(1)	S34C2(1)	S33C2(1)	S32C2	63
LCDDATA15	S31C2	S30C2	S29C2	S28C2	S27C2	S26C2	S25C2	S24C2	63
LCDDATA14	S23C2	S22C2	S21C2	S20C2	S19C2	S18C2	S17C2	S16C2	63
LCDDATA13	S15C2	S14C2	S13C2	S12C2	S11C2	S10C2	S09C2	S08C2	63
LCDDATA12	S07C2	S06C2	S05C2	S04C2	S03C2	S02C2	S01C2	S00C2	63
LCDDATA11(1)	S47C1	S46C1	S45C1	S44C1	S43C1	S42C1	S41C1	S40C1	63
LCDDATA10	S39C1(1)	S38C1(1)	S37C1(1)	S36C1(1)	S35C1(1)	S34C1(1)	S33C1(1)	S32C1	63
LCDDATA9	S31C1	S30C1	S29C1	S28C1	S27C1	S26C1	S25C1	S24C1	63
LCDDATA8	S23C1	S22C1	S21C1	S20C1	S19C1	S18C1	S17C1	S16C1	63
LCDDATA7	S15C1	S14C1	S13C1	S12C1	S11C1	S10C1	S09C1	S08C1	63
LCDDATA6	S07C1	S06C1	S05C1	S04C1	S03C1	S02C1	S01C1	S00C1	63
LCDDATA5(1)	S47C0	S46C0	S45C0	S44C0	S43C0	S42C0	S41C0	S40C0	63
LCDDATA4	S39C0(1)	S38C0(1)	S37C0(1)	S36C0(1)	S35C0(1)	S34C0(1)	S33C0(1)	S32C0	61
LCDDATA3	S31C0	S30C0	S29C0	S28C0	S27C0	S26C0	S25C0	S24C0	61
LCDDATA2	S23C0	S22C0	S21C0	S20C0	S19C0	S18C0	S17C0	S16C0	61
LCDDATA1	S15C0	S14C0	S13C0	S12C0	S11C0	S10C0	S09C0	S08C0	61
LCDDATA0	S07C0	S06C0	S05C0	S04C0	S03C0	S02C0	S01C0	S00C0	61
LCDSE5(1)	SE47	SE46	SE45	SE44	SE43	SE42	SE41	SE40	61
LCDSE4	SE39(1)	SE38(1)	SE37(1)	SE36(1)	SE35(1)	SE34(1)	SE33(1)	SE32	61
LCDSE3	SE31	SE30	SE29	SE28	SE27	SE26	SE25	SE24	61
LCDSE2	SE23	SE22	SE21	SE20	SE19	SE18	SE17	SE16	61
LCDSE1	SE15	SE14	SE13	SE12	SE11	SE10	SE09	SE08	61
LCDSE0	SE07	SE06	SE05	SE04	SE03	SE02	SE01	SE00	61
LCDCON	LCDEN	SLPEN	WERR	—	CS1	CS0	LMUX1	LMUX0	61
LCDPS	WFT	BIASMD	LCDA	WA	LP3	LP2	LP1	LP0	61
LCDREG	—	CPEN	BIAS2	BIAS1	BIAS0	MODE13	CKSEL1	CKSEL0	60

图注：　　— = 未实现，读为 0。 LCD 操作不使用阴影单元。

注　1：　这些寄存器或位在 PIC18F6XJ90 器件上未实现。

注：

18.0 主同步串行口（MSSP）模块

18.1 主 SSP（MSSP）模块概述

主同步串行口（MSSP）模块是用于同其他外设或单片机进行通信的串行接口。这些外设可以是串行 EEPROM、移位寄存器、显示驱动器和 A/D 转换器等。MSSP 模块有以下两种工作模式：

- 串行外设接口（Serial Peripheral Interface，SPI）
- I^2C™
  - 完全的主模式
  - 从模式（支持广播地址呼叫）

I^2C 接口硬件上支持以下模式：

- 主模式
- 多主模式
- 从模式

18.2 控制寄存器

每个 MSSP 模块都有三个相关的控制寄存器，包括一个状态寄存器（SSPSTAT）和两个控制寄存器（SSPCON1 和 SSPCON2）。根据 MSSP 模块是在 SPI 模式还是 I^2C 模式下工作，这些寄存器及其各自的位的使用将有很大不同。

下面各节会提供更多详细信息。

18.3 SPI 模式

SPI 模式允许同时同步发送和接收 8 位数据。器件支持 SPI 的所有四种模式。通常使用以下 3 个引脚来实现通信：

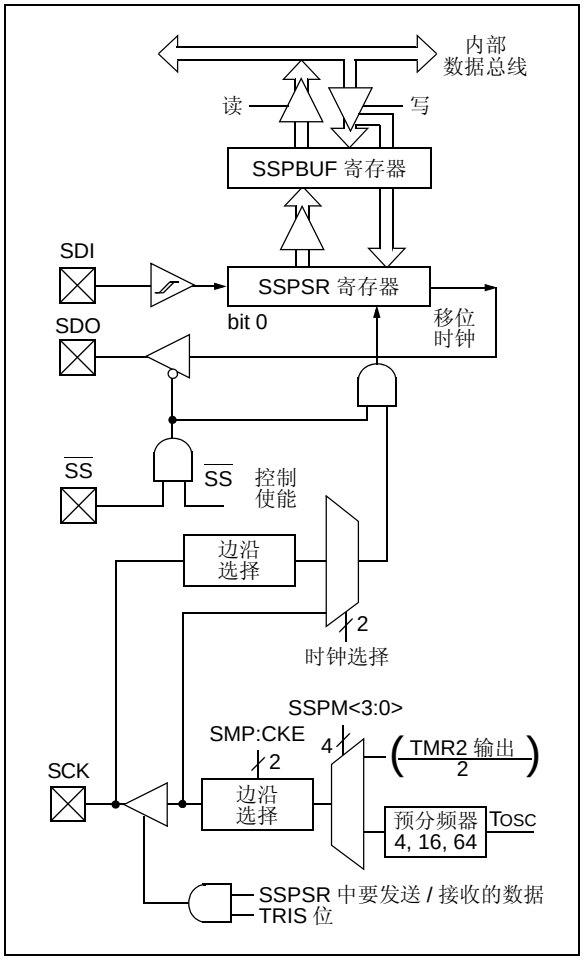
- 串行数据输出（Serial Data Out，SDO）——RC5/SDO/SEG12
- 串行数据输入（Serial Data In，SDI）——RC4/SDI/SDA/SEG16
- 串行时钟（Serial Clock，SCK）——RC3/SCK/SCL/SEG17

此外，当处于从工作模式时要使用第 4 个引脚：

- 从选择（Slave Select，$\overline{SS}$）——RF7/AN5/$\overline{SS}$/SEG25

图18-1给出了MSSP模块在SPI模式下的工作原理框图。

图 18-1：　　MSSP 框图（SPI 模式）



18.3.1 寄存器

MSSP 模块有四个寄存器用于 SPI 工作模式。这些寄存器包括：

- MSSP 控制寄存器 1 （SSPCON1）
- MSSP 状态寄存器 （SSPSTAT）
- 串行接收 / 发送缓冲寄存器 （SSPBUF）
- MSSP 移位寄存器（SSPSR） —— 不可直接访问

SSPCON1 和 SSPSTAT 是 SPI 模工作式下的控制寄存器和状态寄存器。 SSPCON1 寄存器是可读写的。SSPSTAT 的低 6 位是只读的，而高 2 位是可读写的。

SSPSR是用来将数据移入或移出的移位寄存器。SSPBUF是缓冲寄存器，可用于数据字节的写入或读出。

接收数据时，SSPSR 和 SSPBUF 共同构成一个双重缓冲接收器。当 SSPSR 接收到一个完整的字节之后，该字节会被送入SSPBUF，同时将中断标志位SSPIF置1。

在数据发送过程中， SSPBUF 不是双重缓冲的，对 SSPBUF 的写操作将同时写入 SSPBUF 和 SSPSR。

寄存器 18-1： SSPSTAT：MSSP 状态寄存器 （SPI 模式）

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R0	R-0
SMP	CKE(1)	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
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 **SMP**：采样位
SPI 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
SPI 从模式：
当 SPI 工作在从模式时，必须将 SMP 清零。

bit 6 **CKE**：SPI 时钟选择位 (1)
1 = 时钟状态从有效转换到空闲时发送
0 = 时钟状态从空闲转换到有效时发送

bit 5 **D/$\overline{A}$**：数据 / 地址位
仅在 I^2C™ 模式下使用。

bit 4 **P**：停止位
仅在 I^2C 模式下使用。当禁止 MSSP 模块 （SSPEN 清零）时，该位被清零。

bit 3 **S**：启动位
仅在 I^2C 模式下使用。

bit 2 **R/$\overline{W}$**：读 / $\overline{写}$信息位
仅在 I^2C 模式下使用。

bit 1 **UA**：更新地址位
仅在 I^2C 模式下使用。

bit 0 **BF**：缓冲区满状态位 （仅用于接收模式）
1 = 接收完成， SSPBUF 已满
0 = 接收未完成， SSPBUF 为空

注 1： 时钟状态的极性由 CKP 位 （SSPCON1<4>）设置。

寄存器 18-2： SSPCON1：MSSP 控制寄存器 1 （SPI 模式）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV(1)	SSPEN(2)	CKP	SSPM3(3)	SSPM2(3)	SSPM1(3)	SSPM0(3)
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 **WCOL**：写冲突检测位 （仅用于发送模式）
1 = 正在发送前一个字时，又有数据写入 SSPBUF 寄存器 （必须用软件清零）
0 = 未发生冲突

bit 6 **SSPOV**：接收溢出指示位 (1)
SPI 从模式：
1 = SSPBUF 寄存器中仍保存前一数据时，又接收到一个新的字节。如果发生溢出， SSPSR 中的数据会丢失。溢出只会在从模式下发生。即使只是发送数据，用户也必须读 SSPBUF，以避免将溢出标志位置 1 （该位必须用软件清零）。
0 = 无溢出

bit 5 **SSPEN**：主同步串口使能位 (2)
1 = 使能串口并将 SCK、 SDO、 SDI 和 $\overline{SS}$ 配置为串口引脚
0 = 禁止串口并将上述引脚配置为 I/O 端口引脚

bit 4 **CKP**：时钟极性选择位
1 = 空闲状态时，时钟为高电平
0 = 空闲状态时，时钟为低电平

bit 3-0 **SSPM<3:0>**：主同步串口模式选择位 (3)
0101 = SPI 从模式，时钟 = SCK 引脚，禁止 $\overline{SS}$ 引脚控制，可将 $\overline{SS}$ 用作 I/O 引脚
0100 = SPI 从模式，时钟 = SCK 引脚，使能 $\overline{SS}$ 引脚控制
0011 = SPI 主模式，时钟 = TMR2 输出 /2
0010 = SPI 主模式，时钟 = Fosc/64
0001 = SPI 主模式，时钟 = Fosc/16
0000 = SPI 主模式，时钟 = Fosc/4

注 1： 在主模式下，溢出位不会被置 1，因为每次接收 （和发送）新数据都是通过写入 SSPBUF 寄存器启动的。
2： 当使能时，必须将这些引脚正确地配置为输入或输出。
3： 在此未列出的位组合被保留或仅在 I^2C™ 模式下使用。

18.3.2 工作原理

初始化 SPI 时需要指定几个选项。可以通过编程相应的控制位 （SSPCON1<5:0> 和 SSPSTAT<7:6>）来指定这些选项。这些控制位用于指定以下选项：

- 主模式 （SCK 作为时钟输出）
- 从模式 （SCK 作为时钟输入）
- 时钟极性 （SCK 的空闲状态）
- 数据输入采样阶段 （数据输出时间的中间或末尾）
- 时钟边沿 （在 SCK 的上升沿 / 下降沿输出数据）
- 时钟速率 （仅用于主模式）
- 从选择模式 （仅用于从模式）

每个MSSP模块由一个发送/接收移位寄存器（SSPSR）和一个缓冲寄存器 （SSPBUF）组成。 SSPSR 将数据移入 / 移出器件，先移位 MSb。在新数据接收完毕前，SSPBUF 保存上次写入 SSPSR 的数据。一旦 8 位数据接收完毕，该字节就被移入 SSPBUF 寄存器。然后，缓冲区满检测位BF（SSPSTAT<0>）和中断标志位SSPIF被置 1。这种双重缓冲数据接收方式 （SSPBUF），允许在 CPU 读取刚接收的数据之前，就开始接收下一个字节。在数据发送 / 接收期间，任何试图写 SSPBUF 寄存器的操作都将被忽略，并且写冲突检测位 WCOL（SSPCON1<7>）将被置1。用户必须用软件将WCOL位清零才能判断以后对 SSPBUF 寄存器的写入是否成功。

为确保应用软件能有效地接收数据，在下一个要发送的数据字节写入 SSPBUF 之前，读取 SSPBUF 中现有的数据。缓冲区满位 BF （SSPSTAT<0>）用于表示何时SSPBUF装入了接收到的数据（发送完成）。当SSPBUF中的数据被读取后，BF 位即被清零。如果 SPI 仅作为一个发送器，则不必理会该数据。通常，可用 MSSP 中断来判断发送 / 接收是否已完成。必须读取和 / 或写入SSPBUF。如果不打算使用中断，用软件查询的方法同样可确保不会发生写冲突。例 18-1 举例说明了装载 SSPBUF （SSPSR）进行数据发送的过程。

不能直接读写 SSPSR 寄存器，只能通过寻址 SSPBUF 寄存器来访问。此外， SSPSTAT 寄存器用于指示各种状态条件。

例 18-1： 装载 SSPBUF （SSPSR）寄存器

```
LOOP    BTFSS   SSPSTAT, BF     ;Has data been received (transmit complete)?
        BRA     LOOP            ;No
        MOVF    SSPBUF, W       ;WREG reg = contents of SSPBUF
        MOVWF   RXDATA          ;Save in user RAM, if data is meaningful
        MOVF    TXDATA, W       ;W reg = contents of TXDATA
        MOVWF   SSPBUF          ;New data to xmit
```

18.3.3 使能 SPI I/O

要使能串口，MSSP 使能位 SSPEN（SSPCON1<5>）必须置 1。要复位或重新配置 SPI 模式，要先将 SSPEN 位清零，重新初始化 SSPCON 寄存器，然后将 SSPEN 位置 1。这将把 SDI、SDO、SCK 和 $\overline{SS}$ 引脚配置为串口引脚。要将上述引脚用于串口功能，必须正确设置其中一些引脚的数据方向位（在 TRIS 寄存器中）：

- SDI 由 SPI 模块自动控制
- SDO 必须将 TRISC<5> 位清零
- SCK（主模式）必须将 TRISC<3> 位清零
- SCK（从模式）必须将 TRISC<3> 位置 1
- $\overline{SS}$ 必须将 TRISF<7> 位置 1

对于不需要的串口功能，可通过将对应的数据方向寄存器（TRIS）设置为相反值来改写。

18.3.4 漏极开路输出选项

SDO 输出和 SCK 时钟引脚的驱动器可以有选择地配置为漏极开路输出。此功能使引脚上的电平可通过外部上拉电阻上拉至较高的电平，并且无需额外的电平转换器就可使输出与外部电路进行通信。

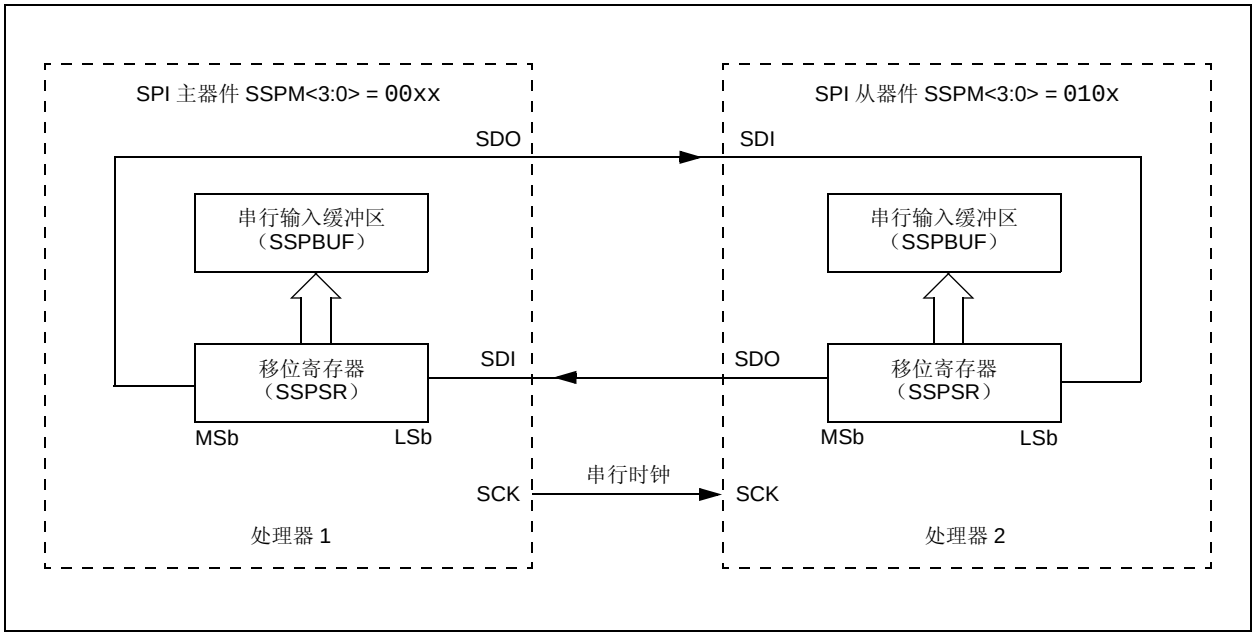
漏极开路输出选项由 SPIOD 位（TRISG<7>）控制。通过将该位置 1 可将两个引脚配置为漏极开路操作。

18.3.5 典型连接

图 18-2 给出了两个单片机之间的典型连接。主器件（处理器 1）通过发送 SCK 信号来启动数据传输。数据在编程设定的时钟边沿被从两个移位寄存器移出，并在相反的时钟边沿锁存。必须将两个处理器的时钟极性（CKP）设置为相同，这样就可以同时收发数据。数据是否有意义（或无效数据），取决于应用程序软件。这就导致以下三种数据传输情形：

- 主器件发送数据 —— 从器件发送无效（Dummy）数据
- 主器件发送数据 —— 从器件发送数据
- 主器件发送无效数据 —— 从器件发送数据

图 18-2： SPI 主 / 从器件连接



18.3.6 主模式

因为由主器件控制 SCK 信号，所以它可以在任意时刻启动数据传输。主器件根据软件协议确定从器件（图 18-2 中的处理器 2）在何时广播数据。

在主模式下，一写入 SSPBUF 寄存器就发送或接收数据。如果只打算将 SPI 作为接收器，则可以禁止 SDO 输出（将其编程设置为输入）。SSPSR 寄存器按设置的时钟速率，对 SDI 引脚上的信号连续移入。每接收到一个字节，就将其装入 SSPBUF 寄存器，就像接收到普通字节一样（中断和状态位相应置 1）。这在以“线路活动监控”（Line Activity Monitor）方式工作的接收器应用中很有用。

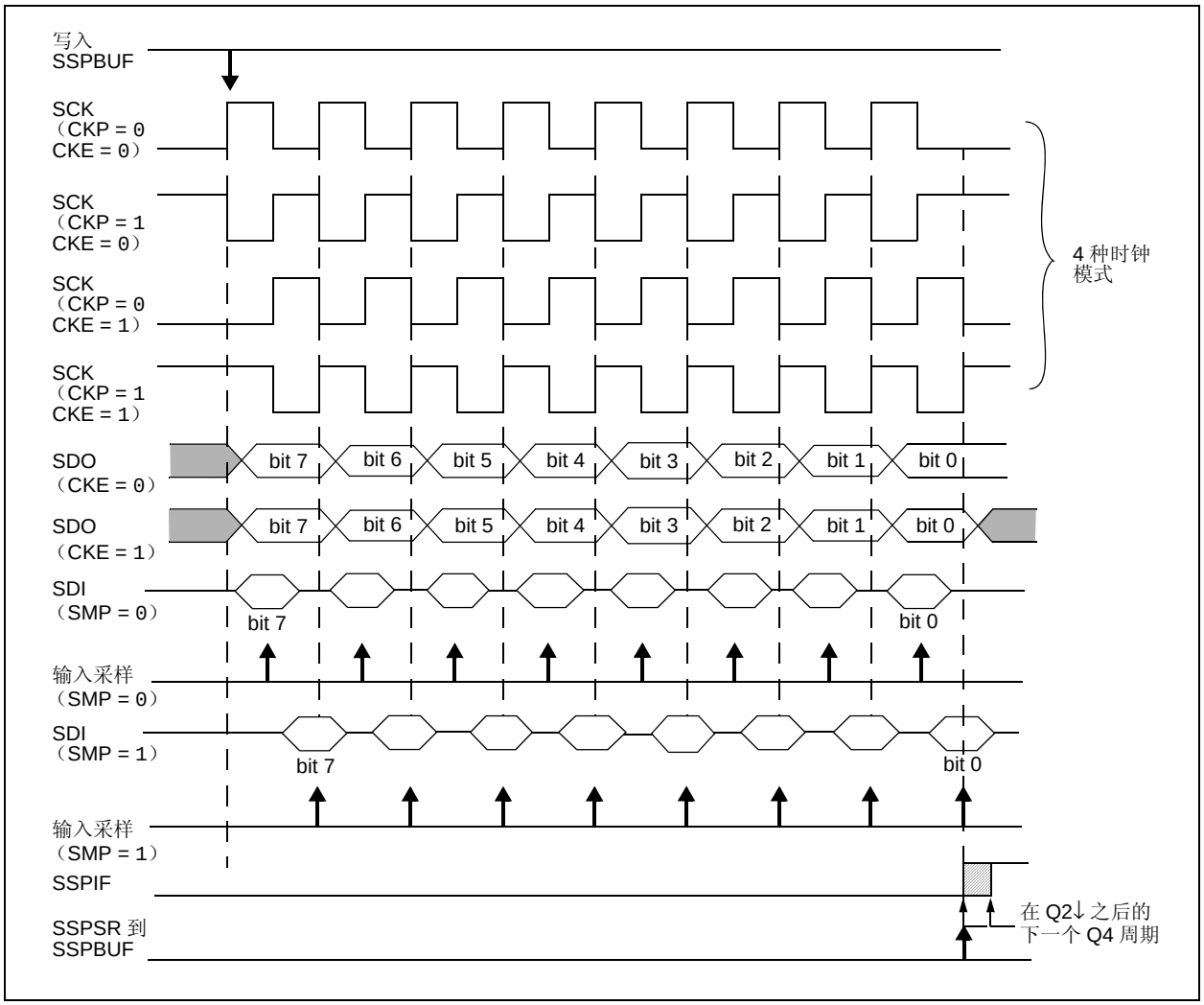
可通过对 CKP 位（SSPCON1<4>）进行适当的编程来选择时钟极性。图 18-3、图 18-5 和图 18-6 将给出 SPI 通信的波形图，其中 MSB 先发送。在主模式下，SPI 时钟速率（比特率）可由用户编程设定为以下几种之一：

- FOSC/4（或 TCY）
- FOSC/16（或 4 • TCY）
- FOSC/64（或 16 • TCY）
- Timer2 输出 /2

这样可使数据速率最高达到 10.00 Mbps（时钟频率为 40 MHz）。

图 18-3 给出了主模式的波形图。当 CKE 位置 1 时，SDO 数据在 SCK 上出现时钟边沿前一直有效。图中所示的输入采样的变化由 SMP 位的状态反映。图中给出了将接收到的数据装入 SSPBUF 的时刻。

图 18-3： SPI 模式波形图（主模式）



18.3.7 从模式

在从模式下，当 SCK 引脚上出现外部时钟脉冲时发送和接收数据。当最后一位数据被锁存后，中断标志位 SSPIF 置 1。

在 SPI 从模式下使能该模块前，时钟线必须处于相应的空闲状态。时钟线可通过读 SCK 引脚来查看。空闲状态由 CKP 位（SSPCON1<4>）决定。

在从模式下，外部时钟由 SCK 引脚上的外部时钟源提供。外部时钟必须满足电气规范中规定的高电平和低电平的最短时间要求。

在休眠模式下，从器件仍可发送 / 接收数据。当接收到一个字节时，器件从休眠状态唤醒。

18.3.8 从选择同步

$\overline{SS}$ 引脚允许器件工作于同步从模式。SPI 必须处于从模式，并使能 $\overline{SS}$ 引脚控制（SSPCON1<3:0> = 04h）。当 $\overline{SS}$ 引脚为低电平时，使能数据的发送和接收，同时 SDO 引脚被驱动。当 $\overline{SS}$ 引脚变为高电平时，即使是在字节的发送过程中，也不再驱动 SDO 引脚，而是变成悬空输出状态。可根据应用需要，在 SDO 引脚上外接上拉 / 下拉电阻。

注 1： 当 SPI 处于从模式且 $\overline{SS}$ 引脚控制使能（SSPCON1<3:0> = 0100）时，如果 $\overline{SS}$ 引脚设置为 VDD， SPI 模块将会复位。

2： 如果 SPI 工作在从模式并且 CKE 置 1，则必须使能 $\overline{SS}$ 引脚控制。

当 SPI 模块复位后，位计数器被强制为 0。这是通过强制将 $\overline{SS}$ 引脚拉为高电平或将 SSPEN 位清零来实现的。

可将 SDO 引脚和 SDI 引脚相连，来仿真二线制通信。当 SPI 需要作为接收器工作时， SDO 引脚可被配置为输入端。这样就禁止了从 SDO 发送数据。因为 SDI 不会引起总线冲突，所以可以一直将其保留为输入（SDI 功能）。

图 18-4： 从同步波形图

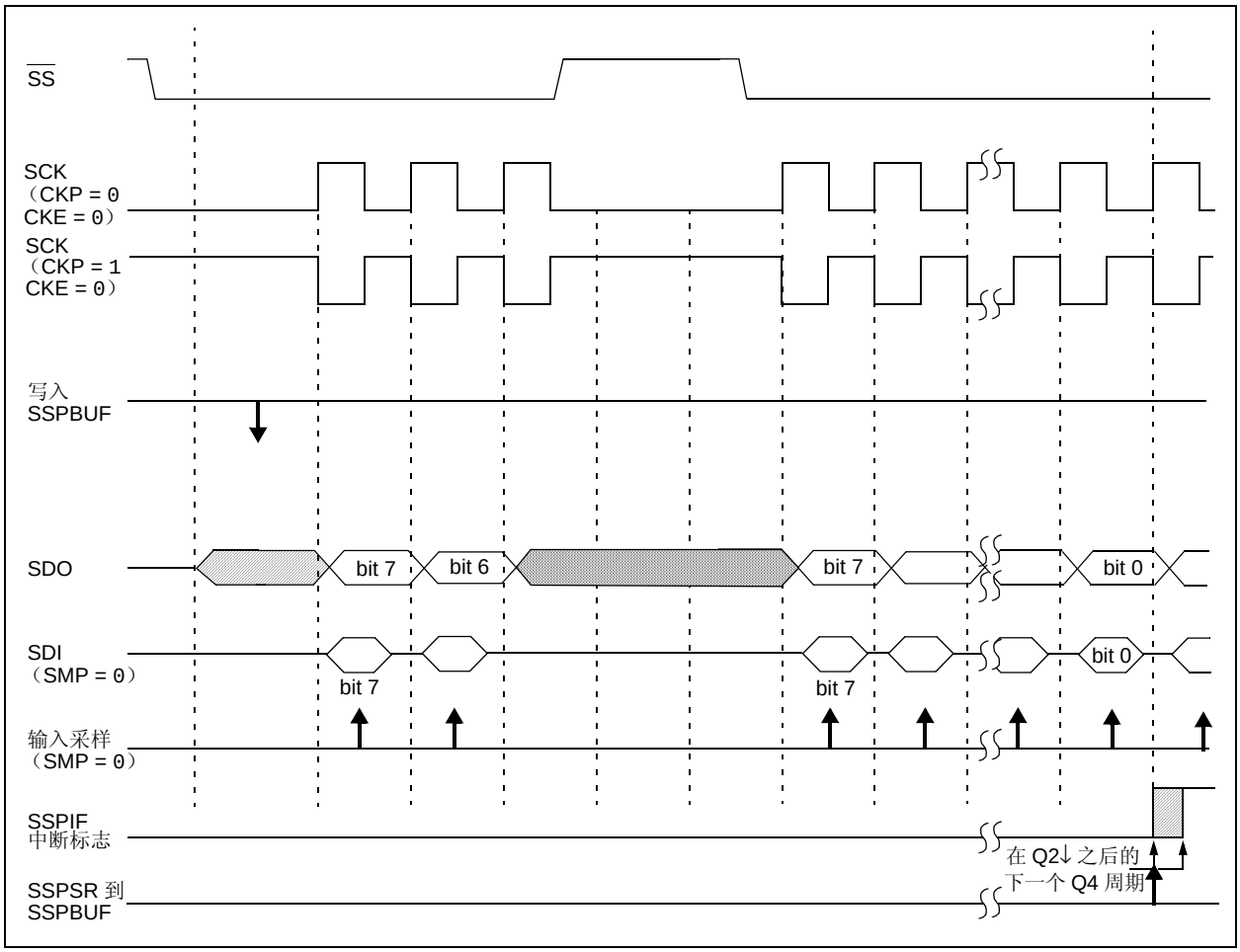


图 18-5：　　SPI 模式波形图 （从模式， CKE = 0）

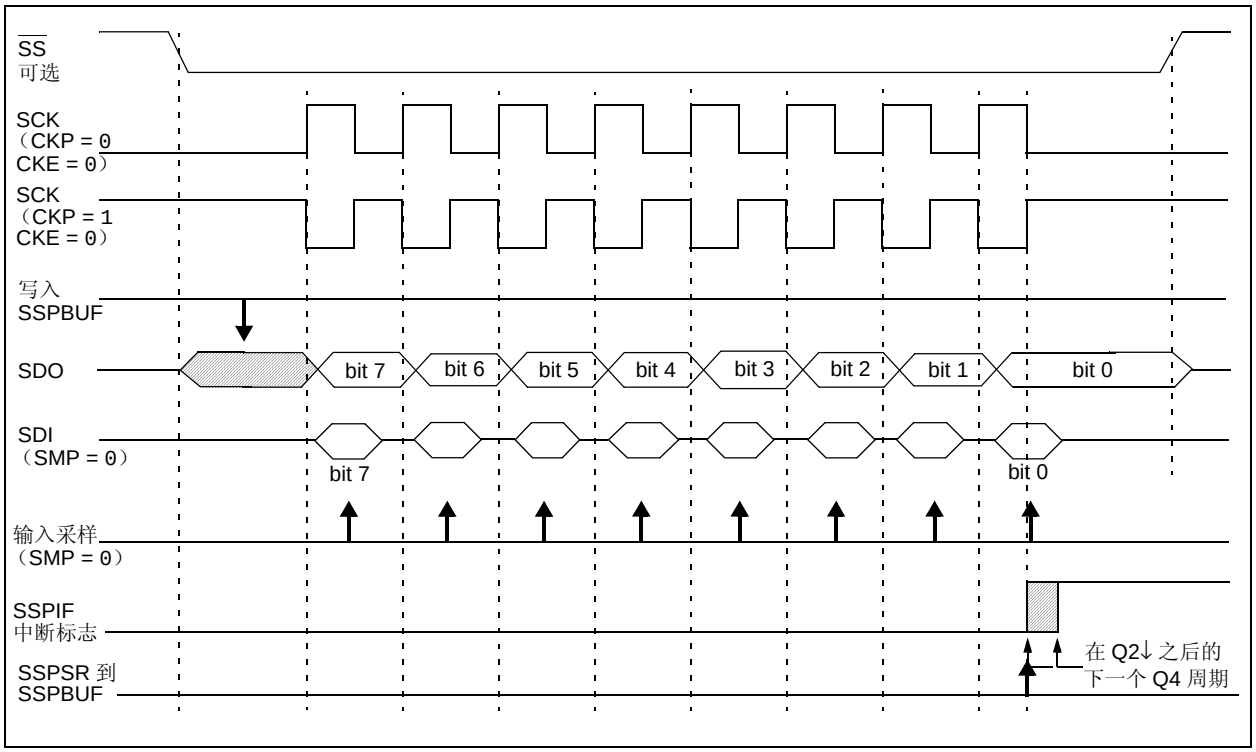
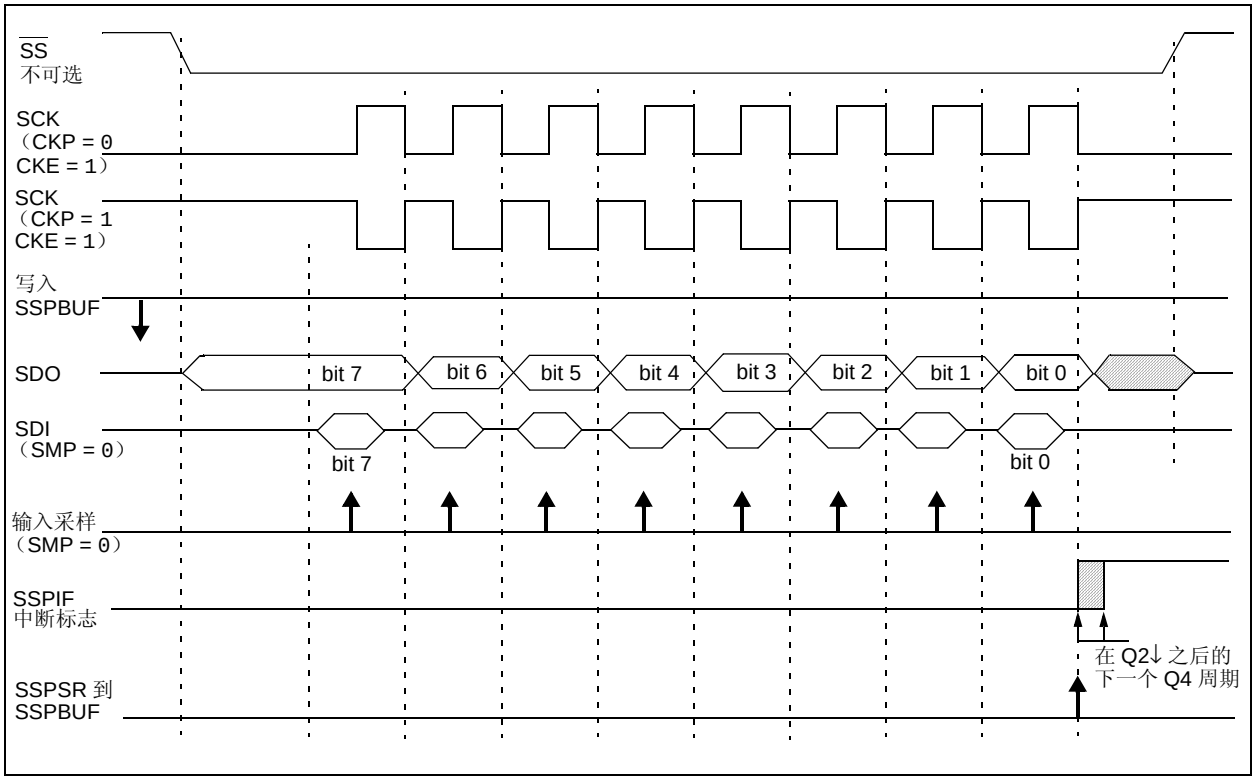


图 18-6：　　SPI 模式波形图 （从模式， CKE = 1）



18.3.9 在功耗管理模式下的操作

在 SPI 主模式下，模块时钟速度可能与全功耗模式下的不同；处于休眠模式时，所有时钟都停止。

在空闲模式下，需要为外设提供一个时钟。该时钟应该来自于主时钟源、辅助时钟源 （32.768 kHz 的 Timer1 振荡器）或 INTRC 时钟源。更多信息，请参见**第 3.3 节 “ 时钟源与振荡器切换 ”**。

在大多数情况下，主器件为 SPI 数据提供的时钟速度并不重要；但是，每个系统都应该评估此因素。

如果允许了 MSSP 中断，那么当主器件发送完数据时，这些中断可以将控制器从休眠模式或某种空闲模式唤醒。如果不想从休眠或空闲模式退出，应该禁止 MSSP 中断。

如果选择了休眠模式，所有模块的时钟都将停止，并且在器件被唤醒前，发送 / 接收将保持此停止状态。当器件返回到运行模式后，MSSP 模块将重新开始发送和接收数据。

在 SPI 从模式下， SPI 发送 / 接收移位寄存器与器件异步工作。这可使器件处于任何功耗管理模式下，而且数据仍可被移入 SPI 发送 / 接收移位寄存器。当 8 位数据全部接收到后， MSSP 中断标志位将置 1，并且如果允许中断的话，器件被唤醒。

18.3.10 复位的影响

复位会禁止 MSSP 模块并终止当前的数据传输。

18.3.11 总线模式兼容性

表 18-1 显示了标准 SPI 模式与 CKP 和 CKE 控制位状态之间的兼容性。

表 18-1： SPI 总线模式

标准 SPI 模式术语	控制位状态	
	CKP	CKE
0, 0	0	1
0, 1	0	0
1, 0	1	1
1, 1	1	0

还有一个 SMP 位用来控制数据何时被采样。

表 18-2： 与 SPI 操作相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	62
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	62
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	62
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	62
TRISF	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	—	62
TRISG	SPIOD	CCP2OD	CCP1OD	TRISG4	TRISG3	TRISG2	TRISG1	TRISG0	62
SSPBUF	MSSP 接收缓冲 / 发送寄存器								60
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	60
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	60

图注： SPI 模式下的 MSSP 模块不使用阴影单元。

18.4 I²C 模式

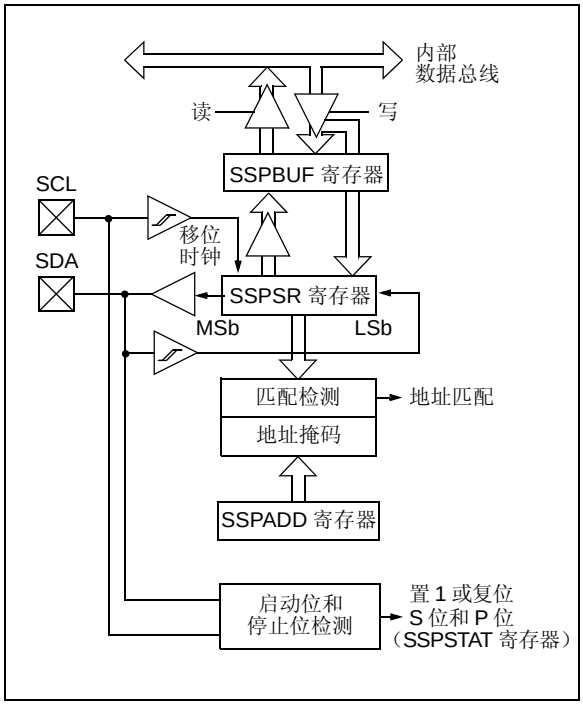
MSSP 模块工作在 I²C 模式时，可以实现所有的主和从功能 （包括广播呼叫支持），并且硬件上提供启动位和停止位的中断来判断总线何时空闲 （多主器件功能）。MSSP 模块实现了标准模式规范以及 7 位和 10 位寻址。

有两个引脚用于数据传输：

- 串行时钟 （SCL） ——RC3/SCK/SCL
- 串行数据 （SDA） ——RC4/SDI/SDA

用户必须通过将 TRISC<4:3> 位置 1 将上述引脚配置为输入引脚。

图 18-7： MSSP 框图 （I²C™ 模式）



18.4.1 寄存器

MSSP 模块有 6 个寄存器用于 I²C 操作。这些寄存器包括：

- MSSP 控制寄存器 1 （SSPCON1）
- MSSP 控制寄存器 2 （SSPCON2）
- MSSP 状态寄存器 （SSPSTAT）
- 串行接收 / 发送缓冲寄存器 （SSPBUF）
- MSSP 移位寄存器 （SSPSR） —— 不可直接访问
- MSSP 地址寄存器 （SSPADD）

SSPCON1、 SSPCON2 和 SSPSTAT 是在 I²C 模式下的控制寄存器和状态寄存器。SSPCON1 和 SSPCON2 寄存器是可读写的。 SSPSTAT 的低 6 位是只读的，而高 2 位是可读写的。

SSPCON2的多数位会呈现不同的功能，取决于该模块是运行在主模式还是从模式下；SSPCON2<5:2>位在从模式下也会显示不同的名称。寄存器 18-5 （主模式）和寄存器 18-6 （从模式）显示了 SSPCON2 的不同方面。

SSPSR 是用来将数据移入或移出的移位寄存器。SSPBUF 是缓冲寄存器，可用于数据字节的写入或读出。

当 MSSP 被配置为工作在 I²C 从模式下时，SSPADD 寄存器将保存从器件的地址。当 MSSP 工作在主模式下时， SSPADD 的低 7 位用作波特率发生器的重载值。

接收数据时，SSPSR 和 SSPBUF 共同构成一个双重缓冲接收器。当 SSPSR 接收到一个完整的字节之后，该字节会被送入SSPBUF，同时将中断标志位SSPIF置1。

在数据发送过程中， SSPBUF 不是双重缓冲的，对 SSPBUF 的写操作将同时写入 SSPBUF 和 SSPSR。

寄存器 18-3： SSPSTAT：MSSP 状态寄存器 （I²C™ 模式）

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P(1)	S(1)	R/$\overline{W}$	UA	BF
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 **SMP：**边沿斜率控制位
在主模式或从模式下：
1 = 标准速度模式下禁止边沿斜率控制 （100 kHz 和 1 MHz）
0 = 高速模式下使能边沿斜率控制 （400 kHz）

bit 6 **CKE：**SMBus 选择位
在主模式或从模式下：
1 = 使能 SMBus 特定输入
0 = 禁止 SMBus 特定输入

bit 5 **D/$\overline{A}$：**数据 / 地址位
在主模式下：
保留。
在从模式下：
1 = 表示上一个接收或发送的字节是数据
0 = 表示上一个接收或发送的字节是地址

bit 4 **P：**停止位 (1)
1 = 表示上次检测到停止位
0 = 上次未检测到停止位

bit 3 **S：**启动位 (1)
1 = 表示上次检测到启动位
0 = 上次未检测到启动位

bit 2 **R/$\overline{W}$：**读 / 写信息位 （仅用于 I²C™ 模式）
在从模式下：(2)
1 = 读
0 = 写
在主模式下：(3)
1 = 正在进行发送
0 = 不在进行发送

bit 1 **UA：**更新地址位 （仅用于 10 位从模式）
1 = 表示用户需要更新 SSPADD 寄存器中的地址
0 = 不需要更新地址

bit 0 **BF：**缓冲区满状态位
在发送模式下：
1 = SSPBUF 已满
0 = SSPBUF 为空
在接收模式下：
1 = SSPBUF 已满 （不包括 $\overline{ACK}$ 位和停止位）
0 = SSPBUF 为空 （不包括 $\overline{ACK}$ 位和停止位）

注 1： 该位在复位及 SSPEN 清零时被清零。
2： 该位保存最后一个地址匹配后的 R/$\overline{W}$ 位信息。该位仅在从地址匹配到下一个启动位、停止位或非 $\overline{ACK}$ 位之间有效。
3： 将该位与 SEN、 RSEN、 PEN、 RCEN 或 ACKEN 进行逻辑或运算将指示 MSSP 是否处于有效模式。

寄存器 18-4： SSPCON1：MSSP 控制寄存器 1 （I²C™ 模式）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN(1)	CKP	SSPM3	SSPM2	SSPM1	SSPM0
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 **WCOL**：写冲突检测位

在主发送模式下：

1 = 当 I²C™ 不满足启动发送数据的条件时，试图向 SSPBUF 寄存器写入数据 （必须用软件清零）

0 = 未发生冲突

在从发送模式下：

1 = 正在发送前一个字时，又有数据写入 SSPBUF 寄存器 （必须用软件清零）

0 = 未发生冲突

在接收模式 （主或从模式）下：

该位是无关位。

bit 6 **SSPOV**：接收溢出指示位

在接收模式下：

1 = SSPBUF 寄存器仍保存前一字节时，接收到一个新的字节 （必须用软件清零）

0 = 无溢出

在发送模式下：

在发送模式下，该位是无关位。

bit 5 **SSPEN**：主同步串口使能位 (1)

1 = 使能串口并将 SDA 和 SCL 引脚配置为串口引脚

0 = 禁止串口并将上述引脚配置为 I/O 端口引脚

bit 4 **CKP**：SCK 释放控制位

在从模式下：

1 = 释放时钟

0 = 保持时钟低电平 （时钟延长），用来确保数据建立时间

在主模式下：

在此模式下未使用。

bit 3-0 **SSPM<3:0>**：同步串口模式选择位

1111 = I²C 从模式， 10 位地址，并允许启动位和停止位中断

1110 = I²C 从模式， 7 位地址，并允许启动位和停止位中断

1011 = I²C 固件控制的主模式 （从器件空闲）

1000 = I²C 主模式，时钟 = FOSC/(4 * (SSPADD + 1))

0111 = I²C 从模式， 10 位地址

0110 = I²C 从模式， 7 位地址

此处未列出的位组合被保留或仅在 SPI 模式下实现。

注 1： 当该位被使能时，必须将 SDA 和 SCL 引脚配置为输入引脚。

寄存器 18-5： SSPCON2：MSSP 控制寄存器 2 （I²C™ 主模式）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT(1)	ACKEN(2)	RCEN(2)	PEN(2)	RSEN(2)	SEN(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 7 **GCEN：**广播呼叫使能位
在主模式下未使用。

bit 6 **ACKSTAT：**应答状态位 （仅用于主发送模式）
1 = 未收到来自从器件的应答
0 = 收到来自从器件的应答

bit 5 **ACKDT：**应答数据位 （仅用于主接收模式） (1)
1 = 无应答
0 = 应答

bit 4 **ACKEN：**应答序列使能位 (2)
1 = 在 SDA 和 SCL 引脚上发出应答序列，并发送 ACKDT 数据位。由硬件自动清零。
0 = 应答序列空闲

bit 3 **RCEN：**接收使能位 （仅用于主接收模式） (2)
1 = 使能 I²C™ 接收模式
0 = 接收空闲

bit 2 **PEN：**停止条件使能位 (2)
1 = 在 SDA 和 SCL 引脚上发出停止条件。由硬件自动清零。
0 = 停止条件空闲

bit 1 **RSEN：**重复启动条件使能位 (2)
1 = 在 SDA 和 SCL 引脚上发出重复启动条件。由硬件自动清零。
0 = 重复启动条件空闲

bit 0 **SEN：**启动条件使能位 (2)
1 = 在 SDA 和 SCL 引脚上发出启动条件。由硬件自动清零。
0 = 启动条件空闲

注 1： 当用户在接收末尾发出一个应答序列时，发送该值。
2： 如果 I²C 模块处于工作状态，可能这些位不会被置 1 （无并行工作），并且可能不会写入 SSPBUF （或禁止写 SSPBUF）。

寄存器 18-6： SSPCON2：MSSP 控制寄存器 2 （I²C™ 从模式）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ADMSK5	ADMSK4	ADMSK3	ADMSK2	ADMSK1	SEN(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 **GCEN：**广播呼叫使能位
1 = 当 SSPSR 中接收到广播呼叫地址 （0000h）时允许中断
0 = 禁止广播呼叫地址

bit 6 **ACKSTAT：**应答状态位
在从模式下未使用。

bit 5-2 **ADMSK<5:2>：**从地址掩码选择位
1 = 使能 SSPADD 相应位的掩码
0 = 禁止 SSPADD 相应位的掩码

bit 1 **ADMSK1：**从地址最低有效位掩码选择位
在 7 位寻址模式下：
1 = 仅使能 SSPADD<1> 的掩码
0 = 仅禁止 SSPADD<1> 的掩码
在 10 位寻址模式下：
1 = 使能 SSPADD<1:0> 的掩码
0 = 禁止 SSPADD<1:0> 的掩码

bit 0 **SEN：**延长使能位 (1)
1 = 为从发送和从接收 （已使能时钟延长）使能时钟延长
0 = 禁止时钟延长

注 1： 如果 I²C™ 模块处于工作状态，可能该位不会被置 1 （无并行工作），并且可能不会写入 SSPBUF （或禁止写 SSPBUF）。

18.4.2 工作原理

通过将 MSSP 使能位 SSPEN（SSPCON1<5>）置 1，可使能 MSSP 模块。

SSPCON1 寄存器用于控制 I^2C 工作模式。可通过设置 4 个模式选择位 （SSPCON1<3:0>）选择以下 I^2C 模式之一：

- I^2C 主模式，时钟 = (FOSC/4) x (SSPADD + 1)
- I^2C 从模式 （7 位地址）
- I^2C 从模式 （10 位地址）
- I^2C 从模式（7 位地址），允许启动位和停止位中断
- I^2C从模式（10位地址），允许启动位和停止位中断
- I^2C 固件控制的主模式，从器件空闲

如果通过将相应的 TRISC 或 TRISD 位置 1，将 SCL 和 SDA 引脚编程为输入引脚，则在 SSPEN 位置 1 时选择任何 I^2C 模式，将强制这两个引脚为漏极开路。要确保此模块正常工作，必须为 SCL 和 SDA 引脚外接上拉电阻。

18.4.3 从模式

在从模式下， SCL 和 SDA 引脚必须被配置为输入（TRISC<4:3> 置 1）。必要时 MSSP 模块将使用输出数据改写输入状态 （从发送器）。

I^2C 从模式硬件总是在地址精确匹配时产生中断。此外，地址掩码功能则可使硬件在多个地址发生匹配时（7 位寻址模式下最多 31 个， 10 位寻址模式下最多 63 个）产生中断。用户也可以通过模式选择位，选择使用启动位或停止位中断。

当地址匹配或在地址匹配后发送的数据被接收时，硬件会自动产生一个应答 （$\overline{ACK}$）脉冲，并把当前 SSPSR 寄存器中接收到的值装入 SSPBUF 寄存器。

只要满足以下条件之一，MSSP 模块就不会产生此 $\overline{ACK}$ 脉冲：

- 在接收到数据前，缓冲区满位 BF（SSPSTAT<0>）被置 1。
- 在接收到数据前，溢出位 SSPOV（SSPCON1<6>）被置 1。

在上述情况下，SSPSR寄存器的值不会装入SSPBUF，但 SSPIF 位会置 1。BF 位是通过读取 SSPBUF 寄存器清零的，而 SSPOV 位是通过软件清零的。

为确保正常工作，SCL时钟输入必须满足最小高电平和最小低电平时间要求。在时序参数 100 和参数 101 中给出了 I^2C 规范的高低电平时间和对 MSSP 模块的具体要求。

18.4.3.1 寻址

一旦 MSSP 模块被使能，它就会等待启动条件出现。启动条件出现后，8 位数据被移入 SSPSR 寄存器。在时钟（SCL）线的上升沿采样所有的输入位。寄存器 SSPSR<7:1> 的值会和 SSPADD 寄存器的值比较，该比较是在第 8 个时钟 （SCL）脉冲的下降沿进行的。如果地址匹配，并且 BF 位和 SSPOV 位都被清零，会发生以下事件：

1. SSPSR 寄存器的值被装入 SSPBUF 寄存器。
2. 缓冲区满标志位 BF 被置 1。
3. 产生 $\overline{ACK}$ 脉冲。
4. 在第 9 个 SCL 脉冲的下降沿， MSSP 中断标志位 SSPIF 置 1 （如果允许中断，则产生中断）。

在 10 位寻址模式下，从器件需要接收两个地址字节。第一个地址字节的高 5 位 （MSb）将指定这是否是一个 10 位地址。R/$\overline{W}$（SSPSTAT<2>）位必须指定写操作，这样从器件才能接收到第二个地址字节。对于 10 位地址，第一个字节应该是 “11110 A9 A8 0”，其中 “A9” 和 “A8” 是该地址的高 2 位。 10 位寻址模式的操作步骤如下，其中 7-9 步是针对从发送器而言的。

1. 接收地址的第一个 （高）字节 （SSPIF、 BF 和 UA 位 （SSPSTAT<1>）置 1）。
2. 用地址的第二个 （低）字节更新 SSPADD 寄存器 （UA 位清零并释放 SCL 线）。
3. 读 SSPBUF 寄存器 （BF 位清零）并将标志位 SSPIF 清零。
4. 接收地址的第二个 （低）字节 （SSPIF、 BF 和 UA 位置 1）。
5. 使用地址的第一个 （高）字节更新 SSPADD 寄存器。如果匹配的话就释放 SCL 时钟线，这将清零 UA 位。
6. 读 SSPBUF 寄存器 （BF 位清零）并将标志位 SSPIF 清零。
7. 接收重复启动条件。
8. 接收地址的第一个 （高）字节 （SSPIF 和 BF 位置 1）。
9. 读 SSPBUF 寄存器 （BF 位清零）并将标志位 SSPIF 清零。

18.4.3.2 地址掩码

将地址的某一位掩码意味着该位可为任意值，此时会应答两个地址并产生一个中断。由于同一时刻可以有多个地址位被掩码，所以在 7 位模式下可应答最多 31 个地址，而在 10 位模式下则可应答最多 63 个地址（见例 18-2）。

不管是否使用地址掩码， I^2C 从器件的工作方式保持不变。但当使用地址掩码时， I^2C 从器件能够应答多个地址并产生中断，此时需要通过查询 SSPBUF 来判断是哪一个地址引起的中断。

在 7 位寻址模式下，地址掩码位 ADMSK<5:1>（SSPCON<5:1>）可用来掩码 SSPADD 寄存器中对应的地址位。如果 ADMSK 的某位被置 1（ADMSK<n> = 1），则对应的地址位被忽略 （SSPADD<n> = x）。对于发出地址应答的模块，只要与没被掩码的地址位匹配就可以了。

在10位寻址模式下，ADMSK<5:2>位可用来掩码SSPADD寄存器中对应的地址位，而 ADMSK1 可以同时掩码地址的低 2 位（SSPADD<1:0>）。如果 ADMSK 的某位是有效的（ADMSK<n> = 1），则对应的地址位被忽略（SSPADD<n> = x）。需要注意的是，尽管在 10 位寻址模式下，高地址位也要用到 SSPADD 寄存器中的某些位，但地址掩码位对这些位不起作用，地址掩码位只会影响低地址位。

注 1： ADMSK1 掩码地址的低 2 位。

2： 地址掩码不会对地址的高 2 位起作用。

例 18-2： 地址掩码示例

7 位寻址：

SSPADD<7:1> = A0h （1010000）（SSPADD<0> 假设为 0）

ADMSK<5:1> = 00111

可被应答的地址：A0h, A2h, A4h, A6h, A8h, AAh, ACh, AEh

10 位寻址：

SSPADD<7:0> = A0h （10100000）（本例中地址高 2 位被忽略，因为它们不受掩码影响）

ADMSK<5:1> = 00111

可被应答的地址：A0h, A1h, A2h, A3h, A4h, A5h, A6h, A7h, A8h, A9h, AAh, ABh, ACh, ADh, AEh, AFh

18.4.3.3 接收

当地址字节的 R/$\overline{W}$ 位清零并发生地址匹配时，SSPSTAT 寄存器的 R/$\overline{W}$ 位清零。接收的地址被装入 SSPBUF 寄存器，且 SDA 线保持低电平（$\overline{ACK}$）。

当发生地址字节溢出时，则不会产生应答脉冲（$\overline{ACK}$）。溢出条件定义为 BF 位（SSPSTAT<0>）被置 1，或 SSPOV 位（SSPCON1<6>）被置 1。

每传输一个数据字节都会产生一个 MSSP 中断。中断标志位 SSPIF 必须用软件清零。通过 SSPSTAT 寄存器可以确定该字节的状态。

如果 SEN 被使能（SSPCON2<0> = 1），SCK/SCL 将在每次数据传输后保持低电平（时钟延长）。必须通过将 CKP 位（SSPCON1<4>）置 1 来释放时钟。更多详细信息，请参见**第 18.4.4 节 “ 时钟延长 ”**。

18.4.3.4 发送

当输入的地址字节的 R/$\overline{W}$ 位置 1 并发生地址匹配时，SSPSTAT 寄存器的 R/$\overline{W}$ 位置 1。接收到的地址被装入 SSPBUF 寄存器。 $\overline{ACK}$ 脉冲在第 9 位发送，同时不管 SEN 的值如何，RC3 引脚保持低电平（更多详细信息，请参见**第 18.4.4 节 “ 时钟延长 ”**）。通过延长时钟，主器件只有在从器件准备好发送数据时，才发出另一个时钟脉冲。发送的数据必须被装入 SSPBUF 寄存器，同时也被装入 SSPSR 寄存器。然后，应通过将 CKP（SSPCON1<4>）置 1 来使能 RC3 引脚。 8 个数据位在 SCL 输入的下降沿被移出。这可确保在 SCL 为高电平期间 SDA 信号是有效的 （图 18-10）。

来自主接收器的 $\overline{ACK}$ 脉冲将在第 9 个 SCL 输入脉冲的上升沿锁存。如果 SDA 信号为高电平（无 $\overline{ACK}$ 应答），那么表示数据传输已完成。在这种情况下，如果从器件锁存了 $\overline{ACK}$，将复位从逻辑，同时从器件监视下一个启动位的出现。如果 SDA 线为低电平 （$\overline{ACK}$），则必须将下一个要发送的数据装入 SSPBUF 寄存器。同样，必须通过将 CKP 位置 1 来使能 RC3 引脚。

每传输一个数据字节都会产生一个 MSSP 中断。SSPIF 位必须用软件清零， SSPSTAT 寄存器用于确定字节的状态。 SSPIF 位在第 9 个时钟脉冲的下降沿被置 1。

图 18-8：　　I^2C™ 从模式接收时序 （SEN = 0， 7 位寻址）

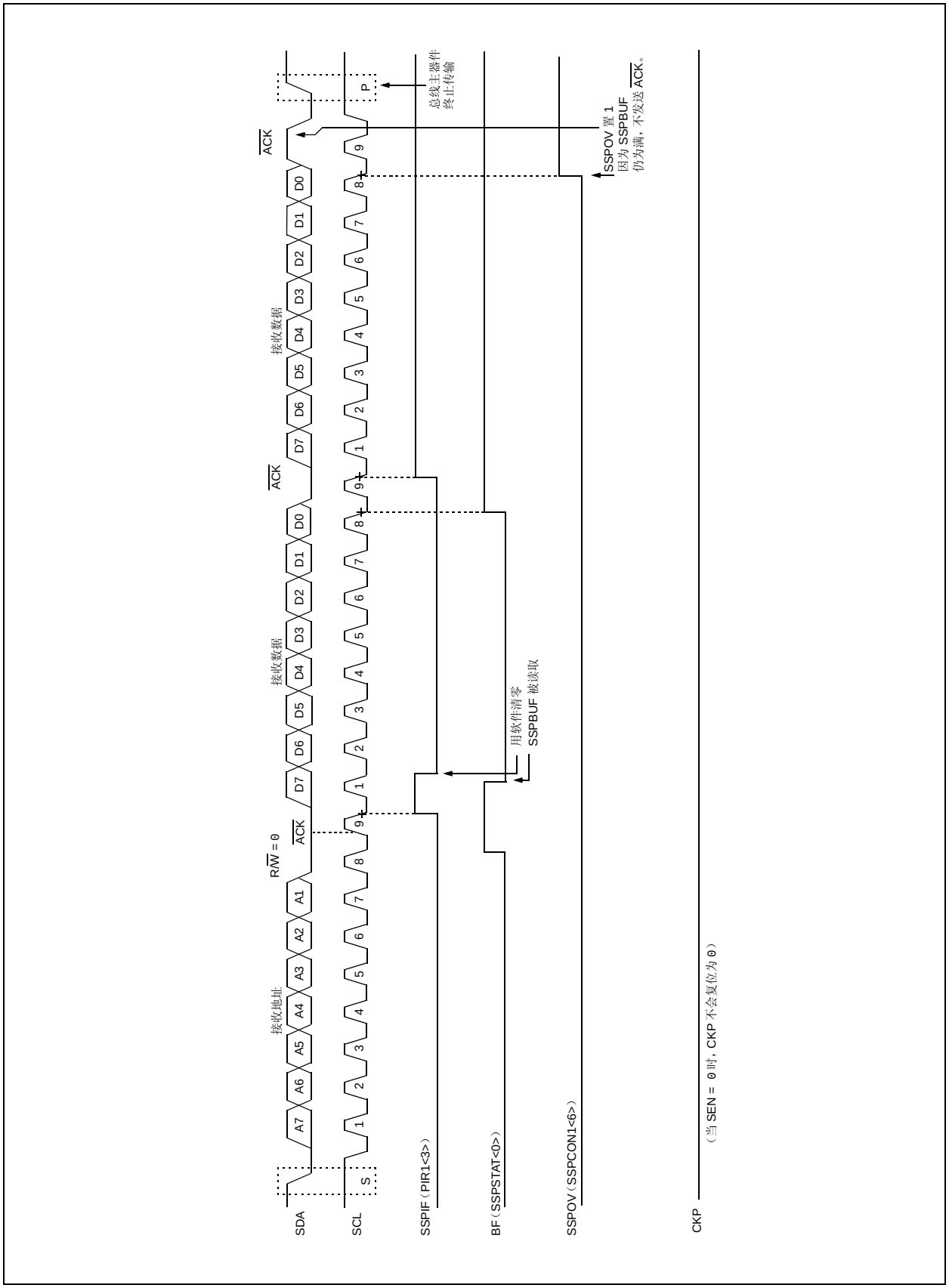
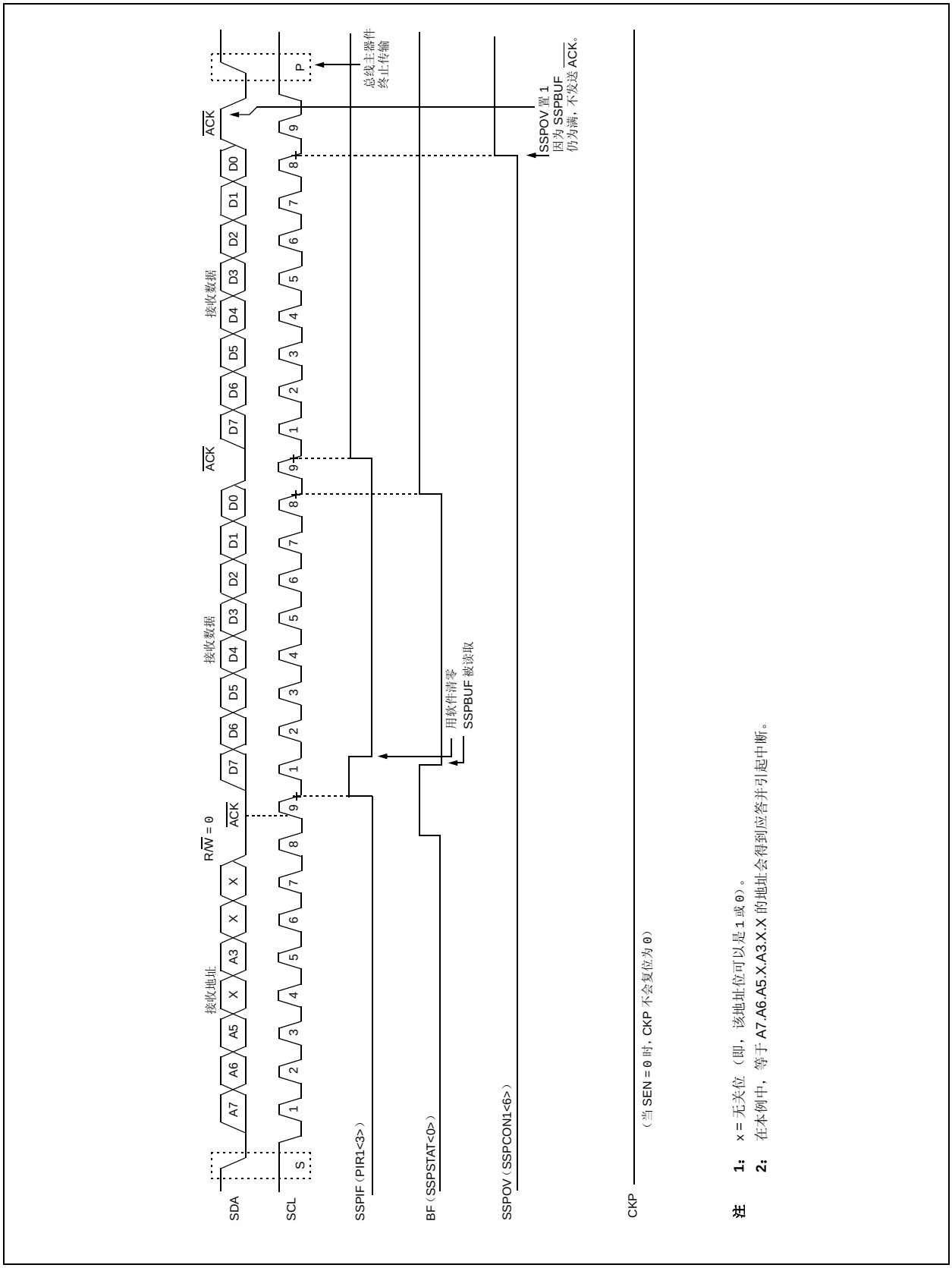


图 18-9：　　I²C™ 从模式接收时序 （SEN = 0 且 ADMSK<5:1> = 01011， 7 位寻址）



注　**1：** x = 无关位 （即，该地址位可以是 1 或 0）。

2： 在本例中，等于 A7.A6.A5.X.A3.X.X 的地址会得到应答并引起中断。

图 18-10：　　I^2C™ 从模式发送时序 （7 位寻址）

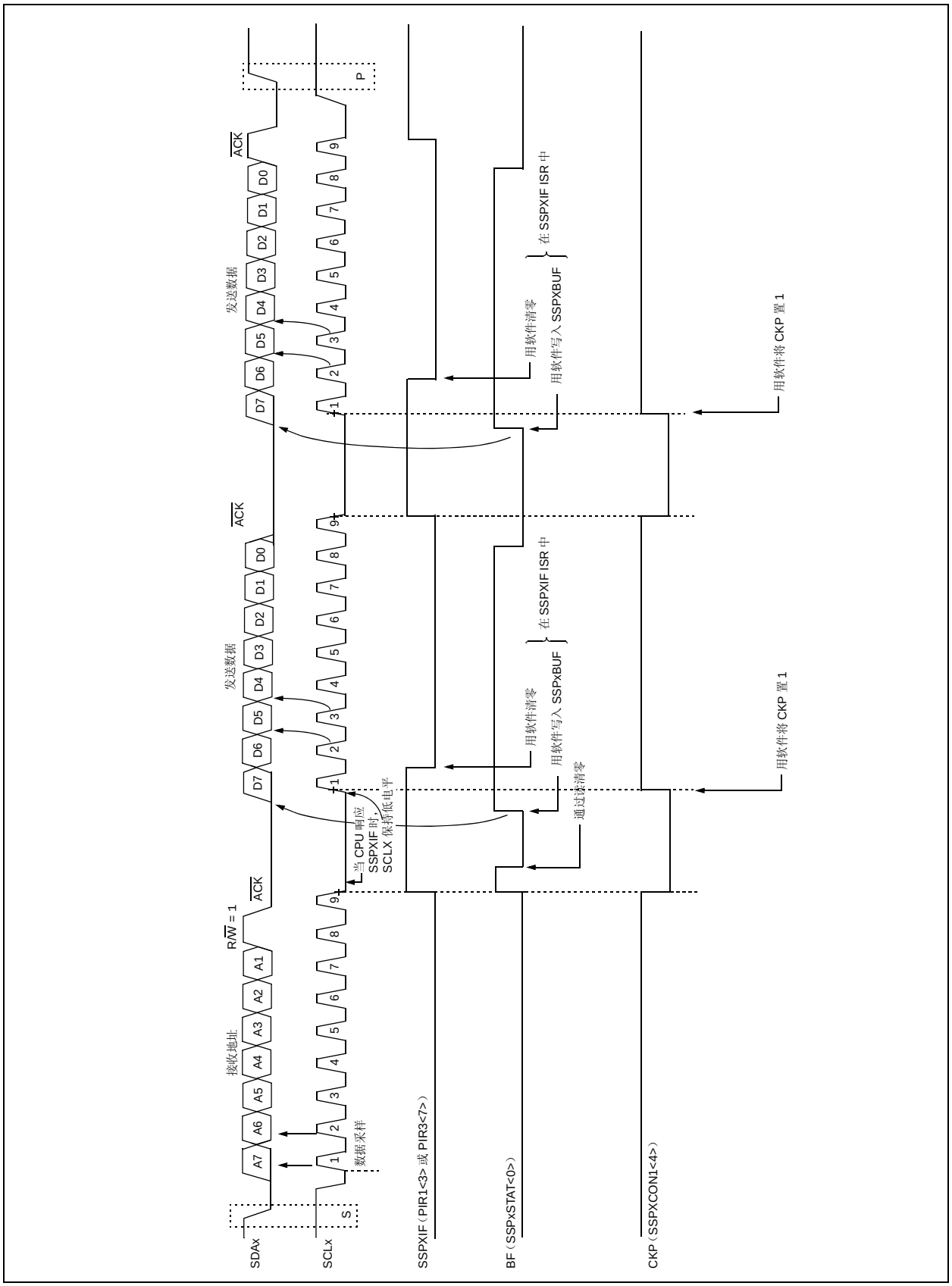


图 18-11：　　I²C™ 从模式接收时序 （SEN = 0， 10 位寻址）

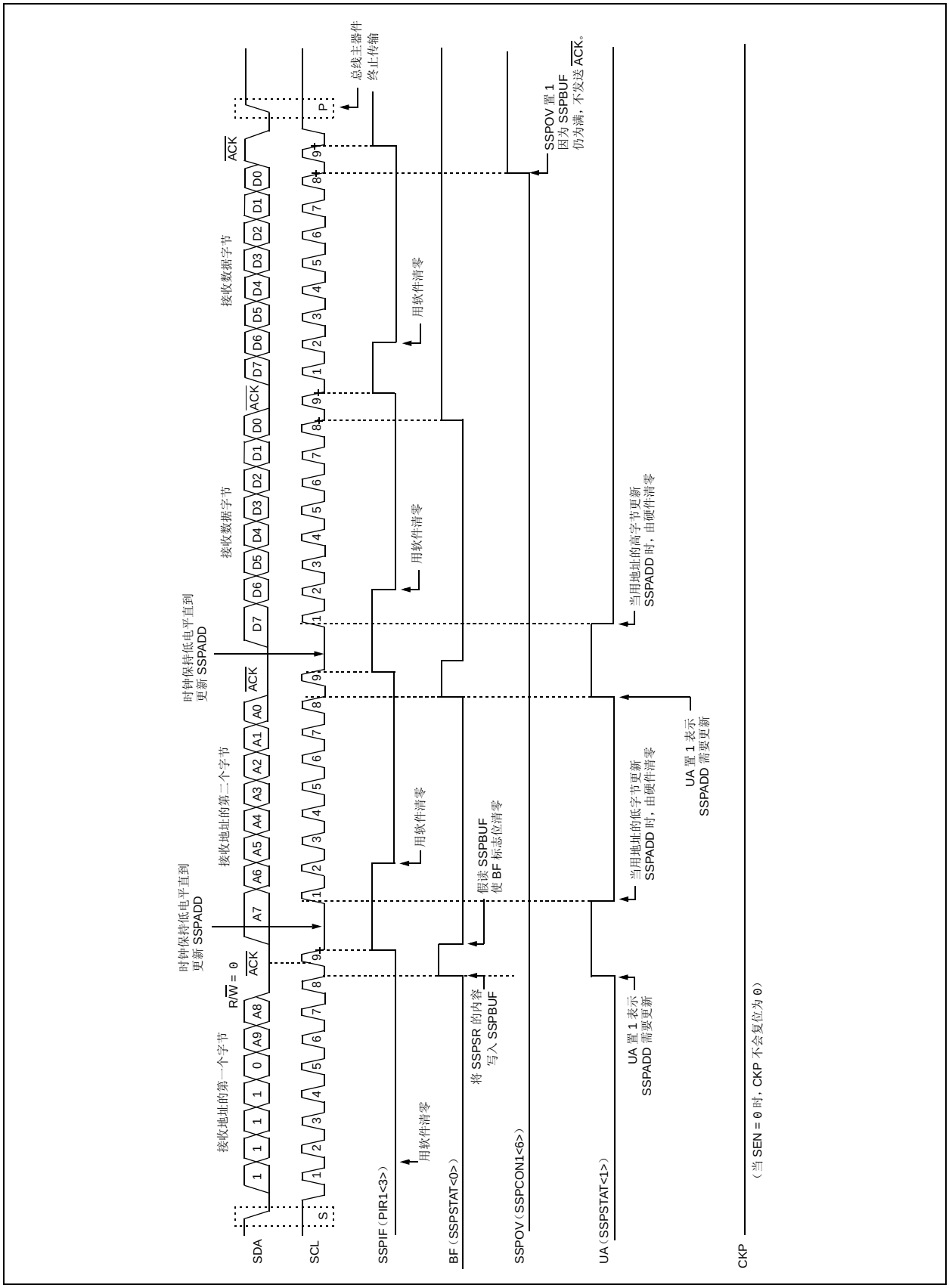
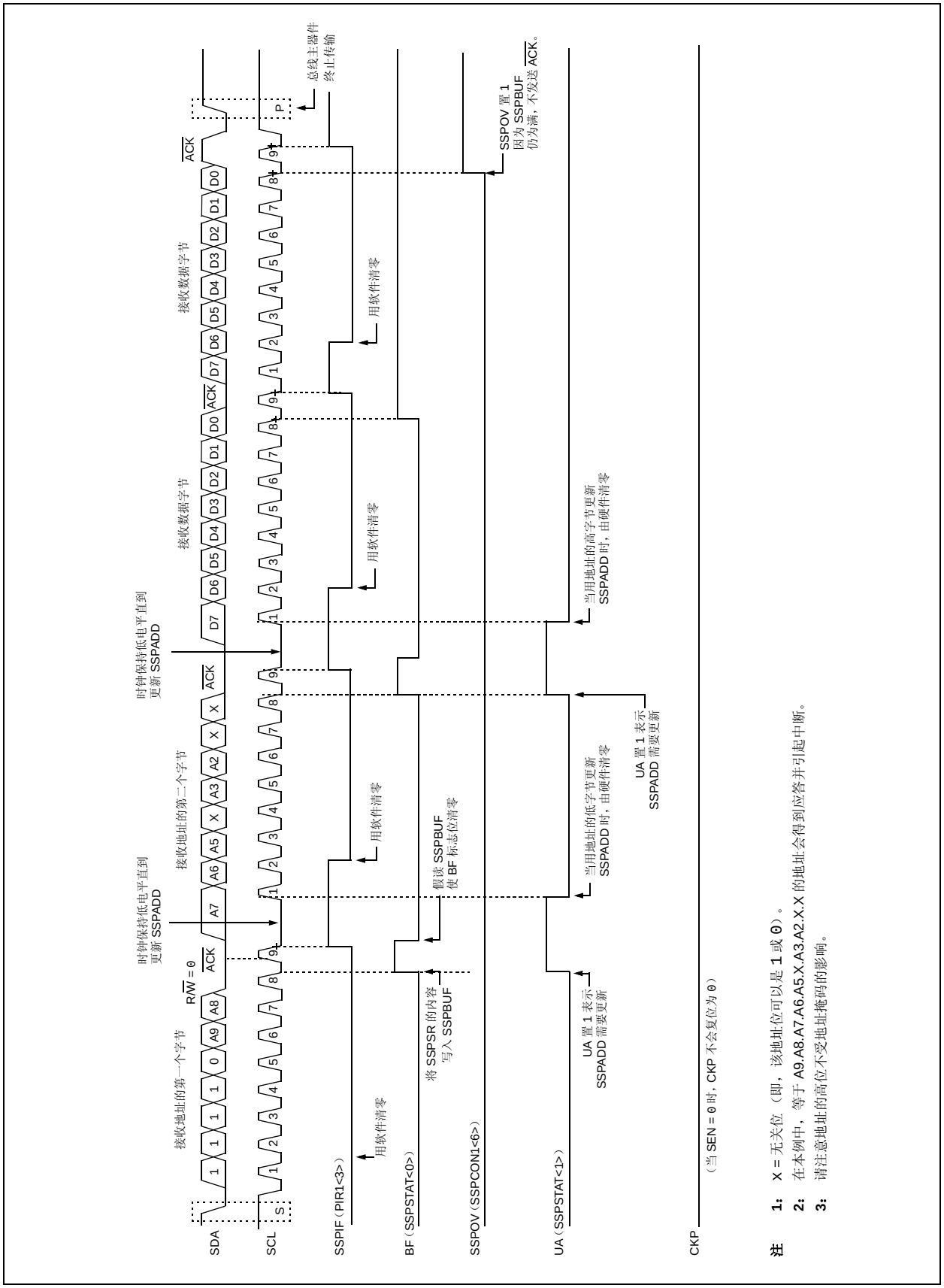


图 18-12：　　I²C™ 从模式接收时序 （SEN = 0 且 ADMSK<5:1> = 01001， 10 位寻址）

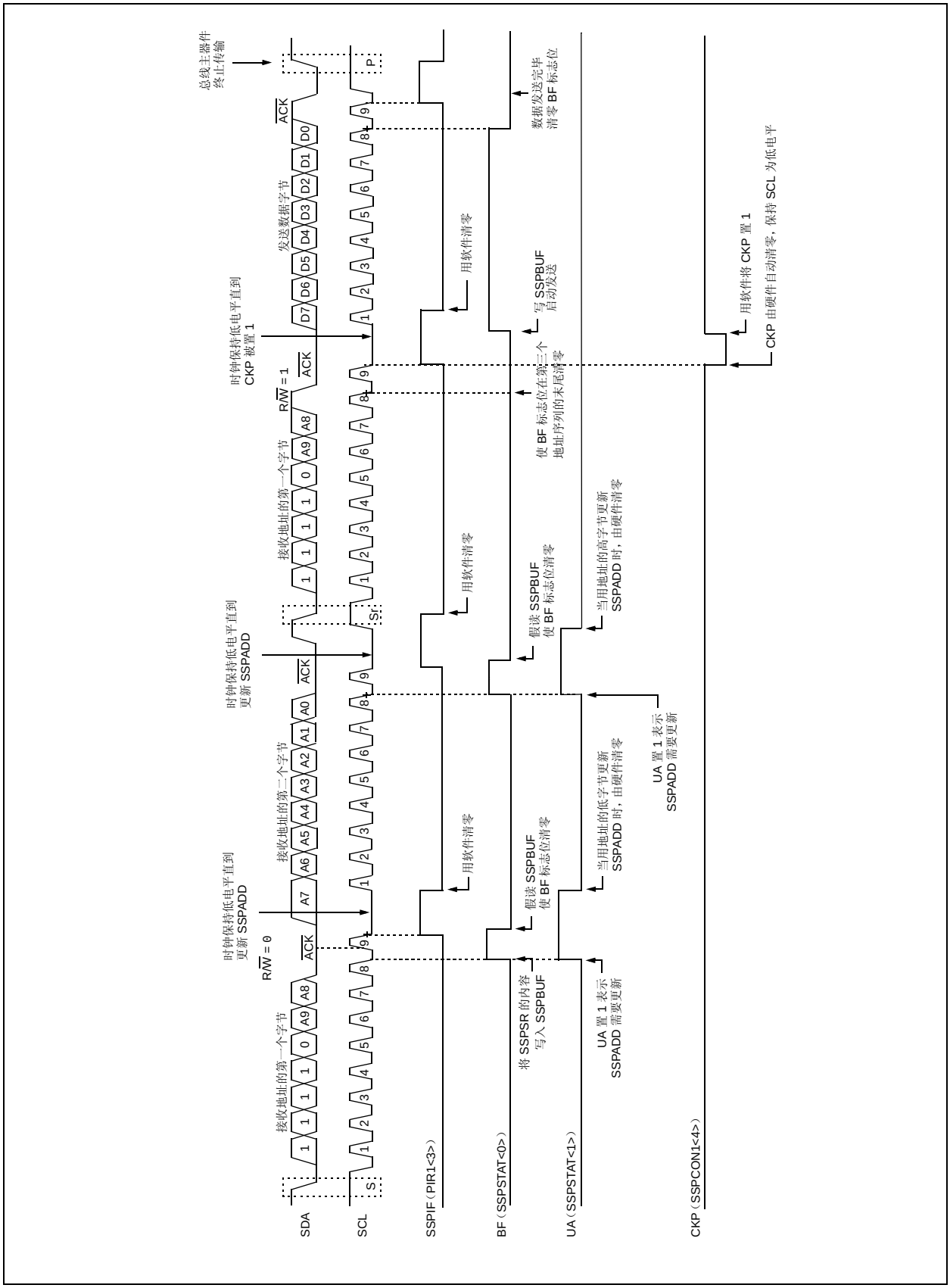


注 1： X = 无关位 （即，该地址位可以是 1 或 0）。

2： 在本例中，等于 A9.A8.A7.A6.A5.X.A3.A2.X.X 的地址会得到应答并引起中断。

3： 请注意地址的高位不受地址掩码的影响。

图 18-13：　　I²C™ 从模式发送时序 （10 位寻址）



18.4.4 时钟延长

7 位和 10 位从模式均能在发送序列期间实现自动时钟延长。

SEN 位 （SSPCON2<0>）允许在接收期间使能时钟延长。将 SEN 置 1 将使 SCL 引脚在每个数据接收序列的末尾保持低电平。

18.4.4.1 7 位从接收模式 （SEN = 1）的时钟延长

在 7 位从接收模式下，如果在 $\overline{\text{ACK}}$ 序列末尾的第 9 个时钟的下降沿将 BF 位置 1，则 SSPCON1 寄存器中的 CKP 位就会自动清零，强制 SCL 输出保持在低电平。CKP 被清零会将 SCL 线拉为低电平。在允许继续接收之前，必须在用户的 ISR 中将 CKP 位置 1。保持 SCL 线为低电平，用户可以在主器件发起另一个接收序列之前，有时间执行 ISR 并读取 SSPBUF 的内容。这将防止发生缓冲区溢出 （见图 18-15）。

注 1： 如果用户在第 9 个时钟的下降沿到来之前读取了 SSPBUF 的内容，使得 BF 位被清零，那么 CKP 位就不会被清零，也不会发生时钟延长。

2： 不管 BF 位的状态如何，CKP 位都可以用软件置 1。为避免溢出，在下一个接收序列开始之前，用户要注意在 ISR 中清零 BF 位。

18.4.4.2 10 位从接收模式 （SEN = 1）的时钟延长

在 10 位从接收模式下，在地址序列中会自动发生时钟延长，但是 CKP 位不会被清零。在这期间，如果 UA 位在第 9 个时钟之后置 1，将启动时钟延长。UA 位在接收到 10 位地址的高字节后被置 1，然后接收 10 位地址的第二个字节并清零 R/$\overline{\text{W}}$ 位。在更新 SSPADD 的时候释放时钟线。如同 7 位模式一样，在每个数据接收序列中均会发生时钟延长。

注： 如果用户在第9个时钟的下降沿出现之前查询 UA 位，并通过更新 SSPADD 寄存器清零 UA 位，而且在此之前用户没有读取 SSPBUF 寄存器使 BF 位清零，则 CKP 位的电平仍然不会被拉低。基于 BF 位状态的时钟延长仅在数据序列中出现，不会出现在地址序列中。

18.4.4.3 7 位从发送模式的时钟延长

如果 BF 位被清零， 7 位从发送模式将通过在第 9 个时钟的下降沿出现后清零 CKP 位来实现时钟延长。上述情形与 SEN 位的状态无关。

用户的 ISR 必须先将 CKP 位置 1 才可以继续发送。在保持 SCL 线为低电平期间，用户在主器件发起另一个发送序列之前，将有时间执行 ISR 并装入 SSPBUF 的内容 （见图 18-10）。

注 1： 如果用户在第9个时钟的下降沿之前就装入 SSPBUF 的内容，使得 BF 位被置 1，CKP 位就不会被清零，也不会发生时钟延长。

2： 不管 BF 位的状态如何， CKP 位都可以用软件置 1。

18.4.4.4 10 位从发送模式的时钟延长

在 10 位从发送模式下，在前两个地址序列中由 UA 位的状态来控制时钟延长，正如同10位从接收模式一样。头两个地址后跟着第三个地址序列，该地址序列包含 10 位地址的高位和被置为 1 的 R/$\overline{\text{W}}$ 位。在执行完第三个地址序列后， UA 位不置 1，此时模块配置为发送模式， BF 标志位控制时钟延长，正如 7 位从发送模式一样 （见图 18-13）。

18.4.4.5 时钟同步和 CKP 位

当 CKP 位被清零时，SCL 输出被强制为 0。然而，将 CKP 位清零不会将 SCL 输出拉为低电平，除非已经采样到 SCL 输出为低电平。因此， CKP 位不会将 SCL 线拉为低电平，除非外部 I^2C 主器件将 SCL 线拉低。SCL 输出将保持低电平，直到 CKP 位置 1 且 I^2C 总线上的所有其他器件将 SCL 电平拉高为止。这可以确保对 CKP 位的写操作不会违反 SCL 的最小高电平时间要求 （见图 18-14）。

图 18-14： 时钟同步时序

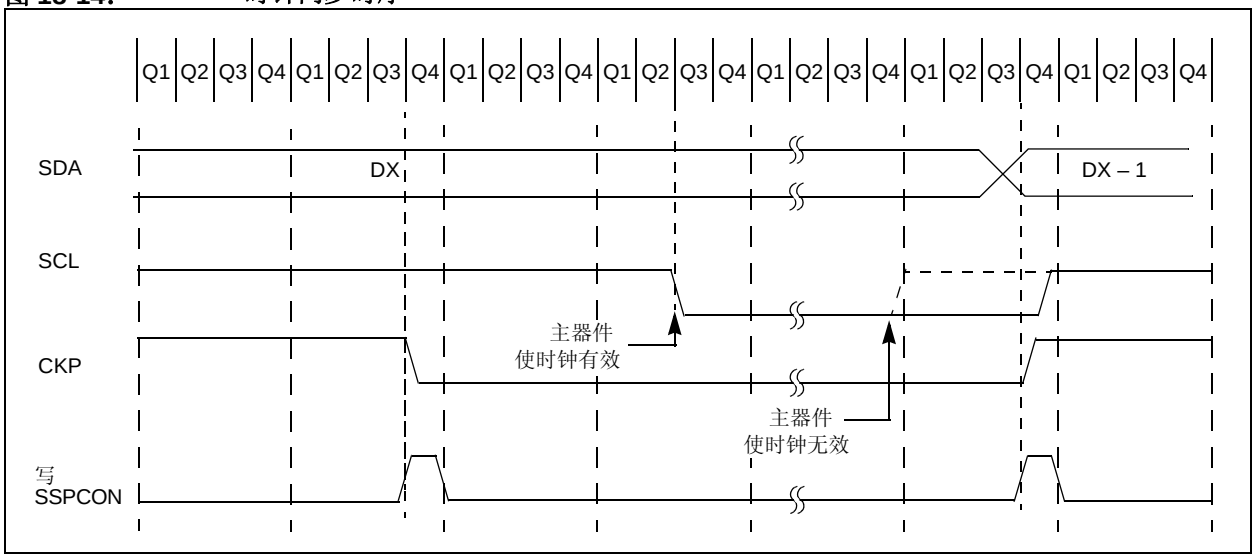


图 18-15：　　I^2C™ 从模式接收时序 （SEN = 1， 7 位寻址）

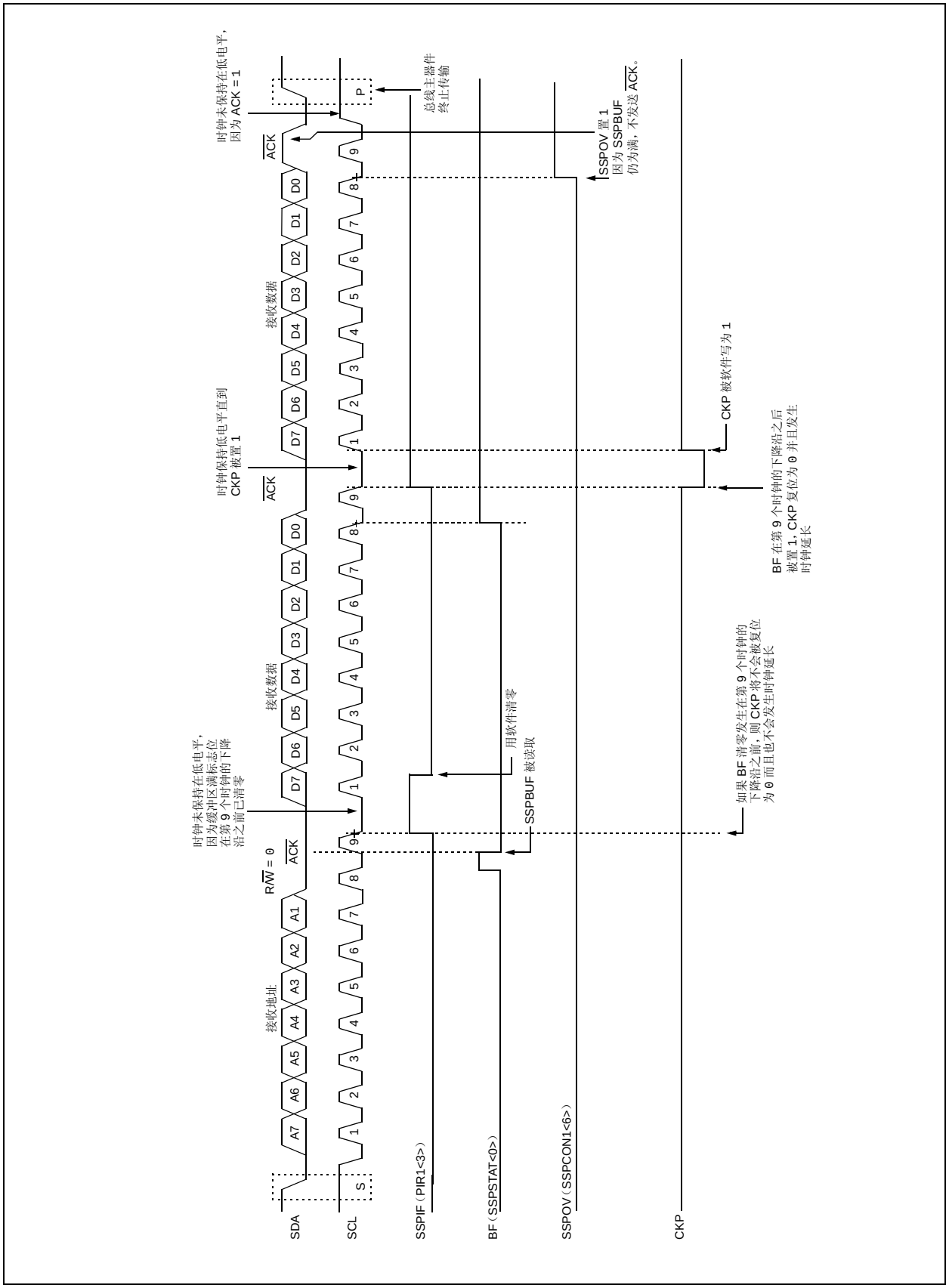
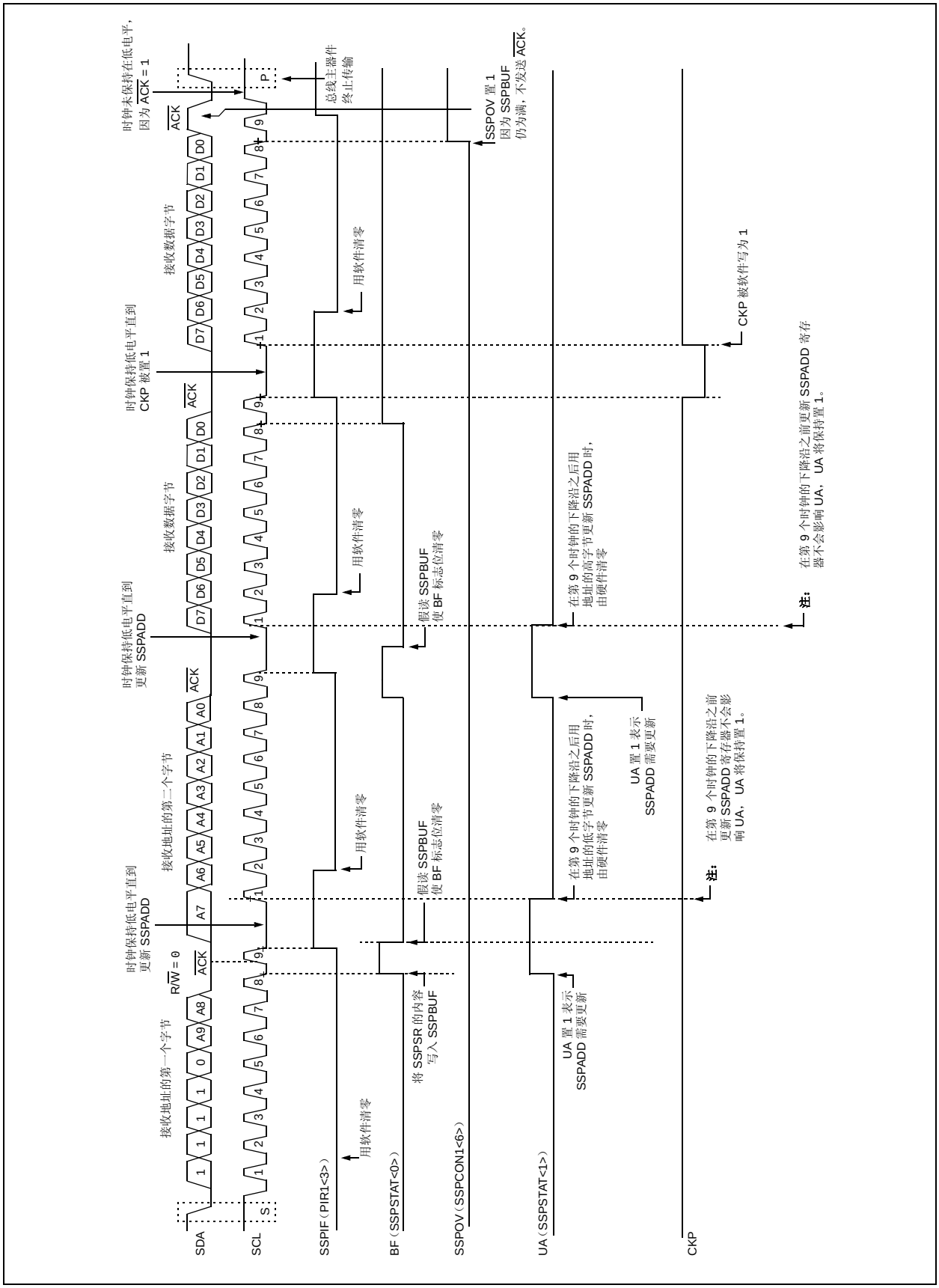


图 18-16：　　I²C™ 从模式接收时序 （SEN = 1， 10 位寻址）



18.4.5 广播呼叫地址支持

在 I^2C 总线的寻址过程中，通常由启动条件后的第一个字节决定主器件将寻址哪个从器件。但广播呼叫地址例外，它能寻址所有器件。当使用这个地址时，理论上所有的器件都应该发送一个应答信号来响应。

广播呼叫地址是 I^2C 协议为特定目的保留的 8 个地址之一。它由全 0 组成，且 R/$\overline{W}$ = 0。

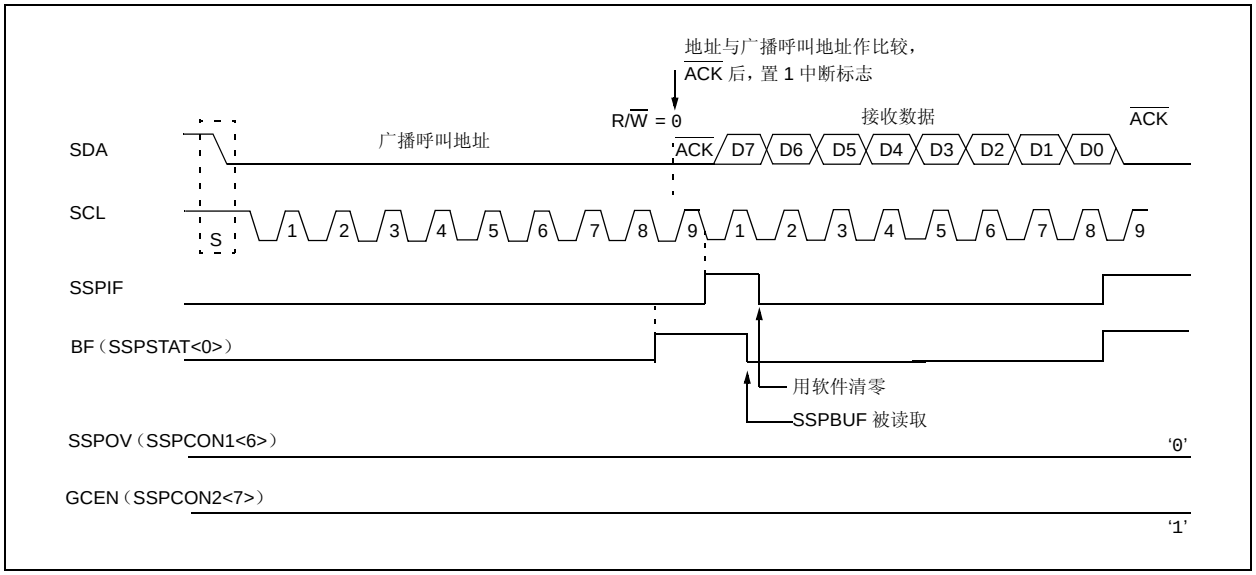
当使能广播呼叫使能位（GCEN）（SSPCON2<7> 置 1）时，即可识别广播呼叫地址。检测到启动位后，8 位数据会被移入 SSPSR，同时将地址与 SSPADD 进行比较。它还会与广播呼叫地址进行比较并用硬件设定。

如果与广播呼叫地址匹配， SSPSR 的值将被传输到 SSPBUF， BF 标志位（第 8 位）置 1，并且 SSPIF 中断标志位在第 9 位 （$\overline{ACK}$ 位）的下降沿置 1。

当中断得到响应时，可以通过读取 SSPBUF 的内容来检查中断源。该值可用于判断是特定器件的地址还是一个广播呼叫地址。

在 10 位模式下，需要更新 SSPADD 用来匹配地址的后半部分，同时 UA 位置 1（SSPSTAT<1>）。如果 GCEN 位置 1 时采样到广播呼叫地址，同时从器件被配置为 10 位寻址模式，则不再需要地址的后半部分，也不会将 UA 位置 1，从器件将在应答后开始接收数据 （图 18-17）。

图 18-17： 从模式广播呼叫地址序列 （7 位或 10 位寻址模式）



18.4.6 主模式

通过将 SSPCON1 中的相应 SSPM 位置 1 和清零，同时将 SSPEN 位置 1，可以使能主模式。在主模式下，SCL 和 SDA 线由 MSSP 硬件控制。

主模式通过在检测到启动条件和停止条件时产生中断来工作。停止（P）位和启动（S）位在复位时或禁止 MSSP 模块时清零。当 P 位置 1 时，可以获得 I^2C 总线的控制权；或者，总线处于空闲状态， P 位和 S 位都清零。

在固件控制的主模式下，用户代码根据启动和停止位条件执行所有的 I^2C 总线操作。

一旦使能主模式，用户即可选择以下 6 项操作。

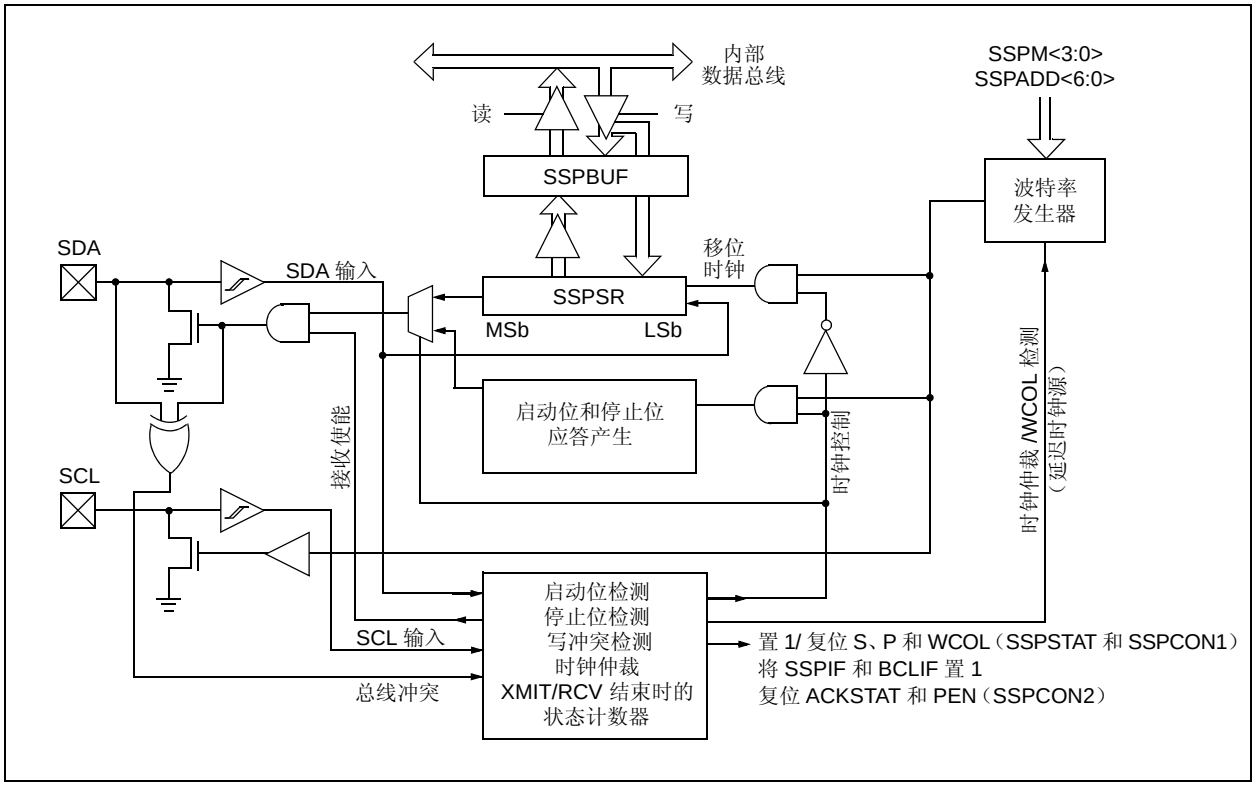
1. 在 SDA 和 SCL 上发出一个启动条件。
2. 在 SDA 和 SCL 上发出一个重复启动条件。
3. 写入 SSPBUF 寄存器，启动数据 / 地址的发送。
4. 配置 I^2C 端口用于接收数据。
5. 在接收数据字节末尾产生应答信号。
6. 在 SDA 和 SCL 上产生一个停止条件。

注： 当配置为 I^2C 主模式时，MSSP 模块不允许事件排队。例如，在启动条件结束前，不允许用户立即写 SSPBUF 寄存器以启动传输。在这种情况下，将不会执行写 SSPBUF，WCOL 位将被置 1，这表明没有发生对 SSPBUF 的写操作。

以下事件会使 MSSP 中断标志位 SSPIF 置 1 （如果允许 MSSP 中断，则产生中断）：

- 启动条件
- 停止条件
- 数据字节发送 / 接收
- 应答发送
- 重复启动

图 18-18： MSSP 框图 （I^2C™ 主模式）



18.4.6.1 I²C 主模式工作原理

主器件产生所有的串行时钟脉冲、启动条件和停止条件。以停止条件或重复启动条件结束传输过程。因为重复启动条件也是下一次串行传输的开始，因此 I²C 总线不会被释放。

在主发送器模式下，串行数据通过 SDA 输出，而串行时钟由 SCL 输出。发送的第一个字节包括作为接收方的从器件地址（7 位）和读 / 写（R/W）位。在这种情况下，R/W 位将是逻辑 0。一次发送 8 位串行数据。每发送一个字节，会收到一个应答位。输出启动和停止条件，表明串行传输的开始和结束。

在主接收模式下，发送的第一个字节包括作为发送方的从器件地址（7 位）和 R/W 位。在这种情况下，R/W 将是逻辑 1。因此，发送的第一个字节是一个 7 位从器件地址，后面跟 1 表示接收。串行数据通过 SDA 接收，而串行时钟由 SCL 输出。一次接收 8 位串行数据。每接收到一个字节，都会发送一个应答位。启动和停止条件分别表明发送的开始和结束。

在 I²C 模式下，将使用 SPI 工作模式中使用的波特率发生器将 SCL 时钟频率设置为 100 kHz、400 kHz 或 1 MHz。更多详细信息，请参见**第 18.4.7 节“波特率”**。

下面是一个典型的发送序列：

1. 用户通过将启动使能位 SEN （SSPCON2<0>）置 1，产生启动条件。
2. SSPIF 置 1。在进行任何其他操作前，MSSP 模块将等待所需的启动时间。
3. 用户将从器件地址装入 SSPBUF 进行发送。
4. 器件地址从 SDA 引脚移出，直到发送完所有 8 位地址数据。
5. MSSP 模块移入来自从器件的 ACK 位，并将它的值写入 SSPCON2 寄存器（SSPCON2<6>）。
6. MSSP 模块在第 9 个时钟周期的末尾将 SSPIF 置 1，产生一个中断。
7. 用户将 8 位数据装入 SSPBUF。
8. 数据从 SDA 引脚移出，直到发送完所有 8 位数据。
9. MSSP 模块移入来自从器件的 ACK 位，并将它的值写入 SSPCON2 寄存器（SSPCON2<6>）。
10. MSSP 模块在第 9 个时钟周期的末尾将 SSPIF 置 1，产生一个中断。
11. 用户通过将停止使能位 PEN （SSPCON2<2>）置 1，产生停止条件。
12. 一旦停止条件完成，将产生一个中断。

18.4.7 波特率

在 I²C 主模式下，波特率发生器（Baud Rate Generator，BRG）的重载值存放在 SSPADD 寄存器的低 7 位（图 18-19）。当发生对 SSPBUF 的写操作时，波特率发生器将自动开始计数。BRG 会递减计数至 0，然后停止直到再次发生重载。 BRG 计数器会在每个指令周期（TCY）中的 Q2 和 Q4 时钟周期上进行两次递减计数。在 I²C 主模式下，会自动重载 BRG。

如果指定操作完成（即，在传输的最后一个数据位后面跟着 ACK），内部时钟将自动停止计数，SCL 引脚将保持在其最后的状态。

表 18-3 给出了不同的指令周期下的时钟频率以及装入 SSPADD 的 BRG 值。

18.4.7.1 功耗管理模式下的波特率发生

当器件工作在任何一种功耗管理模式下时，BRG的时钟源可能会改变频率甚至停止，这取决于模式和所选的时钟源。从辅助时钟或内部振荡器切换到运行或空闲模式很可能会改变 BRG 的时钟速率。在休眠模式下， BRG 根本不会有时钟源。

图 18-19： 波特率发生器框图

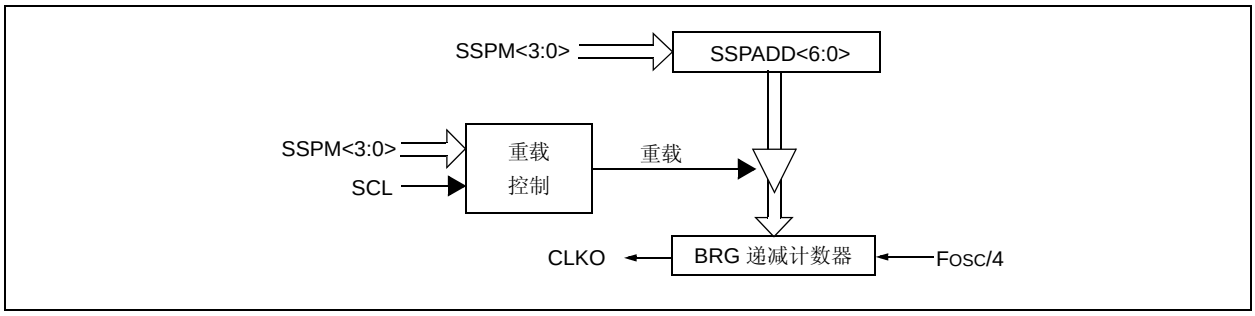


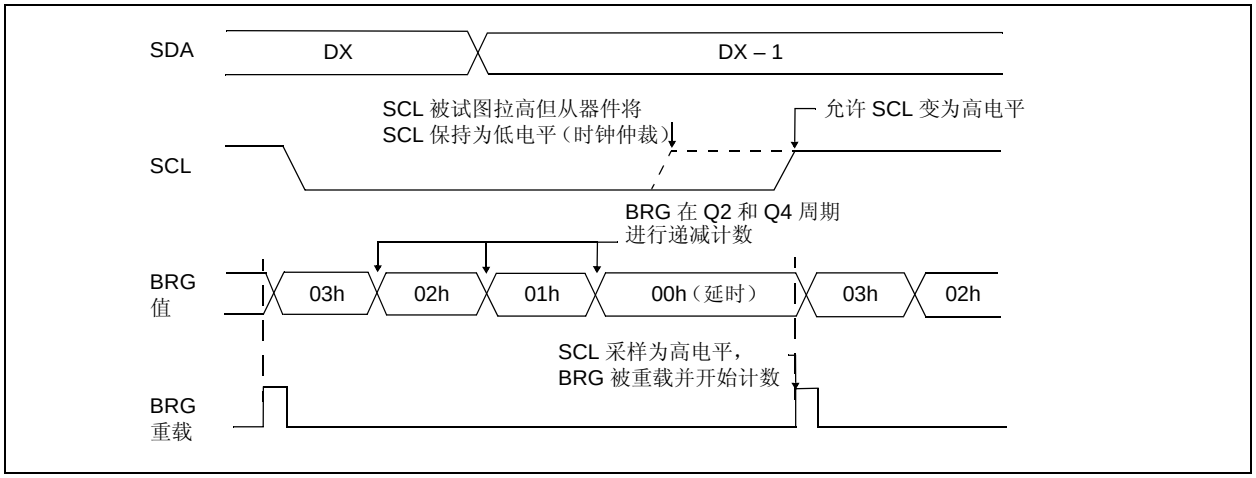
表 18-3： 使用 BRG 的 I²C™ 时钟频率

FCY	FCY * 2	BRG 值	FSCL（两次 BRG 计满返回）
10 MHz	20 MHz	18h	400 kHz
10 MHz	20 MHz	1Fh	312.5 kHz
10 MHz	20 MHz	63h	100 kHz
4 MHz	8 MHz	09h	400 kHz
4 MHz	8 MHz	0Ch	308 kHz
4 MHz	8 MHz	27h	100 kHz
1 MHz	2 MHz	02h	333 kHz
1 MHz	2 MHz	09h	100 kHz
1 MHz	2 MHz	00h	1 MHz

18.4.7.2 时钟仲裁

如果在任何接收、发送或重复启动 / 停止条件期间，主器件拉高了 SCL 引脚（允许 SCL 引脚悬空为高电平），就会发生时钟仲裁。当允许 SCL 引脚悬空为高电平时，波特率发生器 （BRG）暂停计数，直到 SCL 引脚被实际采样到高电平为止。然后波特率发生器将被重新装入 SSPADD<6:0> 的值并开始计数。这可以保证当外部器件将时钟拉低时， SCL 在至少一个 BRG 计满返回计数周期内保持高电平 （图 18-20）。

图 18-20： 带有时钟仲裁的波特率发生器时序



18.4.8 I²C 主模式启动条件时序

要发出启动条件，用户应将启动条件使能位 SEN（SSPCON2<0>）置 1。当 SDA 和 SCL 引脚采样为高电平时，波特率发生器重新装入 SSPADD<6:0> 的内容并开始计数。如果波特率发生器发生超时（TBRG）时，SCL 和 SDA 都采样为高电平，则 SDA 引脚被驱动为低电平。当 SCL 为高电平时，将 SDA 驱动为低电平将产生启动条件，并使 S 位（SSPSTAT<3>）置 1。随后波特率发生器重新装入SSPADD<6:0>的内容并恢复计数。当波特率发生器再次超时（TBRG）时，SEN 位（SSPCON2<0>）将自动被硬件清零，波特率发生器暂停工作，SDA 线保持低电平，启动条件结束。

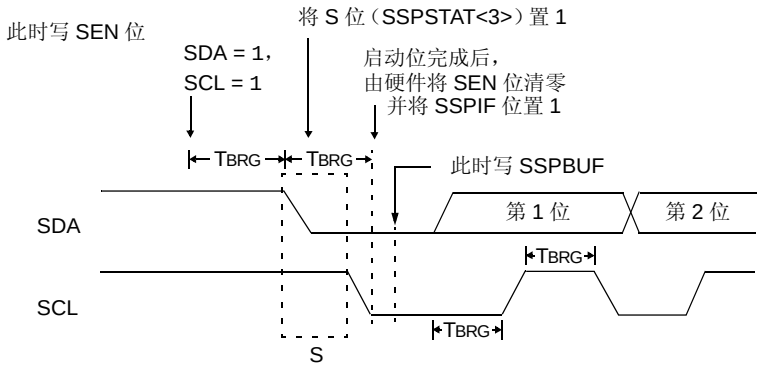
注： 如果在启动条件开始时，SDA 和 SCL 引脚已经采样为低电平，或者在启动条件期间，SCL 在 SDA 线被驱动为低电平之前已经采样为低电平，则会发生总线冲突。总线冲突中断标志位 BCLIF 置 1，启动条件中止，I²C 模块复位到空闲状态。

18.4.8.1 WCOL 状态标志

如果用户在启动序列过程中写 SSPBUF，则 WCOL 位被置 1，同时缓冲区内容不变（写操作无效）。

注： 由于不允许事件排队，在启动条件结束之前，不能写 SSPCON2 的低 5 位。

图 18-21： 第一个启动位时序



18.4.9 I²C 主模式重复启动条件时序

将 RSEN 位（SSPCON2<1>）编程为高电平，并且 I²C 逻辑模块处于空闲状态时，就会产生重复启动条件。当 RSEN 位置 1 时， SCL 引脚被拉为低电平。当 SCL 引脚采样为低电平时，波特率发生器装入 SSPADD<6:0> 的值并开始计数。在该波特率发生器计数周期（TBRG）内 SDA 引脚被释放 （被拉高）。当波特率发生器超时时，如果 SDA 采样为高电平， SCL 引脚将被拉高。当 SCL 被采样为高电平时，波特率发生器重新装入 SSPADD<6:0> 的值并开始计数。 SDA 和 SCL 必须在一个计数周期 TBRG 内采样为高电平。接下来，在一个 TBRG 中，将 SDA 引脚驱动为低电平 （SDA = 0），同时 SCL 保持高电平。然后 RSEN 位（SSPCON2<1>）将自动清零，这次波特率发生器不会重载，SDA 引脚保持低电平。一旦在 SDA 和 SCL 引脚上检测到启动条件，S 位 （SSPSTAT<3>）将被置 1。直到波特率发生器发生超时后， SSPIF 位才会置 1。

注 1： 有任何其他事件在进行时，编程设置 RSEN 无效。

2： 在重复启动条件期间，以下事件将会导致总线冲突：

- 当 SCL 由低电平变为高电平时， SDA 采样为低电平。
- 在 SDA 被拉低之前， SCL 变为低电平。这表明另一个主器件正试图发送一个数据 1。

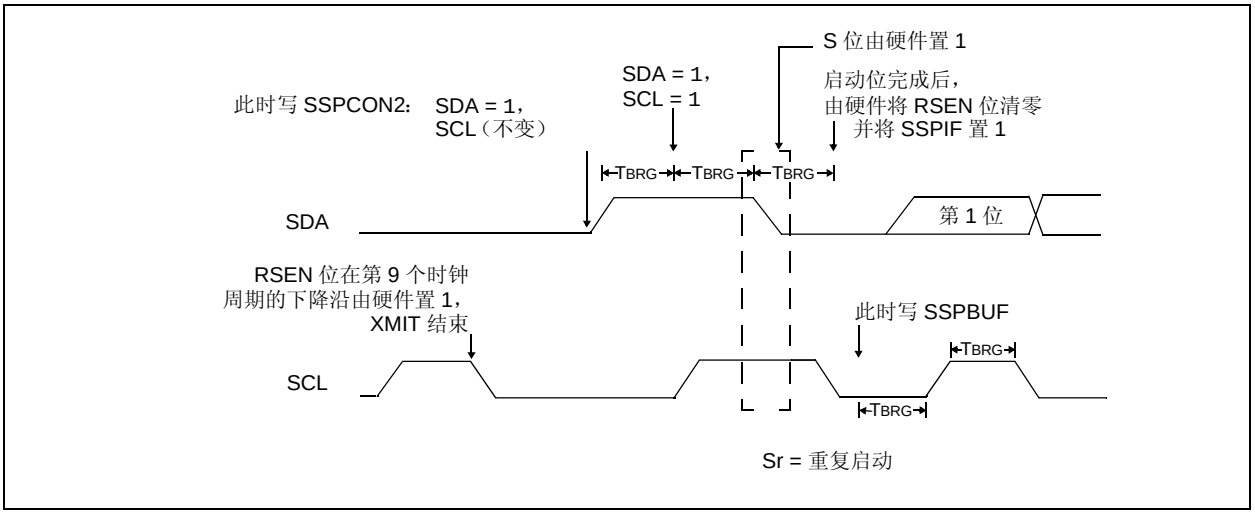
一旦 SSPIF 位被置 1，用户便可以在 7 位地址模式下将 7 位地址，或者在 10 位地址模式下将默认的第一个地址字节写入 SSPBUF。当发送完第一个 8 位数据并接收到一个 ACK 后，用户可以发送另外 8 位地址 （10 位地址模式）或 8 位数据 （7 位地址模式）。

18.4.9.1 WCOL 状态标志

如果用户在重复启动序列过程中写 SSPBUF，则 WCOL 标志被置 1，同时缓冲区内容不变 （写操作无效）。

注： 由于不允许事件排队，在重复启动条件结束之前，不能写 SSPCON2 的低 5 位。

图 18-22： 重复启动条件波形图



18.4.10 I²C 主模式下的发送

发送一个数据字节、一个 7 位地址或一个 10 位地址的一半，都是通过写一个值到 SSPBUF 寄存器来实现的。该操作将使缓冲区满位 BF 置 1，波特率发生器开始计数，同时开始下一次发送。在 SCL 的下降沿有效后（见数据保持时间规范参数 106），地址 / 数据的每一位被移出至 SDA 引脚。在一个波特率发生器计满返回计数周期（TBRG）内，SCL 保持低电平。数据应该在 SCL 释放为高电平前保持有效（见数据建立时间规范参数 107）。当 SCL 引脚释放为高电平时，它将在一个 TBRG 内保持高电平状态。在此期间以及 SCL 的下一个下降沿之后的一段时间内，SDA 引脚上的数据必须保持稳定。在第 8 位数据被移出（第 8 个时钟周期的下降沿）之后，BF 标志位被清零，同时主器件释放 SDA。此时如果发生地址匹配或是数据被正确接收，被寻址的从器件将在第 9 个时钟周期发出一个 ACK 位作为响应。ACK 的状态在第 9 个时钟周期的下降沿写入 ACKDT 位。主器件接收到应答之后，应答状态位 ACKSTAT 会被清零。如果未收到应答，则该位被置 1。第 9 个时钟周期之后，SSPIF 位会置 1，主时钟（波特率发生器）暂停，直到下一个数据字节装入 SSPBUF，SCL 引脚保持低电平，SDA 保持不变（图 18-23）。

在写 SSPBUF 之后，地址的每一位在 SCL 的下降沿被移出，直到所有 7 位地址位和 R/W 位都被移出。在第 8 个时钟的下降沿，主器件将 SDA 引脚拉为高电平，以允许从器件发出一个应答响应。在第 9 个时钟的下降沿，主器件通过采样 SDA 引脚来判断地址是否被从器件识别。ACK 位的状态被装入 ACKSTAT 状态位（SSPCON2<6>）。在发送地址的第 9 个时钟下降沿之后，SSPIF 置 1，BF 标志位清零，波特率发生器关闭直到下一次写 SSPBUF，且 SCL 引脚保持低电平，允许 SDA 引脚悬空。

18.4.10.1 BF 状态标志

在发送模式下，BF 位（SSPSTAT<0>）在 CPU 写 SSPBUF 时置 1，在所有 8 位数据移出后清零。

18.4.10.2 WCOL 状态标志

如果用户在发送过程中（即，SSPSR 仍在移出数据字节时）写 SSPBUF，则 WCOL 位被置 1，并且在写 SSPBUF 之后的 2 个 TCY 内缓冲区内容不变（写操作无效）。如果在 2 个 TCY 内 SSPBUF 被重新写入，则 WCOL 位被置 1 并且 SSPBUF 被更新。这可能导致传输被破坏。

用户应在每次写 SSPBUF 后检查 WCOL 标志位是否清零，以确保传输正确。在所有情况下，WCOL 都必须用软件清零。

18.4.10.3 ACKSTAT 状态标志

在发送模式下，当从器件已发送应答响应（ACK = 0）时，ACKSTAT 位（SSPCON2<6>）清零；当从器件没有应答（ACK = 1）时，该位置 1。从器件在识别出其地址（包括广播呼叫地址）或正确接收数据后，会发送一个应答。

18.4.11 I²C 主模式接收

通过编程接收使能位 RCEN（SSPCON2<3>）使能主模式接收。

注：	将 RCEN 位置 1 前，MSSP 必须处于空闲状态，否则对 RCEN 位置 1 将无效。

波特率发生器开始计数，每次计满返回时，SCL 引脚的状态发生改变（由高变低或由低变高），数据被移入 SSPSR。第 8 个时钟的下降沿之后，接收使能标志位自动清零，SSPSR 的内容装入 SSPBUF，BF 标志位置 1，SSPIF 标志位置 1，波特率发生器暂停计数，且 SCL 保持为低电平。此时 MSSP 处于空闲状态，等待下一条命令。当 CPU 读缓冲区时，BF 标志位将自动清零。通过将应答序列使能位 ACKEN（SSPCON2<4>）置 1，用户可以在接收结束时发送应答位。

18.4.11.1 BF 状态标志

接收数据过程中，把地址或数据字节从 SSPSR 装入 SSPBUF 时，BF 位置 1。在读 SSPBUF 寄存器时将其清零。

18.4.11.2 SSPOV 状态标志

接收数据过程中，当SSPSR接收到8位数据时，SSPOV 位置 1，BF 标志位已经在上一次接收时置 1。

18.4.11.3 WCOL 状态标志

如果用户在接收过程中（即，SSPSR 仍在移入数据字节时）写 SSPBUF，则 WCOL 位被置 1，同时缓冲区内容不变（写操作无效）。

图 18-23：　　I²C™ 主模式发送波形图 （7 位或 10 位寻址）

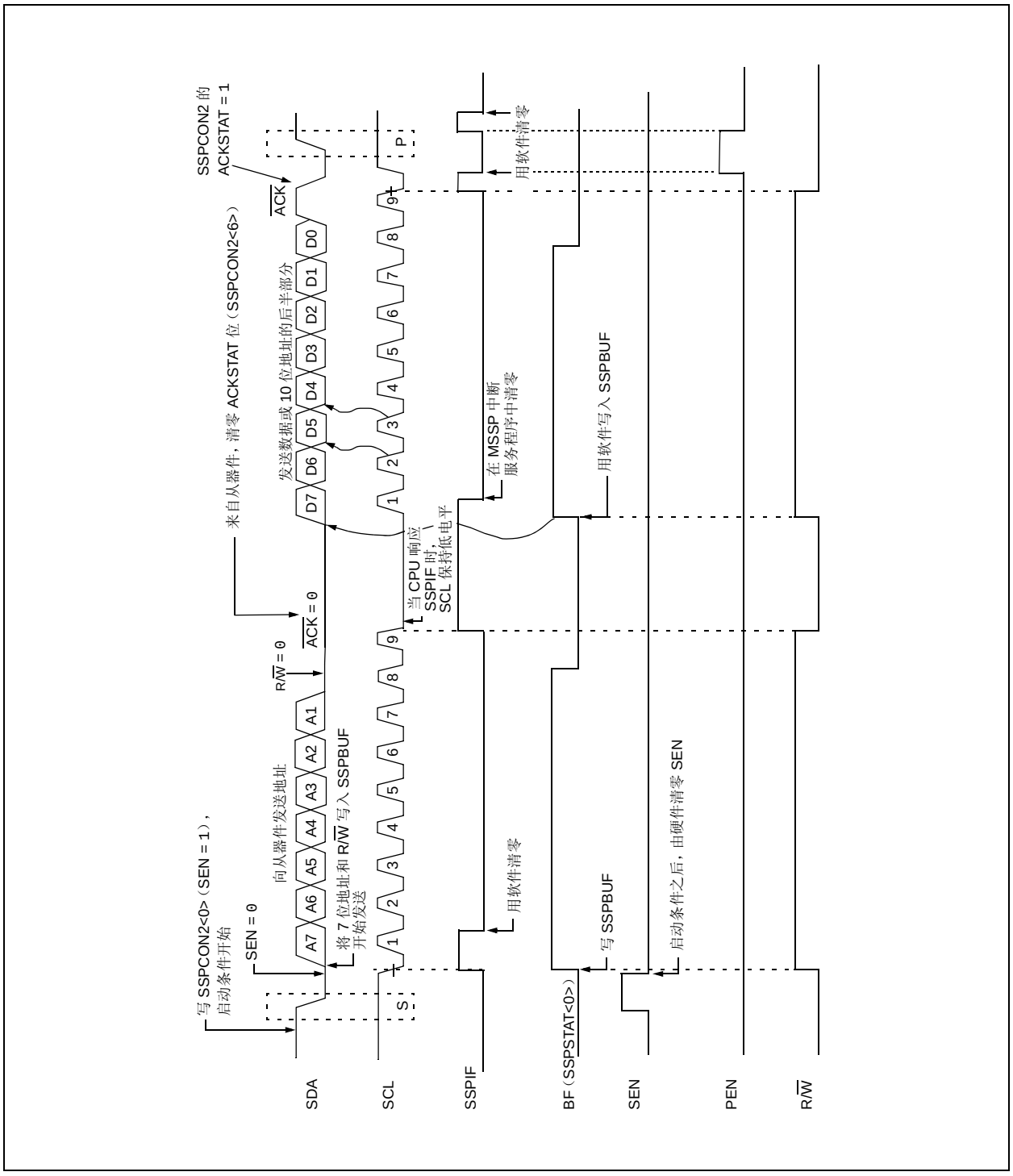
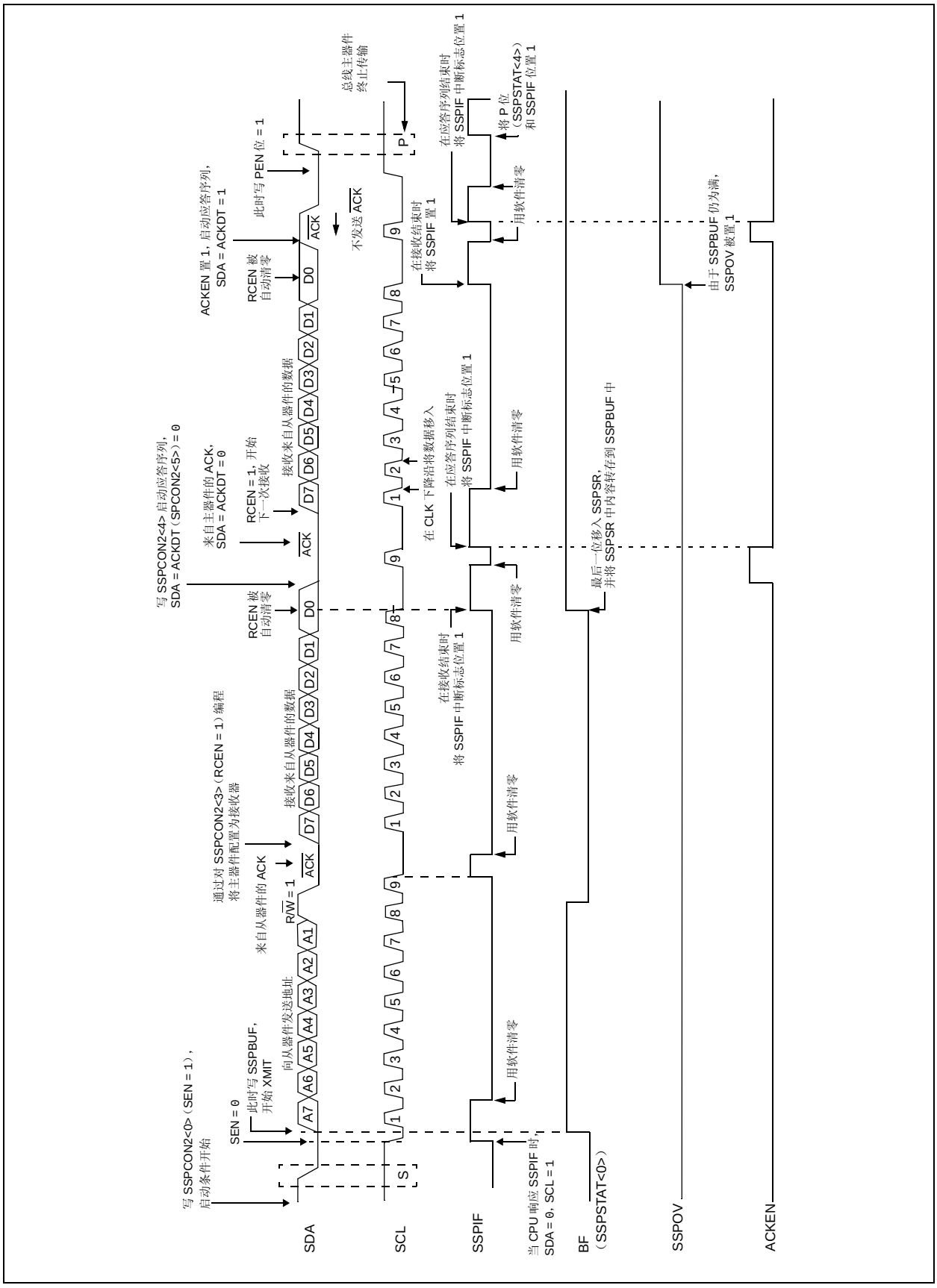


图 18-24：　　I²C™ 主模式接收波形图 （7 位寻址）



18.4.12 应答序列时序

将应答序列使能位 ACKEN（SSPCON2<4>）置 1 即可使能应答序列。当该位被置 1 时， SCL 引脚被拉低，应答数据位的内容输出到 SDA 引脚上。如果用户希望产生一个应答，则应该将 ACKDT 位清零。否则，用户要在应答序列开始前将 ACKDT 位置 1。然后波特率发生器进行一个周期 （TBRG）的计数，随后 SCL 引脚电平被拉高。当 SCL 引脚采样为高电平（时钟仲裁）时，波特率发生器再进行一个 TBRG 周期的计数。然后 SCL 引脚被拉低。在这之后，ACKEN 位自动清零，波特率发生器关闭， MSSP 模块进入空闲模式 （图 18-25）。

18.4.12.1 WCOL 状态标志

如果用户在应答序列进行过程中写SSPBUF，则WCOL 被置 1，同时缓冲区内容不变 （写操作无效）。

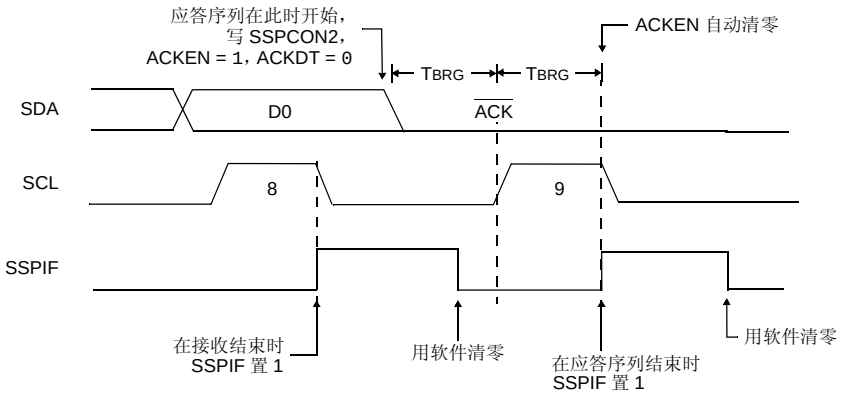
18.4.13 停止条件时序

如果将停止序列使能位 PEN （SSPCON2<2>）置 1，则在接收 / 发送结束时，SDA 引脚上将产生停止位。在接收 / 发送结束时，SCL 引脚在第 9 个时钟的下降沿后保持低电平。当 PEN 位置 1 时，主器件将 SDA 线置为低电平。当 SDA 线采样为低电平时，波特率发生器被重载并递减计数至0。当波特率发生器发生超时时，SCL 引脚被拉为高电平，在一个 TBRG （波特率发生器计满返回周期）之后， SDA 引脚将被拉高。当 SDA 引脚采样为高电平且SCL也是高电平时，P位（SSPSTAT<4>）置 1。另一个 TBRG 之后， PEN 位被清零，同时 SSPIF 位被置 1 （图 18-26）。

18.4.13.1 WCOL 状态标志

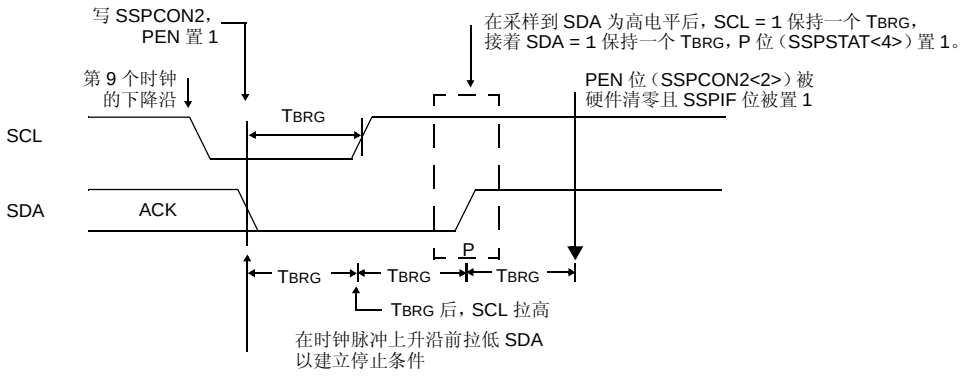
如果用户在停止序列进行过程中写SSPBUF，则WCOL 位被置 1，同时缓冲区内容不变 （写操作无效）。

图 18-25： 应答序列波形图



注： TBRG = 一个波特率发生器周期。

图 18-26： 停止条件接收或发送模式



注： TBRG = 一个波特率发生器周期。

18.4.14 休眠模式下的操作

在休眠模式下，I²C 模块能够接收地址或数据，并且当地址匹配或字节传输完成时，如果允许 MSSP 中断，将唤醒处理器。

18.4.15 复位的影响

复位会禁止 MSSP 模块并终止当前的数据传输。

18.4.16 多主模式

在多主模式下，在检测到启动和停止条件时将产生中断，这可用于判断总线是否空闲。停止（P）位和启动（S）位在复位或禁止 MSSP 模块时清零。当 P 位（SSPSTAT<4>）置 1 时，可以取得 I²C 总线的控制权；或者，总线处于空闲状态， S 位和 P 位都清零。当总线忙时，一旦出现停止条件，将产生 MSSP 中断。

在多主模式下，必须一直监视 SDA 线来进行仲裁，查看信号电平是否为期望的输出电平。此操作由硬件实现，其结果保存在 BCLIF 位中。

可能导致仲裁失败的情况是：

- 地址传输
- 数据传输
- 启动条件
- 重复启动条件
- 应答条件

18.4.17 多主器件通信、总线冲突和总线仲裁

多主模式是通过总线仲裁来支持的。当主器件将地址 / 数据位输出到 SDA 引脚时，如果一个主器件在 SDA 上输出 1（将 SDA 引脚悬空为高电平），而另一个主器件输出 0，就会发生总线仲裁。如果 SDA 引脚上期望的数据是 1，而实际采样到的数据是 0，则发生了总线冲突。主器件将把总线冲突中断标志位 BCLIF 置 1，并将 I²C 端口复位到空闲状态 （图 18-27）。

如果在发送过程中发生总线冲突，则发送操作停止，BF 标志位被清零， SDA 和 SCL 线被拉高，并且可写入 SSPBUF。当执行总线冲突中断服务程序时，如果 I²C 总线空闲，用户可通过发出启动条件恢复通信。

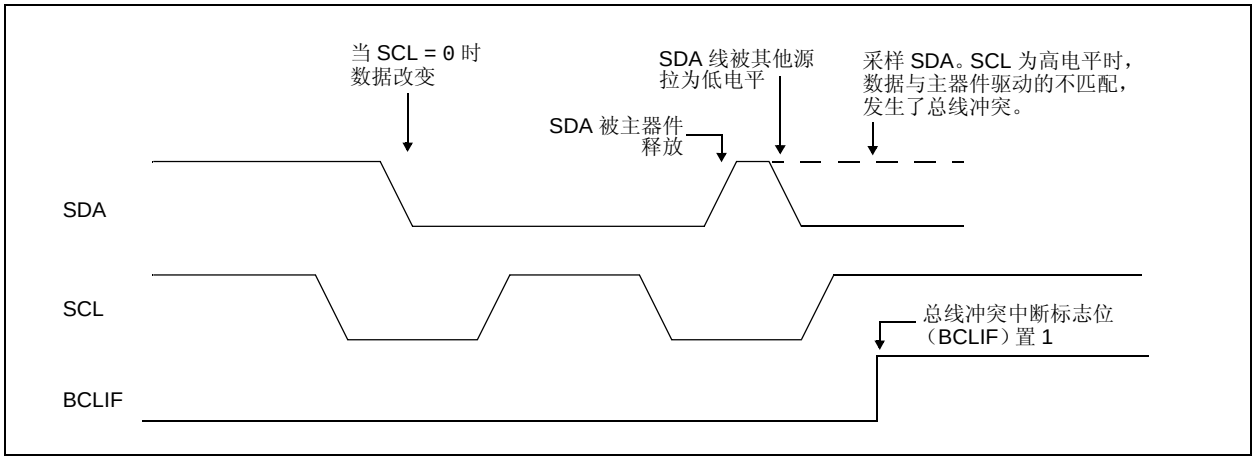
如果在启动、重复启动、停止或应答信号的执行过程中发生总线冲突，则这种状态被中止，SDA 和 SCL 线被拉高， SSPCON2 寄存器中的对应控制位清零。当执行完总线冲突中断服务程序后，如果 I²C 总线空闲，用户可通过发出启动条件恢复通信。

主器件将继续监视 SDA 和 SCL 引脚。一旦出现停止条件， SSPIF 位将被置 1。

发生总线冲突时无论发送的进度如何，写入 SSPBUF 都会从第一个数据位开始发送数据。

在多主模式下，通过在检测到启动条件和停止条件时产生中断可以确定总线何时空闲。 SSPSTAT 寄存器中的 P 位置 1 时，可以取得 I²C 总线的控制权；或者，总线处于空闲状态， S 位和 P 位都清零。

图 18-27： 发送和应答时的总线冲突时序



18.4.17.1 启动条件期间的总线冲突

启动条件期间，以下事件将导致总线冲突：

a) 在启动条件开始时， SDA 或 SCL 被采样为低电平 （图 18-28）。

b) SDA被拉低之前，SCL采样为低电平（图18-29）。

在启动条件期间， SDA 和 SCL 引脚都会被监视。

如果 SDA 引脚已经是低电平，或 SCL 引脚已经是低电平，则：

- 中止启动条件，
- BCLIF 标志位置 1，并且
- MSSP 模块复位为空闲状态 （图 18-28）。

启动条件从 SDA 和 SCL 引脚被拉高开始。当 SDA 引脚采样为高电平时，波特率发生器装入 SSPADD<6:0> 的值并递减计数至 0。如果在 SDA 为高电平时，SCL 引脚采样为低电平，则发生了总线冲突，因为这表示另一个主器件在启动条件期间试图驱动一个数据 1。

如果 SDA 引脚在该计数周期内采样为低电平，则 BRG 复位，同时 SDA 线保持原值（图 18-30）。但是，如果 SDA 引脚采样为 1，则在 BRG 计数结束时该引脚将被置为低电平。接着，波特率发生器被重载并递减计数至 0。在此期间，如果 SCL 引脚采样到 0，则不会发生总线冲突。在BRG计数结束时，SCL引脚被拉为低电平。

注：	在启动条件期间不太可能发生总线冲突，因为两个总线主器件不可能精确地在同一时刻发出启动条件。因此一个主器件将总是先于另一个主器件将 SDA 拉低。但是上述情况不会引起总线冲突，因为两个主器件一定会对启动条件后的第一个地址进行仲裁。如果地址是相同的，必须继续对数据部分、重复启动条件或停止条件进行仲裁。

图 18-28： 启动条件期间的总线冲突 （仅 SDA）

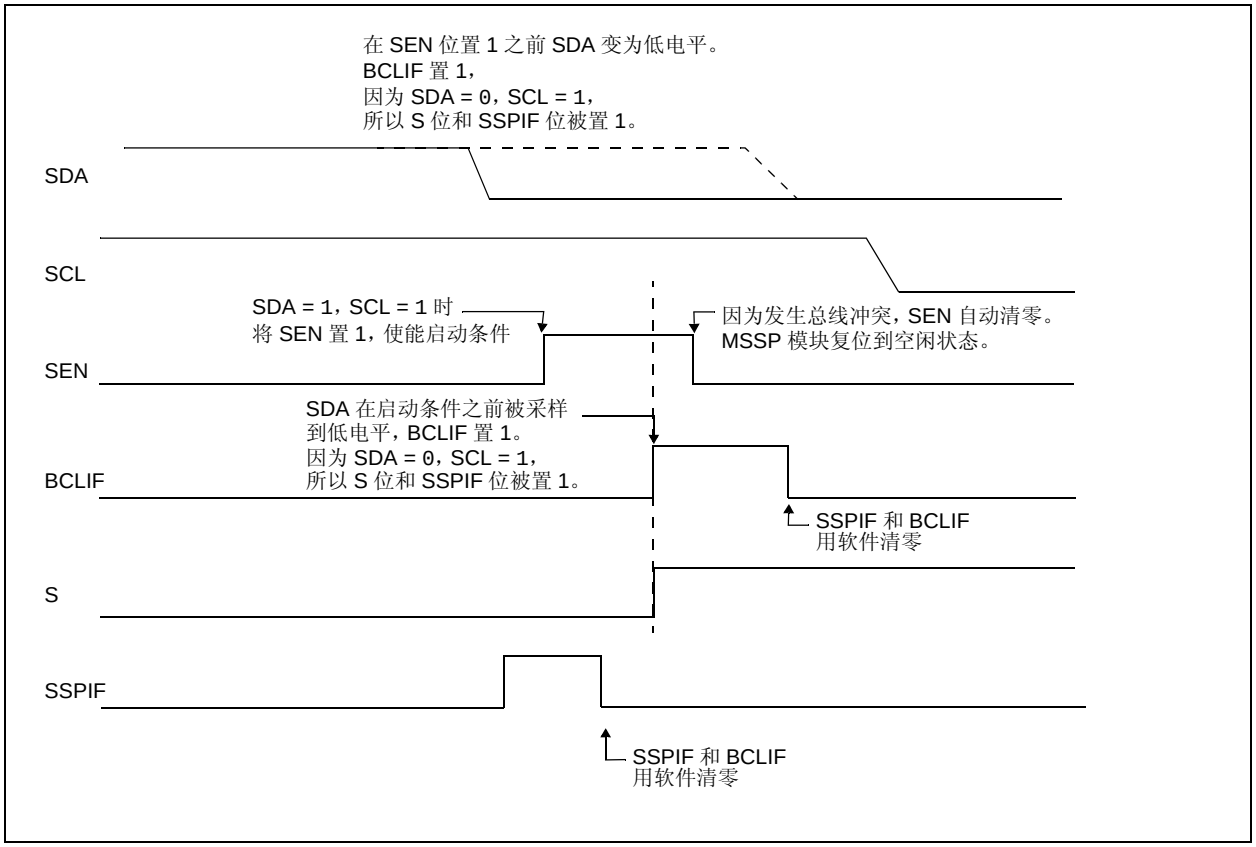


图 18-29：　　启动条件期间的总线冲突 （SCL = 0）

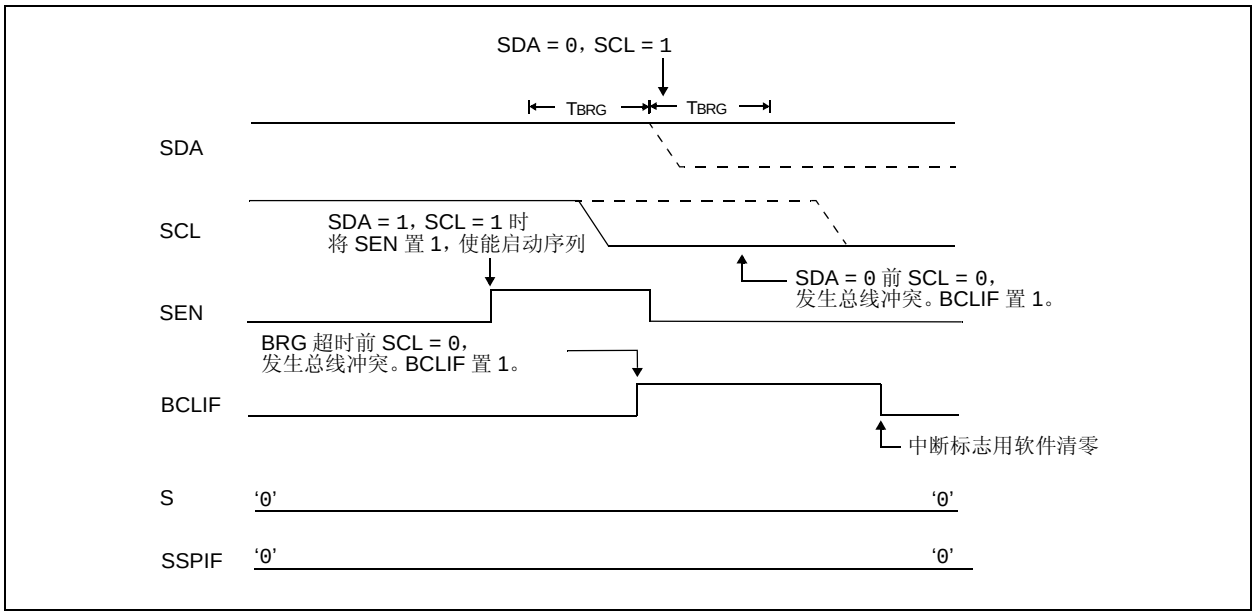
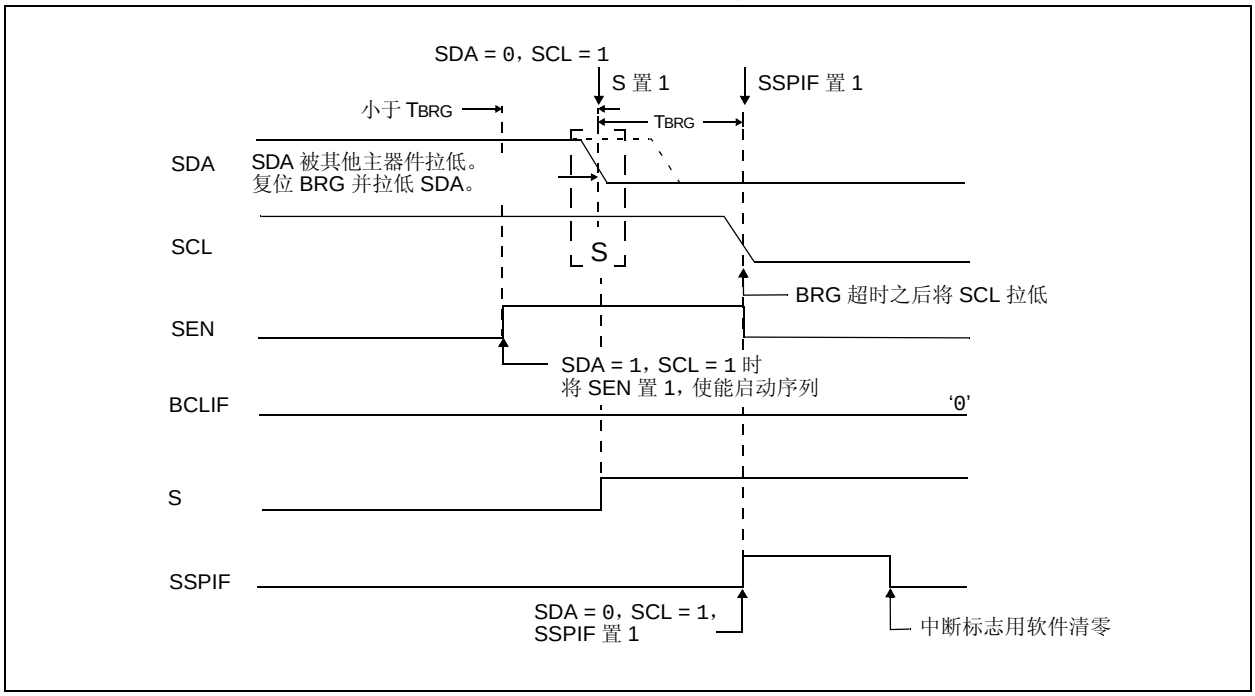


图 18-30：　　启动条件期间由 SDA 仲裁引起的 BRG 复位



18.4.17.2 重复启动条件期间的总线冲突

在以下情况中，重复启动条件期间会发生总线冲突：

a) 在 SCL 由低电平变为高电平期间，在 SDA 上采样到低电平。

b) 在 SDA 被拉为低电平之前， SCL 变为低电平，表示另一个主器件正试图发送一个数据 1。

当用户拉高 SDA 并允许该引脚悬空时， BRG 装入 SSPADD<6:0> 中的值并递减计数至 0。接着 SCL 引脚被拉高，当 SCL 引脚采样到高电平时，对 SDA 引脚进行采样。

如果 SDA 为低电平，则已发生了总线冲突（即，另一个主器件正试图发送一个数据 0，见图 18-31）。如果 SDA 被采样到高电平，则 BRG 被重新装入值并开始计数。如果 SDA 在 BRG 超时之前从高电平变为低电平，则不会发生总线冲突，因为两个主器件不可能精确地在同一时刻将 SDA 拉低。

如果 SCL 在 BRG 超时之前从高电平变为低电平，且 SDA 尚未被拉低，那么将发生总线冲突。在此情况下，另一个主器件在重复启动条件期间正试图发送一个数据 1（见图 18-32）。

如果在 BRG 超时结束时 SCL 和 SDA 都仍然是高电平，则 SDA 引脚被拉低， BRG 重新装入值并开始计数。在计数结束时，不管 SCL 引脚的状态如何，SCL 引脚都被拉低，重复启动条件结束。

图 18-31： 重复启动条件期间的总线冲突 （情形 1）

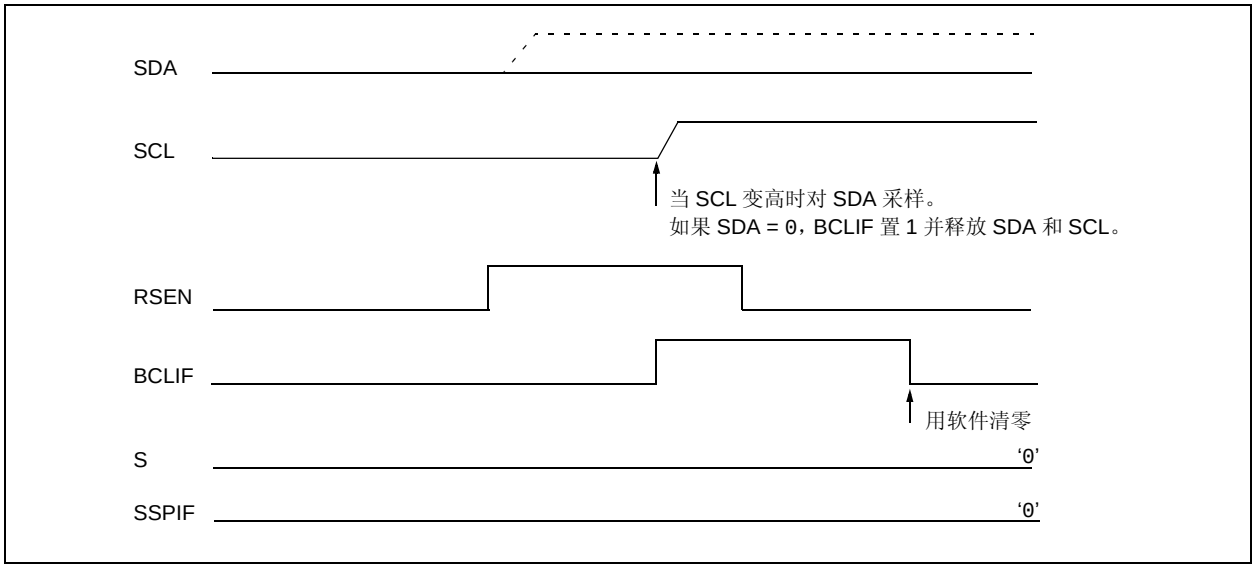
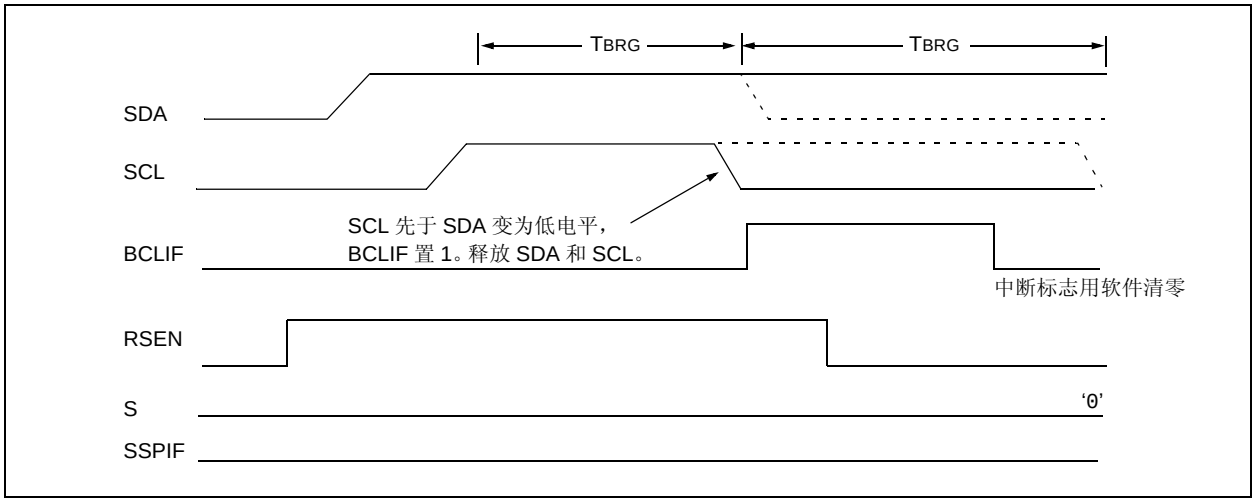


图 18-32： 重复启动条件期间的总线冲突 （情形 2）



18.4.17.3 停止条件期间的总线冲突

以下事件会导致停止条件期间发生总线冲突：

a) SDA已被拉高并允许悬空为高电平之后，SDA在BRG 超时后被采样到低电平。

b) SCL 引脚被拉高之后， SCL 在 SDA 变成高电平之前被采样到低电平。

停止条件从 SDA 被置成低电平开始。当 SDA 采样为低电平时， SCL 引脚被允许悬空。当 SDA 被采样到高电平（时钟仲裁）时，波特率发生器装入 SSPADD<6:0> 的值并递减计数至 0。 BRG 超时后， SDA 被采样。如果 SDA 采样为低电平，则已发生总线冲突。这是因为另一个主器件正试图发送一个数据 0（图 18-33）。如果 SCL 引脚在允许 SDA 悬空为高电平前被采样到低电平，也会发生总线冲突。这是另一个主器件正试图发送一个数据 0 的另外一种情况 （图 18-34）。

图 18-33： 停止条件期间的总线冲突 （情形 1）

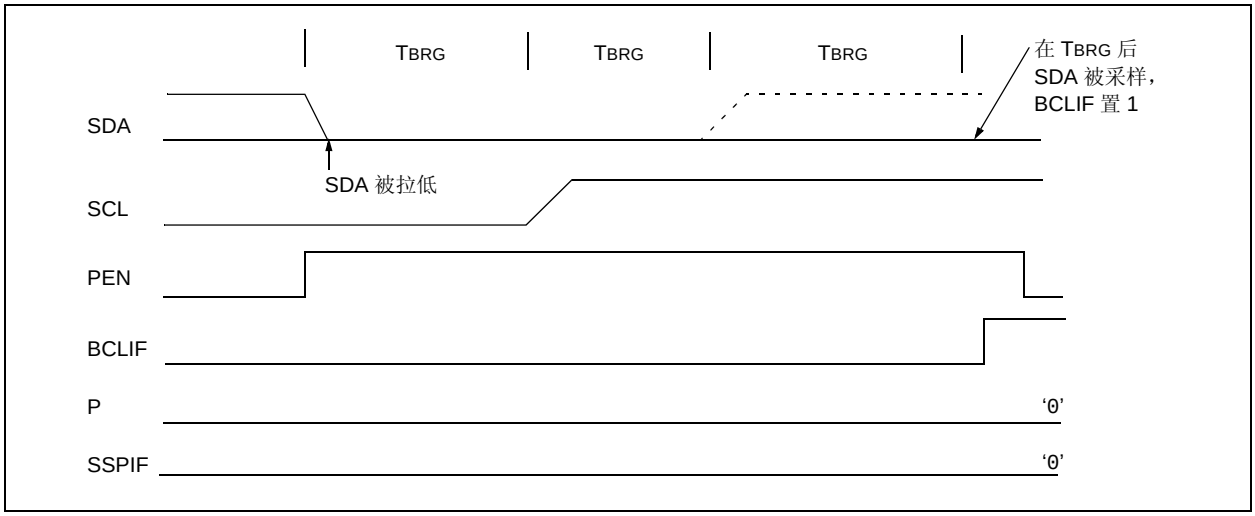


图 18-34： 停止条件期间的总线冲突 （情形 2）

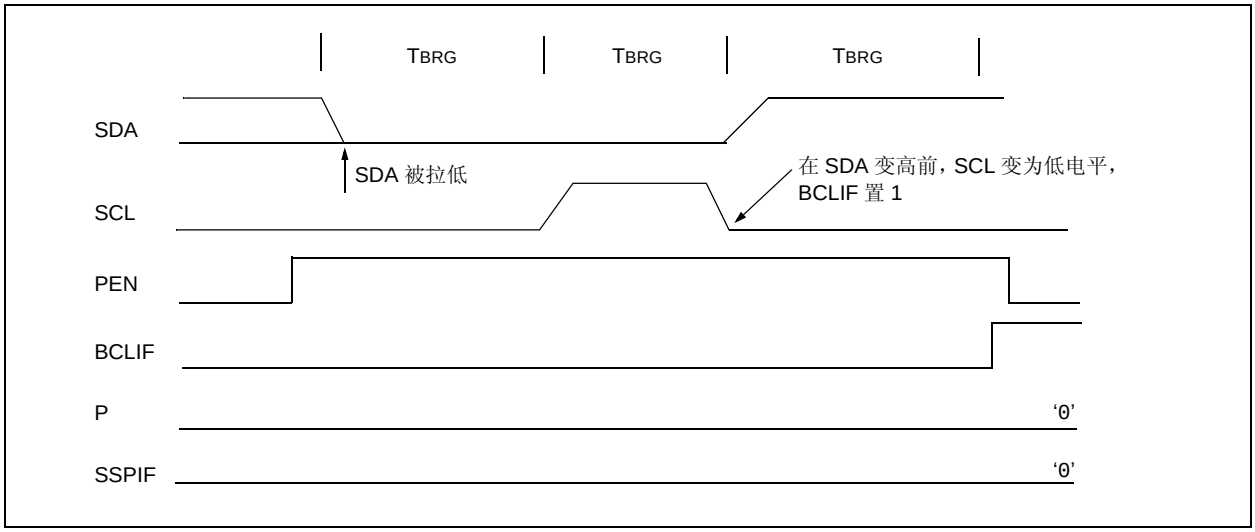


表 18-4：　　与 I²C™ 操作相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	62
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	62
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	62
PIR2	OSCFIF	CMIF	—	—	BCLIF	LVDIF	TMR3IF	—	62
PIE2	OSCFIE	CMIE	—	—	BCLIE	LVDIE	TMR3IE	—	62
IPR2	OSCFIP	CMIP	—	—	BCLIP	LVDIP	TMR3IP	—	62
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	62
SSPBUF	MSSP 接收缓冲 / 发送寄存器								60
SSPADD	MSSP 地址寄存器（I²C™ 从模式）， MSSP 波特率重载寄存器（I²C 主模式）								60
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	60
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	60
	GCEN	ACKSTAT	ADMSK5(1)	ADMSK4(1)	ADMSK3(1)	ADMSK2(1)	ADMSK1(1)	SEN	
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	60

图注：　　— = 未实现，读为 0。I²C™ 模式下的 MSSP 模块不使用阴影单元。

注　1： 仅在 I²C 从模式操作中使用的备用位定义。

19.0 增强型通用同步 / 异步收发器（EUSART）

PIC18F87J90 系列器件有三个串行 I/O 模块：一个是在上一章中讨论过的 MSSP 模块，另外还有两个通用同步 / 异步收发器（Universal Synchronous Asynchronous Receiver Transmitter，USART）模块。（通常 USART 也称为 “ 串行通信接口 ” 或 SCI。）可以将 USART 配置为能与 CRT 终端和个人计算机等外设通信的全双工异步系统。也可以将它配置成能够与 A/D 或 D/A 集成电路、串行 EEPROM 等外设通信的半双工同步系统。

USART 模块在这些器件中有两种不同的实现方式：一种是这里讨论的增强型 USART（EUSART），另一种是下一章中要讨论的可寻址 USART（AUSART）。对于本系列器件，USART1 指的是 EUSART，而 USART2 指的是 AUSART。

EUSART 和 AUSART 模块实现了相同的串行通信核心功能，它们的基本操作也大致相同。EUSART 模块提供了更多的功能，包括自动波特率检测和校准、接收到同步间隔字符时的自动唤醒和 12 位间隔字符发送。因为具有这些特性，所以用在局域互联网总线（LIN/J2602 总线）系统中非常适用。

EUSART 可配置为以下几种工作模式：

- 异步模式（全双工）：
  - 接收到字符时自动唤醒
  - 自动波特率校准
  - 12 位间隔字符发送
- 同步 —— 主模式（半双工），时钟极性可选
- 异步 —— 从模式（半双工），时钟极性可选

EUSART 的引脚与 PORTC（RC6/TX1/CK1/SEG27 和 RC7/RX1/DT1/SEG28）的功能复用。要将这些引脚配置为 EUSART：

- SPEN 位（RCSTA1<7>）必须置 1（= 1）
- TRISC<7> 位必须置 1（= 1）
- TRISC<6> 位必须置 1（= 1）

注：	EUSART 控制根据需要会自动将引脚从输入重新配置为输出。

TX1 输出引脚的驱动器也可以有选择地配置为漏极开路输出。此功能使引脚上的电平可通过外部上拉电阻上拉至较高的电平，并且无需额外的电平转换器就可使输出与外部电路进行通信。

漏极开路输出选项由 U1OD 位（LATG<6>）控制。通过将该位置 1 可将该引脚配置为漏极开路操作。

19.1 控制寄存器

增强型 USART 模块的操作由以下 3 个寄存器控制：

- 发送状态和控制寄存器 1（TXSTA1）
- 接收状态和控制寄存器 1（RCSTA1）
- 波特率控制寄存器 1（BAUDCON1）

这些寄存器如寄存器 19-1、寄存器 19-2 和寄存器 19-3 中所述。

寄存器 19-1： TXSTA1：EUSART 发送状态和控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN(1)	SYNC	SENDB	BRGH	TRMT	TX9D
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 **CSRC**：时钟源选择位

异步模式：
无关位。

同步模式：
1 = 主模式 （时钟由内部 BRG 产生）
0 = 从模式 （时钟来自外部时钟源）

bit 6 **TX9**：9 位发送使能位

1 = 选择 9 位发送
0 = 选择 8 位发送

bit 5 **TXEN**：发送使能位 (1)

1 = 使能发送
0 = 禁止发送

bit 4 **SYNC**：EUSART 模式选择位

1 = 同步模式
0 = 异步模式

bit 3 **SENDB**：发送间隔字符位

异步模式：
1 = 在下一次发送时发送同步间隔字符 （完成时由硬件清零）
0 = 同步间隔字符发送完成

同步模式：
无关位。

bit 2 **BRGH**：高波特率选择位

异步模式：
1 = 高速
0 = 低速

同步模式：
在此模式下未使用。

bit 1 **TRMT**：发送移位寄存器状态位

1 = TSR 空
0 = TSR 满

bit 0 **TX9D**：发送数据的第 9 位

可以是地址 / 数据位或奇偶校验位。

注 1： 在同步模式下， SREN/CREN 可改写 TXEN。

寄存器 19-2： RCSTA1：EUSART 接收状态和控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
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 **SPEN**：串口使能位
1 = 使能串口 （将 RX1/DT1 和 TX1/CK1 引脚配置为串口引脚）
0 = 禁止串口 （保持在复位状态）

bit 6 **RX9**：9 位接收使能位
1 = 选择 9 位接收
0 = 选择 8 位接收

bit 5 **SREN**：单字节接收使能位
异步模式：
无关位。
同步主模式：
1 = 使能单字节接收
0 = 禁止单字节接收
此位在接收完成后清零。
同步从模式：
无关位。

bit 4 **CREN**：连续接收使能位
异步模式：
1 = 使能接收器
0 = 禁止接收器
同步模式：
1 = 使能连续接收，直到使能位 CREN 清零 （CREN 的优先级高于 SREN）
0 = 禁止连续接收

bit 3 **ADDEN**：地址检测使能位
9 位异步模式 （RX9 = 1）：
1 = 当 RSR<8> 置 1 时，使能地址检测，允许中断并装入接收缓冲区
0 = 禁止地址检测，接收所有字节并且第 9 位可作为奇偶校验位
9 位异步模式 （RX9 = 0）：
无关位。

bit 2 **FERR**：帧错误位
1 = 帧错误 （可以通过读 RCREG1 寄存器更新该位并接收下一个有效字节）
0 = 无帧错误

bit 1 **OERR**：溢出错误位
1 = 溢出错误 （可以通过清零 CREN 位来清零该位）
0 = 无溢出错误

bit 0 **RX9D**：接收数据的第 9 位
该位可以是地址 / 数据位或奇偶校验位，并且必须由用户固件计算得到。

寄存器 19-3： BAUDCON1：波特率控制寄存器 1

R/W-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN
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 **ABDOVF**：自动波特率采集计满返回状态位
1 = 在自动波特率检测模式下发生了 BRG 计满返回 （必须用软件清零）
0 = 没有发生 BRG 计满返回

bit 6 **RCIDL**：接收操作空闲状态位
1 = 接收操作处于空闲状态
0 = 接收操作处于工作状态

bit 5 **RXDTP**：接收数据极性选择位
异步模式：
1 = RXx 数据反相
0 = RXx 数据未反相
同步模式：
1 = CKx 时钟反相
0 = CKx 时钟未反相

bit 4 **TXCKP**：时钟和数据极性选择位
异步模式：
1 = TXx 数据反相
0 = TXx 数据未反相
同步模式：
1 = CKx 时钟反相
0 = CKx 时钟未反相

bit 3 **BRG16**：16 位波特率寄存器使能位
1 = 16 位波特率发生器 ——SPBRGH1 和 SPBRG1
0 = 8 位波特率发生器 —— 仅 SPBRG1 （兼容模式），忽略 SPBRGH1 的值

bit 2 **未实现**：读为 0

bit 1 **WUE**：唤醒使能位
异步模式：
1 = EUSART 将继续采样 RX1 引脚 —— 中断在下降沿产生，在下一个上升沿由硬件清零该位
0 = 未监视 RX1 引脚或检测到了上升沿
同步模式：
在此模式下未使用。

bit 0 **ABDEN**：自动波特率检测使能位
异步模式：
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
同步模式：
在此模式下未使用。

19.2 EUSART 波特率发生器 （BRG）

BRG 是一个专用的 8 位或 16 位发生器，支持 EUSART 的异步和同步模式。默认情况下，BRG 工作在 8 位模式下，通过将 BRG16 位（BAUDCON1<3>）置 1 可以选择 16 位模式。

SPBRGH1:SPBRG1 寄存器对控制自由运行定时器的周期。在异步模式下， BRGH （TXSTA1<2>）和 BRG16（BAUDCON1<3>）位也用于控制波特率。在同步模式下， BRGH 位会被忽略。表 19-1 所示为不同 EUSART 模式的波特率计算公式，但仅适用于主模式 （由内部产生时钟信号）。

给出期望的波特率和 Fosc 值，就可以使用表 19-1 中的公式计算 SPBRGH1:SPBRG1 寄存器的最近似整数值。这样就可以确定波特率误差。例 19-1 给出了计算示例。表 19-2 给出了不同异步模式下典型的波特率和误差值。

使用高波特率 （BRGH = 1）或 16 位 BRG 有助于降低波特率误差，或者在快速振荡器频率下取得较缓慢的波特率。

将新值写入 SPBRGH1:SPBRG1 寄存器会导致 BRG 定时器复位 （或清零）。这可以确保 BRG 无需等待定时器溢出就可以输出新的波特率。

19.2.1 在功耗管理模式下的操作

器件时钟用于产生所需的波特率。当进入一种功耗管理模式时，新时钟源可能会工作在一个不同的频率下。这可能需要调整 SPBRG1 寄存器对中的值。

19.2.2 采样

择多检测电路对 RX1 引脚采样三次，以判定 RX1 引脚上出现的是高电平还是低电平。

表 19-1： 波特率公式

配置位			BRG/EUSART 模式	波特率公式
SYNC	BRG16	BRGH		
0	0	0	8 位 / 异步	Fosc/[64 (n + 1)]
0	0	1	8 位 / 异步	Fosc/[16 (n + 1)]
0	1	0	16 位 / 异步	
0	1	1	16 位 / 异步	Fosc/[4 (n + 1)]
1	0	x	8 位 / 同步	
1	1	x	16 位 / 同步	

图注： x = 无关位， n = SPBRGH1:SPBRG1 寄存器对的值

例 19-1： 计算波特率误差

器件工作在 Fosc = 16 MHz，目标波特率 = 9600，异步模式， 8 位 BRG：
目标波特率 = Fosc/(64 ([SPBRGH1:SPBRG1] + 1))
求解 SPBRGH1:SPBRG1：
X = ((Fosc/ 目标波特率)/64) – 1
= ((16000000/9600)/64) – 1
= [25.042] = 25
计算波特率 = 16000000/(64 (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%

表 19-2： 与波特率发生器相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	61
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	61
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	63
SPBRGH1	EUSART 波特率发生器寄存器的高字节								63
SPBRG1	EUSART 波特率发生器寄存器的低字节								61

图注： — = 未实现，读为 0。 BRG 不使用阴影单元。

表 19-3： 异步模式下的波特率

波特率（K）	SYNC = 0， BRGH = 0， BRG16 = 0											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	1.221	1.73	255	1.202	0.16	129	1.201	-0.16	103
2.4	2.441	1.73	255	2.404	0.16	129	2.404	0.16	64	2.403	-0.16	51
9.6	9.615	0.16	64	9.766	1.73	31	9.766	1.73	15	9.615	-0.16	12
19.2	19.531	1.73	31	19.531	1.73	15	19.531	1.73	7	—	—	—
57.6	56.818	-1.36	10	62.500	8.51	4	52.083	-9.58	2	—	—	—
115.2	125.000	8.51	4	104.167	-9.58	2	78.125	-32.18	1	—	—	—

波特率（K）	SYNC = 0， BRGH = 0， BRG16 = 0								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）
0.3	0.300	0.16	207	0.300	-0.16	103	0.300	-0.16	51
1.2	1.202	0.16	51	1.201	-0.16	25	1.201	-0.16	12
2.4	2.404	0.16	25	2.403	-0.16	12	—	—	—
9.6	8.929	-6.99	6	—	—	—	—	—	—
19.2	20.833	8.51	2	—	—	—	—	—	—
57.6	62.500	8.51	0	—	—	—	—	—	—
115.2	62.500	-45.75	0	—	—	—	—	—	—

波特率（K）	SYNC = 0， BRGH = 1， BRG16 = 0											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	—	—	—	—	—	—	—	—	—
2.4	—	—	—	—	—	—	2.441	1.73	255	2.403	-0.16	207
9.6	9.766	1.73	255	9.615	0.16	129	9.615	0.16	64	9.615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19.230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55.555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

波特率（K）	SYNC = 0， BRGH = 1， BRG16 = 0								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）
0.3	—	—	—	—	—	—	0.300	-0.16	207
1.2	1.202	0.16	207	1.201	-0.16	103	1.201	-0.16	51
2.4	2.404	0.16	103	2.403	-0.16	51	2.403	-0.16	25
9.6	9.615	0.16	25	9.615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

表 19-3： 异步模式下的波特率 （续）

波特率（K）	SYNC = 0， BRGH = 0， BRG16 = 1											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）
0.3	0.300	0.00	8332	0.300	0.02	4165	0.300	0.02	2082	0.300	-0.04	1665
1.2	1.200	0.02	2082	1.200	-0.03	1041	1.200	-0.03	520	1.201	-0.16	415
2.4	2.402	0.06	1040	2.399	-0.03	520	2.404	0.16	259	2.403	-0.16	207
9.6	9.615	0.16	259	9.615	0.16	129	9.615	0.16	64	9.615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19.230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55.555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

波特率（K）	SYNC = 0， BRGH = 0， BRG16 = 1								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）
0.3	0.300	0.04	832	0.300	-0.16	415	0.300	-0.16	207
1.2	1.202	0.16	207	1.201	-0.16	103	1.201	-0.16	51
2.4	2.404	0.16	103	2.403	-0.16	51	2.403	-0.16	25
9.6	9.615	0.16	25	9.615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

波特率（K）	SYNC = 0， BRGH = 1， BRG16 = 1 或 SYNC = 1， BRG16 = 1											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）
0.3	0.300	0.00	33332	0.300	0.00	16665	0.300	0.00	8332	0.300	-0.01	6665
1.2	1.200	0.00	8332	1.200	0.02	4165	1.200	0.02	2082	1.200	-0.04	1665
2.4	2.400	0.02	4165	2.400	0.02	2082	2.402	0.06	1040	2.400	-0.04	832
9.6	9.606	0.06	1040	9.596	-0.03	520	9.615	0.16	259	9.615	-0.16	207
19.2	19.193	-0.03	520	19.231	0.16	259	19.231	0.16	129	19.230	-0.16	103
57.6	57.803	0.35	172	57.471	-0.22	86	58.140	0.94	42	57.142	0.79	34
115.2	114.943	-0.22	86	116.279	0.94	42	113.636	-1.36	21	117.647	-2.12	16

波特率（K）	SYNC = 0， BRGH = 1， BRG16 = 1 或 SYNC = 1， BRG16 = 1								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）
0.3	0.300	0.01	3332	0.300	-0.04	1665	0.300	-0.04	832
1.2	1.200	0.04	832	1.201	-0.16	415	1.201	-0.16	207
2.4	2.404	0.16	415	2.403	-0.16	207	2.403	-0.16	103
9.6	9.615	0.16	103	9.615	-0.16	51	9.615	-0.16	25
19.2	19.231	0.16	51	19.230	-0.16	25	19.230	-0.16	12
57.6	58.824	2.12	16	55.555	3.55	8	—	—	—
115.2	111.111	-3.55	8	—	—	—	—	—	—

19.2.3 自动波特率检测

增强型 USART 模块支持波特率自动检测和校准。此功能仅在异步模式下当 WUE 位清零时有效。

只要接收到启动位并且 ABDEN 位已置 1，就会开始自动波特率测量序列（图 19-1）。波特率计算采用自平均的方式。

在自动波特率检测 （ABD）模式下， BRG 的时钟信号反向。 BRG 并不为进入的 RX1 信号提供时钟信号，而是由 RX1 信号为 BRG 定时。在 ABD 模式下，内部波特率发生器被用作计数器来计算输入的串行字节流的位间隔时间。

一旦 ABDEN 位置 1，状态机就会清零 BRG 并寻找启动位。为了正确计算比特率，自动波特率检测必须接收到一个值为 55h （ASCII 字符 U，也是 LIN/J2602 总线的同步字符）的字节。为了尽量减少输入信号不对称造成的影响，测量时间段内要包含一个高位和一个低位时间。在启动位后， SPBRG1 使用预先选择的时钟源在 RX1 的第一个上升沿开始计数。在 RX1 引脚传输了 8 个位，或在检测到第 5 个上升沿后，会将相应 BRG 周期内的累加值保存在SPBRGH1:SPBRG1寄存器对中。当第 5 个时钟边沿出现时（应与停止位对应），ABDEN 位会自动清零。

如果发生了 BRG 计满返回（从 FFFFh 溢出到 0000h），会在 ABDOVF 状态位 （BAUDCON1<7>）有所反映。该位可在 BRG 计满返回时由硬件置 1，也可以由用户通过软件置 1 或清零。在发生计满返回事件后，ABD 模式继续有效， ABDEN 位保持置 1 （图 19-2）。

在校准波特率周期时，BRG寄存器时钟频率为预配置时钟频率的 1/8。请注意 BRG 时钟可由 BRG16 和 BRGH 位配置。必须将 BRG16 位置 1，才能将 SPBRG1 和 SPBRGH1 用作 16 位计数器。用户通过检查 SPBRGH1 寄存器的值是否为 00h，可以验证 8 位模式下是否发生了进位。表 19-4 所示为 BRG 计数器的时钟速率。

当发生 ABD 序列时，EUSART 状态机保持在空闲状态。一旦在 RX1 上检测到第 5 个上升沿，中断标志位 RC1IF 就会置 1。需要读取 RCREG1 中的值，来清零中断标志位 RC1IF。应丢弃 RCREG1 的值。

注	**1：** 如果 WUE 位和 ABDEN 位都置 1，自动波特率检测将在间隔字符*之后*的字节开始。
	2： 需要由用户来判断输入字符的波特率是否处于所选 BRG 时钟源范围内。由于位误差率的原因，某些振荡器频率和 EUSART 波特率的组合是无法实现的。使用自动波特率检测功能时，必须综合考虑系统总的时序和通信波特率。

表 19-4： BRG 计数器时钟速率

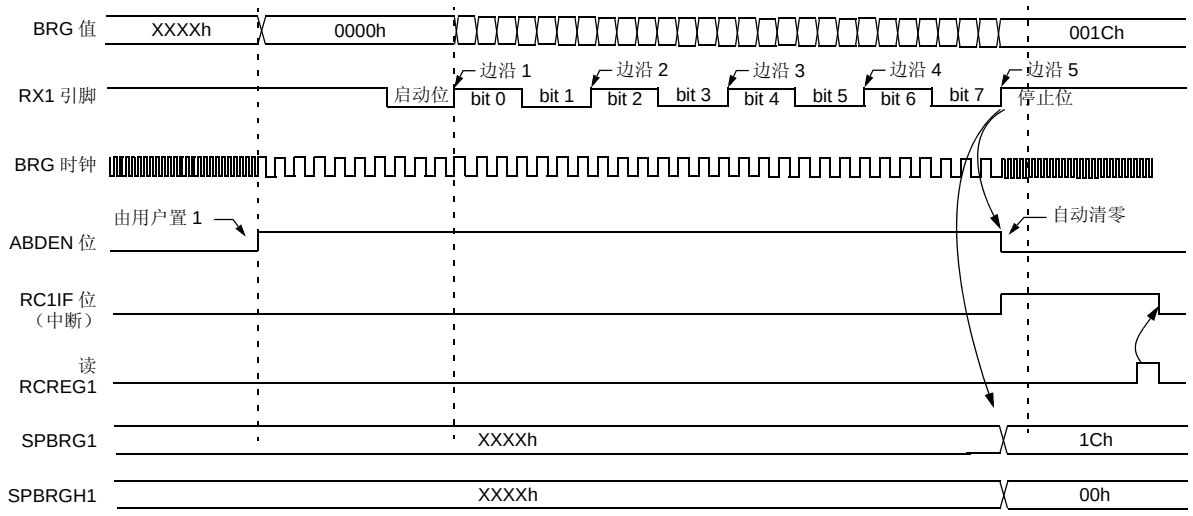
BRG16	BRGH	BRG 计数器时钟
0	0	Fosc/512
0	1	Fosc/128
1	0	Fosc/128
1	1	Fosc/32

注： 在 ABD 序列期间，SPBRG1 和 SPBRGH1 都被用作 16 位计数器，与 BRG16 的设置无关。

19.2.3.1 ABD 和 EUSART 发送

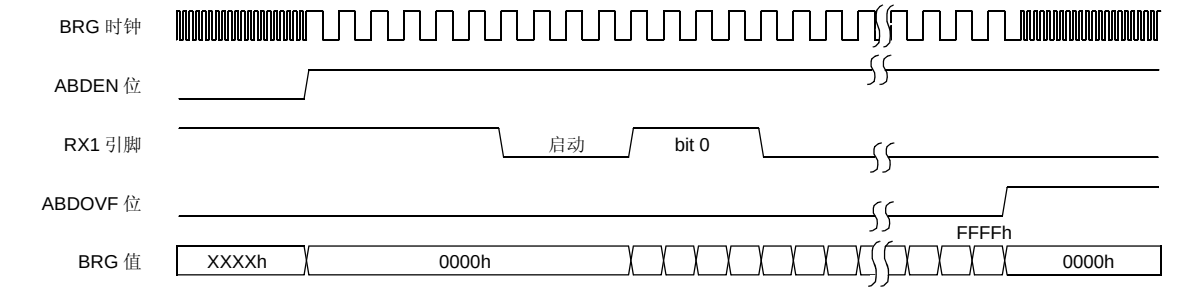
由于 ABD 采集期间 BRG 时钟是反相的，因此在 ABD 期间不能使用 EUSART 发送器。这意味着只要 ABDEN 位置 1，就不能写入 TXREG1。用户还应确保在发送序列期间 ABDEN 不能为置 1 状态，否则可能会导致无法预料的 EUSART 操作。

图 19-1：　　自动波特率计算



注：　需要将 EUSART 模块配置为异步模式并且 WUE = 0，才能进行 ABD 序列。

图 19-2：　　BRG 溢出序列



19.3 EUSART 异步模式

通过将 SYNC 位 （TXSTA1<4>）清零可选择异步工作模式。在此模式下， EUSART 使用标准的不归零码 （Non-Return-to-Zero，NRZ）格式（1 个启动位、8 个或 9 个数据位和 1 个停止位）。最常见的数据格式为 8 位。片上专用 8 位 /16 位波特率发生器可用于从振荡器产生标准波特率频率。

EUSART 先发送和接收 LSb。EUSART 的发送器和接收器在功能上是相互独立的，但采用相同的数据格式和波特率。根据 BRGH 和 BRG16 位 （TXSTA1<2> 和 BAUDCON1<3>）的设置值，波特率发生器可以产生两种不同的波特率时钟，频率分别为移位速率的 16 倍或 64 倍。硬件不支持奇偶校验，但可通过软件实现并作为第 9 个数据位存储。

当工作在异步模式下时，EUSART 模块包括以下重要组成部分：

- 波特率发生器
- 采样电路
- 异步发送器
- 异步接收器
- 接收到同步间隔字符时自动唤醒
- 12 位间隔字符发送
- 自动波特率检测

19.3.1 EUSART 异步发送器

图 19-3 给出了 EUSART 发送器框图。发送器的核心是发送 （串行）移位寄存器 （Transmit Shift register，TSR）。移位寄存器从读 / 写发送缓冲寄存器 TXREG1 中获取数据。TXREG1 寄存器中的数据由软件装入。在前一次装入数据的停止位发送前，不会向 TSR 寄存器装入新数据。一旦停止位发送完毕，TXREG1 寄存器中的新数据 （如果有）就会被装入 TSR。

一旦 TXREG1 寄存器向 TSR 寄存器传输了数据 （在 1 个 TCY 内发生），TXREG1 寄存器就为空，同时 TX1IF 标志位（PIR1<4>）置 1。可以通过将中断允许位 TX1IE （PIE1<4>） 置 1 或清零来允许或禁止该中断。不管 TX1IE 的状态如何，只要中断发生，TX1IF 就会置 1 并且不能用软件清零。TX1IF 不会在 TXREG1 装入新数据时立即被清零，而是在装入指令后的第二个指令周期被清零。因此在 TXREG1 装入新数据后立即查询 TX1IF，会得到无效结果。

TX1IF 指示的是 TXREG1 寄存器的状态，而另一个位 TRMT （TXSTA1<1>）则指示 TSR 寄存器的状态。TRMT 是只读位，它在 TSR 寄存器为空时被置 1。TRMT 位与任何中断逻辑均无关联，因此要确定 TSR 寄存器是否为空，用户只能对此位进行查询。

注 1： TSR 寄存器不映射到数据存储器中，因此用户无法使用。

2： 当使能位TXEN置1时，标志位TX1IF置1。

设置异步发送操作的步骤如下：

1. 对 SPBRGH1:SPBRG1 寄存器进行初始化，设置合适的波特率。按需要将 BRGH 和 BRG16 位置 1 或清零，以获得所需的波特率。
2. 通过清零 SYNC 位并将 SPEN 位置 1，使能异步串口。
3. 如果需要中断，将中断允许位 TX1IE 置 1。
4. 如果需要 9 位发送，将发送位 TX9 置 1；可以作为地址 / 数据位使用。
5. 通过将 TXEN 位置 1 使能发送，此操作同时也会将 TX1IF 位置 1。
6. 如果选择了 9 位发送，应将第 9 位装入 TX9D 位。
7. 将数据装入 TXREG1 寄存器 （启动发送）。
8. 如果使用中断，应确保 INTCON 寄存器中的 GIE 和 PEIE 位 （INTCON<7:6>）置 1。

图 19-3： EUSART 发送框图

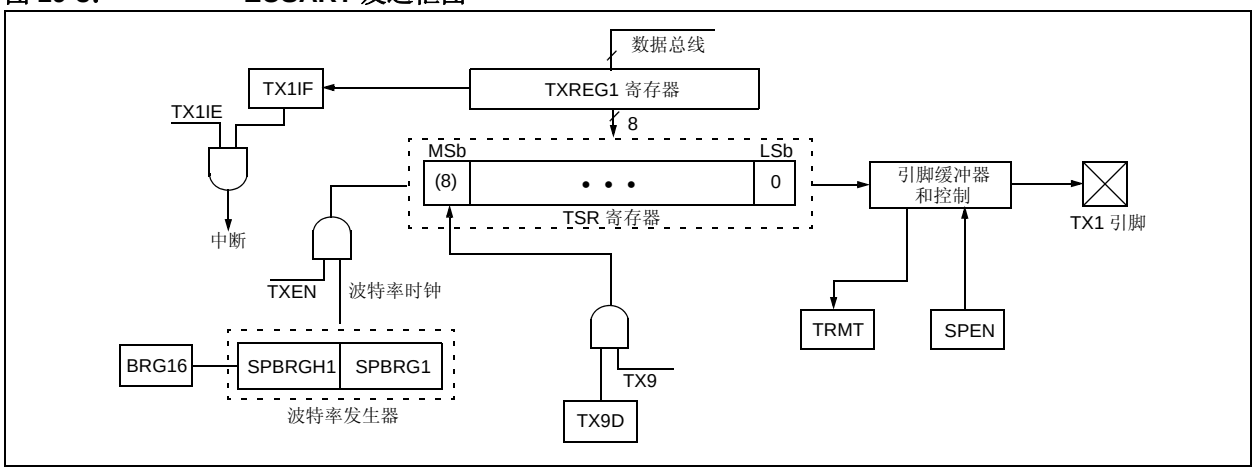


图 19-4：　　异步发送

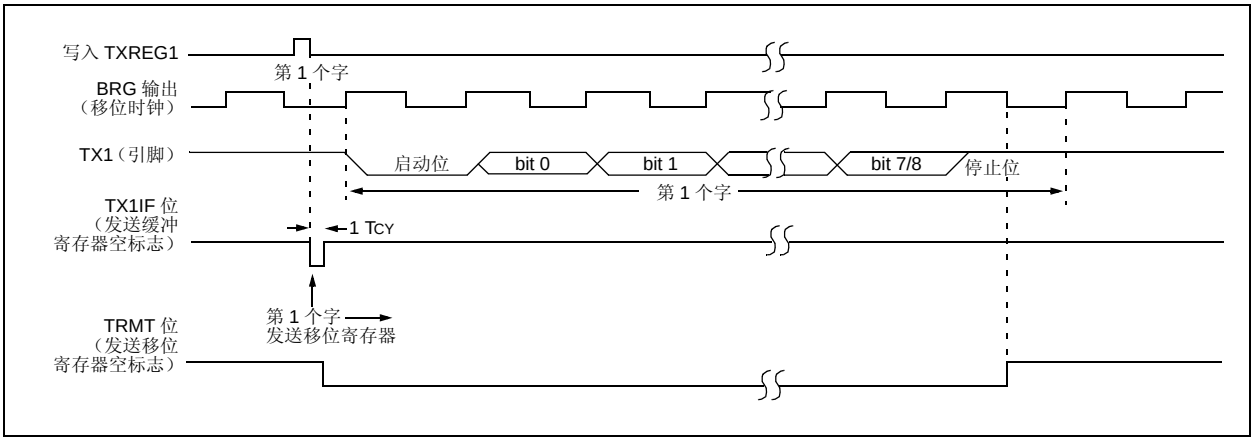
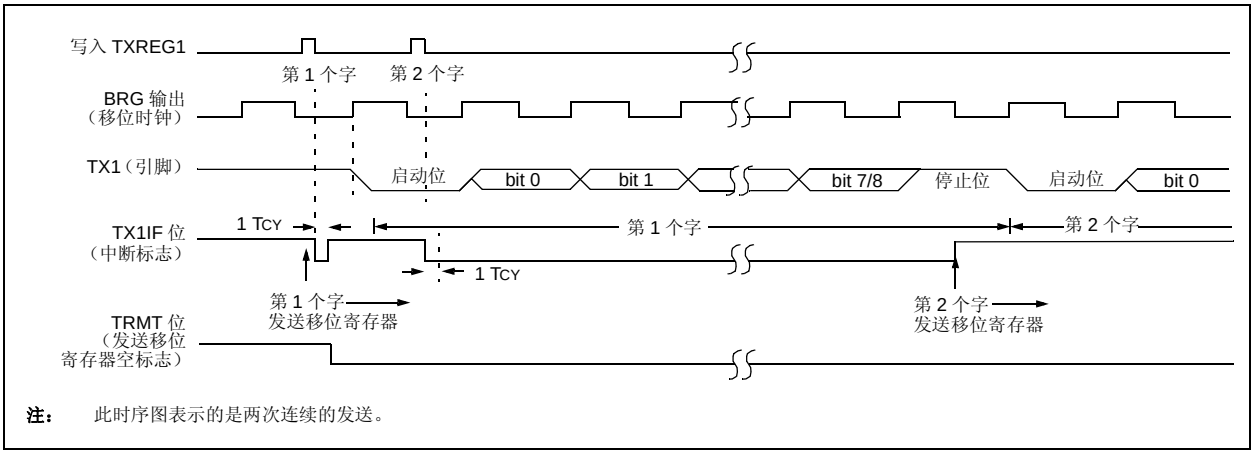


图 19-5：　　异步发送 （背对背）



注：　此时序图表示的是两次连续的发送。

表 19-5：　　与异步发送相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	62
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	62
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	62
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	61
TXREG1	EUSART 发送寄存器								61
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	61
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	63
SPBRGH1	EUSART 波特率发生器寄存器的高字节								63
SPBRG1	EUSART 波特率发生器寄存器的低字节								61
LATG	U2OD	U1OD	—	LATG4	LATG3	LATG2	LATG1	LATG0	62

图注：　— = 未实现位，读为 0。异步发送不使用阴影单元。

19.3.2 EUSART 异步接收器

图 19-6 给出了接收器框图。在 RX1 引脚上接收数据，并驱动数据恢复模块。数据恢复模块实际上是一个工作频率为 16 倍波特率的高速移位器，而主接收串行移位器的工作频率等于比特率或 FOSC。此模式通常用于 RS-232 系统中。

设置异步接收操作的步骤如下：

1. 对 SPBRGH1:SPBRG1 寄存器进行初始化，设置合适的波特率。按需要将 BRGH 和 BRG16 位置 1 或清零，以获得所需的波特率。
2. 通过清零 SYNC 位并将 SPEN 位置 1，使能异步串口。
3. 如果需要中断，将中断允许位 RC1IE 置 1。
4. 如果需要接收 9 位数据，将 RX9 位置 1。
5. 通过将 CREN 位置 1 使能接收。
6. 当接收完成时标志位 RC1IF 将被置 1，此时如果中断允许位RC1IE已置1，则还将产生一个中断。
7. 读 RCSTA1 寄存器获取第 9 位数据 （如果已使能），并判断在接收过程中是否发生了错误。
8. 读 RCREG1 寄存器来读取接收到的 8 位数据。
9. 如果发生错误，将使能位 CREN 清零以清除错误。
10. 如果使用中断，应确保 INTCON 寄存器中的 GIE 和 PEIE 位 （INTCON<7:6>）置 1。

19.3.3 设置带有地址检测功能的 9 位模式

此模式通常用于 RS-485 系统中。设置使能地址检测的异步接收的步骤如下：

1. 对 SPBRGH1:SPBRG1 寄存器进行初始化，设置合适的波特率。按需要将 BRGH 和 BRG16 位置 1 或清零，以获得所需的波特率。
2. 通过清零 SYNC 位并将 SPEN 位置 1，使能异步串口。
3. 如果需要中断，将 RCEN 位置 1 并用 RC1IP 位选择所需的优先级。
4. 将 RX9 位置 1 使能 9 位接收。
5. 将 ADDEN 位置 1 使能地址检测。
6. 将 CREN 位置 1 使能接收。
7. 当接收完成时 RC1IF 位将被置 1。此时如果 RC1IE 和 GIE 位已置 1，还将响应中断。
8. 读 RCSTA1 寄存器判断在接收时是否发生了错误，同时读取第 9 位数据 （如果适用）。
9. 读 RCREG1 判断是否正在对器件进行寻址。
10. 如果发生错误，将 CREN 位清零。
11. 如果已寻址到器件，将 ADDEN 位清零以允许所有接收到的数据被送入接收缓冲区并发送中断信号给 CPU。

图 19-6： EUSART 接收框图

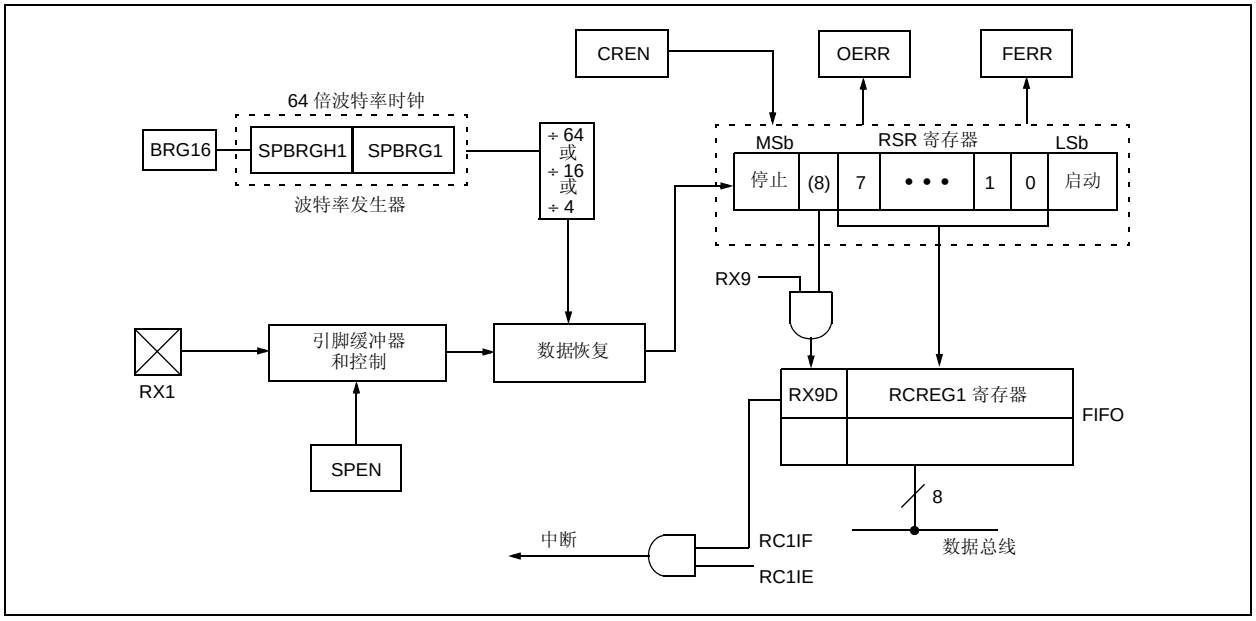
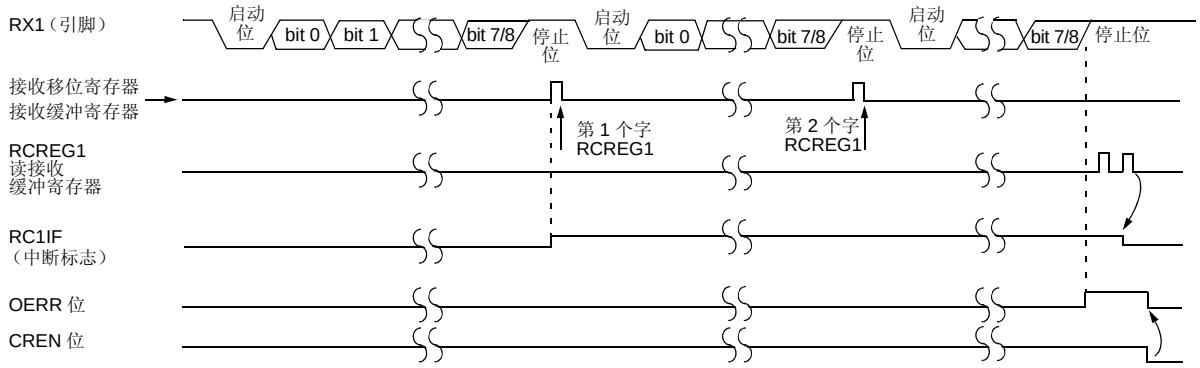


图 19-7：　　异步接收



注：　此时序图显示了在 RX1 输入引脚上接收的 3 个字。在第 3 个字之后读 RCREG1 （接收缓冲寄存器）会使 OERR （溢出）位置 1。

表 19-6：　　与异步接收相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	62
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	62
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	62
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	61
RCREG1	EUSART 接收寄存器								61
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	61
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	63
SPBRGH1	EUSART 波特率发生器寄存器的高字节								63
SPBRG1	EUSART 波特率发生器寄存器的低字节								61

图注：　— = 未实现位，读为 0。异步接收不使用阴影单元。

19.3.4 接收到同步间隔字符时自动唤醒

在休眠模式下， EUSART 的所有时钟都会暂停。因此，波特率发生器处于非激活状态，且无法进行正确的数据接收。自动唤醒功能允许当 RX1/DT1 线上有事件发生时唤醒控制器，该功能需要 EUSART 工作在异步模式下。

通过将 WUE 位 （BAUDCON<1>）置 1，使能自动唤醒功能。该功能启用后，将禁止 RX1/DT1 上的典型接收操作，且 EUSART 保持在空闲状态并监视唤醒事件（不管 CPU 运行模式如何）。唤醒事件是指 RX1/DT1 线上发生高电平到低电平的转换。（这与同步间隔字符的开始或 LIN/J2602 协议的唤醒信号字符一致。）

唤醒事件后，模块产生一个 RC1IF 中断。在正常工作模式下，中断会与 Q 时钟同步产生 （图 19-8）；如果器件处于休眠模式，则两者是不同步的（图 19-9）。通过读 RCREG1 寄存器可清除中断条件。

唤醒事件后，当 RX1 线上出现由低向高的电平转换时，WUE 位自动清零。此时， EUSART 模块将从空闲模式返回正常工作模式。这向用户表明同步间隔事件结束。

19.3.4.1 使用自动唤醒功能的特别注意事项

因为自动唤醒功能是通过检测 RX1/DT1 上的上升沿跳变实现的，所以在停止位前该引脚上任何的状态改变都可能会产生错误的字符结束信号（End-Of-Character，EOF）并导致数据或帧错误。因此，为了确保正常的传输，必须首先发送全 0 字符。对于标准 RS-232 器件，该字符是 00h （8 字节）；而对于 LIN/J2602 总线则是 000h （12 位）。

另外还必须考虑振荡器起振时间，尤其在采用起振时间较长的振荡器 （即， XT 或 HS 模式）应用中更要注意这一点。“同步间隔”（或唤醒信号）字符必须足够长，并且跟有足够长的时间间隔，以便使选定振荡器有充足的时间起振来保证 EUSART 正确初始化。

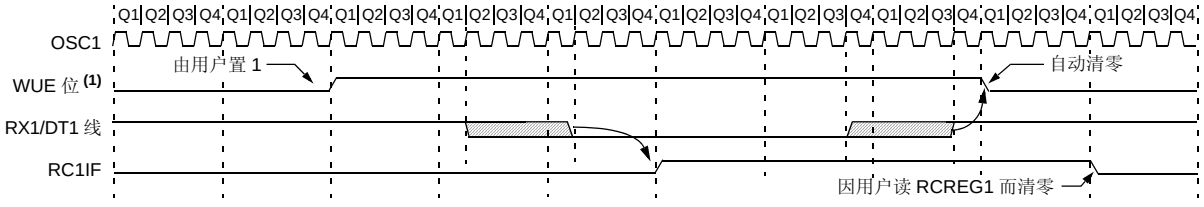
19.3.4.2 使用 WUE 位的特别注意事项

使用 WUE 和 RC1IF 事件的时序来判断接收数据的有效性时，有可能会引起一些混淆。如前所述，将 WUE 位置 1 会使 EUSART 进入空闲状态。唤醒事件会通过将 RC1IF 位置 1 产生一个接收中断。此后当 RX1/DT1 上出现上升沿时， WUE 位被清零。然后通过读 RCREG1 寄存器可清除中断条件。一般情况下，RCREG1 中的数据是无效数据，应该丢弃。

WUE 位清零 （或仍然置 1）同时 RC1IF 标志位置 1 并不能表明 RCREG1 中接收的数据是完整的。用户应考虑在固件中同时验证是否完整地接收了数据。

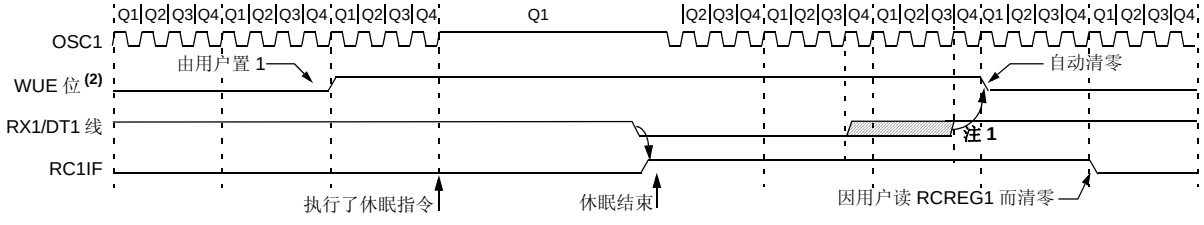
要确保没有丢失有效数据，应检查 RCIDL 位来验证是否还在接收数据。如果未发生接收操作，可在进入休眠模式前将 WUE 位置 1。

图 19-8： 正常工作模式下的自动唤醒位 （WUE）时序



注 1： WUE 位置 1 时， EUSART 保持在空闲状态。

图 19-9： 休眠模式下的自动唤醒位 （WUE）时序



注 1： 如果唤醒事件需要较长的振荡器预热时间，在 *stposc* 信号仍然有效时会发生 WUE 位自动清零。该时序与 Q 时钟出现与否无关。

2： WUE 位置 1 时， EUSART 保持在空闲状态。

19.3.5 间隔字符序列

增强型 EUSART 模块能够发送符合 LIN/J2602 总线标准的特殊间隔字符序列。发送的间隔字符包含 1 个启动位，随后的 12 个 0 位和 1 个停止位。当发送移位寄存器装有数据时，只要 SENDB 和 TXEN 位（TXSTA<3> 和 TXSTA<5>）置 1，就会发送帧间隔字符。请注意写入 TXREG1 的数据值会被忽略，并会发送全 0。

在发送了相应的停止位后，硬件会自动将 SENDB 位复位。这样用户可以在间隔字符（在 LIN/J2602 规范中通常是同步字符）后预先将下一个要发送字节装入发送 FIFO。

请注意在发送间隔字符时写入TXREG1的数据值会被忽略。写入仅仅是为了启动正确的序列。

正如在正常发送操作中一样，TRMT 位表明发送正在进行还是处于空闲状态。关于发送间隔字符的时序，请参见图 19-10。

19.3.5.1 间隔和同步发送序列

以下序列将发送一个报文帧头，它由一个间隔字符和其后的自动波特率同步字节组成。这是 LIN/J2602 总线主器件的典型序列。

1. 将 EUSART 配置为所需的模式。
2. 将 TXEN 和 SENDB 位置 1 以设置间隔字符。
3. 将无效字符装入 TXREG1，启动发送（该值会被忽略）。
4. 将55h写入TXREG1，以便把同步字符装入FIFO缓冲区。
5. 发送间隔字符后，SENDB 位被硬件复位。此时，同步字符会以预先配置的模式发送。

当 TXREG1 为空时（由 TX1IF 指示），下一个数据字节会写入 TXREG1。

19.3.6 接收间隔字符

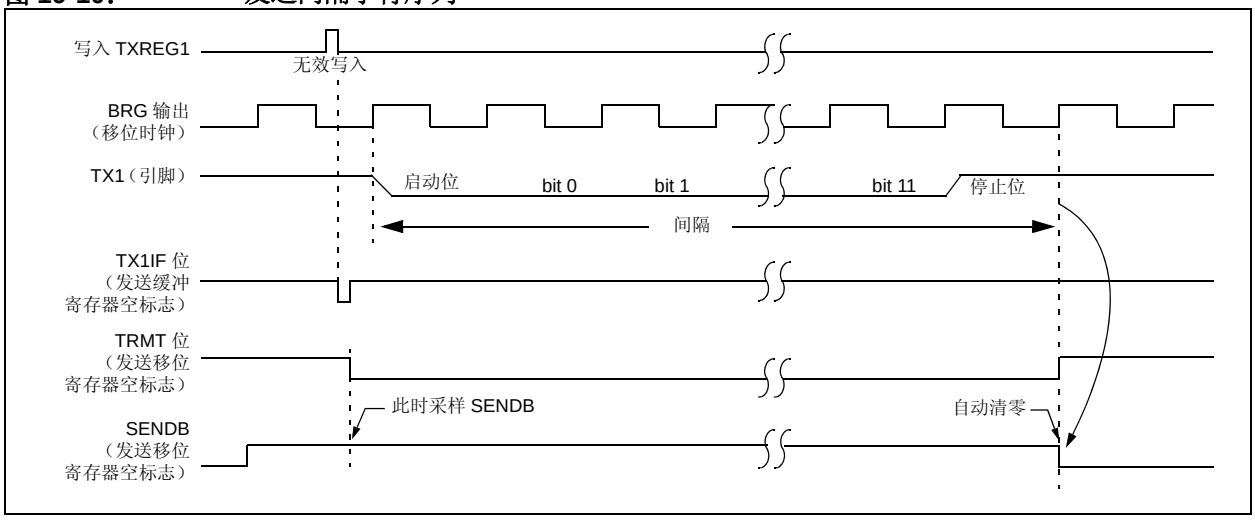
增强型 USART 模块接收间隔字符有两种方法。

第一种方法是强制将波特率配置为典型速度的 9/13。这可以使停止位在正确的采样点（对于间隔字符为启动位之后的 13 位，对于典型数据则是 8 个数据位）跳变。

第二种方法采用**第 19.3.4 节 “ 接收到同步间隔字符时自动唤醒 ”** 中所述的自动唤醒功能。通过使能此功能，EUSART 将采样 RX1/DT1 线的下两次跳变，产生一个 RC1IF 中断，接收下一个数据字节，并在随后产生另一个中断。

请注意在间隔字符后，用户通常希望使能自动波特率检测功能。无论使用哪种方法，用户都可以在检测到 TX1IF 中断时立即将 ABD 位置 1。

图 19-10： 发送间隔字符序列



19.4 EUSART 同步主模式

将 CSRC 位（TXSTA<7>）置 1 可以进入同步主模式。在此模式下，数据以半双工方式发送（即发送和接收不能同时进行）。发送数据时禁止接收，反之亦然。将 SYNC 位（TXSTA<4>）置 1 可以进入同步模式。此外，应将使能位 SPEN（RCSTA1<7>）置 1，分别把 TX1 和 RX1 引脚配置为 CK1（时钟）和 DT1（数据）线。

主模式意味着处理器在 CK1 时钟线上发送主时钟信号。时钟极性是通过 SCKP 位（BAUDCON<4>）选择的。将 SCKP 置 1 是将空闲状态时的 CK1 设为高电平，将该位清零则将空闲状态时的 CK1 设为低电平。此选项支持将本模块与 Microwire 器件配合使用。

19.4.1 EUSART 同步主发送

图 19-3 给出了 EUSART 发送器框图。发送器的核心是发送（串行）移位寄存器（TSR）。移位寄存器从读 / 写发送缓冲寄存器 TXREG1 中获取数据。用软件向 TXREG1 寄存器装入数据。在前一次装入数据的最后一位发送完成后，才向 TSR 寄存器装入新数据。一旦最后一位发送完成，就会将 TXREG1 寄存器的新数据（如果有）装入 TSR。

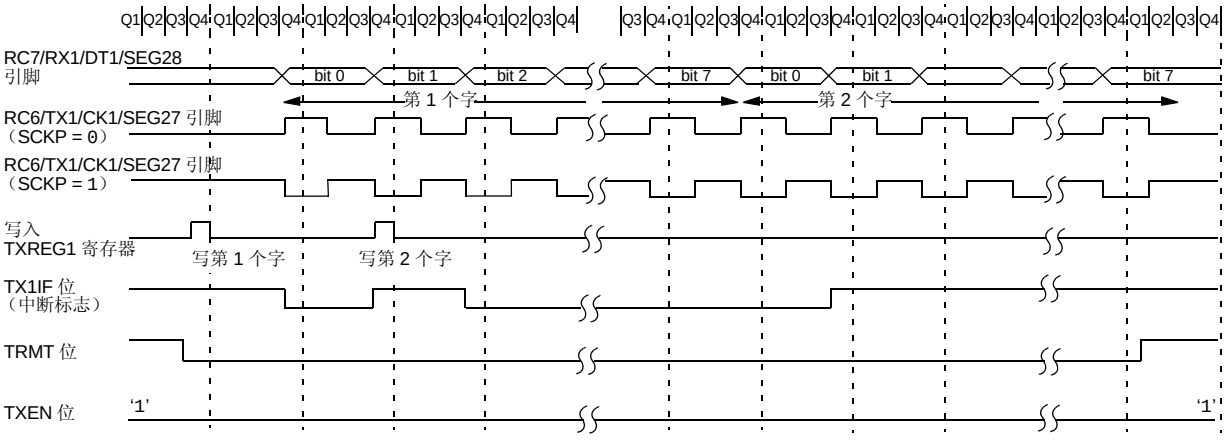
一旦 TXREG1 寄存器向 TSR 寄存器传输了数据（在 1 个 TCYCLE 内发生），TXREG1 就为空，同时 TX1IF 标志位（PIR1<4>）被置 1。可以通过将中断允许位 TX1IE（PIE1<4>）置 1 或清零来允许或禁止该中断。TX1IF 的设置不受 TX1IE 状态的影响，且不能用软件清零。只有在新数据装入 TXREG1 寄存器时，TX1IF 才会复位。

标志位 TX1IF 表示的是 TXREG1 寄存器的状态，而另一个标志位 TRMT（TXSTA<1>）则表示 TSR 寄存器的状态。TRMT 位是一个只读位，当 TSR 为空时，TRMT 被置 1。TRMT 位与任何中断逻辑均无关联，因此要确定 TSR 寄存器是否为空，用户只能对此位进行查询。TSR 不映射到数据存储器中，因此用户无法使用。

设置同步主发送操作的步骤如下：

1. 对 SPBRGH1:SPBRG1 寄存器进行初始化，设置合适的波特率。按需要将 BRG16 位置 1 或清零，以获得所需的波特率。
2. 通过将 SYNC、SPEN 和 CSRC 位置 1，使能同步主串口。
3. 如果需要中断，将中断允许位 TX1IE 置 1。
4. 如果需要 9 位发送，将 TX9 位置 1。
5. 通过将 TXEN 位置 1 使能发送。
6. 如果选择了 9 位发送，应将第 9 位装入 TX9D 位。
7. 将数据装入 TXREG1 寄存器，启动发送。
8. 如果使用中断，应确保 INTCON 寄存器中的 GIE 和 PEIE 位（INTCON<7:6>）置 1。

图 19-11： 同步发送



注： 同步主模式，SPBRG1 = 0，连续发送两个 8 位字。

图 19-12：　　同步发送 （由 TXEN 位控制）

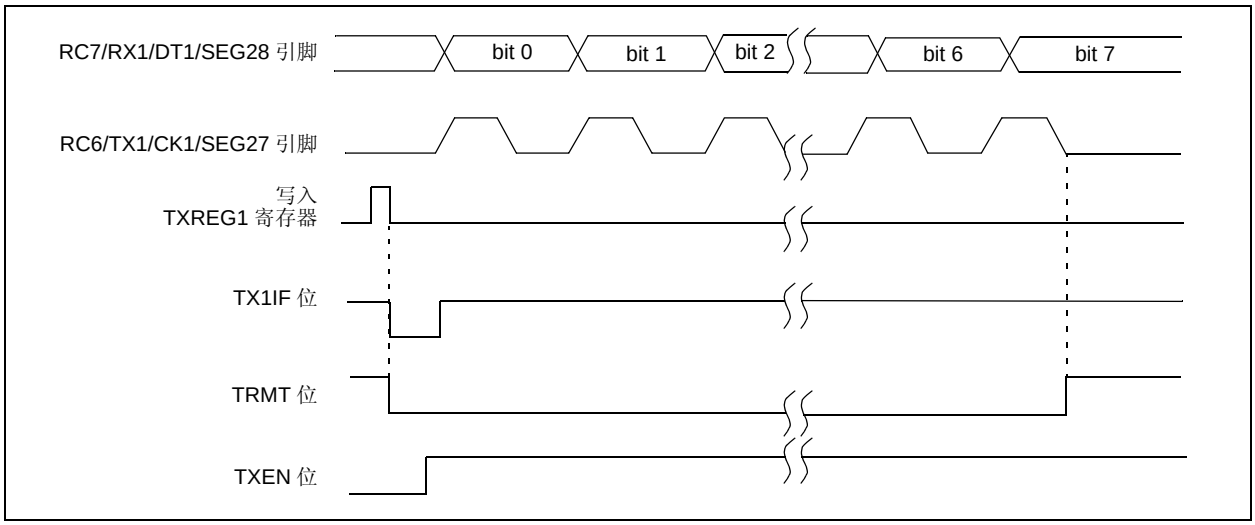


表 19-7：　　与同步主发送相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	62
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	62
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	62
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	61
TXREG1	EUSART 发送寄存器								61
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	61
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	63
SPBRGH1	EUSART 波特率发生器寄存器的高字节								63
SPBRG1	EUSART 波特率发生器寄存器的低字节								61
LATG	U2OD	U1OD	—	LATG4	LATG3	LATG2	LATG1	LATG0	62

图注：　— = 未实现，读为 0。同步主发送不使用阴影单元。

19.4.2 EAUSART 同步主接收

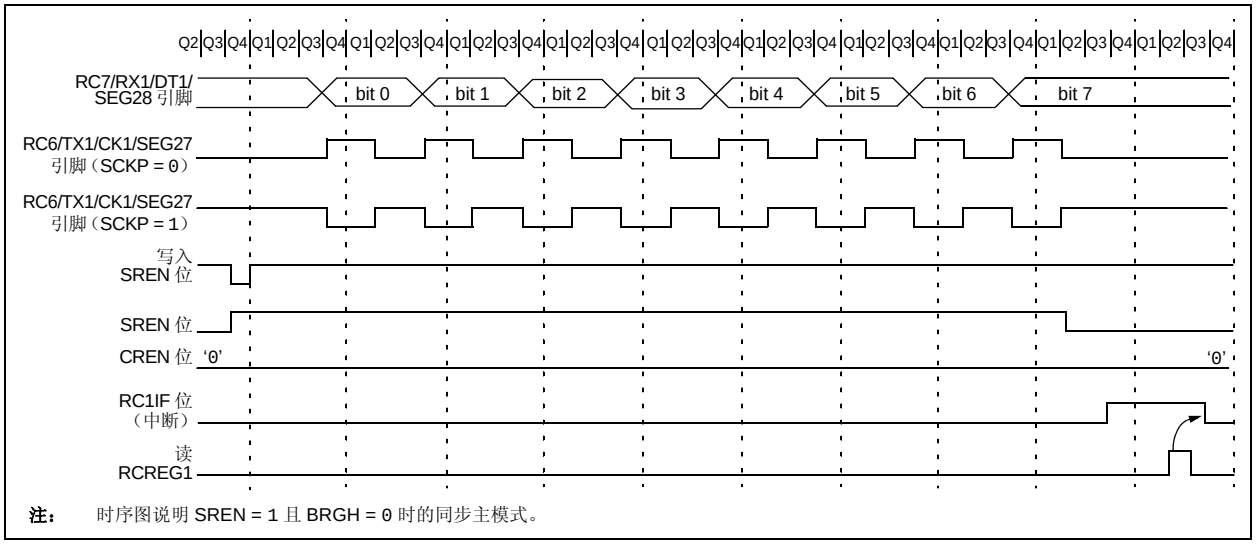
一旦选择了同步模式，只要将单字节接收使能位 SREN（RCSTA1<5>）或连续接收使能位 CREN（RCSTA1<4>）置 1，即可使能接收。在时钟的下降沿采样 RX1 引脚上的数据。

如果使能位 SREN 置 1，则只接收单个字。如果将使能位 CREN 置 1，则会连续接收数据，直到 CREN 位清零。如果两个位均被置 1，则 CREN 优先。

设置同步主接收操作的步骤如下：

1. 对 SPBRGH1:SPBRG1 寄存器进行初始化，设置合适的波特率。按需要将 BRG16 位置 1 或清零，以获得所需的波特率。
2. 通过将 SYNC、SPEN 和 CSRC 位置 1，使能同步主串口。
3. 确保将 CREN 和 SREN 位清零。
4. 如果需要中断，将中断允许位 RC1IE 置 1。
5. 如果需要接收 9 位数据，将 RX9 位置 1。
6. 如果需要单字接收，将 SREN 位置 1。如果需要连续接收，将 CREN 位置 1。
7. 当接收完成时中断标志位 RC1IF 将置 1，此时如果中断允许位 RC1IE 已置 1，则还将产生一个中断。
8. 读 RCSTA1 寄存器获取第 9 位数据（如果已使能），并判断在接收过程中是否发生了错误。
9. 读 RCREG1 寄存器来读取接收到的 8 位数据。
10. 如果发生错误，将 CREN 位清零以清除错误。
11. 如果使用中断，应确保 INTCON 寄存器中的 GIE 和 PEIE 位（INTCON<7:6>）置 1。

图 19-13： 同步接收（主模式， SREN）



注： 时序图说明 SREN = 1 且 BRGH = 0 时的同步主模式。

表 19-8： 与同步主接收相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	62
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	62
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	62
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	61
RCREG1	EUSART 接收寄存器								61
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	61
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	63
SPBRGH1	EUSART 波特率发生器寄存器的高字节								63
SPBRG1	EUSART 波特率发生器寄存器的低字节								61

图注： — = 未实现，读为 0。同步主接收不使用阴影单元。

19.5 EUSART 同步从模式

将 CSRC 位（TXSTA<7>）清零可以进入同步从模式。此模式与同步主模式的区别在于移位时钟由CK1引脚上的外部时钟提供（主模式中由内部时钟提供）。这使器件能在任何低功耗模式下发送或接收数据。

19.5.1 EUSART 同步从发送

除了休眠模式以外，同步主模式和从模式的工作原理完全相同。

如果向 TXREG1 写 2 个字，然后执行 SLEEP 指令，则会发生以下事件：

a) 第一个字将立即传送到 TSR 寄存器并发送。
b) 第二个字仍保留在 TXREG1 寄存器中。
c) 不会将标志位 TX1IF 置 1。
d) 当第一个字移出 TSR 后，TXREG1 寄存器将把第二个字传送给 TSR，同时将标志位 TX1IF 置 1。
e) 如果中断允许位 TX1IE 置 1，中断将把器件从休眠状态唤醒。如果允许了全局中断，程序则会跳转到中断向量处执行。

设置同步从发送操作的步骤如下：

1. 通过将 SYNC 和 SPEN 位置 1 并将 CSRC 位清零，使能同步从串口。
2. 清零 CREN 和 SREN 位。
3. 如果需要中断，将中断允许位 TX1IE 置 1。
4. 如果需要 9 位发送，将 TX9 位置 1。
5. 通过将使能位 TXEN 置 1 使能发送。
6. 如果选择了 9 位发送，应将第 9 位装入 TX9D 位。
7. 将数据装入 TXREG1 寄存器，启动发送。
8. 如果使用中断，应确保 INTCON 寄存器中的 GIE 和 PEIE 位（INTCON<7:6>）置 1。

表 19-9： 与同步从发送相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	62
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	62
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	62
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	61
TXREG1	EUSART 发送寄存器								61
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	61
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	63
SPBRGH1	EUSART 波特率发生器寄存器的高字节								63
SPBRG1	EUSART 波特率发生器寄存器的低字节								61
LATG	U2OD	U1OD	—	LATG4	LATG3	LATG2	LATG1	LATG0	62

图注： — = 未实现，读为 0。同步从发送不使用阴影单元。

19.5.2 EUSART 同步从接收

除了休眠模式、空闲模式以及在从模式下忽略SREN位的情况以外，同步主模式和从模式的工作原理完全相同。

如果在进入休眠或任何空闲模式前将CREN位置1使能接收，那么在低功耗模式下可以接收一个数据字。接收到数据字后，RSR 寄存器将把数据发送到 RCREG1 寄存器。如果中断允许位 RC1IE 已置 1，产生的中断将把芯片从低功耗模式唤醒。如果允许了全局中断，程序则会跳转到中断向量处执行。

设置同步从接收操作的步骤如下：

1. 通过将 SYNC 和 SPEN 位置 1 并将 CSRC 位清零，使能同步从串口。
2. 如果需要中断，将中断允许位 RC1IE 置 1。
3. 如果需要接收 9 位数据，将 RX9 位置 1。
4. 将使能位 CREN 置 1 以使能接收。
5. 当接收完成时， RC1IF 位将被置 1。如果中断允许位 RC1IE 已置 1，则还将产生一个中断。
6. 读 RCSTA1 寄存器获取第 9 位数据 （如果已使能），并判断在接收过程中是否发生了错误。
7. 读 RCREG1 寄存器来读取接收到的 8 位数据。
8. 如果发生错误，将 CREN 位清零以清除错误。
9. 如果使用中断，应确保 INTCON 寄存器中的 GIE 和 PEIE 位 （INTCON<7:6>）置 1。

表 19-10： 与同步从接收相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	62
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	62
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	62
RCSTA1	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	61
RCREG1	EUSART 接收寄存器								61
TXSTA1	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	61
BAUDCON1	ABDOVF	RCIDL	RXDTP	TXCKP	BRG16	—	WUE	ABDEN	63
SPBRGH1	EUSART 波特率发生器寄存器的高字节								63
SPBRG1	EUSART 波特率发生器寄存器的低字节								61

图注： — = 未实现，读为 0。同步从接收不使用阴影单元。

20.0 可寻址的通用同步 / 异步收发器（AUSART）

可寻址的通用同步/异步收发器（Addressable Universal Synchronous Asynchronous Receiver Transmitter，AUSART）模块在功能上与上一章讨论的增强型 USART 模块非常类似。AUSART 为与不需要自动波特率检测或 LIN/J2602 总线支持的应用场合中的外部器件进行串行通信提供了一个额外的通道。

AUSART 可配置为以下几种工作模式：

- 异步模式（全双工）
- 同步 —— 主模式（半双工）
- 同步 —— 从模式（半双工）

AUSART 模块的引脚与 PORTG 的功能复用（分别为 RG1/TX2/CK2 和 RG2/RX2/DT2/V_{LCAP1} 引脚）。要将这些引脚配置为用于 AUSART：

- SPEN 位（RCSTA2<7>）必须置 1（= 1）
- TRISG<2> 位必须置 1（= 1）
- TRISG<1> 位必须清零（= 0），使该模块工作于异步和同步主模式
- TRISG<1> 位必须置 1（= 1），使该模块工作于同步从模式

注：	AUSART 控制根据需要会自动将引脚从输入重新配置为输出。

TX2 输出引脚的驱动器也可以有选择地配置为漏极开路输出。此功能使引脚上的电平可通过外部上拉电阻上拉至较高的电平，并且无需额外的电平转换器就可使输出与外部电路进行通信。

漏极开路输出选项由 U2OD 位（LATG<7>）控制。通过将该位置 1 可将该引脚配置为漏极开路操作。

20.1 控制寄存器

可寻址 USART 模块的操作是由 TXSTA2 和 RXSTA2 这两个寄存器控制的。这两个寄存器将在寄存器 20-1 和寄存器 20-2 中分别详细介绍。

寄存器 20-1： TXSTA2：AUSART 发送状态和控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN(1)	SYNC	—	BRGH	TRMT	TX9D
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 **CSRC**：时钟源选择位
异步模式：
无关位。
同步模式：
1 = 主模式 （时钟由内部 BRG 产生）
0 = 从模式 （时钟来自外部时钟源）

bit 6 **TX9**：9 位发送使能位
1 = 选择 9 位发送
0 = 选择 8 位发送

bit 5 **TXEN**：发送使能位 (1)
1 = 使能发送
0 = 禁止发送

bit 4 **SYNC**：AUSART 模式选择位
1 = 同步模式
0 = 异步模式

bit 3 **未实现**：读为 0

bit 2 **BRGH**：高波特率选择位
异步模式：
1 = 高速
0 = 低速
同步模式：
在此模式下未使用。

bit 1 **TRMT**：发送移位寄存器状态位
1 = TSR 空
0 = TSR 满

bit 0 **TX9D**：发送数据的第 9 位
可以是地址 / 数据位或奇偶校验位。

注 1： 在同步模式下， SREN/CREN 优先于 TXEN。

寄存器 20-2： RCSTA2：AUSART 接收状态和控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
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 **SPEN**：串口使能位
1 = 使能串口 （将 RX2/DT2 和 TX2/CK2 （TXEN = 1）引脚配置为串口引脚）
0 = 禁止串口 （保持在复位状态）

bit 6 **RX9**：9 位接收使能位
1 = 选择 9 位接收
0 = 选择 8 位接收

bit 5 **SREN**：单字节接收使能位
异步模式：
无关位。
同步主模式：
1 = 使能单字节接收
0 = 禁止单字节接收
此位在接收完成后清零。
同步从模式：
无关位。

bit 4 **CREN**：连续接收使能位
异步模式：
1 = 使能接收器
0 = 禁止接收器
同步模式：
1 = 使能连续接收，直到使能位 CREN 清零 （CREN 的优先级高于 SREN）
0 = 禁止连续接收

bit 3 **ADDEN**：地址检测使能位
9 位异步模式 （RX9 = 1）：
1 = 当 RSR<8> 置 1 时，使能地址检测，允许中断并装入接收缓冲区
0 = 禁止地址检测，接收所有字节并且第 9 位可作为奇偶校验位
9 位异步模式 （RX9 = 0）：
无关位。

bit 2 **FERR**：帧错误位
1 = 帧错误 （可以通过读 RCREGx 寄存器清零该位并接收下一个有效字节）
0 = 无帧错误

bit 1 **OERR**：溢出错误位
1 = 溢出错误 （可以通过清零 CREN 位来清零该位）
0 = 无溢出错误

bit 0 **RX9D**：接收数据的第 9 位
该位可以是地址 / 数据位或奇偶校验位，并且必须由用户固件计算得到。

20.2 AUSART 波特率发生器 （BRG）

BRG 是一个专用的 8 位发生器，支持 AUSART 的异步和同步模式。

SPBRG2 寄存器控制自由运行定时器的周期。在异步模式下，BRGH 位（TXSTA<2>）也用于控制波特率。在同步模式下， BRGH 位会被忽略。表 20-1 所示为不同 AUSART 模式的波特率计算公式，但仅适用于主模式（由内部产生时钟信号）。

给出目标波特率和 FOSC 值，就可以使用表 20-1 中的公式计算 SPBRG2 寄存器的最近似整数值。这样就可以确定波特率误差。例 20-1 给出了计算示例。表 20-2 给出了不同异步模式下典型的波特率和误差值。使用高波特率 （BRGH = 1）有助于降低波特率误差，或者在快速振荡频率下取得较缓慢的波特率。

将新值写入 SPBRG2 寄存器会导致 BRG 定时器复位（或清零）。这可以确保 BRG 无需等待定时器溢出就可以输出新的波特率。

20.2.1 在功耗管理模式下的操作

器件时钟用于产生所需的波特率。当进入一种功耗管理模式时，新时钟源可能会工作在一个不同的频率下。这可能需要调整 SPBRG2 寄存器中的值。

20.2.2 采样

择多检测电路对 RX2 引脚采样三次，以判定 RX2 引脚上出现的是高电平还是低电平。

表 20-1： 波特率公式

配置位		BRG/AUSART 模式	波特率公式
SYNC	BRGH		
0	0	异步	FOSC/[64 (n + 1)]
0	1	异步	FOSC/[16 (n + 1)]
1	x	同步	FOSC/[4 (n + 1)]

图注： x = 无关位， n = SPBRG2 寄存器的值

例 20-1： 计算波特率误差

器件工作在 FOSC = 16 MHz，目标波特率 = 9600，异步模式， BRGH = 0：

目标波特率 = FOSC/(64 ([SPBRG2] + 1))

求解 SPBRG2：

X = ((FOSC/ 目标波特率)/64) – 1
= ((16000000/9600)/64) – 1
= [25.042] = 25

计算波特率 = 16000000/(64 (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%

表 20-2： 与波特率发生器相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
TXSTA2	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	64
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	64
SPBRG2	AUSART 波特率发生器寄存器								64

图注： BRG 不使用阴影单元。

表 20-3： 异步模式下的波特率

波特率（K）	BRGH = 0											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	1.221	1.73	255	1.202	0.16	129	1.201	-0.16	103
2.4	2.441	1.73	255	2.404	0.16	129	2.404	0.16	64	2.403	-0.16	51
9.6	9.615	0.16	64	9.766	1.73	31	9.766	1.73	15	9.615	-0.16	12
19.2	19.531	1.73	31	19.531	1.73	15	19.531	1.73	7	—	—	—
57.6	56.818	-1.36	10	62.500	8.51	4	52.083	-9.58	2	—	—	—
115.2	125.000	8.51	4	104.167	-9.58	2	78.125	-32.18	1	—	—	—

波特率（K）	BRGH = 0								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）
0.3	0.300	0.16	207	0.300	-0.16	103	0.300	-0.16	51
1.2	1.202	0.16	51	1.201	-0.16	25	1.201	-0.16	12
2.4	2.404	0.16	25	2.403	-0.16	12	—	—	—
9.6	8.929	-6.99	6	—	—	—	—	—	—
19.2	20.833	8.51	2	—	—	—	—	—	—
57.6	62.500	8.51	0	—	—	—	—	—	—
115.2	62.500	-45.75	0	—	—	—	—	—	—

波特率（K）	BRGH = 1											
	Fosc = 40.000 MHz			Fosc = 20.000 MHz			Fosc = 10.000 MHz			Fosc = 8.000 MHz		
	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	—	—	—	—	—	—	—	—	—
2.4	—	—	—	—	—	—	2.441	1.73	255	2.403	-0.16	207
9.6	9.766	1.73	255	9.615	0.16	129	9.615	0.16	64	9.615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19.230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55.555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

波特率（K）	BRGH = 1								
	Fosc = 4.000 MHz			Fosc = 2.000 MHz			Fosc = 1.000 MHz		
	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）	实际波特率（K）	% 误差	SPBRG 值（十进制）
0.3	—	—	—	—	—	—	0.300	-0.16	207
1.2	1.202	0.16	207	1.201	-0.16	103	1.201	-0.16	51
2.4	2.404	0.16	103	2.403	-0.16	51	2.403	-0.16	25
9.6	9.615	0.16	25	9.615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

20.3 AUSART 异步模式

通过将 SYNC 位（TXSTA2<4>）清零可选择异步工作模式。在此模式下，AUSART 使用标准的不归零码（NRZ）格式（1 个启动位、8 个或 9 个数据位和 1 个停止位）。最常见的数据格式为 8 位。片上专用 8 位波特率发生器可用于从振荡器产生标准波特率频率。

AUSART 先发送和接收 LSb。 AUSART 的发送器和接收器在功能上是相互独立的，但采用相同的数据格式和波特率。根据 BRGH 位 （TXSTA2<2>）的设置值，波特率发生器可以产生两种不同的波特率时钟，频率分别为移位速率的 16 倍或 64 倍。硬件不支持奇偶校验，但可通过软件实现并作为第 9 个数据存储。

当工作在异步模式下时，AUSART 模块包括以下重要组成部分：

- 波特率发生器
- 采样电路
- 异步发送器
- 异步接收器

20.3.1 AUSART 异步发送器

图 20-1 给出了 AUSART 发送器的框图。发送器的核心是发送（串行）移位寄存器（TSR）。移位寄存器从读 / 写发送缓冲寄存器 TXREG2 中获取数据。用软件向 TXREG2 寄存器装入数据。在前一次装入数据的停止位发送前，不会向 TSR 寄存器装入新数据。一旦停止位发送完毕， TXREG2 寄存器中的新数据 （如果有）就会被装入 TSR。

一旦 TXREG2 寄存器向 TSR 寄存器传输了数据 （在 1 个 TCY 内发生），TXREG2 寄存器就为空，同时标志位 TX2IF（PIR3<4>）置 1。可以通过将中断允许位 TX2IE（PIE3<4>）置 1 或清零来允许或禁止该中断。不管 TX2IE 的状态如何，只要中断发生， TX2IF 就会置 1 并且不能用软件清零。TX2IF 不会在 TXREG2 装入新数据时立即被清零，而是在装入指令后的第二个指令周期被清零。因此在 TXREG2 装入新数据后立即查询 TX2IF，会得到无效结果。

TX2IF 指示 TXREG2 寄存器的状态，而另一个位 TRMT（TXSTA2<1>）则指示 TSR 寄存器的状态。 TRMT 是只读位，它在 TSR 寄存器为空时置 1。TRMT 位与任何中断逻辑均无关联，因此要确定 TSR 寄存器是否为空，用户只能对此位进行查询。

注	**1：** TSR 寄存器不映射到数据存储器中，因此用户无法使用。
	2： 当使能位TXEN置1时，标志位TX2IF置1。

设置异步发送操作的步骤如下：

1. 对 SPBRG2 寄存器进行初始化，设置合适的波特率。按需要将 BRGH 位置 1 或清零，以获得所需的波特率。
2. 通过清零 SYNC 位并将 SPEN 位置 1，使能异步串口。
3. 如果需要中断，将中断允许位 TX2IE 置 1。
4. 如果需要 9 位发送，将发送位 TX9 置 1。可以作为地址 / 数据位使用。
5. 通过将 TXEN 位置 1 使能发送，此操作同时也会将 TX2IF 位置 1。
6. 如果选择了 9 位发送，应将第 9 位装入 TX9D 位。
7. 将数据装入 TXREG2 寄存器 （启动发送）。
8. 如果使用中断，应确保 INTCON 寄存器中的 GIE 和 PEIE 位 （INTCON<7:6>）置 1。

图 20-1： AUSART 发送框图

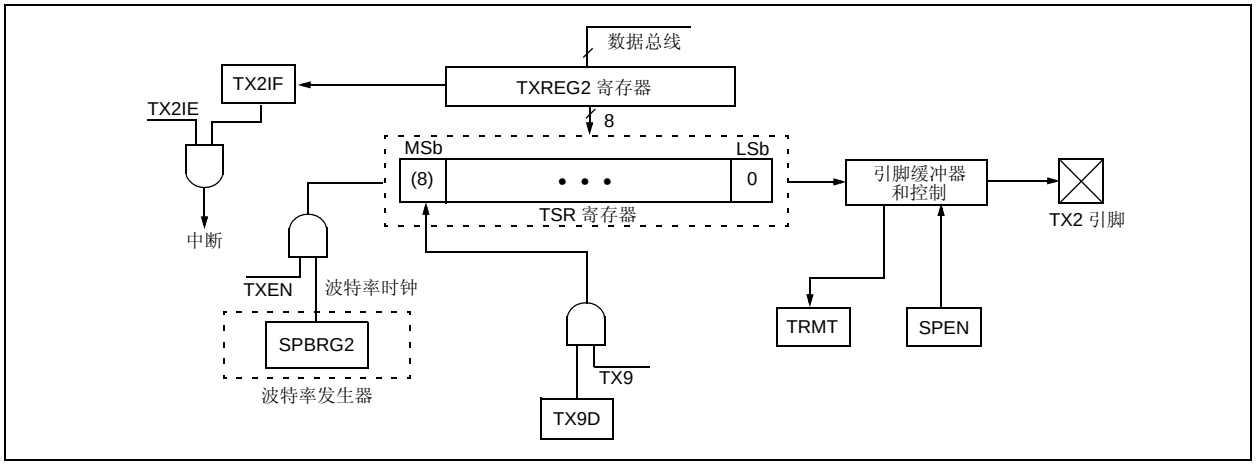


图 20-2：　　异步发送

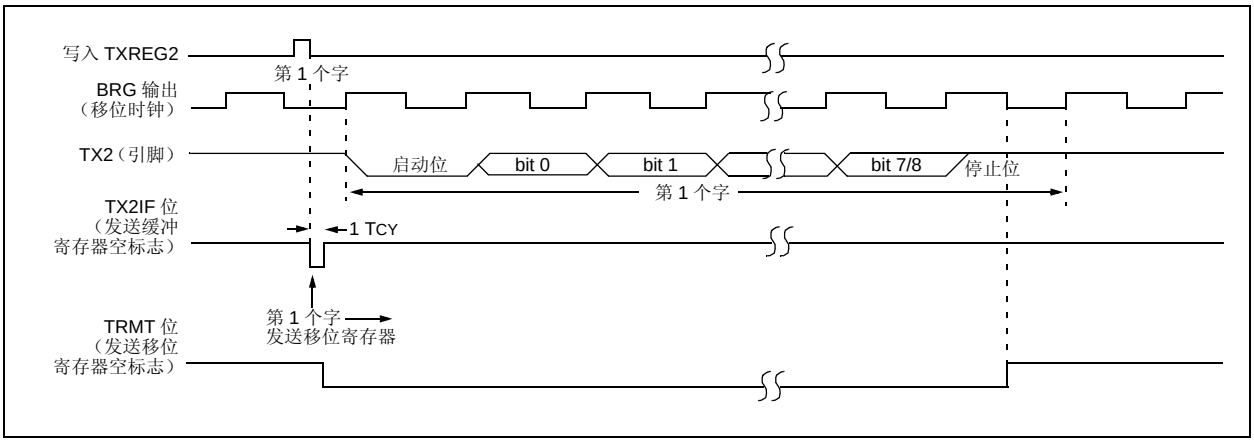
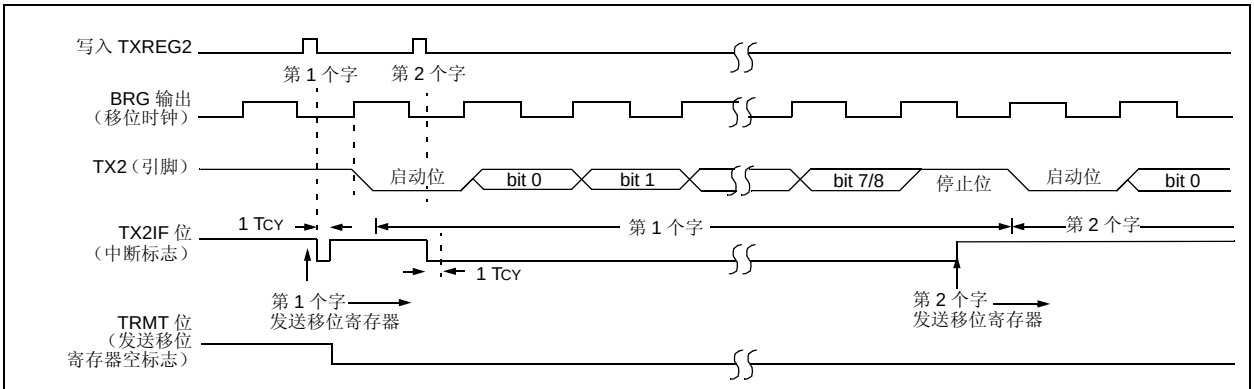


图 20-3：　　异步发送 （背对背）



注：　此时序图表示的是两次连续的发送。

表 20-4：　　与异步发送相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR3	—	LCDIF	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	RTCCIF	62
PIE3	—	LCDIE	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	RTCCIE	62
IPR3	—	LCDIP	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	RTCCIP	62
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	64
TXREG2	AUSART 发送寄存器								64
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	64
SPBRG2	AUSART 波特率发生器寄存器								64
LATG	U2OD	U1OD	—	LATG4	LATG3	LATG2	LATG1	LATG0	62

图注：　— = 未实现位，读为 0。异步发送不使用阴影单元。

20.3.2 AUSART 异步接收器

图 20-4 给出了接收器框图。在 RX2 引脚上接收数据，并驱动数据恢复模块。数据恢复模块实际上是一个工作频率为 16 倍波特率的高速移位器，而主接收串行移位器的工作频率等于比特率或 FOSC。此模式通常用于 RS-232 系统中。

设置异步接收操作的步骤如下：

1. 对 SPBRG2 寄存器进行初始化，设置合适的波特率。按需要将 BRGH 位置 1 或清零，以获得所需的波特率。
2. 通过清零 SYNC 位并将 SPEN 位置 1，使能异步串口。
3. 如果需要中断，将中断允许位 RC2IE 置 1。
4. 如果需要接收 9 位数据，将 RX9 位置 1。
5. 通过将 CREN 位置 1 使能接收。
6. 当接收完成时标志位 RC2IF 将被置 1，此时如果中断允许位RC2IE已置1，则还将产生一个中断。
7. 读 RCSTA2 寄存器获取第 9 位数据 （如果已使能），并判断在接收过程中是否发生了错误。
8. 读 RCREG2 寄存器来读取接收到的 8 位数据。
9. 如果发生错误，将使能位CREN清零以清除错误。
10. 如果使用中断，应确保 INTCON 寄存器中的 GIE 和 PEIE 位 （INTCON<7:6>）置 1。

20.3.3 设置带有地址检测功能的 9 位模式

此模式通常用于 RS-485 系统中。设置使能地址检测的异步接收的步骤如下：

1. 对 SPBRG2 寄存器进行初始化，设置合适的波特率。按需要将 BRGH 和 BRG16 位置 1 或清零，以获得所需的波特率。
2. 通过清零 SYNC 位并将 SPEN 位置 1，使能异步串口。
3. 如果需要中断，将 RCEN 位置 1 并用 RC2IP 位选择所需的优先级别。
4. 将 RX9 位置 1 使能 9 位接收。
5. 将 ADDEN 位置 1 使能地址检测。
6. 将 CREN 位置 1 使能接收。
7. 当接收完成时RC2IF位将被置1。此时如果RC2IE和 GIE 位已置 1，还将响应中断。
8. 读 RCSTA2 寄存器判断在接收时是否发生了错误，同时读取第 9 位数据 （如果适用）。
9. 读 RCREG2 判断是否正在对器件进行寻址。
10. 如果发生错误，将 CREN 位清零。
11. 如果已寻址到器件，将 ADDEN 位清零以允许所有接收到的数据被送入接收缓冲区并发送中断信号给 CPU。

图 20-4： AUSART 接收框图

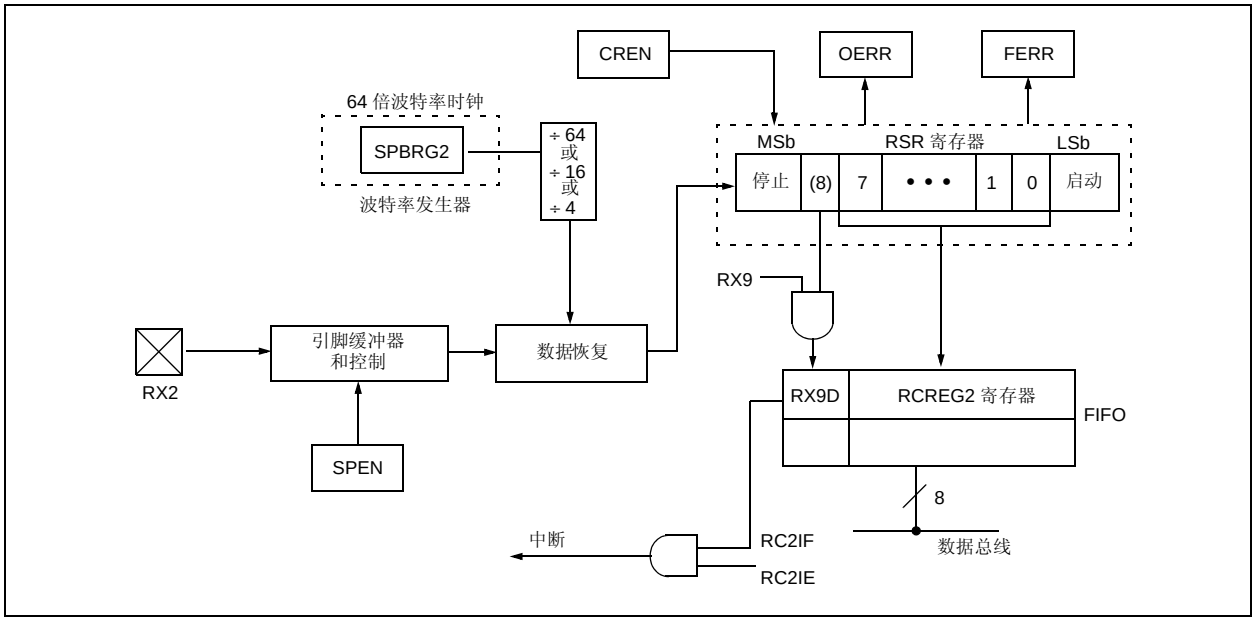
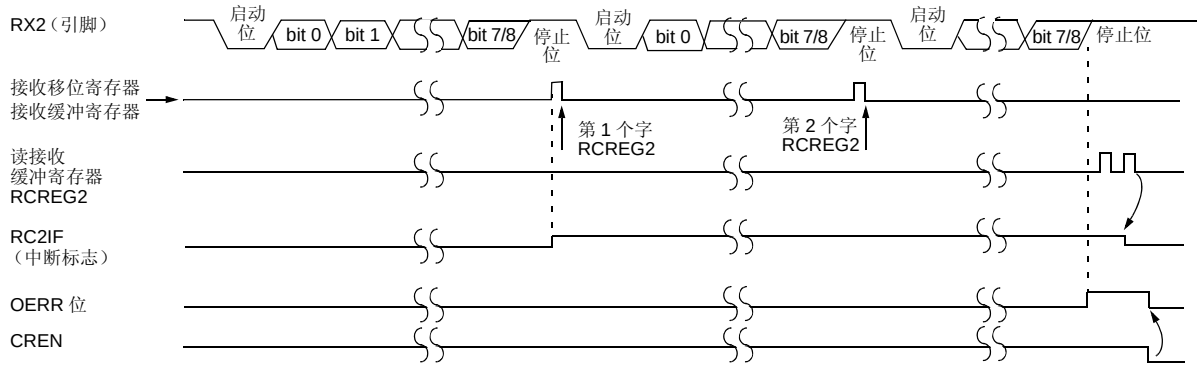


图 20-5：　　异步接收



注：　此时序图显示了在 RX2 输入引脚上接收的 3 个字。在第 3 个字之后读 RCREG2（接收缓冲寄存器 2）会使 OERR（溢出）位置 1。

表 20-5：　　与异步接收相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR3	—	LCDIF	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	RTCCIF	62
PIE3	—	LCDIE	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	RTCCIE	62
IPR3	—	LCDIP	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	RTCCIP	62
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	64
RCREG2	AUSART 接收寄存器								64
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	64
SPBRG2	AUSART 波特率发生器寄存器								64

图注：　— = 未实现位，读为 0。异步接收不使用阴影单元。

20.4 AUSART 同步主模式

将 CSRC 位（TXSTA2<7>）置 1 可以进入同步主模式。在此模式下，数据以半双工方式发送（即发送和接收不能同时进行）。发送数据时禁止接收，反之亦然。将 SYNC 位（TXSTA2<4>）置 1 可以进入同步模式。此外，应将使能位 SPEN（RCSTA2<7>）置 1，分别把 TX2 和 RX2 引脚配置为 CK2（时钟）和 DT2（数据）线。

主模式意味着处理器在 CK2 时钟线上发送主时钟信号。

20.4.1 AUSART 同步主发送

图 20-1 给出了 AUSART 发送器的框图。发送器的核心是发送（串行）移位寄存器（TSR）。移位寄存器从读 / 写发送缓冲寄存器 TXREG2 中获取数据。用软件向 TXREG2 寄存器装入数据。在前一次装入数据的最后一位发送完成后，才向 TSR 寄存器装入新数据。一旦最后一位发送完成，就会将 TXREG2 寄存器的新数据（如果有）装入 TSR。

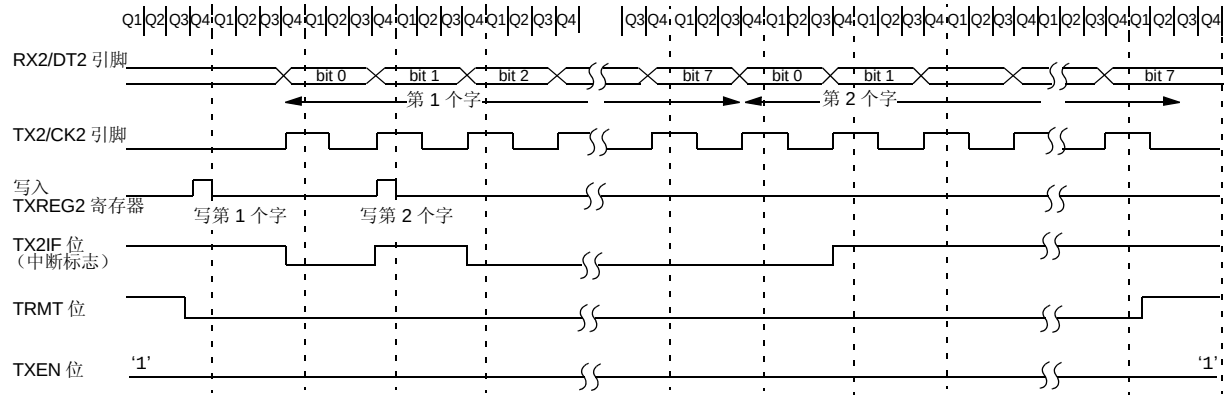
一旦 TXREG2 寄存器向 TSR 寄存器传输了数据（在 1 个 TCYCLE 内发生），TXREG2 就为空，同时 TX2IF 标志位（PIR3<4>）被置 1。可以通过将中断允许位 TX2IE（PIE3<4>）置 1 或清零来允许或禁止该中断。TX2IF 的设置不受允许位 TX2IE 状态的影响，且不能用软件清零。只有在新数据装入 TXREG2 寄存器时，TX2IF 才会复位。

标志位 TX2IF 指示 TXREG2 寄存器的状态，而另一个标志位 TRMT（TXSTA2<1>）则指示 TSR 寄存器的状态。TRMT 位是一个只读位，当 TSR 为空时，TRMT 被置 1。TRMT 位与任何中断逻辑均无关联，因此要确定 TSR 寄存器是否为空，用户只能对此位进行查询。TSR 不映射到数据存储器中，因此用户无法使用。

设置同步主发送操作的步骤如下：

1. 对 SPBRG2 寄存器进行初始化，设置合适的波特率。
2. 通过将 SYNC、SPEN 和 CSRC 位置 1，使能同步主串口。
3. 如果需要中断，将中断允许位 TX2IE 置 1。
4. 如果需要 9 位发送，将 TX9 位置 1。
5. 通过将 TXEN 位置 1 使能发送。
6. 如果选择了 9 位发送，应将第 9 位装入 TX9D 位。
7. 将数据装入 TXREG2 寄存器，启动发送。
8. 如果使用中断，应确保 INTCON 寄存器中的 GIE 和 PEIE 位（INTCON<7:6>）置 1。

图 20-6： 同步发送



注： 同步主模式，SPBRG2 = 0，连续发送两个 8 位字。

图 20-7：　　同步发送 （由 TXEN 位控制）

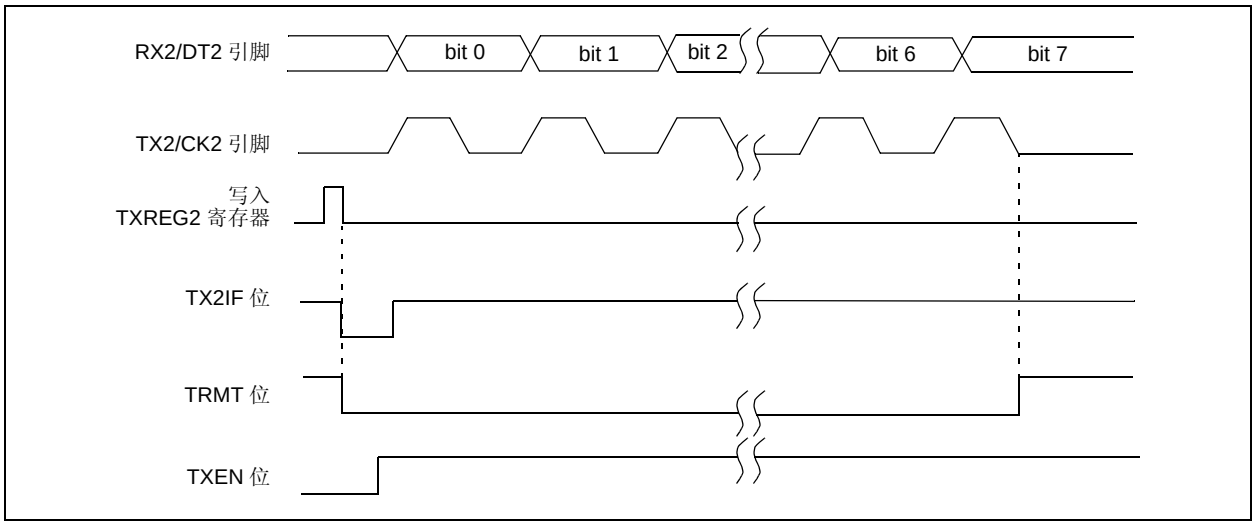


表 20-6：　　与同步主发送相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR3	—	LCDIF	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	RTCCIF	62
PIE3	—	LCDIE	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	RTCCIE	62
IPR3	—	LCDIP	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	RTCCIP	62
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	64
TXREG2	AUSART 发送寄存器								64
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	64
SPBRG2	AUSART 波特率发生器寄存器								64
LATG	U2OD	U1OD	—	LATG4	LATG3	LATG2	LATG1	LATG0	62

图注：　— = 未实现，读为 0。同步主发送不使用阴影单元。

20.4.2 AUSART 同步主接收

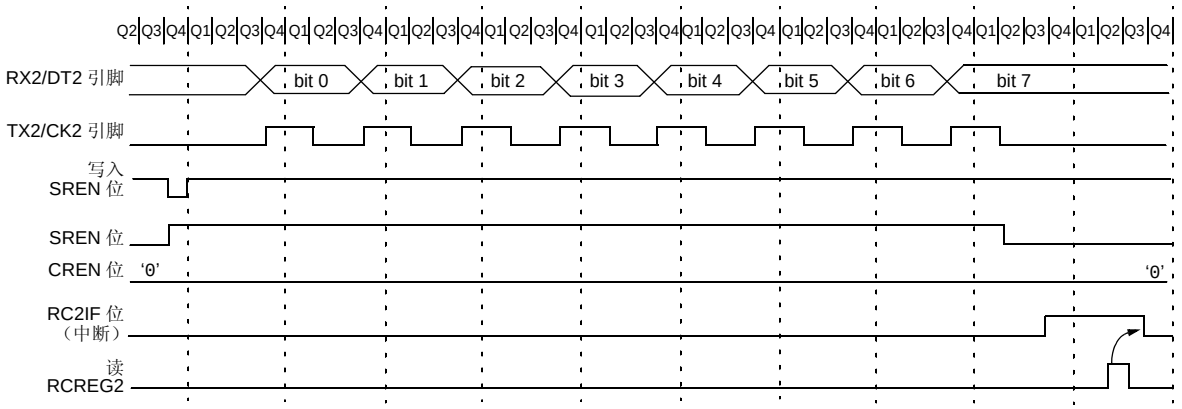
一旦选择了同步模式，只要将单字节接收使能位 SREN（RCSTA2<5>）或连续接收使能位 CREN（RCSTA2<4>）置 1，即可使能接收。在时钟的下降沿采样 RX2 引脚上的数据。

如果使能位 SREN 置 1，则只接收单个字。如果将使能位 CREN 置 1，则会连续接收数据，直到将 CREN 位清零。如果两个位均被置 1，则 CREN 优先。

设置同步主接收操作的步骤如下：

1. 对 SPBRG2 寄存器进行初始化，设置合适的波特率。
2. 通过将 SYNC、SPEN 和 CSRC 位置 1，使能同步主串口。
3. 确保将 CREN 和 SREN 位清零。
4. 如果需要中断，将中断允许位 RC2IE 置 1。
5. 如果需要接收 9 位数据，将 RX9 位置 1。
6. 如果需要单字接收，将 SREN 位置 1。如果需要连续接收，将 CREN 位置 1。
7. 当接收完成时中断标志位 RC2IF 将置 1，此时如果中断允许位 RC2IE 已置 1，则还将产生一个中断。
8. 读 RCSTA2 寄存器获取第 9 位数据（如果已使能），并判断在接收过程中是否发生了错误。
9. 读 RCREG2 寄存器来读取接收到的 8 位数据。
10. 如果发生错误，将 CREN 位清零以清除错误。
11. 如果使用中断，应确保 INTCON 寄存器中的 GIE 和 PEIE 位（INTCON<7:6>）置 1。

图 20-8：　同步接收（主模式，SREN）



注： 时序图说明 SREN = 1 且 BRGH = 0 时的同步主模式。

表 20-7：　与同步主接收相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR3	—	LCDIF	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	RTCCIF	62
PIE3	—	LCDIE	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	RTCCIE	62
IPR3	—	LCDIP	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	RTCCIP	62
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	64
RCREG2	AUSART 接收寄存器								64
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	64
SPBRG2	AUSART 波特率发生器寄存器								64

图注： — = 未实现，读为 0。同步主接收不使用阴影单元。

20.5 AUSART 同步从模式

将 CSRC 位（TXSTA2<7>）清零可以进入同步从模式。此模式与同步主模式的区别在于移位时钟由CK2引脚上的外部时钟提供 （主模式中由内部时钟提供）。这使器件能在任何低功耗模式下发送或接收数据。

20.5.1 AUSART 同步从发送

除了休眠模式以外，同步主模式和从模式的工作原理完全相同。

如果向 TXREG2 写 2 个字，然后执行 SLEEP 指令，则会发生以下事件：

a) 第一个字将立即传送到 TSR 寄存器并发送。
b) 第二个字仍保留在 TXREG2 寄存器中。
c) 不会将标志位 TX2IF 置 1。
d) 当第一个字移出 TSR 后， TXREG2 寄存器将把第二个字传送给TSR，同时将标志位TX2IF置1。
e) 如果中断允许位 TX2IE 置 1，中断将把器件从休眠状态唤醒。如果允许了全局中断，程序则会跳转到中断向量处执行。

设置同步从发送操作的步骤如下：

1. 通过将 SYNC 和 SPEN 位置 1 并将 CSRC 位清零，使能同步从串口。
2. 清零 CREN 和 SREN 位。
3. 如果需要中断，将中断允许位 TX2IE 置 1。
4. 如果需要 9 位发送，将 TX9 位置 1。
5. 通过将使能位 TXEN 置 1 使能发送。
6. 如果选择了 9 位发送，应将第 9 位装入 TX9D 位。
7. 将数据装入 TXREG2 寄存器，启动发送。
8. 如果使用中断，应确保 INTCON 寄存器中的 GIE 和 PEIE 位 （INTCON<7:6>）置 1。

表 20-8： 与同步从发送相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR3	—	LCDIF	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	RTCCIF	62
PIE3	—	LCDIE	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	RTCCIE	62
IPR3	—	LCDIP	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	RTCCIP	62
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	64
TXREG2	AUSART 发送寄存器								64
TXSTA2	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	64
SPBRG2	AUSART 波特率发生器寄存器								64
LATG	U2OD	U1OD	—	LATG4	LATG3	LATG2	LATG1	LATG0	62

图注： — = 未实现，读为 0。同步从发送不使用阴影单元。

20.5.2 AUSART 同步从接收

除了休眠模式、空闲模式以及在从模式下忽略 SREN 位以外，同步主模式和从模式的工作原理完全相同。

如果在进入休眠或空闲模式前将 CREN 位置 1 使能接收，那么在低功耗模式下可以接收一个数据字。接收到数据字后，RSR 寄存器将把数据发送到 RCREG2 寄存器。如果中断允许位 RC2IE 已置 1，产生的中断将把芯片从低功耗模式唤醒。如果允许了全局中断，程序则会跳转到中断向量处执行。

设置同步从接收操作的步骤如下：

1. 通过将 SYNC 和 SPEN 位置 1 并将 CSRC 位清零，使能同步从串口。
2. 如果需要中断，将中断允许位 RC2IE 置 1。
3. 如果需要接收 9 位数据，将 RX9 位置 1。
4. 将使能位 CREN 置 1 以使能接收。
5. 当接收完成时，RC2IF 位将被置 1。如果中断允许位 RC2IE 置 1，则还将产生一个中断。
6. 读 RCSTA2 寄存器获取第 9 位数据（如果已使能），并判断在接收过程中是否发生了错误。
7. 读 RCREG2 寄存器来读取接收到的 8 位数据。
8. 如果发生错误，将 CREN 位清零以清除错误。
9. 如果使用中断，应确保 INTCON 寄存器中的 GIE 和 PEIE 位（INTCON<7:6>）置 1。

表 20-9： 与同步从接收相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR3	—	LCDIF	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	RTCCIF	62
PIE3	—	LCDIE	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	RTCCIE	62
IPR3	—	LCDIP	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	RTCCIP	62
RCSTA2	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	64
RCREG2	AUSART 接收寄存器								64
TXSTA2	CSRC	TX9	TXEN	SYNC		BRGH	TRMT	TX9D	64
SPBRG2	AUSART 波特率发生器寄存器								64

图注： — = 未实现，读为 0。同步从接收不使用阴影单元。

21.0 10 位模数转换器 （A/D）模块

所有PIC18F87J90系列器件的模数（Analog-to-Digital，A/D）转换器模块有 12 路输入。该模块能将一个模拟输入信号转换成相应的 10 位数字。

该模块有 5 个寄存器：

- A/D 结果高位寄存器 （ADRESH）
- A/D 结果低位寄存器 （ADRESL）
- A/D 控制寄存器 0 （ADCON0）
- A/D 控制寄存器 1 （ADCON1）
- A/D 控制寄存器 2 （ADCON2）

ADCON0 寄存器（如寄存器 21-1 所示）控制 A/D 模块的工作。ADCON1 寄存器（如寄存器 21-2 所示）配置端口引脚功能。 ADCON2 寄存器 （如寄存器 21-3 所示）配置 A/D 时钟源、可编程采集时间和输出结果的对齐方式。

寄存器 21-1： ADCON0：A/D 控制寄存器 0

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADCAL	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON
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 **ADCAL**：A/D 校准位
1 = 在下次 A/D 转换时执行校准
0 = 正常 A/D 转换器操作 （不执行校准）

bit 6 **未实现**：读为 0

bit 5-2 **CHS<3:0>**：模拟通道选择位
0000 = 通道 00 （AN0）
0001 = 通道 01 （AN1）
0010 = 通道 02 （AN2）
0011 = 通道 03 （AN3）
0100 = 通道 04 （AN4）
0101 = 通道 05 （AN5）
0110 = 通道 06 （AN6）
0111 = 通道 07 （AN7）
1000 = 通道 08 （AN8）
1001 = 通道 09 （AN9）
1010 = 通道 10 （AN10）
1011 = 通道 11 （AN11）
11xx = 未使用

bit 1 **GO/$\overline{\text{DONE}}$**：A/D 转换状态位
当 ADON = 1 时：
1 = A/D 转换正在进行
0 = A/D 空闲

bit 0 **ADON**：A/D 使能位
1 = 使能 A/D 转换器模块
0 = 禁止 A/D 转换器模块

寄存器 21-2： ADCON1：A/D 控制寄存器 1

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
TRIGSEL	—	VCFG1	VCFG0	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 7 **TRIGSEL**：特殊事件触发信号选择位
1 = 选择来自 CTMU 的特殊事件触发信号
0 = 选择来自 CCP2 的特殊事件触发信号

bit 6 **未实现**：读为 0

bit 5 **VCFG1**：参考电压配置位 （V_{REF-} 电压源）
1 = V_{REF-}（AN2）
0 = AV_{SS}

bit 4 **VCFG0**：参考电压配置位 （V_{REF+} 电压源）
1 = V_{REF+}（AN3）
0 = AV_{DD}

bit 3-0 **PCFG<3:0>**：A/D 端口配置控制位：

PCFG<3:0>	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
0000	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A
0011	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	A	A	A	A	A	A	A	A	A
0111	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D

A = 模拟输入 D = 数字 I/O

寄存器 21-3： ADCON2：A/D 控制寄存器 2

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
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 **ADFM：**A/D 结果格式选择位
1 = 右对齐
0 = 左对齐

bit 6 **未实现：**读为 0

bit 5-3 **ACQT<2:0>：**A/D 采集时间选择位
111 = 20 TAD
110 = 16 TAD
101 = 12 TAD
100 = 8 TAD
011 = 6 TAD
010 = 4 TAD
001 = 2 TAD
000 = 0 TAD(1)

bit 2-0 **ADCS<2:0>：**A/D 转换时钟选择位
111 = FRC （时钟来自 A/D RC 振荡器）(1)
110 = FOSC/64
101 = FOSC/16
100 = FOSC/4
011 = FRC （时钟来自 A/D RC 振荡器）(1)
010 = FOSC/32
001 = FOSC/8
000 = FOSC/2

注 1： 如果为 A/D 选择了 FRC 时钟源，在 A/D 时钟启动之前会加上一个 TCY （指令周期）的延时。这可以保证在启动转换之前执行 SLEEP 指令。

可通过软件选择将器件的正负电源电压（VDD 和 VSS）或 RA3/AN3/VREF+ 和 RA2/AN2/VREF- 引脚上的电压作为 A/D 转换的模拟参考电压。

A/D 转换器具有可在休眠状态下工作的特性。要在休眠模式下工作，A/D 转换时钟必须来自 A/D 的内部 RC 振荡器。

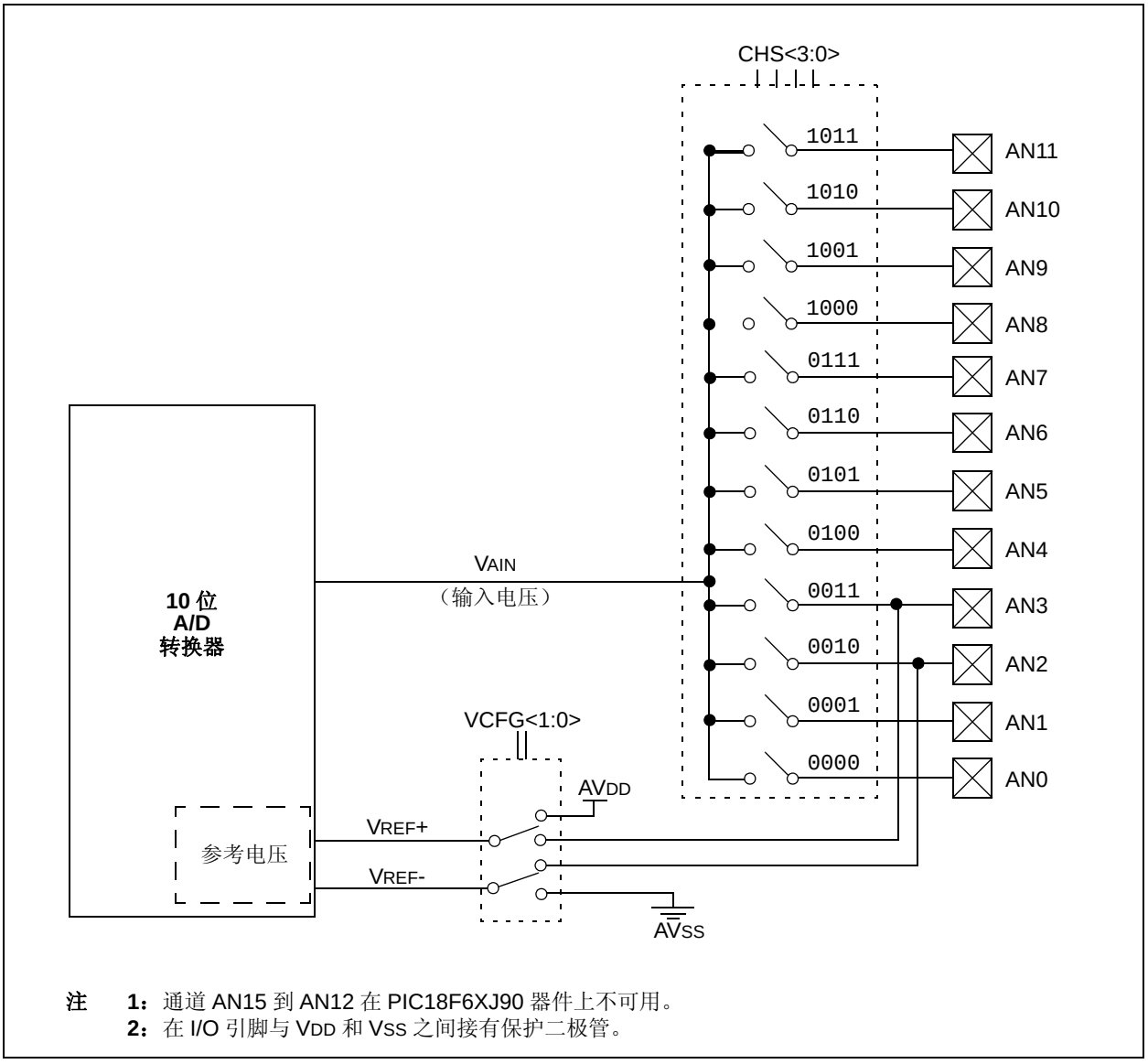
采样保持电路的输出是转换器的输入，A/D 转换器采用逐次逼近法得到转换结果。

与 A/D 转换器相关的每个端口引脚都可以被配置为模拟输入或数字 I/O。ADRESH 和 ADRESL 寄存器保存 A/D 转换的结果。当 A/D 转换完成时，结果被装入 ADRESH:ADRESL 寄存器，GO/DONE 位（ADCON0<1>）被清零且 A/D 中断标志位 ADIF 被置 1。

器件复位将强制所有寄存器为复位状态。这将强制关闭 A/D 模块并中止任何正在进行的转换。上电复位时，ADRESH:ADRESL 寄存器对的值保持不变。上电复位后，这些寄存器的值不确定。

图 21-1 给出了 A/D 模块的框图。

图 21-1：　　A/D 框图 (1,2)



注　1：通道 AN15 到 AN12 在 PIC18F6XJ90 器件上不可用。
2：在 I/O 引脚与 VDD 和 VSS 之间接有保护二极管。

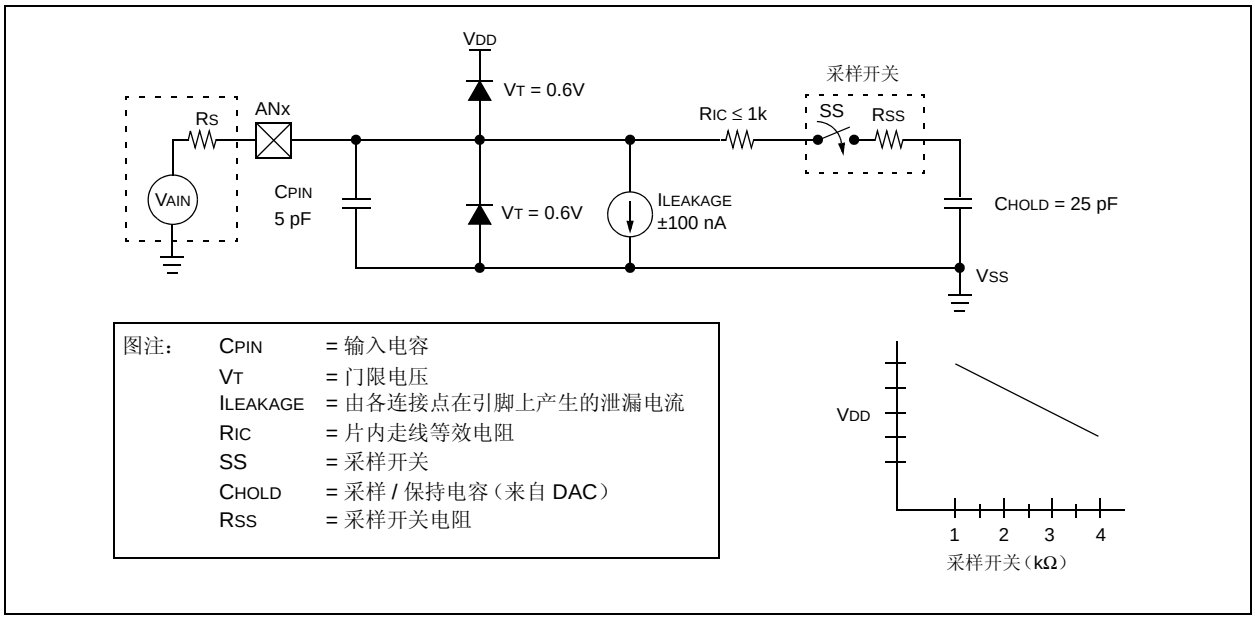
在根据需要配置好 A/D 模块之后，必须在转换开始之前对选定的通道进行采集。必须将模拟输入通道相应的 TRIS 位选择为输入。采集时间的确定，请参见**第 21.1 节 “A/D 采集要求 ”**。采集时间一结束，即可启动 A/D 转换。可将采集时间编程设定在 GO/DONE 位置 1 和实际转换启动之间。

在执行 A/D 转换时应该遵循以下步骤：

1. 配置 A/D 模块：
   - 配置模拟引脚、参考电压和数字 I/O （ADCON1）
   - 选择 A/D 输入通道 （ADCON0）
   - 选择 A/D 采集时间 （ADCON2）
   - 选择 A/D 转换时钟 （ADCON2）
   - 开启 A/D 模块 （ADCON0）
2. 需要时，配置 A/D 中断：
   - 清零 ADIF 位
   - 将 ADIE 位置 1
   - 将 GIE 位置 1
3. 如果需要，等待所需的采集时间。
4. 启动转换：
   - 将 GO/DONE 位 （ADCON0<1>）置 1
5. 等待 A/D 转换完成，可通过以下两种方法之一判断转换是否完成：
   - 查询 GO/DONE 位是否被清零

   或
   - 等待 A/D 中断
6. 读取 A/D 结果寄存器（ADRESH:ADRESL），需要时将 ADIF 位清零。
7. 如需进行下一次 A/D 转换，返回步骤 1 或步骤 2。每位的 A/D 转换时间定义为 TAD。在下一次采集开始前需要等待至少 2 TAD 的时间。

图 21-2： 模拟输入模型



21.1 A/D 采集要求

为了使 A/D 转换器达到规定的精度，必须使充电保持电容（CHOLD）充满至输入通道的电压值。模拟输入模型见图 21-2。模拟信号源阻抗（RS）和内部采样开关阻抗（RSS）直接影响为电容 CHOLD 充电所需的时间。采样开关阻抗值（RSS）随器件电压（VDD）不同而改变。模拟输入信号源阻抗会影响模拟输入的失调电压（由于引脚泄漏电流的原因）。**模拟信号源的最大阻抗推荐值为 2.5 kΩ**。选择（改变）模拟输入通道后，必须对通道进行采样才能启动转换，采样时间必须大于最小采集时间。

注：	当启动转换时，要将保持电容与输入引脚断开。

可以使用公式 21-1 来计算最小采集时间。该公式假设误差为 1/2 LSb（A/D 转换需要 1024 步）。1/2 LSb 的误差是 A/D 达到规定分辨率所能允许的最大误差。

公式 21-3 说明了所需的最小采集时间 TACQ 的计算过程。计算结果基于以下对应用系统的假设：

CHOLD	=	25 pF
RS	=	2.5 kΩ
转换误差	≤	1/2 LSb
VDD	=	3V → RSS = 2 kΩ
温度	=	85°C （系统最大值）

公式 21-1： A/D 采集时间

$$T_{ACQ} = \text{放大器稳定时间} + \text{保持电容充电时间} + \text{温度系数}$$
$$= T_{AMP} + T_C + T_{COFF}$$

公式 21-2： A/D 最小充电时间

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \bullet (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$

或

$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S)\ln(1/2048)$$

公式 21-3： 计算所需要的最小采集时间

$$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$$

$$T_{AMP} = 0.2\ \mu s$$

$$T_{COFF} = (\text{温度} - 25°C)(0.02\ \mu s/°C)$$
$$= (85°C - 25°C)(0.02\ \mu s/°C)$$
$$= 1.2\ \mu s$$

只有在温度 > 25°C 时才需要温度系数。当温度低于 25°C 时， $T_{COFF} = 0$ ms。

$$T_C = -(C_{HOLD})(R_{IC} + R_{SS} + R_S)\ln(1/2048)\ \mu s$$
$$= -(25\ pF)(1\ k\Omega + 2\ k\Omega + 2.5\ k\Omega)\ln(0.0004883)\ \mu s$$
$$= 1.05\ \mu s$$

$$T_{ACQ} = 0.2\ \mu s + 1\ \mu s + 1.2\ \mu s$$
$$= 2.4\ \mu s$$

21.2 选择和配置自动采集时间

用户可以利用 ADCON2 寄存器选择采集时间，该采集时间发生在每次 GO/DONE 位置 1 之后。

当 GO/DONE 位被置 1 时，采样停止并启动转换。用户必须确保在选择所需的输入通道和将 GO/DONE 位置 1 之间已经过了所需的采集时间。这发生在 ACQT<2:0> 位 （ADCON2<5:3>）保持在其复位状态 （000）的情况下，与不提供可编程采集时间的器件相兼容。

如果需要，可设置 ACQT 位以便为 A/D 模块选择可编程采集时间。当 GO/DONE 位被置 1 时，A/D 模块会继续在选定采集时间内采样输入通道，然后自动启动一次转换。由于采集时间是可编程的，因此没有必要在选择通道和将 GO/DONE 位置 1 之间等待一个采集时间。

在这两种情况下，当转换完成时， GO/DONE 位均被清零， ADIF 标志位均被置 1 并且 A/D 开始再次对当前选定的通道进行采样。如果采集时间已被编程，那么将不会有任何采集时间结束、转换开始的指示。

21.3 选择 A/D 转换时钟

每位的 A/D 转换时间定义为 TAD。每完成一次 10 位 A/D 转换需要 11 个 TAD。可用软件选择 A/D 转换的时钟源。

TAD 有以下 7 种可能的选择：

- 2 TOSC
- 4 TOSC
- 8 TOSC
- 16 TOSC
- 32 TOSC
- 64 TOSC
- 内部 RC 振荡器

为了实现正确的 A/D 转换， A/D 转换时钟 （TAD）必须尽可能小，但它必须大于最小 TAD （更多信息，请参见表 28-25 中的参数 130）。

表 21-1 给出了器件在不同工作频率下和选择不同的 A/D 时钟源时得到的 TAD。

表 21-1： 不同器件工作频率下的 TAD

A/D 时钟源 （TAD）		最高器件频率
工作状态	ADCS<2:0>	
2 TOSC	000	2.86 MHz
4 TOSC	100	5.71 MHz
8 TOSC	001	11.43 MHz
16 TOSC	101	22.86 MHz
32 TOSC	010	40.0 MHz
64 TOSC	110	40.0 MHz
RC(2)	x11	1.00 MHz(1)

注 1： RC 时钟源的典型 TAD 时间为 4 µS。

2： 当器件工作频率高于 1 MHz 时，整个转换过程必须在休眠模式下进行，否则 A/D 转换精度可能超出规范。

21.4 配置模拟端口引脚

ADCON1、 TRISA、 TRISF 和 TRISH 寄存器控制 A/D 端口引脚的操作。必须将配置为模拟输入的端口引脚对应的 TRIS 位置 1 （输入）。如果将 TRIS 位清零 （输出），则数字输出电平 （VOH 或 VOL）将被转换。

A/D转换操作与CHS<3:0>位和TRIS位的状态无关。

注 1： 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按模拟输入进行转换。配置为数字输入的引脚上的模拟电平将被精确转换。

2： 定义为数字输入引脚上的模拟电平可能会导致数字输入缓冲器消耗的电流超出器件规范。

21.5 A/D 转换

图 21-3 显示了在 GO/DONE 位置 1 且 ACQT<2:0> 位被清零后 A/D 转换器的工作状态。转换在下一条指令执行之后开始，以允许器件在转换开始之前进入休眠模式。

图 21-4 显示了在 GO/DONE 位置 1，ACQT<2:0> 位被设置为 010，且在转换开始之前选择 4 TAD 采集时间后 A/D 转换器的工作状态。

在转换期间清零 GO/DONE 位将中止当前的 A/D 转换。不会用部分完成的 A/D 转换结果更新 A/D 结果寄存器对。这意味着 ADRESH:ADRESL 寄存器将仍然保存上一次的转换结果（或上一次写入 ADRESH:ADRESL 寄存器的值）。

在 A/D 转换完成或中止后，需要等待 2 个 TAD 才能开始下一次采集。等待时间一到，将自动开始对选定通道进行采集。

注：	**不应**在开启 A/D 模块的指令中将 GO/DONE 位置 1。

21.6 CCP2 触发信号的使用

CCP2 模块的“特殊事件触发信号”可以启动 A/D 转换。这需要将 CCP2M<3:0> 位（CCP2CON<3:0>）设置为 1011，并且使能 A/D 模块（ADON 位置 1）。发生触发事件时，GO/DONE 位被置 1，启动 A/D 采集和转换并将 Timer1（或 Timer3）计数器复位为 0。复位 Timer1（或 Timer3）可自动重复 A/D 采集周期，最大限度地降低了软件开销（将 ADRESH:ADRESL 内容传送到所需存储单元）。特殊事件触发信号将 GO/DONE 位置 1（启动转换）之前，用户必须选择正确的模拟输入通道和最小采集周期，最小采集周期由用户设定或通过选择适当的 TACQ 时间来设定。

如果未使能 A/D 模块（ADON 清零），则特殊事件触发信号将被 A/D 模块忽略，但它仍会将 Timer1（或 Timer3）计数器复位。

图 21-3： A/D 转换 TAD 周期（ACQT<2:0> = 000， TACQ = 0）

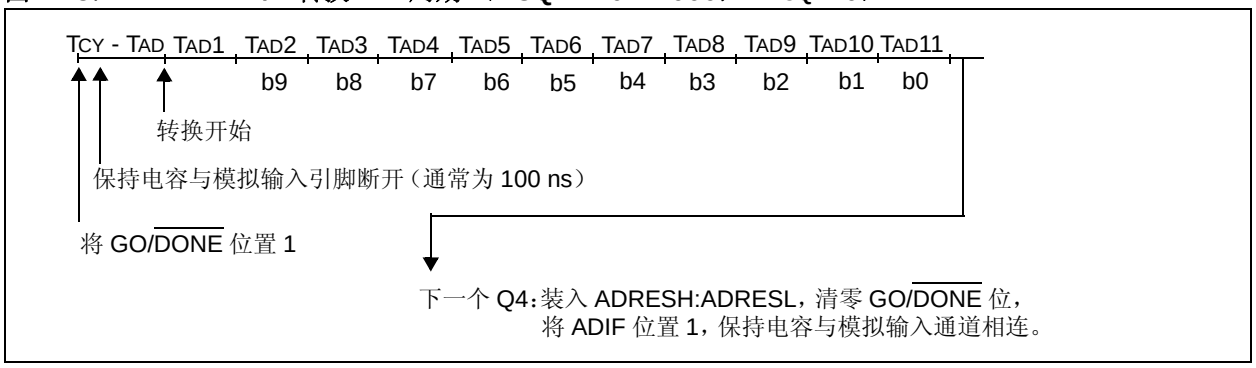
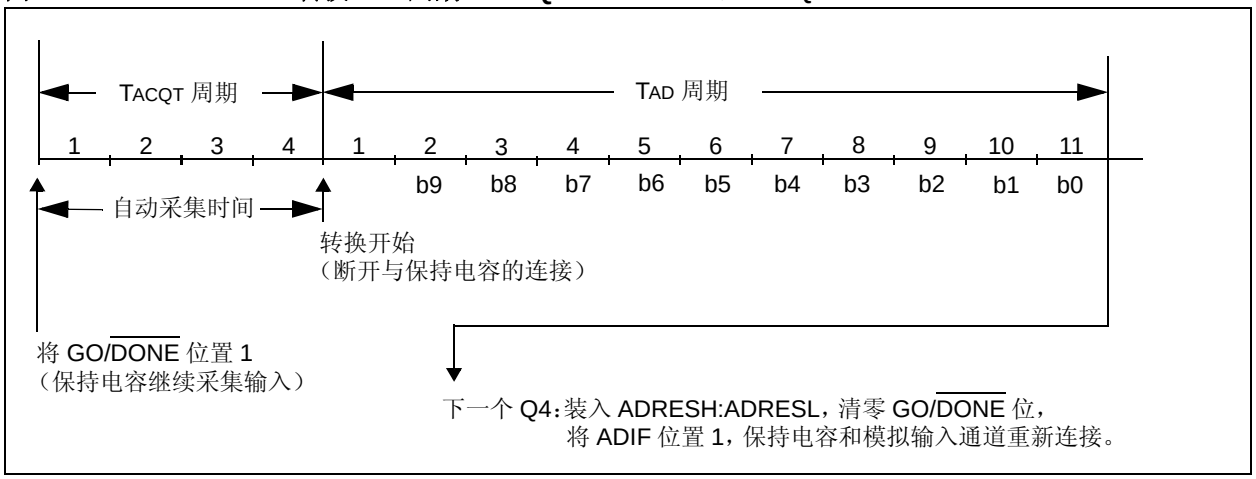


图 21-4： A/D 转换 TAD 周期（ACQT<2:0> = 010， TACQ = 4 TAD）



21.7 A/D 转换器校准

PIC18F87J90 系列器件中的 A/D 转换器包括自校准功能，能补偿模块中产生的任何偏移。校准过程是通过将 ADCAL 位（ADCON0<7>）置 1 而自动启动的。GO/DONE 位下次置 1 时，模块将执行“假”转换（即不读输入通道），将结果值在内部存储起来以补偿偏移。这样可实现后续偏移的补偿。

校准过程假定器件处于相对稳态工作条件下。如果使用了 A/D 校准功能，应在每次器件复位后或工作条件有重大变化时执行校准操作。

21.8 在功耗管理模式下的操作

在功耗管理模式下，自动采集时间和 A/D 转换时钟的选择一定程度上可由时钟源和频率决定。

如果希望器件处于功耗管理模式时 A/D 继续工作，就应该根据要在功耗管理模式下使用的时钟对 ADCON2 中的 ACQT<2:0> 和 ADCS<2:0> 位进行更新。在进入功耗管理模式后（两种功耗管理运行模式之一），就可以开始 A/D 采集或转换。采集或转换开始以后，器件应继续使用与功耗管理模式相同的时钟源直到转换完成。如果需要，在转换期间也可以将器件置于相应的功耗管理空闲模式。

如果功耗管理模式的时钟频率小于 1 MHz，就应该选择 A/D RC 时钟源。

在休眠模式下工作需要选择 A/D RC 时钟。如果将 ACQT<2:0> 位设置为 000 并启动转换，转换将延时一个指令周期以允许执行 SLEEP 指令并进入休眠模式。OSCCON 寄存器中的 IDLEN 和 SCSx 位必须在启动转换之前已被清零。

表 21-2： A/D 寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR1	—	ADIF	RC1IF	TX1IF	SSPIF	—	TMR2IF	TMR1IF	62
PIE1	—	ADIE	RC1IE	TX1IE	SSPIE	—	TMR2IE	TMR1IE	62
IPR1	—	ADIP	RC1IP	TX1IP	SSPIP	—	TMR2IP	TMR1IP	62
PIR3	—	LCDIF	RC2IF	TX2IF	CTMUIF	CCP2IF	CCP1IF	RTCCIF	62
PIE3	—	LCDIE	RC2IE	TX2IE	CTMUIE	CCP2IE	CCP1IE	RTCCIE	62
IPR3	—	LCDIP	RC2IP	TX2IP	CTMUIP	CCP2IP	CCP1IP	RTCCIP	62
ADRESH	A/D 结果寄存器的高字节								61
ADRESL	A/D 结果寄存器的低字节								61
ADCON0	ADCAL	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON	61
ADCON1	TRIGSEL	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	61
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	61
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	64
PORTA	RA7(1)	RA6(1)	RA5	RA4	RA3	RA2	RA1	RA0	63
TRISA	TRISA7(1)	TRISA6(1)	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	62
PORTF	RF7	RF6	RF5	RF4	RF3	RF2	RF1	—	62
TRISF	TRISF5	TRISF4	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	—	62

图注： — = 未实现，读为 0。 A/D 转换不使用阴影单元。

注 1： 仅当内部振荡器被选为默认时钟源（FOSC2 配置位 = 0）时， RA<7:6> 及其相关的锁存和方向位才能被配置为端口引脚；否则，这些位将被禁止且读为 0。

注：

22.0 比较器模块

模拟比较器模块包含两个比较器，可以用多种方式对它们进行配置。比较器的输入可以从与 RF1 到 RF6 引脚复用的模拟输入和片上参考电压中选择（见**第 23.0 节“比较器参考电压模块”**）。数字输出（常规或反相的）可从引脚电平读取，也可通过控制寄存器读取。

CMCON 寄存器（寄存器 22-1）选择比较器的输入和输出配置。图 22-1 给出了各种比较器配置的框图。

寄存器 22-1： CMCON：比较器模块控制寄存器

R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-1	R/W-1
C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0
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 **C2OUT**：比较器 2 输出位
当 C2INV = 0 时：
1 = C2 VIN+ > C2 VIN-
0 = C2 VIN+ < C2 VIN-
当 C2INV = 1 时：
1 = C2 VIN+ < C2 VIN-
0 = C2 VIN+ > C2 VIN-

bit 6 **C1OUT**：比较器 1 输出位
当 C1INV = 0 时：
1 = C1 VIN+ > C1 VIN-
0 = C1 VIN+ < C1 VIN-
当 C1INV = 1 时：
1 = C1 VIN+ < C1 VIN-
0 = C1 VIN+ > C1 VIN-

bit 5 **C2INV**：比较器 2 输出反相位
1 = C2 输出反相
0 = C2 输出不反相

bit 4 **C1INV**：比较器 1 输出反相位
1 = C1 输出反相
0 = C1 输出不反相

bit 3 **CIS**：比较器输入切换位
当 CM<2:0> = 110 时：
1 = C1 VIN- 连接到 RF5/AN10/CVREF/SEG23/C1INB
C2 VIN- 连接到 RF3/AN8/SEG21/C2INB
0 = C1 VIN- 连接到 RF6/AN11/SEG24/C1INA
C2 VIN- 连接到 RF4/AN9/SEG22/C2INA

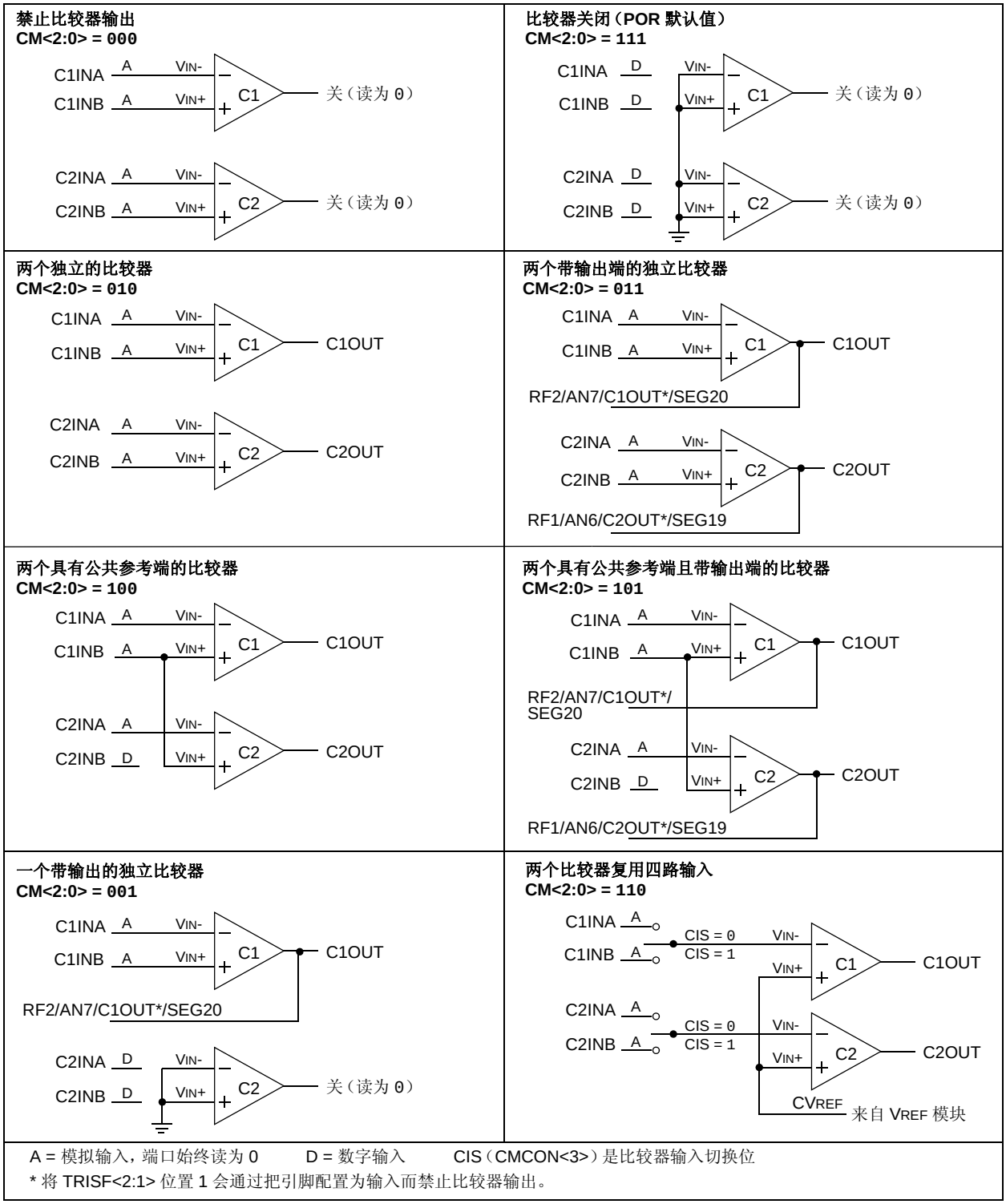
bit 2-0 **CM<2:0>**：比较器模式位
图 22-1 给出了比较器的几种模式以及相应 CM<2:0> 位的设置。

22.1 比较器配置

图 22-1 给出了比较器的 8 种工作模式。 CMCON 寄存器的 CM<2:0> 位用于选择模式。 TRISF 寄存器控制每种模式下比较器引脚的数据方向。如果改变比较器模式，由于存在指定的模式改变延时 （如**第 28.0 节 “ 电气特性 ”** 所示），比较器的输出电平可能会在此延时期间无效。

注：	改变比较器模式期间应禁止比较器中断；否则会产生错误的中断。

图 22-1： 比较器 I/O 工作模式



A = 模拟输入，端口始终读为 0　　D = 数字输入　　CIS（CMCON<3>）是比较器输入切换位

* 将 TRISF<2:1> 位置 1 会通过把引脚配置为输入而禁止比较器输出。

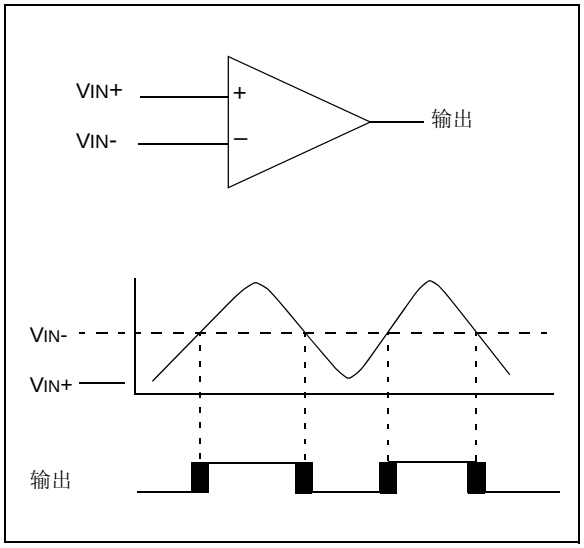
22.2 比较器工作原理

图 22-2 所示为单个比较器以及模拟输入电平与数字输出之间的关系。当 VIN+ 上的模拟输入电压小于 VIN- 上的模拟输入电压时，比较器输出数字低电平。当 VIN+ 上的模拟输入电压大于 VIN- 上的模拟输入电压时，比较器输出为数字高电平。图 22-2 中比较器输出的黑色区域表示因输入失调电压和响应时间所造成的输出不确定区域。

22.3 比较器参考电压

根据不同的比较器工作模式，可选择使用外部或内部参考电压。将 VIN- 上的模拟信号与 VIN+ 上的信号作比较，并相应地调整比较器的数字输出 （图 22-2）。

图 22-2： 单比较器



22.3.1 外部参考电压信号

当使用外部参考电压时，可将比较器模块中的两个比较器配置为使用同一个参考源或使用不同的参考源。但是，门限检测电路可能要求使用同一个参考源。参考信号必须在 VSS 和 VDD 之间，并且可被施加到比较器的任一引脚上。

22.3.2 内部参考电压信号

比较器模块也可以选择使用内部比较器参考电压模块产生的参考电压。在**第 23.0 节 “ 比较器参考电压模块 ”** 中详细介绍了该模块。

只有在两个比较器复用四路输入的模式（CM<2:0> = 110）中才可使用内部参考电压。在该模式下，内部参考电压被施加到两个比较器的 VIN+ 引脚上。

22.4 比较器响应时间

响应时间是指从选定一个新的参考电压或输入源到比较器输出达到一个有效电平的最短时间。如果内部参考电压改变了，在使用比较器输出时，必须考虑内部参考电压的最大延时。否则，应使用比较器的最大延时（见**第 28.0 节 “ 电气特性 ”**）。

22.5 比较器输出

通过 CMCON 寄存器读取比较器输出。这些位是只读的。比较器输出也可以直接输出到 RF1 和 RF2 I/O 引脚。当使能 RF1 和 RF2 引脚输出时，RF1 和 RF2 引脚输出路径上的多路开关会发生切换，并且每个引脚的输出与比较器输出是不同步的。每个比较器输出的不确定区域的大小与规范中给出的输入失调电压和响应时间有关。图 22-3 给出了比较器的输出框图。

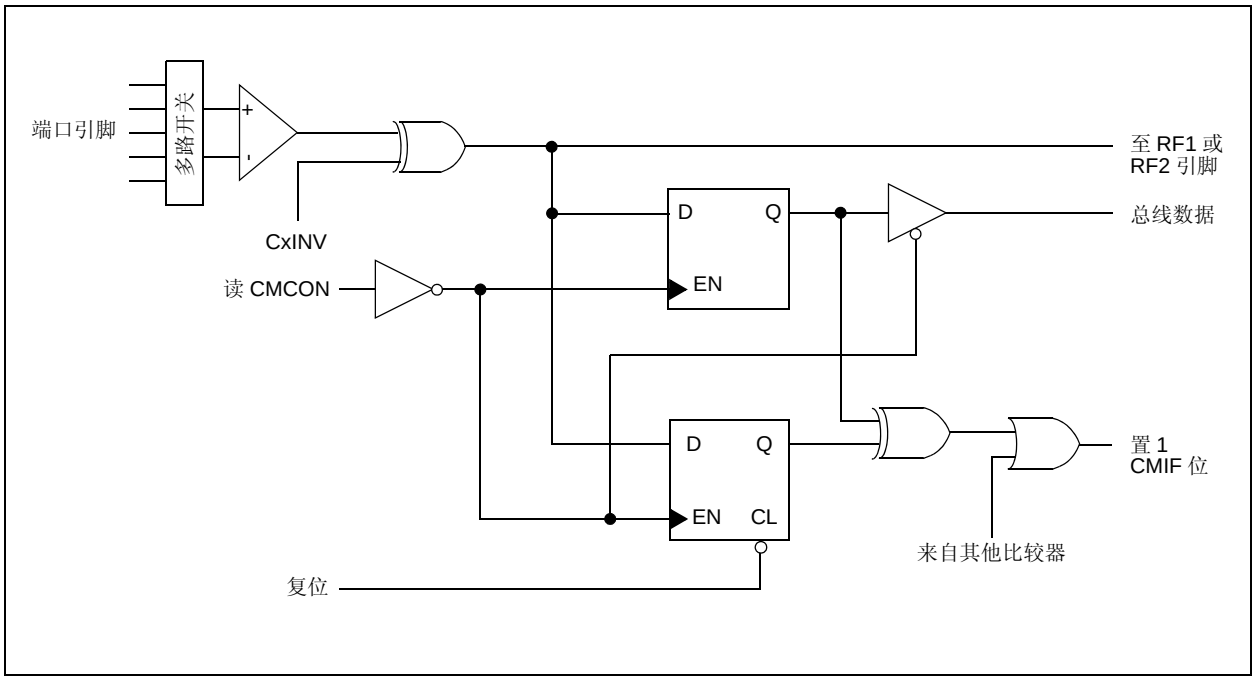
在该模式下，TRISF 仍作为 RF1 和 RF2 引脚的输出使能 / 禁止位。

使用 C2INV 和 C1INV 位（CMCON<5:4>）改变比较器输出的极性。

注 1： 读端口寄存器时，所有配置为模拟输入的引脚将读为 0。配置为数字输入的引脚将根据施密特触发器输入规范，对模拟输入进行相应转换。

2： 定义为数字输入引脚上的模拟电平可能会使输入缓冲器的电流消耗超过规定值。

图 22-3：　　比较器输出框图



22.6　比较器中断

任一比较器的输出值发生变化，都会将比较器中断标志位置1。需要用软件保存输出位的状态信息（从CMCON <7:6> 读取），以确定实际发生的变化。CMIF 位（PIR2<6>）是比较器中断标志位，必须通过清零复位。由于可以向该寄存器位写入 1，因此可以产生模拟的中断。

必须将 CMIE 位（PIE2<6>）和 PEIE 位（INTCON<6>）置1以允许中断。此外，还必须将GIE位（INTCON<7>）置 1。如果这些位中的任何一位被清零，将不会允许中断，尽管中断条件发生时仍会将 CMIF 位置 1。

注：	当执行读操作时（Q2周期开始），如果CMCON 寄存器（C1OUT 或 C2OUT）的值发生变化，那么 CMIF（PIR2<6>）中断标志位可能不会被置 1。

用户可用以下方式在中断服务程序中清除该中断：

a)　任何对 CMCON 的读或写都将结束不匹配条件。

b)　将标志位 CMIF 清零。

不匹配条件将继续把标志位 CMIF 置 1。读 CMCON 将结束不匹配条件并允许将标志位 CMIF 清零。

22.7　休眠期间的比较器操作

当比较器处于工作状态而器件处于休眠模式时，比较器仍保持工作状态并可产生中断（如果中断被允许）。中断会把器件从休眠模式唤醒。每个处于工作状态中的比较器都会消耗额外的电流，如比较器规范中所示。若要将休眠模式下的功耗降至最低，可在进入休眠模式前关闭比较器（CM<2:0> = 111）。如果器件从休眠状态唤醒，CMCON 寄存器的内容不受影响。

22.8　复位的影响

器件复位强制 CMCON 寄存器进入复位状态，从而使比较器模块进入关闭模式（CM<2:0> = 111）。但是，器件复位时输入引脚（RF3 到 RF6）被默认配置为模拟输入。这些引脚的 I/O 配置由 PCFG<3:0> 位（ADCON1<3:0>）的设置决定。因此，当复位时引脚呈现模拟输入状态，将使得器件电流降至最小。

22.9 模拟输入连接注意事项

模拟输入的简化电路如图22-4所示。由于模拟引脚被连接到数字输出端，它们与 VDD 和 VSS 之间连有反向偏置的二极管。因此，模拟输入必须在 VSS 和 VDD 之间。

如果输入电压超出该范围 0.6V 以上，就可能发生一个二极管正向偏置而使输入电压钳位。模拟信号源的最大阻抗推荐值为 10 kΩ。任何连接到模拟输入引脚的外部元件（如电容或齐纳二极管），要保证其泄漏电流极小。

图 22-4：　　比较器模拟输入模型

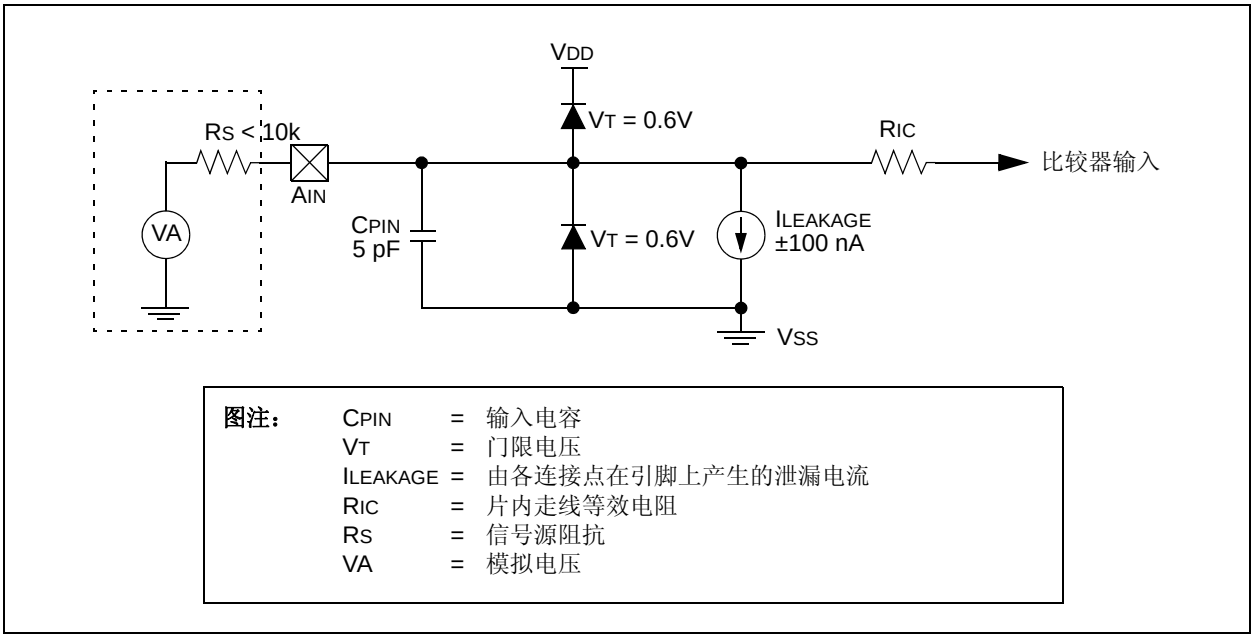


表 22-1：　　与比较器模块相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	59
PIR2	OSCFIF	CMIF	—	—	BCLIF	LVDIF	TMR3IF	—	62
PIE2	OSCFIE	CMIE	—	—	BCLIE	LVDIE	TMR3IE	—	62
IPR2	OSCFIP	CMIP	—	—	BCLIP	LVDIP	TMR3IP	—	62
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	61
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	61
PORTF	RF7	RF6	RF5	RF4	RF3	RF2	RF1	—	62
LATF	LATF7	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	—	62
TRISF	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	—	62

图注：　— = 未实现，读为 0。比较器模块不使用阴影单元。

注：

23.0 比较器参考电压模块

比较器参考电压模块是一个 16 阶的梯形电阻网络，可提供多个参考电压供选择。虽然它的主要目的是为模拟比较器提供参考电压，但也可将它用于其他场合。

图 23-1 给出了该模块的框图。梯形电阻经过分段可提供两种范围的 CVREF 值，并且该网络还具有断电功能，可以在不使用参考电压的情况下节省功耗。器件的 VDD/VSS 或外部参考电压都可以作为该模块的参考电压源。

23.1 配置比较器参考电压

比较器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。

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），该步长由 CVREF 选择位（CVR<3:0>）来决定。下面是计算比较器参考电压输出值的公式：

如果 CVRR = 1：
$CV_{REF} = ((CVR<3:0>)/24) \times (CV_{RSRC})$

如果 CVRR = 0：
$CV_{REF} = (CV_{RSRC}/4) + ((CVR<3:0>)/32) \times (CV_{RSRC})$

比较器参考电压模块的电压源可以来自 VDD 和 VSS，也可以来自与 RA2 和 RA3 复用的外部 VREF+ 和 VREF-。电压源由 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（见**第 28.0 节“电气特性”**中的表 28-3）。

寄存器 23-1： CVRCON：比较器参考电压控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CVREN	CVROE(1)	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 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)
1 = CVREF 电压也从 RF5/AN10/CVREF/SEG23/C1INB 引脚输出
0 = CVREF 电压从 RF5/AN10/CVREF/SEG23/C1INB 引脚断开

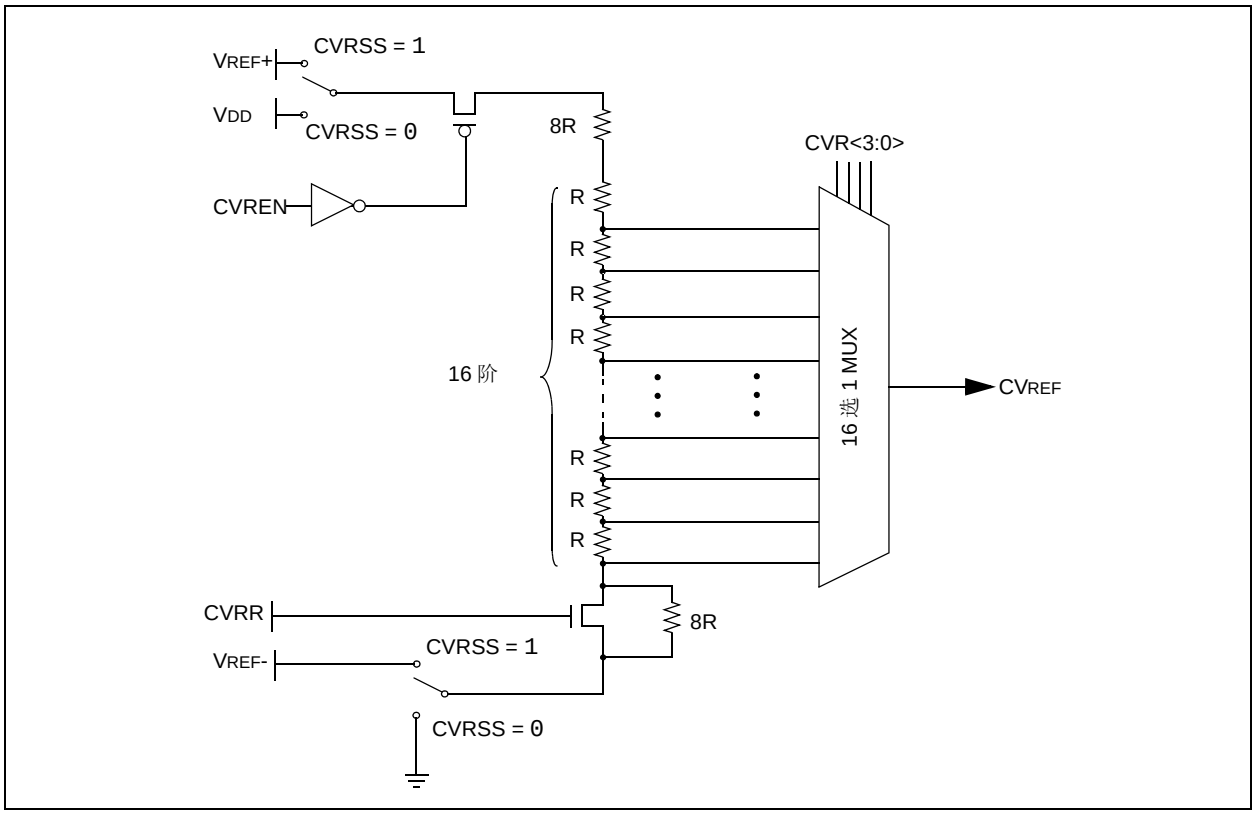
bit 5 **CVRR**：比较器 VREF 范围选择位
1 = 0 到 0.667 CVRSRC，步长为 CVRSRC/24（低电压范围）
0 = 0.25 CVRSRC 到 0.75 CVRSRC，步长为 CVRSRC/32（高电压范围）

bit 4 **CVRSS**：比较器 VREF 源选择位
1 = 比较器参考电压源， $CV_{RSRC} = (V_{REF+}) - (V_{REF-})$
0 = 比较器参考电压源， $CV_{RSRC} = V_{DD} - V_{SS}$

bit 3-0 **CVR<3:0>**：比较器 VREF 值选择位（$0 \le (CVR<3:0>) \le 15$）
当 CVRR = 1 时：
$CV_{REF} = ((CVR<3:0>)/24) \bullet (CV_{RSRC})$
当 CVRR = 0 时：
$CV_{REF} = (CV_{RSRC}/4) + ((CVR<3:0>)/32) \bullet (CV_{RSRC})$

注 1： CVROE 改写 TRISF<5> 位设置。

图 23-1：　　比较器参考电压模块框图



23.2　参考电压精度 / 误差

由于模块结构的限制，并不能实现整个参考电压范围的满量程输出。梯形电阻网络顶部和底部的晶体管（图 23-1）使 CVREF 值不能达到参考电压源的满幅值。参考电压是由参考电压源分压而来的，因此 CVREF 输出随参考电压源的波动而变化。经过测试的参考电压的绝对精度，请参见**第 28.0 节 “ 电气特性 ”**。

23.3　休眠期间的操作

如果因中断或看门狗定时器超时将器件从休眠模式唤醒，CVRCON 寄存器的内容将不受影响。为了降低休眠模式下的电流消耗，应禁止参考电压模块。

23.4　复位的影响

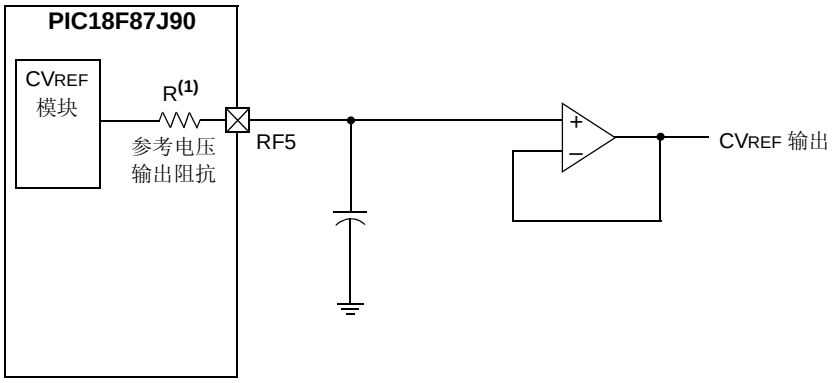
器件复位时，CVREN 位（CVRCON<7>）将被清零从而禁止参考电压模块。复位还将 CVROE 位（CVRCON <6>）清零，使参考电压从 RA2 引脚断开；同时通过将 CVRR 位（CVRCON<5>）清零选择高电压范围。CVR 值选择位也将清零。

23.5　连接注意事项

参考电压模块的工作独立于比较器模块。如果 CVROE 位被置 1，那么参考电压发生器的输出可能与 RF5 引脚相连。当 RA2 被配置为数字输入时，将参考电压输出到 RA2 引脚，将会增加电流消耗。使能 CVRSS 时，将 RF5 连接到数字输出也将增加电流消耗。

RF5 引脚可被用作简单的 D/A 输出，但是其驱动能力有限。要提高电流驱动能力，VREF 参考电压输出端必须外接缓冲器。图 23-2 举例说明了这一缓冲技术。

图 23-2：　　参考电压输出缓冲示例



注　　1： R 的值取决于比较器参考电压位 CVRCON<5> 和 CVRCON<3:0>。

表 23-1：　　与比较器参考电压相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	61
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	61
TRISF	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	—	62

图注：　　— = 未实现，读为 0。比较器参考电压不使用阴影单元。

注：

24.0 充电时间测量单元（CTMU）

充电时间测量单元（CTMU）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。CTMU 可与其他片上模拟模块配合使用，精确测量时间、电容、电容的相对变化，或生成具有特定延时的输出脉冲。CTMU 是与电容式传感器接口的理想选择。

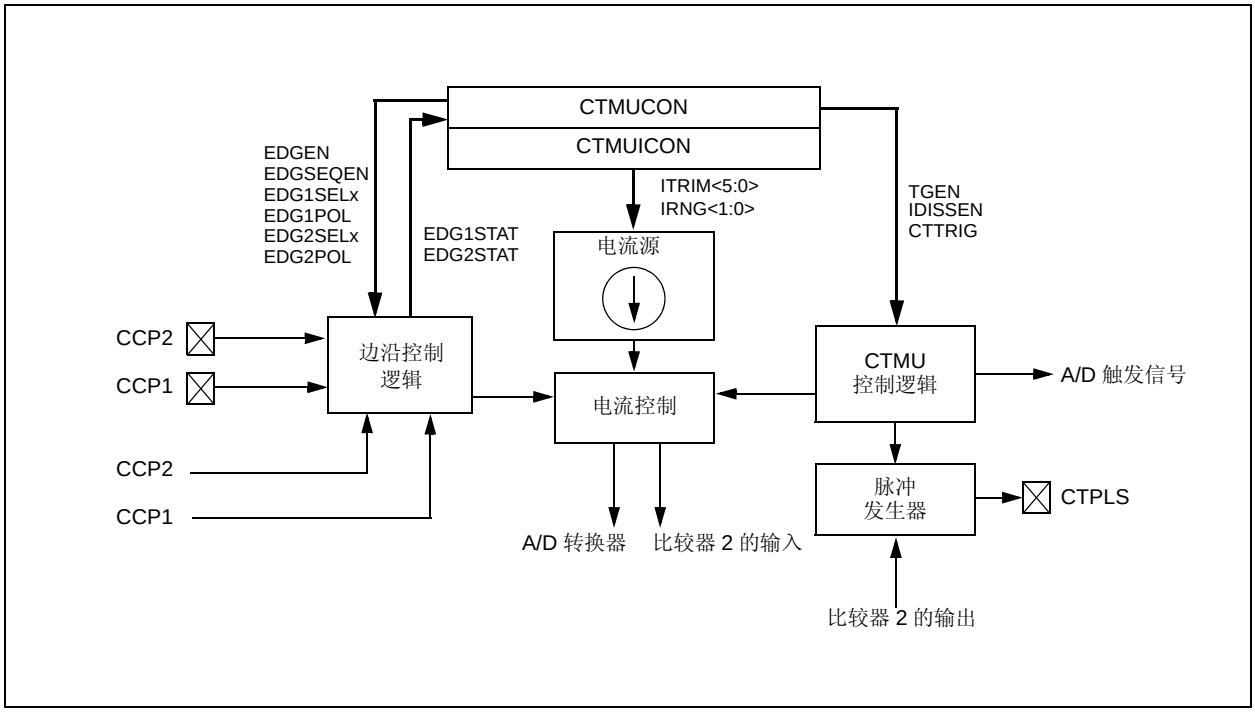
该模块具有以下主要特性：

- 最多 13 路通道，可用于电容或时间测量输入
- 片上精确电流源
- 4 个边沿输入触发源
- 每个边沿源的极性控制
- 边沿序列控制
- 控制对边沿的响应
- 1 纳秒的时间测量分辨率
- 高精度时间测量
- 与系统时钟异步的外部或内部信号的延时
- 适合测量电容的精确电流源

CTMU 与 A/D 转换器配合工作，根据具体器件和可用的 A/D 通道数，最多可提供 13 路通道用于时间或电荷测量。如果配置为产生延时，那么 CTMU 连接到其中一个模拟比较器。电平敏感输入边沿源可以从 4 个源中选择：两个外部输入或 CCP1/CCP2 特殊事件触发信号。

图 24-1 给出了 CTMU 的框图。

图 24-1： CTMU 框图



24.1 CTMU 工作原理

CTMU 的工作方式是使用固定电流源来对电路进行充电。电路的类型取决于要进行的测量的类型。在进行电荷测量的情况下，电流是固定的，向电路施加电流的时间也是固定的。这样，只要通过 A/D 测得电压就可以测得电路的电容。在进行时间测量的情况下，电流和电路的电容都是固定的。这种情况下，由 A/D 读取的电压可以代表从电流源开始对电路进行充电到停止充电经过的时间。

如果 CTMU 用于产生延时，那么电容和电流源，以及向比较器电路提供的电压都是固定的。信号的延时由将电压充电到比较器门限电压所需的时间决定。

24.1.1 工作原理

CTMU 的工作原理基于以下电荷公式：

$$C = I \cdot \frac{dV}{dT}$$

简单来说，在电路中测量的电荷（以库仑为单位）定义为：以安培为单位的电流（*I*）乘以以秒为单位的电流流动时间（*t*）。电荷也可以定义为：以法拉为单位的电容（*C*）乘以电路的电压（*V*）。可得：

$$I \cdot t = C \cdot V$$

CTMU 模块提供了恒定、已知的电流源。A/D 转换器用于测量公式中的电压（*V*），剩下两个未知量：电容（*C*）和时间（*t*）。以上公式可用于计算电容或时间，根据以下关系，使用电路的已知固定电容：

$$t = (C \cdot V)/I$$

或根据：

$$C = (I \cdot t)/V$$

使用电流源施加于电路的固定时间。

24.1.2 电流源

CTMU 的核心是高精度电流源，旨在提供用于测量的恒定基准。用户可以从三个范围或总共两个数量级的电流中选择电流等级，并可以按 ±2% 的增量（标称值）对输出进行微调。电流范围通过IRNG<1:0>位（CTMUICON<1:0>）进行选择，值 00 代表最低范围。

电流微调通过 ITRIM<5:0> 位（CTMUICON<7:2>）进行。这 6 个位使得可以按大约每步 2% 的步阶微调电流源。请注意，其中一半的范围用于正向调整电流源电平，另一半用于负向调整电流源电平。值 000000 是中性位置（无变化）。值 100000 代表最大负调整（大约 -62%），011111 代表最大正调整（大约 +62%）。

24.1.3 边沿选择和控制

CTMU 测量通过在模块的两路输入通道中发生的边沿事件进行控制。每路通道（称为边沿 1 和边沿 2）可以配置为接收来自一个边沿输入引脚（CTEDG1 和 CTEDG2）或 CCPx 特殊事件触发信号。输入通道是电平敏感通道，响应通道中的瞬时电平，而不是电平跳变。输入使用 EDG1SEL 和 EDG2SEL 位对（CTMUCONL<3:2 和 6:5>）选择。

除了电流源之外，还可以使用EDGE2POL和EDGE1POL位（CTMUCONL<7,4>）配置每路通道的事件极性。还可以对输入通道进行过滤以选择边沿事件顺序（边沿 1 在边沿 2 之前发生），方法是将 EDGSEQEN 位（CTMUCONH<2>）置 1。

24.1.4 边沿状态

CTMUCON 寄存器还包含两个状态位 EDG2STAT 和 EDG1STAT（CTMUCONL<1:0>）。它们的主要功能是显示在相应的通道中是否发生了边沿响应。当在通道中检测到边沿响应时，CTMU 会自动将特定的位置 1。输入通道的电平敏感特性也意味着，如果通道的配置或其电流状态发生改变，那么状态位会立即置 1。

模块使用边沿状态位来控制到外部模拟模块（如 A/D 转换器）的电流源输出。只有其中一个状态位置 1 而不是两个状态位同时置 1 时，才会向外部模块提供电流，如果两个位同时置 1 或同时清零，则会切断电流。这使 CTMU 可以仅测量两个边沿事件之间的电流。在两个状态位都置 1 后，必须先将它们清零，然后才能进行另一次测量。两个位应同时清零（如果可能），以避免重新使能 CTMU 电流源。

除了可以由 CTMU 硬件置 1 之外，边沿状态位也可以用软件置 1。也就是说可以在用户应用程序中手动使能或禁止电流源。将其中任意一位置 1（但不是同时置 1）即可使能电流源。将两位同时置 1 或清零即可立即禁止电流源。

24.1.5 中断

每当电流源先使能，然后禁止时，CTMU 就会将其中断标志位（PIR3<2>）置 1。只有相应的中断允许位（PIE3<2>）也置 1 时，才会产生中断。如果未使能边沿顺序（即，边沿 1 必须在边沿 2 之前发生），则需要监视边沿状态位，确定上次发生并导致中断的是哪一个边沿事件。

24.2 CTMU 模块初始化

以下过程是用于初始化 CTMU 模块的通用指南：

1. 使用 IRNG 位（CTMUICON<1:0>）选择电流源范围。
2. 使用ITRIM位（CTMUICON<7:2>）微调电流源。
3. 通过设置EDG1SEL和EDG2SEL位（CTMUCONL<3:2 和 6:5>）配置边沿 1 和边沿 2 的边沿输入源。
4. 使用 EDG1POL 和 EDG2POL 位（CTMUCONL<4,7>）配置边沿输入的输入极性。默认配置是使用负边沿极性（从高至低跳变）。
5. 使用 EDGSEQEN 位（CTMUCONH<2>）使能边沿序列。默认情况下，将禁止边沿顺序。
6. 使用 TGEN 位选择工作模式（测量或产生延时）。默认模式是时间 / 电容测量。
7. 使用 CTTRIG 位（CTMUCONH<0>）将模块配置为在发生第二个边沿事件时自动触发 A/D 转换。默认情况下，会禁止转换触发。
8. 通过将 IDISSEN 位（CTMUCONH<1>）置 1，对所连接电路放电；在等待足够时间，让电路完成放电之后，清零 IDISSEN。
9. 通过清零 CTMUEN 位（CTMUCONH<7>）禁止该模块。
10. 清零边沿状态位 EDG2STAT 和 EDG1STAT（CTMUCONL<1:0>）。
11. 通过将 EDGEN 位（CTMUCONH<3>）置 1 使能两个边沿输入。
12. 通过将 CTMUEN 位置 1 使能该模块。

根据要执行的测量或脉冲生成的类型，可能还需要再初始化和配置一个或更多其他模块，与 CTMU 模块配合使用：

- 边沿源生成：除了外部边沿输入引脚之外，CCPx 特殊事件触发信号也可以用作 CTMU 的边沿源。
- 电容或时间测量：CTMU 模块使用 A/D 转换器来测量连接到一路模拟输入通道的电容两端的电压。
- 脉冲生成：在生成独立于系统时钟的输出脉冲时，CTMU模块使用比较器2和关联的比较器参考电压。

24.3 校准 CTMU 模块

要精确测量电容和时间，以及产生精确延时，需要对 CTMU 进行校准。如果应用只需要测量电容或时间的相对变化，则通常不需要校准。此类应用的示例包括电容触摸开关，在这种应用中，触摸电路具有基本电容，所增加的人体电容会改变电路的总电容。

如果需要测量实际的电容或时间，则必须进行两项硬件校准：电流源需要进行校准，以使其提供精确的电流；要测量的电路也需要进行校准，以测量和 / 或抵消要测量电容之外的所有其他电容。

24.3.1 电流源校准

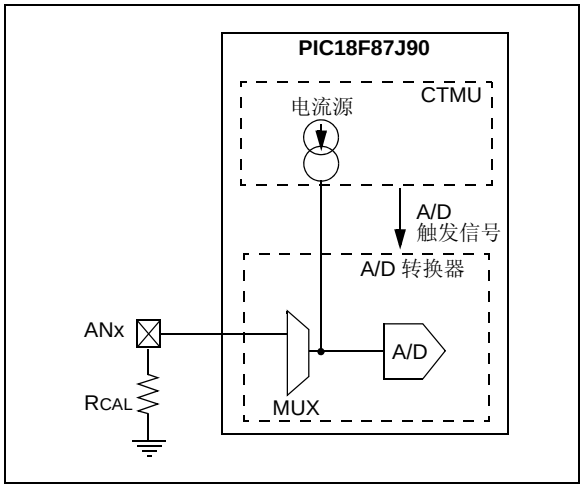
CTMU 模块随附的电流源具有三种电流范围，其中每种范围都可以在其标称值 ±60% 的范围内进行调节。因此，要进行精确测量，可以通过在未用模拟通道上放置一个高精度电阻 RCAL，测量并调整该电流源。图 24-2 给出了示例电路。电流源测量使用以下步骤执行：

1. 初始化 A/D 转换器。
2. 初始化 CTMU。
3. 通过将 EDG1STAT（CTMUCONL<0>）置 1 使能电流源。
4. 产生稳定时间延时。
5. 执行 A/D 转换。
6. 使用 $I = V/R_{CAL}$ 计算电流源电流；其中，RCAL 是高精度电阻，V 通过执行 A/D 转换来测量。

CTMU 电流源可以使用 CTMUICON 中的微调位进行微调，通过迭代过程来获取所需的精确电流。或者，也可以使用未经调整的标称值；可以由软件存储调整后的电流值，用于所有后续的电容或时间测量。

要计算 RCAL 的值，必须选择标称电流，然后就可以计算电阻。例如，如果 A/D 转换器参考电压为 3.3V，使用满量程的 70%（或 2.31V）作为要由 A/D 转换器读取的所需近似电压。如果 CTMU 电流源的范围选择为 0.55 µA，所需的电阻值使用 RCAL = 2.31V/0.55 µA 计算，得到值为 4.2 MΩ。类似地，如果电流源选择为 5.5 µA，RCAL 将为 420,000Ù ；如果电流源设置为 55 µA，则为 42,000Ù。

图 24-2： CTMU 电流源校准电路



选择满量程电压 70% 的值，以确保 A/D 转换器处于充分高于基底噪声的范围。请记住，如果选择了某个需要结合使用 CTMUICON 的微调位的精确电流，则可能需要对 RCAL 的电阻值进行相应调整。可能还需要再次调整 RCAL，以允许选择可用的电阻值。考虑将使用 CTMU 进行测量的电路所需的精度， RCAL 应选择可用的最高精度。建议最小精度是允许 0.1% 的误差。

以下示例给出了执行 CTMU 电流校准的一种典型方法。例 24-1 演示如何初始化 A/D 转换器和 CTMU；该程序是同时使用两个模块的应用的典型程序。例 24-2 演示了实际校准程序的一种方法。请注意，该方法需要手动触发 A/D 转换器，这么做是为了演示整个逐步过程。也可以通过将 CTMU 的 CTTRIG 位（CTMUCONH<0>）置 1 来自动触发转换。

例 24-1： CTMU 校准设置程序

```
#include "p18cxxx.h"
/*****************************************************************************/
/*Setup CTMU ****************************************************************/
/*****************************************************************************/
void setup(void)

{ //CTMUCON - CTMU Control register

    CTMUCONH = 0x00;              //make sure CTMU is disabled
    CTMUCONL = 0X90;
    //CTMU continues to run when emulator is stopped,CTMU continues
    //to run in idle mode,Time Generation mode disabled, Edges are blocked
    //No edge sequence order, Analog current source not grounded, trigger
    //output disabled, Edge2 polarity = positive level, Edge2 source =
    //source 0, Edge1 polarity = positive level, Edge1 source = source 0,
    // Set Edge status bits to zero


    //CTMUICON - CTMU Current Control Register
    CTMUICON = 0x01;              //0.55uA, Nominal - No Adjustment

/*****************************************************************************/
//Setup AD converter;
/*****************************************************************************/

    TRISA=0x04;                   //set channel 2 as an input

    // ADCON2
    ADCON2bits.ADFM=1;            // Resulst format 1= Right justified
    ADCON2bits.ACQT=1;            // Acquition time 7 = 20TAD 2 = 4TAD 1=2TAD
    ADCON2bits.ADCS=2;            // Clock conversion bits 6= FOSC/64 2=FOSC/32

    // ADCON1
    ADCON1bits.ADCAL=0;           // Normal A/D conversion operation
    ADCON1bits.PCFG=0xC;          // Configures AN0 to AN2 as analog

    // ADCON0
    ADCON0bits.VCFG0 =0;          // Vref+ = AVdd
    ADCON0bits.VCFG1 =0;          // Vref- = AVss
    ADCON0bits.CHS=2;             // Select ADC channel

    ADCON0bits.ADON=1;            // Turn on ADC

}
```

例 24-2：　　电流校准程序

```
#include "p18cxxx.h"

#define COUNT 500                       //@ 8MHz = 125uS.
#define DELAY for(i=0;i<COUNT;i++)
#define RCAL .027                       //R value is 4200000 (4.2M)
                                        //scaled so that result is in
                                        //1/100th of uA
#define ADSCALE 1023                    //for unsigned conversion 10 sig bits
#define ADREF 3.3                       //Vdd connected to A/D Vr+

int main(void)
{
    int i;
    int j = 0; //index for loop
    unsigned int Vread = 0;
    double VTot = 0;
    float Vavg=0, Vcal=0, CTMUISrc = 0;  //float values stored for calcs

//assume CTMU and A/D have been setup correctly
//see Example 25-1 for CTMU & A/D setup
setup();

CTMUCONHbits.CTMUEN = 1;                //Enable the CTMU
    for(j=0;j<10;j++)
    {
        CTMUCONHbits.IDISSEN = 1;       //drain charge on the circuit
        DELAY;                          //wait 125us
        CTMUCONHbits.IDISSEN = 0;       //end drain of circuit

        CTMUCONLbits.EDG1STAT = 1;      //Begin charging the circuit
                                        //using CTMU current source
        DELAY;                          //wait for 125us
        CTMUCONLbits.EDG1STAT = 0;      //Stop charging circuit

        PIR1bits.ADIF = 0;              //make sure A/D Int not set
        ADCON0bits.GO=1;                //and begin A/D conv.
        while(!PIR1bits.ADIF);          //Wait for A/D convert complete

        Vread = ADRES;                  //Get the value from the A/D
        PIR1bits.ADIF = 0;              //Clear A/D Interrupt Flag
        VTot += Vread;                  //Add the reading to the total
    }

    Vavg = (float)(VTot/10.000);        //Average of 10 readings
    Vcal = (float)(Vavg/ADSCALE*ADREF);
    CTMUISrc = Vcal/RCAL;               //CTMUISrc is in 1/100ths of uA

}
```

24.3.2 电容校准

内部 A/D 转换器采样电容和电路板走线与焊盘的杂散电容虽然容值较小，但仍会影响电容测量的精度。在确保先取下期望测量的电容的情况下，可以对杂散电容进行测量。然后，测量使用以下步骤执行：

1. 初始化 A/D 转换器和 CTMU。
2. 将 EDG1STAT 置 1 （= 1）。
3. 等待固定延时 *t*。
4. 清零 EDG1STAT。
5. 执行 A/D 转换。
6. 计算杂散电容和 A/D 采样电容：

$$C_{OFFSET} = C_{STRAY} + C_{AD} = (I \bullet t)/V$$

其中，I 从电流源测量步骤获知，t 是固定延时，V 通过执行 A/D 转换来测量。

然后，可以存储该测量值，用于时间测量时的计算，或在电容测量时减去该值。要进行校准，需要大致了解 C_{STRAY} + C_{AD} 的电容值。 C_{AD} 约为 4 pF。

可能需要使用一个迭代过程来调整时间 t，该时间是对电路进行充电，以从 A/D 转换器获得合理电压读数的时间。 t 的值可以通过将 C_{OFFSET} 设置为理论值，然后求解 t 来确定。例如，如果 C_{STRAY} 的理论计算值为 11 pF，V 预期为 V_{DD} 的 70% （或 2.31V），那么 t 为：

$$(4\ pF + 11\ pF) \bullet 2.31V/0.55\ \mu A$$

或 63 µs。

请参见例 24-3 了解 CTMU 电容校准的典型程序。

例 24-3：　　电容校准程序

```
#include "p18cxxx.h"

#define COUNT 25                        //@ 8MHz INTFRC = 62.5 us.
#define ETIME COUNT*2.5                 //time in uS
#define DELAY for(i=0;i<COUNT;i++)
#define ADSCALE 1023                    //for unsigned conversion 10 sig bits
#define ADREF 3.3                       //Vdd connected to A/D Vr+
#define RCAL .027                       //R value is 4200000 (4.2M)
                                        //scaled so that result is in
                                        //1/100th of uA

int main(void)
{
    int i;
    int j = 0;                          //index for loop
    unsigned int Vread = 0;
    float CTMUISrc, CTMUCap, Vavg, VTot, Vcal;

//assume CTMU and A/D have been setup correctly
//see Example 25-1 for CTMU & A/D setup
setup();

CTMUCONHbits.CTMUEN = 1;                //Enable the CTMU
    for(j=0;j<10;j++)
    {
        CTMUCONHbits.IDISSEN = 1;       //drain charge on the circuit
DELAY;                                  //wait 125us
        CTMUCONHbits.IDISSEN = 0;       //end drain of circuit

        CTMUCONLbits.EDG1STAT = 1;      //Begin charging the circuit
                                        //using CTMU current source
DELAY;                                  //wait for 125us
        CTMUCONLbits.EDG1STAT = 0;      //Stop charging circuit

        PIR1bits.ADIF = 0;              //make sure A/D Int not set
        ADCON0bits.GO=1;                //and begin A/D conv.
        while(!PIR1bits.ADIF);          //Wait for A/D convert complete


        Vread = ADRES;                  //Get the value from the A/D
        PIR1bits.ADIF = 0;              //Clear A/D Interrupt Flag
VTot += Vread;                          //Add the reading to the total
    }

    Vavg = (float)(VTot/10.000);        //Average of 10 readings
    Vcal = (float)(Vavg/ADSCALE*ADREF);
    CTMUISrc = Vcal/RCAL;               //CTMUISrc is in 1/100ths of uA
    CTMUCap = (CTMUISrc*ETIME/Vcal)/100;
}
```

24.4 使用 CTMU 测量电容

使用 CTMU 测量电容有两种相对独立的方法。第一种是绝对方法，该方法需要测量实际电容值。第二种是相对方法，该方法不需要实际电容，只需要电容的变化量。

24.4.1 绝对电容测量

对于绝对电容测量，应遵循**第 24.3 节 “ 校准 CTMU 模块 ”** 中的电流和电容校准步骤。然后，电容测量使用以下步骤执行：

1. 初始化 A/D 转换器。
2. 初始化 CTMU。
3. 将 EDG1STAT 置 1。
4. 等待固定延时 T。
5. 清零 EDG1STAT。
6. 执行 A/D 转换。
7. 计算总电容 CTOTAL = (I * T)/V ； 其中， I 从电流源测量步骤（**第 24.3.1 节 “ 电流源校准 ”**）获知，T 是固定延时， V 通过执行 A/D 转换来测量。
8. 从 CTOTAL 中减去杂散电容和 A/D 采样电容（COFFSET 来自**第 24.3.2 节 “ 电容校准 ”**），确定被测电容的值。

24.4.2 相对电荷测量

有些应用可能并不需要精确的电容测量。例如，在检测电容式开关的有效按压时，只需要检测电容的相对变化。在此类应用中，当开关打开 （未被触摸）时，总电容是电路板走线和 A/D 转换器等的组合电容。此时 A/D 转换器将会测量到较大的电压。当开关关闭 （被触摸）时，由于以上所列电容中增加了人体的电容，总电容增大， A/D 转换器将测量到较小的电压。

使用 CTMU 检测电容变化可以使用以下步骤简单实现：

1. 初始化 A/D 转换器和 CTMU。
2. 将 EDG1STAT 置 1。
3. 等待固定延时。
4. 清零 EDG1STAT。
5. 执行 A/D 转换。

通过执行 A/D 转换测量的电压可以指示相对电容。请注意，在这种情况下，不需要对电流源或电路电容测量进行校准。请参见例 24-4 了解电容触摸开关的软件程序示例。

例 24-4：　　用于电容触摸开关的程序

```
#include "p18cxxx.h"

#define COUNT 500                       //@ 8MHz = 125uS.
#define DELAY for(i=0;i<COUNT;i++)
#define OPENSW 1000                     //Un-pressed switch value
#define TRIP 300                        //Difference between pressed
                                        //and un-pressed switch
#define HYST 65                         //amount to change
                                        //from pressed to un-pressed
#define PRESSED 1
#define UNPRESSED 0

int main(void)
{
unsigned int Vread;                     //storage for reading
    unsigned int switchState;
    int i;

    //assume CTMU and A/D have been setup correctly
    //see Example 25-1 for CTMU & A/D setup
    setup();

CTMUCONHbits.CTMUEN = 1;                //Enable the CTMU

        CTMUCONHbits.IDISSEN = 1;       //drain charge on the circuit
DELAY;                                  //wait 125us
        CTMUCONHbits.IDISSEN = 0;       //end drain of circuit

        CTMUCONLbits.EDG1STAT = 1;      //Begin charging the circuit
                                        //using CTMU current source
DELAY;                                  //wait for 125us
        CTMUCONLbits.EDG1STAT = 0;      //Stop charging circuit

        PIR1bits.ADIF = 0;              //make sure A/D Int not set
        ADCON0bits.GO=1;                //and begin A/D conv.
        while(!PIR1bits.ADIF);          //Wait for A/D convert complete

        Vread = ADRES;                  //Get the value from the A/D

    if(Vread < OPENSW - TRIP)
    {
        switchState = PRESSED;
    }
    else if(Vread > OPENSW - TRIP + HYST)
    {
        switchState = UNPRESSED;
    }
}
```

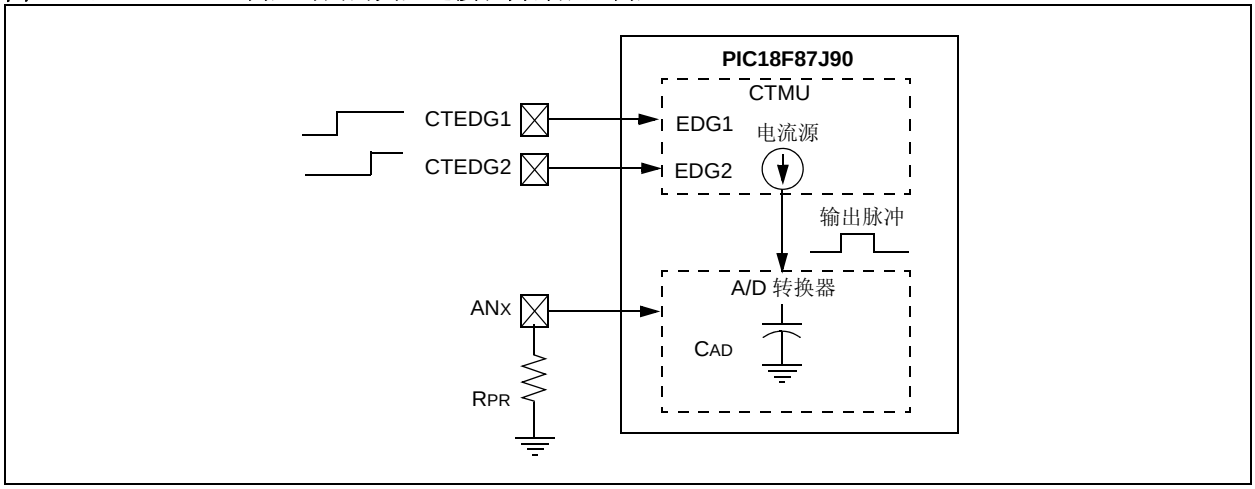
24.5 使用 CTMU 模块测量时间

通过电流和电容校准步骤测量比率（C/I）之后，可以使用以下步骤精确测量时间：

1. 初始化 A/D 转换器和 CTMU。
2. 将 EDG1STAT 置 1。
3. 将 EDG2STAT 置 1。
4. 执行 A/D 转换。
5. 根据 T = (C/I) * V 计算边沿之间的时间；其中，I 在电流校准步骤（**第 24.3.1 节 “ 电流源校准 ”**）中计算，C 在电容校准步骤（**第 24.3.2 节 “ 电容校准 ”**）中计算，V 通过执行 A/D 转换来测量。

假定所测量的时间足够小，电容 COFFSET 可以向 A/D 转换器提供有效的电压。要进行最小的时间测量，请始终将 A/D 通道选择寄存器（AD1CHS）设置为未用的 A/D 通道；该通道的相应引脚不连接到任何电路板走线。这可以最大程度减小所增加的杂散电容，保持总电路电容接近于 A/D 转换器自身的电容（25 pF）。要测量较长的时间间隔，可以将一个外部电容连接到 A/D 通道，并在进行时间测量时选择该通道。

图 24-3： 测量时间的典型连接和内部配置测量



24.6 使用 CTMU 模块产生延时

CTMU 模块具有一种独特功能，即它可以根据外部电容值产生独立于系统时钟的输出脉冲。这通过使用内部比较器参考电压模块、比较器2输入引脚和外部电容实现。脉冲输出到 CTPLS 引脚上。要使能该模式，需将 TGEN 位置 1。

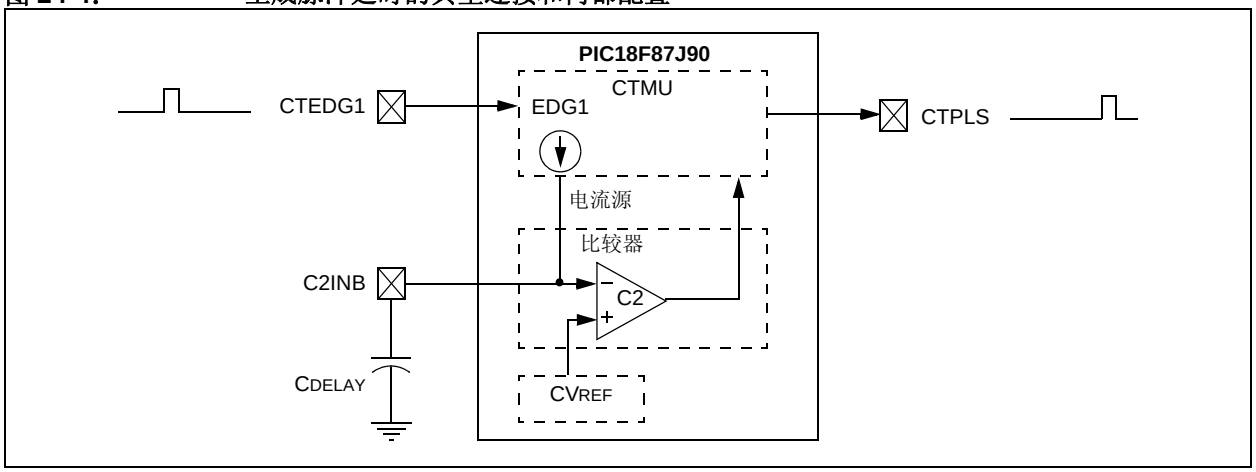
示例电路请参见图 24-4。CPULSE 由用户选择，用于确定 CTPLS 上的输出脉冲宽度。脉冲宽度根据 T = (CPULSE/I)*V 计算；其中，I 从电流源测量步骤（**第 24.3.1 节 “ 电流源校准 ”**）获知，V 是内部参考电压 （CVREF）。

该功能的使用示例是与基于可变电容的传感器进行连接，例如湿度传感器。当湿度发生变化时，CTPLS 上的脉宽输出也会变化。CTPLS输出引脚可以连接到输入捕捉引脚，通过测量变化的脉冲宽度来确定应用环境的湿度。

执行以下步骤来使用该功能：

1. 初始化比较器 2。
2. 初始化比较器参考电压。
3. 初始化 CTMU，并通过将 TGEN 位置 1 来使能延时生成。
4. 将 EDG1STAT 置 1。
5. 当CPULSE充电到参考电压跳变点的值时，在CTPLS上会产生输出脉冲。

图 24-4： 生成脉冲延时的典型连接和内部配置



24.7 休眠 / 空闲模式期间的操作

24.7.1 休眠模式和深度休眠模式

当器件进入休眠模式时， CTMU 模块电流源将始终禁止。如果调用休眠模式时，CTMU 正在执行依赖于电流源的操作，则操作可能不会正确终止。电容和时间测量可能会返回错误值。

24.7.2 空闲模式

CTMU在空闲模式下的行为由CTMUSIDL位（CTMUCONH<5>）决定。如果 CTMUSIDL 清零，在空闲模式下，模块将继续工作。如果 CTMUSIDL 置 1，则在器件进入空闲模式时，模块的电流源会被禁止。如果调用空闲模式时，模块正在执行操作，这种情况下，结果将类似于休眠模式下的结果。

24.8 复位对 CTMU 的影响

在复位时， CTMU 的所有寄存器都会被清零。这使 CTMU 模块处于禁止状态，它的电流源被关闭，所有配置选项恢复为它们的默认设置。在任意复位之后，模块都需要重新初始化。

如果发生复位时， CTMU 正在进行测量，测量结果将丢失。正在测量的电路可能会存在部分充电的情况，在随后 CTMU 尝试进行测量之前，应正确进行放电。电路放电方法是，在 A/D 转换器连接到相应通道的同时，先将 IDISSEN 位 （CTMUCONH<1>）置 1，然后再将其清零。

24.9 寄存器

有 3 个用于 CTMU 的控制寄存器：

- CTMUCONH
- CTMUCONL
- CTMUICON

CTMUCONH 和 CTMUCONL 寄存器（寄存器 24-1 和寄存器 24-2）包含一些控制位，这些控制位用于配置 CTMU 模块边沿源选择、边沿源极性选择、边沿顺序、A/D 触发、模拟电路电容放电和使能。 CTMUICON 寄存器（寄存器 24-3）包含一些用于选择电流源范围和电流源微调的位。

寄存器 24-1： CTMUCONH：CTMU 控制高字节寄存器

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	EDGEN	EDGSEQEN	IDISSEN	CTTRIG
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 **CTMUEN**：CTMU 使能位
1 = 使能模块
0 = 禁止模块

bit 6 **未实现**：读为 0

bit 5 **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 4 **TGEN**：时间生成使能位
1 = 使能边沿延时生成
0 = 禁止边沿延时生成

bit 3 **EDGEN**：边沿使能位
1 = 不阻止边沿
0 = 阻止边沿

bit 2 **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 1 **IDISSEN**：模拟电流源控制位
1 = 模拟电流源输出接地
0 = 模拟电流源输出不接地

bit 0 **CTTRIG**：触发控制位
1 = 使能触发信号输出
0 = 禁止触发信号输出

寄存器 24-2：　　CTMUCONL：CTMU 控制低字节寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
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 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 = CTEDG1 引脚
10 = CTEDG2 引脚
01 = CCP1 特殊事件触发信号
00 = CCP2 特殊事件触发信号

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

bit 3-2　**EDG1SEL<1:0>：**边沿 1 源选择位
11 = CTEDG1 引脚
10 = CTEDG2 引脚
01 = CCP1 特殊事件触发信号
00 = CCP2 特殊事件触发信号

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

寄存器 24-3： 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 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 7-2 **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 1-0 **IRNG<1:0>：**电流源范围选择位

11 = 100 x 基本电流
10 = 10 x 基本电流
01 = 基本电流等级 （标称值为 0.55 μA）
00 = 禁止电流源

表 24-1： 与 CTMU 模块相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
CTMUCONH	CTMUEN	—	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	CTTRIG	—
CTMUCONL	EDG2POL	EDG2SEL1	EDG2SEL0	EDG1POL	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT	—
CTMUICON	ITRIM5	ITRIM4	ITRIM3	ITRIM2	ITRIM1	ITRIM0	IRNG1	IRNG0	—

图注： — = 未实现，读为 0。 CTMU 模块不使用阴影单元。

注：

25.0 CPU 的特殊功能

PIC18F87J90 系列器件包含的一些特殊功能旨在最大限度地提高系统可靠性，并通过减少外部元件把系统成本降到最低。这些功能包括：

- 振荡器选择
- 复位：
  - 上电复位 （POR）
  - 上电延时定时器 （PWRT）
  - 振荡器起振定时器 （OST）
  - 欠压复位 （BOR）
- 中断
- 看门狗定时器 （WDT）
- 故障保护时钟监视器
- 双速启动
- 代码保护
- 在线串行编程

要根据具体应用对频率、功耗、精度和成本的要求来选择振荡器。在**第 3.0 节 “ 振荡器配置 ”** 中详细讨论了所有的选项。

在本数据手册的前面几章中已完整地讨论了器件的复位和中断。

除了为复位提供上电延时定时器和振荡器起振定时器之外，PIC18F87J90 系列器件还具有一个可配置的看门狗定时器，该定时器由软件控制。

器件自带的内部 RC 振荡器还提供了故障保护时钟监视器（Fail-Safe Clock Monitor，FSCM）和双速启动这两个额外的功能。FSCM 对外设时钟进行后台监视，并在外设时钟发生故障时自动切换时钟源。双速启动使得几乎可在启动发生那一刻立即执行代码，同时主时钟源进行其起振延时。

通过设置相应的配置寄存器位可以使能和配置所有这些功能。

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）来选择各种器件配置。这些配置位被映射到程序存储器以 300000h 开始的单元中。表 25-2 给出了完整的列表。从寄存器 25-1 至寄存器 25-6 详细说明了各配置位的功能。

25.1.1 配置 PIC18F87J90 系列器件的注意事项

PIC18F87J90 系列器件不再使用持久性存储寄存器存储配置信息。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 4 个字中，这些字被称为闪存配置字。配置字按表 25-2 中相同的次序存储在程序存储器中， CONFIG1L 位于地址最低的单元，CONFIG3H 位于地址最高的单元。在器件上电时这些数据被自动装入正确的配置寄存器。

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。

在上电复位时用于配置位的易失性存储单元始终复位为 1。对于其他类型的复位事件，将保留和使用先前已编程的值，而无需从程序存储器重新装入数据。

程序存储器中 CONFIG1H、CONFIG2H 和 CONFIG3H 的高 4 位也应为 1111。这使得在意外执行这些配置字单元的极少事件中，这些配置字作为 NOP 指令执行。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。

为了避免在代码执行期间配置被意外更改，所有的可编程配置位只可被写入一次。在上电周期内对一个位进行初始化编程之后就不能再次写该位了。改变器件的配置需要对器件重复上电。

表 25-1： 闪存配置字到配置寄存器的映射

配置字节	代码空间地址	配置寄存器地址
CONFIG1L	XXXF8h	300000h
CONFIG1H	XXXF9h	300001h
CONFIG2L	XXXFAh	300002h
CONFIG2H	XXXFBh	300003h
CONFIG3L	XXXFCh	300004h
CONFIG3H	XXXFDh	300005h

表 25-2： 配置位和器件 ID

寄存器名称		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	默认 / 未编程值 (1)
300000h	CONFIG1L	$\overline{\text{DEBUG}}$	XINST	STVREN	—	—	—	—	WDTEN	111- ---1
300001h	CONFIG1H	—(2)	—(2)	—(2)	—(2)	—(3)	CP0	—	—	---- 01--
300002h	CONFIG2L	IESO	FCMEN	—	LPT1OSC	T1DIG	FOSC2	FOSC1	FOSC0	11-1 1111
300003h	CONFIG2H	—(2)	—(2)	—(2)	—(2)	WDTPS3	WDTPS2	WDTPS1	WDTPS0	---- 1111
300004h	CONFIG3L	—(2)	—(2)	—(2)	—(2)	—	—	RTCOSC	—	---- --1-
300005h	CONFIG3H	—(2)	—(2)	—(2)	—(2)	—	—	—	CCP2MX	---- ---1
3FFFFEh	DEVID1	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	xxxx xxxx(4)
3FFFFFh	DEVID2	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	0000 10x1(4)

图注： x = 未知， — = 未实现。阴影单元未实现，读为 0。

注 1： 这些值反映出厂时和上电复位后的未编程状态。在所有其他复位状态中，配置字节保持原先的编程状态。

2： 程序存储器中这些位的值应始终为 1。这样可确保如果意外执行了这些单元，将会作为 NOP 指令执行。

3： 该位应始终保持为 0。

4： DEVID 值请参见寄存器 25-7 和寄存器 25-8。这些寄存器为只读寄存器，用户不能对其进行编程。

寄存器 25-1： CONFIG1L：配置寄存器 1 的低字节 （字节地址为 300000h）

R/WO-1	R/WO-1	R/WO-1	U-0	U-0	U-0	U-0	R/WO-1
$\overline{\text{DEBUG}}$	XINST	STVREN	—	—	—	—	WDTEN
bit 7							bit 0

图注：		
R = 可读位	WO = 一次性写入位	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 7 **$\overline{\text{DEBUG}}$：**后台调试器使能位
1 = 禁止后台调试器； RB6 和 RB7 被配置为通用 I/O 引脚
0 = 使能后台调试器； RB6 和 RB7 专用于在线调试

bit 6 **XINST：**扩展指令集使能位
1 = 使能指令集扩展和变址寻址模式
0 = 禁止指令集扩展和变址寻址模式 （传统模式）

bit 5 **STVREN：**堆栈上溢 / 下溢复位使能位
1 = 使能堆栈上溢 / 下溢复位
0 = 禁止堆栈上溢 / 下溢复位

bit 4-1 **未实现：**读为 0

bit 0 **WDTEN：**看门狗定时器使能位
1 = 使能 WDT
0 = 禁止 WDT （由 SWDTEN 位控制）

寄存器 25-2： CONFIG1H：配置寄存器 1 的高字节 （字节地址为 300001h）

U-0	U-0	U-0	U-0	U-0	R/WO-1	U-0	U-0
—(1)	—(1)	—(1)	—(1)	—(2)	CP0	—	—
bit 7							bit 0

图注：		
R = 可读位	WO = 一次性写入位	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 7-3 **未实现：**读为 0

bit 2 **CP0：**代码保护位
1 = 程序存储器不受代码保护
0 = 程序存储器受代码保护

bit 1-0 **未实现：**读为 0

注 1： 程序存储器中这些位的值应始终为 1。这样可确保如果意外执行了这些单元，将会作为 NOP 指令执行。
2： 该位应始终保持为 0。

寄存器 25-3： CONFIG2L：配置寄存器 2 的低字节 （字节地址为 300002h）

R/WO-1	R/WO-1	U-0	R/WO-1	R/WO-1	R/WO-1	R/WO-1	R/WO-1
IESO	FCMEN	—	LPT1OSC	T1DIG	FOSC2	FOSC1	FOSC0
bit 7							bit 0

图注：			
R = 可读位	WO = 一次性写入位	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 7 **IESO**：双速启动 （内部 / 外部振荡器切换）控制位
1 = 使能双速启动
0 = 禁止双速启动

bit 6 **FCMEN**：故障保护时钟监视器使能位
1 = 使能故障保护时钟监视器
0 = 禁止故障保护时钟监视器

bit 5 **未实现**：读为 0

bit 4 **LPT1OSC**：T1OSC/SOSC 功耗选择配置位
1 = 选择高功耗 T1OSC/SOSC 电路
0 = 选择低功耗 T1OSC/SOSC 电路

bit 3 **T1DIG**：T1CKI 用作数字输入时钟使能位
1 = 不使能 T1OSCEN， T1CKI 也可用作数字输入时钟
0 = 不使能 T1OSCEN， T1CKI 不能用作数字输入时钟

bit 2-0 **FOSC<2:0>**：振荡器选择位
111 = ECPLL OSC1/OSC2 作为主振荡器；带 PLL 使能的 ECPLL 振荡器； RA6 用作 CLKO 引脚
110 = EC OSC1/OSC2 作为主振荡器；带 FOSC/4 输出的外部时钟
101 = HSPLL OSC1/OSC2 作为主振荡器；带软件 PLL 控制的高速晶振 / 谐振器
100 = HS OSC1/OSC2 作为主振荡器；高速晶振 / 谐振器
011 = INTPLL1，带软件 PLL 控制的内部振荡器模块； FOSC/4 输出
010 = INTIO1 内部振荡器模块， RA6 引脚输出 FOSC/4 信号， RA7 用作 I/O 引脚
001 = INTPLL2，带软件 PLL 控制的内部振荡器模块， RA6 和 RA7 均用作 I/O 引脚
000 = INTIO2 内部振荡器模块， RA6 和 RA7 均用作 I/O 引脚

寄存器 25-4： CONFIG2H：配置寄存器 2 的高字节 （字节地址为 300003h）

U-0	U-0	U-0	U-0	R/WO-1	R/WO-1	R/WO-1	R/WO-1
—(1)	—(1)	—(1)	—(1)	WDTPS3	WDTPS2	WDTPS1	WDTPS0
bit 7							bit 0

图注：			
R = 可读位	WO = 一次性写入位	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 7-4 **未实现：**读为 0

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： 程序存储器中这些位的值应始终为 1。这样可确保如果意外执行了这些单元，将会作为 NOP 指令执行。

寄存器 25-5： CONFIG3L：配置寄存器 3 的低字节 （字节地址为 300004h）

U-0	U-0	U-0	U-0	U-0	U-0	R/WO-1	U-0
—(1)	—(1)	—(1)	—(1)	—	—	RTCOSC	—
bit 7							bit 0

图注：			
R = 可读位	WO = 一次性写入位	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 7-2 **未实现：**读为 0

bit 1 **RTCOSC：**RTCC 参考时钟选择位
1 = RTCC 使用 T1OSC/T1CKI 作为参考时钟
0 = RTCC 使用 INTRC 作为参考时钟

bit 0 **未实现：**读为 0

注 1： 程序存储器中这些位的值应始终为 1。这样可确保如果意外执行了这些单元，将会作为 NOP 指令执行。

寄存器 25-6： CONFIG3H：配置寄存器 3 的高字节 （字节地址为 300005h）

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/WO-1
—(1)	—(1)	—(1)	—(1)	—	—	—	CCP2MX
bit 7							bit 0

图注：			
R = 可读位	WO = 一次性写入位	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 7-1 **未实现：**读为 0

bit 0 **CCP2MX：**CCP2 复用位
1 = CCP2 与 RC1 复用
0 = CCP2 与 RE7 复用

注 1： 程序存储器中这些位的值应始终为 1。这样可确保如果意外执行了这些单元，将会作为 NOP 指令执行。

寄存器 25-7： DEVID1：PIC18F87J90 系列器件的器件 ID 寄存器 1

R	R	R	R	R	R	R	R
DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0
bit 7							bit 0

图注：
R = 只读位

bit 7-5 **DEV<2:0>：**器件 ID 位
101 = PIC18F87J90
100 = PIC18F86J90
001 = PIC18F67J90
000 = PIC18F66J90

bit 4-0 **REV<4:0>：**版本 ID 位
这些位用于表明器件版本。

寄存器 25-8： DEVID2：PIC18F87J90 系列器件的器件 ID 寄存器 2

R	R	R	R	R	R	R	R
DEV10(1)	DEV9(1)	DEV8(1)	DEV7(1)	DEV6(1)	DEV5(1)	DEV4(1)	DEV3(1)
bit 7							bit 0

图注：
R = 只读位

bit 7-0 **DEV<10:3>：**器件 ID 位 (1)
这些位与器件 ID 寄存器 1 中的 DEV<2:0> 位一起用于标识部件编号。
0101 0000 = PIC18F87J90 系列器件

注 1： DEV<10:3> 的值可能会与其他器件系列相同。特定器件总是通过使用整个 DEV<10:0> 位序列来标识的。

25.2 看门狗定时器（WDT）

PIC18F87J90 系列器件的 WDT 是由 INTRC 振荡器驱动的。当使能 WDT 时，其时钟源也将同时使能。WDT 定时周期的标称值为 4 ms，其稳定性与 INTRC 振荡器相同。

4 ms 的 WDT 定时周期将与 16 位后分频器的分频值相乘来得到更长的时间周期。通过配置寄存器 2H 中的 WDTPS 位来控制一个多路开关以对 WDT 后分频器的输出进行选择。因此可获得的定时周期范围为 4 ms 至 131.072 秒（2.18 分钟）。当发生以下任一事件时，WDT 和后分频器将被清零，这些事件包括：执行了 SLEEP 或 CLRWDT 指令，或者发生了时钟故障（主时钟或 Timer1 振荡器）。

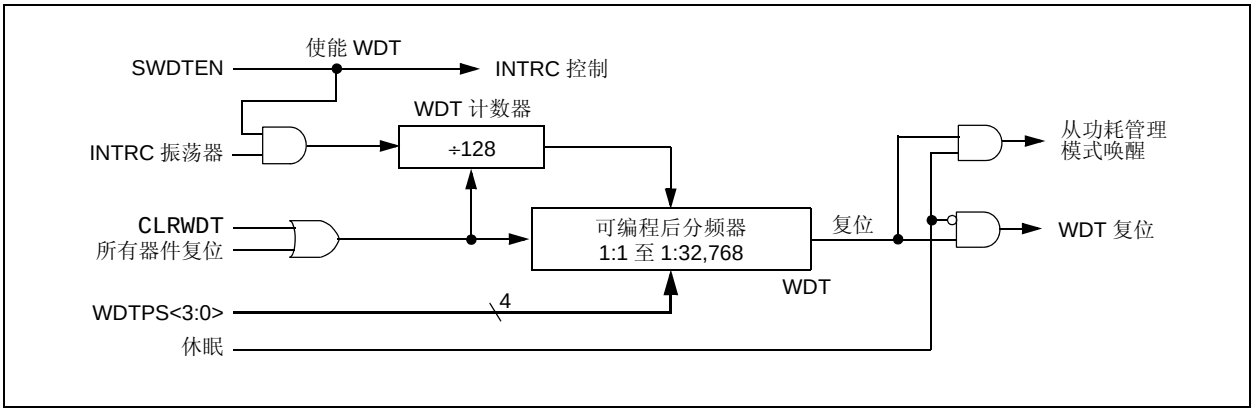
注 1： 当执行 CLRWDT 和 SLEEP 指令时，WDT 和后分频器的计数值将被清零。

2： 当执行 CLRWDT 指令时，后分频器的计数值将被清零。

25.2.1 控制寄存器

WDTCON 寄存器（寄存器 25-9）是可读写寄存器。SWDTEN 位使能或禁止 WDT 操作。仅当用配置位禁止 WDT 时，才允许用软件改写 WDTEN 配置位并使能 WDT。

图 25-1： WDT 框图



寄存器 25-9： WDTCON：看门狗定时器控制寄存器

R/W-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
REGSLP(1)	—	—	—	—	—	—	SWDTEN(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 7 **REGSLP**：稳压器低功耗工作使能位 (1)
1 = 当器件进入休眠模式时，片上稳压器进入低功耗工作状态
0 = 在休眠模式下，片上稳压器继续正常工作

bit 6-1 **未实现**：读为 0

bit 0 **SWDTEN**：软件控制的看门狗定时器使能位 (2)
1 = 打开看门狗定时器
0 = 关闭看门狗定时器

注 1： 当发生低压检测条件时， REGSLP 位被自动清零。
2： 当使能 WDTEN 配置位时该位不起作用。

表 25-3： 看门狗定时器寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
RCON	IPEN	—	$\overline{CM}$	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	60
WDTCON	REGSLP	—	—	—	—	—	—	SWDTEN	60

图注： — = 未实现，读为 0。看门狗定时器不使用阴影单元。

25.3 片上稳压器

所有的 PIC18F87J90 系列器件使用标称值 2.5V 的电压为其内核数字逻辑供电。对于需要工作在更高电压（如典型电压值为 3.3V）下的设计， PIC18F87J90 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 ENVREG 引脚控制该稳压器。把 VDD 连接到该引脚将使能稳压器，然后稳压后的电压通过其他 VDD 引脚向内核供电。当使能稳压器时，必须将一个低 ESR 滤波电容连接到 VDDCORE/VCAP 引脚（图 25-2），这有利于保持稳压器的稳定性。**第 28.3 节 “ 直流特性：PIC18F87J90 系列（工业级）”** 中提供了该滤波电容的推荐值。

如果 ENVREG 与 VSS 相连，则禁止稳压器。在这种情况下，必须使用独立的 2.5V 标称值的电源通过 VDDCORE/VCAP 引脚为器件的内核逻辑供电，从而将 I/O 引脚工作在一个较高的电压，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25-2了解可能的配置。

25.3.1 稳压和低压检测

当片上稳压器被使能时，它可以向数字内核逻辑提供标称值为 2.5V 的恒压。稳压器可从约 2.5V 直到最大达器件 VDDMAX 的 VDD 提供此电压。它不能使 VDD 电压降到 2.5V 以下。

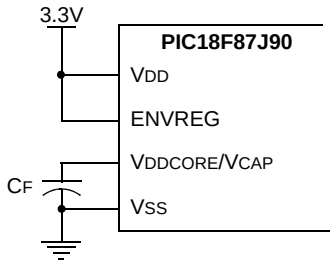
为防止提供给稳压器的电压降得过低时产生 “ 欠压 ” 条件，稳压器进入跟踪模式。在跟踪模式下，稳压器输出跟随 VDD，并带有 100 mV 的典型压降。

片上稳压器含有简单的低压检测 （Low-Voltage Detect，LVD）电路。如果 VDD 降得太低，无法在 VDDCORE 上维持约 2.45V 的电压，电路会将低压检测中断标志 LVDIF（PIR2<2>）置 1 并清零 REGSLP 位 （WDTCON<7>）（如果已置 1）。

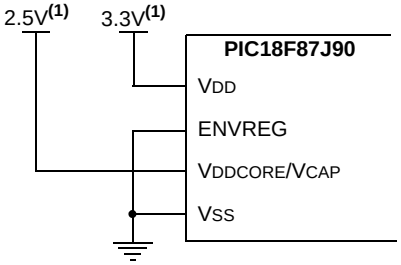
这可用于产生中断并将应用置于低功耗工作模式，或者触发器件按顺序关闭。低压检测仅在使能稳压器时才可用。

图 25-2： 片上稳压器连接

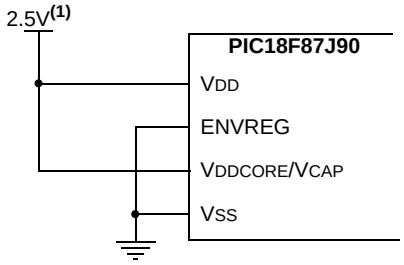
使能稳压器（ENVREG 连接到 VDD）：



禁止稳压器（ENVREG 接地）：



禁止稳压器（VDD 连接到 VDDCORE）：



注 1： 这些为典型的工作电压。关于VDD和VDDCORE的全部工作电压范围，请参见**第 28.1 节 “ 直流特性：PIC18F87J90 系列的供电电压（工业级） ”**。

25.3.2 片上稳压器和 BOR

当使能片上稳压器时，PIC18F87J90 系列器件也会有一个简单的欠压复位功能。如果向稳压器提供的电压下降到不足以维持全速工作所需的稳定输出，那么稳压器复位电路将产生欠压复位。BOR 标志位（RCON<0>）会捕捉该事件。

第 5.4 节“ 欠压复位 （BOR）” 和**第 5.4.1 节“ 检测 BOR”** 详细描述了 BOR 的工作原理。

25.3.3 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， V_{DDCORE} 决不能比 V_{DD} 高出 0.3V 或以上。

25.3.4 休眠模式下的操作

当片上稳压器使能时，它总是消耗比 I_{DD} 多一点的电流量。器件处于休眠模式时也是如此，即使内核数据逻辑不需要供电。要在功耗资源极其重要的应用中进一步节省功耗，可以将稳压器配置为每当器件进入休眠模式时就自动禁止。该功能由 REGSLP 位 （WDTCON<7>）控制。置 1 该位将禁止在休眠模式下使用稳压器，并将电流消耗降到最低。

通过将 REGSLP 位置 1，在休眠模式下可以节省大量功耗，但器件的唤醒时间将增加，以确保稳压器有足够的时间进行稳定。

当发生低压检测条件时，REGSLP 位被硬件自动清零。可以用软件将 REGSLP 位再次置 1，这将继续使稳压器保持在低功耗模式下。但是，如果将要对闪存执行任何写操作，则不推荐这样做。

25.4 双速启动

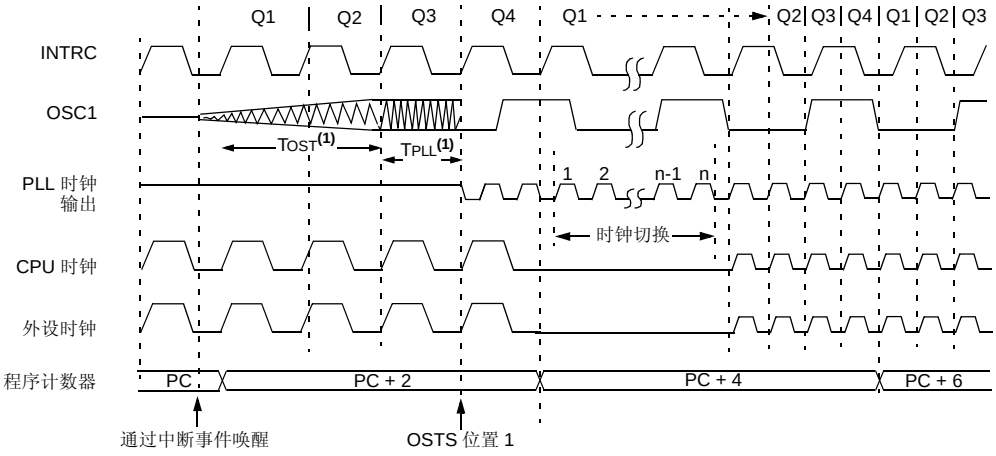
双速启动功能允许单片机在主时钟源稳定之前使用 INTRC 振荡器作为时钟源，从而帮助器件最大限度地缩短从振荡器起振到代码执行之间的延时。通过将 IESO 配置位置 1 可使能该功能。

仅当主振荡器模式为 HS 或 HSPLL （基于晶振）模式时才可使能双速启动。由于 EC 和 ECPLL 模式不需要 OST 起振延时，因此应禁止双速启动。

使能双速启动时，当复位或从休眠模式唤醒时，器件将被配置为：在使能上电复位后、接着上电延时定时器发生超时后，使用内部振荡器模块作为时钟源。这使得在主振荡器起振、OST 运行的同时，代码几乎立即开始执行。一旦OST超时，器件就自动切换到PRI_RUN模式。

在所有其他功耗管理模式下，不使用双速启动。器件将使用当前选定的时钟源直到主时钟源可用为止。 IESO 位的设置被忽略。

图 25-3： 双速启动时钟切换的时序图 （从 INTRC 切换到 HSPLL）



注 1： T_{OST} = 1024 T_{OSC}； T_{PLL} = 2 ms （近似值）。这些时间间隔未按比例绘制。

25.4.1 使用双速启动时的注意事项

当在双速启动模式中使用 INTRC 振荡器时，器件仍将遵守进入功耗管理模式的正常指令顺序，包括执行多条 SLEEP 指令（见**第 4.1.4 节 “ 多条 SLEEP 命令 ”**）。实际上，这意味着在 OST 超时前用户代码可以改变 SCS<1:0> 位的设置或执行 SLEEP 指令。这就使应用程序能短暂地唤醒器件，执行 “ 日常事务 ”，并在器件开始使用主时钟源前返回休眠状态。

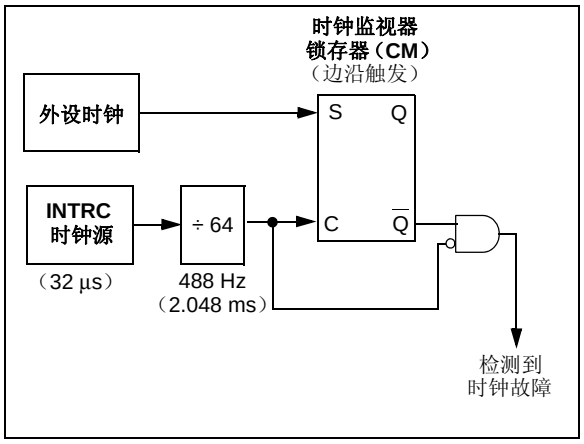
用户代码还能通过检查 OSTS 位（OSCCON<3>）的状态来确定主时钟源是否正在为器件系统提供时钟。如果该位置 1，则表示主振荡器正在为器件系统提供时钟。否则，表示当器件复位或从休眠模式唤醒期间由内部振荡器模块为器件系统提供时钟。

25.5 故障保护时钟监视器

故障保护时钟监视器（FSCM）可使单片机在外部振荡器发生故障时，自动将器件系统时钟切换到内部振荡器模块以保证器件能继续工作。将 FCMEN 配置位置 1 可使能 FSCM 功能。

当使能 FSCM 时，INTRC 振荡器将一直保持运行以监视外设时钟，并且在外设时钟发生故障时作为备用时钟。时钟监视（如图 25-4 所示）通过创建一个采样时钟信号实现，该信号为 INTRC 输出的 64 分频。这样就使得 FSCM 采样时钟脉冲之间有充足的时间间隔，从而保证在此期间至少有一个外设时钟边沿出现。外设时钟和采样时钟作为时钟监视（CM）锁存器的输入。CM 在器件时钟源的下降沿被置 1，在采样时钟的上升沿被清零。

图 25-4： FSCM 框图



在采样时钟的下降沿检测时钟故障。如果在出现采样时钟的下降沿时， CM 仍置 1，就表示检测到外部时钟故障（图 25-5）。这将引发以下事件：

- 通过将 OSCFIF（PIR2<7>）置 1，由 FSCM 产生振荡器故障中断；
- 器件时钟源切换到内部振荡器模块（OSCCON 不会被更新，因此无法显示当前时钟源——这就是故障保护状态）；并且
- WDT 复位。

切换过程中，对于时序敏感的应用，内部振荡器模块的后分频器频率可能不够稳定。在这些情况下，最好选择另一种时钟配置并进入其他功耗管理模式。可以尝试部分恢复或执行受控关闭。更多详细信息，请参见**第 4.1.4 节 “ 多条 SLEEP 命令 ”** 和**第 25.4.1 节 “ 使用双速启动时的注意事项 ”**。

FSCM 只能检测出主时钟源或辅助时钟源的故障。如果内部振荡器模块发生故障，将不会被检测到故障，当然也不可能采取任何措施。

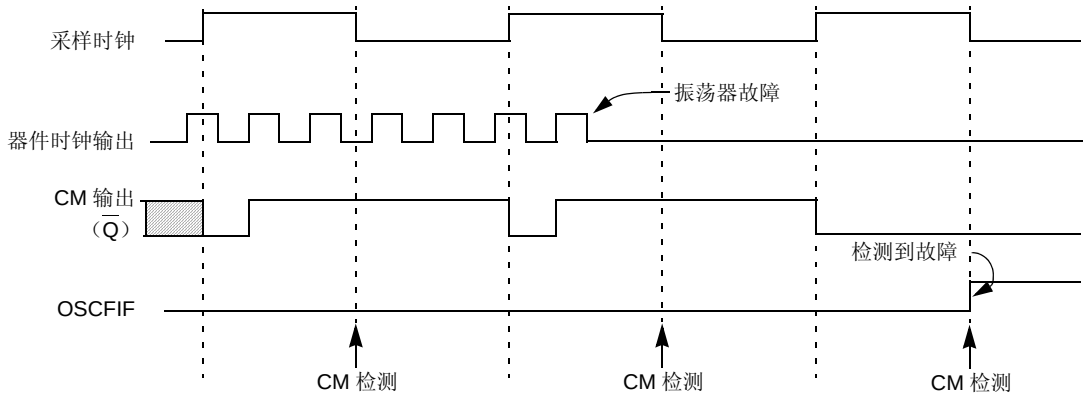
25.5.1 FSCM 和看门狗定时器

FSCM 和 WDT 均以 INTRC 振荡器作为时钟源。由于 WDT 使用独立的分频器和计数器，使能 FSCM 时，禁止 WDT 对 INTRC 振荡器的操作没有任何影响。

如前所述，当检测到时钟故障时，时钟源将切换到 INTRC 时钟；这可能意味着代码执行速度会发生很大的变化。如果使能 WDT 时使用的是小预分频值，时钟速度的下降将引起 WDT 超时，随后使器件复位。由于这个原因，故障保护时钟监视器事件也会使 WDT 和后分频器复位，使 WDT 从执行速度发生变化那一刻起开始重新计数，从而降低发生错误超时的可能。

如果禁止了该中断，处于空闲模式之后产生的中断将使 CPU 开始执行指令，同时以 INTRC 时钟源作为系统时钟源。

图 25-5：　　FSCM 时序图



注：　通常，器件时钟的频率比采样时钟频率高很多。本例中为便于说明选用相对频率。

25.5.2　退出故障保护工作模式

器件复位或进入功耗管理模式均可结束故障保护状态。发生复位时，控制器启动在配置寄存器 2H 中指定的主时钟源（伴有如 OST 或 PLL 定时器等所需的起振延时）。INTRC 振荡器将在主时钟源就绪之前提供器件系统时钟（类似于双速启动）。随后器件系统时钟源切换为主时钟（由 OSCCON 寄存器中的 OSTS 位置 1 指示）。然后，故障保护时钟监视器恢复对外设时钟的监视。

在起振期间，主时钟源可能永远不能就绪。在这种情况下，器件运行将以 INTOSC 多路开关作为时钟源。OSCCON 寄存器将保持复位状态直到进入功耗管理模式为止。

25.5.3　功耗管理模式下的 FSCM 中断

进入功耗管理模式时，时钟多路开关选择由 OSCCON 寄存器选定的时钟源。在该模式下将恢复对功耗管理时钟源的故障保护时钟监视。

如果在功耗管理工作模式下发生了振荡器故障，接下来的操作取决于是否允许了振荡器故障中断。如果允许了（OSCFIF = 1），代码执行将以 INTRC 多路开关的输出作为时钟源，并且不会自动转换回发生故障的时钟源。

25.5.4　POR 或从休眠中唤醒

FSCM 在器件退出上电复位（POR）或低功耗休眠模式后开始检测振荡器故障。当器件主时钟为 EC 或 INTRC 模式时，监视会在这些事件发生后立即开始。

对于 HS 或 HSPLL 模式，情况会有所不同。由于这类振荡器需要的起振时间可能比FSCM采样时钟时间长很多，因此可能会检测到假的时钟故障。为避免这种情况，内部振荡器模块会被自动配置为器件时钟并一直工作到主时钟稳定为止（OST 和 PLL 定时器发生超时）。这与双速启动模式相同。一旦主时钟稳定下来，INTRC 就将重新作为 FSCM 时钟源。

注：	用于防止在POR或从休眠状态唤醒时发生错误中断的逻辑，同样也将阻止随后对振荡器故障的检测。通过监视 OSTS 位，并使用定时程序来确定振荡器起振时间是否过长，可避免这个问题。即使如此，在检测到振荡器故障时也不会将振荡器故障中断标志位置 1。

正如**第 25.4.1 节“使用双速启动时的注意事项”**中所述，在等待系统主时钟稳定的过程中，可以选择另一种时钟配置和另一种功耗管理模式。当选择了新的功耗管理模式时，主时钟将被禁止。

25.6 程序校验和代码保护

对于 PIC18F87J90 系列中的所有器件，片上程序存储空间被视为一个统一的存储块。配置位 CP0 控制该存储块的代码保护。该位阻止外部对程序存储空间的读写操作。但对正常的代码执行没有直接影响。

25.6.1 配置寄存器保护

有两种方式保护配置寄存器使其免遭破坏性的改写或读取。主要的保护方式是配置位的一次写入功能，该功能阻止对在上电周期内完成编程的位进行再配置。为防止不可预见的事件造成损害，由于单独的存储单元破坏（如 ESD 事件）产生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 CP0 位置 1 时，因此也将保护器件配置的源数据。

25.7 在线串行编程

PIC18F87J90 系列单片机可以在最终的应用电路中进行串行编程。只需要 5 根线即可实现这一操作，其中时钟线、数据线各一根，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

将 $\overline{\text{DEBUG}}$ 配置位编程为 0，可使能在线调试功能。该功能允许与 MPLAB® IDE 配合使用进行简单的调试。当使能了单片机的这项功能时，有些资源就不再是通用的了。表 25-4 给出了后台调试器所需的资源。

表 25-4： 调试器资源

I/O 引脚：	RB6 和 RB7
堆栈：	2 级深度
程序存储器：	512 字节
数据存储器：	10 字节

注：

26.0 指令集汇总

PIC18F87J90 系列器件具有一个包含 75 条核心 PIC18 指令的标准指令集，和一个包含 8 条新指令的扩展指令集，扩展指令集用于优化递归代码或使用软件堆栈的代码。本章后面的部分将讨论扩展指令集。

26.1 标准指令集

标准的 PIC18 MCU 指令集与以前的 PIC® MCU 指令集相比，添加了很多增强功能，并保持了易于从这些 PIC MCU 指令集移植的特点。大部分指令为单程序存储字指令（16 位），只有 4 条指令需要两个程序存储单元。

每个单字指令都是一个 16 位字，由操作码 （指明指令类型）和一个或多个操作数 （指定指令操作）组成。

整个指令集具有高度的正交性，可以分为以下 4 种基本类型：

- **字节**操作类指令
- **位**操作类指令
- **立即数**操作类指令
- **控制**操作类指令

表 26-2 为 PIC18 指令集汇总，列出了上述四类指令。表 26-1 给出了操作码字段的说明。

大部分**字节**操作类的指令都含有三种操作数：

1. 文件寄存器 （由 “f” 指定）
2. 保存结果的目标寄存器 （由 “d” 指定）
3. 被访问存储区 （由 “a” 指定）

文件寄存器标识符“f”指定了指令将会使用哪一个文件寄存器。目标寄存器标识符 “d” 指定了操作结果的存放位置。如果 “d” 为 0，操作结果存入 WREG 寄存器中。如果 “d” 为 1，操作结果存入指令指定的文件寄存器中。

所有**位**操作类指令都含有三种操作数：

1. 文件寄存器 （由 “f” 指定）
2. 文件寄存器中的位 （由 “b” 指定）
3. 被访问存储区 （由 “a” 指定）

位域标识符 “b” 选择操作所影响的位的编号，而文件寄存器标识符“f”则代表这些位所在的文件寄存器地址。

立即数操作类指令使用以下操作数：

- 要装入到文件寄存器中的立即数 （由 “k” 指定）
- 要装入立即数的 FSR 寄存器 （由 “f” 指定）
- 不需要操作数 （由 “—” 指定）

控制类指令可以使用以下操作数：

- 程序存储器地址 （由 “n” 指定）
- CALL 或 RETURN 指令的模式 （由 “s” 指定）
- 表读和表写指令的模式 （由 “m” 指定）
- 不需要操作数 （由 “—” 指定）

除了 4 条双字指令外，其他所有的指令都是单字指令。双字指令将所需的信息保存在 32 位中。第二个字的高 4 位都是 1。如果第二个字作为一条指令执行，它会执行为 NOP 指令。

除非条件测试结果为真或者指令执行改变了程序计数器的值，否则执行所有的单字指令都只需要一个指令周期。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。

执行双字指令需要两个指令周期。

每个指令周期由 4 个振荡器周期组成。因此，如果振荡器频率为 4 MHz，正常的指令执行时间为 1 µs。如果条件测试结果为真或者指令执行改变了程序计数器的值，则该指令的执行时间为 2 µs。双字跳转指令 （如果为真）的执行则需要 3 µs。

图 26-1 给出了指令的几种通用格式。所有示例均使用 “nnh” 来表示十六进制数。

指令集汇总（见表 26-2）列出了可被 Microchip MPASM™ 汇编器识别的标准指令。

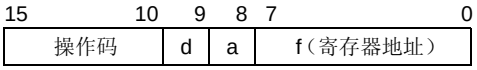
第 26.1.1 节 “ 标准指令集 ” 中对每条指令进行了介绍。

表 26-1： 操作码字段说明

字段	说明
a	快速操作 RAM 位： a = 0：快速操作 RAM 内的 RAM 存储单元 （BSR 寄存器被忽略） a = 1：由 BSR 寄存器指定 RAM 存储区
bbb	8 位文件寄存器内的位地址 （0 到 7）。
BSR	存储区选择寄存器。用于选择当前的 RAM 存储区。
C、 DC、 Z、 OV 和 N	ALU 状态位：进位、半进位、全零、溢出和负标志位。
d	目标寄存器选择位： d = 0：结果保存至 WREG 寄存器 d = 1：结果保存至文件寄存器 f
dest	目标寄存器：可以是 WREG 寄存器或指定的文件寄存器地址。
f	8 位文件寄存器地址 （00h 到 FFh），或 2 位 FSR 标识符 （0h 到 3h）。
f_s	12 位文件寄存器地址 （000h 到 FFFh）。这是源地址。
f_d	12 位文件寄存器地址 （000h 到 FFFh）。这是目标地址。
GIE	全局中断允许位。
k	立即数、常数或者标号 （可能是 8 位、 12 位或 20 位的值）。
label	标号名称。
mm * *+ *- +*	表读和表写指令的 TBLPTR 寄存器模式。 只与表读和表写指令一起使用： 不改变寄存器 （如用于表读和表写的 TBLPTR） 后递增寄存器 （如用于表读和表写的 TBLPTR） 后递减寄存器 （如用于表读和表写的 TBLPTR） 预递增寄存器 （如用于表读和表写的 TBLPTR）
n	相对跳转指令的相对地址 （二进制补码形式），或 Call/ 跳转和 Return 指令的直接地址。
PC	程序计数器。
PCL	程序计数器低字节。
PCH	程序计数器高字节。
PCLATH	程序计数器高字节锁存器。
PCLATU	程序计数器最高字节锁存器。
$\overline{PD}$	掉电位。
PRODH	乘积的高字节。
PRODL	乘积的低字节。
s	快速调用 / 返回模式选择位： s = 0：不对影子寄存器进行更新，也不用影子寄存器的内容更新其他寄存器 s = 1：将寄存器的值装入影子寄存器或把影子寄存器中的值装入寄存器 （快速模式）
TBLPTR	21 位表指针 （指向程序存储器地址）。
TABLAT	8 位表锁存器。
$\overline{TO}$	超时位。
TOS	栈顶。
u	未使用或未改变。
WDT	看门狗定时器。
WREG	工作寄存器 （累加器）。
x	无关位 （0 或 1）。汇编器将生成 x = 0 的代码。为了与所有的 Microchip 软件工具兼容，建议使用这种格式。
z_s	对寄存器 （源）进行间接寻址的 7 位偏移量。
z_d	对寄存器 （目标）进行间接寻址的 7 位偏移量。
{ }	可选参数。
[text]	表示变址地址。
(text)	text 的内容。
[expr]<n>	表示由指针 expr 指向的寄存器中的位 n。
→	赋值。
< >	寄存器位域。
∈	表示属于某个集合。
斜体文字	用户定义项 （字体为 Courier New）。

图 26-1：　　指令的通用格式

针对字节的文件寄存器操作　　**指令示例**



ADDWF MYREG, W, B

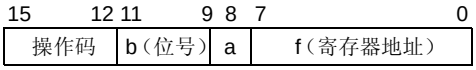
d = 0，表示结果存入 WREG 寄存器
d = 1，表示结果存入文件寄存器（f）
a = 0，强制使用快速操作存储区
a = 1，根据 BSR 选择存储区
f = 8 位文件寄存器地址

字节到字节的传送操作（双字）指令

MOVFF MYREG1, MYREG2

f = 12 位文件寄存器地址

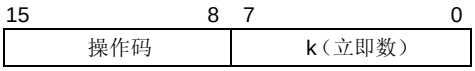
针对位的文件寄存器操作指令



BSF MYREG, bit, B

b = 占 3 位，表示文件寄存器（f）中位的位置
a = 0，强制使用快速操作存储区
a = 1，根据 BSR 选择存储区
f = 8 位文件寄存器地址

立即数操作指令

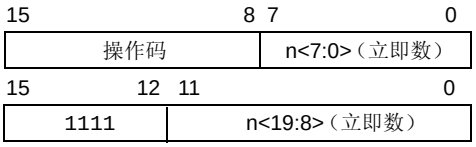


MOVLW 7Fh

k = 8 位立即数的值

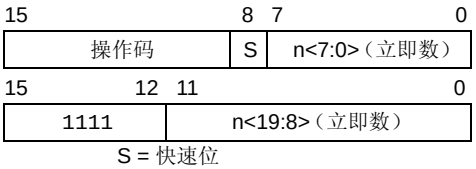
控制操作指令

CALL、GOTO **和跳转**操作类指令



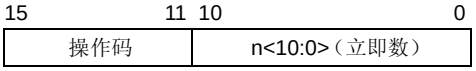
GOTO Label

n = 20 位立即数的值

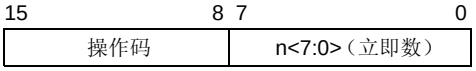


CALL MYFUNC

S = 快速位



BRA MYFUNC



BC MYFUNC

表 26-2： PIC18F87J90 系列指令集

助记符，操作数		说明	周期数	16 位指令字 MSb	LSb	受影响的状态位	注
针对字节的操作类指令							
ADDWF	f, d, a	WREG 与 f 相加	1	0010 01da	ffff ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	WREG 与 f 带进位相加	1	0010 00da	ffff ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	WREG 和 f 作逻辑与运算	1	0001 01da	ffff ffff	Z, N	1,2
CLRF	f, a	将 f 清零	1	0110 101a	ffff ffff	Z	2
COMF	f, d, a	对 f 取反	1	0001 11da	ffff ffff	Z, N	1, 2
CPFSEQ	f, a	将 f 与 WREG 作比较，相等则跳过	1（2或3）	0110 001a	ffff ffff	无	4
CPFSGT	f, a	将 f 与 WREG 作比较，大于则跳过	1（2或3）	0110 010a	ffff ffff	无	4
CPFSLT	f, a	将 f 与 WREG 作比较，小于则跳过	1（2或3）	0110 000a	ffff ffff	无	1, 2
DECF	f, d, a	f 递减 1	1	0000 01da	ffff ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	f 递减 1，为 0 则跳过	1（2或3）	0010 11da	ffff ffff	无	1, 2, 3, 4
DCFSNZ	f, d, a	f 递减 1，非 0 则跳过	1（2或3）	0100 11da	ffff ffff	无	1, 2
INCF	f, d, a	f 递增 1	1	0010 10da	ffff ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	f 递增 1，为 0 则跳过	1（2或3）	0011 11da	ffff ffff	无	4
INFSNZ	f, d, a	f 递增 1，非 0 则跳过	1（2或3）	0100 10da	ffff ffff	无	1, 2
IORWF	f, d, a	WREG 和 f 作逻辑或运算	1	0001 00da	ffff ffff	Z, N	1, 2
MOVF	f, d, a	传送 f	1	0101 00da	ffff ffff	Z, N	1
MOVFF	f_s, f_d	从 f_s（源）送到 第一个字 f_d（目标） 第二个字	2	1100 ffff 1111 ffff	ffff ffff ffff ffff	无	
MOVWF	f, a	将 WREG 内容传送到 f	1	0110 111a	ffff ffff	无	
MULWF	f, a	WREG 与 f 相乘	1	0000 001a	ffff ffff	无	1, 2
NEGF	f, a	对 f 取补	1	0110 110a	ffff ffff	C, DC, Z, OV, N	
RLCF	f, d, a	f 带进位循环左移	1	0011 01da	ffff ffff	C, Z, N	1, 2
RLNCF	f, d, a	f 循环左移 （不带进位）	1	0100 01da	ffff ffff	Z, N	
RRCF	f, d, a	f 带进位循环右移	1	0011 00da	ffff ffff	C, Z, N	
RRNCF	f, d, a	f 循环右移 （不带进位）	1	0100 00da	ffff ffff	Z, N	
SETF	f, a	将 f 的内容置为全 1	1	0110 100a	ffff ffff	无	1, 2
SUBFWB	f, d, a	WREG 减去 f （带借位）	1	0101 01da	ffff ffff	C, DC, Z, OV, N	
SUBWF	f, d, a	f 减去 WREG	1	0101 11da	ffff ffff	C, DC, Z, OV, N	1, 2
SUBWFB	f, d, a	f 减去 WREG （带借位）	1	0101 10da	ffff ffff	C, DC, Z, OV, N	
SWAPF	f, d, a	将 f 中的两个半字节进行交换	1	0011 10da	ffff ffff	无	4
TSTFSZ	f, a	测试 f，为 0 则跳过	1（2或3）	0110 011a	ffff ffff	无	1, 2
XORWF	f, d, a	WREG 和 f 作逻辑异或运算	1	0001 10da	ffff ffff	Z, N	

注 1： 当 PORT 寄存器修改自身时（例如，MOVF PORTB, 1, 0），修改时使用的值是引脚上的当前值。例如，如果将一引脚配置为输入，其对应数据锁存器中的值将为 1，但此时若有外部器件将该引脚驱动为低电平，则被写回数据锁存器的数据值将是 0。

2： 当对 TMR0 寄存器执行该指令 （并且 d = 1）时，如果已为其分配了预分频器，则将该预分频器清零。

3： 如果程序计数器 （PC）被修改或者条件检测为真，则该指令需要两个周期。第二个周期执行一条 NOP 指令。

4： 某些指令是双字指令。除非指令的第一个字获取这 16 位中包含的信息，否则第二个字将作为 NOP 指令执行。这将确保所有程序存储单元内存储的都是合法的指令。

表 26-2： PIC18F87J90 系列指令集 （续）

助记符，操作数		说明	周期数	16 位指令字 MSb LSb	受影响的状态位	注
针对位的操作类指令						
BCF	f, b, a	将 f 中的某位清零	1	1001 bbba ffff ffff	无	1, 2
BSF	f, b, a	将 f 中的某位置 1	1	1000 bbba ffff ffff	无	1, 2
BTFSC	f, b, a	测试 f 中的某位，为 0 则跳过	1（2或3）	1011 bbba ffff ffff	无	3, 4
BTFSS	f, b, a	测试 f 中的某位，为 1 则跳过	1（2或3）	1010 bbba ffff ffff	无	3, 4
BTG	f, b, a	将 f 中的某位取反	1	0111 bbba ffff ffff	无	1, 2
控制类指令						
BC	n	进位则跳转	1 （2）	1110 0010 nnnn nnnn	无	
BN	n	为负则跳转	1 （2）	1110 0110 nnnn nnnn	无	
BNC	n	无进位则跳转	1 （2）	1110 0011 nnnn nnnn	无	
BNN	n	不为负则跳转	1 （2）	1110 0111 nnnn nnnn	无	
BNOV	n	不溢出则跳转	1 （2）	1110 0101 nnnn nnnn	无	
BNZ	n	不为零则跳转	1 （2）	1110 0001 nnnn nnnn	无	
BOV	n	溢出则跳转	1 （2）	1110 0100 nnnn nnnn	无	
BRA	n	无条件跳转	2	1101 0nnn nnnn nnnn	无	
BZ	n	为零则跳转	1 （2）	1110 0000 nnnn nnnn	无	
CALL	n, s	调用子程序 第一个字	2	1110 110s kkkk kkkk	无	
		第二个字		1111 kkkk kkkk kkkk		
CLRWDT	—	将看门狗定时器清零	1	0000 0000 0000 0100	$\overline{TO}$, $\overline{PD}$	
DAW	—	对 WREG 进行十进制调整	1	0000 0000 0000 0111	C	
GOTO	n	跳转到地址 第一个字	2	1110 1111 kkkk kkkk	无	
		第二个字		1111 kkkk kkkk kkkk		
NOP	—	空操作	1	0000 0000 0000 0000	无	
NOP	—	空操作	1	1111 xxxx xxxx xxxx	无	4
POP	—	弹出返回堆栈栈顶（TOS）的内容	1	0000 0000 0000 0110	无	
PUSH	—	将数据压入返回堆栈栈顶（TOS）	1	0000 0000 0000 0101	无	
RCALL	n	相对调用	2	1101 1nnn nnnn nnnn	无	
RESET		软件器件复位	1	0000 0000 1111 1111	全部	
RETFIE	s	中断返回允许	2	0000 0000 0001 000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	返回时将立即数送入 WREG	2	0000 1100 kkkk kkkk	无	
RETURN	s	从子程序返回	2	0000 0000 0001 001s	无	
SLEEP	—	进入待机模式	1	0000 0000 0000 0011	$\overline{TO}$, $\overline{PD}$	

注 1： 当 PORT 寄存器修改自身时（例如，`MOVF PORTB, 1, 0`），修改时使用的值是引脚上的当前值。例如，如果将一引脚配置为输入，其对应数据锁存器中的值将为 1，但此时若有外部器件将该引脚驱动为低电平，则被写回数据锁存器的数据值将是 0。

2： 当对 TMR0 寄存器执行该指令 （并且 d = 1）时，如果已为其分配了预分频器，则将该预分频器清零。

3： 如果程序计数器 （PC）被修改或者条件检测为真，则该指令需要两个周期。第二个周期执行一条 NOP 指令。

4： 某些指令是双字指令。除非指令的第一个字获取这 16 位中包含的信息，否则第二个字将作为 NOP 指令执行。这将确保所有程序存储单元内存储的都是合法的指令。

表 26-2： PIC18F87J90 系列指令集 （续）

助记符，操作数		说明	周期数	16 位指令字 MSb LSb	受影响的状态位	注
立即数操作类指令						
ADDLW	k	WREG 与立即数相加	1	0000 1111 kkkk kkkk	C, DC, Z, OV, N	
ANDLW	k	WREG 和立即数进行逻辑与运算	1	0000 1011 kkkk kkkk	Z, N	
IORLW	k	WREG 和立即数进行逻辑或运算	1	0000 1001 kkkk kkkk	Z, N	
LFSR	f, k	传送立即数 （12 位） 第二个字 到 FSR(f) 第一个字	2	1110 1110 00ff kkkk 1111 0000 kkkk kkkk	无	
MOVLB	k	将立即数送入 BSR<3:0>	1	0000 0001 0000 kkkk	无	
MOVLW	k	将立即数送入 WREG	1	0000 1110 kkkk kkkk	无	
MULLW	k	WREG 和立即数相乘	1	0000 1101 kkkk kkkk	无	
RETLW	k	返回时将立即数送入 WREG	2	0000 1100 kkkk kkkk	无	
SUBLW	k	立即数减去 WREG	1	0000 1000 kkkk kkkk	C, DC, Z, OV, N	
XORLW	k	WREG 和立即数进行逻辑异或运算	1	0000 1010 kkkk kkkk	Z, N	
数据存储器 ↔ 程序存储器操作						
TBLRD*		表读	2	0000 0000 0000 1000	无	
TBLRD*+		后递增表读		0000 0000 0000 1001	无	
TBLRD*-		后递减表读		0000 0000 0000 1010	无	
TBLRD+*		预递增表读		0000 0000 0000 1011	无	
TBLWT*		表写	2	0000 0000 0000 1100	无	
TBLWT*+		后递增表写		0000 0000 0000 1101	无	
TBLWT*-		后递减表写		0000 0000 0000 1110	无	
TBLWT+*		预递增表写		0000 0000 0000 1111	无	

注 1： 当 PORT 寄存器修改自身时 （例如， MOVF PORTB, 1, 0），修改时使用的值是引脚上的当前值。例如，如果将一引脚配置为输入，其对应数据锁存器中的值将为 1，但此时若有外部器件将该引脚驱动为低电平，则被写回数据锁存器的数据值将是 0。

2： 当对 TMR0 寄存器执行该指令 （并且 d = 1）时，如果已为其分配了预分频器，则将该预分频器清零。

3： 如果程序计数器 （PC）被修改或者条件检测为真，则该指令需要两个周期。第二个周期执行一条 NOP 指令。

4： 某些指令是双字指令。除非指令的第一个字获取这 16 位中包含的信息，否则第二个字将作为 NOP 指令执行。这将确保所有程序存储单元内存储的都是合法的指令。

26.1.1 标准指令集

ADDLW **W 与立即数相加**

语法： ADDLW k

操作数： 0 ≤ k ≤ 255

操作： (W) + k → W

受影响的状态位： N、 OV、 C、 DC 和 Z

机器码：

0000	1111	kkkk	kkkk

说明： 将 W 寄存器的内容与 8 位立即数 k 相加，结果存储在 W 寄存器中。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	写入 W

示例： ADDLW 15h

执行指令前
W = 10h
执行指令后
W = 25h

ADDWF **W 与 f 相加**

语法： ADDWF f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (W) + (f) → dest

受影响的状态位： N、 OV、 C、 DC 和 Z

机器码：

0010	01da	ffff	ffff

说明： 将 W 的内容与 f 寄存器的内容相加。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例： ADDWF REG, 0, 0

执行指令前
W = 17h
REG = 0C2h
执行指令后
W = 0D9h
REG = 0C2h

注： 所有的 PIC18 指令都可能在其指令助记符之前使用可选的标号参数，用于符号寻址。如果使用了标号，那么指令格式将变为：{label} 指令参数。

ADDWFC **W 与 f 带进位相加**

语法： ADDWFC f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (W) + (f) + (C) → dest

受影响的状态位： N、 OV、 C、 DC 和 Z

机器码：

0010	00da	ffff	ffff

说明： 将 W 的内容、进位标志位与数据存储单元 f 的内容相加。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存储在数据存储单元 f 中。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例：

```
ADDWFC    REG,  0,  1
```

执行指令前
进位标志位 = 1
REG = 02h
W = 4Dh

执行指令后
进位标志位 = 0
REG = 02h
W = 50h

ANDLW **立即数和 W 寄存器作逻辑与运算**

语法： ANDLW k

操作数： 0 ≤ k ≤ 255

操作： (W) .AND. k → W

受影响的状态位： N 和 Z

机器码：

0000	1011	kkkk	kkkk

说明： 将 W 的内容与 8 位立即数 k 进行逻辑与运算。结果存储在 W 寄存器中。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	写入 W

示例：

```
ANDLW     05Fh
```

执行指令前
W = A3h

执行指令后
W = 03h

ANDWF **将 W 和 f 作逻辑与运算**

语法： ANDWF f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (W) .AND.(f) → dest

受影响的状态位： N 和 Z

机器码：

0001	01da	ffff	ffff

说明： 将 W 的内容与寄存器 f 的内容进行逻辑与运算。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例： ANDWF REG, 0, 0

执行指令前
W = 17h
REG = C2h
执行指令后
W = 02h
REG = C2h

BC **进位则跳转**

语法： BC n

操作数： -128 ≤ n ≤ 127

操作： 如果进位标志位为 1，

(PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1110	0010	nnnn	nnnn

说明： 如果进位标志位为 1，程序将跳转。

“2n”（以二进制补码表示）与 PC 相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。

指令字数： 1

指令周期数： 1（2）

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例： HERE BC 5

执行指令前
PC = 地址（HERE）
执行指令后
如果进位标志位 = 1；
PC = 地址（HERE + 12）
如果进位标志位 = 0；
PC = 地址（HERE + 2）

BCF **将 f 中的某位清零**

语法： BCF f, b {,a}

操作数： 0 ≤ f ≤ 255

0 ≤ b ≤ 7

a ∈ [0,1]

操作： 0 → f<b>

受影响的状态位： 无

机器码：

1001	bbba	ffff	ffff

说明： 将寄存器 f 中的位 b 清零。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写寄存器 f

示例： BCF FLAG_REG, 7, 0

执行指令前
FLAG_REG = C7h
执行指令后
FLAG_REG = 47h

BN **为负则跳转**

语法： BN n

操作数： -128 ≤ n ≤ 127

操作： 如果负标志位为 1，

(PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1110	0110	nnnn	nnnn

说明： 如果负标志位为 1，程序将跳转。

“2n”（以二进制补码表示）与 PC 相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。

指令字数： 1

指令周期数： 1（2）

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例： HERE BN Jump

执行指令前
PC = 地址（HERE）
执行指令后
如果负标志位 = 1；
PC = 地址（Jump）
如果负标志位 = 0；
PC = 地址（HERE + 2）

BNC **无进位则跳转**

语法： BNC n

操作数： -128 ≤ n ≤ 127

操作： 如果进位标志位为 0，
(PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1110	0011	nnnn	nnnn

说明： 如果进位标志位为 0，程序将跳转。
“2n”（以二进制补码表示）与 PC 相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。

指令字数： 1

指令周期数： 1（2）

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例：
```
HERE    BNC  Jump
执行指令前
    PC          = 地址（HERE）
执行指令后
    如果进位标志位 = 0；
        PC      = 地址（Jump）
    如果进位标志位 = 1；
        PC      = 地址（HERE + 2）
```

BNN **不为负则跳转**

语法： BNN n

操作数： -128 ≤ n ≤ 127

操作： 如果负标志位为 0，
(PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1110	0111	nnnn	nnnn

说明： 如果负标志位为 0，程序将跳转。
“2n”（以二进制补码表示）与 PC 相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。

指令字数： 1

指令周期数： 1（2）

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例：
```
HERE    BNN  Jump
执行指令前
    PC          =  地址（HERE）
执行指令后
    如果负标志位 =  0；
        PC      =  地址（Jump）
    如果负标志位 =  1；
        PC      =  地址（HERE + 2）
```

BNOV　　不溢出则跳转

语法：　BNOV　n

操作数：　-128 ≤ n ≤ 127

操作：　如果溢出标志位为 0，
(PC) + 2 + 2n → PC

受影响的状态位：　无

机器码：

1110	0101	nnnn	nnnn

说明：　如果溢出标志位为 0，程序将跳转。
“2n”（以二进制补码表示）与 PC 相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。

指令字数：　1

指令周期数：　1（2）

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例：　HERE　BNOV　Jump

执行指令前
PC = 地址（HERE）
执行指令后
如果溢出标志位 = 0；
PC = 地址（Jump）
如果溢出标志位 = 1；
PC = 地址（HERE + 2）

BNZ　　不为零则跳转

语法：　BNZ　n

操作数：　-128 ≤ n ≤ 127

操作：　如果全零标志位为 0，
(PC) + 2 + 2n → PC

受影响的状态位：　无

机器码：

1110	0001	nnnn	nnnn

说明：　如果全零标志位为 0，程序将跳转。
“2n”（以二进制补码表示）与 PC 相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。

指令字数：　1

指令周期数：　1（2）

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例：　HERE　BNZ　Jump

执行指令前
PC = 地址（HERE）
执行指令后
如果全零标志位 = 0；
PC = 地址（Jump）
如果全零标志位 = 1；
PC = 地址（HERE + 2）

BRA **无条件跳转**

语法： BRA n

操作数： -1024 ≤ n ≤ 1023

操作： (PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1101	0nnn	nnnn	nnnn

说明： “2n”（以二进制补码表示）与 PC 相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

示例： HERE BRA Jump

执行指令前
PC = 地址 （HERE）
执行指令后
PC = 地址 （Jump）

BSF **将 f 中的某位置 1**

语法： BSF f, b {,a}

操作数： 0 ≤ f ≤ 255

0 ≤ b ≤ 7

a ∈ [0,1]

操作： 1 → f<b>

受影响的状态位： 无

机器码：

1000	bbba	ffff	ffff

说明： 将寄存器 f 的位 b 置 1。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写寄存器 f

示例： BSF FLAG_REG, 7, 1

执行指令前
FLAG_REG = 0Ah
执行指令后
FLAG_REG = 8Ah

BTFSC **测试文件寄存器中的某位，为 0 则跳过**

语法： BTFSC f, b {,a}

操作数： 0 ≤ f ≤ 255

0 ≤ b ≤ 7

a ∈ [0,1]

操作： 如果 (f<b>) = 0，则跳过

受影响的状态位： 无

机器码：

1011	bbba	ffff	ffff

说明： 如果寄存器 f 的位 b 为 0，则跳过下一条指令。即在 b 位为 0 时，丢弃在执行当前指令期间取的下一条指令转而执行一条 NOP 指令，使该指令变成双周期指令。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f≤95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1（2）

注： 如果跳过且后面跟有 2 字指令，则执行 BTFSC 需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	空操作

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过且后面跟有 2 字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    BTFSC   FLAG, 1, 0
FALSE   :
TRUE    :
```

执行指令前
PC = 地址（HERE）
执行指令后
如果 FLAG<1> = 0；
PC = 地址（TRUE）
如果 FLAG<1> = 1；
PC = 地址（FALSE）

BTFSS **测试文件寄存器中的某位，为 1 则跳过**

语法： BTFSS f, b {,a}

操作数： 0 ≤ f ≤ 255

0 ≤ b < 7

a ∈ [0,1]

操作： 如果 (f<b>) = 1，则跳过

受影响的状态位： 无

机器码：

1010	bbba	ffff	ffff

说明： 如果寄存器 f 的位 b 为 1，则跳过下一条指令。即在 b 位为 1 时，丢弃在执行当前指令期间取的下一条指令转而执行一条 NOP 指令，使该指令变成双周期指令。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f≤95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1（2）

注： 如果跳过且后面跟有 2 字指令，则执行 BTFSS 需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	空操作

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过且后面跟有 2 字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    BTFSS   FLAG, 1, 0
FALSE   :
TRUE    :
```

执行指令前
PC = 地址（HERE）
执行指令后
如果 FLAG<1> = 0；
PC = 地址（FALSE）
如果 FLAG<1> = 1；
PC = 地址（TRUE）

BTG **将 f 中的某位取反**

语法： BTG f, b {,a}

操作数： 0 ≤ f ≤ 255

0 ≤ b < 7

a ∈ [0,1]

操作： $(\overline{f<b>}) \rightarrow f<b>$

受影响的状态位： 无

机器码：

0111	bbba	ffff	ffff

说明： 将数据存储单元 f 中的位 b 取反。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写寄存器 f

示例：
```
BTG     PORTC,  4, 0
```
执行指令前：
PORTC = 0111 0101 [75h]
执行指令后：
PORTC = 0110 0101 [65h]

BOV **溢出则跳转**

语法： BOV n

操作数： -128 ≤ n ≤ 127

操作： 如果溢出标志位为 1，

(PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1110	0100	nnnn	nnnn

说明： 如果溢出标志位为 1，程序将跳转。

“2n”（以二进制补码表示）与 PC 相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。

指令字数： 1

指令周期数： 1（2）

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例：
```
HERE    BOV   Jump
```
执行指令前
PC = 地址（HERE）
执行指令后
如果溢出标志位 = 1；
PC = 地址（Jump）
如果溢出标志位 = 0；
PC = 地址（HERE + 2）

BZ **为零则跳转**

语法： BZ n

操作数： -128 ≤ n ≤ 127

操作： 如果全零标志位为 1，
(PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1110	0000	nnnn	nnnn

说明： 如果全零标志位为 1，程序将跳转。
“2n”（以二进制补码表示）与 PC 相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。

指令字数： 1

指令周期数： 1（2）

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	空操作

示例： HERE BZ Jump

执行指令前
PC = 地址（HERE）
执行指令后
如果全零标志位 = 1；
PC = 地址（Jump）
如果全零标志位 = 0；
PC = 地址（HERE + 2）

CALL **调用子程序**

语法： CALL k {,s}

操作数： 0 ≤ k ≤ 1048575
s ∈ [0,1]

操作： (PC) + 4 → TOS，
k → PC<20:1>；
如果 s = 1
(W) → WS，
(STATUS) → STATUSS，
(BSR) → BSRS

受影响的状态位： 无

机器码：

第一个字（k<7:0>）	1110 1111	110s k_{19}kkk	$k_7$kkk kkkk	kkkk$_0$ kkkk$_8$
第二个字（k<19:8>）				

说明： 可在整个 2 MB 的存储器范围内进行子程序调用。首先，将返回地址（PC + 4）压入返回堆栈。如果 s = 1，还会将 W、STATUS 和 BSR 寄存器的内容存入它们各自的影子寄存器 WS、STATUSS 和 BSRS。如果 s = 0，将不会进行任何更新。然后，将20位的值k装入PC<20:1>。CALL 是一条双周期指令。

指令字数： 2

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k<7:0>	将 PC 压入堆栈	读立即数 k<19:8>，写入 PC
空操作	空操作	空操作	空操作

示例： HERE CALL THERE,1

执行指令前
PC = 地址（HERE）
执行指令后
PC = 地址（THERE）
TOS = 地址（HERE + 4）
WS = W
BSRS = BSR
STATUSS = STATUS

CLRF **将 f 清零**

语法： CLRF f {,a}

操作数： 0 ≤ f ≤ 255

a ∈ [0,1]

操作： 000h → f，

1 → Z

受影响的状态位： Z

机器码：

0110	101a	ffff	ffff

说明： 清零指定寄存器的内容。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写寄存器 f

示例： CLRF FLAG_REG,1

执行指令前
FLAG_REG = 5Ah
执行指令后
FLAG_REG = 00h

CLRWDT **将看门狗定时器清零**

语法： CLRWDT

操作数： 无

操作： 000h → WDT，

000h → WDT 后分频器，

1 → $\overline{TO}$，

1 → $\overline{PD}$

受影响的状态位： $\overline{TO}$ 和 $\overline{PD}$

机器码：

0000	0000	0000	0100

说明： CLRWDT指令复位看门狗定时器及其后分频器。状态位 $\overline{TO}$ 和 $\overline{PD}$ 置 1。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	处理数据	空操作

示例： CLRWDT

执行指令前
WDT 计数器 = ?
执行指令后
WDT 计数器 = 00h
WDT 后分频器 = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

COMF **对 f 取反**

语法： COMF f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： $\bar{f} \rightarrow$ dest

受影响的状态位： N 和 Z

机器码：

0001	11da	ffff	ffff

说明： 将寄存器 f 的内容取反。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例：
```
COMF    REG, 0, 0
```
执行指令前
REG = 13h
执行指令后
REG = 13h
W = ECh

CPFSEQ **比较 f 和 W，如果 f = W 则跳过**

语法： CPFSEQ f {,a}

操作数： 0 ≤ f ≤ 255

a ∈ [0,1]

操作： (f) – (W)，
如果 (f) = (W)，则跳过 （无符号比较）

受影响的状态位： 无

机器码：

0110	001a	ffff	ffff

说明： 通过执行无符号的减法，将数据存储单元 f 的内容与 W 的内容作比较。

如果 f = W，则所取的指令被丢弃并执行一条 NOP 指令，使该指令成为双周期指令。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1（2）

注： 如果跳过且后面跟有 2 字指令，则执行 CPFSEQ 需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	空操作

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过且后面跟有 2 字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：
```
HERE    CPFSEQ REG, 0
NEQUAL  :
EQUAL   :
```
执行指令前
PC 地址 = HERE
W = ?
REG = ?
执行指令后
如果 REG = W；
PC = 地址 （EQUAL）
如果 REG ≠ W；
PC = 地址 （NEQUAL）

CPFSGT **比较 f 和 W，如果 f > W 则跳过**

语法： CPFSGT f {,a}

操作数： 0 ≤ f ≤ 255

a ∈ [0,1]

操作： (f) – (W)，
如果 (f) > (W)，则跳过 （无符号比较）

受影响的状态位： 无

机器码：

0110	010a	ffff	ffff

说明： 通过执行无符号的减法，将数据存储单元 f 的内容与 W 的内容作比较。

如果 f > W，则所取的指令被丢弃并执行一条 NOP 指令，使该指令成为双周期指令。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1 （2）

注： 如果跳过且后面跟有 2 字指令，则执行 CPFSGT 需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	空操作

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过且后面跟有 2 字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE      CPFSGT REG, 0
NGREATER  :
GREATER   :
```

执行指令前
PC = 地址 （HERE）
W = ?
执行指令后
如果 REG > W；
PC = 地址 （GREATER）
如果 REG ≤ W；
PC = 地址 （NGREATER）

CPFSLT **比较 f 和 W，如果 f < W 则跳过**

语法： CPFSLT f {,a}

操作数： 0 ≤ f ≤ 255

a ∈ [0,1]

操作： (f) – (W)，
如果 (f) < (W)，则跳过 （无符号比较）

受影响的状态位： 无

机器码：

0110	000a	ffff	ffff

说明： 通过执行无符号的减法，将数据存储单元 f 的内容与 W 的内容作比较。

如果 f < W，则所取的指令被丢弃并执行一条 NOP 指令，使该指令成为双周期指令。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

指令字数： 1

指令周期数： 1 （2）

注： 如果跳过且后面跟有 2 字指令，则执行 CPFSLT 需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	空操作

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过且后面跟有 2 字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    CPFSLT REG, 1
NLESS   :
LESS    :
```

执行指令前
PC = 地址 （HERE）
W = ?
执行指令后
如果 REG < W；
PC = 地址 （LESS）
如果 REG ≥ W；
PC = 地址 （NLESS）

DAW **对 W 寄存器进行十进制调整**

语法： DAW

操作数： 无

操作： 如果 [W<3:0> > 9] 或 [DC = 1]，则
(W<3:0>) + 6 → W<3:0> ；
否则
(W<3:0>) → W<3:0>

如果 [W<7:4> > 9] 或 [C = 1]，则
(W<7:4>) + 6 → W<7:4> ；
C = 1 ；
否则
(W<7:4>) → W<7:4>

受影响的状态位： C

机器码：

0000	0000	0000	0111

说明： DAW 指令调整 W 寄存器内的 8 位值，即之前两个压缩 BCD 格式的变量之和，并产生一个正确的压缩 BCD 格式结果。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 W	处理数据	写 W

例 1：

```
DAW
```

执行指令前
W = A5h
C = 0
DC = 0
执行指令后
W = 05h
C = 1
DC = 0

例 2：

执行指令前
W = CEh
C = 0
DC = 0
执行指令后
W = 34h
C = 1
DC = 0

DECF **f 递减 1**

语法： DECF f {,d {,a}}

操作数： 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (f) – 1 → dest

受影响的状态位： C、 DC、 N、 OV 和 Z

机器码：

0000	01da	ffff	ffff

说明： 将寄存器 f 的内容递减 1。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例：

```
DECF    CNT,   1, 0
```

执行指令前
CNT = 01h
Z = 0
执行指令后
CNT = 00h
Z = 1

DECFSZ **f 递减 1，为 0 则跳过**

语法： DECFSZ f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (f) – 1 → dest，
如果结果 = 0 则跳过

受影响的状态位： 无

机器码：

0010	11da	ffff	ffff

说明： 将寄存器 f 的内容递减 1。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果结果为0，则丢弃已取的下一条指令转而执行一条 NOP 指令，使该指令成为双周期指令。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1（2）

注： 如果跳过且后面跟有 2 字指令，则执行 DECFSZ 需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过且后面跟有 2 字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    DECFSZ  CNT, 1, 1
        GOTO    LOOP
CONTINUE
```

执行指令前
PC = 地址（HERE）
执行指令后
CNT = CNT – 1
如果 CNT = 0；
PC = 地址（CONTINUE）
如果 CNT ≠ 0；
PC = 地址（HERE + 2）

DCFSNZ **f 递减 1，非 0 则跳过**

语法： DCFSNZ f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (f) – 1 → dest，
如果结果 ≠ 0 则跳过

受影响的状态位： 无

机器码：

0100	11da	ffff	ffff

说明： 将寄存器 f 的内容递减 1。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果结果不为0，则丢弃已取的下一条指令转而执行一条 NOP 指令，使该指令成为双周期指令。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1（2）

注： 如果跳过且后面跟有 2 字指令，则执行 DCFSNZ 需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过且后面跟有 2 字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    DCFSNZ  TEMP, 1, 0
ZERO    :
NZERO   :
```

执行指令前
TEMP = ?
执行指令后
TEMP = TEMP – 1，
如果 TEMP = 0；
PC = 地址（ZERO）
如果 TEMP ≠ 0；
PC = 地址（NZERO）

GOTO **无条件跳转**

语法： GOTO k

操作数： 0 ≤ k ≤ 1048575

操作： k → PC<20:1>

受影响的状态位： 无

机器码：

第一个字 （k<7:0>）	1110	1111	$k_7$kkk	kkkk$_0$
第二个字 （k<19:8>）	1111	k_{19}kkk	kkkk	kkkk$_8$

说明： GOTO指令允许无条件跳转到整个2 MB存储器范围中的任何位置。将 20 位值 k 装入 PC<20:1>。GOTO 始终为双周期指令。

指令字数： 2

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k<7:0>	空操作	读立即数 k<19:8>，写入 PC
空操作	空操作	空操作	空操作

示例： GOTO THERE

执行指令后

PC = 地址 （THERE）

INCF **f 递增 1**

语法： INCF f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (f) + 1 → dest

受影响的状态位： C、 DC、 N、 OV 和 Z

机器码：

0010	10da	ffff	ffff

说明： 将寄存器 f 的内容递增 1。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例： INCF CNT, 1, 0

执行指令前

CNT = FFh

Z = 0

C = ?

DC = ?

执行指令后

CNT = 00h

Z = 1

C = 1

DC = 1

INCFSZ **f 递增 1，为 0 则跳过**

语法： INCFSZ f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (f) + 1 → dest，

如果结果 = 0 则跳过

受影响的状态位： 无

机器码：

0011	11da	ffff	ffff

说明： 将寄存器 f 的内容递增 1。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果结果为 0，则丢弃已取的下一条指令转而执行一条 NOP 指令，使该指令成为双周期指令。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节“ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1（2）

注： 如果跳过且后面跟有 2 字指令，则执行 INCFSZ 需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过且后面跟有 2 字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    INCFSZ   CNT, 1, 0
NZERO   :
ZERO    :
```

执行指令前
PC = 地址（HERE）
执行指令后
CNT = CNT + 1
如果 CNT = 0；
PC = 地址（ZERO）
如果 CNT ≠ 0；
PC = 地址（NZERO）

INFSNZ **f 递增 1，非 0 则跳过**

语法： INFSNZ f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (f) + 1 → dest，
如果结果 ≠ 0 则跳过

受影响的状态位： 无

机器码：

0100	10da	ffff	ffff

说明： 将寄存器 f 的内容递增 1。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果结果不为 0，则丢弃已取的下一条指令转而执行一条 NOP 指令，使该指令成为双周期指令。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节“ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1（2）

注： 如果跳过且后面跟有 2 字指令，则执行 INFSNZ 需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过且后面跟有 2 字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    INFSNZ  REG, 1, 0
ZERO
NZERO
```

执行指令前
PC = 地址（HERE）
执行指令后
REG = REG + 1
如果 REG ≠ 0；
PC = 地址（NZERO）
如果 REG = 0；
PC = 地址（ZERO）

IORLW **将立即数与 W 作逻辑或运算**

语法： IORLW k

操作数： 0 ≤ k ≤ 255

操作： (W) .OR. k → W

受影响的状态位： N 和 Z

机器码：

0000	1001	kkkk	kkkk

说明： 将 W 的内容与 8 位立即数 k 进行逻辑或运算。结果存储在 W 寄存器中。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	写入 W

示例： IORLW 35h

执行指令前
W = 9Ah
执行指令后
W = BFh

IORWF **将 W 与 f 作逻辑或运算**

语法： IORWF f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (W) .OR.(f) → dest

受影响的状态位： N 和 Z

机器码：

0001	00da	ffff	ffff

说明： 将 W 的内容与寄存器 f 的内容进行逻辑或运算。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例： IORWF RESULT, 0, 1

执行指令前
RESULT = 13h
W = 91h
执行指令后
RESULT = 13h
W = 93h

LFSR **装入 FSR**

语法： LFSR f, k

操作数： 0 ≤ f ≤ 2

0 ≤ k ≤ 4095

操作： k → FSRf

受影响的状态位： 无

机器码：

1110 1111	1110 0000	00ff $k_7$kkk	k_{11}kkk kkkk

说明： 将 12 位立即数 k 装入 f 所指向的文件选择寄存器。

指令字数： 2

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k 的 MSB	处理数据	将立即数 k 的 MSB 写入 FSRfH
译码	读立即数 k 的 LSB	处理数据	将立即数 k 的 LSB 写入 FSRfL

示例： LFSR 2, 3ABh

执行指令后
FSR2H = 03h
FSR2L = ABh

MOVF **传送 f**

语法： MOVF f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： f → dest

受影响的状态位： N 和 Z

机器码：

0101	00da	ffff	ffff

说明： 根据 d 的状态，将寄存器 f 的内容送入目标寄存器。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。f 可以为 256 字节存储区中的任何地址单元。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写 W

示例： MOVF REG, 0, 0

执行指令前
REG = 22h
W = FFh
执行指令后
REG = 22h
W = 22h

MOVFF **将源寄存器的内容送入目标寄存器**

语法： MOVFF f_s,f_d

操作数： $0 \le f_s \le 4095$

$0 \le f_d \le 4095$

操作： $(f_s) \rightarrow f_d$

受影响的状态位： 无

机器码：

第一个字（源）	1100	ffff	ffff	$ffff_s$
第二个字（目标）	1111	ffff	ffff	$ffff_d$

说明： 将源寄存器 f_s 的内容送入目标寄存器 f_d。源寄存器 f_s 可以是 4096 字节数据空间（000h 到 FFFh）中的任何单元，目标寄存器 f_d 也可以是 000h 到 FFFh 中的任何单元。

源或目标寄存器都可以是 W（这是个有用的特例）。

MOVFF 指令对于将数据存储单元中的内容送入外设寄存器（如发送缓冲区或 I/O 端口）的场合非常有用。

MOVFF 指令不能使用 PCL、TOSU、TOSH 或 TOSL 作为目标寄存器。

指令字数： 2

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f（源寄存器）	处理数据	空操作
译码	空操作 无假读	空操作	写寄存器 f（目标寄存器）

示例：
```
MOVFF   REG1, REG2
```
执行指令前
REG1 = 33h
REG2 = 11h
执行指令后
REG1 = 33h
REG2 = 33h

MOVLB **将立即数送入 BSR 的低半字节**

语法： MOVLW k

操作数： $0 \le k \le 255$

操作： $k \rightarrow BSR$

受影响的状态位： 无

机器码：

0000	0001	kkkk	kkkk

说明： 将 8 位立即数 k 装入存储区选择寄存器（BSR）。不管 k_7:k_4 的值如何，BSR<7:4> 的值将始终保持为 0。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	将立即数 k 写入 BSR

示例：
```
MOVLB   5
```
执行指令前
BSR 寄存器 = 02h
执行指令后
BSR 寄存器 = 05h

MOVLW **将立即数送入 W**

语法： MOVLW k

操作数： 0 ≤ k ≤ 255

操作： k → W

受影响的状态位： 无

机器码：

0000	1110	kkkk	kkkk

说明： 将 8 位立即数 k 装入 W。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	写入 W

示例： MOVLW 5Ah

执行指令后
W = 5Ah

MOVWF **将 W 的内容送入 f**

语法： MOVWF f {,a}

操作数： 0 ≤ f ≤ 255

a ∈ [0,1]

操作： (W) → f

受影响的状态位： 无

机器码：

0110	111a	ffff	ffff

说明： 将 W 寄存器中的数据送入寄存器 f。f 可以是256字节存储区中的任何地址单元。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写寄存器 f

示例： MOVWF REG, 0

执行指令前
W = 4Fh
REG = FFh
执行指令后
W = 4Fh
REG = 4Fh

MULLW **将立即数与 W 中的内容相乘**

语法： MULLW k

操作数： 0 ≤ k ≤ 255

操作： (W) x k → PRODH:PRODL

受影响的状态位： 无

机器码：

0000	1101	kkkk	kkkk

说明： 将 W 的内容与 8 位立即数 k 进行无符号的乘法运算。16 位的结果存储在 PRODH:PRODL 寄存器对中，其中 PRODH 用于存储高字节。

W 的内容不改变。

所有状态标志位都不受影响。

请注意此操作不可能发生溢出或进位。结果有可能为全零，但不会被检测到。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	写寄存器 PRODH: PRODL

示例： MULLW 0C4h

执行指令前
W = E2h
PRODH = ?
PRODL = ?

执行指令后
W = E2h
PRODH = ADh
PRODL = 08h

MULWF **将 W 与 f 的内容相乘**

语法： MULWF f {,a}

操作数： 0 ≤ f ≤ 255

a ∈ [0,1]

操作： (W) x (f) → PRODH:PRODL

受影响的状态位： 无

机器码：

0000	001a	ffff	ffff

说明： 将 W 的内容与寄存器单元 f 的内容执行无符号的乘法运算。运算的 16 位结果保存在 PRODH:PRODL 寄存器对中，其中 PRODH 用于存储高字节。 W 和 f 的内容都不改变。

所有状态标志位都不受影响。

请注意此操作不可能发生溢出或进位。结果有可能为全零，但不会被检测到。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f≤95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写寄存器 PRODH: PRODL

示例： MULWF REG, 1

执行指令前
W = C4h
REG = B5h
PRODH = ?
PRODL = ?

执行指令后
W = C4h
REG = B5h
PRODH = 8Ah
PRODL = 94h

NEGF **对 f 取补**

语法： NEGF f {,a}

操作数： $0 \le f \le 255$

$a \in [0,1]$

操作： $(\bar{f}) + 1 \rightarrow f$

受影响的状态位： N、 OV、 C、 DC 和 Z

机器码：

0110	110a	ffff	ffff

说明： 用二进制补码对存储单元 f 取补，结果存储在数据存储单元 f 中。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写寄存器 f

示例：

```
NEGF    REG, 1
```

执行指令前
REG = 0011 1010 [3Ah]
执行指令后
REG = 1100 0110 [C6h]

NOP **空操作**

语法： NOP

操作数： 无

操作： 空操作

受影响的状态位： 无

机器码：

0000 1111	0000 xxxx	0000 xxxx	0000 xxxx

说明： 不执行任何操作。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	空操作	空操作

示例：

无。

POP **弹出返回堆栈栈顶的内容**

语法： POP

操作数： 无

操作： (TOS) → 位桶 （丢弃）

受影响的状态位： 无

机器码：

0000	0000	0000	0110

说明： 从返回堆栈弹出 TOS 值并丢弃。然后，前一个压入返回堆栈的值成为 TOS 值。

此指令可以让用户正确管理返回堆栈，从而实现软件堆栈。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	弹出 TOS 值	空操作

示例：

```
POP
GOTO    NEW
```

执行指令前
TOS = 0031A2h
堆栈 （下一级） = 014332h

执行指令后
TOS = 014332h
PC = NEW

PUSH **将数据压入返回堆栈栈顶**

语法： PUSH

操作数： 无

操作： (PC + 2) → TOS

受影响的状态位： 无

机器码：

0000	0000	0000	0101

说明： PC + 2 的值被压入返回堆栈的栈顶。原先的 TOS 值被压入堆栈的下一级。

此指令允许通过修改 TOS 并将其压入返回堆栈来实现软件堆栈。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	将 PC + 2 压入返回堆栈	空操作	空操作

示例：

```
PUSH
```

执行指令前
TOS = 345Ah
PC = 0124h

执行指令后
PC = 0126h
TOS = 0126h
堆栈 （下一级） = 345Ah

RCALL **相对调用**

语法： RCALL n

操作数： -1024 ≤ n ≤ 1023

操作： (PC) + 2 → TOS，
(PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1101	1nnn	nnnn	nnnn

说明： 从当前地址跳转（最多 1K）来调用子程序。首先，将返回地址（PC + 2）压入堆栈。然后，将 “2n”（以二进制补码表示）与 PC 相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 n 将 PC 压入堆栈	处理数据	写入 PC
空操作	空操作	空操作	空操作

示例： HERE RCALL Jump

执行指令前
PC = 地址（HERE）
执行指令后
PC = 地址（Jump）
TOS = 地址（HERE + 2）

RESET **复位**

语法： RESET

操作数： 无

操作： 将所有受 $\overline{\text{MCLR}}$ 复位影响的寄存器和标志位复位。

受影响的状态位： 全部

机器码：

0000	0000	1111	1111

说明： 此指令可实现用软件执行 $\overline{\text{MCLR}}$ 复位。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	开始复位	空操作	空操作

示例： RESET

执行指令后
寄存器 = 复位值
标志位 * = 复位值

RETFIE **从中断返回**

语法： RETFIE {s}

操作数： s ∈ [0,1]

操作： (TOS) → PC，
1 → GIE/GIEH 或 PEIE/GIEL；
如果 s = 1，
(WS) → W，
(STATUSS) → STATUS，
(BSRS) → BSR，
PCLATU 和 PCLATH 保持不变

受影响的状态位： GIE/GIEH 和 PEIE/GIEL

机器码：

0000	0000	0001	000s

说明： 从中断返回。执行出栈操作，将栈顶（TOS）的内容装入 PC。通过将高或低优先级全局中断允许位置 1，来允许中断。如果 s = 1，则影子寄存器 WS、STATUSS 和 BSRS 的内容将被装入对应的寄存器 W、STATUS 和 BSR。如果 s = 0，则不更新这些寄存器。

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	空操作	从堆栈弹出 PC 值 将 GIEH 或 GIEL 置 1
空操作	空操作	空操作	空操作

示例： RETFIE 1

中断后
PC = TOS
W = WS
BSR = BSRS
STATUS = STATUSS
GIE/GIEH, PEIE/GIEL = 1

RETLW **将立即数返回到 W**

语法： RETLW k

操作数： 0 ≤ k ≤ 255

操作： k → W，
(TOS) → PC，
PCLATU 和 PCLATH 保持不变

受影响的状态位： 无

机器码：

0000	1100	kkkk	kkkk

说明： 将 8 位立即数 k 装入 W。将栈顶内容（返回地址）装入程序计数器。高字节地址锁存器（PCLATH）内容保持不变。

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	从堆栈弹出 PC 值，写入 W
空操作	空操作	空操作	空操作

示例：

```
    CALL TABLE ; W contains table
               ; offset value
               ; W now has
               ; table value
   :
TABLE
    ADDWF PCL  ; W = offset
    RETLW k0   ; Begin table
    RETLW k1   ;
    :
    :
    RETLW kn   ; End of table
```

执行指令前
W = 07h
执行指令后
W = kn 的值

RETURN **从子程序返回**

语法： RETURN {s}

操作数： s ∈ [0,1]

操作： (TOS) → PC；
如果 s = 1，
(WS) → W，
(STATUSS) → STATUS，
(BSRS) → BSR，
PCLATU 和 PCLATH 保持不变

受影响的状态位： 无

机器码：

0000	0000	0001	001s

说明： 从子程序返回。执行出栈操作，将栈顶（TOS）的内容装入程序计数器。如果 s = 1，则影子寄存器 WS、STATUSS 和 BSRS 的内容将被装入对应的寄存器 W、STATUS 和 BSR。如果 s = 0，则不更新这些寄存器。

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	处理数据	从堆栈弹出 PC 值
空操作	空操作	空操作	空操作

示例： RETURN

执行指令后：
PC = TOS

RLCF **f 带进位循环左移**

语法： RLCF f {,d {,a}}

操作数： 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (f<n>) → dest<n + 1>，
(f<7>) → C，
(C) → dest<0>

受影响的状态位： C、N 和 Z

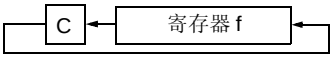
机器码：

0011	01da	ffff	ffff

说明： 将寄存器f的内容连同进位标志位一起循环左移 1 位。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。



指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例：
```
RLCF    REG, 0, 0
```
执行指令前
```
REG  =  1110 0110
C    =  0
```
执行指令后
```
REG  =  1110 0110
W    =  1100 1100
C    =  1
```

RLNCF **f 循环左移 （不带进位）**

语法： RLNCF f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (f<n>) → dest<n + 1>，

(f<7>) → dest<0>

受影响的状态位： N 和 Z

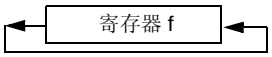
机器码：

0100	01da	ffff	ffff

说明： 将寄存器 f 的内容循环左移 1 位。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “立即数变址寻址模式中针对字节和针对位的指令”**。



指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例： RLNCF REG, 1, 0

执行指令前
REG = 1010 1011
执行指令后
REG = 0101 0111

RRCF **f 带进位循环右移**

语法： RRCF f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (f<n>) → dest<n – 1>，

(f<0>) → C，

(C) → dest<7>

受影响的状态位： C、 N 和 Z

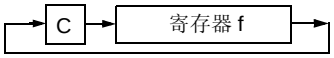
机器码：

0011	00da	ffff	ffff

说明： 将寄存器 f 的内容连同进位标志位一起循环右移 1 位。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令”**。



指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例： RRCF REG, 0, 0

执行指令前
REG = 1110 0110
C = 0
执行指令后
REG = 1110 0110
W = 0111 0011
C = 0

RRNCF **f 循环右移 （不带进位）**

语法： RRNCF f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (f<n>) → dest<n – 1>,

(f<0>) → dest<7>

受影响的状态位： N 和 Z

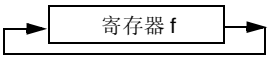
机器码：

0100	00da	ffff	ffff

说明： 将寄存器 f 的内容循环右移 1 位。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区，忽略 BSR 的值。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。



指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

例 1：

```
RRNCF   REG, 1, 0
执行指令前
    REG    =   1101 0111
执行指令后
    REG    =   1110 1011
```

例 2：

```
RRNCF   REG, 0, 0
执行指令前
    W      =   ?
    REG    =   1101 0111
执行指令后
    W      =   1110 1011
    REG    =   1101 0111
```

SETF **将 f 的内容置为全 1**

语法： SETF f {,a}

操作数： 0 ≤ f ≤ 255

a ∈ [0,1]

操作： FFh → f

受影响的状态位： 无

机器码：

0110	100a	ffff	ffff

说明： 将指定寄存器的内容置为 FFh。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写寄存器 f

示例：

```
SETF    REG,1
执行指令前
    REG    =   5Ah
执行指令后
    REG    =   FFh
```

SLEEP **进入休眠模式**

语法： SLEEP

操作数： 无

操作： 00h → WDT，

0 → WDT 后分频器，

1 → $\overline{TO}$，

0 → $\overline{PD}$

受影响的状态位： $\overline{TO}$ 和 $\overline{PD}$

机器码：

0000	0000	0000	0011

说明： 掉电状态位（$\overline{PD}$）清零。超时状态位（$\overline{TO}$）置 1。看门狗定时器及其后分频器清零。

振荡器停振，处理器进入休眠模式。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	处理数据	进入休眠模式

示例： SLEEP

执行指令前

$\overline{TO}$ = ?

$\overline{PD}$ = ?

执行指令后

$\overline{TO}$ = 1†

$\overline{PD}$ = 0

† 如果由 WDT 引起唤醒，则该位将被清零。

SUBFWB **W 减去 f （带借位）**

语法： SUBFWB f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (W) – (f) – ($\overline{C}$) → dest

受影响的状态位： N、 OV、 C、 DC 和 Z

机器码：

0101	01da	ffff	ffff

说明： 将 W 的内容减去 f 寄存器的内容和进位标志位（借位）（通过二进制补码方式进行运算）。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f≤95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

例 1： SUBFWB REG, 1, 0

执行指令前

REG = 3

W = 2

C = 1

执行指令后

REG = FF

W = 2

C = 0

Z = 0

N = 1 ；结果为负

例 2： SUBFWB REG, 0, 0

执行指令前

REG = 2

W = 5

C = 1

执行指令后

REG = 2

W = 3

C = 1

Z = 0

N = 0 ；结果为正

例 3： SUBFWB REG, 1, 0

执行指令前

REG = 1

W = 2

C = 0

执行指令后

REG = 0

W = 2

C = 1

Z = 1 ；结果为全零

N = 0

SUBLW **立即数减去 W 的内容**

语法： SUBLW k

操作数： 0 ≤ k ≤ 255

操作： k – (W) → W

受影响的状态位： N、 OV、 C、 DC 和 Z

机器码：

0000	1000	kkkk	kkkk

说明： 用 8 位立即数 k 减去 W。结果存储在 W 寄存器中。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	写入 W

例 1：
```
SUBLW  02h
执行指令前
    W   =  01h
    C   =  ?
执行指令后
    W   =  01h
    C   =  1    ; 结果为正
    Z   =  0
    N   =  0
```

例 2：
```
SUBLW  02h
执行指令前
    W   =  02h
    C   =  ?
执行指令后
    W   =  00h
    C   =  1    ; 结果为全零
    Z   =  1
    N   =  0
```

例 3：
```
SUBLW  02h
执行指令前
    W   =  03h
    C   =  ?
执行指令后
    W   =  FFh  ;（二进制补码）
    C   =  0    ; 结果为负
    Z   =  0
    N   =  1
```

SUBWF **f 减去 W**

语法： SUBWF f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (f) – (W) → dest

受影响的状态位： N、 OV、 C、 DC 和 Z

机器码：

0101	11da	ffff	ffff

说明： 用寄存器 f 中的内容减去 W 寄存器的内容（通过二进制补码方式进行运算）。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95 （5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

例 1：
```
SUBWF   REG, 1, 0
执行指令前
    REG   =  3
    W     =  2
    C     =  ?
执行指令后
    REG   =  1
    W     =  2
    C     =  1    ; 结果为正
    Z     =  0
    N     =  0
```

例 2：
```
SUBWF   REG, 0, 0
执行指令前
    REG   =  2
    W     =  2
    C     =  ?
执行指令后
    REG   =  2
    W     =  0
    C     =  1    ; 结果为全零
    Z     =  1
    N     =  0
```

例 3：
```
SUBWF   REG, 1, 0
执行指令前
    REG   =  1
    W     =  2
    C     =  ?
执行指令后
    REG   =  FFh  ;（二进制补码）
    W     =  2
    C     =  0    ; 结果为负
    Z     =  0
    N     =  1
```

SUBWFB **f 减去 W （带借位）**

语法： SUBWFB f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (f) – (W) – ($\overline{C}$) → dest

受影响的状态位： N、 OV、 C、 DC 和 Z

机器码：

0101	10da	ffff	ffff

说明： 用 f 寄存器的内容减去 W 的内容和进位标志位（借位）（通过二进制补码方式进行运算）。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

例 1：

```
SUBWFB  REG, 1, 0
执行指令前
    REG  =  19h     (0001 1001)
    W    =  0Dh     (0000 1101)
    C    =  1
执行指令后
    REG  =  0Ch     (0000 1011)
    W    =  0Dh     (0000 1101)
    C    =  1
    Z    =  0
    N    =  0       ; 结果为正
```

例 2：

```
SUBWFB  REG, 0, 0
执行指令前
    REG  =  1Bh     (0001 1011)
    W    =  1Ah     (0001 1010)
    C    =  0
执行指令后
    REG  =  1Bh     (0001 1011)
    W    =  00h
    C    =  1
    Z    =  1       ; 结果为全零
    N    =  0
```

例 3：

```
SUBWFB  REG, 1, 0
执行指令前
    REG  =  03h     (0000 0011)
    W    =  0Eh     (0000 1101)
    C    =  1
执行指令后
    REG  =  F5h     (1111 0100)
                    ; [二进制补码]
    W    =  0Eh     (0000 1101)
    C    =  0
    Z    =  0
    N    =  1       ; 结果为负
```

SWAPF **将 f 的高半字节和低半字节交换**

语法： SWAPF f {,d {,a}}

操作数： 0 ≤ f ≤ 255

d ∈ [0,1]

a ∈ [0,1]

操作： (f<3:0>) → dest<7:4>，

(f<7:4>) → dest<3:0>

受影响的状态位： 无

机器码：

0011	10da	ffff	ffff

说明： 寄存器 f 的高半字节和低半字节相互交换。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例：

```
SWAPF   REG, 1, 0
执行指令前
    REG  =  53h
执行指令后
    REG  =  35h
```

TBLRD **表读**

语法： TBLRD (*; *+; *-; +*)

操作数： 无

操作： 如果执行 TBLRD *，

(程序存储单元 (TBLPTR)) → TABLAT ；

TBLPTR 不改变

如果执行 TBLRD *+，

(程序存储单元 (TBLPTR)) → TABLAT ；

(TBLPTR) + 1 → TBLPTR

如果执行 TBLRD *-，

(程序存储单元 (TBLPTR)) → TABLAT ；

(TBLPTR) – 1 → TBLPTR

如果执行 TBLRD +*，

(TBLPTR) + 1 → TBLPTR ；

(程序存储单元 (TBLPTR)) → TABLAT

受影响的状态位：无

机器码：

0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*

说明： 该指令用于读取程序存储单元 （P.M.）的内容。使用表指针 （TBLPTR）对程序存储单元进行寻址。

TBLPTR （一个 21 位指针）指向程序存储器中的每个字节。TBLPTR的寻址范围为2 MB。

TBLPTR[0] = 0： 程序存储字的低有效字节

TBLPTR[0] = 1： 程序存储字的高有效字节

TBLRD指令可用如下方法修改TBLPTR的值：

- 不变
- 后递增
- 后递减
- 预递增

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	空操作	空操作
空操作	空操作 （读程序存储器）	空操作	空操作 （写 TABLAT）

TBLRD **表读 （续）**

例 1： `TBLRD *+ ;`

执行指令前

TABLAT	=	55h
TBLPTR	=	00A356h
存储单元 （00A356h）	=	34h

执行指令后

TABLAT	=	34h
TBLPTR	=	00A357h

例 2： `TBLRD +* ;`

执行指令前

TABLAT	=	AAh
TBLPTR	=	01A357h
存储单元 （01A357h）	=	12h
存储单元 （01A358h）	=	34h

执行指令后

TABLAT	=	34h
TBLPTR	=	01A358h

TBLWT **表写**

语法： TBLWT (*; *+; *-; +*)

操作数： 无

操作： 如果执行 TBLWT*，

(TABLAT) → 保持寄存器；

TBLPTR 不改变

如果执行 TBLWT*+，

(TABLAT) → 保持寄存器；

(TBLPTR) + 1 → TBLPTR

如果执行 TBLWT*-，

(TABLAT) → 保持寄存器；

(TBLPTR) – 1 → TBLPTR

如果执行 TBLWT+*，

(TBLPTR) + 1 → TBLPTR ；

(TABLAT) → 保持寄存器

受影响的状态位： 无

机器码：

0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*

说明： 此指令使用 TBLPTR 的低 3 位来确定要将 TABLAT 中的内容写入 8 个保持寄存器中的哪一个。该保持寄存器用于对程序存储单元（P.M.）的内容编程。（关于对闪存程序存储器编程的更多详细信息，请参见**第 6.0 节 “ 存储器构成 ”**。）

TBLPTR （一个 21 位指针）指向程序存储器中的每个字节。TBLPTR的寻址范围为2 MB。TBLPTR 的 LSb 选择要访问程序存储单元的哪个字节。

TBLPTR[0] = 0：程序存储字的低有效字节

TBLPTR[0] = 1：程序存储字的高有效字节

TBLWT 指令可用如下方法修改 TBLPTR 的值：

- 不变
- 后递增
- 后递减
- 预递增

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	空操作	空操作	空操作
空操作	空操作（读 TABLAT）	空操作	空操作（写保持寄存器）

TBLWT **表写 （续）**

例 1： `TBLWT *+;`

执行指令前

TABLAT	=	55h
TBLPTR	=	00A356h
保持寄存器 （00A356h）	=	FFh

执行指令后 （表写操作完成）

TABLAT	=	55h
TBLPTR	=	00A357h
保持寄存器 （00A356h）	=	55h

例 2： `TBLWT +*;`

执行指令前

TABLAT	=	34h
TBLPTR	=	01389Ah
保持寄存器 （01389Ah）	=	FFh
保持寄存器 （01389Bh）	=	FFh

执行指令后 （表写操作完成）

TABLAT	=	34h
TBLPTR	=	01389Bh
保持寄存器 （01389Ah）	=	FFh
保持寄存器 （01389Bh）	=	34h

TSTFSZ **测试 f，为 0 则跳过**

语法： TSTFSZ f {,a}

操作数： 0 ≤ f ≤ 255

a ∈ [0,1]

操作： f = 0 则跳过

受影响的状态位： 无

机器码：

0110	011a	ffff	ffff

说明： 如果 f = 0，丢弃执行当前指令过程中已取的下一条指令并执行一条 NOP 指令，使该指令成为双周期指令。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节“ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1（2）

注： 如果跳过且后面跟有 2 字指令，则执行 TSTFSZ 需要 3 个周期。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	空操作

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过且后面跟有 2 字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    TSTFSZ  CNT, 1
NZERO   :
ZERO    :
```

执行指令前
PC = 地址（HERE）
执行指令后
如果 CNT = 00h，
PC = 地址（ZERO）
如果 CNT ≠ 00h，
PC = 地址（NZERO）

XORLW **将立即数与 W 作逻辑异或运算**

语法： XORLW k

操作数： 0 ≤ k ≤ 255

操作： (W) .XOR. k → W

受影响的状态位： N 和 Z

机器码：

0000	1010	kkkk	kkkk

说明： 将 W 的内容与 8 位立即数 k 进行逻辑异或运算。结果存储在 W 寄存器中。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	写入 W

示例：

```
XORLW   0AFh
```

执行指令前
W = B5h
执行指令后
W = 1Ah

XORWF **将 W 与 f 作逻辑异或运算**

语法： XORWF f {,d {,a}}

操作数： $0 \le f \le 255$

$d \in [0,1]$

$a \in [0,1]$

操作： (W) .XOR.(f) → dest

受影响的状态位： N 和 Z

机器码：

0001	10da	ffff	ffff

说明： 将 W 的内容与寄存器 f 的内容进行逻辑异或运算。如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

如果 a 为 0，选择快速操作存储区。如果 a 为 1，使用 BSR 选择 GPR 存储区。

如果 a 为 0 且使能了扩展指令集，只要 f ≤ 95（5Fh），指令就将以立即数变址寻址模式进行操作。详情请参见**第 26.2.3 节 “ 立即数变址寻址模式中针对字节和针对位的指令 ”**。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例：

```
XORWF   REG, 1, 0
```

执行指令前
REG = AFh
W = B5h
执行指令后
REG = 1Ah
W = B5h

26.2 扩展指令集

除了PIC18指令集的75条标准指令之外，PIC18F87J90系列器件还提供了针对核心CPU功能的可选扩展指令。这些新增的功能包括 8 条额外的指令，增加了间接和变址寻址操作，并使得许多标准 PIC18 指令可以实现立即数变址寻址。

扩展指令集的额外功能在默认情况下对未编程器件是使能的。用户必须在编程期间将 XINST 配置位正确置 1 或清零，从而使能或禁止这些功能。

扩展指令集中的指令可以全部被归为立即数操作类指令，它们既可以对文件选择寄存器进行操作，也可以使用这些寄存器进行变址寻址。还为其中两条指令 ADDFSR 和 SUBFSR 提供了使用 FSR2 的特例形式，即 ADDULNK 和 SUBULNK，这两条指令允许在执行后自动返回。

这些扩展的指令专门用于优化用高级语言特别是C语言编写的可重入程序代码（也就是递归或使用软件堆栈的代码）。此外，它们使用户能更高效地用高级语言对数据结构执行某些操作。这些操作包括：

- 在进入和退出子程序时对软件堆栈空间进行动态分配和释放
- 函数指针调用
- 对软件堆栈指针进行操作
- 对软件堆栈中的变量进行操作

表 26-3 提供了扩展指令集中的指令汇总。**第 26.2.2 节 “ 扩展指令集 ”** 对这些指令进行了详细说明。表 26-1（第 340 页）提供了标准和扩展的 PIC18 指令集的操作码字段说明。

注：	扩展指令集和立即数变址寻址模式是专为优化用 C 语言编写的应用程序而设计的，用户可能不会在汇编程序中直接使用这些指令。对于那些需要查看编译器生成代码的用户，这些命令的语法可作为参考。

26.2.1 扩展指令的语法

大部分扩展指令都使用变址参数，使用一个文件选择寄存器和某一偏移量来指定源寄存器或目标寄存器。当指令的参数作为变址寻址的一部分时，会用方括号（“[ ]”）把它括起来。这用于表示此参数用作变址或偏移量。如果 MPASM™ 汇编器发现一个变址或偏移量没有被括起来，它就会给出出错信息。

当使能扩展指令集时，方括号也用于表示针对字节和针对位的指令中的变址参数。这是对指令语法的额外更改。更多详细信息，请参见**第 26.2.3.1 节 “ 标准 PIC18 命令的扩展指令语法 ”**。

注：	以前，在 PIC18 和早期的指令集中使用方括号来表示可选参数。在此文本和以后的文本中，可选参数将用大扩号 （“{ }”）表示。

表 26-3： PIC18 指令集的扩展

助记符，操作数		说明	周期数	16 位指令字				受影响的状态位
				MSb			LSb	
ADDFSR	f, k	将立即数加到 FSR	1	1110	1000	ffkk	kkkk	无
ADDULNK	k	将立即数加到 FSR2 并返回	2	1110	1000	11kk	kkkk	无
CALLW		使用 WREG 调用子程序	2	0000	0000	0001	0100	无
MOVSF	z_s, f_d	将 z_s（源）移入 第一个字	2	1110	1011	0zzz	zzzz	无
		f_d（目标） 第二个字		1111	ffff	ffff	ffff	
MOVSS	z_s, z_d	将 z_s（源）移入 第一个字	2	1110	1011	1zzz	zzzz	无
		z_d（目标） 第二个字		1111	xxxx	xzzz	zzzz	
PUSHL	k	将立即数保存到 FSR2， FSR2 递减 1	1	1110	1010	kkkk	kkkk	无
SUBFSR	f, k	FSR 减去立即数	1	1110	1001	ffkk	kkkk	无
SUBULNK	k	FSR2 减去立即数并返回	2	1110	1001	11kk	kkkk	无

26.2.2 扩展指令集

ADDFSR **将立即数加到 FSR**

语法： ADDFSR f, k

操作数： $0 \le k \le 63$

$f \in [0, 1, 2]$

操作： FSR(f) + k → FSR(f)

受影响的状态位： 无

机器码：

1110	1000	ffkk	kkkk

说明： 将 6 位立即数 k 加到由 f 指定的 FSR 的内容。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	写入 FSR

示例： ADDFSR 2, 23h

执行指令前
FSR2 = 03FFh

执行指令后
FSR2 = 0422h

ADDULNK **将立即数加到 FSR2 并返回**

语法： ADDULNK k

操作数： $0 \le k \le 63$

操作： FSR2 + k → FSR2，
(TOS) → PC

受影响的状态位： 无

机器码：

1110	1000	11kk	kkkk

说明： 将 6 位立即数 k 加到 FSR2 的内容。然后通过将 TOS 的值装入 PC，执行 RETURN。

执行该指令需要两个周期；在第二个周期执行一条 NOP 指令。

该指令可以被认为是 ADDFSR 指令的特例，其中 f = 3（二进制 “11”），它仅针对 FSR2 进行操作。

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	写入 FSR
空操作	空操作	空操作	空操作

示例： ADDULNK 23h

执行指令前
FSR2 = 03FFh
PC = 0100h

执行指令后
FSR2 = 0422h
PC = (TOS)

注： 所有的 PIC18 指令都可能在其指令助记符之前使用可选的标号参数，用于符号寻址。如果使用了标号，那么指令格式将变为：{label} 指令参数。

CALLW **使用 WREG 调用子程序**

语法： CALLW

操作数： 无

操作： (PC + 2) → TOS，
(W) → PCL，
(PCLATH) → PCH，
(PCLATU) → PCU

受影响的状态位： 无

机器码：

0000	0000	0001	0100

说明 首先，返回地址（PC + 2）被压入返回堆栈。接下来，将 W 寄存器的内容写入 PCL，PCL 现有的值被丢弃。然后，PCLATH 和 PCLATU 的内容被分别锁存到 PCH 和 PCU。第二个周期执行一条 NOP 指令，并同时取下一条新指令。

和 CALL 不一样，该指令没有更新 W、STATUS 或 BSR 寄存器的选项。

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 WREG	将 PC 压入堆栈	空操作
空操作	空操作	空操作	空操作

示例： HERE CALLW

执行指令前
PC = 地址（HERE）
PCLATH = 10h
PCLATU = 00h
W = 06h

执行指令后
PC = 001006h
TOS = 地址（HERE + 2）
PCLATH = 10h
PCLATU = 00h
W = 06h

MOVSF **将变址寻址单元内容送入 f**

语法： MOVSF $[z_s]$, f_d

操作数： $0 \le z_s \le 127$
$0 \le f_d \le 4095$

操作： $((FSR2) + z_s) \rightarrow f_d$

受影响的状态位： 无

机器码：

第一个字（源）	1110	1011	0zzz	$zzzz_s$
第二个字（目标）	1111	ffff	ffff	$ffff_d$

说明： 将源寄存器的内容送入目标寄存器 f_d。通过将第一个字中的 7 位立即数偏移量 z_s 与FSR2的值相加来确定源寄存器的实际地址。第二个字中的 12 位立即数 f_d 指向目标寄存器的地址。两个地址均可以是 4096 字节的数据空间（000h 到 FFFh）中的任何存储单元。

MOVSF 指令中的目标寄存器不能是 PCL、TOSU、TOSH 或 TOSL。

如果计算得到的源地址指向间接寻址寄存器，将返回 00h。

指令字数： 2

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	确定源地址	确定源地址	读源寄存器
译码	空操作 无假读	空操作	写寄存器 f（目标寄存器）

示例： MOVSF [05h], REG2

执行指令前
FSR2 = 80h
85h 单元的内容 = 33h
REG2 = 11h

执行指令后
FSR2 = 80h
85h 单元的内容 = 33h
REG2 = 33h

MOVSS **在变址寻址单元之间传送数据**

语法： MOVSS [z_s], [z_d]

操作数： $0 \le z_s \le 127$

$0 \le z_d \le 127$

操作： ((FSR2) + z_s) → ((FSR2) + z_d)

受影响的状态位： 无

机器码：

第一个字 （源）	1110	1011	1zzz	$zzzz_s$
第二个字 （目标）	1111	xxxx	xzzz	$zzzz_d$

说明 将源寄存器的内容送入目标寄存器。通过将 FSR2 中的值分别加上 7 位立即数偏移量 z_s 和 z_d 来确定源寄存器和目标寄存器的地址。两个寄存器都可以是 4096 字节数据存储空间 （000h 到 FFFh）中的任意存储单元。

MOVSS 指令不能使用 PCL、TOSU、TOSH 或 TOSL 作为目标寄存器。

如果计算得到的源地址指向间接寻址寄存器，将返回 00h。如果计算得到的目标地址指向间接寻址寄存器，将执行一条 NOP 指令。

指令字数： 2

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	确定源地址	确定源地址	读源寄存器
译码	确定目标地址	确定目标地址	写目标寄存器

示例：
```
MOVSS [05h], [06h]
```
执行指令前
FSR2 = 80h
85h 单元的内容 = 33h
86h 单元的内容 = 11h
执行指令后
FSR2 = 80h
85h 单元的内容 = 33h
86h 单元的内容 = 33h

PUSHL **将立即数保存到 FSR2， FSR2 递减 1**

语法： PUSHL k

操作数： $0 \le k \le 255$

操作： k → (FSR2)，
FSR2 – 1 → FSR2

受影响的状态位： 无

机器码：

1111	1010	kkkk	kkkk

说明： 8 位立即数 k 被写入由 FSR2 指定的数据存储单元。操作完后 FSR2 递减 1。

此指令允许用户将值压入软件堆栈。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读取 k	处理数据	写入目标寄存器

示例：
```
PUSHL 08h
```
执行指令前
FSR2H:FSR2L = 01ECh
存储单元 （01ECh） = 00h

执行指令后
FSR2H:FSR2L = 01EBh
存储单元 （01ECh） = 08h

SUBFSR **FSR 减去立即数**

语法： SUBFSR f, k

操作数： 0 ≤ k ≤ 63

f ∈ [0, 1, 2]

操作： FSRf – k → FSRf

受影响的状态位： 无

机器码：

1110	1001	ffkk	kkkk

说明： 用 f 指定的 FSR 的内容减去 6 位立即数 k。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例： SUBFSR 2, 23h

执行指令前
FSR2 = 03FFh

执行指令后
FSR2 = 03DCh

SUBULNK **FSR2 减去立即数并返回**

语法： SUBULNK k

操作数： 0 ≤ k ≤ 63

操作： FSR2 – k → FSR2，
(TOS) → PC

受影响的状态位： 无

机器码：

1110	1001	11kk	kkkk

说明： 用 FSR 的内容减去 6 位立即数 k，然后通过将 TOS 的值装入 PC，执行 RETURN。

执行该指令需要两个指令周期，第二个指令周期执行一条 NOP 指令。

该指令可以被认为是 SUBFSR 指令的特例，其中 f = 3（二进制 “11”）；它仅针对 FSR2 进行操作。

指令字数： 1

指令周期数： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器
空操作	空操作	空操作	空操作

示例： SUBULNK 23h

执行指令前
FSR2 = 03FFh
PC = 0100h

执行指令后
FSR2 = 03DCh
PC = (TOS)

26.2.3 立即数变址寻址模式中针对字节和针对位的指令

注：	使能 PIC18 扩展指令集可能导致常规应用程序运行不正常或完全失败。

一旦使能扩展指令集，除了可以使用 8 条新命令之外，还可以使用立即数变址寻址模式（**第 6.6.1 节 “ 使用立即数偏移量进行变址寻址 ”**）。这将导致标准 PIC18 指令集中大部分指令的地址解析方法有很大变化。

当禁止扩展指令集时，嵌入在操作码中的地址被视作立即数存储单元：可以是快速操作存储区中的存储单元（a = 0），或由 BSR 指定的 GPR 存储区中的存储单元（a = 1）。当使能扩展指令集且 a = 0 时，地址小于或等于 5Fh 的文件寄存器参数被解析为 FSR2 中的指针值的偏移量，而不是一个立即数地址。对于实际应用来说，这意味着所有使用快速操作 RAM 位作为参数的指令，即所有针对字节或针对位的指令，或者几乎半数的核心 PIC18指令，在使能了扩展指令集时操作都会有所不同。

当 FSR2 的内容为 00h 时，快速操作 RAM 的边界会被重新映射到它们的原始值。这对于编写向下兼容的代码很有用处。如果使用此技术，有必要在 C 程序调用汇编子程序时保存 FSR2 的值并在返回时将它恢复，这样做的目的是保护堆栈指针。用户还必须记住扩展指令集的语法要求（见**第 26.2.3.1 节 “ 标准 PIC18 命令的扩展指令语法 ”**）。

虽然立即数变址寻址模式对于动态堆栈和指针操作很有用处，但是如果不小心对错误的寄存器进行了简单的算术运算也会非常麻烦。已经习惯使用 PIC18 编程的用户必须记住，在使能了扩展指令集后，地址小于或等于 5Fh 的寄存器用于立即数变址寻址。

下页提供了在立即数变址寻址模式中，一些针对字节和位的指令的代表示例，通过这些示例可以看出指令执行如何受到影响。示例中的操作数条件适用于所有这些类型的指令。

26.2.3.1 标准 PIC18 命令的扩展指令语法

当使能了扩展指令集时，立即数偏移量 “k” 被用来替换标准的针对字节和位的命令中的文件寄存器参数 “f”。如前所述，只有在 “f” 小于或等于 5Fh 时才会发生这种情况。当使用偏移量时，偏移量必须用方括号 “[ ]” 标出。因为在扩展指令集中，编译器将方括号中的值解析为变址地址或偏移量。省略方括号，或在方括号内使用大于 5Fh 的值会在 MPASM 汇编器中产生错误。

如果变址参数已被加上了方括号，那么就不再需要指定快速操作 RAM 参数；此参数被自动假定为 0。这与标准操作 （禁止扩展指令集时）刚好相反。在变址寻址模式中，声明快速操作 RAM 位也将在 MPASM 汇编器中产生错误。

目标参数 “d” 的操作和以前一样。

在 MPASM 汇编器的最新版本中，必须明确调用对扩展指令集的语言支持。可以通过命令行选项 /y 或在源代码中加入 PE 伪指令进行调用。

26.2.4 使能扩展指令集时的注意事项

需要注意的是，并非所有用户都有必要使用扩展指令集，尤其是那些不使用软件堆栈的用户。

此外，立即数变址寻址模式可能会给写入 PIC18 汇编器的常规应用程序带来问题。这是因为常规的指令会尝试寻址快速操作存储区中地址低于 5Fh 的寄存器。当使能了扩展指令集时，这些地址被解析为相对于 FSR2 的立即数偏移量，所以应用程序可能会读或写错误的地址。

将应用程序移植到 PIC18F87J90 系列器件时，考虑代码的类型是非常重要的。在使用扩展指令集时，用 C 语言编写的代码较长的可重入应用程序会运行得很好，而大量使用快速操作存储区的常规应用程序不会获得任何益处。

ADDWF **将 W 与变址寻址单元的内容相加（立即数变址寻址模式）**

语法： ADDWF [k] {,d}

操作数： 0 ≤ k ≤ 95

d ∈ [0,1]

操作： (W) + ((FSR2) + k) → dest

受影响的状态位： N、 OV、 C、 DC 和 Z

机器码：

0010	01d0	kkkk	kkkk

说明： 将 W 的内容与由 FSR2 加上偏移量 k 指定的寄存器的内容相加。

如果 d 为 0，结果存储在 W 中。如果 d 为 1，结果存回寄存器 f。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读取 k	处理数据	写入目标寄存器

示例： ADDWF [OFST],0

执行指令前

W = 17h
OFST = 2Ch
FSR2 = 0A00h
0A2Ch 单元的内容 = 20h

执行指令后

W = 37h
0A2Ch 单元的内容 = 20h

BSF **将变址寻址单元相应位置 1（立即数变址寻址模式）**

语法： BSF [k], b

操作数： 0 ≤ f ≤ 95

0 ≤ b ≤ 7

操作： 1 → ((FSR2) + k)<b>

受影响的状态位： 无

机器码：

1000	bbb0	kkkk	kkkk

说明： 将由 FSR2 加上偏移量 k 指定的寄存器中的位 b 置 1。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入目标寄存器

示例： BSF [FLAG_OFST], 7

执行指令前

FLAG_OFST = 0Ah
FSR2 = 0A00h
0A0Ah 单元的内容 = 55h

执行指令后

0A0Ah 单元的内容 = D5h

SETF **将变址寻址单元置全 1（立即数变址寻址模式）**

语法： SETF [k]

操作数： 0 ≤ k ≤ 95

操作： FFh → ((FSR2) + k)

受影响的状态位： 无

机器码：

0110	1000	kkkk	kkkk

说明： 将由 FSR2 加上偏移量 k 指定的寄存器的内容置为 FFh。

指令字数： 1

指令周期数： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读取 k	处理数据	写寄存器

示例： SETF [OFST]

执行指令前

OFST = 2Ch
FSR2 = 0A00h
0A2Ch 单元的内容 = 00h

执行指令后

0A2Ch 单元的内容 = FFh

26.2.5 使用 MICROCHIP MPLAB® IDE 工具的注意事项

最新版本的 Microchip 软件工具完全支持 PIC18F87J90 系列器件的扩展指令集。软件工具包括 MPLAB C18 C 编译器、MPASM 汇编器和 MPLAB 集成开发环境（Integrated Development Environment，IDE）。

在选择了目标器件进行软件开发后，MPLAB IDE 将自动按默认模式设置该器件的配置位。XINST 配置位的默认设置是 1，使能扩展指令集和立即数变址寻址模式。在编程时必须将 XINST 位置 1 才能确保使用扩展指令集开发的应用程序能够正确执行。

要使用扩展指令集开发软件，用户必须设置他们的语言工具以实现对扩展指令和变址寻址模式的支持。根据所使用的环境，可以通过以下几种方法：

- 开发环境中的菜单选项或对话框，允许用户配置项目的语言工具及其设置
- 命令行选项
- 源代码中的伪指令

这些选项在不同的编译器、汇编器和开发环境中将有所不同。建议用户在其开发系统所附带的文档中查询相应的文档。

27.0 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

27.1 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。

27.2 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。

27.3 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。

27.4 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

27.5 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

27.6 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

27.7 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。

27.8 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。

27.9 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。

27.10 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）。

27.11 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）。

27.12 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。

27.13 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）。

28.0 电气特性

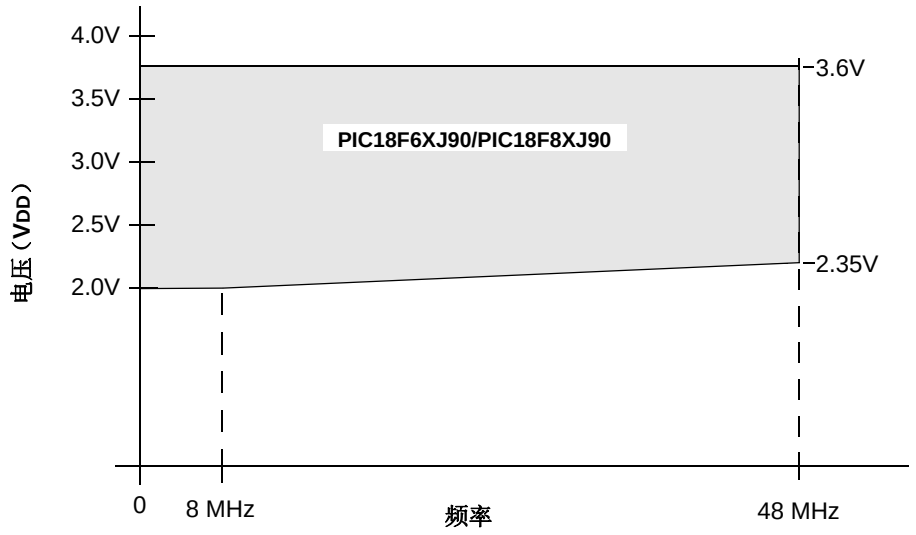
绝对最大额定值 (†)

环境温度..... -40°C 至 +100°C
储存温度..... -65°C 至 +150°C
任一仅用作数字功能的 I/O 引脚或 $\overline{MCLR}$ 相对于 VSS 的电压 （除 VDD 外）.....-0.3V 至 6.0V
任一数模组合引脚相对于 VSS 的电压 （除 VDD 和 $\overline{MCLR}$ 外）.....-0.3V 至 （VDD + 0.3V）
VDDCORE 引脚相对于 VSS 的电压 .....-0.3V 至 2.75V
VDD 引脚相对于 VSS 的电压.....-0.3V 至 3.6V
总功耗 （**注 1**）.....1.0W
流出 VSS 引脚的最大电流.....300 mA
流入 VDD 引脚的最大电流 .....250 mA
PORTA<7:6>、 PORTB 和 PORTC I/O 引脚的最大输出灌电流 .....25 mA
PORTD、 PORTE 和 PORTJ I/O 引脚的最大输出灌电流.....8 mA
PORTA<5:0>、 PORTF、 PORTG 和 PORTH I/O 引脚的最大输出灌电流 .....2 mA
PORTA<7:6>、 PORTB 和 PORTC I/O 引脚的最大输出拉电流 .....25 mA
PORTD、 PORTE 和 PORTJ I/O 引脚的最大输出拉电流.....8 mA
PORTA<5:0>、 PORTF、 PORTG 和 PORTH I/O 引脚的最大输出拉电流 .....2 mA
所有端口的最大总灌电流 .....200 mA

注 1： 功耗按如下公式计算：
$Pdis = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$

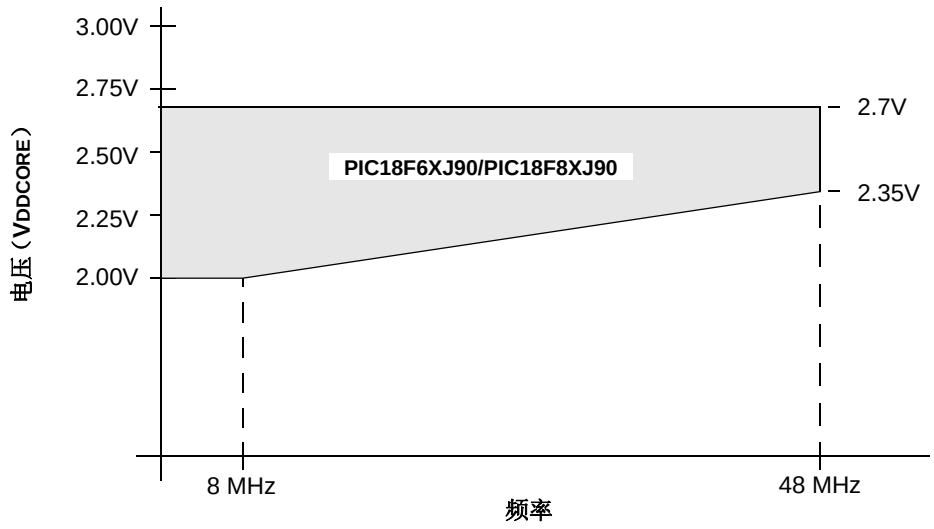
† **注：** 如果运行条件超过了上述 “ 绝对额定最大值 ”，即可能对器件造成永久性损坏。这仅是极限参数，我们不建议器件工作在极限值甚至超过上述极限值。器件长时间工作在额定最大值条件下，其稳定性可能受到影响。

图 28-1：　　电压－频率关系图，使能稳压器 （工业级）(1)



注　1： 当使能片上稳压器时，其 BOR 电路在 VDD 达到无法进行全速工作的电压之前，将自动触发器件复位。

图 28-2：　　电压－频率关系图，禁止稳压器 （工业级）(1,2)



注　1： 当禁止片上稳压器时， VDD 和 VDDCORE 必须维持在 VDDCORE ≤ VDD ≤ 3.6V。

28.1 直流特性： PIC18F87J90 系列的供电电压 （工业级）

PIC18F87J90 系列 （工业级）			标准工作条件 （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	最大值	单位	条件
D001	VDD	供电电压	VDDCORE 2.0	— —	3.6 3.6	V V	ENVREG 连接到 VSS ENVREG 连接到 VDD
D001B	VDDCORE	单片机内核的外部电源	2.0	—	2.70	V	ENVREG 连接到 VSS
D001C	AVDD	模拟供电电压	VDD – 0.3	—	VDD + 0.3	V	
D001D	AVSS	模拟地电位	VSS – 0.3	—	VSS + 0.3	V	
D002	VDR	RAM 数据保持电压 (1)	1.5	—	—	V	
D003	VPOR	确保内部上电复位信号的 VDD 启动电压	—	—	0.7	V	详情请参见第 5.3 节 “ 上电复位 （POR） ”
D004	SVDD	确保内部上电复位信号的 VDD 上升速率	0.05	—	—	V/ms	详情请参见第 5.3 节 “ 上电复位 （POR） ”
D005	VBOR	欠压复位电压	—	1.8	—	V	

注 1： 这是在不丢失 RAM 数据的前提下，休眠模式或器件复位期间 VDD 所能降到的最小电压值。

28.2 直流特性： PIC18F87J90 系列的掉电电流和供电电流 （工业级）

PIC18F87J90 系列 （工业级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）				
参数编号	器件	典型值	最大值	单位	条件	
	掉电电流 （IPD） (1)					
	所有器件	0.4	1.4	µA	-40°C	VDD = 2.0V(4) （休眠模式）
		0.1	1.4	µA	+25°C	
		0.8	6	µA	+60°C	
		5.5	10.2	µA	+85°C	
	所有器件	0.5	1.5	µA	-40°C	VDD = 2.5V(4) （休眠模式）
		0.1	1.5	µA	+25°C	
		1	8	µA	+60°C	
		6.8	12.6	µA	+85°C	
	所有器件	2.9	7	µA	-40°C	VDD = 3.3V(5) （休眠模式）
		3.6	7	µA	+25°C	
		4.1	10	µA	+60°C	
		9.6	19	µA	+85°C	

注 1： 在休眠模式下，掉电电流与振荡器类型无关。掉电电流是在器件处于休眠模式、所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，禁止所有会带来新增电流的功能部件 （如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要受工作电压、频率和模式的影响。其他因素，如 I/O 引脚负载和开关速率、振荡器类型和电路、内部代码执行模式和温度也会对电流消耗产生影响。

有效工作模式下，所有 IDD 测量值的测试条件为：

OSC1 = 外部方波，轨到轨满幅；所有 I/O 引脚均为三态，上拉至 VDD ；

$\overline{\text{MCLR}}$ = VDD ； 根据具体应用使能 / 禁止 WDT。

3： 标准低成本 32 kHz 晶振的工作温度范围为 -10°C 至 +70°C。扩展级温度晶振的成本则高很多。

4： 禁止稳压器 （ENVREG = 0，连接到 VSS）。

5： 使能稳压器 （ENVREG = 1，连接到 VDD， REGSLP = 1）。

28.2 直流特性： PIC18F87J90 系列的掉电电流和供电电流 （工业级）（续）

PIC18F87J90 系列 （工业级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）					
参数编号	器件	典型值	最大值	单位	条件		
	供电电流 （IDD） (2,3)						
	所有器件	5	14.2	µA	-40°C	VDD = 2.0V, VDDCORE = 2.0V(4)	FOSC = 31 kHz （**RC_RUN** 模式，内部振荡器作为时钟源）
		5.5	14.2	µA	+25°C		
		10	19.0	µA	+85°C		
	所有器件	6.8	16.5	µA	-40°C	VDD = 2.5V, VDDCORE = 2.5V(4)	
		7.6	16.5	µA	+25°C		
		14	22.4	µA	+85°C		
	所有器件	37	84	µA	-40°C	VDD = 3.3V(5)	
		51	84	µA	+25°C		
		72	108	µA	+85°C		
	所有器件	0.43	0.82	mA	-40°C	VDD = 2.0V, VDDCORE = 2.0V(4)	FOSC = 1 MHz （**RC_RUN** 模式，内部振荡器作为时钟源）
		0.47	0.82	mA	+25°C		
		0.52	0.95	mA	+85°C		
	所有器件	0.52	0.98	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V(4)	
		0.57	0.98	mA	+25°C		
		0.63	1.10	mA	+85°C		
	所有器件	0.59	0.96	mA	-40°C	VDD = 3.3V(5)	
		0.65	0.96	mA	+25°C		
		0.72	1.18	mA	+85°C		
	所有器件	0.88	1.45	mA	-40°C	VDD = 2.0V, VDDCORE = 2.0V(4)	FOSC = 4 MHz （**RC_RUN** 模式，内部振荡器作为时钟源）
		1	1.45	mA	+25°C		
		1.1	1.58	mA	+85°C		
	所有器件	1.2	1.72	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V(4)	
		1.3	1.72	mA	+25°C		
		1.4	1.85	mA	+85°C		
	所有器件	1.3	2.87	mA	-40°C	VDD = 3.3V(5)	
		1.4	2.87	mA	+25°C		
		1.5	2.96	mA	+85°C		

注 1： 在休眠模式下，掉电电流与振荡器类型无关。掉电电流是在器件处于休眠模式、所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，禁止所有会带来新增电流的功能部件 （如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要受工作电压、频率和模式的影响。其他因素，如 I/O 引脚负载和开关速率、振荡器类型和电路、内部代码执行模式和温度也会对电流消耗产生影响。

有效工作模式下，所有 IDD 测量值的测试条件为：

OSC1 = 外部方波，轨到轨满幅；所有 I/O 引脚均为三态，上拉至 VDD ；

MCLR = VDD ； 根据具体应用使能 / 禁止 WDT。

3： 标准低成本 32 kHz 晶振的工作温度范围为 -10°C 至 +70°C。扩展级温度晶振的成本则高很多。

4： 禁止稳压器 （ENVREG = 0，连接到 VSS）。

5： 使能稳压器 （ENVREG = 1，连接到 VDD， REGSLP = 1）。

28.2 直流特性： PIC18F87J90 系列的掉电电流和供电电流 （工业级）（续）

PIC18F87J90 系列 （工业级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）					
参数编号	器件	典型值	最大值	单位	条件		
	供电电流 （IDD）续 (2,3)						
	所有器件	3	9.4	µA	-40°C	VDD = 2.0V, VDDCORE = 2.0V(4)	FOSC = 31 kHz （**RC_IDLE** 模式，内部振荡器作为时钟源）
		3.3	9.4	µA	+25°C		
		8.5	17.2	µA	+85°C		
	所有器件	4	10.5	µA	-40°C	VDD = 2.5V, VDDCORE = 2.5V(4)	
		4.3	10.5	µA	+25°C		
		10.3	19.5	µA	+85°C		
	所有器件	34	82	µA	-40°C	VDD = 3.3V(5)	
		48	82	µA	+25°C		
		69	105	µA	+85°C		
	所有器件	0.33	0.75	mA	-40°C	VDD = 2.0V, VDDCORE = 2.0V(4)	FOSC = 1 MHz （**RC_IDLE** 模式，内部振荡器作为时钟源）
		0.37	0.75	mA	+25°C		
		0.41	0.84	mA	+85°C		
	所有器件	0.39	0.78	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V(4)	
		0.42	0.78	mA	+25°C		
		0.47	0.91	mA	+85°C		
	所有器件	0.43	0.82	mA	-40°C	VDD = 3.3V(5)	
		0.48	0.82	mA	+25°C		
		0.54	0.95	mA	+85°C		
	所有器件	0.53	0.98	mA	-40°C	VDD = 2.0V, VDDCORE = 2.0V(4)	FOSC = 4 MHz （**RC_IDLE** 模式，内部振荡器作为时钟源）
		0.57	0.98	mA	+25°C		
		0.61	1.12	mA	+85°C		
	所有器件	0.63	1.14	mA	-40°C	VDD = 2.5V, VDDCORE = 2.5V(4)	
		0.67	1.14	mA	+25°C		
		0.72	1.25	mA	+85°C		
	所有器件	0.7	1.27	mA	-40°C	VDD = 3.3V(5)	
		0.76	1.27	mA	+25°C		
		0.82	1.45	mA	+85°C		

注 1： 在休眠模式下，掉电电流与振荡器类型无关。掉电电流是在器件处于休眠模式、所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，禁止所有会带来新增电流的功能部件 （如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要受工作电压、频率和模式的影响。其他因素，如 I/O 引脚负载和开关速率、振荡器类型和电路、内部代码执行模式和温度也会对电流消耗产生影响。

有效工作模式下，所有 IDD 测量值的测试条件为：

OSC1 = 外部方波，轨到轨满幅；所有 I/O 引脚均为三态，上拉至 VDD ；

$\overline{\text{MCLR}}$ = VDD ； 根据具体应用使能 / 禁止 WDT。

3： 标准低成本 32 kHz 晶振的工作温度范围为 -10°C 至 +70°C。扩展级温度晶振的成本则高很多。

4： 禁止稳压器 （ENVREG = 0，连接到 VSS）。

5： 使能稳压器 （ENVREG = 1，连接到 VDD， REGSLP = 1）。

28.2 直流特性： PIC18F87J90 系列的掉电电流和供电电流 （工业级）（续）

PIC18F87J90 系列 （工业级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）					
参数编号	器件	典型值	最大值	单位	条件		
	供电电流 （IDD）续 (2,3)						
	所有器件	0.17	0.35	mA	-40°C	VDD = 2.0V， VDDCORE = 2.0V(4)	FOSC = 1 MHz （PRI_RUN 模式， EC 振荡器）
		0.18	0.35	mA	+25°C		
		0.20	0.42	mA	+85°C		
	所有器件	0.29	0.52	mA	-40°C	VDD = 2.5V， VDDCORE = 2.5V(4)	
		0.31	0.52	mA	+25°C		
		0.34	0.61	mA	+85°C		
	所有器件	0.59	1.1	mA	-40°C	VDD = 3.3V(5)	
		0.44	0.85	mA	+25°C		
		0.42	0.85	mA	+85°C		
	所有器件	0.70	1.25	mA	-40°C	VDD = 2.0V， VDDCORE = 2.0V(4)	FOSC = 4 MHz （PRI_RUN 模式， EC 振荡器）
		0.75	1.25	mA	+25°C		
		0.79	1.36	mA	+85°C		
	所有器件	1.10	1.7	mA	-40°C	VDD = 2.5V， VDDCORE = 2.5V(4)	
		1.10	1.7	mA	+25°C		
		1.12	1.82	mA	+85°C		
	所有器件	1.55	1.95	mA	-40°C	VDD = 3.3V(5)	
		1.47	1.89	mA	+25°C		
		1.54	1.92	mA	+85°C		
	所有器件	9.9	14.8	mA	-40°C	VDD = 2.5V， VDDCORE = 2.5V(4)	FOSC = 48 MHz （PRI_RUN 模式， EC 振荡器）
		9.5	14.8	mA	+25°C		
		10.1	15.2	mA	+85°C		
	所有器件	13.3	23.2	mA	-40°C	VDD = 3.3V(5)	
		12.2	22.7	mA	+25°C		
		12.1	22.7	mA	+85°C		

注 1： 在休眠模式下，掉电电流与振荡器类型无关。掉电电流是在器件处于休眠模式、所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，禁止所有会带来新增电流的功能部件 （如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要受工作电压、频率和模式的影响。其他因素，如 I/O 引脚负载和开关速率、振荡器类型和电路、内部代码执行模式和温度也会对电流消耗产生影响。

有效工作模式下，所有 IDD 测量值的测试条件为：

OSC1 = 外部方波，轨到轨满幅；所有 I/O 引脚均为三态，上拉至 VDD ；

MCLR = VDD ； 根据具体应用使能 / 禁止 WDT。

3： 标准低成本 32 kHz 晶振的工作温度范围为 -10°C 至 +70°C。扩展级温度晶振的成本则高很多。

4： 禁止稳压器 （ENVREG = 0，连接到 VSS）。

5： 使能稳压器 （ENVREG = 1，连接到 VDD， REGSLP = 1）。

28.2 直流特性： PIC18F87J90 系列的掉电电流和供电电流 （工业级）（续）

PIC18F87J90 系列 （工业级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）					
参数编号	器件	典型值	最大值	单位	条件		
	供电电流 （IDD）续 (2,3)						
	所有器件	4.5	5.2	mA	-40°C	VDD = 2.5V， VDDCORE = 2.5V(4)	FOSC = 4 MHz， 16 MHz 内部 （**PRI_RUN HSPLL** 模式）
		4.4	5.2	mA	+25°C		
		4.5	5.2	mA	+85°C		
	所有器件	5.7	6.7	mA	-40°C	VDD = 3.3V(5)	
		5.5	6.3	mA	+25°C		
		5.3	6.3	mA	+85°C		
	所有器件	10.8	13.5	mA	-40°C	VDD = 2.5V， VDDCORE = 2.5V(4)	FOSC = 10 MHz， 40 MHz 内部 （**PRI_RUN HSPLL** 模式）
		10.8	13.5	mA	+25°C		
		9.9	13.0	mA	+85°C		
	所有器件	13.4	24.1	mA	-40°C	VDD = 3.3V(5)	
		12.3	20.2	mA	+25°C		
		11.2	19.5	mA	+85°C		

注 1： 在休眠模式下，掉电电流与振荡器类型无关。掉电电流是在器件处于休眠模式、所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，禁止所有会带来新增电流的功能部件 （如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要受工作电压、频率和模式的影响。其他因素，如 I/O 引脚负载和开关速率、振荡器类型和电路、内部代码执行模式和温度也会对电流消耗产生影响。

有效工作模式下，所有 IDD 测量值的测试条件为：

OSC1 = 外部方波，轨到轨满幅；所有 I/O 引脚均为三态，上拉至 VDD ；

MCLR = VDD ； 根据具体应用使能 / 禁止 WDT。

3： 标准低成本 32 kHz 晶振的工作温度范围为 -10°C 至 +70°C。扩展级温度晶振的成本则高很多。

4： 禁止稳压器 （ENVREG = 0，连接到 VSS）。

5： 使能稳压器 （ENVREG = 1，连接到 VDD， REGSLP = 1）。

28.2 直流特性： PIC18F87J90 系列的掉电电流和供电电流 （工业级）（续）

PIC18F87J90 系列 （工业级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）					
参数编号	器件	典型值	最大值	单位	条件		
	供电电流 （IDD）续 (2,3)						
	所有器件	0.10	0.26	mA	-40°C	VDD = 2.0V，VDDCORE = 2.0V(4)	FOSC = 1 MHz （**PRI_IDLE** 模式，EC 振荡器）
		0.07	0.18	mA	+25°C		
		0.09	0.22	mA	+85°C		
	所有器件	0.25	0.48	mA	-40°C	VDD = 2.5V，VDDCORE = 2.5V(4)	
		0.13	0.30	mA	+25°C		
		0.10	0.26	mA	+85°C		
	所有器件	0.45	0.68	mA	-40°C	VDD = 3.3V(5)	
		0.26	0.45	mA	+25°C		
		0.30	0.54	mA	+85°C		
	所有器件	0.36	0.60	mA	-40°C	VDD = 2.0V，VDDCORE = 2.0V(4)	FOSC = 4 MHz （**PRI_IDLE** 模式，EC 振荡器）
		0.33	0.56	mA	+25°C		
		0.35	0.56	mA	+85°C		
	所有器件	0.52	0.81	mA	-40°C	VDD = 2.5V，VDDCORE = 2.5V(4)	
		0.45	0.70	mA	+25°C		
		0.46	0.70	mA	+85°C		
	所有器件	0.80	1.15	mA	-40°C	VDD = 3.3V(5)	
		0.66	0.98	mA	+25°C		
		0.65	0.98	mA	+85°C		
	所有器件	5.2	6.5	mA	-40°C	VDD = 2.5V，VDDCORE = 2.5V(4)	FOSC = 48 MHz （**PRI_IDLE** 模式，EC 振荡器）
		4.9	5.9	mA	+25°C		
		3.4	4.5	mA	+85°C		
	所有器件	6.2	12.4	mA	-40°C	VDD = 3.3V(5)	
		5.9	11.5	mA	+25°C		
		5.8	11.5	mA	+85°C		

注 1： 在休眠模式下，掉电电流与振荡器类型无关。掉电电流是在器件处于休眠模式、所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，禁止所有会带来新增电流的功能部件 （如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要受工作电压、频率和模式的影响。其他因素，如 I/O 引脚负载和开关速率、振荡器类型和电路、内部代码执行模式和温度也会对电流消耗产生影响。

有效工作模式下，所有 IDD 测量值的测试条件为：

OSC1 = 外部方波，轨到轨满幅；所有 I/O 引脚均为三态，上拉至 VDD ；

MCLR = VDD ； 根据具体应用使能 / 禁止 WDT。

3： 标准低成本 32 kHz 晶振的工作温度范围为 -10°C 至 +70°C。扩展级温度晶振的成本则高很多。

4： 禁止稳压器 （ENVREG = 0，连接到 VSS）。

5： 使能稳压器 （ENVREG = 1，连接到 VDD， REGSLP = 1）。

28.2 直流特性： PIC18F87J90 系列的掉电电流和供电电流 （工业级）（续）

PIC18F87J90 系列 （工业级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）					
参数编号	器件	典型值	最大值	单位	条件		
	供电电流 （IDD）续 (2,3)						
	所有器件	18	35	µA	-40°C	VDD = 2.0V， VDDCORE = 2.0V(4)	FOSC = 32 kHz(3) （**SEC_RUN** 模式， Timer1 作为时钟源）
		19	35	µA	+25°C		
		28	49	µA	+85°C		
	所有器件	20	45	µA	-40°C	VDD = 2.5V， VDDCORE = 2.5V(4)	
		21	45	µA	+25°C		
		32	61	µA	+85°C		
	所有器件	0.06	0.11	mA	-40°C	VDD = 3.3V(5)	
		0.07	0.11	mA	+25°C		
		0.09	0.15	mA	+85°C		
	所有器件	14	28	µA	-40°C	VDD = 2.0V， VDDCORE = 2.0V(4)	FOSC = 32 kHz(3) （**SEC_IDLE** 模式， Timer1 作为时钟源）
		15	28	µA	+25°C		
		24	43	µA	+85°C		
	所有器件	15	31	µA	-40°C	VDD = 2.5V， VDDCORE = 2.5V(4)	
		16	31	µA	+25°C		
		27	50	µA	+85°C		
	所有器件	0.05	0.10	mA	-40°C	VDD = 3.3V(5)	
		0.06	0.10	mA	+25°C		
		0.08	0.14	mA	+85°C		

注 1： 在休眠模式下，掉电电流与振荡器类型无关。掉电电流是在器件处于休眠模式、所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，禁止所有会带来新增电流的功能部件 （如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要受工作电压、频率和模式的影响。其他因素，如 I/O 引脚负载和开关速率、振荡器类型和电路、内部代码执行模式和温度也会对电流消耗产生影响。

有效工作模式下，所有 IDD 测量值的测试条件为：

OSC1 = 外部方波，轨到轨满幅；所有 I/O 引脚均为三态，上拉至 VDD ；

MCLR = VDD ； 根据具体应用使能 / 禁止 WDT。

3： 标准低成本 32 kHz 晶振的工作温度范围为 -10°C 至 +70°C。扩展级温度晶振的成本则高很多。

4： 禁止稳压器 （ENVREG = 0，连接到 VSS）。

5： 使能稳压器 （ENVREG = 1，连接到 VDD， REGSLP = 1）。

28.2 直流特性： PIC18F87J90 系列的掉电电流和供电电流 （工业级）（续）

PIC18F87J90 系列 （工业级）		标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）					
参数编号	器件	典型值	最大值	单位	条件		
D022	**模块增加电流 （ΔIWDT、 ΔIOSCB 和 ΔIAD）**						
	看门狗定时器	2.1	7.0	µA	-40°C	VDD = 2.0V， VDDCORE = 2.0V(4)	
		2.2	7.0	µA	+25°C		
		4.3	9.5	µA	+85°C		
		3.0	8.0	µA	-40°C	VDD = 2.5V， VDDCORE = 2.5V(4)	
		3.1	8.0	µA	+25°C		
		5.5	10.4	µA	+85°C		
		5.9	12.1	µA	-40°C	VDD = 3.3V	
		6.2	12.1	µA	+25°C		
		6.9	13.6	µA	+85°C		
D024 （ΔILCD）	**LCD 模块**	2(6,7)	5	µA	+25°C	VDD = 2.0V	梯形电阻网络 CPEN = 0； CKSEL<1:0> = 00； CS<1:0> = 10； LP<3:0> = 0100
		2.7(6,7)	5	µA	+25°C	VDD = 2.5V	
		3.5(6,7)	7	µA	+25°C	VDD = 3.0V	
		16(7)	25	µA	+25°C	VDD = 2.0V	电荷泵 BIAS<2:0> = 111； CPEN = 1； CKSEL<1:0> = 11； CS<1:0> = 10
		17(7)	25	µA	+25°C	VDD = 2.5V	
		24(7)	40	µA	+25°C	VDD = 3.0V	
D025 （ΔIOSCB）	**RTCC + 带 32 kHz 晶振的 Timer1 振荡器 (6)**	0.9	4.0	µA	-10°C	VDD = 2.0V， VDDCORE = 2.0V(4)	Timer1 采用 32 kHz 晶振 (3)
		1.0	4.5	µA	+25°C		
		1.1	4.5	µA	+85°C		
		1.1	4.5	µA	-10°C	VDD = 2.5V， VDDCORE = 2.5V(4)	Timer1 采用 32 kHz 晶振 (3)
		1.2	5.0	µA	+25°C		
		1.2	5.0	µA	+85°C		
		1.6	6.5	µA	-10°C	VDD = 3.3V	Timer1 采用 32 kHz 晶振 (3)
		1.6	6.5	µA	+25°C		
		2.1	8.0	µA	+85°C		
D026 （ΔIAD）	**A/D 转换器**	3.0	10.0	µA	-40°C 至 +85°C	VDD = 2.0V， VDDCORE = 2.0V(4)	A/D 启动，但不在进行转换
		3.0	10.0	µA	-40°C 至 +85°C	VDD = 2.5V， VDDCORE = 2.5V(4)	

注 1： 在休眠模式下，掉电电流与振荡器类型无关。掉电电流是在器件处于休眠模式、所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，禁止所有会带来新增电流的功能部件 （如 WDT、 Timer1 振荡器和 BOR 等）时测得的。

2： 供电电流主要受工作电压、频率和模式的影响。其他因素，如 I/O 引脚负载和开关速率、振荡器类型和电路、内部代码执行模式和温度也会对电流消耗产生影响。

有效工作模式下，所有 IDD 测量值的测试条件为：

OSC1 = 外部方波，轨到轨满幅；所有 I/O 引脚均为三态，上拉至 VDD；

MCLR = VDD； 根据具体应用使能 / 禁止 WDT。

3： 标准低成本 32 kHz 晶振的工作温度范围为 -10°C 至 +70°C。扩展级温度晶振的成本则高很多。

4： 禁止稳压器 （ENVREG = 0，连接到 VSS）。

5： 使能稳压器 （ENVREG = 1，连接到 VDD， REGSLP = 1）。

28.3 直流特性： PIC18F87J90 系列 （工业级）

直流特性			标准工作条件 （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	最大值	单位	条件
	VIL	**输入低电压**				
		所有 I/O 端口：				
D030		带 TTL 缓冲器	VSS	0.15 VDD	V	VDD < 3.3V
D030A			—	0.8	V	3.3V ≤ VDD ≤ 3.6V
D031		带施密特触发器缓冲器	VSS	0.2 VDD	V	
D031A		带 RC3 和 RC4	VSS	0.3 VDD	V	使能 I^2C™
D031B			VSS	0.8	V	使能 SMBus
D032		$\overline{MCLR}$	VSS	0.2 VDD	V	
D033		OSC1	VSS	0.3 VDD	V	HS 和 HSPLL 模式
D033A		OSC1	VSS	0.2 VDD	V	EC 和 ECPLL 模式
D034		T13CKI	VSS	0.3	V	
	VIH	**输入高电压**				
		不能承受 5.5V 电压的 I/O 端口：				
D040		带 TTL 缓冲器	0.25 VDD + 0.8V	VDD	V	VDD < 3.3V
D040A			2.0	VDD	V	3.3V ≤ VDD ≤ 3.6V
D041		带施密特触发器缓冲器	0.8 VDD	VDD	V	
D041A		带 RC3 和 RC4	0.7 VDD	VDD	V	使能 I^2C
D041B			2.1	VDD	V	使能 SMBus
		能承受 5.5V 电压的 I/O 端口：				
		带 TTL 缓冲器	0.25 VDD + 0.8V	5.5	V	VDD < 3.3V
			2.0	5.5	V	3.3V ≤ VDD ≤ 3.6V
		带施密特触发器缓冲器	0.8 VDD	5.5	V	
D042		$\overline{MCLR}$	0.8 VDD	VDD	V	
D043		OSC1	0.7 VDD	VDD	V	HS 和 HSPLL 模式
D043A		OSC1	0.8 VDD	VDD	V	EC 和 ECPLL 模式
D044		T13CKI	1.6	VDD	V	
	IIL	**输入泄漏电流 (1)**				
D060		具有模拟功能的 I/O 端口	—	200	nA	VSS ≤ VPIN ≤ VDD， 引脚处于高阻态
		仅用作数字功能的 I/O 端口	—	200	nA	VSS ≤ VPIN ≤ 5.5V
D061		$\overline{MCLR}$	—	±1	µA	VSS ≤ VPIN ≤ VDD
D063		OSC1	—	±1	µA	VSS ≤ VPIN ≤ VDD
	IPU	**弱上拉电流**				
D070	IPURB	PORTB 弱上拉电流	30	400	µA	VDD = 3.3V， VPIN = VSS

注 1： 负电流定义为引脚的拉电流。

28.3 直流特性： PIC18F87J90 系列 （工业级）（续）

直流特性			标准工作条件 （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	最大值	单位	条件
	VOL	**输出低电压**				
D080		I/O 端口：				
		PORTA、 PORTF、 PORTG 和 PORTH	—	0.4	V	IOL = 2 mA， VDD = 3.3V，-40°C 至 +85°C
		PORTD、 PORTE 和 PORTJ	—	0.4	V	IOL = 3.4 mA， VDD = 3.3V，-40°C 至 +85°C
		PORTB 和 PORTC	—	0.4	V	IOL = 3.4 mA， VDD = 3.3V，-40°C 至 +85°C
D083		OSC2/CLKO （EC 和 ECPLL 模式）	—	0.4	V	IOL = 1.6 mA， VDD = 3.3V，-40°C 至 +85°C
	VOH	**输出高电压 (1)**				
D090		I/O 端口：			V	
		PORTA、 PORTF、 PORTG 和 PORTH	2.4	—	V	IOH = -2 mA， VDD = 3.3V，-40°C 至 +85°C
		PORTD、 PORTE 和 PORTJ	2.4	—	V	IOH = -2 mA， VDD = 3.3V，-40°C 至 +85°C
		PORTB 和 PORTC	2.4	—	V	IOH = -2 mA， VDD = 3.3V，-40°C 至 +85°C
D092		OSC2/CLKO （INTOSC、EC 和 ECPLL 模式）	2.4	—	V	IOH = -1 mA， VDD = 3.3V，-40°C 至 +85°C
		输出引脚上的容性负载规范				
D100(4)	COSC2	OSC2 引脚	—	15	pF	当外部时钟用于驱动 OSC1 时、处于 HS 模式下
D101	CIO	所有 I/O 引脚和 OSC2	—	50	pF	为满足交流时序规范
D102	CB	SCL 和 SDA	—	400	pF	I2C™ 规范

注 1： 负电流定义为引脚的拉电流。

28.4 直流特性：CTMU 电流源规范

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
	IOUT1	CTMU 电流源，基本范围	—	550	—	nA	CTMUICON<1:0> = 01
	IOUT2	CTMU 电流源， 10x 范围	—	5.5	—	µA	CTMUICON<1:0> = 10
	IOUT3	CTMU 电流源， 100x 范围	—	55	—	µA	CTMUICON<1:0> = 11

注 1： 电流微调范围的中点为标称值 （CTMUICON<7:2> = 000000）。

表 28-1： 存储器编程要求

直流特性			标准工作条件 （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 †	最大值	单位	条件
		闪存程序存储器					
D130	EP	单元耐擦写能力	10K	—	—	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
D132B	VPEW	自定时擦除或写操作的电压 VDD VDDCORE	 2.35 2.25	 — —	 3.6 2.7	 V V	 ENVREG 连接到 VDD ENVREG 连接到 VSS
D133A	TIW	自定时写周期	—	2.8	—	ms	
D133B	TIE	自定时块擦除周期	—	33	—	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	—	3	14	mA	
D140	TWE	每个擦除周期的写入次数	—	—	1		对于每个物理地址

† 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 28-2： 比较器规范

工作条件：3.0V ≤ 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	—	±5.0	±25	mV	
D301	VICM	输入共模电压	0	—	AVDD – 1.5	V	
D302	CMRR	共模抑制比	55	—	—	dB	
D303	TRESP	响应时间 (1)	—	150	400	ns	
D304	TMC2OV	比较器模式改变到输出有效的时间 *	—	—	10	µs	

注 1： 响应时间是在比较器的一个输入端电压为 (AVDD – 1.5)/2，而另一个输入端从 VSS 跳变到 VDD 时测得的。

表 28-3： 参考电压规范

工作条件：3.0V ≤ 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**
D310	VRES	分辨率	VDD/24	—	VDD/32	LSb	
D311	VRAA	绝对精度	—	—	1/2	LSb	
D312	VRUR	单位电阻值 （R）	—	2k	—	Ω	
D313	TSET	稳定时间 (1)	—	—	10	µs	

注 1： 稳定时间是在 CVRR = 1 并且 CVR<3:0> 从 0000 跳变到 1111 时测得的。

表 28-4： 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**
	VRGOUT	稳压器输出电压	—	2.5	—	V	
	CEFC	外部滤波电容值	4.7	10	—	µF	电容必须具有较低的等效串联电阻 （< 5Ω）

表 28-5： 内部 LCD 稳压器规范

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**
	CFLY	反激电容	0.47	4.7	—	µF	电容必须具有较低的等效串联电阻
	VBIAS	LCDBIAS0 和 LCDBIAS3 之间的 VPK-PK	—	3.40	3.6	V	BIAS<2:0> = 111
			—	3.27	—	V	BIAS<2:0> = 110
			—	3.14	—	V	BIAS<2:0> = 101
			—	3.01	—	V	BIAS<2:0> = 100
			—	2.88	—	V	BIAS<2:0> = 011
			—	2.75	—	V	BIAS<2:0> = 010
			—	2.62	—	V	BIAS<2:0> = 001
			—	2.49	—	V	BIAS<2:0> = 000

28.5 交流（时序）特性

28.5.1 时序参数符号体系

时序参数符号采用以下格式之一进行创建：

1. TppS2ppS
2. TppS
3. TCC:ST （仅用于 I²C 规范）
4. Ts （仅用于 I²C 规范）

T			
	F	频率	T 时间

小写字母（pp）及其含义：

pp			
	cc	CCP1	osc OSC1
	ck	CLKO	rd $\overline{RD}$
	cs	$\overline{CS}$	rw $\overline{RD}$ 或 $\overline{WR}$
	di	SDI	sc SCK
	do	SDO	ss $\overline{SS}$
	dt	数据输入	t0 T0CKI
	io	I/O 端口	t1 T13CKI
	mc	$\overline{MCLR}$	wr $\overline{WR}$

大写字母及其含义：

S			
	F	下降	P 周期
	H	高	R 上升
	I	无效（高阻）	V 有效
	L	低	Z 高阻
仅用于 I²C			
	AA	输出访问	High 高
	BUF	总线空闲	Low 低

TCC:ST （仅用于 I²C 规范）

CC			
	HD	保持	SU 建立
ST			
	DAT	数据输入保持	STO 停止条件
	STA	启动条件	

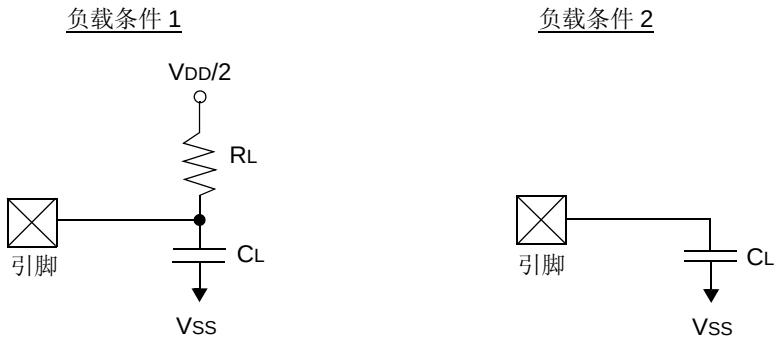
28.5.2 时序条件

表 28-6 中指定的温度和电压适用于所有的时序规范（除非另外声明）。图 28-3 规定了时序规范的负载条件。

表 28-6： 温度和电压规范 —— 交流

交流特性	**标准工作条件 （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**和**第 28.3 节**中所述。

图 28-3： 器件时序规范的负载条件



RL = 464Ω

CL = 50 pF 对于除 OSC2/CLKO/RA6 外的所有引脚包括作为输出的 D 和 E 端口

CL = 15 pF 对于 OSC2/CLKO/RA6

28.5.3 时序图和规范

图 28-4： 外部时钟时序

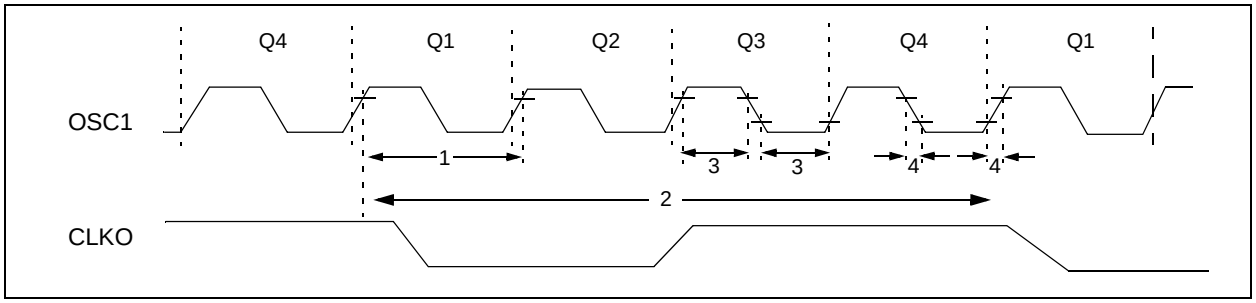


表 28-7： 外部时钟时序要求

参数编号	符号	特性	最小值	最大值	单位	条件
1A	FOSC	外部 CLKI 频率 (1)	DC	48	MHz	EC 振荡器模式
			DC	10		ECPLL 振荡器模式
		振荡器频率 (1)	4	25	MHz	HS 振荡器模式
			4	10		HSPLL 振荡器模式
1	TOSC	外部 CLKI 周期 (1)	20.8	—	ns	EC 振荡器模式
			100	—		ECPLL 振荡器模式
		振荡器周期 (1)	40.0	250	ns	HS 振荡器模式
			100	250		HSPLL 振荡器模式
2	TCY	指令周期 (1)	83.3	—	ns	TCY = 4/FOSC，工业级
3	TOSL，TOSH	外部时钟输入（OSC1）的高电平或低电平时间	10	—	ns	HS 振荡器模式
4	TOSR，TOSF	外部时钟输入（OSC1）的上升或下降时间	—	7.5	ns	HS 振荡器模式

注 1： 对于除 PLL 外的所有配置，指令周期（TCY）等于输入振荡器时基周期的四倍。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 OSC1/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）。

表 28-8： PLL 时钟时序规范 （VDD = 2.15V 至 3.6V）

参数编号	符号	特性	最小值	典型值 †	最大值	单位	条件
F10	FOSC	振荡器频率范围	4	—	10	MHz	仅用于 HS 模式
F11	FSYS	片上 VCO 系统频率	16	—	40	MHz	仅用于 HS 模式
F12	t_{rc}	PLL 起振时间 （锁定时间）	—	—	2	ms	
F13	ΔCLK	CLKO 稳定性 （抗抖动性）	-2	—	+2	%	

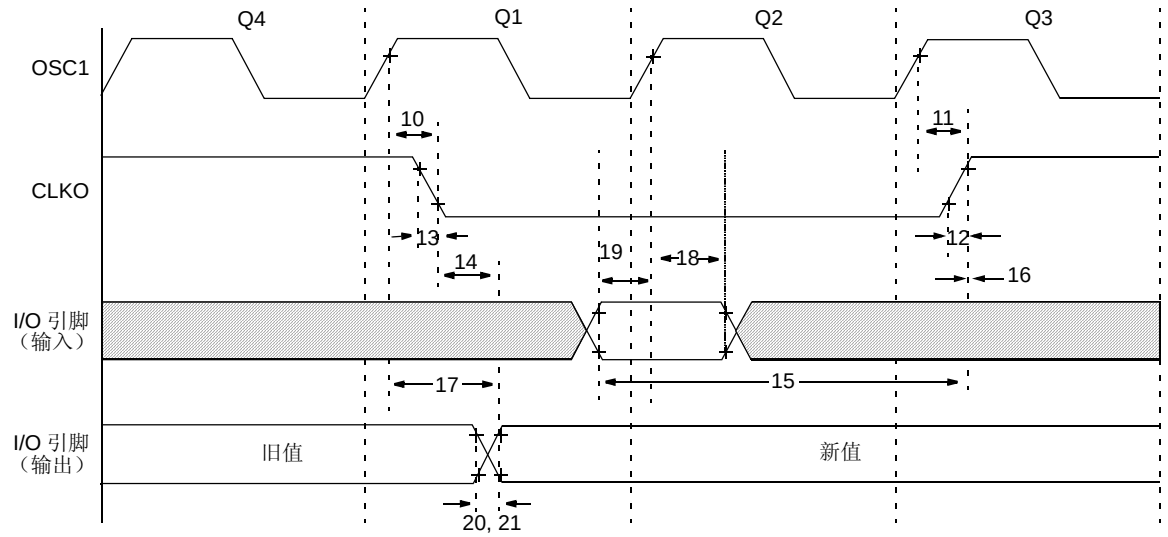
† 除非另外声明，否则“典型值”栏中的数据均为3.3V和25°C条件下的值。这些参数仅供设计参考，未经测试。

表 28-9： 内部 RC 精度 （INTOSC 和 INTRC 时钟源）

PIC18F87J90 系列 （工业级）		**标准工作条件 （除非另外声明）** 工作温度 -40°C ≤ TA ≤ +85°C （工业级）					
参数编号	**器件**	**最小值**	**典型值**	**最大值**	**单位**	**条件**	
	在频率为 8 MHz、4 MHz、2 MHz、1 MHz、500 kHz、250 kHz、125 kHz 和 31 kHz 时的 INTOSC 精度 (1)						
	所有器件	-2	+/-1	2	%	+25°C	VDD = 2.7-3.3V
		-5	—	5	%	-10°C至+85°C	VDD = 2.0-3.3V
		-10	+/-1	10	%	-40°C至 +85°C	VDD = 2.0-3.3V
	在频率为 31 kHz 时的 INTRC 精度 (1)						
	所有器件	21.7	—	40.3	kHz	-40°C 至 +85°C	VDD = 2.0-3.3V

注 1： 31 kHz 时钟的精度规范由给定时间内为其提供时钟的时钟源决定。当 INTSRC（OSCTUNE<7>）为 1 时，使用 INTOSC 精度规范。当 INTSRC 为 0 时，使用 INTRC 精度规范。

图 28-5：　　CLKO 和 I/O 时序



注：　负载条件请参见图 28-3。

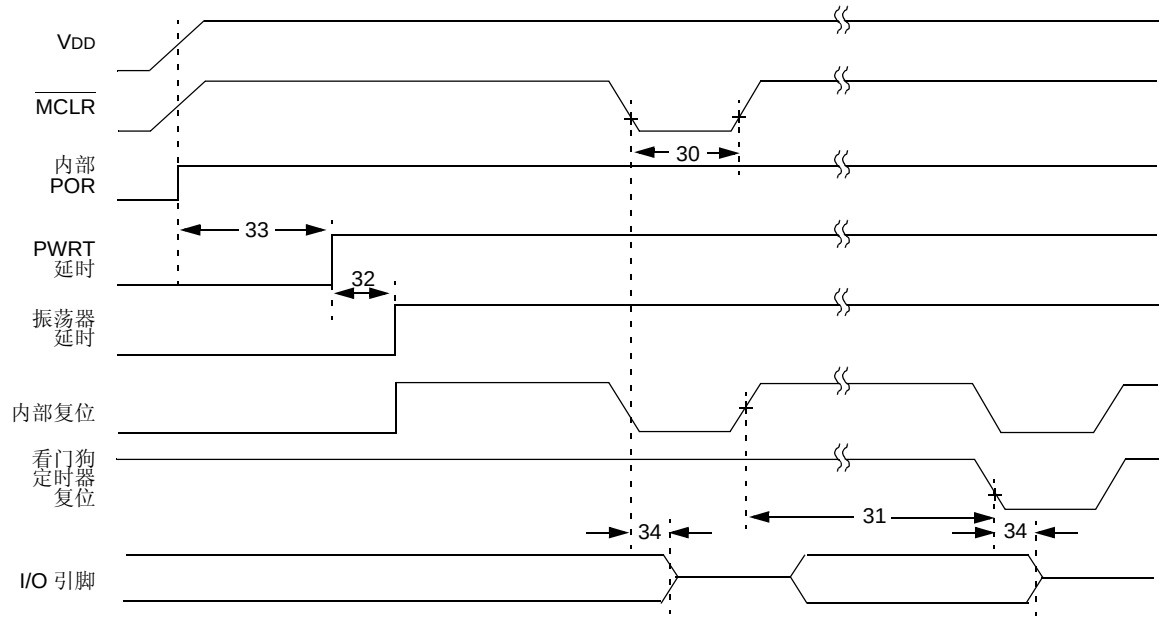
表 28-10：　　CLKO 和 I/O 时序要求

参数编号	符号	特性	最小值	典型值	最大值	单位	条件
10	TosH2ckL	OSC1↑到 CLKO↓的时间	—	75	200	ns	（注 1）
11	TosH2ckH	OSC1↑到 CLKO↑的时间	—	75	200	ns	（注 1）
12	TckR	CLKO 上升时间	—	15	30	ns	（注 1）
13	TckF	CLKO 下降时间	—	15	30	ns	（注 1）
14	TckL2ioV	CLKO↓到端口输出有效的时间	—	—	0.5 TCY + 20	ns	
15	TioV2ckH	CLKO↑之前端口输入有效的时间	0.25 TCY + 25	—	—	ns	
16	TckH2ioI	CLKO↑之后端口输入保持的时间	0	—	—	ns	
17	TosH2ioV	OSC1↑（Q1 周期）到端口输出有效的时间	—	50	150	ns	
18	TosH2ioI	OSC1↑（Q2 周期）到端口输入无效的时间 （I/O 输入保持时间）	100	—	—	ns	
19	TioV2osH	端口输入有效到 OSC1↑的时间（I/O 输入建立时间）	0	—	—	ns	
20	TioR	端口输出上升时间	—	—	6	ns	
21	TioF	端口输出下降时间	—	—	5	ns	
22†	TINP	INTx 引脚高电平或低电平时间	TCY	—	—	ns	
23†	TRBP	RB<7:4> 电平变化中断 INTx 高电平或低电平时间	TCY	—	—	ns	

† 这些参数是与任何内部时钟边沿无关的异步事件。

注　1： 测量是在 EC 模式下进行的，其中 CLKO 输出为 4 x TOSC。

图 28-6：　　复位、看门狗定时器、振荡器起振定时器和上电延时定时器时序



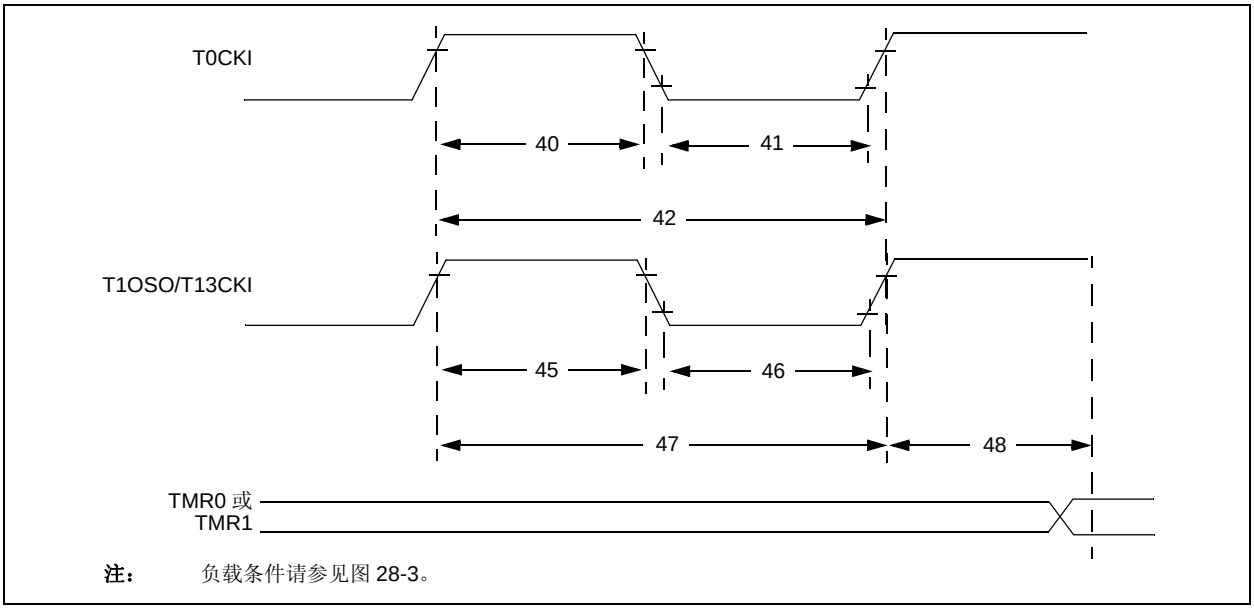
注：　负载条件请参见图 28-3。

表 28-11：　　复位、看门狗定时器、振荡器起振定时器、上电延时定时器和欠压复位要求

参数编号	符号	特性	最小值	典型值	最大值	单位	条件
30	TMCL	$\overline{\text{MCLR}}$ 脉冲宽度 （低电平）	2 TCY	10 TCY	—		**（注 1）**
31	TWDT	看门狗定时器超时周期 （无后分频器）	3.4	4.0	4.6	ms	
32	TOST	振荡器起振定时器周期	1024 TOSC	—	1024 TOSC		TOSC = OSC1 周期
33	TPWRT	上电延时定时器周期	45.8	65.5	85.2	ms	
34	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	—	2	—	µs	
38	TCSD	CPU 启动时间	—	10	—	µs	
				200		µs	使能稳压器并置于休眠状态
39	TIOBST	INTOSC 稳定时间	—	1	—	µs	

注　1：为确保器件复位， $\overline{\text{MCLR}}$ 必须保持低电平至少 2 个 TCY 或 400 µs （取二者中较小值）。

图 28-7：　　TIMER0 和 TIMER1 外部时钟时序

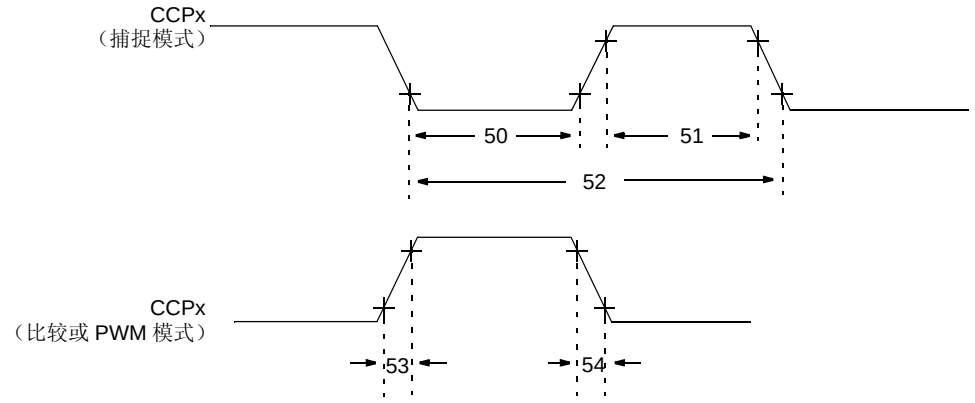


注：　负载条件请参见图 28-3。

表 28-12：　　TIMER0 和 TIMER1 外部时钟要求

参数编号	符号	特性		最小值	最大值	单位	条件
40	TT0H	T0CKI 高电平脉冲宽度	无预分频器	0.5 TCY + 20	—	ns	
			带预分频器	10	—	ns	
41	TT0L	T0CKI 低电平脉冲宽度	无预分频器	0.5 TCY + 20	—	ns	
			带预分频器	10	—	ns	
42	TT0P	T0CKI 周期	无预分频器	TCY + 10	—	ns	
			带预分频器	取如下二者中较大值：20 ns 或 (TCY + 40)/N	—	ns	N = 预分频值（1, 2, 4,..., 256）
45	TT1H	T13CKI 高电平时间	同步，无预分频器	0.5 TCY + 20	—	ns	
			同步，带预分频器	10	—	ns	
			异步	30	—	ns	
46	TT1L	T13CKI 低电平时间	同步，无预分频器	0.5 TCY + 5	—	ns	
			同步，带预分频器	10	—	ns	
			异步	30	—	ns	
47	TT1P	T13CKI 输入周期	同步	取如下二者中较大值：20 ns 或 (TCY + 40)/N	—	ns	N = 预分频值（1, 2, 4, 8）
			异步	60	—	ns	
	FT1	T13CKI 振荡器输入频率范围		DC	50	kHz	
48	TCKE2TMRI	从外部 T13CKI 时钟边沿到定时器递增的延时		2 TOSC	7 TOSC	—	

图 28-8：　　捕捉 / 比较 /PWM 时序 （CCP1 和 CCP2 模块）

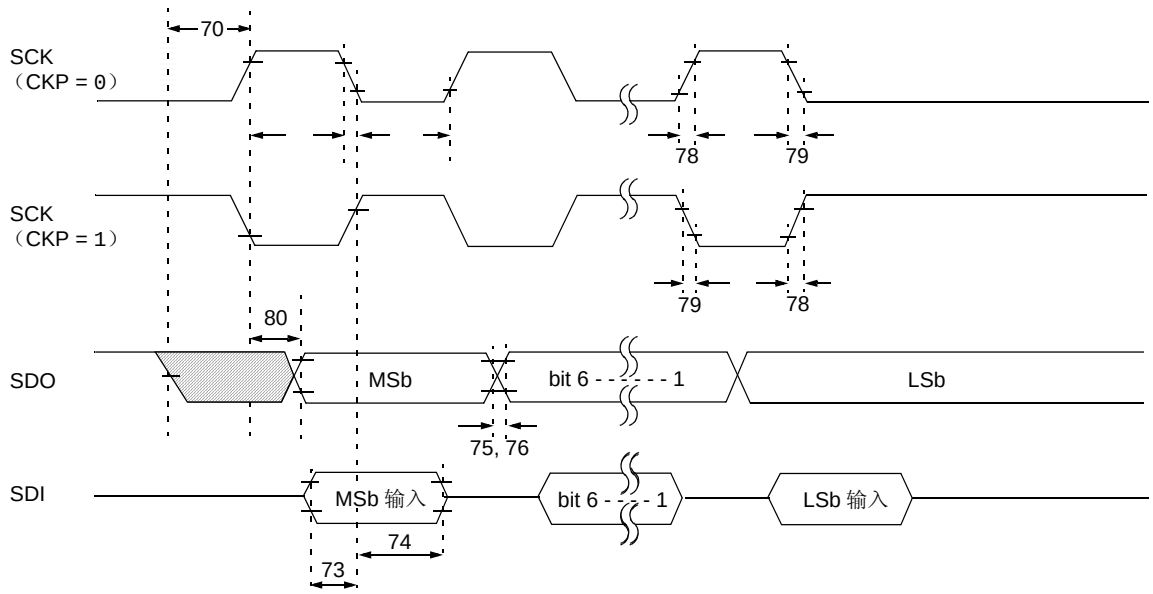


注：　负载条件请参见图 28-3。

表 28-13：　　捕捉 / 比较 /PWM 要求 （CCP1 和 CCP2 模块）

参数编号	符号	特性		最小值	最大值	单位	条件
50	TccL	CCPx 输入低电平时间	无预分频器	0.5 TCY + 20	—	ns	
			带预分频器	10	—	ns	
51	TccH	CCPx 输入高电平时间	无预分频器	0.5 TCY + 20	—	ns	
			带预分频器	10	—	ns	
52	TccP	CCPx 输入周期		$\frac{3\ T_{CY} + 40}{N}$	—	ns	N = 预分频值（1、 4 或 16）
53	TccR	CCPx 输出下降时间		—	25	ns	
54	TccF	CCPx 输出下降时间		—	25	ns	

图 28-9：　　SPI 主模式时序示例 （CKE = 0）



注： 负载条件请参见图 28-3。

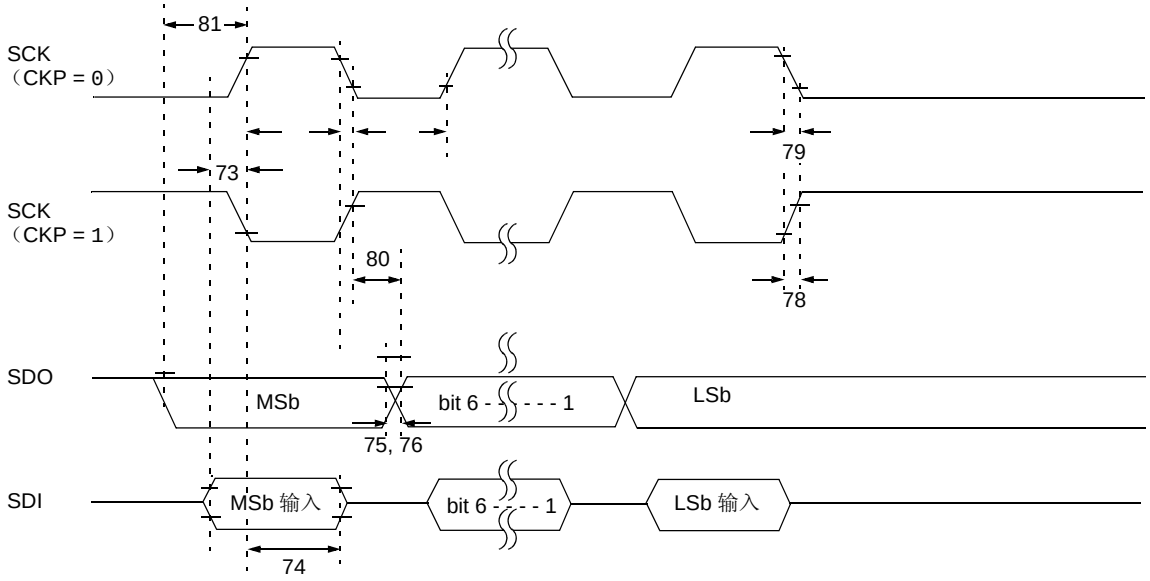
表 28-14：　　SPI 模式要求示例 （主模式， CKE = 0）

参数编号	符号	特性	最小值	最大值	单位	条件
73	TDIV2SCH，TDIV2SCL	SDI 数据输入到 SCK 边沿的建立时间	20	—	ns	
73A	TB2B	字节 1 的最后一个时钟边沿到字节 2 的第一个时钟边沿之间的时间	1.5 TCY + 40	—	ns	（注 2）
74	TSCH2DIL，TSCL2DIL	SDI 数据输入到 SCK 边沿的保持时间	40	—	ns	
75	TDOR	SDO 数据输出上升时间	—	25	ns	
76	TDOF	SDO 数据输出下降时间	—	25	ns	
78	TSCR	SCK 输出上升时间 （主模式）	—	25	ns	
79	TSCF	SCK 输出下降时间 （主模式）	—	25	ns	
80	TSCH2DOV，TSCL2DOV	在 SCK 边沿之后 SDO 数据输出有效的时间	—	50	ns	

注　1： 要求使用参数 73A。

2： 仅当使用参数 71A 和 72A 时。

图 28-10： SPI 主模式时序示例 （CKE = 1）



注： 负载条件请参见图 28-3。

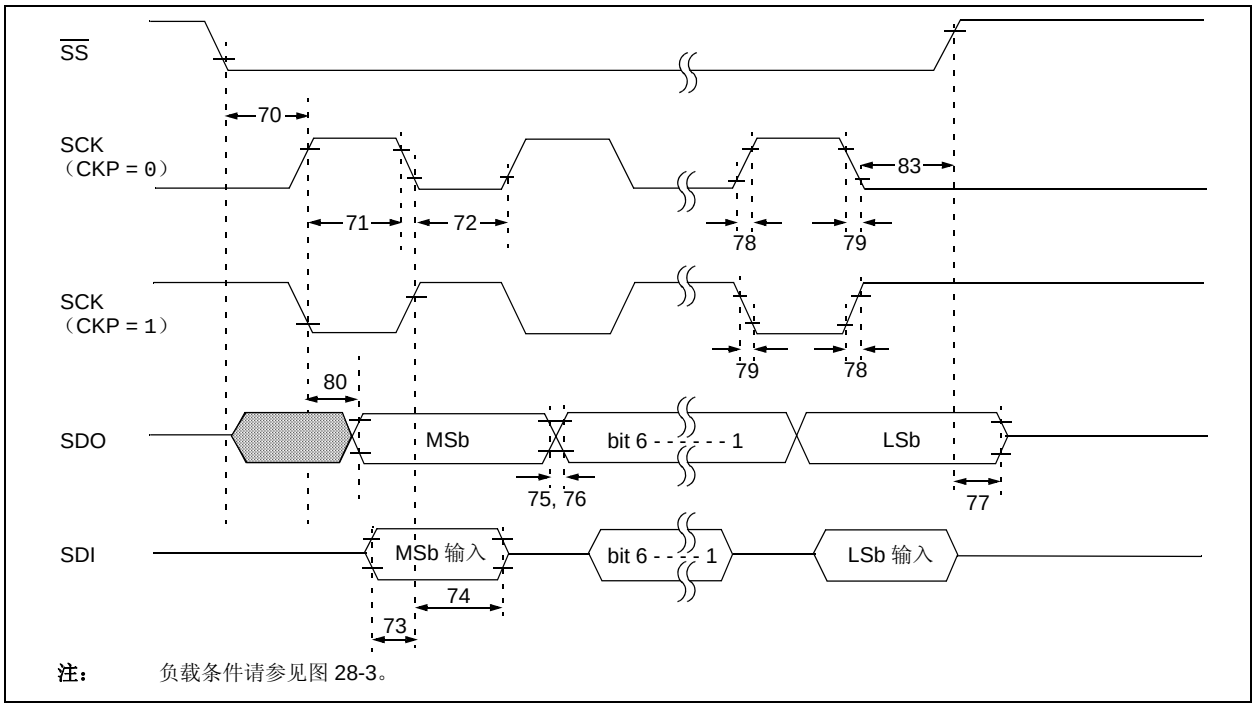
表 28-15： SPI 模式要求示例 （主模式， CKE = 1）

参数编号	符号	特性	最小值	最大值	单位	条件
73	TDIV2SCH，TDIV2SCL	SDI 数据输入到 SCK 边沿的建立时间	20	—	ns	
73A	TB2B	字节 1 的最后一个时钟边沿到字节 2 的第一个时钟边沿之间的时间	1.5 TCY + 40	—	ns	（注 2）
74	TSCH2DIL，TSCL2DIL	SDI 数据输入到 SCK 边沿的保持时间	40	—	ns	
75	TDOR	SDO 数据输出上升时间	—	25	ns	
76	TDOF	SDO 数据输出下降时间	—	25	ns	
78	TSCR	SCK 输出上升时间 （主模式）	—	25	ns	
79	TSCF	SCK 输出下降时间 （主模式）	—	25	ns	
80	TSCH2DOV，TSCL2DOV	在 SCK 边沿之后 SDO 数据输出有效的时间	—	50	ns	
81	TDOV2SCH，TDOV2SCL	SDO 数据输出建立到出现 SCK 边沿的时间	TCY	—	ns	

注 1： 要求使用参数 73A。

2： 仅当使用参数 71A 和 72A 时。

图 28-11：　　SPI 从模式时序示例 （CKE = 0）



注：　负载条件请参见图 28-3。

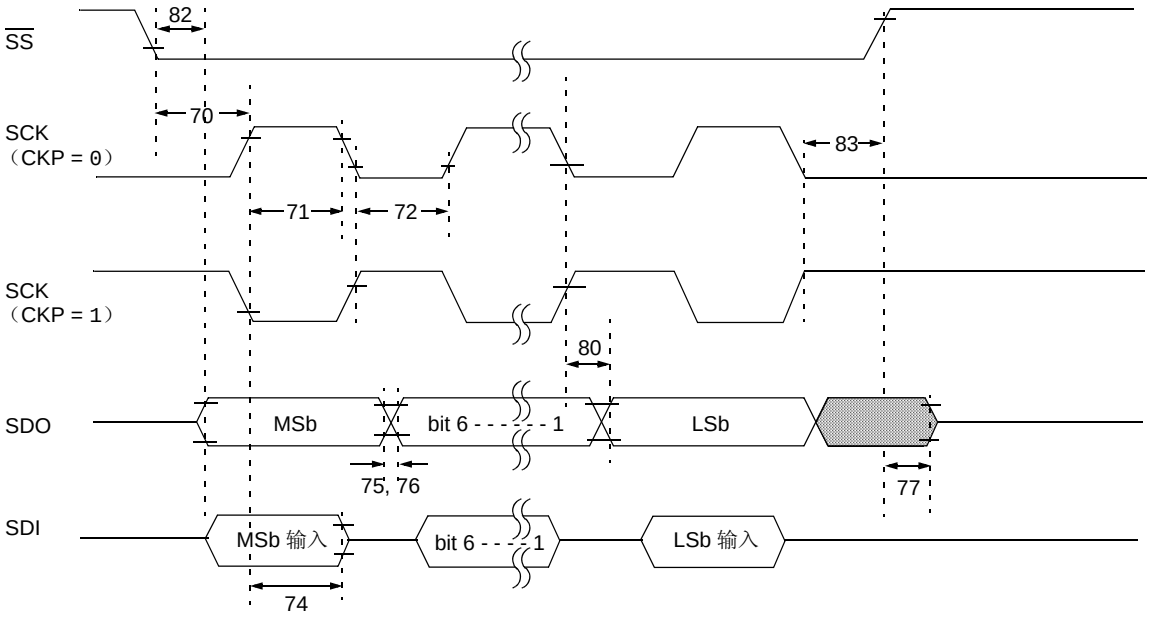
表 28-16：　　SPI 模式要求示例 （从模式时序， CKE = 0）

参数编号	符号	特性		最小值	最大值	单位	条件
70	TssL2scH，TssL2scL	$\overline{SS}$ ↓ 到 SCK ↓ 或 SCK ↑ 输入的时间		3 TCY	—	ns	
70A	TssL2WB	$\overline{SS}$ 到写入 SSPBUF 的时间		3 TCY	—	ns	
71	TscH	SCK 输入高电平时间（从模式）	连续	1.25 TCY + 30	—	ns	
71A			单字节	40	—	ns	（注 1）
72	TscL	SCK 输入低电平时间（从模式）	连续	1.25 TCY + 30	—	ns	
72A			单字节	40	—	ns	（注 1）
73	TdiV2scH，TdiV2scL	SDI 数据输入到 SCK 边沿的建立时间		20	—	ns	
73A	Tb2b	字节 1 的最后一个时钟边沿到字节 2 的第一个时钟边沿之间的时间		1.5 TCY + 40	—	ns	（注 2）
74	TscH2diL，TscL2diL	SDI 数据输入到 SCK 边沿的保持时间		40	—	ns	
75	TdoR	SDO 数据输出上升时间		—	25	ns	
76	TdoF	SDO 数据输出下降时间		—	25	ns	
77	TssH2doZ	$\overline{SS}$ ↑ 到 SDO 输出高阻态的时间		10	50	ns	
78	TscR	SCK 输出上升时间 （主模式）		—	25	ns	
79	TscF	SCK 输出下降时间 （主模式）		—	25	ns	
80	TscH2doV，TscL2doV	在 SCK 边沿之后 SDO 数据输出有效的时间		—	50	ns	
83	TscH2ssH，TscL2ssH	在 SCK 边沿之后出现 $\overline{SS}$ ↑ 的时间		1.5 TCY + 40	—	ns	

注　1： 要求使用参数 73A。

2： 仅当使用参数 71A 和 72A 时。

图 28-12： SPI 从模式时序示例 （CKE = 1）



注： 负载条件请参见图 28-3。

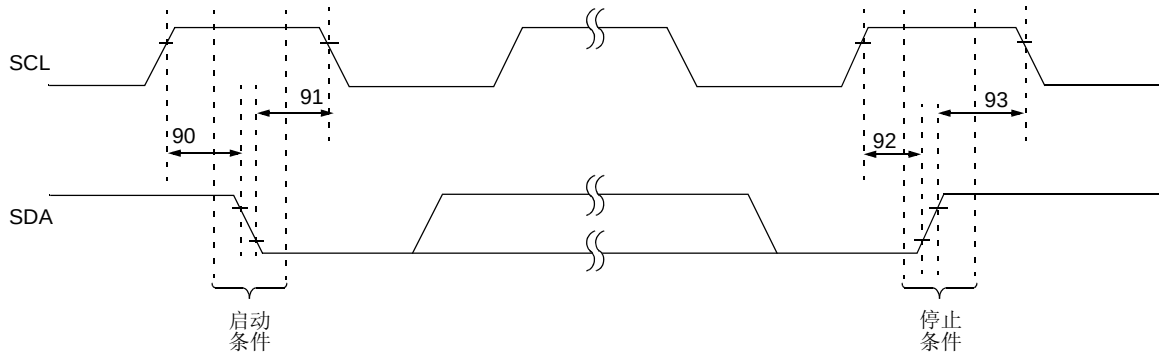
表 28-17： SPI 从模式要求示例 （CKE = 1）

参数编号	符号	特性		最小值	最大值	单位	条件
70	TssL2scH，TssL2scL	$\overline{SS}$ ↓ 到 SCK ↓ 或 SCK ↑ 输入的时间		3 Tcy	—	ns	
70A	TssL2WB	$\overline{SS}$ 到写入 SSPBUF 的时间		3 Tcy	—	ns	
71	TscH	SCK 输入高电平时间（从模式）	连续	1.25 Tcy + 30	—	ns	
71A			单字节	40	—	ns	**（注 1）**
72	TscL	SCK 输入低电平时间（从模式）	连续	1.25 Tcy + 30	—	ns	
72A			单字节	40	—	ns	**（注 1）**
73A	Tb2b	字节 1 的最后一个时钟边沿到字节 2 的第一个时钟边沿之间的时间		1.5 Tcy + 40	—	ns	**（注 2）**
74	TscH2diL，TscL2diL	SDI 数据输入到 SCK 边沿的保持时间		40	—	ns	
75	TdoR	SDO 数据输出上升时间		—	25	ns	
76	TdoF	SDO 数据输出下降时间		—	25	ns	
77	TssH2doZ	$\overline{SS}$ ↑ 到 SDO 输出高阻态的时间		10	50	ns	
78	TscR	SCK 输出上升时间 （主模式）		—	25	ns	
79	TscF	SCK 输出下降时间 （主模式）		—	25	ns	
80	TscH2doV，TscL2doV	在 SCK 边沿之后 SDO 数据输出有效的时间		—	50	ns	
82	TssL2doV	在 $\overline{SS}$ ↓ 边沿之后 SDO 数据输出有效的时间		—	50	ns	
83	TscH2ssH，TscL2ssH	在 SCK 边沿之后出现 $\overline{SS}$ ↑ 的时间		1.5 Tcy + 40	—	ns	

注 1： 要求使用参数 73A。

2： 仅当使用参数 71A 和 72A 时。

图 28-13：　　I²C™ 总线启动位 / 停止位时序

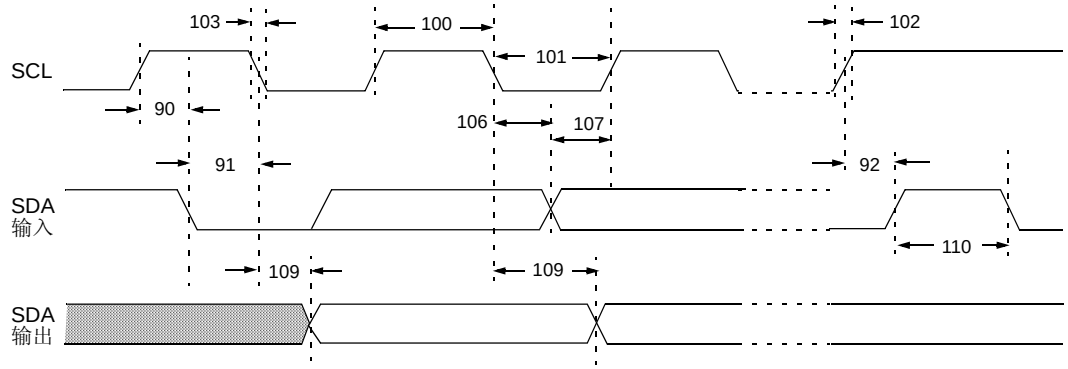


注：　负载条件请参见图 28-3。

表 28-18：　　I²C™ 总线启动位 / 停止位要求 （从模式）

参数编号	符号	特性		最小值	最大值	单位	条件
90	TSU:STA	启动条件建立时间	100 kHz 模式	4700	—	ns	仅与重复启动条件相关
			400 kHz 模式	600	—		
91	THD:STA	启动条件保持时间	100 kHz 模式	4000	—	ns	这个周期后产生第一个时钟脉冲
			400 kHz 模式	600	—		
92	TSU:STO	停止条件建立时间	100 kHz 模式	4700	—	ns	
			400 kHz 模式	600	—		
93	THD:STO	停止条件保持时间	100 kHz 模式	4000	—	ns	
			400 kHz 模式	600	—		

图 28-14：　　I²C™ 总线数据时序



注：　负载条件请参见图 28-3。

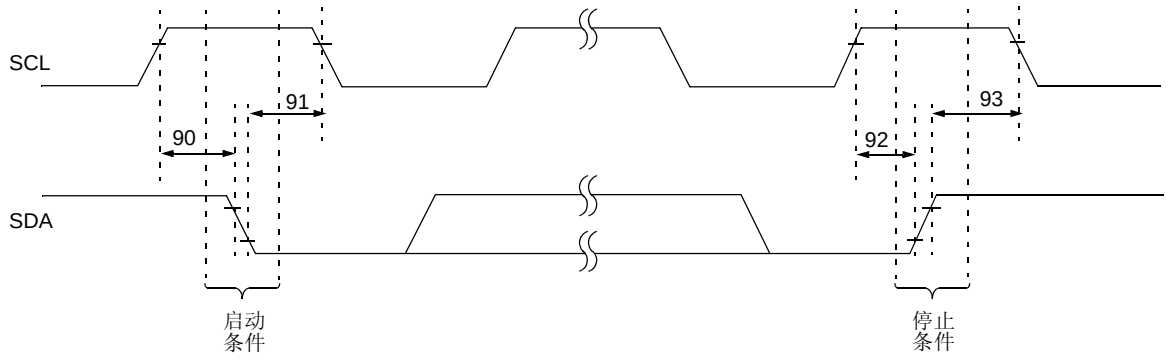
表 28-19：　　I²C™ 总线数据要求 （从模式）

参数编号	符号	特性		最小值	最大值	单位	条件
100	THIGH	时钟高电平时间	100 kHz 模式	4.0	—	µs	
			400 kHz 模式	0.6	—	µs	
			MSSP 模块	1.5 TCY	—		
101	TLOW	时钟低电平时间	100 kHz 模式	4.7	—	µs	
			400 kHz 模式	1.3	—	µs	
			MSSP 模块	1.5 TCY	—		
102	TR	SDA 和 SCL 上升时间	100 kHz 模式	—	1000	ns	
			400 kHz 模式	20 + 0.1 CB	300	ns	CB 值规定在 10 pF 至 400 pF 之间
103	TF	SDA 和 SCL 下降时间	100 kHz 模式	—	300	ns	
			400 kHz 模式	20 + 0.1 CB	300	ns	CB 值规定在 10 pF 至 400 pF 之间
90	TSU:STA	启动条件建立时间	100 kHz 模式	4.7	—	µs	仅与重复启动条件相关
			400 kHz 模式	0.6	—	µs	
91	THD:STA	启动条件保持时间	100 kHz 模式	4.0	—	µs	这个周期后产生第一个时钟脉冲
			400 kHz 模式	0.6	—	µs	
106	THD:DAT	数据输入保持时间	100 kHz 模式	0	—	ns	
			400 kHz 模式	0	0.9	µs	
107	TSU:DAT	数据输入建立时间	100 kHz 模式	250	—	ns	**（注 2）**
			400 kHz 模式	100	—	ns	
92	TSU:STO	停止条件建立时间	100 kHz 模式	4.7	—	µs	
			400 kHz 模式	0.6	—	µs	
109	TAA	从时钟有效到输出有效的时间	100 kHz 模式	—	3500	ns	**（注 1）**
			400 kHz 模式	—	—	ns	
110	TBUF	总线空闲时间	100 kHz 模式	4.7	—	µs	在启动一个新的传输前总线必须保持空闲的时间
			400 kHz 模式	1.3	—	µs	
D102	CB	总线容性负载		—	400	pF	

注　1：　为避免产生意外的启动或停止条件，作为发送器的器件必须提供这个内部最小延时以补偿 SCL 下降沿的未定义区域（最小值 300 ns）。

2：　快速模式的 I²C™ 总线器件也可在标准模式的 I²C 总线系统中使用，但必须满足 TSU:DAT ≥ 250 ns 的要求。如果快速模式器件没有延长 SCL 信号的低电平周期，则必然满足此条件。如果该器件延长了 SCL 信号的低电平周期，它必须将下一个数据位输出到 SDA 线。 SCL 线被释放前，根据标准模式 I²C 总线规范， TR max. + TSU:DAT = 1000 + 250 = 1250 ns。

图 28-15：　　MSSP I²C™ 总线启动位 / 停止位时序波形



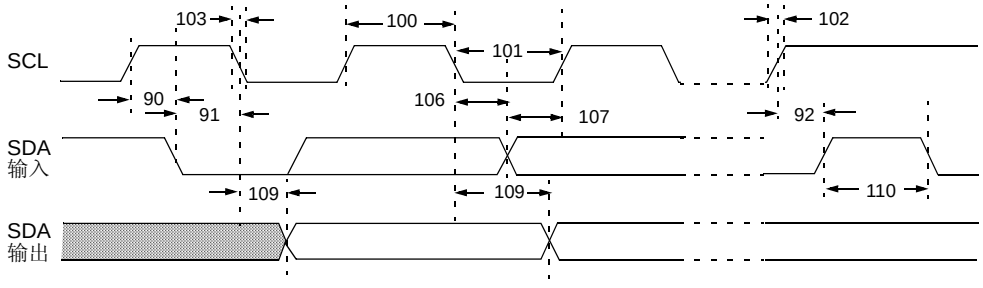
注： 负载条件请参见图 28-3。

表 28-20：　　MSSP I²C™ 总线启动位 / 停止位要求

参数编号	符号	特性		最小值	最大值	单位	条件
90	TSU:STA	启动条件建立时间	100 kHz 模式	2(Tosc)(BRG + 1)	—	ns	仅与重复启动条件相关
			400 kHz 模式	2(Tosc)(BRG + 1)	—		
			1 MHz 模式 (1)	2(Tosc)(BRG + 1)	—		
91	THD:STA	启动条件保持时间	100 kHz 模式	2(Tosc)(BRG + 1)	—	ns	这个周期后产生第一个时钟脉冲
			400 kHz 模式	2(Tosc)(BRG + 1)	—		
			1 MHz 模式 (1)	2(Tosc)(BRG + 1)	—		
92	TSU:STO	停止条件建立时间	100 kHz 模式	2(Tosc)(BRG + 1)	—	ns	
			400 kHz 模式	2(Tosc)(BRG + 1)	—		
			1 MHz 模式 (1)	2(Tosc)(BRG + 1)	—		
93	THD:STO	停止条件保持时间	100 kHz 模式	2(Tosc)(BRG + 1)	—	ns	
			400 kHz 模式	2(Tosc)(BRG + 1)	—		
			1 MHz 模式 (1)	2(Tosc)(BRG + 1)	—		

注　1： 对于所有 I²C™ 引脚，最大引脚电容均为 10 pF。

图 28-16：　　MSSP I²C™ 总线数据时序



注： 负载条件请参见图 28-3。

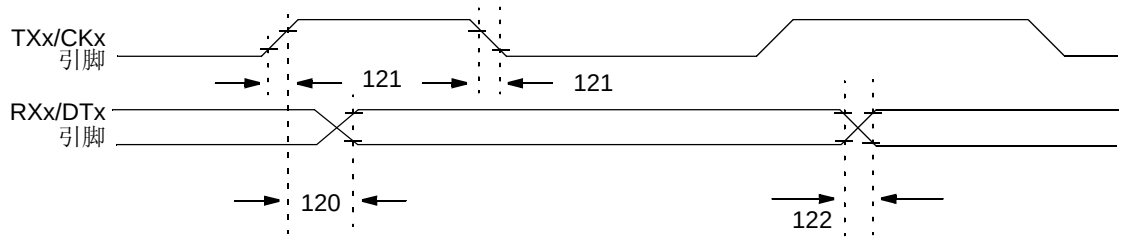
表 28-21： MSSP I²C™ 总线数据要求

参数编号	符号	特性		最小值	最大值	单位	条件
100	THIGH	时钟高电平时间	100 kHz 模式	2(TOSC)(BRG + 1)	—	ms	
			400 kHz 模式	2(TOSC)(BRG + 1)	—	ms	
			1 MHz 模式 (1)	2(TOSC)(BRG + 1)	—	ms	
101	TLOW	时钟低电平时间	100 kHz 模式	2(TOSC)(BRG + 1)	—	ms	
			400 kHz 模式	2(TOSC)(BRG + 1)	—	ms	
			1 MHz 模式 (1)	2(TOSC)(BRG + 1)	—	ms	
102	TR	SDA 和 SCL 上升时间	100 kHz 模式	—	1000	ns	CB 值规定在 10 pF 至 400 pF 之间
			400 kHz 模式	20 + 0.1 CB	300	ns	
			1 MHz 模式 (1)	—	300	ns	
103	TF	SDA 和 SCL 下降时间	100 kHz 模式	—	300	ns	CB 值规定在 10 pF 至 400 pF 之间
			400 kHz 模式	20 + 0.1 CB	300	ns	
			1 MHz 模式 (1)	—	100	ns	
90	TSU:STA	启动条件建立时间	100 kHz 模式	2(TOSC)(BRG + 1)	—	ms	仅与重复启动条件相关
			400 kHz 模式	2(TOSC)(BRG + 1)	—	ms	
			1 MHz 模式 (1)	2(TOSC)(BRG + 1)	—	ms	
91	THD:STA	启动条件保持时间	100 kHz 模式	2(TOSC)(BRG + 1)	—	ms	这个周期后产生第一个时钟脉冲
			400 kHz 模式	2(TOSC)(BRG + 1)	—	ms	
			1 MHz 模式 (1)	2(TOSC)(BRG + 1)	—	ms	
106	THD:DAT	数据输入保持时间	100 kHz 模式	0	—	ns	
			400 kHz 模式	0	0.9	ms	
107	TSU:DAT	数据输入建立时间	100 kHz 模式	250	—	ns	（注 2）
			400 kHz 模式	100	—	ns	
92	TSU:STO	停止条件建立时间	100 kHz 模式	2(TOSC)(BRG + 1)	—	ms	
			400 kHz 模式	2(TOSC)(BRG + 1)	—	ms	
			1 MHz 模式 (1)	2(TOSC)(BRG + 1)	—	ms	
109	TAA	从时钟有效到输出有效的时间	100 kHz 模式	—	3500	ns	
			400 kHz 模式	—	1000	ns	
			1 MHz 模式 (1)	—	—	ns	
110	TBUF	总线空闲时间	100 kHz 模式	4.7	—	ms	在启动一个新的传输前总线必须保持空闲的时间
			400 kHz 模式	1.3	—	ms	
D102	CB	总线容性负载		—	400	pF	

注 1： 对于所有 I²C™ 引脚，最大引脚电容均为 10 pF。

2： 快速模式的 I²C 总线器件也可在标准模式的 I²C 总线系统中使用，但必须满足参数 107 ≥ 250 ns 的要求。如果快速模式器件没有延长 SCL 信号的低电平周期，则必然满足此条件。如果该器件延长了 SCL 信号的低电平周期，它必须将下一个数据位输出到 SDA 线。 SCL 线被释放前，在 100 kHz 模式下，参数 102 + 参数 107 = 1000 + 250 = 1250 ns。

图 28-17： EUSART/AUSART 同步发送 （主 / 从）时序

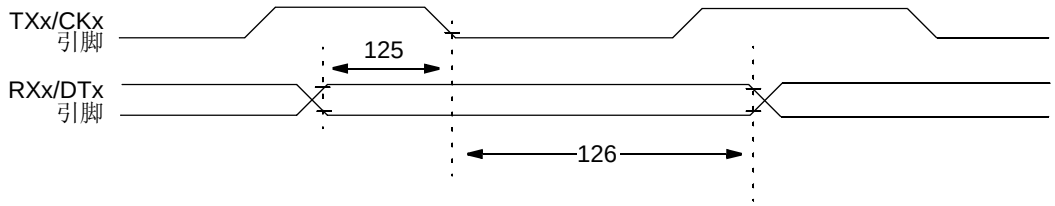


注： 负载条件请参见图 28-3。

表 28-22： EUSART/AUSART 同步发送要求

参数编号	符号	特性	最小值	最大值	单位	条件
120	TCKH2DTV	同步发送 （主和从） 时钟高电平到数据输出有效的时间	—	40	ns	
121	TCKRF	时钟输出上升时间和下降时间 （主模式）	—	20	ns	
122	TDTRF	数据输出上升时间和下降时间	—	20	ns	

图 28-18： EUSART/AUSART 同步接收 （主 / 从）时序



注： 负载条件请参见图 28-3。

表 28-23： EUSART/AUSART 同步接收要求

参数编号	符号	特性	最小值	最大值	单位	条件
125	TDTV2CKL	同步接收 （主和从） 在 CKx ↓ 之前数据的保持时间 （DTx 保持时间）	10	—	ns	
126	TCKL2DTL	在 CKx ↓ 之后数据的保持时间 （DTx 保持时间）	15	—	ns	

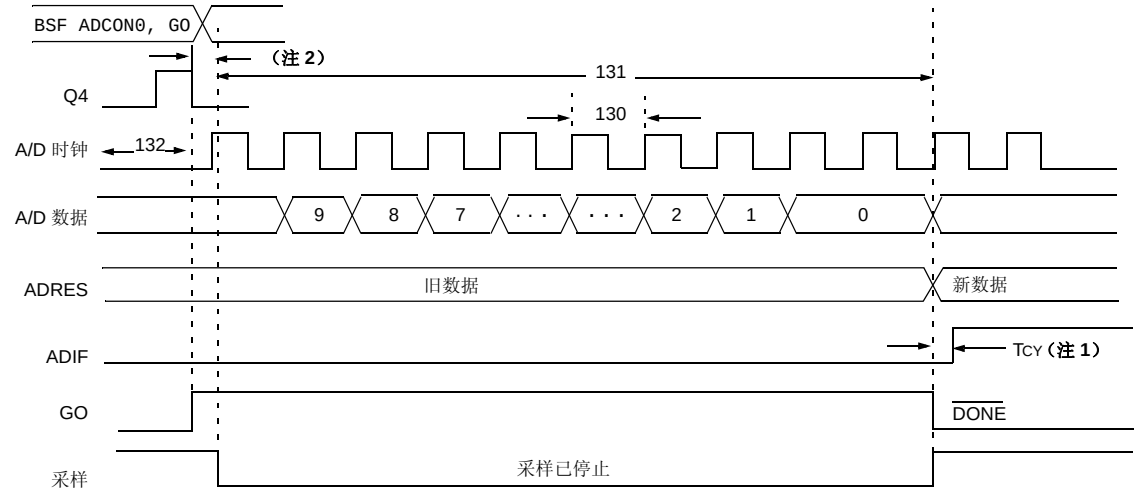
表 28-24： A/D 转换器特性：PIC18F87J90 系列 （工业级）

参数编号	符号	特性	最小值	典型值	最大值	单位	条件
A01	NR	分辨率	—	—	10	位	
A03	EIL	积分线性误差	—	—	<±1	LSb	ΔVREF ≥ 3.0V
A04	EDL	微分线性误差	—	—	<±1	LSb	ΔVREF ≥ 3.0V
A06	EOFF	失调误差	—	—	<±3	LSb	ΔVREF ≥ 3.0V
A07	EGN	增益误差	—	—	<±3	LSb	ΔVREF ≥ 3.0V
A10	—	单调性	保证 (1)			—	VSS ≤ VAIN ≤ VREF
A20	ΔVREF	参考电压范围 （VREFH – VREFL）	2.0 2.0	— —	— —	V V	VDD < 3.0V VDD > 3.0V
A21	VREFH	参考电压高电压	VSS + ΔVREF	—	VDD	V	
A22	VREFL	参考电压低电压	VSS – 0.3V	—	VDD – 3.0V	V	
A25	VAIN	模拟输入电压	VREFL	—	VREFH	V	
A30	ZAIN	模拟信号源的推荐阻抗	—	—	2.5	kΩ	
A50	IREF	VREF 输入电流 (2)	— —	— —	5 150	μA μA	在采集 VAIN 期间。 在 A/D 转换周期期间。

注 1： A/D 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： VREFH 电流来自选择作为 VREFH 源的 RA3/AN3/VREF+ 引脚或 VDD。
VREFL 电流来自选择作为 VREFL 源的 RA2/AN2/VREF- 引脚或 VSS。

图 28-19： A/D 转换时序



注 1： 如果将 A/D 时钟源选择为 RC，在 A/D 时钟启动前要加上一个 TCY 时间，用以执行 SLEEP 指令。

2： 这是最小的 RC 延时 （典型值为 100 ns），在这段时间保持电容与模拟输入端断开。

表 28-25： A/D 转换要求

参数编号	符号	特性	最小值	最大值	单位	条件
130	TAD	A/D 时钟周期	0.7	25.0(1)	µs	基于 TOSC， VREF ≥ 3.0V
			—	1	µs	A/D RC 模式
131	TCNV	转换时间 （不包括采集时间） (2)	11	12	TAD	
132	TACQ	采集时间 (3)	1.4	—	µs	-40°C 至 +85°C
135	TSWC	转换 → 采样的切换时间	—	（注 4）		
136	TDIS	电容放电时间	0.2	—	µs	

注 1： A/D 时钟周期取决于器件频率和 TAD 时钟分频比。

2： ADRES 寄存器可在下一个 TCY 周期被读取。

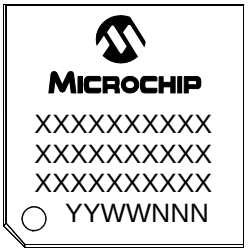
3： 转换完成后当电压满量程变化时 （VDD 至 VSS 或 VSS 至 VDD），保持电容采集一个 “ 新 ” 输入电压所需的时间。在输入通道上的信号源阻抗 （*RS*）为 50Ω。

4： 在器件时钟的下一个周期。

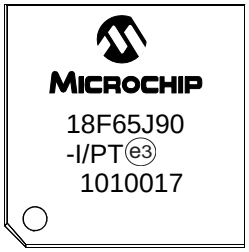
29.0 封装信息

29.1 封装标识信息

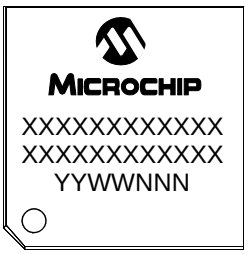
64 引脚 TQFP



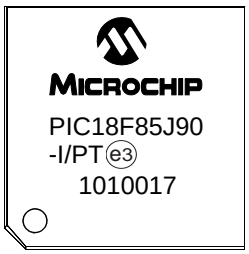
示例



80 引脚 TQFP



示例



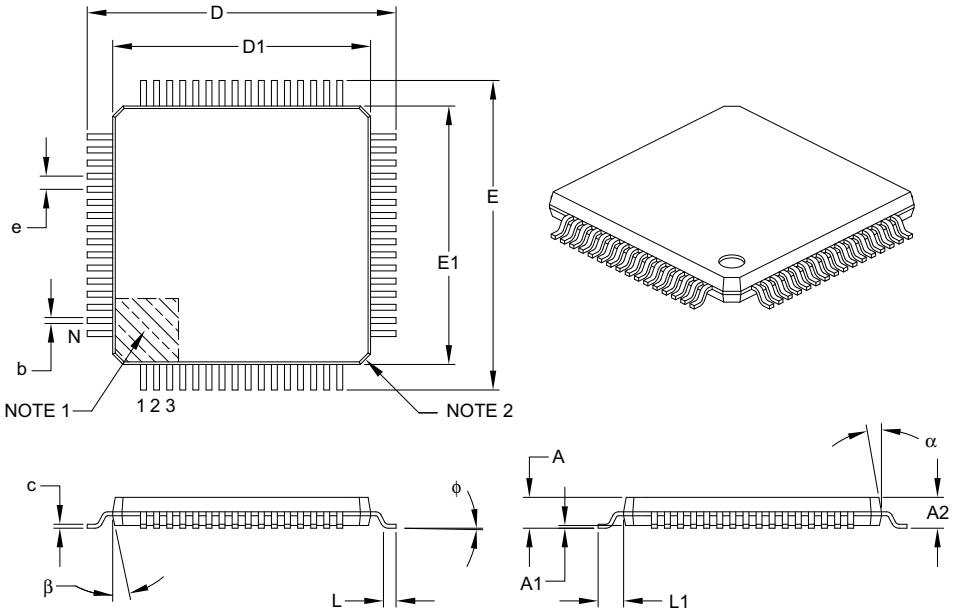
图注：	XX...X	客户信息
	Y	年份代码 （日历年的最后一位数字）
	YY	年份代码 （日历年的最后两位数字）
	WW	星期代码 （一月一日的星期代码为 “01”）
	NNN	以字母数字排序的追踪代码
	(e3)	雾锡 （Matte Tin， Sn）的 JEDEC 无铅标志
	*	本封装为无铅封装。 JEDEC 无铅标志 （(e3)）标示于此种封装的外包装上。
注：	Microchip 元器件编号如果无法在同一行内完整标注，将换行标出，因此会限制表示客户信息的字符数。	

29.2 封装详细信息

以下部分将介绍各种封装的技术细节。

64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		64	
Lead Pitch	e		0.50 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	0.17	0.22	0.27
Mold Draft Angle Top	α	11°	12°	13°
Mold Draft Angle Bottom	β	11°	12°	13°

Notes:

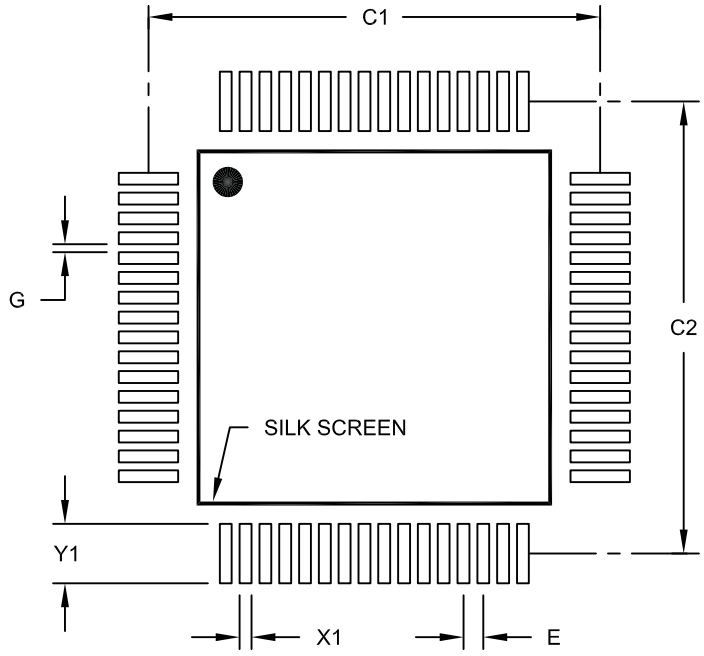
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-085B

64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		0.50 BSC	
Contact Pad Spacing	C1		11.40	
Contact Pad Spacing	C2		11.40	
Contact Pad Width (X64)	X1			0.30
Contact Pad Length (X64)	Y1			1.50
Distance Between Pads	G	0.20		

Notes:

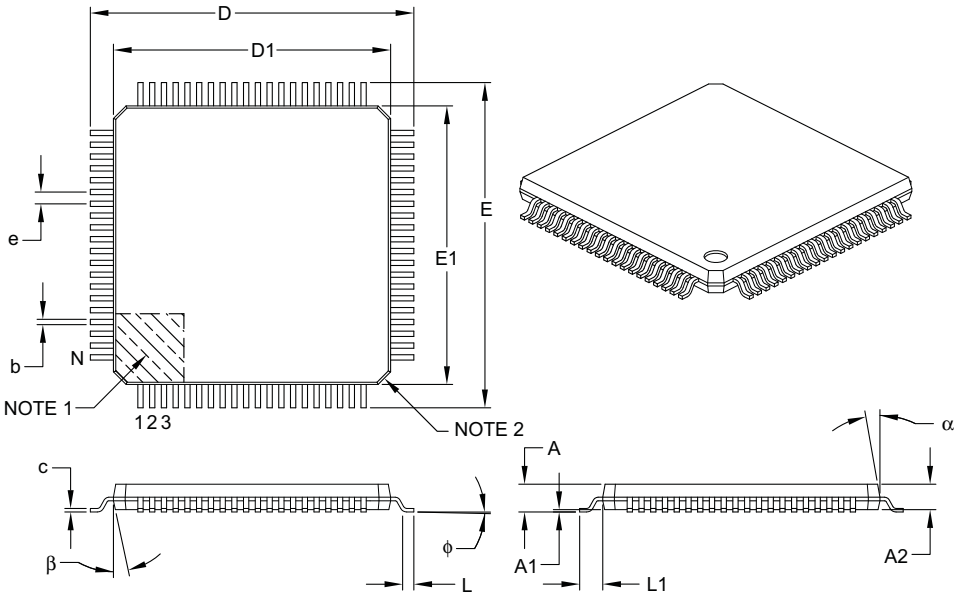
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2085A

80引脚塑封薄型正方扁平封装（**PT**）——主体**12x12x1 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		80	
Lead Pitch	e		0.50 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		14.00 BSC	
Overall Length	D		14.00 BSC	
Molded Package Width	E1		12.00 BSC	
Molded Package Length	D1		12.00 BSC	
Lead Thickness	c	0.09	–	0.20
Lead Width	b	0.17	0.22	0.27
Mold Draft Angle Top	α	11°	12°	13°
Mold Draft Angle Bottom	β	11°	12°	13°

Notes:

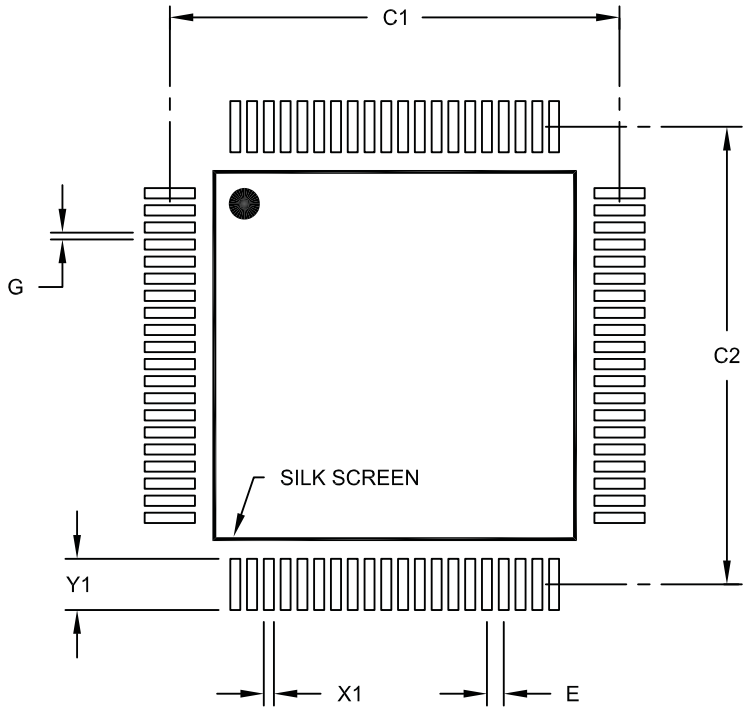
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Chamfers at corners are optional; size may vary.
3. Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-092B

80引脚塑封薄型正方扁平封装（**PT**）——主体**12x12x1 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		0.50 BSC	
Contact Pad Spacing	C1		13.40	
Contact Pad Spacing	C2		13.40	
Contact Pad Width (X80)	X1			0.30
Contact Pad Length (X80)	Y1			1.50
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. C04-2092A

注：

附录 A： 版本历史

版本 A （2008 年 10 月）

PIC18F87J90 系列器件的原始数据手册。

版本 B （2008 年 12 月）

更改了**第 28.2 节 “ 直流特性：PIC18F87J90 系列的掉电电流和供电电流（工业级）”**，去除了**第 27.5 节 “ 直流特性：RTCC 掉电电流 （I_{PD}） ”**。

版本 C （2009 年 2 月）

在**第 25.0 节 “CPU 的特殊功能 ”** 中添加了寄存器 CONFIG3L。另外做了少许其他修正。

版本 D （2010 年 1 月）

在**第 28.0 节 “ 电气特性 ”** 中添加了 60°C IPD 规范。删除了 “ 初稿 ” 条件标志。通篇做了少量编辑。

附录 B： 从 PIC18F85J90 到 PIC18F87J90 的移植

PIC18F87J90 和 PIC18F85J90 系列器件的功能和特性非常相似。可以将代码从18F85J90移植到PIC18F87J90而无需进行许多更改。表 B-1 列出了这两种器件系列之间的差异。

表 B-1： PIC18F87J90 和 PIC18F85J90 系列之间的显著差异

特性	PIC18F87J90 系列	18F85J90 系列
最高工作频率	48 MHz	40 MHz
最大程序存储器	128 KB	32 KB
数据存储器	3,923 字节	2,048 字节
程序存储器耐用性	10,000 次写 / 擦除 （最小值）	1,000 次写 / 擦除 （最小值）
闪存的单字写入	有	无
振荡器选项	PLL 可以与 INTOSC 一起使用	PLL 不能与 INTOSC 一起使用
CTMU	有	无
RTCC	有	无
Timer1 振荡器选项	Timer1 的低功耗振荡器选项	无
TICKI 时钟	T1CKI 可以在不使能 Timer1 振荡器的情况下用作时钟	无

注：

索引

A

B

L

M

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 的变更通知客户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。在“支持”（Support）下，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://microchip.com/support 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: PIC18F87J90 系列　　文献编号: DS39933D_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。

器件编号	X	/XX	XXX
器件	温度范围	封装	模式

器件 (1,2)	PIC18F66J90 和 PIC18F66J90T PIC18F67J90 和 PIC18F67J90T PIC18F86J90 和 PIC18F86J90T PIC18F87J90 和 PIC18F87J90T
温度范围	I = -40°C 至 +85°C（工业级）
封装	PT = TQFP（薄型正方扁平封装）
模式	QTP、 SQTP、代码或特殊要求（其他情况空白）

示例：

a) PIC18F87J90-I/PT 301 = 工业级温度， TQFP 封装， QTP 模式 #301。

b) PIC18F87J90T-I/PT = 卷带式，工业级温度，TQFP 封装。

注 **1**：F = 标准电压范围
2：T = 卷带式

全球销售及服务网点

美洲

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

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

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

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

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

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

底特律 Detroit
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

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

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

圣克拉拉 Santa Clara
Santa Clara, CA
Tel: 1-408-961-6444
Fax: 1-408-961-6445

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: 86-10-8569-7000
Fax: 86-10-8528-2104

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

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

中国 – 杭州
Tel: 86-571-2819-3187
Fax: 86-571-2819-3189

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

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

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

亚太地区

台湾地区 – 高雄
Tel: 886-7-536-4818
Fax: 886-7-330-9305

台湾地区 – 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

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

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

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

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

印度 India - Pune
Tel: 91-20-2566-1512
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

08/02/11