PIC18F/LF1XK50
数据手册

采用 nanoWatt XLP 技术的
20 引脚 USB 闪存单片机

请注意以下有关 Microchip 器件代码保护功能的要点：

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

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

ISBN：978-1-60932-853-5

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

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

PIC18F/LF1XK50

采用 nanoWatt XLP 技术的 20 引脚 USB 闪存单片机

通用串行总线特性：

- 符合 USB V2.0 规范的 SIE
- 全速 （12 Mb/s）和低速 （1.5 Mb/s）
- 支持控制、中断、同步和批量传输
- 支持最多 16 个端点 （双向为 8 个）
- 用于 USB 的 256 字节双通道快速操作 RAM
- D+/D- 上的输入电平变化中断，用于检测到 USB 主机的物理连接

高性能 RISC CPU：

- 优化的 C 编译器架构：
  - 为优化可重入代码而设计的可选的扩展指令集
  - 256 字节数据 EEPROM
  - 最多可寻址 16 KB 的线性程序存储器
  - 最多可寻址 768 字节的线性数据存储器
- 中断优先级
- 8 x 8 单周期硬件乘法器

灵活的振荡器结构：

- CPU 分频器使内核的运行速度比 USB 外设慢
- 16 MHz 内部振荡器模块：
  - 频率可由软件选择为从 31 kHz 到 16 MHz
  - 当与 PLL 结合使用时可提供全部的时钟频率范围，从 31 kHz 到 32 MHz
  - 用户可对该电路进行调节以补偿频率漂移
- 4 种晶振模式，频率最高为 48 MHz
- 外部时钟模式，频率最高为 48 MHz
- 4 倍频锁相环
- 辅助振荡器使用 Timer1 （工作频率为 32 kHz）
- 故障保护时钟监视器：
  - 当主振荡器或辅助振荡器停止时可使器件安全断电
- 双速振荡器启动

单片机特性：

- 在 5.5V 条件下全速工作 ——PIC18F1XK50
- 在 1.8V-3.6V 条件下工作 ——PIC18LF1XK50
- 在软件控制下自编程
- 可编程欠压复位 （Brown-out Reset， BOR）
  - 带软件使能选项
- 扩展型看门狗定时器 （Watchdog Timer， WDT）：
  - 可编程周期从 4 ms 到 131s
- 通过两个引脚依靠 3V 电源进行在线串行编程 （In-Circuit Serial Programming™， ICSP™）

采用 nanoWatt XLP 技术实现 PIC18LF1XK50 的极低功耗管理：

- 休眠模式：24 nA
- 看门狗定时器：450 nA
- Timer1 振荡器：790 nA @ 32 kHz

模拟特性：

- 模数转换器 （Analog-to-Digital Converter， ADC）模块：
  - 10 位分辨率， 9 路外部通道
  - 能够自动采集
  - 可在休眠模式下转换
  - 内部 1.024V 固定参考电压 （Fixed Voltage Reference， FVR）通道
  - 独立的输入多路选择
- 两个模拟比较器
  - 轨到轨工作
  - 独立的输入多路选择
- 参考电压模块：
  - 可编程参考电压 （V_{DD} 的 %）， 16 步
  - 两个从 V_{REF} 引脚输入产生的参考电压范围，每个电压范围包含 16 个可选电压
  - 可编程固定参考电压 （Fixed Voltage Reference， FVR）， 3 个可选电压
- 片上 3.2V LDO 稳压器 —— （PIC18F1XK50）

外设特点：

- 14 个 I/O 引脚和一个只能用作输入的引脚：
  - 高灌 / 拉电流：25 mA/25 mA
  - 7 个可编程弱上拉
  - 7 个可编程电平变化中断引脚
  - 3 个可编程外部中断
  - 可编程压摆率
- 增强型捕捉 / 比较 /PWM （Enhanced Capture/ Compare/PWM， ECCP）模块：
  - 1、 2、 3 或 4 路 PWM 输出
  - 可选择的极性
  - 可编程的死区
  - 自动关闭和自动重启
- 主同步串行口 （Master Synchronous Serial Port， MSSP）模块
  - 支持 3 线 SPI （所有 4 种模式）
  - I^2C™ 主模式和从模式 （从模式地址掩码）
- 增强型通用同步异步收发器 （Enhanced Universal Synchronous Asynchronous Receiver Transmitter， EUSART）模块：
  - 支持 RS-485、 RS-232 和 LIN 2.0
  - 使用内部振荡器的 RS-232 工作模式
  - 自动波特率检测
  - 接收到间隔字符时自动唤醒
- SR 锁存器模式

器件	程序存储器		数据存储器		I/O(1)	10 位 A/D（通道）(2)	ECCP（PWM）	MSSP		EUSART	比较器	8/16 位定时器	USB
	闪存（字节）	单字指令数	SRAM（字节）	EEPROM（字节）				SPI	主 I^2C™				
PIC18F13K50/ PIC18LF13K50	8K	4096	512(3)	256	15	11	1	有	有	1	2	1/3	有
PIC18F14K50/ PIC18LF14K50	16K	8192	768(3)	256	15	11	1	有	有	1	2	1/3	有

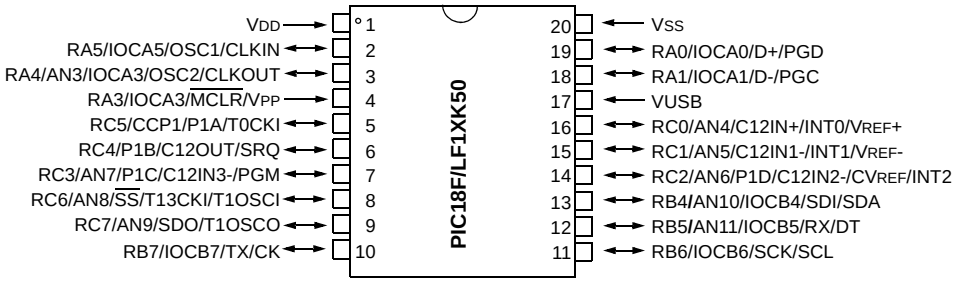
注 **1：** 1 个引脚为仅能用作输入功能的引脚。

2： 通道数包括内部的固定参考电压 （FVR）和可编程参考电压 （CVREF）通道。

3： 包括 USB 模块使用的双端口 RAM，该 RAM 与数据存储器共用。

引脚图

20 引脚 PDIP、SSOP 和 SOIC（300 MIL）



引脚图

20 引脚 QFN（5x5）

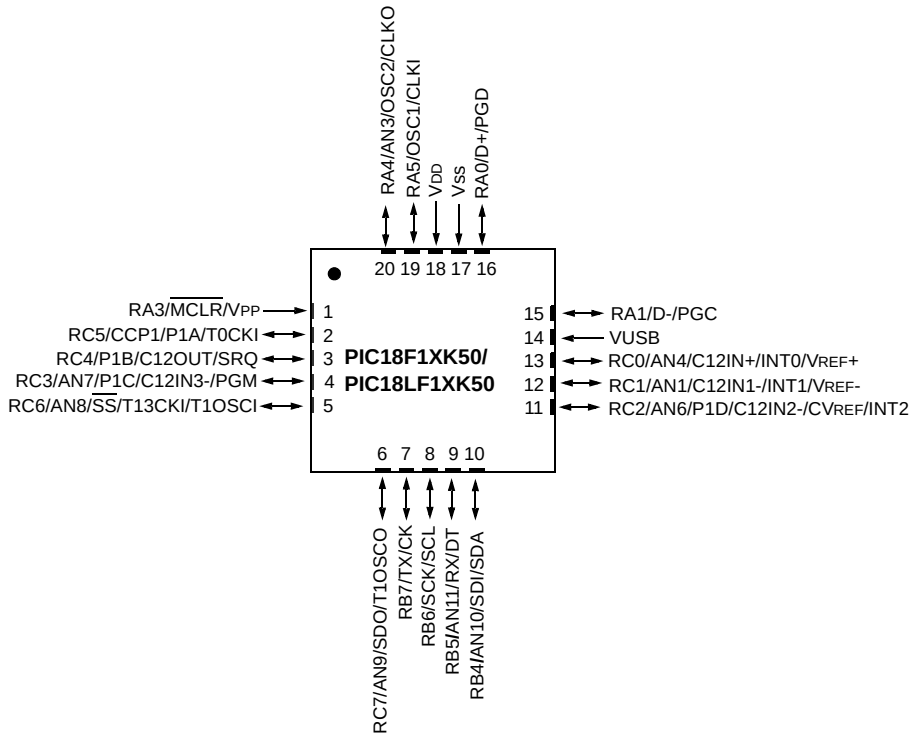


表 1： PIC18F/LF1XK50 引脚汇总

引脚	I/O	模拟	比较器	参考电压	ECCP	EUSART	MSSP	定时器	中断	上拉	USB	基本功能
19	RA0								IOCA0		D+	PGD
18	RA1								IOCA1		D-	PGC
4	RA3(1)								IOCA3	有		$\overline{\text{MCLR}}$/VPP
3	RA4	AN3							IOCA4	有		OSC2/CLKOUT
2	RA5								IOCA5	有		OSC1/CLKIN
13	RB4	AN10					SDI/SDA		IOCB4	有		
12	RB5	AN11				RX/DT			IOCB5	有		
11	RB6						SCL/SCK		IOCB6	有		
10	RB7					TX/CK			IOCB7	有		
16	RC0	AN4	C12IN+	VREF+					INT0			
15	RC1	AN5	C12IN1-	VREF-					INT1			
14	RC2	AN6	C12IN2-	CVREF	P1D				INT2			
7	RC3	AN7	C12IN3-		P1C							PGM
6	RC4		C12OUT		P1B							SRQ
5	RC5				CCP1/P1A			T0CKI				
8	RC6	AN8					$\overline{\text{SS}}$	T13CKI/T1OSCI				
9	RC7	AN9					SDO	T1OSCO				
17											VUSB	
1												VDD
20												VSS

注 1： 仅用作输入。

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。

注：

1.0 器件概述

本文档涉及以下器件的具体信息：

- PIC18F13K50
- PIC18F14K50
- PIC18LF13K50
- PIC18LF14K50

本系列具备所有 PIC18 单片机固有的优点，即以实惠的价格提供出色的计算性能，以及非常耐用的闪存程序存储器。除了这些特性之外，PIC18F/LF1XK50 系列还增强了器件设计，使得该系列单片机成为许多高性能但对功耗有严格要求的应用的明智选择。

1.1 新的内核特性

1.1.1 nanoWatt XLP 技术

PIC18F/LF1XK50 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 或内部振荡器模块作为单片机时钟源，可使代码执行时的功耗降低大约 90%。
- **多种空闲模式：**单片机还可在其 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该功耗管理模式，允许用户将节能的理念融入到他们的应用软件设计中。
- **关键模块低功耗 ：**Timer1 和看门狗定时器模块的功耗需求可降至最小。具体数值请参见**第 27.0 节 “ 电气规范 ”**。

1.1.2 多个振荡器选项和特性

PIC18F/LF1XK50 系列的所有器件可提供 10 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：

- 4 种晶振模式，使用晶振或陶瓷谐振器
- 外部时钟模式，提供使用两个引脚（振荡器输入引脚和四分频时钟输出引脚）或一个引脚（振荡器输入引脚，四分频时钟输出引脚重新分配为通用 I/O 引脚）的选项
- 外部 RC 振荡器模式，具有与外部时钟模式相同的引脚选项
- 内部振荡器模块包含一个16 MHz HFINTOSC振荡器和一个 31 kHz LFINTOSC 振荡器，它们共同提供 8 个可供用户选择的时钟频率，范围从 31 kHz 到 16 MHz。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。
- 一个锁相环 （Phase Lock Loop， PLL）倍频器，可在高速晶振和内部振荡器模式下使用，可使时钟速度最高达到 48 MHz。 PLL 和内部振荡器配合使用，可以向用户提供频率范围从 31 kHz 到 32 MHz 的时钟速度以供选择，而且不需要使用外部晶振或时钟电路。

除了可被用作时钟源外，内部振荡器模块还提供了一个稳定的参考源，增加了以下功能以使器件更安全地工作：

- **故障保护时钟监视器：**该部件持续监视主时钟源，将其与 LFINTOSC 提供的参考信号作比较。如果时钟发生了故障，单片机会将时钟源切换到内部振荡器模块，使器件可继续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。

1.2 其他特殊功能

- **存储器耐用性：**程序存储器和数据 EEPROM 的闪存单元经评测，可以耐受数千次擦写，程序存储器最高可达 1,000 次， EEPROM 最高可达 100,000 次。在不刷新的情况下，数据保存时间保守地估计在 40 年以上。
- **自编程能力：**这些器件能在内部软件控制下写入各自的程序存储空间。通过使用受代码保护的引导区中的自举程序，可创建能在现场进行自我更新的应用程序。
- **扩展指令集：**PIC18F/LF1XK50 系列在 PIC18 指令集的基础上进行了可选择的扩展，添加了 8 条新指令和一个变址寻址模式。此扩展是为优化可重入应用程序代码而特别设计的，这些代码原来是使用高级语言 （如 C 语言）开发的。
- **增强型 CCP 模块：**在 PWM 模式下，该模块提供用于控制半桥或全桥驱动器的 1、 2 或 4 路调制输出。其他特性包括：
  - 自动关闭，用于禁止中断或其他选定条件下的 PWM 输出
  - 自动重启，用于在条件清除后重新激活输出
  - 输出控制，用于有选择地使能 4 路输出中的一个或多个以提供 PWM 信号
- **增强型可寻址 USART ：**该串行通信模块可进行标准的 RS-232 通信并支持 LIN 总线协议。其他增强功能包括自动波特率检测和分辨率更高的16位波特率发生器。
- **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。
- **扩展型看门狗定时器（WDT）：**该增强型版本加入了一个 16 位后分频器，可以延长超时范围，在这个范围内工作电压和温度的变化不会对器件工作的稳定性造成影响。超时周期请参见**第 27.0 节“ 电气规范 ”**。

1.3 系列中各产品的详细说明

PIC18F/LF1XK50 系列器件以 20 引脚封装形式提供。图 1-1 为这两类器件的框图。

这两类器件在以下方面存在差异：

1. 闪存程序存储器：
   - PIC18F13K50/PIC18LF13K50 为 8 KB
   - PIC18F14K50/PIC18LF14K50 为 16 KB
2. PIC18F13K50 和 PIC18F14K50 具有片上 3.2V LDO 稳压器。

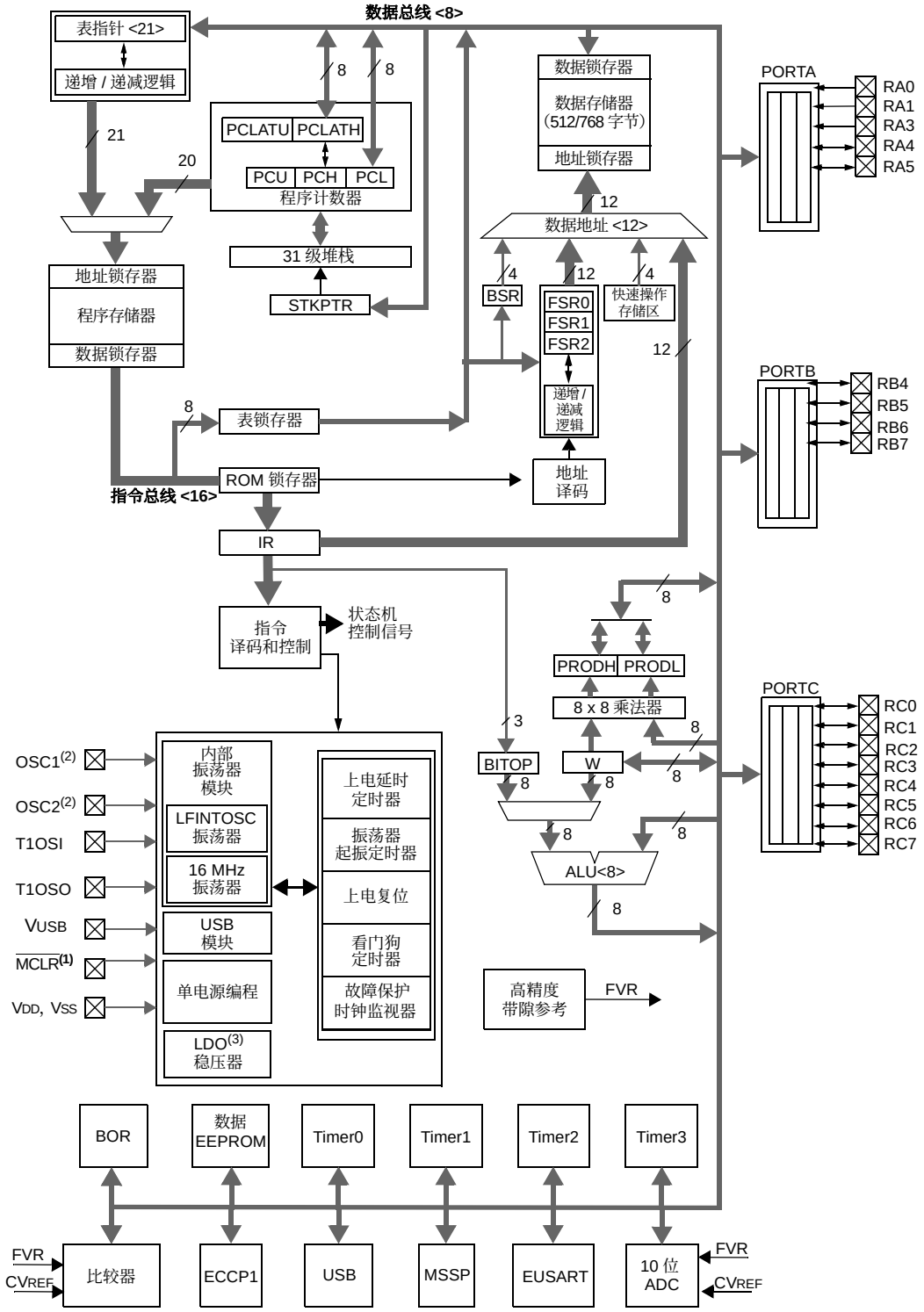
本系列器件的其他功能都是相同的。表 1-1 汇总了这些功能。

表1给出了所有器件的引脚排列，表1-2给出了I/O说明。

表 1-1： PIC18F/LF1XK50 器件特性 （20 引脚器件）

特性	PIC18F13K50	PIC18LF13K50	PIC18F14K50	PIC18LF14K50
LDO 稳压器	有	无	有	无
程序存储器 （字节）	8K		16K	
程序存储器 （指令）	4096		8192	
数据存储器 （字节）	512		768	
工作频率	DC – 48 MHz			
中断源	30			
I/O 端口	端口 A、 B 和 C			
定时器	4			
增强型捕捉 / 比较 /PWM 模块	1			
串行通信	MSSP、增强型 USART 和 USB			
10 位模数转换模块	9 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	20 引脚 PDIP、 SSOP、 SOIC （300 mil）和 QFN （5x5）			

图 1-1：　　PIC18F/LF1XK50 框图



注　**1：** 只有在禁止 $\overline{MCLR}$ 功能时， RA3 才可用。

2： OSC1/CLKIN 和 OSC2/CLKOUT 只有在选择振荡器模式时才可用，并且这些引脚不被用作数字 I/O 引脚。更多信息，请参见**第 2.0 节 “ 振荡器模块 ”**。

3： 仅限 PIC18F13K50/PIC18F14K50。

表 1-2： PIC18F/LF1XK50 引脚 I/O 说明

引脚名称	引脚编号	引脚类型	缓冲器类型	说明
RA0/D+/PGD	19			
RA0		I	TTL	数字输入
D+		I/O	XCVR	USB 差分正信号线 （输入 / 输出）
PGD		I/O	ST	ICSP™ 编程数据引脚
RA1/D-/PGC	18			
RA1		I	TTL	数字输入
D-		I/O	XCVR	USB 差分负信号线 （输入 / 输出）
PGC		I/O	ST	ICSP™ 编程时钟引脚
RA3/$\overline{\text{MCLR}}$/V_{PP}	4			主复位 （输入）或编程电压 （输入）
RA3		I	ST	数字输入
$\overline{\text{MCLR}}$		I	ST	低电平有效的主复位，带内部上拉
V_{PP}		P	—	高压编程输入
RA4/AN3/OSC2/CLKOUT	3			
RA4		I/O	TTL	数字 I/O
AN3		I	模拟	ADC 通道 3
OSC2		O	XTAL	振荡器晶振输出。在晶振模式下，该引脚与晶振或谐振器相连
CLKOUT		O	CMOS	在 RC 模式下， OSC2 引脚输出 CLKOUT 振荡信号，该信号是 OSC1 引脚上振荡信号的 4 分频，该频率等于指令周期的倒数
RA5/OSC1/CLKIN	2			
RA5		I/O	TTL	数字 I/O
OSC1		I	XTAL	振荡器晶振输入或外部时钟输入
				配置为 RC 模式时为 ST 缓冲器输入；否则为模拟输入
CLKIN		I	CMOS	外部时钟源输入。总是与引脚功能 OSC1 相关联 （见相关的 OSC1/CLKIN、OSC2 和 CLKOUT 引脚信息）
RB4/AN10/SDI/SDA	13			
RB4		I/O	TTL	数字 I/O
AN10		I	模拟	ADC 通道 10
SDI		I	ST	SPI 数据输入
SDA		I/O	ST	I^2C™ 数据 I/O
RB5/AN11/RX/DT	12			
RB5		I/O	TLL	数字 I/O
AN11		I	模拟	ADC 通道 11
RX		I	ST	EUSART 异步接收
DT		I/O	ST	EUSART 同步数据 （见相关的 RX/TX 引脚信息）
RB6/SCK/SCI	11			
RB6		I/O	TLL	数字 I/O
SCK		I/O	ST	SPI 模式的同步串行时钟输入 / 输出
SCI		I/O	ST	I^2C™ 模式的同步串行时钟输入 / 输出
RB7/TX/CK	10			
RB7		I/O	TLL	数字 I/O
TX		O	CMOS	EUSART 异步发送
CK		I/O	ST	EUSART 同步时钟 （见相关的 RX/DT 引脚信息）

图注： TTL = TTL 兼容输入
ST = 施密特触发器输入
O = 输出
XTAL = 晶振
CMOS = CMOS 兼容输入或输出
I = 输入
P = 电源
XCVR = USB 差分收发器

表 1-2： PIC18F/LF1XK50 引脚 I/O 说明 （续）

引脚名称	引脚编号	引脚类型	缓冲器类型	说明
RC0/AN4/C12IN+/INT0/VREF+	16			
RC0		I/O	ST	数字 I/O
AN4		I	模拟	ADC 通道 4
C12IN+		I	模拟	比较器 C1 和 C2 的同相输入
INT0		I	ST	外部中断 0
VREF+		I	模拟	比较器参考电压 （高）输入
RC1/AN5/C12IN-/INT1/VREF-	15			
RC1		I/O	ST	数字 I/O
AN5		I	模拟	ADC 通道 5
C12IN-		I	模拟	比较器 C1 和 C2 的反相输入
INT1		I	ST	外部中断 0
VREF-		I	模拟	比较器参考电压 （低）输入
RC2/AN6/P1D/C12IN2-/CVREF/INT2	14			
RC2		I/O	ST	数字 I/O
AN6		I	模拟	ADC 通道 6
P1D		O	CMOS	增强型 CCP1 PWM 输出
C12IN2-		I	模拟	比较器 C1 和 C2 的反相输入
CVREF		O	模拟	比较器参考电压输出
INT2		I	ST	外部中断 0
RC3/AN7/P1C/C12IN3-/PGM	7			
RC3		I/O	ST	数字 I/O
AN7		I	模拟	ADC 通道 7
P1C		O	CMOS	增强型 CCP1 PWM 输出
C12IN3-		I	模拟	比较器 C1 和 C2 的反相输入
PGM		I/O	ST	低压 ICSP 编程使能引脚
RC4/P1B/C12OUT/SRQ	6			
RC4		I/O	ST	数字 I/O
P1B		O	CMOS	增强型 CCP1 PWM 输出
C12OUT		O	CMOS	比较器 C1 和 C2 的输出
SRQ		O	CMOS	SR 锁存器输出
RC5/CCP1/P1A/T0CKI	5			
RC5		I/O	ST	数字 I/O
CCP1		I/O	ST	捕捉 1 输入 / 比较 1 输出 /PWM1 输出
P1A		O	CMOS	增强型 CCP1 PWM 输出
T0CKI		I	ST	Timer0 外部时钟输入
RC6/AN8/$\overline{SS}$/T13CKI/T1OSCI	8			
RC6		I/O	ST	数字 I/O
AN8		I	模拟	ADC 通道 8
$\overline{SS}$		I	TTL	SPI 从选择输入
T13CKI		I	ST	Timer0 和 Timer3 外部时钟输入
T1OSCI		I	XTAL	Timer1 振荡器输入
RC7/AN9/SDO/T1OSCO	9			
RC7		I/O	ST	数字 I/O
AN9		I	模拟	ADC 通道 9
SDO		O	CMOS	SPI 数据输出
T1OSCO		O	XTAL	Timer1 振荡器输出
VSS	20	P	—	逻辑和 I/O 引脚的参考地
VDD	1	P	—	逻辑和 I/O 引脚的正电源
VUSB	17	P	—	USB 收发器的正电源

图注： TTL = TTL 兼容输入
ST = 施密特触发器输入
O = 输出
XTAL = 晶振
CMOS = CMOS 兼容输入或输出
I = 输入
P = 电源
XCVR = USB 差分收发器

2.0 振荡器模块

2.1 概述

振荡器模块具有多种时钟源和特性，广泛使用于各种应用中，同时最大限度地发挥应用性能并降低功耗。图 2-1 为振荡器模块的框图。

振荡器模块的主要特性包括：

- 系统时钟选择
  - 主外部振荡器
  - 辅助外部振荡器
  - 内部振荡器
- 振荡器起振定时器
- 系统时钟选择
- 时钟切换
- 4 倍频锁相环倍频器
- CPU 时钟分频器
- USB 操作
  - 低速
  - 全速
- 双速启动模式
- 故障保护时钟监视

2.2 系统时钟选择

OSCCON 寄存器的 SCS 位用于选择以下时钟源：

- 主外部振荡器
- 辅助外部振荡器
- 内部振荡器

注： 在本文档中，系统时钟的频率将称为 FOSC。

表 2-1： 系统时钟选择

配置	选择
SCS <1:0>	系统时钟
1x	内部振荡器
01	辅助外部振荡器
00 （复位后的默认值）	由 FOSC<3:0> 定义的振荡器

SCS 位的默认状态会将系统时钟设置为由 CONFIG1H 配置寄存器的 FOSC 位定义的振荡器。系统时钟将始终由 FOSC 位定义，直到在软件中修改 SCS 位为止。

当选择内部振荡器作为系统时钟时， OSCCON 寄存器的 IRCF 位和 OSCTUNE 寄存器的 INTSRC 位将选择 LFINTOSC 或 HFINTOSC。当 IRCF<2:0> = 000，且 INTSRC 位清零时，将选择 LFINTOSC。 IRCF 位和 INTSRC 位的所有其他组合都选择 HFINTOSC 作为系统时钟。

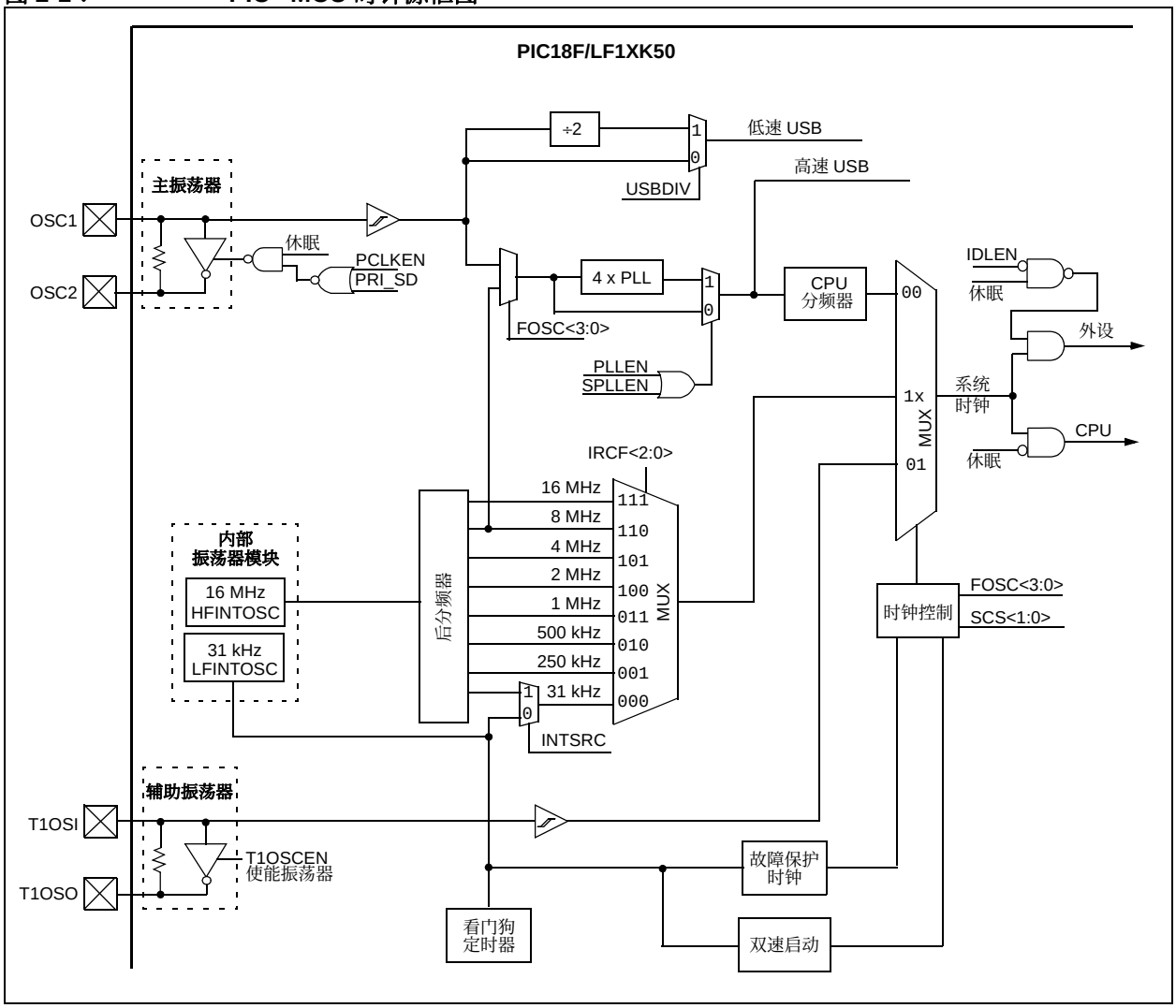
2.3 主外部振荡器

主外部振荡器的工作模式通过设置 CONFIG1H 配置寄存器的 FOSC<3:0> 位进行选择。振荡器可设置为以下几种模式：

- LP：低功耗晶振
- XT：晶振 / 陶瓷谐振器
- HS：高速晶振 / 谐振器
- RC：外部 RC 振荡器
- EC：外部时钟

此外，还可以在固件控制下关闭主外部振荡器，以节省功耗。

图 2-1：　　PIC® MCU 时钟源框图



2.3.1 主外部振荡器关闭

主外部振荡器可以通过软件使能或禁止。要允许软件控制主外部振荡器，必须将 CONFIG1H 配置寄存器的 PCLKEN 位置 1。PCLKEN 位置 1 时，主外部振荡器由 OSCCON2 寄存器的 PRI_SD 位控制。当 PRI_SD 位置 1 时，主外部振荡器将被使能；当 PRI_SD 位清零时，它将被禁止。

注： 当选择主外部振荡器作为系统时钟时，将不能关闭它。要关闭该振荡器，系统时钟源必须为辅助振荡器或内部振荡器。

2.3.2 LP、 XT 和 HS 振荡器模式

LP、XT 和 HS 模式支持使用连接到 OSC1 和 OSC2 的石英晶体谐振器或陶瓷谐振器 （图 2-2）。模式选择内部反相放大器的低、中或高增益设定，以支持各种谐振器类型及速度。

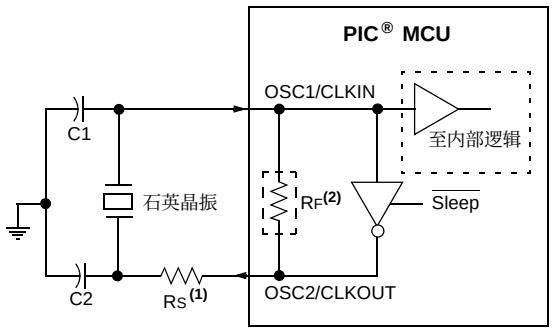
LP振荡器模式选择内部反相放大器的最低增益设定。LP 模式的电流消耗在三种模式中最小。该模式最适合驱动具备低驱动电平规格要求的谐振器，例如，音叉（Tuning Fork）型晶振。

XT 振荡器模式选择内部反相放大器的中等增益设定。XT 模式的电流消耗在三种模式中居中。该模式最适合驱动具备中等驱动电平规格要求的谐振器。

HS 振荡器模式选择内部反相放大器的最高增益设定。HS 模式的电流消耗在三种模式中最大。该模式最适合驱动需要高驱动设定的谐振器。

图 2-2 和图 2-3 分别给出了石英晶体谐振器和陶瓷谐振器的典型电路。

图 2-2： 石英晶振的工作原理（LP、 XT 或 HS 模式）



注 1： 对于具有低驱动电平规格的石英晶振，可能需要串联一个电阻 （RS）。

2： RF 的值根据所选的振荡器模式变化（典型值在 2 MΩ 到 10 MΩ 之间）。

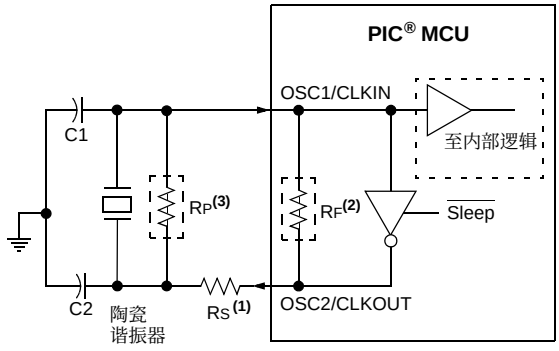
注 1： 石英晶振的特性随类型、封装和制造商而变化。要了解规格说明和推荐应用，应查阅制造商提供的数据手册。

2： 应始终验证振荡器在应用要求的VDD和温度范围内的性能。

3： 如需振荡器设计帮助，请参见以下 Microchip 应用笔记：

- AN826，“*Crystal Oscillator Basics and Crystal Selection for rfPIC® and PIC® Devices*”（DS00826）
- AN849，“*Basic PIC® Oscillator Design*”（DS00849）
- AN943，“*Practical PIC® Oscillator Analysis and Design*”（DS00943）
- AN949，“*Making Your Oscillator Work*”（DS00949）

图 2-3：　　陶瓷谐振器的工作原理（XT 或 HS 模式）



注　**1：**对于具有低驱动电平规格的陶瓷谐振器，可能需要串联一个电阻（RS）。

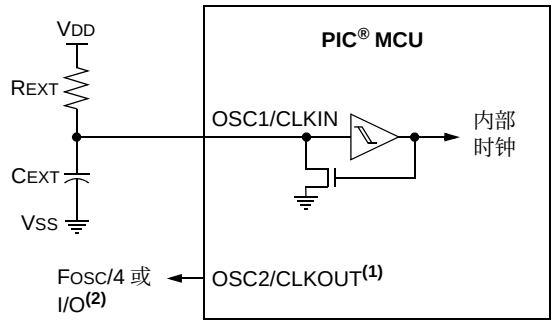
2：RF 的值根据所选的振荡器模式变化（典型值在 2 MΩ 到 10 MΩ 之间）。

3：为使陶瓷谐振器正常工作，可能需要并联一个反馈电阻（RP）。

2.3.3　外部 RC

外部阻容（Resistor-Capacitor，RC）模式支持使用外部 RC 电路。对时钟精度要求不高时，这使设计人员有了很大的频率选择空间，且保持成本最低。在 RC 模式下，RC 电路连接到 OSC1，使 OSC2 可以配置为 I/O 或 CLKOUT。CLKOUT 功能通过 CONFIG1H 配置寄存器的 FOSC 位进行选择。当 OSC2 配置为 CLKOUT 时，引脚上的频率是 RC 振荡器频率的 4 分频。图 2-4 给出了外部 RC 模式的连接图。

图 2-4：　　外部 RC 模式



推荐值：　10 kΩ ≤ REXT ≤ 100 kΩ
CEXT > 20 pF

注　**1：**第 1.0 节“器件概述”中列出了备用引脚功能。

2：输出取决于 RC 或 RCIO 时钟模式。

RC 振荡器频率与供电电压、电阻 REXT 和电容 CEXT 以及工作温度有关。影响振荡器频率的其他因素有：

- 输入电压门限值变化
- 元件公差
- 不同封装的电容

2.3.4　外部时钟

外部时钟（External Clock，EC）模式允许外部产生的逻辑电平时钟用作系统时钟源。在该模式下工作时，外部时钟源连接到 OSC1，允许将 OSC2 配置为 I/O 或 CLKOUT。CLKOUT 功能通过 CONFIG1H 配置寄存器的 FOSC 位进行选择。当 OSC2 配置为 CLKOUT 时，引脚上的频率是 EC 振荡器频率的 4 分频。

对于 EC 模式，有三种不同的功耗设置可用。如果已知 EC 时钟处于特定范围之内，则可以通过功耗设置降低器件的 IDD。如果 EC 时钟存在预期的频率范围，则选择对应于最高频率的功耗模式。

EC	低功耗	0 – 250 kHz
EC	中等功耗	250 kHz – 4 MHz
EC	高功耗	4 – 48 MHz

2.4　辅助外部振荡器

辅助外部振荡器设计为驱动外部 32.768 kHz 晶振。可通过 T1CON 寄存器的 T1OSCEN 位使能或禁止该振荡器。更多信息，请参见**第 11.0 节“Timer1 模块”**。

2.5 内部振荡器

内部振荡器模块包含两个独立的振荡器，它们是：

- LFINTOSC：低频内部振荡器
- HFINTOSC：高频内部振荡器

当使用其中任一振荡器工作时，OSC1 将作为 I/O，OSC2 将作为 I/O 或 CLKOUT。CLKOUT 功能通过 CONFIG1H 配置寄存器的 FOSC 位进行选择。当 OSC2 配置为 CLKOUT 时，引脚上的频率是内部振荡器频率的 4 分频。

2.5.1 LFINTOSC

低频内部振荡器（LFINTOSC）是 31 kHz 内部时钟源。LFINTOSC 振荡器是以下模块的时钟源：

- 上电延时定时器
- 看门狗定时器
- 故障保护时钟监视器

当以下任一条件为真时，将使能 LFINTOSC：

- 上电延时定时器使能 （PWRTEN = 0）
- 看门狗定时器使能 （WDTEN = 1）
- 看门狗定时器通过软件使能 （WDTEN = 0 且 SWDTEN = 1）
- 故障保护时钟监视器使能 （FCMEM = 1）
- SCS1 = 1、IRCF<2:0> = 000 且 INTSRC = 0
- FOSC<3:0> 选择内部振荡器作为主时钟，IRCF<2:0> = 000 且 INTSRC = 0
- IESO = 1 （双速启动）、IRCF<2:0> = 000 且 INTSRC = 0

2.5.2 HFINTOSC

高频内部振荡器（HFINTOSC）是高精度振荡器，出厂时校准为以 16 MHz 工作。HFINTOSC 的输出连接到后分频器和多路开关 （见图 2-1）。使用 OSCCON 寄存器的 IRCF<2:0> 位，可选择 8 个频率之一。可以从 HFINTOSC 获取以下频率：

- 16 MHZ
- 8 MHZ
- 4 MHZ
- 2 MHZ
- 1 MHz （复位后的默认值）
- 500 kHz
- 250 kHz
- 31 kHz

OSCCON 寄存器的 HFIOFS 位用于指示 HFINTOSC 是否稳定。

> **注** **1：** 选择从 HFINTOSC 振荡器获取 31 kHz 时钟时，要求 IRCF<2:0> = 000，且 OSCTUNE 寄存器的 INTSRC 位置 1。如果 INTSRC 位清零，系统时钟将来自 LFINTOSC。
>
> **2:** 此外，通过 OSCTUNE 寄存器，还可以对 HFINTOSC 的频率进行进一步调整。更多详细信息，请参见寄存器 2-3。

当以下任一条件为真时，将使能 HFINTOSC：

- SCS1 = 1 且 IRCF<2:0> ≠ 000
- SCS1 = 1、IRCF<2:0> = 000 且 INTSRC = 1
- FOSC<3:0> 位选择内部振荡器作为主时钟，并且
  - IRCF<2:0> ≠ 000 或
  - IRCF<2:0> = 000 且 INTSRC = 1
- IESO = 1 （双速启动），并且
  - IRCF<2:0> ≠ 000 或
  - IRCF<2:0> = 000 且 INTSRC = 1
- FCMEM = 1 （故障保护时钟监视），并且
  - IRCF<2:0> ≠ 000 或
  - IRCF<2:0> = 000 且 INTSRC = 1

2.6 振荡器控制

振荡器控制 （OSCCON）（寄存器 2-1）和振荡器控制 2 （OSCCON2）（寄存器 2-2）寄存器用于控制系统时钟和频率选择选项。

寄存器 2-1： OSCCON：振荡器控制寄存器

R/W-0	R/W-0	R/W-1	R/W-1	R-q	R-0	R/W-0	R/W-0
IDLEN	IRCF2	IRCF1	IRCF0	OSTS(1)	HFIOFS	SCS1	SCS0
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	q = 值取决于具体条件
-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>：**内部振荡器频率选择位
111 = 16 MHz
110 = 8 MHz
101 = 4 MHz
100 = 2 MHz
011 = 1 MHz(3)
010 = 500 kHz
001 = 250 kHz
000 = 31 kHz(2)

bit 3 **OSTS：**振荡器起振延时状态位 (1)
1 = 器件运行在 CONFIG1 寄存器的 FOSC<2:0> 定义的时钟之下
0 = 器件运行在内部振荡器 （HFINTOSC 或 LFINTOSC）之下

bit 2 **HFIOFS：**HFINTOSC 频率稳定位
1 = HFINTOSC 频率稳定
0 = HFINTOSC 频率不稳定

bit 1-0 **SCS<1:0>：**系统时钟选择位
1x = 内部振荡器模块
01 = 辅助 （Timer1）振荡器
00 = 主时钟 （由 CONFIG1H[FOSC<3:0>] 决定）

注 1： 复位状态取决于 IESO 配置位的状态。
2： 时钟源由 OSCTUNE 寄存器的 INTSRC 位选择，请参见上文。
3： 复位时 HFINTOSC 的默认输出频率。

寄存器 2-2：　　OSCCON2：振荡器控制寄存器 2

U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-0	R-x
—	—	—	—	—	PRI_SD	HFIOFL	LFIOFS
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	q = 值取决于具体条件
-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　**PRI_SD：**主振荡器驱动电路关闭位
1 = 振荡器驱动电路开启
0 = 振荡器驱动电路关闭 （零功耗）

bit 1　**HFIOFL：**HFINTOSC 频率锁定位
1 = HFINTOSC 处于锁定状态
0 = HFINTOSC 尚未锁定

bit 0　**LFIOFS：**LFINTOSC 频率稳定位
1 = LFINTOSC 稳定
0 = LFINTOSC 不稳定

2.6.1 OSCTUNE 寄存器

HFINTOSC 在出厂时已校准，但可通过在软件中写入 OSCTUNE 寄存器（寄存器 2-3）的 TUN<5:0> 位来进行调节。

TUN<5:0> 的默认值为 000000。该值是一个 6 位的二进制补码。

当 OSCTUNE 寄存器被修改时， HFINTOSC 频率将开始转变为新频率。转变期间，代码将继续执行，是否已发生频率转变并无明确的指示。

OSCTUNE 不影响 LFINTOSC 频率。依赖于 LFINTOSC 时钟源频率的功能，如上电延时定时器 （PWRT）、看门狗定时器 （WDT）、故障保护时钟监视器 （FSCM）以及外设等，其工作不受频率改变的影响。

OSCTUNE 寄存器也有 INTSRC 和 SPLLEN 位，它们控制内部振荡器模块的某些功能。

当选取 31 kHz 频率后，用户可通过 INTSRC 位选择用作时钟源的内部振荡器。在**第 2.5.1 节 “ LFINTOSC ”**中对此进行了更详细的说明。

SPLLEN 位控制倍频器的工作。关于 SPLLEN 位的功能的更多详细信息，请参见**第 2.9 节 “ 4 倍频锁相环倍频器 ”**。

寄存器 2-3： 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	SPLLEN	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 16 MHz HFINTOSC 源的 31.25 kHz 器件时钟 （使能 512 分频）
0 = 直接来自 LFINTOSC 内部振荡器的 31 kHz 器件时钟

bit 6 **SPLLEN：**软件控制的倍频器 PLL 位
1 = 使能 PLL （仅限 HFINTOSC 8 MHz）
0 = 禁止 PLL

bit 5-0 **TUN<5:0>：**频率调节位
011111 = 最高频率
011110 =
•••
000001 =
000000 = 振荡器模块运行在出厂时已校准的频率下
111111 =
•••
100000 = 最低频率

2.7 振荡器起振定时器

当配置为 LP、 XT 或 HS 模式时，主外部振荡器会配合使用振荡器起振定时器 （Oscillator Start-up Timer，OST）。OST 用于确保振荡器起振，为振荡器模块提供稳定的时钟。当 OSC1 上发生 1024 次振荡时，OST 发生超时。在 OST 周期中，如果系统时钟设置为主外部振荡器，程序计数器不会递增，程序暂停执行。在 OST 周期中，将发生以下事件：

- 上电复位 （POR）
- 欠压复位 （BOR）
- 从休眠状态唤醒
- 使能振荡器
- 上电延时定时器 （PWRT）延时结束

为了使外部振荡器起振和代码执行之间的延时最小，可选择双速启动模式。更多信息，请参见**第 2.12 节 “ 双速启动模式 ”**。

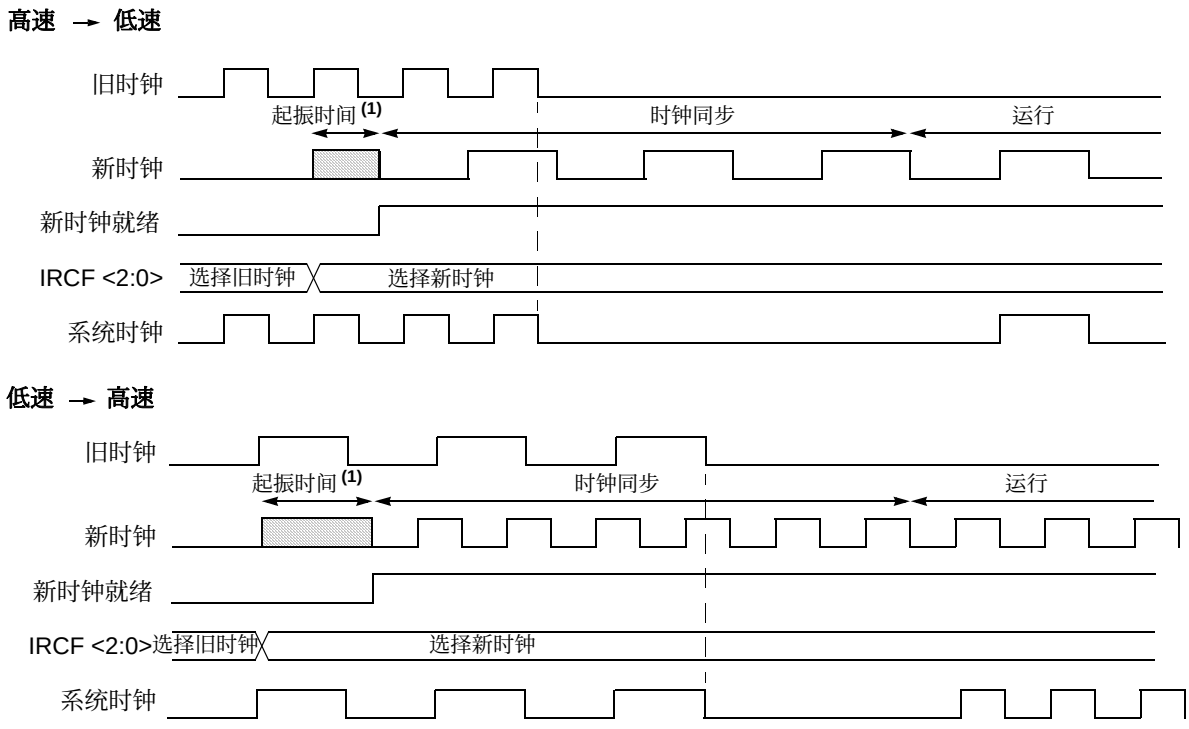
2.8 时钟切换

器件含有用于防止由于系统时钟源改变而产生时钟 “ 毛刺 ” 的电路。为此，在时钟切换时，系统时钟会有短暂的停顿。如果新时钟源未稳定（即，OST 有效），器件将使用旧时钟源继续执行，直到新时钟源稳定为止。时钟切换时序如下：

1. 修改 OSCCON 寄存器的 SCS<1:0> 位。
2. 系统时钟将使用旧时钟继续工作，直到新时钟就绪。
3. 时钟切换电路在新时钟就绪后等待旧时钟的两个连续的上升沿。
4. 从旧时钟的下一个下降沿开始，系统时钟保持低电平。
5. 时钟切换电路再等待新时钟的两个上升沿。
6. 在新时钟的下一个下降沿，将系统时钟拉为低电平，并将新时钟切换为系统时钟。
7. 时钟切换完成。

更多详细信息，请参见图 2-5。

图 2-5： 时钟切换时序



注 1： 起振时间包括外部时钟的 TOST（1024 个 TOSC），加上 HSPLL 模式的 TPLL（大约 2 ms）。

表 2-2： 由于时钟切换导致的延时示例

切换自	切换到	振荡器延时
休眠 /POR	LFINTOSC HFINTOSC	振荡器预热延时 （TWARM）
休眠 /POR	LP、 XT 或 HS	1024 个时钟周期
休眠 /POR	EC 或 RC	8 个时钟周期

2.9 4 倍频锁相环倍频器

对于希望使用较低频率外部振荡器或使用 HFINTOSC 以 32 MHz 工作的用户，提供了锁相环 （PLL）电路作为一种选项。PLL 设计为使用 4 MHz 至 12 MHz 的输入频率。使能 PLL 时，PLL 会将其输入频率进行 4 倍频。对于担心高频晶振引起 EMI 的用户，这可能会很有用。

有两个位控制 PLL：CONFIG1H 配置寄存器的 PLLEN 位和 OSCTUNE 寄存器的 SPLLEN 位。当 PLLEN 位置 1时，PLL被使能；当PLLEN位清零时，它由软件控制。

表 2-3： PLL 配置

PLLEN	SPLLEN	PLL 状态
1	x	使能 PLL
0	1	使能 PLL
0	0	禁止 PLL

2.9.1 32 MHz 内部振荡器频率选择

内部振荡器模块可同与外部振荡器模块有关的 4 倍频锁相环配合使用来产生一个 32 MHz 的内部系统时钟源。使用 32 MHz 的内部时钟源需要进行下列设置：

- 要使用 INTOSC 源作为器件的系统时钟，必须设置 CONFIG1H 配置寄存器中的 FOSC 位 （FOSC<3:0> = 1000 或 1001）。
- OSCCON 寄存器中的 SCS 位必须清零，以使用由 CONFIG1H 寄存器中的 FOSC<3:0> 决定的时钟 （SCS<1:0> = 00）。
- OSCCON 寄存器中的 IRCF 位必须设置为使用 8 MHz HFINTOSC （IRCF<2:0> = 110）。
- OSCTUNE 寄存器中的 SPLLEN 位必须置 1 以使能 4 倍频锁相环，或 CONFIG1H 寄存器中的 PLLEN 位必须被编程为 1。

注： 使用 CONFIG1H 配置寄存器的 PLLEN 位时，4 倍频锁相环不能通过软件禁用，且 8 MHz HFINTOSC 选项不再可用。

当 OSCCON 寄存器的 SCS 位设置为 1x 时，4 倍频锁相环不能和内部振荡器一起使用。要使 4 倍频锁相环与内部振荡器一起使用， SCS 位必须设置为 00。

2.10 CPU 时钟分频器

CPU 时钟分频器使系统时钟可以低于低速 / 全速 USB 模块时钟的速度运行，但两者共用相同的时钟源。只有由 CONFIG1H 配置寄存器的 FOSC 位的设置定义的振荡器可以用于 CPU 时钟分频器。CPU 时钟分频器通过 CONFIG1L 配置寄存器的 CPUDIV 位进行控制。设置 CPUDIV 位可以将系统时钟设置为：

- 等于 USB 模块的时钟速度
- USB 模块时钟速度的一半
- USB 模块时钟速度的三分之一
- USB 模块时钟速度的四分之一

关于 CPU 时钟分频器的更多信息，请参见图 2-1 和寄存器 24-1 CONFIG1L。

2.11 USB 操作

USB 模块设计为以两种不同模式工作：

- 低速
- 全速

由于 USB 规范所规定的时序要求， USB 模块需要主外部振荡器。 CONFIG1H 配置寄存器的 FOSC 位必须设置为外部时钟 （EC）高功耗或时钟频率为 6、 12 或 48 MHz 的 HS 模式。

2.11.1 低速操作

对于低速 USB 操作，USB 模块需要 6 MHz 的时钟。要产生 6 MHz 时钟，仅允许 2 种振荡器模式：

- EC 高功耗模式
- HS 模式

表 2-4 列出了对于低速操作建议的时钟模式。

注：	如果使用 24 MHz 或更低的 （64 MHz 最理想）CPU 时钟频率，用户必须使用 USB 低速操作。如果使用的频率大于 24 MHz，则需要至少 14 个指令周期的固件延时。

2.11.2 全速操作

对于全速 USB 操作， USB 模块需要 48 MHz 的时钟。要产生 48 MHz 时钟，仅允许 2 种振荡器模式：

- EC 高功耗模式
- HS 模式

表 2-5 列出了对于全速操作建议的时钟模式。

表 2-4： 低速 USB 时钟设置

时钟模式	时钟频率	USBDIV	使能 4 倍频 PLL	CPUDIV<1:0>	系统时钟频率 （MHz）
EC 高功耗 /HS	12 MHz	1	是	00	48
				01	24
				10	16
				11	12
			否	00	12
				01	6
				10	4
				11	3
	6 MHz	0	是	00	24
				01	12
				10	8
				11	6
			否	00	6
				01	3
				10	2
				11	1.5

注： 只有 OSCCON 寄存器位 SCS<1:0> = 00 时，表 2-4 中的系统时钟频率才适用。通过更改这些位，系统时钟最低可以 31 kHz 的频率工作。

表 2-5： 全速 USB 时钟设置

时钟模式	时钟频率	使能 4 倍频 PLL	CPUDIV<1:0>	系统时钟频率 （MHz）
EC 高功耗	48 MHz	否	00	48
			01	24
			10	16
			11	12
EC 高功耗 /HS	12 MHz	是	00	48
			01	24
			10	16
			11	12

注： 只有 OSCCON 寄存器位 SCS<1:0> = 00 时，上表中的系统时钟频率才适用。通过更改这些位，系统时钟最低可以 31 kHz 的频率工作。

2.12 双速启动模式

双速启动模式通过最大限度地缩短外部振荡器起振定时器 （OST）与代码执行之间的延时，进一步节省了功耗。在频繁使用休眠模式的应用中，双速启动可以消除 OST 周期，这可以降低器件的总体功耗。

通过将 CONFIG1H 配置寄存器的 IESO 位置 1 来使能双速启动模式。使能双速启动时，在主外部振荡器 OST 周期中，器件将使用内部振荡器来执行指令。

当系统时钟设置为主外部振荡器，且振荡器配置为 LP、XT 或 HS 模式时，在 OST 周期中，器件将不会执行代码。OST 将暂停程序执行，直到完成 1024 次振荡计数。双速启动模式在 OST 有效时使用内部振荡器进行工作，最大限度地缩短了代码执行的延时。在 OST 周期结束后，系统时钟将切换回主外部振荡器。

在以下事件之后，双速启动将变为有效：

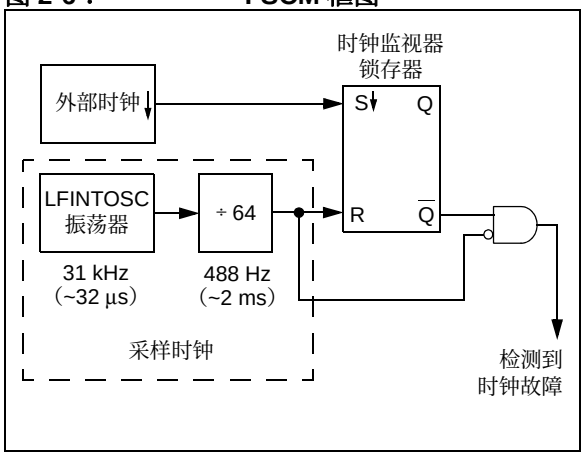
- 上电复位 （POR）
- 上电延时定时器 （PWRT）（如果使能）
- 从休眠状态唤醒

OSCCON 寄存器的 OSTS 位报告器件当前使用哪个振荡器工作。当 OSTS 位置 1 时，器件使用由 CONFIG1H 配置寄存器的 FOSC 位定义的振荡器工作。当 OSTS 位清零时，器件使用内部振荡器工作。

2.13 故障保护时钟监视器

故障保护时钟监视器 （FSCM）使得器件在出现外部振荡器故障时仍能继续工作。FSCM 能在振荡器起振延时定时器 （OST）延时结束后的任一时刻检测振荡器故障。FSCM 通过将 CONFIG1H 配置寄存器中的 FCMEN 位置1来使能。FSCM可用于所有外部振荡器模式（LP、XT、 HS、 EC 和 RC）。

图 2-6： FSCM 框图



2.13.1 故障保护检测

FSCM 模块通过将外部振荡器与 FSCM 采样时钟比较来检测振荡器故障。 LFINTOSC 64 分频，就产生了采样时钟。请参见图 2-6。故障检测器内部有一个锁存器。在外部时钟的每个下降沿，锁存器被置 1。在采样时钟的每个上升沿，锁存器被清零。如果已经经过采样时钟的整个半周期，但主时钟仍未变为低电平，则会检测到故障。

2.13.2 故障保护操作

当外部时钟出现故障时，FSCM 将器件时钟切换到内部时钟源，并将 PIR2 寄存器的 OSFIF 标志位置 1。如果 PIE2 寄存器的 OSCFIE 位也置 1，则 OSCFIF 标志将产生一个中断。器件固件随后会采取措施减轻可能由故障时钟所产生的问题。系统时钟将继续来自内部时钟源，直到器件固件成功重启外部振荡器且切换回外部振荡器进行操作。不会自动转换回发生故障的时钟源。

FSCM 所选的内部时钟源由 OSCCON 寄存器的 IRCF<2:0> 位决定。这使得可以在故障发生前配置内部振荡器。

2.13.3 故障保护条件清除

故障保护条件通过以下事件之一清除：

- 任何复位
- 通过翻转 OSCCON 寄存器的 SCS1 位

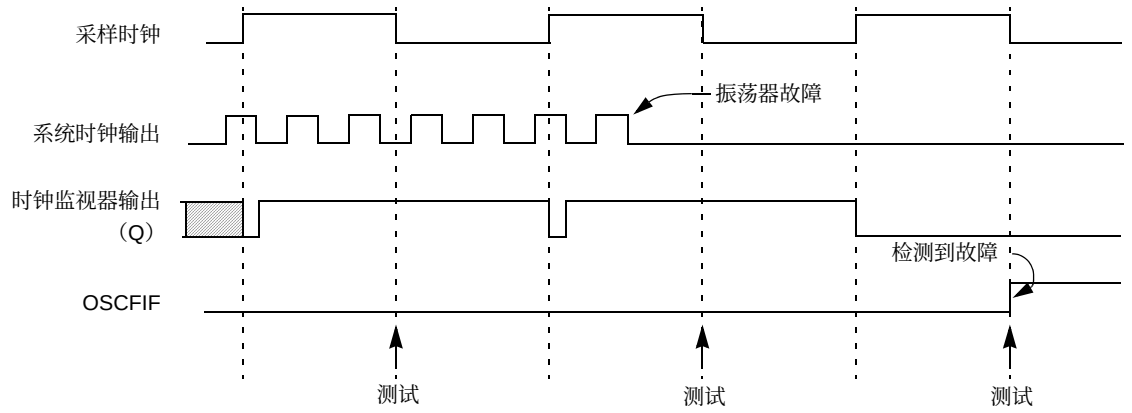
这两个条件都会重新启动 OST。OST 运行时，器件继续依靠 OSCCON 中选定的 INTOSC 进行操作。OST 超时后，故障保护条件被清除，器件自动切换到外部时钟源。在 OSCFIF 标志清零之前，不需要清除故障保护条件。

2.13.4 复位或从休眠中唤醒

FSCM 设计为能在振荡器起振延时定时器（OST）延时结束后的任一时刻检测振荡器故障。OST 用在从休眠状态唤醒后以及任何类型的复位后。OST 不能在 EC 或 RC 时钟模式下使用，所以一旦复位或唤醒完成，FSCM 就处于活动状态。当 FSCM 被使能时，双速启动也被使能。因此，当 OST 运行时，器件总是处于代码执行阶段。

注：	由于振荡器起振时间范围较大，在振荡器起振期间（即，从复位或休眠中退出时），故障保护电路不处于活动状态。经过一段适当的时间后，用户应检查 OSCCON 寄存器的 OSTS 位，以验证振荡器是否已成功起振以及系统时钟是否切换成功。

图 2-7： FSCM 时序图



注： 系统时钟频率通常远高于采样时钟频率。本例中显示的差距只是为了便于说明。

表 2-6： 与时钟源相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
CONFIG1H	IESO	FCMEN	PCLKEN	PLLEN	FOSC3	FOSC2	FOSC1	FOSC0	294
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	285
OSCCON	IDLEN	IRCF2	IRCF1	IRCF0	OSTS	HFIOFS	SCS1	SCS0	286
OSCTUNE	INTSRC	SPLLEN	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0	288
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	288
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	288
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	105

图注： x = 未知，u = 不变，— = 未实现位（读为 0）。振荡器不使用阴影单元。

注 1： 其他（非上电）复位包括在正常操作期间的 $\overline{\text{MCLR}}$ 复位和看门狗定时器复位。

3.0 存储器构成

PIC18 增强型单片机器件有三种类型的存储器：

- 程序存储器
- 数据 RAM
- 数据 EEPROM

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。出于实用目的，可将数据 EEPROM 当作外设，因为它可以通过一组控制寄存器进行寻址和访问。

第 4.0 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。数据 EEPROM将单独在**第 5.0 节 “ 数据 EEPROM 存储器 ”** 中讨论。

3.1 程序存储器构成

PIC18 单片机具有一个 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这个 2 MB 地址之间的存储单元会返回全 0 （NOP 指令）。

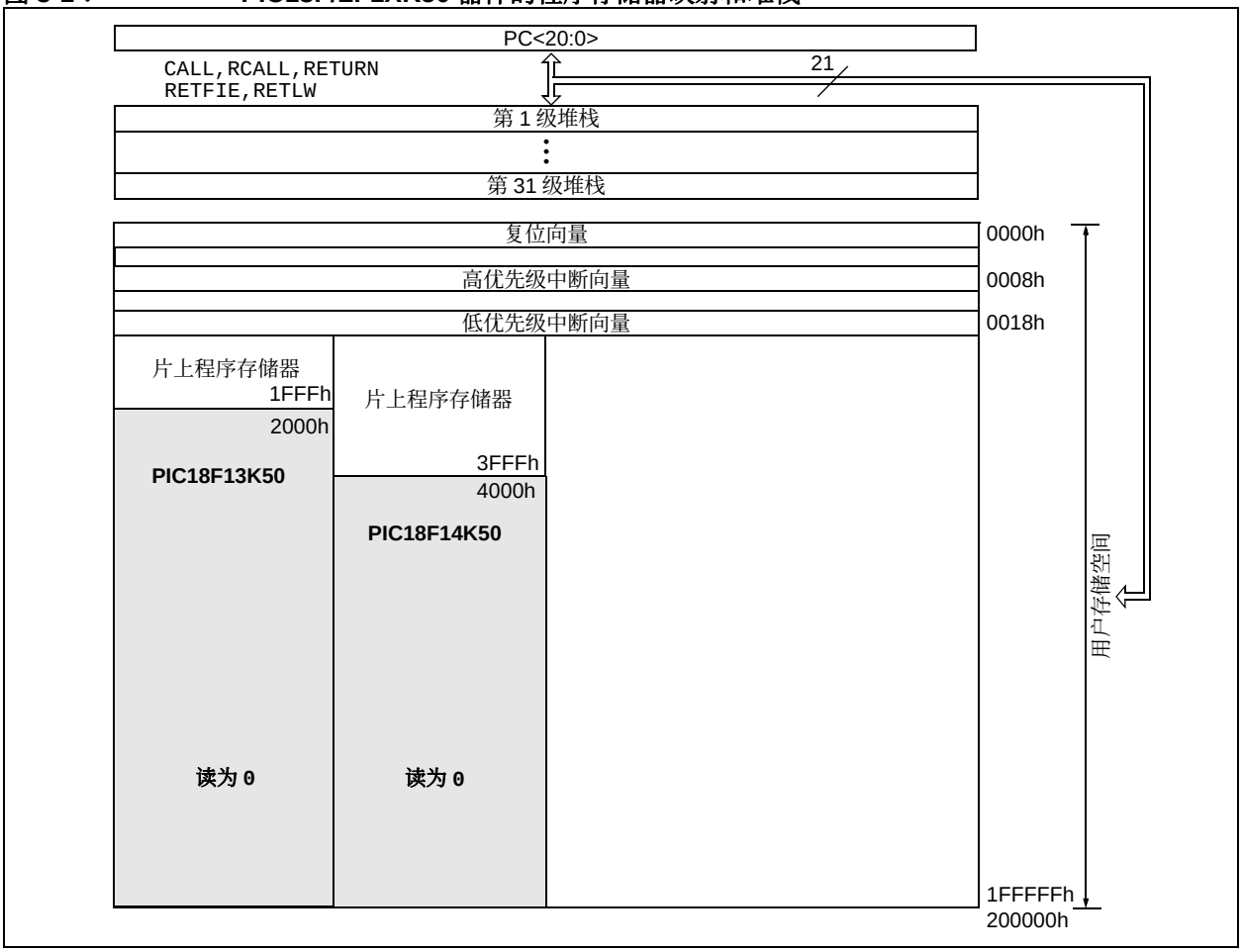
本系列器件包含的闪存如下：

- PIC18F13K50 ：8 KB 闪存，最多 4,096 条单字指令
- PIC18F14K50 ：16 KB闪存，最多8,192条单字指令

PIC18 器件有两个中断向量和一个复位向量。复位向量地址为 0000h，中断向量地址为 0008h 和 0018h。

图 3-1 给出了 PIC18F/LF1XK50 器件的程序存储器映射。存储器框图的详细信息如图 24-2 所示。

图 3-1： PIC18F/LF1XK50 器件的程序存储器映射和堆栈



3.1.1 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。这对于计算 PC 的偏移量很有用处（见**第 3.1.4.1 节“计算 GOTO”**）。

PC 是按字节寻址程序存储器的。为了防止 PC 不能正确获取字指令，需要将 PCL 的最低有效位（Least Significant bit，LSb）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 和程序转移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。

3.1.2 返回地址堆栈

返回地址堆栈允许保存最多 31 个程序调用地址和中断向量。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（Top-of-Stack，TOS）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，产生压栈操作：首先堆栈指针加 1，并且将 PC 的内容写入堆栈指针所指向的地址单元（PC 已经指向 CALL 下一条指令）。执行 RETURN 类型的指令时，产生出栈操作：STKPTR 寄存器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。

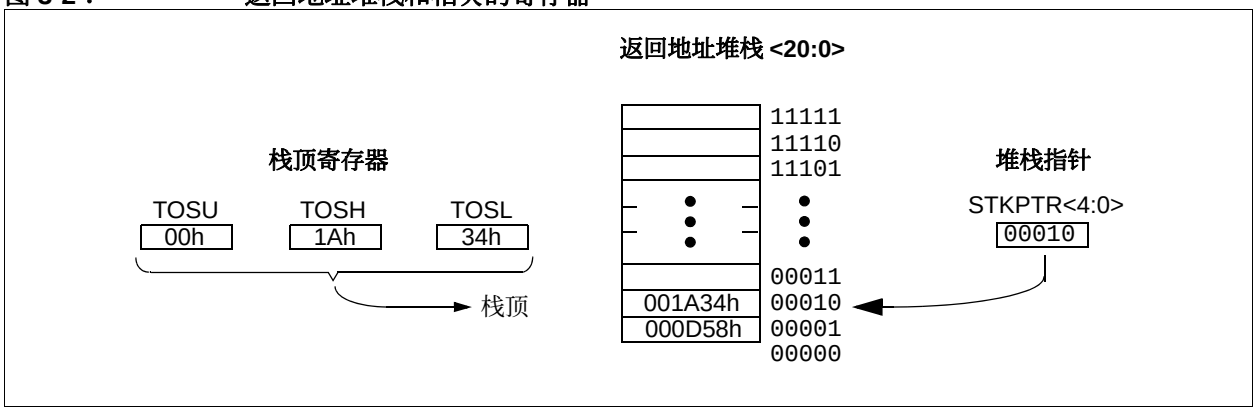
所有复位后，堆栈指针均会初始化为 00000。堆栈指针值 00000 不指向任何 RAM 单元；它仅仅是一个复位值。状态位表明堆栈是已满、上溢还是下溢。

3.1.2.1 访问栈顶

只可读写返回地址堆栈的栈顶（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 3-2）。这可以让用户在必要时实现软件堆栈。在 CALL、RCALL 或中断后，软件可以通过读取TOSU:TOSH:TOSL寄存器来读取压入堆栈的值。这些值可以被置入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 3-2：　返回地址堆栈和相关的寄存器



3.1.2.2 返回堆栈指针 （STKPTR）

STKPTR 寄存器（寄存器 3-1）包含堆栈指针值、STKFUL（堆栈满）位和 STKUNF（堆栈下溢）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 使 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 24.1 节 “ 配置位 ”**。）如果 STVREN 位已经置 1（默认），第 31 次进栈将把 （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 已经清零，第 31 次进栈时 STKFUL 位将会置 1，堆栈指针递增到 31。任何其他进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。

3.1.2.3 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。这样就可以修改 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 3-1： 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：			
R = 可读位	W = 可写位	U = 未实现	C = 只可清零位
-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 = 堆栈未满或未溢出

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。

3.1.2.4 堆栈满和下溢复位

通过将配置寄存器 4L 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使 STKFUL 或 STKUNF 位清零。

3.1.3 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。每个堆栈只有一级且不可读写。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, FAST` 指令从中断返回，这些寄存器中的值就会被装回相关的寄存器。

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。如果在为低优先级中断提供服务时，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。在为低优先级中断提供服务时，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后恢复 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 3-1 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 3-1： 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
```

3.1.4 程序存储器中的查找表

有的编程场合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

3.1.4.1 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 3-2 给出了一个示例。

使用 `ADDWF PCL` 指令和一组 `RETLW nn` 指令可以创建一个查找表。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 （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 3-2： 使用偏移量计算 GOTO

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

3.1.4.2 表读与表写

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。

使用表读和表写，每个程序字可以存储 2 个字节的查找表数据。表指针 （TBLPTR）寄存器指定字节地址，而表锁存寄存器 （TABLAT）则存储从程序存储器中读取或写入的数据。一次只能与程序存储器传送一个字节。

第 4.1 节 “ 表读与表写 ” 将进一步讨论表读和表写操作。

3.2 PIC18 指令周期

3.2.1 时钟分配

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）。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中。指令的译码和执行在下一个 Q1 到 Q4 周期完成。图 3-3 所示为时钟和指令执行的流程图。

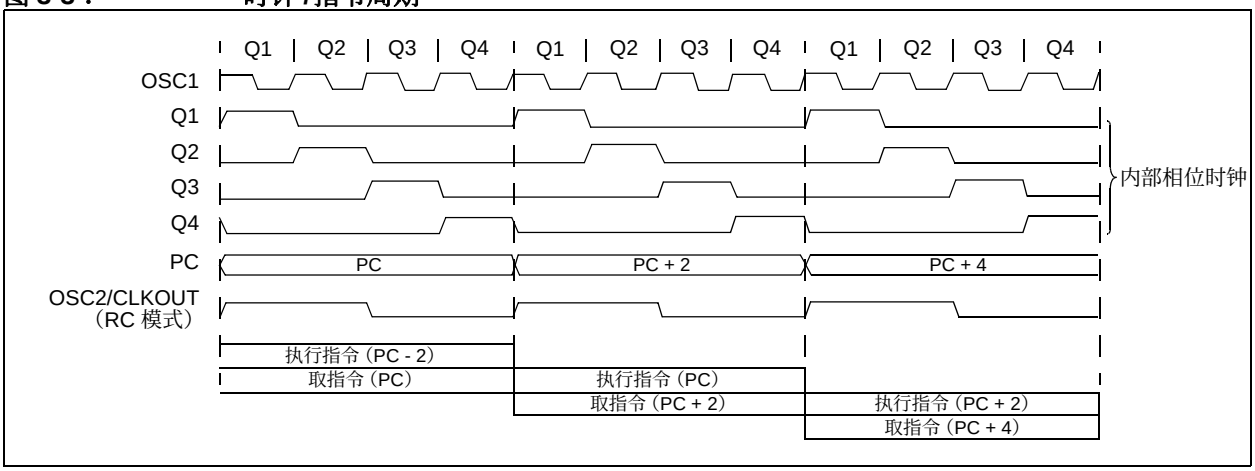
3.2.2 指令流 / 流水线

一个指令周期由 Q1 到 Q4 四个周期组成。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 3-3）。

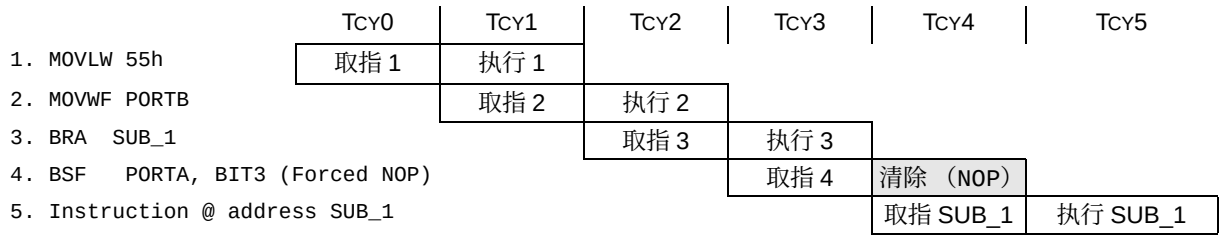
程序计数器（PC）在 Q1 周期加 1，开始取指令这意味着取指周期的开始。

指令执行周期中：在 Q1 周期，将所取指令锁存到指令寄存器（Instruction Register，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 3-3： 时钟 / 指令周期



例 3-3： 指令流水线流程



除程序转移指令外，所有的指令都是单周期指令。由于程序转移指令将导致一条已取指令从流水线清除，需要重新取指令，然后执行指令，所以程序转移指令需要两个周期。

3.2.3 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）。要保证正确指向指令单元，PC 必须以 2 为单位递增，并且 LSb 总是读为 0（见**第 3.1.1 节 “ 程序计数器 ”**）。

图 3-4 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。由于指令总是存储为一个字长，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。图 3-4 中的指令 2 给出了指令 GOTO 0006h 在程序存储器中的译码过程。程序转移指令也采取同样的方式对相对地址偏移量进行译码。存储在转移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 25.0 节 “ 指令集汇总 ”** 提供了指令集的更多详细信息。

图 3-4： 程序存储器中的指令

```
                                                    字地址
                                  LSB = 1   LSB = 0    ↓
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  │
                                  └─────────┴─────────┴──────────┘
```

3.2.4 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 4 个最高有效位（Most Significant bits，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，那么将执行一条 NOP 指令取而代之。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 3-4 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 3.6 节 “ PIC18 指令执行和扩展指令集 ”**。

例 3-4： 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

3.3 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 3.5 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，每个寄存器都有 12 位地址，数据存储器最多有 4096 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。图 3-5 和图 3-6 给出了 PIC18F/LF1XK50 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则在用户应用程序中存储数据和临时存储操作的中间结果。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存取常用寄存器（SFR 和所选的 GPR），PIC18 器件设置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 SFR 和 GPR Bank 0 的低地址单元的快速存取，而无需使用存储区选择寄存器（Bank Select Register，BSR）。**第 3.3.3 节 “ 快速操作存储区 ”**提供了对于快速操作 RAM 的详细说明。

3.3.1 USB RAM

数据存储器的部分实际上映射到特殊的双端口快速操作 RAM。当禁止 USB 模块时，这些存储区中的 GPR 和数据存储空间中的任何其他 GPR 一样使用。

当使能 USB 模块时，这些存储区中的存储空间分配为用于 USB 操作的缓冲 RAM。该区域在单片机内核和 USB 串行接口引擎（Serial Interface Engine，SIE）之间共用，用于在两者间直接传输数据。

将未分配为 USB 缓冲区的 USB RAM 单元用作普通临时存储空间或其他变量的存储，理论上是可能的。但实际上，缓冲区分配的动态特性使这种做法的风险很大。**第 22.0 节 “ 通用串行总线（USB）”**中提供了 USB RAM 和缓冲区操作的更多信息。

3.3.2 存储区选择寄存器（BSR）

容量较大的数据存储器需要有效的寻址机制，以便对所有地址进行快速存取。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存取的。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（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；它们始终读为 0 且不能被写入。可以通过使用 MOVLB 指令直接装入 BSR。

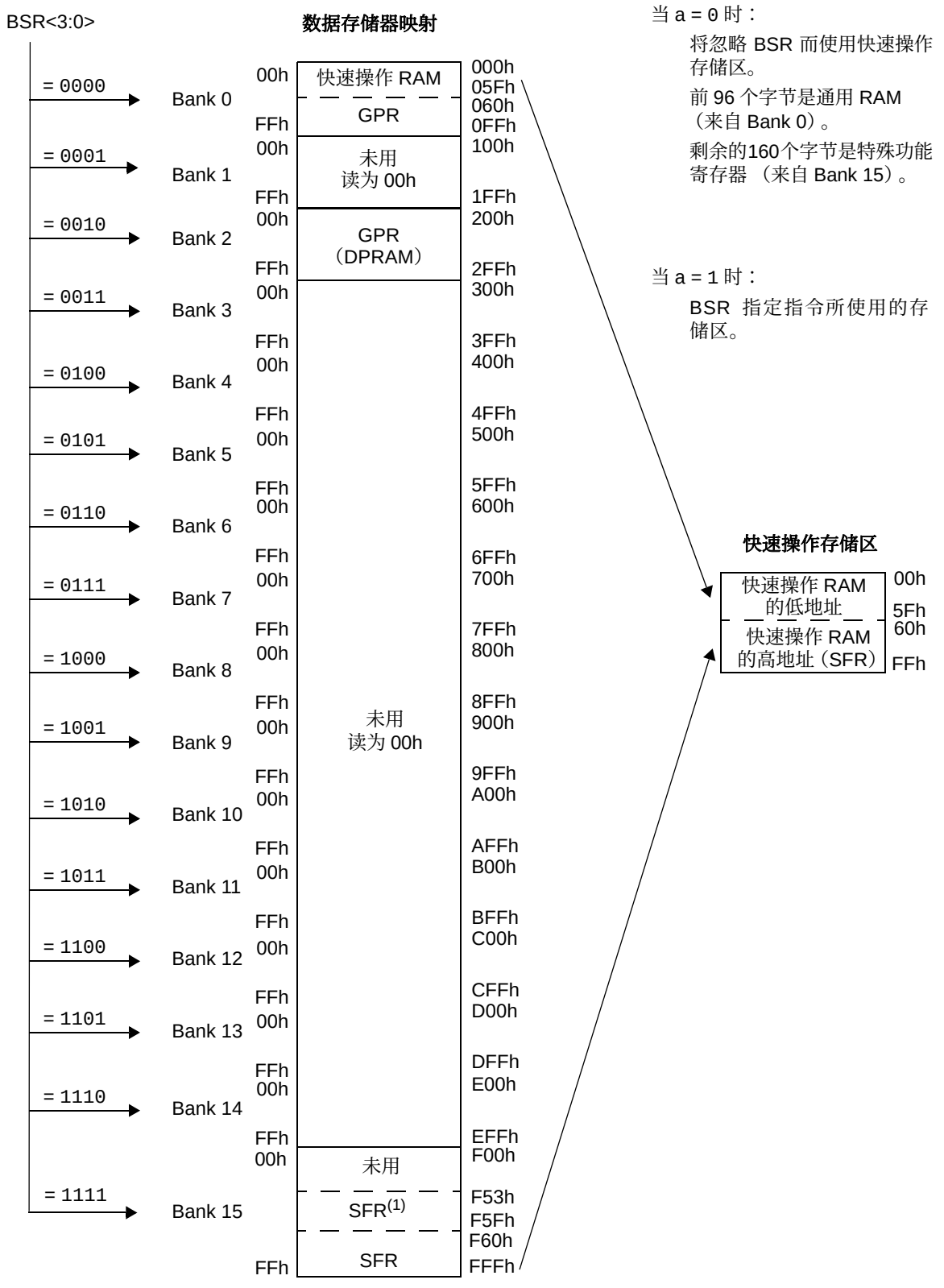
BSR 的值代表数据存储器中的存储区；指令中的 8 位指向存储区中的存储单元，可以将它看作距离存储区下边界的偏移量。图 3-5 和图 3-6 显示了 BSR 的值与数据存储器中的存储区之间的关系。

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，将导致程序计数器重新赋值。

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 3-5 和图 3-6 中的数据存储器映射指出了已实现的存储区。

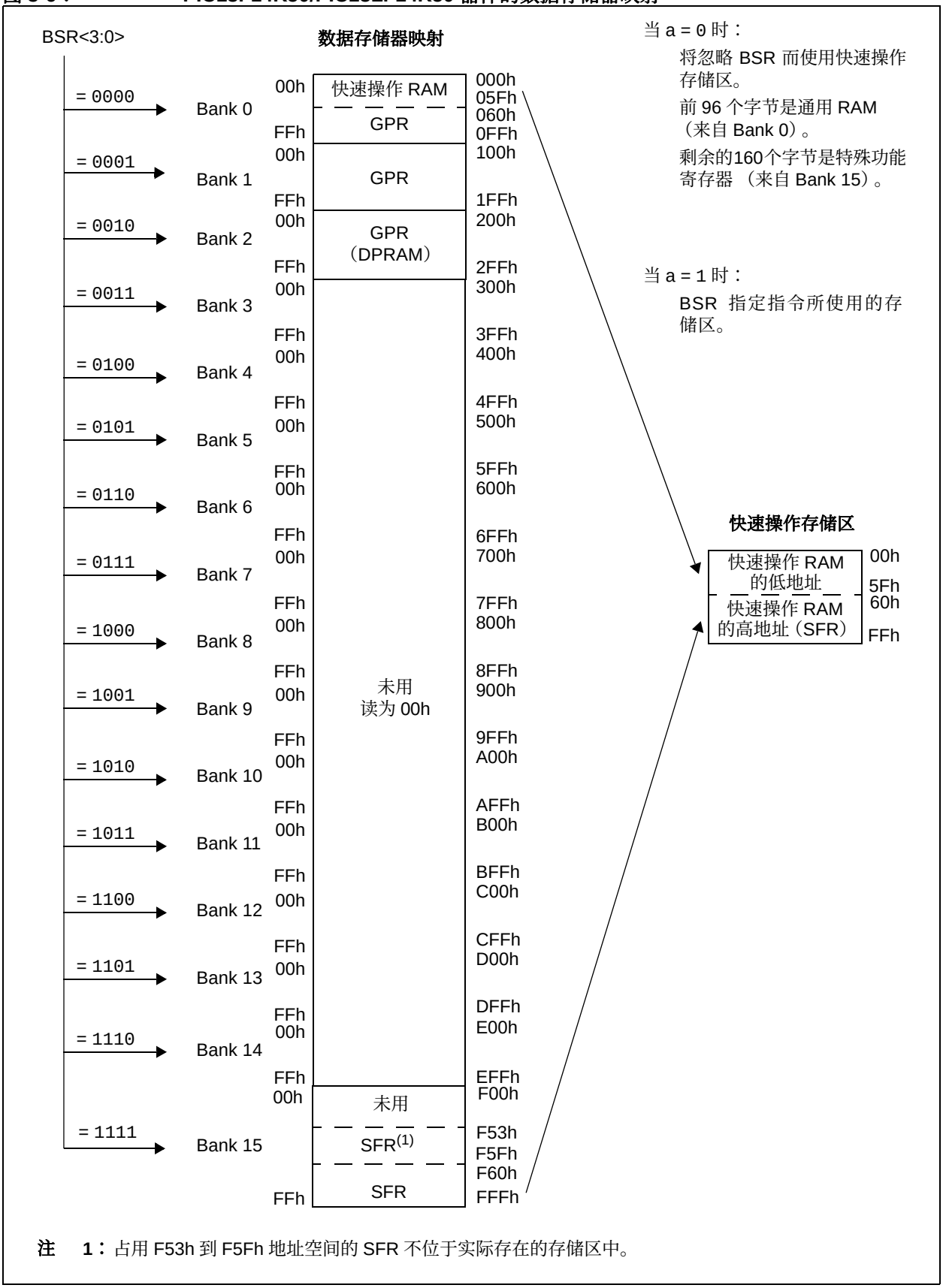
在 PIC18 的内核指令集中，只有 MOVFF 指令指定源寄存器和目标寄存器的完整 12 位地址。该指令在执行时完全忽略 BSR。所有其他指令仅包含作为操作数的低位地址，而且必须使用 BSR 或快速操作存储区来寻址目标寄存器。

图 3-5：　　PIC18F13K50/PIC18LF13K50 器件的数据存储器映射



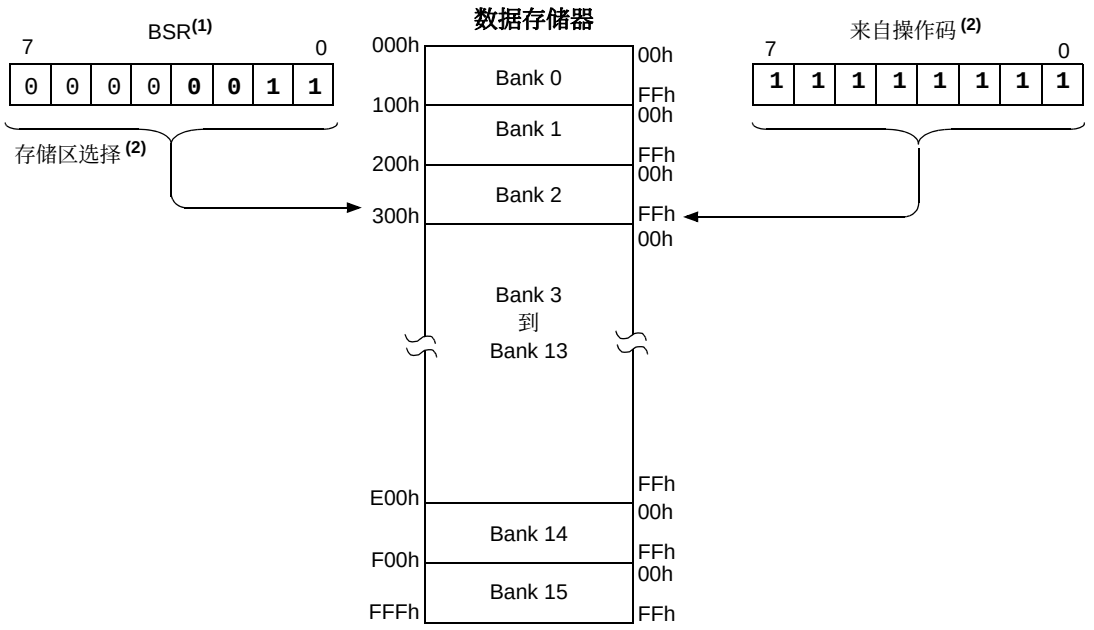
注　1： 占用 F53h 到 F5Fh 地址空间的 SFR 不位于实际存在的存储区中。

图 3-6：　　PIC18F14K50/PIC18LF14K50 器件的数据存储器映射



注　1： 占用 F53h 到 F5Fh 地址空间的 SFR 不位于实际存在的存储区中。

图 3-7：　　使用存储区选择寄存器 （直接寻址）



注　　1： 指令的快速操作 RAM 位强制跳过选定的存储区 （BSR<3:0>）直接寻址快速操作存储区的寄存器。

2： MOVFF 指令嵌入了完整的 12 位地址。

3.3.3 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确认和 / 或更改 BSR 会严重影响工作效率。

为了连续访问大多数常用数据存储单元，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 3-5 和图 3-6）。

包括快速操作 RAM 位（指令中的 “ a ” 参数）的 PIC18 内核指令在执行时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有效地对 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有效的现场保护和变量切换。

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 3.5.3 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。

3.3.4 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，所有指令均可访问它。 GPR 区从 Bank 0 的底部 （地址 000h）开始向上直到 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。

3.3.5 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 3-1 和表 3-2 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 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 通常分布在受其控制的外设中。未使用的 SFR 单元是未实现的，读为 0。

表 3-1： PIC18F/LF1XK50 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
FFFh	TOSU	FD7h	TMR0H	FAFh	SPBRG	F87h	—(2)	F5Fh	UEIR
FFEh	TOSH	FD6h	TMR0L	FAEh	RCREG	F86h	—(2)	F5Eh	UFRMH
FFDh	TOSL	FD5h	T0CON	FADh	TXREG	F85h	—(2)	F5Dh	UFRML
FFCh	STKPTR	FD4h	—(2)	FACh	TXSTA	F84h	—(2)	F5Ch	UADDR
FFBh	PCLATU	FD3h	OSCCON	FABh	RCSTA	F83h	—(2)	F5Bh	UEIE
FFAh	PCLATH	FD2h	OSCCON2	FAAh	—	F82h	PORTC	F5Ah	UEP7
FF9h	PCL	FD1h	WDTCON	FA9h	EEADR	F81h	PORTB	F59h	UEP6
FF8h	TBLPTRU	FD0h	RCON	FA8h	EEDATA	F80h	PORTA	F58h	UEP5
FF7h	TBLPTRH	FCFh	TMR1H	FA7h	EECON2(1)	F7Fh	ANSELH	F57h	UEP4
FF6h	TBLPTRL	FCEh	TMR1L	FA6h	EECON1	F7Eh	ANSEL	F56h	UEP3
FF5h	TABLAT	FCDh	T1CON	FA5h	—(2)	F7Dh	—(2)	F55h	UEP2
FF4h	PRODH	FCCh	TMR2	FA4h	—(2)	F7Ch	—(2)	F54h	UEP1
FF3h	PRODL	FCBh	PR2	FA3h	—(2)	F7Bh	—(2)	F53h	UEP0
FF2h	INTCON	FCAh	T2CON	FA2h	IPR2	F7Ah	IOCB		
FF1h	INTCON2	FC9h	SSPBUF	FA1h	PIR2	F79h	IOCA		
FF0h	INTCON3	FC8h	SSPADD	FA0h	PIE2	F78h	WPUB		
FEFh	INDF0(1)	FC7h	SSPSTAT	F9Fh	IPR1	F77h	WPUA		
FEEh	POSTINC0(1)	FC6h	SSPCON1	F9Eh	PIR1	F76h	SLRCON		
FEDh	POSTDEC0(1)	FC5h	SSPCON2	F9Dh	PIE1	F75h	—(2)		
FECh	PREINC0(1)	FC4h	ADRESH	F9Ch	—(2)	F74h	—(2)		
FEBh	PLUSW0(1)	FC3h	ADRESL	F9Bh	OSCTUNE	F73h	—(2)		
FEAh	FSR0H	FC2h	ADCON0	F9Ah	—(2)	F72h	—(2)		
FE9h	FSR0L	FC1h	ADCON1	F99h	—(2)	F71h	—(2)		
FE8h	WREG	FC0h	ADCON2	F98h	—(2)	F70h	—(2)		
FE7h	INDF1(1)	FBFh	CCPR1H	F97h	—(2)	F6Fh	SSPMASK		
FE6h	POSTINC1(1)	FBEh	CCPR1L	F96h	—(2)	F6Eh	—(2)		
FE5h	POSTDEC1(1)	FBDh	CCP1CON	F95h	—(2)	F6Dh	CM1CON0		
FE4h	PREINC1(1)	FBCh	REFCON2	F94h	TRISC	F6Ch	CM2CON1		
FE3h	PLUSW1(1)	FBBh	REFCON1	F93h	TRISB	F6Bh	CM2CON0		
FE2h	FSR1H	FBAh	REFCON0	F92h	TRISA	F6Ah	—(2)		
FE1h	FSR1L	FB9h	PSTRCON	F91h	—(2)	F69h	SRCON1		
FE0h	BSR	FB8h	BAUDCON	F90h	—(2)	F68h	SRCON0		
FDFh	INDF2(1)	FB7h	PWM1CON	F8Fh	—(2)	F67h	—(2)		
FDEh	POSTINC2(1)	FB6h	ECCP1AS	F8Eh	—(2)	F66h	—(2)		
FDDh	POSTDEC2(1)	FB5h	—(2)	F8Dh	—(2)	F65h	—(2)		
FDCh	PREINC2(1)	FB4h	—(2)	F8Ch	—(2)	F64h	UCON		
FDBh	PLUSW2(1)	FB3h	TMR3H	F8Bh	LATC	F63h	USTAT		
FDAh	FSR2H	FB2h	TMR3L	F8Ah	LATB	F62h	UIR		
FD9h	FSR2L	FB1h	T3CON	F89h	LATA	F61h	UCFG		
FD8h	STATUS	FB0h	SPBRGH	F88h	—(2)	F60h	UIE		

注 **1：** 这不是实际存在的寄存器。
2： 未实现的寄存器读为 0。

表 3-2： 寄存器文件汇总 （PIC18F/LF1XK50）

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	—	—	—	栈顶寄存器最高字节 （TOS<20:16>）					---0 0000	285, 30
TOSH	栈顶寄存器高字节 （TOS<15:8>）								0000 0000	285, 30
TOSL	栈顶寄存器低字节 （TOS<7:0>）								0000 0000	285, 30
STKPTR	STKFUL	STKUNF	—	SP4	SP3	SP2	SP1	SP0	00-0 0000	285, 31
PCLATU	—	—	—	PC<20:16> 的保持寄存器					---0 0000	285, 30
PCLATH	PC<15:8> 的保持寄存器								0000 0000	285, 30
PCL	PC 低字节 （PC<7:0>）								0000 0000	285, 30
TBLPTRU	—	—	—	程序存储器表指针最高字节 （TBLPTR<20:16>）					---0 0000	285, 54
TBLPTRH	程序存储器表指针高字节 （TBLPTR<15:8>）								0000 0000	285, 54
TBLPTRL	程序存储器表指针低字节 （TBLPTR<7:0>）								0000 0000	285, 54
TABLAT	程序存储器表锁存器								0000 0000	285, 54
PRODH	乘积寄存器的高字节								xxxx xxxx	285, 65
PRODL	乘积寄存器的低字节								xxxx xxxx	285, 65
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	0000 000x	285, 70
INTCON2	$\overline{\text{RABPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RABIP	1111 -1-1	285, 71
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00	285, 72
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	285, 47
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	285, 47
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	285, 47
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	285, 47
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 的偏移量由 W 寄存器提供								N/A	285, 47
FSR0H	—	—	—	—	间接数据存储器地址指针 0 的高字节				---- 0000	285, 47
FSR0L	间接数据存储器地址指针 0 的低字节								xxxx xxxx	285, 47
WREG	工作寄存器								xxxx xxxx	285
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	285, 47
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	285, 47
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	285, 47
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	285, 47
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 的偏移量由 W 寄存器提供								N/A	285, 47
FSR1H	—	—	—	—	间接数据存储器地址指针 1 的高字节				---- 0000	286, 47
FSR1L	间接数据存储器地址指针 1 的低字节								xxxx xxxx	286, 47
BSR	—	—	—	—	存储区选择寄存器				---- 0000	286, 35
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	286, 47
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	286, 47
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	286, 47
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	286, 47
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 的偏移量由 W 寄存器提供								N/A	286, 47
FSR2H	—	—	—	—	间接数据存储器地址指针 2 的高字节				---- 0000	286, 47
FSR2L	间接数据存储器地址指针 2 的低字节								xxxx xxxx	286, 47
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx	286, 45

图注： x = 未知， u = 不变， — = 未实现， q = 值取决于具体条件

注 1： SBOREN 位仅在 BOREN<1:0> 配置位 = 01 时可用；否则，它被禁止且读为 0。请参见**第 23.4 节 “ 欠压复位 （BOR）”**。

2： RA3 位仅在主复位被禁止 （MCLRE 配置位 = 0）时可用。否则， RA3 读为 0。该位是只读的。

3： RA0 和 RA1 位仅在 USB 被禁止时可用。

表 3-2： 寄存器文件汇总 （PIC18F/LF1XK50）（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 寄存器的高字节								0000 0000	286, 103
TMR0L	Timer0 寄存器的低字节								xxxx xxxx	286, 103
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	286, 101
OSCCON	IDLEN	IRCF2	IRCF1	IRCF0	OSTS	IOSF	SCS1	SCS0	0011 qq00	286, 20
OSCCON2	—	—	—	—	—	PRI_SD	HFIOFL	LFIOFS	---- -10x	286, 21
WDTCON	—	—	—	—	—	—	—	SWDTEN	--- ---0	286, 303
RCON	IPEN	SBOREN(1)	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	0q-1 11q0	277, 284, 79
TMR1H	Timer1 寄存器的高字节								xxxx xxxx	286, 110
TMR1L	Timer1 寄存器的低字节								xxxx xxxx	286, 110
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	0000 0000	286, 105
TMR2	Timer2 寄存器								0000 0000	286, 112
PR2	Timer2 周期寄存器								1111 1111	286, 112
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	286, 111
SSPBUF	SSP 接收缓冲区 / 发送寄存器								xxxx xxxx	286, 147
SSPADD	I^2C™ 从模式下的 SSP 地址寄存器。 I^2C 主模式下的 SSP 波特率重载寄存器。								0000 0000	286, 144
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	286, 147
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	286, 147
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	286, 180
ADRESH	A/D 结果寄存器的高字节								xxxx xxxx	287, 221
ADRESL	A/D 结果寄存器的低字节								xxxx xxxx	287, 221
ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	ADON	--00 0000	287, 215
ADCON1	—	—	—	—	PVCFG1	PVCFG0	NVCFG1	NVCFG0	---- 0000	287, 216
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	0-00 0000	287, 217
CCPR1H	捕捉 / 比较 /PWM 寄存器 1 的高字节								xxxx xxxx	287, 138
CCPR1L	捕捉 / 比较 /PWM 寄存器 1 的低字节								xxxx xxxx	287, 138
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	0000 0000	287, 117
REFCON2	—	—	—	DAC1R4	DAC1R3	DAC1R2	DAC1R1	DAC1R0	---0 0000	287, 248
REFCON1	D1EN	D1LPS	DAC1OE	—	D1PSS1	D1PSS0	—	D1NSS	000- 00-0	287, 248
REFCON0	FVR1EN	FVR1ST	FVR1S1	FVR1S0	—	—	—	—	0001 00--	287, 247
PSTRCON	—	—	—	STRSYNC	STRD	STRC	STRB	STRA	---0 0001	287, 134
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	0100 0-00	287, 192
PWM1CON	PRSEN	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0	0000 0000	287, 133
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0	0000 0000	287, 129
TMR3H	Timer3 寄存器的高字节								xxxx xxxx	287, 115
TMR3L	Timer3 寄存器的低字节								xxxx xxxx	287, 115
T3CON	RD16	—	T3CKPS1	T3CKPS0	T3CCP1	$\overline{T3SYNC}$	TMR3CS	TMR3ON	0-00 0000	287, 113

图注： x = 未知， u = 不变， — = 未实现， q = 值取决于具体条件

注 1： SBOREN 位仅在 BOREN<1:0> 配置位 = 01 时可用；否则，它被禁止且读为 0。请参见**第 23.4 节 “ 欠压复位 （BOR）”**。

2： RA3 位仅在主复位被禁止 （MCLRE 配置位 = 0）时可用。否则， RA3 读为 0。该位是只读的。

3： RA0 和 RA1 位仅在 USB 被禁止时可用。

表 3-2： 寄存器文件汇总 （PIC18F/LF1XK50）（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）：
SPBRGH	EUSART 波特率发生器寄存器的高字节								0000 0000	287, 181
SPBRG	EUSART 波特率发生器寄存器的低字节								0000 0000	287, 181
RCREG	EUSART 接收寄存器								0000 0000	287, 182
TXREG	EUSART 发送寄存器								0000 0000	287, 181
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	287, 190
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	287, 191
EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	0000 0000	287, 52, 61
EEDATA	EEPROM 数据寄存器								0000 0000	287, 52, 61
EECON2	EEPROM 控制寄存器 2 （不是实际存在的寄存器）								0000 0000	287, 52, 61
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	287, 53, 61
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	—	1111 111-	288, 78
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	0000 000-	288, 74
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	0000 000-	288, 76
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	-111 1111	288, 77
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	-000 0000	288, 73
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	-000 0000	288, 75
OSCTUNE	INTSRC	SPLLEN	TUN5	TUN4	TUN3	TUN2	TUN1	TUN0	0000 0000	22, 288
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	1111 1111	288, 94
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	1111 ----	288, 89
TRISA	—	—	TRISA5	TRISA4	—	—	—	—	--11 ----	288, 83
LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	xxxx xxxx	288, 94
LATB	LATB7	LATB6	LATB5	LATB4	—	—	—	—	xxxx ----	288, 89
LATA	—	—	LATA5	LATA4	—	—	—	—	--xx ----	288, 83
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	288, 94
PORTB	RB7	RB6	RB5	RB4	—	—	—	—	xxxx ----	288, 89
PORTA	—	—	RA5	RA4	RA3(2)	—	RA1(3)	RA0(3)	--xx x-xx	288, 83
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	---- 1111	288, 99
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	—	—	—	1111 1---	288, 98
IOCB	IOCB7	IOCB6	IOCB5	IOCB4	—	—	—	—	0000 ----	288, 93
IOCA	—	—	IOCA5	IOCA4	IOCA3	—	IOCA1	IOCA0	--00 0-00	288, 83
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	—	—	—	—	1111 ----	288, 89
WPUA	—	—	WPUA5	WPUA4	WPUA3	—	—	—	--11 1---	285, 89
SLRCON	—	—	—	—	—	SLRC	SLRB	SLRA	---- -111	288, 100
SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	1111 1111	288, 154
CM1CON0	C1ON	C1OUT	C1OE	C1POL	C1SP	C1R	C1CH1	C1CH0	0000 1000	288, 229
CM2CON1	MC1OUT	MC2OUT	C1RSEL	C2RSEL	C1HYS	C2HYS	C1SYNC	C2SYNC	0000 0000	288, 230
CM2CON0	C2ON	C2OUT	C2OE	C2POL	C2SP	C2R	C2CH1	C2CH0	0000 1000	288, 230
SRCON1	SRSPE	SRSCKE	SRSC2E	SRSC1E	SRRPE	SRRCKE	SRRC2E	SRRC1E	0000 0000	288, 243
SRCON0	SRLEN	SRCLK2	SRCLK1	SRCLK0	SRQEN	SRNQEN	SRPS	SRPR	0000 0000	288, 242
UCON	—	PPBRST	SE0	PKTDIS	USBEN	RESUME	SUSPND	—	-0x0 000-	288, 252

图注： x = 未知， u = 不变， — = 未实现， q = 值取决于具体条件

注 1： SBOREN 位仅在 BOREN<1:0> 配置位 = 01 时可用；否则，它被禁止且读为 0。请参见**第 23.4 节 “ 欠压复位 （BOR）”**。

2： RA3 位仅在主复位被禁止 （MCLRE 配置位 = 0）时可用。否则， RA3 读为 0。该位是只读的。

3： RA0 和 RA1 位仅在 USB 被禁止时可用。

表 3-2：　寄存器文件汇总 （PIC18F/LF1XK50）（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）：
USTAT	—	ENDP3	ENDP2	ENDP1	ENDP0	DIR	PPBI	—	-xxx xxx-	289, 256
UIR	—	SOFIF	STALLIF	IDLEIF	TRNIF	ACTVIF	UERRIF	URSTIF	-000 0000	289, 266
UCFG	UTEYE	—	—	UPUEN	—	FSEN	PPB1	PPB0	0--0 -000	289, 254
UIE	—	SOFIE	STALLIE	IDLEIE	TRNIE	ACTVIE	UERRIE	URSTIE	-000 0000	289, 268
UEIR	BTSEF	—	—	BTOEF	DFN8EF	CRC16EF	CRC5EF	PIDEF	0--0 0000	289, 269
UFRMH	—	—	—	—	—	FRM10	FRM9	FRM8	---- -xxx	289, 252
UFRML	FRM7	FRM6	FRM5	FRM4	FRM3	FRM2	FRM1	FRM0	xxxx xxxx	289, 252
UADDR	—	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0	-000 0000	289, 258
UEIE	BTSEE	—	—	BTOEE	DFN8EE	CRC16EE	CRC5EE	PIDEE	0--0 0000	289, 270
UEP7	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	289, 257
UEP6	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	289, 257
UEP5	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	289, 257
UEP4	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	289, 257
UEP3	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	289, 257
UEP2	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	289, 257
UEP1	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	289, 257
UEP0	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	---0 0000	285, 257

图注： x = 未知， u = 不变， — = 未实现， q = 值取决于具体条件

注　1： SBOREN 位仅在 BOREN<1:0> 配置位 = 01 时可用；否则，它被禁止且读为 0。请参见**第 23.4 节 “ 欠压复位 （BOR）”**。

2： RA3 位仅在主复位被禁止 （MCLRE 配置位 = 0）时可用。否则， RA3 读为 0。该位是只读的。

3： RA0 和 RA1 位仅在 USB 被禁止时可用。

3.3.6 状态寄存器

如寄存器 3-2 所示，STATUS 寄存器包含 ALU 的算术运算状态。和任何其他 SFR 一样，它也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，将不会直接写入结果，而是根据指令的执行更新 STATUS 寄存器。因此，当执行一条把 STATUS 寄存器作为目标寄存器的指令后，执行结果可能与预想的不同。例如， CLRF STATUS 将 Z 位置 1 并保持其余状态位不变 （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 25-2 和表 25-3 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 3-2： STATUS：状态寄存器

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(1)
bit 7							bit 0

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

bit 7-5 **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）。它可以表示结果是否为负 （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）。表明运算结果溢出了 7 位二进制数的范围，溢出导致结果的符号位 （bit 7）发生改变。
1 = 有符号算术运算中发生溢出 （本次运算）
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 （ADDWF、 ADDLW、 SUBLW 和 SUBWF 指令） (1)
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 （ADDWF、 ADDLW、 SUBLW 和 SUBWF 指令） (1)
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最高位或最低位。

3.4 数据寻址模式

注： 当使能 PIC18 扩展指令集时，PIC18 内核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 3.5 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 3.5.1 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。

3.4.1 固有和立即数寻址

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 寄存器添加或移入立即数值。其他立即数寻址指令，例如 CALL 和 GOTO，它们包括 20 位的程序存储器地址。

3.4.2 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 PIC18 内核指令集中，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 3.3.4 节“通用寄存器文件”**）或快速操作存储区（**第 3.3.3 节“快速操作存储区”**）中的单元地址作为指令数据源。

快速操作 RAM 位“a”决定地址的解析方式。当“a”为 1 时，BSR（**第 3.3.2 节“存储区选择寄存器（BSR）”**）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。

3.4.3 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）作为指向被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，因此也可在程序中直接控制它们。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，从而自动控制指针的值。它通过使用循环提高代码执行效率，如例 3-5 所示的清零整个 RAM 存储区的操作。

例 3-5：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
```

3.4.3.1 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。每对 FSR 保存一个 12 位值，因此 FSRnH 寄存器的高 4 位未使用。 12 位 FSR 值可以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空间而不是通过物理方式实现的。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。

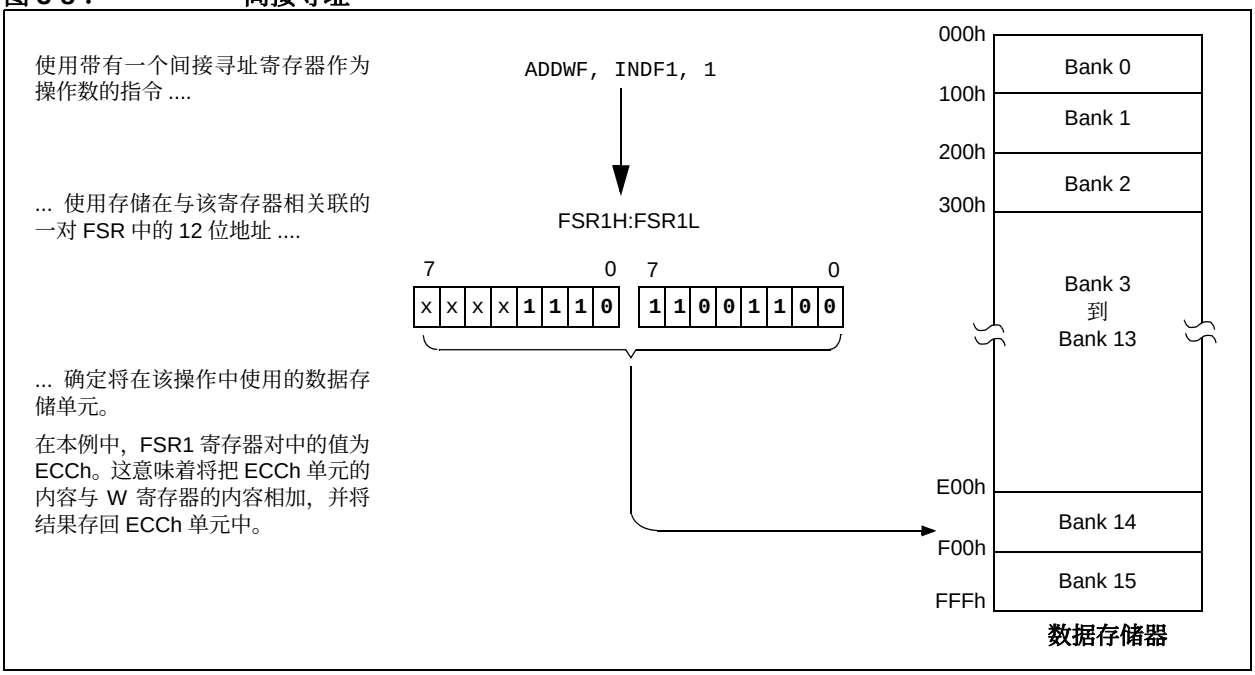
3.4.3.2 FSR 寄存器和 POSTINC、POSTDEC、 PREINC 以及 PLUSW

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 寄存器所指向的地址单元，并对 FSR 值进行特定的操作。这些位的功能分别是：

- POSTDEC：访问 FSR 指向的地址单元，然后将 FSR 的值自动减 1
- POSTINC：访问 FSR 指向的地址单元，然后将 FSR 的值自动加 1
- PREINC：将 FSR 的值自动加 1，然后在操作中使用 FSR 指向的地址单元
- PLUSW：将 W 寄存器中有符号的值 （从 -127 到 128）与 FSR 中的值相加，并在操作中使用这一加法的结果指向的地址单元

在本文中使用相关 FSR 寄存器中的值（不会更改此值）访问 INDF 寄存器。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。

图 3-8： 间接寻址



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 到 00h 溢出并向 FSRnH 寄存器进位。但这些操作的结果不会更改 STATUS 寄存器中的标志位（如 Z、N 和 OV 等）。

PLUSW 寄存器可用于在数据存储空间实现变址寻址。通过控制 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。

3.4.3.3 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 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 寄存器时会把同样的值写入 FSR2H:FSR2L。

由于 FSR 是映射到 SFR 空间中的物理寄存器，所以可以通过直接寻址来控制它们。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。

3.5 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 PIC18 内核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。

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。PIC18内核指令也仍然以直接和间接寻址模式进行操作；固有和立即数指令操作照旧。 FSR0 和 FSR1 的间接寻址模式也保持不变。

3.5.1 使用立即数偏移量进行变址寻址

使能 PIC18 扩展指令集将更改快速操作 RAM 中使用 FSR2 寄存器对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。这种特定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。

3.5.2 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PIC18内核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使用快速操作存储区 （快速操作 RAM 位为 1）或包含 60h 及以上的文件地址，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 3-9 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 25.2.1 节 “扩展指令的语法”**中对此进行了更详细的说明。

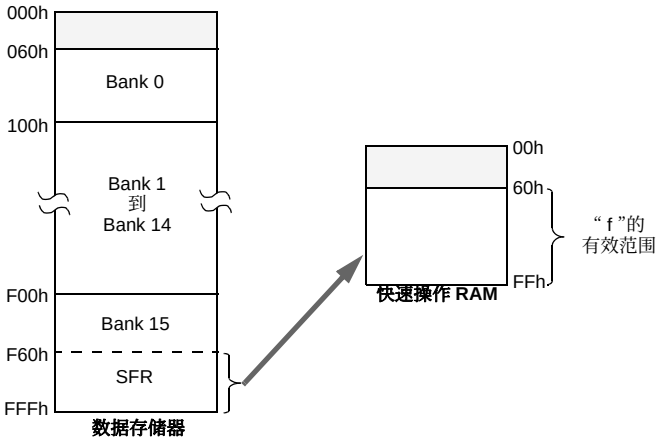
图 3-9：　　针对位和针对字节的指令的寻址模式对比 （使能了扩展指令集）

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 0FFh 之间的单元地址，该地址也是数据存储器的 F60h 到 FFFh （Bank 15）。

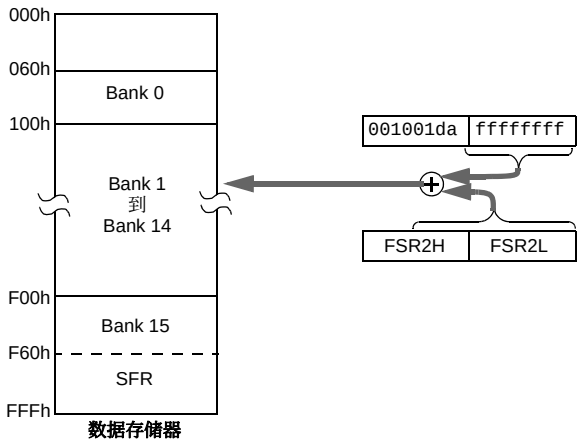
不可用此模式寻址地址低于 60h 的单元。



当 a = 0 且 f ≤ 5Fh 时：

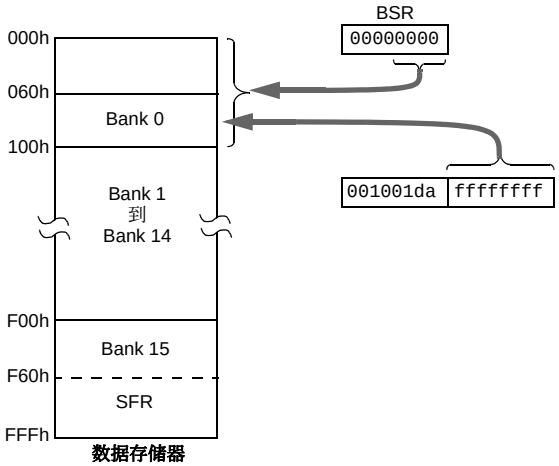
此指令以立即数变址寻址模式执行。“f”被解析为 FSR2 中地址值的偏移量。将这两个值相加可以得到指令的目标寄存器的地址。此地址可以在数据存储空间的任何地方。

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。



3.5.3 在立即数变址模式下映射快速操作存储区

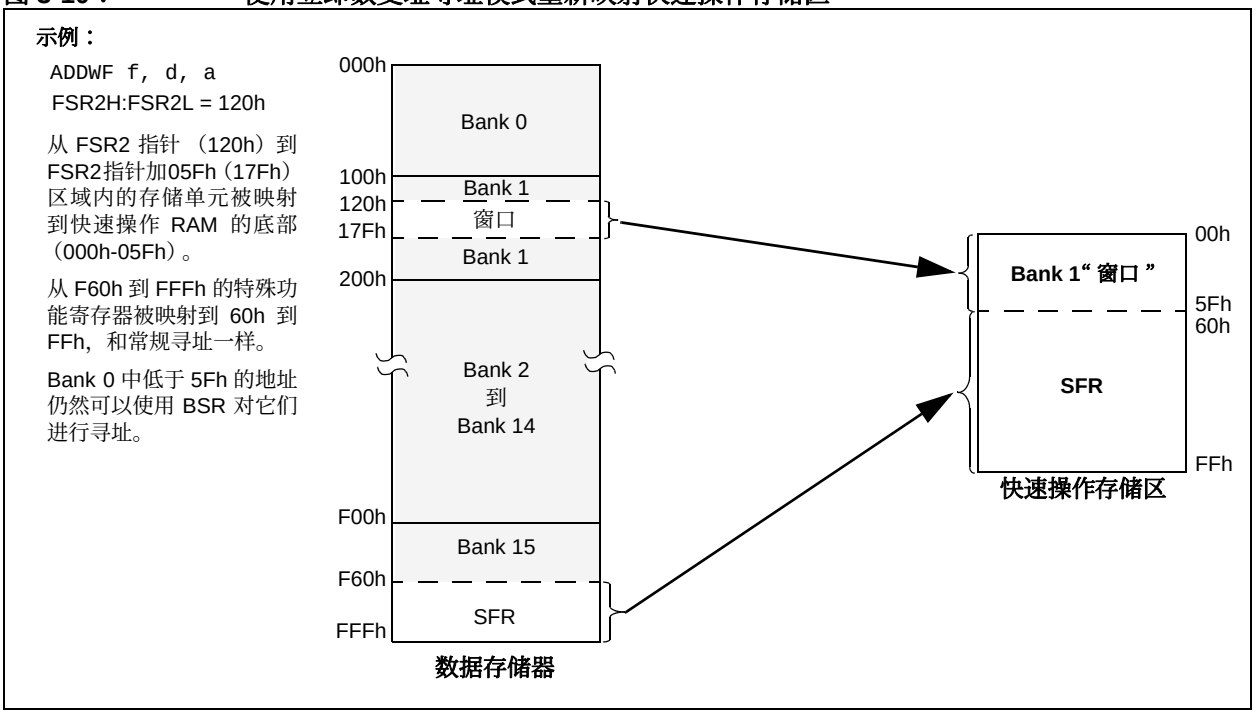
使用立即数变址寻址模式能有效改变快速操作 RAM 前 96 个地址单元（00h 到 5Fh）的映射方式。此模式映射由用户定义的、可位于数据存储空间中任何地方的“窗口”内容，而不仅仅映射 Bank 0 底部的内容。FSR2 的值定义映射到窗口的地址的下边界，而上边界则由 FSR2 加 95（5Fh）决定。地址为 5Fh 以上的快速操作 RAM 的映射方法如前所述（见**第 3.3.3 节“快速操作存储区”**）。图 3-10 显示了在此寻址模式下重新映射的快速操作存储区示例。

快速操作存储区的重新映射*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。

3.6 PIC18 指令执行和扩展指令集

使能扩展指令集将 8 条其他命令添加到现有的 PIC18 指令集中。这些指令如**第 25.2 节“扩展指令集”**中所述执行。

图 3-10：　使用立即数变址寻址模式重新映射快速操作存储区



4.0 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，根据具体的器件，每次写一个 16 或 8 字节的块（见表 4-1）。擦除程序存储器时，每次擦除一个 64 字节的块。由于写和擦除块大小的差别，所以需要 1 至 8 次块写操作来恢复单次块擦除操作的内容。用户代码不能执行批量擦除操作。

表 4-1： 写 / 擦除操作块大小

器件	写操作块大小（字节）	擦除操作块大小（字节）
PIC18F13K50	8	64
PIC18F14K50	16	64

在擦写程序存储器时，系统会停止取指令直到操作完成。擦写期间不能访问该程序存储器，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。

4.1 表读与表写

为了读写程序存储器，有两条操作指令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）

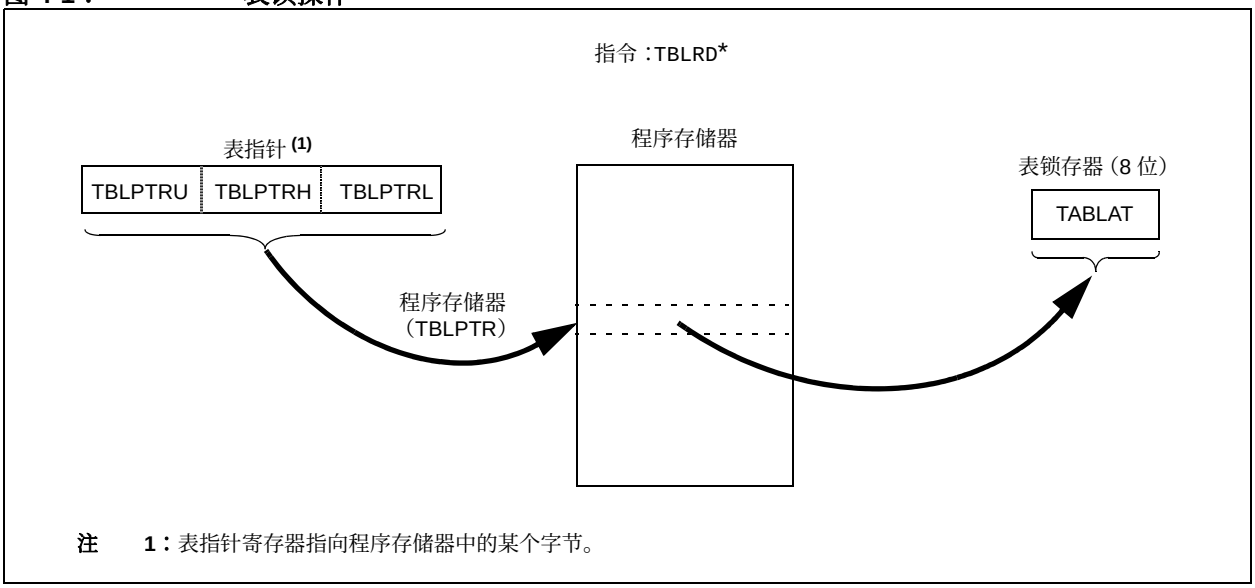
程序存储空间为 16 位宽，而数据 RAM 空间为 8 位宽。表读和表写操作通过一个 8 位寄存器 （TABLAT）在这两个存储空间之间传送数据。

表读操作仅从程序存储器直接取一个字节的数据，然后将其放入 TABLAT 寄存器。图 4-1 显示了表读操作。

表写操作将一个字节的数据从 TABLAT 寄存器存储到写操作块保持寄存器中。**第 4.5 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 4-2 显示了程序存储器和数据 RAM 之间的一次表写操作。

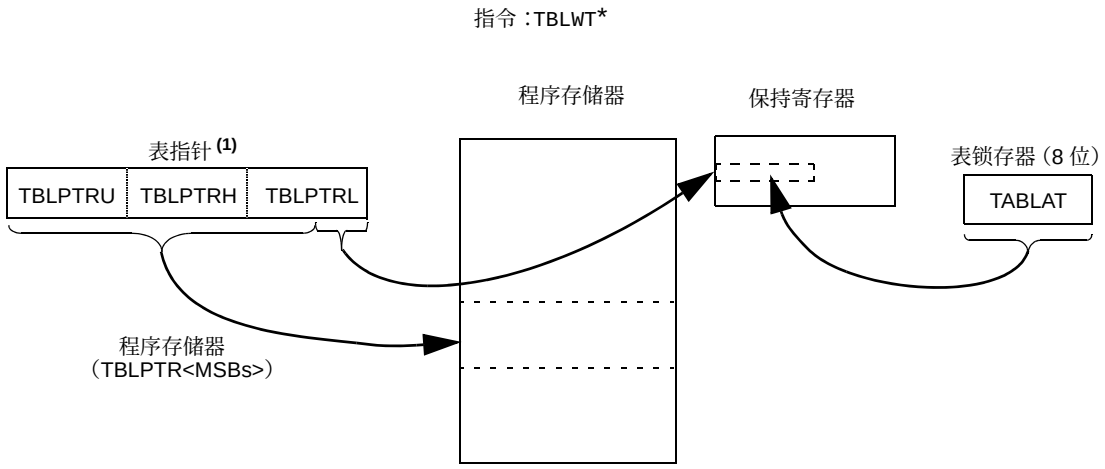
表操作以字节为单位。包含数据而非程序指令的表不需要按字对齐。因此，表可以在任何字节地址开始和结束。如果使用表写操作向程序存储器写入可执行代码，程序指令必须按字对齐。

图 4-1： 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 4-2：　　表写操作



注　1：在表写操作期间，表指针不直接指向程序存储器。 TBLPRTL 的 LSB 实际指向写操作块保持寄存器中的一个地址。表指针的 MSB 决定最终写操作块将被写入什么位置。**第 4.5 节 “ 写闪存程序存储器 ”** 中将讨论将保持寄存器的内容写入程序存储器阵列的过程。

4.2　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

4.2.1　EECON1 和 EECON2 寄存器

EECON1 寄存器（寄存器 4-1）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。

EEPGD 控制位决定访问的是程序存储器还是数据 EEPROM 存储器。当 EEPGD 清零时，任何后续操作将针对数据 EEPROM 存储器进行。当 EEPGD 置 1 时，任何后续操作都将针对程序存储器进行。

CFGS 控制位决定访问的是配置 / 校准寄存器还是程序存储器 / 数据 EEPROM 存储器。当 CFGS 置 1 时，不管 EEPGD 的值如何，后续操作将针对配置寄存器进行（见**第 24.0 节 “ CPU 的特殊功能 ”**）。当 CFGS 清零时，则由 EEPGD 来选择访问的存储器。

FREE 位允许进行程序存储器的擦除操作。当 FREE 置 1 时，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。 WREN 位在上电时被清零。

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。用固件只能将 WR 位置 1 而无法清零。写操作完成时，由硬件将 WR 位清零。

注：	当写操作完成时， PIR2 寄存器的 EEIF 中断标志位被置 1。EEIF 标志保持置 1 状态直到被固件清零为止。

寄存器 4-1：　　EECON1：数据 EEPROM 控制寄存器 1

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

图注：			
R = 可读位	W = 可写位		
S = 该位可用软件置 1，但不能清零		U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 7　**EEPGD：**闪存程序存储器或数据 EEPROM 存储器选择位
1 = 访问闪存程序存储器
0 = 访问数据 EEPROM 存储器

bit 6　**CFGS：**闪存程序存储器 / 数据 EEPROM 存储器或配置寄存器选择位
1 = 访问配置寄存器
0 = 访问闪存程序存储器或数据 EEPROM 存储器

bit 5　**未实现：**读为 0

bit 4　**FREE：**闪存行 （块）擦除使能位
1 = 在下一条 WR 命令时擦除 TBLPTR 指定的程序存储器块 （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 / 数据 EEPROM 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 / 数据 EEPROM 存储器写使能位
1 = 允许对闪存程序存储器 / 数据 EEPROM 存储器的写周期
0 = 禁止对闪存程序存储器 / 数据 EEPROM 存储器的写周期

bit 1　**WR：**写控制位
1 = 启动数据 EEPROM 擦除 / 写周期或程序存储器的擦除周期或写周期。
（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，但不能清零。）
0 = EEPROM 写周期完成

bit 0　**RD：**读控制位
1 = 启动 EEPROM 读操作（读操作需要一个指令周期。RD 由硬件清零。用软件只能将 RD 位置 1，但不能清零。 EEPGD = 1 或 CFGS = 1 时， RD 位无法置 1。）
0 = 不启动 EEPROM 读操作

注　1： 当发生 WRERR 时， EEPGD 和 CFGS 位不会被清零。这样可以跟踪错误情况。

4.2.2 TABLAT—— 表锁存寄存器

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。

4.2.3 TBLPTR—— 表指针寄存器

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 位允许器件寻址最高 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 4-2 列出了这些操作。这些操作只会影响 TBLPTR 的低 21 位。

4.2.4 表指针范围

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直接读入 TABLAT 寄存器。

当执行 TBLWT 时，TABLAT 寄存器中的字节将写入保持寄存器（而不是闪存），以准备进行程序存储器写操作。保持寄存器包含一个写操作块，写操作块因器件而异（见表 4-1）。 TBLPTRL 寄存器的 3、 4 或 5 LSb 决定数据写入保持寄存器块中的哪个具体地址。在 TBLWT 操作期间，表指针的 MSB 没有作用。

当执行程序存储器写操作时，整个保持寄存器块的内容写入到闪存中，写入地址由 TBLPTR 的 MSb 决定。在闪存写操作期间， 3、 4 或 5 LSB 被忽略。更多详细信息，请参见**第 4.5 节 “ 写闪存程序存储器 ”**。

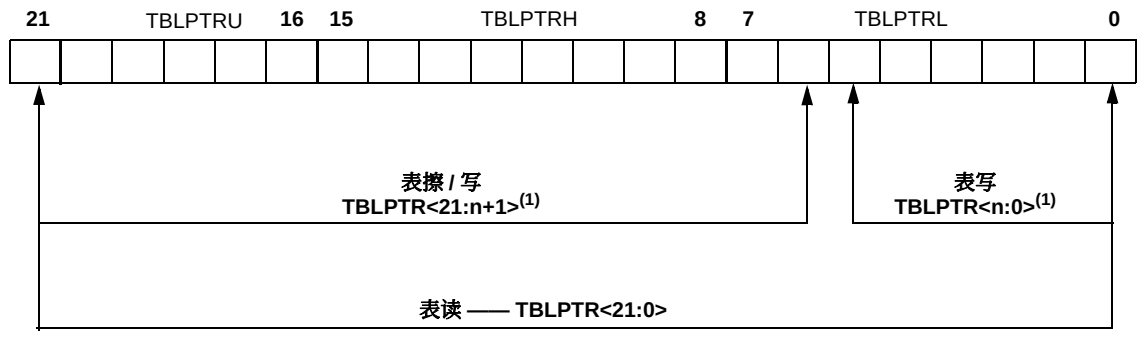
当执行擦除程序存储器时，表指针寄存器的高 16 位（TBLPTR<21:6>）指向将要擦除的 64 字节块。低有效位 （TBLPTR<5:0>）被忽略。

图 4-3 说明了基于闪存程序存储器操作的 TBLPTR 相关范围。

表 4-2： 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 4-3： 基于操作的表指针范围



注 1： n = 3、 4、 5 或 6，分别对应于 8、 16、 32 或 64 字节的块大小。

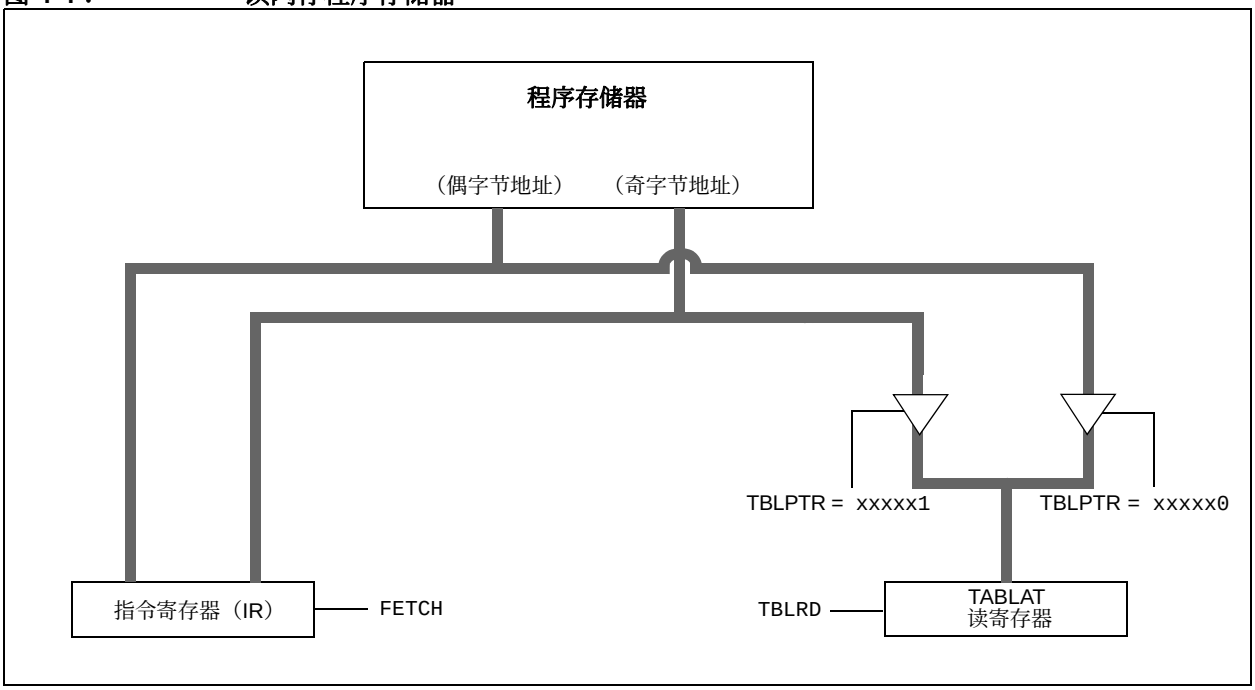
4.3 读闪存程序存储器

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 4-4 显示了内部程序存储器和 TABLAT 之间的接口。

图 4-4： 读闪存程序存储器



例 4-1： 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
            MOVFW     TABLAT, W                ; get data
            MOVF      WORD_ODD
```

4.4 擦除闪存程序存储器

最小擦除块大小为 32 个字或 64 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。

当单片机本身启动一个擦除序列时，会擦除一个 64 字节的程序存储器块。高 16 位 TBLPTR<21:6> 指向要擦除的块。 TBLPTR<5:0> 被忽略。

擦除操作由 EECON1 寄存器控制。必须将 EEPGD 位置 1 以指向闪存程序存储器。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。

EECON2 的写操作启动序列 （在**第 4.4.1 节 “ 闪存程序存储器擦除序列 ”**中显示为步骤 4 至 6）用于防止意外的写操作。这有时称为长写操作。

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。

4.4.1 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. 将要擦除的块地址装入表指针寄存器。
2. 设置 EECON1 寄存器来执行擦除操作：
   - 将 EEPGD 位置 1 以指向程序存储器；
   - 将 CFGS 位清零以访问程序存储器；
   - 将 WREN 位置 1 以使能写操作；
   - 将 FREE 位置 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。这将开始块擦除周期。
7. CPU 在擦除期间将会停止工作（使用内部定时器约为 2 ms）。
8. 重新允许中断。

例 4-2： 擦除闪存程序存储器块

```
                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_BLOCK
                BSF     EECON1, EEPGD           ; point to Flash program memory
                BCF     EECON1, CFGS            ; access Flash program memory
                BSF     EECON1, WREN            ; enable write to memory
                BSF     EECON1, FREE            ; enable block Erase operation
                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 erase (CPU stall)
                BSF     INTCON, GIE             ; re-enable interrupts
```

4.5 写闪存程序存储器

编程块大小为 8 或 16 字节，这取决于器件（见表 4-1）。不支持字或字节编程。

在内部使用表写命令将需要写入闪存的内容装入保持寄存器中。保持寄存器的数量与写操作块中的字节数相同（见表 4-1）。

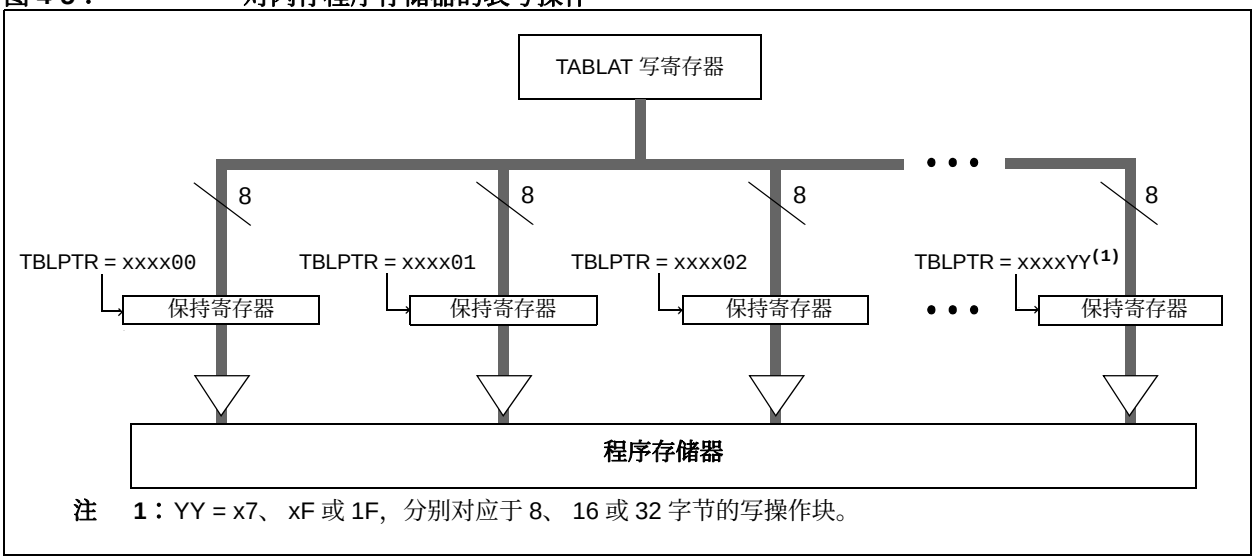
由于表锁存器 （TABLAT）只是单字节寄存器，所以每次编程操作 TBLWT 指令可能需要执行 8 或 16 次（取决于具体器件）。因为只写保持寄存器，所以所有的表写操作实际上都是短写。在数据写入所有保持寄存器之后，该存储器块的编程操作通过以下操作启动：先配置 EECON1 寄存器来进行程序存储器写操作，然后执行长写序列。

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 EEPROM 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，该电荷泵可以工作在器件的电压范围内。

注： 器件复位和写操作完成后保持寄存器的默认值为 FFh。将 FFh 写入保持寄存器不会修改其中的数值。这意味着可以修改程序存储器的各字节，但前提是不能将任何位从 0 更改为 1。当修改各字节时，无需在执行长写操作前装入所有保持寄存器。

图 4-5： 对闪存程序存储器的表写操作



注 1： YY = x7、 xF 或 1F，分别对应于 8、 16 或 32 字节的写操作块。

4.5.1 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 64 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. 执行块擦除过程。
5. 将要写入的第一个字节的地址装入表指针寄存器。
6. 通过自动递增将 8 或 16 个字节块写入保持寄存器。
7. 设置 EECON1 寄存器来执行写操作：
   - 将 EEPGD 位置 1 以指向程序存储器；
   - 将 CFGS 位清零以访问程序存储器；
   - 将 WREN 位置 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 在写入期间将会停止工作 （使用内部定时器约为 2 ms）。
13. 重新允许中断。
14. 对每个块重复步骤 6 至 13，直到全部 64 个字节都已写入。
15. 校验存储器 （表读）。

该过程需要大约 6 ms 的时间来更新存储器的每个写操作块。例 4-3 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字节的预期地址范围内。

例 4-3：　　写闪存程序存储器

```
                    MOVLW     D'64'                    ; number of bytes in erase block
                    MOVWF     COUNTER
                    MOVLW     BUFFER_ADDR_HIGH         ; point to buffer
                    MOVWF     FSR0H
                    MOVLW     BUFFER_ADDR_LOW
                    MOVWF     FSR0L
                    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
READ_BLOCK
                    TBLRD*+                            ; read into TABLAT, and inc
                    MOVF      TABLAT, W                ; get data
                    MOVWF     POSTINC0                 ; store data
                    DECFSZ    COUNTER                  ; done?
                    BRA       READ_BLOCK               ; repeat
MODIFY_WORD
                    MOVLW     BUFFER_ADDR_HIGH         ; point to buffer
                    MOVWF     FSR0H
                    MOVLW     BUFFER_ADDR_LOW
                    MOVWF     FSR0L
                    MOVLW     NEW_DATA_LOW             ; update buffer word
                    MOVWF     POSTINC0
                    MOVLW     NEW_DATA_HIGH
                    MOVWF     INDF0
ERASE_BLOCK
                    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
                    BSF       EECON1, EEPGD            ; point to Flash program memory
                    BCF       EECON1, CFGS             ; access Flash program memory
                    BSF       EECON1, WREN             ; enable write to memory
                    BSF       EECON1, FREE             ; enable Erase operation
                    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 erase (CPU stall)
                    BSF       INTCON, GIE              ; re-enable interrupts
                    TBLRD*-                            ; dummy read decrement
                    MOVLW     BUFFER_ADDR_HIGH         ; point to buffer
                    MOVWF     FSR0H
                    MOVLW     BUFFER_ADDR_LOW
                    MOVWF     FSR0L
WRITE_BUFFER_BACK
                    MOVLW     BlockSize                ; number of bytes in holding register
                    MOVWF     COUNTER
                    MOVLW     D'64'/BlockSize          ; number of write blocks in 64 bytes
                    MOVWF     COUNTER2
WRITE_BYTE_TO_HREGS
                    MOVF      POSTINC0, W              ; 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.
```

例 4-3：　写闪存程序存储器 （续）

```
                DECFSZ  COUNTER              ; loop until holding registers are full
                BRA     WRITE_WORD_TO_HREGS
PROGRAM_MEMORY
                BSF     EECON1, EEPGD        ; point to Flash program memory
                BCF     EECON1, CFGS         ; access Flash 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)
                DCFSZ   COUNTER2             ; repeat for remaining write blocks
                BRA     WRITE_BYTE_TO_HREGS  ;
                BSF     INTCON, GIE          ; re-enable interrupts
                BCF     EECON1, WREN         ; disable write to memory
```

4.5.2　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。

4.5.3　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验证，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 $\overline{MCLR}$ 复位或 WDT 超时复位而中断，则 WRERR 位将被置 1，用户可以检查该位以确定是否需要重写该单元。

4.5.4　防止误写操作的保护措施

为防止对闪存程序存储器的误写操作，必须遵循写操作的启动顺序。更多详细信息，请参见**第 24.0 节 “ CPU 的特殊功能 ”**。

4.6　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 24.3 节 “ 程序校验和代码保护 ”**。

表 4-3：　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>）					285
TBPLTRH	程序存储器表指针高字节 （TBLPTR<15:8>）								285
TBLPTRL	程序存储器表指针低字节 （TBLPTR<7:0>）								285
TABLAT	程序存储器表锁存器								285
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	285
EECON2	EEPROM 控制寄存器 2 （不是实际存在的寄存器）								287
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	287
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	—	288
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	288
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	288

图注：　— = 未实现，读为 0。闪存 /EEPROM 访问期间不使用阴影单元。

注：

5.0 数据 EEPROM 存储器

数据 EEPROM 是非易失性的存储器阵列，独立于数据 RAM 和程序存储器，用于程序数据的长期存储。它并不直接映射到寄存器文件或程序存储空间，而是通过特殊功能寄存器（SFR）来间接寻址。在整个 VDD 范围内的正常运行期间， EEPROM 是可读写的。

有 5 个 SFR 用于读写数据 EEPROM 以及程序存储器。它们是：

- EECON1
- EECON2
- EEDATA
- EEADR

数据 EEPROM 允许以字节为单位进行读写。当与数据存储器模块接口时， EEDATA 存放 8 位读写数据，而 EEADR 寄存器存放被访问的 EEPROM 存储单元的地址。

EEPROM 数据存储器具有高耐擦写次数。字节写操作会自动擦除目标存储单元并写入新数据（在写入前擦除）。写入时间由片上定时器控制，其值根据电压、温度和不同的芯片而不同。具体的限制值，请参见参数 US122（**第 27.0 节 “ 电气规范 ”** 中的表 27-13）。

5.1 EEADR 寄存器

EEADR 寄存器用于寻址数据 EEPROM 以进行读写操作。8 位的寄存器可寻址 256 字节（00h 至 FFh）的存储器范围。

5.2 EECON1 和 EECON2 寄存器

对数据 EEPROM 的访问由 EECON1 和 EECON2 两个寄存器控制。它们也用来控制对程序存储器的访问，两者使用的方法非常类似。

EECON1 寄存器（寄存器 5-1）是数据和程序存储器访问的控制寄存器。控制位 EEPGD 决定访问的是程序存储器还是数据 EEPROM 存储器。当 EEPGD 位清零时，操作将访问数据 EEPROM 存储器。当 EEPGD 位置 1 时，将访问程序存储器。

控制位 CFGS 决定访问的是配置寄存器还是程序存储器 / 数据 EEPROM 存储器。当 CFGS 位置 1 时，后续操作将访问配置寄存器。当 CFGS 位清零时，则由 EEPGD 位来选择具体访问闪存程序存储器还是数据 EEPROM 存储器。

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。

注：	当写操作完成时， PIR2 寄存器的 EEIF 中断标志位被置 1。它必须用软件清零。

控制位 RD 和 WR 分别启动读和擦写操作。这些位由固件置 1，并在操作完成时由硬件清零。

当访问程序存储器（EEPGD = 1）时，RD 位无法置 1。程序存储器是通过表读指令读取的。关于表读的信息，请参见**第 4.1 节 “ 表读与表写 ”**。

EECON2 寄存器不是实际存在的寄存器。它专用于存储器写和擦除过程。读 EECON2 将得到全 0。

寄存器 5-1：　　EECON1：数据 EEPROM 控制寄存器 1

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

图注：			
R = 可读位	W = 可写位		
S = 该位可用软件置 1，但不能清零		U = 未实现位，读为 0	
-n = POR 时的值	1 = 置 1	0 = 清零	x = 未知

bit 7　**EEPGD：**闪存程序存储器或数据 EEPROM 存储器选择位
1 = 访问闪存程序存储器
0 = 访问数据 EEPROM 存储器

bit 6　**CFGS：**闪存程序存储器 / 数据 EEPROM 存储器或配置寄存器选择位
1 = 访问配置寄存器
0 = 访问闪存程序存储器或数据 EEPROM 存储器

bit 5　**未实现：**读为 0

bit 4　**FREE：**闪存行 （块）擦除使能位
1 = 在下一条 WR 命令时擦除 TBLPTR 指定的程序存储器块 （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 / 数据 EEPROM 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 / 数据 EEPROM 存储器写使能位
1 = 允许对闪存程序存储器 / 数据 EEPROM 存储器的写周期
0 = 禁止对闪存程序存储器 / 数据 EEPROM 存储器的写周期

bit 1　**WR：**写控制位
1 = 启动数据 EEPROM 擦除 / 写周期或程序存储器的擦除周期或写周期。
（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，但不能清零。）
0 = EEPROM 写周期完成

bit 0　**RD：**读控制位
1 = 启动 EEPROM 读操作（读操作需要一个指令周期。RD 由硬件清零。用软件只能将 RD 位置 1，但不能清零。 EEPGD = 1 或 CFGS = 1 时， RD 位无法置 1。）
0 = 不启动 EEPROM 读操作

注　1： 当发生 WRERR 时， EEPGD 和 CFGS 位不会被清零。这样可以跟踪错误情况。

5.3 读数据 EEPROM 存储器

要读取数据存储单元，用户必须将地址写入 EEADR 寄存器，清零 EECON1 寄存器的 EEPGD 控制位，然后将控制位 RD 置 1。可在下一个指令周期访问该数据；因此，EEDATA 寄存器可由下一条指令读取。EEDATA 将把此值保存至下一次读取或用户向该单元写入数据时（写操作）为止。

基本过程如例 5-1 中所示。

5.4 写数据 EEPROM 存储器

要向 EEPROM 数据存储单元写入数据，必须首先将地址写入 EEADR 寄存器，并将数据写入 EEDATA 寄存器。必须遵循例 5-2 中的序列启动写周期。

如果没有完全遵循该指令序列（即首先将 55h 写入 EECON2，随后将 0AAh 写入 EECON2，最后将 WR 位置 1）写每个字节，将不会启动写操作。强烈建议在这段代码执行期间禁止中断。

此外，必须将 EECON1 中的 WREN 位置 1 以使能写操作。这种机制可防止由于意外执行代码（即程序失控）导致误写数据 EEPROM。除了更新 EEPROM 时以外，WREN 位应始终保持清零。WREN 位不会被硬件清零。

一个写序列启动后，EECON1、EEADR 和 EEDATA 不能被修改。除非 WREN 位置 1，否则 WR 位将禁止被置 1。WR 和 WREN 不能在同一条指令中置 1。

写周期完成后，WR 位由硬件清零并且 EEPROM 中断标志位 EEIF 被置 1。用户可以允许此中断或查询此位。EEIF 必须用软件清零。

5.5 写校验

根据具体应用，将写入存储器的值与原始值校验比对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。

例 5-1：　读数据 EEPROM

```
        MOVLW   DATA_EE_ADDR    ;
        MOVWF   EEADR           ; Data Memory Address to read
        BCF     EECON1, EEPGD   ; Point to DATA memory
        BCF     EECON1, CFGS    ; Access EEPROM
        BSF     EECON1, RD      ; EEPROM Read
        MOVF    EEDATA, W       ; W = EEDATA
```

例 5-2：　写数据 EEPROM

```
                MOVLW   DATA_EE_ADDR_LOW    ;
                MOVWF   EEADR               ; Data Memory Address to write
                MOVLW   DATA_EE_DATA        ;
                MOVWF   EEDATA              ; Data Memory Value to write
                BCF     EECON1, EEPGD       ; Point to DATA memory
                BCF     EECON1, CFGS        ; Access EEPROM
                BSF     EECON1, WREN        ; Enable writes
                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          ; Set WR bit to begin write
                BSF     INTCON, GIE         ; Enable Interrupts

                                            ; User code execution
                BCF     EECON1, WREN        ; Disable writes on write complete (EEIF set)
```

5.6 代码保护期间的操作

数据 EEPROM 存储器在配置字中有它自己的代码保护位。如果代码保护机制被使能，外部读写操作就被禁止。

单片机本身可以读写内部数据 EEPROM，与代码保护配置位的状态无关。更多信息，请参见**第 24.0 节“ CPU 的特殊功能 ”**。

5.7 防止误写操作的保护措施

有些情况下，用户并不希望写入数据 EEPROM 存储器。为了防止 EEPROM 误写操作，器件内建了各种保护机制。上电时， WREN 位被清零。而且，上电延时定时期间（TPWRT，参数33）也会阻止对EEPROM进行写操作。

在欠压、电源故障或软件故障期间，写操作的启动序列以及 WREN 位可共同防止意外写操作的发生。

5.8 使用数据 EEPROM

数据 EEPROM 是高耐用可字节寻址的阵列，已将其优化以便存储频繁变动的信息（例如，程序变量或其他经常更新的数据）。如果一个段中的变量经常发生改变，而另一个段中的变量不发生改变，就可能造成超出对 EEPROM 的总写周期数，而不超出对某个字节的总写周期数。如果这样，必须执行一次阵列刷新。出于此原因，不经常更新的变量（如常量、ID 和校准值等）应存放在闪存程序存储器中。

例 5-3： 数据 EEPROM 刷新程序

```
        CLRF    EEADR                   ; Start at address 0
        BCF     EECON1, CFGS            ; Set for memory
        BCF     EECON1, EEPGD           ; Set for Data EEPROM
        BCF     INTCON, GIE             ; Disable interrupts
        BSF     EECON1, WREN            ; Enable writes
Loop                                    ; Loop to refresh array
        BSF     EECON1, RD              ; Read current address
        MOVLW   55h                     ;
        MOVWF   EECON2                  ; Write 55h
        MOVLW   0AAh                    ;
        MOVWF   EECON2                  ; Write 0AAh
        BSF     EECON1, WR              ; Set WR bit to begin write
        BTFSC   EECON1, WR              ; Wait for write to complete
        BRA     $-2
        INCFSZ  EEADR, F                ; Increment address
        BRA     LOOP                    ; Not zero, do it again

        BCF     EECON1, WREN            ; Disable writes
        BSF     INTCON, GIE             ; Enable interrupts
```

表 5-1： 与数据 EEPROM 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
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	285
EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	287
EEDATA	EEPROM 数据寄存器								287
EECON2	EEPROM 控制寄存器 2（不是实际存在的寄存器）								287
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	287
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	—	288
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	288
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	288

图注： — = 未实现，读为 0。闪存 /EEPROM 访问期间不使用阴影单元。

6.0 8 x 8 硬件乘法器

6.1 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减少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 6-1 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。

6.2 工作原理

例 6-1 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乘数时，实现该运算仅需一条指令。

例 6-2 给出了一个 8 x 8 有符号乘法运算的指令序列。要弄清乘数的符号位，必须检查每个乘数的最高有效位（MSb），并做相应的减法。

例 6-1： 8 x 8 无符号乘法程序

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

例 6-2： 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 6-1： 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 （最多）	时间		
				40 MHz 时	10 MHz 时	4 MHz 时
8 x 8 无符号	软件乘法	13	69	6.9 µs	27.6 µs	69 µs
	硬件乘法	1	1	100 ns	400 ns	1 µs
8 x 8 有符号	软件乘法	33	91	9.1 µs	36.4 µs	91 µs
	硬件乘法	6	6	600 ns	2.4 µs	6 µs
16 x 16 无符号	软件乘法	21	242	24.2 µs	96.8 µs	242 µs
	硬件乘法	28	28	2.8 µs	11.2 µs	28 µs
16 x 16 有符号	软件乘法	52	254	25.4 µs	102.6 µs	254 µs
	硬件乘法	35	40	4.0 µs	16.0 µs	40 µs

例 6-3 给出了一个 16 x 16 无符号乘法运算的指令序列。公式 6-1 为所使用的算法。32 位结果存储在 4 个寄存器（RES<3:0>）中。

公式 6-1： 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 6-3： 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 6-4 给出了 16 x 16 有符号乘法运算的指令序列。公式 6-2 为所使用的算法。 32 位结果存储在 4 个寄存器 （RES<3:0>）中。要弄清乘数的符号位，必须检查每个乘数的最高有效位 （MSb），并做相应的减法。

公式 6-2： 16 x 16 有符号乘法算法

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

例 6-4： 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
    :
```

7.0 中断

PIC18F/LF1XK50 器件具有多个中断源及一个中断优先级功能，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 10 个寄存器用于控制中断操作。这些寄存器是：

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1 和 PIR2
- PIE1 和 PIE2
- IPR1 和 IPR2

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 / 编译器能够自动识别指定寄存器内的这些位。

通常，中断源有 3 个位用于控制其操作。这些位的功能分别是：

- **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

7.1 中档兼容性

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® 单片机中档器件兼容的。在兼容模式下，IPRx 寄存器的中断优先级位不起作用。INTCON 寄存器的 PEIE 位是外设的全局中断允许位。PEIE 位只能禁止外设中断源，在 GIE 位也置 1 时可以允许外设中断源。INTCON 寄存器的 GIE 位是全局中断允许位，它可以允许所有非外设中断源，可以禁止包括外设在内的所有中断源。在兼容模式下，所有中断均跳转到地址 0008h。

7.2 中断优先级

通过将 RCON 寄存器的 IPEN 位置 1，可使能中断优先级功能。当使能中断优先级时，兼容模式的 GIE 和 PEIE 全局中断允许位被 GIEH（高优先级）和 GIEL（低优先级）全局中断允许位替代。当置 1 时，INTCON 寄存器的 GIEH 位可以允许所有相应的 IPRx 寄存器或 INTCONx 寄存器优先级位置 1（高优先级）的中断。当清零时，GIEH 位可以禁止包括低优先级中断源在内的所有中断源。当清零时，INTCON 寄存器的 GIEL 位只能禁止相应的优先级位清零（低优先级）的中断。当置 1 时，GIEL 位可以在 GIEH 位也置 1 时允许低优先级中断源。

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，中断将根据中断源的优先级位的级别立即跳转到地址 0008h（高优先级）或 0018h（低优先级）。也可以通过设置相应的中断允许位来禁止单个中断。

7.3 中断响应

当响应中断时，全局中断允许位被清零以禁止其他中断。当 IPEN 位清零时，GIE 位是全局中断允许位。当 IPEN 位置 1 时（即使能中断优先级），GIEH 位是高优先级全局中断允许位，GIEL 位是低优先级全局中断允许位。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响应。

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 INTCONx 和 PIRx 寄存器中的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，以避免重复响应同一中断。

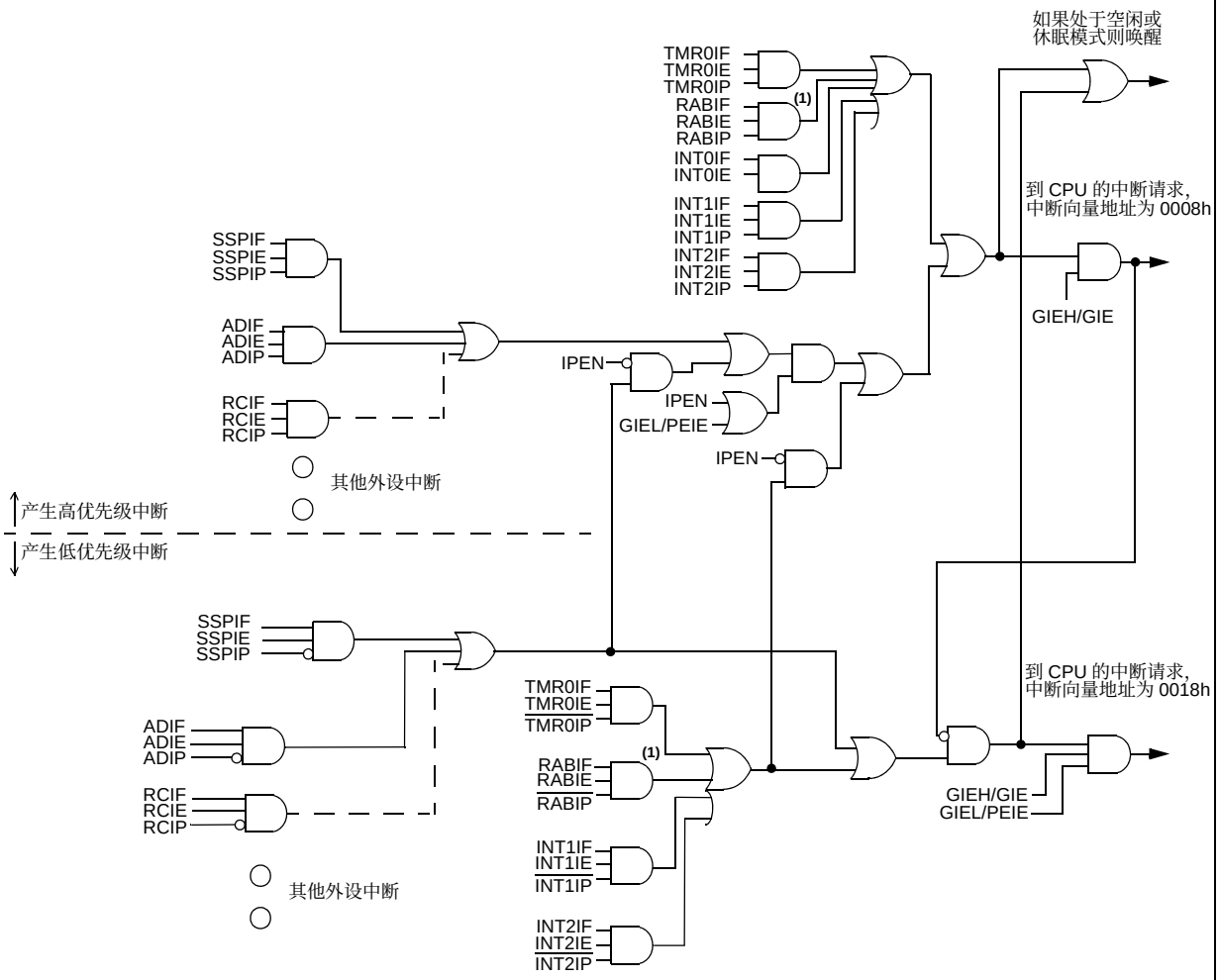
执行“从中断返回”指令 RETFIE 将退出中断程序，同时将 GIE 位（若使用中断优先级则为 GIEH 或 GIEL 位）置 1，从而重新允许中断。

对于外部中断事件，例如 INT 引脚中断或者 PORTB 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全局中断允许位状态的影响。

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 7-1：　　PIC18 中断逻辑



注　　1： RABIF 中断还要求通过 IOCA 和 IOCB 单独使能各个引脚。

7.4 INTCON 寄存器

INTCON 寄存器是可读写的寄存器，包含多个允许位、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。中断标志位可由软件查询。

寄存器 7-1： 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	RABIE	TMR0IF	INT0IF	RABIF
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 （包括外设中断）
当 IPEN = 1 时：
1 = 允许所有高优先级中断
0 = 禁止所有中断 （包括低优先级中断）

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中断
0 = 禁止所有低优先级的中断

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 **RABIE：**RA 和 RB 端口电平变化中断允许位 **(2)**
1 = 允许 RA 和 RB 端口电平变化中断
0 = 禁止 RA 和 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 **RABIF：**RA 和 RB 端口电平变化中断标志位 **(1)**
1 = RA <5:3> 或 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 = RA<5:3> 或 RB<7:4> 引脚电平状态没有改变

注 1： 不匹配条件将继续把 RABIF 位置 1。读取 PORTA 和 PORTB 可以结束不匹配情况，并将该位清零。
2： RA 和 RB 端口各个引脚的电平变化中断还要求通过 IOCA 和 IOCB 单独使能。

寄存器 7-2：　　INTCON2：中断控制寄存器 2

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{\text{RABPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RABIP
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　**$\overline{\text{RABPU}}$：**PORTA 和 PORTB 上拉使能位
1 = 禁止所有 PORTA 和 PORTB 上拉
0 = 如果引脚是输入引脚，且对应的 WPUA 和 WPUB 位置 1，则使能 PORTA 和 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　**未实现：**读为 0

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

bit 1　**未实现：**读为 0

bit 0　**RABIP：**RA 和 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。中断标志位可由软件查询。

寄存器 7-3：　　INTCON3：中断控制寄存器 3

R/W-1	R/W-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
INT2IP	INT1IP	—	INT2IE	INT1IE	—	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　**未实现：**读为 0

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　**未实现：**读为 0

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。中断标志位可由软件查询。

7.5 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，有两个外设中断请求标志寄存器 （PIR1 和 PIR2）。

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 （在 INTCON 寄存器中）的状态如何，中断标志位都将置 1。

2： 用户软件应在允许使能中断前和处理完中断后，将相应的中断标志位清零。

寄存器 7-4： 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	R/W-0	R/W-0	R/W-0
—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	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 = 一次 A/D 转换已完成 （必须用软件清零）
0 = A/D 转换未完成或尚未开始

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

bit 4 **TXIF：**EUSART 发送中断标志位
1 = EUSART 发送缓冲区 TXREG 为空 （写入 TXREG 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 **CCP1IF：**CCP1 中断标志位
捕捉模式：
1 = 发生了 TMR1 寄存器捕捉 （必须用软件清零）
0 = 未发生 TMR1 寄存器捕捉
比较模式：
1 = 发生了 TMR1 寄存器的比较匹配 （必须用软件清零）
0 = 未发生 TMR1 寄存器的比较匹配
PWM 模式：
在此模式下未使用

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 7-5：　　PIR2：外设中断请求 （标志）寄存器 2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	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 HFINTOSC 作为时钟输入 （必须用软件清零）
0 = 器件时钟正常工作

bit 6　**C1IF：**比较器 C1 中断标志位
1 = 比较器 C1 输出已改变 （必须用软件清零）
0 = 比较器 C1 输出未改变

bit 5　**C2IF：**比较器 C2 中断标志位
1 = 比较器 C2 输出已改变 （必须用软件清零）
0 = 比较器 C2 输出未改变

bit 4　**EEIF：**数据 EEPROM/ 闪存写操作中断标志位
1 = 写操作完成 （必须用软件清零）
0 = 写操作未完成或尚未开始

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　**USBIF：**USB 中断标志位
1 = USB 已请求中断 （必须用软件清零）
0 = 无 USB 中断请求

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

7.6 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，有两个外设中断允许寄存器（PIE1 和 PIE2）。当 IPEN = 0 时，要允许任一外设中断，必须将 PEIE 位置 1。

寄存器 7-6： PIE1：外设中断允许 （标志）寄存器 1

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	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 **RCIE：**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 **TXIE：**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 **CCP1IE：**CCP1 中断允许位
1 = 允许 CCP1 中断
0 = 禁止 CCP1 中断

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 7-7：　　PIE2：外设中断允许 （标志）寄存器 2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	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　**C1IE：**比较器 C1 中断允许位
1 = 允许
0 = 禁止

bit 5　**C2IE：**比较器 C2 中断允许位
1 = 允许
0 = 禁止

bit 4　**EEIE：**数据 EEPROM/ 闪存写操作中断允许位
1 = 允许
0 = 禁止

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

bit 2　**USBIE：**USB 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

7.7 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，有两个外设中断优先级寄存器（IPR1 和 IPR2）。使用优先级位时，要求将中断优先级使能（IPEN）位置 1。

寄存器 7-8： 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	R/W-1	R/W-1	R/W-1
—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	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 **RCIP：**EUSART 接收中断优先级位
1 = 高优先级
0 = 低优先级

bit 4 **TXIP：**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 **CCP1IP：**CCP1 中断优先级位
1 = 高优先级
0 = 低优先级

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 7-9：　　IPR2：外设中断优先级寄存器 2

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	U-0
OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	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　**C1IP：**比较器 C1 中断优先级位
1 = 高优先级
0 = 低优先级

bit 5　**C2IP：**比较器 C2 中断优先级位
1 = 高优先级
0 = 低优先级

bit 4　**EEIP：**数据 EEPROM/ 闪存写操作中断优先级位
1 = 高优先级
0 = 低优先级

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　**USBIP：**USB 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

7.8 RCON 寄存器

RCON寄存器中包含的位可用来确定器件上次复位或从空闲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。

SBOREN位和复位标志位的操作已在**第23.1节“ RCON 寄存器 ”**中详细讨论。

寄存器 7-10： RCON：复位控制寄存器

R/W-0	R/W-1	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	SBOREN(1)	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$(2)	$\overline{BOR}$
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 **SBOREN：**BOR 软件使能位 (1)
如果 BOREN<1:0> = 01：
1 = 使能 BOR
0 = 禁止 BOR
如果 BOREN<1:0> = 00、 10 或 11：
该位被禁止并读为 0。

bit 5 **未实现：**读为 0

bit 4 **$\overline{RI}$：**RESET 指令标志位
1 = 未执行 RESET 指令 （由固件置 1 或在上电复位时置 1）
0 = 执行了 RESET 指令，导致器件复位 （发生代码执行复位后必须由固件置 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 指令清零

bit 1 **$\overline{POR}$：**上电复位状态位 (2)
1 = 未发生上电复位
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 (3)
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 POR 或欠压复位后必须由固件置 1）

注 1： 如果使能了 SBOREN，其复位状态为 1 ；否则为 0。
2： $\overline{POR}$ 的实际复位值由器件复位的类型决定。更多信息，请参见该寄存器下方的 “ 注 ” 和**第 23.6 节 “ 寄存器的复位状态 ”**。
3： 请参见表 23-3。

7.9 INTn 引脚中断

RC0/INT0、RC1/INT1 和 RC2/INT2 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 RCx/INTx 引脚上出现一个有效边沿时，相应的标志位 INTxF 被置 1。通过清零相应的允许位 INTxE，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 INTxF 清零。

如果 INTxE 位在进入空闲或休眠模式前被置 1，则所有的外部中断（INT0、INT2 和 INT2）均能将处理器从空闲或休眠模式唤醒。如果全局中断允许位 GIE 被置 1，则处理器将在被唤醒之后转移到中断向量处执行程序。

INT1 和 INT2 的中断优先级由 INTCON3 寄存器的中断优先级位 INT1IP 和 INT2IP 中的值决定。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。

7.10 TMR0 中断

在 8 位模式 （默认模式）下， TMR0 寄存器的溢出（FFh → 00h）会使 TMR0IF 标志位置 1。在 16 位模式下， TMR0H:TMR0L 寄存器对的溢出 （FFFFh → 0000h）会使 TMR0IF 标志位置 1。可以通过置 1/ 清零 INTCON寄存器的允许位TMR0IE来允许/禁止该中断。Timer0 的中断优先级由 INTCON2 寄存器的中断优先级位 TMR0IP 中的值决定。欲进一步了解 Timer0 模块的详细信息，请参见**第 10.0 节 “ Timer0 模块 ”**。

7.11 PORTA 和 PORTB 电平变化中断

PORTA 或 PORTB 上的输入电平变化会将 INTCON 寄存器的标志位 RABIF 置 1。可以通过置 1/ 清零 INTCON 寄存器的允许位 RABIE 来允许 / 禁止该中断。各个引脚还必须使用 IOCA 和 IOCB 寄存器分别使能。PORTA 和 PORTB 电平变化中断的优先级由 INTCON2 寄存器的中断优先级位 RABIP 中的值决定。

7.12 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 3.3 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 7-1 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 7-1： 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
```

8.0 低压差 （LDO）稳压器

PIC18F1XK50 器件与 PIC18LF1XK50 器件的不同之处在于是否具有内置低压差（Low Dropout， LDO）稳压器。PIC18F1XK50 包含内置 LDO，而 PIC18LF1XK50 不包含。

裸片的光刻技术允许在内部数字逻辑上施加最高标称值 3.6V 的工作电压。为继续支持 5.0V 的设计，裸片上集成了 LDO 稳压器。LDO 稳压器允许内部数字逻辑工作于 3.3V，而 I/O 工作于 5.0V （V_{DD}）。

LDO 稳压器需要外接旁路电容来提供稳定性。 V_{USB} 引脚需要外接一个旁路电容。建议使用 0.22 至 0.47 µF 之间的陶瓷电容。

上电时，外部电容对 LDO 稳压器而言就像个大负载。为防止错误操作，恒流源对外部电容充电时，器件保持在复位状态。电容完全充电后，器件从复位状态释放。更多信息，请参见**第 27.0 节 “ 电气规范 ”**。

注：

9.0 I/O 端口

最多有 3 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。

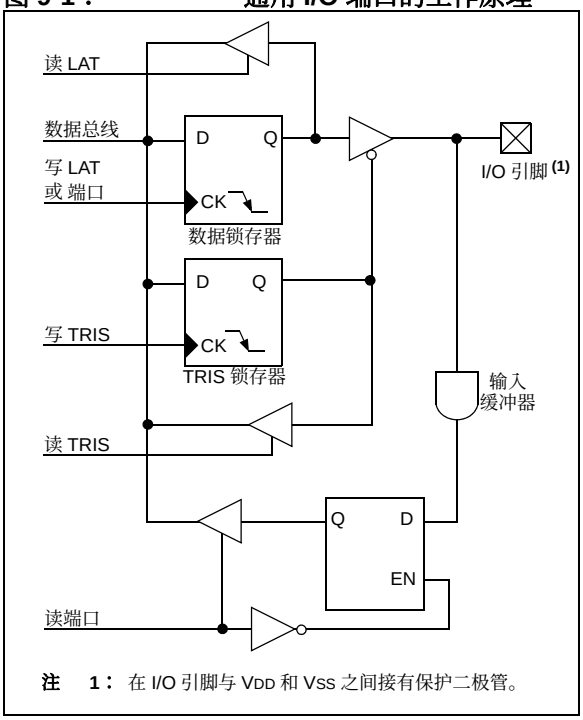
每个端口都有三个工作寄存器。这些寄存器是：

- 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 （输出锁存器）

在对 I/O 引脚驱动值进行读 - 修改 - 写时会用到 PORTA 数据锁存器 （LATA 寄存器）。

图 9-1 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 9-1： 通用 I/O 端口的工作原理



注 1： 在 I/O 引脚与 VDD 和 VSS 之间接有保护二极管。

9.1 PORTA、 TRISA 和 LATA 寄存器

PORTA 为 5 位宽。 PORTA<5:4> 位是双向端口，而 PORTA<3,1:0> 位是仅输入端口。对应的数据方向寄存器是 TRISA。将 TRISA 某位置 1 (= 1) 时，会将 PORTA 的相应引脚设为输入 (即，禁止输出驱动器)。将TRISA 中的某位清零 （= 0）时，会将 PORTA 的相应引脚设为输出 （即，使能输出驱动器并将输出锁存器中的内容输出到选中引脚）。

读 PORTA 寄存器将读出相应引脚的状态，而对其进行写操作则是将数据写入端口锁存器。

PORTA 数据锁存寄存器 （LATA）也是存储器映射的。对 LATA 寄存器执行读 - 修改 - 写操作将读写 PORTA 的输出锁存值。

所有 PORTA 引脚被单独配置为具有电平变化中断功能的引脚。 IOCA 寄存器中的控制位可以允许 （置 1 时）或禁止 （清零时）每个引脚的中断功能。

当置 1 时， INTCON 寄存器的 RABIE 位可以允许将对应的 IOCA 位置 1 的所有引脚上的中断。当清零时，RABIE 位可以禁止所有电平变化中断。

仅当将这些引脚配置为输入时，才具有此中断功能 （即当任何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）。

对于允许了电平变化中断功能的引脚，其值将与上次读取的 PORTA 的旧锁存值相比较。所有与上次读取值不匹配的输出进行逻辑或运算，运算结果用来设置 INTCON 寄存器中的 PORTA 电平变化中断标志位 （RABIF）。

该中断可将器件从休眠模式或任何空闲模式唤醒。用户可用以下方式在中断服务程序中清除该中断：

a) 通过读或写 PORTA 来清除不匹配条件 （当 PORTA 是 MOVFF 指令的源或目标时除外）。
b) 清零标志位 RABIF。

不匹配条件将继续把 RABIF 标志位置 1。读或写 PORTA 将结束不匹配条件并允许将 RABIF 位清零。保存上一次读取值的锁存器不受 MCLR 或欠压复位的影响。在这些复位之后，如果存在不匹配情况， RABIF 标志位还将继续被置 1。

注 **1：** 当读操作正在执行时发生了 I/O 引脚电平变化 （Q2 周期的起始时刻），则 RABIF 中断标志位可能不会被置 1。此外，由于端口上的读或写操作会影响该端口的所有位，所以在电平变化中断模式下使用多个引脚时必须特别小心。在处理一个引脚上的电平变化时，若另一个引脚的电平也发生了变化，可能无法察觉。

2： 当配置为 USB 操作时，RA0 和 RA1 上的电平变化中断功能被自动禁止。

3: 要使能RA<1:0>端口引脚上的数字输入功能，必须使能电平变化中断引脚（IOCA <1:0> = 11）并禁止 USB 模块（USBEN = 0）。

建议使用电平变化中断功能实现按键唤醒操作，以及那些仅用到 PORTA 的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 PORTA 的状态。

每个 PORTA 引脚都具有单独控制的内部弱上拉功能。当置 1 时，WPUA 寄存器的每个位可以使能相应的引脚上拉功能。当清零时， INTCON2 寄存器的 $\overline{\text{RABPU}}$ 位可以使能将对应的 WPUA 位置 1 的所有引脚上的上拉功能。当置 1 时，$\overline{\text{RABPU}}$ 位可以禁止所有弱上拉功能。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。

注： 上电复位时，默认情况下 RA4 被配置为模拟输入且读为 0 ； RA<1:0> 和 RA<5:3> 则被配置为数字输入。

RA0 和 RA1 与 USB 模块复用，可作为片上 USB 收发器的差分数据线。
RA0 和 RA1 不具备与它们相关的 TRISA 位。作为数字端口引脚，它们只能用作数字输入。当配置为 USB 操作时，数据方向是由给定时间内 USB 模块的配置和状态决定的。

RA3 是仅输入引脚。其操作由 CONFIG3H 寄存器的 MCLRE 位控制。当被选为端口引脚（MCLRE = 0）时，它仅用作数字输入引脚；这样，它不具备与操作相关的 TRIS 或 LAT 位。

注： 上电复位时，仅当主复位功能禁止时，才能将 RA3 使能为数字输入引脚。

RA4 和 RA5 引脚与主振荡器引脚复用；通过在配置寄存器 （详情请参见**第 24.1 节 “ 配置位 ”**）中对主振荡器进行配置可将这两个引脚使能为振荡器或 I/O 引脚。当没被用作端口引脚时，RA4 和 RA5 及其相关的 TRIS 和 LAT 位均读为 0。

引脚 RA4 与模拟输入复用。通过将 ANSEL 寄存器中的 ANS3 位置 1 （上电复位后的默认设置），可以将引脚 RA4 用作模拟输入。

注： 上电复位后， RA4 被配置为模拟输入且读为 0。

例 9-1： 初始化 PORTA

```
CLRF    PORTA   ; Initialize PORTA by
                ; clearing output
                ; data latches
CLRF    LATA    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   030h    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISA   ; Set RA<5:4> as output
```

寄存器 9-1： PORTA：PORTA 寄存器

U-0	U-0	R/W-x	R/W-x	R-x	U-0	R/W-x	R/W-x
—	—	RA5	RA4	RA3	—	RA1	RA0
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 5-3 **RA<5:3>**：PORTA I/O 引脚位 **(1)**
1 = 端口引脚大于 VIH
0 = 端口引脚小于 VIL

bit 2 **未实现**：读为 0

bit 1-0 **RA<1:0>**：PORTA I/O 引脚位
1 = 端口引脚大于 VIH
0 = 端口引脚小于 VIL

注 1： RA3 位仅在主复位被禁止 （MCLRE 配置位 = 0）时可用。否则， RA3 读为 0。该位是只读的。

寄存器 9-2： TRISA：PORTA 三态寄存器

U-0	U-0	R/W-1	R/W-1	U-0	U-0	U-0	U-0
—	—	TRISA5	TRISA4	—	—	—	—
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 **TRISA<5:4>：**PORTA 三态控制位
1 = PORTA 引脚配置为输入 （三态）
0 = PORTA 引脚配置为输出

bit 3-0 **未实现**：读为 0

注 1： 在 XT、 HS 和 LP 振荡器模式下， TRISA<5:4> 总是读为 1。

寄存器 9-3：　　WPUA：弱上拉 PORTA 寄存器

U-0	U-0	R/W-1	R/W-1	RW-1	U-0	U-0	U-0
—	—	WPUA5	WPUA4	WPUA3	—	—	—
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 5-3　**WPUA<5:3>**：弱上拉使能位
1 = 使能上拉
0 = 禁止上拉

bit 2　**未实现**：读为 0

bit 1-0　**WPUA<1:0>**：弱上拉使能位
1 = 使能上拉
0 = 禁止上拉

寄存器 9-4：　　IOCA：电平变化中断 PORTA 寄存器

U-0	U-0	R/W-0	R/W-0	R-0	U-0	R/W-0	R/W-0
—	—	IOCA5	IOCA4	IOCA3	—	IOCA1	IOCA0
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 5-3　**IOCA<5:3>**：PORTA I/O 引脚位
1 = 允许电平变化中断
0 = 禁止电平变化中断

bit 2　**未实现**：读为 0

bit 1-0　**IOCA<1:0>**：PORTA I/O 引脚位
1 = 允许电平变化中断
0 = 禁止电平变化中断

寄存器 9-5：　　LATA：PORTA 数据锁存寄存器

U-0	U-0	R/W-x	R/W-x	U-0	U-0	U-0	U-0
—	—	LATA5	LATA4	—	—	—	—
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　**LATA<5:4>**：RA<5:4> 端口 I/O 输出锁存寄存器位

bit 3-0　**未实现**：读为 0

表 9-1： PORTA I/O 汇总

引脚	功能	TRIS 设置	I/O	I/O 类型	说明
RA0/IOCA0/D+/ PGD	RA0	—(1)	I	TTL	PORTA<0> 数据输入；当使能 USB 时被禁止。
	IOCA0	—(1)	I	TTL	引脚电平变化中断；当使能 USB 时被禁止。
	D+	—(1)	I	XCVR	USB 总线差分正信号线输入 （内部收发器）。
		—(1)	O	XCVR	USB 总线差分正信号线输出 （内部收发器）。
	PGD	—(1)	O	DIG	用于 ICSP™ 的串行执行数据输出。
		—(1)	I	ST	用于 ICSP™ 的串行执行数据输入。
RA1/IOCA1/D-/ PGC	RA1	—(1)	I	TTL	PORTA<1> 数据输入；当使能 USB 时被禁止。
	IOCA1	—(1)	I	TTL	引脚电平变化中断；当使能 USB 时被禁止。
	D-	—(1)	I	XCVR	USB 总线差分负信号线输入 （内部收发器）。
		—(1)	O	XCVR	USB 总线差分负信号线输出 （内部收发器）。
	PGC	—(1)	O	DIG	用于 ICSP™ 的串行执行时钟输出。
		—(1)	I	ST	用于 ICSP™ 的串行执行时钟输入。
RA3/IOCA3/$\overline{\text{MCLR}}$/ V_{PP}	RA3	—(2)	I	ST	PORTA<3> 数据输入；当 MCLRE 配置位清零时被使能；可编程的弱上拉。
	IOCA3	—(1)	I	TTL	引脚电平变化中断。
	$\overline{\text{MCLR}}$	—	I	ST	外部主复位输入；当 MCLRE 配置位置 1 时被使能。
	V_{PP}	—	I	ANA	高压检测；用于 ICSP™ 模式进入检测。始终可用，与引脚模式无关。
RA4/IOCA4/AN3/ OSC2/CLKOUT	RA4	0	O	DIG	LATA<4> 数据输出。仅在 RCIO、 INTIO2 和 ECIO 模式下使能。
		1	I	TTL	PORTA<4> 数据输入；可编程的弱上拉。仅在 RCIO、 INTIO2 和 ECIO 模式下使能。
	IOCA4	1	I	TTL	引脚电平变化中断。
	AN3	1	I	ANA	A/D 输入通道 3。 POR 时的默认配置。
	OSC2	x	O	ANA	主振荡器反馈输出连接 （XT、 HS 和 LP 模式）。
	CLKOUT	x	O	DIG	RC、 INTIO1 和 EC 振荡器模式下的系统周期时钟输出 （$F_{OSC}/4$）。
RA5/IOCA5/OSC1/ CLKIN	RA5	0	O	DIG	LATA<5> 数据输出。在外部振荡器模式下被禁止。
		1	I	TTL	PORTA<5> 数据输入。在外部振荡器模式下被禁止；可编程的弱上拉。
	IOCA5	1	I	TTL	引脚电平变化中断。
	OSC1	x	I	ANA	主振荡器输入连接。
	CLKIN	x	I	ANA	主时钟输入连接。

图注： DIG = 数字电平输出； TTL = TTL 输入缓冲器； ST = 施密特触发输入缓冲器； ANA = 模拟电平输入 / 输出； x = 无关位 （TRIS 位不影响端口方向或在此可忽略）。

注 1： RA0 和 RA1 没有对应的 TRISA 位。在端口模式下，这些引脚只作为输入。 USB 数据方向由 USB 配置决定。

2： RA3 没有对应的 TRISA 位。无论模式如何，该引脚总是输入引脚。

表 9-2：　　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	—	—	RA5(1)	RA4(1)	RA3(2)	—	RA1(3)	RA0(3)	288
LATA	—	—	LATA5(1)	LATA4(1)	—	—	—	—	288
TRISA	—	—	TRISA5(1)	TRISA4(1)	—	—	—	—	288
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	—	—	—	288
SLRCON	—	—	—	—	—	SLRC	SLRB	SLRA	288
IOCA	—	—	IOCA5	IOCA4	IOCA3(2)	—	IOCA1(3)	IOCA0(3)	288
WPUA	—	—	WPUA5	WPUA4	WPUA3(2)	—	—	—	288
UCON	—	PPBRST	SE0	PKTDIS	USBEN	RESUME	SUSPND	—	288
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	285
INTCON2	$\overline{\text{RABPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RABIP	285

图注：　— = 未实现，读为 0。 PORTA 不使用阴影单元。

注 1：　RA<5:4> 及其相关的锁存器和数据方向位根据振荡器配置使能为 I/O 引脚；否则，它们将读为 0。

2：　仅当主复位功能被禁止 （MCLRE 配置位 = 0）时才实现。

3：　当 USB 模块被禁止 （UCON<3> = 0）时， RA1 和 RA0 只能用作端口引脚。

9.2 PORTB、 TRISB 和 LATB 寄存器

PORTB 是一个 4 位宽的双向端口，对应的数据方向寄存器是 TRISB。将 TRISB 某位置 1 （= 1）时，会将 PORTB 的相应引脚设为输入（即，禁止输出驱动器）。将 TRISB 某位清零 （= 0）时，会将 PORTB 的相应引脚设为输出（即，使能输出驱动器并将输出锁存器中的内容输出到选中引脚）。

PORTB 数据锁存寄存器 （LATB）也是存储器映射的。对 LATB 寄存器执行读 - 修改 - 写操作将读写 PORTB 的输出锁存值。

例 9-2： 初始化 PORTB

```
CLRF    PORTB   ; Initialize PORTB by
                ; clearing output
                ; data latches
CLRF    LATB    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0F0h    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISB   ; Set RB<7:4> as outputs
```

所有 PORTB 引脚被单独配置为具有电平变化中断功能的引脚。 IOCB 寄存器中的控制位可以允许 （置 1 时）或禁止 （清零时）每个引脚的中断功能。

当置 1 时， INTCON 寄存器的 RABIE 位可以允许将对应的 IOCB 位置 1 的所有引脚上的中断。当清零时，RABIE 位可以禁止所有电平变化中断。

仅当将这些引脚配置为输入时，才具有此中断功能（即当任何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）。

对于允许了电平变化中断功能的引脚，其值将与上次读取的 PORTB 的旧锁存值相比较。所有与上次读取值不匹配的输出进行逻辑或运算，运算结果用来设置 INTCON 寄存器中的 PORTB 电平变化中断标志位（RABIF）。

该中断可将器件从休眠模式或任何空闲模式唤醒。用户可用以下方式在中断服务程序中清除该中断：

a) 通过读或写 PORTB 来清除不匹配条件 （当 PORTB 是 MOVFF 指令的源或目标时除外）。
b) 清零标志位 RABIF。

不匹配条件将继续把 RABIF 标志位置 1。读或写 PORTB 将结束不匹配条件并允许将 RABIF 位清零。保存上一次读取值的锁存器不受 $\overline{\text{MCLR}}$ 或欠压复位的影响。在这些复位之后，如果存在不匹配情况， RABIF 标志位还将继续被置 1。

> **注：** 当读操作正在执行时发生了I/O引脚电平变化 （Q2 周期的起始时刻），则 RABIF 中断标志位可能不会被置 1。此外，由于端口上的读或写操作会影响该端口的所有位，所以在电平变化中断模式下使用多个引脚时必须特别小心。在处理一个引脚上的电平变化时，若另一个引脚的电平也发生了变化，可能无法察觉。

建议使用电平变化中断功能实现按键唤醒操作，以及那些仅用到 PORTB 的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。

所有 PORTB 引脚都具有单独控制的弱上拉功能。当置 1 时，WPUB 寄存器的每个位可以使能相应的引脚上拉功能。当清零时， INTCON2 寄存器的 $\overline{\text{RABPU}}$ 位可以使能将对应的 WPUB 位置 1 的所有引脚上的上拉功能。当置 1 时， $\overline{\text{RABPU}}$ 位可以禁止所有弱上拉功能。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。

> **注：** 上电复位时，默认情况下 RB<5:4> 被配置为模拟输入且读为 0。

寄存器 9-6：　　PORTB：PORTB 寄存器

R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0	U-0
RB7	RB6	RB5	RB4	—	—	—	—
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　**RB<7:4>**：PORTB I/O 引脚位
1 = 端口引脚大于 VIH
0 = 端口引脚小于 VIL

bit 3-0　**未实现**：读为 0

寄存器 9-7：　　TRISB：PORTB 三态寄存器

R/W-1	R/W-1	R/W-1	R/W-1	U-0	U-0	U-0	U-0
TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—
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　**TRISB<7:4>**：PORTB 三态控制位
1 = PORTB 引脚配置为输入 （三态）
0 = PORTB 引脚配置为输出

bit 3-0　**未实现**：读为 0

寄存器 9-8：　　WPUB：弱上拉 PORTB 寄存器

R/W-1	R/W-1	R/W-1	R/W-1	U-0	U-0	U-0	U-0
WPUB7	WPUB6	WPUB5	WPUB4	—	—	—	—
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　**WPUB<7:4>**：弱上拉使能位
1 = 使能上拉
0 = 禁止上拉

bit 3-0　**未实现**：读为 0

寄存器 9-9：　　IOCB：电平变化中断 PORTB 寄存器

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
IOCB7	IOCB6	IOCB5	IOCB4	—	—	—	—
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　**IOCB<7:4>**：电平变化中断位
1 = 允许电平变化中断
0 = 禁止电平变化中断

bit 3-0　**未实现**：读为 0

寄存器 9-10：　　LATB：PORTB 数据锁存寄存器

R/W-x	R/W-x	R/W-x	R/W-x	U-0	U-0	U-0	U-0
LATB7	LATB6	LATB5	LATB4	—	—	—	—
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　**LATB<7:4>**：RB<7:4> 端口 I/O 输出锁存寄存器位

bit 3-0　**未实现**：读为 0

表 9-3： PORTB I/O 汇总

引脚	功能	TRIS 设置	I/O	I/O 类型	说明
RB4/IOCB4/AN10/ SDI/SDA	RB4	0	O	DIG	LATB<4> 数据输出；不受模拟输入影响。
		1	I	TTL	PORTB<4> 数据输入；可编程的弱上拉。
	IOCB4	1	I	TTL	引脚电平变化中断。
	AN10	1	I	ANA	A/D 输入通道 10。
	SDI	1	I	ST	SPI 数据输入 （MSSP 模块）。
	SDA	1	I	DIG	I²C™ 数据输出 （MSSP 模块）；优先于端口数据。
		1	O	I2C	I²C™ 数据输入 （MSSP 模块）；输入类型取决于模块设置。
RB5/IOCB5/AN11/ RX/DT	RB5	0	O	DIG	LATB<5> 数据输出。
		1	I	TTL	PORTB<5> 数据输入；可编程的弱上拉。
	IOCB5	1	I	TTL	引脚电平变化中断。
	AN11	1	I	ANA	A/D 输入通道 11。
	RX	1	I	ST	异步串行接收数据输入 （USART 模块）。
	DT	1	O	DIG	同步串行数据输出 （USART 模块）；优先于端口数据。
		1	I	ST	同步串行数据输入 （USART 模块）。用户必须将其配置为输入。
RB6/IOCB6/SCK/ SCL	RB6	0	O	DIG	LATB<6> 数据输出。
		1	I	TTL	PORTB<6> 数据输入；可编程的弱上拉。
	IOCB6	1	I	TTL	引脚电平变化中断。
	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	I²C™ 时钟输出 （MSSP 模块）；优先于端口数据。
		1	I	I2C	I²C™ 时钟输入 （MSSP 模块）；输入类型取决于模块设置。
RB7/IOCB7/TX/CK	RB7	0	O	DIG	LATB<7> 数据输出。
		1	I	TTL	PORTB<7> 数据输入；可编程的弱上拉。
	IOCB7	1	I	TTL	引脚电平变化中断。
	TX	1	O	DIG	异步串行发送数据输出 （USART 模块）；优先于端口数据。用户必须将其配置为输出。
	CK	1	O	DIG	同步串行时钟输出 （USART 模块）；优先于端口数据。
		1	I	ST	同步串行时钟输入 （USART 模块）。

图注： DIG = 数字电平输出； TTL = TTL 输入缓冲器； ST = 施密特触发输入缓冲器； ANA = 模拟电平输入 / 输出；
x = 无关位 （TRIS 位不影响端口方向或在此可忽略）。

表 9-4：　　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	—	—	—	—	288
LATB	LATB7	LATB6	LATB5	LATB4	—	—	—	—	288
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	288
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	—	—	—	—	288
IOCB	IOCB7	IOCB6	IOCB5	IOCB4	—	—	—	—	288
SLRCON	—	—	—	—	—	SLRC	SLRB	SLRA	288
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	285
INTCON2	$\overline{\text{RABPU}}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RABIP	285
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	288
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	287
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	287
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	286

图注：　— = 未实现，读为 0。 PORTB 不使用阴影单元。

9.3 PORTC、 TRISC 和 LATC 寄存器

PORTC 是一个 8 位宽的双向端口，对应的数据方向寄存器是 TRISC。将 TRISC 某位置 1 （= 1）时，会将 PORTC 的相应引脚设为输入（即，禁止输出驱动器）。将 TRISC 某位清零 （= 0）时，会将 PORTC 的相应引脚设为输出（即，使能输出驱动器并将输出锁存器中的内容输出到选中引脚）。

PORTC 数据锁存寄存器（LATC）也是存储器映射的。对 LATC 寄存器执行读 - 修改 - 写操作将读写 PORTC 的输出锁存值。

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

注：	上电复位时，RC<7:6> 和 RC<3:0> 被配置为模拟输入且读为 0。

例 9-3： 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
```

寄存器 9-11： PORTC：PORTC 寄存器

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
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 **RC<7:0>**：PORTC I/O 引脚位
1 = 端口引脚大于 VIH
0 = 端口引脚小于 VIL

寄存器 9-12： TRISC：PORTC 三态寄存器

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
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 **TRISC<7:0>**：PORTC 三态控制位
1 = PORTC 引脚配置为输入 （三态）
0 = PORTC 引脚配置为输出

寄存器 9-13：　　LATC：PORTC 数据锁存寄存器

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0
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　　**LATC<7:0>**：RC<7:0> 端口 I/O 输出锁存寄存器位

表 9-14： PORTC I/O 汇总

引脚	功能	TRIS 设置	I/O	I/O 类型	说明
RC0/AN4/ C12IN+/VREF+/ INT0	RC0	0	O	DIG	LATC<0> 数据输出。
		1	I	ST	PORTC<0> 数据输入。
	AN4	1	I	ANA	A/D 输入通道 4。
	C12IN+	1	I	ANA	比较器 C1 和 C2 的同相输入。模拟选择与 ADC 共用。
	VREF+	1	I	ANA	ADC 和比较器参考电压高电平输入。
	INT0	1	I	ST	外部中断 0 输入。
RC1/AN5/ C12IN1-/VREF-/ INT1	RC1	0	O	DIG	LATC<1> 数据输出。
		1	I	ST	PORTC<1> 数据输入。
	AN5	1	I	ANA	A/D 输入通道 5。
	C12IN1-	1	I	ANA	比较器 C1 和 C2 的反相输入。模拟选择与 ADC 共用。
	VREF-	1	I	ANA	ADC 和比较器参考电压低电平输入。
	INT1	1	I	ST	外部中断 1 输入。
RC2/AN6/ C12IN2-/CVREF/ P1D/INT2	RC2	0	O	DIG	LATC<2> 数据输出。
		1	I	ST	PORTC<2> 数据输入。
	AN6	1	I	ANA	A/D 输入通道 6。
	C12IN2-	1	I	ANA	比较器 C1 和 C2 的反相输入，通道 2。模拟选择与 ADC 共用。
	CVREF	x	O	ANA	参考电压输出。使能该功能将禁止数字 I/O。
	P1D	0	O	DIG	ECCP1 增强型 PWM 输出，通道 D。可以在增强型 PWM 关闭期间被配置为三态。优先于端口数据。
	INT2	1	I	ST	外部中断 2 输入。
RC3/AN7/ C12IN3-/P1C/ PGM	RC3	0	O	DIG	LATC<3> 数据输出。
		1	I	ST	PORTC<3> 数据输入。
	AN7	1	I	ANA	A/D 输入通道 7。
	C12IN3-	1	I	ANA	比较器 C1 和 C2 的反相输入，通道 3。模拟选择与 ADC 共用。
	P1C	0	O	DIG	ECCP1 增强型 PWM 输出，通道 C。可以在增强型 PWM 关闭期间被配置为三态。优先于端口数据。
	PGM	x	I	ST	单电源供电编程模式选择 （ICSP™）。由 LVP 配置位使能；所有其他引脚功能被禁止。
RC4/C12OUT/ P1B	RC4	0	O	DIG	LATC<4> 数据输出。
		1	I	ST	PORTC<4> 数据输入。
	C12OUT	0	O	DIG	比较器 1 和 2 的输出；优先于端口数据。
	P1B	0	O	DIG	ECCP1 增强型 PWM 输出，通道 B。可以在增强型 PWM 关闭期间被配置为三态。优先于端口数据。

图注： DIG = 数字电平输出； TTL = TTL 输入缓冲器； ST = 施密特触发输入缓冲器； ANA = 模拟电平输入 / 输出；
I^2C/SMB = I^2C/SMBus 输入缓冲器； x = 无关位 （TRIS 位不影响端口方向或在此可忽略）。

表 9-14： PORTC I/O 汇总 （续）

引脚	功能	TRIS 设置	I/O	I/O 类型	说明
RC5/CCP1/P1A/ T0CKI	RC5	0	O	DIG	LATC<5> 数据输出。
		1	I	ST	PORTC<5> 数据输入。
	CCP1	0	O	DIG	ECCP1 比较输出或 PWM 输出；优先于端口数据。
		1	I	ST	ECCP1 捕捉输入。
	P1A	0	0	DIG	ECCP1 增强型 PWM 输出，通道 A。可以在增强型 PWM 关闭期间被配置为三态。优先于端口数据。
	T0CKI	1	I	ST	Timer0 计数器输入。
RC6/AN8/$\overline{SS}$/ T13CKI/T1OSCI	RC6	0	O	DIG	LATC<6> 数据输出。
		1	I	ST	PORTC<6> 数据输入。
	AN8	1	I	ANA	A/D 输入通道 8。
	$\overline{SS}$	1	I	TTL	SSP 的从选择输入 （MSSP 模块）。
	T13CKI	1	I	ST	Timer1 和 Timer3 计数器输入。
	T1OSCI	x	O	ANA	Timer1 振荡器输入；当使能 Timer1 振荡器时被使能。禁止数字 I/O。
RC7/AN9/SDO/ T1OSCO	RC7	0	O	DIG	LATC<7> 数据输出。
		1	I	ST	PORTC<7> 数据输入。
	AN9	1	I	ANA	A/D 输入通道 9。
	SDO	0	I	DIG	SPI 数据输出 （MSSP 模块）；优先于端口数据。
	T1OSCO	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。

图注： DIG = 数字电平输出； TTL = TTL 输入缓冲器； ST = 施密特触发输入缓冲器； ANA = 模拟电平输入 / 输出； I^2C/SMB = I^2C/SMBus 输入缓冲器； x = 无关位 （TRIS 位不影响端口方向或在此可忽略）。

表 9-5： 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	288
LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	288
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	288
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	—	—	—	288
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	288
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	286
T3CON	RD16	—	T3CKPS1	T3CKPS0	T3CCP1	$\overline{T3SYNC}$	TMR3CS	TMR3ON	287
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	286
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	287
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0	287
PSTRCON	—	—	—	STRSYNC	STRD	STRC	STRB	STRA	287
SLRCON	—	—	—	—	—	SLRC	SLRB	SLRA	288
REFCON1	D1EN	D1LPS	DAC1OE	---	D1PSS1	D1PSS0	---	D1NSS	287
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	285
INTCON2	$\overline{RABPU}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RABIP	285
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	285

9.4 端口模拟控制

一些端口引脚与模拟功能 （例如模数转换器和比较器）复用。当这些 I/O 引脚要用作模拟输入时，必须禁止数字输入缓冲器，以避免数字输入的错误偏置导致过量的电流。通过 ANSEL 和 ANSELH 寄存器，可以对与模拟功能复用的引脚上的数字输入缓冲器进行单独控制。将 ANSx 位设为高电平可以禁止关联的数字输入缓冲器，并导致对该引脚的所有读操作返回 0，同时允许该引脚的模拟功能正确工作。

ANSx 位的状态不会影响数字输出功能。相关 TRISx 位清零且 ANSx 位置 1 的引脚将仍作为数字输出工作，但输入模式将变为模拟。

寄存器 9-15： ANSEL：模拟选择寄存器 1

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	U-0	U-0	U-0
ANS7	ANS6	ANS5	ANS4	ANS3	—	—	—
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 **ANS7：**RC3 模拟选择控制位
1 = 禁止 RC3 的数字输入缓冲器
0 = 使能 RC3 的数字输入缓冲器

bit 6 **ANS6：**RC2 模拟选择控制位
1 = 禁止 RC2 的数字输入缓冲器
0 = 使能 RC2 的数字输入缓冲器

bit 5 **ANS5：**RC1 模拟选择控制位
1 = 禁止 RC1 的数字输入缓冲器
0 = 使能 RC1 的数字输入缓冲器

bit 4 **ANS4：**RC0 模拟选择控制位
1 = 禁止 RC0 的数字输入缓冲器
0 = 使能 RC0 的数字输入缓冲器

bit 3 **ANS3：**RA4 模拟选择控制位
1 = 禁止 RA4 的数字输入缓冲器
0 = 使能 RA4 的数字输入缓冲器

bit 2-0 **未实现：**读为 0

寄存器 9-16：　　ANSELH：模拟选择寄存器 2

U-0	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	—	ANS11	ANS10	ANS9	ANS8
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　**未实现：**读为 0

bit 3　**ANS11：**RB5 模拟选择控制位
1 = 禁止 RB5 的数字输入缓冲器
0 = 使能 RB5 的数字输入缓冲器

bit 2　**ANS10：**RB4 模拟选择控制位
1 = 禁止 RB4 的数字输入缓冲器
0 = 使能 RB4 的数字输入缓冲器

bit 1　**ANS9：**RC7 模拟选择控制位
1 = 禁止 RC7 的数字输入缓冲器
0 = 使能 RC7 的数字输入缓冲器

bit 0　**ANS8：**RC6 模拟选择控制位
1 = 禁止 RC6 的数字输入缓冲器
0 = 使能 RC6 的数字输入缓冲器

9.5 端口压摆率控制

可对每个端口的输出压摆率进行编程，以选择标准转换速率或降低的转换速率（标准转换速率的 0.1 倍，最大程度降低 EMI）。对于所有端口，默认压摆率是降低的转换速率。

寄存器 9-17： SLRCON：压摆率控制寄存器

U-0	U-0	U-0	U-0	U-0	R/W-1	R/W-1	R/W-1
—	—	—	—	—	SLRC	SLRB	SLRA
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 **SLRC：**PORTC 压摆率控制位
1 = PORTC 上的所有输出的压摆率为标准速率的 0.1 倍
0 = PORTC 上的所有输出的压摆率为标准速率

bit 1 **SLRB：**PORTB 压摆率控制位
1 = PORTB 上的所有输出的压摆率为标准速率的 0.1 倍
0 = PORTB 上的所有输出的压摆率为标准速率

bit 0 **SLRA：**PORTA 压摆率控制位
1 = PORTA 上的所有输出的压摆率为标准速率的 0.1 倍 **(1)**
0 = PORTA 上的所有输出的压摆率为标准速率

注 1：当引脚用作 CLKOUT 时， RA4 的压摆率默认设为标准速率。

10.0 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 10-1）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 10-1 给出了 8 位模式下 Timer0 模块的简化框图。
图 10-2 给出了 16 位模式下 Timer0 模块的简化框图。

寄存器 10-1： 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 （CLKOUT）

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

10.1 Timer0 工作原理

Timer0 既可用作定时器也可用作计数器；可以通过 T0CON 寄存器的 T0CS 位来选择模式。在定时器模式（T0CS = 0）下，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 10.3 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。通过调整写入 TMR0 寄存器的值来补偿预期的错过的递增，用户可以解决这一问题。

通过将 T0CS 位置 1（= 1）选择计数器模式。在该模式下， Timer0 可在 T0CKI 引脚信号的每个上升沿或下降沿递增。递增边沿由 T0CON 寄存器的 Timer0 时钟源边沿选择位 T0SE 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。

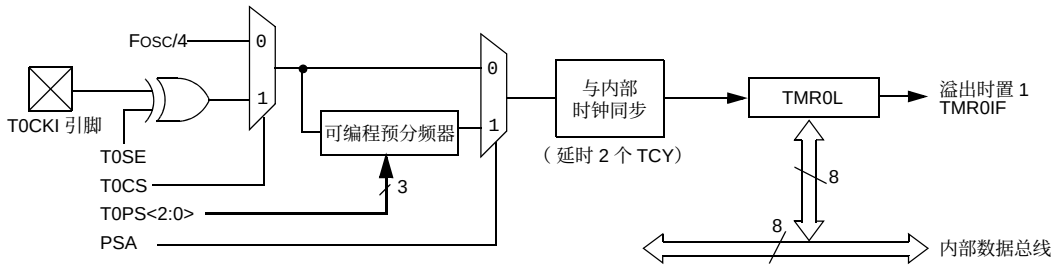
可以使用外部时钟源来驱动 Timer0 ；但是，必须满足一定要求（见表 27-6），以确保外部时钟和内部相位时钟（Tosc）保持同步。在同步之后，定时器 / 计数器需要一定的延时才开始递增。

10.2 16 位模式下 Timer0 的读写操作

TMR0H 并不是 16 位模式下 Timer0 的高字节，而是被缓存的 Timer0 高字节，不可以被直接读写（见图 10-2）。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之间读取的有效性。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。

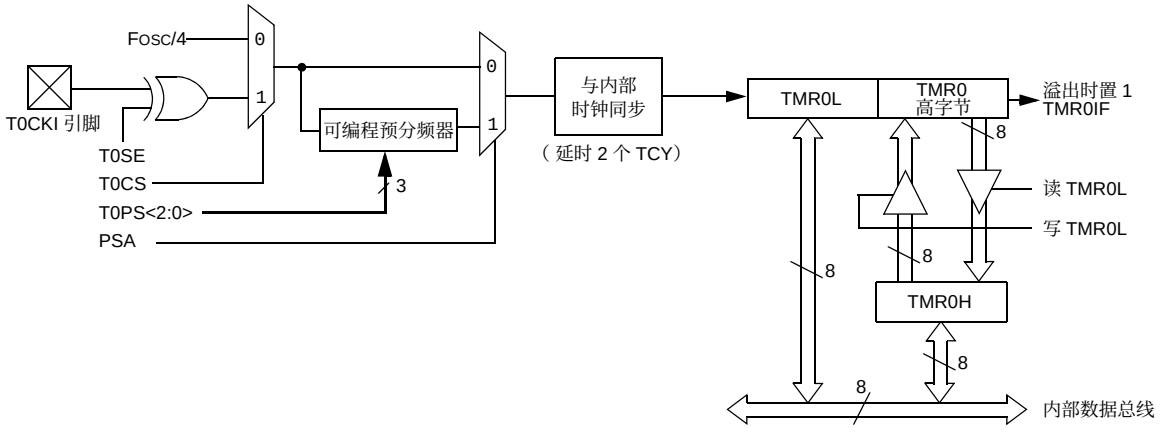
同样，写入 Timer0 的高字节也是通过 TMR0H 缓冲寄存器来操作的。写入 TMR0H 不会直接影响 Timer0。相反，在写入 TMR0L 的同时，使用 TMR0H 的内容更新 Timer0 的高字节。这样一次就可以完成 Timer0 全部 16 位的更新。

图 10-1： 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 10-2：　　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。

10.3 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T0CON寄存器的T0PS<2:0>位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。如果已经分配了预分频器，预分频值可在 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。

10.3.1 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。

10.4 Timer0 中断

8 位模式下的 TMR0 寄存器从 FFh 溢出到 00h，或 16 位模式下的 TMR0 从 FFFFh 溢出到 0000h 时，将产生 TMR0 中断。这种溢出会使 TMR0IF 标志位置 1。可以通过清零 INTCON 寄存器的 TMR0IE 位来屏蔽该中断。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 10-1：　　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								286
TMR0H	Timer0 寄存器的高字节								286
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	285
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	286
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	288

图注： Timer0 不使用阴影单元。

注　1： PORTA<7:6> 及其方向位根据不同的主振荡器模式被单独配置为端口引脚。当被禁止时，这些位读为 0。

注：

11.0 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）
- 可选择的内部或外部时钟源以及 Timer1 振荡器选项
- 溢出时产生中断
- CCP 特殊事件触发模块复位
- 器件时钟状态标志位 （T1RUN）

图 11-1 给出了 Timer1 模块的简化框图。图 11-2 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。

仅需极少量外部元件和代码开销，Timer1 就可为应用提供实时时钟 （Real-Time Clock， RTC）。

Timer1 由 T1CON 控制寄存器 （寄存器 11-1）控制。该寄存器还包含 Timer1 振荡器使能位 （T1OSCEN）。可以通过将 T1CON 寄存器的控制位 TMR1ON 置 1 或清零来使能或禁止 Timer1。

寄存器 11-1： 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 = 主时钟由 Timer1 振荡器产生
0 = 主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：
该位为无关位。当 TMR1CS = 0 时， Timer1 使用内部时钟。

bit 1 **TMR1CS：**Timer1 时钟源选择位
1 = 使用 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

11.1 Timer1 工作原理

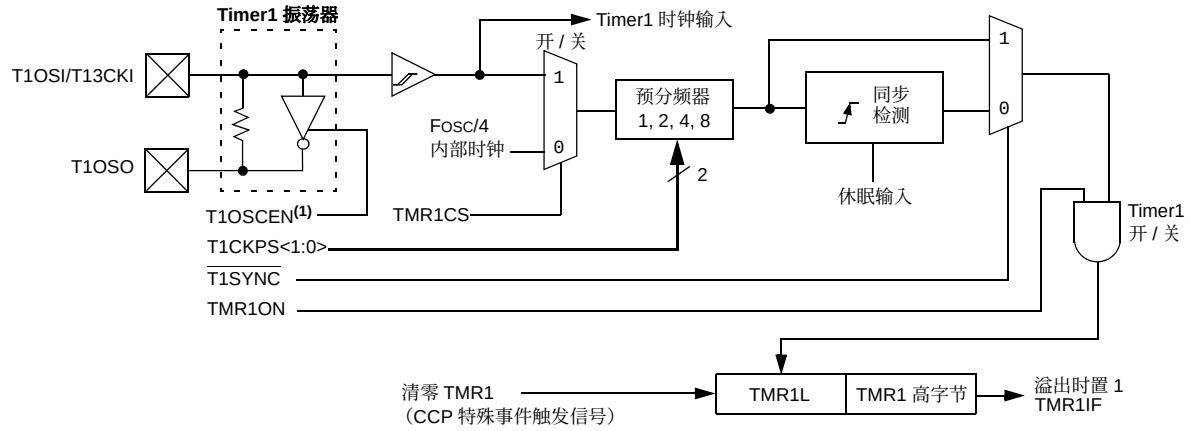
Timer1 可工作在以下模式之一：

- 定时器
- 同步计数器
- 异步计数器

工作模式由 T1CON 寄存器的时钟选择位 TMR1CS 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。

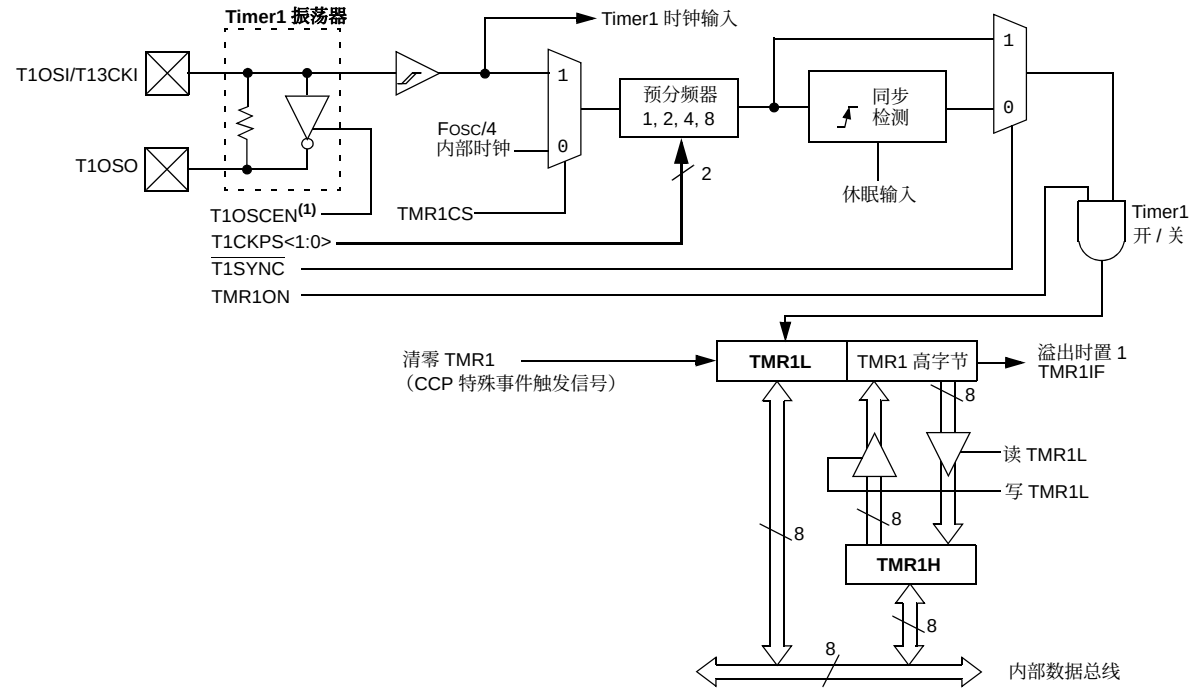
当使能 Timer1 振荡器时，与 T1OSI 和 T1OSO 引脚相关的数字电路被禁止。这意味着 TRISC<1:0> 的值被忽略并且这些引脚将读为 0。

图 11-1： TIMER1 框图



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 11-2： 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。

11.2 Timer1 的 16 位读 / 写模式

可将 Timer1 配置为 16 位读写模式（见图 11-2）。当 T1CON 寄存器的 RD16 控制位置 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 高字节缓冲区。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或进位，而不得不验证读取的有效性。

写入 TMR1H 不会直接影响 Timer1。相反，在写入 TMR1L 的同时，使用 TMR1H 的内容更新 Timer1 的高字节。这样一次就可以完成 Timer1 全部 16 位的更新。

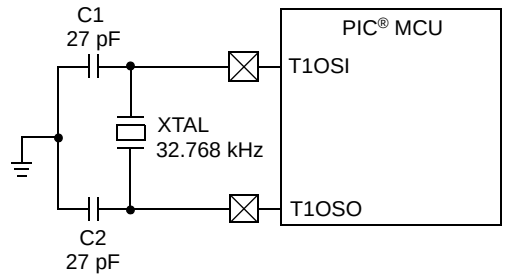
在该模式下不能直接读写 Timer1 的高字节。所有读写都必须通过 Timer1 高字节缓冲寄存器来进行。写入 TMR1H 不会清零 Timer1 预分频器。只有在写 TMR1L 时才会清零该预分频器。

11.3 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 T1CON 寄存器的 Timer1 振荡器使能位 T1OSCEN 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 11-3 所示为典型的 LP 振荡器电路。表 11-1 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 11-3： 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 11-1 下方的“注”。

表 11-1： 定时器振荡器的电容选择

振荡器类型	频率	C1	C2
LP	32 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。

11.3.1 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 OSCCON 寄存器的时钟选择位 SCS<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 OSCCON 寄存器的 IDLEN 位被清零并且执行了 SLEEP 指令，器件将进入 SEC_IDLE 模式。更多详细信息，请参见**第 19.0 节“功耗管理模式”**。

无论何时将 Timer1 振荡器用作时钟源，T1CON 寄存器的 Timer1 系统时钟状态标志位 T1RUN 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。

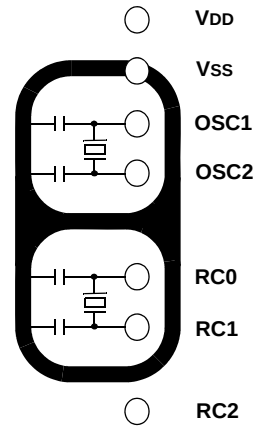
11.3.2 TIMER1 振荡器布线注意事项

Timer1 振荡器电路在工作期间仅消耗极少的电流。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 11-3 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 11-4 所示），或再加一个地平面可能会有帮助。

图 11-4： 带有接地保护环的振荡器电路



注：未按比例绘制。

11.4 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 PIR1 寄存器的 TMR1IF 中断标志位。可以通过将 PIE1 寄存器的 TMR1IE 中断允许位置 1 或清零来允许或禁止该中断。

11.5 使用 CCP 特殊事件触发信号复位 Timer1

如果 CCP 模块配置为在比较模式（CCP1M<3:0> 或 CCP2M<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 14.3.4 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，CCPRH:CCPRL 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：	CCP2 模块产生的特殊事件触发信号不会将PIR1寄存器的TMR1IF中断标志位置1。

11.6 使用 Timer1 作为实时时钟

为 Timer1 外接一个 LP 振荡器（如**第 11.3 节 “ Timer1 振荡器 ”** 中所述），可以允许用户在他们的应用中包括 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程序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大容量电容作为电源时，可省去另外的 RTC 器件和备用电池。

应用代码程序 RTCisr（如例 11-1 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，32.768 kHz 时钟源需要 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 11-1： 使用 TIMER1 中断服务实现实时时钟

```
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 11-2： 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	RABIE	TMR0IF	INT0IF	RABIF	285
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	288
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	288
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	288
TMR1L	Timer1 寄存器的低字节								286
TMR1H	Timer1 寄存器的高字节								286
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	286
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	288
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	288
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	286

图注： — = 未实现，读为 0。 Timer1 模块不使用阴影单元。

12.0 TIMER2 模块

Timer2 模块定时器具有以下特性：

- 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器 （分频比为 1:1、 1:4 和 1:16）
- 可软件编程的后分频器 （分频比为 1:1 到 1:16）
- TMR2 与 PR2 匹配时中断
- 可选择用作 MSSP 模块的移位时钟

此模块由 T2CON 寄存器（寄存器 12-1）控制，此寄存器使能或禁止定时器并配置预分频器和后分频器。可以通过清零 T2CON 寄存器的控制位 TMR2ON 关闭 Timer2，以实现功耗最小。

图 12-1 给出了此模块的简化框图。

12.1 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 T2CON 寄存器的预分频比控制位 T2CKPS<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到 00h，并驱动输出计数器 / 后分频器 （见**第 12.2 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 12-1： 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

12.2 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。此计数器产生的 TMR2 匹配中断标志位为 PIR1 寄存器的 TMR2IF。可以通过将 PIE1 寄存器的 TMR2 匹配中断允许位 TMR2IE 置 1 来允许此中断。

可以通过T2CON寄存器的后分频比控制位T2OUTPS<3:0>在 16 个后分频比选项（从 1:1 到 1:16）中选择其一。

12.3 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 15.0 节 “ 主同步串行口 （MSSP）模块 ”**中提供了更多信息。

图 12-1： TIMER2 框图

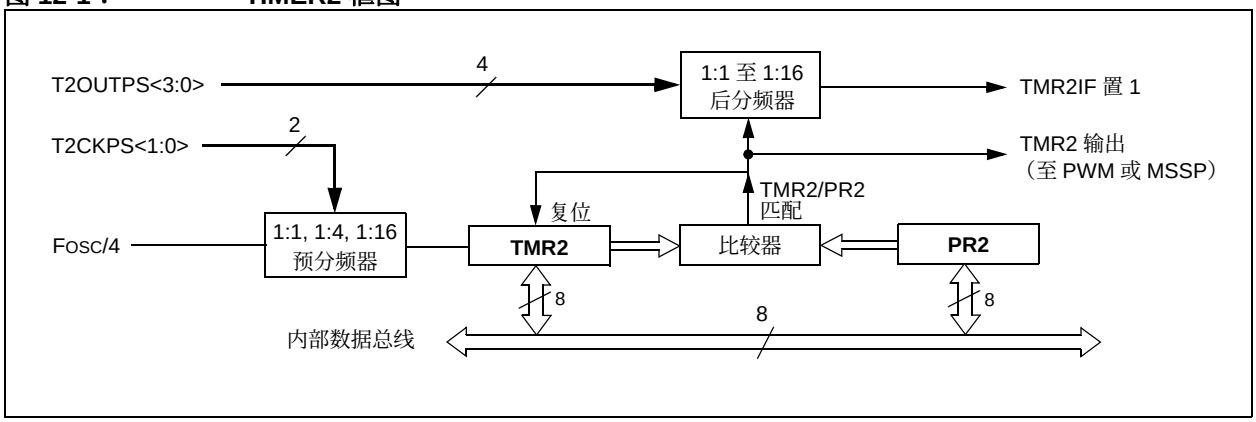


表 12-1： 与 TIMER2 作为定时器 / 计数器相关的寄存器

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	RABIE	TMR0IF	INT0IF	RABIF	285
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	288
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	288
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	288
TMR2	Timer2 寄存器								286
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	286
PR2	Timer2 周期寄存器								286

图注： — = 未实现，读为 0。 Timer2 模块不使用阴影单元。

13.0 TIMER3 模块

Timer3 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 （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
- CCP 特殊事件触发模块复位

图 13-1 给出了 Timer3 模块的简化框图。图 13-2 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 13-1）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 14.1.1 节 “ CCP 模块和定时器资源 ”**）。

寄存器 13-1： T3CON：TIMER3 控制寄存器

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	—	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 **未实现**：读为 0

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 3 **T3CCP1**：ECCP1 的 Timer3 和 Timer1 使能位
1 = Timer3 是 ECCP1 的捕捉 / 比较时钟源
0 = Timer1 是 ECCP1 的捕捉 / 比较时钟源

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：
该位为无关位。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

13.1 Timer3 工作原理

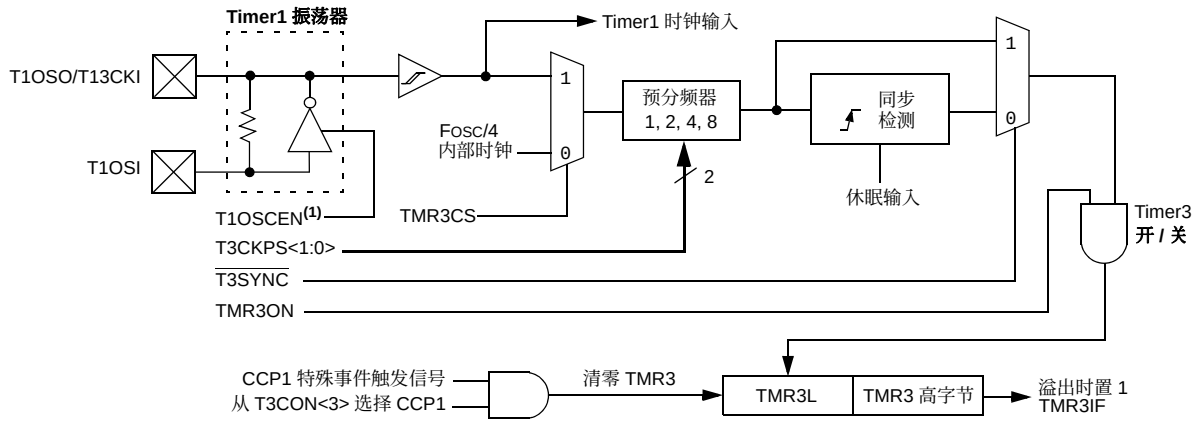
Timer3 可工作在以下三种模式之一：

- 定时器
- 同步计数器
- 异步计数器

工作模式由 T3CON 寄存器的时钟选择位 TMR3CS 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。

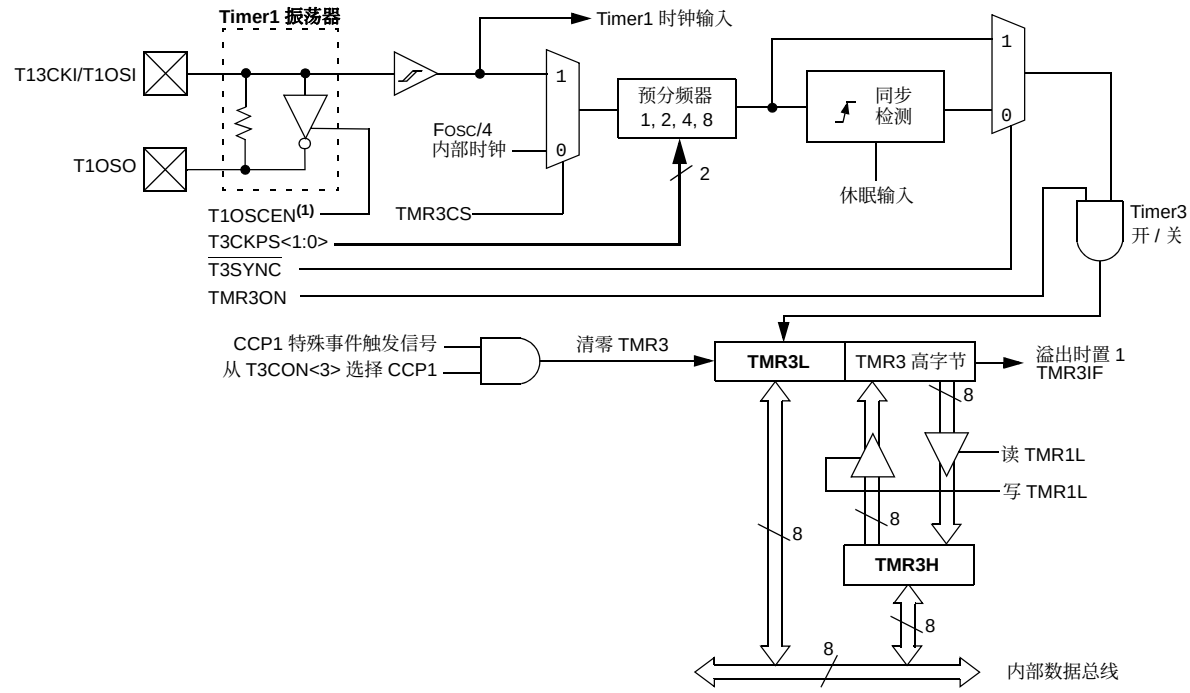
当使能Timer1振荡器时，与RC1/T1OSI和RC0/T1OSO/T13CKI引脚相关的数字电路被禁止。这意味着TRISC<1:0>的值被忽略并且这些引脚将读为 0。

图 13-1： TIMER3 框图



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 13-2：　　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。

13.2　Timer3 的 16 位读 / 写模式

可将 Timer3 配置为 16 位读写模式 （见图 13-2）。当 T3CON 寄存器的 RD16 控制位置 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进位，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 16 位值一次写入 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。

13.3　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 T1CON 寄存器的 T1OSCEN 位置 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 11.0 节 “ Timer1 模块 ”** 中对 Timer1 振荡器进行了描述。

13.4　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 PIR2 寄存器的中断标志位 TMR3IF 中。可以通过将 PIE2 寄存器的 Timer3 中断允许位 TMR3IE 置 1 或清零来允许或禁止该中断。

13.5 使用 CCP 特殊事件触发信号复位 Timer3

如果 CCP1 模块配置为在比较模式（CCP1M<3:0>）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，还将启动 A/D 转换（更多信息，请参见**第 17.2.8 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， CCPR1H:CCPR1L 寄存器对实际上变成了 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。

表 13-1： 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	RABIE	TMR0IF	INT0IF	RABIF	285
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	CCP2IF	288
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	CCP2IE	288
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	CCP2IP	288
TMR3L	Timer3 寄存器的低字节								287
TMR3H	Timer3 寄存器的高字节								287
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	286
T3CON	RD16	—	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	287
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	288
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	288

图注： — = 未实现，读为 0。 Timer3 模块不使用阴影单元。

14.0 增强型捕捉 / 比较 /PWM （ECCP）模块

PIC18F/LF1XK50 器件有一个 ECCP （捕捉 / 比较 /PWM）模块。该模块包含一个 16 位寄存器，可用作 16 位捕捉寄存器、 16 位比较寄存器或 PWM 主 / 从占空比寄存器。

CCP1 实现为具有增强型 PWM 功能的标准 CCP 模块。这些功能包括：

- 提供 2 或 4 路输出通道
- 输出转向
- 可编程极性
- 可编程死区控制
- 自动关闭和重启

第 14.4 节 “ PWM （增强型模式）”将详细讨论增强功能。

寄存器 14-1： CCP1CON：增强型捕捉 / 比较 /PWM 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
P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
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 **P1M<1:0>**：增强型 PWM 输出配置位
如果 CCP1M<3:2> = 00、 01 和 10：
xx = P1A 配置为捕捉 / 比较输入 / 输出； P1B、 P1C 和 P1D 配置为端口引脚
如果 CCP1M<3:2> = 11：
00 = 单输出：P1A、 P1B、 P1C 和 P1D 通过转向控制 （见**第 14.4.7 节 “ 脉冲转向模式 ”**）
01 = 全桥正向输出：P1D 被调制； P1A 有效； P1B 和 P1C 无效
10 = 半桥输出：P1A 和 P1B 被调制，带有死区控制； P1C 和 P1D 配置为端口引脚
11 = 全桥反向输出：P1B 被调制； P1C 有效； P1A 和 P1D 无效

bit 5-4 **DC1B<1:0>**：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：
这些位是 10 位 PWM 占空比的低 2 位。占空比的高 8 位在 CCPR1L 中。

bit 3-0 **CCP1M<3:0>**：增强型 CCP 模式选择位
0000 = 捕捉 / 比较 /PWM 关闭 （复位 ECCP 模块）
0001 = 保留
0010 = 比较模式，匹配时输出电平翻转
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 CCP1 引脚为低电平，比较匹配时输出置 1 （CCP1IF 置 1）
1001 = 比较模式，初始化 CCP1 引脚为高电平，比较匹配时输出清零 （CCP1IF 置 1）
1010 = 比较模式，仅产生软件中断， CCP1 引脚回复到 I/O 状态
1011 = 比较模式，触发特殊事件 （ECCP 复位 TMR1 或 TMR3，启动 A/D 转换， CC1IF 位置 1）
1100 = PWM 模式； P1A 和 P1C 高电平有效； P1B 和 P1D 高电平有效
1101 = PWM 模式； P1A 和 P1C 高电平有效； P1B 和 P1D 低电平有效
1110 = PWM 模式； P1A 和 P1C 低电平有效； P1B 和 P1D 高电平有效
1111 = PWM 模式； P1A 和 P1C 低电平有效； P1B 和 P1D 低电平有效

除了可使用 CCP1CON 寄存器和 ECCP1AS 寄存器提供的扩展模式外，ECCP 模块还有两个与增强型 PWM 操作和自动关闭功能相关的寄存器。它们是：

- PWM1CON （死区延时）
- PSTRCON （输出转向）

14.1 ECCP 输出和配置

取决于所选定的工作模式，增强型 CCP 模块最多有 4 路 PWM 输出。这些指定为 P1A 到 P1D 的输出，可以与 PORTC 的 I/O 引脚复用。输出根据选定的 CCP 工作模式被激活。表 14-2 中总结了引脚分配。

若要将 I/O 引脚配置为 PWM 输出，必须通过设置 P1M<1:0> 和 CCP1M<3:0> 位来选择适当的 PWM 模式。端口引脚的相应 TRISC 方向位也必须设置为输出。

14.1.1 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。Timer1 和 Timer3 适用于工作在捕捉或比较模式下的模块，而 Timer2 适用于工作在 PWM 模式下的模块。

表 14-1： CCP 模式 —— 定时器资源

CCP/ECCP 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 CCP 模块是由 T3CON 寄存器 （寄存器 13-1）中的 “ Timer-to-CCP （将定时器分配给 CCP）” 使能位决定的。图 14-1 总结了这两个模块间的相互关系。在异步计数器模式下，可能无法进行捕捉操作。

14.2 捕捉模式

在捕捉模式下，当相应的 CCP1 引脚发生以下事件时，CCPR1H:CCPR1L 寄存器对捕捉 TMR1 或 TMR3 寄存器的 16 位值。事件定义为下列情况之一：

- 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CCP1CON寄存器的模式选择位CCP1M<3:0>选择。当完成一次捕捉时，中断请求标志位 CCP1IF 置 1 ；它必须用软件清零。如果在读取寄存器 CCPR1 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。

14.2.1 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 CCP1 引脚配置为输入。

注：	如果 CCP1 引脚被配置为输出，则写端口将产生一次捕捉条件。

14.2.2 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 14.1.1 节 “ CCP 模块和定时器资源 ”**）。

14.2.3 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 CCP1IE 中断允许位清零以避免错误中断。应在工作模式改变后清零中断标志位 CCP1IF。

14.2.4 CCP 预分频器

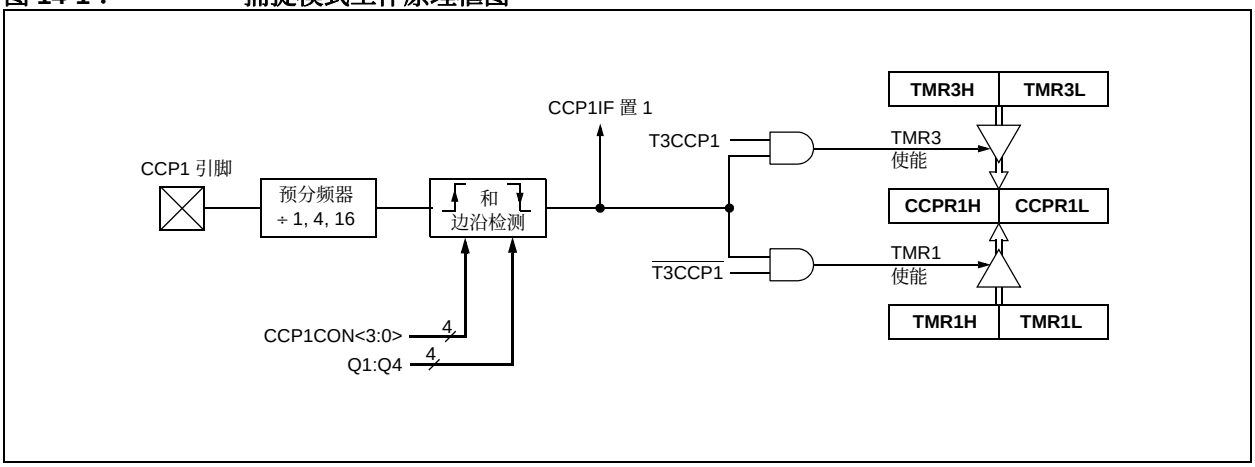
在捕捉模式下有 4 种预分频比设置；它们作为工作模式的一部分由模式选择位（CCP1M<3:0>）指定。只要关闭 CCP 模块或禁止捕捉模式，预分频器计数器就会被清零。这意味着任何复位都会将预分频器计数器清零。

在两个预分频比之间切换会产生中断。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 14-1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 14-1： 改变捕捉预分频比

```
CLRF   CCP1CON     ; Turn CCP module off
MOVLW  NEW_CAPT_PS ; Load WREG with the
                   ; new prescaler mode
                   ; value and CCP ON
MOVWF  CCP1CON     ; Load CCP1CON with
                   ; this value
```

图 14-1： 捕捉模式工作原理框图



14.3 比较模式

在比较模式下，16 位 CCPR1 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， CCP1 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（CCP1M<3:0>）的值。同时，中断标志位 CCP1IF 置 1。

14.3.1 CCP 引脚配置

用户必须通过将相应的 TRIS 位清零，将 CCP1 引脚配置为输出。

注： 清零 CCP1CON 寄存器会将 CCP1 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 I/O 数据锁存器。

14.3.2 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。

14.3.3 软件中断模式

当选择了“ 产生软件中断 ”模式（CCP1M<3:0> = 1010）时，CCP1 引脚不受影响。只会影响 CCP1IF 中断标志。

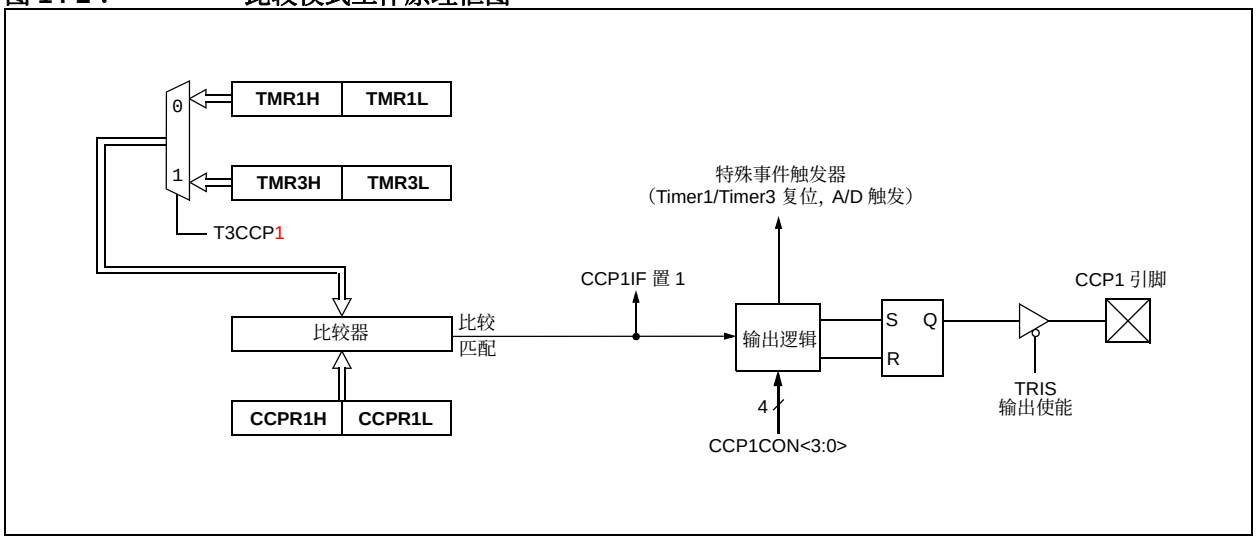
14.3.4 特殊事件触发器

CCP 模块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 （CCP1M<3:0> = 1011），使能特殊事件触发器。

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器寄存器对复位。这样CCPR1 寄存器可用作两个定时器的可编程周期寄存器。

特殊事件触发器还能启动 A/D 转换。要实现此功能，必须首先使能 A/D 转换器。

图 14-2： 比较模式工作原理框图



14.4 PWM （增强型模式）

增强型 PWM 模式可以在最多 4 个输出引脚上产生 PWM 信号，最高可达 10 位分辨率。可通过 4 种 PWM 输出模式实现：

- 单 PWM
- 半桥 PWM
- 全桥 PWM，正向模式
- 全桥 PWM，反向模式

要选择增强型 PWM 模式，CCP1CON 寄存器的 P1M 位必须适当置 1。

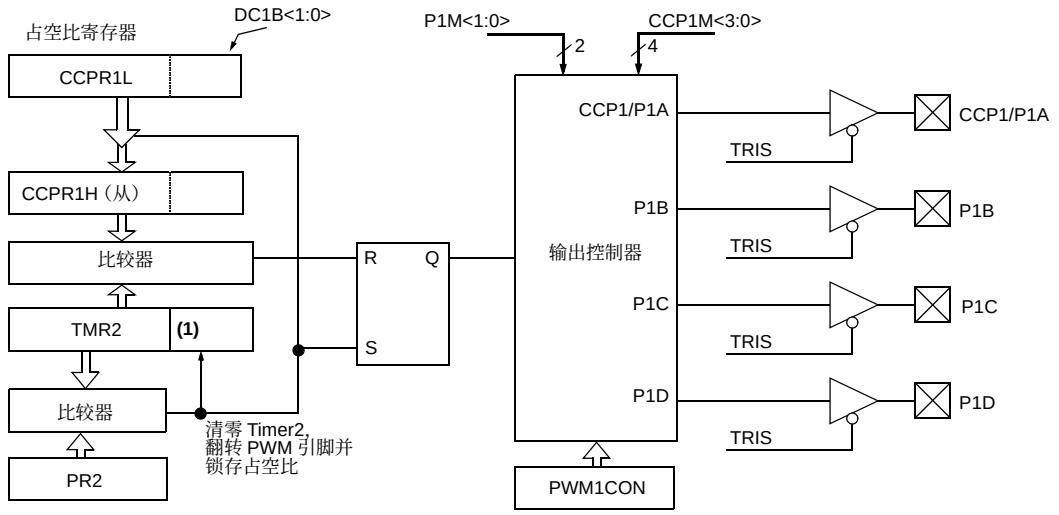
PWM 输出与 I/O 引脚复用，指定为 P1A、P1B、P1C 和 P1D。PWM 引脚的极性是可配置的，通过将 CCP1CON 寄存器中的 CCP1M 位适当置 1 来选择。

表 14-2 给出了每种增强型 PWM 模式的引脚分配。

图 14-3 给出了增强型 PWM 模块的简化框图的示例。

注： 为防止在第一次使能 PWM 时产生不完整的波形，ECCP 模块在产生 PWM 信号前会等待直到新的 PWM 周期开始。

图 14-3： 增强型 PWM 模式的简化框图示例



注 1： 8 位定时器 TMR2 寄存器与 2 位内部 Q 时钟或预分频器中的 2 位一起构成 10 位时基。

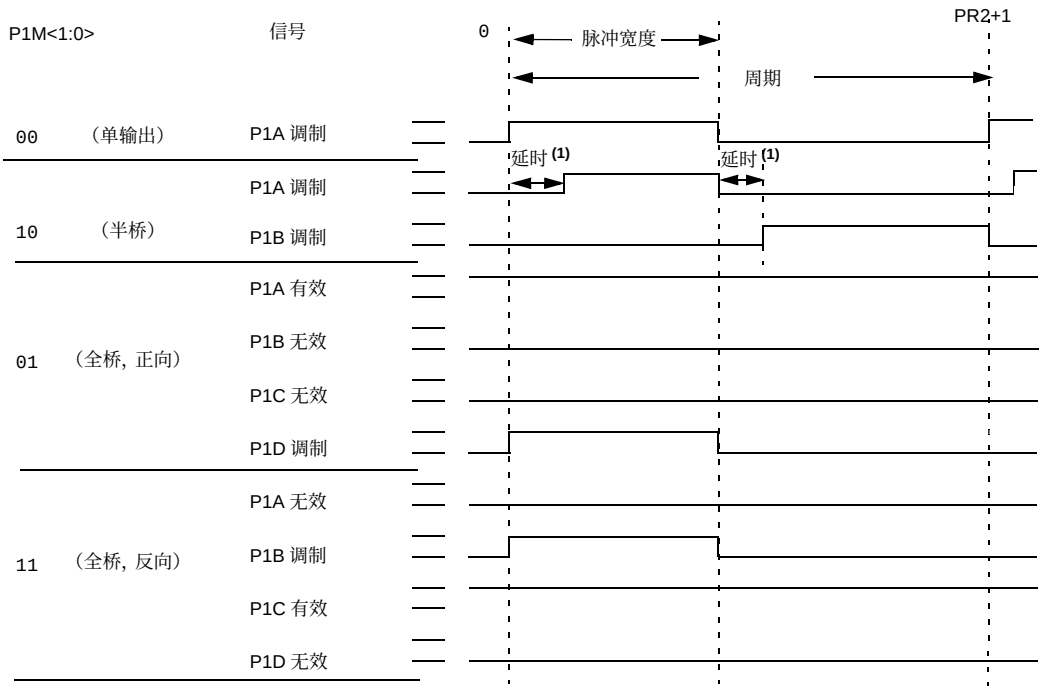
注 1： 每个 PWM 输出的 TRIS 寄存器值必须进行适当配置。
2： 增强型 PWM 模式没有使用的任何引脚均可用于备用引脚功能。

表 14-2： 各种 PWM 增强型模式的引脚分配示例

ECCP 模式	P1M<1:0>	CCP1/P1A	P1B	P1C	P1D
单	00	有 (1)	有 (1)	有 (1)	有 (1)
半桥	10	有	有	无	无
全桥，正向	01	有	有	有	有
全桥，反向	11	有	有	有	有

注 1： 输出通过单模式下的脉冲转向使能。请参见寄存器 14-4。

图 14-4：　　PWM （增强型模式）输出关系示例 （高电平有效状态）

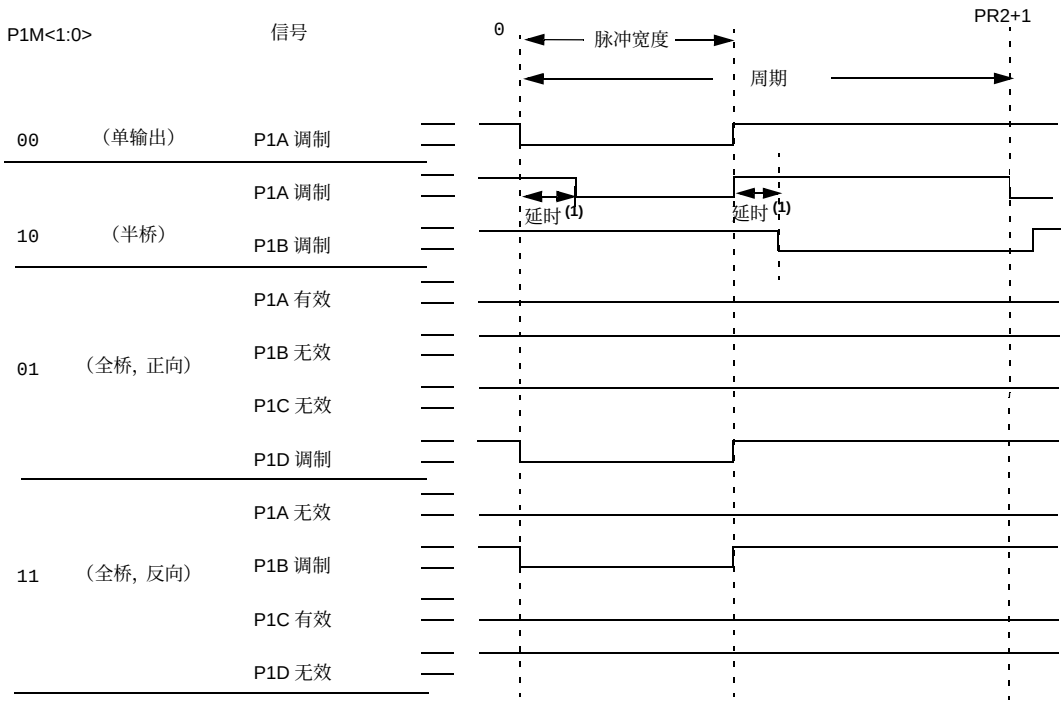


关系：

- 周期 = 4 * Tosc * (PR2 + 1) * (TMR2 预分频值)
- 脉冲宽度 = Tosc * (CCPR1L<7:0>:CCP1CON<5:4>) * (TMR2 预分频值)
- 延时 = 4 * Tosc * (PWM1CON<6:0>)

注　1： 使用 PWM1CON 寄存器对死区延时编程 （**第 14.4.6 节 “ 可编程死区延时模式 ”**）。

图 14-5：　　增强型 PWM 输出关系示例 （低电平有效状态）



关系：

- 周期 = 4 * Tosc * (PR2 + 1) * (TMR2 预分频值)
- 脉冲宽度 = Tosc * (CCPR1L<7:0>:CCP1CON<5:4>) * (TMR2 预分频值)
- 延时 = 4 * Tosc * (PWM1CON<6:0>)

注　1： 使用 PWM1CON 寄存器对死区延时编程 （**第 14.4.6 节 “ 可编程死区延时模式 ”**）。

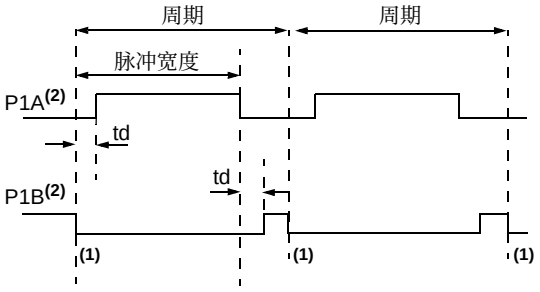
14.4.1 半桥模式

在半桥模式下，有两个引脚用作输出驱动推挽式负载。CCP1/P1A 引脚输出 PWM 输出信号，P1B 引脚输出互补的 PWM 输出信号（见图 14-6）。这种模式可用于半桥应用（如图 14-7 所示），或者用于全桥应用，这种情况下使用两个 PWM 信号调制 4 个功率开关。

在半桥模式下，可编程死区延时可用来防止半桥功率器件中流过直通（Shoot-through）电流。PWM1CON 寄存器的 PDC<6:0> 位的值对应在输出被驱动为有效之前的指令周期数。如果这个值比占空比大，则在整个周期中相应的输出保持为无效。关于死区延时操作的更多详细信息，请参见**第 14.4.6 节 “ 可编程死区延时模式 ”**。

由于 P1A 和 P1B 输出与端口数据锁存器是复用的，相关的TRIS位必须清零，从而将P1A和P1B配置为输出。

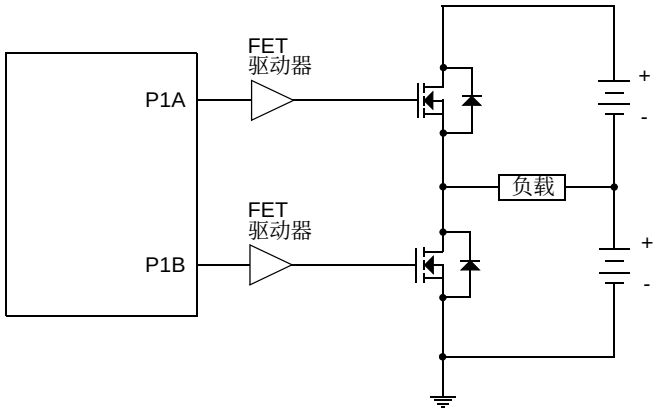
图 14-6： 半桥 PWM 输出示例



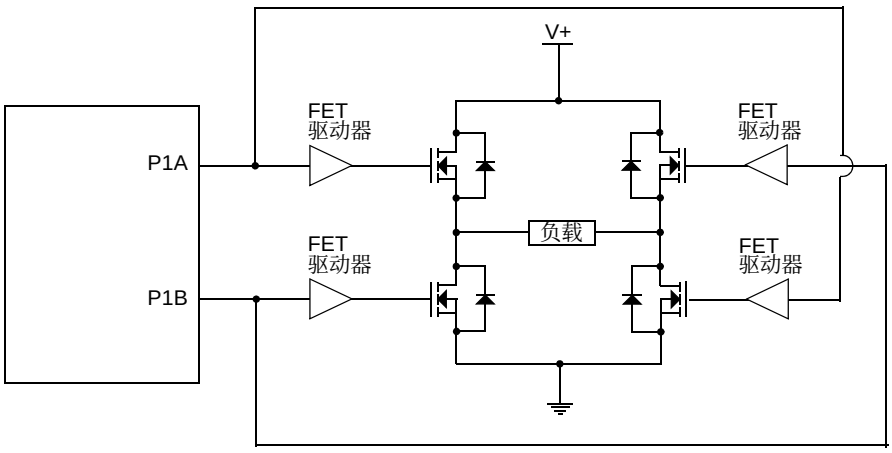
注 **1：** 此时 TMR2 寄存器和 PR2 寄存器的值相等。
2： 图中输出信号为高电平有效。

图 14-7： 半桥应用示例

标准半桥电路（“ 推挽式 ”）



驱动全桥电路的半桥输出



14.4.2 全桥模式

在全桥模式下， 4 个引脚都用作输出。全桥应用的示例如图 14-8 所示。

在正向模式下，引脚 CCP1/P1A 被驱动为有效状态，引脚 P1D 被调制，而 P1B 和 P1C 将被驱动为无效状态，如图 14-9 所示。

在反向模式下，P1C 被驱动为有效状态，引脚 P1B 被调制，而P1A和P1D将被驱动为无效状态，如图14-9所示。

P1A、P1B、P1C 和 P1D 输出与端口数据锁存器复用。相关的 TRIS 位必须清零，从而将 P1A、P1B、P1C 和 P1D 引脚配置为输出。

图 14-8： 全桥应用示例

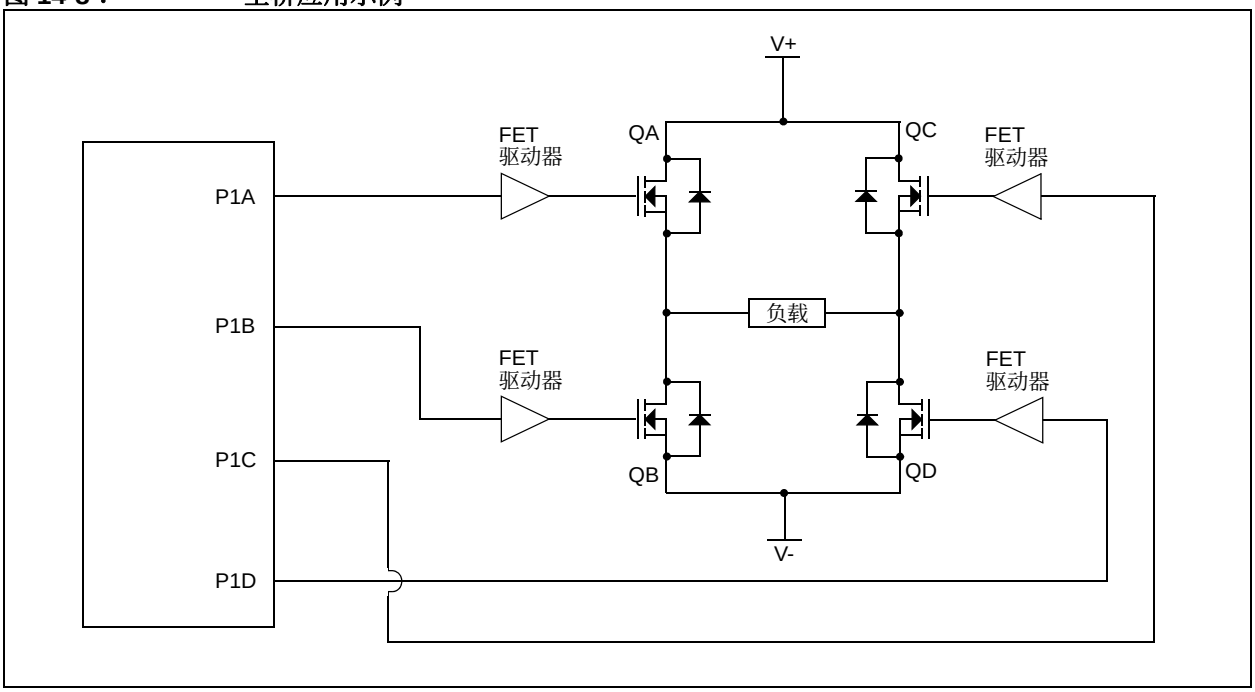
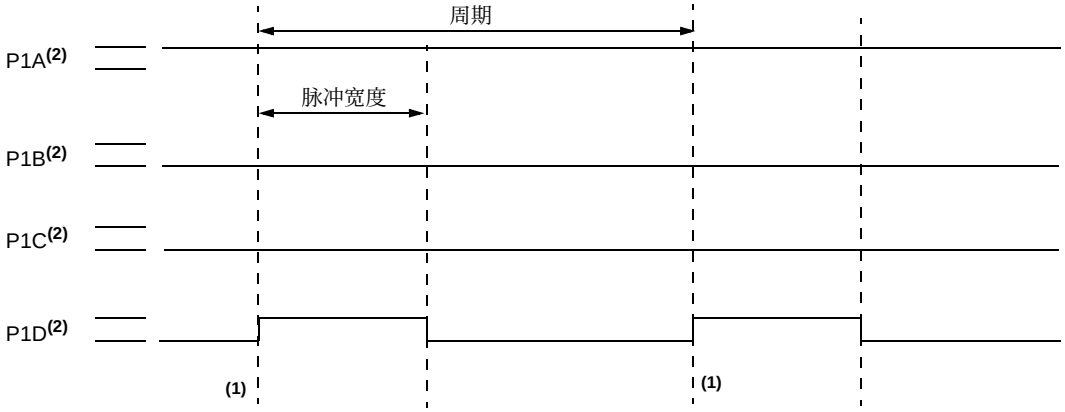
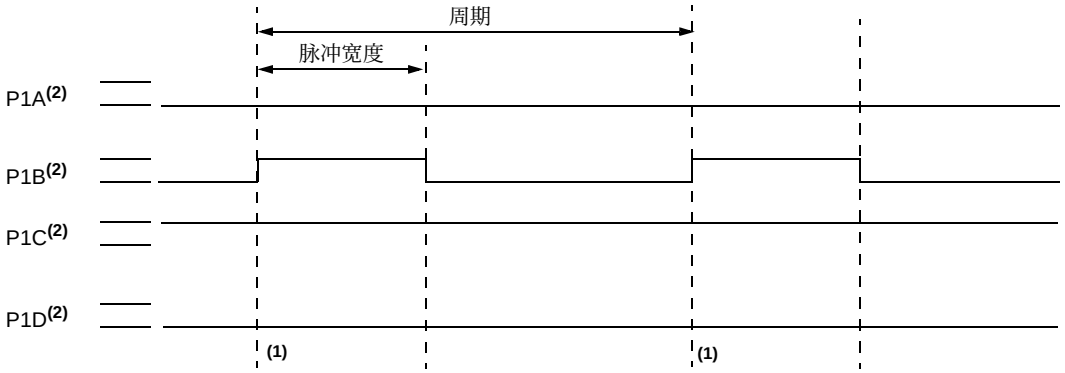


图 14-9：　　全桥 PWM 输出示例

正向模式



反向模式



注　1： 此时 TMR2 寄存器和 PR2 寄存器的值相等。

2： 图中输出信号为高电平有效。

14.4.2.1 全桥模式中的方向改变

在全桥模式下，CCP1CON 寄存器中的 P1M1 位允许用户控制正 / 反方向。当应用程序固件改变这个方向控制位时，模块将在下一个 PWM 周期改用新的方向。

通过改变 CCP1CON 寄存器的 P1M1 位，可以用软件改变方向。以下序列在当前 PWM 周期结束前发生：

- 调制输出 （P1B 和 P1D）进入无效状态。
- 相关的未调制输出 （P1A 和 P1C）被切换到以相反的方向驱动。
- PWM 调制在下一个周期开始继续。

关于该序列的说明，请参见图 14-10。

全桥模式不提供死区延时。因为始终只有一个输出被调制，所以通常不需要死区延时。有一种情况需要死区延时。这一情况发生在以下两个条件同时满足时：

1. 当输出的占空比达到或者接近 100% 时， PWM 输出方向改变。
2. 功率开关（包括功率器件和驱动电路）的关断时间比导通时间要长。

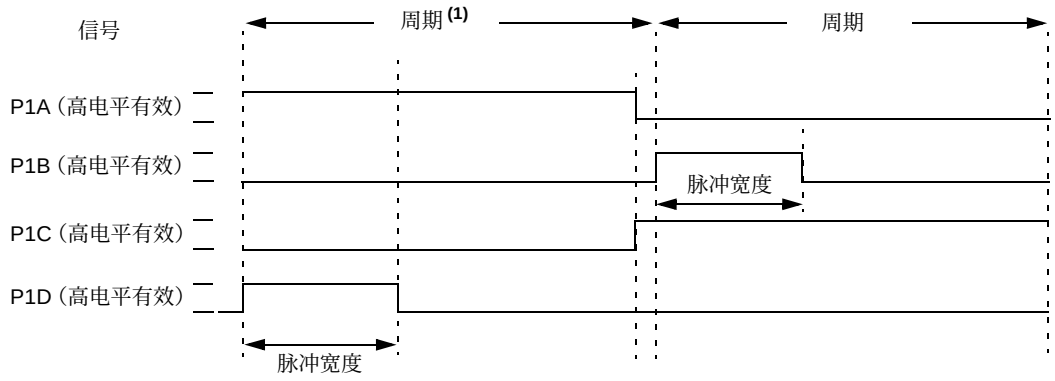
在图 14-11 所示的示例中，在占空比接近 100% 时，PWM 方向从正向改变到反向。在这个示例中，在时间 t1，输出 P1A 和 P1D 变为无效，而输出 P1C 变为有效。因为功率器件的关断时间比导通时间要长，在 “ t ” 时间内，功率器件 QC 和 QD 中可能流过直通电流（见图 14-8）。当 PWM 方向从反向改变到正向时，功率器件 QA 和 QB 也将出现相同的现象。

如果应用中需要在高占空比时改变 PWM 方向，避免直通电流的两种可能方法是：

1. 在改变方向之前的一个 PWM 周期降低 PWM 占空比。
2. 使用开关驱动电路，使开关管的关断时间比导通时间短。

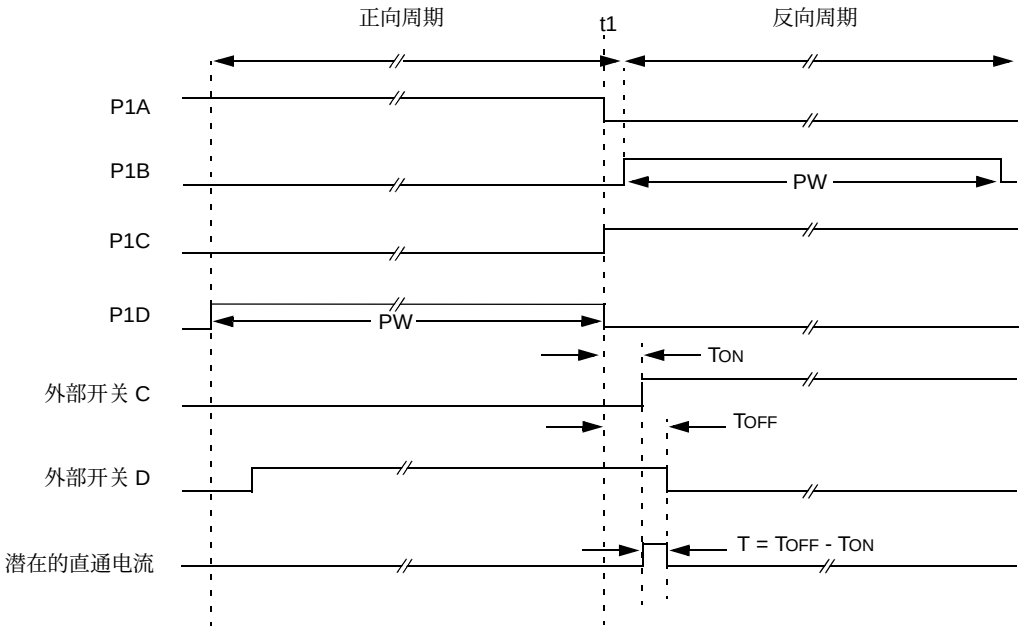
也可能存在其他避免直通电流的方案。

图 14-10： PWM 方向改变的示例



注 1： 可在 PWM 周期的任意时刻，写 CCP1CON 寄存器的方向位 P1M1。

图 14-11： 在占空比接近 100% 时改变 PWM 方向的示例



注 1： 图中所有信号均为高电平有效。

2： TON 是功率开关 QC 及其驱动电路的导通延时。

3： TOFF 是功率开关 QD 及其驱动电路的关断延时。

14.4.3 启动注意事项

当使用 PWM 模式时，应用硬件必须在 PWM 输出引脚上外接适当的上拉和 / 或下拉电阻。

注：	当单片机退出复位状态时，所有 I/O 引脚呈高阻态。外部电路必须保持功率开关器件处于截止状态，直到单片机将 I/O 引脚驱动为适当的信号电平，或者激活 PWM 输出为止。

CCP1CON 寄存器的 CCP1M<1:0> 位允许用户为每一对 PWM 输出引脚（P1A/P1C 和 P1B/P1D）选择 PWM 输出信号是高电平有效还是低电平有效。PWM 输出极性必须在使能 PWM 引脚输出驱动器之前选择。由于可能导致应用电路的损坏，因此不推荐在使能 PWM 引脚输出驱动器的同时改变极性配置。

当 PWM 模块初始化时，P1A、P1B、P1C 和 P1D 输出锁存器可能不在正确的状态。这样在使能增强型 PWM 模式的同时使能 PWM 引脚输出驱动器，可能损坏应用电路。应首先将增强型 PWM 模式配置为正确的输出模式并经过一个完整的 PWM 周期之后，再使能 PWM 引脚输出驱动器。当第二个 PWM 周期开始时，PIR1 寄存器的 TMR2IF 位置 1 表明一个完整的 PWM 周期结束了。

14.4.4 增强型 PWM 自动关闭模式

PWM模式支持自动关闭模式，当外部关闭事件发生时将禁止 PWM 输出。自动关闭模式将 PWM 输出引脚置于预先确定的状态。该模式用于防止 PWM 损坏应用。

通过使用 ECCPAS 寄存器的 ECCPAS<2:0> 位来选择自动关闭源。关闭事件由以下条件产生：

- INT0 引脚上的逻辑 0
- 比较器 （Cx）输出上的逻辑 1

关闭条件由 ECCPAS 寄存器的 ECCPASE （自动关闭事件状态）位指定。如果该位为 0， PWM 引脚正常工作。如果该位为 1， PWM 输出处于关闭状态。

当关闭事件发生时，会发生以下两件事：

ECCPASE 位被设为 1。 ECCPASE 将保持置 1 直到由固件清零或发生自动重启 （见**第 14.4.5 节 “ 自动重启模式 ”**）。

使能的 PWM 引脚被异步置为其关闭状态。PWM 输出引脚被分组为 [P1A/P1C] 和 [P1B/P1D] 对。每对引脚的状态由 ECCPAS 寄存器的 PSSAC 和 PSSBD 位决定。每对引脚可以设置为以下 3 种状态之一：

- 驱动逻辑 1
- 驱动逻辑 0
- 三态 （高阻态）

寄存器 14-2： ECCP1AS：增强型捕捉 / 比较 /PWM 自动关闭控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0
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 **ECCPASE：**ECCP 自动关闭事件状态位
1 = 发生了关闭事件； ECCP 输出为关闭状态
0 = ECCP 输出正常工作

bit 6-4 **ECCPAS<2:0>：**ECCP 自动关闭源选择位
000 = 禁止自动关闭
001 = 比较器 C1OUT 输出为高电平
010 = 比较器 C2OUT 输出为高电平
011 = 比较器 C1OUT 或 C2OUT 为高电平
100 = INT0 引脚电压为 VIL
101 = INT0 引脚电压为 VIL 或比较器 C1OUT 输出为高电平
110 = INT0 引脚电压为 VIL 或比较器 C2OUT 输出为高电平
111 = INT0 引脚电压为 VIL，或比较器 C1OUT 或比较器 C2OUT 输出为高电平

bit 3-2 **PSSACn：**引脚 P1A 和 P1C 关闭状态控制位
00 = 驱动引脚 P1A 和 P1C 为 0
01 = 驱动引脚 P1A 和 P1C 为 1
1x = 引脚 P1A 和 P1C 为三态

bit 1-0 **PSSBDn：**引脚 P1B 和 P1D 关闭状态控制位
00 = 驱动引脚 P1B 和 P1D 为 0
01 = 驱动引脚 P1B 和 P1D 为 1
1x = 引脚 P1B 和 P1D 为三态

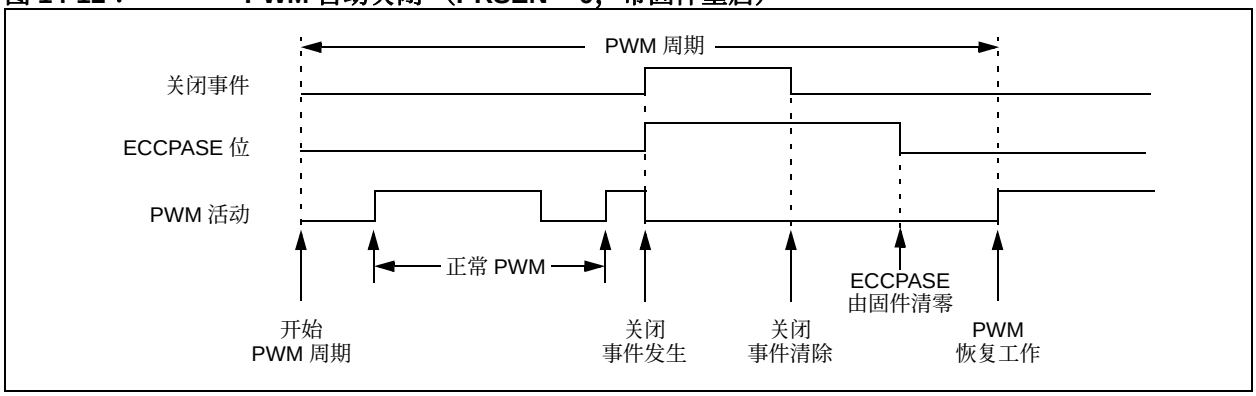
注 1： 自动关闭条件是基于电平的信号，而不是基于边沿的信号。只要电平存在，自动关闭就将持续。

2： 当自动关闭条件持续时，禁止写 ECCPASE 位。

3： 一旦自动关闭条件被移除并且 PWM 重启（通过固件或自动重启），PWM 信号将总是在下一个 PWM 周期重启。

4： 由比较器输出或 INT 引脚事件引起的自动关闭事件发生之前，可通过将 CCPxASE 位置 1 用固件触发软件关闭。自动重启功能跟踪仅由比较器输出或 INT 引脚事件引起的关闭的有效状态，因此，如果此时被使能，在下一个 PWM 周期开始时，自动重启功能会立即清零该位，并重新启动 ECCP 模块。

图 14-12： PWM 自动关闭 （PRSEN = 0，带固件重启）

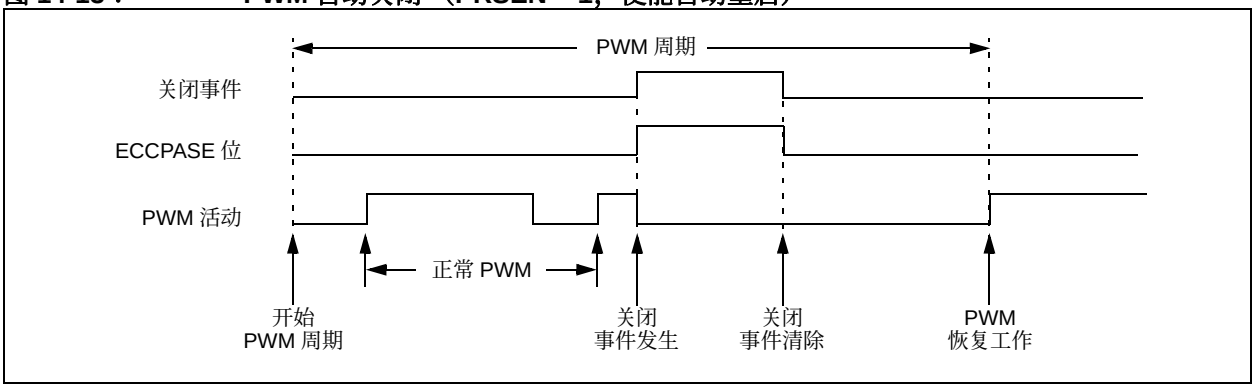


14.4.5 自动重启模式

一旦自动关闭条件被移除，增强型 PWM 可被配置为自动重启 PWM 信号。通过将 PWM1CON 寄存器中的 PRSEN 位置 1 使能自动重启。

如果使能了自动重启，只要自动关闭条件有效，ECCPASE 位将保持置 1。当自动关闭条件被移除时，ECCPASE 位将由硬件清零并继续正常工作。

图 14-13： PWM 自动关闭 （PRSEN = 1，使能自动重启）

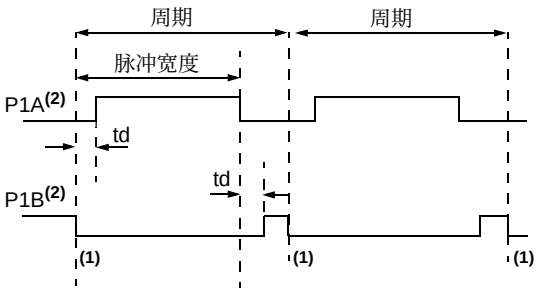


14.4.6 可编程死区延时模式

在所有功率开关管都以 PWM 频率调制的半桥应用中，功率开关管关断通常比导通需要更多的时间。如果上下两个功率开关管在同一时间开关（一个导通，另一个关断），那么在一段很短的时间里，两个开关管可能同时导通，直到一个开关管完全关断为止。在这短暂的时间中，两个功率开关管中可能流过较高的电流（直通电流），将逆变桥的电源与地短路。为避免开关过程中可能会出现的破坏性直通电流，通常需要延迟功率开关管的导通，保证在另一个开关管完全关断之后，再导通相应的功率开关管。

在半桥模式下，可采用数字可编程死区延时来避免出现损坏逆变桥功率开关管的直通电流。在信号从无效状态切换到有效状态时增加延时。请参见图 14-14。PWM1CON 寄存器（寄存器 14-3）的低 7 位以单片机指令周期（TCY 或 4 TOSC）为单位设置延时。

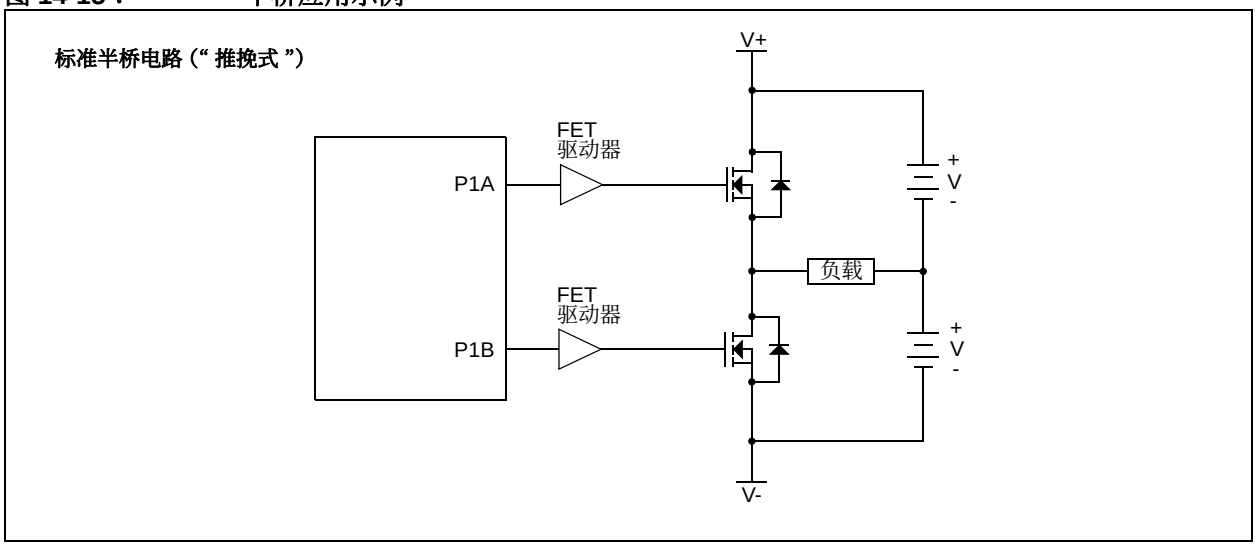
图 14-14： 半桥 PWM 输出示例



td = 死区延时

注 **1：** 此时 TMR2 寄存器和 PR2 寄存器的值相等。
2： 图中输出信号为高电平有效。

图 14-15： 半桥应用示例



寄存器 14-3： PWM1CON：增强型 PWM 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
PRSEN	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
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 **PRSEN：**PWM 重启使能位
1 = 自动关闭时，一旦关闭事件消失， ECCPASE 位自动清零； PWM 自动重启
0 = 自动关闭时， ECCPASE 必须用软件清零以重启 PWM

bit 6-0 **PDC<6:0>：**PWM 延时计数位
PDCn = 在 PWM 信号**应该**转换为有效的预定时间和转换为有效的**实际**时间之间的 Fosc/4 (4 * Tosc) 周期数

14.4.7 脉冲转向模式

在单输出模式下，脉冲转向允许任何 PWM 引脚为调制信号。此外，多个引脚上可以同时使用同一 PWM 信号。

一旦选择了单输出模式（CCP1CON 寄存器的 CCP1M<3:2> = 11 和 P1M<1:0> = 00），通过将 PSTRCON 寄存器的相应 STR<D:A> 位置 1，用户固件可将同一 PWM 信号传送到 1、 2、 3 或 4 个输出引脚，如表 14-2 所示。

注：	必须将相关的 TRIS 位设为输出（0）以使能引脚输出驱动器，从而在引脚上看到 PWM 信号。

当 PWM 转向模式有效时，CCP1CON 寄存器的 CCP1M<1:0>位将为P1<D:A>引脚选择PWM输出极性。

PWM 自动关闭操作也适用于 PWM 转向模式，如**第 14.4.4 节“增强型 PWM 自动关闭模式”**中所述。自动关闭事件只对使能 PWM 输出的引脚有影响。

寄存器 14-4： PSTRCON：脉冲转向控制寄存器 (1)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1
—	—	—	STRSYNC	STRD	STRC	STRB	STRA
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 **STRSYNC：**转向同步位
1 = 在下一个 PWM 周期发生输出转向更新
0 = 在指令周期边界的开始发生输出转向更新

bit 3 **STRD：**转向使能位 D
1 = P1D 引脚的 PWM 波形极性受 CCP1M<1:0> 控制
0 = P1D 引脚被分配为端口引脚

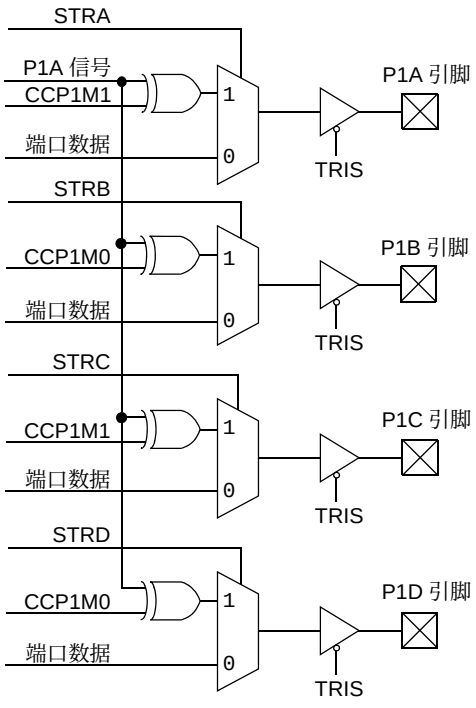
bit 2 **STRC：**转向使能位 C
1 = P1C 引脚的 PWM 波形极性受 CCP1M<1:0> 控制
0 = P1C 引脚被分配为端口引脚

bit 1 **STRB：**转向使能位 B
1 = P1B 引脚的 PWM 波形极性受 CCP1M<1:0> 控制
0 = P1B 引脚被分配为端口引脚

bit 0 **STRA：**转向使能位 A
1 = P1A 引脚的 PWM 波形极性受 CCP1M<1:0> 控制
0 = P1A 引脚被分配为端口引脚

注 1： PWM 转向模式仅在 CCP1CON 寄存器位 CCP1M<3:2> = 11 和 P1M<1:0> = 00 时可用。

图 14-16：　　转向简化框图



注　**1：** 当 CCP1CON 寄存器位 P1M<1:0> = 00 和 CCP1M<3:2> = 11 时，端口输出如图所示配置。

2： 单 PWM 输出需要将至少一个 STRx 位置 1。

14.4.7.1 转向同步

当转向事件发生时，PSTRCON 寄存器的 STRSYNC 位向用户提供两种选择。当 STRSYNC 位为 0 时，转向事件将发生在写 PSTRCON 寄存器指令结束时。在这种情况下， P1<D:A> 引脚的输出信号可能是一个不完整的 PWM 波形。用户固件需要立即从引脚移除 PWM 信号时，该操作非常有用。

当 STRSYNC 位为 1 时，在下一个 PWM 周期的开始将发生有效的转向更新。此时，开 / 关 PWM 输出，使其转向，将始终产生一个完整的 PWM 波形。

图 14-17 和 14-18 是根据 STRSYNC 设置的 PWM 转向时序图。

图 14-17： 指令结束时发生的转向事件的示例 （STRSYNC = 0）

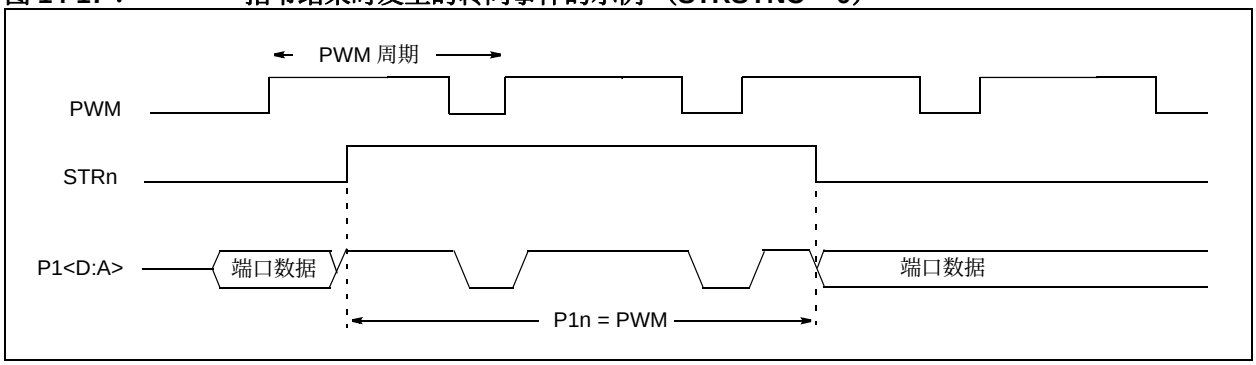
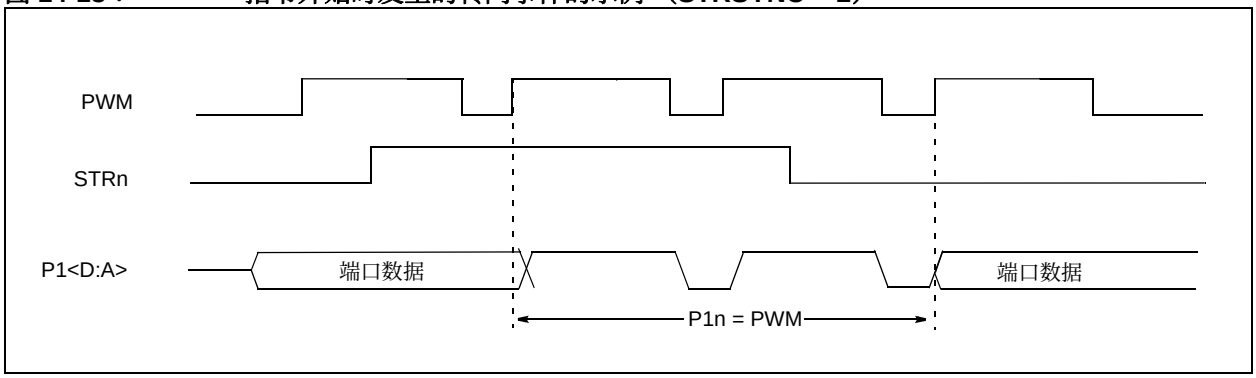


图 14-18： 指令开始时发生的转向事件的示例 （STRSYNC = 1）



14.4.8 在功耗管理模式下的操作

在休眠模式下，所有时钟源都被禁止。 Timer2 不再递增，模块的状态也不会改变。如果 ECCP 引脚正在驱动一个值，则会继续驱动该值。当器件被唤醒时，将从该状态继续。如果使能了双速启动，来自 HFINTOSC 和后分频器的初始启动频率可能不会立即稳定。

在 PRI_IDLE 模式下，主时钟将继续作为 ECCP 模块的时钟源，保持不变。在所有其他功耗管理模式下，选定的功耗管理模式时钟将作为 Timer2 的时钟源。功耗管理模式下时钟的频率很可能与主时钟频率不同。

14.4.8.1 使用故障保护时钟监视器操作

如果使能了故障保护时钟监控器，时钟故障将强制器件进入 RC_RUN 功耗管理模式，并将 PIR2 寄存器的 OSCFIF 位置 1。 ECCP 将从内部振荡器时钟源获取时钟信号，可能与主时钟的时钟频率不同。

更多详细信息，请参见前面的章节。

14.4.9 复位的影响

上电复位及后续的复位都将强制所有端口为输入模式，并强制 CCP 寄存器为复位状态。

这将强制增强型 CCP 模块复位到与标准 CCP 模块兼容的状态。

表 14-3：　　与 ECCP1 模块和 TIMER1 到 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	RABIE	TMR0IF	INT0IF	RABIF	285
RCON	IPEN	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	284
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	288
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	288
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	288
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	288
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	288
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	—	288
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	288
TMR1L	Timer1 寄存器的低字节								286
TMR1H	Timer1 寄存器的高字节								286
T1CON	RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{T1SYNC}$	TMR1CS	TMR1ON	286
TMR2	Timer2 寄存器								286
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	286
PR2	Timer2 周期寄存器								286
TMR3L	Timer3 寄存器的低字节								287
TMR3H	Timer3 寄存器的高字节								287
T3CON	RD16	—	T3CKPS1	T3CKPS0	T3CCP1	$\overline{T3SYNC}$	TMR3CS	TMR3ON	287
CCPR1L	捕捉 / 比较 /PWM 寄存器 1 的低字节								287
CCPR1H	捕捉 / 比较 /PWM 寄存器 1 的高字节								287
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	287
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0	287
PWM1CON	PRSEN	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0	287

图注：　— = 未实现，读为 0。 ECCP 操作期间不使用阴影单元。

15.0 主同步串行口 （MSSP）模块

15.1 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器件进行通信的串行接口。这些外设器件可以是串行EEPROM、移位寄存器、显示驱动器和 A/D 转换器等。MSSP 模块有下列两种工作模式：

- 串行外设接口 （Serial Peripheral Interface， SPI）
- I^2C™
  - 全功能主模式
  - 从模式 （支持广播地址呼叫）

I^2C 接口硬件上支持下列模式：

- 主模式
- 多主器件模式
- 从模式

15.2 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：

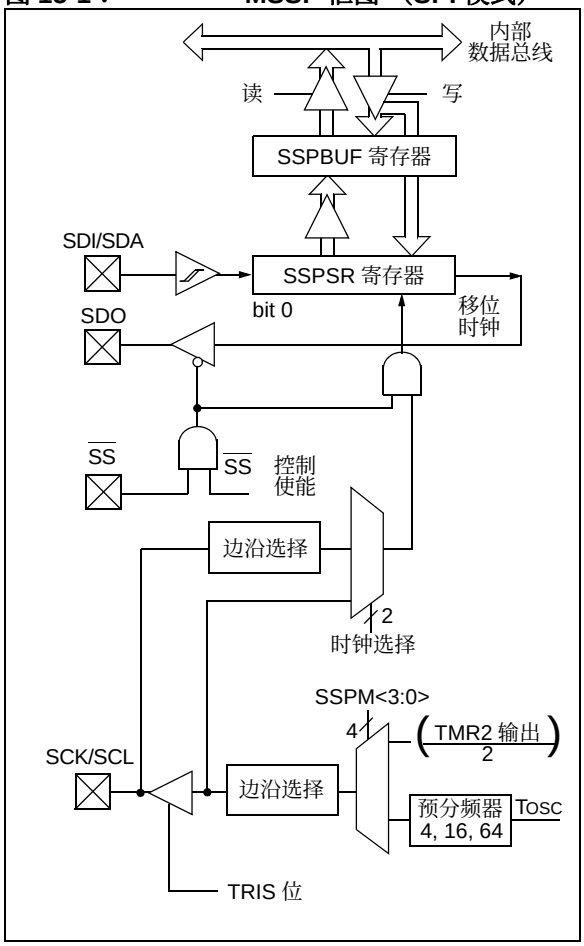
- 串行数据输出 ——SDO
- 串行数据输入 ——SDI
- 串行时钟 ——SCK

此外，当处于从工作模式时要使用第 4 个引脚：

- 从选择 ——$\overline{SS}$

图15-1给出了MSSP模块在SPI模式下的工作原理框图。

图 15-1： MSSP 框图 （SPI 模式）



15.2.1 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：

- SSPCON1—— 控制寄存器
- SSPSTAT—— 状态寄存器
- SSPBUF—— 串行接收 / 发送缓冲区
- SSPSR—— 移位寄存器 （不可直接访问）

SSPCON1 和 SSPSTAT 是 SPI 模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 是用来将数据移入和移出的移位寄存器。SSPBUF 提供对 SSPSR 寄存器的间接访问。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 15-1： SSPSTAT：MSSP 状态寄存器 （SPI 模式）

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	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 / $\overline{地址}$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 SSPCON1 寄存器的 CKP 位设置。

寄存器 15-2： 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	SSPEN	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 （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>：**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用于保留或仅在 I^2C 模式下使用。

15.2.2 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）

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，SSPSTAT 寄存器的缓冲区满检测位 BF 和中断标志位 SSPIF 被置 1。这种双重缓冲数据接收方式 （通过 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。当 SSPBUF 寄存器正在发送 / 接收数据时，对它写入的任何数据都将被忽略，同时 SSPCON1 寄存器的写冲突检测位 WCOL 被置 1。用户软件必须将 WCOL 位清零才能使以后对 SSPBUF 寄存器的写入成功完成。

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。SSPSTAT 寄存器的缓冲区满位 BF 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。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 15-1 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，MSSP 状态寄存器（SSPSTAT）用于指示各种状态。

例 15-1： 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
```

15.2.3 使能 SPI I/O

要使能串口，SSPCON1 寄存器的 SSP 使能位 SSPEN 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。要让上述引脚充当串口，必须正确设置引脚的数据方向位 （在 TRIS 寄存器中）：

- SDI 由 SPI 模块自动控制
- SDO 必须将对应的 TRIS 位清零
- SCK （主模式）必须将对应的 TRIS 位清零
- SCK （从模式）必须将对应的 TRIS 位置 1
- $\overline{SS}$ 必须将对应的 TRIS 位置 1

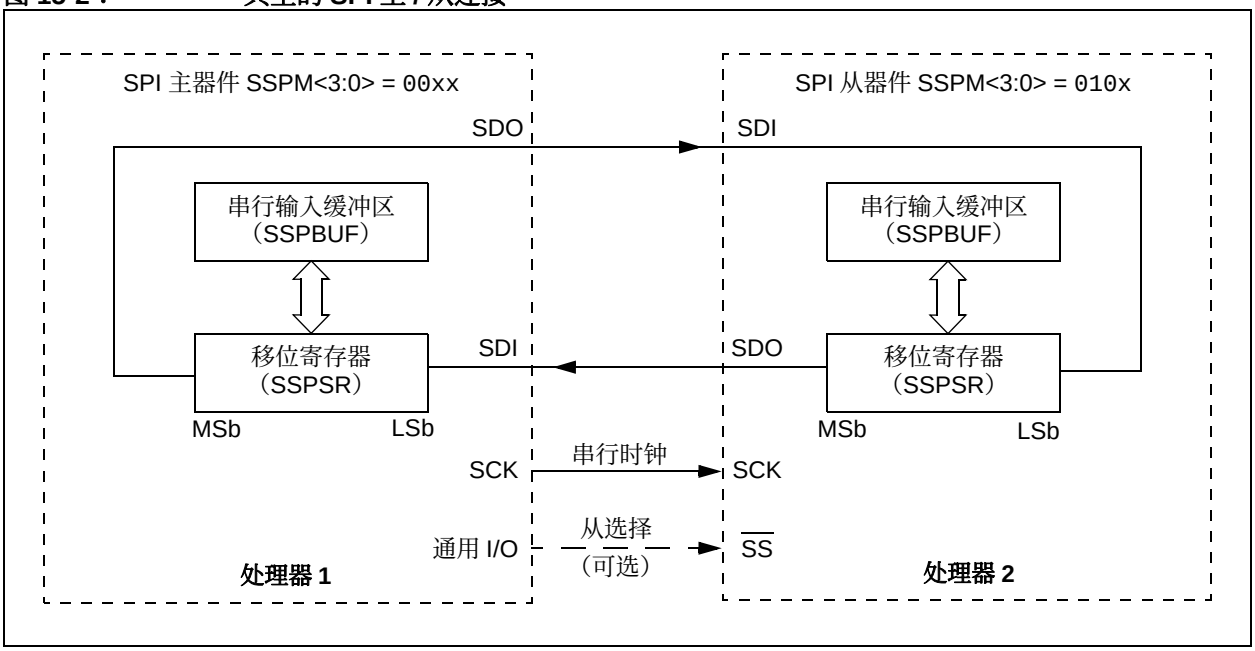
对于不需要的串口功能，可通过将对应的数据方向寄存器 （TRIS）设置为相反值来屏蔽。

15.2.4 典型连接

图 15-2 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。在两个处理器的移位寄存器之间，数据在编程设定的时钟边沿被传送，并在相反的时钟边沿被锁存。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 15-2： 典型的 SPI 主 / 从连接



15.2.5 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 15-2 中的处理器 2）应在何时广播数据。

在主模式下，数据一旦写入 SSPBUF 寄存器就开始发送或接收。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 引脚上的信号进行连续移位输入。每收到一个字节，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）。

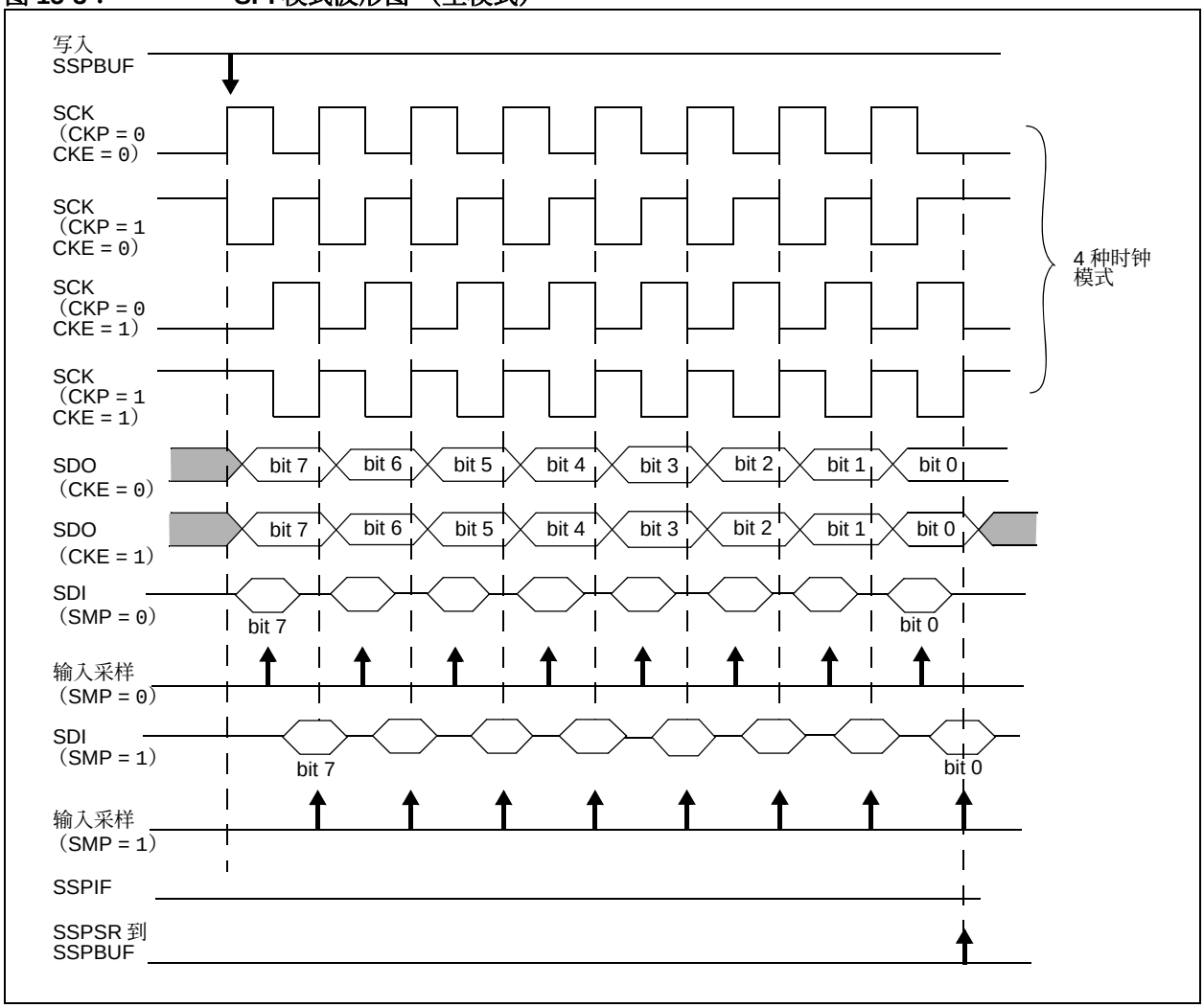
可通过对 SSPCON1 寄存器的 CKP 位进行适当的编程来选择时钟极性。图 15-3、图 15-5 和图 15-6 将给出 SPI 通信的波形图，其中 MSB 最先发送。在主模式下，SPI 时钟速率 （比特率）可由用户编程设定为以下几种之一：

- 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 16.00 Mbps （时钟频率为 64 MHz）。

图 15-3 给出了主模式的波形图。当 CKE 位置 1 时，SDO 数据在 SCK 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 的时间。

图 15-3： SPI 模式波形图 （主模式）



15.2.6 从模式

在从模式下，SCK 上出现外部时钟脉冲时，进行数据的发送和接收。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 SSPCON1 寄存器的 CKP 位决定。

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，器件从休眠状态中唤醒。

15.2.7 从选择同步

$\overline{SS}$ 引脚允许器件工作于同步从模式。要将 SPI 设置为从模式，需要使能 $\overline{SS}$ 引脚控制（SSPCON1<3:0> = 0100）。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，而是将其变成悬空输出状态。根据应用需要，可在 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（SSPCON<3:0> = 0100）时，如果 $\overline{SS}$ 引脚设置为 V_{DD}，SPI 模块将会复位。

2: 如果SPI工作在从模式下并且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。

图 15-4： 从同步波形图

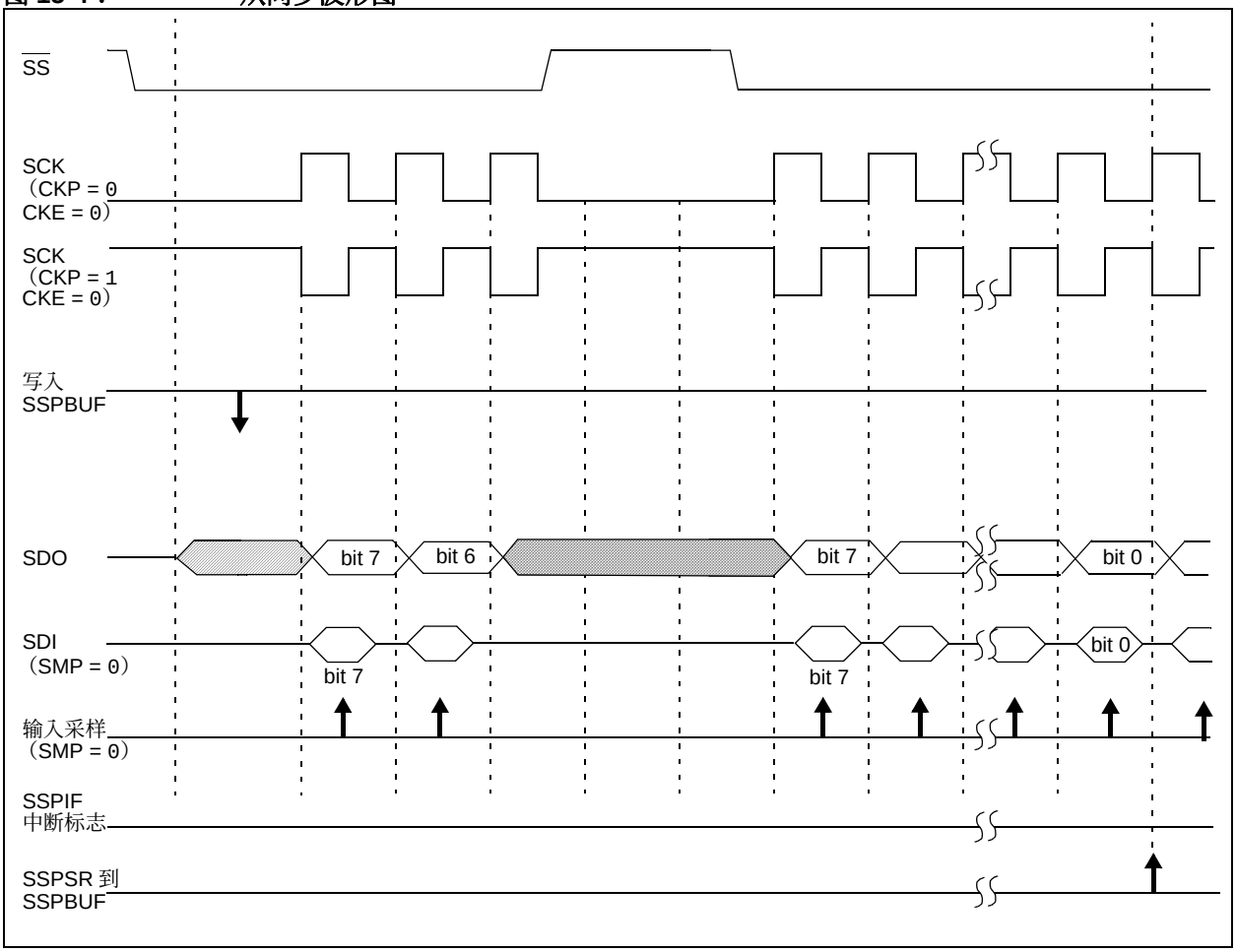


图 15-5：　　SPI 模式波形图 （从模式， CKE = 0）

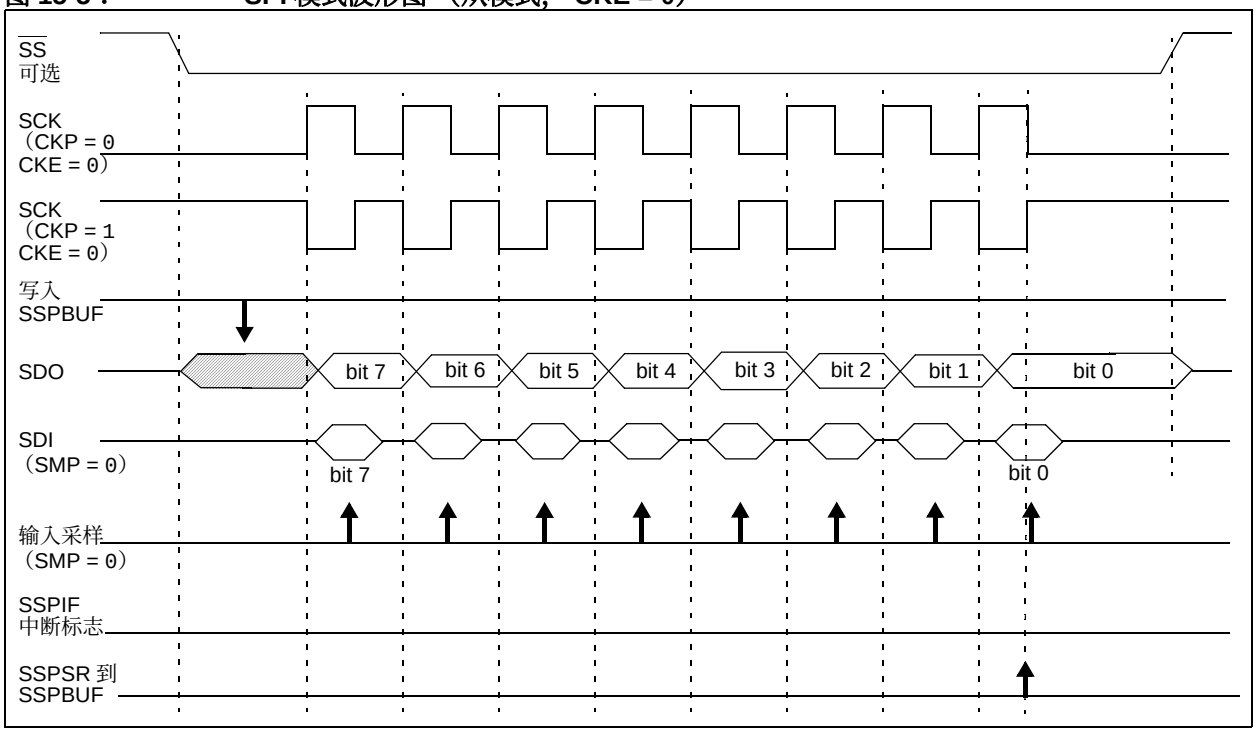
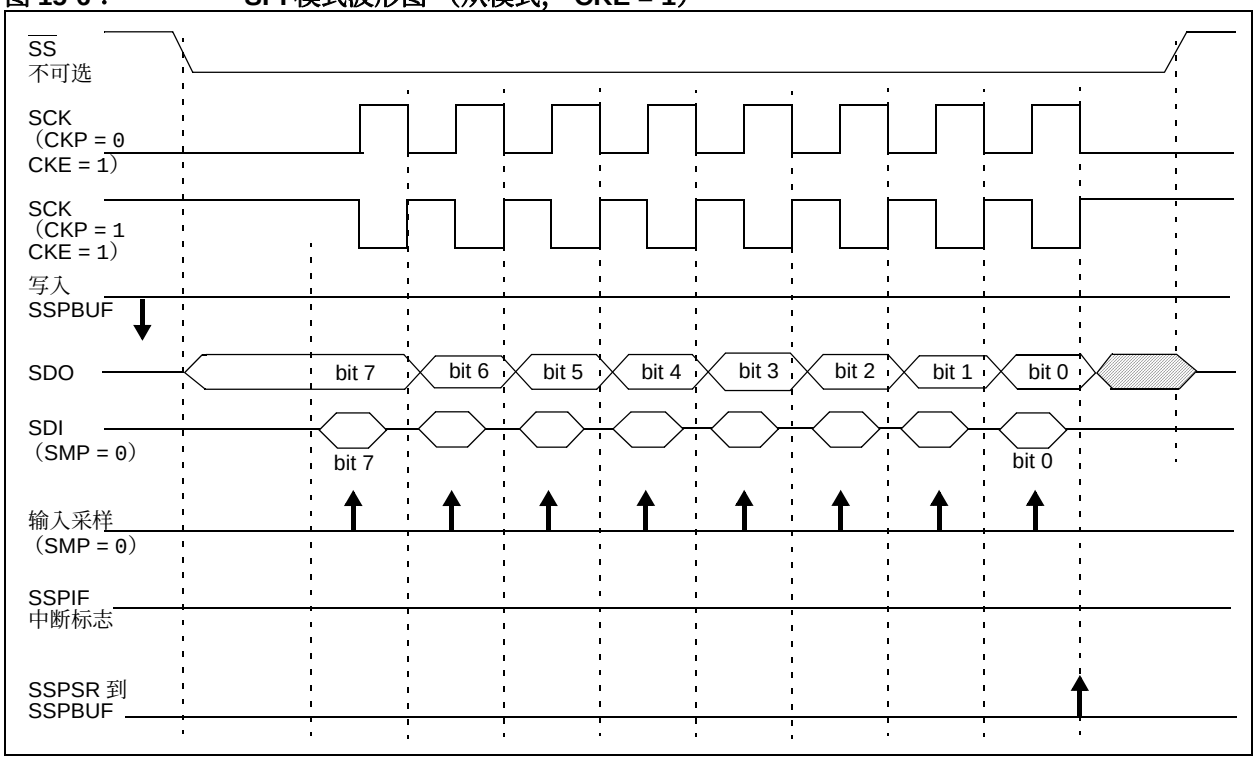


图 15-6：　　SPI 模式波形图 （从模式， CKE = 1）



15.2.8 在功耗管理模式下的操作

在 SPI 主模式下，模块时钟速度与全功耗模式下的不同；处于休眠模式时，所有时钟都停止。

在所有空闲模式下，需要为外设提供一个时钟。该时钟可能来自主时钟源、辅助时钟源（32.768 kHz 的 Timer1 振荡器）或 INTOSC 时钟源。更多信息，请参见**第 19.0 节 “ 功耗管理模式 ”**。

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，MSSP 中断将唤醒控制器：

- 从休眠模式唤醒 （在从模式下）
- 从空闲模式唤醒 （在从模式或主器件模式下）

如果不想从休眠或空闲模式退出，应该禁止MSSP中断。

在 SPI 主模式下，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 / 接收将保持此停滞状态。当器件返回到运行模式后，该模块将恢复发送和接收数据。

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。

15.2.9 复位的影响

复位会禁止 MSSP 模块并终止当前的数据传输。

15.2.10 总线模式兼容性

表 15-1 显示了标准 SPI 模式与 CKP 和 CKE 控制位状态之间的兼容性。

表 15-1： 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 15-2： 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	RABIE	TMR0IF	INT0IF	RABIF	285
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	288
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	288
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	288
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	288
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	288
SSPBUF	SSP 接收缓冲区 / 发送寄存器								286
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	286
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	286

图注： SPI 模式下的 MSSP 不使用阴影单元。

15.3 I²C 模式

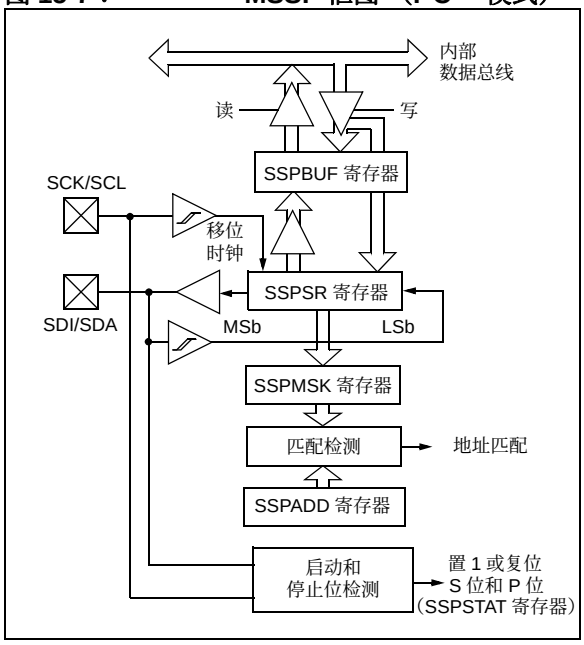
MSSP 模块工作在 I²C 模式时，可以实现所有的主和从功能 （包括广播呼叫支持），并且其硬件提供遇到启动位和停止位中断的功能来判断总线何时空闲（多主器件功能）。MSSP 模块实现了标准模式规范以及 7 位和 10 位寻址。

有两个引脚用于数据传输：

- 串行时钟 ——SCL
- 串行数据 ——SDA

注： 用户必须通过设置对应的TRIS位将这些引脚配置为输入引脚。

图 15-7： MSSP 框图 （I²C™ 模式）



15.3.1 寄存器

MSSP模块有7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）
- MSSP 地址掩码寄存器 （SSPMSK）

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。

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 被配置为工作在主模式下时，SSPADD 寄存器用于存放波特率发生器的重载值。当 MSSP 被配置为工作在 I²C 从模式下时，SSPADD 寄存器保存从器件的地址。通过使用 SSPMSK 寄存器检验地址寄存器的选定位， MSSP 可被配置为响应一个地址范围。

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 15-3： 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/$\overline{W}$(2, 3)	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：**压摆率控制位
在主或从模式下：
1 = 标准速度模式 （100 kHz 和 1 MHz）下禁止压摆率控制
0 = 高速模式 （400 kHz）下使能压摆率控制

bit 6 **CKE：**SMBus 选择位
在主或从模式下：
1 = 使能 SMBus 特定输入
0 = 禁止 SMBus 特定输入

bit 5 **D/$\overline{A}$：**数据 / $\overline{地址}$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 / $\overline{写}$信息位 （仅用于 I²C 模式） (2, 3)
在从模式下：
1 = 读
0 = 写
在主模式下：
1 = 正在进行发送
0 = 未进行发送

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 $\overline{ACK}$ 和停止位）
0 = SSPBUF 为空 （不包括 $\overline{ACK}$ 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 进行逻辑或运算的结果指示主模式是否有效。

寄存器 15-4： 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	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 寄存器仍在保存前一字节时，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：**同步串口使能位
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
当该位被使能时，必须将 SDA 和 SCL 引脚正确配置为输入引脚。

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：
在此模式下不使用。

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只用于 SPI 模式。

寄存器 15-5： SSPCON2：MSSP 控制寄存器 2 （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
GCEN	ACKSTAT	ACKDT(2)	ACKEN(1)	RCEN(1)	PEN(1)	RSEN(1)	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 （仅用于从模式）
1 = 当 SSPSR 接收到广播呼叫地址 0x00 或 00h 时产生中断
0 = 禁止广播呼叫地址

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） **(2)**
1 = 无应答
0 = 应答

bit 4 **ACKEN：**应答序列使能位 （仅用于主接收模式） **(1)**
1 = 在 SDA 和 SCL 引脚上发起应答序列，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 （仅用于主模式） **(1)**
1 = 使能 I²C 接收模式
0 = 接收空闲

bit 2 **PEN：**停止条件使能位 （仅用于主模式） **(1)**
1 = 在 SDA 和 SCL 引脚上发起停止条件。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 （仅用于主模式） **(1)**
1 = 在 SDA 和 SCL 引脚上发起重复启动条件。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 / 延长使能位 **(1)**
在主模式下：
1 = 在 SDA 和 SCL 引脚上发起启动条件。由硬件自动清零。
0 = 启动条件空闲
在从模式下：
1 = 为从发送和从接收 （已使能延长）使能时钟延长
0 = 禁止时钟延长

注 1： 对于 ACKEN、 RCEN、 PEN、 RSEN 和 SEN 位：如果 I²C 模块不处于空闲模式，可能这些位不会被置 1 （没有缓存），并且可能不会写入 SSPBUF （或禁止写 SSPBUF）。
2： 用户在接收结束时发起一个应答序列，同时发送该值。

15.3.2 工作原理

MSSP 模块功能通过将 SSPCON1 寄存器的 SSPEN 位置 1 使能。

SSPCON1 寄存器用于控制 I²C 工作模式。可通过设置 SSPCON1寄存器的模式选择位选择以下I²C模式之一：

- I²C 主模式，时钟 = (Fosc/(4*(SSPADD + 1))
- I²C 从模式 （7 位地址）
- I²C 从模式 （10 位地址）
- I²C 从模式 （7 位地址），允许启动位和停止位中断
- I²C 从模式 （10 位地址），允许启动位和停止位中断
- I²C 固件控制的主模式，从器件空闲

假设已通过将相应的 TRIS 位置 1，将 SCL 和 SDA 引脚编程为输入引脚；那么在 SSPEN 位置 1 时选择任何 I²C 模式，将强制上述引脚漏极开路。

注：	要确保此模块正常工作，必须为 SCL 和 SDA 引脚外接上拉电阻。

15.3.3 从模式

在从模式下，SCL 引脚和 SDA 引脚必须被配置为输入。必要时 MSSP 模块将使用输出数据改写输入状态 （从发送器）。

I²C 从模式硬件总是在地址匹配时产生中断。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 （$\overline{\text{ACK}}$）脉冲，并把当时 SSPSR 寄存器中接收到的值装入 SSPBUF 寄存器。

只要满足下列条件之一，MSSP 模块就不会产生此 $\overline{\text{ACK}}$ 脉冲：

- 在接收到数据前， SSPSTAT 寄存器的缓冲区满位 BF 被置 1。
- 在接收到数据前， SSPCON1 寄存器的溢出位 SSPOV 被置 1。

在上述情况下， SSPSR 寄存器的值不会装入 SSPBUF，但 PIR1 寄存器的 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。**第 27.0 节 “ 电气规范 ”** 中显示了 I²C 规范的高低电平时间和对 MSSP 模块的具体要求。

15.3.3.1 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，会发生下列事件：

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 $\overline{\text{ACK}}$ 脉冲。
4. 在第 9 个 SCL 脉冲的下降沿， PIR1 寄存器的 MSSP 中断标志位 SSPIF 被置 1 （如果允许中断，则产生中断）。

在 10 位地址模式下，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。 SSPSTAT 寄存器的 R/W 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 ” 是该地址的两个最高有效位。 10 位地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、SSPSTAT 寄存器的 BF 和 UA 位被置 1）。
2. 读 SSPBUF 寄存器 （BF 位清零）并将标志位 SSPIF 清零。
3. 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）。
4. 接收地址的第二个 （低）字节 （SSPIF、 BF 和 UA 位置 1）。如果地址匹配，则 SCL 将保留直到下一步。否则 SCL 线不会保留。
5. 读 SSPBUF 寄存器 （BF 位清零）并将标志位 SSPIF 清零。
6. 使用地址的第一个 （高）字节更新 SSPADD 寄存器。（这将清零 UA 位并释放保留的 SCL 线。）
7. 接收重复启动条件。
8. 接收地址的第一个 （高）字节并将 R/$\overline{\text{W}}$ 位置 1 （SSPIF、 BF 和 R/$\overline{\text{W}}$ 位置 1）。
9. 读 SSPBUF 寄存器 （BF 位清零）并将标志位 SSPIF 清零。
10. 将从器件要发送的字节装入 SSPBUF 并将 BF 位置 1。
11. 将 CKP 位置 1 以释放 SCL。

15.3.3.2 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 SSPSTAT 寄存器的 BF 位被置 1，或 SSPCON1 寄存器的 SSPOV 位被置 1。

每传输一个数据字节都会产生一个 MSSP 中断。 PIR1 寄存器的标志位 SSPIF 必须用软件清零。

当 SSPCON2 寄存器的 SEN 位被置 1 时，SCL 将在每次数据传输后保持低电平 （时钟延长） 。必须通过将 SSPCON1 寄存器的 CKP 位置 1 来释放时钟。更多详细信息，请参见**第 15.3.4 节 “ 时钟延长 ”**。

15.3.3.3 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 位上发送，同时不管 SEN 的值如何，SCK/SCL 引脚保持低电平（更多详细信息，请参见**第 15.3.4 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 SSPCON1 寄存器的 CKP 位置 1 来释放 SCK/SCL 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 15-9）。

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 线为高电平 （无 $\overline{ACK}$ 应答信号），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（复位 SSPSTAT 寄存器），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 来释放 SCK/SCL 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 15-8：　　I^2C™ 从模式接收时序 （SEN = 0， 7 位地址）

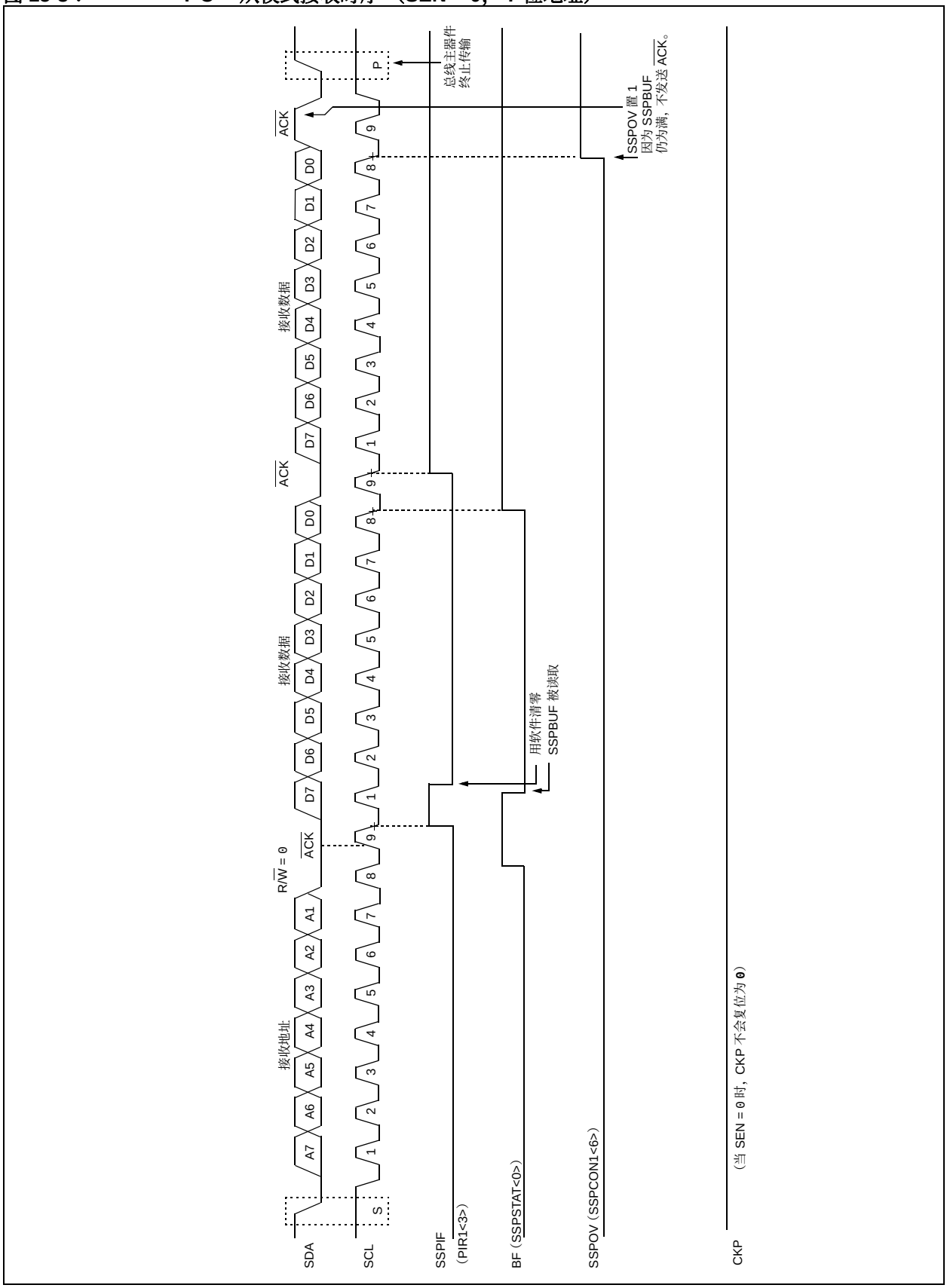


图 15-9：　　I^2C™ 从模式发送时序 （7 位地址）

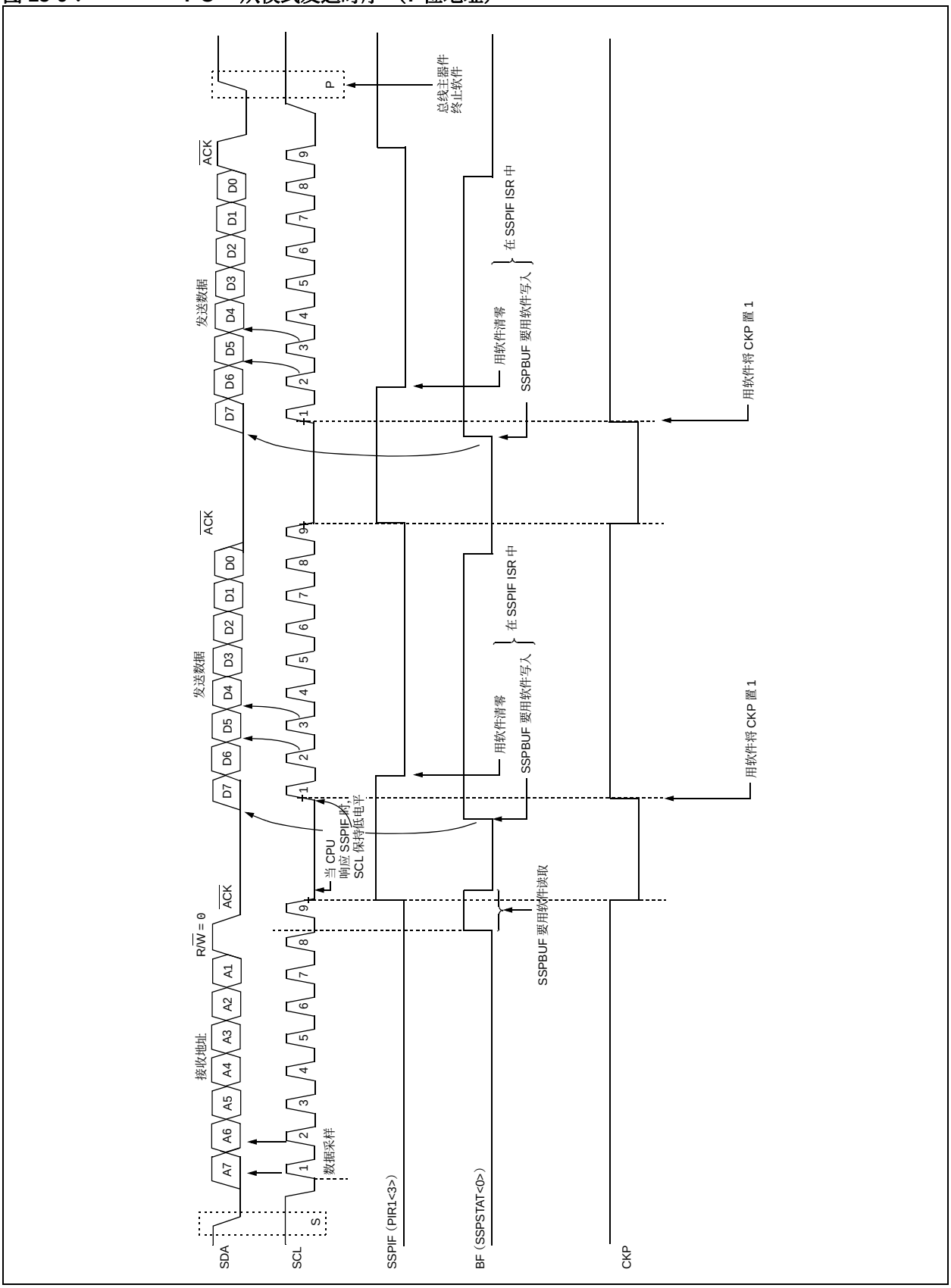


图 15-10： I²C™ 从模式接收时序 （SEN = 0， 10 位地址）

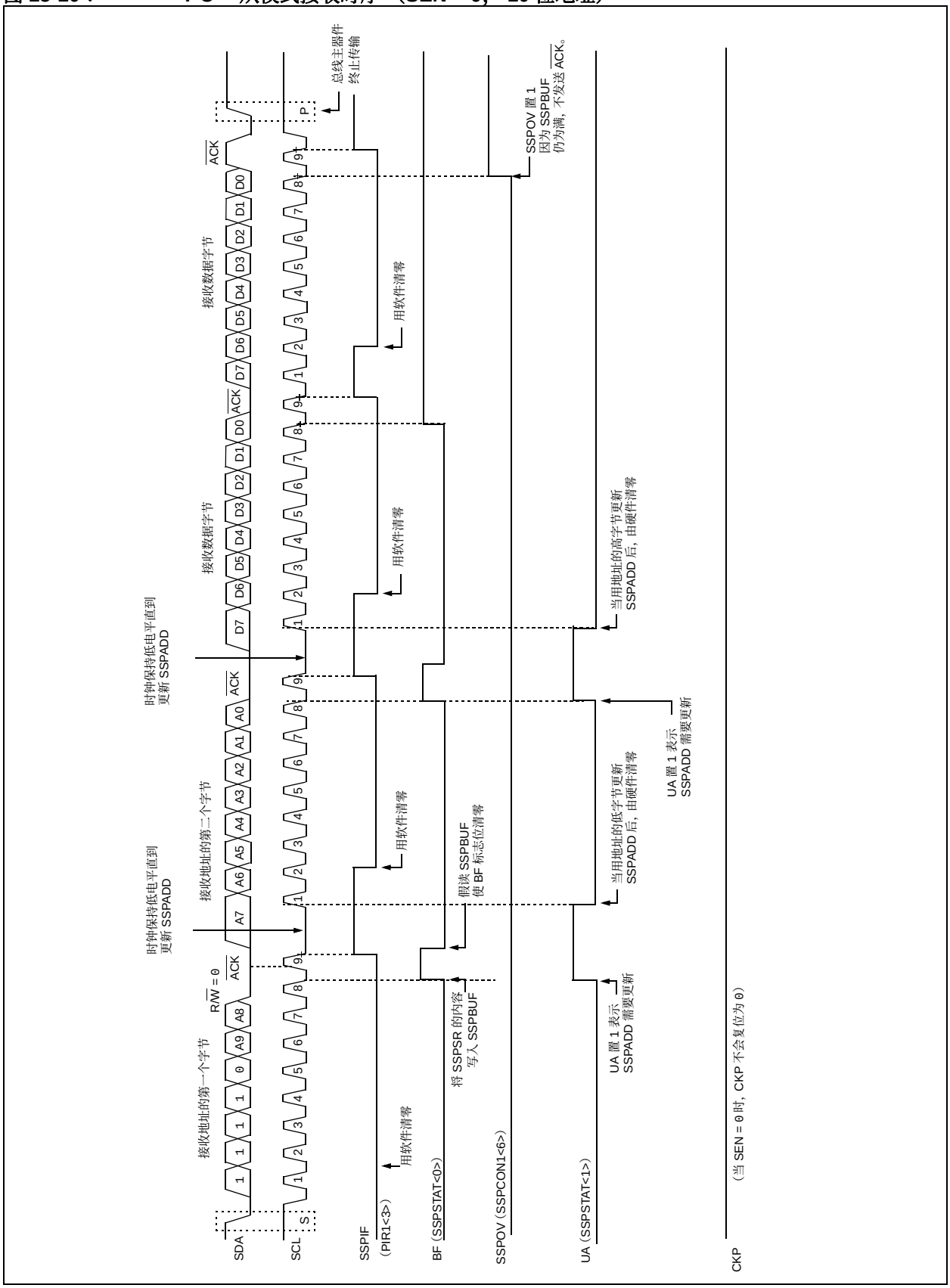
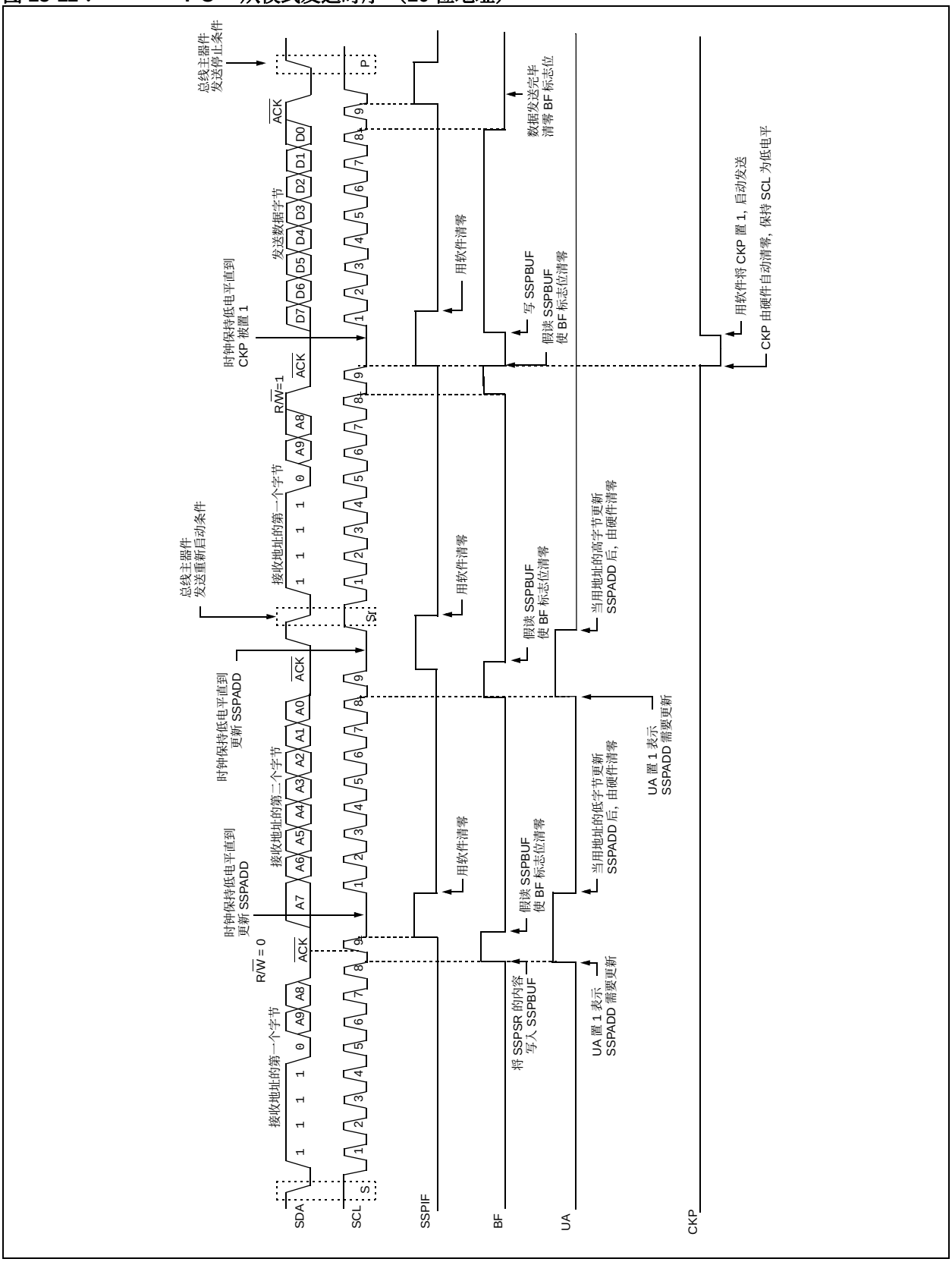


图 15-11： I^2C™ 从模式发送时序 （10 位地址）



15.3.3.4 SSP 掩码寄存器

SSP 掩码（SSPMSK）寄存器在 I²C 从模式下可用，用于在地址比较操作期间屏蔽 SSPSR 寄存器中保存的值。SSPMSK 寄存器中的零（0）位会忽略 SSPSR 寄存器中的相应位。

任何复位后，该寄存器都会复位到全 1 状态，因此，在写入掩码值之前对标准 SSP 操作没有影响。

在将 SSPM<3:0> 位置 1 前必须初始化该寄存器，以便选择 I²C 从模式 （7 位或 10 位地址）。

SSP 掩码寄存器在以下期间保持活动状态：

- 7 位地址模式：A<7:1> 的地址比较。
- 10 位地址模式：仅针对 A<7:0> 的地址比较。在接收地址的第一个 （高）字节期间， SSP 掩码没有影响。

寄存器 15-6： SSPMSK：SSP 掩码寄存器

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0(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 7-1 **MSK<7:1>：**掩码位
1 = 接收到的地址位 n 与 SSPADD<n> 相比较来检测 I²C 地址匹配
0 = 接收到的地址位 n 不用于检测 I²C 地址匹配

bit 0 **MSK<0>：**掩码位用于 I²C 从模式， 10 位地址 **(1)**
I²C 从模式， 10 位地址 （SSPM<3:0> = 0111）：
1 = 接收到的地址位 0 与 SSPADD<0> 相比较来检测 I²C 地址匹配
0 = 接收到的地址位 0 不用于检测 I²C 地址匹配

注 1：MSK0 位仅在 10 位从模式下使用。在所有其他模式下，该位不起作用。

寄存器 15-7： SSPADD：MSSP 地址和波特率寄存器 （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
ADD7	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0
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

主模式：

bit 7-0 **ADD<7:0>：**波特率时钟分频比位
SCL 引脚时钟周期 = ((ADD<7:0> + 1) *4)/FOSC

10 位从模式 —— 高位地址字节：

bit 7-3 **未使用：**不使用高位地址字节。该寄存器的位状态为无关位。主器件发送的位格式由 I²C 规范确定，必须等于 11110。但是，那些位通过硬件进行比较，并且不受该寄存器中的值影响。

bit 2-1 **ADD<9:8>：**10 位地址的高 2 位

bit 0 **未使用：**在此模式下不使用。位状态为无关位。

10 位从模式 —— 低位地址字节：

bit 7-0 **ADD<7:0>：**10 位地址的低 8 位

7 位从模式：

bit 7-1 **ADD<6:0>：**7 位地址

bit 0 **未使用：**在此模式下不使用。位状态为无关位。

15.3.4 时钟延长

7 位和 10 位从模式均能在发送序列期间自动实现时钟延长。

SSPCON2 寄存器的 SEN 位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。

15.3.4.1 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 $\overline{ACK}$ 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，必须在用户的中断服务程序中将 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数据传输序列之前，有时间处理中断服务程序并读取 SSPBUF 的内容。这将防止发生缓冲区溢出（见图 15-13）。

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，用户要注意在中断服务程序中清零 BF 位。

15.3.4.2 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，其 R/$\overline{W}$ 位是清零的。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会发生时钟延长。

15.3.4.3 7 位从发送模式的时钟延长

7 位从发送模式通过在第 9 个时钟的下降沿出现后清零 CKP 位，以实现时钟延长。上述情形与 SEN 位的状态无关。

用户的中断服务程序必须先将 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数据传输序列之前，将有时间处理中断服务程序并装入 SSPBUF 的内容 （见图 15-9）。

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 位置 1，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。

15.3.4.4 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/$\overline{W}$ 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，通过硬件清零 CKP 自动时钟延长，正如 7 位从发送模式一样 （见图 15-11）。

15.3.4.5 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 的最短高电平时间要求（见图 15-12）。

图 15-12：　　时钟同步时序

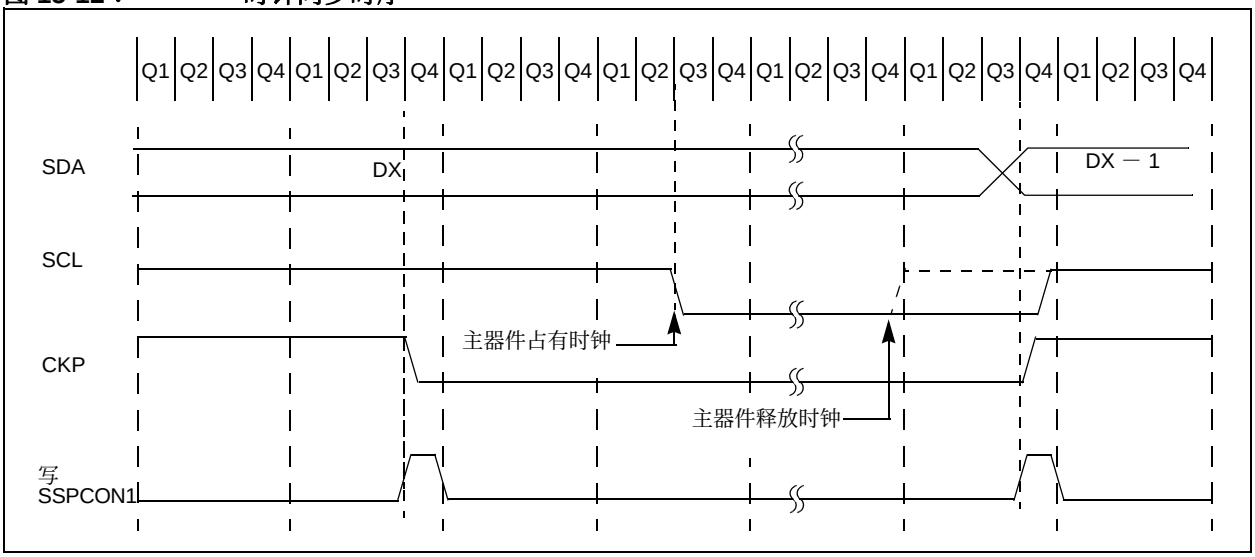


图 15-13： I^2C™ 从模式接收时序 （SEN = 1， 7 位地址）

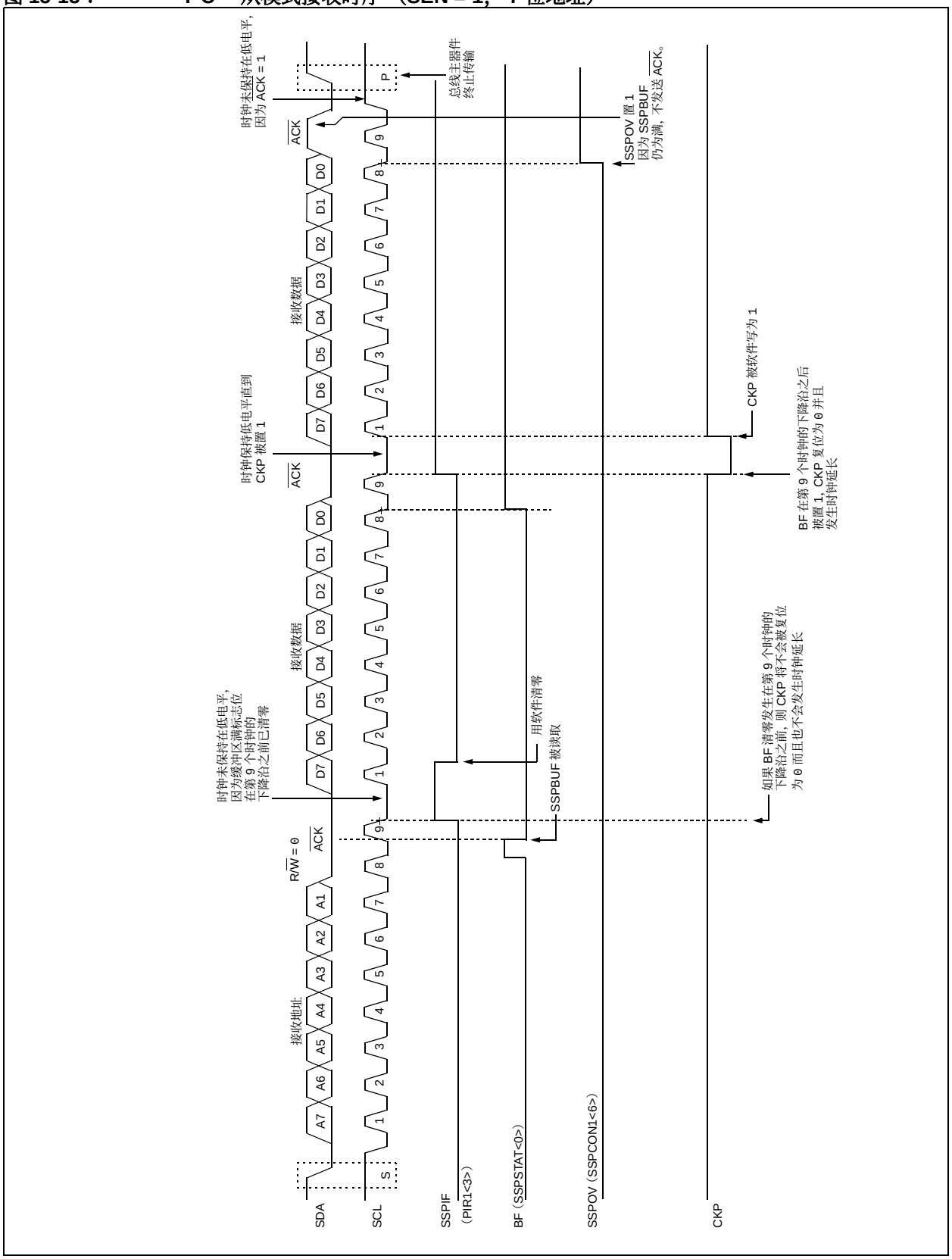
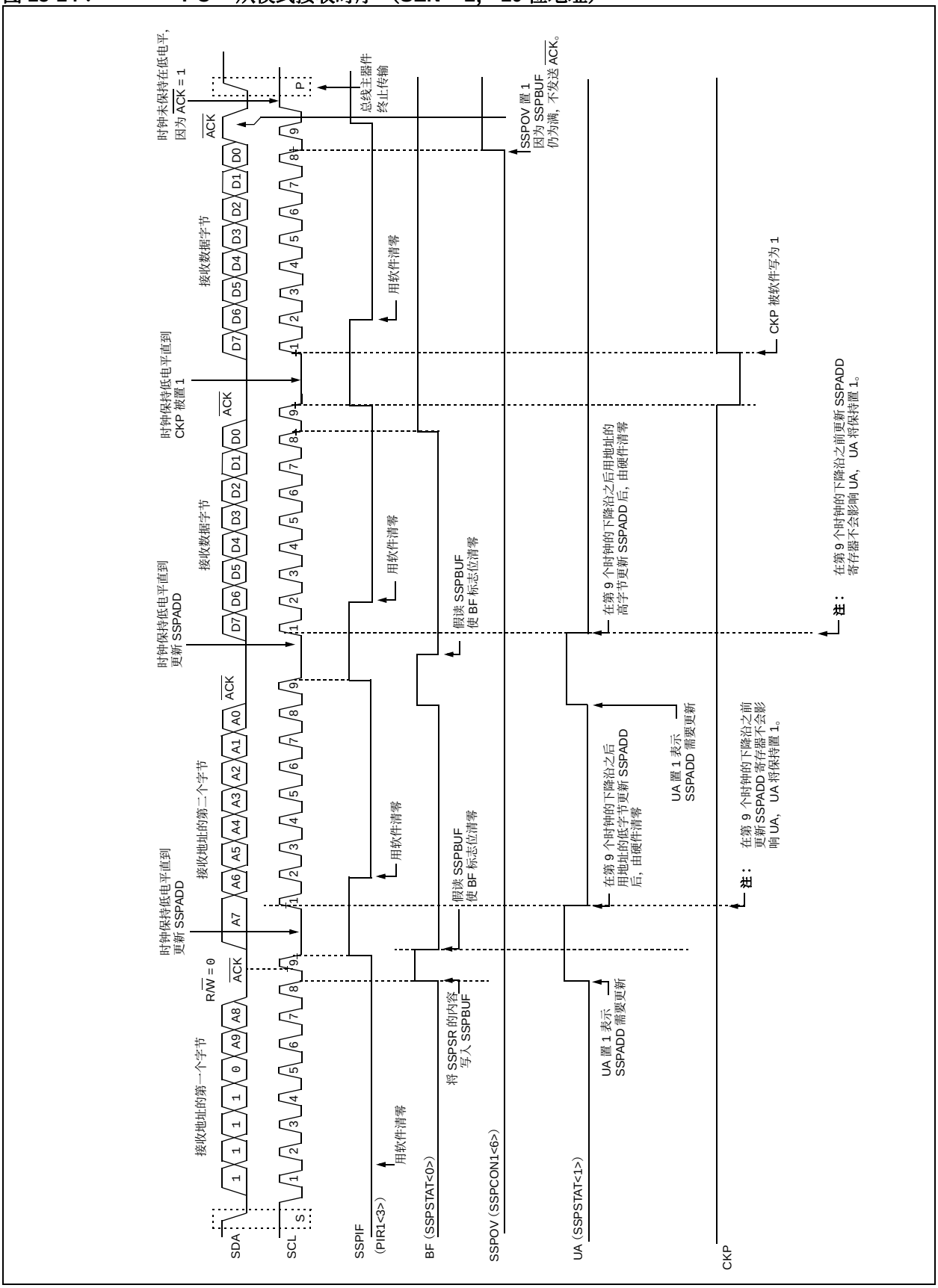


图 15-14： I²C™ 从模式接收时序 （SEN = 1， 10 位地址）



15.3.5 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/W = 0。

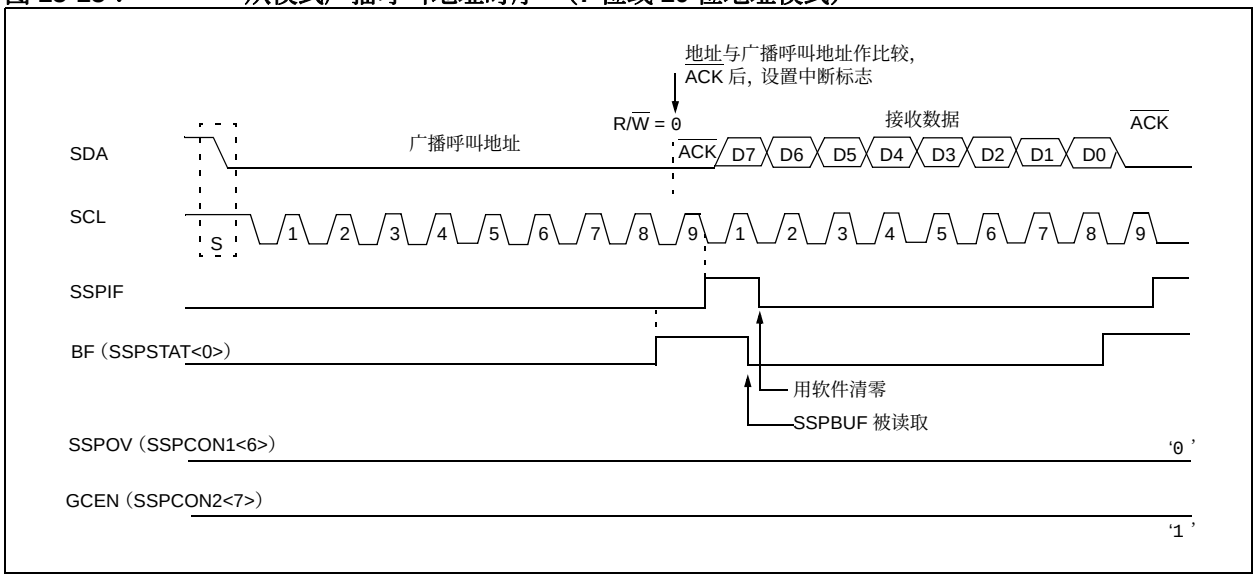
当 SSPCON2 的 GCEN 位置 1 时，既可识别广播呼叫地址。检测到启动位后， 8 位数据会被移入 SSPSR，同时将该地址与 SSPADD 进行比较。它还会与广播呼叫地址进行比较并用硬件设定。

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 （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 SSPSTAT 寄存器的 UA 位置 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 位地址模式，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 15-15）。

图 15-15： 从模式广播呼叫地址时序 （7 位或 10 位地址模式）



15.3.6 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；否则， P 位和 S 位都清零，总线处于空闲状态。

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。

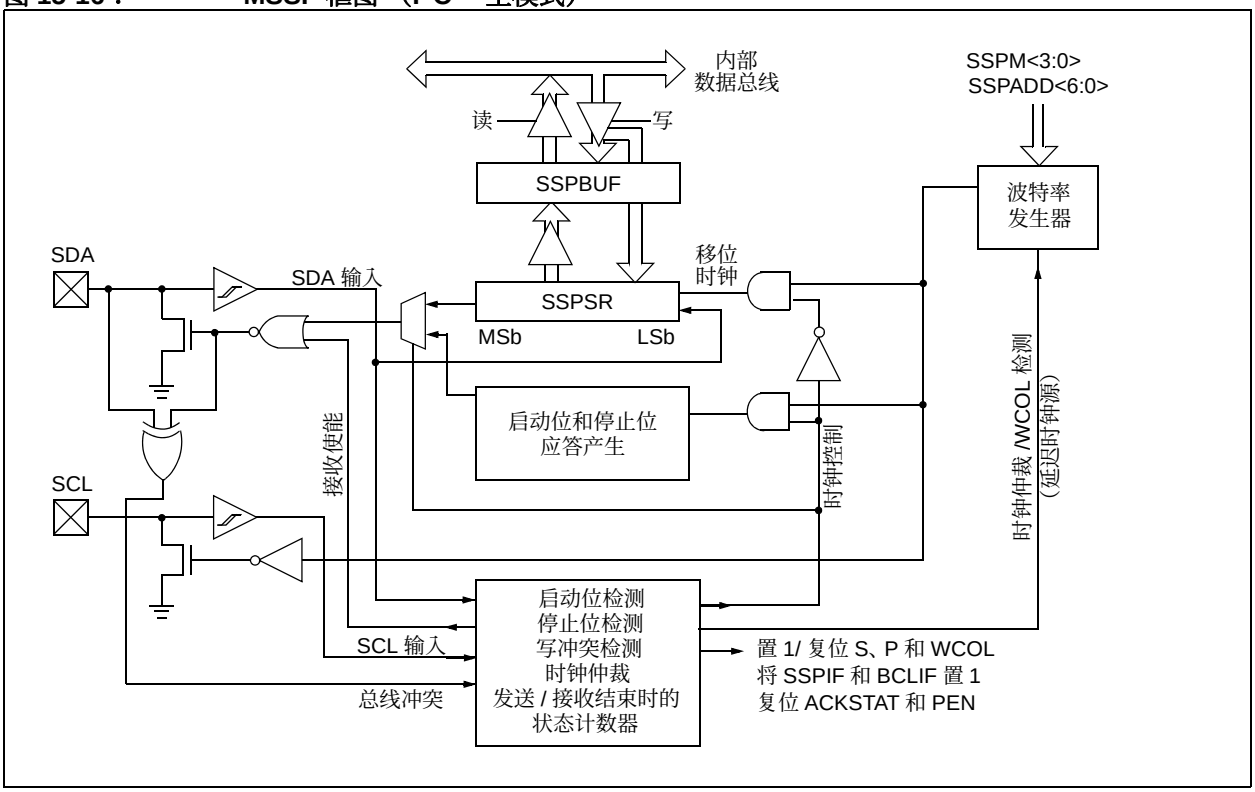
1. 在 SDA 和 SCL 上发出一个启动条件。
2. 在 SDA 和 SCL 上发出一个重复启动条件。
3. 写入 SSPBUF 寄存器，启动数据 / 地址的发送。
4. 配置 I^2C 端口用于接收数据。
5. 在接收数据字节末尾产生应答信号。
6. 在 SDA 和 SCL 上产生停止条件。

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。

下列事件会使 SSP 中断标志位 SSPIF 置 1 （如果允许 SSP 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 15-16： MSSP 框图 （I^2C™ 主模式）



15.3.6.1 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 总线一直保持不被释放的状态。

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代表发送的开始和结束。

波特率发生器用于设置 SCL 上的时钟频率输出。更多详细信息，请参见**第 15.3.7 节 “ 波特率 ”**。

下面是一个典型的发送时序：

1. 用户通过将 SSPCON2 寄存器的 SEN 位置 1，产生启动条件。
2. SSPIF 置 1。在进行下一步操作前，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 寄存器的 ACKSTAT 位。
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 寄存器的 ACKSTAT 位。
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 SSPCON2 寄存器的 PEN 位置 1，产生停止条件。
12. 一旦停止条件完成，将产生一个中断。

15.3.7 波特率

在 I²C 主模式下，波特率发生器（Baud Rate Generator，BRG）的重载值位于 SSPADD 寄存器中（图 15-17）。当发生对 SSPBUF 的写操作时，波特率发生器将自动开始计数。

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 15-3 给出了不同的指令周期下的时钟频率以及装入 SSPADD 的 BRG 值。

公式 15-1：

$$F_{SCL} = \frac{F_{OSC}}{(SSPADD + 1)(4)}$$

图 15-17： 波特率发生器框图

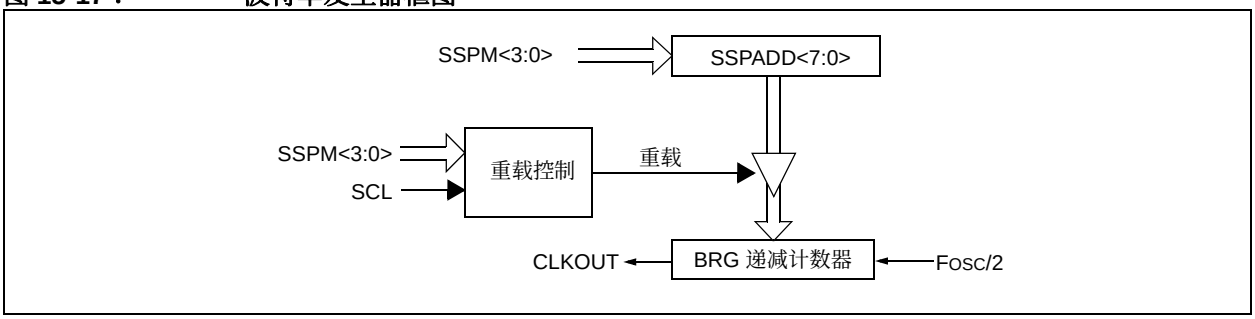


表 15-3： 使用 BRG 的 I²C™ 时钟频率

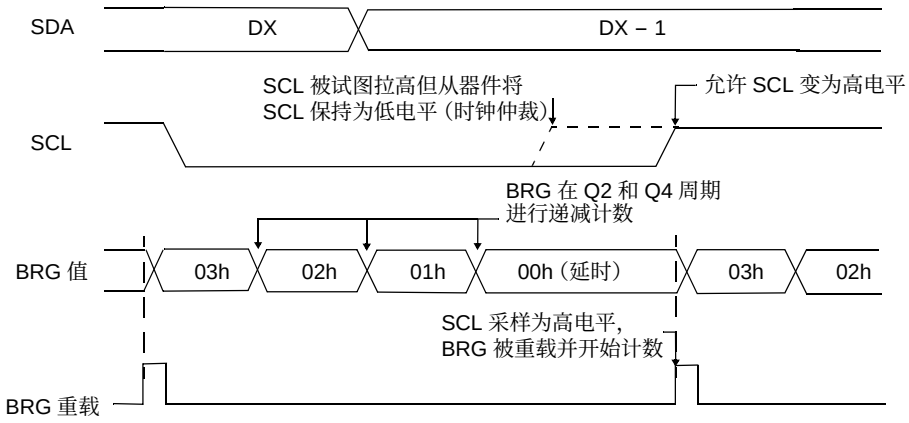
FOSC	FCY	BRG 值	FSCL（两次 BRG 计满返回）
48 MHz	12 MHz	0Bh	1 MHz(1)
48 MHz	12 MHz	1Dh	400 kHz
48 MHz	12 MHz	77h	100 kHz
40 MHz	10 MHz	18h	400 kHz(1)
40 MHz	10 MHz	1Fh	312.5 kHz
40 MHz	10 MHz	63h	100 kHz
16 MHz	4 MHz	09h	400 kHz(1)
16 MHz	4 MHz	0Ch	308 kHz
16 MHz	4 MHz	27h	100 kHz
4 MHz	1 MHz	02h	333 kHz(1)
4 MHz	1 MHz	09h	100 kHz
4 MHz	1 MHz	00h	1 MHz(1)

注 1： 虽然 I²C 接口各方面都不符合 400 kHz I²C 规范（该规范适用于大于 100 kHz 的频率），但在需要较高频率的应用场合可以慎重使用。

15.3.7.1 时钟仲裁

如果在任何接收、发送或重复启动 / 停止条件过程中，主器件释放了 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 周期内保持高电平 （图 15-18）。

图 15-18： 带有时钟仲裁的波特率发生器时序



15.3.8 I²C 主模式启动条件时序

要产生启动条件，用户应将 SSPCON2 寄存器的启动使能位 SEN 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 SSPSTAT1 寄存器的 S 位置 1。随后波特率发生器重新装入 SSPADD<7:0> 的内容并恢复计数。当波特率发生器再次超时（TBRG）时，SSPCON2 寄存器的 SEN 位将自动被硬件清零，波特率发生器暂停工作，SDA 线保持低电平，启动条件结束。

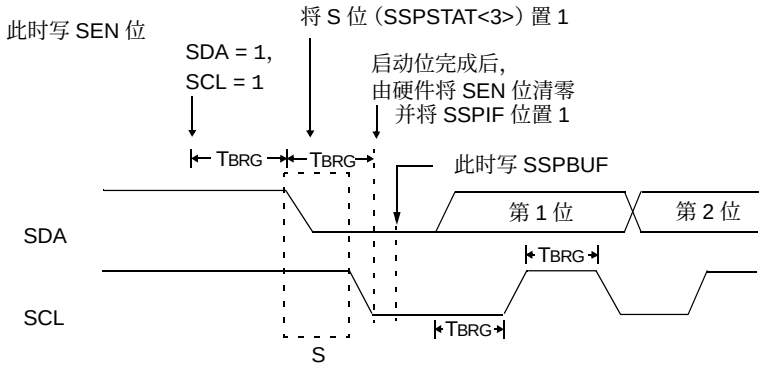
注： 如果在启动条件开始时，SDA 和 SCL 引脚已经采样为低电平，或者在启动条件期间，SCL 在 SDA 线被驱动为低电平之前已经采样为低电平，则会发生总线冲突。总线冲突中断标志位 BCLIF 置 1，启动条件中止，I²C 模块复位到空闲状态。

15.3.8.1 WCOL 状态标志

如果用户在启动序列过程中写 SSPBUF，则 WCOL 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 15-19： 第一个启动位时序



15.3.9 I²C 主模式重复启动条件时序

将 SSPCON2 寄存器的 RSEN 位编程为高电平，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（其引脚电平被拉高）。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 SSPCON2 寄存器的 RSEN 位将自动清零，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，SSPSTAT 寄存器的 S 位将被置 1。直到波特率发生器发生超时后， SSPIF 位才会置 1。

注 1： 有其他事件在进行时，编程设置对 RSEN 无效。

2： 在重复启动条件期间，下列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。

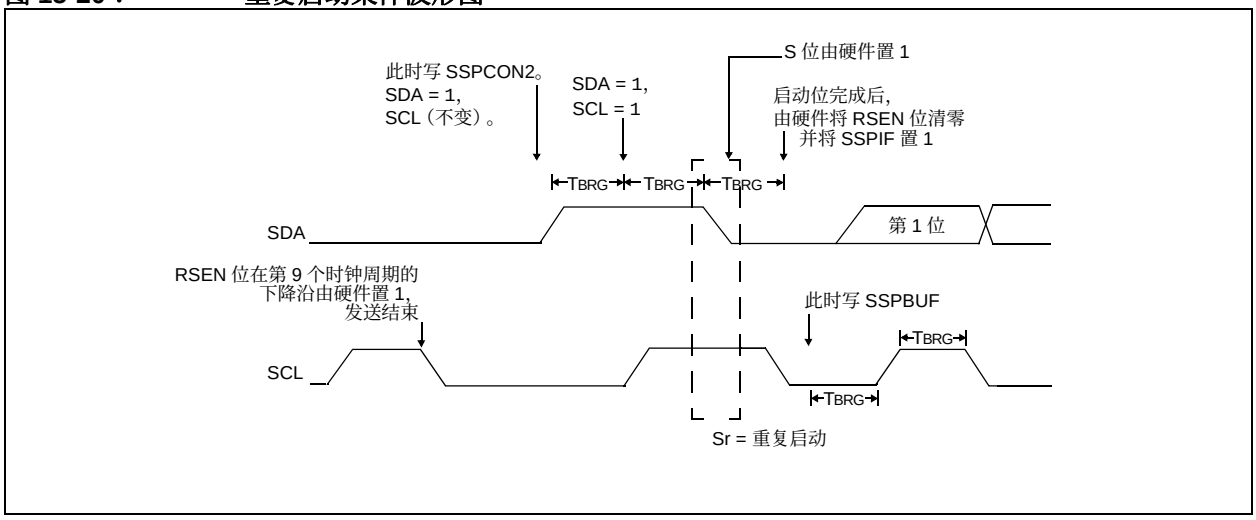
一旦 SSPIF 位被置 1，用户便可以在 7 位地址模式下将 7 位地址，或者在 10 位地址模式下将默认的第一个地址字节写入 SSPBUF。当发送完第一个 8 位数据并接收到一个 ACK 后，用户可以发送另外 8 位地址（10 位模式）或 8 位数据 （7 位模式）。

15.3.9.1 WCOL 状态标志

如果用户在重复启动序列过程中写SSPBUF，则WCOL 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 15-20： 重复启动条件波形图



15.3.10 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 位地址的另一半，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标志位 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 SP106），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 SP107）。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 15-21）。

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 SSPCON2 寄存器的 ACKSTAT 状态位。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。

15.3.10.1 BF 状态标志

在发送模式下，SSPSTAT 寄存器的 BF 位在 CPU 写 SSPBUF 时置 1，在所有 8 位数据移出后清零。

15.3.10.2 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 被置 1，同时缓冲区内容不变（写操作无效）。

在下一次发送前 WCOL 必须用软件清零。

15.3.10.3 ACKSTAT 状态标志

在发送模式下，当从器件发送应答响应（ACK = 0）时，SSPCON2 寄存器的 ACKSTAT 位清零；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，会发出一个应答。

15.3.11 I²C 主模式接收

通过编程 SSPCON2 寄存器的接收使能位 RCEN 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 SSPCON2 寄存器的应答序列使能位 ACKEN 置 1，用户可以在接收结束后发送应答位。

15.3.11.1 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。

15.3.11.2 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。

15.3.11.3 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 15-21： I^2C™ 主模式发送波形图 （7 位或 10 位地址）

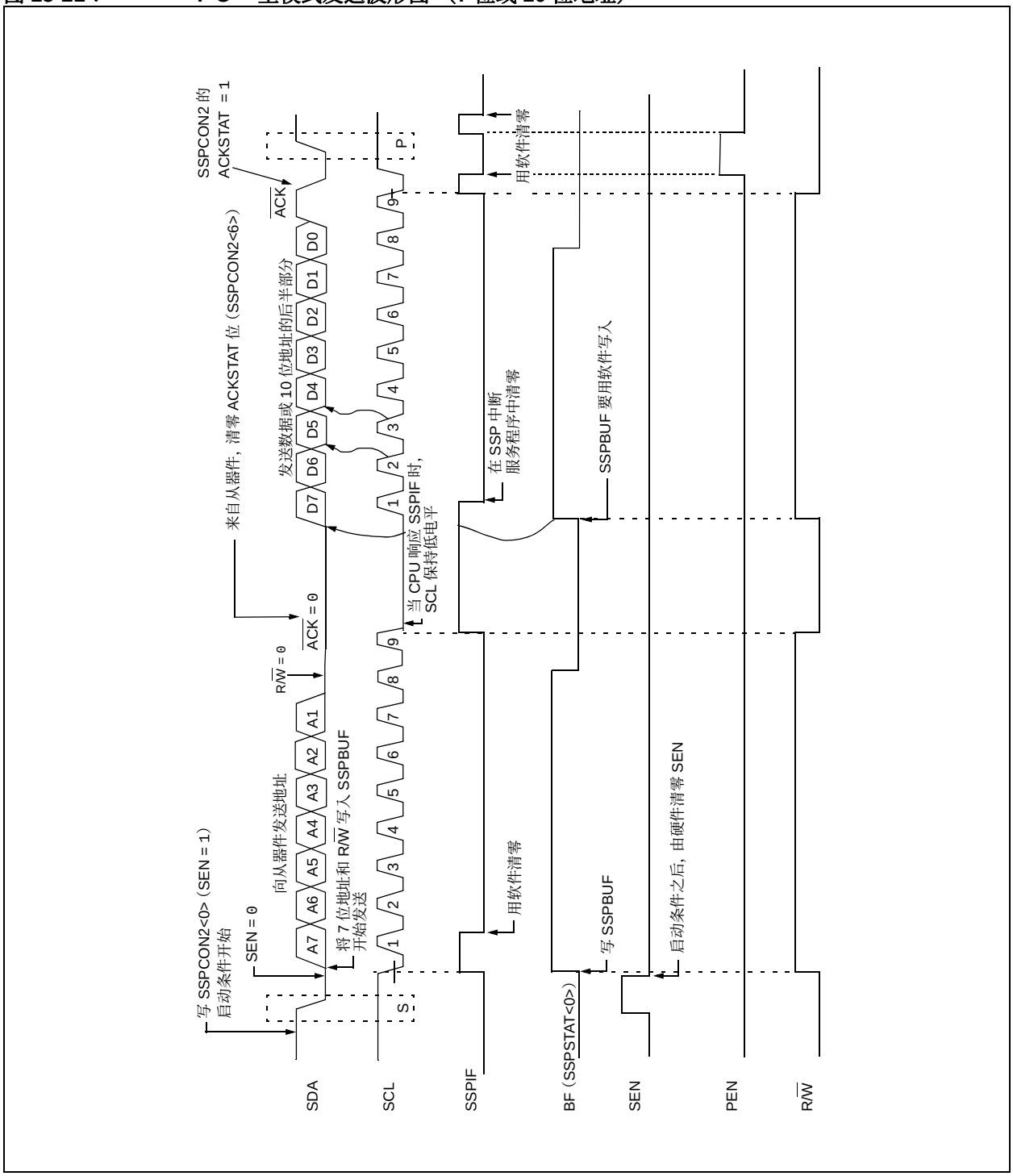
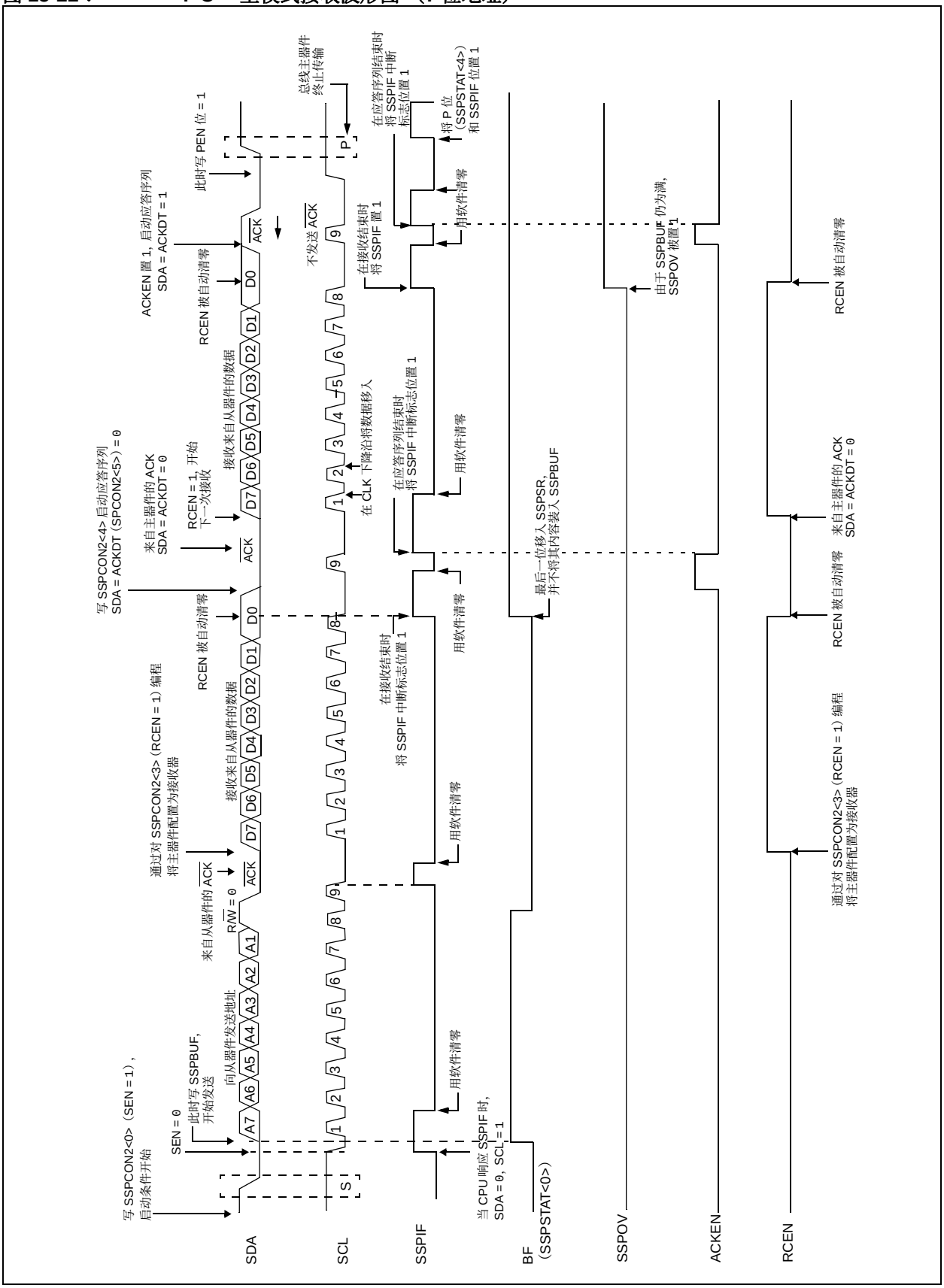


图 15-22： I²C™ 主模式接收波形图 （7 位地址）



15.3.12 应答序列时序

将 SSPCON2 寄存器的应答序列使能位 ACKEN 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 15-23）。

15.3.12.1 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）。

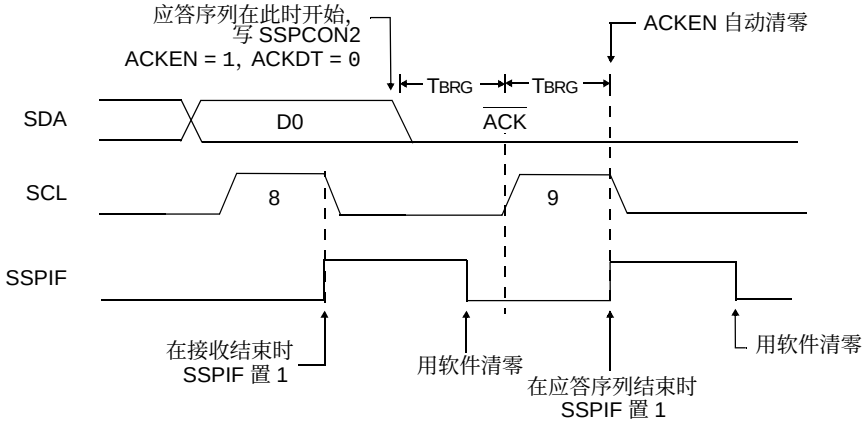
15.3.13 停止条件时序

如果将 SSPCON2 寄存器的停止序列使能位 PEN 置 1，则在接收 / 发送结束后， 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，SSPSTAT 寄存器的 P 位置 1。另一个 TBRG 之后， PEN 位被清零，同时 SSPIF 位被置 1 （图 15-24）。

15.3.13.1 WCOL 状态标志

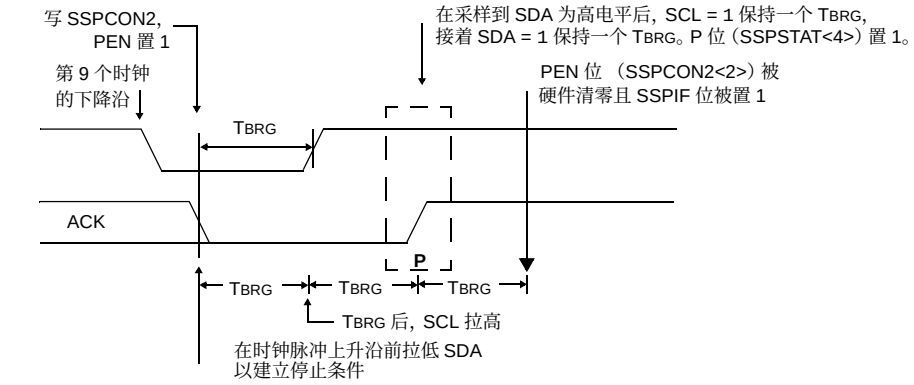
如果用户在停止序列进行过程中写SSPBUF，则WCOL位被置 1，同时缓冲区内容不变 （写操作无效）。

图 15-23： 应答序列波形图



注： TBRG = 一个波特率发生器周期。

图 15-24： 停止条件接收或发送模式



注： TBRG = 一个波特率发生器周期。

15.3.14 休眠模式下的操作

在休眠模式下，I²C 从模块能够接收地址或数据，并且在地址匹配或字节传输完成后，如果允许 MSSP 中断，将唤醒处理器。

15.3.15 复位的影响

复位会禁止 MSSP 模块并终止当前的数据传输。

15.3.16 多主器件模式

在多主器件模式下，在检测到启动和停止条件时将产生中断，这可以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 SSPSTAT 寄存器的 P 位置 1 时，可以取得 I²C 总线的控制权；否则，S 位和 P 位都清零，总线处于空闲状态。当总线忙时，一旦出现停止条件，将产生 SSP 中断。

在多主器件模式下，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

15.3.17 多主器件通信、总线冲突和总线仲裁

多主器件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 15-25）。

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。

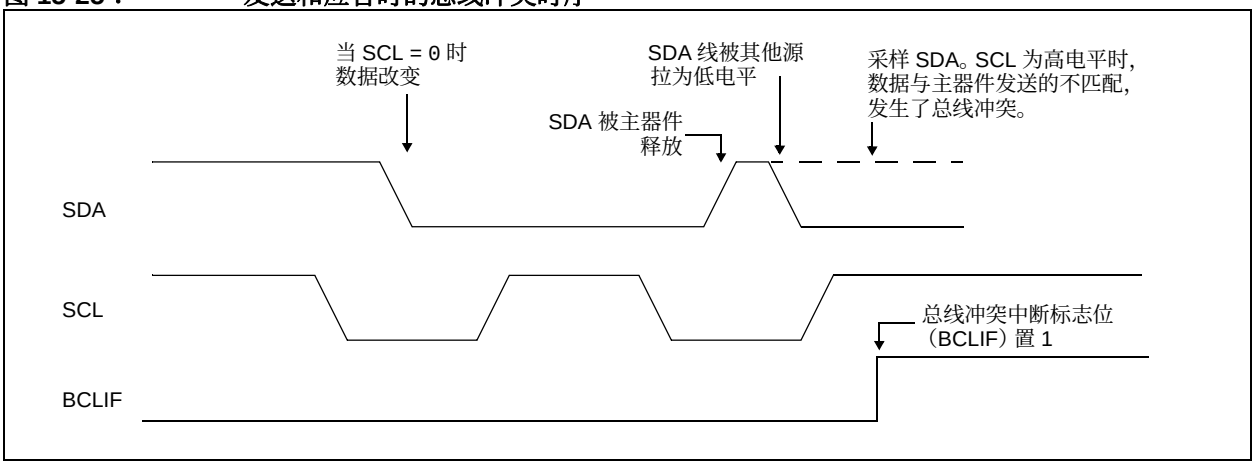
如果在启动、重复启动、停止或应答信号的执行过程中发生总线冲突，则这种状态被中止，SDA 和 SCL 线被拉高，SSPCON2 寄存器中的对应控制位清零。当执行完总线冲突中断服务程序后，如果 I²C 总线空闲，用户可通过发出启动条件恢复通信。

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。

在多主器件模式下，通过在检测到启动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；否则，S 位和 P 位都清零，总线处于空闲状态。

图 15-25： 发送和应答时的总线冲突时序



15.3.17.1 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 15-26）。

b) SDA 被拉低之前，SCL 采样为低电平（图 15-27）。

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，
- MSSP 模块复位为空闲状态 （图 15-26）。

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，波特率发生器装入值并递减计数。如果在 SDA 为高电平时， SCL 引脚采样为低电平，则发生总线冲突，因为这表示另一个主器件在启动条件期间试图发送一个数据 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 15-28）。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 15-26： 启动条件期间的总线冲突 （仅 SDA）

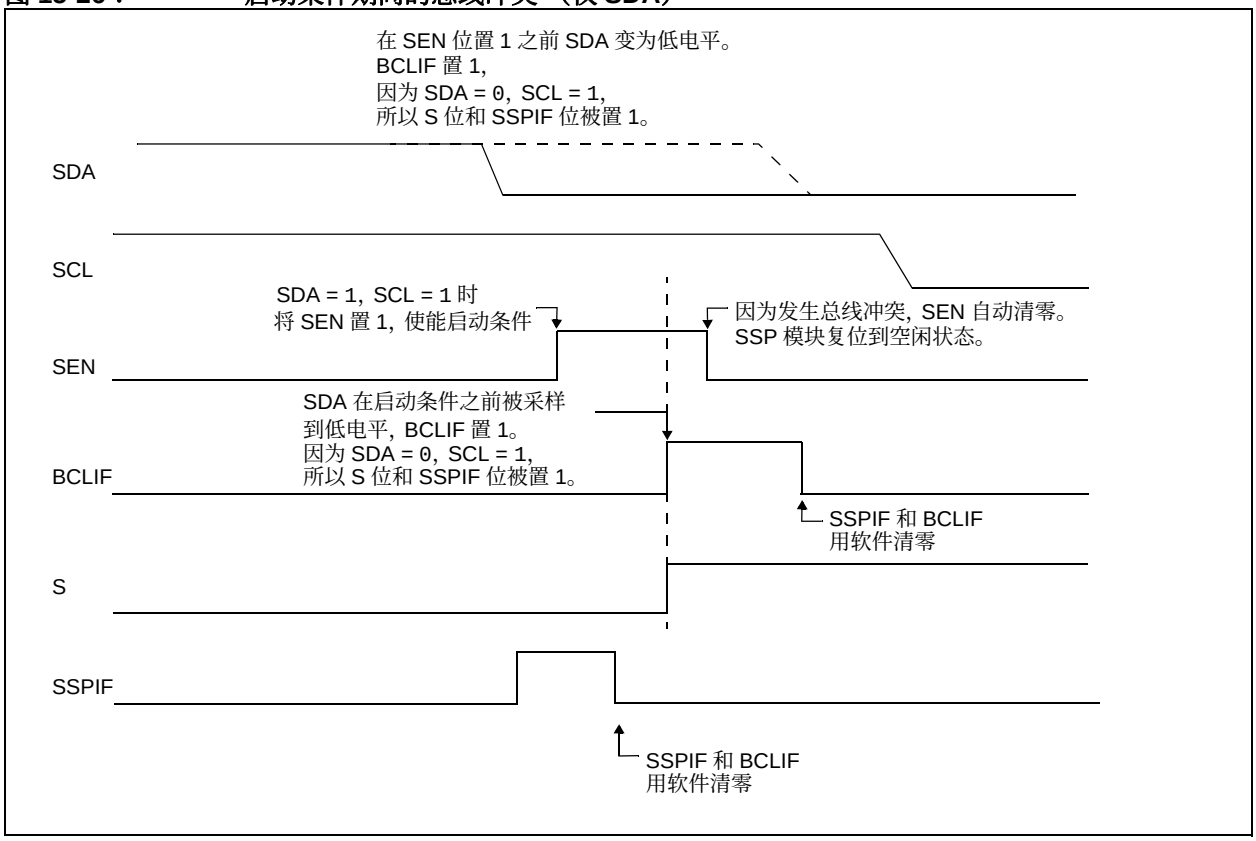


图 15-27：　　启动条件期间的总线冲突 （SCL = 0）

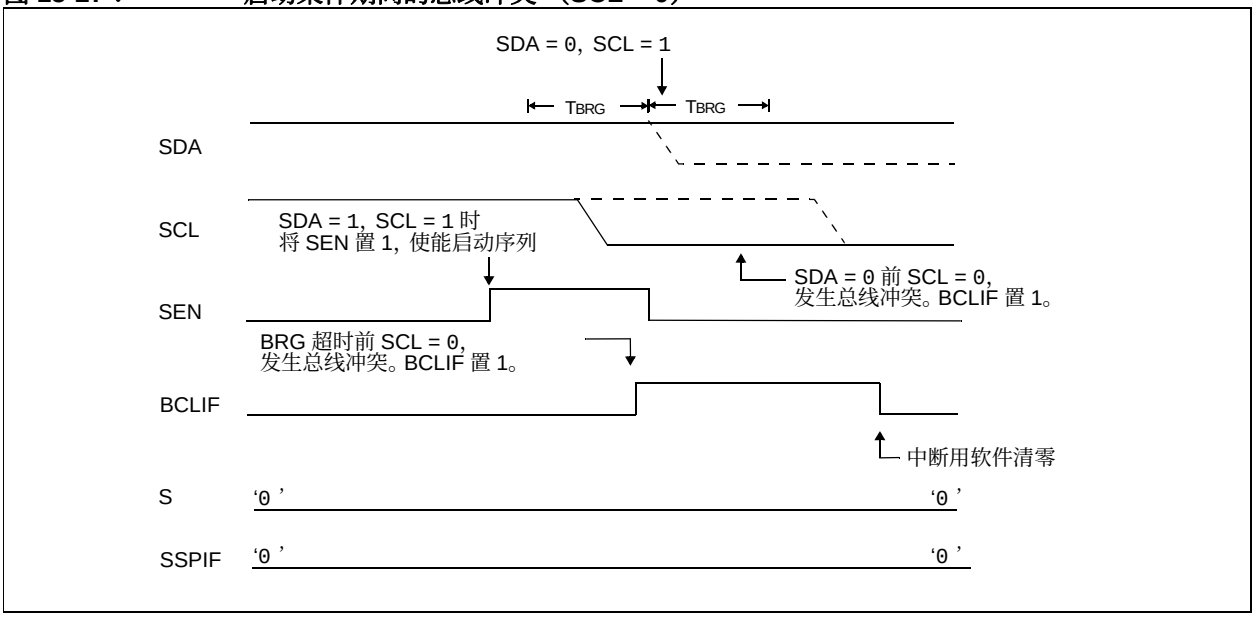
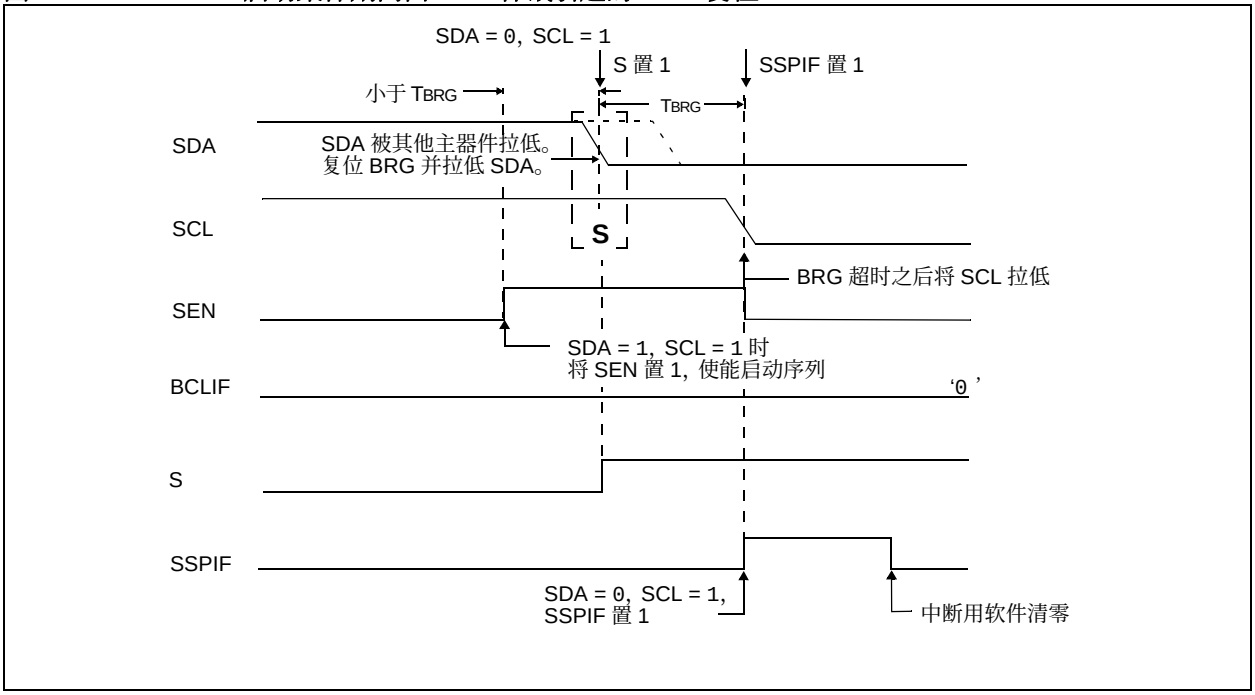


图 15-28：　　启动条件期间由 SDA 仲裁引起的 BRG 复位



15.3.17.2 重复启动条件期间的总线冲突

在下列情况中，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 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 15-29）。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，另一个主控器在重复启动条件期间正试图发送一个数据 1（见图 15-30）。

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 15-29： 重复启动条件期间的总线冲突 （情形 1）

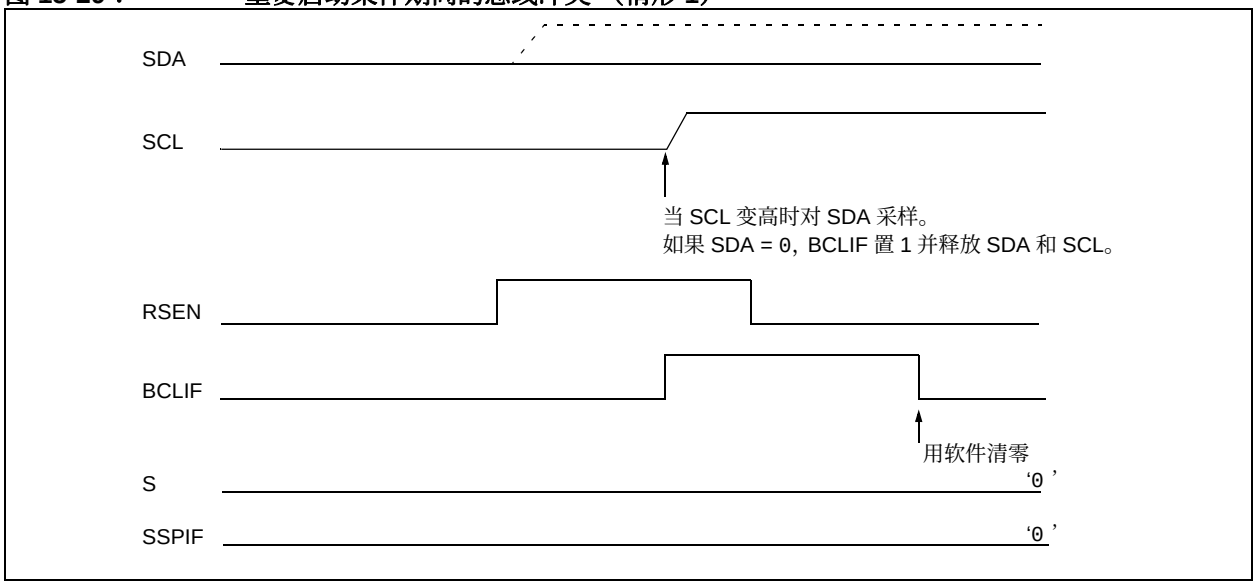
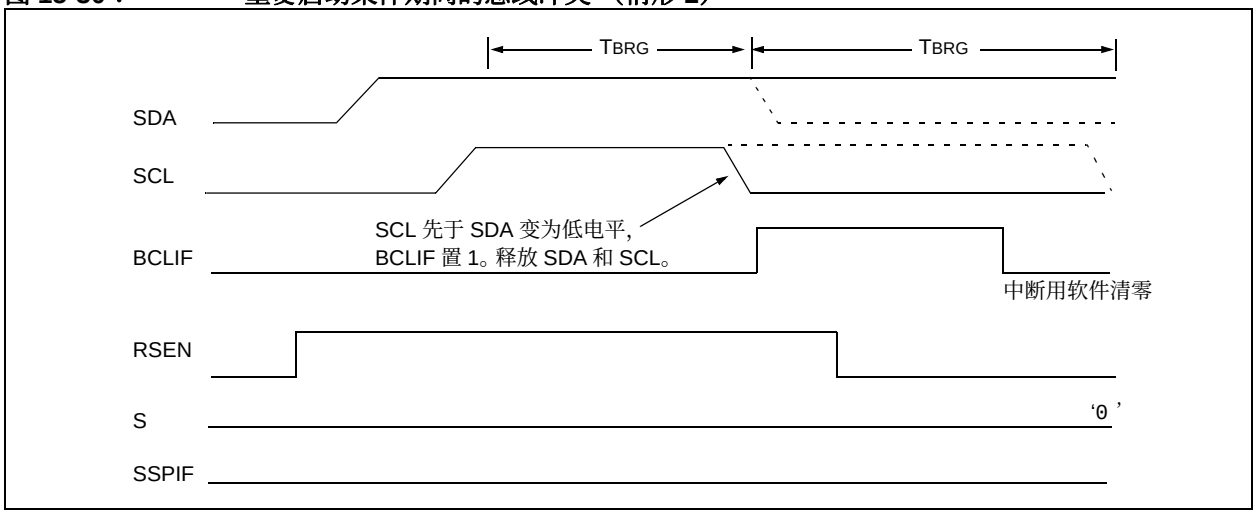


图 15-30： 重复启动条件期间的总线冲突 （情形 2）



15.3.17.3 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 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 15-31）。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 15-32）。

图 15-31： 停止条件期间的总线冲突 （情形 1）

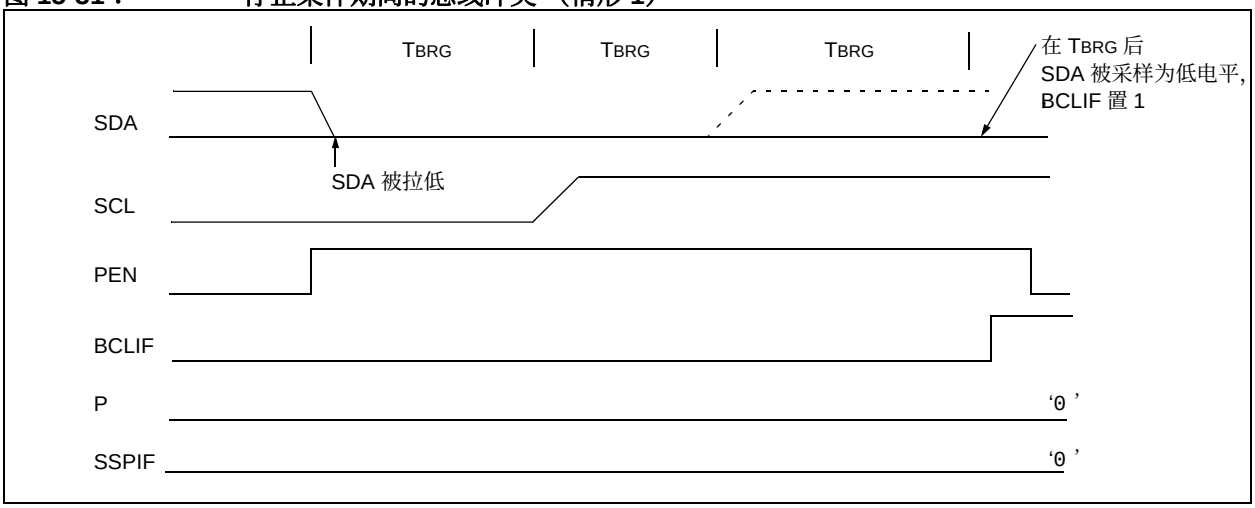


图 15-32： 停止条件期间的总线冲突 （情形 2）

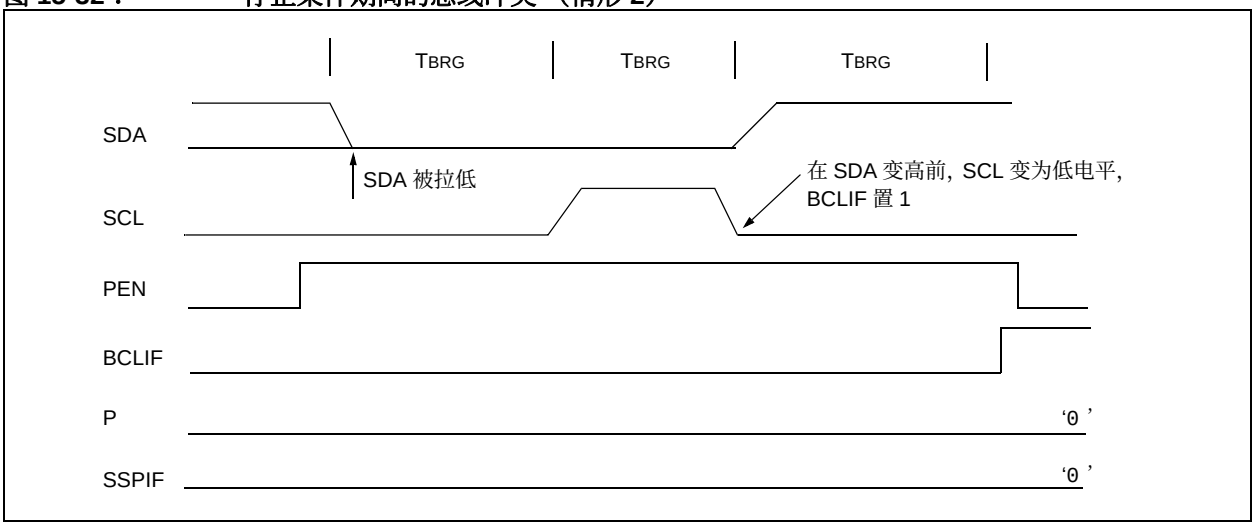


表 15-4：　　与 I²C™ 操作相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	288
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	288
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	288
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	—	288
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	288
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	288
SSPADD	I²C™ 从模式下的 SSP 地址寄存器。 I²C 主模式下的 SSP 波特率重载寄存器。								286
SSPBUF	SSP 接收缓冲区 / 发送寄存器								286
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	286
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	286
SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0	288
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	286
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	288

图注：　　— = 未实现，读为 0。 I²C™ 不使用阴影单元。

16.0 增强型通用同步 / 异步收发器（EUSART）

增强型通用同步 / 异步收发器（Enhanced Universal Synchronous Asynchronous Receiver Transmitter，EUSART）模块是一种串行 I/O 通信外设。它包含用来完成与器件程序执行无关的输入或输出串行数据传输所需的所有时钟发生器、移位寄存器和数据缓冲区等。EUSART 也是一种串行通信接口（Serial Communications Interface，SCI），可配置为全双工异步系统或半双工同步系统。全双工模式可用来与外设系统通信，如 CRT 终端和个人计算机。半双工同步模式用于与外设器件通信，如 A/D 或 D/A 集成电路、串行 EEPROM 或其他单片机。这些器件通常不具备用以产生波特率的内部时钟，并需要由主同步器件提供外部时钟信号。

EUSART 模块具备以下功能：

- 全双工异步收发
- 双字符输入缓冲区
- 单字符输出缓冲区
- 可编程 8 位或 9 位字符长度
- 9 位模式下的地址检测
- 输入缓冲区溢出错误检测
- 接收字符帧错误检测
- 半双工同步主模式
- 半双工同步从模式
- 可编程的时钟和数据极性

EUSART 模块还具备以下特性，使其成为局部互联网（Local Interconnect Network，LIN）总线系统的理想选择：

- 自动检测和波特率校准
- 接收间隔（Break）字符时唤醒
- 13 位间隔字符发送

EUSART 发送器和接收器的框图分别如图 16-1 和图 16-2 所示。

图 16-1：　　EUSART 发送框图

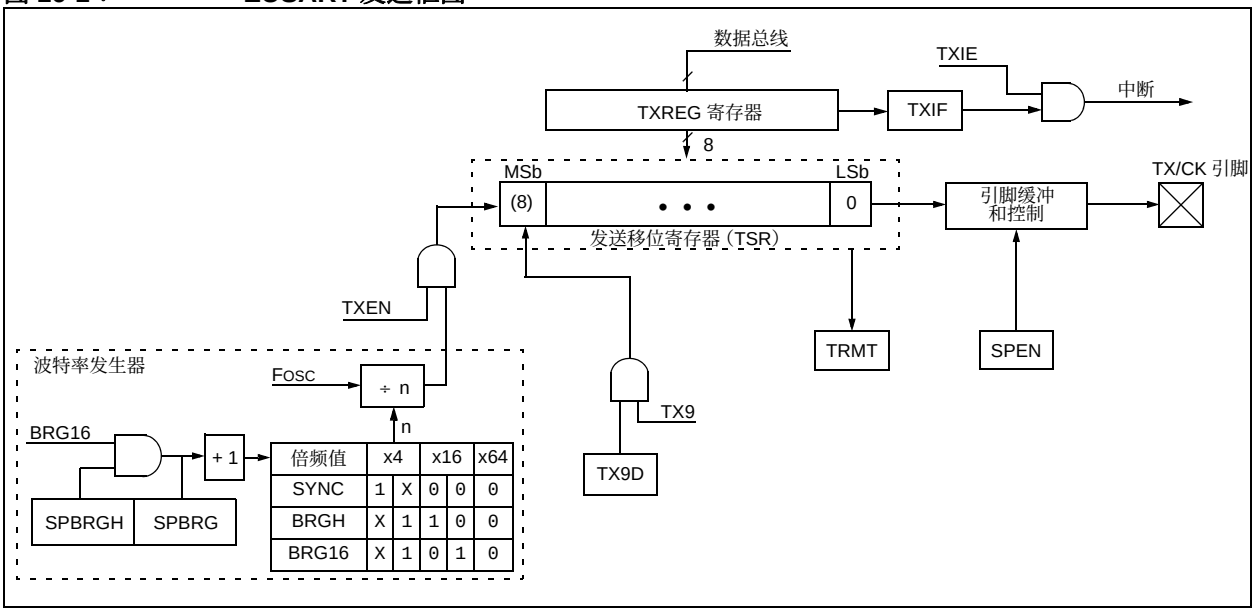
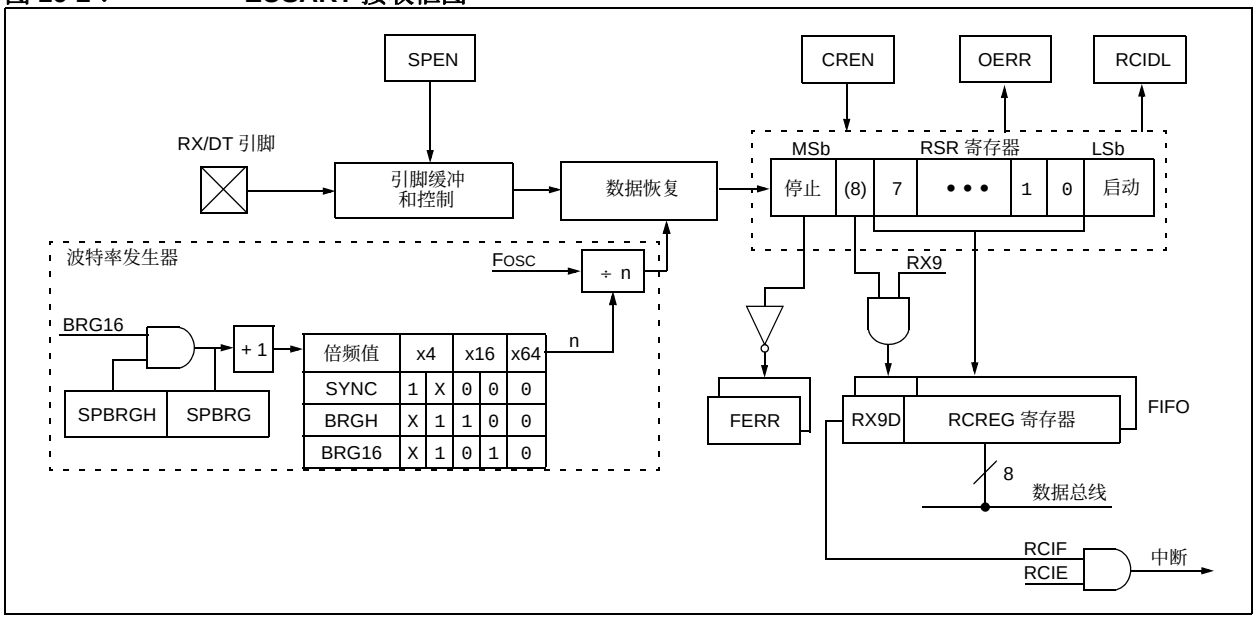


图 16-2：　　EUSART 接收框图



EUSART 模块的操作由以下 3 个寄存器控制：

- 发送状态和控制 （TXSTA）
- 接收状态和控制 （RCSTA）
- 波特率控制 （BAUDCTL）

这些寄存器的详细信息请分别参见寄存器 16-1、寄存器 16-2 和寄存器 16-3。

对于 EUSART 操作的所有模式，与 RX/DT 和 TX/CK 引脚对应的 TRIS 控制位应被置 1。EUSART 控制将根据需要自动将引脚从输入重新配置为输出。

16.1 EUSART 异步模式

EUSART 采用标准不归零（non-return-to-zero，NRZ）格式发送和接收数据。NRZ 实现为两种电平：VOH 标记状态（mark state）代表“1”数据位，而 VOL 空格状态（space state）代表“0”数据位。NRZ 指的是当具有相同值的连续数据位被发送时，输出电平始终保持不变，而不会在每个位之间回到中间电平。NRZ 发送端口在标记状态空闲。每个字符发送包含 1 个启动位及随后的 8 个或 9 个数据位，并始终由 1 个或多个停止位终止。启动位始终是一个空格，停止位始终是标记。最常见的数据格式为 8 位。每个发送位保持 1/(波特率) 个周期。使用片上专用 8 位 /16 位波特率发生器从系统振荡器产生标准波特率频率。波特率配置示例请参见表 16-5。

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，但它们的数据格式和波特率相同。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。

16.1.1 EUSART 异步发送器

图 16-1 给出了 EUSART 发送器框图。发送器的核心是串行发送移位寄存器（Transmit Shift Register，TSR），该寄存器不可用软件直接访问。TSR 从发送缓冲区（即 TXREG 寄存器）取得数据。

16.1.1.1 使能发送器

EUSART 发送器可通过配置以下 3 个控制位使能为异步操作：

- TXEN = 1
- SYNC = 0
- SPEN = 1

假定所有其他 EUSART 控制位均处于其默认状态。

将 TXSTA 寄存器的 TXEN 位置 1 使能 EUSART 的发送器电路。清零 TXSTA 寄存器的 SYNC 位将 EUSART 配置为异步操作。将 RCSTA 寄存器的 SPEN 位置 1 使能 EUSART 并自动将 TX/CK I/O 引脚配置为输出。如果 TX/CK 引脚与模拟外设共用，那么必须通过清零对应的 ANSEL 位禁止模拟 I/O 功能。

注 1： 当 SPEN 位置 1 时，RX/DT I/O 引脚被自动配置为输入，无论相应 TRIS 位的状态如何以及 EUSART 接收器是否被使能。通过正常的端口读操作可读取 RX/DT 引脚的数据，但不包括端口锁存器的数据输出值。

2: TXEN 允许位置 1 时，TXIF 发送器中断标志位置 1。

16.1.1.2 发送数据

向 TXREG 寄存器写入一个字符时启动发送。如果这是首字符，或前一个字符被完全从 TSR 中送出，TXREG 中的数据就立即被传送到 TSR 寄存器。如果 TSR 中仍保存前一个字符的全部或部分，则新字符被保存在 TXREG 中，直到前一个字符的停止位被发送。之后，在 TXREG 中等待的字符在停止位发送后 1 个 TCY 内被传送到 TSR 中。TXREG 中的数据被传送到 TSR 后，启动位、数据位和停止位的发送序列立即开始。

16.1.1.3 发送数据极性

可通过 BAUDCON 寄存器的 CKTXP 位来控制发送数据的极性。该位的默认状态为 0，选择高电平有效发送空闲和数据位。将 CKTXP 位设为 1 将发送数据的极性取反，从而选择低电平有效空闲和数据位。CKTXP 位仅在异步模式下控制发送数据的极性。在同步模式下，CKTXP 位有不同的功能。

16.1.1.4 发送中断标志

只要 EUSART 发送器被使能，而且 TXREG 中没有等待发送的字符，PIR1 寄存器的 TXIF 中断标志位就被置 1。换句话说，只有在 TSR 中有字符，并且 TXREG 中还有一个排队等待发送的新字符时，TXIF 位才被清零。写入 TXREG 后并不立即清零 TXIF 标志位。执行写操作后的第二个指令周期 TXIF 才有效。写入 TXREG 后立即查询 TXIF 位将返回无效结果。 TXIF 位是只读的，不能用软件置 1 或清零。

将 PIE1 寄存器的 TXIE 中断允许位置 1 可允许 TXIF 中断。但是，只要 TXREG 为空，TXIF 标志位就会被置 1，无论 TXIE 允许位的状态如何。

要在发送数据时使用中断，应只在仍有数据要发送时才将 TXIE 位置 1。在将发送的最后一个字符写入 TXREG 后应清零 TXIE 中断允许位。

16.1.1.5 TSR 状态

TXSTA 寄存器的 TRMT 位指示 TSR 寄存器的状态。该位是只读位。TSR 寄存器为空时，TRMT 位置 1，而当一个字符从 TXREG 传送到 TSR 寄存器中时，该位清零。 TRMT 位将保持清零，直到所有位移出 TSR 寄存器。该位不与任何中断逻辑相关联，因此用户需要查询该位以确定 TSR 的状态。

注：	TSR 寄存器不映射到数据存储器中，因此用户无法使用。

16.1.1.6 发送 9 位字符

EUSART 支持 9 位字符发送。当 TXSTA 寄存器的 TX9 位置 1 时， EUSART 将在发送每个字符时移出 9 位。TXSTA 寄存器的 TX9D 位是第 9 个数据位，也是最高有效位。发送 9 位数据时， TX9D 数据位必须先于低 8 位写入 TXREG。写入 TXREG 后，所有 9 个位将被立即传送到 TSR 移位寄存器中。

有多个接收器时，可使用一种特殊的 9 位地址模式。关于地址模式的更多信息，请参见**第 16.1.2.8 节 “ 地址检测 ”**。

16.1.1.7 异步发送设置

1. 初始化SPBRGH:SPBRG寄存器对以及BRGH和BRG16 位，获得所需的波特率 （见**第 16.3 节 “ EUSART 波特率发生器 （BRG）”**）。
2. 清零SYNC位并将SPEN位置1，使能异步串口。
3. 如果需要 9 位发送，将 TX9 控制位置 1。如果第 9 个数据位置 1，则表示发送器置于检测地址时，低 8 位为地址。
4. 如果需要反相的发送数据极性，将 CKTXP 控制位置 1。
5. 将 TXEN 控制位置 1 使能发送。这将导致 TXIF 中断标志位置 1。
6. 如果需要中断，将 TXIE 中断允许位置 1。如果 INTCON 寄存器的 GIE 和 PEIE 位也置 1，则立即产生中断。
7. 如果选择了 9 位发送，应将第 9 位装入 TX9D 数据位。
8. 将8位数据装入TXREG寄存器。这将启动发送。

图 16-3：　　异步发送

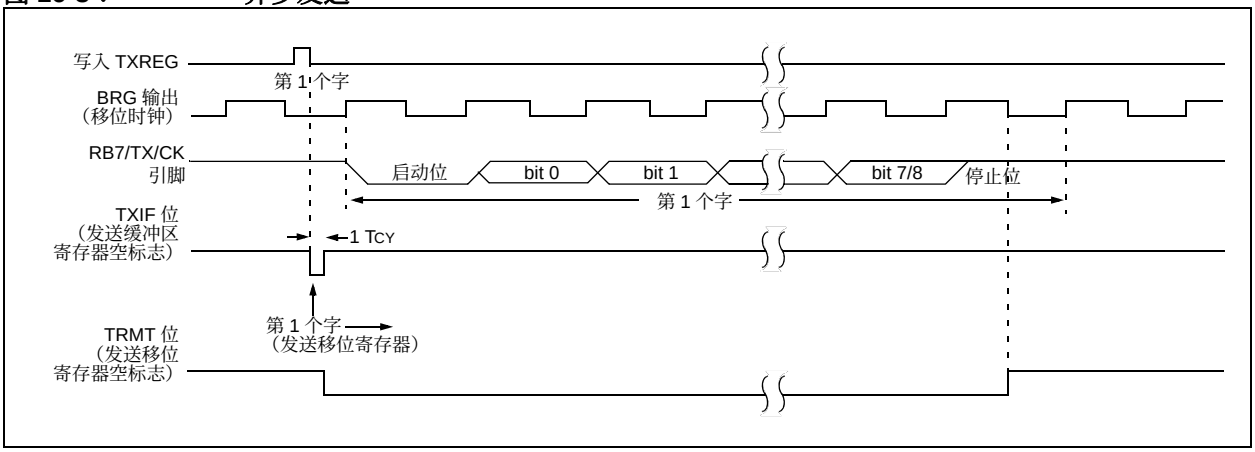
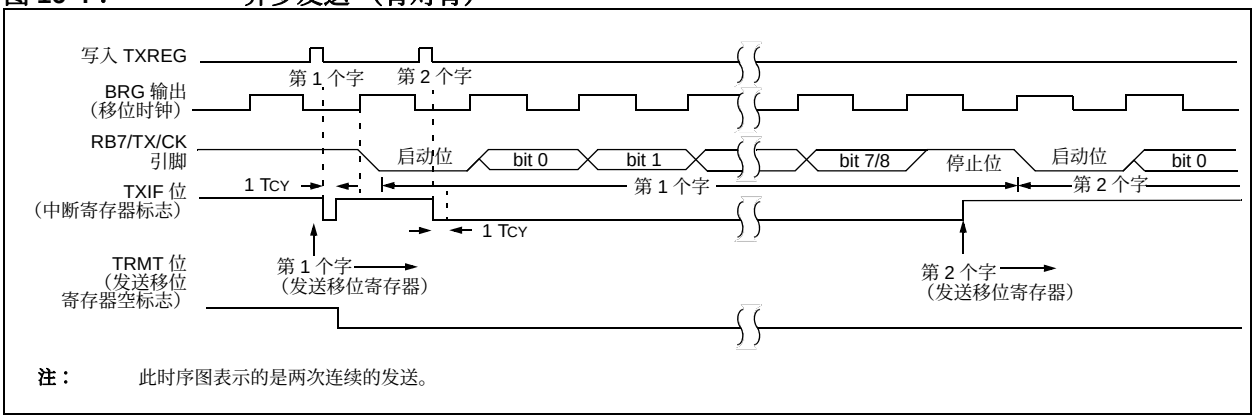


图 16-4：　　异步发送 （背对背）



注：　此时序图表示的是两次连续的发送。

表 16-1：　　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	RABIE	TMR0IF	INT0IF	RABIF	285
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	288
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	288
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	288
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	287
TXREG	EUSART 发送寄存器								287
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	287
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	287
SPBRGH	EUSART 波特率发生器寄存器的高字节								287
SPBRG	EUSART 波特率发生器寄存器的低字节								287

图注：　— = 未实现位，读为 0。异步发送不使用阴影单元。

16.1.2 EUSART 异步接收器

异步模式一般用于 RS-232 系统中。图 16-2 给出了接收器框图。数据在 RX/DT 引脚上接收并驱动数据恢复电路。数据恢复电路实际上是一个高速移位器，工作频率为 16 倍波特率，而串行接收移位寄存器（Receive Shift Register， RSR）工作频率为比特率。所有 8 位或 9 位字符移入后被立即传送到双字符的先进先出（First-In-First-Out，FIFO）存储区中。FIFO 缓冲区允许先接收两个完整字符和第三个字符的开始部分后，再启动软件服务 EUSART 接收器。FIFO 和 RSR 寄存器不能直接用软件访问。通过 RCREG 寄存器访问接收数据。

16.1.2.1 使能接收器

EUSART 接收器可通过配置以下 3 个控制位使能为异步操作：

- CREN = 1
- SYNC = 0
- SPEN = 1

假定所有其他 EUSART 控制位均处于其默认状态。

将 RCSTA 寄存器的 CREN 位置 1 使能 EUSART 的接收器电路。清零 TXSTA 寄存器的 SYNC 位将 EUSART 配置为异步操作。将 RCSTA 寄存器的 SPEN 位置 1 可使能 EUSART。必须通过将对应的 TRIS 控制位置 1 将 RX/DT I/O 引脚配置为输入。如果 RX/DT 引脚与模拟外设共用，那么必须通过清零对应的 ANSEL 位禁止模拟 I/O 功能。

注：	当 SPEN 位置 1 时， TX/CK I/O 引脚被自动配置为输出，无论相应 TRIS 位的状态如何以及 EUSART 发送器是否被使能。端口锁存器与输出驱动器断开，因此不可能将 TX/CK 引脚用作通用输出。

16.1.2.2 接收数据

接收器的数据恢复电路在第一位的下降沿启动字符接收。第一位也称启动（Start）位，始终为零。数据恢复电路计数一个半位时间至启动位的中点并验证该位是否仍为零。如果该位非零则数据恢复电路中止字符接收，不产生错误，并恢复寻找启动位的下降沿。如果启动位被验证为零，则数据恢复电路计数一整个位时间至下个位的中点。该位被一个择多检测电路采样，其结果（0 或 1）被移入 RSR。重复此过程直到所有数据位均被采样并移入 RSR。最后一个位时间被测量且其电平被采样。此为停止（Stop）位，始终为 1。如果数据恢复电路在停止位处采样到 0，则产生此字符的帧错误，否则此字符的帧错误被清零。关于帧错误的更多信息，请参见**第 16.1.2.5 节“ 接收帧错误 ”**。

所有数据位和停止位被接收后，RSR 中的字符就被立即传送到 EUSART 接收 FIFO，且 PIR1 寄存器的 RCIF 中断标志位被置 1。读取 RCREG 寄存器时，FIFO 中顶部的字符被送出 FIFO。

注：	如果接收 FIFO 溢出，在溢出条件被清除前不会接收更多字符。关于溢出错误的更多信息，请参见 **第 16.1.2.6 节“ 接收溢出错误 ”**。

16.1.2.3 接收数据极性

可通过 BAUDCON 寄存器的 DTRXP 位来控制接收数据的极性。该位的默认状态为 0，选择高电平有效接收空闲和数据位。将 DTRXP 位设为 1 将反相接收数据，从而选择低电平有效空闲和数据位。 DTRXP 位仅在异步模式下控制接收数据的极性。在同步模式下， DTRXP 位有不同的功能。

16.1.2.4 接收中断

只要 EUSART 接收器被使能且接收 FIFO 中存在未被读取的字符， PIR1 寄存器的 RCIF 中断标志位就会被置 1。RCIF 中断标志位是只读位，不能用软件置 1 或清零。

将以下位置 1 可允许 RCIF 中断：

- PIE1 寄存器的 RCIE 中断允许位
- INTCON 寄存器的 PEIE 外设中断允许位
- INTCON 寄存器的 GIE 全局中断允许位

当 FIFO 中存在未被读取的字符时，无论中断允许位的状态如何， RCIF 中断标志位均会被置 1。

16.1.2.5 接收帧错误

接收 FIFO 缓冲区中的每个字符都有相应的帧错误状态位。帧错误表明在预期时间内未见到停止位。通过 RCSTA 寄存器的 FERR 位可访问帧错误状态。 FERR 位表示接收 FIFO 中顶部的未读字符的状态。因此，在读取 RCREG 前必须读出 FERR 位。

FERR 位是只读位，只用于接收 FIFO 中顶部的未读字符。帧错误 （FERR = 1）不排除接收额外字符。此时不必将 FERR 位清零。从 FIFO 缓冲区读出下一个字符将使操作针对 FIFO 的下一个字符和下一个相应的帧错误。

将 RCSTA 寄存器的 SPEN 位清零可复位 EUSART，这样就可将 FERR 位强制清零。将 RCSTA 寄存器的 CREN 位清零不影响 FERR 位。自身产生的帧错误不会产生中断。

注：	如果接收 FIFO 中的所有接收字符均有帧错误，反复读取 RCREG 不会将 FERR 位清零。

16.1.2.6 接收溢出错误

接收 FIFO 缓冲区可容纳两个字符。在访问 FIFO 前接收到完整的第三个字符时会产生溢出错误。此时，RCSTA 寄存器的 OERR 位置 1。FIFO 缓冲区中已有的字符可被读出，但溢出错误被清除前不能再接收其他字符。将 RCSTA 寄存器的 CREN 位清零或通过将 RCSTA 寄存器的 SPEN 位清零复位 EUSART，可清除该错误。

16.1.2.7 接收 9 位字符

EUSART 支持 9 位字符接收。当 RCSTA 寄存器的 RX9 位置 1 时，EUSART 将在接收每个字符时将 9 个位移入 RSR。 RCSTA 寄存器的 RX9D 位是第 9 位，也是接收 FIFO 顶部未读字符的最高有效位。从接收 FIFO 缓冲区读取 9 位数据时，在读取 RCREG 的低 8 位前必须先读取 RX9D 数据位。

16.1.2.8 地址检测

当多个接收器共用同一条发送线时，如在 RS-485 系统中，有一个特殊的地址检测模式可供使用。将 RCSTA 寄存器的 ADDEN 位置 1 可使能地址检测。

地址检测要求接收 9 位字符。使能地址检测时，只有第 9 个数据位置 1 的字符会被传送到接收 FIFO 缓冲区，并将 RCIF 中断标志位置 1。所有其他字符均被忽略。

接收到地址字符后，用户软件判断地址是否与本地地址匹配。地址匹配时，发生下一个停止位前，用户软件必须通过清零 ADDEN 位禁止地址检测。当用户软件根据所使用的报文协议检测到报文的末尾时，软件将 ADDEN 位置 1，将接收器重新置于地址检测模式。

16.1.2.9 异步接收设置

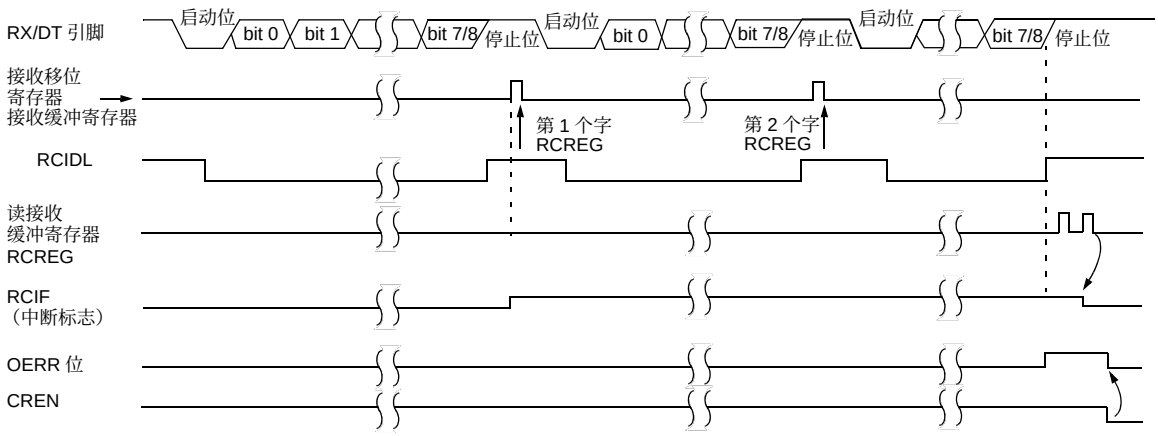
1. 初始化 SPBRGH:SPBRG 寄存器对以及 BRGH 和 BRG16 位，获得所需的波特率（见**第 16.3 节 “ EUSART 波特率发生器 （BRG）”**）。
2. 将 SPEN 位和 RX/DT 引脚的 TRIS 位置 1 使能串口。 SYNC 位必须清零才能进行异步操作。
3. 如果需要中断，将 RCIE 中断允许位置 1，并将 INTCON 寄存器的 GIE 和 PEIE 位置 1。
4. 如果需要接收 9 位数据，将 RX9 位置 1。
5. 如果需要反相的接收极性，将 DTRXP 置 1。
6. 将 CREN 位置 1 使能接收。
7. 当字符从 RSR 被移入接收缓冲区时，RCIF 中断标志位将被置 1。如果 RCIE 中断允许位也置 1，则产生中断。
8. 读取 RCSTA 寄存器取得错误标志，以及第 9 个数据位 （9 位数据接收使能时）。
9. 读取 RCREG 寄存器从接收缓冲区取得接收数据的低 8 位。
10. 发生溢出时，通过清零 CREN 接收器使能位清零 OERR 标志位。

16.1.2.10 9 位地址检测模式设置

此模式通常用于 RS-485 系统中。要设置使能地址检测的异步接收：

1. 初始化 SPBRGH:SPBRG 寄存器对以及 BRGH 和 BRG16 位，获得所需的波特率（见**第 16.3 节 “ EUSART 波特率发生器 （BRG）”**）。
2. 将 SPEN 位置 1 使能串口。SYNC 位必须清零才能进行异步操作。
3. 如果需要中断，将 RCIE 中断允许位置 1，并将 INTCON 寄存器的 GIE 和 PEIE 位置 1。
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. 如果需要反相的接收极性，将 DTRXP 置 1。
7. 将 CREN 位置 1 使能接收。
8. 当第 9 位置 1 的字符从 RSR 被移入接收缓冲区时，RCIF 中断标志位将被置 1。如果 RCIE 中断允许位也置 1，则产生中断。
9. 读取 RCSTA 寄存器取得错误标志。第 9 个数据位将始终置 1。
10. 读取 RCREG 寄存器从接收缓冲区取得接收数据的低 8 位。软件将判断此地址是否是器件地址。
11. 发生溢出时，通过清零 CREN 接收器使能位清零 OERR 标志位。
12. 如果器件被寻址，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产生中断。

图 16-5： 异步接收



注： 此时序图显示了在 RX 输入引脚上顺序接收的 3 个字。在第 3 个字后读 RCREG （接收缓冲区）会使 OERR （溢出）位置 1。

表 16-2：　　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	RABIE	TMR0IF	INT0IF	RABIF	285
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	288
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	288
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	288
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	287
RCREG	EUSART 接收寄存器								287
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	288
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	287
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	287
SPBRGH	EUSART 波特率发生器寄存器的高字节								287
SPBRG	EUSART 波特率发生器寄存器的低字节								287

图注：　— = 未实现位，读为 0。异步接收不使用阴影单元。

16.2 异步操作的时钟精确性

内部振荡器模块输出（HFINTOSC）在出厂时做了校准。但是，VDD 或温度变化时 HFINTOSC 频率有可能漂移，这将直接影响异步波特率。有两种方法可用来调整波特率时钟，但它们都需要某种参考时钟源。

第一种（推荐）方法使用 OSCTUNE 寄存器调整 HFINTOSC 输出。调整 OSCTUNE 寄存器的值可对系统时钟源的分辨率进行微调。更多信息，请参见**第 2.6.1 节“OSCTUNE 寄存器”**。

另一种方法调整波特率发生器的值。自动波特率检测可自动完成这种调整（见**第 16.3.1 节“自动波特率检测”**）。调整波特率发生器以补偿外设时钟频率的逐渐变化时，分辨率可能不够理想。

寄存器 16-1： TXSTA：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（时钟来自内部 BRG）
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（完成后由硬件清零）
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 的优先级高于 TXEN。

寄存器 16-2： RCSTA：接收状态和控制寄存器 (1)

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 RX/DT 和 TX/CK 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、允许中断和装入接收缓冲区
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
8 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 RCREG 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 16-3：　　BAUDCON：波特率控制寄存器

R-0	R-1	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
ABDOVF	RCIDL	DTRXP	CKTXP	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：**自动波特率检测溢出位
异步模式：
1 = 自动波特率定时器溢出
0 = 自动波特率定时器未溢出
同步模式：
无关位

bit 6　**RCIDL**：接收空闲标志位
异步模式：
1 = 接收器空闲
0 = 已检测到启动位且接收器处于活动状态
同步模式：
无关位

bit 5　**DTRXP**：数据 / 接收极性选择位
异步模式：
1 = 接收数据 （RX）反相 （低电平有效）
0 = 接收数据 （RX）未反相 （高电平有效）
同步模式：
1 = 数据 （DT）反相 （低电平有效）
0 = 数据 （DT）未反相 （高电平有效）

bit 4　**CKTXP**：时钟 / 发送极性选择位
异步模式：
1 = 发送 （TX）的空闲状态为低电平
0 = 发送 （TX）的空闲状态为高电平
同步模式：
1 = 在时钟的下降沿改变数据，在时钟的上升沿采样数据
0 = 在时钟的上升沿改变数据，在时钟的下降沿采样数据

bit 3　**BRG16：**16 位波特率发生器位
1 = 使用 16 位波特率发生器 （SPBRGH:SPBRG）
0 = 使用 8 位波特率发生器 （SPBRG）

bit 2　**未实现：**读为 0

bit 1　**WUE：**唤醒使能位
异步模式：
1 = 接收器正在等待下降沿。不会接收到任何字符，但 RCIF 在下降沿将被置 1。WUE 在上升沿将自动清零。
0 = 接收器正常工作
同步模式：
无关位

bit 0　**ABDEN**：自动波特率检测使能位
异步模式：
1 = 使能自动波特率模式 （完成自动波特率检测后清零）
0 = 禁止自动波特率模式
同步模式：
无关位

16.3 EUSART 波特率发生器 （BRG）

波特率发生器 （BRG）是 8 位或 16 位定时器，专用于支持异步和同步 EUSART 操作。默认情况下， BRG 工作在 8 位模式下。将 BAUDCON 寄存器的 BRG16 位置 1 可选择 16 位模式。

SPBRGH:SPBRG 寄存器对决定自由运行波特率定时器的周期。在异步模式下，波特率周期的倍数由 TXSTA 寄存器的 BRGH 位和 BAUDCON 寄存器的 BRG16 位决定。在同步模式下， BRGH 位被忽略。

表 16-3 提供了确定波特率的公式。例 16-1 提供了确定波特率和波特率误差的计算示例。

为便于您使用，各种异步模式的典型波特率和误差值已经计算出来，如表 16-5 所示。使用高波特率 （BRGH = 1）或 16 位 BRG （BRG16 = 1）有助于降低波特率误差。16 位 BRG 模式用于在高速振荡器频率下取得较缓慢的波特率。

将新值写入 SPBRGH:SPBRG 寄存器对将导致 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。

如果系统时钟在有效的接收操作过程中发生变化，可能会导致接收错误或数据丢失。为避免此问题，应检查 RCIDL 位的状态，以确保在改变系统时钟前接收操作处于空闲状态。

例 16-1： 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：

$$\text{目标波特率} = \frac{FOSC}{64([SPBRGH:SPBRG] + 1)}$$

求解 SPBRGH:SPBRG：

$$X = \left(\frac{FOSC}{64 * (\text{目标波特率})}\right) - 1$$

$$= \left(\frac{16{,}000{,}000}{64 * 9600}\right) - 1$$

$$= [25.042] = 25$$

$$\text{计算波特率} = \frac{16000000}{64(25 + 1)}$$

$$= 9615$$

$$\text{误差} = \frac{\text{计算波特率} - \text{目标波特率}}{\text{目标波特率}}$$

$$= \frac{(9615 - 9600)}{9600} = 0.16\%$$

表 16-3： 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 = SPBRGH:SPBRG 寄存器对的值

表 16-4：　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
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	287
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	287
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	287
SPBRGH	EUSART 波特率发生器寄存器的高字节								287
SPBRG	EUSART 波特率发生器寄存器的低字节								287

图注：　— = 未实现，读为 0。 BRG 不使用阴影单元。

表 16-5：　异步模式下的波特率

波特率	SYNC = 0， BRGH = 0， BRG16 = 0											
	Fosc = 48.000 MHz			Fosc = 18.432 MHz			Fosc = 12.000 MHz			Fosc = 11.0592 MHz		
	实际波特率	误差 %	SPBRG 值（十进制）	实际波特率	误差 %	SPBRG 值（十进制）	实际波特率	误差 %	SPBRG 值（十进制）	实际波特率	误差 %	SPBRG 值（十进制）
300	—	—	—	—	—	—	—	—	—	—	—	—
1200	—	—	—	1200	0.00	239	1202	0.16	155	1200	0.00	143
2400	—	—	—	2400	0.00	119	2404	0.16	77	2400	0.00	71
9600	9615	0.16	77	9600	0.00	29	9375	-2.34	19	9600	0.00	17
10417	10417	0.00	71	10286	-1.26	27	10417	0.00	17	10165	-2.42	16
19.2k	19.23k	0.16	38	19.20k	0.00	14	18.75k	-2.34	9	19.20k	0.00	8
57.6k	57.69k	0.16	12	57.60k	0.00	7	—	—	—	57.60k	0.00	2
115.2k	—	—	—	—	—	—	—	—	—	—	—	—

波特率	SYNC = 0， BRGH = 0， BRG16 = 0											
	Fosc = 8.000 MHz			Fosc = 4.000 MHz			Fosc = 3.6864 MHz			Fosc = 1.000 MHz		
	实际波特率	误差 %	SPBRG 值（十进制）	实际波特率	误差 %	SPBRG 值（十进制）	实际波特率	误差 %	SPBRG 值（十进制）	实际波特率	误差 %	SPBRG 值（十进制）
300	—	—	—	300	0.16	207	300	0.00	191	300	0.16	51
1200	1202	0.16	103	1202	0.16	51	1200	0.00	47	1202	0.16	12
2400	2404	0.16	51	2404	0.16	25	2400	0.00	23	—	—	—
9600	9615	0.16	12	—	—	—	9600	0.00	5	—	—	—
10417	10417	0.00	11	10417	0.00	5	—	—	—	—	—	—
19.2k	—	—	—	—	—	—	19.20k	0.00	2	—	—	—
57.6k	—	—	—	—	—	—	57.60k	0.00	0	—	—	—
115.2k	—	—	—	—	—	—	—	—	—	—	—	—

波特率	SYNC = 0， BRGH = 1， BRG16 = 0											
	Fosc = 48.000 MHz			Fosc = 18.432 MHz			Fosc = 12.000 MHz			Fosc = 11.0592 MHz		
	实际波特率	误差 %	SPBRG 值（十进制）	实际波特率	误差 %	SPBRG 值（十进制）	实际波特率	误差 %	SPBRG 值（十进制）	实际波特率	误差 %	SPBRG 值（十进制）
300	—	—	—	—	—	—	—	—	—	—	—	—
1200	—	—	—	—	—	—	—	—	—	—	—	—
2400	—	—	—	—	—	—	—	—	—	—	—	—
9600	—	—	—	9600	0.00	119	9615	0.16	77	9600	0.00	71
10417	—	—	—	10378	-0.37	110	10417	0.00	71	10473	0.53	65
19.2k	19.23k	0.16	155	19.20k	0.00	59	19.23k	0.16	38	19.20k	0.00	35
57.6k	57.69k	0.16	51	57.60k	0.00	19	57.69k	0.16	12	57.60k	0.00	11
115.2k	115.38	0.16	25	115.2k	0.00	9	—	—	—	115.2k	0.00	5

表 16-5：　　异步模式下的波特率 （续）

波特率	SYNC = 0， BRGH = 1， BRG16 = 0											
	FOSC = 8.000 MHz			FOSC = 4.000 MHz			FOSC = 3.6864 MHz			FOSC = 1.000 MHz		
	实际波特率	误差 %	SPBRG 值（十进制）	实际波特率	误差 %	SPBRG 值（十进制）	实际波特率	误差 %	SPBRG 值（十进制）	实际波特率	误差 %	SPBRG 值（十进制）
300	—	—	—	—	—	—	—	—	—	300	0.16	207
1200	—	—	—	1202	0.16	207	1200	0.00	191	1202	0.16	51
2400	2404	0.16	207	2404	0.16	103	2400	0.00	95	2404	0.16	25
9600	9615	0.16	51	9615	0.16	25	9600	0.00	23	—	—	—
10417	10417	0.00	47	10417	0.00	23	10473	0.53	21	10417	0.00	5
19.2k	19231	0.16	25	19.23k	0.16	12	19.2k	0.00	11	—	—	—
57.6k	55556	-3.55	8	—	—	—	57.60k	0.00	3	—	—	—
115.2k	—	—	—	—	—	—	115.2k	0.00	1	—	—	—

波特率	SYNC = 0， BRGH = 0， BRG16 = 1											
	FOSC = 48.000 MHz			FOSC = 18.432 MHz			FOSC = 12.000 MHz			FOSC = 11.0592 MHz		
	实际波特率	误差 %	SPBRGH: SPBRG（十进制）	实际波特率	误差 %	SPBRGH: SPBRG（十进制）	实际波特率	误差 %	SPBRGH: SPBRG（十进制）	实际波特率	误差 %	SPBRGH: SPBRG（十进制）
300	300.0	0.00	9999	300.0	0.00	3839	300	0.00	2499	300.0	0.00	2303
1200	1200.1	0.00	2499	1200	0.00	959	1200	0.00	624	1200	0.00	575
2400	2400	0.00	1249	2400	0.00	479	2404	0.16	311	2400	0.00	287
9600	9615	0.16	311	9600	0.00	119	9615	0.16	77	9600	0.00	71
10417	10417	0.00	287	10378	-0.37	110	10417	0.00	71	10473	0.53	65
19.2k	19.23k	0.16	155	19.20k	0.00	59	19.23k	0.16	38	19.20k	0.00	35
57.6k	57.69k	0.16	51	57.60k	0.00	19	57.69k	0.16	12	57.60k	0.00	11
115.2k	115.38	0.16	25	115.2k	0.00	9	—	—	—	115.2k	0.00	5

波特率	SYNC = 0， BRGH = 0， BRG16 = 1											
	FOSC = 8.000 MHz			FOSC = 4.000 MHz			FOSC = 3.6864 MHz			FOSC = 1.000 MHz		
	实际波特率	误差 %	SPBRGH: SPBRG（十进制）	实际波特率	误差 %	SPBRGH: SPBRG（十进制）	实际波特率	误差 %	SPBRGH: SPBRG（十进制）	实际波特率	误差 %	SPBRGH: SPBRG（十进制）
300	299.9	-0.02	1666	300.1	0.04	832	300.0	0.00	767	300.5	0.16	207
1200	1199	-0.08	416	1202	0.16	207	1200	0.00	191	1202	0.16	51
2400	2404	0.16	207	2404	0.16	103	2400	0.00	95	2404	0.16	25
9600	9615	0.16	51	9615	0.16	25	9600	0.00	23	—	—	—
10417	10417	0.00	47	10417	0.00	23	10473	0.53	21	10417	0.00	5
19.2k	19.23k	0.16	25	19.23k	0.16	12	19.20k	0.00	11	—	—	—
57.6k	55556	-3.55	8	—	—	—	57.60k	0.00	3	—	—	—
115.2k	—	—	—	—	—	—	115.2k	0.00	1	—	—	—

表 16-5：　　异步模式下的波特率 （续）

波特率	SYNC = 0，BRGH = 1，BRG16 = 1 或 SYNC = 1，BRG16 = 1											
	FOSC = 48.000 MHz			FOSC = 18.432 MHz			FOSC = 12.000 MHz			FOSC = 11.0592 MHz		
	实际波特率	误差 %	SPBRGH: SPBRG（十进制）	实际波特率	误差 %	SPBRGH: SPBRG（十进制）	实际波特率	误差 %	SPBRGH: SPBRG（十进制）	实际波特率	误差 %	SPBRGH: SPBRG（十进制）
300	300	0.00	39999	300.0	0.00	15359	300	0.00	9999	300.0	0.00	9215
1200	1200	0.00	9999	1200	0.00	3839	1200	0.00	2499	1200	0.00	2303
2400	2400	0.00	4999	2400	0.00	1919	2400	0.00	1249	2400	0.00	1151
9600	9600	0.00	1249	9600	0.00	479	9615	0.16	311	9600	0.00	287
10417	10417	0.00	1151	10425	0.08	441	10417	0.00	287	10433	0.16	264
19.2k	19.20k	0.00	624	19.20k	0.00	239	19.23k	0.16	155	19.20k	0.00	143
57.6k	57.69k	0.16	207	57.60k	0.00	79	57.69k	0.16	51	57.60k	0.00	47
115.2k	115.38	0.16	103	115.2k	0.00	39	115.38k	0.16	25	115.2k	0.00	23

波特率	SYNC = 0，BRGH = 1，BRG16 = 1 或 SYNC = 1，BRG16 = 1											
	FOSC = 8.000 MHz			FOSC = 4.000 MHz			FOSC = 3.6864 MHz			FOSC = 1.000 MHz		
	实际波特率	误差 %	SPBRGH: SPBRG（十进制）	实际波特率	误差 %	SPBRGH: SPBRG（十进制）	实际波特率	误差 %	SPBRGH: SPBRG（十进制）	实际波特率	误差 %	SPBRGH: SPBRG（十进制）
300	300.0	0.00	6666	300.0	0.01	3332	300.0	0.00	3071	300.1	0.04	832
1200	1200	-0.02	1666	1200	0.04	832	1200	0.00	767	1202	0.16	207
2400	2401	0.04	832	2398	0.08	416	2400	0.00	383	2404	0.16	103
9600	9615	0.16	207	9615	0.16	103	9600	0.00	95	9615	0.16	25
10417	10417	0.00	191	10417	0.00	95	10473	0.53	87	10417	0.00	23
19.2k	19.23k	0.16	103	19.23k	0.16	51	19.20k	0.00	47	19.23k	0.16	12
57.6k	57.14k	-0.79	34	58.82k	2.12	16	57.60k	0.00	15	—	—	—
115.2k	117.6k	2.12	16	111.1k	-3.55	8	115.2k	0.00	7	—	—	—

16.3.1 自动波特率检测

EUSART 模块支持波特率自动检测和校准。

在自动波特率检测（Auto-Baud Rate Detect，ABD）模式下，BRG 的时钟信号反向。BRG 并不为进入的 RX 信号提供时钟信号，而是相反由 RX 信号为 BRG 定时。波特率发生器用于为接收的 55h（ASCII “ U ”）定时，这是 LIN 总线的同步字符。此字符的特殊之处在于它具有包括停止位边沿在内的 5 个上升沿。

将 BAUDCON 寄存器的 ABDEN 位置 1 将启动自动波特率校准序列（图 16-6）。当发生 ABD 序列时，EUSART 状态机保持在空闲状态。在接收线的第一个上升沿（启动位之后），SPBRG 使用 BRG 计数器时钟递增计数，如表 16-6 所示。在第 8 位周期的末尾将在 RX 引脚上出现第 5 个上升沿。此时，累计数据即正确的 BRG 周期总数被留在 SPBRGH:SPBRG 寄存器对中，ABDEN 位被自动清零而 RCIF 中断标志被置 1。要清除 RCIF 中断，需要执行对 RCREG 的读操作。RCREG 的内容应该被丢弃。校准不使用 SPBRGH 寄存器的模式时，用户可通过查询 SPBRGH 寄存器的值是否为 00h 来验证 SPBRG 寄存器是否溢出。

BRG 自动波特率时钟由 BRG16 和 BRGH 位决定，如表 16-6 所示。在 ABD 期间，SPBRGH 和 SPBRG 寄存器共同用作 16 位计数器，这与 BRG16 位的设置无关。在校准波特率周期时，SPBRGH 和 SPBRG 寄存器的时钟频率为 BRG 基本时钟频率的 1/8。得到的字节测量结果为全速下的平均位时间。

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（见**第 16.3.3 节 “ 接收到间隔字符时自动唤醒 ”**）。

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。某些振荡器频率和 EUSART 波特率组合不可能实现。

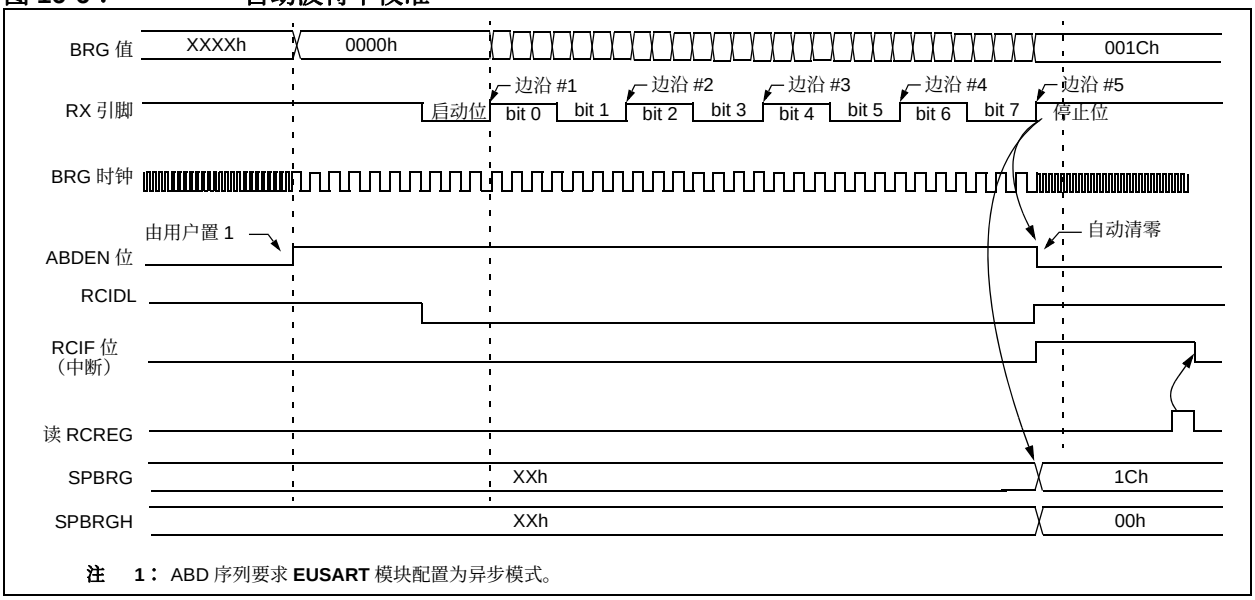
3: 在自动波特率过程中，自动波特率计数器从 1 开始计数。自动波特率序列完成后，为了得到最准确的结果，应从 SPBRGH:SPBRG 寄存器对的值中减去 1。

表 16-6： BRG 计数器时钟速率

BRG16	BRGH	BRG 基本时钟	BRG ABD 时钟
0	0	FOSC/64	FOSC/512
0	1	FOSC/16	FOSC/128
1	0	FOSC/16	FOSC/128
1	1	FOSC/4	FOSC/32

注： 在 ABD 序列期间，SPBRG 和 SPBRGH 寄存器同时用作 16 位计数器，与 BRG16 的设置无关。

图 16-6： 自动波特率校准



注 **1：** ABD 序列要求 **EUSART** 模块配置为异步模式。

16.3.2 自动波特率溢出

在自动波特率检测过程中，如果在 RX 引脚上检测到第 5 个上升沿之前波特率计数器溢出，则 BAUDCON 寄存器的 ABDOVF 位将被置 1。 ABDOVF 位指示计数器已超出适合 SPBRGH:SPBRG 寄存器对的 16 位的最大计数。在 ABDOVF 置 1 后，计数器将继续计数，直到在 RX 引脚上检测到第 5 个上升沿为止。一旦检测到第 5 个 RX 边沿，硬件将把 RCIF 中断标志置 1，并将 BAUDCON 寄存器的 ABDEN 位清零。可以通过读取 RCREG 寄存器将 RCIF 标志清零。BAUDCON 寄存器的 ABDOVF 标志可以用软件直接清零。

若要在 RCIF 标志置 1 前终止自动波特率进程，请先将 ABDEN 位清零，然后将 BAUDCON 寄存器的 ABDOVF 位清零。如果没有先将 ABDEN 位清零，ABDOVF 位将保持置 1 状态。

16.3.3 接收到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，波特率发生器处于不工作状态，不能正常进行字符接收。自动唤醒功能使控制器可被 RX/DT 线上的活动唤醒。该功能只在异步模式下可用。

自动唤醒功能可通过将 BAUDCON 寄存器的 WUE 位置 1 来使能。一旦置 1， RX/DT 上的正常接收序列就被禁止， EUSART 保持在空闲状态，监视与 CPU 模式无关的唤醒事件。唤醒事件包含 RX/DT 线上由高至低的跳变。（这正好也是同步间隔字符的开始或LIN协议的唤醒信号字符。）

EUSART 模块产生的 RCIF 中断与唤醒事件同步。在正常CPU工作模式下，中断产生与Q时钟同步（图16-7），而器件处于休眠模式时则异步发生 （图 16-8）。通过读 RCREG 寄存器可清除中断条件。

在间隔字符末尾RX线由低至高的跳变将自动清零WUE位。这向用户表明间隔事件结束。此时，EUSART 模块处于空闲模式，等待接收下一个字符。

16.3.3.1 特殊注意事项

间隔字符

在发生唤醒事件期间为了避免字符错误或字符碎片，唤醒字符必须为全零。

唤醒被使能时，其工作与数据流的低电平时间无关。如果 WUE 位置 1 并接收到了有效的非零字符，则从启动位至第一个上升沿的低电平时间将被解读为唤醒事件。字符的其余位将作为碎片字符接收，后续字符有可能产生帧或溢出错误。

因此，发送的首字符必须为全 0。这必须持续 10 个或更长的位时间，对于 LIN 总线建议持续 13 个位时间，而标准 RS-232 器件可为任意个位时间。

振荡器起振时间

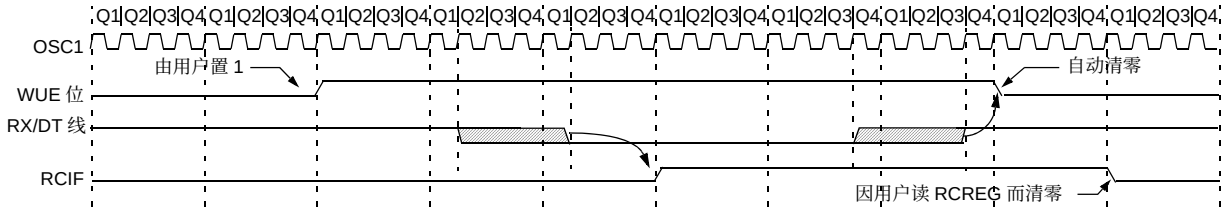
必须考虑振荡器起振时间，特别在使用具有较长起振时间的振荡器（即，LP、XT 或 HS/PLL 模式）的应用中。同步间隔（或唤醒信号）字符必须足够长，并随后有一个足够长的间隔时间，以使所选的振荡器有足够的时间起振并在这段时间对 EUSART 进行初始化。

WUE 位

唤醒事件会通过将RCIF位置1产生一个接收中断。WUE 位在 RX/DT 的上升沿由硬件清零。之后软件通过读取 RCREG 寄存器并丢弃其内容将中断条件清除。

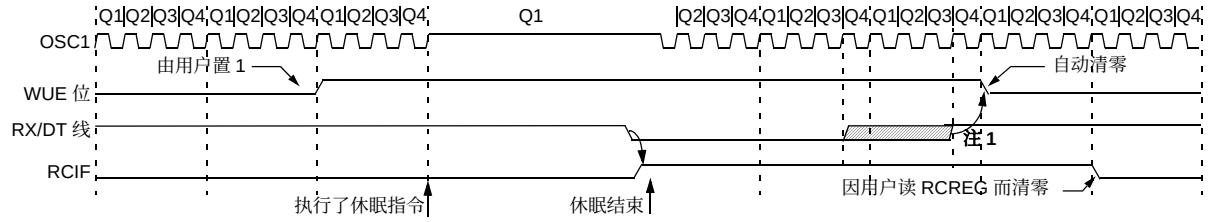
要确保不丢失实际数据，应在将 WUE 位置 1 前检查 RCIDL 位，验证没有接收操作在进行。如果未发生接收操作，可在进入休眠模式前将 WUE 位置 1。

图 16-7：　　正常操作时的自动唤醒位 （WUE）时序



注　1： WUE 位置 1 时， **EUSART** 保持在空闲状态。

图 16-8：　　休眠时的自动唤醒位 （WUE）时序



注　1： 如果唤醒事件需要较长的振荡器起振时间，在 stposc 信号仍然有效时可能发生 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。

16.3.4 间隔字符序列

EUSART 模块能够发送符合 LIN 总线标准的特殊间隔字符序列。间隔字符包含 1 个启动位，随后的 12 个 0 位和 1 个停止位。

要发送间隔字符，应将 TXSTA 寄存器的 SENDB 和 TXEN 位置 1。随后对 TXREG 执行写操作可启动间隔字符发送。写入 TXREG 的数据值会被忽略并发送全 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 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。

TXSTA 寄存器的 TRMT 位表明发送操作何时处于激活或空闲状态，这与正常发送时相同。关于发送间隔字符的时序，请参见图 16-9。

16.3.4.1 间隔和同步发送序列

以下序列将启动报文帧头（header），它由间隔字符和其后的自动波特率同步字节组成。这是 LIN 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 使能间隔序列。
3. 将无效字符装入 TXREG，启动发送（该值会被忽略）。
4. 将“55h”写入 TXREG，以便将同步字符装入发送 FIFO 缓冲区。
5. 发送间隔字符后，SENDB 位被硬件复位，同步字符随后被发送。

当 TXREG 为空时（由 TXIF 指出），下一个数据字节会写入 TXREG。

16.3.5 接收间隔字符

增强型 EUSART 模块接收间隔字符有两种方法。

第一种检测间隔字符的方法采用 RCSTA 寄存器的 FERR 位和 RCREG 所指示的接收数据。假定波特率发生器已初始化为所需的波特率。

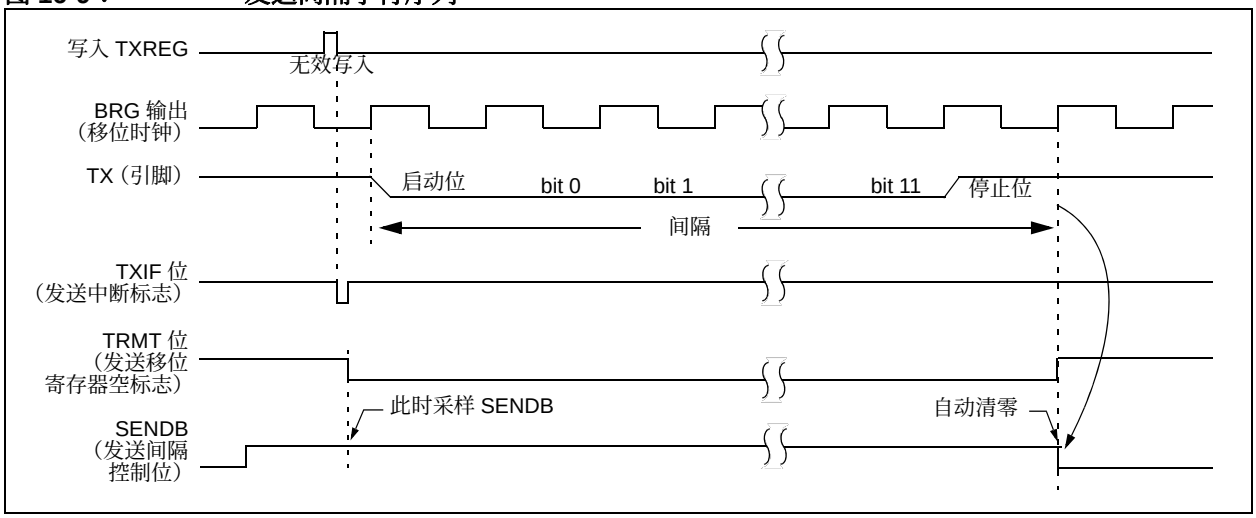
以下成立时，表明接收到间隔字符：

- RCIF 位被置 1
- FERR 位被置 1
- RCREG = 00h

第二种方法采用**第 16.3.3 节“接收到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 RX/DT 上的下两次跳变，产生 RCIF 中断，并接收下一个数据字节并再产生一次中断。

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。采用这两种方法时，用户均可在 EUSART 进入休眠模式前将 BAUDCON 寄存器的 ABDEN 位置 1。

图 16-9：　　发送间隔字符序列



16.4 EUSART 同步模式

同步串行通信通常用于具有一个主器件和一个或多个从器件的系统中。主器件包含生成波特率所需的电路，可将时钟提供给系统中的所有器件。从器件使用主时钟，可不再需要内部时钟生成电路。

同步模式下有两条信号线：双向数据线和时钟线。从器件使用主器件提供的外部时钟将串行数据移入或移出相应的接收和发送移位寄存器。由于数据线是双向的，同步操作只能是半双工的。半双工指主从器件能够接收和发送数据，但不能同时进行相同的操作。EUSART 可作为主器件，也可作为从器件。

同步发送时不使用启动位和停止位。

16.4.1 同步主模式

使用以下位将 EUSART 配置为同步主操作：

- SYNC = 1
- CSRC = 1
- SREN = 0（用于发送）；SREN = 1（用于接收）
- CREN = 0（用于发送）；CREN = 1（用于接收）
- SPEN = 1

将 TXSTA 寄存器的 SYNC 位置 1 将器件配置为同步操作。将 TXSTA 寄存器的 CSRC 位置 1 可将器件配置为主器件。将 RCSTA 寄存器的 SREN 和 CREN 位清零可确保器件处于发送模式，否则器件将被配置为接收。将 RCSTA 寄存器的 SPEN 位置 1 可使能 EUSART。如果 RX/DT 或 TX/CK 引脚与模拟外设共用，那么必须通过清零对应的 ANSEL 位禁止模拟 I/O 功能。

与 RX/DT 和 TX/CK 引脚对应的 TRIS 位应置 1。

16.4.1.1 主时钟

同步数据传送使用独立的时钟线，它与数据同步。配置为主器件的器件将时钟信号发送到 TX/CK 线上。EUSART 配置为同步发送或接收操作时，自动使能 TX/CK 引脚输出驱动器。串行数据位在时钟前沿改变，以确保其在时钟的后续边沿有效。为每个数据位产生一个时钟周期。数据位有多少，就产生多少个时钟周期。

16.4.1.2 时钟极性

提供了时钟极性选项以与 Microwire 兼容。时钟极性通过 BAUDCON 寄存器的 CKTXP 位选择。将 CKTXP 位置 1 将时钟空闲状态设置为高电平。当 CKTXP 位置 1 时，在每个时钟的下降沿改变数据，在每个时钟的上升沿采样数据。将 CKTXP 位清零将时钟空闲状态设置为低电平。当 CKTXP 位清零时，在每个时钟的上升沿改变数据，在每个时钟的下降沿采样数据。

16.4.1.3 同步主发送

从器件的 RX/DT 引脚输出数据。 EUSART 配置为同步主发送操作时， RX/DT 和 TX/CK 引脚的输出驱动器被自动使能。

向 TXREG 寄存器写入一个字符时启动发送。如果 TSR 中仍保存前一个字符的全部或部分，则新字符被保存在 TXREG 中，直到前一个字符的最后一位被发送。如果这是首字符，或前一个字符被完全从 TSR 中送出，TXREG 中的数据就立即被传送到 TSR。字符发送在数据从 TXREG 送入 TSR 后立即开始。

每个数据位在主时钟的时钟前沿改变，并在下一个时钟前沿到来前保持有效。

注：	TSR 寄存器不映射到数据存储器中，因此用户无法使用。

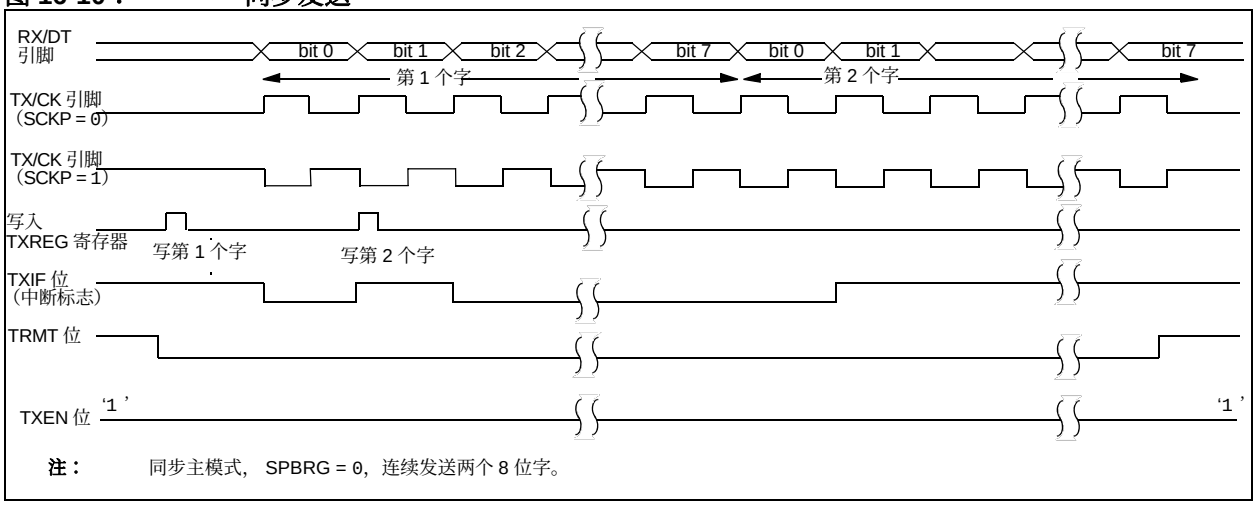
16.4.1.4 数据极性

可通过 BAUDCON 寄存器的 DTRXP 位来控制发送和接收数据的极性。该位的默认状态为 0，选择高电平有效发送和接收数据。将 DTRXP 位设为 1 将数据极性取反，从而选择低电平有效发送和接收数据。

16.4.1.5 同步主发送设置

1. 初始化 SPBRGH:SPBRG 寄存器对以及 BRGH 和 BRG16 位，获得所需的波特率（见**第 16.3 节 “ EUSART 波特率发生器 （BRG）”**）。
2. 将 SYNC、SPEN 和 CSRC 位置 1 使能同步主串口。将与 RX/DT 和 TX/CK I/O 引脚对应的 TRIS 位置 1。
3. 将 SREN 和 CREN 位清零禁止接收模式。
4. 将 TXEN 位置 1 使能发送模式。
5. 如果需要 9 位发送，将 TX9 位置 1。
6. 如果需要中断，将 TXIE、GIE 和 PEIE 中断允许位置 1。
7. 如果选择了 9 位发送，应将第 9 位装入 TX9D 位。
8. 将数据装入 TXREG 寄存器，启动发送。

图 16-10： 同步发送



注： 同步主模式， SPBRG = 0，连续发送两个 8 位字。

图 16-11： 同步发送 （由 TXEN 位控制）

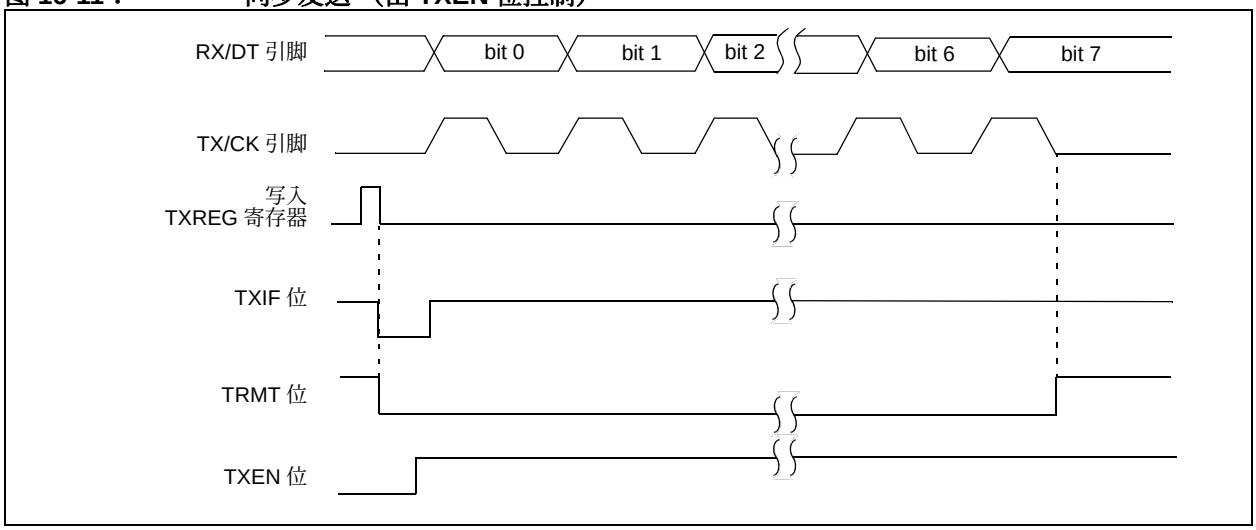


表 16-7： 与同步主发送相关的寄存器

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	RABIE	TMR0IF	INT0IF	RABIF	285
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	288
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	288
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	288
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	287
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	288
TXREG	EUSART 发送寄存器								287
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	287
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	287
SPBRGH	EUSART 波特率发生器寄存器的高字节								287
SPBRG	EUSART 波特率发生器寄存器的低字节								287

图注： — = 未实现，读为 0。同步主发送不使用阴影单元。

16.4.1.6 同步主接收

数据在 RX/DT 引脚上接收。将 EUSART 配置为同步主接收操作时，必须通过将对应的 TRIS 位置 1 来禁止 RX/DT 引脚输出驱动器。

在同步模式下，可通过将单字节接收使能位 （RCSTA 寄存器的 SREN）或连续接收使能位 （RCSTA 寄存器的 CREN）置 1 使能接收。

SREN 置 1 且 CREN 清零时，一个字符中有多少数据位就产生多少个时钟周期。一个字符接收完成后 SREN 位被自动清零。CREN置1时，将连续产生时钟直到CREN被清零。如果 CREN 在字符接收过程中被清零，则 CK 时钟立即停止，接收到的部分字符被丢弃。如果 SREN 和 CREN 同时置 1，则首字符接收完成时 SREN 被清零， CREN 优先。

要启动接收，将 SREN 或 CREN 置 1。在 TX/CK 时钟引脚的后续边沿对 RX/DT 引脚上的数据进行采样，并移入接收移位寄存器 （RSR） 。当完整的字符被接收进 RSR 后，RCIF 位置 1 且该字符被自动送入两个字符的接收 FIFO。接收 FIFO 中顶部字符的低 8 位在 RCREG 中。只要接收 FIFO 中有未读字符，RCIF 位就保持置 1。

16.4.1.7 从时钟

同步数据传送使用独立的时钟线，时钟与数据同步。配置为从器件的器件在 TX/CK 线上接收时钟信号。将器件配置为同步从发送或接收操作时，必须通过将相关的 TRIS 位置 1 来禁止 TX/CK 引脚输出驱动器。串行数据位在时钟前沿改变，以确保其在时钟的后续边沿有效。每个时钟周期传送一个数据位。数据位有多少，就产生多少个接收时钟周期。

16.4.1.8 接收溢出错误

接收 FIFO 缓冲区可容纳两个字符。在 RCREG 被读取以访问 FIFO 前，接收到完整的第三个字符时会产生溢出错误。此时，RCSTA 寄存器的 OERR 位置 1。FIFO 中的前一个数据不会被覆盖。 FIFO 缓冲区中的两个字符可被读出，但错误被清除前不能再接收其他字符。只有清除了溢出条件才可将 OERR 位清零。如果 SREN 位置 1 且 CREN 清零时发生溢出错误，则读取 RCREG 可清除错误。如果 CREN 位置 1 时发生溢出，则可通过清零 RCSTA 寄存器的 CREN 位或清零可将 EUSART 复位的 SPEN 位清除错误条件。

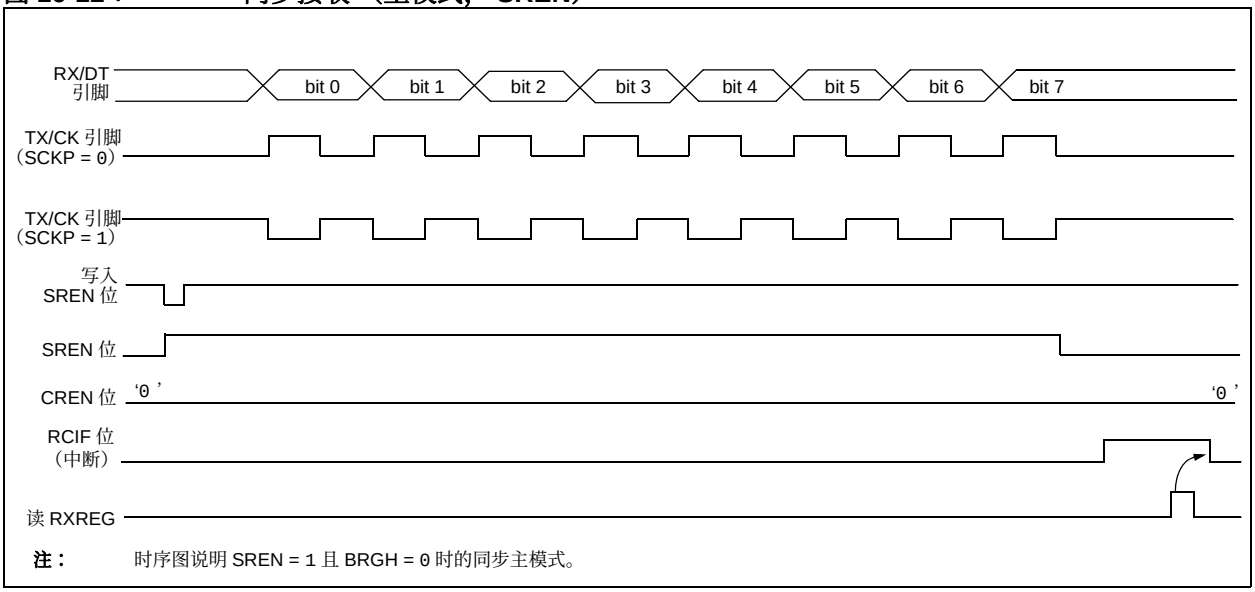
16.4.1.9 接收 9 位字符

EUSART 支持 9 位字符接收。当 RCSTA 寄存器的 RX9 位置 1 时，EUSART 将在接收每个字符时将 9 个位移入 RSR。RCSTA 寄存器的 RX9D 位是第 9 位，也是接收 FIFO 顶部未读字符的最高有效位。从接收 FIFO 缓冲区读取 9 位数据时，在读取 RCREG 的低 8 位前必须先读取 RX9D 数据位。

16.4.1.10 同步主接收设置

1. 初始化 SPBRGH:SPBRG 寄存器对，获得所需的波特率。按需要将 BRGH 和 BRG16 位置 1 或清零，获得所需的波特率。
2. 将 SYNC、SPEN 和 CSRC 位置 1 使能同步主串口。将对应的 TRIS 位置 1 禁止 RX/DT 和 TX/CK 输出驱动器。
3. 确保将 CREN 和 SREN 位清零。
4. 如果使用中断，将 INTCON 寄存器的 GIE 和 PEIE 位置 1 并将 RCIE 置 1。
5. 如果需要接收 9 位数据，将 RX9 位置 1。
6. 将 SREN 位置 1 启动接收，或将 CREN 位置 1 使能连续接收。
7. 字符接收完成时中断标志位 RCIF 将被置 1。如果允许位 RCIE 已置 1，则产生中断。
8. 读取 RCSTA 寄存器取得第 9 位（如果已使能），并确定接收时是否发生了错误。
9. 通过读取 RCREG 寄存器来读取接收到的 8 位数据。
10. 如果发生了溢出错误，可通过清零 RCSTA 寄存器的 CREN 位或清零可将 EUSART 复位的 SPEN 位清除错误。

图 16-12： 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 16-8：　　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	RABIE	TMR0IF	INT0IF	RABIF	285
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	288
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	288
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	288
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	287
RCREG	EUSART 接收寄存器								287
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	287
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	287
SPBRGH	EUSART 波特率发生器寄存器的高字节								287
SPBRG	EUSART 波特率发生器寄存器的低字节								287

图注：　— = 未实现，读为 0。同步主接收不使用阴影单元。

16.4.2　同步从模式

使用以下位将 EUSART 配置为同步从操作：

- SYNC = 1
- CSRC = 0
- SREN = 0 （用于发送）；SREN = 1 （用于接收）
- CREN = 0 （用于发送）；CREN = 1 （用于接收）
- SPEN = 1

将 TXSTA 寄存器的 SYNC 位置 1 将器件配置为同步操作。将 TXSTA 寄存器的 CSRC 位清零将器件配置为从器件。将 RCSTA 寄存器的 SREN 和 CREN 位清零可确保器件处于发送模式，否则器件将被配置为接收。将 RCSTA 寄存器的 SPEN 位置 1 可使能 EUSART。如果 RX/DT 或 TX/CK 引脚与模拟外设共用，那么必须通过清零对应的 ANSEL 位禁止模拟 I/O 功能。

必须通过将对应的 TRIS 位置 1 来禁止 RX/DT 和 TX/CK 输出驱动器。

16.4.2.1　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是相同的 （见**第 16.4.1.3 节 “ 同步主发送 ”**）。

如果向 TXREG 写入两个字，然后执行 SLEEP 指令，则会发生以下事件：

1. 第一个字符将立即传送到 TSR 寄存器并发送。
2. 第二个字将保留在 TXREG 寄存器中。
3. TXIF 位不会被置 1。
4. 第一个字符移出 TSR 后，TXREG 寄存器会将第二个字符传送到 TSR，此时 TXIF 位将置 1。
5. 如果 PEIE 和 TXIE 位均置 1，则发生中断将器件从休眠唤醒，并执行下一条指令。如果 GIE 位也置 1，程序将调用中断服务程序。

16.4.2.2　同步从发送设置

1. 将 SYNC 和 SPEN 位置 1 并清零 CSRC 位。将与 RX/DT 和 TX/CK I/O 引脚对应的 TRIS 位置 1。
2. 清零 CREN 和 SREN 位。
3. 如果使用中断，应确保 INTCON 寄存器的 GIE 和 PEIE 位置 1 并将 TXIE 位置 1。
4. 如果需要 9 位发送，将 TX9 位置 1。
5. 将 TXEN 位置 1 使能发送。
6. 如果选择了9位发送，将最高有效位写入TX9D位。
7. 将低 8 位写入 TXREG 寄存器，启动发送。

表 16-9：　　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	RABIE	TMR0IF	INT0IF	RABIF	285
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	288
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	288
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	288
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	287
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	288
TXREG	EUSART 发送寄存器								287
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	287
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	287
SPBRGH	EUSART 波特率发生器寄存器的高字节								287
SPBRG	EUSART 波特率发生器寄存器的低字节								287

图注：　— = 未实现，读为 0。同步从发送不使用阴影单元。

16.4.2.3　EUSART 同步从接收

除下列各项外，同步主模式和从模式的工作原理是相同的（**第 16.4.1.6 节 “ 同步主接收 ”**）：

- 休眠
- CREN 位始终置 1，因此接收器从不空闲
- SREN 位在从模式下为 “ 无关位 ”

进入休眠前将 CREN 位置 1，可在休眠模式下接收一个字符。接收到该字后， RSR 寄存器将把数据发送到 RCREG 寄存器。如果 RCIE 允许位置 1，产生的中断会将器件从休眠唤醒并执行下一条指令。如果 GIE 位也置 1，程序将跳转到中断向量。

16.4.2.4　同步从接收设置

1. 将 SYNC 和 SPEN 位置 1 并清零 CSRC 位。将与 RX/DT 和 TX/CK I/O 引脚对应的 TRIS 位置 1。
2. 如果使用中断，应确保将 INTCON 寄存器的 GIE 和 PEIE 位置 1 并将 RCIE 位置 1。
3. 如果需要接收 9 位数据，将 RX9 位置 1。
4. 将 CREN 位置 1 使能接收。
5. 接收完成时 RCIF 位将被置 1。如果 RCIE 位已置 1，则产生中断。
6. 如果使能了 9 位模式，从 RCSTA 寄存器的 RX9D 位取出最高有效位。
7. 读取 RCREG 寄存器，从接收 FIFO 取出低 8 位。
8. 如果发生了溢出错误，可通过清零 RCSTA 寄存器的 CREN 位或清零可将 EUSART 复位的 SPEN 位清除错误。

表 16-10：　　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	RABIE	TMR0IF	INT0IF	RABIF	285
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	288
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	288
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	288
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	287
RCREG	EUSART 接收寄存器								287
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	287
BAUDCON	ABDOVF	RCIDL	DTRXP	CKTXP	BRG16	—	WUE	ABDEN	287
SPBRGH	EUSART 波特率发生器寄存器的高字节								287
SPBRG	EUSART 波特率发生器寄存器的低字节								287

图注：　— = 未实现，读为 0。同步从接收不使用阴影单元。

注：

17.0 模数转换器 （ADC）模块

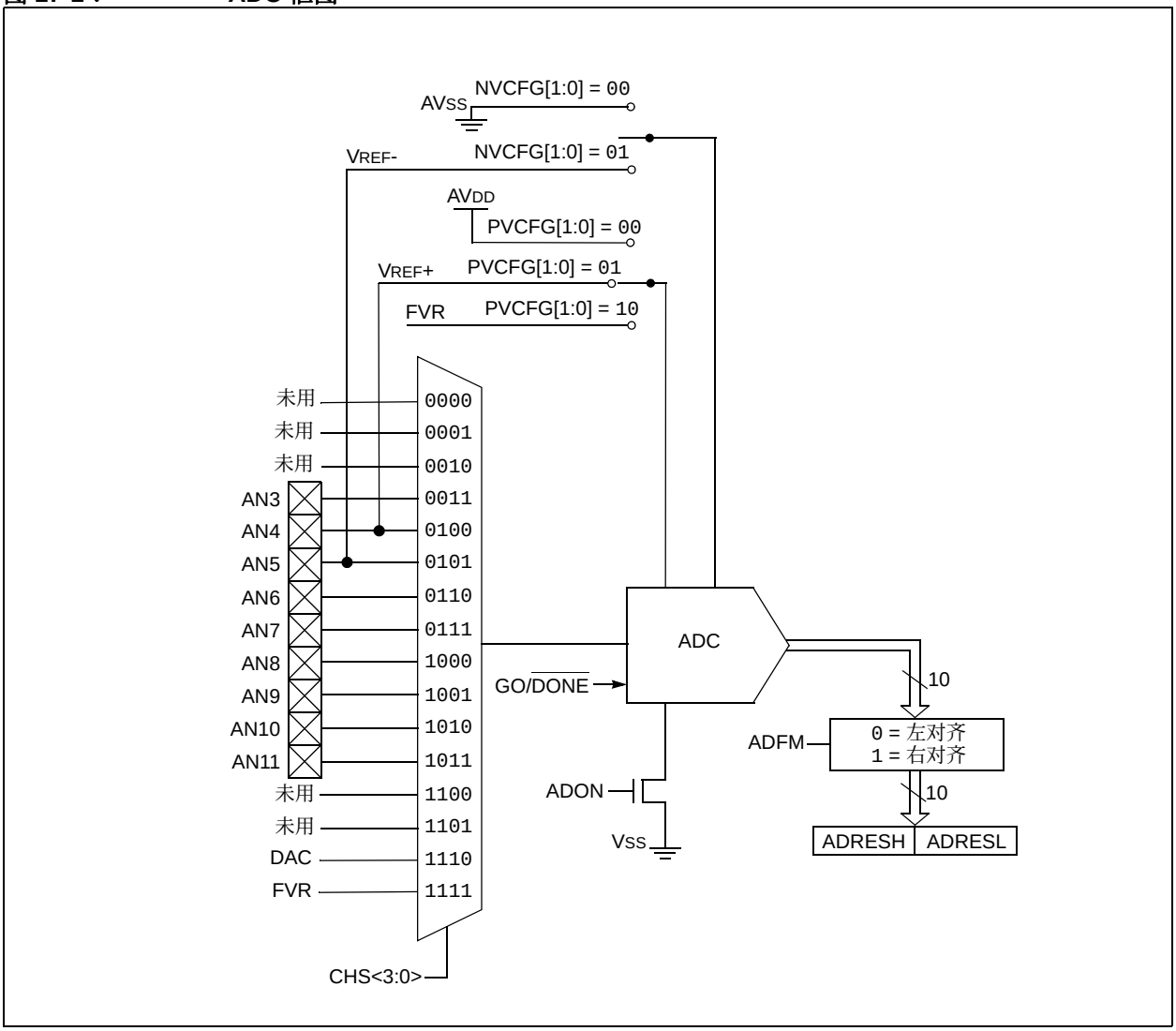
模数转换器（Analog-to-digital Converter，ADC）可将模拟输入信号转换为信号的 10 位二进制表示。该模块使用模拟输入，这些输入通过多路开关连接到同一个采样和保持电路。采样保持电路的输出与转换器的输入相连接。转换器通过逐次逼近法产生 10 位二进制结果，并将转换结果存储在 ADC 结果寄存器 （ADRESL 和 ADRESH）中。

可通过软件方式选择 VDD 或施加在外部参考引脚上的电压作为 ADC 参考电压。

ADC 可在转换完成时产生中断。该中断可用于将器件从休眠状态唤醒。

图 17-1 给出了 ADC 的框图。

图 17-1： ADC 框图



17.1 ADC 配置

配置和使用 ADC 时必须考虑以下功能：

- 端口配置
- 通道选择
- ADC 参考电压选择
- ADC 转换时钟源
- 中断控制
- 结果格式

17.1.1 端口配置

ANSEL、ANSELH、TRISA、TRISB 和 TRISE 寄存器均可用于配置 A/D 端口引脚。任何需要用作模拟输入的端口引脚都应将其对应的 ANSx 位置 1 以禁止数字输入缓冲区，并将 TRISx 位置 1 以禁止数字输出驱动器。如果 TRISx 位清零，则数字输出电平 （VOH 或 VOL）将被转换。

A/D 转换操作与 ANSx 位和 TRIS 位的状态无关。

注	**1：** 当读取端口寄存器时，对应 ANSx 位置 1 的所有引脚均读为零 （低电平）。但是，对于配置为数字输入的引脚 （ANSx 位清零， TRISx 位置 1），将可以正确进行模拟转换。
	2: 对应 ANSx 位清零的任何引脚上的模拟电平可能会导致数字输入缓冲区消耗的电流超出器件规范。

17.1.2 通道选择

ADCON0 寄存器的 CHS 位决定与采样和保持电路相连接的通道。

改变通道时，在开始下一次转换前需要一段延时。更多信息，请参见**第 17.2 节 “ ADC 工作原理 ”**。

17.1.3 ADC 参考电压

ADCON1 寄存器的 PVCFG 和 NVCFG 位分别提供对正负参考电压的独立控制。正参考电压可以是 VDD、FVR 或外部电压源。负参考电压可以是 VSS 或外部电压源。

17.1.4 选择和配置采集时间

ADCON2 寄存器允许用户选择采集时间，该时间在每当 GO/DONE 位置 1 时发生。

采集时间使用 ADCON2 寄存器的 ACQT<2:0> 位设置。采集延时的范围为 2 至 20 个 TAD。当 GO/DONE 位置 1 时， A/D 模块继续对输入进行采样，采样时间为所选择的采集时间，然后自动启动转换。由于采集时间已被编程，因此在选择通道和 GO/DONE 位置 1 之间无需另外等待一个采集时间。

如果 ACQT<2:0> = 000，则表示选择手动采集。当 GO/DONE 位置 1 时，采样停止并启动转换。用户应确保在选定所需要的输入通道之后到 GO/DONE 位置 1 之间经过了所需要的采集时间。此选项也是 ACQT<2:0> 位的默认复位状态，并且与不提供可编程采集时间的器件相兼容。

在这两种情况下，当转换完成时，GO/$\overline{\text{DONE}}$位被清零，ADIF 标志位均被置 1 且 A/D 再次开始对当前选定的通道进行采样。当采集时间被编程时，不会有关于何时采集时间结束、转换开始的指示。

17.1.5 转换时钟

可通过软件方式设置 ADCON2 寄存器的 ADCS 位来选择转换时钟源。有以下 7 种时钟频率可供选择：

- FOSC/2
- FOSC/4
- FOSC/8
- FOSC/16
- FOSC/32
- FOSC/64
- FRC （专用内部振荡器）

完成一个位转换所需的时间定义为 TAD。一次完整的 10 位转换需要 11 个 TAD 周期，如图 17-3 中所示。

为正确转换，必须满足适当的 TAD 规范。更多信息，请参见表 27-9 中的 A/D 转换要求。表 17-1 给出了适当的 ADC 时钟选择的示例。

注：	除非使用 FRC，否则系统时钟频率的任何改变都会改变 ADC 时钟频率，从而对 ADC 结果产生不利影响。

17.1.6 中断

ADC 模块可在模数转换完成时产生中断。 ADC 中断标志位是 PIR1 寄存器中的 ADIF 位。ADC 中断允许位是 PIE1 寄存器中的 ADIE 位。 ADIF 位必须用软件清零。

注： ADIF 位在每次转换完成时置 1，与是否允许了 ADC 中断无关。

器件运行或休眠时都可产生该中断。如果器件处于休眠状态，该中断会唤醒器件。从休眠状态唤醒时，总是执行 SLEEP 指令后紧跟的下一条指令。如果用户试图从休眠状态唤醒器件并恢复顺序代码执行，必须禁止全局中断。如果允许了全局中断，执行将切换到中断服务程序。更多信息，请参见**第 17.1.6 节 “ 中断 ”**。

表 17-1： ADC 时钟周期 （TAD）与器件工作频率关系表

ADC 时钟周期 （TAD）		器件频率 （FOSC）			
ADC 时钟源	**ADCS<2:0>**	**48 MHz**	**16 MHz**	**4 MHz**	**1 MHz**
FOSC/2	000	41.67 ns(2)	125 ns(2)	500 ns(2)	2.0 µs
FOSC/4	100	83.33 ns(2)	250 ns(2)	1.0 µs	4.0 µs
FOSC/8	001	167 ns(2)	500 ns(2)	2.0 µs	8.0 µs(3)
FOSC/16	101	333 ns(2)	1.0 µs	4.0 µs	16.0 µs(3)
FOSC/32	010	667 ns(2)	2.0 µs	8.0 µs(3)	32.0 µs(3)
FOSC/64	110	1.33 µs	4.0 µs	16.0 µs(3)	64.0 µs(3)
FRC	x11	1-4 µs(1,4)	1-4 µs(1,4)	1-4 µs(1,4)	1-4 µs(1,4)

图 注： 阴影单元表示超出了建议范围。

注 1： FRC 时钟源具有 1.7 µs 的典型 TAD 时间。

2： 这些值均违反了所需的最小 TAD 时间。

3： 为了加快转换速度，建议选用其他时钟源。

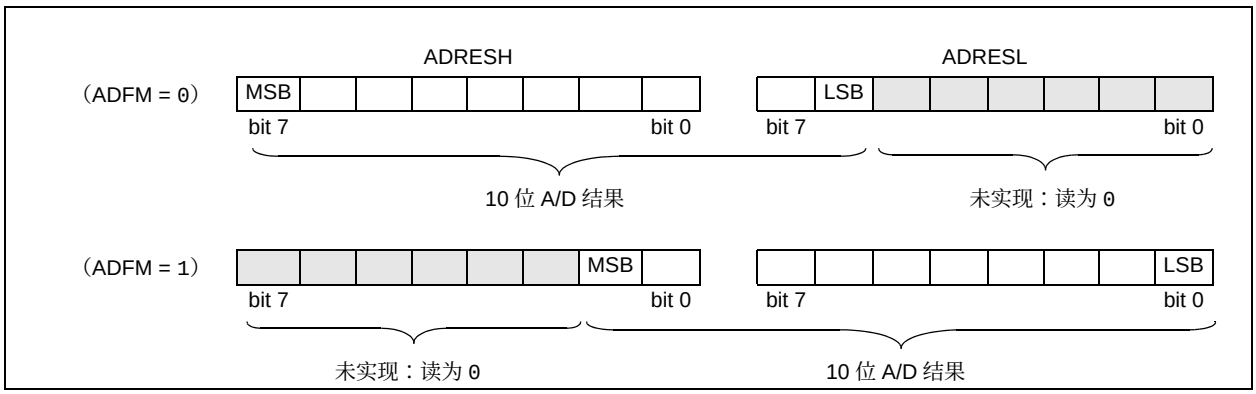
4： 当器件频率高于 1 MHz 时，仅当在休眠状态下进行转换时才推荐使用 FRC 时钟源。

17.1.7 结果格式

10 位 A/D 转换结果可以两种格式提供：左对齐或右对齐。 ADCON2 寄存器的 ADFM 位控制输出格式。

图 17-2 给出了两种输出格式。

图 17-2： 10 位 A/D 转换结果格式



17.2 ADC 工作原理

17.2.1 启动转换

要使能 ADC 模块，ADCON0 寄存器的 ADON 位必须设为 1。将 ADCON0 寄存器的 GO/DONE 位设为 1，根据 ADCON2 寄存器的 ACQT 位的状态，将立即启动模数转换或在模数转换后启动采集延时。

图 17-3 显示了在 GO 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 17-4 显示了在 GO 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。

注： 不应在启动 ADC 的同一条指令中将 GO/DONE 位置 1。请参见**第 17.2.9 节 “ A/D 转换步骤 ”**。

图 17-3： A/D 转换 TAD 周期 （ACQT<2:0> = 000， TACQ = 0）

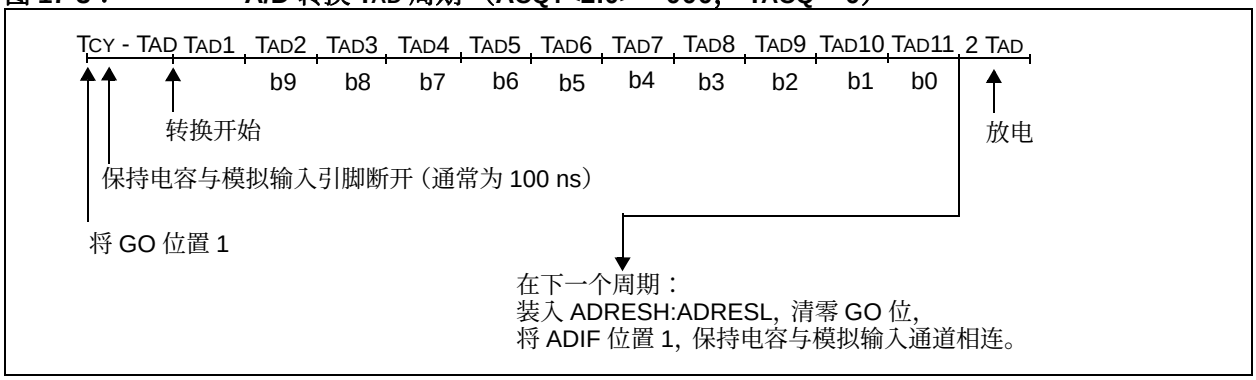


图 17-4： A/D 转换 TAD 周期 （ACQT<2:0> = 010， TACQ = 4 TAD）

17.2.2 转换完成

转换完成时， ADC 模块将：

- 清零 GO/DONE 位
- 将 ADIF 标志位置 1
- 用新的转换结果更新 ADRESH:ADRESL 寄存器

17.2.3 放电

放电过程用于对电容阵列的值进行初始化。在每次采样之后都会对此阵列放电。这一特性有助于优化单位增益放大器，因为每次需要重新为电容阵列充电，而不是根据以前测量的值进行充放电。

17.2.4 终止转换

如果必须在转换完成前终止转换，可用软件将 GO/DONE 位清零。会用部分完成的模数转换结果更新 ADRESH:ADRESL 寄存器。未转换的位将用最后转换的位替代。

注：	器件复位将强制所有寄存器为复位状态。因此， ADC 模块被关闭，任何进行中的转换操作被终止。

17.2.5 转换之间的延时

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。等待该时间之后，当前选定的通道会重新连接到充电保持电容，开始下一次采集。

17.2.6 在功耗管理模式下的 ADC 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进行 A/D 采集转换，就应该根据该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之后，就可以开始 A/D 采集或转换。采集或转换开始以后，器件应继续使用相同的时钟源直到转换完成。

如果需要，器件也可以在转换过程中被置于相应的空闲模式。如果器件时钟频率小于 1 MHz，就应该选择 A/D FRC 时钟源。

17.2.7 休眠期间的 ADC 操作

ADC 模块可以在休眠模式下工作。这需要把 ADC 转换时钟设置为 FRC 选项。选择 FRC 时钟源后， ADC 需等待一个指令周期后才能启动转换操作。这使得可以执行 SLEEP 指令，以降低转换期间的系统噪声。如果允许了 ADC 中断，转换完成时器件将从休眠状态唤醒。如果禁止了 ADC 中断，尽管 ADON 位仍保持置 1，转换完成后 ADC 模块将关闭。

ADC 时钟源不是 FRC 时，尽管 ADON 位仍保持置 1，SLEEP 指令会导致当前转换中止， ADC 模块关闭。

17.2.8 特殊事件触发器

CCP1 特殊事件触发器允许定期测试 ADC 而无需软件干预。当出现触发信号后， GO/DONE 位由硬件置 1，Timer1 或 Timer3 计数器复位为零。

使用特殊事件触发器不能确保正确的 ADC 时序。用户需负责确保 ADC 时序要求得到满足。

更多信息，请参见**第 14.3.4 节 “ 特殊事件触发器 ”**。

17.2.9 A/D 转换步骤

以下是用 ADC 执行模数转换的示例步骤：

1. 配置端口：
   - 禁止引脚输出驱动器 （见 TRIS 寄存器）
   - 将引脚配置为模拟输入
2. 配置 ADC 模块：
   - 选择 ADC 转换时钟
   - 配置参考电压
   - 选择 ADC 输入通道
   - 选择结果格式
   - 选择采集延时
   - 开启 ADC 模块
3. 配置 ADC 中断 （可选）：
   - 清零 ADC 中断标志
   - 允许 ADC 中断
   - 允许外设中断
   - 允许全局中断 **(1)**
4. 等待所需采集时间 **(2)**。
5. 通过将 GO/DONE 位置 1 启动转换。
6. 通过以下方式之一等待 ADC 转换完成：
   - 查询 GO/DONE 位
   - 等待 ADC 中断 （已允许中断）
7. 读取 ADC 结果
8. 清零 ADC 中断标志（如果已允许中断则需要）。

注 1： 如果用户试图从休眠状态唤醒器件并恢复顺序代码执行，必须禁止全局中断。

2: 如果 ACQT 位设置为零延时，则需要软件延时。请参见**第 17.3 节“ A/D 采集要求 ”**。

例 17-1： A/D 转换

```
;This code block configures the ADC
;for polling, Vdd and Vss as reference, Frc
clock and AN4 input.
;
;Conversion start & polling for completion
; are included.
;
MOVLW     B’10101111’  ;right justify, Frc,
MOVWF     ADCON2       ; & 12 TAD ACQ time
MOVLW     B’00000000’  ;ADC ref = Vdd,Vss
MOVWF     ADCON1       ;
BSF       TRISC,0      ;Set RC0 to input
BSF       ANSEL,4      ;Set RC0 to analog
MOVLW     B’00010001’  ;AN4, ADC on
MOVWF     ADCON0       ;
BSF       ADCON0,GO    ;Start conversion
ADCPoll:
BTFSC     ADCON0,GO    ;Is conversion done?
BRA       ADCPoll      ;No, test again
; Result is complete - store 2 MSbits in
; RESULTHI and 8 LSbits in RESULTLO
MOVFF     ADRESH,RESULTHI
MOVFF     ADRESL,RESULTLO
```

17.2.10 ADC 寄存器定义

以下寄存器用于控制 ADC 的操作。

注：	模拟引脚由 ANSEL 和 ANSELH 寄存器控制。关于 ANSEL 和 ANSELH 寄存器的信息，请分别参见寄存器9-15和寄存器9-16。

寄存器 17-1：　　ADCON0：A/D 控制寄存器 0

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/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 7-6　**未实现：**读为 0

bit 5-2　**CHS<3:0>：模拟通道选择位**

0000 = 保留
0001 = 保留
0010 = 保留
0011 = AN3
0100 = AN4
0101 = AN5
0110 = AN6
0111 = AN7
1000 = AN8
1001 = AN9
1010 = AN10
1011 = AN11
1100 = 保留
1101 = 保留
1110 = DAC
1111 = FVR

bit 1　**GO/DONE：**A/D 转换状态位

1 = A/D 转换正在进行。将该位置 1 可启动 A/D 转换周期。
　　A/D 转换完成后，该位由硬件自动清零。
0 = A/D 转换已完成 / 未进行

bit 0　**ADON：**ADC 使能位

1 = 使能 ADC
0 = 禁止 ADC，不消耗工作电流

注　1： 选择保留通道将产生无法预料的结果，因为未实现的输入通道保持悬空状态。

寄存器 17-2：　　ADCON1：A/D 控制寄存器 1

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	PVCFG1	PVCFG0	NVCFG1	NVCFG0
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　**未实现：**读为 0

bit 3-2　**PVCFG<1:0>：**正参考电压选择位
00 = 正参考电压由 VDD 在内部提供。
01 = 正参考电压通过 VREF+ 引脚从外部提供。
10 = 正参考电压通过 FVR 从内部提供。
11 = 保留。

bit 1-0　**NVCFG<1:0>：**负参考电压选择位
00 = 负参考电压由 VSS 在内部提供。
01 = 负参考电压通过 VREF- 引脚从外部提供。
10 = 保留。
11 = 保留。

寄存器 17-3：　　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 转换结果格式选择位
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。采集时间是从 GO/$\overline{\text{DONE}}$ 位置 1 的时刻到转换开始时， A/D 充电保持电容保持连接到 A/D 通道的持续时间。
000 = 0**(1)**
001 = 2 TAD
010 = 4 TAD
011 = 6 TAD
100 = 8 TAD
101 = 12 TAD
110 = 16 TAD
111 = 20 TAD

bit 2-0　　**ADCS<2:0>：**A/D 转换时钟选择位
000 = FOSC/2
001 = FOSC/8
010 = FOSC/32
011 = FRC**(1)**（由专用内部振荡器产生的时钟，其频率的标称值为 600 kHz）
100 = FOSC/4
101 = FOSC/16
110 = FOSC/64
111 = FRC**(1)**（由专用内部振荡器产生的时钟，其频率的标称值为 600 kHz）

注　1： 当选择 FRC 作为 A/D 时钟源时，在 GO/$\overline{\text{DONE}}$ 位置 1 后，转换开始时间会延迟 1 个指令周期，以允许执行 SLEEP 指令。

寄存器 17-4： ADRESH：ADC 结果寄存器的高字节 （ADRESH） ADFM = 0

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2
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 **ADRES<9:2>**：ADC 结果寄存器位
10 位转换结果的高 8 位

寄存器 17-5： ADRESL：ADC 结果寄存器的低字节 （ADRESL） ADFM = 0

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
ADRES1	ADRES0	—	—	—	—	—	—
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 **ADRES<1:0>**：ADC 结果寄存器位
10 位转换结果的低 2 位

bit 5-0 **保留**：不要使用。

寄存器 17-6： ADRESH：ADC 结果寄存器的高字节 （ADRESH） ADFM = 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
—	—	—	—	—	—	ADRES9	ADRES8
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 **保留**：不要使用。

bit 1-0 **ADRES<9:8>**：ADC 结果寄存器位
10 位转换结果的高 2 位

寄存器 17-7： ADRESL：ADC 结果寄存器的低字节 （ADRESL） ADFM = 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
ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2	ADRES1	ADRES0
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 **ADRES<7:0>**：ADC 结果寄存器位
10 位转换结果的低 8 位

17.3 A/D 采集要求

为了使 ADC 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平。模拟输入模型见图 17-5。模拟信号源阻抗（RS）和内部采样开关阻抗（RSS）直接影响电容 CHOLD 的充电时间。采样开关阻抗（RSS）随器件电压（VDD）的变化而变化，请参见图 17-5。**模拟信号源的最大阻抗推荐值为 10 kΩ**。采集时间随着源阻抗的降低而缩短。在选择（或改变）模拟输入通道后，必须在开始转换前完成 A/D 采集。可以使用公式 17-1 来计算最小采集时间。该公式假设误差为 1/2 LSb（ADC 转换需要 1024 步）。1/2 LSb 误差是 ADC 达到规定分辨率所允许的最大误差。

公式 17-1： 采集时间示例

假设： 温度 = 50°C，外部阻抗10kΩ， 3.0 V VDD

放大器稳定时间 + 保持电容充电时间 + 温度系数

$= T_{AMP} + T_C + T_{COFF}$

$= 5\mu s + T_C + [(\text{温度} - 25°C)(0.05\mu s/°C)]$

TC 值可以用以下公式近似计算：

$$V_{APPLIED}\left(1 - \frac{1}{2047}\right) = V_{CHOLD} \quad ;[1] \text{ 充电到 } V_{CHOLD} \text{（1/2 LSb 误差范围）}$$

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{CHOLD} \quad ;[2] \text{ 响应 } V_{APPLIED} \text{ 充电到 } V_{CHOLD}$$

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{APPLIED}\left(1 - \frac{1}{2047}\right) \quad ; \text{结合 [1] 和 [2]}$$

求解 TC：

$T_C = -C_{HOLD}(R_{IC} + R_{SS} + R_S)\ ln(1/2047)$

$= -13.5pF(1k\Omega + 700\Omega + 10k\Omega)\ ln(0.0004885)$

$= 1.20\mu s$

因此：

$T_{ACQ} = 5\mu s + 1.20\mu s + [(50°C - 25°C)(0.05\mu s/°C)]$

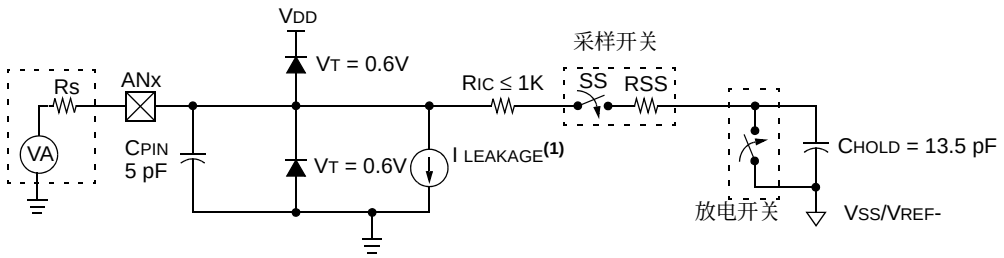
$= 7.45\mu s$

注 1：因为参考电压（VREF）自行抵消，因此它对该公式没有影响。

2：充电保持电容（CHOLD）在每次转换后放电。

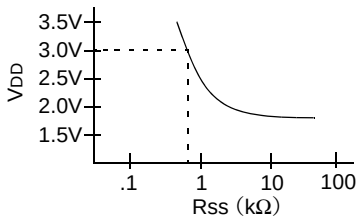
3: 模拟信号源的最大阻抗推荐值为 10 kΩ。此要求是为了符合引脚漏电流规范。

图 17-5：　　模拟输入模型



图注：

CPIN	=	输入电容
VT	=	门限电压
I LEAKAGE	=	由各连接点在引脚上产生的泄漏电流
RIC	=	片内走线等效电阻
SS	=	采样开关
CHOLD	=	采样 / 保持电容



注　1： 请参见**第 27.0 节 “ 电气规范 ”**。

图 17-6：　　ADC 传递函数

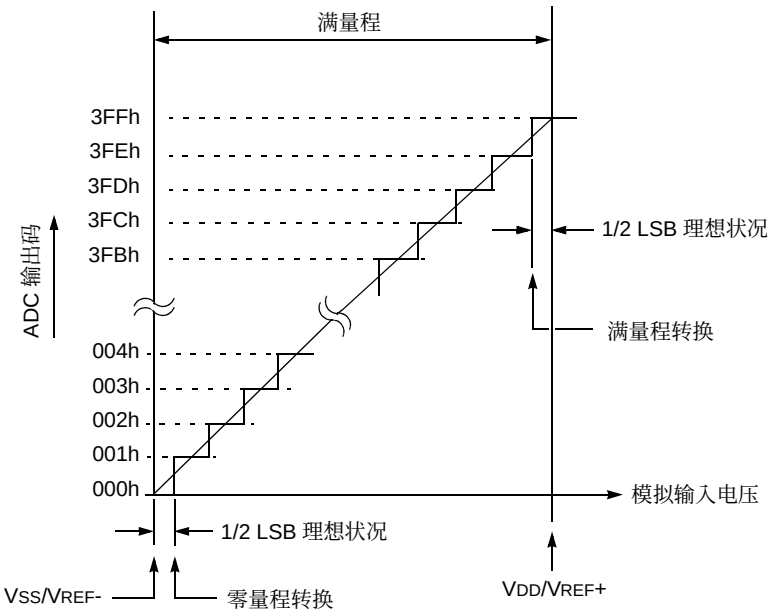


表 17-2：　　与 A/D 操作相关的寄存器

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	RABIE	TMR0IF	INT0IF	RABIF	285
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	288
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	288
IPR1	—	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	288
ADRESH	A/D 结果寄存器的高字节								287
ADRESL	A/D 结果寄存器的低字节								287
ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON	287
ADCON1	—	—	—	—	PVCFG1	PVCFG0	NVCFG1	NVCFG0	287
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	287
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	—	—	—	288
ANSELH	—	—	—	—	ANS11	ANS10	ANS9	ANS8	288
TRISA	—	—	TRISA5	TRISA4	—	—	—	—	288
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	—	—	—	—	288
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	288

图注：　— = 未实现，读为 0。 A/D 转换不使用阴影单元。

注：

18.0 比较器模块

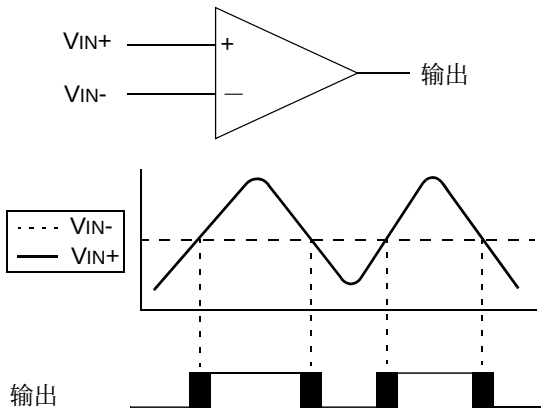
比较器模块通过比较两个模拟电压并提供其相对幅值的数字表示，用于建立模拟电路与数字电路的接口。比较器是非常有用的混合信号模块，因为它提供了与程序执行相独立的模拟功能。模拟比较器模块包含以下特性：

- 独立的比较器控制
- 可编程输入选择
- 有内部 / 外部比较器输出
- 可编程输出极性
- 电平变化中断
- 从休眠状态唤醒
- 可编程的速度 / 功耗优化
- PWM 关闭
- 可编程和固定参考电压

18.1 比较器概述

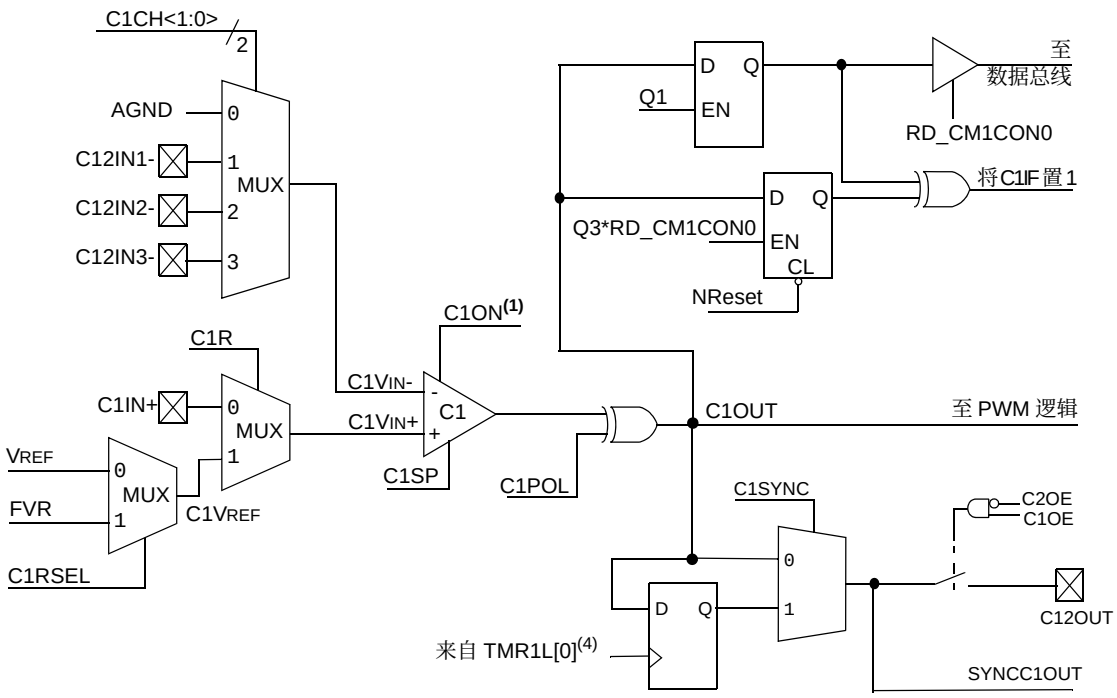
图18-1所示为单比较器以及模拟输入电平与数字输出之间的关系。当 VIN+ 上的模拟电压小于 VIN- 上的模拟电压值时，比较器输出为数字低电平。当 VIN+ 上的模拟电压大于 VIN- 上的模拟电压值时，比较器输出为数字高电平。

图 18-1： 单比较器



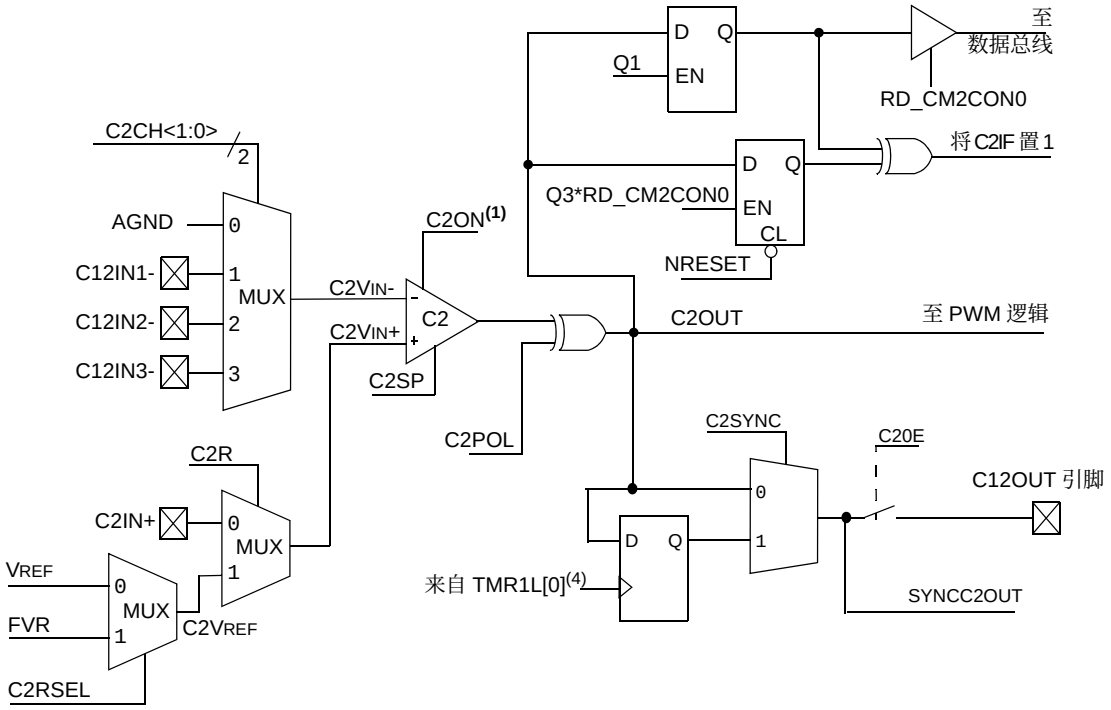
注： 比较器输出的黑色区域表示因输入失调电压和响应时间所造成的输出不确定区域。

图 18-2：　　比较器 C1 的简化框图



注　　1： 当 C1ON = 0 时， C1 比较器将输出 0 到异或门。
2： Q1 和 Q3 是四拍系统时钟 （Fosc）的两拍。
3： Q1 在休眠模式下保持高电平。
4： 该位的上升沿和下降沿上产生的正向脉冲。

图 18-3：　　比较器 C2 的简化框图



注　**1：** 当 C2ON = 0 时， C2 比较器将输出 0 到异或门。
2： Q1 和 Q3 是四拍系统时钟 （FOSC）的两拍。
3： Q1 在休眠模式下保持高电平。
4： 该位的上升沿和下降沿上产生的正向脉冲。

18.2 比较器控制

每个比较器都有独立的控制和配置寄存器：比较器 C1 为 CM1CON0，比较器 C2 为 CM2CON0。此外，比较器 C2 还有另外一个控制寄存器 CM2CON1，用于控制与 Timer1 的交互和两个比较器输出的同时读取。

CM1CON0 和 CM2CON0 寄存器（分别见寄存器 18-1 和 18-2）包含以下控制和状态位：

- 使能
- 输入选择
- 参考电压选择
- 输出选择
- 输出极性
- 速度选择

18.2.1 比较器使能

将 CMxCON0 寄存器的 CxON 位置 1 可以使能比较器操作。清零 CxON 位可以禁止比较器，以使电流消耗降至最低。

18.2.2 比较器输入选择

CMxCON0 寄存器的 CxCH<1:0> 位指示 4 个模拟输入引脚之一连接到比较器的反相输入。

注：	要将CxIN+和C12INx-引脚用作模拟输入，必须将 ANSEL 寄存器中相应的位置 1，同时也必须置1相应的TRIS位来禁止输出驱动器。

18.2.3 比较器参考电压选择

通过将 CMxCON0 寄存器的 CxR 位置 1，将内部参考电压或模拟输入引脚连接到比较器的同相输入。关于内部参考电压模块的更多信息，请参见**第 21.0 节 “ 参考电压 ”**。

18.2.4 比较器输出选择

可以通过读 CMxCON0 寄存器的 CxOUT 位或 CM2CON1 寄存器的 MCxOUT 位监视比较器的输出。为了使输出可用于外部连接，必须满足以下条件：

- 必须将 CMxCON0 寄存器的 CxOE 位置 1
- 必须清零相应的 TRIS 位
- 必须将 CMxCON0 寄存器的 CxON 位置 1

两个比较器共用相同的输出引脚 （C12OUT）。优先级由 C1OE 和 C2OE 位的状态决定。

表 18-1： 比较器输出优先级

C10E	C2OE	C12OUT
0	0	I/O
0	1	C2OUT
1	0	C1OUT
1	1	C2OUT

注	**1：** CxOE 位改写端口数据锁存器。将 CxON 置 1 对端口改写没有影响。 **2:** 比较器的内部输出在每个指令周期被锁存。除非另外指定，否则不锁存外部输出。

18.2.5 比较器输出极性

反相比较器的输出在功能上等同于交换比较器输入。可以通过将 CMxCON0 寄存器的 CxPOL 位置 1 来使比较器输出的极性反相。清零 CxPOL 位导致不反相输出。

表 18-2 给出了输出状态与输入条件的关系 （包括极性控制）。

表 18-2： 比较器输出状态与输入条件

输入条件	CxPOL	CxOUT
CxVIN- > CxVIN+	0	0
CxVIN- < CxVIN+	0	1
CxVIN- > CxVIN+	1	1
CxVIN- < CxVIN+	1	0

18.2.6 比较器速度选择

速度与功耗之间的折衷可以在程序执行期间通过 CxSP 控制位进行优化。该位的默认状态为 1，选择正常速度模式。器件功耗可以通过将 CxSP 位清零进行优化，代价是比较器传输延时变长。

18.3 比较器响应时间

在改变输入源或选择新的参考电压后，比较器输出要经过一段不确定状态的时间。这段时间被称为响应时间。比较器的响应时间不同于参考电压的稳定时间。因此，在确定比较器输入改变的总响应时间时，必须考虑这两个时间。更多详细信息，请参见**第 27.0 节 “ 电气规范 ”** 中的比较器和参考电压规范。

18.4 比较器中断操作

只要比较器的输出值发生变化，相应的比较器中断标志位就会置 1。比较器输出值的变化由不匹配电路识别，该电路由两个锁存器和一个异或门组成 （见图 18-2 和图 18-3）。当读 CMxCON0 寄存器时，一个锁存器更新为比较器的输出电平。该锁存器将保持该值，直到下次读 CMxCON0 寄存器，或者发生复位。不匹配电路的另一个锁存器在每个 Q1 系统时钟更新。当比较器的输出变化在 Q1 时钟周期输入到第二个锁存器时，即产生不匹配条件。这时两个不匹配锁存器具有相反的输出电平，由异或门检测并被送到中断电路。不匹配条件持续直到读 CMxCON0 寄存器或者比较器输出返回先前的状态。

注 **1：** 对 CMxCON0 寄存器的写操作也将清除不匹配条件，因为所有写操作的写周期开始都包含一个读操作。

2： 比较器中断的正常工作与 CxOE 的状态无关。

比较器中断是根据不匹配边沿而不是不匹配电平置 1 的。这意味着不需要额外的读取或写入 CMxCON0 寄存器来将不匹配寄存器清零这一步骤，即可将中断标志复位。当不匹配寄存器清零时，比较器输出返回前一个状态将发生中断，否则不会发生中断。

需要用软件保存比较器输出状态的信息 （从 CMxCON0 寄存器或 CM2CON1 寄存器读取），以确定实际发生的变化。请参见图 18-4 和 18-5。

PIR2 寄存器的 CxIF 位是比较器中断标志位。必须用软件将该位清零以将其复位。由于可以对该寄存器写入 1，因此可产生中断。

在中档兼容模式下，必须将 PIE2 寄存器的 CxIE 位以及 INTCON 寄存器的 PEIE 和 GIE 位都置 1 以允许比较器中断。如果上述位中有任何一位被清零，则无法允许中断，尽管中断条件发生时仍会将 PIR2 寄存器的 CxIF 位置 1。

18.4.1 预设不匹配锁存器

比较器不匹配锁存器可以在使能比较器之前预设为所需的状态。当比较器关闭时， CxPOL 位控制 CxOUT 电平。在 CxON 位清零时，将 CxPOL 位设置为所需的 CxOUT 非中断电平。然后，在 CxON 位置 1 的相同指令中配置所需的 CxPOL 电平。因为所有的寄存器写操作都按读 - 修改 - 写的方式执行，所以不匹配锁存器将在指令 “ 读 ” 阶段被清零，实际配置 CxON 和 CxPOL 位将在最后的 “ 写 ” 阶段进行。

图 18-4： 比较器中断时序 （不带 CMxCON0 读取）

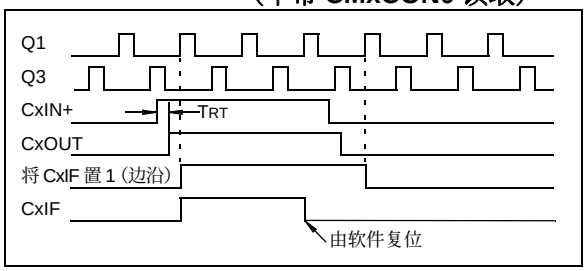
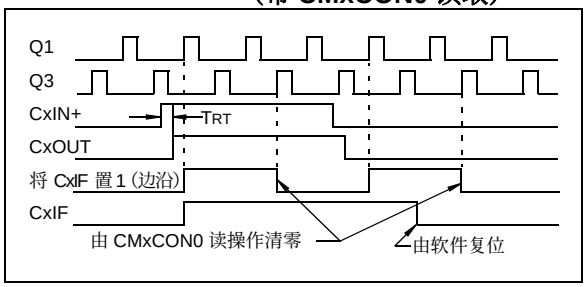


图 18-5： 比较器中断时序 （带 CMxCON0 读取）



注 **1：** 在读操作执行过程中 （Q2 周期的起始时刻），如果 CMxCON0 寄存器 （CxOUT） 发生变化，那么 PIR2 寄存器的 CxIF 中断标志位可能不会被置 1。

2: 当两个比较器之一先使能时，比较器模块中的偏置电路在稳定前可能产生比较器的无效输出。应允许偏置电路有 1 µs 的稳定时间，然后在允许比较器中断前将不匹配条件和中断标志清除。

18.5 休眠期间的操作

如果在进入休眠模式之前比较器使能，则它在休眠期间仍将处于运行状态。**第 27.0 节“电气规范”**中单独给出了比较器消耗的额外电流。如果不使用比较器来唤醒器件，则在休眠模式下可通过关闭比较器使功耗降至最低。可以通过清零 CMxCON0 寄存器的 CxON 位来关闭每个比较器。

比较器输出变化可以将器件从休眠状态唤醒。要使比较器能将器件从休眠状态唤醒，必须将 PIE2 寄存器的 CxIE 位和 INTCON 寄存器的 PEIE 位置 1。紧跟 SLEEP 指令后的指令总是在从休眠状态唤醒之后执行。如果也将 INTCON 寄存器的 GIE 位置 1，则器件将执行中断服务程序。

18.6 复位的影响

器件复位强制 CMxCON0 和 CM2CON1 寄存器为复位状态。这使比较器和参考电压被强制为“关闭”状态。

寄存器 18-1： CM1CON0：比较器 1 的控制寄存器 0

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
C1ON	C1OUT	C1OE	C1POL	C1SP	C1R	C1CH1	C1CH0
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 **C1ON：**比较器 C1 使能位
1 = 使能比较器 C1
0 = 禁止比较器 C1

bit 6 **C1OUT：**比较器 C1 输出位
如果 C1POL = 1（极性反相）：
当 C1VIN+ > C1VIN- 时， C1OUT = 0
当 C1VIN+ < C1VIN- 时， C1OUT = 1
如果 C1POL = 0（极性不反相）：
当 C1VIN+ > C1VIN- 时， C1OUT = 1
当 C1VIN+ < C1VIN- 时， C1OUT = 0

bit 5 **C1OE：**比较器 C1 输出使能位
如果 C2OE = 0（C2 输出禁止）
0 = C1OUT 仅在内部有效
1 = C1OUT 出现在 C12OUT 引脚上 **(1)**
如果 C2OE = 1（C2 输出使能）
0 = C1OUT 仅在内部有效
1 = C2OUT 出现在 C12OUT 引脚上 **(1)**

bit 4 **C1POL：**比较器 C1 输出极性选择位
1 = C1OUT 逻辑反相
0 = C1OUT 逻辑不反相

bit 3 **C1SP：**比较器 C1 速度 / 功耗选择位
1 = C1 工作在正常功耗、高速模式下
0 = C1 工作在低功耗、低速模式下

bit 2 **C1R：**比较器 C1 参考电压选择位 （同相输入）
1 = C1VIN+ 连接至 C1VREF 输出
0 = C1VIN+ 连接至 C12IN+ 引脚

bit 1-0 **C1CH<1:0>：**比较器 C1 通道选择位
00 = C1VIN- 连接至 AGND
01 = C1 的 C12IN1- 引脚连接至 C1VIN-
10 = C1 的 C12IN2- 引脚连接至 C1VIN-
11 = C1 的 C12IN3- 引脚连接至 C1VIN-

注 1： 比较器输出需要以下 3 个条件：C1OE = 1， C1ON = 1 且相应的端口 TRIS 位 = 0。

寄存器 18-2： CM2CON0：比较器 2 的控制寄存器 0

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
C2ON	C2OUT	C2OE	C2POL	C2SP	C2R	C2CH1	C2CH0
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 **C2ON：**比较器 C2 使能位
1 = 使能比较器 C2
0 = 禁止比较器 C2

bit 6 **C2OUT：**比较器 C2 输出位
如果 C2POL = 1 （极性反相）：
当 C2VIN+ > C2VIN- 时， C2OUT = 0
当 C2VIN+ < C2VIN- 时， C2OUT = 1
如果 C2POL = 0 （极性不反相）：
当 C2VIN+ > C2VIN- 时， C2OUT = 1
当 C2VIN+ < C2VIN- 时， C2OUT = 0

bit 5 **C2OE：**比较器 C2 输出使能位
1 = C2OUT 出现在 C12OUT 引脚上 **(1)**
0 = C2OUT 仅在内部有效

bit 4 **C2POL：**比较器 C2 输出极性选择位
1 = C2OUT 逻辑反相
0 = C2OUT 逻辑不反相

bit 3 **C2SP：**比较器 C2 速度 / 功耗选择位
1 = C2 工作在正常功耗、高速模式下
0 = C2 工作在低功耗、低速模式下

bit 2 **C2R：**比较器 C2 参考电压选择位 （同相输入）
1 = C2VIN+ 连接至 C2VREF
0 = C2VIN+ 连接至 C2IN+ 引脚

bit 1-0 **C2CH<1:0>：**比较器 C2 通道选择位
00 = C1VIN- 连接至 AGND
01 = C2 的 C12IN1- 引脚连接至 C2VIN-
10 = C2 的 C12IN2- 引脚连接至 C2VIN-
11 = C2 的 C12IN3- 引脚连接至 C2VIN-

注 1： 比较器输出需要以下 3 个条件：C2OE = 1， C2ON = 1 且相应的端口 TRIS 位 = 0。

18.7 模拟输入连接注意事项

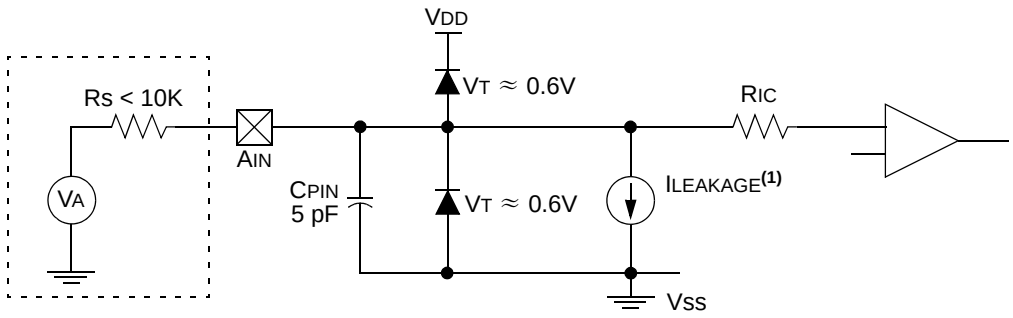
模拟输入的简化电路如图 18-6 所示。由于模拟输入引脚与数字输入共用连接，它们在 VDD 和 VSS 之间连有反向偏置的 ESD 保护二极管。因此，模拟输入必须在 VSS 和 VDD 之间。如果输入电压与这一范围偏离的绝对值超过 0.6V，就可能发生一个二极管正向导通，从而可能导致锁死发生。

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），应保证其泄漏电流极小以使引入的误差降至最低。

注 1： 读端口寄存器时，所有配置为模拟输入的引脚均读为 0。根据输入规范，将把配置为数字输入的引脚作为模拟输入来进行转换。

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 18-6： 模拟输入模型



图注：

CPIN = 输入电容
ILEAKAGE = 由各连接点在引脚上产生的泄漏电流
RIC = 片内走线等效电阻
RS = 信号源阻抗
VA = 模拟电压
VT = 门限电压

注 1： 请参见**第 27.0 节 “ 电气规范 ”**。

18.8 其他比较器特性

共有 4 个额外的比较器特性：

- 比较器输出的同时读取
- 内部参考电压选择
- 滞后选择
- 输出同步

18.8.1 比较器输出的同时读取

CM2CON1 寄存器的 MC1OUT 和 MC2OUT 位是两个比较器输出的镜像副本。从同一个寄存器中同时读取两个输出的功能，可以消除从不同寄存器读取时存在的时序错位。

注	**1：** 通过读CM2CON1获取C1OUT或C2OUT的状态并不影响比较器中断不匹配寄存器。

18.8.2 内部参考电压选择

有两个内部参考电压可供每个比较器的同相输入使用。其中一个是固定参考电压 （FVR），另一个是可变比较器参考电压 （CVREF）。 CM2CON 寄存器的 CxRSEL 位决定其中的哪一个参考电压连到比较器参考电压输出（CXVREF）。进一步连接到比较器通过 CMxCON0 寄存器的 CxR 位实现。更多详细信息，请参见**第 21.1 节 “ 参考电压 ”**、图 18-2 和图 18-3。

18.8.3 比较器滞后

比较器 Cx 具有可选择的滞后。通过将 CM2CON1 寄存器的 CxHYS 位置 1 可以使能滞后。更多详细信息，请参见**第 27.0 节 “ 电气规范 ”**。

18.8.4 将比较器输出与 TIMER1 同步

通过将 CM2CON1 寄存器的 CxSYNC 位置 1，可以使比较器 Cx 的输出与 Timer1 保持同步。使能 Cx 的输出时，Cx的输出在Timer1时钟源的上升沿被锁存。如果 Timer1 使用了预分频器，则比较器的输出在经过预分频后被锁存。为了防止发生竞争，比较器的输出在 Timer1 时钟源的上升沿被锁存，而 Timer1 在其时钟源的上升沿递增。更多信息，请参见比较器框图（图 18-2 和图 18-3）和 Timer1 框图 （图 18-2）。

寄存器 18-3：　　CM2CON1：比较器 2 的控制寄存器 1

R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
MC1OUT	MC2OUT	C1RSEL	C2RSEL	C1HYS	C2HYS	C1SYNC	C2SYNC
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　**MC1OUT**：C1OUT 的镜像副本位

bit 6　**MC2OUT**：C2OUT 的镜像副本位

bit 5　**C1RSEL**：比较器 C1 参考电压选择位
1 = FVR 连接至 C1VREF 输入
0 = CVREF 连接至 C1VREF 输入

bit 4　**C2RSEL**：比较器 C2 参考电压选择位
1 = FVR 连接至 C2VREF 输入
0 = CVREF 连接至 C2VREF 输入

bit 3　**C1HYS**：比较器 C1 滞后使能位
1 = 使能比较器 C1 滞后
0 = 禁止比较器 C1 滞后

bit 2　**C2HYS**：比较器 C2 滞后使能位
1 = 使能比较器 C2 滞后
0 = 禁止比较器 C2 滞后

bit 1　**C1SYNC**：C1 输出同步模式位
1 = C1 输出与 TMR1 时钟的上升沿同步
0 = C1 输出异步

bit 0　**C2SYNC**：C2 输出同步模式位
1 = C2 输出与 TMR1 时钟的上升沿同步
0 = C2 输出异步

表 18-3：　　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
CM1CON0	C1ON	C1OUT	C1OE	C1POL	C1SP	C1R	C1CH1	C1CH0	288
CM2CON0	C2ON	C2OUT	C2OE	C2POL	C2SP	C2R	C2CH1	C2CH0	288
CM2CON1	MC1OUT	MC2OUT	C1RSEL	C2RSEL	C1HYS	C2HYS	C1SYNC	C2SYNC	288
REFCON0	FVR1EN	FVR1ST	FVR1S1	FVR1S0	—	—	—	—	287
REFCON1	D1EN	D1LPS	DAC1OE	---	D1PSS1	D1PSS0	—	D1NSS	287
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	285
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCLIF	USBIF	TMR3IF	—	288
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCLIE	USBIE	TMR3IE	—	288
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCLIP	USBIP	TMR3IP	—	288
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	288
LATC	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0	288
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	288
ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	—	—	—	288

图注：　— = 未实现，读为 0。比较器模块不使用阴影单元。

19.0 功耗管理模式

PIC18F/LF1XK50 器件总共提供 7 种工作模式，可以更有效地进行功耗管理。这些工作模式提供了多种选择，可在资源受限的应用（即，电池供电的设备）中节省功耗。

功耗管理模式有三种类别：

- 运行模式
- 空闲模式
- 休眠模式

这些类别定义了需要为器件的哪些部分提供时钟，有时还需要定义时钟的速度。运行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模块）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®单片机器件提供的节省功耗的功能。其中之一就是时钟切换功能，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® 单片机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。

19.1 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
- 选择何种时钟源

OSCCON 寄存器的 IDLEN 位控制是否为 CPU 提供时钟源，SCS<1:0> 位选择时钟源。表 19-1 总结了各个模式下的位设置、时钟源和受影响的模块。

19.1.1 时钟源

SCS<1:0> 位允许从三个时钟源中任选一个作为功耗管理模式的时钟源。它们是：

- 主时钟，由 FOSC<3:0> 配置位定义
- 辅助时钟 （Timer1 振荡器）
- 内部振荡器模块

19.1.2 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。更多信息，请参见**第 2.8 节 “ 时钟切换 ”**。

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 OSCCON 寄存器的 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的位，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 指令就可实现模式切换。

表 19-1： 功耗管理模式

模式	OSCCON 位		模块时钟		可用时钟和振荡器源
	IDLEN(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	00	提供时钟	提供时钟	主时钟 —— LP、 XT、 HS、 RC、 EC 和内部振荡器模块 (2)。 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	1x	提供时钟	提供时钟	内部振荡器模块 (2)
PRI_IDLE	1	00	关闭	提供时钟	主时钟 ——LP、 XT、 HS、 HSPLL、 RC 和 EC
SEC_IDLE	1	01	关闭	提供时钟	辅助时钟 ——Timer1 振荡器
RC_IDLE	1	1x	关闭	提供时钟	内部振荡器模块 (2)

注 1： IDLEN 反映它在执行 SLEEP 指令时的值。

2： 包含 HFINTOSC 和 HFINTOSC 后分频器以及 LFINTOSC 源。

19.1.3 SLEEP 命令的多种功能

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 OSCCON 寄存器的 IDLEN 位的设置决定。当 IDLEN 位清零时，如果执行 SLEEP 指令，所有时钟将停止，功耗达到最低。当 IDLEN 位置 1 时，如果执行 SLEEP 指令，系统时钟会继续向外设提供时钟，但从 CPU 断开。

19.2 运行模式

在运行模式中，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。

19.2.1 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 2.12 节“ 双速启动模式 ”**），该模式也是器件复位后的默认模式。在此模式下，器件采用由CONFIGH配置寄存器的FOSC位定义的振荡器工作。

19.2.2 SEC_RUN 模式

在 SEC_RUN 模式下， CPU 和外设将辅助外部振荡器作为时钟源。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 OSCCON 寄存器的 SCS<1:0> 位设置为 01 可以进入 SEC_RUN 模式。当工作于 SEC_RUN 模式时，将发生以下所有情况：

- 主时钟源被切换到辅助外部振荡器
- 主外部振荡器被关闭
- T1CON 寄存器的 T1RUN 位被置 1
- OSTS 位被清零

注：	辅助外部振荡器应该在进入 SEC_RUN 模式之前就已经运行了。如果 SCS<1:0> 位设置为 01 时，T1OSCEN 位未置 1，则只有 T1OSCEN 位置 1 且辅助外部振荡器就绪时，才会进入 SEC_RUN 模式。

19.2.3 RC_RUN 模式

在 RC_RUN 模式下，CPU 和外设将内部振荡器作为时钟源。在此模式下，主外部振荡器关闭。当 LFINTOSC 是系统时钟时，RC_RUN 模式是所有运行模式中最节省功耗的模式。

通过将 SCS1 位置 1 可以进入 RC_RUN 模式。当时钟源从主振荡器切换到内部振荡器时，主振荡器被关闭并且 OSTS 位被清零。在任何时候修改 IRCF 位可以立即改变时钟速度。

19.3 休眠模式

PIC18F/LF1XK50 器件的功耗管理休眠模式和所有其他 PIC® 单片机器件提供的传统休眠模式相同。通过清零 OSCCON 寄存器的 IDLEN 位并执行 SLEEP 指令即可进入此模式。这将关闭选定的振荡器（图 19-1）并将所有的时钟源状态位清零。

从运行或空闲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， LFINTOSC 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）准备好之前器件将没有时钟源（见图 19-2），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模块作为时钟源（见**第 24.0 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。这种唤醒不会影响 IDLEN 和 SCS 位。

19.4 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单片机的 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， LFINTOSC 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 没有执行指令，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 会在其准备好执行代码前延时一个 TCSD 间隔。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）。这种唤醒不会影响 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 19-1： 进入休眠模式的转换时序

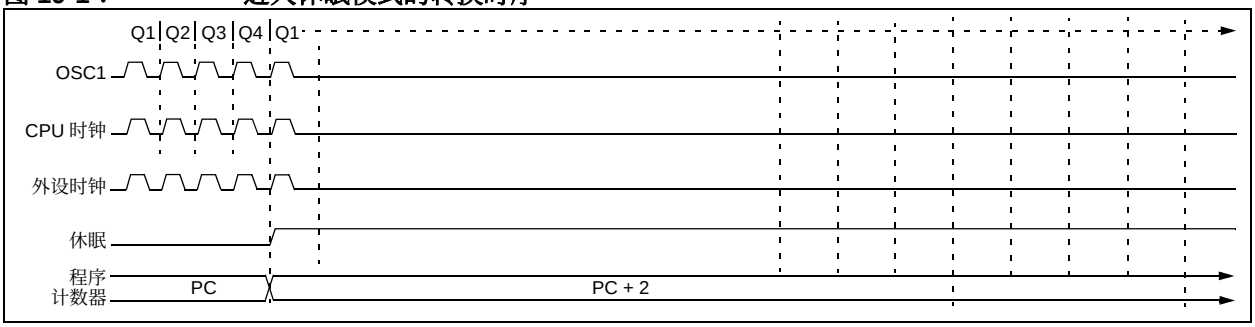
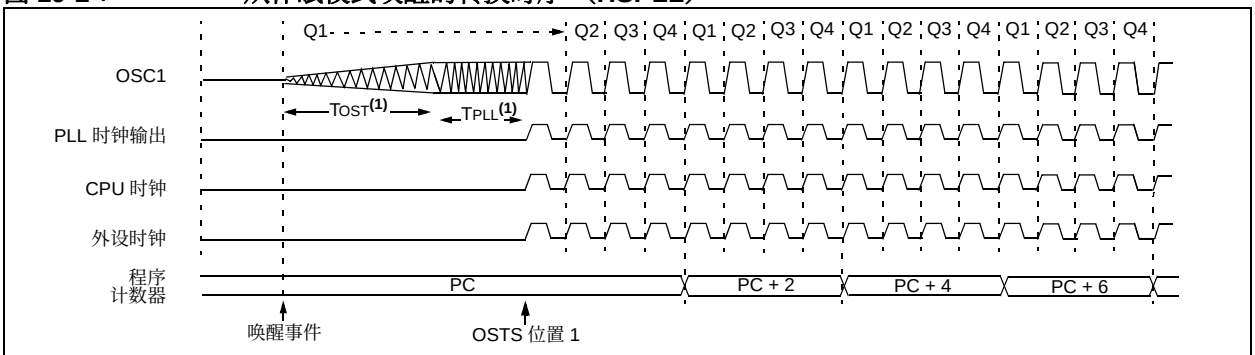


图 19-2： 从休眠模式唤醒的转换时序 （HSPLL）



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

19.4.1 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，选用此模式可以使对时序敏感的应用以最快的速度恢复器件运行，并使用较精确的主时钟源。

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 位清零并执行 SLEEP。虽然 CPU 已被禁止，但外设仍可继续使用由 FOSC<3:0> 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 19-3）。

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 19-4）。

19.4.2 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 位置 1，然后将 SCS<1:0> 位设置为 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 19-4）。

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，那么主系统时钟将继续以先前选定的模式工作，并进入相应的空闲模式 （即， PRI_IDLE 或 RC_IDLE）。

图 19-3： 进入空闲模式的转换时序

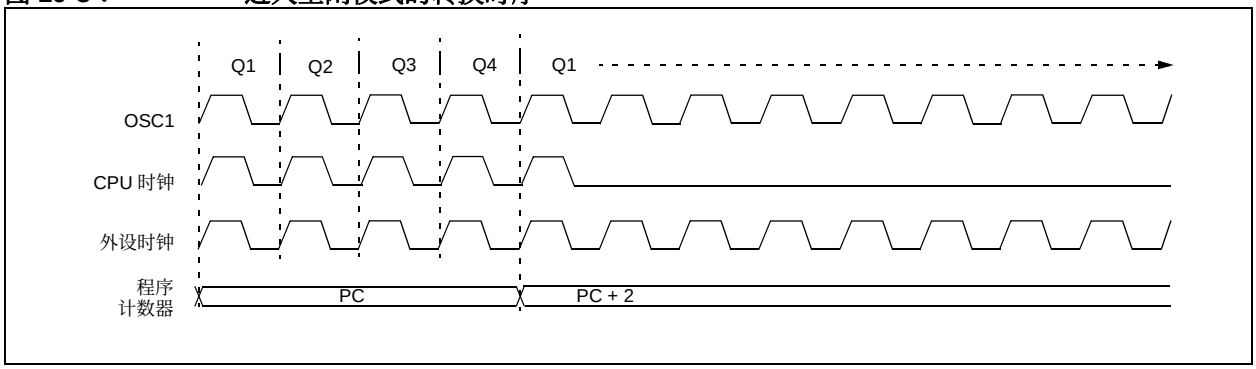
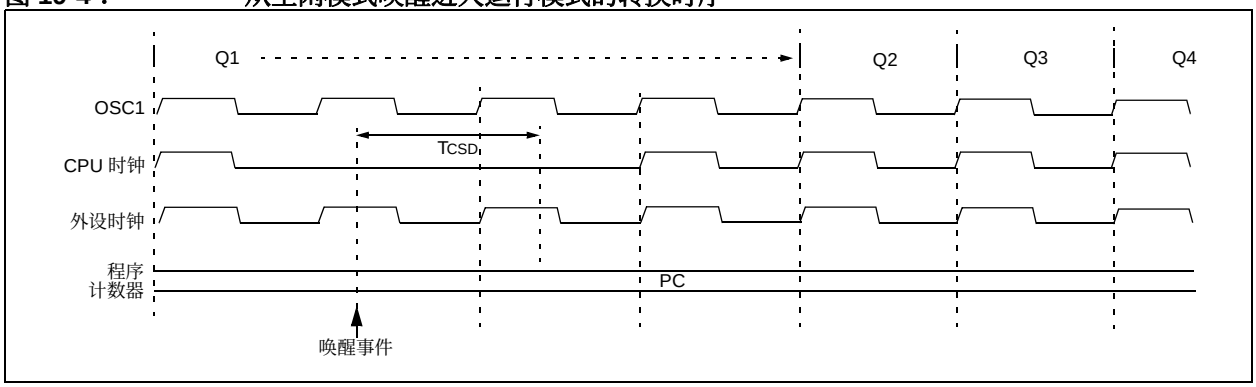


图 19-4： 从空闲模式唤醒进入运行模式的转换时序



19.4.3 RC_IDLE 模式

在RC_IDLE模式下，CPU被禁止，但仍通过HFINTOSC多路开关使用内部振荡器模块为外设提供时钟。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，可以先将 IDLEN 位置 1，然后再将 SCS1 位置 1 并执行 SLEEP。虽然 SCS0 的值常常被忽略，但仍建议将其清零，这将保证与未来器件的软件兼容性。通过在执行 SLEEP 指令之前修改 IRCF 位，可以使用 HFINTOSC 多路开关来选择更高的时钟频率。当时钟源切换到 HFINTOSC 多路开关时，主振荡器被关闭并且 OSTS 位被清零。

如果 IRCF 位被设置为任何非零值，或者 INTSRC 位被置 1，就会使能 HFINTOSC 输出。在一个 TIOBST 间隔之后 HFINTOSC 输出将趋于稳定，随后 IOSF 位置 1。外设的时钟继续运行直到 HFINTOSC 时钟源稳定。如果之前的 IRCF 位为一个非零值或者在执行 SLEEP 指令之前 INTSRC 已置 1，并且当前 HFINTOSC 时钟源已稳定，IOSF 位将保持置 1 状态。如果 IRCF 和 INTSRC 位全部清零，就不会使能 HFINTOSC 输出，IOSF 位将保持清零状态，此时将不会有当前时钟源的任何指示。

当唤醒事件发生时，外设继续将 HFINTOSC 多路开关输出作为时钟源。在唤醒事件后的 TCSD 间隔之后，CPU 开始执行代码并使用HFINTOSC 多路开关作为时钟源。这种唤醒不会影响 IDLEN 和 SCS 位。如果使能了 WDT 或故障保护时钟监视器， LFINTOSC 时钟源将继续运行。

19.5 退出空闲和休眠模式

通过以下任一事件可以触发从休眠模式或任意空闲模式退出：

- 中断
- 复位
- 看门狗定时器超时

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源子系统的工作 （见**第 19.2 节 “ 运行模式 ”**、**第 19.3 节 “ 休眠模式 ”** 和**第 19.4 节 “ 空闲模式 ”**）。

19.5.1 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。如果所需的中断允许位位于 PIE 寄存器中，则还必须将 PEIE 位置 1。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，将执行紧随在 SLEEP 指令后的指令。然后，如果 INTCON 寄存器的 GIE/GIEH 位置 1，代码将跳转到中断向量处执行，否则代码将继续执行，不进行跳转（见**第 7.0 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。

19.5.2 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 19.2 节 “ 运行模式 ”** 和**第 19.3 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 24.2 节 “ 看门狗定时器 （WDT） ”**）。

WDT 定时器和后分频器可由以下任一事件清零：

- 执行 SLEEP 指令
- 执行 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）
- 修改 OSCCON 寄存器中的 IRCF 位 （如果内部振荡器模块是器件时钟源）

19.5.3 通过复位退出

通过复位退出休眠和空闲模式会导致代码从地址 0 重新开始执行。更多详细信息，请参见**第 23.0 节 “ 复位 ”**。

从复位状态退出到开始执行代码期间的延迟时间由唤醒前后的时钟源以及振荡器的类型决定。表 19-2 中总结了退出延时。

19.5.4 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完全不需要 OST 延时。有以下两种情形：

- 主时钟源不停止的 PRI_IDLE 模式
- 主时钟源不是 LP、 XT、 HS 或 HSPLL 中的任意一种模式

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（RC、EC、INTOSC 和 INTOSCIO 模式）。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。

表 19-2： 通过复位从休眠模式或任何空闲模式唤醒的退出延时 （按时钟源分类）

唤醒之前的时钟源	唤醒之后的时钟源	退出延时	时钟就绪状态位 (OSCCON)
主器件时钟 （PRI_IDLE 模式）	LP、 XT 或 HS	TCSD(1)	OSTS
	HSPLL		
	EC 或 RC		
	HFINTOSC(2)		IOSF
T1OSC 或 LFINTOSC(1)	LP、 XT 或 HS	TOST(3)	OSTS
	HSPLL	TOST + t_{PLL}(3)	
	EC 或 RC	TCSD(1)	
	HFINTOSC(1)	TIOBST(4)	IOSF
HFINTOSC(2)	LP、 XT 或 HS	TOST(4)	OSTS
	HSPLL	TOST + t_{PLL}(3)	
	EC 或 RC	TCSD(1)	
	HFINTOSC(1)	无	IOSF
无 （休眠模式）	LP、 XT 或 HS	TOST(3)	OSTS
	HSPLL	TOST + t_{PLL}(3)	
	EC 或 RC	TCSD(1)	
	HFINTOSC(1)	TIOBST(4)	IOSF

注 1： 当从休眠模式和所有空闲模式唤醒时都需要 TCSD 延时，该延时与所需的其他延时并行 （见**第 19.4 节 “ 空闲模式 ”**）。复位时， HFINTOSC 默认值为 1 MHz。

2： 包括 HFINTOSC 16 MHz 时钟源和后分频器产生的频率。

3： TOST 是振荡器起振定时器的延迟时间。 t_{PLL} 是 PLL 锁定延时定时器的延迟时间 （参数 F12）。

4： 在 HFINTOSC 稳定周期 TIOBST 延时期间，代码继续执行。

20.0 SR 锁存器

模块由单个 SR 锁存器组成，该锁存器具有多个置 1 和复位输入，以及可选择的锁存器输出。 SR 锁存器模块包含以下特性：

- 可编程输入选择
- 有内部 / 外部 SR 锁存器输出
- 可选择的 Q 和 $\overline{Q}$ 输出
- 固件置 1 和复位

20.1 锁存器操作

锁存器是不依赖于时钟源的置 1- 复位锁存器。每个置 1 和复位输入均为高电平有效。锁存器可以由 CxOUT、INT1 引脚或可变时钟置 1 或复位。此外， SRCON0 寄存器的 SRPS 和 SRPR 位可以分别用于置 1 或复位 SR 锁存器。锁存器是复位优先型的，因此，如果置 1 和复位输入同时为高电平，则锁存器将进入复位状态。SRPS 和 SRPR 位都是自复位的，也就是说，对两个位中的任一个位执行一次写操作是完成锁存器置 1 或复位操作的必要条件。

20.2 锁存器输出

SRCON0 寄存器的 SRQEN 和 SRNQEN 位用于控制锁存器输出选择。每次， SR 锁存器只有一个输出可以直接输出到 I/O 引脚。优先级由 SRCON0 寄存器中的 SRQEN 和 SRNQEN 位的状态决定。

表 20-1： SR 锁存器输出控制

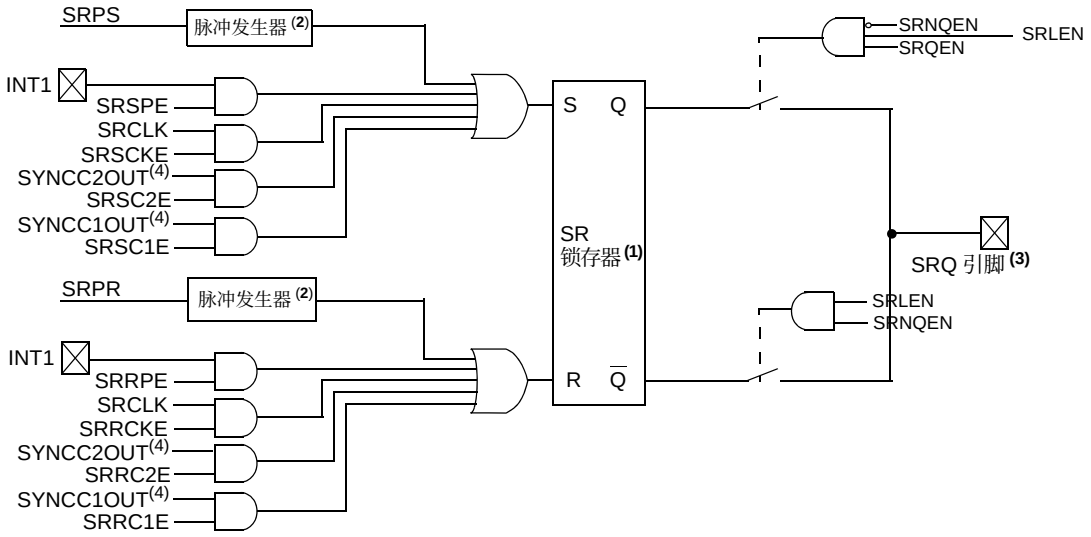
SRLEN	SRQEN	SRNQEN	SR 锁存器输出至端口 I/O
0	X	X	I/O
1	0	0	I/O
1	0	1	$\overline{Q}$
1	1	0	Q
1	1	1	$\overline{Q}$

必须清零相应端口对应的 TRIS 位以使能端口引脚输出驱动器。

20.3 复位的影响

在任何器件复位情况下， SR 锁存器均不被初始化。用户的固件负责在允许锁存器输出传到输出引脚之前初始化锁存器输出。

图 20-1： SR 锁存器简化框图



注 1： 如果 R = 1 且 S = 1，则 Q = 0， $\overline{Q}$ = 1。
2： 脉冲发生器产生 2 个 Q 状态脉冲宽度。
3： 图中的输出仅供参考。更多详细信息，请参见 I/O 端口引脚框图。
4： 名称表示比较器输出的连接源。

表 20-2： SRCLK 频率表

SRCLK	分频比	Fosc = 20 MHz	Fosc = 16 MHz	Fosc = 8 MHz	Fosc = 4 MHz	Fosc = 1 MHz
111	512	25.6 µs	32 µs	64 µs	128 µs	512 µs
110	256	12.8 µs	16 µs	32 µs	64 µs	256 µs
101	128	6.4 µs	8 µs	16 µs	32 µs	128 µs
100	64	3.2 µs	4 µs	8 µs	16 µs	64 µs
011	32	1.6 µs	2 µs	4 µs	8 µs	32 µs
010	16	0.8 µs	1 µs	2 µs	4 µs	16 µs
001	8	0.4 µs	0.5 µs	1 µs	2 µs	8 µs
000	4	0.2 µs	0.25 µs	0.5 µs	1 µs	4 µs

寄存器 20-1： SRCON0：SR 锁存器控制寄存器 0

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SRLEN	SRCLK2	SRCLK1	SRCLK0	SRQEN	SRNQEN	SRPS	SRPR
bit 7							bit 0

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

bit 7 **SRLEN：**SR 锁存器使能位 **(1)**
1 = 使能 SR 锁存器
0 = 禁止 SR 锁存器

bit 6-4 **SRCLK<2:0>**(1)**：**SR 锁存器时钟分频比位
000 = 1/4 外设周期时钟
001 = 1/8 外设周期时钟
010 = 1/16 外设周期时钟
011 = 1/32 外设周期时钟
100 = 1/64 外设周期时钟
101 = 1/128 外设周期时钟
110 = 1/256 外设周期时钟
111 = 1/512 外设周期时钟

bit 3 **SRQEN：**SR 锁存器 Q 输出使能位
如果 SRNQEN = 0
1 = Q 出现在 RC4 引脚
0 = Q 仅在内部出现

bit 2 **SRNQEN：**SR 锁存器 $\overline{Q}$ 输出使能位
1 = $\overline{Q}$ 出现在 RC4 引脚
0 = Q 仅在内部出现

bit 1 **SRPS：**使 SR 锁存器置 1 的脉冲输入
1 = 脉冲输入
0 = 始终读回 0

bit 0 **SRPR：**使 SR 锁存器复位的脉冲输入
1 = 脉冲输入
0 = 始终读回 0

注 1： 在 SR 锁存器使能时更改 SRCLK 位可能导致错误触发锁存器的置 1 和复位输入。

寄存器 20-2： SRCON1：SR 锁存器控制寄存器 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SRSPE	SRSCKE	SRSC2E	SRSC1E	SRRPE	SRRCKE	SRRC2E	SRRC1E
bit 7							bit 0

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

bit 7 **SRSPE：**SR 锁存器外设置 1 使能位
1 = INT1 引脚状态将 SR 锁存器置 1
0 = INT1 引脚状态对 SR 锁存器没有影响

bit 6 **SRSCKE：**SR 锁存器置 1 时钟使能位
1 = SRCLK 为 SR 锁存器的置 1 输入提供脉冲
0 = SRCLK 不为 SR 锁存器的置 1 输入提供脉冲

bit 5 **SRSC2E：**SR 锁存器 C2 置 1 使能位
1 = C2 比较器输出将 SR 锁存器置 1
0 = C2 比较器输出对 SR 锁存器没有影响

bit 4 **SRSC1E：**SR 锁存器 C1 置 1 使能位
1 = C1 比较器输出将 SR 锁存器置 1
0 = C1 比较器输出对 SR 锁存器没有影响

bit 3 **SRRPE：**SR 锁存器外设复位使能位
1 = INT1 引脚将 SR 锁存器复位
0 = INT1 引脚对 SR 锁存器没有影响

bit 2 **SRRCKE：**SR 锁存器复位时钟使能位
1 = SRCLK 为 SR 锁存器的复位输入提供脉冲
0 = SRCLK 不为 SR 锁存器的复位输入提供脉冲

bit 1 **SRRC2E：**SR 锁存器 C2 复位使能位
1 = C2 比较器输出将 SR 锁存器复位
0 = C2 比较器输出对 SR 锁存器没有影响

bit 0 **SRRC1E：**SR 锁存器 C1 复位使能位
1 = C1 比较器输出将 SR 锁存器复位
0 = C1 比较器输出对 SR 锁存器没有影响

表 20-3： 与 SR 锁存器相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	复位值所在页
SRCON0	SRLEN	SRCLK2	SRCLK1	SRCLK0	SRQEN	SRNQEN	SRPS	SRPR	288
SRCON1	SRSPE	SRSCKE	SRSC2E	SRSC1E	SRRPE	SRRCKE	SRRC2E	SRRC1E	288
CM2CON1	MC1OUT	MC2OUT	C1RSEL	C2RSEL	C1HYS	C2HYS	C1SYNC	C2SYNC	288
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	285
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	288

图注： SR 锁存器不使用阴影单元。

注：

21.0 参考电压

有两种独立的参考电压：

- 可编程参考电压
- 1.024V 固定参考电压

21.1 参考电压

参考电压模块为比较器和DAC模块提供了内部产生的参考电压。它具有以下特性：

- 与比较器操作独立
- 单个 32 级电压范围
- 输出钳位到 VSS
- 与 VDD 成比例
- 1.024V 固定参考电压 （FVR）

REFCON1 寄存器（寄存器 21-2）控制参考电压模块，如图 21-1 所示。

公式 21-1： VREF 输出电压

如果 D1EN = 1

$$V_{OUT} = \left((V_{SOURCE+} - V_{SOURCE-}) \times \frac{DAC1R[4:0]}{2^5} + V_{SOURCE-} \right)$$

如果 D1EN = 0 & D1LPS = 1 & DAC1R[4:0] = 11111:

$$V_{OUT} = V_{SOURCE+}$$

如果 D1EN = 0 & D1LPS = 1 & DAC1R[4:0] = 00000:

$$V_{OUT} = V_{SOURCE-}$$

21.1.1 独立操作

参考电压独立于比较器配置。将 REFCON1 寄存器的 D1EN 位置 1，允许电流流入 VREF 分压器，将使能参考电压。当 D1EN 位清零时，禁止电流流入 VREF 分压器，从而将参考电压外设的功耗降至最低。

21.1.2 输出电压选择

VREF 参考电压有 32 个电压值。32 个电压通过 REFCON2 寄存器的 DAC1R<4:0> 位进行设置。

VREF 输出电压由以下公式确定：

21.1.3 输出与 VDD 成比例

比较器参考电压来自 VDD，因此，VREF 输出会随着 VDD 的变化而变化。经过测试的比较器参考电压的绝对精度，请参见**第 27.0 节 “ 电气规范 ”**。

21.1.4 参考电压输出

通过将 REFCON1 寄存器的 DAC1OE 位设为 1，可以将 VREF 参考电压输出到器件 CVREF 引脚。选择输出到 VREF 引脚的参考电压会自动改写数字输出缓冲器和该引脚的数字输入门限值检测器功能。当 CVREF 引脚已被配置为参考电压输出时，读取该引脚始终返回 0。

要提高电流驱动能力，CVREF 参考电压输出端必须外接缓冲器。图 21-2 举例说明了这一缓冲技术。

21.1.5 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，RECON1 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。

21.1.6 复位的影响

器件复位会产生以下影响：

- 禁止参考电压
- 禁止固定参考电压
- VREF 从 CVREF 引脚移除
- DAC1R<4:0> 范围选择位被清零

21.2 FVR 参考电压模块

FVR 参考电压是稳定的固定参考电压，独立于 VDD，输出电压标称值为 1.024V。可以通过将 REFCON0 寄存器的 FVR1EN 位设为 1 使能该参考电压。 FVR 参考电压可以连接至比较器或 ADC 输入通道。

21.2.1 FVR 稳定周期

当固定参考电压模块使能时，需要一段时间使参考电压及其放大电路达到稳定。用户程序必须包含允许模块稳定的延时子程序。REFCON0 寄存器的 FVR1ST 稳定位还用于指示FVR参考电压是否已工作足够长时间达到稳定。关于最小延时要求，请参见**第 27.0 节“ 电气规范 ”**。

图 21-1： 参考电压框图

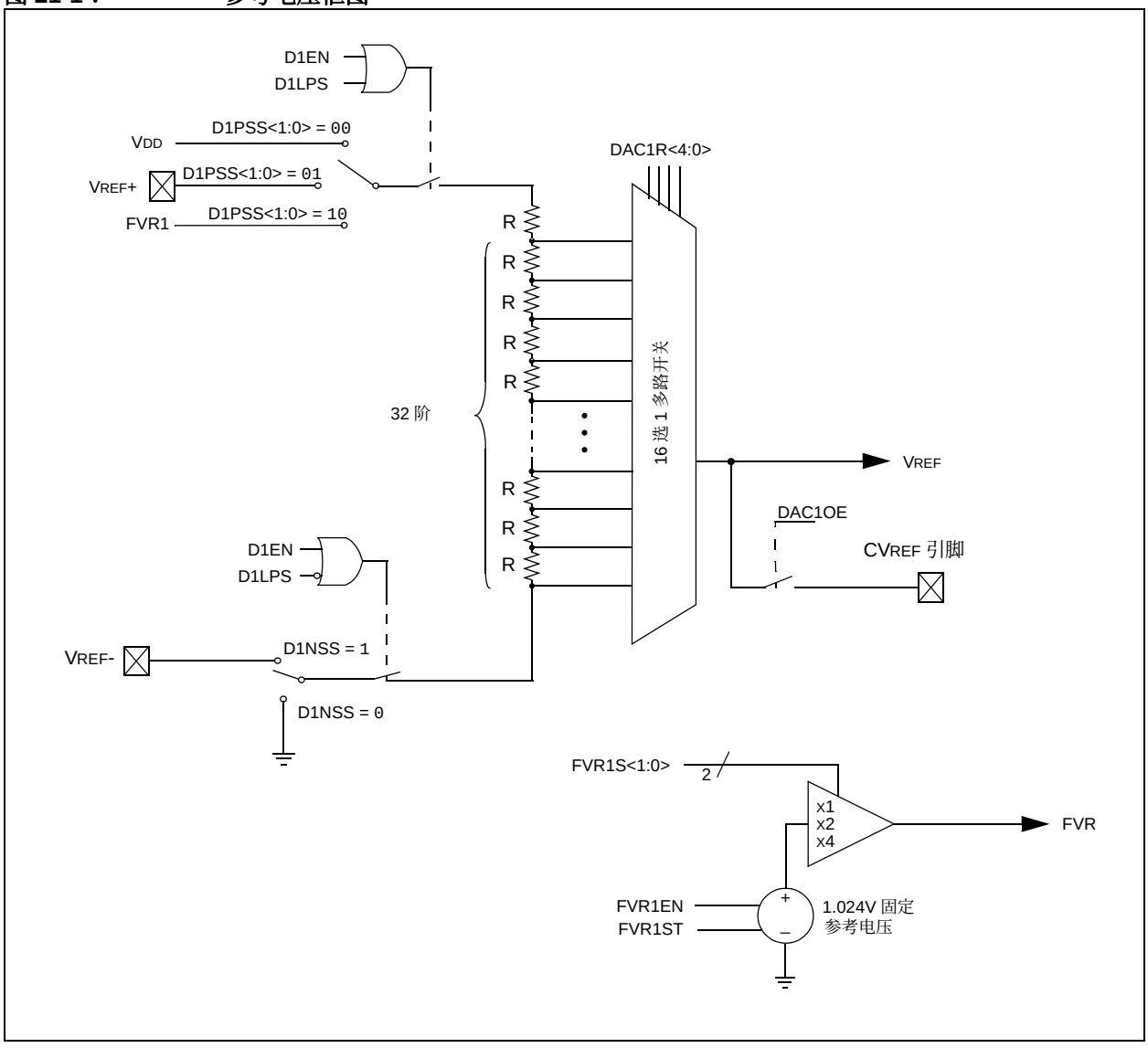
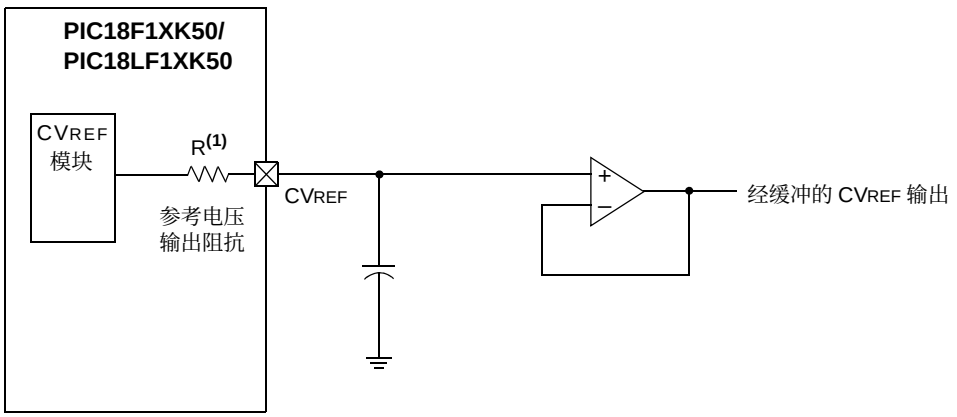


图 21-2：　　参考电压输出缓冲示例



注　1： R 的值取决于参考电压配置位 CVR<3:0> 和 CVRR。

寄存器 21-1：　　REFCON0：参考电压控制寄存器 0

R/W-0	R-0	R/W-0	R/W-1	U-0	U-0	U-0	U-0
FVR1EN	FVR1ST	FVR1S1	FVR1S0	—	—	—	—
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　　**FVR1EN：**固定参考电压 1 使能位
0 =　禁止 FVR
1 =　使能 FVR

bit 6　　**FVR1ST：**固定参考电压 1 稳定位
0 =　FVR 不稳定
1 =　FVR 稳定

bit 5-4　　**FVR1S<1:0>：**固定参考电压 1 电压选择位
00 = 保留，不要使用
01 = 1.024V （x1）
10 = 2.048V （x2）
11 = 4.096V （x4）

bit 3-0　　**未实现：**读为 0

寄存器 21-2： REFCON1：参考电压控制寄存器 1

R/W-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0	U-0	R/W-0
D1EN	D1LPS	DAC1OE	---	D1PSS1	D1PSS0	---	D1NSS
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 **D1EN：**DAC1 使能位
0 = 禁止 DAC1
1 = 使能 DAC1

bit 6 **D1LPS：**DAC1 低功耗电压状态选择位
0 = V_{DAC} = 选择 DAC1 负参考电压源
1 = V_{DAC} = 选择 DAC1 正参考电压源

bit 5 **DAC1OE：**DAC1 电压输出使能位
1 = DAC1 电压也从 RC2/AN6/P1D/C12IN2-/CV_{REF}/INT2 引脚输出
0 = DAC1 电压从 RC2/AN6/P1D/C12IN2-/CV_{REF}/INT2 引脚断开

bit 4 **未实现：**读为 0

bit 3-2 **D1PSS<1:0>：**DAC1 正参考电压源选择位
00 = V_{DD}
01 = V_{REF+}
10 = FVR 输出
11 = 保留，不要使用

bit 1 **未实现：**读为 0

bit 0 **D1NSS：**DAC1 负参考电压源选择位
0 = V_{SS}
1 = V_{REF-}

寄存器 21-3： REFCON2：参考电压控制寄存器 2

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
---	---	---	DAC1R4	DAC1R3	DAC1R2	DAC1R1	DAC1R0
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 4-0 **DAC1R<4:0>：**DAC1 电压输出选择位
V_{OUT} = (($V_{SOURCE+}$) - ($V_{SOURCE-}$))*(DAC1R<4:0>/(2^5)) + $V_{SOURCE-}$

注 1： 输出选择位始终保持右对齐，以确保可以使用任意个位而不影响寄存器布局。

表 21-1：　　与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
REFCON0	FVR1EN	FVR1ST	FVR1S1	FVR1S0	—	—	—	—	287
REFCON1	D1EN	D1LPS	DAC1OE	---	D1PSS1	D1PSS0	—	D1NSS	287
REFCON2	—	—	—	DAC1R4	DAC1R3	DAC1R2	DAC1R1	DAC1R0	287
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	288

图注：　参考电压不使用阴影单元。

注：

22.0 通用串行总线 （USB）

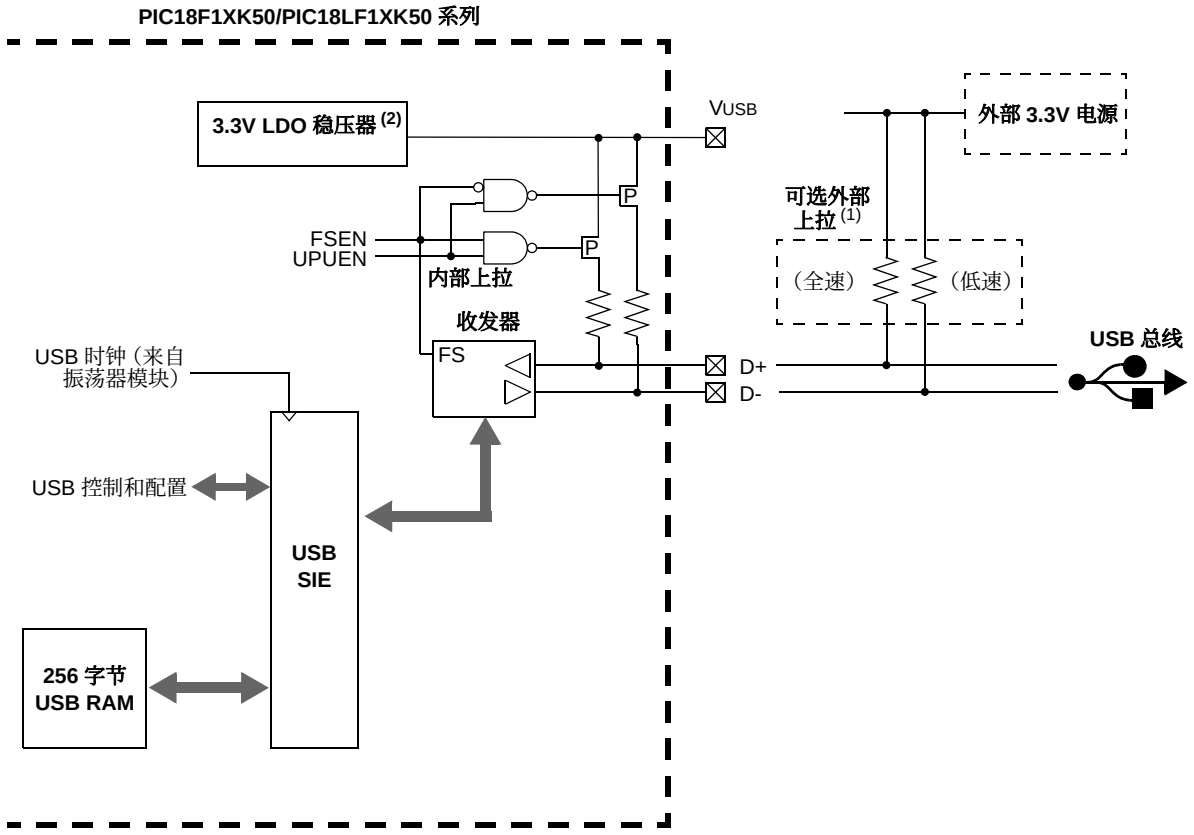
本节将详细介绍 USB 外设。由于本模块的特殊性质，要求读者具备 USB 方面的基础知识。在**第 22.10 节“ USB 概述 ”** 中提供了一些 USB 的高级信息，仅供应用程序设计参考。我们鼓励设计人员参考 USB Implementers Forum （USB-IF）发布的官方规范，以获得最新的信息。本文档发布时， USB 规范 2.0 版是最新的规范。

22.1 USB 外设概述

PIC18F1XK50/PIC18LF1XK50 器件包含全速和低速兼容的 USB 串行接口引擎 （SIE），它允许在 USB 主机和 PIC® 单片机之间实现快速通信。 SIE 可使用内部收发器直接与 USB 接口。

该模块包括了一些特殊硬件功能以便改善性能。提供了器件的数据存储空间 （USB RAM）中的双存取端口存储器，用于单片机内核与 SIE 之间的直接存储器访问。还提供了缓冲区描述符，允许用户自由地设置 USB RAM 空间内对端点存储单元的使用。图 22-1 提供了 USB 外设及其特性的一般概述。

图 22-1： USB 外设和选项



注 1： 如果使用了外部上拉电阻，则应禁止内部上拉电阻 （UPUEN = 0）。

2： 仅适用于 PIC18F13K50/PIC18F14K50。

22.2 USB 状态和控制

USB 模块的操作是通过 3 个控制寄存器配置和管理的。此外，总共使用了 14 个寄存器管理实际 USB 事务。这些寄存器是：

- USB 控制寄存器 （UCON）
- USB 配置寄存器 （UCFG）
- USB 传输状态寄存器 （USTAT）
- USB 器件地址寄存器 （UADDR）
- 帧编号寄存器 （UFRMH:UFRML）
- 端点使能寄存器 0 到 7 （UEPn）

22.2.1 USB 控制寄存器 （UCON）

USB 控制寄存器 （寄存器 22-1）包含控制传输期间的模块行为所需的控制位。寄存器包含控制以下内容的位：

- 主 USB 外设使能
- 乒乓缓冲区指针复位
- 挂起模式的控制
- 数据包传输禁止

此外，USB 控制寄存器还包含状态位 SE0 （UCON<5>），用于指示总线上出现了单端 0。USB 模块使能时，应监视该位，以确定差分数据线是否出自单端 0 条件。这有助于区分初始上电状态和 USB 复位信号。

USB 模块的总体操作是由 USBEN 位 （UCON<3>）控制的。将该位置 1 将激活模块，并将缓冲区描述符表中的所有 PPBI 位复位为 0。该位还将激活内部上拉电阻（如果已使能）。因此，该位可用作 USB 的软连接 / 断开控制。尽管清零该位时，所有状态和控制位都将被忽略，但是还是需要在设置该位前预先配置好 USB 模块。只有为 USB 模块提供有效时钟源之后，才能将该位置 1。如果使用了 PLL，则应先将它使能至少 2 毫秒 （以便有足够的时间让 PLL 锁定），然后再尝试将 USBEN 位置 1。

寄存器 22-1： UCON：USB 控制寄存器

U-0	R/W-0	R-x	R/C-0	R/W-0	R/W-0	R/W-0	U-0
—	PPBRST	SE0	PKTDIS	USBEN(1)	RESUME	SUSPND	—
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 **未实现：**读为 0

bit 6 **PPBRST：**乒乓缓冲区复位位
1 = 将所有乒乓缓冲区指针复位到偶编号缓冲区描述符 （Buffer Descriptor， BD）存储区
0 = 乒乓缓冲区指针不复位

bit 5 **SE0：**有效单端 0 标志位
1 = 在 USB 总线上存在有效的单端 0
0 = 没有检测到单端 0

bit 4 **PKTDIS：**数据包传输禁止位
1 = 禁止处理 SIE 令牌和数据包，收到 SETUP 令牌时自动置 1
0 = 使能 SIE 令牌和数据包处理

bit 3 **USBEN：**USB 模块使能位 (1)
1 = 使能 USB 模块和支持电路 （器件已连接）
0 = 禁止 USB 模块和支持电路 （器件已断开连接）

bit 2 **RESUME：**恢复信号传输使能位
1 = 激活恢复信号传输
0 = 禁止恢复信号传输

bit 1 **SUSPND：**挂起 USB 位
1 = USB 模块和支持电路处于节能模式， SIE 时钟无效
0 = USB 模块和支持电路处于正常工作模式， SIE 时钟按配置的速率运行

bit 0 **未实现**：读为 0

注 1： 如果 USB 模块没有适当的时钟源，则不能将该位置 1。

PPBRST 位（UCON<6>）控制使用双重缓冲模式（乒乓缓冲）时的复位状态。PPBRST 位置 1 时，所有乒乓缓冲区指针都指向偶编号缓冲区。PPBRST 必须由固件清零。该位在不使用乒乓缓冲模式时被忽略。

PKTDIS 位 （UCON<4>）是指示 SIE 已禁止数据包发送和接收的标志。该位在收到允许设置处理的SETUP令牌时由 SIE 置 1。单片机无法将该位置 1，只能清零；将其清零可使 SIE 继续发送和 / 或接收。缓冲区描述符表中的任何挂起事件仍有效，由 USTAT 寄存器的 FIFO 缓冲区指示。

RESUME 位（UCON<2>）允许外设通过执行恢复信号传输执行远程唤醒。为了产生有效的远程唤醒，固件必须将 RESUME 置 1 并保持 10 ms，然后将该位清零。关于“ 恢复信号传输 ”的更多信息，请参见 **“ Universal Serial Bus Specification Revision 2.0 ”**（通用串行总线规范 2.0 版）。

SUSPND 位（UCON<1>）将模块和支持电路置于低功耗模式。SIE 的输入时钟也将被禁止。在响应 IDLEIF 中断时，该位应由软件置 1。它在发生 ACTVIF 中断后应由单片机固件复位。该位有效时，器件保持与总线的连接，但是收发器输出保持空闲。 VUSB 引脚上的电压可能随该位值的不同而变化。在 IDLEIF 请求之前将该位置 1 将导致不可预测的总线行为。

注：	在挂起模式下，典型总线供电的 USB 设备的电流上限是 500 µA。这是 PIC 器件及其支持电路消耗的全部电流。器件进入挂起模式后，要小心确保消耗电流最低。

22.2.2 USB 配置寄存器 （UCFG）

通过 USB 进行通信前，必须配置该模块的相关内部和 / 或外部硬件。多数配置都是通过 UCFG 寄存器 （寄存器 22-2）执行的。 UFCG 寄存器包含大多数控制 USB 模块系统级行为的控制位。这些操作包括：

- 总线速度 （全速与低速）
- 片上上拉电阻使能
- 乒乓缓冲区使用

UTEYE 位（UCFG<7>）用于使能眼图生成，可辅助模块测试、调试和 USB 认证。

注：	只应在模块设置阶段配置 USB 速度、收发器和上拉电路。建议不要在使能模块时改变这些设置。

22.2.2.1 内部收发器

USB 外设具有内置的、USB 2.0 全速和低速收发器，从内部连接到 SIE。该功能部件对于低成本单芯片应用很有用。使能 USB 模式 （USBEN = 1）也将使能内部收发器。FSEN 位（UCFG<2>）控制收发器速度；将该位置 1 将使能全速工作。

片上 USB 上拉电阻通过 UPUEN 位（UCFG<4>）进行控制。只有使能片上收发器时，才能选择它们。

内部 USB 收发器从 VUSB 引脚获取电能。为了满足 USB 信号电平规范，必须为 VUSB 提供介于 3.0V 和 3.6V 之间的电源。使用 3.3V 电源，并使用大容量陶瓷电容进行本地旁路时，可以获得最佳的电气信号质量。电容应尽可能靠近位于封装同一侧的 VUSB 和 VSS 引脚 （即，对于 20 引脚 PDIP、 SOIC、 SSOP 和 QFN 封装的部件，将电容的地连接到 VSS 引脚 20）。

D+ 和 D- 信号线可以直接连接到 USB 连接器或电缆的相应引脚 （对于硬连线应用）。不需要任何其他电阻、电容或磁性元件，因为 D+ 和 D- 驱动器具有受控制的压摆率，以及与 USB 电缆的特性阻抗相匹配的输出阻抗。

为了满足 USB 规范，走线长度应低于 30 厘米。理想情况下，这些走线应设计为特性阻抗与 USB 电缆阻抗相匹配。

寄存器 22-2： UCFG：USB 配置寄存器

R/W-0	U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0
UTEYE	—	—	UPUEN(1)	—	FSEN(1)	PPB1	PPB0
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 **UTEYE：**USB 眼图测试使能位
1 = 使能眼图测试
0 = 禁止眼图测试

bit 6-5 **未实现：**读为 0

bit 4 **UPUEN：**USB 片上上拉使能位 (1)
1 = 使能片上上拉 （FSEN = 1 时， D+ 上拉或 FSEN = 0 时， D- 上拉）
0 = 禁止片上上拉

bit 3 **未实现：**读为 0

bit 2 **FSEN：**全速使能位 (1)
1 = 全速器件：控制收发器边沿速率；要求输入时钟运行在 48 MHz
0 = 低速器件：控制收发器边沿速率；要求输入时钟运行在 6 MHz

bit 1-0 **PPB<1:0>：**乒乓缓冲区配置位
11 = 使能端点 1 到 15 的奇 / 偶乒乓缓冲区
10 = 使能所有端点的奇 / 偶乒乓缓冲区
01 = 使能 OUT 端点 0 的奇 / 偶乒乓缓冲区
00 = 禁止奇 / 偶乒乓缓冲区

注 1： UPUEN 和 FSEN 位在 USB 模块使能期间不要更改。必须在使能模块前预先配置这些值。

22.2.2.2 内部上拉电阻

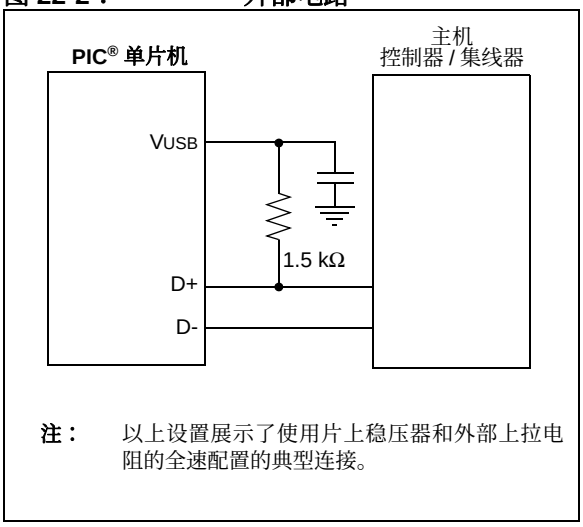
PIC18F1XK50/PIC18LF1XK50 器件具有内置的上拉电阻，设计用来满足低速和全速 USB 的要求。UPUEN 位（UCFG<4>）用于使能内部上拉。图 22-1 给出了上拉及其控制的图示。

注： 官方 USB 规范要求 USB 器件永远不能在 USB 电缆的+5V VBUS线上引入任何电流。此外，每当 +5V VBUS 线的电压低于 1.17V 时，USB 器件永远不能在 D+ 和 D- 数据线上引入任何电流。为了满足这种要求，不是纯粹使用总线供电的应用应当监视VBUS线，并避免在 VBUS 大于 1.17V 之前开启 USB 模块和 D+ 或 D- 上拉电阻。为此，可以将 VBUS 连接到任何可承受 5V 的 I/O 引脚并通过该引脚进行监视。

22.2.2.3 外部上拉电阻

此外，也可以使用外部上拉电阻。可用 VUSB 引脚上拉 D+或D-。按照USB规范的要求，上拉电阻必须是1.5 kΩ（±5%）。图 22-2 给出了一个示例。

图 22-2： 外部电路



注： 以上设置展示了使用片上稳压器和外部上拉电阻的全速配置的典型连接。

22.2.2.4 乒乓缓冲区配置

乒乓缓冲区的使用是通过 PPB<1:0> 位配置的。关于乒乓缓冲区的完整说明，请参见**第 22.4.4 节“ 乒乓缓冲 ”**。

22.2.2.5 使能眼图测试

UCFG<7> 位置 1 时，可通过该模块生成自动的眼图测试。眼图输出的可观察性取决于模块设置，这意味着用户首先要负责配置 SIE 时钟设置、上拉电阻和收发器模式。此外，必须使能该模块。

一旦 UTEYE 被置 1，该模块将仿真从接收切换到发送状态，并开始发送 J-K-J-K 位序列（对于全速是 K-J-K-J）。当使能了眼图测试模式时，该序列将无限重复。

注意，该模块连接到实际 USB 系统时，不应将该位置 1。该测试模式用于电路板检验，并帮助进行 USB 认证测试。它供系统开发者了解 USB 信号的噪声完整性，USB 信号噪声来自于电路板布线、阻抗不匹配和邻近其他系统组件的影响。它不能正确测试从接收到发送状态的转换过程。尽管眼图不能代替更复杂的 USB 认证测试，但它在首轮系统调试中会有帮助。

22.2.3 USB 状态寄存器 （USTAT）

USB 状态寄存器报告 SIE 内的事务状态。SIE 发出 USB 传输完成中断时，应读取 USTAT 以确定传输的状态。USTAT 包含传输端点号、方向和乒乓缓冲区指针值（如果使用了）。

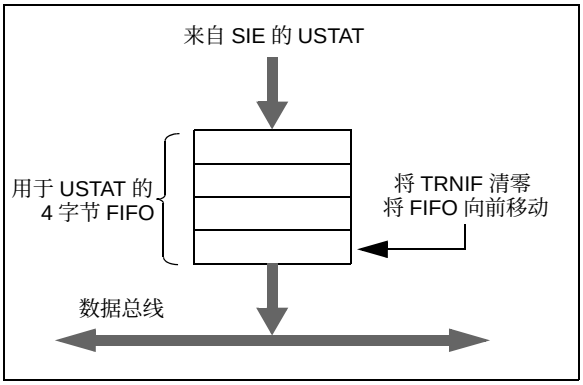
> **注：** 在TRNIF中断标志被拉为低电平之后的两个 SIE 时钟，USB 状态寄存器中的数据有效。
>
> 在系统时钟以48 MHz工作的低速操作中，在接收到 TRNIF 中断和处理 USTAT 寄存器中的数据之间可能需要一定的延时。

USTAT 寄存器实际上是由 SIE 维护的四字节状态 FIFO 的读取窗口。它允许单片机在 SIE 处理其他端点时处理一次传输（图 22-3）。当 SIE 使用缓冲区完成读写数据时，它将更新 USTAT 寄存器。如果在响应事务完成中断前进行了另一次 USB 传输，SIE 将把下次传输的状态存储到状态 FIFO 中。

将传输完成标志位 TRNIF 清零将使 SIE 把 FIFO 向前移动。如果 FIFO 保持寄存器中的下一个数据有效，则 SIE 将在清零 TRNIF 的 6 个 TCY 内将中断标志重新拉为低电平。如果没有其他数据，TRNIF 将保持清零，USTAT 数据将不再可靠。

> **注：** 如果 USTAT FIFO 装满时收到端点请求，SIE 将自动将 NAK 发回主机。

图 22-3： USTAT FIFO



寄存器 22-3： USTAT：USB 状态寄存器

U-0	U-0	R-x	R-x	R-x	R-x	R-x	U-0
—	—	ENDP2	ENDP1	ENDP0	DIR	PPBI(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 7-6 **未实现：**读为 0

bit 5-3 **ENDP<2:0>：**上次活动的端点的编号 （代表上个 USB 传输更新的 BDT 号）

111 = 端点 7
110 = 端点 6
....
001 = 端点 1
000 = 端点 0

bit 2 **DIR：**上次 BD 方向指示位

1 = 上次事务是 IN 令牌
0 = 上次事务是 OUT 或 SETUP 令牌

bit 1 **PPBI：**乒乓 BD 指针指示位 (1)

1 = 上次事务针对奇编号 BD 存储区
0 = 上次事务针对偶编号 BD 存储区

bit 0 **未实现：**读为 0

注 1： 该位仅对具有可用的偶编号和奇编号 BD 寄存器的端点有效。

22.2.4 USB 端点控制

8 个可能的双向端点中的每一个都有各自的独立控制寄存器 UEPn （其中 “ n ” 代表端点号）。每个寄存器内控制位的设置都是相同的。原型如寄存器 22-4 所示。

EPHSHK 位 （UEPn<4>）控制端点的握手；将该位置 1 将使能 USB 握手。通常该位总是置 1 的，除非使用同步端点。

EPCONDIS 位 （UEPn<3>）用于使能或禁止通过端点进行 USB 控制操作 （SETUP） 。将该位清零将使能 SETUP 事务。注意，对应的 EPINEN 和 EPOUTEN 位必须置 1，才能使能 IN 和 OUT 事务。对于端点 0，该位应总是清零的，因为 USB 规范将端点 0 视为默认的控制端点。

EPOUTEN 位 （UEPn<2>）用于使能或禁止主机发起的 USB OUT 事务。将该位置 1 将使能 OUT 事务。类似地， EPINEN 位 （UEPn<1>）使能或禁止主机发起的 USB IN 事务。

EPSTALL 位 （UEPn<0>）用于指示端点的 STALL 状态。如果在特定端点上发出了 STALL， SIE 将把那个端点对的 EPSTALL 位置 1。EPSTALL 位将保持置 1 状态直到固件将其清零，或 SIE 复位。

寄存器 22-4： UEPn：USB 端点 n 控制寄存器 （UEP0 到 UEP7）

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL(1)
bit 7							bit 0

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

bit 7-5 **未实现：**读为 0

bit 4 **EPHSHK：**端点握手使能位
1 = 使能端点握手
0 = 禁止端点握手 （通常用于同步端点）

bit 3 **EPCONDIS：**双向端点控制位
如果 EPOUTEN = 1 并且 EPINEN = 1：
1 = 禁止端点 n 的控制传输；只允许 IN 和 OUT 传输
0 = 使能端点 n 的控制 （SETUP）传输；同时允许 IN 和 OUT 传输

bit 2 **EPOUTEN：**端点输出使能位
1 = 使能端点 n 输出
0 = 禁止端点 n 输出

bit 1 **EPINEN：**端点输入使能位
1 = 使能端点 n 输入
0 = 禁止端点 n 输入

bit 0 **EPSTALL：**端点停止使能位 (1)
1 = 端点 n 已停止
0 = 端点 n 未停止

注 1： 仅当使能端点 n 时有效；否则将忽略该位。

22.2.5 USB 地址寄存器（UADDR）

USB 地址寄存器包含惟一的 USB 地址，当此地址有效时，可被外设识别。在收到 URSTIF 指示的“USB 复位”或从单片机收到“复位”时，UADDR 将复位为 00h。USB 地址必须在 USB 启动阶段（枚举）由单片机写入，该操作受 Microchip USB 固件支持。

22.2.6 USB 帧编号寄存器（UFRMH:UFRML）

帧编号寄存器包含 11 位的帧编号。低位字节保存在 UFRML 中，而高 3 位保存在 UFRMH 中。每当收到 SOF 令牌时，就用当前帧编号更新寄存器对。对于单片机，这些寄存器是只读的。帧编号寄存器主要用于同步传输。仅当48 MHz SIE时钟有效时，UFRMH和UFRML寄存器的内容才有效（即，当 SUSPND（UCON<1>）位 = 1 时，它们的内容是不准确的）。

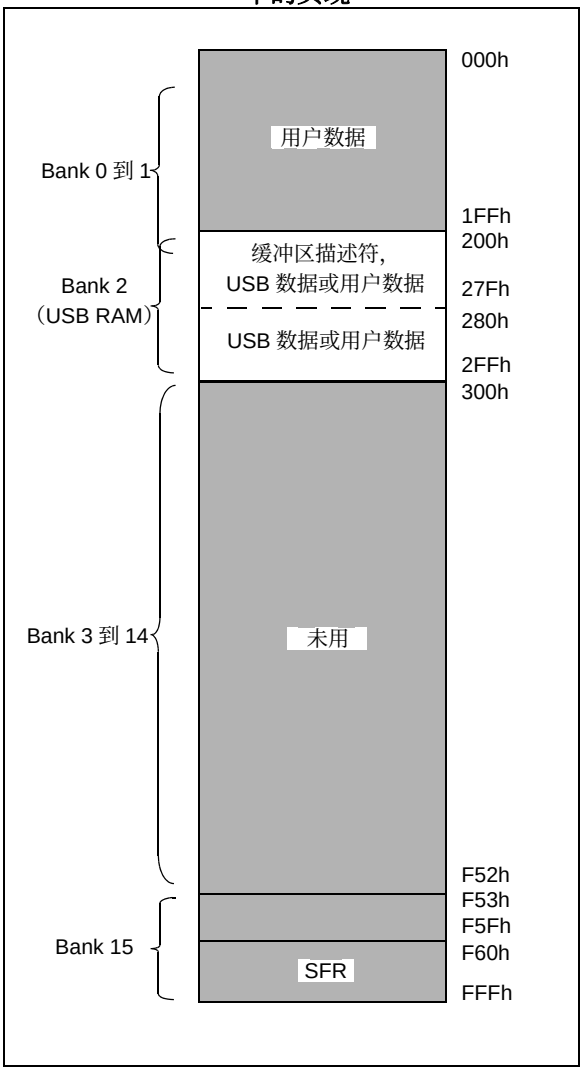
22.3 USB RAM

USB 数据通过称为 USB RAM 的存储空间在单片机内核与 SIE 之间传送。这是特殊的双端口存储器，被映射到 Bank 2（200h 到 2FFh）中的普通数据存储空间，总共 256 字节（图 22-4）。

Bank 2（200h 到 27Fh）专门用于端点缓冲区控制。根据使用的缓冲类型，除 8 个字节外，Bank 2 的所有其他单元都可用作 USB 缓冲空间。

尽管单片机可使用 USB RAM 作为数据存储器，但是正由 SIE 访问的部分不应被单片机访问。信号机制被用来确定任意给定时间对特定缓冲区的访问。这将在**第 22.4.1.1 节“缓冲区所有权”**中讨论。

图 22-4：　USB RAM 在数据存储空间中的实现



22.4 缓冲区描述符和缓冲区描述符表

Bank 2 中的寄存器组成称为缓冲区描述符表 （Buffer Descriptor Table，BDT）的结构，专门用于端点缓冲区控制。它为用户提供了灵活的方法，用于构造和控制不同长度及配置的端点缓冲区。

BDT 由缓冲区描述符 （BD）组成，后者用于定义和控制 USB RAM 空间中的实际缓冲区。每个 BD 依次由四个寄存器组成，其中 n 代表 32 个可能的 BD 中的一个（范围是 0 到 31）：

- BDnSTAT：BD 状态寄存器
- BDnCNT：BD 字节计数寄存器
- BDnADRL：BD 低位地址寄存器
- BDnADRH：BD 高位地址寄存器

BD 在序列中总是以一个四字节的数据块 （BDnSTAT:BDnCNT:BDnADRL:BDnADRH）出现。BDnSTAT 的地址总是一个相对于 200h 的偏移量 （4n – 1）（十六进制），其中 n 是缓冲区描述符编号。

根据使用的缓冲配置 （**第 22.4.4 节 “ 乒乓缓冲 ”**），最多可以有 16、17 或 32 组缓冲区描述符。BDT 至少要有 8 个字节长。这是因为 USB 规范要求每个器件都必须有输入和输出端点 0，用于建立最初的通信。根据端点和缓冲配置， BDT 可以长达 128 字节。

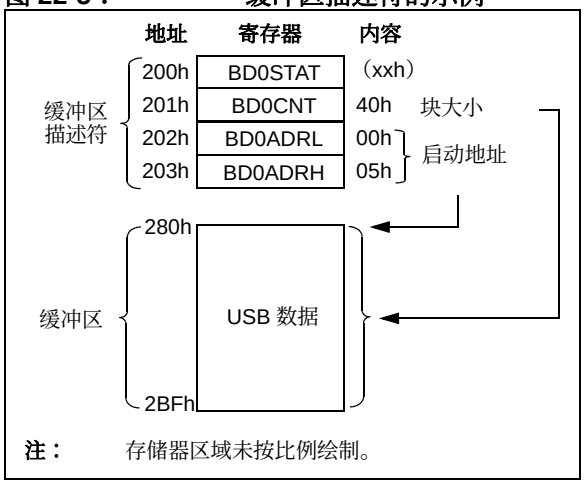
尽管缓冲区描述符状态和地址寄存器可被视为特殊功能寄存器，但它们和 Bank 15 中的传统单片机 SFR 不同，并不是硬件映射的。如果对应于特定 BD 的端点未被使能，其寄存器将不会被用到。它们将显示为可用的 RAM，而非未实现地址。仅当通过将 UEPn<1> 位置 1 使能了端点时，那些地址中的存储器才能用作 BD 寄存器。正如数据存储空间中的任何单元一样，BD 寄存器在任何器件复位时，包含的都是一个不确定的值。

图 22-5 所示为用于 64 字节缓冲区 （从 280h 开始）的 BD 的示例。特定的 BD 寄存器组仅当相应端点已使用 UEPn 寄存器使能后才有效。所有 BD 寄存器都在 USB RAM 中可用。每个端点的 BD 都应在使能该端点前设置。

22.4.1 BD 状态和配置

缓冲区描述符不仅定义端点缓冲区的大小，还确定其配置和控制。多数配置是通过 BD 状态寄存器 BDnSTAT 完成的。每个 BD 都有其惟一的、对应编号的 BDnSTAT 寄存器。

图 22-5： 缓冲区描述符的示例



和其他控制寄存器不同， BDnSTAT 寄存器的位的配置与具体应用息息相关。有两种不同的配置，取决于在某个特定时间修改 BD 和缓冲区的是单片机还是 USB 模块。只有 3 个位是两者共用的。

22.4.1.1 缓冲区所有权

由于缓冲区及其BD是在CPU和USB模块之间共用的，因此采用了简单的信号机制来区分允许哪个模块更新存储器中的 BD 和相关缓冲区。

这是通过使用 UOWN 位 （BDnSTAT<7>）作为信号来实现的。UOWN 是在 BDnSTAT 的两个配置之间共用的惟一一个位。

UOWN 清零时，BD 归单片机内核 “ 所有 ”。UOWN 位置 1 时， BD 和缓冲存储区归 USB 外设 “ 所有 ”。此时内核不应修改 BD 或其对应的数据缓冲区。注意， SIE 拥有缓冲区时，单片机内核仍可以读取 BDnSTAT，反之亦然。

根据寄存器的更新源的不同，缓冲区描述符具有不同的意义。在分配给 USB 外设所有权之前，用户可以通过 BDnSTAT 位配置外设的基本操作。此时也可设置字节计数和缓冲位置寄存器。

UOWN 置 1 时，用户不能再信赖写入 BD 的值。从此时起，SIE 根据需要更新 BD，覆盖原始 BD 值。BDnSTAT 寄存器由 SIE 用令牌 PID 更新，传输计数 BDnCNT 也被更新。

在准备与端点通信时，最后更新的字节应该总是BDT的BDnSTAT字节。SIE在事务完成时将把UOWN位清零。

UOWN位置1时，没有任何硬件机制来阻止单片机对存储器的访问。因此，如果单片机试图在SIE拥有存储器时修改该存储器，可能会出现无法预料的结果。同样，在 USB 外设将所有权归还单片机之前，读取该存储器可能会导致数据出错。

22.4.1.2 BDnSTAT 寄存器 （CPU 模式）

当 UOWN = 0 时，单片机内核拥有 BD。此时，寄存器的其他 7 位担负控制功能。

数据翻转同步使能位 DTSEN （BDnSTAT<3>）控制数据翻转奇偶检验。将 DTSEN 置 1 将使能 SIE 数据翻转同步。使能时，会将数据包的奇偶检验位与 DTS （BDnSTAT<6>）的值进行比对。如果数据包到达时同步不正确，数据基本上将被忽略。它不会被写入 USB RAM， USB 传输完成中断标志也不会被置 1。但是，SIE 将把 ACK 令牌发送回主机，对接收作出应答。表 22-1 中总结了 DTSEN 位对 SIE 的影响。

缓冲区停止位 BSTALL （BDnSTAT<2>）支持控制传输，通常在端点 0 停止一次。它还支持 USB 规范第 9 章中指定的SET_FEATURE/CLEAR_FEATURE命令；通常连续将STALL令牌传送给除了默认控制端点外的其他端点。

BSTALL 位使能缓冲区停止。将 BSTALL 置 1， SIE 会把 STALL 令牌归还主机 （如果收到的令牌使用该地址的 BD）。将 STALL 发送到主机时，对应的 UEPn 控制寄存器中的 EPSTALL 位置 1 并产生 STALL 中断。收到 SETUP 令牌前，UOWN 位将保持置 1 且 BD 不会更改。在这种情况下，STALL 状态将被清零，BD 的所有权将归还给单片机内核。

BD<9:8> 位（BDnSTAT<1:0>）存储 SIE 字节计数的高 2 位；低 8 位将存储在对应的 BDnCNT 寄存器中。更多信息，请参见**第 22.4.2 节 “ BD 字节计数 ”**。

表 22-1： DTSEN 位对奇 / 偶 （DATA0/DATA1）数据包接收的影响

来自主机的 OUT 数据包	BDnSTAT 设置		收到数据包后的器件响应			
	DTSEN	DTS	握手	UOWN	TRNIF	BDnSTAT 和 USTAT 状态
DATA0	1	0	ACK	0	1	已更新
DATA1	1	0	ACK	1	0	未更新
DATA0	1	1	ACK	1	0	未更新
DATA1	1	1	ACK	0	1	已更新
两者中任一	0	x	ACK	0	1	已更新
任一，有错误	x	x	NAK	1	0	未更新

图注： x = 无关位

寄存器 22-5： BDnSTAT：缓冲区描述符 n 状态寄存器 （BD0STAT 到 BD31STAT）， CPU 模式（数据写入 CPU 控制的存储区）

R/W-x	R/W-x	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
UOWN(1)	DTS(2)	—(3)	—(3)	DTSEN	BSTALL	BC9	BC8
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 **UOWN：**USB 所有权位 (1)
0 = 单片机内核拥有 BD 及其对应缓冲区

bit 6 **DTS：**数据翻转同步位 (2)
1 = 数据 1 数据包
0 = 数据 0 数据包

bit 5-4 **未实现：**这些位应总是设定为 0(3)。

bit 3 **DTSEN：**数据翻转同步使能位
1 = 使能数据翻转同步；SETUP事务除外，同步值错误的数据包将被忽略；对于SETUP事务，即使数据翻转位不匹配，数据包也会被接受
0 = 不会执行数据翻转同步

bit 2 **BSTALL：**缓冲区停止使能位
1 = 使能缓冲区停止；如果收到使用指定地址BD的令牌，将发出停止握手（UOWN位保持置1，BD值不变）
0 = 禁止缓冲区停止

bit 1-0 **BC<9:8>：**字节计数的 bit 9 和 bit 8
该字节计数位代表将为某 IN 令牌发送或 OUT 令牌期间接收的字节数。和 BC<7:0> 一起，有效字节计数是 0-1023。

注 1： 必须在使能 USB 模块之前，由用户将该位初始化为所需值。
2： 除非 DTSEN = 1，否则该位将被忽略。
3： 如果这些位被置 1， USB 可能无法进行通信。因此，这些位应该始终保持为 0。

22.4.1.3 BDnSTAT 寄存器 （SIE 模式）

BD 及其缓冲区由 SIE 所有时，BDnSTAT 中的大多数位都有了不同的含义。配置如寄存器 22-6 所示。一旦 UOWN 位被置 1，先前用户写入的任何数据或控制设置都将被来自 SIE 的数据覆盖。

由 SIE 使用令牌数据包标识符（Packet Identifier，PID）更新 BDnSTAT 寄存器，该 PID 存储在 BDnSTAT<5:3> 中。对应 BDnCNT 寄存器中的传输计数值也会更新。该 8 位寄存器的值溢出得到计数值的高 2 位，存储在 BDnSTAT<1:0> 中。

22.4.2 BD 字节计数

字节计数代表 IN 传输期间将发送的字节总数。 IN 传输完成后， SIE 将返回发送到主机的字节数。

对于 OUT 传输，字节计数代表可接收并存储在 USB RAM 中的最大字节数。 OUT 传输完成后， SIE 将返回接收到的实际字节数。如果接收到的字节数超过了设定的字节计数，数据包将被拒绝，并生成一个 NAK 握手信号。在这种情况下，不会更新字节计数。

10 位字节计数保存在两个寄存器中。计数的低 8 位驻留在 BDnCNT 寄存器中，而高 2 位驻留在 BDnSTAT<1:0> 中。这代表字节数的有效范围是 0 到 1023。

22.4.3 BD 地址验证

BD 地址寄存器对包含对应端点缓冲区的起始 RAM 地址。没有可以验证 BD 地址有效性的硬件机制。

如果 BD 地址的值没有指向 USB RAM，或指向其他端点缓冲区中的地址，数据很可能丢失或被覆盖。同样，如果接收缓冲（OUT 端点）和使用中的 BD 地址重叠，也会导致无法预料的后果。开发 USB 应用程序时，用户可能需要考虑在代码中加入基于软件的地址验证。

寄存器 22-6： BDnSTAT：缓冲区描述符 n 状态寄存器 （BD0STAT 到 BD31STAT）， SIE 模式 （从 SIE 返回数据到 MCU）

R/W-x	U-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
UOWN	—	PID3	PID2	PID1	PID0	BC9	BC8
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 **UOWN**：USB 所有权位
1 = SIE 拥有 BD 及其对应缓冲区

bit 6 **保留**：不是由 SIE 写入的

bit 5-2 **PID<3:0>**：数据包标识符位
上次传输收到的令牌 PID 值 （仅适用于 IN、 OUT 或 SETUP 事务）。

bit 1-0 **BC<9:8>**：字节计数的 bit 9 和 bit 8
这些位由 SIE 更新，以便反映 OUT 传输中收到的和 IN 传输中发送的实际字节数。

22.4.4 乒乓缓冲

当端点有两组 BD 项时（一组用于偶数数据传输，一组用于奇数数据传输），则该端点被定义为具有乒乓缓冲区。这样就允许在 CPU 处理一组 BD 的同时，SIE 处理另一组 BD。以这种方式实现的双重缓冲 BD 能获得最高的 USB 输入 / 输出流量。

USB 模块支持 4 种工作模式：

- 不支持乒乓缓冲区
- 仅对 OUT 端点 0 提供乒乓缓冲区支持
- 对所有端点提供乒乓缓冲区支持
- 对除端点 0 外的所有其他端点提供乒乓缓冲区支持

乒乓缓冲区设置是用 UCFG 寄存器中的 PPB<1:0> 位配置的。

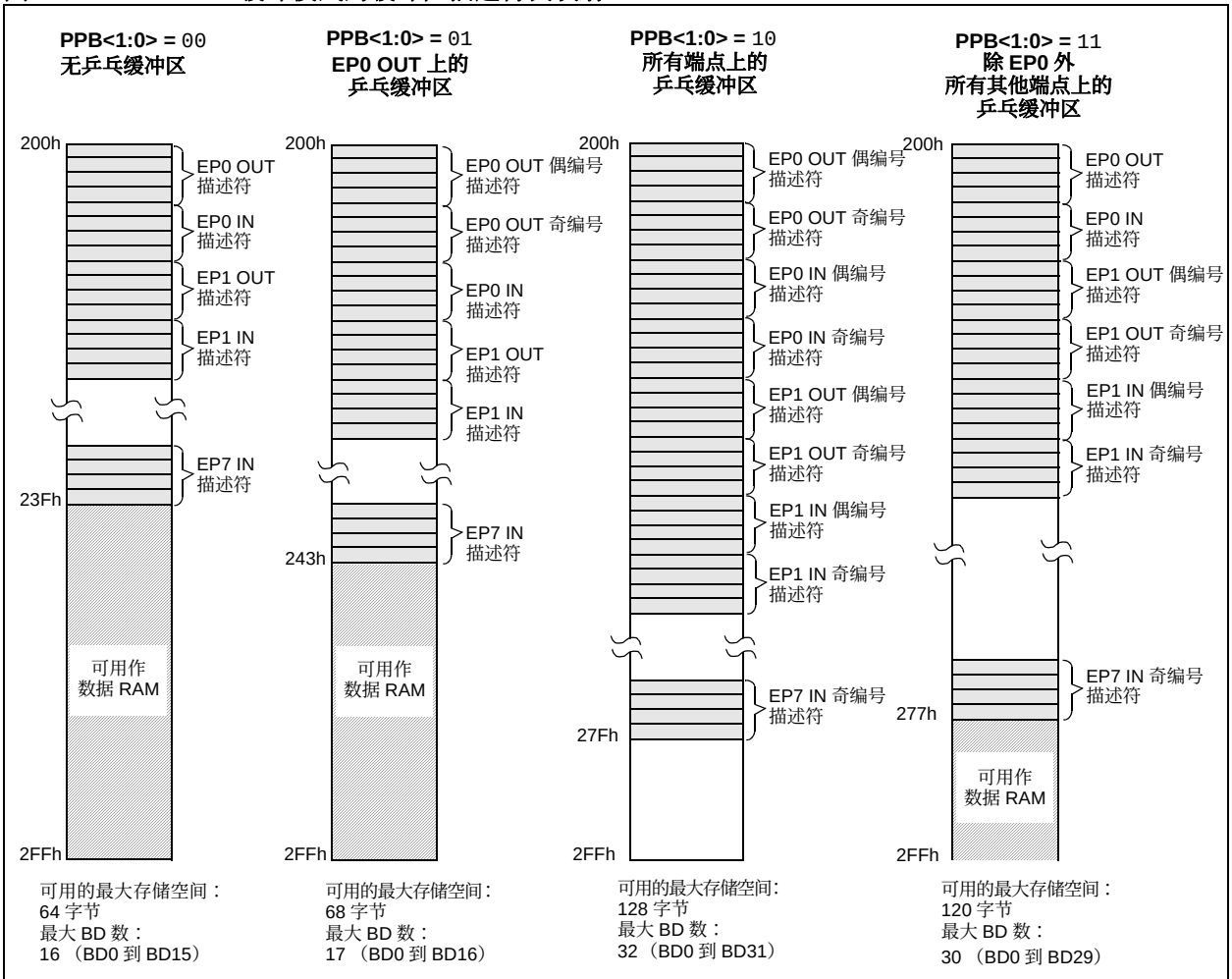
USB 模块跟踪每个端点的乒乓指针。该模块使能时，所有指针初始时都复位到偶编号 BD。事务完成后（UOWN 由 SIE 清零），指针转向奇编号 BD。下个事务完成后，指针指回偶编号 BD，以此类推。

上个事务的奇 / 偶状态存储在 USTAT 寄存器的 PPBI 位中。用户可以用 PPBRST 位将所有乒乓指针复位到偶编号 BD。

图 22-6 显示了 4 种不同的工作模式，以及 BD 是如何填充 USB RAM 的。

根据缓冲配置， BD 和特定端点有着固定的对应关系。BD 到端点的映射在表 22-2 中有详细描述。这一关系也意味着如果端点不是连续使能的，BDT 中可能出现空白区域。已禁止端点的 BD 理论上可用作缓冲空间，但在实际应用中，用户应避免使用 BDT 中的这类空间，除非采取了验证 BD 地址的措施。

图 22-6： 缓冲模式的缓冲区描述符表映射



注： 存储器区域未按比例显示。

表 22-2： 不同缓冲模式的缓冲区描述符分配

端点	分配给端点的 BD							
	模式 0（无乒乓缓冲区）		模式 1（EP0 OUT 上的乒乓缓冲区）		模式 2（所有端点上的乒乓缓冲区）		模式 3（除 EP0 外所有其他端点上的乒乓缓冲区）	
	输出	输入	输出	输入	输出	输入	输出	输入
0	0	1	0 (E), 1 (O)	2	0 (E), 1 (O)	2 (E), 3 (O)	0	1
1	2	3	3	4	4 (E), 5 (O)	6 (E), 7 (O)	2 (E), 3 (O)	4 (E), 5 (O)
2	4	5	5	6	8 (E), 9 (O)	10 (E), 11 (O)	6 (E), 7 (O)	8 (E), 9 (O)
3	6	7	7	8	12 (E), 13 (O)	14 (E), 15 (O)	10 (E), 11 (O)	12 (E), 13 (O)
4	8	9	9	10	16 (E), 17 (O)	18 (E), 19 (O)	14 (E), 15 (O)	16 (E), 17 (O)
5	10	11	11	12	20 (E), 21 (O)	22 (E), 23 (O)	18 (E), 19 (O)	20 (E), 21 (O)
6	12	13	13	14	24 (E), 25 (O)	26 (E), 27 (O)	22 (E), 23 (O)	24 (E), 25 (O)
7	14	15	15	16	28 (E), 29 (O)	30 (E), 31 (O)	26 (E), 27 (O)	28 (E), 29 (O)

图注： (E) = 偶数事务缓冲， (O) = 奇数事务缓冲

表 22-3： USB 缓冲区描述符表寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BDnSTAT(1)	UOWN	DTS(4)	PID3(2)	PID2(2)	PID1(2) DTSEN(3)	PID0(2) BSTALL(3)	BC9	BC8
BDnCNT(1)	字节计数							
BDnADRL(1)	缓冲区地址低字节							
BDnADRH(1)	缓冲区地址高字节							

注 1： 对于缓冲区描述符寄存器， n 是 0 到 31 之间的某个值。为了简明起见，所有 32 个寄存器都显示为一个通用名称代号。所有寄存器都有不确定的复位值 （xxxx xxxx）。

2： BDnSTAT 寄存器的 bit 5 到 bit 2 是在该寄存器受控于 SIE 后 （UOWN 位置 1），由 SIE 用来返回 PID<3:0> 值的。一旦寄存器处于 SIE 控制下， DTSEN 和 BSTALL 写入的值就不再有效。

3： 将缓冲区描述符所有权移交给 SIE 前 （UOWN 位清零）， BDnSTAT 寄存器的 bit 5 到 bit 2 用于配置 DTSEN 和 BSTALL 设置。

4： 除非 DTSEN = 1，否则该位将被忽略。

22.5 USB 中断

USB 模块可生成多个中断条件。为了识别所有这些中断源，该模块配备有自身的中断逻辑结构（和单片机的结构相似）。USB 中断是用一组控制寄存器允许的，并用一组独立的标志寄存器来捕捉中断事件。所有中断源在单片机的中断逻辑中都归为一个 USB 中断请求 USBIF（PIR2<2>）。

图 22-7 给出了 USB 模块的中断逻辑。USB 模块的中断寄存器分为两层。顶层全部由 USB 状态中断组成；这些中断分别是在 UIE 及 UIR 寄存器中允许和标记的。第二层由USB错误条件中断组成，分别是在UEIE及UEIR寄存器中允许和标记的。上述任意一个中断条件都将触发顶层的 USB 错误中断标志 （UERRIF）。

可使用中断来处理 USB 传输中的常规事务。图 22-8 展示了USB帧传输过程中的某些常见事件及其对应中断。

图 22-7： USB 中断逻辑

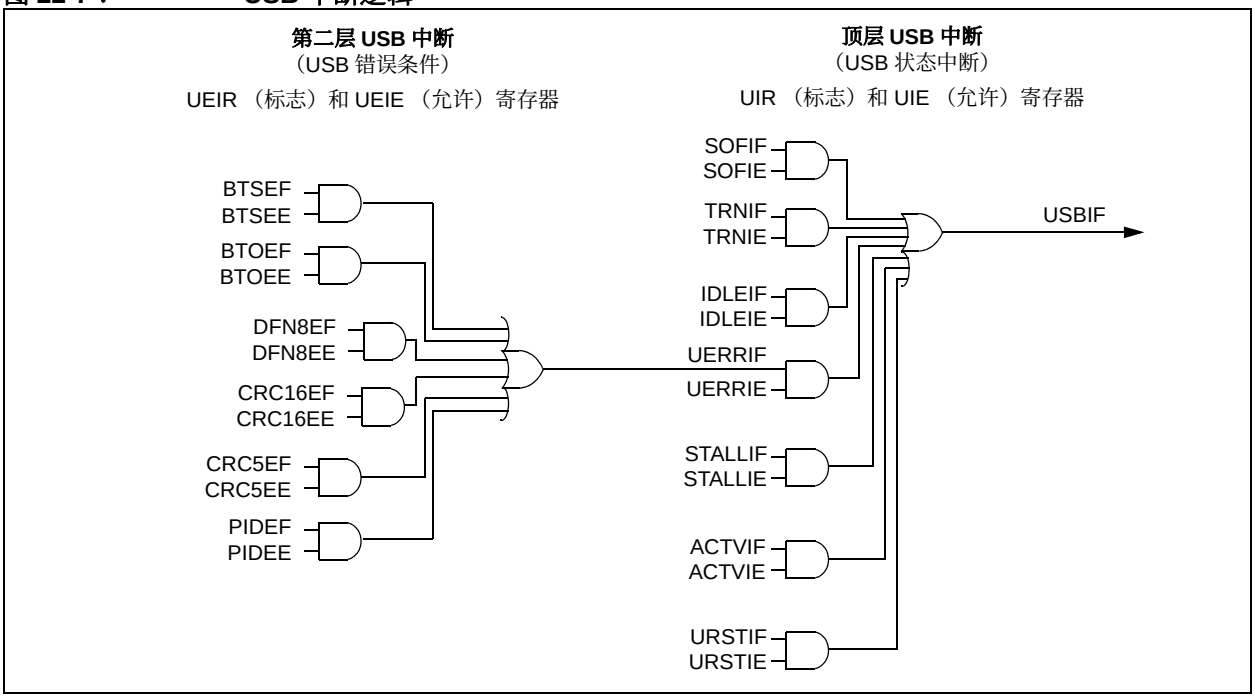
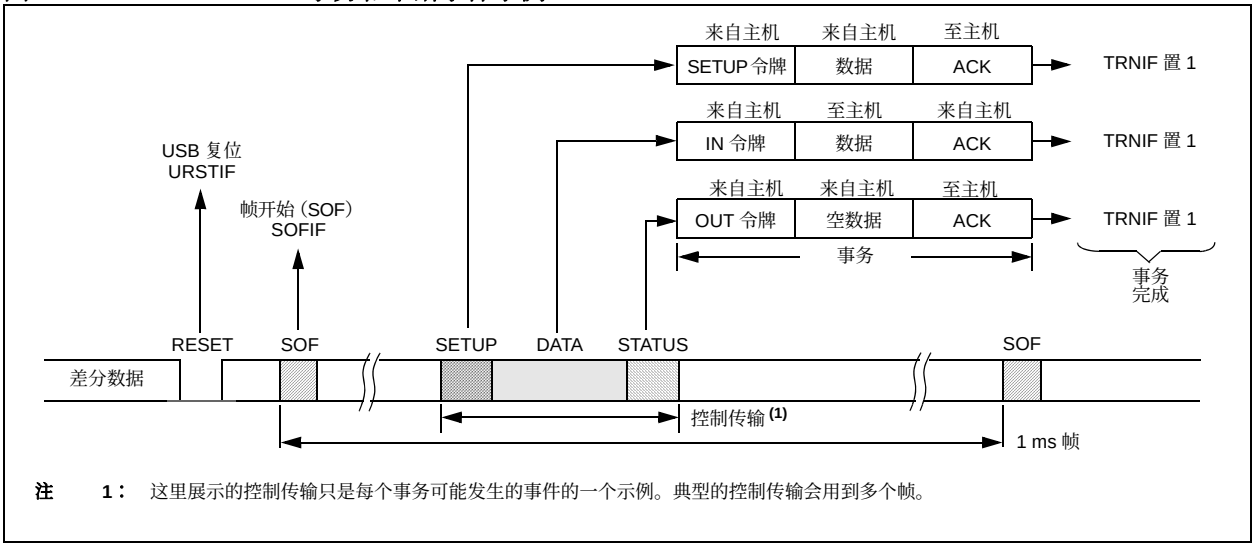


图 22-8： USB 事务和中断事件示例



注 1： 这里展示的控制传输只是每个事务可能发生的事件的一个示例。典型的控制传输会用到多个帧。

22.5.1 USB 中断状态寄存器 （UIR）

USB 中断状态寄存器（寄存器 22-7）包含每个 USB 状态中断源的标志位。这些中断源中的每一个在 UIE 寄存器中都有相应的中断允许位。所有 USB 状态标志一起作逻辑或运算，生成单片机的 USBIF 中断标志。

一旦由 SIE 将中断位置 1 后，必须由软件通过写入 0 来清零。也可以用软件置 1 标志位，这有助于固件调试。

寄存器 22-7： UIR：USB 中断状态寄存器

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0
—	SOFIF	STALLIF	IDLEIF(1)	TRNIF(2)	ACTVIF(3)	UERRIF(4)	URSTIF
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 **SOFIF：**SOF 令牌中断位
1 = SIE 收到的 SOF 令牌
0 = SIE 没有收到 SOF 令牌

bit 5 **STALLIF：**STALL 握手中断位
1 = SIE 发出了一个 STALL 握手
0 = 未发出 STALL 握手

bit 4 **IDLEIF：**空闲检测中断位 (1)
1 = 检测到空闲状态 （3 ms 或更长的连续空闲状态）
0 = 未检测到空闲状态

bit 3 **TRNIF：**事务完成中断位 (2)
1 = 待处理事务处理完成；从 USTAT 寄存器读取端点信息
0 = 未处理完待处理事务或没有待处理事务

bit 2 **ACTVIF：**总线活动检测中断位 (3)
1 = D+/D- 线上检测到活动
0 = D+/D- 线上没有检测到活动

bit 1 **UERRIF：**USB 错误条件中断位 (4)
1 = 发生了未屏蔽的错误条件
0 = 未发生未屏蔽的错误条件

bit 0 **URSTIF：**USB 复位中断位
1 = 发生有效的 USB 复位； 00h 装入 UADDR 寄存器
0 = 没有发生 USB 复位

注 1： 一旦检测到空闲状态，用户可能希望将 USB 模块置于挂起模式中。
2： 将该位清零可导致 USTAT FIFO 前移 （仅对 IN、 OUT 和 SETUP 令牌有效）。
3： 该位通常只在检测到 UIDLE 中断事件后是未屏蔽的。
4： 只有通过 UEIE 寄存器使能的错误条件才能将该位置 1。该位只是状态位，用户不能将它置 1 或清零。

22.5.1.1 总线活动检测中断位 （ACTVIF）

在USB模块从挂起模式唤醒后，或USB模块被挂起时，不能立即清零 ACTVIF 位。需要先用一些时钟周期来同步内部硬件状态机，然后固件才能清零 ACTVIF 位。在内部硬件同步之前清零 ACTVIF 位可能不会对 ACTVIF 值产生影响。此外，如果 USB 模块使用来自 48 MHz PLL 源的时钟，那么在清零 SUSPND 位之后，USB 模块需要等待48 MHz PLL锁定，可能不会立即开始工作。应用程序代码应按例 22-1 中所示，清零 ACTVIF 标志。

从 USB 总线空闲状态中重新开始工作时，只会产生一个 ACTVIF 中断。如果用户固件清零 ACTVIF 位，即使存在连续的总线通信，该位也不会立即再次置 1。总线通信必须停止足够长的时间来产生另一个IDLEIF条件，然后才会产生另一个 ACTVIF 中断。

例 22-1： 清零 ACTVIF 位 （UIR<2>）

汇编语言：

```
    BCF     UCON, SUSPND
LOOP:
    BTFSS   UIR, ACTVIF
    BRA     DONE
    BCF     UIR, ACTVIF
    BRA     LOOP
DONE:
```

C 语言：

```
UCONbits.SUSPND = 0;
while (UIRbits.ACTVIF) { UIRbits.ACTVIF = 0; }
```

22.5.2 USB 中断允许寄存器 （UIE）

USB 中断允许寄存器（寄存器 22-8）包含 USB 状态中断源的允许位。将这些位中的任何一个置 1 将允许 UIR 寄存器中的对应中断源。

该寄存器中的值只影响是否将该中断条件传递给单片机中断逻辑。标志位仍是由其中断条件置 1 的，允许对它们进行查询和处理而无需实际生成中断。

寄存器 22-8： UIE：USB 中断允许寄存器

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	SOFIE	STALLIE	IDLEIE	TRNIE	ACTVIE	UERRIE	URSTIE
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 **SOFIE：**SOF 令牌中断允许位
1 = 允许 SOF 令牌中断
0 = 禁止 SOF 令牌中断

bit 5 **STALLIE：**STALL 握手中断允许位
1 = 允许 STALL 中断
0 = 禁止 STALL 中断

bit 4 **IDLEIE：**空闲检测中断允许位
1 = 允许空闲检测中断
0 = 禁止空闲检测中断

bit 3 **TRNIE：**事务完成中断允许位
1 = 允许事务中断
0 = 禁止事务中断

bit 2 **ACTVIE：**总线活动检测中断允许位
1 = 允许总线活动检测中断
0 = 禁止总线活动检测中断

bit 1 **UERRIE：**USB 错误中断允许位
1 = 允许 USB 错误中断
0 = 禁止 USB 错误中断

bit 0 **URSTIE：**USB 复位中断允许位
1 = 允许 USB 复位中断
0 = 禁止 USB 复位中断

22.5.3 USB 错误中断状态寄存器 （UEIR）

USB 错误中断状态寄存器（寄存器 22-9）包含 USB 外设中每个错误源的标志位。这些源中的每一个都由 UEIE 寄存器中相应的中断允许位控制。所有 USB 错误标志一起进行或运算，生成中断逻辑顶层的 USB 错误中断标志 （UERRIF）。

一旦检测到错误条件，就会立即将对应的错误标志位置 1。因此，中断通常不会在令牌处理结束时产生。

一旦 SIE 将某中断位置 1，就必须通过软件写入 0 进行清零。

寄存器 22-9： UEIR：USB 错误中断状态寄存器

R/C-0	U-0	U-0	R/C-0	R/C-0	R/C-0	R/C-0	R/C-0
BTSEF	—	—	BTOEF	DFN8EF	CRC16EF	CRC5EF	PIDEF
bit 7							bit 0

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

bit 7 **BTSEF：**位填充错误标志位
1 = 检测到位填充错误
0 = 无位填充错误

bit 6-5 **未实现：**读为 0

bit 4 **BTOEF：**总线周转 （Turnaround）超时错误标志位
1 = 发生总线周转超时 （在上个 EOP 之后，已经过了 16 比特以上的空闲时间）
0 = 未发生总线周转超时

bit 3 **DFN8EF：**数据字段大小错误标志位
1 = 数据字段的字节数不是整数
0 = 数据字段的字节数是整数

bit 2 **CRC16EF：**CRC16 失败标志位
1 = CRC16 失败
0 = CRC16 通过

bit 1 **CRC5EF：**CRC5 主机错误标志位
1 = 令牌包由于 CRC5 错误而被拒绝
0 = 令牌包被接受

bit 0 **PIDEF：**PID 检查失败标志位
1 = PID 检查失败
0 = PID 检查通过

22.5.4 USB 错误中断允许寄存器 （UEIE）

USB 错误中断允许寄存器 （寄存器 22-10）包含每个 USB 错误中断源的允许位。将这些位中的任何一个置 1，都将允许把 UEIR 寄存器中的对应错误中断源传递到中断逻辑顶层的 UERR 位中。

同 UIE 寄存器一样，该允许位只影响是否将中断条件传递给单片机中断逻辑。标志位仍是由其中断条件置 1 的，允许对它们进行查询和处理而无需实际生成中断。

寄存器 22-10： UEIE：USB 错误中断允许寄存器

R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
BTSEE	—	—	BTOEE	DFN8EE	CRC16EE	CRC5EE	PIDEE
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 **BTSEE：**位填充错误中断允许位
1 = 允许位填充错误中断
0 = 禁止位填充错误中断

bit 6-5 **未实现：**读为 0

bit 4 **BTOEE：**总线周转超时错误中断允许位
1 = 允许总线周转超时错误中断
0 = 禁止总线周转超时错误中断

bit 3 **DFN8EE：**数据字段大小错误中断允许位
1 = 允许数据字段大小错误中断
0 = 禁止数据字段大小错误中断

bit 2 **CRC16EE：**CRC16 失败中断允许位
1 = 允许 CRC16 失败中断
0 = 禁止 CRC16 失败中断

bit 1 **CRC5EE：**CRC5 主机错误中断允许位
1 = 允许 CRC5 主机错误中断
0 = 禁止 CRC5 主机错误中断

bit 0 **PIDEE：**PID 检查失败中断允许位
1 = 允许 PID 检查失败中断
0 = 禁止 PID 检查失败中断

22.6 USB 电源模式

许多USB应用都可能有多组不同的电源要求和配置。最常见的电源模式是仅总线供电（Bus Power Only）、仅自供电（Self-Power Only）和以自供电为主的双电源供电（Dual Power）模式。这里列出了最常见的情况。同时还提供了一种估算 USB 收发器电流消耗的方法。

22.6.1 仅总线供电

在仅总线供电模式中，应用的所有电源都来自 USB（图 22-9）。这是对设备有效供电的最简单方式。

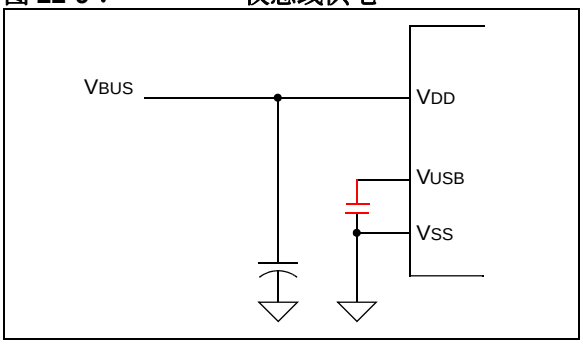
为了满足 USB 2.0 规范的涌入电流要求， VBUS 和地之间的有效总电容不能超出 10 µF。如果超出，则需要某种涌入电流限制措施。更多详细信息，请参见 USB 2.0 规范的第 7.2.4 节。

根据 USB 2.0 规范，所有 USB 器件还必须支持低功耗挂起模式。在 USB 挂起模式下，器件从 USB 电缆的 5V VBUS 线消耗的电流不能超出 500 µA（或者，对于支持远程唤醒的高功耗设备，不能超出 2.5 mA）。

主机通过以下方式指示 USB 器件进入挂起模式：停止到该 USB 器件的所有 USB 通信，保持超过 3 ms 的时间。该条件会导致 UIR 寄存器中的 IDLEIF 位置 1。

在 USB 挂起模式下，D+ 或 D- 上拉电阻必须保持有效，这会消耗一部分允许的挂起电流：500 µA/2.5 mA 的预算电流。

图 22-9： 仅总线供电



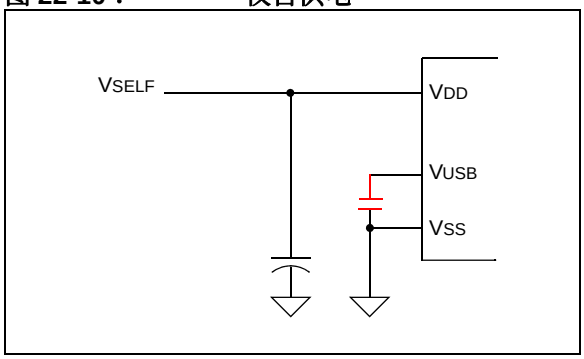
22.6.2 仅自供电

在仅自供电模式中，USB 应用为其自身提供电源，仅有很少一部分电源来自 USB。图 22-10 给出了一个示例。

为了满足合规性规范，只有主机主动将 VBUS 驱动为高电平之后，才能使能 USB 模块（以及 D+ 或 D- 上拉电阻）。

应用不能在 USB 电缆的 5V VBUS 引脚引入任何电流。

图 22-10： 仅自供电

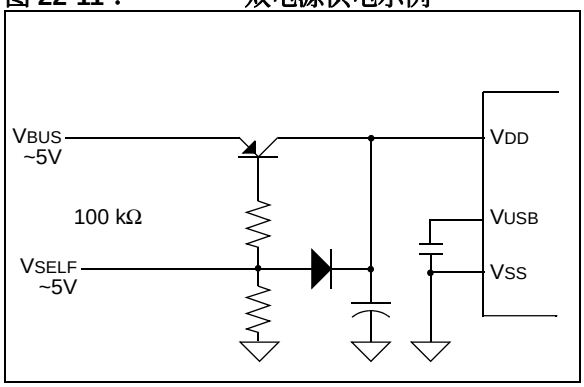


22.6.3 以自供电为主的双电源供电

某些应用可能需要两个供电选项。在该供电模式下，应用主要使用内部电源，但无内部电源可用时，可切换到 USB 供电。图 22-11 所示为一个以自供电为主的双电源供电模式的示例，它可在仅自供电模式和仅 USB 总线供电模式之间自动切换。

双电源供电器件还必须满足对于涌入电流和挂起模式电流的所有特殊要求，并且只有 VBUS 驱动为高电平之后才能使能 USB 模块。关于这些要求的说明，请参见**第 22.6.1 节“ 仅总线供电 ”**和**第 22.6.2 节“ 仅自供电 ”**。此外，双电源供电器件永远不能在 USB 电缆的 5V VBUS 引脚引入任何电流。

图 22-11： 双电源供电示例



注：	用户应牢记由 USB 总线供电的电流限制。根据 USB 规范 2.0，对单个低功耗设备，不能超过100 mA；对单个高功耗设备，不能超过 500 mA。

22.6.4 USB 收发器的电流消耗

根据 USB 电缆的特性阻抗、电缆的长度、VUSB 供电电压和通过 USB 电缆的实际数据模式， USB 收发器会消耗不同的电流量。电缆越长，电容就越大，在切换输出状态时，消耗的总电能就越多。

由“ IN ”（传入）通信组成数据模式消耗的电流远高于“ OUT ”（外发）通信。IN 通信要求 PIC® 器件驱动 USB 电缆，而 OUT 通信要求主机驱动 USB 电缆。

通过 USB 电缆发送的数据采用 NRZI 编码。在 NRZI 编码方案中，数据位为 0 时，收发器的输出状态会发生翻转 （从“ J ”状态变为“ K ”状态，或者反之）。进行位填充时的情况除外， NRZI 编码的数据位为 1 时，收发器的输出状态不会改变。因此，由值为 0 的数据位组成的IN通信会消耗最大的电流，因为收发器必须对USB电缆充电 / 放电，以便改变状态。

关于NRZI编码和位填充的更多详细信息，请参见USB 2.0规范的第 7.1 节，虽然使用 PIC18F1XK50/PIC18LF1XK50 单片机实现 USB 应用并不需要知道这些详细信息。除了其他方面之外，SIE 还在硬件中负责处理位填充 / 去填充、NRZI 编码 / 解码和 CRC生成 / 检查。

收发器消耗的总电流将取决于具体应用。但是，为了帮助估算在全速应用中实际可能需要多少电流，可以使用公式 22-1。

例 22-2 说明了可以如何对理论应用使用该公式。

公式 22-1： 估算 USB 收发器的电流消耗

$$I_{XCVR} = \frac{(60\ mA \bullet V_{USB} \bullet P_{ZERO} \bullet P_{IN} \bullet L_{CABLE})}{(3.3V \bullet 5m)} + I_{PULLUP}$$

图注：

V_{USB}： 以伏特为单位，施加到 V_{USB} 引脚的电压。（应为 3.0V 至 3.6V。）

P_{ZERO}： PIC® 器件发送的 IN 通信位中，值为 0 的数据位的百分比 （以小数表示）。

P_{IN}： 总线总带宽中用于 IN 通信的百分比 （以小数表示）。

L_{CABLE}： USB 电缆的长度 （以米为单位）。 USB 2.0 规范要求全速应用使用的电缆不超出 5 米。

I_{PULLUP}： 1.5 kΩ 标称上拉电阻 （使能时）必须为 USB 电缆提供的电流。在 USB 电缆的主机或集线器端，存在 15 kΩ 的标称电阻（14.25 kΩ 至 24.8 kΩ），它将 D+ 和 D- 线拉为地电压。在总线空闲条件下（例如，在两个数据包之间或在 USB 挂起模式下），电压为 3.3V 时，这会产生高达 218 µA 的静态电流。

I_{PULLUP} 也取决于总线通信条件，当数据在大多数时间都将线路驱动为 “ K ” 状态， USB 带宽得到完全利用 （IN 或 OUT 通信）时，电流最高可以达到 2.2 mA。

例 22-2： 计算 USB 收发器电流 †

对于该示例，应用存在以下假设：

- 为 V_{USB} 和 V_{DD} 施加 3.3V 的电压，并使能内核稳压器。
- 它是全速应用，使用一个可以每隔 1 ms 发送一个 64 字节数据包的中断 IN 端点，并且对发送的字节值没有限制。在 OUT 端点上，应用可能有也可能没有另外的通信。
- 在应用电路板中，将使用常规的 USB “ B ” 或 “ mini-B ” 连接器。

在此例中， P_{ZERO} = 100% = 1，因为对通过 IN 端点的数据值没有限制。所有 64 kBps 的数据都可能是值为 00h 的字节。因为数据位为 0 时，收发器的输出状态会发生翻转，所以它们会导致 USB 收发器消耗额外的电流来对电缆进行充电 / 放电。在此例中，所发送的数据位可以 100% 为值 0。该值应视为 “ 最大 ” 值，因为通常情况下，数据将由一定比例的 1 和 0 组成。

对于 IN 通信，该应用使用 1.5 MBps （12 Mbps）总线总带宽中的 64 kBps，因此：

$$Pin = \frac{64\ kBps}{1.5\ MBps} = 4.3\% = 0.043$$

因为该应用使用常规的 “ B ” 或 “ mini-B ” 连接器，所以最终用户可以插入任意类型的电缆 （最大允许长度为 5 米）。因此，使用极限情况的长度：

L_{CABLE} = 5 米

假设 I_{PULLUP} = 2.2 mA。I_{PULLUP} 的实际值可能更接近 218 µA，但这样的假设可以容许极限情况。USB 带宽由插入根端口（通过集线器）的所有设备共用。如果将应用插入已插有其他设备的 USB 1.1 集线器，则用户的设备可能会在总线上检测到主机到设备的通信，即使通信不是发送给该设备的。因为不论来源如何，所有通信都会使 I_{PULLUP} 电流升到高于基本电流 218 µA，所以容许极限情况 （2.2 mA 的电流）是最安全的。

因此：

$$I_{XCVR} = \frac{(60\ mA \bullet 3.3V \bullet 1 \bullet 0.043 \bullet 5m)}{(3.3V \bullet 5m)} + 2.2\ mA = 4.8\ mA$$

计算得到的值应视为近似值，建议采用额外的保护带或执行取决于具体应用的产品测试。收发器电流与 PIC18F1XK50/PIC18LF1XK50 器件为了运行内核、驱动其他 I/O 线、为各种模块供电等方面需要消耗的其他电流一起形成总电流消耗。

22.7 振荡器

USB 模块有特定的时钟要求。对于全速工作模式，时钟源必须是 48 MHz。虽然如此，单片机内核和其他外设不一定要以该时钟频率运行。可用的时钟选项在**第 2.11 节 “ USB 操作 ”** 中有详细描述。

22.8 D+/D- 引脚的电平变化中断

PIC18F/LF1XK50 在 D+ 和 D- 数据引脚上具有电平变化中断功能。该功能使器件在首次连接到 USB 主机 / 集线器时，可以检测到电平变化。

USB 主机 / 集线器在 D+ 和 D- 引脚上具有 15K 的上拉电阻。当 PIC18F/LF1XK50 连接到总线时，D+ 和 D- 引脚可以检测到电压变化。对于每个引脚，需要使用外部电阻，以便在断开时在引脚上维持高电平状态。

必须禁止 USB 模块 （USBEN = 0），电平变化中断功能才会起作用。使能 USB 模块 （USBEN = 1）时，将会自动禁止 D+ 和 D- 引脚的电平变化中断功能。更多详细信息，请参见**第 7.11 节 “ PORTA 和 PORTB 电平变化中断 ”**。

22.9 USB 固件和驱动程序

Microchip 提供一系列针对特定应用的资源，例如 USB 固件和驱动程序。请访问 www.microchip.com 获得最新的固件和驱动程序。

表 22-4： 与 USB 模块操作相关的寄存器 (1)

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	详情请见（页）：
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RABIE	TMR0IF	INT0IF	RABIF	70
IPR2	OSCFIP	C1IP	C2IP	EEIP	BCL1IP	USBIP	TMR3IP	—	78
PIR2	OSCFIF	C1IF	C2IF	EEIF	BCL1IF	USBIF	TMR3IF	—	74
PIE2	OSCFIE	C1IE	C2IE	EEIE	BCL1IE	USBIE	TMR3IE	—	76
UCON	—	PPBRST	SE0	PKTDIS	USBEN	RESUME	SUSPND	—	252
UCFG	UTEYE	—	—	UPUEN	—	FSEN	PPB1	PPB0	254
USTAT	—	ENDP3	ENDP2	ENDP1	ENDP0	DIR	PPBI	—	256
UADDR	—	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0	258
UFRML	FRM7	FRM6	FRM5	FRM4	FRM3	FRM2	FRM1	FRM0	252
UFRMH	—	—	—	—	—	FRM10	FRM9	FRM8	252
UIR	—	SOFIF	STALLIF	IDLEIF	TRNIF	ACTVIF	UERRIF	URSTIF	266
UIE	—	SOFIE	STALLIE	IDLEIE	TRNIE	ACTVIE	UERRIE	URSTIE	268
UEIR	BTSEF	—	—	BTOEF	DFN8EF	CRC16EF	CRC5EF	PIDEF	269
UEIE	BTSEE	—	—	BTOEE	DFN8EE	CRC16EE	CRC5EE	PIDEE	270
UEP0	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	257
UEP1	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	257
UEP2	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	257
UEP3	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	257
UEP4	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	257
UEP5	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	257
UEP6	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	257
UEP7	—	—	—	EPHSHK	EPCONDIS	EPOUTEN	EPINEN	EPSTALL	257

图注： — = 未实现，读为 0。 USB 模块不使用阴影单元。

注 1： 该表只包含在数据存储空间 Bank 15 中硬件映射的 SFR。缓冲区描述符寄存器映射到 Bank 4，它们不是真正的 SFR，另外列在表 22-3 中。

22.10 USB 概述

本节介绍 USB 的一些基本概念和设计 USB 设备所需的有用信息。尽管本节涵盖了很多内容，但 USB 规范和分类规范中提供了更为详尽的信息。因此，建议读者参考 USB 规范了解更多信息（www.usb.org）。如果您对 USB 非常熟悉，阅读本节可以帮您快速重温 USB 的相关知识。

22.10.1 分层框架

USB 设备的功能是以分层架构的形式实现的，如图 22-12 所示。每一层都和设备的某个功能层次关联。最高层是配置层。一个设备可有多个配置。例如，在仅自供电或仅总线供电模式的基础上，某个特定器件可能有多种供电要求。

对每种配置，可能有多个接口。每个接口支持该配置的某个特定模式。

接口之下是端点。数据直接在该层传送。最多可有 16 个双向端点。端点 0 总是控制端点，并且当设备在总线上时，默认情况下端点 0 必须可用于配置设备。

22.10.2 帧

在总线上通信的信息通常被划分为多个1 ms长的时隙，称为“ 帧 ”。每个帧可包含不同设备和端点的许多事务。图 22-8 给出了帧内一个事务的示例。

22.10.3 传输

USB 规范中定义了四种传输类型。

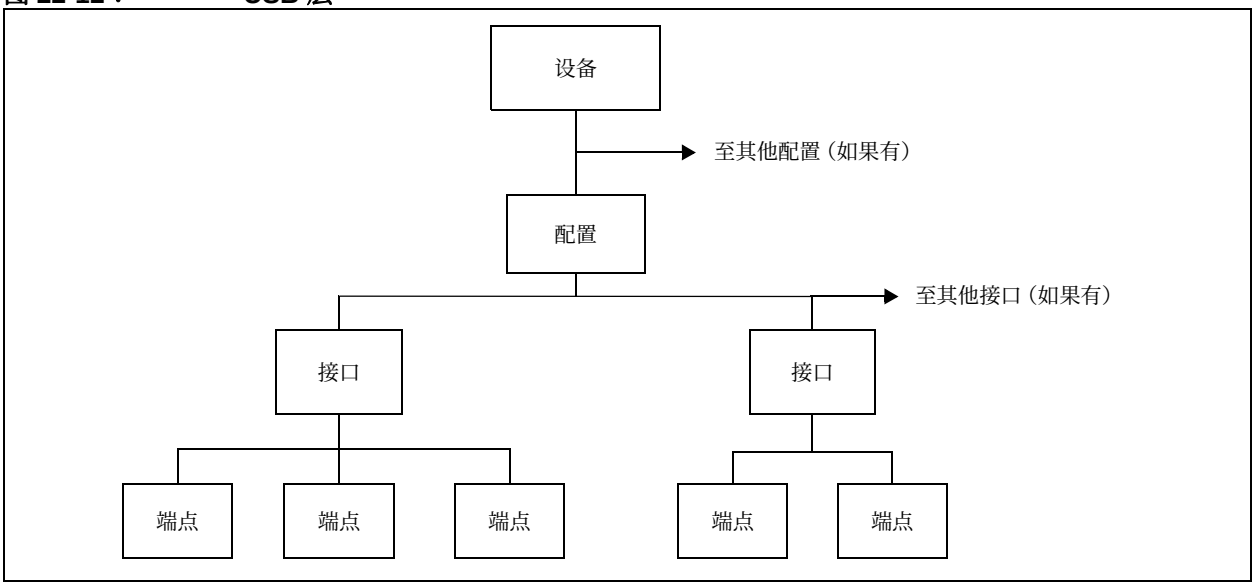
- **同步：**该类型可确保及时传递大量数据 （最多可达 1023 字节）；但不能保证数据的完整性。这适用于少量数据丢失影响不大的数据流应用，例如音频。
- **批量：**这类传输方式允许传输大量数据并确保数据的完整性；但不能保证传输时限。
- **中断：**这类传输可确保小块数据的及时传递，并能保证数据的完整性。
- **控制：**这类传输是为设备启动控制提供的。

全速设备支持所有传输类型，而低速设备只限于中断和控制传输。

22.10.4 电源

电源可从通用串行总线获得。USB 规范定义了总线供电的要求。设备既可以使用自身电源，也可以通过总线供电。自供电设备从外部电源获得电能，而总线供电设备使用总线提供的电源。

图 22-12： USB 层



USB 规范对取自总线的电能做出了限制。在电压大约为 5V 的条件下，每个设备确保能获得 100 mA 的电流（一个单位负载）。若需要更多的电量，最多可达 500 mA。注意，可以请求供给超过一个单位负载的电源功率，主机或集线器不一定会提供额外电流。因此，耗电量可能超过单位负载的设备必要时必须能够置于低功耗配置（一个单位负载及以下）中。

USB 规范还定义了挂起模式。在这种情况下，1 秒内的平均电流不能超过 500 µA。总线上没有任何活动 3 ms（即 3 ms 内没有 SOF 令牌）后，设备必须进入挂起状态。进入挂起模式的设备必须在挂起后 10 ms 内降低电流消耗。类似地，在唤醒时，设备必须在 10 ms 内拉高电流超出挂起门限值以示唤醒。

22.10.5 枚举

设备第一次连接到总线时，主机进入枚举进程，试图识别设备。基本上，主机会询问设备，收集类似功耗、数据传输速率和大小、协议及其他描述信息；描述符将包含这些信息。下面是一个典型的枚举进程：

1. USB 复位：将设备复位。因此，设备未配置，也没有地址（地址 0）。
2. 获取设备描述符：主机请求设备描述符的一小部分。
3. USB 复位：再次将设备复位。
4. 设置地址：主机将地址分配给设备。
5. 获取设备描述符：主机检索设备描述符，收集类似制造商、设备类型、最大控制包大小等信息。
6. 获取配置描述符。
7. 获取任何其他描述符。
8. 设置配置。

具体的枚举进程取决于主机。

22.10.6 描述符

有 8 种不同的标准描述符类型，其中 5 种对设备来讲非常重要。

22.10.6.1 设备描述符

设备描述符提供常规信息，例如制造商、产品编号、序列号、设备类别和配置数。只有一个设备描述符。

22.10.6.2 配置描述符

配置描述符提供设备电源要求以及在该配置下支持多少不同接口的信息。一个设备可能有多个配置（即低功耗和高功耗配置）。

22.10.6.3 接口描述符

接口描述符详细描述该接口中使用的端点数，以及接口类型。一个配置可能有多个接口。

22.10.6.4 端点描述符

端点描述符确定传输类型（**第 22.10.3 节“ 传输 ”**）和方向，以及端点的一些其他定义。一个设备中可以有许多端点，端点可在不同配置间共用。

22.10.6.5 字符串描述符

许多前面的描述符都引用了一个或多个字符串描述符。字符串描述符提供关于它们描述的层（**第 22.10.1 节“ 分层框架 ”**）的可读信息。通常，这些字符串出现在主机中，帮助用户识别设备。为了节省存储空间，字符串描述符通常是可选的，采用 unicode 格式编码。

22.10.7 总线速度

每个 USB 设备都必须告知主机其总线上是否有连接以及总线速度。在连接时这是通过连接到总线的 1.5 kΩ 电阻实现的。

取决于设备速度，电阻可将 D+ 或 D- 线上拉到 3.3V。对于低速设备，上拉电阻连接到 D- 线。对于全速设备，上拉电阻连接到 D+ 线。

22.10.8 分类规范和驱动程序

USB 规范包含操作系统供应商可选择支持的分类规范。类别包括诸如音频、大容量存储、通信和人机界面（Human Interface， HID）。多数情况下，主机端需要驱动程序来与 USB 设备“ 对话 ”。在定制应用中，可能需要开发驱动程序。幸运的是，对于绝大多数常见类型的设备来说，常见主机系统中均配有驱动程序。因此，这些驱动程序可以重复使用。

23.0 复位

PIC18F/LF1XK50 器件有以下几种不同类型的复位：

a) 上电复位 （POR）
b) 正常工作期间的 MCLR 复位
c) 功耗管理模式下的 MCLR 复位
d) 看门狗定时器 （WDT）复位 （执行程序期间）
e) 可编程欠压复位 （BOR）
f) RESET 指令
g) 堆栈满复位
h) 堆栈下溢复位

本节讨论了由 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 3.1.2.4 节 “ 堆栈满和下溢复位 ”** 中讨论。 WDT 复位将在**第 24.2 节 “ 看门狗定时器 （WDT） ”** 中讨论。

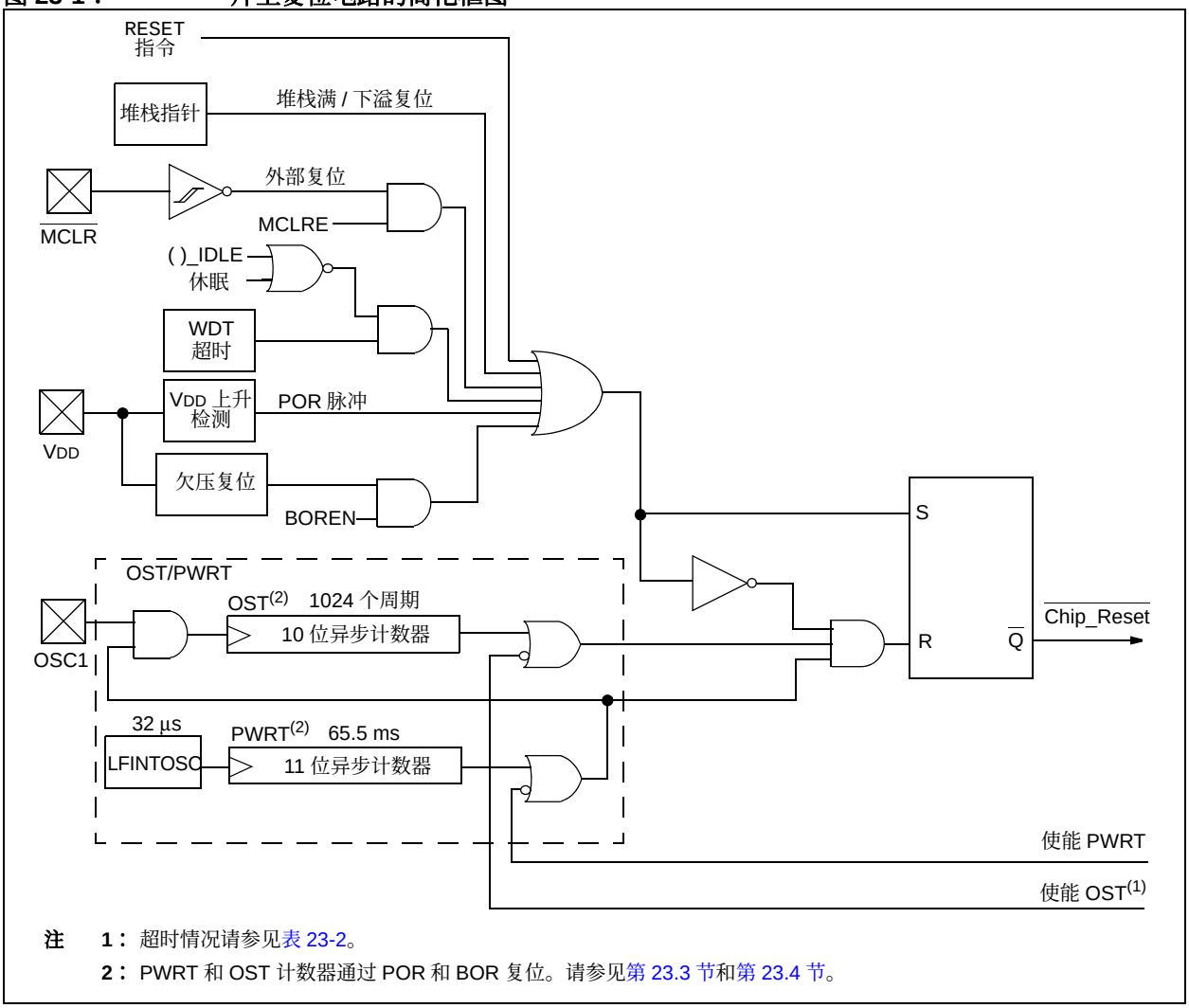
图 23-1 给出了片上复位电路的简化框图。

23.1 RCON 寄存器

通过 RCON 寄存器（寄存器 23-1）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清零，而且必须在事件发生后由应用程序将它们置 1。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 23.6 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）和对 BOR 进行软件控制的控制位 （SBOREN）。在**第 7.0 节 “ 中断 ”** 中讨论了中断优先级。在**第 23.4 节 “ 欠压复位 （BOR） ”** 中讨论了 BOR。

图 23-1： 片上复位电路的简化框图



注 1： 超时情况请参见表 23-2。
2： PWRT 和 OST 计数器通过 POR 和 BOR 复位。请参见第 23.3 节和第 23.4 节。

寄存器 23-1：　　RCON：复位控制寄存器

R/W-0	R/W-1	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	SBOREN(1)	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$(2)	$\overline{BOR}$
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

bit 6　**SBOREN**：BOR 软件使能位 (1)
如果 BOREN<1:0> = 01：
1 = 使能 BOR
0 = 禁止 BOR
如果 BOREN<1:0> = 00、10 或 11：
该位被禁止并读为 0。

bit 5　**未实现**：读为 0

bit 4　**$\overline{RI}$**：RESET 指令标志位
1 = 未执行 RESET 指令 （由固件置 1 或在上电复位时置 1）
0 = 执行了 RESET 指令，导致器件复位 （发生代码执行复位后必须由固件置 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 (2)
1 = 未发生上电复位
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 (3)
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 POR 或欠压复位后必须由固件置 1）

注　1： 如果使能了 SBOREN，其复位状态为 1 ；否则为 0。
2： $\overline{POR}$ 的实际复位值由器件复位的类型决定。更多信息，请参见该寄存器下方的 “ 注 ” 和**第 23.6 节 “ 寄存器的复位状态 ”**。
3： 请参见表 23-3。

23.2 主复位（MCLR）

MCLR 引脚提供了触发外部复位器件的方法。将该引脚拉低可以产生复位信号。这些器件在 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 MCLR 引脚驱动为低电平。

在 PIC18F/LF1XK50 器件中，可以用 MCLRE 配置位禁止 MCLR 输入。当禁止 MCLR 时，该引脚将成为一个数字输入引脚。更多信息，请参见**第 9.1 节“PORTA、TRISA 和 LATA 寄存器”**。

23.3 上电复位（POR）

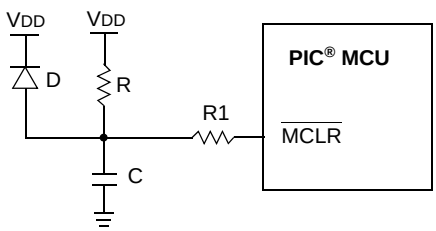
只要当 VDD 上升到某个门限时，就会在片上产生上电复位脉冲。这使得 VDD 达到满足器件正常工作的值时，器件会以初始化状态启动。

为了利用 POR 电路，需要将 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。

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。

POR 事件由 RCON 寄存器的 POR 位捕捉。每当发生 POR 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 POR 复位为 1。要捕捉多个事件，用户必须在 POR 之后用软件手动将该位设置为 1。

图 23-2： 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。

23.4 欠压复位 （BOR）

PIC18F/LF1XK50 器件带有一个 BOR 电路，它将为用户提供一系列配置和节能选项。BOR 由 CONFIG2L 配置寄存器的 BORV<1:0> 和 BOREN<1:0> 位控制。总共有 4 种 BOR 配置，归纳在表 23-1 中。

BOR 门限值由 BORV<1:0> 位设置。如果使能了 BOR（BOREN<1:0> 为除 00 以外的任何值），只要 VDD 低于 VBOR 的时间大于 TBOR，就会复位器件。如果 VDD 降到 VBOR 以下的时间小于 TBOR，器件是否发生复位不确定。芯片将保持欠压复位状态，直至 VDD 上升到 VBOR 以上。

如果使能了上电延时定时器，则它将在 VDD 上升到超过 VBOR 之后开始工作；并使芯片在延时 TPWRT 期间保持复位。如果在上电延时定时器运行过程中，VDD 电压降到 VBOR 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 VBOR 以上，上电延时定时器将重新执行延时。

BOR 和上电延时定时器 （PWRT）是分别配置的。使能 BOR 复位并不会自动使能 PWRT。

23.4.1 用软件使能 BOR

当 BOREN<1:0> = 01 时，用户可以用软件使能或禁止 BOR。这通过使用 RCON 寄存器的 SBOREN 控制位实现。如前所述，将 SBOREN 置 1 可使能 BOR。清零 SBOREN 将完全禁止 BOR。SBOREN 位只在该模式下工作；否则读为 0。

用软件控制 BOR 位可使用户能更灵活地定制应用程序以使其适应环境，而无需通过对器件重新编程来更改 BOR 配置。它还允许用户通过减少 BOR 消耗的电流，用软件调节器件的功耗。虽然 BOR 的电流通常很小，但是它可能对低功耗应用有一些影响。

注：	即使当 BOR 受软件控制时， BOR 复位电平仍将由 BORV<1:0> 配置位设置。该值不能用软件更改。

23.4.2 检测 BOR

使能 BOR 后，在发生 BOR 或 POR 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过 BOR 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 POR 事件后， $\overline{\text{POR}}$ 和 $\overline{\text{BOR}}$ 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 BOR 事件。

23.4.3 在休眠模式下禁止 BOR

当 BOREN<1:0> = 10 时，BOR 受硬件控制并且像前面描述的那样工作。每当器件进入休眠模式时，就会自动禁止 BOR。当器件返回到任何其他工作模式时，又将自动重新使能 BOR。

该模式使应用程序能在有效执行代码的同时从欠压状态恢复，这也是器件最需要 BOR 保护的状况。同时，通过消除增加的 BOR 电流，可以省去休眠模式下的额外功耗。

表 23-1： BOR 配置

BOR 配置		SBOREN 的状态（RCON<6>）	BOR 操作
BOREN1	BOREN0		
0	0	不可用	禁止 BOR ；必须对配置位重新编程才能使能 BOR。
0	1	可用	用软件使能 BOR ；工作模式由 SBOREN 控制。
1	0	不可用	用硬件在运行和空闲模式下使能 BOR，在休眠模式下禁止。
1	1	不可用	用硬件使能 BOR ；必须对配置位重新编程才能禁止 BOR。

23.5 器件复位定时器

PIC18F/LF1XK50 器件包含了三个独立的片上定时器，有助于调节上电复位过程。它们的主要功能是确保在代码执行之前器件时钟稳定。这些定时器是：

- 上电延时定时器 （PWRT）
- 振荡器起振定时器 （OST）
- PLL 锁定延时定时器

23.5.1 上电延时定时器 （PWRT）

PIC18F/LF1XK50 器件的上电延时定时器 （PWRT）是一个 11 位计数器，它使用 LFINTOSC 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。

上电延迟时间取决于LFINTOSC时钟，并且由于温度和工艺的不同，不同器件的延迟时间也将各不相同。详情请参见**第 27.0 节 “ 电气规范 ”**。

通过清零 $\overline{\text{PWRTEN}}$ 配置位可使能 PWRT。

23.5.2 振荡器起振定时器 （OST）

在 PWRT 延时结束以后，由振荡器起振定时器（OST）提供一个 1024 振荡周期 （来自 OSC1 输入）的延时，从而确保晶振或谐振器的起振和稳定工作。

只有在 XT、LP、HS 和 HSPLL 模式下，并且仅当发生上电复位或从所有功耗管理模式退出 （停止外部振荡器）时，才启动 OST 延时。

23.5.3 PLL 锁定延时定时器

当在 PLL 模式下使能 PLL 时，上电复位后的延时时序与其他振荡器模式略有不同。在 PLL 模式下需要使用一个独立的定时器来提供一段足够让PLL锁定主振荡器频率的固定延时。 PLL 锁定延时 （TPLL）通常为 2 ms，且在振荡器起振延时后发生。

23.5.4 延时时序

上电延时时序如下：

1. POR脉冲清零后，启动PWRT延时（如果使能）。
2. 然后， OST 被激活。

总延迟时间取决于振荡器配置和PWRT的状态。图23-3、图 23-4、图 23-5、图 23-6 和图 23-7 各自描述了不同的上电延时时序，其中上电延时定时器被使能，并且器件工作在HS振荡器模式下。图 23-3到图 23-6也适用于在 XT 或 LP 模式下工作的器件。对于工作在 RC 模式下的器件以及禁止了 PWRT 的器件，将根本没有延时。

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，所有延时将结束，将 $\overline{\text{MCLR}}$ 电平拉高后器件将立即开始执行程序 （图 23-5）。这对于测试或同步多个并行工作的PIC18F1XK50/PIC18LF1XK50器件来说是非常有用的。

表 23-2： 不同情形下的延时

振荡器配置	上电复位 (2) 和欠压复位		从功耗管理模式退出
	$\overline{\text{PWRTEN}}$ = 0	$\overline{\text{PWRTEN}}$ = 1	
HSPLL	66 ms(1) + 1024 TOSC + 2 ms(2)	1024 TOSC + 2 ms(2)	1024 TOSC + 2 ms(2)
HS、 XT 和 LP	66 ms(1) + 1024 TOSC	1024 TOSC	1024 TOSC
EC 和 ECIO	66 ms(1)	—	—
RC 和 RCIO	66 ms(1)	—	—
INTIO1 和 INTIO2	66 ms(1)	—	—

注 **1**：66 ms （65.5 ms）是上电延时定时器 （PWRT）延迟时间的标称值。
2：2 ms 是 PLL 锁定所需的标称时间。

图 23-3： 上电延时时序 （$\overline{MCLR}$ 连接到 VDD， VDD 电压上升时间 < TPWRT）

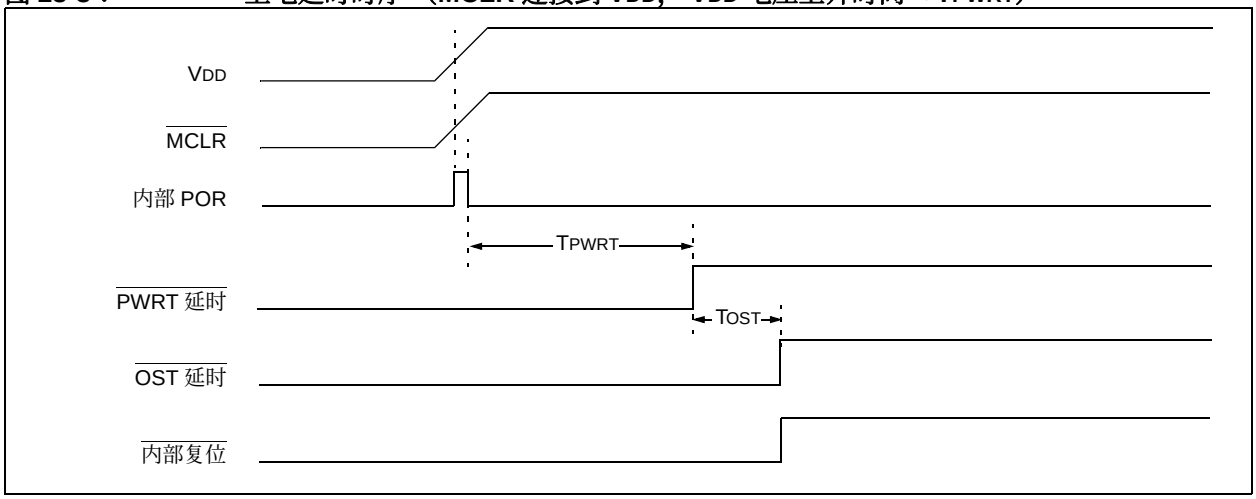


图 23-4： 上电延时时序 （$\overline{MCLR}$ 未连接到 VDD）：情形 1

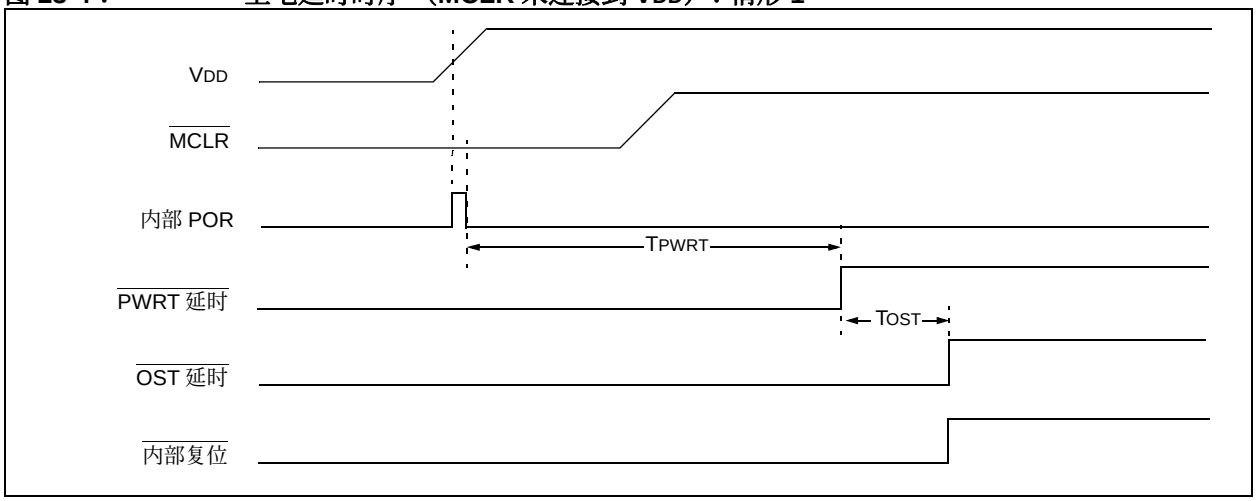


图 23-5： 上电延时时序 （$\overline{MCLR}$ 未连接到 VDD）：情形 2

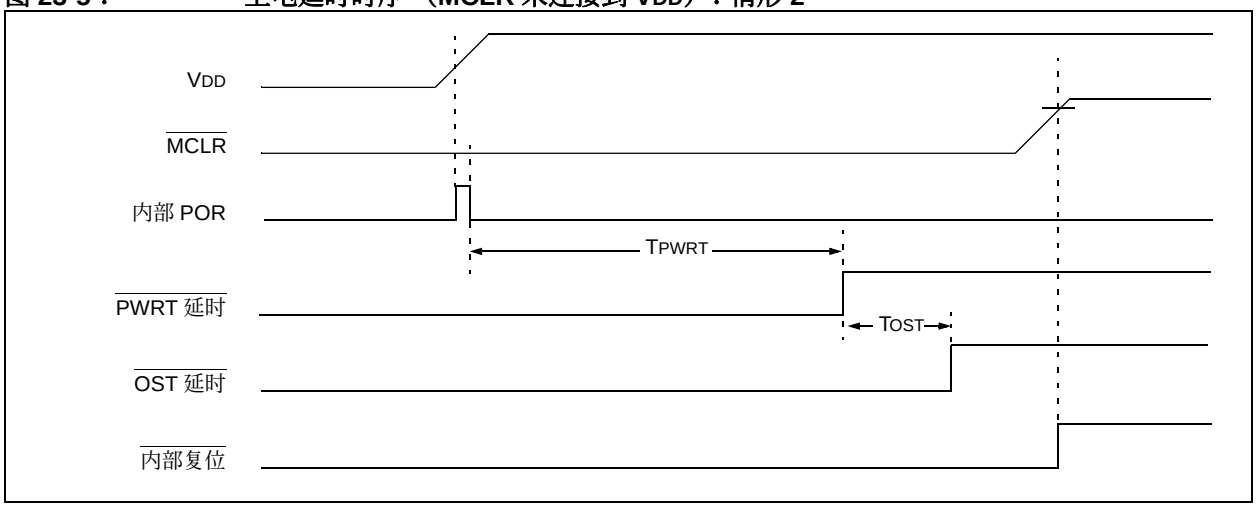


图 23-6：　　缓慢上升时间 （$\overline{\text{MCLR}}$ 连接到 VDD， VDD 电压上升时间 > TPWRT）

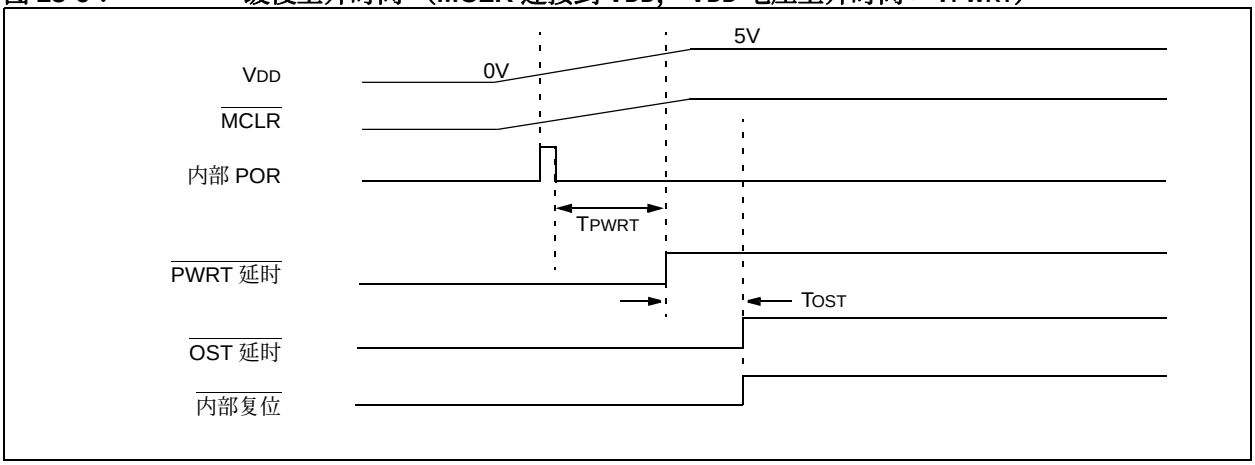
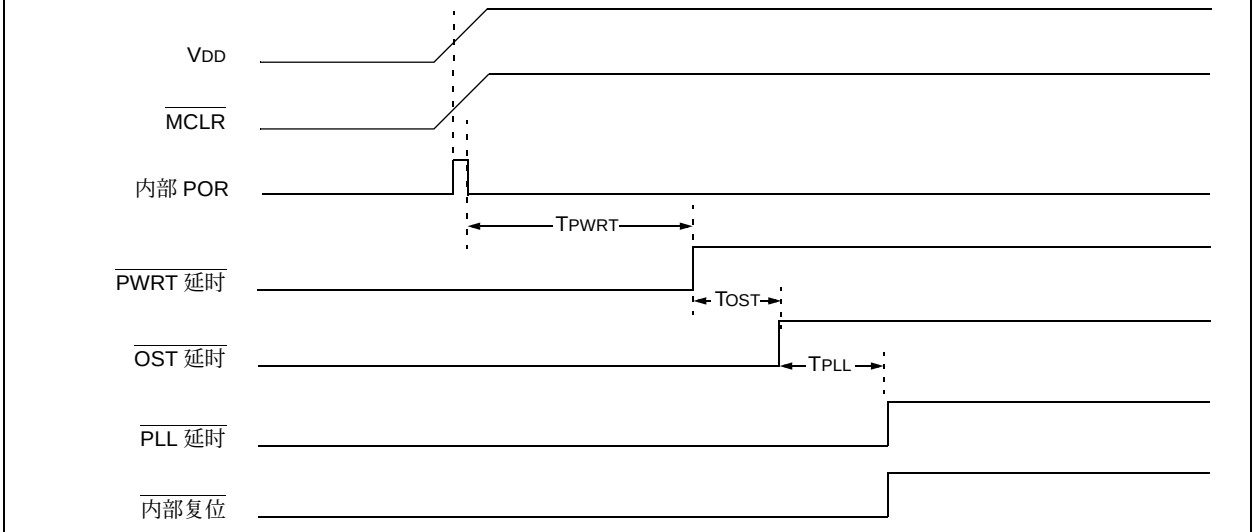


图 23-7：　　在 PLL 使能时 POR 的延时时序 （$\overline{\text{MCLR}}$ 连接到 VDD）



注：　TOST = 1024 个时钟周期。
TPLL ≈ 2 ms （最大）， PWRT 定时器延时的前三个阶段。

23.6 寄存器的复位状态

一些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继续。如表 23-3 所示，RCON 寄存器中的状态位（RI、TO、PD、POR 和 BOR）在不同的复位情形中会分别被置 1 或清零。可在软件中使用这些位判断复位的性质。

表 23-4 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 23-3： RCON 寄存器的状态位、含义以及初始化状态

条件	程序计数器	RCON 寄存器						STKPTR 寄存器	
		SBOREN	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	STKFUL	STKUNF
上电复位	0000h	1	1	1	1	0	0	0	0
RESET 指令	0000h	u(2)	0	u	u	u	u	u	u
欠压复位	0000h	u(2)	1	1	1	u	0	u	u
功耗管理运行模式下的 $\overline{MCLR}$ 复位	0000h	u(2)	u	1	u	u	u	u	u
功耗管理空闲和休眠模式下的 MCLR 复位	0000h	u(2)	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(2)	u	0	u	u	u	u	u
全功耗执行期间的 $\overline{MCLR}$ 复位	0000h	u(2)	u	u	u	u	u	u	u
堆栈满复位 （STVREN = 1）	0000h	u(2)	u	u	u	u	u	1	u
堆栈下溢复位 （STVREN = 1）	0000h	u(2)	u	u	u	u	u	u	1
堆栈下溢错误 （不是真正的复位， STVREN = 0）	0000h	u(2)	u	u	u	u	u	u	1
功耗管理空闲或休眠模式下的 WDT 超时	PC + 2	u(2)	u	0	0	u	u	u	u
通过中断从功耗管理模式退出	PC + 2(1)	u(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 位置 1 时， PC 装入中断向量 （008h 或 0018h）。

2： 当软件使能 BOR （BOREN<1:0> 配置位 = 01 且 SBOREN = 1）时， POR 的复位状态为 1 且所有其他复位不能改变该状态。否则，其复位状态为 0。

表 23-4：　所有寄存器的初始化状态

寄存器	地址	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
TOSU	FFFh	---0 0000	---0 0000	---0 uuuu(3)
TOSH	FFEh	0000 0000	0000 0000	uuuu uuuu(3)
TOSL	FFDh	0000 0000	0000 0000	uuuu uuuu(3)
STKPTR	FFCh	00-0 0000	uu-0 0000	uu-u uuuu(3)
PCLATU	FFBh	---0 0000	---0 0000	---u uuuu
PCLATH	FFAh	0000 0000	0000 0000	uuuu uuuu
PCL	FF9h	0000 0000	0000 0000	PC + 2(2)
TBLPTRU	FF8h	---0 0000	---0 0000	---u uuuu
TBLPTRH	FF7h	0000 0000	0000 0000	uuuu uuuu
TBLPTRL	FF6h	0000 0000	0000 0000	uuuu uuuu
TABLAT	FF5h	0000 0000	0000 0000	uuuu uuuu
PRODH	FF4h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODL	FF3h	xxxx xxxx	uuuu uuuu	uuuu uuuu
INTCON	FF2h	0000 000x	0000 000u	uuuu uuuu(1)
INTCON2	FF1h	1111 -1-1	1111 -1-1	uuuu -u-u(1)
INTCON3	FF0h	11-0 0-00	11-0 0-00	uu-u u-uu(1)
INDF0	FEFh	N/A	N/A	N/A
POSTINC0	FEEh	N/A	N/A	N/A
POSTDEC0	FEDh	N/A	N/A	N/A
PREINC0	FECh	N/A	N/A	N/A
PLUSW0	FEBh	N/A	N/A	N/A
FSR0H	FEAh	---- 0000	---- 0000	---- uuuu
FSR0L	FE9h	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	FE8h	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF1	FE7h	N/A	N/A	N/A
POSTINC1	FE6h	N/A	N/A	N/A
POSTDEC1	FE5h	N/A	N/A	N/A
PREINC1	FE4h	N/A	N/A	N/A
PLUSW1	FE3h	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： INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。
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： 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。
4： 具体条件下的复位值，请参见表 23-3。
5： 如果 CONFIG3H 的 PBADEN 位为 0，则 ANSELH 寄存器的所有位初始化为 0。

表 23-4：　所有寄存器的初始化状态 （续）

寄存器	地址	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	FE2h	---- 0000	---- 0000	---- uuuu
FSR1L	FE1h	xxxx xxxx	uuuu uuuu	uuuu uuuu
BSR	FE0h	---- 0000	---- 0000	---- uuuu
INDF2	FDFh	N/A	N/A	N/A
POSTINC2	FDEh	N/A	N/A	N/A
POSTDEC2	FDDh	N/A	N/A	N/A
PREINC2	FDCh	N/A	N/A	N/A
PLUSW2	FDBh	N/A	N/A	N/A
FSR2H	FDAh	---- 0000	---- 0000	---- uuuu
FSR2L	FD9h	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	FD8h	---x xxxx	---u uuuu	---u uuuu
TMR0H	FD7h	0000 0000	0000 0000	uuuu uuuu
TMR0L	FD6h	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	FD5h	1111 1111	1111 1111	uuuu uuuu
OSCCON	FD3h	0011 qq00	0011 qq00	uuuu uuuu
OSCCON2	FD2h	---- -10x	---- -10x	---- -uuu
WDTCON	FD1h	---- ---0	---- ---0	---- ---u
RCON(4)	FD0h	0q-1 11q0	0q-q qquu	uq-u qquu
TMR1H	FCFh	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	FCEh	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	FCDh	0000 0000	u0uu uuuu	uuuu uuuu
TMR2	FCCh	0000 0000	0000 0000	uuuu uuuu
PR2	FCBh	1111 1111	1111 1111	1111 1111
T2CON	FCAh	-000 0000	-000 0000	-uuu uuuu
SSPBUF	FC9h	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPADD	FC8h	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	FC7h	0000 0000	0000 0000	uuuu uuuu
SSPCON1	FC6h	0000 0000	0000 0000	uuuu uuuu
SSPCON2	FC5h	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： INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。
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： 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。
4： 具体条件下的复位值，请参见表 23-3。
5： 如果 CONFIG3H 的 PBADEN 位为 0，则 ANSELH 寄存器的所有位初始化为 0。

表 23-4：　所有寄存器的初始化状态 （续）

寄存器	地址	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
ADRESH	FC4h	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	FC3h	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	FC2h	--00 0000	--00 0000	--uu uuuu
ADCON1	FC1h	---- 0000	---- 0000	---- uuuu
ADCON2	FC0h	0-00 0000	0-00 0000	u-uu uuuu
CCPR1H	FBFh	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	FBEh	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	FBDh	0000 0000	0000 0000	uuuu uuuu
REFCON2	FBCh	---0 0000	---0 0000	---u uuuu
REFCON1	FBBh	000- 00-0	000- 00-0	uuu- uu-u
REFCON0	FBAh	0001 00--	0001 00--	uuuu uu--
PSTRCON	FB9h	---0 0001	---0 0001	---u uuuu
BAUDCON	FB8h	0100 0-00	0100 0-00	uuuu u-uu
PWM1CON	FB7h	0000 0000	0000 0000	uuuu uuuu
ECCP1AS	FB6h	0000 0000	0000 0000	uuuu uuuu
TMR3H	FB3h	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	FB2h	xxxx xxxx	uuuu uuuu	uuuu uuuu
T3CON	FB1h	0000 0000	uuuu uuuu	uuuu uuuu
SPBRGH	FB0h	0000 0000	0000 0000	uuuu uuuu
SPBRG	FAFh	0000 0000	0000 0000	uuuu uuuu
RCREG	FAEh	0000 0000	0000 0000	uuuu uuuu
TXREG	FADh	0000 0000	0000 0000	uuuu uuuu
TXSTA	FACh	0000 0010	0000 0010	uuuu uuuu
RCSTA	FABh	0000 000x	0000 000x	uuuu uuuu
EEADR	FAAh	0000 0000	0000 0000	uuuu uuuu
EEDATA	FA8h	0000 0000	0000 0000	uuuu uuuu
EECON2	FA7h	0000 0000	0000 0000	0000 0000
EECON1	FA6h	xx-0 x000	uu-0 u000	uu-0 u000

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： INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。
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： 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。
4： 具体条件下的复位值，请参见表 23-3。
5： 如果 CONFIG3H 的 PBADEN 位为 0，则 ANSELH 寄存器的所有位初始化为 0。

表 23-4：　所有寄存器的初始化状态 （续）

寄存器	地址	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
IPR2	FA2h	1111 111-	1111 111-	uuuu uuu-
PIR2	FA1h	0000 000-	0000 000-	uuuu uuu-(1)
PIE2	FA0h	0000 000-	0000 000-	uuuu uuu-
IPR1	F9Fh	-111 1111	-111 1111	-uuu uuuu
PIR1	F9Eh	-000 0000	-000 0000	-uuu uuuu(1)
PIE1	F9Dh	-000 0000	-000 0000	-uuu uuuu
OSCTUNE	F9Bh	0000 0000	0000 0000	uuuu uuuu
TRISC	F95h	1111 1111	1111 1111	uuuu uuuu
TRISB	F94h	1111 ----	1111 ----	uuuu ----
TRISA	F93h	--11 ----	--11 ----	--uu ----
LATC	F8Bh	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATB	F8Ah	xxxx ----	uuuu ----	uuuu ----
LATA	F89h	--xx ----	--uu ----	--uu ----
PORTC	F82h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTB	F81h	xxxx ----	uuuu ----	uuuu ----
PORTA	F80h	--xx x-xx	--xx x-xx	--uu u-uu
ANSELH(5)	F7Fh	---- 1111	---- 1111	---- uuuu
ANSEL	F7Eh	1111 1---	1111 1---	uuuu u---
IOCB	F7Ah	0000 ----	0000 ----	uuuu ----
IOCA	F79h	--00 0-00	--00 0-00	--uu u-uu
WPUB	F78h	1111 ----	1111 ----	uuuu ----
WPUA	F77h	--11 1---	--11 1---	--uu u---
SLRCON	F76h	---- -111	---- -111	---- -uuu
SSPMSK	F6Fh	1111 1111	1111 1111	uuuu uuuu
CM1CON0	F6Dh	0000 0000	0000 0000	uuuu uuuu
CM2CON1	F6Ch	0000 0000	0000 0000	uuuu uuuu
CM2CON0	F6Bh	0000 0000	0000 0000	uuuu uuuu
SRCON1	F69h	0000 0000	0000 0000	uuuu uuuu
SRCON0	F68h	0000 0000	0000 0000	uuuu uuuu
UCON	F64h	-0x0 000-	-0x0 000-	-uuu 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： INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。
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： 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。
4： 具体条件下的复位值，请参见表 23-3。
5： 如果 CONFIG3H 的 PBADEN 位为 0，则 ANSELH 寄存器的所有位初始化为 0。

表 23-4：　所有寄存器的初始化状态 （续）

寄存器	地址	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
USTAT	F63h	-xxx xxx-	-xxx xxx-	-uuu uuu-
UIR	F62h	-000 0000	-000 0000	-uuu uuuu
UCFG	F61h	0--0 -000	0--0 -000	u--u -uuu
UIE	F60h	-000 0000	-000 0000	-uuu uuuu
UEIR	F5Fh	0--0 0000	0--0 0000	u--u uuuu
UFRMH	F5Eh	---- -xxx	---- -xxx	---- -uuu
UFRML	F5Dh	xxxx xxxx	xxxx xxxx	uuuu uuuu
UADDR	F5Ch	-000 0000	-000 0000	-uuu uuuu
UEIE	F5Bh	0--0 0000	0--0 0000	u--u uuuu
UEP7	F5Ah	----0 0000	----0 0000	----u uuuu
UEP6	F59h	----0 0000	----0 0000	----u uuuu
UEP5	F58h	----0 0000	----0 0000	----u uuuu
UEP4	F57h	----0 0000	----0 0000	----u uuuu
UEP3	F56h	----0 0000	----0 0000	----u uuuu
UEP2	F55h	----0 0000	----0 0000	----u uuuu
UEP1	F54h	----0 0000	----0 0000	----u uuuu
UEP0	F53h	----0 0000	----0 0000	----u 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： INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。
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： 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。
4： 具体条件下的复位值，请参见表 23-3。
5： 如果 CONFIG3H 的 PBADEN 位为 0，则 ANSELH 寄存器的所有位初始化为 0。

注：

24.0 CPU 的特殊功能

PIC18F/LF1XK50 器件包含的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）
- 代码保护
- ID 存储单元
- 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 2.0 节 “ 振荡器模块 ”** 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。

除了为复位提供了上电延时定时器和振荡器起振定时器之外，PIC18F/LF1XK50 器件还提供了一个看门狗定时器，该定时器可配置成永久使能或用软件控制（如果使能位被禁止的话）。

器件自带的内部 RC 振荡器还提供了故障保护时钟监视器（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，同时主时钟源继续其起振延时。

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。

24.1 配置位

可以通过对配置位编程（读为 0）或不编程（读为 1）来选择不同的器件配置。这些配置位被映射到程序存储器以 300000h 开始的单元中。

用户会注意到地址300000h超出了用户程序存储空间范围。事实上，它属于配置存储空间（300000h-3FFFFFh），这一空间仅能通过表读和表写进行访问。

对配置寄存器进行编程的方式与对闪存进行编程的方式类似。 EECON1 寄存器中的 WR 位启动自定时写入配置寄存器。在正常操作模式下，带有指向配置寄存器的 TBLPTR 的 TBLWT 指令会建立写配置寄存器要用到的地址和数据。将 WR 位置 1 会启动对配置寄存器的长写操作。配置寄存器一次被写入一个字节。TBLWT 指令可以将 1 或 0 写入单元来改写配置单元的内容。关于闪存编程的更多详细信息，请参见**第 4.5 节 “ 写闪存程序存储器 ”**。

表 24-1： 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
300000h	CONFIG1L	—	—	USBDIV	CPUDIV1	CPUDIV0	—	—	—	--00 0---
300001h	CONFIG1H	IESO	FCMEN	PCLKEN	PLLEN	FOSC3	FOSC2	FOSC1	FOSC0	0010 0111
300002h	CONFIG2L	—	—	—	BORV1	BORV0	BOREN1	BOREN0	$\overline{\text{PWRTEN}}$	---1 1111
300003h	CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN	---1 1111
300005h	CONFIG3H	MCLRE	—	—	—	HFOFST	—	—	—	1--- 1---
300006h	CONFIG4L	$\overline{\text{BKBUG}}$(2)	ENHCPU	—	—	BBSIZ	LVP	—	STVREN	-0-- 01-1
300008h	CONFIG5L	—	—	—	—	—	—	CP1	CP0	---- --11
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	11-- ----
30000Ah	CONFIG6L	—	—	—	—	—	—	WRT1	WRT0	---- --11
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—	111- ----
30000Ch	CONFIG7L	—	—	—	—	—	—	EBTR1	EBTR0	---- --11
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	-1-- ----
3FFFFEh	DEVID1(1)	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	qqqq qqqq(1)
3FFFFFh	DEVID2(1)	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	0000 1100

图注： x = 未知， u = 不变， — = 未实现， q = 值取决于具体条件。
阴影单元未实现，读为 0。

注 1： DEVID1 值请参见寄存器 24-13。 DEVID 寄存器为只读寄存器，用户不能对其进行编程。
2： $\overline{\text{BKBUG}}$ 仅用于 ICD 器件。否则，该位未实现，读为 1。

寄存器 24-1：　　CONFIG1L：配置寄存器 1 的低字节

U-0	U-0	R/P-0	R/P-0	R/P-0	U-0	U-0	U-0
—	—	USBDIV	CPUDIV1	CPUDIV0	—	—	—
bit 7							bit 0

图注：		
R = 可读位	P = 可编程位	U = 未实现位，读为 0
-n = 未对器件编程时的值		x = 未知

bit 7-6　**未实现：**读为 0

bit 5　**USBDIV：**USB 时钟选择位
选择用于低速 USB 操作的时钟源
1 = USB 时钟来自 OSC1/OSC2 的 2 分频
0 = USB 时钟直接来自 OSC1/OSC2 振荡器模块；不分频

bit 4-3　**CPUDIV<1:0>：**CPU 系统时钟选择位
11 = CPU 系统时钟 4 分频
10 = CPU 系统时钟 3 分频
01 = CPU 系统时钟 2 分频
00 = CPU 系统时钟不分频

bit 2-0　**未实现：**读为 0

寄存器 24-2： CONFIG1H：配置寄存器 1 的高字节

R/P-0	R/P-0	R/P-1	R/P-0	R/P-0	R/P-1	R/P-1	R/P-1
IESO	FCMEN	PCLKEN	PLLEN	FOSC3	FOSC2	FOSC1	FOSC0
bit 7							bit 0

图注：		
R = 可读位	P = 可编程位	U = 未实现位，读为 0
-n = 未对器件编程时的值		x = 未知

bit 7 **IESO**：内部 / 外部振荡器切换位
1 = 使能振荡器切换模式
0 = 禁止振荡器切换模式

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 **PCLKEN**：主时钟使能位
1 = 使能主时钟
0 = 主时钟受软件控制

bit 4 **PLLEN**：4 倍频 PLL 使能位
1 = 振荡器频率 4 倍频
0 = PLL 受软件控制

bit 3-0 **FOSC<3:0>**：振荡器选择位
1111 = 外部 RC 振荡器， OSC2 用作 CLKOUT 引脚
1110 = 外部 RC 振荡器， OSC2 用作 CLKOUT 引脚
1101 = EC （低功耗）
1100 = EC， OSC2 用作 CLKOUT 引脚 （低功耗）
1011 = EC （中等功耗）
1010 = EC， OSC2 用作 CLKOUT 引脚 （中等功耗）
1001 = 内部 RC 振荡器， OSC2 用作 CLKOUT 引脚
1000 = 内部 RC 振荡器
0111 = 外部 RC 振荡器
0110 = 外部 RC 振荡器， OSC2 用作 CLKOUT 引脚
0101 = EC （高功耗）
0100 = EC， OSC2 用作 CLKOUT 引脚 （高功耗）
0011 = 外部 RC 振荡器， OSC2 用作 CLKOUT 引脚
0010 = HS 振荡器
0001 = XT 振荡器
0000 = LP 振荡器

寄存器 24-3： CONFIG2L：配置寄存器 2 的低字节

U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	BORV1(1)	BORV0(1)	BOREN1(2)	BOREN0(2)	$\overline{\text{PWRTEN}}$(2)
bit 7							bit 0

图注：		
R = 可读位	P = 可编程位	U = 未实现位，读为 0
-n = 未对器件编程时的值		x = 未知

bit 7-5 **未实现：**读为 0

bit 4-3 **BORV<1:0>：**欠压复位电压位 (1)
11 = V_{BOR} 设置为 1.9V 标称值
10 = V_{BOR} 设置为 2.2V 标称值
01 = V_{BOR} 设置为 2.7V 标称值
00 = V_{BOR} 设置为 3.0V 标称值

bit 2-1 **BOREN<1:0>：**欠压复位使能位 (2)
11 = 只能由硬件使能欠压复位 （禁止 SBOREN）
10 = 只能由硬件使能欠压复位，休眠模式下被禁止 （禁止 SBOREN）
01 = 由软件使能和控制欠压复位 （使能 SBOREN）
00 = 禁止使用硬件或软件使能欠压复位

bit 0 **$\overline{\text{PWRTEN}}$：**上电延时定时器使能位 (2)
1 = 禁止 PWRT
0 = 使能 PWRT

注 1： 请参见表 27-5 获取规范信息。
2： 上电延时定时器与欠压复位是相互独立的，可以分别控制两者的操作。

寄存器 24-4： CONFIG2H：配置寄存器 2 的高字节

U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN
bit 7							bit 0

图注：		
R = 可读位	P = 可编程位	U = 未实现位，读为 0
-n = 未对器件编程时的值		x = 未知

bit 7-5 **未实现：**读为 0

bit 4-1 **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

bit 0 **WDTEN：**看门狗定时器使能位
1 = WDT 总是使能的。 SWDTEN 位不起作用。
0 = WDT 由 WDTCON 寄存器的 SWDTEN 位控制

寄存器 24-5： CONFIG3H：配置寄存器 3 的高字节

R/P-1	U-0	U-0	U-0	R/P-1	U-0	U-0	U-0
MCLRE	—	—	—	HFOFST	—	—	—
bit 7							bit 0

图注：		
R = 可读位	P = 可编程位	U = 未实现位，读为 0
-n = 未对器件编程时的值		x = 未知

bit 7 **MCLRE：**$\overline{\text{MCLR}}$ 引脚使能位
1 = 使能 $\overline{\text{MCLR}}$ 引脚；禁止 RA3 输入引脚
0 = 使能 RA3 输入引脚；禁止 $\overline{\text{MCLR}}$

bit 6-4 **未实现：**读为 0

bit 3 **HFOFST：**HFINTOSC 快速起振位
1 = HFINTOSC 开始为 CPU 提供时钟而无需等待振荡器稳定下来
0 = 系统时钟关闭直到 HFINTOSC 稳定为止

bit 2-0 **未实现：**读为 0

寄存器 24-6： CONFIG4L：配置寄存器 4 的低字节

R/W-1(1)	R/W-0	U-0	U-0	R/P-0	R/P-1	U-0	R/P-1
$\overline{\text{BKBUG}}$	ENHCPU	—	—	BBSIZ	LVP	—	STVREN
bit 7							bit 0

图注：		
R = 可读位	P = 可编程位	U = 未实现位，读为 0
-n = 未对器件编程时的值		x = 未知

bit 7 **$\overline{\text{BKBUG}}$：**后台调试器使能位 (1)
1 = 禁止后台调试器
0 = 使能后台调试器

bit 6 **ENHCPU：**增强型 CPU 使能位
1 = 使能增强型 CPU
0 = 禁止增强型 CPU

bit 5-4 **未实现：**读为 0

bit 3 **BBSIZ：**引导区大小选择位
1 = 对于 PIC18F14K50/PIC18LF14K50 引导区大小为 2 kW（对于 PIC18F13K50/PIC18LF13K50 引导区大小为 1 kW）
0 = 对于 PIC18F14K50/PIC18LF14K50 引导区大小为 1 kW（对于 PIC18F13K50/PIC18LF13K50 引导区大小为 512W）

bit 2 **LVP：**单电源 ICSP™ 使能位
1 = 使能单电源 ICSP
0 = 禁止单电源 ICSP

bit 1 **未实现：**读为 0

bit 0 **STVREN：**堆栈满 / 下溢复位使能位
1 = 堆栈满 / 下溢导致复位
0 = 堆栈满 / 下溢不会导致复位

注 1： $\overline{\text{BKBUG}}$ 仅用于 ICD 器件。否则，该位未实现，读为 1。

寄存器 24-7：　　CONFIG5L：配置寄存器 5 的低字节

U-0	U-0	U-0	U-0	U-0	U-0	R/C-1	R/C-1
—	—	—	—	—	—	CP1	CP0
bit 7							bit 0

图注：	
R = 可读位	U = 未实现位，读为 0
-n = 未对器件编程时的值	C = 只可清零位

bit 7-2　**未实现：**读为 0

bit 1　**CP1：**代码保护位
1 = Block 1 不受代码保护
0 = Block 1 受代码保护

bit 0　**CP0：**代码保护位
1 = Block 0 不受代码保护
0 = Block 0 受代码保护

寄存器 24-8：　　CONFIG5H：配置寄存器 5 的高字节

R/C-1	R/C-1	U-0	U-0	U-0	U-0	U-0	U-0
CPD	CPB	—	—	—	—	—	—
bit 7							bit 0

图注：	
R = 可读位	U = 未实现位，读为 0
-n = 未对器件编程时的值	C = 只可清零位

bit 7　**CPD：**数据 EEPROM 代码保护位
1 = 数据 EEPROM 不受代码保护
0 = 数据 EEPROM 受代码保护

bit 6　**CPB：**引导区代码保护位
1 = 引导区不受代码保护
0 = 引导区受代码保护

bit 5-0　**未实现：**读为 0

寄存器 24-9： CONFIG6L：配置寄存器 6 的低字节

U-0	U-0	U-0	U-0	U-0	U-0	R/C-1	R/C-1
—	—	—	—	—	—	WRT1	WRT0
bit 7							bit 0

图注：	
R = 可读位	U = 未实现位，读为 0
-n = 未对器件编程时的值	C = 只可清零位

bit 7-2 **未实现：**读为 0

bit 1 **WRT1：**写保护位
1 = Block 1 不受写保护
0 = Block 1 受写保护

bit 0 **WRT0：**写保护位
1 = Block 0 不受写保护
0 = Block 0 受写保护

寄存器 24-10： CONFIG6H：配置寄存器 6 的高字节

R/C-1	R/C-1	R-1	U-0	U-0	U-0	U-0	U-0
WRTD	WRTB	WRTC(1)	—	—	—	—	—
bit 7							bit 0

图注：	
R = 可读位	U = 未实现位，读为 0
-n = 未对器件编程时的值	C = 只可清零位

bit 7 **WRTD：**数据 EEPROM 写保护位
1 = 数据 EEPROM 不受写保护
0 = 数据 EEPROM 受写保护

bit 6 **WRTB：**引导区写保护位
1 = 引导区不受写保护
0 = 引导区受写保护

bit 5 **WRTC：**配置寄存器写保护位 (1)
1 = 配置寄存器不受写保护
0 = 配置寄存器受写保护

bit 4-0 **未实现：**读为 0

注 1： 在正常执行模式下，该位是只读的；该位仅在编程模式下可写入。

寄存器 24-11：　CONFIG7L：配置寄存器 7 的低字节

U-0	U-0	U-0	U-0	U-0	U-0	R/C-1	R/C-1
—	—	—	—	—	—	EBTR1	EBTR0
bit 7							bit 0

图注：	
R = 可读位	U = 未实现位，读为 0
-n = 未对器件编程时的值	C = 只可清零位

bit 7-2　**未实现：**读为 0

bit 1　**EBTR1：**表读保护位
1 = 其他块可对 Block 1 执行表读操作
0 = 禁止其他块对 Block 1 执行表读操作

bit 0　**EBTR0：**表读保护位
1 = 其他块可对 Block 0 执行表读操作
0 = 禁止其他块对 Block 0 执行表读操作

寄存器 24-12：　CONFIG7H：配置寄存器 7 的高字节

U-0	R/C-1	U-0	U-0	U-0	U-0	U-0	U-0
—	EBTRB	—	—	—	—	—	—
bit 7							bit 0

图注：	
R = 可读位	U = 未实现位，读为 0
-n = 未对器件编程时的值	C = 只可清零位

bit 7　**未实现：**读为 0

bit 6　**EBTRB：**引导区表读保护位
1 = 其他块可对引导区执行表读操作
0 = 禁止其他块对引导区执行表读操作

bit 5-0　**未实现：**读为 0

寄存器 24-13： DEVID1：PIC18F1XK50/PIC18LF1XK50 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 = 可读位	U = 未实现位，读为 0
-n = 未对器件编程时的值	C = 只可清零位

bit 7-5 **DEV<2:0>：**器件 ID 位
010 = PIC18F13K50
011 = PIC18F14K50

bit 4-0 **REV<4:0>：**版本 ID 位
这些位用于表明器件版本。

寄存器 24-14： DEVID2：PIC18F1XK50/PIC18LF1XK50 器件的器件 ID 寄存器 2

R	R	R	R	R	R	R	R
DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3
bit 7							bit 0

图注：	
R = 可读位	U = 未实现位，读为 0
-n = 未对器件编程时的值	C = 只可清零位

bit 7-0 **DEV<10:3>：**器件 ID 位
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。
0010 0000 = PIC18F1XK50/PIC18LF1XK50 器件

注 1： DEV<10:3> 的值可能与其他器件相同。特定器件总是通过使用整个 DEV<10:0> 位序列来标识的。

24.2 看门狗定时器 （WDT）

PIC18F/LF1XK50 器件的 WDT 是由 LFINTOSC 时钟源驱动的。当使能 WDT 时，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 LFINTOSC 振荡器相同。

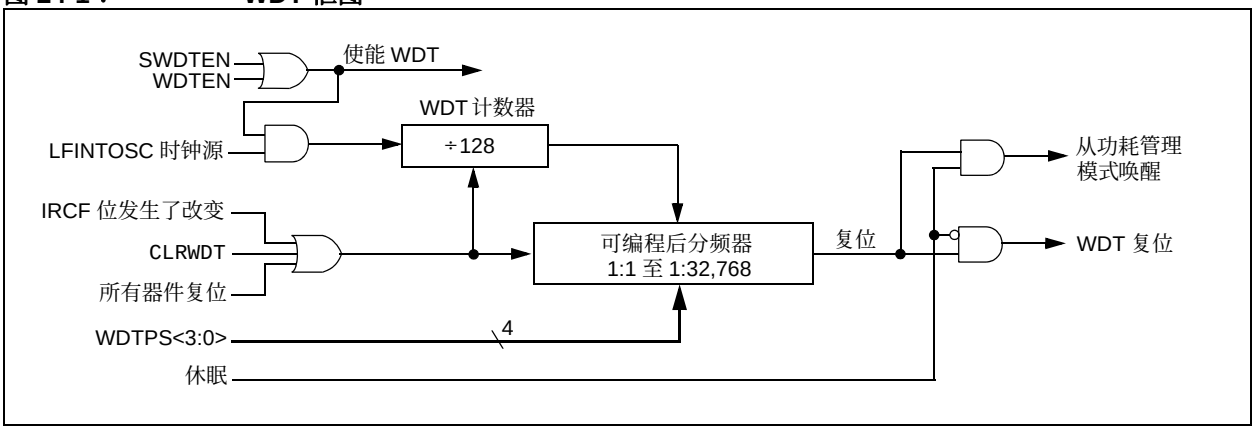
4 ms 的 WDT 定时周期将与 16 位后分频器的值相乘来得到更长的时间周期。通过配置寄存器 2H 中的位来控制一个多路开关以对 WDT 后分频器的输出进行选择。因此可获得的定时周期范围为 4 ms 至 131.072 秒（2.18 分钟）。当发生以下任一事件时，WDT 和后分频器将被清零，这些事件包括：执行了 SLEEP 或 CLRWDT 指令、改变了 OSCCON 寄存器的 IRCF 位或发生了时钟故障。

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： 更改 OSCCON 寄存器的 IRCF 位的设置会清零 WDT 和后分频器的计数值。

3： 当执行 CLRWDT 指令时，后分频器的计数值将被清零。

图 24-1： WDT 框图



24.2.1 控制寄存器

寄存器 24-15 所示为 WDTCON 寄存器。它是可读写寄存器并包含一个控制位，仅当 WDT 被配置为禁止时，该控制位才允许使用软件改写 WDT 使能配置位。

寄存器 24-15： WDTCON：看门狗定时器控制寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	SWDTEN(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 7-1 **未实现：**读为 0

bit 0 **SWDTEN：**软件使能或禁止看门狗定时器位 (1)
1 = WDT 开启
0 = WDT 关闭 （复位值）

注 1：当使能 WDTEN 配置位时该位不起作用。

表 24-2： 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	SBOREN	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$	278
WDTCON	—	—	—	—	—	—	—	SWDTEN	286
CONFIG2H				WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN	296

图注： — = 未实现，读为 0。看门狗定时器不使用阴影单元。

24.3 程序校验和代码保护

PIC18 闪存器件的整个代码保护结构与以往的 PIC® 单片机器件完全不同。

用户程序存储器被分成了5个存储区。其中一个为0.5 KB或 2 KB 的引导区，取决于具体器件。存储器的剩余部分按二进制边界被分为单独的存储区。

每个存储区都有与其相关的 3 个代码保护位。它们是：

- 代码保护位 （CPn）
- 写保护位 （WRTn）
- 外部存储区表读位 （EBTRn）

图 24-2 给出了 8 KB、 16 KB 和 32 KB 器件的程序存储器构成以及与每个存储区相关的特定代码保护位。表 24-3 中总结了这些位的实际地址。

图 24-2： PIC18F/LF1XK50 的受代码保护的程序存储器

<table>
<tr><td rowspan="3">地址 （起始 / 终止）</td><td colspan="4">器件</td></tr>
<tr><td colspan="2">14K50</td><td colspan="2">13K50</td></tr>
<tr><td>BBSIZ = 1</td><td>BBSIZ = 0</td><td>BBSIZ = 1</td><td>BBSIZ = 0</td></tr>
<tr><td>0000h
01FFh</td><td rowspan="4">引导区
2 KW CPB,
WRTB, EBTRB</td><td rowspan="2">引导区
1 KW CPB,
WRTB, EBTRB</td><td rowspan="2">引导区
1 KW CPB,
WRTB, EBTRB</td><td>引导区
0.512 KW CPB,
WRTB, EBTRB</td></tr>
<tr><td>0200h
03FFh</td><td rowspan="3">Block 0
1.512 KW
CP0, WRT0, EBTR0</td></tr>
<tr><td>0400h
05FFh</td><td rowspan="3">Block 0
3 KW
CP0, WRT0, EBTR0</td><td rowspan="2">Block 0
1 KW
CP0, WRT0, EBTR0</td></tr>
<tr><td>0600h
07FFh</td></tr>
<tr><td>0800h
0FFFh</td><td>Block 0
2 KW
CP0, WRT0, EBTR0</td><td>Block 1
2 KW
CP1, WRT1, EBTR1</td><td>Block 1
2 KW
CP1, WRT1, EBTR1</td></tr>
<tr><td>1000h
1FFFh</td><td>Block 1
4 KW
CP1, WRT1, EBTR1</td><td>Block 1
4 KW
CP1, WRT1, EBTR1</td><td rowspan="17">读为全 0</td><td rowspan="17">读为全 0</td></tr>
<tr><td>2000h
27FFh</td><td rowspan="16">读为全 0</td><td rowspan="16">读为全 0</td></tr>
<tr><td>2800h
2FFFh</td></tr>
<tr><td>3000h
37FFh</td></tr>
<tr><td>3800h
3FFFh</td></tr>
<tr><td>4000h
47FFh</td></tr>
<tr><td>4800h
4FFFh</td></tr>
<tr><td>5000h
57FFh</td></tr>
<tr><td>5800h
5FFFh</td></tr>
<tr><td>6000h
67FFh</td></tr>
<tr><td>6800h
6FFFh</td></tr>
<tr><td>7000h
77FFh</td></tr>
<tr><td>7800h
7FFFh</td></tr>
<tr><td>8000h
FFFFh</td></tr>
</table>

注： 关于测试存储器映射的要求，请参见测试章节。

表 24-3：　　代码保护寄存器汇总

寄存器名称		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
300008h	CONFIG5L	—	—	—	—	—	—	CP1	CP0
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—
30000Ah	CONFIG6L	—	—	—	—	—	—	WRT1	WRT0
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—
30000Ch	CONFIG7L	—	—	—	—	—	—	EBTR1	EBTR0
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—

图注：　阴影单元未实现。

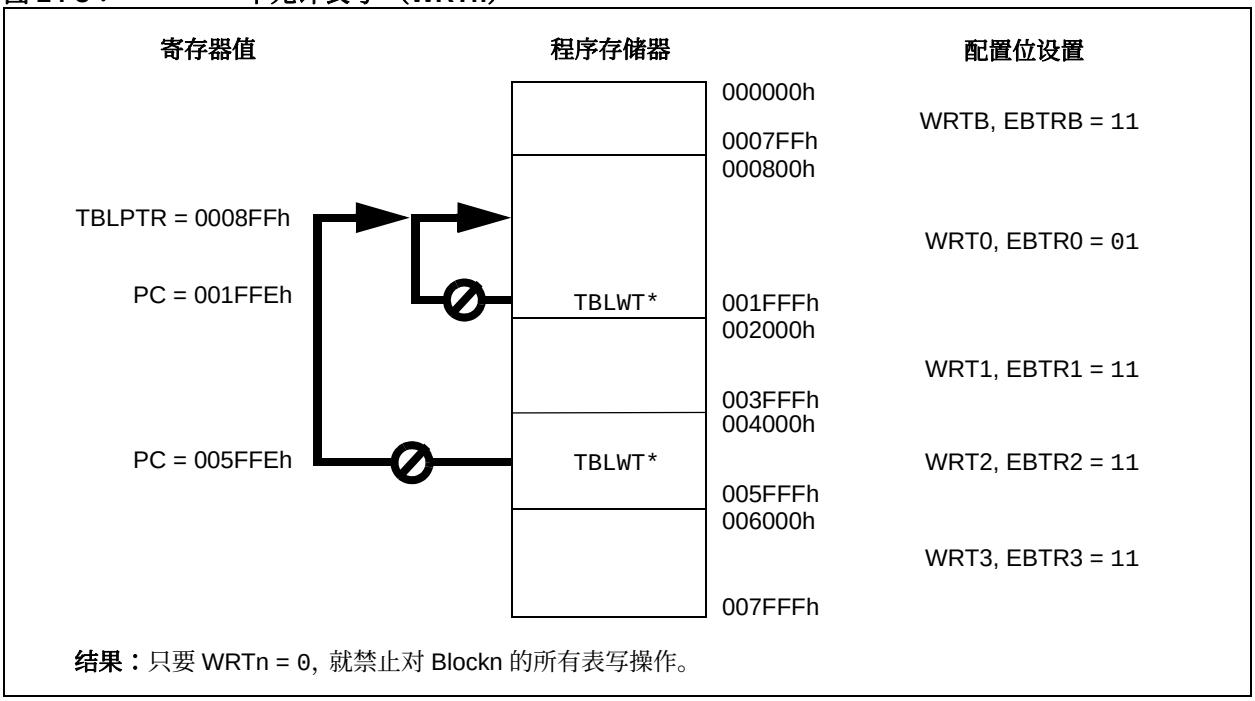
24.3.1　程序存储器代码保护

可使用表读和表写指令从任何存储单元读写程序存储器。器件 ID 可以通过表读指令进行读取。配置寄存器可以通过表读和表写指令进行读写操作。

在正常执行模式下，CPn 位不起任何作用。CPn 位禁止外部读写。如果 WRTn 配置位为 0，则用户存储器的存储区可被保护不受表写指令的影响。EBTRn 位控制表读操作。对于 EBTRn 位清零的用户存储器的存储区，允许在该存储区内执行表读指令。存储区以外的存储单元执行的表读指令则不被允许，将导致读为 0。图 24-3 到 24-5 给出了表写和表读保护的图示。

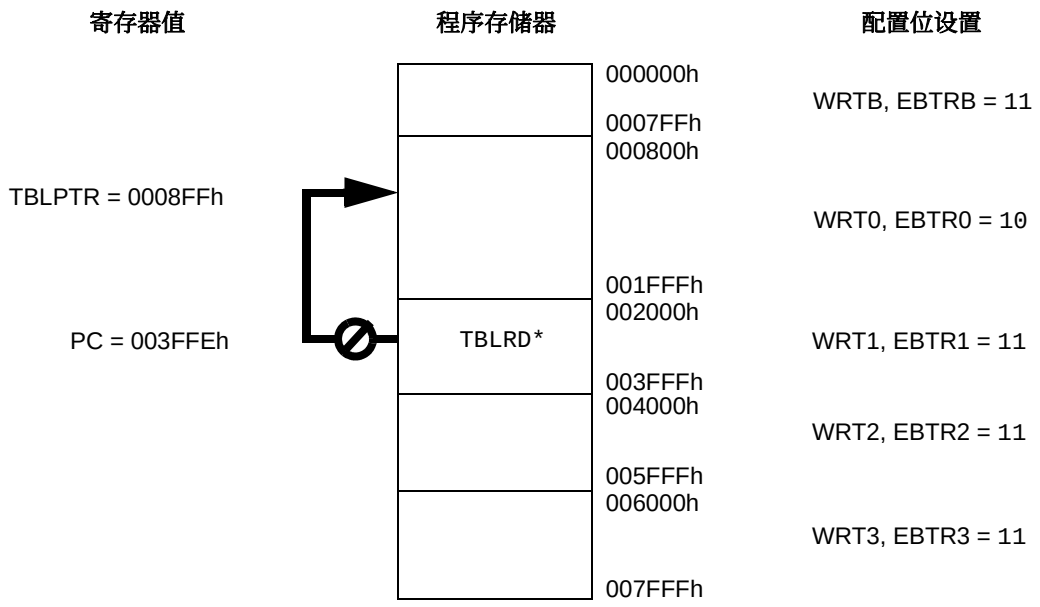
> **注：**　代码保护位只能从 1 改写到 0 状态。不可能将处于 0 状态的位改写为 1。只有通过整片擦除或块擦除功能才能将代码保护位设置为1。整片擦除和块擦除功能只能通过 ICSP 或外部编程器启动。

图 24-3：　　不允许表写 （WRTn）



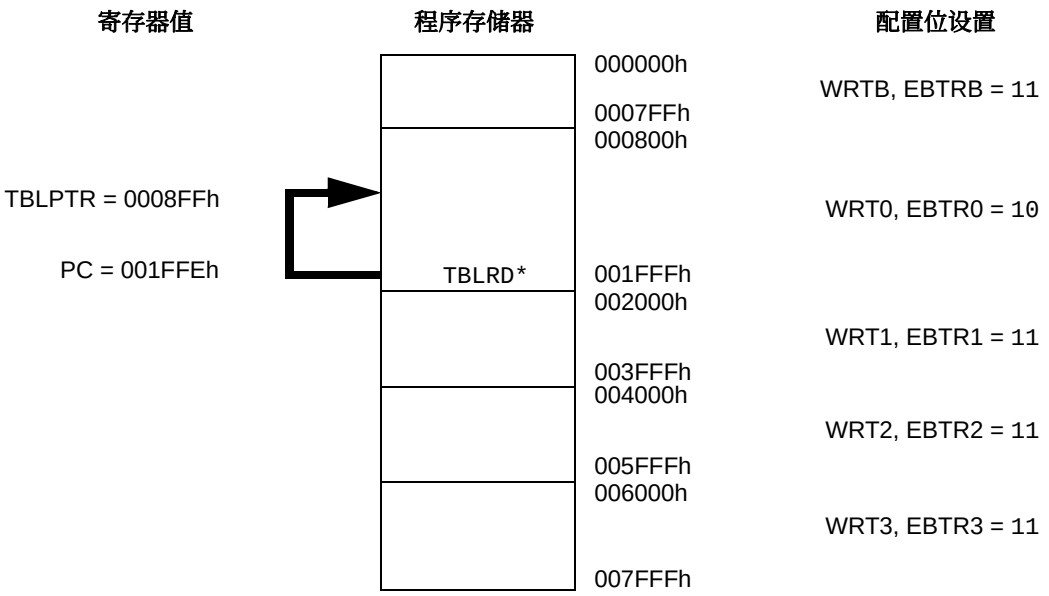
结果：只要 WRTn = 0，就禁止对 Blockn 的所有表写操作。

图 24-4： 不允许外部存储区的表读 （EBTRn）



结果： 只要 EBTRn = 0，就禁止来自外部存储区对 Blockn 的所有表读操作。
TABLAT 寄存器返回 0。

图 24-5： 允许外部存储区的表读 （EBTRn）



结果： 即使当 EBTRBn = 0 时，也允许在 Blockn 内执行表读操作。
TABLAT 寄存器返回 TBLPTR 指向的存储单元的数据值。

24.3.2 数据 EEPROM 代码保护

整个数据 EEPROM 受外部读写的保护，这通过 CPD 和 WRTD 两个位来实现。CPD 禁止数据 EEPROM 的外部读写。 WRTD 禁止从内部和外部写数据 EEPROM。在正常操作下， CPU 可以始终读数据 EEPROM，与保护位的设置无关。

24.3.3 配置寄存器保护

配置寄存器可以是写保护的。WRTC 位控制配置寄存器的保护。在正常执行模式下， WRTC 位是只读的。WRTC 只能通过 ICSP 或外部编程器写入。

24.4 ID 存储单元

有 8 个存储单元（200000h-200007h）被指定为 ID 单元，供用户存储校验和或其他代码标识。在执行程序时可通过 TBLRD 和 TBLWT 指令读写这些单元；在编程/验证时，也可读写这些单元。当器件有代码保护时，也可读取 ID 单元。

24.5 在线串行编程

PIC18F/LF1XK50 器件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。

24.6 在线调试器

将 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 24-4 给出了后台调试器所需的资源。

表 24-4： 调试器资源

I/O 引脚：	RA0 和 RA1
堆栈：	2 级深度
程序存储器：	512 字节
数据存储器：	10 字节

要使用单片机的在线调试功能，设计必须实现在线串行编程到以下引脚的连接：

- $\overline{MCLR}$/VPP/RA3
- VDD
- VSS
- RA0
- RA1

从而为 Microchip 或第三方开发工具公司提供的在线调试器模块提供交互的接口。

24.7 单电源 ICSP 编程

LVP 配置位使能单电源 ICSP 编程（原来称为低压 ICSP 编程或 LVP）。当使能单电源编程时，单片机可以在无需对 MCLR/VPP/RA3 引脚施加高压的情况下进行编程，但 RC3/PGM 引脚专用于控制编程模式进入，不能再用作通用 I/O 引脚。

使用单电源编程时， VDD 被施加到 $\overline{MCLR}$/VPP/RA3 引脚，如同在正常执行模式下一样。要进入编程模式，VDD 被施加到 PGM 引脚。

注 1： 通过将 VIHH 施加到 $\overline{MCLR}$ 引脚，就可以进行高压编程，与 LVP 位或 PGM 引脚的状态无关。

2： 默认情况下，未编程器件 （如 Microchip 提供的）和已擦除器件均使能了单电源 ICSP 编程。

3： 当使能单电源编程时，RC3 引脚不能再用作通用 I/O 引脚。

4： 当使能 LVP 时，从外部将 PGM 引脚拉至 VSS 可以允许正常程序执行。

如果不再使用单电源 ICSP 编程模式，则 LVP 位可以被清零。 RC3/PGM 随后可作为数字 I/O 引脚 RC3。LVP 位仅在使用标准高压编程 （VIHH 被施加到 MCLR/VPP/RA3 引脚）时可被置 1 或清零。一旦 LVP 被禁止，只能使用标准高压编程来对器件进行编程。

不受代码保护的存储器可以使用块擦除或逐行擦除进行擦除，然后在任何指定的 VDD 下进行写入。如果要擦除受代码保护的存储器，需要进行块擦除。

注：

25.0 指令集汇总

PIC18F/LF1XK50 器件具有一个含有 75 条 PIC18 内核指令的标准指令集，和一个含有优化递归或软件堆栈代码的 8 条新指令的扩展指令集。本章后面的部分将讨论扩展指令集。

25.1 标准指令集

标准的 PIC18 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 25-2 为 PIC18 指令集汇总，列出了上述四类指令。表 25-1 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. 被访问存储器 （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. 被访问存储器 （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寄存器编号。

立即数操作类指令使用以下操作数：

- 要装入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，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 25-1 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 25-2）列出了可被 Microchip 汇编器（MPASM™）识别的标准指令。

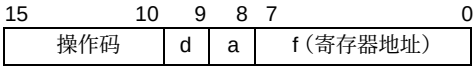
第 25.1.1 节 “ 标准指令集 ” 中对每条指令进行了介绍。

表 25-1： 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），或 CALL/BRANCH 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将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。
$\overline{TO}$	超时溢出位。
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）。汇编器将产生 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 指定的寄存器中的位 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）。

图 25-1：　　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

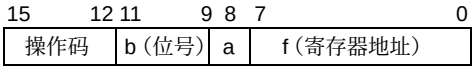
d = 0 表示结果存入 WREG 寄存器
d = 1 表示结果存入文件寄存器 (f)
a = 0 强制使用快速操作存储区
a = 1 根据 BSR 选择存储区
f = 8 位文件寄存器地址

字节到字节的传送操作（双字）

MOVFF MYREG1, MYREG2

f = 12 位文件寄存器地址

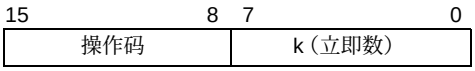
针对位的文件寄存器操作



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

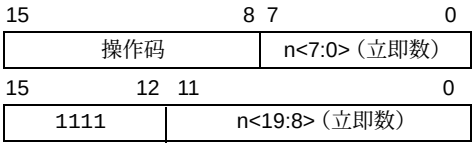


MOVLW 7Fh

k = 8 位立即数的值

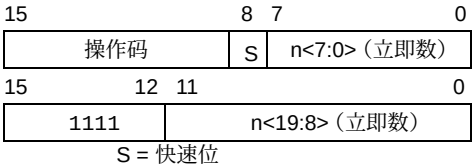
控制操作

CALL、GOTO 和跳转类操作指令



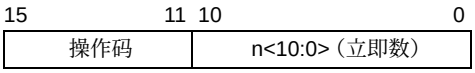
GOTO Label

n = 20 位立即数的值

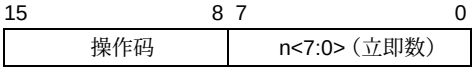


CALL MYFUNC

S = 快速位



BRA MYFUNC



BC MYFUNC

表 25-2： PIC18FXXXX 指令集

助记符，操作数		说明	周期数	16 位指令字				受影响的状态位	注
				MSb			LSb		
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	传送 f_s（源）到 第一个字	2	1100	ffff	ffff	ffff	无	
		f_d（目标）第二个字		1111	ffff	ffff	ffff		
MOVWF	f, a	将 WREG 移入 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： 端口寄存器的值随端口状态的变化而修改（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 25-2： PIC18FXXXX 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, d, 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		用软件使器件复位	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： 端口寄存器的值随端口状态的变化而修改 （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 25-2： PIC18FXXXX 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	2	1110	1110	00ff	kkkk	无	
		到 FSR （f） 第一个字		1111	0000	kkkk	kkkk		
MOVLB	k	将立即数移入 BSR<3:0>	1	0000	0001	0000	kkkk	无	
MOVLW	k	将立即数移入 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： 端口寄存器的值随端口状态的变化而修改 （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。

25.1.1 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) → 目标

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（默认）。
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（默认）。
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 25.2.3 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，那么指令语法将变为：{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) → 目标

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（默认）。
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 25.2.3 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) → 目标

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（默认）。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（默认）。
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 25.2.3 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（默认）。
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 25.2.3 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（默认）。
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 25.2.3 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，丢弃下一条指令（执行当前指令期间取的指令）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（默认）。
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 25.2.3 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： 如果跳过的指令后面跟有 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

如果跳过的指令后面跟有 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，丢弃下一条指令（执行当前指令期间取的指令）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（默认）。
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 25.2.3 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： 如果跳过的指令后面跟有 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

如果跳过的指令后面跟有 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（默认）。
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 25.2.3 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	110s	$k_7$kkk	kkkk_0
第二个字 （k<19:8>）	1111	k_{19}kkk	kkkk	kkkk_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（默认）。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（默认）。
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 25.2.3 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 → 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
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： $(\bar{f}) \rightarrow$ 目标

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（默认）。
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（默认）。
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 25.2.3 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（默认）。
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 25.2.3 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： 如果跳过的指令后面跟有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

如果跳过的指令后面跟有 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（默认）。
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 25.2.3 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： 如果跳过的指令后面跟有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

如果跳过的指令后面跟有 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（默认）。

指令字数：　1

指令周期数：　1 （2）
注： 如果跳过的指令后面跟有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

如果跳过的指令后面跟有 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> + DC > 9] 或 [C = 1]，则
(W<7:4>) + 6 + DC → W<7:4> ；
否则
(W<7:4>) + DC → 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 → 目标

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 （默认）。
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（默认）。
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 25.2.3 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 → 目标，
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（默认）。
如果结果为 0，则丢弃已取的指令转而执行一条 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（默认）。
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 25.2.3 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： 如果跳过的指令后面跟有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

如果跳过的指令后面跟有 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 → 目标，
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（默认）。
如果结果不为 0，则丢弃已取的指令转而执行一条 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（默认）。
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 25.2.3 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： 如果跳过的指令后面跟有 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

如果跳过的指令后面跟有 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 → 目标

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 （默认）。
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（默认）。
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 25.2.3 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 → 目标，
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 f （默认）。
如果结果为 0，则丢弃已取的指令转而执行一条 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（默认）。
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 25.2.3 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： 如果跳过的指令后面跟有 2 字指令，则执行 INFSNZ 需要 3 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

如果跳过的指令后面跟有 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 → 目标，
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 （默认）。
如果结果不为 0，则丢弃已取的指令转而执行一条 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（默认）。
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 25.2.3 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： 如果跳过的指令后面跟有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

如果跳过的指令后面跟有 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) → 目标

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（默认）。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（默认）。
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 25.2.3 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 **移动 f**

语法： MOVF f {,d {,a}}

操作数： 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： f → 目标

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 的内容移入目标寄存器。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（默认）。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（默认）。
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 25.2.3 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 **将源寄存器的内容移入目标寄存器**

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 的内容移入目标寄存器 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移入外设寄存器 （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 （3）

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 无效读取	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 **将立即数移入 BSR 的低半字节**

语法： MOVLB 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 **将立即数移入 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 的内容移入 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 寄存器中的数据移入寄存器 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（默认）。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 25.2.3 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（默认）。
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 25.2.3 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 ≤ f ≤ 255
a ∈ [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（默认）。
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 25.2.3 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) → 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）压入返回堆栈。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 （默认）。

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 **将立即数返回给 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。高位地址锁存器 （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 （默认）。

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]

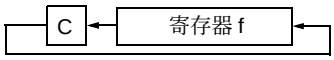
操作： (f<n>) → dest<n + 1>,
(f<7>) → C,
(C) → dest<0>

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 （默认）。
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（默认）。
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 25.2.3 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]

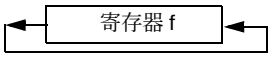
操作： (f<n>) → dest<n + 1>,
(f<7>) → dest<0>

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 （默认）。
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（默认）。
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 25.2.3 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]

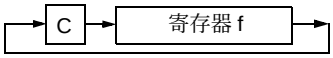
操作： (f<n>) → dest<n – 1>,
(f<0>) → C,
(C) → dest<7>

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 （默认）。
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（默认）。
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 25.2.3 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]

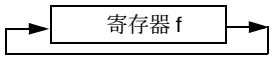
操作： (f<n>) → dest<n – 1>,
(f<0>) → dest<7>

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 （默认）。
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 （默认）。
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 25.2.3 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（默认）。
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 25.2.3 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 → 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) – ($\overline{C}$) → 目标

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（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（默认）。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（默认）。
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 25.2.3 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) → 目标

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（默认）。
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（默认）。
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 25.2.3 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) – (C) → 目标

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（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 （默认）。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（默认）。
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 25.2.3 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 （默认）。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（默认）。
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 25.2.3 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 *,
(程序存储器 (TBLPTR)) → TABLAT ；
TBLPTR 不改变；
如果执行 TBLRD *+,
(程序存储器 (TBLPTR)) → TABLAT ；
(TBLPTR) + 1 → TBLPTR ；
如果执行 TBLRD *-,
(程序存储器 (TBLPTR)) → TABLAT ；
(TBLPTR) – 1 → TBLPTR ；
如果执行 TBLRD +*,
(TBLPTR) + 1 → TBLPTR ；
(程序存储器 (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器（P.M.）的内容。使用表指针 （TBLPTR）对程序存储器进行寻址。
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： 程序存储器字的低有效字节
TBLPTR[0] = 1： 程序存储器字的高有效字节

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 BLWT*,
(TABLAT) → 保持寄存器；
TBLPTR 不改变；
如果执行 BLWT*+,
(TABLAT) → 保持寄存器；
(TBLPTR) + 1 → TBLPTR ；
如果执行 BLWT*-,
(TABLAT) → 保持寄存器；
(TBLPTR) – 1 → TBLPTR ；
如果执行 BLWT+*,
(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器（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 4.0 节 “ 闪存程序存储器 ”**。）
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： 程序存储器字的低有效字节
TBLPTR[0] = 1： 程序存储器字的高有效字节

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（默认）。
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 25.2.3 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： 如果跳过的指令后面跟有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

如果跳过的指令后面跟有 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 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (W) .XOR.(f) → 目标

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（默认）。
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（默认）。
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 25.2.3 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

25.2 扩展指令集

除了 PIC18 指令集的 75 条标准指令之外，PIC18F/LF1XK50 器件还提供了针对内核 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，它们可以实现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是禁止的。用户必须通过将 XINST 配置位置 1，才能使能它们。

扩展指令集中的指令可以全部被归为立即数操作类指令，它们既可以控制文件选择寄存器，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，它们使用户能更有效地用高级语言对数据结构执行特定的操作。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控制
- 对软件堆栈中的变量进行控制

表 25-3 提供了扩展指令集中的指令汇总。**第 25.2.2 节 “扩展指令集”** 对这些指令进行了详细说明。表 25-1（第 310 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器中直接使用这些指令。对于那些查看编译器生成代码的用户，这些命令的语法可作为参考。

25.2.1 扩展指令的语法

大部分扩展指令都使用变址参数，同时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。这时表示此参数用作变址地址或偏移量。如果 MPASM™ 汇编器发现一个变址地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，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 25.2.3.1 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 25-3： 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）传送到（第一个字） f_d（目标）（第二个字）	2	1110 1111	1011 ffff	0zzz ffff	zzzz ffff	无
MOVSS	z_s, z_d	将 z_s（源）传送到（第一个字） z_d（目标）（第二个字）	2	1110 1111	1011 xxxx	1zzz xzzz	zzzz zzzz	无
PUSHL	k	将立即数保存在 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

25.2.2 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 的内容并返回**

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，执行一条 RETURN 指令。
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，那么指令语法将变为：{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移入 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) → 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： 将源寄存器的内容移入目标寄存器 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 无效读取	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 将源寄存器的内容移入目标寄存器。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 = 11h
86h 单元的内容 = 11h

执行指令后
FSR2 = 80h
85h 单元的内容 = 11h
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 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： FSR(f) – 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，执行一条 RETURN 指令。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)

25.2.3 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 3.5.1 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单元（a = 0），或由 BSR 指定的 GPR 存储区中的单元 （a = 1）。当使能扩展指令集且 a = 0 时，地址为 5Fh 或以下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 PIC18 内核指令，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 25.2.3.1 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。

下面是在立即数变址寻址模式中，一些针对字节和位的指令的示例，通过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。

25.2.3.1 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，编译器将括号中的值解析为变址地址或偏移量。省略括号，或在括号内使用大于 5Fh 的值会在 MPASM™ 汇编器中产生错误。

如果变址参数已被正确加上了括号，那么就不再需要指定快速操作 RAM 参数；此参数被假定为 0。这与标准操作 （禁止扩展指令集时）刚好相反，在标准操作中，根据目标地址对 “ a ” 进行设置。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。

25.2.4 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，所以应用程序会读或写错误的地址。

将应用程序移植到 PIC18F/LF1XK50 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地很好，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) → 目标

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（默认）。

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

25.2.5 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 PIC18F/LF1XK50 系列器件的扩展指令集。包括 MPLAB® C18 C 编译器、 MPASM 汇编器和 MPLAB 集成开发环境（Integrated Development Environment， IDE）。

在选择了使用软件开发的目标器件后， 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 0，禁止扩展指令集和立即数变址寻址模式。在编程过程中必须将 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信息。

26.0 开发支持

一系列软件及硬件开发工具对PIC® 单片机和dsPIC® 数字信号控制器提供支持：

- 集成开发环境
  - MPLAB® IDE 软件
- 编译器 / 汇编器 / 链接器
  - 适用于各种器件系列的 MPLAB C 编译器
  - 适用于各种器件系列的 HI-TECH C 编译器
  - MPASM™ 汇编器
  - MPLINK™ 目标链接器 / MPLIB™ 目标库管理器
  - 适用于各种器件系列的 MPLAB 汇编器 / 链接器 / 库管理器
- 模拟器
  - MPLAB SIM 软件模拟器
- 仿真器
  - MPLAB REAL ICE™ 在线仿真器
- 在线调试器
  - MPLAB ICD 3
  - PICkit™ 3 Debug Express
- 器件编程器
  - PICkit™ 2 编程器
  - MPLAB PM3 器件编程器
- 低成本演示 / 开发板、评估工具包及入门工具包

26.1 MPLAB 集成开发环境软件

MPLAB IDE 软件为8/16/32 位单片机市场提供了前所未有的易于使用的软件开发平台。 MPLAB IDE 是基于Windows® 操作系统的应用软件，包括：

- 一个包含所有调试工具的图形界面
  - 模拟器
  - 编程器 （单独销售）
  - 在线仿真器 （单独销售）
  - 在线调试器 （单独销售）
- 具有彩色上下文代码显示的全功能编辑器
- 多项目管理器
- 内容可直接编辑的可定制式数据窗口
- 高级源代码调试
- 鼠标停留在变量上进行查看的功能
- 将变量从源代码窗口拖放到 Watch （观察）窗口
- 丰富的在线帮助
- 集成了可选的第三方工具，如 IAR C 编译器

MPLAB IDE 可以让您：

- 编辑源文件 （C 语言或汇编语言）
- 点击一次即可完成编译或汇编，并将代码下载到仿真器和模拟器工具中 （自动更新所有项目信息）
- 可使用如下各项进行调试：
  - 源文件 （C 语言或汇编语言）
  - 混合 C 语言和汇编语言
  - 机器码

MPLAB IDE 在单个开发范例中支持使用多种调试工具，包括从成本效益高的模拟器到低成本的在线调试器，再到全功能的仿真器。这样缩短了用户升级到更加灵活而功能强大的工具时的学习时间。

26.2 适用于各种器件系列的 MPLAB C 编译器

MPLAB C 编译器代码开发系统是完全的 ANSI C 编译器，适用于 Microchip 的 PIC18、PIC24 和 PIC32 系列单片机及 dsPIC30 和 dsPIC33 系列数字信号控制器。这些编译器提供强大的集成功能和出众的代码优化能力，且使用方便。

为便于源代码调试，编译器提供针对 MPLAB IDE 调试器优化的符号信息。

26.3 适用于各种器件系列的 HI-TECH C 编译器

HI-TECH C 编译器代码开发系统是完全的 ANSI C 编译器，适用于 Microchip 的 PIC 系列单片机及 dsPIC 系列数字信号控制器。这些编译器提供强大的集成功能和全知代码生成能力，且使用方便。

为便于源代码调试，编译器提供针对 MPLAB IDE 调试器优化的符号信息。

编译器包括一个宏汇编器、链接器、预处理程序和单步驱动程序，可以在多种平台上运行。

26.4 MPASM 汇编器

MPASM 汇编器是全功能通用宏汇编器，适用于 PIC10/12/16/18 MCU。

MPASM 汇编器可生成用于 MPLINK 目标链接器的可重定位目标文件、Intel® 标准 HEX 文件、详细描述存储器使用状况和符号参考的 MAP 文件、包含源代码行及生成机器码的绝对 LST 文件以及用于调试的 COFF 文件。

MPASM 汇编器具有如下特性：

- 集成在 MPLAB IDE 项目中
- 用户定义的宏可简化汇编代码
- 对多用途源文件进行条件汇编
- 允许完全控制汇编过程的指令

26.5 MPLINK 目标链接器 / MPLIB 目标库管理器

MPLINK 目标链接器包含了由 MPASM 汇编器、MPLAB C18 C 编译器产生的可重定位目标。通过使用链接器脚本中的指令，它还可链接预编译库中的可重定位目标。

MPLIB目标库管理器管理预编译代码库文件的创建和修改。当从源文件调用库中的一段子程序时，只有包含此子程序的模块被链接到应用程序。这样可使大型库在许多不同应用中被高效地利用。

目标链接器 / 库管理器具有如下特性：

- 高效地连接单个的库而不是许多小文件
- 通过将相关的模块组合在一起来增强代码的可维护性
- 只要列出、替换、删除和抽取模块，便可灵活地创建库

26.6 适用于各种器件系列的 MPLAB 汇编器、链接器和库管理器

MPLAB 汇编器为 PIC24、PIC32 和 dsPIC 器件从符号汇编语言生成可重定位机器码。 MPLAB C 编译器使用该汇编器生成目标文件。汇编器产生可重定位目标文件之后，可将这些目标文件存档，或与其他可重定位目标文件和存档链接以生成可执行文件。该汇编器有如下显著特性：

- 支持整个器件指令集
- 支持定点数据和浮点数据
- 命令行界面
- 丰富的指令集
- 灵活的宏语言
- MPLAB IDE 兼容性

26.7 MPLAB SIM 软件模拟器

MPLAB SIM 软件模拟器通过在指令级对 PIC MCU 和 dsPIC® DSC 进行模拟，可在 PC 主机环境下进行代码开发。对于任何给定的指令，都可以对数据区进行检查或修改，并通过一个全面的激励控制器来施加激励。可以将各寄存器记录在文件中，以便进行进一步的运行时分析。跟踪缓冲区和逻辑分析器的显示使软件模拟器还能记录和跟踪程序的执行、 I/O 的动作、大部分的外设及内部寄存器。

MPLAB SIM 软件模拟器完全支持使用 MPLAB C 编译器以及 MPASM 和 MPLAB 汇编器的符号调试。该软件模拟器可用于在硬件实验室环境外灵活地开发和调试代码，是一款完美且经济的软件开发工具。

26.8 MPLAB REAL ICE 在线仿真器系统

MPLAB REAL ICE 在线仿真器系统是 Microchip 针对其闪存 DSC 和 MCU 器件而推出的新一代高速仿真器。结合 MPLAB 集成开发环境 （IDE）所具有的易于使用且功能强大的图形用户界面，该仿真器可对 PIC® 闪存 MCU 和 dsPIC® 闪存 DSC 进行调试和编程。 IDE 是随每个工具包一起提供的。

该仿真器通过高速 USB 2.0 接口与设计工程师的 PC 相连，并利用与在线调试器系统兼容的连接器（RJ11）或新型抗噪声、高速低压差分信号 （LVDS）互连电缆（CAT5）与目标板相连。

可通过 MPLAB IDE 下载将来版本的固件，对该仿真器进行现场升级。在即将推出的 MPLAB IDE 版本中，会支持许多新器件，还将增加一些新特性。在同类仿真器中， MPLAB REAL ICE 的优势十分明显：低成本、全速仿真、运行时变量查看、跟踪分析、复杂断点、耐用的探针接口及较长 （长达 3 米）的互连电缆。

26.9 MPLAB ICD 3 在线调试器系统

MPLAB ICD 3 在线调试器系统是 Microchip 成本效益最高的高速硬件调试器 / 编程器，适用于 Microchip 闪存数字信号控制器 （DSC）和单片机 （MCU）器件。结合 MPLAB 集成开发环境 （IDE）所具有的功能强大但易于使用的图形用户界面，该调试器可对 PIC® 闪存单片机和 dsPIC® DSC 进行调试和编程。

MPLAB ICD 3 在线调试器通过高速 USB 2.0 接口与设计工程师的PC相连，并利用与MPLAB ICD 2或MPLAB REAL ICE 系统兼容的连接器（RJ-11）与目标板相连。MPLAB ICD 3 支持所有 MPLAB ICD 2 转接器。

26.10 PICkit 3 在线调试器 / 编程器及 PICkit 3 Debug Express

结合 MPLAB 集成开发环境 （IDE）所具有的功能强大的图形用户界面，MPLAB PICkit 3 可对 PIC® 闪存单片机和 dsPIC® 数字信号控制器进行调试和编程，且价位较低。MPLAB PICkit 3 通过全速 USB 接口与设计工程师的 PC 相连，并利用 Microchip 调试（RJ-11）连接器（与 MPLAB ICD 3 和 MPLAB REAL ICE 兼容）与目标板相连。连接器使用两个器件 I/O 引脚和复位线来实现在线调试和在线串行编程。

PICkit 3 Debug Express 包括 PICkit 3、演示板和单片机、连接电缆和光盘 （内含用户指南、课程、教程、编译器和 MPLAB IDE 软件）。

26.11 PICkit 2 开发编程器 / 调试器及 PICkit 2 Debug Express

PICkit™ 2 开发编程器 / 调试器是一款低成本开发工具，具有易于使用的界面，适用于对 Microchip 的闪存系列单片机进行编程和调试。这一全功能的 Windows® 编程界面支持低档（PIC10F、PIC12F5xx 和 PIC16F5xx）、中档（PIC12F6xx 和 PIC16F）、PIC18F、PIC24、dsPIC30、dsPIC33 和 PIC32 系列的 8 位、16 位及 32 位单片机，以及许多 Microchip 串行 EEPROM 产品。结合 Microchip 功能强大的 MPLAB 集成开发环境（IDE），PICkit 2 可对大多数 PIC® 单片机进行在线调试。即使 PIC 单片机已嵌入应用，在线调试功能仍可以运行、暂停和单步执行程序。在断点处暂停时，可以检查和修改文件寄存器。

PICkit 2 Debug Express 包括 PICkit 2、演示板和单片机、连接电缆和光盘（内含用户指南、课程、教程、编译器和 MPLAB IDE 软件）。

26.12 MPLAB PM3 器件编程器

MPLAB PM3 器件编程器是一款符合 CE 规范的通用器件编程器，在 VDDMIN 和 VDDMAX 点对其可编程电压进行校验以确保可靠性最高。它有一个用来显示菜单和错误消息的大 LCD 显示器（128 x 64），以及一个支持各种封装类型的可拆卸模块化插槽装置。编程器标准配置中带有一根 ICSP™ 电缆。在单机模式下，MPLAB PM3 器件编程器不必与 PC 相连即可对 PIC 器件进行读取、校验和编程。在该模式下它还可设置代码保护。MPLAB PM3 通过 RS-232 或 USB 电缆连接到 PC 主机上。MPLAB PM3 具备高速通信能力以及优化算法，可对具有大存储器的器件进行快速编程。它还包含了 MMC 卡，用于文件存储及数据应用。

26.13 演示 / 开发板、评估工具包及入门工具包

有许多演示、开发和评估板可用于各种 PIC MCU 和 dsPIC DSC，实现对全功能系统的快速应用开发。大多数的演示、开发和评估板都有实验布线区，供用户添加定制电路；还有应用固件和源代码，用于检查和修改。

这些板支持多种功能部件，包括 LED、温度传感器、开关、扬声器、RS-232 接口、LCD 显示器、电位计和附加 EEPROM 存储器。

演示和开发板可用于教学环境，在实验布线区设计定制电路，从而掌握各种单片机应用。

除了 PICDEM™ 和 dsPICDEM™ 演示 / 开发板系列电路外，Microchip 还有一系列评估工具包和演示软件，适用于模拟滤波器设计、KEELOQ® 数据安全产品 IC、CAN、IrDA®、PowerSmart 电池管理、SEEVAL® 评估系统、Σ–Δ ADC、流速传感器，等等。

同时还提供入门工具包，其中包含体验指定器件功能所需的所有软硬件。通常提供单个应用以及调试功能，都包含在一块电路板上。

有关演示、开发和评估工具包的完整列表，请访问 Microchip 网站（www.microchip.com）。

27.0 电气规范

绝对最大值 (†)

环境温度.....-40°C 至 +125°C
储存温度.....-65°C 至 +150°C
VDD 引脚相对于 VSS 的电压， PIC18F1XK50 .....-0.3V 至 +6.0V
VDD 引脚相对于 VSS 的电压， PIC18LF1XK50 .....-0.3V 至 +4.0V
$\overline{\text{MCLR}}$ 引脚相对于 Vss 的电压 .....-0.3V 至 +9.0V
VUSB 引脚相对于 VSS 的电压 .....-0.3V 至 +4.0V
D+ 和 D- 引脚相对于 VSS 的电压 ..... -0.3V 至 （VUSB + 0.3V）
所有其他引脚相对于 VSS 的电压.....-0.3V 至 （VDD + 0.3V）
总功耗 (1) .....800 mW
流出 VSS 引脚的最大电流.....95 mA
流入 VDD 引脚的最大电流 .....95 mA
钳位电流 IK （VPIN < 0 或 VPIN > VDD）..... ± 20 mA
任一 I/O 引脚的最大输出灌电流 .....25 mA
任一 I/O 引脚的最大输出拉电流 .....25 mA
所有端口的最大灌电流 .....90 mA
所有端口的最大拉电流 .....90 mA

注 **1：** 功耗按如下公式计算：PDIS = VDD x {IDD – ∑ IOH} + ∑ {(VDD – VOH) x IOH} + ∑(Vol x IOL)。
2： Vusb 必须始终保持 ≤ VDD + 0.3V。

† 注：如果器件工作条件超过上述“ 绝对最大值 ”，可能引起器件永久性损坏。这仅是极限参数，我们不建议器件工作在极限值甚至超过上述极限值。器件长时间工作在极限值条件下可能会影响其可靠性。

27.1 直流特性：PIC18F/LF1XK50-I/E （工业级，扩展级）

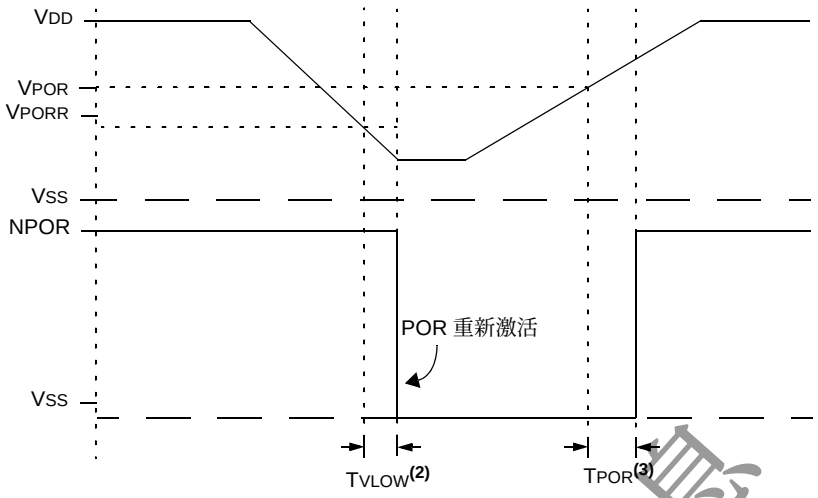
PIC18LF1XK50			标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
PIC18F1XK50			标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
参数编号	**符号**	**特性**	**最小值**	**典型值 †**	**最大值**	**单位**	**条件**
D001	VDD	**供电电压**					
		PIC18LF1XK50	1.8 2.7	— —	3.6 3.6	V V	FOSC < = 20 MHz FOSC < = 48 MHz
D001		PIC18F1XK50	1.8 2.7	— —	5.5 5.5	V V	FOSC < = 20 MHz FOSC < = 48 MHz
D002*	VDR	**RAM 数据保持电压 (1)**					
		PIC18LF1XK50	1.5	—	—	V	器件处于休眠模式
D002*		PIC18F1XK50	1.7	—	—	V	器件处于休眠模式
	VPOR*	**上电复位释放电压**	—	1.6	—	V	
	VPORR*	**上电复位重新激活 （rearm） 电压**	—	0.8	—	V	
D004*	SVDD	**VDD 上升速率** （确保内部上电复位信号）	0.05	—	—	V/ms	

* 这些参数为特性值，未经测试。

† 除非另外声明，否则“ 典型值 ”栏中的数据均为 3.0V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

注 1： 这是在不丢失 RAM 数据的前提下，休眠模式下 VDD 的下限值。

图 27-1：　　具有缓慢上升 VDD 的 POR 和 POR 重新激活



注　1： 当 NPOR 为低电平时，器件保持在复位状态。
2： TPOR 典型值为 1 µs。
3： TVLOW 典型值为 2.7 µs。

27.2 直流特性：PIC18F/LF1XK50-I/E （工业级，扩展级）

PIC18LF1XK50			标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
PIC18F1XK50			标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
参数编号	器件特性	最小值	典型值†	最大值	单位	条件	
						VDD	注
	供电电流 （IDD） (1, 2)						
D009	LDO 稳压器	—	30	—	µA	—	
		—	5	—	µA	—	LP 时钟模式和休眠 （需要禁止 FVR 和 BOR）
D010		—	6.0	9	µA	1.8	Fosc = 32 kHz LP 振荡器 (4) -40°C ≤ TA ≤ +85°C
		—	7	12	µA	3.0	
D010		—	6	11	µA	1.8	Fosc = 32 kHz LP 振荡器 (4) -40°C ≤ TA ≤ +85°C
		—	7	17	µA	3.0	
		—	12	20	µA	5.0	
D011*		—	6.0	12	µA	1.8	Fosc = 32 kHz LP 振荡器 -40°C ≤ TA ≤ +125°C
		—	9.0	16	µA	3.0	
D011*		—	8.0	15	µA	1.8	Fosc = 32 kHz LP 振荡器 (4) -40°C ≤ TA ≤ +125°C
		—	11	25	µA	3.0	
		—	12	35	µA	5.0	
D011*		—	170	220	µA	1.8	FOSC = 1 MHz XT 振荡器
		—	280	370	µA	3.0	
D011*		—	200	250	µA	1.8	FOSC = 1 MHz XT 振荡器
		—	310	400	µA	3.0	
		—	380	490	µA	5.0	
D011*		—	75	110	µA	1.8	FOSC = 1 MHz XT 振荡器 CPU 空闲
		—	130	190	µA	3.0	
D011*		—	90	130	µA	1.8	FOSC = 1 MHz XT 振荡器 CPU 空闲
		—	140	210	µA	3.0	
		—	160	250	µA	5.0	

* 这些参数为特性值，未经测试。

注 **1：** 有效工作模式下，所有 IDD 测量的测试条件为：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 ；禁止 WDT。

2： 供电电流主要受工作电压和频率的影响。其他因素如 I/O 引脚负载和开关速率、振荡器类型、内部代码执行模式以及温度也对电流消耗有影响。

3： 对于 RC 振荡器配置，该电流不包括流经 REXT 的电流。流经该电阻的电流可以由公式 IR = VDD/2REXT （mA）来估算，其中 REXT 的单位是 kΩ。

4： 禁止 FVR 和 BOR。

5： VUSB 引脚上的电容为 330 nF。

27.2 直流特性：PIC18F/LF1XK50-I/E （工业级，扩展级） （续）

PIC18LF1XK50	标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）
PIC18F1XK50	标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）

参数编号	器件特性	最小值	典型值†	最大值	单位	条件	
						VDD	注
	供电电流 （IDD） (1, 2)						
D012		—	300	700	µA	1.8	FOSC = 4 MHz XT 振荡器
		—	500	1200	µA	3.0	
D012		—	330	700	µA	1.8	FOSC = 4 MHz XT 振荡器
		—	530	1200	µA	3.0	
		—	730	1400	µA	5.0	
D012A		—	240	300	µA	1.8	FOSC = 4 MHz XT 振荡器 CPU 空闲
		—	440	550	µA	3.0	
D012A		—	230	300	µA	1.8	FOSC = 4 MHz XT 振荡器 CPU 空闲
		—	400	550	µA	3.0	
		—	470	640	µA	5.0	
D013		—	140	180	µA	1.8	FOSC = 1 MHz EC 振荡器 （中等功耗）
		—	230	300	µA	3.0	
D013		—	160	210	µA	1.8	FOSC = 1 MHz EC 振荡器 （中等功耗） (5)
		—	250	310	µA	3.0	
		—	290	380	µA	5.0	
D013A		—	50	64	µA	1.8	FOSC = 1 MHz EC 振荡器 （中等功耗） CPU 空闲
		—	86	110	µA	3.0	
D013A		—	70	100	µA	1.8	FOSC = 1 MHz EC 振荡器 （中等功耗） CPU 空闲 (5)
		—	100	150	µA	3.0	
		—	120	170	µA	5.0	
D014		—	500	640	µA	1.8	FOSC = 4 MHz EC 振荡器 （中等功耗）
		—	830	1100	µA	3.0	
D014		—	520	770	µA	1.8	FOSC = 4 MHz EC 振荡器 （中等功耗） (5)
		—	860	1200	µA	3.0	
		—	1000	1370	µA	5.0	

* 这些参数为特性值，未经测试。

注 1： 有效工作模式下，所有 IDD 测量的测试条件为：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 ；禁止 WDT。

2： 供电电流主要受工作电压和频率的影响。其他因素如 I/O 引脚负载和开关速率、振荡器类型、内部代码执行模式以及温度也对电流消耗有影响。

3： 对于 RC 振荡器配置，该电流不包括流经 REXT 的电流。流经该电阻的电流可以由公式 IR = VDD/2REXT （mA）来估算，其中 REXT 的单位是 kΩ。

4： 禁止 FVR 和 BOR。

5： VUSB 引脚上的电容为 330 nF。

27.2 直流特性：PIC18F/LF1XK50-I/E （工业级，扩展级）（续）

PIC18LF1XK50			标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
PIC18F1XK50			标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
参数编号	器件特性	最小值	典型值†	最大值	单位	条件	
						VDD	注
	供电电流 （IDD） (1, 2)						
D014A		—	200	250	µA	1.8	FOSC = 4 MHz EC 振荡器 （中等功耗） CPU 空闲
		—	340	440	µA	3.0	
D014A		—	210	303	µA	1.8	FOSC = 4 MHz EC 振荡器 （中等功耗） CPU 空闲 (5)
		—	360	520	µA	3.0	
		—	430	670	µA	5.0	
D015		—	820	1000	µA	1.8	FOSC = 6 MHz EC 振荡器 （高功耗）
		—	1500	1900	µA	3.0	
D015		—	830	1100	µA	1.8	FOSC = 6 MHz EC 振荡器 （高功耗） (5)
		—	1500	1900	µA	3.0	
		—	1700	2200	µA	5.0	
D015A		—	300	370	µA	1.8	FOSC = 6 MHz EC 振荡器 （高功耗） CPU 空闲
		—	510	640	µA	3.0	
D015A		—	320	430	µA	1.8	FOSC = 6 MHz EC 振荡器 （高功耗） CPU 空闲 (5)
		—	530	690	µA	3.0	
		—	740	840	µA	5.0	
D015B		—	4.7	6.0	mA	3.0	FOSC = 24 MHz 6 MHz EC 振荡器 （高功耗） PLL 使能
D015B		—	4.7	6.1	mA	3.0	FOSC = 24 MHz 6 MHz EC 振荡器 （高功耗） PLL 使能 (5)
		—	5.6	7.4	mA	5.0	
D015C		—	2.0	2.5	mA	3.0	FOSC = 24 MHz 6 MHz EC 振荡器 （高功耗） PLL 使能， CPU 空闲
D015C		—	2.0	2.5	mA	3.0	FOSC = 24 MHz 6 MHz EC 振荡器 （高功耗） PLL 使能， CPU 空闲 (5)
		—	2.3	3.0	mA	5.0	

* 这些参数为特性值，未经测试。

注 1： 有效工作模式下，所有 IDD 测量的测试条件为：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 ；禁止 WDT。

2： 供电电流主要受工作电压和频率的影响。其他因素如 I/O 引脚负载和开关速率、振荡器类型、内部代码执行模式以及温度也对电流消耗有影响。

3： 对于 RC 振荡器配置，该电流不包括流经 REXT 的电流。流经该电阻的电流可以由公式 IR = VDD/2REXT （mA）来估算，其中 REXT 的单位是 kΩ。

4： 禁止 FVR 和 BOR。

5： VUSB 引脚上的电容为 330 nF。

27.2 直流特性：PIC18F/LF1XK50-I/E （工业级，扩展级）（续）

PIC18LF1XK50			**标准工作条件 （除非另外声明）** 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
PIC18F1XK50			**标准工作条件 （除非另外声明）** 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
参数编号	**器件特性**	**最小值**	**典型值†**	**最大值**	**单位**	**条件**	
						VDD	**注**
	供电电流 （IDD） (1, 2)						
D016		—	2.6	3.3	mA	3.0	FOSC = 12 MHz EC 振荡器 （高功耗）
D016		—	2.6	3.3	mA	3.0	FOSC = 12 MHz EC 振荡器 （高功耗） (5)
		—	3.1	4.1	mA	5.0	
D017		—	1.0	1.3	mA	3.0	FOSC = 12 MHz EC 振荡器 （高功耗） CPU 空闲
D017		—	1.0	1.3	mA	3.0	FOSC = 12 MHz EC 振荡器 （高功耗） CPU 空闲 (5)
		—	1.2	1.6	mA	5.0	
D017A		—	9	12	mA	3.0	FOSC = 48 MHz 12 MHz EC 振荡器 （高功耗） PLL 使能
D017A		—	8.9	12	mA	3.0	FOSC = 48 MHz 12 MHz EC 振荡器 （高功耗） PLL 使能 (5)
		—	11	14	mA	5.0	
D017B		—	3.9	5.0	mA	3.0	FOSC = 48 MHz 12 MHz EC 振荡器 （高功耗） PLL 使能， CPU 空闲
D017B		—	3.9	5.0	mA	3.0	FOSC = 48 MHz 12 MHz EC 振荡器 （高功耗） PLL 使能， CPU 空闲 (5)
		—	4.7	6.0	mA	5.0	
D018		—	19	38	µA	1.8	FOSC = 32 kHz LFINTOSC 振荡器模式 (3, 5)
		—	23	44	µA	3.0	
D018		—	21	40	µA	1.8	FOSC = 32 kHz LFINTOSC 振荡器模式 (3, 5)
		—	25	46	µA	3.0	
		—	26	48	µA	5.0	
D019		—	16	33	µA	1.8	FOSC = 32 kHz LFINTOSC 振荡器 CPU 空闲
		—	18	38	µA	3.0	
D019		—	18	35	µA	1.8	FOSC = 32 kHz LFINTOSC 振荡器 CPU 空闲 (5)
		—	20	40	µA	3.0	
		—	21	42	µA	5.0	

* 这些参数为特性值，未经测试。

注 **1：** 有效工作模式下，所有 IDD 测量的测试条件为：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 ；禁止 WDT。

2： 供电电流主要受工作电压和频率的影响。其他因素如 I/O 引脚负载和开关速率、振荡器类型、内部代码执行模式以及温度也对电流消耗有影响。

3： 对于 RC 振荡器配置，该电流不包括流经 REXT 的电流。流经该电阻的电流可以由公式 IR = VDD/2REXT （mA）来估算，其中 REXT 的单位是 kΩ。

4： 禁止 FVR 和 BOR。

5： VUSB 引脚上的电容为 330 nF。

27.2 直流特性：PIC18F/LF1XK50-I/E （工业级，扩展级）（续）

PIC18LF1XK50			标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
PIC18F1XK50			标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
参数编号	器件特性	最小值	典型值†	最大值	单位	条件	
						VDD	注
	供电电流 （IDD） (1, 2)						
D020		—	320	470	µA	1.8	FOSC = 500 kHz LFINTOSC 振荡器
		—	460	670	µA	3.0	
D020		—	350	500	µA	1.8	FOSC = 500 kHz LFINTOSC 振荡器 (5)
		—	490	700	µA	3.0	
		—	540	780	µA	5.0	
D021		—	380	600	µA	1.8	FOSC = 1 MHz HFINTOSC 振荡器
		—	550	870	µA	3.0	
D021		—	410	600	µA	1.8	FOSC = 1 MHz HFINTOSC 振荡器 (5)
		—	580	870	µA	3.0	
		—	650	970	µA	5.0	
D021A		—	290	600	µA	1.8	FOSC = 1 MHz HFINTOSC 振荡器 CPU 空闲
		—	410	760	µA	3.0	
D021A		—	320	820	µA	1.8	FOSC = 1 MHz HFINTOSC 振荡器 CPU 空闲 (5)
		—	440	770	µA	3.0	
		—	490	880	µA	5.0	
D022		—	1.2	1.6	mA	1.8	FOSC = 8 MHz HFINTOSC 振荡器
		—	2.1	2.9	mA	3.0	
D022		—	1.2	1.6	mA	1.8	FOSC = 8 MHz HFINTOSC 振荡器 (5)
		—	2.1	2.9	mA	3.0	
		—	2.4	3.5	mA	5.0	
D023		—	2.0	2.7	mA	1.8	FOSC = 16 MHz HFINTOSC 振荡器
		—	3.5	4.8	mA	3.0	
D023		—	2.0	2.7	mA	1.8	FOSC = 16 MHz HFINTOSC 振荡器 (5)
		—	3.5	4.8	mA	3.0	
		—	4.0	6.0	mA	5.0	
D023A		—	0.9	1.3	mA	1.8	FOSC = 16 MHz HFINTOSC 振荡器 CPU 空闲
		—	1.5	2.1	mA	3.0	
D023A		—	0.9	1.3	mA	1.8	FOSC = 16 MHz HFINTOSC 振荡器 CPU 空闲 (5)
		—	1.5	2.1	mA	3.0	
		—	1.7	2.6	mA	5.0	

* 这些参数为特性值，未经测试。

注 **1：** 有效工作模式下，所有 IDD 测量的测试条件为：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 ；禁止 WDT。

2： 供电电流主要受工作电压和频率的影响。其他因素如 I/O 引脚负载和开关速率、振荡器类型、内部代码执行模式以及温度也对电流消耗有影响。

3： 对于 RC 振荡器配置，该电流不包括流经 REXT 的电流。流经该电阻的电流可以由公式 IR = VDD/2REXT （mA）来估算，其中 REXT 的单位是 kΩ。

4： 禁止 FVR 和 BOR。

5： VUSB 引脚上的电容为 330 nF。

27.2 直流特性：PIC18F/LF1XK50-I/E （工业级，扩展级）（续）

PIC18LF1XK50	标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）
PIC18F1XK50	标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）

参数编号	器件特性	最小值	典型值†	最大值	单位	条件	
						VDD	注
	供电电流 （IDD） (1, 2)						
D024		—	0.5	0.7	mA	1.8	FOSC = 4 MHz EXTRC 振荡器模式
		—	0.9	1.1	mA	3.0	
D024		—	0.5	0.7	mA	1.8	FOSC = 4 MHz EXTRC 振荡器模式 (5)
		—	0.9	1.1	mA	3.0	
		—	1.0	1.4	mA	5.0	
D025		—	1.0	1.5	mA	1.8	FOSC = 6 MHz HS 振荡器
		—	1.7	2.1	mA	3.0	
D025		—	1.0	1.5	mA	1.8	FOSC = 6 MHz HS 振荡器 (5)
		—	1.7	2.1	mA	3.0	
		—	2.1	2.5	mA	5.0	
D025A		—	5.4	6.0	mA	3.0	FOSC = 24 MHz 6 MHz HS 振荡器 PLL 使能
D025A		—	5.4	6.0	mA	3.0	FOSC = 24 MHz 6 MHz HS 振荡器 PLL 使能 (5)
		—	7.1	7.6	mA	5.0	
D026		—	3.2	3.3	mA	3.0	FOSC = 12 MHz HS 振荡器
D026		—	3.2	3.3	mA	3.0	FOSC = 12 MHz HS 振荡器 (5)
		—	4.8	4.2	mA	5.0	
D026A		—	10	12	mA	3.0	FOSC = 48 MHz 12 MHz HS 振荡器 PLL 使能
D026A		—	10	12	mA	3.0	FOSC = 48 MHz 12 MHz HS 振荡器 PLL 使能 (5)
		—	13	15	mA	5.0	

* 这些参数为特性值，未经测试。

注 1： 有效工作模式下，所有 IDD 测量的测试条件为：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 ；禁止 WDT。

2： 供电电流主要受工作电压和频率的影响。其他因素如 I/O 引脚负载和开关速率、振荡器类型、内部代码执行模式以及温度也对电流消耗有影响。

3： 对于 RC 振荡器配置，该电流不包括流经 REXT 的电流。流经该电阻的电流可以由公式 IR = VDD/2REXT （mA）来估算，其中 REXT 的单位是 kΩ。

4： 禁止 FVR 和 BOR。

5： VUSB 引脚上的电容为 330 nF。

27.3 直流特性：PIC18F/LF1XK50-I/E （掉电）

PIC18LF1XK50	标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）
PIC18F1XK50	标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）

参数编号	器件特性	最小值	典型值 †	最大值 +85°C	最大值 +125°C	单位	条件	
							VDD	注
	掉电基本电流 （IPD） (2)							
D027		—	0.024	0.7	6.7	µA	1.8	禁止 WDT、 BOR、 FVR、稳压器和 T1OSC，所有外设不工作
		—	0.078	1.9	8.5	µA	3.0	
D027		—	6.0	7.0	13	µA	1.8	禁止 WDT、 BOR、 FVR 和 T1OSC，所有外设不工作
		—	7.0	10	15	µA	3.0	
		—	8.0	12	19	µA	5.0	
	掉电模块电流							
D028		—	0.45	1.3	4.4	µA	1.8	LPWDT 电流 (1)
		—	0.75	2.0	6.0	µA	3.0	
D028		—	6.5	7.0	10.5	µA	1.8	LPWDT 电流 (1)
		—	9.6	10.6	17.6	µA	3.0	
		—	10.5	16.5	20	µA	5.0	
D029		—	12	17	25	µA	1.8	FVR 电流 (3)
		—	22	19	25	µA	3.0	
D029		—	28	42	50	µA	1.8	FVR 电流 (3, 5)
		—	35.6	45.6	55	µA	3.0	
		—	38.5	49	60	µA	5.0	
D030		—	—	21	27	µA	3.0	BOR 电流 (1, 3)
D030		—	27	48	51	µA	3.0	BOR 电流 (1, 3, 5)
		—	36.5	51	55	µA	5.0	
D031		—	0.79	3.6	5.3	µA	1.8	T1OSC 电流 (1)
		—	1.8	2.9	6.9	µA	3.0	
D031		—	8.0	7.5	10	µA	1.8	T1OSC 电流 (1)
		—	8.5	10.5	15	µA	3.0	
		—	10.5	12.5	24	µA	5.0	

* 这些参数为特性值，未经测试。

† 除非另外声明，否则 “ 典型值 ” 栏中的数据均为 3.0V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

注 1： 外设电流为基本 IDD 或 IPD 与该外设使能时所额外消耗的电流之和。可通过从该极限值中减去基本 IDD 或 IPD 电流，以确定外设 Δ 电流。在计算总电流消耗时应使用最大值。

2： 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 时测得的。

3： 只要 BOR 使能，就会自动使能固定参考电压。

4： A/D 振荡器源是 FRC。

5： VUSB 引脚上的电容为 330 µF。

27.3 直流特性：PIC18F/LF1XK50-I/E （掉电） （续）

PIC18LF1XK50			标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）					
PIC18F1XK50			**标准工作条件 （除非另外声明）** 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）					
参数编号	**器件特性**	**最小值**	**典型值 †**	**最大值 +85°C**	**最大值 +125°C**	**单位**	**条件**	
							VDD	**注**
	掉电模块电流							
D032		—	—	1.8	8	µA	1.8	A/D 电流 (1, 4)，无转换进行
		—	—	3	10	µA	3.0	
D032		—	—	6	12	µA	1.8	A/D 电流 (1, 4)，无转换进行
		—	—	10	17	µA	3.0	
		—	—	11.5	22	µA	5.0	
D033		—	—	38	44	µA	1.8	比较器电流，低功耗
		—	—	40	47	µA	3.0	
D033		—	30	40	49	µA	2.0	比较器电流，低功耗
		—	34	44	53	µA	3.0	
		—	36	50	60	µA	5.0	
D033A		—	—	239	244	µA	1.8	比较器电流，高功耗
		—	—	242	249	µA	3.0	
D033A		—	144	243	250	µA	2.0	比较器电流，高功耗
		—	146	247	256	µA	3.0	
		—	151	253	264	µA	5.0	
D034		—	—	18	23	µA	1.8	参考电压电流
		—	—	30	35	µA	3.0	
D034		—	35	36	44	µA	2.0	参考电压电流
		—	43	44	60	µA	3.0	
		—	55	65	74	µA	5.0	

* 这些参数为特性值，未经测试。

† 除非另外声明，否则“ 典型值 ”栏中的数据均为 3.0V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

注 1： 外设电流为基本 IDD 或 IPD 与该外设使能时所额外消耗的电流之和。可通过从该极限值中减去基本 IDD 或 IPD 电流，以确定外设 Δ 电流。在计算总电流消耗时应使用最大值。

2： 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 时测得的。

3： 只要 BOR 使能，就会自动使能固定参考电压。

4： A/D 振荡器源是 FRC。

5： VUSB 引脚上的电容为 330 µF。

27.4 直流特性：PIC18F/LF1XK50-I/E

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） -40°C ≤ TA ≤ +125°C （扩展级）				
参数编号	**符号**	**特性**	**最小值**	**典型值 †**	**最大值**	**单位**	**条件**
	VIL	**输入低电压**					
		I/O 端口：					
D036		带 TTL 缓冲器	—	—	0.8	V	4.5V ≤ VDD ≤ 5.5V
D036A			—	—	0.15 VDD	V	1.8V ≤ VDD ≤ 4.5V
D037		带施密特触发缓冲器	—	—	0.2 VDD	V	1.8V ≤ VDD ≤ 5.5V
		带 I^2C 电平	—	—	0.3 VDD	V	
D038		$\overline{MCLR}$ 和 OSC1 （RC 模式） (1)	—	—	0.2 VDD	V	
D039A		OSC1 （HS 模式）	—	—	0.3 VDD	V	
	VIH	**输入高电压**					
		I/O 端口：		—	—		
D040		带 TTL 缓冲器	2.0	—	—	V	4.5V ≤ VDD ≤ 5.5V
D040A			0.25 VDD + 0.8	—	—	V	1.8V ≤ VDD ≤ 4.5V
D041		带施密特触发缓冲器	0.8 VDD	—		V	1.8V ≤ VDD ≤ 5.5V
		带 I^2C 电平	0.7 VDD	—	—	V	
D042		$\overline{MCLR}$	0.8 VDD	—	—	V	
D043A		OSC1 （HS 模式）	0.7 VDD	—	—	V	
D043B		OSC1 （RC 模式）	0.9 VDD		—	V	（注 1）
	IIL	**输入泄漏电流** (2)					
D060		I/O 端口	—	± 5	± 100	nA	VSS ≤ VPIN ≤ VDD，引脚处于高阻态
D061		$\overline{MCLR}$ (3)	—	± 50	± 200	nA	VSS ≤ VPIN ≤ VDD
	IPUR	**PORTB 弱上拉电流**					
D070*			50	250	400	µA	VDD = 5.0V， VPIN = VSS
	VOL	**输出低电压** (4)					
D080		I/O 端口	—	—	VSS+0.6 VSS+0.6 VSS+0.6	V	IOH = 8mA， VDD = 5V IOH = 6mA， VDD = 3.3V IOH = 1.8mA， VDD = 1.8V
	VOH	**输出高电压** (4)					
D090		I/O 端口	VDD-0.7 VDD-0.7 VDD-0.7	—	—	V	IOL = 3.5mA， VDD = 5V IOL = 3mA， VDD = 3.3V IOL = 1mA， VDD = 1.8V

* 这些参数为特性值，未经测试。

† 除非另外声明，否则“ 典型值 ”栏中的数据均为 3.0V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

注 1： 在 RC 振荡器配置中， OSC1/CLKIN 引脚被配置为施密特触发器输入。在 RC 模式下，建议不要使用外部时钟。

2： 负电流定义为引脚的拉电流。

3： MCLR 引脚上的泄漏电流主要取决于所施加电压。规定电压为正常工作条件下的电压。在不同的输入电压下可能测得更高的泄漏电流。

4： 在 CLKOUT 模式下包括 OSC2。

27.4 直流特性：PIC18F/LF1XK50-I/E （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） -40°C ≤ TA ≤ +125°C （扩展级）				
参数编号	**符号**	**特性**	**最小值**	**典型值 †**	**最大值**	**单位**	**条件**
		输出引脚上的容性负载规范					
D101*	COSC2	OSC2 引脚	—	—	15	pF	当外部时钟用于驱动 OSC1 时处于 XT、 HS 和 LP 模式下
D101A*	CIO	所有 I/O 引脚	—	—	50	pF	
		闪存					
D130	EP	单元耐擦写能力	10K 100K	—	—	E/W	闪存程序存储器 闪存数据存储器
D131		读操作时的 VDD	VMIN	—	—	V	
		擦除 / 编程时 MCLR/VPP 上的电压	8.0	—	9.0	V	编程时的温度：-40°C ≤ TA ≤ 85°C
		批量擦除时的 VDD	2.7	—	VDD Max	V	编程时的温度：10°C ≤ TA ≤ 40°C
D132	VPEW	写或行擦除时的 VDD	VDD Min	—	VDD Max	V	VMIN = 最小工作电压 VMAX = 最大工作电压
	IPPPGM	擦除 / 写操作时 MCLR/VPP 上的电流	—	—	1.0	mA	
	IDDPGM	擦除 / 写操作时 VDD 上的电流	—		5.0	mA	
D133	TPEW	擦除 / 写周期	—	4.0	5.0	ms	
D134	TRETD	特性保持时间	40	—	—	年	假设没有违反其他规范
		VUSB 电容充电					
D135		充电电流	—	200	—	µA	
D135A		充电结束时的拉 / 灌能力	—	0	—	mA	

* 这些参数为特性值，未经测试。

† 除非另外声明，否则 “ 典型值 ” 栏中的数据均为 3.0V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

注 1： 在 RC 振荡器配置中， OSC1/CLKIN 引脚被配置为施密特触发器输入。在 RC 模式下，建议不要使用外部时钟。

2： 负电流定义为引脚的拉电流。

3： MCLR 引脚上的泄漏电流主要取决于所施加电压。规定电压为正常工作条件下的电压。在不同的输入电压下可能测得更高的泄漏电流。

4： 在 CLKOUT 模式下包括 OSC2。

27.5 USB 模块规范

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	条件
D313	VUSB	USB 电压	3.0	—	3.6	V	VUSB 引脚上的电压必须在此范围内以确保 USB 正常工作
D314	IIL	引脚上的输入泄漏电流	—	—	± 1	µA	VSS ≤ VPIN ≤ VDD，引脚处于高阻态
D315	VILUSB	USB 缓冲区的输入低电压	—	—	0.8	V	用于 VUSB 范围
D316	VIHUSB	USB 缓冲区的输入高电压	2.0	—	—	V	用于 VUSB 范围
D318	VDIFS	差分输入灵敏度	—	—	0.2	V	当处于 VCM 时， D+ 和 D- 之间的差值必须超出此值。
D319	VCM	差分共模范围	0.8	—	2.5	V	
D320	ZOUT	驱动器输出阻抗 (1)	28	—	44	Ω	
D321	VOL	输出低电压	0.0	—	0.3	V	1.5 kΩ 负载连接到 3.6V
D322	VOH	输出高电压	2.8	—	3.6	V	1.5 kΩ 负载接地

注 1： D+ 和 D- 信号线具有内置的阻抗匹配电阻。在 PIC18F1XK50/PIC18LF1XK50 系列器件和 USB 电缆之间的 D+/D- 信号通路上不需要外接任何电阻、电容或磁性元件。

27.6 散热考虑

标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +125°C					
参数编号	**符号**	**特性**	**典型值**	**单位**	**条件**
TH01	θJA	热阻 （结点到环境）	62.4	°C/W	20 引脚 PDIP 封装
			85.2	°C/W	20 引脚 SOIC 封装
			108.1	°C/W	20 引脚 SSOP 封装
			TBD	°C/W	20 引脚 QFN 5x5mm 封装
TH02	θJC	热阻 （结点到管壳）	31.4	°C/W	20 引脚 PDIP 封装
			24	°C/W	20 引脚 SOIC 封装
			24	°C/W	20 引脚 SSOP 封装
TH03	TJMAX	最高结温	150	°C	
TH04	PD	功耗	—	W	PD = PINTERNAL + PI/O
TH05	PINTERNAL	内部功耗	—	W	PINTERNAL = IDD x VDD(1)
TH06	PI/O	I/O 功耗	—	W	PI/O = Σ (IOL * VOL) + Σ (IOH * (VDD - VOH))
TH07	PDER	减额功耗	—	W	PDER = PDMAX (TJ - TA)/θJA(2)

图注： TBD = 待定

注 1： IDD 为不驱动输出引脚上任何负载时使芯片独立运行的电流。

2： TA = 环境温度

3： TJ = 结点温度

27.7 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

T			
F	频率	T	时间

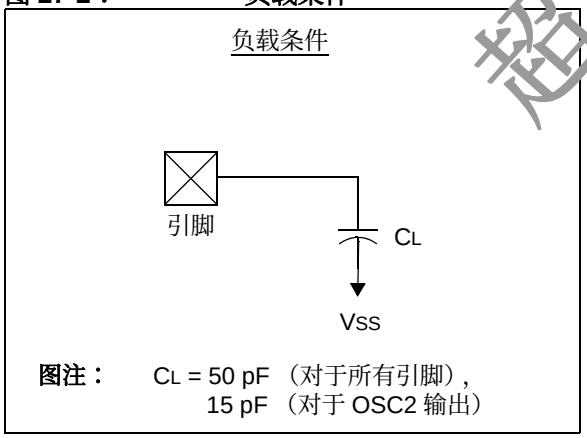
小写字母（pp）及其含义：

pp			
cc	CCP1	osc	OSC1
ck	CLKOUT	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	T1CKI
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（高阻态）	V	有效
L	低	Z	高阻态

图 27-2： 负载条件



图注： C_L = 50 pF （对于所有引脚），
15 pF （对于 OSC2 输出）

27.8 交流特性：PIC18F1XK50/PIC18LF1XK50-I/E

图 27-3： 时钟时序

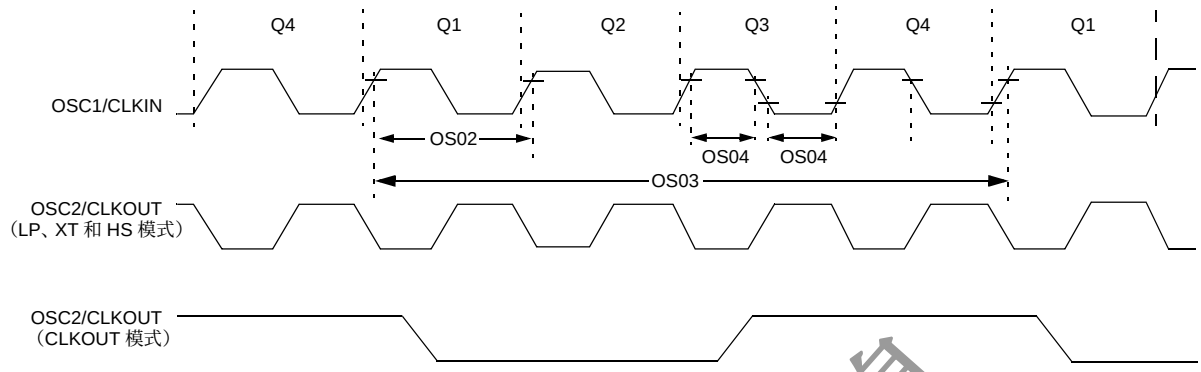
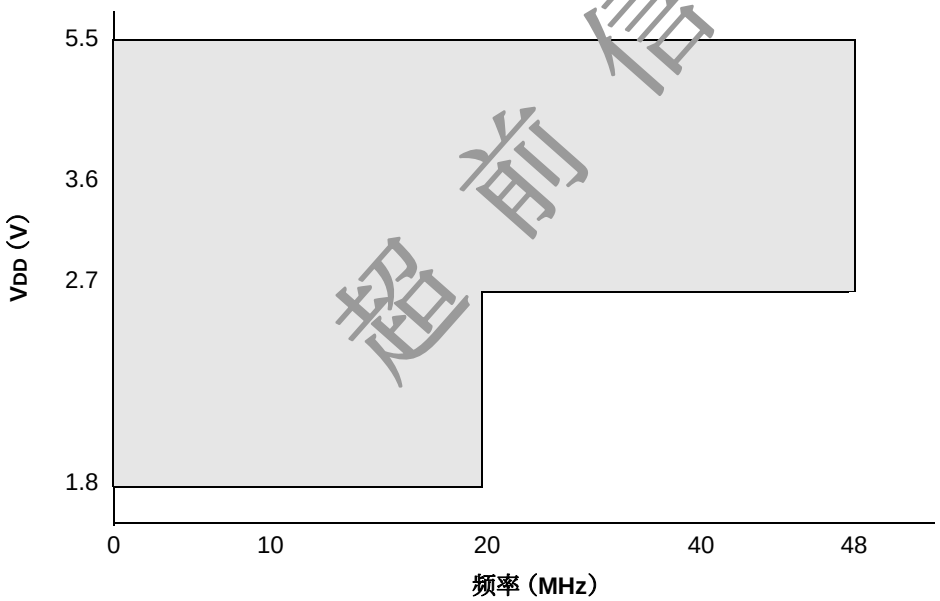
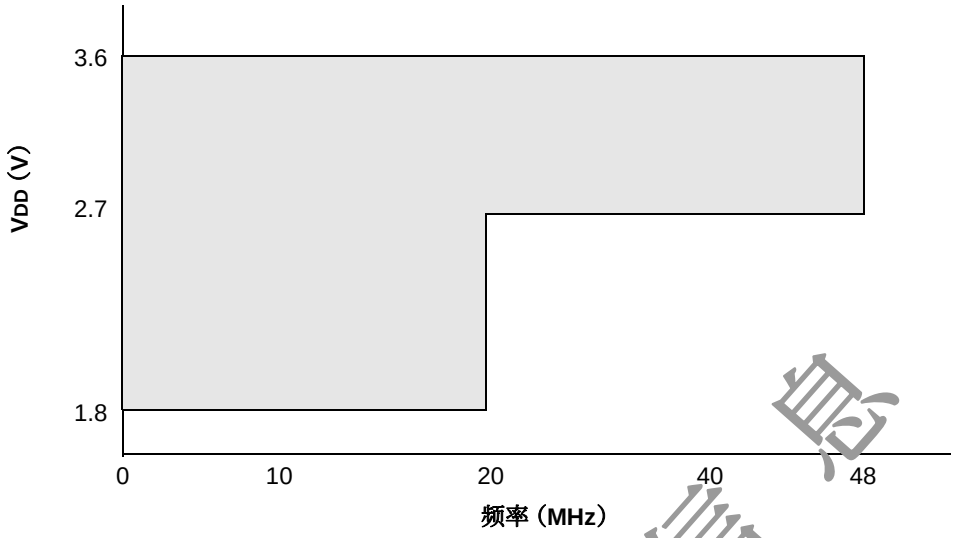


图 27-4： PIC18F1XK50 电压—频率关系图， -40°C ≤ TA ≤ +85°C



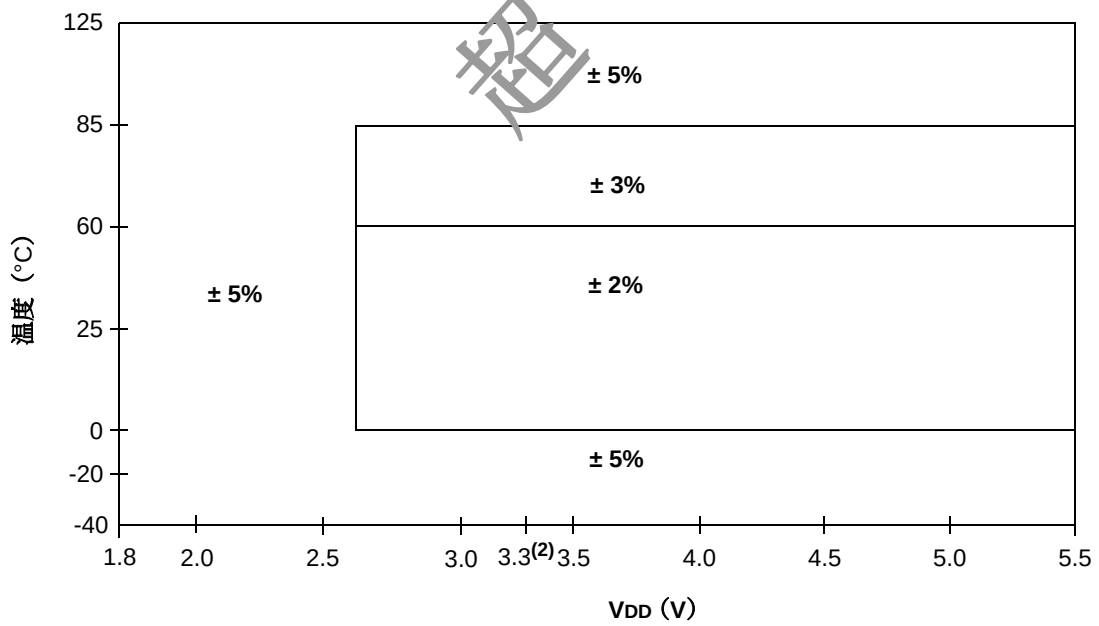
注 1： 阴影区域表示允许的电压频率组合。
2： 请参见表 27-1 了解每种振荡器模式所支持的频率。

图 27-5：　　PIC18LF1XK50 电压－频率关系图，　-40°C ≤ TA ≤ +125°C



注　**1：** 阴影区域表示允许的电压频率组合。
2： 请参见表 27-1 了解每种振荡器模式所支持的频率。

图 27-6：　　不同器件 VDD 和温度下的 HFINTOSC 频率精度



注　**1：** 该图涵盖了稳压器使能和稳压器禁止两种状态。
2： 稳压器标称电压。

表 27-1： 时钟振荡器时序要求

参数编号	符号	特性	最小值	典型值 †	最大值	单位	条件
标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +125°C							
OS01	FOSC	外部 CLKIN 频率 (1)	DC	—	37	kHz	EC 振荡器模式 （低功耗）
			DC	—	4	MHz	EC 振荡器模式 （中等功耗）
			DC	—	48	MHz	EC 振荡器模式 （高功耗）
		振荡器频率 (1)	—	32.768	—	kHz	LP 振荡器模式
			0.1	—	4	MHz	XT 振荡器模式
			1	—	20	MHz	HS 振荡器模式
			DC	—	4	MHz	RC 振荡器模式
OS02	TOSC	外部 CLKIN 周期 (1)	27	—	∞	µs	LP 振荡器模式
			250	—	∞	ns	XT 振荡器模式
			50	—	∞	ns	HS 振荡器模式
			20.80	—	∞	ns	EC 振荡器模式
		振荡器周期 (1)	—	—	—	µs	LP 振荡器模式
			250	—	10,000	ns	XT 振荡器模式
			50	—	1,000	ns	HS 振荡器模式
			250	—	—	ns	RC 振荡器模式
OS03	TCY	指令周期 (1)	83	TCY	DC	ns	TCY = 4/FOSC
OS04*	TosH, TosL	外部 CLKIN 高电平时间，外部 CLKIN 低电平时间	2	—	—	µs	LP 振荡器
			100	—	—	ns	XT 振荡器
			20	—	—	ns	HS 振荡器
OS05*	TosR, TosF	外部 CLKIN 上升时间，外部 CLKIN 下降时间	0	—	∞	ns	LP 振荡器
			0	—	∞	ns	XT 振荡器
			0	—	∞	ns	HS 振荡器

* 这些参数为特性值，未经测试。

† 除非另外声明，否则 “ 典型值 ” 栏中的数据均为 3.0V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

注 1： 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 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 27-2： 振荡器参数

标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +125°C								
参数编号	**符号**	**特性**	**频率容差**	**最小值**	**典型值 †**	**最大值**	**单位**	**条件**
OS08	HFOSC	内部已校准的 HFINTOSC 频率 (2)	±2%	—	16.0	—	MHz	0°C ≤ TA ≤ +60°C
			±3%	—	16.0	—	MHz	60°C ≤ TA ≤ +85°C
			±5%	—	16.0	—	MHz	-40°C ≤ TA ≤ +125°C
OS10*	TIOSC ST	HFINTOSC 从休眠模式唤醒的启动时间	—	—	5	8	µs	VDD = 2.0V， -40°C 至 +85°C
			—	—	5	8	µs	VDD = 3.0V， -40°C 至 +85°C
			—	—	5	8	µs	VDD = 5.0V， -40°C 至 +85°C

* 这些参数为特性值，未经测试。

† 除非另外声明，否则“ 典型值 ”栏中的数据均为 3.0V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

注 1： 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 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）。

2： 为了确保振荡器频率容差，必须尽可能靠近器件，在 VDD 和 VSS 之间接去耦电容。建议并联 0.1 µF 和 0.01 µF 的电容。

3： 根据设计。

表 27-3： PLL 时钟时序规范 （VDD = 42.7V 至 5.5V）

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	—	12	MHz	
F11	FSYS	片上 VCO 系统频率	16	—	48	MHz	
F12	t_{rc}	PLL 起振时间 （锁定时间）	—	—	2	ms	
F13*	ΔCLK	CLKOUT 稳定性 （抗抖动性）	-0.25%	—	+0.25%	%	

* 这些参数为特性值，未经测试。

† 除非另外声明，否则“ 典型值 ”栏中的数据均为 3.0V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

图 27-7：　　CLKOUT 和 I/O 时序

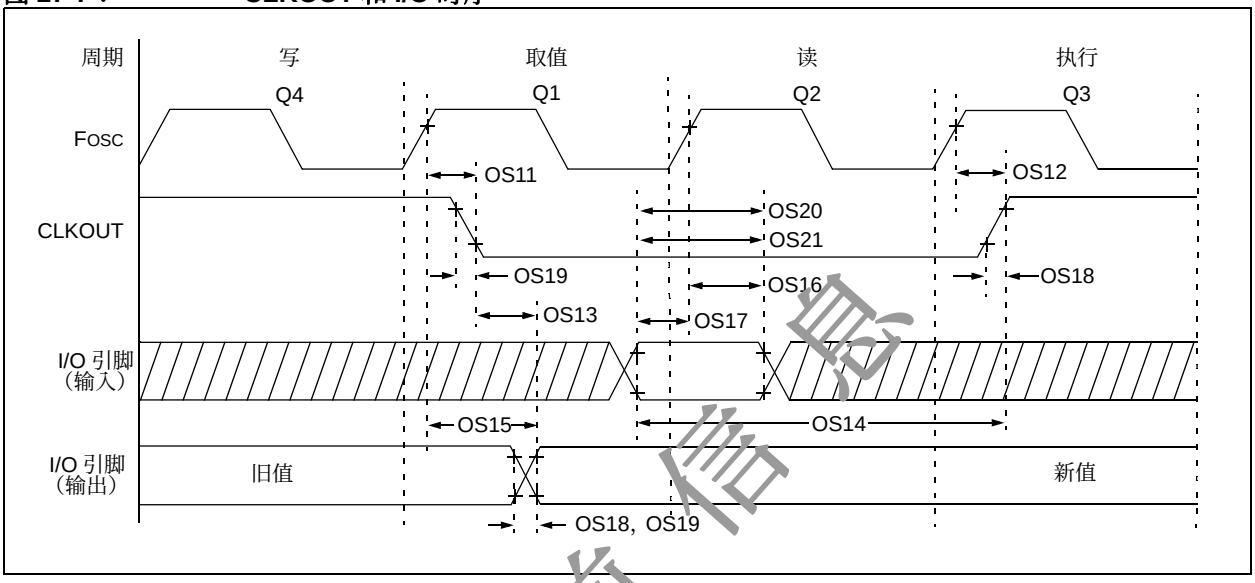


表 27-4：　　CLKOUT 和 I/O 时序参数

标准工作条件 （除非另外声明）
工作温度　　-40°C ≤ TA ≤ +125°C

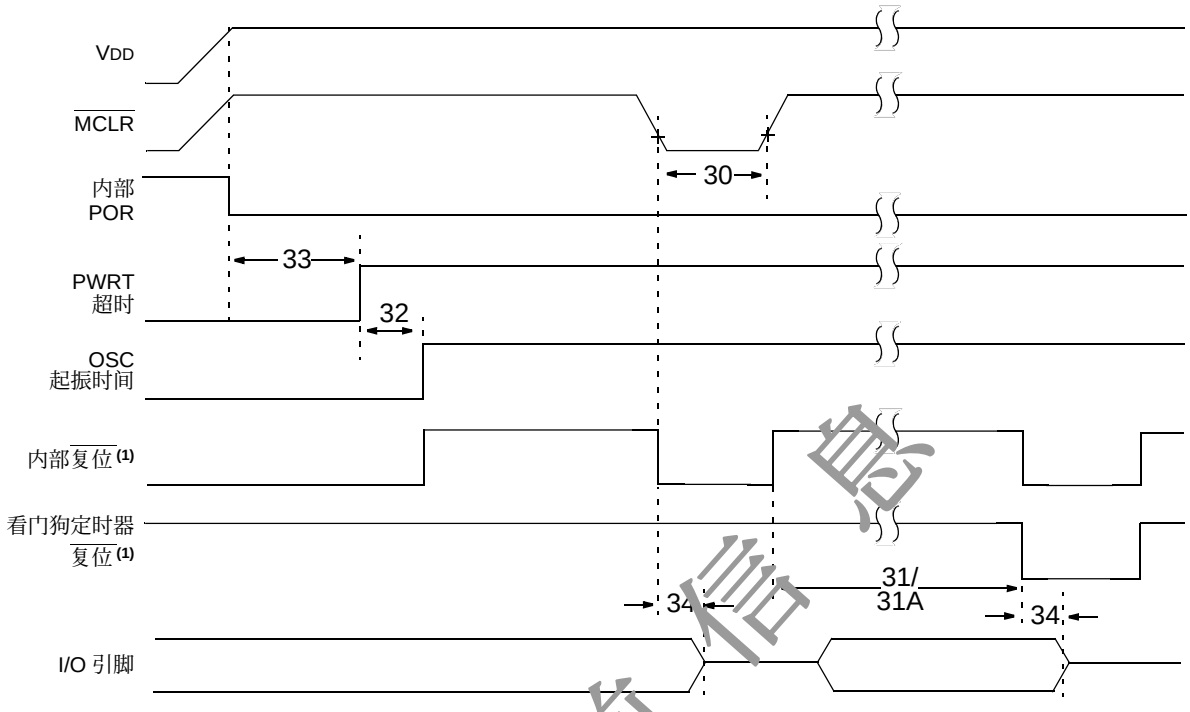
参数编号	符号	特性	最小值	典型值 †	最大值	单位	条件
OS11	TosH2ckL	Fosc↑ 到 CLKOUT↓ 的时间 (1)	—	—	70	ns	VDD = 3.0-5.0V
OS12	TosH2ckH	Fosc↑ 到 CLKOUT↑ 的时间 (1)	—	—	72	ns	VDD = 3.0-5.0V
OS13	TckL2ioV	CLKOUT↓ 到端口输出有效的时间 (1)	—	—	20	ns	
OS14	TioV2ckH	CLKOUT↑ 前端口输入有效的时间 (1)	TOSC + 200 ns	—	—	ns	
OS15	TosH2ioV	Fosc↑ （Q1 周期）到端口输出有效的时间	—	50	70*	ns	VDD = 3.0-5.0V
OS16	TosH2ioI	Fosc↑ （Q2 周期）到端口输入无效的时间 （I/O 输入保持时间）	50	—	—	ns	VDD = 3.0-5.0V
OS17	TioV2osH	端口输入有效到 Fosc↑ （Q2 周期）的时间 （I/O 输入建立时间）	20	—	—	ns	
OS18	TioR	端口输出上升时间	— —	90 55	140 80	ns	VDD = 2.0V VDD = 3.3-5.0V
OS19	TioF	端口输出下降时间	— —	60 44	80 60	ns	VDD = 2.0V VDD = 3.3-5.0V
OS20*	Tinp	INT 引脚输入高电平或低电平时间	25	—	—	ns	
OS21*	Trbp	PORTB 电平变化中断新输入电平时间	TCY	—	—	ns	

*　这些参数为特性值，未经测试。

†　除非另外声明，否则 “ 典型值 ” 栏中的数据均为 3.0V 和 25°C 条件下的值。

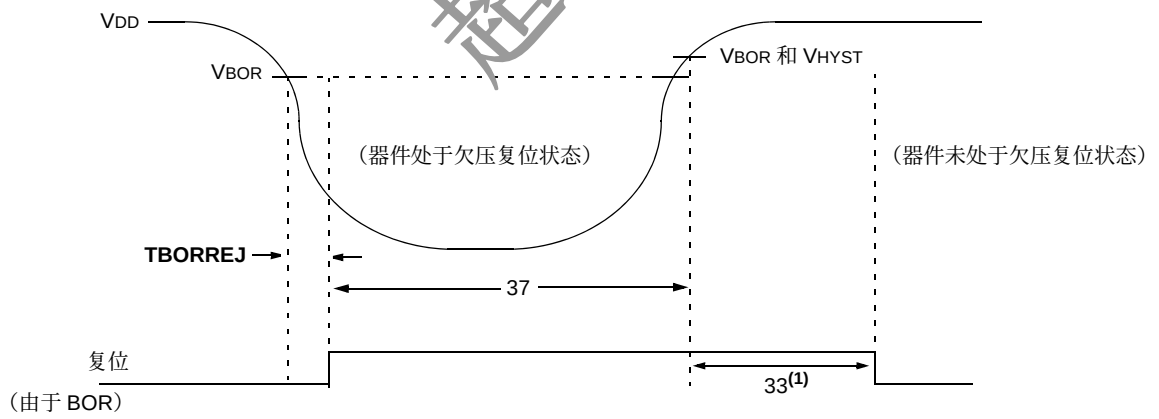
注　1： 测量是在 RC 模式下进行的，其中 CLKOUT 输出为 4 x TOSC。

图 27-8：　　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　1： 拉低。

图 27-9：　　欠压复位时序和特性



注　1： 仅在配置字寄存器中的 $\overline{\text{PWRTE}}$ 位编程为 0 时才有 64 ms 延时。如果 $\overline{\text{PWRTE}}$ = 0，则有 2 ms 延时。

表 27-5： 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参数

标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +125°C							
参数编号	**符号**	**特性**	**最小值**	**典型值 †**	**最大值**	**单位**	**条件**
30	TMCL	$\overline{MCLR}$ 脉冲宽度 （低电平）	2 5	— —	— —	µs µs	VDD = 3.3-5V， -40°C 至 +85°C VDD = 3.3-5V
31	TWDT	标准看门狗定时器超时周期 **(5)**	10 10	17 17	27 30	ms ms	VDD = 3.3V-5V， -40°C 至 +85°C VDD = 3.3V-5V
31A	TWDTLP	低功耗看门狗定时器超时周期	10 10	18 18	27 33	ms ms	VDD = 3.3V-5V， -40°C 至 +85°C VDD = 3.3V-5V
32	TOST	振荡器起振定时器周期 **(1), (2)**	—	1024	—	Tosc	**（注 3）**
33*	TPWRT	上电延时定时器周期， $\overline{PWRTE}$ = 0	40	65	140	ms	
34*	TIOZ	自 $\overline{MCLR}$ 低电平或看门狗定时器复位起 I/O 处于高阻态的时间	—	—	2.0	µs	
35	VBOR	欠压复位电压	1.8 2.09 2.57 2.71	1.9 2.2 2.7 2.85	2.05 2.35 2.85 3.0	V V V V	BORV = 1.9V BORV = 2.2V BORV = 2.7V BORV = 2.85V
36*	VHYST	欠压复位滞后	25	50	75	mV	-40°C 至 +85°C
37*	TBORDC	欠压复位 DC 响应时间	0	1	40	µs	VDD ≤ VBOR

图注： TBD = 待定

* 这些参数为特性值，未经测试。

† 除非另外声明，否则 “ 典型值 ” 栏中的数据均为 3V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

注 1： 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 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）。

2： 根据设计。

3： 较慢时钟的周期。

4： 为了确保这些电压容差，必须尽可能靠近器件，在 VDD 和 VSS 之间接去耦电容。建议并联 0.1 µF 和 0.01 µF 的电容。

5： 设计目标。如果无法满足该目标，可以提高最大值，但不能更改最小值。

图 27-10：　　TIMER0 和 TIMER1 外部时钟时序

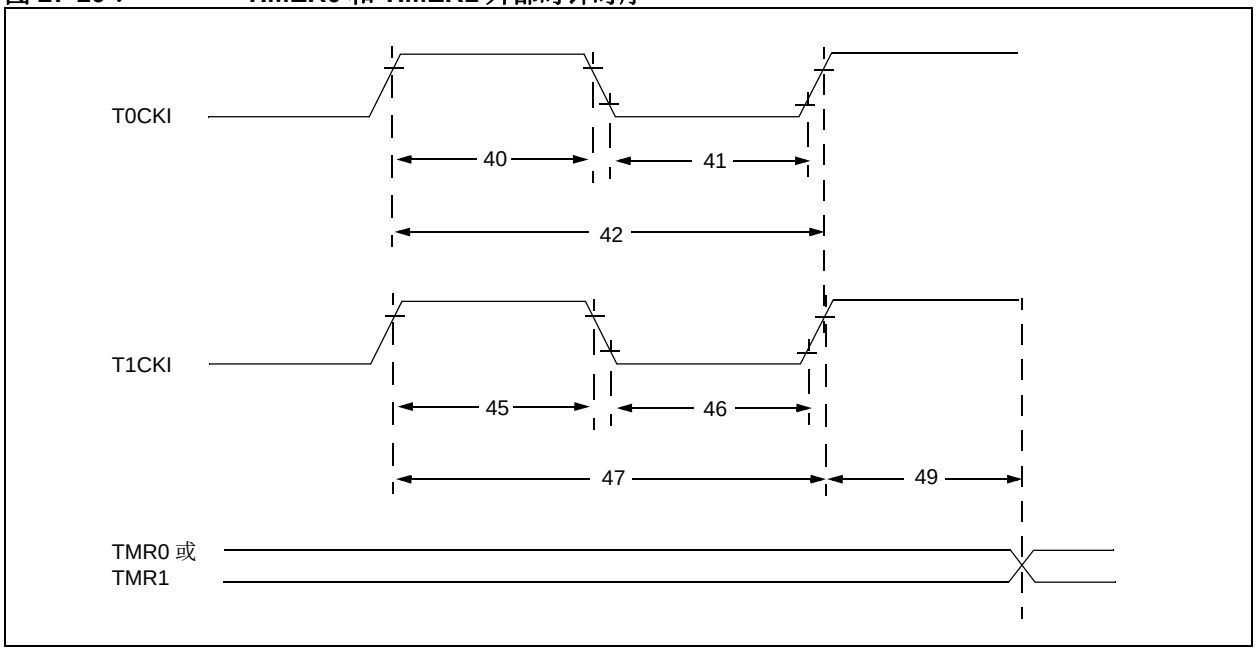


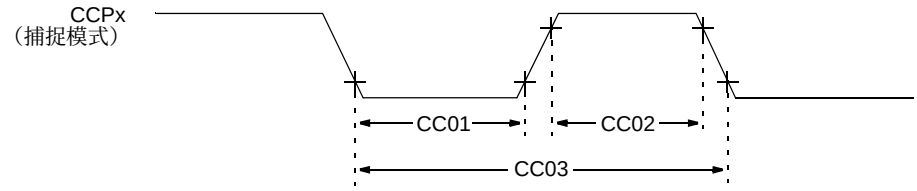
表 27-6：　　TIMER0 和 TIMER1 外部时钟要求

标准工作条件 （除非另外声明） 工作温度　　-40°C ≤ TA ≤ +125°C								
参数编号	**符号**	**特性**		**最小值**	**典型值†**	**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		取如下二者中较大值： 20 或 TCY + 40 N	—	—	ns	N = 预分频值 （2, 4, ..., 256）
45*	TT1H	T1CKI 高电平时间	同步，无预分频器	0.5 TCY + 20	—	—	ns	
			同步，带预分频器	15	—	—	ns	
			异步	30	—	—	ns	
46*	TT1L	T1CKI 低电平时间	同步，无预分频器	0.5 TCY + 20	—	—	ns	
			同步，带预分频器	15	—	—	ns	
			异步	30	—	—	ns	
47*	TT1P	T1CKI 输入周期	同步	取如下二者中较大值： 30 或 TCY + 40 N	—	—	ns	N = 预分频值 （1, 2, 4, 8）
			异步	60	—	—	ns	
48	FT1	Timer1 振荡器输入频率范围 （通过将 T1OSCEN 位置 1，使能振荡器）		32.4	32.768	33.1	kHz	
49*	TCKEZTMR1	从外部时钟边沿到定时器递增的延时		2 TOSC	—	7 TOSC	—	同步模式下的定时器

*　这些参数为特性值，未经测试。

†　除非另外声明，否则“ 典型值 ”栏中的数据均为 3V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

图 27-11： 捕捉 / 比较 /PWM 时序 （CCP）



注： 负载条件请参见图 27-2。

表 27-7： 捕捉 / 比较 /PWM 要求 （CCP）

标准工作条件 （除非另外声明） 工作温度 -40°C ≤ TA ≤ +125°C								
参数编号	**符号**	**特性**		**最小值**	**典型值 †**	**最大值**	**单位**	**条件**
CC01*	TccL	CCPx 输入低电平时间	无预分频器	0.5TCY + 20	—	—	ns	
			带预分频器	20	—	—	ns	
CC02*	TccH	CCPx 输入高电平时间	无预分频器	0.5TCY + 20	—	—	ns	
			带预分频器	20	—	—	ns	
CC03*	TccP	CCPx 输入周期		3TCY + 40 N	—	—	ns	N = 预分频值 （1、 4 或 16）

* 这些参数为特性值，未经测试。

† 除非另外声明，否则 “ 典型值 ” 栏中的数据均为 3.0V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

表 27-8： PIC18F1XK50/PIC18LF1XK50 A/D 转换器 （ADC）特性

标准工作条件 （除非另外声明） 工作温度 TA = 25°C							
参数编号	**符号**	**特性**	**最小值**	**典型值 †**	**最大值**	**单位**	**条件**
AD01	NR	分辨率	—	—	10	bit	
AD02	EIL	积分误差	—	—	±2	LSb	VREF = 3.0V
AD03	EDL	微分误差	—	—	1.5	LSb	无丢失编码 VREF = 3.0V
AD04	EOFF	失调误差	—	—	±3	LSb	VREF = 3.0V
AD05	EGN	增益误差	—	—	±3	LSb	VREF = 3.0V
AD06	VREF	参考电压的变化 = VREF+ - VREF- **(3)**	1.8	—	VDD	V	1.8 ≤ VREF+ ≤ VDD + 0.3V VSS - 0.3V ≤ VREF- ≤ VREF+ - 1.8V
AD07	VAIN	满量程	VSS	—	VREF	V	
AD08	ZAIN	模拟信号源的推荐阻抗	—	—	2.5	kΩ	如果输入引脚上有外部 0.01 µF 电容，则该值可以更高。
AD09*	IREF	VREF 输入电流 **(3)**	10	—	1000	µA	VAIN 采集期间。 基于 VHOLD 对 VAIN 的微分。
			—	—	10	µA	在 A/D 转换期间。

* 这些参数为特性值，未经测试。

† 除非另外声明，否则 “ 典型值 ” 栏中的数据均为 3.0V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

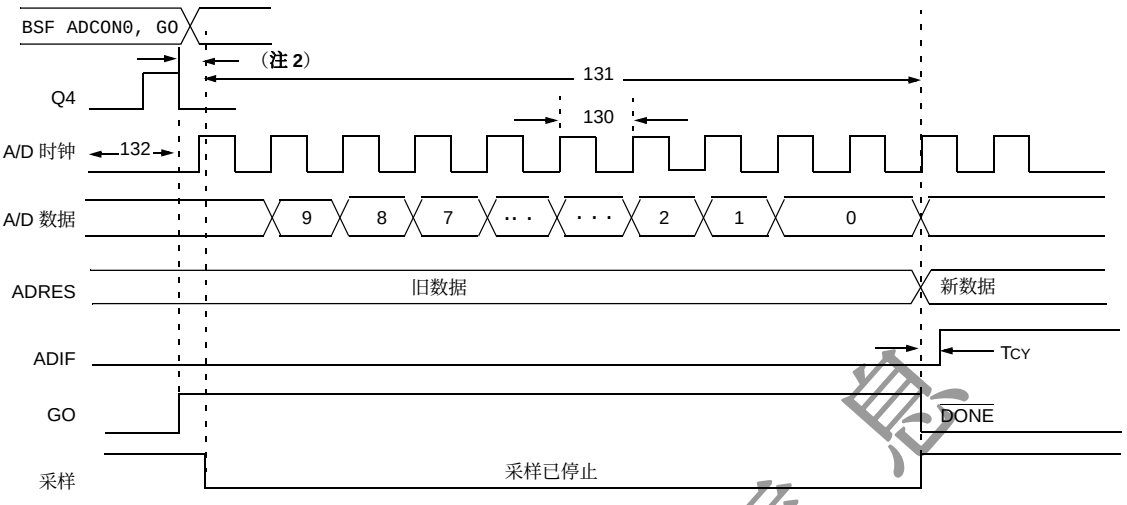
注 1： 总的绝对误差包括积分、微分、失调和增益误差。

2： 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。

3： ADC VREF 来自选择作为参考输入的外部 VREF、 VDD 引脚或 FVR。

4： 当 ADC 关闭时，它除了消耗很少的泄漏电流外，不消耗任何其他电流。掉电电流规范包括 ADC 模块消耗的任何泄漏电流。

图 27-12：　　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 27-9：　　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
			0.7	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 TBD	— —	µs µs	-40°C 至 +85°C 0°C ≤ 至 ≤ +85°C
135	TSWC	转换 → 采样的切换时间	—	（注 4）		
TBD	TDIS	电容放电时间	0.2	—	µs	

图注：　TBD = 待定
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 （*R*S）为 50Ω。
4： 在器件时钟的下一个周期。

表 27-10： 比较器规范

工作条件：1.8V < VDD < 3.6V， -40°C < TA < +125°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**
CM01	VIOFF	输入失调电压	— —	±7.5 —	±50 ±80	mV mV	高功耗模式 低功耗模式
CM02	VICM	输入共模电压	0	—	VDD	V	
CM03	CMRR	共模抑制比	55	—	—	dB	
CM04	TRESP	响应时间	—	150	400	ns	**注 1**
CM05	TMC2OV	比较器模式改变到输出有效的时间 *	—	—	10	µs	
CM06	CHYSTER	比较器滞后电压	—	65	—	mV	

* 这些参数为特性值，未经测试。

注 1： 响应时间是在比较器的一个输入端电压为 VDD/2，而另一个输入端从 VSS 跳变到 VDD 时测得的。

表 27-11： CVREF 比较器参考电压规范

工作条件：1.8V < VDD < 3.6V， -40°C < TA < +125°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**
CV01*	CLSB	步长 (2)	— —	VDD/24 VDD/32	— —	V V	低电平范围 （VRR = 1） 高电平范围 （VRR = 0）
CV02*	CACC	绝对精度	— —	— —	± 1/4 ± 1/2	LSb LSb	低电平范围 （VRR = 1） 高电平范围 （VRR = 0）
CV03*	CR	单位电阻值 （R）	—	2k	—	Ω	
CV04*	CST	稳定时间 (1)	—	—	10	µs	

* 这些参数为特性值，未经测试。

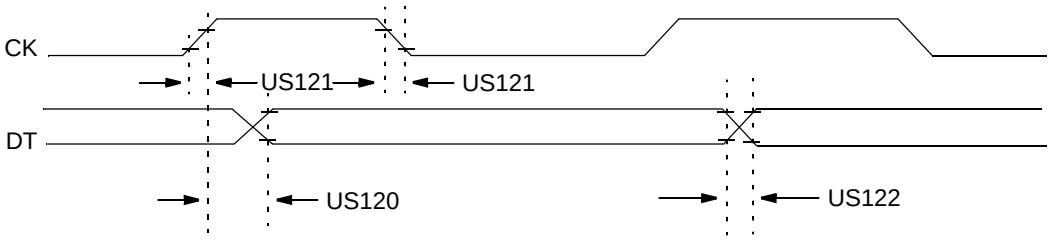
注 1： 稳定时间是在 CVRR = 1 并且 CVR3:CVR0 从 0000 跳变到 1111 时测得的。

表 27-12： 固定参考电压 （FVR）规范

工作条件：1.8V < VDD < 5.5V， -40°C < TA < +85°C （除非另外声明）。							
VR 参考电压规范			**标准工作条件 （除非另外声明）** 工作温度 -40°C ≤ TA ≤ +85°C -40°C ≤ TA ≤ +125°C				
参数编号	**符号**	**特性**	**最小值**	**典型值**	**最大值**	**单位**	**备注**
D003	VADFVR	**固定参考电压**	-7 -8 -7 -8 -7 -8	— — — — — —	6 6 6 6 6 6	%	1.024V, VDD ≥ 2.5V, 85°C 1.024V, VDD ≥ 2.5V, 125°C 2.048V, VDD ≥ 2.5V, 85°C 2.048V, VDD ≥ 2.5V, 125°C 4.096V, VDD ≥ 4.75V, 85°C 4.096V, VDD ≥ 4.75V, 125°C
D003C*	TCVFVR	**温度系数，固定参考电压**	—	-114	—	ppm/°C	
D003D*	ΔVFVR/ ΔVIN	**线路调整率，固定参考电压**	—	0.225	—	%/V	
D004*	SVDD	**VDD 上升速率，**确保内部上电复位信号	0.05	—	—	V/ms	详情请参考 **第 23.3 节 “ 上电复位 （POR）”**。

* 这些参数为特性值，未经测试。

图 27-13： USART 同步发送 （主 / 从）时序



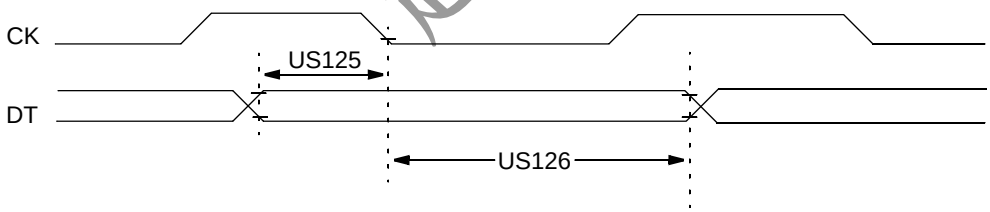
注： 负载条件请参见图 27-2。

表 27-13： USART 同步发送要求

标准工作条件 （除非另外声明）
工作温度 -40°C ≤ TA ≤ +125°C

参数编号	符号	特性		最小值	最大值	单位	条件
US120	TCKH2DTV	同步发送 （主 / 从） 时钟高电平到数据输出有效的时间	3.0-5.5V	—	80	ns	
			1.8-5.5V	—	100	ns	
US121	TCKRF	时钟输出上升时间和下降时间 （主模式）	3.0-5.5V	—	45	ns	
			1.8-5.5V	—	50	ns	
US122	TDTRF	数据输出上升时间和下降时间	3.0-5.5V	—	45	ns	
			1.8-5.5V	—	50	ns	

图 27-14： USART 同步接收 （主 / 从）时序



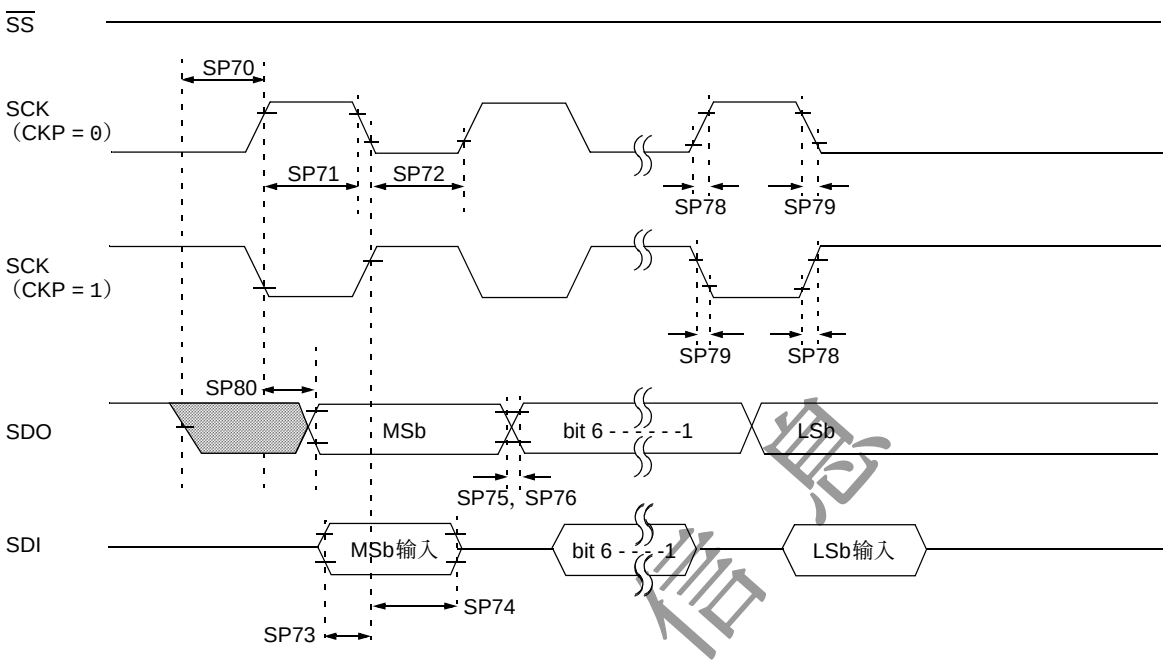
注： 负载条件请参见图 27-2。

表 27-14： USART 同步接收要求

标准工作条件 （除非另外声明）
工作温度 -40°C ≤ TA ≤ +125°C

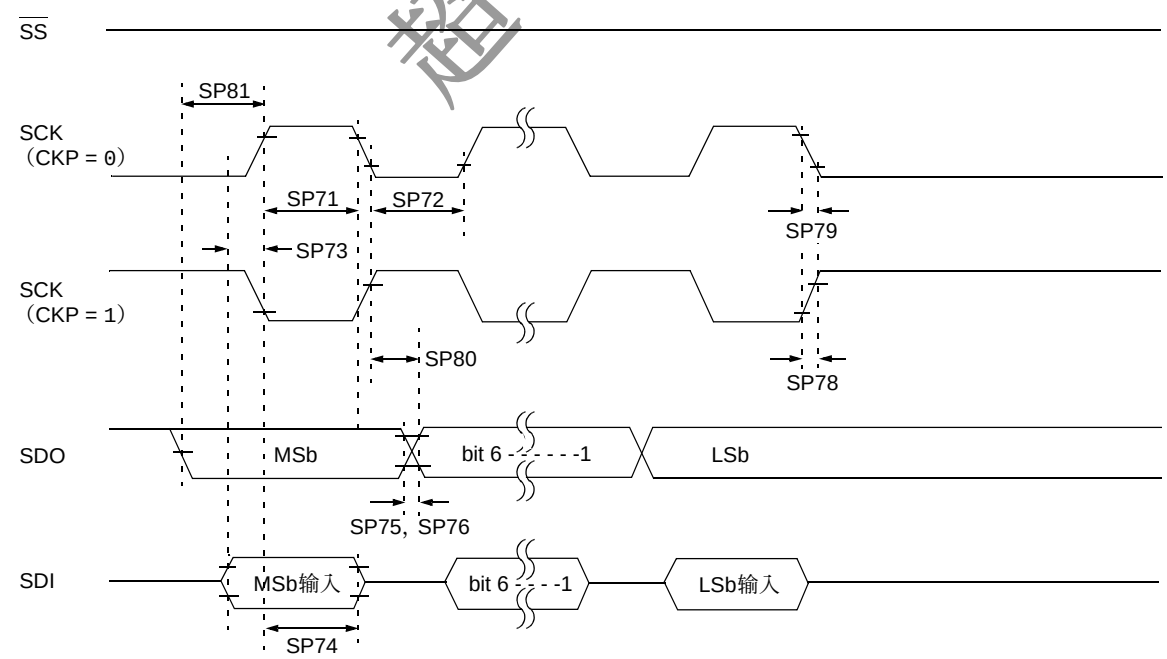
参数编号	符号	特性	最小值	最大值	单位	条件
US125	TDTV2CKL	同步接收 （主 / 从） CK ↓ 前的数据保持时间 （DT 保持时间）	10	—	ns	
US126	TCKL2DTL	CK ↓ 后的数据保持时间 （DT 保持时间）	15	—	ns	

图 27-15： SPI 主模式时序 （CKE = 0， SMP = 0）



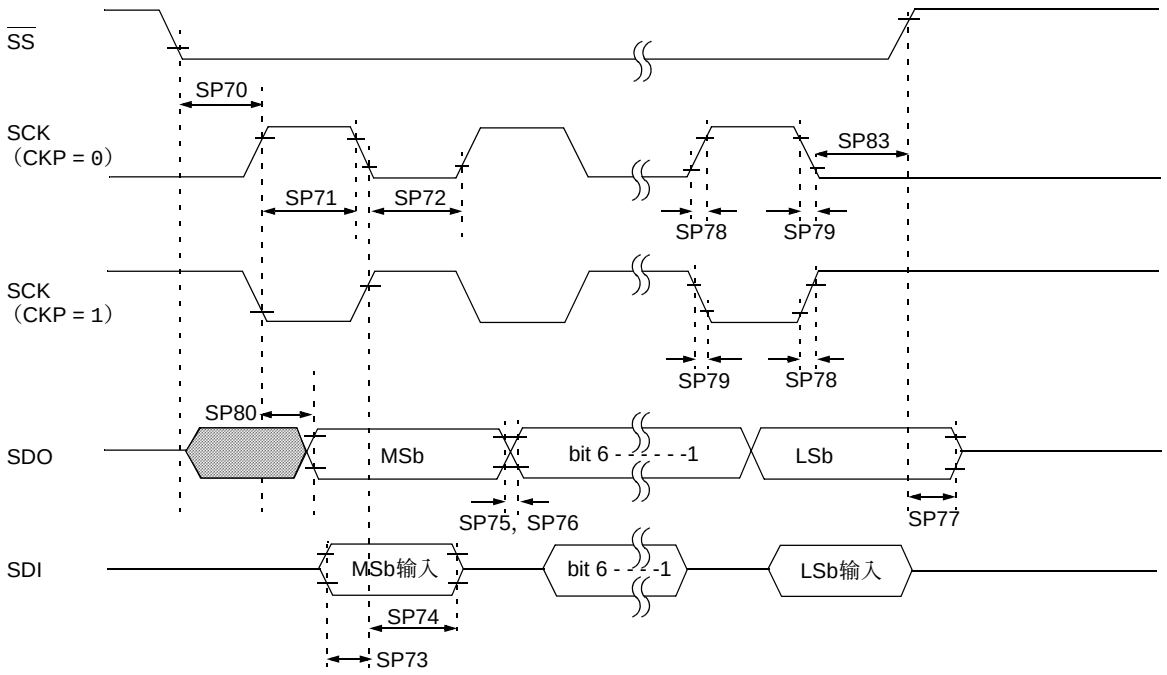
注：负载条件请参见图 27-2。

图 27-16： SPI 主模式时序 （CKE = 1， SMP = 1）



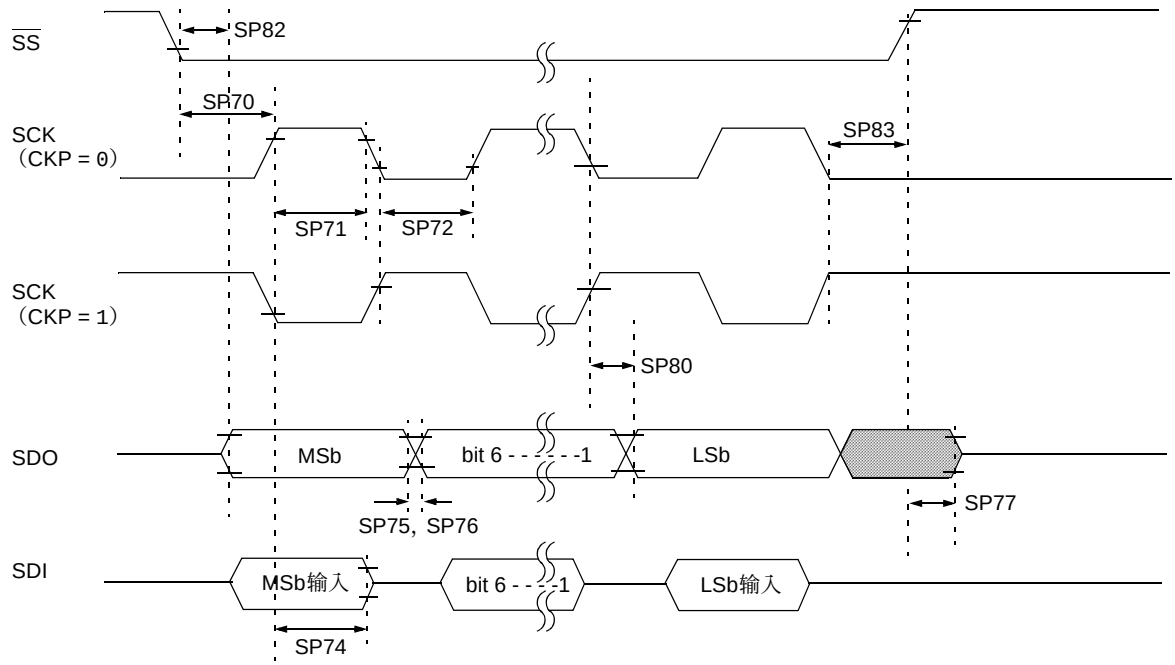
注：负载条件请参见图 27-2。

图 27-17：　　SPI 从模式时序 （CKE = 0）



注：负载条件请参见图 27-2。

图 27-18：　　SPI 从模式时序 （CKE = 1）



注：负载条件请参见图 27-2。

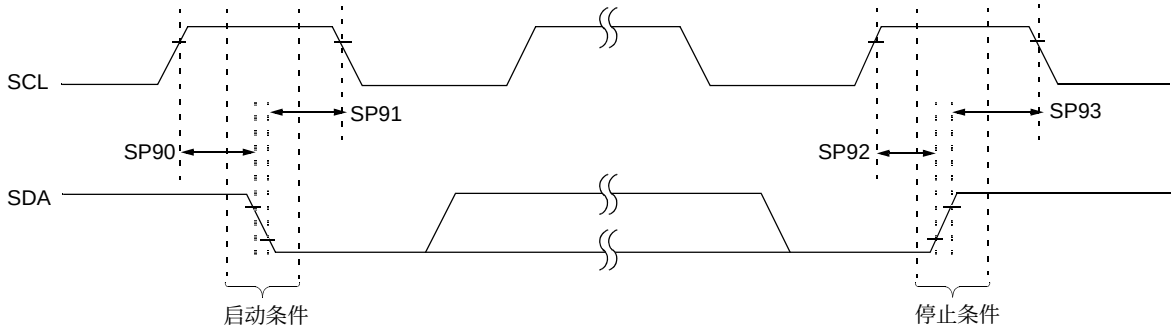
表 27-15： SPI 模式要求

参数编号	符号	特性		最小值	典型值 †	最大值	单位	条件
SP70*	TssL2scH, TssL2scL	$\overline{SS}$↓ 到 SCK↓ 或 SCK↑ 输入的时间		Tcy	—	—	ns	
SP71*	TscH	SCK 输入高电平时间 （从模式）		Tcy + 20	—	—	ns	
SP72*	TscL	SCK 输入低电平时间 （从模式）		Tcy + 20	—	—	ns	
SP73*	TdiV2scH, TdiV2scL	SDI 数据输入到 SCK 边沿的建立时间		100	—	—	ns	
SP74*	TscH2diL, TscL2diL	SDI 数据输入到 SCK 边沿的保持时间		100	—	—	ns	
SP75*	TdoR	SDO 数据输出上升时间	3.0-5.5V	—	10	25	ns	
			1.8-5.5V	—	25	50	ns	
SP76*	TdoF	SDO 数据输出下降时间		—	10	25	ns	
SP77*	TssH2doZ	$\overline{SS}$↑ 到 SDO 输出高阻抗的时间		10	—	50	ns	
SP78*	TscR	SCK 输出上升时间 （主模式）	3.0-5.5V	—	10	25	ns	
			1.8-5.5V	—	25	50	ns	
SP79*	TscF	SCK 输出下降时间 （主模式）		—	10	25	ns	
SP80*	TscH2doV, TscL2doV	SCK 边沿后 SDO 数据输出有效的时间	3.0-5.5V	—	—	50	ns	
			1.8-5.5V	—	—	145	ns	
SP81*	TdoV2scH, TdoV2scL	SDO 数据输出建立到 SCK 边沿的时间		Tcy	—	—	ns	
SP82*	TssL2doV	$\overline{SS}$↓ 边沿后 SDO 数据输出有效的时间		—	—	50	ns	
SP83*	TscH2ssH, TscL2ssH	SCK 边沿后出现 $\overline{SS}$↑ 的时间		1.5Tcy + 40	—	—	ns	

* 这些参数为特性值，未经测试。

† 除非另外声明，否则 “ 典型值 ” 栏中的数据均为 3.0V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

图 27-19： I^2C™ 总线启动位 / 停止位时序



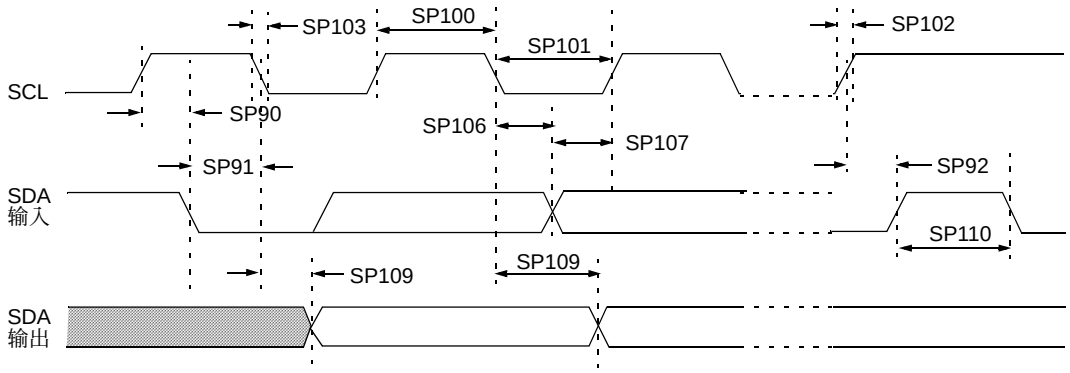
注：负载条件请参见图 27-2。

表 27-16： I²C™ 总线启动位 / 停止位要求

参数编号	符号	特性		最小值	典型值	最大值	单位	条件
SP90*	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	—	—		
SP91*	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	—	—		
SP92*	TSU:STO	停止条件建立时间	100 kHz 模式	4700	—	—	ns	
			400 kHz 模式	600	—	—		
SP93	THD:STO	停止条件保持时间	100 kHz 模式	4000	—	—	ns	
			400 kHz 模式	600	—	—		

* 这些参数为特性值，未经测试。

图 27-20： I²C™ 总线数据时序



注：负载条件请参见图 27-2。

表 27-17： I^2C™ 总线数据要求

参数编号	符号	特性		最小值	最大值	单位	条件
SP100*	THIGH	时钟高电平时间	100 kHz 模式	4.0	—	µs	器件工作频率不得低于 1.5 MHz
			400 kHz 模式	0.6	—	µs	器件工作频率不得低于 10 MHz
			SSP 模块	1.5TCY	—		
SP101*	TLOW	时钟低电平时间	100 kHz 模式	4.7	—	µs	器件工作频率不得低于 1.5 MHz
			400 kHz 模式	1.3	—	µs	器件工作频率不得低于 10 MHz
			SSP 模块	1.5TCY	—	—	
SP102*	TR	SDA 和 SCL 上升时间	100 kHz 模式	—	1000	ns	
			400 kHz 模式	20 + 0.1CB	300	ns	CB 值规定在 10-400 pF 之间
SP103*	TF	SDA 和 SCL 下降时间	100 kHz 模式	—	250	ns	
			400 kHz 模式	20 + 0.1CB	250	ns	CB 值规定在 10-400 pF 之间
SP90*	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	
SP91*	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	
SP106*	THD:DAT	数据输入保持时间	100 kHz 模式	0	—	ns	
			400 kHz 模式	0	0.9	µs	
SP107*	TSU:DAT	数据输入建立时间	100 kHz 模式	250	—	ns	**（注 2）**
			400 kHz 模式	100	—	ns	
SP92*	TSU:STO	停止条件建立时间	100 kHz 模式	4.7	—	µs	
			400 kHz 模式	0.6	—	µs	
SP109*	TAA	时钟输出有效时间	100 kHz 模式	—	3500	ns	**（注 1）**
			400 kHz 模式	—	—	ns	
SP110*	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	
SP	CB	总线容性负载		—	400	pF	

* 这些参数为特性值，但未经测试。

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覆盖 SCL 下降沿的未定义区域 （最小值 300 ns）。

2： 快速模式 （400 kHz）的 I^2C 总线器件也可在标准模式 （100 kHz）的 I^2C 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，其下一个数据位必须输出到 SDA 线。 SCL 线被释放前，根据标准模式 I^2C 总线规范， TR max. + TSU:DAT = 1000 + 250 = 1250 ns。

注：

28.0 直流和交流特性图表

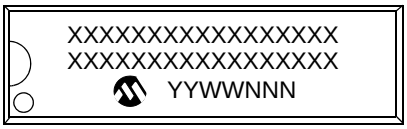
当前没有可用图表。

注：

29.0 封装信息

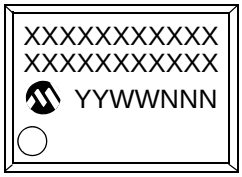
29.1 封装标识信息

20 引脚 PDIP



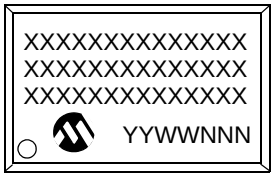
示例

20 引脚 SSOP



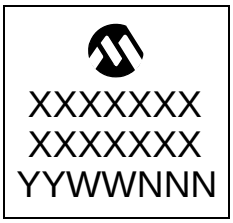
示例

20 引脚 SOIC（0.300 英寸）



示例

20 引脚 QFN



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。

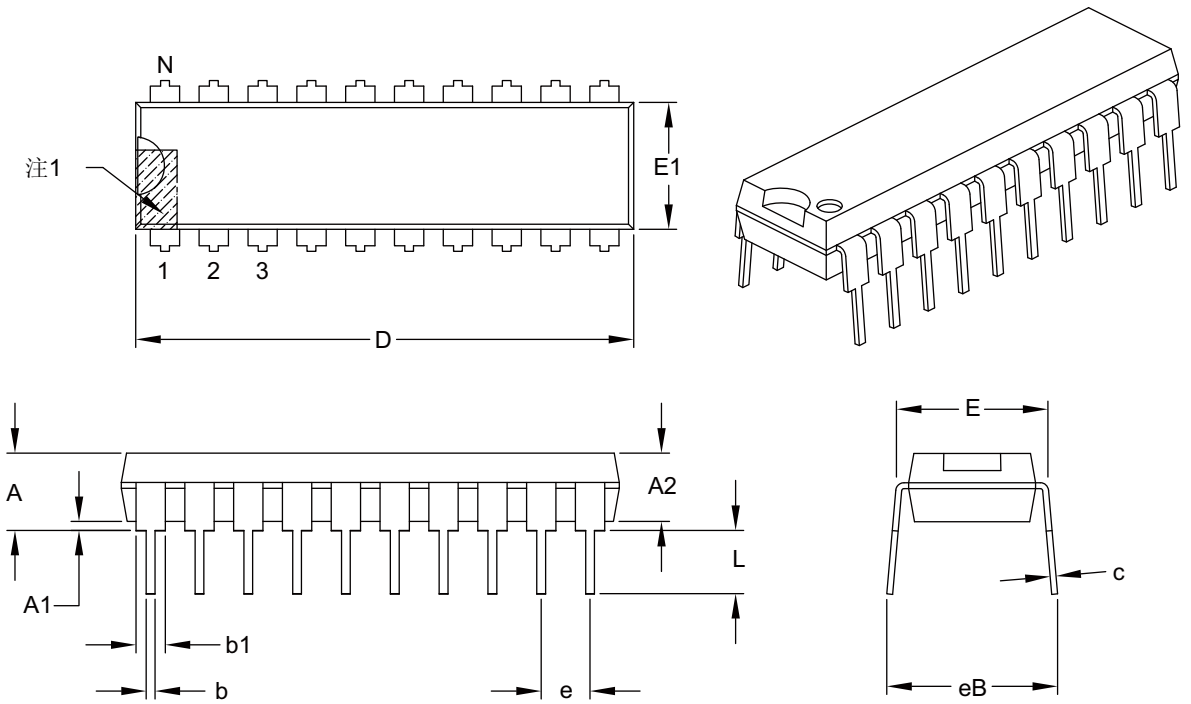
注： Microchip 元器件编号如果无法在同一行内完整标注，将换行标出，因此会限制表示客户信息的字符数。

29.2 封装详细信息

以下部分将介绍各种封装的技术细节。

20引脚塑封双列直插式封装（**P**）——主体**300 mil [PDIP]**

注： 最新封装图请至http://www.microchip.com/packaging查看Microchip封装规范。



单位		英寸		
尺寸范围		最小	正常	最大
引脚数	N	20		
引脚间距	e	.100 BSC		
顶端到固定面高度	A	–	–	.210
塑模封装厚度	A2	.115	.130	.195
塑模底面到固定面高度	A1	.015	–	–
肩到肩宽度	E	.300	.310	.325
塑模封装宽度	E1	.240	.250	.280
总长度	D	.980	1.030	1.060
引脚尖到固定面高度	L	.115	.130	.150
引脚厚度	c	.008	.010	.015
引脚上部宽度	b1	.045	.060	.070
引脚下部宽度	b	.014	.018	.022
总排列间距 §	eB	–	–	.430

注：

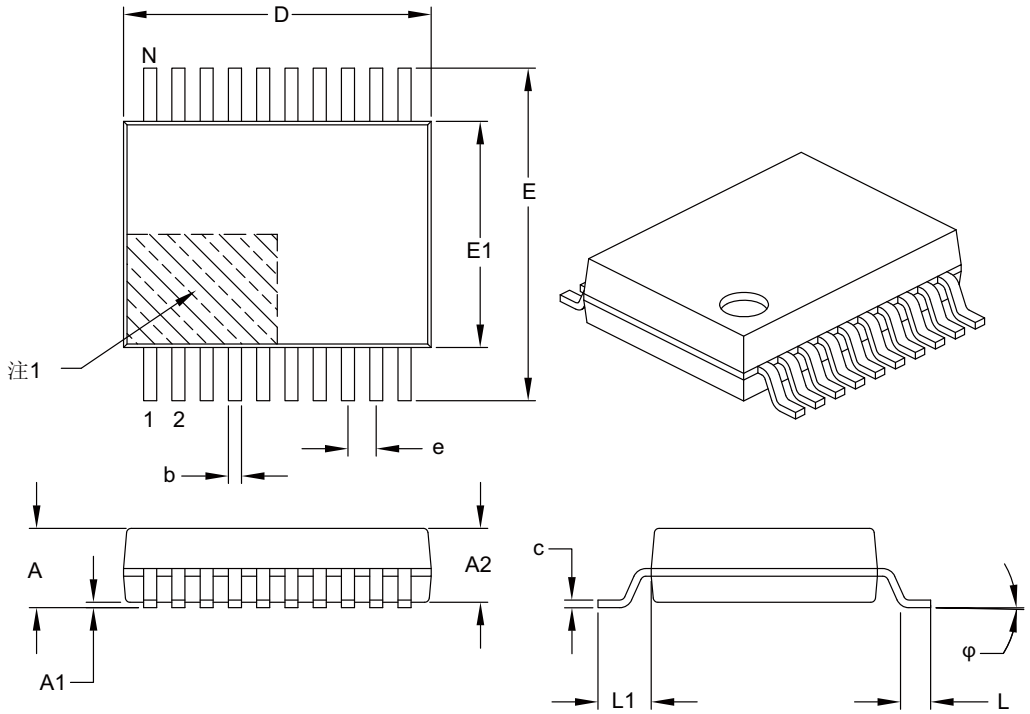
1. 引脚1的可见定位标记可能不同，但必须在阴影区域内。
2. § 重要特性。
3. 尺寸D和E1不包括塑模毛边或突起。塑模每侧的毛边或突起不得超过0.010英寸。
4. 尺寸和公差遵循ASME Y14.5M。

BSC：基本尺寸。显示的是没有公差的理论精确值。

Microchip Technology图号C04-019B

20引脚塑封缩小型小外形封装（**SS**）——主体**5.30 mm [SSOP]**

注： 最新封装图请至http://www.microchip.com/packaging查看Microchip封装规范。



单位		毫米		
尺寸范围		最小	正常	最大
引脚数	N		20	
引脚间距	e		0.65 BSC	
总高度	A	–	–	2.00
塑模封装厚度	A2	1.65	1.75	1.85
悬空间隙	A1	0.05	–	–
总宽度	E	7.40	7.80	8.20
塑模封装宽度	E1	5.00	5.30	5.60
总长度	D	6.90	7.20	7.50
底脚长度	L	0.55	0.75	0.95
引脚投影长度	L1		1.25 REF	
引脚厚度	c	0.09	–	0.25
底脚倾斜角	ϕ	0°	4°	8°
引脚宽度	b	0.22	–	0.38

注：

1. 引脚1的可见定位标记可能不同，但必须在阴影区域内。
2. 尺寸D和E1不包括塑模毛边或突起。塑模每侧的毛边或突起不得超过0.20毫米。
3. 尺寸和公差遵循ASME Y14.5M。

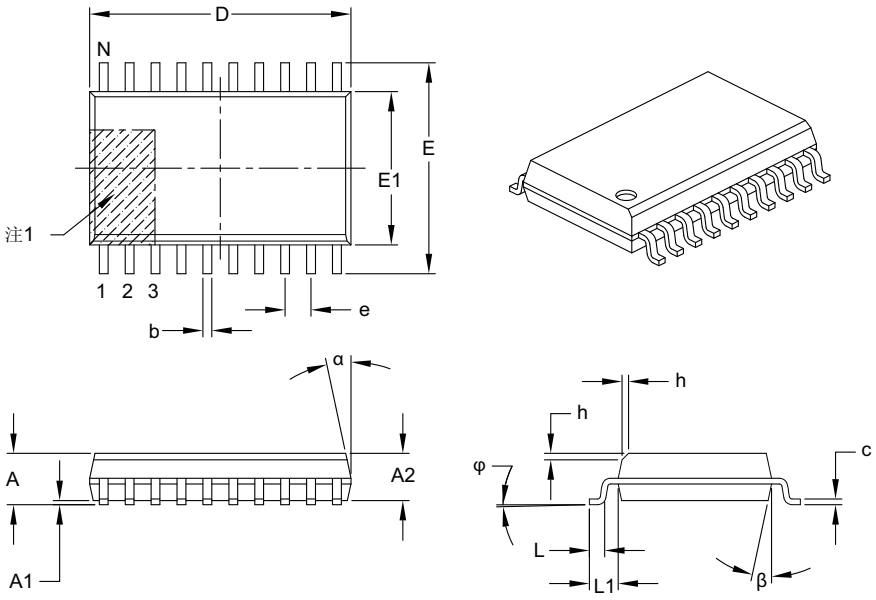
BSC：基本尺寸。显示的是没有公差的理论精确值。

REF：参考尺寸，通常无公差，仅供参考。

Microchip Technology图号C04-072B

20引脚塑封宽条小外形封装（**SO**）——主体**7.50 mm [SOIC]**

注： 最新封装图请至http://www.microchip.com/packaging查看Microchip封装规范。



单位		毫米		
尺寸范围		最小	正常	最大
引脚数	N		20	
引脚间距	e		1.27 BSC	
总高度	A	–	–	2.65
塑模封装厚度	A2	2.05	–	–
悬空间隙 §	A1	0.10	–	0.30
总宽度	E		10.30 BSC	
塑模封装宽度	E1		7.50 BSC	
总长度	D		12.80 BSC	
倒棱距离（可选）	h	0.25	–	0.75
底脚长度	L	0.40	–	1.27
引脚投影长度	L1		1.40 REF	
底脚倾斜角	ϕ	0°	–	8°
引脚厚度	c	0.20	–	0.33
引脚宽度	b	0.31	–	0.51
塑模顶部锥度	α	5°	–	15°
塑模底部锥度	β	5°	–	15°

注：

1. 引脚1的可见定位标记可能不同，但必须在阴影区域内。
2. § 重要特性。
3. 尺寸D和E1不包括塑模毛边或突起。塑模每侧的毛边或突起不得超过0.15毫米。
4. 尺寸和公差遵循ASME Y14.5M。

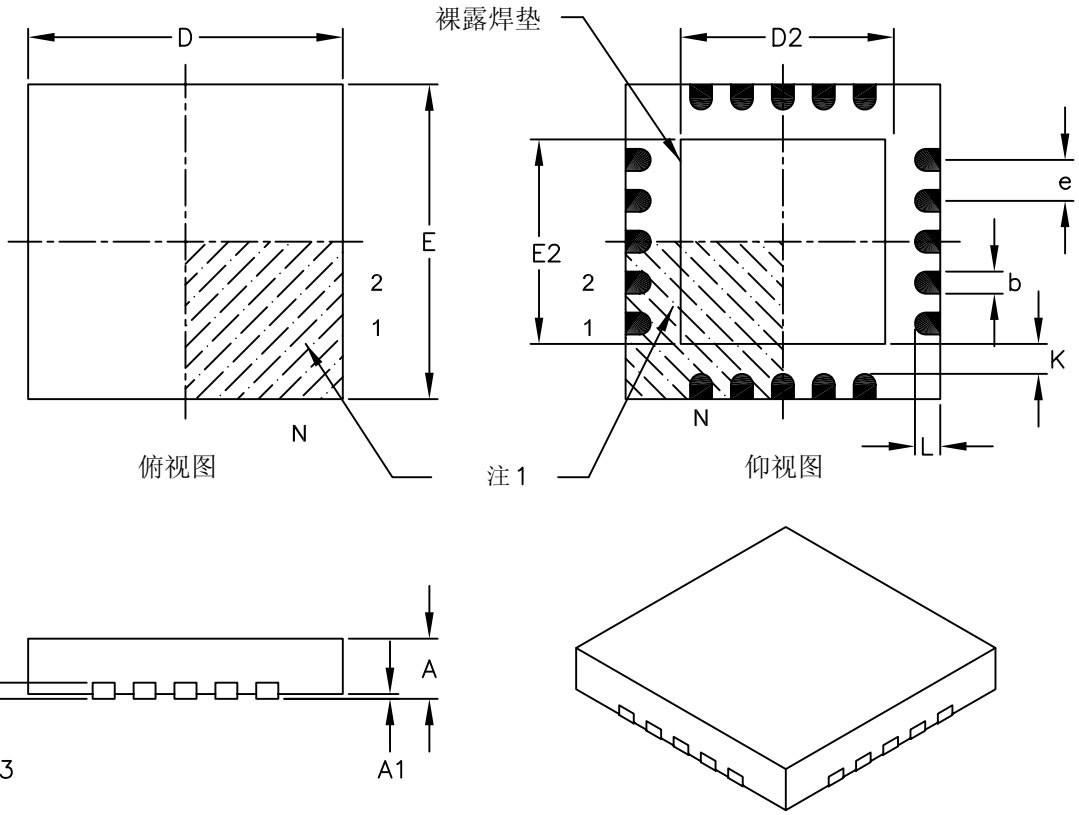
BSC：基本尺寸。显示的是没有公差的理论精确值。

REF：参考尺寸，通常无公差，仅供参考。

Microchip Technology图号C04-094B

20引脚塑封正方扁平无脚封装（**MQ**）—— 主体**5x5x0.9 mm[QFN]**

注：最新封装图请至http://www.microchip.com/packaging查看Microchip封装规范。



单位		毫米		
尺寸范围		最小	正常	最大
引脚数	N	20		
引脚间距	e	0.65 BSC		
总高度	A	0.80	0.90	1.00
悬空间隙	A1	0.00	0.02	0.05
触点厚度	A3	0.20 REF		
总宽度	E	5.00 BSC		
裸露金属焊垫宽度	E2	3.15	3.25	3.35
总长度	D	5.00 BSC		
裸露金属焊垫长度	D2	3.15	3.25	3.35
触点宽度	b	0.25	0.30	0.35
触点长度	L	0.35	0.40	0.45
触点到裸露金属焊垫的距离	K	0.20	-	-

注：

1. 引脚1的可见定位标记可能变化，但必须位于阴影区域内。
2. 封装为切割分离。
3. 尺寸和公差遵循ASME Y14.5M。

BSC：基本尺寸。显示的是没有公差的理论精确值。
REF：参考尺寸，通常无公差，仅供参考。

Microchip Technology 图号 C04-139B

注：

附录 A：版本历史

版本 A （2008 年 5 月）

PIC18F1XK50/PIC18LF1XK50 器件的原始数据手册。

版本 B （2008 年 6 月）

修改了 27.4 直流特性表。

版本 C （2009 年 4 月）

修改了数据手册标题以及与某些功能特性有关的章节。修改了表 1-2、表 3-1 和表 3-2。在第 9.1 节增加了注 3。修改了寄存器 14-1、例 16-1、第 18.8.4 节、寄存器 18-3、表 20-2、第 22.2.1 节、第 22.2.2 节、第 22.5.1.1 节、第 22.7 节，以及表 23-4、表 27-1、表 27-2、表 27-3、表 27-4 和表 27-8。

版本 D （2010 年 5 月）

修改了 20 引脚 PDIP、SSOP 和 SOIC 封装图；增加了 20 引脚 QFN 封装图；修改了表 1 和表 1-1 ；修改了图 2-1 ；增加了第 2.11.1 节 （低速操作）中的注；修改了表 3-1 和表 3-2 ；修改了第 4 章 （闪存程序存储器）和第 5 章 （数据 EEPROM 存储器）；修改了例 5-2 和表 5-1 ；删除了寄存器 7-4 和 7-8 中的注 1 ；修改了表 9-1 和 9-3 ；修改了第 14.1 节（ECCP 输出和配置）和第 14.4.4 节 （增强型 PWM 自动关闭模式）；增加了寄存器 14-2 中的注 4 ；修改了图 14-10 ；修改了公式 17-1 ；修改了表 18-3 和表 20-3 ；修改了公式 21-1 ；删除了第 21.1.3 节（输出钳位到 VSS）；修改了图 21-1 ；修改了表 21-1、表 23-4 和表 24-1 ；增加了表 24-1 中的注 2 ；修改了寄存器 24-6 ；删除了表 24-3 中的注 1 ；修改了第 27 章（多个表格中的内容）；增加了 20 引脚 QFN 封装标识信息和封装详细信息；修改了产品标识体系部分；其他较小更正。

版本 E （2010 年 10 月）

更新了第 27.0 章“ 电气规范 ”。

附录 B：器件差异

表 B-1 为本数据手册中所列器件间的差异。

表 B-1：器件差异

特性	PIC18F13K50	PIC18F14K50	PIC18LF13K50	PIC18F26K20	PIC18LF14K50	PIC18F44K20	PIC18F45K20	PIC18F46K20
程序存储器（字节）	8192	16384	32768	65536	8192	16384	32768	65536
程序存储器（指令）	4096	8192	16384	32768	4096	8192	16384	32768
中断源	19	19	19	19	20	20	20	20
I/O 端口	端口 A、B、C 和（E）	端口 A、B、C 和（E）	端口 A、B、C 和（E）	端口 A、B、C 和（E）	端口 A、B、C、D 和 E	端口 A、B、C、D 和 E	端口 A、B、C、D 和 E	端口 A、B、C、D 和 E
捕捉 / 比较 /PWM 模块	1	1	1	1	1	1	1	1
增强型捕捉 / 比较 / PWM 模块	1	1	1	1	1	1	1	1
并行通信（PSP）	无	无	无	无	有	有	有	有
10 位模数转换模块	11 路输入通道	11 路输入通道	11 路输入通道	11 路输入通道	14 路输入通道	14 路输入通道	14 路输入通道	14 路输入通道
封装	20 引脚 PDIP 20 引脚 SOIC 20 引脚 SSOP 20 引脚 QFN	20 引脚 PDIP 20 引脚 SOIC 20 引脚 SSOP 20 引脚 QFN	20 引脚 PDIP 20 引脚 SOIC 20 引脚 SSOP 20 引脚 QFN	28 引脚 PDIP 28 引脚 SOIC 28 引脚 SSOP 28 引脚 QFN	20 引脚 PDIP 20 引脚 SOIC 20 引脚 SSOP 20 引脚 QFN	40 引脚 PDIP 44 引脚 TQFP 44 引脚 QFN	40 引脚 PDIP 44 引脚 TQFP 44 引脚 QFN	40 引脚 PDIP 44 引脚 TQFP 44 引脚 QFN

索引

D

E

F

G

H

I

J

K

T

U

V

W

X

Y

Z

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： PIC18F/LF1XK50　　文献编号： DS41350E_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。

部件编号	X	X	/XX	XXX
器件	封装选择	温度范围	封装	模式

器件： PIC18F13K50(1)、PIC18F14K50(1)、PIC18LF13K50(1) 和 PIC18LF14K50

封装选择： 空白 = 标准封装（管式或托盘式）
T = 卷带式(1)

温度范围： E = -40°C 至 +125°C（扩展级）
I = -40°C 至 +85°C（工业级）

封装： P = PDIP
SO = SOIC
SS = SSOP
MQ = QFN

模式： QTP、SQTP、代码或特殊要求
（其他情况空白）

示例：

a) PIC18F14K50-E/P 301 = 扩展级温度，PDIP 封装，扩展 VDD 范围，QTP 模式 #301。
b) PIC18LF14K50-E/SO = 扩展级温度，SOIC 封装。
c) PIC18LF14K50-E/P = 扩展级温度，PDIP 封装。
d) PIC18LF14K50-E/MQ = 扩展级温度，QFN 封装。
e) PIC18F14K50-I/P = 工业级温度，PDIP 封装。

注 1： T = 卷带式封装只适用于 ML、MV、PT、SO 和 SS 封装的工业级温度范围的器件。

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: 86-10-8528-2100
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 - 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

台湾地区 - 高雄
Tel: 886-7-213-7830
Fax: 886-7-330-9305

台湾地区 - 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

亚太地区

台湾地区 - 新竹
Tel: 886-3-6578-300
Fax: 886-3-6578-370

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

02/18/11

初稿