简介

CN5225 单片机系列拥有一系列数字外设和模拟外设，有助于实现成本敏感型传感器和实时控制应用。该产品系列采用 14 引脚 TSSOP 封装，存储器容量为 3.5 KB（CN5223）或 14 KB（CN5225），速度最高可达 32 MHz。该系列配备 10 位模数转换器（Analog-to-Digital Converter，ADC）、外设引脚选择（Peripheral Pin Select，PPS）功能、帮助用户实现数据保护和自举程序应用的存储器访问分区（Memory Access Partition，MAP）、数字通信外设、定时器以及波形发生器。这种小尺寸封装器件非常适合低成本传感器和控制应用。

CN5225 系列汇总

表 1.

器件	闪存程序存储器（字节）	数据 SRAM（字节）	存储器访问分区	I/O 引脚[1]/外设引脚选择	带 HLT 的 8 位定时器/16 位定时器[2]	10 位 PWM/CCP	10 位 ADC 通道（外部/内部）	MSSP	EUSART	SMBus™兼容型 I/O 焊盘	外部中断引脚	电平变化中断引脚	看门狗定时器
CN5223	3.5K	256	Y	12/Y	1/2	2/2	9/1	1	1	N	1	12	Y
CN5225	14K	1K	Y	12/Y	1/2	2/2	9/1	1	1	N	1	12	Y

注：

1. 总 I/O 数包含一个仅用作输入的引脚（$\overline{\text{MCLR}}$）。
2. Timer0 可配置为 8 位或 16 位定时器。
3. 要对 CN5223 进行编程，请选择 PIC16F15223。
4. 要对 CN5225 进行编程，请选择 PIC16F15225。

内核特性

- 为 C 编译器优化的 RISC 架构
- 工作速度：
  - DC – 32 MHz 时钟输入
  - 最小指令时间为 125 ns
- 16 级硬件堆栈
- 低电流上电复位（Power-on Reset，POR）
- 可配置上电延时定时器（Power-up Timer，PWRT）
- 欠压复位（Brown-out Reset，BOR）
- 看门狗定时器（Watchdog Timer，WDT）

存储器

- 最大 14 KB 的闪存程序存储器

- 最大 1 KB 的数据 SRAM 存储器
- 存储器访问分区（MAP）：闪存程序存储器可划分为：
  - 应用程序块
  - 引导块
  - 存储区闪存（Storage Area Flash，SAF）块
- 可编程代码保护和写保护
- 器件特性区（Device Characteristics Area，DCA），用于存储：
  - 编程/擦除行大小
  - 引脚数详细信息
- 直接、间接和相对寻址模式

工作特性

- 工作电压范围：
  - 1.8V 至 5.5V
- 温度范围：
  - 工业级：-40°C 至 85°C

节能功能

- 休眠：
  - 降低器件功耗
  - 在执行 ADC 转换时降低系统电气噪声
- 低功耗模式特性：
  - 休眠：
    - 3V/25°C（使能 WDT）时的典型值 < 900 nA
    - 3V/25°C（禁止 WDT）时的典型值 < 600 nA
  - 工作电流：
    - 32 kHz、3V/25°C 时的典型值为 48 µA
    - 4 MHz、5V/25°C 时的典型值 < 1 mA

数字外设

- 两个捕捉/比较/PWM（Capture/Compare/PWM，CCP）模块：
  - 捕捉/比较模式的分辨率为 16 位
  - PWM 模式的分辨率为 10 位
- 两个脉宽调制器（Pulse-Width Modulator，PWM）：
  - 10 位分辨率
  - 独立脉冲输出
- 一个可配置的 8/16 位定时器（TMR0）
- 一个带门控的 16 位定时器（TMR1）
- 一个带硬件限制定时器（Hardware Limit Timer，HLT）的 8 位定时器（TMR2）
- 一个增强型通用同步/异步收发器（Enhanced Universal Synchronous Asynchronous Receiver Transmitter，EUSART）：
  - 兼容 RS-232、RS-485 和 LIN

  - 接收到启动字符时自动唤醒
- 一个主同步串行端口（Host Synchronous Serial Port，MSSP）：
  - 串行外设接口（Serial Peripheral Interface，SPI）模式
    - 从选择同步
  - I^2C 模式
    - 7/10 位寻址模式
- 外设引脚选择（PPS）：
  - 支持数字 I/O 的引脚映射
- 器件 I/O 端口特性：
  - 12 个 I/O 引脚
  - 1 个仅输入引脚（RA3）
  - 单独控制 I/O 方向、漏极开路、输入阈值、压摆率和弱上拉
  - 所有 I/O 引脚上均具有电平变化中断（Interrupt-On-Change，IOC）功能
  - 1 个外部中断引脚

模拟外设

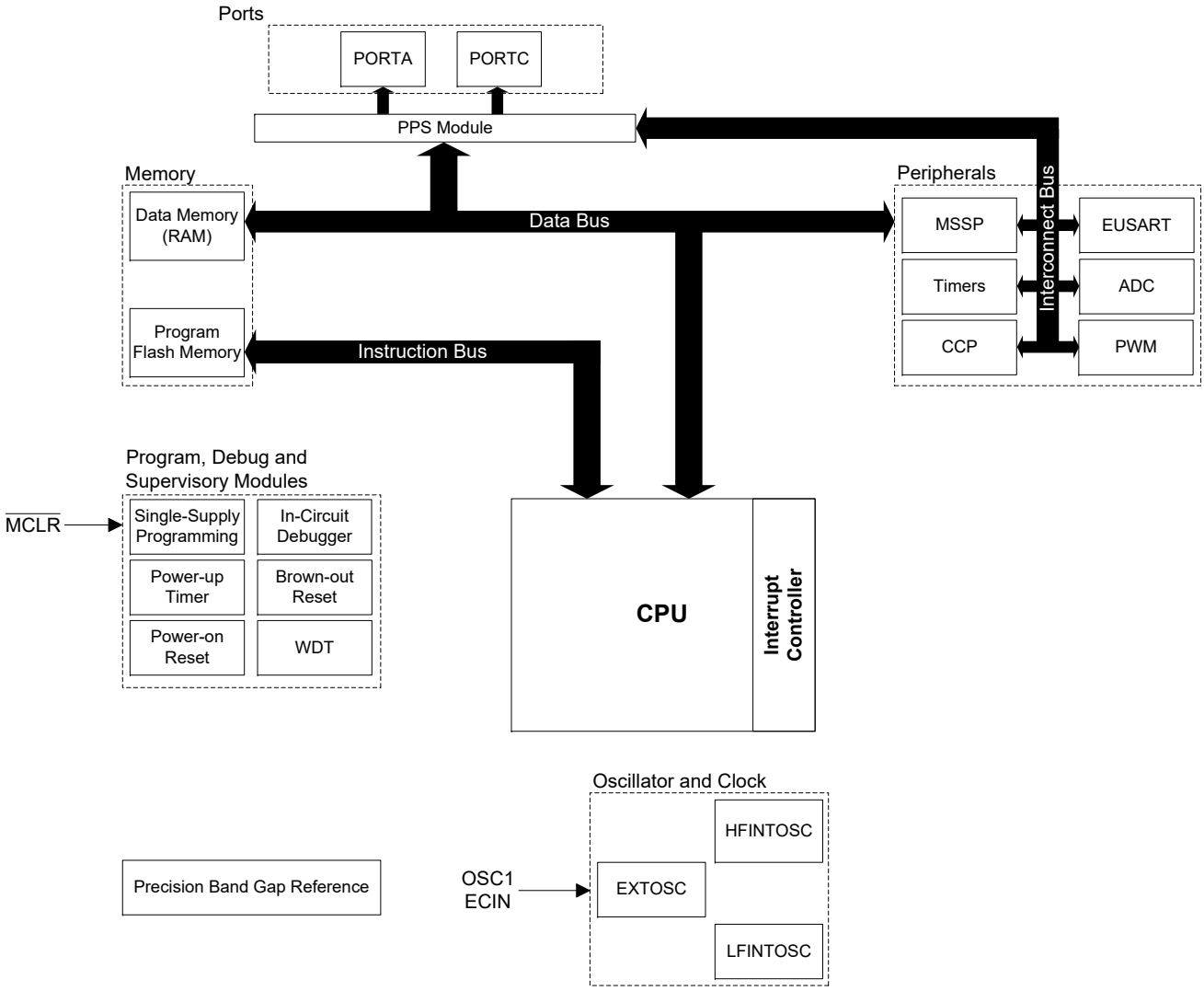
- 模数转换器（ADC）：
  - 10 位分辨率
  - 9 个外部输入通道
  - 1 个内部输入通道
  - 内部 ADC 振荡器（ADCRC）
  - 可在休眠模式下工作
  - 可选自动转换触发源

时钟结构

- 高精度内部振荡器模块（HFINTOSC）：
  - 可选择最高 32 MHz 的频率
  - 校准精度±2%
- 内部 31 kHz 振荡器（LFINTOSC）
- 外部高频时钟输入：
  - 两种外部时钟（External Clock，EC）功耗模式

框图

图 1. CN5223/25 框图



目录

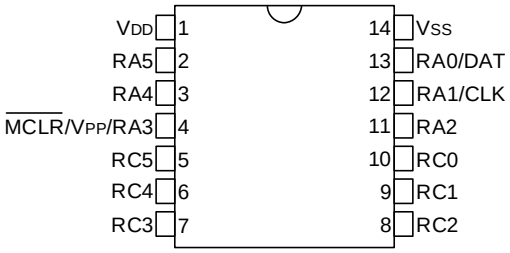
1. 封装

表 1-1. 封装

器件	14 引脚 TSSOP
CN5223	•
CN5225	•

2. 引脚图

图 2-1. 14-Pin TSSOP



3. 引脚分配表

表 3-1. 14 引脚分配表

I/O	14 引脚 TSSOP	ADC	参考电压	定时器	CCP	10 位 PWM	MSSP	EUSART	IOC	中断	基本连接
RA0	13	ANA0	—	—	—	—	—	—	IOCA0	—	ICSPDAT ICDDAT
RA1	12	ANA1	V_{REF}+（ADC）	—	—	—	—	—	IOCA1	—	ICSPCLK ICDCLK
RA2	11	ANA2	—	T0CKI(1)	—	—	—	—	IOCA2	INT(1)	—
RA3	4	—	—	—	—	—	—	—	IOCA3	—	$\overline{MCLR}$ V_{PP}
RA4	3	ANA4	—	T1G(1)	—	—	—	—	IOCA4	—	CLKOUT
RA5	2	ANA5	—	T1CKI(1) T2IN(1)	—	—	—	—	IOCA5	—	CLKIN
RC0	10	—	—	—	—	—	SCL1(1,3,4) SCK1(1,3,4)	—	IOCC0	—	—
RC1	9	—	—	—	—	—	SDA1(1,3,4) SDI1(1,3,4)	—	IOCC1	—	—
RC2	8	ANC2 ADACT(1)	—	—	—	—	—	—	IOCC2	—	—
RC3	7	ANC3	—	—	CCP2(1)	—	$\overline{SS1}$(1)	—	IOCC3	—	—
RC4	6	ANC4	—	—	—	—	—	CK1(1,3)	IOCC4	—	—
RC5	5	ANC5	—	—	CCP1(1)	—	—	RX1(1) DT1(1, 3)	IOCC5	—	—
V_{DD}	1	—	—	—	—	—	—	—	—	—	V_{DD}
V_{SS}	14	—	—	—	—	—	—	—	—	—	V_{SS}
OUT(2)	—	—	—	TMR0	CCP1 CCP2	PWM3 PWM4	SCL1 SCK1 SDA1 SDO1	TX1 DT1 CK1	—	—	—

注：

1. 此信号为 PPS 可重映射输入信号。输入功能可从图示的默认位置移至任何 PORTx 引脚。
2. 此行中显示的所有输出信号均为 PPS 可重映射输出信号。
3. 此信号为双向信号。为确保正常工作，用户软件必须通过 PPS 输入和 PPS 输出寄存器将此信号映射到同一引脚。
4. 这些引脚可通过 RxyI2C 寄存器配置为 I^2C 或 SMBus 逻辑电平。SCL1/SDA1 信号可分配给这些引脚进行预期的操作。这些信号可通过 PPS 功能分配给其他引脚；但逻辑电平将是通过 INLVL 寄存器选择的标准 TTL/ST。

4. 寄存器和位命名约定

4.1. 寄存器名称

当器件中的同一外设存在多个实例时，外设控制寄存器将被描述为外设标识符、外设实例和控制标识符的组合。控制寄存器部分会使用“x”代替所有寄存器名称中的外设实例编号，从而将其显示为一个实例。这一命名约定也可用于该器件外设只有一个实例的情况，以便与同系列中其他包含多个实例的器件保持一致。

4.2. 位名称

位名称有两种形式：

- 短名称：位功能缩写
- 长名称：外设缩写 + 短名称

4.2.1. 短位名称

短位名称是位功能的缩写。例如，一些外设使用 EN 位来使能。寄存器中显示的位名称为短名称形式。

短位名称在 C 程序中访问位时非常有用。通过短名称访问位的一般格式为 RegisterNamebits.ShortName。例如，ADCON0 寄存器中的使能位 ON 可在 C 程序中通过指令 `ADCON0bits.ON = 1` 置 1。

短名称在汇编程序中没什么用，因为不同的外设可能在不同的位位置使用相同的名称。发生这种情况时，在生成包含文件期间，短位名称实例后会被加上一个下划线以及位所在的寄存器的名称，以避免命名争用。

4.2.2. 长位名称

长位名称是通过为短名称添加外设缩写前缀构成的。前缀对于外设是惟一的，因此每个长位名称都是惟一的。ADC 使能位的长位名称是 ADC 的前缀 AD 加上使能位短名称 ON，从而得到惟一的位名称 ADON。

长位名称在 C 程序和汇编程序中均非常实用。例如，在 C 程序中，ADCON0 使能位可通过 `ADON = 1` 指令置 1。在汇编程序中，此位可通过 `BSF ADCON0,ADON` 指令置 1。

4.2.3. 位域

位域是同一寄存器中的两个或多个相邻位。位域仅遵循短位命名约定。例如，ADCON2 寄存器的低三位包含 ADC 工作模式选择位。该位域的短名称为 MD，长名称为 ADMD。仅可在 C 程序中进行位域访问。下例演示了用于将 ADC 设为在累加模式下工作的 C 程序指令：

```
ADCON2bits.MD = 0b001;
```

位域中的各个位也可通过长位名称和短位名称访问。每个位都是在位域名称后附加该位在位域中的位置编号而构成。例如，最高有效模式位的短位名称为 MD2，长位名称为 ADMD2。下面两个示例演示了用于将 ADC 设为在累加模式下工作的汇编程序序列：

```
MOVLW  ~(1<<MD2 | 1<<MD1)
ANDWF  ADCON2,F
MOVLW  1<<MD0
IORWF  ADCON2,F
```

```
BCF    ADCON2,ADMD2
BCF    ADCON2,ADMD1
BSF    ADCON2,ADMD0
```

4.3. 寄存器和位命名例外情况

4.3.1. 状态、中断和镜像位

状态、中断允许、中断标志和镜像位包含在跨多个外设的寄存器中。在这类情况下，显示的位名称是惟一的，因此不存在前缀或短名称形式。

5. 寄存器图例

表 5-1. 寄存器图例

符号	定义
R	可读位
W	可写位
HS	硬件置 1 位
HC	硬件清零位
S	仅置 1 位
C	仅清零位
U	未实现位，读为 0
1	位值置 1
0	位值清零
x	位值未知
u	位值不变
q	位值视条件而定
m	位值预定义

6. 器件配置

器件配置由配置字、用户 ID、器件 ID 和器件配置信息（Device Configuration Information，DCI）区组成。

6.1. 配置字

提供 5 个配置字，可供用户选择器件振荡器、复位和存储器保护选项。这些字在 0x8007 - 0x800B 地址范围内实现。

注：配置字寄存器中的 DEBUG 位由器件开发工具（包括调试器和编程器）自动管理。对于正常的器件操作，此位需保持为 1。

6.2. 代码保护

代码保护用于保护器件不受未经授权的访问。任何代码保护设置都不会影响对程序存储器的内部访问。

整个程序存储空间都通过 $\overline{CP}$ 位来防止外部读写操作。当 $\overline{CP}$= 0 时，将禁止对程序存储器的外部读写操作，读取时将返回全 0。无论保护位的设置如何，CPU 都可以继续读取程序存储器。对程序存储器的自写操作则取决于写保护设置。

6.3. 写保护

写保护用于保护器件不受意外的自写访问。这样在允许修改程序存储器其他区域的同时可以保护应用程序，例如自举程序。

$\overline{WRTn}$ 配置位用于确定受保护的程序存储块：

- 应用程序块写保护： $\overline{WRTAPP}$ 配置位用于对应用程序块进行写保护。
- 存储区闪存（SAF）写保护： $\overline{WRTSAF}$ 配置位用于对 SAF 进行写保护。
- 配置寄存器写保护： $\overline{WRTC}$ 配置位用于对配置寄存器进行写保护。
- 引导块写保护： $\overline{WRTB}$ 配置位用于对引导块进行写保护。

使能时，相应的存储单元受写保护，禁止进行进一步的编程，除非对配置存储器区域执行批量擦除操作。在程序执行期间，仍然可以通过 NVM 固件对引导块、SAF 和/或应用程序块进行编程和读取。

6.4. 用户 ID

存储空间中有 4 个字（8000h-8003h）被指定为 ID 存储单元，供用户存储校验和或其他代码标识号。在正常执行过程中可以读写这些单元。有关访问这些存储单元的更多信息，请参见**“对 DCI、用户 ID、DEV/REV ID 和配置字进行 NVMREG 访问”**一节。有关对这些存储单元进行编程所需的电气参数信息，请参见**“电气规范”**一章中的**“存储器编程规范”**一节。有关校验和计算的更多信息，请参见**“系列编程规范”**。

6.5. 器件 ID 和版本 ID

14 位器件 ID 字位于 8006h，14 位版本 ID 位于 8005h。这些单元是只读的，不能擦除或修改。

开发工具（如器件编程器和调试器）可用于读取器件 ID、版本 ID 和配置字。有关访问这些存储单元的更多信息，请参见“**NVM——非易失性存储器控制**”一章。

6.6. 寄存器定义：配置设置

6.6.1. 配置字 1

名称： CONFIG1
偏移量： 0x8007

位	15	14	13	12	11	10	9	8
				VDDAR				$\overline{\text{CLKOUTEN}}$
访问				R/W				R/W
复位				1				1

位	7	6	5	4	3	2	1	0
			RSTOSC[1:0]				FEXTOSC[1:0]	
访问			R/W	R/W			R/W	R/W
复位			1	1			1	1

Bit 12 – VDDAR V_{DD} 模拟范围校准选择

值	说明
1	校准内部模拟系统，以便在 V_{DD} = 2.3V - 5.5V 范围内工作
0	校准内部模拟系统，以便在 V_{DD} = 1.8V - 3.6V 范围内工作

Bit 8 – $\overline{\text{CLKOUTEN}}$ 时钟输出使能

值	说明
1	禁止 CLKOUT 功能；CLKOUT 引脚为 I/O 功能
0	使能 CLKOUT 功能；CLKOUT 引脚为 F_{OSC}/4 时钟

Bit 5:4 – RSTOSC[1:0] COSC 的上电默认值位
选择用户软件使用的振荡器源。请参见 COSC 工作模式。

值	说明
11	EXTOSC，工作模式取决于 FEXTOSC 位
10	HFINTOSC，FRQ = 1 MHz
01	LFINTOSC
00	HFINTOSC，FRQ = 32 MHz

Bit 1:0 – FEXTOSC[1:0] 外部振荡器模式选择

值	说明
11	ECH（16 MHz 及更高值）
10	ECL（低于 16 MHz）
01	振荡器未使能
00	保留

6.6.2. 配置字 2

名称： CONFIG2
偏移量： 0x8008

位	15	14	13	12	11	10	9	8
			DEBUG	STVREN	PPS1WAY		BORV	
访问			R/W	R/W	R/W		R/W	
复位			1	1	1		1	

位	7	6	5	4	3	2	1	0
	BOREN[1:0]			WDTE[1:0]		PWRTS[1:0]		MCLRE
访问	R/W	R/W		R/W	R/W	R/W	R/W	R/W
复位	1	1		1	1	1	1	1

Bit 13 – DEBUG 调试器使能(1)

值	说明
1	禁止后台调试器
0	使能后台调试器

Bit 12 – STVREN 堆栈上溢/下溢复位使能

值	说明
1	堆栈上溢或下溢将导致复位
0	堆栈上溢或下溢不会导致复位

Bit 11 – PPS1WAY PPSLOCKED 一次置 1 使能

值	说明
1	PPSLOCKED 位只能在执行解锁序列之后置 1 一次；将 PPSLOCKED 置 1 后，可防止将来对 PPS 寄存器执行任何更改
0	PPSLOCKED 位可根据需要置 1 和清零（需要解锁序列）

Bit 9 – BORV 欠压复位（BOR）电压选择(2)

值	说明
1	欠压复位电压（V_{BOR}）设为 1.9V
0	欠压复位电压（V_{BOR}）设为 2.85V

Bit 7:6 – BOREN[1:0] 欠压复位（BOR）使能(3)

值	说明
11	使能欠压复位，忽略 SBOREN 位
10	运行时使能欠压复位，休眠时禁止；忽略 SBOREN 位
01	按照 SBOREN 使能欠压复位
00	禁止欠压复位

Bit 4:3 – WDTE[1:0] 看门狗定时器（WDT）使能

值	说明
11	WDT 使能（无论是否处于休眠状态）；忽略 WDTCON 的 SEN 位
10	WDT 在 Sleep = 0 时使能，在 Sleep = 1 时暂停；忽略 WDTCON 的 SEN 位
01	WDT 由 WDTCON 的 SEN 位使能/禁止
00	WDT 禁止，忽略 WDTCON 的 SEN 位

Bit 2:1 – PWRTS[1:0] 上电延时定时器（PWRT）选择

值	说明
11	禁止 PWRT
10	PWRT 设为 64 ms
01	PWRT 设为 16 ms
00	PWRT 设为 1 ms

Bit 0 – MCLRE 主复位（$\overline{\text{MCLR}}$）使能

值	条件	说明
x	如果 LVP = 1	$\overline{\text{MCLR}}$ 引脚为 $\overline{\text{MCLR}}$
1	如果 LVP = 0	$\overline{\text{MCLR}}$ 引脚为 $\overline{\text{MCLR}}$
0	如果 LVP = 0	$\overline{\text{MCLR}}$ 引脚功能为端口定义的功能

注：

1. $\overline{\text{DEBUG}}$ 位由器件开发工具（包括调试器和编程器）自动管理。对于正常的器件操作，此位需保持为 1。
2. 如果工作频率为 16 MHz 或更高值，建议选择更高电压。
3. 使能后，欠压复位电压（V_{BOR}）通过 BORV 位置 1。

6.6.3. 配置字 3

名称： CONFIG3
偏移量： 0x8009

注：该寄存器保留。

位	15	14	13	12	11	10	9	8
访问								
复位								

位	7	6	5	4	3	2	1	0
访问								
复位								

6.6.4. 配置字 4

名称： CONFIG4
偏移量： 0x800A

位	15	14	13	12	11	10	9	8
			LVP		WRTSAF		WRTC	WRTB
访问			R/W		R/W		R/W	R/W
复位			1		1		1	1

位	7	6	5	4	3	2	1	0
	WRTAPP			SAFEN	BBEN	BBSIZE[2:0]		
访问	R/W			R/W	R/W	R/W	R/W	R/W
复位	1			1	1	1	1	1

Bit 13 – LVP 低电压编程使能(1)

值	说明
1	使能低电压编程。$\overline{MCLR}/V_{PP}$ 引脚功能是 $\overline{MCLR}$。MCLRE 位被忽略。
0	必须使 $\overline{MCLR}/V_{PP}$ 为高电压（HV）才能进行编程

Bit 11 – WRTSAF 存储区闪存（SAF）写保护(2,3)

值	说明
1	SAF 不受写保护
0	SAF 受写保护

Bit 9 – WRTC 配置寄存器写保护(2)

值	说明
1	配置寄存器不受写保护
0	配置寄存器受写保护

Bit 8 – WRTB 引导块写保护(2,4)

值	说明
1	引导块不受写保护
0	引导块受写保护

Bit 7 – WRTAPP 应用程序块写保护(2)

值	说明
1	应用程序块不受写保护
0	应用程序块受写保护

Bit 4 – SAFEN 存储区闪存（SAF）使能(2)

值	说明
1	禁止 SAF
0	使能 SAF

Bit 3 – BBEN 引导块使能(2)

值	说明
1	禁止引导块

值	说明
0	使能引导块

Bit 2:0 – BBSIZE[2:0] 引导块大小选择(5,6)

表 6-1. 引导块大小

$\overline{\text{BBEN}}$	BBSIZE	引导块结束地址	引导块大小（字）	
			CN5223	CN5225
1	xxx	-	-	
0	111	01FFh	512	
0	110	03FFh	1024	
0	101	07FFh	– (6)	2048
0	100	0FFFh	– (6)	4096
0	011	1FFFh	– (6)	
0	010	3FFFh	– (6)	
0	001	3FFFh	– (6)	
0	000	3FFFh	– (6)	

注：

1. 通过 LVP 编程接口工作时，不能对 LVP 位进行写操作（写为 0）。该规则的目的是防止用户在通过 LVP 模式编程时退出 LVP 模式，或意外地从配置状态中删除 LVP 模式。
2. 使能保护后，只能通过批量擦除操作复位。
3. 仅在 $\overline{\text{SAFEN}}$ = 0 时适用。
4. 仅在 $\overline{\text{BBEN}}$ = 0 时适用。
5. BBSIZE[2:0]位只能在 $\overline{\text{BBEN}}$ = 1 时更改。当 $\overline{\text{BBEN}}$ = 0 后，BBSIZE[2:0]只能通过批量擦除操作进行更改。
6. 最大引导块大小是用户程序存储器大小的一半。如果选择的引导块大小超过器件程序存储器大小的一半，则会默认设为最大引导块大小，即 PFM 的一半。例如，对于 CN5223（最大 PFM = 2048 字），选择从 110 到 000 的 BBSIZE 设置时均会设为最大引导块大小，即 1024 字。

6.6.5. 配置字 5(1)

名称： CONFIG5
偏移量： 0x800B

位	15	14	13	12	11	10	9	8
访问								
复位								

位	7	6	5	4	3	2	1	0
								$\overline{CP}$
访问								R/W
复位								1

Bit 0 – $\overline{CP}$ 闪存程序存储器（PFM）代码保护(2)

值	说明
1	禁止 PFM 代码保护
0	使能 PFM 代码保护

注：

1. 由于器件代码保护会立即生效，因此该配置字需最后写入。
2. 代码保护使能后，只能通过批量擦除操作取消。

6.7. 寄存器定义：器件 ID 和版本 ID

6.7.1. 器件 ID

名称： DEVICEID
偏移量： 0x8006

器件 ID 寄存器

位	15	14	13	12	11	10	9	8
			保留	保留	DEV[11:8]			
访问			R	R	R	R	R	R
复位			1	1	q	q	q	q

位	7	6	5	4	3	2	1	0
	DEV[7:0]							
访问	R	R	R	R	R	R	R	R
复位	q	q	q	q	q	q	q	q

Bit 13 – 保留 保留——读为 1

Bit 12 – 保留 保留——读为 1

Bit 11:0 – DEV[11:0] 器件 ID

器件	器件 ID
CN5223	30E4h
CN5225	30E9h

6.7.2. 版本 ID

名称： REVISIONID
偏移量： 0x8005

版本 ID 寄存器

位	15	14	13	12	11	10	9	8
			保留	保留	MJRREV[5:2]			
访问			R	R	R	R	R	R
复位			1	0	q	q	q	q

位	7	6	5	4	3	2	1	0
	MJRREV[1:0]		MNRREV[5:0]					
访问	R	R	R	R	R	R	R	R
复位	q	q	q	q	q	q	q	q

Bit 13 – 保留 保留——读为 1

Bit 12 – 保留 保留——读为 0

Bit 11:6 – MJRREV[5:0] 主要版本 ID
这些位用于标识主要版本（A0、B0 和 C0 等）。

Bit 5:0 – MNRREV[5:0] 次要版本 ID
这些位用于标识次要版本。

7. 存储器构成

CN5225 单片机有 2 种类型的存储器：

- 程序存储器
  - 闪存程序存储器
  - 配置字
  - 器件 ID
  - 版本 ID
  - 用户 ID
  - 器件配置信息（DCI）
- 数据存储器
  - 内核寄存器
  - 特殊功能寄存器（Special Function Register，SFR）
  - 通用 RAM（General Purpose RAM，GPR）
  - 公共 RAM

在哈佛架构器件中，数据存储器和程序存储器分别使用单独的总线，因此支持对两个存储空间的并发访问。

有关闪存程序存储器操作的更多详细信息，请参见**“NVM——非易失性存储器控制”**一章。

7.1. 程序存储器构成

增强型中档内核具有一个 15 位程序计数器，能够寻址 32K x 14 的程序存储空间。下表给出了已实现的存储容量。访问超出地址边界的单元将导致操作返回到已实现的存储空间内。

复位向量地址为 0000h，中断向量地址为 0004h。更多详细信息，请参见**“INT——中断”**一章。

表 7-1. 器件存储器容量和地址

器件	程序存储器容量（字）	程序存储器的最后一个地址
CN5223	2,048	07FFh
CN5225	8,192	1FFFh

图 7-1. 程序存储器和堆栈（CN5223）

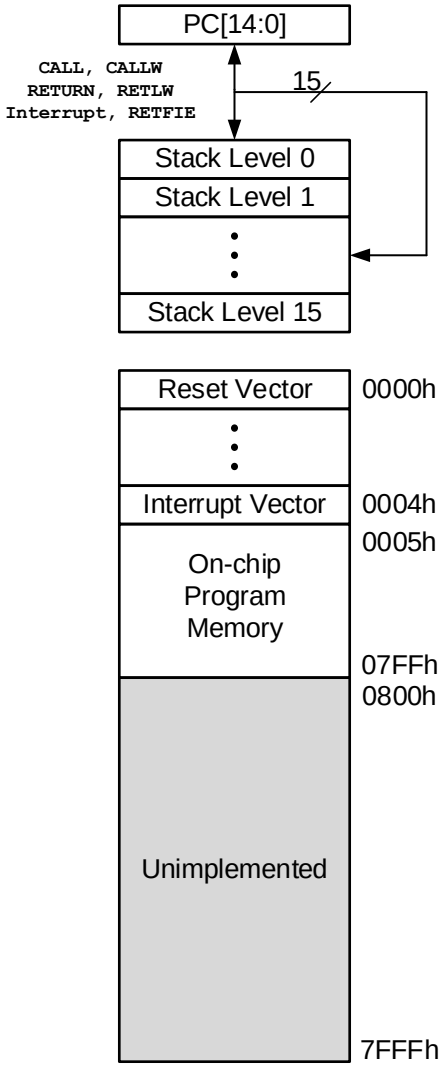
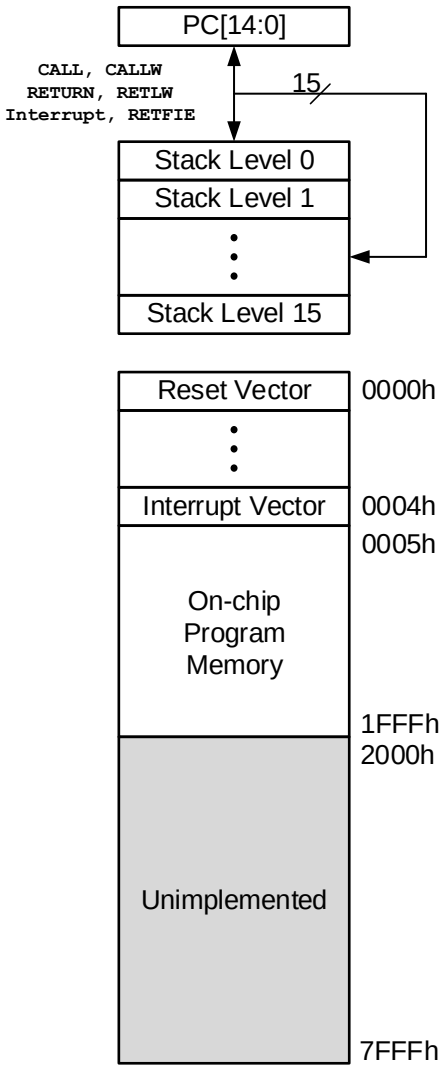


图 7-2. 程序存储器和堆栈（CN5225）



7.1.1. 将程序存储器当作数据存储器读取

有三种方法可访问程序存储器中的常量。第一种方法是使用 RETLW 指令表，第二种方法是设置 FSR 以指向程序存储器，第三种方法是使用 NVMREG 接口访问程序存储器。

7.1.1.1. RETLW 指令

RETLW 指令用于访问常量表。以下示例给出了创建这种表的推荐方法。

例 7-1. 使用 RETLW 指令访问常量表

```
constants
    BRW                     ;Add Index in W to
                            ;program counter to
                            ;select data
    RETLW DATA0             ;Index0 data
    RETLW DATA1             ;Index1 data
```

```
    RETLW DATA2
    RETLW DATA3
my_function
    ;LOTS OF CODE....
    MOVLW       DATA_INDEX
    call constants
    ;THE CONSTANT IS IN W
```

BRW 指令可轻松实现这种表。

7.1.1.2. 使用 FSR 间接读取

通过将 FSRxH 寄存器的 bit 7 置 1，并读取匹配的 INDFx 寄存器，可以将程序存储器作为数据进行访问。MOVIW 指令会将所寻址字的低 8 位放入 W 寄存器。无法通过 INDFx 寄存器对程序存储器进行写操作。通过 FSR 读取程序存储器的指令需要一个额外的指令周期才能完成。以下示例给出了通过 FSR 读取程序存储器的过程。

如果某个标号指向程序存储器中的存储单元，HIGH 伪指令会将 bit 7 置 1。这适用于如下所示的汇编代码。

例 7-2. 使用 FSR 寄存器读取程序存储器

```
constants
    RETLW DATA0             ;Index0 data
    RETLW DATA1             ;Index1 data
    RETLW DATA2
    RETLW DATA3
my_function
    ;LOTS OF CODE....
    MOVLW    LOW constants
    MOVWF    FSR1L
    MOVLW    HIGH constants
    MOVWF    FSR1H
    MOVIW    2[FSR1]        ;DATA2 IS IN W
```

7.1.2. 存储器访问分区（MAP）

用户闪存可分为：

- 应用程序块
- 引导块
- 存储区闪存（SAF）块

要分配存储器，用户可将 $\overline{\text{BBEN}}$ 位置 1，通过 BBSIZE 位定义分区大小，然后通过 $\overline{\text{SAFEN}}$ 位使能 SAF。

7.1.2.1. 应用程序块

配置位的默认设置（$\overline{\text{BBEN}}$= 1 和 $\overline{\text{SAFEN}}$= 1）会将用户闪存区域内的所有存储器分配到应用程序块中。

7.1.2.2. 引导块

如果 $\overline{\text{BBEN}}$= 1，则使能引导块，并基于 BBSIZE 位的值将特定地址范围分配给引导块。

7.1.2.3. 存储区闪存（SAF）

通过清零 $\overline{\text{SAFEN}}$ 位使能存储区闪存（SAF）。如果 SAF 已使能，则 SAF 块位于存储器的末尾，大小为 128 个字。如果 SAF 已使能，则 SAF 区域不可用于程序执行。

重要：使能时，SAF 可用于存储变量或其他信息，通常用于没有 EEPROM 的器件；访问 SAF 的方式与其他闪存区域相同。

7.1.2.4. 存储器写保护

所有存储器块都具有相应的写保护位（WRTAPP、WRTB、WRTC、 和 WRTSAF）。如果从 NVMCON 寄存器写入受写保护的存储单元，则存储器不发生更改并且 NVMCON1 寄存器的 WRERR 位置 1，如 **“NVM——非易失性存储器控制”** 一章中的 **“WRERR 位”** 一节中所述。

7.1.2.5. 存储器违例

执行从有效执行区域之外取出的指令时会触发存储器执行违例复位，从而会将 MEMV 位清零。有关可用的有效程序执行区域以及 PCON1 寄存器中 MEMV 位条件的定义，请参见 **“复位”** 一章中的 **“存储器执行违例”** 一节。

表 7-2. 存储器访问分区

REG	地址	分区			
		BBEN= 1 SAFEN= 1	BBEN= 1 SAFEN= 0	BBEN= 0 SAFEN= 1	BBEN= 0 SAFEN= 0
PFM	00 0000h ...块存储器的最后一个地址	应用程序块(4)	应用程序块(4)	引导块(4)	引导块(4)
	引导块存储器的最后一个地址 + 1(1) ...程序存储器的最后一个地址 - 80h			应用程序块(4)	应用程序块(4)
	程序存储器的最后一个地址 - 7Fh(2)...程序存储器的最后一个地址		SAF(4)		SAF(4)
配置	配置存储器地址(3)	配置			

注：

1. 引导块存储器的最后一个地址基于 BBSIZE 配置位。
2. 最后一个程序存储器地址显示在器件存储器容量和地址表中。
3. 配置存储器地址是 **“NVM——非易失性存储器控制”** 一章中的 **“对 DCI、用户 ID、DEV/REV ID 和配置字进行 NVMREG 访问”** 一节中给出的配置字的地址单元。
4. 每个存储块都有一个对应的写保护熔丝，由 WRTAPP、WRTB、WRTC、 和 WRTSAF 配置位定义。

7.1.3. 器件配置信息（DCI）

器件配置信息（DCI）是存储器中的专用区域，其中存储的器件相关信息对于编程和自举程序应用十分有用。该区域中存储的数据是只读的，不能修改/擦除。有关完整的 DCI 表地址和说明，请参见下表。

表 7-3. 器件配置信息

地址	名称	说明	值		单位
			CN5223	CN5225	
8200h	ERSIZ	擦除行大小	32		字
8201h	WLSIZ	每行写锁存器数量	32		字
8202h	URSIZ	用户可擦除行数量	64	256	行
8203h	EESIZ	数据 EEPROM 存储器大小	0		字节
8204h	PCNT	引脚数	14		引脚

7.1.3.1. DCI 访问

DCI 数据是只读的，不能擦除或修改。有关访问这些存储单元的更多信息，请参见 **“NVM——非易失性存储器控制”** 一章中的 **“对 DCI、用户 ID、DEV/REV ID 和配置字进行 NVMREG 访问”** 一节。

开发工具（例如器件编程器和调试器）可以用来读取 DCI 区域，与器件 ID 和版本 ID 类似。

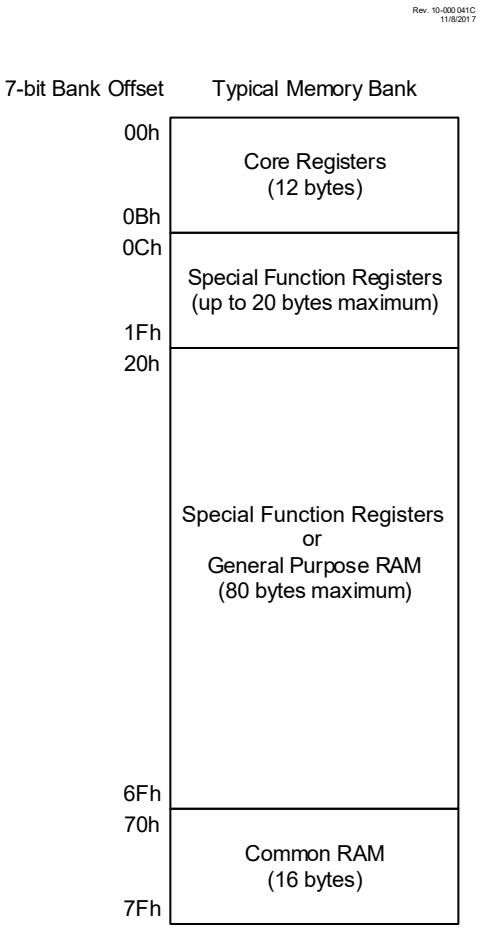
7.2. 数据存储器构成

数据存储器划分为最多 64 个存储区，每个存储区有 128 字节。每个存储区都包含：

- 12 个内核寄存器

- 最多 20 个特殊功能寄存器（SFR）
- 最多 80 字节的通用 RAM（GPR）
- 16 字节的公共 RAM

图 7-3. 存储器分区



7.2.1. 存储区选择

可通过将存储区号写入存储区选择寄存器（BSR）来选择有效存储区。所有的数据存储器既可通过使用文件寄存器的指令直接访问，也可通过 2 个文件选择寄存器（FSR）间接访问。数据存储器使用 13 位地址。高 6 位的地址定义存储区地址，低 7 位选择该存储区的寄存器/RAM。

7.2.2. 内核寄存器

内核寄存器包含直接影响基本操作的寄存器。内核寄存器占据数据存储器每个存储区的前 12 个地址。下面的内核寄存器表列出了这些寄存器。

表 7-4. 内核寄存器

BANKx 中的地址	内核寄存器
x00h 或 x80h	INDF0
x01h 或 x81h	INDF1
x02h 或 x82h	PCL

表 7-4. 内核寄存器 （续）

BANKx 中的地址	内核寄存器
x03h 或 x83h	STATUS
x04h 或 x84h	FSR0L
x05h 或 x85h	FSR0H
x06h 或 x86h	FSR1L
x07h 或 x87h	FSR1H
x08h 或 x88h	BSR
x09h 或 x89h	WREG
x0Ah 或 x8Ah	PCLATH
x0Bh 或 x8Bh	INTCON

7.2.3. 特殊功能寄存器

特殊功能寄存器（SFR）是应用用来控制器件中外设功能所需操作的寄存器。SFR 占据内核寄存器之后的数据存储区 0-59 的前 20 个字节和数据存储区 60-63 的前 100 个字节。

与外设操作有关的 SFR 将在本数据手册的相应外设章节中讲述。

7.2.4. 通用 RAM（GPR）

每个数据存储区中有最大 80 字节的通用 RAM（GPR）。可以通过 FSR 以非存储区方式访问 GPR。这可以简化对大存储器结构的访问。

有关线性存储器访问的详细信息，请参见**“存储器构成”**一章中的**“线性数据存储器”**一节。

7.2.5. 公共 RAM

提供 16 字节的公共 RAM，可从所有存储区访问。

7.2.6. 器件存储器映射

下图列出了本数据手册中所述器件的存储器映射。

图 7-4. 存储器映射——Bank 0-7

BANK 0		BANK 1		BANK 2		BANK 3		BANK 4		BANK 5		BANK 6		BANK 7	
000h	Core Registers	080h	Core Registers	100h	Core Registers	180h	Core Registers	200h	Core Registers	280h	Core Registers	300h	Core Registers	380h	Core Registers
00Bh		08Bh		10Bh		18Bh		20Bh		28Bh		30Bh		38Bh	
00Ch	PORTA	08Ch	—	10Ch	—	18Ch	SSP1BUF	20Ch	TMR1L	28Ch	T2TMR	30Ch	CCPR1L	38Ch	—
00Dh	—	08Dh	—	10Dh	—	18Dh	SSP1ADD	20Dh	TMR1H	28Dh	T2PR	30Dh	CCPR1H	38Dh	—
00Eh	PORTC	08Eh	—	10Eh	RC0I2C	18Eh	SSP1MASK	20Eh	T1CON	28Eh	T2CON	30Eh	CCP1CON	38Eh	—
00Fh	—	08Fh	—	10Fh	RC1I2C	18Fh	SSP1STAT	20Fh	T1GCON	28Fh	T2HLT	30Fh	CCP1CAP	38Fh	—
010h	—	090h	—	110h	—	190h	SSP1CON1	210h	T1GATE	290h	T2CLKCON	310h	CCPR2L	390h	—
011h	—	091h	—	111h	—	191h	SSP1CON2	211h	T1CLK	291h	T2RST	311h	CCPR2H	391h	—
012h	TRISA	092h	—	112h	—	192h	SSP1CON3	212h	—	292h	—	312h	CCP2CON	392h	—
013h	—	093h	—	113h	—	193h	—	213h	—	293h	—	313h	CCP2CAP	393h	—
014h	TRISC	094h	—	114h	—	194h	—	214h	—	294h	—	314h	PWM3DCL	394h	—
015h	—	095h	—	115h	—	195h	—	215h	—	295h	—	315h	PWM3DCH	395h	—
016h	—	096h	—	116h	—	196h	—	216h	—	296h	—	316h	PWM3CON	396h	—
017h	—	097h	—	117h	—	197h	—	217h	—	297h	—	317h	—	397h	—
018h	LATA	098h	—	118h	—	198h	—	218h	—	298h	—	318h	PWM4DCL	398h	—
019h	—	099h	—	119h	RC1REG	199h	—	219h	—	299h	—	319h	PWM4DCH	399h	—
01Ah	LATC	09Ah	CPCON	11Ah	TX1REG	19Ah	—	21Ah	—	29Ah	—	31Ah	PWM4CON	39Ah	—
01Bh	—	09Bh	ADRESL	11Bh	SP1BRGL	19Bh	—	21Bh	—	29Bh	—	31Bh	—	39Bh	—
01Ch	—	09Ch	ADRESH	11Ch	SP1BRGH	19Ch	—	21Ch	—	29Ch	—	31Ch	—	39Ch	—
01Dh	—	09Dh	ADCON0	11Dh	RC1STA	19Dh	—	21Dh	—	29Dh	—	31Dh	—	39Dh	—
01Eh	—	09Eh	ADCON1	11Eh	TX1STA	19Eh	—	21Eh	—	29Eh	—	31Eh	—	39Eh	—
01Fh	—	09Fh	ADACT	11Fh	BAUD1CON	19Fh	—	21Fh	—	29Fh	—	31Fh	—	39Fh	—
020h	General Purpose Register 80 Bytes	0A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h	General Purpose Register 80 Bytes(1)	220h	General Purpose Register 80 Bytes(1)	2AFh	General Purpose Register 80 Bytes(1)	320h	General Purpose Register 80 Bytes(1)	3A0h	General Purpose Register 80 Bytes(1)
06Fh		0EFh		16Fh		1EFh		26Fh		2EFh		36Fh		3EFh	
070h	Common RAM (Accesses 70h-7Fh)	0F0h	Common RAM (Accesses 70h-7Fh)	170h	Common RAM (Accesses 70h-7Fh)	1F0h	Common RAM (Accesses 70h-7Fh)	270h	Common RAM (Accesses 70h-7Fh)	2F0h	Common RAM (Accesses 70h-7Fh)	370h	Common RAM (Accesses 70h-7Fh)	3F0h	Common RAM (Accesses 70h-7Fh)
07Fh		0FFh		17Fh		1FFh		27Fh		2FFh		73Fh		3FFh	

Note: **1. Available on CN5225 devices only**

Legend:
Unimplemented data memory locations, read as '0'

图 7-5. 存储器映射——Bank 8-15

BANK 8		BANK 9		BANK 10		BANK 11		BANK 12		BANK 13		BANK 14		BANK 15	
400h	Core Registers	480h	Core Registers	500h	Core Registers	580h	Core Registers	600h	Core Registers	680h	Core Registers	700h	Core Registers	780h	Core Registers
40Bh		48Bh		50Bh		58Bh		60Bh		68Bh		70Bh		78Bh	
40Ch	—	48Ch	—	50Ch	—	58Ch	—	60Ch	—	68Ch	—	70Ch	PIR0	78Ch	—
40Dh	—	48Dh	—	50Dh	—	58Dh	—	60Dh	—	68Dh	—	70Dh	PIR1	78Dh	—
40Eh	—	48Eh	—	50Eh	—	58Eh	—	60Eh	—	68Eh	—	70Eh	PIR2	78Eh	—
40Fh	—	48Fh	—	50Fh	—	58Fh	—	60Fh	—	68Fh	—	70Fh	—	78Fh	—
410h	—	490h	—	510h	—	590h	—	610h	—	690h	—	710h	—	790h	—
411h	—	491h	—	511h	—	591h	—	611h	—	691h	—	711h	—	791h	—
412h	—	492h	—	512h	—	592h	—	612h	—	692h	—	712h	—	792h	—
413h	—	493h	—	513h	—	593h	—	613h	—	693h	—	713h	—	793h	—
414h	—	494h	—	514h	—	594h	—	614h	—	694h	—	714h	—	794h	—
415h	—	495h	—	515h	—	595h	—	615h	—	695h	—	715h	—	795h	—
416h	—	496h	—	516h	—	596h	—	616h	—	696h	—	716h	PIE0	796h	—
417h	—	497h	—	517h	—	597h	—	617h	—	697h	—	717h	PIE1	797h	—
418h	—	498h	—	518h	—	598h	—	618h	—	698h	—	718h	PIE2	798h	—
419h	—	499h	—	519h	—	599h	—	619h	—	699h	—	719h	—	799h	—
41Ah	—	49Ah	—	51Ah	—	59Ah	—	61Ah	—	69Ah	—	71Ah	—	79Ah	—
41Bh	—	49Bh	—	51Bh	—	59Bh	—	61Bh	—	69Bh	—	71Bh	—	79Bh	—
41Ch	—	49Ch	—	51Ch	—	59Ch	TMR0L	61Ch	—	69Ch	—	71Ch	—	79Ch	—
41Dh	—	49Dh	—	51Dh	—	59Dh	TMR0H	61Dh	—	69Dh	—	71Dh	—	79Dh	—
41Eh	—	49Eh	—	51Eh	—	59Eh	T0CON0	61Eh	—	69Eh	—	71Eh	—	79Eh	—
41Fh	—	49Fh	—	51Fh	—	59Fh	T0CON1	61Fh	—	69Fh	—	71Fh	—	79Fh	—
420h	General Purpose Register 80 bytes(1)	4A0h	General Purpose Register 80 bytes(1)	520h	General Purpose Register 80 bytes(1)	5A0h	General Purpose Register 80 bytes(1)	620h	General Purpose Register 48 bytes(1)	6A0h	Unimplemented Read as '0'	720h	Unimplemented Read as '0'	7A0h	Unimplemented Read as '0'
46Fh		4EFh		56Fh		5EFh		64Fh 650h	Unimplemented Read as '0'	6EFh		76Fh		7EFh	
470h	Common RAM (Accesses 70h-7Fh)	4F0h	Common RAM (Accesses 70h-7Fh)	570h	Common RAM (Accesses 70h-7Fh)	5F0h	Common RAM (Accesses 70h-7Fh)	670h	Common RAM (Accesses 70h-7Fh)	6F0h	Common RAM (Accesses 70h-7Fh)	770h	Common RAM (Accesses 70h-7Fh)	7F0h	Common RAM (Accesses 70h-7Fh)
47Fh		4FFh		57Fh		5FFh		67Fh		6FFh		77Fh		7FFh	

Note: **1. Available on CN5225 devices only**

Legend:
Unimplemented data memory locations, read as '0'

图 7-6. 存储器映射——Bank 16-23

BANK 16		BANK 17		BANK 18		BANK 19		BANK 20		BANK 21		BANK 22		BANK 23	
800h	Core Registers	880h	Core Registers	900h	Core Registers	980h	Core Registers	A00h	Core Registers	A80h	Core Registers	B00h	Core Registers	B80h	Core Registers
80Bh		88Bh		90Bh		98Bh		A0Bh		A8Bh		B0Bh		B8Bh	
80Ch	WDTCON	88Ch	—	90Ch	—	98Ch	—	A0Ch	—	A8Ch	—	B0Ch	—	B8Ch	Unimplemented Read as '0'
80Dh	—	88Dh	—	90Dh	—	98Dh	—	A0Dh	—	A8Dh	—	B0Dh	—		
80Eh	—	88Eh	OSCCON	90Eh	—	98Eh	—	A0Eh	—	A8Eh	—	B0Eh	—		
80Fh	—	88Fh	—	90Fh	—	98Fh	—	A0Fh	—	A8Fh	—	B0Fh	—		
810h	—	890h	OSCSTAT	910h	—	990h	—	A10h	—	A90h	—	B10h	—		
811h	BORCON	891h	OSCEN	911h	—	991h	—	A11h	—	A91h	—	B11h	—		
812h	—	892h	OSCTUNE	912h	—	992h	—	A12h	—	A92h	—	B12h	—		
813h	PCON0	893h	OSCFRQ	913h	—	993h	—	A13h	—	A93h	—	B13h	—		
814h	PCON1	894h	—	914h	—	994h	—	A14h	—	A94h	—	B14h	—		
815h	—	895h	—	915h	—	995h	—	A15h	—	A95h	—	B15h	—		
816h	—	896h	—	916h	—	996h	—	A16h	—	A96h	—	B16h	—		
817h	—	897h	—	917h	—	997h	—	A17h	—	A97h	—	B17h	—		
818h	—	898h	—	918h	—	998h	—	A18h	—	A98h	—	B18h	—		
819h	—	899h	—	919h	—	999h	—	A19h	—	A99h	—	B19h	—		
81Ah	NVMADRL	89Ah	—	91Ah	—	99Ah	—	A1Ah	—	A9Ah	—	B1Ah	—		
81Bh	NVMADRH	89Bh	—	91Bh	—	99Bh	—	A1Bh	—	A9Bh	—	B1Bh	—		
81Ch	NVMDATL	89Ch	—	91Ch	—	99Ch	—	A1Ch	—	A9Ch	—	B1Ch	—		
81Dh	NVMDATH	89Dh	—	91Dh	—	99Dh	—	A1Dh	—	A9Dh	—	B1Dh	—		
81Eh	NVMCON1	89Eh	—	91Eh	—	99Eh	—	A1Eh	—	A9Eh	—	B1Eh	—		
81Fh	NVMCON2	89Fh	—	91Fh	—	99Fh	—	A1Fh	—	A9Fh	—	B1Fh	—		
820h	Unimplemented Read as '0'	8A0h	Unimplemented Read as '0'	920h	Unimplemented Read as '0'	9A0h	Unimplemented Read as '0'	A20h	Unimplemented Read as '0'	AA0h	Unimplemented Read as '0'	B20h	Unimplemented Read as '0'		
86Fh		8EFh		96Fh		9EFh		A6Fh		AEFh		B6Fh		BEFh	
870h	Common RAM (Accesses 70h-7Fh)	8F0h	Common RAM (Accesses 70h-7Fh)	970h	Common RAM (Accesses 70h-7Fh)	9F0h	Common RAM (Accesses 70h-7Fh)	A70h	Common RAM (Accesses 70h-7Fh)	AF0h	Common RAM (Accesses 70h-7Fh)	B70h	Common RAM (Accesses 70h-7Fh)	BF0h	Common RAM (Accesses 70h-7Fh)
87Fh		8FFh		97Fh		9FFh		A7Fh		AFFh		BFFh		BFFh	

Legend:
Unimplemented data memory locations, read as '0'

图 7-7. 存储器映射——Bank 24-31

BANK 24		BANK 25		BANK 26		BANK 27		BANK 28		BANK 29		BANK 30		BANK 31	
C00h	Core Registers	C80h	Core Registers	D00h	Core Registers	D80h	Core Registers	E00h	Core Registers	E80h	Core Registers	F00h	Core Registers	F80h	Core Registers
C0Bh		C8Bh		D0Bh		D8Bh		E0Bh		E8Bh		F0Bh		F8Bh	
C0Ch	—	C8Ch	—	D0Ch	—	D8Ch	—	E0Ch	—	E8Ch	—	F0Ch	—	F8Ch	—
C0Dh	—	C8Dh	—	D0Dh	—	D8Dh	—	E0Dh	—	E8Dh	—	F0Dh	—	F8Dh	—
C0Eh	—	C8Eh	—	D0Eh	—	D8Eh	—	E0Eh	—	E8Eh	—	F0Eh	—	F8Eh	—
C0Fh	—	C8Fh	—	D0Fh	—	D8Fh	—	E0Fh	—	E8Fh	—	F0Fh	—	F8Fh	—
C10h	—	C90h	—	D10h	—	D90h	—	E10h	—	E90h	—	F10h	—	F90h	—
C11h	—	C91h	—	D11h	—	D91h	—	E11h	—	E91h	—	F11h	—	F91h	—
C12h	—	C92h	—	D12h	—	D92h	—	E12h	—	E92h	—	F12h	—	F92h	—
C13h	—	C93h	—	D13h	—	D93h	—	E13h	—	E93h	—	F13h	—	F93h	—
C14h	—	C94h	—	D14h	—	D94h	—	E14h	—	E94h	—	F14h	—	F94h	—
C15h	—	C95h	—	D15h	—	D95h	—	E15h	—	E95h	—	F15h	—	F95h	—
C16h	—	C96h	—	D16h	—	D96h	—	E16h	—	E96h	—	F16h	—	F96h	—
C17h	—	C97h	—	D17h	—	D97h	—	E17h	—	E97h	—	F17h	—	F97h	—
C18h	—	C98h	—	D18h	—	D98h	—	E18h	—	E98h	—	F18h	—	F98h	—
C19h	—	C99h	—	D19h	—	D99h	—	E19h	—	E99h	—	F19h	—	F99h	—
C1Ah	—	C9Ah	—	D1Ah	—	D9Ah	—	E1Ah	—	E9Ah	—	F1Ah	—	F9Ah	—
C1Bh	—	C9Bh	—	D1Bh	—	D9Bh	—	E1Bh	—	E9Bh	—	F1Bh	—	F9Bh	—
C1Ch	—	C9Ch	—	D1Ch	—	D9Ch	—	E1Ch	—	E9Ch	—	F1Ch	—	F9Ch	—
C1Dh	—	C9Dh	—	D1Dh	—	D9Dh	—	E1Dh	—	E9Dh	—	F1Dh	—	F9Dh	—
C1Eh	—	C9Eh	—	D1Eh	—	D9Eh	—	E1Eh	—	E9Eh	—	F1Eh	—	F9Eh	—
C1Fh	—	C9Fh	—	D1Fh	—	D9Fh	—	E1Fh	—	E9Fh	—	F1Fh	—	F9Fh	—
C20h	Unimplemented Read as '0'	CA0h	Unimplemented Read as '0'	D20h	Unimplemented Read as '0'	DA0h	Unimplemented Read as '0'	E20h	Unimplemented Read as '0'	EA0h	Unimplemented Read as '0'	F20h	Unimplemented Read as '0'	FA0h	Unimplemented Read as '0'
C6Fh		CEFh		D6Fh		DEFh		E6Fh		EEFh		F6Fh		FEFh	
C70h	Common RAM (Accesses 70h-7Fh)	CF0h	Common RAM (Accesses 70h-7Fh)	D70h	Common RAM (Accesses 70h-7Fh)	DF0h	Common RAM (Accesses 70h-7Fh)	E70h	Common RAM (Accesses 70h-7Fh)	EF0h	Common RAM (Accesses 70h-7Fh)	F70h	Common RAM (Accesses 70h-7Fh)	FF0h	Common RAM (Accesses 70h-7Fh)
C7Fh		CFFh		D7Fh		DFFh		E7Fh		EFFh		F7Fh		FFFh	

Legend:
Unimplemented data memory locations, read as '0'

图 7-8. 存储器映射——Bank 32-39

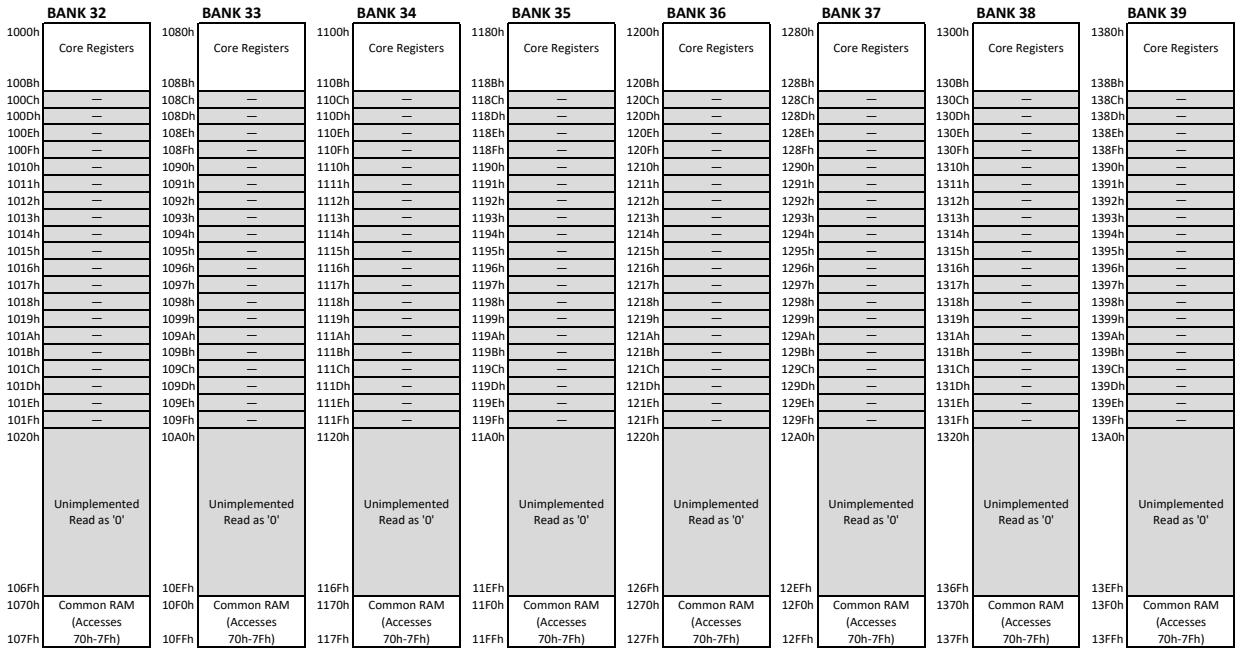


图 7-9. 存储器映射——Bank 40-47

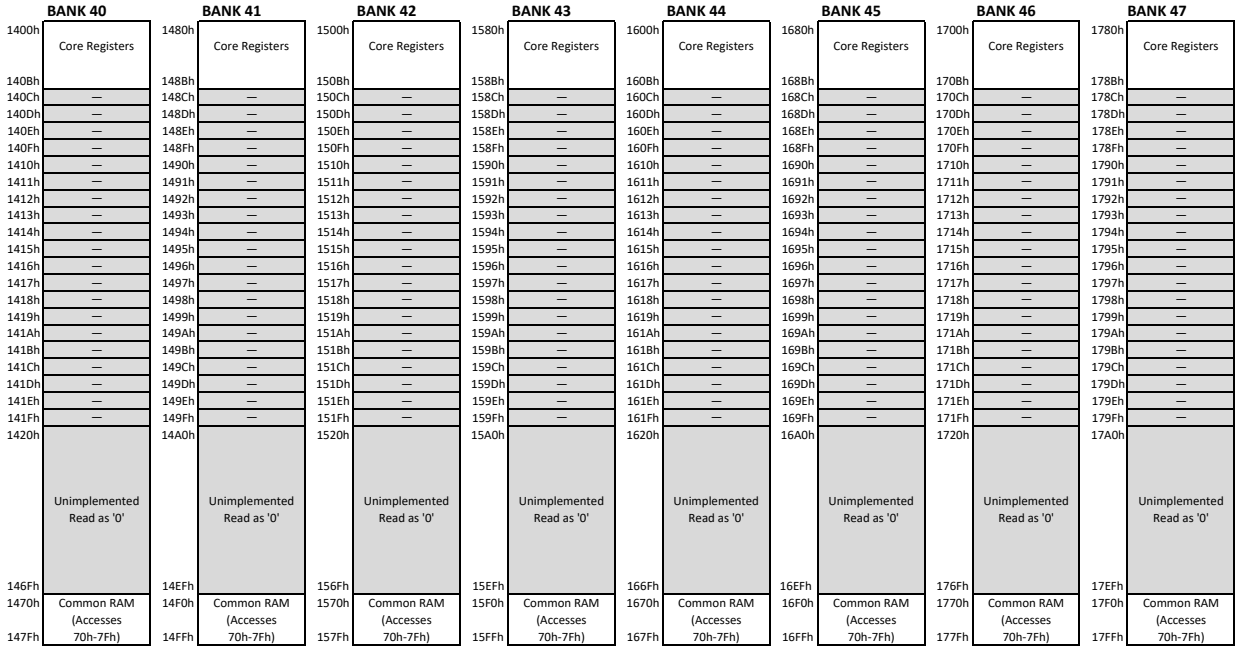


图 7-10. 存储器映射——Bank 48-55

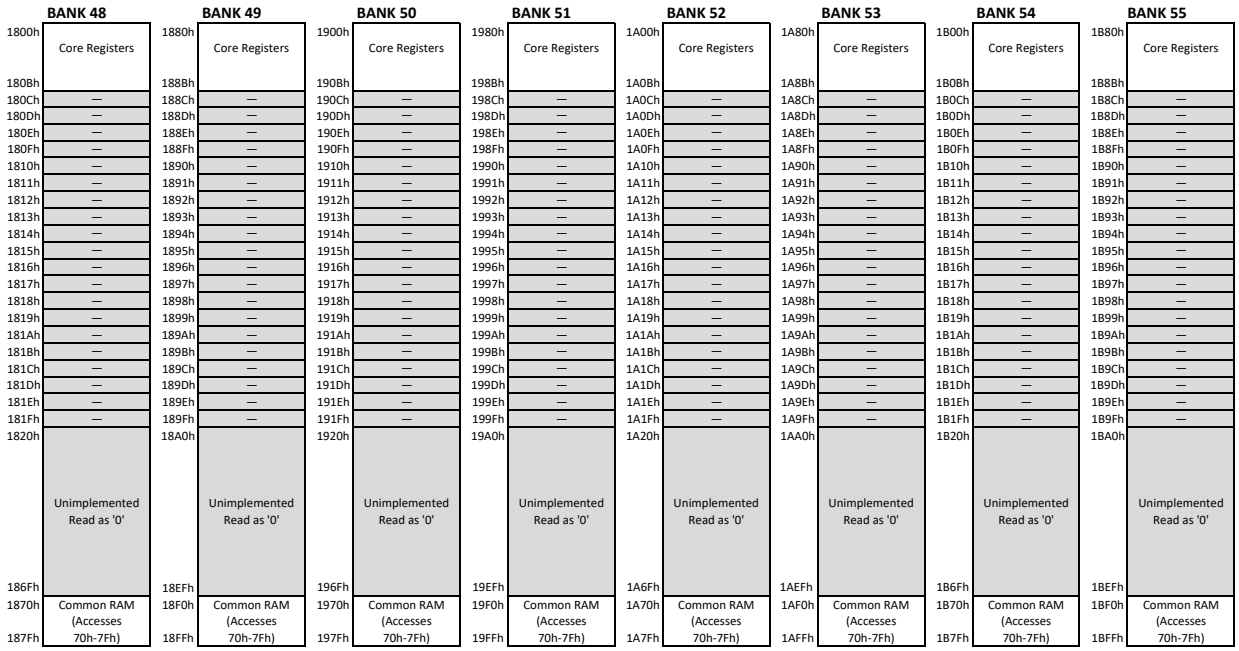


图 7-11. 存储器映射——Bank 56-63

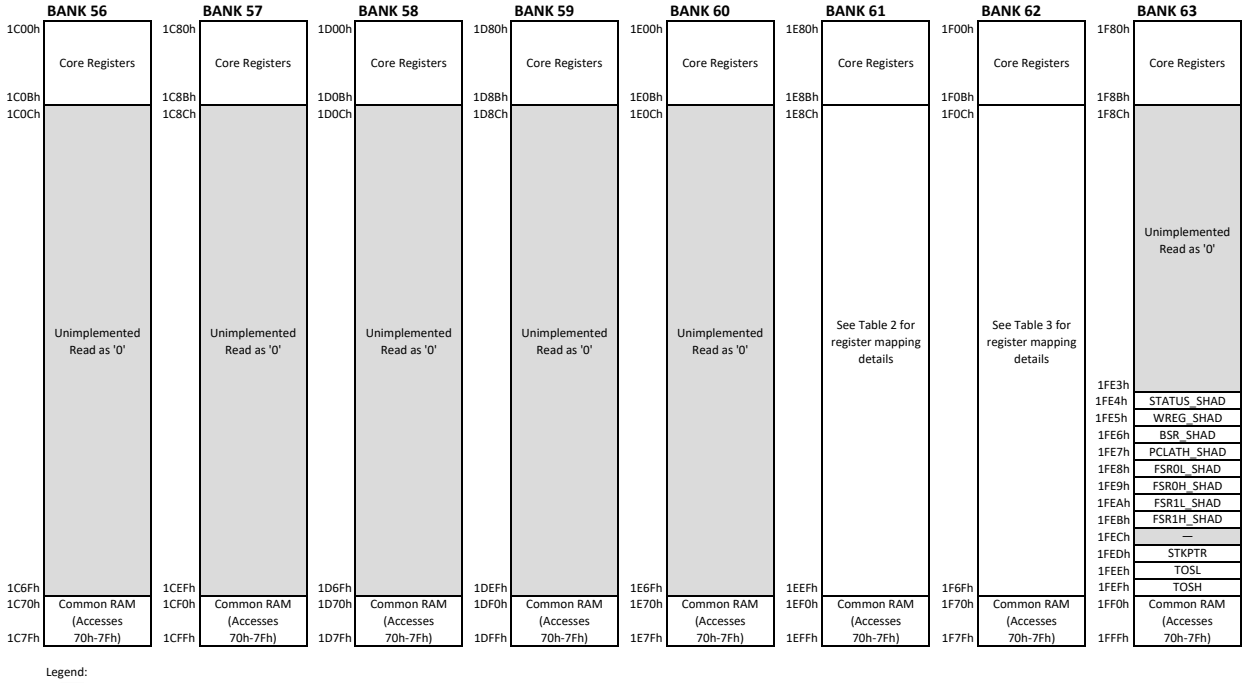


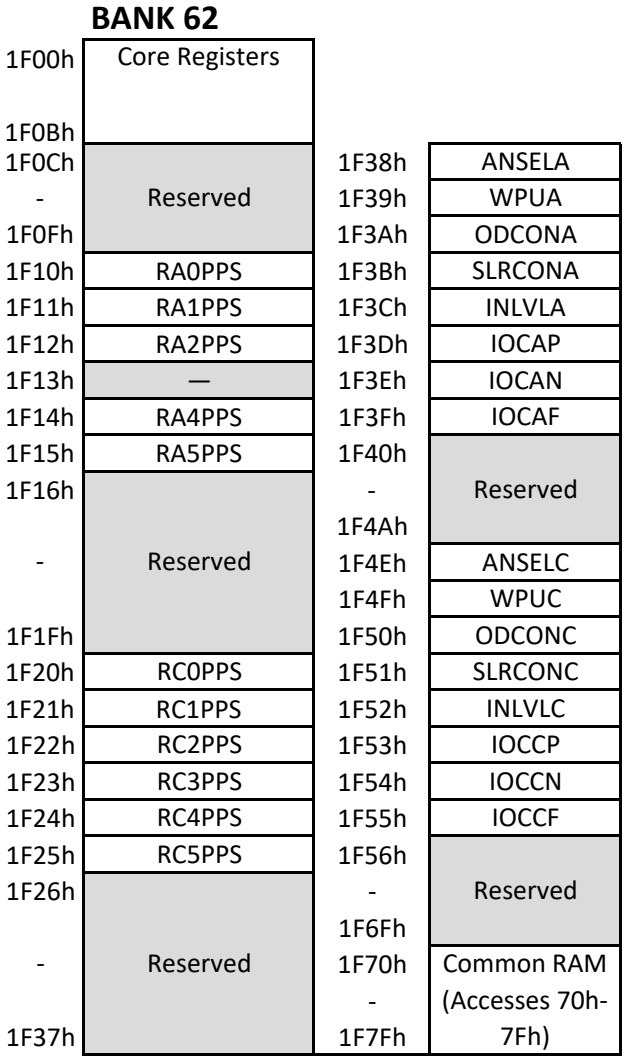
图 7-12. 存储器映射——Bank 61

BANK 61	
1E80h 1E8Bh	Core Registers
1E8Ch	—
1E8Dh	—
1E8Eh	—
1E8Fh	PPSLOCK
1E90h	INTPPS
1E91h	T0CKIPPS
1E92h	T1CKIPPS
1E93h	T1GPPS
1E94h 1E9Bh	—
1E9Ch	T2INPPS
1E9Dh	—
1E9Eh	—
1E9 Fh	—
1EA0h	—
1EA1h	CCP1PPS
1EA2h	CCP2PPS
1EA3h 1EC2h	—
1EC3h	ADACTPPS
1EC4h	—
1EC5h	SSP1CLKPPS
1EC6h	SSP1DATPPS
1EC7h	SSP1SSPPS
1EC8h	—
1EC9h	—
1ECAh	—
1ECBh	RX1PPS
1ECCh	TX1PPS
1ECDh - 1EEFh	—
1EF0h 1EFFh	Common RAM (Accesses 70h-7Fh)

Legend:

Unimplemented data memory locations, read as '0'

图 7-13. 存储器映射——Bank 62



7.3. STATUS 寄存器

STATUS 寄存器包含：

- ALU 的算术运算状态
- 复位状态

与任何其他寄存器一样，STATUS 寄存器可作为任何指令的目标寄存器。如果 STATUS 寄存器是影响 Z、DC 或 C 位的指令的目标寄存器，那么将禁止对这 3 位进行写操作。这些位根据器件逻辑被置 1 或清零。此外， $\overline{TO}$ 和 $\overline{PD}$ 位均不可写。因此，当执行一条将 STATUS 寄存器作为其目标寄存器的指令时，运行结果可能会与预想的不同。

例如，`CLRF STATUS` 会清零 bit [4:3]和[1:0]，并将 Z 位置 1。这将使 STATUS 寄存器中的值保留为 `000u u1uu`（其中 u = 不变）。

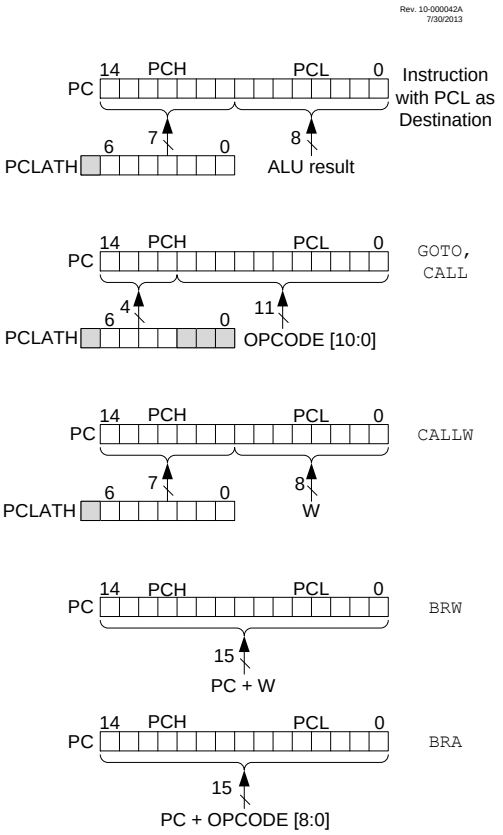
因此，建议仅使用 BCF、BSF、SWAPF 和 MOVWF 指令来改变 STATUS 寄存器的值，因为这些指令不会影响任何状态位。如需了解其他不影响任何状态位的指令，请参见**“指令集汇总”**一章。

重要：在减法运算中，C 位和 DC 位分别作为借位和半借位位。

7.4. PCL 和 PCLATH

程序计数器（Program Counter，PC）为 15 位宽。其低字节来自可读写的 PCL 寄存器。高字节（PC[14:8]）来自 PCLATH，不可直接读写。任何复位都将清零 PC。 图 7-14 显示了装载 PC 的 5 种情形。

图 7-14. 不同情形下 PC 的装载



7.4.1. 修改 PCL

在执行以 PCL 寄存器作为目标寄存器的任何指令的同时，也会使程序计数器的 PC[14:8]位（PCH）被 PCLATH 寄存器的内容所代替。因此可通过将所需的高 7 位写入 PCLATH 寄存器来改变整个程序计数器的内容。当将低 8 位写入 PCL 寄存器时，程序计数器的所有 15 位将变为 PCLATH 寄存器中的值和写入 PCL 寄存器的值。

7.4.2. 计算 GOTO

计算 GOTO 是通过向程序计数器加一个偏移量（ADDWF PCL）来实现的。当使用计算 GOTO 方法执行表读操作时，必须注意表地址是否会导致 PCL 的值超出存储边界（每个存储区为 256 个字节）。

7.4.3. 计算函数调用

利用计算函数 CALL，程序可维护函数表并提供另一种方法来执行状态机或查找表。当使用计算函数 CALL 执行表读操作时，必须注意表地址是否会导致 PCL 的值超出存储边界（每个存储区为 256 个字节）。

如果使用 CALL 指令，PCH[2:0]和 PCL 寄存器中将装入 CALL 指令的操作数。PCH[6:3]中将装入 PCLATH[6:3]。

CALLW 指令通过将 PCLATH 和 W 组合成目标地址来实现计算调用。计算 CALLW 通过将所需地址装入 W 寄存器并执行 CALLW 指令来实现。PCL 寄存器中装入 W 寄存器的值，PCH 中装入 PCLATH 的值。

7.4.4. 转移

转移指令会将一个偏移量与 PC 相加。这样就能实现可重定位代码和跨越页边界的代码。有两种转移形式：BRW 和 BRA。在两种形式中，PC 都会发生递增，以便取下一条指令。使用任一转移指令时，都可以跨越 PCL 存储器边界。

如果使用 BRW，则向 W 寄存器中装入所需的无符号地址，然后执行 BRW。整个 PC 中将装入地址 PC + 1 + W。

如果使用 BRA，则整个 PC 中将装入 PC + 1 + (BRA 指令操作数的有符号值)。

7.5. 堆栈

所有器件都具有 16 级 x15 位宽的硬件堆栈。堆栈既不占用程序存储空间，也不占用数据存储空间。当执行 CALL 或 CALLW 指令或者中断导致程序转移时，PC 值将被压入堆栈。而在执行 RETURN、RETLW 或 RETFIE 指令时，将从堆栈中弹出 PC 值。PCLATH 不受压栈或出栈操作的影响。

如果 STVREN 配置位被设定为 0，堆栈将作为循环缓冲区工作。这意味着在压栈 16 次后，第 17 次压入堆栈的值将会覆盖第一次压栈时所保存的值，而第 18 次压入堆栈的值将覆盖第二次压栈时所保存的值，依此类推。无论是否使能复位，STKOVF 和 STKUNF 标志位都将在上溢/下溢时置 1。

如果 STVREN 位被设定为 1，则在压栈操作超过堆栈第 16 级或出栈操作超过堆栈第 1 级时，器件会发生复位，并将相应位（分别为 STKOVF 或 STKUNF）置 1。

重要：不存在被称为 PUSH 或 POP 的指令/助记符。堆栈的压入或弹出是源于执行了 CALL、CALLW、RETURN、RETLW 和 RETFIE 指令，或源于跳转到中断向量地址。

7.5.1. 访问堆栈

可通过 TOSH、TOSL 和 STKPTR 寄存器来访问堆栈。STKPTR 是堆栈指针的当前值。TOSH:TOSL 寄存器对指向栈顶。这两个寄存器可读/写。由于 PC 的大小为 15 位，所以 TOS 划分为 TOSH 和 TOSL 两部分。要访问堆栈，可调整用来定位 TOSH:TOSL 的 STKPTR 值，然后对 TOSH:TOSL 执行读/写操作。此外，STKPTR 还允许检测上溢和下溢条件。

重要：在允许中断的情况下，修改 STKPTR 时必须谨慎。

在正常程序运行期间，CALL、CALLW 和中断会使 STKPTR 值递增 1，而 RETLW、RETURN 和 RETFIE 会使 STKPTR 值递减 1。可监视 STKPTR 以获取堆栈存储器在任意给定时间保留的值。STKPTR 总是指向堆栈中当前使用的单元。因此，CALL 或 CALLW 指令会先使 STKPTR 值递增 1，然后写入 PC；而返回操作会先从堆栈中删除 PC 值，然后使 STKPTR 值递减 1。

关于访问堆栈的示例，请参见以下各图。

图 7-15. 访问堆栈示例 1

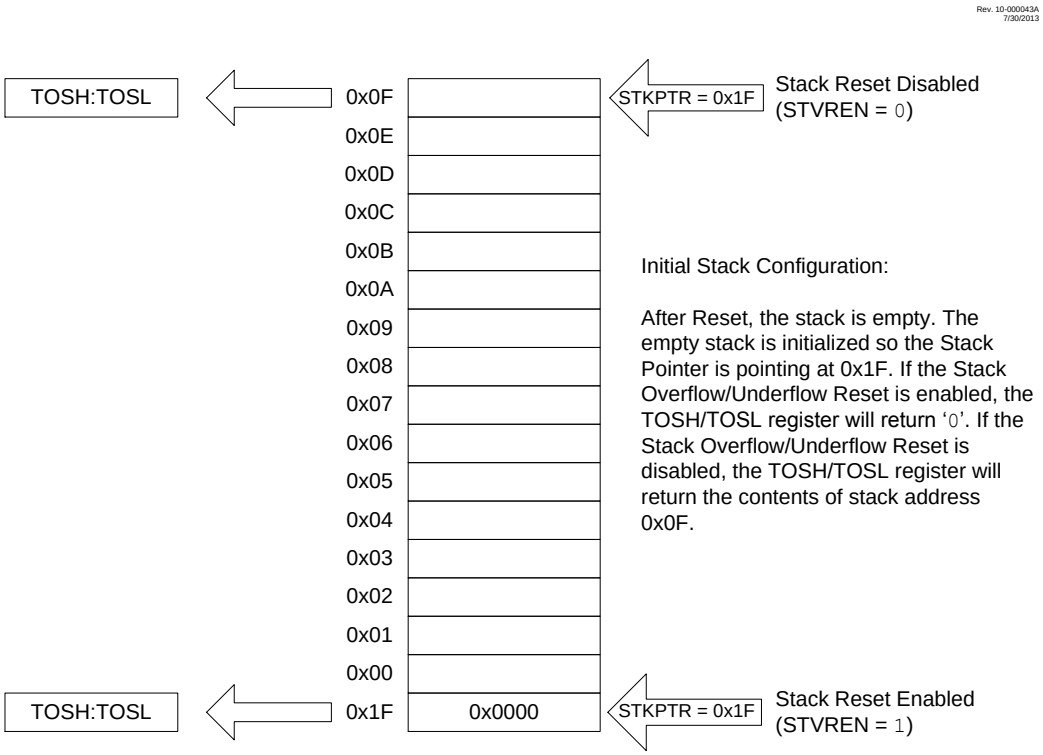


图 7-16. 访问堆栈示例 2

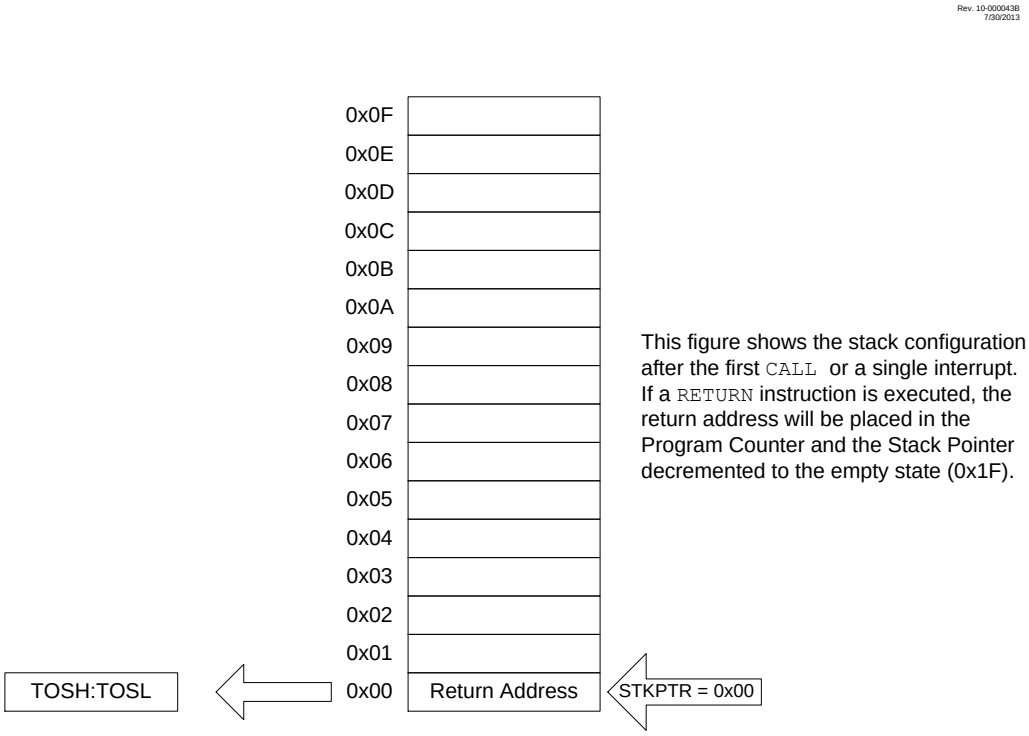


图 7-17. 访问堆栈示例 3

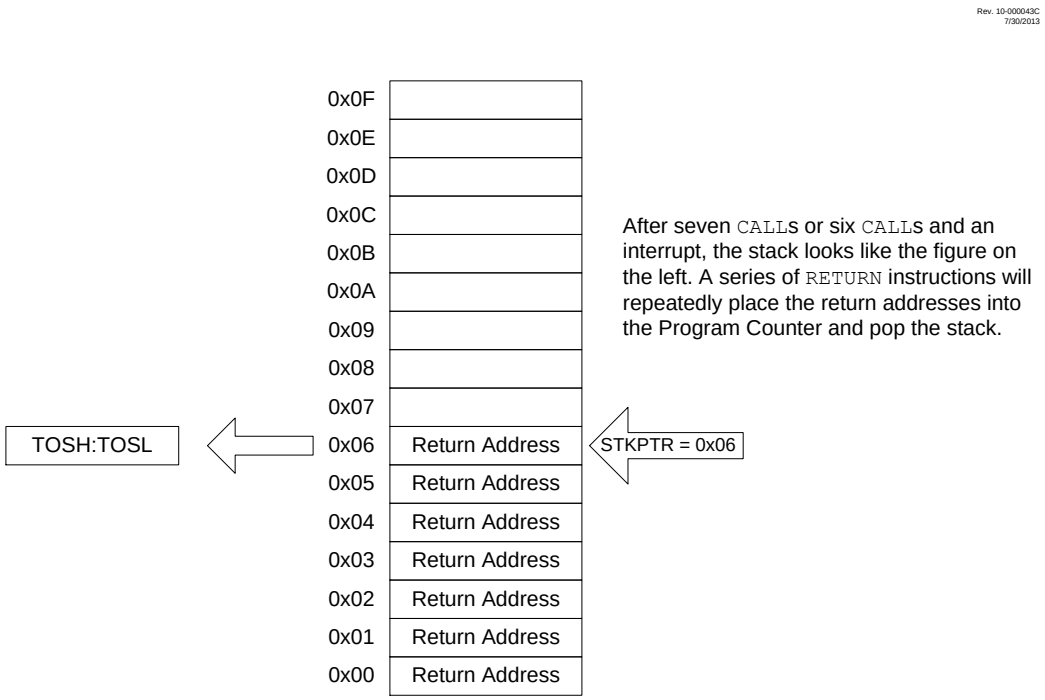
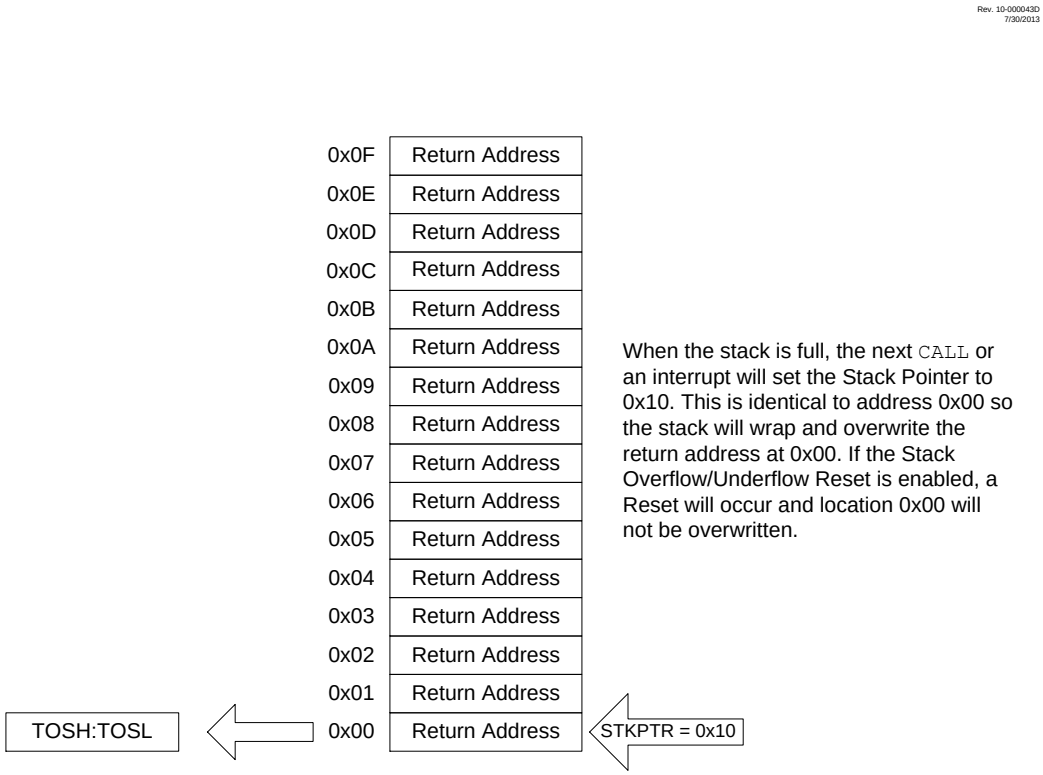


图 7-18. 访问堆栈示例 4



7.5.2. 上溢/下溢复位

如果 STVREN 位被设定为 1，则在压栈操作超过堆栈第 16 级或出栈操作超过堆栈第 1 级时，器件会发生复位，并将相应位（分别为 STKOVF 或 STKUNF）置 1。

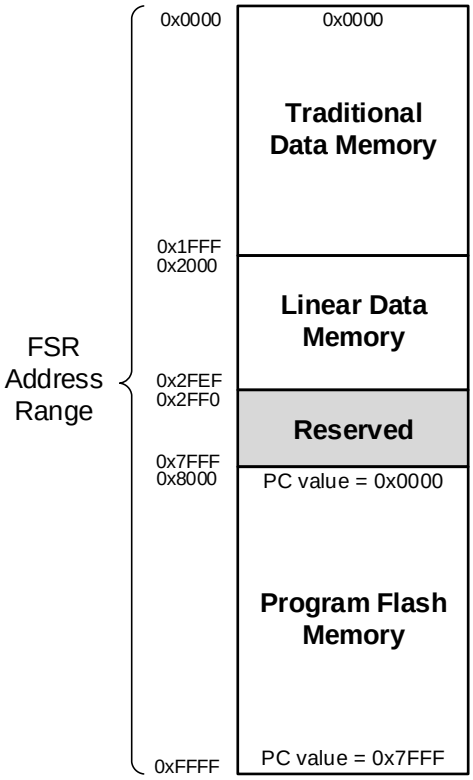
7.6. 间接寻址

INDFn 寄存器不是物理寄存器。访问 INDFn 寄存器的所有指令实际上访问的是由文件选择寄存器（FSR）指定的地址处的寄存器。如果 FSRn 地址指定了 2 个 INDFn 寄存器中的任何一个，执行读操作将返回 0，而写操作无法实现（但状态位会受影响）。FSRn 寄存器值由 FSRnH 和 FSRnL 对构成。

FSR 寄存器构成一个 16 位地址，支持 65536 个存储单元的寻址空间。这些存储单元分为 3 个存储器区域：

- 传统/分区数据存储器
- 线性数据存储器
- 闪存程序存储器

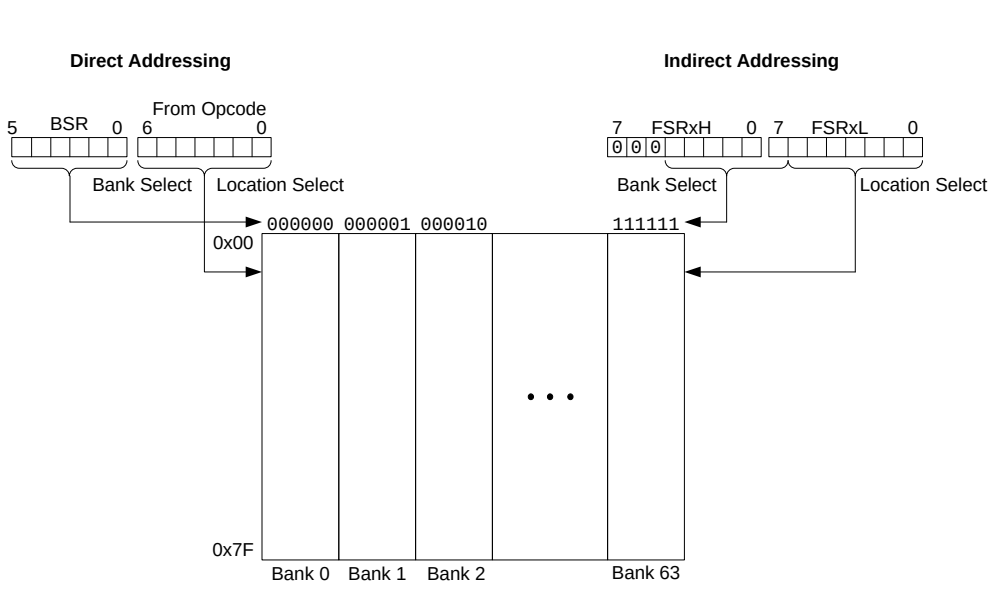
图 7-19. 间接寻址



7.6.1. 传统/分区数据存储器

传统或分区数据存储器指的是从 FSR 地址 0x0000 到 0x1FFF 的区域。此地址对应所有 SFR、GPR 和公共寄存器的绝对地址。

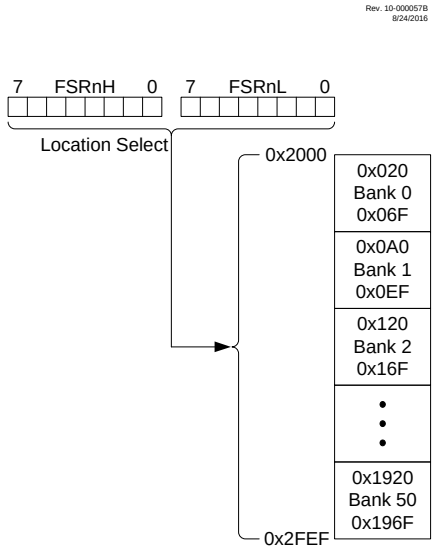
图 7-20. 传统/分区数据存储器映射



7.6.2. 线性数据存储器

线性数据存储器指的是从 FSR 地址 0x2000 到 0x2FEF 的区域。该区域是一个虚拟区域，指向所有存储区中 80 字节的 GPR 存储块。有关线性数据存储器映射，请参见图 7-21。

图 7-21. 线性数据存储器映射



重要：地址范围 0x2000 至 0x2FEF 表示这些器件中完整的可寻址线性数据存储器（最多到 Bank 50）。系列中不同器件能够实现的线性数据存储器各不相同。

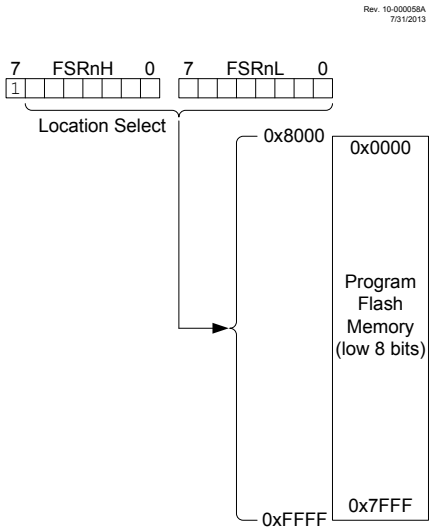
未实现的存储区读为 0x00。通过使用线性数据存储器区域，可以支持大于 80 字节的缓冲区，因为在 FSR 递增至超过一个存储区时，将会直接转至下一个存储区的 GPR 存储器。

线性数据存储器区域不包含 16 字节的公共存储器。

7.6.3. 闪存程序存储器

为了更方便地访问常量数据，整个闪存程序存储器都映射到 FSR 地址空间的上半部分。当 FSRnH 的 MSb 置 1 时，低 15 位就是可通过 INDF 访问的程序存储器的地址。对于每个存储单元，只有低 8 位可通过 INDF 访问。对闪存程序存储器的写操作无法通过 FSR/INDF 接口实现。对于通过 FSR/INDF 接口访问闪存程序存储器的所有指令，都需要一个额外的指令周期才能完成操作。

图 7-22. 闪存程序存储器映射



7.7. 寄存器定义：存储器构成

7.7.1. INDF0

名称： INDF0
偏移量： 0x0000

间接数据寄存器。该寄存器是虚拟寄存器。由 FSR0 寄存器寻址的 GPR/SFR 寄存器是所有操作（包括 INDF0 寄存器）的目标寄存器。

位	7	6	5	4	3	2	1	0
	INDF0[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7:0 – INDF0[7:0]
FSR0 寄存器指向的间接数据

7.7.2. INDF1

名称： INDF1
偏移量： 0x0001

间接数据寄存器。该寄存器是虚拟寄存器。由 FSR1 寄存器寻址的 GPR/SFR 寄存器是所有操作（包括 INDF1 寄存器）的目标寄存器。

位	7	6	5	4	3	2	1	0
	INDF1[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7:0 – INDF1[7:0]
FSR1 寄存器指向的间接数据

7.7.3. PCL

名称： PCL
偏移量： 0x0002

程序计数器的低字节

位	7	6	5	4	3	2	1	0
	PCL[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7:0 – PCL[7:0]
提供对程序计数器的直接读写访问

7.7.4. STATUS

名称： STATUS
偏移量： 0x0003

状态寄存器

位	7	6	5	4	3	2	1	0
				$\overline{TO}$	$\overline{PD}$	Z	DC	C
访问				R	R	R/W	R/W	R/W
复位				1	1	0	0	0

Bit 4 – $\overline{TO}$ 超时

复位状态： POR/BOR = 1
所有其他复位 = q

值	说明
1	上电时置 1，或者通过执行 CLRWDT 或 SLEEP 指令置 1
0	发生了 WDT 超时

Bit 3 – $\overline{PD}$ 掉电

复位状态： POR/BOR = 1
所有其他复位 = q

值	说明
1	上电时置 1，或者通过执行 CLRWDT 指令置 1
0	通过执行 SLEEP 指令清零

Bit 2 – Z 零

复位状态： POR/BOR = 0
所有其他复位 = u

值	说明
1	算术运算或逻辑运算结果为零
0	算术运算或逻辑运算结果不为零

Bit 1 – DC 半进位/$\overline{借位}$(1)

ADDWF、ADDLW、SUBLW 和 SUBWF 指令

复位状态： POR/BOR = 0
所有其他复位 = u

值	说明
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/$\overline{借位}$(1)

ADDWF、ADDLW、SUBLW 和 SUBWF 指令

复位状态： POR/BOR = 0
所有其他复位 = u

值	说明
1	结果的最高有效位发生了进位
0	结果的最高有效位未发生进位

注：

1. 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（RRCF 和 RLCF），此位中将装入源寄存器的高位或低位。

7.7.5. FSR0

名称： FSR0
偏移量： 0x0004

间接地址寄存器

FSR0 值是 INDF0 寄存器指向的数据的地址。

位	15	14	13	12	11	10	9	8
	FSR0[15:8]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

位	7	6	5	4	3	2	1	0
	FSR0[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 15:0 – FSR0[15:0] INDF0 数据的地址

注：该多字节寄存器中的各个字节可使用以下寄存器名称进行访问：

1. FSR0H：访问高字节 FSR0[15:8]。
2. FSR0L：访问低字节 FSR0[7:0]。

7.7.6. FSR1

名称： FSR1
偏移量： 0x0006

间接地址寄存器

FSR1 是 INDF1 寄存器指向的数据的地址。

位	15	14	13	12	11	10	9	8
	FSR1[15:8]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

位	7	6	5	4	3	2	1	0
	FSR1[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 15:0 – FSR1[15:0]
INDF1 数据的地址

注：该多字节寄存器中的各个字节可使用以下寄存器名称进行访问：

1. FSR1H：访问高字节 FSR1[15:8]。
2. FSR1L：访问低字节 FSR1[7:0]。

7.7.7. BSR

名称： BSR
偏移量： 0x0008

存储区选择寄存器
BSR 通过将存储区号写入寄存器来指示数据存储区。所有数据存储器既可通过指令直接访问，也可通过 FSR 间接访问。

位	7	6	5	4	3	2	1	0
			BSR[5:0]					
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			0	0	0	0	0	0

Bit 5:0 – BSR[5:0]
数据存储地址的高六位

7.7.8. WREG

名称： WREG
偏移量： 0x0009

工作数据寄存器

位	7	6	5	4	3	2	1	0
	WREG[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7:0 – WREG[7:0]

7.7.9. PCLATH

名称： PCLATH
偏移量： 0x000A

程序计数器锁存器

程序计数器高 7 位的写缓冲区

位	7	6	5	4	3	2	1	0
		PCLATH[6:0]						
访问		R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位		0	0	0	0	0	0	0

Bit 6:0 – PCLATH[6:0] PC 锁存器高字节寄存器
程序计数器位[6:0]的保持寄存器

7.8. 寄存器汇总——存储器构成

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00	INDF0	7:0	INDF0[7:0]							
0x01	INDF1	7:0	INDF1[7:0]							
0x02	PCL	7:0	PCL[7:0]							
0x03	STATUS	7:0				$\overline{TO}$	$\overline{PD}$	Z	DC	C
0x04	FSR0	7:0	FSR0[7:0]							
		15:8	FSR0[15:8]							
0x06	FSR1	7:0	FSR1[7:0]							
		15:8	FSR1[15:8]							
0x08	BSR	7:0			BSR[5:0]					
0x09	WREG	7:0	WREG[7:0]							
0x0A	PCLATH	7:0		PCLATH[6:0]						

8. 复位

有多种方式可以复位此器件：

- 上电复位（POR）
- 欠压复位（BOR）
- $\overline{MCLR}$ 复位
- WDT 复位
- RESET 指令
- 堆栈上溢
- 堆栈下溢
- 退出编程模式

要使 V_{DD} 稳定，可以使能可选的上电延时定时器，以在 BOR 或 POR 事件后延长复位时间。

8.1. 上电复位（POR）

POR 电路将器件保持在复位状态，直到 V_{DD} 达到足以使器件正常工作的最低电平。V_{DD} 上升缓慢、高速运行或要求一定模拟性能时，所需电压可能高于最低 V_{DD}。在满足所有器件工作条件之前，可以使用 PWRT、BOR 或 $\overline{MCLR}$ 功能延长启动周期。

8.1.1. 退出编程模式

在退出编程模式时，器件的行为与刚刚发生 POR 时的情况相同。

8.2. 欠压复位（BOR）

当 V_{DD} 达到可选的最低电平时，BOR 电路将器件保持在复位状态。在 POR 和 BOR 之间，可在整个电压范围内对器件的执行进行保护。

欠压复位模块有 4 种工作模式，由 BOREN 位控制。这 4 种工作模式是：

- BOR 始终使能
- BOR 在休眠时禁止
- BOR 由软件控制
- BOR 始终禁止

更多信息，请参见表 8-1。

通过对 BORV 位进行配置，可以选择欠压复位电压。

V_{DD} 噪声抑制滤波器可以防止在发生小事件时触发 BOR。如果 V_{DD} 下降到 V_{BOR} 以下且持续时间大于参数 T_{BORDC}，则器件将复位，$\overline{BOR}$ 位将清零以表示发生欠压复位条件。请参见图 8-1。

8.2.1. BOR 始终使能

BOREN 位设置为 11 时，BOR 始终使能。器件启动延时直到 BOR 就绪并且 V_{DD} 高于 BOR 阈值。

BOR 保护功能在休眠期间有效。BOR 不会使从休眠唤醒延时。

8.2.2. BOR 在休眠时禁止

BOREN 位设置为 10 时，除了在休眠时之外，BOR 始终使能。BOR 保护在休眠期间无效，但器件唤醒会被延迟，直到 BOR 可确定 V_{DD} 高于 BOR 阈值。器件唤醒延时直到 BOR 就绪。

8.2.3. BOR 由软件控制

当配置字的 BOREN 位编程为 01 时，BOR 将通过 SBOREN 位进行控制。器件启动不会被 BOR 就绪条件或 V_{DD} 电平延时。

BOR 保护会在 BOR 电路就绪时立即开始。BOR 电路的状态在 BORRDY 位中反映。

BOR 保护功能不受休眠影响。

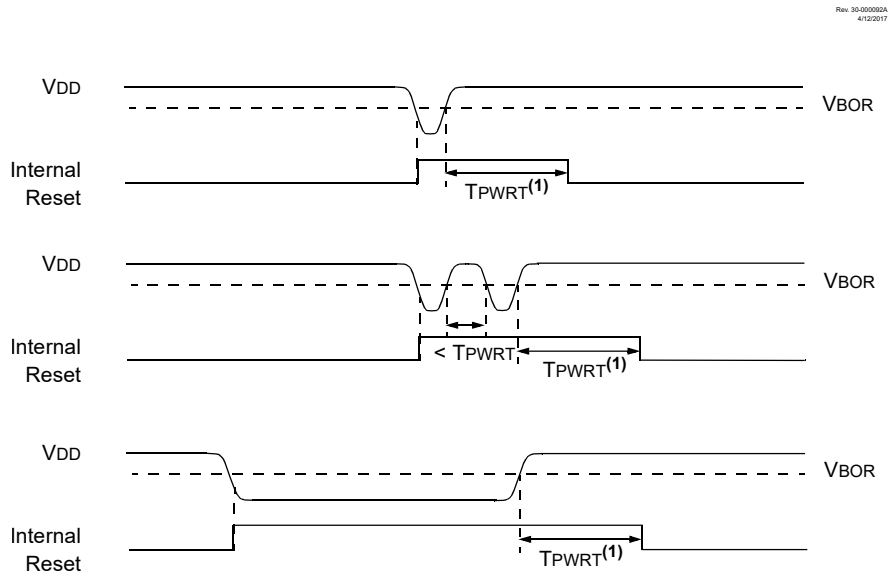
表 8-1. BOR 工作模式

BOREN	SBOREN	器件模式	BOR 模式	在以下情况下执行的指令：	
				POR 释放	从休眠模式唤醒
11(1)	X	X	活动	等待 BOR 释放（BORRDY = 1）	立即开始
10	X	唤醒	活动	等待 BOR 释放（BORRDY = 1）	N/A
		休眠	冬眠	N/A	等待 BOR 释放（BORRDY = 1）
01	1	X	活动	等待 BOR 释放（BORRDY = 1）	立即开始
	0	X	冬眠		
00	X	X	禁止	立即开始	

注：

1. 在“POR 释放”和“从休眠模式唤醒”的特定情况下，启动时没有任何延时。在 CPU 准备好执行指令之前，BOR 就绪标志会置 1（BORRDY = 1），这是因为 BOR 电路通过 BOREN 位被强制使能。

图 8-1. 欠压情形



注：当 PWRTS 位使能（$\overline{PWRTS}$!= 00）时，T_{PWRT} 延迟。

8.2.4. BOR 始终禁止

BOREN 位设置为 00 时，BOR 始终禁止。在该配置中，将 SBOREN 位置 1 对 BOR 操作没有影响。

8.3. $\overline{MCLR}$ 复位

$\overline{MCLR}$ 是可复位器件的可选外部输入。$\overline{MCLR}$ 功能由 MCLRE 位和 LVP 位控制（见表 8-2）。如果发生 $\overline{MCLR}$，$\overline{RMCLR}$ 位将设置为 0。

表 8-2. $\overline{\text{MCLR}}$ 配置

MCLRE	LVP	$\overline{\text{MCLR}}$
x	1	使能
1	0	使能
0	0	禁止

8.3.1. $\overline{\text{MCLR}}$ 使能

当使能 $\overline{\text{MCLR}}$ 并且引脚保持低电平时，器件会保持在复位状态。$\overline{\text{MCLR}}$ 引脚通过内部弱上拉连接到 V_{DD}。

器件在 $\overline{\text{MCLR}}$ 复位路径上有一个噪声滤波器。该滤波器能检测并滤除小脉冲。

重要：内部复位事件（RESET 指令、BOR、WDT、POR、STKOVF、STKUNF）不会将 $\overline{\text{MCLR}}$ 引脚驱动为低电平。

8.3.2. $\overline{\text{MCLR}}$ 禁止

当 $\overline{\text{MCLR}}$ 被禁止时，$\overline{\text{MCLR}}$ 仅用作输入，内部弱上拉等引脚功能由软件控制。

8.4. 看门狗定时器（WDT）复位

如果固件在超时周期内未发出 CLRWDT 指令，看门狗定时器将产生复位。TO、PD 和 $\overline{\text{RWDT}}$ 位会发生改变，指示定时器溢出导致 WDT 复位。

8.5. RESET 指令

RESET 指令将导致器件复位。 $\overline{\text{RI}}$ 位将设置为 0。关于发生 RESET 指令之后的默认条件，请参见表 8-4。

8.6. 堆栈上溢/下溢复位

器件可以在堆栈上溢或下溢时复位。STKOVF 或 STKUNF 位指示复位条件。通过将 STVREN 位置 1 可以使能这些复位。

8.7. 上电延时定时器（PWRT）

上电延时定时器在 POR 或 BOR 时提供最高 64 ms 的延时。只要 PWRT 处于活动状态，器件就保持在复位状态。PWRT 延时为 V_{DD} 上升到所需的电平提供额外的时间。

上电延时定时器由 PWRTS 位控制。上电延时定时器在 POR 和 BOR 释放后启动。

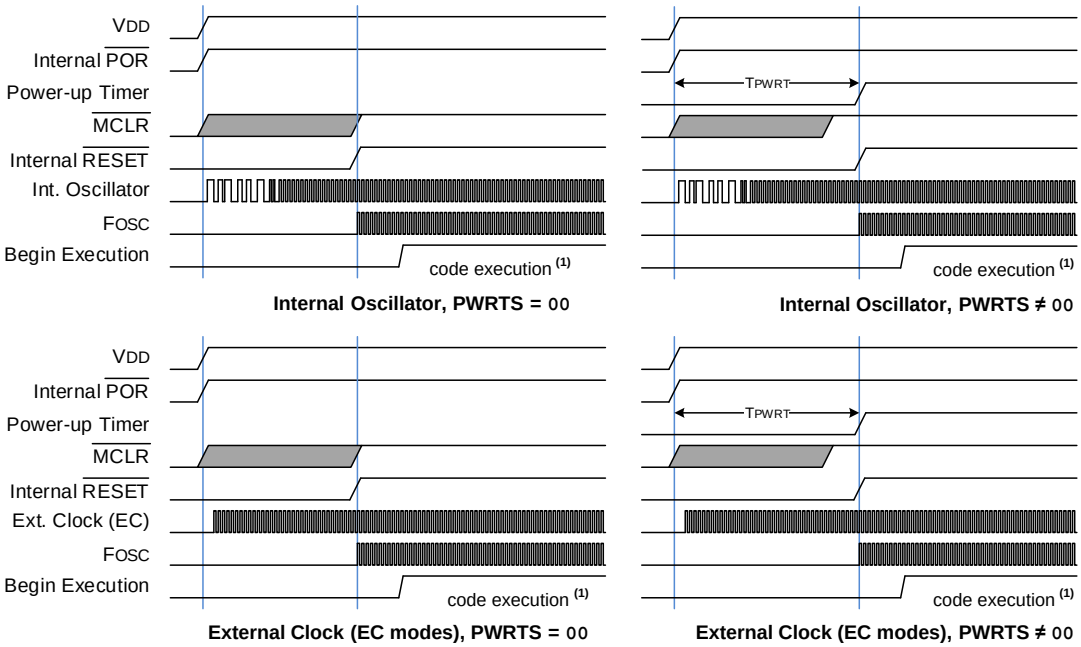
8.8. 启动序列

在 POR 或 BOR 释放时，只有先发生以下事件，器件才会开始执行代码：

1. 上电延时定时器运行结束（如果使能）。
2. $\overline{\text{MCLR}}$ 必须被释放（如果使能）。

上电延时定时器独立于 $\overline{\text{MCLR}}$ 复位运行。如果 $\overline{\text{MCLR}}$ 保持足够长时间的低电平，上电延时定时器将超时。将 $\overline{\text{MCLR}}$ 电平拉高后，器件将在 10 个 F_{OSC} 周期后开始执行代码（见图 8-2）。这对于测试或同步多个并行工作的器件来说非常有用。

图 8-2. 复位启动时序



注：

1. 将在 F_{OSC} 时钟释放 10 个 F_{OSC} 周期后开始执行代码。

8.9. 存储器执行违例

执行从有效执行区域之外取出的指令时会触发存储器执行违例复位。无效执行区域包括：

1. 所实现程序存储器以外的地址。有关可用闪存大小的详细信息，请参见**“存储器构成”**一章。
2. 程序存储器内的存储区闪存（SAF）（如果已使能）。

产生存储器执行违例时，器件复位且 $\overline{MEMV}$ 位清零以指示复位原因。在发生存储器执行违例复位后，必须在用户代码中将 $\overline{MEMV}$ 位置 1 以继续检测违例复位。

8.10. 确定复位原因

在发生任何复位时，STATUS、PCON0 和 PCON1 寄存器中会有多个位发生更新，以指示复位的原因。下表列出了这些寄存器的复位条件。

表 8-3. 复位状态位及其含义

STKOVF	STKUNF	$\overline{RWDT}$	$\overline{RMCLR}$	$\overline{RI}$	POR	$\overline{BOR}$	$\overline{TO}$	PD	$\overline{MEMV}$	条件
0	0	1	1	1	0	x	1	1	1	上电复位
0	0	1	1	1	0	x	0	x	u	非法，$\overline{TO}$ 在 POR 时被置 1
0	0	1	1	1	0	x	x	0	u	非法，$\overline{PD}$ 在 POR 时被置 1
0	0	u	1	1	u	0	1	1	u	欠压复位
u	u	0	u	u	u	u	0	u	u	WDT 复位
u	u	u	u	u	u	u	0	0	u	WDT 从休眠模式唤醒
u	u	u	u	u	u	u	1	0	u	通过中断从休眠模式唤醒

表 8-3. 复位状态位及其含义（续）

STKOVF	STKUNF	RWDT	RMCLR	RI	POR	BOR	TO	PD	MEMV	条件
u	u	u	0	u	u	u	u	u	1	MCLR 在正常工作期间复位
u	u	u	0	u	u	u	1	0	u	MCLR 在休眠期间复位
u	u	u	u	0	u	u	u	u	u	RESET 执行指令
1	u	u	u	u	u	u	u	u	u	堆栈上溢复位（STVREN = 1）
u	1	u	u	u	u	u	u	u	u	堆栈下溢复位（STVREN = 1）
u	u	u	u	u	u	u	u	u	0	存储器违例复位

表 8-4. 特殊寄存器的复位条件

条件	程序计数器	状态寄存器	PCON0 寄存器	PCON1 寄存器
上电复位	0	---1 1000	0011 110x	---- --1-
欠压复位	0	---1 1000	0011 11u0	---- --u-
MCLR 在正常工作期间复位	0	-uuu uuuu	uuuu 0uuu	---- --1-
MCLR 在休眠期间复位	0	---1 0uuu	uuuu 0uuu	---- --u-
WDT 超时复位	0	---0 uuuu	uuu0 uuuu	---- --u-
WDT 从休眠模式唤醒	PC + 1	---0 0uuu	uuuu uuuu	---- --u-
通过中断从休眠模式唤醒	PC + 1(1)	---1 0uuu	uuuu uuuu	---- --u-
RESET 执行指令	0	---u uuuu	uuuu u0uu	---- --u-
堆栈上溢复位（STVREN = 1）	0	---u uuuu	1uuu uuuu	---- --u-
堆栈下溢复位（STVREN = 1）	0	---u uuuu	u1uu uuuu	---- --u-
存储器违例复位	0	-uuu uuuu	uuuu uuuu	---- --0-

图注： uu = 不变，x = 未知， - = 未实现位，读为 0。

注：

1. 如果器件被中断唤醒且全局中断允许（GIE）位置 1，则执行 PC + 1 后，返回地址被压入堆栈且 PC 装入中断向量（0004h）。

8.11. 电源控制（PCONx）寄存器

电源控制（PCONx）寄存器包含区分以下复位的标志位：

- 欠压复位（BOR）
- 上电复位（POR）
- RESET 指令复位（RI）
- MCLR 复位（RMCLR）
- 看门狗定时器复位（RWDT）
- 堆栈下溢复位（STKUNF）
- 堆栈上溢复位（STKOVF）
- 存储器违例复位（MEMV）

硬件将在复位过程中改变相应的寄存器位；如果复位不是由相应条件引起的，对应位保持不变。

在重启后，软件可将相应位复位为无效状态（硬件不会复位相应位）。

软件还可将任意 PCONx 位设置为有效状态，这样可测试用户代码，但不会产生任何复位操作。

8.12. 寄存器定义：电源控制

8.12.1. BORCON

名称： BORCON
偏移量： 0x811

欠压复位控制寄存器

位	7	6	5	4	3	2	1	0
	SBOREN							BORRDY
访问	R/W							R
复位	1							q

Bit 7 – SBOREN 软件欠压复位使能

复位状态： POR/BOR = 1
所有其他复位 = u

值	条件	说明
—	如果 BOREN ≠01	SBOREN 位是可读写的，但是对欠压复位没有影响
1	如果 BOREN =01	使能 BOR
0	如果 BOREN =01	禁止 BOR

Bit 0 – BORRDY 欠压复位电路就绪状态

复位状态： POR/BOR = q
所有其他复位 = u

值	说明
1	欠压复位电路有效且已就绪
0	欠压复位电路已禁止或仍在预热阶段

8.12.2. PCON0

名称： PCON0
偏移量： 0x813

电源控制寄存器 0

位	7	6	5	4	3	2	1	0
	STKOVF	STKUNF		$\overline{RWDT}$	$\overline{RMCLR}$	$\overline{RI}$	$\overline{POR}$	$\overline{BOR}$
访问	R/W/HS	R/W/HS		R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC
复位	0	0		1	1	1	0	q

Bit 7 – STKOVF 堆栈上溢标志

复位状态： POR/BOR = 0
所有其他复位 = q

值	说明
1	发生了堆栈上溢（CALL 数量超出堆栈范围）
0	未发生堆栈上溢或该位由固件设置为 0

Bit 6 – STKUNF 堆栈下溢标志

复位状态： POR/BOR = 0
所有其他复位 = q

值	说明
1	发生了堆栈下溢（RETURN 多于 CALL）
0	未发生堆栈下溢或该位由固件设置为 0

Bit 4 – $\overline{RWDT}$ WDT 复位标志

复位状态： POR/BOR = 1
所有其他复位 = q

值	说明
1	未发生 WDT 上溢/超时复位或由固件置 1
0	发生了 WDT 上溢/超时复位（发生 WDT 复位时由硬件设置为 0）

Bit 3 – $\overline{RMCLR}$ $\overline{MCLR}$ 复位标志

复位状态： POR/BOR = 1
所有其他复位 = q

值	说明
1	$\overline{MCLR}$ 复位未发生或由固件置 1
0	$\overline{MCLR}$ 发生了复位（发生 $\overline{MCLR}$ 复位时由硬件设置为 0）

Bit 2 – $\overline{RI}$ RESET 指令标志

复位状态： POR/BOR = 1
所有其他复位 = q

值	说明
1	未执行 RESET 指令或由固件置 1
0	执行了 RESET 指令（执行 RESET 指令时由硬件设置为 0）

Bit 1 – $\overline{POR}$ 上电复位状态

复位状态： POR/BOR = 0

所有其他复位 = u

值	说明
1	未发生上电复位或由固件置 1
0	发生了上电复位（发生上电复位时由硬件设置为 0）

Bit 0 – $\overline{\text{BOR}}$ 欠压复位状态

复位状态： POR/BOR = q
所有其他复位 = u

值	说明
1	未发生欠压复位或由固件置 1
0	发生了欠压复位（发生欠压复位时由硬件设置为 0）

8.12.3. PCON1

名称： PCON1
偏移量： 0x814

电源控制寄存器 1

位	7	6	5	4	3	2	1	0
							$\overline{\text{MEMV}}$	
访问							R/W/HC	
复位							1	

Bit 1 – $\overline{\text{MEMV}}$ 存储器违例标志

复位状态： POR/BOR = 1
所有其他复位 = u

值	说明
1	未发生存储器违例复位或由固件置 1。
0	发生了存储器违例复位（发生存储器违例复位时由硬件设置为 0）

8.13. 寄存器汇总——电源控制

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0810	保留									
0x0811	BORCON	7:0	SBOREN							BORRDY
0x0812	保留									
0x0813	PCON0	7:0	STKOVF	STKUNF		$\overline{RWDT}$	$\overline{RMCLR}$	$\overline{RI}$	$\overline{POR}$	$\overline{BOR}$
0x0814	PCON1	7:0							$\overline{MEMV}$	

9. OSC——振荡器模块

9.1. 振荡器模块概述

振荡器模块包含多种时钟源和选择特性，不仅能够广泛用于各种应用，还能最大程度地提高性能并降低功耗。

时钟源既可以由内部提供，也可以从外部提供。外部源包括：

- 外部时钟（EC）振荡器

内部源包括：

- 高频内部振荡器（HFINTOSC）
- 低频内部振荡器（LFINTOSC）
- 模数转换器 RC 振荡器（ADCRC）

振荡器模块具有以下特性：

- HFINTOSC 频率调节：可调节 HFINTOSC 频率。

复位振荡器（RSTOSC）配置位决定在器件复位后运行（包括器件初次上电）时使用的振荡器类型（见下表）。

表 9-1. RSTOSC 选择表

RSTOSC	SFR 复位值		时钟源
	COSC	OSCFRQ	
11	11	000	EXTOSC 遵循 FEXTOSC
10	10		HFINTOSC @ 1 MHz
01	01		LFINTOSC
00	00	101	HFINTOSC @ 32 MHz

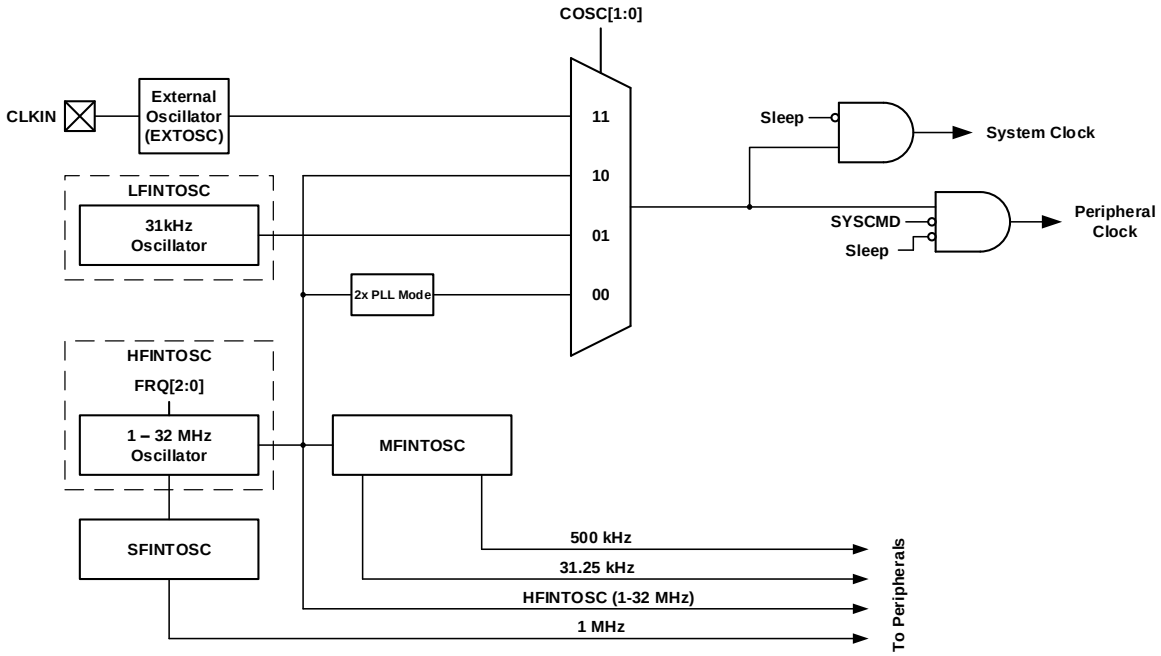
如果通过 RSTOSC 位选择外部时钟源，必须使用外部振荡器模式选择（FEXTOSC）配置位来选择外部时钟模式。外部时钟模式包括：

- ECL：外部时钟低功耗模式
- ECH：外部时钟高功耗模式

ECH 和 ECL 模式依靠外部逻辑电平信号作为器件时钟源。针对特定的频率范围对每种模式进行了优化。内部振荡器模块可以产生低频和高频时钟信号，分别用 LFINTOSC 和 HFINTOSC 表示。这两个时钟源可产生多种系统工作频率。

图 9-1 给出了振荡器模块的框图。

图 9-1. 时钟源框图



9.2. 时钟源类型

时钟源可以分为外部或内部两类。

外部时钟源依赖外部电路作为时钟源工作，例如数字振荡器模块。

内部时钟源内置在振荡器模块中。内部振荡器模块具有两个内部振荡器，用于产生内部系统时钟源。高频内部振荡器（HFINTOSC）可产生各种频率，这些频率通过 HFINTOSC 频率选择（OSCFRQ）寄存器确定。低频内部振荡器（LFINTOSC）产生固定的 31 kHz 标称时钟信号。内部振荡器模块还具有专用于模数转换器（ADC）的 RC 振荡器（ADCRC）。

重要：CN5225 单片机系列不允许通过时钟切换来更改系统时钟源。一旦 RSTOSC 配置位选择了振荡器源，就不能通过软件进行更改。如果选择 HFINTOSC 作为时钟源，可通过修改 FRQ 位来更改 HFINTOSC 频率。

当 CLKOUT 引脚不使用时，指令时钟（$F_{OSC}/4$）可输送到该引脚。时钟输出使能（$\overline{\text{CLKOUTEN}}$）配置位用于控制 CLKOUT 信号的功能。当 $\overline{\text{CLKOUTEN}}$ 清零（$\overline{\text{CLKOUTEN}}$= 0）时，CLKOUT 信号输送到 CLKOUT 引脚。当 $\overline{\text{CLKOUTEN}}$ 置 1（$\overline{\text{CLKOUTEN}}$= 1）时，CLKOUT 引脚用作 I/O 引脚。

9.2.1. 外部时钟源

通过执行以下操作，可将外部时钟源用作器件系统时钟：

- 编程 RSTOSC 配置位以选择外部时钟源（RSTOSC = 111）
- 编程 FEXTOSC 配置位以选择合适的外部时钟（EC）模式：
  - ECH 模式用于以 16 MHz 及更高频率工作的振荡器（FEXTOSC = 11）

- ECL 模式用于以低于 16 MHz 的频率工作的振荡器（FEXTOSC = 01）

9.2.1.1. EC 模式

外部时钟（EC）模式将外部产生的逻辑电平信号作为系统时钟源。在 EC 模式下工作时，外部时钟源应连接到 CLKIN 输入引脚。CLKOUT 引脚可用作通用 I/O 引脚或 CLKOUT 信号引脚。

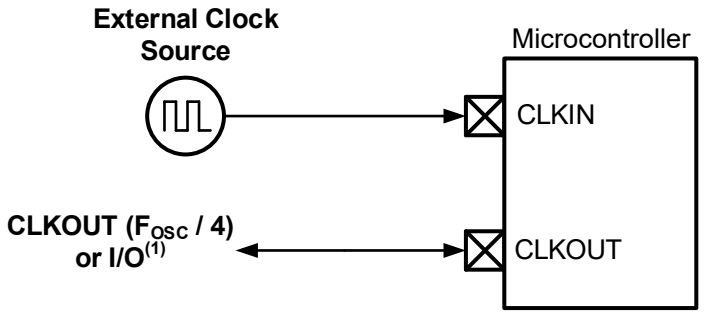
EC 模式有两种功耗模式可供选择：

- ECH：高功耗模式
- ECL：低功耗模式

当选择 EC 模式时，上电复位（POR）后或者从休眠中唤醒后的操作不存在延时。因为单片机设计是完全静态的，停止外部时钟输入将使器件暂停工作并保持所有数据完整。当再次启动外部时钟时，器件恢复工作，就好像没有停止过一样。

图 9-2 给出了 EC 模式的引脚连接图。

图 9-2. 外部时钟（EC）模式工作原理



注：

1. 输出取决于 $\overline{\text{CLKOUTEN}}$ 配置位的设置。

9.2.2. 内部时钟源

内部振荡器模块包含两个独立的振荡器，可产生两个内部系统时钟源：

- 高频内部振荡器（HFINTOSC）
- 低频内部振荡器（LFINTOSC）

通过编程 RSTOSC 配置位选择 INTOSC 源之一，可使用内部振荡器源作为器件系统时钟。

在 INTOSC 模式下，CLKIN 和 CLKOUT 引脚可用作通用 I/O，前提是未连接任何外部振荡器。CLKOUT 引脚的功能由 $\overline{\text{CLKOUTEN}}$ 配置位决定。当 $\overline{\text{CLKOUTEN}}$ 置 1（$\overline{\text{CLKOUTEN}}$= 1）时，该引脚用作通用 I/O。当 $\overline{\text{CLKOUTEN}}$ 清零（$\overline{\text{CLKOUTEN}}$= 0）时，系统指令时钟（$F_{OSC}$/4）用作该引脚上的输出信号。

9.2.2.1. HFINTOSC

高频内部振荡器（HFINTOSC）是一个经过出厂校准的高精度数字控制内部时钟源，可产生各种稳定的时钟频率。通过编程 RSTOSC 配置位选择器件复位或上电后的两个 HFINTOSC 选项之一，可使能 HFINTOSC。

通过 HFINTOSC 频率选择（FRQ）位选择 HFINTOSC 频率。通过 HFINTOSC 频率调节（TUN）位微调 HFINTOSC。

9.2.2.1.1. HFINTOSC 频率调节

HFINTOSC 频率可通过 HFINTOSC 频率调节寄存器（OSCTUNE）微调。OSCTUNE 寄存器提供对 HFINTOSC 标称频率的微小调节。

OSCTUNE 寄存器包含 HFINTOSC 频率调节（TUN）位。TUN 位默认为 6 位二进制补码值 0x00，表示振荡器在所选频率下工作。向 TUN 位写入 0x01 与 0x1F 之间的值时，HFINTOSC 频率增大。向 TUN 位写入 0x3F 与 0x20 之间的值时，HFINTOSC 频率减小。

当修改 OSCTUNE 寄存器时，振荡器频率将开始转变到新的频率。在频率转变期间代码继续执行。不会有任何迹象表明频率发生了转变。

重要：OSCTUNE 调节不会影响 LFINTOSC 频率。

9.2.2.2. MFINTOSC

中频内部振荡器（MFINTOSC）可产生两个恒定时钟输出（500 kHz 和 31.25 kHz）。MFINTOSC 时钟信号是使用动态分频器逻辑从 HFINTOSC 生成的信号。无论选择哪种 HFINTOSC 频率，动态分频器逻辑均可确保提供恒定的 MFINTOSC 时钟速率。

MFINTOSC 无法用作系统时钟，但可用作特定外设（例如定时器）的时钟源。

9.2.2.3. SFINTOSC

指定频率内部振荡器（SFINTOSC）用于生成 1 MHz 的输出时钟。SFINTOSC 时钟信号是使用动态分频器逻辑从 HFINTOSC 生成的信号。无论选择哪种 HFINTOSC 频率，动态分频器逻辑均可确保提供恒定的 SFINTOSC 时钟速率。

SFINTOSC 无法用作系统时钟，但可选作特定外设（例如定时器）的时钟源。

9.2.2.4. LFINTOSC

低频内部振荡器（LFINTOSC）在出厂时已校准为 31 kHz 内部时钟源。

LFINTOSC 可用作系统时钟源，并且可用作特定外设模块的时钟源。此外，LFINTOSC 还为以下定时器提供时基：

- 上电延时定时器（PWRT）
- 看门狗定时器（WDT）

通过编程 RSTOSC 配置位选择 LFINTOSC，可使能 LFINTOSC。

9.2.2.5. ADCRC

模数转换器 RC（ADCRC）振荡器专用于 ADC 模块。该振荡器也称为 FRC 时钟。ADCRC 以大约 600 kHz 的固定频率运行，并用作转换时钟源。ADCRC 允许 ADC 模块在休眠模式下运行，从而可以在 ADC 转换期间降低系统噪声。当选择 ADCRC 作为 ADC 模块的时钟源，或作为可能使用该振荡器的任何外设的时钟源时，ADCRC 会自动使能。也可以通过 ADC 振荡器使能（ADOEN）位手动使能 ADCRC，从而避免间断使用该源时产生的起振延时。

9.2.3. 振荡器状态和手动使能

振荡器状态（OSCSTAT）寄存器显示以下每个振荡器的就绪状态：

- HFINTOSC
- MFINTOSC
- LFINTOSC
- ADCRC

- SFINTOSC

HFINTOSC 振荡器就绪（HFOR）、MFINTOSC 振荡器就绪（MFOR）、LFINTOSC 振荡器就绪（LFOR）、ADCRC 振荡器就绪（ADOR）和 SFINTOSC 振荡器就绪（SFOR）状态位指示相应振荡器是否准备就绪。这些时钟源均可随时使用，但可能需要一定的时间才能达到指定的精度等级。当振荡器准备就绪并达到指定精度时，模块硬件会将相应的位置 1。

当有新值载入 OSCFRQ 寄存器时，HFOR 位将由硬件清零，并在 HFINTOSC 就绪后再次置 1。在 OSCFRQ 更改尚未生效期间，HFINTOSC 会在高电平或低电平状态停滞，直到该振荡器锁定新频率并恢复运行。

振荡器使能（OSCEN）寄存器可用于手动使能以下振荡器：

- HFINTOSC
- MFINTOSC
- LFINTOSC
- ADCRC

9.3. 寄存器定义：振荡器控制

9.3.1. OSCCON

名称： OSCCON
偏移量： 0x88E

振荡器控制寄存器

位	7	6	5	4	3	2	1	0
			COSC[1:0]					
访问			R	R				
复位			q	q				

Bit 5:4 – COSC[1:0] 当前振荡器源选择
按照 RSTOSC 配置位指示当前振荡器源

9.3.2. OSCSTAT

名称： OSCSTAT
偏移量： 0x890

振荡器状态寄存器

位	7	6	5	4	3	2	1	0
		HFOR	MFOR	LFOR		ADOR	SFOR	
访问		R	R	R		R	R	
复位		0	0	0		0	0	

Bit 6 – HFOR HFINTOSC 就绪

值	说明
1	HFINTOSC 就绪
0	HFINTOSC 未使能，或尚未就绪

Bit 5 – MFOR MFINTOSC 就绪

值	说明
1	MFINTOSC 就绪
0	MFINTOSC 未使能，或尚未就绪

Bit 4 – LFOR LFINTOSC 就绪

值	说明
1	LFINTOSC 就绪
0	LFINTOSC 未使能，或尚未就绪备用

Bit 2 – ADOR ADCRC 振荡器就绪

值	说明
1	ADCRC 振荡器就绪
0	ADCRC 振荡器未使能，或尚未就绪备用

Bit 1 – SFOR 指定频率振荡器就绪

值	说明
1	SFINTOSC 就绪
0	SFINTOSC 尚未就绪备用

9.3.3. OSCEN

名称： OSCEN
偏移量： 0x891

振荡器使能寄存器

位	7	6	5	4	3	2	1	0
		HFOEN	MFOEN	LFOEN		ADOEN		
访问		R/W	R/W	R/W		R/W		
复位		0	0	0		0		

Bit 6 – HFOEN HFINTOSC 使能

值	说明
1	已明确使能 HFINTOSC，具体操作由 OSCFRQ 寄存器指定
0	可通过外设请求使能 HFINTOSC

Bit 5 – MFOEN MFINTOSC 使能

值	说明
1	已明确使能 MFINTOSC
0	可通过外设请求使能 MFINTOSC

Bit 4 – LFOEN LFINTOSC 使能

值	说明
1	已明确使能 LFINTOSC
0	可通过外设请求使能 LFINTOSC

Bit 2 – ADOEN ADCRC 振荡器使能

值	说明
1	已明确使能 ADCRC
0	可通过外设请求使能 ADCRC

9.3.4. OSCFRQ

名称： OSCFRQ
偏移量： 0x893

HFINTOSC 频率选择寄存器

位	7	6	5	4	3	2	1	0
						FRQ[2:0]		
访问						R/W	R/W	R/W
复位						0	0	0

Bit 2:0 – FRQ[2:0] HFINTOSC 频率选择

FRQ	标称频率（MHz）
111-110	保留
101	32
100	16
011	8
010	4
001	2
000	1

9.3.5. OSCTUNE

名称： OSCTUNE
偏移量： 0x892

HFINTOSC 频率调节寄存器

位	7	6	5	4	3	2	1	0
			TUN[5:0]					
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			0	0	0	0	0	0

Bit 5:0 – TUN[5:0] HFINTOSC 频率调节

TUN	条件
01 1111	最高频率
•	•
•	•
•	•
00 0000	中心频率。振荡器在所选标称频率下工作。（默认值）
•	•
•	•
•	•
10 0000	最低频率

9.4. 寄存器汇总——振荡器控制

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x088D	保留									
0x088E	OSCCON	7:0			COSC[1:0]					
0x088F	保留									
0x0890	OSCSTAT	7:0		HFOR	MFOR	LFOR		ADOR	SFOR	
0x0891	OSCEN	7:0		HFOEN	MFOEN	LFOEN		ADOEN		
0x0892	OSCTUNE	7:0			TUN[5:0]					
0x0893	OSCFRQ	7:0						FRQ[2:0]		

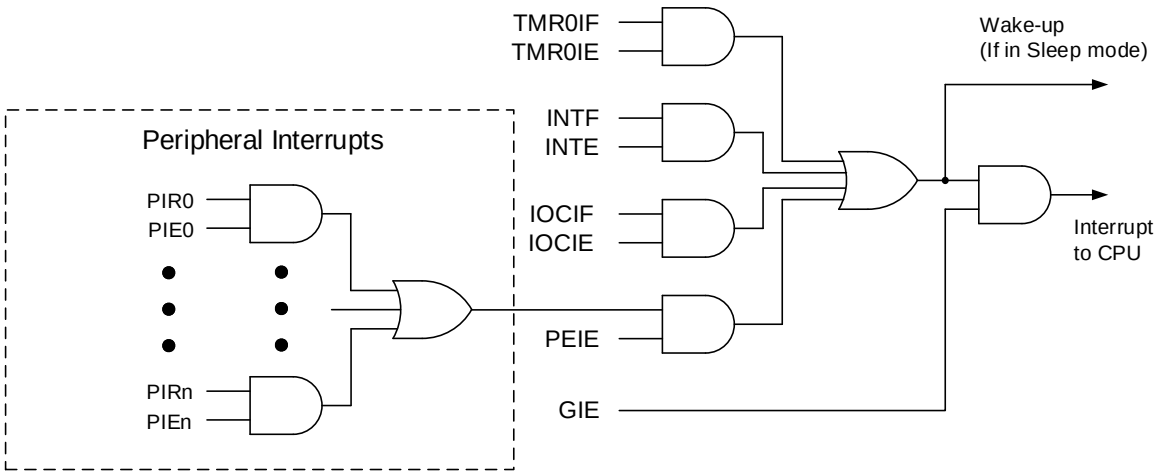
10. 中断

通过中断功能，一些事件可以抢占正常的程序流。固件用于确定中断源，并执行相应的操作。有些中断可配置为将 MCU 从休眠模式唤醒。

许多外设可以产生中断。详细信息请参见相应的章节。

图 10-1 给出了中断逻辑的框图。

图 10-1. 中断逻辑



10.1. INTCON 寄存器

中断控制（INTCON）寄存器是可读写寄存器，包含全局中断允许（GIE）位、外设中断允许（PEIE）位和外部中断边沿选择（INTEDG）位。

10.2. PIE 寄存器

外设中断允许（PIE）寄存器包含各外设中断的允许位。由于外设中断源众多，所以 CN5225 系列中共有 3PIE 寄存器。

10.3. PIR 寄存器

外设中断请求（PIR）寄存器包含各外设中断的标志位。由于外设中断源众多，所以共有 3PIR 寄存器。

10.4. 工作原理

任何器件复位都将禁止中断。可通过将以下位置 1 来允许中断：

- GIE 位
- PEIE 位（如果中断事件的中断允许位包含在 PIE 寄存器中）
- 特定中断事件的中断允许位

PIR 寄存器通过中断标志位记录各个中断。中断标志位是否置 1 与 GIE、PEIE 和各个中断允许位的状态无关。

当 GIE 位置 1 时，中断事件的发生会引发以下事件：

- 清除当前预取的指令

- GIE 位清零
- 当前程序计数器（PC）值被压入堆栈
- 重要寄存器的内容自动保存到影子寄存器（见自动现场保护）
- PC 装载中断向量 0004h

中断服务程序（ISR）中的固件可通过查询中断标志位来确定中断源。在退出 ISR 之前必须将中断标志位清零，以避免重复的中断。由于 GIE 位被清零，所以执行 ISR 期间发生的任何中断都将会通过其中断标志进行记录，但是不会使处理器重定向到中断向量。

RETFIE 指令会将先前的地址从堆栈中弹出，从影子寄存器恢复保存的内容，然后将 GIE 位置 1，以此退出 ISR。

如需了解关于特定中断操作的更多信息，请参见其相应的外设章节。

重要：

1. 各个中断标志位是否置 1 与任何其他中断允许位的状态无关。
2. 当 GIE 位清零时，忽略所有中断。GIE 位清零时发生的任何中断在 GIE 位再次置 1 后才会处理。

10.5. 中断延时

中断延时定义为从发生中断事件到开始执行中断向量处的代码所需的时间。中断在指令周期的 Q1 阶段期间采样。之后，实际的中断延时取决于检测到中断时执行的指令。更多详细信息，请参见下图。

图 10-2. 中断延时

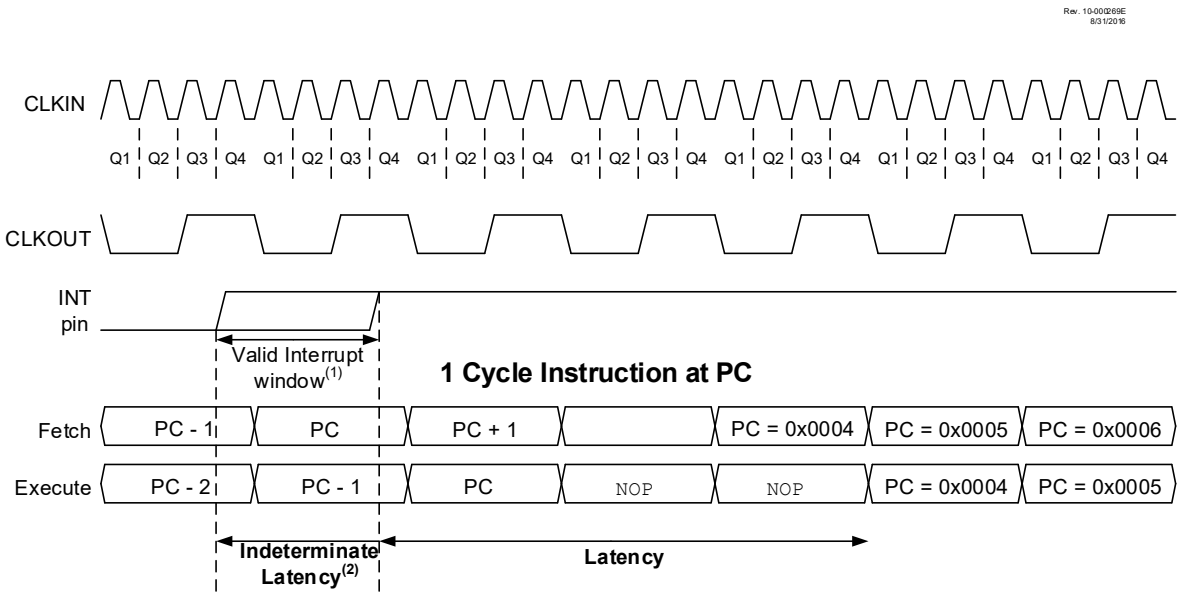
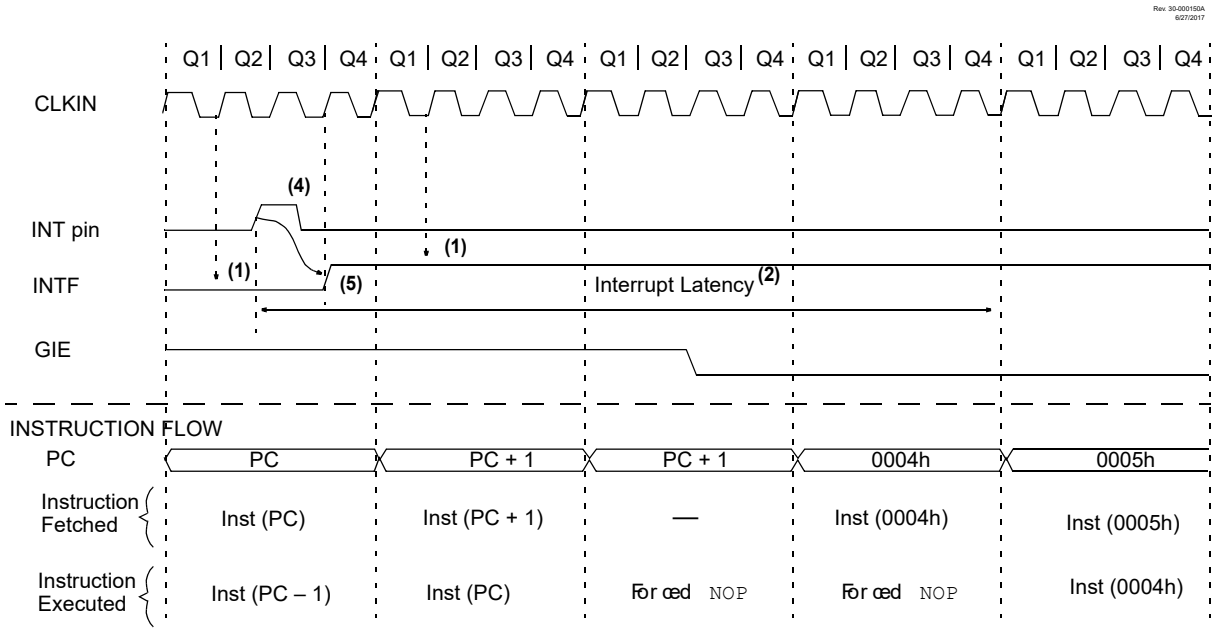


图 10-3. INT 引脚中断时序



注：

1. 在此采样 INTF 标志（每个 Q1 周期）。
2. 异步中断延时为 3-5 个 T_{CY}。同步中断延时为 3-4 个 T_{CY}，其中 T_{CY} = 指令周期时间。无论 Inst (PC)是单周期还是双周期指令，中断延时都是一样的。
3. 关于 INT 脉冲的最小宽度，请参见**“电气规范”**中的交流规范。
4. 允许在 Q4-Q1 周期内的任意时刻将 INTF 置 1。

10.6. 休眠期间的中断

中断可用于将器件从休眠模式唤醒。要从休眠模式唤醒器件，外设必须能在没有系统时钟的情况下工作。进入休眠模式前，中断源必须将相应的中断允许位置 1。

从休眠模式唤醒时，如果 GIE 位也置 1，则处理器将跳转到中断向量。否则，处理器将继续执行 SLEEP 指令后的指令。紧接 SLEEP 指令后的指令总是会在跳转到 ISR 前执行。

10.7. INT 引脚

INT 引脚可用于产生异步边沿触发中断。可以通过将外部中断允许（INTE）位置 1 来允许该中断。外部中断边沿选择（INTEDG）位确定中断在哪个边沿发生。INTEDG 位置 1 时，上升沿将引起中断。INTEDG 位清零时，下降沿将引起中断。外部中断标志（INTF）位将在 INT 引脚上出现有效边沿时置 1。如果 GIE 和 INTE 位也置 1，则处理器会将程序执行重定向到中断向量。

10.8. 自动现场保护

进入中断时，PC 的返回地址被保存在堆栈中。此外，以下寄存器会被自动保存到影子寄存器中：

- WREG 寄存器
- STATUS 寄存器（$\overline{TO}$ 和 $\overline{PD}$ 除外）
- BSR 寄存器
- FSR 寄存器
- PCLATH 寄存器

在退出中断服务程序时，将会自动恢复这些寄存器。在 ISR 期间对这些寄存器进行的任何修改都会丢失。如果需要修改其中的任意寄存器，则可修改相应的影子寄存器，该值在退出 ISR 时将会被恢复。影子寄存器位于 Bank 63 中，它们是可读写寄存器。根据用户的应用，可能还需要保存其他寄存器。

10.9. 寄存器定义：中断控制

10.9.1. INTCON

名称： INTCON
偏移量： 0x000B

中断控制寄存器

位	7	6	5	4	3	2	1	0
	GIE	PEIE						INTEDG
访问	R/W	R/W						R/W
复位	0	0						1

Bit 7 – GIE 全局中断允许

值	说明
1	允许所有有效中断
0	禁止所有中断

Bit 6 – PEIE 外设中断允许

值	说明
1	允许所有有效外设中断
0	禁止所有外设中断

Bit 0 – INTEDG 外部中断边沿选择

值	说明
1	INT 引脚的上升沿触发中断
0	INT 引脚的下降沿触发中断

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（GIE）位的状态如何，中断标志位都将置1。用户软件需确保先将相应的中断标志位清零，然后再允许中断。此功能允许软件轮询。

10.9.2. PIE0

名称： PIE0
偏移量： 0x716

外设中断允许寄存器 0

位	7	6	5	4	3	2	1	0
			TMR0IE	IOCIE				INTE
访问			R/W	R/W				R/W
复位			0	0				0

Bit 5 – TMR0IE Timer0 中断允许

值	说明
1	允许 TMR0 中断
0	禁止 TMR0 中断

Bit 4 – IOCIE 电平变化中断允许

值	说明
1	允许 IOC 中断
0	禁止 IOC 中断

Bit 0 – INTE 外部中断允许[(1)]

值	说明
1	允许外部中断
0	禁止外部中断

注：

1. 外部中断 INT 引脚由 INTPPS 选择。
2. 要允许任何一个由 PIE1 与 PIE2 寄存器控制的外设中断，必须将 INTCON 寄存器的 PEIE 位置 1。PIE0 寄存器控制的中断源不需要将 PEIE 位置 1 来允许中断向量（INTCON 寄存器中的 GIE 位置 1 时）。

10.9.3. PIE1

名称： PIE1
偏移量： 0x717

外设中断允许寄存器 1

位	7	6	5	4	3	2	1	0
	CCP1IE	TMR2IE	TMR1IE	RC1IE	TX1IE	BCL1IE	SSP1IE	ADIE
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7 – CCP1IE CCP1 中断允许

值	说明
1	允许 CCP1 中断
0	禁止 CCP1 中断

Bit 6 – TMR2IE TMR2 中断允许

值	说明
1	允许 TMR2 中断
0	禁止 TMR2 中断

Bit 5 – TMR1IE TMR1 中断允许

值	说明
1	允许 TMR1 中断
0	禁止 TMR1 中断

Bit 4 – RC1IE EUSART1 接收中断允许

值	说明
1	允许 EUSART1 接收中断
0	禁止 EUSART1 接收中断

Bit 3 – TX1IE EUSART1 发送中断允许

值	说明
1	允许 EUSART1 发送中断
0	禁止 EUSART1 发送中断

Bit 2 – BCL1IE MSSP1 总线冲突中断允许

值	说明
1	允许 MSSP1 总线冲突中断
0	禁止 MSSP1 总线冲突中断

Bit 1 – SSP1IE MSSP1 中断允许

值	说明
1	允许 MSSP1 中断
0	禁止 MSSP1 中断

Bit 0 – ADIE ADC 中断允许

值	说明
1	允许 ADC 中断
0	禁止 ADC 中断

注：要允许任何一个由寄存器 PIE1 和 PIE2 控制的外设中断，必须将 INTCON 寄存器的 PEIE 位置 1。

10.9.4. PIE2

名称： PIE2
偏移量： 0x718

外设中断允许寄存器 2

位	7	6	5	4	3	2	1	0
	CCP2IE	NVMIE	TMR1GIE					
访问	R/W	R/W	R/W					
复位	0	0	0					

Bit 7 – CCP2IE CCP2 中断允许

值	说明
1	允许 CCP2 中断
0	禁止 CCP2 中断

Bit 6 – NVMIE NVM 中断允许

值	说明
1	允许 NVM 中断
0	禁止 NVM 中断

Bit 5 – TMR1GIE TMR1 门控中断允许

值	说明
1	允许 TMR1 门控中断
0	禁止 TMR1 门控中断

注：要允许任何一个由 PIE1 和 PIE2 寄存器控制的外设中断，必须将 INTCON 寄存器的 PEIE 位置 1。

10.9.5. PIR0

名称： PIR0
偏移量： 0x70C

外设中断请求寄存器 0

位	7	6	5	4	3	2	1	0
			TMR0IF	IOCIF				INTF
访问			R/W/HS	R				R/W/HS
复位			0	0				0

Bit 5 – TMR0IF Timer0 中断标志

值	说明
1	TMR0 寄存器已溢出（必须用软件清零）
0	TMR0 寄存器未溢出

Bit 4 – IOCIF 电平变化中断标志[(2)]

值	说明
1	当前有一个或多个 IOCAF-IOCCF 寄存器位置 1，表示 IOC 模块检测到使能的边沿。
0	当前没有任何 IOCAF-IOCCF 寄存器位置 1

Bit 0 – INTF 外部中断标志[(1)]

值	说明
1	发生了外部中断
0	未发生外部中断

注：

1. 外部中断 INT 引脚由 INTPPS 选择。
2. IOCIF 位是所有 IOCAF-IOCCF 标志的逻辑或结果。因此，要清零 IOCIF 标志，应用程序固件必须清零所有低电平 IOCAF-IOCCF 寄存器位。
3. 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（GIE）位的状态如何，中断标志位都将置 1。用户软件应确保先将相应的中断标志位清零，然后再允许中断。

10.9.6. PIR1

名称： PIR1
偏移量： 0x70D

外设中断请求寄存器 1

位	7	6	5	4	3	2	1	0
	CCP1IF	TMR2IF	TMR1IF	RC1IF	TX1IF	BCL1IF	SSP1IF	ADIF
访问	R/W/HS	R/W/HS	R/W/HS	R	R	R/W/HS	R/W/HS	R/W/HS
复位	0	0	0	0	0	0	0	0

Bit 7 – CCP1IF CCP1 中断标志

值	CCP 模式		
	捕捉	比较	PWM
1	发生了捕捉（必须用软件清零）	发生了比较匹配（必须用软件清零）	出现了输出后沿（必须用软件清零）
0	未发生捕捉	未发生比较匹配	未出现输出后沿

Bit 6 – TMR2IF TMR2 中断标志

值	说明
1	发生了中断（必须用软件清零）
0	未发生中断

Bit 5 – TMR1IF TMR1 中断标志

值	说明
1	发生了中断（必须用软件清零）
0	未发生中断

Bit 4 – RC1IF EUSART1 接收中断标志(1)

值	说明
1	EUSART1 接收缓冲区（RC1REG）不为空（至少包含一个字节）
0	EUSART1 接收缓冲区为空

Bit 3 – TX1IF EUSART1 发送中断标志(2)

值	说明
1	EUSART1 发送缓冲区（TX1REG）为空
0	EUSART1 发送缓冲区不为空

Bit 2 – BCL1IF MSSP1 总线冲突中断标志

值	说明
1	检测到总线冲突（必须用软件清零）
0	未检测到总线冲突

Bit 1 – SSP1IF MSSP1 中断标志

值	说明
1	发生了中断（必须用软件清零）
0	未发生中断

Bit 0 – ADIF ADC 中断标志

值	说明
1	发生了中断（必须用软件清零）
0	未发生中断

注：

1. RC1IF 是只读位。用户软件必须读取 RC1REG 才能清零 RC1IF。
2. TX1IF 是只读位。用户软件必须装载 TX1REG 才能清零 TX1IF。TX1IF 标志不指示发送完成（而是使用 TRMT 进行指示）。
3. 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（GIE）位的状态如何，中断标志位都将置 1。用户软件应确保先将相应的中断标志位清零，然后再允许中断。

10.9.7. PIR2

名称： PIR2
偏移量： 0x70E

外设中断请求寄存器 2

位	7	6	5	4	3	2	1	0
	CCP2IF	NVMIF	TMR1GIF					
访问	R/W/HS	R/W/HS	R/W/HS					
复位	0	0	0					

Bit 7 – CCP2IF CCP2 中断标志

值	CCP 模式		
	捕捉	比较	PWM
1	发生了捕捉（必须用软件清零）	发生了比较匹配（必须用软件清零）	出现了输出后沿（必须用软件清零）
0	未发生捕捉	未发生比较匹配	未出现输出后沿

Bit 6 – NVMIF 非易失性存储器（NVM）中断标志

值	说明
1	请求的 NVM 操作已完成（必须用软件清零）
0	未发生中断

Bit 5 – TMR1GIF TMR1 门控中断标志

值	说明
1	TMR1 门控已变为无效（必须用软件清零）
0	TMR1 门控有效

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（GIE）位的状态如何，中断标志位都将置 1。用户软件应确保先将相应的中断标志位清零，然后再允许中断。

10.10. 寄存器汇总——中断控制

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x070B	保留									
0x070C	PIR0	7:0			TMR0IF	IOCIF				INTF
0x070D	PIR1	7:0	CCP1IF	TMR2IF	TMR1IF	RC1IF	TX1IF	BCL1IF	SSP1IF	ADIF
0x070E	PIR2	7:0	CCP2IF	NVMIF	TMR1GIF					
0x070F ... 0x0715	保留									
0x0716	PIE0	7:0			TMR0IE	IOCIE				INTE
0x0717	PIE1	7:0	CCP1IE	TMR2IE	TMR1IE	RC1IE	TX1IE	BCL1IE	SSP1IE	ADIE
0x0718	PIE2	7:0	CCP2IE	NVMIE	TMR1GIE					

11. 休眠模式

11.1. 休眠模式操作

器件通过执行 SLEEP 指令进入休眠模式。

进入休眠模式时，存在以下条件：

1. WDT 之外的复位不受休眠模式影响；WDT 将清零但是保持运行（如果使能了在休眠期间工作）。
2. $\overline{PD}$ 位清零。
3. $\overline{TO}$ 位置 1。
4. CPU 和系统时钟处于禁止状态。
5. 如果任何外设请求 LFINTOSC 和/或 HFINTOSC 作为时钟源或者 HFOEN、MFOEN 或 LFOEN 位置 1，则 LFINTOSC 和/或 HFINTOSC 将保持使能状态。
6. 如果选择了 ADCRC 振荡器，则 ADC 不受影响。ADC 时钟不是 ADCRC 时，尽管 ADON 位仍保持有效，但 SLEEP 指令会导致当前转换中止，ADC 模块被关闭。
7. 仅当 I/O 端口连接的外设无效时，I/O 端口才会保持执行 SLEEP 指令之前的状态（驱动为高电平、低电平或高阻态）。

关于外设在休眠期间工作的更多详细信息，请参见各个章节。

要最大程度地降低电流消耗，需要考虑以下条件：

- I/O 引脚不应悬空
- 来自 I/O 引脚的外部电路灌电流
- 来自 I/O 引脚的内部电路拉电流
- 从带内部弱上拉的引脚汲取的电流
- 使用任何振荡器的模块

为了避免输入引脚悬空而引入开关电流，应在外部将为高阻抗输入的 I/O 引脚拉到 V_{DD} 或 V_{SS}。

11.1.1. 从休眠模式唤醒

发生以下任一事件会将器件从休眠模式唤醒：

1. $\overline{MCLR}$ 引脚上的外部复位输入（如果使能）。
2. BOR 复位（如果使能）。
3. POR 复位。
4. 看门狗定时器（如果使能）。
5. 任何外部中断。
6. 能够在休眠期间运行的外设产生的中断（更多信息，请参见各个外设）。

前三个事件会导致器件复位。后三个事件视为继续执行程序。要确定是发生了器件复位还是唤醒事件，请参见“**复位**”一章中的**“确定复位原因”**一节。

当执行 SLEEP 指令时，下一条指令（PC + 1）被预取出。如果希望通过中断事件唤醒器件，则必须允许相应的中断允许位。唤醒与 GIE 位的状态无关。如果 GIE 位清零，器件将继续执行 SLEEP 指令后的指令。如果 GIE 位置 1，器件先执行 SLEEP 指令后的指令，然后将调用中断服务程序。如果不希望执行 SLEEP 指令之后的指令，用户应在 SLEEP 指令后面放置一条 NOP 指令。

器件从休眠模式唤醒时，WDT 将被清零，而与唤醒源无关。

11.1.2. 使用中断唤醒

当禁止全局中断（GIE 被清零）并且有任一中断源将其中断允许位和中断标志位置 1 时，将会发生下列某一事件：

- 如果在执行 SLEEP 指令之前发生中断
  - SLEEP 指令将作为 NOP 执行
  - WDT 和 WDT 预分频器不会清零
  - $\overline{TO}$ 位不会置 1
  - $\overline{PD}$ 位不会清零
- 如果在执行 SLEEP 指令期间或之后发生中断：
  - 将完整执行 SLEEP 指令
  - 器件将立即从休眠模式唤醒
  - WDT 和 WDT 预分频器将清零
  - $\overline{TO}$ 位将置 1
  - $\overline{PD}$ 位将清零

即使在执行 SLEEP 指令之前，检查到标志位为 0，这些标志位也有可能在 SLEEP 指令执行完毕之前被置 1。要确定是否执行了 SLEEP 指令，可测试 $\overline{PD}$ 位。如果 $\overline{PD}$ 位置 1，则说明 SLEEP 指令作为 NOP 指令执行了。

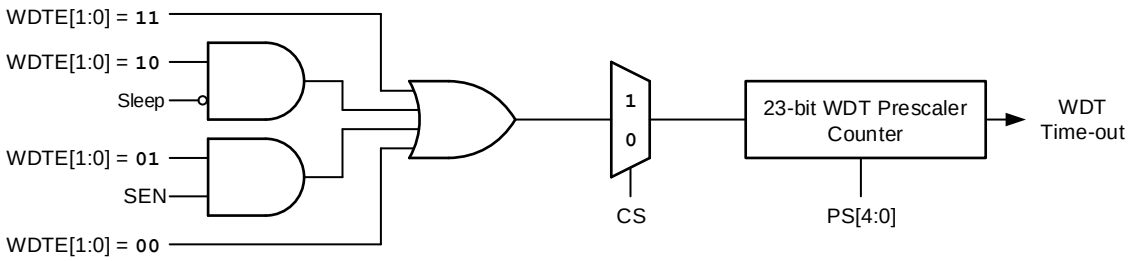
12. WDT——看门狗定时器

看门狗定时器（WDT）是一个系统定时器，如果固件未在超时周期内发出 CLRWDT 指令，看门狗定时器会产生复位事件。看门狗定时器通常用于在发生软件故障时复位处理器，但也可用于将器件从休眠模式唤醒。

WDT 具有以下特性：

- 可选的时钟源
- 多种工作模式：
  - WDT 始终使能
  - WDT 在休眠时禁止
  - WDT 由软件控制
  - WDT 始终禁止
- 可配置的超时周期为 1 ms 至 256s（标称值）
- 多个复位条件
- 可在休眠期间工作

图 12-1. WDT 框图



12.1. 可选的时钟源

WDT 可从 31.25 kHz MFINTOSC 或 31 kHz LFINTOSC（由 WDT 时钟源选择（CS）位选择）获得其时基。

重要：本节详细说明的时间间隔均基于由 LFINTOSC 时钟源产生的 1 ms 最小标称时间间隔。

12.2. WDT 工作模式

WDT 模块具有 4 种工作模式，这些工作模式由看门狗定时器使能（WDTE）位控制。请参见表 12-1。

表 12-1. WDT 工作模式

WDTE[1:0]	SEN	器件模式	WDT 模式
11	x	X	有效
10	x	唤醒	有效
		休眠	禁止

表 12-1. WDT 工作模式 （续）

WDTE[1:0]	SEN	器件模式	WDT 模式
01	1	X	有效
	0	X	禁止
00	x	X	禁止

12.2.1. WDT 始终使能

当 WDTE 位设置为 11 时，WDT 始终使能。WDT 保护功能在休眠模式期间有效。

12.2.2. WDT 在休眠模式下禁止

当 WDTE 位设置为 10 时，除非处于休眠模式，否则 WDT 将使能。在休眠模式下，WDT 保护禁止。

12.2.3. WDT 由软件控制

当 WDTE 位设置为 01 时，WDT 将通过软件看门狗定时器使能（SEN）位进行控制。当 SEN 置 1（SEN = 1）时，WDT 保护有效。当 SEN 清零（SEN = 0）时，WDT 保护禁止。

12.2.4. WDT 禁止

当 WDTE 位设置为 00 时，WDT 禁止。在该模式下，SEN 位被忽略。

12.3. WDT 超时周期

看门狗定时器预分频比选择（PS）位用于设置从 1 ms 至 256s（标称值）的超时周期。在复位之后，默认的超时周期为 2 秒。

12.4. 清零 WDT

当发生以下任何条件时，WDT 被清零：

- 任何复位
- 执行了有效的 CLRWDT 指令
- 器件进入休眠模式
- 器件从休眠模式唤醒
- 对 WDTCON 寄存器的任何写操作

12.5. 休眠期间的 WDT 操作

当 WDT 进入休眠模式时，WDT 会被清零。如果使能 WDT 在休眠期间工作，WDT 会继续计数。当 WDT 退出休眠模式时，WDT 会被再次清零。

在器件处于休眠模式的情况下发生 WDT 超时时，不会产生复位。器件将会唤醒并继续工作。超时（$\overline{TO}$）和掉电（$\overline{PD}$）位将清零以指示事件。此外，看门狗定时器复位标志（$\overline{RWDT}$）位也将清零，指示发生了 WDT 复位事件。

12.6. 寄存器定义：WDT 控制

12.6.1. WDTCON

名称： WDTCON
偏移量： 0x80C

看门狗定时器控制寄存器

位	7	6	5	4	3	2	1	0
	CS		PS[4:0]					SEN
访问	R/W		R/W	R/W	R/W	R/W	R/W	R/W
复位	0		0	0	0	0	0	0

Bit 7 – CS 看门狗定时器时钟源选择

值	说明
1	MFINTOSC（31.25 kHz）
0	LFINTOSC（31 kHz）

Bit 5:1 – PS[4:0] 看门狗定时器预分频比选择(1)

值	说明
11111 – 10011	保留。产生最小的时间间隔（1:32）
10010	1:8388608（时间间隔标称值为 256s）
10001	1:4194304（时间间隔标称值为 128s）
10000	1:2097152（时间间隔标称值为 64s）
01111	1:1048576（时间间隔标称值为 32s）
01110	1:524288（时间间隔标称值为 16s）
01101	1:262144（时间间隔标称值为 8s）
01100	1:131072（时间间隔标称值为 4s）
01011	1:65536（时间间隔标称值为 2s）（复位值）
01010	1:32768（时间间隔标称值为 1s）
01001	1:16384（时间间隔标称值为 512 ms）
01000	1:8192（时间间隔标称值为 256 ms）
00111	1:4096（时间间隔标称值为 128 ms）
00110	1:2048（时间间隔标称值为 64 ms）
00101	1:1024（时间间隔标称值为 32 ms）
00100	1:512（时间间隔标称值为 16 ms）
00011	1:256（时间间隔标称值为 8 ms）
00010	1:128（时间间隔标称值为 4 ms）
00001	1:64（时间间隔标称值为 2 ms）
00000	1:32（时间间隔标称值为 1 ms）

Bit 0 – SEN 软件 WDT 使能/禁止

值	条件	说明
x	如果 WDTE[1:0] ≠01	忽略此位
1	如果 WDTE[1:0] =01	使能 WDT
0	如果 WDTE[1:0] =01	禁止 WDT

注：

1. 时间均为近似值，基于 31 kHz LFINTOSC 时钟源。

12.7. 寄存器汇总——WDT 控制

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x080B	保留									
0x080C	WDTCON	7:0	CS		PS[4:0]					SEN

13. NVM——非易失性存储器控制

非易失性存储器（Nonvolatile Memory，NVM）模块提供对闪存程序存储器（PFM）和配置位的运行时读/写访问。PFM 包括程序存储器和用户 ID 空间。

NVM 可使用 FSR 和 INDF 寄存器访问或通过 NVMREG 寄存器接口进行访问（见表 13-1）。

写入时间由片上定时器控制。写入/擦除电压由片上电荷泵产生，此电荷泵在器件的工作电压范围内工作。

PFM 可通过两种方式进行保护：代码保护和写保护。代码保护（配置位 $\overline{CP}$）通过外部器件编程器禁止 PFM 读访问和写访问。写保护可防止用户软件写入标记为受 $\overline{WRTn}$ 配置位保护的 NVM 区域。代码保护不会影响自写和擦除功能，而写保护则会产生影响。尝试写入受保护的存储单元会将 WRERR 位置 1。代码保护和写保护只能通过外部编程器执行的批量擦除来复位。

批量擦除命令用于完全擦除程序存储器。批量擦除命令只能通过外部编程器发出。运行时无法访问该命令。

如果器件受到代码保护，当发出针对配置存储器的批量擦除命令时，所有其他存储器区域也将一并擦除。更多详细信息，请参见**“系列编程规范”**。

表 13-1. NVM 构成和访问信息

主要值			NVMREG 访问			FSR 访问	
存储器功能	存储器类型	程序计数器（PC）地址	NVMREGS 位（NVMCON1）	NVMADR[14:0]	允许的操作	FSR 地址	FSR 编程访问
复位向量	闪存程序存储器	0x0000	0	0x0000	读/写	0x8000	只读
用户存储器		0x0001	0	0x0001		0x8001	
		0x0003		0x0003		0x8003	
INT 向量		0x0004	0	0x0004		0x8004	
用户存储器		0x0005	0	0x0005		0x8005	
		0x3FFF(1)		0x3FFF(1)		0xFFFF	
用户 ID	闪存程序存储器	0x8000	1	0x0000	读/写	无访问	
		0x8003		0x0003			
保留	—	—	—	0x0004	—		
版本 ID	在闪存程序存储器中硬编码	0x8005	1	0x0005	读		
器件 ID		0x8006	1	0x0006			
CONFIG1	闪存程序存储器	0x8007	1	0x0007	读/写		
CONFIG2		0x8008	1	0x0008			
CONFIG3		0x8009	1	0x0009			
CONFIG4		0x800A	1	0x000A			
CONFIG5		0x800B	1	0x000B			
DIA 和 DCI	在闪存程序存储器中硬编码	0x8100	1	0x0100	读		
		0x82FF	1	0x02FF			

注：

1. CN5225 系列的最大闪存程序存储器地址是 0x1FFF。

13.1. 闪存程序存储器（PFM）

在整个 V_{DD} 范围内的正常工作期间，闪存程序存储器（PFM）可读写且可擦除。

PFM 包含以下区域：

- 用户程序存储区（读/写）
- 配置字（读/写）
- 器件 ID（只读）
- 版本 ID（只读）

- 用户 ID（读/写）
- 器件信息区（只读）
- 器件配置信息（只读）

PFM 可通过以下方式读和/或写：

- CPU 取指（只读）
- FSR/INDF 间接访问（只读）
- NVMREG 访问（读/写）

要进行擦除和编程操作，了解程序存储器结构非常重要。程序存储器按行排列。每一行都包含 32 个 14 位程序存储字。行是可以通过用户软件擦除的最小大小。无法从用户代码发出批量擦除命令。

读操作返回存储器的单个字。写操作和擦除操作以行为单位。程序存储器擦除后为逻辑 1，编程后为逻辑 0。

可以编程整行或一行中的部分内容。写入程序存储器行的数据将被写入 14 位宽的数据写锁存器中。用户不能直接访问这些锁存器，但是可以通过对 NVMDATH:NVMDATL 寄存器对的连续写入来加载写锁存器的内容。

重要：如果只修改先前已编程行的一部分内容，则必须读取整行内容。然后，可以将新数据和保留的数据写入写锁存器，以对程序存储器行进行再编程。但如果是未经过编程的单元，则无需先擦除行即可写入。这种情况下，不需要保存并重新写入其他先前已编程的单元。

写入或擦除程序存储器将停止获取指令，直到操作完成。在写入或擦除期间无法访问程序存储器，因此代码无法执行。内部编程定时器用于控制程序存储器写操作和擦除操作的写入时间。

写入程序存储器的值无需是有效指令。执行形成无效指令的程序存储单元会导致 NOP。

13.1.1. FSR 和 INDF 访问

文件选择寄存器（FSR）和 INDF 寄存器允许间接访问闪存程序存储器。间接寻址是一种通过另一个寄存器来确定指令中存储器地址的模式。FSR 寄存器的值用于确定要访问的存储器地址单元。

13.1.1.1. FSR 读

FSR 用于提供对程序存储器的读访问。

通过将要读取的地址装入 FSRxH:FSRxL 寄存器对并将 FSRxH 寄存器的 bit 7 置 1 来访问程序存储器。执行 MOVIW 指令或任何访问 INDFx 的指令时，装入 FSRx 寄存器对中的值指向要访问的程序存储器中的存储单元。如果 FSRx 寄存器对指向 INDFx 寄存器，则读操作将返回 0。

读 NVM 操作需要一个指令周期。CPU 操作在读操作期间暂停，并在完成之后立即恢复。读操作返回存储器的单个字节。

13.1.1.2. FSR 写

CN5225 单片机系列中不支持通过 FSR 寄存器写/擦除 NVM（例如 MOVWI 指令）。

13.1.2. NVMREG 访问

NVMREG 接口允许对可通过 FSR 访问的所有存储单元以及用户 ID 存储单元和配置字进行读/写访问，而只能对器件 ID 和版本 ID 寄存器进行读访问。

当器件受写保护时，阻止通过 NVMREG 接口写或擦除 NVM。

13.1.2.1. NVMREG 读操作

要使用 NVMREG 接口读 NVM 存储单元，用户必须：

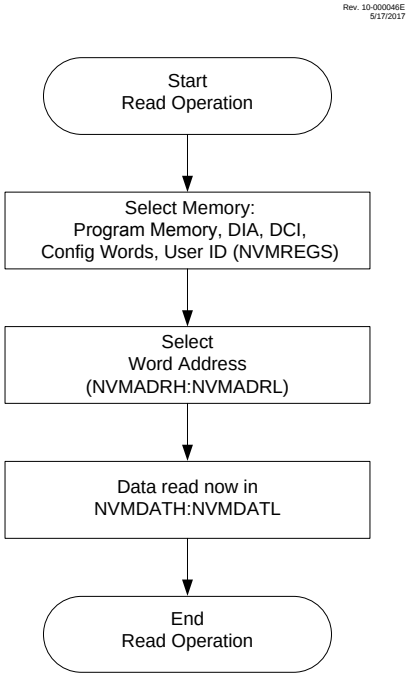
1. 如果用户打算访问程序存储器存储单元，则清零 NVMREGS 位，或者如果用户打算访问用户 ID 或配置存储单元，则将 NMVREGS 置 1。
2. 将所需地址写入 NVMADRH:NVMADRL 寄存器对。
3. 将 RD 位置 1 以启动读操作。

将读控制位置 1 后，CPU 操作在读操作期间暂停，并在完成操作后立即恢复。在紧接着的下一个周期，数据出现在 NVMDATH:NVMDATL 寄存器对中；因此，可在随后的指令中读取为两个字节。

NVMDATH:NVMDATL 寄存器对将保存该值直到另一次读操作或用户写入新值为止。

完成时，RD 位由硬件清零。

图 13-1. 闪存程序存储器读序列



例 13-1. 程序存储器读操作

```
// This code block will read 1 word of program memory

NVMCON1bits.NVMREGS = 0;          // Point to PFM
NVMADR = PFM_ADDRESS;             // Load NVMADRH:NVMADRL with PFM address
NVMCON1bits.RD = 1;               // Initiate read cycle
PFM_DATA_LOW = NVMDATL;           // PFM data low byte
PFM_DATA_HIGH = NVMDATH;          // PFM data high byte
```

13.1.2.2. NVM 解锁序列

解锁序列是一种用于保护 NVM 免于发生意外自写编程或擦除的机制。只有在无中断情况下执行并完成序列时，才能成功地完成以下操作之一：

- PFM 行擦除
- 将 PFM 写锁存器内容写入 PFM 存储器
- 将 PFM 写锁存器内容写入用户 ID

- 写入到配置字

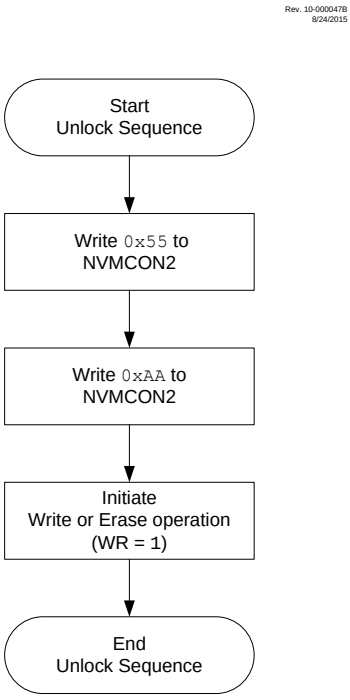
解锁序列由以下步骤组成且必须按照以下顺序完成：

- 将 55h 写入 NVMCON2
- 将 AAh 写入 NMVCON2
- 将 WR 位置 1

在 WR 位置 1 后，处理器会暂停内部操作，直到操作完成为止，然后再继续执行下一条指令。

由于在执行解锁序列的过程中不能发生中断，所以在执行解锁序列之前必须先禁止全局中断，然后在完成解锁序列之后重新允许。

图 13-2. NVM 解锁序列



例 13-2. NVM 解锁序列

```
NVMCON1bits.WREN = 1;           // Enable write/erase
INTCONbits.GIE = 0;             // Disable global interrupts

// The next three steps are the required unlock sequence
NVMCON2 = 0x55;                 // First unlock code
NVMCON2 = 0xAA;                 // Second unlock code
NVMCON1bits.WR = 1;             // Initiate write/erase cycle

INTCONbits.GIE = 1;             // Enable global interrupts
NVMCON1bits.WREN = 0;           // Disable further write/erase cycles
```

注：解锁序列开始于写 NVMCON2；必须按所示的精确周期顺序执行三个解锁步骤。如果中断或调试器暂停导致序列的时序被破坏，则不会执行该操作。

13.1.2.3. NVMREG 擦除程序存储器

在写入程序存储器之前，要写入的字必须已擦除或先前未写入。程序存储器每次只能擦除一行。在启动对程序存储器进行写操作时，并不会发生自动擦除操作。要擦除程序存储器行：

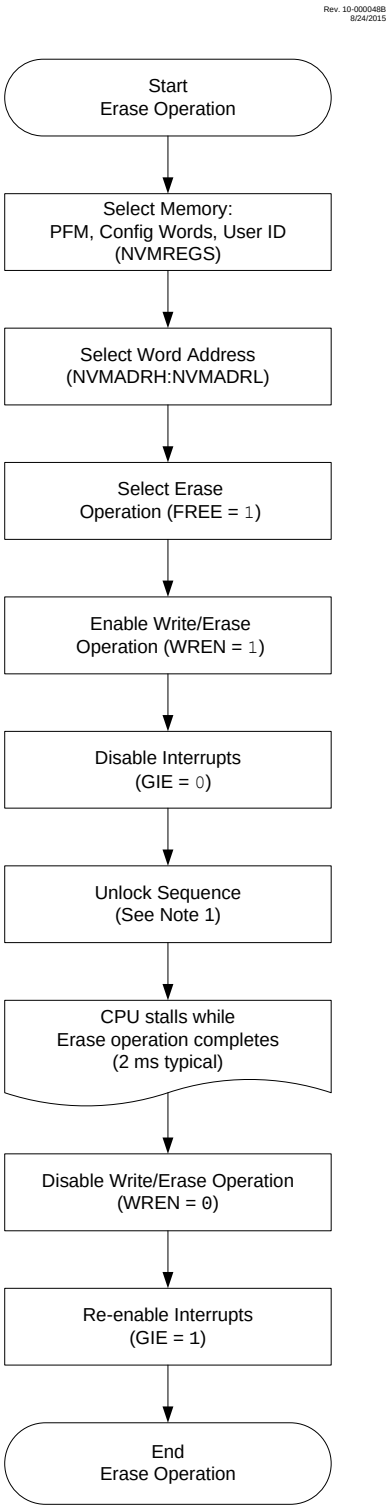
1. 清零 NVMREGS 位以擦除程序存储器存储单元或者将 NMVREGS 位置 1 以擦除用户 ID 单元。
2. 将所需地址写入 NVMADRH:NVMADRL 寄存器对。
3. 将 FREE 和 WREN 位置 1。
4. 按照 NVM 解锁序列一节所述执行解锁序列。

如果程序存储器地址受写保护，则清零 WR 位且不会执行擦除操作。

擦除程序存储器时，CPU 操作暂停，并在操作完成时恢复。操作完成时，NVMIF 位置 1，并且在 NVMIE 位也置 1 时将发生中断。

写锁存器数据不受擦除操作影响，且 WREN 将保持不变。

图 13-3. NVM 擦除序列



注：

1. 请参见 NVM 解锁序列部分

例 13-3. 擦除闪存程序存储器的一行

```
NVMCON1bits.NVMREGS = 0;          // Point to PFM
NVMADR = PFM_ADD;                 // 14-bit PFM address
NVMCON1bits.FREE = 1;             // Specify an erase operation
NVMCON1bits.WREN = 1;             // Enable write/erase cycle
INTCONbits.GIE = 0;               // Disable interrupts during unlock sequence

//The next three steps are the required unlock sequence
NVMCON2 = 0x55;                   // First unlock code
NVMCON2 = 0xAA;                   // Second unlock code
NVMCON1bits.WR = 1;               // Initiate write/erase cycle

INTCONbits.GIE = 1;               // Enable interrupts
NVMCON1bits.WREN = 1;             // Disable writes
```

13.1.2.4. NVMREG 写入程序存储器

使用以下步骤编程程序存储器：

1. 将要编程的行的地址装入 NVMADRH:NVMADRL。
2. 通过 NMVDATH:NVMDATL 寄存器向每个写锁存器中装入数据。
3. 启动编程操作。
4. 重复第 1 至 3 步，直到写入所有数据。

在写入程序存储器之前，要写入的字必须已擦除或先前未写入。程序存储器每次只能擦除一行。在启动写操作时，并不会发生自动擦除操作。

程序存储器每次可以写入一个或多个字。每次可以写入的最多字数等于写锁存器的数量。更多详细信息，请参见图 13-4。

写锁存器将对齐到由 NVMADRH:NVMADRL 高 10 位（NVMADRH[6:0]:NVMADRL[7:5]）定义的闪存行地址边界处，NVMADRL 的低 5 位（NVMADRL[4:0]）将决定要装入的写锁存器。写操作不会跨越这些边界。在程序存储器写操作完成时，写锁存器中的数据会复位为包含 0x3FFF。

要装入写锁存器并对程序存储器的一行进行编程，必须完成以下步骤。这些步骤分为两个部分。首先，在 LWLO = 1 的情况下，使用解锁序列将来自 NVMDATH:NVMDATL 的数据装入每个写锁存器。当要装入写锁存器的最后一个字就绪时，清零 LWLO 位并执行解锁序列。这将启动编程操作，将所有锁存器内容写入闪存程序存储器。

重要：要向写锁存器装入数据或启动闪存编程操作，需要执行一个特殊的解锁序列。如果在执行解锁序列的过程中发生中断，则不会启动对锁存器或程序存储器的写操作。

1. 将 WREN 位置 1。
2. 清零 NVMREGS 位。
3. 将 LWLO 位置 1。当 LWLO 位置 1（LWLO = 1）时，写操作之后只会向写锁存器装入数据，而不会启动对闪存程序存储器的写操作。
4. 将要写入的存储单元的地址装入 NVMADRH:NVMADRL 寄存器对。
5. 将要写入的程序存储器数据装入 NVMDATH:NVMDATL 寄存器对。
6. 执行解锁序列。此时，将数据装入写锁存器。
7. 递增 NVMADRH:NVMADRL 寄存器对，使之指向下一个存储单元。
8. 重复第 5 至 7 步，直到除最后一个写锁存器以外的所有写锁存器都已装载完毕为止。

9. 清零 LWLO 位。当 LWLO 位清零（LWLO = 0）时，写序列会启动对闪存程序存储器的写操作。
10. 将要写入的程序存储器数据装入 NVMDATH:NVMDATL 寄存器对。
11. 执行解锁序列。整个程序存储器锁存器的内容现在会被写入闪存程序存储器中。

重要：在每个写操作或擦除操作完成时，程序存储器写锁存器将复位为空白状态（0x3FFF）。因此，不需要向所有程序存储器写锁存器中装入数据。未装入的锁存器将保持空白状态。

例 13-4 给出了一个完整写序列的示例。初始地址装入 NVMADRH:NVMADRL 寄存器对；数据使用间接寻址方式装入。

图 13-4. NVMREG 写入带 32 个写锁存器的闪存程序存储器

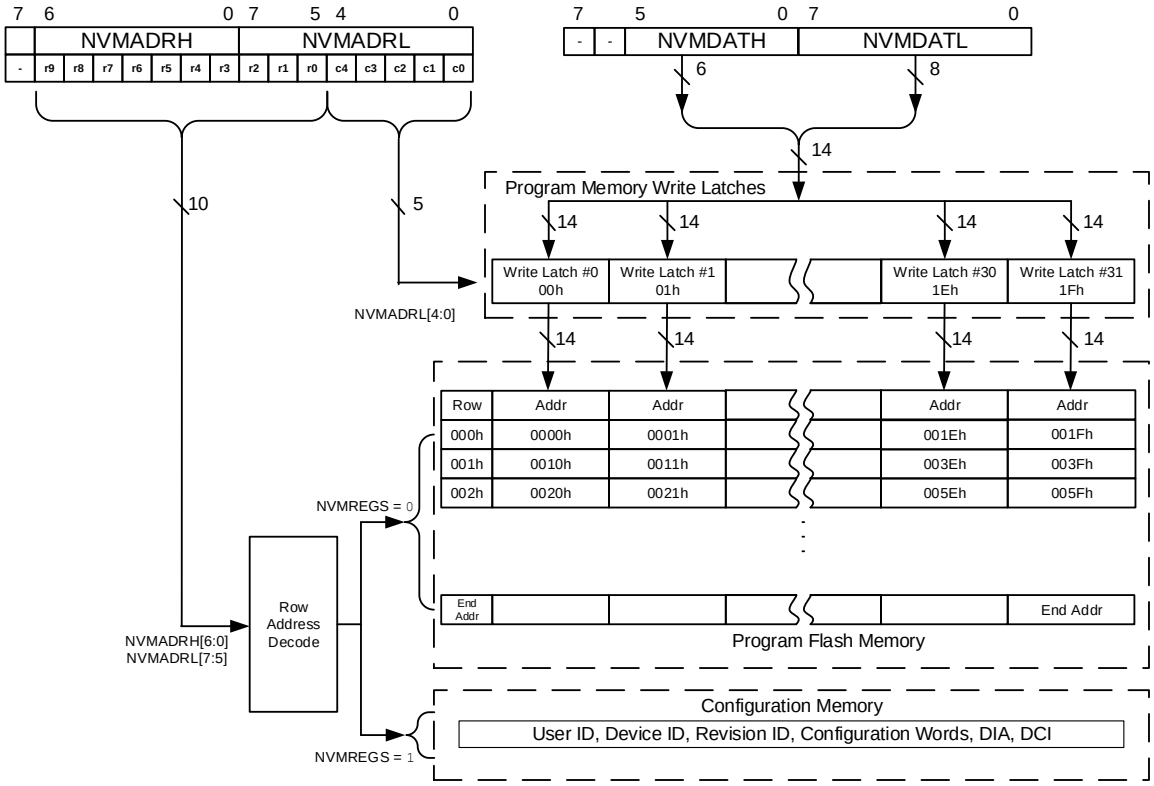
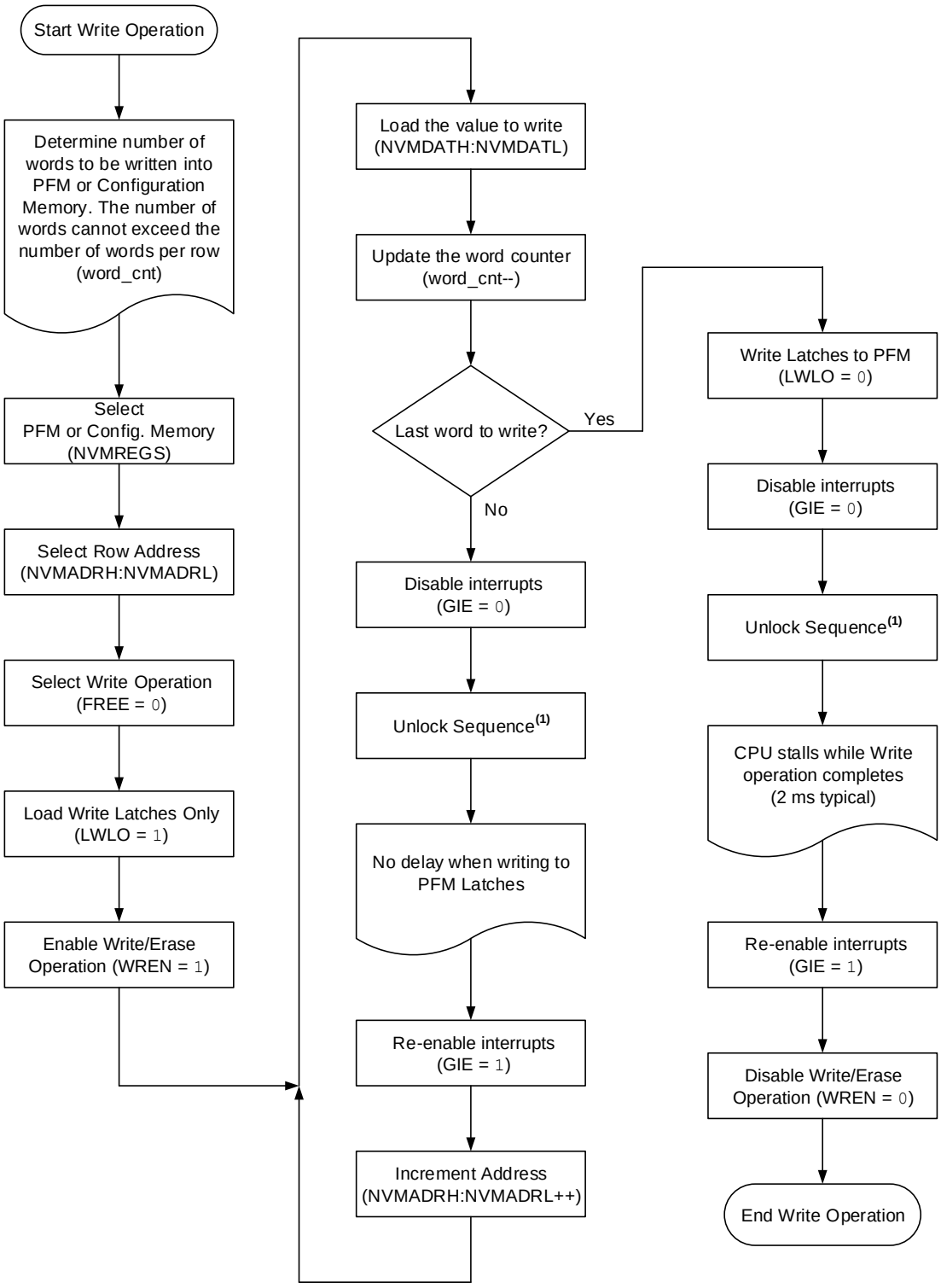


图 13-5. 闪存程序存储器写序列



注：

1. 请参见 NVM 解锁序列部分

例 13-4. 写入闪存程序存储器

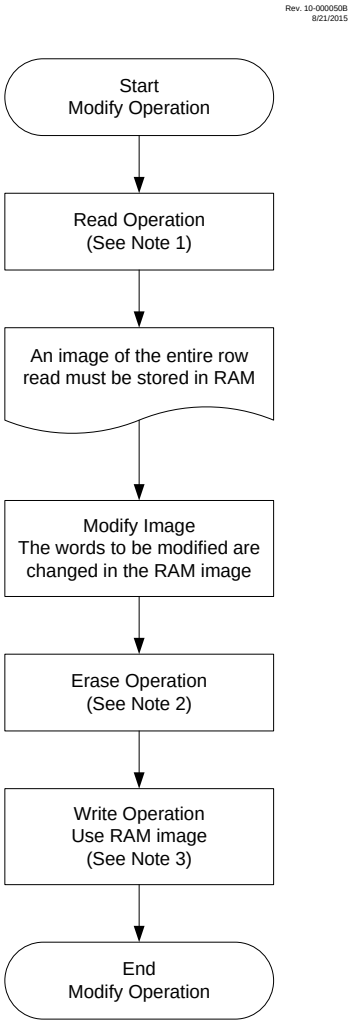
```
INTCONbits.GIE = 0;                 // Disable interrupts

// PFM row must be erased before writes can occur
NVMCON1bits.NVMREGS = 0;            // Point to PFM
NVMADR = PFMStartAddress;           // Must start at beginning of PFM row
NVMCON1bits.FREE = 1;               // Specify an erase operation
NVMCON1bits.WREN = 1;               // Allow erase cycle

// Required unlock sequence
NVMCON2 = 0x55;
NVMCON2 = 0xAA;
NVMCON1bits.WR = 1;

NVMCON1bits.LWLO = 1;               // Load write latches

// Write to the data latches
for (i = 0; i < PFM_ROW_SIZE; i++)
{
    NVMADR = PFMStartAddress;       // Load starting address
    NVMDAT = PFM_WRITE_DATA;        // Load data

    // Required unlock sequence
    NVMCON2 = 0x55;
    NVMCON2 = 0xAA;
    NVMCON1bits.WR = 1;

    PFMStartAddress++;              // Increment address
    if(i = (PFM_ROW_SIZE - 1))      // All latches loaded?
    {
        NVMCON1bits.LWLO = 0;       // Start PFM write
    }
}

NVMCON1bits.WREN = 0;               // Disable writes
INTCONbits.GIE = 1;                 // Enable interrupts
```

13.1.2.5. 修改闪存程序存储器

当要修改程序存储器中的已有数据时，必须读取存储器行中的数据并将其保存到 RAM 映像中。要修改程序存储器，请执行以下步骤：

1. 装入要修改的行的起始地址。
2. 从行中读取现有数据并将其保存到 RAM 映像中。
3. 修改 RAM 映像以包含要写入到程序存储器的新数据。
4. 装入要重新写入的行的起始地址。
5. 擦除程序存储器行。
6. 将来自 RAM 映像的数据装入写锁存器。
7. 启动编程操作。

图 13-6. 闪存程序存储器修改序列



注：

1. 请参见图 13-1。
2. 请参见图 13-3。
3. 请参见图 13-5。

13.1.2.6. 对 DIA、DCI、用户 ID、器件 ID、版本 ID 和配置字进行 NVMREG 访问

NVMREGS 可用于访问以下存储器区域：

- 器件信息区（DIA）
- 器件配置信息（DCI）
- 用户 ID 区域
- 器件 ID 和版本 ID
- 配置字

将 NVMREGS 的值置 1 以访问这些区域。上面列出的存储器区域是 PC[15] = 1 时指向的区域，但并不是所有地址都引用有效数据。可能存在不同的读写访问权限。请参见下表。对下表列出的参数以外的地址进行读访问时，NVMDATH: NVMDATL 寄存器对将被清零，读回 0。

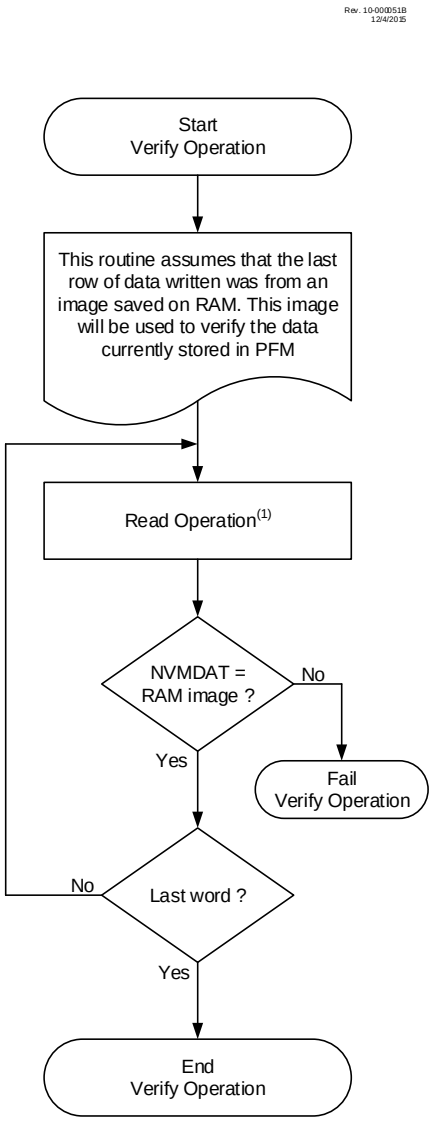
表 13-2. 对 DCI、用户 ID、器件 ID、版本 ID 和配置字进行 NVMREG 访问（NVMREGS = 1）

地址	功能	读访问	写访问
0x8000 - 0x8003	用户 ID	支持	支持
0x8005 - 0x8006	器件 ID/版本 ID	支持	不支持
0x8007 - 0x800B	配置字 1-5	支持	支持
0x8200 - 0x82FF	DCI	支持	不支持

13.1.2.7. 写校验

校验程序存储器写入数据是否与预期值一致是一种良好的编程习惯。由于程序存储器以整行形式存储，因此所存储的程序存储器内容将在最后一次写操作完成之后与 RAM 中存储的预期数据进行比较。

图 13-7. 闪存程序存储器写校验序列



注：

1. 请参见图 13-1。

13.1.2.8. WRERR 位

WRERR 位可用于确定是否发生写错误。如果发生以下其中一种情况，WRERR 将置 1：

- 如果在 NVMADRH:NMVADRL 指向写保护地址时 WR 置 1
- 当正在进行自写操作时发生复位
- 解锁序列中断

WRERR 位通常由硬件置 1，但可由用户因测试目的而置 1。WRERR 置 1 后，必须用软件清零。

表 13-3. WR = 1 时对 PFM 执行的操作

FREE	LWLO	WR = 1 时对 PFM 执行的操作	备注
1	x	擦除 NVMADRH:NVMADRL 存储单元的 32 字行。	• 如果使能 WP，则 WR 清零且 WRERR 置 1 • 擦除所有 32 字 • 忽略 NVMDATH:NVMDATL
0	1	将 NVMDATH:NVMDATL 的内容复制到对应于 NVMADR LSB 的写锁存器。	• 忽略写保护 • 未发生存储器访问
0	0	将写锁存器数据写入 PFM 行。	• 如果使能 WP，则 WR 清零且 WRERR 置 1 • 写锁存器复位为 0x3FFF • 忽略 NVMDATH:NVMDATL

13.2. 寄存器定义：非易失性存储器控制

13.2.1. NVMADR

名称： NVMADR
偏移量： 0x1C8C

非易失性存储器地址寄存器

位	15	14	13	12	11	10	9	8
		NVMADR[14:8]						
访问		R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位		0	0	0	0	0	0	0

位	7	6	5	4	3	2	1	0
	NVMADR[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 14:0 – NVMADR[14:0] NVM 地址位

注：

1. 该多字节寄存器中的各个字节可使用以下寄存器名称进行访问：
   - NVMADRH：访问高字节 NVMADR[15:8]
   - NVMADRL：访问低字节 NVMADR[7:0]
2. 当 WR = 1 时，Bit [15]未定义。

13.2.2. NVMDAT

名称： NVMDAT
偏移量： 0x1C8E

非易失性存储器数据寄存器

位	15	14	13	12	11	10	9	8
			NVMDAT[13:8]					
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			x	x	x	x	x	x

位	7	6	5	4	3	2	1	0
	NVMDAT[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	x	x	x	x	x	x	x	x

Bit 13:0 – NVMDAT[13:0] NVM 数据位

复位状态： POR/BOR = xxxxxxxxxxxxxx
所有其他复位 = uuuuuuuuuuuuuu

注：该多字节寄存器中的各个字节可使用以下寄存器名称进行访问：

- NVMDATH：访问高字节 NVMDAT[13:8]
- NVMDATL：访问低字节 NVMDAT[7:0]

13.2.3. NVMCON1

名称： NVMCON1
偏移量： 0x1C90

非易失性存储器控制 1 寄存器

位	7	6	5	4	3	2	1	0
		NVMREGS	LWLO	FREE	WRERR	WREN	WR	RD
访问		R/W	R/W	R/S/HC	R/W/HS	R/W	R/S/HC	R/S/HC
复位		0	0	0	0	0	0	0

Bit 6 – NVMREGS NVM 区域选择

值	说明
1	访问 DIA、DCI、配置、用户 ID、版本 ID 和器件 ID 寄存器
0	访问闪存程序存储器

Bit 5 – LWLO 仅装入写锁存器

值	条件	说明
1	FREE =0	下一条 WR 命令更新行内该字的写锁存器；不启动存储器操作
0	FREE =0	下一条 WR 命令写数据或执行擦除操作
–	其他情况：	忽略此位

Bit 4 – FREE 闪存程序存储器擦除使能

值	说明
1	在下一条 WR 命令时执行擦除操作；擦除包含所指示地址的 32 字伪行（为全 1）以为写操作做准备
0	下一条 WR 命令写数据而不执行擦除操作

Bit 3 – WRERR
写复位错误标志位(1,2,3)

值	说明
1	发生写操作错误
0	所有写操作均正常完成

Bit 2 – WREN 编程/擦除使能

值	说明
1	允许编程/擦除周期
0	禁止对程序闪存的编程/擦除操作

Bit 1 – WR 写控制(4,5,6)

值	说明
1	在相应的 NVM 存储单元启动编程/擦除操作
0	对 NVM 的编程/擦除操作已完成并且变为无效

Bit 0 – RD 读控制

值	说明
1	在地址 = NVMADR 处启动读操作
0	NVM 读操作已完成并且变为无效。

注：

1. WR = 1 时位未定义
2. 该位必须由软件清零；硬件不能将该位清零。
3. 该位可由用户写入 1 以实现测试序列。
4. 只能在“**NVM 解锁序列**”一节所述的序列后将该位置 1。
5. 操作是自计时的，并且当操作完成时 WR 位由硬件清零。
6. 启动写操作之后，将该位设置为 0 将不起作用。

13.2.4. NVMCON2

名称： NVMCON2
偏移量： 0x1C91

非易失性存储器控制 2 寄存器

位	7	6	5	4	3	2	1	0
	NVMCON2[7:0]							
访问	WO	WO	WO	WO	WO	WO	WO	WO
复位	0	0	0	0	0	0	0	0

Bit 7:0 – NVMCON2[7:0] 闪存解锁模式位

注：要对写操作进行解锁，必须先写入 0x55，接着写入 0xAA，然后再将 NVMCON1 寄存器的 WR 位置 1。写入该寄存器的值用于对写操作进行解锁。

13.3. 寄存器汇总——NVM 控制

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x1C8B	保留									
0x1C8C	NVMADR	7:0	NVMADR[7:0]							
		15:8		NVMADR[14:8]						
0x1C8E	NVMDAT	7:0	NVMDAT[7:0]							
		15:8			NVMDAT[13:8]					
0x1C90	NVMCON1	7:0		NVMREGS	LWLO	FREE	WRERR	WREN	WR	RD
0x1C91	NVMCON2	7:0	NVMCON2[7:0]							

14. I/O 端口

14.1. 概述

表 14-1. 每款器件可用的端口

器件	PORTA	PORTB	PORTC
14 引脚器件	●		●

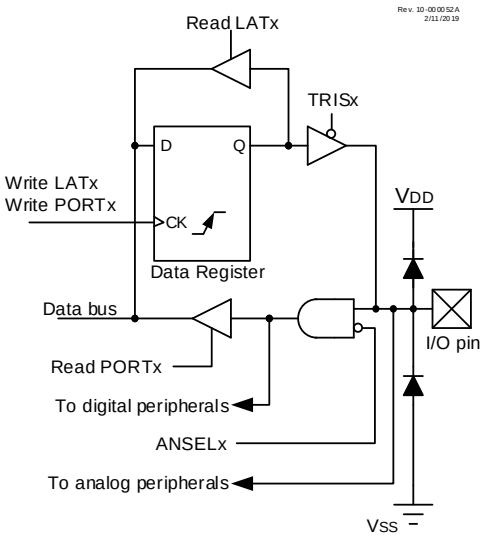
每个端口都有 8 个控制操作的寄存器。这些寄存器包括：

- PORTx 寄存器（读取器件引脚的电平）
- LATx 寄存器（输出锁存器）
- TRISx 寄存器（数据方向）
- ANSELx 寄存器（模拟选择）
- WPUx 寄存器（弱上拉）
- INLVLx（输入电平控制）
- SLRCONx 寄存器（压摆率控制）
- ODCONx 寄存器（漏极开路控制）

在本节中，PORTx、LATx 和 TRISx 等通用名称可与 PORTA、PORTB 和 PORTC 等相关联，具体取决于每款器件可用的端口。

下图给出了通用 I/O 端口的简化模型，图中未显示与其他外设的接口：

图 14-1. 通用 I/O 端口的工作原理



14.2. PORTx——数据寄存器

PORTx 是一个双向端口，对应的数据方向寄存器是 TRISx。

读 PORTx 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。所有写操作都是读-修改-写操作。因此，对端口的写操作意味着先读 PORT 引脚电平状态，然后修改此值，最后再写入

PORT 数据锁存器（LATx）。PORT 数据锁存器 LATx 保存输出端口数据，并包含写入 LATx 或 PORTx 的最新值。以下示例显示了如何初始化 PORTA。

例 14-1. 用汇编语言初始化 PORTA

```
; This code example illustrates initializing the PORTA register.
; The other ports are initialized in the same manner.

    BANKSEL    PORTA        ;
    CLRF       PORTA        ;Clear PORTA
    BANKSEL    LATA         ;
    CLRF       LATA         ;Clear Data Latch
    BANKSEL    ANSELA       ;
    CLRF       ANSELA       ;Enable digital drivers
    BANKSEL    TRISA        ;
    MOVLW      B'00111000'  ;Set RA[5:3] as inputs
    MOVWF      TRISA        ;and set others as outputs
```

例 14-2. 用 C 语言初始化 PORTA

```
// This code example illustrates initializing the PORTA register.
// The other ports are initialized in the same manner.

    PORTA  = 0x00;       // Clear PORTA
    LATA   = 0x00;       // Clear Data Latch
    ANSELA = 0x00;       // Enable digital drivers
    TRISA  = 0x38;       // Set RA[5:3] as inputs and set others as outputs
```

重要：大多数 PORT 引脚与器件模拟外设和数字外设共用功能。通常，当使能某个 PORT 引脚上的外设时，该引脚将不能用作通用输出，但仍然可以对该引脚进行读操作。

14.3. LATx——输出锁存器

数据锁存器（LATx 寄存器）用于对 I/O 引脚所驱动的值进行读-修改-写操作。

对 LATx 寄存器的写操作与写入相应 PORTx 寄存器的效果相同。读取 LATx 寄存器时，将会读取 I/O PORT 锁存器中保存的值，而读取 PORTx 寄存器时，将会读取实际的 I/O 引脚值。

重要：一般来说，对端口的输出操作必须使用 LAT 寄存器，以避免读-修改-写问题。例如，通过位置 1 或清零操作来读取端口、修改位并将结果写回端口中。当连续执行两次位操作时，已更改位的输出装载可能导致输出更改发生延迟，在这种情况下，第二次位操作会误读位并写入意外值。LAT 寄存器与端口装载隔离，因此更改不会延迟。

14.4. TRISx——方向控制

TRISx 寄存器用于控制 PORTx 引脚输出驱动器，即使引脚被用作模拟输入。当引脚用作模拟输入时，用户必须确保 TRISx 寄存器中的相应位置 1。配置为模拟输入的 I/O 引脚总是读为 0。

将 TRISx 位置 1（TRISx = 1）时，会将 PORTx 的相应引脚设为输入（即，禁止输出驱动器）。将 TRISx 位清零（TRISx = 0）时，会将 PORTx 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定的引脚）。

14.5. ANSELx——模拟控制

支持模拟输入的端口具有相关的 ANSELx 寄存器。ANSELx 寄存器用于将 I/O 引脚的输入模式配置为模拟。将 ANSELx 位设置为高电平将禁止与该位相关的数字输入缓冲区，并导致相应输入值始终读为 0（在 PORTx 寄存器中读取值或通过 PPS 将值选择为外设输入）。

禁止输入缓冲器可以防止该引脚上介于逻辑高电平和低电平之间的模拟信号电压在逻辑输入电路中产生过大的电流。

ANSELx 位的状态不会影响数字或模拟输出功能。TRIS 清零且 ANSEL 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。当在 PORTx 寄存器上执行读-修改-写指令时，引脚行为可能与预期不符。

重要：在发生复位之后，ANSELx 位默认设为模拟模式。要将任意引脚用作数字通用输入或外设输入，必须由用户将相应的 ANSEL 位更改为 0。

14.6. WPUx——弱上拉控制

WPUx 寄存器用于控制每个 PORT 引脚的各个弱上拉功能。当 WPUx 位置 1（WPUx = 1）时，将为相应引脚使能弱上拉功能。当 WPUx 位清零（WPUx = 0）时，将为相应引脚禁止弱上拉功能。

14.7. INLVLx——输入阈值控制

INLVLx 寄存器用于控制每个可用 PORTx 输入引脚的输入阈值电压。用户可以选择施密特触发器 CMOS 阈值和 TTL 兼容阈值。输入阈值对于确定 PORTx 寄存器的读取值很重要，同时它也是发生电平变化中断的临界电压（如果使能该功能）。有关阈值电压的更多详细信息，请参见**“电气规范”**一章中的“I/O 端口”表。

重要：如果要更改所选择的输入阈值，则必须先禁止所有外设模块再执行该操作。在模块处于活动状态时更改阈值电压可能会意外产生与输入引脚相关联的电平变化，不论该引脚上的实际电压如何。

14.8. SLRCONx——压摆率控制

SLRCONx 寄存器用于控制每个 PORT 引脚的压摆率选项。每个 PORT 引脚的压摆率可以独立进行控制。当 SLRCONx 位置 1（SLRCONx = 1）时，相应 PORT 引脚驱动器的压摆率会受到限制。当 SLRCONx 位清零（SLRCONx = 0）时，相应 PORT 引脚驱动器的压摆率将为最大可能值。

14.9. ODCONx——漏极开路控制

ODCONx 寄存器用于控制端口的漏极开路功能。每个引脚的漏极开路操作可以独立进行选择。当 ODCONx 位置 1（ODCONx = 1）时，相应的端口输出会变为只能灌入电流的漏极开路驱动器。当 ODCONx 位清零（ODCONx = 0）时，相应的端口输出引脚是能够拉出和灌入电流的标准推挽驱动器。

重要：当将引脚用于 I^2C 功能时，需要设置漏极开路控制。

14.10. 边沿可选的电平变化中断

可通过检测具有上升沿或下降沿的 PORT 引脚的信号来产生中断。任何一个引脚都可以配置为产生中断。更多详细信息，请参见“**IOC——电平变化中断**”一章。

14.11. I²C 焊盘控制

对于该系列器件，RC0 和 RC1 引脚上提供 I²C 特定焊盘。其中每个引脚的 I²C 特性由 RxyI2C 寄存器控制。这些特性包括使能 I²C 特定压摆率（相对于标准 GPIO 压摆率），为 I²C 引脚选择内部上拉，以及根据 SMBus 规范选择相应的输入阈值。

重要：使用 I²C 引脚的任何外设都将读取 I²C 输入电平（通过 RxyI2C 使能时）。

14.12. I/O 优先级

在复位之后，每个引脚均默认设为数据锁存器。其他功能通过外设引脚选择逻辑进行选择。更多详细信息，请参见**“PPS——外设引脚选择模块”**一章。

外设引脚选择列表中未列出模拟输入功能，例如 ADC 和比较器输入。这些输入在使用 ANSELx 寄存器将 I/O 引脚设置为模拟模式时有效。当引脚处于模拟模式时，数字输出功能可以继续控制引脚。

当使能模拟输出时，模拟输出优先于数字输出且强制数字输出驱动器为高阻态。

引脚功能的优先级如下所示：

1. 由配置位决定的端口功能。
2. 模拟输出（必须禁止输入缓冲器）。
3. 模拟输入。
4. PPS 的端口输入和输出。

14.13. $\overline{MCLR}/V_{PP}$/RA3 引脚

$\overline{MCLR}/V_{PP}$ 引脚只用作输入引脚。其操作由 MCLRE 配置位控制。当选作 PORT 引脚（MCLRE = 0）时，它只能用作数字输入引脚；因此没有与其操作相关的 TRISx 和 LATx 位。其他情况下，它用作器件的主复位输入。在任意一种配置中，$\overline{MCLR}/V_{PP}$ 引脚均还可用作高电压编程期间的编程电压输入引脚。

$\overline{MCLR}/V_{PP}$ 引脚是只读位，在 MCLRE = 1（即，使能主复位）时读为 1。

重要：在上电复位时，只有在禁止主复位功能的情况下，才会使能 $\overline{MCLR}/V_{PP}$ 引脚作为数字输入。

$\overline{MCLR}/V_{PP}$ 引脚具有单独控制的内部弱上拉功能。置 1 时，相应 WPU 位将使能上拉功能。当 $\overline{MCLR}/V_{PP}$ 引脚配置为 $\overline{MCLR}$（MCLRE = 1 且 LVP = 0）或配置为低电压编程（MCLRE = x 且 LVP = 1）时，始终使能上拉，WPU 位不产生任何影响。

14.14. 寄存器定义：端口控制

14.14.1. PORTx

名称： PORTx

PORTx 寄存器

位	7	6	5	4	3	2	1	0
	Rx7	Rx6	Rx5	Rx4	Rx3	Rx2	Rx1	Rx0
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	x	x	x	x	x	x	x	x

Bit 0, 1, 2, 3, 4, 5, 6, 7 – Rxn 端口 I/O 值

复位状态： POR/BOR = xxxxxxxx
所有其他复位 = uuuuuuuu

值	说明
1	PORT 引脚电压 ≥ V_{IH}
0	PORT 引脚电压 ≤ V_{IL}

重要：

- 写入 PORTx 时，实际上会写入相应的 LATx 寄存器。读取 PORTx 寄存器时，将返回实际的 I/O 引脚值。
- 与 $\overline{MCLR}$ 引脚关联的 PORT 位为只读位，在使能 $\overline{MCLR}$ 功能（LVP = 1 或（LVP = 0 且 MCLRE = 1））时读为 1
- 有关 $\overline{MCLR}$ 引脚和每个端口的引脚可用性的详细信息，请参见**“引脚分配表”**
- 未实现位将读回 0
- 在调试模式下，RB6 和 RB7 位读为 1

14.14.2. LATx

名称： LATx

输出锁存器寄存器

位	7	6	5	4	3	2	1	0
	LATx7	LATx6	LATx5	LATx4	LATx3	LATx2	LATx1	LATx0
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	x	x	x	x	x	x	x	x

Bit 0, 1, 2, 3, 4, 5, 6, 7 – LATxn 输出锁存器值

复位状态： POR/BOR = xxxxxxxx
所有其他复位 = uuuuuuuu

重要：

- 写入 LATx 相当于写入相应的 PORTx 寄存器。读取 LATx 寄存器时，将返回寄存器值，而不是 I/O 引脚值。
- 有关每个端口的引脚可用性的详细信息，请参见**“引脚分配表”**
- 未实现的位将读回 0

14.14.3. TRISx

名称： TRISx

三态控制寄存器

位	7	6	5	4	3	2	1	0
	TRISx7	TRISx6	TRISx5	TRISx4	TRISx3	TRISx2	TRISx1	TRISx0
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	1	1	1	1	1	1	1	1

Bit 0, 1, 2, 3, 4, 5, 6, 7 – TRISxn 端口 I/O 三态控制

值	说明
1	禁止 PORTx 输出驱动器。PORTx 引脚配置为输入（三态）。
0	使能 PORTx 输出驱动器。PORTx 引脚配置为输出。

重要：

- 与 $\overline{MCLR}$ 引脚关联的 TRIS 位为只读位，值为 1
- 有关 $\overline{MCLR}$ 引脚和每个端口的引脚可用性的详细信息，请参见**“引脚分配表”**
- 未实现的位将读回 0

14.14.4. ANSELx

名称： ANSELx

模拟选择寄存器

位	7	6	5	4	3	2	1	0
	ANSELx7	ANSELx6	ANSELx5	ANSELx4	ANSELx3	ANSELx2	ANSELx1	ANSELx0
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	1	1	1	1	1	1	1	1

Bit 0, 1, 2, 3, 4, 5, 6, 7 – ANSELxn RX 引脚的模拟选择

值	说明
1	模拟输入。引脚被指定为模拟输入。数字输入缓冲器被禁止。
0	数字 I/O。引脚被配置为端口或数字特殊功能。

重要：

- 当将某个引脚设置为模拟输入时，必须将相应的 TRIS 位设置为输入模式，以允许从外部控制引脚电压
- 有关每个端口的引脚可用性的详细信息，请参见“**引脚分配表**”
- 未实现的位将读回 0

14.14.5. WPUx

名称： WPUx

弱上拉寄存器

位	7	6	5	4	3	2	1	0
	WPUx7	WPUx6	WPUx5	WPUx4	WPUx3	WPUx2	WPUx1	WPUx0
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 0, 1, 2, 3, 4, 5, 6, 7 – WPUxn 弱上拉 PORTx 控制

值	说明
1	使能弱上拉
0	禁止弱上拉

重要：

- 如果引脚被配置为输出但该寄存器保持不变，则自动禁止弱上拉器件
- 如果 MCLRE = 1，则 $\overline{\text{MCLR}}$ 引脚的弱上拉始终使能，相应的 WPU 位不受影响
- 有关每个端口的引脚可用性的详细信息，请参见“**引脚分配表**”
- 未实现的位将读回 0

14.14.6. INLVLx

名称： INLVLx

输入电平控制寄存器

位	7	6	5	4	3	2	1	0
	INLVLx7	INLVLx6	INLVLx5	INLVLx4	INLVLx3	INLVLx2	INLVLx1	INLVLx0
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	1	1	1	1	1	1	1	1

Bit 0, 1, 2, 3, 4, 5, 6, 7 – INLVLxn RX 引脚的输入电平选择

值	说明
1	对于端口读操作和电平变化中断，使用 ST 输入
0	对于端口读操作和电平变化中断，使用 TTL 输入

重要：

- 有关每个端口的引脚可用性的详细信息，请参见“**引脚分配表**”
- 未实现的位将读回 0
- 使用 I^2C 引脚的任何外设都将读取 I^2C ST 输入（通过 RxyI2C 使能时）

14.14.7. SLRCONx

名称： SLRCONx

压摆率控制寄存器

位	7	6	5	4	3	2	1	0
	SLRx7	SLRx6	SLRx5	SLRx4	SLRx3	SLRx2	SLRx1	SLRx0
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	1	1	1	1	1	1	1	1

Bit 0, 1, 2, 3, 4, 5, 6, 7 – SLRxn RX 引脚的压摆率控制

值	说明
1	PORT 引脚的压摆率受到限制
0	PORT 引脚的压摆率为最大值

重要：

- 有关每个端口的引脚可用性的详细信息，请参见“**引脚分配表**”
- 未实现的位将读回 0

14.14.8. ODCONx

名称： ODCONx

压漏极开路控制寄存器

位	7	6	5	4	3	2	1	0
	ODCx7	ODCx6	ODCx5	ODCx4	ODCx3	ODCx2	ODCx1	ODCx0
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 0, 1, 2, 3, 4, 5, 6, 7 – ODCxn Rx 引脚的漏极开路配置

值	说明
1	PORT 引脚作为漏极开路驱动器工作（仅灌电流）
0	PORT 引脚作为标准推挽驱动器工作（拉电流和灌电流）

重要：

- 有关每个端口的引脚可用性的详细信息，请参见“**引脚分配表**”
- 未实现的位将读回 0

14.14.9. RxyI2C

名称： RxyI2C

I²C 焊盘 Rxy 控制寄存器

位	7	6	5	4	3	2	1	0
		SLEW	PU[1:0]				TH[1:0]	
访问		R/W	R/W	R/W			R/W	R/W
复位		0	0	0			0	0

Bit 6 – SLEW I²C 特定压摆率限制控制

值	说明
1	使能 I²C 特定压摆率限制。禁止标准焊盘压摆率限制。SLRxy 位被忽略
0	标准 GPIO 压摆率；通过 SLRxy 位使能/禁止

Bit 5:4 – PU[1:0] I²C 上拉选择

值	说明
11	保留
10	标准弱上拉电流的 10 倍
01	标准弱上拉电流的 2 倍
00	标准 GPIO 弱上拉，通过 WPUxy 位使能

Bit 1:0 – TH[1:0] I²C 输入阈值选择

值	说明
11	保留
10	SMBus 2.0（2.1V）输入阈值
01	I²C 特定输入阈值
00	标准 GPIO 输入上拉，通过 INLVLxy 寄存器使能

重要：

- 有关每个端口的引脚可用性的详细信息，请参见“**引脚分配表**”
- 未实现的位将读回 0

14.15. 寄存器汇总——I/O 端口

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0B	保留									
0x0C	PORTA	7:0			RA5	RA4	RA3	RA2	RA1	RA0
0x0D	保留									
0x0E	PORTC	7:0			RC5	RC4	RC3	RC2	RC1	RC0
0x0F ... 0x11	保留									
0x12	TRISA	7:0			TRISA5	TRISA4	保留	TRISA2	TRISA1	TRISA0
0x13	保留									
0x14	TRISC	7:0			TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
0x15 ... 0x17	保留									
0x18	LATA	7:0			LATA5	LATA4		LATA2	LATA1	LATA0
0x19	保留									
0x1A	LATC	7:0			LATC5	LATC4	LATC3	LATC2	LATC1	LATC0
0x1B ... 0x010D	保留									
0x010E	RC0I2C	7:0		SLEW	PU[1:0]				TH[1:0]	
0x010F	RC1I2C	7:0		SLEW	PU[1:0]				TH[1:0]	
0x0110 ... 0x1F37	保留									
0x1F38	ANSELA	7:0			ANSELA5	ANSELA4		ANSELA2	ANSELA1	ANSELA0
0x1F39	WPUA	7:0			WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
0x1F3A	ODCONA	7:0			ODCA5	ODCA4		ODCA2	ODCA1	ODCA0
0x1F3B	SLRCONA	7:0			SLRA5	SLRA4		SLRA2	SLRA1	SLRA0
0x1F3C	INLVLA	7:0			INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0
0x1F3D ... 0x1F4D	保留									
0x1F4E	ANSELC	7:0			ANSELC5	ANSELC4	ANSELC3	ANSELC2	ANSELC1	ANSELC0
0x1F4F	WPUC	7:0			WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
0x1F50	ODCONC	7:0			ODCC5	ODCC4	ODCC3	ODCC2	ODCC1	ODCC0
0x1F51	SLRCONC	7:0			SLRC5	SLRC4	SLRC3	SLRC2	SLRC1	SLRC0
0x1F52	INLVLC	7:0			INLVLC5	INLVLC4	INLVLC3	INLVLC2	INLVLC1	INLVLC0

15. IOC——电平变化中断

15.1. 概述

下表所示的引脚可配置为用作该器件的电平变化中断（Interrupt-On-Change，IOC）引脚。可通过检测具有上升沿或下降沿的信号来产生中断。任意一个 PORT 引脚或 PORT 引脚组合都可以配置为产生中断。

表 15-1. 每款器件可用的 IOC 引脚

器件	PORTA	PORTB	PORTC
14 引脚器件	●		●

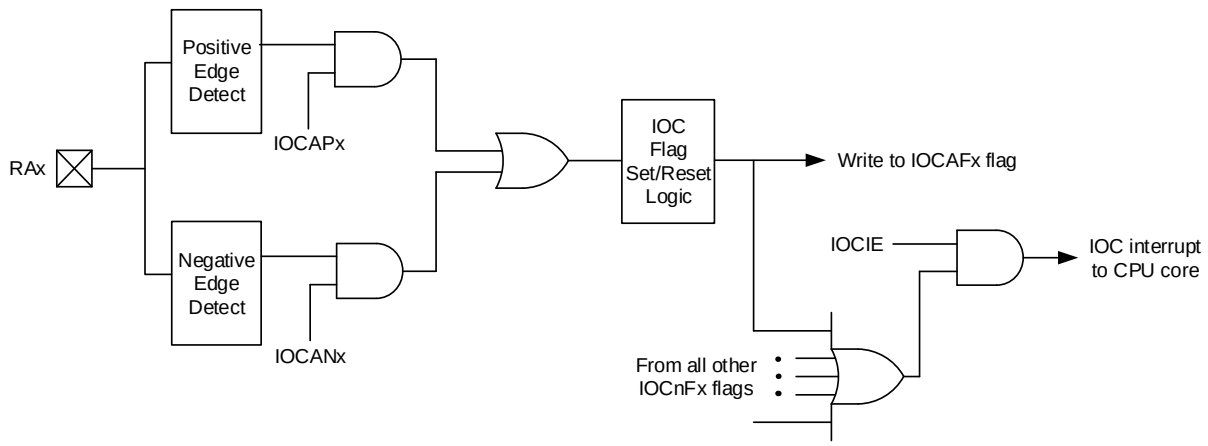
重要：如果 MCLRE = 1 或 LVP = 1，则 $\overline{\text{MCLR}}$ 引脚端口功能将被禁止，该引脚上的 IOC 不可用。

电平变化中断模块具有以下功能：

- 电平变化中断允许（主开关）
- 独立的引脚配置
- 上升沿和下降沿检测
- 独立的引脚中断标志

下图给出了 IOC 模块的框图。

图 15-1. 电平变化中断框图（以 PORTA 为例）



15.2. 使能模块

为了使各个 PORT 引脚产生中断，必须将外设中断允许（PIEx）寄存器的 IOC 中断允许（IOCIE）位置 1。如果 IOC 中断允许位被禁止，在引脚上仍然会发生边沿检测，但不会产生中断。

15.3. 独立的引脚配置

对于每个 PORT 引脚，都提供了上升沿检测器和下降沿检测器。要允许引脚检测上升沿，必须将 IOCxP 寄存器的相关位置 1。要允许引脚检测下降沿，必须将 IOCxN 寄存器的相关位置 1。通过同时将 IOCxP 和 IOCxN 寄存器的相关位置 1，一个 PORT 引脚可以配置为同时检测上升沿和下降沿。

15.4. 中断标志

IOCxF 寄存器中的各个位是与每个端口的电平变化中断引脚对应的状态标志位。如果在正确使能的引脚上检测到期望的边沿，则对应于该引脚的状态标志会置 1，并且如果 IOCIE 位也置 1，则还会产生中断。相应外设中断请求（PIRx）寄存器中的 IOCIF 位是所有 IOCxF 位的逻辑或运算结果。IOCIF 位是只读位。必须将所有 IOCxF 状态位都清零才能清零 IOCIF 位。

15.5. 清零中断标志

各个状态标志（IOCxF 寄存器位）将通过将其复位为零的方式清零。如果在该清零操作期间检测到另一个边沿，则无论实际写入的值如何，相关的状态标志都会在序列结束时置 1。

为了确保清零标志时不丢失所检测到的边沿，必须仅执行“逻辑与”操作，将除变化的位以外的其余位掩码。以下序列是使用该方法清零 IOC 中断标志的示例。

例 15-1. 清零中断标志（以 PORTA 为例）

```
MOVLW    0xff
XORWF    IOCAF, W
ANDWF    IOCAF, F
```

15.6. 休眠期间的操作

如果 IOCIE 位置 1，电平变化中断事件会将器件从休眠模式唤醒。如果在处于休眠模式时检测到边沿，则在退出休眠模式执行第一条指令之前，会先更新 IOCxF 寄存器。

15.7. 寄存器定义：电平变化中断控制

15.7.1. IOCxF

名称： IOCxF

电平变化中断标志寄存器

位	7	6	5	4	3	2	1	0
	IOCxF7	IOCxF6	IOCxF5	IOCxF4	IOCxF3	IOCxF2	IOCxF1	IOCxF0
访问	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS
复位	0	0	0	0	0	0	0	0

Bit 0, 1, 2, 3, 4, 5, 6, 7 – IOCxFn 电平变化中断标志

值	条件	说明
1	IOCxP[n] =1	检测到 Rx[n]引脚上有正边沿
1	IOCxN[n] =1	检测到 Rx[n]引脚上有负边沿
0	IOCxP[n] =x 且 IOCxN[n] =x	未检测到电平变化或用户清除了检测到的电平变化

重要：

- 如果 MCLRE = 1 或 LVP = 1，则禁止 $\overline{\text{MCLR}}$ 引脚端口功能，并且该引脚上的 IOC 功能不可用
- 有关可按端口配置 IOC 的引脚的详细信息，请参见**“引脚分配表”**

15.7.2. IOCxN

名称： IOCxN

电平变化中断负边沿寄存器示例

位	7	6	5	4	3	2	1	0
	IOCxN7	IOCxN6	IOCxN5	IOCxN4	IOCxN3	IOCxN2	IOCxN1	IOCxN0
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 0, 1, 2, 3, 4, 5, 6, 7 – IOCxNn 电平变化中断负边沿使能

值	说明
1	允许 IOCx 引脚上的负边沿电平变化中断。检测到边沿时将相关的状态位和中断标志置 1。
0	禁止关联引脚的负边沿电平变化中断。

重要：

- 如果 MCLRE = 1 或 LVP = 1，则禁止 $\overline{\text{MCLR}}$ 引脚端口功能，并且该引脚上的 IOC 功能不可用
- 有关可按端口配置 IOC 的引脚的详细信息，请参见**“引脚分配表”**

15.7.3. IOCxP

名称： IOCxP

电平变化中断正边沿寄存器

位	7	6	5	4	3	2	1	0
	IOCxP7	IOCxP6	IOCxP5	IOCxP4	IOCxP3	IOCxP2	IOCxP1	IOCxP0
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 0, 1, 2, 3, 4, 5, 6, 7 – IOCxPn 电平变化中断正边沿使能

值	说明
1	允许 IOCx 引脚上的正边沿电平变化中断。检测到边沿时将相关的状态位和中断标志置 1。
0	禁止关联引脚的正边沿电平变化中断

重要：

- 如果 MCLRE = 1 或 LVP = 1，$\overline{\text{MCLR}}$ 引脚端口功能将被禁止，该引脚上的 IOC 功能不可用
- 有关可按端口配置 IOC 的引脚的详细信息，请参见**“引脚分配表”**

15.8. 寄存器汇总——电平变化中断

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x1F3C	保留									
0x1F3D	IOCAP	7:0			IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0
0x1F3E	IOCAN	7:0			IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0
0x1F3F	IOCAF	7:0			IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0
0x1F40 ... 0x1F52	保留									
0x1F53	IOCCP	7:0			IOCCP5	IOCCP4	IOCCP3	IOCCP2	IOCCP1	IOCCP0
0x1F54	IOCCN	7:0			IOCCN5	IOCCN4	IOCCN3	IOCCN2	IOCCN1	IOCCN0
0x1F55	IOCCF	7:0			IOCCF5	IOCCF4	IOCCF3	IOCCF2	IOCCF1	IOCCF0

16. PPS——外设引脚选择模块

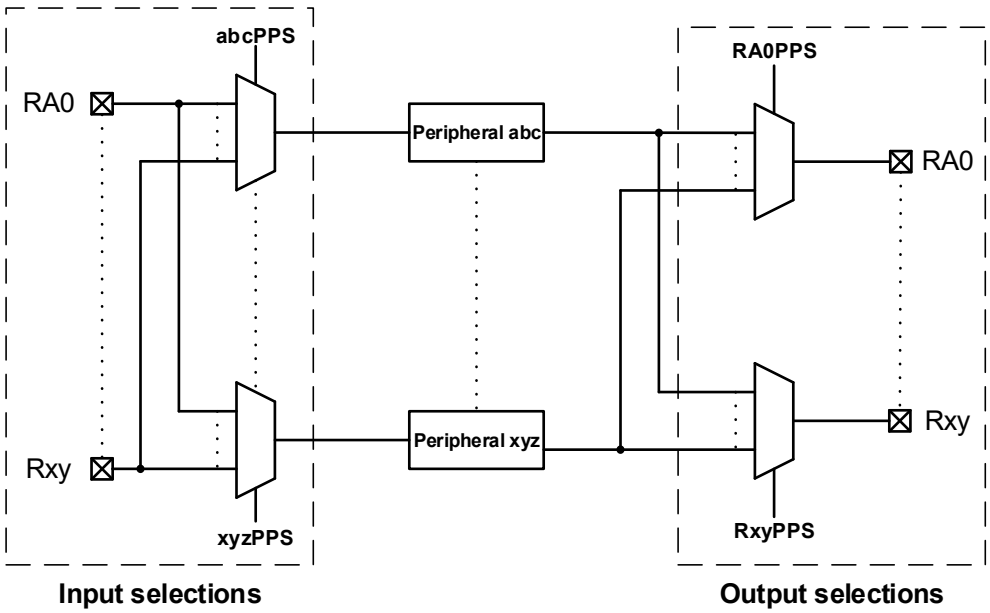
16.1. 概述

外设引脚选择（PPS）模块用于将外设输入和输出与器件 I/O 引脚连接。选择范围仅包含数字信号。

重要：所有模拟输入和输出保持固定连接至为其分配的引脚，无法通过 PPS 更改。

输入和输出选择是独立的，如下图所示。

图 16-1. PPS 框图



16.2. PPS 输入

每个数字外设均具有一个专用的 PPS 外设输入选择（xxxPPS）寄存器，用于选择外设输入引脚。引脚数不超过 20（8/14/16/20）的器件允许 PPS 连接到任何 I/O 引脚，而引脚数超出 28 的器件则允许 PPS 连接到包含在两个端口内的 I/O（见下表）。

重要：通用寄存器名称中的符号“xxx”是外设标识符的占位符。例如，xxx = T0CKI 代表 T0CKIPPS 寄存器。

多个外设可以同时使用同一个源工作。无论外设 PPS 选择为何，端口读操作始终返回引脚电平。如果某个引脚还具有关联的模拟功能，则必须清零该引脚的 ANSEL 位才可使能数字输入缓冲区。

表 16-1. PPS 输入选择表

外设	PPS 输入寄存器	POR 时的默认引脚选择
外部中断	INTPPS	RA2

表 16-1. PPS 输入选择表 （续）

外设	PPS 输入寄存器	POR 时的默认引脚选择
Timer0 时钟	T0CKIPPS	RA2
Timer1 时钟	T1CKIPPS	RA5
Timer1 门控	T1GPPS	RA4
Timer2 输入	T2INPPS	RA5
CCP1	CCP1PPS	RC5
CCP2	CCP2PPS	RC3
SCL1/SCK1	SSP1CLKPPS(1)	RC0
SDA1/SDI1	SSP1DATPPS(1)	RC1
$\overline{SS1}$	SSP1SSPPS	RC3
RX1/DT1	RX1PPS	RC5
CK1	CK1PPS	RC4
ADC 转换触发器	ADACTPPS	RC2

注：

1. 双向引脚。相应输出必须选择同一引脚。

16.3. PPS 输出

每个数字外设均具有一个专用的引脚 Rxy 输出源选择（RxyPPS）寄存器，用于选择引脚输出源。除了少数例外情况，与该引脚相关的端口 TRIS 控制将保持对引脚输出驱动器的控制权。作为外设操作的一部分，控制引脚输出驱动器的外设将根据需要改写 TRIS 控制。例如 I²C 模块就是这样的外设。

重要：符号“Rxy”是引脚标识符的占位符。“x”是 PORT 字母的占位符，“y”是位编号的占位符。例如，Rxy = RA0 代表 RA0PPS 寄存器。

下表列出了每个外设的输出代码以及可选的端口。

表 16-2. PPS 输出选择表

RxyPPS	输出源
0x09	TMR0
0x08	SDA1/SDO1(1)
0x07	SCL1/SCK1(1)
0x06	DT1
0x05	TX1/CK1
0x04	PWM4
0x03	PWM3
0x02	CCP2
0x01	CCP1
0x00	LATxy

注：

1. 双向引脚。相应输入必须选择同一引脚。

16.4. 双向引脚

对于在单个引脚上具有双向信号的外设，在进行 PPS 选择时必须使 PPS 输入和 PPS 输出选择同一引脚。例如 I²C 串行时钟（SCL）和串行数据（SDA）引脚。

重要：I²C 默认引脚以及有限数量的其他备用引脚与 I²C 和 SMBus 兼容。SDA 和 SCL 信号可输送到任何引脚；但是不具有 I²C 兼容性的引脚将以标准 TTL/ST 逻辑电平（由端口的 INLVL 寄存器选择）工作。

16.5. PPS 锁定

PPS 模块提供一个额外的保护层来防止意外更改 PPS 选择寄存器。PPSLOCKED 位与特定代码执行模块搭配使用来锁定/解锁 PPS 选择寄存器。

重要：PPSLOCKED 位默认清零（PPSLOCKED = 0），因此无需解锁序列便可修改 PPS 选择寄存器。

当 PPSLOCKED 位置 1（PPSLOCKED = 1）时，PPS 选择寄存器被锁定。将 PPSLOCKED 位置 1 需要以下示例中所示的特定锁定序列（C 语言和汇编语言）。

当 PPSLOCKED 位清零（PPSLOCKED = 0）时，PPS 选择寄存器解锁。将 PPSLOCKED 位清零需要以下示例中所示的特定解锁序列（C 语言和汇编语言）。

重要：为确保正常执行，必须在启动锁定/解锁序列之前禁止所有中断。

例 16-1. PPS 锁定序列（汇编语言）

```
; suspend interrupts
   BCF      INTCON0,GIE
   BANKSEL  PPSLOCK
; required sequence, next 5 instructions
   MOVLW    0x55
   MOVWF    PPSLOCK
   MOVLW    0xAA
   MOVWF    PPSLOCK
; Set PPSLOCKED bit
   BSF      PPSLOCK,PPSLOCKED
; restore interrupts
   BSF      INTCON0,GIE
```

例 16-2. PPS 锁定序列（C 语言）

```
INTCON0bits.GIE = 0;         //Suspend interrupts
PPSLOCK = 0x55;              //Required sequence
PPSLOCK = 0xAA;              //Required sequence
PPSLOCKbits.PPSLOCKED = 1;   //Set PPSLOCKED bit
INTCON0bits.GIE = 1;         //Restore interrupts
```

例 16-3. PPS 解锁序列（汇编语言）

```
; suspend interrupts
   BCF      INTCON0,GIE
   BANKSEL  PPSLOCK
; required sequence, next 5 instructions
   MOVLW    0x55
   MOVWF    PPSLOCK
   MOVLW    0xAA
   MOVWF    PPSLOCK
; Clear PPSLOCKED bit
   BCF      PPSLOCK,PPSLOCKED
; restore interrupts
   BSF      INTCON0,GIE
```

例 16-4. PPS 解锁序列（C 语言）

```
INTCON0bits.GIE = 0;          //Suspend interrupts
PPSLOCK = 0x55;               //Required sequence
PPSLOCK = 0xAA;               //Required sequence
PPSLOCKbits.PPSLOCKED = 0;    //Clear PPSLOCKED bit
INTCON0bits.GIE = 1;          //Restore interrupts
```

16.5.1. PPS 单向锁定

PPS1WAY 配置位还可用于防止意外修改 PPS 选择寄存器。

PPS1WAY 位置 1（PPS1WAY = 1）时，PPSLOCKED 位只能在器件复位之后置 1 一次。PPSLOCKED 位置 1 后，除非执行器件复位，否则该位无法再次清零。

当 PPS1WAY 位清零（PPS1WAY = 0）时，可根据需要将 PPSLOCKED 位置 1 或清零；但必须执行 PPS 锁定/解锁序列。

16.6. 休眠期间的操作

PPS 输入和输出选择不会受休眠影响。

16.7. 复位的影响

器件上电复位（POR）或欠压复位（BOR）会将所有 PPS 输入选择寄存器恢复为其默认值，并清零所有 PPS 输出选择寄存器。所有其他复位会将这些选择保留不变。PPS 输入寄存器详细信息表中列出了默认的输入选择。在所有复位条件下，PPSLOCKED 位均会清零。

16.8. 寄存器定义：外设引脚选择（PPS）

16.8.1. xxxPPS

名称： xxxPPS

外设输入选择寄存器

位	7	6	5	4	3	2	1	0
			PORT[2:0]			PIN[2:0]		
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			m	m	m	m	m	m

Bit 5:3 – PORT[2:0] 外设输入 PORT 选择(1)
有关可用端口和默认引脚位置的列表，请参见 PPS 输入选择表。

PORT	选择
100	PORTE(1)
011	PORTD(1)
010	PORTC
001	PORTB(1)
000	PORTA

注：
1. 这些配置不适用于 14 引脚器件。对于 CN5225 系列，这些设置为保留设置，仅供参考。

复位状态： POR = mmm
所有其他复位 = uuu

Bit 2:0 – PIN[2:0] 外设输入 PORT 引脚选择(2)

复位状态： POR = mmm
所有其他复位 = uuu

值	说明
111	外设输入为 PORTx 引脚 7（Rx7）
110	外设输入为 PORTx 引脚 6（Rx6）
101	外设输入为 PORTx 引脚 5（Rx5）
100	外设输入为 PORTx 引脚 4（Rx4）
011	外设输入为 PORTx 引脚 3（Rx3）
010	外设输入为 PORTx 引脚 2（Rx2）
001	外设输入为 PORTx 引脚 1（Rx1）
000	外设输入为 PORTx 引脚 0（Rx0）

注：
1. 复位值“m”由该输入的器件默认位置确定。
2. 有关每个端口的可用引脚的详细信息，请参见**“引脚分配表”**。

16.8.2. RxyPPS

名称： RxyPPS

引脚 Rxy 输出源选择寄存器

位	7	6	5	4	3	2	1	0
			RxyPPS[5:0]					
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			0	0	0	0	0	0

Bit 5:0 – RxyPPS[5:0] 引脚 Rxy 输出源选择
有关 RxyPPS 输出源代码的列表，请参见 PPS 输出选择表。

复位状态： POR = 000000
所有其他复位 = uuuuuu

16.8.3. PPSLOCK

名称： PPSLOCK

PPS 锁定寄存器

位	7	6	5	4	3	2	1	0
								PPSLOCKED
访问								R/W
复位								0

Bit 0 – PPSLOCKED PPS 锁定

复位状态： POR = 0
所有其他复位 = 0

值	说明
1	PPS 已锁定。无法更改 PPS 选择。对任何 PPS 寄存器执行的写操作都将被忽略。
0	PPS 未锁定。可以更改 PPS 选择，但可能需要 PPS 锁定/解锁序列。

16.9. 寄存器汇总——外设引脚选择模块

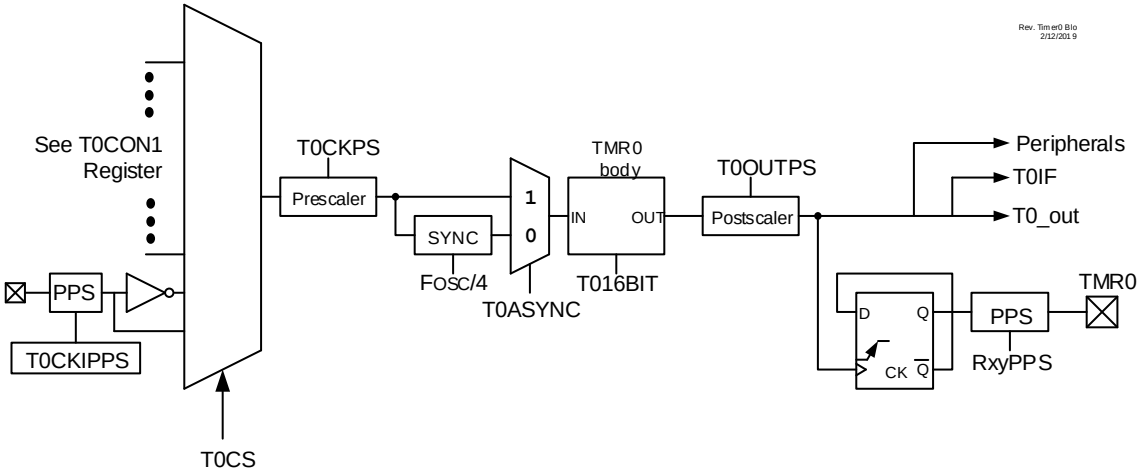
偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x1E8E	保留									
0x1E8F	PPSLOCK	7:0								PPSLOCKED
0x1E90	INTPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1E91	T0CKIPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1E92	T1CKIPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1E93	T1GPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1E94 ... 0x1E9B	保留									
0x1E9C	T2INPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1E9D ... 0x1EA0	保留									
0x1EA1	CCP1PPS	7:0			PORT[2:0]			PIN[2:0]		
0x1EA2	CCP2PPS	7:0			PORT[2:0]			PIN[2:0]		
0x1EA3 ... 0x1EC2	保留									
0x1EC3	ADACTPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1EC4	保留									
0x1EC5	SSP1CLKPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1EC6	SSP1DATPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1EC7	SSP1SSPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1EC8 ... 0x1ECA	保留									
0x1ECB	RX1PPS	7:0			PORT[2:0]			PIN[2:0]		
0x1ECC	CK1PPS	7:0			PORT[2:0]			PIN[2:0]		
0x1ECD ... 0x1F0F	保留									
0x1F10	RA0PPS	7:0			RA0PPS[5:0]					
0x1F11	RA1PPS	7:0			RA1PPS[5:0]					
0x1F12	RA2PPS	7:0			RA2PPS[5:0]					
0x1F13	保留									
0x1F14	RA4PPS	7:0			RA4PPS[5:0]					
0x1F15	RA5PPS	7:0			RA5PPS[5:0]					
0x1F16 ... 0x1F1F	保留									
0x1F20	RC0PPS	7:0			RC0PPS[5:0]					
0x1F21	RC1PPS	7:0			RC1PPS[5:0]					
0x1F22	RC2PPS	7:0			RC2PPS[5:0]					
0x1F23	RC3PPS	7:0			RC3PPS[5:0]					
0x1F24	RC4PPS	7:0			RC4PPS[5:0]					
0x1F25	RC5PPS	7:0			RC5PPS[5:0]					

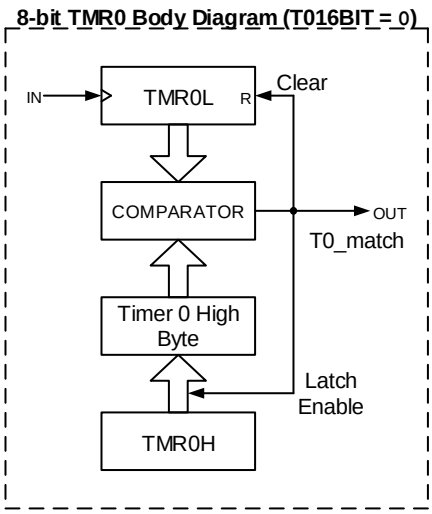
17. TMR0——Timer0 模块

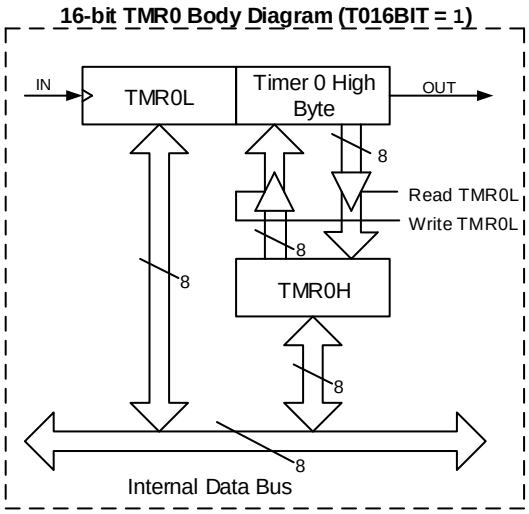
Timer0 模块具有以下特性：

- 带有可编程周期的 8 位定时器
- 16 位定时器
- 可选的时钟源
- 同步和异步操作
- 可编程预分频器（独立于看门狗定时器）
- 可编程后分频器
- 匹配或溢出时中断
- （通过 PPS）在 I/O 引脚上输出或输出至其他外设
- 可在休眠期间工作

图 17-1. Timer0 框图







17.1. Timer0 工作原理

Timer0 可用作 8 位或 16 位定时器，具体模式通过 MD16 位来选择。

17.1.1. 8 位模式

在此模式下，Timer0 在所选时钟源的上升沿递增。时钟输入上的预分频器提供几个预分频选项（见预分频比控制位 CKPS）。在该模式下（如图 17-1 所示），保留 TMR0H 的缓冲版本。

在每个所选时钟源的周期，该值都会与 TMR0L 的值进行比较。当两个值匹配时，发生以下事件：

- 复位 TMR0L
- TMR0H 的内容复制到 TMR0H 缓冲区以便进行下一次比较

17.1.2. 16 位模式

在此模式下，Timer0 在所选时钟源的上升沿递增。时钟输入上的预分频器提供几个预分频选项（见预分频比控制位 CKPS）。在该模式下，TMR0H:TMR0L 构成 16 位定时器值。如图 17-1 所示，读写 TMR0H 寄存器都会经过缓冲。在读 TMR0L 寄存器时使用 Timer0 高字节的内容更新 TMR0H 寄存器。同样，写入 TMR0L 寄存器时会使 TMR0H 寄存器值传输到 Timer0 高字节。

这种缓冲可以同时完成 Timer0 全部 16 位的读写操作。Timer0 在递增至超过 `0xFFFF` 时会计满返回到 `0x0000`。这样，定时器便可以自由运行。在 16 位模式下工作时，可读取 Timer0 值但不能写入。

17.2. 时钟选择

Timer0 具有多种时钟源选项、同步/异步工作选项以及一个可编程的预分频器。CS 位用于选择 Timer0 的时钟源。

17.2.1. 同步模式

当 ASYNC 位清零时，Timer0 时钟与系统时钟（$F_{OSC}/4$）同步。当在同步模式下工作时，Timer0 时钟频率无法超过 $F_{OSC}/4$。在休眠模式期间，系统时钟不可用，并且 Timer0 无法工作。

17.2.2. 异步模式

当 ASYNC 位置 1 时，Timer0 在输入源（或预分频器的输出，如果使用预分频器的话）的每个上升沿递增。只要选择的时钟源在休眠期间运行，异步模式就允许 Timer0 在休眠模式期间继续运行。

17.2.3. 可编程预分频器

Timer0 有 16 个可编程的输入预分频比选项，范围从 1:1 到 1:32768。预分频值可通过 CKPS 位进行选择。预分频器计数器不可直接读写。发生以下事件时，预分频器计数器会清零：

- 对 TMR0L 寄存器进行写操作
- 对 T0CON0 或 T0CON1 寄存器进行写操作
- 任何器件复位

17.2.4. 可编程后分频器

Timer0 有 16 个可编程的输出后分频比选项，范围从 1:1 到 1:16。后分频值可通过 OUTPS 位进行选择。后分频器按照选定的比率将 Timer0 的输出分频。后分频器计数器不可直接读写。发生以下事件时，后分频器计数器会清零：

- 对 TMR0L 寄存器进行写操作
- 对 T0CON0 或 T0CON1 寄存器进行写操作
- 任何器件复位

17.3. Timer0 输出和中断

17.3.1. Timer0 输出

在 8 位模式下，TMR0_out 会在 TMR0L 和 TMR0H 每次匹配时翻转；在 16 位模式下，TMR0_out 会在 TMR0H:TMR0L 计满返回时翻转。如果使用输出后分频器，则输出会按所选比例进行缩放。可通过 RxyPPS 输出选择寄存器将 Timer0 输出输送到 I/O 引脚，也可在内部将其输送到多个独立于内核的外设。Timer0 输出可使用软件通过 OUT 输出位进行监视。

重要：在 8 位模式下，当 PR0 = 0（装入 0 或复位为 0）时，TMR0 输出保持高电平，并且不产生中断。

17.3.2. Timer0 中断

Timer0 中断标志（TMR0IF）位在 TMR0_out 翻转时置 1。如果允许 Timer0 中断（TMR0IE），则 CPU 将在 TMR0IF 位置 1 时中断。当后分频比位（T0OUTPS）设置为 1:1 操作（无分频）时，T0IF 标志位将在每次发生 TMR0 匹配或计满返回时置 1。通常，TMR0IF 标志位将在每发生 T0OUTPS +1 次匹配或计满返回时置 1。

17.3.3. Timer0 示例

Timer0 配置：

- Timer0 模式 = 16 位
- 时钟源 = $F_{OSC}/4$（250 kHz）
- 同步操作
- 预分频比 = 1:1
- 后分频比 = 1:2（T0OUTPS = 1）

在这种情况下，TMR0H:TMR0L 每计满返回两次，TMR0_out 就翻转一次。
即，(0xFFFF) * 2 * (1/250 kHz) = 524.28 ms

17.4. 休眠期间的操作

同步工作时，如果器件进入休眠模式，Timer0 将暂停。异步工作且所选时钟源有效时，Timer0 将继续递增计数，并会在允许 Timer0 中断的情况下将器件从休眠模式唤醒。

17.5. 寄存器定义：Timer0 控制

17.5.1. T0CON0

名称： T0CON0
偏移量： 0x059E

Timer0 控制寄存器 0

位	7	6	5	4	3	2	1	0
	EN		OUT	MD16	OUTPS[3:0]			
访问	R/W		R	R/W	R/W	R/W	R/W	R/W
复位	0		0	0	0	0	0	0

Bit 7 – EN TMR0 使能

值	说明
1	模块已使能并且正在工作
0	模块已禁止

Bit 5 – OUT TMR0 输出

Bit 4 – MD16 16 位定时器操作选择

值	说明
1	TMR0 是 16 位定时器
0	TMR0 是 8 位定时器

Bit 3:0 – OUTPS[3:0] TMR0 输出后分频比（分频比）选择

值	说明
1111	1:16 后分频比
1110	1:15 后分频比
1101	1:14 后分频比
1100	1:13 后分频比
1011	1:12 后分频比
1010	1:11 后分频比
1001	1:10 后分频比
1000	1:9 后分频比
0111	1:8 后分频比
0110	1:7 后分频比
0101	1:6 后分频比
0100	1:5 后分频比
0011	1:4 后分频比
0010	1:3 后分频比
0001	1:2 后分频比
0000	1:1 后分频比

17.5.2. T0CON1

名称： T0CON1
偏移量： 0x059F

Timer0 控制寄存器 1

位	7	6	5	4	3	2	1	0
	CS[2:0]			ASYNC	CKPS[3:0]			
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7:5 – CS[2:0] Timer0 时钟源选择

表 17-1. Timer0 时钟源选择

T0CS	时钟源
111–110	保留
101	MFINTOSC（500 kHz）
100	LFINTOSC
011	HFINTOSC
010	$F_{OSC}/4$
001	通过 T0CKIPPS 选择的引脚（反相）
000	通过 T0CKIPPS 选择的引脚（未反相）

Bit 4 – ASYNC TMR0 输入异步使能

值	说明
1	TMR0 计数器输入与系统时钟不同步
0	TMR0 计数器输入与 Fosc/4 同步

Bit 3:0 – CKPS[3:0] 预分频比选择

值	说明
1111	1:32768
1110	1:16384
1101	1:8192
1100	1:4096
1011	1:2048
1010	1:1024
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

17.5.3. TMR0H

名称： TMR0H
偏移量： 0x059D

Timer0 周期/计数高字节寄存器

位	7	6	5	4	3	2	1	0
	TMR0H[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	1	1	1	1	1	1	1	1

Bit 7:0 – TMR0H[7:0] TMR0 计数器高位

值	条件	说明
xxxxxxxx	MD16 = 0	8 位 Timer0 周期值。达到该值时，TMR0L 继续从 0 开始计数。
xxxxxxxx	MD16 = 1	16 位 Timer0 最高有效字节

17.5.4. TMR0L

名称： TMR0L
偏移量： 0x059C

Timer0 周期/计数低字节寄存器

位	7	6	5	4	3	2	1	0
	TMR0L[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7:0 – TMR0L[7:0] TMR0 计数器低位

值	条件	说明
xxxxxxxx	MD16 = 0	8 位 Timer0 计数器位
xxxxxxxx	MD16 = 1	16 位 Timer0 最低有效字节

17.6. 寄存器汇总——Timer0

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x059B	保留									
0x059C	TMR0L	7:0	TMR0L[7:0]							
0x059D	TMR0H	7:0	TMR0H[7:0]							
0x059E	T0CON0	7:0	EN		OUT	MD16	OUTPS[3:0]			
0x059F	T0CON1	7:0	CS[2:0]			ASYNC	CKPS[3:0]			

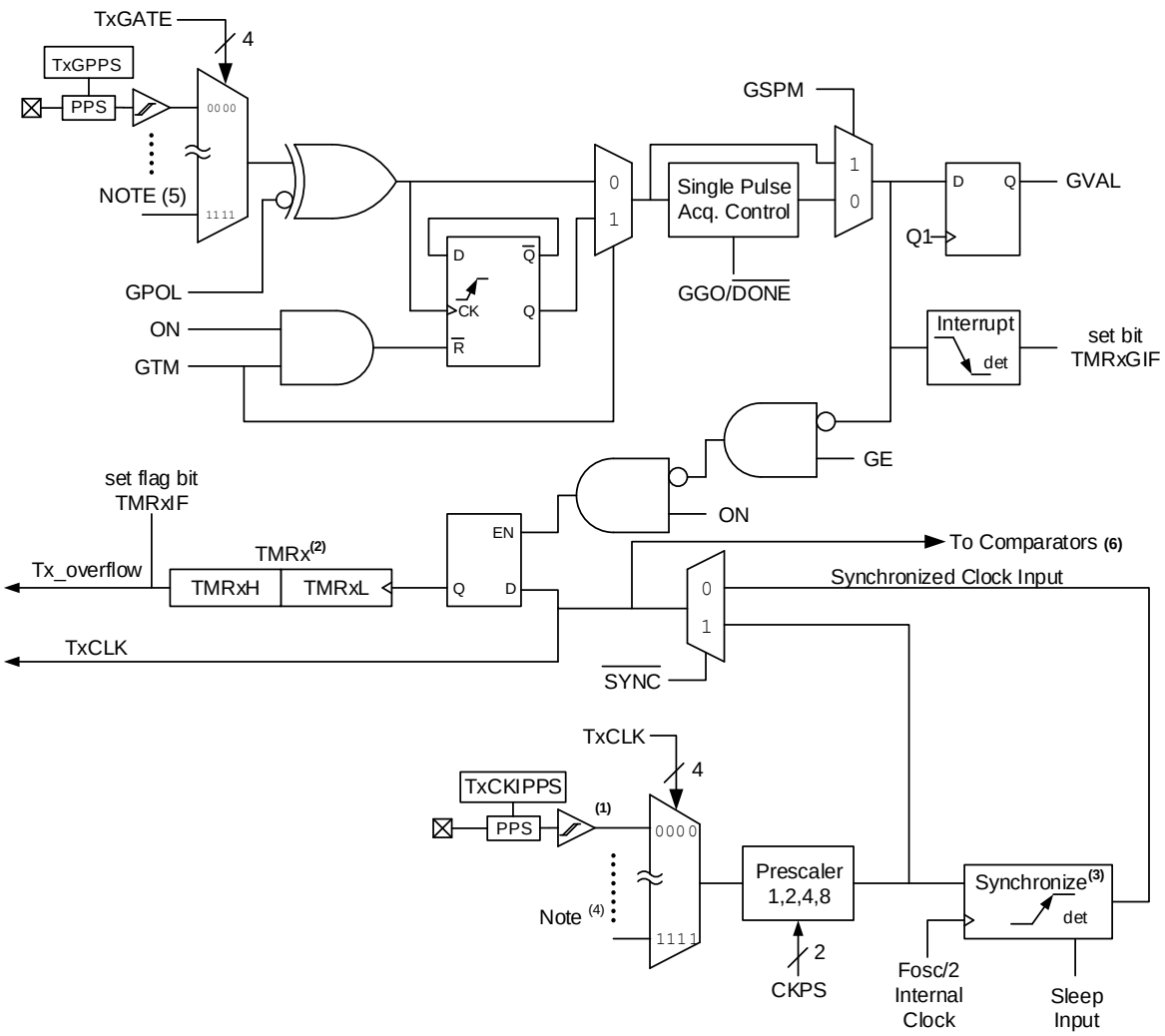
18. TMR1——带门控的 Timer1 模块

Timer1 模块是 16 位定时器/计数器，具有以下特性：

- 16 位定时器/计数器寄存器对（TMRxH:TMRxL）
- 可编程的内部或外部时钟源
- 2 位预分频器
- 用于可选比较器同步的时钟源
- 多个 Timer1 门控（计数使能）源
- 溢出时中断
- 溢出时唤醒（仅限外部时钟，异步模式）
- 16 位读/写操作
- 用于 CCP 模块的捕捉/比较功能的时基
- 特殊事件触发器（带 CCP）
- 可选门控信号源极性
- 门控翻转模式
- 门控单脉冲模式
- 门控值状态
- 门控事件中断

重要：涉及模块 Timer1 的内容同样适用于该器件上所有奇数编号的定时器。

图 18-1. Timer1 框图



注：

1. 该信号来自 Timer1 PPS 寄存器选择的引脚。
2. TMRx 寄存器在上升沿递增。
3. 休眠时不进行同步。
4. 关于时钟源选择，请参见 TxCLK。
5. 关于门控源选择，请参见 TxGATE。
6. 不得将同步比较器输出与同步输入时钟一起使用。

18.1. Timer1 工作状态

Timer1 模块是 16 位递增计数器，可通过 TMRx 寄存器访问。写 TMRx 会直接更新计数器。与内部时钟源一起使用时，该模块为定时器，在每个指令周期递增 1。与外部时钟源一起使用时，该模块可用作定时器或计数器，在外部时钟源的每个选定边沿递增 1。

配置 ON 和 GE 位可使能 Timer1。表 18-1 列出了 Timer1 的使能选项。

表 18-1. Timer1 的使能选项

ON	GE	Timer1 工作状态
1	1	使能计数
1	0	始终开启
0	1	关闭
0	0	关闭

18.2. 时钟源选择

CS 位用于选择 Timer1 的时钟源。这些位可用于选择多种同步和异步时钟源。

18.2.1. 内部时钟源

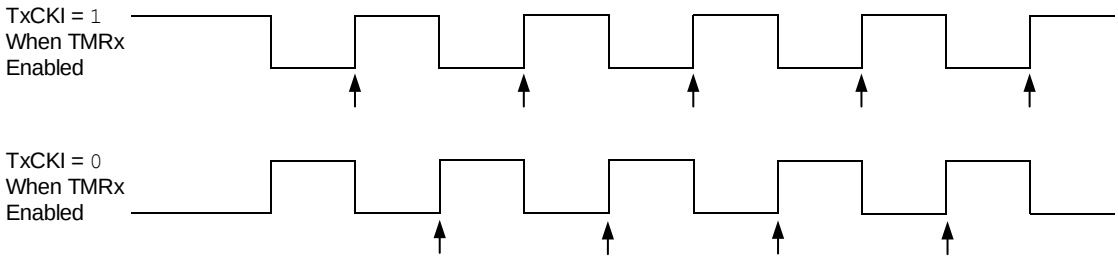
当选择内部时钟源时，TMRx 寄存器的递增频率将为 F_{OSC} 的整数倍（取决于 Timer1 预分频器）。

选择 F_{OSC} 内部时钟源时，TMRx 寄存器的值将在每个指令时钟周期中递增 4 次。由于这个原因，在读取 TMRx 值时，分辨率将会出现两个 LSb 的误差。为了利用 Timer1 的全部分辨率，必须使用异步输入信号来对 Timer1 时钟输入进行门控。

重要：在计数器模式下，在发生以下任何一个或多个条件时必须先经过一个下降沿，计数器才可以在上升沿进行第一次递增计数：

- 上电复位后使能 Timer1
- 写 TMRxH 或 TMRxL
- 禁止 Timer1
- TxCKI 为高电平时禁止 Timer1（ON = 0），TxCKI 为低电平时使能 Timer1（ON = 1）。请参见下图。

图 18-2. Timer1 递增边沿



注：

1. 箭头指示计数器递增。
2. 在计数器模式下，必须先经过一个下降沿，计数器才可以在时钟上升沿进行第一次递增计数。

18.2.2. 外部时钟源

选择外部时钟源时，TMRx 模块可作为定时器或计数器。当使能计数模式时，Timer1 在外部时钟输入 TxCKIPPS 引脚的上升沿递增。该外部时钟源既可以与系统时钟同步，也可以异步运行。

18.3. Timer1 预分频器

Timer1 有 4 个预分频比选项，允许对时钟输入进行 1、2、4 或 8 分频。CKPS 位控制预分频器计数器。对预分频器计数器不能直接进行读写操作；但是，通过写入 TMRx 可将预分频器计数器清零。

18.4. 异步计数器模式下的 Timer1 操作

SYNC 控制位置 1 时，外部时钟输入不同步。定时器异步于内部相位时钟递增计数。如果选择了外部时钟源，在休眠期间定时器将继续运行，并在溢出时产生中断以唤醒处理器。但是，用软件对定时器进行读/写操作时，要特别当心。

重要：当从同步切换到异步操作时，可能会跳过一次递增。当从异步切换到同步操作时，可能会产生一次额外递增。

18.4.1. 在异步计数器模式下读写 TMRx

当定时器采用外部异步时钟运行时，可确保对 TMRxH 或 TMRxL 的读操作为有效读操作（由硬件实现）。但用户必须注意的是，通过读两个 8 位值来读取 16 位定时器本身就会产生某些问题，这是因为 TMRxL:TMRxH 可能在两次读操作之间产生进位。

对于写操作，建议用户直接停止定时器，然后写入所需的值。如果定时器寄存器正进行递增计数，对定时器寄存器进行写操作可能会导致写争用，这可能在 TMRxH:TMRxL 寄存器对中产生不可预测的值。

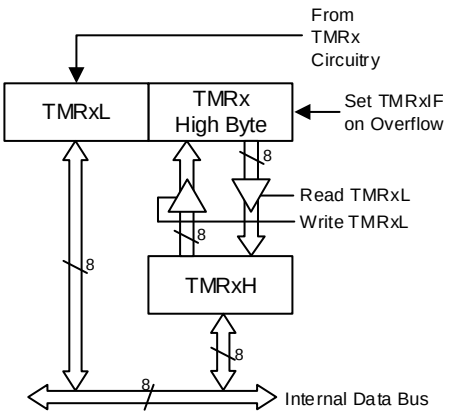
18.5. Timer1 16 位读/写模式

Timer1 可配置为同时对 8 位 TMRxL 和 TMRxH 寄存器读/写所有 16 位数据。通过将 RD16 位置 1 可使能 16 位读/写操作。为实现此功能，TMRxH 寄存器值将映射到称为 TMRxH 缓冲寄存器的缓冲寄存器。在 16 位模式下，TMRxH 寄存器不能直接读写，所有读写操作都是通过使用该 TMRxH 缓冲寄存器来实现的。

当请求读取 TMRxL 寄存器时，TMRxH 寄存器的值会同时装入 TMRxH 缓冲寄存器中。当请求读取 TMRxH 寄存器时，该值由 TMRxH 缓冲寄存器提供。这使用户能够精确而及时地从单个时间实例中读取所有 16 位 Timer1 值（更多详细信息，请参见图 18-3）。相比之下，当不处于 16 位模式时，用户必须单独读取每个寄存器，并确定这些值是否已因读操作之间可能发生的计满返回而变为无效。

当请求写入 TMRxL 寄存器时，会同时使用 TMRxH 寄存器的内容更新 TMRxH 缓冲寄存器。在执行 TMRxL 寄存器写请求之前，必须将 TMRxH 的值预先装入 TMRxH 缓冲寄存器中。这样用户就可以将所有 16 位数据一次写入 TMRx 寄存器。任何直接写入 TMRxH 的请求都不会将 Timer1 预分频值清零。预分频值只能通过对 TMRxL 寄存器的写请求清零。

图 18-3. Timer1 16 位读/写模式框图



18.6. Timer1 门控

Timer1 可配置为自由计数或用 Timer1 门控电路使能和禁止计数。这也称为 Timer1 门控使能。Timer1 门控也可通过多个可选信号源来驱动。

18.6.1. Timer1 门控使能

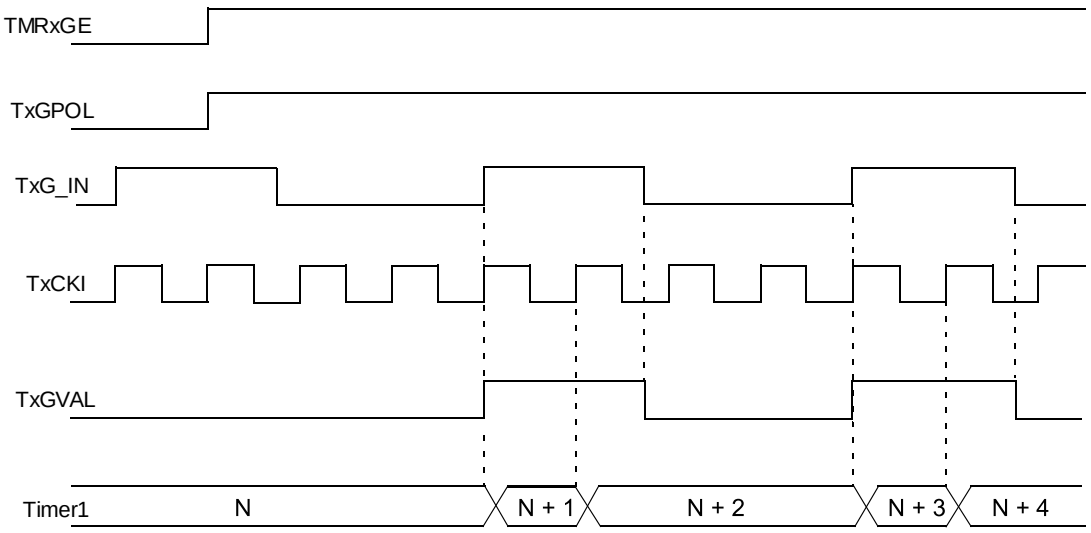
通过将 GE 位置 1 使能 Timer1 门控使能模式。使用 GPOL 位来配置 Timer1 门控使能模式的极性。

使能 Timer1 门控使能模式时，Timer1 将在 Timer1 时钟源的上升沿递增。当 Timer1 门控信号无效时，定时器不会发生递增并且将保持当前计数。禁止使能模式时，不会发生递增，Timer1 将保持当前计数。时序详细信息请参见图 18-4。

表 18-2. Timer1 门控使能选择

TMRxCLK	GPOL	TxG	Timer1 工作状态
↑	1	1	计数
↑	1	0	保持计数
↑	0	1	保持计数
↑	0	0	计数

图 18-4. Timer1 门控使能模式



18.6.2. Timer1 门控信号源选择

通过 GSS 位选择 Timer1 的门控源。门控源极性的选择由 GPOL 位控制。

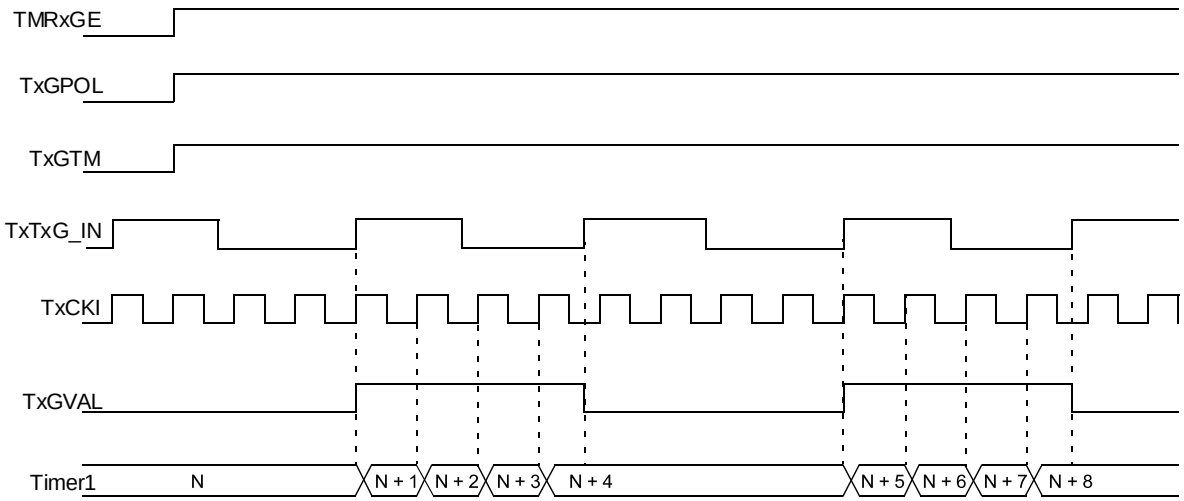
18.6.3. Timer1 门控翻转模式

使能 Timer1 门控翻转模式时，可测量 Timer1 门控信号整个周期的长度，而不是单电平脉冲的持续时间。Timer1 门控源经由一个单稳态触发器输送到 Timer1，该单稳态触发器在信号的每个递增边沿改变状态。有关时序的详细信息，请参见下图。

通过将 GTM 位置 1 使能 Timer1 门控翻转模式。GTM 位清零时，将清零触发器并保持清零。该模式对于控制要计数的边沿是必需的。

重要：使能翻转模式的同时更改门控信号的极性可能会导致操作不确定。

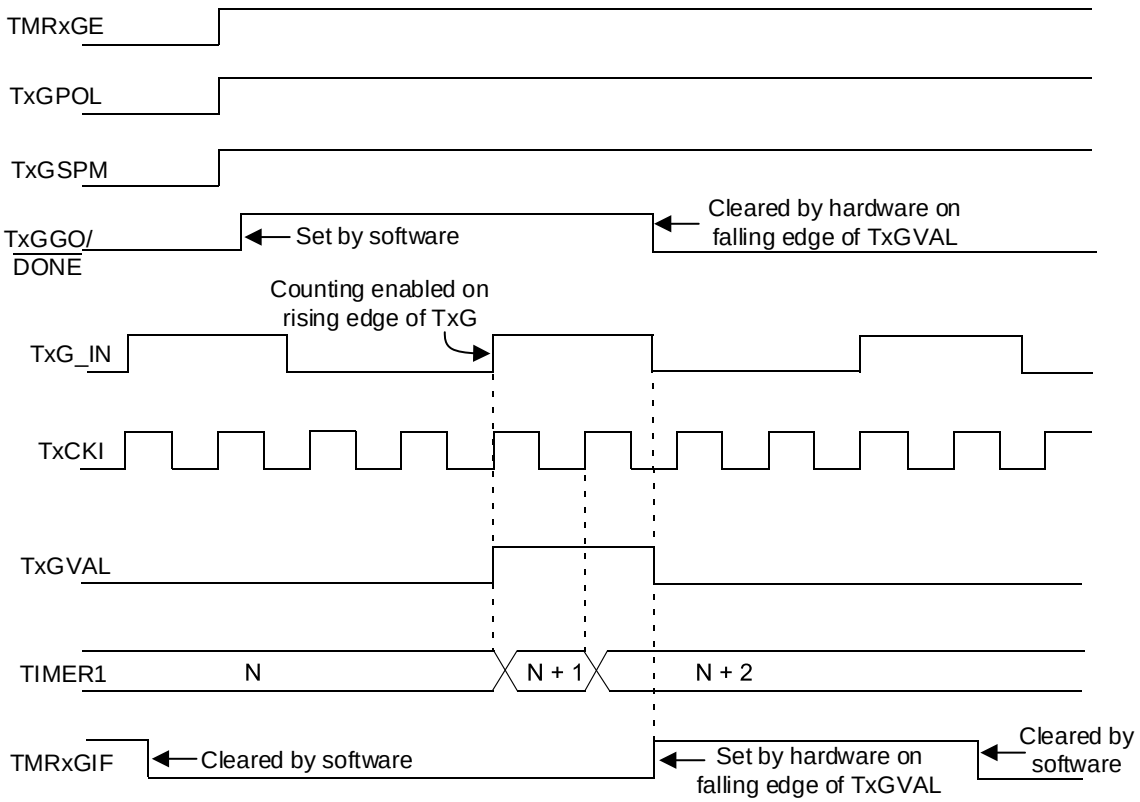
图 18-5. Timer1 门控翻转模式



18.6.4. Timer1 门控单脉冲模式

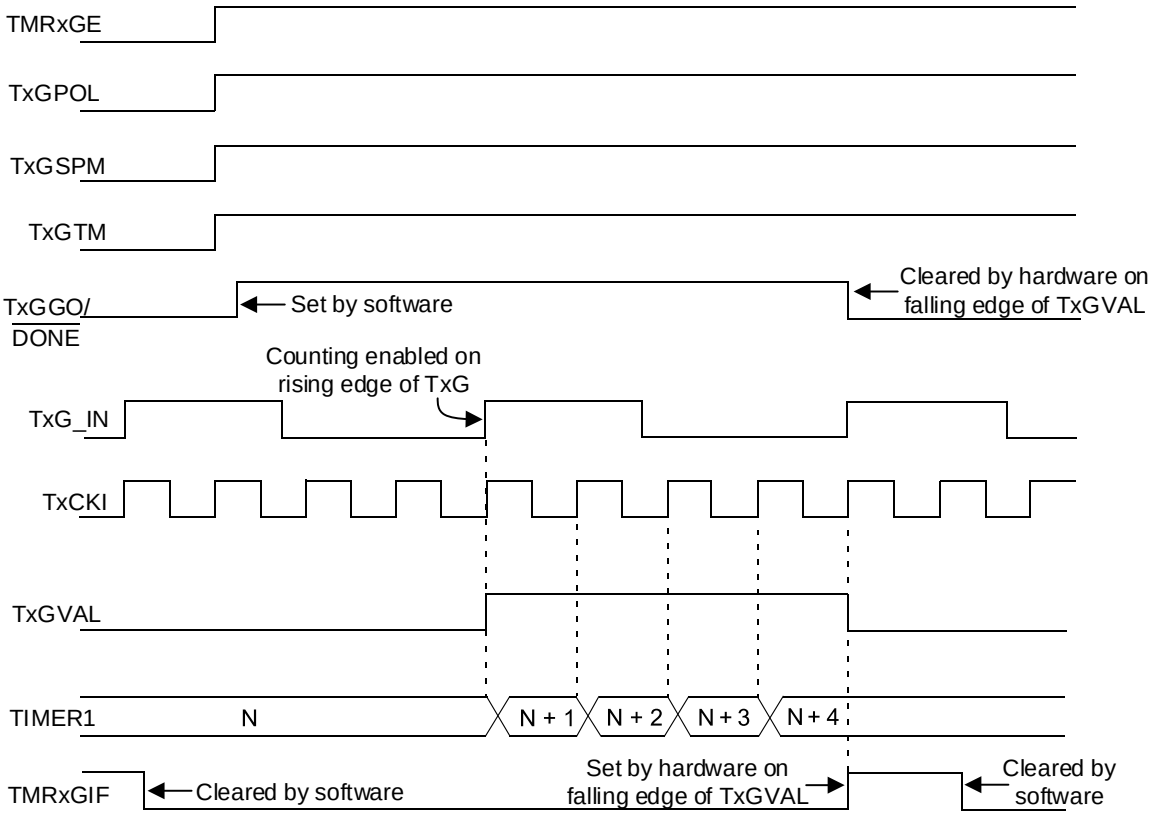
使能了 Timer1 门控单脉冲模式时，可以捕捉单脉冲门控事件。首先，通过将 GSPM 位置 1 来使能 Timer1 门控单脉冲模式。然后，必须将 GGO/DONE 置 1。Timer1 将在下一个递增沿完全使能。在脉冲的下一个后沿，GGO/$\overline{\text{DONE}}$ 位将自动清零。GGO/$\overline{\text{DONE}}$ 位再次由软件置 1 前，其他门控事件无法递增 Timer1。

图 18-6. Timer1 门控单脉冲模式



清零 GSPM 位也将清零 GGO/$\overline{\text{DONE}}$ 位。有关时序的详细信息，请参见下图。同时使能翻转模式和单脉冲模式将允许两种模式协同工作。这样就可以测量 Timer1 门控源的周期时间。有关时序的详细信息，请参见下图。

图 18-7. Timer1 门控单脉冲和翻转组合模式



18.6.5. Timer1 门控值状态

使用 Timer1 门控值状态时，可读取门控值的最新电平。该值存储在 TxGCON 寄存器的 GVAL 位中。即使 Timer1 门控未使能（GE 位清零），GVAL 位也是有效的。

18.6.6. Timer1 门控事件中断

允许 Timer1 门控事件中断时，可在门控事件完成时产生一个中断。出现 GVAL 的下降沿时，TMRxGIF 标志位将置 1。如果相应 PIE 寄存器中的 TMRxGIE 位置 1，则会识别出一个中断。

即使 Timer1 门控未使能（GE 位清零），TMRxGIF 标志位也是有效的。

18.7. Timer1 中断

TMRx 寄存器递增到 FFFFh，然后计满返回到 0000h。当 TMRx 计满返回时，PIRx 寄存器的 TMRx 中断标志（TMRxIF）位将置 1。为允许计满返回时的中断，必须将以下位置 1：

- ON TxCON 寄存器的位
- PIEx 寄存器的 TMRxIE 位
- 必须允许全局中断

在中断服务程序中以任务形式将 TMRxIF 位清零，以清除中断。

重要：在允许中断前，需将 TMRx 寄存器以及 TMRxIF 位清零。

18.8. 休眠期间的 Timer1 操作

只有在配置为异步计数器时，Timer1 才能在休眠模式下工作。在该模式下，可使用多种时钟源使计数器递增计数。要设置定时器以唤醒器件：

- 必须将 ON 位置 1
- 必须将 PIEx 寄存器的 TMRxIE 位置 1
- 必须允许全局中断
- 必须将 SYNC 位置 1
- 将 TxCLK 寄存器配置为使用 F_{OSC} 和 $F_{OSC}/4$ 以外的任何时钟源

器件将在溢出时被唤醒并执行下一条指令。如果允许全局中断，器件将调用 IRS。不管 SYNC 位是否置 1，辅助振荡器都将在休眠模式下继续工作。

18.9. CCP 捕捉/比较时基

当工作在捕捉或比较模式下时，CCP 模块使用 TMRx 作为时基。在捕捉模式下，当发生捕捉事件时，TMRx 中的值被复制到 CCPRx 寄存器中。在比较模式下，当 CCPRx 寄存器中的值与 TMRx 中的值相匹配时触发事件。该事件可以是特殊事件触发信号。

18.10. CCP 特殊事件触发器

当任意 CCP 配置为触发特殊事件时，触发信号将清零 TMRx 寄存器。该特殊事件不会引起 Timer1 中断。CCP 模块仍可配置为产生 CCP 中断。在这种工作模式下，CCPRx 寄存器变成了 Timer1 的周期寄存器。Timer1 必须进行同步，并且必须选择 $F_{OSC}/4$ 作为时钟源，以利用特殊事件触发信号。Timer1 的异步操作会导致错过特殊事件触发信号。如果对 TMRxH 或 TMRxL 的写操作与来自 CCP 的特殊事件触发信号同时发生，则写操作优先。

18.11. 寄存器定义：Timer1 控制

下表列出了定时器寄存器的长位名称前缀，其中“x”表示定时器实例编号。更多信息，请参见“**寄存器和位命名约定**”一章中的“**长位名称**”一节。

表 18-3. Timer1 寄存器位名称前缀

外设	位名称前缀
Timer1	T1

18.11.1. TxCON

名称： TxCON
偏移量： 0x020E

定时器控制寄存器

位	7	6	5	4	3	2	1	0
			CKPS[1:0]			$\overline{\text{SYNC}}$	RD16	ON
访问			R/W	R/W		R/W	R/W	R/W
复位			0	0		0	0	0

Bit 5:4 – CKPS[1:0] 定时器输入时钟预分频比选择

复位状态： POR/BOR = 00
所有其他复位 = uu

值	说明
11	1:8 预分频值
10	1:4 预分频值
01	1:2 预分频值
00	1:1 预分频值

Bit 2 – $\overline{\text{SYNC}}$ 定时器外部时钟输入同步控制

复位状态： POR/BOR = 0
所有其他复位 = u

值	条件	说明
x	CS = $F_{OSC}/4$ 或 F_{OSC}	忽略该位。定时器按原样使用传入时钟。
1	所有其他时钟源	不同步外部时钟输入
0	所有其他时钟源	将外部时钟输入与系统时钟同步

Bit 1 – RD16 16 位读/写模式使能

复位状态： POR/BOR = 0
所有其他复位 = u

值	说明
1	使能通过一次 16 位操作读/写定时器的寄存器
0	使能通过两次 8 位操作读/写定时器的寄存器

Bit 0 – ON 定时器使能

复位状态： POR/BOR = 0
所有其他复位 = u

值	说明
1	使能定时器
0	禁止定时器

18.11.2. TxGCON

名称： TxGCON
偏移量： 0x020F

定时器门控寄存器

位	7	6	5	4	3	2	1	0
	GE	GPOL	GTM	GSPM	GGO/$\overline{\text{DONE}}$	GVAL		
访问	R/W	R/W	R/W	R/W	R/W	R		
复位	0	0	0	0	0	x		

Bit 7 – GE 定时器门控使能

复位状态： POR/BOR = 0
所有其他复位 = u

值	条件	说明
1	ON =1	定时器计数由定时器门控功能控制
0	ON =1	定时器始终计数
x	ON =0	忽略该位

Bit 6 – GPOL 定时器门控极性

复位状态： POR/BOR = 0
所有其他复位 = u

值	说明
1	定时器门控为高电平有效（当门控信号为高电平时定时器计数）
0	定时器门控为低电平有效（当门控信号为低电平时定时器计数）

Bit 5 – GTM 定时器门控翻转模式
使能翻转模式时，定时器门控触发器在每个上升沿翻转。

复位状态： POR/BOR = 0
所有其他复位 = u

值	说明
1	使能定时器门控翻转模式
0	禁止定时器门控翻转模式并清除翻转触发器

Bit 4 – GSPM 定时器门控单脉冲模式

复位状态： POR/BOR = 0
所有其他复位 = u

值	说明
1	使能定时器门控单脉冲模式，并在使用该模式时控制定时器门控
0	禁止定时器门控单脉冲模式

Bit 3 – GGO/$\overline{\text{DONE}}$ 定时器门控单脉冲采集状态
当 TxGSPM 位清零时，该位自动清零。

复位状态： POR/BOR = 0
所有其他复位 = u

值	说明
1	定时器门控单脉冲采集就绪，正在等待边沿出现
0	定时器门控单脉冲采集已经结束或尚未开始

Bit 2 – GVAL 定时器门控当前状态

指示可提供给 TMRxH:TMRxL 的定时器门控信号的当前状态
不受定时器门控使能（GE）位的影响

18.11.3. TxCLK

名称： TxCLK
偏移量： 0x0211

定时器时钟源选择寄存器

位	7	6	5	4	3	2	1	0
				CS[4:0]				
访问				R/W	R/W	R/W	R/W	R/W
复位				0	0	0	0	0

Bit 4:0 – CS[4:0] 定时器时钟源选择

表 18-4. 定时器时钟源

CS	时钟源
11111-01001	保留
01000	TMR0_OUT
00111	MFINTOSC（32 kHz）
00110	MFINTOSC（500 kHz）
00101	SFINTOSC（1 MHz）
00100	LFINTOSC
00011	HFINTOSC
00010	F_{OSC}
00001	$F_{OSC}/4$
00000	通过 T1CKIPPS 选择的引脚

复位状态： POR/BOR = 00000
所有其他复位 = uuuuu

18.11.4. TxGATE

名称： TxGATE
偏移量： 0x0210

定时器门控源选择寄存器

位	7	6	5	4	3	2	1	0
				GSS[4:0]				
访问				R/W	R/W	R/W	R/W	R/W
复位				0	0	0	0	0

Bit 4:0 – GSS[4:0] 定时器门控源选择

表 18-5. 定时器门控源

GSS	门控源
11111-00111	保留
00110	PWM4_OUT
00101	PWM3_OUT
00100	CCP2_OUT
00011	CCP1_OUT
00010	TMR2_Postscaled_OUT
00001	TMR0_OUT
00000	通过 T1GPPS 选择的引脚

18.11.5. TMRx

名称： TMRx
偏移量： 0x020C

定时器寄存器

位	15	14	13	12	11	10	9	8
	TMRx[15:8]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

位	7	6	5	4	3	2	1	0
	TMRx[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 15:0 – TMRx[15:0] 定时器寄存器值

复位状态： POR/BOR = 0000000000000000
所有其他复位 = uuuuuuuuuuuuuuuu

注：该多字节寄存器中的各个字节可使用以下寄存器名称进行访问：

- TMRxH：访问高字节 TMRx[15:8]
- TMRxL：访问低字节 TMRx[7:0]

18.12. 寄存器汇总——Timer1

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x020B	保留									
0x020C	TMR1	7:0	TMR1[7:0]							
		15:8	TMR1[15:8]							
0x020E	T1CON	7:0			CKPS[1:0]			$\overline{\text{SYNC}}$	RD16	ON
0x020F	T1GCON	7:0	GE	GPOL	GTM	GSPM	GGO/$\overline{\text{DONE}}$	GVAL		
0x0210	T1GATE	7:0				GSS[4:0]				
0x0211	T1CLK	7:0				CS[4:0]				

19. TMR2——Timer2 模块

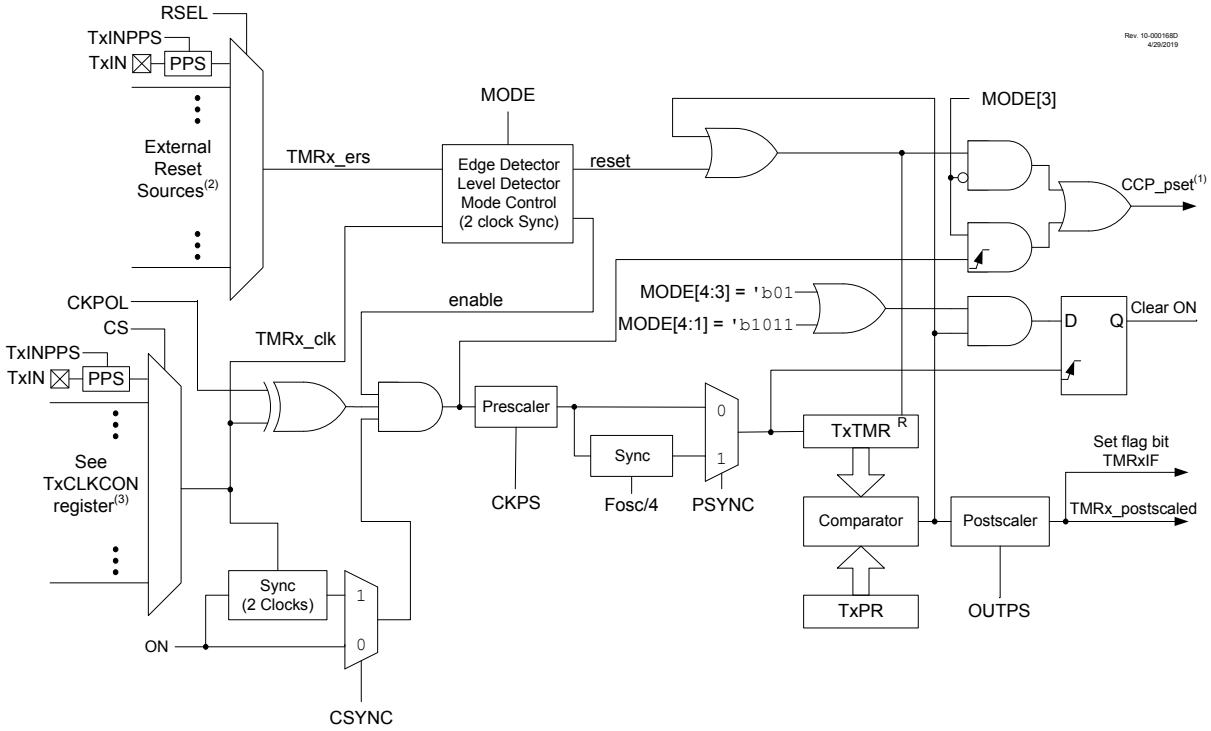
Timer2 模块是 8 位定时器，具有以下特性：

- 8 位定时器和周期寄存器
- 可读写
- 可软件编程的预分频比（1:1 至 1:128）
- 可软件编程的后分频比（1:1 至 1:16）
- T2TMR 与 T2PR 匹配时产生中断
- 单触发操作
- 全异步操作
- 包含硬件限制定时器（HLT）
- 备用时钟源
- 外部定时器复位信号源
- 可配置的定时器复位操作

Timer2 框图请参见下图。

重要：涉及模块 Timer2 的内容同样适用于该器件上的所有偶数编号的定时器（Timer2 和 Timer4 等）。

图 19-1. Timer2 与硬件限制定时器（HLT）框图



注：

1. PWM 模式下用于 PWM 脉冲触发的 CCP 外设信号。
2. 关于外部复位源，请参见 RSEL。
3. 关于时钟源选择，请参见 CS。

19.1. Timer2 工作原理

Timer2 具有以下三种主要工作模式：

- 自由运行周期
- 单次
- 单稳态

每种工作模式下都有多种启动、停止和复位选项。 表 19-1 列出了这些选项。

在所有模式下，T2TMR 计数寄存器都在来自可编程预分频器的时钟信号上升沿递增。当 T2TMR 等于 T2PR 时，会向后分频器计数器输出高电平信号。T2TMR 在下一个时钟输入清零。

来自硬件的外部信号也可以配置为对定时器操作进行门控或强制 T2TMR 计数复位。在门控模式下，计数器在禁止门控时停止，并在使能门控时恢复计数。在复位模式下，T2TMR 计数在外部源的任一电平或边沿复位。

T2TMR 和 T2PR 寄存器均可直接读写。在任何器件复位时，T2TMR 寄存器都会清零，而 T2PR 寄存器则初始化为 `0xFF`。发生以下事件时，预分频器和后分频器计数器都将清零：

- 对 T2TMR 寄存器进行写操作
- 对 T2CON 寄存器进行写操作
- 任何器件复位
- 复位定时器的外部复位源事件

重要：写 T2CON 时 T2TMR 不会清零。

19.1.1. 自由运行周期模式

在每个时钟周期，T2TMR 的值都会与周期寄存器 T2PR 中的值进行比较。当二者匹配时，比较器会在下一个周期将 T2TMR 的值复位为 `0x00`，并使输出后分频器计数器递增。当后分频器计数等于 T2CON 寄存器的 OUTPS 位中的值时，TMR2_postscaled 输出上会出现一个时钟周期宽的脉冲，并且后分频器计数清零。

19.1.2. 单触发模式

单触发模式与自由运行周期模式类似，只是当 T2TMR 与 T2PR 匹配时，ON 位清零并且定时器停止工作，直到 ON 位禁止再使能后才重新开始工作。在该模式下，由于定时器在第一个周期事件时停止工作，而后分频器会在定时器重新启动时复位，因此除零以外的后分频器（OUTPS）值将被忽略。

19.1.3. 单稳态模式

单稳态模式与单触发模式类似，只是 ON 位不清零，并且定时器可以通过外部复位事件重启。

19.2. Timer2 输出

Timer2 模块的主输出是 TMR2_postscaled，每当后分频器计数器与 T2CON 寄存器的 OUTPS 位匹配时，它都会生成一个 TMR2_clk 周期脉冲。每当 T2TMR 值与 T2PR 值匹配时，后分频器都会递增。还可以选择该信号作为其他独立于内核的外设的输入。

此外，CCP 模块也会使用 Timer2 在 PWM 模式下生成脉冲。有关设置 Timer2 与 CCP 和 PWM 模块配合使用的更多详细信息，请参见**“CCP——捕捉/比较/PWM 模块”**一章中的“**PWM 概述**”一节和“**PWM 周期**”一节。

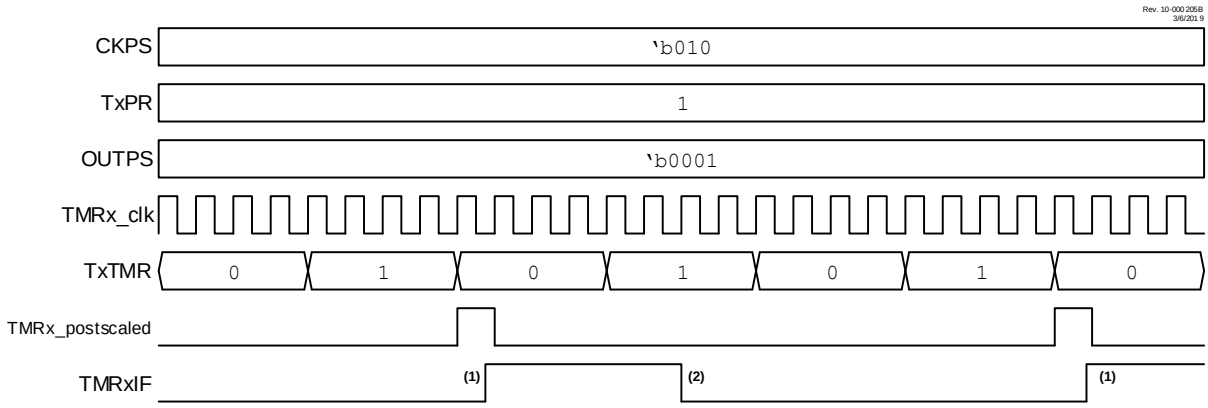
19.3. 外部复位源

除时钟源外，Timer2 也可通过外部复位源输入驱动。各定时器的外部复位输入使用相应的 TxRST 寄存器进行选择。外部复位输入可以根据使用的模式，对定时器的启动、停止以及复位进行控制。

19.4. Timer2 中断

Timer2 也可以产生器件中断。当后分频器计数器与所选后分频值（T2CON 寄存器的 OUTPS 位）匹配时产生中断。可以通过将 TMR2IE 中断允许位置 1 来允许该中断。中断时序如下图所示。

图 19-2. Timer2 预分频器、后分频器和中断时序图



Notes: 1. Setting the interrupt flag is synchronized with the instruction clock.
Synchronization may take as many as two instruction cycles.
2. Cleared by software.

19.5. PSYNC 位

将 PSYNC 位置 1 可使预分频器输出与 $F_{OSC}/4$ 同步。如果所选定时器时钟与 $F_{OSC}/4$ 异步，则读取 Timer2 计数器寄存器时需要将该位置 1。

注：要将 PSYNC 置 1，预分频器的输出需慢于 $F_{OSC}/4$。当预分频器的输出大于或等于 $F_{OSC}/4$ 时，将 PSYNC 置 1 可能导致意外结果。

19.6. CSYNC 位

默认情况下，Timer2 SFR 中的所有位均与 $F_{OSC}/4$（而非 Timer2 输入时钟）同步。因此，如果 Timer2 输入时钟与 $F_{OSC}/4$ 不同步，则 Timer2 输入时钟可能在 ON 位通过软件置 1 的同时发生切换，这可能导致计数器中出现非预期的行为和毛刺。将 CSYNC 位置 1 时会将 ON 位与 Timer2 输入时钟（而非 $F_{OSC}/4$）同步，从而解决这一问题。但由于该同步使用 TMR2 输入时钟的边沿，因此当 CSYNC 置 1 时，最多将占用一个输入时钟周期并且 Timer2 不会对该时钟周期进行计数。相反，将 CSYNC 位清零时会将 ON 位与 $F_{OSC}/4$ 同步，这不会占用任何时钟边沿，但会面临前述的毛刺风险。

19.7. 工作模式

定时器的模式由 MODE 位控制。边沿触发模式要求两个外部触发信号之间间隔 6 个定时器时钟周期。电平触发模式要求触发电平之间至少间隔 3 个定时器时钟周期。处于调试模式时，外部触发信号会被忽略。

表 19-1. 工作模式表

模式	MODE		输出操作	操作	定时器控制		
	[4:3]	[2:0]			启动	复位	停止
自由运行周期	00	000	周期脉冲	软件门控（图 19-3）	ON = 1	—	ON = 0
		001		硬件门控，高电平有效（图 19-4）	ON = 1 且 TMRx_ers = 1	—	ON = 0 或 TMRx_ers = 0
		010		硬件门控，低电平有效	ON = 1 且 TMRx_ers = 0	—	ON = 0 或 TMRx_ers = 1
		011	周期脉冲和硬件复位	上升沿或下降沿复位	ON = 1	TMRx_ers ↕	ON = 0
		100		上升沿复位（图 19-5）		TMRx_ers ↑	
		101		下降沿复位		TMRx_ers ↓	
		110		低电平复位		TMRx_ers = 0	ON = 0 或 TMRx_ers = 0
		111		高电平复位（图 19-6）		TMRx_ers = 1	ON = 0 或 TMRx_ers = 1
单触发	01	000	单触发	软件启动（图 19-7）	ON = 1	—	ON = 0 或 TxTMR = TxPR 后的下一个时钟（注 2）
		001	边沿触发启动（注 1）	上升沿启动（图 19-8）	ON = 1 且 TMRx_ers ↑	—	
		010		下降沿启动	ON = 1 且 TMRx_ers ↓	—	
		011		任意边沿启动	ON = 1 且 TMRx_ers ↕	—	
		100	边沿触发启动和硬件复位（注 1）	上升沿启动和上升沿复位（图 19-9）	ON = 1 且 TMRx_ers ↑	TMRx_ers ↑	
		101		下降沿启动和下降沿复位	ON = 1 且 TMRx_ers ↓	TMRx_ers ↓	
		110		上升沿启动和低电平复位（图 19-10）	ON = 1 且 TMRx_ers ↑	TMRx_ers = 0	
		111		下降沿启动和高电平复位	ON = 1 且 TMRx_ers ↓	TMRx_ers = 1	
单稳态	10	000	保留				
		001	边沿触发启动（注 1）	上升沿启动（图 19-11）	ON = 1 且 TMRx_ers ↑	—	ON = 0 或 后的下一个时钟 后的下一个时钟（注 3）
		010		下降沿启动	ON = 1 且 TMRx_ers ↓	—	
		011		任意边沿启动	ON = 1 且 TMRx_ers ↕	—	
保留		100	保留				
保留		101	保留				
单触发		110	电平触发启动和硬件复位	高电平启动和低电平复位（图 19-12）	ON = 1 且 TMRx_ers = 1	TMRx_ers = 0	ON = 0 或 保持在复位状态（注 2）
		111		低电平启动和高电平复位	ON = 1 且 TMRx_ers = 0	TMRx_ers = 1	
保留	11	xxx	保留				

注：

1. 如果 ON = 0，则在 ON = 1 后需要一个边沿来重启定时器。
2. 当 T2TMR = T2PR 时，下一个时钟会清零 ON 并使 T2TMR 停止在 00h 处。
3. 当 T2TMR = T2PR 时，下一个时钟会使 T2TMR 停止在 00h 处，但不会清零 ON。

19.8. 操作示例

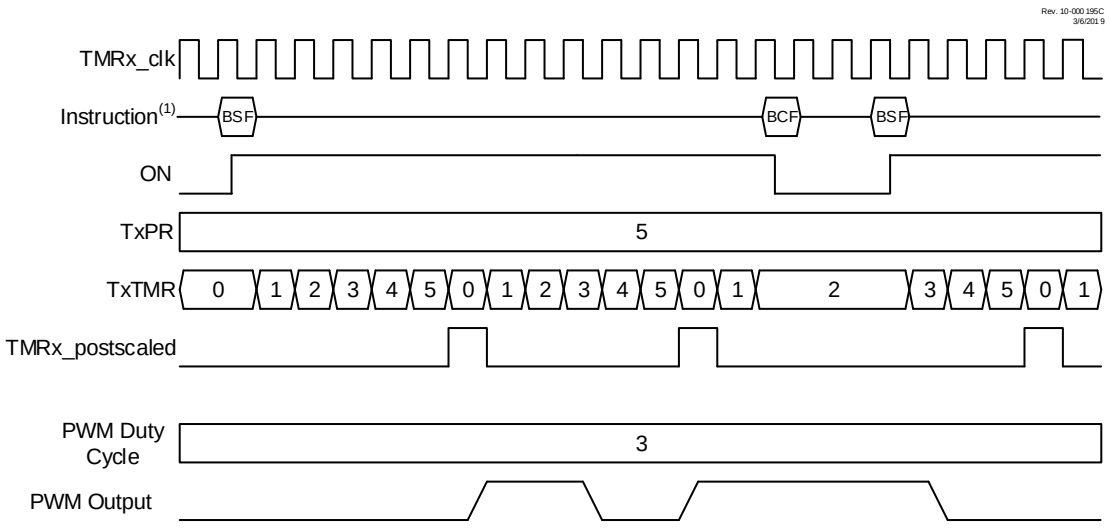
除非另外说明，否则下述内容适用于下列时序图：

- 预分频器和后分频器均设置为 1:1（CKPS 和 OUTPS 位）。
- 这些图显示了除 $F_{OSC}/4$ 外的所有时钟，并给出了 ON 和 TMRx_ers 至少两个完整周期的时钟同步延时。使用 $F_{OSC}/4$ 时，TMRx_ers 的时钟同步延时至少为一个指令周期；而 ON 则适用于下一个指令周期。
- 仅对 ON 和 TMRx_ers 进行了概括说明，时钟同步延时产生的结果可能与说明中略有不同。
- 在定时器用于 CCP 模块的 PWM 功能（如“**CCP——捕捉/比较/PWM 模块**”一章中的“**PWM 概述**”一节所述）的前提下，介绍了 PWM 占空比和 PWM 输出。这些信号不是 Timer2 模块的一部分。

19.8.1. 软件门控模式

该模式对应于传统的 Timer2 操作。当 ON = 1 时，定时器随每一个时钟输入递增；而当 ON = 0 时，定时器不递增。当 TxTMR 计数等于 TxPR 周期计数时，定时器在下一个时钟复位并从 0 开始继续计数。ON 位由软件控制的操作如图 19-3 所示。当 TxPR = 5 时，计数器递增至 TxTMR = 5 并在下一个时钟时变为零。

图 19-3. 软件门控模式时序图（MODE = 'b00000）



Note: 1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

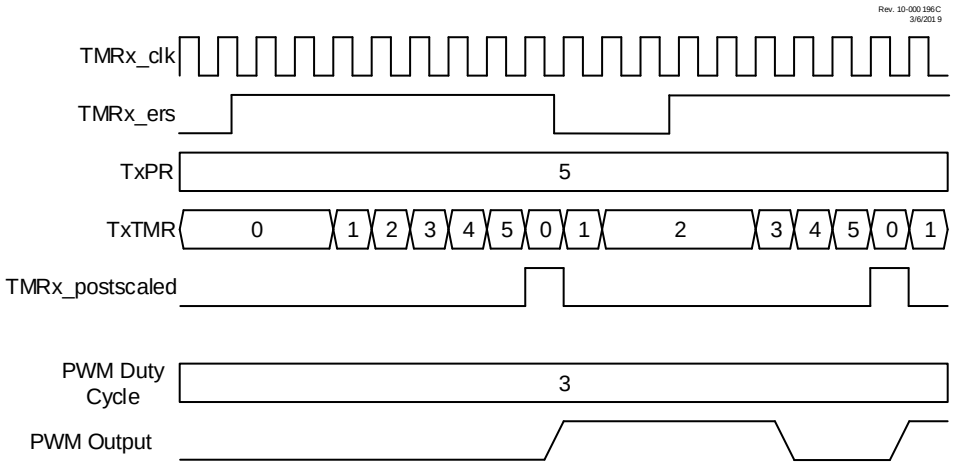
19.8.2. 硬件门控模式

硬件门控模式的工作方式与软件门控模式相同，惟一的区别是 TMRx_ers 外部信号也可对定时器进行门控。与 CCP 一起使用时，门控会延长 PWM 周期。如果定时器在 PWM 输出为高电平时停止，占空比也会延长。

如果 MODE = 'b00001，则定时器会在外部信号为高电平时停止。如果 MODE = 'b00010，则定时器会在外部信号为低电平时停止。

图 19-4 给出了 MODE = 'b00001 时的硬件门控模式，此时输入高电平会启动计数器。

图 19-4. 硬件门控模式时序图（MODE = 'b00001）



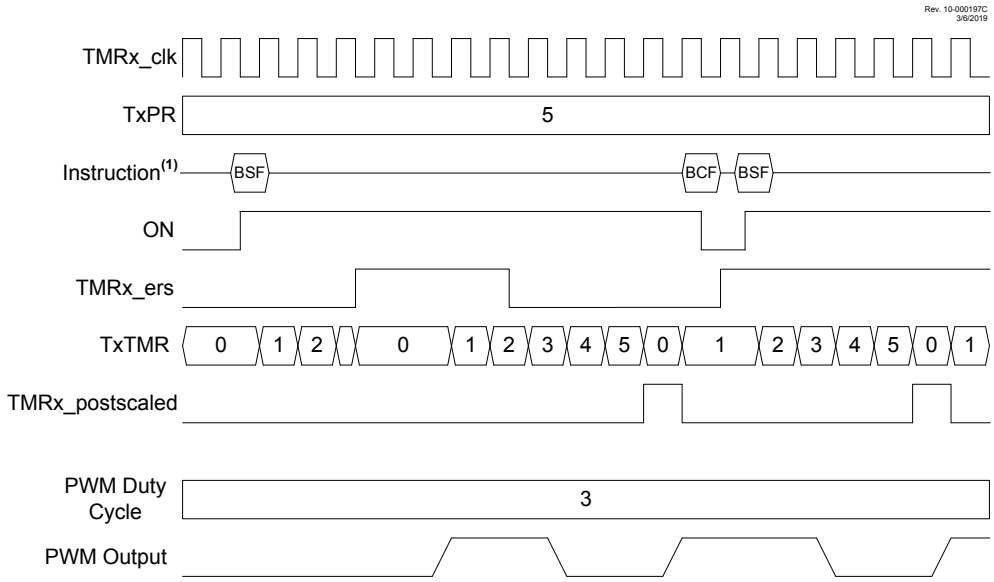
19.8.3. 边沿触发硬件限制模式

在硬件限制模式下，可在定时器达到周期计数之前通过 TMRx_ers 外部信号复位定时器。可实现三种类型的复位：

- 在上升沿或下降沿复位（MODE = 'b00011）
- 在上升沿复位（MODE = 'b00100）
- 在下降沿复位（MODE = 'b00101）

当定时器与 CCP 一起用于 PWM 模式时，提前复位会缩短周期，并会在两个时钟延时之后重新启动 PWM 脉冲。 请参见图 19-5。

图 19-5. 边沿触发硬件限制模式时序图（MODE = 'b00100）



Note: 1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

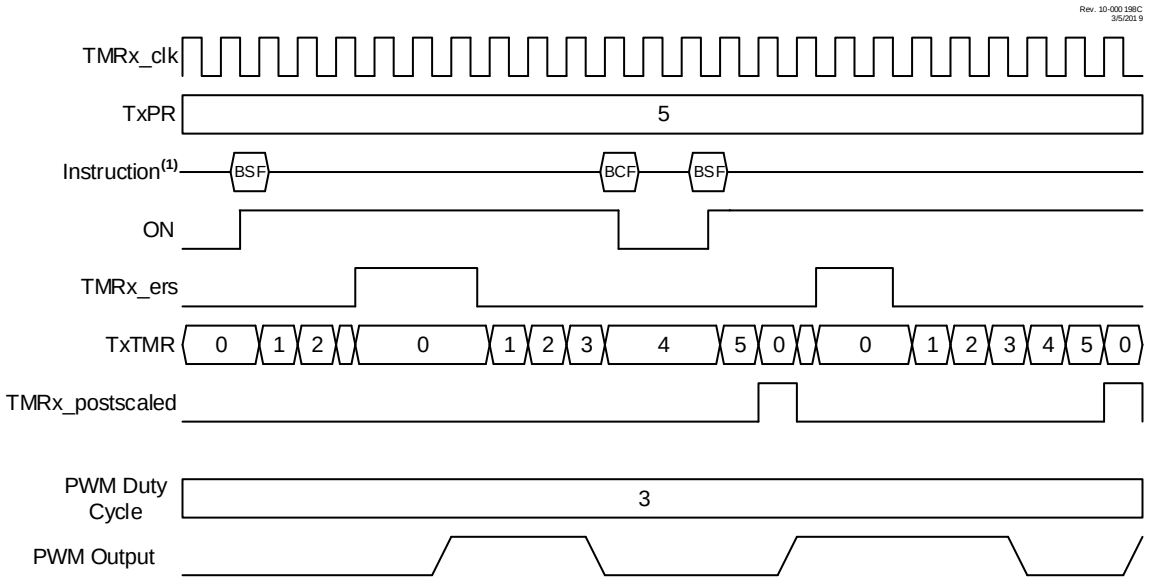
19.8.4. 电平触发硬件限制模式

在电平触发硬件限制定时器模式下，计数器通过高电平或低电平的外部信号 TMRx_ers 来复位，如图 19-6 所示。选择 MODE = 'b00110 时，定时器将由低电平外部信号复位。选择 MODE = 'b00111 时，定时器将由高电平外部信号复位。在本示例中，计数器在 TMRx_ers = 1 时复位。ON 由 BSF 和 BCF 指令控制。当 ON = 0 时，将忽略外部信号。

当 CCP 使用定时器作为 PWM 时基时，PWM 输出将在定时器开始计数时置为高电平，并且仅在定时器计数与 CCPRx 值匹配时置为低电平。当定时器计数与 TxPR 值匹配时，或在外部复位信号变为真并保持两个时钟周期后，定时器复位。

在 TxPR 发生匹配后的时钟周期或者外部复位信号放弃复位的两个时钟周期后，定时器开始计数，PWM 输出置为高电平。PWM 输出将保持为高电平，直到定时器递增计数至与 CCPRx 脉宽值匹配。如果外部复位信号在 PWM 输出为高电平时变为真，则 PWM 输出将保持为高电平，直到复位信号被释放，允许定时器递增计数到与 CCPRx 值匹配。

图 19-6. 电平触发硬件限制模式时序图（MODE = 'b00111）



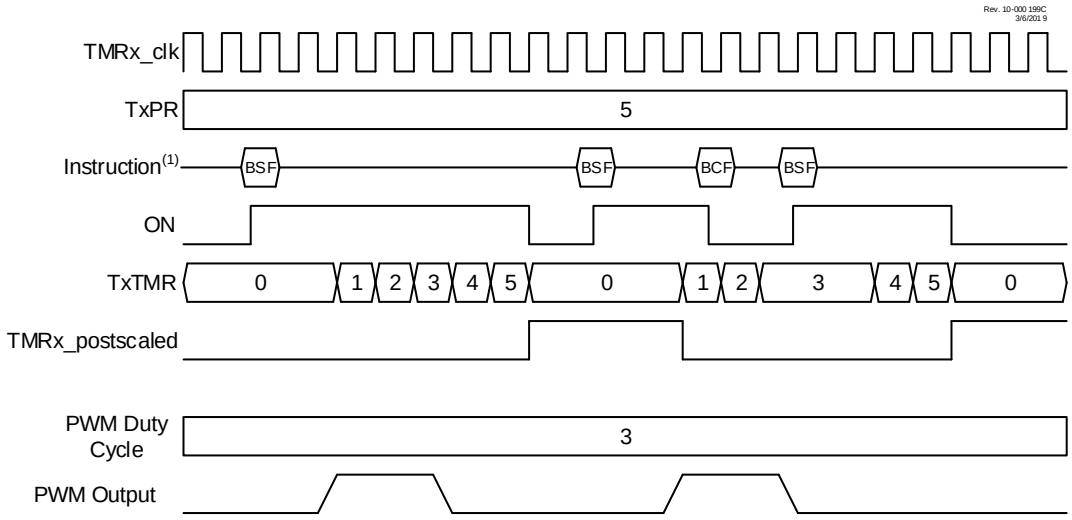
Note: 1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

19.8.5. 软件启动单触发模式

在单触发模式下，当定时器值与 TxPR 周期值匹配时，定时器复位，ON 位清零。ON 位必须通过软件置 1 才能启动另一定时器周期。设置 MODE = 'b01000 会选择图 19-7 所示的单触发模式。在示例中，ON 位通过 BSF 和 BCF 指令控制。在第一种情况下，BSF 指令将 ON 位置 1，计数器运行至计数完成并将 ON 位清零。在第二种情况下，BSF 指令启动周期，BCF/BSF 指令在该周期内关闭和启动计数器，然后计数器运行至计数完成。

当单触发模式与 CCP PWM 操作一起使用时，PWM 脉冲驱动会在 ON 位置 1 时启动。在 PWM 驱动激活时将 ON 位清零会延长 PWM 驱动。当定时器值与 CCPRx 脉冲宽度值匹配时，PWM 驱动将终止。PWM 驱动将保持关闭，直至软件将 ON 位置 1 以启动另一周期。如果软件在 CCPRx 匹配之后、但在 TxPR 匹配之前将 ON 位清零，PWM 驱动将延长，具体延长时间为 ON 位保持清零的时长。仅当 ON 位通过 TxPR 周期计数匹配清零后，才能通过将 ON 位置 1 的方式启动另一计时周期。

图 19-7. 软件启动单触发模式时序图（MODE = 'b01000）



Note: 1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

19.8.6. 边沿触发单触发模式

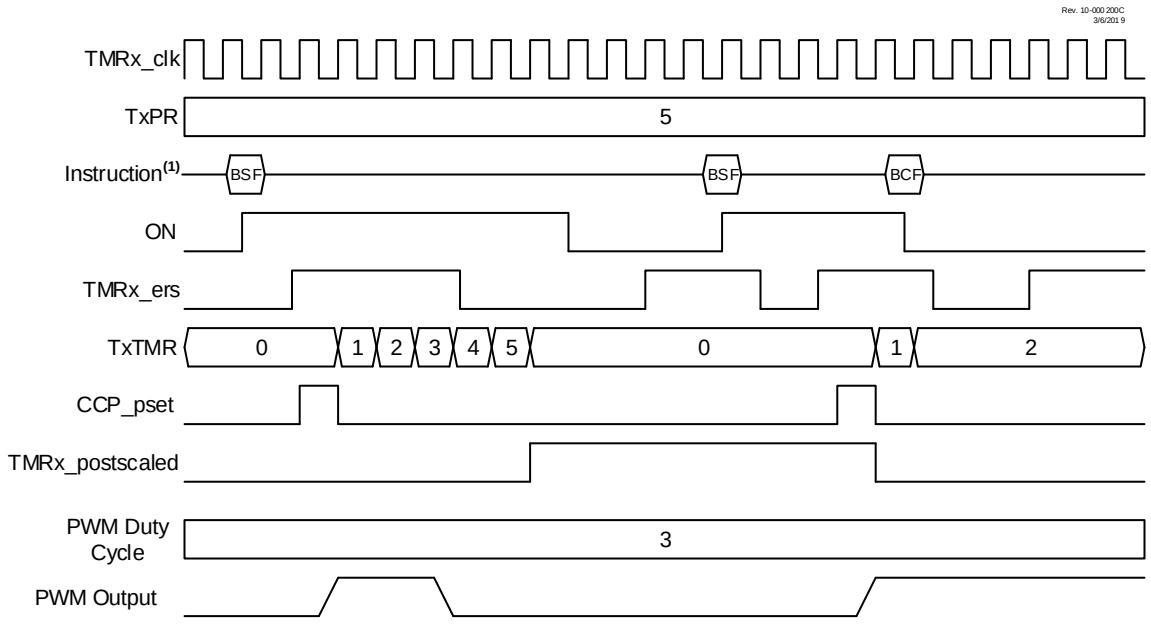
边沿触发单触发模式在 ON 位置 1 后通过外部信号输入的边沿启动定时器，并在定时器与 TxPR 周期值匹配时清零 ON 位。以下边沿将启动定时器：

- 上升沿（MODE = 'b01001）
- 下降沿（MODE = 'b01010）
- 上升沿或下降沿（MODE = 'b01011）

如果通过将 ON 位清零暂停了定时器，则在将 ON 位置 1 后需要另一个 TMRx_ers 边沿来恢复计数。 图 19-8 给出了上升沿单触发模式下的操作。

当边沿触发单触发模式与 CCP 配合使用时，边沿触发信号将激活 PWM 驱动，并在定时器与 CCPRx 脉宽值匹配时，禁止 PWM 驱动。当定时器因发生 TxPR 周期计数匹配而暂定时，PWM 驱动保持禁止状态。

图 19-8. 边沿触发单触发模式时序图（MODE = 'b01001）



Note: 1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

19.8.7. 边沿触发硬件限制单触发模式

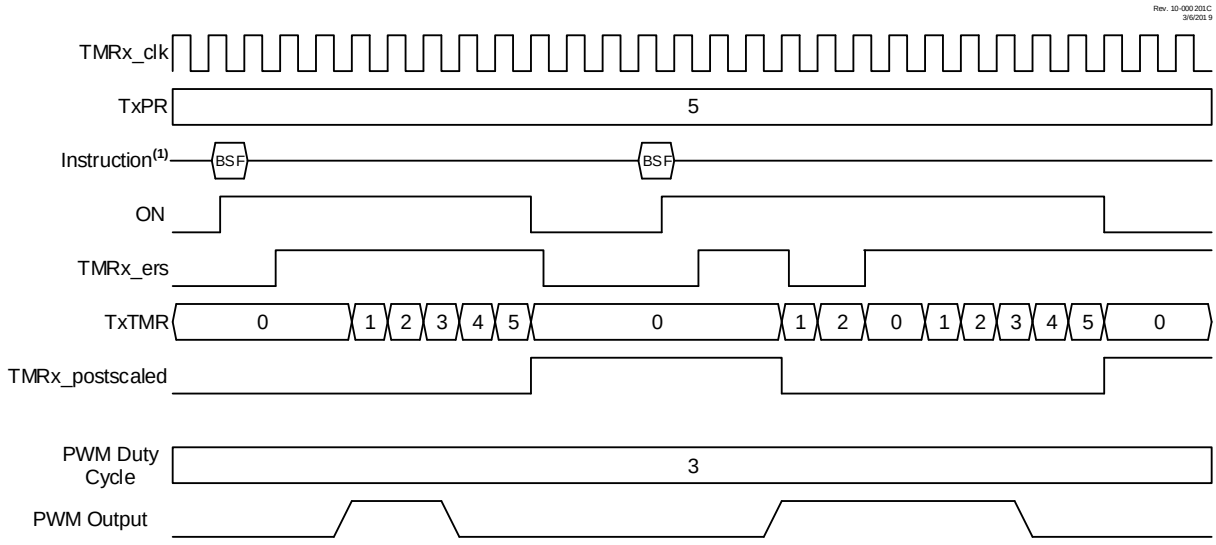
在边沿触发硬件限制单触发模式下，定时器在 ON 位置 1 后的第一个外部信号边沿启动，并在随后的所有边沿复位。只需要通过 ON 位置 1 后的第一个边沿来启动定时器即可。在随后的所有外部复位边沿的两个时钟周期后，计数器将自动恢复计数。边沿触发信号如下所示：

- 上升沿启动和复位（MODE = 'b01100）
- 下降沿启动和复位（MODE = 'b01101）

当定时器值与 TxPR 周期值发生匹配时，定时器将复位并清零 ON 位。外部信号边沿在软件将 ON 位置 1 后才会起作用。 图 19-9 给出了上升沿硬件限制单触发操作。

当该模式与 CCP 配合使用时，第一个启动边沿触发信号和随后的所有复位边沿都将激活 PWM 驱动。当定时器与 CCPRx 脉宽值匹配时，PWM 驱动将被禁止，除非外部信号边沿在 TxPR 周期发生匹配前将定时器复位，否则 PWM 驱动将保持禁止状态，直到定时器因发生匹配而暂停。

图 19-9. 边沿触发硬件限制单触发模式时序图（MODE = 'b01100）



Note: 1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

19.8.8. 电平复位、边沿触发硬件限制单触发模式

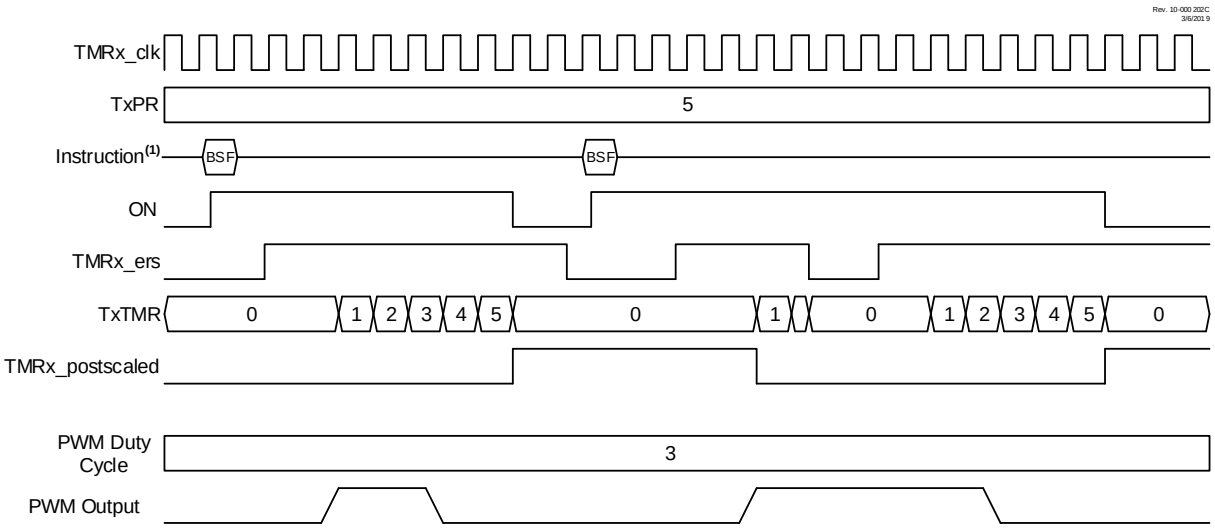
在电平触发单触发模式下，当 ON 位置 1 时，定时器计数通过外部信号电平复位，并在复位电平转变为有效电平的上升沿/下降沿开始计数。复位电平选择如下：

- 低电平复位（MODE = 'b01110）
- 高电平复位（MODE = 'b01111）

当定时器计数与 TxPR 周期计数匹配时，定时器会发生复位，ON 位会被清零。当 TxPR 匹配或软件控制将 ON 位清零时，在 ON 位置 1 后需要一个新的外部信号边沿来启动计数器。

当电平触发复位单触发模式与 CCP PWM 操作配合使用时，PWM 驱动通过启动定时器的外部信号边沿进入有效状态。当定时器计数等于 CCPRx 脉宽计数时，PWM 驱动进入无效状态。当定时器计数因 TxPR 周期计数匹配而清零时，PWM 驱动不会进入有效状态。

图 19-10. 低电平复位、边沿触发硬件限制单触发模式时序图（MODE = 'b01110）



Note: 1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

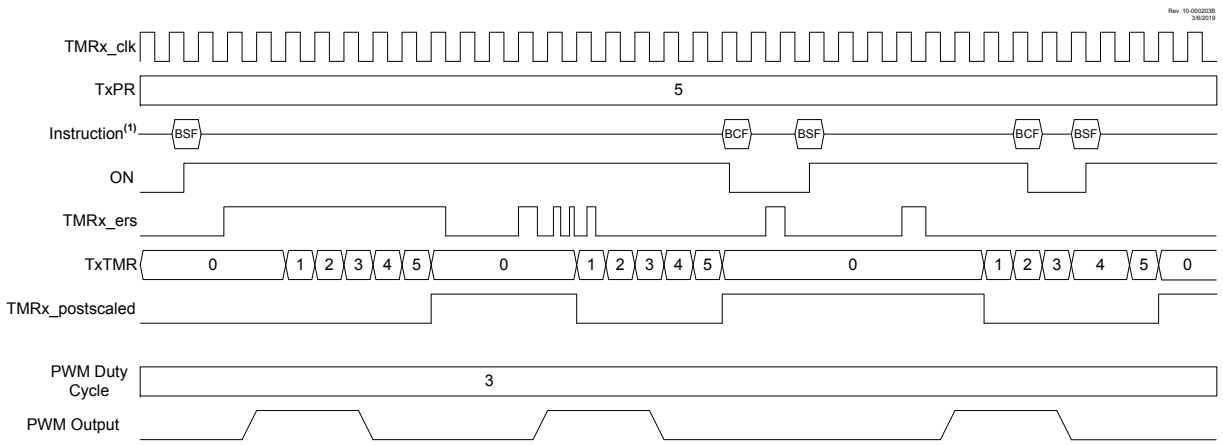
19.8.9. 边沿触发单稳态模式

边沿触发单稳态模式在 ON 位置 1 后通过外部复位信号输入的边沿启动定时器，并在定时器与 TxPR 周期值匹配时停止递增定时器。以下边沿将启动定时器：

- 上升沿（MODE = 'b10001）
- 下降沿（MODE = 'b10010）
- 上升沿或下降沿（MODE = 'b10011）

当边沿触发单稳态模式与 CCP PWM 操作配合使用时，PWM 驱动通过启动定时器的外部复位信号边沿进入有效状态，但在定时器与 TxPR 值匹配时将不变为有效。当定时器递增时，外部复位信号的其他边沿不会影响 CCP PWM。

图 19-11. 上升沿触发单稳态模式时序图（MODE = 'b10001）



Note: 1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

19.8.10. 电平触发硬件限制单触发模式

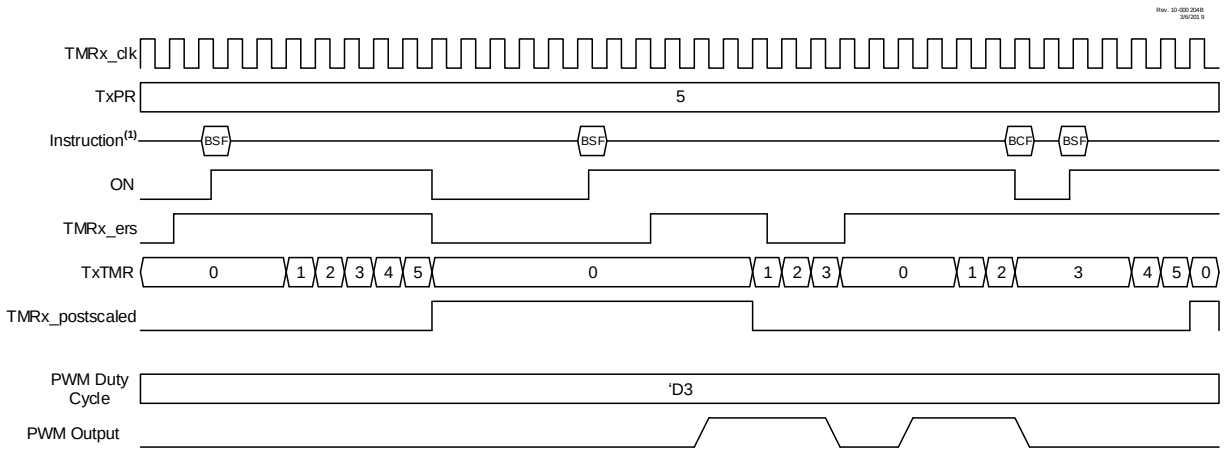
在电平触发硬件限制单触发模式下，定时器在外部复位电平出现时保持复位状态，并在 ON 位置 1 且外部信号不在复位电平时开始计数。如果其中一个外部信号不在复位电平或 ON 位置 1，则其他置为有效的信号将启动定时器。复位电平选择如下：

- 低电平复位（MODE = 'b10110）
- 高电平复位（MODE = 'b10111）

当定时器计数与 TxPR 周期计数匹配时，定时器会发生复位，ON 位会被清零。当 ON 位由 TxPR 匹配或软件控制清零时，定时器将保持复位状态，直到 ON 位置 1 且外部信号不在复位电平。

当电平触发硬件限制单触发模式与 CCP PWM 操作配合使用时，PWM 驱动会随外部信号边沿或 ON 位置 1（两者均会启动定时器）变为有效。

图 19-12. 电平触发硬件限制单触发模式时序图（MODE = 'b10110）



Note: 1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

19.9. 休眠期间的 Timer2 操作

当 PSYNC = 1 时，如果处理器处于休眠模式，Timer2 将无法工作。在处理器处于休眠模式时，T2TMR 和 T2PR 寄存器的内容将保持不变。

当 PSYNC = 0 时，只要选择的时钟源仍在运行，Timer2 就可在休眠模式下工作。如果将任一内部振荡器选作时钟源，则振荡器会在休眠模式期间保持工作状态。

19.10. 寄存器定义：Timer2 控制

下表列出了 Timer2 外设的长位名称前缀。更多信息，请参见**“寄存器和位命名约定”**一章中的**“长位名称”**一节。

表 19-2. Timer2 长位名称前缀

外设	位名称前缀
Timer2	T2

重要：涉及模块 Timer2 的内容同样适用于该器件上所有偶数编号的定时器（Timer2 和 Timer4 等）。

19.10.1. TxTMR

名称： TxTMR
偏移量： 0x028C

定时器计数器寄存器

位	7	6	5	4	3	2	1	0
	TxTMR[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7:0 – TxTMR[7:0] Timerx 计数器

19.10.2. TxPR

名称： TxPR
偏移量： 0x028D

定时器周期寄存器

位	7	6	5	4	3	2	1	0
	TxPR[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	1	1	1	1	1	1	1	1

Bit 7:0 – TxPR[7:0] 定时器周期寄存器

值	说明
0至255	当TxTMR达到TxPR值时，定时器从0重新启动

19.10.3. TxCON

名称： TxCON
偏移量： 0x028E

Timerx 控制寄存器

位	7	6	5	4	3	2	1	0
	ON	CKPS[2:0]			OUTPS[3:0]			
访问	R/W/HC	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7 – ON 定时器使能[(1)]

值	说明
1	使能定时器
0	禁止定时器：所有计数器和状态机均复位

Bit 6:4 – CKPS[2:0] 定时器时钟预分频比选择

值	说明
111	1:128 预分频比
110	1:64 预分频比
101	1:32 预分频比
100	1:16 预分频比
011	1:8 预分频比
010	1:4 预分频比
001	1:2 预分频比
000	1:1 预分频比

Bit 3:0 – OUTPS[3:0] 定时器输出后分频比选择

值	说明
1111	1:16 后分频比
1110	1:15 后分频比
1101	1:14 后分频比
1100	1:13 后分频比
1011	1:12 后分频比
1010	1:11 后分频比
1001	1:10 后分频比
1000	1:9 后分频比
0111	1:8 后分频比
0110	1:7 后分频比
0101	1:6 后分频比
0100	1:5 后分频比
0011	1:4 后分频比
0010	1:3 后分频比
0001	1:2 后分频比
0000	1:1 后分频比

注：

1. 在某些模式下，ON 位将由硬件自动清零。请参见表 19-1。

19.10.4. TxHLT

名称： TxHLT
偏移量： 0x028F

定时器硬件限制控制寄存器

位	7	6	5	4	3	2	1	0
	PSYNC	CPOL	CSYNC	MODE[4:0]				
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7 – PSYNC 定时器预分频器同步使能(1, 2)

值	说明
1	定时器预分频器输出与 $F_{OSC}/4$ 同步
0	定时器预分频器输出不与 $F_{OSC}/4$ 同步

Bit 6 – CPOL 定时器时钟极性选择(3)

值	说明
1	输入时钟的下降沿驱动定时器/预分频器
0	输入时钟的上升沿驱动定时器/预分频器

Bit 5 – CSYNC 定时器时钟同步使能(4, 5)

值	说明
1	ON 位与定时器时钟输入同步
0	ON 位与定时器时钟输入不同步

Bit 4:0 – MODE[4:0] 定时器控制模式选择(6, 7)

值	说明
00000 至 11111	见 表 19-1

注：

1. 将该位置 1 可确保读取 TxTMR 时返回有效数据值。
2. 当该位为 1 时，定时器无法在休眠模式中运行。
3. 当 ON = 1 时，不得更改 CKPOL。
4. 将该位置 1 可以确保使能或禁止 ON 时无毛刺。
5. 当该位置 1 时，ON 位置 1 后定时器操作会延时两个输入时钟。
6. 除非另外说明，否则所有模式均在 ON = 1 时启动，ON = 0 时停止（停止不会影响 TxTMR 的值）。
7. 当 TxTMR = TxPR 时，无论工作模式如何，下一个时钟均会清零 TxTMR。

19.10.5. TxCLKCON

名称： TxCLKCON
偏移量： 0x0290

定时器时钟源选择寄存器

位	7	6	5	4	3	2	1	0
						CS[2:0]		
访问						R/W	R/W	R/W
复位						0	0	0

Bit 2:0 – CS[2:0] 定时器时钟源选择

表 19-3. 时钟源选择

CS	时钟源
111	保留
110	MFINTOSC（32 kHz）
101	MFINTOSC（500 kHz）
100	LFINTOSC
011	HFINTOSC
010	F_{OSC}
001	$F_{OSC}/4$
000	通过 T2INPPS 选择的引脚

19.10.6. TxRST

名称： TxRST
偏移量： 0x0291

定时器外部复位信号选择寄存器

位	7	6	5	4	3	2	1	0
					RSEL[3:0]			
访问					R/W	R/W	R/W	R/W
复位					0	0	0	0

Bit 3:0 – RSEL[3:0] 外部复位源选择

表 19-4. 外部复位源

RSEL	复位源
1111-0101	保留
0100	PWM4_OUT
0011	PWM3_OUT
0010	CCP2_OUT
0001	CCP1_OUT
0000	通过 T2INPPS 选择的引脚

19.11. 寄存器汇总——Timer2

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x028B	保留									
0x028C	T2TMR	7:0	T2TMR[7:0]							
0x028D	T2PR	7:0	T2PR[7:0]							
0x028E	T2CON	7:0	ON	CKPS[2:0]			OUTPS[3:0]			
0x028F	T2HLT	7:0	PSYNC	CPOL	CSYNC	MODE[4:0]				
0x0290	T2CLKCON	7:0						CS[2:0]		
0x0291	T2RST	7:0					RSEL[3:0]			

20. CCP——捕捉/比较/PWM 模块

捕捉/比较/PWM 模块是允许用户计时和控制不同事件，以及产生脉宽调制（PWM）信号的外设。在捕捉模式下，外设允许对事件的持续时间进行计时。当超过预先确定的时间时，比较模式允许用户触发一个外部事件。PWM 模式可以产生不同频率和占空比的脉宽调制信号。

所有 CCP 模块的捕捉和比较功能都相同。

重要：在具有多个 CCP 模块的器件中，要特别注意所使用的寄存器名称，这一点很重要。本章使用前缀“CCPx”代替具体的编号来统一表示。前缀中“x”位置上的编号用于区分不同的模块。例如，CCP1CON 和 CCP2CON 分别控制两个完全不同 CCP 模块相同的工作特性。

20.1. CCP 模块配置

每个捕捉/比较/PWM 模块都与一个控制寄存器（CCPxCON）、一个捕捉输入选择寄存器（CCPxCAP）和一个数据寄存器（CCPRx）相关联。数据寄存器由两个 8 位寄存器（CCPRxL（低字节）和 CCPRxH（高字节））组成。

20.1.1. CCP 模块和定时器资源

CCP 模块使用定时器 1 和 2，具体因所选模式而异。CCP 模块可以在捕捉、比较或 PWM 模式下使用不同的定时器，如下表所示。

表 20-1. CCP 模式——定时器资源

CCP 模式	定时器资源
捕捉	Timer1
比较	
PWM	Timer2

如果所有模块配置为同时在同一模式（捕捉/比较或 PWM）下工作，则这些模块可以立即进入工作状态，并且可以共用同一定时器资源。

20.1.2. 漏极开路输出选项

在输出模式（比较模式或 PWM 模式）下工作时，可选择将 CCPx 引脚的驱动器配置为漏极开路输出。此功能允许通过外部上拉电阻将引脚上的电压拉高，并允许输出与外部电路通信而无需额外的电平转换器。

20.2. 捕捉模式

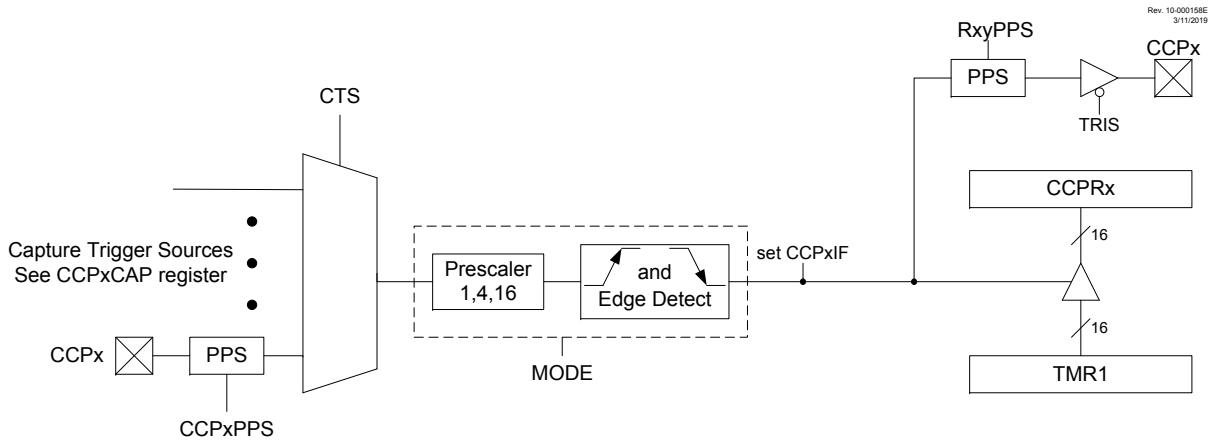
捕捉模式使用 16 位奇数编号的定时器资源（Timer1）。当捕捉源发生事件时，16 位 CCPRx 寄存器捕捉并存储 TMRx 寄存器的 16 位值。事件的定义如下所示，由 MODE 位配置：

- CCPx 输入的每个下降沿
- CCPx 输入的每个上升沿
- CCPx 输入的每 4 个上升沿
- CCPx 输入的每 16 个上升沿
- CCPx 输入的每个边沿（上升沿或下降沿）

当进行捕捉时，PIRx 寄存器的 CCP 中断标志（CCPxIF）位将置 1。中断标志必须用软件清零。如果在读取 CCPRx 寄存器中的值之前发生了另一次捕捉，那么之前捕捉的值会被新捕捉值覆盖。下图给出了捕捉操作的简化框图。

重要：如果在进行 2 字节读取期间发生事件，则高字节和低字节数据将来自不同的事件。建议在读取 CCPRx 寄存器对时禁止模块或读取寄存器对两次以确保数据完整性。

图 20-1. 捕捉模式工作原理框图



20.2.1. 捕捉源

通过 CTS 位选择捕捉源。

在捕捉模式下，必须通过将相应的 TRIS 控制位置 1 把 CCPx 引脚配置为输入引脚。

重要：如果将 CCPx 引脚配置为输出引脚，对该端口的写操作可能引发一次捕捉事件。

20.2.2. 使用捕捉功能时的 Timer1 模式

Timer1 运行在定时器模式或同步计数器模式下时，CCP 模块才能使用捕捉功能。在异步计数器模式下，可能无法进行捕捉操作。

有关配置 Timer1 的更多信息，请参见**“TMR1——带门控的 Timer1 模块”**一节。

20.2.3. 软件中断模式

当捕捉模式更改时，可能会产生错误捕捉中断。用户需保持 PIEx 寄存器中的 CCPxIE 中断允许位为零以避免产生误中断。此外，用户还需在工作模式发生任何变化后将 PIRx 寄存器的 CCPxIF 中断标志位清零。

重要：在捕捉模式下，不得使用系统时钟（F_{OSC}）作为 Timer1 的时钟源。要使捕捉模式能够识别 CCPx 引脚上的触发事件，Timer1 的时钟必须来自指令时钟（$F_{OSC}/4$）或外部时钟源。

20.2.4. CCP 预分频器

MODE 位指定了四个预分频比设置。每当 CCP 模块关闭或 CCP 模块未处于捕捉模式时，预分频器计数器便会清零。任何复位都会将预分频器计数器清零。

从一个捕捉预分频比切换为另一个捕捉预分频比不会清零预分频器，并可能产生一次错误中断。为避免此意外操作，可在改变预分频比前通过清零 CCPxCON 寄存器来关闭模块。以下示例给出了执行此功能的代码。

例 20-1. 切换捕捉预分频比

```
BANKSEL CCP1CON
CLRF    CCP1CON         ;Turn CCP module off
MOVLW   NEW_CAPT_PS     ;CCP ON and Prescaler select → W
MOVWF   CCP1CON         ;Load CCP1CON with this value
```

20.2.5. 休眠期间的捕捉

捕捉模式依靠 Timer1 模块才能正确工作。可选用两种方式在捕捉模式下驱动 Timer1 模块。它可以由指令时钟（$F_{OSC}/4$）或由外部时钟源驱动。

当 Timer1 由 $F_{OSC}/4$ 提供时钟时，Timer1 在休眠期间不会递增。当器件从休眠状态唤醒时，Timer1 将从其之前的状态继续工作。

当 Timer1 通过外部时钟源提供时钟时，捕捉模式将在休眠模式期间继续工作。

20.3. 比较模式

本节介绍的比较模式功能适用于所有 CCP 模块，对于不同的模块没有差别。

比较模式使用 16 位奇数编号的定时器资源（Timer1）。CCPRx 寄存器的 16 位值不断与 TMR1 寄存器的 16 位值进行比较。出现匹配时，可能会发生以下某一事件：

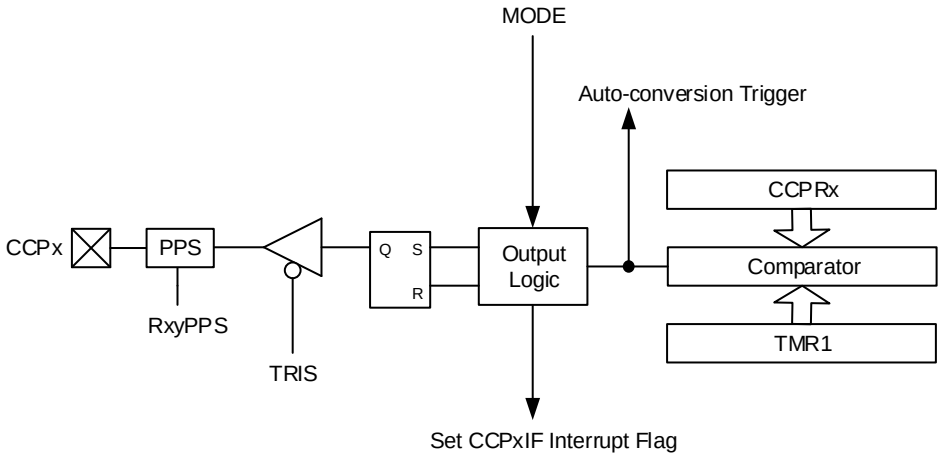
- 翻转 CCPx 输出并清零 TMR1
- 翻转 CCPx 输出但不清零 TMR1
- CCPx 输出高电平
- CCPx 输出低电平
- 产生脉冲输出
- 产生脉冲输出并清零 TMR1

引脚上的操作由 MODE 控制位的值决定。

所有比较模式都能产生中断。当 MODE = 'b0001 或'b1011 时，CCP 将复位 TMR1 寄存器。

下图给出了比较操作的简化框图。

图 20-2. 比较模式工作原理框图



20.3.1. CCPx 引脚配置

软件必须通过清零相应的 TRIS 位并借助 RxyPPS 寄存器定义相应输出引脚的方式将 CCPx 引脚配置为输出引脚。有关详细信息，请参见**“PPS——外设引脚选择模块”**一章。

CCP 输出也可用作其他外设的输入。

重要：清零 CCPxCON 寄存器会将 CCPx 比较输出锁存器强制设为默认的低电平状态。这不是端口 I/O 数据锁存器。

20.3.2. 使用比较功能时的 Timer1 模式

在比较模式下，Timer1 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。

有关配置 Timer1 的更多信息，请参见**“TMR1——带门控的 Timer1 模块”**一章。

重要：在比较模式下，不得使用系统时钟（F_{OSC}）作为 Timer1 的时钟源。欲使比较模式能够识别 CCPx 引脚上的触发事件，Timer1 的时钟必须来自指令时钟（$F_{OSC}/4$）或外部时钟源。

20.3.3. 休眠期间的比较

由于 F_{OSC} 在休眠模式下关闭，比较模式在休眠模式下将不能正常工作，除非定时器正在运行。器件将在发生中断（如果允许）时唤醒。

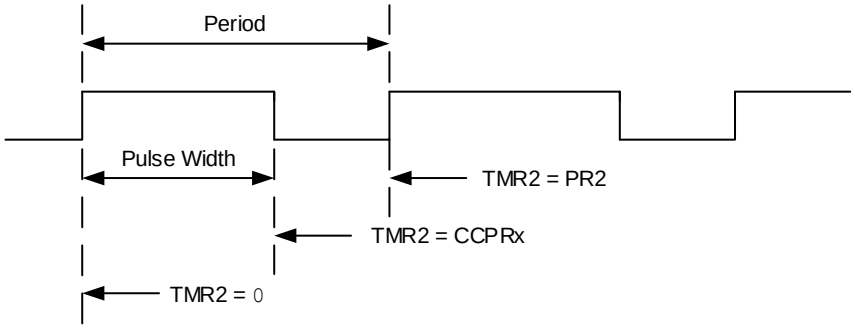
20.4. PWM 概述

脉宽调制（PWM）是一种通过在完全开启和完全关闭状态之间进行快速切换来控制负载功率的方案。PWM 信号类似于方波，信号的高电平部分视为开启状态，信号的低电平部分视为关闭状态。高电平部分（也称为脉宽）可以随时间而变，并以步为单位进行定义。施加的步数越多（这会增大脉宽），为负载提供的功率就越大。施加的步数减少时（这会缩短脉宽），提供的功率就越小。PWM 周期定义为一个完整周期的持续时间，或者开启和关闭时间相加的总时间。

PWM 分辨率定义可以在单个 PWM 周期中出现的最大步数。分辨率越高，就可以越精确地控制施加在负载上的功率。

占空比这一术语描述开启时间与关闭时间之间以百分比形式表示的比例，0%代表完全关闭，100%代表完全开启。占空比越低，施加的功率就越低；占空比越高，施加的功率就越高。下图给出了 PWM 信号的典型波形。

图 20-3. CCP PWM 输出信号



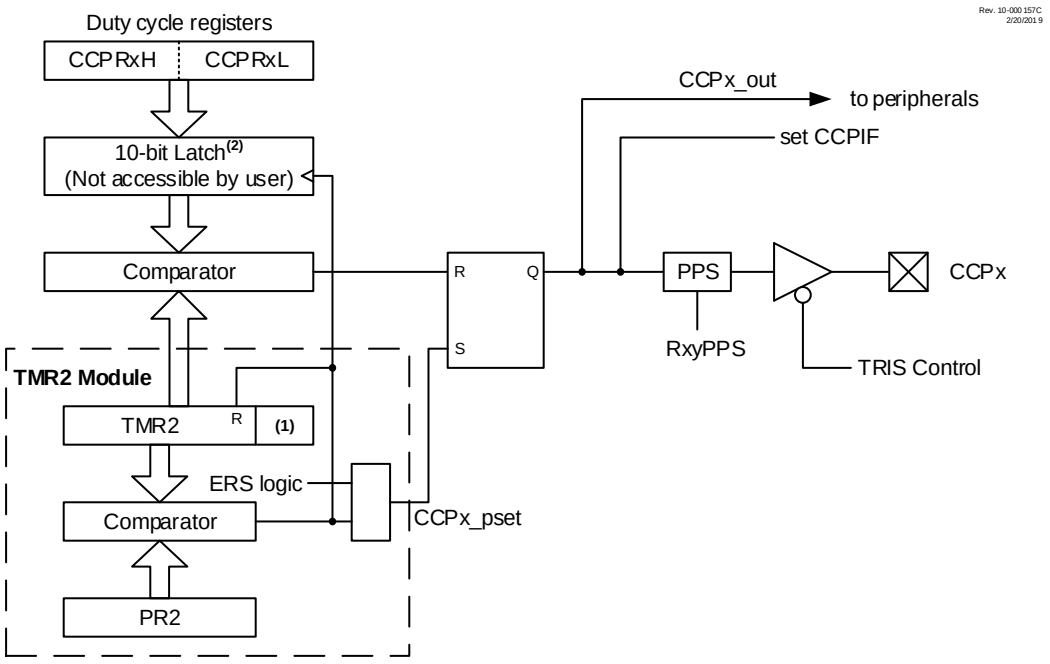
20.4.1. 标准 PWM 操作

本节介绍的标准 PWM 功能适用于所有 CCP 模块，对于不同的模块没有差别。它可以在 CCPx 引脚上产生最高 10 位分辨率的 PWM 信号。由下列寄存器控制周期、占空比和分辨率：

- 偶数编号的 TxPR（T2PR）寄存器
- 偶数编号的 TxCON（T2CON）寄存器
- 16 位 CCPRx 寄存器
- CCPxCON 寄存器

为确保 PWM 正常工作，需要将 $F_{OSC}/4$ 作为 T2TMR 的时钟输入。下图给出了 PWM 操作的简化框图。

图 20-4. PWM 简化框图



Notes: 1. An 8-bit timer is concatenated with two bits generated by Fosc or two bits of the internal prescaler to create 10-bit time base.
2. The alignment of the 10 bits from the CCPR register is determined by the CCPxFMT bit.

重要：必须将 CCPx 引脚对应的 TRIS 位清零，才能使能该引脚上的 PWM 输出。

20.4.2. Timer2 定时器资源

PWM 标准模式使用 8 位 Timer2 定时器资源来指定 PWM 周期。

20.4.3. PWM 周期

PWM 周期由 Timer2 的 T2PR 寄存器来指定。PWM 周期可利用下面的公式计算。

公式 20-1. PWM 周期

$$PWM\ Period = [(T2PR + 1)] \bullet 4 \bullet T_{OSC} \bullet (TMR2\ Prescale\ Value)$$

其中 $T_{OSC} = 1/F_{OSC}$

当 T2TMR 中的值与 T2PR 中的值相等时，在下一个递增事件将发生以下 3 个事件：

- T2TMR 被清零
- CCPx 引脚被置 1（例外情况：如果 PWM 占空比 = 0%，引脚将不会被置 1）
- PWM 占空比从 CCPRx 寄存器传送到 10 位缓冲区

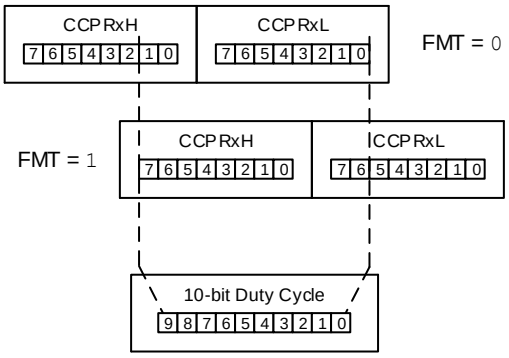
重要：不使用定时器后分频器（见**“TMR2——Timer2 模块”**一章中的**“Timer2 中断”**一节）来确定 PWM 频率。

20.4.4. PWM 占空比

可通过将一个 10 位值写入 CCPRx 寄存器来指定 PWM 占空比。10 位值的对齐方式由 FMT 位决定（见图 20-5）。可以随时写入 CCPRx 寄存器。但在 T2PR 和 T2TMR 之间发生匹配之前，占空比的值不会被锁存到 10 位缓冲区中。

使用下面的公式计算 PWM 脉宽和 PWM 占空比。

图 20-5. PWM 10 位对齐方式



公式 20-2. 脉冲宽度

$$\text{脉冲宽度} = (CCPRxH{:}CCPRxL \text{ 寄存器值}) \bullet T_{OSC} \bullet (TMR2 \text{ 分频比值})$$

公式 20-3. 占空比

$$\text{占空比} = \frac{(CCPRxH{:}CCPRxL \text{ 寄存器值})}{4(T2PR + 1)}$$

CCPRx 寄存器用于为 PWM 占空比提供双重缓冲。这种双重缓冲极其重要，可以避免在 PWM 工作过程中产生毛刺。

8 位定时器 T2TMR 寄存器与 2 位内部系统时钟（F_{OSC}）或预分频器的 2 位一起构成 10 位时基。如果 Timer2 预分频比设置为 1:1，则使用系统时钟。

当 10 位时基与 CCPRx 寄存器中的值匹配时，清零 CCPx 引脚（见图 20-4）。

20.4.5. PWM 分辨率

分辨率决定在给定周期内的可用占空比数。例如，10 位分辨率将产生 1024 个离散的占空比，而 8 位分辨率将产生 256 个离散的占空比。

当 T2PR 为 0xFF 时，最大 PWM 分辨率为 10 位。分辨率是 T2PR 寄存器值的函数，如下图所示。

公式 20-4. PWM 分辨率

$$\text{分辨率} = \frac{\log[4(T2PR + 1)]}{\log(2)} \text{位}$$

重要：如果脉宽值大于周期值，则指定的 PWM 引脚将保持不变。

表 20-2. PWM 频率和分辨率示例（F_{OSC} = 20 MHz）

PWM 频率	1.22 kHz	4.88 kHz	19.53 kHz	78.12 kHz	156.3 kHz	208.3 kHz
定时器预分频值	16	4	1	1	1	1
T2PR 值	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
最高分辨率（位）	10	10	10	8	7	6.6

表 20-3. PWM 频率和分辨率示例（F_{OSC} = 8 MHz）

PWM 频率	1.22 kHz	4.90 kHz	19.61 kHz	76.92 kHz	153.85 kHz	200.0 kHz
定时器预分频值	16	4	1	1	1	1
T2PR 值	0x65	0x65	0x65	0x19	0x0C	0x09
最高分辨率（位）	8	8	8	6	5	5

20.4.6. 休眠模式下的操作

在休眠模式下，T2TMR 寄存器将不会递增，模块状态也不会改变。如果 CCPx 引脚正在驱动一个值，则会继续驱动该值。当器件被唤醒时，T2TMR 将从先前状态继续。

20.4.7. 改变系统时钟频率

PWM 频率是由系统时钟频率产生的。系统时钟频率的任何改变都将导致 PWM 频率的改变。有关更多详细信息，请参见“**OSC——振荡器模块**”一节。

20.4.8. 复位的影响

任何复位都将强制所有端口为输入模式，并强制 CCP 寄存器为其复位状态。

20.4.9. 设置 PWM 操作

以下步骤说明了如何将 CCP 模块配置为标准 PWM 操作：

1. 使用 RxyPPS 控制选择所需输出引脚，以选择 CCPx 作为源。通过将相关的 TRIS 位置 1，禁止所选引脚输出驱动器。稍后，将在 PWM 设置结束时使能输出。
2. 将 PWM 周期值装入定时器周期寄存器 T2PR。
3. 将合适的值装入 CCPxCON 寄存器，从而将 CCP 模块配置为 PWM 模式。
4. 将 PWM 占空比值装入 CCPRx 寄存器，并配置 FMT 位以设置适当的寄存器对齐方式。
5. 配置并启动定时器：
   - 清零 PIRx 寄存器的 TMR2IF 中断标志位。请参见下面的注意事项。
   - 选择 F_{OSC}/4 作为定时器时钟源。只有这样才能确保 PWM 模块正常工作。
   - 用所需定时器预分频值配置 T2CON 寄存器的 CKPS 位。
   - 通过将 T2CON 寄存器的 ON 位置 1 来使能定时器。
6. 使能 PWM 输出：
   - 等待定时器溢出并且 PIRx 寄存器的 TMR2IF 位置 1。请参见下面的注意事项。
   - 通过将相关的 TRIS 位清零，使能 CCPx 引脚输出驱动器。

重要：要在第一个 PWM 输出时发送完整的占空比和周期，设置过程必须包含上述步骤。如果不需要在第一个输出就发送完整的 PWM 信号，可以忽略步骤 6。

20.5. 寄存器定义：CCP 控制

下表列出了 CCP 外设的长位名称前缀。更多信息，请参见“**寄存器和位命名约定**”一章中的**“长位名称”**一节。

表 20-4. CCP 长位名称前缀

外设	位名称前缀
CCP1	CCP1
CCP2	CCP2

20.5.1. CCPxCON

名称： CCPxCON
偏移量： 0x30E,0x312

CCP 控制寄存器

位	7	6	5	4	3	2	1	0
	EN		OUT	FMT	MODE[3:0]			
访问	R/W		R	R/W	R/W	R/W	R/W	R/W
复位	0		x	0	0	0	0	0

Bit 7 – EN CCP 模块使能

值	说明
1	使能 CCP
0	禁止 CCP

Bit 5 – OUT CCP 输出数据（只读）

Bit 4 – FMT CCPxRH:L 值对齐（PWM 模式）

值	条件	说明
x	捕捉模式	未使用
x	比较模式	未使用
1	PWM 模式	左对齐格式
0	PWM 模式	右对齐格式

Bit 3:0 – MODE[3:0] CCP 模式选择

表 20-5. CCPx 模式选择

值	说明	将 CCPxIF 置 1
11xx	PWM 模式，PWM 操作	是
1011	比较输出；脉冲输出；清零 TMR1(2)	是
1010	比较模式，脉冲输出	是
1001	比较模式，清零输出(1)	是
1000	比较模式，将输出置(1)	是
0111	比较模式，CCPx 输入的每 16 个上升沿	是
0110	比较模式，CCPx 输入的每 4 个上升沿	是
0101	捕捉模式，CCPx 输入的每个上升沿	是
0100	捕捉模式，CCPx 输入的每个下降沿	是
0011	捕捉模式，CCPx 输入的每个边沿	是
0010	比较模式，翻转输出	是
0001	脉冲输出；翻转输出；清零 TMR1(2)	是
0000	禁止	—

注：

1. 比较模式的置 1 和清零操作通过设置 MODE = 'b0000 或 EN = 0 来复位。
2. 当 MODE = 'b0001 或'b1011 时，与 CCP 模块相关的定时器清零。TMR1 是 CCP 模块的默认选择，因此仅用于说明目的。

20.5.2. CCPxCAP

名称： CCPxCAP
偏移量： 0x30F,0x313

捕捉触发输入选择寄存器

位	7	6	5	4	3	2	1	0
							CTS[1:0]	
访问							R/W	R/W
复位							0	0

Bit 1:0 – CTS[1:0] 捕捉触发输入选择

表 20-6. 捕捉触发源

CTS 值	源
11–10	保留
01	IOC 中断
00	通过 CCPxPPS 选择的引脚

20.5.3. CCPRx

名称： CCPRx
偏移量： 0x30C,0x310

捕捉/比较/脉宽寄存器

位	15	14	13	12	11	10	9	8
	CCPR[15:8]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	x	x	x	x	x	x	x	x

位	7	6	5	4	3	2	1	0
	CCPR[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	x	x	x	x	x	x	x	x

Bit 15:0 – CCPR[15:0] 捕捉/比较/脉宽

复位状态： POR/BOR = xxxxxxxxxxxxxxxx
所有其他复位 = uuuuuuuuuuuuuuuu

注：该多字节寄存器中的各个字节可使用以下寄存器名称进行访问：

- 当 MODE = 捕捉或比较时
  - CCPRxH：访问高字节 CCPR[15:8]
  - CCPRxL：访问低字节 CCPR[7:0]
- 当 MODE = PWM 且 FMT = 0 时
  - CCPRx[15:10]：未使用
  - CCPRxH[1:0]：访问高 2 位（CCPR[9:8]）
  - CCPRxL：访问低 8 位（CCPR[7:0]）
- 当 MODE = PWM 且 FMT = 1 时
  - CCPRxH：访问高 8 位（CCPR[9:2]）
  - CCPRxL[7:6]：访问低 2 位（CCPR[1:0]）
  - CCPRx[5:0]：未使用

20.6. 寄存器汇总——CCP 控制

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x030B	保留									
0x030C	CCPR1	7:0	CCPR[7:0]							
		15:8	CCPR[15:8]							
0x030E	CCP1CON	7:0	EN		OUT	FMT	MODE[3:0]			
0x030F	CCP1CAP	7:0							CTS[1:0]	
0x0310	CCPR2	7:0	CCPR[7:0]							
		15:8	CCPR[15:8]							
0x0312	CCP2CON	7:0	EN		OUT	FMT	MODE[3:0]			
0x0313	CCP2CAP	7:0							CTS[1:0]	

21. PWM——脉宽调制

PWM 模块可产生由占空比、周期和分辨率决定的脉宽调制信号，占空比、周期和分辨率则通过以下寄存器进行配置：

- T2PR
- T2CON
- PWMxDC
- PWMxCON

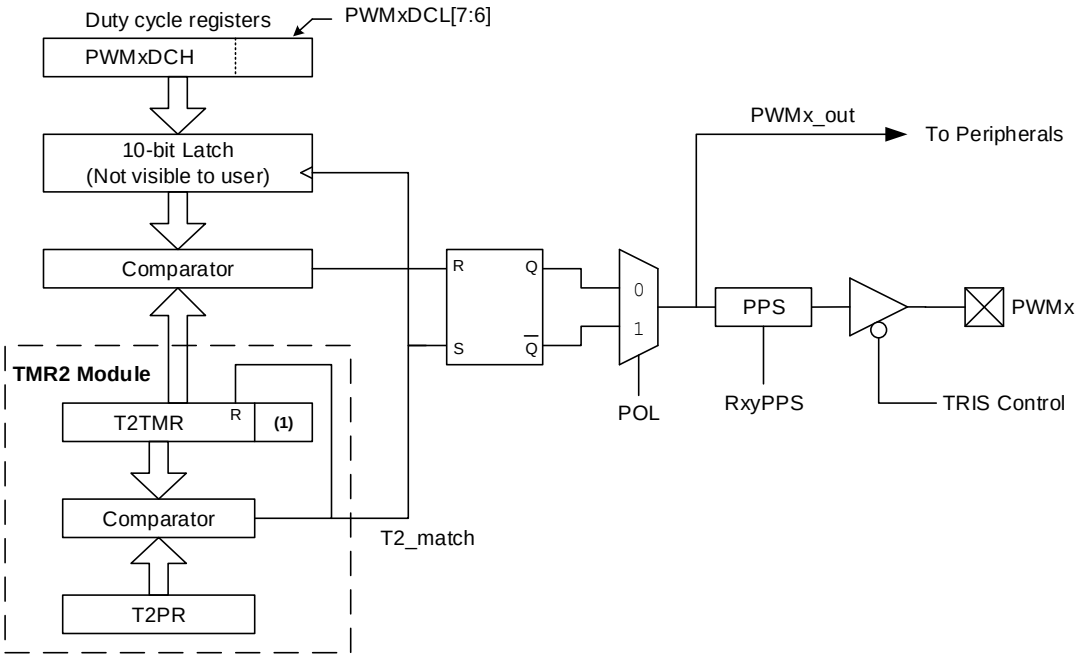
重要：必须将 PWMx 引脚对应的 TRIS 位清零，才能使能该引脚上的 PWM 输出。

每个 PWM 模块使用相同的定时器源 Timer2 来控制各个模块。

图 21-1 给出了 PWM 操作的简化框图。

图 21-2 给出了 PWM 信号的典型波形。

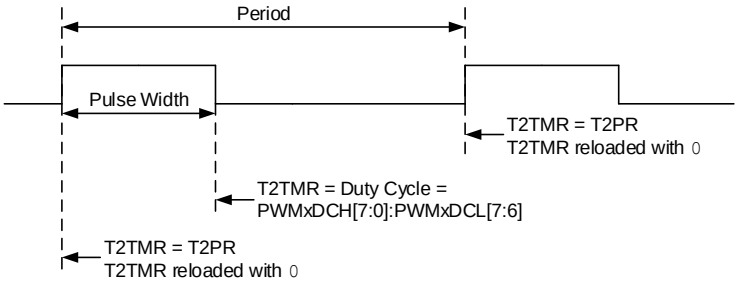
图 21-1. PWM 简化框图



注：

1. 8 位定时器与 F_{OSC} 生成的 2 位或内部预分频器的 2 位连接，以构成 10 位时基。

图 21-2. PWM 输出



关于如何设置该模块使之工作于 PWM 模式的详细步骤，请参见使用 PWMx 输出引脚设置 PWM 操作。

21.1. 基本操作

PWM 模块可产生一个 10 位分辨率的输出。PWMx 的定时器选择为 TMR2。T2TMR 和 T2PR 用于设置 PWM 的周期。PWMxDCL 和 PWMxDCH 寄存器用于配置占空比。周期由所有 PWM 模块共用，而占空比则独立进行控制。

重要：在确定 PWM 频率时不会用到 Timer2 后分频比。后分频器可用不同于 PWM 输出频率的频率进行伺服数据更新。

当 T2TMR 清零时，与 Timer2 相关的所有 PWM 输出都会置 1。当 T2TMR 等于相应 PWMxDCH（8 MSb）和 PWMxDCL[7:6]（2 LSb）寄存器指定的值时，每个 PWMx 都会清零。当值大于等于 T2PR 时，PWM 输出永远不会清零（占空比为 100%）。

重要：PWMxDCH 和 PWMxDCL 寄存器是双重缓冲的。当 T2TMR 与 T2PR 匹配时，缓冲区会发生更新。在定时器匹配发生之前更新两个寄存器时必须非常小心。

21.2. PWM 输出极性

输出极性通过将 POL 位置 1 进行翻转。

21.3. PWM 周期

PWM 周期由 T2PR 寄存器指定。PWM 周期可由公式 21-1 计算。为确保 PWM 正常工作，需要选择 $F_{OSC}/4$ 作为定时器的时钟输入。

公式 21-1. PWM 周期

$$PWM\ Period = [(T2PR\ ) + 1] \bullet 4 \bullet Tosc \bullet (TMR2\ \text{预分频}\ escale\ Value\ )$$

注：$T_{OSC} = 1/F_{OSC}$

当 T2TMR 中的值与 T2PR 中的值相等时，在下一个递增周期将发生以下 3 个事件：

- T2TMR 清零
- PWM 输出有效（例外情况：当 PWM 占空比 = 0%时，PWM 输出将保持无效）
- PWMxDCH 和 PWMxDCL 寄存器的值被锁存到缓冲区中。

重要：Timer2 后分频器对 PWM 操作没有任何作用。

21.4. PWM 占空比

PWM 占空比通过将一个 10 位值写入 PWMxDCH 和 PWMxDCL 寄存器对来指定。PWMxDCH 寄存器包含高 8 位，而 PWMxDCL[7:6]包含低 2 位。PWMxDCH 和 PWMxDCL 寄存器可以在任意时刻写入。

使用下面的公式计算 PWM 脉宽和 PWM 占空比。

公式 21-2. 脉冲宽度

$$Pulse\ Width = (PWMxDCH:PWMxDCL[7:6]) \bullet Tosc \bullet (TMR2\ 预分频\ Value)$$

注：$T_{OSC} = 1/F_{OSC}$

公式 21-3. 占空比

$$DutyCycleRatio = \frac{(PWMxDCH:PWMxDCL[7:6])}{4(T2PR+1)}$$

8 位定时器 T2TMR 寄存器与 $1/F_{OSC}$ 的低 2 位连接，通过 Timer2 预分频器进行调节，构成 10 位时基。如果 Timer2 预分频比设置为 1:1，则使用系统时钟。

21.5. PWM 分辨率

分辨率决定在给定周期内的可用占空比数。例如，10 位分辨率将产生 1024 个离散的占空比，而 8 位分辨率将产生 256 个离散的占空比。

当 T2PR 为 255 时，最大 PWM 分辨率为 10 位。分辨率是 T2PR 寄存器值的函数，如下所示。

公式 21-4. PWM 分辨率

$$分辨率solution = \frac{\log[4(T2PR+1)]}{\log(2)} bits$$

重要：如果脉宽值大于周期值，则指定的 PWM 引脚将保持不变。

表 21-1. PWM 频率和分辨率示例（F_{OSC} = 20 MHz）

PWM 频率	0.31 kHz	4.88 kHz	19.53 kHz	78.12 kHz	156.3 kHz	208.3 kHz
定时器预分频值	64	4	1	1	1	1
T2PR 值	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
最高分辨率（位）	10	10	10	8	7	6.6

表 21-2. PWM 频率和分辨率示例（F_{OSC} = 8 MHz）

PWM 频率	0.31 kHz	4.90 kHz	19.61 kHz	76.92 kHz	153.85 kHz	200.0 kHz
定时器预分频值	64	4	1	1	1	1
T2PR 值	0x65	0x65	0x65	0x19	0x0C	0x09
最高分辨率（位）	8	8	8	6	5	5

21.6. 休眠模式下的操作

在休眠模式下，T2TMR 寄存器将不会递增，模块状态也不会改变。如果 PWMx 引脚正在驱动一个值，则会继续驱动该值。当器件被唤醒时，T2TMR 将从先前状态继续。

21.7. 改变系统时钟频率

PWM 频率是由系统时钟频率（F_{OSC}）产生的。系统时钟频率的任何改变都将导致 PWM 频率的改变。

21.8. 复位的影响

任何复位都将强制所有端口为输入模式，并强制 PWM 寄存器为其复位状态。

21.9. 使用 PWMx 输出引脚设置 PWM 操作

当使用 PWMx 引脚将模块配置为 PWM 操作时，请按以下步骤操作：

1. 通过将相关的 TRIS 位置 1，禁止 PWMx 引脚输出驱动器。
2. 清零 PWMxCON 寄存器。
3. 将 PWM 周期值装入 T2PR 寄存器。
4. 将 PWM 占空比值装入 PWMxDCH 寄存器和 PWMxDCL 寄存器的 bit[7:6]。
5. 配置并启动 Timer2：
   - 清零 PIRx 寄存器的 TMR2IF 中断标志位。(1)
   - 需使用 T2CLKCON 寄存器选择 $F_{OSC}/4$ 作为定时器时钟源。只有这样才能确保 PWM 模块正常工作。
   - 用 Timer2 预分频值配置 T2CON 寄存器的 CKPS 位。
   - 通过将 T2CON 寄存器的 ON 位置 1 来使能 Timer2。
6. 使能 PWM 输出引脚并等待至 Timer2 溢出，PIRx 寄存器的 TMR2IF 位置 1。(2)
7. 通过将相关的 TRIS 位清零并将所需的引脚 PPS 控制位置 1，使能 PWMx 引脚输出驱动器。
8. 通过将相应值装入 PWMxCON 寄存器来配置 PWM 模块。

注：

1. 要在第一个 PWM 输出中发送完整的占空比和周期，必须按给出的顺序执行上述步骤。如果并非必须以一个完整 PWM 信号开始，则用步骤 8 来代替步骤 4。
2. 对于仅针对其他外设的操作，请禁止 PWMx 引脚输出。

21.9.1. PWMx 引脚配置

所有 PWM 输出都与 PORT 数据锁存器复用。用户必须通过清零相关的 TRIS 位将引脚配置为输出。

21.10. 为其他器件外设设置 PWM 操作

当配置模块 PWM 操作以供其他器件外设使用时，请按以下步骤操作：

1. 通过将相关的 TRIS 位置 1，禁止 PWMx 引脚输出驱动器。
2. 清零 PWMxCON 寄存器。
3. 将 PWM 周期值装入 T2PR 寄存器。
4. 将 PWM 占空比值装入 PWMxDCH 寄存器和 PWMxDCL 寄存器的 bit[7:6]。
5. 配置并启动 Timer2：
   - 清零 PIRx 寄存器的 TMR2IF 中断标志位。(1)
   - 需使用 T2CLKCON 寄存器选择 $F_{OSC}/4$ 作为定时器时钟源。只有这样才能确保 PWM 模块正常工作。
   - 用 Timer2 预分频值配置 T2CON 寄存器的 CKPS 位。
   - 通过将 T2CON 寄存器的 ON 位置 1 来使能 Timer2。
6. 等待至 Timer2 溢出，将 PIRx 寄存器的 TMR2IF 位置 1。(1)

7. 通过将相应值装入 PWMxCON 寄存器来配置 PWM 模块。

注：

1. 要在第一个 PWM 输出时发送完整的占空比和周期，设置过程必须包含上述步骤。如果在第一个输出时以完整的 PWM 信号起始并非至关重要，则可以忽略步骤 6。

21.11. 寄存器定义：PWM 控制

下表列出了 PWM 外设的长位名称前缀。更多信息，请参见“**长位名称**”一节。

表 21-3. PWM 位名称前缀

外设	位名称前缀
PWM3	PWM3
PWM4	PWM4

21.11.1. PWMxCON

名称： PWMxCON
偏移量： 0x316,0x31A

PWM 控制寄存器

位	7	6	5	4	3	2	1	0
	EN		OUT	POL				
访问	R/W		R	R/W				
复位	0		0	0				

Bit 7 – EN PWM 模块使能位

值	说明
1	使能 PWM 模块
0	禁止 PWM 模块

Bit 5 – OUT PWM 模块输出电平
指示读取该位时的 PWM 模块输出电平

Bit 4 – POL PWM 输出极性选择位

值	说明
1	PWM 输出反相
0	PWM 输出正常

21.11.2. PWMxDC

名称： PWMxDC
偏移量： 0x314,0x318

PWM 占空比寄存器

位	15	14	13	12	11	10	9	8
	DCH[7:0]							
访问								
复位	x	x	x	x	x	x	x	x

位	7	6	5	4	3	2	1	0
	DCL[1:0]							
访问								
复位	x	x						

Bit 15:8 – DCH[7:0] PWM 占空比高 8 位
这些位是 PWM 占空比的高 8 位。

复位状态： POR/BOR = xxxxxxxx
所有其他复位 = uuuuuuuu

Bit 7:6 – DCL[1:0] PWM 占空比低 2 位
这两位是 PWM 占空比的低 2 位。

复位状态： POR/BOR = xx
所有其他复位 = uu

21.12. 寄存器汇总——PWM

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0313	保留									
0x0314	PWM3DC	7:0	DCL[1:0]							
		15:8	DCH[7:0]							
0x0316	PWM3CON	7:0	EN		OUT	POL				
0x0317	保留									
0x0318	PWM4DC	7:0	DCL[1:0]							
		15:8	DCH[7:0]							
0x031A	PWM4CON	7:0	EN		OUT	POL				

22. MSSP——主同步串行端口模块

主同步串行端口（MSSP）模块是用于和其他外设或单片机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 模块支持以下两种工作模式：

- 串行外设接口（Serial Peripheral Interface，SPI）
- I^2C

SPI 接口可在主模式或从模式下工作，支持以下特性：

- 可选的时钟奇偶校验
- 从选择同步（仅限从模式）
- 从器件的菊花链连接

I^2C 接口可在主模式或从模式下工作，支持以下模式和特性：

- 不应答字节（从模式）
- 有限多主器件支持
- 7 位和 10 位寻址模式
- 启动和停止中断
- 中断屏蔽
- 时钟延长
- 总线冲突检测
- 广播呼叫地址匹配
- 地址掩码
- 地址保持和数据保持模式
- 可选的 SDA 保持时间

22.1. SPI 模式概述

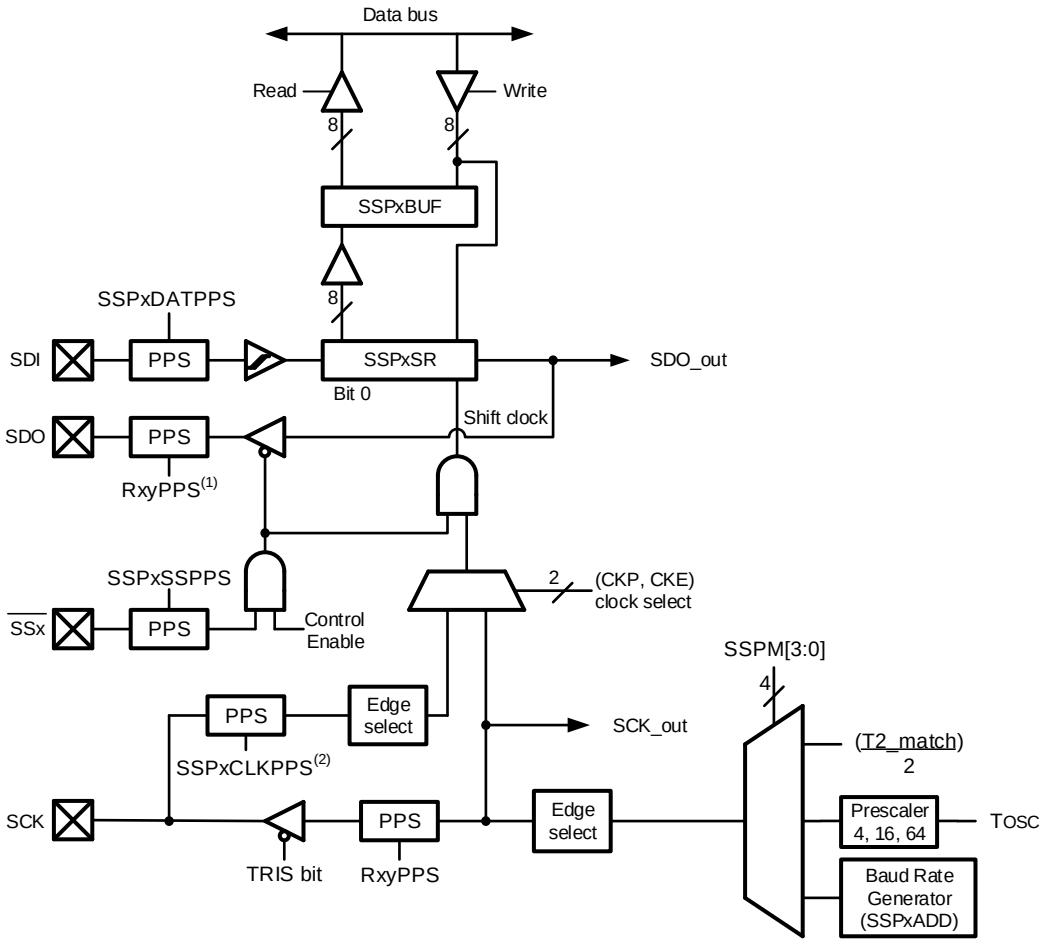
串行外设接口（SPI）是以全双工模式工作的同步串行数据通信总线。器件在由主器件启动通信的主/从环境中进行通信。进行通信的从器件通过从选择功能来选择。

SPI 总线指定 4 个信号连接：

- 串行时钟（SCK）
- 串行数据输出（SDO）
- 串行数据输入（SDI）
- 从器件选择（$\overline{SS}$）

图 22-1 给出了 MSSP 模块在 SPI 模式下工作时的框图。

图 22-1. MSSP 框图（SPI 模式）

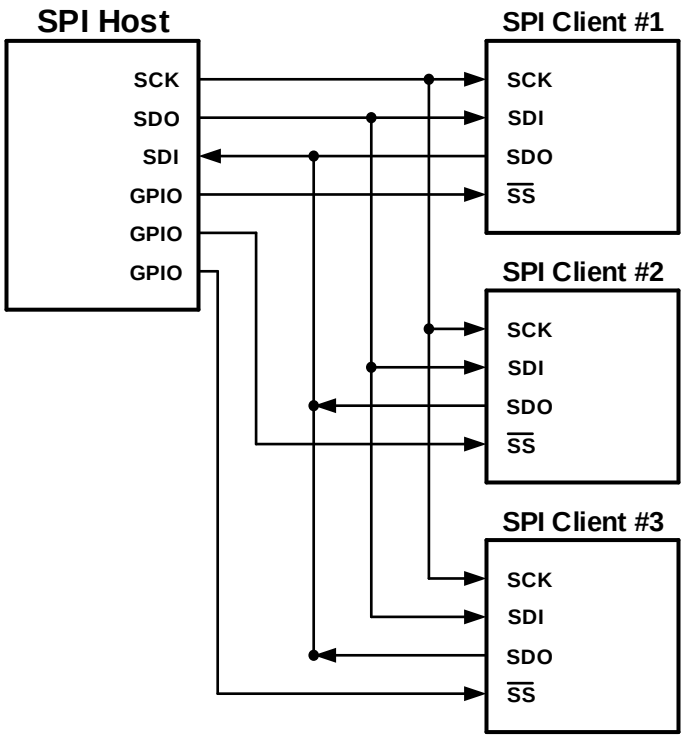


Notes: 1. Output selection for Host mode.
2. Input selection for Client and Host modes.

SPI 总线工作时可以有一个主器件，以及一个或多个从器件。使用多个从器件时，从主器件到每个从器件都需要独立的从器件选择连接。主器件每次仅选择一个从器件。大多数从器件都具有三态输出，所以在未选择它们时，它们的输出信号与总线断开。

图 22-2 给出了主器件和多个从器件之间的典型连接。

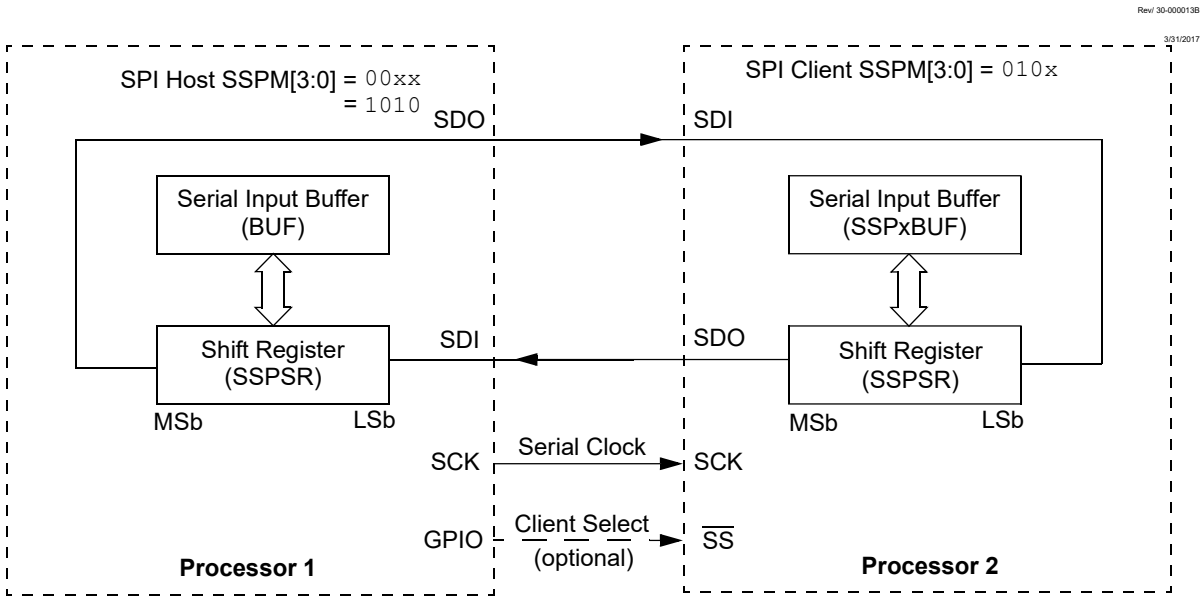
图 22-2. SPI 主器件与多个从器件之间的连接



发送操作涉及两个移位寄存器（8 位大小）：一个在主器件中，一个在从器件中。每次只移出一位数据，并且首先移出最高有效位（MSb）。与此同时，新的最低有效位（LSb）移入同一寄存器。

图 22-3 给出了分别配置为主器件和从器件的两个处理器之间的典型连接。

图 22-3. SPI 主器件/从器件连接



数据在所设定的时钟边沿从两个移位寄存器移出，并在相反的时钟边沿锁存。

主器件在与从器件的 SDI 输入引脚连接的 SDO 输出引脚上发出信息，随后由从器件 SDI 输入引脚接收该信息。从器件在与主器件的 SDI 输入引脚连接的 SDO 输出引脚上发出信息，随后由主器件 SDI 输入引脚接收该信息。

要开始进行通信，主器件需要送出其移位寄存器中的 MSb 以及时钟信号。主器件和从器件都需要配置为相同的时钟极性。在每个 SPI 时钟周期中，会发生全双工数据发送。这意味着，在主器件从其移位寄存器中发送出 MSb（在其 SDO 引脚上），从器件读取该位并将它保存为其移位寄存器的 LSb 的同时，从器件也会从其移位寄存器中发送出 MSb（在其 SDO 引脚上），而主器件也会读取该位并将它保存为其移位寄存器的 LSb。

移出 8 位数据后，主器件和从器件交换了寄存器值。如果有更多数据要交换，移位寄存器装入新数据并重复此过程。

数据是有意义还是无意义（无效数据），取决于应用软件。这就导致以下三种数据传输情形：

- 主器件发送有用的数据，从器件发送无效数据
- 主器件发送有用的数据，从器件发送有用的数据
- 主器件发送无效数据，从器件发送有用的数据

发送执行时间必须为 8 个时钟周期的倍数。在没有更多数据需要发送时，主器件会停止发送时钟信号，并取消选择从器件。

每个与总线连接的从器件，如果尚未通过其从器件选择线路被选定，则会丢弃时钟和发送信号，且不会发出任何自己的数据。

22.1.1. SPI 模式寄存器

MSSP 模块有 6 个寄存器用于 SPI 工作模式。具体包括：

- MSSP 状态寄存器（SSPxSTAT）
- MSSP 控制寄存器 1（SSPxCON1）
- MSSP 控制寄存器 3（SSPxCON3）

- MSSP 数据缓冲寄存器（SSPxBUF）
- MSSP 地址寄存器（SSPxADD）
- MSSP 移位（SSPSR）寄存器（不可直接访问）

SSPxCON1 和 SSPxSTAT 是在 SPI 模式下工作的控制寄存器和状态寄存器。SSPxCON1 寄存器是可读写的。SSPxSTAT 的低 6 位是只读的。SSPxSTAT 的高 2 位是可读写的。

5 种 SPI 主模式之一使用 SSPxADD 值来确定波特率发生器的时钟频率。有关波特率发生器的更多信息，请参见波特率发生器。

SSPSR 是用来移入/移出数据的移位寄存器。SSPxBUF 提供了对 SSPSR 寄存器的间接访问。SSPxBUF 是缓冲寄存器，可用于数据字节的写入或读出。

在接收操作中，SSPSR 和 SSPxBUF 一起组成了缓冲接收区。在 SSPSR 接收到一个完整的字节后，该字节被传送到 SSPxBUF 且 SSPxIF 中断标志位置 1。

在发送期间，SSPxBUF 不是可缓冲的。对 SSPxBUF 的写操作相当于同时写入 SSPxBUF 和 SSPSR。

22.1.2. SPI 模式工作原理

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（SSPxCON1[5:0]和 SSPxSTAT[7:6]）来指定这些选项。这些控制位用于指定以下选项：

- 主模式（SCK 作为时钟输出）
- 从模式（SCK 作为时钟输入）
- 时钟极性（SCK 的空闲状态）
- 输入数据的采样阶段（数据输出时间的中间或末端）
- 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）

要使能串行端口，SSP 使能（SSPEN）位必须置 1。要复位或重新配置 SPI 模式，先将 SSPEN 位清零，重新初始化 SSPxCONy 寄存器，然后再将 SSPEN 位置 1。SDI、SDO、SCK 和 $\overline{SS}$ 串行端口引脚通过 PPS 控制选择。要将引脚用作串行端口功能，必须正确设置其中一些引脚的数据方向位（在 TRIS 寄存器中）：

- 对于 SDI，必须将相应的 TRIS 位置 1
- 对于 SDO，必须将相应的 TRIS 位清零
- 对于 SCK（主模式），必须将相应的 TRIS 位清零
- 对于 SCK（从模式），必须将相应的 TRIS 位置 1
- RxyPPS 和 SSPxCLKPPS 控制必须选择相同的引脚
- 对于 $\overline{SS}$，必须将相应的 TRIS 位置 1

对于不需要的任何串行端口功能，可通过将对应的数据方向（TRIS）寄存器设置为相反值来屏蔽。

MSSP 由一个发送/接收移位寄存器（SSPSR）和一个缓冲寄存器（SSPxBUF）组成。SSPSR 将数据移入/移出器件，MSb 在前。在数据接收完毕前，SSPxBUF 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 SSPxBUF 寄存器。然后，缓冲区满状态（BF）位和 MSSP 中断标志（SSPxIF）位被置 1。这种双重缓冲数据接收方式允许在读取刚接收的数据之前就开始接收下一个字节。在发送/接收数据期间对 SSPxBUF 寄存器的任何写操作都将被忽略，同时写冲突检测（WCOL）位被置 1。用户软件必须将 WCOL 位清零，才能使后续对 SSPxBUF 寄存器的写操作成功完成。

为确保应用软件能接收有效数据，在下一个要发送的数据字节写入 SSPxBUF 之前，必须读取 SSPxBUF 中现有的数据。BF 位用于指示何时 SSPxBUF 装入了接收到的数据（发送完成）。SSPxBUF 中的数据被读取

后，BF 位被清零。如果 SPI 仅作为一个发送器，则不必理会该数据。MSSP 中断用于确定发送/接收何时结束。如果不打算使用中断方法，用软件查询的方法同样可确保不会发生写冲突。

重要：SSPSR 不能直接读写，只能通过寻址 SSPxBUF 寄存器访问。

22.1.2.1. SPI 主模式

因为主器件控制 SCK 线，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 22-3 中的处理器 2）在何时广播数据。

在主模式下，数据一写入 SSPxBUF 寄存器就发送/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 寄存器按所设定的时钟速率，对 SDI 引脚上的信号进行连续移入。每接收到一个字节，就将其装入 SSPxBUF 寄存器（中断和状态位相应置 1）。

通过适当地设定时钟极性选择（CKP）位和 SPI 时钟边沿选择（CKE）位，可以选择时钟极性。图 22-4 给出了四种时钟配置。当 CKE 位置 1 时，SDO 数据在 SCK 上出现时钟边沿之前半个时钟周期即有效，并且在从有效时钟状态切换到空闲时钟状态时进行发送。当 CKE 位清零时，SDO 数据在 SCK 上出现时钟边沿时有效，并且在从空闲时钟状态切换到有效时钟状态时进行发送。

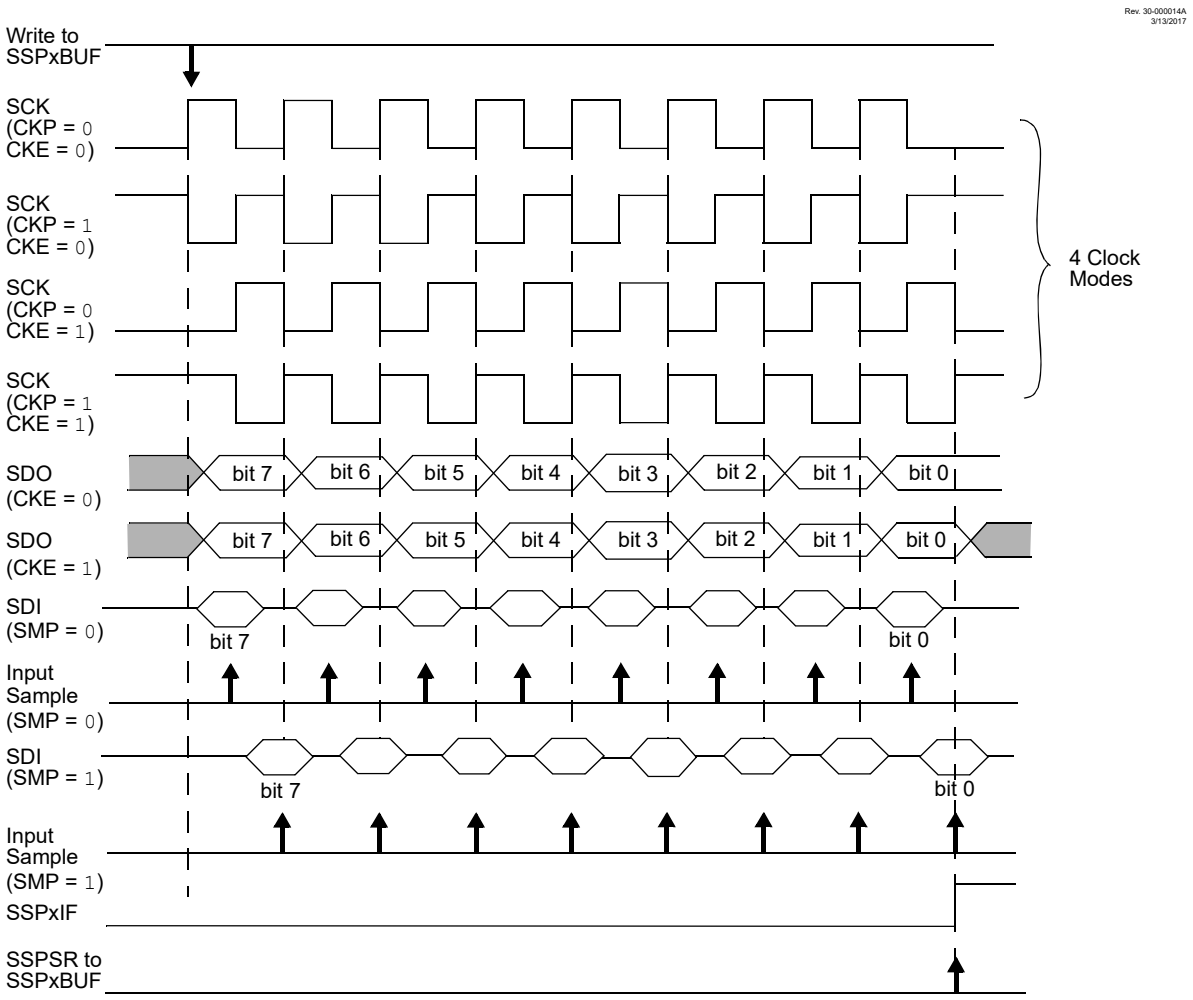
SPI 数据输入采样（SMP）位决定何时采样 SDI 输入。当 SMP 置 1 时，在数据输出时间的末端采样输入数据。当 SMP 清零时，在数据输出时间的中间采样输入数据。

SPI 时钟速率（比特率）可由用户编程为以下几种之一：

- $F_{OSC}/4$（或 T_{CY}）
- $F_{OSC}/16$（或 $4 * T_{CY}$）
- $F_{OSC}/64$（或 $16 * T_{CY}$）
- Timer2 输出/2
- $F_{OSC}/(4 * (SSPxADD + 1))$

重要：在主模式下，送至 SCK 引脚的时钟信号输出也是送至外设的时钟信号输入。通过 RxyPPS 寄存器选择作为输出的引脚也必须通过 SSPxCLKPPS 寄存器选择作为外设输入。

图 22-4. SPI 模式波形图（主模式）



22.1.2.2. SPI 从模式

在从模式下，当 SCK 上出现外部时钟脉冲时发送和接收数据。当最后一位数据被锁存后，SSPxIF 中断标志位置 1。

在 SPI 从模式下使能该模块前，时钟线必须处于相应的空闲状态。时钟线可通过读 SCK 引脚来查看。空闲状态由 CKP 位决定。

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。移位寄存器通过 SCK 引脚输入提供时钟，当接收到一个字节时，器件会产生中断。如果允许中断，器件会从休眠模式唤醒。

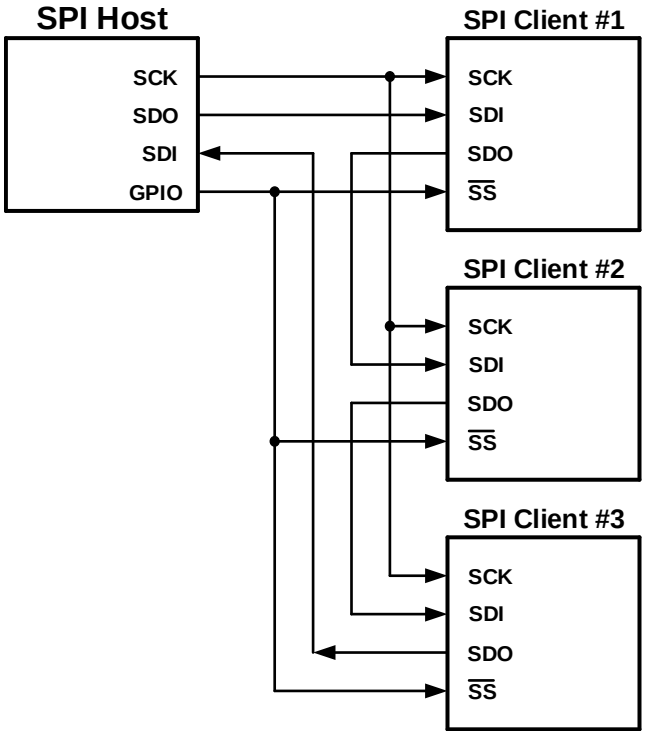
22.1.2.3. 菊花链配置

SPI 总线有时会采用菊花链配置进行连接。第一个从器件的输出与第二个从器件的输入连接，第二个从器件的输出与第三个从器件的输入连接，依此类推。最后一个从器件的输出与主器件的输入连接。在第二组时钟脉冲期间，每个从器件会送出在第一组时钟脉冲期间所接收数据的精确副本。整个链充当一个大的通信移位寄存器。菊花链功能只需要从主器件引出一条从选择线。

在菊花链配置中，从器件只需要总线上最近的一个字节。将缓冲区改写使能（BOEN）位置 1 时，即使尚未读取前一个字节，也允许数据写入 SSPxBUF 寄存器。这使软件可以忽略不适用于它的数据。

图 22-5 给出了在 SPI 模式下工作时典型菊花链连接的框图。

图 22-5. SPI 菊花链连接



22.1.2.4. 从选择同步

从选择也可以用于对通信进行同步（见图 22-6）。从选择线会保持高电平，直到主器件准备好进行通信。当从选择线下拉为低电平时，从器件就知道新的数据发送正在启动。

如果从器件未能正确地接收到通信，它会在从选择线恢复为高电平状态、数据发送结束时发生复位。然后，从器件会在从选择线再次下拉为低电平时准备好接收新的发送数据。如果不使用从选择线，则会存在从器件最终与主器件脱离同步的风险。如果从器件丢失了某个位，则在之后的数据发送中，它将总是偏离一位。使用从选择线可以让从器件和主器件在每次发送开始时保持同步。

$\overline{SS}$ 引脚允许器件工作于同步从模式。SPI 必须处于从模式，并使能 $\overline{SS}$ 引脚控制（MSSP 模式选择（SSPM）位 = 0100）。

当 $\overline{SS}$ 引脚为低电平时，使能数据的发送和接收，同时驱动 SDO 引脚。

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。根据具体应用，可能需要使用外部上拉/下拉电阻。

当 SPI 模块复位时，位计数器会被强制为 0。这可以通过将 $\overline{SS}$ 引脚强制设为高电平或清零 SSPEN 位实现。

重要：

1. 当 SPI 处于从模式且使能 $\overline{SS}$ 引脚控制（SSPM = 0100）时，如果 $\overline{SS}$ 引脚设置为 V_{DD}，SPI 模块将会复位。
2. 当 SPI 用于从模式且 CKE 置 1 时，用户必须使能 $\overline{SS}$ 引脚控制（见图 22-8）。如果 CKE 清零，$\overline{SS}$ 引脚控制可选（见图 22-7）。
3. 工作于 SPI 从模式下时，SMP 位必须保持清零。

图 22-6. 从选择同步波形图

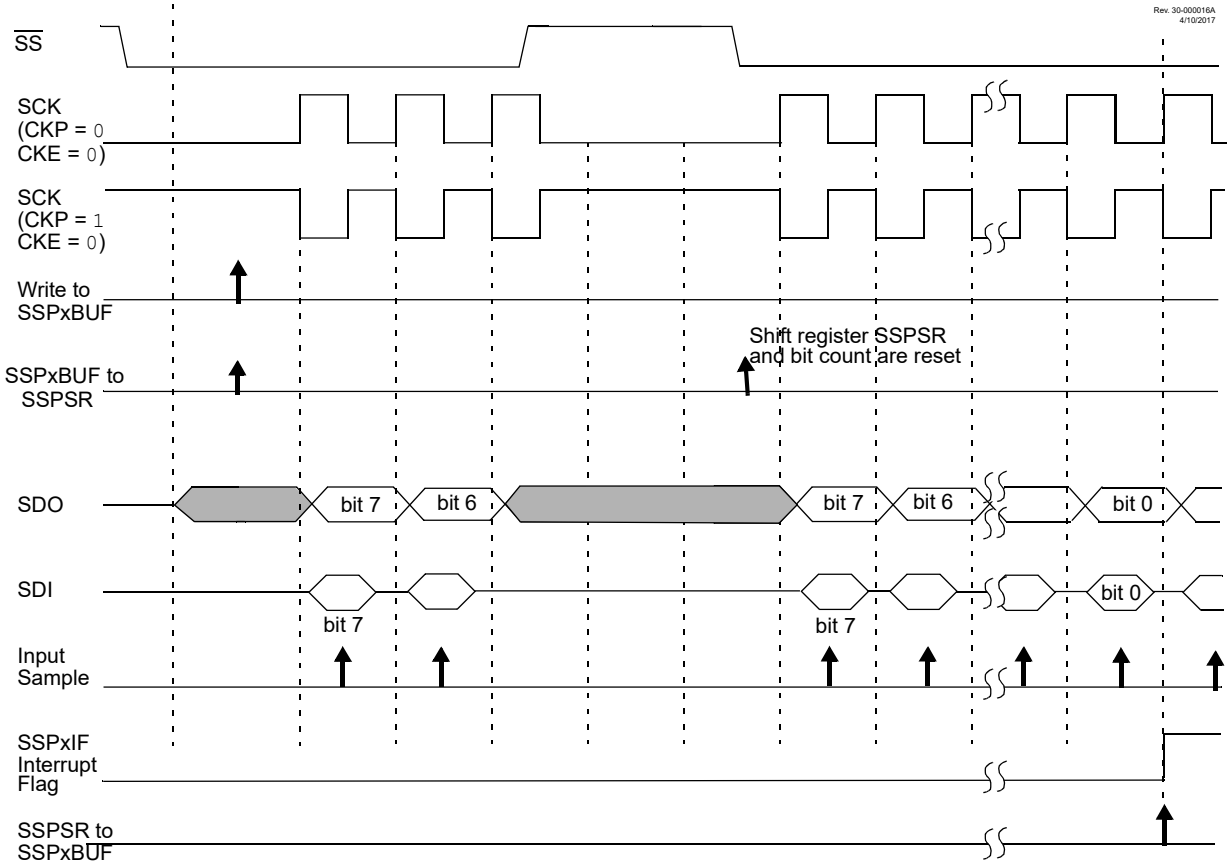


图 22-7. SPI 模式波形图（从模式，CKE = 0）

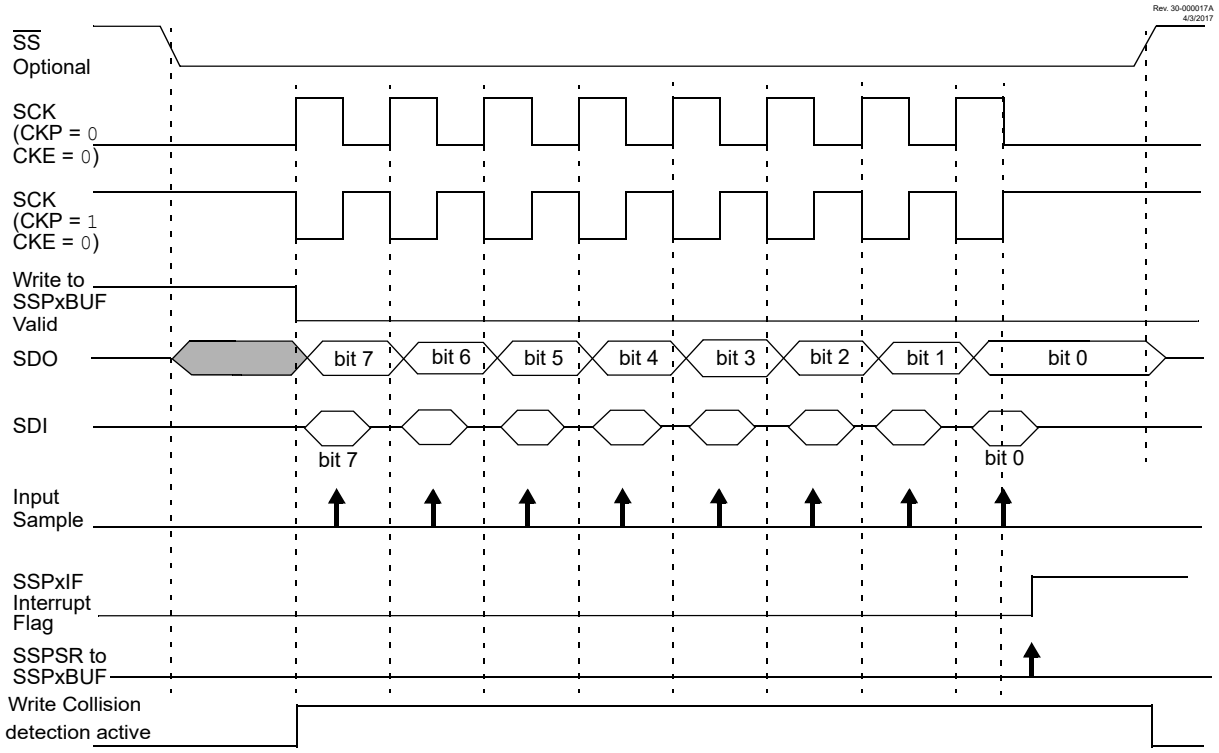
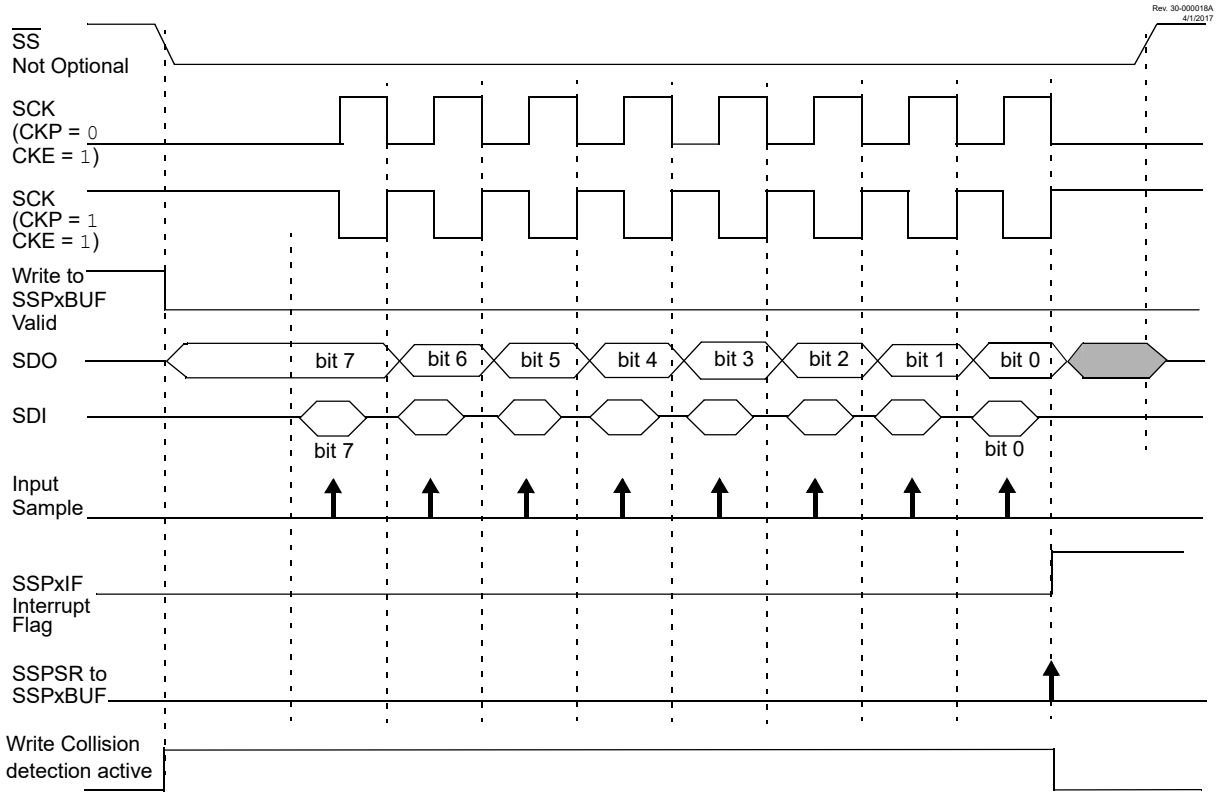


图 22-8. SPI 模式波形图（从模式，CKE = 1）



22.1.2.5. SPI 在休眠模式下的工作

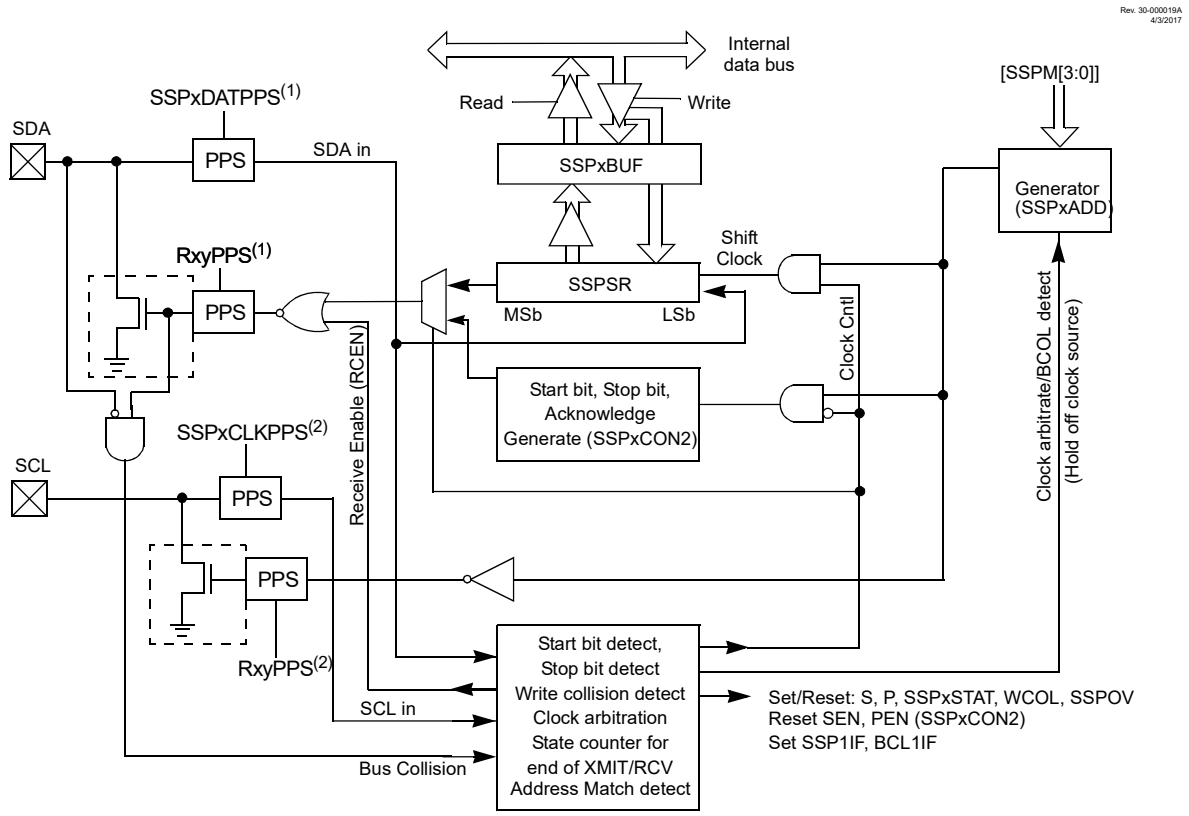
在 SPI 主模式下，当选择休眠模式时，所有模块的时钟都将暂停，并且在器件被唤醒前，发送/接收将保持此暂停状态。器件返回到运行模式之后，模块将恢复发送和接收数据。

在 SPI 从模式下，SPI 发送/接收移位寄存器与器件异步工作。这可使器件置于休眠模式，且数据被移入 SPI 发送/接收移位寄存器。当接收到全部 8 位数据时，MSSP 中断标志位将置 1，并且如果允许该中断，则将唤醒器件。

22.2. I²C 模式概述

I²C 总线是一种多主器件串行数据通信总线。器件可以在主/从环境下通信，在该环境下，由主器件发起通信。从器件通过寻址进行控制。 图 22-9 和图 22-10 分别显示了 I²C 主模式和从模式的框图。

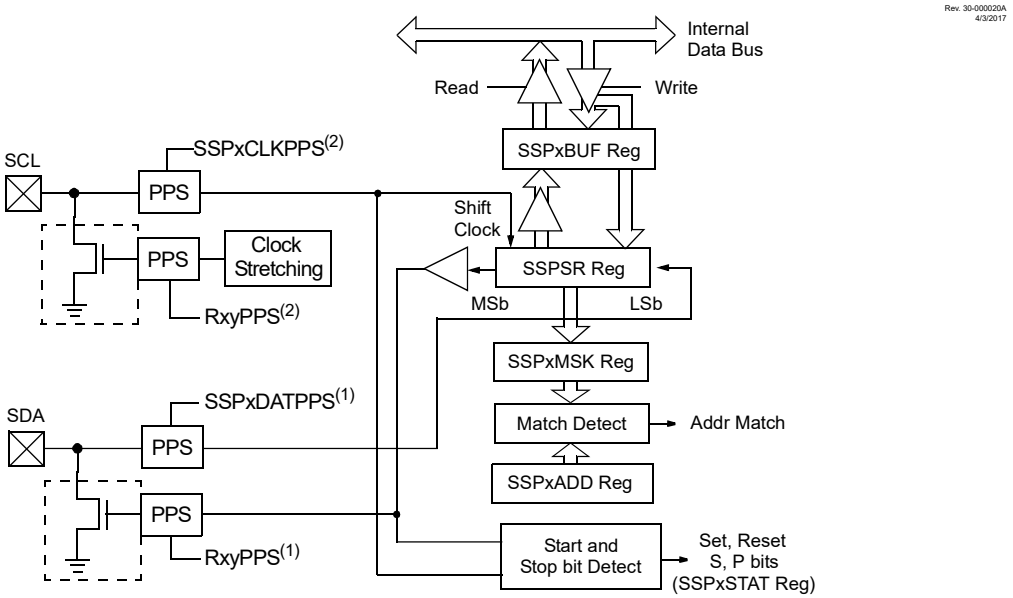
图 22-9. MSSP 框图（I²C 主模式）



Notes: 1. SDA pin selections must be the same for input and output.

2. SCL pin selections must be the same for input and output.

图 22-10. MSSP 框图（I²C 从模式）



Notes: 1. SDA pin selections must be the same for input and output.
2. SCL pin selections must be the same for input and output.

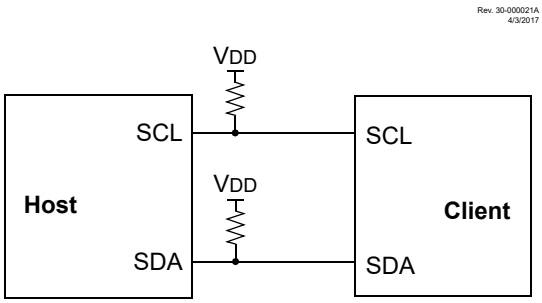
I²C 总线指定两种信号连接：

- 串行时钟（SCL）
- 串行数据（SDA）

SCL 和 SDA 连接都是双向的漏极开路线，它们都需要通过上拉电阻连接到电源电压。线路下拉到地时，信号视为逻辑 0；线路保持悬空时，信号视为逻辑 1。

图 22-11 给出了分别配置为主器件和从器件的两个处理器之间的典型连接。

图 22-11. I²C 主/从连接



I²C 总线工作时可以有一个主器件，以及一个或多个从器件。

对于给定器件，有四种可能的工作模式：

- 主发送模式（主器件向从器件发送数据）
- 主接收模式（主器件从从器件接收数据）
- 从发送模式（从器件向主器件发送数据）

- 从接收模式（从器件从主器件接收数据）

要开始进行通信，主器件需发送启动位，后跟需要通信的从器件的地址字节。启动条件的指示方式为 SCL 线保持为高电平，SDA 线由高至低发生跳变。地址和数据字节随后送出，先发送 MSb，后跟单个读/写信息（R/$\overline{W}$）位，该位决定主器件是向从器件发送数据还是从从器件接收数据。R/$\overline{W}$ 位作为逻辑 1 发出，当主器件想写入数据到从器件时，该位作为逻辑 0 发出。

如果总线上存在所请求的从器件，从器件会使用应答序列（也称为 $\overline{ACK}$）进行响应。应答序列是低电平有效信号，它会将 SDA 线保持为低电平，用于指示发送器从器件已接收到发送数据，并已准备好接收更多数据。主器件随后继续向从器件发送数据或从从器件接收数据。

数据位的跳变总是在 SCL 线保持低电平时执行。在 SCL 线保持高电平时发生的跳变用于指示启动条件和停止条件。

如果主器件希望向从器件写入数据，则会重复发送一个字节的数据，而从器件在接收每个字节之后使用 $\overline{ACK}$ 序列进行响应。在该示例中，主器件处于主发送模式，从器件处于从接收模式。

如果主器件希望从从器件读取数据，则会从从器件重复接收一个字节的数据，并在接收每个字节之后使用 $\overline{ACK}$ 序列进行响应。在此示例中，主器件处于主接收模式，从器件处于从发送模式。

在传输最后一个数据字节之后，主器件可以通过发送停止条件来结束数据发送。如果主器件处于接收模式，它会发送停止条件代替最后一个 $\overline{ACK}$ 序列。停止条件的指示方式为 SCL 线保持高电平，SDA 线由低至高发生跳变。

在某些情况下，主器件可能希望维持对总线的控制，并重新启动另一次数据发送。如果是这样，主器件可以在处于接收模式时，发送重复启动条件来代替停止条件或最后一个 $\overline{ACK}$ 序列。

I^2C 总线规定了三种报文协议：

- 主器件向从器件写数据的单一报文
- 主器件从从器件读数据的单一报文
- 主器件对一个或多个从器件启动至少两次写操作、两次读操作，或者读写操作组合的组合报文

22.2.1. I^2C 模式寄存器

MSSP 模块有 8 个寄存器用于 I^2C 工作模式。

这些寄存器包括：

- MSSP 状态寄存器（SSPxSTAT）
- MSSP 控制 1 寄存器（SSPxCON1）
- MSSP 控制 2 寄存器（SSPxCON2）
- MSSP 控制 3 寄存器（SSPxCON3）
- 串行接收/发送缓冲区寄存器（SSPxBUF）
- MSSP 地址寄存器（SSPxADD）
- I^2C 从模式地址掩码寄存器（SSPxMSK）
- MSSP 移位（SSPSR）寄存器——不可直接访问

SSPxCON1、SSPxCON2、SSPxCON3 和 SSPxSTAT 是在 I^2C 模式下工作的控制寄存器和状态寄存器。SSPxCON1、SSPxCON2 和 SSPxCON3 寄存器可读写。SSPxSTAT 的低 6 位是只读的。SSPxSTAT 的高 2 位是可读写的。SSPxMSK 保存地址比较中使用的从器件地址掩码值。当 MSSP 配置为 I^2C 从模式时，SSPxADD 会包含从器件地址。当 MSSP 配置为主模式时，SSPxADD 将用作波特率发生器重载值。

SSPSR 是用来移入/移出数据的移位寄存器。SSPxBUF 是缓冲寄存器，可用于数据字节的写入或读出。在接收操作中，SSPSR 和 SSPxBUF 一起组成了双缓冲接收器。在 SSPSR 接收到一个完整的字节后，该字节

被传送到 SSPxBUF 且 SSPxIF 中断标志位置 1。在发送期间，SSPxBUF 不是双缓冲的。对 SSPxBUF 的写操作相当于同时写入 SSPxBUF 和 SSPSR。

22.2.2. I²C 模式工作原理

所有 MSSP I²C 通信都是面向字节的，且首先移出最高有效位。8 个 SFR 寄存器和 2 个中断标志用作模块与单片机和用户软件的接口。该模块还采用了 2 个引脚（SDA 和 SCL）与其他外部 I²C 器件通信。

22.2.2.1. I²C 术语定义

在 I²C 通信的描述中存在一些用语和术语，它们具有特定于 I²C 的定义。下面定义了词语的用法，在本文档其他部分中，将不加说明地使用它们。此表根据 Philips/NXP I²C 规范改写。

表 22-1. I²C 术语

术语	说明
发送器	将数据移出到总线上的器件
接收器	从总线上移入数据的器件
主模式	启动数据传输、产生时钟信号和终止数据传输的器件
从模式	主器件寻址到的器件
多主器件	有多个器件可以启动数据传输的总线
仲裁	确保一个时刻只有一个主器件控制总线的过程。赢得仲裁可确保报文不被破坏。
同步	用于将总线上两个或更多器件的时钟进行同步的过程
空闲	没有任何主器件在控制总线，并且 SDA 和 SCL 线均为高电平
活动	每当有一个或多个主器件在控制总线时
被寻址的从器件	已接收到匹配地址并且正在由主器件提供时钟的从器件
匹配地址	从器件接收到的地址字节与存储在 SSPxADD 中的值相匹配
写请求	从器件接收到 R/$\overline{W}$ 位清零的匹配地址，并已准备随时钟移入数据
读请求	主器件发送 R/$\overline{W}$ 位置 1 的地址字节，表示要求从器件随时钟移出数据。从器件在接收到该地址字节后会立即移出所有数据字节，直到发生重复启动或停止条件。
时钟延长	总线上的器件通过将 SCL 保持为低电平来暂停通信的时间
总线冲突	每当模块进行输出并期望 SDA 线为高电平，却采样到 SDA 线为低电平时

22.2.2.2. 字节格式

I²C 模式下的所有通信都采用 9 位数据段。从主器件向从器件（或者反之）发送一个字节之后，将会送回一个应答序列。在 SCL 线的第 8 个下降沿之后，器件将数据输出到 SDA 引脚，将该引脚变为输入引脚，然后在下一个时钟脉冲时读取应答值。

时钟信号 SCL 由主器件提供。在 SCL 信号为低电平时，数据可以有效地更改，并且在时钟上升沿进行采样在 SCL 线为高电平时，SDA 线上的电平变化定义总线上的一些特殊条件，例如启动条件或停止条件。

22.2.2.3. SDA 和 SCL 引脚

在 SSPEN 位置 1 的情况下选择任意 I²C 模式时，SCL 和 SDA 引脚将会强制设为漏极开路。这些引脚必须通过将相应 TRIS 位置 1 的方式配置为输入引脚。

重要：通过 PPS 外设，可以选择任意器件引脚用于 SDA 和 SCL 功能。这些功能是双向的。SDA 输入通过 SSPxDATPPS 寄存器进行选择。SCL 输入通过 SSPxCLKPPS 寄存器进行选择。输出通过 RxyPPS 寄存器进行选择。用户需要负责确保在进行选择时，使每个功能的输入和输出处于同一引脚上。

22.2.2.4. SDA 保持时间

SDA 引脚的保持时间通过 SDA 保持时间选择（SDAHT）位进行选择。保持时间是 SDA 在 SCL 的下降沿之后保持有效的时间。将 SDAHT 位置 1 可以选择最低 300 ns 的较长保持时间，这对于电容较大的总线会有帮助。

22.2.2.5. 时钟延长

当总线上的某个器件将 SCL 线保持为低电平而有效暂停通信时，就发生了时钟延长现象。从器件可以延长时钟，以便可以有更多时间来处理数据或准备响应主器件。时钟延长时不关心主器件的工作，因为任何时候只需总线上主器件处于活动状态但是不传输数据就可以被认为是时钟延长。由从器件进行的任何时钟延长对于主器件软件都是不可见的，都由产生 SCL 的硬件进行处理。

CKP 位用于通过软件控制延长。每当 CKP 位清零时，模块就会等待 SCL 线变为低电平，然后保持低电平状态不变。将 CKP 置 1 将会释放 SCL，允许继续进行通信。

22.2.2.6. 仲裁

每个主器件都必须监视总线上是否出现启动位和停止条件。如果器件检测到总线正忙，则在总线恢复为空闲状态之前，它无法开始新的报文。

但是，可能会有两个主器件尝试同时或几乎同时启动数据发送。发生这种情况时，将会开始仲裁过程。每个发送器会检查 SDA 数据线的电平，并将它与自己期望的电平进行比较。发现两个电平不匹配的第一个发送器会在仲裁中失败，必须停止在 SDA 线上发送数据。

例如，如果一个发送器将 SDA 线保持为逻辑 1（SDA 保留悬空），而第二个发送器将它保持为逻辑 0（将 SDA 下拉为低电平），则结果是 SDA 线将为低电平。那么，第一个发送器会发现线路电平与期望电平不同，并断定有另一个发送器正在进行通信。

发现电平不同的第一个发送器将是仲裁失败的发送器，必须停止驱动 SDA 线。如果该发送器同时也是主器件，则它还必须停止驱动 SCL 线。然后，它可以在尝试重新启动数据发送之前监视线路上是否出现停止条件。与此同时，另一个未发现期望电平与 SDA 线实际电平不同的器件将继续原来的数据发送。它可以无需进行任何复杂处理，因为到目前为止，发送情况与所期望的完全相同，没有其他发送器对报文产生干扰。

当主器件对多个从器件进行寻址时，也会对从发送模式进行仲裁，但这种情况较少见。

22.2.2.7. 启动条件

I^2C 规范将启动条件定义为在 SCL 线为高电平时，SDA 从高电平变为低电平状态。启动条件总是由主器件产生，指示总线从空闲状态变为有效状态。 图 22-12 给出了启动条件和停止条件的波形图。

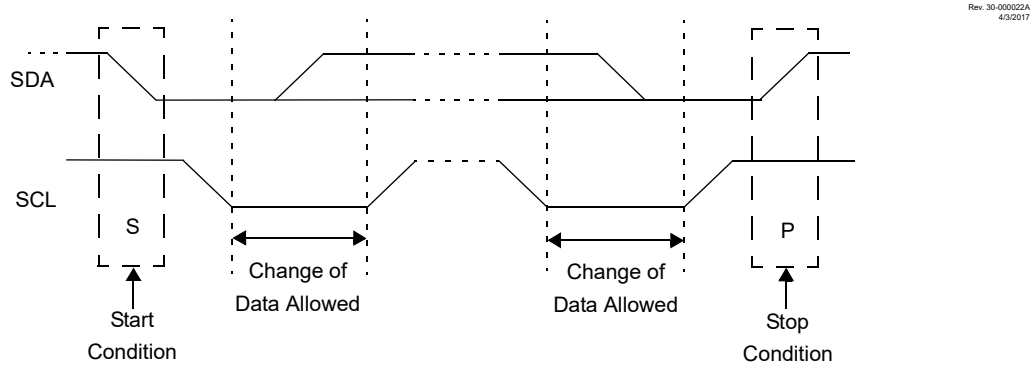
如果模块在将 SDA 线置为低电平之前采样到 SDA 线为低电平，则会在产生启动条件时发生总线冲突。这一点不符合 I^2C 规范，该规范规定启动时不能发生总线冲突。

22.2.2.8. 停止条件

停止条件定义为在 SCL 线为高电平时，SDA 线从低电平变为高电平状态。

重要：在停止条件有效之前必须至少出现一个 SCL 低电平时间，因此，如果 SDA 线先变为低电平，然后又变为高电平，而 SCL 线始终保持高电平，则只能检测到启动条件。

图 22-12. I^2C 启动和停止条件



22.2.2.9. 启动/停止条件中断屏蔽

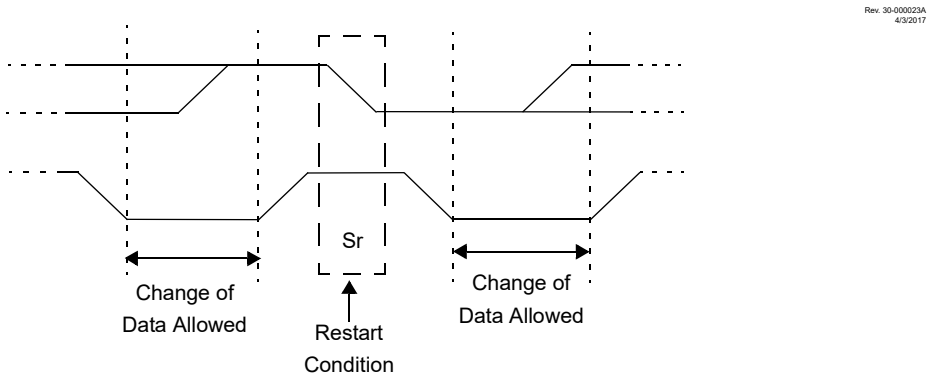
启动条件中断允许（SCIE）位和停止条件中断允许（PCIE）位可以用于允许在通常不支持中断功能的从模式下产生中断。对于已允许启动和停止检测中断的从模式，这两个位没有任何作用。

22.2.2.10. 重复启动条件

重复启动条件在每次停止条件有效的时候有效。如果主器件希望在终止当前传输之后保持对总线的控制，主器件可以发出重复启动条件。重复启动对从器件产生的影响与启动条件相同，即复位所有从器件逻辑并使之准备接收一个地址。主器件可以寻址同一个或另一个从器件。 图 22-13 给出了重复启动条件的波形图。

在 10 位寻址从模式下，要从寻址到的从器件中移出数据，主器件需要产生重复启动条件。从器件完全寻址（高地址字节和低地址字节均匹配）之后，主器件可以发出重复启动条件和 R/$\overline{W}$ 位置 1 的高地址字节。从器件逻辑会保持时钟，并准备送出数据。

图 22-13. I^2C 重复启动条件



22.2.2.11. 应答序列

在 I^2C 中，所有传输字节的第 9 个 SCL 脉冲都专门用作应答序列（$\overline{ACK}$）。它使接收器件可以通过将 SDA 线拉为低电平来响应发送器。发送器在该时间内必须释放对线路的控制，以移入响应信号。应答（$\overline{ACK}$）信号是低电平有效信号，它会将 SDA 线拉为低电平，用于指示发送器器件已接收到发送数据并已准备好接收更多数据。

$\overline{ACK}$ 的结果位于应答状态（ACKSTAT）位中。

当地址保持使能（AHEN）和数据保持使能（DHEN）位置 1 时，从软件允许用户选择要回送到发送器的 $\overline{ACK}$ 值。可以通过置 1/清零应答数据（ACKDT）位来决定响应。

在大多数情况下，从器件硬件会产生 $\overline{ACK}$ 响应。但如果在接收到字节时，BF 位或接收溢出指示符（SSPOV）位置 1，则从器件不会发送 $\overline{ACK}$。

对模块进行寻址时，在总线上的第 8 个 SCL 下降沿之后，应答时间状态（ACKTIM）位会置 1。ACKTIM 位指示有效总线的应答时间。ACKTIM 位仅在 AHEN 位或 DHEN 位使能时有效。

22.2.3. I^2C 从模式操作

MSSP 从模式可以在 4 种模式之一下工作，这些模式通过 MSSP 模式选择（SSPM）位进行选择。这些模式可以分为 7 位和 10 位寻址模式。10 位寻址模式的工作方式与 7 位寻址模式相同，只是在处理较大地址时需要一些额外的开销。

带启动条件和停止条件中断的模式的工作方式与其他模式相同，只是在检测到启动、重复启动或停止条件时，另外会将 SSPxIF 置 1。

22.2.3.1. 从模式地址

SSPxADD 寄存器包含从模式地址。在启动或重复启动条件之后接收到的第一个字节将与该寄存器中的存储值进行比较。如果字节匹配，则值会被装入 SSPxBUF 寄存器，并产生中断。如果值不匹配，则模块会进入空闲状态，并且不会向软件指示是否发生了什么情况。

SSPxMSK 寄存器会影响地址匹配过程。更多信息，请参见 SSP 掩码寄存器。

22.2.3.1.1. I²C 从模式 7 位寻址模式

在 7 位寻址模式下，当确定是否发生地址匹配时忽略已接收数据字节的 LSb。

22.2.3.1.2. I²C 从模式 10 位寻址模式

在 10 位寻址模式下，接收到的第一个字节将与二进制值“1 1 1 1 0 A9 A8 0”进行比较。A9 和 A8 是 10 位地址的高 2 位，分别存储在 SSPxADD 寄存器的 bit 2 和 bit 1 中。

在应答高字节之后，更新地址（UA）位会置 1，SCL 会保持低电平，直到用户使用低地址更新 SSPxADD 为止。在低地址字节送入之后，全部 8 位将与 SSPxADD 中的低地址值进行比较。即使地址不匹配，SSPxIF 和 UA 也会置 1，SCL 会保持低电平，直到 SSPxADD 发生更新可再次接收高字节为止。当 SSPxADD 发生更新时，UA 位会被清零。这可以确保模块准备好在下一次通信时接收高地址字节。

在所有 10 位寻址通信开始时，都需要以写请求方式进行高地址和低地址匹配。在寻址到从器件后，通过发出重复启动条件并随着时钟移入 R/$\overline{W}$ 位置 1 的高地址。然后，从硬件将会应答读请求，并准备好随着时钟移出数据。这只有在从器件接收到完全匹配的高地址和低地址字节之后才有效。

22.2.3.2. 时钟延长

当总线上的某个器件将 SCL 线保持为低电平而有效暂停通信时，就发生了时钟延长现象。从器件可以延长时钟，以便可以有更多时间来处理数据或准备响应主器件。时钟延长时不关心主器件的工作，因为任何时候只需总线上主器件处于活动状态但是不传输数据就可以被认为是时钟延长。由从器件进行的任何时钟延长对于主器件软件都是不可见的，都由产生 SCL 的硬件进行处理。

CKP 位用于通过软件控制延长。每当 CKP 位清零时，模块就会等待 SCL 线变为低电平，然后保持低电平状态不变。将 CKP 置 1 将会释放 SCL，允许继续进行通信。

22.2.3.2.1. 正常的时钟延长

在 $\overline{ACK}$ 之后，如果 R/$\overline{W}$ 位置 1（读请求），则从器件硬件会清零 CKP。这使得从器件有时间使用传送到主器件的数据更新 SSPxBUF。如果延长使能（SEN）位置 1，则在 $\overline{ACK}$ 序列之后，从器件硬件将始终延长时钟。在从器件就绪之后，软件会将 CKP 置 1，并继续进行通信。

22.2.3.2.2. 10 位寻址模式

在 10 位寻址模式下，当 UA 位置 1 时，时钟总是会被延长。这是无需清零 CKP 就会延长 SCL 的惟一情形。在写入 SSPxADD 之后，SCL 会立即被释放。

22.2.3.2.3. 不应答字节

当 AHEN 位置 1 时，在所接收匹配地址字节的第 8 个 SCL 下降沿之后，硬件会将 CKP 清零。当 DHEN 位置 1 时，在所接收数据的第 8 个 SCL 下降沿之后，CKP 会被清零。

在 SCL 信号的第 8 个下降沿之后延长时钟，可以使从器件能够查看接收到的地址或数据，并决定它应答（$\overline{ACK}$）还是不应答（NACK）接收的地址或数据。

22.2.3.3. 时钟同步和 CKP 位

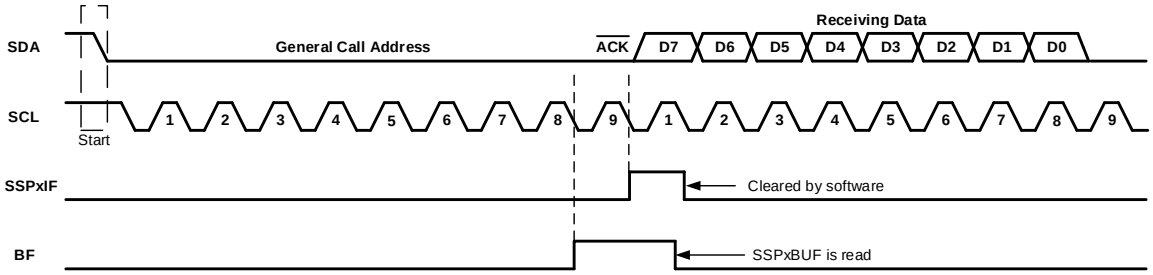
每当 CKP 位清零时，模块就会等待 SCL 线变为低电平，然后保持低电平状态不变。但是，只有当已经采样到 SCL 输出为低电平时，清零 CKP 位才会将 SCL 输出置为低电平。因此，CKP 位不会将 SCL 线拉为低电平，除非外部 I²C 主器件已将 SCL 线拉为低电平。SCL 输出将保持低电平，直到 CKP 位置 1 且 I²C 总线上的所有其他器件已释放 SCL 为止。

22.2.3.4. 广播呼叫地址支持

在 I²C 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。当使用此地址时，理论上所有器件都必须发送一个 $\overline{ACK}$ 作为响应。

在 I²C 协议中，广播呼叫地址是保留地址，定义为地址 0x00。当广播呼叫使能（GCEN）位置 1 时，无论 SSPxADD 中存储的值如何，在接收到该地址后，从模块都会自动发送 ACK。在从器件随时钟移入 R/W 位清零的全零地址之后，将会产生中断，从器件软件可以读取 SSPxBUF 并进行响应。 图 22-14 显示了广播呼叫接收序列。

图 22-14. 从模式广播呼叫地址序列



在 10 位地址模式下，UA 位不会在接收到广播呼叫地址时置 1。从器件会准备接收作为数据的第二个字节，这与在 7 位模式下相同。

如果 AHEN 位置 1，则与接收到任意其他地址时相同，从器件硬件会在 SCL 的第 8 个下降沿之后延长时钟。从器件必须将其应答序列使能（ACKEN）位置 1 并释放时钟。

22.2.3.5. SSP 掩码寄存器

MSSP 掩码（SSPxMSK）寄存器在 I²C 从模式下可用，用作地址比较操作期间 SSPSR 寄存器中保存的值的掩码。SSPxMSK 寄存器中的零（0）位可使接收地址中相应位变为“无关位”。

任何复位条件都可将此寄存器复位为全 1，因此在写入掩码值之前对标准 MSSP 操作没有影响。

SSPxMSK 在以下模式下有效：

- 7 位地址模式：用于比较地址的 A[7:1]
- 10 位地址模式：仅用于比较地址的 A[7:0]。在接收地址的第一个（高）字节期间，MSSP 掩码没有影响。

22.2.3.6. 从器件接收

当接收到的匹配地址字节的 R/W 位清零时，R/W 位也清零。接收到的地址被装入 SSPxBUF 寄存器并产生应答信号。

当接收到的地址存在溢出条件时，将会发送无应答信号（NACK），并且接收溢出指示符（SSPOV）位将置 1。缓冲区改写使能（BOEN）位可修改此操作。

每个传输的数据字节都会产生 MSSP 中断。SSPxIF 标志位必须用软件清零。

当 SEN 位置 1 时，每接收到一个字节 SCL 都会保持低电平（时钟延长）状态。必须通过将 CKP 位置 1 来释放时钟，10 位模式下的特殊情况除外。有关详细信息，请参见 10 位寻址模式。

22.2.3.6.1. 7 位寻址接收

本节介绍在 7 位寻址模式下，配置为 I²C 从器件的 MSSP 模块的标准事件序列。 图 22-15 和图 22-16 用直观的方式对此作了说明。

以下列出了实现 I²C 通信时通常必须完成的步骤。

1. 检测到启动条件。

2. 启动（S）位置 1；如果启动条件中断允许（SCIE）位置 1，则 SSPxIF 也置 1。
3. 从器件接收到 R/$\overline{W}$ 位清零的匹配地址。
4. 从器件将 SDA 线拉为低电平，向主器件发送 $\overline{ACK}$，并将 SSPxIF 位置 1。
5. 用软件清零 SSPxIF 位。
6. 软件从 SSPxBUF 读取接收的地址，使 BF 标志清零。
7. 如果 SEN = 1，从器件软件会通过将 CKP 位置 1 来释放 SCL 线。
8. 主器件随着时钟移出数据字节。
9. 从器件将 SDA 线驱动为低电平，向主器件发送 $\overline{ACK}$，并将 SSPxIF 位置 1。
10. 用软件清零 SSPxIF 位。
11. 软件从 SSPxBUF 读取接收的字节，使 BF 位清零。
12. 对于从主器件接收到的所有字节重复步骤 8-12。
13. 主器件发送停止条件，将停止（P）位置 1，总线变为空闲状态。

图 22-15. I²C 从模式接收时序（SEN = 0，AHEN = 0，DHEN = 0，7 位地址）

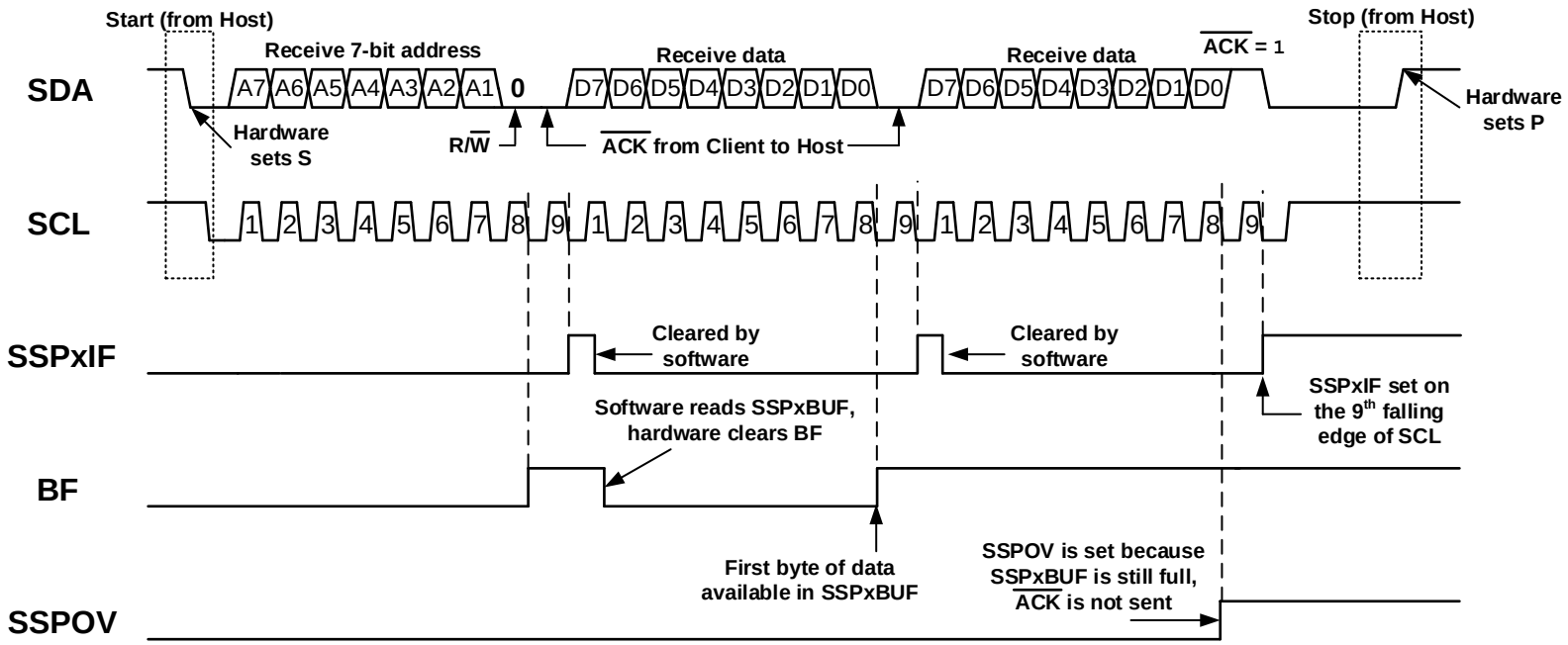
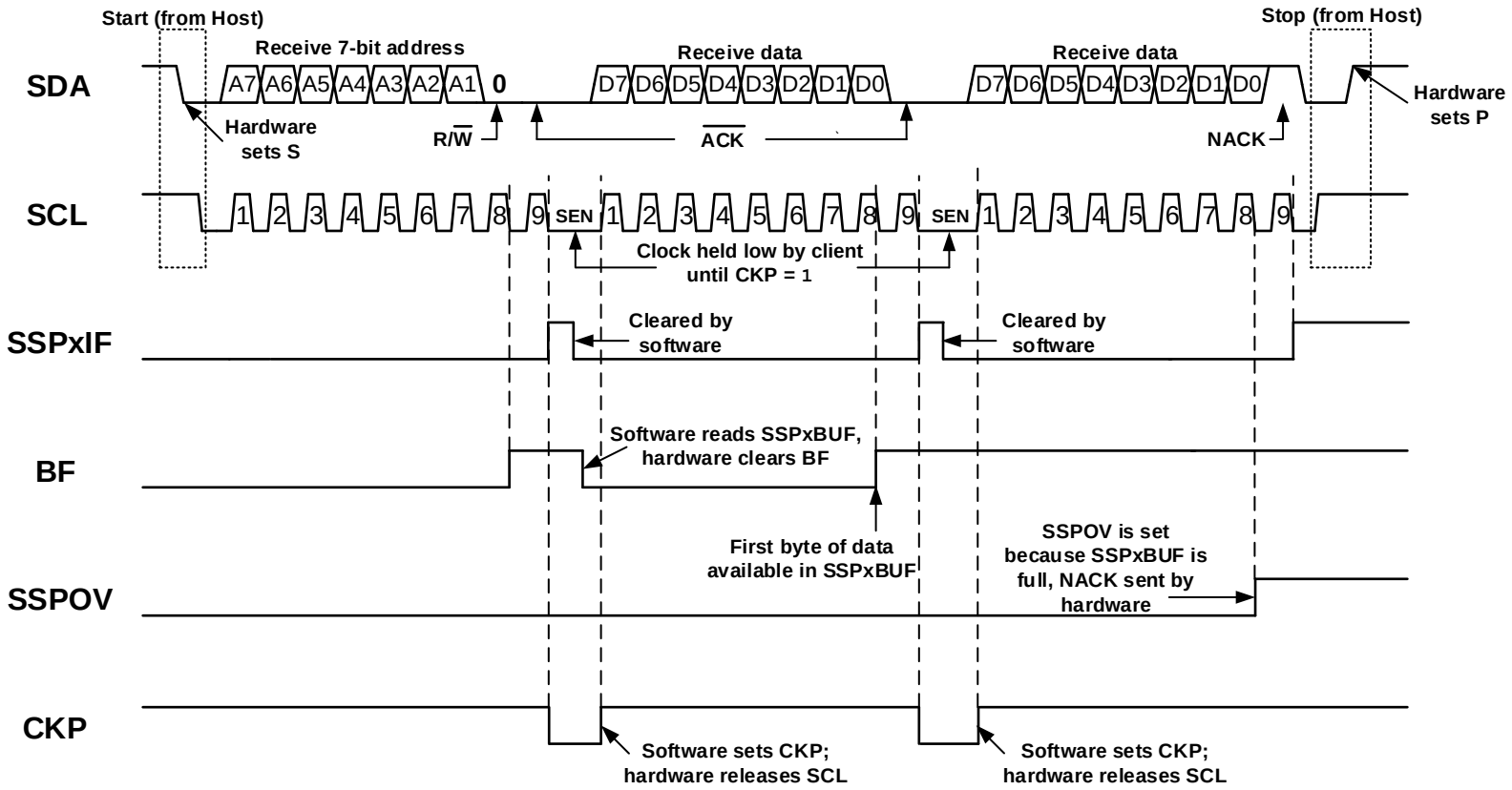


图 22-16. I²C 从模式接收时序（SEN = 1，AHEN = 0，DHEN = 0，7 位地址）



22.2.3.6.2. AHEN 和 DHEN 位置 1 的 7 位接收

在 AHEN 和 DHEN 置 1 时，从器件接收的工作方式与不使用这些选项时的工作方式相同，只是在 SCL 的第 8 个下降沿之后添加了额外的中断和时钟延长。这些额外中断允许从器件软件决定是否应答（$\overline{ACK}$）接收的地址或数据字节，而不是由硬件决定。该功能增加了对 PMBus™的支持，先前版本的该模块不支持这一功能。

下面列出了要对 I^2C 通信使用这些选项时，从器件软件需要执行的步骤。 图 22-17 显示了同时使用地址和数据保持功能的模块。 图 22-18 包含了 SEN 位置 1 时的操作。

1. 启动（S）位置 1；如果 SCIE 置 1，则 SSPxIF 也置 1。
2. R/$\overline{W}$ 位清零的匹配地址随时钟移入。在 SCL 的第 8 个下降沿之后，SSPxIF 置 1，CKP 清零。
3. 用软件清零 SSPxIF。
4. 从器件可以查看 ACKTIM 位，以确定 SSPxIF 是在 $\overline{ACK}$ 之前还是之后置 1。
5. 从器件从 SSPxBUF 中读取地址值，使 BF 标志清零。
6. 从器件通过清零 ACKDT 向主器件发送 $\overline{ACK}$。
7. 从器件通过将 CKP 置 1 来释放时钟。
8. SSPxIF 会在 $\overline{ACK}$ 之后置 1，不会在 NACK 之后置 1。
9. 如果 SEN = 1，从器件硬件会在 $\overline{ACK}$ 之后延长时钟。
10. 从器件清零 SSPxIF。

> **重要：**即使不进行时钟延长，且 BF 已清零，SSPxIF 仍然会在 SCL 的第 9 个下降沿之后置 1。只有向主器件发送了 NACK 信号后，SSPxIF 才不会置 1。

11. 对于接收到的数据字节，在 SCL 的第 8 个下降沿之后，SSPxIF 置 1，CKP 清零。
12. 从器件通过查看 ACKTIM 位来确定中断源。
13. 从器件从 SSPxBUF 中读取接收的数据，使 BF 位清零。
14. 对于每个接收的数据字节重复步骤 7-14。
15. 从器件发送 NACK，或主器件发送停止条件可结束通信。如果已发送停止条件并且停止条件中断允许（PCIE）位清零，则从器件只能通过查询停止（P）位来了解通信是否结束。

图 22-17. I²C 从模式接收时序（SEN = 0，AHEN = 1，DHEN = 1，7 位地址）

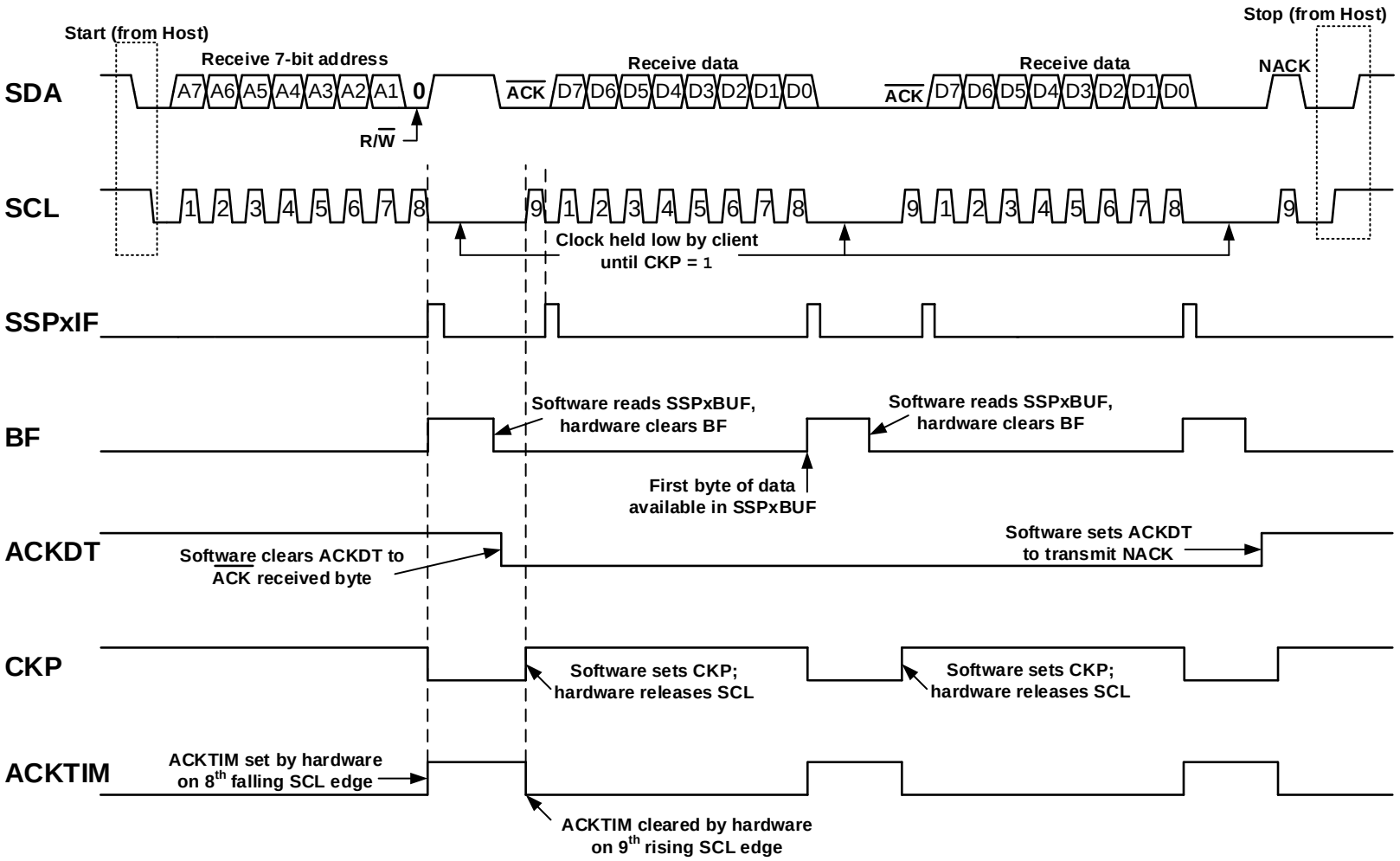
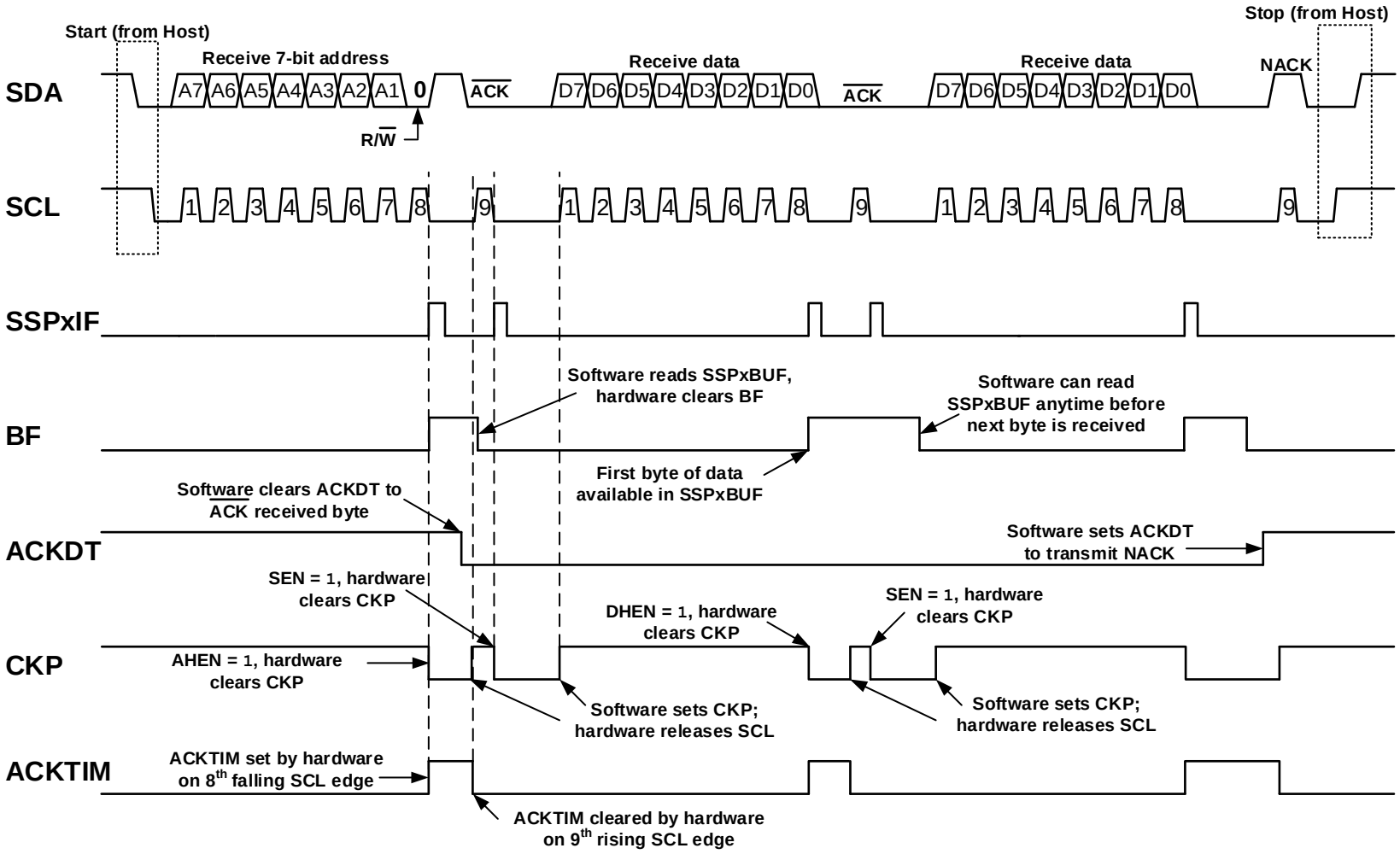


图 22-18. I²C 从模式接收时序（SEN = 1，AHEN = 1，DHEN = 1，7 位地址）



22.2.3.6.3. 从模式 10 位地址接收

本节介绍在 10 位寻址模式下，配置为 I²C 从器件的 MSSP 模块的标准事件序列。 图 22-19 给出了使能时钟延长时 10 位寻址模式下从接收器的标准波形图。

下面列出了实现 I²C 通信时从器件软件必须完成的步骤。

1. 总线启动时为空闲模式。
2. 主器件发送启动条件；S 位置 1；如果 SCIE 置 1，则 SSPxIF 也置 1。
3. 主器件发送 R/$\overline{W}$ 位清零的匹配高地址；UA 位置 1。
4. 从器件发送 $\overline{ACK}$，SSPxIF 置 1。
5. 用软件清零 SSPxIF 位。
6. 软件从 SSPxBUF 读取接收的地址，使 BF 标志清零。
7. 从器件将低地址装入 SSPxADD，释放 SCL。
8. 主器件向从器件发送匹配的低地址字节；UA 位置 1。

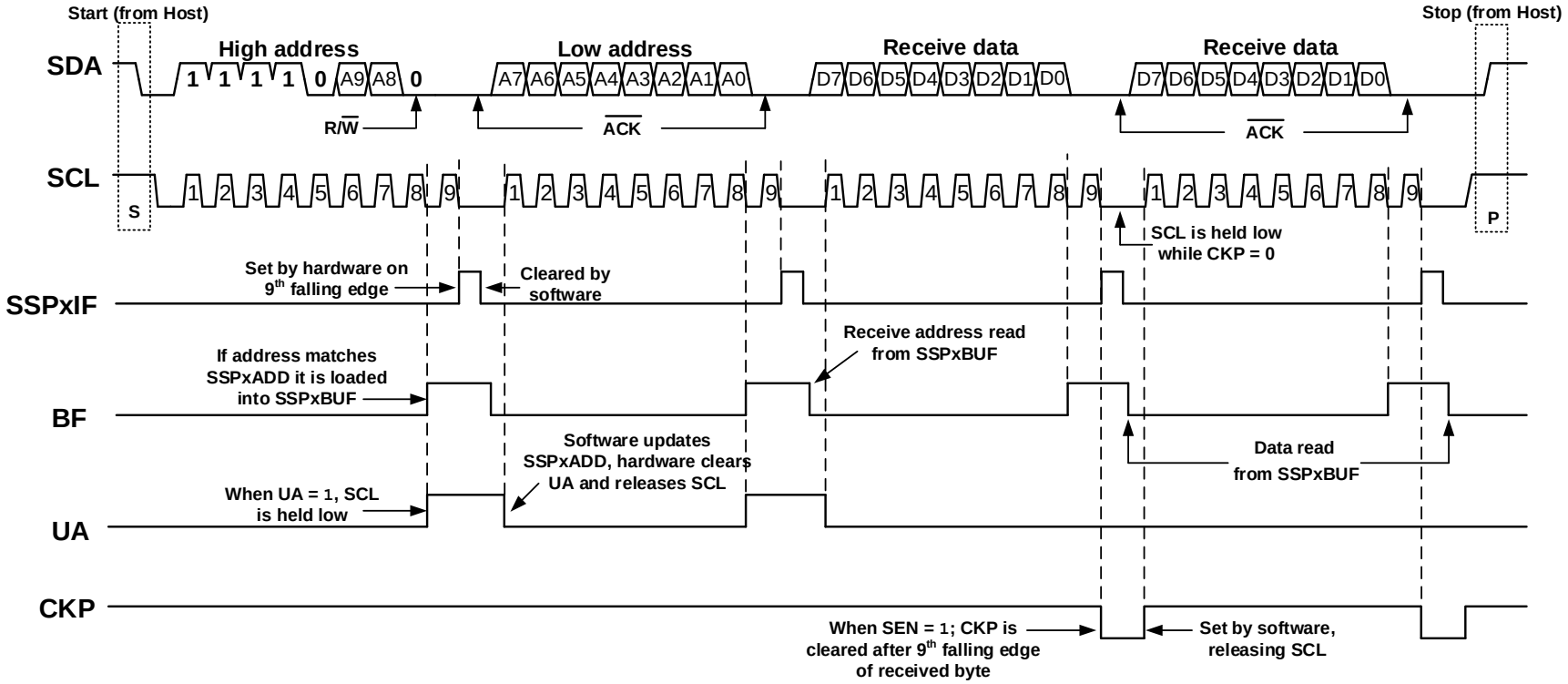
> **重要：**只有在 $\overline{ACK}$ 序列之后，才允许更新 SSPxADD 寄存器。

9. 从器件发送 $\overline{ACK}$，SSPxIF 置 1。

> **重要：**如果低地址不匹配，SSPxIF 和 UA 仍然置 1，以便从器件软件可以将 SSPxADD 设置回高地址。BF 不置 1，因为没有发生匹配。CKP 不受影响。

10. 从器件清零 SSPxIF。
11. 从器件从 SSPxBUF 中读取接收的匹配地址，使 BF 清零。
12. 从器件将高地址装入 SSPxADD。
13. 主器件随着时钟将数据字节移入从器件，并在第 9 个 SCL 脉冲随着时钟将 $\overline{ACK}$ 移出从器件；SSPxIF 置 1。
14. 如果 SEN 位置 1，CKP 会被硬件清零，时钟会被延长。
15. 从器件清零 SSPxIF。
16. 从器件从 SSPxBUF 中读取接收的字节，使 BF 清零。
17. 如果 SEN 置 1，从器件软件会将 CKP 置 1 以释放 SCL。
18. 对于每个接收的字节重复步骤 13-17。
19. 主器件发送停止条件以结束发送。

图 22-19. I²C 从模式接收时序（SEN = 1，AHEN = 0，DHEN = 0，10 位地址）



22.2.3.6.4. 带地址或数据保持的 10 位寻址

在 AHEN 或 DHEN 置 1 时，使用 10 位寻址的接收方式与 7 位模式相同。惟一的区别是需要使用 UA 位来更新 SSPxADD 寄存器。所有功能（特别是在 CKP 位清零，SCL 线保持低电平时）都是相同的。 图 22-20 可以用作 AHEN 置 1 时 10 位寻址模式下从器件的参考图示。

图 22-21 给出了 10 位寻址模式下从发送器的标准波形图。

图 22-20. I²C 从模式接收时序（SEN = 0，AHEN = 1，DHEN = 0，10 位地址）

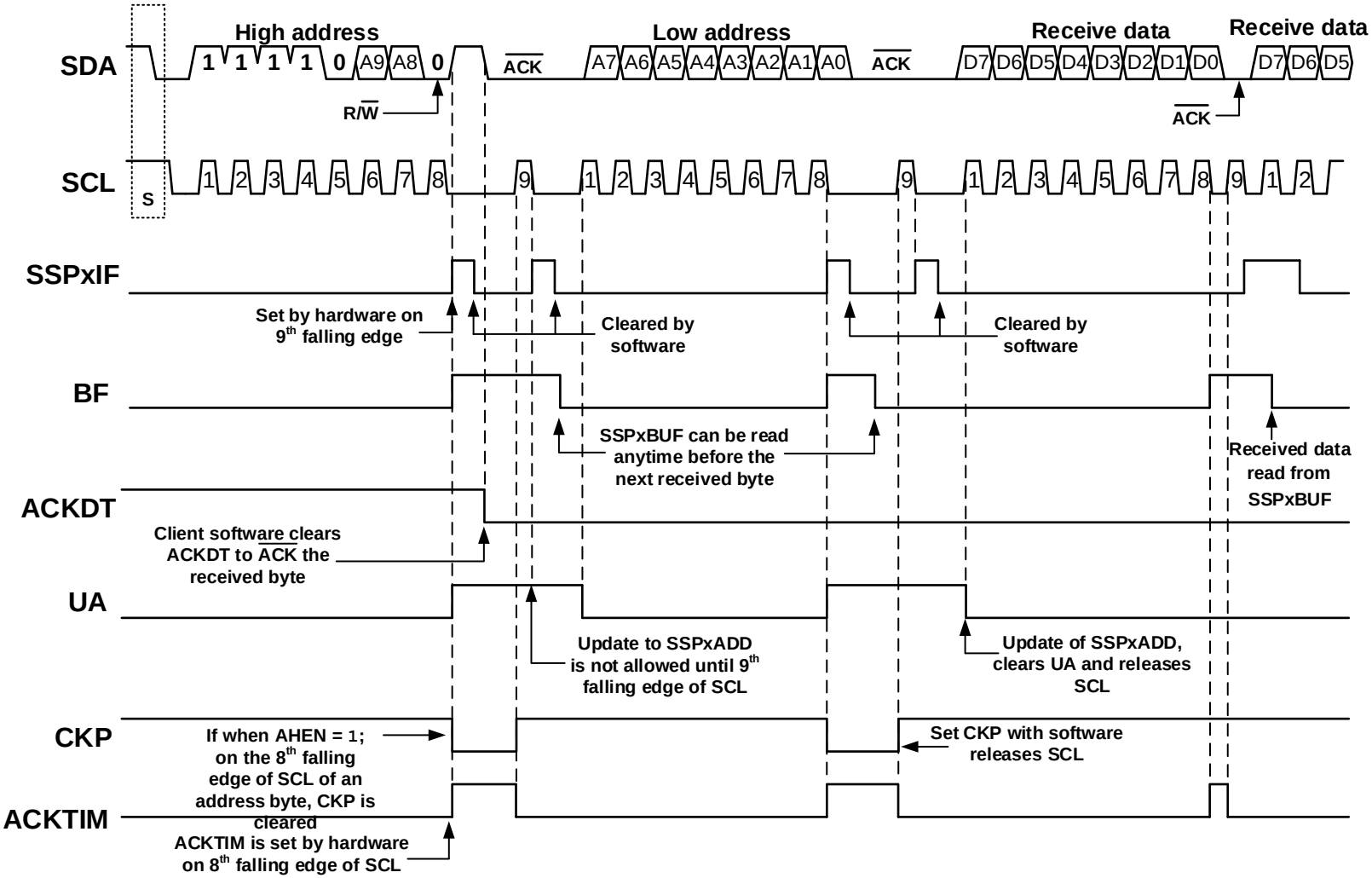
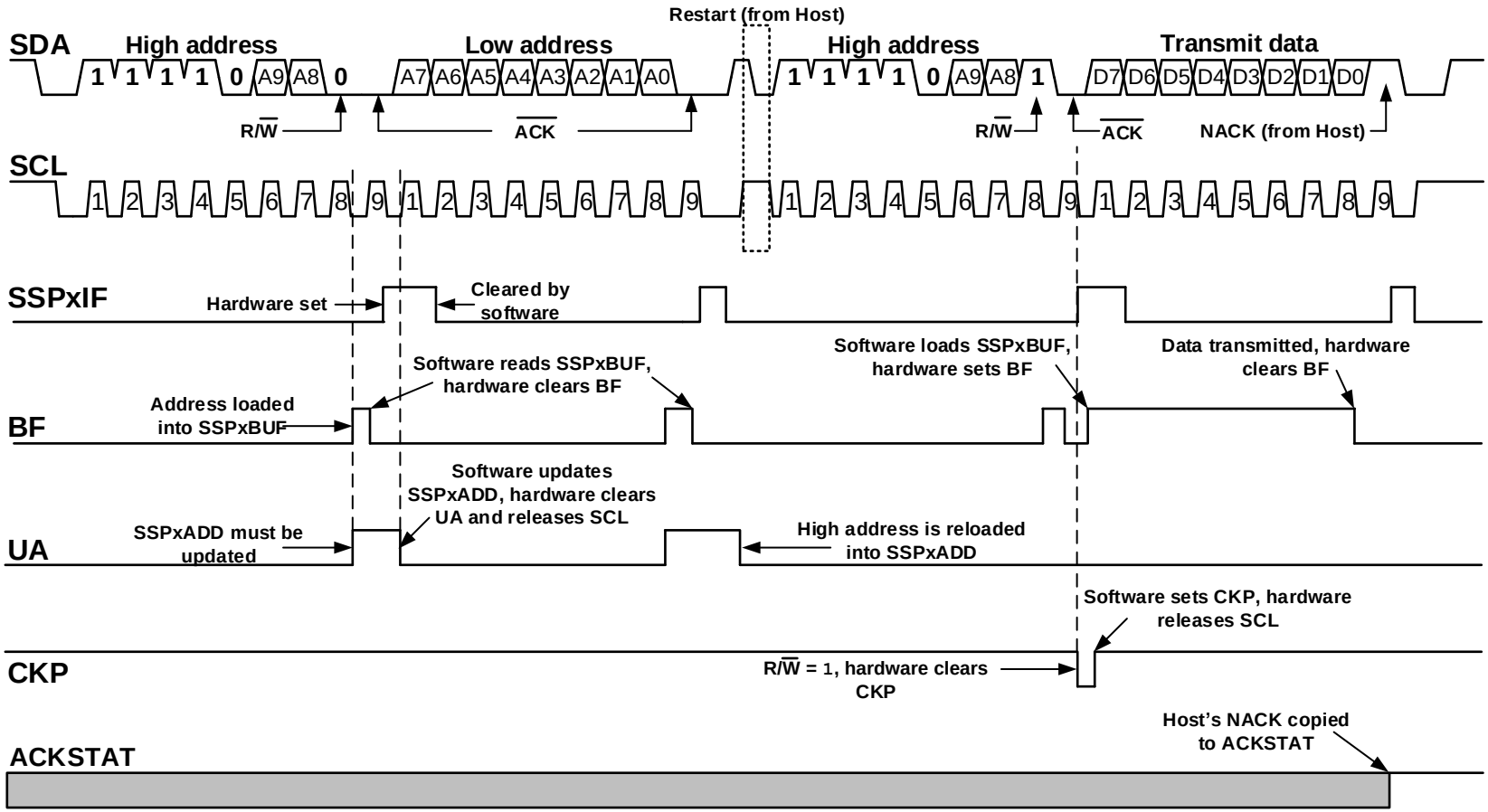


图 22-21. I²C 从模式发送时序（SEN = 0，AHEN = 0，DHEN = 0，10 位地址）



22.2.3.7. 从器件发送

当传入的地址字节的 R/$\overline{W}$ 位置 1 并发生地址匹配时，R/$\overline{W}$ 位置 1。接收到的地址将装入 SSPxBUF 寄存器，且在第 9 个位由从器件发送一个 $\overline{ACK}$ 脉冲。

在 $\overline{ACK}$ 之后，从器件硬件会清零 CKP 位，并且 SCL 引脚保持低电平（有关详细信息，请参见时钟延长）。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。

发送数据必须装入 SSPxBUF 寄存器，同时也装入 SSPSR 寄存器。然后，通过将 CKP 位置 1 来释放 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。

来自主器件接收器的 $\overline{ACK}$ 脉冲将在第 9 个 SCL 输入脉冲的上升沿锁存。该 $\overline{ACK}$ 值会被复制到 ACKSTAT 位中。如果 ACKSTAT 置 1（NACK），则表示数据传输已完成。这种情况下，当从器件锁存了 NACK 值时，从器件进入空闲模式，等待出现下一个启动条件。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 SSPxBUF 寄存器。同样，必须通过将 CKP 位置 1 来释放 SCL 引脚。

每个传输的数据字节都会产生 MSSP 中断。SSPxIF 位必须用软件清零，SSPxSTAT 寄存器用于确定字节的状态。SSPxIF 位在第 9 个时钟脉冲的下降沿被置 1。

22.2.3.7.1. 从模式总线冲突

从器件接收到读请求，开始在 SDA 线上移出数据。如果检测到总线冲突并且从模式总线冲突检测使能（SBCDE）位置 1，则 PIRx 寄存器的总线冲突中断标志（BCLxIF）位会置 1。在检测到总线冲突时，从器件会变为空闲状态，等待再次被寻址。用户软件可以通过使用 BCLxIF 位来处理从器件总线冲突。

22.2.3.7.2. 7 位发送

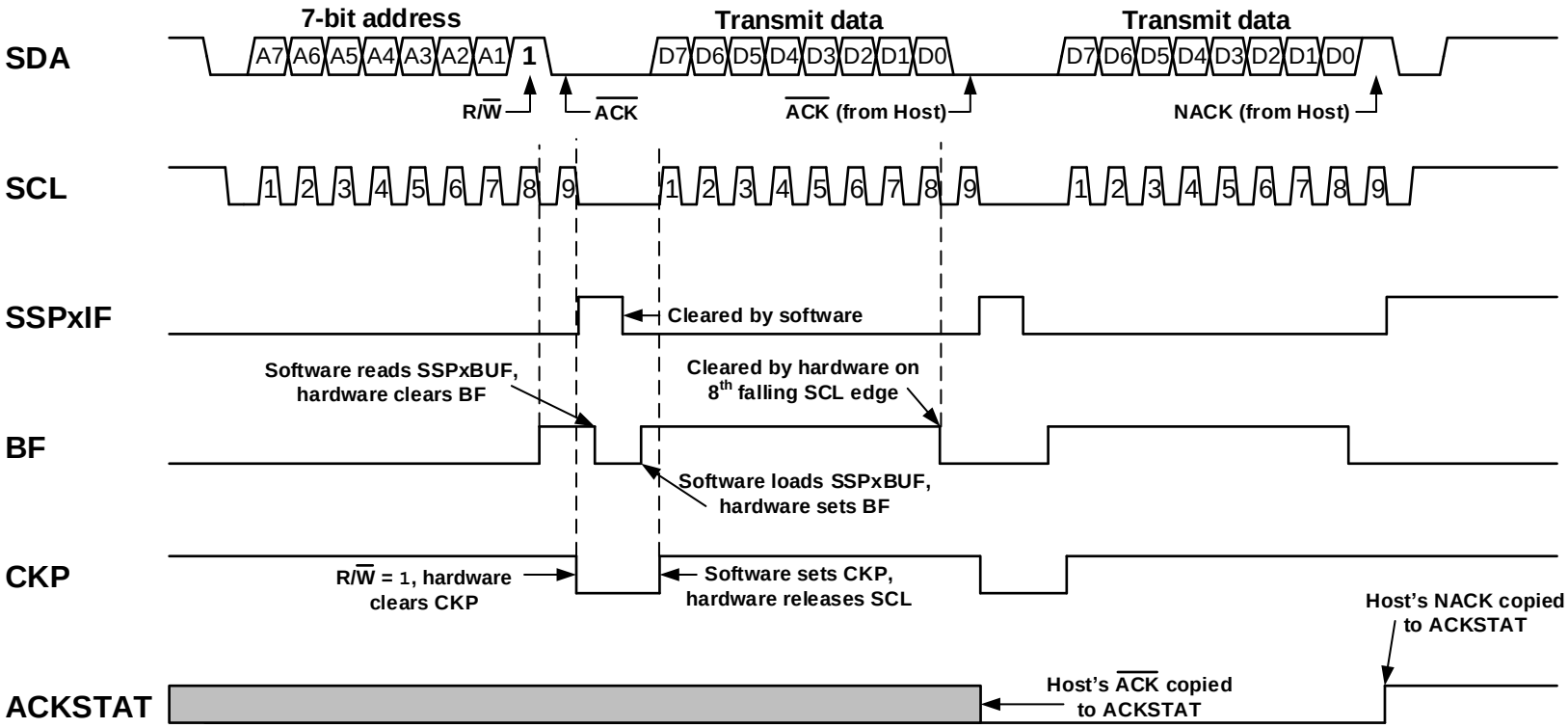
主器件可以向从器件发送读请求，然后随时钟从从器件中移出数据。下面列出了在实现标准数据发送时，从软件需要执行的操作。 图 22-22 可用作该列表的参考。

1. 主器件发送启动条件。
2. 启动（S）位置 1；如果 SCIE 置 1，则 SSPxIF 也置 1。
3. 从器件接收到 R/$\overline{W}$ 位置 1 的匹配地址，将 SSPxIF 位置 1。
4. 从器件硬件产生 $\overline{ACK}$ 并将 SSPxIF 置 1。
5. 用软件清零 SSPxIF 位。
6. 软件从 SSPxBUF 中读取接收的地址，使 BF 清零。
7. R/$\overline{W}$ 置 1，所以 CKP 在 $\overline{ACK}$ 之后自动清零。
8. 从器件软件将发送数据装入 SSPxBUF。
9. 用软件将 CKP 位置 1，释放 SCL，从而允许主器件将数据随着时钟移出从器件。
10. 来自主器件的 $\overline{ACK}$ 响应装入 ACKSTAT 位之后，SSPxIF 置 1。
11. SSPxIF 位清零。
12. 从器件软件通过检查 ACKSTAT 位来确定主器件是否要移出更多数据。

> **重要：**
> 1. 如果主器件发送了 $\overline{ACK}$ 信号，时钟将被延长。
> 2. ACKSTAT 是惟一一个在 SCL 时钟上升沿（第 9 个）而不是下降沿发生更新的位。

13. 对于每个发送的字节重复步骤 9-13。
14. 如果主器件发送非 $\overline{ACK}$ 信号，则时钟不被延长，但 SSPxIF 标志位仍置 1。
15. 主器件发送重复启动条件或停止条件。

图 22-22. I²C 从模式发送时序（AHEN = 0，7 位地址）



22.2.3.7.3. 使能了地址保持的 7 位发送

将 AHEN 位置 1 时，器件会在所接收匹配地址的第 8 个下降沿之后延长时钟和产生中断。随着时钟移入匹配地址后，CKP 位清零且 SSPxIF 中断位置 1。

图 22-23 给出了在使能 AHEN 时 7 位地址从发送的标准波形图。

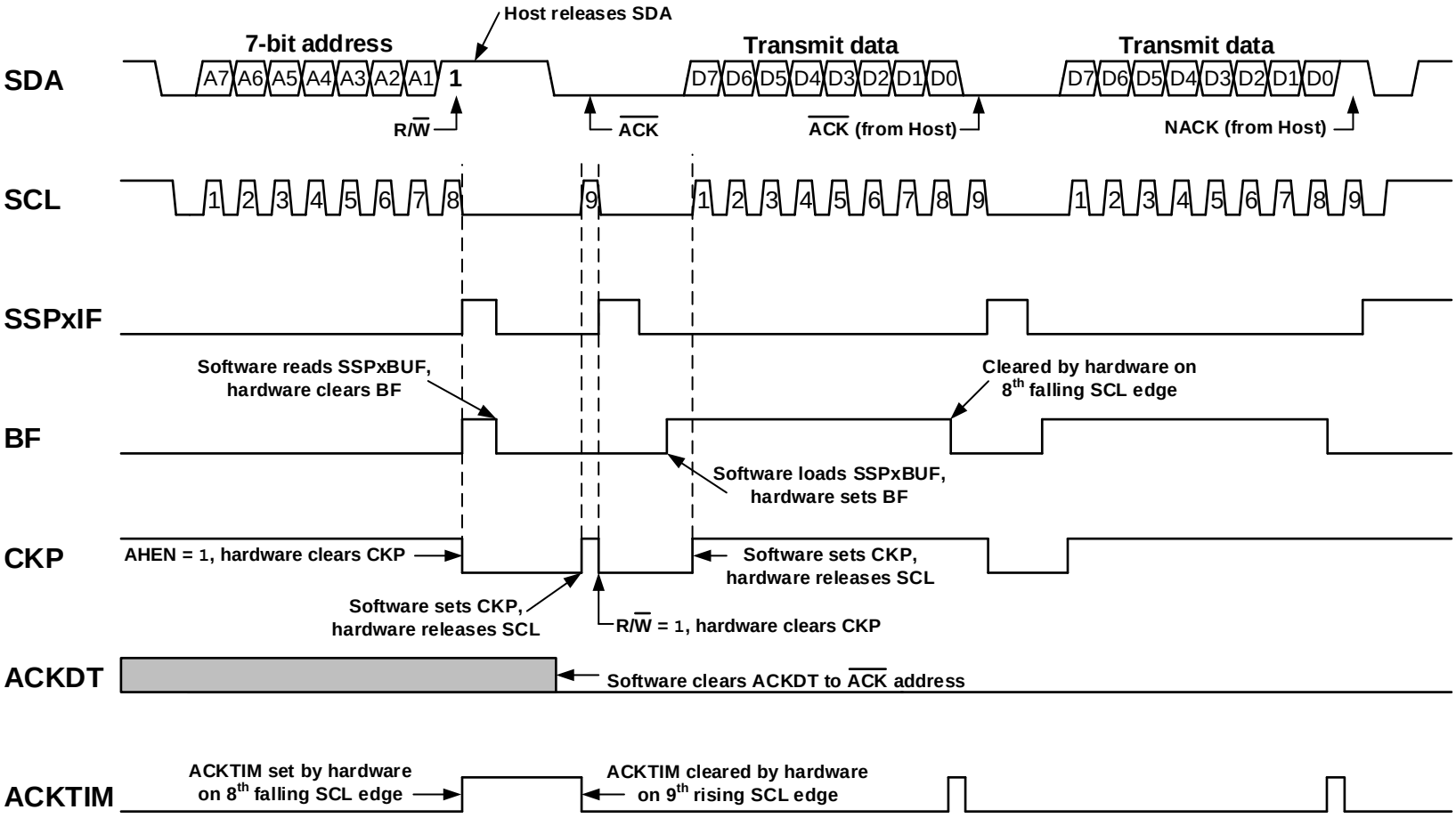
1. 总线启动时为空闲模式。
2. 主器件发送启动条件；S 位置 1；如果 SCIE 置 1，则 SSPxIF 也置 1。
3. 主器件发送 R/$\overline{W}$ 位置 1 的匹配地址。在 SCL 线的第 8 个下降沿之后，CKP 位清零，并产生 SSPxIF 中断。
4. 从器件软件清零 SSPxIF。
5. 从器件软件读取 ACKTIM、R/$\overline{W}$ 和 D/$\overline{A}$ 位来确定中断源。
6. 从器件从 SSPxBUF 寄存器中读取地址值，使 BF 位清零。
7. 从器件软件根据该信息确定它是产生 $\overline{ACK}$ 还是产生 NACK，并相应地设置 ACKDT 位。
8. 从器件软件将 CKP 位置 1，以释放 SCL。
9. 主器件随着时钟移入来自从器件的 $\overline{ACK}$ 值。
10. 如果 R/$\overline{W}$ 位置 1，则从器件硬件在 $\overline{ACK}$ 之后自动清零 CKP 位并将 SSPxIF 置 1。
11. 从器件软件清零 SSPxIF。
12. 从器件将要发送给主器件的值装入 SSPxBUF，使 BF 位置 1。

> **重要：**SSPxBUF 在接收到 $\overline{ACK}$ 之后才能装入数据。

13. 从器件软件将 CKP 位置 1 以释放时钟。
14. 主器件随着时钟从从器件中移出数据，并在第 9 个 SCL 脉冲发送 $\overline{ACK}$ 值。
15. 从器件硬件将 $\overline{ACK}$ 值复制到 ACKSTAT 位中。
16. 对于从从器件发送到主器件的每个字节重复步骤 10-15。
17. 如果主器件发送非 $\overline{ACK}$，从器件会释放总线，让主器件可以发送停止条件和结束通信。

> **重要：**主器件必须对于最后一个字节发送非 $\overline{ACK}$，以确保从器件释放 SCL 线来接收停止条件。

图 22-23. I²C 从模式发送时序（AHEN = 1，7 位地址）



22.2.4. I²C 主模式

通过配置相应的 SSPM 位，同时将 SSPEN 位置 1，可以使能主模式。在主模式下，必须将 SDA 和 SCL 引脚配置为输入。当需要将引脚驱动为低电平时，MSSP 外设硬件将改写输出驱动器的 TRIS 控制。

通过在检测到启动和停止条件时产生中断来支持主操作模式。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 或总线空闲时，可以获得 I²C 总线的控制权。

在固件控制的主模式下，用户代码基于启动和停止条件检测功能执行所有的 I²C 总线操作。在该模式下，启动和停止条件检测是惟一有效的电路。所有其他通信都通过用户软件直接操作 SDA 和 SCL 线来完成。

以下事件会使 MSSP 中断标志（SSPxIF）位置 1（如果允许 MSSP 中断，则产生中断）：

- 检测到启动条件
- 检测到停止条件
- 发送/接收到数据传送字节
- 发送/接收到应答
- 产生了重复启动条件

重要：

1. 当配置为 I²C 主模式时，MSSP 模块不允许事件排队。例如，不允许用户在发出启动条件后，在启动条件结束前立即写 SSPxBUF 寄存器以启动发送。在这种情况下，将不会写 SSPxBUF，写冲突检测（WCOL）位将被置 1，指示没有发生对 SSPxBUF 的写操作。
2. 当通过 SEN/PEN 控制位发送启动/停止条件时，主模式将暂停启动/停止检测。当硬件清零控制位时，SSPxIF 位将在启动/停止生成过程结束时置 1。

22.2.4.1. I²C 主模式操作

主器件负责产生所有的串行时钟脉冲、启动条件和停止条件。传输过程以停止条件或重复启动条件结束。因为重复启动条件也是下一次串行传输的开始，因此 I²C 总线不会被释放。

在主发送器模式下，串行数据通过 SDA 输出，而串行时钟由 SCL 输出。发送的第一个字节包括接收器件的从器件地址（7 位）和 R/$\overline{W}$ 位。在这种情况下，R/$\overline{W}$ 位将为逻辑 0。一次发送 8 位串行数据。每发送一个字节，都会接收到一个应答位。输出启动和停止条件以指示串行传输的开始和结束。

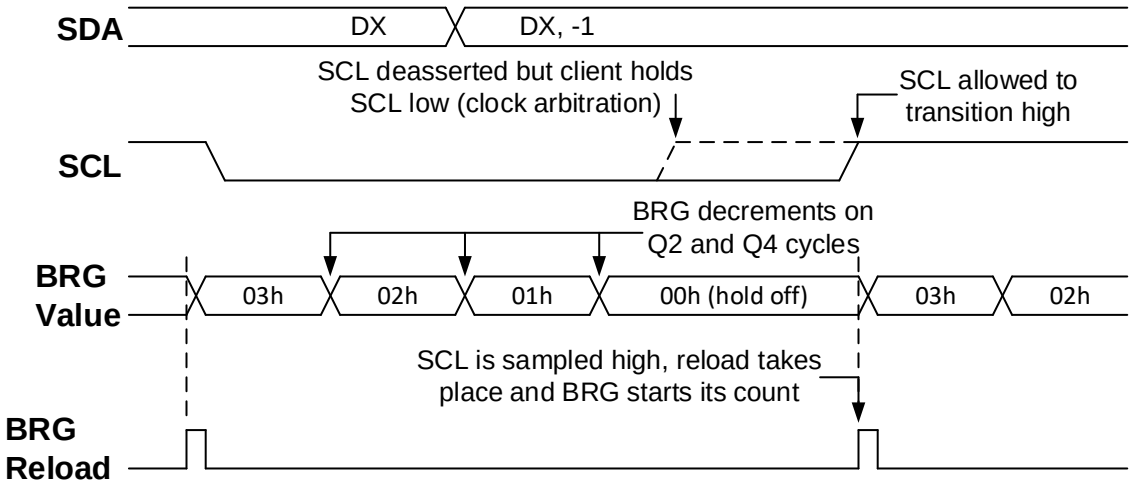
在主接收模式下，发送的第一个字节包括发送器件的从器件地址（7 位）和 R/$\overline{W}$ 位。在这种情况下，R/$\overline{W}$ 位将为逻辑 1。因此，第一个发送的字节是 7 位从器件地址以及随后用于指示接收位的 1。串行数据通过 SDA 接收，而串行时钟由 SCL 输出。一次接收 8 位串行数据。每接收到一个字节，都会发送一个应答序列。启动条件和停止条件指示发送的开始和结束。

波特率发生器用于设置从 SCL 输出的时钟频率。有关详细信息，请参见波特率发生器。

22.2.4.1.1. 时钟仲裁

如果在任何接收、发送或重复启动/停止条件期间，主器件释放了 SCL 引脚（允许 SCL 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（Baud Rate Generator，BRG）暂停计数，直到 SCL 引脚被实际采样到高电平为止。当采样到 SCL 引脚为高电平时，波特率发生器重新装入 SSPxADD 的内容并开始计数。这可以确保在外部器件将时钟保持低电平时，SCL 在至少一个 BRG 计满返回计数周期内总是保持高电平（如图 22-24 所示）。

图 22-24. 带有时钟仲裁的波特率发生器时序



22.2.4.1.2. WCOL 状态标志

如果在启动、重复启动、停止、接收或发送序列过程中用户写 SSPxBUF，则写冲突检测（WCOL）位被置 1，同时缓冲区内容不变（未发生写操作）。每当 WCOL 位置 1 时，它指示在模块不处于空闲状态时对 SSPxBUF 尝试了某个操作。

重要：由于不允许事件排队，在启动条件结束之前，不能写 SSPxCON2 的低 5 位。

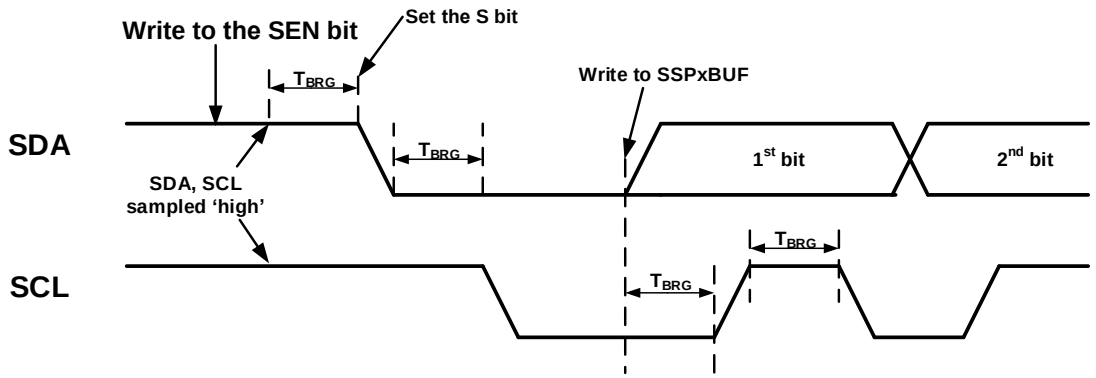
22.2.4.1.3. I²C 主模式启动条件时序

要发起启动条件（见图 22-25），用户应将启动条件使能（SEN）位置 1。当采样到 SDA 和 SCL 引脚均为高电平时，波特率发生器重新装入 SSPxADD 的内容并开始计数。如果波特率发生器超时（T_{BRG}）时，采样到 SCL 和 SDA 均为高电平，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，并使启动（S）位置 1。随后波特率发生器重新装入 SSPxADD 的内容并恢复计数。当波特率发生器超时（T_{BRG}）时，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。

重要：

1. 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 线在 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（BCLxIF）置 1，启动条件中止，I²C 模块复位到空闲状态。
2. Philips I²C 规范规定启动时不能发生总线冲突。

图 22-25. 第一个启动位时序



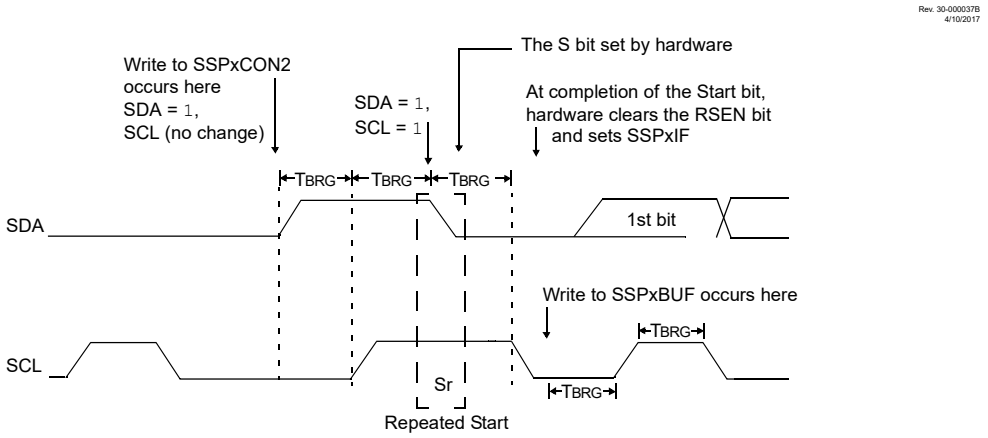
22.2.4.1.4. I²C 主模式重复启动条件时序

当重复启动条件使能（RSEN）位设定为高电平，并且主器件状态机空闲时，会产生重复启动条件（见图 22-26）。当 RSEN 位置 1 时，SCL 引脚被拉为低电平。当采样到 SCL 引脚为低电平时，波特率发生器会装入值并开始计数。在一个波特率发生器计数周期（T_{BRG}）内 SDA 引脚被释放（拉为高电平）。当波特率发生器超时时，如果采样到 SDA 为高电平，SCL 引脚将被置为无效（拉为高电平）。当采样到 SCL 为高电平时，波特率发生器被重载并开始计数。SDA 和 SCL 必须在一个 T_{BRG} 内保持高电平。随后模块硬件将 SDA 线拉为低电平（此时 SCL 保持高电平）并持续一个 T_{BRG}，然后再将 SCL 线拉为低电平。随后，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，S 位将被置 1。SSPxIF 位在波特率发生器超时之前不会被置 1。

重要：

1. 如果在任何其他事件正在进行时编程 RSEN，此操作将不起作用。
2. 在重复启动条件期间如果发生以下情况，将发生总线冲突：
   - SCL 从低电平变为高电平时，采样到 SDA 为低电平。
   - 在 SDA 被置为低电平之前，SCL 变为低电平。这指示另一个主器件正尝试发送数据 1。

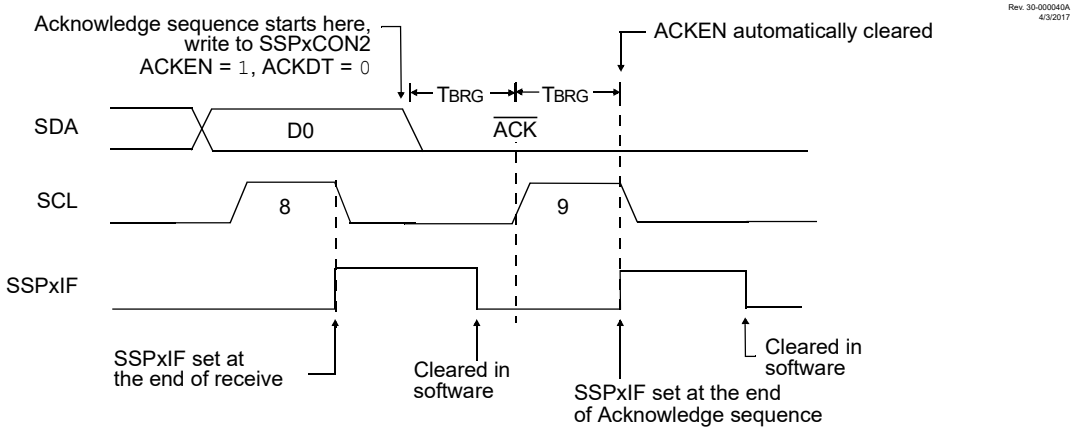
图 22-26. 重复启动条件波形图



22.2.4.1.5. 应答序列时序

通过将应答序列使能（ACKEN）位置 1 可使能应答序列（见图 22-27）。当该位被置 1 时，SCL 引脚被拉为低电平，应答数据（ACKDT）位的内容输出到 SDA 引脚上。如果用户希望产生应答，则必须将 ACKDT 位清零。否则，用户必须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计满返回周期（T_{BRG}）的计数，SCL 引脚被置为无效（拉为高电平）。当采样到 SCL 引脚为高电平（时钟仲裁）时，波特率发生器再进行一个 T_{BRG} 周期的计数。然后 SCL 引脚被拉为低电平。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 22-27. 应答序列波形图



Note: TBRG = one Baud Rate Generator period.

应答写冲突

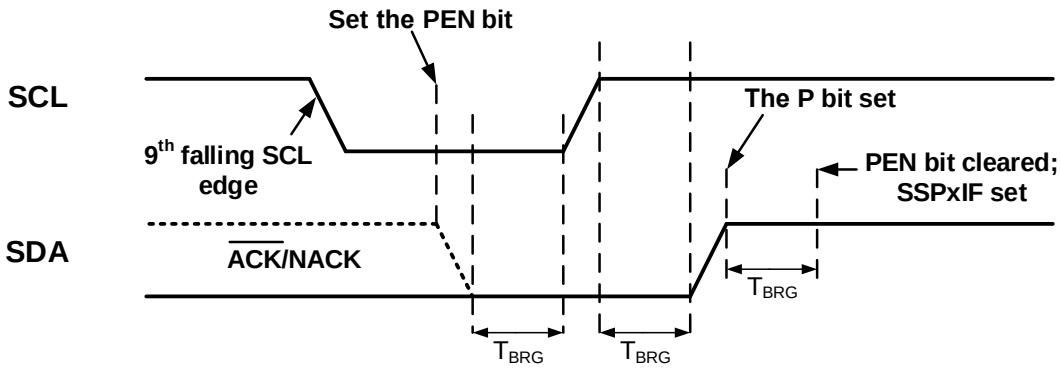
如果在应答序列进行过程中用户写 SSPxBUF，则 WCOL 位被置 1，同时缓冲区内容不变（未发生写操作）。

22.2.4.1.6. 停止条件时序

如果将停止条件使能（PEN）位置 1，则在接收/发送结束时，SDA 引脚上将产生停止条件（见图 22-28）。在接收/发送结束时，SCL 线在第 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。当采样到 SDA 线为低电平时，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 T_{BRG}（波特率发生器计满返回周期）之后，SDA 引脚将被拉

高。当采样到 SDA 引脚为高电平且 SCL 也是高电平时，P 位置 1。一个 T_{BRG} 后，PEN 位清零，且 SSPxIF 位置 1。

图 22-28. 接收或发送模式下的停止条件



停止条件下的写冲突

如果在停止序列进行过程中用户写 SSPxBUF，则 WCOL 位被置 1，同时缓冲区内容不变（未发生写操作）。

22.2.4.1.7. 在休眠模式下工作

在休眠模式下，I^2C 从模块可以接收地址或数据，并在发生地址匹配或完成字节传输时，将处理器从休眠模式唤醒（如果允许了 MSSP 中断）。

22.2.4.1.8. 复位的影响

复位会禁止 MSSP 模块并终止当前的数据传输。

22.2.4.2. I^2C 主模式发送

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简单地向 SSPxBUF 寄存器写入一个值来实现的。该操作将使缓冲区满状态（BF）位置 1，并使波特率发生器开始计数和开始下一次发送。

地址/数据的每一位将在 SCL 的下降沿置为有效之后移出到 SDA 引脚。在一个波特率发生器计满返回计数周期（T_{BRG}）内，SCL 保持低电平。在 SCL 被释放为高电平之前，数据必须保持有效。当 SCL 引脚释放为高电平时，它将在一个 T_{BRG} 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保持时间内，SDA 引脚上的数据必须保持稳定。在第 8 位数据被移出（第 8 个时钟的下降沿）之后，BF 标志被清零，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 个位时间发出一个 $\overline{ACK}$ 序列作为响应。$\overline{ACK}$ 的状态在第 9 个时钟的上升沿被写入应答状态（ACKSTAT）位。如果主器件接收到 $\overline{ACK}$，ACKSTAT 位会被清零。如果接收到 NACK，则 ACKSTAT 位被置 1。在第 9 个时钟之后，SSPxIF 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 SSPxBUF，SCL 保持低电平，SDA 保持不变（见图 22-29）。

在写 SSPxBUF 之后，地址的每一位在 SCL 的下降沿被移出，直到所有 7 个地址位和 $R/\overline{W}$ 位都被移出。在第 8 个时钟的下降沿，主器件将释放 SDA 引脚，以允许从器件发出 $\overline{ACK}$ 作为响应。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。$\overline{ACK}$ 位的状态被装入 ACKSTAT 位中。

在发送地址的第 9 个时钟下降沿之后，SSPxIF 置 1，BF 标志清零，波特率发生器关闭直到发生下一次写 SSPxBUF，且 SCL 保持低电平，允许 SDA 悬空。

22.2.4.2.1. BF 状态标志

在发送模式下，缓冲区满状态（BF）位在 CPU 写 SSPxBUF 时置 1，在所有 8 位数据移出后清零。

22.2.4.2.2. WCOL 状态标志

如果在发送过程中（即，SSPSR 仍在移出数据字节时）用户写 SSPxBUF，则写冲突检测（WCOL）位被置 1，同时缓冲区内容不变（未发生写操作）。

在下一次发送前 WCOL 位必须用软件清零。

22.2.4.2.3. ACKSTAT 状态标志

在发送模式下，当从器件已发送应答（$\overline{ACK}$= 0）时，应答状态（ACKSTAT）位被清零；当从器件发出 NACK 时，该位被置 1。从器件在识别出其地址（包括广播呼叫地址）或正确接收数据后，会发送一个 $\overline{ACK}$。

22.2.4.2.4. 典型发送序列

1. 主器件通过将 SEN 位置 1 来产生启动条件。
2. 启动条件完成时，SSPxIF 由硬件置 1。
3. SSPxIF 由软件清零。
4. 在发生任何其他操作之前，MSSP 模块将等待所需的启动时间。
5. 软件为 SSPxBUF 装入从器件地址和 R/$\overline{W}$ 位。在主发送模式下，R/$\overline{W}$ 值为 0。
6. 地址从 SDA 引脚移出，直到发送完所有 8 位地址。写 SSPxBUF 时便开始发送。
7. MSSP 模块移入来自从器件的 $\overline{ACK}$ 值，并将其值写入 ACKSTAT 位。
8. 在第 9 个时钟周期结束时，MSSP 模块通过将 SSPxIF 位置 1 产生中断。
9. 软件将 8 位数据装入 SSPxBUF。
10. 数据从 SDA 引脚移出，直到发送完所有 8 位数据。
11. MSSP 模块移入来自从器件的 $\overline{ACK}$ 位，并将其值写入 ACKSTAT 位。
12. 对于所有发送的数据字节重复步骤 8-11。
13. 用户通过分别将 PEN 或 RSEN 位置 1 来产生停止或重复启动条件。停止/重复启动条件完成时产生中断。

图 22-29. I^2C 主模式波形图（发送，7 位地址）

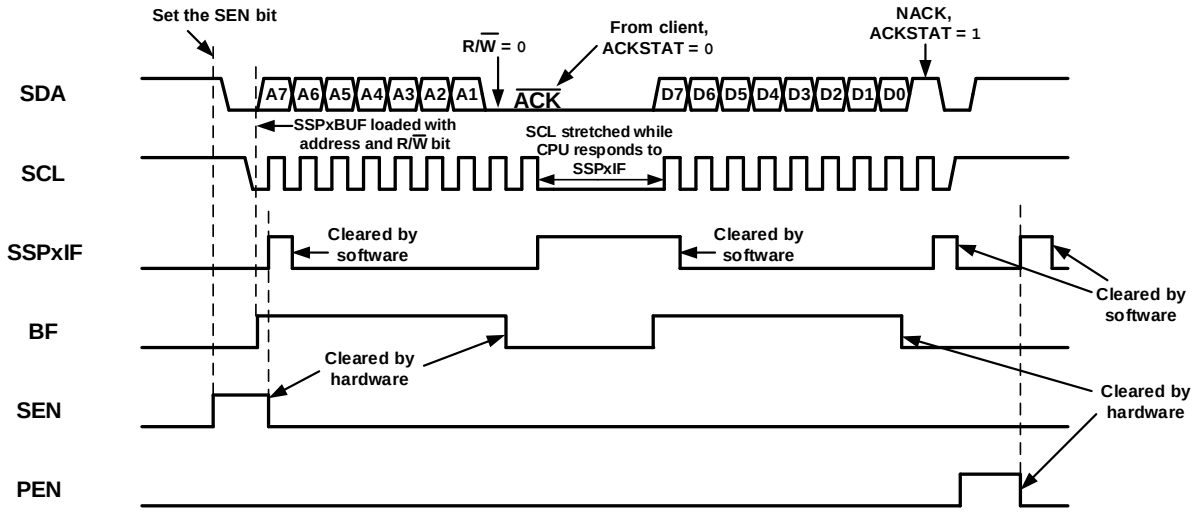
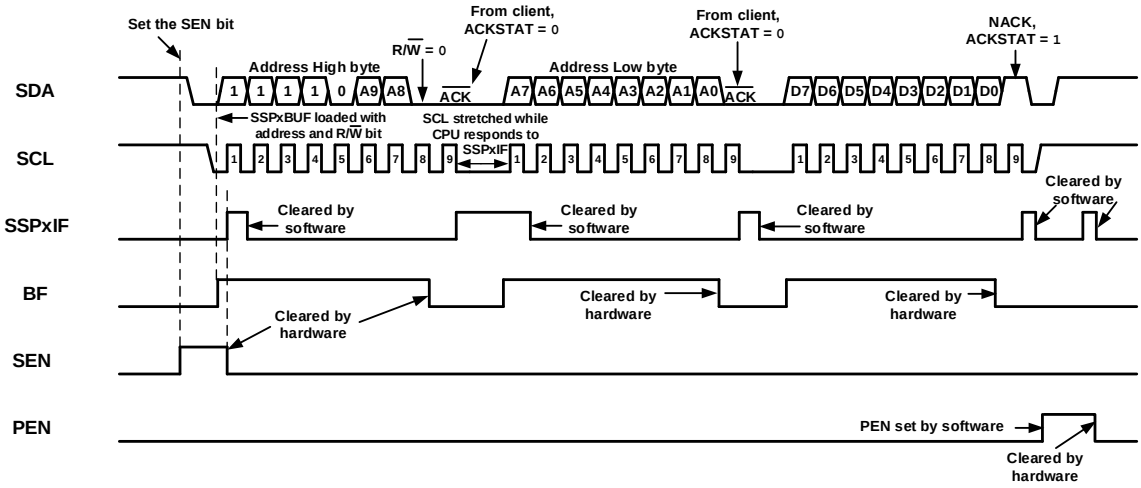


图 22-30. I²C 主模式波形图（发送，10 位地址）



22.2.4.3. I²C 主模式接收

通过将接收使能（RCEN）位置 1 可使能主模式接收（见图 22-31）。

重要：将 RCEN 位置 1 前，MSSP 模块必须处于空闲状态，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 SSPxSR。在第 8 个时钟的下降沿之后，将发生以下所有事件：

- RCEN 由硬件自动清零。
- SSPxSR 的内容装入 SSPxBUF。
- BF 标志位置 1。
- SSPxIF 标志位置 1。
- 波特率发生器暂停计数。
- SCL 引脚保持为低电平。

此时 MSSP 处于空闲状态，等待下一条命令。当 CPU 读缓冲区时，BF 标志位会自动清零。通过将应答序列使能（ACKEN）位置 1，主器件可以在接收结束时发送应答序列。

22.2.4.3.1. BF 状态标志

在接收操作中，当地址或数据字节从 SSPSR 装入到 SSPxBUF 中时，BF 位置 1。读取 SSPxBUF 寄存器时，该位清零。

22.2.4.3.2. SSPOV 状态标志

在接收操作中，当 SSPxSR 接收到 8 位且 BF 标志位已经在上一次接收中置 1 时，SSPOV 位置 1。

22.2.4.3.3. WCOL 状态标志

如果在接收过程中（即，SSPxSR 仍在移入数据字节时）用户写 SSPxBUF，则 WCOL 位被置 1，同时缓冲区内容不变（未发生写操作）。

22.2.4.3.4. 典型接收序列

1. 主器件通过将 SEN 位置 1 来产生启动条件。
2. 启动条件完成时，SSPxIF 由硬件置 1。
3. SSPxIF 由软件清零。
4. 软件将要发送的从器件地址写入 SSPxBUF 且 R/$\overline{W}$ 位置 1。
5. 地址从 SDA 引脚移出，直到发送完所有 8 位地址。写 SSPxBUF 时便开始发送。
6. MSSP 模块移入来自从器件的 $\overline{ACK}$ 值，并将其写入 ACKSTAT 位。
7. 在第 9 个时钟周期结束时，MSSP 模块通过将 SSPxIF 位置 1 产生中断。
8. 软件将 RCEN 位置 1，主器件从从器件移入一个字节。
9. 在 SCL 的第 8 个下降沿之后，SSPxIF 和 BF 置 1。
10. 主器件清零 SSPxIF，并从 SSPxBUF 中读取接收到的字节，使 BF 清零。
11. 主器件通过将 ACKEN 位置 1 来清零 ACKDT 位并启动 $\overline{ACK}$ 序列。
12. 主器件随时钟将 $\overline{ACK}$ 移出到从器件，SSPxIF 置 1。
13. 用户清零 SSPxIF。
14. 对于从从器件接收到的每个字节重复步骤 8-13。
15. 主器件通过发送 NACK 信号或停止条件来结束通信。

图 22-31. I²C 主模式波形图（接收，7 位地址）

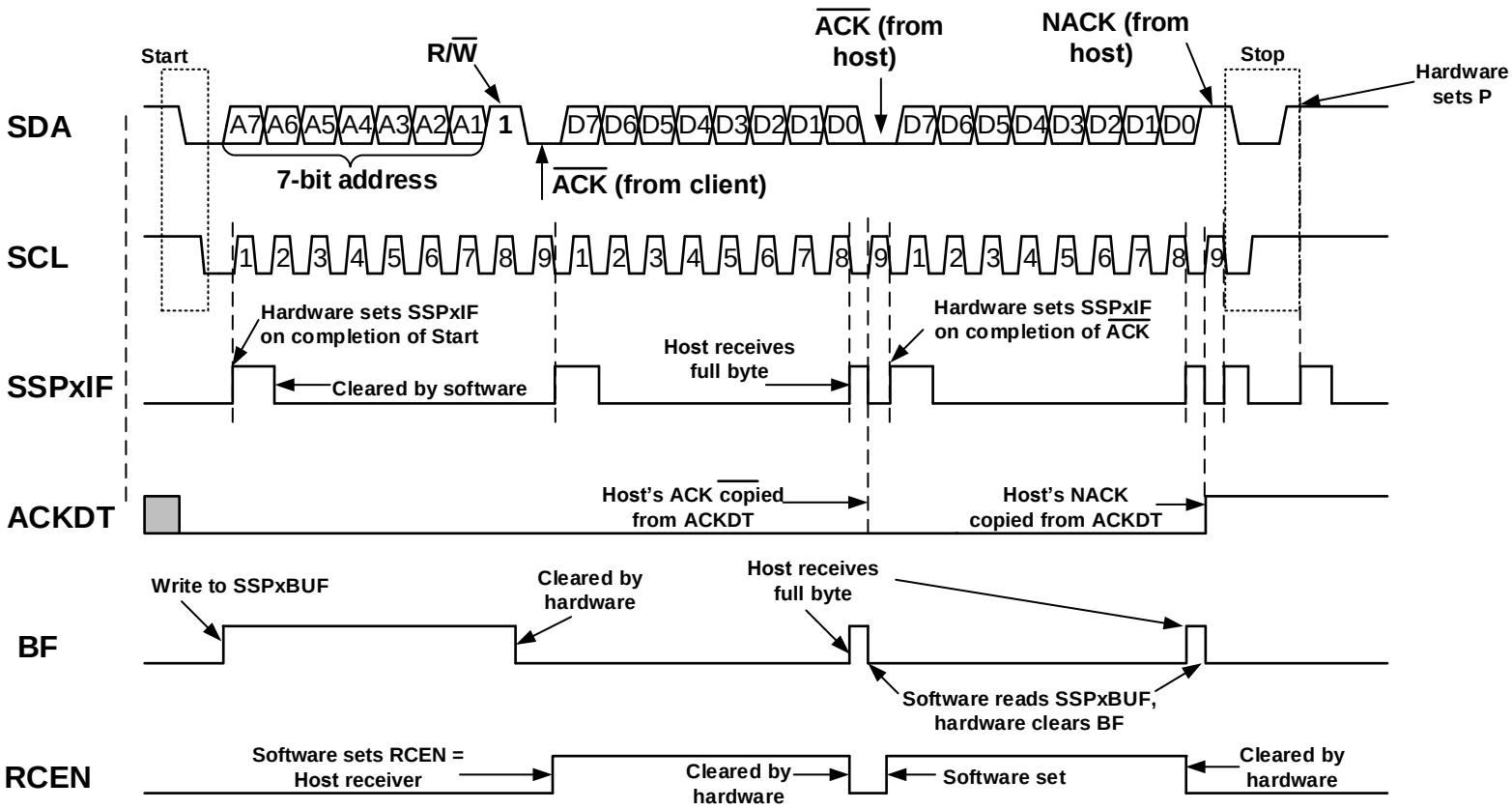
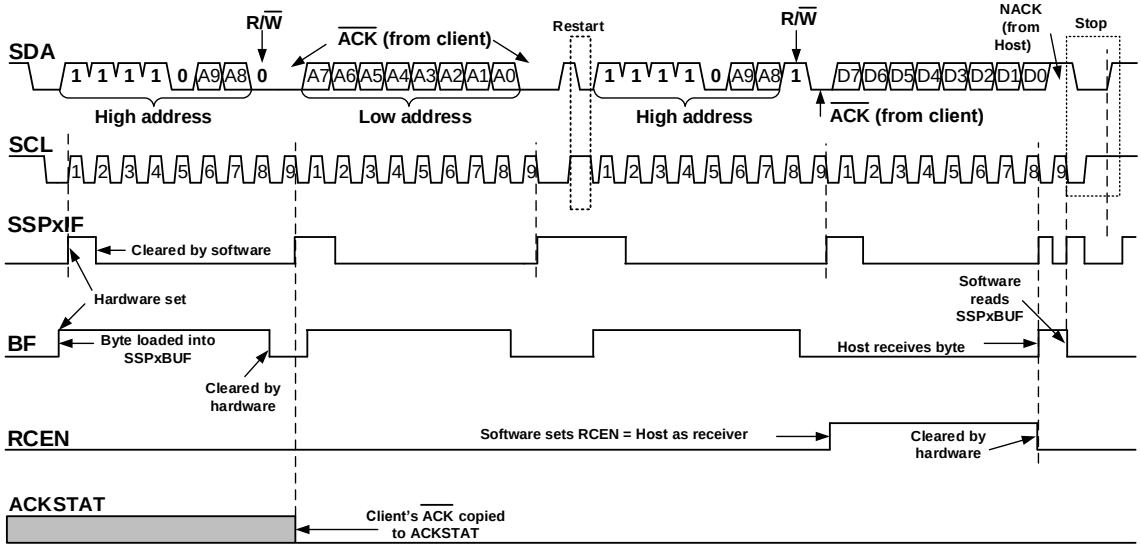


图 22-32. I²C 主模式波形图（接收，10 位地址）



22.2.5. 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，这可用于判断总线何时空闲。停止（P）位和启动（S）位在复位或禁止 MSSP 模块时清零。当 P 位置 1 或总线空闲时，可以获得 I²C 总线的控制权，S 位和 P 位都将清零。当总线忙且允许 MSSP 中断时，一旦发生停止条件便产生中断。

在多主器件操作中，必须监视 SDA 线来进行仲裁，以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 BCLxIF 位中。

可能导致仲裁失败的状态为：

- 地址传输
- 数据传输
- 启动条件
- 重复启动条件
- 应答条件

22.2.5.1. 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 引脚上输出 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。当 SCL 引脚悬空为高电平时，数据可处于稳定状态。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。主器件会将总线冲突中断标志位（BCLxIF）置 1，并将 I²C 端口复位为空闲状态（图 22-33）。

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 线被置为无效，并且可写入 SSPxBUF。当执行总线冲突中断服务程序时，如果 I²C 总线空闲，软件可通过发出启动条件恢复通信。

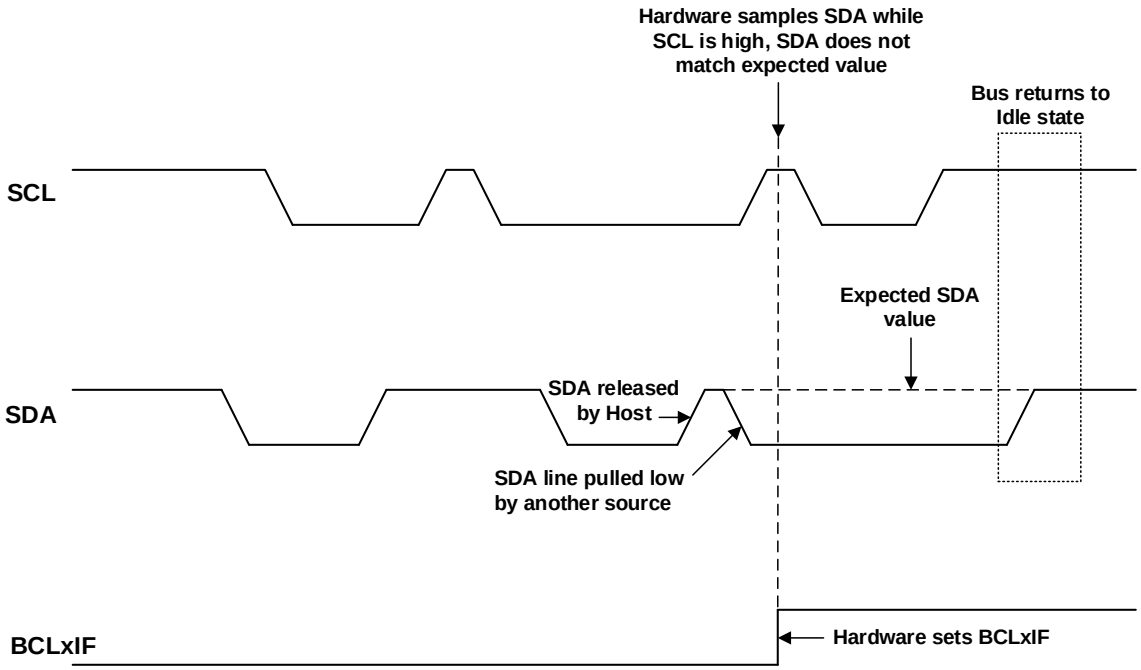
如果在启动、重复启动、停止或应答条件的进行过程中发生总线冲突，则条件将被中止，SDA 和 SCL 线被置为无效，SSPxCON2 寄存器中的对应控制位清零。当执行总线冲突中断服务程序时，如果 I²C 总线空闲，软件可通过发出启动条件恢复通信。

主器件将继续监视 SDA 和 SCL 引脚。如果出现停止条件，SSPxIF 位将置 1。

发生总线冲突时无论发送的进度如何，写入 SSPxBUF 都会从第一个数据位开始发送数据。

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，这可用于判断总线何时空闲。当 P 位置 1 或总线空闲时，可以获得 I^2C 总线的控制权，S 位和 P 位都清零。

图 22-33. 发送和应答时的总线冲突时序



22.2.5.1.1. 启动条件期间的总线冲突

启动条件期间，出现以下情况时发生总线冲突：

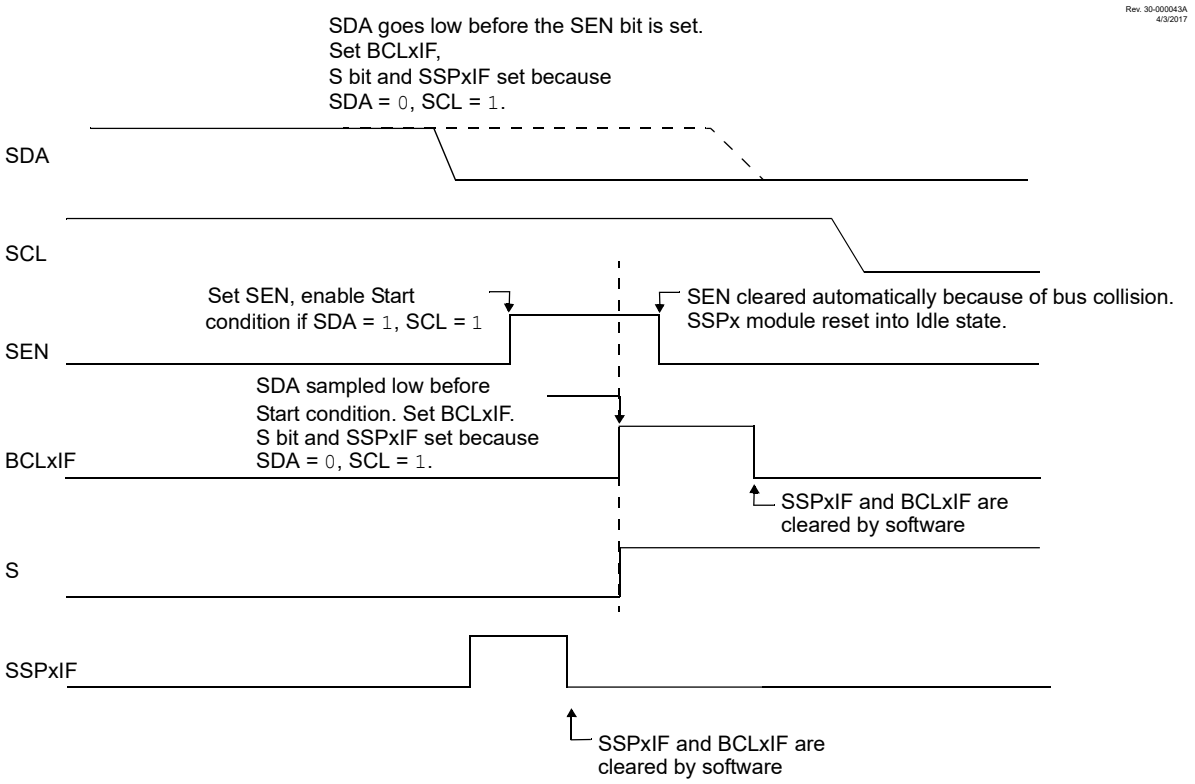
1. 在启动条件开始时，采样到 SDA 或 SCL 为低电平（见图 22-34）。
2. SDA 置为低电平之前，采样到 SCL 为低电平（见图 22-35）。

启动条件期间，监视 SDA 和 SCL 引脚。

如果 SDA 引脚或 SCL 引脚已经是低电平，则发生以下所有事件：

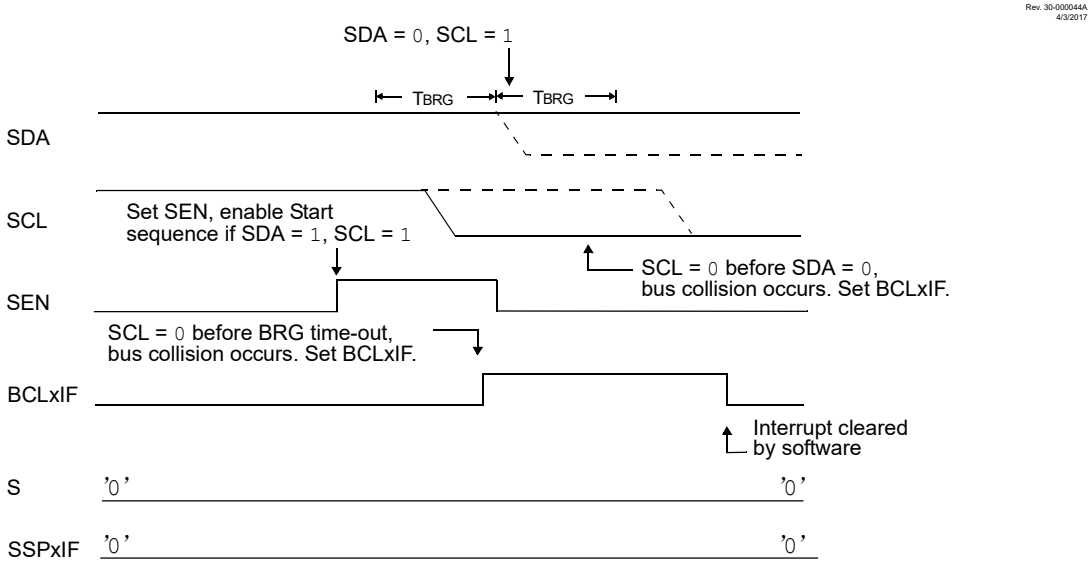
- 中止启动条件，
- BCLxIF 标志置 1，并且
- MSSP 模块复位为空闲状态（见图 22-34）。

图 22-34. 启动条件期间的总线冲突（仅用于 SDA）



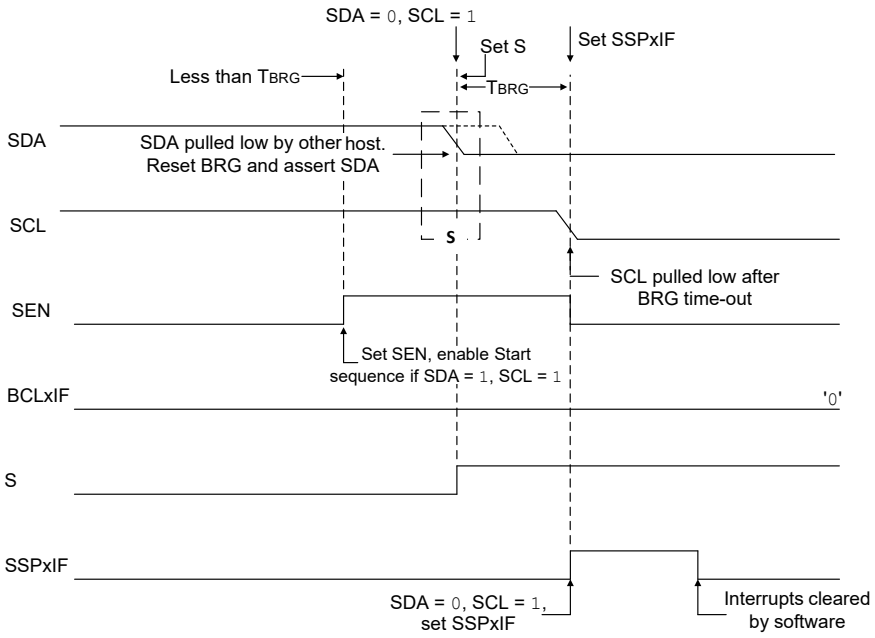
启动条件从 SDA 和 SCL 引脚被置为无效时开始。当采样到 SDA 引脚为高电平时，波特率发生器装入值并递减计数。如果在 SDA 为高电平时，采样到 SCL 引脚为低电平，则发生总线冲突，因为这表示另一个主器件在启动条件期间尝试驱动数据 1。

图 22-35. 启动条件期间的总线冲突（SCL = 0）



如果采样到 SDA 引脚在该计数周期内为低电平，则 BRG 复位，且 SDA 线提前置为有效（见图 22-36）。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 引脚被置为低电平。

图 22-36. 启动条件期间由 SDA 仲裁引起的 BRG 复位



重要：在启动条件期间不会发生总线冲突，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 置为有效。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。

22.2.5.1.2. 重复启动条件期间的总线冲突

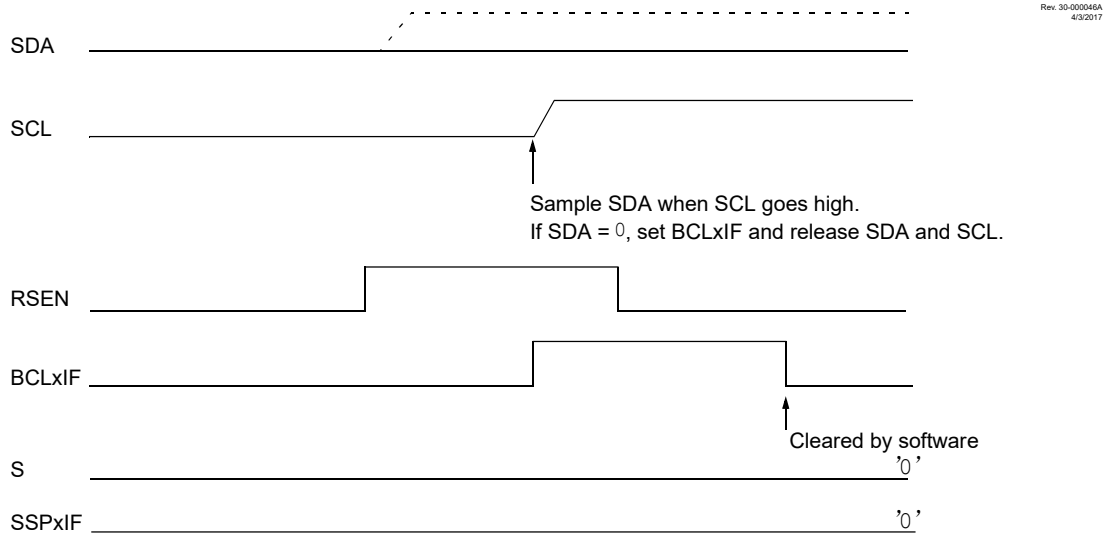
在重复启动条件期间如果发生以下情况，将发生总线冲突：

1. 在 SCL 由低电平变为高电平期间，在 SDA 上采样到低电平（见图 22-37）。
2. 在 SDA 置为低电平之前，SCL 变为低电平，表示另一个主器件正尝试发送数据 1（见图 22-38）。

当用户释放 SDA 并允许该引脚悬空为高电平时，BRG 装入 SSPxADD 的值并递减计数至 0。接着 SCL 引脚被置为无效，当采样到 SCL 引脚为高电平时，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，另一个主器件正尝试发送数据 0，见图 22-37）。如果采样到 SDA 为高电平，则 BRG 被重载并开始计数。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 置为有效。

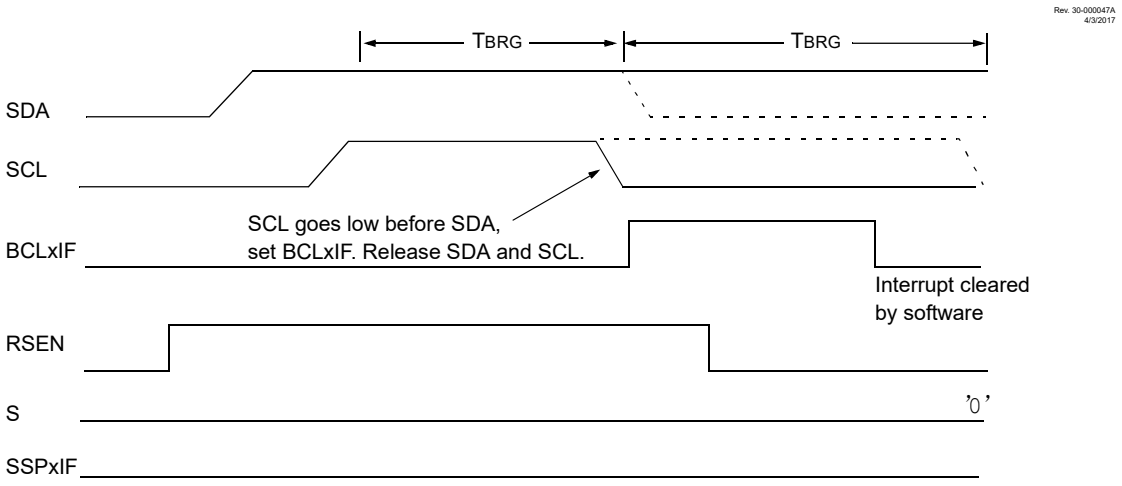
图 22-37. 重复启动条件期间的总线冲突（情形 1）



如果 SCL 在 BRG 超时之前从高电平变为低电平，且 SDA 尚未被置为有效，则将发生总线冲突。在这种情况下，另一个主器件在重复启动条件期间正在尝试发送数据 1（见图 22-38）。

如果在 BRG 超时结束时 SCL 和 SDA 都仍然是高电平，则 SDA 引脚被驱动为低电平，BRG 被重载并开始计数。在计数结束时，不管 SCL 引脚的状态如何，SCL 引脚都被驱动为低电平，重复启动条件结束。

图 22-38. 重复启动条件期间的总线冲突（情形 2）



22.2.5.1.3. 停止条件期间的总线冲突

在停止条件期间如果发生以下情况，将发生总线冲突：

1. 在 SDA 引脚已置为无效并允许悬空为高电平之后，在 BRG 超时后采样到 SDA 引脚为低电平（见图 22-39）。

2. 在 SCL 引脚被置为无效后，在 SDA 变为高电平之前采样到 SCL 引脚为低电平（见图 22-40）。

停止条件从 SDA 被置为低电平开始。当采样到 SDA 为低电平时，允许 SCL 引脚悬空。当采样到引脚为高电平（时钟仲裁）时，波特率发生器装入 SSPxADD 的值并递减计数至 0。BRG 超时后，SDA 被采样。如果采样到 SDA 为低电平，则已发生总线冲突。这是因为另一个主器件正在尝试驱动数据 0（见图 22-39）。如果在允许 SDA 悬空为高电平前 SCL 引脚被采样到低电平，也会发生总线冲突。这是有另一个主器件正在尝试驱动数据 0 的另一种情况（图 22-40）。

图 22-39. 停止条件期间的总线冲突（情形 1）

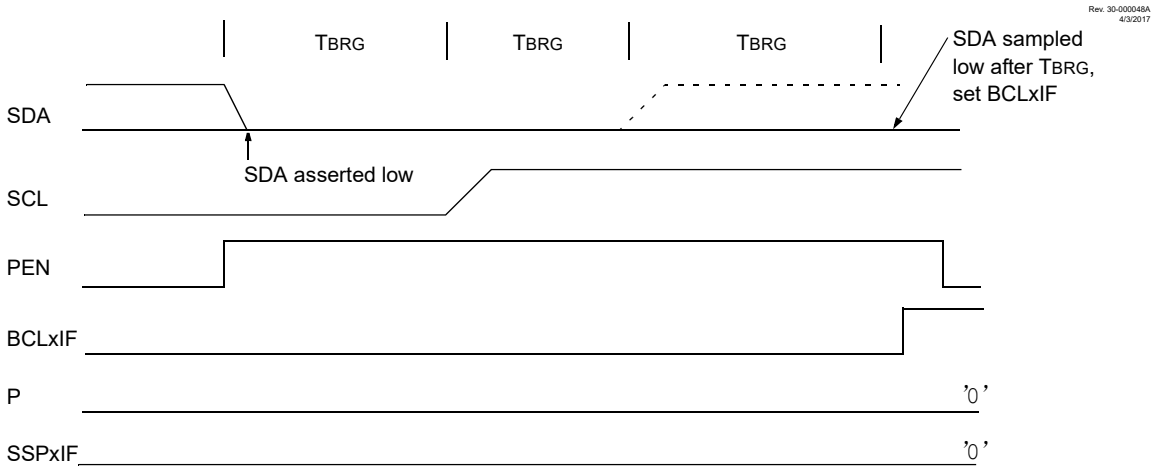
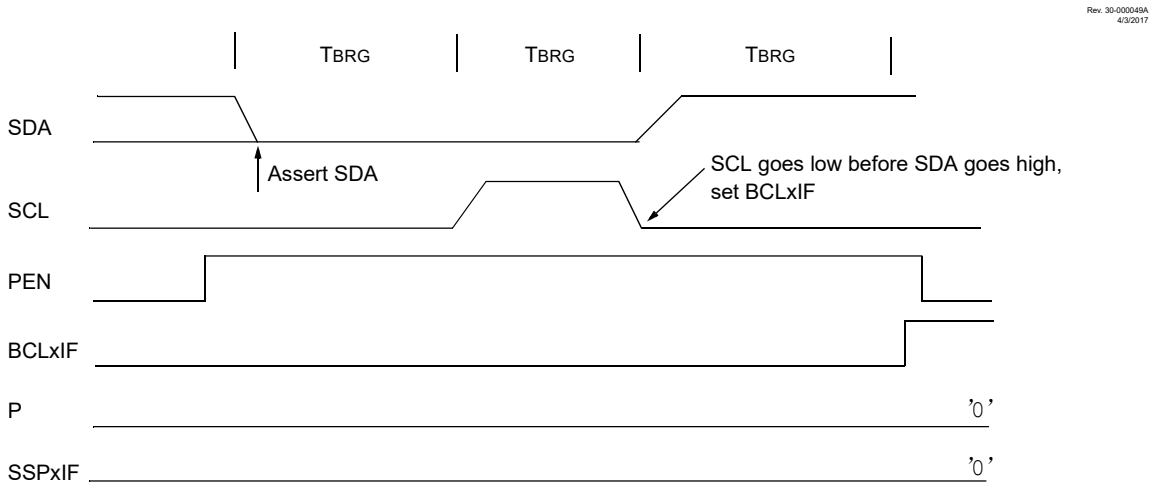


图 22-40. 停止条件期间的总线冲突（情形 2）



22.3. 波特率发生器

MSSP 模块具有波特率发生器（BRG），可用于在 I^2C 和 SPI 主模式下产生时钟。BRG 的重载值保存在 SSPxADD 寄存器中。写 SSPxBUF 时，BRG 将自动开始递减计数。 例 22-1 给出了 SSPxADD 值的计算方式。

完成给定操作时，内部时钟将自动停止计数，时钟引脚将保持最新的状态。

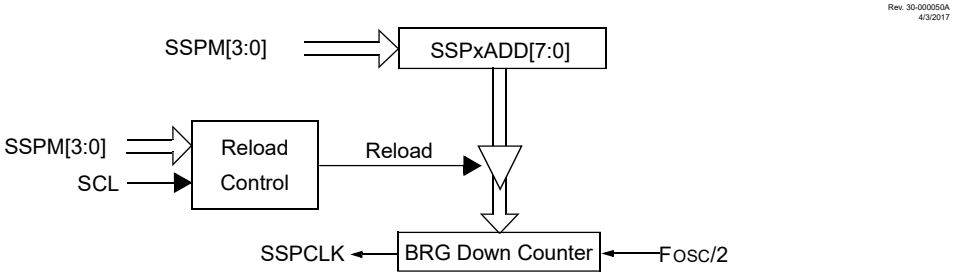
图 22-41 中显示的内部“重载”信号会触发将 SSPxADD 值装入 BRG 计数器。对于模块时钟线的每次振荡，这会发生两次。

表 22-2 演示了不同指令周期下的时钟速率以及装入 SSPxADD 的 BRG 值。

例 22-1. MSSP 波特率发生器频率公式

$$F_{CLOCK} = \frac{F_{OSC}}{4 \times (SSPxADD + 1)}$$

图 22-41. 波特率发生器框图



重要：在用作 I²C 的波特率发生器时，值 0x00、0x01 和 0x02 对于 SSPxADD 是无效的。这是固有的限制。

表 22-2. 使用 BRG 的 MSSP 时钟速率

F_{OSC}	F_{CY}	BRG 值	F_{CLOCK}（2 次 BRG 计满返回）
32 MHz	8 MHz	13h	400 kHz
32 MHz	8 MHz	19h	308 kHz
32 MHz	8 MHz	4Fh	100 kHz
16 MHz	4 MHz	09h	400 kHz
16 MHz	4 MHz	0Ch	308 kHz
16 MHz	4 MHz	27h	100 kHz
4 MHz	1 MHz	09h	100 kHz

注：要确保所设计的系统支持所有要求，请参见**“电气规范”**一章中“内部振荡器参数”部分的 I/O 端口电气规范。

22.4. 寄存器定义：MSSP 控制

22.4.1. SSPxBUF

名称： SSPxBUF
偏移量： 0x018C

MSSP 数据缓冲寄存器

位	7	6	5	4	3	2	1	0
	BUF[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	x	x	x	x	x	x	x	x

Bit 7:0 – BUF[7:0] MSSP 输入和输出数据缓冲位

22.4.2. SSPxADD

名称： SSPxADD
偏移量： 0x018D

MSSP 波特率分频比和地址寄存器

位	7	6	5	4	3	2	1	0
	ADD[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7:0 – ADD[7:0]

- SPI 和 I^2C 主模式：波特率分频比
- I^2C 从模式：地址位

值	Mode	说明
11111111 – 00000011	SPI 和 I^2C 主模式	波特率分频比。SCK/SCL 引脚时钟周期 = ((n + 1) * 4)/F_{OSC}。在 I^2C 模式下，小于 3 的值无效。
xxxxx11x– xxxxx00x	I^2C 10 位从模式 MS 地址	Bit [7:3]和 Bit 0 不使用，它们是无关位。Bit [2:1]为 10 位从模式最高有效地址的 bit [9:8]。
11111111 – 00000000	I^2C 10 位从模式 LS 地址	10 位从模式最低有效地址的 bit [7:0]
1111111x – 0000000x	I^2C 7 位从模式	Bit 0 不使用，它是无关位。Bit [7:1]是 7 位从器件地址。

22.4.3. SSPxMSK

名称： SSPxMSK
偏移量： 0x018E

MSSP 地址掩码寄存器

位	7	6	5	4	3	2	1	0
	MSK[6:0]							MSK0
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	1	1	1	1	1	1	1	1

Bit 7:1 – MSK[6:0] 掩码位

值	Mode	说明
1	I^2C 从模式	接收到的地址 bit n 与 SSPxADD bit n 相比较来检测 I^2C 模式下地址是否匹配
0	I^2C 从模式	接收到的地址 bit n 不用于检测 I^2C 模式下地址是否匹配

Bit 0 – MSK0 用于 I^2C 10 位从模式的掩码位

值	Mode	说明
x	SPI 或 I^2C 7 位	不使用该位
1	I^2C 10 位从模式	接收到的地址 bit 0 与 SSPxADD bit 0 相比较来检测 I^2C 模式下地址是否匹配
0	I^2C 10 位从模式	接收到的地址 bit 0 不用于检测 I^2C 模式下地址是否匹配

22.4.4. SSPxSTAT

名称： SSPxSTAT
偏移量： 0x018F

MSSP 状态寄存器

位	7	6	5	4	3	2	1	0
	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
访问	R/W	R/W	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0

Bit 7 – SMP 压摆率控制位

值	Mode	说明
1	SPI 主模式	在数据输出时间的末端采样输入数据
0	SPI 主模式	在数据输出时间的中间采样输入数据
0	SPI 从模式	在 SPI 从模式下，必须将该位清零
1	I^2C	标准速度模式（100 kHz）下禁止压摆率控制
0	I^2C	高速模式（400 kHz）下使能压摆率控制

Bit 6 – CKE SPI：时钟选择位[4]；I^2C：SMBus 选择位

值	Mode	说明
1	SPI	时钟状态从有效转换到空闲时发送
0	SPI	时钟状态从空闲转换到有效时发送
1	I^2C	使能 SMBus 特定输入
0	I^2C	禁止 SMBus 特定输入

Bit 5 – D/$\overline{A}$ 数据/$\overline{地址}$位

值	Mode	说明
x	SPI 或 I^2C 主模式	不使用该位
1	I^2C 从模式	指示上一个接收或发送的字节是数据
0	I^2C 从模式	指示上一个接收或发送的字节是地址

Bit 4 – P 停止位[1]

值	Mode	说明
x	SPI	不使用该位
1	I^2C	上次检测到停止位
0	I^2C	上次未检测到停止位

Bit 3 – S 启动位[1]

值	Mode	说明
x	SPI	不使用该位
1	I^2C	上次检测到启动位
0	I^2C	上次未检测到起始位

Bit 2 – R/$\overline{W}$ 读/$\overline{写}$信息位[2,3]

值	Mode	说明
x	SPI	不使用该位
1	I^2C 从模式	读
0	I^2C 从模式	写

值	Mode	说明
1	I²C 主模式	正在进行发送
0	I²C 主模式	未进行发送

Bit 1 – UA 更新地址位（仅限 10 位 I²C 从模式）

值	Mode	说明
x	所有其他模式	不使用该位
1	I²C 10 位从模式	表示用户需要更新 SSPxADD 寄存器中的地址
0	I²C 10 位从模式	不需要更新地址

Bit 0 – BF 缓冲区满状态位(5)

值	Mode	说明
1	I²C 发送	发送正在进行中，SSPxBUF 已满
0	I²C 发送	发送完成，SSPxBUF 为空
1	SPI 和 I²C 接收	接收完成，SSPxBUF 已满
0	SPI 和 I²C 接收	接收未完成，SSPxBUF 为空

注：

1. 该位在复位时以及 SSPEN 清零时清零。
2. 在 I²C 从模式下，该位保存上一次地址匹配后的 R/$\overline{W}$ 位信息。该位仅在从地址匹配到出现下一个启动位、停止位或非 $\overline{ACK}$ 位之间有效。
3. 将该位与 SEN、RSEN、PEN、RCEN 或 ACKEN 的或运算结果指示 MSSP 是否处于工作模式。
4. 时钟极性状态由 CKP 位设置。
5. I²C 接收状态不包含 $\overline{ACK}$ 和停止位。

22.4.5. SSPxCON1

名称： SSPxCON1
偏移量： 0x0190

MSSP 控制寄存器 1

位	7	6	5	4	3	2	1	0
	WCOL	SSPOV	SSPEN	CKP	SSPM[3:0]			
访问	R/W/HS	R/W/HS	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7 – WCOL 写冲突检测位

值	Mode	说明
x	主器件或从器件接收	不使用该位
1	SPI 或 I²C 主器件或从器件发送	正在发送前一个字时，又有数据写入 SSPxBUF 寄存器（必须用软件清零）
0	SPI 或 I²C 主器件或从器件发送	无冲突

Bit 6 – SSPOV 接收溢出指示位(1)

值	Mode	说明
x	SPI 主器件或 I²C 主器件发送	不使用该位
1	SPI 从器件	SSPxBUF 寄存器仍存有前一字节时，又接收到一个字节。移位寄存器中的数据将被丢弃。即使只是发送数据，用户也必须读 SSPxBUF，以避免溢出标志位置 1。
1	I²C 接收	SSPxBUF 寄存器仍存有前一字节时，又接收到一个字节（必须用软件清零）。
0	SPI 从器件或 I²C 接收	无溢出

Bit 5 – SSPEN 主同步串行端口使能位(2)

值	说明
1	使能串行端口
0	禁止串行端口并将这些引脚配置为 I/O 端口引脚

Bit 4 – CKP SCK 释放控制位

值	Mode	说明
x	I²C 主器件	不使用该位
1	SPI	时钟的空闲状态为高电平
0	SPI	时钟的空闲状态为低电平
1	I²C 从器件	释放时钟
0	I²C 从器件	保持时钟为低电平（时钟延长），用来确保数据建立时间

Bit 3:0 – SSPM[3:0] 主同步串行端口模式选择位(4)

值	说明
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
1101	保留，不要使用
1100	保留，不要使用
1011	I²C 固件控制的主模式（从器件空闲）
1010	SPI 主模式：时钟 = $F_{OSC}/(4*(SSPxADD+1))$
1001	保留，不要使用
1000	I²C 主模式：时钟 = $F_{OSC}/(4 * (SSPxADD + 1))$(3)
0111	I²C 从模式：10 位地址

值	说明
0110	I^2C 从模式：7 位地址
0101	SPI 从模式：时钟 = SCKx 引脚。禁止 $\overline{SSx}$ 引脚控制
0100	SPI 从模式：时钟 = SCKx 引脚。使能 $\overline{SSx}$ 引脚控制
0011	SPI 主模式：时钟 = TMR2 输出/2
0010	SPI 主模式：时钟 = F_{OSC}/64
0001	SPI 主模式：时钟 = F_{OSC}/16
0000	SPI 主模式：时钟 = F_{OSC}/4

注：

1. 在主模式下，溢出位不会置 1，因为每次都通过写入 SSPxBUF 寄存器来启动新数据的接收（和发送）。
2. 当使能时，必须将这些引脚正确地配置为输入或输出。
3. I^2C 模式不支持 SSPxADD 值为 0、1 和 2 的情况。
4. 此处未明确列出的位组合保留或仅在 I^2C 模式下实现。

22.4.6. SSPxCON2

名称： SSPxCON2
偏移量： 0x0191

MSSP 控制寄存器 2

控制寄存器（仅限 I²C 操作）

位	7	6	5	4	3	2	1	0
	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
访问	R/W	R//HC/HS	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7 – GCEN 广播呼叫使能位（仅限从模式）

值	模式	说明
x	主模式	无关
1	从模式	使能广播呼叫
0	从模式	不使能广播呼叫

Bit 6 – ACKSTAT 应答状态位（仅限主发送模式）

值	说明
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（仅限主接收模式）(1)

值	说明
1	不应答
0	应答

Bit 4 – ACKEN 应答序列使能位(2)

值	说明
1	在 SDAx 和 SCLx 引脚上发出应答序列，并发送 ACKDT 数据位；由硬件自动清零
0	应答序列空闲

Bit 3 – RCEN 接收使能位（仅限主接收模式）(2)

值	说明
1	使能 I²C 接收模式
0	接收空闲

Bit 2 – PEN 停止条件使能位（仅限主模式）(2)

值	说明
1	在 SDAx 和 SCLx 引脚上发出停止条件；由硬件自动清零
0	停止条件空闲

Bit 1 – RSEN 重复启动条件使能位（仅限主模式）(2)

值	说明
1	在 SDAx 和 SCLx 引脚上发出重复启动条件；由硬件自动清零
0	重复启动条件空闲

Bit 0 – SEN 启动条件使能位(2)

值	模式	说明
1	主模式	在 SDAx 和 SCLx 引脚上发出启动条件；由硬件自动清零
0	主模式	启动条件空闲
1	从模式	使能时钟延长
0	从模式	禁止时钟延长

注：

1. 当用户在接收结束后发出应答序列时将发送的值。
2. 如果 I^2C 模块处于工作模式，这些位可能不会置 1（没有缓存），并且可能也不会写入 SSPxBUF（或禁止写入 SSPxBUF）。

22.4.7. SSPxCON3

名称： SSPxCON3
偏移量： 0x0192

MSSP 控制寄存器 3

位	7	6	5	4	3	2	1	0
	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
访问	R/HS/HC	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7 – ACKTIM 应答时间状态位

值	Mode	说明
x	SPI 或 I²C 主模式	不使用该位
1	I²C 从模式且 AHEN = 1 或 DHEN = 1	SCL 的第 8 个下降沿已出现且 $\overline{ACK}$/NACK 状态有效
0	I²C 从模式	$\overline{ACK}$/NACK 状态无效。在 SCL 的第 9 个上升沿转换为低电平。

Bit 6 – PCIE 停止条件中断允许位

值	Mode	说明
x	SPI 或 SSPM = 1111 或 1110	不使用该位
1	SSPM ≠ 1111 且 SSPM ≠ 1110	允许在检测到停止条件时产生中断
0	SSPM ≠ 1111 且 SSPM ≠ 1110	禁止在检测到停止条件时产生中断

Bit 5 – SCIE 启动条件中断允许位

值	Mode	说明
x	SPI 或 SSPM = 1111 或 1110	不使用该位
1	SSPM ≠ 1111 且 SSPM ≠ 1110	允许在检测到启动条件时产生中断
0	SSPM ≠ 1111 且 SSPM ≠ 1110	禁止在检测到启动条件时产生中断

Bit 4 – BOEN 缓冲区改写使能位[(1)]

值	Mode	说明
1	SPI	每次移入一个新的数据字节，便会更新 SSPxBUF，与 BF 位无关
0	SPI	如果在 BF 位已置 1 的条件下接收到新字节，则 SSPOV 位置 1 且 SSPxBUF 不更新
1	I²C	每次移入一个新的数据字节，便会更新 SSPxBUF，可忽略更新缓冲区时的 SSPOV 影响
0	I²C	仅当 SSPOV 位清零时更新 SSPxBUF

Bit 3 – SDAHT SDA 保持时间选择位

值	Mode	说明
x	SPI	在 SPI 模式下不使用
1	I²C	在 SCL 的下降沿之后 SDA 至少保持 300 ns 的时间
0	I²C	在 SCL 的下降沿之后 SDA 至少保持 100 ns 的时间

Bit 2 – SBCDE 从模式总线冲突检测使能位
在主模式下未使用。

值	Mode	说明
x	SPI 或 I²C 主模式	不使用该位
1	I²C 从模式	使能总线冲突检测
0	I²C 从模式	不使能总线冲突检测

Bit 1 – AHEN 地址保持使能位

值	Mode	说明
x	SPI 或 I²C 主模式	不使用该位
1	I²C 从模式	使能地址保持。因此，在所接收地址字节的第 8 个 SCL 下降沿之后，CKP 位将清零。软件必须将 CKP 位置 1 才能恢复操作。
0	I²C 从模式	不使能地址保持

Bit 0 – DHEN 数据保持使能位

值	Mode	说明
x	SPI 或 I²C 主模式	不使用该位
1	I²C 从模式	使能数据保持。因此，在所接收数据字节的第 8 个 SCL 下降沿之后，CKP 位将清零。软件必须将 CKP 位置 1 才能恢复操作。
0	I²C 从模式	不使能数据保持

注：

1. 用于菊花链 SPI 操作；使用户可以忽略除最后一个接收到的字节之外的所有字节。当接收到新字节且 BF = 1 时，SSPOV 位仍然置 1，但是硬件继续将最新字节写入 SSPxBUF。

22.5. 寄存器汇总——MSSP 控制

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x018B	保留									
0x018C	SSP1BUF	7:0	BUF[7:0]							
0x018D	SSP1ADD	7:0	ADD[7:0]							
0x018E	SSP1MSK	7:0	MSK[6:0]							MSK0
0x018F	SSP1STAT	7:0	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
0x0190	SSP1CON1	7:0	WCOL	SSPOV	SSPEN	CKP	SSPM[3:0]			
0x0191	SSP1CON2	7:0	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
0x0192	SSP1CON3	7:0	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN

23. EUSART——增强型通用同步/异步收发器

增强型通用同步/异步收发器（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独立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。EUSART 也可称为串行通信接口（Serial Communication 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 模块包含以下功能：

- 全双工异步收发
- 双字符输入缓冲区
- 单字符输出缓冲区
- 字符长度可编程为 8 位或 9 位
- 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
- 同步模式下的可编程时钟极性
- 在休眠模式下工作

EUSART 模块还实现了以下功能，使其成为局域互联网络（Local Interconnect Network，LIN）总线系统的理想选择：

- 波特率的自动检测和校准
- 接收到断开字符时唤醒
- 13 位断开字符发送

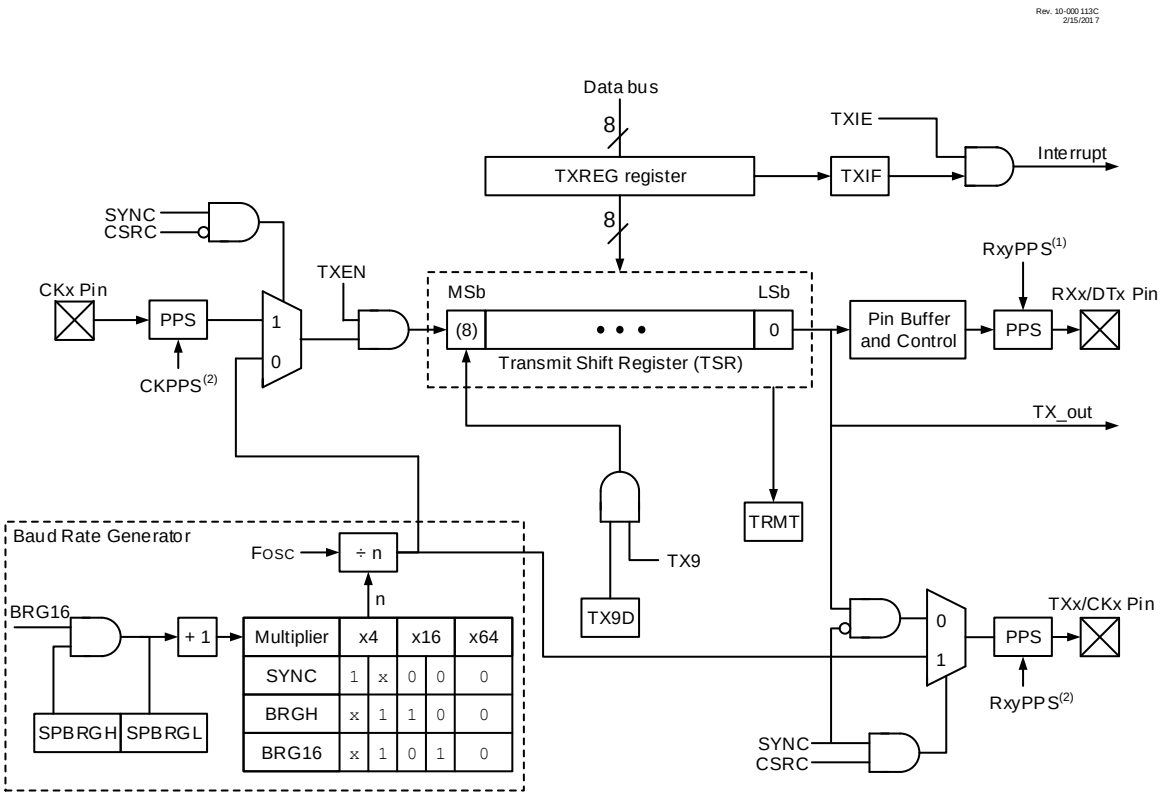
EUSART 发送器和接收器的框图如图 23-1 和图 23-2 所示。

EUSART 模块的操作由以下 6 个寄存器控制：

- 发送状态和控制寄存器（TXxSTA）
- 接收状态和控制寄存器（RCxSTA）
- 波特率控制寄存器（BAUDxCON）
- 波特率值寄存器（SPxBRG）
- 接收数据寄存器（RCxREG）
- 发送数据寄存器（TXxREG）

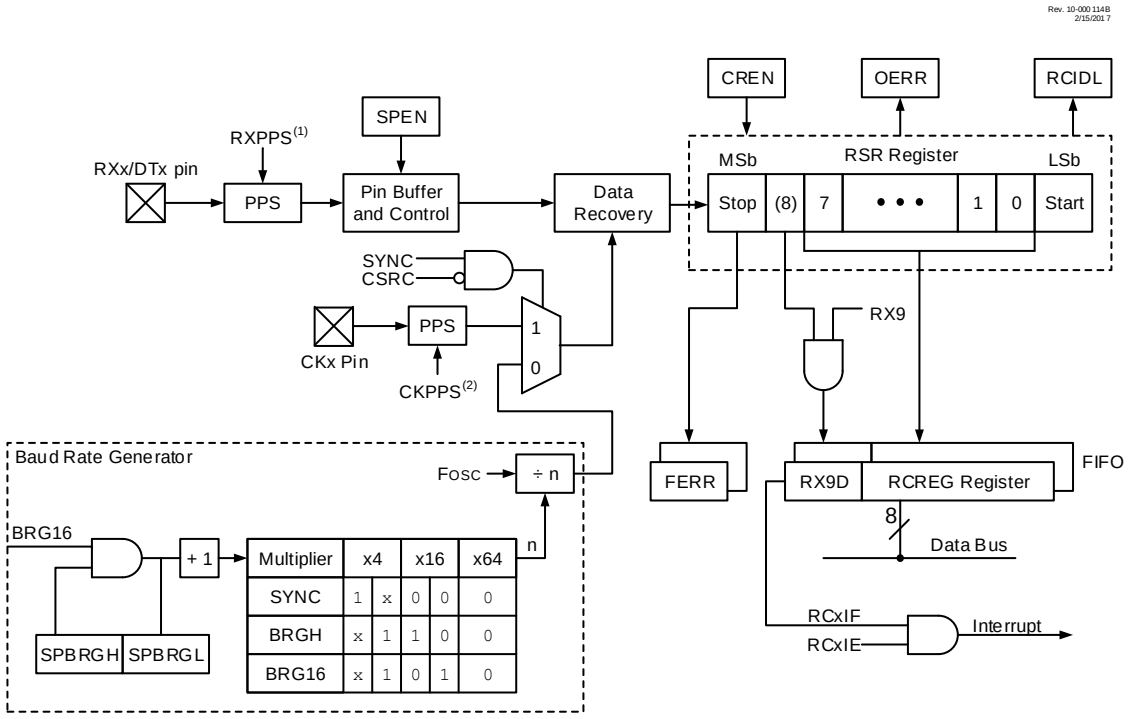
RXx/DTx 和 TXx/CKx 输入引脚分别使用 RXxPPS 和 TXxPPS 寄存器进行选择。TXx、CKx 和 DTx 输出引脚使用每个引脚的 RxyPPS 寄存器进行选择。由于在同步模式下 RX 输入与 DT 输出耦合在一起，所以在同步模式下工作时，用户需要负责为这两个功能选择同一引脚。EUSART 控制逻辑将自动控制数据方向驱动器。

图 23-1. EUSART 发送框图



Notes: 1. In Synchronous mode, the DT output and RX input PPS selections will enable the same pin.
2. In Host Synchronous mode, the TX output and CK input PPS selections will enable the same pin.

图 23-2. EUSART 接收框图



Notes: 1. In Synchronous mode, the DT output and RX input PPS selections will enable the same pin.
2. In Host Synchronous mode, the TX output and CK input PPS selections will enable the same pin.

23.1. 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：V_{OH} 标记状态（Mark state）代表 1 数据位，而 V_{OL} 空格状态（Space state）代表 0 数据位。NRZ 指的是连续发送的具有相同值的数据位保持在相应位的输出电平，而不会在发送完每个位之后回到中间电平。NRZ 发送端口在标记状态下为空闲。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。最常用的数据格式为 8 位。每个发送的位保持时间为 1/(波特率)。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 23-2。

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，但可通过软件实现奇偶校验，并将奇偶校验位作为第 9 个数据位存储。

23.1.1. EUSART 异步发送器

图 23-1 给出了发送器的简化框图。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 TXxREG 寄存器）取得数据。

23.1.1.1. 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：

- 将发送使能（TXEN）位设置为 1，以使能 EUSART 的发送器电路
- 将 EUSART 模式选择（SYNC）位设置为 0，以将 EUSART 配置为异步操作
- 将串行端口使能（SPEN）位设置为 1，以使能 EUSART 接口并允许自动选择 RxyPPS 的输出驱动器作为 TXx/CKx 输出

假定所有其他 EUSART 控制位均处于其默认状态。

如果 EUSART 与模拟外设共用 TXx/CKx 引脚，必须通过清零相应的 ANSEL 位禁止模拟 I/O 功能。

重要：TXEN 使能位置 1 且发送移位寄存器（TSR）空闲时，PIRx 寄存器中的 TXxIF 发送器中断标志将置 1。

23.1.1.2. 发送数据

向 TXxREG 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，TXxREG 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 TXxREG 中，直到前一个字符的停止位被发送。之后，在 TXxREG 中等待的字符在停止位发送后 1 个 T_{CY} 内被传送到 TSR 中。TXxREG 数据被传送到 TSR 后，启动位、数据位和停止位序列的发送立即开始。

23.1.1.3. 发送数据极性

可通过时钟/发送极性选择（SCKP）位来控制发送数据的极性。该位的默认状态为 0，选择高电平发送空闲和数据位。将 SCKP 位设置为 1 会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。SCKP 位仅在异步模式下控制发送数据的极性。在同步模式下，SCKP 位有不同的功能。有关详细信息，请参见时钟极性。

23.1.1.4. 发送中断标志

只要 EUSART 发送器被使能且 TXxREG 中没有等待发送的字符，PIRx 寄存器的 EUSART 发送中断标志（TXxIF）位就被置 1。换句话说，只有在 TSR 正在处理字符且 TXxREG 中还有一个排队等待发送的新字符时，TXxIF 位才被清零。写入 TXxREG 时并不立即清零 TXxIF 标志位。TXxIF 在执行写操作后的第 2 个指令周期才有效。写入 TXxREG 后立即查询 TXxIF 位将返回无效结果。TXxIF 位是只读位，不能用软件置 1 或清零。

将 PIEx 寄存器的 EUSART 发送中断允许（TXxIE）位置 1 可允许 TXxIF 中断。但是，只要 TXxREG 为空，不管 TXxIE 使能位的状态如何，TXxIF 标志位都将被置 1。

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

23.1.1.5. TSR 状态

发送移位寄存器状态（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 TXxREG 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关联，因此用户需要查询该位以确定 TSR 的状态。

重要：TSR 寄存器并未映射到数据存储器中，因此用户不能访问它。

23.1.1.6. 发送 9 位字符

EUSART 支持 9 位字符发送。当 9 位发送使能（TX9）位置 1 时，EUSART 将在发送每个字符时移出 9 位。TX9D 位是第 9 个数据位，也是最高有效数据位。发送 9 位数据时，TX9D 数据位必须先于低 8 位写入 TXxREG。写入 TXxREG 后，所有 9 个数据位将被立即传送到 TSR 寄存器。

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，请参见地址检测。

23.1.1.7. 异步发送设置

1. 初始化 SPxBRGH:SPxBRGL 寄存器对以及 BRGH 和 BRG16 位，以获得所需的波特率（见 EUSART 波特率发生器（BRG））。
2. 通过向 RxyPPS 寄存器写入适当的值来选择发送输出引脚。

3. 通过清零 SYNC 位并将 SPEN 位置 1，使能异步串行端口。
4. 如果需要发送 9 位数据，将 TX9 控制位置 1。接收器置于地址检测模式时，表示 8 个低数据位为地址。
5. 如果需要翻转发送，将 SCKP 位置 1。
6. 将 TXEN 控制位置 1 使能发送。这将导致 TXxIF 中断标志位置 1。
7. 如果需要中断，将 PIEx 寄存器的 TXxIE 中断允许位置 1。
8. 如果 INTCON 寄存器的 GIE 和 PEIE 位也置 1，则立即产生中断。
9. 如果选择发送 9 位数据，则会将第 9 位装入 TX9D 数据位。
10. 将 8 位数据装入 TXxREG 寄存器。这将启动发送。

图 23-3. 异步发送

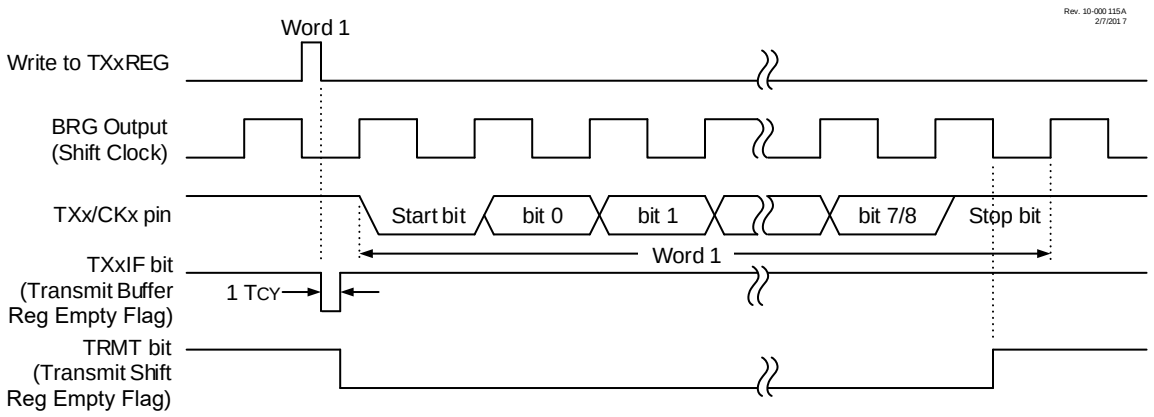
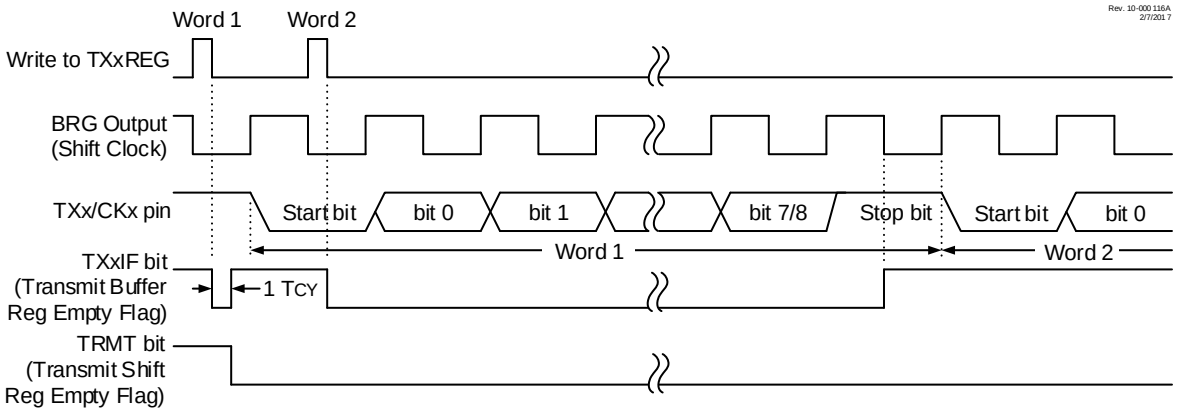


图 23-4. 异步发送（连续）



23.1.2. EUSART 异步接收器

异步模式通常用于 RS-232 系统中。图 23-2 给出了接收器的简化框图。数据在 RXx/DTx 引脚上接收并驱动数据恢复模块。数据恢复模块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。软件开始处理 EUSART 接收器前，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。FIFO 和 RSR 寄存器不能直接用软件访问。通过 RCxREG 寄存器访问接收数据。

23.1.2.1. 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：

- 将连续接收使能（CREN）位设置为 1，以使能 EUSART 的接收器电路
- 将 EUSART 模式选择（SYNC）位设置为 0，以将 EUSART 配置为异步操作
- 将串行端口使能（SPEN）位设置为 1，以使能 EUSART 接口

假定所有其他 EUSART 控制位均处于其默认状态。

用户必须通过设置 RXxPPS 寄存器来选择 RXx/DTx I/O 引脚，并且必须将相应的 TRIS 位置 1 以将相应引脚配置为输入。

重要：如果 RX/DT 功能在模拟引脚上，则必须清零相应的 ANSEL 位以使接收器正常工作。

23.1.2.2. 接收数据

接收器的数据恢复电路在第一个位的下降沿启动字符接收。第一个位也称启动位，始终为零。数据恢复电路计数半个位的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。它是停止位，总是为 1。如果数据恢复电路在停止位处采样到 0，则置 1 此字符的帧错误标志位，否则清零此字符的帧错误标志位。关于帧错误的更多信息，请参见接收帧错误。

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 PIRx 寄存器的 EUSART 接收中断标志（RCxIF）位被置 1。读取 RCxREG 寄存器时，FIFO 中顶部的字符被送出 FIFO。

重要：如果接收 FIFO 溢出，在溢出条件被清除前不会接收更多字符。更多信息，请参见接收帧错误。

23.1.2.3. 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，PIRx 寄存器的 EUSART 接收中断标志（RCxIF）位就会被置 1。RCxIF 中断标志位是只读位，不能用软件置 1 或清零。

将以下所有位置 1 可允许 RCxIF 中断：

- PIEx 寄存器的中断允许位 RCxIE
- INTCON 寄存器的外设中断允许位 PEIE
- INTCON 寄存器的全局中断允许位 GIE

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，RCxIF 中断标志位均会被置 1。

23.1.2.4. 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接收到停止位。通过帧错误（FERR）位可获取帧错误状态。FERR 位表示接收 FIFO 顶部未读字符的状态。因此，在读 RCxREG 寄存器之前必须先读 FERR 位。

FERR 位是只读位，只用于接收 FIFO 顶部的未读字符。帧错误（FERR = 1）不会阻止接收其他字符。此时不必将 FERR 位清零。从 FIFO 缓冲区读出下一个字符将前进至 FIFO 的下一个字符和下一个对应的帧错误。

将 SPEN 位清零可复位 EUSART，从而将 FERR 位强制清零。将 CREN 位清零不影响 FERR 位。帧错误本身不会产生中断。

重要：如果接收 FIFO 中的所有接收字符均有帧错误，反复读 RCxREG 寄存器也不会将 FERR 位清零。

23.1.2.5. 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，溢出错误（OERR）位置 1。FIFO 缓冲区中已有的字符可被读出，但错误被清除前不能再接收其他字符。将 CREN 位清零或通过将 SPEN 位清零复位 EUSART，可清除该错误。

23.1.2.6. 接收 9 位字符

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

23.1.2.7. 地址检测

当多个接收器共用同一条传输线时（如在 RS-485 系统中），有一个特殊的地址检测模式可供使用。将地址检测使能（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，从而将 RCxIF 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，用户软件可判断地址是否与自身匹配。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。

23.1.2.8. 异步接收设置

1. 初始化 SPxBRGH:SPxBRGL 寄存器对以及 BRGH 和 BRG16 位，以获得所需的波特率（见 EUSART 波特率发生器（BRG））。
2. 设置 RXxPPS 寄存器以选择 RXx/DTx 输入引脚。
3. 清零 RXx 引脚的 ANSEL 位（如适用）。
4. 通过将 SPEN 位置 1，使能串行端口。SYNC 位必须清零才能进行异步操作。
5. 如果需要中断，将 PIEx 寄存器的 RCxIE 位以及 INTCON 寄存器的 GIE 和 PEIE 位置 1。
6. 如果需要接收 9 位数据，将 RX9 位置 1。
7. 通过将 CREN 位置 1，使能接收。
8. 当字符从 RSR 被移入接收缓冲区时，RCxIF 中断标志位将被置 1。如果 RCxIE 中断允许位也置 1，则产生中断。
9. 读取 RCxSTA 寄存器取得错误标志和第 9 个数据位（9 位数据接收使能时）。
10. 读取 RCxREG 寄存器从接收缓冲区取得接收数据的低 8 位。
11. 如果发生溢出，则通过清零 CREN 接收器使能位来清零 OERR 标志。

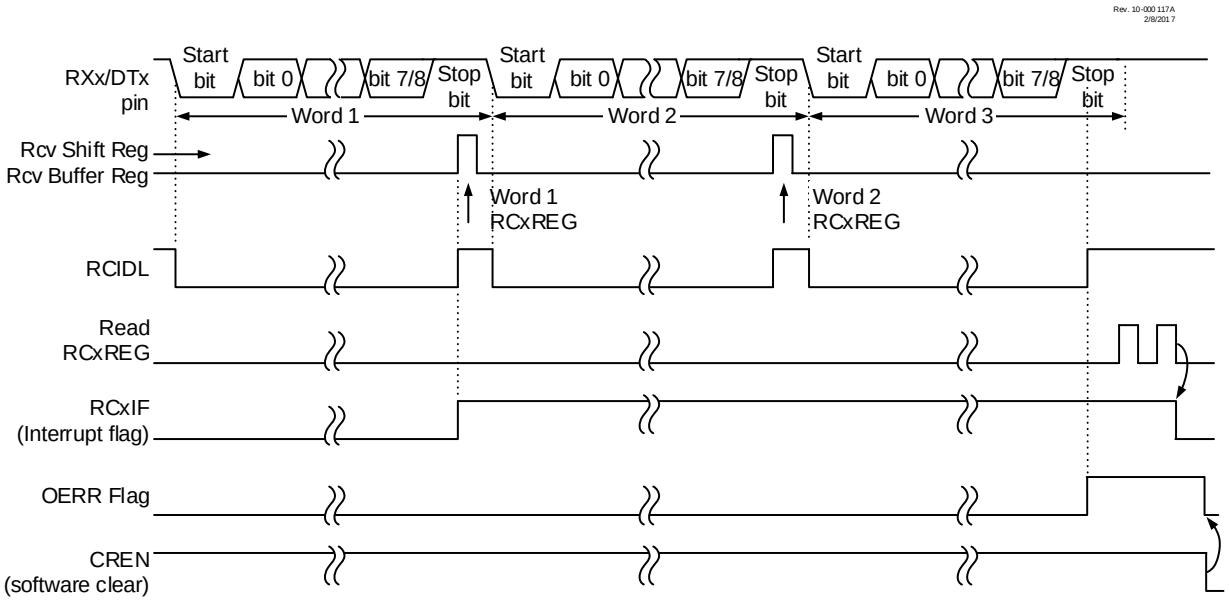
23.1.2.9. 9 位地址检测模式设置

此模式通常用于 RS-485 系统中。设置使能地址检测的异步接收的步骤如下：

1. 初始化 SPxBRGH:SPxBRGL 寄存器对以及 BRGH 和 BRG16 位，以获得所需的波特率（见 EUSART 波特率发生器（BRG））。
2. 设置 RXxPPS 寄存器以选择 RXx 输入引脚。

3. 清零 RXx 引脚的 ANSEL 位（如适用）。
4. 通过将 SPEN 位置 1，使能串行端口。SYNC 位必须清零才能进行异步操作。
5. 如果需要中断，将 PIEx 寄存器的 RCxIE 位以及 INTCON 寄存器的 GIE 和 PEIE 位置 1。
6. 通过将 RX9 位置 1，使能 9 位接收。
7. 通过将 ADDEN 位置 1，使能地址检测。
8. 通过将 CREN 位置 1，使能接收。
9. 当第 9 位置 1 的字符从 RSR 被移入接收缓冲区时，RCxIF 中断标志位将被置 1。如果 RCxIE 中断允许位也置 1，则产生中断。
10. 读取 RCxSTA 寄存器以获取错误标志。第 9 个数据位将始终置 1。
11. 读取 RCxREG 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。
12. 如果发生溢出，则通过清零 CREN 接收器使能位来清零 OERR 标志。
13. 如果器件被寻址，将 ADDEN 位清零以允许所有接收到的数据送入接收缓冲区并产生中断。

图 23-5. 异步接收



Note: This timing diagram shows three bytes appearing on the RXx input. The OERR flag is set because the RCxREG register is not read before the third word is received.

23.2. 异步操作的时钟精度

内部振荡器模块输出（INTOSC）在出厂时做了校准。但是，V_{DD} 或温度变化时，INTOSC 频率有可能漂移，进而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（首选）方法使用 OSCTUNE 寄存器调整 INTOSC 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。

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（见自动波特率检测）。通过调整波特率发生器来补偿外设时钟频率的逐渐变化时，可能无法足够精细地调节分辨率。

23.3. EUSART 波特率发生器（BRG）

波特率发生器（BRG）是一个 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 BRG16 位置 1 可选择 16 位模式。

SPxBRGH:SPxBRGL 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，波特率周期的倍频值由 BRGH 和 BRG16 位决定。在同步模式下，BRGH 位被忽略。

表 23-1 提供了确定波特率的公式。 公式 23-1 提供了确定波特率和波特率误差的计算示例。

下表给出了已计算好的各种异步模式下的典型波特率和误差值。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 模式用于在快速振荡器频率条件下实现低波特率。BRGH 位用于实现极高波特率。

将新值写入 SPxBRGH:SPxBRGL 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。

如果在有效接收操作期间更改了系统时钟，则可能导致接收错误或数据丢失。要避免这个问题，请检查接收空闲标志（RCIDL）位的状态以确保在更改系统时钟之前接收操作处于空闲状态。

公式 23-1. 计算波特率误差

针对工作在异步模式下，F_{OSC} 为 16 MHz，目标波特率为 9600，采用 8 位 BRG 的器件：

$$DesiredBaudrate = \frac{F_{OSC}}{64 \times (SPxBRG + 1)}$$

求解 SPxBRG：

$$SPxBRG = \frac{F_{OSC}}{64 \times DesiredBaudrate} - 1$$

$$SPxBRG = \frac{16000000}{64 \times 9600} - 1$$

$$SPxBRG = 25.042 \simeq 25$$

$$CalculatedBaudrate = \frac{16000000}{64 \times (25 + 1)}$$

$$CalculatedBaudrate = 9615$$

$$Error = \frac{CalculatedBaudrate - DesiredBaudrate}{DesiredBaudrate}$$

$$Error = \frac{9615 - 9600}{9600}$$

$$Error = 0.16\,\%$$

表 23-1. 波特率公式

配置位			BRG/EUSART 模式	波特率公式
SYNC	BRG16	BRGH		
0	0	0	8 位/异步	$F_{OSC}/[64\ (n+1)]$
0	0	1	8 位/异步	$F_{OSC}/[16\ (n+1)]$
0	1	0	16 位/异步	
0	1	1	16 位/异步	$F_{OSC}/[4\ (n+1)]$
1	0	x	8 位/同步	
1	1	x	16 位/同步	

注： x = 任意值，n = SPxBRGH:SPxBRGL 寄存器对的值。

表 23-2. 异步模式的波特率示例

波特率	SYNC = 0，BRGH = 0，BRG16 = 0											
	F_{OSC} = 32.000 MHz			F_{OSC} = 20.000 MHz			F_{OSC} = 18.432 MHz			F_{OSC} = 11.0592 MHz		
	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）
300	—	—	—	—	—	—	—	—	—	—	—	—
1200	—	—	—	1221	1.73	255	1200	0.00	239	1200	0.00	143
2400	2404	0.16	207	2404	0.16	129	2400	0.00	119	2400	0.00	71
9600	9615	0.16	51	9470	-1.36	32	9600	0.00	29	9600	0.00	17
10417	10417	0.00	47	10417	0.00	29	10286	-1.26	27	10165	-2.42	16
19.2k	19.23k	0.16	25	19.53k	1.73	15	19.20k	0.00	14	19.20k	0.00	8
57.6k	55.55k	-3.55	3	—	—	—	57.60k	0.00	7	57.60k	0.00	2
115.2k	—	—	—	—	—	—	—	—	—	—	—	—

波特率	SYNC = 0，BRGH = 0，BRG16 = 0											
	F_{OSC} = 8.000 MHz			F_{OSC} = 4.000 MHz			F_{OSC} = 3.6864 MHz			F_{OSC} = 1.000 MHz		
	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	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											
	F_{OSC} = 32.000 MHz			F_{OSC} = 20.000 MHz			F_{OSC} = 18.432 MHz			F_{OSC} = 11.0592 MHz		
	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）
300	—	—	—	—	—	—	—	—	—	—	—	—
1200	—	—	—	—	—	—	—	—	—	—	—	—
2400	—	—	—	—	—	—	—	—	—	—	—	—
9600	9615	0.16	207	9615	0.16	129	9600	0.00	119	9600	0.00	71
10417	10417	0.00	191	10417	0.00	119	10378	-0.37	110	10473	0.53	65
19.2k	19.23k	0.16	103	19.23k	0.16	64	19.20k	0.00	59	19.20k	0.00	35
57.6k	57.14k	-0.79	34	56.82k	-1.36	21	57.60k	0.00	19	57.60k	0.00	11
115.2k	117.64k	2.12	16	113.64k	-1.36	10	115.2k	0.00	9	115.2k	0.00	5

波特率	SYNC = 0，BRGH = 1，BRG16 = 0											
	F_{OSC} = 8.000 MHz			F_{OSC} = 4.000 MHz			F_{OSC} = 3.6864 MHz			F_{OSC} = 1.000 MHz		
	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	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											
	F_{OSC} = 32.000 MHz			F_{OSC} = 20.000 MHz			F_{OSC} = 18.432 MHz			F_{OSC} = 11.0592 MHz		
	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）
300	300.0	0.00	6666	300.0	-0.01	4166	300.0	0.00	3839	300.0	0.00	2303
1200	1200	-0.02	3332	1200	-0.03	1041	1200	0.00	959	1200	0.00	575
2400	2401	-0.04	832	2399	-0.03	520	2400	0.00	479	2400	0.00	287
9600	9615	0.16	207	9615	0.16	129	9600	0.00	119	9600	0.00	71
10417	10417	0.00	191	10417	0.00	119	10378	-0.37	110	10473	0.53	65
19.2k	19.23k	0.16	103	19.23k	0.16	64	19.20k	0.00	59	19.20k	0.00	35
57.6k	57.14k	-0.79	34	56.818	-1.36	21	57.60k	0.00	19	57.60k	0.00	11
115.2k	117.6k	2.12	16	113.636	-1.36	10	115.2k	0.00	9	115.2k	0.00	5

波特率	SYNC = 0，BRGH = 0，BRG16 = 1											
	F_{OSC} = 8.000 MHz			F_{OSC} = 4.000 MHz			F_{OSC} = 3.6864 MHz			F_{OSC} = 1.000 MHz		
	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（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	—	—	—

波特率	SYNC = 0，BRGH = 1，BRG16 = 1 或 SYNC = 1，BRG16 = 1											
	F_{OSC} = 32.000 MHz			F_{OSC} = 20.000 MHz			F_{OSC} = 18.432 MHz			F_{OSC} = 11.0592 MHz		
	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）
300	300.0	0.00	26666	300.0	0.00	16665	300.0	0.00	15359	300.0	0.00	9215
1200	1200	0.00	6666	1200	-0.01	4166	1200	0.00	3839	1200	0.00	2303
2400	2400	0.01	3332	2400	0.02	2082	2400	0.00	1919	2400	0.00	1151
9600	9604	0.04	832	9597	-0.03	520	9600	0.00	479	9600	0.00	287
10417	10417	0.00	767	10417	0.00	479	10425	0.08	441	10433	0.16	264
19.2k	19.18k	-0.08	416	19.23k	0.16	259	19.20k	0.00	239	19.20k	0.00	143
57.6k	57.55k	-0.08	138	57.47k	-0.22	86	57.60k	0.00	79	57.60k	0.00	47
115.2k	115.9k	0.64	68	116.3k	0.94	42	115.2k	0.00	39	115.2k	0.00	23

波特率	SYNC = 0，BRGH = 1，BRG16 = 1 或 SYNC = 1，BRG16 = 1											
	F_{OSC} = 8.000 MHz			F_{OSC} = 4.000 MHz			F_{OSC} = 3.6864 MHz			F_{OSC} = 1.000 MHz		
	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（十进制）	实际速率	误差百分比	SPBRG 值（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	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	—	—	—

23.3.1. 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（ABD）模式下，提供给 BRG 的时钟信号是反向的。BRG 并不为传入的 RX 信号提供时钟信号，而是由 RX 信号为 BRG 定时。波特率发生器用于为接收的 55h（ASCII “U”）定时，55h 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。

通过将自动波特率检测使能（ABDEN）位置 1，可以启动自动波特率校准序列。当发生 ABD 序列时，EUSART 状态机保持在空闲状态。在启动位之后，SPxBRG 寄存器使用 BRG 计数器时钟在接收信号的第一个上升沿开始递增计数（如图 23-6 所示）。在第 8 个位周期的末尾将在 RXx 引脚上出现第 5 个上升沿。此时，正确的 BRG 周期总数累计值被保存在 SPxBRGH:SPxBRGL 寄存器对中，ABDEN 位被自动清零而 RCxIF 中断标志被置 1。要清除 RCxIF 中断，需要读取 RCxREG 寄存器中的值。RCxREG 的内容可以丢弃。在不使用 SPxBRGH 寄存器的模式下进行校准时，用户可通过检查 SPxBRGH 寄存器的值是否为 00h 来验证 SPxBRGL 寄存器是否溢出。

BRG 自动波特率时钟由 BRG16 和 BRGH 位决定，如表 23-3 所示。在 ABD 期间，SPxBRGH 和 SPxBRGL 寄存器都被用作 16 位计数器，与 BRG16 位的设置无关。在校准波特率周期时，SPxBRGH 和 SPxBRGL 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时的平均位时间。

注：

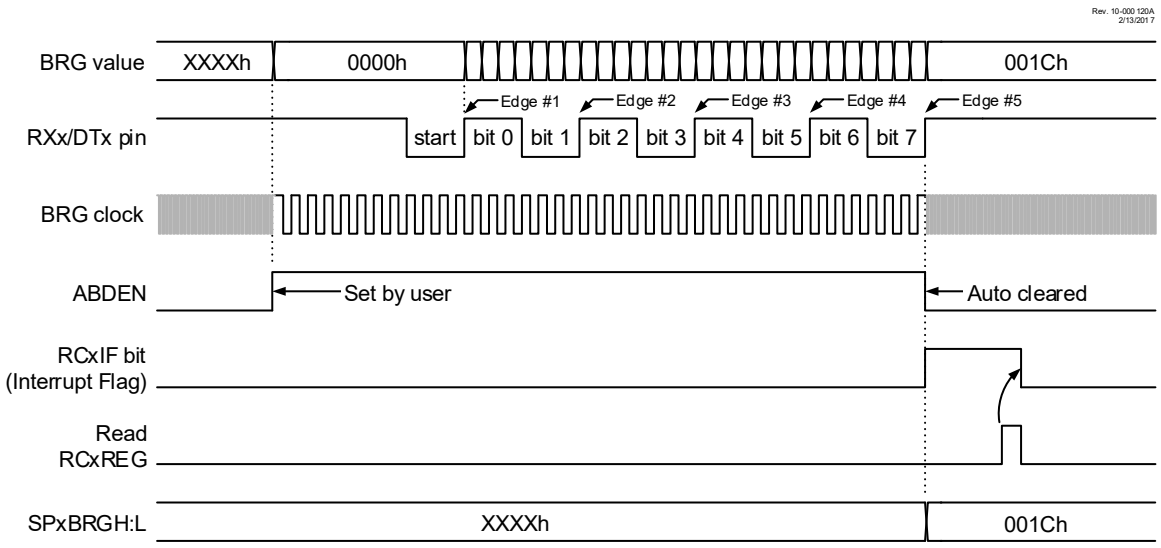
1. 如果唤醒使能（WUE）位和 ABDEN 位都置 1，自动波特率检测将从间隔字符之后的字节开始（见接收到间隔字符时自动唤醒）。
2. 需要由用户来判断输入字符的波特率是否处于所选 BRG 时钟源范围内。可能无法实现某些振荡器频率和 EUSART 波特率组合。
3. 在自动波特率检测过程中，自动波特率计数器从 1 开始计数。自动波特率序列完成后，为了得到最准确的结果，应从 SPxBRGH:SPxBRGL 寄存器对的值中减去 1。

表 23-3. BRG 计数器时钟速率

BRG16	BRGH	BRG 基本时钟	BRG ABD 时钟
1	1	F_{OSC}/4	F_{OSC}/32
1	0	F_{OSC}/16	F_{OSC}/128
0	1	F_{OSC}/16	F_{OSC}/128
0	0	F_{OSC}/64	F_{OSC}/512

注：在 ABD 序列期间，SPxBRGL 和 SPxBRGH 寄存器都被用作 16 位计数器，与 BRG16 的设置无关。

图 23-6. 自动波特率校准



23.3.2. 自动波特率溢出

在自动波特率检测过程中，如果在 RXx 引脚上检测到第 5 个上升沿之前波特率计数器溢出，则自动波特率检测溢出（ABDOVF）位将被置 1。ABDOVF 位指示计数器已超出 SPxBRGH:SPxBRGL 寄存器对的 16 位所能允许的最大计数值。在 ABDOVF 位置 1 后，计数器将继续计数，直到在 RXx 引脚上检测到第 5 个上升沿为止。检测到第 5 个 RX 边沿时，硬件会将 RCxIF 中断标志位置 1 并将 ABDEN 位清零。随后通过读 RCxREG 寄存器，RCxIF 中断标志位也将清零。可以通过软件直接清零 ABDOVF 位。

要在 RCxIF 中断标志置 1 之前终止自动波特率检测过程，可依次将 ABDEN 位和 ABDOVF 位清零。如果没有先将 ABDEN 位清零，则 ABDOVF 位将保持置 1 状态。

23.3.3. 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 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 线上电平由高至低的跳变。这与同步间隔字符或 LIN 协议唤醒信号字符的起始条件一致。

EUSART 模块产生的 RCxIF 中断与唤醒事件同时发生。在正常 CPU 工作模式下，与 Q 时钟同步产生中断（如图 23-7 所示）；在休眠模式下，与 Q 时钟异步产生中断（如图 23-8 所示）。通过读 RCxREG 寄存器可清除中断条件。

RX 线在间隔字符末尾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。

23.3.3.1. 特殊注意事项

间隔字符

为了避免唤醒事件期间的字符错误或字符分割，唤醒字符必须全部为零。

唤醒被使能时，其工作状况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错误或溢出错误。

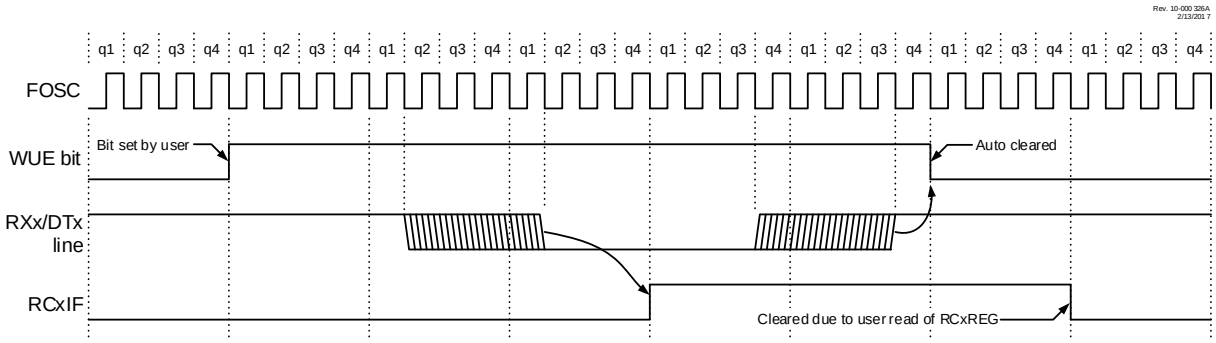
因此，发送的首字符必须为全 0。这必须持续 10 个或更长的位时间，对于 LIN 总线，建议持续 13 个位时间，而对于标准 RS-232 器件，可为任意个位时间。

WUE 位

唤醒事件会通过将 RCxIF 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 RCxREG 寄存器并丢弃其内容将中断条件清除。

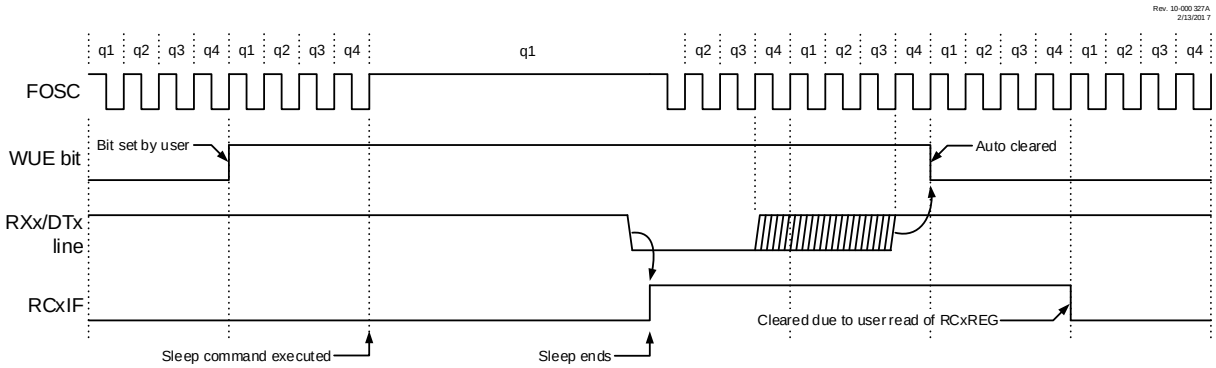
要确保不丢失实际数据，应在将 WUE 位置 1 前检查 RCIDL 位，验证没有接收操作在进行。如果未发生接收操作，可在进入休眠模式前将 WUE 位置 1。

图 23-7. 正常工作时的自动唤醒（WUE）位时序



Note: The EUSART remains in Idle while the WUE bit is set.

图 23-8. 休眠时的自动唤醒（WUE）位时序



Note: The EUSART remains in Idle while the WUE bit is set.

23.3.4. 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。间隔字符包括一个启动位以及随后的 12 个 0 位和一个停止位。

要发送间隔字符，应将发送间隔字符（SENDB）和发送使能（TXEN）位置 1。随后对 TXxREG 执行写操作可启动间隔字符发送。写入 TXxREG 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。

发送移位寄存器状态（TRMT）位表明发送操作何时处于有效或空闲状态，这与正常发送时相同。有关详细信息，请参见图 23-9。

23.3.4.1. 断开字符和同步字符发送序列

以下序列将启动报文帧头，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 配置为需要的模式。
2. 将 TXEN 和 SENDB 位置 1，以使能间隔字符序列。
3. 将无效字符装入 TXxREG，启动发送（该值会被忽略）。
4. 将 55h 写入 TXxREG，以将同步字符装入发送 FIFO 缓冲区。
5. 发送了间隔字符之后，由硬件将 SENDB 位复位，然后发送同步字符。

当 TXxREG 为空（由 TXxIF 位指示）时，下一个数据字节会写入 TXxREG。

23.3.5. 接收间隔字符

EUSART 模块可采用两种方式接收间隔字符。

第一种检测间隔字符的方法采用帧错误（FERR）位和 RCxREG 所指示的接收数据。假定波特率发生器已初始化为期望的波特率。

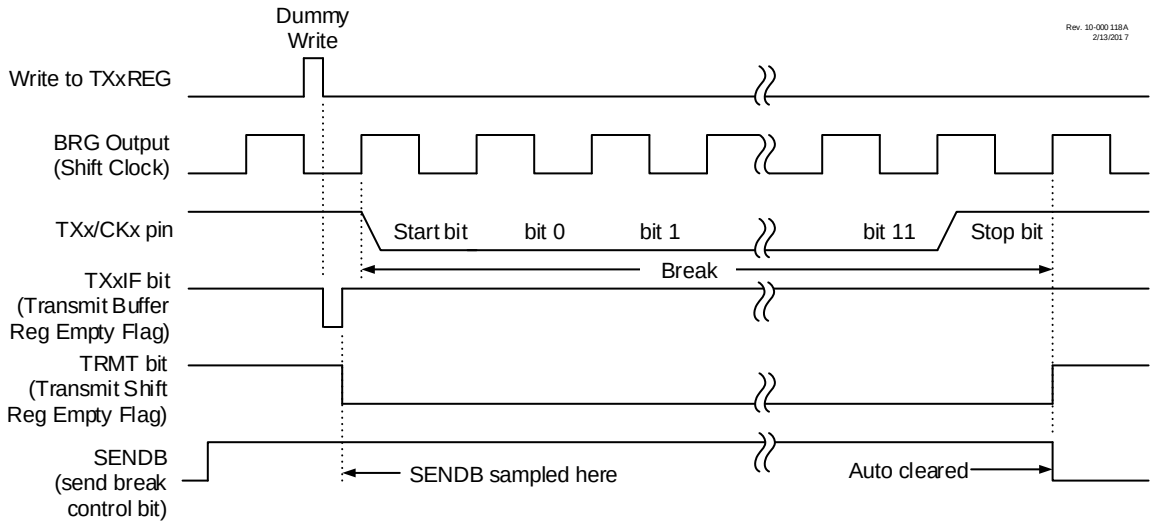
当以下三个条件全部为真时，表示接收到间隔字符：

- RCxIF 位置 1
- FERR 位置 1
- RCxREG = 00h

第二种方法使用接收到间隔字符时自动唤醒中描述的自动唤醒功能。通过使能此功能，EUSART 将采样 RX/DT 引脚上的下两次电平跳变，产生一次 RCxIF 中断，接收下一个数据字节，之后再产生一次中断。

注意在间隔字符后，用户通常想要使能自动波特率检测功能。对于这两种方法，用户都可以在 EUSART 进入休眠模式之前，将 ABDEN 位置 1。

图 23-9. 发送间隔字符序列



23.4. 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，并为系统中的所有器件提供时钟。从器件可使用主器件时钟，从而无需内部时钟发生电路。

同步模式下有两条信号线：一根双向数据线（DT）和一根时钟线（CK）。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，所以同步操作只能是半双工的。半双工指主从器件都能够接收和发送数据，但不能同时进行。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。

23.4.1. 同步主模式

使用以下位将 EUSART 配置为同步主操作：

- 将 SYNC 位设置为 1，以将 EUSART 配置为同步操作
- 将时钟源选择（CSRC）位设置为 1，以将 EUSART 配置为主器件
- 将单字符接收使能（SREN）位设置为 0 可进行发送；将 SREN 设置为 1 可进行接收（建议设置为接收 1 字节）
- 将连续字符接收使能（CREN）位设置为 0 可进行发送；将 CREN 设置为 1 可进行连续接收
- 将 SPEN 位设置为 1，以使能 EUSART 接口

重要：通过将 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。

23.4.1.1. 主时钟

同步数据传输使用独立于数据线但与数据线同步的时钟线。配置为主器件的器件在 TX/CK 线上发送时钟信号。EUSART 配置为同步发送或接收操作时，自动使能 TXx/CKx 引脚的输出驱动器。串行数据位在每个时钟的前沿改变，以确保其在时钟的后沿有效。每个数据位都会产生一个时钟周期。数据位有多少，就会产生多少个时钟周期。

23.4.1.2. 时钟极性

为了与 Microwire 兼容，提供了时钟极性选项。时钟极性通过时钟/发送极性选择（SCKP）位进行选择。将 SCKP 位置 1 时，可将时钟空闲状态设置为高电平。当 SCKP 位置 1 时，数据在每个时钟的下降沿改变。将 SCKP 位清零会将空闲状态设置为低电平。当 SCKP 位清零时，数据在每个时钟的上升沿改变。

23.4.1.3. 同步主发送

数据在器件的 RXx/DTx 引脚上发出。EUSART 配置为同步主发送操作时，RXx/DTx 和 TXx/CKx 引脚的输出驱动器被自动使能。

向 TXxREG 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 TXxREG 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，TXxREG 中的数据就立即被传送到 TSR。字符发送在数据从 TXxREG 送入 TSR 后立即开始。

每个数据位在主时钟的前沿改变，并在下一个时钟的前沿到来前保持有效。

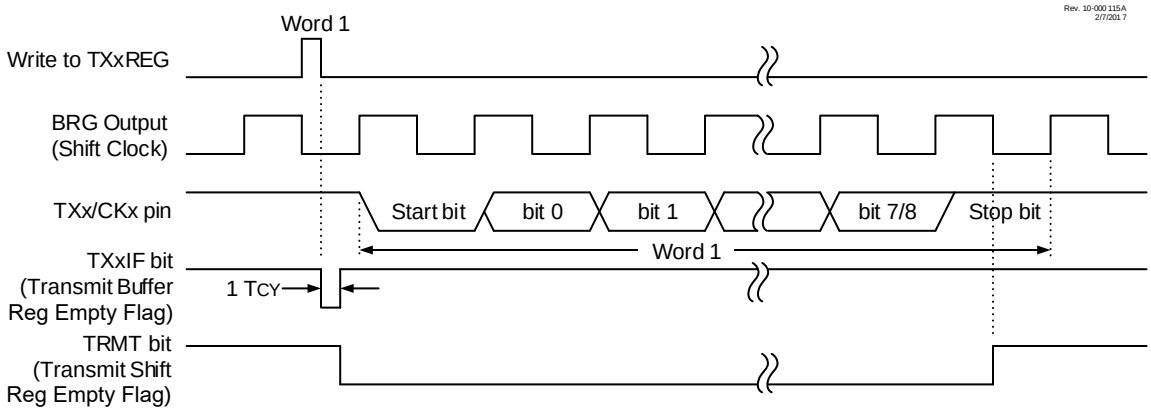
注：TSR 寄存器并未映射到数据存储器中，因此用户不能访问它。

23.4.1.4. 同步主发送设置

1. 初始化 SPxBRGH;SPxBRGL 寄存器对以及 BRG16 位，以获得所需的波特率（见 EUSART 波特率发生器（BRG））。
2. 通过向 RxyPPS 寄存器和 RXxPPS 寄存器写入适当的值来选择发送输出引脚。两种选择可以使能同一引脚。
3. 通过向 RxyPPS 寄存器和 TXxPPS 寄存器写入适当的值来选择时钟输出引脚。两种选择可以使能同一引脚。
4. 通过将 SYNC、SPEN 和 CSRC 位置 1，使能同步主串行端口。
5. 通过将 SREN 和 CREN 位清零，禁止接收模式。
6. 通过将 TXEN 位置 1，使能发送模式。
7. 如果需要发送 9 位数据，将 TX9 位置 1。

8. 如果需要中断，将 PIEx 寄存器的 TXxIE 位以及 INTCON 寄存器的 GIE 和 PEIE 位置 1。
9. 如果选择发送 9 位数据，将第 9 位数据装入 TX9D 位。
10. 将数据装入 TXxREG 寄存器，启动发送。

图 23-10. 同步发送



23.4.1.5. 同步主接收

数据在 RXx/DTx 引脚上接收。将 EUSART 配置为同步主接收操作时，自动禁止 RXx/DTx 引脚输出驱动器。

在同步模式下，可通过将单字符接收使能（SREN）位或连续接收使能（CREN）位置 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。一个字符接收完成时 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 在接收一个字符的过程中被清零，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 时钟引脚的后沿对 RXx/DTx 引脚上的数据进行采样，并移入接收移位寄存器（RSR）。在完整的字符被接收到 RSR 时，RCxIF 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 RCxREG 中。只要接收 FIFO 中有未读字符，RCxIF 位就保持置 1。

注：如果 RX/DT 功能在模拟引脚上，则必须清零相应的 ANSEL 位以使接收器正常工作。

23.4.1.6. 从时钟

同步数据传输使用独立于数据线但与数据线同步的时钟线。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，自动禁止 TXx/CKx 引脚输出驱动器。串行数据位在每个时钟的前沿改变，以确保其在时钟的后沿有效。每个时钟周期传送一个数据位。数据位有多少，就会产生多少个接收时钟周期。

重要：如果器件配置为从器件，且 TX/CK 功能在模拟引脚上，则必须清零相应 ANSEL 位。

23.4.1.7. 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，溢出错误（OERR）位置 1。FIFO 缓冲区中已有的字符可被读出，但错误被清除前不能再接收其他字符。将 CREN 位清零或通过将 SPEN 位清零复位 EUSART，可清除该错误。

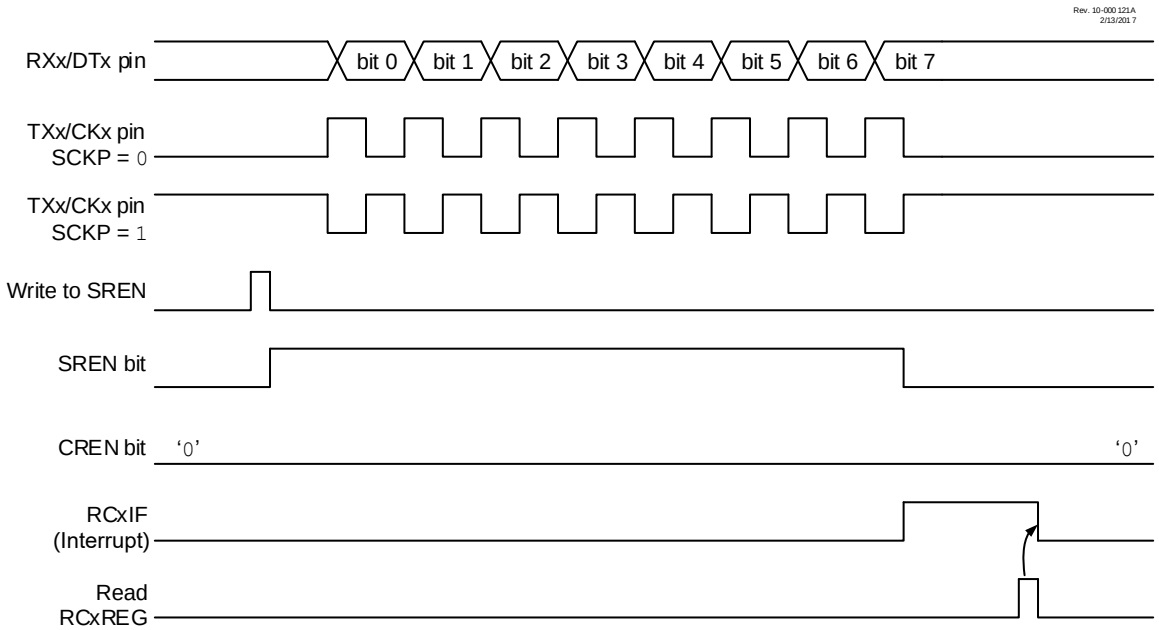
23.4.1.8. 接收 9 位字符

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

23.4.1.9. 同步主接收设置

1. 初始化 SPxBRGH:SPxBRGL 寄存器对并按需要将 BRG16 位置 1 或清零，获得所需的波特率。
2. 通过向 RxyPPS 和 RXxPPS 寄存器写入适当的值来选择接收输入引脚。两种选择可以使能同一引脚。
3. 通过向 RxyPPS 和 TXxPPS 寄存器写入适当的值来选择时钟输出引脚。两种选择可以使能同一引脚。
4. 清零 RXx 引脚的 ANSEL 位（如适用）。
5. 通过将 SYNC、SPEN 和 CSRC 位置 1，使能同步主串行端口。
6. 确保 CREN 和 SREN 位清零。
7. 如果需要中断，将 PIEx 寄存器的 RCxIE 位以及 INTCON 寄存器的 GIE 和 PEIE 位置 1。
8. 如果需要接收 9 位数据，将 RX9 位置 1。
9. 将 SREN 位置 1 启动接收，或将 CREN 位置 1 启动连续接收。
10. 字符接收完成时，RCxIF 中断标志位将被置 1。如果 RCxIE 允许位已置 1，则产生中断。
11. 读取 RCxSTA 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。
12. 读 RCxREG 寄存器来读取接收到的 8 位数据。
13. 如果发生了溢出错误，通过清零 CREN 位或清零 SPEN 位（该位清零会将 EUSART 复位），可以清除错误。

图 23-11. 同步接收（主模式，通过 SREN 控制）



23.4.2. 同步从模式

使用以下位将 EUSART 配置为同步从操作：

- SYNC = 1（将 EUSART 配置为同步操作）
- CSRC = 0（将 EUSART 配置为从操作）

- SREN = 0（用于发送）；SREN = 1（用于单字节接收）
- CREN = 0（用于发送）；CREN = 1（建议设置为连续接收）
- SPEN = 1（使能 EUSART）

重要：通过将 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。

23.4.2.1. EUSART 同步从发送

除休眠模式外，同步主发送模式和同步从发送模式的工作原理是相同的（见同步主发送）。

如果向 TXxREG 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. 第二个字保留在 TXxREG 寄存器中。
3. TXxIF 位不会置 1。
4. 第一个字符移出 TSR 后，TXxREG 寄存器将第二个字符传送到 TSR，此时 TXxIF 位置 1。
5. 如果 PEIE 和 TXxIE 位均置 1，则中断会将器件从休眠状态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。

23.4.2.2. 同步从发送设置

1. 将 SYNC 和 SPEN 位置 1 并清零 CSRC 位。
2. 通过向 RxyPPS 寄存器和 RXxPPS 寄存器写入适当的值来选择发送输出引脚。两种选择可以使能同一引脚。
3. 通过向 TXxPPS 寄存器写入适当的值来选择时钟输入引脚。
4. 清零 CKx 引脚的 ANSEL 位（如适用）。
5. 将 CREN 和 SREN 位清零。
6. 如果需要中断，将 PIEx 寄存器的 TXxIE 位以及 INTCON 寄存器的 GIE 和 PEIE 位置 1。
7. 如果需要发送 9 位数据，将 TX9 位置 1。
8. 将 TXEN 位置 1 使能发送。
9. 如果选择发送 9 位数据，则将最高有效位装入 TX9D 位。
10. 将低 8 位写入 TXxREG 寄存器，准备发送。将发送该字以响应 CKx 引脚上的主时钟。

23.4.2.3. EUSART 同步从接收

除下列各项外，同步主接收模式和从接收模式的工作是相同的（见同步主接收）：

- 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

在进入休眠模式之前，通过将 CREN 位置 1 可在休眠模式下接收字符。接收到数据字后，RSR 寄存器会将数据传送到 RCxREG 寄存器。如果 RCxIE 中断允许位置 1，产生的中断会将器件从休眠模式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。

23.4.2.4. 同步从接收设置

1. 将 SYNC 和 SPEN 位置 1 并清零 CSRC 位。
2. 通过向 RXxPPS 寄存器写入适当的值来选择接收输入引脚。

3. 通过向 TXxPPS 寄存器写入适当的值来选择时钟输入引脚。
4. 清零 TXx/CKx 和 RXx/DTx 引脚的 ANSEL 位（如适用）。
5. 如果需要中断，将 PIEx 寄存器的 RCxIE 位以及 INTCON 寄存器的 GIE 和 PEIE 位置 1。
6. 如果需要接收 9 位数据，将 RX9 位置 1。
7. 将 CREN 位置 1，以使能接收。
8. 接收完成时 RCxIF 位将被置 1。如果 RCxIE 位已置 1，则产生中断。
9. 如果使能了 9 位模式，从 RX9D 位取出最高有效位。
10. 读取 RCxREG 寄存器，从接收 FIFO 取出低 8 位。
11. 如果发生了溢出错误，通过清零 CREN 位或清零 SPEN 位（该位清零会将 EUSART 复位），可以清除错误。

23.5. EUSART 休眠期间的工作

EUSART 仅在同步从模式下才会在休眠期间保持活动状态。所有其他模式都需要系统时钟，因此在休眠模式下无法产生使发送或接收移位寄存器工作必需的信号。

同步从模式使用外部生成的时钟来运行发送移位寄存器和接收移位寄存器。

23.5.1. 休眠期间的同步接收

要在休眠期间进行接收，在进入休眠模式前必须满足以下所有条件：

- 必须将 RCxSTA 和 TXxSTA 控制寄存器配置为同步从接收（见同步从接收设置）。
- 如果需要中断，将 PIEx 寄存器的 RCxIE 位以及 INTCON 寄存器的 GIE 和 PEIE 位置 1。
- 必须通过读 RCxREG 清零 RCxIF 中断标志位，以卸载接收缓冲区中等待处理的任何字符。

进入休眠模式时，器件将准备好分别接受 RXx/DTx 和 TXx/CKx 引脚上的数据和时钟。当数据字已完全由外部器件随时钟输入时，会将 PIRx 寄存器的 RCxIF 中断标志位置 1，从而将处理器从休眠中唤醒。

从休眠中唤醒后，将执行 SLEEP 指令之后的指令。如果 INTCON 寄存器的全局中断允许（GIE）位也置 1，则调用中断服务程序（ISR）。

23.5.2. 休眠期间的同步发送

要在休眠期间进行发送，在进入休眠模式前必须满足以下条件：

- 必须将 RCxSTA 和 TXxSTA 控制寄存器配置为同步从发送（见同步从发送设置）。
- 必须通过将输出数据写入 TXxREG 进而填充 TSR 和发送缓冲区来清零 TXxIF 中断标志位。
- 必须向 PIEx 寄存器的 TXxIE 中断允许位和 INTCON 寄存器的 PEIE 位写入 1。
- 如果需要中断，将 INTCON 寄存器的 GIE 位置 1。

进入休眠模式时，器件将在 TXx/CKx 引脚上接收时钟信号，在 RXx/DTx 引脚上发送数据。TSR 寄存器中的数据字完全由外部器件随着时钟移出后，TXxREG 中等待的字节将传输到 TSR，TXxIF 标志位置 1，从而将处理器从休眠中唤醒。此时，TXxREG 可用于接受另一个要发送的字符。写入 TXxREG 将清零 TXxIF 标志。

从休眠中唤醒后，将执行 SLEEP 指令之后的指令。如果全局中断允许（GIE）位也置 1，则调用中断服务程序（ISR）。

23.6. 寄存器定义：EUSART 控制

23.6.1. TXxSTA

名称： TXxSTA
偏移量： 0x011E

发送状态和控制寄存器

位	7	6	5	4	3	2	1	0
	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
访问	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
复位	0	0	0	0	0	0	1	0

Bit 7 – CSRC 时钟源选择

值	条件	说明
1	SYNC = 1	主模式（时钟由内部 BRG 产生）
0	SYNC = 1	从模式（时钟来自外部时钟源）
X	SYNC = 0	无关

Bit 6 – TX9 9 位发送使能

值	说明
1	选择 9 位发送
0	选择 8 位发送

Bit 5 – TXEN 发送使能
使能发送器(1)

值	说明
1	使能发送
0	禁止发送

Bit 4 – SYNC EUSART 模式选择

值	说明
1	同步模式
0	异步模式

Bit 3 – SENDB 发送间隔字符

值	条件	说明
1	SYNC = 0	在下一次发送时发送同步间隔字符（完成时由硬件清零）
0	SYNC = 0	禁止或已完成同步间隔字符的发送
X	SYNC = 1	无关

Bit 2 – BRGH 高波特率选择

值	条件	说明
1	SYNC = 0	高速，如果 BRG16 = 1，则波特率为 baudclk/4；否则为 baudclk/16
0	SYNC = 0	低速
X	SYNC = 1	无关

Bit 1 – TRMT 发送移位寄存器（TSR）状态

值	说明
1	TSR 为空

值	说明
0	TSR 非空

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 和 CREN 位的优先级高于 TXEN。

23.6.2. RCxSTA

名称： RCxSTA
偏移量： 0x011D

接收状态和控制寄存器

位	7	6	5	4	3	2	1	0
	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
访问	R/W	R/W	R/W/HC	R/W	R/W	R	R/HC	R/HC
复位	0	0	0	0	0	0	0	0

Bit 7 – SPEN 串行端口使能

值	说明
1	使能串行端口
0	禁止串行端口（保持在复位状态）

Bit 6 – RX9 9 位接收使能

值	说明
1	选择 9 位接收
0	选择 8 位接收

Bit 5 – SREN 单字节接收使能
控制接收。接收完成后，该位将由硬件清零。

值	条件	说明
1	SYNC = 1 且 CSRC = 1	开始单字节接收
0	SYNC = 1 且 CSRC = 1	单字节接收已完成
X	SYNC = 0 或 CSRC = 0	无关

Bit 4 – CREN 连续接收使能

值	条件	说明
1	SYNC = 1	使能连续接收，直到使能位 CREN 清零（CREN 的优先级高于 SREN）
0	SYNC = 1	禁止连续接收
1	SYNC = 0	使能接收器
0	SYNC = 0	禁止接收器

Bit 3 – ADDEN 地址检测使能

值	条件	说明
1	SYNC = 0 且 RX9 = 1	仅当接收的第 9 个位置 1 时才装入接收缓冲区并产生中断
0	SYNC = 0 且 RX9 = 1	接收所有字节且始终产生中断。第 9 个位可用作奇偶校验位
X	RX9 = 0 或 SYNC = 1	无关

Bit 2 – FERR 帧错误

值	说明
1	RCxREG 中的未读字节有帧错误
0	RCxREG 中的未读字节没有帧错误

Bit 1 – OERR 溢出错误

值	说明
1	溢出错误（清零 SPEN 或 CREN 位可清除该错误）
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，具体由用户固件确定。

23.6.3. BAUDxCON

名称： BAUDxCON
偏移量： 0x011F

波特率控制寄存器

位	7	6	5	4	3	2	1	0
	ABDOVF	RCIDL		SCKP	BRG16		WUE	ABDEN
访问	R	R		R/W	R/W		R/W	R/W
复位	0	1		0	0		0	0

Bit 7 – ABDOVF 自动波特率检测溢出

值	条件	说明
1	SYNC = 0	自动波特率定时器溢出
0	SYNC = 0	自动波特率定时器未溢出
X	SYNC = 1	无关

Bit 6 – RCIDL 接收空闲标志

值	条件	说明
1	SYNC = 0	接收器空闲
0	SYNC = 0	已接收到启动位且接收器正在接收
X	SYNC = 1	无关

Bit 4 – SCKP 时钟/发送极性选择

值	条件	说明
1	SYNC = 0	发送（TX）的空闲状态为低电平（发送数据反相）
0	SYNC = 0	发送（TX）的空闲状态为高电平（发送数据同相）
1	SYNC = 1	在时钟的上升沿传送数据
0	SYNC = 1	在时钟的下降沿传送数据

Bit 3 – BRG16 16 位波特率发生器选择

值	说明
1	使用 16 位波特率发生器
0	使用 8 位波特率发生器

Bit 1 – WUE 唤醒使能

值	条件	说明
1	SYNC = 0	接收器正在等待下降沿。在下降沿将不接收任何字符，并且 RCxIF 标志将置 1。WUE 将在 RCxIF 置 1 后自动清零。
0	SYNC = 0	接收器正常工作
X	SYNC = 1	无关

Bit 0 – ABDEN 自动波特率检测使能

值	条件	说明
1	SYNC = 0	使能自动波特率检测模式（自动波特率检测完成后清零）
0	SYNC = 0	已完成自动波特率检测或禁止该模式
X	SYNC = 1	无关

23.6.4. RCxREG

名称： RCxREG
偏移量： 0x0119

接收数据寄存器

位	7	6	5	4	3	2	1	0
	RCREG[7:0]							
访问	R	R	R	R	R	R	R	R
复位	0	0	0	0	0	0	0	0

Bit 7:0 – RCREG[7:0] 接收数据

23.6.5. TXxREG

名称： TXxREG
偏移量： 0x011A

发送数据寄存器

位	7	6	5	4	3	2	1	0
	TXREG[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7:0 – TXREG[7:0] 发送数据

23.6.6. SPxBRG

名称： SPxBRG
偏移量： 0x011B

EUSART 波特率发生器

位	15	14	13	12	11	10	9	8
	SPBRG[15:8]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

位	7	6	5	4	3	2	1	0
	SPBRG[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 15:0 – SPBRG[15:0] 波特率寄存器

注：该多字节寄存器中的各个字节可使用以下寄存器名称进行访问：

- SPxBRGH：访问高字节 SPBRG[15:8]
- SPxBRGL：访问低字节 SPBRG[7:0]

23.7. 寄存器汇总——EUSART

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0118	保留									
0x0119	RC1REG	7:0	RCREG[7:0]							
0x011A	TX1REG	7:0	TXREG[7:0]							
0x011B	SP1BRG	7:0	SPBRG[7:0]							
		15:8	SPBRG[15:8]							
0x011D	RC1STA	7:0	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
0x011E	TX1STA	7:0	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
0x011F	BAUD1CON	7:0	ABDOVF	RCIDL		SCKP	BRG16		WUE	ABDEN

24. ADC——模数转换器

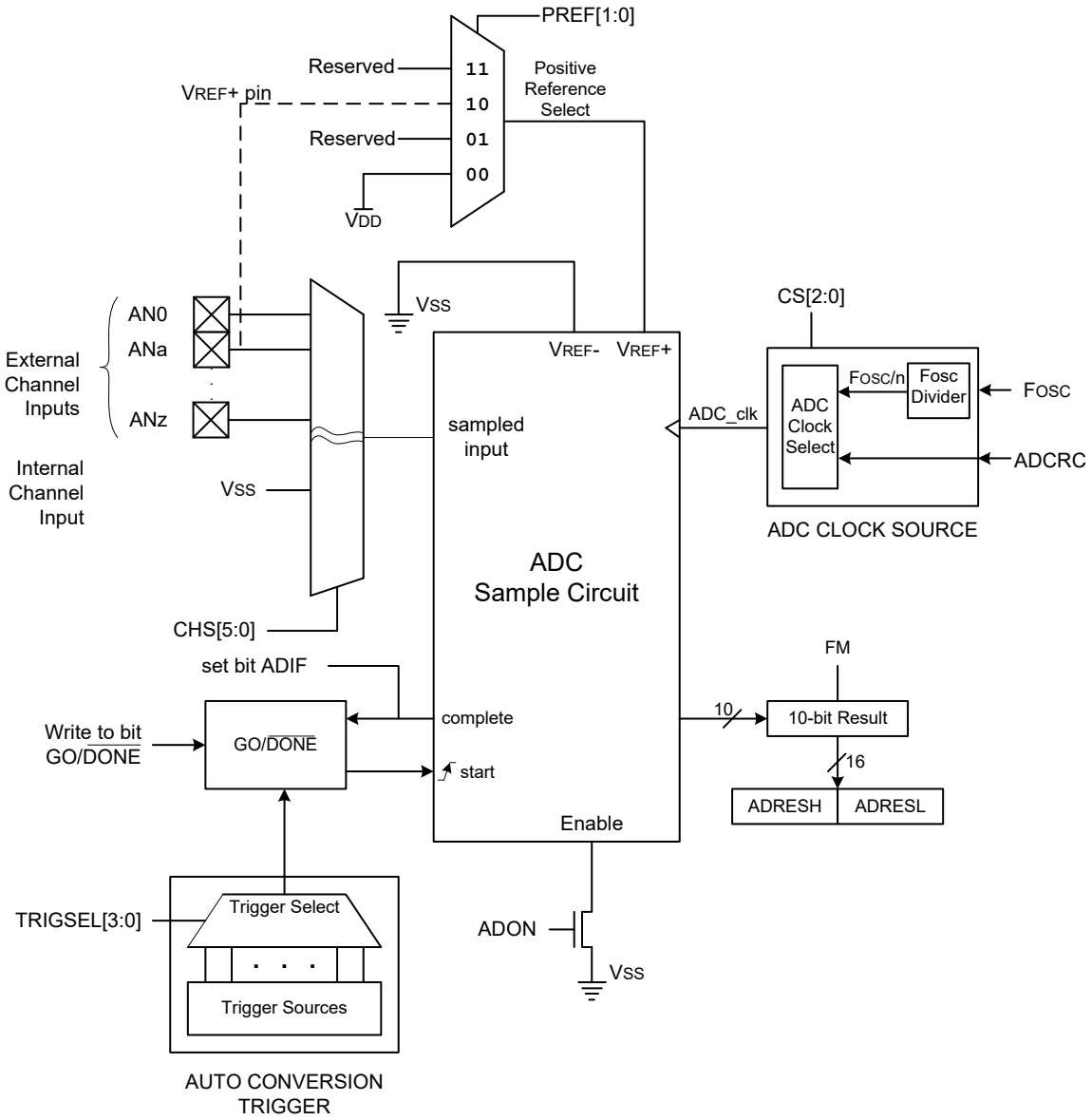
模数转换器（ADC）可将模拟输入信号转换为该信号的 10 位二进制形式。该器件使用模拟输入，这些输入通过多路开关连接到同一个采样保持电路。采样保持电路的输出与转换器的输入相连接。转换器通过逐次逼近法产生 10 位二进制结果，并将结果存储在 ADC 结果寄存器（ADRESH:ADRESL 寄存器对）中。

ADC 参考电压可通过软件选择为内部产生的参考电压或由外部提供。

ADC 可在转换完成时产生中断。此中断可用于将器件从休眠状态中唤醒。

图 24-1 给出了 ADC 的框图。

图 24-1. ADC 框图



24.1. ADC 配置

配置和使用 ADC 时必须注意以下事项：

- 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

24.1.1. 端口配置

无论 ANSEL 位是否置 1，ADC 都会对引脚电压进行转换。转换模拟信号时，可通过设置相关的 TRIS 和 ANSEL 位将 I/O 引脚配置为模拟。更多信息，请参见“**I/O 端口**”一节。

重要：在任何定义为数字输入的引脚上施加模拟电压可能导致输入缓冲器的电流过大。

24.1.2. 通道选择

模拟通道选择（CHS）位决定连接采样保持电路以进行转换的通道。切换通道时，建议在开始下一次转换前增加一段采集延时。更多信息，请参见“ADC 工作原理”一节。

重要：为减少测量误差，建议在切换 ADC 通道时通过以下方式使采样保持电容放电：在连接到 V_{SS} 的通道上启动转换，然后在采集时间结束后终止转换。如果 ADC 没有专用的 V_{SS} 输入通道，则可以将一个空闲的输入通道连接到 V_{SS} 来代替专用输入通道。

24.1.3. ADC 参考电压

ADC 正参考电压选择（PREF）位控制正参考电压。有关可用正电压源的列表，请参见 ADC 控制存器 1（ADCON1）。

24.1.4. 转换时钟

转换时钟源通过 ADC 转换时钟选择（CS）位选择。可用时钟源包含系统时钟（F_{OSC}）的多种衍生时钟以及称为 ADCRC 的专用内部固定频率时钟。

完成一个位的转换所需的时间定义为 T_{AD}。有关 ADC 转换的完整时序详细信息，请参见图 24-2。

为正确转换，必须满足相应的 T_{AD} 规范。有关更多详细信息，请参见“**电气规范**”一节中的 *ADC 时序规范*表。 表 24-1 给出了适当的 ADC 时钟选择的示例。

重要：

- 除 ADCRC 时钟源外，系统时钟频率的任何变化都会改变 ADC 时钟频率，这会影响 ADC 结果。
- ADC 的内部控制逻辑基于 CS 位选择的时钟工作。当 CS 位选择 ADCRC 时，如果将 ADC 控制位置 1，则可能出现意外延时。

表 24-1. 不同器件频率（F_{OSC}）下的 ADC 时钟周期（T_{AD}）

ADC 时钟源	CS[2:0]	不同器件频率（F_{OSC}）下的 ADC 时钟周期（T_{AD}）					
		32 MHz	20 MHz	16 MHz	8 MHz	4 MHz	1 MHz
$F_{OSC}/2$	'b000	62.5 ns(2)	100 ns(2)	125 ns(2)	250 ns(2)	500 ns	2.0 µs

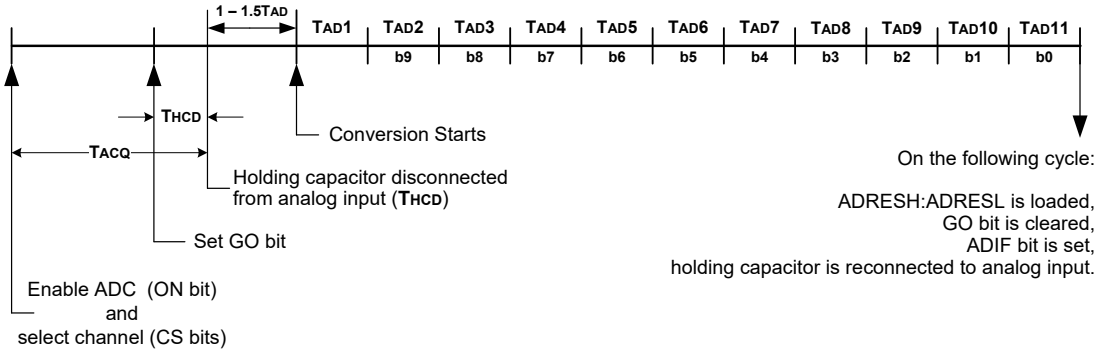
表 24-1. 不同器件频率（F_{OSC}）下的 ADC 时钟周期（T_{AD}）（续）

ADC 时钟源	CS[2:0]	不同器件频率（F_{OSC}）下的 ADC 时钟周期（T_{AD}）					
		32 MHz	20 MHz	16 MHz	8 MHz	4 MHz	1 MHz
F_{OSC}/4	'b100	125 ns(2)	200 ns(2)	250 ns(2)	500 ns	1.0 µs	4.0 µs
F_{OSC}/8	'b001	250 ns(2)	400 ns(2)	500 ns	1.0 µs	2.0 µs	8.0 µs
F_{OSC}/16	'b101	500 ns	800 ns	1.0 µs	2.0 µs	4.0 µs	16.0 µs(2)
F_{OSC}/32	'b010	1.0 µs	1.6 µs	2.0 µs	4.0 µs	8.0 µs	32.0 µs(2)
F_{OSC}/64	'b110	2.0 µs	3.2 µs	4.0 µs	8.0 µs	16.0 µs(2)	64.0 µs(2)
ADCRC	'bx11	1.0 - 6.0(1,3)					

注：

1. 有关 ADCRC 源典型 T_{AD} 值的 T_{AD} 参数，请参见“**电气规范**”一节。
2. 这些值违反了 T_{AD} 时间要求。
3. 通过系统时钟 F_{OSC} 来产生 ADC 时钟时，可以最大程度缩短 ADC 时钟周期（T_{AD}）和 ADC 总转换时间。但是，如果要在器件处于休眠模式时执行转换，则必须使用 ADCRC 振荡器源。

图 24-2. 10 位模数转换 T_{AD} 周期



24.1.5. 中断

ADC 模块可在模数转换完成时产生中断。每次转换完成时都会将 ADC 中断标志（ADIF）位置 1。如果 ADC 中断允许（ADIE）位置 1，则会产生 ADC 中断事件。必须用软件将 ADIF 位清零。

重要：

1. 不管是否允许 ADC 中断，每次转换完成时都会将 ADIF 位置 1。
2. 仅当选择了 ADCRC 振荡器作为时钟源时，ADC 才能在休眠模式下工作。

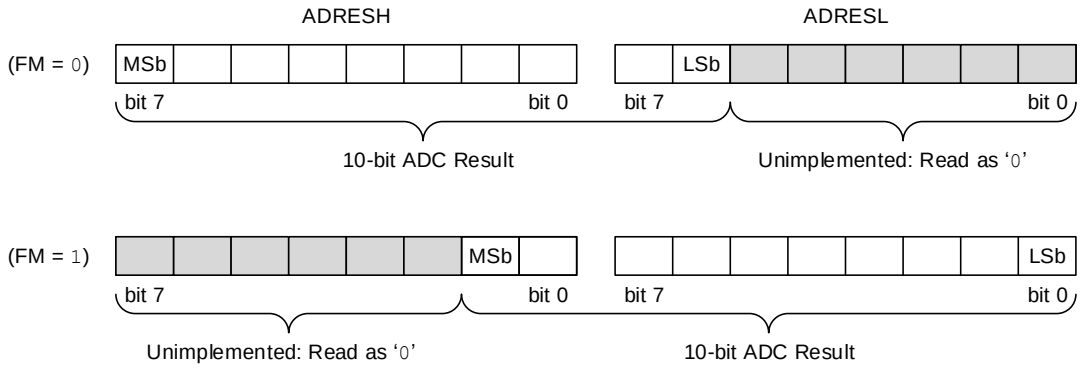
器件工作或休眠时都可产生 ADC 中断。器件处于休眠模式时：

- 如果 ADIE = 1、PEIE = 1 且 GIE = 0：中断可以将器件从休眠模式唤醒。从休眠状态唤醒时，将执行 SLEEP 指令之后的指令。不会执行中断服务程序。
- 如果 ADIE = 1、PEIE = 1 且 GIE = 1：中断可以将器件从休眠模式唤醒。从休眠状态唤醒时，总是执行紧跟 SLEEP 指令后的下一条指令。随后执行将切换到中断服务程序。

24.1.6. ADC 结果格式

10 位 ADC 转换的结果可采用两种格式：左对齐或右对齐。ADC 结果格式/对齐选择（FM）位控制输出格式，如图 24-3 所示。

图 24-3. 10 位 ADC 转换结果的格式



重要：写入 ADRES 寄存器对时始终采用右对齐，与所选的格式模式无关。因此，如果 FM = 0，则在写入 ADRES 后读取数据时将左移四位。

24.2. ADC 工作原理

24.2.1. 启动转换

要使能 ADC 模块，必须将 ON 位置 1。可通过以下任一方式启动转换：

- 使用软件将 GO 位置 1
- 外部触发信号（通过 ADC 自动转换触发源（ADACT）寄存器选择触发源）

重要：不得在启动 ADC 的同一指令中将 GO 位置 1。更多详细信息，请参见 ADC 转换步骤。

24.2.2. 转换完成

当转换完成时，ADC 模块将：

- 清零 GO 位
- 将 ADIF 位置 1
- 用新的转换结果更新 ADRES 寄存器

24.2.3. 休眠期间的 ADC 操作

ADC 模块可以在休眠模式下工作。这需要将 ADC 时钟源设置为 ADCRC 选项。当选择 ADC 振荡器源时，ADC 需等待一个额外的指令周期后才能启动转换。因此，可以执行 SLEEP 指令，从而降低转换期间的系统

噪声。如果允许了 ADC 中断（ADIE = 1），转换完成时器件将从休眠状态唤醒。如果禁止了 ADC 中断（ADIE = 0），尽管 ON 位仍保持置 1，器件也会处于休眠状态，ADC 模块被关闭。

ADC 时钟源不是 ADCRC 时，尽管 ON 位仍保持置 1，但 SLEEP 指令会导致当前转换中止，ADC 被关闭。

24.2.3.1. 休眠期间的外部触发器

如果在 ADC 处于休眠状态时接收到外部触发信号，则会记录触发信号，但直到器件退出休眠模式后才会开始转换。

24.2.4. 自动转换触发器

自动转换触发器允许定期进行 ADC 测量而无需软件干预。当出现选定源的上升沿时，GO 位由硬件置 1。

自动转换触发源使用自动转换触发源选择（ACT）位进行选择。

重要：使用自动转换触发器不能确保正确的 ADC 时序。用户需负责确保满足 ADC 时序要求。

24.2.5. ADC 转换步骤

以下是使用 ADC 执行模数转换的示例步骤：

1. 配置端口：
   a. 禁止引脚输出驱动器（见 TRISx 寄存器）
   b. 将引脚配置为模拟引脚（见 ANSELx 寄存器）
2. 配置 ADC 模块：
   a. 选择 ADC 转换时钟
   b. 配置参考电压
   c. 选择 ADC 输入通道
   d. 配置结果格式
   e. 启动 ADC 模块
3. 配置 ADC 中断（可选）：
   a. 清零 ADC 中断标志
   b. 允许 ADC 中断
   c. 允许外设中断（PEIE 位）
   d. 允许全局中断（GIE 位）(1)
4. 等待所需的采集时间(2)
5. 通过将 GO 位置 1 来启动转换
6. 通过以下任一方式等待 ADC 转换完成：
   - 轮询 GO 位
   - 等待 ADC 中断（如果允许了中断）
7. 读取 ADC 结果
8. 清零 ADC 中断标志（如果允许了中断）

注：

1. 禁止全局中断（GIE = 0）后，器件将从休眠模式唤醒，但不会进入中断服务程序。
2. 更多详细信息，请参见 ADC 采集要求。

例 24-1. ADC 转换（汇编语言）

```
; This code block configures the ADC for polling,
; VDD and VSS references,
; ADCRC oscillator, and AN0 input.
; Conversion start & polling for completion are included.

    BANKSEL TRISA
    BSF     TRISA,0     ; Set RA0 to input
    BANKSEL ANSEL
    BSF     ANSEL,0     ; Set RA0 to analog
    BANKSEL ADCON0
    CLRF    ADCON0
    CLRF    ADCON1
    CLRF    ADACT       ; Auto-conversion disabled
    BSF     ADCON0,0    ; CHS = RA0, ADC ON
    MOVLW   B'11110000' ; FM = Right-justified, CS = ADCRC, PREF = VDD
    MOVWF   ADCON1
    CALL    SampleTime  ; Acquisition delay
    BANKSEL ADCON0
    BSF     ADCON0,GO   ; Start conversion
    BTFSC   ADCON0,GO   ; Is conversion done?
    GOTO    $-1         ; No, test again
    BANKSEL ADRESH
    MOVF    ADRESH,W    ; Read upper byte
    MOVWF   RESULTHI    ; Store in GPR space
    MOVF    ADRESL,W    ; Read lower byte
    MOVWF   RESULTLO    ; Store in GPR space
```

例 24-2. ADC 转换（C 语言）

```
/*This code block configures the ADC
for polling, VDD and VSS references,
ADCRC oscillator and AN0 input.
Conversion start & polling for completion
are included.
  */
    void main() {
        //System Initialize
        initializeSystem();

        // Configure Port
        TRISAbits.TRISA0 = 1;       // Set RA0 to input
        ANSELAbits.ANSELA0 = 1;     // Set RA0 to analog

        // Configure ADC
        ADCON1bits.CS = 1;          // ADCRC Clock
        ADCON1bits.PREF = 'b11;     // VDD
        ADCON0bits.CHS = 'b000000; // RA0
        ADCON1bits.FM = 1;          // Right justify
        ADCON0bits.ON = 1;          // Turn ADC On

    while (1) {
        ADCON0bits.GO = 1;          // Start conversion
        while (ADCON0bits.GO);      // Wait for conversion done
        resultHigh = ADRESH;        // Read result
        resultLow = ADRESL;         // Read result
    }
 }
```

24.3. ADC 采集要求

为了使 ADC 达到规定的精度，必须使充电保持电容（C_{HOLD}）完全充电至输入通道的电压。模拟输入模型如图 24-4 所示。源阻抗（R_S）和内部采样开关（R_{SS}）阻抗直接影响电容 C_{HOLD} 的充电时间。采样开关

（R_{SS}）阻抗随器件电压（V_{DD}）的变化而变化。模拟信号源的最大阻抗推荐值为 10 kΩ。采集时间会随着源阻抗的降低而缩短。在选择（或更改）模拟输入通道后，必须在启动转换前完成 ADC 采集。可以使用公式 24-1 来计算最小采集时间。该公式假设误差为 1/2 LSb。1/2 LSb 误差是 ADC 达到规定分辨率所能允许的最大误差。

公式 24-1. 采集时间示例

假设：温度 = 50°C，外部阻抗 = 10 kΩ，V_{DD} = 5.0V

T_{ACQ} = 放大器稳定时间 + 保持电容充电时间 + 温度系数

$$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$$

$$T_{ACQ} = 2\,\mu s + T_C + \left[(Temperature - 25°C)\ (0.05\,\mu s/°C)\right]$$

T_C 值可以用以下公式计算近似值：

$$V_{APPLIED}\left(1 - \frac{1}{\left(2^{n+1}\right) - 1}\right) = V_{CHOLD}$$；[1]充电到 V_{CHOLD}（½ lsb 误差范围）

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{CHOLD}$$；[2]$V_{APPLIED}$ 与 V_{CHOLD} 的对应关系

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{APPLIED}\left(1 - \frac{1}{\left(2^{n+1}\right) - 1}\right)$$；合并[1]和[2]

注：其中，n 为 ADC 的分辨率（单位为位）

求解 T_C：

$$T_C = -C_{HOLD}(R_{IC} + R_{SS} + R_S)\ ln\ (1/2047)$$

$$T_C = -10\,pF(1\,k\Omega + 7\,k\Omega + 10\,k\Omega)\ ln\ (0.0004885)$$

$$T_C = 1.37\,\mu s$$

因此：

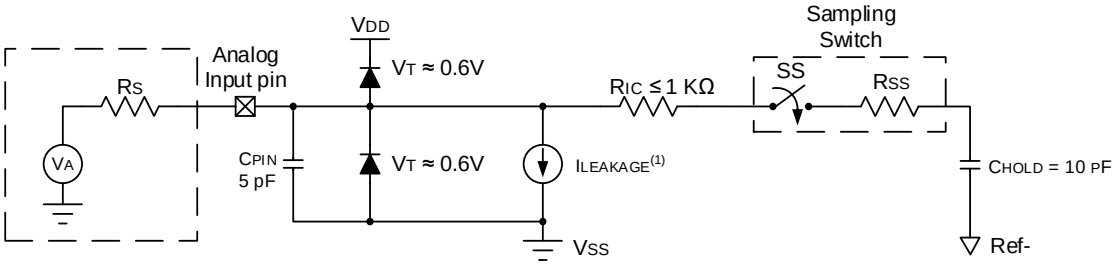
$$T_{ACQ} = 2\,\mu s + 1.37\,\mu s + \left[(50°C - 25°C)\ (0.05\,\mu s/°C)\right]$$

$$T_{ACQ} = 4.62\,\mu s$$

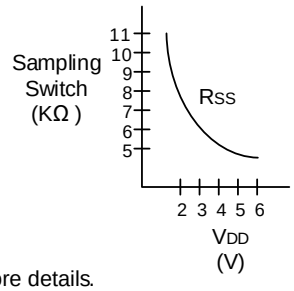
重要：

- 因为参考电压（V_{REF}）自行抵消，因此它对该公式没有影响。
- 充电保持电容（C_{HOLD}）在每次转换后不会放电。
- 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泄漏电流规范。

图 24-4. 模拟输入电路模型

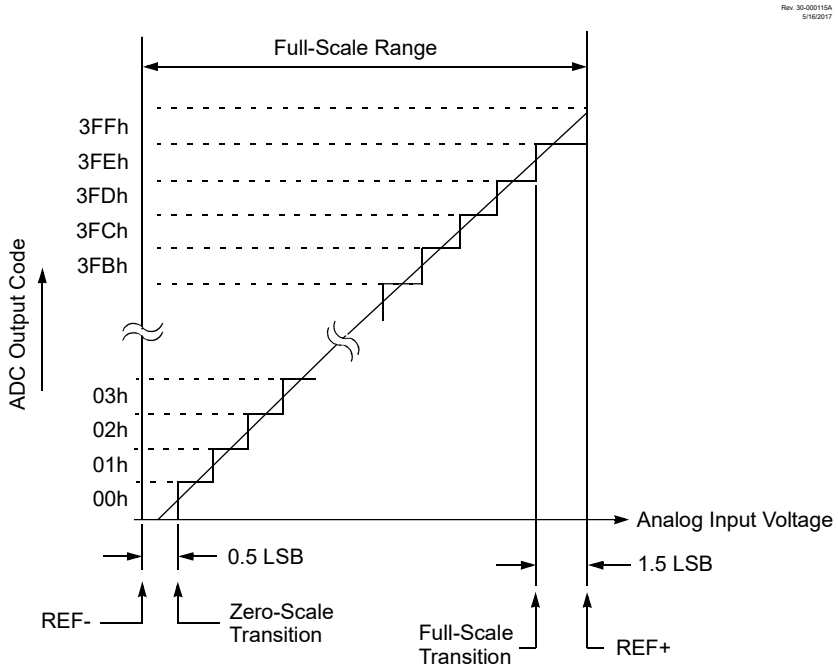


Legend: CPIN = Input Capacitance
ILEAKAGE = Leakage Current at the pin due to various junctions
RIC = Interconnect Resistance
RS = Source Impedance
VA = Analog Voltage
VT = Diode Forward Voltage
SS = Sampling Switch
RSS = Resistance of the Sampling Switch
CHOLD = Sample/Hold Capacitance



Note:
1. Refer to the *Electrical Specifications* section of the device data sheet for more details.

图 24-5. ADC 传递函数



24.4. 寄存器定义：ADC 控制

表 24-2. ADC 长位名称前缀

外设	位名称前缀
ADC	AD

24.4.1. ADCON0

名称： ADCON0
偏移量： 0x09D

ADC 控制寄存器 0

位	7	6	5	4	3	2	1	0
	CHS[5:0]						GO	ON
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W/HS/HC	R/W
复位	0	0	0	0	0	0	0	0

Bit 7:2 – CHS[5:0] 模拟通道选择

值	说明
111111-011100	保留
011011	V_{SS}
011010-010110	保留
010101	RC5
010100	RC4
010011	RC3
010010	RC2
010001-000110	保留
000101	RA5
000100	RA4
000011	保留
000010	RA2
000001	RA1
000000	RA0

Bit 1 – GO ADC 转换状态

值	说明
1	ADC 转换正在进行。将该位置 1 可启动 ADC 转换周期。当 ADC 转换完成后，该位由硬件自动清零。
0	ADC 转换已完成/未进行

Bit 0 – ON ADC 使能

值	说明
1	使能 ADC
0	禁止 ADC，不消耗工作电流

24.4.2. ADCON1

名称： ADCON1
偏移量： 0x09E

ADC 控制寄存器 1

位	7	6	5	4	3	2	1	0
	FM	CS[2:0]					PREF[1:0]	
访问	R/W	R/W	R/W	R/W			R/W	R/W
复位	0	0	0	0			0	0

Bit 7 – FM ADC 结果格式/对齐选择

值	说明
1	右对齐。ADRESH 的高 6 位用 0 填充。
0	左对齐。ADRESL 的低 6 位用 0 填充。

Bit 6:4 – CS[2:0] ADC 转换时钟选择

值	说明
111	ADCRC
110	$F_{OSC}/64$
101	$F_{OSC}/16$
100	$F_{OSC}/4$
011	ADCRC
010	$F_{OSC}/32$
001	$F_{OSC}/8$
000	$F_{OSC}/2$

Bit 1:0 – PREF[1:0] ADC 正参考电压配置

值	说明
11	保留
10	V_{REF}+连接到外部 V_{REF}+引脚
01	保留
00	V_{REF}+连接到 V_{DD}

24.4.3. ADACT

名称： ADACT
偏移量： 0x09F

ADC 自动转换触发源选择寄存器

位	7	6	5	4	3	2	1	0
					ACT[3:0]			
访问					R/W	R/W	R/W	R/W
复位					0	0	0	0

Bit 3:0 – ACT[3:0] 自动转换触发源选择

值	说明
1111-1110	保留
1101	软件读取/写入 ADRESH
1100-1010	保留
1001	电平变化中断标志位
001000	PWM4_OUT
0111	PWM3_OUT
0110	CCP2_OUT
0101	CCP1_OUT
0100	TMR2_postscaled_OUT
0011	TMR1_overflow
0010	TMR0_overflow
0001	通过 ADACTPPS 选择的引脚
0000	禁止外部触发

24.4.4. ADRES

名称： ADRES
偏移量： 0x09B

ADC 结果寄存器

位	15	14	13	12	11	10	9	8
	ADRES[15:8]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

位	7	6	5	4	3	2	1	0
	ADRES[7:0]							
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 15:0 – ADRES[15:0] ADC 采样结果

注：该多字节寄存器中的各个字节可使用以下寄存器名称进行访问：

- ADRESH：访问高字节 ADRES[15:18]
- ADRESL：访问低字节 ADRES[7:0]

24.5. 寄存器汇总——ADC

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x9A	保留									
0x9B	ADRES	7:0	ADRES[7:0]							
		15:8	ADRES[15:8]							
0x9D	ADCON0	7:0	CHS[5:0]						GO	ON
0x9E	ADCON1	7:0	FM	CS[2:0]					PREF[1:0]	
0x9F	ADACT	7:0					ACT[3:0]			

25. 电荷泵

该系列器件具有一个专用电荷泵，可通过电荷泵控制（CPCON）寄存器进行控制。电荷泵的主要用途是为模拟外设中包含的晶体管器件的栅极、信号以及参考输入传输门提供恒定电压，以防止晶体管在低工作电压下的性能下降。

电荷泵提供以下模式：

- 手动使能
- 自动使能
- 禁止

25.1. 手动使能

电荷泵可通过电荷泵使能（CPON）位手动使能。当 CPON 位配置为 11 时，使能电荷泵。在这种情况下，无论 V_{DD} 电压如何，电荷泵都会为所有模拟系统提供额外的电压，但也会消耗额外的电流。

25.2. 自动使能

电荷泵也可以自动使能。在这种情况下，应用程序可决定何时使能电荷泵。如果在 V_{DD} 电压高于阈值时使能电荷泵，电荷泵的模拟性能不会提高，但会消耗更多的电流。通过允许硬件监视 V_{DD} 并决定何时使能电荷泵，可以防止不必要的电流消耗。

当 CPON 位配置为 10 时，电荷泵硬件会监视 V_{DD} 并将 V_{DD} 电压与参考电压阈值（V_{AUTO}）进行比较，该阈值设置为 4.6V。当硬件检测到 V_{DD} 电压低于阈值时，会自动使能电荷泵。如果 V_{DD} 恢复为高于阈值的电压，则硬件会自动禁止电荷泵。

当 CPON 位配置为 01 时，电荷泵硬件会等待模拟外设（例如 ADC）使能后再监视 V_{DD}。在这种情况下，电荷泵硬件会监视所有模拟外设。一旦有模拟外设使能，硬件便开始比较 V_{DD} 与 V_{AUTO}。当硬件检测到 V_{DD} 电压低于阈值时，会使能电荷泵。如果 V_{DD} 恢复为高于阈值的电压或者模拟外设被禁止，则会自动禁止电荷泵。

25.3. 禁止

电荷泵默认禁止（CPON = 00）。清零 CPON 位将禁止电荷泵。

25.4. 电荷泵阈值

电荷泵阈值（CPT）位指示 V_{DD} 是否为可接受的工作电压。电荷泵硬件会将 V_{DD} 与 4.6V 的阈值电压（V_{AUTO}）进行比较。如果 V_{DD} 高于 V_{AUTO}，CPT 位置 1（CPT = 1）。如果 V_{DD} 低于 V_{AUTO}，CPT 位清零（CPT = 0）。

25.5. 电荷泵就绪

电荷泵就绪状态（CPRDY）位指示电荷泵是否就绪备用。当 CPRDY 置 1（CPRDY = 1）时，电荷泵已达到稳态，就绪备用。当 CPRDY 清零（CPRDY = 0）时，电荷泵处于关闭状态或尚未达到稳态。

25.6. 寄存器定义：电荷泵

25.6.1. CPCON

名称： CPCON
偏移量： 0x009A

电荷泵控制寄存器

位	7	6	5	4	3	2	1	0
	CPON[1:0]						CPT	CPRDY
访问	R/W	R/W					R	R
复位	0	0					0	0

Bit 7:6 – CPON[1:0]
电荷泵使能

值	说明
11	使能电荷泵
10	当 $V_{DD} < V_{AUTO}$（V_{AUTO} = 4.6V）时，自动使能电荷泵
01	当模拟外设使能（ADCON0.ON = 1）且 $V_{DD} < V_{AUTO}$ 时，自动使能电荷泵
00	禁止电荷泵

Bit 1 – CPT
电荷泵阈值

值	说明
1	V_{DD} 高于电荷泵自动使能阈值（V_{AUTO}）
0	V_{DD} 低于电荷泵自动使能阈值（V_{AUTO}）

Bit 0 – CPRDY
电荷泵就绪状态

值	说明
1	电荷泵已达到稳态
0	电荷泵关闭或尚未达到稳态

26. 指令集汇总

CN5225 器件包含标准指令集（50 条指令）。每条指令都是一个 14 位字，包含操作码和所有必需的操作数。操作码可分为以下三大类：

- 面向字节操作类指令
- 面向位操作类指令
- 立即数和控制操作类指令

立即数和控制类指令字格式最为丰富。

表 26-3 列出了指令集中包含的指令。

所有指令都可在单指令周期内执行完毕，但也有例外，以下指令需要两或三个周期：

- 子程序进入需要两个周期（CALL 和 CALLW）
- 从中断或子程序返回需要两个周期（RETURN、RETLW 和 RETFIE）
- 程序跳转需要两个周期（GOTO、BRA、BRW、BTFSS、BTFSC、DECFSZ 和 INCSFZ）
- 当任何指令引用间接文件寄存器时以及当文件选择寄存器指向程序存储器时，还需要多用一个指令周期

一个指令周期由 4 个振荡周期组成；如果振荡器频率为 4 MHz，则可提供 1 MHz 的标称指令执行速率。

所有指令示例均使用 0xhh 格式表示一个十六进制数，其中 h 表示一位十六进制数字。

26.1. 读-修改-写操作

所有指定了文件寄存器的指令都将执行读-修改-写（RMW）操作。根据目标寄存器指示符“d”的状态，对工作（W）寄存器或原始文件寄存器进行读取，修改数据并存储结果（更多信息，请参见表 26-1）。即使指令是写入寄存器，也会先读取该寄存器。

表 26-1. 操作码字段说明

字段	说明
f	文件寄存器地址（0x00 至 0x7F）
W	工作寄存器（累加器）
b	8 位文件寄存器内的位地址
k	立即数字段、常量数据或标号
x	忽略（= 0 或 1）
d	目标寄存器选择；d = 0：结果保存至 W，d = 1：结果保存至文件寄存器 f
n	FSR 或 INDF 编号（0-1）
mm	预/后增/减模式选择

表 26-2. 缩写说明

字段	说明
PC	程序计数器
$\overline{TO}$	超时位
C	进位位
DC	半进位位
Z	全零标志位
$\overline{PD}$	掉电位

26.2. 标准指令集

表 26-3. 指令集

助记符和操作数		说明	周期数	14 位操作码				受影响的状态位	注
				MSb			LSb		
面向字节的操作类指令									
ADDWF	f, d	WREG 和 f 相加	1	00	0111	dfff	ffff	C、DC 和 Z	2
ADDWFC	f, d	WREG 和 f 进行带进位的相加	1	11	1101	dfff	ffff	C、DC 和 Z	2
ANDWF	f, d	WREG 和 f 作逻辑与运算	1	00	0101	dfff	ffff	Z	2
ASRF	f, d	算术右移	1	11	0111	dfff	ffff	C 和 Z	2
LSLF	f, d	逻辑左移	1	11	0101	dfff	ffff	C 和 Z	2
LSRF	f, d	逻辑右移	1	11	0110	dfff	ffff	C 和 Z	2
CLRF	f	将 f 清零	1	00	0001	lfff	ffff	Z	2
CLRW	-	将 WREG 清零	1	00	0001	0000	00xx	Z	
COMF	f, d	f 求逆	1	00	1001	dfff	ffff	Z	2
DECF	f, d	f 递减 1	1	00	0011	dfff	ffff	Z	2
INCF	f, d	f 递增 1	1	00	1010	dfff	ffff	Z	2
IORWF	f, d	WREG 与 f 作逻辑或运算	1	00	0100	dfff	ffff	Z	2
MOVF	f, d	传送 f	1	00	1000	dfff	ffff	Z	2
MOVWF	f	将 WREG 中的内容送入 f	1	00	0000	1fff	ffff	无	2
RLF	f, d	对 f 执行带进位的循环左移	1	00	1101	dfff	ffff	C	2
RRF	f, d	对 f 执行带进位的循环右移	1	00	1100	dfff	ffff	C	2
SUBWF	f, d	f 减去 WREG	1	00	0010	dfff	ffff	C、DC 和 Z	2
SUBWFB	f, d	f 减去 WREG（带借位）	1	11	1011	dfff	ffff	C、DC 和 Z	2
SWAPF	f, d	将 f 中的两个半字节进行交换	1	00	1110	dfff	ffff	无	2
XORWF	f, d	WREG 与 f 作逻辑异或运算	1	00	0110	dfff	ffff	Z	2
面向字节的跳过操作									
DECFSZ	f, d	f 递减 1，为 0 则跳过	1(2)	00	1011	dfff	ffff	无	1 和 2

表 26-3. 指令集 （续）

助记符和操作数		说明	周期数	14 位操作码				受影响的状态位	注
				MSb			LSb		
INCFSZ	f, d	f 递增 1，为 0 则跳过	1(2)	00	1111	dfff	ffff	无	1 和 2
面向位的文件寄存器操作									
BCF	f, b	将 f 中的指定位清零	1	01	00bb	bfff	ffff	无	2
BSF	f, b	将 f 中的指定位置 1	1	01	01bb	bfff	ffff	无	2
面向位的跳过操作									
BTFSC	f, b	对 f 中的指定位进行测试，如果为零则跳过	1(2)	01	10bb	bfff	ffff	无	1 和 2
BTFSS	f, b	对 f 中的指定位进行测试，如果为 1 则跳过	1(2)	1010	11bb	bfff	ffff	无	1 和 2
立即数操作									
ADDLW	k	立即数和 WREG 相加	1	11	1110	kkkk	kkkk	C、DC 和 Z	
ANDLW	k	立即数与 WREG 作逻辑与运算	1	11	1001	kkkk	kkkk	Z	
IORLW	k	立即数与 WREG 作逻辑或运算	1	11	1000	kkkk	kkkk	Z	
MOVLB	k	将立即数传送到 BSR	1	00	000	0k	kkkk	无	
MOVLP	k	将立即数传送到 PCLATH	1	11	0001	1kkk	kkkk	无	
MOVLW	k	将立即数传送到 W	1	11	0000	kkkk	kkkk	无	
SUBLW	k	从立即数中减去 W 的内容	1	11	1100	kkkk	kkkk	C、DC 和 Z	
XORLW	k	立即数与 W 作逻辑异或运算	1	11	1010	kkkk	kkkk	Z	
控制类操作									

表 26-3. 指令集 （续）

助记符和操作数		说明	周期数	14 位操作码				受影响的状态位	注
				MSb			LSb		
BRA	k	相对跳转	2	11	001k	kkkk	kkkk	无	
BRW	—	将 WREG 寄存器的内容作为偏移量进行相对跳转	2	00	0000	0000	1011	无	
CALL	k	调用子程序	2	10	0kkk	kkkk	kkkk	无	
CALLW	—	调用地址由 WREG 寄存器指定的子程序	2	00	0000	0000	1010	无	
GOTO	k	跳转到地址	2	10	1kkk	kkkk	kkkk	无	
RETFIE	k	从中断返回	2	00	0000	0000	1001	无	
RETLW	k	返回并将立即数传送到 WREG	2	11	0100	kkkk	kkkk	无	
RETURN	—	从子程序返回	2	00	0000	0000	1000	无	
固有操作									
CLRWDT	—	将看门狗定时器清零	1	00	0000	0110	0100	$\overline{TO}$ 和 $\overline{PD}$	
NOP	—	空操作	1	00	0000	0000	0000	无	
RESET	—	软件器件复位	1	00	0000	0000	0001	无	
SLEEP	—	进入待机模式	1	00	0000	0110	0011	$\overline{TO}$ 和 $\overline{PD}$	
TRIS	f	将 WREG 寄存器的内容装入 TRIS 寄存器	1	00	0000	0110	0fff	无	
优化的 C 编译器指令									
ADDFSR	n, k	立即数 k 和 FSRn 相加	1	11	0001	0nkk	kkkk	无	
MOVIW	n, mm	将间接 FSRn 的内容传送到 WREG 寄存器，带预/后增/减修改符 mm	1	00	0000	0001	0nmm	Z	2 和 3
	k[n]	将 INDFn 的内容传送到 WREG 寄存器，采用变址间接寻址模式	1	11	1111	0nkk	kkkk	Z	2
MOVWI	n, mm	将 WREG 寄存器的内容传送到间接 FSRn，带预/后增/减修改符 mm	1	00	0000	0001	1nmm	无	2 和 3
	k[n]	将 WREG 寄存器的内容传送到 INDFn，采用变址间接寻址模式	1	11	1111	1nkk	kkkk	无	2

注：

1. 如果程序计数器（PC）被修改或条件测试结果为真，则该指令需要两个周期。第二个周期执行一条 NOP 指令。
2. 如果该指令寻址的是 INDF 寄存器且相应 FSR 的 MSb 置 1，则该指令需要一个额外的指令周期。
3. 有关 MOVIW 和 MOVWI 指令的详细说明，请参见下一节。

26.2.1. 标准指令集

ADDFSR	立即数和 FSRn 相加
语法：	[*标号*] ADDFSR FSRn, k
操作数：	-32 ≤ k ≤ 31； n ∈ [0, 1]
操作：	FSR(n) + k → FSR(n)
受影响的状态位：	无
说明：	将有符号 6 位立即数 k 与 FSRnH:FSRnL 寄存器对的内容相加。 FSRn 地址范围限制为 0000h-FFFFh。地址超出该边界时，FSR 会发生折回。

ADDLW	立即数和 W 相加
语法：	[*标号*] ADDLW k
操作数：	0 ≤ k ≤ 255
操作：	(W) + k → (W)
受影响的状态位：	C、DC 和 Z
说明：	将 W 寄存器的内容与 8 位立即数 k 相加，结果存入 W 寄存器。

ADDWF	W 和 f 相加
语法：	[*标号*] ADDWF f, d
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]
操作：	(W) + (f) → 目标寄存器
受影响的状态位：	C、DC 和 Z
说明：	将 W 寄存器的内容与寄存器 f 的内容相加。如果 d 为 0，结果存入 W 寄存器。 如果 d 为 1，结果存回寄存器 f。

ADDWFC	W 和 f 进行带进位的相加
语法：	[*标号*] ADDWFC f {,d}
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]
操作：	(W) + (f) + (C) → 目标寄存器
受影响的状态位：	C、DC 和 Z
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，结果存放到 W 寄存器。如果 d 为 1，结果存放到数据存储单元 f。

ANDLW	立即数和 W 作逻辑与运算
语法：	[*标号*] ANDLW k
操作数：	0 ≤ k ≤ 255
操作：	(W) .AND. k → (W)
受影响的状态位：	Z
说明：	将 W 寄存器的内容与 8 位立即数 k 进行逻辑与运算。 结果存入 W 寄存器。

ANDWF	W 和 f 作逻辑与运算
语法：	[*标号*] ANDWF f, d
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]
操作：	(W) .AND.(f) → 目标寄存器

标准指令集
（续）

ANDWF	W 和 f 作逻辑与运算
受影响的状态位：	Z
说明：	将 W 寄存器的内容与寄存器 f 的内容进行逻辑与运算。如果 d 为 0，结果存入 W 寄存器。 如果 d 为 1，结果存回寄存器 f。

ASRF	算术右移
语法：	[*标号*] ASRF f, d
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]
操作：	(f[7]) → dest[7] (f[7:1]) → dest[6:0] (f[0]) → C
受影响的状态位：	C 和 Z
说明：	将寄存器 f 的内容连同进位标志位一起右移 1 位。 MSb 保持不变。 如果 d 为 0，结果存入 W 寄存器。 如果 d 为 1，结果存回寄存器 f。 **寄存器 f → C**

BCF	将 f 中的指定位清零
语法：	[*标号*] BCF f, b
操作数：	0 ≤ f ≤ 127 0 ≤ b ≤ 7
操作：	0 → f[b]
受影响的状态位：	无
说明：	将寄存器 f 中的 bit b 清零。

BRA	相对跳转
语法：	[*标号*] BRA label [*标号*] BRA $+k
操作数：	-256 ≤ 标号 - PC + ≤ 255 -256 ≤ k ≤ 255
操作：	(PC) + 1 + k → PC
受影响的状态位：	无
说明：	将有符号 9 位立即数 k 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 + 1 + k。 该指令为一条双周期指令。该转移的地址范围存在限制。

BRW	将 W 寄存器的内容作为偏移量进行相对转移
语法：	[*标号*] BRW
操作数：	无
操作：	(PC) + (W) → PC
受影响的状态位：	无
说明：	W 的内容（无符号）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 + 1 + (W)。 该指令为一条双周期指令。

BSF	将 f 中的指定位置 1
语法：	[*标号*] BSF f, b
操作数：	0 ≤ f ≤ 127 0 ≤ b ≤ 7
操作：	1 → (f[b])
受影响的状态位：	无
说明：	将寄存器 f 中的 bit b 置 1。

BTFSC	测试文件中的某位，为 0 则跳过
语法：	[*标号*] BTFSC f, b
操作数：	0 ≤ f ≤ 127 0 ≤ b ≤ 7
操作：	如果(f[b]) = 0，则跳过
受影响的状态位：	无
说明：	如果寄存器 f 的位 b 为 1，则执行下一条指令。 如果寄存器 f 的 bit b 为 0，则丢弃下一条指令，代之执行一条 NOP 指令，使之成为一条双周期指令。

BTFSS	测试文件中的某位，为 1 则跳过
语法：	[*标号*] BTFSS f, b
操作数：	0 ≤ f ≤ 127 0 ≤ b < 7
操作：	如果(f[b]) = 1，则跳过
受影响的状态位：	无
说明：	如果寄存器 f 中的 bit b 为 0，则执行下一条指令。 如果位 b 为 1，则丢弃下一条指令，代之执行一条 NOP 指令，使之成为一条双周期指令。

CALL	调用子程序
语法：	[*标号*] CALL k
操作数：	0 ≤ k ≤ 2047
操作：	(PC) + 1 → TOS， k → PC[10:0]， (PCLATH[6:3]) → PC[14:11]
受影响的状态位：	无
说明：	调用子程序。首先，将返回地址（PC + 1）压入堆栈。 11 位直接地址被装入 PC 的 bit[10:0]。 将 PCLATH 的内容装入 PC 的高位。 CALL 是一条双周期指令。

CALLW	使用 W 调用子程序
语法：	[*标号*] CALLW
操作数：	无
操作：	(PC) + 1 → TOS， (W) → PC[7:0]， (PCLATH[6:0]) → PC[14:8]
受影响的状态位：	无
说明：	使用 W 调用子程序。首先，将返回地址（PC + 1）压入返回堆栈。 然后，W 的内容被装入 PC[7:0]，PCLATH 的内容被装入 PC[14:8]。 CALLW 是一条双周期指令。

CLRF	将 f 清零
语法：	[*标号*] CLRF f
操作数：	0 ≤ f ≤ 127
操作：	000h → f 1 → Z
受影响的状态位：	Z
说明：	寄存器 f 的内容被清零，并且 Z 位被置 1。

CLRW	将 W 寄存器清零
语法：	[*标号*] CLRW
操作数：	无
操作：	00h → (W) 1 → Z
受影响的状态位：	Z
说明：	W 寄存器被清零。全零标志（Z）位置 1。

CLRWDT	将看门狗定时器清零
语法：	[*标号*] CLRWDT
操作数：	无
操作：	00h → WDT， 00h → WDT 预分频器， 1 →$\overline{TO}$ 1 →$\overline{PD}$
受影响的状态位：	$\overline{TO}$ 和 $\overline{PD}$
说明：	`CLRWDT` 指令复位看门狗定时器。 还将复位 WDT 的预分频器。 状态位 $\overline{TO}$ 和 $\overline{PD}$ 均被置 1。

COMF	f 取反
语法：	[*标号*] COMF f, d
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]
操作：	($\bar{f}$) → 目标寄存器
受影响的状态位：	Z
说明：	对寄存器 f 的内容求补。 如果 d 为 `0`，结果存入 W 寄存器。 如果 d 为 `1`，结果存回寄存器 f。

DECF	f 递减 1
语法：	[*标号*] DECF f, d
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]
操作：	(f) – 1 → 目标寄存器
受影响的状态位：	Z
说明：	将寄存器 f 的内容递减 1。 如果 d 为 `0`，结果存入 W 寄存器。 如果 d 为 `1`，结果存回寄存器 f。

DECFSZ	f 递减 1，为 0 则跳过
语法：	[*标号*] DECFSZ f, d

标准指令集
（续）

DECFSZ	f 递减 1，为 0 则跳过
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]
操作：	(f) – 1 → 目标寄存器， 结果 = 0 则跳过
说明：	将寄存器 f 的内容递减 1。 如果 d 为 0，结果存入 W 寄存器。 如果 d 为 1，结果存回寄存器 f。 如果结果为 1，则执行下一条指令。 如果结果为 0，代之执行一条 NOP 指令，使之成为一条双周期指令。

GOTO	无条件转移
语法：	[*标号*] GOTO k
操作数：	0 ≤ k ≤ 2047
操作：	k → PC[10:0] PCLATH[6:3] → PC[14:11]
受影响的状态位：	无
说明：	GOTO 是无条件转移指令。 11 位立即数值被装入 PC 的 bit[10:0]。 将 PCLATH[4:3]的内容装入 PC 的高位。 GOTO 是一条双周期指令。

INCF	f 递增 1
语法：	[*标号*] INCF f, d
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]
操作：	(f) + 1 → 目标寄存器
受影响的状态位：	Z
说明：	将寄存器 f 的内容递增 1。 如果 d 为 0，结果存入 W 寄存器。 如果 d 为 1，结果存回寄存器 f。

INCFSZ	f 递增 1，为 0 则跳过
语法：	[*标号*] INCFSZ f, d
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]
操作：	(f) + 1 → 目标寄存器， 结果 = 0 则跳过
受影响的状态位：	无
说明：	将寄存器 f 的内容递增 1。 如果 d 为 0，结果存入 W 寄存器。 如果 d 为 1，结果存回寄存器 f。 如果结果为 1，则执行下一条指令。 如果结果为 0，则转而执行一条 NOP 指令，使该指令成为双周期指令。

IORLW	立即数与 W 作逻辑或运算
语法：	[*标号*] IORLW k
操作数：	0 ≤ k ≤ 255
操作：	(W) .OR. k → (W)

标准指令集
（续）

IORLW	立即数与 W 作逻辑或运算
受影响的状态位：	Z
说明：	将 W 寄存器的内容与 8 位立即数 k 进行逻辑或运算。 结果存入 W 寄存器。

IORWF	W 与 f 作逻辑或运算
语法：	IORWF f, d
操作数：	0 ⩽ f ⩽ 127 d ∈ [0,1]
操作：	(W) .OR.(f) → 目标寄存器
受影响的状态位：	Z
说明：	将 W 寄存器的内容与寄存器 f 的内容进行逻辑或运算。 如果 d 为 0，结果存入 W 寄存器。 如果 d 为 1，结果存回寄存器 f。

LSLF	逻辑左移
语法：	[*标号*] LSLF f {,d}
操作数：	0 ⩽ f ⩽ 127 d ∈ [0,1]
操作：	(f[7]) → C (f[6:0]) → dest[7:1] 0 → dest[0]
受影响的状态位：	C 和 Z
说明：	将寄存器 f 的内容连同进位标志位一起左移 1 位。 0 移入 LSb。 如果 d 为 0，结果存入 W 寄存器。 如果 d 为 1，结果存回寄存器 f。 **C ← 寄存器 f ← 0**

LSRF	逻辑右移
语法：	[*标号*] LSRF f {,d}
操作数：	0 ⩽ f ⩽ 127 d ∈ [0,1]
操作：	0 → dest[7] (f[7:1]) → dest[6:0]， (f[0]) → C
受影响的状态位：	C 和 Z
说明：	将寄存器 f 的内容连同进位标志位一起右移 1 位。 0 移入 MSb。 如果 d 为 0，结果存入 W 寄存器。 如果 d 为 1，结果存回寄存器 f。 **0 → 寄存器 f → C**

MOVF	传送 f
语法：	[*标号*] MOVF f, d
操作数：	0 ⩽ f ⩽ 127 d ∈ [0,1]
操作：	f → 目标寄存器

标准指令集
（续）

MOVF	传送 f
受影响的状态位：	Z
说明：	根据 d 的状态，将寄存器 f 的内容传送到目标寄存器。 如果 d = 0，目标寄存器为 W 寄存器。 如果 d = 1，则目标寄存器为文件寄存器。 由于状态标志位 Z 会受影响，可用 d = 1 对文件寄存器内容进行检测。
指令字数：	1
指令周期数：	1

示例：	`MOVF  FSR, 0`
执行指令后 W = FSR 寄存器中的值 Z = 1	

MOVIW	将 INDFn 的内容传送到 W 寄存器
语法：	[*标号*] MOVIW ++FSRn [*标号*] MOVIW --FSRn [*标号*] MOVIW FSRn++ [*标号*] MOVIW FSRn-- [*标号*] MOVIW k[FSRn]
操作数：	n ∈ [0,1] mm ∈ [00,01,10,11] -32 ≤ k ≤ 31
操作：	INDFn → (W) 有效的地址由以下项决定 • FSR + 1（预增 1） • FSR - 1（预减 1） • FSR + k（相对偏移） 执行传送指令后，FSR 值为以下任一项： • FSR + 1（所有值都加 1） • FSR - 1（所有值都减 1） • 不变
受影响的状态位：	Z

MODE	语法	mm
预增	++FSRn	00
预减	--FSRn	01
后增	FSRn++	10
后减	FSRn--	11

说明：	该指令用于在 W 寄存器和任一间接寄存器（INDFn）之间传送数据。 执行该传送指令之前/之后，可通过对其进行预/后增/减来更新指针（FSRn）。 INDFn 寄存器不是物理寄存器。 任何访问 INDFn 寄存器的指令实际上访问的是由 FSRn 指定的地址处的寄存器。 FSRn 地址范围限制为 0000h-FFFFh。 地址递增/递减到超出这些边界时，将导致它发生折回。

MOVLB	将立即数传送到 BSR
语法：	[*标号*] MOVLB k

标准指令集
（续）

MOVLB	将立即数传送到 BSR
操作数：	0 ≤ k ≤ 127
操作：	k → BSR
受影响的状态位：	无
说明：	将 6 位立即数 k 装入存储区选择寄存器（BSR）。

MOVLP	将立即数传送到 PCLATH
语法：	[*标号*] MOVLP k
操作数：	0 ≤ k ≤ 127
操作：	k → PCLATH
受影响的状态位：	无
说明：	将 7 位立即数 k 装入 PCLATH 寄存器。

MOVLW	将立即数传送到 W			
语法：	[*标号*] MOVLW k			
操作数：	0 ≤ k ≤ 255			
操作：	k → (W)			
受影响的状态位：	无			
说明：	将 8 位立即数 k 装入 W 寄存器。 其余无关位均汇编为 0。			
指令字数：	1			
指令周期数：	1			

示例：	MOVLW	5Ah
执行指令后 W = 5Ah		

MOVWF	将 W 的内容传送到 f			
语法：	[*标号*] MOVWF f			
操作数：	0 ≤ f ≤ 127			
操作：	(W) → f			
受影响的状态位：	无			
说明：	将 W 的数据传送到寄存器 f。			
指令字数：	1			
指令周期数：	1			

示例：	MOVWF	LATA
执行指令前 LATA = FFh W = 4Fh 执行指令后 LATA = 4Fh W = 4Fh		

MOVWI	将 W 寄存器的内容传送到 INDFn
语法：	[*标号*] MOVWI ++FSRn [*标号*] MOVWI --FSRn [*标号*] MOVWI FSRn++ [*标号*] MOVWI FSRn-- [*标号*] MOVWI k[FSRn]
操作数：	n ∈ [0,1] mm ∈ [00,01,10,11] -32 ≤ k ≤ 31
操作：	(W) → INDFn 有效的地址由以下项决定 • FSR + 1（预增 1） • FSR - 1（预减 1） • FSR + k（相对偏移） 执行传送指令后，FSR 值为以下任一项： • FSR + 1（所有值都加 1） • FSR - 1（所有值都减 1） • 不变
受影响的状态位：	无

MODE	语法	mm
预增	++FSRn	00
预减	--FSRn	01
后增	FSRn++	10
后减	FSRn--	11

说明：	该指令用于在 W 寄存器和任一间接寄存器（INDFn）之间传送数据。 执行该传送指令之前/之后，可通过对其进行预/后增/减来更新指针（FSRn）。 INDFn 寄存器不是物理寄存器。 任何访问 INDFn 寄存器的指令实际上访问的是由 FSRn 指定的地址处的寄存器。 FSRn 地址范围限制为 0000h-FFFFh。 地址递增/递减到超出这些边界时，将导致它发生折回。 对 FSRn 的递增/递减操作不会影响任何状态位。

NOP	空操作			
语法：	[*标号*] NOP			
操作数：	无			
操作：	空操作。			
受影响的状态位：	无			
说明：	空操作。			
指令字数：	1			
指令周期数：	1			

示例：	NOP
无。	

RESET	软件复位
语法：	[*标号*] RESET
操作数：	无
操作：	执行器件复位。 复位 PCON 寄存器的 $\overline{RI}$ 标志。

标准指令集
（续）

RESET	软件复位
受影响的状态位：	无
说明：	此指令可实现用软件执行硬件复位。

RETFIE	从中断返回			
语法：	[*标号*] RETFIE k			
操作数：	无			
操作：	(TOS) → PC， 1 → GIE			
受影响的状态位：	无			
说明：	从中断返回。 执行出栈操作，将栈顶（TOS）的内容装入 PC。 通过将全局中断允许位 GIE（INTCON[7]）置 1，来允许中断。 该指令是双周期指令。			
指令字数：	1			
指令周期数：	2			

示例：	RETFIE
中断后 PC = TOS GIE = 1	

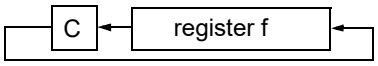
RETLW	将立即数返回 W			
语法：	[*标号*] RETLW k			
操作数：	0 ≤ k ≤ 255			
操作：	k → (W)， (TOS) → PC，			
受影响的状态位：	无			
说明：	将 8 位立即数 k 装入 W 寄存器。 将栈顶内容（返回地址）装入程序计数器。 该指令是双周期指令。			
指令字数：	1			
指令周期数：	2			

示例：

```
      CALL TABLE            ; W contains table
; offset value
; W now has
; table value
   :
TABLE
ADDWF PC              ; W = offset
RETLW k1              ; Begin table
RETLW k2              ;
:
:
RETLW kn              ; End of table
```

执行指令前 W = 07h 执行指令后 W = k8 的值

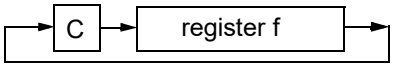
RETURN	从子程序返回			
语法：	[*标号*] RETURN			
操作数：	无			
操作：	(TOS) → PC，			
受影响的状态位：	无			
指令编码：	0000	0000	0001	001s
说明：	从子程序返回。 执行出栈操作，将栈顶（TOS）内容装入程序计数器。 该指令是双周期指令。			

RLF	对 f 执行带进位的循环左移			
语法：	[*标号*] RLF f, d			
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]			
操作：	(f[n]) → dest[n + 1]， (f[7]) → C， (C) → dest[0]			
受影响的状态位：	C			
指令编码：	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，结果存入 W 寄存器。 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			

示例：	`RLF`	`REG1, 0`
执行指令前 REG1 = `1110 0110` C = `0` 执行指令后 REG = `1110 0110` W = `1100 1100` C = `1`		

RRF	对 f 执行带进位的循环右移			
语法：	[*标号*] RRF f, d			
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]			
操作：	(f[n]) → dest[n － 1]， (f[0]) → C， (C) → dest[7]			

标准指令集
（续）

RRF	对 f 执行带进位的循环右移
受影响的状态位：	C
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，结果存入 W 寄存器。 如果 d 为 1，结果存回寄存器 f（默认）。 

SLEEP	进入休眠模式
语法：	[*标号*] SLEEP
操作数：	无
操作：	00h → WDT， 0 → WDT 预分频器， 1 →$\overline{TO}$， 0 →$\overline{PD}$
受影响的状态位：	$\overline{TO}$ 和 $\overline{PD}$
说明：	掉电（$\overline{PD}$）状态位清零。 超时（$\overline{TO}$）状态位置 1。 看门狗定时器及其预分频器被清零。

SUBLW	从立即数中减去 W 的内容
语法：	[*标号*] SUBLW k
操作数：	0 ≤ k ≤ 255
操作：	k – (W) → (W)
受影响的状态位：	C、DC 和 Z
说明	用 8 位立即数 k 减去 W 寄存器的内容（通过二进制补码方式进行运算）。 结果存入 W 寄存器。 C = 0，W > k C = 1，W ≤ k DC = 0，W[3:0] > k[3:0] DC = 1，W[3:0] ≤ k[3:0]

SUBWF	f 减去 W
语法：	[*标号*] SUBWF f, d
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]
操作：	(f) - (W) → (目标寄存器)
受影响的状态位：	C、DC 和 Z
说明	用寄存器 f 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，结果存入 W 寄存器。 如果 d 为 1，结果存回寄存器 f。 C = 0，W > f C = 1, W ≤ f DC = 0, W[3:0] > f[3:0] DC = 1, W[3:0] ≤ f[3:0]

SUBWFB	用 f 的内容减去 W 的内容（带借位）
语法：	[*标号*] SUBFWB f {,d}
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]
操作：	(W) – (f) – ($\overline{B}$) → 目标寄存器
受影响的状态位：	C、DC 和 Z
说明：	从寄存器 f 中减去 W 的内容以及借位（进位）标志位（采用二进制补码方法进行运算）。 如果 d 为 0，结果存入 W 寄存器。 如果 d 为 1，结果存回寄存器 f。

SWAPF	将 f 中的两个半字节进行交换
语法：	[*标号*] SWAPF f, d
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]
操作：	(f[3:0]) → dest[7:4]， (f[7:4]) → dest[3:0]
受影响的状态位：	无
说明：	将寄存器 f 的高半字节和低半字节交换。 如果 d 为 0，结果存入 W 寄存器。 如果 d 为 1，结果存入寄存器 f（默认）。

TRIS	将 W 寄存器的内容装入 TRIS 寄存器
语法：	[*标号*] TRIS f
操作数：	5 ≤ f ≤ 7
操作：	(W) → TRIS 寄存器 f
受影响的状态位：	无
说明：	将 W 寄存器中的数据传送到 TRIS 寄存器。 当 f = 5 时，装入 TRISA。 当 f = 6 时，装入 TRISB。 当 f = 7 时，装入 TRISC。

XORLW	立即数与 W 作逻辑异或运算
语法：	[*标号*] XORLW k
操作数：	0 ≤ k ≤ 255
操作：	(W) .XOR. k → (W)
受影响的状态位：	Z
说明：	将 W 的内容与 8 位立即数 k 进行逻辑异或运算。 结果存入 W 寄存器。

XORWF	W 与 f 作逻辑异或运算
语法：	[*标号*] XORWF f, d
操作数：	0 ≤ f ≤ 127 d ∈ [0,1]
操作：	(W) .XOR.(f) → 目标寄存器
受影响的状态位：	Z
说明：	将 W 寄存器的内容与寄存器 f 的内容进行逻辑异或运算。 如果 d 为 0，结果存入 W 寄存器。 如果 d 为 1，结果存回寄存器 f。

27. 编程

编程可以在组装流程之后完成，从而可以使用最新版本的固件或者定制固件对器件编程。编程需要 5 个引脚：

- CLK
- DAT
- $\overline{MCLR}/V_{PP}$
- V_{DD}
- V_{SS}

在编程/校验模式下，通过串行通信对程序存储器、用户 ID 和配置位进行编程。DAT 引脚是用于传输串行数据的双向 I/O，CLK 引脚是时钟输入引脚。

27.1. 高电压编程进入模式

通过将 CLK 和 DAT 引脚保持为低电平，然后将 $\overline{MCLR}/V_{PP}$ 引脚上的电压升至 V_{IH}，可将器件置于高电压编程模式。

27.2. 低电压编程进入模式

低电压编程（Low-Voltage Programming，LVP）进入模式允许器件在没有高电压的情况下仅使用 V_{DD} 进行编程。当 LVP 配置位设置为 1 时，将会使能 LVP 进入模式。要禁止低电压模式，必须将 LVP 位编程为 0。

进入 LVP 进入模式需要以下步骤：

1. 将 $\overline{MCLR}$ 拉至 V_{IL}。
2. 在 DAT 引脚上输出 32 位密钥序列，在 CLK 引脚上输出时钟。

在密钥序列完成后，$\overline{MCLR}$ 必须保持在 V_{IL} 电压才能维持编程/校验模式下。

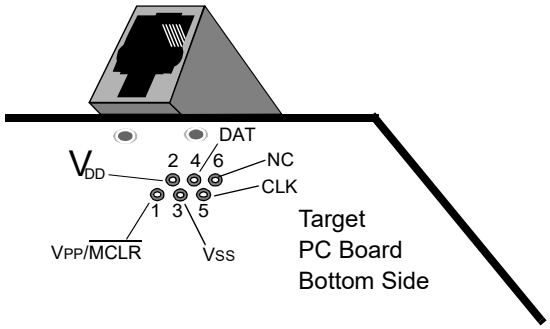
如果使能了 LVP（LVP = 1），将自动使能 $\overline{MCLR}$ 复位功能，且无法禁止。更多信息，请参见 *$\overline{MCLR}$* 部分。

只能通过使用高电压编程模式将 LVP 位重新编程为 0。

27.3. 通用编程接口

与目标器件的连接通常通过采用 6P6C（6 个引脚，6 个连接器）配置的 RJ-11 连接器完成。请参见图 27-1。

图 27-1. RJ-11 型连接器接口



引脚说明

1 = V_{PP}/$\overline{MCLR}$

2 = V_{DD} 目标

3 = V_{SS}（地）

4 = DAT

5 = CLK

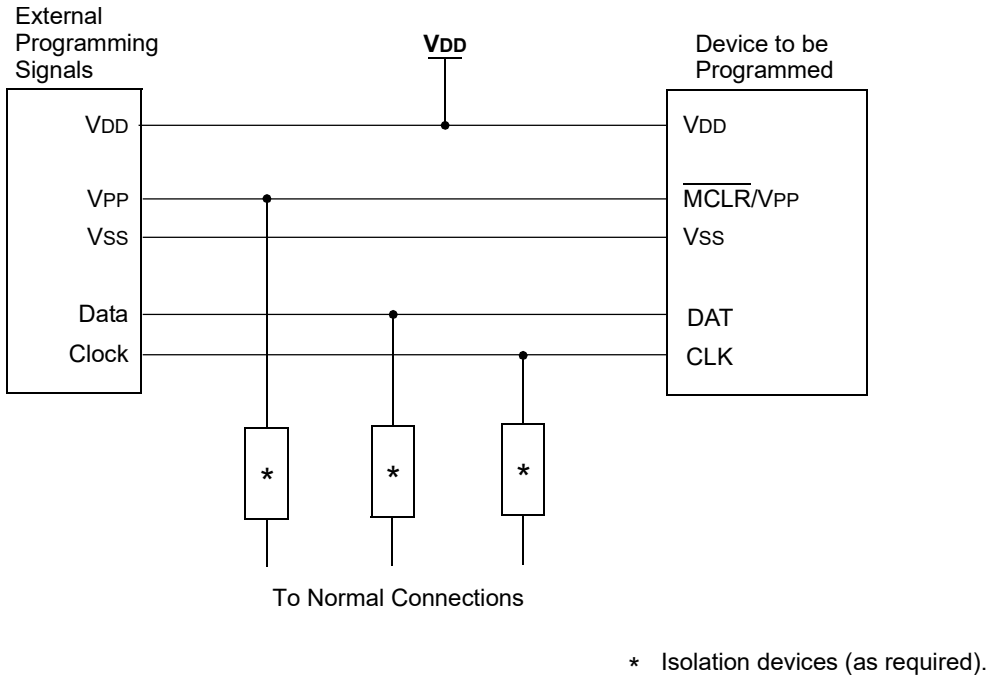
6 = 无连接

建议使用隔离器件将编程引脚与其他电路隔离。隔离类型主要取决于具体应用，可能包括电阻、二极管甚至跳线之类的器件。更多信息，请参见图 27-2。

注：

1. 6 引脚连接头（0.100"间距）可使用 0.025"方形引脚。

图 27-2. 编程的典型连接



* Isolation devices (as required).

28. 寄存器汇总

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00	INDF0	7:0	INDF0[7:0]							
0x01	INDF1	7:0	INDF1[7:0]							
0x02	PCL	7:0	PCL[7:0]							
0x03	STATUS	7:0				$\overline{TO}$	$\overline{PD}$	Z	DC	C
0x04	FSR0	7:0	FSR0[7:0]							
		15:8	FSR0[15:8]							
0x06	FSR1	7:0	FSR1[7:0]							
		15:8	FSR1[15:8]							
0x08	BSR	7:0			BSR[5:0]					
0x09	WREG	7:0	WREG[7:0]							
0x0A	PCLATH	7:0		PCLATH[6:0]						
0x0B	保留									
0x0C	PORTA	7:0			RA5	RA4	RA3	RA2	RA1	RA0
0x0D	保留									
0x0E	PORTC	7:0			RC5	RC4	RC3	RC2	RC1	RC0
0x0F ... 0x11	保留									
0x12	TRISA	7:0			TRISA5	TRISA4	保留	TRISA2	TRISA1	TRISA0
0x13	保留									
0x14	TRISC	7:0			TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
0x15 ... 0x17	保留									
0x18	LATA	7:0			LATA5	LATA4		LATA2	LATA1	LATA0
0x19	保留									
0x1A	LATC	7:0			LATC5	LATC4	LATC3	LATC2	LATC1	LATC0
0x1B ... 0x99	保留									
0x9A	CPCON	7:0	CPON[1:0]						CPT	CPRDY
0x9B	ADRES	7:0	ADRES[7:0]							
		15:8	ADRES[15:8]							
0x9D	ADCON0	7:0	CHS[5:0]						GO	ON
0x9E	ADCON1	7:0	FM	CS[2:0]					PREF[1:0]	
0x9F	ADACT	7:0					ACT[3:0]			
0xA0 ... 0x010D	保留									
0x010E	RC0I2C	7:0		SLEW	PU[1:0]				TH[1:0]	
0x010F	RC1I2C	7:0		SLEW	PU[1:0]				TH[1:0]	
0x0110 ... 0x0118	保留									
0x0119	RC1REG	7:0	RCREG[7:0]							
0x011A	TX1REG	7:0	TXREG[7:0]							
0x011B	SP1BRG	7:0	SPBRG[7:0]							
		15:8	SPBRG[15:8]							
0x011D	RC1STA	7:0	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
0x011E	TX1STA	7:0	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
0x011F	BAUD1CON	7:0	ABDOVF	RCIDL		SCKP	BRG16		WUE	ABDEN
0x0120 ... 0x018B	保留									
0x018C	SSP1BUF	7:0	BUF[7:0]							
0x018D	SSP1ADD	7:0	ADD[7:0]							
0x018E	SSP1MSK	7:0	MSK[6:0]							MSK0
0x018F	SSP1STAT	7:0	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF

寄存器汇总（续）

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x0190	SSP1CON1	7:0	WCOL	SSPOV	SSPEN	CKP	SSPM[3:0]			
0x0191	SSP1CON2	7:0	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
0x0192	SSP1CON3	7:0	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
0x0193 ... 0x020B	保留									
0x020C	TMR1	7:0	TMR1[7:0]							
		15:8	TMR1[15:8]							
0x020E	T1CON	7:0			CKPS[1:0]			SYNC	RD16	ON
0x020F	T1GCON	7:0	GE	GPOL	GTM	GSPM	GGO/DONE	GVAL		
0x0210	T1GATE	7:0				GSS[4:0]				
0x0211	T1CLK	7:0				CS[4:0]				
0x0212 ... 0x028B	保留									
0x028C	T2TMR	7:0	T2TMR[7:0]							
0x028D	T2PR	7:0	T2PR[7:0]							
0x028E	T2CON	7:0	ON		CKPS[2:0]		OUTPS[3:0]			
0x028F	T2HLT	7:0	PSYNC	CPOL	CSYNC	MODE[4:0]				
0x0290	T2CLKCON	7:0						CS[2:0]		
0x0291	T2RST	7:0					RSEL[3:0]			
0x0292 ... 0x030B	保留									
0x030C	CCPR1	7:0	CCPR[7:0]							
		15:8	CCPR[15:8]							
0x030E	CCP1CON	7:0	EN		OUT	FMT	MODE[3:0]			
0x030F	CCP1CAP	7:0							CTS[1:0]	
0x0310	CCPR2	7:0	CCPR[7:0]							
		15:8	CCPR[15:8]							
0x0312	CCP2CON	7:0	EN		OUT	FMT	MODE[3:0]			
0x0313	CCP2CAP	7:0							CTS[1:0]	
0x0314	PWM3DC	7:0	DCL[1:0]							
		15:8	DCH[7:0]							
0x0316	PWM3CON	7:0	EN		OUT	POL				
0x0317	保留									
0x0318	PWM4DC	7:0	DCL[1:0]							
		15:8	DCH[7:0]							
0x031A	PWM4CON	7:0	EN		OUT	POL				
0x031B ... 0x059B	保留									
0x059C	TMR0L	7:0	TMR0L[7:0]							
0x059D	TMR0H	7:0	TMR0H[7:0]							
0x059E	T0CON0	7:0	EN		OUT	MD16	OUTPS[3:0]			
0x059F	T0CON1	7:0	CS[2:0]			ASYNC	CKPS[3:0]			
0x05A0 ... 0x070B	保留									
0x070C	PIR0	7:0			TMR0IF	IOCIF				INTF
0x070D	PIR1	7:0	CCP1IF	TMR2IF	TMR1IF	RC1IF	TX1IF	BCL1IF	SSP1IF	ADIF
0x070E	PIR2	7:0	CCP2IF	NVMIF	TMR1GIF					
0x070F ... 0x0715	保留									
0x0716	PIE0	7:0			TMR0IE	IOCIE				INTE
0x0717	PIE1	7:0	CCP1IE	TMR2IE	TMR1IE	RC1IE	TX1IE	BCL1IE	SSP1IE	ADIE
0x0718	PIE2	7:0	CCP2IE	NVMIE	TMR1GIE					
0x0719 ... 0x080B	保留									

寄存器汇总（续）

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x080C	WDTCON	7:0	CS		PS[4:0]					SEN
0x080D ... 0x0810	保留									
0x0811	BORCON	7:0	SBOREN							BORRDY
0x0812	保留									
0x0813	PCON0	7:0	STKOVF	STKUNF		$\overline{\text{RWDT}}$	$\overline{\text{RMCLR}}$	$\overline{\text{RI}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
0x0814	PCON1	7:0							$\overline{\text{MEMV}}$	
0x0815 ... 0x088D	保留									
0x088E	OSCCON	7:0			COSC[1:0]					
0x088F	保留									
0x0890	OSCSTAT	7:0		HFOR	MFOR	LFOR		ADOR	SFOR	
0x0891	OSCEN	7:0		HFOEN	MFOEN	LFOEN		ADOEN		
0x0892	OSCTUNE	7:0			TUN[5:0]					
0x0893	OSCFRQ	7:0						FRQ[2:0]		
0x0894 ... 0x1C8B	保留									
0x1C8C	NVMADR	7:0	NVMADR[7:0]							
		15:8		NVMADR[14:8]						
0x1C8E	NVMDAT	7:0	NVMDAT[7:0]							
		15:8			NVMDAT[13:8]					
0x1C90	NVMCON1	7:0		NVMREGS	LWLO	FREE	WRERR	WREN	WR	RD
0x1C91	NVMCON2	7:0	NVMCON2[7:0]							
0x1C92 ... 0x1E8E	保留									
0x1E8F	PPSLOCK	7:0								PPSLOCKED
0x1E90	INTPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1E91	T0CKIPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1E92	T1CKIPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1E93	T1GPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1E94 ... 0x1E9B	保留									
0x1E9C	T2INPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1E9D ... 0x1EA0	保留									
0x1EA1	CCP1PPS	7:0			PORT[2:0]			PIN[2:0]		
0x1EA2	CCP2PPS	7:0			PORT[2:0]			PIN[2:0]		
0x1EA3 ... 0x1EC2	保留									
0x1EC3	ADACTPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1EC4	保留									
0x1EC5	SSP1CLKPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1EC6	SSP1DATPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1EC7	SSP1SSPPS	7:0			PORT[2:0]			PIN[2:0]		
0x1EC8 ... 0x1ECA	保留									
0x1ECB	RX1PPS	7:0			PORT[2:0]			PIN[2:0]		
0x1ECC	CK1PPS	7:0			PORT[2:0]			PIN[2:0]		
0x1ECD ... 0x1F0F	保留									
0x1F10	RA0PPS	7:0			RA0PPS[5:0]					
0x1F11	RA1PPS	7:0			RA1PPS[5:0]					

寄存器汇总（续）

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x1F12	RA2PPS	7:0			RA2PPS[5:0]					
0x1F13	保留									
0x1F14	RA4PPS	7:0			RA4PPS[5:0]					
0x1F15	RA5PPS	7:0			RA5PPS[5:0]					
0x1F16 ... 0x1F1F	保留									
0x1F20	RC0PPS	7:0			RC0PPS[5:0]					
0x1F21	RC1PPS	7:0			RC1PPS[5:0]					
0x1F22	RC2PPS	7:0			RC2PPS[5:0]					
0x1F23	RC3PPS	7:0			RC3PPS[5:0]					
0x1F24	RC4PPS	7:0			RC4PPS[5:0]					
0x1F25	RC5PPS	7:0			RC5PPS[5:0]					
0x1F26 ... 0x1F37	保留									
0x1F38	ANSELA	7:0			ANSELA5	ANSELA4		ANSELA2	ANSELA1	ANSELA0
0x1F39	WPUA	7:0			WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
0x1F3A	ODCONA	7:0			ODCA5	ODCA4		ODCA2	ODCA1	ODCA0
0x1F3B	SLRCONA	7:0			SLRA5	SLRA4		SLRA2	SLRA1	SLRA0
0x1F3C	INLVLA	7:0			INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0
0x1F3D	IOCAP	7:0			IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0
0x1F3E	IOCAN	7:0			IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0
0x1F3F	IOCAF	7:0			IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0
0x1F40 ... 0x1F4D	保留									
0x1F4E	ANSELC	7:0			ANSELC5	ANSELC4	ANSELC3	ANSELC2	ANSELC1	ANSELC0
0x1F4F	WPUC	7:0			WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
0x1F50	ODCONC	7:0			ODCC5	ODCC4	ODCC3	ODCC2	ODCC1	ODCC0
0x1F51	SLRCONC	7:0			SLRC5	SLRC4	SLRC3	SLRC2	SLRC1	SLRC0
0x1F52	INLVLC	7:0			INLVLC5	INLVLC4	INLVLC3	INLVLC2	INLVLC1	INLVLC0
0x1F53	IOCCP	7:0			IOCCP5	IOCCP4	IOCCP3	IOCCP2	IOCCP1	IOCCP0
0x1F54	IOCCN	7:0			IOCCN5	IOCCN4	IOCCN3	IOCCN2	IOCCN1	IOCCN0
0x1F55	IOCCF	7:0			IOCCF5	IOCCF4	IOCCF3	IOCCF2	IOCCF1	IOCCF0
0x1F56 ... 0x8004	保留									
0x8005	REVISIONID	7:0	MJRREV[1:0]		MNRREV[5:0]					
		15:8			保留	保留	MJRREV[5:2]			
0x8006	DEVICEID	7:0	DEV[7:0]							
		15:8			保留	保留	DEV[11:8]			
0x8007	CONFIG1	7:0			RSTOSC[1:0]				FEXTOSC[1:0]	
		15:8				VDDAR				$\overline{\text{CLKOUTEN}}$
0x8008	CONFIG2	7:0	BOREN[1:0]			WDTE[1:0]		PWRTS[1:0]		MCLRE
		15:8			$\overline{\text{DEBUG}}$	STVREN	PPS1WAY		BORV	
0x8009	CONFIG3	7:0								
		15:8								
0x800A	CONFIG4	7:0	$\overline{\text{WRTAPP}}$			$\overline{\text{SAFEN}}$	$\overline{\text{BBEN}}$	BBSIZE[2:0]		
		15:8			LVP		$\overline{\text{WRTSAF}}$		$\overline{\text{WRTC}}$	$\overline{\text{WRTB}}$
0x800B	CONFIG5	7:0								$\overline{\text{CP}}$
		15:8								

29. 电气规范

29.1. 绝对最大额定值

表 29-1.

参数		额定值
偏置时的环境温度		-40°C 至+85°C
储存温度		-65°C 至+150°C
引脚相对于 V_{SS} 的电压		
• **V_{DD} 引脚：**		-0.3V 至+6.5V
• **$\overline{MCLR}$ 引脚：**		-0.3V 至+9.0V
• **其他所有引脚：**		-0.3V 至（V_{DD} + 0.3V）
最大电流		
• **V_{SS} 引脚(1)**	-40°C ≤ T_A ≤ +85°C	300 mA
• **V_{DD} 引脚(1)**	-40°C ≤ T_A ≤ +85°C	250 mA
• **任何标准 I/O 引脚**		±25 mA
钳位电流，I_K（V_{PIN} < 0 或 V_{PIN} > V_{DD}）		±20 mA
总功耗(2)		800 mW

注：

1. 最大额定电流要求 I/O 引脚上具有均匀的负载分布。最大额定电流可以通过器件封装功耗特性进行限制，请参见热特性来计算器件规范值。
2. 功耗按如下公式计算：
   $P_{DIS} = V_{DD} \times \{I_{DD} - \Sigma\ I_{OH}\} + \Sigma\ \{(V_{DD} - V_{OH}) \times I_{OH}\} + \Sigma\ (V_{OI} \times I_{OL})$
3. 内部功耗按如下公式计算：
   $P_{INTERNAL} = I_{DD} \times V_{DD}$

   其中 I_{DD} 为输出引脚上不驱动任何负载时使芯片独立运行的电流。
4. I/O 功耗按如下公式计算：
   $P_{I/O} = \Sigma(I_{OL}*V_{OL}) + \Sigma(I_{OH}*(V_{DD}-V_{OH}))$
5. 降额功耗按如下公式计算：
   $P_{DER} = PD_{MAX}(T_J-T_A)/\theta_{JA}$

   其中 T_A = 环境温度，T_J = 结温。

NOTICE 如果器件工作条件超过上述*绝对最大额定值*，可能对器件造成永久性损坏。上述值仅代表极限工作条件，未表明器件处于或超过本规范指定的极限值条件下仍可正常工作。器件长时间工作在最大值条件下，其可靠性可能受到影响。

29.2. 标准工作条件

器件的标准工作条件定义如下：

参数	条件
工作电压：	$V_{DDMIN} \leqslant V_{DD} \leqslant V_{DDMAX}$
工作温度：	$T_{AMIN} \leqslant T_A \leqslant T_{AMAX}$

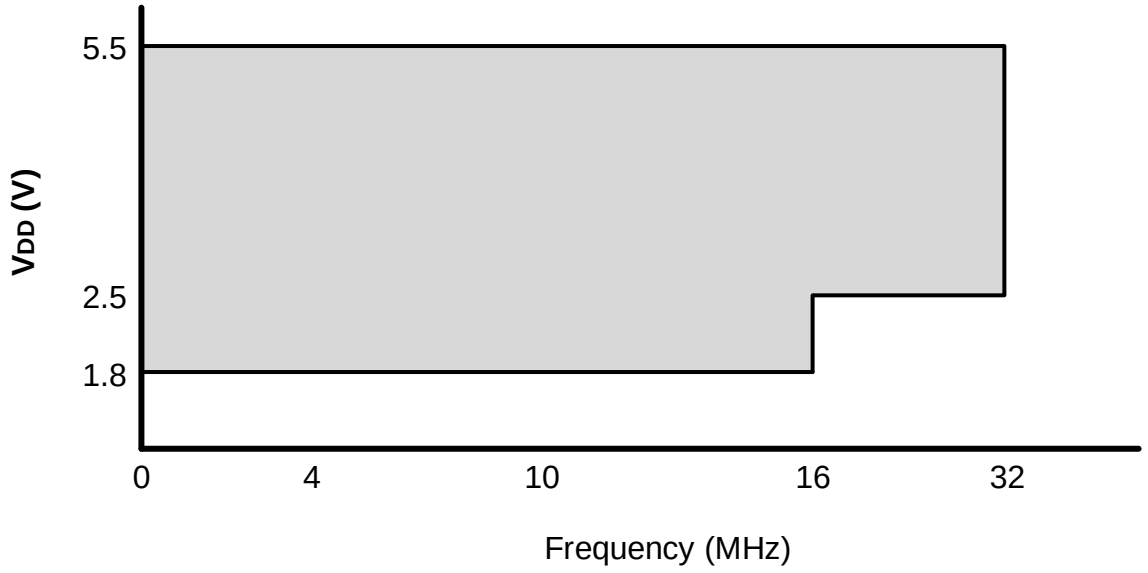
表 29-2.

参数		额定值
V_{DD}——工作电源电压(1)		
	V_{DDMIN}（$F_{OSC} \leqslant$ 16 MHz）	+1.8V
	V_{DDMIN}（$F_{OSC} \leqslant$ 32 MHz）	+2.5V
	V_{DDMAX}	+5.5V
T_A——工作环境温度范围		
工业级温度	T_{A_MIN}	-40°C
	T_{A_MAX}	+85°C

注：

1. 请参见参数 **D002**，直流特性：电源电压。

图 29-1. 电压—频率关系图，-40°C $\leqslant T_A \leqslant$ +85°C



注：

1. 阴影区域表示允许的电压和频率的组合。
2. 有关各个振荡器模式所支持的频率，请参见“**外部时钟/振荡器时序要求**”一节。

29.3. 直流特性

29.3.1. 电源电压

表 29-3.

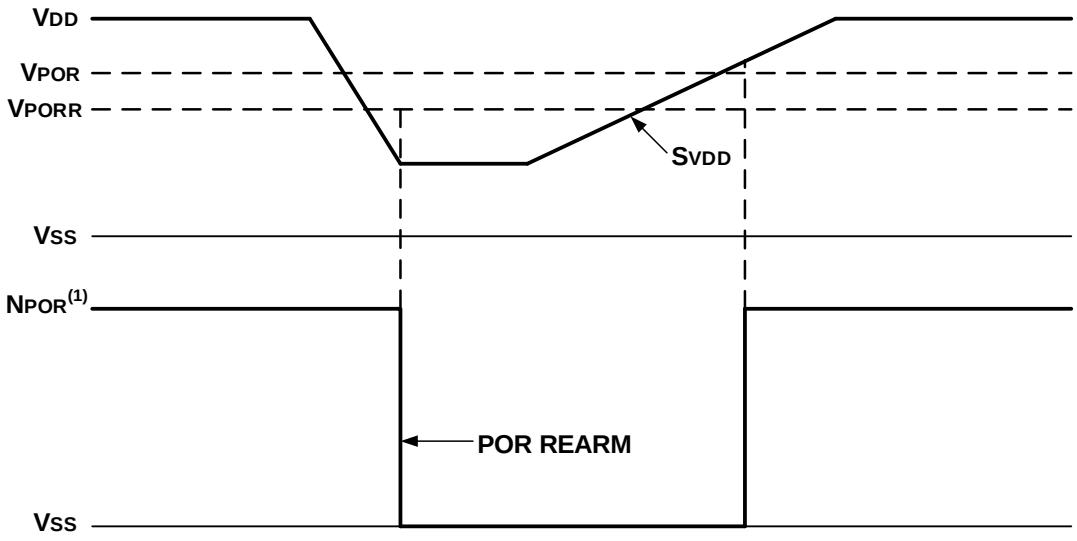
标准工作条件（除非另外声明）							
参数编号	**符号**	**特性**	**最小值**	**典型值†**	**最大值**	**单位**	**条件**
电源电压							
D002	V_{DD}		1.8	—	5.5	V	$F_{OSC} \leqslant 16$ MHz
			2.5	—	5.5	V	$F_{OSC} > 16$ MHz
RAM 数据保持电压(1)							
D003	V_{DR}		1.7	—	—	V	器件工作在休眠模式下
上电复位释放电压(2)							
D004	V_{POR}		—	1.6	—	V	禁止 BOR(3)
上电复位重新激活电压(2)							
D005	V_{PORR}		—	1.1	—	V	禁止 BOR(3)
确保内部上电复位信号的 V_{DD} 上升速率(2)							
D006	S_{VDD}		0.05	—	—	V/ms	禁止 BOR(3)

† - 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. 这是在确保不丢失 RAM 数据的前提下，休眠模式时 V_{DD} 所能达到的下限值。
2. 请参见下图：**V_{DD} 缓慢上升时，POR 和 POR 重新激活**。
3. 有关 BOR 跳变点的信息，请参见“**复位、WDT、上电延时定时器和欠压复位规范**”。

图 29-2. V_{DD} 缓慢上升时，POR 和 POR 重新激活



注：

1. 当 N_{POR} 为低电平时，器件保持在复位状态。

29.3.2. 供电电流（I_{DD}）(1,2)

表 29-4.

标准工作条件（除非另外声明）								
参数编号	符号	器件特性	最小值	典型值†	最大值	单位	条件	
							V_{DD}	注
D101	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	1.5	—	mA	3.0V	
D102	$I_{DD_{HFOPLL}}$	HFINTOSC = 32 MHz	—	2.7	—	mA	3.0V	

† - 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. 有效工作模式下，所有 I_{DD} 测量值的测试条件为：所有 I/O 引脚均为输出，驱动为低电平；$\overline{MCLR}$= V_{DD}；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、内部代码执行模式和温度也会对电流消耗产生影响。

29.3.3. 掉电电流（I_{PD}）(1,2,3)

表 29-5.

标准工作条件（除非另外声明）								
参数编号	符号	器件特性	最小值	典型值†	最大值+85°C	单位	条件	
							V_{DD}	注
D200	I_{PD}	基本 I_{PD} 电流	—	0.40	—	μA	3.0V	
D201	I_{PD_WDT}	低频内部振荡器/WDT	—	0.5	—	μA	3.0V	
D204	I_{PD_BOR}	欠压复位（BOR）	—	25	—	μA	3.0V	
D207	I_{PD_ADCA}	ADC——工作	—	3	—	μA	3.0V	ADC 未在进行转换 (4)

† - 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 I_{DD} 与该外设使能时所额外消耗的电流之和。可通过从该参数值中减去基本 I_{DD} 或 I_{PD} 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 V_{SS} 时测得的。
3. 如果多个外设可用，那么列出的所有外设电流均是基于每个外设的。
4. ADC 时钟源为 ADCRC。

29.3.4. I/O 端口

表 29-6.

标准工作条件（除非另外声明）							
参数编号	符号	器件特性	最小值	典型值†	最大值	单位	条件
输入低电平电压							

表 29-6.（续）

标准工作条件（除非另外声明）							
参数编号	**符号**	**器件特性**	**最小值**	**典型值†**	**最大值**	**单位**	**条件**
	V_{IL}	I/O 端口：					
D300		• 带 TTL 缓冲器	—	0.8	—	V	$4.5V \leqslant V_{DD} \leqslant 5.5V$
D301			—	0.15 V_{DD}	—	V	$1.8V \leqslant V_{DD} \leqslant 4.5V$
D302		• 带施密特触发缓冲器	—	0.2 V_{DD}	—	V	$1.8V \leqslant V_{DD} \leqslant 5.5V$
D303		• 带 I^2C 电平	—	0.3 V_{DD}	—	V	$2.0V \leqslant V_{DD} \leqslant 5.5V$
D304		• 带 SMBus 电平	—	0.8	—	V	$2.7V \leqslant V_{DD} \leqslant 5.5V$
D305		$\overline{MCLR}$	—	0.2 V_{DD}	—	V	
输入高电平电压							
	V_{IH}	I/O 端口：					
D320		• 带 TTL 缓冲器	—	2.0	—	V	$4.5V \leqslant V_{DD} \leqslant 5.5V$
D321			—	0.25 V_{DD}+0.8	—	V	$1.8V \leqslant V_{DD} \leqslant 4.5V$
D322		• 带施密特触发缓冲器	—	0.8V_{DD}	—	V	$1.8V \leqslant V_{DD} \leqslant 5.5V$
D323		• 带 I^2C 电平	—	0.7 V_{DD}	—	V	
D324		• 带 SMBus 电平	—	2.1	—	V	$2.7V \leqslant V_{DD} \leqslant 5.5V$
D325		$\overline{MCLR}$	—	0.8 V_{DD}	—	V	
输入泄漏电流(1)							
D340	I_{IL}	I/O 端口	—	±5	—	nA	$V_{SS} \leqslant V_{PIN} \leqslant V_{DD}$，引脚处于高阻态，85°C
D341			—	±5	—	nA	$V_{SS} \leqslant V_{PIN} \leqslant V_{DD}$，引脚处于高阻态，125°C
D342		$\overline{MCLR}$(2)	—	±50	—	nA	$V_{SS} \leqslant V_{PIN} \leqslant V_{DD}$，引脚处于高阻态，85°C
弱上拉电流							
D350	I_{PUR}		—	120	—	μA	V_{DD} = 3.0V，$V_{PIN} = V_{SS}$
输出低电平电压							
D360	V_{OL}	标准 I/O 端口	—	0.6	—	V	I_{OL} = 10 mA，V_{DD} = 3.0V
输出高电平电压							
D370	V_{OH}	标准 I/O 端口	—	V_{DD}-0.7	—	V	I_{OH} = 6 mA，V_{DD} = 3.0V
所有 I/O 引脚							
D380	C_{IO}		—	5	—	pF	

† - 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. 负电流定义为引脚的拉电流。
2. $\overline{MCLR}$ 引脚上的泄漏电流主要取决于所施加的电压。规定电压为正常工作条件下的电压。在不同的输入电压下可能会测得更高的泄漏电流。

29.3.5. 存储器编程规范

表 29-7.

标准工作条件（除非另外声明）							
参数编号	**符号**	**器件特性**	**最小值**	**典型值†**	**最大值**	**单位**	**条件**
高电压进入编程模式规范							
MEM01	V_{IHH}	进入编程模式的 $\overline{MCLR}/V_{PP}$ 引脚电压	7.9	—	9	V	**（注 2）**
MEM02	I_{PPGM}	编程模式期间 $\overline{MCLR}/V_{PP}$ 引脚上的电流	—	—	0.6	mA	**（注 2）**
编程模式规范							
MEM10	V_{BE}	用于批量擦除的 V_{DD}	—	2.9	—	V	**（注 3）**

表 29-7.（续）

标准工作条件（除非另外声明）							
参数编号	**符号**	**器件特性**	**最小值**	**典型值†**	**最大值**	**单位**	**条件**
MEM11	I_{DDPGM}	编程操作时的电源电流	—	—	10	mA	
闪存程序存储器规范							
MEM30	E_P	闪存存储器单元耐擦写能力	10k	—	—	E/W	-40°C ≤ T_A ≤ +85°C **（注 1）**
MEM32	T_{P_RET}	特性保持时间	—	40	—	年	假设未违反其他规范
MEM33	V_{P_RD}	用于读操作的 V_{DD}	V_{DDMIN}	—	V_{DDMAX}	V	
MEM34	V_{P_REW}	用于行擦除/写操作的 V_{DD}	V_{DDMIN}	—	V_{DDMAX}	V	
MEM35	T_{P_REW}	自定时行擦除或自定时写周期	—	2.8	—	ms	

† - 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. 闪存存储器单元耐擦写能力定义为：一次行擦除操作和一次自定时写操作。
2. 仅当配置字的 LVP 位禁止时需要。
3. 更多信息，请参见系列编程规范。

29.3.6. 热特性

表 29-8.

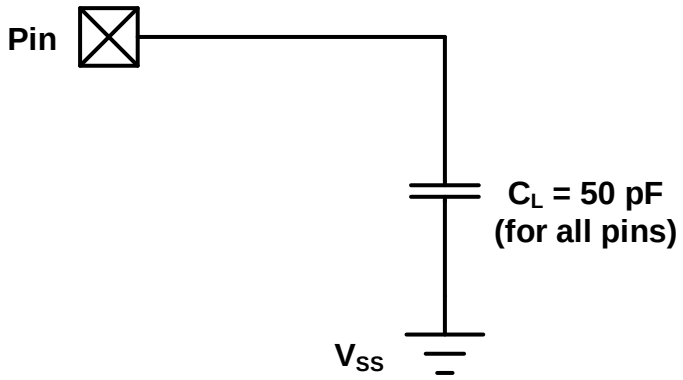
标准工作条件（除非另外声明） 工作温度：-40°C ≤ T_A ≤ +85°C					
参数编号	**符号**	**特性**	**典型值**	**单位**	**条件**
TH01	θ_{JA}	结点到环境的热阻	100.0	°C/W	14 引脚 TSSOP 封装
TH03	T_{JMAX}	最高结温	150	°C	
TH04	PD	功耗	—	W	PD = $P_{INTERNAL}$+$P_{I/O}$
TH05	$P_{INTERNAL}$	内部功耗	—	W	$P_{INTERNAL}$ = I_{DD} x V_{DD} **(1)**
TH06	$P_{I/O}$	I/O 功耗	—	W	$P_{I/O}$ = $\Sigma(I_{OL}*V_{OL})+\Sigma(I_{OH}*(V_{DD}-V_{OH}))$
TH07	P_{DER}	降额功耗	—	W	P_{DER} = PD_{MAX} $(T_J-T_A)/\theta_{JA}$ **(2)**

注：

1. I_{DD} 为输出引脚上不驱动任何负载时使芯片独立运行的电流。
2. T_A = 环境温度，T_J = 结温。

29.4. 交流特性

图 29-3. 负载条件



29.4.1. 外部时钟/振荡器时序要求

图 29-4. 时钟时序

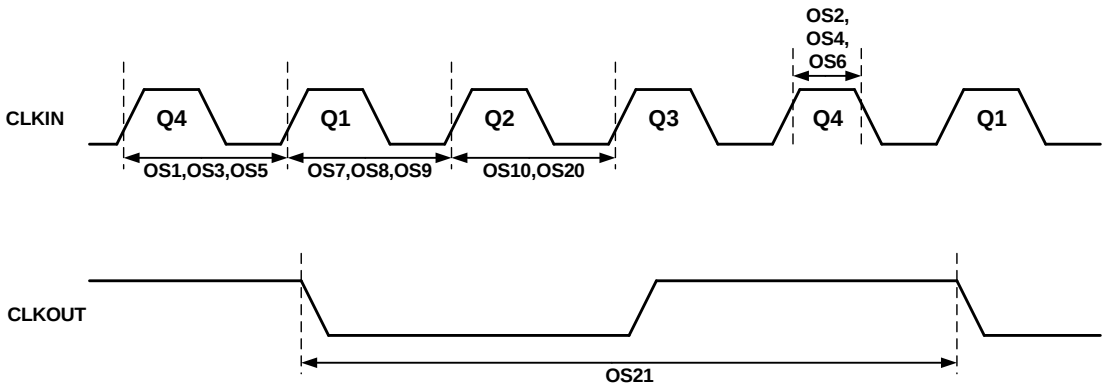


表 29-9.

标准工作条件（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**
ECL 振荡器							
OS1	F_{ECL}	时钟频率	—	—	16	MHz	
OS2	T_{ECL_DC}	时钟占空比	40	—	60	%	
ECH 振荡器							
OS5	F_{ECH}	时钟频率	—	—	32	MHz	
OS6	T_{ECH_DC}	时钟占空比	40	—	60	%	
系统振荡器							

表 29-9.（续）

标准工作条件（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**
OS20	F_{OSC}	系统时钟频率	—	—	32	MHz	**（注 2 和注 3）**
OS21	F_{CY}	指令频率	—	$F_{OSC}/4$	—	MHz	
OS22	T_{CY}	指令周期	125	$1/F_{CY}$	—	ns	**注 1**

***** 这些参数通过表征确定，但未经测试。
† - 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（T_{CY}）等于输入振荡器时基周期的四倍。所有规范值均基于器件在标准工作条件下执行代码时对应特定振荡器类型的表征数据。如果超出这些规定的限定值，可能导致振荡器运行不稳定和/或电流消耗超出预期值。所有器件在测试“最小”值时，都在 CLKIN 引脚上连接了外部时钟。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”（没有时钟）。
2. 系统时钟频率（F_{OSC}）通过“主时钟切换控制”选择，如“**OSC——振荡器模块**”一章所述。
3. 系统时钟频率（F_{OSC}）必须满足**“标准工作条件”**一节中定义的电压要求。

29.4.2. 内部振荡器参数(1)

表 29-10.

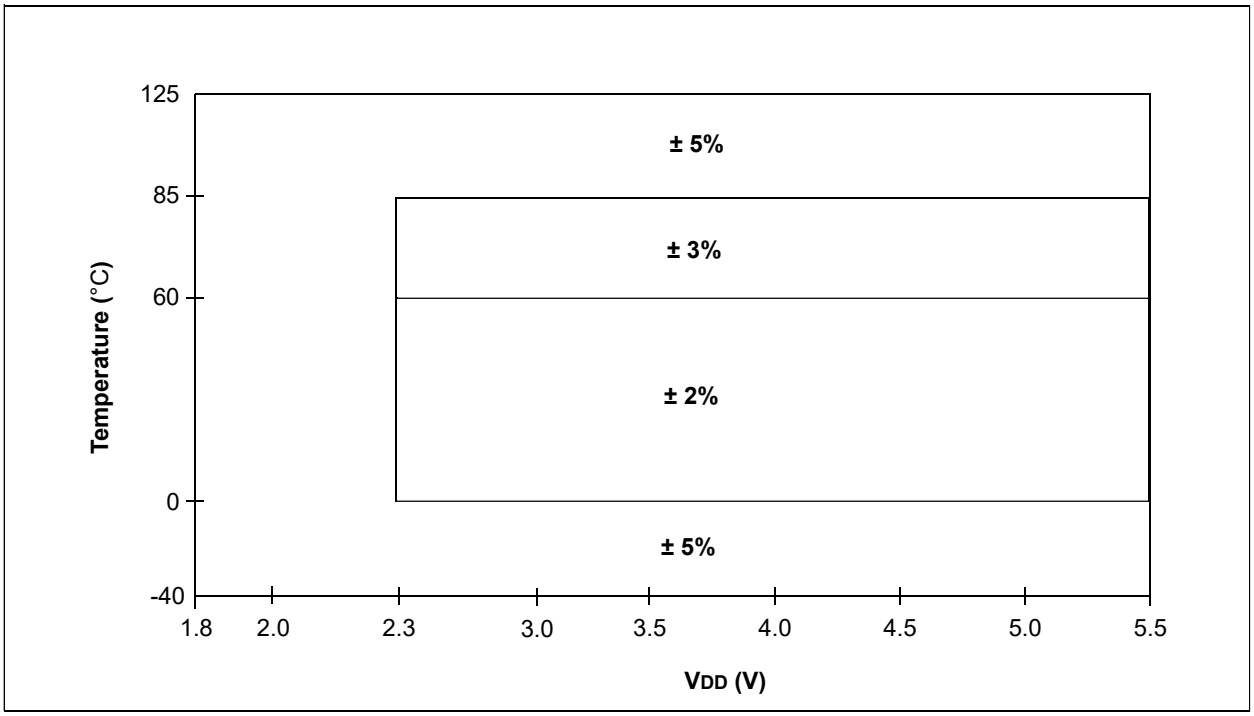
标准工作条件（除非另外声明）							
参数编号	**符号**	**特性**	**最小值**	**典型值†**	**最大值**	**单位**	**条件**
OS50	F_{HFOSC}	已校准精度的 HFINTOSC 频率	—	4 8 16 32	—	MHz	**注 2**
OS51	$F_{HFOSCLP}$	针对低功耗优化的 HFINTOSC 频率	— —	1 2	— —	MHz MHz	-40°C 至 85°C -40°C 至 85°C
OS52	F_{MFOSC}	经过校准的内部 MFINTOSC 频率	—	500	—	kHz	
OS53	F_{LFOSC}	内部 LFINTOSC 频率	—	31	—	kHz	
OS54	$T_{HFOSCST}$	HFINTOSC 从休眠模式唤醒的起振时间	—	19	—	µs	
OS56	$T_{LFOSCST}$	LFINTOSC 从休眠模式唤醒的起振时间	—	0.2	—	ms	

† - 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. 为了确保振荡器频率容差，必须在尽可能靠近器件的位置在 V_{DD} 和 V_{SS} 之间接去耦电容。建议并联 0.1 µF 和 0.01 µF 的电容。
2. 请参见**图 32-5**。

图 29-5. 器件 V_{DD} 和温度范围内的 HFINTOSC 频率精度（已校准）



29.4.3. I/O 和 CLKOUT 时序规范

图 29-6. CLKOUT 和 I/O 时序

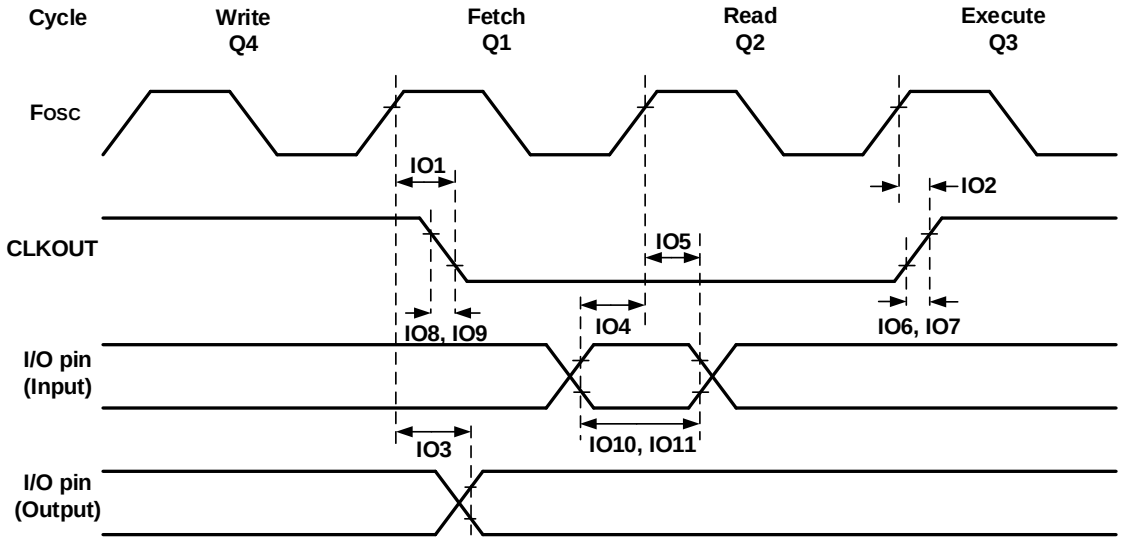
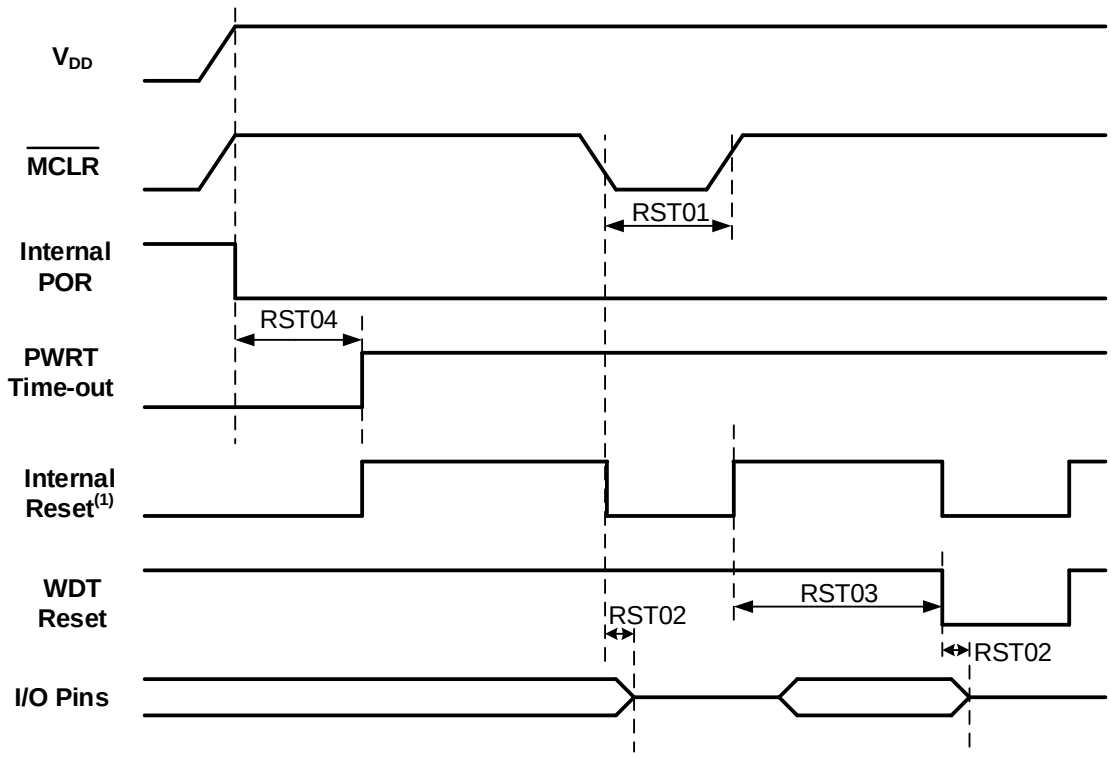


表 29-11. I/O 和 CLKOUT 时序规范

标准工作条件（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
IO1*	$T_{CLKOUTH}$	CLKOUT 上升沿延时（F_{OSC} 上升沿（Q1 周期）到 CLKOUT 下降沿的时间）	—	70	—	ns	
IO2*	$T_{CLKOUTL}$	CLKOUT 下降沿延时（F_{OSC} 上升沿（Q3 周期）到 CLKOUT 上升沿的时间）	—	72	—	ns	
IO3*	T_{IO_VALID}	端口输出有效时间（F_{OSC} 上升沿（Q1 周期）到端口有效的时间）	—	50	—	ns	
IO4*	T_{IO_SETUP}	端口输入建立时间（F_{OSC} 上升沿（Q2 周期）之前的建立时间）	—	20	—	ns	
IO5*	T_{IO_HOLD}	端口输入保持时间（F_{OSC} 上升沿（Q2 周期）之后的保持时间）	—	50	—	ns	
IO6*	T_{IOR_SLREN}	端口 I/O 上升时间，使能压摆率	—	25	—	ns	V_{DD} = 3.0V
IO7*	T_{IOR_SLRDIS}	端口 I/O 上升时间，禁止压摆率	—	5	—	ns	V_{DD} = 3.0V
IO8*	T_{IOF_SLREN}	端口 I/O 下降时间，使能压摆率	—	25	—	ns	V_{DD} = 3.0V
IO9*	T_{IOF_SLRDIS}	端口 I/O 下降时间，禁止压摆率	—	5	—	ns	V_{DD} = 3.0V
IO10*	T_{INT}	INT 引脚触发中断的高电平时间或低电平时间	—	25	—	ns	
IO11*	T_{IOC}	电平变化中断触发中断的最短高电平或低电平时间	—	25	—	ns	
* - 这些参数通过表征确定，但未经测试。							

29.4.4. 复位、WDT、上电延时定时器和欠压复位规范

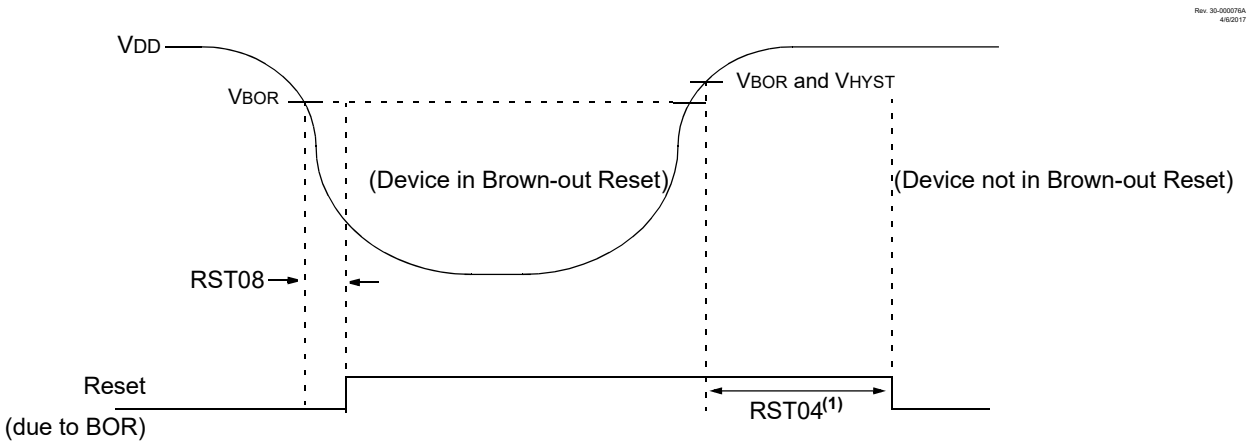
图 29-7. 复位、看门狗定时器和上电延时定时器时序



注：

1. 低电平有效。

图 29-8. 欠压复位时序和特性



注：

1. 仅在配置字寄存器中的 $\overline{\text{PWRTE}}$ 位被设置为 1 时；如果 $\overline{\text{PWRTE}}$= 0，则为 2 ms 延时。

表 29-12.

标准工作条件（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**
RST01*	T_{MCLR}	确保复位的 $\overline{\text{MCLR}}$ 脉冲宽度（低电平）	—	2	—	μs	
RST02*	T_{IOZ}	自检测到复位起 I/O 处于高阻态的时间	—	2	—	μs	
RST03	T_{WDT}	看门狗定时器超时周期	—	16	—	ms	1:512 预分频比
RST04*	T_{PWRT}	上电延时定时器周期	—	65	—	ms	PWRTS = 10（64 ms）
RST06	V_{BOR}	欠压复位电压	—	2.8	—	V	BORV = 0
			—	1.9	—	V	BORV = 1
RST07	V_{BORHYS}	欠压复位滞后	—	40	—	mV	
RST08	T_{BORDC}	欠压复位响应时间	—	3	—	μs	

* - 这些参数通过表征确定，但未经测试。

† 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。

29.4.5. 模数转换器（ADC）精度规范(1,2)

表 29-13.

标准工作条件（除非另外声明） V_{DD} = 3.0V，T_A = 25°C							
参数编号	**符号**	**特性**	**最小值**	**典型值†**	**最大值**	**单位**	**条件**
AD01	N_R	分辨率	—	—	10	位	
AD02	E_{IL}	积分误差	—	0.5	—	LSb	$ADC_{REF}+$ = 3.0V，$ADC_{REF}-$ = V_{SS}
AD03	E_{DL}	微分误差	—	0.3	—	LSb	$ADC_{REF}+$ = 3.0V，$ADC_{REF}-$ = V_{SS}
AD04	E_{OFF}	失调误差	—	1	—	LSb	$ADC_{REF}+$ = 3.0V，$ADC_{REF}-$ = V_{SS}
AD05	E_{GN}	增益误差	—	0.8	—	LSb	$ADC_{REF}+$ = 3.0V，$ADC_{REF}-$ = V_{SS}
AD06	V_{ADREF}	ADC 参考电压（$AD_{REF}+$）	1.8	—	V_{DD}	V	
AD07	V_{AIN}	满量程范围	V_{SS}	—	$AD_{REF}+$	V	
AD08	Z_{AIN}	模拟信号源的推荐阻抗	—	10	—	kΩ	
AD09	R_{VREF}	ADC 参考电压梯形阻抗	—	50	—	kΩ	

表 29-13.（续）

标准工作条件（除非另外声明） V_{DD} = 3.0V，T_A = 25°C							
参数编号	**符号**	**特性**	**最小值**	**典型值†**	**最大值**	**单位**	**条件**

* - 这些参数通过表征确定，但未经测试。

† 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. 总绝对误差是失调误差、增益误差和积分非线性（INL）误差的总和。
2. ADC 转换结果不会因输入的增加而减小，并且不会丢失编码。

29.4.6. 模数转换器（ADC）转换时序规范

表 29-14.

标准工作条件（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**
AD20	T_{AD}	ADC 时钟周期	0.5	—	9	μs	F_{OSC} 时钟源
AD21			1	2	6	μs	ADCRC 时钟源
AD22	T_{CNV}	转换时间	—	11	—	T_{AD}	GO 位置 1 到 GO 位清零的时间
AD23	T_{ACQ}	采集时间	—	2	—	μs	
AD24	T_{HCD}	采样和保持电容断开时间	—	—	—	—	F_{OSC} 时钟源

* - 这些参数通过表征确定，但未经测试。

† 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 29-9. ADC 转换时序（ADC 时钟基于 F_{OSC}）

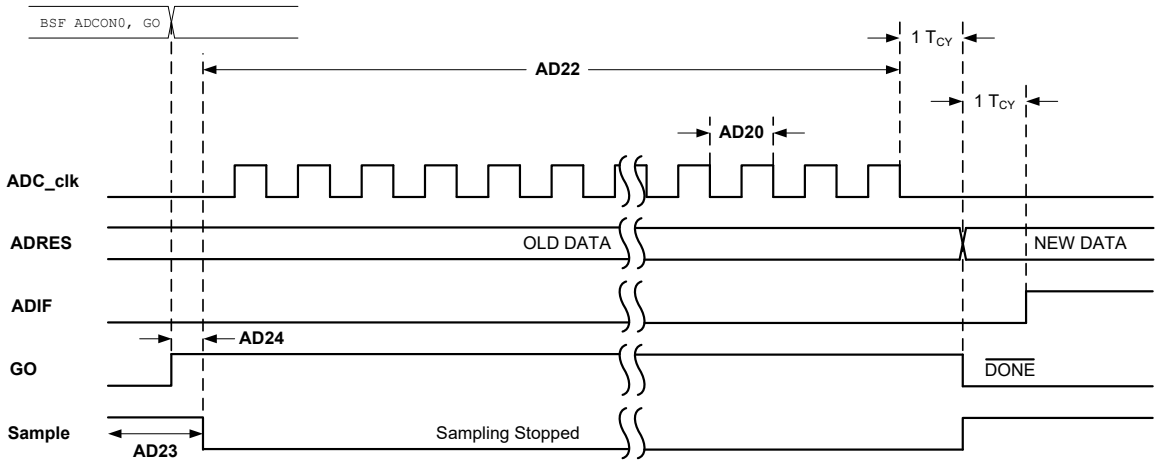
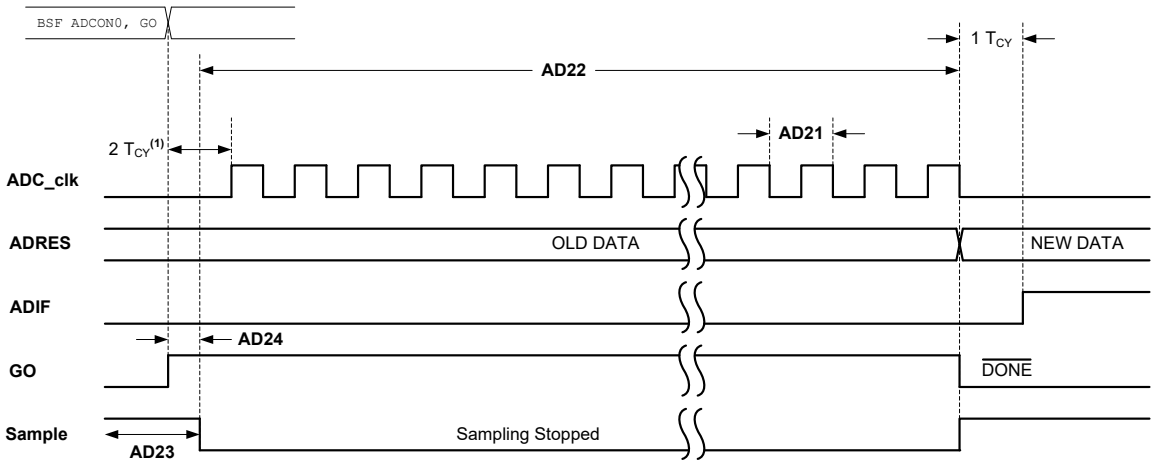


图 29-10. ADC 转换时序（ADC 时钟来自 ADCRC）



注：

1. 如果选择 ADCRC 作为 ADC 时钟源，在 ADC 时钟启动前要加上一个 T_{CY} 时间，用于执行 SLEEP 指令。

29.4.7. Timer0 和 Timer1 外部时钟要求

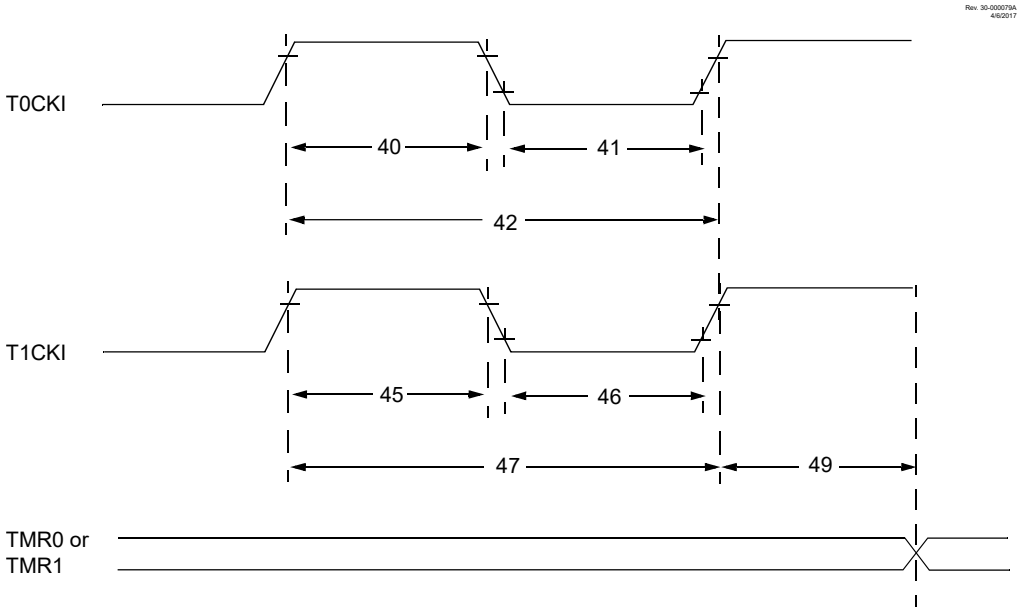
表 29-15.

标准工作条件（除非另外声明） 工作温度：-40°C ≤ T_A ≤ +85°C								
参数编号	**符号**	**特性**		**最小值**	**典型值†**	**最大值**	**单位**	**条件**
40*	T_T0H	T0CKI 高电平脉冲宽度	无预分频器	—	$0.5T_{CY}$+20	—	ns	
			有预分频器	—	10	—	ns	
41*	T_T0L	T0CKI 低电平脉冲宽度	无预分频器	—	$0.5T_{CY}$+20	—	ns	
			有预分频器	—	10	—	ns	
42*	T_T0P	T0CKI 周期		—	—	—	ns	N = 预分频值
45*	T_T1H	T1CKI 高电平时间	同步，无预分频器	—	$0.5T_{CY}$+20	—	ns	
			同步，有预分频器	—	15	—	ns	
			异步	—	30	—	ns	
46*	T_T1L	T1CKI 低电平时间	同步，无预分频器	—	$0.5T_{CY}$+20	—	ns	
			同步，有预分频器	—	15	—	ns	
			异步	—	30	—	ns	
47*	T_T1P	T1CKI 输入周期	同步	—	—	—	ns	N = 预分频值
			异步	—	60	—	ns	
49*	$TCKEZ_{TMR}1$	从外部时钟边沿到定时器递增的延时		—	7 T_{OSC}	—	—	同步模式下的定时器

* - 这些参数通过表征确定，但未经测试。

† 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 29-11. Timer0 和 Timer1 外部时钟时序



29.4.8. 捕捉/比较/PWM（CCP）要求

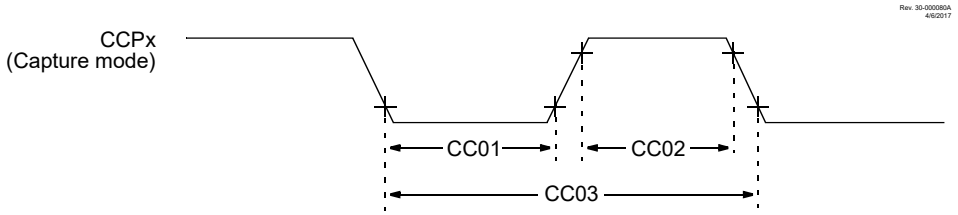
表 29-16.

标准工作条件（除非另外声明） 工作温度：-40°C ≤ T_A ≤ +85°C								
参数编号	**符号**	**特性**		**最小值**	**典型值†**	**最大值**	**单位**	**条件**
CC01*	$T_{CC}L$	CCPx 输入低电平时间	无预分频器	—	$0.5T_{CY}$+20	—	ns	
			有预分频器	—	20	—	ns	
CC02*	$T_{CC}H$	CCPx 输入高电平时间	无预分频器	—	$0.5T_{CY}$+20	—	ns	
			有预分频器	—	20	—	ns	
CC03*	$T_{CC}P$	CCPx 输入周期		—	$(3T_{CY}+40)/N$	—	ns	N = 预分频值

* - 这些参数通过表征确定，但未经测试。

† 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 29-12. 捕捉/比较/PWM（CCP）时序



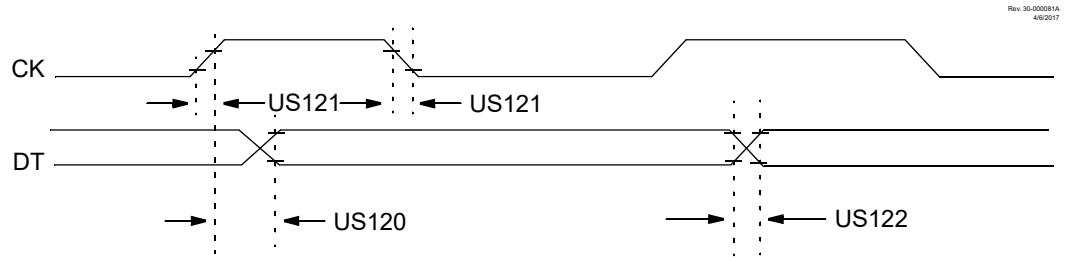
注： 负载条件请参见图 29-3。

29.4.9. EUSART 同步发送要求

表 29-17.

标准工作条件（除非另外声明）						
参数编号	符号	特性	最小值	最大值	单位	条件
US120	$T_{CK}H2_{DT}V$	同步发送（主从模式） 时钟高电平到数据输出有效的时间	—	80	ns	3.0V ≤ V_{DD} ≤ 5.5V
			—	100	ns	1.8V ≤ V_{DD} ≤ 5.5V
US121	T_{CKRF}	时钟输出上升和下降时间 （主模式）	—	45	ns	3.0V ≤ V_{DD} ≤ 5.5V
			—	50	ns	1.8V ≤ V_{DD} ≤ 5.5V
US122	T_{DTRF}	数据输出上升和下降时间	—	45	ns	3.0V ≤ V_{DD} ≤ 5.5V
			—	50	ns	1.8V ≤ V_{DD} ≤ 5.5V

图 29-13. EUSART 同步发送（主/从）时序



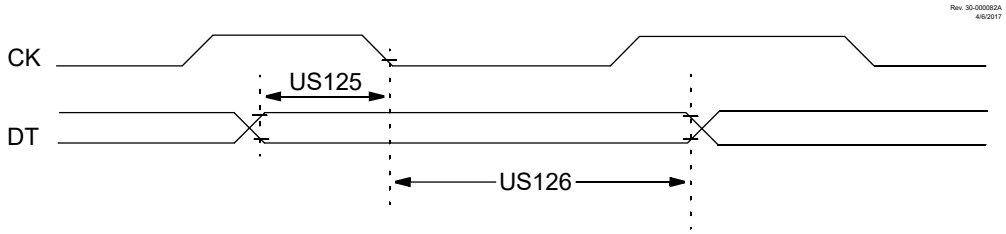
注： 负载条件请参见图 29-3。

29.4.10. EUSART 同步接收要求

表 29-18.

标准工作条件（除非另外声明）						
参数编号	符号	特性	最小值	最大值	单位	条件
US125	$T_{DT}V2_{CKL}$	同步接收（主从模式） CK ↓ 前的数据建立时间（数据保持时间）	10	—	ns	
US126	$T_{CK}L2_{DTL}$	CK ↓ 后的数据保持时间（数据保持时间）	15	—	ns	

图 29-14. EUSART 同步接收（主/从）时序



注： 负载条件请参见图 29-3。

29.4.11. SPI 模式要求

表 29-19.

标准工作条件（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
SP70*	$T_{SS}L2_{SC}H$， $T_{SS}L2_{SC}L$	$\overline{SS}$ ↓ 到 SCK ↓ 或 SCK ↑ 输入的时间	—	2.25*T_{CY}	—	ns	

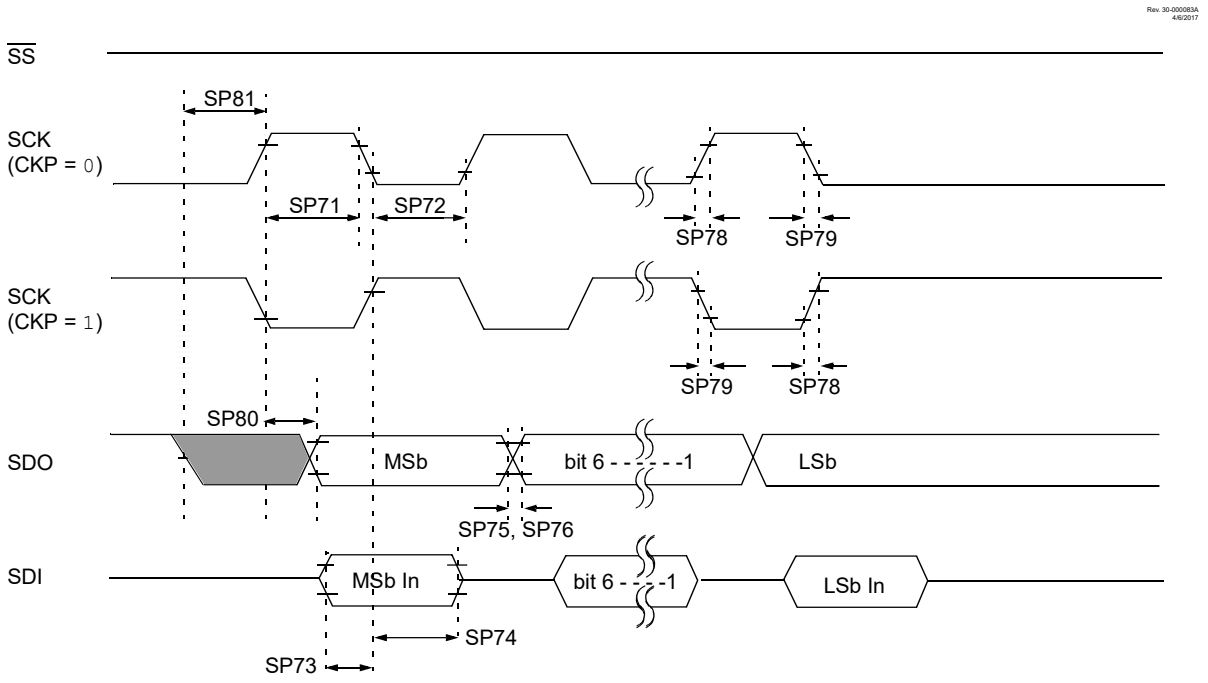
表 29-19.（续）

标准工作条件（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**
SP71*	$T_{SC}H$	SCK 输入高电平时间（从模式）	—	T_{CY} + 20	—	ns	
SP72*	$T_{SC}L$	SCK 输入低电平时间（从模式）	—	T_{CY} + 20	—	ns	
SP73*	$T_{DI}V2_{SC}H$，$T_{DI}V2_{SC}L$	SDI 数据输入至 SCK 边沿的建立时间	—	100	—	ns	
SP74*	$T_{SC}H2_{DI}L$，$T_{SC}L2_{DI}L$	SDI 数据输入至 SCK 边沿的保持时间	—	100	—	ns	
SP75*	$T_{DO}R$	SDO 数据输出上升时间	—	25	—	ns	1.8V ⩽ V_{DD} ⩽ 5.5V
SP76*	$T_{DO}F$	SDO 数据输出下降时间	—	10	—	ns	
SP77*	$T_{SS}H2_{DO}Z$	$\overline{SS}$ ↑ 到 SDO 输出高阻态的时间	—	50	—	ns	
SP78*	$T_{SC}R$	SCK 输出上升时间（主模式）	—	25	—	ns	1.8V ⩽ V_{DD} ⩽ 5.5V
SP79*	$T_{SC}F$	SCK 输出下降时间（主模式）	—	10	—	ns	
SP80*	$T_{SC}H2_{DO}V$，$T_{SC}L2_{DO}V$	SCK 边沿后 SDO 数据输出有效的时间	—	145	—	ns	1.8V ⩽ V_{DD} ⩽ 5.5V
SP81*	$T_{DO}V2_{SC}H$，$T_{DO}V2_{SC}L$	SDO 数据输出建立至 SCK 边沿的时间	—	1 T_{CY}	—	ns	
SP82*	$T_{SS}L2_{DO}V$	在 $\overline{SS}$ ↓ 边沿之后 SDO 数据输出有效的时间	—	50	—	ns	
SP83*	$T_{SC}H2_{SS}H$，$T_{SC}L2_{SS}H$	$\overline{SCK}$ 边沿之后 $\overline{SS}$ ↑ 的时间	—	1.5 T_{CY} + 40	—	ns	

* - 这些参数通过表征确定，但未经测试。

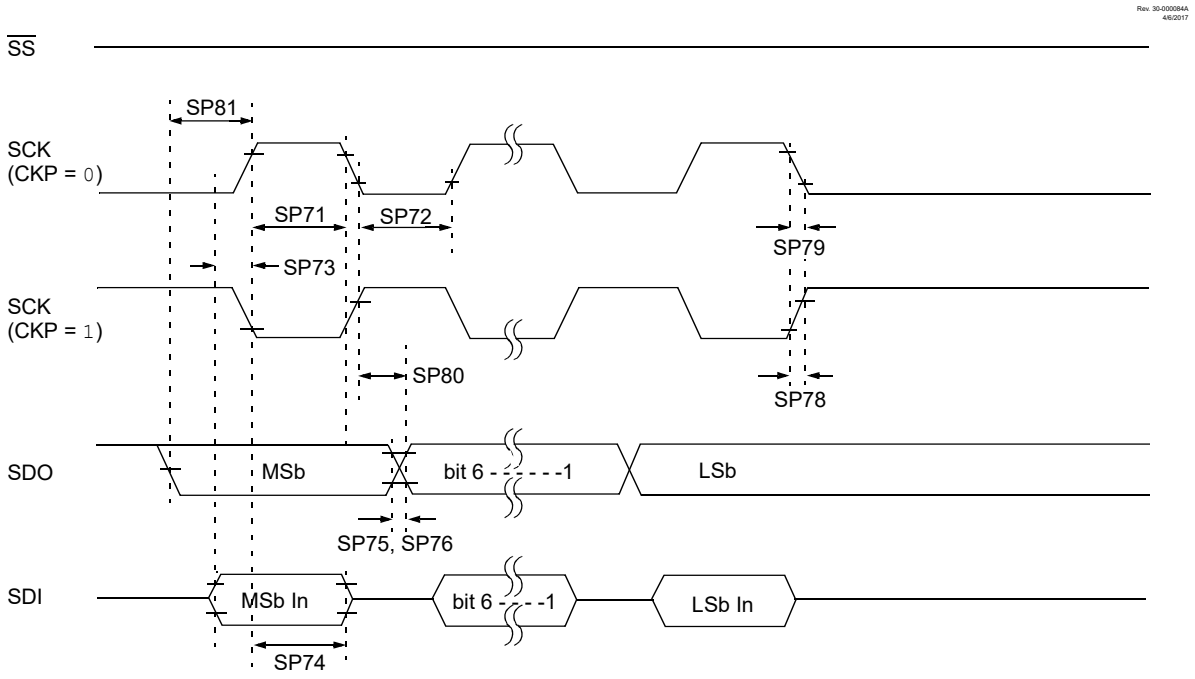
† 除非另外声明，否则“典型值”栏中的数据均为 3.0V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

图 29-15. SPI 主模式时序（CKE = 0，SMP = 0）



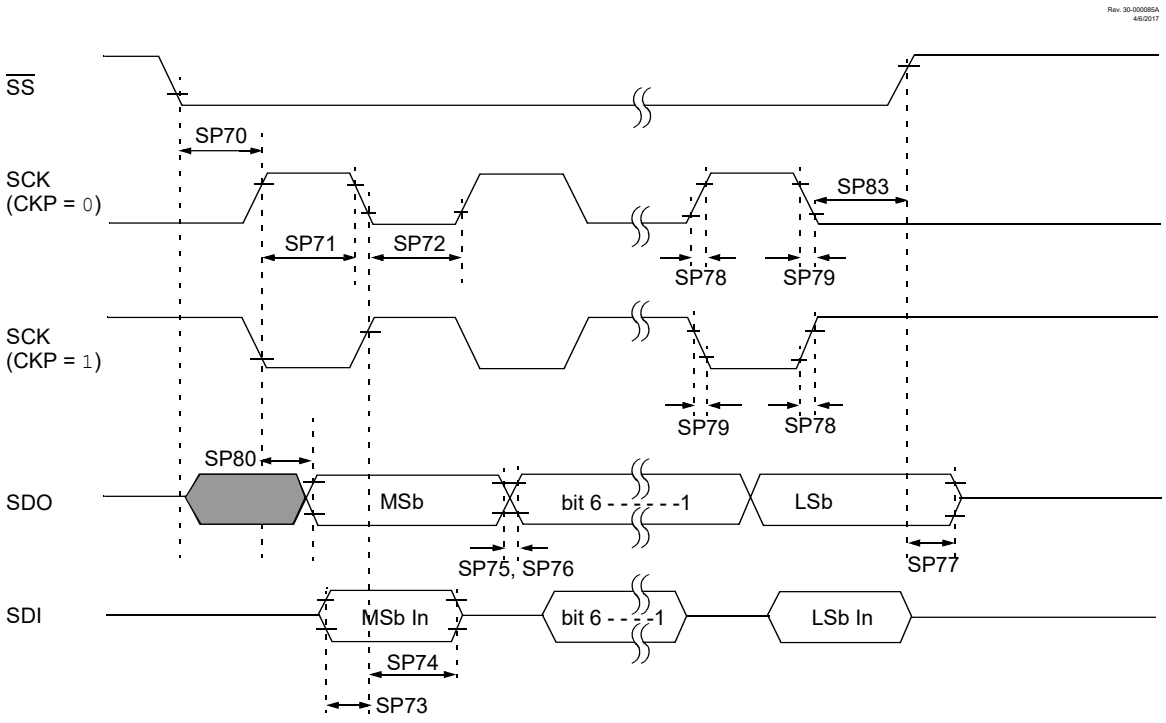
注：负载条件请参见图 29-3。

图 29-16. SPI 主模式时序（CKE = 1，SMP = 1）



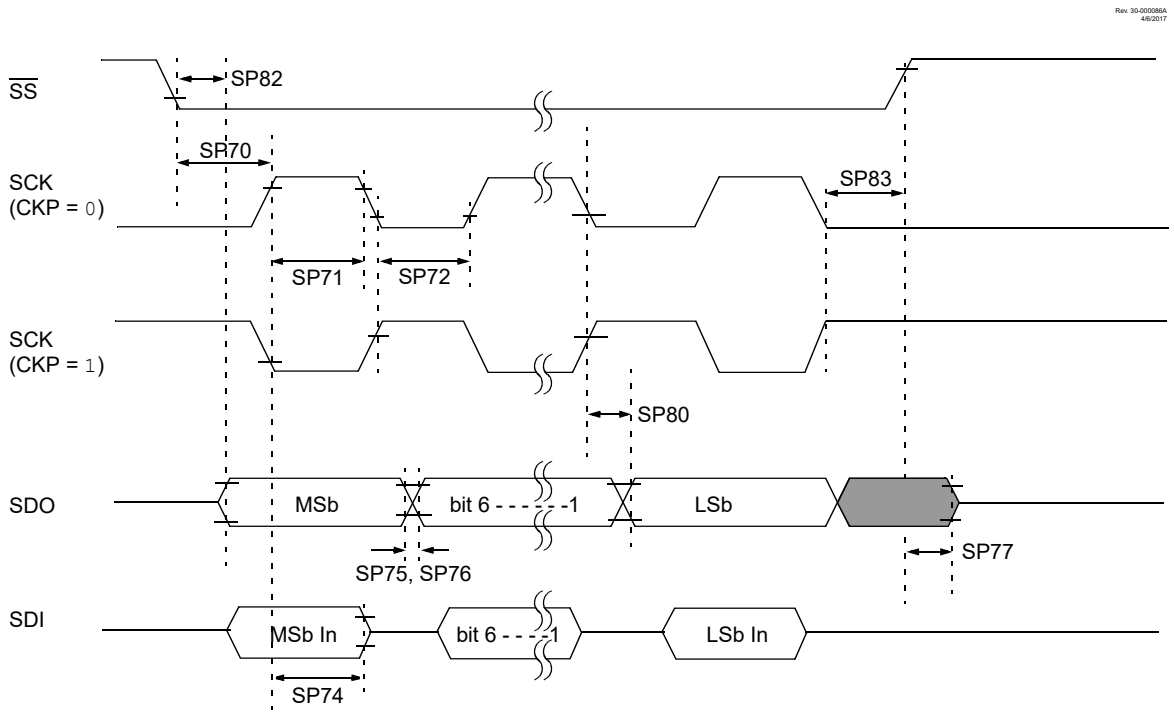
注：负载条件请参见图 29-3。

图 29-17. SPI 从模式时序（CKE = 0）



注：负载条件请参见图 29-3。

图 29-18. SPI 从模式时序（CKE = 1）



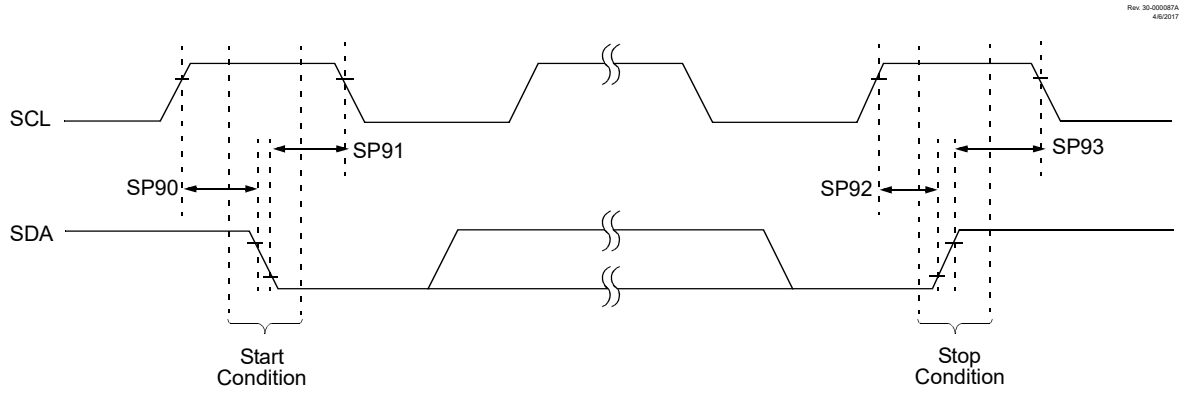
注：负载条件请参见图 29-3。

29.4.12. I²C 总线启动位/停止位要求

表 29-20.

标准工作条件（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**
SP90*	$T_{SU:STA}$	启动条件建立时间	100 kHz 模式	—	4700	—	ns	仅与重复起始条件相关
			400 kHz 模式	—	600	—		
SP91*	$T_{HD:STA}$	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*	$T_{SU:STO}$	停止条件建立时间	100 kHz 模式	—	4700	—	ns	
			400 kHz 模式	—	600	—		
SP93*	$T_{HD:STO}$	停止条件保持时间	100 kHz 模式	—	4000	—	ns	
			400 kHz 模式	—	600	—		
* - 这些参数通过表征确定，但未经测试。								

图 29-19. I²C 总线启动/停止位时序



注： 负载条件请参见图 29-3。

29.4.13. I²C 总线数据要求

表 29-21.

标准工作条件（除非另外声明）							
参数编号	**符号**	**特性**		**最小值**	**最大值**	**单位**	**条件**
SP100*	T_{HIGH}	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	$1.5T_{CY}$	—		
SP101*	T_{LOW}	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	$1.5T_{CY}$	—		
SP102*	T_R	SDA 和 SCL 上升时间	100 kHz 模式	—	1000	ns	
			400 kHz 模式	$20 + 0.1C_B$	300	ns	C_B 值规定在 10-400 pF 之间
SP103*	T_F	SDA 和 SCL 下降时间	100 kHz 模式	—	250	ns	
			400 kHz 模式	$20 + 0.1C_B$	250	ns	C_B 值规定在 10-400 pF 之间
SP106*	$T_{HD:DAT}$	数据输入保持时间	100 kHz 模式	0	—	ns	
			400 kHz 模式	0	0.9	µs	
SP107*	$T_{SU:DAT}$	数据输入建立时间	100 kHz 模式	250	—	ns	**注 2**
			400 kHz 模式	100	—	ns	
SP109*	T_{AA}	自时钟边沿到输出有效的时间	100 kHz 模式	—	3500	ns	**注 1**
			400 kHz 模式	—	—	ns	
SP110*	T_{BUF}	总线空闲时间	100 kHz 模式	4.7	—	µs	在新的传输启动之前总线必须保持空闲的时间
			400 kHz 模式	1.3	—	µs	
SP111	C_B	总线容性负载		—	400	pF	

表 29-21.（续）

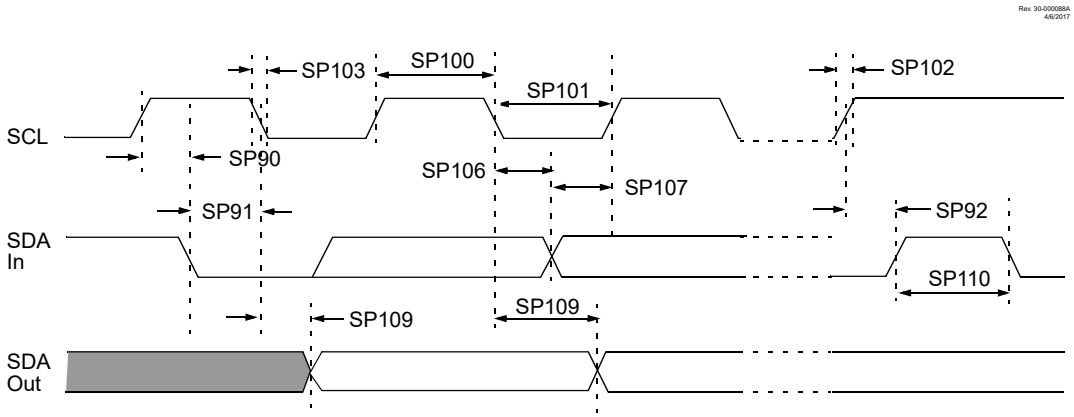
标准工作条件（除非另外声明）						
参数编号	**符号**	**特性**	**最小值**	**最大值**	**单位**	**条件**

* - 这些参数通过表征确定，但未经测试。

注：

1. 为避免意外产生启动或停止条件，作为发送器的器件必须提供这个内部最小延时（最小值 300 ns）以补偿 SCL 下降沿的未定义区域。
2. 快速模式（400 kHz）的 I²C 总线器件也可在标准模式（100 kHz）的 I²C 总线系统中使用，但必须满足 $T_{SU:DAT} \geqslant 250$ ns 的要求。如果器件没有延长 SCL 信号的低电平时间，则自动满足此条件。如果该器件延长了 SCL 信号的低电平时间，其下一个数据位必须输出到 SDA 线。在 SCL 线被释放前，根据标准模式 I²C 总线规范，TR max. + $T_{SU:DAT}$ = 1000 + 250 = 1250 ns。

图 29-20. I²C 总线数据时序



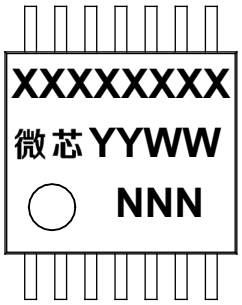
注：负载条件请参见图 29-3。

30. 封装信息

封装标识信息

图注：	XX...X	客户指定信息或部件编号
	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	由字母数字组成的追踪代码
	(e3)	雾锡（Matte Tin，Sn）的JEDEC®无铅标志
注：	部件编号如果无法在同一行内完整标注，将换行标出，因此会限制表示客户指定信息的字符数。	

14引脚TSSOP（4.4 mm）

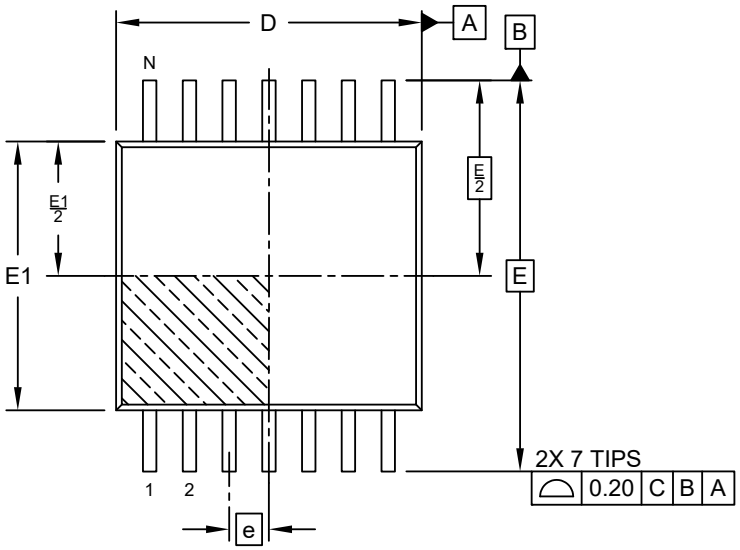


示例

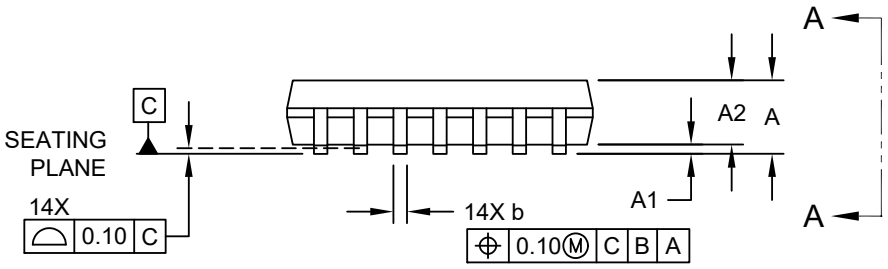
30.1. 封装详细信息

以下部分给出了封装的技术详细信息。

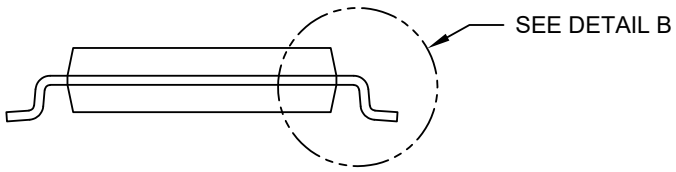
14-Lead Plastic Thin Shrink Small Outline Package [ST] - 4.4 mm Body [TSSOP]



TOP VIEW

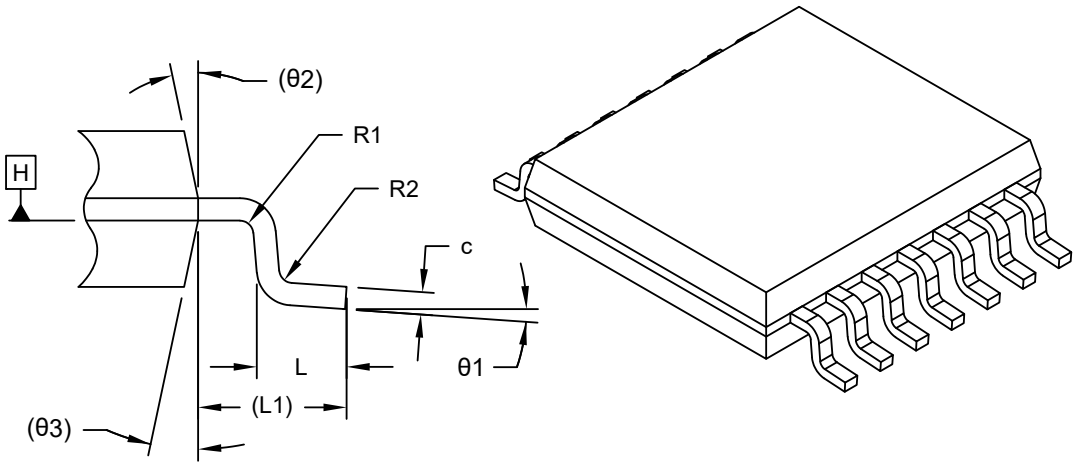


SIDE VIEW



VIEW A–A

14-Lead Plastic Thin Shrink Small Outline Package [ST] - 4.4 mm Body [TSSOP]



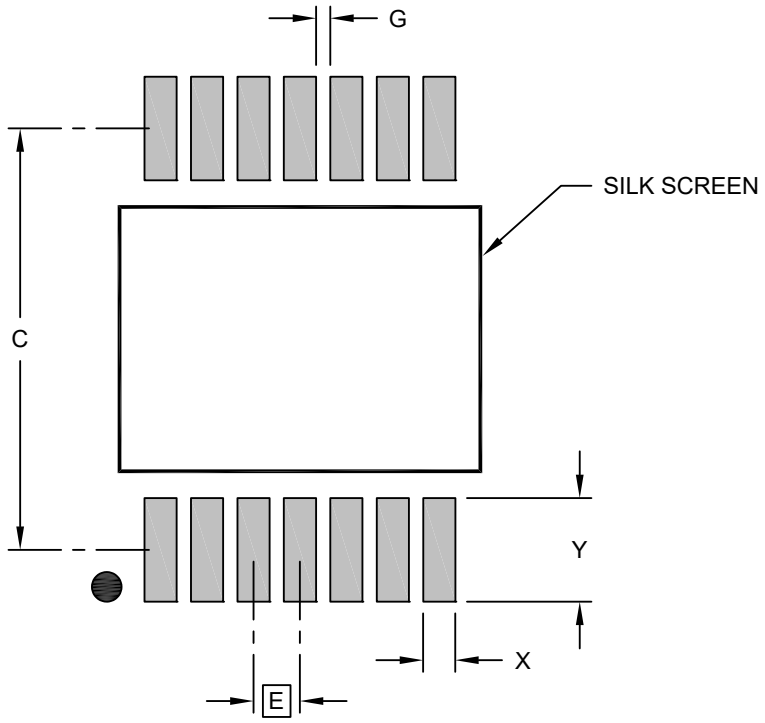
DETAIL B

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		14	
Pitch	e		0.65 BSC	
Overall Height	A	–	–	1.20
Standoff	A1	0.05	–	0.15
Molded Package Thickness	A2	0.80	1.00	1.05
Overall Length	D	4.90	5.00	5.10
Overall Width	E		6.40 BSC	
Molded Package Width	E1	4.30	4.40	4.50
Terminal Width	b	0.19	–	0.30
Terminal Thickness	c	0.09	–	0.20
Terminal Length	L	0.45	0.60	0.75
Footprint	L1		1.00 REF	
Lead Bend Radius	R1	0.09	–	–
Lead Bend Radius	R2	0.09	–	–
Foot Angle	θ1	0°	–	8°
Mold Draft Angle	θ2	–	12° REF	–
Mold Draft Angle	θ3	–	12° REF	–

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

14-Lead Plastic Thin Shrink Small Outline Package [ST] – 4.4 mm Body [TSSOP]



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.65 BSC	
Contact Pad Spacing	C		5.90	
Contact Pad Width (X14)	X			0.45
Contact Pad Length (X14)	Y			1.45
Contact Pad to Contact Pad (X12)	G	0.20		

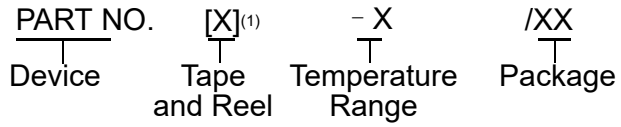
Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

产品标识体系

欲订货或获取信息，请访问 www.weixinsemi.com。



器件：	CN5223 和 CN5225	
卷带式选项：	T	= 卷带式
温度范围：	I	= -40°C 至+85°C（工业级）
封装(1)：	ST	= 14 引脚 TSSOP
封装类型：	空白	标准

示例：

- CN5223T-I/ST：卷带式，工业级温度，14 引脚 TSSOP
- CN5225-I/ST：工业级温度，14 引脚 TSSOP

制造商信息

商标

本文档中的名称、徽标和品牌均为制造商或其关联公司和/或子公司在中国和/或其他国家或地区的注册商标或商标。

法律声明

本出版物仅适用于制造商的产品，包括设计、测试以及将制造商的产品集成到用户的应用中。以其他任何方式使用这些信息都将被视为违反条款。

不涉及任何制造商知识产权的使用许可。

如果将制造商的器件用于生命维持和/或生命安全应用，一切风险由买方自负。

器件应用的详细信息仅供参考，内容可能随时更新。用户须自行确保应用符合规范。如需支持，请通过 www.weixinsemi.com 联系制造商。

用户须遵守所有适用的出口管制与经济制裁规定。

本文档中的信息“按原样”提供。制造商对这些信息不作任何形式的担保，包括但不限于针对非侵权性、适销性和特定用途的适用性的担保。除法律强制要求外，对于因这些信息或使用这些信息而产生的任何损失，制造商概不承担任何责任。在法律允许的最大范围内，制造商概不承担任何间接或附带损害赔偿。制造商在任何情况下所承担的全部责任均不超出用户为获得这些信息而向制造商支付的金额（如有）。

制造商的器件代码保护功能

请注意以下有关制造商产品的代码保护功能的要点：

- 制造商的产品均达到制造商数据手册中所述的技术规范。
- 制造商确信：在正常使用且符合工作规范的情况下，其产品非常安全。
- 制造商注重并积极保护其知识产权。严禁任何试图破坏制造商的代码保护功能的行为。
- 制造商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产品是“牢不可破”的。代码保护功能处于持续发展中。制造商承诺将不断改进产品的代码保护功能。

中国销售及服务

如需获取更多信息或支持，请通过以下方式联系我们：

邮箱：sales@weixinsemi.com

网址：www.weixinsemi.com