简介

CN2710 单片机系列具有模拟外设、独立于内核的外设和通信外设，广泛适用于各种通用和低功耗应用。该系列 28 引脚器件配有带计算功能的 10 位 ADC（ADCC），可自动采用电容分压器（Capacitive Voltage Divider，CVD）技术实现高级触摸传感、平均值处理、滤波、过采样和自动阈值比较。此外，该系列器件还提供了一组独立于内核的外设，例如互补波形发生器（Complementary Waveform Generator，CWG）、窗口看门狗定时器（Windowed Watchdog Timer，WWDT）、循环冗余校验（Cyclic Redundancy Check，CRC）/存储器扫描、过零检测（Zero-Cross Detect，ZCD）、可配置逻辑单元（Configurable Logic Cell，CLC）(1)和外设引脚选择（Peripheral Pin Select，PPS），有助于提高设计灵活性和降低系统成本。

注：

1. CN2510 器件上未提供 CLC。

CN2710 系列汇总

器件	闪存程序存储器（字节）	数据 SRAM（字节）(1)	数据 EEPROM（字节）	I/O 引脚/ PPS	带 HLT 的 8 位定时器/ 16 位定时器(2)	比较器	10 位 ADC 通道（外部/内部）	5 位 DAC	过零检测	CCP/ 10 位 PWM	CWG	CLC	低电压检测（LVD）	窗口看门狗定时器	具有存储器扫描功能的 16 位 CRC	EUSART	I²C/ SPI	外设模块禁止	温度指示器
CN2510	16K	2304	256	25/有	3/4	2	24/4	1	1	2/2	1	0	1	有	有	1	1/1	有	有
CN2610	32K	3584	1024	25/有	3/4	2	24/4	1	1	2/2	1	8	1	有	有	2	2/2	有	有
CN2710	64K	3584	1024	25/有	3/4	2	24/4	1	1	2/2	1	8	1	有	有	2	2/2	有	有

注：

1. SRAM 包含 256 字节的扇区空间。
2. Timer0 可配置为 8 位或 16 位定时器。
3. 要对 CN2510 进行编程，请选择 PIC18F25Q10。
4. 要对 CN2610 进行编程，请选择 PIC18F26Q10。
5. 要对 CN2710 进行编程，请选择 PIC18F27Q10。

内核特性

- 为 C 编译器优化的 RISC 架构
- 工作速度：
  - DC – 64 MHz 时钟输入
  - 最小指令周期为 62.5 ns
- 可编程 2 级中断优先级
- 31 级硬件堆栈
- 低电流上电复位（Power-on Reset，POR）
- 上电延时定时器（Power-up Timer，PWRT）

- 欠压复位（Brown-out Reset，BOR）
- 低功耗 BOR（Low-Power BOR，LPBOR）选项
- 窗口看门狗定时器（WWDT）

存储器

- 最大 64 KB 的闪存程序存储器
- 最大 3,584B 的数据 SRAM 存储器
- 最大 1 KB 的数据 EEPROM
- 可编程代码保护和写保护
- 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

节能工作模式

- 休眠模式：1.8V 时的典型值为 50 nA
- 看门狗定时器：1.8V 时的典型值为 500 nA
- 辅助振荡器：32 kHz 时为 500 nA
- 工作电流：
  - 32 kHz、1.8V 时的典型值为 8 μA
  - 1.8V 时的典型值为 32 μA/MHz

数字外设

- 八个可配置逻辑单元（CLC）：
  - 集成组合和顺序逻辑

  注：CN2510 器件上未提供。
- 一个互补波形发生器（CWG）：
  - 上升沿和下降沿死区控制
  - 全桥、半桥和单通道驱动
  - 多个信号源
- 一个数据信号调制器（Data Signal Modulator，DSM）
  - 通过数字数据调制载波信号，以生成定制载波同步输出波形
- 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 10 位脉宽调制器（Pulse-Width Modulator，PWM）：
  - 10 位分辨率
  - 独立脉冲输出
- 3 个带有硬件限制定时器（Hardware Limit Timer，HLT）的 8 位定时器（TMR2/4/6）

- 4 个 16 位定时器（TMR0/1/3/5）
- 具有存储器扫描功能的 16 位可编程 CRC：
  - 可靠的数据/程序存储器监视功能，用于实现故障保护（如 B 类）
  - 为闪存或 EEPROM 的任何部分计算 CRC
  - 高速或后台操作
- 外设引脚选择（PPS）：
  - 支持数字 I/O 的引脚映射
- 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
- 最多两个主同步串行端口（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
- 器件 I/O 端口特性：
  - 25 个 I/O 引脚
  - 1 个仅输入引脚（RE3）
  - 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
  - 3 个外部中断引脚

模拟外设

- 带计算功能的 10 位模数转换器（Analog-to-Digital Converter with Computation，ADCC）：
  - 24 个外部通道
  - 4 个内部模拟通道
  - 可在休眠模式下进行转换
  - 内部和外部触发选项
  - 自动对输入信号进行数学函数运算：
    - 平均值计算、滤波器计算、过采样和阈值比较
- 支持硬件电容分压器（CVD）：
  - 8 位预充电定时器
  - 可调节采样保持电容阵列
  - 保护环数字输出驱动器
- 过零检测（ZCD）：
  - 检测引脚上的交流信号何时越过地电压
- 5 位数模转换器（Digital-to-Analog Converter，DAC）：
  - 输出可供外部使用
  - 可编程 5 位电压（V_{DD} 的百分比、[V_{REF+} - V_{REF-}]或 FVR）

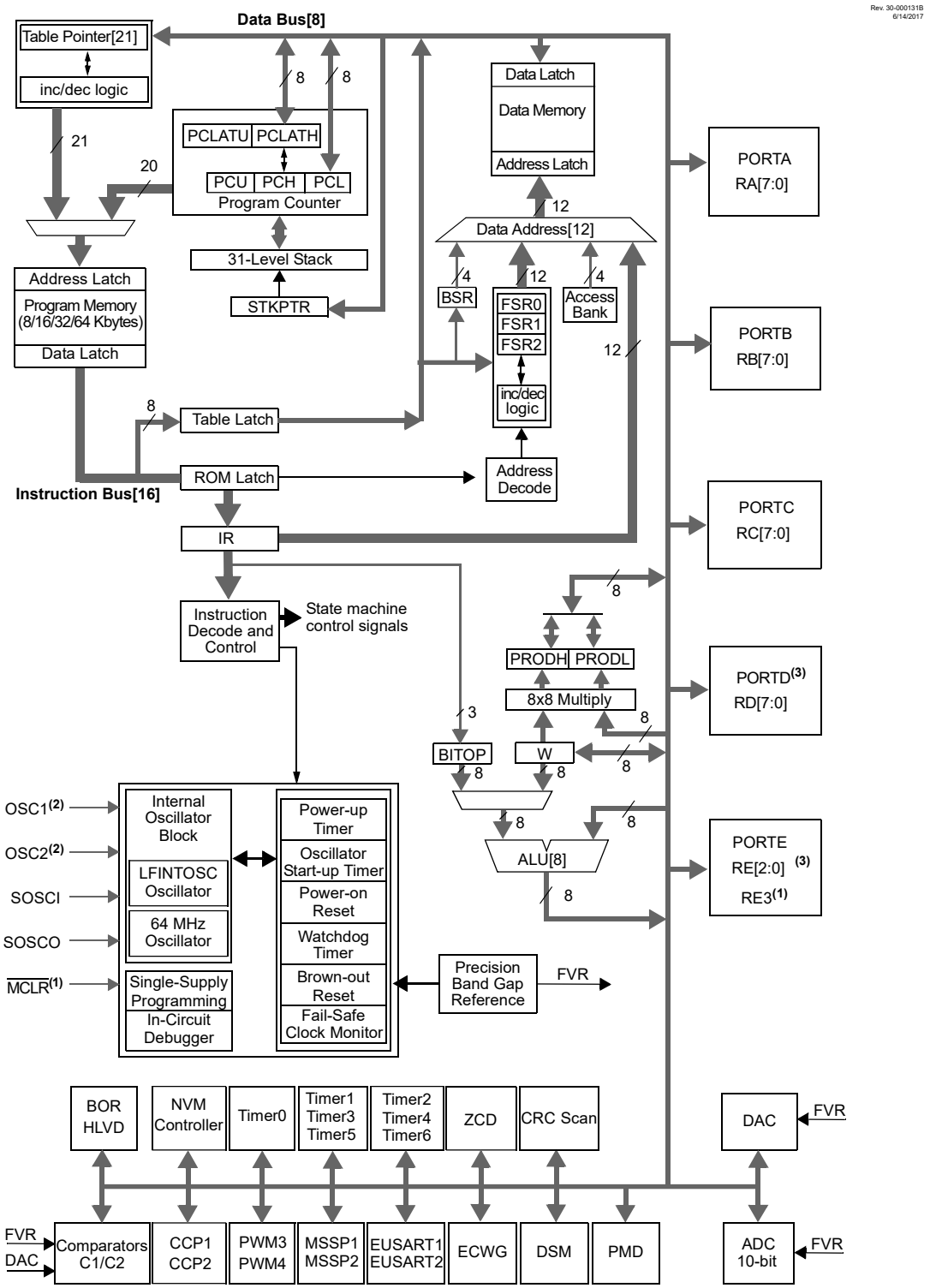
  - 内部连接至比较器和 ADC
- 2 个比较器（CMP）：
  - 4 个外部输入
  - 通过 PPS 选择外部输出
- 固定参考电压（Fixed Voltage Reference，FVR）模块：
  - 1.024V、2.048V 和 4.096V 输出电压
  - 2 个缓冲输出：一个用于 DAC/CMP，另一个用于 ADC

时钟结构

- 高精度内部振荡器模块（HFINTOSC）：
  - 可选择最高 64 MHz 的频率
  - 校准精度±1%
- 32 kHz 低功耗内部振荡器（LFINTOSC）
- 外部 32 kHz 晶振（SOSC）
- 外部高频振荡器模块：
  - 3 种晶振/谐振器模式
  - 数字时钟输入模式
  - 可搭配外部时钟源的 4x PLL
- 故障保护时钟监视器：
  - 允许在外部时钟停止时安全关断
- 振荡器起振定时器（Oscillator Start-up Timer，OST）

框图

图 1. CN2510/2610/2710 框图



Note 1: RE3 is only available when $\overline{\text{MCLR}}$ functionality is disabled.

2: OSC1/CLKIN and OSC2/CLKOUT are only available in select oscillator modes.

3: PORTD and PORTE[2:0] not implemented on 28-pin devices.

目录

简介.....1

CN2710 系列汇总.....1

内核特性.....1

1. 封装.....8
2. 引脚图.....9
3. 引脚分配表.....10
4. 寄存器和位命名约定.....12
5. 寄存器图例.....14
6. 器件配置.....15
7. OSC——振荡器模块.....29
8. REFCLK——参考时钟输出模块.....48
9. 节能工作模式.....53
10. PMD——外设模块禁止.....61
11. 复位.....70
12. WWDT——窗口看门狗定时器.....82
13. 存储器构成.....92
14. NVM——非易失性存储器控制.....124
15. 8x8 硬件乘法器.....152
16. CRC——带存储器扫描器的循环冗余校验模块.....157
17. 中断.....176
18. I/O 端口.....205
19. 电平变化中断.....237
20. PPS——外设引脚选择模块.....252
21. TMR0——Timer0 模块.....262
22. TMR1——带门控的 Timer1 模块.....270
23. TMR2——Timer2 模块.....286
24. CCP——捕捉/比较/PWM 模块.....307
25. PWM——脉宽调制.....320

26. CCP 和 PWM 定时器选择..... 328
27. CWG——互补波形发生器模块..... 331
28. CLC——可配置逻辑单元..... 357
29. DSM——数据信号调制器模块..... 377
30. MSSP——主同步串行端口模块..... 388
31. EUSART——增强型通用同步/异步收发器..... 449
32. FVR——固定参考电压..... 478
33. 温度指示器模块..... 483
34. ADCC——带计算功能的模数转换器模块..... 485
35. DAC——5 位数模转换器..... 527
36. CMP——比较器模块..... 533
37. HLVD——高/低电压检测..... 544
38. ZCD——过零检测模块..... 551
39. 编程..... 558
40. 指令集汇总..... 560
41. 寄存器汇总..... 628
42. 电气规范..... 637
43. 封装信息..... 661
产品标识体系..... 665
制造商信息..... 666

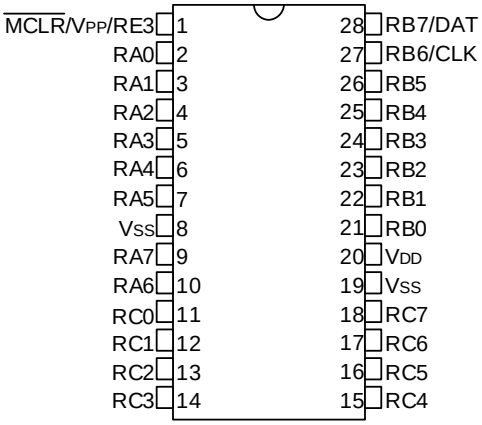
1. 封装

表 1-1. 封装

器件	28 引脚 SSOP
CN2510	•
CN2610	•
CN2710	•

2. 引脚图

图 2-1. 28 引脚 SSOP



3. 引脚分配表

表 3-1. 28 引脚分配表

I/O[2]	28 引脚 SSOP	A/D	参考电压	比较器	定时器	CCP	CWG	ZCD	中断	EUSART	DSM	CLC	MSSP	基本功能
RA0	2	ANA0	—	C1IN0- C2IN0-	—	—	—	—	IOCA0	—	—	CLCIN0[6] CLCIN4[6]	—	—
RA1	3	ANA1	—	C1IN1- C2IN1-	—	—	—	—	IOCA1	—	—	CLCIN1[6] CLCIN5[6]	—	—
RA2	4	ANA2	DAC1OUT1 V_{REF}- (DAC) V_{REF}-（ADC）	C1IN0+ C2IN0+	—	—	—	—	IOCA2	—	—	—	—	—
RA3	5	ANA3	V_{REF}+（DAC） V_{REF}+（ADC）	C1IN1+	—	—	—	—	IOCA3	—	MDCARL[1]	—	—	—
RA4	6	ANA4	—	—	T0CKI[1]	—	—	—	IOCA4	—	MDCARH[1]	—	—	—
RA5	7	ANA5	—	—	—	—	—	—	IOCA5	—	MDSRC[1]	—	$\overline{SS1}$[1]	—
RA6	10	ANA6	—	—	—	—	—	—	IOCA6	—	—	—	—	CLKOUT OSC2
RA7	9	ANA7	—	—	—	—	—	—	IOCA7	—	—	—	—	OSC1 CLKIN
RB0	21	ANB0	—	C2IN1+	—	—	CWG1[1]	ZCDIN	IOCB0 INT0[1]	—	—	—	$\overline{SS2}$[1,6]	—
RB1	22	ANB1	—	C1IN3- C2IN3-	—	—	—	—	IOCB1 INT1[1]	—	—	—	SCK2[1,6] SCL2[3,4,6]	—
RB2	23	ANB2	—	—	—	—	—	—	IOCB2 INT2[1]	—	—	—	SDI2[1,6] SDA2[3,4,6]	—
RB3	24	ANB3	—	C1IN2- C2IN2-	—	—	—	—	IOCB3	—	—	—	—	—
RB4	25	ANB4	—	—	T5G[1]	—	—	—	IOCB4	—	—	—	—	—
RB5	26	ANB5	—	—	T1G[1]	—	—	—	IOCB5	—	—	—	—	—
RB6	27	ANB6	—	—	—	—	—	—	IOCB6	CK2[1,3,6]	—	CLCIN2[6] CLCIN6[6]	—	CLK
RB7	28	ANB7	DAC1OUT2	—	T6IN[1]	—	—	—	IOCB7	RX2/DT2[1,3,6]	—	CLCIN3[6] CLCIN7[6]	—	DAT
RC0	11	ANC0	—	—	T1CKI[1] T3CKI[1] T3G[1]	—	—	—	IOCC0	—	—	—	—	SOSCO
RC1	12	ANC1	—	—	—	CCP2[1]	—	—	IOCC1	—	—	—	—	SOSCI
RC2	13	ANC2	—	—	T5CKI[1]	CCP1[1]	—	—	IOCC2	—	—	—	—	—
RC3	14	ANC3	—	—	T2IN[1]	—	—	—	IOCC3	—	—	—	SCK1[1] SCL1[3,4]	—
RC4	15	ANC4	—	—	—	—	—	—	IOCC4	—	—	—	SDI1[1] SDA1[3,4]	—
RC5	16	ANC5	—	—	T4IN[1]	—	—	—	IOCC5	—	—	—	—	—
RC6	17	ANC6	—	—	—	—	—	—	IOCC6	CK1[1,3]	—	—	—	—
RC7	18	ANC7	—	—	—	—	—	—	IOCC7	RX1/DT1[1,3]	—	—	—	—
RE3	1	—	—	—	—	—	—	—	IOCE3	—	—	—	—	V_{PP}/$\overline{MCLR}$
V_{SS}	19	—	—	—	—	—	—	—	—	—	—	—	—	V_{SS}
V_{DD}[5]	20	—	—	—	—	—	—	—	—	—	—	—	—	V_{DD}
V_{SS}	8	—	—	—	—	—	—	—	—	—	—	—	—	V_{SS}

表 3-1. 28 引脚分配表（续）

I/O[2]	28 引脚 SSOP	A/D	参考电压	比较器	定时器	CCP	CWG	ZCD	中断	EUSART	DSM	CLC	MSSP	基本功能
OUT[2]	—	ADGRDA ADGRDB	—	C1OUT C2OUT	TMR0	CCP1 CCP2 PWM3 PWM4	CWG1A CWG1B CWG1C CWG1D	—	—	TX1/CK1[3] DT1[3] TX2/CK2[3,6] DT2[3,6]	DSM	CLC1OUT[6] CLC2OUT[6] CLC3OUT[6] CLC4OUT[6] CLC5OUT[6] CLC6OUT[6] CLC7OUT[6] CLC8OUT[6]	SDO1 SCK1 SDO2[6] SCK2[6]	—

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其他多个 PORTx 引脚之一。有关可用于此信号的端口引脚的详细信息，请参见外设输入选择表。
2. 此行中显示的所有输出信号均为 PPS 可重映射输出信号。这些信号可映射到外设输出选择表中所列的任一 PORTx 引脚选项的输出。
3. 此信号为双向信号。为使模块正常工作，固件应将此信号映射到 PPS 输入和 PPS 输出寄存器中的同一引脚。
4. 这些引脚配置为 I^2C 逻辑电平。SCLx/SDAx 信号可分配给这些引脚中的任意一个。通过 PPS 分配给其他引脚（如 RB1）可以工作，但输入逻辑电平将为通过 INLVL 寄存器选择的标准 TTL/ST，而不是 I^2C 特定的或 SMBus 输入缓冲器阈值。
5. V_{DD} 引脚需要通过一个 0.1 µF 的旁路电容连接到 V_{SS}。
6. CN2510 器件上未提供。

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

器件配置功能由配置字、代码保护、器件 ID 和版本 ID 组成。

6.1. 配置字

有六个配置字可供用户用于选择器件振荡器、复位和存储器保护选项。这些配置字实现为位于 300000h 到 30000Bh 的配置字 1 到配置字 6。

重要：配置字中的 $\overline{\text{DEBUG}}$ 位由器件开发工具（包括调试器和编程器）自动管理。为使器件正常工作，该位必须保持为 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。程序存储器保护和数据存储器保护独立进行控制。任何代码保护设置都不会影响对程序存储器的内部访问。

6.2.1. 程序存储器保护

整个程序存储空间都通过 $\overline{\text{CP}}$ 位来防止外部读写操作。当 $\overline{\text{CP}}$ = 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。

6.2.2. 数据存储器保护

整个数据 EEPROM 存储空间都通过 $\overline{\text{CPD}}$ 位来防止外部读写操作。当 $\overline{\text{CPD}}$= 0 时，将禁止对 EEPROM 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 都可以继续读取数据 EEPROM 存储器。

6.3. 写保护

通过写保护，可以防止器件发生意外的自写操作。在保护应用程序（如自举程序软件）的同时，可以允许对程序存储器的其他区域进行修改。

WRTn 位定义了受保护的程序存储块的大小。

6.4. 用户 ID

存储空间中有 256 个字节（200000h-20000FFh）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。有关如何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**“用户 ID、器件 ID 和配置字访问”**一节。有关校验和计算的更多信息，请参见*“CN2710 Memory Programming Specification”*（DS40001874）。

6.5. 器件 ID 和版本 ID

16 位器件 ID 字位于 0x3FFFFE，16 位版本 ID 位于 0x3FFFFC。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：配置字

6.6.1. CONFIG1

名称： CONFIG1
偏移量： 0x300000

配置字 1

振荡器

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

位	7	6	5	4	3	2	1	0
		RSTOSC[2:0]				FEXTOSC[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 – FCMEN 故障保护时钟监视器使能位

值	说明
1	使能故障保护时钟监视器
0	禁止故障保护时钟监视器

Bit 11 – CSWEN 时钟切换使能位

值	说明
1	允许写入 NOSC 和 NDIV
0	用户软件无法更改 NOSC 和 NDIV 位

Bit 8 – $\overline{\text{CLKOUTEN}}$ 时钟输出使能位
如果 FEXTOSC = HS、XT 或 LP，该位被忽略。
其他情况：

值	说明
1	禁止 CLKOUT 功能；OSC2 为 I/O 功能
0	使能 CLKOUT 功能；OSC2 为 F_{OSC}/4 时钟

Bit 6:4 – RSTOSC[2:0] COSC 的上电默认值位
该值是 COSC 的复位默认值，用于选择用户软件首次使用的振荡器。请参见 COSC 操作。

值	说明
111	EXTOSC 根据 FEXTOSC 位操作（器件制造时的默认设置）
110	HFFRQ = 4 MHz、CDIV = 4:1 的 HFINTOSC
101	LFINTOSC
100	SOSC
011	保留
010	采用 4x PLL 的 EXTOSC；EXTOSC 根据 FEXTOSC 位操作
001	保留
000	HFFRQ = 64 MHz、CDIV = 1:1 的 HFINTOSC 将 COSC/NOSC 复位为 b'110。

Bit 2:0 – FEXTOSC[2:0] FEXTOSC 外部振荡器模式选择位

值	说明
111	ECH（外部时钟），高于 16 MHz

值	说明
110	ECM（外部时钟），500 kHz 至 16 MHz
101	ECL（外部时钟），低于 500 kHz
100	振荡器未使能
011	保留（不要使用）
010	HS（晶振），高于 4 MHz
001	XT（晶振），500 kHz 至 4 MHz
000	LP（晶振），针对 32.768 kHz 经过了优化

6.6.2. CONFIG2

名称： CONFIG2
偏移量： 0x300002

配置字 2

监控器

位	15	14	13	12	11	10	9	8
	$\overline{XINST}$		$\overline{DEBUG}$	STVREN	PPS1WAY	$\overline{ZCD}$	BORV[1:0]	
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

位	7	6	5	4	3	2	1	0
	BOREN[1:0]		$\overline{LPBOREN}$				$\overline{PWRTE}$	MCLRE
访问	R/W	R/W	R/W				R/W	R/W
复位	0	1	1				1	1

Bit 15 – $\overline{XINST}$ 扩展指令集使能位

值	说明
1	禁止扩展指令集和变址寻址模式（传统模式）
0	使能扩展指令集和变址寻址模式

Bit 13 – $\overline{DEBUG}$ 调试器使能位

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/下溢复位使能位

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 位一次置 1 使能位

值	说明
1	PPSLOCKED 位只能在执行解锁序列之后置 1 一次；将 PPSLOCK 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 或清零（前提是已执行解锁序列）

Bit 10 – $\overline{ZCD}$ ZCD 禁止位

值	说明
1	禁止 ZCD。将 ZCDCON 的 ZCDSEN 位置 1 可使能 ZCD
0	ZCD 始终使能，忽略 PMDx[ZCDMD]位

Bit 9:8 – BORV[1:0] 欠压复位电压选择位

值	说明
11	欠压复位电压（V_{BOR}）设为 1.90V
10	欠压复位电压（V_{BOR}）设为 2.45V
01	欠压复位电压（V_{BOR}）设为 2.7V
00	欠压复位电压（V_{BOR}）设为 2.85V

Bit 7:6 – BOREN[1:0] 欠压复位使能位
使能后，欠压复位电压（V_{BOR}）通过 BORV 位设置

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
01	按照 SBOREN 使能欠压复位
00	禁止欠压复位

Bit 5 – $\overline{\text{LPBOREN}}$ 低功耗 BOR 使能位

值	说明
1	禁止低功耗欠压复位
0	使能低功耗欠压复位

Bit 1 – $\overline{\text{PWRTE}}$ 上电延时定时器使能位

值	说明
1	禁止 PWRT
0	使能 PWRT

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

值	条件	说明
x	如果 LVP = 1	RE3 引脚功能为 $\overline{\text{MCLR}}$
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

6.6.3. CONFIG3

名称： CONFIG3
偏移量： 0x300004

配置字 3

窗口看门狗定时器

位	15	14	13	12	11	10	9	8
			WDTCCS[2:0]			WDTCWS[2:0]		
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			1	1	1	1	1	1

位	7	6	5	4	3	2	1	0
		WDTE[1:0]		WDTCPS[4:0]				
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 13:11 – WDTCCS[2:0] WDT 输入时钟选择器位

值	条件	说明
x	WDTE = 00	这些位不起任何作用
111	WDTE ≠ 00	软件控制
110 至 010	WDTE ≠ 00	保留（默认为 LFINTOSC）
001	WDTE ≠ 00	WDT 参考时钟为 31.25 kHz MFINTOSC
000	WDTE ≠ 00	WDT 参考时钟为 31.0 kHz LFINTOSC（默认值）

Bit 10:8 – WDTCWS[2:0] WDT 窗口选择位

WDTCWS	POR 时的 WDTCON1[WINDOW]			对 WINDOW 进行软件控制？	需要进行密钥访问？
	值	窗口延时时间百分比	窗口打开时间百分比		
111	111	n/a	100	是	否
110	110	n/a	100	否	是
101	101	25	75		
100	100	37.5	62.5		
011	011	50	50		
010	010	62.5	37.5		
001	001	75	25		
000	000	87.5	12.5		

Bit 6:5 – WDTE[1:0] WDT 工作模式位

值	说明
11	WDT 使能（无论是否处于休眠状态）；忽略 WDTCON0 中的 SEN 位
10	WDT 在休眠 = 0 时使能，在休眠 = 1 时暂停；忽略 WDTCON0 中的 SEN 位
01	WDT 由 WDTCON0 中的 SEN 位使能/禁止
00	WDT 禁止；忽略 WDTCON0 中的 SEN 位

Bit 4:0 – WDTCPS[4:0] WDT 周期选择位

WDTCPS	POR 时的 WDTCON0[WDTPS]				对 WDTPS 进行软件控制？
	值	分频比		典型超时（F_{IN} = 31 kHz）	
11111	01011	1:65536	2^{16}	2s	是
11110 ... 10011	11110 ...10011	1:32	2^{5}	1 ms	否

WDTCPS （续）

WDTCPS	POR 时的 WDTCON0[WDTPS]				对 WDTPS 进行软件控制？
	值	分频比		典型超时（F_{IN} = 31 kHz）	
10010	10010	1:8388608	2^{23}	256s	否
10001	10001	1:4194304	2^{22}	128s	
10000	10000	1:2097152	2^{21}	64s	
01111	01111	1:1048576	2^{20}	32s	
01110	01110	1:524299	2^{19}	16s	
01101	01101	1:262144	2^{18}	8s	
01100	01100	1:131072	2^{17}	4s	
01011	01011	1:65536	2^{16}	2s	
01010	01010	1:32768	2^{15}	1s	
01001	01001	1:16384	2^{14}	512 ms	
01000	01000	1:8192	2^{13}	256 ms	
00111	00111	1:4096	2^{12}	128 ms	
00110	00110	1:2048	2^{11}	64 ms	
00101	00101	1:1024	2^{10}	32 ms	
00100	00100	1:512	2^{9}	16 ms	
00011	00011	1:256	2^{8}	8 ms	
00010	00010	1:128	2^{7}	4 ms	
00001	00001	1:64	2^{6}	2 ms	
00000	00000	1:32	2^{5}	1 ms	

6.6.4. CONFIG4

名称： CONFIG4
偏移量： 0x300006

配置字 4

存储器写保护

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

位	7	6	5	4	3	2	1	0
					WRT3	WRT2	WRT1	WRT0
访问					R/W	R/W	R/W	R/W
复位					1	1	1	1

Bit 13 – LVP 低电压编程使能位
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。

值	说明
1	使能低电压编程。$\overline{MCLR}/V_{PP}$ 引脚功能是 $\overline{MCLR}$。忽略 MCLRE 配置位。
0	必须使用 $\overline{MCLR}/V_{PP}$ 的 HV 进行编程

Bit 12 – SCANE 扫描器使能位

值	说明
1	扫描器模块可供使用，PMD0[SCANMD]位用于使能模块
0	扫描器模块不可用，忽略 PMD0[SCANMD]位

Bit 10 – WRTD 数据 EEPROM 写保护位

值	说明
1	数据 EEPROM 不受写保护
0	数据 EEPROM 受写保护

Bit 9 – WRTB 引导块写保护位

值	说明
1	引导块不受写保护
0	引导块受写保护

Bit 8 – WRTC 配置寄存器写保护位

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 0, 1, 2, 3 – WRTn 用户非易失性存储器（Nonvolatile Memory，NVM）自写保护位

值	说明
1	相应的存储器块不受写保护
0	相应的存储器块受写保护

6.6.5. CONFIG5

名称： CONFIG5
偏移量： 0x300008

配置字 5

代码保护

位	15	14	13	12	11	10	9	8
访问								
复位								

位	7	6	5	4	3	2	1	0
							$\overline{CPD}$	$\overline{CP}$
访问							RO	RO
复位							1	1

Bit 1 – $\overline{CPD}$ 数据 NVM（DFM）存储器代码保护位

值	说明
1	禁止数据 NVM 代码保护
0	使能数据 NVM 代码保护

Bit 0 – $\overline{CP}$ 用户 NVM 程序存储器代码保护位

值	说明
1	禁止用户 NVM 代码保护
0	使能用户 NVM 代码保护

6.6.6. CONFIG6

名称： CONFIG6
偏移量： 0x30000A

配置字 6

存储器读保护

位	15	14	13	12	11	10	9	8
							EBTRB	
访问							R/W	
复位							1	

位	7	6	5	4	3	2	1	0
					EBTR3	EBTR2	EBTR1	EBTR0
访问					R/W	R/W	R/W	R/W
复位					1	1	1	1

Bit 9 – EBTRB 表读保护位

值	说明
1	存储器引导块未受保护，可从其他块中对其执行表读操作
0	存储器引导块受保护，不能从其他块中对其执行表读操作

Bit 0, 1, 2, 3 – EBTRn 表读保护位

值	说明
1	相应的存储器块未受保护，可从其他块中对其执行表读操作
0	相应的存储器块受保护，不能从其他块中对其执行表读操作

6.7. 寄存器汇总——配置字

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x2FFFFF	保留									
0x300000	CONFIG1	7:0		RSTOSC[2:0]				FEXTOSC[2:0]		
		15:8			FCMEN		CSWEN			$\overline{\text{CLKOUTEN}}$
0x300002	CONFIG2	7:0	BOREN[1:0]		$\overline{\text{LPBOREN}}$				$\overline{\text{PWRTE}}$	MCLRE
		15:8	$\overline{\text{XINST}}$		$\overline{\text{DEBUG}}$	STVREN	PPS1WAY	$\overline{\text{ZCD}}$	BORV[1:0]	
0x300004	CONFIG3	7:0		WDTE[1:0]		WDTCPS[4:0]				
		15:8			WDTCCS[2:0]			WDTCWS[2:0]		
0x300006	CONFIG4	7:0					WRT3	WRT2	WRT1	WRT0
		15:8			LVP	SCANE		WRTD	WRTB	WRTC
0x300008	CONFIG5	7:0							$\overline{\text{CPD}}$	$\overline{\text{CP}}$
		15:8								
0x30000A	CONFIG6	7:0					EBTR3	EBTR2	EBTR1	EBTR0
		15:8							EBTRB	

6.8. 寄存器定义：器件和版本

6.8.1. 器件 ID

名称： 器件 ID
偏移量： 0x3FFFFE

器件 ID 寄存器

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

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

Bit 15:0 – DEV[15:0]
器件 ID 位

器件	器件 ID
CN2510	71A0h
CN2610	7180h
CN2710	7100h

6.8.2. 版本 ID

名称： 版本 ID
偏移量： 0x3FFFFC

版本 ID 寄存器

位	15	14	13	12	11	10	9	8
	1010[3:0]				MJRREV[5:2]			
访问	RO	RO	RO	RO	RO	RO	RO	RO
复位	1	0	1	0	q	q	q	q

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

Bit 15:12 – 1010[3:0] 读为 1010。
对于该系列的所有器件，这些位的值固定为 1010。

Bit 11:6 – MJRREV[5:0] 主版本 ID 位
这些位用于标识主版本。主版本用全层版本表示（A0、B0 和 C0 等）。
版本 A = b'00 0000'

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

6.9. 寄存器汇总——器件和版本

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x3FFFFB	保留									
0x3FFFFC	版本 ID	7:0	MJRREV[1:0]		MNRREV[5:0]					
		15:8	1010[3:0]				MJRREV[5:2]			
0x3FFFFE	器件 ID	7:0	DEV[7:0]							
		15:8	DEV[15:8]							

7. OSC——振荡器模块

7.1. 概述

振荡器模块具有多种时钟源和选择特性，从而使其应用非常广泛，同时最大程度地提高性能并降低功耗。图 7-1 给出了振荡器模块的框图。

时钟源可由外部振荡器、石英晶体谐振器和陶瓷谐振器提供。此外，系统时钟源还可以由两个内部振荡器中的一个和 PLL 电路提供，并可通过软件选择时钟速度。其他时钟特性包括：

- 通过软件选择外部或内部系统时钟源。
- 故障保护时钟监视器（FSCM），用来检测外部时钟源（LP、XT、HS、ECH、ECM 和 ECL）故障并自动切换到内部振荡器。
- 振荡器起振定时器（OST）可以确保晶振源的稳定性。

配置字 1 的 RSTOSC 位决定在器件复位后运行（包括初次上电）时使用的振荡器类型。

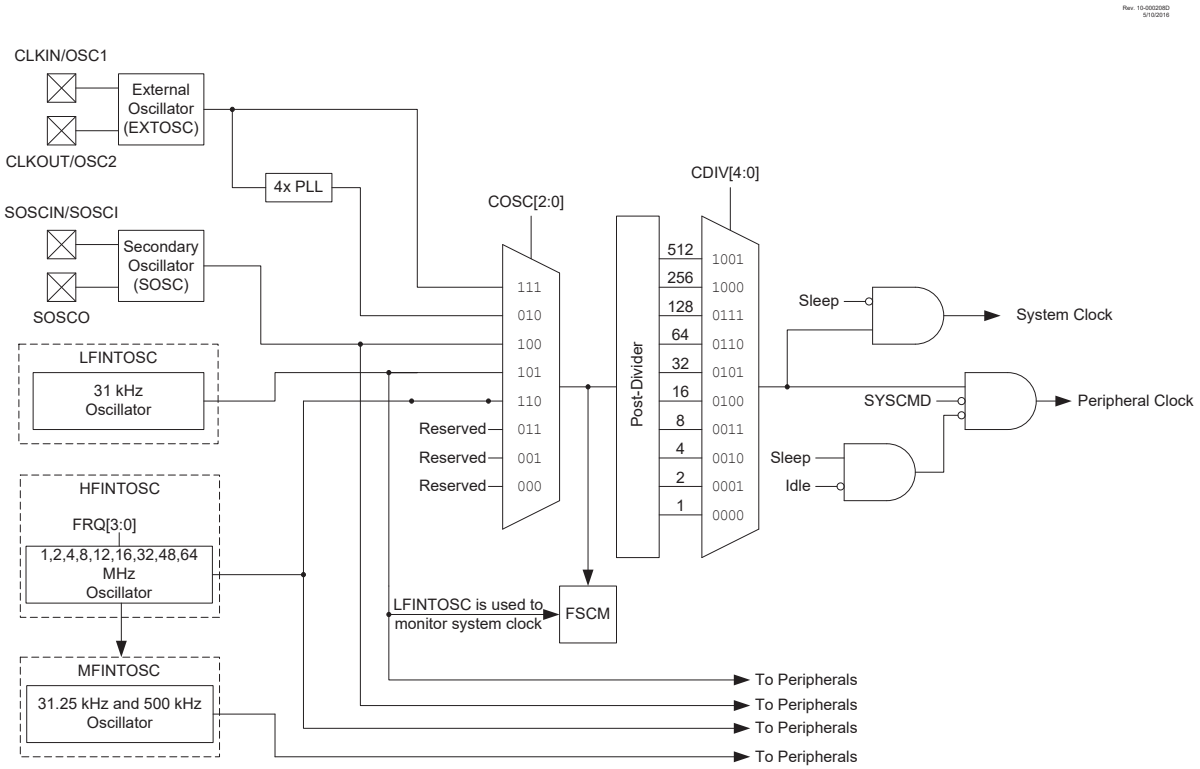
如果选择外部时钟源，必须结合使用配置字 1 的 FEXTOSC 位和 RSTOSC 位来选择外部时钟模式。

可通过设置配置字 1 的 FEXTOSC[2:0]位将外部振荡器模块配置为以下时钟模式之一：

- ECL——外部时钟低功耗模式 （低于 1 MHz）
- ECM——外部时钟中等功耗模式 （1 MHz 至 16 MHz）
- ECH——外部时钟高功耗模式 （高于 16 MHz）
- LP——32 kHz 低功耗晶振模式
- XT——中等增益晶振或陶瓷谐振器振荡器模式（500 kHz 至 4 MHz）
- HS——高增益晶振或陶瓷谐振器模式（高于 4 MHz）

ECH、ECM 和 ECL 时钟模式依靠外部逻辑电平信号作为器件时钟源。LP、XT 和 HS 时钟模式要求器件在外部连接一个晶振或谐振器。针对不同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源，分别用 LFINTOSC 和 HFINTOSC 表示。这两个时钟源可产生多种器件时钟频率。

图 7-1. 简化时钟源框图



7.2. 时钟源类型

时钟源可分为外部时钟源和内部时钟源。

外部时钟源依靠外部电路工作。例如：振荡器模块（ECH、ECM 和 ECL 模式）、石英晶振或陶瓷谐振器（LP、XT 和 HS 模式）。

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，用于产生内容系统时钟源。高频内部振荡器（HFINTOSC）可产生 1、2、4、8、12、16、32、48 和 64 MHz 时钟。可以通过 OSCFRQ 寄存器控制频率。低频内部振荡器（LFINTOSC）产生固定的 31 kHz 频率。

提供 4x PLL，可与外部时钟配合使用。

可通过 NOSC 位在外部或内部时钟源之间选择系统时钟。对于不使用 OSC2 引脚的任何模式，可以在 OSC2/CLKOUT 引脚上产生系统时钟。时钟输出功能由 CONFIG1H 寄存器中的 $\overline{\text{CLKOUTEN}}$ 位管理。如果使能，则时钟输出信号的频率始终为 $F_{OSC}/4$。

7.2.1. 外部时钟源

通过执行以下操作之一，可以使用外部时钟源作为器件系统时钟：

- 编程配置字中的 RSTOSC[2:0]和 FEXTOSC[2:0]位，选择在器件复位时用作默认系统时钟的外部时钟源。
- 写入 NOSC[2:0]和 NDIV[3:0]位，切换系统时钟源。

7.2.1.1. EC 模式

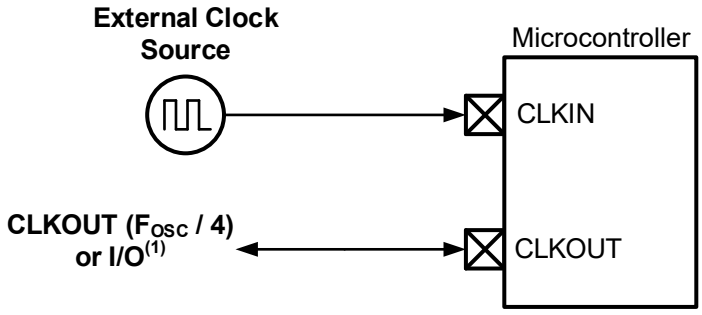
外部时钟（External Clock，EC）模式将外部产生的逻辑电平信号作为系统时钟源。工作在此模式下时，外部时钟源应连接到 OSC1 输入引脚。OSC2/CLKOUT 可用作通用 I/O 或 CLKOUT。下图给出了 EC 模式的引脚连接图。

EC 模式有 3 种功耗模式，可通过配置字进行选择：

- ECH——高功耗，高于 16 MHz
- ECM——中等功耗，1 MHz 至 16 MHz
- ECL——低功耗，低于 1 MHz

当选择 EC 模式时，振荡器起振定时器（OST）被禁止。因此，在上电复位（POR）或从休眠模式唤醒后，不会延时操作。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 7-2. 外部时钟（EC）模式工作原理



注：

1. 输出取决于配置字（CONFIG1H）的 $\overline{\text{CLKOUTEN}}$ 位。

7.2.1.2. LP、XT 和 HS 模式

LP、XT 和 HS 模式支持使用连接到 OSC1 和 OSC2 的石英晶振或陶瓷谐振器（图 7-3）。这三种模式可以选择内部反相放大器的低、中等或高增益设置以支持多种谐振器类型和速度。

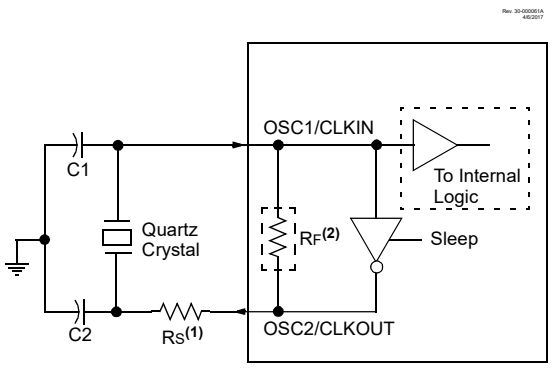
LP 振荡器模式选择内部反相放大器的最低增益设置。LP 模式的电流消耗在这三种模式中最低。此模式只适合于驱动 32.768 kHz 的音叉型晶振（时钟晶体）。

XT 振荡器模式选择内部反相放大器的中等增益设置。XT 模式的电流消耗在这三种模式中居中。该模式最适合驱动具备中等驱动电流要求的谐振器（介于 100 kHz 和 4 MHz 之间）。

HS 振荡器模式选择内部反相放大器的最高增益设置。HS 模式的电流消耗在这三种模式中最高。该模式最适合驱动需要高驱动电流的谐振器（高于 4 MHz）。

图 7-3 和图 7-4 分别给出了石英晶振和陶瓷谐振器的典型电路。

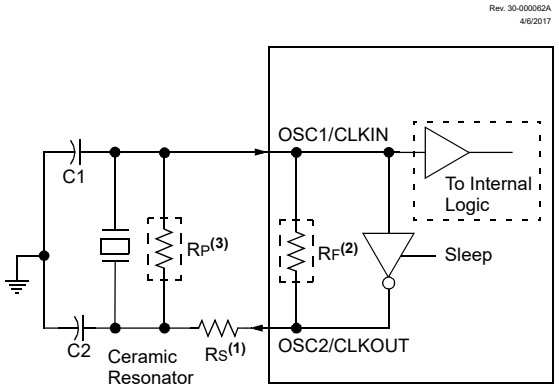
图 7-3. 石英晶振工作原理（LP、XT 或 HS 模式）



注：

1. 具有低驱动能力的石英晶振可能需要一个串联电阻（R_S）。
2. R_F 的值随选定的振荡器模式而变化（通常介于 2 MΩ 和 10 MΩ 之间）。

图 7-4. 陶瓷谐振器工作原理（XT 或 HS 模式）



注：

1. 具有低驱动能力的陶瓷谐振器可能需要一个串联电阻（R_S）。
2. R_F 的值随选定的振荡器模式而变化（通常介于 2 MΩ 和 10 MΩ 之间）。
3. 要让陶瓷谐振器正常工作可能还需要一个并联反馈电阻（R_P）。

7.2.1.3. 振荡器起振定时器（OST）

如果振荡器模块配置为 LP、XT 或 HS 模式，则振荡器起振定时器（OST）将对 OSC1 引脚的振荡计数 1024 次。这发生在上电复位（POR）或从休眠状态唤醒时。OST 确保使用石英晶振或陶瓷谐振器的振荡器电路已起振并且为振荡器模块提供稳定的系统时钟。

7.2.1.4. 4x PLL

振荡器模块包含了一个 4x PLL，它可以与外部时钟源配合使用，用于提供系统时钟源。PLL 的输入频率必须处于规范值范围内。

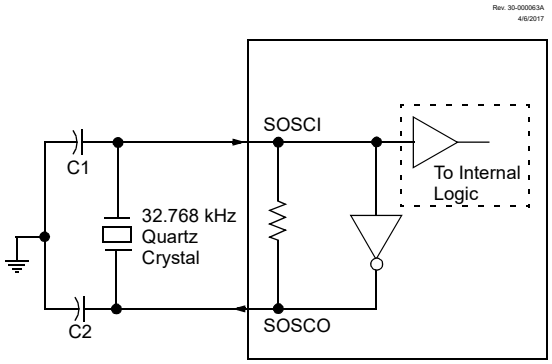
使用时，可以通过以下两种方法之一使能 PLL：

1. 将配置字 1 中的 RSTOSC 位编程为 `010`（使能带有 4x PLL 的 EXTOSC）。
2. 将 `010` 写入 NOSC 位（使能具有 4x PLL 的 EXTOSC）。

7.2.1.5. 辅助振荡器

辅助振荡器是一个单独的振荡器模块，可用作备用系统时钟源。辅助振荡器针对 32.768 kHz 进行了优化，可与连接到 SOSCI 和 SOSCO 器件引脚的外部晶振或连接到 SOSCIN 引脚的外部时钟源配合使用。辅助振荡器可在运行期间通过时钟切换来选择。

图 7-5. 石英晶振工作原理（辅助振荡器）



7.2.2. 内部时钟源

通过执行以下操作之一，可以将器件配置为使用内部振荡器模块作为系统时钟：

- 编程配置字中的 RSTOSC[2:0]位，选择 INTOSC 时钟作为器件复位时的默认系统时钟。
- 写入 NOSC[2:0]位可以在运行时将系统时钟源切换为内部振荡器。

在 INTOSC 模式下，OSC1/CLKIN 可用作通用 I/O。OSC2/CLKOUT 可用作通用 I/O 或 CLKOUT。

OSC2/CLKOUT 引脚的功能由配置字中的 $\overline{\text{CLKOUTEN}}$ 位决定。

内部振荡器模块有两个独立的振荡器，可产生两个内部系统时钟源。

1. **HFINTOSC**（高频内部振荡器）出厂时已校准，工作频率为 1 至 64 MHz。HFINTOSC 的频率可通过 OSCFRQ 频率选择寄存器进行选择，并通过 OSCTUNE 寄存器实现微调。
2. **LFINTOSC**（低频内部振荡器）出厂时已校准，工作频率为 31 kHz。

7.2.2.1. HFINTOSC

高频内部振荡器（HFINTOSC）是一个高精度数字控制内部时钟源，可产生最高 64 MHz 的稳定时钟。HFINTOSC 可通过以下方法之一来使能：

- 将配置字 1 中的 RSTOSC 位设为 `110`（F_{OSC} = 1 MHz）或 `000`（F_{OSC} = 64 MHz），以设置器件上电或复位时的振荡器。
- 在运行期间写入 NOSC 位。

HFINTOSC 频率可通过设置 HFFRQ 位来选择。

NDIV 位允许对 HFINTOSC 输出进行 1:1 到 1:512 之间的分频。

7.2.2.2. MFINTOSC

该模块提供两个（500 kHz 和 31.25 kHz）恒定频率时钟输出。这些时钟是 HFINTOSC 时钟的数字分频器。动态分频器逻辑用于为 HFINTOSC 的所有设置提供恒定 MFINTOSC 时钟速率。

MFINTOSC 无法驱动系统，但可以用作诸如定时器和 WWDT 等特定模块的时钟。

7.2.2.3. 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 的频率是上电延时定时器（PWRT）、窗口看门狗定时器（WWDT）和故障保护时钟监视器（FSCM）的时钟频率。

可以通过以下方法之一使能 LFINTOSC：

- 编程配置字 1 的 RSTOSC[2:0]位来使能 LFINTOSC。

- 在运行期间写入 NOSC[2:0]位。

7.2.2.4. ADCRC（也称为 FRC）

ADCRC 是专用于 ADCC 模块的振荡器。可以使用 ADOEN 位手动使能 ADCRC 振荡器。ADCRC 以 600 kHz 的固定频率运行。如果选择将 ADCRC 作为 ADCC 模块的时钟源，ADCRC 会自动使能。

7.2.3. 振荡器状态与调整

7.2.3.1. 内部振荡器频率调节

内部振荡器在出厂时已校准。该内部振荡器可以通过用软件写入 OSCTUNE 寄存器进行调整。

OSCTUNE 不会影响 LFINTOSC 频率。依赖 LFINTOSC 时钟源频率工作的部件，诸如上电延时定时器（PWRT）、WWDT、故障保护时钟监视器（FSCM）以及外设，它们的工作不受频率更改的影响。

OSCTUNE 寄存器的默认值为 00h。该值是一个 6 位的二进制补码。值 1Fh 将提供对最大频率的调节。值 20h 将提供对最小频率的调节。

当 OSCTUNE 寄存器被修改时，振荡器频率将开始转变为新频率。在频率转变期间代码继续执行。不会有任何迹象表明时钟发生了转变。

7.2.3.2. 振荡器状态和手动使能

每个振荡器（包括 ADCRC 振荡器）的就绪状态均在 OSCSTAT 中显示。可以通过 OSCEN 明确使能振荡器（而非 PLL）。

7.2.3.3. HFOR 和 MFOR 位

HFOR 和 MFOR 位用于指示 HFINTOSC 和 MFINTOSC 是否已就绪。这些时钟始终有效，但只有在就绪时才准确。

当 OSCFRQ 寄存器中装入新值时，HFOR 和 MFOR 位会清零，当振荡器就绪时，会重新置 1。在未决 OSCFRQ 更改期间，MFINTOSC 时钟会停顿在高电平或低电平状态，直到 HFINTOSC 恢复工作。

7.3. 时钟切换

可通过软件使用新振荡器源（NOSC）位在外部和内部时钟源之间切换系统时钟源。可以选择以下时钟源：

- 外部振荡器
- 内部振荡器模块（INTOSC）

> **重要：**配置字 1 中的时钟切换使能位可用于使能或禁止时钟切换功能。清零时，用户软件无法更改 NOSC 和 NDIV 位。置 1 时，允许写入 NOSC 和 NDIV，从而切换时钟频率。

7.3.1. 新振荡器源（NOSC）和新分频比选择请求（NDIV）位

新振荡器源（NOSC）和新分频比选择请求（NDIV）位用于选择 CPU 和外设使用的系统时钟源和频率。

当将 NOSC 和 NDIV 的新值写入 OSCCON1 时，当前振荡器选择将在等待新时钟源指示其已稳定并就绪时继续运行。在某些情况下，新请求的时钟源可能已在使用并立即就绪。对于仅发生分频比更改的情况，新时钟源和旧时钟源相同，因此旧时钟源将立即就绪。器件在等待切换时可能进入休眠状态。

当新振荡器就绪时，新振荡器已就绪（NOSCR）位和 PIR1 的时钟切换中断标志（CSWIF）位均置 1。如果允许时钟切换中断（CSWIE = 1），则此时将产生中断。除了中断，还可查询振荡器就绪（ORDY）位来确定振荡器何时已就绪。

重要：CSWIF 中断不会将系统从休眠模式唤醒。

如果时钟切换保持（CSWHOLD）位清零，在新振荡器已就绪（NOSCR）位置 1 时将发生振荡器切换，并且将以新振荡器设置处理中断（如果允许了中断的话）。

如果 CSWHOLD 置 1，振荡器切换将暂停，而执行将继续使用当前（旧）时钟源。当 NOSCR 位置 1 时，软件将：

- 设置 CSWHOLD = 0 以完成切换，或
- 将 COSC 复制到 NOSC 以放弃切换。

如果 DOZE 有效，则将在下一个时钟周期发生切换，而无论 CPU 在该周期期间是否正在运行。

更改时钟后分频比而不更改时钟源（如，将 F_{OSC} 从 1 MHz 更改为 2 MHz）的处理方式与更改时钟源相同，如前面所述。时钟源已经在工作，因此切换相对较快。CSWHOLD 必须清零（CSWHOLD = 0），切换才能完成。

当前 COSC 和 CDIV 在 OSCCON2 寄存器中指示，直至发生实际切换，此时 OSCCON2 将更新且 ORDY 置 1。NOSCR 由硬件清零以指示切换完成。

7.3.2. PLL 输入切换

如上所述管理 PLL 和任何非 PLL 源之间的切换。当 NOSC 选择 PLL 时建立 PLL 输入，并通过 COSC 设置维持。

当 NOSC 和 COSC 选择 PLL 和不同输入源时，系统继续使用 COSC 设置运行，并且根据 NOSC 使能新振荡器源。当新振荡器就绪（且 CSWHOLD = 0）时，系统操作在切换 PLL 输入且 PLL 获得锁定时暂停。这提供了一种真正的无毛刺时钟切换操作。

重要：如果 PLL 无法锁定，则 FSCM 将会触发。

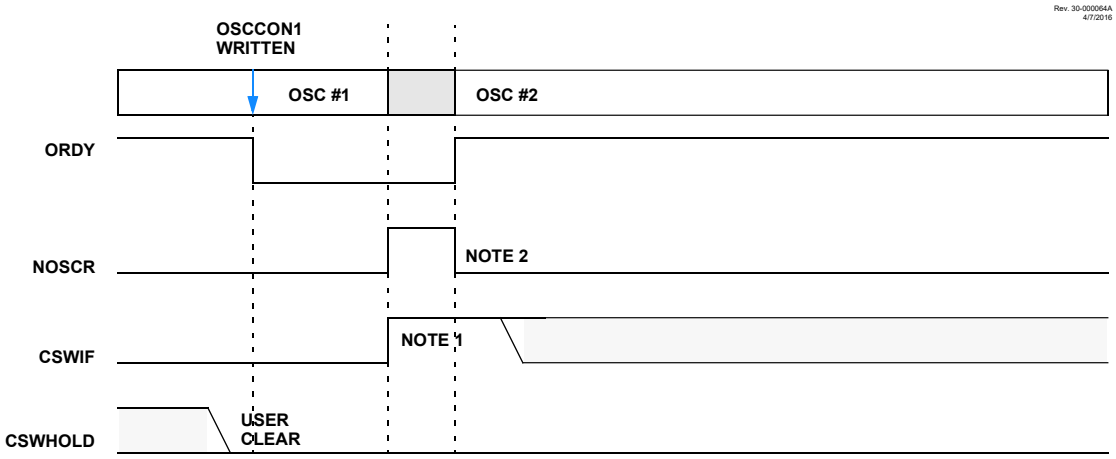
7.3.3. 时钟切换和休眠

如果 OSCCON1 写入新值并且在切换完成之前将器件置于休眠模式，则不会发生切换且器件进入休眠模式。

当器件从休眠状态唤醒且 CSWHOLD 位清零时，器件唤醒时将采用“新”时钟，且时钟切换中断标志（CSWIF）位将置 1。

当器件从休眠模式唤醒且 CSWHOLD 位置 1 时，器件唤醒时将采用“旧”时钟并且再次请求新时钟。

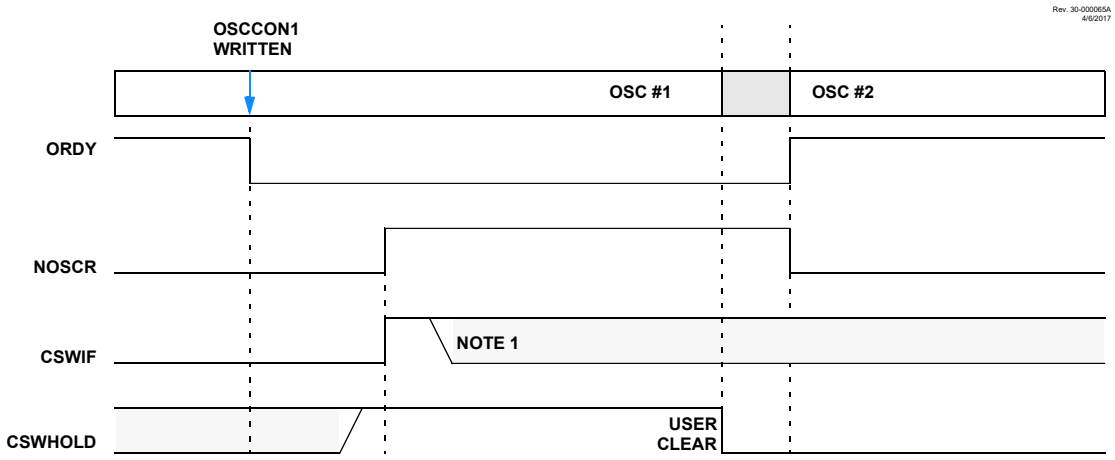
图 7-6. 时钟切换（CSWHOLD = 0）



注：

1. CSWIF 与 NOSCR 同时有效；以 OSC #2 速度处理中断。
2. NOSCR 是否有效对用户是隐藏的，这是因为其仅在切换期间出现。

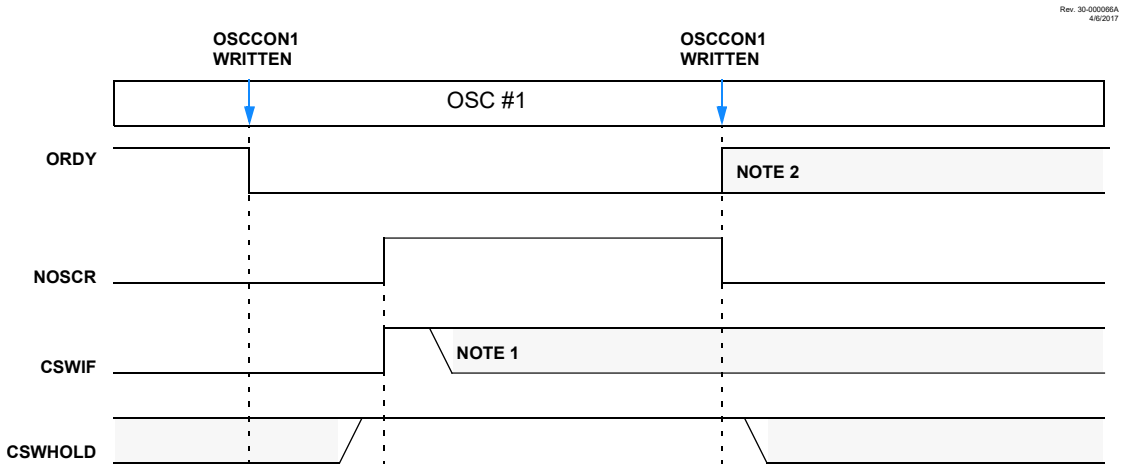
图 7-7. 时钟切换（CSWHOLD = 1）



注：

1. CSWIF 与 NOSCR 同时有效，并且可在将 CSWHOLD 清零之前或之后清零。

图 7-8. 放弃时钟切换



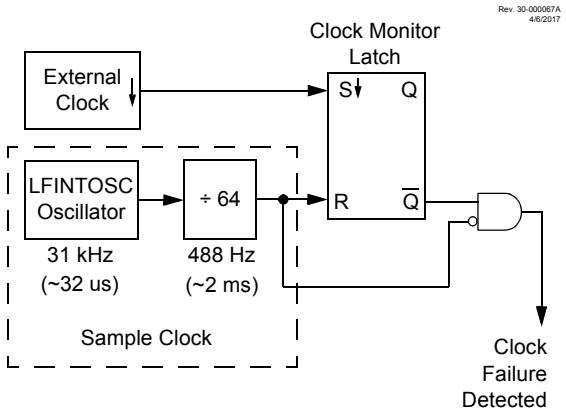
注：

1. CSWIF 可在重写 OSCCON1 之前或之后清零；不自动清零 CSWIF。
2. 如果 OSCCON1 与 OSCCON2 不匹配，则 ORDY = 0；将开始新的切换。

7.4. 故障保护时钟监视器

故障保护时钟监视器（FSCM）旨在使器件能在外部振荡器发生故障时继续运行。通过将配置字 1 寄存器中的 FCMEN 位置 1 使能 FSCM。FSCM 可用于所有外部振荡器模式（LP、XT、HS、ECL/M/H 和辅助振荡器）。

图 7-9. FSCM 框图



7.4.1. 故障保护检测

FSCM 模块通过将外部振荡器和 FSCM 采样时钟进行比较来检测有故障的振荡器。通过对 LFINTOSC 时钟进行 64 分频得到采样时钟。请参见图 7-9。故障检测电路内部有一个锁存器。在外部时钟的每个下降沿上将锁存器置 1。在采样时钟的每个上升沿将锁存器清零。如果采样时钟的一个完整半周期在外部时钟变为低电平之前结束，则会检测到故障。

7.4.2. 故障保护工作原理

当外部时钟出现故障时，FSCM 会覆盖 COSC 位，以选择 HFINTOSC（3'b110）。HFINTOSC 的频率由 HFFRQ 位之前的状态以及 NDIV/CDIV 位确定。PIR1 寄存器的标志位 OSCFIF 置 1。如果 PIE1 寄存器的 OSCFIE 位也置 1，将产生中断。器件固件可采取措施以减轻可能由故障时钟造成的问题。系统时钟将继续

采用内部时钟源，直到器件固件成功地重启外部振荡器并通过写入 NOSC 和 NDIV 位切换回外部时钟源进行工作。

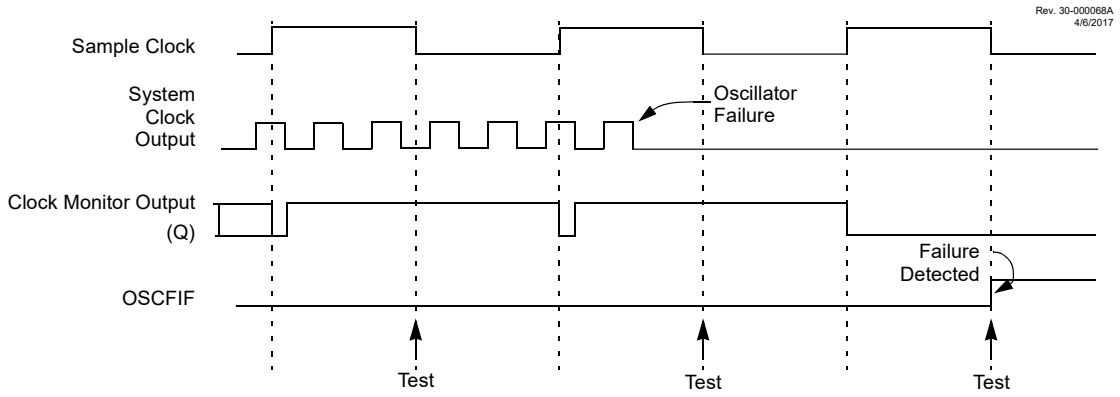
7.4.3. 清除故障保护条件

在复位、执行 SLEEP 指令或更改 OSCCON1 寄存器的 NOSC 和 NDIV 位之后，故障保护条件被清除。当切换到外部振荡器或带 PLL 的外部振荡器时，重启 OST。OST 运行时，器件将依靠 OSCCON1 中选定的 INTOSC 继续工作。如果 OST 超时，在成功切换到外部时钟源后，故障保护条件会被清除。在切换到外部时钟源之前，必须先清零 OSFIF 位。如果故障保护条件仍然存在，硬件会再次将 OSFIF 标志置 1。

7.4.4. 复位或从休眠模式唤醒

FSCM 设计为在振荡器起振定时器（OST）延时结束后检测振荡器故障。从休眠模式唤醒以及任何类型的复位之后都会启动 OST。OST 不与 EC 时钟模式一起使用，因此 FSCM 将在复位或唤醒后立即生效。因此，在使用 EC 模式之一时，器件在 OST 运行期间将始终执行代码。

图 7-10. FSCM 时序图



注：通常，器件时钟的频率比采样时钟频率高很多。本例中选择的相对频率是为了说明起见。

7.5. 寄存器汇总——OSC

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0ED2	保留									
0x0ED3	OSCCON1	7:0		NOSC[2:0]			NDIV[3:0]			
0x0ED4	OSCCON2	7:0		COSC[2:0]			CDIV[3:0]			
0x0ED5	OSCCON3	7:0	CSWHOLD	SOSCPWR		ORDY	NOSCR			
0x0ED6	OSCSTAT	7:0	EXTOR	HFOR	MFOR	LFOR	SOR	ADOR		PLLR
0x0ED7	OSCEN	7:0	EXTOEN	HFOEN	MFOEN	LFOEN	SOSCEN	ADOEN		
0x0ED8	OSCTUNE	7:0			HFTUN[5:0]					
0x0ED9	OSCFRQ	7:0					HFFRQ[3:0]			

7.6. 寄存器定义：振荡器控制

7.6.1. OSCCON1

名称： OSCCON1
偏移量： 0xED3

振荡器控制寄存器 1

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

Bit 6:4 – NOSC[2:0] 新振荡器源请求位(1,2,3)
设置按照表 7-2 请求源振荡器和 PLL 组合。

表 7-1. 默认振荡器设置

CONFIG1[RSTOSC]	SFR 复位值（fff ffff）			初始 F_{OSC} 频率
	NOSC/COSC	NDIV/CDIV	OSCFRQ	
111	111	0000	4 MHz	EXTOSC，频率取决于 FEXTOSC
110	110	0010		F_{OSC} = 1 MHz（4 MHz/4）
101	101	0000		LFINTOSC
100	100	0000		SOSC
011	保留			
010	010	0000	4 MHz	EXTOSC + 4xPLL(4)
001	保留			
000	110	0000	64 MHz	F_{OSC} = 64 MHz

表 7-2. NOSC 位设置

NOSC[2:0]	时钟源
111	EXTOSC(5)
110	HFINTOSC(6)
101	LFINTOSC
100	SOSC
011	保留
010	EXTOSC + 4x PLL(7)
001	保留
000	保留

Bit 3:0 – NDIV[3:0] 新分频比选择请求位(2,3)
设置按照表 7-3 确定后分频器新分频比。

表 7-3. NDIV 位设置

NDIV[3:0]	时钟分频比
1111–1010	保留
1001	512
1000	256
0111	128
0110	64
0101	32
0100	16
0011	8
0010	4
0001	2
0000	1

注：

1. 默认值（f/f）由 CONFIG1[RSTOSC]配置位决定。请参见表 7-1。
2. 如果 NOSC 写入保留的值（表 7-2），则会忽略操作并且不会写入 NOSC。
3. 当 CONFIG1[CSWEN] = 0 时，该寄存器为只读且不能更改为 POR 值以外的值。
4. EXTOSC 必须符合 PLL 规范。
5. EXTOSC 由 CONFIG1[FEXTOSC]配置。
6. HFINTOSC 频率由 HFFRQ 位设置。
7. EXTOSC 必须符合 PLL 规范。

7.6.2. OSCCON2

名称： OSCCON2
偏移量： 0xED4

振荡器控制寄存器 2

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

Bit 6:4 – COSC[2:0] 当前振荡器源选择位（只读）(1,2)
指示当前源振荡器和 PLL 组合，如下表所示。

表 7-4. COSC 位设置

COSC/NOSC	时钟源
111	EXTOSC(3)
110	HFINTOSC(4)
101	LFINTOSC
100	SOSC
011	保留
010	EXTOSC + 4x PLL(5)
001	保留
000	保留

Bit 3:0 – CDIV[3:0] 当前分频比选择位（只读）(1,2)
指示当前后分频比，如下表所示。

表 7-5. CDIV 位设置

CDIV/NDIV	时钟分频比
1111-1010	保留
1001	512
1000	256
0111	128
0110	64
0101	32
0100	16
0011	8
0010	4
0001	2
0000	1

注：

1. POR 值是开始执行用户代码时的值。
2. 复位值（q/q）与 NOSC/NDIV 位相同。
3. EXTOSC 由 CONFIG1[FEXTOSC]位配置。
4. HFINTOSC 频率由 HFFRQ 位设置。
5. EXTOSC 必须符合 PLL 规范。

7.6.3. OSCCON3

名称： OSCCON3
偏移量： 0xED5

振荡器控制寄存器 3

位	7	6	5	4	3	2	1	0
	CSWHOLD	SOSCPWR		ORDY	NOSCR			
访问	R/W/HC	R/W		RO	RO			
复位	0	1		0	0			

Bit 7 – CSWHOLD 时钟切换保持位

值	说明
1	当 NOSC 选择的振荡器就绪时，时钟切换将暂停（中断）
0	当 NOSC 选择的振荡器就绪时，时钟切换可继续；当 NOSCR 变为 1 时，将发生切换

Bit 6 – SOSCPWR 辅助振荡器功耗模式选择位

值	说明
1	辅助振荡器工作在高功耗模式下
0	辅助振荡器工作在低功耗模式下

Bit 4 – ORDY 振荡器就绪位（只读）

值	说明
1	OSCCON1 = OSCCON2；当前的系统时钟是由 NOSC 指定的时钟
0	正在进行时钟切换

Bit 3 – NOSCR 新振荡器就绪位（只读）**(1)**

值	说明
1	正在进行时钟切换并且 NOSC 所选的振荡器指示“就绪”条件
0	未在进行时钟切换，或者 NOSC 选择的振荡器尚未就绪

注：

1. 如果 CSWHOLD = 0，则用户可能无法看到该位置 1，因为该位置 1 的时间小于一个指令周期。

7.6.4. OSCSTAT

名称： OSCSTAT
偏移量： 0xED6

振荡器状态寄存器 1

位	7	6	5	4	3	2	1	0
	EXTOR	HFOR	MFOR	LFOR	SOR	ADOR		PLLR
访问	RO	RO	RO	RO	RO	RO		RO
复位	q	q	q	q	q	q		q

Bit 7 – EXTOR EXTOSC（外部）振荡器就绪位

值	说明
1	振荡器已就绪备用
0	振荡器未使能，或尚未就绪备用

Bit 6 – HFOR HFINTOSC 振荡器就绪位

值	说明
1	振荡器已就绪备用
0	振荡器未使能，或尚未就绪备用

Bit 5 – MFOR MFINTOSC 振荡器就绪位

值	说明
1	振荡器已就绪备用
0	振荡器未使能，或尚未就绪备用

Bit 4 – LFOR LFINTOSC 振荡器就绪位

值	说明
1	振荡器已就绪备用
0	振荡器未使能，或尚未就绪备用

Bit 3 – SOR 辅助（Timer1）振荡器就绪位

值	说明
1	振荡器已就绪备用
0	振荡器未使能，或尚未就绪备用

Bit 2 – ADOR ADC 振荡器就绪位

值	说明
1	振荡器已就绪备用
0	振荡器未使能，或尚未就绪备用

Bit 0 – PLLR PLL 就绪位

值	说明
1	PLL 已就绪备用
0	PLL 未使能，所需输入源未就绪，或 PLL 未锁定

7.6.5. OSCFRQ

名称： OSCFRQ
偏移量： 0xED9

HFINTOSC 频率选择寄存器

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

Bit 3:0 – HFFRQ[3:0] HFINTOSC 频率选择位

HFFRQ	标称频率（MHz）
1001	保留
1010	
1111	
1110	
1101	
1100	
1011	
1000(1)	64
0111	48
0110	32
0101	16
0100	12
0011	8
0010(1)	4
0001	2
0000	1

注：

1. 更多信息，请参见表 7-1。

7.6.6. OSCTUNE

名称： OSCTUNE
偏移量： 0xED8

HFINTOSC 调节寄存器

位	7	6	5	4	3	2	1	0
			HFTUN[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 – HFTUN[5:0] HFINTOSC 频率调节位

值	说明
01 1111	最高频率
00 0000	中心频率振荡器模块以校准的频率运行（默认值）。
10 0000	最低频率

7.6.7. OSCEN

名称： OSCEN
偏移量： 0xED7

振荡器手动使能寄存器

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

Bit 7 – EXTOEN 外部振荡器手动请求使能位

值	说明
1	已明确使能 EXTOSC，具体操作由 CONFIG1[FEXTOSC]指定
0	仅在外设请求时使能 EXTOSC

Bit 6 – HFOEN HFINTOSC 振荡器手动请求使能位

值	说明
1	已明确使能 HFINTOSC，具体操作由 OSCFRQ 指定
0	仅在外设请求时使能 HFINTOSC

Bit 5 – MFOEN MFINTOSC（500 kHz/31.25 kHz）振荡器手动请求使能位（源自 HFINTOSC）

值	说明
1	已明确使能 MFINTOSC
0	仅在外设请求时使能 MFINTOSC

Bit 4 – LFOEN LFINTOSC（31 kHz）振荡器手动请求使能位

值	说明
1	已明确使能 LFINTOSC
0	仅在外设请求时使能 LFINTOSC

Bit 3 – SOSCEN 辅助振荡器手动请求使能位

值	说明
1	已明确使能辅助振荡器，具体操作由 SOSCPWR 指定
0	仅在外设请求时使能辅助振荡器

Bit 2 – ADOEN ADC 振荡器手动请求使能位

值	说明
1	已明确使能 ADC 振荡器
0	仅在外设请求时使能 ADC 振荡器

8. REFCLK——参考时钟输出模块

参考时钟输出（REFCLK）模块能够将时钟信号发送到参考时钟输出引脚（CLKR）。参考时钟输出信号还可在内部路由至其他外设（如数据信号调制器（DSM）、存储器扫描器和定时器模块）。

参考时钟输出模块具有以下特性：

- 可通过 CLKRCLK 寄存器选择时钟源
- 可编程时钟分频比
- 可选占空比

图 8-1. 参考时钟框图

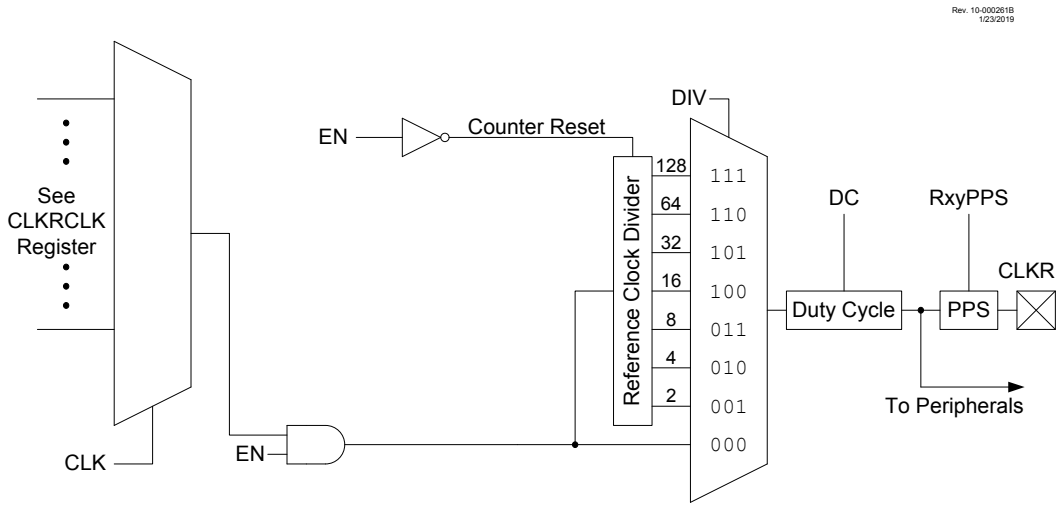
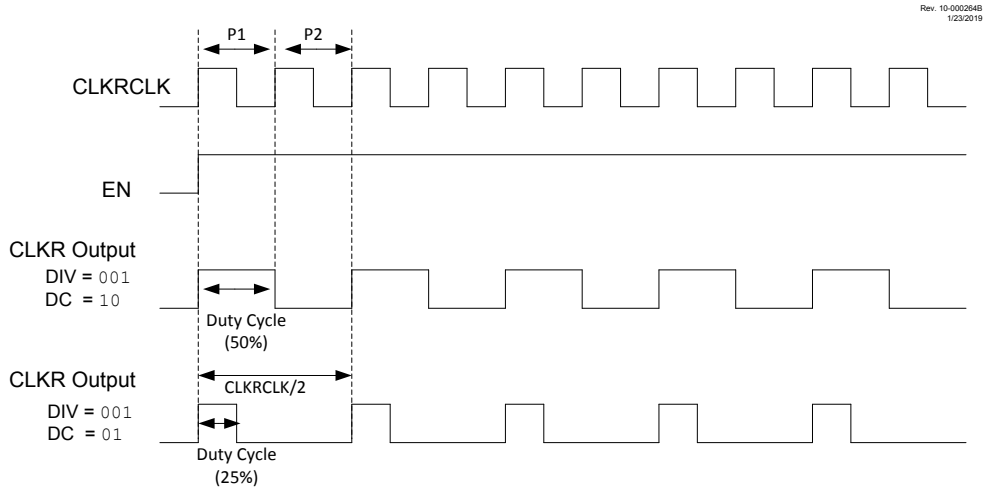


图 8-2. 参考时钟时序



8.1. 时钟源

参考时钟外设的时钟源通过 CLKRCLK 寄存器中的 CLK 位来选择。下面列出了可用的时钟源：

- CLC
- 辅助振荡器
- MFINTOSC
- LFINTOSC
- F_{OSC}

8.1.1. 时钟同步

当通过将 CLKRCON 寄存器中的 EN 位置 1 来启动模块和使能 CLKR 输出时，可确保 CLKR 输出信号无毛刺。

当禁止参考时钟输出时，输出信号将立即禁止。

当使能模块时，可更改时钟分频比和时钟占空比，但这样做可能会导致输出端出现毛刺。为了避免可能出现的毛刺，可仅在 EN 清零时更改时钟分频比和时钟占空比。

8.2. 可编程时钟分频比

模块接收时钟输入并根据 DIV 位的值对其进行分频。

以下配置可用：

- 基本 F_{OSC} 值
- F_{OSC} 2 分频
- F_{OSC} 4 分频
- F_{OSC} 8 分频
- F_{OSC} 16 分频
- F_{OSC} 32 分频
- F_{OSC} 64 分频
- F_{OSC} 128 分频

时钟分频值可在模块使能时更改。但是，为了避免输出端产生毛刺，应仅在模块禁止（EN = 0）时更改 DIV 位。

8.3. 可选占空比

CLKRCON 寄存器中的 DC 位用于修改输出时钟的占空比。当 DIV 值不为 000 时，可以为所有时钟速率选择 0%、25%、50%或 75%的占空比。当 DIV = 000 时，所有 DC 值的占空比默认为 50%，但 DC 值等于 00 时除外，这种情况下的占空比为 0%（恒定低电平输出）。

重要：复位时的 DC 值为 10。因此，默认占空比为 50%，而非 0%。

重要：当使能模块时，可更改时钟分频比和时钟占空比，但这样做可能会导致输出端出现毛刺。为了避免可能出现的毛刺，可仅在模块禁止（EN = 0）时更改时钟分频比和时钟占空比。

8.4. 在休眠模式下工作

参考时钟模块可以在休眠模式下继续工作，并且为除 F_{OSC}（CLK = 0）外的所有时钟源选项提供信号输出。

8.5. 寄存器定义：参考时钟

8.5.1. CLKRCON

名称： CLKRCON
偏移量： 0xF39

参考时钟控制寄存器

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

Bit 7 – EN

参考时钟模块使能位

值	说明
1	使能参考时钟模块
0	禁止参考时钟模块

Bit 4:3 – DC[1:0]

参考时钟占空比位(1)

值	说明
11	时钟输出占空比为 75%
10	时钟输出占空比为 50%
01	时钟输出占空比为 25%
00	时钟输出占空比为 0%

Bit 2:0 – DIV[2:0]

参考时钟分频比位

值	说明
111	基本时钟值被 128 分频
110	基本时钟值被 64 分频
101	基本时钟值被 32 分频
100	基本时钟值被 16 分频
011	基本时钟值被 8 分频
010	基本时钟值被 4 分频
001	基本时钟值被 2 分频
000	基本时钟值

注：

1. 这些位对于为 2 或更大的参考时钟分频比值有效，基本时钟无法再进一步分频。

8.5.2. CLKRCLK

名称： CLKRCLK
偏移量： 0xF3A

参考时钟选择多路开关

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

Bit 3:0 – CLK[3:0] CLKR 时钟选择位

表 8-1. CLKR 时钟源

CLK	时钟源
1111	CLC8_out(1)
1110	CLC7_out(1)
1101	CLC6_out(1)
1100	CLC5_out(1)
1011	CLC4_out(1)
1010	CLC3_out(1)
1001	CLC2_out(1)
1000	CLC1_out(1)
0111–0101	**保留**
0100	SOSC
0011	MFINTOSC（500 kHz）
0010	LFINTOSC（31 kHz）
0001	HFINTOSC
0000	F_{OSC}

注：
1. CN2510 器件上未提供。

8.6. 寄存器汇总——参考时钟

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0F38	保留									
0x0F39	CLKRCON	7:0	EN			DC[1:0]		DIV[2:0]		
0x0F3A	CLKRCLK	7:0					CLK[3:0]			

9. 节能工作模式

掉电模式的目的是降低功耗。有三种掉电模式：

- 打盹模式
- 空闲模式
- 休眠模式

9.1. 打盹模式

打盹模式通过减少 CPU 操作和闪存程序存储器（Program Flash Memory，PFM）访问而不影响外设操作来支持节能。打盹模式与休眠模式的不同之处在于，带隙和系统振荡器继续运行，仅 CPU 和 PFM 受影响。通过消除 CPU 和存储器中的不必要操作来减少操作执行，从而节省功耗。

当打盹使能位置 1（DOZEN = 1）时，CPU 每 N 个周期仅执行一个指令周期（通过 DOZE 位定义）。例如，如果 DOZE = 001，则指令周期比是 1:4。CPU 和存储器执行一个指令周期，然后在三个指令周期内保持空闲。未使用的周期期间，外设继续以系统时钟速度运行。

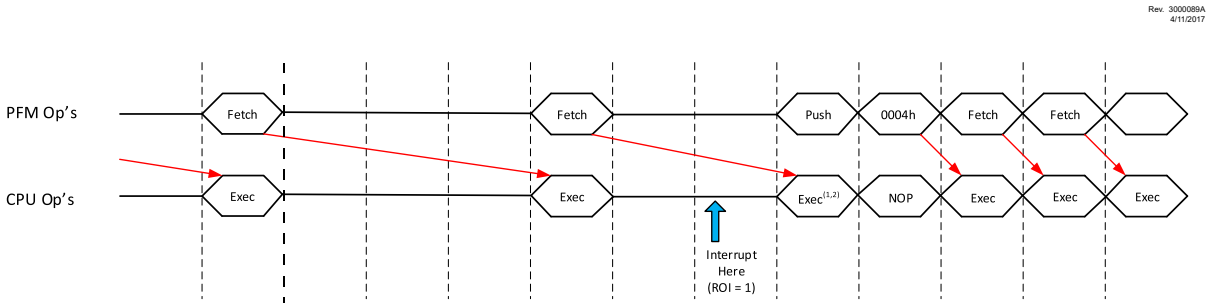
9.1.1. 打盹操作

图 9-1 给出了打盹操作。对于本例：

- 打盹已使能（DOZEN = 1）
- DOZE = 001（比为 1:4）
- 中断恢复已使能（ROI = 1）

与正常操作一样，PFM 为下一个指令周期取指。至外设的 Q 时钟继续运行。

图 9-1. 打盹模式操作示例（DOZE[2:0] = 001，1:4）



Note 1: Multi-cycle instructions are executed to completion before fetching 0004h.
2: If the pre-fetched instruction clears GIE, the ISR will not occur, but DOZEN is still cleared and the CPU will resume execution at full speed.

9.1.2. 打盹期间的中断

如果发生中断并且在中断时中断恢复位清零（ROI = 0），中断服务程序（Interrupt Service Routine，ISR）继续以 DOZE[2:0]选择的速率执行。可通过 DOZE[2:0]比延长中断延时。

如果发生中断并且在中断时 ROI 位置 1（ROI = 1），DOZEN 位清零且 CPU 以全速执行。执行预取指令，然后执行中断向量序列。在图 9-1 中，在打盹周期的第 2 个指令周期发生中断，并立即使 CPU 退出打盹模式。如果在执行 RETFIE 操作时退出打盹（Doze-On-Exit，DOE）位置 1（DOE = 1），则 DOZEN 置 1，并且 CPU 基于 DOZE[2:0]比降速执行。

例 9-1. 打盹软件

```
//Mainline operation
bool somethingToDo = FALSE:
void main()
```

```
{
initializeSystem();
// DOZE = 64:1 (for example)
// ROI = 1;
GIE = 1; // enable interrupts
while (1)
{
// If ADC completed, process data
if (somethingToDo)
{
doSomething();
DOZEN = 1; // resume low-power
}
}
}
// Data interrupt handler
void interrupt()
{
// DOZEN = 0 because ROI = 1
if (ADIF)
{
somethingToDo = TRUE;
DOE = 0; // make main() go fast
ADIF = 0;
}
// else check other interrupts...
if (TMR0IF)
{
timerTick++;
DOE = 1; // make main() go slow
TMR0IF = 0;
}
}
```

9.2. 休眠模式

通过执行 SLEEP 指令，并且 CPUDOZE 寄存器的空闲使能（IDLEN）位清零（IDLEN = 0）时，器件进入休眠模式。如果在 IDLEN 位置 1（IDLEN = 1）时执行 SLEEP 指令，CPU 将进入空闲模式。

进入休眠模式时，存在以下情况：

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. STATUS 寄存器的 $\overline{PD}$ 位清零。
3. STATUS 寄存器的 $\overline{TO}$ 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 31 kHz LFINTOSC、HFINTOSC 和 SOSC 作为时钟源或者 OSCEN 寄存器的 HFOEN、LFOEN 或 SOSCEN 位置 1，则 LFINTOSC、HFINTOSC 和 SOSC 将保持使能状态。
6. 如果选择了 FRC 振荡器，则 ADC 不受影响。ADC 时钟不是 FRC 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）。

有关外设在休眠期间工作的详细信息，请参见各个外设对应的章节。

为了最大限度地降低电流消耗，应考虑以下条件：

- I/O 引脚不得悬空
- I/O 引脚的外部电路灌电流
- I/O 引脚的内部电路拉电流
- 带有内部弱上拉电阻的引脚的电流消耗
- 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，需在外部将为高阻抗输入的 I/O 引脚拉到 V_{DD} 或 V_{SS}。

可能拉电流的内部电路示例包括如 DAC 和 FVR 模块之类的模块。

9.2.1. 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. 低功耗欠压复位（LPBOR）（如果使能）。
4. POR 复位。
5. 窗口看门狗定时器（如果使能）。
6. 除时钟切换中断外的所有中断源都可以唤醒器件。

前五个事件会导致器件复位。最后一个事件视为继续执行程序。要确定是发生器件复位还是唤醒事件，请参见“**确定复位原因**”一节。

当执行 SLEEP 指令时，下一条指令（PC + 2）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，对于不在 PIR0 中的每个中断，还必须允许外设中断允许位（PEIE = 1）。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。如果不想执行 SLEEP 指令后的指令，用户需在 SLEEP 指令后面放置一条 NOP 指令。

器件从休眠模式唤醒时，WDT 将清零，与唤醒源无关。

从休眠模式唤醒后，内核将等待至出现三个条件的组合后再开始执行操作。这三个条件分别为：

- PFM 就绪
- COSC 选择的振荡器就绪
- BOR 就绪（禁止 BOR 时除外）

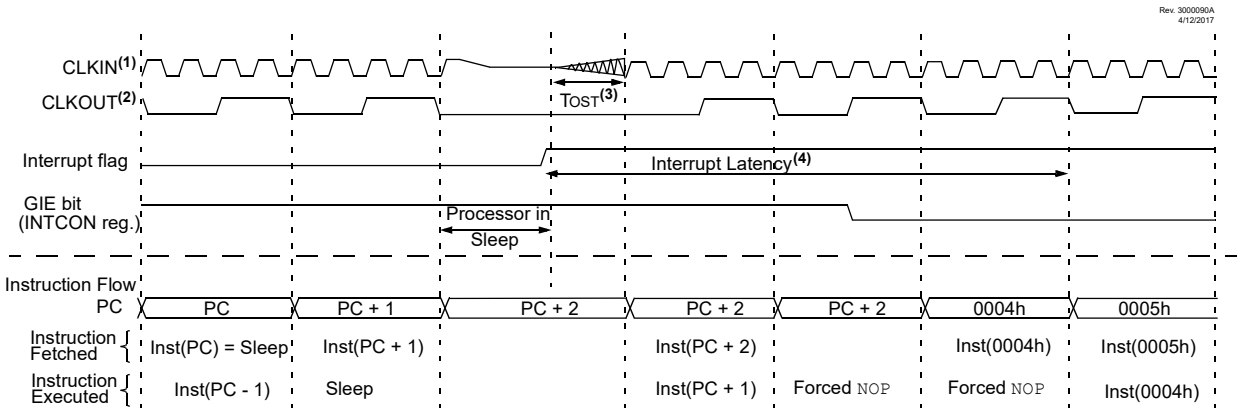
9.2.2. 使用中断唤醒

当禁止全局中断（GIE 被清零）时，并且任一中断源（时钟切换中断除外）的中断允许位和中断标志位都置 1，将会发生以下事件之一：

- 如果在执行 SLEEP 指令之前发生中断
  - SLEEP 指令将作为 NOP 指令执行
  - WDT 和 WDT 预分频器不会被清零
  - STATUS 寄存器的 $\overline{TO}$ 位不会被置 1
  - STATUS 寄存器的 $\overline{PD}$ 位不会被清零
- 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
  - STATUS 寄存器的 $\overline{TO}$ 位将被置 1
  - STATUS 寄存器的 $\overline{PD}$ 位将被清零

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。

图 9-2. 通过中断将器件从休眠模式唤醒



注：

1. 外部时钟。假设采用高、中或低功耗模式。
2. 此处显示的 CLKOUT 用于时序参考。
3. T_{OST} = 1024 T_{OSC}。此延时不适用于 EC 和 INTOSC 振荡器模式。
4. 假定 GIE = `1`。这种情况下，处理器被唤醒后，将调用 0004h 处的 ISR。如果 GIE = `0`，程序继续执行。

9.2.3. 低功耗休眠模式

CN2710 器件系列包含一个内部低压差（Low Dropout，LDO）稳压器，它让器件 I/O 引脚可以使用最高 5.5V 的电压工作，而内部器件逻辑可以使用较低的电压工作。在器件处于休眠模式时，LDO 及其相关的参考电压电路必须保持活动状态。

CN2710 器件允许用户根据应用需求来优化休眠模式下的工作电流。

通过将 VREGCON 寄存器的 VREGPM 位置 1，可以选择低功耗休眠模式。

9.2.3.1. 休眠电流与唤醒时间

在默认工作模式下，处于休眠模式时，LDO 和参考电路会保持为正常配置。由于所有电路都保持活动状态，所以器件能够快速地退出休眠模式。在低功耗休眠模式下，从休眠模式中唤醒时，这些电路需要一个额外的延时，然后才会恢复为正常配置并稳定下来。

低功耗休眠模式对于需要长时间处于休眠模式的应用非常有益。正常模式对于需要快速地、频繁地从休眠模式中唤醒的应用非常有益。

9.2.3.2. 休眠模式下的外设使用

选择低功耗休眠模式时，一些可以在休眠模式下工作的外设将无法正常工作。低功耗休眠模式旨在用于以下外设：

- 欠压复位（BOR）
- 窗口看门狗定时器（WWDT）
- 外部中断引脚/电平变化中断引脚
- 依靠外部辅助时钟源运行的外设

最终用户负责确定在进行 VREGPM 设置以确保休眠模式下的操作时其应用可接受的外设。

9.3. 空闲模式

当 IDLEN 置 1（IDLEN = `1`）时，`SLEEP` 指令将器件置于空闲模式。在空闲模式下，CPU 和存储器操作暂停，但外设时钟继续运行。该模式与打盹模式类似，但在空闲模式下 CPU 和 PFM 关闭。

重要：如果 $\overline{\text{CLKOUTEN}}$ 使能（$\overline{\text{CLKOUTEN}}$ = 0，配置字 1H），输出在空闲模式下将继续工作。

9.3.1. 空闲和中断

发生中断时退出空闲模式（即使 GIE = 0 也是如此），但 IDLEN 不变。器件可通过执行 SLEEP 指令重新进入空闲模式。

如果同时使能中断恢复（ROI = 1）和打盹模式，则发生中断时器件退出空闲模式并且 CPU 恢复全速执行。

9.3.2. 空闲和 WWDT

在空闲模式下，WWDT 复位被禁止，但会将器件唤醒。WWDT 唤醒不是中断，因此 ROI 不适用。

重要：WWDT 可使器件退出空闲模式，与器件退出休眠模式的方式相同。DOZEN 位不受影响。

9.4. 节能模式下的外设操作

在空闲和打盹模式下，所有选定时钟源以及依靠它们运行的外设均处于工作状态。仅在休眠模式下，F_{OSC} 和 $F_{OSC}/4$ 时钟不可用。对于所有其他时钟源，如果在器件进入休眠模式之前手动或通过外设时钟选择将其使能，则它们均处于工作状态。

9.5. 寄存器定义：节能控制

9.5.1. VREGCON

名称： VREGCON
偏移量： 0xEDA

注：

1. 在 ULP 模式下，系统和外设输入不得超过 500 kHz。

稳压器控制寄存器

位	7	6	5	4	3	2	1	0
			PMSYS[1:0]				VREGPM[1:0]	
访问			RO	RO			R/W	R/W
复位			g	g			1	0

Bit 5:4 – PMSYS[1:0] 系统电源模式状态位

值	说明
11	保留
10	ULP 稳压器处于工作状态
01	LP 模式下的主稳压器处于工作状态
00	HP 模式下的主稳压器处于工作状态

Bit 1:0 – VREGPM[1:0] 稳压器功耗模式选择位

值	说明
11	保留。不要使用。
10	ULP 稳压器(1)
01	主稳压器处于 LP 模式
00	主稳压器处于 HP 模式

9.5.2. CPUDOZE

名称： CPUDOZE
偏移量： 0xED2

打盹和空闲寄存器

位	7	6	5	4	3	2	1	0
	IDLEN	DOZEN	ROI	DOE		DOZE[2:0]		
访问	R/W	R/W/HC/HS	R/W	R/W		R/W	R/W	R/W
复位	0	0	0	0		0	0	0

Bit 7 – IDLEN 空闲使能位

复位状态： POR/BOR = 0
所有其他复位时的值 = u

值	说明
1	SLEEP 指令禁止 CPU 时钟，但不禁止外设时钟
0	SLEEP 指令将器件置于完全休眠模式

Bit 6 – DOZEN
打盹使能位(1)

值	说明
1	CPU 根据 DOZE 设置执行指令周期
0	CPU 执行所有指令周期（最快和最高功耗操作）

Bit 5 – ROI 中断恢复位

值	说明
1	进入中断服务程序（ISR）使 DOZEN = 0，CPU 全速运行
0	进入中断不改变 DOZEN 位状态

Bit 4 – DOE 退出打盹位

值	说明
1	执行 RETFIE 使 DOZEN = 1，CPU 降速运行
0	RETFIE 不改变 DOZEN 位状态

Bit 2:0 – DOZE[2:0] CPU 指令周期与外设指令周期之比

值	说明
111	1:256
110	1:128
101	1:64
100	1:32
011	1:16
010	1:8
001	1:4
000	1:2

注：

1. 当 ROI = 1 或 DOE = 1 时，通过进入和/或退出硬件中断改变 DOZEN 位状态。

9.6. 寄存器汇总——节能控制

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0ED1	保留									
0x0ED2	CPUDOZE	7:0	IDLEN	DOZEN	ROI	DOE		DOZE[2:0]		
0x0ED3 ... 0x0ED9	保留									
0x0EDA	VREGCON	7:0			PMSYS[1:0]				VREGPM[1:0]	

10. PMD——外设模块禁止

CN2710 器件具备禁止所选模块的功能，可将这些模块置于功耗尽可能低的模式。

重要：为与旧款器件兼容，所有模块在器件复位后默认设为开启状态。

10.1. 禁止模块

将 PMDx 寄存器中相应的外设禁止位置 1 可以禁止外设。禁止模块有以下影响：

- 模块的所有时钟和控制输入暂停；不存在任何逻辑转换，模块无法正常工作
- 模块保持在复位状态
  - 禁止写入特殊功能寄存器（Special Function Register，SFR）
  - 读操作返回 0x00
- 禁止模拟输出；数字输出读为 0

10.2. 使能模块

当寄存器位清零时，模块重新使能并处于复位状态；SFR 数据将反映 POR 复位值。根据模块的不同，可能需要最多一个完整指令周期，模块才能变为活动状态。

重要：重新使能模块后，必须至少在一个指令周期内不要与模块有任何交互（例如，写寄存器）。

10.3. 寄存器定义：外设模块禁止

10.3.1. PMD0

名称： PMD0
偏移量： 0xEDC

PMD 控制寄存器 0

位	7	6	5	4	3	2	1	0
	SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD
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 – SYSCMD 禁止外设系统时钟网络位
禁止系统时钟网络(1)

值	说明
1	禁止系统时钟网络（F_{OSC}）
0	使能系统时钟网络

Bit 6 – FVRMD 禁止固定参考电压位

值	说明
1	禁止 FVR 模块
0	使能 FVR 模块

Bit 5 – HLVDMD 禁止高/低电压检测位

值	说明
1	禁止 HLVD 模块
0	使能 HLVD 模块

Bit 4 – CRCMD 禁止 CRC 引擎位

值	说明
1	禁止 CRC 模块
0	使能 CRC 模块

Bit 3 – SCANMD 禁止 NVM 存储器扫描器位
禁止扫描器模块(2)

值	说明
1	禁止 NVM 存储器扫描模块
0	使能 NVM 存储器扫描模块

Bit 2 – NVMMD NVM 模块禁止位
禁止 NVM 模块(3)

值	说明
1	禁止读写所有存储器；不能写入 NVMCON 寄存器
0	使能 NVM 模块

Bit 1 – CLKRMD 禁止时钟参考位

值	说明
1	禁止 CLKR 模块
0	使能 CLKR 模块

Bit 0 – IOCMD 禁止所有端口的电平变化中断位

值	说明
1	禁止 IOC 模块
0	使能 IOC 模块

注：

1. 清零 SYSCMD 位将禁止外设的系统时钟（F_{OSC}），但不会影响时钟为 $F_{OSC}/4$ 的外设。
2. *受配置字 4* 中的 SCANE 位影响。
3. 使能 NVM 时，访问数据之前需要最长 1 μs 的延时。

10.3.2. PMD1

名称： PMD1
偏移量： 0xEDD

PMD 控制寄存器 1

位	7	6	5	4	3	2	1	0
		TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD
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 0, 1, 2, 3, 4, 5, 6 – TMRnMD 禁止定时器 n 位

值	说明
1	禁止 TMRn 模块
0	使能 TMRn 模块

10.3.3. PMD2

名称： PMD2
偏移量： 0xEDE

PMD 控制寄存器 2

位	7	6	5	4	3	2	1	0
		DACMD	ADCMD			CMP2MD	CMP1MD	ZCDMD
访问		R/W	R/W			R/W	R/W	R/W
复位		0	0			0	0	0

Bit 6 – DACMD 禁止 DAC 位

值	说明
1	禁止 DAC 模块
0	使能 DAC 模块

Bit 5 – ADCMD 禁止 ADC 位

值	说明
1	禁止 ADC 模块
0	使能 ADC 模块

Bit 1, 2 – CMPnMD 禁止比较器 CMPn 位

值	说明
1	禁止 CMPn 模块
0	使能 CMPn 模块

Bit 0 – ZCDMD 禁止过零检测模块位(1)

值	说明
1	禁止 ZCD 模块
0	使能 ZCD 模块

注：

1. *受配置字 2* 中的 $\overline{\text{ZCD}}$ 位影响。

10.3.4. PMD3

名称： PMD3
偏移量： 0xEDF

PMD 控制寄存器 3

位	7	6	5	4	3	2	1	0
	CLC8MD	CLC7MD	CLC6MD	CLC5MD	PWM4MD	PWM3MD	CCP2MD	CCP1MD
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 4, 5, 6, 7 – CLCnMD 禁止 CLCn 位

值	说明
1	禁止 CLCn 模块
0	使能 CLCn 模块

Bit 2, 3 – PWMnMD 禁止 PWMn 位

值	说明
1	禁止 PWMn 模块
0	使能 PWMn 模块

Bit 0, 1 – CCPnMD 禁止捕捉/比较/PWMn 位

值	说明
1	禁止 CCPn 模块
0	使能 CCPn 模块

10.3.5. PMD4

名称： PMD4
偏移量： 0xEE0

PMD 控制寄存器 4

位	7	6	5	4	3	2	1	0
	UART2MD	UART1MD	MSSP2MD	MSSP1MD				CWG1MD
访问	R/W	R/W	R/W	R/W				R/W
复位	0	0	0	0				0

Bit 6, 7 – UARTnMD 禁止 EUSARTn 位

值	说明
1	禁止 EUSARTn 模块
0	使能 EUSARTn 模块

Bit 4, 5 – MSSPnMD 禁止 MSSPn 位

值	说明
1	禁止 MSSPn 模块
0	使能 MSSPn 模块

Bit 0 – CWG1MD 禁止 CWG1 模块位

值	说明
1	禁止 CWG1 模块
0	使能 CWG1 模块

10.3.6. PMD5

名称： PMD5
偏移量： 0xEE1

PMD 控制寄存器 5

位	7	6	5	4	3	2	1	0
	CLC4MD	CLC3MD	CLC2MD	CLC1MD				DSMMD
访问	R/W	R/W	R/W	R/W				R/W
复位	0	0	0	0				0

Bit 4, 5, 6, 7 – CLCnMD 禁止 CLCn 位

值	说明
1	禁止 CLCn 模块
0	使能 CLCn 模块

Bit 0 – DSMMD 禁止 DSM 位

值	说明
1	禁止 DSM 模块
0	使能 DSM 模块

10.4. 寄存器汇总——PMD

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0EDB	保留									
0x0EDC	PMD0	7:0	SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD
0x0EDD	PMD1	7:0		TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD
0x0EDE	PMD2	7:0		DACMD	ADCMD			CMP2MD	CMP1MD	ZCDMD
0x0EDF	PMD3	7:0	CLC8MD	CLC7MD	CLC6MD	CLC5MD	PWM4MD	PWM3MD	CCP2MD	CCP1MD
0x0EE0	PMD4	7:0	UART2MD	UART1MD	MSSP2MD	MSSP1MD				CWG1MD
0x0EE1	PMD5	7:0	CLC4MD	CLC3MD	CLC2MD	CLC1MD				DSMMD

11. 复位

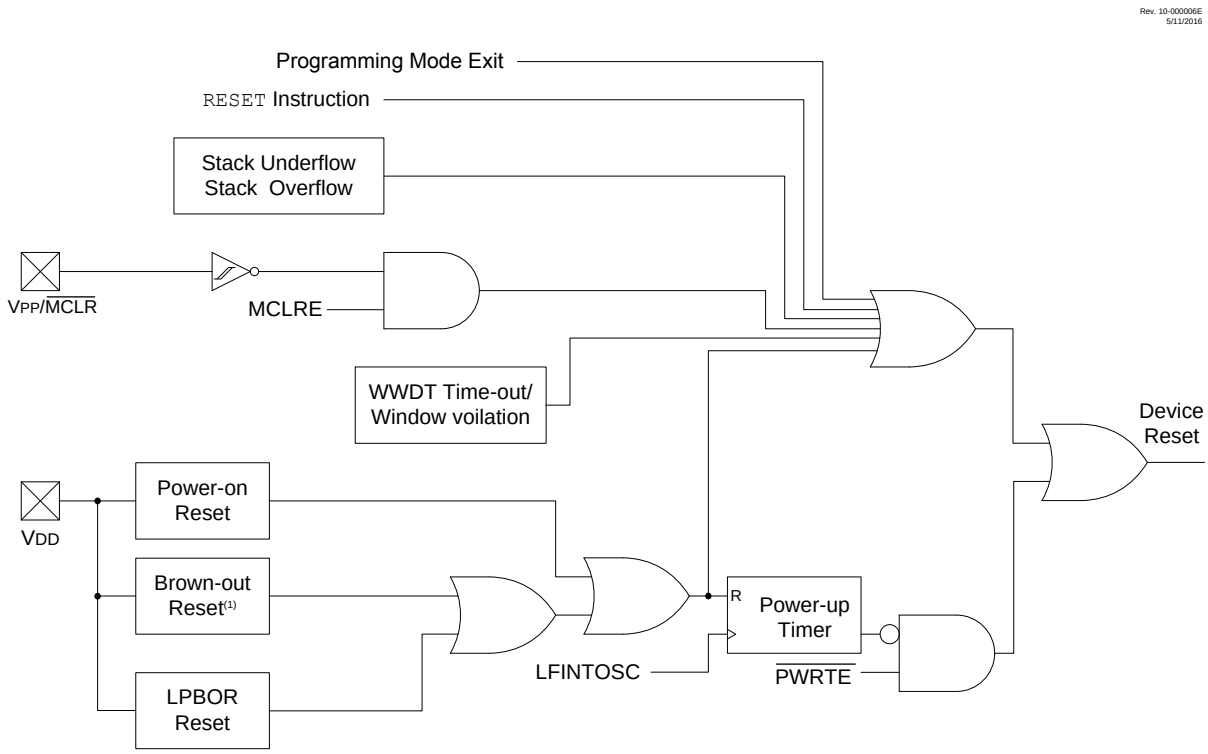
此系列器件有多种复位方式：

- 上电复位（POR）
- 欠压复位（BOR）
- 低功耗欠压复位（LPBOR）
- $\overline{\text{MCLR}}$ 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。

下面的框图给出了片上复位电路的简化框图。

图 11-1. 片上复位电路的简化框图



注：有关 BOR 有效条件，请参见**“BOR 工作模式”**表。

11.1. 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 $\overline{\text{MCLR}}$ 功能延长启动周期。

11.2. 欠压复位（BOR）

当 V_{DD} 达到可选的最低电压时，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。

欠压复位模块具有 4 种工作模式，它们由 CONFIG2 中的 BOREN[1:0]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 BOR 工作模式。

对 CONFIG2 中的 BORV[1:0]位进行配置来选择欠压复位电压。

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}，器件将复位，并且 PCON0 寄存器中的 BOR 位将清零以表示发生掉电复位。有关详细信息，请参见**“BOR 由软件控制”**一节中的**“欠压情形”**图。

11.2.1. BOR 始终使能

配置字 2 的 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迟。

11.2.2. BOR 在休眠时禁止

配置字 2 的 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。器件唤醒延迟直到 BOR 就绪。

11.2.3. 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} 电平延迟。

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。

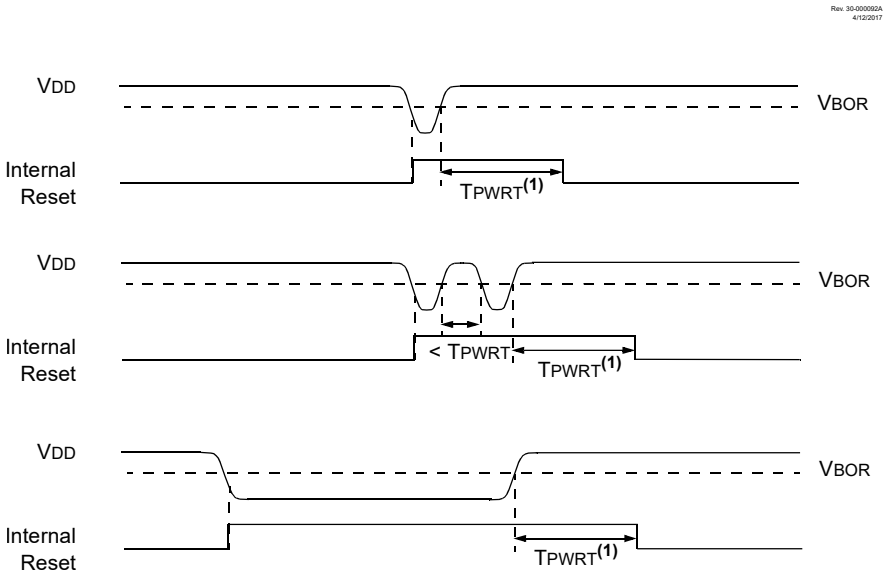
表 11-1. BOR 工作模式

BOREN[1:0]	SBOREN	器件模式	BOR 模式	在以下情况下指令的执行：	
				POR 释放	从休眠模式唤醒
11	X	X	工作	等待 BOR 释放（BORRDY = 1）	立即开始
10	X	唤醒	工作	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	工作	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[1:0]位被强制开启。

图 11-2. 欠压情形



注：仅在 $\overline{\text{PWRTE}}$ 位被编程为 0 时，才应用 T_{PWRT} 延时。

11.2.4. BOR 和批量擦除

通过在 PFM 批量擦除操作期间强制开启 BOR，可确保系统代码保护不会因为 V_{DD} 降低而受到影响。

在批量擦除期间，BOR 会在电压达到 1.9V（无论配置为任何值）时被使能。如果 V_{DD} 下降，擦除周期将会中止，但器件不会复位。

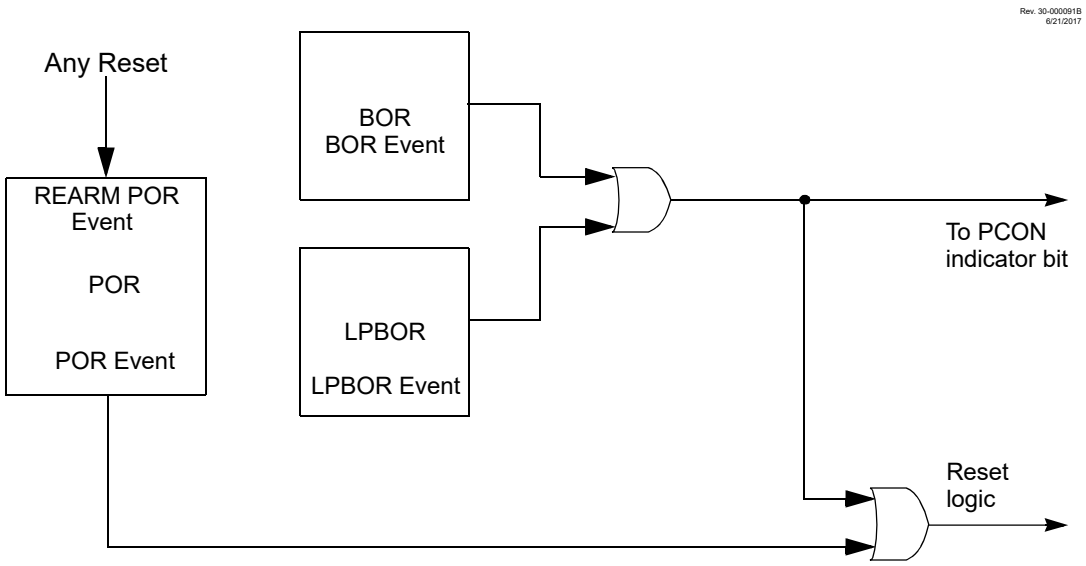
11.3. 低功耗欠压复位（LPBOR）

低功耗欠压复位（LPBOR）电路提供了一种备选保护方案来应对欠压状态。当 V_{DD} 降至 LPBOR 阈值以下时，器件将保持复位状态。此时，PCON0 寄存器的 BOR 位将清零以指示发生了欠压复位。当 BOR 或 LPBOR 电路检测到 BOR 条件时，BOR 位将清零。

无论是否使能 BOR，都可以使用 LPBOR 功能。当在 BOR 使能的情况下使用时，如果 BOR 电路未能检测到 BOR 条件，LPBOR 可作为辅助保护电路使用。此外，除非在休眠模式下，否则当 BOR 使能（BOREN[1:0] = 10）时，如果 V_{DD} 低于 LPBOR 阈值，LPBOR 电路将使器件保持复位状态，同时还将重新激活 POR（关于 LPBOR 复位电压级别，请参见复位、WDT、振荡器起振定时器、上电定时器、欠压复位和低功耗欠压复位规范）。

当在未使能 BOR 的情况下使用时，LPBOR 电路提供单一复位跳变点，并且有利于降低电流消耗。

图 11-3. LPBOR、BOR 和 POR 的关系



11.3.1. 使能 LPBOR

LPBOR 由配置字 2 的 $\overline{\text{LPBOREN}}$ 位控制。在器件被擦除后，LPBOR 模块默认设为禁止。

11.3.1.1. LPBOR 模块输出

LPBOR 模块的输出是一个用于指示是否要将复位置为有效的信号。该信号与 BOR 模块的复位信号进行逻辑或运算，用以提供通用 $\overline{\text{BOR}}$ 信号，并送至 PCON0 寄存器和电源控制模块。

11.4. $\overline{\text{MCLR}}$ 复位

$\overline{\text{MCLR}}$ 是可复位器件的可选外部输入。 $\overline{\text{MCLR}}$ 功能由配置字 2 的 MCLRE 位和配置字 4 的 LVP 位控制（见下表）。如果发生 $\overline{\text{MCLR}}$，则 PCON0 寄存器中的 $\overline{\text{RMCLR}}$ 位将设置为 0。

表 11-2. $\overline{\text{MCLR}}$ 配置

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

11.4.1. $\overline{\text{MCLR}}$ 使能

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

器件在 $\overline{\text{MCLR}}$ 复位路径中有一个噪声滤波器。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、WWDT、POR、STKOVF、STKUNF）不会将 $\overline{\text{MCLR}}$ 引脚驱动为低电平。

11.4.2. $\overline{\text{MCLR}}$ 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。

11.5. 窗口看门狗定时器（WWDT）复位

如果固件在超时周期内未发出 CLRWDT 指令，窗口看门狗定时器将产生复位。STATUS 寄存器中的 $\overline{TO}$ 和 $\overline{PD}$ 位以及 $\overline{RWDT}$ 位会更改，表示发生 WDT 复位。 $\overline{WDTWV}$ 位用于指示是否由于超时或窗口超限导致 WDT 复位。

11.6. RESET 指令

RESET 指令将导致器件复位。 $\overline{RI}$ 位将设置为 0。有关 RESET 指令发生后的默认状态，请参见**“特殊寄存器的复位条件”**。

11.7. 堆栈上溢/下溢复位

器件可以在堆栈上溢或下溢时复位。PCON0 寄存器中的 STKOVF 或 STKUNF 位指示复位条件。通过将配置字 2 中的 STVREN 位置 1 可以使能这些复位。

11.8. 编程模式退出

退出编程模式时，器件执行的操作与发生 POR 时一样。

11.9. 上电延时定时器（PWRT）

上电延时定时器在 POR 或欠压复位时提供 66 ms（2048 个 LFINTOSC 时钟周期）标称值的延时。

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} 上升到所需的电压提供额外的时间。可通过清零配置字中的 $\overline{PWRTE}$ 位使能上电延时定时器。

上电延时定时器在 POR 和 BOR 释放后启动。

11.10. 启动序列

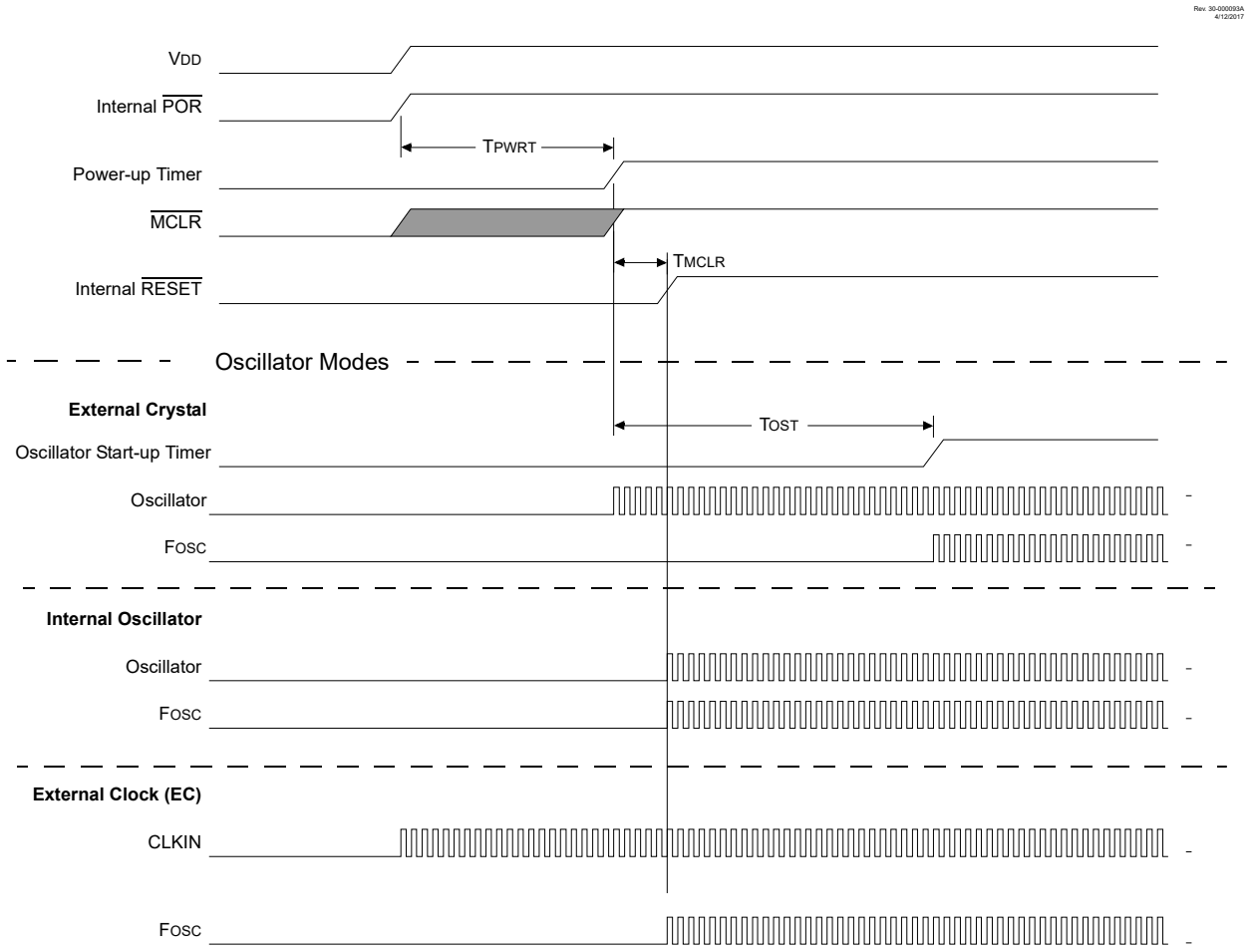
在 POR 或 BOR 释放时，只有先发生以下事件，器件才会开始执行：

1. 上电延时定时器运行完毕（如果使能）。
2. 振荡器起振定时器运行完毕（如果对于选定振荡器源需要）。
3. $\overline{MCLR}$ 必须被释放（如果使能）。

总超时时间将根据振荡器配置和上电延时定时器配置而变化。

上电延时定时器和振荡器起振定时器独立于 $\overline{MCLR}$ 复位运行。如果 $\overline{MCLR}$ 保持足够长时间的低电平，上电延时定时器和振荡器起振定时器将超时。将 $\overline{MCLR}$ 电平拉高后，器件将在 10 个 F_{OSC} 周期后开始执行（见下图）。这对于测试或同步多个并行工作的器件来说是非常有用的。

图 11-4. 复位启动序列



11.11. 确定复位原因

在发生任何复位时，STATUS 和 PCON0 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 11-3. 特殊寄存器的复位条件

条件	程序计数器	STATUS 寄存器(2,3)	PCON0 寄存器	PCON1 寄存器
上电复位	0	-110 0000	0011 110x	---- -1-1
欠压复位	0	-110 0000	0011 11u0	---- -u-u
正常工作期间的 $\overline{MCLR}$ 复位	0	-uuu uuuu	uuuu 0uuu	uuuu-u-u
休眠期间的 $\overline{MCLR}$ 复位	0	-10u uuuu	uuuu 0uuu	uuuu-u-u
WDT 超时复位	0	-0uu uuuu	uuu0 uuuu	uuuu-u-u
WDT 从休眠模式唤醒	PC + 2	-00u uuuu	uuuu uuuu	uuuu-u-u
WWDT 窗口超限复位	0	-uuu uuuu	uu0u uuuu	uuuu-u-u
通过中断从休眠模式唤醒	PC + 2(1)	-10u 0uuu	uuuu uuuu	uuuu-u-u
执行了 RESET 指令	0	-uuu uuuu	uuuu u0uu	uuuu-u-u
堆栈上溢复位（STVREN = 1）	0	-uuu uuuu	1uuu uuuu	uuuu-u-u

表 11-3. 特殊寄存器的复位条件 （续）

条件	程序计数器	STATUS 寄存器(2,3)	PCON0 寄存器	PCON1 寄存器
堆栈下溢复位（STVREN = 1）	0	-uuu uuuu	u1uu uuuu	uuuu-u-u
数据保护（熔丝故障）	0	---u uuuu	uuuu uuuu	---- -u-0
VREG 或 ULP 就绪故障	0	---1 1000	0011 001u	---- -0-1

图注： u = 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位（GIE）置 1，则执行 PC + 2 后，返回地址被压入堆栈且 PC 装入相应的中断向量（取决于中断源的优先级高低）。
2. 如果状态位未实现，该位将读为 0。
3. 状态位 Z、C 和 DC 由 POR/BOR 复位。

11.12. 电源控制（PCON0）寄存器

电源控制（PCON0）寄存器包含区分以下各种复位的标志位：

- 欠压复位（$\overline{\text{BOR}}$）
- 上电复位（$\overline{\text{POR}}$）
- Reset 指令复位（$\overline{\text{RI}}$）
- $\overline{\text{MCLR}}$ 复位（$\overline{\text{RMCLR}}$）
- 看门狗定时器复位（$\overline{\text{RWDT}}$）
- 看门狗窗口超限（$\overline{\text{WDTWV}}$）
- 堆栈下溢复位（STKUNF）
- 堆栈上溢复位（STKOVF）

电源控制寄存器位如 PCON0 所示。

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（表 11-3）。

在重启后，软件应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 PCON0 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。

11.13. 寄存器定义：功耗控制

11.13.1. BORCON

名称： BORCON
偏移量： 0xEDB

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 软件欠压复位使能位

复位状态： POR/BOR = 1
所有其他复位 = u

值	条件	说明
—	如果 BOREN ≠01	SBOREN 可读/写，但对 BOR 没有任何作用
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位

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

11.13.2. PCON0

名称： PCON0
偏移量： 0xFD7

电源控制寄存器 0

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

Bit 7 – STKOVF 堆栈上溢标志位

复位状态： POR/BOR = 0
所有其他复位时的值 = 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 堆栈下溢标志位

复位状态： POR/BOR = 0
所有其他复位时的值 = 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 5 – $\overline{\text{WDTWV}}$ 看门狗窗口超限标志位

复位状态： POR/BOR = 1
所有其他复位时的值 = q

值	说明
1	未发生 WDT 窗口超限或由固件置 1
0	WDT 复位窗口关闭时会发出 CLRWDT 指令（发生 WDT 窗口超限复位时会由硬件设置为 0）

Bit 4 – $\overline{\text{RWDT}}$ WDT 复位标志位

复位状态： POR/BOR = 1
所有其他复位时的值 = 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 – $\overline{\text{RMCLR}}$ $\overline{\text{MCLR}}$ 复位标志位

复位状态： POR/BOR = 1
所有其他复位时的值 = q

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

Bit 2 – $\overline{\text{RI}}$ RESET 指令标志位

复位状态： POR/BOR = 1

所有其他复位时的值 = 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 – $\overline{\text{POR}}$ 上电复位状态位

复位状态： POR/BOR = 0
所有其他复位时的值 = 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 – $\overline{\text{BOR}}$ 欠压复位状态位

复位状态： POR/BOR = q
所有其他复位时的值 = 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）

11.13.3. PCON1

名称： PCON1
偏移量： 0xFD6

电源控制寄存器 1

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

Bit 2 – $\overline{\text{RVREG}}$ 主 LDO 稳压器复位标志位

复位状态： POR/BOR = 1
所有其他复位 = q

值	说明
1	未发生 LDO 或 ULP “就绪”复位；或由固件置 1
0	发生了 LDO 或 ULP “就绪”复位（VDDCORE 达到其最低规范值）

Bit 0 – $\overline{\text{RCM}}$ 配置存储器复位标志位

复位状态： POR/BOR = 1
所有其他复位 = q

值	说明
1	配置和校准锁存器未损坏
0	由于配置和/或校准数据锁存器损坏而发生了复位

11.14. 寄存器汇总——BOR 控制和电源控制

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0EDA	保留									
0x0EDB	BORCON	7:0	SBOREN							BORRDY
0x0EDC ... 0x0FD5	保留									
0x0FD6	PCON1	7:0						RVREG		RCM
0x0FD7	PCON0	7:0	STKOVF	STKUNF	WDTWV	RWDT	RMCLR	RI	POR	BOR

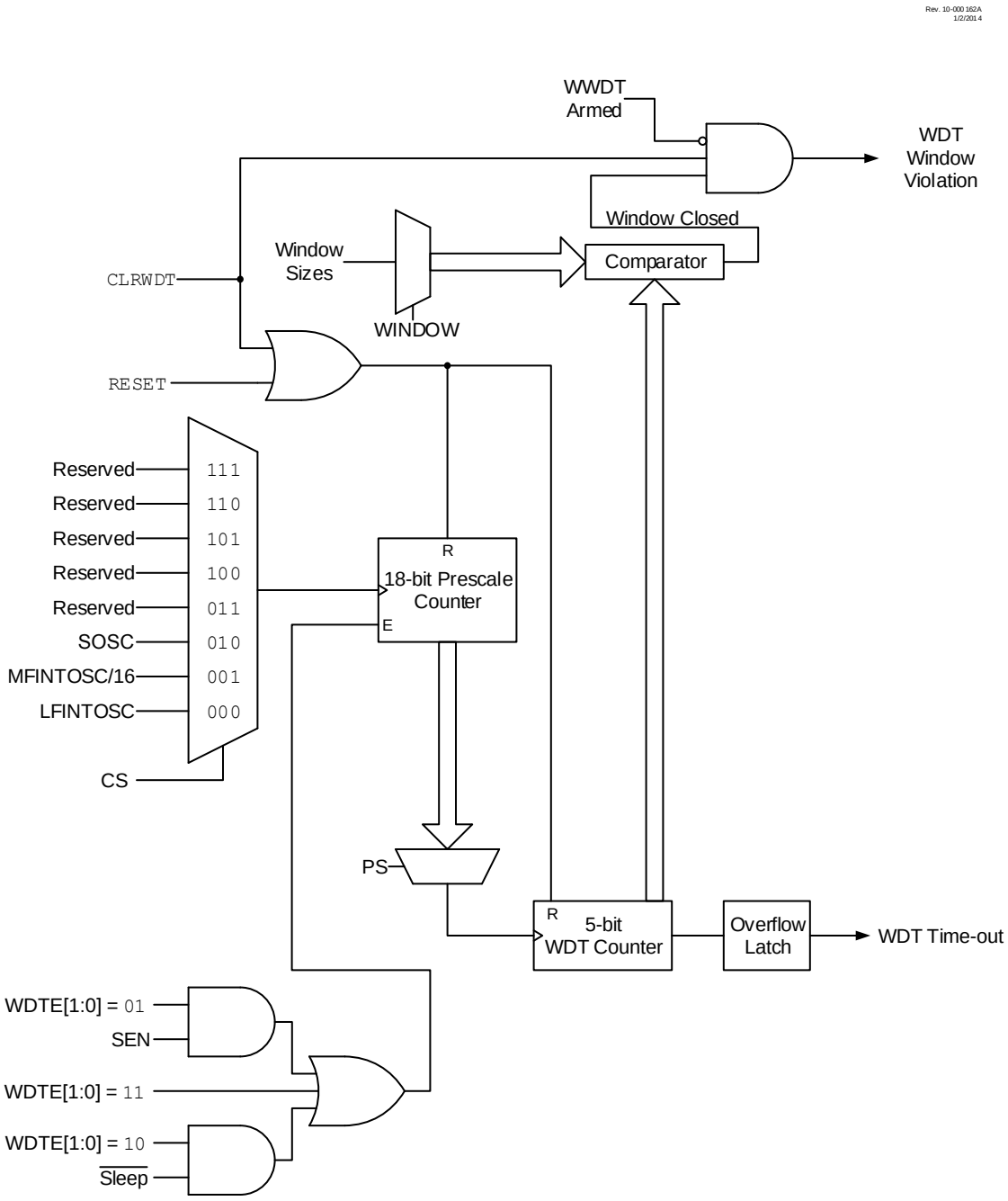
12. WWDT——窗口看门狗定时器

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。看门狗定时器通常用于使系统从意外事件中恢复。窗口看门狗定时器（WWDT）的不同之处在于，CLRWDT 指令只有在超时周期的特定窗口内执行，才会被接受。

WWDT 具有以下特性：

- 可选的时钟源
- 多种工作模式
  - WWDT 总是开启
  - WWDT 在休眠模式下关闭
  - WWDT 通过软件进行控制
  - WWDT 总是关闭
- 可配置的超时周期为 1 ms 至 256s（标称值）
- 可配置的窗口大小为超时周期的 12.5%至 100%
- 多种复位条件

图 12-1. 窗口看门狗定时器框图



12.1. 独立时钟源

WWDT 从 31 kHz LFINTOSC 或 31.25 kHz MFINTOSC 的内部振荡器获得其时基，具体取决于 CONFIG3 中的 WDTE 位的值。

如果 WDTE = 'b1x，则将根据 CONFIG3 中的 WDTCCS 位使能时钟源。

如果 WDTE = 'b01，则必须使用软件将 WDTCON0 寄存器中的 SEN 位置 1 以使能 WWDT，并通过 WDTCON1 寄存器中的 WDTCS 位使能时钟源。

本章中的时间间隔均基于 1 ms 的最小标称时间间隔。关于 LFINTOSC 和 MFINTOSC 容差，请参见内部振荡器参数。

12.2. WWDT 工作模式

窗口看门狗定时器模块具有 4 种工作模式，这些工作模式由 CONFIG3 中的 WDTE 位控制。请参见表 12-1。

表 12-1. WWDT 工作模式

WDTE[1:0]	SEN	器件模式	WWDT 模式
11	X	x	工作
10	X	唤醒	工作
		休眠	禁止
01	1	x	工作
	0	x	禁止
00	X	x	禁止

12.3. 超时周期

如果 CONFIG3 中的 WDTCPS 位设置为 0'b11111，则 WDTPS 位用于设置 1 ms 到 256s（标称值）的超时周期。如果将除默认值以外的任何值赋给 WDTCPS 配置位，则定时器周期将基于 CONFIG3 寄存器中的 WDTCPS 位。复位后，默认的超时周期为 2s。

12.4. 看门狗窗口

窗口看门狗定时器具有一种可选的窗口模式，该模式通过配置字 3 中的 WDTCWS 配置位和 WDTCON1 寄存器中的 WINDOW 位控制。在窗口模式下，CLRWDT 指令必须出现在 WDT 周期的允许窗口中。在此窗口外出现的任何 CLRWDT 指令都将触发窗口超限并导致 WWDT 复位，类似于 WWDT 超时。有关示例，请参见图 12-2。

如果配置字 3 中的 WDTCWS 位设置为 111，则窗口大小由 WDTCON1 中的看门狗定时器窗口选择（WINDOW）位控制。

WDTTMR 寄存器中的 WDTTMR 位用于确定窗口是否打开，具体由 WINDOW 位定义。

如果发生窗口超限，将产生复位，并且 PCON0 寄存器的 $\overline{\text{WDTWV}}$ 位将清零。该位由 POR 置 1，也可由固件置 1。

12.5. 将 WWDT 清零

当发生以下任何条件时，WWDT 被清零：

- 任何复位
- 执行了有效的 CLRWDT 指令
- 器件进入休眠模式
- 因中断而退出休眠模式
- WWDT 被禁止
- 振荡器起振定时器（OST）正在运行
- 对 WDTCON0 或 WDTCON1 寄存器的任何写操作

12.5.1. CLRWDT 注意事项（窗口模式）

处于窗口模式时，必须在 CLRWDT 指令将定时器清零之前配置 WWDT。通过读取 WDTCON0 寄存器来执行该操作。执行 CLRWDT 指令而不执行此类配置操作将触发窗口超限，而与窗口是否打开无关。

有关更多信息，请参见表 12-2。

12.6. 休眠期间的操作

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

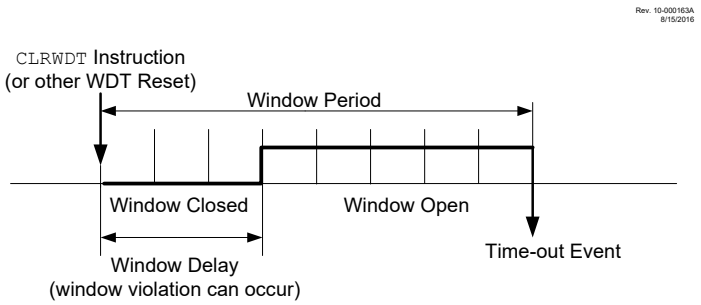
WWDT 保持清零，直到振荡器起振定时器（OST）延时结束为止（如果使能）。

在器件处于休眠模式的情况下发生 WWDT 超时时，不会产生复位。器件将会唤醒并继续工作。STATUS 寄存器中的 $\overline{TO}$ 和 $\overline{PD}$ 位会发生改变，指示发生的事件。也可以使用 PCON0 寄存器中的 $\overline{RWDT}$ 位。

表 12-2. WWDT 清零条件

条件	WWDT
WDTE = 00	清零
WDTE = 01 且 SEN = 0	
WDTE = 10 且进入休眠模式	
CLRWDT 命令	
检测到振荡器故障	
退出休眠 + 系统时钟 = SOSC、EXTRC、INTOSC 或 EXTCLK	
退出休眠 + 系统时钟 = XT、HS 或 LP	清零，直到 OST 延时结束
更改 INTOSC 分频比（IRCF 位）	不受影响

图 12-2. 窗口周期和延时



12.7. 寄存器定义：窗口看门狗定时器控制

12.7.1. WDTCON0

名称： WDTCON0
偏移量： 0xECD

看门狗定时器控制寄存器 0

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

Bit 5:1 – WDTPS[4:0] 看门狗定时器预分频比选择位(1)
位值 = 预分频比

值	说明
11111 至 10011	保留。产生最小的时间间隔（1 ms）
10010	1:8388608（2^{23}）（时间间隔标称值为 256s）
10001	1:4194304（2^{22}）（时间间隔标称值为 128s）
10000	1:2097152（2^{21}）（时间间隔标称值为 64s）
01111	1:1048576（2^{20}）（时间间隔标称值为 32s）
01110	1:524288（2^{19}）（时间间隔标称值为 16s）
01101	1:262144（2^{18}）（时间间隔标称值为 8s）
01100	1:131072（2^{17}）（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 看门狗定时器软件使能/禁止位

值	条件	说明
—	如果 WDTE = 1x	该位被忽略
1	如果 WDTE = 01	WDT 开启
0	如果 WDTE = 01	WDT 关闭
—	如果 WDTE = 00	该位被忽略

注：

1. 时间均为近似值。WDT 时间基于 31 kHz LFINTOSC。
2. 当 CONFIG3 中的 WDTCPS = 11111 时，WDTPS 的复位值（q）为 01011。否则，WDTPS 的复位值等于 CONFIG3 中的 WDTCPS。
3. 当 CONFIG3L 中的 WDTCPS ≠ 11111 时，这些位为只读位。

12.7.2. WDTCON1

名称： WDTCON1
偏移量： 0xECE

看门狗定时器控制寄存器 1

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

Bit 6:4 – WDTCS[2:0] 看门狗定时器时钟选择位

值	说明
111 至 010	保留
001	MFINTOSC 31.25 kHz
000	LFINTOSC 31 kHz

Bit 2:0 – WINDOW[2:0] 看门狗定时器窗口选择位

WINDOW	窗口延时时间百分比	窗口打开时间百分比
111	N/A	100
110	12.5	87.5
101	25	75
100	37.5	62.5
011	50	50
010	62.5	37.5
001	75	25
000	87.5	12.5

注：

1. 如果 CONFIG3 中的 WDTCCS = 111，WDTCS 的复位值为 000。
2. WINDOW 的复位值（q）由 CONFIG3 寄存器中的 WDTCWS 值确定。
3. 如果 CONFIG3 中的 WDTCCS ≠ 111，这些位为只读位。
4. 如果 CONFIG3 中的 WDTCWS ≠ 111，这些位为只读位。

12.7.3. WDTPSL

名称： WDTPSL
偏移量： 0xECF

WWDT 预分频比选择低字节寄存器（只读）

位	7	6	5	4	3	2	1	0
	PSCNTL[7:0]							
访问	RO	RO	RO	RO	RO	RO	RO	RO
复位	0	0	0	0	0	0	0	0

Bit 7:0 – PSCNTL[7:0] 预分频比选择低字节位[1]

注：

1. 18 位 WDT 预分频值 PSCNT[17:0]包括 WDTPSL、WDTPSH 和 WDTTMR 寄存器的低两位。PSCNT[17:0]用于调试操作，可在正常操作期间读取。

12.7.4. WDTPSH

名称： WDTPSH
偏移量： 0xED0

WWDT 预分频比选择高字节寄存器（只读）

位	7	6	5	4	3	2	1	0
	PSCNTH[7:0]							
访问	RO	RO	RO	RO	RO	RO	RO	RO
复位	0	0	0	0	0	0	0	0

Bit 7:0 – PSCNTH[7:0] 预分频比选择高字节位(1)

注：

1. 18 位 WDT 预分频值 PSCNT[17:0]包括 WDTPSL、WDTPSH 和 WDTTMR 寄存器的低两位。PSCNT[17:0]用于调试操作，可在正常操作期间读取。

12.7.5. WDTTMR

名称： WDTTMR
偏移量： 0xED1

WDT 定时器寄存器（只读）

位	7	6	5	4	3	2	1	0
	WDTTMR[4:0]					STATE	PSCNT[1:0]	
访问	RO	RO	RO	RO	RO	RO	RO	RO
复位	0	0	0	0	0	0	0	0

Bit 7:3 – WDTTMR[4:0] 看门狗窗口值位

WINDOW	WDT 窗口状态		打开时间百分比
	关闭	打开	
111	**N/A**	00000-11111	100
110	00000-00011	00100-11111	87.5
101	00000-00111	01000-11111	75
100	00000-01011	01100-11111	62.5
011	00000-01111	10000-11111	50
010	00000-10011	10100-11111	37.5
001	00000-10111	11000-11111	25
000	00000-11011	11100-11111	12.5

Bit 2 – STATE WDT 就绪状态位

值	说明
1	WDT 已就绪
0	WDT 未就绪

Bit 1:0 – PSCNT[1:0] 预分频比选择高位(1)

注：

1. 18 位 WDT 预分频值 PSCNT[17:0]包括 WDTPSL、WDTPSH 和 WDTTMR 寄存器的低两位。PSCNT[17:0]用于调试操作，可在正常操作期间读取。

12.8. 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 ... 0x0ECC	保留									
0x0ECD	WDTCON0	7:0			WDTPS[4:0]					SEN
0x0ECE	WDTCON1	7:0		WDTCS[2:0]				WINDOW[2:0]		
0x0ECF	WDTPSL	7:0	PSCNTL[7:0]							
0x0ED0	WDTPSH	7:0	PSCNTH[7:0]							
0x0ED1	WDTTMR	7:0	WDTTMR[4:0]					STATE	PSCNT[1:0]	

13. 存储器构成

增强型单片机有三种类型的存储器：

- 程序存储器
- 数据 RAM
- 数据 EEPROM

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。在实际应用中，可将数据 EEPROM 视为外设，因为它通过一组控制寄存器进行寻址和访问。

有关闪存程序存储器和数据 EEPROM 存储器操作的更多详细信息，请参见非易失性存储器（NVM）控制部分。

13.1. 程序存储器构成

单片机实现了一个 21 位程序计数器，能够寻址 2 MB 的程序存储空间。访问物理实现的存储器上边界与 2 MB 地址之间的存储单元时将返回全 0（一条 NOP 指令）。

有关器件存储器映射和代码保护配置位与 PFM 各部分之间的关系，请参见下表。

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

表 13-1. 程序与数据存储器映射

<table>
<tr><th rowspan="2">地址</th><th colspan="3">器件</th></tr>
<tr><th>CN2510</th><th>CN2610</th><th>CN2710</th></tr>
<tr><td>00 0000h
至
00 7FFFh</td><td>闪存程序存储器（16 KW）(1)</td><td rowspan="2">闪存程序存储器（32 KW）(1)</td><td rowspan="3">闪存程序存储器（64 KW）(1)</td></tr>
<tr><td>00 4000h
至
00 FFFFh</td><td rowspan="3">不存在(2)</td></tr>
<tr><td>00 8000h
至
01 FFFFh</td><td rowspan="2">不存在(2)</td></tr>
<tr><td>02 0000h
至
1F FFFFh</td><td>不存在(2)</td></tr>
<tr><td>20 0000h
至
20 00FFh</td><td colspan="3">用户 ID（128 字）(3)</td></tr>
<tr><td>20 0100h
至
2F FFFFh</td><td colspan="3">保留</td></tr>
<tr><td>30 0000h
至
30 000Bh</td><td colspan="3">配置字节(3)</td></tr>
<tr><td>30 000Ch
至
30 02FFh</td><td colspan="3">保留</td></tr>
</table>

表 13-1. 程序与数据存储器映射 （续）

地址	器件		
	CN2510	CN2610	CN2710
30 0300h 至 30 FFFFh	**未实现**		
31 0000h 至 31 00FFh	数据 EEPROM（256 字节）	数据 EEPROM（1 KB）	
31 0100h 至 31 3FFFh	**未实现**		
31 0400h 至 3F FFFBh		**未实现**	
3F FFFCh 至 3F FFFDh	版本 ID（1 字）**(3)(4)(5)**		
3F FFFEh 至 3F FFFFh	器件 ID（1 字）**(3)(4)(5)**		

注：

1. 复位和中断向量包含在 PFM 存储空间中。
2. 地址不会计满返回。此区域读为 0。使用 NVMCON 寄存器访问这些区域时，读操作和/或写操作会将 NVMERR 位置 1。
3. 不受代码保护。
4. 在芯片中硬编码。
5. 该区域无法由用户写入，并且不受批量擦除的影响。

表 13-2. 存储器映射和代码保护控制

区域	地址	器件		
		CN2510	CN2610	CN2710
PFM	00 0000h 至 00 07FFh	引导块 1 KW CP、WRTB 和 EBTRB		
	00 0800h 至 00 1FFFh	块 0 3 KW CP、WRT0 和 EBTR0	块 0 7 KW CP、WRT0 和 EBTR0	
	00 2000h 至 00 3FFFh	块 1 4 KW CP、WRT1 和 EBTR1		
	00 4000h 至 00 5FFFh	块 2 4 KW CP、WRT2 和 EBTR2	块 1 8 KW CP、WRT1 和 EBTR1	
	00 6000h 至 00 7FFFh	块 3 4 KW CP、WRT3 和 EBTR3		
	00 8000h 至 00 BFFFh	不存在	块 2 8 KW CP、WRT2 和 EBTR2	
	00 C000h 至 00 FFFFh		块 3 8 KW CP、WRT3 和 EBTR3	
	01 0000h 至 01 3FFFh		不存在	块 4 8 KW CP、WRT4 和 EBTR4
	01 4000h 至 01 7FFFh			块 5 8 KW CP、WRT5 和 EBTR5
	01 8000h 至 01 BFFFh			块 6 8 KW CP、WRT6 和 EBTR6
	01 C000h 至 01 FFFFh			块 7 8 KW CP、WRT7 和 EBTR7
配置	30 0000h 至 30 000Bh	配置字 WRTC		
数据 EEPROM	31 0000h 至 30 00FFh	256 字节 CPD 和 WRTD	1024 字节 CPD 和 WRTD	
	31 0100h 至 31 03FFh	未实现		

13.1.1. 程序计数器

程序计数器（Program Counter，PC）指定即将获取并执行的指令的地址。PC 为 21 位宽，包含在三个独立的 8 位寄存器中。其低字节（称为 PCL 寄存器）可读写。高字节（称为 PCH 寄存器）包含 PC[15:8]位，不可直接读写。PCH 寄存器通过 PCLATH 寄存器来执行更新。最高字节称为 PCU。该寄存器包含 PC[20:16]位，同样不可直接读写。PCU 寄存器通过 PCLATU 寄存器来执行更新。

PCLATH 和 PCLATU 的内容通过写入 PCL 的任何操作传送到程序计数器。类似地，程序计数器的两个高字节通过读取 PCL 的操作传送到 PCLATH 和 PCLATU。这对于 PC 的偏移计算很有用（见计算 GOTO）。

PC 寻址程序存储器中的字节。为防止 PC 与字指令不对齐，PCL 的最低有效位固定为值 0。PC 递增 2 可寻址程序存储器中的连续指令。

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

13.1.2. 返回地址堆栈

返回地址堆栈最多允许 31 个程序调用和中断的任意组合。当执行 CALL 或 RCALL 指令或应答中断后，PC 值将被压入堆栈。执行 RETURN、RETLW 或 RETFIE 时，会将 PC 值从堆栈中弹出。PCLATU 和 PCLATH 不受任何 RETURN 或 CALL 指令的影响。

堆栈操作通常为由 21 位 RAM 和 5 位堆栈指针实现的 31 字操作，在 ICD 模式下则为由 21 位 RAM 和 6 位堆栈指针实现的 35 字操作。堆栈既不占用程序存储空间，也不占用数据存储空间。堆栈指针可读写，堆栈顶部的地址可通过栈顶（Top-of-Stack，TOS）特殊文件寄存器进行读写。此外，也可以使用这些寄存器将数据压入堆栈或从堆栈中弹出。

CALL 类型的指令会触发压栈操作：首先递增堆栈指针，然后将 PC（已指向 CALL 后面的指令）的内容写入堆栈指针指向的存储单元。RETURN 类型的指令触发出栈操作：首先将 STKPTR 指向的存储单元中的内容传送到 PC，然后递减堆栈指针。

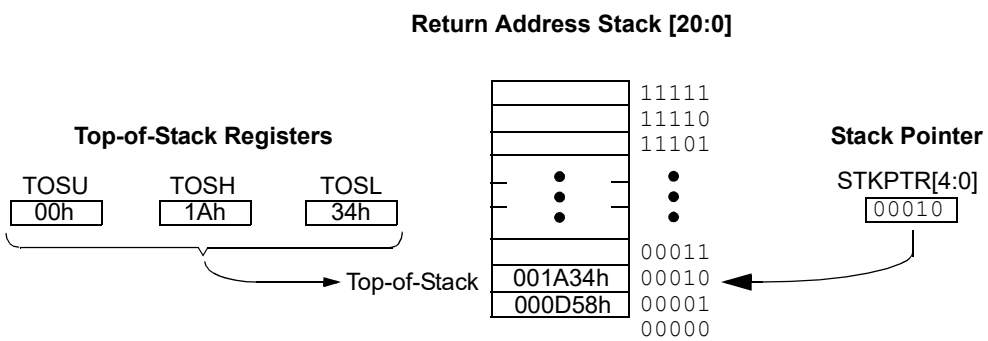
发生任何复位后，堆栈指针均会初始化为“0b00000”。不存在与堆栈指针值“0b00000”对应的存储单元关联的 RAM；这只是一个复位值。PCON0 寄存器中的状态位指示堆栈处于已满、上溢还是下溢状态。

13.1.2.1. 栈顶访问

只有返回地址栈顶（TOS）可读写。通过一组三个寄存器（TOSU:TOSH:TOSL）保存 STKPTR 寄存器指向的堆栈单元的内容（见图 13-1）。这样，用户可在必要时实现软件堆栈。在 CALL、RCALL 或中断后，软件可通过读取 TOSU:TOSH:TOSL 寄存器来读取压入堆栈的值。这些值可置于用户定义的软件堆栈中。在返回时，软件可将这些值返回到 TOSU:TOSH:TOSL，然后再执行返回操作。

访问堆栈时，用户必须禁止全局中断允许（GIE）位，以防止对堆栈造成意外损坏。

图 13-1. 返回地址堆栈和相关寄存器



13.1.2.2. 返回堆栈指针

STKPTR 寄存器包含堆栈指针值。可以使用 PCON0 寄存器访问 STKOVF（堆栈上溢）状态位和 STKUNF（堆栈下溢）状态位。堆栈指针的值介于 0 和 31 之间。在复位时，堆栈指针的值将为零。用户可以读取和写入堆栈指针值。实时操作系统（Real-Time Operating System，RTOS）可以使用此特性来保持堆栈。将 PC 压入堆栈 32 次（在此期间未从堆栈中弹出任何值）后，STKOVF 位将置 1。STKOVF 位通过软件或 POR 清零。堆栈已满时触发的操作取决于 STVREN（堆栈上溢复位使能）配置位的状态。

如果 STVREN 置 1（默认值），将产生复位，并且 STKOVF 位将在启动第 32 次压栈操作时指示堆栈上溢。这包括 CALL 和 CALLW 指令，以及在中断响应期间对返回地址执行堆栈操作。STKOVF 位将保持置 1，堆栈指针将设置为 0。

如果 STVREN 清零，则在启动第 32 次压栈操作时，STKOVF 位将置 1，并且堆栈指针将保持为 31 而不会发生复位。任何额外的压栈操作都会覆盖第 31 次压入堆栈的内容，但 STKPTR 将始终保持为 31。

在软件中设置 STKOVF = 1 将改变该位，但不会产生复位。

当出栈操作返回值 0 时，STKUNF 位将置 1。STKUNF 位通过软件或 POR 清零。堆栈已满时触发的操作取决于 STVREN（堆栈上溢复位使能）配置位的状态。

如果 STVREN 置 1（默认值），并且出栈操作次数已达到需要为堆栈卸载的限值，则下一次出栈操作会向 PC 返回值 0、将 STKUNF 位置 1 并产生复位。此条件可通过 RETURN、RETLW 和 RETFIE 指令生成。

如果 STVREN 清零，则 STKUNF 位将置 1，但不会发生复位。

重要：在发生下溢时向 PC 返回值 0 会将程序引导到复位向量，之后可在其中采取适当的操作来验证堆栈条件。这与复位有所不同，因为 SFR 的内容不受影响。

13.1.2.3. 堆栈上溢和下溢复位

通过将配置字中的 STVREN 配置位置 1，可以使能器件在发生堆栈上溢和堆栈下溢时复位。当 STVREN 置 1 时，已满或下溢条件会将相应的 STKOVF 或 STKUNF 位置 1，然后导致器件复位。当 STVREN 清零时，已满或下溢条件会将相应的 STKOVF 或 STKUNF 位置 1，但不会导致器件复位。STKOVF 或 STKUNF 位通过用户软件或上电复位清零。

13.1.2.4. PUSH 和 POP 指令

由于栈顶是可读写的，因此能够在不干扰正常程序执行的情况下将值压入堆栈和将值从堆栈中弹出是一种理想的特性。指令集包含 PUSH 和 POP 两条指令，允许在软件控制下操作 TOS。通过修改 TOSU、TOSH 和 TOSL，可以将数据或返回地址置于堆栈中。

PUSH 指令可将当前 PC 值置于堆栈中。具体过程为先递增堆栈指针，然后将当前 PC 值压入堆栈中。

POP 指令可通过递减堆栈指针来丢弃当前 TOS。先前压入堆栈的值随之成为 TOS 值。

13.1.2.5. 快速寄存器堆栈

STATUS、WREG 和 BSR 寄存器均配有一个快速寄存器堆栈，旨在为中断提供“快速返回”选项。每个寄存器的堆栈均只有 1 级深且不可读写。当处理器处理中断向量指向的中断时，此堆栈中会压入相应寄存器的当前值。所有中断源均会将值压入快速寄存器堆栈。如果随后使用 RETFIE，FAST 指令从中断返回，则这些寄存器中的值会装回相关的寄存器中。

重要：不会在此操作中复制 STATUS 寄存器的 $\overline{TO}$ 和 $\overline{PD}$ 位。

如果同时允许低优先级中断和高优先级中断，则无法可靠地使用堆栈寄存器从低优先级中断返回。如果在处理低优先级中断时发生高优先级中断，则低优先级中断存储的堆栈寄存器值将被覆盖。在上述情况下，用户必须在处理低优先级中断期间通过软件保存密钥寄存器。

如果未使用中断优先级，则所有中断均可使用快速寄存器堆栈从中断返回。如果未使用中断，则可使用快速寄存器堆栈在子程序调用结束时恢复 STATUS、WREG 和 BSR 寄存器。要使用快速寄存器堆栈进行子程序调用，必须执行 CALL *label*, FAST 指令将 STATUS、WREG 和 BSR 寄存器的内容保存到快速寄存器堆栈中。之后，可以执行 RETURN, FAST 指令从快速寄存器堆栈中恢复这些寄存器的内容。

下面给出了在子程序调用和返回期间使用快速寄存器堆栈的源代码示例。

例 13-1. 快速寄存器堆栈代码

```
     CALL SUB1, FAST  ;STATUS, WREG, BSR SAVED IN FAST REGISTER STACK
         •
         •
SUB1:
         •
         •
     RETURN, FAST     ;RESTORE VALUES SAVED IN FAST REGISTER STACK
```

13.1.3. 程序存储器中的查找表

在编程过程中，有时可能需要在程序存储器中创建数据结构或查找表。对于器件，可以通过以下两种方式实现查找表：

- 计算 GOTO
- 表读

13.1.3.1. 计算 GOTO

计算 GOTO 是通过向程序计数器加一个偏移量来实现的。下面给出了代码示例。

可搭配使用 ADDWF PCL 指令和一组 RETLW nn 指令来构成查找表。在对此查找表执行调用之前，会在 W 寄存器中装载一个查找表中的偏移量。被调用程序的第一条指令是 ADDWF PCL 指令。执行的下一条指令将是其中一条 RETLW *nn* 指令，此指令会将值 nn 返回到调用函数。

偏移值（在 WREG 中）指定程序计数器必须前进的字节数，它必须是 2 的倍数（LSb = 0）。

在此方法中，每个指令存储单元只能存储一个数据字节，并且需要在返回地址堆栈中留出空间。

例 13-2. 使用偏移值的计算 GOTO

```
     RLNCF   OFFSET, W   ; W must be an even number, Max OFFSET = 127
     CALL    TABLE

     ORG     nn00h    ; 00 in LSByte ensures no addition overflow
TABLE:
     ADDWF   PCL      ; Add OFFSET to program counter
     RETLW   A        ; Value @ OFFSET=0
     RETLW   B        ; Value @ OFFSET=1
     RETLW   C        ; Value @ OFFSET=2
       .
       .
       .
```

13.1.3.2. 表读和表写

对于在程序存储器中存储数据而言，表读和表写是一种更精简的方法，可在每个指令单元中存储两个字节的数据。

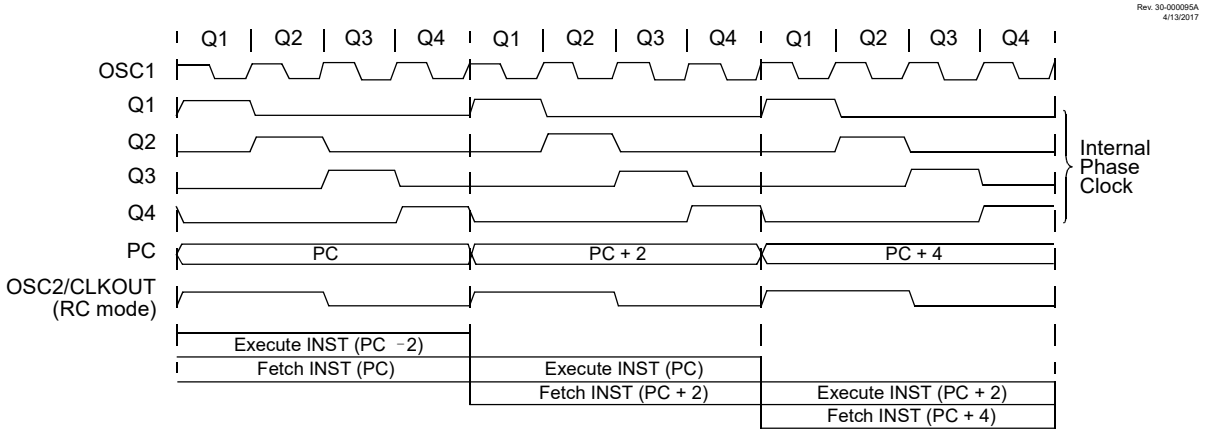
通过使用表读和表写，每个程序字可以存储两个字节的查找表数据。表指针（TBLPTR）寄存器指定字节地址，表锁存器（TABLAT）寄存器包含从程序存储器读取或向其中写入的数据。每次均与程序存储器之间传送一个字节。

13.2. 指令周期

13.2.1. 时钟机制

无论单片机时钟输入来自内部还是外部源，均会在内部进行四分频，以生成四个非重叠的正交时钟（Q1、Q2、Q3 和 Q4）。在内部，程序计数器会在每个 Q1 递增一次；指令从程序存储器中获取并在 Q4 期间锁存到指令寄存器中。指令在随后的 Q1 至 Q4 期间译码并执行。时钟和指令执行流程如下图所示。

图 13-2. 时钟/指令周期



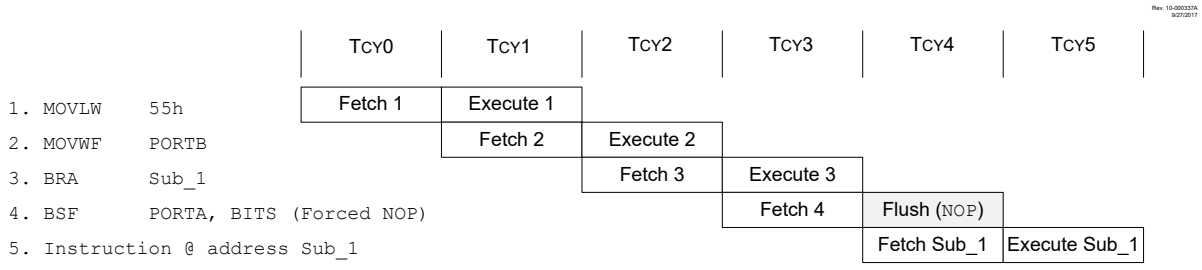
13.2.2. 指令流/流水线

一个“指令周期”由四个 Q 周期组成：Q1 至 Q4。指令的获取与执行采用流水线形式，即获取指令占用一个指令周期，而译码和执行占用另一个指令周期。但是，由于流水线操作的原因，每条指令在一个周期内有效执行。如果某条指令导致程序计数器发生改变（如 GOTO），则需要两个周期才能完成指令，如下图所示。

取指周期从 Q1 中的程序计数器（PC）递增开始。

在执行周期中，获取的指令在周期 Q1 中被锁存到指令寄存器（Instruction Register，IR）中。此指令随后在 Q2、Q3 和 Q4 周期内译码并执行。在 Q2（操作数读操作）期间读取数据存储器，在 Q4（目标写操作）期间写入数据存储器。

图 13-3. 指令流水线流程



All instructions are single cycle except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

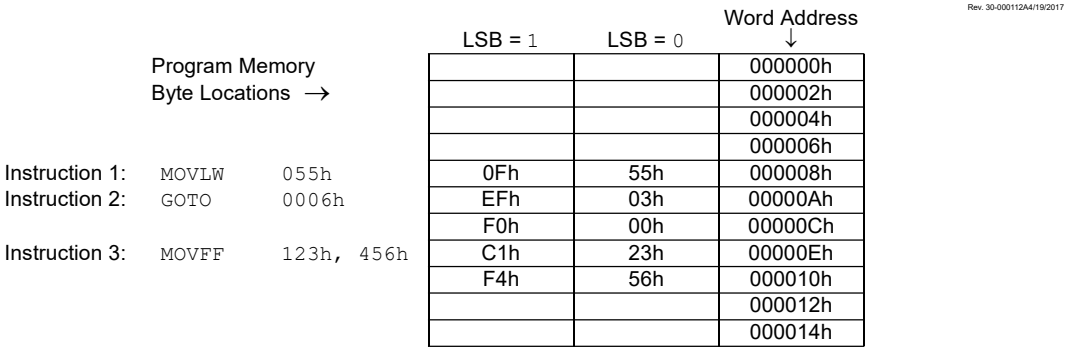
13.2.3. 程序存储器中的指令

程序存储器按字节寻址。指令在程序存储器中以两个字节或四个字节为单位存储。指令字的最低有效字节始终存储在地址为偶数（LSb = 0）的程序存储单元中。为了与指令边界保持对齐，PC 以 2 为增量递增，最低有效位（Least Significant bit，LSb）将始终读为 0（见程序计数器）。

下面的程序存储器图中的指令说明了如何将指令字存储在程序存储器中。

CALL 和 GOTO 指令已将绝对程序存储器地址嵌入到指令中。由于指令始终存储在字边界上，因此指令中包含的数据是字地址。字地址写入 PC[20:1]，以访问程序存储器中所需的字节地址。示例中的指令 2 显示了如何在程序存储器中编码指令 GOTO 0006h。程序分支指令以相同的方式编码相对地址偏移量。存储在分支指令中的偏移值表示 PC 将偏移的单字指令数。指令集汇总提供了指令集的更多详细信息。

图 13-4. 程序存储器中的指令



13.2.4. 双字指令

标准指令集有 4 条双字指令：CALL、MOVFF、GOTO 和 LFSR。在所有情况下，指令的第二个字始终将 1111 作为高 4 位；其他 12 位是立即数数据，通常为数据存储器地址。

在指令的 4 个最高有效位（Most Significant bit，MSb）中使用 1111 指定了一种特殊形式的 NOP。如果指令以适当的顺序执行（紧接在第一个字之后），则指令序列将访问和使用第二个字中的数据。如果由于某种原因跳过第一个字而第二个字由其自身执行，则会转为执行 NOP。当双字指令前面有更改 PC 的条件指令时，这十分必要。下面的双字指令图显示了其工作原理。

重要：有关扩展指令集中双字指令的信息，请参见指令执行和扩展指令集一节。

图 13-5. 双字指令

Rev. 30-000113A
4/19/2017

CASE 1:

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

CASE 2:

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

13.3. 数据存储器构成

重要：使能扩展指令集时，数据存储器某些方面的操作会发生变化。更多信息，请参见指令执行和扩展指令集。

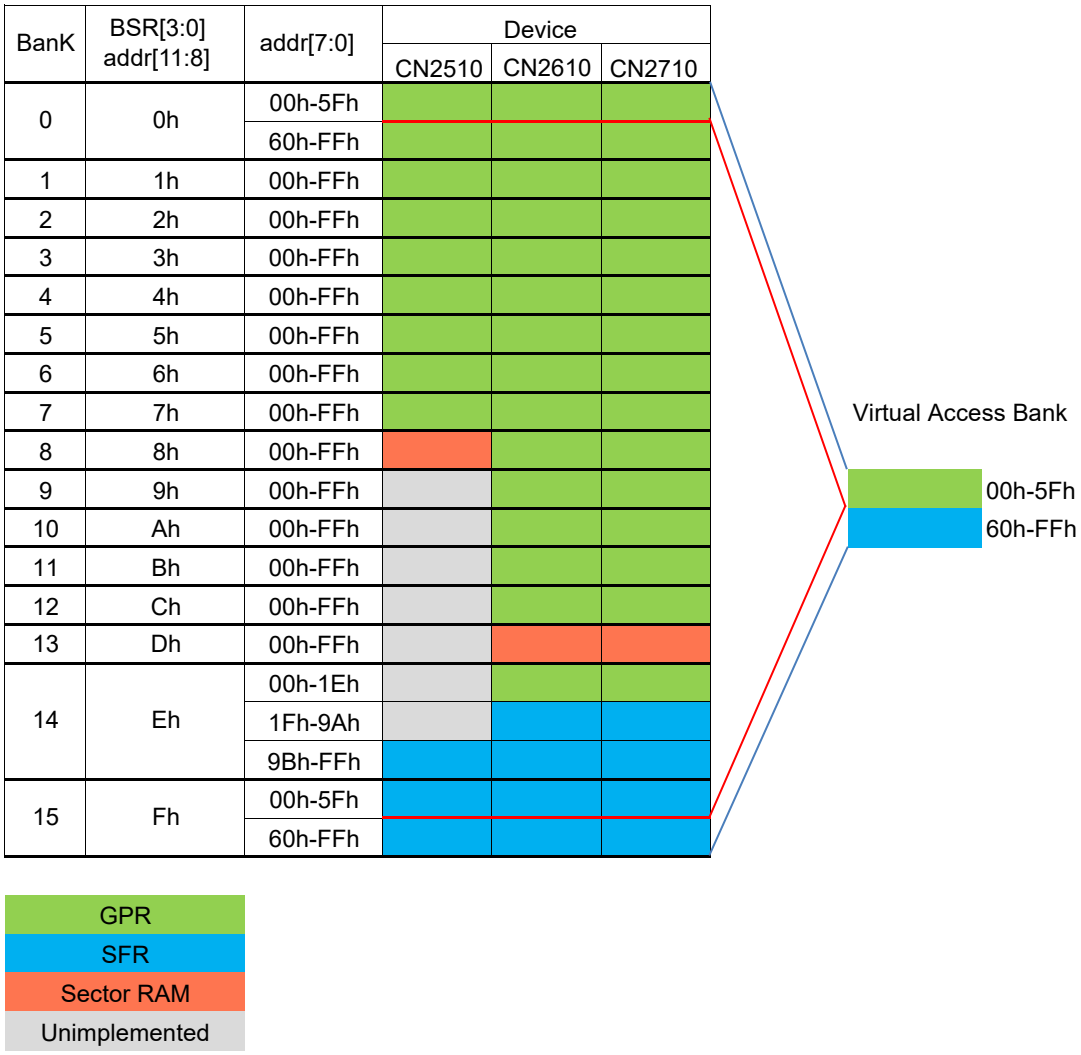
器件中的数据存储器以静态 RAM 的形式实现。数据存储器中的每个寄存器均具有 12 位地址，可提供最多 4096 字节的数据存储空间。存储空间最多分为 16 个存储区，每个存储区包含 256 个字节。下图给出了器件系列中所有器件的数据存储器构成。

数据存储器包含特殊功能寄存器（SFR）和通用寄存器（General Purpose Register，GPR）。SFR 用于控制器和外设功能的控制和状态，而 GPR 用于用户应用程序中的数据存储和暂存操作。对未实现存储单元的任何读操作均将读为 0。

指令集和架构允许跨所有存储区进行操作。可以通过直接、间接或变址寻址模式访问整个数据存储器。本小节稍后将讨论寻址模式。

为确保可在单个周期内访问常用寄存器（SFR 和选定的 GPR），器件实现了快速操作存储区。这是一个 256 字节的存储空间，可在不使用存储区选择寄存器（Bank Select Register，BSR）的情况下快速访问 SFR 以及 GPR 存储区 0 的下半部分。快速操作存储区一节给出了快速操作 RAM 的详细说明。

图 13-6. 数据存储器映射



13.3.1. 存储区选择寄存器

数据存储器的空间较大，因此需要采用一种高效的寻址方案，以便快速访问任何可能的地址。理想情况下，这意味着无需为每次读操作或写操作提供完整地址。为实现此目的，器件采用了 RAM 分区方案。此方案将存储空间划分为 16 个 256 字节的连续存储区。根据指令的不同，每个存储单元可通过其完整的 12 位地址或者搭配 8 位低位地址和 4 位存储区指针来直接寻址。

指令集中的大多数指令均使用存储区指针，即存储区选择寄存器（BSR）。此 SFR 保存存储单元地址的高 4 位；指令本身包括低 8 位。仅实现了 BSR 的低 4 位（BSR[3:0]）。高 4 位未使用，始终读为 0 且无法写入。可以使用 MOVLB 指令直接装入 BSR。

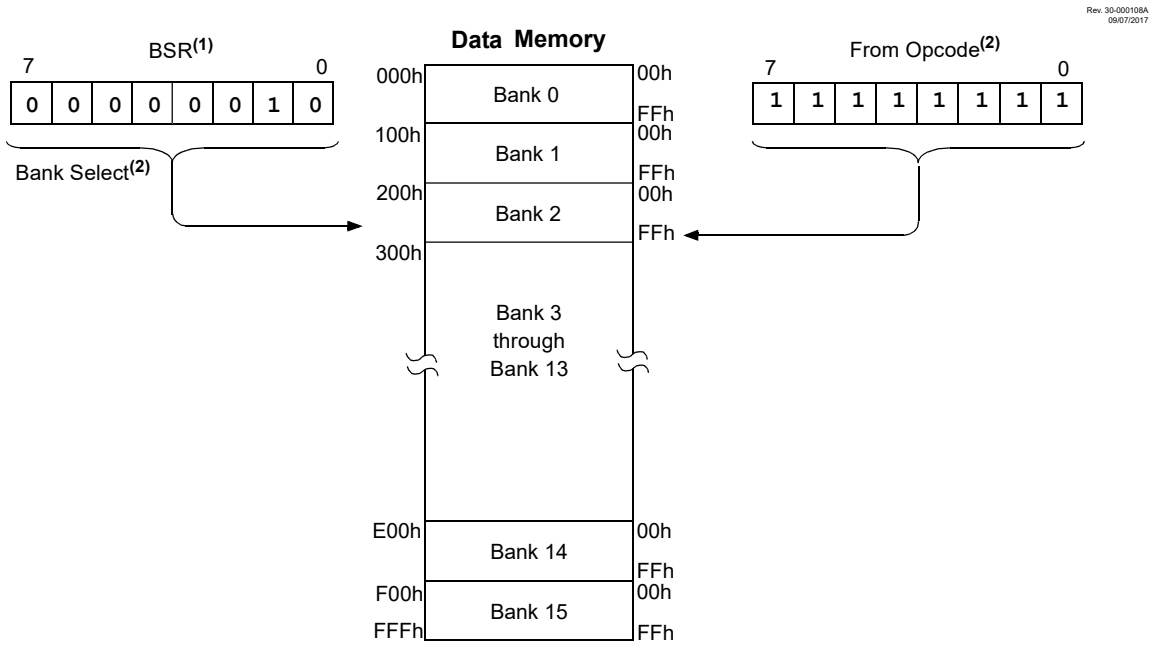
BSR 的值表示数据存储器中的存储区；指令中的 8 位表示存储区中的位置，可以认为是相对于存储区下边界的偏移量。BSR 的值与数据存储器中各存储区之间的关系如下图所示。

由于可能有最多 16 个寄存器共用相同的低位地址，因此用户必须时刻注意，确保在执行数据读操作或写操作之前选择正确的存储区。例如，当 BSR 为 0Fh 时，如果将相应的程序数据写入 F9h 的 8 位地址，则最终将复位程序计数器。

虽然可以选择任何存储区，但只有那些实际实现的存储区才能被读取或写入。写入未实现的存储区时会被忽略，而读取未实现的存储区时将返回“0”。即便如此，STATUS 寄存器仍然会受到影响，就像操作成功完成时一样。下图中的数据存储器映射指明了已实现哪些存储区。

在核心指令集中，只有 MOVFF 指令完全指定源和目标寄存器的 12 位地址。此指令在执行时完全忽略 BSR。所有其他指令仅包含低位地址（作为操作数），并且必须使用 BSR 或快速操作存储区来定位其目标寄存器。

图 13-7. 使用存储区选择寄存器（直接寻址）



Notes 1: The Access RAM bit of the instruction can be used to force an override of the selected bank (BSR[3:0]) to the registers of the Access Bank.

2: The MOVFF instruction embeds the entire 12-bit address in the instruction.

13.3.2. 快速操作存储区

尽管搭配使用 BSR 和内嵌的 8 位地址能够让用户寻址整个数据存储器范围，但这同时也意味着用户必须始终确保选择正确的存储区。否则，可能会对错误的存储单元读写数据。如果写操作的预期目标是 GPR，但写入的却是 SFR，则可能会造成灾难性后果。如果每次读取或写入数据存储器时均验证和/或更改 BSR，则效率会变得十分低下。

为了简化对最常用数据存储单元的访问，数据存储器配有快速操作存储区，允许用户访问映射的存储器块而无需指定 BSR。快速操作存储区由存储区 0 中前 96 个字节的存储单元（00h-5Fh）和块 15 中最后 160 个字节的存储单元（60h-FFh）组成。下半部分称为“快速操作 RAM”，由 GPR 组成。上半部分也是器件的 SFR 映射的位置。这两个区域在快速操作存储区中连续映射，可以通过 8 位地址以线性方式寻址（见数据存储器映射）。

快速操作存储区由包括快速操作 RAM 位（指令中的“a”参数）的核心指令使用。当“a”等于 1 时，指令搭配使用 BSR 和操作码中包含的 8 位地址来访问数据存储器地址。但是，当“a”为 0 时，将强制指令使用快速操作存储区地址映射；此时会完全忽略 BSR 的当前值。

使用这种“强制”寻址方案，指令可在单个周期内对数据地址进行操作，而无需首先更新 BSR。对于 60h 及以上的 8 位地址，这意味着用户可以更高效地评估和操作 SFR。快速操作 RAM 中 60h 以下的地址区域非常适合存储用户需要快速访问的数据值，例如即时计算结果或常用程序变量。此外，快速操作 RAM 还能够实现更快、代码效率更高的变量现场保护与切换。

使能扩展指令集（XINST 配置位 = 1）时，快速操作存储区的映射略有不同。相关内容将在在立即数变址寻址模式下映射快速操作存储区一节中更为详细地讨论。

13.3.3. 通用寄存器文件

器件可以在 GPR 区域中具有存储器分区。此分区是数据 RAM，可供所有指令使用。GPR 从存储区 0（地址 000h）的底部开始，向上延伸到 SFR 区域的底部。GPR 在发生上电复位时不会进行初始化，并且在发生所有其他复位时也不会更改。

13.3.4. 特殊功能寄存器

特殊功能寄存器（SFR）是供 CPU 和外设模块用于控制所需器件操作的寄存器。这类寄存器以静态 RAM 的形式实现。SFR 从数据存储器顶部（FFFh）开始并向下延伸。寄存器汇总表中列出了这类寄存器。

SFR 可分为两组：一组与“核心”器件功能（ALU、复位和中断）相关，另一组与外设功能相关。复位和中断寄存器将在各自对应的章节中介绍，而 ALU 的 STATUS 寄存器将在本节后面部分介绍。与外设功能操作相关的寄存器将在对应的外设章节中介绍。

SFR 通常分布在功能受其控制的外设之间。未使用的 SFR 单元未实现，读为 0。

13.3.5. 状态寄存器

STATUS 寄存器包含 ALU 的算术运算状态。与任何其他 SFR 一样，该寄存器可以是任何指令的操作数。

如果 STATUS 寄存器是影响 Z、DC、C、OV 或 N 位的指令的目标寄存器，则不写入指令的结果；相反，STATUS 寄存器将根据执行的指令进行更新。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` 会将 Z 位置 1，同时使其余状态位保持不变（“000u u1uu”）。

建议仅使用 BCF、BSF、SWAPF、MOVFF 和 MOVWF 指令来改变 STATUS 寄存器的值，因为这些指令不会影响 STATUS 寄存器中的 Z、C、DC、OV 或 N 位。

有关不影响状态位的其他指令，请参见指令集汇总。

重要：在减法运算中，C 位和 DC 位分别作为借位和半借位。

13.4. 数据空间寻址模式

重要：使能扩展指令集时，核心指令集中某些指令的执行会发生变化。更多信息，请参见数据存储器和扩展指令集一节。

可通过多种方式寻址数据存储空间中的信息。对于大多数指令，寻址模式是固定的。其他指令最多可使用三种模式，具体取决于使用的操作数以及是否使能扩展指令集。

寻址模式包括：

- 固有寻址
- 立即数寻址
- 直接寻址
- 间接寻址

使能扩展指令集（XINST 配置位 = 1）时，还可以额外使用“立即数变址”寻址模式。立即数变址寻址中将更详细地讨论相关操作。

13.4.1. 固有寻址和立即数寻址

许多控制指令根本不需要任何参数；它们要么执行对器件产生全局影响的操作，要么在一个寄存器上以隐性方式操作。这种寻址模式称为固有寻址。示例包括 SLEEP、RESET 和 DAW。

其他指令以类似的方式工作，但在操作码中需要一个额外的显式参数。这称为立即数寻址模式，因为它们需要一些立即数值作为参数。示例包括 ADDLW 和 MOVLW，它们分别用于将立即数值加到或传送到 W 寄存器。其他示例包括 CALL 和 GOTO，其中包含一个 20 位程序存储器地址。

13.4.2. 直接寻址

直接寻址指定操作码本身内部操作的全部或部分源和/或目标地址。具体选项由指令附带的参数指定。

在核心指令集中，面向位和面向字节的指令默认使用某种形式的直接寻址。所有这些指令均包含某个 8 位立即数地址作为其最低有效字节。此地址指定其中一个数据 RAM 存储区中的寄存器地址（见通用寄存器文件）或快速操作存储区中的一个存储单元（见快速操作存储区）作为指令的数据源。

快速操作 RAM 位“a”确定如何解析地址。当“a”为 1 时，BSR 的内容（见存储区选择寄存器）与地址一起用于确定寄存器的完整 12 位地址。当“a”为 0 时，地址被解析为快速操作存储区中的寄存器。使用快速操作 RAM 的寻址方式有时也称为直接强制寻址模式。

一些指令（如 MOVFF）的操作码中包含完整的 12 位（源或目标）地址。在这类情况下，将完全忽略 BSR。

操作结果的目标由目标位“d”确定。当“d”为 1 时，结果将存储回源寄存器，覆盖其原始内容。当“d”为 0 时，结果存入 W 寄存器。如果指令没有“d”参数，则其目标隐含在指令中（正在操作的目标寄存器或 W 寄存器）。

13.4.3. 间接寻址

间接寻址允许用户访问数据存储器中的存储单元，而无需在指令中给出固定地址。这是通过使用文件选择寄存器（File Select Register，FSR）作为指针指向要读取或写入的存储单元来实现的。由于 FSR 本身作为特殊文件寄存器位于 RAM 中，因此也可以在程序控制下直接操作。这使得 FSR 在实现数据结构（例如，数据存储器中的表和数组）时非常有用。

间接寻址寄存器是通过间接文件操作数（INDF）实现的，它允许通过自动递增、自动递减或使用其他值作为偏移量三种方式来自动操作指针值。这样便可以使用循环实现高效的代码，例如下面关于清空整个 RAM 存储区的示例。

例 13-3. 如何使用间接寻址清空 RAM（存储区 1）

```
     LFSR    FSR0,100h    ; Set FSR0 to beginning of Bank1
NEXT:
     CLRF    POSTINC0     ; Clear location in Bank1 then increment FSR0

     BTFSS   FSR0H,1      ; Has high FSR0 byte incremented to next bank?
     BRA     NEXT         ; NO, clear next byte in Bank1

CONTINUE:                 ; YES, continue
```

13.4.3.1. FSR 寄存器和 INDF 操作数

间接寻址的核心是三组寄存器：FSR0、FSR1 和 FSR2。每组寄存器均为一对 8 位寄存器 FSRnH 和 FSRnL。每个 FSR 对保存 12 位值，因此不使用 FSRnH 寄存器的高 4 位。12 位 FSR 值可以线性方式寻址整个数据存储器范围。FSR 寄存器随后用作指向数据存储单元的指针。

间接寻址通过一组间接文件操作数（INDF0 到 INDF2）来实现。可将这些操作数视为“虚拟”寄存器；它们映射到 SFR 空间中，但没有在物理上实现。读写特定的 INDF 寄存器实际上访问的是其对应的 FSR 寄存器对。例如，读取 INDF1 时读取的是 FSR1H:FSR1L 指示的地址处的数据。使用 INDF 寄存器作为操作数的指令实际上使用的是其对应 FSR 的内容作为指向指令目标的指针。INDF 操作数只是一种使用指针的便捷方式。

由于间接寻址使用完整的 12 位地址，因此无论 BSR 值如何，FSR 值均可以定位到任何存储区中的任何存储单元。但是，必须将快速操作 RAM 位清零，以确保快速操作空间中的 INDF 寄存器是操作的对象，而不是其他存储区之一中的寄存器。在定位任何间接操作数时，快速操作 RAM 位的汇编器默认值为零。

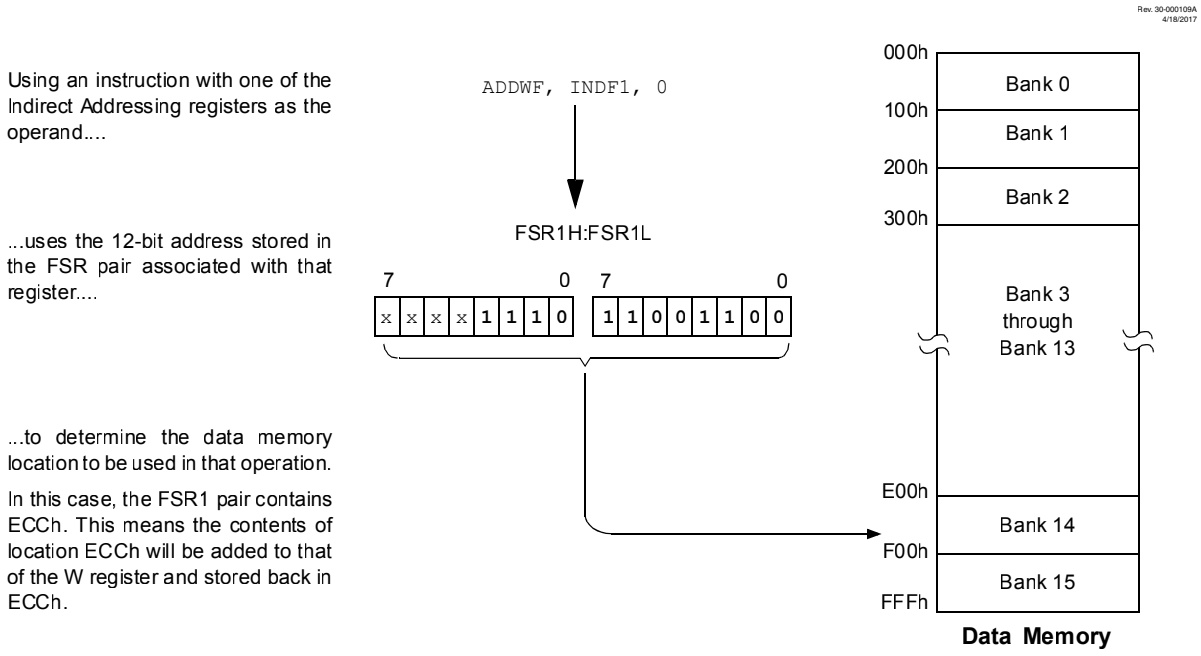
13.4.3.2. FSR 寄存器以及 POSTINC、POSTDEC、PREINC 和 PLUSW

除了 INDF 操作数之外，每个 FSR 寄存器对还有 4 个额外的间接操作数。与 INDF 一样，这些操作数也属于“虚拟”寄存器，无法直接读写。访问这些寄存器时实际上访问的是相关 FSR 寄存器对指向的存储单元，并且还会对 FSR 值执行特定操作。具体包括：

- POSTDEC：访问 FSR 指向的存储单元，然后自动将 FSR 递减 1
- POSTINC：访问 FSR 指向的存储单元，然后自动将 FSR 递增 1
- PREINC：自动将 FSR 递增 1，然后在操作中使用 FSR 指向的存储单元
- PLUSW：将 W 寄存器的有符号值（介于-127 和 128 之间）加到 FSR 的有符号值中，然后在操作中使用结果指向的存储单元

在这种情况下，访问 INDF 寄存器时使用的是相关 FSR 寄存器中的值，并且不会对其进行更改。类似地，访问 PLUSW 寄存器时会使 FSR 值发生偏移，偏移量为 W 寄存器中的值；但是 W 和 FSR 实际上在操作中都没有发生改变。访问其他虚拟寄存器时会更改 FSR 寄存器的值。

图 13-8. 间接寻址



使用 POSTDEC、POSTINC 和 PREINC 对 FSR 进行操作时会影响整个寄存器对；也就是说，当 FSRnL 寄存器从 FFh 回卷到 00h 时，会进位到 FSRnH 寄存器。另一方面，这些操作的结果不会改变 STATUS 寄存器中任何标志的值（如 Z、N 和 OV 等）。

PLUSW 寄存器可用于在数据存储空间中实现某种形式的变址寻址。通过操作 W 寄存器中的值，用户可以到达相对于指针地址有固定偏移量的地址。在某些应用程序中，这可用于在数据存储器内部实现一些强大的程序控制结构，例如软件堆栈。

13.4.3.3. FSR 对 FSR 的操作

针对其他 FSR 或虚拟寄存器的间接寻址操作属于特殊情况。例如，使用 FSR 指向其中一个虚拟寄存器时，最终将不会成功完成操作。在特定情况下，假设 FSR0H:FSR0L 包含 INDF1 的地址。尝试使用 INDF0 作为操作数读取 INDF1 的值时将返回 00h。尝试使用 INDF0 作为操作数写入 INDF1 时将执行 NOP 指令。

另一方面，使用虚拟寄存器写入 FSR 对时可能会与预期不符。在这类情况下，数值将写入 FSR 对，但不会递增或递减。因此，无论写入 INDF2 还是 POSTDEC2 寄存器均会将相同的值写入 FSR2H:FSR2L。

由于 FSR 是映射在 SFR 空间中的物理寄存器，因此可以通过所有直接操作对其进行操作。用户在操作这些寄存器时必须十分谨慎，特别是在其代码使用间接寻址时。

同样，所有其他 SFR 上都允许通过间接寻址进行操作。用户必须保持适度的谨慎，避免无意中更改可能影响器件操作的设置。

13.5. 数据存储器和扩展指令集

使能扩展指令集（XINST 配置位 = 1）会导致数据存储器及其寻址模式在某些方面发生显著变化。具体而言，许多核心指令对于快速操作存储区的使用都会有所不同；这是由于为数据存储空间引入了新的寻址模式所致。

没有发生改变的部分同样也很重要。数据存储空间的大小及其线性寻址模式没有发生改变。SFR 映射保持不变。核心指令仍然可以在直接和间接寻址模式下工作；固有指令和立即数指令没有发生任何改变。使用 FSR0 和 FSR1 的间接寻址也保持不变。

13.5.1. 立即数变址寻址

使能扩展指令集时，会更改使用 FSR2 寄存器对的间接寻址在快速操作 RAM 中的行为。在适当的条件下，使用快速操作存储区的指令（即大多数面向位和面向字节的指令）可以利用指令中指定的偏移量调用某种形式的变址寻址。这种特殊的寻址模式称为立即数变址寻址。

使用扩展指令集时，此寻址模式需要满足以下条件：

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

在这些条件下，指令的文件地址不会被解析为地址的低字节（在直接寻址中与 BSR 一起使用）或快速操作存储区中的 8 位地址。相反，该值会被解析为相对于地址指针的偏移值（由 FSR2 指定）。将偏移值与 FSR2 的内容相加可获得操作的目标地址。

13.5.2. 受立即数变址寻址模式影响的指令

任何可使用直接寻址的内核指令都可能受立即数变址寻址模式的影响。这类指令包括所有面向字节和面向位的指令，几乎占标准指令集的一半。仅使用固有或立即数寻址模式的指令不受影响。

此外，如果面向字节和面向位的指令不使用快速操作存储区（快速操作 RAM 位为 1）或者包含 60h 或以上的文件地址，则不会受到影响。满足上述条件的指令将继续像以前一样执行。下图对比了使能扩展指令集时的不同寻址模式。

如果希望在立即数变址寻址模式下使用面向字节或面向位的指令，则必须注意这种模式的汇编语法变化。相关内容将在**“扩展指令语法”**一节中更为详细地介绍。

图 13-9. 面向位和面向字节的指令的寻址选项对比（使能扩展指令集）

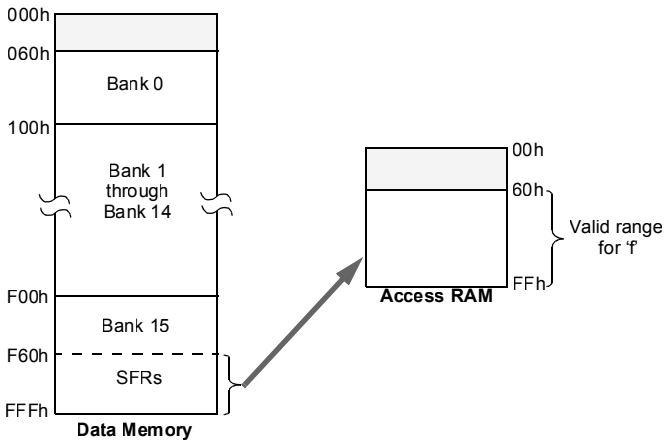
Rev. 30-000110A
4/18/2017

EXAMPLE INSTRUCTION: `ADDWF, f, d, a` (Opcode: `0010 01da ffff ffff`)

When 'a' = 0 and f ≥ 60h:

The instruction executes in Direct Forced mode. 'f' is interpreted as a location in the Access RAM between 060h and 0FFh. This is the same as locations F60h to FFFh (Bank 15) of data memory.

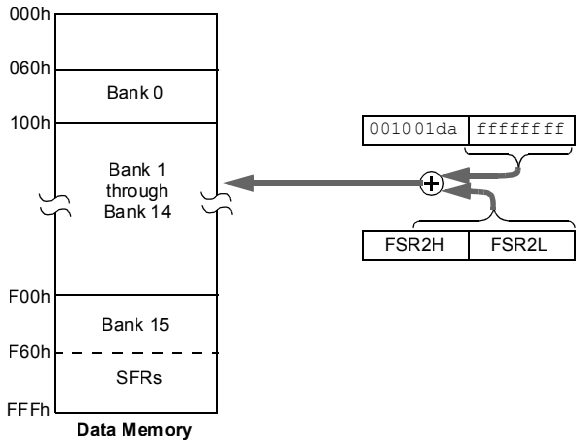
Locations below 60h are not available in this addressing mode.



When 'a' = 0 and f ≤ 5Fh:

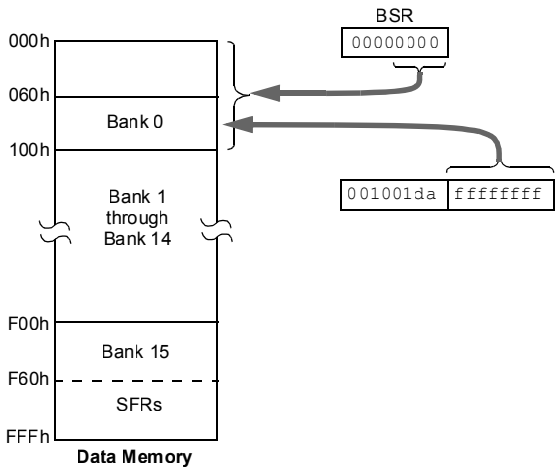
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

Note that in this mode, the correct syntax is now:
`ADDWF [k], d`
where 'k' is the same as 'f'.



When 'a' = 1 (all values of f):

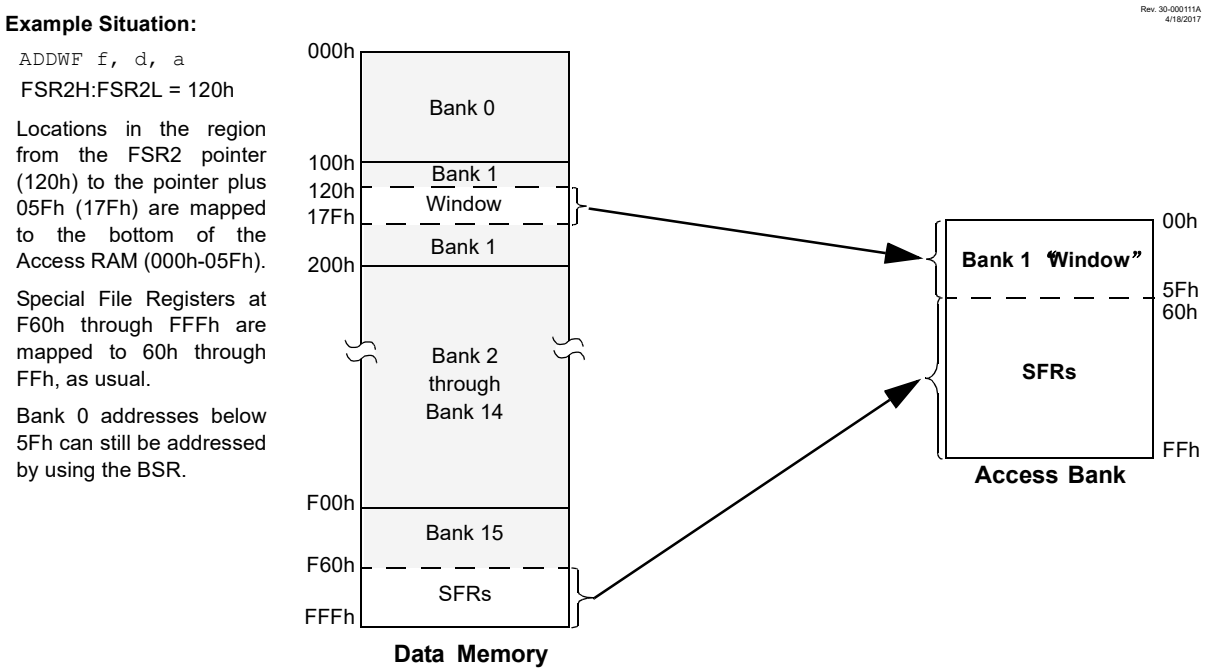
The instruction executes in Direct mode (also known as Direct Long mode). 'f' is interpreted as a location in one of the 16 banks of the data memory space. The bank is designated by the Bank Select Register (BSR). The address can be in any implemented bank in the data memory space.



13.5.3. 在立即数变址寻址模式下映射快速操作存储区

立即数变址寻址模式的使用有效地改变了快速操作 RAM（00h 到 5Fh）的前 96 个存储单元的映射方式。此模式并非仅包含存储区 0 底部的内容，它还会映射用户定义的“窗口”中的内容，此窗口可位于数据存储空间中的任何位置。FSR2 的值建立映射到此窗口的地址的下边界，而上边界由 FSR2 与 95（5Fh）之和定义。快速操作 RAM 中 5Fh 以上的地址按照前文所述进行映射（见快速操作存储区）。下图给出了此寻址模式下的快速操作存储区重映射示例。

图 13-10. 使用立即数变址寻址重映射快速操作存储区



重映射快速操作存储区仅适用于使用立即数变址寻址模式的操作。使用 BSR（快速操作 RAM 位为 1）的操作将继续像以前一样使用直接寻址。

13.6. 指令执行和扩展指令集

使能扩展指令集时会向现有指令集中添加 8 个附加命令。这些指令按照*扩展指令集*一节中所述执行。

13.7. 寄存器定义：存储器和状态

13.7.1. PCL

名称： PCL
偏移量： 0xFF9

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

13.7.2. PCLAT

名称： PCLAT
偏移量： 0xFFA

程序计数器锁存器。程序计数器（PC）bit [21:9]的保持寄存器。读取 PCL 寄存器时会将 PC 的高位传送到 PCLAT 寄存器。写入 PCL 寄存器时会将 PCLAT 值传送到 PC。

位	15	14	13	12	11	10	9	8
				PCLATU[4:0]				
访问				R/W	R/W	R/W	R/W	R/W
复位				0	0	0	0	0

位	7	6	5	4	3	2	1	0
	PCLATH[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 12:8 – PCLATU[4:0] PC 锁存器最高字节寄存器
程序计数器 bit [21:17]的保持寄存器

Bit 7:0 – PCLATH[7:0] PC 锁存器高字节寄存器
程序计数器 bit [16:8]的保持寄存器

13.7.3. TOS

名称： TOS
偏移量： 0xFFD

栈顶寄存器。
STKPTR 寄存器指向的堆栈的内容。这是在 RETURN 或 RETFIE 指令之后装入程序计数器的值。

位	23	22	21	20	19	18	17	16
				TOSU[4:0]				
访问				R/W	R/W	R/W	R/W	R/W
复位				0	0	0	0	0

位	15	14	13	12	11	10	9	8
	TOSH[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

位	7	6	5	4	3	2	1	0
	TOSL[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 20:16 – TOSU[4:0] TOS 寄存器的最高字节
TOS 的 Bit [21:17]

Bit 15:8 – TOSH[7:0] TOS 寄存器的高字节
TOS 的 Bit [16:8]

Bit 7:0 – TOSL[7:0] TOS 寄存器的低字节
TOS 的 Bit [7:0]

13.7.4. STKPTR

名称： STKPTR
偏移量： 0xFFC

堆栈指针寄存器

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

Bit 4:0 – STKPTR[4:0] 堆栈指针位置位

13.7.5. STATUS

名称： STATUS
偏移量： 0xFD8

状态寄存器

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

Bit 6 – $\overline{TO}$ 超时位

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 5 – $\overline{PD}$ 掉电位

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 4 – N 负标志位
用于有符号的算术运算（二进制补码）；指示结果是否为负（ALU MSb = 1）。

复位状态： POR/BOR = 0
所有其他复位 = u

值	说明
1	结果为负
0	结果为正

Bit 3 – OV 溢出位
用于有符号的算术运算（二进制补码）；指示 7 位幅值溢出，这会导致符号位（bit 7）改变状态。

复位状态： POR/BOR = 0
所有其他复位 = u

值	说明
1	当前有符号算术运算发生溢出
0	未发生溢出

Bit 2 – Z 全零标志位

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/$\overline{\text{借位}}$位

ADDWF、ADDLW、SUBLW 和 SUBWF 指令(1)

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/$\overline{\text{借位}}$位

ADDWF、ADDLW、SUBLW 和 SUBWF 指令(1,2)

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$\overline{\text{借位}}$，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。
2. 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。

13.7.6. WREG

名称： WREG
偏移量： 0xFE8

工作数据寄存器的影子寄存器

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	x	x	x	x	x	x	x	x

Bit 7:0 – WREG[7:0]

13.7.7. INDF

名称： INDFx
偏移量： 0xFEF,0xFE7,0xFDF

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

位	7	6	5	4	3	2	1	0
	INDF[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 – INDF[7:0]
FSRx 寄存器指向的间接数据

13.7.8. POSTDEC

名称： POSTDECx
偏移量： 0xFED,0xFE5,0xFDD

采用后递减的间接数据寄存器。该寄存器是虚拟寄存器。由 FSRx 寄存器寻址的 GPR/SFR 寄存器是所有操作（包括 POSTDECx 寄存器）的目标寄存器。FSRx 在读/写操作之后递减。

位	7	6	5	4	3	2	1	0
	POSTDEC[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 – POSTDEC[7:0]

13.7.9. POSTINC

名称： POSTINCx
偏移量： 0xFEE,0xFE6,0xFDE

采用后递增的间接数据寄存器。该寄存器是虚拟寄存器。由 FSRx 寄存器寻址的 GPR/SFR 寄存器是所有操作（包括 POSTINCx 寄存器）的目标寄存器。FSRx 在读/写操作之后递增。

位	7	6	5	4	3	2	1	0
	POSTINC[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 – POSTINC[7:0]

13.7.10. PREINC

名称： PREINCx
偏移量： 0xFEC,0xFE4,0xFDC

采用预递增的间接数据寄存器。该寄存器是虚拟寄存器。由“FSRx 寄存器 + 1”寻址的 GPR/SFR 寄存器是所有操作（包括 PREINCx 寄存器）的目标寄存器。FSRx 在读/写操作之前递增。

位	7	6	5	4	3	2	1	0
	PREINC[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 – PREINC[7:0]

13.7.11. PLUSW

名称： PLUSWx
偏移量： 0xFEB,0xFE3,0xFDB

采用 WREG 偏移量的间接数据寄存器。该寄存器是虚拟寄存器。由“FSRx 寄存器 + W 寄存器的有符号值”寻址的 GPR/SFR 寄存器是所有操作（包括 PLUSWx 寄存器）的目标寄存器。

位	7	6	5	4	3	2	1	0
	PLUSW[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 – PLUSW[7:0]

13.7.12. FSR

名称： FSRx
偏移量： 0xFE9,0xFE1,0xFD9

间接地址寄存器。FSR 值是 INDF 寄存器指向的数据的地址。

位	15	14	13	12	11	10	9	8
					FSRH[3:0]			
访问					R/W	R/W	R/W	R/W
复位					0	0	0	0

位	7	6	5	4	3	2	1	0
	FSRL[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 11:8 – FSRH[3:0]
INDF 数据的最高有效地址

Bit 7:0 – FSRL[7:0]
INDF 数据的最低有效地址

13.7.13. BSR

名称： BSR
偏移量： 0xFE0

存储区选择寄存器
BSR 表示对应于 GPR 地址的 bit<11:8>的数据存储区。

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

Bit 3:0 – BSR[3:0]
数据存储地址的高四位

13.8. 寄存器汇总——存储器和状态

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0FD7	保留									
0x0FD8	STATUS	7:0		$\overline{TO}$	$\overline{PD}$	N	OV	Z	DC	C
0x0FD9	FSR2	7:0	FSRL[7:0]							
		15:8					FSRH[3:0]			
0x0FDB	PLUSW2	7:0	PLUSW[7:0]							
0x0FDC	PREINC2	7:0	PREINC[7:0]							
0x0FDD	POSTDEC2	7:0	POSTDEC[7:0]							
0x0FDE	POSTINC2	7:0	POSTINC[7:0]							
0x0FDF	INDF2	7:0	INDF[7:0]							
0x0FE0	BSR	7:0					BSR[3:0]			
0x0FE1	FSR1	7:0	FSRL[7:0]							
		15:8					FSRH[3:0]			
0x0FE3	PLUSW1	7:0	PLUSW[7:0]							
0x0FE4	PREINC1	7:0	PREINC[7:0]							
0x0FE5	POSTDEC1	7:0	POSTDEC[7:0]							
0x0FE6	POSTINC1	7:0	POSTINC[7:0]							
0x0FE7	INDF1	7:0	INDF[7:0]							
0x0FE8	WREG	7:0	WREG[7:0]							
0x0FE9	FSR0	7:0	FSRL[7:0]							
		15:8					FSRH[3:0]			
0x0FEB	PLUSW0	7:0	PLUSW[7:0]							
0x0FEC	PREINC0	7:0	PREINC[7:0]							
0x0FED	POSTDEC0	7:0	POSTDEC[7:0]							
0x0FEE	POSTINC0	7:0	POSTINC[7:0]							
0x0FEF	INDF0	7:0	INDF[7:0]							
0x0FF0 ... 0x0FF8	保留									
0x0FF9	PCL	7:0	PCL[7:0]							
0x0FFA	PCLAT	7:0	PCLATH[7:0]							
		15:8				PCLATU[4:0]				
0x0FFC	STKPTR	7:0				STKPTR[4:0]				
0x0FFD	TOS	7:0	TOSL[7:0]							
		15:8	TOSH[7:0]							
		23:16				TOSU[4:0]				

14. NVM——非易失性存储器控制

非易失性存储器分为三类：程序闪存（PFM）（包括用户 ID）、配置字和数据闪存（Data Flash Memory，DFM）。DFM 也称为 EEPROM，因为它一次写入一个字节，并且会在写入前自动擦除。虽然用户 ID 位于 PFM 部分之上，但其包含在 PFM 类别中，因为二者的读写访问权限相同。

写入和擦除时间由片上定时器控制。写入和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 和 DFM 可通过两种方式进行保护：代码保护和写保护。代码保护（用于 PFM 的配置位 $\overline{CP}$ 和用于 DFM 的配置位 $\overline{CPD}$）通过外部器件编程器禁止读访问和写访问。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。代码保护和写保护只能通过外部器件编程器执行的批量擦除来复位。PFM 批量擦除会清除程序空间、配置位和用户 ID。批量擦除仅在 $\overline{CPD}$= 0 时才会清除 DFM。当 $\overline{CPD}$= 1 时，批量擦除 DFM 需要在发出批量擦除命令之前将程序计数器设置为 DFM 区域，然后仅清除 DFM 区域。有关详细信息，请参见编程规范。写保护可防止写入标记为受 WRTn 配置位保护的 NVM 区域。尝试写入受保护的存储单元会将 NVMERR 位置 1。

可通过表指针或 NVM 控制访问 PFM 和配置字。只能通过 NVM 控制访问 DFM。PFM 访问按单字节、单字或完整扇区进行。扇区为 256 字节（128 个 PFM 字）。扇区存储器占用 RAM 空间的一个完整存储区（位于 RAM 存储区中，处于最后占用的 GPR 存储区之后）。在本文档的其他位置，扇区存储器也称为写入块保持寄存器。表指针按字节访问存储器。NVM 控制按字节访问 DFM 部分，对于其他部分则按字和扇区访问。

NVM 控制包括五个独立的访问功能和五个相应的控制位。控制功能如下：

- RD——单字节/字读取
- WR——单字节/字写入
- SECRD——扇区读取
- SECWR——扇区写入
- SECER——扇区擦除

NVMADR 寄存器确定 NVM 控制正在访问的存储区的地址。TBLPTR 寄存器确定表指针功能正在访问的存储器的地址。下表显示了每个区域中的控制操作。

表 14-1. NVM 构成表

区域	地址范围	表指针 TBLRD	NVMCON1				
			RD	WR	SECRD	SECWR	SECER
PFM	00 0000h 01 FFFFh	●	●	●	●	●	●
用户 ID	20 0000h 20 00FFh	●	●	●	●	●	●
CONFIG	30 0000h 30 000Bh	●	●		●	●	
DFM	31 0000h 31 00FFh		●	●			

14.1. 程序闪存

在整个 V_{DD} 范围内的正常工作期间，程序闪存可读写且可擦除。

一次从程序存储器读取一个字节。一次对 n 个字节的块（也称为扇区）执行一次程序存储器擦除。有关写入和擦除块大小，请参见下表。无法从用户代码发出批量擦除操作。可按扇区或单字对程序存储器进行写操作。

表 14-2. 各器件闪存构成

器件	扇区擦除大小（字）	保持寄存器（字节）	TBLPTR LSb（保持地址）	闪存程序存储器（字）	数据闪存（字节）
CN2510	128	256	8	16384	256
CN2610				32768	1024
CN2710				65536	

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。内部编程定时器终止程序存储器写入和擦除。

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。

要进行擦除和编程操作，了解 PFM 存储器结构非常重要。程序存储器字大小为 16 位宽。PFM 按扇区排列。扇区是可以通过用户软件擦除的最小大小。

擦除某个扇区后，可对该扇区的全部或部分内容进行编程。可使用 NVMADR 和 NVMCON1 控制将数据直接写入 PFM 中，一次写入一个 16 位字，也可采用完整扇区形式从扇区 RAM（也称为保持寄存器）进行写入。这些 8 位寄存器位于 RAM 存储区（位于最后一个 GPR RAM 存储区之后）。保持寄存器可以像任何其他 SFR/GPR 寄存器那样直接访问，也可通过连续写操作使用 TABLAT 和 TBLPTR 寄存器进行装载。

重要：如果只修改先前已编程扇区的一部分内容，则必须在擦除扇区之前先读取整个扇区的内容，并保存到 RAM 中。最简单的方法是 SECRD 操作。然后，可以将新数据写入保持寄存器以重新编程 PFM 的扇区。不过，可使用单字写操作写入任何未编程的存储单元，而无需先擦除扇区。

14.1.1. 表指针操作

为读写程序存储器，有两种操作允许处理器在程序存储空间和数据 RAM 之间传送字节：

- 表读（TBLRD*）
- 表写（TBLWT*）

与这些操作相关的 SFR 寄存器包括：

- TABLAT 寄存器
- TBLPTR 寄存器

程序存储空间为 16 位宽，而数据 RAM 空间为 8 位宽。TBLPTR 寄存器确定 NVM 存储器的一个字节的地址。表读将一个字节的数据从 NVM 空间传送到 TABLAT 寄存器，表写将 TABLAT 数据传送到保持寄存器，以准备随后通过 NVM 控制写入 NVM 空间。

14.1.1.1. 表指针寄存器

表指针（TBLPTR）寄存器寻址程序存储器中的一个字节。TBLPTR 包含三个 SFR 寄存器：表指针最高字节、表指针高字节和表指针低字节（TBLPTRU:TBLPTRH:TBLPTRL）。这三个寄存器共同形成一个 22 位宽的指针（位 0 到 21）。位 0 到 20 允许器件寻址最高 2 MB 的程序存储空间。位 21 允许访问器件 ID、用户 ID 和配置位。

表指针寄存器 TBLPTR 由 TBLRD 和 TBLWT 指令使用。这些指令可递增和递减 TBLPTR，具体取决于特定的附加字符，如下表所示。TBLPTR 的递增和递减操作仅影响位 0 到 20。

表 14-3. 通过 TBLRD 和 TBLWT 指令进行的表指针操作

示例	对表指针的操作
TBLRD* TBLWT*	TBLPTR 未修改

表 14-3. 通过 TBLRD 和 TBLWT 指令进行的表指针操作 （续）

示例	对表指针的操作
TBLRD*+ TBLWT*+	TBLPTR 在读/写操作之后递增
TBLRD*- TBLWT*-	TBLPTR 在读/写操作之后递减
TBLRD+* TBLWT+*	TBLPTR 在读/写操作之前递增

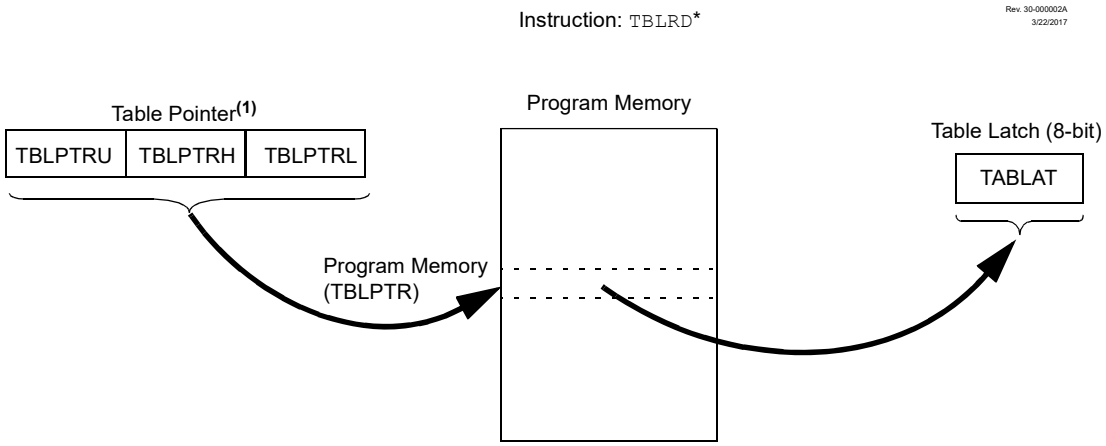
14.1.1.2. 表锁存器寄存器

表锁存器（TABLAT）是一个映射到 SFR 空间的 8 位寄存器。表锁存器寄存器接收由 TBLRD*指令产生的一个字节 NVM 数据，此寄存器是因 TBLWT*指令而发送到保持寄存器空间的 8 位数据的来源。

14.1.1.3. 表读操作

表读操作直接从 TBLPTR 寄存器指向的程序存储器中检索一个字节的数据，并将其置于 TABLAT 寄存器中。 图 14-1 显示了表读操作。

图 14-1. 表读操作

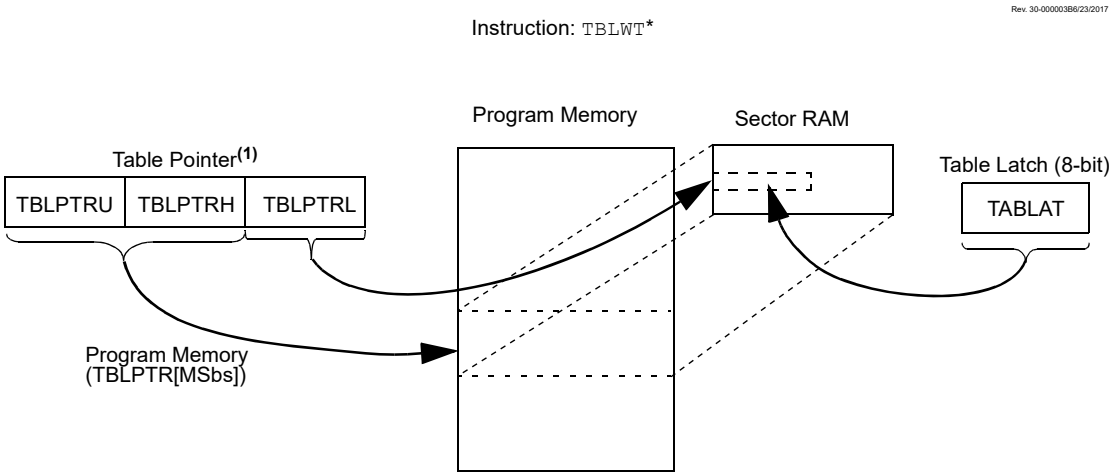


Note 1: Table Pointer register points to a byte in program memory.

14.1.1.4. 表写操作

表写操作将来自 TABLAT 寄存器的一个字节数据存储到扇区 RAM 保持寄存器中。下图显示了从 TABLAT 寄存器到保持寄存器空间的表写操作。 **“写入程序闪存”** 部分详细介绍了将保持寄存器的内容写入程序存储器的过程。

图 14-2. 表写操作



Note 1: During table writes the Table Pointer points to two areas. The LSbs of TBLPRTL point to an address within the sector RAM space. The MSbs of the Table Pointer determine where the sector will eventually be written.

表操作使用字节。包含数据（而非程序指令）的表无需按字对齐。因此，表可以在任意字节地址处开始和结束。如果使用表写操作将可执行代码写入程序存储器，则程序指令需要按字对齐。

14.1.1.5. 表指针边界

TBLPTR 用于程序闪存的读、写和擦除。

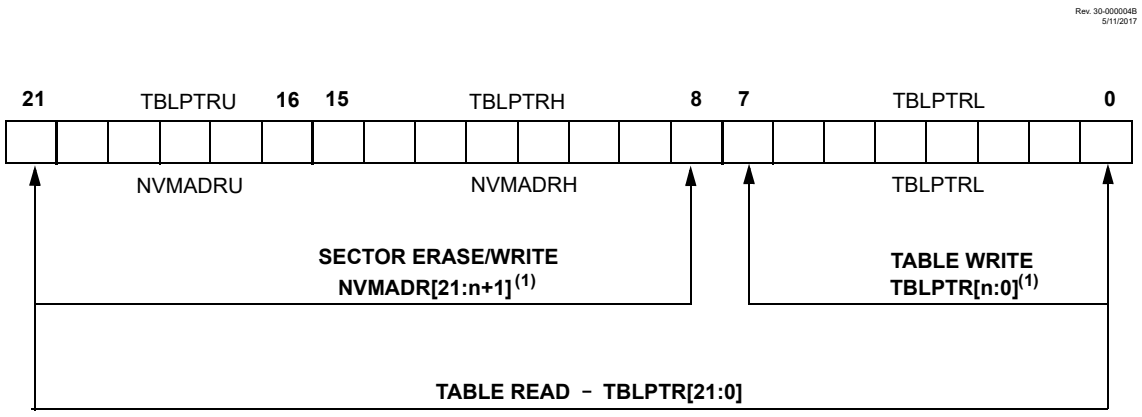
执行 TBLRD 时，TBLPTR 的全部 22 位确定直接从程序存储器读入到 TABLAT 寄存器的字节。

执行 TBLWT 时，TABLAT 寄存器中的字节被写入保持寄存器（而非闪存）以准备程序存储器写入操作。保持寄存器构成一个写入块，它可能因器件而异（见“**程序闪存**”一节中的*“各器件闪存构成”*）。TBLPTRL 寄存器的 LSb 确定保持寄存器块中的哪个特定地址被写入。写入块的大小决定了 LSb 数。在 TBLWT 操作期间，表指针的 MSb 无效。

执行 PFM 扇区写入时，整个保持寄存器块被写入闪存扇区（位于由 NVMADR 的 MSb 确定的地址处）。在扇区写入期间忽略 LSb。有关详细信息，请参见写入程序闪存部分。

下图给出了基于 NVM 控制操作的 TBLPTR 和 NVMADR 的相关边界。

图 14-3. 基于操作的表指针边界



注：

1. 有关写保持寄存器块大小，请参见“**程序闪存**”一节的*“各器件闪存构成”*表。

14.1.1.6. 读取程序闪存

TBLRD 指令从程序存储器（在 TBLPTR 存储单元处）检索数据，并将其置于 TABLAT SFR 寄存器中。从程序存储器进行表读操作，一次读取一个字节。此外，可自动修改 TBLPTR 以进行下一个表读操作。

CPU 操作在读操作期间暂停，并在完成之后立即恢复。从用户角度而言，TABLAT 在下一个指令周期内有效。

内部程序存储器通常按字组织。地址的最低有效位在字的高字节和低字节之间进行选择。 图 14-4 显示了内部程序存储器和 TABLAT 之间的接口。

图 14-4. 从程序闪存读取

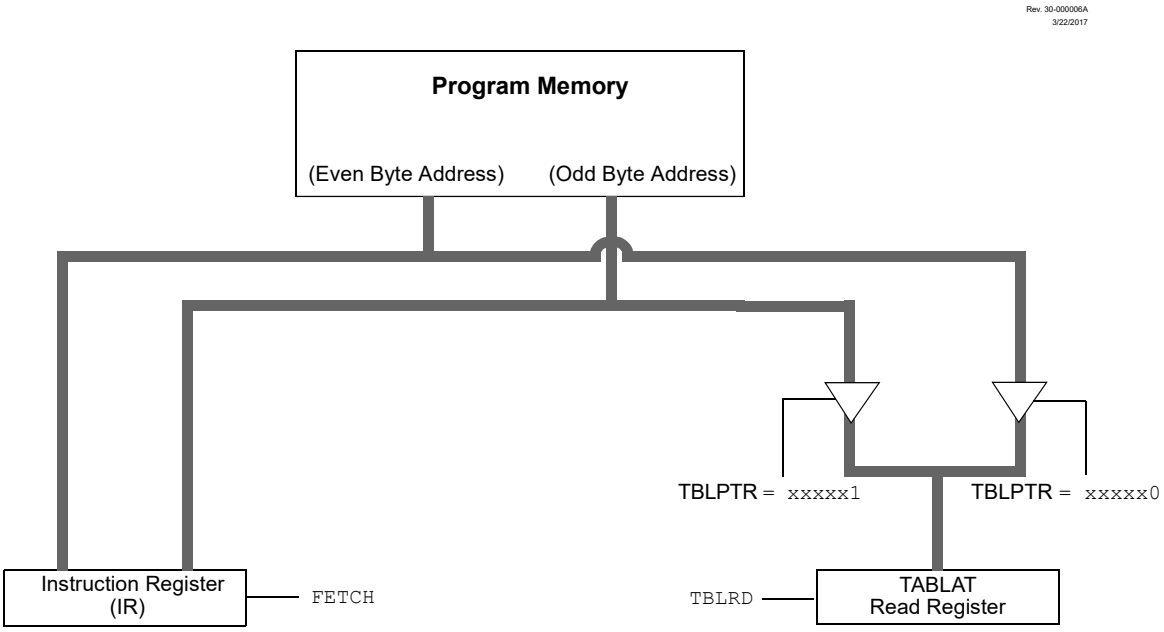
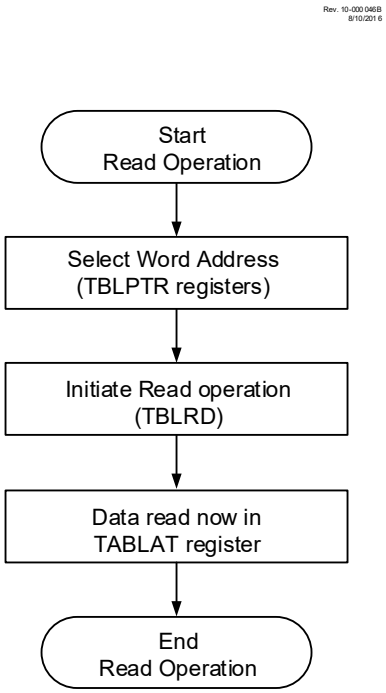


图 14-5. 程序闪存读取流程图



例 14-1. 读取程序闪存字

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

14.1.2. 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、扇区读取和擦除的机制。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 扇区擦除
- PFM 扇区从保持寄存器写入
- PFM 扇区读取到写入块保持寄存器
- PFM 字直接写入到 NVM
- DFM 字节直接写入到 NVM
- 写入到配置字

这些操作中的每一个都对应于四个解锁代码序列之一，如下表所示：

表 14-4. NVM 解锁代码

操作	第一个解锁字节	第二个解锁字节	NVMCON1 操作位
字/字节写入	55h	AAh	WR
扇区写入	DDh	22h	SECWR
扇区擦除	CCh	33h	SECER
扇区读取	BBh	44h	SECRD

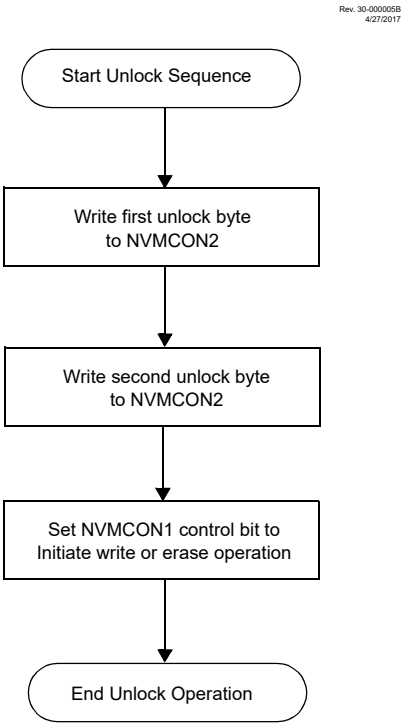
常规解锁序列由以下步骤组成且必须按照以下顺序完成：

- 将第一个解锁字节写入 NVMCON2
- 将第二个解锁字节写入 NMVCON2
- 将 NVMCON1 的操作控制位置 1

对于 PFM 和配置字操作，在控制位置 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。DFM 操作不会暂停 CPU。

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应先禁止全局中断，然后在完成解锁序列之后重新允许中断。

图 14-6. NVM 解锁序列流程图



例 14-2. 用于 PFM 字写入的 NVM 解锁序列

```
    BCF     INTCON,GIE      ; Recommended so sequence is not interrupted
    MOVF    UpperAddr,W     ; set the target address
    MOVWF   NVMADRU
    MOVF    HighAddr,W
    MOVWF   NVMADRH
    MOVF    LowAddr,W
    MOVWF   NVMADRL
```

```
    MOVF    HighByte,W      ; high byte of word to be written
    MOVWF   NVMDATH         ; store in high byte transfer register
    MOVF    LowByte,W       ; low byte of word to be written
    MOVWF   NVMDATL         ; store in low byte transfer register
    BSF     NVMCON0,NVMEN   ; Enable NVM operation
    MOVLW   55h             ; Load first unlock byte
;  ------------------ Required Sequence  -----------------------------
    MOVWF   NVMCON2         ; Step 1: Load first byte into NVMCON2
    MOVLW   AAh             ; Step 2: Load W with second unlock byte
    MOVWF   NVMCON2         ; Step 3: Load second byte into NVMCON2
    BSF     NVMCON1,WR      ; Step 4: Set WR bit to begin write
;  ---------------------------------------------------------------------
    BSF     INTCON,GIE      ; Re-enable interrupts
```

重要：

1. 解锁序列开始于写 NVMCON2；必须按所示的精确周期顺序执行步骤 1-4。如果中断或调试器暂停导致步骤 1 到 4 的时序被破坏，则不会执行该操作。
2. 操作码仅用作说明；任何具有所指示作用的指令均可使用。

14.1.3. 擦除程序闪存（PFM）

最小擦除块始终为一个扇区。只有通过使用外部编程器才能批量擦除更大的程序存储器块。不支持闪存阵列中的字擦除。

例如，从扇区擦除大小为 128 字的单片机启动擦除序列时，会擦除 128 字（256 字节）的程序存储器块。NVMADR[21:8]位指向要擦除的块。NVMADR[7:0]位被忽略。

NVMCON0 和 NVMCON1 寄存器发出擦除操作命令。为使能写操作，NVMEN 位必须置 1。为启动擦除操作，SECER 位置 1。

必须使用 NVM 解锁序列部分介绍的 NVM 解锁序列，以防止意外写入。这一序列有时称为长时间写操作。

擦除内部闪存需要长时间写操作。在长时间写操作周期内，停止指令执行。长时间写操作由内部编程定时器终止。

14.1.3.1. PFM 擦除序列

擦除内部程序存储器块的事件序列如下：

1. 将 NVMADR 设置为预期扇区内的地址
2. 将 NVMEN 位置 1 以使能 NVM
3. 按 NVM 解锁序列部分所述执行解锁序列
4. 将 SECER 位置 1

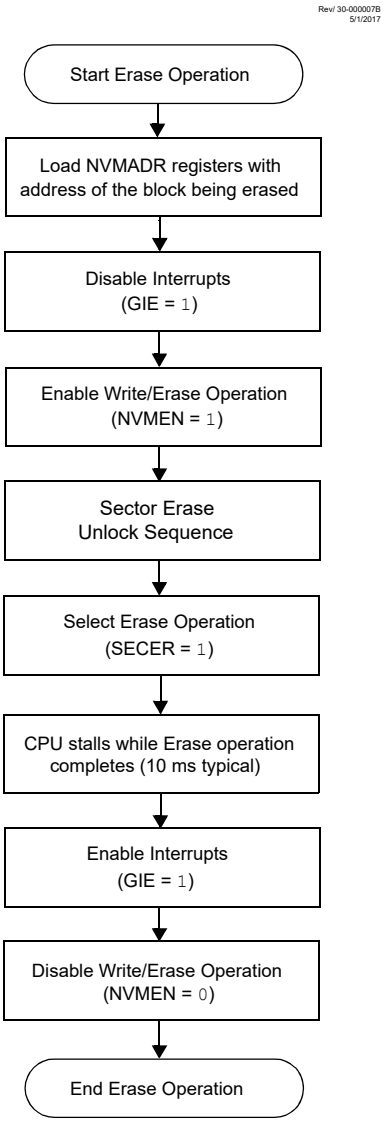
如果 PFM 地址受写保护，则清零 SECER 位且不会执行擦除操作，在这种情况下会发出 NVMERR 信号。

该操作通过屏蔽当前 NVMADR 的 LSb 来擦除所指示的扇区。

擦除 PFM 时，CPU 操作暂停，并在操作完成时恢复。操作完成时，SECER 位由硬件清零，NVMIF 置 1，并且在 NVMIE 位也置 1 时将发生中断。

保持寄存器数据不受擦除操作影响，且 NVMEN 将保持不变。

图 14-7. PFM 扇区擦除流程图



例 14-3. 擦除程序闪存块

```
; This sample sector erase routine assumes that the target address
; specified by CODE_ADDR_UPPER and CODE_ADDR_HIGH contain a value within
; the PFM address range of the device.

           MOVLW   CODE_ADDR_UPPER    ; load NVMADR with the base
           MOVWF   NVMADRU            ; address of the memory block
           MOVLW   CODE_ADDR_HIGH
           MOVWF   NVMADRH
ERASE_BLOCK:
           BSF     NVMCON0, NVMEN     ; enable Program Flash Memory
           BCF     INTCON, GIE        ; disable interrupts
Required   MOVLW   CCh                ; first unlock byte for erase
Sequence   MOVWF   NVMCON2            ; write CCh
           MOVLW   33h                ; second unlock byte for erase
           MOVWF   NVMCON2            ; write 33h
           BSF     NVMCON1, SECER     ; start erase (CPU stalls)
           BSF     INTCON, GIE        ; re-enable interrupts
           BCF     NVMCON0, NVMEN     ; disable writes to memory
```

重要：

1. 如果写入或擦除操作被意外事件终止，则 NVMERR 位将置 1，用户可对此进行检查以确定是否需要重写存储单元。
2. 在 NVMADR 指向受写保护的地址时，如果向 SECER 写入 1，则 NVMERR 置 1。
3. 在 NVMADR 指向无效地址存储单元时，如果向 SECER 写入 1，则 NVMERR 置 1（见器件存储器映射和 NVM 构成表）。

14.1.4. 写入程序闪存

程序存储器每次可写入一个字或一个扇区。

通过将 NVMADR 设置为目标地址并使用所需字装载 NVMDAT 来写入单个字。然后，基于 WR 解锁和写序列，将该字传入闪存。

通过先装载保持寄存器块然后执行扇区写序列来写入扇区。编程写入块大小被指定为保持寄存器，在各器件闪存构成表中，也被称为扇区 RAM。表写操作用于写入保持寄存器字节，然后基于 SECWR 解锁和写序列将其传入闪存。只存在与写入块中的字节一样多的保持寄存器。

表锁存器（TABLAT）只是一个字节，对于每个扇区编程操作，需要多次执行 TBLWT 指令。对于表写操作，会忽略写保护状态。所有表写操作基本上都是短时间写操作，因为只对保持寄存器进行写操作。写入保持寄存器时，NVMIF 不受影响。

写入所有保持寄存器后，通过将 NVMADR 设置为目标扇区内的地址并执行扇区写解锁序列，然后立即将 SECWR 位置 1 来启动该存储器扇区的编程操作。

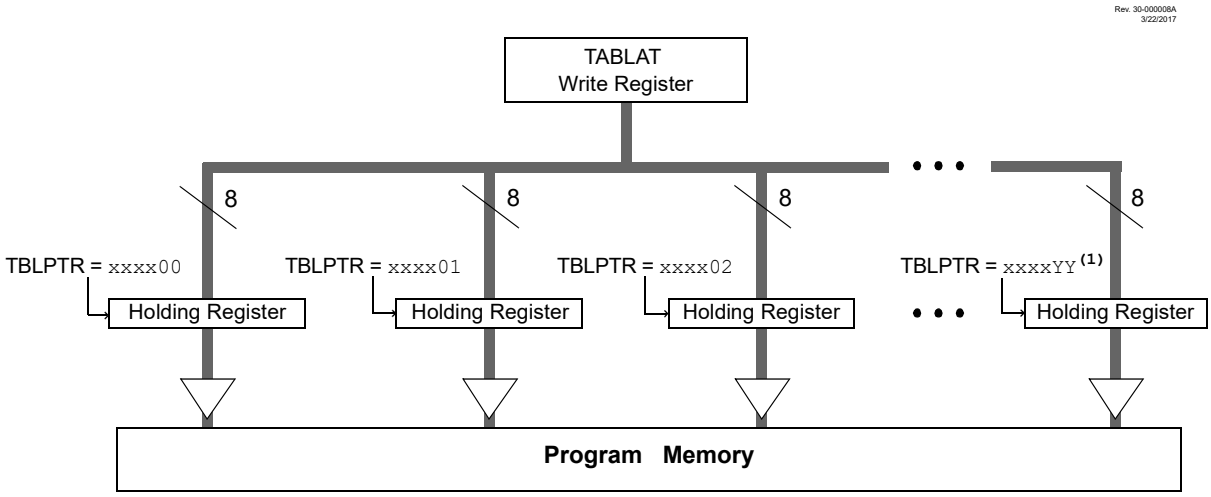
如果 NVMADR 中的 PFM 地址受写保护，或 NVMADR 指向无效存储单元，则 SECWR 位清零且无任何影响，NVMERR 将置 1。

CPU 操作在长时间写操作周期内暂停，并在操作完成后恢复。长时间写操作在一个扩展指令周期内完成。完成后，SECWR 或 WR 位由硬件清零，NVMIF 置 1。如果 NVMIE 也置 1，则会发生中断。保持寄存器不变。NVMEN 不变。

内部编程定时器控制写时间。写入/擦除电压是由片上电荷泵产生的，此电荷泵在器件的电压范围内工作。

重要：保持寄存器在器件复位时未定义，在写操作后不变。可以修改程序存储器的各个字节，前提是修改不会尝试将任何位从 0 更改为 1。基于扇区写操作修改各个字节时，必须在执行长时间写操作之前，使用 FFh 或存储器的现有内容装载所有保持寄存器。为此，执行扇区读取操作是最快的方法。

图 14-8. 对程序闪存进行表写



注：有关保持寄存器的数量，请参见各器件闪存构成表（例如，YY = FFh 表示有 256 个保持寄存器）。

14.1.4.1. PFM 扇区写序列

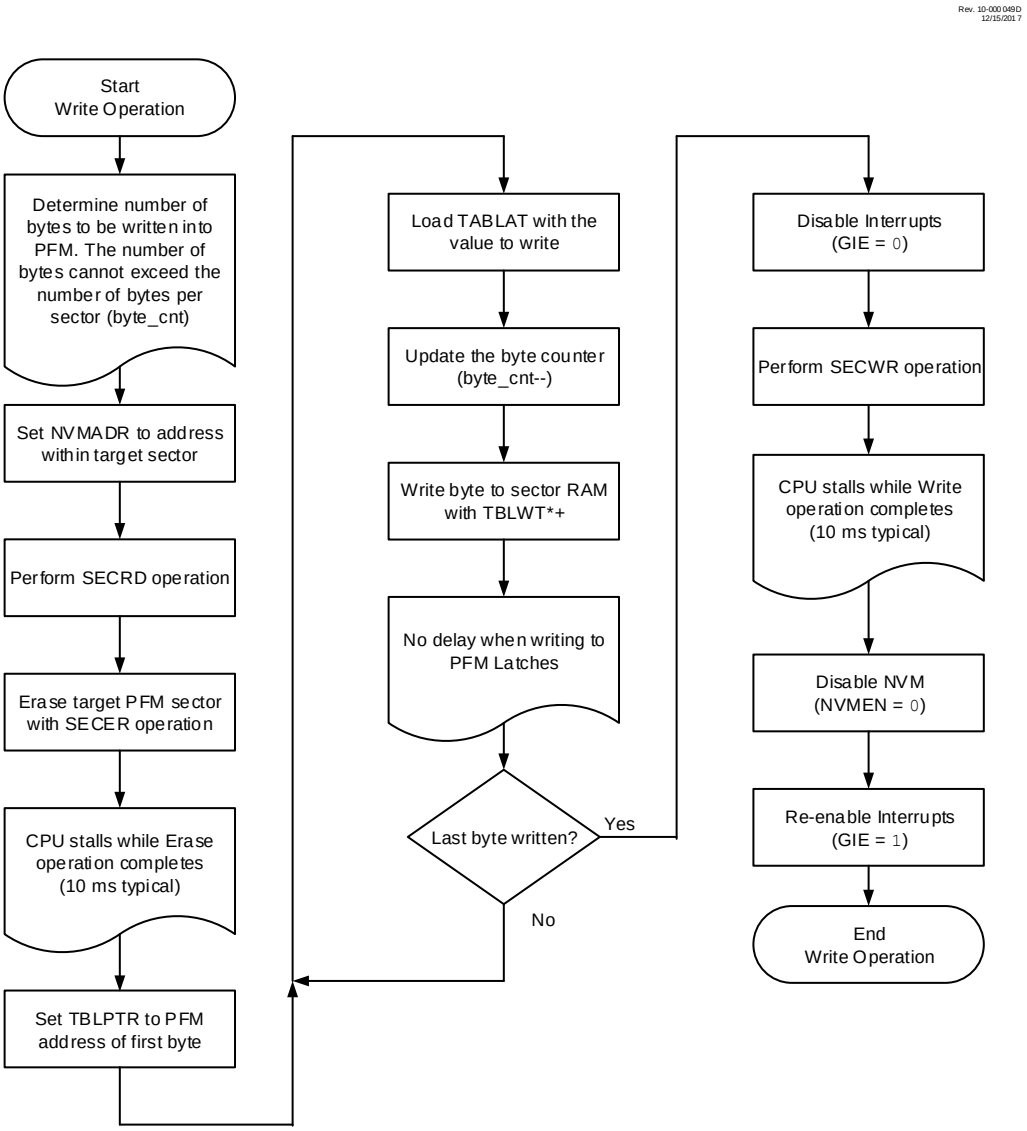
编程内部程序存储单元块的事件序列如下：

1. 使用目标扇区地址设置 NVMADR。
2. 使用 SECRD 操作将 PFM 扇区读入 RAM。
3. 执行扇区擦除过程（见 PFM 擦除序列）。
4. 作为擦除序列的最后一步，将 SECER 置 1。
5. 在擦除期间 CPU 将暂停（约 10 ms，使用内部定时器）。
6. 装载 TBLPTR，更新第一个字节的地址。
7. 采用自动递增方式将 n 字节块写入保持寄存器。有关保持寄存器的数量，请参见各器件闪存构成表。
8. 禁止中断。
9. 执行扇区写解锁序列（见 NVM 解锁序列）。
10. 作为解锁序列的最后一步，将 SECWR 位置 1。
11. 在写操作期间 CPU 将暂停（约 10 ms，使用内部定时器）。
12. 重新允许中断。
13. 校验存储器（表读）。

此过程将需要约 20 ms 来更新每个存储器块。有关详细信息，请参见“**存储器编程规范**”。下面给出了所需代码的示例。

重要：在将 SECWR 位置 1 之前，NVMADR 值必须处于目标 PFM 扇区的预期地址范围内。

图 14-9. 程序闪存（PFM）写流程图



例 14-4. 写入程序闪存的扇区

```
; Code sequence to modify one word in a programmed sector of PFM
; Calling routine should check WREG for the following errors:
;
;    00h = Successful modification
;    01h = Read error
;    02h = Erase error
;    03h = Write error
;
READ_BLOCK:
        MOVLW   CODE_ADDR_UPPER     ; Load NVMADR with the base
        MOVWF   NVMADRU             ; address of the memory sector
        MOVLW   CODE_ADDR_HIGH
        MOVWF   NVMADRH
        MOVLW   CODE_ADDR_LOW
        MOVWF   NVMADRL
        BCF     INTCON, GIE         ; Disable interrupts
        BSF     NVMCON0, NVMEN      ; Enable NVM
; ----- Required Sequence -----
        MOVLW   0BBh
```

```
        MOVWF   NVMCON2             ; First unlock byte = 0BBh
        MOVLW   44h
        MOVWF   NVMCON2             ; Second unlock byte = 44h
        BSF     NVMCON1, SECRD      ; Start sector read (CPU stall)
; ------------------------------
        BTFSC   NVMCON0, NVMERR     ; Verify no error occurred during read
        BRA     NVM_RDERR           ; Return read error code
ERASE_BLOCK:                        ; NVMADR is already pointing to target
block
; ----- Required Sequence -----
        MOVLW   0CCh
        MOVWF   NVMCON2             ; First unlock byte = 0CCh
        MOVLW   33h
        MOVWF   NVMCON2             ; Second unlock byte = 33h
        BSF     NVMCON1, SECER      ; start sector erase (CPU stall)
; ------------------------------
        BTFSC   NVMCON0, NVMERR     ; Verify no error occurred during erase
        BRA     NVM_ERERR           ; Return erase error code
MODIFY_WORD:
        MOVLW   TARGET_ADDR_UPPER   ; Load TBLPTR with the target address
        MOVWF   TBLPTRU             ; of the LSByte
        MOVLW   TARGET_ADDR_HIGH
        MOVWF   TBLPTRH
        MOVLW   TARGET_ADDR_LOW
        MOVWF   TBLPTRL
        MOVLW   NEW_DATA_LOW        ; Update holding register
        MOVWF   TABLAT
        TBLWT*+
        MOVLW   NEW_DATA_HIGH
        MOVWF   TABLAT
        TBLWT*+
PROGRAM_MEMORY:                     ; NVMADR is already pointing to target
block
; ----- Required Sequence -----
        MOVLW   0DDh
        MOVWF   NVMCON2             ; First unlock byte = 0DDh
        MOVLW   22h
        MOVWF   NVMCON2             ; Second unlock byte = 22h
        BSF     NVMCON1, SECWR      ; Start sector programming (CPU stall)
; ------------------------------
        BTFSC   NVMCON0, NVMERR     ; Verify no error occurred during write
        BRA     NVM_WRERR           ; Return sector write error code
        CLRF    WREG,F              ; Return with no error
        BRA     NVM_EXIT
NVM_RDERR:
        MOVLW   01h
        BRA     NVM_EXIT
NVM_ERERR:
        MOVLW   02h
        BRA     NVM_EXIT
NVM_WRERR:
        MOVLW   03h
NVM_EXIT:
        BCF    NVMCON0, NVMEN       ; Disable NVM
        BSF    INTCON, GIE          ; Re-enable interrupts
        RETURN
```

14.1.4.2. PFM 字写序列

编程内部程序存储单元已擦除单字的事件序列如下：

1. 使用目标字地址设置 NVMADR。
2. 用所需字装载 NVMDAT。
3. 禁止中断。
4. 执行字/字节写解锁序列（见 NVM 解锁序列）。
5. 作为解锁序列的最后一步，将 WR 位置 1。
6. 在写操作期间 CPU 将暂停（约 50 µs，使用内部定时器）。
7. 重新允许中断。

8. 校验存储器（字读）。

例 14-5. 写入程序闪存的字

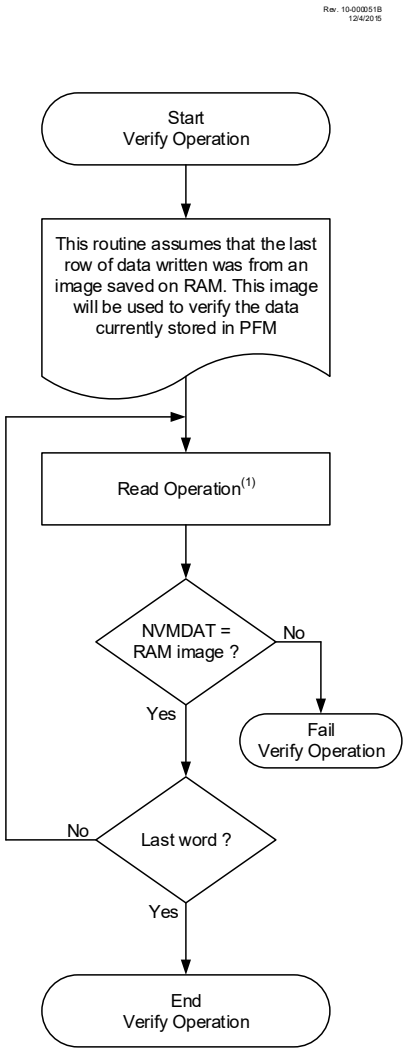
```
; Code sequence to program one erased word of PFM
; Target address is in WORD_ADDR_UPPER:WORD_ADDR_HIGH:WORD_ADDR_LOW
; Target data is in WORD_HIGH_BYTE:WORD_LOW_BYTE
; Calling routine should check WREG for the following errors:
;
;    00h = Successful modification
;    03h = Write error
;
WORD_WRITE:
        MOVF    WORD_ADDR_UPPER, W  ; load NVMADR with the target
        MOVWF   NVMADRU             ; address of the word
        MOVF    WORD_ADDR_HIGH, W
        MOVWF   NVMADRH
        MOVF    WORD_ADDR_LOW, W
        MOVWF   NVMADRL
        MOVF    WORD_HIGH_BYTE, W   ; load NVMDAT with desired
        MOVWF   NVMDATH             ; word
        MOVF    WORD_LOW_BYTE, W
        MOVWF   NVMDATL
        BCF     INTCON, GIE         ; disable interrupts
        BSF     NVMCON0, NVMEN      ; enable NVM
PROGRAM_WORD:
; ----- Required Sequence -----
        MOVLW   055h
        MOVWF   NVMCON2             ; first unlock byte = 055h
        MOVLW   0AAh
        MOVWF   NVMCON2             ; second unlock byte = 0AAh
        BSF     NVMCON1, WR         ; start word programming (CPU stall)
; -----------------------------
        BTFSC   NVMCON0, NVMERR     ; Verify no error occurred during write
        BRA     NVM_WRERR           ; return sector write error code
VERIFY_WORD:
        BSF     NVMCON0, RD         ; Retrieve word from PFM
        MOVF    WORD_HIGH_BYTE, W   ; Verify high byte
        CPFSEQ  NVMDATH
        BRA     NVM_RDERR           ; not equal - return error code
        MOVF    WORD_LOW_BYTE, W    ; Verify low byte
        CPFSEQ  NVMDATL
        BRA     NVM_RDERR           ; not equal - return error code

        CLRF    WREG,F              ; return with no error
        BRA     NVM_EXIT
NVM_RDERR:
        MOVLW   01h
        BRA     NVM_EXIT
NVM_WRERR:
        MOVLW   03h
NVM_EXIT:
        BCF    NVMCON0, NVMEN       ; disable NVM
        BSF    INTCON, GIE          ; re-enable interrupts
        RETURN
```

14.1.4.3. 写校验

根据具体应用，好的编程做法可能要求校验要写入存储器的值与初始值是否一致。应将其用于过多写操作会迫使位接近规范限值的应用。由于程序存储器以整页形式存储，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扇区 RAM 中存储的预期数据进行比较。

图 14-10. 程序闪存校验操作流程图



14.1.4.4. 写操作意外终止

如果写操作因意外事件（例如断电或意外复位）终止，则应验证刚编程的存储单元并在需要时重新编程。如果在正常工作期间写操作因 $\overline{\text{MCLR}}$ 复位或 WDT 超时复位而中断，则 NVMERR 位将置 1，用户可对此进行检查以确定是否需要重写存储单元。

14.1.4.5. 防止误写操作的保护措施

只有满足以下两个条件时，写序列才有效。这可防止可能导致数据损坏的误写操作。

1. WR、RD、SECWR、SECRD 和 SECER 位通过 NVMEN 位选通。除存储器读写期间外，建议始终将 NVMEN 位清零。这可以在意外将任意控制位置 1 的情况下，防止存储器操作。
2. 除了 RD 操作之外，每次都必须执行 NVM 解锁序列。

14.2. 用户 ID、器件 ID 和配置字访问

NVMADR 值确定访问的 NVM 地址空间。用户 ID 和配置字区域允许读写，而器件和版本 ID 允许只读（见 NVM 构成表）。

读写用户 ID 空间与写入 PFM 空间相同，如前面的段落所述。

只能通过 SECWR 操作写入配置空间。不能通过 SECER 操作对配置空间执行扇区擦除操作。对配置空间执行 SECWR 操作时，在扇区写入之前自动执行扇区擦除。除非由新值使能，否则在扇区写操作后，未使能的任何代码保护设置都将保持未使能状态。但是，无法通过自写配置空间来禁止任何已使能的代码保护设置。用户可通过以下步骤修改配置空间：

1. 使用解锁和 SECRD 序列读取配置空间。
2. 使用 TBLPTR 地址、TABLAT 数据和 `TBLWT*`指令修改所需配置字保持寄存器。
3. 使用解锁和 SECWR 序列将保持寄存器写入配置空间。

14.3. 数据闪存（DFM）

数据闪存为非易失性存储器阵列，也称为 EEPROM。DFM 与程序闪存分开，后者用于长期存储程序数据。DFM 映射到程序存储空间之上，并通过 NVM 特殊功能寄存器（SFR）间接寻址。在整个 V_{DD} 范围内的正常工作期间，DFM 都是可读写的。

五个 SFR 用于读取和写入数据 DFM。具体包括：

- NVMCON0
- NVMCON1
- NVMCON2
- NVMDAT
- NVMADR

一次只能对 DFM 读写一个字节。与数据存储块接口时，NVMDATL 存放要读/写的 8 位数据，而 NVMADR 寄存器存放正在访问的 DFM 存储单元的地址。

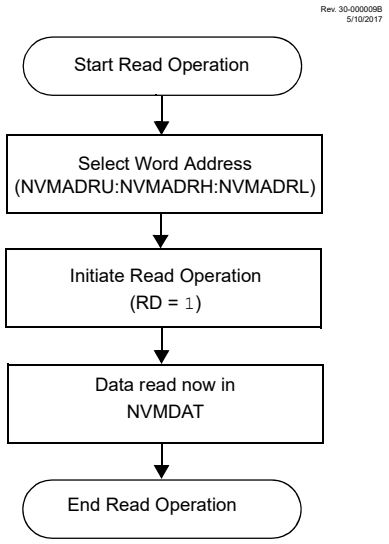
DFM 可耐受极高的擦/写次数。字节写操作会自动擦除存储单元并写入新数据（在写入前擦除）。写时间由内部编程定时器控制；它随电压和温度而变化，并因芯片而异。有关限值，请参见电气规范部分中的数据 EEPROM 存储器参数。

14.3.1. 读取 DFM

要读取 DFM 存储单元，用户必须将地址写入 NVMADR 寄存器，然后将 RD 控制位置 1。在紧接着的下一个指令周期，数据即可使用；因此，可通过随后的指令读取 NVMDAT 寄存器。NVMDAT 会将此值保存至下一次读操作，或保存至用户写入数据时（写操作）为止。

基本过程如以下流程图所示。

图 14-11. DFM 读取流程图



例 14-6. DFM 读取

```
; Data Flash Memory Address to read
        MOVF    DFM_ADDRL, W            ;
        MOVWF   NVMADRL                 ; Setup Address low byte
        MOVF    DFM_ADDRH, W            ;
        MOVWF   NVMADRH                 ; Setup Address high byte
        MOVF    DFM_ADDRU, W            ;
        MOVWF   NVMADRU                 ; Setup Address upper byte
        BSF     NVMCON1, RD             ; Issue EE Read
        MOVF    NVMDAT, W               ; W = EE_DATA
```

DFM 仅支持字节读取。不支持通过 SECRD 操作读取 DFM 块。

14.3.2. 写入 DFM

要写入 DFM 存储单元，必须先将地址写入 NVMADR 寄存器，再将数据写入 NVMDATL 寄存器。必须遵循 NVM 解锁序列中所示的序列来启动写周期。DFM 不支持块写操作（也称为扇区写操作）。

如果每个字节未严格遵循 NVM 解锁序列，则写操作不会开始。强烈建议在此代码段期间禁止中断。

此外，必须将 NVMEN 位置 1 才能使能 NVM 控制。此机制可防止因意外代码执行（即，失控程序）而对 DFM 进行意外的写操作。除更新 DFM 期间外，NVMEN 位应始终保持清零。NVMEN 位不由硬件清零。

启动写序列后，无法修改 NVMCON1、NVMADR 和 NVMDAT。除非 NVMEN 位置 1，否则将禁止 WR 位置 1。

启动写序列后，清零 NVMEN 位不会影响此写周期。写入单个 DFM 字节，此操作包括该字的隐式擦除周期。CPU 继续与之并行执行，写周期完成时，WR 位通过硬件清零，NVM 中断标志位（NVMIF）置 1。用户可允许此中断或轮询此位。必须用软件将 NVMIF 清零。

例 14-7. DFM 写入

```
; Data Flash Memory Address to write
        MOVF    DFM_ADDRL, W        ;
        MOVWF   NVMADRL             ; Setup Address low byte
        MOVF    DFM_ADDRH, W        ;
        MOVWF   NVMADRH             ; Setup Address high byte
        MOVF    DFM_ADDRU, W        ;
```

```
        MOVWF   NVMADRU             ; Setup Address upper byte
; Data Memory Value to write
        MOVF    DMF_DATA, W         ;
        MOVWF   NVMDATL             ;
; Enable writes
        BSF     NVMCON0, NVMEN      ; Enable NVM
; Disable interrupts
        BCF     INTCON, GIE         ;
;  -------- Required Sequence  --------
        MOVLW   55h                 ;
        MOVWF   NVMCON2             ;
        MOVLW   AAh                 ;
        MOVWF   NVMCON2             ;
        BSF     NVMCON1, WR         ; Set WR bit to begin write
;  -----------------------------------
; Wait for write to complete
        BTFSC   NVMCON1, WR         ; DFM writes do not stall the CPU
        BRA     $-2
; Enable INT
        BSF     INTCON, GIE         ;
; Disable writes
        BCF     NVMCON0, NVMEN      ;
```

14.3.3. DFM 写校验

根据具体应用，好的编程做法可能要求校验要写入存储器的值与初始值是否一致。应将其用于过多写操作会迫使位接近规范限值的应用。

14.3.4. 代码保护和写保护期间的操作

DFM 在配置字中有自己的代码保护位。如果使能了代码保护，将禁止在线串行编程读写操作。但是，内部读取正常工作。如果未使能写保护，则内部写入正常工作。

如果 DFM 受写保护，或 NVMADR 指向无效地址存储单元，则 WR 位清零且无任何影响。在这种情况下会发出 NVMERR 信号。

14.3.5. 防止误写操作的保护措施

在有些情况下，用户可能不希望写入 DFM。为防止误写 DFM，实现了各种机制。任何复位都将清零 NVMEN 位。此外，在上电延时定时器周期（T_{PWRT}）内会阻止对 DFM 进行写操作。

在欠压、电源故障或软件故障期间，解锁序列以及 NVMEN 位有助于防止意外写操作的发生。

14.3.6. 擦除 DFM

可通过将 0xFF 写入存储器中需要擦除的所有存储单元来擦除 DFM。

例 14-8. DFM 擦除程序

```
        CLRF    NVMADRL             ; Clear address low byte
        CLRF    NVMADRH             ; Clear address high byte
        MOVLW   31h                 ; EEPROM upper address
        MOVWF   NVMADRU             ; Set address upper byte
        SETF    NVMDAT              ; Load 0xFF to data register
        BCF     INTCON, GIE         ; Disable interrupts
        BSF     NVMCON0, NVMEN      ; Enable NVM
Loop:                               ; Loop to refresh array
        MOVLW   0x55                ; Initiate unlock sequence
        MOVWF   NVMCON2             ;
        MOVLW   0xAA                ;
        MOVWF   NVMCON2             ;
        BSF     NVMCON1, WR         ; Set WR bit to begin write
        BTFSC   NVMCON1, WR         ; Wait for write to complete
        BRA     $-2
        INCFSZ  NVMADRL, F          ; Increment address low byte
        BRA     Loop                ; Not zero, do it again
; The following 4 lines of code are not needed if EEPROM is 256 bytes or less
        INCF    NVMADRH, F          ; Decrement address high byte
        MOVLW   0x03                ; Move 0x03 to working register
```

```
        CPFSGT  NVMADRH             ; Compare address high byte with working
register
        BRA     Loop                ; Skip if greater than working register
                                    ; Else go back to erase loop
        BCF     NVMCON0, NVMEN      ; Disable NVM
        BSF     INTCON, GIE         ; Enable interrupts
```

14.4. 寄存器定义：非易失性存储器

14.4.1. NVMCON0

名称： NVMCON0
偏移量： 0xF7F

非易失性存储器控制 0 寄存器

位	7	6	5	4	3	2	1	0
	NVMEN			NVMERR	保留			
访问	R/W			R/W/HS	R/W			
复位	0			x	0			

Bit 7 – NVMEN NVM 使能位

值	说明
1	对于所有操作，都使能 NVM
0	除单字节或字读取外，对于所有操作都禁止 NVM

Bit 4 – NVMERR
NVM 错误标志位

值	说明
1	尝试的 NVM 写入或扇区读取未成功完成。必须用软件清零。
0	所有 NVM 操作均成功完成

Bit 3 – 保留
保留，不要使用。该位必须保持为 0。

14.4.2. NVMCON1

名称： NVMCON1
偏移量： 0xF80

非易失性存储器控制 1 寄存器

位	7	6	5	4	3	2	1	0
		SECER	SECWR	WR			SECRD	RD
访问		R/S/HC	R/S/HC	R/S/HC			R/S/HC	R/S/HC
复位		0	0	0			0	0

Bit 6 – SECER
NVM 扇区擦除使能控制位

值	条件	说明
1	NVMADR 指向 PFM	在扇区擦除解锁序列之后，立即执行扇区擦除操作。保持置 1，直到操作完成。不能用软件清零。
0	NVMADR 指向 PFM	NVM 扇区擦除操作已完成并且变为无效

Bit 5 – SECWR
NVM 扇区写入使能控制位

值	条件	说明
1	NVMADR 指向 PFM 或 CONFIG	在扇区写入解锁序列之后，立即启动 PFM 扇区写操作。保持置 1，直到写操作完成。不能用软件清零。
0	NVMADR 指向 PFM 或 CONFIG	NVM 扇区写操作已完成并且变为无效

Bit 4 – WR
NVM 写控制位

值	条件	说明
1	NVMADR 指向 DFM	在 DFM 写解锁序列之后，立即使用 NVMDATL 寄存器中的数据启动 DFM 字节擦除/写序列。保持置 1，直到操作完成。不能用软件清零。
1	NVMADR 指向 PFM	在 PFM 写解锁序列之后，立即使用 NVMDATH:L 寄存器对中的数据启动 NVM 字写序列。保持置 1，直到操作完成。不能用软件清零。
0	NVMADR 指向 PFM 或 DFM	NVM 字节/字写操作已完成并且变为无效

Bit 1 – SECRD
PFM 扇区读取使能控制位

值	条件	说明
1	NVMADR 指向 PFM 或 CONFIG	在扇区读取解锁序列之后，立即从 PFM 将一个完整扇区读到扇区 RAM 中。保持置 1，直到读操作完成。不能用软件清零。
0	NVMADR 指向 PFM 或 CONFIG	PFM 扇区读操作已完成并且变为无效

Bit 0 – RD
NVM 读取使能控制位

值	条件	说明
1	NVMADR 指向 DFM	启动从 DFM 将一个字节读到 NVMDATL 寄存器的操作。保持置 1，直到读操作完成。不能用软件清零。
1	NVMADR 指向 PFM	启动从 PFM 将一个字读到 NVMDATH:L 寄存器的操作。保持置 1，直到读操作完成。不能用软件清零。

值	条件	说明
0	NVMADR 指向 PFM 或 DFM	NVM 读操作已完成并且变为无效。

14.4.3. NVMCON2

名称： NVMCON2
偏移量： 0xF81

非易失性存储器控制 2 寄存器
注：无论写入何种数据，该寄存器始终读为零。
请参见 NVM 解锁序列部分

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]

14.4.4. NVMDAT

名称： NVMDAT
偏移量： 0xF7C

NVM 数据寄存器

位	15	14	13	12	11	10	9	8
	NVMDATH[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

位	7	6	5	4	3	2	1	0
	NVMDATL[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 15:8 – NVMDATH[7:0]
写入到 PFM 和从中读取的数据的 NVMDAT 最高有效字节。

Bit 7:0 – NVMDATL[7:0]
写入到 PFM 或 DFM 和从中读取的数据的 NVMDAT 最低有效字节。

14.4.5. NVMADR

名称： NVMADR
偏移量： 0xF79

NVM 地址寄存器

位	23	22	21	20	19	18	17	16
			NVMADRU[5:0]					
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			0	0	0	0	0	0

位	15	14	13	12	11	10	9	8
	NVMADRH[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

位	7	6	5	4	3	2	1	0
	NVMADRL[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 21:16 – NVMADRU[5:0]
NVM 地址位[21:16]。

Bit 15:8 – NVMADRH[7:0]
NVM 地址的高字节。

Bit 7:0 – NVMADRL[7:0]
NVM 地址的低字节。

14.4.6. TABLAT

名称： TABLAT
偏移量： 0xFF5

程序、配置、器件 ID 和用户 ID 存储器数据

位	7	6	5	4	3	2	1	0
	TABLAT[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 – TABLAT[7:0] 在 TBLRD 命令之后从 TBLPTR 中包含的地址返回的 NVM 存储器字节的值，或通过 TBLWT 命令写入锁存器的数据。

14.4.7. TBLPTR

名称： TBLPTR
偏移量： 0xFF6

程序、配置、器件 ID 和用户 ID 存储器地址

位	23	22	21	20	19	18	17	16
			TBLPTR21	TBLPTRU[4:0]				
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			0	0	0	0	0	0

位	15	14	13	12	11	10	9	8
	TBLPTRH[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

位	7	6	5	4	3	2	1	0
	TBLPTRL[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 21 – TBLPTR21 NVM 最高有效地址位

值	说明
1	访问配置、用户 ID、器件 ID 和版本 ID 空间
0	访问程序闪存空间

Bit 20:16 – TBLPTRU[4:0] NVM 最高地址位

Bit 15:8 – TBLPTRH[7:0] NVM 地址位的高字节

Bit 7:0 – TBLPTRL[7:0] NVM 地址位的低字节

14.5. 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 ... 0x0F78	保留									
0x0F79	NVMADR	7:0	NVMADRL[7:0]							
		15:8	NVMADRH[7:0]							
		23:16			NVMADRU[5:0]					
0x0F7C	NVMDAT	7:0	NVMDATL[7:0]							
		15:8	NVMDATH[7:0]							
0x0F7E	保留									
0x0F7F	NVMCON0	7:0	NVMEN			NVMERR	保留			
0x0F80	NVMCON1	7:0		SECER	SECWR	WR			SECRD	RD
0x0F81	NVMCON2	7:0	NVMCON2[7:0]							
0x0F82 ... 0x0FF4	保留									
0x0FF5	TABLAT	7:0	TABLAT[7:0]							
0x0FF6	TBLPTR	7:0	TBLPTRL[7:0]							
		15:8	TBLPTRH[7:0]							
		23:16			TBLPTR21	TBLPTRU[4:0]				

15. 8x8 硬件乘法器

15.1. 简介

所有器件均包含一个 8x8 硬件乘法器（作为 ALU 的一部分）。该乘法器可执行无符号运算并产生一个 16 位运算结果，该结果存储在一对乘积寄存器 PRODH:PRODL 中。该乘法器执行的运算不会影响 STATUS 寄存器中的任何标志。

通过硬件执行乘法运算只需一个指令周期。硬件乘法器具有更高的计算吞吐量并缩短了乘法算法的代码长度，因此可在许多先前只能使用数字信号处理器的应用中使用此系列器件。表 15-1 给出了各种硬件和软件乘法运算的比较，包括所需存储空间和执行时间。

15.2. 工作原理

例 15-1 给出了 8x8 无符号乘法运算的指令序列。如果已将其中一个参数装入 WREG 寄存器，则实现该运算仅需一条指令。

例 15-2 给出了 8x8 有符号乘法运算的指令序列。要确定参数的符号位，需测试每个参数的最高有效位（MSb）并进行相应的减法。

例 15-1. 8x8 无符号乘法程序

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

例 15-2. 8x8 有符号乘法程序

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

表 15-1. 各种乘法运算的性能比较

程序	乘法实现方法	程序存储器（字）	周期（最大值）	时间			
				64 MHz 时	40 MHz 时	10 MHz 时	4 MHz 时
8x8 无符号	软件乘法	13	69	4.3 µs	6.9 µs	27.6 µs	69 µs
	硬件乘法	1	1	62.5 ns	100 ns	400 ns	1 µs
8x8 有符号	软件乘法	33	91	5.7 µs	9.1 µs	36.4 µs	91 µs
	硬件乘法	6	6	375 ns	600 ns	2.4 µs	6 µs
16x16 无符号	软件乘法	21	242	15.1 µs	24.2 µs	96.8 µs	242 µs
	硬件乘法	28	28	1.8 µs	2.8 µs	11.2 µs	28 µs
16x16 有符号	软件乘法	52	254	15.9 µs	25.4 µs	102.6 µs	254 µs
	硬件乘法	35	40	2.5 µs	4.0 µs	16.0 µs	40 µs

例 15-3 给出了 16 x 16 无符号乘法运算的指令序列。以下公式给出了使用的算法。32 位结果存储在四个寄存器（RES[3:0]）中。

16 x 16 无符号乘法算法

$RES3:RES0 = ARG1H:ARG1L \bullet ARG2H:ARG2L = \left(ARG1H \bullet ARG2H \bullet 2^{16}\right) + \left(ARG1H \bullet ARG2L \bullet 2^{8}\right) + \left(ARG1L \bullet ARG2H \bullet 2^{8}\right) + \left(ARG1L \bullet ARG2L\right)$

例 15-3. 16 x 16 无符号乘法程序

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

例 15-4 给出了 16 x 16 有符号乘法运算的指令序列。以下公式给出了使用的算法。32 位结果存储在四个寄存器（RES[3:0]）中。要确定参数的符号位，需测试每个参数对的 MSb 并进行相应的减法。

16 x 16 有符号乘法算法

$RES3:RES0 = ARG1H:ARG1L \bullet ARG2H:ARG2L = \left(ARG1H \bullet ARG2H \bullet 2^{16}\right) + \left(ARG1H \bullet ARG2L \bullet 2^{8}\right) + \left(ARG1L \bullet ARG2H \bullet 2^{8}\right) + \left(ARG1L \bullet ARG2L\right) + \left(-1 \bullet ARG2H\left[7\right] \bullet ARG1H:ARG1L \bullet 2^{16}\right) + \left(-1 \bullet ARG1H\left[7\right] \bullet ARG2H:ARG2L \bullet 2^{16}\right)$

例 15-4. 16 x 16 有符号乘法程序

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

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

SIGN_ARG1:
        BTFSS   ARG1H, 7        ; ARG1H:ARG1L neg?
        BRA     CONT_CODE       ; no, done
        MOVF    ARG2L, W        ;
        SUBWF   RES2            ;
        MOVF    ARG2H, W        ;
        SUBWFB  RES3
;
CONT_CODE:
          :
```

15.3. 寄存器定义：8x8 硬件乘法器

15.3.1. PROD

名称： PROD
偏移量： 0xFF3

乘积寄存器对

PROD 寄存器存储由 8x8 硬件乘法器执行的无符号运算产生的 16 位结果。

位	15	14	13	12	11	10	9	8
	PRODH[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

位	7	6	5	4	3	2	1	0
	PRODL[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 15:8 – PRODH[7:0]
PROD 最高有效位。

Bit 7:0 – PRODL[7:0]
PROD 最低有效位。

15.4. 寄存器汇总——8x8 硬件乘法器

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0FF2	保留									
0x0FF3	PROD	7:0	PRODL[7:0]							
		15:8	PRODH[7:0]							

16. CRC——带存储器扫描器的循环冗余校验模块

循环冗余校验（CRC）模块提供了一个可用软件配置的硬件实现 CRC 校验和生成器。该模块具有以下特性：

- 可使用最高 16 位的任何标准 CRC
- 可配置多项式
- 可使用最高 16 位的任何种子值
- 支持标准和反向位序
- 可自动或由用户添加扩充零
- 存储器扫描器，用于对程序存储器用户数据进行快速 CRC 计算
- 用于通信 CRC 的软件可装入数据寄存器

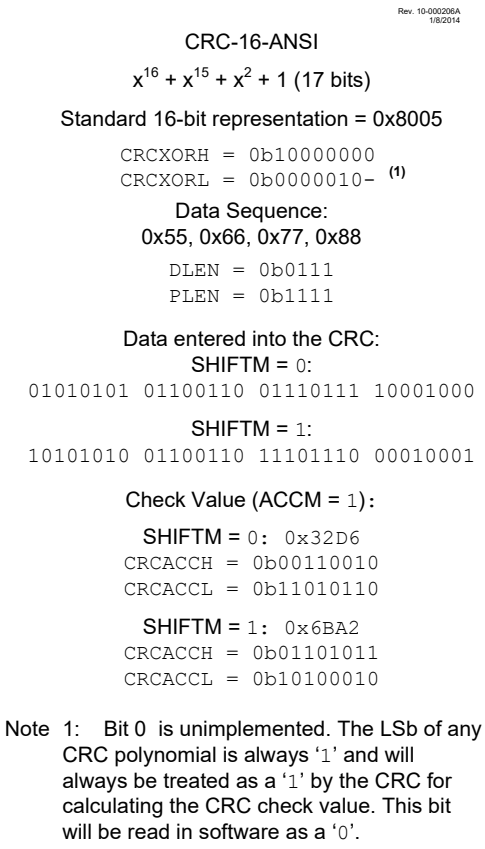
16.1. CRC 模块概述

CRC 模块提供了一种计算程序存储器的校验值的方式。CRC 模块与存储器扫描器配合使用，可实现更快的 CRC 计算。存储器扫描器可自动向 CRC 模块提供数据。此外，也可以不使用扫描器，通过直接向 SFR 写入数据来操作 CRC 模块。

16.2. CRC 功能概述

CRC 模块可用于使用内置存储器扫描器或通过用户输入 RAM 存储器来检测闪存中的位错误。CRC 模块可接受最高 16 位的多项式与最高 16 位的种子值。然后，将在 CRCACC 寄存器中生成 CRC 计算的校验值（或校验和），供用户存储。CRC 模块使用异或移位寄存器实现来执行 CRC 计算所需的多项式除法。

图 16-1. CRC 示例

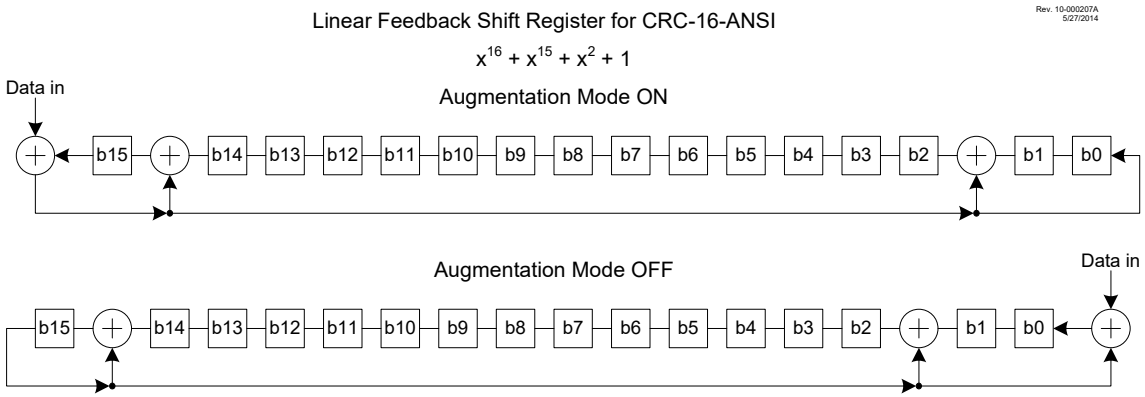


16.3. CRC 多项式实现

可以使用任意多项式。多项式和累加器长度由 PLEN 位确定。对于 n 位累加器，PLEN = n-1，相应的多项式为 n+1 位。因此，累加器可以是最高 16 位的任意大小，相应的多项式最高为 17 位。多项式的 MSb 和 LSb 始终为 1（由硬件强制设定）。但是，CRCXORL 寄存器的 LSb 未实现，始终读为 0。

MSb 和 LSb 之间的所有多项式位均由 CRCXOR 寄存器指定。例如，使用 CRC16-ANSI 时，多项式定义为 $X^{16}+X^{15}+X^2+1$。X^{16} 和 X^0 = 1 项是由硬件控制的 MSb 和 LSb。X^{15} 和 X^2 项通过使用值 0x8004 设置相应的 CRCXOR[15:0]位来指定。实际值为 0x8005，因为硬件会将 LSb 置 1。请参见图 16-1。

图 16-2. CRC LFSR 示例



16.4. CRC 数据源

可通过以下两种方式将数据输入 CRC 模块：

- 对于用户数据，使用 CRCDAT 寄存器
- 对于闪存中的数据，使用程序存储器扫描器

使用 DLEN 位指定每个字的数据位数（最高 16 位）。该模块仅使用由 DLEN 指定的 CRCDATA 寄存器中的数据位数，并忽略 CRCDATA 寄存器中的其他数据位。

数据会被移入 CRCSHIFT，作为一种中间结果来计算 CRCACC 寄存器中的校验值。

SHIFTM 位用于确定移入累加器的数据的位序。如果 SHIFTM 未置 1，则先移入数据的 MSb（大尾数法）。DLEN 的值将决定 MSb。如果 SHIFTM 位置 1，则将按反序（LSb 在前）将数据移入累加器（小尾数法）。

通过在开始 CRC 之前将 CRCACC 寄存器设置为适当的值，可以设置 CRC 模块的初始种子值。

16.4.1. 用户数据的 CRC

要对用户的数据输入使用 CRC 模块，用户必须将数据写入 CRCDAT 寄存器。在对 CRCDATL 寄存器进行任何写操作时，CRCDAT 寄存器中的数据将被锁存到移位寄存器中。

16.4.2. 闪存的 CRC

要对闪存中的数据使用 CRC 模块，用户可按照程序存储器扫描配置部分中的定义初始化程序存储器扫描器。

16.5. CRC 校验值

在 CRC 计算完毕后，CRC 校验值将位于 CRCACC 寄存器中。校验值将取决于 ACCM 和 SHIFTM 模式设置。

如果 ACCM 位置 1，CRC 模块会使用一定数量（等于多项式长度）的零来扩充数据，以确定最终的校验值。如果 ACCM 位未置 1，CRC 将在数据结束处停止。可以向 CRCDAT 输入一定数量（等于多项式长度）的零，以确定与扩充模式相同的校验值。或者，也可以在此时输入预期的校验值，使最终结果等于 0。

在 SHIFTM 位置 1、选择先移入 LSb 且 ACCM 位置 1 的情况下计算得到 CRC 校验值时，CRCACC 寄存器中的最终值会变为反序，使 LSb 处于 MSb 位置，反之亦然。它是以反向位序形式表示的预期校验值。如果要生成追加到数据流后的校验值，则必须对最终值执行倒转位序的操作，以获得正确的校验和。可以使用 CRC 通过以下方法来执行这种倒转：

1. 将 CRCACC 值保存在用户 RAM 空间中。
2. 清零 CRCACC 寄存器。
3. 清零 CRCXOR 寄存器。
4. 将保存的 CRCACC 值写入 CRCDAT 输入。

方向正确的校验值将处于 CRCACC 寄存器中，作为结果。

16.6. CRC 中断

BUSY 位从 1 跳变为 0 时，CRC 将会产生中断。每次 BUSY 位发生跳变时，无论是否允许 CRC 中断，PIRx 寄存器的 CRCIF 中断标志位都会置 1。CRCIF 位只能用软件清零。CRC 中断允许位是 PIEx 寄存器的 CRCIE 位。

16.7. 配置 CRC

以下步骤说明了如何正确配置 CRC。

1. 确定是使用扫描器进行自动程序存储器扫描还是通过 SFR 接口进行手动计算，并根据所作的决策执行 CRC 数据源中指定的操作。
2. 如果需要，在 CRCACC 寄存器中设置起始 CRC 值作为种子。
3. 使用所需的发生器多项式编程 CRCXOR 寄存器。
4. 使用数据字长度 - 1 来编程 DLEN 位（见图 16-1）。这将决定对于每个数据字，移位器将移入累加器的次数。
5. 使用多项式长度-2 来编程 PLEN 位（见图 16-1）。
6. 确定是否需要移入追加零，并相应地设置 ACCM 位。
7. 类似地，确定是必须先移位 MSb 还是 LSb，并相应地写入 SHIFTM 位。
8. 将 GO 位置 1，开始移位过程。
9. 如果使用手动 SFR 输入，则监视 FULL 位。当 FULL = 0 时，可以向 CRCDAT 寄存器写入另一个数据字；请记住，如果数据长度大于 8 位，则必须先写入最高有效字节 CRCDATH，因为移位器会在写入 CRCDATL 寄存器时开始工作。
10. 如果使用扫描器，则只要 SCANGO 位置 1，扫描器就会根据需要自动在 CRCDAT 寄存器中填入数据字。
11. 如果使用闪存扫描器，则通过监视 PIRx SCANIF 位（或 SCANGO 位）来确定扫描器是否完成将信息压入 CRCDATA 寄存器。在扫描器完成之后，通过监视 BUSY 位来确定 CRC 是否已完成，之后可以从 CRCACC 寄存器中读取校验值。如果两个中断标志均置 1（或 BUSY 和 SCANGO 位均清零），则可以从 CRCACC 寄存器中读取完成的 CRC 计算结果。
12. 如果使用手动输入，则通过监视 BUSY 位来确定 CRCACC 寄存器何时保存有效校验值。

16.8. 程序存储器扫描配置

可将程序存储器扫描模块与 CRC 模块结合使用，从而在程序存储器的某个地址范围内执行 CRC 计算。为了设置扫描器，使之与 CRC 配合工作，需要执行以下步骤：

1. 将 EN 和 SCANEN 位置 1。如果它们被禁止，扫描器和 CRC 的所有内部状态都会复位。但是，CRC SFR 寄存器不受影响。
2. 选择要使用的存储器访问模式（见扫描模式）并相应地设置 MODE 位。
3. 根据存储器访问模式，将 INTM 位设置为相应的中断模式（见中断交互）。
4. 基于存储器中要扫描的起始位置和结束位置设置 SCANLADR 和 SCANHADR 寄存器。
5. 先将 GO 位置 1，然后再将 SCANGO 位置 1。通过将 SCANGO 位置 1 来启动扫描。要使用扫描器，EN 和 GO 位必须都使能。如果其中任一位被禁止，则扫描将中止，并且 INVALID 位会置 1。扫描器将

等待 CRC 指示它已准备好处理第一个闪存存储单元的信号，然后开始将数据装入 CRC。它将不断执行该操作，直到达到所配置的结束地址或器件上未实现的地址，此时 SCANGO 位会清零，扫描器功能将停止，并触发 SCANIF 中断。或者，必要时可用软件清零 SCANGO 位提前终止扫描。

16.9. 扫描器中断

SCANGO 位从 1 跳变为 0 时，扫描器将会触发中断。当达到最后一个存储单元，并且数据已输入 CRCDATA 寄存器时，PIRx 的 SCANIF 中断标志会置 1。SCANIF 位只能用软件清零。SCAN 中断允许位是 PIEx 寄存器的 SCANIE 位。

16.10. 扫描模式

存储器扫描器可以使用 4 种模式进行扫描：突发、窥探、并发和触发。这些模式由 MODE 位控制。表 16-1 总结了这 4 种模式。

16.10.1. 突发模式

当 MODE = 01 时，扫描器处于突发模式。在突发模式下，CPU 操作会在将 SCANGO 位置 1 的操作之后停顿，扫描会使用指令时钟开始执行。CPU 会保持当前状态，直到扫描停止为止。请注意，由于 CPU 不执行指令，无法用软件清零 SCANGO 位，因此 CPU 将一直保持停顿，直到出现其中一个硬件结束条件为止。扫描器的突发模式具有最高吞吐量，但代价是会停顿其他执行。

16.10.2. 并发模式

当 MODE = 00 时，扫描器处于并发模式。并发模式类似于突发模式，在执行存储器访问时会停顿 CPU。但是，突发模式会停顿到所有访问完成为止，而并发模式允许 CPU 在访问周期之间执行。

16.10.3. 触发模式

当 MODE = 11 时，扫描器处于触发模式。触发模式的行为与并发模式相同，只是它不是在 SCANGO 位置 1 时立即开始扫描，而是会等待来自一个独立触发源（由 SCANTRIG 寄存器决定）的上升沿。

16.10.4. 窥探模式

当 MODE = 10 时，扫描器处于窥探模式。窥探模式会等待 CPU 不需要访问 NVM 的指令周期（如跳转指令）并使用该周期执行自身的 NVM 访问。这会导致 NVM 访问的吞吐量最低（并且可能比其他模式花费更长的时间来完成扫描），但与其他模式不同的是，该模式不会对执行时间产生任何影响。

表 16-1. 扫描器模式汇总

MODE[1:0]		说明		
		首次扫描访问	CPU 操作	
11	触发	在触发后尽快执行	在 NVM 访问期间停顿	每次访问后 CPU 继续执行
10	窥探	在第一个死周期时执行	时序不受影响	每次访问后 CPU 继续执行
01	突发	尽快执行	在 NVM 访问期间停顿	CPU 被暂停，直到扫描完成为止
00	并发			每次访问后 CPU 继续执行

16.10.5. 中断交互

INTM 位控制扫描器根据 NVM 扫描器所处的模式而对中断作出的响应，如下表所述。

表 16-2. 扫描中断模式

INTM	MODE[1:0]		
	模式 == 突发	模式 == 并发或触发	模式 == 窥探
1	中断会通过改写 SCANGO（为零）来暂停突发操作，中断处理程序以全速执行；扫描器突发操作在中断完成时恢复。	扫描器在中断响应期间暂停（SCANGO = 0）；中断将以全速执行，扫描在中断完成时恢复。	忽略此位

表 16-2. 扫描中断模式 （续）

INTM	MODE[1:0]		
	模式 == 突发	模式 == 并发或触发	模式 == 窥探
0	中断不会改写 SCANGO，扫描（突发）操作将继续进行；中断响应会被延迟，直到扫描完成为止（中断延时将增大）。	扫描器在中断响应期间访问 NVM。	忽略此位

一般来说，如果 INTM = 0，则扫描器优先于中断，导致中断处理速度下降和/或中断响应延时增大。如果 INTM = 1，则中断优先并具有更高的速度，使存储器扫描发生延迟。

16.10.6. WWDT 交互

WWDT 的操作不受扫描器活动的影响。因此，长时间扫描（特别是在突发模式下）可能会超出 WWDT 超时周期，导致意外器件复位。对于还采用了 WWDT 的应用，执行存储器扫描时可能需要考虑这一点。

16.10.7. 在线调试（ICD）交互

在发生 ICD 暂停时，扫描器会冻结并一直保持冻结状态，直到恢复用户模式操作为止。调试器可以通过检查 SCANCON0 和 SCANLADR 寄存器来确定扫描的状态。

下表总结了 ICD 与每种工作模式的交互。

表 16-3. ICD 和扫描器的交互

ICD 暂停	扫描器工作模式		
	窥探	并发 触发	突发
外部暂停	如果扫描器将窥探未执行的指令（由于进入 ICD），窥探操作会在 ICD 退出后在指令执行时发生	如果外部暂停在扫描周期期间生效，则可能会也可能不会在进入 ICD 之前执行（因扫描而延迟）指令，具体取决于外部暂停时序	如果外部暂停在 BSF（SCANCON.GO）期间生效，则会进入 ICD，而突发操作会被延迟到退出 ICD 为止。 否则，将完成当前的 NVM 访问周期，然后中断扫描器来进入 ICD。
		如果外部暂停在紧接扫描周期之前的周期期间生效，则扫描和指令执行均在退出 ICD 后发生	如果外部暂停在突发操作期间生效，则突发操作会被暂停，并随着退出 ICD 而恢复
PC 断点		扫描周期在进入 ICD 之前发生，指令执行在退出 ICD 之后发生	如果 PC 断点（或单步执行）处于 BSF(SCANCON.GO) 上，则会在执行前进入 ICD；突发操作将在退出 ICD 时执行，并且突发操作将运行至完成为止。 请注意，突发操作会被外部暂停中断。
数据断点		执行具有数据断点的指令，之后立即进入 ICD。如果在该周期期间请求扫描，则扫描周期会被推迟到退出 ICD 为止。	
单步执行		如果扫描周期在执行调试指令后就绪，则扫描会读取 PFM，然后重新进入 ICD	
SWBP 和 ICDINST		如果扫描会使 SWBP 停顿，则会发生扫描周期并进入 ICD	如果 SWBP 替换了 BSF(SCANCON.GO)，则会进入 ICD；指令将在退出 ICD 时（通过 ICDINSTR 寄存器）执行，突发操作将运行至完成为止

16.10.8. 外设模块禁止

通过单独将 PMD0 寄存器的 CRCMD 和 SCANMD 位置 1，可分别禁止 CRC 和扫描器模块。仅当配置字 4 的 SCANE 位置 1 时，SCANMD 才能用于使能或禁止扫描器模块。如果 SCANE 位清零，则扫描器模块不可用，SCANMD 位将被忽略。

16.11. 寄存器定义：CRC 和扫描器控制

下表列出了 CRC 的长位名称前缀。更多信息，请参见**“长位名称”**一节。

表 16-4. CRC 长位名称前缀

外设	位名称前缀
CRC	CRC

16.11.1. CRCCON0

名称： CRCCON0
偏移量： 0xF77
复位： 0

CRC 控制寄存器 0

位	7	6	5	4	3	2	1	0
	EN	GO	BUSY	ACCM			SHIFTM	FULL
访问	R/W	R/W	RO	R/W			R/W	RO
复位	0	0	0	0			0	0

Bit 7 – EN CRC 使能位

值	说明
1	从复位状态释放 CRC 模块
0	禁止 CRC，不消耗工作电流

Bit 6 – GO CRC 启动位

值	说明
1	启动 CRC 串行移位器
0	关闭 CRC 串行移位器

Bit 5 – BUSY CRC 忙位

值	说明
1	正在进行或正在等待进行移位
0	移位器中的所有有效位都已移入累加器，EMPTY = 1

Bit 4 – ACCM 累加器模式位

值	说明
1	使用零扩充数据
0	不使用零扩充数据

Bit 1 – SHIFTM 移位模式位

值	说明
1	右移（LSb）
0	左移（MSb）

Bit 0 – FULL 数据路径满指示位

值	说明
1	CRCDATH/L 寄存器已满
0	CRCDATH/L 寄存器已将其数据移入移位器

16.11.2. CRCCON1

名称： CRCCON1
偏移量： 0xF78
复位： 0

CRC 控制寄存器 1

位	7	6	5	4	3	2	1	0
	DLEN[3:0]				PLEN[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 7:4 – DLEN[3:0] 数据长度位
表示数据字长度 -1（见图 16-1）

Bit 3:0 – PLEN[3:0] 多项式长度位
表示多项式长度 -1（见图 16-1）

16.11.3. CRCDAT

名称： CRCDAT
偏移量： 0xF6F

CRC 数据寄存器

位	15	14	13	12	11	10	9	8
	CRCDATH[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

位	7	6	5	4	3	2	1	0
	CRCDATL[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 15:8 – CRCDATH[7:0] CRC 输入/输出数据最高有效字节

Bit 7:0 – CRCDATL[7:0] CRC 输入/输出数据最低有效字节

16.11.4. CRCACC

名称： CRCACC
偏移量： 0xF71
复位： 0

CRC 累加器寄存器

位	15	14	13	12	11	10	9	8
	CRCACCH[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

位	7	6	5	4	3	2	1	0
	CRCACCL[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 15:8 – CRCACCH[7:0] CRC 累加器寄存器最高有效字节
写入该寄存器时将写入 CRC 累加器寄存器的最高有效字节。读取该寄存器时将读取 CRC 累加器的最高有效字节。

Bit 7:0 – CRCACCL[7:0] CRC 累加器寄存器最低有效字节
写入该寄存器时将写入 CRC 累加器寄存器的最低有效字节。读取该寄存器时将读取 CRC 累加器的最低有效字节。

16.11.5. CRCSHIFT

名称： CRCSHIFT
偏移量： 0xF73
复位： 0

CRC 移位寄存器

位	15	14	13	12	11	10	9	8
	CRCSHIFTH[7:0]							
访问	RO	RO	RO	RO	RO	RO	RO	RO
复位	0	0	0	0	0	0	0	0

位	7	6	5	4	3	2	1	0
	CRCSHIFTL[7:0]							
访问	RO	RO	RO	RO	RO	RO	RO	RO
复位	0	0	0	0	0	0	0	0

Bit 15:8 – CRCSHIFTH[7:0] CRC 移位器寄存器最高有效字节
读取该寄存器时将读取 CRC 移位器的最高有效字节。

Bit 7:0 – CRCSHIFTL[7:0] CRC 移位器寄存器最低有效字节
读取该寄存器时将读取 CRC 移位器的最低有效字节。

16.11.6. CRCXOR

名称： CRCXOR
偏移量： 0xF75

CRC XOR 寄存器

位	15	14	13	12	11	10	9	8
	CRCXORH[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

位	7	6	5	4	3	2	1	0
	CRCXORL[6:0]							CRCXORL0
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	U
复位	x	x	x	x	x	x	x	1

Bit 15:8 – CRCXORH[7:0] 多项式的项 XN 的异或使能最高有效字节

Bit 7:1 – CRCXORL[6:0] 多项式的项 XN 的异或使能最低有效字节

Bit 0 – CRCXORL0 未实现 LSb。读为 1

16.11.7. SCANCON0

名称： SCANCON0
偏移量： 0xF4A

扫描器访问控制寄存器 0

位	7	6	5	4	3	2	1	0
	SCANEN	SCANGO	BUSY	INVALID	INTM		MODE[1:0]	
访问	R/W	R/W/HC	R	R	R/W		R/W	R/W
复位	0	0	0	1	0		0	0

Bit 7 – SCANEN 扫描器使能位(1)

值	说明
1	使能扫描器
0	禁止扫描器，内部状态复位

Bit 6 – SCANGO 扫描器 GO 位(2, 3)

值	说明
1	当 CRC 发送就绪信号时，将根据 MDx 访问 NVM，并将数据传递到客户端外设。
0	不会执行扫描器操作

Bit 5 – BUSY 扫描器忙指示位(4)

值	说明
1	正在执行扫描器周期
0	扫描器周期已完成（或从未启动）

Bit 4 – INVALID 扫描器中止信号位

值	说明
1	SCANLADRL/H/U 已递增到无效地址(6)或扫描器尚未正确设置(7)
0	SCANLADRL/H/U 指向有效地址

Bit 3 – INTM NVM 扫描器中断管理模式选择位

值	条件	说明
X	MODE = 10	忽略此位
1	MODE = 01	CPU 停顿，直到传输完所有数据为止。在中断操作期间改写 SCANGO（为零）；从中断返回后恢复扫描器
0	MODE = 01	CPU 停顿，直到传输完所有数据为止。SCANGO 不受中断影响，中断响应将受影响
1	MODE = 00 或 11	在中断操作期间改写 SCANGO（为零）；从中断返回后恢复扫描操作
0	MODE = 00 或 11	中断不会阻止 NVM 访问

Bit 1:0 – MODE[1:0] 存储器访问模式位(5)

值	说明
11	触发模式
10	窥探模式
01	突发模式
00	并发模式

注：

1. 设置 SCANEN = 0（SCANCON0 寄存器）不会影响任何其他寄存器内容。
2. 该位在 LADR > HADR 时清零（并且不会发生数据周期）。
3. 如果 INTM = 1，则在中断响应期间会改写该位（为零，但不会被清除）。
4. 当对 NVM 进行访问时，或当 CRC 发送就绪信号时，BUSY = 1。
5. 有关详细信息，请参见表 16-1。
6. 如果在扫描整个 PFM 范围时 LADR 的值计满返回时，则会出现无效地址。如果扫描低地址寄存器中的值指向未映射到器件存储器映射中的位置，也会出现无效地址。
7. 在将 SCANGO 位置 1 之前，必须先将 CRCEN 和 CRCGO 位置 1。请参见程序存储器扫描配置。

16.11.8. SCANLADR

名称： SCANLADR
偏移量： 0xF44
复位： 0

扫描低地址寄存器

位	23	22	21	20	19	18	17	16
			SCANLADRU[5:0]					
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			0	0	0	0	0	0

位	15	14	13	12	11	10	9	8
	SCANLADRH[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

位	7	6	5	4	3	2	1	0
	SCANLADRL[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 21:16 – SCANLADRU[5:0] 扫描起始/当前地址最高字节
要从中获取数据的当前地址的最高 6 位，其值在每次从存储器获取数据时递增。

Bit 15:8 – SCANLADRH[7:0] 扫描起始/当前地址高字节
要从中获取数据的当前地址的高字节，其值在每次从存储器获取数据时递增。

Bit 7:0 – SCANLADRL[7:0] 扫描起始/当前地址低字节
要从中获取数据的当前地址的低字节，其值在每次从存储器获取数据时递增。

注：

1. 寄存器 SCANLADRU/H/L 构成一个 22 位值，但在原子或异步访问时不受保护；只能在 SCANGO = 0 时读写寄存器。
2. 当 SCANGO = 1 时，将忽略对该寄存器执行的写操作。

16.11.9. SCANHADR

名称： SCANHADR
偏移量： 0xF47
复位： 0

扫描高地址寄存器

位	23	22	21	20	19	18	17	16
			SCANHADRU[5:0]					
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			1	1	1	1	1	1

位	15	14	13	12	11	10	9	8
	SCANHADRH[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

位	7	6	5	4	3	2	1	0
	SCANHADRL[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 21:16 – SCANHADRU[5:0] 扫描结束地址位
所指定扫描的结束地址的最高 6 位

Bit 15:8 – SCANHADRH[7:0] 扫描结束地址位
所指定扫描的结束地址的高字节

Bit 7:0 – SCANHADRL[7:0] 扫描结束地址位
所指定扫描的结束地址的低字节

注：

1. 寄存器 SCANHADRU/H/L 构成一个 22 位值但在原子或异步访问时不受保护；只能在 SCANGO = 0 时读写寄存器。
2. 当 SCANGO = 1 时，将忽略对该寄存器执行的写操作。

16.11.10. SCANTRIG

名称： SCANTRIG
偏移量： 0xF4B
复位： 0

SCAN 触发选择寄存器

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

Bit 4:0 – TSEL[4:0] 扫描器数据触发输入选择位

表 16-5. 扫描触发源

TSEL	触发源
11111-11000	**保留**
10111	CLC8_out(1)
10110	CLC7_out(1)
10101	CLC6_out(1)
10100	CLC5_out(1)
10011	CLC4_out(1)
10010	CLC3_out(1)
10001	CLC2_out(1)
10000	CLC1_out(1)
01111-01001	**保留**
01000	TMR6_postscaled
00111	TMR5_output
00110	TMR4_postscaled
00101	TMR3_output
00100	TMR2_postscaled
00011	TMR1_output
00010	TMR0_output
00001	CLKREF
00000	LFINTOSC

注：
1. CN2510 器件上未提供。

16.12. 寄存器汇总——CRC

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0F43	保留									
0x0F44	SCANLADR	7:0	SCANLADRL[7:0]							
		15:8	SCANLADRH[7:0]							
		23:16			SCANLADRU[5:0]					
0x0F47	SCANHADR	7:0	SCANHADRL[7:0]							
		15:8	SCANHADRH[7:0]							
		23:16			SCANHADRU[5:0]					
0x0F4A	SCANCON0	7:0	SCANEN	SCANGO	BUSY	INVALID	INTM		MODE[1:0]	
0x0F4B	SCANTRIG	7:0				TSEL[4:0]				
0x0F4C ... 0x0F6E	保留									
0x0F6F	CRCDAT	7:0	CRCDATL[7:0]							
		15:8	CRCDATH[7:0]							
0x0F71	CRCACC	7:0	CRCACCL[7:0]							
		15:8	CRCACCH[7:0]							
0x0F73	CRCSHIFT	7:0	CRCSHIFTL[7:0]							
		15:8	CRCSHIFTH[7:0]							
0x0F75	CRCXOR	7:0	CRCXORL[6:0]							CRCXORL0
		15:8	CRCXORH[7:0]							
0x0F77	CRCCON0	7:0	EN	GO	BUSY	ACCM			SHIFTM	FULL
0x0F78	CRCCON1	7:0	DLEN[3:0]				PLEN[3:0]			

17. 中断

CN2710 器件具有多个中断源及一个中断优先级功能，该功能可以给大多数中断源分配高优先级或低优先级。高优先级中断向量位于 0008h，低优先级中断向量位于 0018h。高优先级中断事件可以中断正在处理的低优先级中断。

用于控制中断操作的寄存器如下：

- INTCON
- PIRx（中断标志）
- PIEx（中断允许）
- IPRx（高/低中断优先级）

建议使用所提供头文件中的名称来作为这些寄存器中的符号位名称。这使得汇编器/编译器能够自动存放指定寄存器中的这些位。

通常，中断源有 3 个位用于控制其操作。具体包括：

- **标志位**，用于指示发生了中断事件
- **使能位**，允许程序转移到中断向量地址处执行（当标志位置 1 时）
- **优先级位**，用于选择高优先级或低优先级

17.1. 与中档器件的兼容性

当 IPEN 位清零（默认状态）时，便会禁止中断优先级功能，此时中断与中档器件兼容。在兼容模式下，IPRx 寄存器的中断优先级位不起作用。PEIE/GIEL 位是外设的全局中断允许位。PEIE/GIEL 位只能禁止外设中断源，在 GIE/GIEH 位也置 1 时可以允许外设中断源。GIE/GIEH 位是全局中断允许位，它可以允许所有非外设中断源，可以禁止包括外设在内的所有中断源。在兼容模式下，所有中断均转移到地址 0008h。

17.2. 中断优先级

通过将 IPEN 位置 1，可使能中断优先级功能。使能中断优先级时，兼容模式的 GIE/GIEH 和 PEIE/GIEL 全局中断允许位被 GIEH（高优先级）和 GIEL（低优先级）全局中断允许位替代。当 IPEN 位置 1 时，GIEH 位会允许其在 IPRx 寄存器中的相关位置 1 的所有中断。当 GIEH 位清零时，会禁止所有中断源，包括在 IPRx 寄存器中选为低优先级的中断源。

当 GIEH 和 GIEL 位均置 1 时，会允许所有选择为低优先级中断源的中断。

高优先级中断会立即跳转到地址 00 0008h，低优先级中断会跳转到地址 00 0018h。

17.3. 中断响应

当响应中断时，全局中断允许位被清零以禁止后续中断。当 IPEN 位清零时，GIE/GIEH 位是全局中断允许位。当 IPEN 位置 1 时（即使能中断优先级），GIEH 位是高优先级全局中断允许位，GIEL 位是低优先级全局中断允许位。高优先级中断源会中断低优先级中断。在处理高优先级中断时，低优先级中断将不被处理。

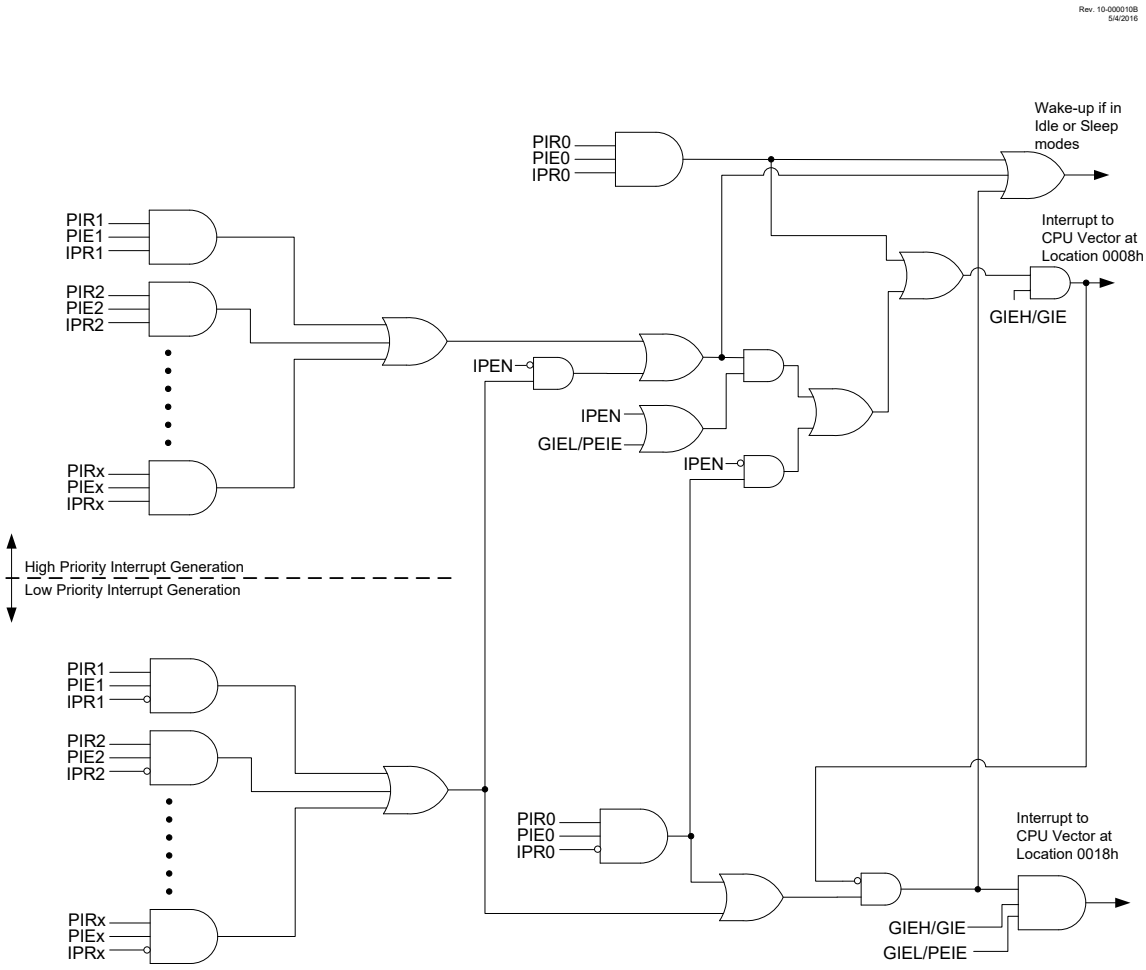
返回地址被压入堆栈，中断向量地址（0008h 或 0018h）被装入 PC。可以在中断服务程序中通过轮询 INTCONx 和 PIRx 寄存器的中断标志位来确定中断源。在重新允许中断前，必须用软件将中断标志位清零，以避免重复同一中断。

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

对于外部中断事件，例如 INT 引脚或电平变化中断引脚，中断响应延时将会是 3 个或 4 个指令周期。对于单周期或双周期指令，中断响应延时完全相同。不管相应的中断允许位或全局中断允许位的状态如何，各中断标志位都将置 1。

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

图 17-1. 中断逻辑



17.4. INTCON 寄存器

INTCON 寄存器是可读写的寄存器，包含各个中断允许位和优先级位。

17.5. PIR 寄存器

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 8 个 PIR 寄存器。

17.6. PIE 寄存器

PIE 寄存器包含各外设中断的允许位。由于外设中断源众多，所以共有 8 个外设中断允许寄存器。当 IPEN = 0 时，要允许任一外设中断，必须将 PEIE/GIEL 位置 1。

17.7. IPR 寄存器

IPR 寄存器包含各外设中断的优先级位。由于外设中断源众多，所以共有 8 个外设中断优先级寄存器。使用优先级位时，要求将中断优先级使能位（IPEN）置 1。

17.8. INTn 引脚中断

CN2710 器件具有外部中断源，它们可以使用 PPS 分配到 PORTA 和 PORTB 上的任意引脚。外部中断源由边沿触发。如果 INTCON 寄存器中相应的 INTxEDG 位置 1（= 1），则由上升沿触发中断。如果该位清零，则由下降沿触发中断。

如果在进入空闲或休眠模式之前 INTxE 位已置 1，则所有外部中断（INT0、INT1 和 INT2）均可将处理器从上述模式唤醒。如果全局中断允许位（GIE/GIEH）置 1，处理器将在唤醒后跳转到中断向量处执行程序。

中断优先级由 IPR0 寄存器的相应中断优先级位（INT0P、INT1P 和 INT2P）的值决定。

17.9. TMR0 中断

在 8 位模式（默认模式）下，TMR0 寄存器溢出（FFh → 00h）会将标志位 TMR0IF 置 1。在 16 位模式下，TMR0H:TMR0L 寄存器对溢出（FFFFh → 0000h）会将 TMR0IF 置 1。可以通过置 1 或清零允许位 TMR0IE 来允许或禁止该中断。Timer0 的中断优先级由中断优先级位 TMR0IP 的值决定。有关 Timer0 模块的详细信息，请参见 **“TMR0——Timer0 模块”** 一章。

17.10. 电平变化中断

支持 IOC 的任何端口引脚上的输入变化都会将标志位 IOCIF 置 1。可以通过置 1/清零允许位 IOCIE 来允许/禁止该中断。各个引脚还必须使用 IOCxP 和 IOCxN 寄存器分别使能。IOCIF 是只读位，该标志可通过清零相应的 IOCxF 寄存器来清零。更多信息，请参见 *“电平变化中断”* 一章。

17.11. 中断现场保护

在中断期间，PC 的返回地址被保存在堆栈中。此外，WREG、STATUS 和 BSR 寄存器的值被压入快速返回堆栈。如果未使用从中断快速返回功能，则用户可能需要在进入中断服务程序前，保存 WREG、STATUS 和 BSR 寄存器的值。根据用户的应用，可能还需要保存其他寄存器。将 STATUS、WREG 和 BSR 寄存器的值保存在 RAM 中在中断服务程序期间保存和恢复 WREG、STATUS 和 BSR 寄存器的值。

例 17-1. 将 STATUS、WREG 和 BSR 寄存器的值保存在 RAM 中

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

17.12. 寄存器定义：中断控制

17.12.1. INTCON

名称： INTCON
偏移量： 0xFF2

中断控制寄存器

位	7	6	5	4	3	2	1	0
	GIE/GIEH	PEIE/GIEL	IPEN			INT2EDG	INT1EDG	INT0EDG
访问	R/W	R/W	R/W			R/W	R/W	R/W
复位	0	0	0			1	1	1

Bit 7 – GIE/GIEH 全局中断允许位

值	条件	说明
1	如果 IPEN = 1	允许所有未被屏蔽的中断，仅高优先级中断可由硬件清零
0	如果 IPEN = 1	禁止所有中断
1	如果 IPEN = 0	允许所有未被屏蔽的中断，所有中断均可由硬件清零
0	如果 IPEN = 0	禁止所有中断

Bit 6 – PEIE/GIEL 外设中断允许位

值	条件	说明
1	如果 IPEN = 1	允许所有低优先级中断，仅低优先级中断可由硬件清零
0	如果 IPEN = 1	禁止所有低优先级中断
1	如果 IPEN = 0	允许所有未被屏蔽的外设中断
0	如果 IPEN = 0	禁止所有外设中断

Bit 5 – IPEN 中断优先级使能位

值	说明
1	使能中断优先级
0	禁止中断优先级

Bit 0, 1, 2 – INTxEDG 外部中断“x”边沿选择位

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

重要： 当中断条件产生时，不管相应的中断允许位或全局中断允许位的状态如何，中断标志位都将置 1。用户软件必须确保在允许一个中断前，先将相应的中断标志位清零。此功能允许软件轮询。

17.12.2. PIR0

名称： PIR0
偏移量： 0xEC5

外设中断请求（标志）寄存器 0

位	7	6	5	4	3	2	1	0
			TMR0IF	IOCIF		INT2IF	INT1IF	INT0IF
访问			R/W	R		R/W	R/W	R/W
复位			0	0		0	0	0

Bit 5 – TMR0IF Timer0 中断标志位(1)

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 电平变化中断标志位(1,2)

值	说明
1	发生了 IOC 事件（必须用软件清零）
0	未发生 IOC 事件

Bit 0, 1, 2 – INTxIF 外部中断“x”标志位(1,3)

值	说明
1	发生了外部中断“x”
0	未发生外部中断“x”

注：

1. PEIE 位不会禁止中断。
2. IOCIF 是只读位；要清除中断条件，必须将 IOCF 寄存器中的所有位清零。
3. 外部中断 GPIO 引脚通过 INTPPS 寄存器选择。

17.12.3. PIR1

名称： PIR1
偏移量： 0xEC6

外设中断请求（标志）寄存器 1

位	7	6	5	4	3	2	1	0
	OSCFIF	CSWIF					ADTIF	ADIF
访问	R/W	R/W					R/W	R/W
复位	0	0					0	0

Bit 7 – OSCFIF 振荡器故障中断标志位

值	说明
1	器件振荡器发生故障，时钟输入已更改为 HFINTOSC（必须用软件清零）
0	器件时钟正在运行

Bit 6 – CSWIF 时钟切换中断标志位(1)

值	说明
1	新振荡器已准备好进行切换（必须用软件清零）
0	新振荡器尚未准备好进行切换或尚未启动

Bit 1 – ADTIF ADC 阈值中断标志位

值	说明
1	发生了 ADC 阈值中断（必须用软件清零）
0	ADC 阈值事件尚未完成或尚未开始

Bit 0 – ADIF ADC 中断标志位

值	说明
1	A/D 转换已完成（必须用软件清零）
0	A/D 转换尚未完成或尚未开始

注：

1. CSWIF 中断不会将系统从休眠模式唤醒。系统将休眠，直到另一个中断将其唤醒。

17.12.4. PIR2

名称： PIR2
偏移量： 0xEC7

外设中断请求（标志）寄存器 2

位	7	6	5	4	3	2	1	0
	HLVDIF	ZCDIF					C2IF	C1IF
访问	R/W	R/W					R/W	R/W
复位	0	0					0	0

Bit 7 – HLVDIF HLVD 中断标志位

值	说明
1	HLVD 中断事件已发生
0	HLVD 中断事件尚未发生或尚未设置

Bit 6 – ZCDIF 过零检测中断标志位

值	说明
1	ZCD 输出已改变（必须用软件清零）
0	ZCD 输出未改变

Bit 0, 1 – CxIF 比较器“x”中断标志位

值	说明
1	比较器 Cx 输出已改变（必须用软件清零）
0	比较器 Cx 输出未改变

17.12.5. PIR3

名称： PIR3
偏移量： 0xEC8

外设中断请求（标志）寄存器 3

位	7	6	5	4	3	2	1	0
	RC2IF	TX2IF	RC1IF	TX1IF	BCL2IF	SSP2IF	BCL1IF	SSP1IF
访问	R	R	R	R	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 5, 7 – RCxIF EUSARTx 接收中断标志位

值	说明
1	EUSARTx 接收缓冲区 RCxREG 已满（通过读取 RCxREG 清零）
0	EUSARTx 接收缓冲区为空

Bit 4, 6 – TXxIF EUSARTx 发送中断标志位

值	说明
1	EUSARTx 发送缓冲区 TXxREG 为空（通过写入 TXxREG 清零）
0	EUSARTx 发送缓冲区已满

Bit 1, 3 – BCLxIF MSSPx 总线冲突中断标志位

值	说明
1	当配置为 I^2C 主模式的 MSSPx 模块执行发送操作时发生了总线冲突（必须用软件清零）
0	未发生总线冲突

Bit 0, 2 – SSPxIF 同步串行端口“x”中断标志位

值	说明
1	发送/接收已完成（必须用软件清零）
0	等待发送/接收

17.12.6. PIR4

名称： PIR4
偏移量： 0xEC9

外设中断请求（标志）寄存器 4

位	7	6	5	4	3	2	1	0
			TMR6IF	TMR5IF	TMR4IF	TMR3IF	TMR2IF	TMR1IF
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			0	0	0	0	0	0

Bit 5 – TMR6IF TMR6 与 PR6 匹配中断标志位

值	说明
1	TMR6 与 PR6 发生匹配（必须用软件清零）
0	TMR6 与 PR6 未发生匹配

Bit 4 – TMR5IF TMR5 溢出中断标志位

值	说明
1	TMR5 寄存器已溢出（必须用软件清零）
0	TMR5 寄存器未溢出

Bit 3 – TMR4IF TMR4 与 PR4 匹配中断标志位

值	说明
1	TMR4 与 PR4 发生匹配（必须用软件清零）
0	TMR4 与 PR4 未发生匹配

Bit 2 – TMR3IF TMR3 溢出中断标志位

值	说明
1	TMR3 寄存器已溢出（必须用软件清零）
0	TMR3 寄存器未溢出

Bit 1 – TMR2IF TMR2 与 PR2 匹配中断标志位

值	说明
1	TMR2 与 PR2 发生匹配（必须用软件清零）
0	TMR2 与 PR2 未发生匹配

Bit 0 – TMR1IF TMR1 溢出中断标志位

值	说明
1	TMR1 寄存器已溢出（必须用软件清零）
0	TMR1 寄存器未溢出

17.12.7. PIR5

名称： PIR5
偏移量： 0xECA

外设中断请求（标志）寄存器 5

位	7	6	5	4	3	2	1	0
	CLC4IF	CLC3IF	CLC2IF	CLC1IF		TMR5GIF	TMR3GIF	TMR1GIF
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 4, 5, 6, 7 – CLCxIF CLCx 中断标志位

值	说明
1	发生了 CLCx 中断
0	未发生 CLCx 中断

Bit 2 – TMR5GIF TMR5 门控中断标志位

值	说明
1	发生了 TMR5 门控中断（必须用软件清零）
0	未发生 TMR5 门控中断

Bit 1 – TMR3GIF TMR3 门控中断标志位

值	说明
1	发生了 TMR3 门控中断（必须用软件清零）
0	未发生 TMR3 门控中断

Bit 0 – TMR1GIF TMR1 门控中断标志位

值	说明
1	发生了 TMR1 门控中断（必须用软件清零）
0	未发生 TMR1 门控中断

17.12.8. PIR6

名称： PIR6
偏移量： 0xECB

PIR6 外设中断请求（标志）寄存器 6

位	7	6	5	4	3	2	1	0
	CLC8IF	CLC7IF	CLC6IF	CLC5IF			CCP2IF	CCP1IF
访问	R/W	R/W	R/W	R/W			R/W	R/W
复位	0	0	0	0			0	0

Bit 4, 5, 6, 7 – CLCyIF CLCx 中断标志位

值	说明
1	发生了 CLCx 中断
0	未发生 CLCx 中断

Bit 1 – CCP2IF ECCP2 中断标志位

值	条件	说明
1	捕捉模式	发生了 TMR 寄存器捕捉（必须用软件清零）
0	捕捉模式	未发生 TMR 寄存器捕捉
1	比较模式	发生了 TMR 寄存器比较匹配（必须用软件清零）
0	比较模式	未发生 TMR 寄存器比较匹配
—	PWM 模式	在 PWM 模式下不使用

Bit 0 – CCP1IF ECCP1 中断标志位

值	条件	说明
1	捕捉模式	发生了 TMR 寄存器捕捉（必须用软件清零）
0	捕捉模式	未发生 TMR 寄存器捕捉
1	比较模式	发生了 TMR 寄存器比较匹配（必须用软件清零）
0	比较模式	未发生 TMR 寄存器比较匹配
—	PWM 模式	在 PWM 模式下不使用

17.12.9. PIR7

名称： PIR7
偏移量： 0xECC

外设中断请求（标志）寄存器 7

位	7	6	5	4	3	2	1	0
	SCANIF	CRCIF	NVMIF					CWG1IF
访问	R/W	R/W	R/W					R/W
复位	0	0	0					0

Bit 7 – SCANIF SCAN 中断标志位

值	说明
1	SCAN 中断已发生（必须用软件清零）
0	SCAN 中断尚未发生或尚未开始

Bit 6 – CRCIF CRC 中断标志位

值	说明
1	CRC 中断已发生（必须用软件清零）
0	CRC 中断尚未发生或尚未开始

Bit 5 – NVMIF NVM 中断标志位

值	说明
1	NVM 中断已发生（必须用软件清零）
0	NVM 中断尚未发生或尚未开始

Bit 0 – CWG1IF CWG 中断标志位

值	说明
1	CWG 中断已发生（必须用软件清零）
0	CWG 中断尚未发生或尚未开始

17.12.10. PIE0

名称： PIE0
偏移量： 0xEBD

外设中断允许寄存器 0

位	7	6	5	4	3	2	1	0
			TMR0IE	IOCIE		INT2IE	INT1IE	INT0IE
访问			R/W	R/W		R/W	R/W	R/W
复位			0	0		0	0	0

Bit 5 – TMR0IE Timer0 中断允许位(1)

值	说明
1	使能
0	禁止

Bit 4 – IOCIE 电平变化中断允许位(1)

值	说明
1	使能
0	禁止

Bit 0, 1, 2 – INTxIE 外部中断“x”允许位(1)

值	说明
1	使能
0	禁止

注：

1. INTCON 寄存器中的 PEIE 位不会禁止 PIR0 中断。

17.12.11. PIE1

名称： PIE1
偏移量： 0xEBE

外设中断允许寄存器 1

位	7	6	5	4	3	2	1	0
	OSCFIE	CSWIE					ADTIE	ADIE
访问	R/W	R/W					R/W	R/W
复位	0	0					0	0

Bit 7 – OSCFIE 振荡器故障中断允许位

值	说明
1	使能
0	禁止

Bit 6 – CSWIE 时钟切换中断允许位

值	说明
1	使能
0	禁止

Bit 1 – ADTIE ADC 阈值中断允许位

值	说明
1	使能
0	禁止

Bit 0 – ADIE ADC 中断允许位

值	说明
1	使能
0	禁止

17.12.12. PIE2

名称： PIE2
偏移量： 0xEBF

外设中断允许寄存器 2

位	7	6	5	4	3	2	1	0
	HLVDIE	ZCDIE					C2IE	C1IE
访问	R/W	R/W					R/W	R/W
复位	0	0					0	0

Bit 7 – HLVDIE HLVD 中断允许位

值	说明
1	使能
0	禁止

Bit 6 – ZCDIE 过零检测中断允许位

值	说明
1	使能
0	禁止

Bit 0, 1 – CxIE 比较器“x”中断允许位

值	说明
1	使能
0	禁止

17.12.13. PIE3

名称： PIE3
偏移量： 0xEC0

外设中断允许寄存器 3

位	7	6	5	4	3	2	1	0
	RC2IE	TX2IE	RC1IE	TX1IE	BCL2IE	SSP2IE	BCL1IE	SSP1IE
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 5, 7 – RCxIE EUSARTx 接收中断允许位

值	说明
1	使能
0	禁止

Bit 4, 6 – TXxIE EUSARTx 发送中断允许位

值	说明
1	使能
0	禁止

Bit 1, 3 – BCLxIE MSSPx 总线冲突中断允许位

值	说明
1	使能
0	禁止

Bit 0, 2 – SSPxIE 同步串行端口“x”中断允许位

值	说明
1	使能
0	禁止

17.12.14. PIE4

名称： PIE4
偏移量： 0xEC1

外设中断允许寄存器 4

位	7	6	5	4	3	2	1	0
			TMR6IE	TMR5IE	TMR4IE	TMR3IE	TMR2IE	TMR1IE
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			0	0	0	0	0	0

Bit 5 – TMR6IE TMR6 与 PR6 匹配中断允许位

值	说明
1	使能
0	禁止

Bit 4 – TMR5IE TMR5 溢出中断允许位

值	说明
1	使能
0	禁止

Bit 3 – TMR4IE TMR4 与 PR4 匹配中断允许位

值	说明
1	使能
0	禁止

Bit 2 – TMR3IE TMR3 溢出中断允许位

值	说明
1	使能
0	禁止

Bit 1 – TMR2IE TMR2 与 PR2 匹配中断允许位

值	说明
1	使能
0	禁止

Bit 0 – TMR1IE TMR1 溢出中断允许位

值	说明
1	使能
0	禁止

17.12.15. PIE5

名称： PIE5
偏移量： 0xEC2

外设中断允许寄存器 5

位	7	6	5	4	3	2	1	0
	CLC4IE	CLC3IE	CLC2IE	CLC1IE		TMR5GIE	TMR3GIE	TMR1GIE
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 4, 5, 6, 7 – CLCxIE CLCx 中断允许位

值	说明
1	允许 CLCx 中断
0	禁止 CLCx 中断

Bit 2 – TMR5GIE TMR5 门控中断允许位

值	说明
1	使能
0	禁止

Bit 1 – TMR3GIE TMR3 门控中断允许位

值	说明
1	使能
0	禁止

Bit 0 – TMR1GIE TMR1 门控中断允许位

值	说明
1	使能
0	禁止

17.12.16. PIE6

名称： PIE6
偏移量： 0xEC3

外设中断允许寄存器 6

位	7	6	5	4	3	2	1	0
	CLC8IE	CLC7IE	CLC6IE	CLC5IE			CCP2IE	CCP1IE
访问	R/W	R/W	R/W	R/W			R/W	R/W
复位	0	0	0	0			0	0

Bit 4, 5, 6, 7 – CLCyIE CLCx 中断允许位

值	说明
1	允许 CLCx 中断
0	禁止 CLCx 中断

Bit 1 – CCP2IE ECCP2 中断允许位

值	说明
1	使能
0	禁止

Bit 0 – CCP1IE ECCP1 中断允许位

值	说明
1	使能
0	禁止

17.12.17. PIE7

名称： PIE7
偏移量： 0xEC4

外设中断允许寄存器 7

位	7	6	5	4	3	2	1	0
	SCANIE	CRCIE	NVMIE					CWG1IE
访问	R/W	R/W	R/W					R/W
复位	0	0	0					0

Bit 7 – SCANIE SCAN 中断允许位

值	说明
1	使能
0	禁止

Bit 6 – CRCIE CRC 中断允许位

值	说明
1	使能
0	禁止

Bit 5 – NVMIE NVM 中断允许位

值	说明
1	使能
0	禁止

Bit 0 – CWG1IE CWG 中断允许位

值	说明
1	使能
0	禁止

17.12.18. IPR0

名称： IPR0
偏移量： 0xEB5

外设中断优先级寄存器 0

位	7	6	5	4	3	2	1	0
			TMR0IP	IOCIP		INT2IP	INT1IP	INT0IP
访问			R/W	R/W		R/W	R/W	R/W
复位			1	1		1	1	1

Bit 5 – TMR0IP Timer0 中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 4 – IOCIP 电平变化中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 0, 1, 2 – INTxIP 外部中断“x”优先级位

值	说明
1	高优先级
0	低优先级

17.12.19. IPR1

名称： IPR1
偏移量： 0xEB6

外设中断优先级寄存器 1

位	7	6	5	4	3	2	1	0
	OSCFIP	CSWIP					ADTIP	ADIP
访问	R/W	R/W					R/W	R/W
复位	1	1					1	1

Bit 7 – OSCFIP 振荡器故障中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 6 – CSWIP 时钟切换中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 1 – ADTIP ADC 阈值中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 0 – ADIP ADC 中断优先级位

值	说明
1	高优先级
0	低优先级

17.12.20. IPR2

名称： IPR2
偏移量： 0xEB7

外设中断优先级寄存器 2

位	7	6	5	4	3	2	1	0
	HLVDIP	ZCDIP					C2IP	C1IP
访问	R/W	R/W					R/W	R/W
复位	1	1					1	1

Bit 7 – HLVDIP HLVD 中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 6 – ZCDIP 过零检测中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 0, 1 – CxIP 比较器“x”中断优先级位

值	说明
1	高优先级
0	低优先级

17.12.21. IPR3

名称： IPR3
偏移量： 0xEB8

外设中断优先级寄存器 3

位	7	6	5	4	3	2	1	0
	RC2IP	TX2IP	RC1IP	TX1IP	BCL2IP	SSP2IP	BCL1IP	SSP1IP
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 5, 7 – RCxIP EUSARTx 接收中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 4, 6 – TXxIP EUSARTx 发送中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 1, 3 – BCLxIP MSSPx 总线冲突中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 0, 2 – SSPxIP 同步串行端口“x”中断优先级位

值	说明
1	高优先级
0	低优先级

17.12.22. IPR4

名称： IPR4
偏移量： 0xEB9

外设中断优先级寄存器 4

位	7	6	5	4	3	2	1	0
			TMR6IP	TMR5IP	TMR4IP	TMR3IP	TMR2IP	TMR1IP
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			1	1	1	1	1	1

Bit 5 – TMR6IP TMR6 与 PR6 匹配中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 4 – TMR5IP TMR5 溢出中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 3 – TMR4IP TMR4 与 PR4 匹配中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 2 – TMR3IP TMR3 溢出中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 1 – TMR2IP TMR2 与 PR2 匹配中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 0 – TMR1IP TMR1 溢出中断优先级位

值	说明
1	高优先级
0	低优先级

17.12.23. IPR5

名称： IPR5
偏移量： 0xEBA

外设中断优先级寄存器 5

位	7	6	5	4	3	2	1	0
	CLC4IP	CLC3IP	CLC2IP	CLC1IP		TMR5GIP	TMR3GIP	TMR1GIP
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 4, 5, 6, 7 – CLCxIP CLCx 中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 2 – TMR5GIP TMR5 门控中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 1 – TMR3GIP TMR3 门控中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 0 – TMR1GIP TMR1 门控中断优先级位

值	说明
1	高优先级
0	低优先级

17.12.24. IPR6

名称： IPR6
偏移量： 0xEBB

外设中断优先级寄存器

位	7	6	5	4	3	2	1	0
	CLC8IP	CLC7IP	CLC6IP	CLC5IP			CCP2IP	CCP1IP
访问	R/W	R/W	R/W	R/W			R/W	R/W
复位	1	1	1	1			1	1

Bit 4, 5, 6, 7 – CLCyIP CLCx 中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 1 – CCP2IP ECCP2 中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 0 – CCP1IP ECCP1 中断优先级位

值	说明
1	高优先级
0	低优先级

17.12.25. IPR7

名称： IPR7
偏移量： 0xEBC

外设中断优先级寄存器

位	7	6	5	4	3	2	1	0
	SCANIP	CRCIP	NVMIP					CWG1IP
访问	R/W	R/W	R/W					R/W
复位	1	1	1					1

Bit 7 – SCANIP SCAN 中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 6 – CRCIP CRC 中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 5 – NVMIP NVM 中断优先级位

值	说明
1	高优先级
0	低优先级

Bit 0 – CWG1IP CWG 中断优先级位

值	说明
1	高优先级
0	低优先级

17.13. 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 ... 0x0EB4	保留									
0x0EB5	IPR0	7:0			TMR0IP	IOCIP		INT2IP	INT1IP	INT0IP
0x0EB6	IPR1	7:0	OSCFIP	CSWIP					ADTIP	ADIP
0x0EB7	IPR2	7:0	HLVDIP	ZCDIP					C2IP	C1IP
0x0EB8	IPR3	7:0	RC2IP	TX2IP	RC1IP	TX1IP	BCL2IP	SSP2IP	BCL1IP	SSP1IP
0x0EB9	IPR4	7:0			TMR6IP	TMR5IP	TMR4IP	TMR3IP	TMR2IP	TMR1IP
0x0EBA	IPR5	7:0	CLC4IP	CLC3IP	CLC2IP	CLC1IP		TMR5GIP	TMR3GIP	TMR1GIP
0x0EBB	IPR6	7:0	CLC8IP	CLC7IP	CLC6IP	CLC5IP			CCP2IP	CCP1IP
0x0EBC	IPR7	7:0	SCANIP	CRCIP	NVMIP					CWG1IP
0x0EBD	PIE0	7:0			TMR0IE	IOCIE		INT2IE	INT1IE	INT0IE
0x0EBE	PIE1	7:0	OSCFIE	CSWIE					ADTIE	ADIE
0x0EBF	PIE2	7:0	HLVDIE	ZCDIE					C2IE	C1IE
0x0EC0	PIE3	7:0	RC2IE	TX2IE	RC1IE	TX1IE	BCL2IE	SSP2IE	BCL1IE	SSP1IE
0x0EC1	PIE4	7:0			TMR6IE	TMR5IE	TMR4IE	TMR3IE	TMR2IE	TMR1IE
0x0EC2	PIE5	7:0	CLC4IE	CLC3IE	CLC2IE	CLC1IE		TMR5GIE	TMR3GIE	TMR1GIE
0x0EC3	PIE6	7:0	CLC8IE	CLC7IE	CLC6IE	CLC5IE			CCP2IE	CCP1IE
0x0EC4	PIE7	7:0	SCANIE	CRCIE	NVMIE					CWG1IE
0x0EC5	PIR0	7:0			TMR0IF	IOCIF		INT2IF	INT1IF	INT0IF
0x0EC6	PIR1	7:0	OSCFIF	CSWIF					ADTIF	ADIF
0x0EC7	PIR2	7:0	HLVDIF	ZCDIF					C2IF	C1IF
0x0EC8	PIR3	7:0	RC2IF	TX2IF	RC1IF	TX1IF	BCL2IF	SSP2IF	BCL1IF	SSP1IF
0x0EC9	PIR4	7:0			TMR6IF	TMR5IF	TMR4IF	TMR3IF	TMR2IF	TMR1IF
0x0ECA	PIR5	7:0	CLC4IF	CLC3IF	CLC2IF	CLC1IF		TMR5GIF	TMR3GIF	TMR1GIF
0x0ECB	PIR6	7:0	CLC8IF	CLC7IF	CLC6IF	CLC5IF			CCP2IF	CCP1IF
0x0ECC	PIR7	7:0	SCANIF	CRCIF	NVMIF					CWG1IF
0x0ECD ... 0x0FF1	保留									
0x0FF2	INTCON	7:0	GIE/GIEH	PEIE/GIEL	IPEN			INT2EDG	INT1EDG	INT0EDG

18. I/O 端口

表 18-1. 每款器件可用的端口

器件	PORTA	PORTB	PORTC	PORTD	PORTE
CN2510/2610/2710	●	●	●		●

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 寄存器（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）

大多数端口引脚与器件模拟外设和数字外设共用功能。通常，当使能某个端口引脚上的外设时，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。

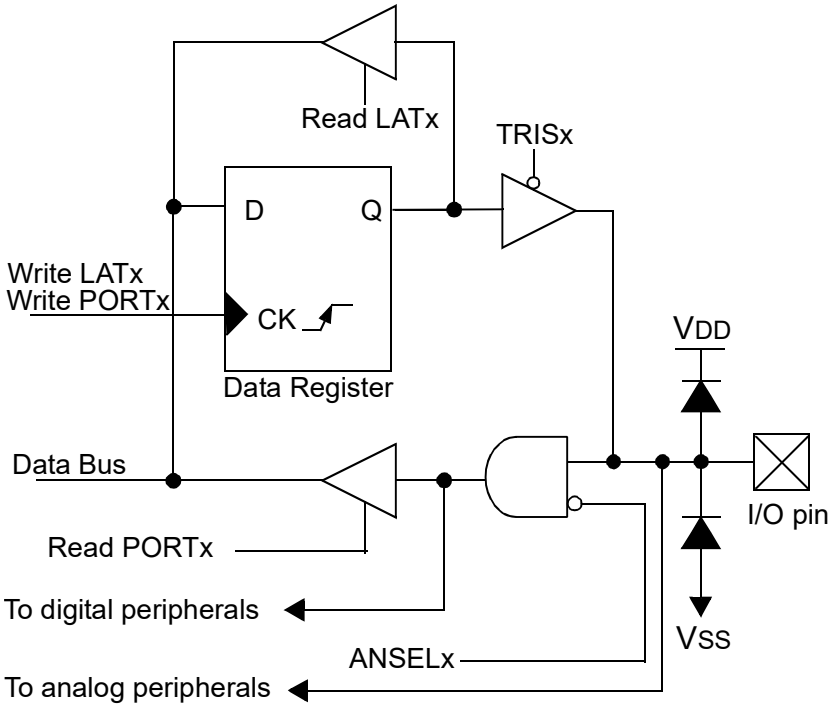
数据锁存器（LATx 寄存器）在对 I/O 引脚驱动值进行读-修改-写操作时非常有用。

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 端口锁存器中保存的值，而读取 PORTx 寄存器时，将会读取实际的 I/O 引脚值。

支持模拟输入的端口具有相关的 ANSELx 寄存器。当某个 ANSELx 位置 1 时，与该位关联的数字输入缓冲器会被禁止。

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压之间的模拟信号电平在逻辑输入电路中产生过大的电流。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 18-1. 通用 I/O 端口的工作原理



18.1. I/O 优先级

在复位之后，每个引脚均默认设为端口数据锁存器。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. 配置位
2. 模拟输出（禁止输入缓冲器）
3. 模拟输入
4. PPS 的端口输入和输出

18.2. PORTx 寄存器

在本节中，可以将通用名称（PORTx、LATx 和 TRISx 等）与所有端口关联。有关 PORTD 的可用性，请参见表 18-1。PORTE 的功能与其他端口有所不同，将另开一个章节单独进行说明。

18.2.1. 数据寄存器

PORTx 是 8 位宽的双向端口。对应的数据方向寄存器是 TRISx。将 TRISx 位置 1（1）时，会将 PORTA 的相应引脚设为输入（即，禁止输出驱动器）。将 TRISx 位清零（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）。**初始化 PORTA** 示例展示了如何初始化 PORTA。

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端口引脚电平状态，然后修改此值，最后再写入该端口的数据锁存器（LATx）。

端口数据锁存器 LATx 保存输出端口数据，并包含写入 LATx 或 PORTx 的最新值。

例 18-1. 初始化 PORTA

```
; This code example illustrates initializing the PORTA register.
; The other ports are initialized in the same manner.
        CLRF    LATA            ; Set all output bits to zero
        MOVLW   B'11111000'     ; Set RA[7:3] as inputs and RA[2:0] as outputs
        MOVWF   TRISA           ;
        BANKSEL ANSELA
        CLRF    ANSELA          ; All pins are digital I/O
```

18.2.2. 方向控制

TRISx 寄存器用于控制 PORTx 引脚输出驱动器，即使它们被用作模拟输入也是如此。当引脚用作模拟输入时，用户必须确保 TRISx 寄存器中的相应位保持置 1。配置为模拟输入的 I/O 引脚总是读为 0。

18.2.3. 模拟控制

ANSELx 寄存器用于将 I/O 引脚的输入模式配置为模拟。将相应的 ANSELx 位设置为高电平后，引脚上的所有数字读操作都将读为 0，引脚上的模拟功能将正常工作。

ANSELx 位的状态不会影响数字输出功能。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。当在受影响的端口上执行 READ-MODIFY-WRITE 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，必须通过用户软件将相应的 ANSEL 位初始化为 0。

18.2.4. 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 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，不需要设置漏极开路控制；I^2C 模块控制该引脚，使引脚漏极开路。

18.2.5. 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 时，相应 PORT 引脚驱动器的压摆率会受到限制。当 SLRCONx 位清零时，相应 PORT 引脚驱动器的压摆率将为最大可能值。

18.2.6. 输入阈值控制

INLVLx 寄存器用于控制每个可用 PORTx 输入引脚的输入电压阈值。用户可以选择施密特触发器 CMOS 阈值或 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）。

重要：如果要更改所选择的输入阈值，可以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。

18.2.7. 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。

18.2.8. 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端口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。电平变化中断模块位于所有引脚上。有关 IOC 模块的详细信息，请参见“**电平变化中断**”一章。

18.3. PORTE 寄存器

例 18-2. 示例 2：初始化 PORTE

```
CLRF        PORTE       ; Initialize PORTE by
                        ; clearing output
                        ; data latches
CLRF        LATE        ; Alternate method
                        ; to clear output
                        ; data latches
CLRF        ANSELE        ; Configure analog pins
                        ; for digital only
MOVLW        05h        ; Value used to
                        ; initialize data
                        ; direction
MOVWF        TRISE        ; Set RE<0> as input
                        ; RE<1> as output
                        ; RE<2> as input
```

18.3.1. 28 引脚器件上的 PORTE

对于 28 引脚器件，PORTE 仅在主复位功能禁止（MCLRE = 0）时可用。在这种情况下，PORTE 为单位仅输入端口，仅由 RE3 组成。引脚工作原理与之前所述相同。PORTE 寄存器的 RE3 是只读位，在 MCLRE = 1 时（即，使能主复位）读为 1。

18.3.2. RE3 弱上拉

端口 RE3 引脚具有单独控制的内部弱上拉功能。置 1 时，WPUE3 位使能 RE3 引脚上拉。当 RE3 端口引脚配置为 $\overline{\text{MCLR}}$（CONFIG2L、MCLRE = 1 且 CONFIG4H、LVP = 0）或配置为低电压编程（MCLRE = x 且 LVP = 1）时，始终使能上拉，WPUE3 位不产生任何影响。

18.3.3. PORTE 电平变化中断

所有器件都只有 RE3 引脚提供电平变化中断功能。

18.4. 寄存器定义：端口控制

18.4.1. PORTA

名称： PORTA
偏移量： 0xF8C

PORTA 寄存器
注：写入 PORTA 时，实际上会写入相应的 LATA 寄存器。
读取 PORTA 寄存器时，将返回实际的 I/O 引脚值。

位	7	6	5	4	3	2	1	0
	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
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 – RAn 端口 I/O 值位

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}

18.4.2. PORTB

名称： PORTB
偏移量： 0xF8D

PORTB 寄存器

位	7	6	5	4	3	2	1	0
	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
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 – RBn 端口 I/O 值位
注：在调试模式下，RB6 和 RB7 位读为 1。

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}

注：写入 PORTB 时，实际上会写入相应的 LATB 寄存器。
读取 PORTB 寄存器时，将返回实际的 I/O 引脚值。

18.4.3. PORTC

名称： PORTC
偏移量： 0xF8E

PORTC 寄存器

位	7	6	5	4	3	2	1	0
	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
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 – RCn 端口 I/O 值位

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}

注：写入 PORTC 时，实际上会写入相应的 LATC 寄存器。
读取 PORTC 寄存器时，将返回实际的 I/O 引脚值。

18.4.4. PORTE

名称： PORTE
偏移量： 0xF90

PORTE 寄存器

位	7	6	5	4	3	2	1	0
					RE3			
访问					R			
复位					x			

Bit 3 – RE3 PORTE I/O 值位

复位状态： POR/BOR = x
所有其他复位 = u

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

注：

- 写入 PORTE 时，实际上会写入相应的 LATE 寄存器。
  读取 PORTE 寄存器时，将返回实际的 I/O 引脚值。
- RE3 位为只读位，在 MCLRE = 1（使能主复位）时读为 1，在使能 $\overline{\text{DEBUG}}$ 时读为 0。

18.4.5. TRISA

名称： TRISA
偏移量： 0xF87

三态控制寄存器

位	7	6	5	4	3	2	1	0
	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
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 – TRISAn TRISA 端口 I/O 三态控制位

值	说明
1	禁止端口输出驱动器
0	使能端口输出驱动器

18.4.6. TRISB

名称： TRISB
偏移量： 0xF88

三态控制寄存器

位	7	6	5	4	3	2	1	0
	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
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 – TRISBn TRISB 端口 I/O 三态控制位
注：在调试模式下，TRISB6 和 TRISB7 位读为 1。

值	说明
1	禁止端口输出驱动器
0	使能端口输出驱动器

18.4.7. TRISC

名称： TRISC
偏移量： 0xF89

三态控制寄存器

位	7	6	5	4	3	2	1	0
	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
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 – TRISCn TRISC 端口 I/O 三态控制位

值	说明
1	禁止端口输出驱动器
0	使能端口输出驱动器

18.4.8. LATA

名称： LATA
偏移量： 0xF82

输出锁存器寄存器

位	7	6	5	4	3	2	1	0
	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0
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 – LATAn 输出锁存器 A 值位

复位状态： POR/BOR = xxxxxxxx
所有其他复位时的值 = uuuuuuuuu

注：写入 LATA 相当于写入相应的 PORTA 寄存器。读取 LATA 寄存器时，将返回寄存器值，而不是 I/O 引脚值。

18.4.9. LATB

名称： LATB
偏移量： 0xF83

输出锁存器寄存器

位	7	6	5	4	3	2	1	0
	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
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 – LATBn 输出锁存器 B 值位

复位状态： POR/BOR = xxxxxxxx
所有其他复位时的值 = uuuuuuuuu

注：写入 LATB 相当于写入相应的 PORTB 寄存器。读取 LATB 寄存器时，将返回寄存器值，而不是 I/O 引脚值。

18.4.10. LATC

名称： LATC
偏移量： 0xF84

输出锁存器寄存器

位	7	6	5	4	3	2	1	0
	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0
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 – LATCn 输出锁存器 C 值位

复位状态： POR/BOR = xxxxxxxx
所有其他复位时的值 = uuuuuuuuu

注：写入 LATC 相当于写入相应的 PORTC 寄存器。读取 LATC 寄存器时，将返回寄存器值，而不是 I/O 引脚值。

18.4.11. ANSELA

名称： ANSELA
偏移量： 0xF0C

模拟选择寄存器

位	7	6	5	4	3	2	1	0
	ANSELA7	ANSELA6	ANSELA5	ANSELA4	ANSELA3	ANSELA2	ANSELA1	ANSELA0
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 – ANSELAn 引脚 RA[7:0]的模拟选择

值	说明
1	禁止数字输入缓冲器
0	使能 ST 和 TTL 输入缓冲器

18.4.12. ANSELB

名称： ANSELB
偏移量： 0xF14
复位： 0x00

模拟选择寄存器

位	7	6	5	4	3	2	1	0
	ANSELB7	ANSELB6	ANSELB5	ANSELB4	ANSELB3	ANSELB2	ANSELB1	ANSELB0
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 – ANSELBn 引脚 RB[7:0]的模拟选择

值	说明
1	禁止数字输入缓冲器
0	使能 ST 和 TTL 输入缓冲器

18.4.13. ANSELC

名称： ANSELC
偏移量： 0xF1C

模拟选择寄存器

位	7	6	5	4	3	2	1	0
	ANSELC7	ANSELC6	ANSELC5	ANSELC4	ANSELC3	ANSELC2	ANSELC1	ANSELC0
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 – ANSELCn 引脚 RC[7:0]的模拟选择

值	说明
1	禁止数字输入缓冲器
0	使能 ST 和 TTL 输入缓冲器

18.4.14. WPUA

名称： WPUA
偏移量： 0xF0B

弱上拉寄存器

位	7	6	5	4	3	2	1	0
	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
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 – WPUAn 弱上拉 PORTA 控制位

值	说明
1	使能弱上拉
0	禁止弱上拉

18.4.15. WPUB

名称： WPUB
偏移量： 0xF13

弱上拉寄存器

位	7	6	5	4	3	2	1	0
	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
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 – WPUBn 弱上拉 PORTA 控制位

值	说明
1	使能弱上拉
0	禁止弱上拉

18.4.16. WPUC

名称： WPUC
偏移量： 0xF1B

弱上拉寄存器

位	7	6	5	4	3	2	1	0
	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
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 – WPUCn 弱上拉 PORTC 控制位

值	说明
1	使能弱上拉
0	禁止弱上拉

18.4.17. WPUE

名称： WPUE
偏移量： 0xF28

弱上拉寄存器

位	7	6	5	4	3	2	1	0
					WPUE3			
访问					R/W			
复位					0			

Bit 3 – WPUE3 弱上拉 PORTE 控制位
注：如果 MCLRE = 1，则 RE3 的弱上拉始终使能；WPUE3 位不受影响。

值	说明
1	使能弱上拉
0	禁止弱上拉

18.4.18. ODCONA

名称： ODCONA
偏移量： 0xF0A

漏极开路控制寄存器

位	7	6	5	4	3	2	1	0
	ODCA7	ODCA6	ODCA5	ODCA4	ODCA3	ODCA2	ODCA1	ODCA0
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 – ODCAn 引脚 Rx[7:0]的漏极开路配置

值	说明
1	输出仅驱动低电平信号（仅灌电流）
0	输出驱动高电平和低电平信号（拉电流和灌电流）

18.4.19. ODCONB

名称： ODCONB
偏移量： 0xF12

漏极开路控制寄存器

位	7	6	5	4	3	2	1	0
	ODCB7	ODCB6	ODCB5	ODCB4	ODCB3	ODCB2	ODCB1	ODCB0
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 – ODCBn 引脚 Rx[7:0]的漏极开路配置

值	说明
1	输出仅驱动低电平信号（仅灌电流）
0	输出驱动高电平和低电平信号（拉电流和灌电流）

18.4.20. ODCONC

名称： ODCONC
偏移量： 0xF1A

漏极开路控制寄存器

位	7	6	5	4	3	2	1	0
	ODCC7	ODCC6	ODCC5	ODCC4	ODCC3	ODCC2	ODCC1	ODCC0
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 – ODCCn 引脚 Rx[7:0]的漏极开路配置

值	说明
1	输出仅驱动低电平信号（仅灌电流）
0	输出驱动高电平和低电平信号（拉电流和灌电流）

18.4.21. SLRCONA

名称： SLRCONA
偏移量： 0xF09

压摆率控制寄存器

位	7	6	5	4	3	2	1	0
	SLRA7	SLRA6	SLRA5	SLRA4	SLRA3	SLRA2	SLRA1	SLRA0
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 – SLRAn 各个 Rx[7:0]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

18.4.22. SLRCONB

名称： SLRCONB
偏移量： 0xF11

压摆率控制寄存器

位	7	6	5	4	3	2	1	0
	SLRB7	SLRB6	SLRB5	SLRB4	SLRB3	SLRB2	SLRB1	SLRB0
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 – SLRBn 各个 Rx[7:0]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

18.4.23. SLRCONC

名称： SLRCONC
偏移量： 0xF19

压摆率控制寄存器

位	7	6	5	4	3	2	1	0
	SLRC7	SLRC6	SLRC5	SLRC4	SLRC3	SLRC2	SLRC1	SLRC0
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 – SLRCn 各个 Rx[7:0]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

18.4.24. INLVLA

名称： INLVLA
偏移量： 0xF08

输入电平控制寄存器

位	7	6	5	4	3	2	1	0
	INLVLA7	INLVLA6	INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0
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 – INLVLAn 各个 Rx[7:0]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

18.4.25. INLVLB

名称： INLVLB
偏移量： 0xF10

输入电平控制寄存器

位	7	6	5	4	3	2	1	0
	INLVLB7	INLVLB6	INLVLB5	INLVLB4	INLVLB3	INLVLB2	INLVLB1	INLVLB0
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 – INLVLBn 各个 Rx[7:0]引脚的输入电平选择
注：INLVLB2/INLVLB1：在 MSSP 输入选择这些引脚且已使能 I^2C 模式时，引脚读取 I^2C ST 输入。

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

18.4.26. INLVLC

名称： INLVLC
偏移量： 0xF18

输入电平控制寄存器

位	7	6	5	4	3	2	1	0
	INLVLC7	INLVLC6	INLVLC5	INLVLC4	INLVLC3	INLVLC2	INLVLC1	INLVLC0
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 – INLVLCn 各个 Rx[7:0]引脚的输入电平选择
注：INLVLC4/INLVLC3：在 MSSP 输入选择这些引脚且已使能 I^2C 模式时，引脚读取 I^2C ST 输入。

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

18.4.27. INLVLE

名称： INLVLE
偏移量： 0xF25

输入电平控制寄存器

位	7	6	5	4	3	2	1	0
					INLVLE3			
访问					R/W			
复位					1			

Bit 3 – INLVLE3 各个 Rx[7:0]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

18.5. 寄存器汇总——输入/输出

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 … 0x0F07	保留									
0x0F08	INLVLA	7:0	INLVLA7	INLVLA6	INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0
0x0F09	SLRCONA	7:0	SLRA7	SLRA6	SLRA5	SLRA4	SLRA3	SLRA2	SLRA1	SLRA0
0x0F0A	ODCONA	7:0	ODCA7	ODCA6	ODCA5	ODCA4	ODCA3	ODCA2	ODCA1	ODCA0
0x0F0B	WPUA	7:0	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
0x0F0C	ANSELA	7:0	ANSELA7	ANSELA6	ANSELA5	ANSELA4	ANSELA3	ANSELA2	ANSELA1	ANSELA0
0x0F0D … 0x0F0F	保留									
0x0F10	INLVLB	7:0	INLVLB7	INLVLB6	INLVLB5	INLVLB4	INLVLB3	INLVLB2	INLVLB1	INLVLB0
0x0F11	SLRCONB	7:0	SLRB7	SLRB6	SLRB5	SLRB4	SLRB3	SLRB2	SLRB1	SLRB0
0x0F12	ODCONB	7:0	ODCB7	ODCB6	ODCB5	ODCB4	ODCB3	ODCB2	ODCB1	ODCB0
0x0F13	WPUB	7:0	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
0x0F14	ANSELB	7:0	ANSELB7	ANSELB6	ANSELB5	ANSELB4	ANSELB3	ANSELB2	ANSELB1	ANSELB0
0x0F15 … 0x0F17	保留									
0x0F18	INLVLC	7:0	INLVLC7	INLVLC6	INLVLC5	INLVLC4	INLVLC3	INLVLC2	INLVLC1	INLVLC0
0x0F19	SLRCONC	7:0	SLRC7	SLRC6	SLRC5	SLRC4	SLRC3	SLRC2	SLRC1	SLRC0
0x0F1A	ODCONC	7:0	ODCC7	ODCC6	ODCC5	ODCC4	ODCC3	ODCC2	ODCC1	ODCC0
0x0F1B	WPUC	7:0	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
0x0F1C	ANSELC	7:0	ANSELC7	ANSELC6	ANSELC5	ANSELC4	ANSELC3	ANSELC2	ANSELC1	ANSELC0
0x0F1D … 0x0F24	保留									
0x0F25	INLVLE	7:0					INLVLE3			
0x0F26 … 0x0F27	保留									
0x0F28	WPUE	7:0					WPUE3			
0x0F29 … 0x0F81	保留									
0x0F82	LATA	7:0	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0
0x0F83	LATB	7:0	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
0x0F84	LATC	7:0	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0
0x0F85 … 0x0F86	保留									
0x0F87	TRISA	7:0	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
0x0F88	TRISB	7:0	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
0x0F89	TRISC	7:0	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
0x0F8A … 0x0F8B	保留									
0x0F8C	PORTA	7:0	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
0x0F8D	PORTB	7:0	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
0x0F8E	PORTC	7:0	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
0x0F8F	保留									
0x0F90	PORTE	7:0					RE3			

19. 电平变化中断

19.1. 特性

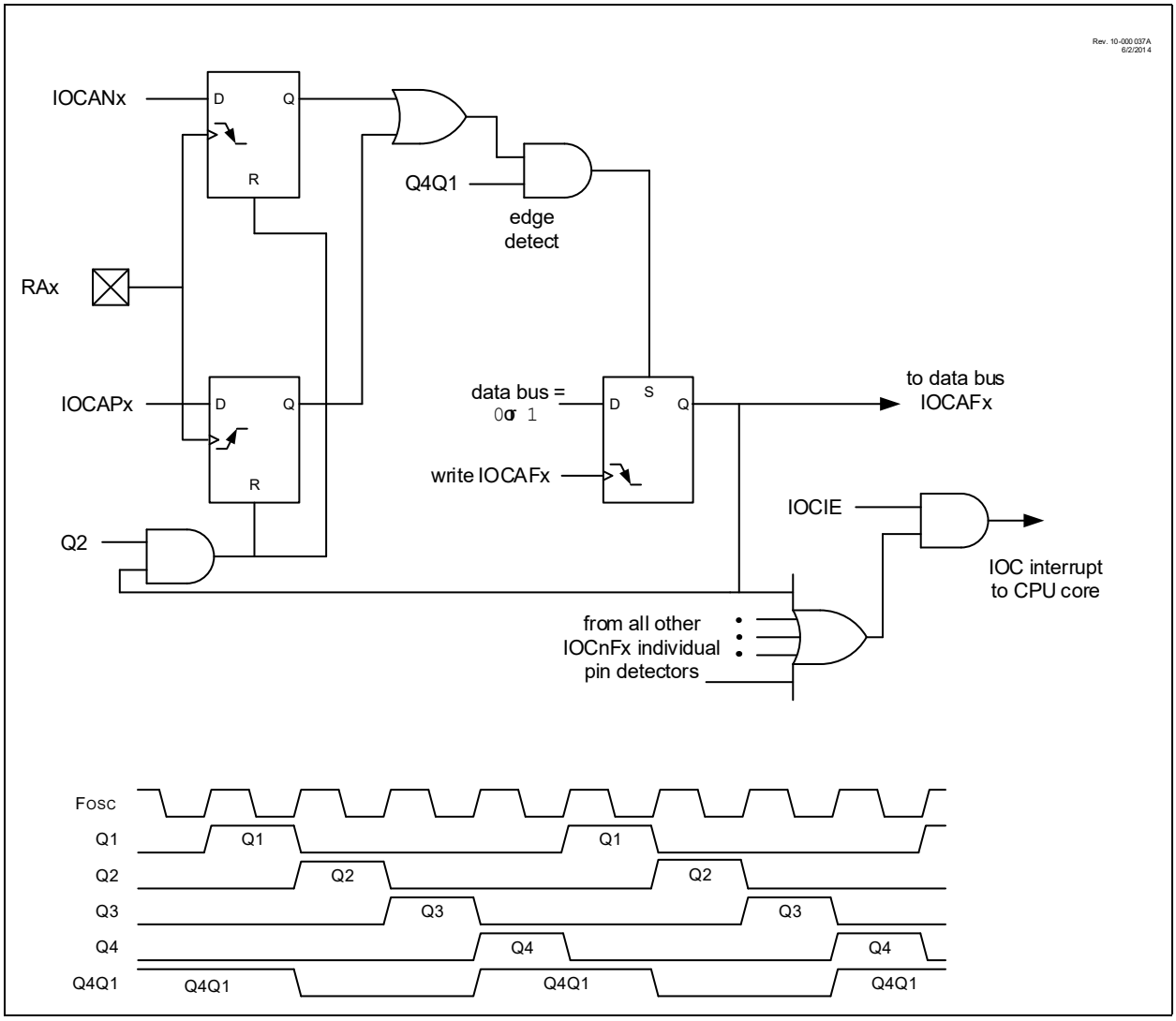
- 电平变化中断允许
- 独立的引脚配置
- 上升沿和下降沿检测
- 独立的引脚中断标志

19.2. 概述

在 CN2710 系列器件上，PORTA、PORTB 和 PORTC 的所有引脚以及 PORTE 的 RE3 引脚在配置后均可用作电平变化中断（IOC）引脚。可以通过检测具有上升沿或下降沿的信号而产生中断。任意一个端口引脚或端口引脚组合都可以配置为产生中断。

19.3. 框图

图 19-1. 电平变化中断框图（以 PORTA 为例）



19.4. 允许模块中断

要允许各个端口引脚产生中断，PIE0 寄存器的 IOCIE 位必须置 1。如果 IOCIE 位被禁止，在引脚上仍然会发生边沿检测，但不会产生中断。

19.5. 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，需要将 IOCxP 寄存器的相关位置 1。要允许引脚检测下降沿，需要将 IOCxN 寄存器的相关位置 1。

通过分别将 IOCxP 和 IOCxN 寄存器的相关位置 1，一个引脚可以配置为同时检测上升沿和下降沿。

19.6. 中断标志

是对应于相关端口的电平变化中断引脚的状态标志。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。PIR0 寄存器的 IOCIF 位影响所有 IOCAFx、IOCBFx、IOCCFx 和 IOCEF3 位的状态。

19.7. 清零中断标志

各状态标志位（IOCAFx、IOCBFx、IOCCFx 和 IOCEF3）可通过将其复位为零进行清零。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。

为了确保在清零标志时不会丢失任何已检测到的边沿，必须仅执行可屏蔽已知更改位的逻辑与操作。以下序列是一个说明必须执行何种操作的示例。

例 19-1. 清零中断标志（以 PORTA 为例）

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

19.8. 休眠模式下的操作

如果 IOCIE 位置 1，电平变化中断序列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。

19.9. 寄存器定义：电平变化中断控制

19.9.1. IOCAF

名称： IOCAF
偏移量： 0xF05

PORTA 电平变化中断标志寄存器示例

位	7	6	5	4	3	2	1	0
	IOCAF7	IOCAF6	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0
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 – IOCAFn 电平变化中断标志位

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

19.9.2. IOCBF

名称： IOCBF
偏移量： 0xF0D

PORTB 电平变化中断标志寄存器示例

位	7	6	5	4	3	2	1	0
	IOCBF7	IOCBF6	IOCBF5	IOCBF4	IOCBF3	IOCBF2	IOCBF1	IOCBF0
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 – IOCBFn 电平变化中断标志位

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

19.9.3. IOCCF

名称： IOCCF
偏移量： 0xF15

PORTC 电平变化中断标志寄存器

位	7	6	5	4	3	2	1	0
	IOCCF7	IOCCF6	IOCCF5	IOCCF4	IOCCF3	IOCCF2	IOCCF1	IOCCF0
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 – IOCCFn 电平变化中断标志位

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

19.9.4. IOCEF

名称： IOCEF
偏移量： 0xF22

PORTE 电平变化中断标志寄存器

位	7	6	5	4	3	2	1	0
					IOCEF3			
访问					R/W/HS			
复位					0			

Bit 3 – IOCEF3 PORTE 电平变化中断标志位(1)

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

注：

1. 如果 MCLRE = 1 或 LVP = 1，则 RE3 端口功能将被禁止，RE3 上的 IOC 不可用。

19.9.5. IOCAN

名称： IOCAN
偏移量： 0xF06

电平变化中断负边沿寄存器示例

位	7	6	5	4	3	2	1	0
	IOCAN7	IOCAN6	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0
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 – IOCANn 电平变化中断负边沿允许位

值	说明
1	在 IOCA 引脚上对于负边沿允许电平变化中断。检测到边沿时将相关的状态位和中断标志置 1。
0	禁止相关引脚的电平变化中断。

19.9.6. IOCBN

名称： IOCBN
偏移量： 0xF0E

电平变化中断负边沿寄存器示例

位	7	6	5	4	3	2	1	0
	IOCBN7	IOCBN6	IOCBN5	IOCBN4	IOCBN3	IOCBN2	IOCBN1	IOCBN0
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 – IOCBNn 电平变化中断负边沿使能位

值	说明
1	在 IOCA 引脚上对于负边沿允许电平变化中断。检测到边沿时将相关的状态位和中断标志置 1。
0	禁止相关引脚的电平变化中断。

19.9.7. IOCCN

名称： IOCCN
偏移量： 0xF16

电平变化中断负边沿寄存器示例

位	7	6	5	4	3	2	1	0
	IOCCN7	IOCCN6	IOCCN5	IOCCN4	IOCCN3	IOCCN2	IOCCN1	IOCCN0
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 – IOCCNn 电平变化中断负边沿使能位

值	说明
1	在 IOCA 引脚上对于负边沿允许电平变化中断。检测到边沿时将相关的状态位和中断标志置 1。
0	禁止相关引脚的电平变化中断。

19.9.8. IOCEN

名称： IOCEN
偏移量： 0xF23

电平变化中断负边沿寄存器示例

位	7	6	5	4	3	2	1	0
					IOCEN3			
访问					R/W			
复位					0			

Bit 3 – IOCEN3 电平变化中断负边沿允许位(1)

值	说明
1	在 IOCA 引脚上对于负边沿允许电平变化中断。检测到边沿时将相关的状态位和中断标志置 1。
0	禁止相关引脚的电平变化中断。

注：

1. 如果 MCLRE = 1 或 LVP = 1，则 RE3 引脚端口功能将被禁止，RE3 上的 IOC 不可用。

19.9.9. IOCAP

名称： IOCAP
偏移量： 0xF07

电平变化中断正边沿寄存器

位	7	6	5	4	3	2	1	0
	IOCAP7	IOCAP6	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0
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 – IOCAPn 电平变化中断正边沿使能位

值	说明
1	在 IOCA 引脚上对于正边沿允许电平变化中断。检测到边沿时将相关的状态位和中断标志置 1。
0	禁止相关引脚的电平变化中断。

19.9.10. IOCBP

名称： IOCBP
偏移量： 0xF0F

电平变化中断正边沿寄存器

位	7	6	5	4	3	2	1	0
	IOCBP7	IOCBP6	IOCBP5	IOCBP4	IOCBP3	IOCBP2	IOCBP1	IOCBP0
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 – IOCBPn 电平变化中断正边沿使能位

值	说明
1	在 IOCB 引脚上对于正边沿允许电平变化中断。检测到边沿时将相关的状态位和中断标志置 1。
0	禁止相关引脚的电平变化中断。

19.9.11. IOCCP

名称： IOCCP
偏移量： 0xF17

电平变化中断正边沿寄存器

位	7	6	5	4	3	2	1	0
	IOCCP7	IOCCP6	IOCCP5	IOCCP4	IOCCP3	IOCCP2	IOCCP1	IOCCP0
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 – IOCCPn 电平变化中断正边沿使能位

值	说明
1	在 IOCC 引脚上对于正边沿允许电平变化中断。检测到边沿时将相关的状态位和中断标志置 1。
0	禁止相关引脚的电平变化中断。

19.9.12. IOCEP

名称： IOCEP
偏移量： 0xF24

电平变化中断（正）边沿寄存器

位	7	6	5	4	3	2	1	0
					IOCEP3			
访问					R/W			
复位					0			

Bit 3 – IOCEP3 电平变化中断正边沿允许位(1)

值	说明
1	在 IOCE 引脚上对于正边沿允许电平变化中断。检测到边沿时将相关的状态位和中断标志置 1。
0	禁止相关引脚的电平变化中断。

注：

1. 如果 MCLRE = 1 或 LVP = 1，则 RE3 引脚端口功能将被禁止，RE3 上的 IOC 不可用。

19.10. 寄存器汇总——电平变化中断

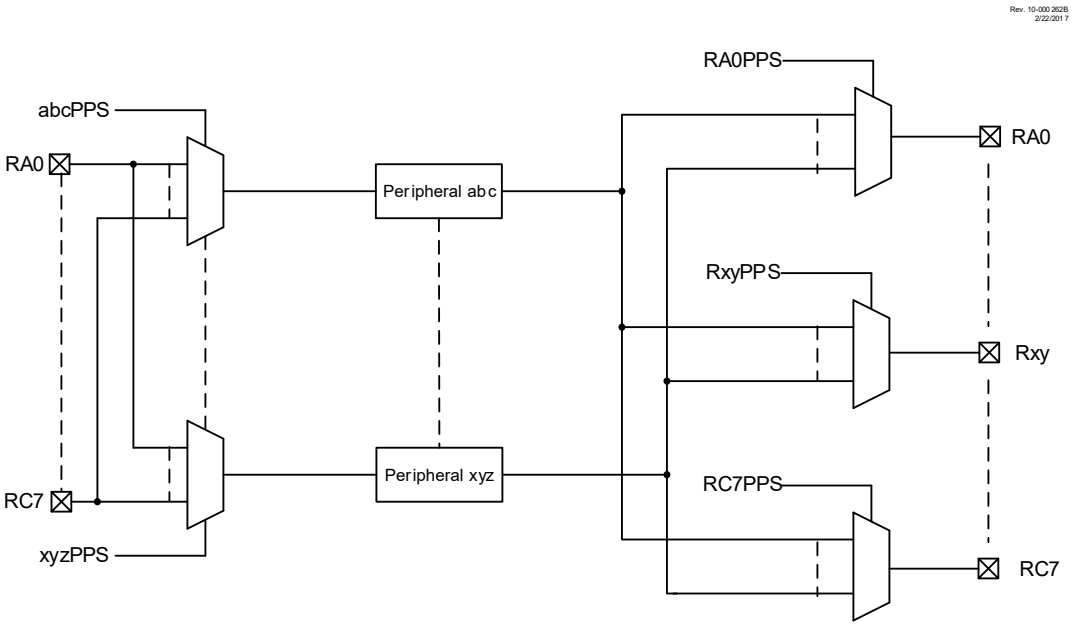
偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0F04	保留									
0x0F05	IOCAF	7:0	IOCAF7	IOCAF6	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0
0x0F06	IOCAN	7:0	IOCAN7	IOCAN6	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0
0x0F07	IOCAP	7:0	IOCAP7	IOCAP6	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0
0x0F08 ... 0x0F0C	保留									
0x0F0D	IOCBF	7:0	IOCBF7	IOCBF6	IOCBF5	IOCBF4	IOCBF3	IOCBF2	IOCBF1	IOCBF0
0x0F0E	IOCBN	7:0	IOCBN7	IOCBN6	IOCBN5	IOCBN4	IOCBN3	IOCBN2	IOCBN1	IOCBN0
0x0F0F	IOCBP	7:0	IOCBP7	IOCBP6	IOCBP5	IOCBP4	IOCBP3	IOCBP2	IOCBP1	IOCBP0
0x0F10 ... 0x0F14	保留									
0x0F15	IOCCF	7:0	IOCCF7	IOCCF6	IOCCF5	IOCCF4	IOCCF3	IOCCF2	IOCCF1	IOCCF0
0x0F16	IOCCN	7:0	IOCCN7	IOCCN6	IOCCN5	IOCCN4	IOCCN3	IOCCN2	IOCCN1	IOCCN0
0x0F17	IOCCP	7:0	IOCCP7	IOCCP6	IOCCP5	IOCCP4	IOCCP3	IOCCP2	IOCCP1	IOCCP0
0x0F18 ... 0x0F21	保留									
0x0F22	IOCEF	7:0					IOCEF3			
0x0F23	IOCEN	7:0					IOCEN3			
0x0F24	IOCEP	7:0					IOCEP3			

20. PPS——外设引脚选择模块

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。所有模拟输入和输出均固定连接至它们所分配的引脚。输入和输出选择是独立的，如下图所示。

外设输入使用外设 xxxPPS 寄存器选择，外设输出使用 PORT RxyPPS 寄存器选择。例如，要选择 PORTC[7]作为 EUSART RX 输入，可以将 RXxPPS 设为 0x17（如输入表所示）；要选择 PORTC[6]作为 EUSART TX 输出，可以将 RC6PPS 设为 0x09（如输出表所示）。

图 20-1. PPS 简化框图



20.1. PPS 输入

每个外设均具有一个用于选择外设输入引脚的 PPS 寄存器。虽然每个外设均具有自己的 PPS 输入选择寄存器，但每个外设的选择是相同的，如 xxxPPS 所示。并非所有端口均可用于输入，具体如 *“PPS 输入选择寄存器详细信息”* 表所示。

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器。

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 = INT 代表 INTPPS 寄存器。

表 20-1. PPS 输入选择寄存器详细信息

外设	PPS 输入寄存器	POR 时的默认引脚选择	POR 时的寄存器复位值	可用的输入端口		
中断 0	INT0PPS	RB0	0x08	A	B	—
中断 1	INT1PPS	RB1	0x09	A	B	—
中断 2	INT2PPS	RB2	0x0A	A	B	—
Timer0 时钟	T0CKIPPS	RA4	0x04	A	B	—
Timer1 时钟	T1CKIPPS	RC0	0x10	A	—	C
Timer1 门控	T1GPPS	RB5	0x0D	—	B	C

表 20-1. PPS 输入选择寄存器详细信息（续）

外设	PPS 输入寄存器	POR 时的默认引脚选择	POR 时的寄存器复位值	可用的输入端口		
Timer3 时钟	T3CKIPPS	RC0	0x10	—	B	C
Timer3 门控	T3GPPS	RC0	0x10	A	—	C
Timer5 时钟	T5CKIPPS	RC2	0x12	A	—	C
Timer5 门控	T5GPPS	RB4	0x0C	—	B	C
Timer2 时钟	T2INPPS	RC3	0x13	A	—	C
Timer4 时钟	T4INPPS	RC5	0x15	—	B	C
Timer6 时钟	T6INPPS	RB7	0x0F	—	B	C
ADC 转换触发器	ADACTPPS	RB4	0x0C	—	B	C
CCP1	CCP1PPS	RC2	0x12	—	B	C
CCP2	CCP2PPS	RC1	0x11	—	B	C
CWG	CWG1PPS	RB0	0x08	—	B	C
DSM 载波低电平	MDCARLPPS	RA3	0x03	A	—	C
DSM 载波高电平	MDCARHPPS	RA4	0x04	A	—	C
DSM 源	MDSRCPPS	RA5	0x05	A	—	C
EUSART1 接收	RX1PPS	RC7	0x17	—	B	C
EUSART1 时钟	CK1PPS	RC6	0x16	—	B	C
EUSART2 接收(1)	RX2PPS	RB7	0x0F	—	B	C
EUSART2 时钟(1)	CK2PPS	RB6	0x0E	—	B	C
MSSP1 时钟	SSP1CLKPPS	RC3	0x13	—	B	C
MSSP1 数据	SSP1DATPPS	RC4	0x14	—	B	C
MSSP1 从选择	SSP1SSPPS	RA5	0x05	A	—	C
MSSP2 时钟(1)	SSP2CLKPPS	RB1	0x09	—	B	C
MSSP2 数据(1)	SSP2DATPPS	RB2	0x0A	—	B	C
MSSP2 从选择(1)	SSP2SSPPS	RB0	0x08	—	B	C
CLCIN0(1)	CLCIN0PPS	RA0	0x00	A	—	C
CLCIN1(1)	CLCIN1PPS	RA1	0x01	A	—	C
CLCIN2(1)	CLCIN2PPS	RB6	0x0E	—	B	C
CLCIN3(1)	CLCIN3PPS	RB7	0x0F	—	B	C
CLCIN4(1)	CLCIN4PPS	RA0	0x00	A	—	C
CLCIN5(1)	CLCIN5PPS	RA1	0x01	A	—	C
CLCIN6(1)	CLCIN6PPS	RB6	0x0E	—	B	C
CLCIN7(1)	CLCIN7PPS	RB7	0x0F	—	B	C

注：

1. CN2510 器件上未提供。

注：

1. 部分焊盘配置为 I^2C 逻辑电平；时钟和数据信号可分配给任何这些引脚。分配给其他引脚（如 RA5）可以工作，但逻辑电平将为通过 INLVL 寄存器选择的标准 TTL/ST。

表 20-2. PPS 输入寄存器值

所需输入引脚	要写入寄存器的值
RA0	0x00
RA1	0x01
RA2	0x02
RA3	0x03

表 20-2. PPS 输入寄存器值 （续）

所需输入引脚	要写入寄存器的值
RA4	0x04
RA5	0x05
RB4	0x0C
RB5	0x0D
RB6	0x0E
RB7	0x0F
RC0	0x10
RC1	0x11
RC2	0x12
RC3	0x13
RC4	0x14
RC5	0x15
RC6	0x16
RC7	0x17

20.2. PPS 输出

每个 I/O 引脚均具有一个用于选择引脚输出源的 PPS 寄存器。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。这些外设包括：

- EUSART（同步操作）
- MSSP（I^2C）

虽然每个引脚均具有自己的 PPS 外设选择寄存器，但每个引脚的选择是相同的，如 RxyPPS 所示。

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。

下表详细列出了每个外设的输出路由选项。

表 20-3. 外设 PPS 输出选择代码

RxyPPS	引脚 Rxy 输出源	可用的输出端口		
0x1F	CLC8OUT(1)	—	B	C
0x1E	CLC7OUT(1)	—	B	C
0x1D	CLC6OUT(1)	A	—	C
0x1C	CLC5OUT(1)	A	—	C
0x1B	CLC4OUT(1)	—	B	C
0x1A	CLC3OUT(1)	—	B	C
0x19	CLC2OUT(1)	A	—	C
0x18	CLC1OUT(1)	A	—	C
0x17	ADGRDB	A	—	C
0x16	ADGRDA	A	—	C
0x15	DSM	A	—	C
0x14	CLKR	—	B	C
0x13	TMR0	—	B	C
0x12	MSSP2（SDO/SDA）(1)	—	B	C
0x11	MSSP2（SCK/SCL）(1)	—	B	C

表 20-3. 外设 PPS 输出选择代码 （续）

RxyPPS	引脚 Rxy 输出源	可用的输出端口		
0x10	MSSP1（SDO/SDA）	—	B	C
0x0F	MSSP1（SCK/SCL）	—	B	C
0x0E	C2	A	—	C
0x0D	C1	A	—	C
0x0C	EUSART2（DT）(1)	—	B	C
0x0B	EUSART2（TX/CK）(1)	—	B	C
0x0A	EUSART1（DT）	—	B	C
0x09	EUSART1（TX/CK）	—	B	C
0x08	PWM4	A	—	C
0x07	PWM3	A	—	C
0x06	CCP2	—	B	C
0x05	CCP1	—	B	C
0x04	CWG1D	—	B	C
0x03	CWG1C	—	B	C
0x02	CWG1B	—	B	C
0x01	CWG1A	—	B	C
0x00	LATxy	A	B	C

注：

1. CN2510 器件上未提供。

20.3. 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。具有双向信号的外设包括：

- EUSART（DT/RXxPPS 和 TX/CKxPPS 引脚用于同步操作）
- MSSP（I^2C SDA/SSPxDATPPS 和 SCL/SSPxCLKPPS）

重要：I^2C 默认输入引脚以及有限数量的其他备用引脚与 I^2C 和 SMBus 兼容。时钟和数据信号可输送到任何引脚，但是不具有 I^2C 兼容性的引脚将以标准 TTL/ST 逻辑电平（由 INLVL 寄存器选择）工作。要确定哪些引脚与 I^2C 和 SMBus 兼容，请参见每个端口的 INLVL 寄存器。

20.4. PPS 锁定

PPS 包含了一种模式，在该模式下可以锁定所有输入和输出选择，以防止意外的更改。PPS 选择通过将 PPSLOCK 寄存器的 PPSLOCKED 位置 1 来进行锁定。置 1 和清零该位需要一个特殊的序列作为额外的预防措施，以防止意外的更改。下面给出了置 1 和清零 PPSLOCKED 位的示例。

例 20-1. PPS 锁定序列

```
; Disable interrupts:
   BCF    INTCON,GIE
; Bank to PPSLOCK register
   BANKSEL PPSLOCK
   MOVLW   55h
; Required sequence, next 4 instructions
   MOVWF   PPSLOCK
   MOVLW   AAh
   MOVWF   PPSLOCK
; Set PPSLOCKED bit to disable writes
; Only a BSF instruction will work
```

```
   BSF     PPSLOCK,PPSLOCKED
; Enable Interrupts
   BSF     INTCON,GIE
```

例 20-2. PPS 解锁序列

```
; Disable interrupts:
   BCF     INTCON,GIE
; Bank to PPSLOCK register
   BANKSEL PPSLOCK
   MOVLW   55h
; Required sequence, next 4 instructions
   MOVWF   PPSLOCK
   MOVLW   AAh
   MOVWF   PPSLOCK
; Clear PPSLOCKED bit to enable writes
; Only a BCF instruction will work
   BCF     PPSLOCK,PPSLOCKED
; Enable Interrupts
   BSF     INTCON,GIE
```

20.5. PPS 单向锁定

使用 PPS1WAY 配置位，可以锁定 PPS 设置。当该位置 1 时，PPSLOCKED 位只能在器件复位后清零并置 1 次。这使得可以清零 PPSLOCKED 位，从而可以在初始化过程中进行输入和输出选择。在完成所有选择之后将 PPSLOCKED 位置 1 时，它将一直保持置 1，直到下一个器件复位事件之后才能清零。

20.6. 休眠期间的操作

PPS 输入和输出选择不会受休眠影响。

20.7. 复位的影响

器件上电复位（POR）会将所有 PPS 输入和输出选择清除为其默认值。所有其他复位会将这些选择保留不变。 *“PPS 输入选择寄存器详细信息”* 表中列出了默认的输入选择。PPS1WAY 也会被移除。

20.8. 寄存器汇总——PPS

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0E1E	保留									
0x0E1F	CLCIN0PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0E20	CLCIN1PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0E21	CLCIN2PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0E22	CLCIN3PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0E23	CLCIN4PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0E24	CLCIN5PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0E25	CLCIN6PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0E26	CLCIN7PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0E27 ... 0x0E87	保留									
0x0E88	RX2PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0E89	CK2PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0E8A	SSP2CLKPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0E8B	SSP2DATPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0E8C	SSP2SSPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0E8D ... 0x0E9A	保留									
0x0E9B	PPSLOCK	7:0								PPSLOCKED
0x0E9C	INT0PPS	7:0					PORT	PIN[2:0]		
0x0E9D	INT1PPS	7:0					PORT	PIN[2:0]		
0x0E9E	INT2PPS	7:0					PORT	PIN[2:0]		
0x0E9F	T0CKIPPS	7:0					PORT	PIN[2:0]		
0x0EA0	T1CKIPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA1	T1GPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA2	T3CKIPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA3	T3GPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA4	T5CKIPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA5	T5GPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA6	T2INPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA7	T4INPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA8	T6INPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA9	ADACTPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAA	CCP1PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAB	CCP2PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAC	CWG1PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAD	MDCARLPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAE	MDCARHPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAF	MDSRCPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB0	RX1PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB1	CK1PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB2	SSP1CLKPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB3	SSP1DATPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB4	SSP1SSPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB5 ... 0x0EE1	保留									
0x0EE2	RA0PPS	7:0						PPS[4:0]		
0x0EE3	RA1PPS	7:0						PPS[4:0]		
0x0EE4	RA2PPS	7:0						PPS[4:0]		
0x0EE5	RA3PPS	7:0						PPS[4:0]		
0x0EE6	RA4PPS	7:0						PPS[4:0]		
0x0EE7	RA5PPS	7:0						PPS[4:0]		
0x0EE8	RA6PPS	7:0						PPS[4:0]		
0x0EE9	RA7PPS	7:0						PPS[4:0]		

寄存器汇总——PPS（续）

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x0EEA	RB0PPS	7:0						PPS[4:0]		
0x0EEB	RB1PPS	7:0						PPS[4:0]		
0x0EEC	RB2PPS	7:0						PPS[4:0]		
0x0EED	RB3PPS	7:0						PPS[4:0]		
0x0EEE	RB4PPS	7:0						PPS[4:0]		
0x0EEF	RB5PPS	7:0						PPS[4:0]		
0x0EF0	RB6PPS	7:0						PPS[4:0]		
0x0EF1	RB7PPS	7:0						PPS[4:0]		
0x0EF2	RC0PPS	7:0						PPS[4:0]		
0x0EF3	RC1PPS	7:0						PPS[4:0]		
0x0EF4	RC2PPS	7:0						PPS[4:0]		
0x0EF5	RC3PPS	7:0						PPS[4:0]		
0x0EF6	RC4PPS	7:0						PPS[4:0]		
0x0EF7	RC5PPS	7:0						PPS[4:0]		
0x0EF8	RC6PPS	7:0						PPS[4:0]		
0x0EF9	RC7PPS	7:0						PPS[4:0]		
0x0EFA	RD0PPS	7:0						PPS[4:0]		
0x0EFB	RD1PPS	7:0						PPS[4:0]		
0x0EFC	RD2PPS	7:0						PPS[4:0]		
0x0EFD	RD3PPS	7:0						PPS[4:0]		
0x0EFE	RD4PPS	7:0						PPS[4:0]		
0x0EFF	RD5PPS	7:0						PPS[4:0]		
0x0F00	RD6PPS	7:0						PPS[4:0]		
0x0F01	RD7PPS	7:0						PPS[4:0]		
0x0F02	RE0PPS	7:0						PPS[4:0]		
0x0F03	RE1PPS	7:0						PPS[4:0]		
0x0F04	RE2PPS	7:0						PPS[4:0]		

20.9. 寄存器定义：PPS 输入和输出选择

20.9.1. 外设 xxx 输入选择

名称： xxxPPS

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

Bit 4:3 – PORT[1:0] 外设 xxx 输入端口选择位
有关可用端口和默认引脚位置的列表，请参见 *“PPS 输入选择寄存器详细信息”* 表。

值	说明
10	PORTC
01	PORTB
00	PORTA

Bit 2:0 – PIN[2:0] 外设 xxx 输入引脚选择位

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）

20.9.2. 引脚 Rxy 输出源选择寄存器

名称： RxyPPS

重要：有关各个寄存器的偏移地址，请参见寄存器汇总——PPS 。

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

Bit 4:0 – RxyPPS[4:0] 引脚 Rxy 输出源选择位
有关源选择的详细信息，请参见 *“外设 PPS 输出选择代码”* 表。

20.9.3. PPS 锁定寄存器

名称： PPSLOCK
偏移量： 0xE9B

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

Bit 0 – PPSLOCKED PPS 锁定位

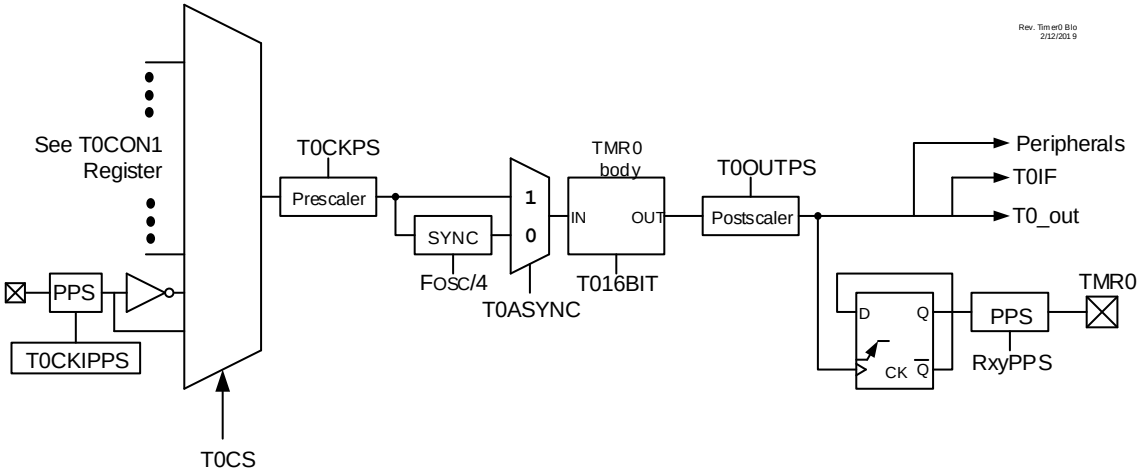
值	说明
1	PPS 已锁定。不能更改 PPS 选择。
0	PPS 未锁定。可以更改 PPS 选择。

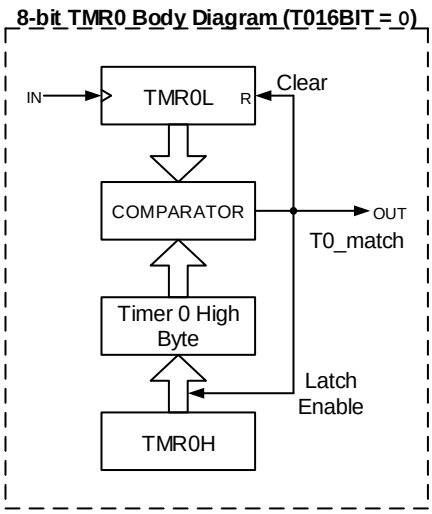
21. TMR0——Timer0 模块

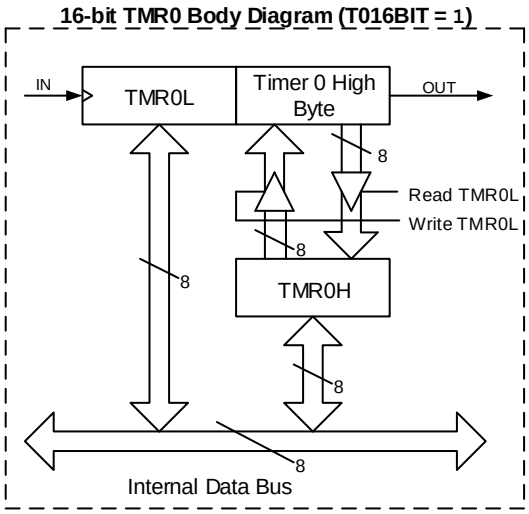
Timer0 模块具有以下特性：

- 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 21-1. Timer0 框图







21.1. 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。

21.1.1. 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 21-1 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

21.1.2. 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 21-1 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。

21.2. 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。

21.2.1. 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。

21.2.2. 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。

21.2.3. 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

21.2.4. 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

21.3. Timer0 输出和中断

21.3.1. 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。

21.3.2. 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。

21.3.3. 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

21.4. 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。

21.5. 寄存器定义：Timer0 控制

21.5.1. T0CON0

名称： T0CON0
偏移量： 0xFD4

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

21.5.2. T0CON1

名称： T0CON1
偏移量： 0xFD5

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 21-1. Timer0 时钟源选择

T0CS	时钟源
111	CLC1_OUT(1)
110	MFINTOSC（500 kHz）
101	SOSC
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（不反相）

注：
1. CN2510 器件上未提供。

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

21.5.3. TMR0H

名称： TMR0H
偏移量： 0xFD3

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

21.5.4. TMR0L

名称： TMR0L
偏移量： 0xFD2

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

21.6. 寄存器汇总——Timer0

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0FD1	保留									
0x0FD2	TMR0L	7:0	TMR0L[7:0]							
0x0FD3	TMR0H	7:0	TMR0H[7:0]							
0x0FD4	T0CON0	7:0	EN		OUT	MD16	OUTPS[3:0]			
0x0FD5	T0CON1	7:0	CS[2:0]			ASYNC	CKPS[3:0]			

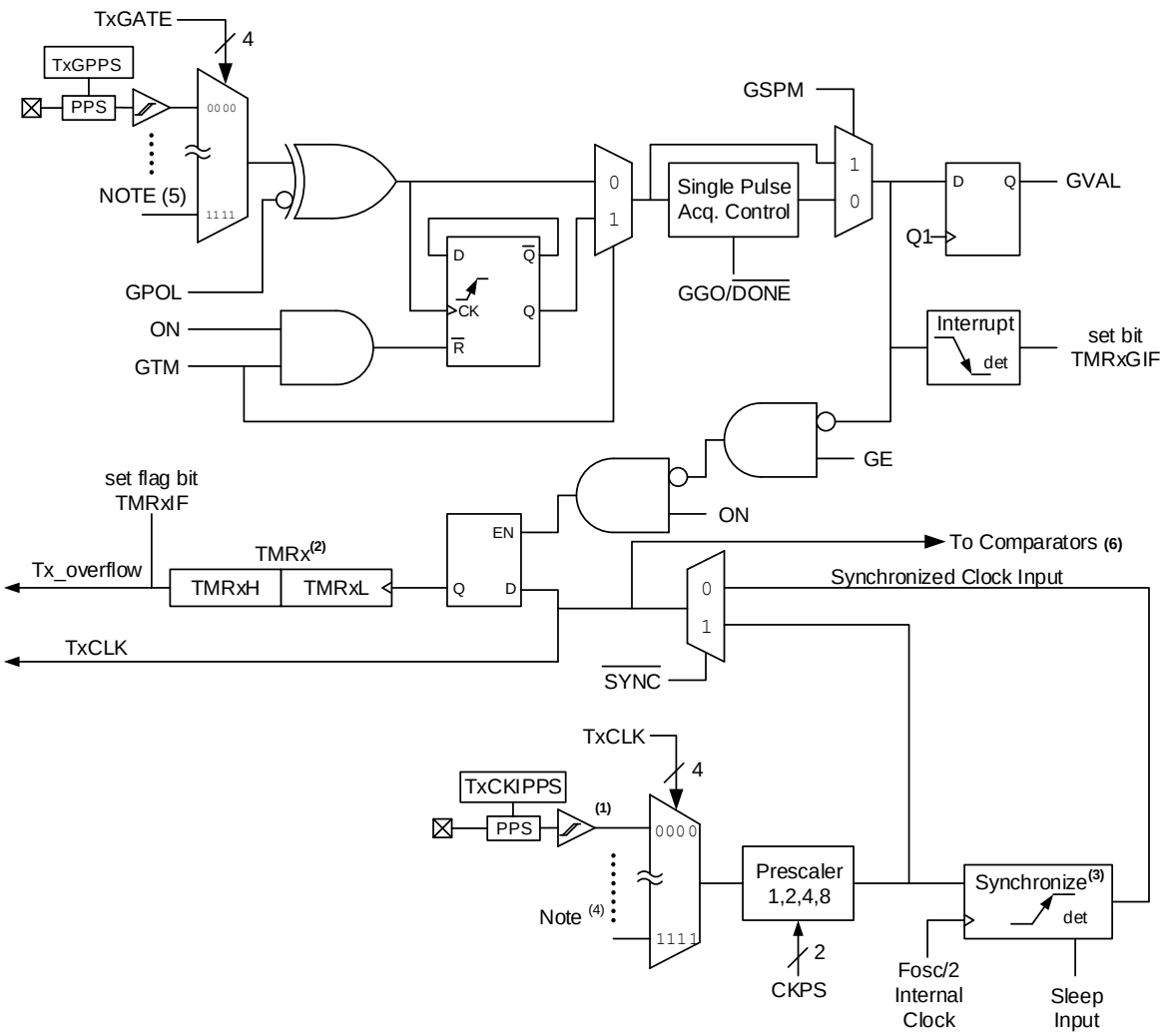
22. TMR1——带门控的 Timer1 模块

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 22-1. 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。

22.1. 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 22-1 列出了 Timer1 的使能选项。

表 22-1. 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

22.2. 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。

22.2.1. 内部时钟源

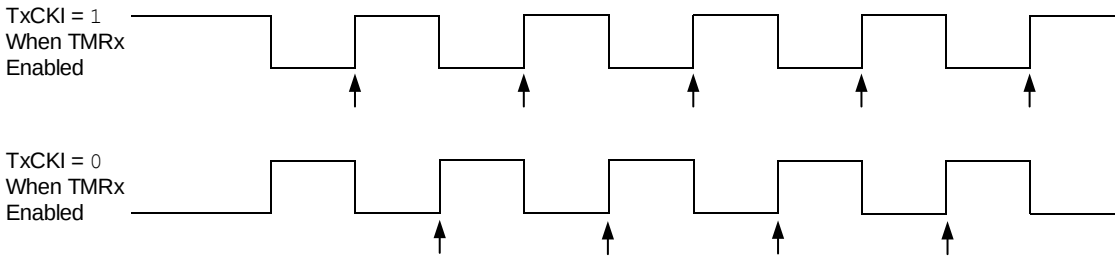
当选择内部时钟源时，TMRx 寄存器的递增频率将为 F_{OSC} 的整数倍（取决于 Timer1 预分频比）。

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 22-2. 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。

22.2.2. 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。

22.3. 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。

22.4. 辅助振荡器

引脚 SOSCI（输入）和引脚 SOSCO（放大器输出）之间内置了一个辅助低功耗 32.768 kHz 振荡器电路。该内部电路与外部 32.768 kHz 晶振结合使用。辅助振荡器并非专用于 Timer1，也可以用于其他模块。

可通过将 OSCEN 寄存器的 SOSCEN 位置 1 来使能该振荡器电路。可以通过 CS 位进行选择，将其用作 Timer1 时钟源之一。该振荡器将在休眠期间继续运行。

重要：在使用振荡器之前需要一段起振和稳定时间。因此，必须将 OSCEN 寄存器的 SOSCEN 置 1 并在经过一段合适的延时后才能使能 Timer1。需要执行软件检查，确认辅助振荡器已使能并准备就绪。这通过轮询辅助振荡器就绪状态位来实现。更多详细信息，请参见**“OSC——振荡器模块（带故障保护时钟监视器）”**一章。

22.5. 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。

22.5.1. 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。

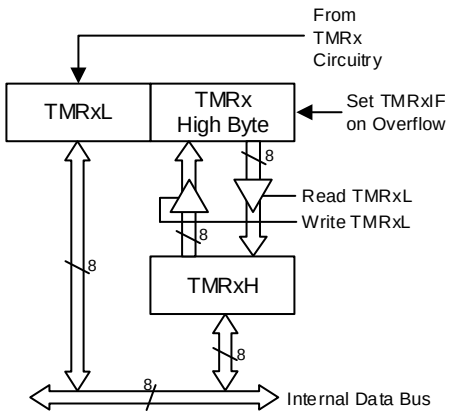
22.6. Timer1 16 位读/写模式

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 22-3）。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 22-3. Timer1 16 位读/写模式框图



22.7. 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。

22.7.1. Timer1 门控使能

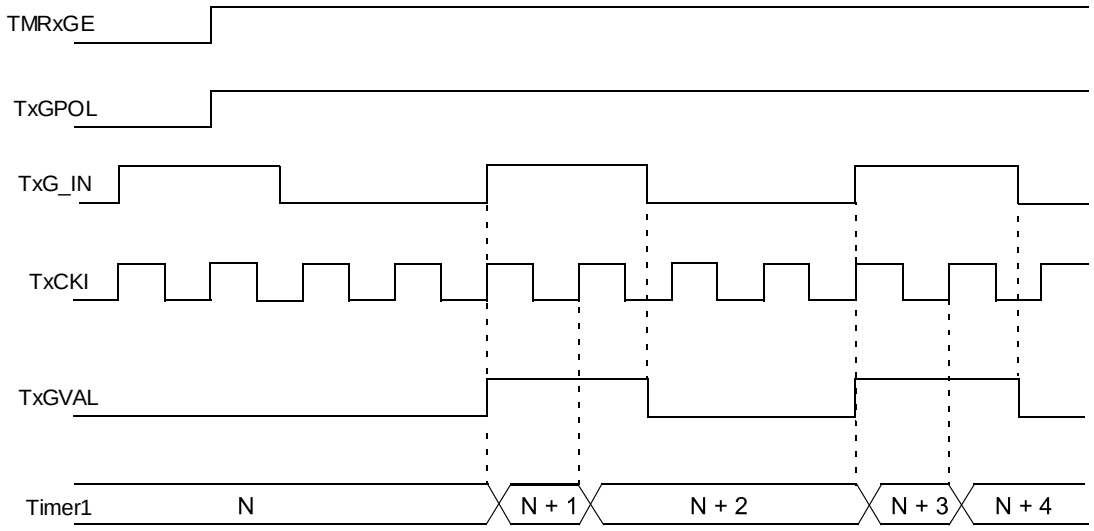
通过将 GE 位置 1 使能 Timer1 门控使能模式。使用 GPOL 位来配置 Timer1 门控使能模式的极性。

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 22-4。

表 22-2. 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 22-4. Timer1 门控使能模式



22.7.2. 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。

上述任何信号都可以用于触发门控。CMPx 的输出可与 Timer1 时钟同步，也可异步运行。更多信息，请参见**“CMP——比较器模块”**一章中的**“比较器输出同步”**一节。

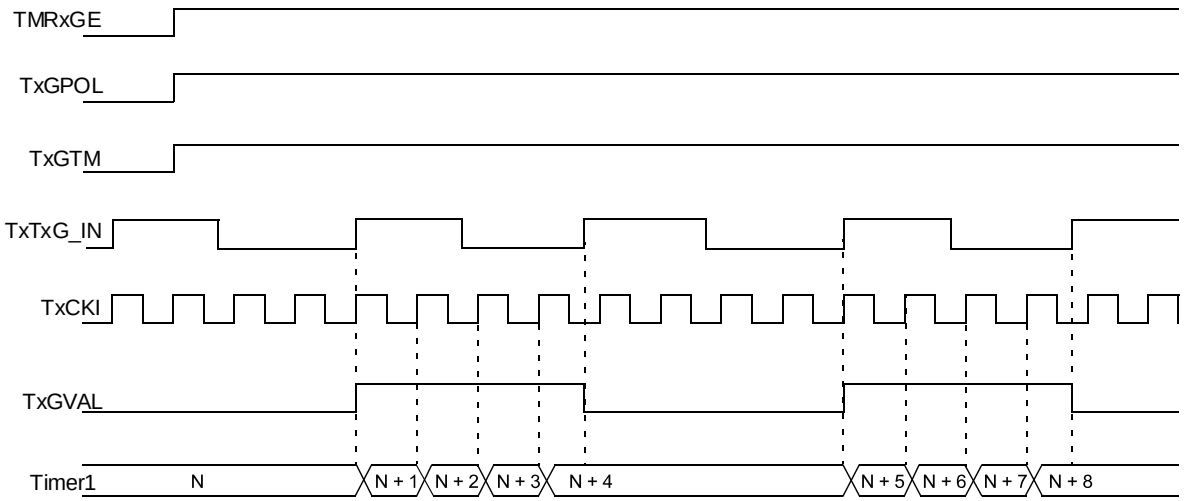
22.7.3. Timer1 门控翻转模式

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。

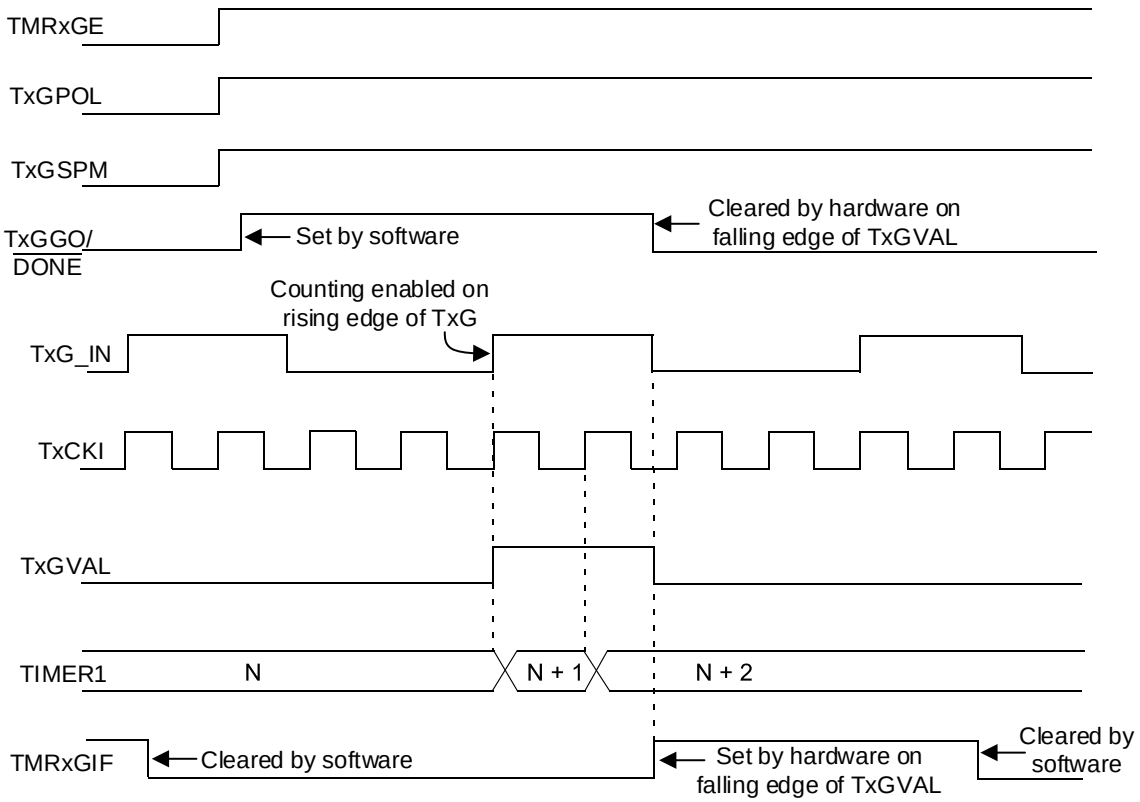
图 22-5. Timer1 门控翻转模式



22.7.4. Timer1 门控单脉冲模式

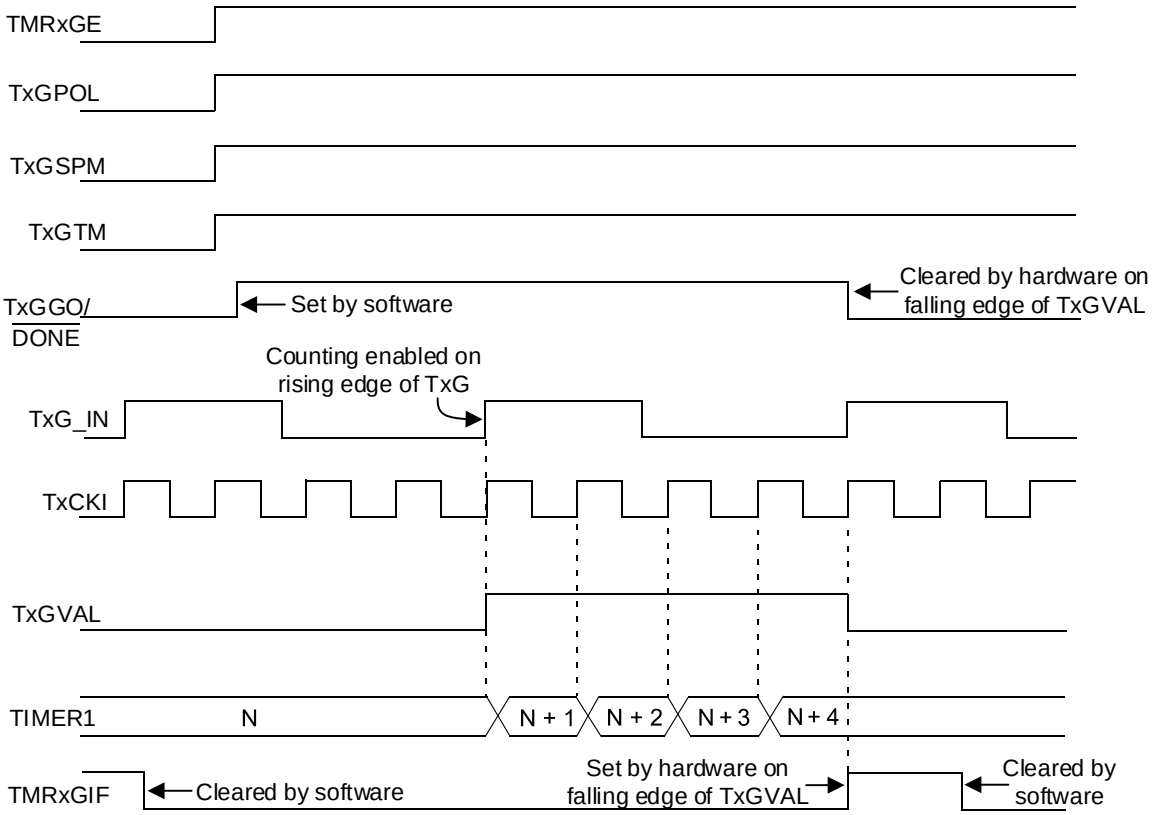
使能了 Timer1 门控单脉冲模式时，可以捕捉单脉冲门控事件。首先，通过将 GSPM 位置 1 来使能 Timer1 门控单脉冲模式。然后，必须将 GGO/DONE 置 1。Timer1 将在下一个递增沿完全使能。在脉冲的下一个后沿，GGO/$\overline{\text{DONE}}$ 位将自动清零。GGO/$\overline{\text{DONE}}$ 位再次由软件置 1 前，其他门控事件无法递增 Timer1。

图 22-6. 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 22-7. Timer1 门控单脉冲和翻转组合模式



22.7.5. 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。

22.7.6. 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，PIR 寄存器中的 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。

22.8. 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 Timer1 中断标志位将置 1。为允许计满返回时的中断，必须将以下位置 1：

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

通过在中断服务程序中将 TMRxIF 位清零（作为任务）清除中断。

重要：在允许中断前，必须将 TMRx 寄存器以及 TMRxIF 位清零。

22.9. 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。

22.10. 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。

22.11. 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。

22.12. 外设模块禁止

当外设未使用或无效时，可以通过将 PMD 寄存器中的模块禁止位置 1 来禁止该模块。这会将功耗降至最低。将 PMD 位置 1 可以使模块保持在复位状态，并断开模块的时钟源。Timer1 的模块禁止位（TMR1MD）位于 PMDx 寄存器中。更多信息，请参见**“PMD——外设模块禁止”**一章。

22.13. 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 22-3. Timer1 寄存器长位名称前缀

外设	位名称前缀
Timer1	T1
Timer3	T3
Timer5	T5

22.13.1. TxCON

名称： TxCON
偏移量： 0xFCE,0xFC8,0xFC2

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

22.13.2. TxGCON

名称： TxGCON
偏移量： 0xFCF,0xFC9,0xFC3

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

22.13.3. TxCLK

名称： TxCLK
偏移量： 0xFD1,0xFCB,0xFC5

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 22-4. 定时器时钟源

CS	时钟源		
	Timer1	Timer3	Timer5
11000-11111	**保留**		
10111	CLC8_OUT(1)		
10110	CLC7_OUT(1)		
10101	CLC6_OUT(1)		
10100	CLC5_OUT(1)		
10011	CLC4_OUT(1)		
10010	CLC3_OUT(1)		
10001	CLC2_OUT(1)		
10000	CLC1_OUT(1)		
01111-01100	**保留**		
01011	TMR5 溢出		**保留**
01010	TMR3 溢出	**保留**	TMR3 溢出
01001	**保留**	TMR1 溢出	
01000	TMR0 溢出		
00111	CLKREF		
00110	SOSC		
00101	MFINTOSC（500 kHz）		
00100	LFINTOSC		
00011	HFINTOSC		
00010	F_{OSC}		
00001	$F_{OSC}/4$		
00000	T1CKIPPS	T3CKIPPS	T5CKIPPS

注：
1. CN2510 器件上未提供。

复位状态： POR/BOR = 00000
所有其他复位 = uuuuu

22.13.4. TxGATE

名称： TxGATE
偏移量： 0xFD0,0xFCA,0xFC4

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 22-5. 定时器门控源

GSS	门控源		
	Timer1	Timer3	Timer5
11000-11111	**保留**		
10111	CLC8_OUT(1)		
10110	CLC7_OUT(1)		
10101	CLC6_OUT(1)		
10100	CLC5_OUT(1)		
10011	CLC4_OUT(1)		
10010	CLC3_OUT(1)		
10001	CLC2_OUT(1)		
10000	CLC1_OUT(1)		
01111	**保留**		
01110	ZCDOUT		
01101	C2_OUT		
01100	C1_OUT		
01011	PWM4_OUT		
01010	PWM3_OUT		
01001	CCP2_OUT		
01000	CCP1_OUT		
00111	TMR6_OUT（后分频）		
00110	TMR5 溢出		**保留**
00101	TMR4_OUT（后分频）		
00100	TMR3 溢出	**保留**	TMR3 溢出
00011	TMR2_OUT（后分频）		
00010	**保留**	TMR1 溢出	
00001	TMR0 溢出		
00000	通过 T1GPPS 选择的引脚	通过 T3GPPS 选择的引脚	通过 T5GPPS 选择的引脚

注：
1. CN2510 器件上未提供。

22.13.5. TMRx

名称： TMRx
偏移量： 0xFCC,0xFC6,0xFC0

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]

22.14. 寄存器汇总——Timer1

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0FBF	保留									
0x0FC0	TMR5	7:0	TMR5[7:0]							
		15:8	TMR5[15:8]							
0x0FC2	T5CON	7:0			CKPS[1:0]			SYNC	RD16	ON
0x0FC3	T5GCON	7:0	GE	GPOL	GTM	GSPM	GGO/DONE	GVAL		
0x0FC4	T5GATE	7:0				GSS[4:0]				
0x0FC5	T5CLK	7:0				CS[4:0]				
0x0FC6	TMR3	7:0	TMR3[7:0]							
		15:8	TMR3[15:8]							
0x0FC8	T3CON	7:0			CKPS[1:0]			SYNC	RD16	ON
0x0FC9	T3GCON	7:0	GE	GPOL	GTM	GSPM	GGO/DONE	GVAL		
0x0FCA	T3GATE	7:0				GSS[4:0]				
0x0FCB	T3CLK	7:0				CS[4:0]				
0x0FCC	TMR1	7:0	TMR1[7:0]							
		15:8	TMR1[15:8]							
0x0FCE	T1CON	7:0			CKPS[1:0]			SYNC	RD16	ON
0x0FCF	T1GCON	7:0	GE	GPOL	GTM	GSPM	GGO/DONE	GVAL		
0x0FD0	T1GATE	7:0				GSS[4:0]				
0x0FD1	T1CLK	7:0				CS[4:0]				

23. TMR2——Timer2 模块

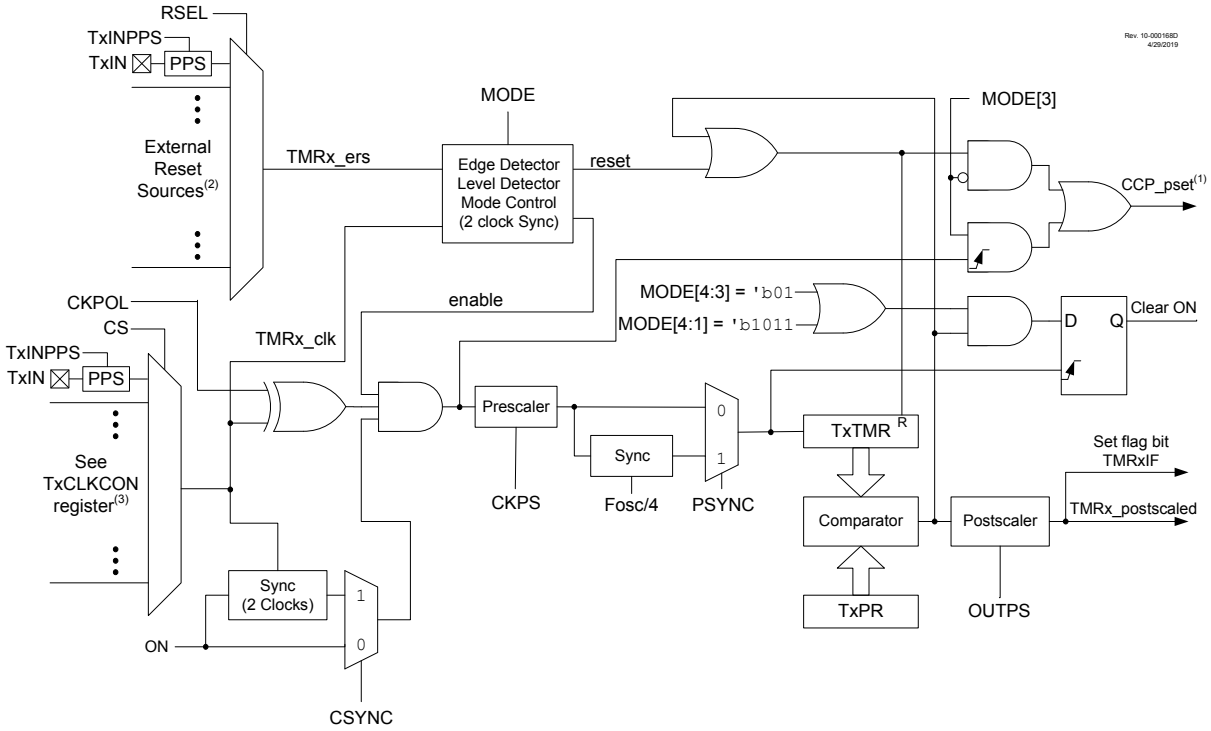
Timer2 模块是 8 位定时器，具有以下特性：

- 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 23-1. 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。

23.1. 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 23-1 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。

23.1.1. 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。

23.1.2. 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。

23.1.3. 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。

23.2. 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。

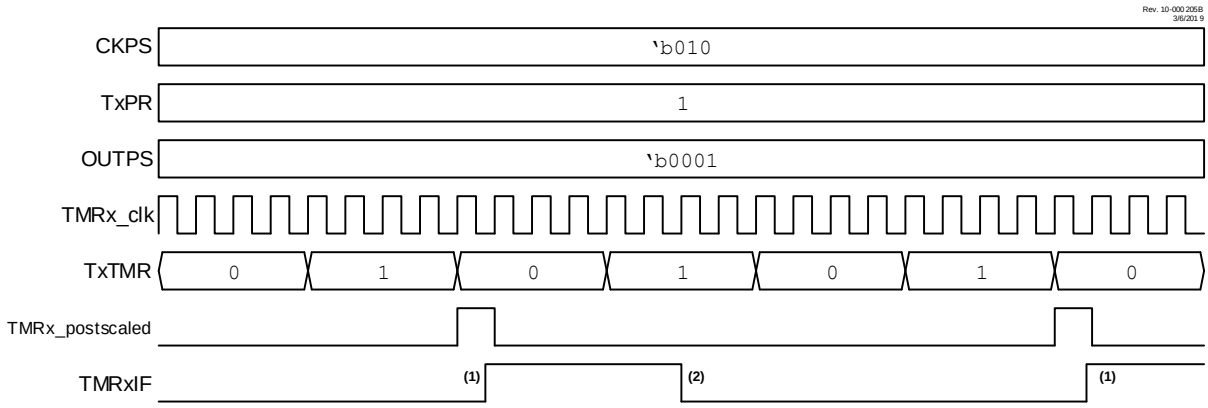
23.3. 外部复位源

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。

23.4. 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 23-2. 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.

23.5. 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。

23.6. 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。

23.7. 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 23-1. 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 23-3）	ON = 1	—	ON = 0
		001		硬件门控，高电平有效（图 23-4）	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 23-5）		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 23-6）		TMRx_ers = 1	ON = 0 或 TMRx_ers = 1
单触发	01	000	单触发	软件启动（图 23-7）	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 23-8）	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 23-9）	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 23-10）	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 23-11）	ON = 1 且 TMRx_ers ↑	—	ON = 0 或 TxTMR = TxPR 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 23-12）	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。

23.8. 操作示例

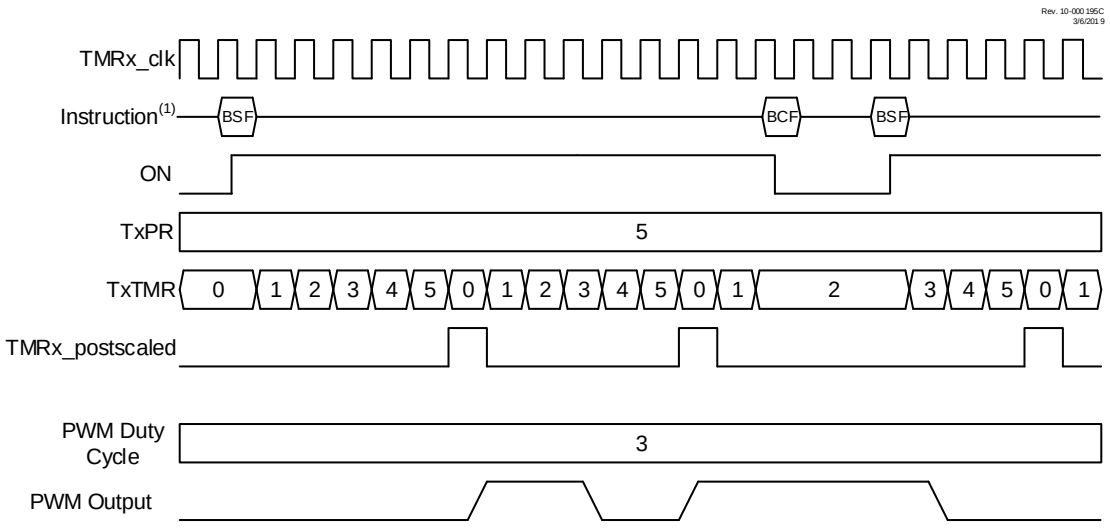
除非另外说明，否则下述内容适用于下列时序图：

- 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。

23.8.1. 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 23-3 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 23-3. 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.

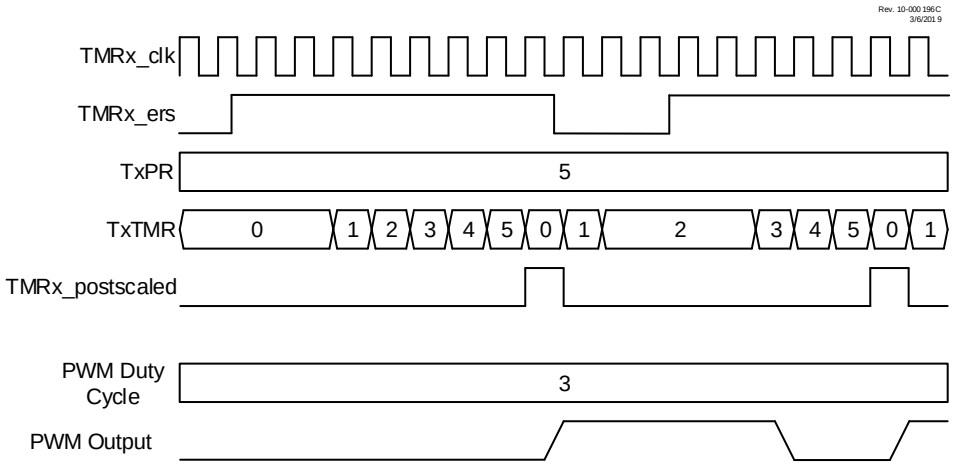
23.8.2. 硬件门控模式

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 23-4 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 23-4. 硬件门控模式时序图（MODE = 'b00001）



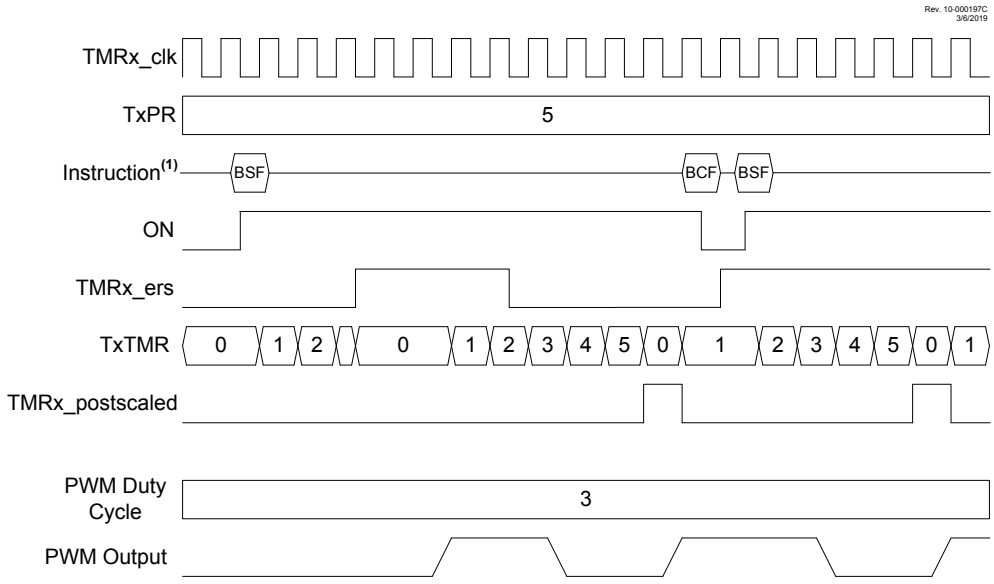
23.8.3. 边沿触发硬件限制模式

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 23-5。

图 23-5. 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.

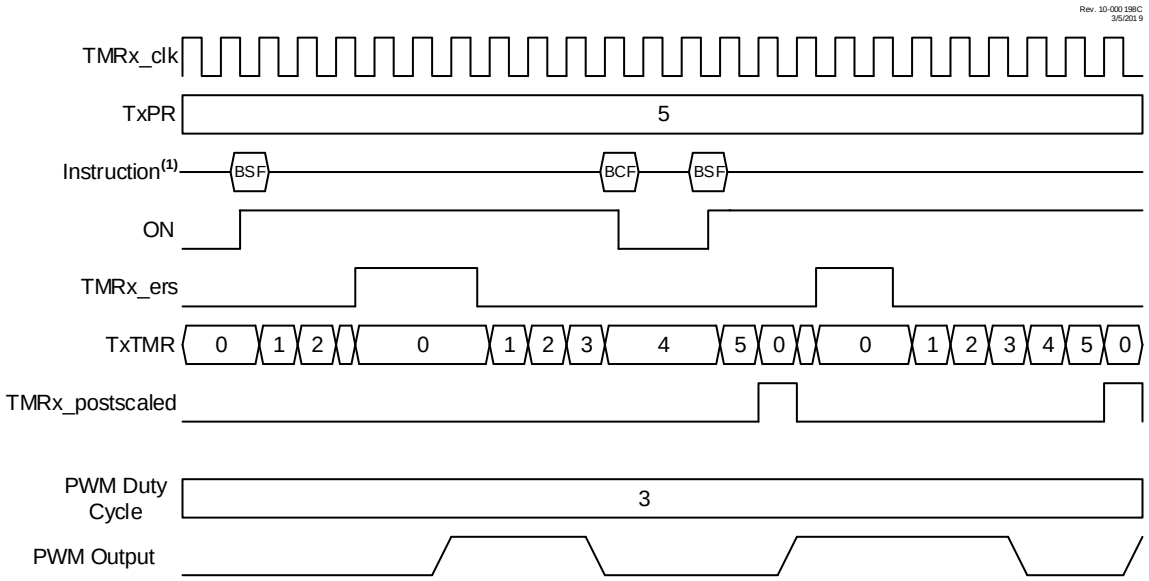
23.8.4. 电平触发硬件限制模式

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 23-6 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 23-6. 电平触发硬件限制模式时序图（MODE = 'b00111）



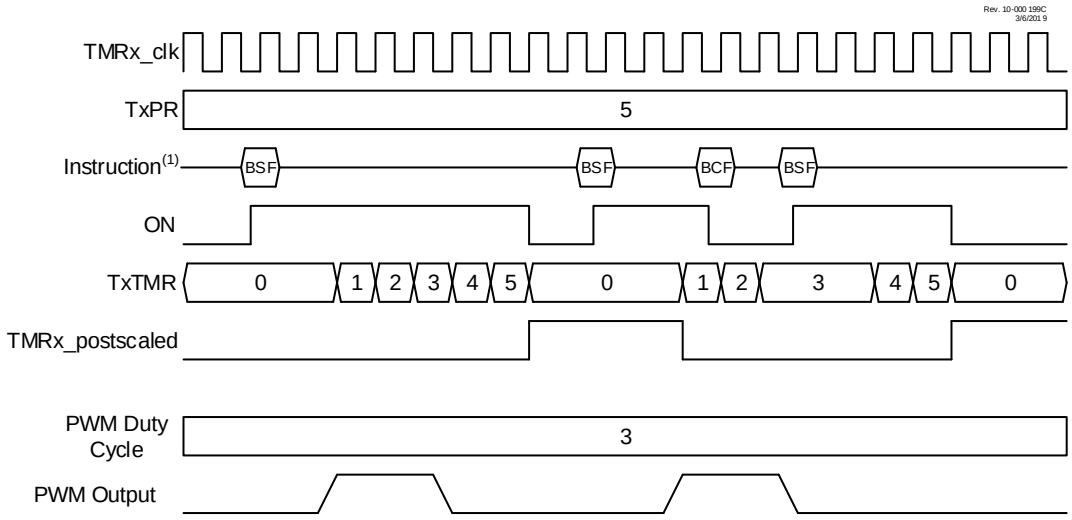
Note: 1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

23.8.5. 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 23-7 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 23-7. 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.

23.8.6. 边沿触发单触发模式

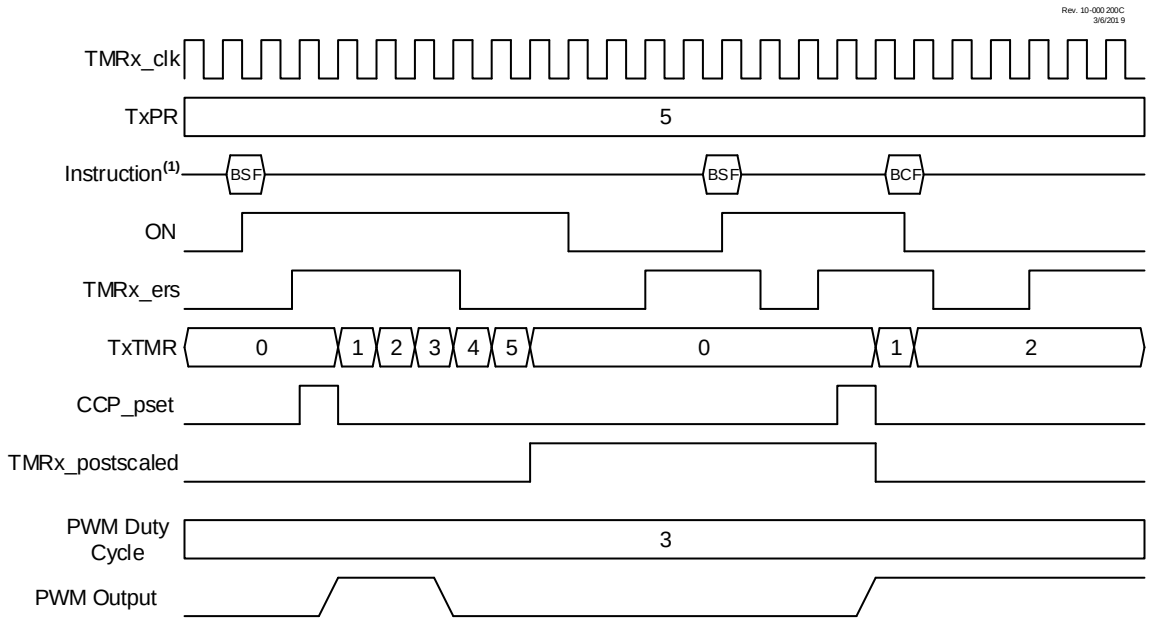
边沿触发单触发模式在 ON 位置 1 后通过外部信号输入的边沿启动定时器，并在定时器与 TxPR 周期值匹配时清零 ON 位。以下边沿将启动定时器：

- 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 23-8 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 周期计数匹配而暂停时，PWM 驱动保持禁止状态。

图 23-8. 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.

23.8.7. 边沿触发硬件限制单触发模式

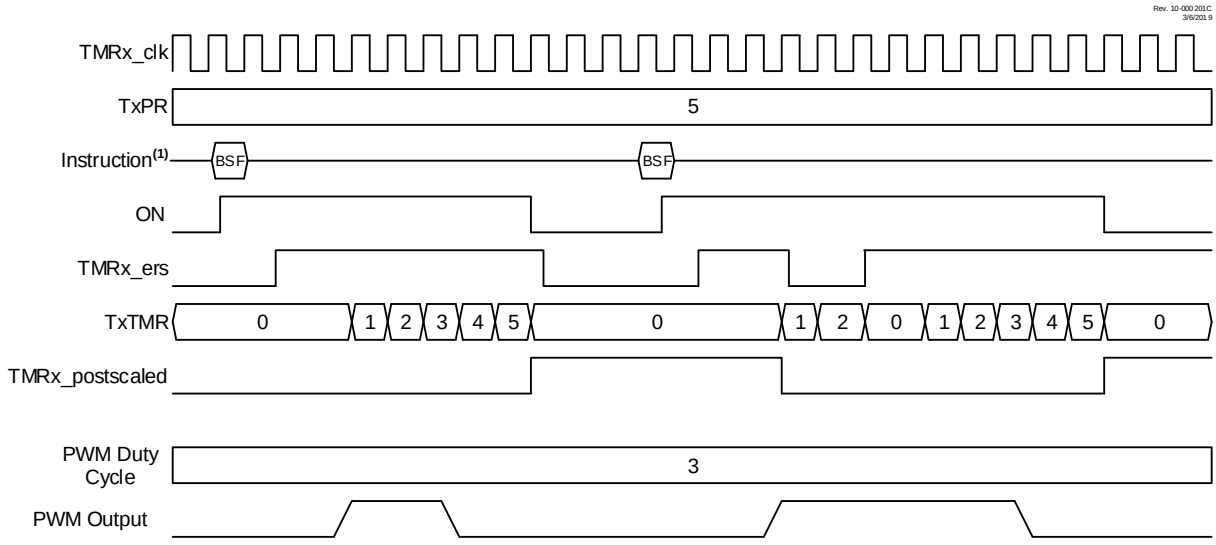
在边沿触发硬件限制单触发模式下，定时器在 ON 位置 1 后的第一个外部信号边沿启动，并在随后的所有边沿复位。只需要通过 ON 位置 1 后的第一个边沿来启动定时器即可。在随后的所有外部复位边沿的两个时钟周期后，计数器将自动恢复计数。边沿触发信号如下所示：

- 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 23-9 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 23-9. 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.

23.8.8. 电平复位、边沿触发硬件限制单触发模式

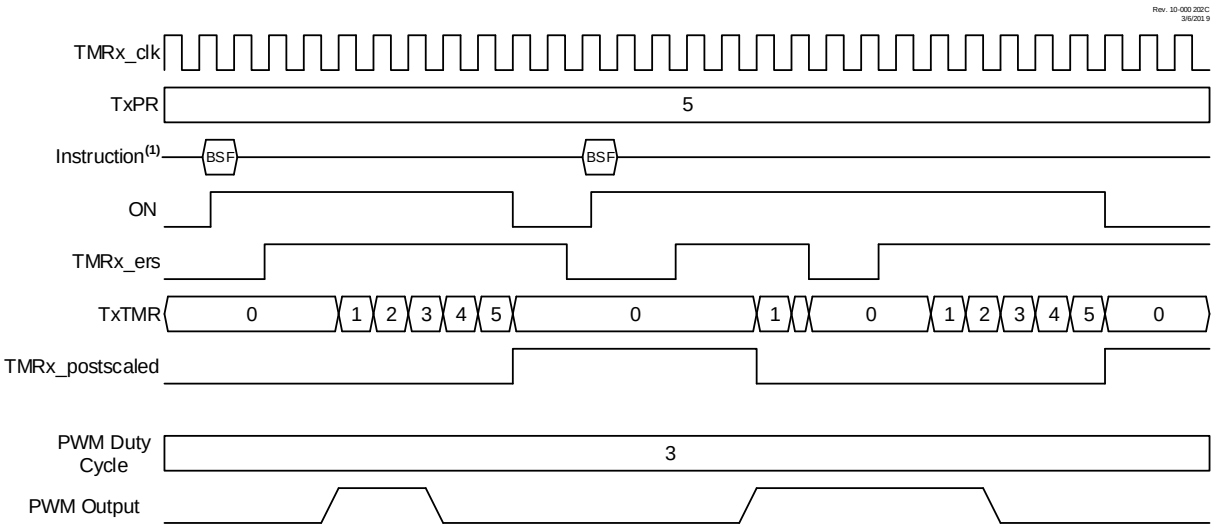
在电平触发单触发模式下，当 ON 位置 1 时，定时器计数通过外部信号电平复位，并在复位电平转变为有效电平的上升沿/下降沿开始计数。复位电平选择如下：

- 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 23-10. 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.

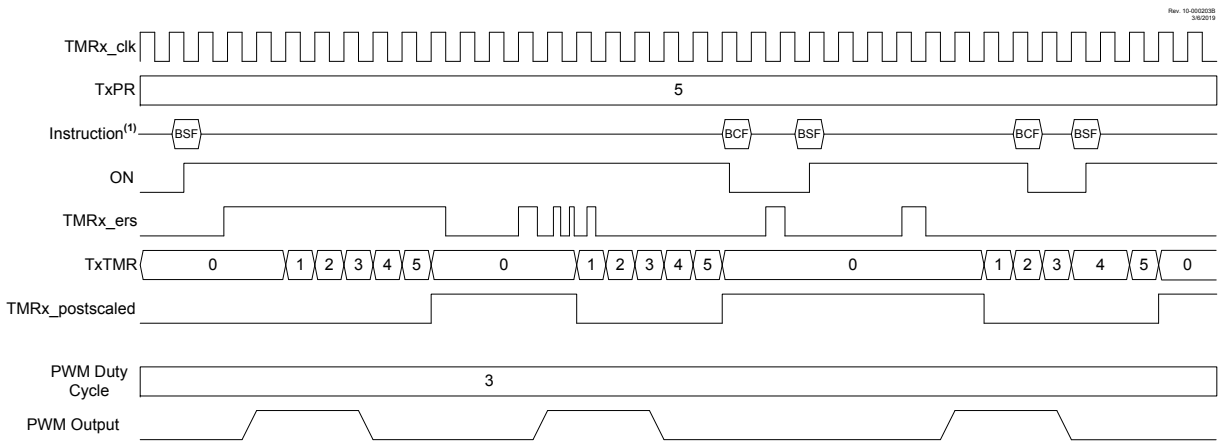
23.8.9. 边沿触发单稳态模式

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 23-11. 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.

23.8.10. 电平触发硬件限制单触发模式

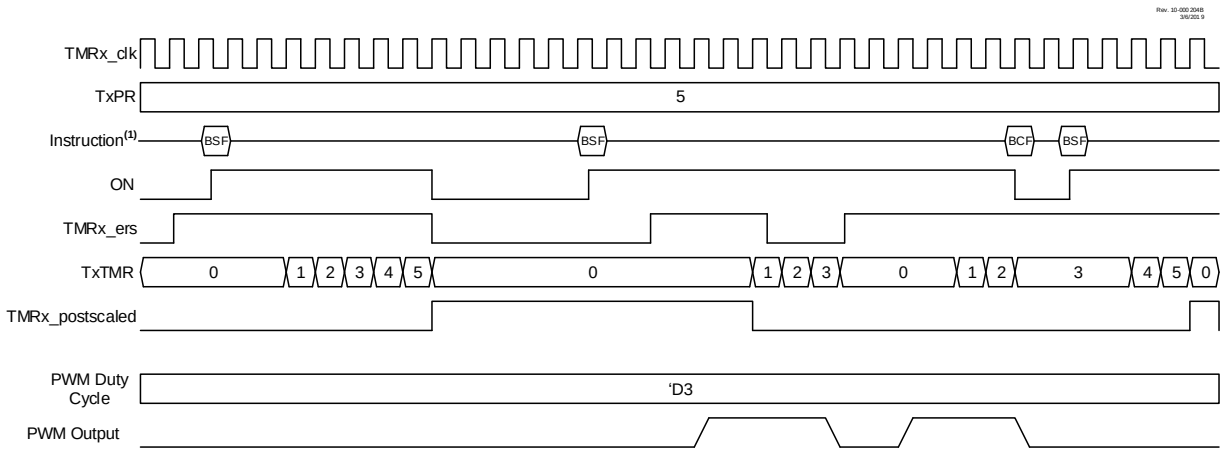
在电平触发硬件限制单触发模式下，定时器在外部复位电平出现时保持复位状态，并在 ON 位置 1 且外部信号不在复位电平时开始计数。如果其中一个外部信号不在复位电平或 ON 位置 1，则其他置为有效的信号将启动定时器。复位电平选择如下：

- 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 23-12. 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.

23.9. 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。

23.10. 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 23-2. Timer2 长位名称前缀

外设	位名称前缀
Timer2	T2
Timer4	T4
Timer6	T6

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）。

23.10.1. TxTMR

名称： TxTMR
偏移量： 0xFBA,0xFB4,0xFAE

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

23.10.2. TxPR

名称： TxPR
偏移量： 0xFBB,0xFB5,0xFAF

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

23.10.3. TxCON

名称： TxCON
偏移量： 0xFBC,0xFB6,0xFB0

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 23-1。

23.10.4. TxHLT

名称： TxHLT
偏移量： 0xFBD,0xFB7,0xFB1

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 23-1

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。

23.10.5. TxCLKCON

名称： TxCLKCON
偏移量： 0xFBE,0xFB8,0xFB2

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 23-3. 时钟源选择

CS[4:0]	时钟源		
	Timer2	Timer4	Timer6
11111-11000	**保留**		
10111	CLC8_OUT(1)		
10110	CLC7_OUT(1)		
10101	CLC6_OUT(1)		
10100	CLC5_OUT(1)		
10011	CLC4_OUT(1)		
10010	CLC3_OUT(1)		
10001	CLC2_OUT(1)		
10000	CLC1_OUT(1)		
01111-01001	**保留**		
01000	ZCD_OUT		
00111	CLKREF_OUT		
00110	SOSC		
00101	MFINTOSC（31 kHz）		
00100	LFINTOSC		
00011	HFINTOSC		
00010	F_{OSC}		
00001	$F_{OSC}/4$		
00000	通过 T2INPPS 选择的引脚	通过 T4INPPS 选择的引脚	通过 T6INPPS 选择的引脚

注：
1. CN2510 器件上未提供。

23.10.6. TxRST

名称： TxRST
偏移量： 0xFBF,0xFB9,0xFB3

定时器外部复位信号选择寄存器

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

Bit 4:0 – RSEL[4:0] 外部复位源选择

表 23-4. 外部复位源

RSEL[4:0]	复位源		
	TMR2	TMR4	TMR6
11111-11000	**保留**		
10111	CLC8_OUT(1)		
10110	CLC7_OUT(1)		
10101	CLC6_OUT(1)		
10100	CLC5_OUT(1)		
10011	CLC4_OUT(1)		
10010	CLC3_OUT(1)		
10001	CLC2_OUT(1)		
10000	CLC1_OUT(1)		
01111	EUSART2 TX/CK(1)		
01110	EUSART2 DT(1)		
01101	EUSART1 TX/CK		
01100	EUSART1 DT		
01011	**保留**		
01010	ZCD_OUT		
01001	C2_OUT		
01000	C1_OUT		
00111	PWM4_OUT		
00110	PWM3_OUT		
00101	CCP2_OUT		
00100	CCP1_OUT		
00011	TMR6 后分频		**保留**
00010	TMR4 后分频	**保留**	TMR4 后分频
00001	**保留**	TMR2 后分频	
00000	通过 T2INPPS 选择的引脚	通过 T4INPPS 选择的引脚	通过 T6INPPS 选择的引脚

注：
1. CN2510 器件上未提供。

23.11. 寄存器汇总——Timer2

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0FAD	保留									
0x0FAE	T6TMR	7:0	T6TMR[7:0]							
0x0FAF	T6PR	7:0	T6PR[7:0]							
0x0FB0	T6CON	7:0	ON		CKPS[2:0]			OUTPS[3:0]		
0x0FB1	T6HLT	7:0	PSYNC	CPOL	CSYNC	MODE[4:0]				
0x0FB2	T6CLKCON	7:0				CS[4:0]				
0x0FB3	T6RST	7:0				RSEL[4:0]				
0x0FB4	T4TMR	7:0	T4TMR[7:0]							
0x0FB5	T4PR	7:0	T4PR[7:0]							
0x0FB6	T4CON	7:0	ON		CKPS[2:0]			OUTPS[3:0]		
0x0FB7	T4HLT	7:0	PSYNC	CPOL	CSYNC	MODE[4:0]				
0x0FB8	T4CLKCON	7:0				CS[4:0]				
0x0FB9	T4RST	7:0				RSEL[4:0]				
0x0FBA	T2TMR	7:0	T2TMR[7:0]							
0x0FBB	T2PR	7:0	T2PR[7:0]							
0x0FBC	T2CON	7:0	ON		CKPS[2:0]			OUTPS[3:0]		
0x0FBD	T2HLT	7:0	PSYNC	CPOL	CSYNC	MODE[4:0]				
0x0FBE	T2CLKCON	7:0				CS[4:0]				
0x0FBF	T2RST	7:0				RSEL[4:0]				

24. 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。

每个 CCP 模块均可选择控制模块的定时器源。使用捕捉/比较模式时，默认选择定时器 Timer1，而对 CCPx 模块使用 PWM 模式时，默认选择定时器 Timer2。

请注意，以下部分将针对 Timer1 介绍捕捉/比较模式操作，并针对 Timer2 介绍 PWM 模式操作。

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。

24.1. 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。

24.1.1. CCP 模块和定时器资源

CCP 模块使用定时器 1 至 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 24-1. CCP 模式——定时器资源

CCP 模式	定时器资源
捕捉	Timer1、Timer3 或 Timer5
比较	
PWM	Timer2、Timer4 或 Timer6

有关将特定定时器分配到模块的选择，请参见**“捕捉、比较和 PWM 定时器选择”**一章。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。

24.1.2. 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。

24.2. 捕捉模式

捕捉模式使用 16 位奇数编号的定时器资源（Timer1 和 Timer3 等）。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。事件定义为以下事件之一，并由 MODE 位配置：

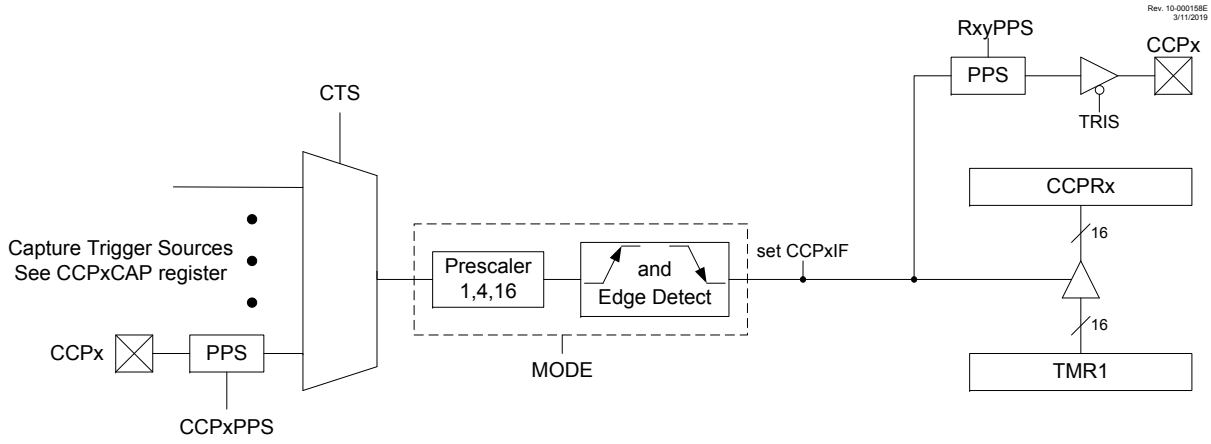
- CCPx 输入的每个下降沿
- CCPx 输入的每个上升沿
- CCPx 输入的每 4 个上升沿
- CCPx 输入的每 16 个上升沿

- CCPx 输入的每个边沿（上升沿或下降沿）

当进行捕捉时，PIRx 寄存器的中断请求标志位 CCPxIF 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，则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事件。建议在读取 CCPRx 寄存器对时禁止模块或将寄存器对读取两次以确保数据完整性。

图 24-1. 捕捉模式工作原理框图



24.2.1. 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。

24.2.2. 使用捕捉功能时的 Timer1 模式

要使 CCP 模块使用捕捉功能，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”**一章。

24.2.3. 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。

24.2.4. 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 24-1. 切换捕捉预分频比

```
BANKSEL CCP1CON         ;only needed when CCP1CON is not in ACCESS space
CLRF    CCP1CON         ;Turn CCP module off
MOVLW   NEW_CAPT_PS     ;CCP ON and Prescaler select → W
MOVWF   CCP1CON         ;Load CCP1CON with this value
```

24.2.5. 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。该模块可以由指令时钟（$F_{OSC}/4$）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。

24.3. 比较模式

本节介绍的比较模式功能适用于所有 CCP 模块而且是相同的。

比较模式使用 16 位奇数编号的定时器资源（Timer1 和 Timer3 等）。CCPRx 寄存器的 16 位值不断与 TMRx 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：

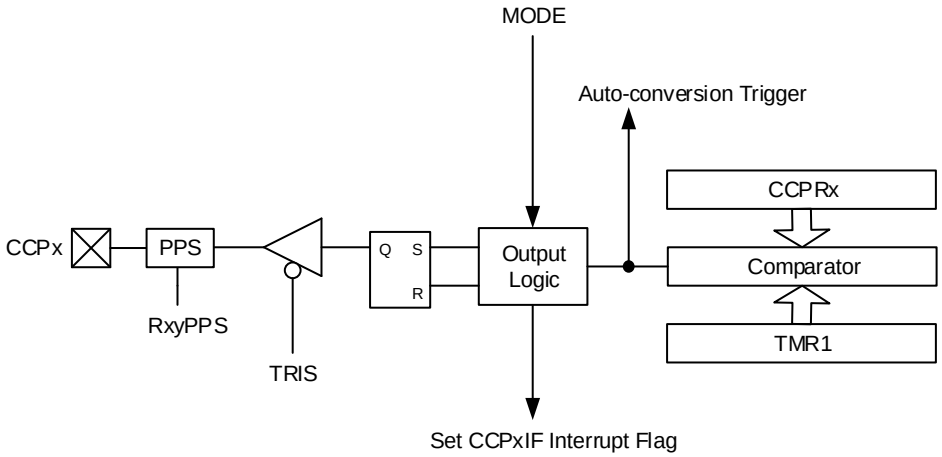
- 翻转 CCPx 输出并清零 TMRx
- 翻转 CCPx 输出但不清零 TMRx
- CCPx 输出高电平
- CCPx 输出低电平
- 产生脉冲输出
- 产生脉冲输出并清零 TMRx

引脚的动作由 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 会复位 TMRx 寄存器。

下图给出了比较操作的简化框图。

图 24-2. 比较模式工作原理框图



24.3.1. 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。

24.3.2. 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。

24.3.3. 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。

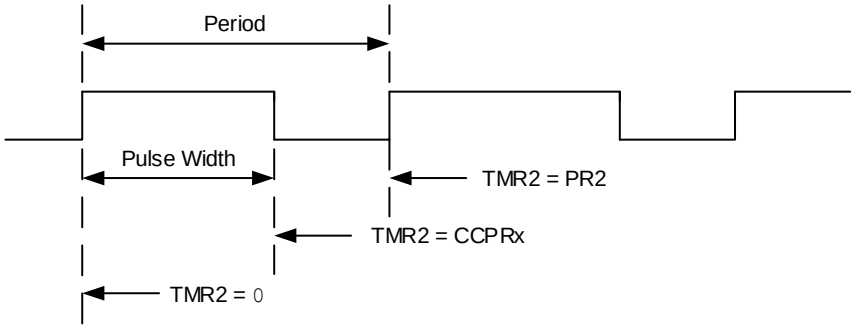
24.4. PWM 概述

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 24-3. CCP PWM 输出信号



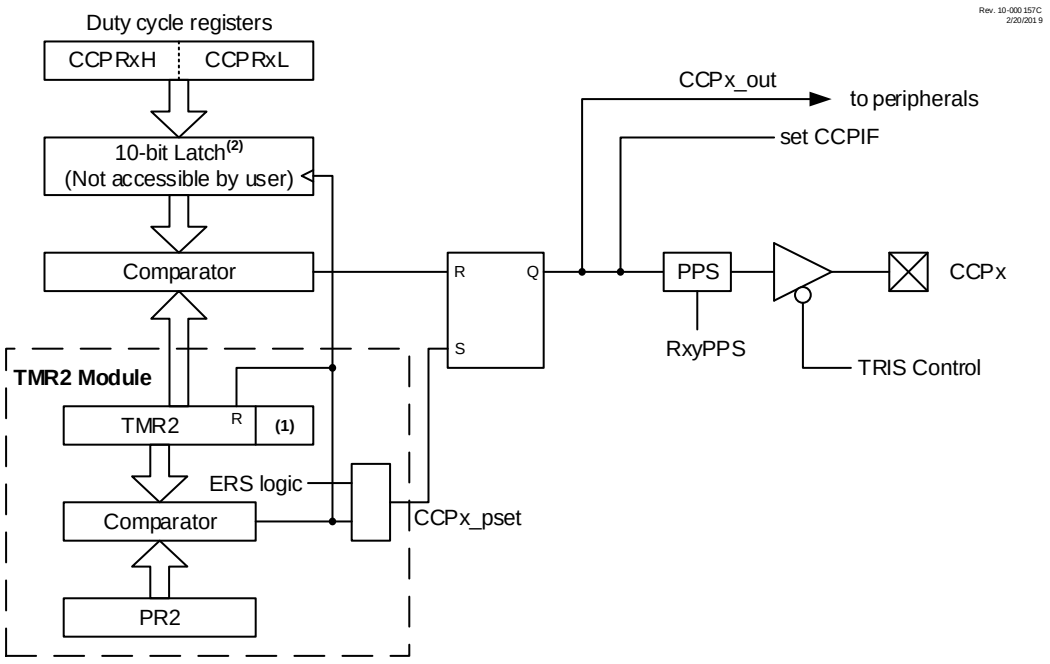
24.4.1. 标准 PWM 操作

本节介绍的标准 PWM 功能适用于所有 CCP 模块而且是相同的。它可以在 CCPx 引脚上产生最高 10 位分辨率的脉宽调制（PWM）信号。通过下列寄存器控制周期、占空比和分辨率：

- 偶数编号的 TxPR 寄存器（T2PR 和 T4PR 等）
- 偶数编号的 TxCON 寄存器（T2CON 和 T4CON 等）
- 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 TxTMR 的时钟输入。下图给出了 PWM 操作的简化框图。

图 24-4. 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。

24.4.2. 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 周期值装入所选的定时器 TxPR 周期寄存器。
3. 通过将合适的值装入 CCPxCON 寄存器来配置 CCP 模块，使其在 PWM 模式下工作。
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. 配置并启动所选的定时器：
   - 清零 PIRx 寄存器的 TMRxIF 中断标志位。请参见下面的重要注意事项。
   - 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。
   - 用所需的定时器预分频值配置 TxCON 寄存器的 TxCKPS 位。
   - 通过将 TxON 位置 1，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 TMRxIF 位置 1。请参见下面的重要注意事项。
   - 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，那么可以忽略步骤 6。

24.4.3. 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。

24.4.4. 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 24-1. PWM 周期

$$PWM\ 周期 = [(T2PR + 1)] \bullet 4 \bullet T_{OSC} \bullet (TMR2\ 预分频值)$$

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

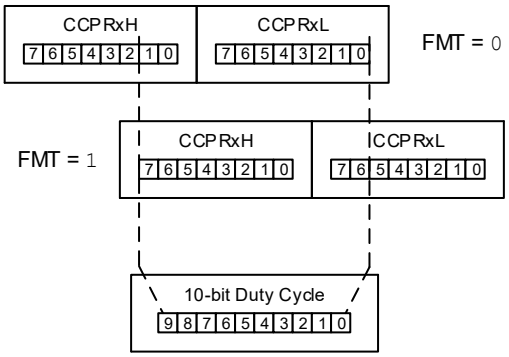
重要：不使用定时器后分频器（见**“TMR2——Timer2 模块”**一章中的**“Timer2 中断”**一节）来确定 PWM 频率。

24.4.5. 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 24-5）。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 24-5. PWM 10 位对齐方式



公式 24-2. 脉冲宽度

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

公式 24-3. 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 24-4）。

24.4.6. 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 24-4. 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 24-2. 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 24-3. 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

24.4.7. 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。

24.4.8. 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。更多详细信息，请参见**“OSC——振荡器模块（带故障保护时钟监视器）”**一章。

24.4.9. 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。

24.5. 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 24-4. CCP 长位名称前缀

外设	位名称前缀
CCP1	CCP1
CCP2	CCP2

24.5.1. CCPxCON

名称： CCPxCON
偏移量： 0xFAB,0xFA7

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 24-5. 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，置 1 输出(1)	是
0111	捕捉模式，CCPx 输入的每 16 个上升沿	是
0110	捕捉模式，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。

24.5.2. CCPxCAP

名称： CCPxCAP
偏移量： 0xFAC,0xFA8

捕捉触发输入选择寄存器

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

Bit 3:0 – CTS[3:0] 捕捉触发输入选择

表 24-6. 捕捉触发源

CTS	源
1111	CLC8_out(1)
1110	CLC7_out(1)
1101	CLC6_out(1)
1100	CLC5_out(1)
1011	CLC4_out(1)
1010	CLC3_out(1)
1001	CLC2_out(1)
1000	CLC1_out(1)
0111–0100	**保留**
0011	IOC 中断
0010	C2_out
0001	C1_out
0000	通过 CCPxPPS 选择的引脚
注：CN2510 器件上未提供 CLC。	

24.5.3. CCPRx

名称： CCPRx
偏移量： 0xFA9,0xFA5

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

24.6. 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 ... 0x0FA4	保留									
0x0FA5	CCPR2	7:0	CCPR[7:0]							
		15:8	CCPR[15:8]							
0x0FA7	CCP2CON	7:0	EN		OUT	FMT	MODE[3:0]			
0x0FA8	CCP2CAP	7:0					CTS[3:0]			
0x0FA9	CCPR1	7:0	CCPR[7:0]							
		15:8	CCPR[15:8]							
0x0FAB	CCP1CON	7:0	EN		OUT	FMT	MODE[3:0]			
0x0FAC	CCP1CAP	7:0					CTS[3:0]			

25. 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：

- TxPR
- TxCON
- PWMxDC
- PWMxCON

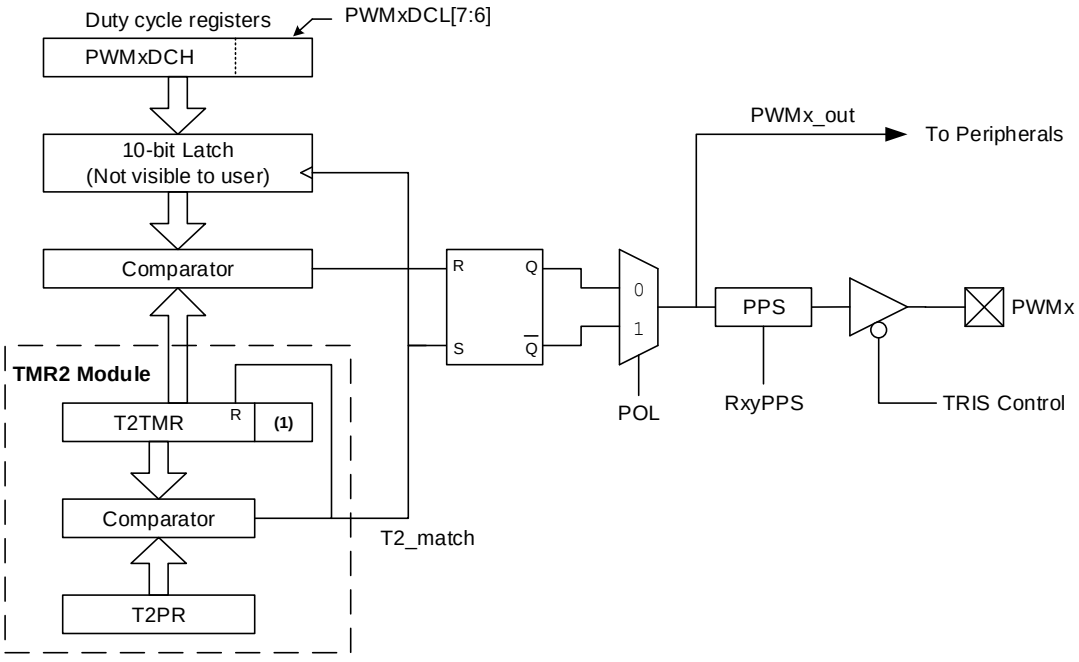
重要：必须将 PWMx 引脚对应的 TRIS 位清零，才能使能该引脚上的 PWM 输出。

每个 PWM 模块使用相同的定时器源 Timer2 来控制各个模块。

图 25-1 给出了 PWM 操作的简化框图。

图 25-2 给出了 PWM 信号的典型波形。

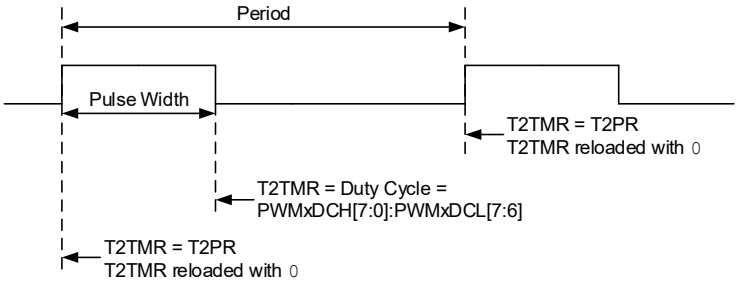
图 25-1. PWM 简化框图



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 25-2. 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。

25.1. 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 TMRx。TxTMR 和 TxPR 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 Timerx 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 TxTMR 清零时，与 Timerx 相关的所有 PWM 输出都会置 1。当 TxTMR 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 TxPR 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 TxTMR 与 TxPR 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。

25.2. PWM 输出极性

输出极性通过将 POL 位置 1 进行翻转。

25.3. PWM 周期

PWM 周期由 TxPR 寄存器指定。PWM 周期可由公式 25-1 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 25-1. PWM 周期

$$PWM周期 = [(T2PR) + 1] \bullet 4 \bullet Tosc \bullet (TMR2\ 预分频值)$$

注：$T_{OSC} = 1/F_{OSC}$

当 TxTMR 中的值与 TxPR 中的值相等时，在下一个递增周期将发生以下 3 个事件：

- TxTMR 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。

25.4. 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 25-2. 脉冲宽度

$$脉冲宽度 = (PWMxDCH:PWMxDCL[7:6]) \bullet Tosc \bullet (TMR2\ 预分频值)$$

注：$T_{OSC} = 1/F_{OSC}$

公式 25-3. 占空比

$$占空比 = \frac{(PWMxDCH:PWMxDCL[7:6])}{4(T2PR+1)}$$

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。

25.5. 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 25-4. PWM 分辨率

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

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 25-1. 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 25-2. 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

25.6. 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。

25.7. 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。

25.8. 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。

25.9. 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 TxPR 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 Timerx：
   - 清零 PIRx 寄存器的 TMRxIF 中断标志位。(1)
   - 需使用 TxCLKCON 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 Timerx 预分频值配置 TxCON 寄存器的 CKPS 位。
   - 通过将 TxCON 寄存器的 ON 位置 1 来使能 Timerx。
6. 使能 PWM 输出引脚并等待至 Timerx 溢出，PIRx 寄存器的 TMRxIF 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。

25.9.1. 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。

25.10. 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 TxPR 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 Timerx：
   - 清零 PIRx 寄存器的 TMRxIF 中断标志位。(1)
   - 需使用 TxCLKCON 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 Timerx 预分频值配置 TxCON 寄存器的 CKPS 位。
   - 通过将 TxCON 寄存器的 ON 位置 1 来使能 Timerx。
6. 等待至 Timerx 溢出，将 PIRx 寄存器的 TMRxIF 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。

25.11. 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 25-3. PWM 位名称前缀

外设	位名称前缀
PWM3	PWM3
PWM4	PWM4

25.11.1. PWMxCON

名称： PWMxCON
偏移量： 0xFA4,0xFA1

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

25.11.2. PWMxDC

名称： PWMxDC
偏移量： 0xFA2,0xF9F

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

25.12. 寄存器汇总——PWM

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0F9E	保留									
0x0F9F	PWM4DC	7:0	DCL[1:0]							
		15:8	DCH[7:0]							
0x0FA1	PWM4CON	7:0	EN		OUT	POL				
0x0FA2	PWM3DC	7:0	DCL[1:0]							
		15:8	DCH[7:0]							
0x0FA4	PWM3CON	7:0	EN		OUT	POL				

26. CCP 和 PWM 定时器选择

这些模块都可以独立选择定时器，通过 CCP/PWM 定时器选择（CCPTMRS）寄存器选择定时器。对于捕捉或比较功能，默认选择定时器 Timer1；对于 PWM 功能，默认选择定时器 Timer2。

26.1. 寄存器定义：CCP 和 PWM 定时器选择

26.1.1. CCP 定时器控制寄存器

名称： CCPTMRS
偏移量： 0xFAD

位	7	6	5	4	3	2	1	0
	P4TSEL[1:0]		P3TSEL[1:0]		C2TSEL[1:0]		C1TSEL[1:0]	
访问	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	1	0	1	0	1	0	1

Bit 4:5, 6:7 – PnTSEL PWMn 定时器选择位

值	说明
11	PWMn 以 Timer6 作为定时器
10	PWMn 以 Timer4 作为定时器
01	PWMn 以 Timer2 作为定时器
00	保留

Bit 0:1, 2:3 – CnTSEL CCPn 定时器选择位

值	说明
11	在捕捉/比较模式下，CCPn 以 Timer5 作为定时器；在 PWM 模式下，CCPn 以 Timer6 作为定时器
10	在捕捉/比较模式下，CCPn 以 Timer3 作为定时器；在 PWM 模式下，CCPn 以 Timer4 作为定时器
01	在捕捉/比较模式下，CCPn 以 Timer1 作为定时器；在 PWM 模式下，CCPn 以 Timer2 作为定时器
00	保留

26.2. 寄存器汇总——CCP 和 PWM 定时器选择

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0FAC	保留									
0x0FAD	CCPTMRS	7:0	P4TSEL[1:0]		P3TSEL[1:0]		C2TSEL[1:0]		C1TSEL[1:0]	

27. CWG——互补波形发生器模块

互补波形发生器（CWG）可产生半桥、全桥和转向模式的 PWM 波形。它与之前的 CCP 功能向后兼容。

CWG 具有如下特性：

- 6 种工作模式：
  - 同步转向模式
  - 异步转向模式
  - 全桥模式，正向
  - 全桥模式，反向
  - 半桥模式
  - 推挽模式
- 输出极性控制
- 输出转向
- 独立的 6 位上升沿和下降沿事件死区定时器：
  - 时钟驱动死区
  - 独立的上升沿和下降沿死区使能
- 具有以下特性的自动关断控制：
  - 可选的关断源
  - 自动重启选项
  - 自动关断引脚改写控制

27.1. 基本操作

CWG 通过所选输入源生成两个输出波形。

每路输出由关到开的转变可能会因其他输出由开到关的转变而延迟，因而在未驱动任何输出前会立即产生延时。该时间被称为死区，死区控制一节将对此进行介绍。

有必要防止可能的电路故障或者反馈事件到达过晚或完全不到达。在这种情况下，必须在故障状况造成损坏之前终止有效驱动。这称为自动关断，自动关断一节将对此进行介绍。

27.2. 工作模式

CWG 模块可以在 6 种不同模式下工作，这些模式由 MODE 位指定：

- 半桥模式
- 推挽模式
- 异步转向模式
- 同步转向模式
- 全桥模式，正向
- 全桥模式，反向

所有模式均接受单脉冲输入，并且提供最多 4 个输出，如以下小节所述。

所有模式均包括自动关断控制，如自动关断一节所述。

重要：除全桥模式（如全桥模式中所述）外，必须仅在 EN = 0 时进行模式更改。

27.2.1. 半桥模式

在半桥模式下，将以输入的真值和取反形式生成两个输出信号，如图 27-1 所示。在两个输出之间插入不重叠（死区）时间，以防止各种电源应用中产生直通电流。死区控制一节对死区控制进行了介绍。该模式下无法使用输出转向功能。该模式的基本框图如图 27-2 所示。

未用输出 CWGxC 和 CWGxD 驱动信号与 CWGxA 和 CWGxB 类似，它们的极性分别由 POLC 和 POLD 位独立控制。

图 27-1. CWG 半桥模式工作原理

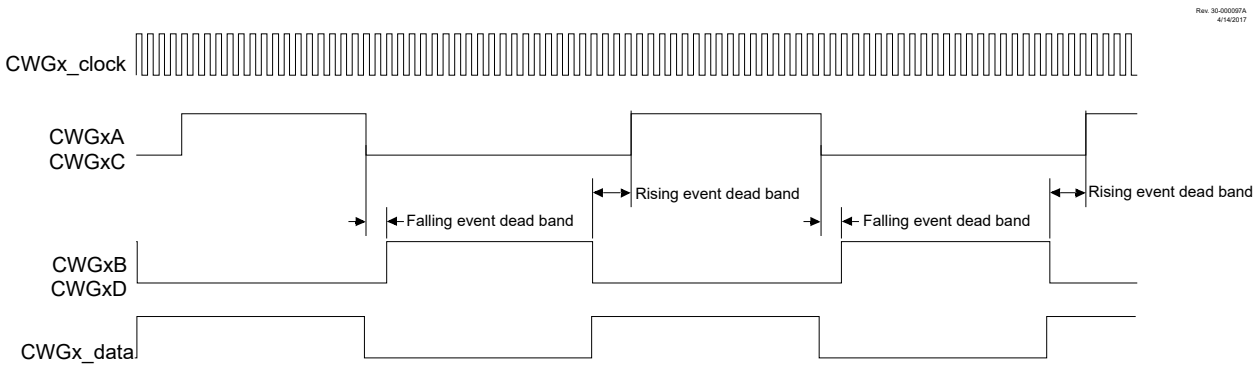
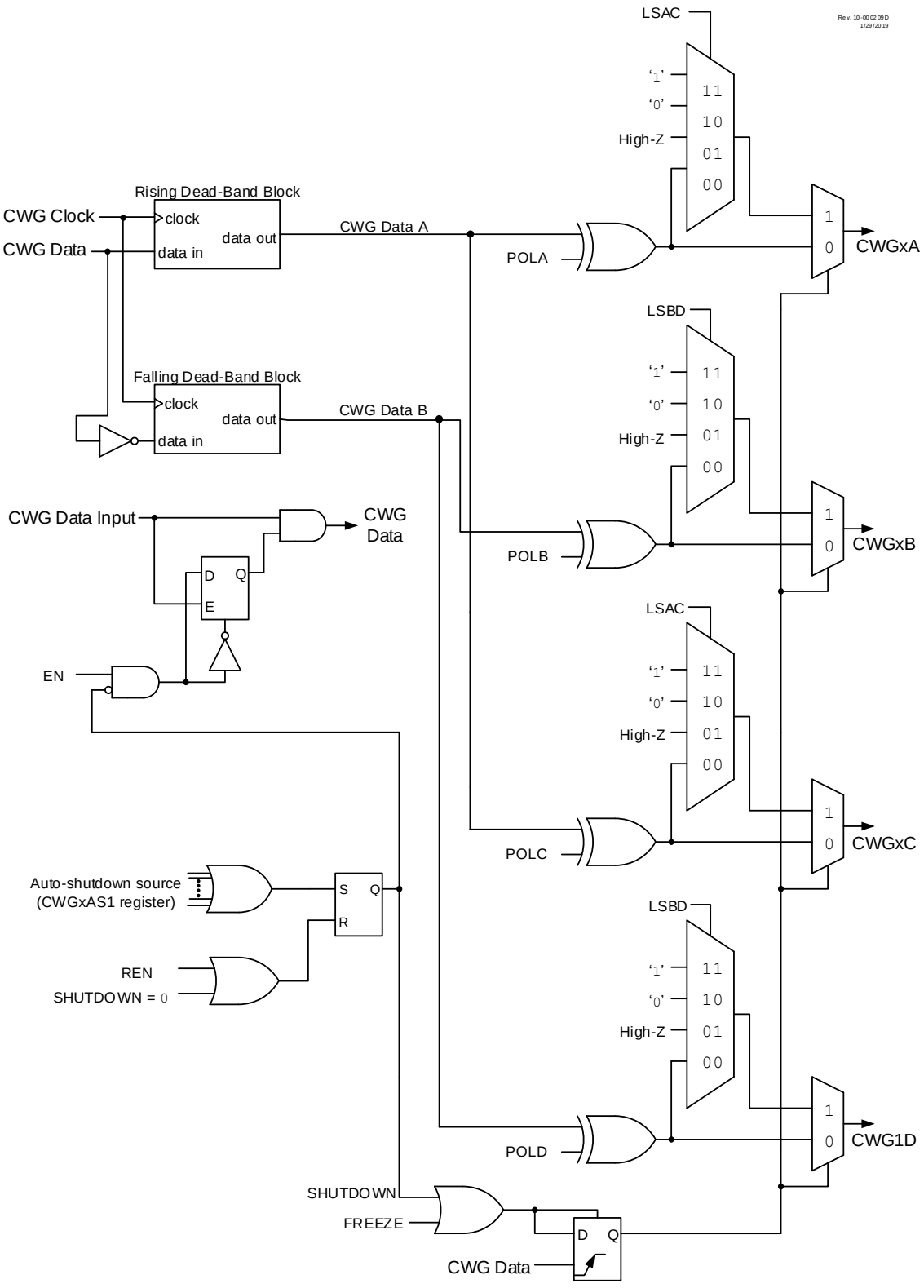


图 27-2. 简化的 CWG 框图（半桥模式，MODE = 'b100）



27.2.2. 推挽模式

在推挽模式下，将生成两个输出信号，它们为输入的交替副本，如图 27-3 所示。这种交替可以产生驱动一些基于变压器的电源设计所需的推挽效应。转向模式不用于半桥模式。图 27-4 给出了推挽模式的基本框图。

推挽定序器在每次 EN = 0 或发生自动关断事件时复位。该定序器由第一个输入脉冲提供时钟，第一个输出出现在 CWGxA 上。

未用输出 CWGxC 和 CWGxD 分别驱动 CWGxA 和 CWGxB 的副本，但它们的极性分别由 POLC 和 POLD 位控制。

图 27-3. CWG 推挽模式工作原理

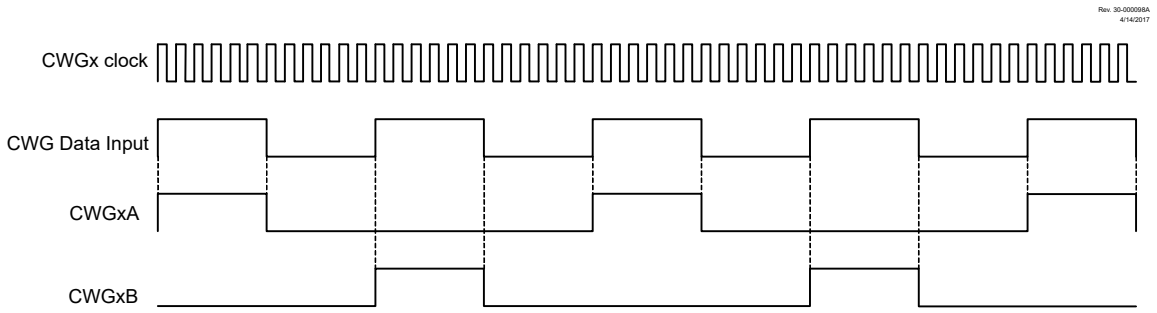
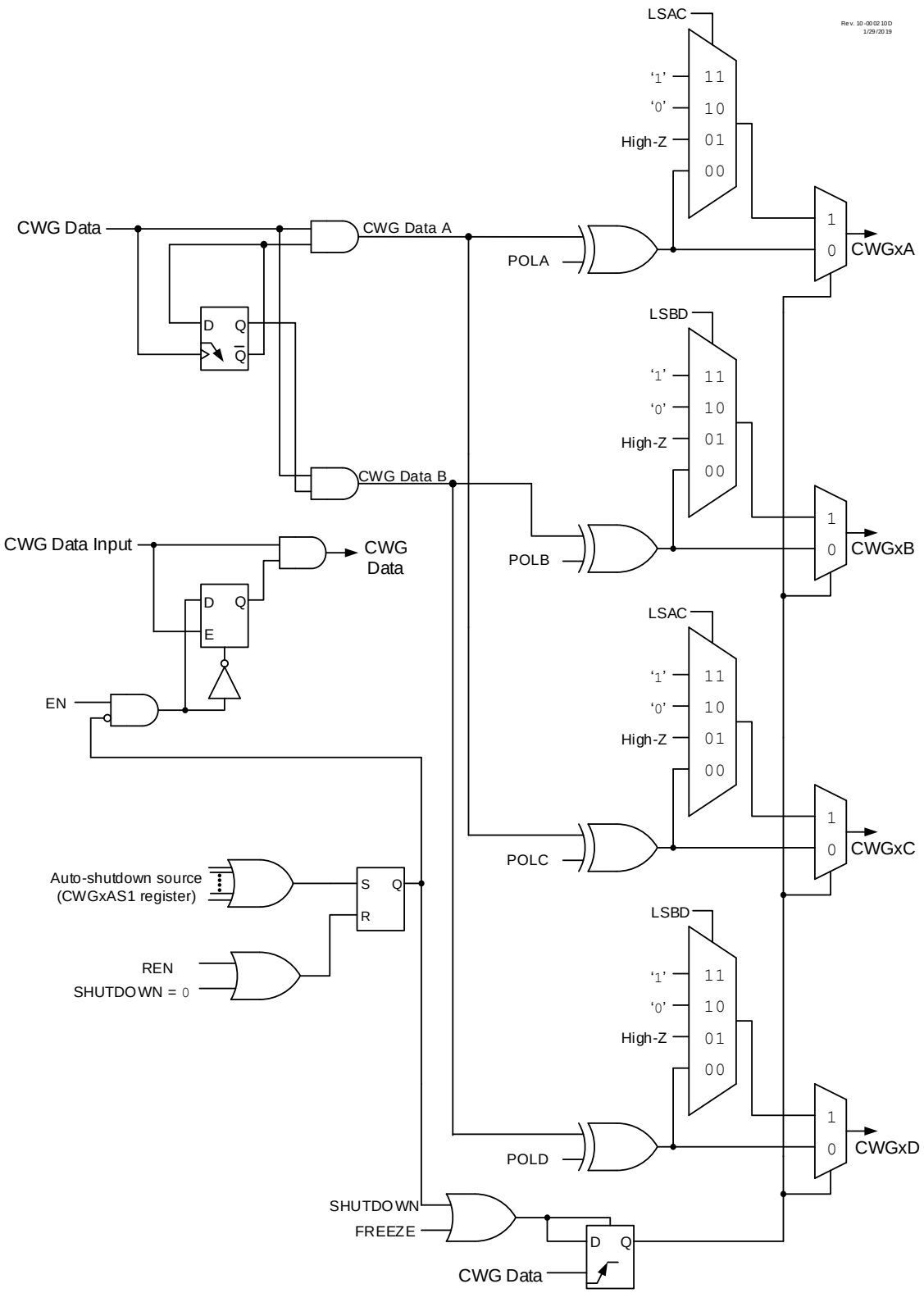


图 27-4. CWG 简化框图（推挽模式，MODE = 'b101）



27.2.3. 全桥模式

在正向和反向全桥模式下，3 个输出驱动静态值，第 4 个输出则通过输入数据信号进行调制。要在正向全桥模式与反向全桥模式之间切换，使 MODE[2:1]位保持静态，同时翻转 CWGxCON0 寄存器的 MODE[0]位，无需禁止 CWG 模块。连接后，如图 27-5 所示，输出适合全桥电机驱动器。每个 CWG 输出信号具有独立极性控制，因此该电路适合高电平有效和低电平有效驱动器。全桥模式的简化框图如图 27-6 所示。

图 27-5. 全桥应用示例

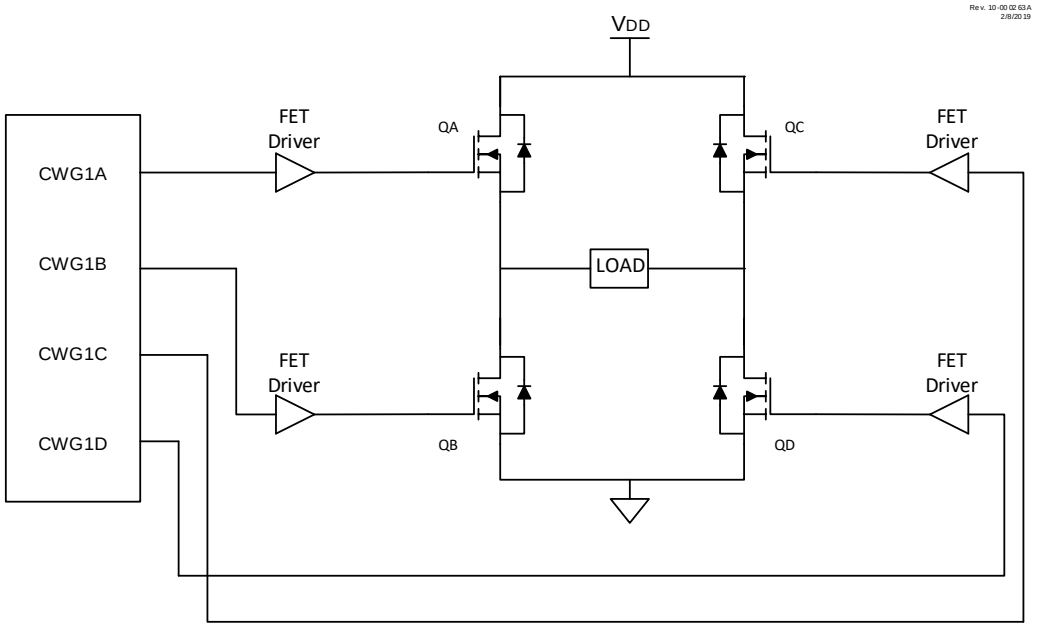
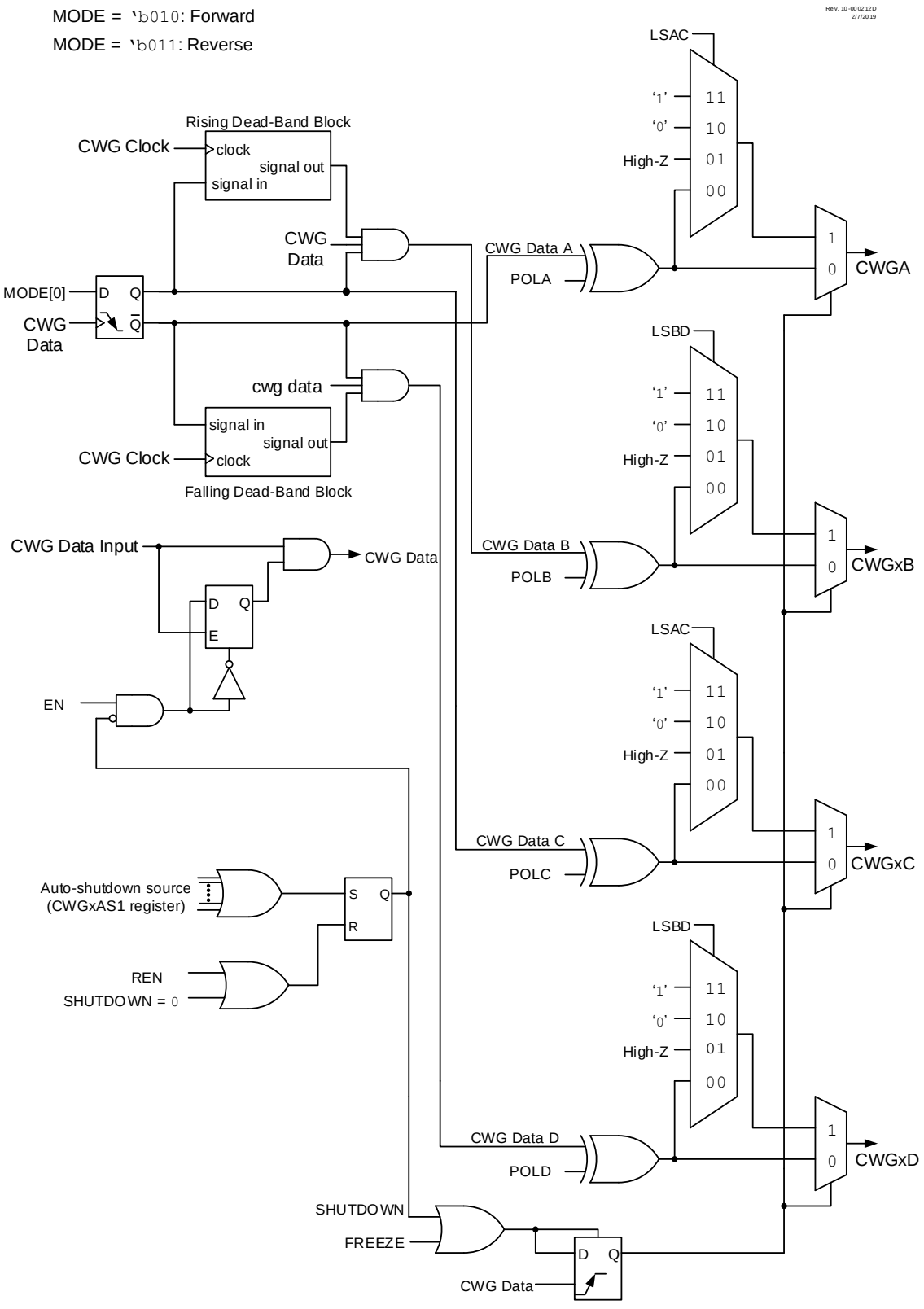


图 27-6. CWG 简化框图（正向和反向全桥模式）

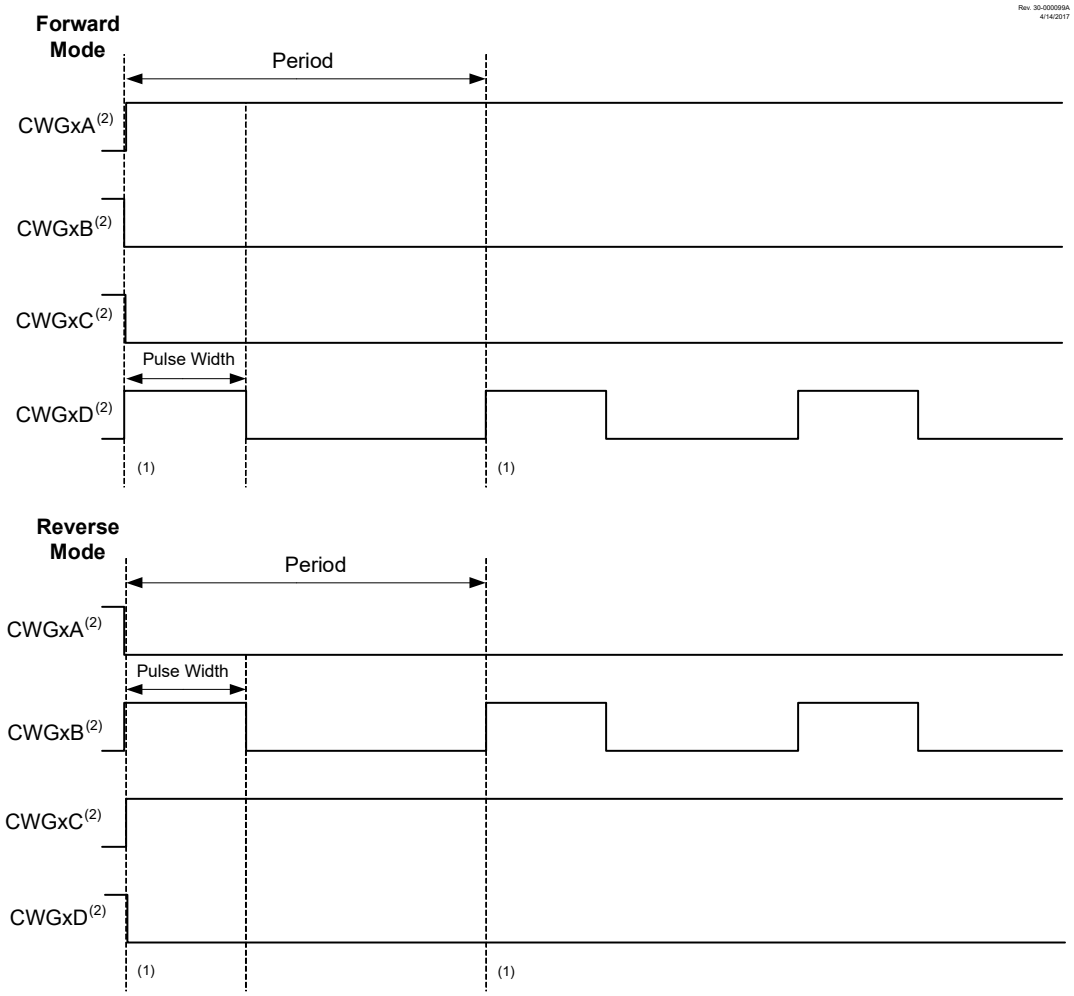


在正向全桥模式（MODE = 'b010）下，CWGxA 驱动为有效状态，CWGxB 和 CWGxC 驱动为无效状态，CWGxD 由输入信号进行调制，如图 27-7 所示。

在反向全桥模式（MODE = 'b011）下，CWGxC 驱动为有效状态，CWGxA 和 CWGxD 驱动为无效状态，CWGxB 由输入信号进行调制，如图 27-7 所示。

在全桥模式下，如果在正向与反向之间切换，会使用死区周期。死区控制一节对死区控制进行了介绍，更多详细信息请参见上升沿和反向死区一节和下降沿和正向死区一节。转向模式不与任一全桥模式配合使用。

图 27-7. 全桥输出示例



注：

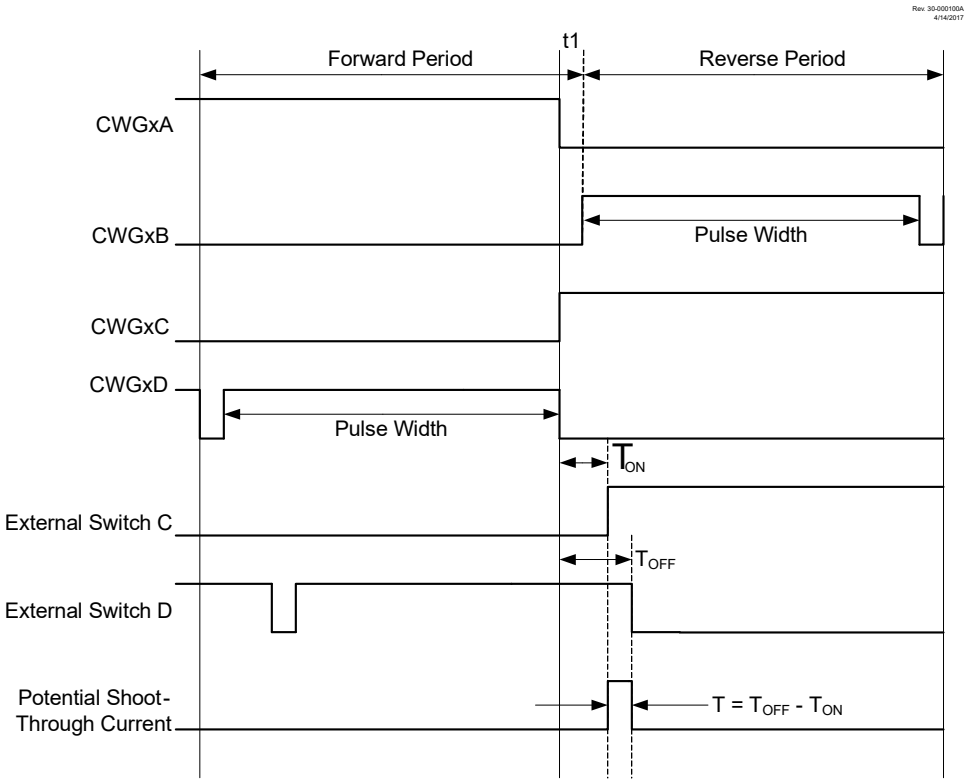
1. 上升的 CWG 数据输入会在调制输出上创建上升沿事件。
2. 输出信号显示为高电平有效；所有 POLy 位均被清零。

27.2.3.1. 全桥模式下的方向更改

在全桥模式下，更改 MODE[0]位控制正向/反向方向。将在调制输入的下一个上升沿更改方向。该序列如下所述，并在图 27-8 中进行了说明。

1. 相关的有效输出 CWGxA 和无效输出 CWGxC 切换为以相反的方向驱动。
2. 先前调制的输出 CWGxD 切换为无效状态，而先前无效的输出 CWGxB 开始调制。
3. 在经过方向切换死区后，恢复 CWG 调制。

图 27-8. 占空比接近 100%时 PWM 方向更改的示例



27.2.3.2. 全桥模式下的死区延时

在以下情况下，死区延时很重要：

- 当数据输入的占空比接近或等于 100%时，CWG 输出的方向发生更改
- 功率开关（包括功率器件和驱动电路）的关断时间大于导通时间

仅可在方向发生更改时插入死区延时，并且仅调制输出受到影响。静态配置输出（CWGxA 和 CWGxC）不提供死区，并且基本上同时开关。

图 27-8 给出了占空比接近 100%时，CWG 输出从正向变为反向的示例。在此示例中，在 t1 时刻 CWGxA 和 CWGxD 输出变为无效，而 CWGxC 输出变为有效。由于功率器件的关断时间比导通时间长，直通电流可能在时间段“T”内流过功率器件 QC 和 QD。当 CWG 方向从反向变为正向时，功率器件 QA 和 QB 上也会发生同样的现象。

如果应用需要在高占空比时更改 CWG 方向，则有两种方法可以避免出现直通电流：

1. 在更改方向前的一个周期减小 CWG 的占空比。
2. 使用可使开关元件的关断速度比导通速度更快的开关驱动器。

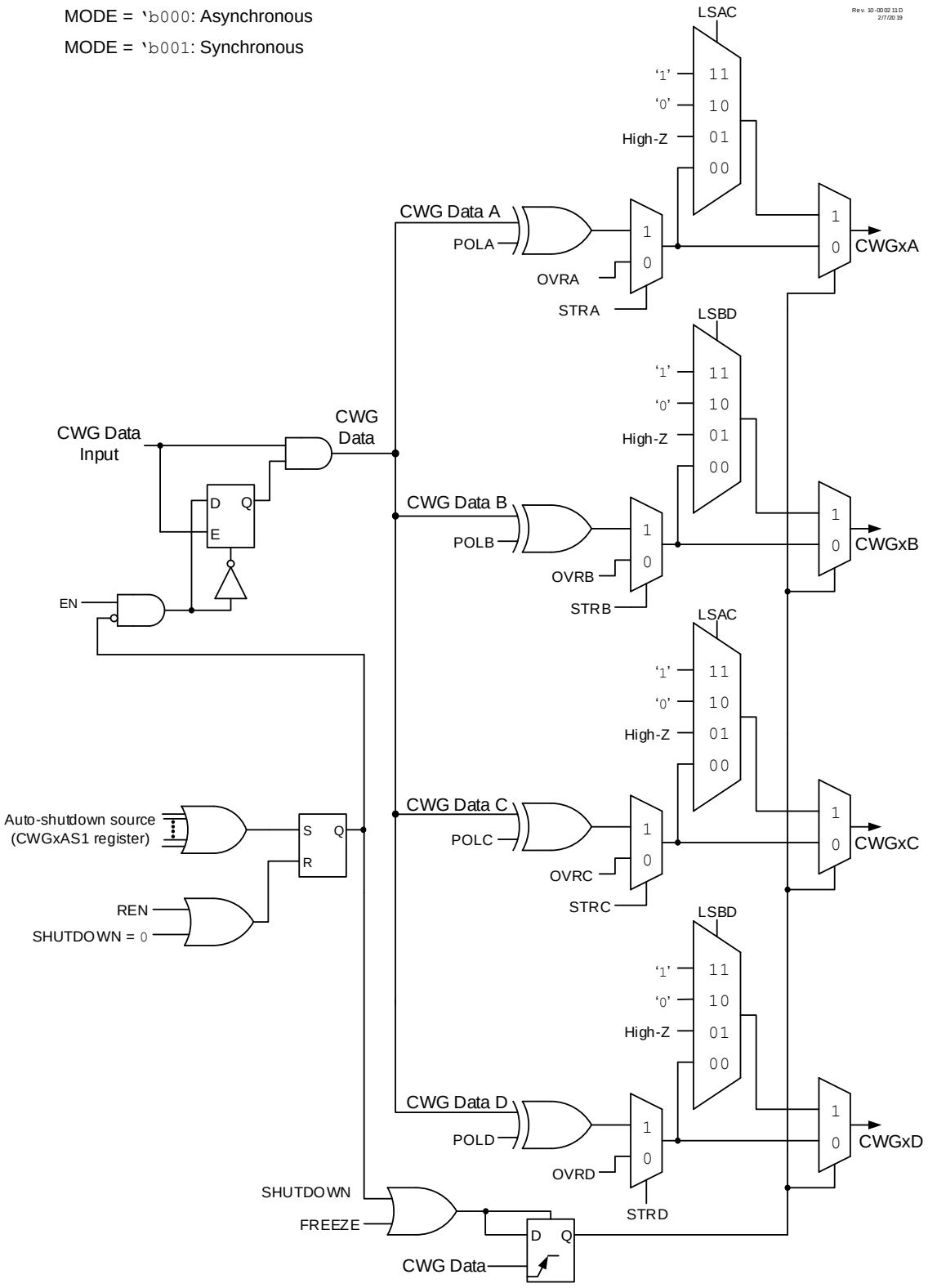
27.2.4. 转向模式

在同步和异步转向模式下，CWG 数据可以转向四个 CWG 输出的任意组合。所有未用于 PWM 输出的输出都会呈现固定值。每个输出具有独立的极性、转向和关断选项。死区控制不用于任何转向模式。

例如，当 STRA = 0 时，相应的引脚保持在由 OVRA 定义的电平。当 STRA = 1 时，引脚由 CWG 数据信号驱动。POLy 位仅在 STRy = 1 时控制信号极性。

CWG 自动关断工作原理也适用于转向模式，如自动关断一节所述。自动关断事件只会影响 STRy = 1 的引脚。

图 27-9. CWG 简化框图（输出转向模式）

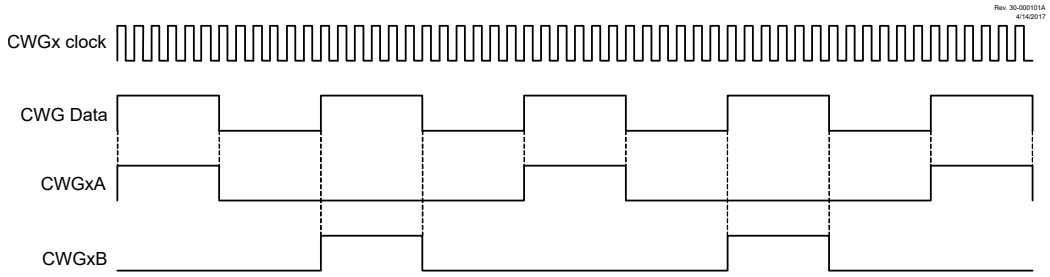


27.2.4.1. 同步转向模式

在同步转向模式下（MODE = 'b001），对转向选择寄存器的更改将在 CWG 数据的下一个上升沿生效（见下图）。在同步转向模式下，输出将始终产生完整波形。

重要：仅 STRx 位同步；OVRx 位不同步。

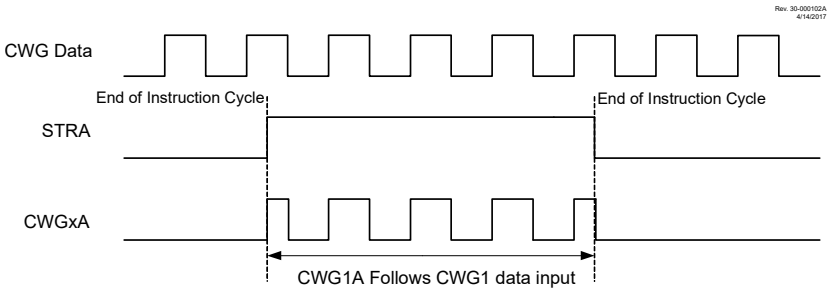
图 27-10. 同步转向示例（MODE = 'b001）



27.2.4.2. 异步转向模式

在异步模式下（MODE = 'b000），转向在写 STRx 的指令周期结束时生效。在异步转向模式下，输出信号可能是不完整波形（见下图）。此操作在用户固件需要立即除去该输出引脚的信号时非常有用。

图 27-11. 异步转向示例（MODE = 'b000）



27.2.4.3. 启动注意事项

应用硬件必须在 CWG 输出引脚上使用适当的外部上拉和 / 或下拉电阻。这是必需的，因为所有 I/O 引脚在复位时强制为高阻抗。

极性控制（POLy）位允许用户选择输出信号是高电平有效还是低电平有效。

27.3. 时钟源

时钟源用于驱动死区时序电路。CWG 模块允许选择以下时钟源：

- F_{OSC}（系统时钟）
- HFINTOSC

选择 HFINTOSC 时，HFINTOSC 将在休眠期间保持运行。因此，需要死区的 CWG 模式可在休眠模式下运行，前提是 CWG 数据输入也在休眠期间有效。使用 CS 位选择时钟源。系统时钟 F_{OSC} 在休眠状态下被禁止，因此无法使用死区控制。

27.4. 可选择的输入源

CWG 通过 ISM 位选择的输入源生成输出波形。更多详细信息，请参见 CWGxISM 寄存器。

27.5. 输出控制

27.5.1. CWG 输出

每个 CWG 输出可通过 RxyPPS 寄存器连接到外设引脚选择（PPS）输出。更多详细信息，请参见**“PPS——外设引脚选择模块”**一章。

27.5.2. 极性控制

每个 CWG 输出的极性可以单独进行选择。当输出极性位置 1 时，相应的输出为高电平有效。清零输出极性位时，相应输出将配置为低电平有效。然而，极性不会影响改写电平。输出极性通过 POLy 位来选择。自动关断和转向选项不受极性影响。

27.6. 死区控制

死区控制用于提供不重叠的互补输出，以防止输出开关时产生直通电流。在半桥和全桥模式下可采用死区操作。CWG 包含两个 6 位死区计数器。一个用于半桥模式下输入源控制的上升沿或用于全桥模式下的反向更改死区。另一个用于半桥模式下输入源控制的下降沿或用于全桥模式下的正向更改死区。

通过对 CWG 时钟周期计数来对死区进行计时，计数范围为零到上升沿或下降沿死区计数器寄存器中的值。

27.6.1. 半桥模式下的死区功能

在半桥模式下，死区计数器决定了正常输出的下降沿与反相输出的上升沿之间的延时，如图 27-1 所示。

27.6.2. 全桥模式下的死区功能

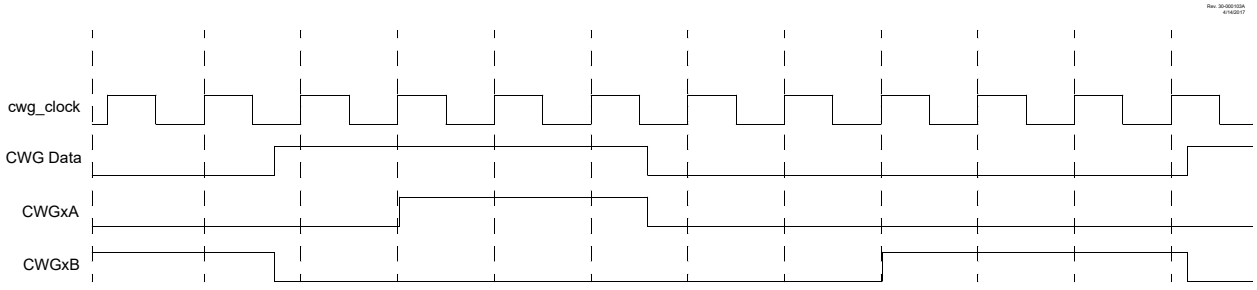
在全桥模式下，当方向更改时使用死区计数器。当 CWG 运行时，可以将 MODE[0]位置 1 或清零，以便从正向模式更改为反向模式。CWGxA 和 CWGxC 信号将在方向更改后的第一个输入上升沿立即改变，但调制信号（CWGxB 或 CWGxD，取决于更改的方向）将经历由死区计数器指示的一段延时。

27.7. 上升沿和反向死区

在半桥模式下，上升沿死区在 CWG 数据输入的上升沿之后延迟 CWGxA 输出的导通时间。在全桥模式下，仅在方向从正向模式变化为反向模式时才插入反向死区延时，并且仅调制输出 CWGxB 受到影响。

CWGxDBR 寄存器用于确定输入源信号上升沿死区时间间隔的持续时间。该持续时间为 0 至 64 个 CWG 时钟周期。下图说明了上升和下降 CWG 数据事件的不同死区延时。

图 27-12. 死区操作（CWGxDBR = 0x01，CWGxDBF = 0x02）



死区总是在输入源信号的边沿启动。计数为 0 表示不存在死区。

如果输入源信号在死区计数完成之前翻转极性，则相应输出上不会出现任何信号。

CWGxDBR 寄存器值是双重缓冲的。当 EN = 0 时，会在写入 CWGxDBR 时装入缓冲区。当 EN = 1 时，在 LD 位置 1 后将在 CWG 数据的第一个下降沿之后的上升沿装入缓冲区。

27.8. 下降沿和正向死区

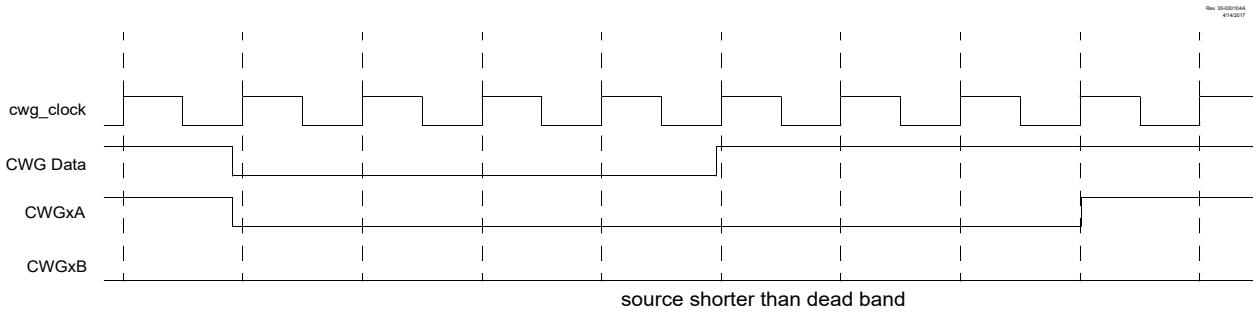
在半桥模式下，下降沿死区在 CWG 数据输入的下降沿延迟 CWGxB 输出的导通时间。在全桥模式下，仅在方向从反向模式变化为正向模式时才插入正向死区延时，并且仅调制输出 CWGxD 受到影响。

CWGxDBF 寄存器用于确定输入源信号下降沿死区时间间隔的持续时间。该持续时间为 0 至 64 个 CWG 时钟周期。

死区延时总是在输入源信号的边沿启动。计数为 0 表示不存在死区。

如果输入源信号在死区计数完成之前翻转极性，则相应输出上不会出现任何信号。

图 27-13. 死区操作（CWGxDBR = 0x03，CWGxDBF = 0x06，信号源短于死区）



CWGxDBF 寄存器值是双重缓冲的。当 EN = 0 时，会在写入 CWGxDBF 时装入缓冲区。当 EN = 1 时，在 LD 位置 1 后将在数据输入的第一个下降沿之后的上升沿装入缓冲区。

27.9. 死区抖动

当输入源的上升沿和下降沿与 CWG 时钟异步时，会在死区延时期间产生抖动。最大抖动等于 1 个 CWG 时钟周期。更多详细信息，请参见下面的公式。

公式 27-1. 死区延时计算

$$T_{DEAD-BAND\_MIN} = \frac{1}{F_{CWG\_CLOCK}} \bullet DBx$$

$$T_{DEAD-BAND\_MAX} = \frac{1}{F_{CWG\_CLOCK}} \bullet (DBx + 1)$$

$$T_{JITTER} = T_{DEAD-BAND\_MAX} - T_{DEAD-BAND\_MIN}$$

$$T_{JITTER} = \frac{1}{F_{CWG\_CLOCK}}$$

$$T_{DEAD-BAND\_MAX} = T_{DEAD-BAND\_MIN} + T_{JITTER}$$

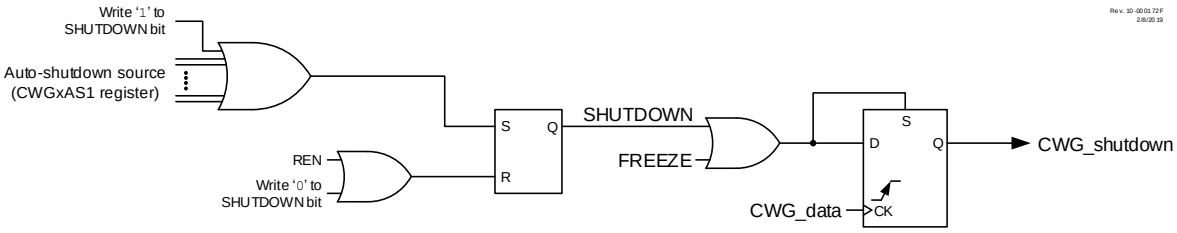
死区延时计算示例

$DBx = 0x0A = 10$
$F_{CWG\_CLOCK} = 8\ MHz$
$T_{JITTER} = \frac{1}{8\ MHz} = 125\ ns$
$T_{DEAD-BAND\_MIN} = 125\ ns \bullet 10 = 1.25\ \mu s$
$T_{DEAD-BAND\_MAX} = 1.25\ \mu s + 0.125\ \mu s = 1.37\ \mu s$

27.10. 自动关断

自动关断是一种使用特定改写信号立即改写 CWG 输出电平，从而安全关断电路的方法。关断状态可自动清除，也可保持到用软件清除。自动关断电路如下图所示。

图 27-14. CWG 关断框图



27.10.1. 关断

可通过以下两种方法之一进入关断状态：

- 软件生成
- 外部输入

27.10.2. 软件生成的关断

将 SHUTDOWN 位置 1 会使 CWG 强制进入关断状态。

禁止自动重启后，只要 SHUTDOWN 位置 1，关断状态就将持续。

在使能自动重启时，SHUTDOWN 位会自动清零，并在发生下一个上升沿事件时继续工作。SHUTDOWN 位指示何时存在关断条件。该位可由软件或硬件置 1 或清零。

27.10.3. 外部输入源

外部关断输入提供了在出现故障条件时安全暂停 CWG 工作的最快方式。当选定的任意关断输入变为有效时，CWG 输出会立即变为选定的改写电平，无任何软件延时。改写电平由 LSBD 和 LSAC 位进行选择。可以选择多个输入源来产生关断条件。所有输入源均为低电平有效。关断输入源由 ASyE 位单独使能。

重要：关闭输入对电平敏感，而不对边沿敏感。只要关断输入电平仍然存在，除非禁止自动关断，否则无法清除关断状态。

27.10.4. 引脚改写电平

在发生自动关断事件期间驱动到 CWG 输出的电平通过 LSBD 和 LSAC 位进行控制。LSBD 位控制 CWGxB/D 输出电平，而 LSAC 位控制 CWGxA/C 输出电平。

27.10.5. 自动关断中断

当发生自动关断事件时，可通过软件或硬件将 SHUTDOWN 置 1，以便将 PIRx 寄存器的 CWGxIF 标志位置 1。

27.11. 自动关断/重启

发生自动关断事件后，可使用两种方法来恢复工作：

- 软件控制
- 自动重启

在任一情况下，在进行重启之前必须清除关断源。即，必须除去关断条件，或者必须清零相应的 ASyE 位。

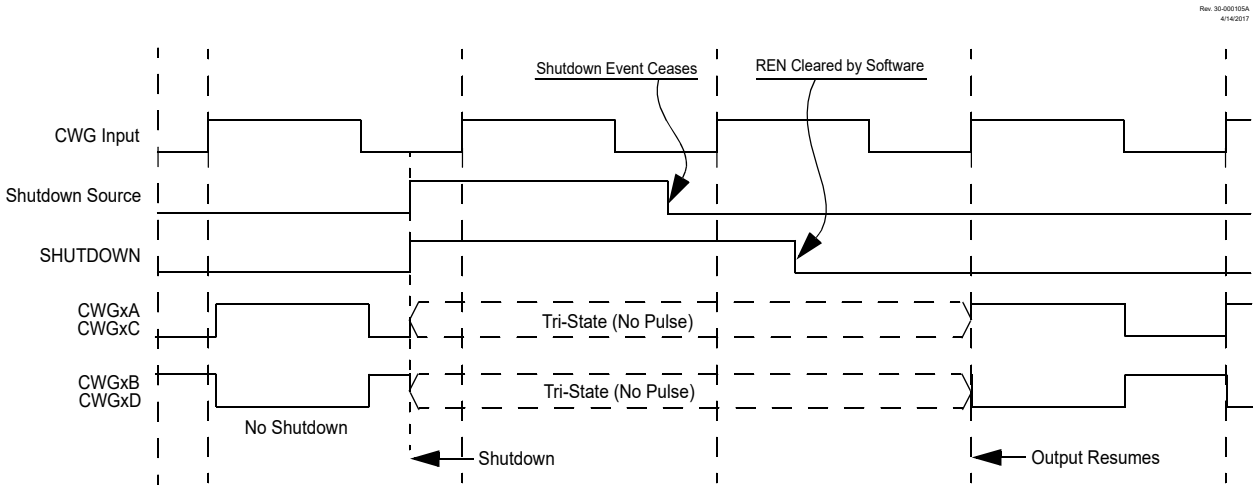
27.11.1. 软件控制重启

当 REN 位清零（REN = 0）时，CWG 模块必须在自动关断事件后通过软件重启。

一旦除去所有自动关断源后，软件必须清零 SHUTDOWN 位。SHUTDOWN 清零后，CWG 模块将在 CWG 数据输入的第一个上升沿继续工作。

重要：如果自动关断条件仍然存在，则无法用软件清零 SHUTDOWN 位。

图 27-15. 自动重启禁止时的关断功能（REN = 0，LSAC = 'b01，LSBD = 'b01）



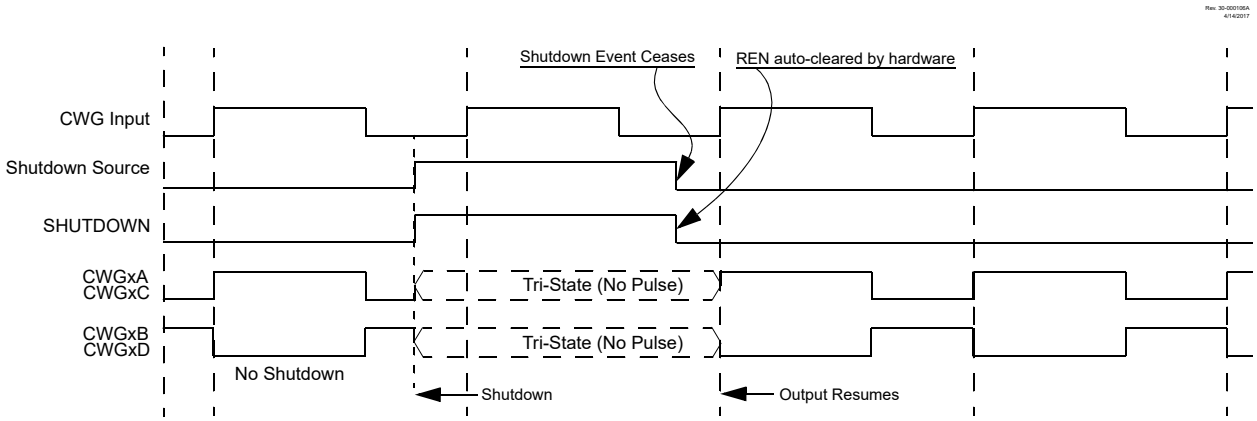
27.11.2. 自动重启

当 REN 位置 1（REN = 1）时，CWG 模块将从关断状态自动重启。

一旦除去所有自动关断条件后，硬件将自动清零 SHUTDOWN 位。SHUTDOWN 清零后，CWG 模块将在 CWG 数据输入的第一个上升沿继续工作。

重要：如果自动关断条件仍然存在，则无法用软件清零 SHUTDOWN 位。

图 27-16. 自动重启使能时的关断功能（REN = 1，LSAC = 'b01，LSBD = 'b01）



27.12. 休眠期间的操作

CWG 模块独立于系统时钟工作，只要选定的时钟源和输入源保持活动状态，它就会继续在休眠期间运行。

满足以下所有条件时，HFINTOSC 会在休眠期间保持活动状态：

- 使能 CWG 模块
- 输入源为活动状态
- 无论选择哪种系统时钟源，都会选择 HFINTOSC 作为时钟源。

换句话说，如果在 CWG 使能且输入源为活动状态时，将 HFINTOSC 同时选作系统时钟和 CWG 时钟源，则在休眠期间 CPU 会进入空闲状态，而 HFINTOSC 将保持活动状态并且 CWG 会继续工作。这会直接影响休眠模式的电流。

27.13. 配置 CWG

1. 确保将 CWG 输出对应的 TRIS 控制位置 1，从而将所有输出配置为输入，确保输出在设置期间无效。外部硬件必须确保引脚电平保持为安全电平。
2. 将 EN 位清零（如果尚未清零）。
3. 配置 MODE 位，以设置输出工作模式。
4. 配置 POLy 位，以设置输出极性。
5. 配置 ISM 位，以选择数据输入源。
6. 如果选择转向模式，则配置 STRy 位，以在 CWG 上选择所需输出。
7. 配置 LSBD 和 LSAC 位，以选择自动关断输出改写状态（即使未使用自动关断，这也是必需的，因为将从关断状态启动）。
8. 如果需要自动重启，则将 REN 位置 1。
9. 如果需要自动关断，则配置 ASyE 位，以选择关断源。
10. 通过 CWGxDBR 和 CWGxDBF 寄存器设置所需的上升和下降死区时间。
11. 通过 CS 位选择时钟源。
12. 将 EN 位置 1，以使能模块。
13. 将 CWG 输出对应的 TRIS 位清零，以将其设为输出。

如果要使用自动重启，则将 REN 位置 1，SHUTDOWN 位将会自动清零。否则，由软件清零 SHUTDOWN 位来启动 CWG。

27.14. 寄存器定义：CWG 控制

下表列出了 CWG 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 27-1. CWG 长位名称前缀

外设	位名称前缀
CWG1	CWG1

27.14.1. CWGxCON0

名称： CWGxCON0
偏移量： 0x0F3F

CWG 控制寄存器 0

位	7	6	5	4	3	2	1	0
	EN	LD				MODE[2:0]		
访问	R/W	R/W/HC				R/W	R/W	R/W
复位	0	0				0	0	0

Bit 7 – EN CWG 使能

值	说明
1	使能模块
0	禁止模块

Bit 6 – LD CWG1 装入缓冲区(1)

值	说明
1	在紧接着该位置 1 后第一个下降沿的 CWG 数据上升沿装入死区计数缓冲区
0	缓冲区保持不变

Bit 2:0 – MODE[2:0] CWG 模式

值	说明
111	保留
110	保留
101	CWG 输出在推挽模式下工作
100	CWG 输出在半桥模式下工作
011	CWG 输出在反向全桥模式下工作
010	CWG 输出在正向全桥模式下工作
001	CWG 输出在同步转向模式下工作
000	CWG 输出在异步转向模式下工作

注：

1. 该位只能在 EN = 1 后置 1；不能在 EN 置 1 的同一周期置 1。

27.14.2. CWGxCON1

名称： CWGxCON1
偏移量： 0x0F40

CWG 控制寄存器 1

位	7	6	5	4	3	2	1	0
			IN		POLD	POLC	POLB	POLA
访问			R		R/W	R/W	R/W	R/W
复位			x		0	0	0	0

Bit 5 – IN CWG 输入值（只读）

值	说明
1	CWG 数据输入是逻辑 1
0	CWG 数据输入是逻辑 0

Bit 0, 1, 2, 3 – POLy CWG 输出“y”极性

值	说明
1	信号输出极性翻转
0	信号输出为正常极性

27.14.3. CWGxCLK

名称： CWGxCLK
偏移量： 0x0F3B

CWG 时钟输入选择寄存器

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

Bit 0 – CS CWG 时钟源选择

值	说明
1	HFINTOSC（在休眠期间继续工作）
0	F_{OSC}

27.14.4. CWGxISM

名称： CWGxISM
偏移量： 0x0F3C

CWGx 输入选择寄存器

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

Bit 3:0 – ISM[3:0] CWG 数据输入源选择

ISM	输入选择
	CWG1
1111	CLC8_OUT(1)
1110	CLC7_OUT(1)
1101	CLC6_OUT(1)
1100	CLC5_OUT(1)
1011	CLC4_OUT(1)
1010	CLC3_OUT(1)
1001	CLC32_OUT(1)
1000	CLC1_OUT(1)
0111	DSM_OUT
0110	C2_OUT
0101	C1_OUT
0100	PWM4_OUT
0011	PWM3_OUT
0010	CCP2_OUT
0001	CCP1_OUT
0000	通过 CWG1PPS 选择的引脚

注：
1. CN2510 器件上未提供。

27.14.5. CWGxSTR

名称： CWGxSTR
偏移量： 0x0F43

CWG 转向控制寄存器[1]

位	7	6	5	4	3	2	1	0
	OVRD	OVRC	OVRB	OVRA	STRD	STRC	STRB	STRA
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 4, 5, 6, 7 – OVRy 转向数据 OVR'y'

值	条件	说明
x	STRy = 1	CWGx'y'输出具有 CWG 数据输入波形，其极性由 POLy 位控制
1	STRy = 0 且 POLy = x	CWGx'y'输出为高电平
0	STRy = 0 且 POLy = x	CWGx'y'输出为低电平

Bit 0, 1, 2, 3 – STRy STR'y'转向使能[2]

值	说明
1	CWGx'y'输出具有 CWG 数据输入波形，其极性由 POLy 位控制
0	CWGx'y'输出指定为 OVRy 位的值

注：

1. 该寄存器中的位仅在 MODE ='b00x（CWGxCON0，转向模式）时适用。
2. MODE = 'b001 时，该位是双重缓冲的。

27.14.6. CWGxAS0

名称： CWGxAS0
偏移量： 0x0F41

CWG 自动关断控制寄存器 0

位	7	6	5	4	3	2	1	0
	SHUTDOWN	REN	LSBD[1:0]		LSAC[1:0]			
访问	R/W/HS/HC	R/W	R/W	R/W	R/W	R/W		
复位	0	0	0	1	0	1		

Bit 7 – SHUTDOWN 自动关断事件状态(1,2)

值	说明
1	自动关断状态有效
0	未发生自动关断事件

Bit 6 – REN 自动重启使能

值	说明
1	使能自动重启
0	禁止自动重启

Bit 5:4 – LSBD[1:0] CWGxB 和 CWGxD 自动关断状态控制

值	说明
11	发生自动关断事件时，将 CWGxB/D 置为逻辑 1
10	发生自动关断事件时，将 CWGxB/D 置为逻辑 1
01	发生自动关断事件时，CWGxB/D 上的引脚处于三态
00	发生自动关断事件时，在所需的死区时间间隔之后，将 CWGxB/D 上的引脚置为无效状态（包括极性）

Bit 3:2 – LSAC[1:0] CWGxA 和 CWGxC 自动关断状态控制

值	说明
11	发生自动关断事件时，将 CWGxA/C 置为逻辑 1
10	发生自动关断事件时，将 CWGxA/C 置为逻辑 1
01	发生自动关断事件时，CWGxA/C 上的引脚处于三态
00	发生自动关断事件时，在所需的死区时间间隔之后，将 CWGxA/C 上的引脚置为无效状态（包括极性）

注：

1. 在 EN = 0 时，可以写入该位，将输出置为关断配置。
2. 输出将一直保持在自动关断状态，直到该位清零后出现 CWG 数据输入的下一个上升沿为止。

27.14.7. CWGxAS1

名称： CWGxAS1
偏移量： 0x0F42

CWG 自动关断控制寄存器 1

位	7	6	5	4	3	2	1	0
	AS7E	AS6E	AS5E	AS4E	AS3E	AS2E	AS1E	AS0E
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 – ASyE CWG 自动关断源使能[1,2]

ASyE	自动关断源
	CWG1
AS7E	CLC6_OUT（低电平会导致关断）(1)
AS6E	CLC2_OUT（低电平会导致关断）(1)
AS5E	C2_OUT（低电平会导致关断）
AS4E	C1_OUT（低电平会导致关断）
AS3E	TMR6_postscaled（高电平会导致关断）
AS2E	TMR4_postscaled（高电平会导致关断）
AS1E	TMR2_postscaled（高电平会导致关断）
AS0E	通过 CWG1PPS 选择的引脚（低电平会导致关断）

注：
1. CN2510 器件上未提供。

值	说明
1	已选择 ASyE 的自动关断源（见上表）
0	未选择自动关断源

注：
1. 当 EN = 0 时，可以写入该位，将输出置为关断配置。
2. 输出将一直保持在自动关断状态，直到该位清零后出现 CWG 数据输入的下一个上升沿为止。

27.14.8. CWGxDBR

名称： CWGxDBR
偏移量： 0x0F3D

CWG 上升沿死区计数寄存器

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

Bit 5:0 – DBR[5:0] CWG 上升沿触发死区计数

复位状态： POR/BOR = xxxxxx
所有其他复位 = uuuuuu

值	说明
xxxxxx	死区在上升沿之后保持有效的时间不少于 n 个 CWG 时钟周期且不多于 n+1 个 CWG 时钟周期（n = DBR[5:0]）。
000000	0 个 CWG 时钟周期。死区生成被旁路。

27.14.9. CWGxDBF

名称： CWGxDBF
偏移量： 0x0F3E

CWG 下降沿死区计数寄存器

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

Bit 5:0 – DBF[5:0] CWG 下降沿触发死区计数

复位状态： POR/BOR = xxxxxx
所有其他复位 = uuuuuu

值	说明
xxxxxx	死区在下降沿之后保持有效的时间不少于 n 个 CWG 时钟周期且不多于 n+1 个 CWG 时钟周期（n = DBF[5:0]）。
000000	0 个 CWG 时钟周期。死区生成被旁路。

27.15. 寄存器汇总——CWG

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0F3A	保留									
0x0F3B	CWG1CLK	7:0								CS
0x0F3C	CWG1ISM	7:0					ISM[3:0]			
0x0F3D	CWG1DBR	7:0			DBR[5:0]					
0x0F3E	CWG1DBF	7:0			DBF[5:0]					
0x0F3F	CWG1CON0	7:0	EN	LD				MODE[2:0]		
0x0F40	CWG1CON1	7:0			IN		POLD	POLC	POLB	POLA
0x0F41	CWG1AS0	7:0	SHUTDOWN	REN	LSBD[1:0]		LSAC[1:0]			
0x0F42	CWG1AS1	7:0	AS7E	AS6E	AS5E	AS4E	AS3E	AS2E	AS1E	AS0E
0x0F43	CWG1STR	7:0	OVRD	OVRC	OVRB	OVRA	STRD	STRC	STRB	STRA

28. CLC——可配置逻辑单元

可配置逻辑单元（CLC）模块提供工作速度可超越软件执行速度限制而工作的可编程逻辑。该逻辑单元最多可接收 64 个输入信号，并通过使用可配置门将 64 个输入缩减为 4 条驱动 8 种可选单输出逻辑功能之一的逻辑线。

重要：CH2510 器件上未提供 CLC 模块。

输入源是以下信号源的组合：

- I/O 引脚
- 内部时钟
- 外设
- 寄存器位

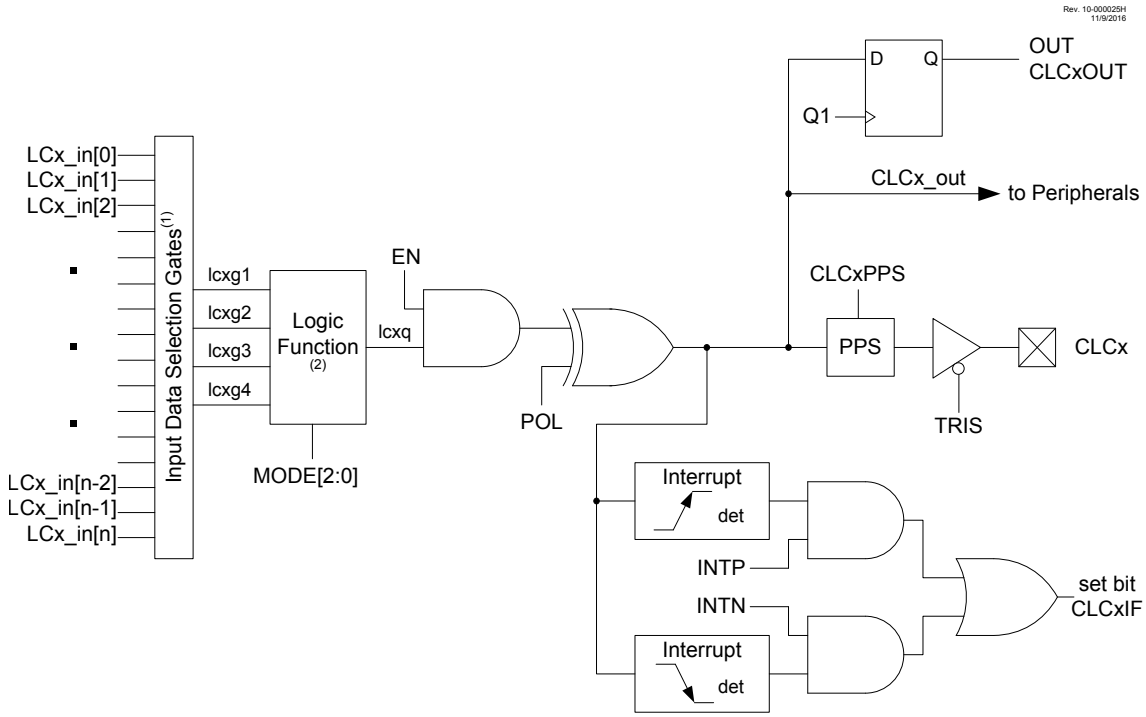
可将输出内部连接到外设和输出引脚。

重要：该器件上有多个 CLC 实例。在本章中，寄存器名称中的小写字母“x”泛指 CLC 实例编号。例如，控制寄存器的第一个实例是 CLC1CON，在本章中统称为 CLCxCON。

下图为显示通过 CLC 的信号流的简化框图。可能的配置包括：

- 组合逻辑：
  - AND
  - NAND
  - AND-OR
  - AND-OR-INVERT
  - OR-XOR
  - OR-XNOR
- 锁存器：
  - S-R
  - 带置 1 和复位功能的时钟控制 D 型锁存器
  - 带置 1 和复位功能的透明 D 型锁存器
  - 带复位功能的时钟控制 J-K 型锁存器

图 28-1. CLC 简化框图



注：

1. 有关输入数据选择和门控的信息，请参见图 28-2。
2. 有关可编程逻辑功能的信息，请参见图 28-3。

28.1. CLC 设置

CLC 模块的编程通过配置逻辑信号流中的 4 级来实现。这 4 级为：

- 数据选择
- 数据门控
- 逻辑功能选择
- 输出极性

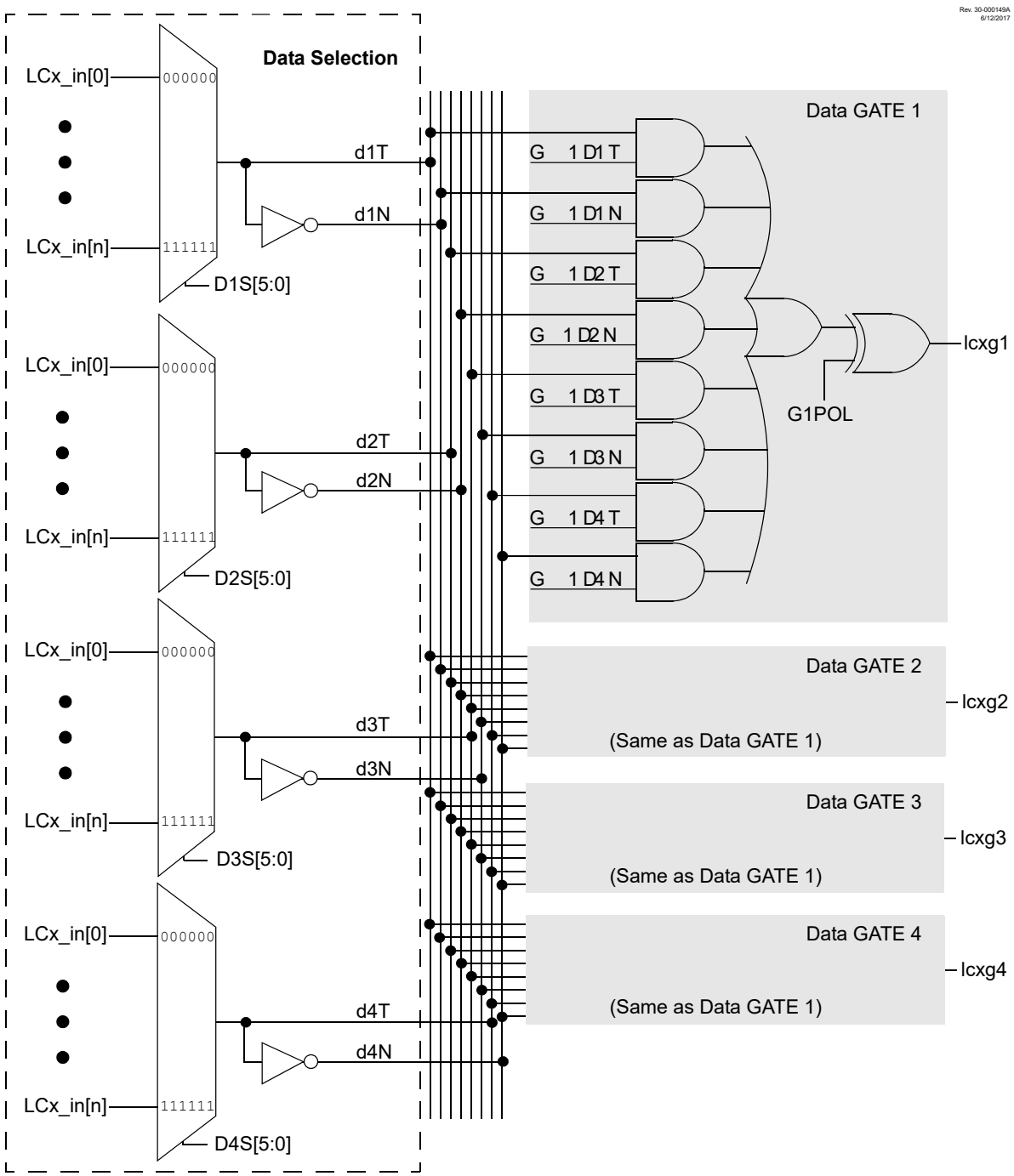
每级都可在运行时通过写入相应的 CLC 特殊功能寄存器（SFR）来进行设置。这具有支持在程序执行期间运行时执行逻辑重新配置的额外优点。

28.1.1. 数据选择

有 64 个信号可用作可配置逻辑的输入。使用 4 个 64 输入多路开关来选择要传递到下一级的输入。

数据选择通过下图左侧所示的 4 个多路开关来进行。图中的数据输入采用通用编号的输入名称表示。

图 28-2. 输入数据选择和门控



Note: All controls are undefined at power-up.

CLCxSEL0、CLCxSEL1、CLCxSEL2 和 CLCxSEL3 数据输入源寄存器用于将每个 CLC 模块中的通用输入名称与实际信号相关联。标有“DyS 值”的列给出了选定数据输入的多路开关选择代码。DyS 是多路开关选择输入代码的缩写：D1S 到 D4S，其中“y”为门编号。

重要：数据选择在上电时是未定义的。

28.1.2. 数据门控

来自输入多路开关的输出将通过数据门控级转送到所需的逻辑功能输入。每个数据门可以转送由 4 个选定输入组成的任意组合。

门级不仅仅是信号转送。可将门配置为将每个输入信号转换为反相或不反相数据。在每个门中，将转送后的信号进行逻辑与运算。每个门的输出可以先进行反相，然后再进入逻辑功能级。

门控实际上是一个 1 至 4 输入的 AND/NAND/OR/NOR 门。如果将每个输入和输出进行反相，则该门的作用是对所有已使能数据输入进行逻辑与。如果输入和输出不进行反相，则该门的作用是对所有已使能输入进行逻辑或。

下表总结了可以通过使用门逻辑选择位在门 1 中获得的基本逻辑。该表列出了具有 4 个输入变量的逻辑，但每个门均可配置为使用少于 4 个输入。如果未选择任何输入，则输出将为 0 或 1，具体取决于门输出极性位。

表 28-1. 数据门控逻辑

CLCxGLSy	GyPOL	门逻辑
0x55	1	AND
0x55	0	NAND
0xAA	1	NOR
0xAA	0	OR
0x00	0	逻辑 0
0x00	1	逻辑 1

用户可以（但建议不要）同时选择同一输入的真值和取反值。如果这样做，则无论其他输入为何，门的输出都将为 0，但可能会出现逻辑毛刺（瞬态电流引起的脉冲）。如果通道的输出必须为 0 或 1，则建议将所有门位设置为 0，并使用门极性位设置所需的电平。

数据门控使用如下逻辑门选择寄存器进行配置：

- 门 1：CLCxGLS0
- 门 2： CLCxGLS1
- 门 3： CLCxGLS2
- 门 4： CLCxGLS3

寄存器编号后缀不同于门编号，这是因为该模块的其他形式在同一寄存器中具有多种门选择。

图 28-2 右侧给出了数据门控的图示。其中仅对一个门进行了详细说明。其余三个门的配置相同，只是数据使能分别对应于相应门的使能信号。

28.1.3. 逻辑功能

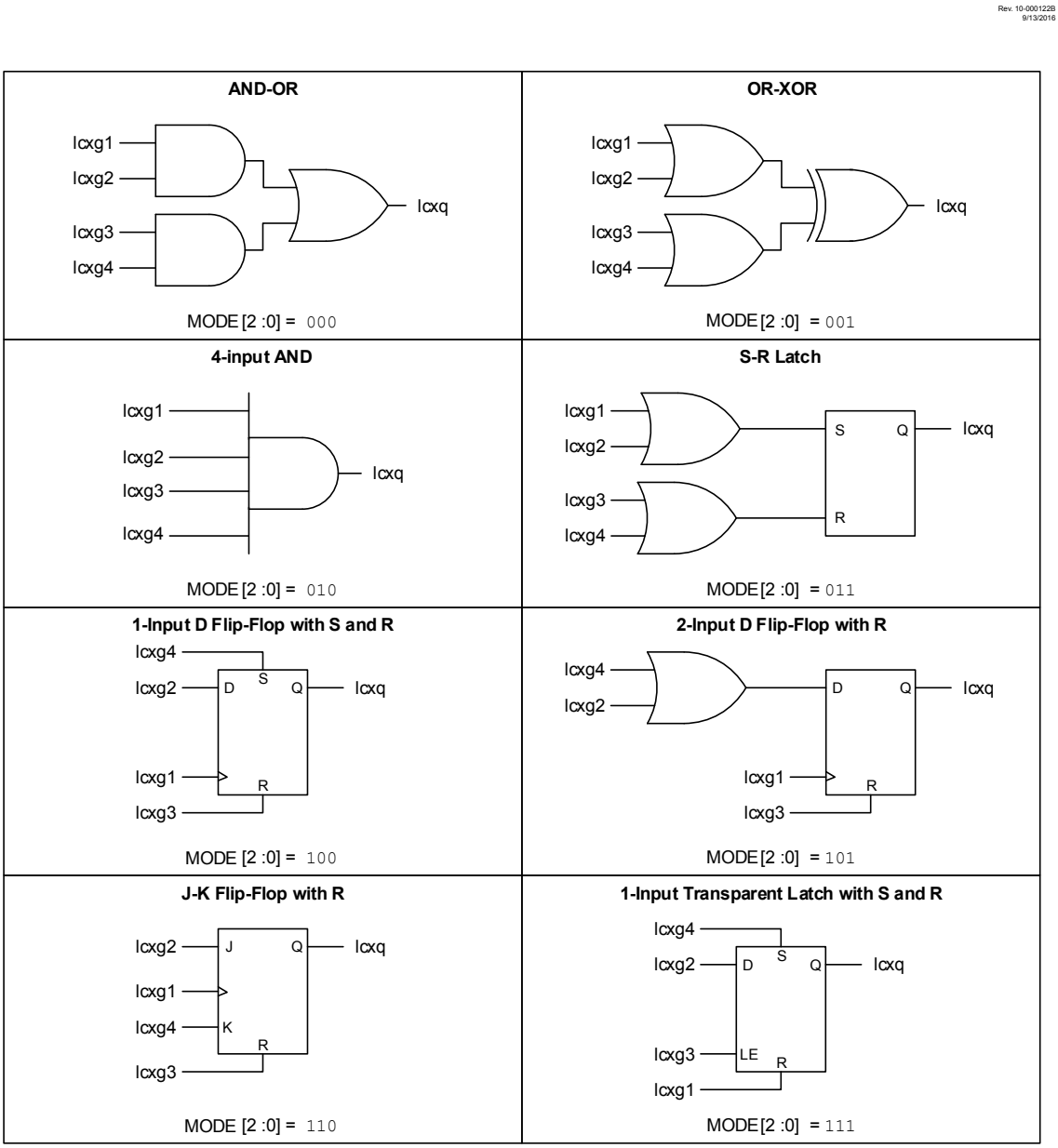
有 8 种可用的逻辑功能，包括：

- AND-OR
- OR-XOR
- AND
- S-R 锁存器
- 带置 1 和复位功能的 D 型触发器

- 带复位功能的 D 型触发器
- 带复位功能的 J-K 型触发器
- 带置 1 和复位功能的透明锁存器

逻辑功能如下图所示。每种逻辑功能都有 4 个输入和 1 个输出。这 4 个输入为前一级的 4 个数据门输出。输出馈送到反相级，接着送到其他外设、输出引脚，然后回到 CLC。

图 28-3. 可编程逻辑功能



28.1.4. 输出极性

CLC 中的最后一级是输出极性。将 CLCxPOL 寄存器中的 POL 位置 1 时，来自逻辑级的输出信号会进行反相。如果在允许中断时改变极性会导致输出结果的变化产生中断。

28.2. CLC 中断

如果相应的中断允许位置 1，则在 CLCx 的输出值改变时，将会产生中断。因此，每个 CLC 中都具有一个上升沿检测器和一个下降沿检测器。

触发其中一个边沿检测器，且其相关的中断允许位置 1 时，相关 PIR 寄存器的 CLCxIF 位会置 1。INTP 位用于允许上升沿中断，INTN 位用于允许下降沿中断。

要完全允许中断，需要将以下位置 1：

- 相应 PIE 寄存器的 CLCxIE 位
- INTP 位（对于上升沿检测）
- INTN 位（对于下降沿检测）
- 如果不使用优先级中断
  - 将 INTCON 寄存器的 IPEN 位清零
  - 将 INTCON 寄存器的 GIE 位置 1
  - 将 INTCON 寄存器的 PEIE 位置 1
- 如果 CLC 为高优先级中断
  - 将 INTCON 寄存器的 IPEN 位置 1
  - 将相应 IPR 寄存器的 CLCxIP 位置 1
  - 将 INTCON 寄存器的 GIEH 位置 1
- 如果 CLC 为低优先级中断
  - 将 INTCON 寄存器的 IPEN 位置 1
  - 将相应 IPR 寄存器的 CLCxIP 位清零
  - 将 INTCON 寄存器的 GIEL 位置 1

作为中断服务程序的一部分，必须用软件将相应 PIR 寄存器的 CLCxIF 位清零。如果在清零该标志时检测到另一个边沿，则标志仍然会在序列结束时置 1。

28.3. 输出镜像副本

所有 CLCxOUT 位的镜像副本包含在 CLCDATA 寄存器中。读取该寄存器将同时读取所有 CLC 的输出。这可以防止由于测试或读取各个 CLCxCON 寄存器中的 OUT 位而导致读取偏差。

28.4. 复位的影响

发生复位后，CLCxCON 寄存器会清零。所有其他选择和门值保持不变。

28.5. 休眠期间的操作

CLC 模块独立于系统时钟工作，只要选定的输入源保持活动状态，它就会继续在休眠期间运行。

如果使能了 CLC 模块，并且选择 HFINTOSC 作为输入源，则无论所选择的系统时钟源如何，HFINTOSC 都会在休眠期间保持活动状态。

换句话说，如果在 CLC 使能时，同时选择 HFINTOSC 作为系统时钟和 CLC 输入源，则在休眠期间 CPU 会进入空闲状态，而 CLC 会继续工作，并且 HFINTOSC 将保持活动状态。

这会直接影响休眠模式的电流。

28.6. CLC 设置步骤

设置 CLC 时遵循以下步骤：

- 通过清零 EN 位禁止 CLC
- 使用 CLCxSEL0 至 CLCxSEL3 寄存器选择所需的输入

- 清零所有关联的 ANSEL 位
- 将与输入关联的所有 TRIS 位置 1
- 使用 CLCxGLS0 至 CLCxGLS3 寄存器通过 4 个门来使能所选输入
- 使用 GyPOL 位选择门输出极性
- 使用 MODE 位选择所需的逻辑功能
- 使用 POL 位选择所需的逻辑输出极性（该步骤可与前面的门输出极性步骤结合）
- 如果驱动某个器件引脚，则设置所需的引脚 PPS 控制寄存器，并另外清零对应于该输出的 TRIS 位
- 配置中断（可选）。请参见 CLC 中断
- 通过将 EN 位置 1，使能 CLC

28.7. 寄存器定义：可配置逻辑单元

28.7.1. CLCDATA

名称： CLCDATA
偏移量： 0xE77

CLC 数据输出寄存器

镜像副本

位	7	6	5	4	3	2	1	0
	MLC8OUT	MLC7OUT	MLC6OUT	MLC5OUT	MLC4OUT	MLC3OUT	MLC2OUT	MLC1OUT
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 – MLCxOUT

CLCx_out 的镜像副本位

值	说明
1	CLCx_out 为 1
0	CLCx_out 为 0

28.7.2. CLCxCON

名称： CLCxCON
偏移量： 0xE27,0xE31,0xE3B,0xE45,0xE4F,0xE59,0xE63,0xE6D

可配置逻辑单元控制寄存器

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

Bit 7 – EN
CLC 使能位

值	说明
1	使能可配置逻辑单元，并混合信号
0	禁止可配置逻辑单元，并输出逻辑 0

Bit 5 – OUT
经过 LCPOL 之后的逻辑单元输出数据。从 CLCxOUT 采样

Bit 4 – INTP
可配置逻辑单元上升边沿中断允许位

值	说明
1	CLCxIF 将在 CLCxOUT 上出现上升沿时置 1
0	CLCxOUT 的上升沿对 CLCxIF 没有影响

Bit 3 – INTN
可配置逻辑单元下降沿中断允许位

值	说明
1	CLCxIF 将在 CLCxOUT 上出现下降沿时置 1
0	CLCxOUT 的下降沿对 CLCxIF 没有影响

Bit 2:0 – MODE[2:0]
可配置逻辑单元功能模式选择位

值	说明
111	单元是带置 1 和复位功能的 1 输入透明锁存器
110	单元是带复位功能的 J-K 型触发器
101	单元是带复位功能的 2 输入 D 型触发器
100	单元是带置 1 和复位功能的 1 输入 D 型触发器
011	单元是 S-R 型锁存器
010	单元是 4 输入 AND 逻辑
001	单元是 OR-XOR 逻辑
000	单元是 AND-OR 逻辑

28.7.3. CLCxPOL

名称： CLCxPOL
偏移量： 0xE28,0xE32,0xE3C,0xE46,0xE50,0xE5A,0xE64,0xE6E

信号极性控制寄存器

位	7	6	5	4	3	2	1	0
	POL				G4POL	G3POL	G2POL	G1POL
访问	R/W				R/W	R/W	R/W	R/W
复位	0				x	x	x	x

Bit 7 – POL

CLCxOUT 输出极性控制位

值	说明
1	逻辑单元的输出反相
0	逻辑单元的输出不反相

Bit 0, 1, 2, 3 – GyPOL

门输出极性控制位

复位状态： 默认值 = xxxx
POR/BOR = x
所有其他复位时的值 = u

值	说明
1	门输出在施加到逻辑单元时反相
0	门输出不反相

28.7.4. CLCxSEL0

名称： CLCxSEL0
偏移量： 0xE29,0xE33,0xE3D,0xE47,0xE51,0xE5B,0xE65,0xE6F

通用 CLCx 数据 1 选择寄存器

位	7	6	5	4	3	2	1	0
			D1S[5:0]					
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			x	x	x	x	x	x

Bit 5:0 – D1S[5:0] CLCx 数据 1 输入选择位

DyS 值	CLC 输入源	DyS 值	CLC 输入源
111111 [63]	保留	011111 [31]	IOC 中断
111110 [62]	保留	011110 [30]	ZCD_out
111101 [61]	保留	011101 [29]	C2_out
111100 [60]	保留	011100 [28]	C1_out
111011 [59]	保留	011011 [27]	PWM4_out
111010 [58]	保留	011010 [26]	PWM3_out
111001 [57]	保留	011001 [25]	CCP2_out
111000 [56]	保留	011000 [24]	CCP1 _out
110111 [55]	保留	010111 [23]	TMR6_out
110110 [54]	保留	010110 [22]	TMR5 _overflow
110101 [53]	保留	010101 [21]	TMR4 _out
110100 [52]	保留	010100 [20]	TMR3 _overflow
110011 [51]	保留	010011 [19]	TMR2 _out
110010 [50]	CWG1B	010010 [18]	TMR1 _overflow
110001 [49]	CWG1A	010001 [17]	TMR0 _overflow
110000 [48]	SCK2	010000 [16]	CLKR _out
101111 [47]	SDO2	001111 [15]	ADCRC
101110 [46]	SCK1	001110 [14]	SOSC
101101 [45]	SDO1	001101 [13]	SFINTOSC（1 MHz）
101100 [44]	EUSART2_TX/CK_out(1)	001100 [12]	MFINTOSC（32 kHz）
101011 [43]	EUSART2_DT_out(1)	001011 [11]	MFINTOSC（500 kHz）
101010 [42]	EUSART1_TX/CK_out	001010 [10]	LFINTOSC
101001 [41]	EUSART1_DT_out	001001 [9]	HFINTOSC
101000 [40]	CLC8_out(1)	001000 [8]	F_{OSC}
100111 [39]	CLC7_out(1)	000111 [7]	CLCIN7PPS(1)
100110 [38]	CLC6_out(1)	000110 [6]	CLCIN6PPS(1)
100101 [37]	CLC5_out(1)	000101 [5]	CLCIN5PPS(1)
100100 [36]	CLC4_out(1)	000100 [4]	CLCIN4PPS(1)
100011 [35]	CLC3_out(1)	000011 [3]	CLCIN3PPS(1)
100010 [34]	CLC2_out(1)	000010 [2]	CLCIN2PPS(1)
100001 [33]	CLC1_out(1)	000001 [1]	CLCIN1PPS(1)
100000 [32]	DSM1_out	000000 [0]	CLCIN0PPS(1)

注：
1. CN2510 器件上未提供。

复位状态： POR/BOR = xxxxxx
所有其他复位 = uuuuuu

28.7.5. CLCxSEL1

名称： CLCxSEL1
偏移量： 0xE2A,0xE34,0xE3E,0xE48,0xE52,0xE5C,0xE66,0xE70

通用 CLCx 数据 2 选择寄存器

位	7	6	5	4	3	2	1	0
			D2S[5:0]					
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			x	x	x	x	x	x

Bit 5:0 – D2S[5:0] CLCx 数据 2 输入选择位

DyS 值	CLC 输入源	DyS 值	CLC 输入源
111111 [63]	保留	011111 [31]	IOC 中断
111110 [62]	保留	011110 [30]	ZCD_out
111101 [61]	保留	011101 [29]	C2_out
111100 [60]	保留	011100 [28]	C1_out
111011 [59]	保留	011011 [27]	PWM4_out
111010 [58]	保留	011010 [26]	PWM3_out
111001 [57]	保留	011001 [25]	CCP2_out
111000 [56]	保留	011000 [24]	CCP1 _out
110111 [55]	保留	010111 [23]	TMR6_out
110110 [54]	保留	010110 [22]	TMR5 _overflow
110101 [53]	保留	010101 [21]	TMR4 _out
110100 [52]	保留	010100 [20]	TMR3 _overflow
110011 [51]	保留	010011 [19]	TMR2 _out
110010 [50]	CWG1B	010010 [18]	TMR1 _overflow
110001 [49]	CWG1A	010001 [17]	TMR0 _overflow
110000 [48]	SCK2	010000 [16]	CLKR _out
101111 [47]	SDO2	001111 [15]	ADCRC
101110 [46]	SCK1	001110 [14]	SOSC
101101 [45]	SDO1	001101 [13]	SFINTOSC（1 MHz）
101100 [44]	EUSART2_TX/CK_out(1)	001100 [12]	MFINTOSC（32 kHz）
101011 [43]	EUSART2_DT_out(1)	001011 [11]	MFINTOSC（500 kHz）
101010 [42]	EUSART1_TX/CK_out	001010 [10]	LFINTOSC
101001 [41]	EUSART1_DT_out	001001 [9]	HFINTOSC
101000 [40]	CLC8_out(1)	001000 [8]	F_{OSC}
100111 [39]	CLC7_out(1)	000111 [7]	CLCIN7PPS(1)
100110 [38]	CLC6_out(1)	000110 [6]	CLCIN6PPS(1)
100101 [37]	CLC5_out(1)	000101 [5]	CLCIN5PPS(1)
100100 [36]	CLC4_out(1)	000100 [4]	CLCIN4PPS(1)
100011 [35]	CLC3_out(1)	000011 [3]	CLCIN3PPS(1)
100010 [34]	CLC2_out(1)	000010 [2]	CLCIN2PPS(1)
100001 [33]	CLC1_out(1)	000001 [1]	CLCIN1PPS(1)
100000 [32]	DSM1_out	000000 [0]	CLCIN0PPS(1)

注：
1. CN2510 器件上未提供。

复位状态： POR/BOR = xxxxxx
所有其他复位 = uuuuuu

28.7.6. CLCxSEL2

名称： CLCxSEL2
偏移量： 0xE2B,0xE35,0xE3F,0xE49,0xE53,0xE5D,0xE67,0xE71

通用 CLCx 数据 3 选择寄存器

位	7	6	5	4	3	2	1	0
			D3S[5:0]					
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			x	x	x	x	x	x

Bit 5:0 – D3S[5:0] CLCx 数据 3 输入选择位

DyS 值	CLC 输入源	DyS 值	CLC 输入源
111111 [63]	保留	011111 [31]	IOC 中断
111110 [62]	保留	011110 [30]	ZCD_out
111101 [61]	保留	011101 [29]	C2_out
111100 [60]	保留	011100 [28]	C1_out
111011 [59]	保留	011011 [27]	PWM4_out
111010 [58]	保留	011010 [26]	PWM3_out
111001 [57]	保留	011001 [25]	CCP2_out
111000 [56]	保留	011000 [24]	CCP1 _out
110111 [55]	保留	010111 [23]	TMR6_out
110110 [54]	保留	010110 [22]	TMR5 _overflow
110101 [53]	保留	010101 [21]	TMR4 _out
110100 [52]	保留	010100 [20]	TMR3 _overflow
110011 [51]	保留	010011 [19]	TMR2 _out
110010 [50]	CWG1B	010010 [18]	TMR1 _overflow
110001 [49]	CWG1A	010001 [17]	TMR0 _overflow
110000 [48]	SCK2	010000 [16]	CLKR _out
101111 [47]	SDO2	001111 [15]	ADCRC
101110 [46]	SCK1	001110 [14]	SOSC
101101 [45]	SDO1	001101 [13]	SFINTOSC（1 MHz）
101100 [44]	EUSART2_TX/CK_out(1)	001100 [12]	MFINTOSC（32 kHz）
101011 [43]	EUSART2_DT_out(1)	001011 [11]	MFINTOSC（500 kHz）
101010 [42]	EUSART1_TX/CK_out	001010 [10]	LFINTOSC
101001 [41]	EUSART1_DT_out	001001 [9]	HFINTOSC
101000 [40]	CLC8_out(1)	001000 [8]	F_{OSC}
100111 [39]	CLC7_out(1)	000111 [7]	CLCIN7PPS(1)
100110 [38]	CLC6_out(1)	000110 [6]	CLCIN6PPS(1)
100101 [37]	CLC5_out(1)	000101 [5]	CLCIN5PPS(1)
100100 [36]	CLC4_out(1)	000100 [4]	CLCIN4PPS(1)
100011 [35]	CLC3_out(1)	000011 [3]	CLCIN3PPS(1)
100010 [34]	CLC2_out(1)	000010 [2]	CLCIN2PPS(1)
100001 [33]	CLC1_out(1)	000001 [1]	CLCIN1PPS(1)
100000 [32]	DSM1_out	000000 [0]	CLCIN0PPS(1)

注：
1. CN2510 器件上未提供。

复位状态： POR/BOR = xxxxxx
所有其他复位 = uuuuuu

28.7.7. CLCxSEL3

名称： CLCxSEL3
偏移量： 0xE2C,0xE36,0xE40,0xE4A,0xE54,0xE5E,0xE68,0xE72

通用 CLCx 数据 4 选择寄存器

位	7	6	5	4	3	2	1	0
			D4S[5:0]					
访问			R/W	R/W	R/W	R/W	R/W	R/W
复位			x	x	x	x	x	x

Bit 5:0 – D4S[5:0] CLCx 数据 4 输入选择位

DyS 值	CLC 输入源	DyS 值	CLC 输入源
111111 [63]	保留	011111 [31]	IOC 中断
111110 [62]	保留	011110 [30]	ZCD_out
111101 [61]	保留	011101 [29]	C2_out
111100 [60]	保留	011100 [28]	C1_out
111011 [59]	保留	011011 [27]	PWM4_out
111010 [58]	保留	011010 [26]	PWM3_out
111001 [57]	保留	011001 [25]	CCP2_out
111000 [56]	保留	011000 [24]	CCP1 _out
110111 [55]	保留	010111 [23]	TMR6_out
110110 [54]	保留	010110 [22]	TMR5 _overflow
110101 [53]	保留	010101 [21]	TMR4 _out
110100 [52]	保留	010100 [20]	TMR3 _overflow
110011 [51]	保留	010011 [19]	TMR2 _out
110010 [50]	CWG1B	010010 [18]	TMR1 _overflow
110001 [49]	CWG1A	010001 [17]	TMR0 _overflow
110000 [48]	SCK2	010000 [16]	CLKR _out
101111 [47]	SDO2	001111 [15]	ADCRC
101110 [46]	SCK1	001110 [14]	SOSC
101101 [45]	SDO1	001101 [13]	SFINTOSC（1 MHz）
101100 [44]	EUSART2_TX/CK_out(1)	001100 [12]	MFINTOSC（32 kHz）
101011 [43]	EUSART2_DT_out(1)	001011 [11]	MFINTOSC（500 kHz）
101010 [42]	EUSART1_TX/CK_out	001010 [10]	LFINTOSC
101001 [41]	EUSART1_DT_out	001001 [9]	HFINTOSC
101000 [40]	CLC8_out(1)	001000 [8]	F_{OSC}
100111 [39]	CLC7_out(1)	000111 [7]	CLCIN7PPS(1)
100110 [38]	CLC6_out(1)	000110 [6]	CLCIN6PPS(1)
100101 [37]	CLC5_out(1)	000101 [5]	CLCIN5PPS(1)
100100 [36]	CLC4_out(1)	000100 [4]	CLCIN4PPS(1)
100011 [35]	CLC3_out(1)	000011 [3]	CLCIN3PPS(1)
100010 [34]	CLC2_out(1)	000010 [2]	CLCIN2PPS(1)
100001 [33]	CLC1_out(1)	000001 [1]	CLCIN1PPS(1)
100000 [32]	DSM1_out	000000 [0]	CLCIN0PPS(1)

注：
1. CN2510 器件上未提供。

复位状态： POR/BOR = xxxxxx
所有其他复位 = uuuuuu

28.7.8. CLCxGLS0

名称： CLCxGLS0
偏移量： 0xE2D,0xE37,0xE41,0xE4B,0xE55,0xE5F,0xE69,0xE73

CLCx 门 1 逻辑选择寄存器

位	7	6	5	4	3	2	1	0
	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
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 1, 3, 5, 7 – G1DyT

dyT：门 1 数据“y”真值（不反相）位

复位状态： 默认值 = xxxx
POR/BOR = x
所有其他复位 = u

值	说明
1	dyT 被门控到 g1
0	dyT 不被门控到 g1

Bit 0, 2, 4, 6 – G1DyN

dyN：门 1 数据“y”取反（反相）位

复位状态： 默认值 = xxxx
POR/BOR = x
所有其他复位 = u

值	说明
1	dyN 被门控到 g1
0	dyN 不被门控到 g1

28.7.9. CLCxGLS1

名称： CLCxGLS1
偏移量： 0xE2E,0xE38,0xE42,0xE4C,0xE56,0xE60,0xE6A,0xE74

CLCx 门 2 逻辑选择寄存器

位	7	6	5	4	3	2	1	0
	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
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 1, 3, 5, 7 – G2DyT

dyT：门 2 数据“y”真值（不反相）位

复位状态： 默认值 = xxxx
POR/BOR = x
所有其他复位 = u

值	说明
1	dyT 被门控到 g2
0	dyT 不被门控到 g2

Bit 0, 2, 4, 6 – G2DyN

dyN：门 2 数据“y”取反（反相）位

复位状态： 默认值 = xxxx
POR/BOR = x
所有其他复位 = u

值	说明
1	dyN 被门控到 g2
0	dyN 不被门控到 g2

28.7.10. CLCxGLS2

名称： CLCxGLS2
偏移量： 0xE2F,0xE39,0xE43,0xE4D,0xE57,0xE61,0xE6B,0xE75

CLCx 门 3 逻辑选择寄存器

位	7	6	5	4	3	2	1	0
	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
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 1, 3, 5, 7 – G3DyT

dyT：门 3 数据“y”真值（不反相）位

复位状态： 默认值 = xxxx
POR/BOR = x
所有其他复位 = u

值	说明
1	dyT 被门控到 g3
0	dyT 不被门控到 g3

Bit 0, 2, 4, 6 – G3DyN

dyN：门 3 数据“y”取反（反相）位

复位状态： 默认值 = xxxx
POR/BOR = x
所有其他复位 = u

值	说明
1	dyN 被门控到 g3
0	dyN 不被门控到 g3

28.7.11. CLCxGLS3

名称： CLCxGLS3
偏移量： 0xE30,0xE3A,0xE44,0xE4E,0xE58,0xE62,0xE6C,0xE76

CLCx 门 4 逻辑选择寄存器

位	7	6	5	4	3	2	1	0
	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
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 1, 3, 5, 7 – G4DyT

dyT：门 4 数据“y”真值（不反相）位

复位状态： 默认值 = xxxx
POR/BOR = x
所有其他复位 = u

值	说明
1	dyT 被门控到 g4
0	dyT 不被门控到 g4

Bit 0, 2, 4, 6 – G4DyN

dyN：门 4 数据“y”取反（反相）位

复位状态： 默认值 = xxxx
POR/BOR = x
所有其他复位 = u

值	说明
1	dyN 被门控到 g4
0	dyN 不被门控到 g4

28.8. 寄存器汇总——CLC 控制

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 … 0x0E26	保留									
0x0E27	CLC1CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E28	CLC1POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E29	CLC1SEL0	7:0			D1S[5:0]					
0x0E2A	CLC1SEL1	7:0			D2S[5:0]					
0x0E2B	CLC1SEL2	7:0			D3S[5:0]					
0x0E2C	CLC1SEL3	7:0			D4S[5:0]					
0x0E2D	CLC1GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E2E	CLC1GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E2F	CLC1GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E30	CLC1GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E31	CLC2CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E32	CLC2POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E33	CLC2SEL0	7:0			D1S[5:0]					
0x0E34	CLC2SEL1	7:0			D2S[5:0]					
0x0E35	CLC2SEL2	7:0			D3S[5:0]					
0x0E36	CLC2SEL3	7:0			D4S[5:0]					
0x0E37	CLC2GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E38	CLC2GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E39	CLC2GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E3A	CLC2GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E3B	CLC3CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E3C	CLC3POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E3D	CLC3SEL0	7:0			D1S[5:0]					
0x0E3E	CLC3SEL1	7:0			D2S[5:0]					
0x0E3F	CLC3SEL2	7:0			D3S[5:0]					
0x0E40	CLC3SEL3	7:0			D4S[5:0]					
0x0E41	CLC3GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E42	CLC3GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E43	CLC3GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E44	CLC3GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E45	CLC4CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E46	CLC4POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E47	CLC4SEL0	7:0			D1S[5:0]					
0x0E48	CLC4SEL1	7:0			D2S[5:0]					
0x0E49	CLC4SEL2	7:0			D3S[5:0]					
0x0E4A	CLC4SEL3	7:0			D4S[5:0]					
0x0E4B	CLC4GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E4C	CLC4GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E4D	CLC4GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E4E	CLC4GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E4F	CLC5CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E50	CLC5POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E51	CLC5SEL0	7:0			D1S[5:0]					
0x0E52	CLC5SEL1	7:0			D2S[5:0]					
0x0E53	CLC5SEL2	7:0			D3S[5:0]					
0x0E54	CLC5SEL3	7:0			D4S[5:0]					
0x0E55	CLC5GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E56	CLC5GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E57	CLC5GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E58	CLC5GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E59	CLC6CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E5A	CLC6POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E5B	CLC6SEL0	7:0			D1S[5:0]					
0x0E5C	CLC6SEL1	7:0			D2S[5:0]					
0x0E5D	CLC6SEL2	7:0			D3S[5:0]					
0x0E5E	CLC6SEL3	7:0			D4S[5:0]					

寄存器汇总——CLC 控制（续）

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x0E5F	CLC6GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E60	CLC6GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E61	CLC6GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E62	CLC6GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E63	CLC7CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E64	CLC7POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E65	CLC7SEL0	7:0			D1S[5:0]					
0x0E66	CLC7SEL1	7:0			D2S[5:0]					
0x0E67	CLC7SEL2	7:0			D3S[5:0]					
0x0E68	CLC7SEL3	7:0			D4S[5:0]					
0x0E69	CLC7GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E6A	CLC7GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E6B	CLC7GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E6C	CLC7GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E6D	CLC8CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E6E	CLC8POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E6F	CLC8SEL0	7:0			D1S[5:0]					
0x0E70	CLC8SEL1	7:0			D2S[5:0]					
0x0E71	CLC8SEL2	7:0			D3S[5:0]					
0x0E72	CLC8SEL3	7:0			D4S[5:0]					
0x0E73	CLC8GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E74	CLC8GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E75	CLC8GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E76	CLC8GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E77	CLCDATA	7:0	MLC8OUT	MLC7OUT	MLC6OUT	MLC5OUT	MLC4OUT	MLC3OUT	MLC2OUT	MLC1OUT

29. DSM——数据信号调制器模块

数据信号调制器（DSM）是一种外设，用户可以通过它将数据流（也称为调制器信号）与载波信号进行混合来产生调制输出。

载波和调制器信号均送到 DSM 模块，信号可以来自内部、某个外设的输出，也可以通过某个输入引脚从外部提供。

调制输出信号通过对载波信号和调制器信号执行逻辑与运算生成，然后提供给 MDOUT 引脚。

载波信号由两个不同的独立信号组成。载波高电平（CARH）信号和载波低电平（CARL）信号。在调制器（MOD）信号处于逻辑高电平状态期间，DSM 会将 CARH 信号与调制器信号进行混合。在调制器信号处于逻辑低电平状态时，DSM 会将 CARL 信号与调制器信号进行混合。

通过这种方法，DSM 可以产生以下几种键控调制方案：

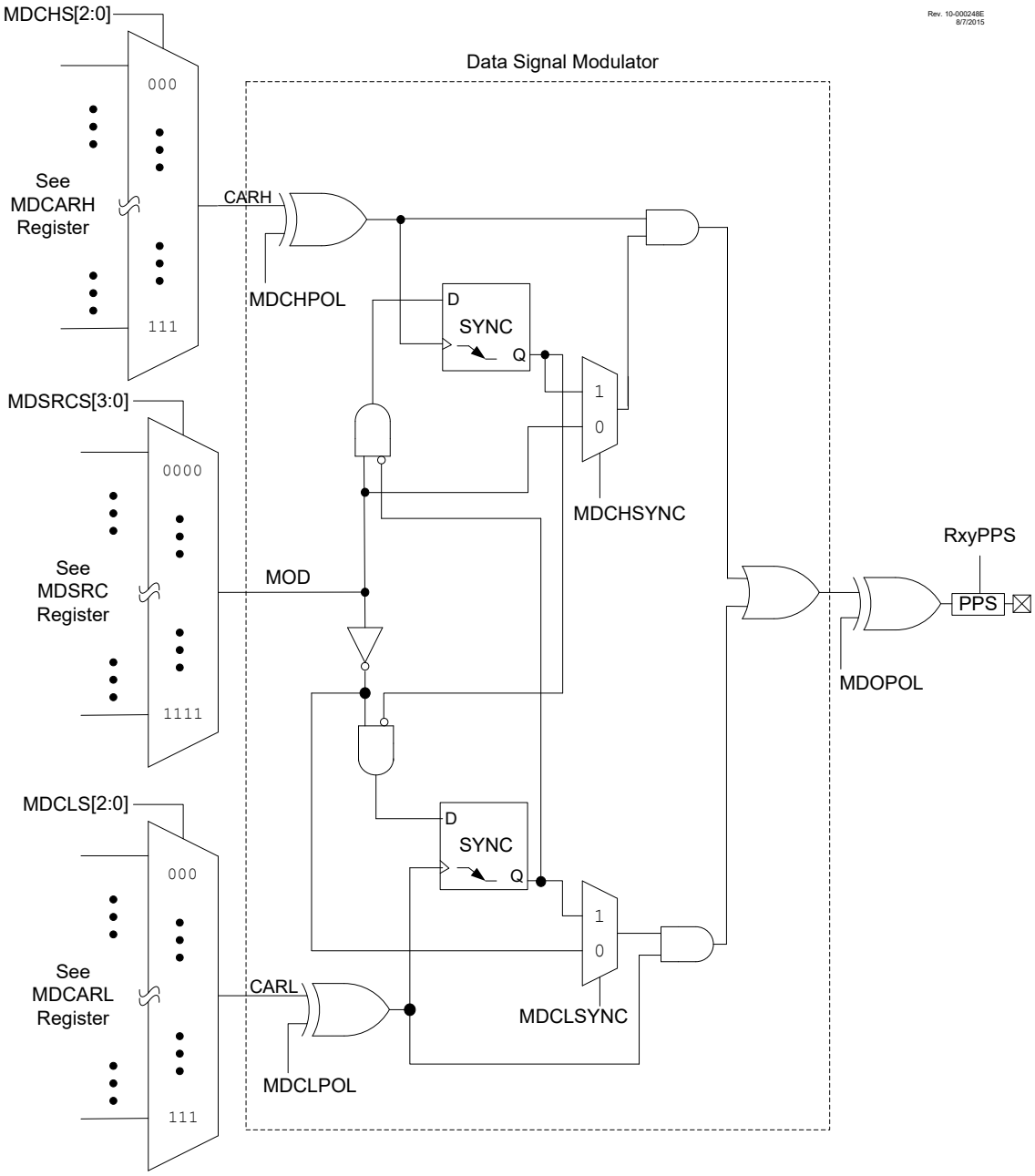
- 频移键控（Frequency-Shift Keying，FSK）
- 相移键控（Phase-Shift Keying，PSK）
- 开关键控（On-Off Keying，OOK）

此外，DSM 模块还提供了以下特性：

- 载波同步
- 载波源极性选择
- 可编程调制器数据
- 调制输出极性选择
- 外设模块禁止，可使 DSM 模块进入最低功耗模式

下图显示了数据信号调制器外设的简化框图。

图 29-1. 数据信号调制器的简化框图



29.1. DSM 工作原理

DSM 模块可通过将 MDCON0 寄存器中的 EN 位置 1 来使能。清零 EN 位将禁止模块的输出，但会保留载波和源信号选择。当 EN 位再次置 1 时，模块将恢复工作。使用 RxyPPS 寄存器可将 DSM 模块的输出重新连接到多个引脚。当 EN 位清零时，输出引脚保持低电平。

29.2. 调制器信号源

调制器信号可以通过以下信号源提供：

- EUSART
- MSSP
- CLC
- CCP/PWM
- 内部 MDBIT
- I/O 引脚

这些信号源使用 MDxSRC 寄存器的 SRCS 位进行选择。

29.3. 载波信号源

载波高电平信号和载波低电平信号可以通过以下信号源提供：

- CCP
- CLC
- PWM
- 参考时钟
- 系统时钟
- I/O 引脚

载波高电平信号通过 MDxCARH 寄存器的 CHS 位进行选择。

载波低电平信号通过 MDxCARL 寄存器的 CLS 位进行选择。

29.4. 载波同步

当 DSM 在高载波信号源和低载波信号源之间切换时，调制输出信号中的载波数据可能会被截短。为了防止这种情况，可以将载波信号与调制器信号进行同步。当使能同步时，DSM 将允许在切换时进行混合的载波脉冲先变为低电平，然后再切换为另一个载波源。

对于高载波信号源和低载波信号源，同步功能单独进行使能。高载波信号的同步通过将 CHSYNC 位置 1 来使能。低载波信号的同步通过将 CLSYNC 位置 1 来使能。

下图给出了使用各种同步方法的时序图。

图 29-2. 开关键控（OOK）同步

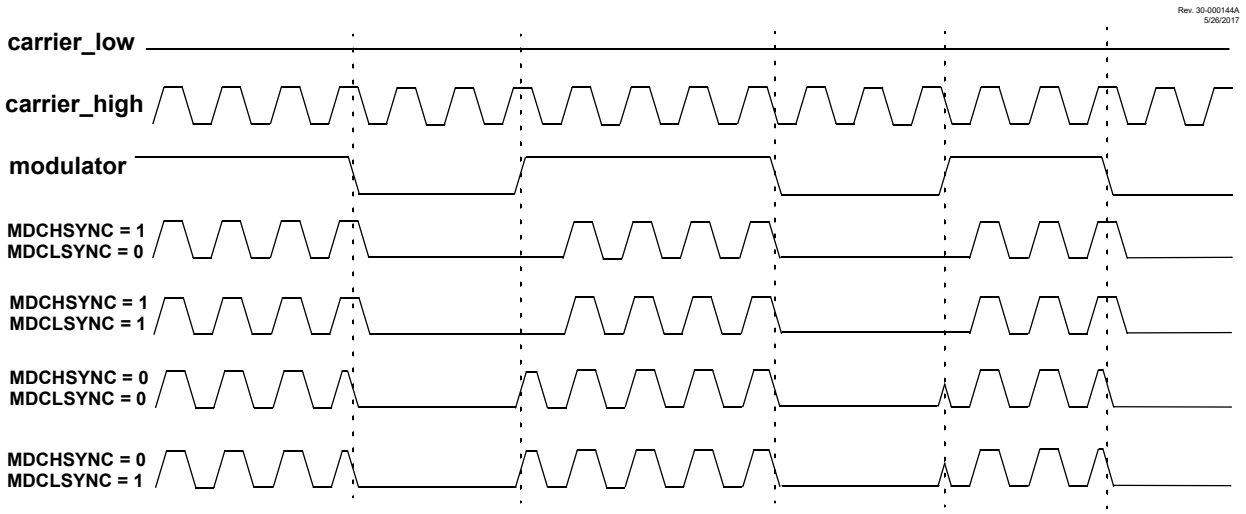


图 29-3. 无同步（MDCHSYNC = 0，MDCLSYNC = 0）

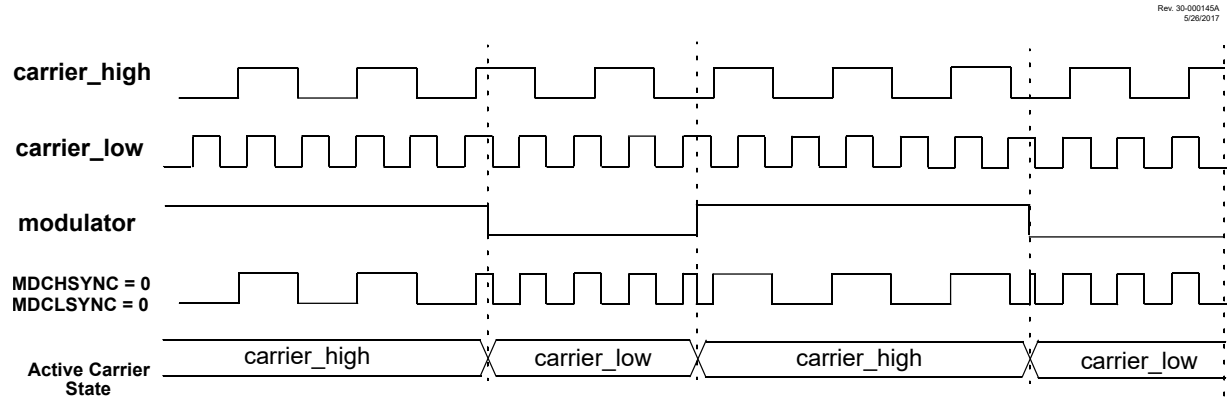


图 29-4. 高载波信号同步（MDCHSYNC = 1，MDCLSYNC = 0）

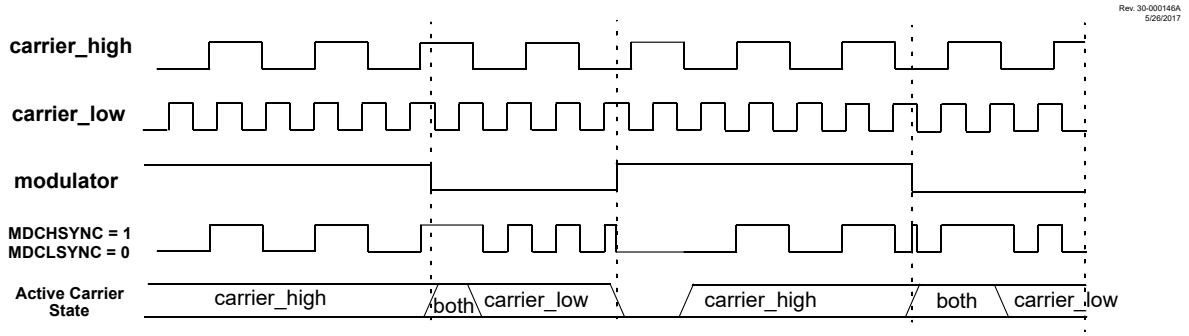


图 29-5. 低载波信号同步（MDCHSYNC = 0，MDCLSYNC = 1）

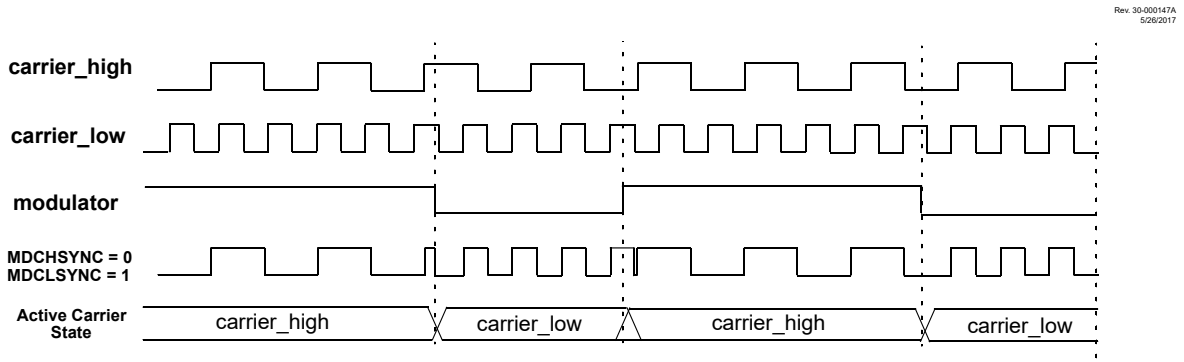
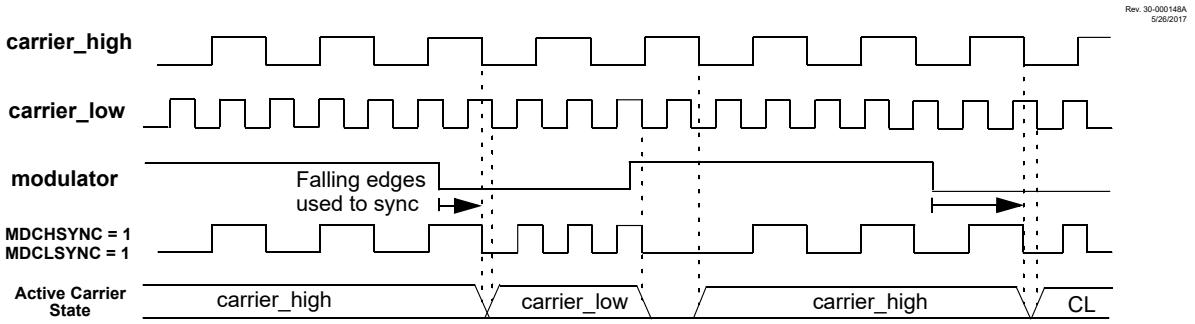


图 29-6. 完全同步（MDCHSYNC = 1，MDCLSYNC = 1）



29.5. 载波源极性选择

高载波信号和低载波信号的任意选定输入源提供的信号都可以进行反相。对高载波信号源信号和低载波信号源信号反相分别通过将 CHPOL 位和 CLPOL 位置 1 来使能。

29.6. 可编程调制器数据

可选择 BIT 控制位作为调制源。这样，用户便可提供由软件驱动的调制。

29.7. 调制输出极性

送到 DSM 引脚上的调制输出信号也可以进行反相。调制输出信号的反相通过将 OPOL 位置 1 来使能。

29.8. 在休眠模式下工作

如果载波源和调制器输入源仍在休眠期间工作，则 DSM 在休眠期间仍然可以工作。有关详细信息，请参见**“节能工作模式”**一章。

29.9. 复位的影响

在发生任何器件复位时，都将禁止 DSM 模块。用户的固件负责在使能输出之前初始化该模块。所有寄存器均会复位为其默认值。

29.10. 外设模块禁止

使用 PMD 模块可以完全禁止 DSM 模块，从而最大限度地降低功耗。当 PMD7 寄存器的 DSM1MD 位置 1 时，DSM 模块将完全禁止。这可使模块进入最低功耗状态。重新使能时，DSM 模块的所有寄存器都默认为 POR 状态。

29.11. 寄存器定义：调制控制

29.11.1. MDCON0

名称： MDCON0
偏移量： 0xF4C

调制控制寄存器 0

位	7	6	5	4	3	2	1	0
	EN		OUT	OPOL				BIT
访问	R/W		R/W	R/W				R/W
复位	0		0	0				0

Bit 7 – EN 调制器模块使能位

值	说明
1	使能调制器模块，且该模块混合输入信号
0	禁止调制器模块，且该模块没有输出

Bit 5 – OUT 调制器输出位
指示调制器模块的当前输出值。

Bit 4 – OPOL 调制器输出极性选择位

值	说明
1	调制器输出信号反相；空闲时输出高电平
0	调制器输出信号不反相；空闲时输出低电平

Bit 0 – BIT 调制源选择输入位
允许使用软件手动设置模块的调制源输入

注：

1. 调制输出频率可能会高于更新该寄存器位的时钟，与该时钟异步；位值对于速度较高的调制器信号或载波信号可能无效。
2. 对于该操作，必须在 MDSRC 寄存器中选择 MDBIT 作为调制源。

29.11.2. MDCON1

名称： MDCON1
偏移量： 0xF4D

调制控制寄存器 1

位	7	6	5	4	3	2	1	0
			CHPOL	CHSYNC			CLPOL	CLSYNC
访问			R/W	R/W			R/W	R/W
复位			0	0			0	0

Bit 5 – CHPOL 调制器高载波信号极性选择位

值	说明
1	选定的高载波信号反相
0	选定的高载波信号不反相

Bit 4 – CHSYNC 调制器高载波信号同步使能位

值	说明
1	调制器先等待高载波信号上出现下降沿，然后再切换为低载波信号
0	调制器输出不与高载波信号进行同步

Bit 1 – CLPOL 调制器低载波信号极性选择位

值	说明
1	选定的低载波信号反相
0	选定的低载波信号不反相

Bit 0 – CLSYNC 调制器低载波信号同步使能位

值	说明
1	调制器先等待低载波信号上出现下降沿，然后再切换为高载波信号
0	调制器输出不与低载波信号进行同步

注：

1. 如果载波未进行同步，则信号流中的载波脉宽可能会变窄，或者可能出现杂散信号。

29.11.3. MDSRC

名称： MDSRC
偏移量： 0xF4E

调制源控制寄存器

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

Bit 4:0 – SRCS[4:0] 调制器源选择位

表 29-1. MDSRC 选择 MUX 连接

SRCS[3:0]	连接
11111-11000	**保留**
10111	CLC8_out(1)
10110	CLC7_out(1)
10101	CLC6_out(1)
10100	CLC5_out(1)
10011	CLC4_out(1)
10010	CLC3_out(1)
10001	CLC2_out(1)
10000	CLC1_out(1)
01111	**保留**
01110	**保留**
01101	EUSART2 TX（TX/CK 输出）(1)
01100	EUSART2 RX（DT 输出）(1)
01011	SDO2(1)
01010	SDO1
01001	EUSART1 TX（TX/CK 输出）
01000	EUSART1 RX（DT 输出）
00111	C2 OUT
00110	C1 OUT
00101	PWM4 OUT
00100	PWM3 OUT
00011	CCP2 OUT
00010	CCP1 OUT
00001	MDBIT
00000	通过 MDSRCPPS 选择的引脚

注：
1. CN2510 器件上未提供。

29.11.4. MDCARL

名称： MDCARL
偏移量： 0xF4F

调制低载波信号控制寄存器

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

Bit 3:0 – CLS[3:0] 调制器低载波信号输入选择位

表 29-2. MDCARL 源选择

MDCARL	
CLS[2:0]	**连接**
1111	CLC8_out(1)
1110	CLC7_out(1)
1101	CLC6_out(1)
1100	CLC5_out(1)
1011	CLC4_out(1)
1010	CLC3_out(1)
1001	CLC2_out(1)
1000	CLC1_out(1)
0111	PWM4 OUT
0110	PWM3 OUT
0101	CCP2 OUT
0100	CCP1 OUT
0011	CLKREF 输出
0010	HFINTOSC
0001	F_{OSC}（系统时钟）
0000	通过 MDCARLPPS 选择的引脚

注：
1. CN2510 器件上未提供。

29.11.5. MDCARH

名称： MDCARH
偏移量： 0xF50

调制高载波信号控制寄存器

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

Bit 3:0 – CHS[3:0] 调制器高载波信号选择位

表 29-3. MDCARH 源选择

MDCARH	
CHS[2:0]	**连接**
1111	CLC8_out(1)
1110	CLC7_out(1)
1101	CLC6_out(1)
1100	CLC5_out(1)
1011	CLC4_out(1)
1010	CLC3_out(1)
1001	CLC2_out(1)
1000	CLC1_out(1)
0111	PWM4 OUT
0110	PWM3 OUT
0101	CCP2 OUT
0100	CCP1 OUT
0011	CLKREF 输出
0010	HFINTOSC
0001	F_{OSC}（系统时钟）
0000	通过 MDCARHPPS 选择的引脚

注：
1. CN2510 器件上未提供。

29.12. 寄存器汇总——DSM

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0F4B	保留									
0x0F4C	MDCON0	7:0	EN		OUT	OPOL				BIT
0x0F4D	MDCON1	7:0			CHPOL	CHSYNC			CLPOL	CLSYNC
0x0F4E	MDSRC	7:0				SRCS[4:0]				
0x0F4F	MDCARL	7:0					CLS[3:0]			
0x0F50	MDCARH	7:0					CHS[3:0]			

30. MSSP——主同步串行端口模块

主同步串行端口（MSSP）模块是用于同其他外设或单片机进行通信的串行接口。这些外设器件可以是串行 EEPROM、移位寄存器、显示驱动器和 A/D 转换器等。

表 30-1. MSSP 模块可用性

器件	MSSP1	MSSP2
CN2510	有	无
CN2610	有	有
CN2710	有	有

MSSP 模块支持以下两种工作模式：

- 串行外设接口（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

30.1. SPI 模式概述

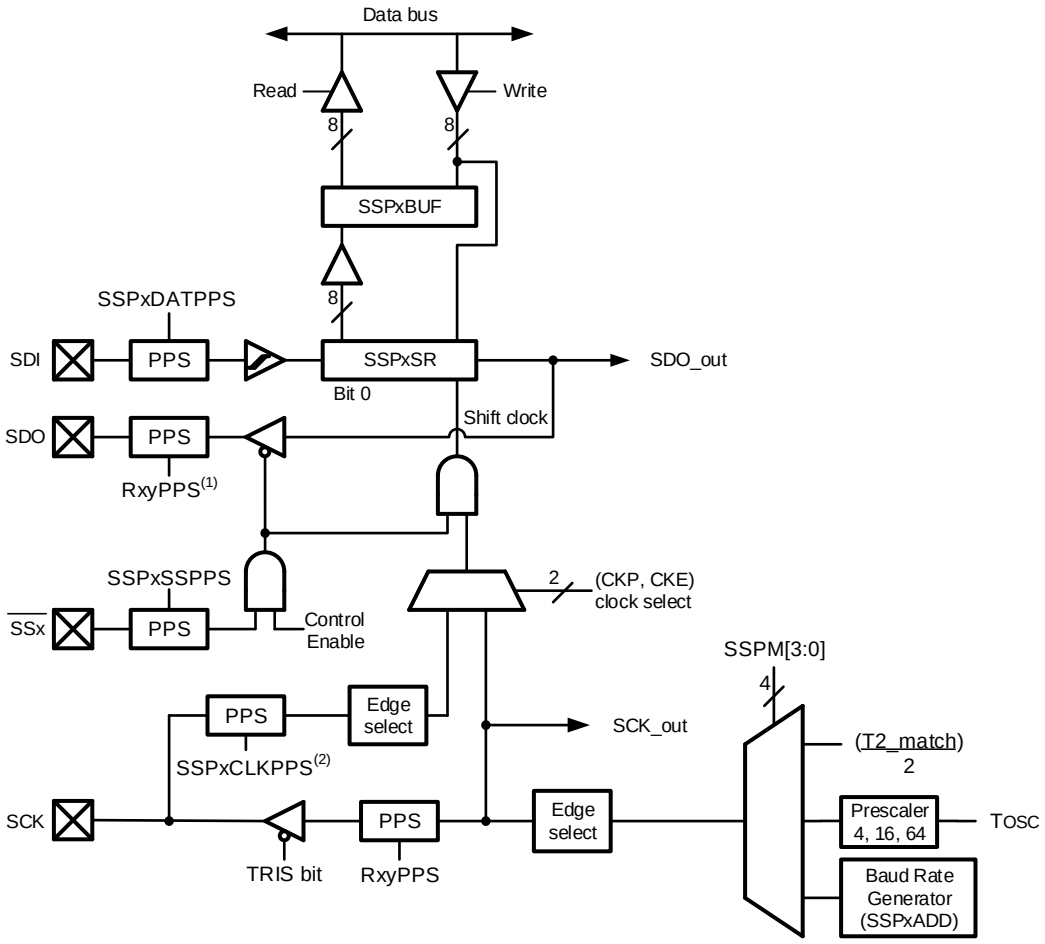
串行外设接口（SPI）是以全双工模式工作的同步串行数据通信总线。器件在由主器件启动通信的主/从环境中进行通信。进行通信的从器件通过从选择功能来选择。

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 30-1 给出了 MSSP 模块在 SPI 模式下工作时的框图。

图 30-1. MSSP 框图（SPI 模式）

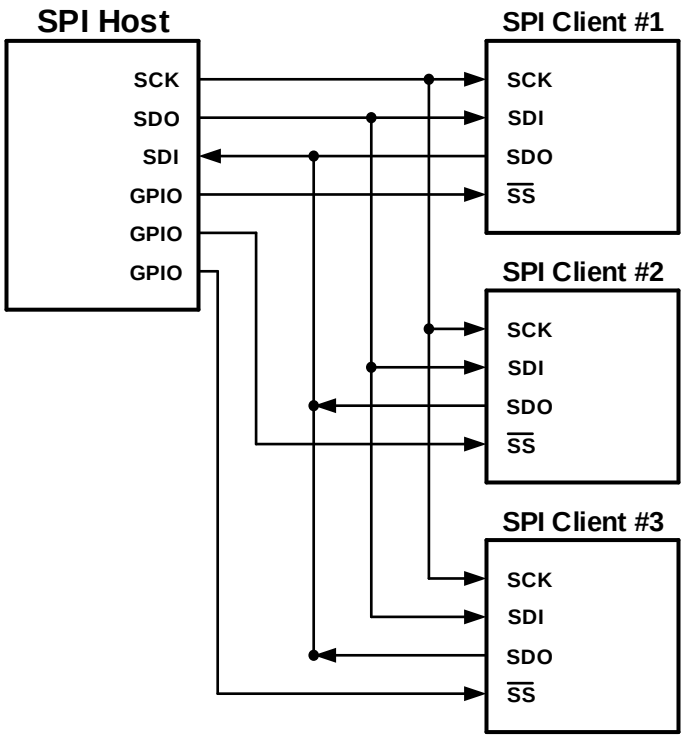


Notes: 1. Output selection for Host mode.
2. Input selection for Client and Host modes.

SPI 总线工作需要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 30-2 给出了主器件和多个从器件之间的典型连接。

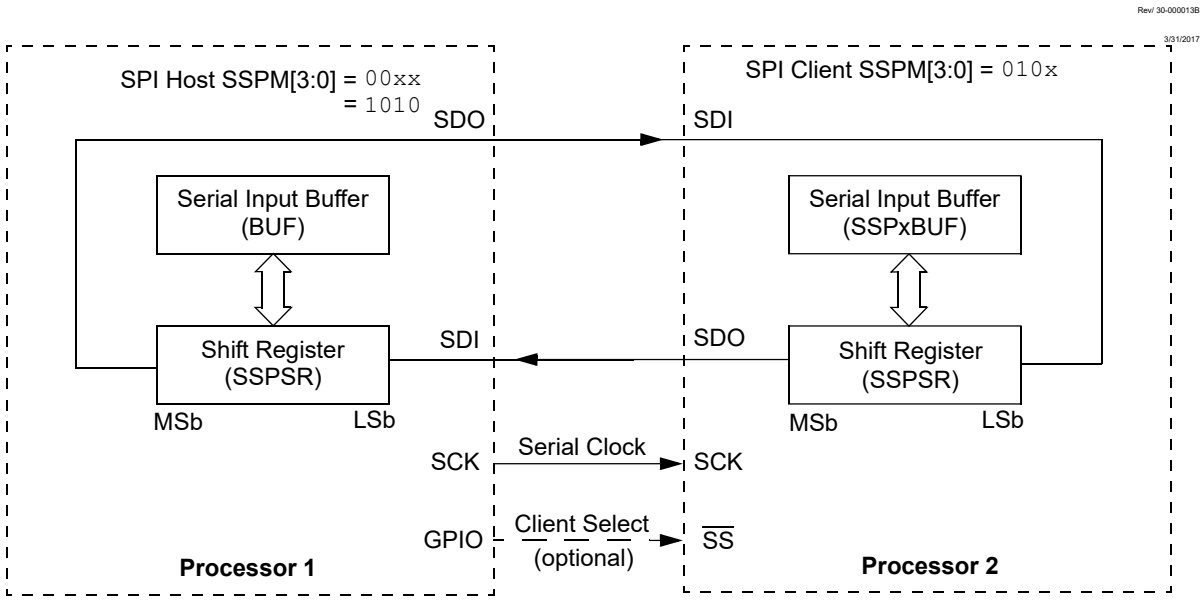
图 30-2. SPI 主器件与多个从器件之间的连接



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 30-3 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 30-3. 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。

主器件通过其 SDO 输出引脚发送信息，该引脚连接到从器件的 SDI 输入引脚并被其接收。从器件通过其 SDO 输出引脚发送信息，该引脚连接到主器件的 SDI 输入引脚并被其接收。

要开始进行通信，主器件从其移位寄存器中发送 MSb 以及时钟信号。主器件和从器件需要配置为相同的时钟极性。在每个 SPI 时钟周期中，会发生全双工数据传输。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（虚拟数据），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，从器件发送虚拟数据
- 主器件发送有用的数据，从器件发送有用的数据
- 主器件发送虚拟数据，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被选定，则会丢弃时钟和发送信号，且不会发送任何自己的数据。

30.1.1. 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。

30.1.2. 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。

30.1.2.1. 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 30-3 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 30-4 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。

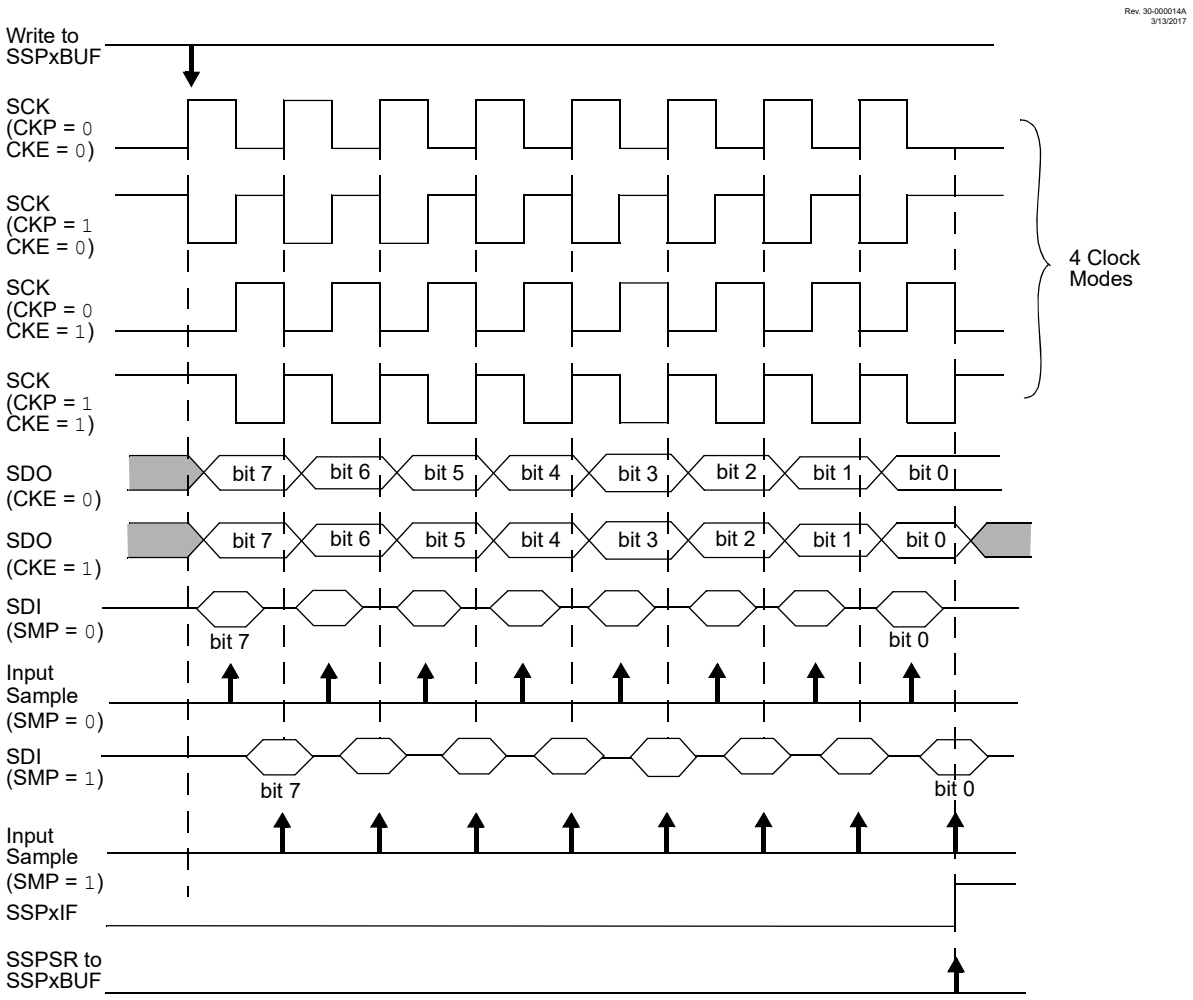
SPI 数据输入采样（SMP）位决定何时采样 SDI 输入。当 SMP 置 1 时，在数据输出时间的末端采样输入数据。当 SMP 清零时，在数据输出时间的中间采样输入数据。

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 30-4. SPI 模式波形图（主模式）



30.1.2.2. 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。

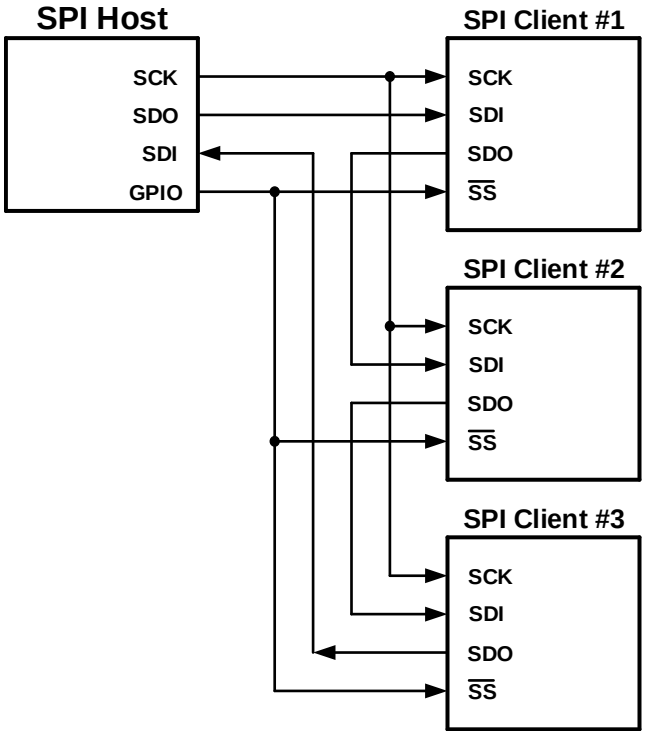
30.1.2.3. 菊花链配置

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 30-5 给出了在 SPI 模式下工作时典型菊花链连接的框图。

图 30-5. SPI 菊花链连接



30.1.2.4. 从选择同步

从选择也可以用于对通信进行同步（见图 30-6）。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 30-8）。如果 CKE 清零，$\overline{SS}$ 引脚控制可选（见图 30-7）。
3. 工作于 SPI 从模式下时，SMP 位必须保持清零。

图 30-6. 从选择同步波形图

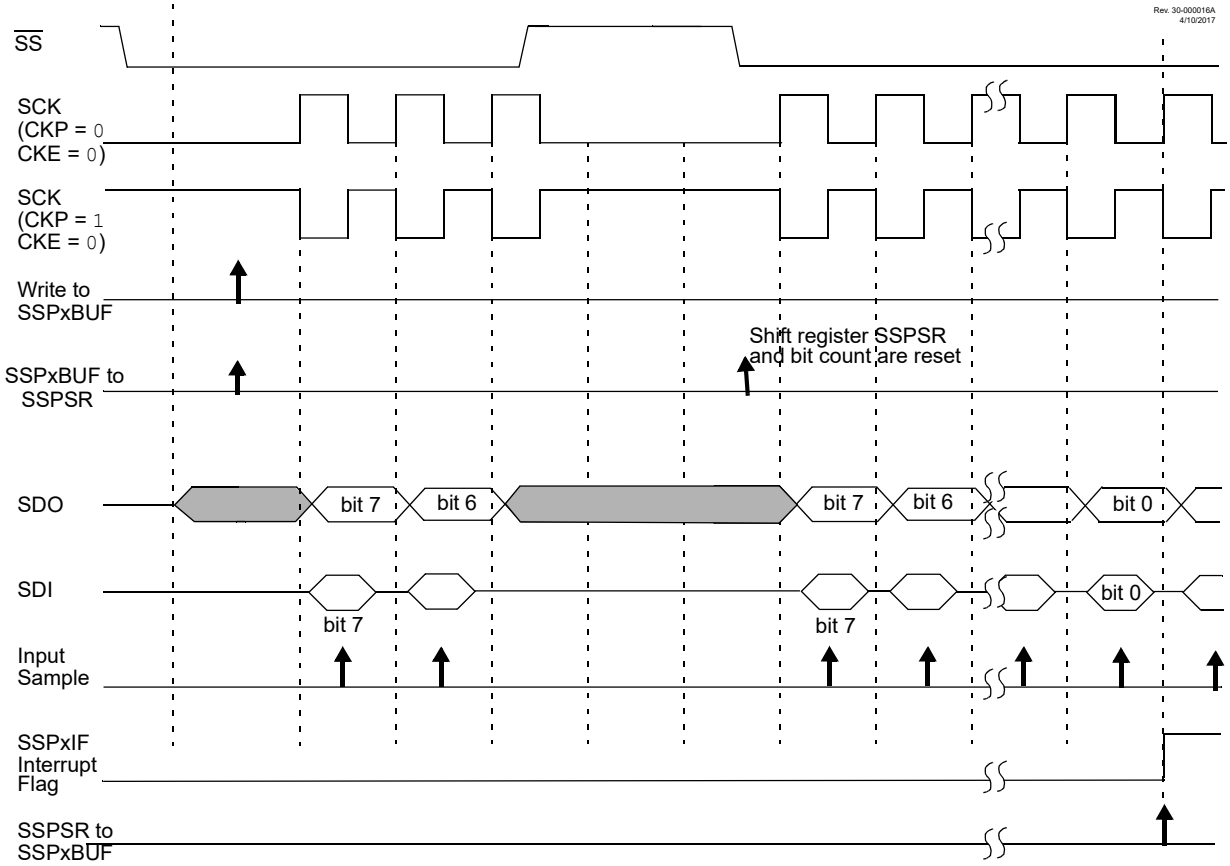


图 30-7. SPI 模式波形图（从模式，CKE = 0）

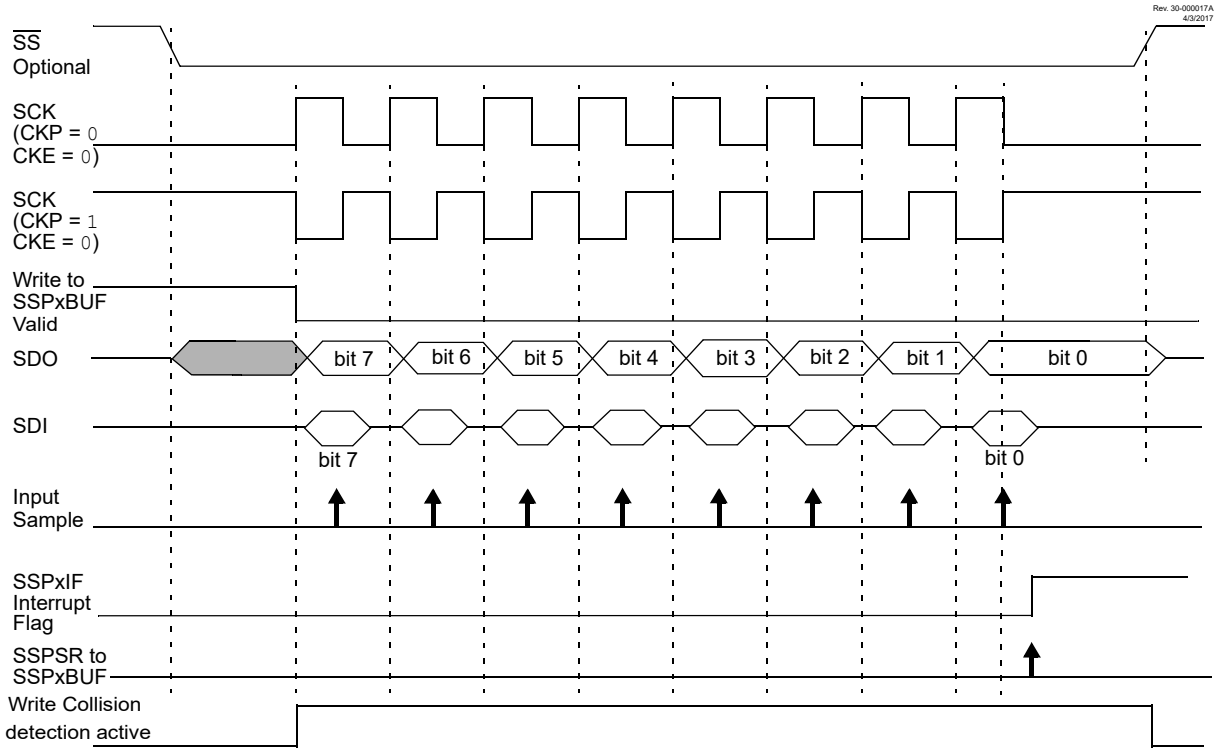
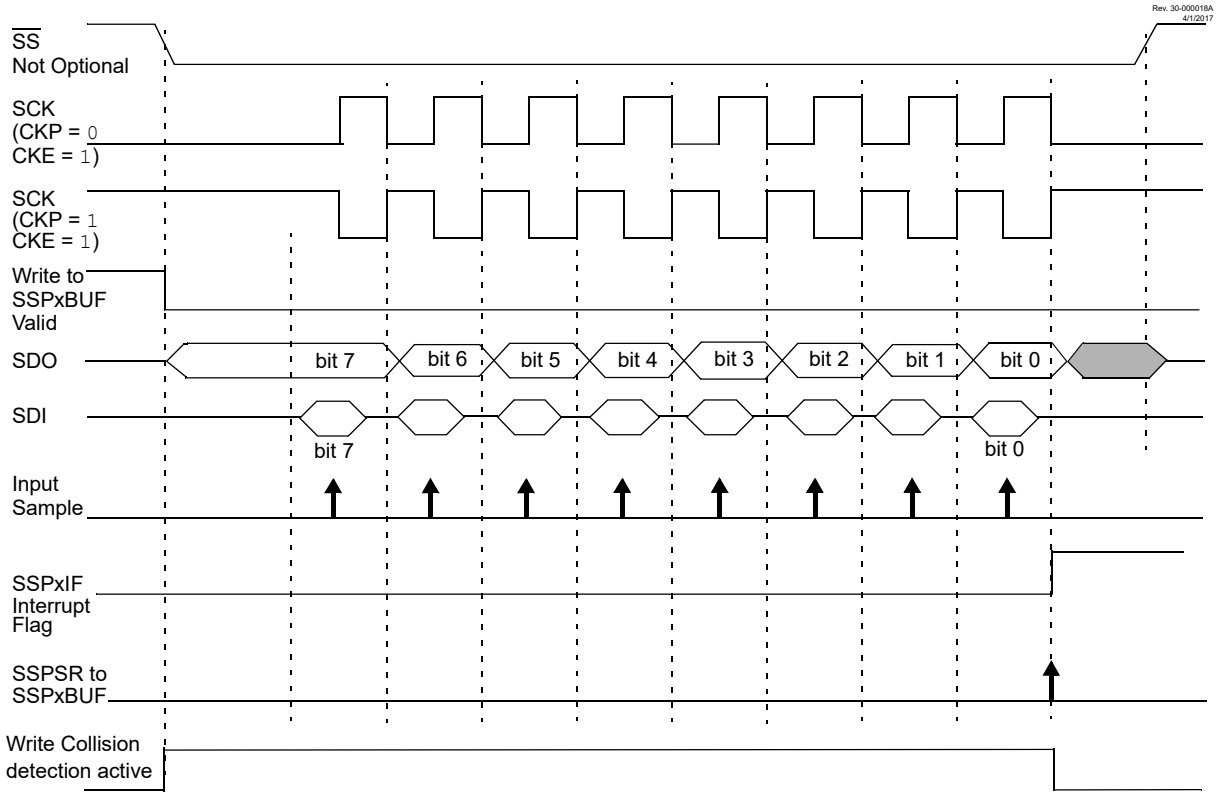


图 30-8. SPI 模式波形图（从模式，CKE = 1）



30.1.2.5. SPI 在休眠模式下的工作

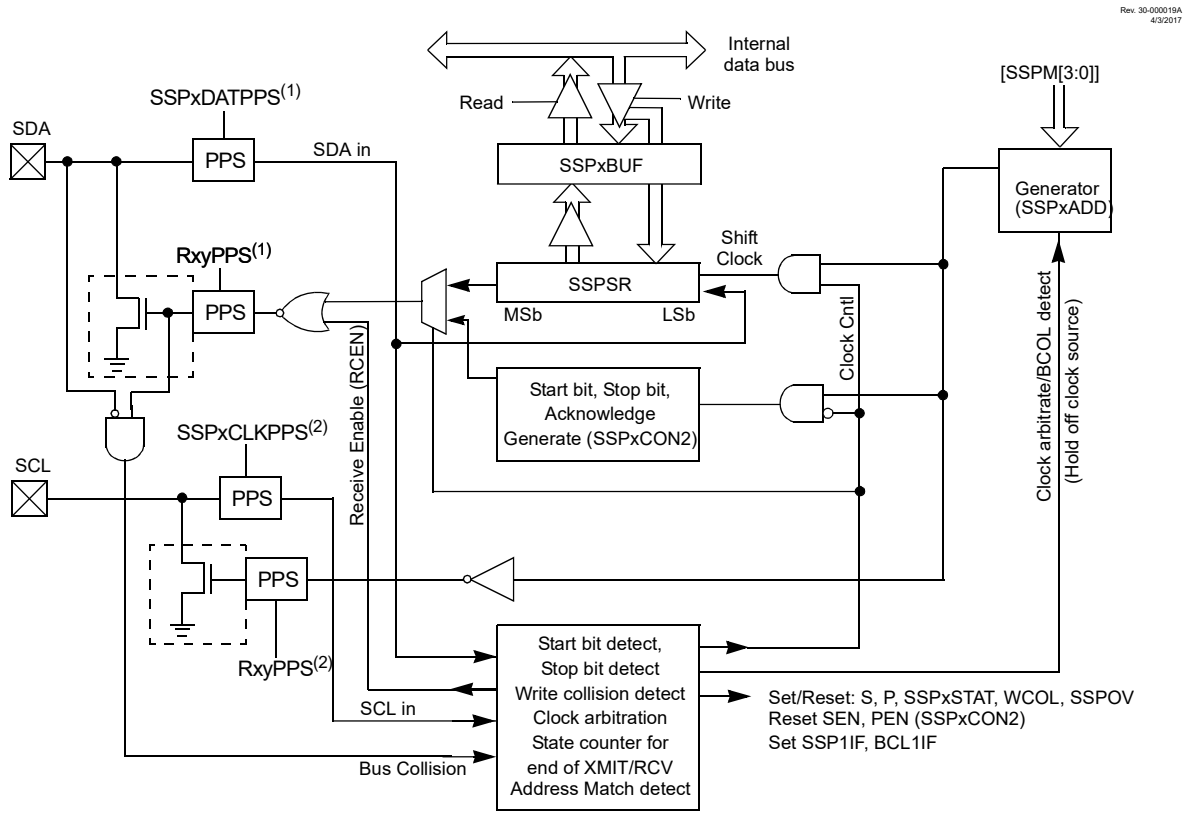
在 SPI 主模式下，当选择休眠模式时，所有模块的时钟都将暂停，并且在器件被唤醒前，发送/接收将保持此暂停状态。器件返回到运行模式之后，模块将恢复发送和接收数据。

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。

30.2. 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 30-9 和图 30-10 分别显示了 I²C 主模式和从模式的框图。

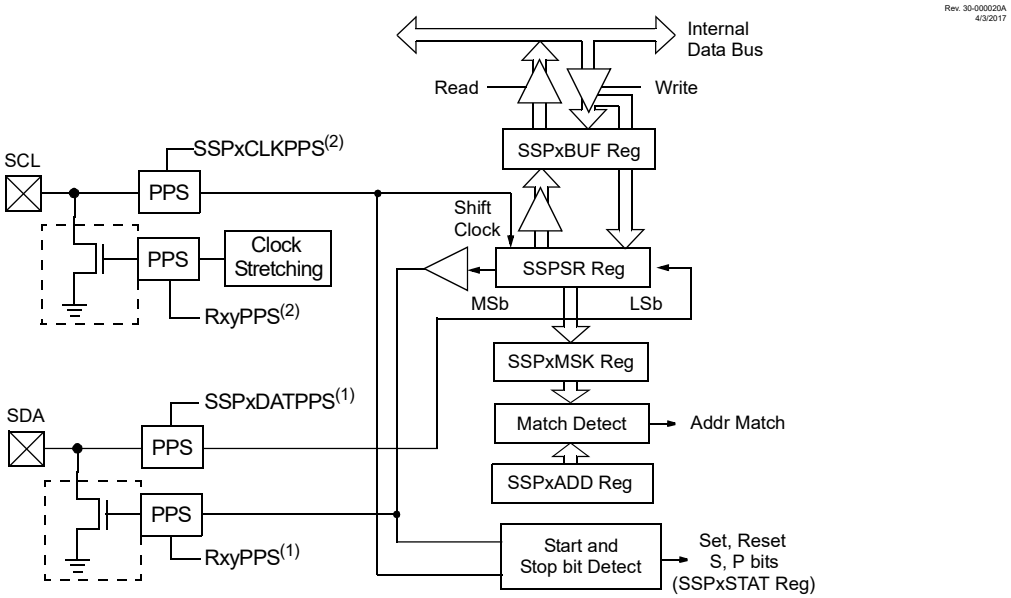
图 30-9. MSSP 框图（I²C 主模式）



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 30-10. 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.

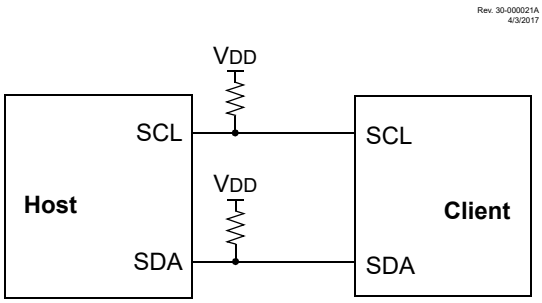
I²C 总线指定两种信号连接：

- 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路，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 30-11 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 30-11. I²C 主/从器件连接



I²C 总线工作需要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，后跟它希望与之通信的从器件的地址字节。启动条件由 SCL 线保持为高电平时 SDA 线的由高至低跳变来指示。地址和数据字节随后发送，先发送 MSb。后面再跟随单个读/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。当主器件希望从从器件读取数据时，R/$\overline{W}$ 位作为逻辑 1 发送，当主器件希望写入数据到从器件时，该位作为逻辑 0 发送。

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，则它会重复发送一个字节的数据，而从器件则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，则它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。在该示例中，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传输。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。停止条件通过 SDA 线由低电平至高电平跳变，同时 SCL 线保持高电平来指示。

在某些情况下，主器件可能希望维持对总线的控制，并重新启动另一次传输。如果是这样，主器件可以在它处于接收模式时，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

30.2.1. 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。

30.2.2. 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 MSb。8 个 SFR 寄存器和 2 个中断标志用作模块与用户软件的接口。该模块还采用了 2 个引脚（SDA 和 SCL）与其他外部 I²C 器件通信。

30.2.2.1. 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 30-2. 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
主器件	启动数据传输、产生时钟信号和终止数据传输的器件
从器件	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

30.2.2.2. 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。

30.2.2.3. 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。

30.2.2.4. 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。

30.2.2.5. 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。

30.2.2.6. 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。

30.2.2.7. 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 30-12 给出了启动条件和停止条件的波形图。

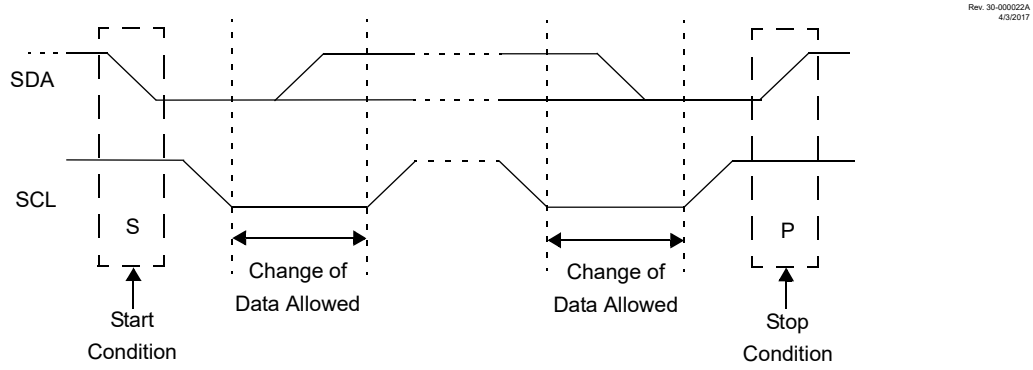
如果模块在将 SDA 线置为低电平之前采样到 SDA 线为低电平，则会在产生启动条件时发生总线冲突。这一点不符合 I^2C 规范，该规范规定启动时不能发生总线冲突。

30.2.2.8. 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 30-12. I^2C 启动和停止条件



30.2.2.9. 启动/停止条件中断屏蔽

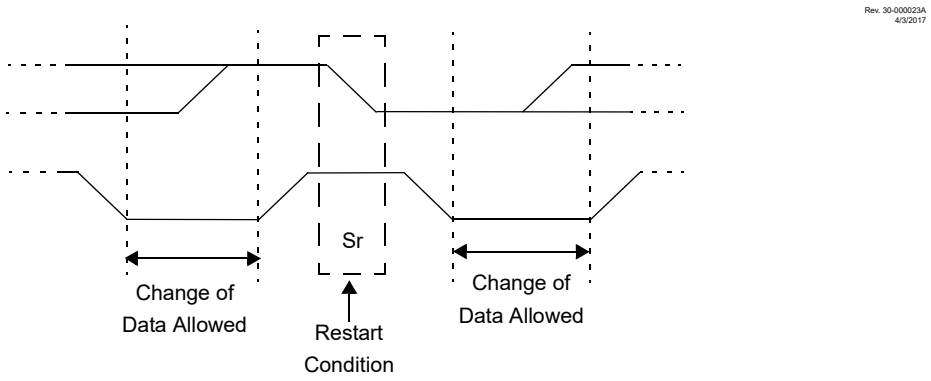
启动条件中断允许（SCIE）位和停止条件中断允许（PCIE）位可以用于允许在通常不支持中断功能的从模式下产生中断。对于已允许启动和停止检测中断的从模式，这两个位没有任何作用。

30.2.2.10. 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 30-13 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 30-13. I^2C 重复启动条件



30.2.2.11. 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。

30.2.3. 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。

30.2.3.1. 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。

30.2.3.1.1. 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。

30.2.3.1.2. 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。

30.2.3.2. 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。

30.2.3.2.1. 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。

30.2.3.2.2. 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。

30.2.3.2.3. 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。

30.2.3.3. 时钟同步和 CKP 位

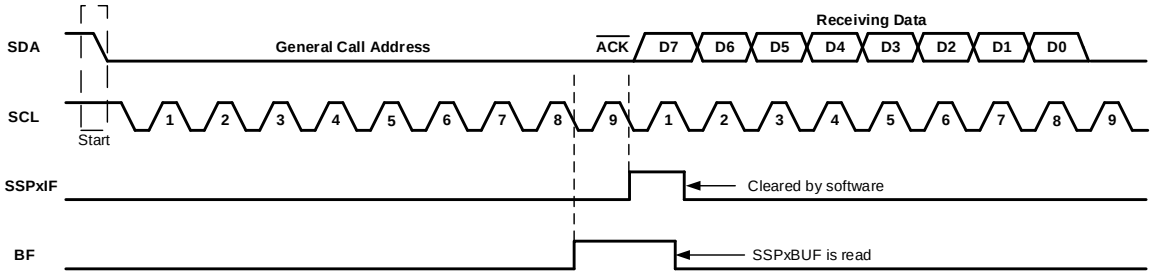
每当 CKP 位清零时，模块就会等待 SCL 线变为低电平，然后保持低电平状态不变。但是，只有当已经采样到 SCL 输出为低电平时，清零 CKP 位才会将 SCL 输出置为低电平。因此，CKP 位不会将 SCL 线拉为低电平，除非外部 I²C 主器件已将 SCL 线拉为低电平。SCL 输出将保持低电平，直到 CKP 位置 1 且 I²C 总线上的所有其他器件已释放 SCL 为止。

30.2.3.4. 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 30-14 显示了广播呼叫接收序列。

图 30-14. 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。

30.2.3.5. 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。

30.2.3.6. 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。

30.2.3.6.1. 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 30-15 和图 30-16 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 30-15. I²C 从模式接收时序（SEN = 0，AHEN = 0，DHEN = 0，7 位地址）

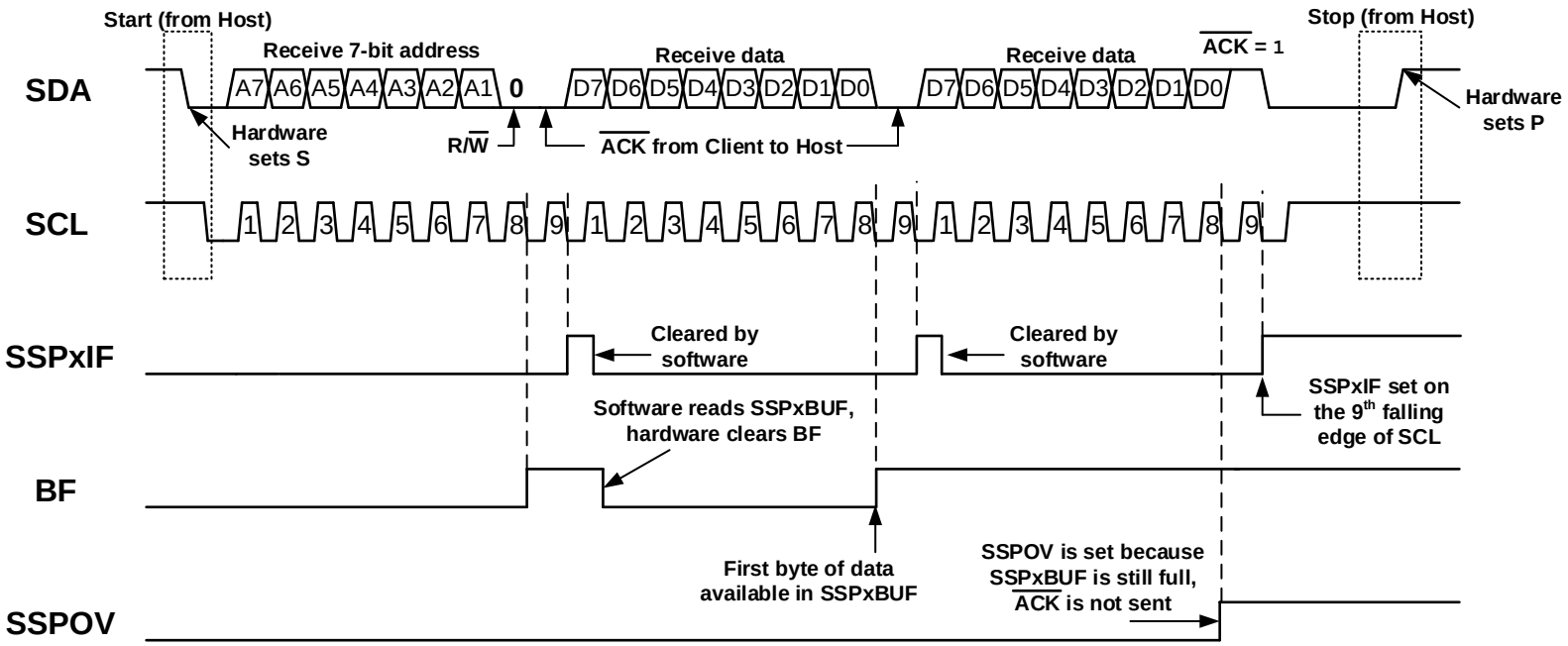
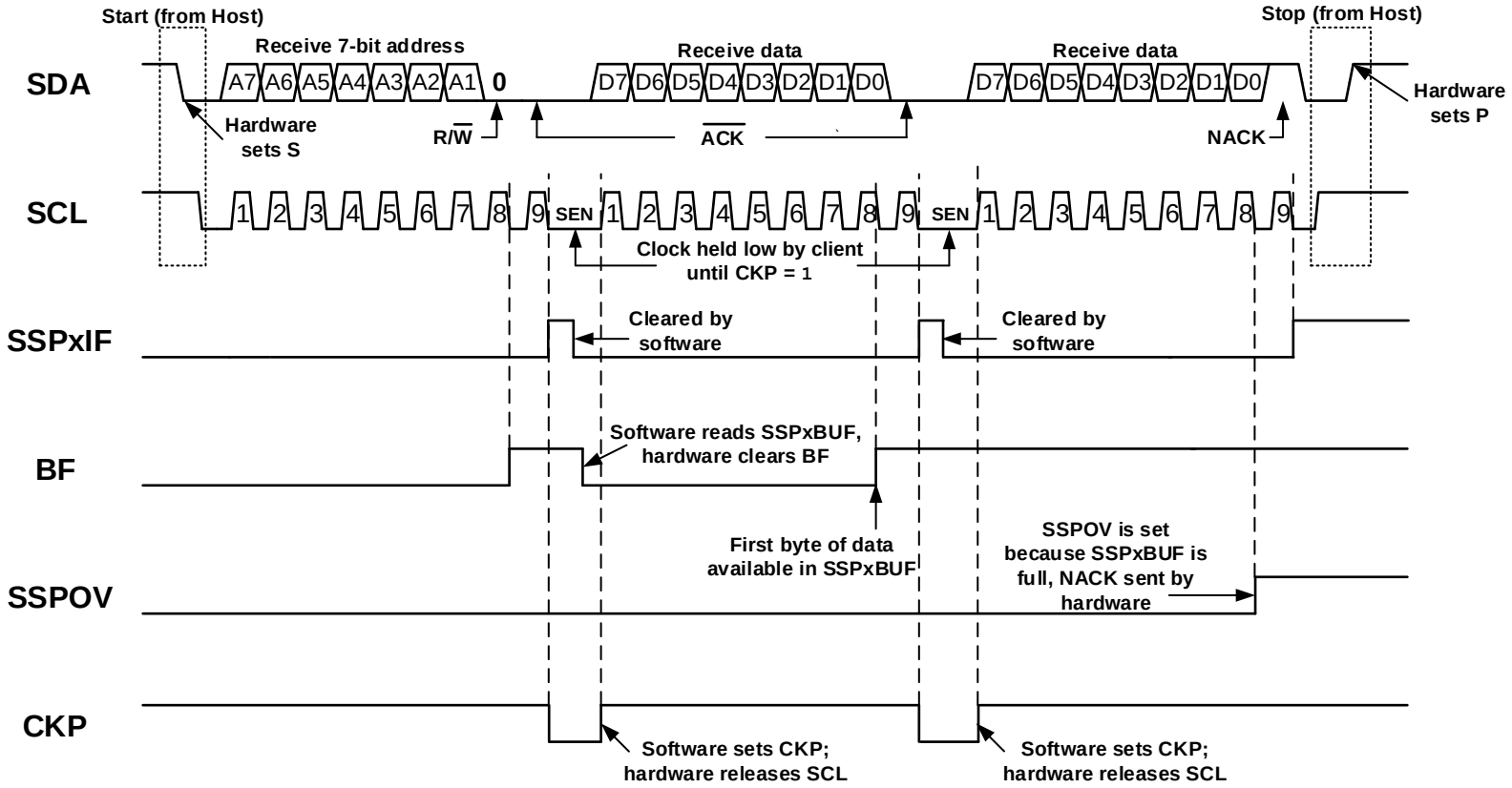


图 30-16. I²C 从模式接收时序（SEN = 1，AHEN = 0，DHEN = 0，7 位地址）



30.2.3.6.2. 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 30-17 显示了同时使用地址和数据保持功能的模块。 图 30-18 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 30-17. I²C 从模式接收时序（SEN = 0，AHEN = 1，DHEN = 1，7 位地址）

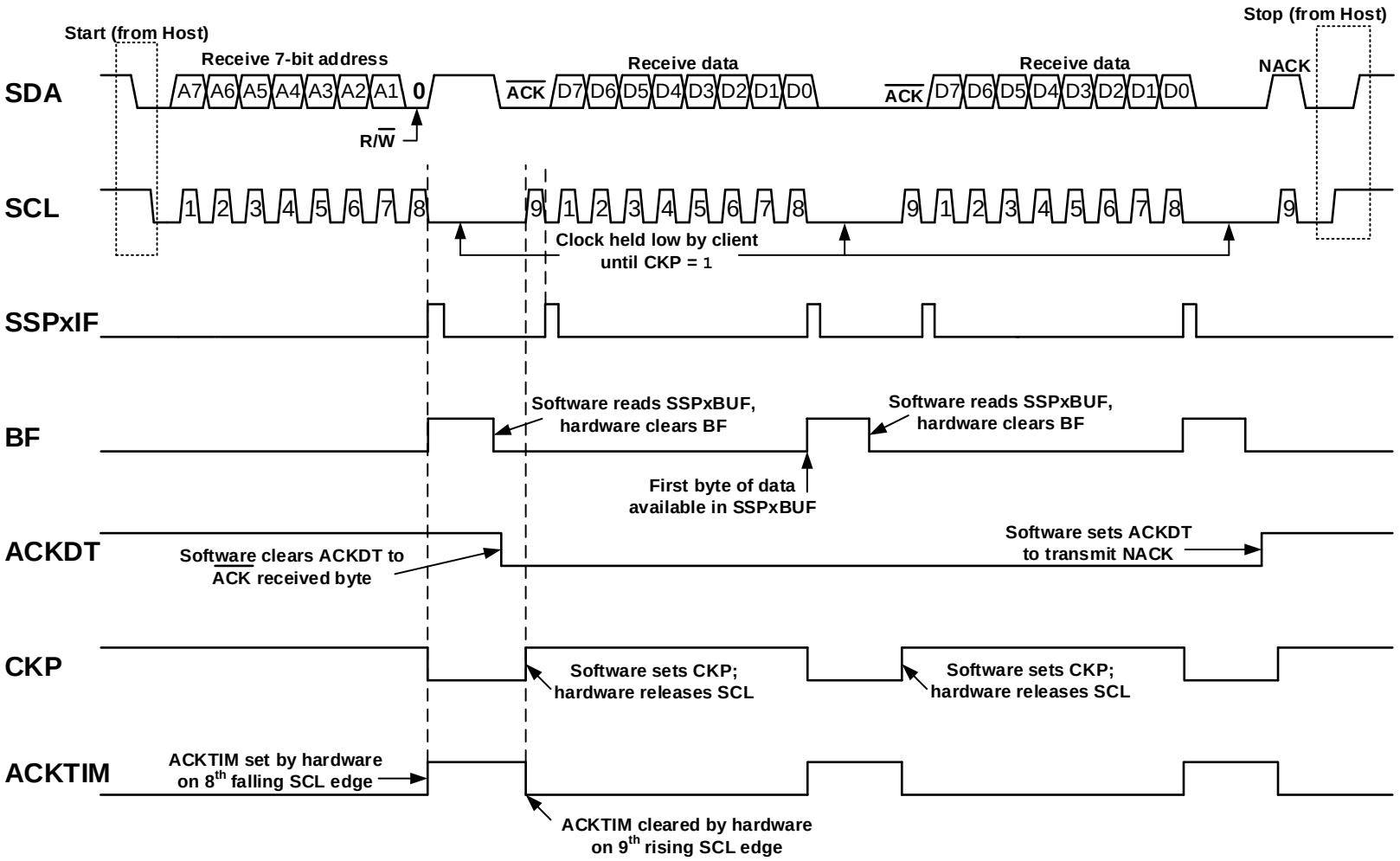
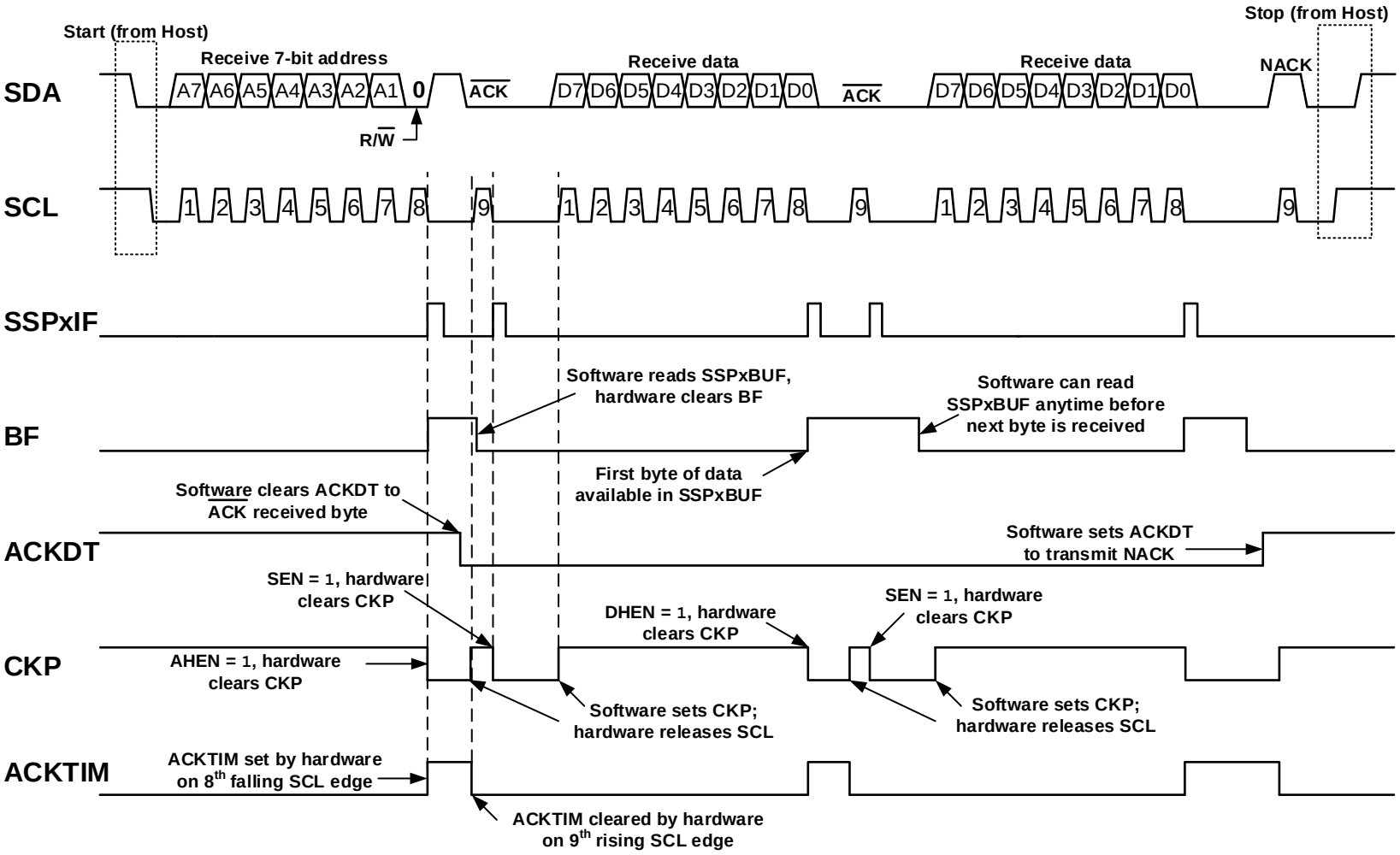


图 30-18. I²C 从模式接收时序（SEN = 1，AHEN = 1，DHEN = 1，7 位地址）



30.2.3.6.3. 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 30-19 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。

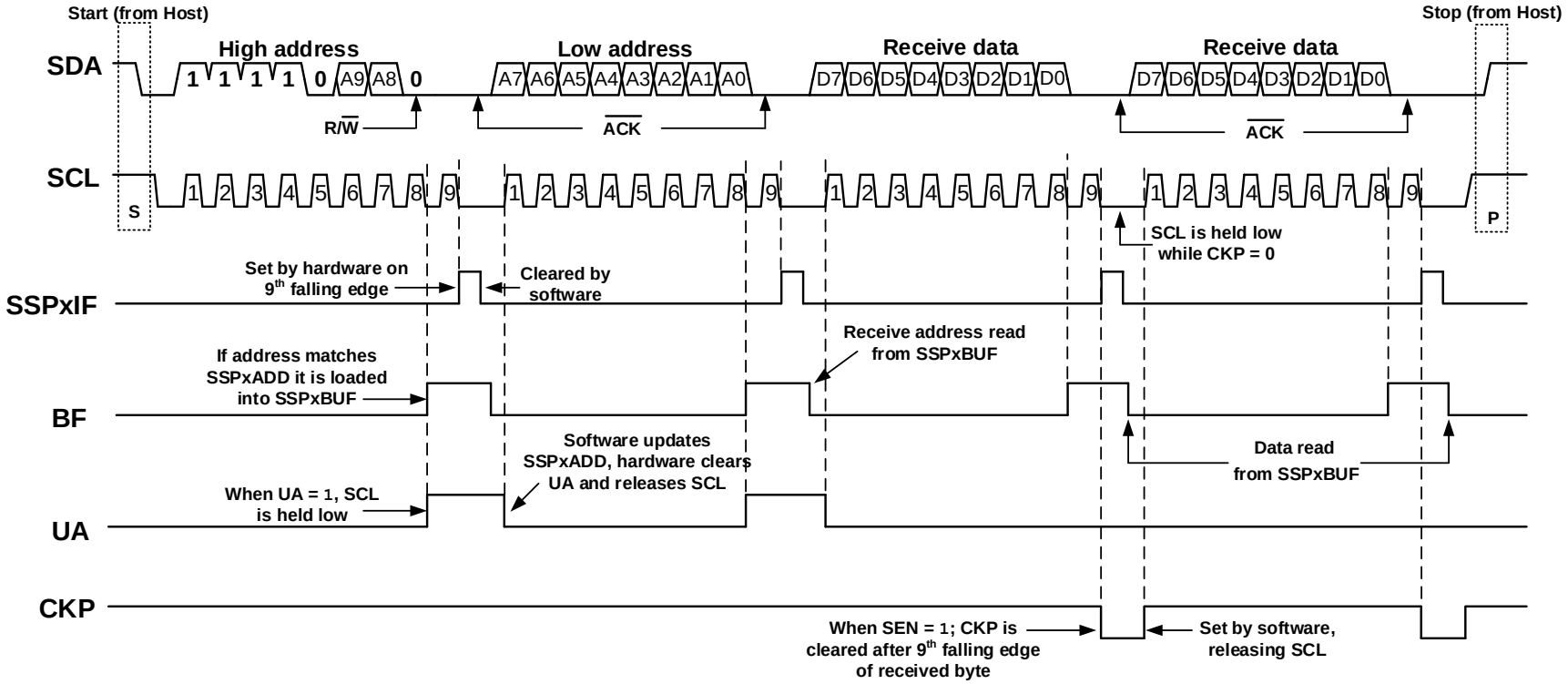
> **重要：**只有在 $\overline{ACK}$ 序列之后，才允许更新 SSPxADD 寄存器。

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 30-19. I²C 从模式接收时序（SEN = 1，AHEN = 0，DHEN = 0，10 位地址）



30.2.3.6.4. 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 30-20 可以用作 AHEN 置 1 时 10 位寻址模式下从器件的参考图示。

图 30-21 给出了 10 位寻址模式下从发送器的标准波形图。

图 30-20. I²C 从模式接收时序（SEN = 0，AHEN = 1，DHEN = 0，10 位地址）

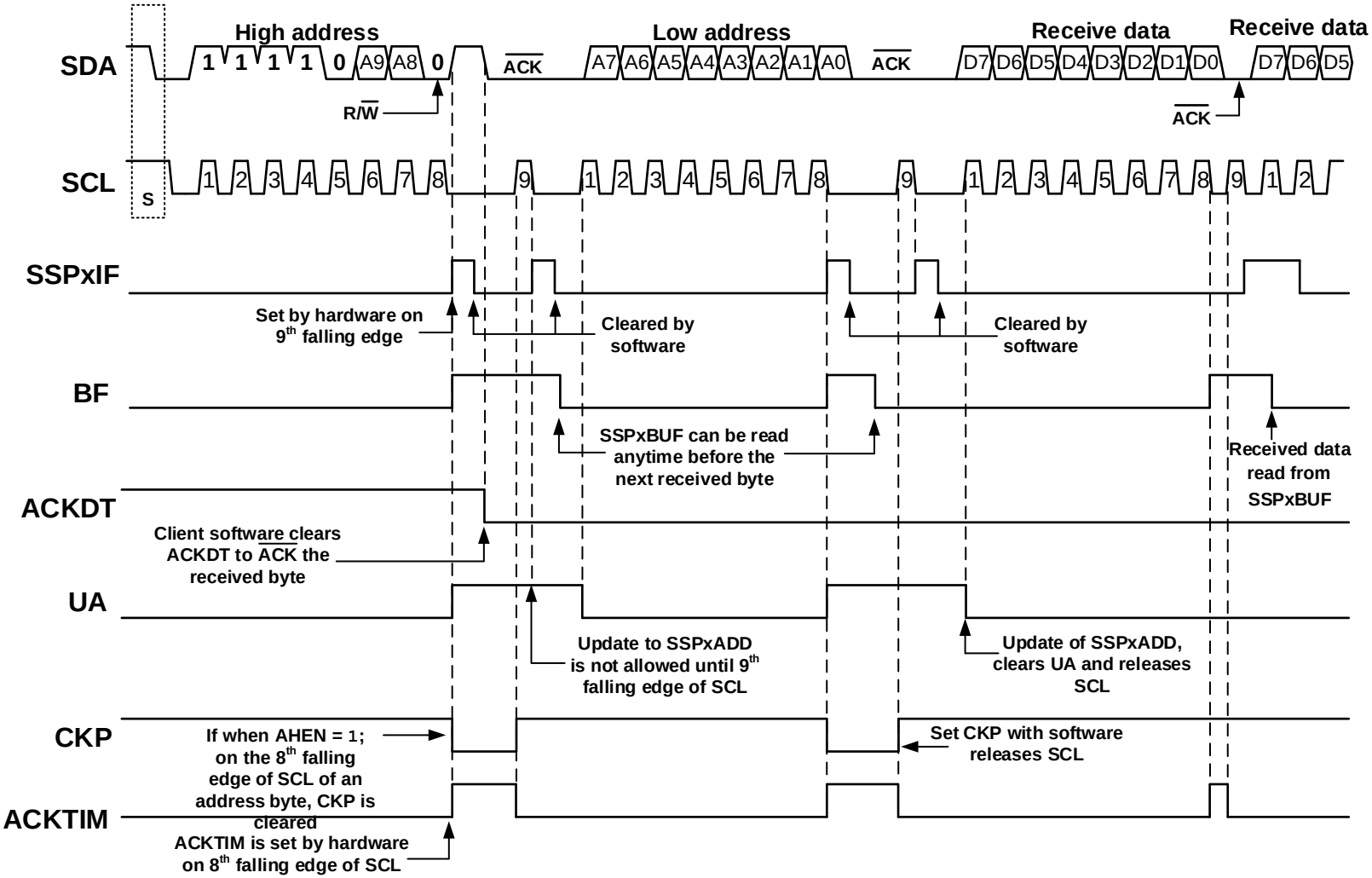
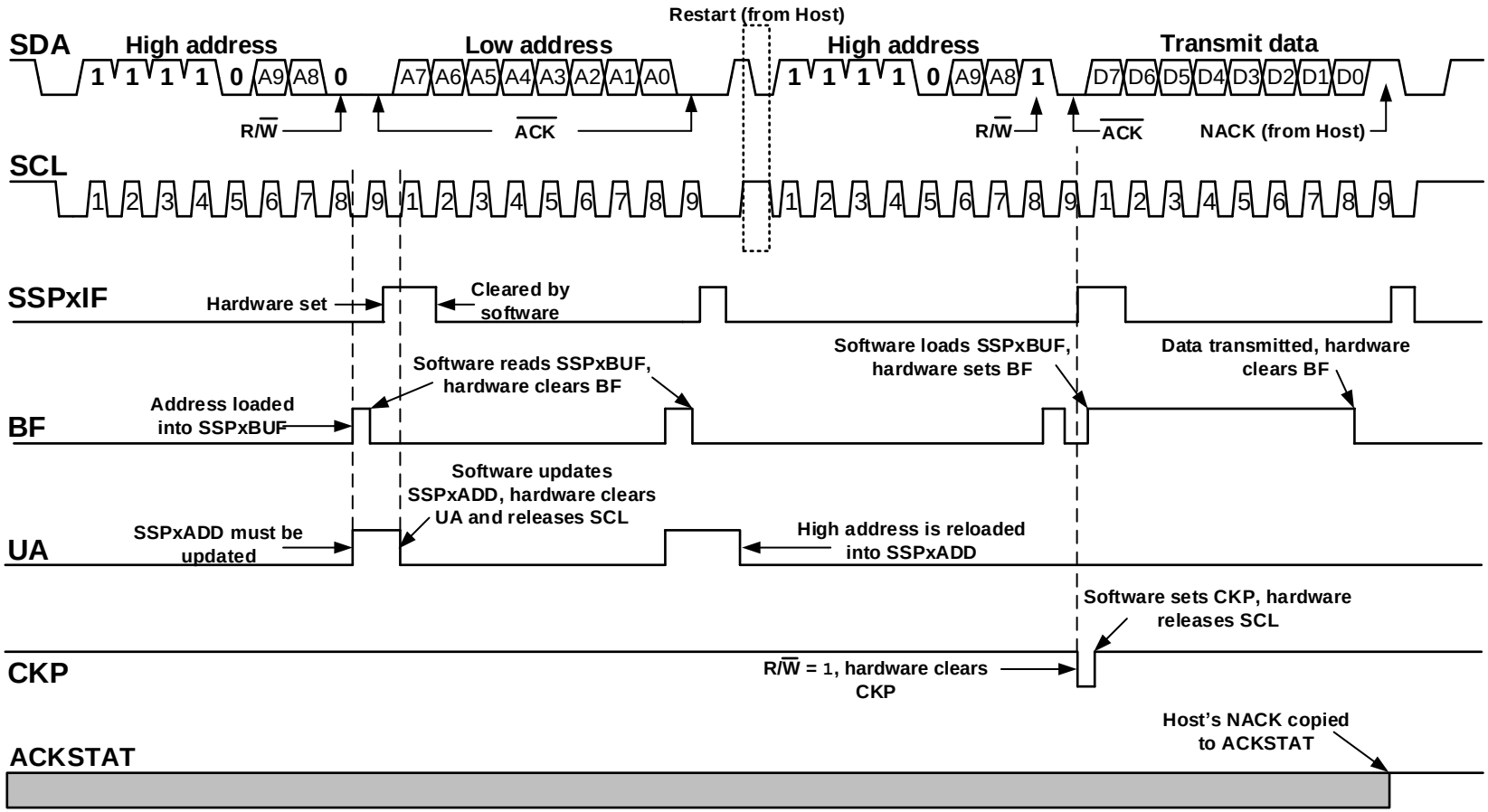


图 30-21. I²C 从模式发送时序（SEN = 0，AHEN = 0，DHEN = 0，10 位地址）



30.2.3.7. 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。

30.2.3.7.1. 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。

30.2.3.7.2. 7 位发送

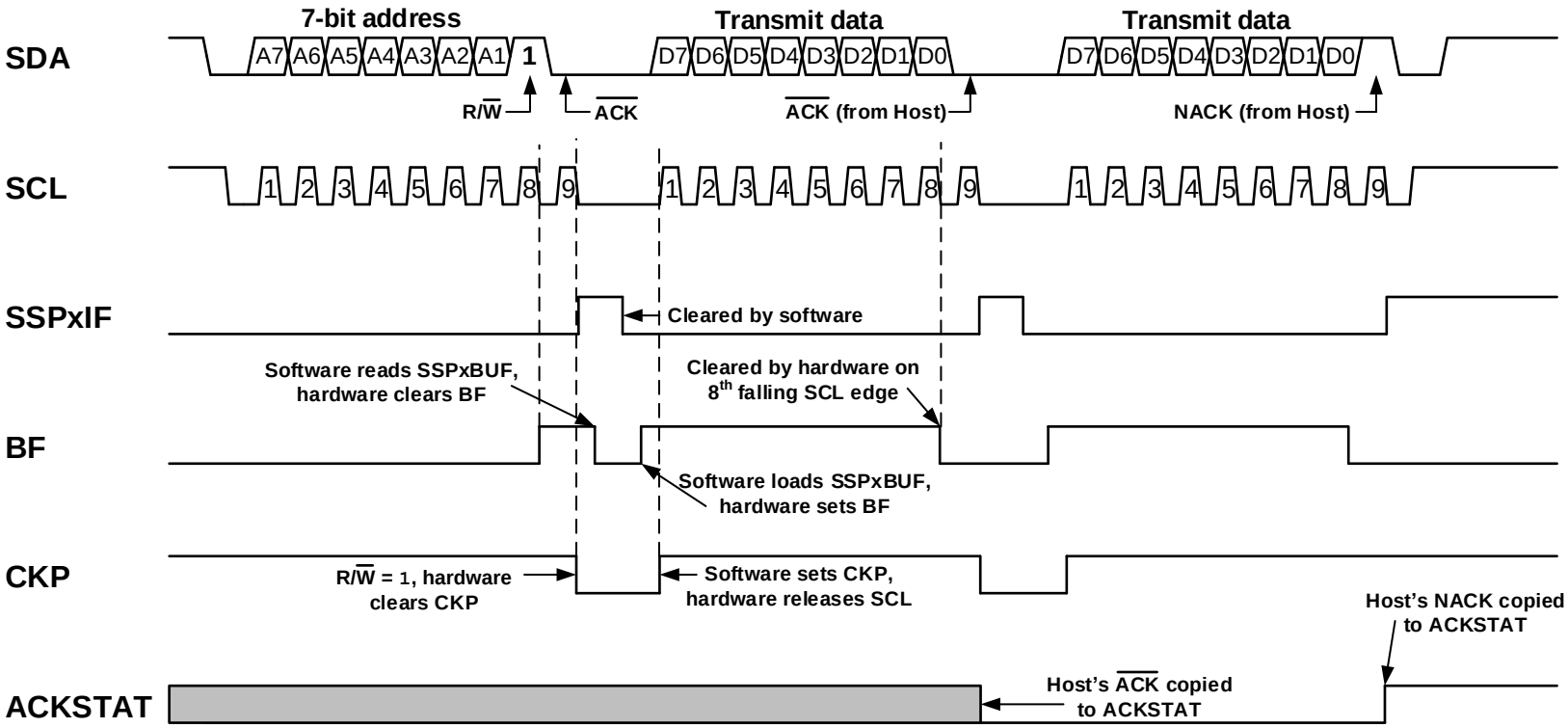
主器件可以向从器件发送读请求，然后随时钟从从器件中移出数据。下面列出了在实现标准数据发送时，从软件需要执行的操作。 图 30-22 可用作该列表的参考。

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 30-22. I²C 从模式发送时序（AHEN = 0，7 位地址）



30.2.3.7.3. 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 30-23 给出了在使能 AHEN 时 7 位地址从发送的标准波形图。

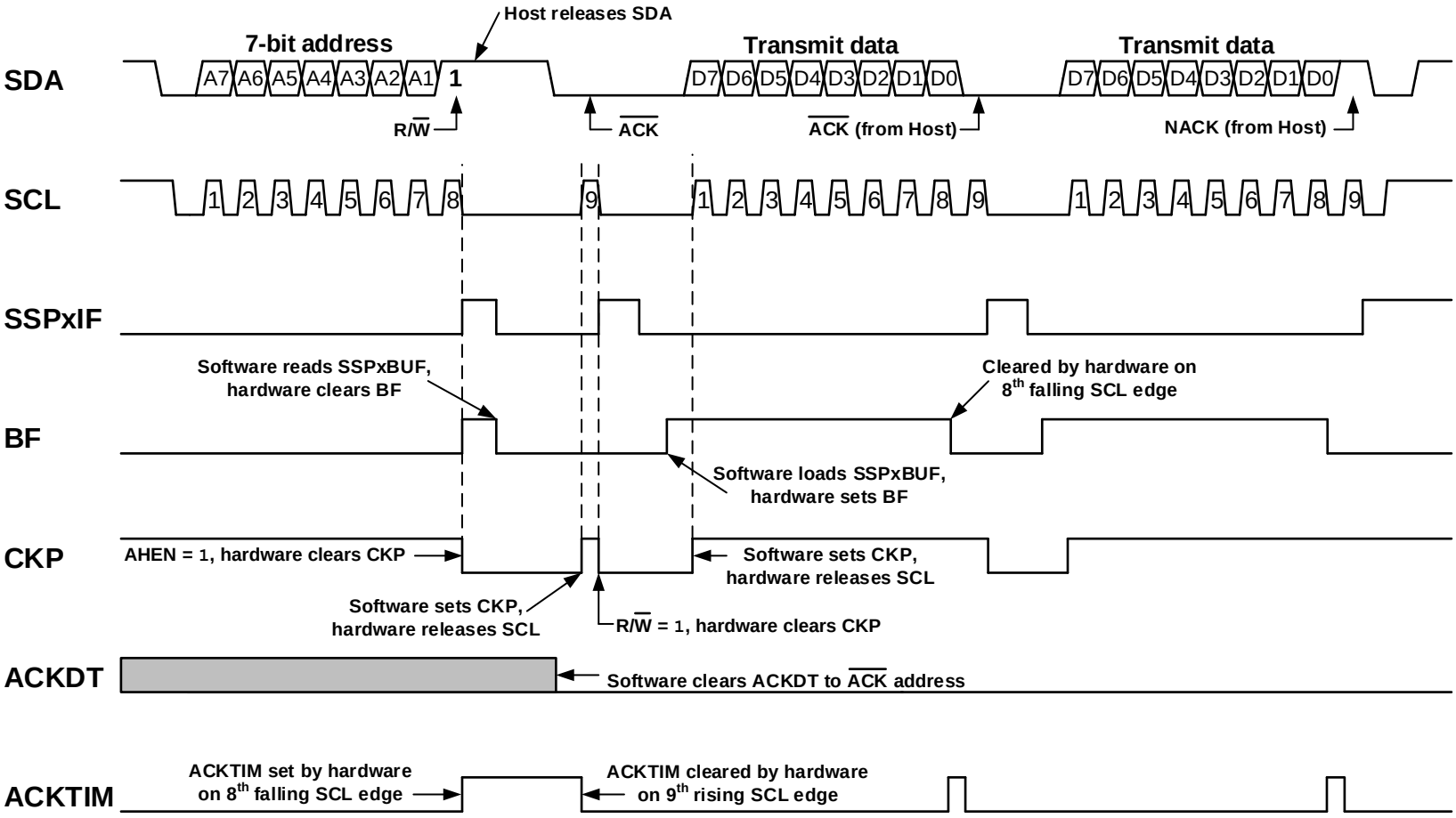
1. 总线启动时为空闲模式。
2. 主器件发送启动条件；S 位置 1；如果 SCIE 置 1，则 SSPxIF 也置 1。
3. 主器件发送 R/$\overline{W}$ 位置 1 的匹配地址。在 SCL 线的第 8 个下降沿之后，CKP 位清零，并产生 SSPxIF 中断。
4. 从器件软件清零 SSPxIF。
5. 从器件软件读取 ACKTIM、R/$\overline{W}$ 和 D/$\overline{A}$ 位来确定中断源。
6. 从器件从 SSPxBUF 寄存器中读取地址值，使 BF 位清零。
7. 从器件软件根据该信息确定它是产生 $\overline{ACK}$ 还是产生 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 30-23. I²C 从模式发送时序（AHEN = 1，7 位地址）



30.2.4. 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。

30.2.4.1. 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。

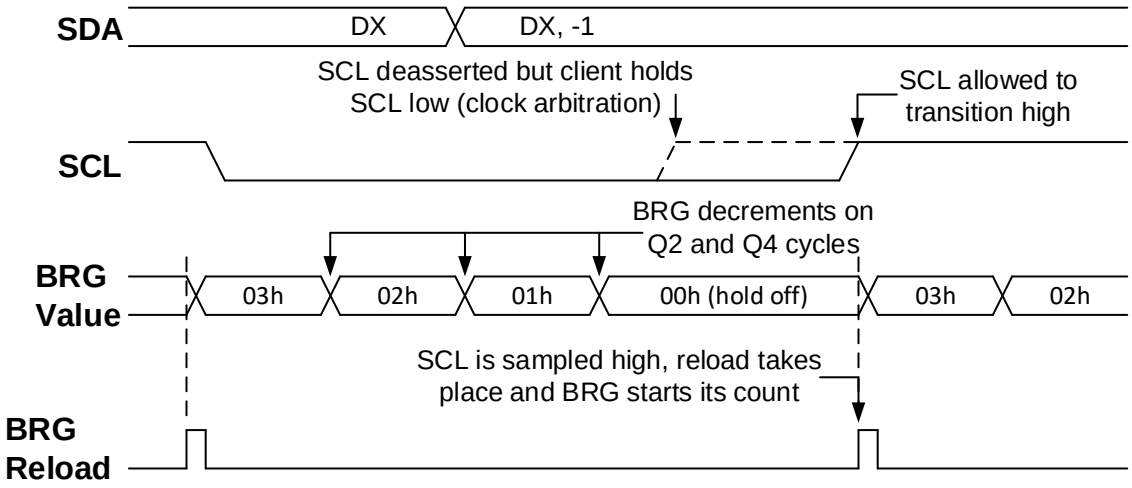
在主接收模式下，发送的第一个字节包括发送器件的从器件地址（7 位）和 R/$\overline{W}$ 位。在这种情况下，R/$\overline{W}$ 位将为逻辑 1。因此，第一个发送的字节是 7 位从器件地址以及随后用于指示接收位的 1。串行数据通过 SDA 接收，而串行时钟由 SCL 输出。一次接收 8 位串行数据。每接收到一个字节，都会发送一个应答序列。启动条件和停止条件指示发送的开始和结束。

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。

30.2.4.1.1. 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 30-24 所示）。

图 30-24. 带有时钟仲裁的波特率发生器时序



30.2.4.1.2. 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。

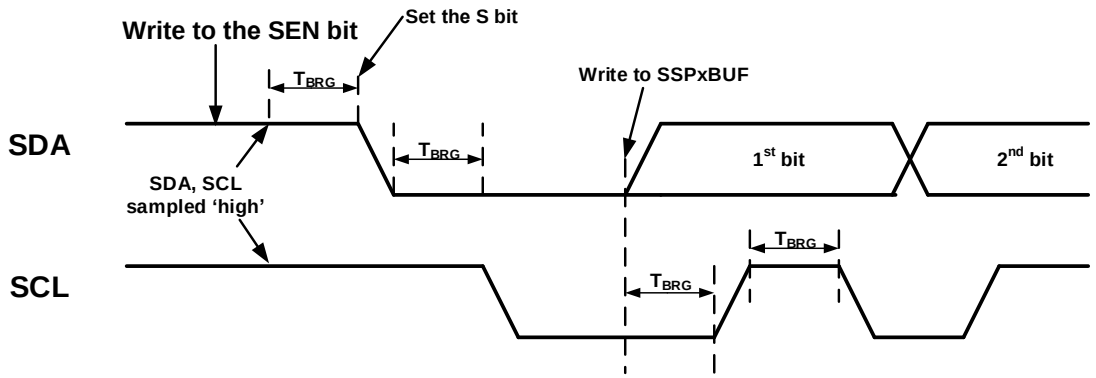
30.2.4.1.3. I²C 主模式启动条件时序

要发起启动条件（见图 30-25），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 30-25. 第一个启动位时序



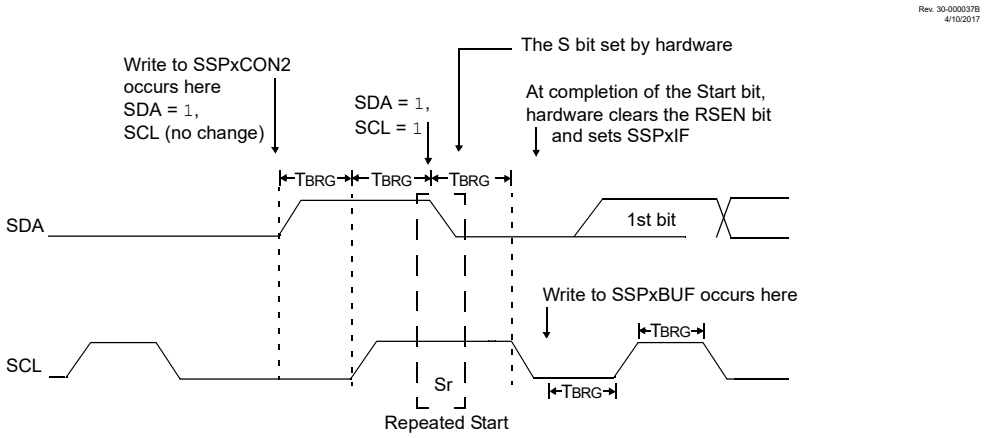
30.2.4.1.4. I²C 主模式重复启动条件时序

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 30-26）。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。

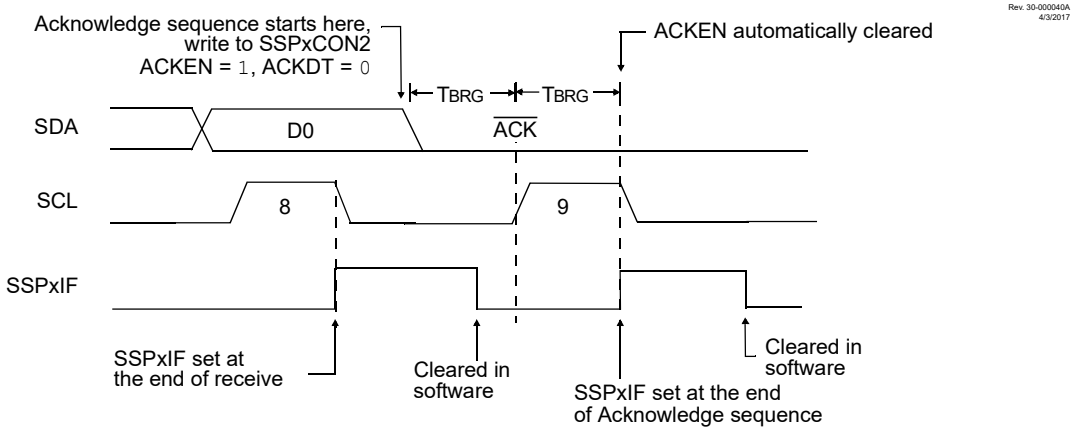
图 30-26. 重复启动条件波形图



30.2.4.1.5. 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 30-27）。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 30-27. 应答序列波形图



Note: TBRG = one Baud Rate Generator period.

应答写冲突

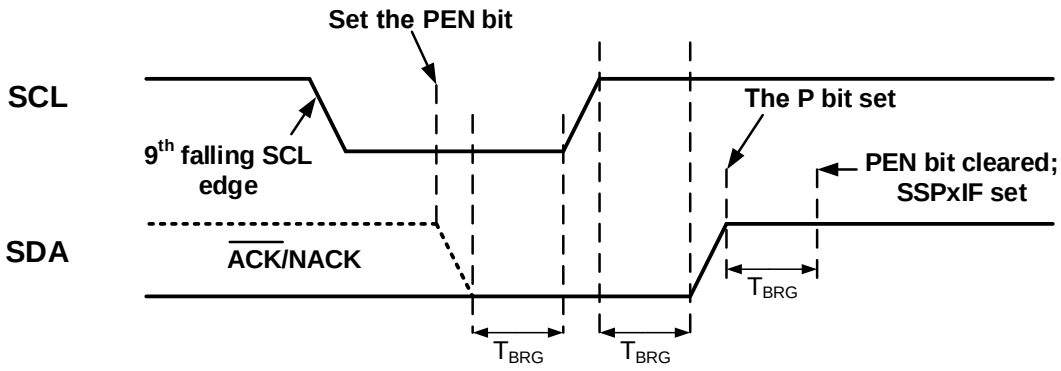
如果在应答序列进行过程中用户写 SSPxBUF，则 WCOL 位被置 1，同时缓冲区内容不变（未发生写操作）。

30.2.4.1.6. 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 30-28）。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 30-28. 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）。

30.2.4.1.7. 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）。

30.2.4.1.8. 复位的影响

复位会禁止 MSSP 模块并终止当前的数据传输。

30.2.4.2. 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 30-29）。

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。

30.2.4.2.1. 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。

30.2.4.2.2. 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。

30.2.4.2.3. 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 $\overline{ACK}$。

30.2.4.2.4. 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 30-29. I^2C 主模式波形图（发送，7 位地址）

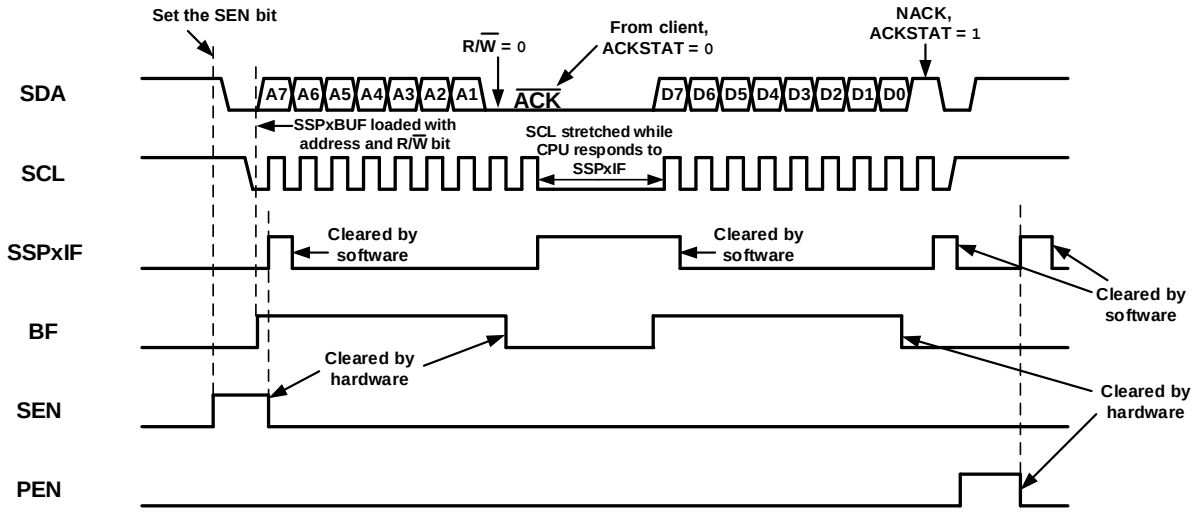
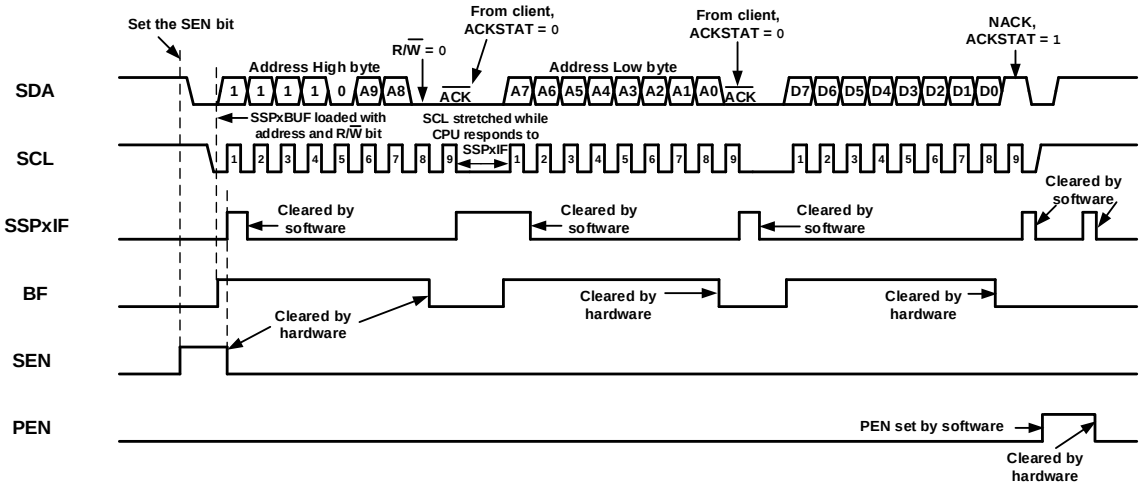


图 30-30. I²C 主模式波形图（发送，10 位地址）



30.2.4.3. 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 30-31）。

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。

30.2.4.3.1. 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。

30.2.4.3.2. 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。

30.2.4.3.3. 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）。

30.2.4.3.4. 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 30-31. I²C 主模式波形图（接收，7 位地址）

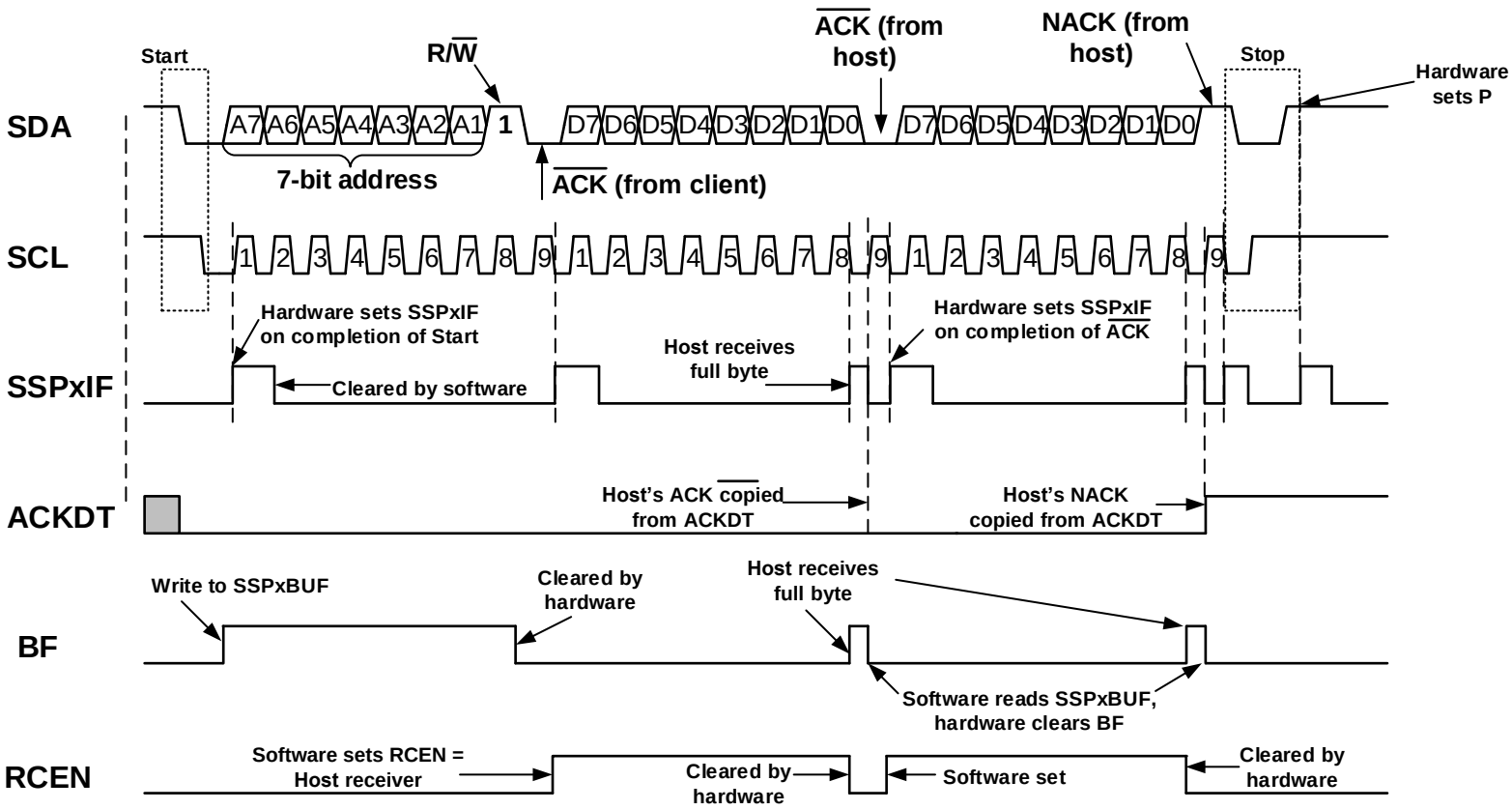
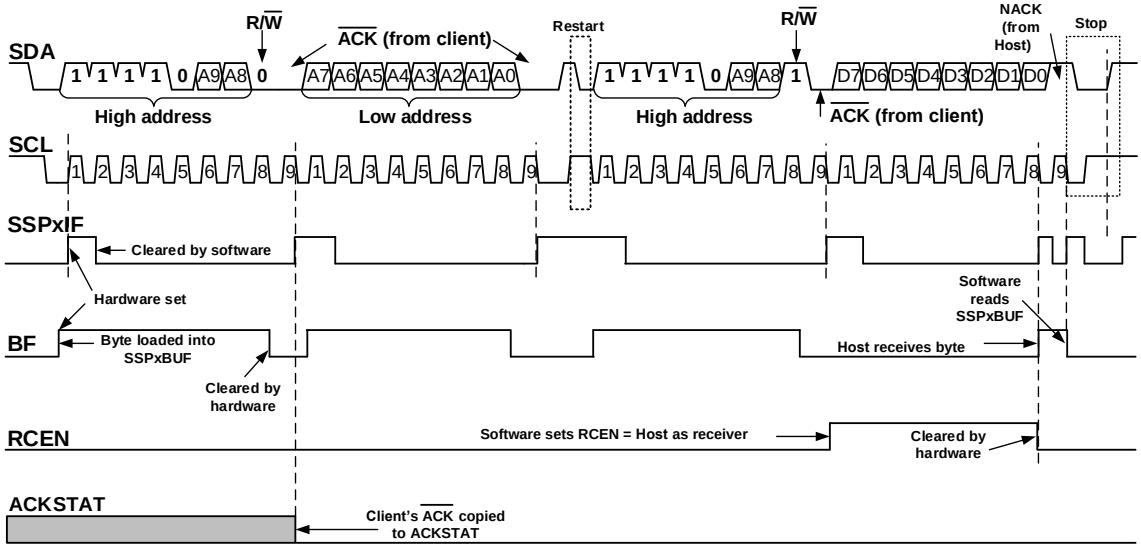


图 30-32. I²C 主模式波形图（接收，10 位地址）



30.2.5. 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

30.2.5.1. 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 30-33）。

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。

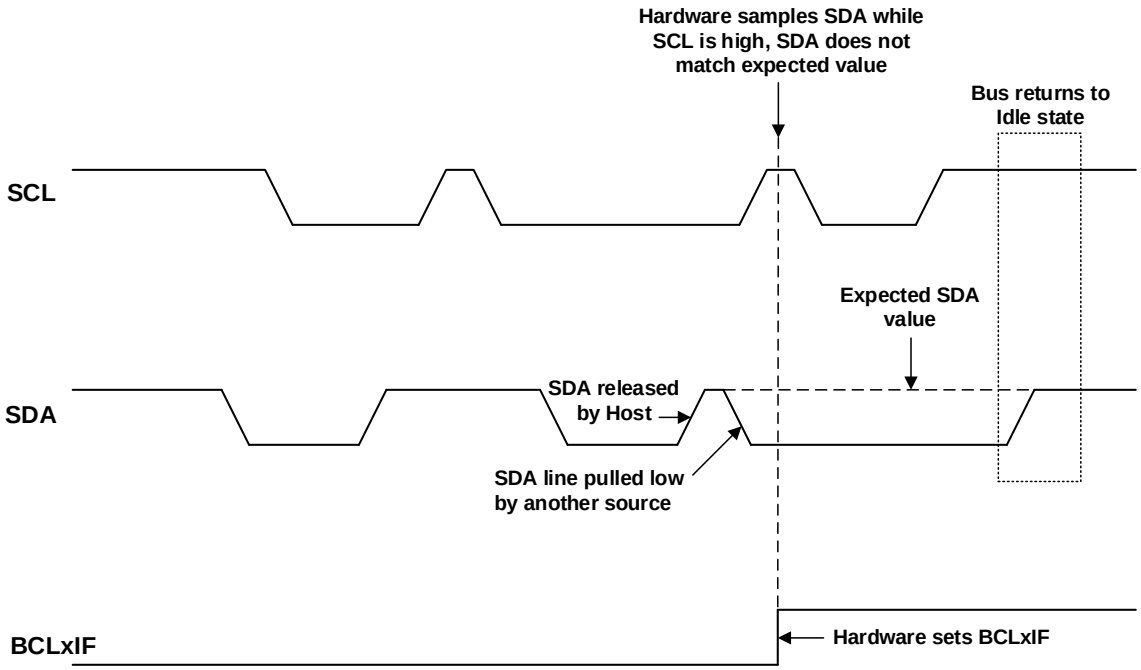
如果在启动、重复启动、停止或应答条件的进行过程中发生总线冲突，则条件将被中止，SDA 和 SCL 线被置为无效，SSPxCON2 寄存器中的对应控制位清零。当执行总线冲突中断服务程序时，如果 I²C 总线空闲，软件可通过发出启动条件恢复通信。

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 30-33. 发送和应答时的总线冲突时序



30.2.5.1.1. 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：

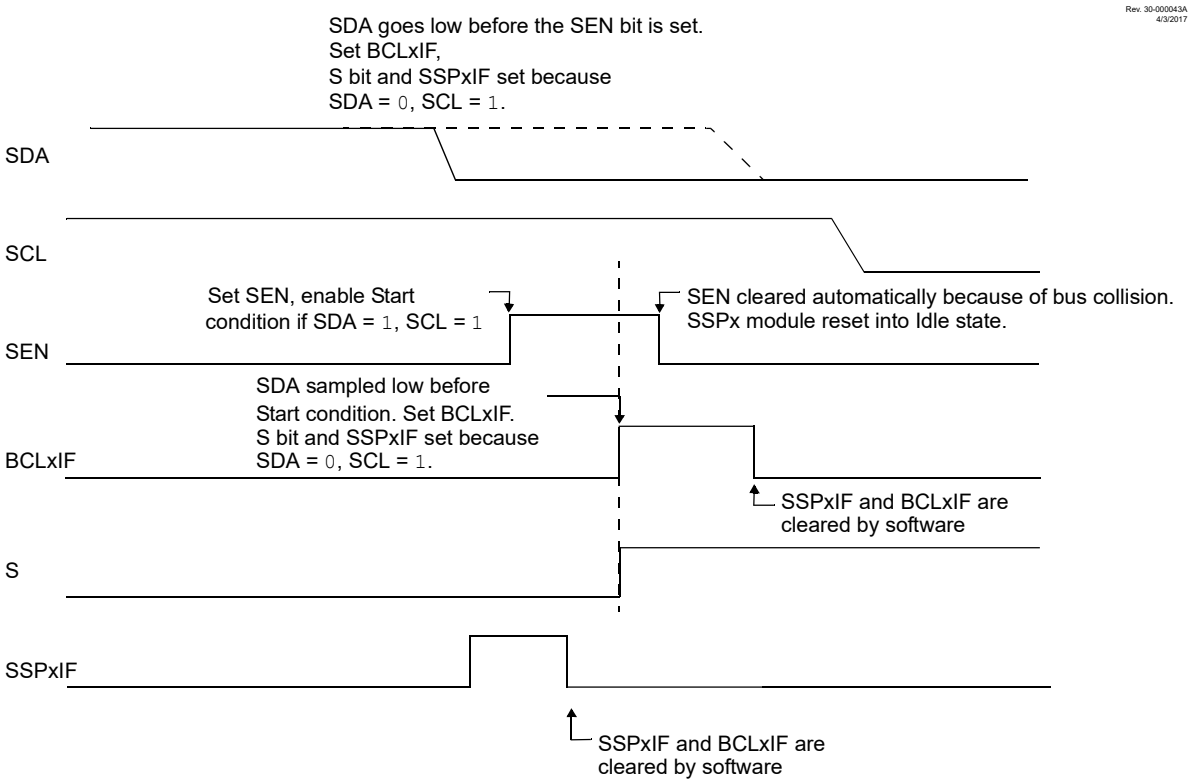
1. 在启动条件开始时，采样到 SDA 或 SCL 为低电平（见图 30-34）。
2. SDA 置为低电平之前，采样到 SCL 为低电平（见图 30-35）。

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：

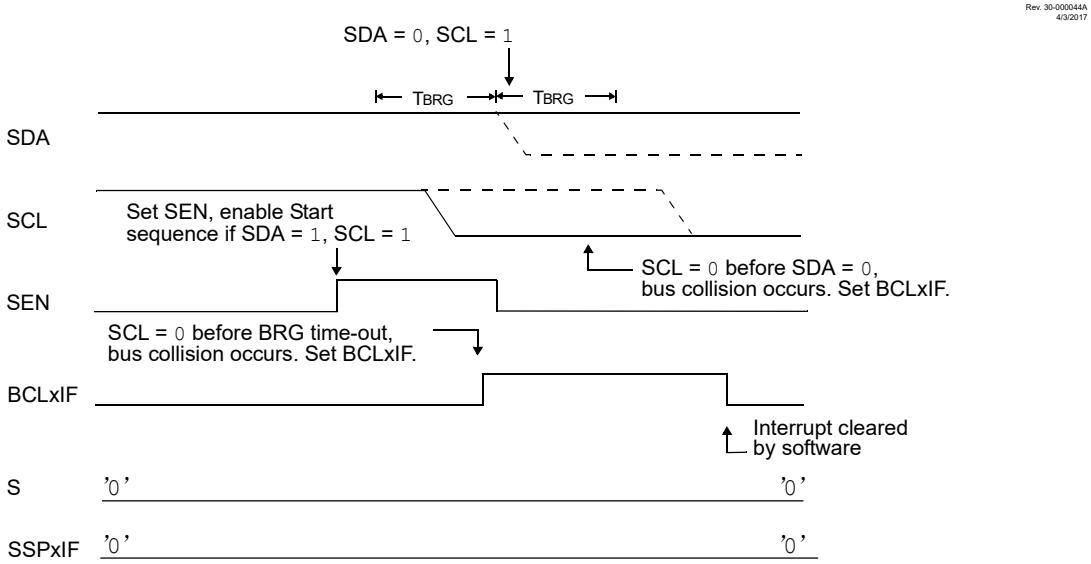
- 中止启动条件，
- BCLxIF 标志置 1，并且
- MSSP 模块复位为空闲状态（见图 30-34）。

图 30-34. 启动条件期间的总线冲突（仅用于 SDA）



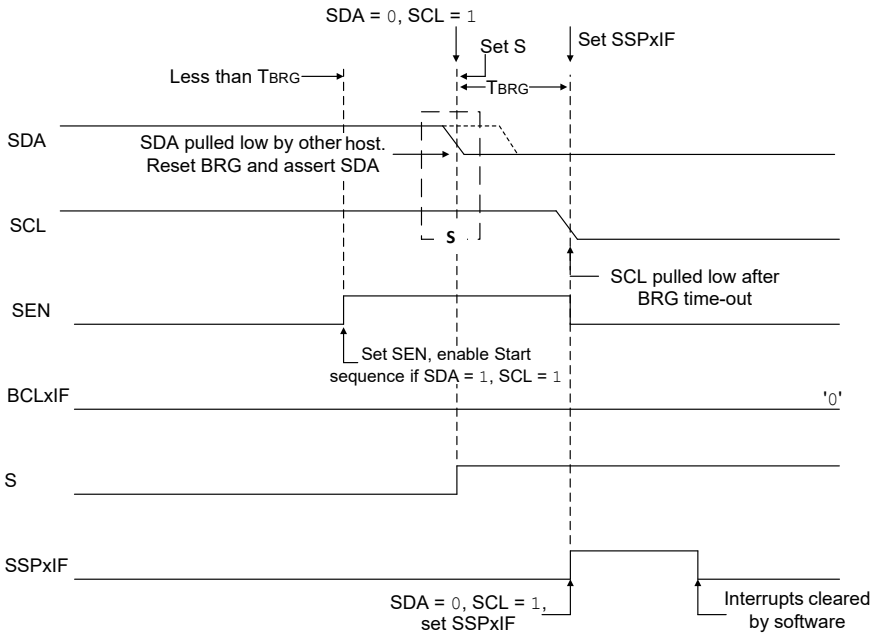
启动条件从 SDA 和 SCL 引脚被置为无效时开始。当采样到 SDA 引脚为高电平时，波特率发生器装入值并递减计数。如果在 SDA 为高电平时，采样到 SCL 引脚为低电平，则发生总线冲突，因为这表示另一个主器件在启动条件期间尝试驱动数据 1。

图 30-35. 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 30-36）。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 30-36. 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。

30.2.5.1.2. 重复启动条件期间的总线冲突

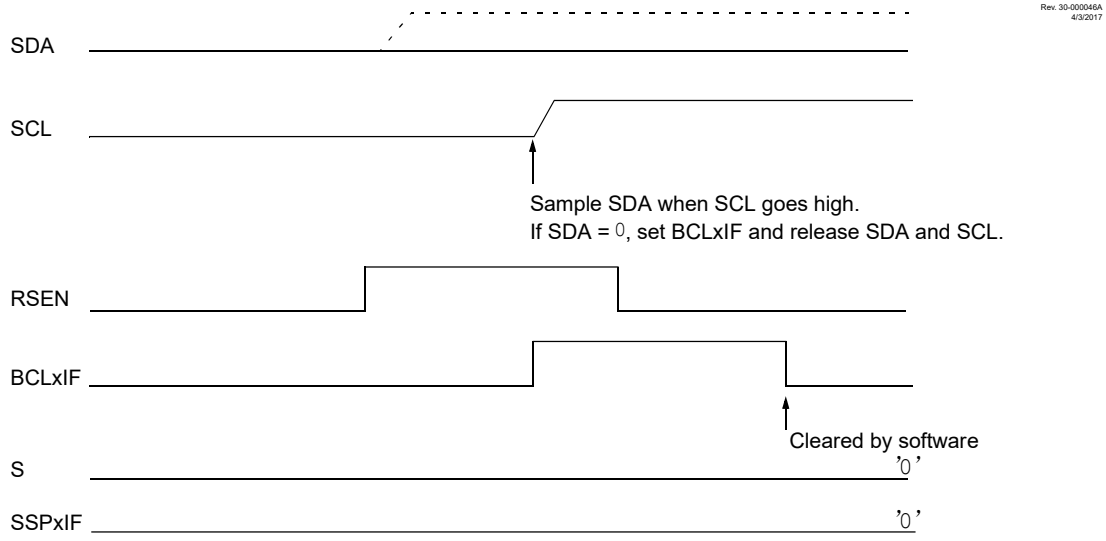
在重复启动条件期间如果发生以下情况，将发生总线冲突：

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 30-37）。
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 30-38）。

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 30-37）。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。

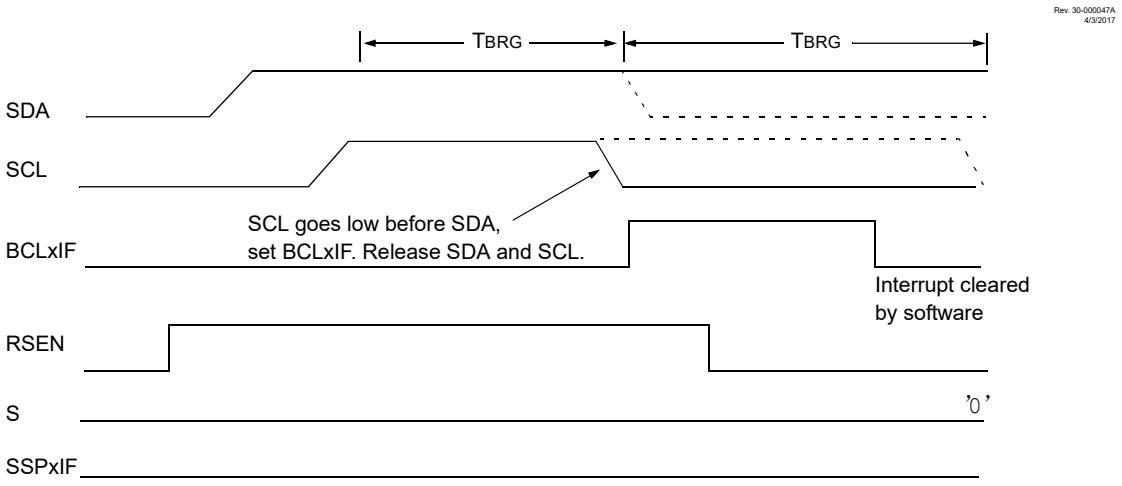
图 30-37. 重复启动条件期间的总线冲突（情形 1）



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 30-38）。

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 30-38. 重复启动条件期间的总线冲突（情形 2）



30.2.5.1.3. 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 30-39）。

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 30-40）。

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 30-39）。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 30-40）。

图 30-39. 停止条件期间的总线冲突（情形 1）

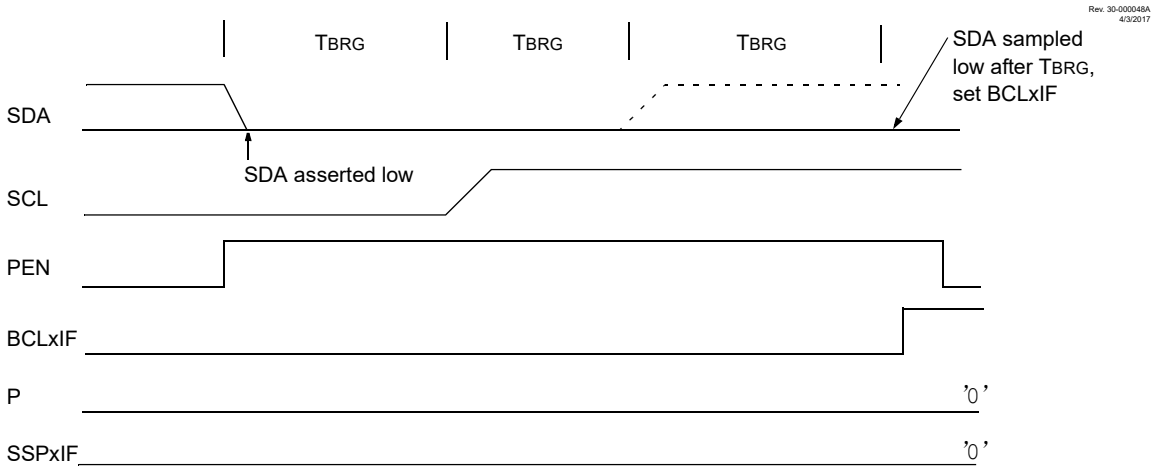
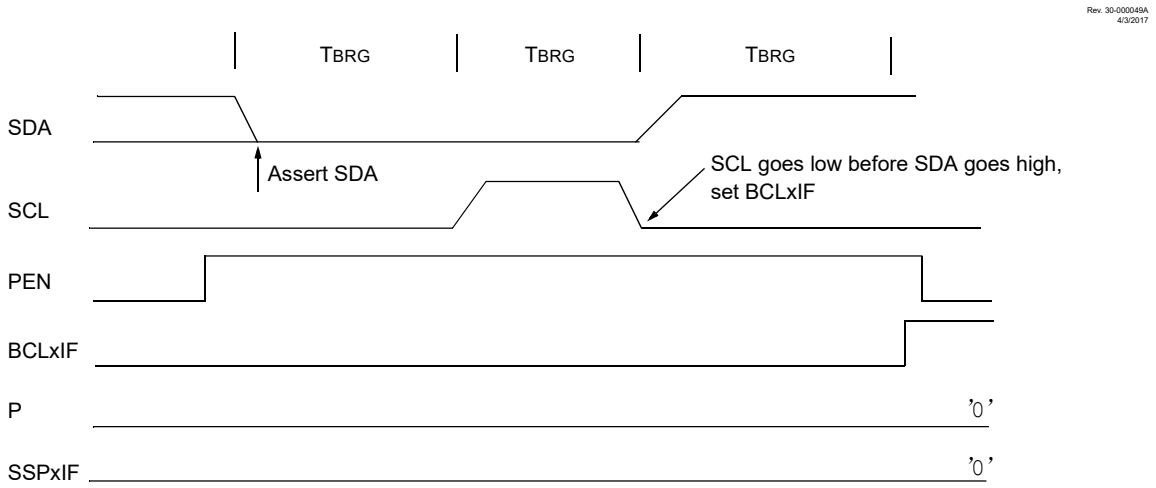


图 30-40. 停止条件期间的总线冲突（情形 2）



30.3. 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 30-1 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。

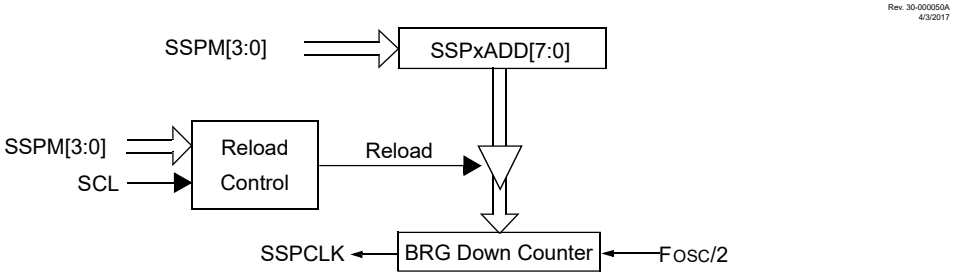
图 30-41 中显示的内部“重载”信号会触发将 SSPxADD 值装入 BRG 计数器。对于模块时钟线的每次振荡，这会发生两次。

表 30-3 演示了不同指令周期下的时钟速率以及装入 SSPxADD 的 BRG 值。

例 30-1. MSSP 波特率发生器频率公式

$$F_{CLOCK} = \frac{F_{OSC}}{4 \times (SSPxADD + 1)}$$

图 30-41. 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 30-3. 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。

30.4. 寄存器定义：MSSP 控制

30.4.1. SSPxBUF

名称： SSPxBUF
偏移量： 0xF91,0xE8D

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

30.4.2. SSPxADD

名称： SSPxADD
偏移量： 0xF92,0xE8E

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	模式	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。

30.4.3. SSPxMSK

名称： SSPxMSK
偏移量： 0xF93,0xE8F

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	模式	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	模式	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

30.4.4. SSPxSTAT

名称： SSPxSTAT
偏移量： 0xF94,0xE90

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	模式	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	模式	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	模式	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	模式	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	模式	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	模式	说明
x	SPI	不使用该位
1	I^2C 从模式	读
0	I^2C 从模式	写

值	模式	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	模式	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	模式	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。

30.4.5. SSPxCON1

名称： SSPxCON1
偏移量： 0xF95,0xE91

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	模式	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	模式	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（必须用软件清零）。
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	模式	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)

值	说明
0111	I^2C 从模式：10 位地址
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。

30.4.6. SSPxCON2

名称： SSPxCON2
偏移量： 0xF96,0xE92

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）。

30.4.7. SSPxCON3

名称： SSPxCON3
偏移量： 0xF97,0xE93

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	模式	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	模式	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	模式	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	模式	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	模式	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	模式	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	模式	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	模式	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。

30.5. 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 ... 0x0E8C	保留									
0x0E8D	SSP2BUF	7:0	BUF[7:0]							
0x0E8E	SSP2ADD	7:0	ADD[7:0]							
0x0E8F	SSP2MSK	7:0	MSK[6:0]							MSK0
0x0E90	SSP2STAT	7:0	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
0x0E91	SSP2CON1	7:0	WCOL	SSPOV	SSPEN	CKP	SSPM[3:0]			
0x0E92	SSP2CON2	7:0	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
0x0E93	SSP2CON3	7:0	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
0x0E94 ... 0x0F90	保留									
0x0F91	SSP1BUF	7:0	BUF[7:0]							
0x0F92	SSP1ADD	7:0	ADD[7:0]							
0x0F93	SSP1MSK	7:0	MSK[6:0]							MSK0
0x0F94	SSP1STAT	7:0	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
0x0F95	SSP1CON1	7:0	WCOL	SSPOV	SSPEN	CKP	SSPM[3:0]			
0x0F96	SSP1CON2	7:0	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
0x0F97	SSP1CON3	7:0	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN

31. 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。

表 31-1. EUSART 模块可用性

器件	EUSART1	EUSART2
CN2510	有	无
CN2610	有	有
CN2710	有	有

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

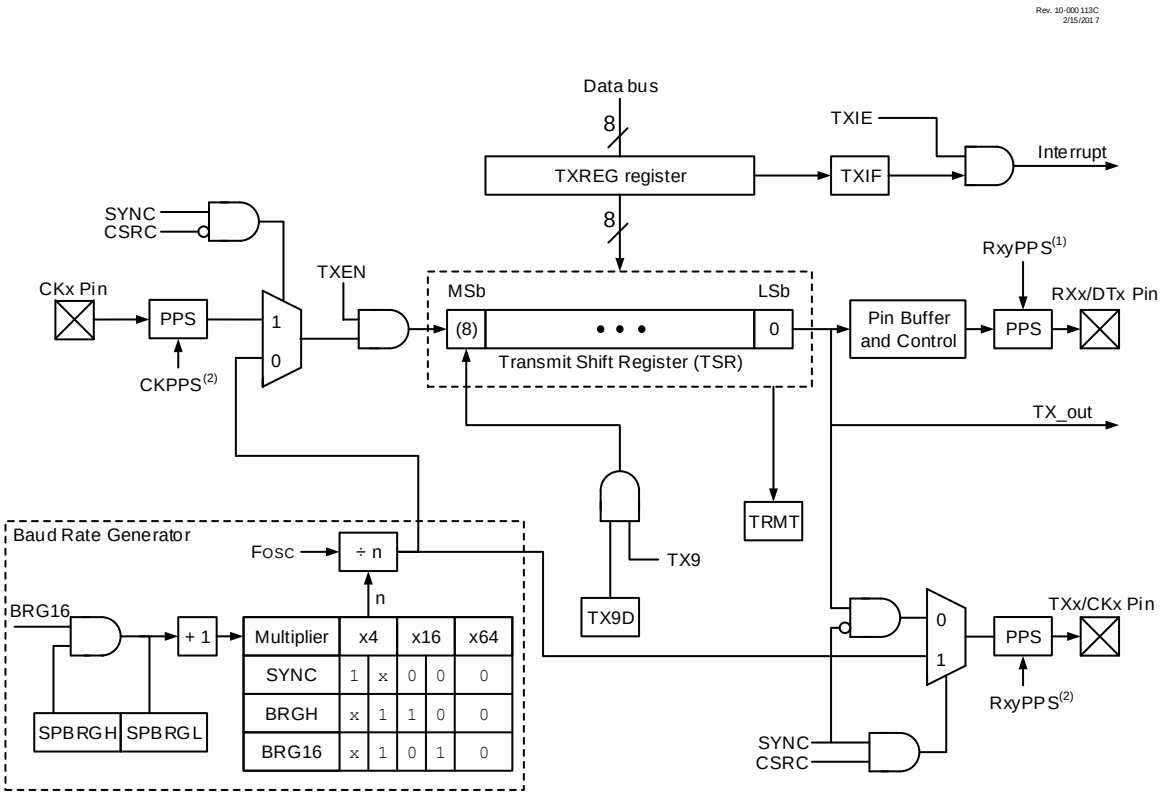
EUSART 发送器和接收器的框图如图 31-1 和图 31-2 所示。

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）

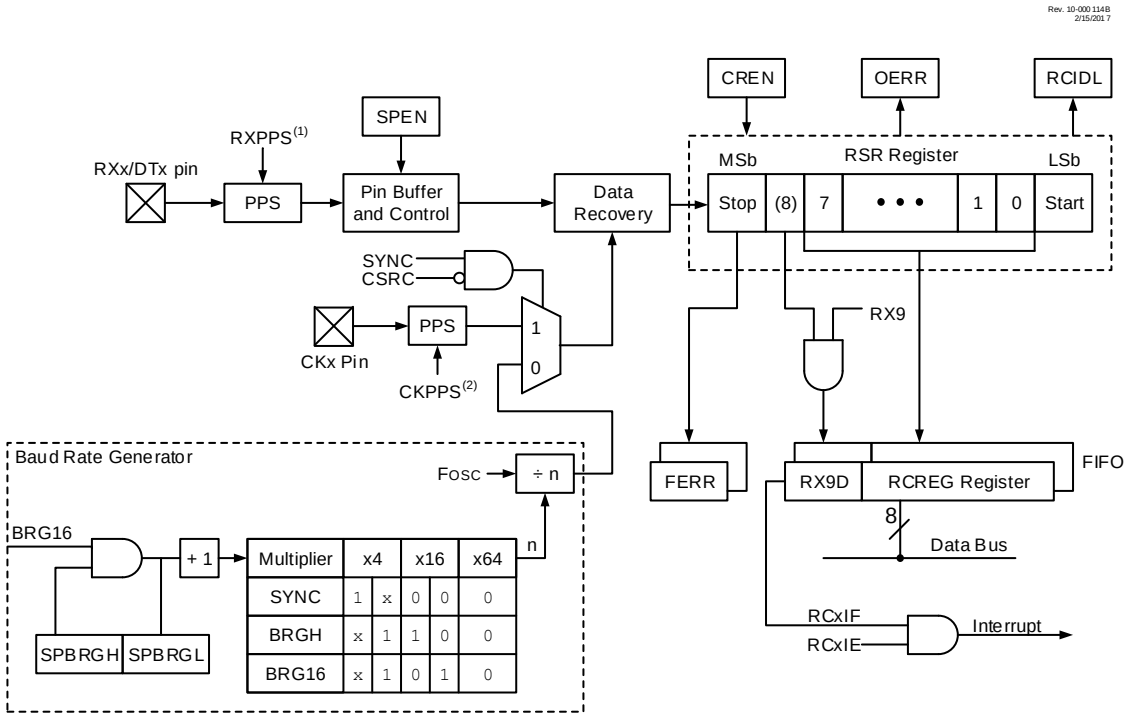
RXx/DTx 和 TXx/CKx 输入引脚分别使用 RXxPPS 和 TXxPPS 寄存器进行选择。TXx、CKx 和 DTx 输出引脚使用每个引脚的 RxyPPS 寄存器进行选择。由于在同步模式下 RX 输入与 DT 输出耦合在一起，所以在同步模式下工作时，用户需要负责为这两个功能选择同一引脚。EUSART 控制逻辑将自动控制数据方向驱动器。

图 31-1. 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 31-2. 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.

31.1. 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 31-3。

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。

31.1.1. EUSART 异步发送器

图 31-1 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。

31.1.1.1. 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。

31.1.1.2. 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。

31.1.1.3. 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。

31.1.1.4. 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。

31.1.1.5. 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。

31.1.1.6. 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。

31.1.1.7. 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 31-3. 异步发送

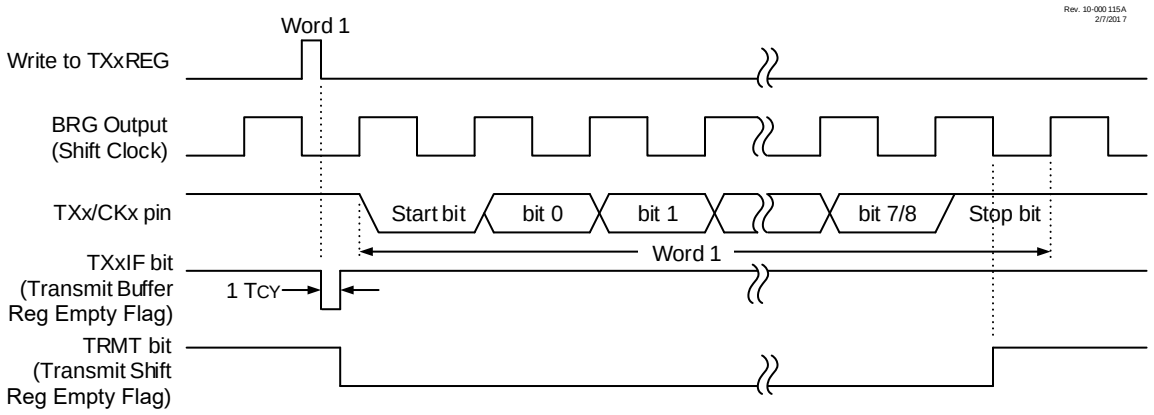
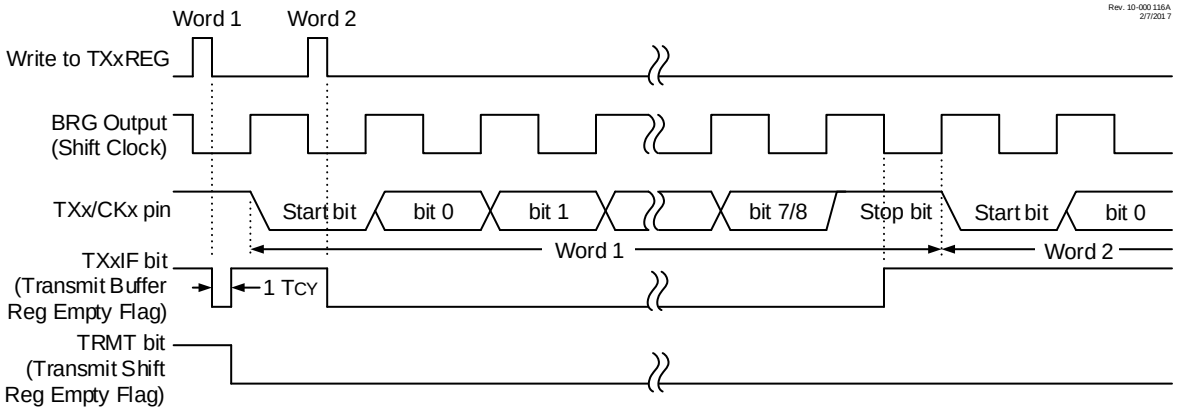


图 31-4. 异步发送（连续）



31.1.2. EUSART 异步接收器

异步模式通常用于 RS-232 系统中。图 31-2 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。

31.1.2.1. 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。

31.1.2.2. 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。

31.1.2.3. 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。

31.1.2.4. 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。

31.1.2.5. 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。

31.1.2.6. 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。

31.1.2.7. 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。

31.1.2.8. 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。

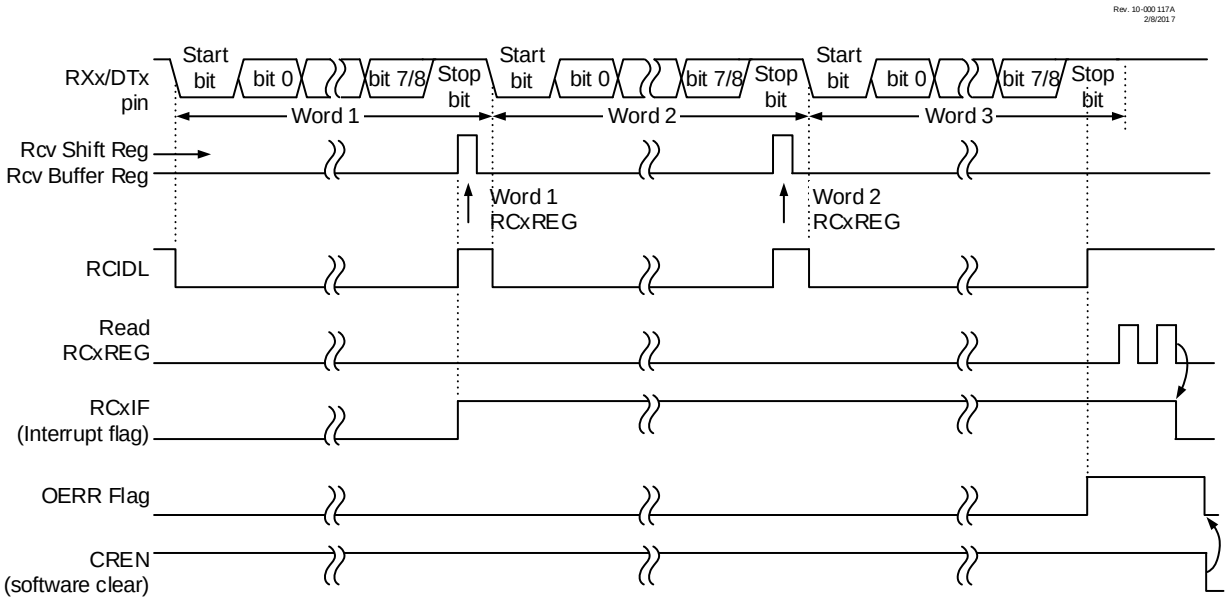
31.1.2.9. 9 位地址检测模式设置

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 31-5. 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.

31.2. 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。

31.3. 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 31-2 提供了确定波特率的公式。 公式 31-1 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 31-1. 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：

$$\text{目标波特率} = \frac{F_{OSC}}{64 \times (SPxBRG + 1)}$$

求解 SPxBRG：

$$SPxBRG = \frac{F_{OSC}}{64 \times \text{目标波特率}} - 1$$

$$SPxBRG = \frac{16000000}{64 \times 9600} - 1$$

$$SPxBRG = 25.042 \simeq 25$$

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

$$\text{计算波特率} = 9615$$

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

$$\text{误差} = \frac{9615 - 9600}{9600}$$

$$\text{误差} = 0.16\,\%$$

表 31-2. 波特率公式

配置位			BRG/EUSART 模式	波特率公式
SYNC	BRG16	BRGH		
0	0	0	8 位/异步	Fosc/[64 (n+1)]
0	0	1	8 位/异步	Fosc/[16 (n+1)]
0	1	0	16 位/异步	
0	1	1	16 位/异步	Fosc/[4 (n+1)]
1	0	x	8 位/同步	
1	1	x	16 位/同步	

注： x = 任意值，n = SPxBRGH:SPxBRGL 寄存器对的值。

表 31-3. 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	—	—	—

31.3.1. 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 31-6 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 31-4 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：

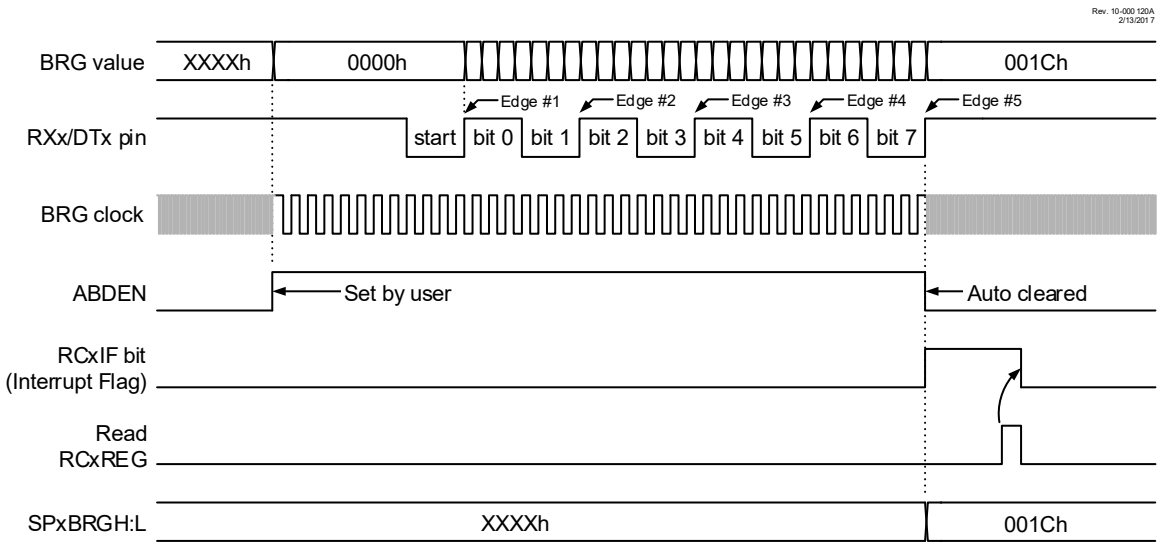
1. 如果唤醒使能（WUE）位和 ABDEN 位都置 1，自动波特率检测将从间隔字符之后的字节开始（见接收到间隔字符时自动唤醒）。
2. 需要由用户来判断输入字符的波特率是否处于所选 BRG 时钟源范围内。可能无法实现某些振荡器频率和 EUSART 波特率组合。
3. 在自动波特率检测过程中，自动波特率计数器从 1 开始计数。自动波特率序列完成后，为了得到最准确的结果，应从 SPxBRGH:SPxBRGL 寄存器对的值中减去 1。

表 31-4. 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 31-6. 自动波特率校准



31.3.2. 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。

31.3.3. 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 31-7 所示）；在休眠模式下，与 Q 时钟异步产生中断（如图 31-8 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。

31.3.3.1. 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。

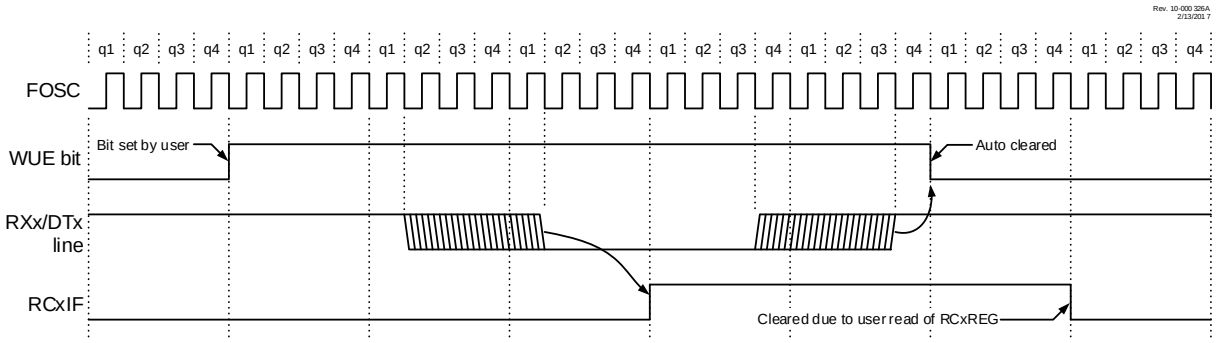
因此，发送的首字符必须为全 0。这必须持续 10 个或更长的位时间，对于 LIN 总线，建议持续 13 个位时间，而对于标准 RS-232 器件，可为任意个位时间。

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。

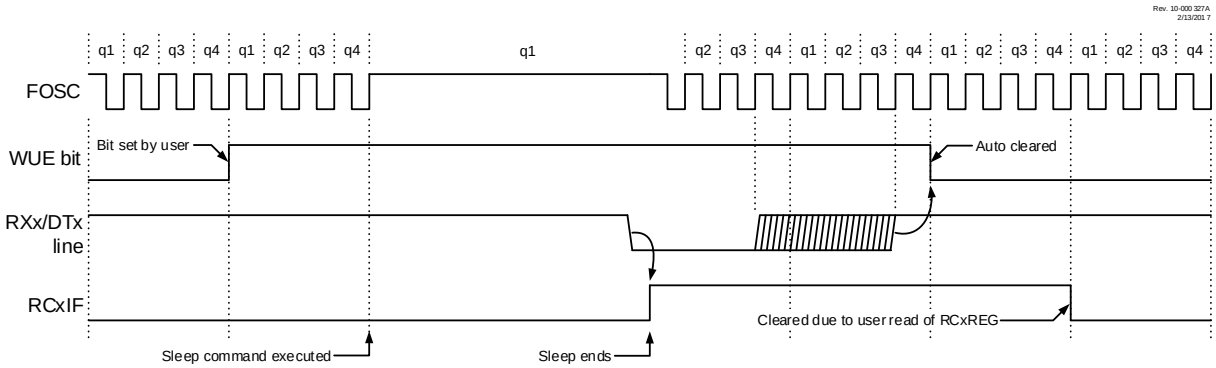
要确保不丢失实际数据，应在将 WUE 位置 1 前检查 RCIDL 位，验证没有接收操作在进行。如果未发生接收操作，可在进入休眠模式前将 WUE 位置 1。

图 31-7. 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 31-8. 休眠时的自动唤醒（WUE）位时序



Note: The EUSART remains in Idle while the WUE bit is set.

31.3.4. 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 31-9。

31.3.4.1. 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。

31.3.5. 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。

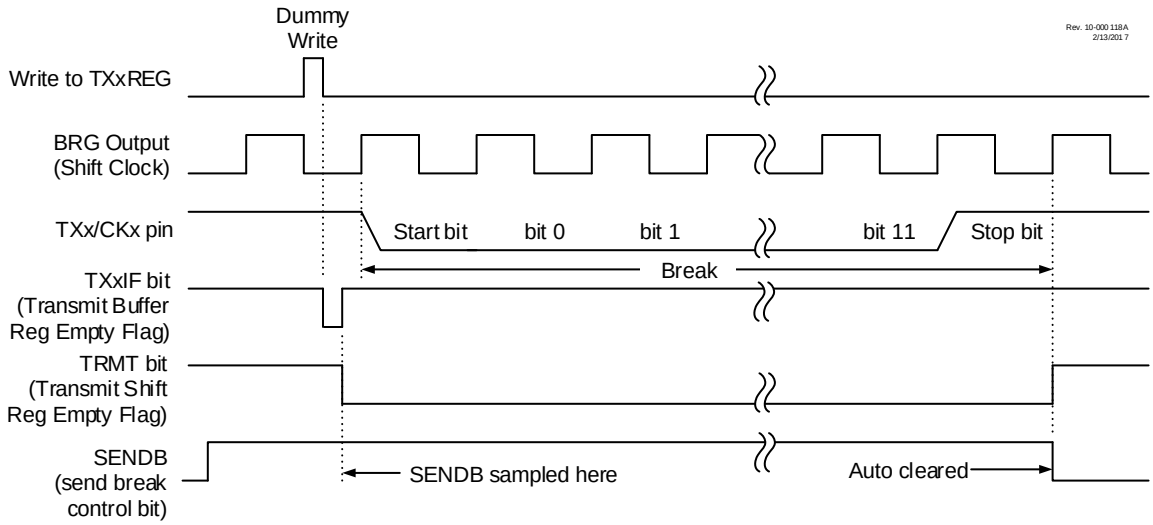
当以下三个条件全部为真时，表示接收到间隔字符：

- 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 31-9. 发送间隔字符序列



31.4. 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。

31.4.1. 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。

31.4.1.1. 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。

31.4.1.2. 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。

31.4.1.3. 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。

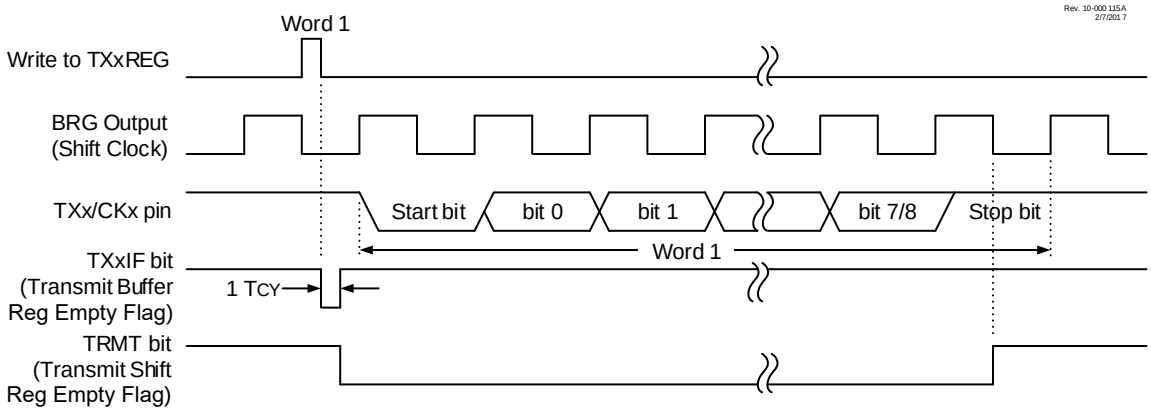
注：TSR 寄存器并未映射到数据存储器中，因此用户不能访问它。

31.4.1.4. 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 31-10. 同步发送



31.4.1.5. 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。

31.4.1.6. 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。

31.4.1.7. 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。

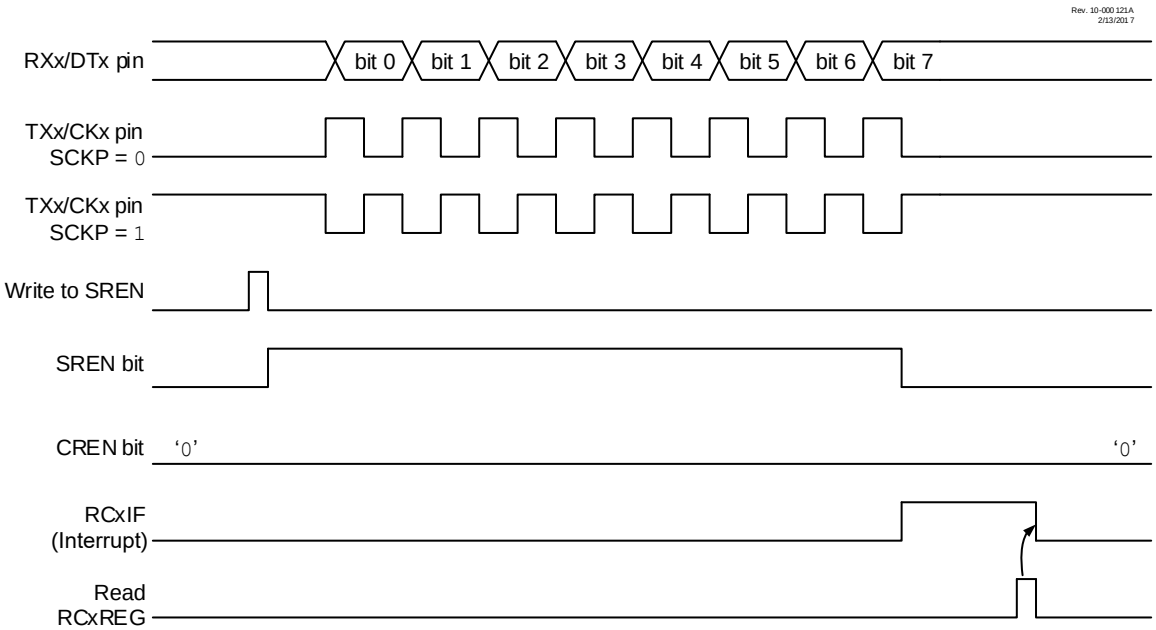
31.4.1.8. 接收 9 位字符

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。

31.4.1.9. 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 31-11. 同步接收（主模式，通过 SREN 控制）



31.4.2. 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。

31.4.2.1. 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。

31.4.2.2. 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。

31.4.2.3. 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。

31.4.2.4. 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。

31.5. 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。

31.5.1. 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）。

31.5.2. 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）。

31.6. 寄存器定义：EUSART 控制

31.6.1. TXxSTA

名称： TXxSTA
偏移量： 0xF9D,0xE99

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。

31.6.2. RCxSTA

名称： RCxSTA
偏移量： 0xF9C,0xE98

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。

31.6.3. BAUDxCON

名称： BAUDxCON
偏移量： 0xF9E,0xE9A

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

31.6.4. RCxREG

名称： RCxREG
偏移量： 0xF98,0xE94

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

31.6.5. TXxREG

名称： TXxREG
偏移量： 0xF99,0xE95

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

31.6.6. SPxBRG

名称： SPxBRG
偏移量： 0xF9A,0xE96

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]

31.7. 寄存器汇总——EUSART

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0E93	保留									
0x0E94	RC2REG	7:0	RCREG[7:0]							
0x0E95	TX2REG	7:0	TXREG[7:0]							
0x0E96	SP2BRG	7:0	SPBRG[7:0]							
		15:8	SPBRG[15:8]							
0x0E98	RC2STA	7:0	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
0x0E99	TX2STA	7:0	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
0x0E9A	BAUD2CON	7:0	ABDOVF	RCIDL		SCKP	BRG16		WUE	ABDEN
0x0E9B ... 0x0F97	保留									
0x0F98	RC1REG	7:0	RCREG[7:0]							
0x0F99	TX1REG	7:0	TXREG[7:0]							
0x0F9A	SP1BRG	7:0	SPBRG[7:0]							
		15:8	SPBRG[15:8]							
0x0F9C	RC1STA	7:0	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
0x0F9D	TX1STA	7:0	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
0x0F9E	BAUD1CON	7:0	ABDOVF	RCIDL		SCKP	BRG16		WUE	ABDEN

32. FVR——固定参考电压

固定参考电压（FVR）是独立于 V_{DD} 的稳定参考电压，可选的输出电压有：

- 1.024V
- 2.048V
- 4.096V

通过配置 FVR 的输出，可为以下各项提供参考电压：

- ADC 输入通道
- ADC 正参考电压
- 比较器输入
- 数模转换器（DAC）

FVR 可以通过将 FVRCON 寄存器的 FVREN 位置 1 来使能。

重要：固定参考电压输出不能超过 V_{DD}。

32.1. 独立的增益放大器

连接到 ADC、比较器和 DAC 的 FVR 输出会经过两个独立的可编程增益放大器。每个放大器都可以编程为 1x、2x 或 4x 增益，从而产生三种可能电压。

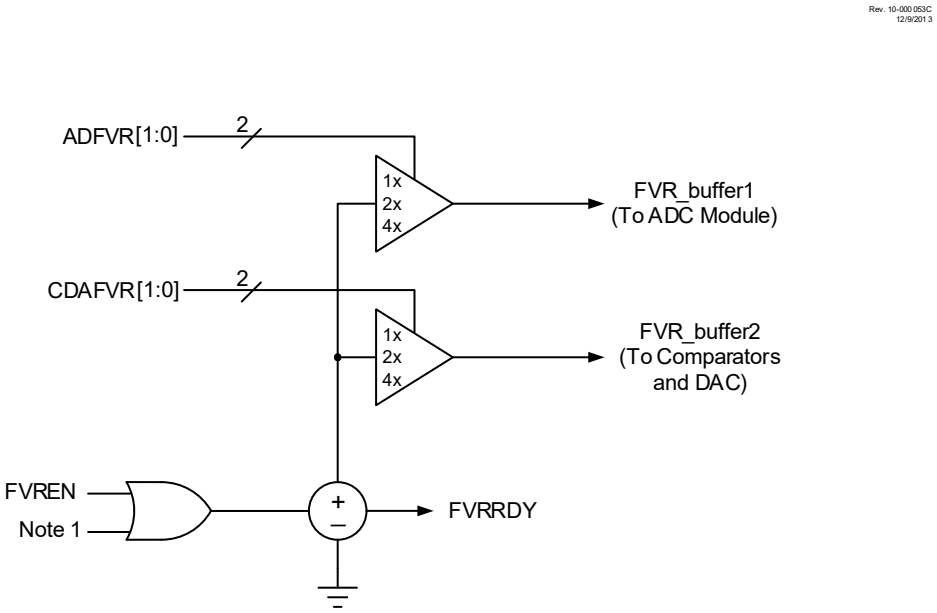
可使用 FVRCON 寄存器的 ADFVR[1:0]位为提供给 ADC 模块的参考电压使能和配置增益放大器设置。

可使用 FVRCON 寄存器的 CDAFVR[1:0]位为提供给 DAC 和比较器模块的参考电压使能和配置增益放大器设置。

32.2. FVR 稳定周期

当固定参考电压模块使能时，参考电压和放大器电路需要一段时间才能达到稳定。在电路稳定下来、可供使用时，FVRCON 寄存器的 FVRRDY 位将会置 1。

图 32-1. 参考电压框图



注：

1. 需要 FVR 的任意外设

32.3. 寄存器定义：FVR 控制

32.3.1. FVRCON

名称： FVRCON
偏移量： 0xF2C

固定参考电压控制寄存器

位	7	6	5	4	3	2	1	0
	FVREN	FVRRDY	TSEN	TSRNG	CDAFVR[1:0]		ADFVR[1:0]	
访问	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7 – FVREN 固定参考电压使能位

值	说明
1	使能固定参考电压
0	禁止固定参考电压

Bit 6 – FVRRDY 固定参考电压就绪标志位(3)

值	说明
1	固定参考电压输出就绪
0	固定参考电压输出未就绪或未使能

Bit 5 – TSEN 温度指示器使能位(2)

值	说明
1	使能温度指示器
0	禁止温度指示器

Bit 4 – TSRNG 温度指示器范围选择位(2)

值	说明
1	$V_{OUT} = V_{DD} - 4\ V_T$（高范围）
0	$V_{OUT} = V_{DD} - 2\ V_T$（低范围）

Bit 3:2 – CDAFVR[1:0] 比较器 FVR 缓冲器增益选择位

值	说明
11	比较器 FVR 缓冲器增益为 4x（4.096V）(1)
10	比较器 FVR 缓冲器增益为 2x（2.048V）(1)
01	比较器 FVR 缓冲器增益为 1x（1.024V）
00	比较器 FVR 缓冲器关闭

Bit 1:0 – ADFVR[1:0] ADC FVR 缓冲器增益选择位

值	说明
11	ADC FVR 缓冲器增益为 4x（4.096V）(1)
10	ADC FVR 缓冲器增益为 2x（2.048V）(1)
01	ADC FVR 缓冲器增益为 1x（1.024V）
00	ADC FVR 缓冲器关闭

注：

1. 固定参考电压输出不能超过 V_{DD}。
2. 更多信息，请参见**“温度指示器模块”**一章。
3. FVRRDY 始终为 1。

32.4. 寄存器汇总——FVR

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0F2B	保留									
0x0F2C	FVRCON	7:0	FVREN	FVRRDY	TSEN	TSRNG	CDAFVR[1:0]		ADFVR[1:0]	

33. 温度指示器模块

本器件系列装配有一个温度电路，该电路旨在测量硅芯片的工作温度。电路的工作温度范围介于-40°C 和 +85°C 之间。其输出是与器件温度成比例的电压。温度指示器的输出内部连接到器件 ADC。

该电路可用作温度阈值检测器或更精确的温度指示器，具体取决于执行的校准级别。单点校准使电路指示该点附近的温度。两点校准使电路能更精确地测量整个温度范围。

33.1. 电路工作原理

图 33-1 给出了温度检测电路的简化框图。与温度成比例的电压输出通过测量多个硅结的正向压降而得到。

以下公式说明了温度指示器的输出特性。

公式 33-1. V_{OUT} 范围

高电压范围： $V_{OUT} = V_{DD} - 4V_T$

低电压范围： $V_{OUT} = V_{DD} - 2V_T$

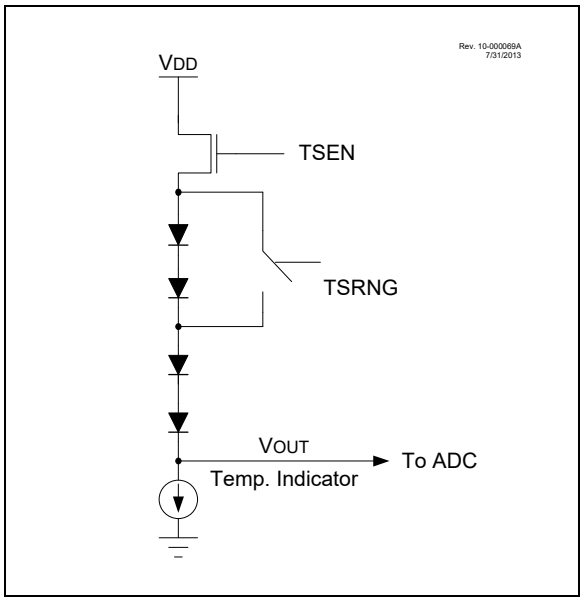
温度检测电路与固定参考电压（FVR）模块集成在一起。更多信息，请参见**“FVR——固定参考电压”**一章。

将 FVRCON 寄存器的 TSEN 位置 1 可使能该电路。禁止时，该电路不消耗电流。

该电路可在高电压范围或低电压范围下工作。高电压范围的选择方式是将 FVRCON 寄存器的 TSRNG 位置 1，从而可提供较宽的输出电压。这样可以在整个温度范围内提供更高的分辨率，但各器件之间的一致性较低。该电压范围需要较高的偏置电压才能工作，所以需要较高的 V_{DD}。

低电压范围的选择方式是将 FVRCON 寄存器的 TSRNG 位清零。低电压范围产生的压降较小，所以只需较低的偏置电压就可以让电路工作。低电压范围旨在用于进行低电压操作。

图 33-1. 温度检测电路图



33.2. 最小工作电压 V_{DD}

当温度电路工作于低电压范围时，器件可以在规范范围内的任意工作电压下工作。

当温度电路工作于高电压范围时，器件工作电压 V_{DD} 必须足够高，以确保正确地偏置温度电路。

表 33-1 给出了建议的最小 V_{DD} 与范围设置。

表 33-1. 建议的 V_{DD} 与范围

最小 V_{DD}，TSRNG = 1	最小 V_{DD}，TSRNG = 0
3.6V	1.8V

33.3. 温度输出

电路的输出使用内部模数转换器测量。保留一路通道用于温度电路输出。有关详细信息，请参见“**ADCC——带计算模块的模数转换器**”一章。

33.4. ADC 采集时间

为了确保精确的温度测量，用户必须在 ADC 输入多路开关连接到温度指示器输出之后至少等待 200 µs，然后再执行转换。此外，用户必须在温度指示器输出的连续两次转换之间等待 200 µs。

34. ADCC——带计算功能的模数转换器模块

带计算模块的模数转换器（ADCC）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，并将转换结果存储在 ADC 结果寄存器中。

此外，ADC 模块还提供了以下特性：

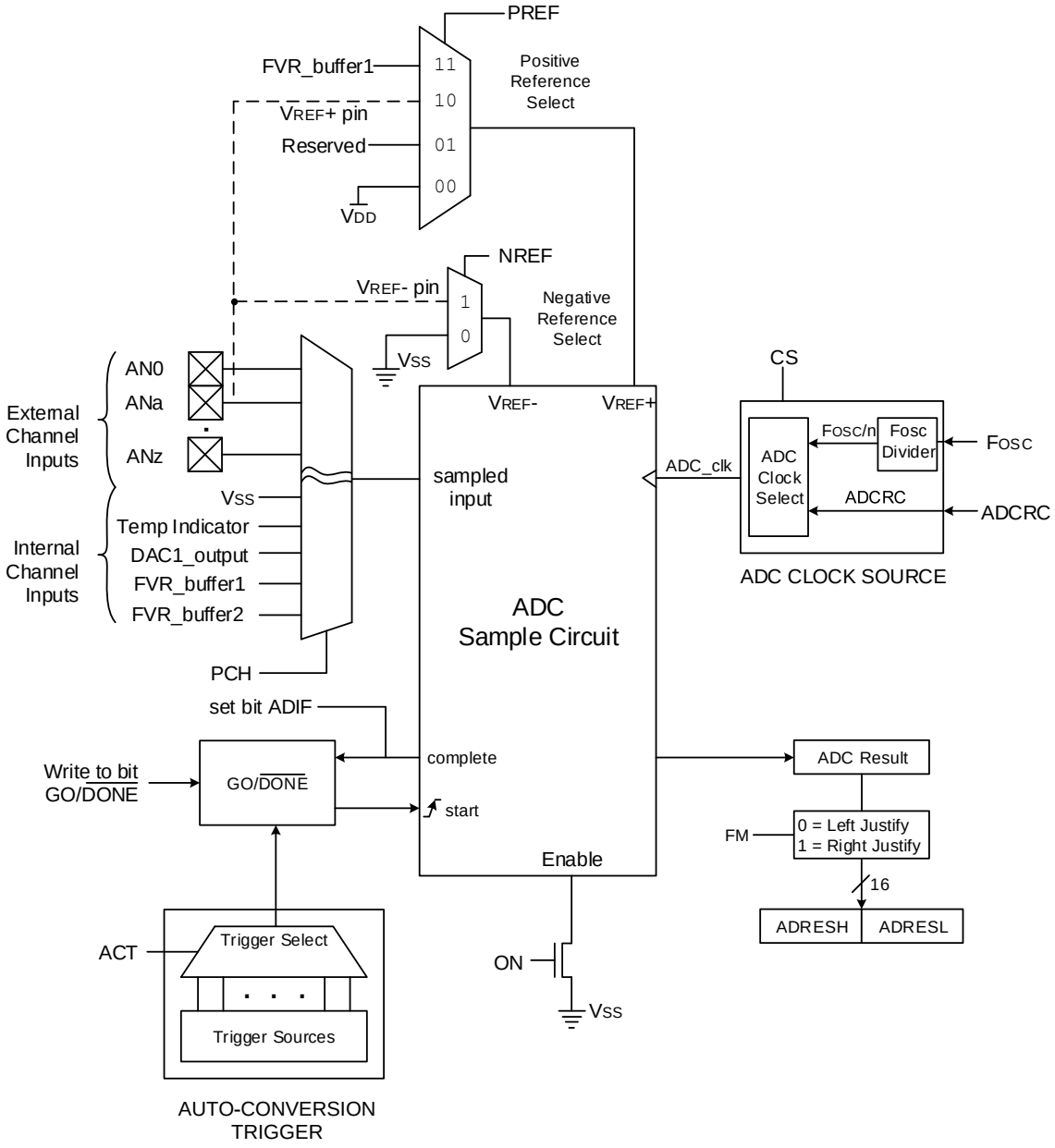
- 采集定时器
- 支持硬件电容分压器（CVD）：
  - 预充电定时器
  - 可调节采样保持电容阵列
  - 保护环数字输出驱动器
- 自动重复和排序：
  - CVD 的自动双采样转换
  - 两组结果寄存器（当前结果和前一个结果）
  - 自动转换触发器
  - 内部再触发器
- 计算特性：
  - 求平均值和低通滤波功能
  - 参考比较
  - 2 级阈值比较
  - 可选中断

图 34-1 给出了 ADC 的框图。

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 可在转换完成时和阈值比较时产生中断。这些中断可用于将器件从休眠模式唤醒。

图 34-1. ADCC 框图



34.1. ADC 配置

配置 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

- 转换触发选择
- ADC 采集时间
- ADC 预充电时间
- 附加采样保持电容
- 单/双采样转换
- 保护环输出

34.1.1. 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，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”** 一章。

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消耗的电流过大。

34.1.2. 通道选择

ADPCH 寄存器决定与采样保持电路相连接以进行转换的通道。切换通道时，建议在开始下一次转换前留出一些采集时间（ADACQ 寄存器）。更多信息，请参见“ADC 工作原理”一节。

重要： 为降低测量误差，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，则可以通过 DAC 输出通道来选择 V_{SS}。如果 DAC 当前正在使用，则可以将一个空闲的输入通道连接到 V_{SS} 来代替 DAC。

34.1.3. ADC 参考电压

PREF 位控制正参考电压。NREF 位控制负参考电压。有关可用正负参考电压源的列表，请参见 ADREF 寄存器。

34.1.4. 转换时钟

转换时钟源使用 CS 位进行选择。当 CS = 1 时，ADC 时钟源为内部固定频率时钟（称为 ADCRC）。当 CS = 0 时，ADC 时钟源自 F_{OSC}。

重要： 当 CS = 0 时，可使用 ADCLK 寄存器对时钟进行分频，以满足 ADC 时钟周期要求。

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 34-2。

为正确转换，必须满足相应的 T_{AD} 规范。有关详细信息，请参见“**电气规范**”一章中的“ADC 时序规范”表。下表给出了适当的 ADC 时钟选择的示例。

表 34-1. ADC 时钟周期（T_{AD}）与器件工作频率关系表(1,3)

ADC 时钟源	ADCLK	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	'b000000	62.5 ns(2)	100 ns(2)	125 ns(2)	250 ns(2)	500 ns	2.0 µs
F_{OSC}/4	'b000001	125 ns(2)	200 ns(2)	250 ns(2)	500 ns	1.0 µs	4.0 µs
F_{OSC}/6	'b000010	187.5 ns(2)	300 ns(2)	375 ns(2)	750 ns	1.5 µs	6.0 µs

表 34-1. ADC 时钟周期（T_{AD}）与器件工作频率关系表(1,3)（续）

ADC 时钟源	ADCLK	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}/8	'b000011	250 ns(2)	400 ns(2)	500 ns	1.0 µs	2.0 µs	8.0 µs
...	...	...	...	...	...	...	...
F_{OSC}/16	'b000111	500 ns	800 ns	1.0 µs	2.0 µs	4.0 µs	16.0 µs(2)
...	...	...	...	...	...	...	...
F_{OSC}/32	'b001111	1.0 µs	1.6 µs	2.0 µs	4.0 µs	8.0 µs	32.0 µs(2)
...	...	...	...	...	...	...	...
F_{OSC}/64	'b0111111	2.0 µs	3.2 µs	4.0 µs	8.0 µs	16.0 µs(2)	64.0 µs(2)
...	...	...	...	...	...	...	...
F_{OSC}/128	'b111111	4.0 µs	6.4 µs	8.0 µs	16.0 µs(2)	32.0 µs(2)	128.0 µs(2)
ADCRC	CS = 1	1.0 µs-6.0 µs	1.0 µs-6.0 µs	1.0 µs-6.0 µs	1.0 µs-6.0 µs	1.0 µs-6.0 µs	1.0 µs-6.0 µs

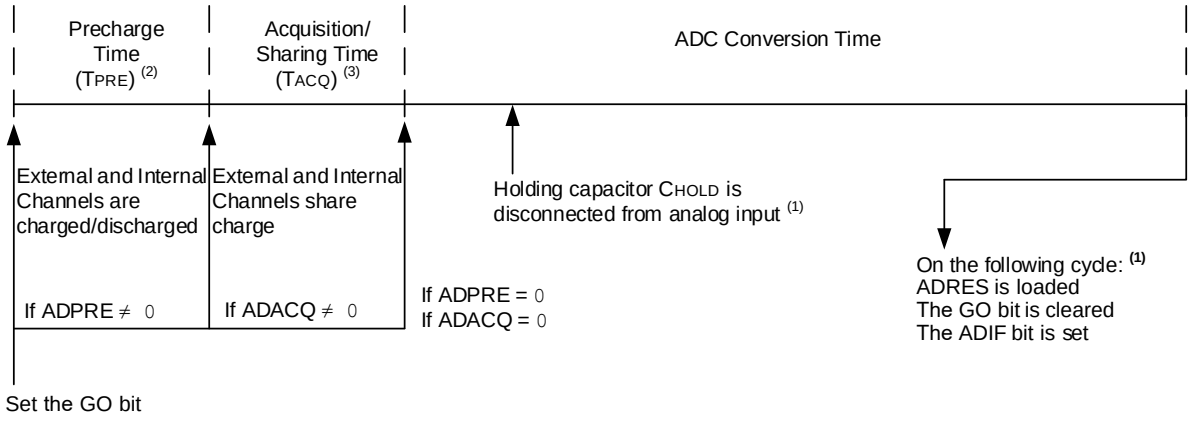
注：

1. 请参见器件数据手册的“**电气规范**”一章，查看 ADCRC 源的 T_{AD} 参数典型值。
2. 这些值违反了所需的 T_{AD} 时间。
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。

重要：

- 除 ADCRC 时钟源外，系统时钟频率的任何改变都会改变 ADC 时钟频率，这会影响 ADC 结果。
- ADC 的内部控制逻辑基于 CS 位选择的时钟运行。如果 CS 位置 1（ADC 基于 ADCRC 运行），则将 ADC 控制位置 1 时，可能会出现意外的工作延时。

图 34-2. 模数转换周期



Note:

1. Refer to the ADC Conversion Timing Specifications table in the **“Electrical Specifications”** chapter.
2. Refer to the ADPRE register for more details.
3. Refer to the ADACQ register for more details.

34.1.5. 中断

ADC 模块可在模数转换完成时产生中断。ADC 中断标志位是 PIRx 寄存器中的 ADIF 位。ADC 中断允许位是 PIEx 寄存器中的 ADIE 位。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 振荡器时，ADC 才能在休眠模式下工作。

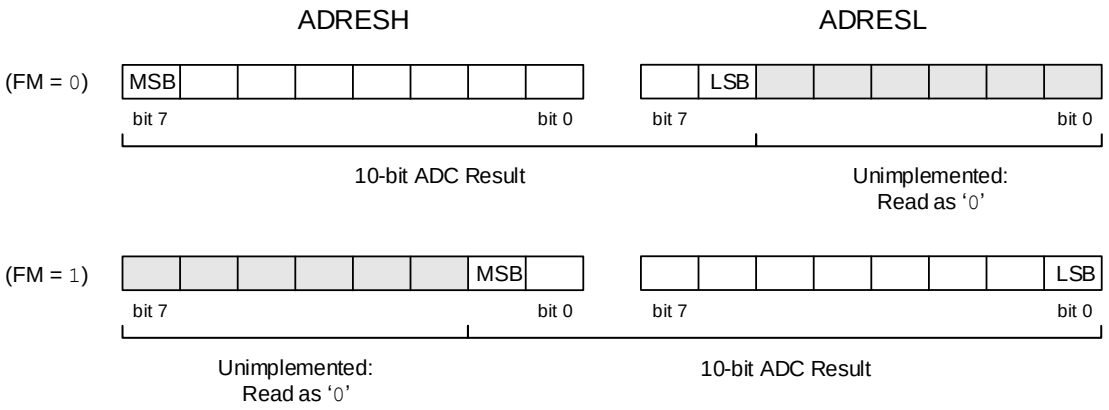
器件处于休眠模式时：

- 如果 ADIE = 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 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。

34.1.6. 结果格式

ADC 转换的结果可采用两种格式：左对齐或右对齐。FM 位控制输出格式，如图 34-3 所示。

图 34-3. 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。

34.2. ADC 工作原理

34.2.1. 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（通过 ADACT 选择触发源）
- 连续模式重新触发（有关详细信息，请参见连续采样模式一节）

重要：GO 位不会在启动 ADC 的同一指令中置 1。有关详细信息，请参见 ADC 转换步骤（基本模式）一节。

34.2.2. 转换完成

当单次转换完成时，ADRES 中的现有值将写入 ADPREV（如果 PSIS = 0），新转换结果出现在 ADRES 中。当转换完成时，ADC 模块会：

- 将 GO 位清零（除非 CONT 位置 1）
- 将 ADIF 中断标志位置 1
- 将 MATH 位置 1
- 更新 ADACC

每次转换（DSEN = 0 时）或者每两次转换后（DSEN = 1 时）后，将发生以下事件：

- 计算 ADERR
- 如果 ADERR 计算满足阈值比较，则 ADTIF 中断标志将置 1

34.2.3. 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 ADCRC 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，转换完成时器件将从休眠状态唤醒。如果禁止了 ADC 中断，尽管 ON 位仍保持置 1，但是在转换完成后，器件会保持休眠状态，并且 ADC 模块会关闭。

34.2.4. 休眠期间的外部触发信号

在休眠期间，如果在 ADC 时钟源设置为 ADCRC 时接收到外部触发信号，则 ADC 模块将执行转换并在完成后将 ADIF 位置 1。

如果在 ADC 时钟源不是 ADCRC 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。

34.2.5. 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 ACT 位进行选择。

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。

34.2.6. ADC 转换步骤（基本模式）

以下是用 ADC 执行模数转换的示例步骤：

1. 配置端口：
   - 禁止引脚输出驱动器（见 TRISx 寄存器）
   - 将引脚配置为模拟引脚（见 ANSELx 寄存器）
2. 配置 ADC 模块：
   - 选择 ADC 转换时钟
   - 配置参考电压
   - 选择 ADC 输入通道
   - 配置预充电（ADPRE）和采集（ADACQ）时间段
   - 开启 ADC 模块
3. 配置 ADC 中断（可选）：
   - 清零 ADC 中断标志
   - 允许 ADC 中断

- 允许全局中断（GIE 位）(1)

4. 如果 ADACQ = 0，软件必须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）时，器件将从休眠模式唤醒，但不会进入中断服务程序。
2. 有关详细信息，请参见 ADC 采集要求一节。

例 34-1. 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 ADCON1      ;
    clrf    ADCON1      ;
    clrf    ADCON2      ; Legacy mode, no filtering, ADRES->ADPREV
    clrf    ADCON3      ; no math functions
    clrf    ADREF       ; Vref = Vdd & Vss
    clrf    ADPCH       ; select RA0/AN0
    clrf    ADACQ       ; software controlled acquisition time
    clrf    ADCAP       ; default S&H capacitance
    clrf    ADRPT       ; no repeat measurements
    clrf    ADACT       ; auto-conversion disabled
    movlw   B'10010100' ; ADC On, right-justified, ADCRC clock
    movwf   ADCON0
    BANKSEL TRISA       ;
    bsf     TRISA,0     ; Set RA0 to input
    BANKSEL ANSEL       ;
    bsf     ANSEL,0     ; Set RA0 to analog
    call    SampleTime  ; Acquisiton delay
    BANKSEL ADCON0
    bsf     ADCON0,GO   ; Start conversion
    btfsc   ADCON0,GO   ; Is conversion done?
    goto    $-2         ; 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 34-2. 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();

        //Setup ADC
        ADCON0bits.FM = 1;      //right justify
        ADCON0bits.CS = 1;      //ADCRC Clock
        ADPCH = 0x00;           //RA0 is Analog channel
        TRISAbits.TRISA0 = 1;   //Set RA0 to input
```

```
        ANSELAbits.ANSELA0 = 1; //Set RA0 to analog
        ADACQ = 32;             //Set acquitisition time
        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
    }
}
```

34.3. 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 34-4 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 34-1 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 34-1. 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

$$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$$

$$T_{ACQ} = 2\mu s + T_C + \left[(\text{温度} - 25°C)\ (0.05\ \mu s/°C)\right]$$

T_C 值可以用以下公式近似计算：

$$V_{APPLIED}\left(1 - \frac{1}{\left(2^{n+1}\right) - 1}\right) = V_{CHOLD}$$：[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]响应 $V_{APPLIED}$ 充电到 V_{CHOLD}

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{APPLIED}\left(1 - \frac{1}{\left(2^{n+1}\right) - 1}\right)$$；合并[1]和[2]

注：其中，n 为 ADC 的分辨率（单位为位）

求解 T_C：

$$T_C = -C_{HOLD}(R_{IC} + R_{SS} + R_S)\ ln\ (1/2047)$$

$$T_C = -10pF(1k\Omega + 7k\Omega + 10k\Omega)\ ln\ (0.0004885)$$

$$T_C = 1.37\mu s$$

因此：

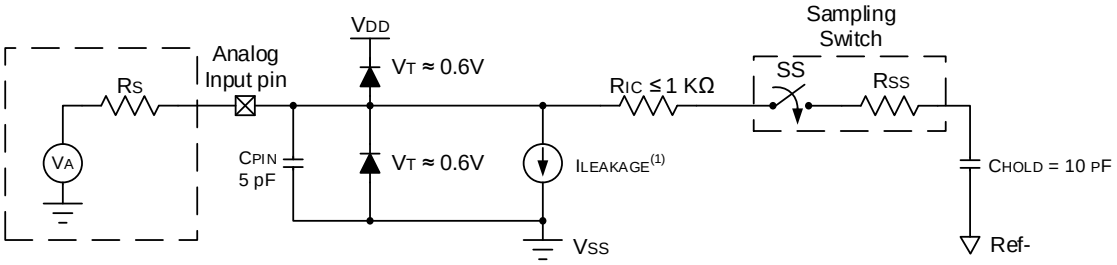
$$T_{ACQ} = 2\ \mu s + 1.37\ \mu s + \left[(50°C - 25°C)\ (0.05\ \mu s/°C)\right]$$

$$T_{ACQ} = 4.62\ \mu s$$

重要：

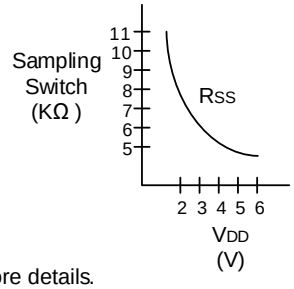
- 因为参考电压（V_{REF}）自行抵消，因此它对该公式没有影响。
- 充电保持电容（C_{HOLD}）在每次转换后不会放电。
- 模拟信号源的最大阻抗推荐值为 10 kΩ。此要求是为了符合引脚泄漏电流规范。

图 34-4. 模拟输入电路模型



Legend:

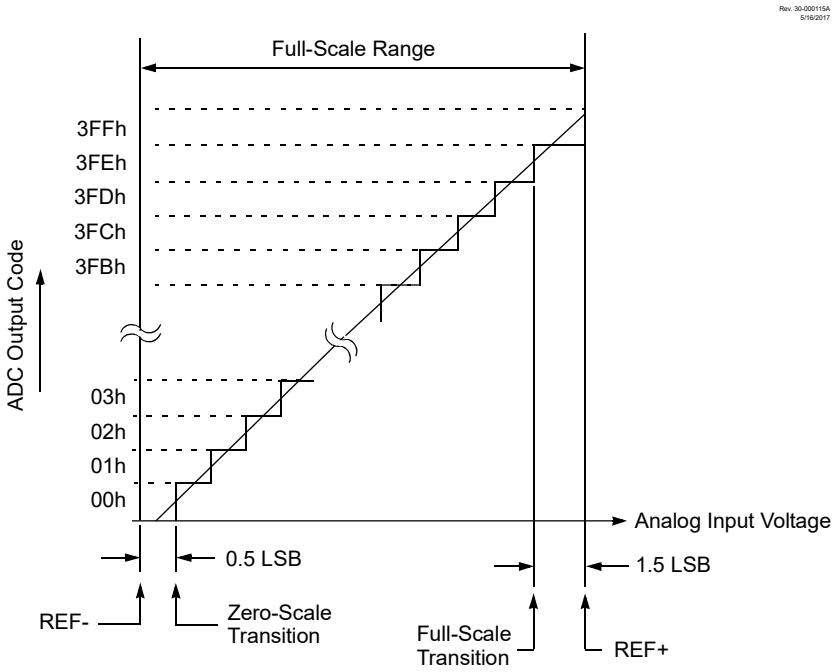
CPIN	=	Input Capacitance
ILEAKAGE	=	Leakage Current at the pin due to various junctions
RIC	=	Interconnect Resistance
RS	=	Source Impedance
VA	=	Analog Voltage
VT	=	Diode Forward Voltage
SS	=	Sampling Switch
RSS	=	Resistance of the Sampling Switch
CHOLD	=	Sample/Hold Capacitance



Note:

1. Refer to the *Electrical Specifications* section of the device data sheet for more details.

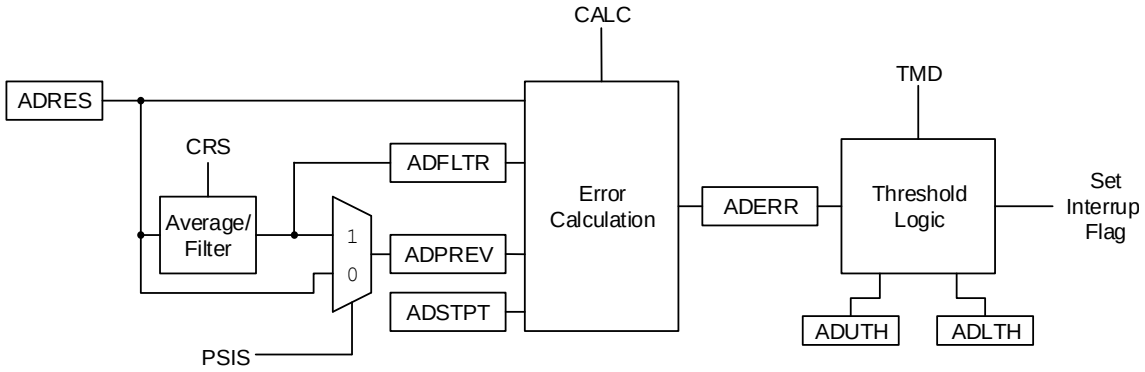
图 34-5. ADC 传递函数



34.4. 计算操作

ADC 模块硬件配有转换后计算功能。这些功能提供后处理功能，如数字滤波/平均和阈值比较。根据计算结果，该模块可以配置为获取额外的采样或停止转换，并可产生中断。

图 34-6. 计算功能简化框图



ADC 计算功能的具体操作由 MD 位控制。

该模块可以运行在以下五种模式下：

- 基本模式：这是传统模式。在该模式下，ADC 转换基于单（DSEN = 0）或双（DSEN = 1）采样进行。ADIF 在每次转换完成后置 1。ADTIF 根据计算模式置 1。
- 累加模式：每次触发时，ADC 转换结果都将加到累加器中，CNT 递增。ADIF 在每次转换后置 1。ADTIF 根据计算模式置 1。

- 平均模式：每次触发时，ADC 转换结果都将加到累加器中。当 RPT 采样数累加完成后，将执行阈值测试。下一次触发时，累加器将清零。对于后续测试，需要累加额外的 ADRPT 采样。
- 突发平均模式：触发时，累加器将清零。随后将重复收集 ADC 转换结果，直至完成 ADRPT 采样累加，最终会测试阈值。
- 低通滤波器（Low-Pass Filter，LPF）模式：每次触发时，ADC 转换结果都会通过滤波器发送。完成 ADRPT 采样后，将执行阈值测试。此后每次触发时，都将通过滤波器发送 ADC 转换结果并再次执行阈值测试。

下表总结了上述五种模式。

表 34-2. 计算模式

模式	MD	寄存器清零事件	周期(1)结束后的值		阈值操作			ADTIF 中断时的值		
		ADACC 和 CNT	ADACC	ADCNT	重新触发	阈值测试	中断	AOV	ADFLTR	ADCNT
基本	0	ACLR = 1	不变	不变	无	每次采样	如果阈值 = true	N/A	N/A	计数
累加	1	ACLR = 1	S1 + ADACC 或(S2 - S1) + ADACC	如果（ADCNT = 0xFF）：ADCNT，否则：ADCNT+1	无	每次采样	如果阈值 = true	ADACC 溢出	ADACC/2^{CRS}	计数
平均	2	ACLR = 1 或 ADCNT ≥ ADRPT（GO 置 1 或重新触发时）	S1 + ADACC 或(S2 -S1) + ADACC	如果（ADCNT = 0xFF）：ADCNT，否则： ADCNT + 1	无	如果 ADCNT ≥ ADRPT	如果阈值 = true	ADACC 溢出	ADACC/2^{CRS}	计数
突发平均	3	ACLR = 1 或在 GO 置 1 或重新触发时	每次重复：与最终平均值相同，包含所有样本的总和	每次重复：与最终平均值相同，ADCNT = ADRPT	ADCNT < ADRPT 时重复	如果 ADCNT ≥ ADRPT	如果阈值 = true	ADACC 溢出	ADACC/2^{CRS}	ADRPT
低通滤波器	4	ACLR = 1	S1 + ADACC-ADACC/2^{CRS} 或(S2 - S1) + ADACC-ADACC/2^{CRS}	如果（ADCNT = 0xFF）：ADCNT，否则：ADCNT + 1	无	如果 ADCNT ≥ ADRPT	如果阈值 = true	ADACC 溢出	ADACC/2^{CRS}（滤波值）	计数

注：

1. 如果 DSEN = 0，则一个周期表示一次转换。如果 DSEN = 1，则一个周期表示两次转换。
2. S1 和 S2 分别为采样 1 和采样 2 的缩写。当 DSEN = 0 时，S1 = ADRES；当 DSEN = 1 时，S1 = ADPREV 且 S2 = ADRES。

34.4.1. 数字滤波器/平均值计算

数字滤波器/平均值模块由具有数据反馈选项的累加器以及用于确定何时需要应用阈值测试的控制逻辑组成。累加器可通过 ADACC 寄存器访问。

每次发生触发事件（GO 位置 1 或外部事件触发）时，ADC 转换结果都将加到累加器中或从累加器中减去。如果累加值超出 $2^{(accumulator\_width)} - 1 = 2^{18} - 1 = 262143$，则 AOV 溢出位置 1。

要累加的采样数由 ADRPT（ADC 重复设置）寄存器决定。每次将采样加到累加器时，ADCNT 寄存器都会递增。ADRPT 采样累加完成（ADCNT = ADRPT）后，累加器会立即根据 ADC 工作模式自动清零。累加器清零命令可以通过将 ACLR 位置 1 在软件中发出。将 ACLR 位置 1 还将清零 AOV（累加器溢出）位以及 ADCNT 寄存器。累加器清零操作完成时，ACLR 位将由硬件清零。

重要：当 ADC 通过 ADCRC 工作时，需要最多五个 ADCRC 时钟周期来执行 ADACC 清零操作。

CRS 位用于控制累加器结果的数据移位，可以有效地对累加器寄存器中的值进行除法运算。对于数字滤波器的累加模式，移位提供了一种简单的换算方法。对于平均/突发平均模式，只有当采样数与移位位数一致时，计算出的平均值才是准确的。对于低通滤波器模式，移位是滤波器不可或缺的一部分，决定了滤波器的截止频率。 表 34-3 显示了以 ωT（弧度）为单位的-3 dB 截止频率以及奈奎斯特频率下（ωT = π）该滤波器的最大信号衰减。

表 34-3. 低通滤波器的-3 dB 截止频率

CRS	-3 dB 截止频率时的 ωT（弧度）	$F_{Nyquist}$ = 1/(2T)时的增益（dB）
1	0.72	-9.5
2	0.284	-16.9
3	0.134	-23.5
4	0.065	-29.8
5	0.032	-36.0
6	0.016	-42.0

34.4.2. 基本模式

基本模式（MD = 'b000）禁止所有附加计算功能。在该模式下，不发生累加，但执行阈值误差比较。双采样、连续模式和所有 CVD 功能仍然可用，但不会执行数字滤波器/平均值计算。

34.4.3. 累加模式

在累加模式（MD = 'b001）下，每次转换后，ADC 结果都会加到 ADACC 寄存器中。ADACC 寄存器按照 CRS 位的值右移。该右移值会复制到 ADFLTR 寄存器中。格式化模式不会影响 ADACC 或 ADFLTR 值的右对齐。每次采样时，ADCNT 会递增，以统计累加的采样数。每次采样和累加后，都会对 ADFLTR 值执行阈值比较（见阈值比较一节），并且可能触发 ADTIF 中断。

34.4.4. 平均模式

在平均模式（MD = 'b010）下，ADACC 寄存器随每次 ADC 采样累加，这与累加模式几乎相同，ADCNT 寄存器随每次采样递增。ADFLTR 寄存器也将用 ADACC 寄存器的右移值进行更新。CRS 位的值管理右移位数。但是，在平均模式下，当 ADCNT 大于或等于用户定义的 ADRPT 值时会执行阈值比较。在该模式下，当 ADRPT = 2^CRS 时，最终的累加值将除以采样数，以便对收集的所有采样的平均值执行阈值比较操作。

34.4.5. 突发平均模式

突发平均模式（MD = 'b011）的作用在大多数方面与平均模式相同。其中一个不同之处在于，即使未使能连续采样模式（见连续采样模式一节），该模式也会连续重新触发 ADC 采样，直到 CNT 值等于 ADRPT。这样可以对 ADC 采样的短突发平均值执行阈值比较。

34.4.6. 低通滤波器模式

低通滤波器模式（MD = 'b100）在采样处理方式上与平均模式相似（累加采样，直到 ADCNT 值大于或等于 RPT，然后触发阈值比较），但是该模式会对所有采样执行低通滤波操作而不是进行简单的平均值计算，从而可降低高频噪声对总值的影响，然后再对结果进行阈值比较。在该模式下，CRS 位决定了低通滤波器的截止频率（如数字滤波器/平均值计算一节所示）。有关数学运算的更多详细说明，请参见“计算操作”一节。

34.4.7. 阈值比较

在每次计算结束时：

- 转换结果在转换结束时捕捉。
- 误差（ADERR）基于 CALC 位选择的差值计算来计算。该值可以是以下计算的结果之一：
  - 单次测量结果的一阶导数
  - 使能双采样时的 CVD 结果
  - 当前结果与 ADSTPT 寄存器中的设定值比较
  - 当前结果与滤波后的结果/平均值计算结果比较
  - 滤波后的值/平均值的一阶导数
  - 滤波/平均值与 ADSTPT 寄存器中的设定值比较
- 计算结果（ADERR）将与上限阈值和下限阈值（ADUTH 和 ADLTH 和寄存器）进行比较，以设置 UTHR 和 LTHR 标志位。阈值逻辑通过 TMD 位选择。阈值触发选项可以是以下各项之一：
  - 永不中断
  - 误差小于下限阈值
  - 误差大于或等于下限阈值
  - 误差介于上限阈值和下限阈值之间（包含上限和下限）
  - 误差超出阈值
  - 误差小于或等于上限阈值
  - 误差大于上限阈值
  - 无论阈值测试结果如何，始终中断
  - 如果满足阈值条件，则阈值中断标志 ADTIF 置 1。

重要：

- 阈值测试为有符号运算。
- 如果 AOV 置 1，则发出阈值中断信号。阈值中断处理程序最好通过检查 AOV 位来验证阈值的有效性。

34.4.8. 重复和采样选项

34.4.8.1. 连续采样模式

CONT 位置 1 时，会在更新 ADACC 寄存器后自动重新触发新的转换周期。这意味着，GO 位保持置 1 以生成自动重新触发信号。如果 SOI = 1，阈值中断条件将清零 GO，转换将停止。

34.4.8.2. 双采样转换

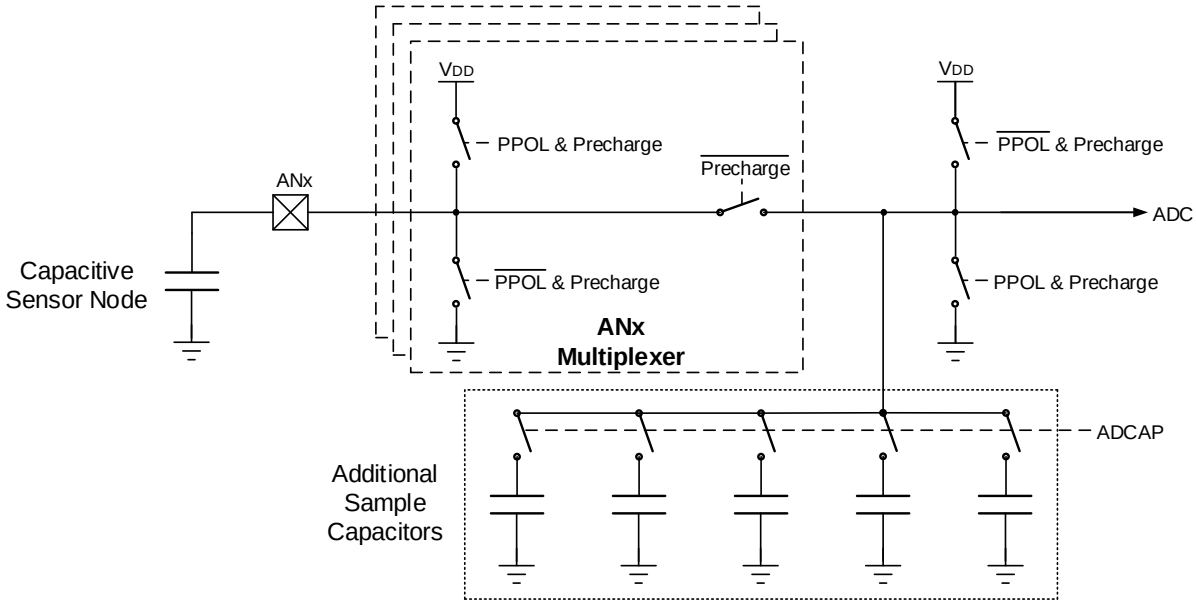
将 DSEN 位置 1 可使能双采样。该位置 1 时，在模块计算阈值误差之前需要进行两次转换。当 CONT = 0 时，每次转换必须单独触发；而当 CONT = 1 时，仅需单次触发即可自动重复执行转换。第一次转换将 MATH 位置 1 并更新 ADACC，但不会计算 ADERR 或触发 ADTIF。当第二次转换完成时，第一个值会传输

到 ADPREV（取决于 PSIS 的设置），而第二次转换的值会放置在 ADRES 中。只有在第二次转换完成时，才会计算 ADERR 和触发 ADTIF（取决于 CALC 的值）。

34.5. 电容分压器（CVD）特性

ADC 模块包含多项特性，允许用户以内部 ADC 采样保持电容作为参考，在任意 ADC 通道上执行相对电容测量。此相对电容测量可用于实现电容触摸或接近传感应用。下图给出了 ADC 模块的 CVD 部分的基本框图。

图 34-7. 硬件电容分压器框图



下面举例说明如何将 ADC 配置为 CVD 操作：

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. 配置预充电（ADPRE）和采集（ADACQ）时间段
   e. 选择预充电极性（PPOL）
   f. 使能双采样（DSEN）
   g. 开启 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. 允许全局中断（GIE 位）(1)
4. 通过将 GO 位置 1，启动双采样转换。

5. 通过以下任一方式等待 ADC 转换完成：
   a. 轮询 GO 位
   b. 等待 ADC 中断（如果允许了中断）
6. 第二次 ADC 转换取决于 CONT 的状态：
   a. 如果 CONT = 1，两次转换将从单次触发自动重复
   b. 如果 CONT = 0，每次转换必须单独触发
7. ADERR 寄存器包含 CVD 结果。
8. 清零 ADC 中断标志（如果允许了中断）。

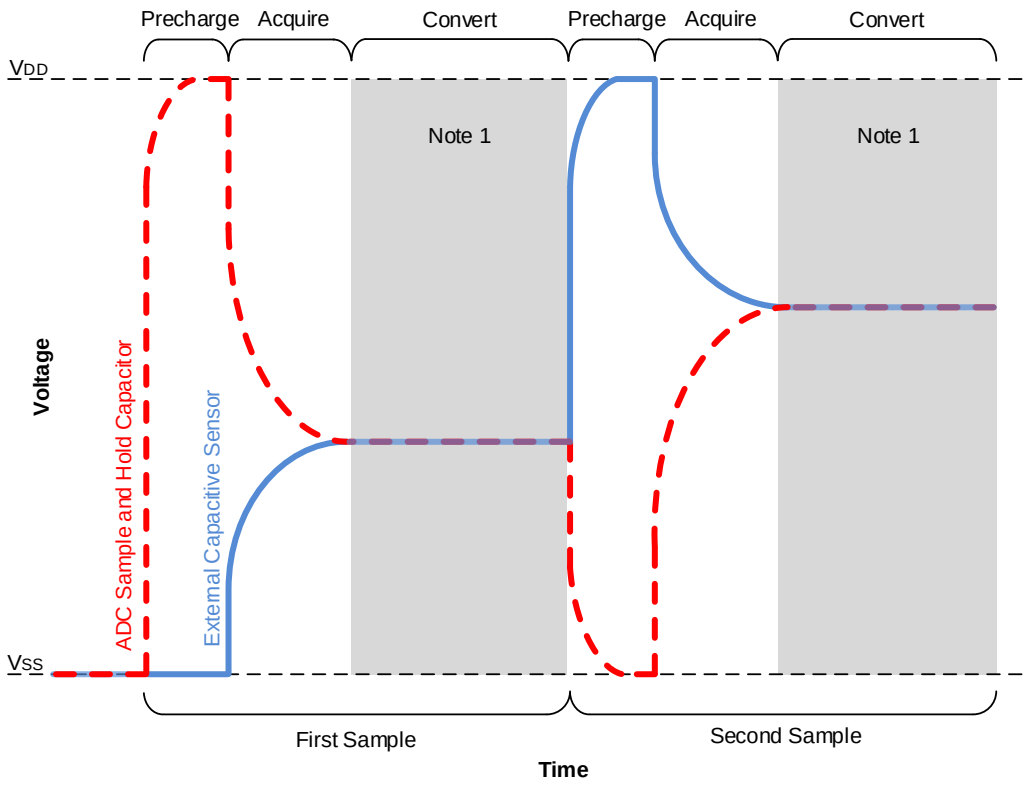
注：

1. 禁止全局中断（GIE = 0）时，器件将从休眠模式唤醒，但不会进入中断服务程序。

34.5.1. CVD 操作

在 CVD 操作中，首先是 ADC 的内部采样保持电容（C_{HOLD}）与将其连接到外部电容传感器节点的路径断开。断开后，C_{HOLD} 预充电至 V_{DD} 或放电至 V_{SS}。传感器节点将放电至 V_{SS} 或充电至 V_{DD}（总是与 C_{HOLD} 电压相反）。当预充电阶段完成时，两个节点的 V_{DD}/V_{SS} 偏置路径断开，C_{HOLD} 与外部传感器节点之间的路径重新连接，此时 CVD 操作的采集阶段开始。在采集期间，预充电的 C_{HOLD} 和传感器节点之间将形成电容分压器，从而可确定 C_{HOLD} 上的最终电压，该值取决于两个节点的电容和预充电电压。完成采集后，ADC 将转换 C_{HOLD} 上的电压。该过程随后会重复执行，但会针对 C_{HOLD} 和传感器节点翻转所选的预充电电压。下图给出了两次 CVD 测量（称为差分 CVD 测量）的波形。

图 34-8. 差分 CVD 测量波形



Note 1: **External Capacitive Sensor voltage during the conversion phase may vary as per the configuration of the corresponding pin.**

34.5.2. 预充电控制

预充电阶段是一个时间段，可将外部通道和内部采样保持电容置于已知电压。通过向 ADPRE 寄存器写入非零值来使能预充电。此阶段在 ADC 转换开始时通过将 GO 位置 1、特殊事件触发信号或由计算功能实现的转换重启来启动。如果 ADC 转换开始时 ADPRE 寄存器清零，则跳过此阶段。

在预充电期间，C_{HOLD} 与通向外部电容传感器的外部采样路径断开并连接到 V_{DD} 或 V_{SS}，具体取决于 PPOL 位的值。同时，所选模拟通道的端口引脚逻辑被改写以驱动数字高电平或低电平输出，以对 ADC 外部采样路径（包括外部传感器）进行预充电。此改写的输出极性由 PPOL 位决定，以确保外部传感器电容与内部 C_{HOLD} 电容的充电极性相反。该预充电时间由 ADPRE 寄存器控制。

重要：外部充电将改写相应 I/O 引脚的 TRIS/LAT/保护环输出设置。如果有一个器件连接到该引脚，则不会使用预充电。

34.5.3. CVD 的采集控制（ADPRE > 0）

采集阶段为内部采样保持电容的电压提供通过所选模拟通道充电或放电所需的时间。该采集时间由 ADACQ 寄存器控制。采集阶段在预充电阶段结束时开始。

采集阶段开始时，所选模拟通道的端口引脚逻辑被改写以关断数字高/低电平输出驱动器，从而使这些驱动器不影响电荷平均的最终结果。此外，所选的 ADC 通道将连接到 C_{HOLD}。这样便可在预充电通道和 C_{HOLD} 电容之间进行电荷平均。

重要：当 ADPRE > 0 时，将 ADACQ 设置为 0 将设置最大采集时间。禁止预充电时，将 ADACQ 设置为 0 将禁止硬件采集时间控制。

34.5.4. 保护环输出

图 34-9 显示了典型的保护环电路。C_{GUARD} 表示 PCB 上保护环走线的电容。用户选择 R_A 和 R_B 的值，使 C_{GUARD} 的电压曲线与所选采集通道匹配。

保护环的作用是产生一个与 CVD 传感信号同相的信号，以最大程度降低寄生电容对传感电极的影响。它还可用作互电容传感的互驱动器。

ADC 有两个保护环驱动输出 ADGRDA 和 ADGRDB。这些输出通过 PPS 控制连接至 I/O 引脚。更多详细信息，请参见 **“PPS——外设引脚选择模块”** 一章。这些输出的极性由 GPOL 和 IPEN 位控制。

在第一个预充电阶段开始时，两个输出都设置为匹配 GPOL 位。采集阶段开始后，ADGRDA 将改变极性，而 ADGRDB 则保持不变。当执行双采样转换时，将 IPEN 位置 1 会导致两个保护环输出在第二个预充电阶段开始时切换为 GPOL 的相反极性，ADGRDA 将再次翻转以进行第二次采集。有关保护环输出时序的更多信息，请参见图 34-10。

图 34-9. 保护环电路

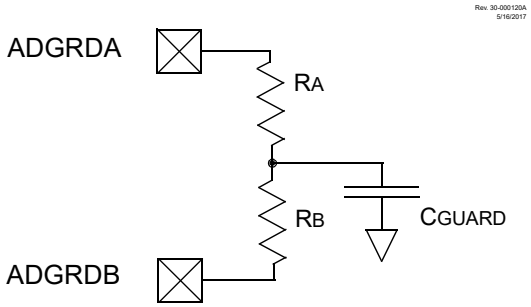
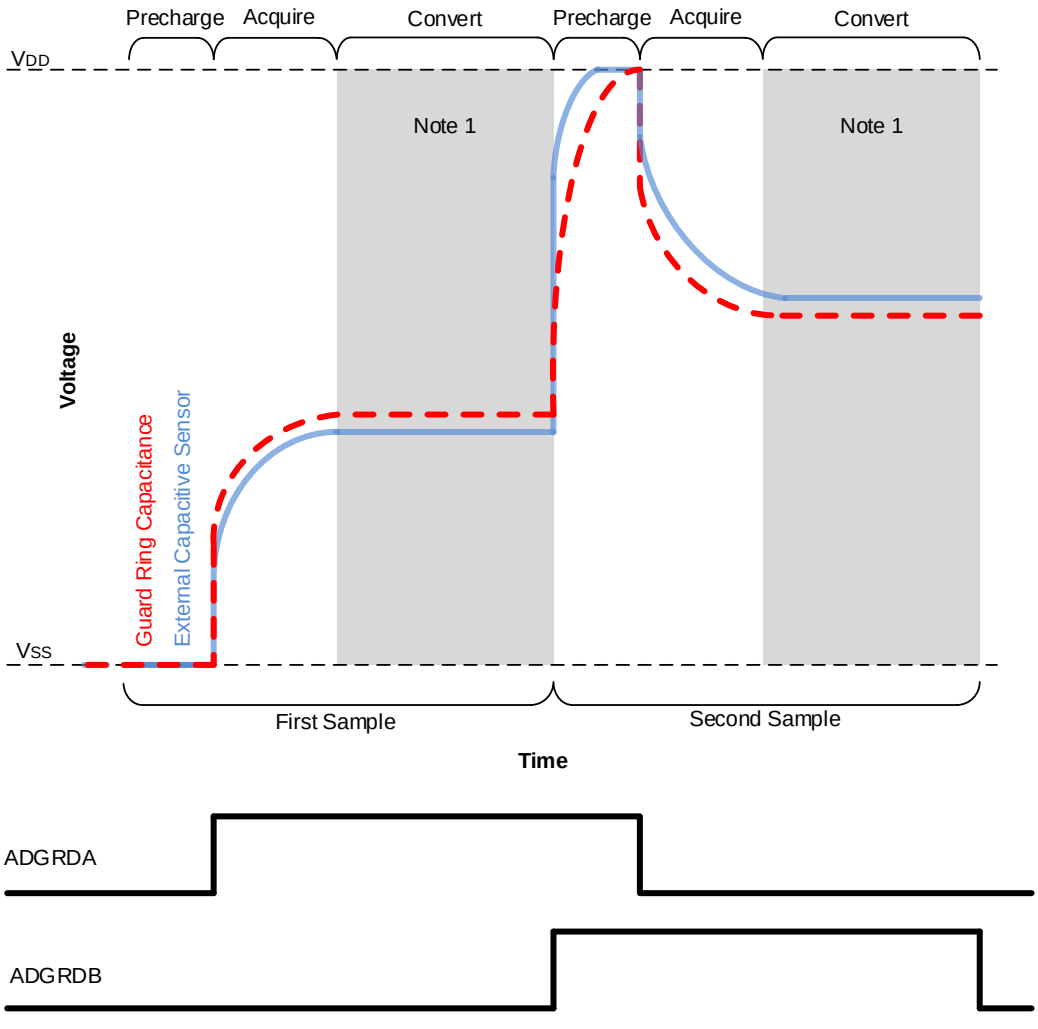


图 34-10. 带保护环的差分 CVD 输出波形



Note 1: **External Capacitive Sensor voltage during the conversion phase may vary as per the configuration of the corresponding pin.**

34.5.5. 附加采样保持电容

可通过 ADCAP 寄存器添加与内部采样保持电容（C_{HOLD}）并联的附加电容。该寄存器会选择一个可数字编程的电容，将其添加到 ADC 转换总线，从而增加 ADC 模块中采样保持电容的有效内部电容。这用于改善内部和外部电容的匹配度，从而实现更好的传感性能。由于转换期间未连接附加电容，因此不会影响 ADC 的模拟性能。

34.6. 寄存器定义：ADC 控制

下表列出了 ADC 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 34-4. ADC 长位名称前缀

外设	位名称前缀
ADC	AD

34.6.1. ADCON0

名称： ADCON0
偏移量： 0xF5B

ADC 控制寄存器 0

位	7	6	5	4	3	2	1	0
	ON	CONT		CS		FM		GO
访问	R/W	R/W		R/W		R/W		R/W/HC/HS
复位	0	0		0		0		0

Bit 7 – ON ADC 使能

值	说明
1	使能 ADC
0	禁止 ADC

Bit 6 – CONT ADC 连续操作使能

值	说明
1	每次转换触发完成时都会重新触发 GO，直到 ADTIF 置 1（如果 SOI 置 1）或 GO 清零（无论 SOI 的值如何）
0	每次转换触发完成时都会清零 ADC

Bit 4 – CS ADC 时钟选择

值	说明
1	由 ADCRC 专用振荡器提供时钟
0	由 F_{OSC} 提供时钟（根据 ADCLK 寄存器分频）

Bit 2 – FM ADC 结果格式/对齐选择

值	说明
1	ADRES 和 ADPREV 数据是右对齐的
0	ADRES 和 ADPREV 数据是左对齐的，并进行零填充

Bit 0 – GO ADC 转换状态[1,2]

值	说明
1	ADC 转换正在进行。将该位置 1 可启动 ADC 转换周期。该位由硬件清零（取决于 CONT 位）
0	ADC 转换已完成/不在进行

注：

1. 该位需要将 ON 位置 1。
2. 如果正在进行转换时使用软件清零，则在此之前的转换结果将传送到 ADRES 并且状态机将复位，但 ADIF 中断标志位不会置 1；滤波和阈值操作将不会执行。

34.6.2. ADCON1

名称： ADCON1
偏移量： 0xF54

ADC 控制寄存器 1

位	7	6	5	4	3	2	1	0
	PPOL	IPEN	GPOL					DSEN
访问	R/W	R/W	R/W					R/W
复位	0	0	0					0

Bit 7 – PPOL 预充电极性
第 1 个预充电阶段中的操作

值	条件	说明
x	ADPRE = 0	该位无影响
1	ADPRE > 0	外部模拟 I/O 引脚连接到 V_{DD}。 内部 AD 采样电容（C_{HOLD}）连接到 V_{SS}。
0	ADPRE > 0	外部模拟 I/O 引脚连接到 V_{SS}。 内部 AD 采样电容（C_{HOLD}）连接到 V_{DD}。

Bit 6 – IPEN A/D 反相预充电使能

值	条件	说明
x	DSEN = 0	该位无影响
1	DSEN = 1	第二个转换周期中的预充电和保护信号的极性与第一个周期中的相反
0	DSEN = 1	两个转换周期均使用 PPOL 和 GPOL 指定的预充电和保护信号

Bit 5 – GPOL 保护环极性选择

值	说明
1	在预充电阶段，ADC 保护环输出以数字高电平开始
0	在预充电阶段，ADC 保护环输出以数字低电平开始

Bit 0 – DSEN 双采样使能

值	说明
1	两次转换作为一组处理。每两次转换后执行一次选定的计算
0	每次转换后都执行一次选定的计算

34.6.3. ADCON2

名称： ADCON2
偏移量： 0xF55

ADC 控制寄存器 2

位	7	6	5	4	3	2	1	0
	PSIS	CRS[2:0]			ACLR	MD[2:0]		
访问	R/W	R/W	R/W	R/W	R/W/HC	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

Bit 7 – PSIS ADC 前次采样输入选择

值	说明
1	开始转换时从 ADFLTR 传输到 ADPREV
0	开始转换时从 ADRES 传输到 ADPREV

Bit 6:4 – CRS[2:0] ADC 累加计算右移选择

值	条件	说明
1 至 6	MD ='b100	低通滤波器时间常数为 2^{CRS}，滤波器增益为 1:1(2)
1 至 6	MD ='b011 至'b001	累加值按 CRS 右移（除以 2^{CRS}）(1,2)
x	MD ='b000	这些位被忽略

Bit 3 – ACLR A/D 累加器清零命令(3)

值	说明
1	将 ADACC 和 ADCNT 寄存器以及 AOV 位清零
0	清零操作已完成（或未开始）

Bit 2:0 – MD[2:0] ADC 工作模式选择(4)

值	说明
111-101	保留
100	低通滤波器模式
011	突发平均模式
010	平均模式
001	累加模式
000	基本（传统）模式

注：

1. 要正确计算平均值，采样数（在 ADRPT 中设置）必须是 2^{CRS}。
2. CRS = 'b111 和'b000 为保留选项。
3. 累加器操作完成时该位由硬件清零；延时可能为多条指令的时间，具体取决于振荡器选择。
4. 有关完整的模式说明，请参见计算操作一节。

34.6.4. ADCON3

名称： ADCON3
偏移量： 0xF56

ADC 控制寄存器 3

位	7	6	5	4	3	2	1	0
		CALC[2:0]			SOI	TMD[2:0]		
访问		R/W	R/W	R/W	R/W/HC	R/W	R/W	R/W
复位		0	0	0	0	0	0	0

Bit 6:4 – CALC[2:0] ADC 误差计算模式选择

表 34-5. ADC 误差计算模式

CALC	ADERR		应用
	DSEN = 0 单采样模式	DSEN = 1 CVD 双采样模式(1)	
111	保留	保留	保留
110	保留	保留	保留
101	ADFLTR - ADSTPT	ADFLTR - ADSTPT	平均值/滤波后的值与设定值之差
100	ADPREV - ADFLTR	ADPREV - ADFLTR	滤波后的值的一阶导数(3)（为负）
011	保留	保留	保留
010	ADRES - ADFLTR	(ADRES - ADPREV) - ADFLTR	实际结果与平均值/滤波后的值之差
001	ADRES - ADSTPT	(ADRES - ADPREV) - ADSTPT	实际结果与设定值之差
000	ADRES - ADPREV	ADRES - ADPREV	单次测量结果的一阶导数(2)
			实际 CVD 结果(2)

注：

1. 当 DSEN = 1 且 PSIS = 0 时，仅两次采样后计算一次 ADERR。
2. PSIS = 0 时。
3. PSIS = 1 时。

Bit 3 – SOI ADC 中断时停止

值	条件	说明
x	CONT = 0	不使用该位
1	CONT = 1	满足阈值条件时清零 GO，否则重新触发转换
0	CONT = 1	不通过硬件清零 GO，必须使用软件清零 GO 以停止重新触发

Bit 2:0 – TMD[2:0] 阈值中断模式选择

值	说明
111	无论阈值测试结果如何都中断
110	ADERR > ADUTH 时中断
101	ADERR ≤ ADUTH 时中断
100	ADERR < ADLTH 或 ADERR > ADUTH 时中断
011	ADERR > ADLTH 且 ADERR < ADUTH 时中断
010	ADERR ≥ ADLTH 时中断
001	ADERR < ADLTH 时中断
000	永不中断

34.6.5. ADSTAT

名称： ADSTAT
偏移量： 0xF60

ADC 状态寄存器

位	7	6	5	4	3	2	1	0
	AOV	UTHR	LTHR	MATH		STAT[2:0]		
访问	R/C/HS/HC	R	R	R/C/HS		R	R	R
复位	0	0	0	0		0	0	0

Bit 7 – AOV ADC 累加器溢出

值	说明
1	ADACC、ADFLTR 或 ADERR 寄存器已溢出
0	ADACC、ADFLTR 和 ADERR 寄存器均未溢出

Bit 6 – UTHR ADC 模块大于上限阈值标志

值	说明
1	ADERR > ADUTH
0	ADERR ≤ ADUTH

Bit 5 – LTHR ADC 模块小于下限阈值标志

值	说明
1	ADERR < ADLTH
0	ADERR ≥ ADLTH

Bit 4 – MATH ADC 模块计算状态
ADC 模块计算状态(1)

值	说明
1	ADACC、ADFLTR、ADUTH 和 ADLTH 寄存器以及 AOV 位正在更新或已更新
0	自该位上次清零以来，相关寄存器/位没有发生变化

Bit 2:0 – STAT[2:0] ADC 模块周期多阶段状态

值	说明
111	ADC 模块处于第二个转换阶段
110	ADC 模块处于第二个采集阶段
101	ADC 模块处于第二个预充电阶段
100	ADC 计算在第 1 次采样与第 2 次采样之间暂停；计算结果不完整，正在等待第 2 次采样的数据(2,3)
011	ADC 模块处于第一个转换阶段
010	ADC 模块处于第一个采集阶段
001	ADC 模块处于第一个预充电阶段
000	ADC 模块未在转换

注：

1. 当 STAT = 'b100 时，MATH 位不能通过软件清零。
2. 如果 ADC 时钟源为 ADCRC 且 F_{OSC} < ADCRC，则指示的状态可能无效。
3. 当 DSEN = 1 且 CONT = 0 时，STAT = 'b100 出现在两次触发之间。

34.6.6. ADCLK

名称： ADCLK
偏移量： 0xF52

ADC 时钟分频比寄存器

位	7	6	5	4	3	2	1	0
			CS[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 – CS[5:0] ADC 时钟分频比选择

值	说明
n	ADC 时钟频率 = F_{OSC}/(2*(n+1))

注：仅当选择 F_{OSC} 作为 ADC 时钟源（CS = 0）时，ADC 时钟分频比才可用。

34.6.7. ADREF

名称： ADREF
偏移量： 0xF53

ADC 参考电压选择寄存器

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

Bit 4 – NREF ADC 负参考电压选择

值	说明
1	V_{REF}-连接到外部 V_{REF}-
0	V_{REF}-连接到 AV_{SS}

Bit 1:0 – PREF[1:0] ADC 正参考电压选择

值	说明
11	V_{REF}+连接到内部固定参考电压（FVR）模块
10	V_{REF}+连接到外部 V_{REF}+
01	保留
00	V_{REF}+连接到 V_{DD}

34.6.8. ADPCH

名称： ADPCH
偏移量： 0xF5A

ADC 正通道选择寄存器

位	7	6	5	4	3	2	1	0
			PCH[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 – PCH[5:0] ADC 正输入通道选择

表 34-6. ADC 正输入通道选择

值	说明
111111	固定参考电压（FVR）(1)
111110	DAC1_OUT
111101	温度指示器(2)
111100	V_{SS}（模拟地）
111011-011000	保留。不连接任何通道。
010111	RC7/ANC7
010110	RC6/ANC6
010101	RC5/ANC5
010100	RC4/ANC4
010011	RC3/ANC3
010010	RC2/ANC2
010001	RC1/ANC1
010000	RC0/ANC0
001111	RB7/ANB7
001110	RB6/ANB6
001101	RB5/ANB5
001100	RB4/ANB4
001011	RB3/ANB3
001010	RB2/ANB2
001001	RB1/ANB1
001000	RB0/ANB0
000111	RA7/ANA7
000110	RA6/ANA6
000101	RA5/ANA5
000100	RA4/ANA4
000011	RA3/ANA3
000010	RA2/ANA2
000001	RA1/ANA1
000000	RA0/ANA0

注：

1. 有关详细信息，请参见**“固定参考电压模块”**一章。
2. 有关详细信息，请参见**“温度指示器模块”**一章。

34.6.9. ADPRE

名称： ADPRE
偏移量： 0xF59

ADC 预充电时间控制寄存器

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

位	7	6	5	4	3	2	1	0
	PRE[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 12:0 – PRE[12:0] 预充电时间选择

表 34-7. 预充电时间

值	说明	
	CS = 0	CS = 1
1 1111 1111 1111	8191 个 F_{OSC} 时钟	8191 个 ADCRC 时钟
1 1111 1111 1110	8190 个 F_{OSC} 时钟	8190 个 ADCRC 时钟
1 1111 1111 1101	8189 个 F_{OSC} 时钟	8189 个 ADCRC 时钟
...	...	...
0 0000 0000 0010	2 个 F_{OSC} 时钟	2 个 ADCRC 时钟
0 0000 0000 0001	1 个 F_{OSC} 时钟	1 个 ADCRC 时钟
0 0000 0000 0000	不包含在数据转换周期中	

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. ADPREH：访问高字节 ADPRE[12:8]。
2. ADPREL：访问低字节 ADPRE[7:0]。

34.6.10. ADACQ

名称： ADACQ
偏移量： 0xF57

ADC 采集时间控制寄存器

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

位	7	6	5	4	3	2	1	0
	ACQ[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 12:0 – ACQ[12:0] 采集（电荷共享时间）选择

表 34-8. 采集时间

值	说明	
	CS = 0	CS = 1
1 1111 1111 1111	8191 个 F_{OSC} 时钟	8191 个 ADCRC 时钟
1 1111 1111 1110	8190 个 F_{OSC} 时钟	8190 个 ADCRC 时钟
1 1111 1111 1101	8189 个 F_{OSC} 时钟	8189 个 ADCRC 时钟
...	...	...
0 0000 0000 0010	2 个 F_{OSC} 时钟	2 个 ADCRC 时钟
0 0000 0000 0001	1 个 F_{OSC} 时钟	1 个 ADCRC 时钟
0 0000 0000 0000	不包含在数据转换周期中(1)	

注：
1. 如果 ADPRE 不等于 0，则 ACQ = 0 表示采集时间为 8192 个 F_{OSC} 或 ADCRC 时钟。

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：

- ADACQH：访问高字节 ADACQ[12:8]
- ADACQL：访问低字节 ADACQ[7:0]

34.6.11. ADCAP

名称： ADCAP
偏移量： 0xF58

ADC 附加采样电容选择寄存器

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

Bit 4:0 – CAP[4:0] ADC 附加采样电容选择

值	说明
1 至 31	附加电容的 pF 数
0	无附加电容

34.6.12. ADRPT

名称： ADRPT
偏移量： 0xF61

ADC 重复设置寄存器

位	7	6	5	4	3	2	1	0
	RPT[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 – RPT[7:0] ADC 重复阈值
确定触发 ADC 进行阈值检查的次数。当 CNT 达到该值时，对误差阈值进行检查。计算模式为低通滤波器模式、突发平均模式或平均模式时使用。有关详细信息，请参见计算操作一节。

34.6.13. ADCNT

名称： ADCNT
偏移量： 0xF62

ADC 重复计数器寄存器

位	7	6	5	4	3	2	1	0
	CNT[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 – CNT[7:0] ADC 重复计数
计数检查阈值之前触发 ADC 的次数。当该值达到 RPT 时，检查阈值。计算模式为低通滤波器模式、突发平均模式或平均模式时使用。更多详细信息，请参见计算操作一节。

34.6.14. ADFLTR

名称： ADFLTR
偏移量： 0xF6D

ADC 滤波器寄存器

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

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

Bit 15:0 – FLTR[15:0] ADC 滤波器输出——有符号二进制补码
在累加模式、平均模式和突发平均模式下，等于按 CRS 位右移的 ACC。在 LPF 模式下，则为低通滤波器的输出。

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：

- ADFLTRH：访问高字节 ADFLTR[15:8]
- ADFLTRL：访问低字节 ADFLTR[7:0]

34.6.15. ADRES

名称： ADRES
偏移量： 0xF5E

ADC 结果寄存器

位	15	14	13	12	11	10	9	8
	RES[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
	RES[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 – RES[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]

34.6.16. ADPREV

名称： ADPREV
偏移量： 0xF5C

ADC 前一个结果寄存器

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

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

Bit 15:0 – PREV[15:0] 前一个 ADC 结果

值	条件	说明
n	PSIS = 1	n = 当前的 ADC 转换开始时的 ADFLTR 值
n	PSIS = 0	n = 当前的 ADC 转换开始时的 ADRES(1)

注：

1. 如果 PSIS = 0，则 ADPREV 的格式与 ADRES 相同，具体取决于 FM 位。
2. 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：
   - ADPREVH：访问 ADPREV[15:8]
   - ADPREVL：访问 ADPREV[7:0]

34.6.17. ADACC

名称： ADACC
偏移量： 0xF6B

ADC 累加器寄存器(1)

更多详细信息，请参见计算操作一节。

重要：该寄存器包含有符号二进制补码累加器值，未使用的高位包含符号位的副本。

位	23	22	21	20	19	18	17	16
							ACC[17:16]	
访问							R/W	R/W
复位							x	x

位	15	14	13	12	11	10	9	8
	ACC[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
	ACC[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 17:0 – ACC[17:0] ADC 累加器——有符号二进制补码

注：

1. 仅当 GO = 0 时，才能写入该寄存器。
2. 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：
   - ADACCU：访问最高字节 ADACC[17:16]
   - ADACCH：访问高字节 ADACC[15:8]
   - ADACCL：访问低字节 ADACC[7:0]

34.6.18. ADSTPT

名称： ADSTPT
偏移量： 0xF63

ADC 阈值设定值寄存器
可使用该寄存器根据 CALC 确定 ADERR。

位	15	14	13	12	11	10	9	8
	STPT[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
	STPT[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 – STPT[15:0] ADC 阈值设定值——有符号二进制补码

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：

- ADSTPTH：访问高字节 ADSTPT[15:8]
- ADSTPTH：访问低字节 ADSTPT[7:0]

34.6.19. ADERR

名称： ADERR
偏移量： 0xF69

ADC 设定值误差寄存器
ADC 设定值误差计算结果由 CALC 位决定。

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

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

Bit 15:0 – ERR[15:0] ADC 设定值误差——有符号二进制补码

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：

- ADERRH：访问高字节 ADERR[15:8]
- ADERRL：访问低字节 ADERR[7:0]

34.6.20. ADLTH

名称： ADLTH
偏移量： 0xF65

ADC 下限阈值寄存器
将 ADLTH 和 ADUTH 与 ADERR 进行比较以设置 UTHR 和 LTHR 位。该比较的结果可能会触发中断，具体取决于 TMD 的设置。

位	15	14	13	12	11	10	9	8
	LTH[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
	LTH[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 – LTH[15:0] ADC 下限阈值——有符号二进制补码

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：

- ADLTHH：访问高字节 ADLTH[15:8]
- ADLTHL：访问低字节 ADLTH[7:0]

34.6.21. ADUTH

名称： ADUTH
偏移量： 0xF67

ADC 上限阈值寄存器
将 ADLTH 和 ADUTH 与 ADERR 进行比较以设置 UTHR 和 LTHR 位。该比较的结果可能会触发中断，具体取决于 TMD 的设置。

位	15	14	13	12	11	10	9	8
	UTH[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
	UTH[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 – UTH[15:0] ADC 上限阈值——有符号二进制补码

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：

- ADUTHH：访问高字节 ADUTH[15:8]
- ADUTHL：访问低字节 ADUTH[7:0]

34.6.22. ADACT

名称： ADACT
偏移量： 0xF51

ADC 自动转换触发源选择寄存器

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

Bit 4:0 – ACT[4:0] 自动转换触发源选择

表 34-9. ADC 自动转换触发源

值	说明
11111	软件写入 ADPCH
11110	**保留**
11101	软件读取/写入 ADRESH
11100	软件读取/写入 ADERRH
11011 至 11000	**保留**
10111	CLC8_out(1)
10110	CLC7_out(1)
10101	CLC6_out(1)
10100	CLC5_out(1)
10011	CLC4_out(1)
10010	CLC3_out(1)
10001	CLC2_out(1)
10000	CLC1_out(1)
01111	电平变化中断
01110	C2_out
01101	C1_out
01100	PWM4_out
01011	PWM3_out
01010	CCP2_trigger
01001	CCP1_trigger
01000	TMR6_postscaled
00111	TMR5_overflow
00110	TMR4_postscaled
00101	TMR3_overflow
00100	TMR2_postscaled
00011	TMR1_overflow
00010	TMR0_overflow
00001	通过 ADACTPPS 选择的引脚
00000	禁止外部触发

注：
1. CN2510 器件上未提供。

34.7. 寄存器汇总——ADC

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0F50	保留									
0x0F51	ADACT	7:0				ACT[4:0]				
0x0F52	ADCLK	7:0			CS[5:0]					
0x0F53	ADREF	7:0				NREF			PREF[1:0]	
0x0F54	ADCON1	7:0	PPOL	IPEN	GPOL					DSEN
0x0F55	ADCON2	7:0	PSIS		CRS[2:0]		ACLR		MD[2:0]	
0x0F56	ADCON3	7:0			CALC[2:0]		SOI		TMD[2:0]	
0x0F57	ADACQ	7:0	ACQ[7:0]							
		15:8				ACQ[12:8]				
0x0F58	ADCAP	7:0				CAP[4:0]				
0x0F59	ADPRE	7:0	PRE[7:0]							
		15:8				PRE[12:8]				
0x0F5A	ADPCH	7:0			PCH[5:0]					
0x0F5B	ADCON0	7:0	ON	CONT		CS		FM		GO
0x0F5C	ADPREV	7:0	PREV[7:0]							
		15:8	PREV[15:8]							
0x0F5E	ADRES	7:0	RES[7:0]							
		15:8	RES[15:8]							
0x0F60	ADSTAT	7:0	AOV	UTHR	LTHR	MATH		STAT[2:0]		
0x0F61	ADRPT	7:0	RPT[7:0]							
0x0F62	ADCNT	7:0	CNT[7:0]							
0x0F63	ADSTPT	7:0	STPT[7:0]							
		15:8	STPT[15:8]							
0x0F65	ADLTH	7:0	LTH[7:0]							
		15:8	LTH[15:8]							
0x0F67	ADUTH	7:0	UTH[7:0]							
		15:8	UTH[15:8]							
0x0F69	ADERR	7:0	ERR[7:0]							
		15:8	ERR[15:8]							
0x0F6B	ADACC	7:0	ACC[7:0]							
		15:8	ACC[15:8]							
		23:16							ACC[17:16]	
0x0F6D	ADFLTR	7:0	FLTR[7:0]							
		15:8	FLTR[15:8]							

35. DAC——5 位数模转换器

数模转换器提供了一个可变参考电压，它与输入源成比例，具有 32 个可选输出电压。

DAC 的正输入源（$V_{SOURCE}+$）可以连接到：

- FVR 缓冲器
- 外部 $V_{REF}+$引脚
- V_{DD} 供电电压

DAC 的负输入源（$V_{SOURCE}-$）可以连接到：

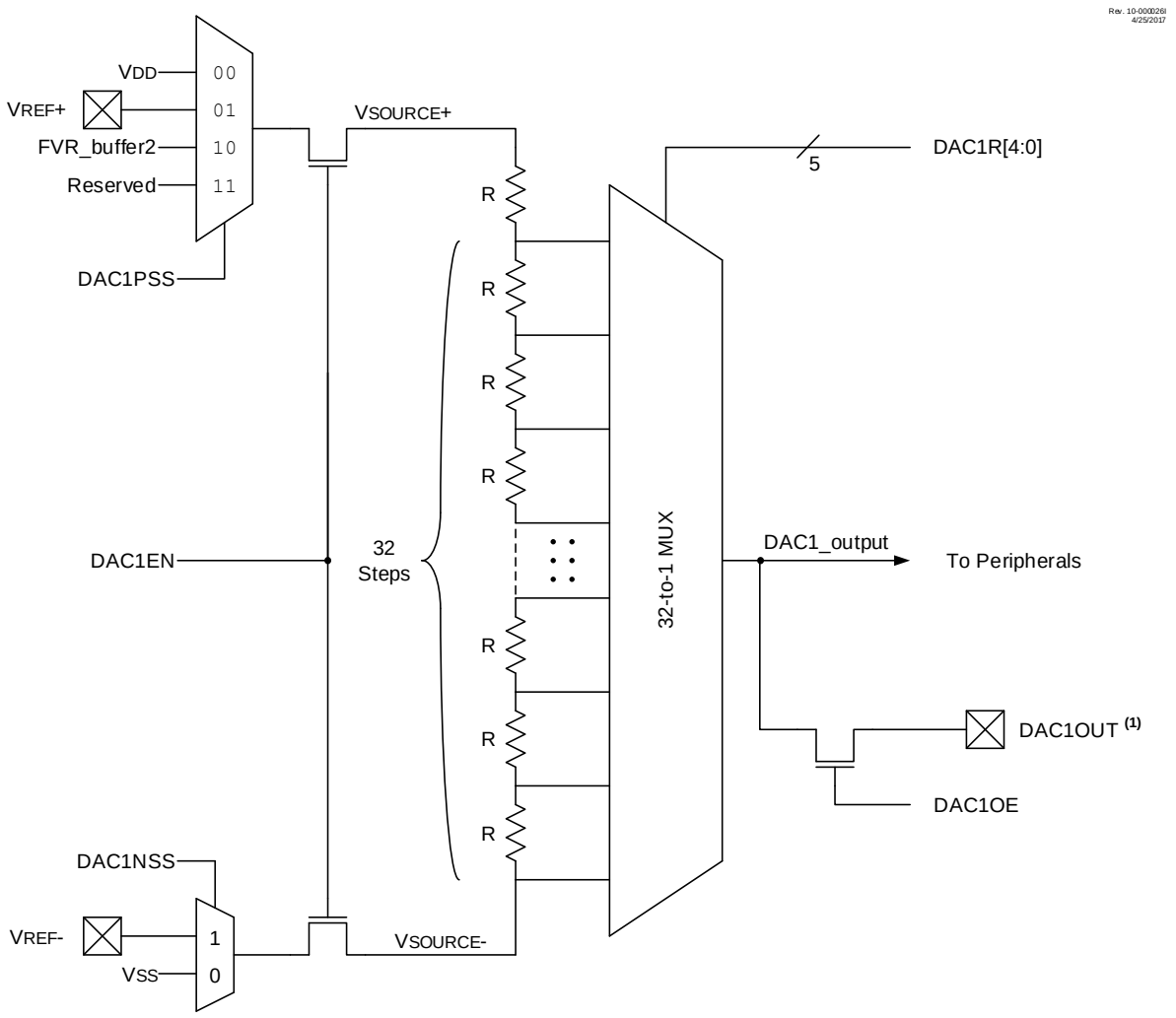
- 外部 $V_{REF}-$引脚
- V_{SS}

DAC 的输出（DACx_output）可选作以下各项的参考电压：

- 比较器同相输入
- ADC 输入通道
- DACxOUT1 引脚
- DACxOUT2 引脚

可通过将 EN 位置 1 来使能数模转换器（DAC）。

图 35-1. 数模转换器框图



35.1. 输出电压选择

DAC 有 32 种输出电压。这 32 种电压通过 DAC1R 位进行设置。

DAC 输出电压可由以下公式确定。

公式 35-1. DAC 输出电压

$$DACx\_output = \left(\left(V_{SOURCE+} - V_{SOURCE-}\right) x \frac{DACR[4:0]}{32}\right) + V_{SOURCE-}$$

注：有关可用 V_{SOURCE}+和 V_{SOURCE}-选择的信息，请参见 DAC1CON0 寄存器。

35.2. 比例输出电压

DAC 输出值通过使用一个梯形电阻网络产生，梯形电阻网络的每一端分别与正参考电压和负参考电压输入源连接。如果任一输入源的电压发生波动，DAC 输出值中会产生类似的波动。

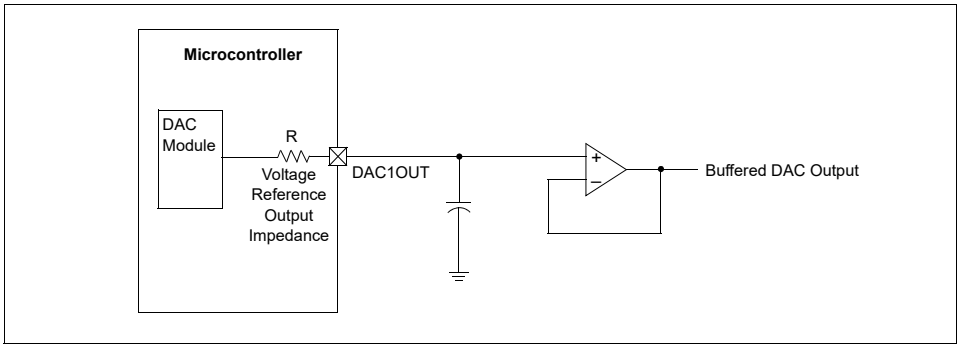
有关梯形电阻网络中各个电阻的值，请参见**“电气规范”**一章中的*“5 位 DAC 规范”*表。

35.3. DAC 参考电压输出

通过将 DAC1CON0 寄存器的 DAC1OE 位置 1，可以将 DAC 电压输出到 DAC1OUT 引脚。选择将 DAC 参考电压输出到 DAC1OUT 引脚会自动改写该引脚的数字输出缓冲器和数字输入阈值检测器功能，并禁止该引脚的弱上拉和恒流驱动功能。当 DAC1OUT 引脚已被配置为 DAC 参考电压输出时，读取该引脚将总是返回 0。

由于电流驱动能力有限，因此必须在 DAC 参考电压输出引脚 DAC1OUT 上外接缓冲器。 图 35-2 举例说明了这一缓冲技术。

图 35-2. 参考电压输出缓冲器示例



35.4. 休眠期间的操作

DAC 在休眠期间继续工作。当器件通过中断或看门狗定时器超时从休眠模式唤醒时，DAC1CON0 寄存器中的内容不受影响。

35.5. 复位的影响

器件复位会产生以下影响：

- DACx 被禁止
- DACx 输出电压从 DACxOUTn 引脚上被移除
- DAC1R 范围选择位被清零

35.6. 寄存器定义：DAC 控制

35.6.1. DAC1CON0

名称： DAC1CON0
偏移量： 0xF2E

DAC 控制寄存器

位	7	6	5	4	3	2	1	0
	EN		OE1	OE2	PSS[1:0]			NSS
访问	R/W		R/W	R/W	R/W	R/W		R/W
复位	0		0	0	0	0		0

Bit 7 – EN DAC 使能位

值	说明
1	使能 DAC
0	禁止 DAC

Bit 5 – OE1 DAC 电压输出使能位

值	说明
1	DAC 电压从 DAC1OUT1 引脚输出
0	DAC 电压从 DAC1OUT1 引脚断开

Bit 4 – OE2 DAC 电压输出使能位

值	说明
1	DAC 电压从 DAC1OUT2 引脚输出
0	DAC 电压从 DAC1OUT2 引脚断开

Bit 3:2 – PSS[1:0] DAC 正电压源选择位

值	说明
11	保留
10	FVR 缓冲区
01	$V_{REF}+$
00	AV_{DD}

Bit 0 – NSS DAC 负电压源选择位

值	说明
1	$V_{REF}-$
0	AV_{SS}

35.6.2. DAC1CON1

名称： DAC1CON1
偏移量： 0xF2F

DAC 数据寄存器

位	7	6	5	4	3	2	1	0
				DAC1R[4:0]				
访问				R/W	R/W	R/W	R/W	R/W
复位				0	0	0	0	0

Bit 4:0 – DAC1R[4:0] DAC 数据输入寄存器位

35.7. 寄存器汇总——DAC 控制

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0F2D	保留									
0x0F2E	DAC1CON0	7:0	EN		OE1	OE2	PSS[1:0]			NSS
0x0F2F	DAC1CON1	7:0				DAC1R[4:0]				

36. CMP——比较器模块

比较器用作模拟电路与数字电路的接口，通过比较两个模拟电压的大小并输出一个数字量以指示输入量的相对大小。比较器是非常有用的混合信号模块，因为它们提供了与程序执行相独立的模拟功能。CN2710器件有两个比较器（C1/C2）。

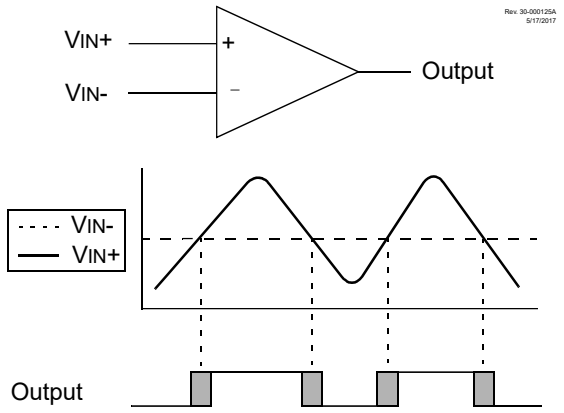
模拟比较器模块具有以下特性：

- 可编程输入选择
- 可编程输出极性
- 输出上升沿/下降沿中断
- 从休眠模式唤醒
- CWG 自动关断源
- 可选参考电压
- ADC 自动触发器
- 奇数编号定时器（Timer1 和 Timer3 等）门控
- 偶数编号定时器（Timer2 和 Timer4 等）复位
- CCP 捕捉模式输入
- DSM 调制器源
- 输入和窗口信号-信号测量定时器

36.1. 比较器概述

图 36-1 所示为单比较器以及模拟输入电压与数字输出之间的关系。当 V_{IN}+上的模拟电压小于 V_{IN}-上的模拟电压时，比较器输出为数字低电平。当 V_{IN}+上的模拟电压大于 V_{IN}-上的模拟电压时，比较器输出为数字高电平。

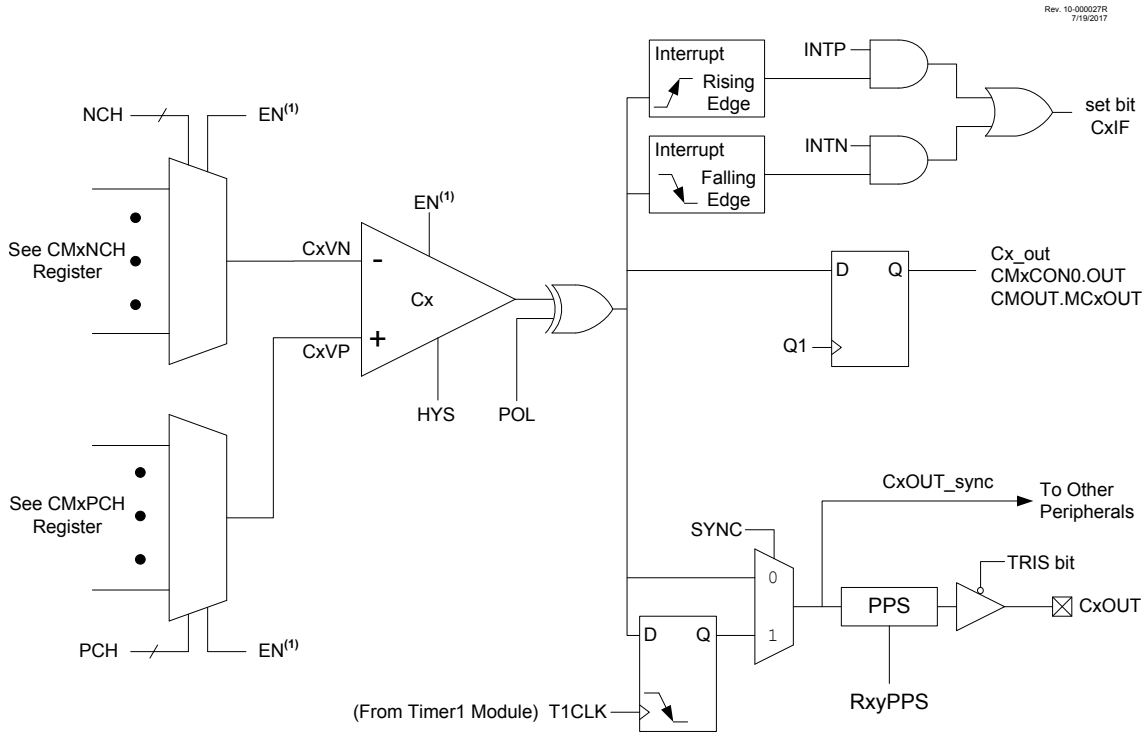
图 36-1. 单比较器



注：

1. 比较器输出的黑色区域表示因输入失调电压和响应时间所造成的输出不确定区域。

图 36-2. 比较器模块简化框图



Note 1: **When EN = 0, all multiplexer inputs are disconnected and the Comparator will produce a '0' at the output.**

36.2. 比较器控制

每个比较器都具有 2 个控制寄存器：CMxCON0 和 CMxCON1。

CMxCON0 寄存器包含以下控制和状态位：

- 使能
- 输出
- 输出极性
- 滞后使能
- Timer1 输出同步

CMxCON1 寄存器包含以下控制位：

- 正/负边沿中断允许
- 同相输入通道选择
- 反相输入通道选择

36.2.1. 比较器使能

将 EN 位置 1 可使能比较器操作。将 CxEN 位清零可禁止比较器，从而使电流消耗降至最低。

36.2.2. 比较器输出

可通过读 CxOUT 位或 MCxOUT 位来监视比较器输出。

比较器输出还可以通过 RxyPPS 寄存器输送到外部引脚。要使能引脚作为输出，必须清零相应的 TRIS 位。

注：

1. 比较器的内部输出在每个指令周期被锁存。除非另外指定，否则不锁存外部输出。

36.2.3. 比较器输出极性

将比较器的输出反相在功能上等效于交换比较器输入。可以通过将 CxPOL 位置 1 来使比较器输出的极性反相。清零 CxPOL 位得到的是同相输出。

表 36-1 给出了输出状态与输入条件的关系（包括极性控制）。

表 36-1. 比较器输出状态与输入条件

输入条件	CxPOL	CxOUT
CxVn > CxVp	0	0
CxVn < CxVp	0	1
CxVn > CxVp	1	1
CxVn < CxVp	1	0

36.3. 比较器滞后

通过在每个比较器的输入引脚上施加不同的正向和负向增长阈值电压，可以为整体操作提供滞后功能。滞后功能通过将 CxHYS 位置 1 来使能。

更多信息，请参见**“比较器规范”**一节。

36.4. Timer1 门控操作

比较器操作产生的输出可以用作奇数编号定时器（Timer1 和 Timer3 等）的门控源。更多信息，请参见定时器门控部分。该功能可用于对模拟事件的持续时间或间隔时间进行计时。

建议通过将 SYNC 位置 1 使比较器输出与定时器同步。这可以确保在比较器中发生变化时，定时器不会递增。但是，只能通过 Timer1 时钟源实现同步。其他奇数编号定时器只有在使用与 Timer1 相同的时钟源时才能与比较器输出同步。

36.4.1. 比较器输出同步

通过将 SYNC 位置 1，可以使比较器的输出与 Timer1 保持同步。

使能比较器的输出时，比较器的输出在 Timer1 时钟源的下降沿被锁存。如果 Timer1 使用了预分频器，则比较器的输出在经过预分频后被锁存。为了防止发生竞争，比较器的输出在 Timer1 时钟源的下降沿被锁存，而 Timer1 在其时钟源的上升沿递增。更多信息，请参见图 36-2 和图 19-1。

36.5. 比较器中断

当上升沿或下降沿检测器检测到每个比较器的输出值变化时，可以产生中断。

当触发任一边沿检测器时，如果它关联的允许位已置 1（CxINTP 和/或 CxINTN 位），则相应的中断标志位（PIR2 寄存器的 CxIF 位）会置 1。

要允许中断，必须将以下位置 1：

- EN 和 POL 位
- PIE2 寄存器的 CxIE 位
- INTP 位（对于上升沿检测）
- INTN 位（对于下降沿检测）
- INTCON 寄存器的 PEIE 和 GIE 位

PIR2 寄存器的相关中断标志位 CxIF 必须用软件清零。如果在该标志位清零期间又检测到一边沿，则在该序列结束时仍将该标志位置 1。

重要：即使比较器被禁止，还是可以通过使用 CxPOL 位更改输出极性来产生中断，或者通过使用 CxEN 位开启或关闭比较器来产生中断。

36.6. 比较器同相输入选择

通过配置 CMxPCH 寄存器中的 PCH 位，将内部参考电压或模拟引脚连接到比较器的同相输入：

- CxINy+模拟引脚
- DAC 输出
- 固定参考电压（FVR）

重要：要将 CxINy+引脚用作模拟输入，必须将 ANSEL 寄存器中的相应位置 1，同时也必须将相应的 TRIS 位置 1 来禁止输出驱动器。

每当禁止比较器（CxEN = 0）时，所有比较器输入都会被禁止。

36.7. 比较器反相输入选择

CMxNCH 寄存器中的 NCH 位指示模拟输入引脚、内部参考电压或模拟地连接到比较器的反相输入：

- CxINy-模拟引脚
- 固定参考电压（FVR）

重要：要将 CxINy-引脚用作模拟输入，必须将 ANSEL 寄存器中的相应位置 1，同时也必须将相应的 TRIS 位置 1 来禁止输出驱动器。

36.8. 比较器响应时间

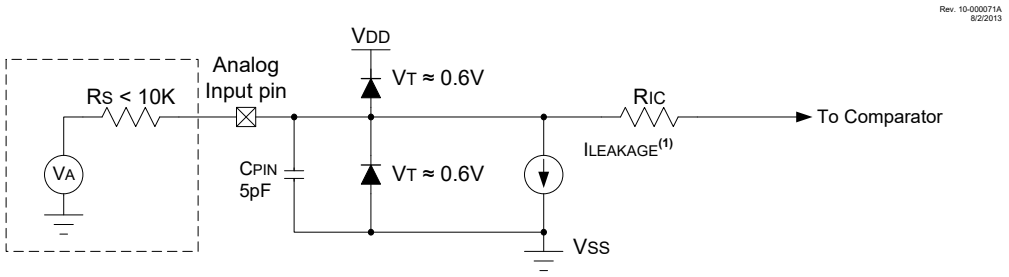
在改变输入源或选择新的参考电压后的一段时间内，比较器的输出状态是不确定的。这段时间被称为响应时间。比较器的响应时间不同于参考电压的稳定时间。因此，在确定比较器输入改变的总响应时间时，必须考虑这两个时间。更多详细信息，请参见“比较器规范和固定参考电压（FVR）规范”部分的比较器和参考电压规范。

36.9. 模拟输入连接注意事项

模拟输入的简化电路如图 36-3 所示。由于模拟输入引脚与数字输入共用连接，它们与 V_{DD} 和 V_{SS} 之间连有反向偏置的 ESD 保护二极管。因此，模拟输入电压必须在 V_{SS} 与 V_{DD} 之间。如果输入电压与这一范围偏离的绝对值超过 0.6V，就可能发生一个二极管正向导通，从而可能导致锁死发生。

模拟信号源的最大阻抗推荐值为 10 kΩ。此外，对于任何连接到模拟输入引脚的外部元件（如电容或齐纳二极管），必须保证其泄漏电流极小以使引入的误差降至最小。

图 36-3. 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 = Threshold Voltage

36.10. CWG1 自动关断源

比较器模块的输出可用作 CWG1 模块的自动关断源。当比较器的输出有效且相应的 WGASxE 使能时，CWG 操作立即暂停。

36.11. ADC 自动触发源

比较器模块的输出可用于触发 ADC 转换。如果将 ADACT 寄存器设置为通过比较器输出触发，则比较器输出变为高电平时将触发 ADC 转换。

36.12. 偶编号定时器复位

比较器模块的输出可用于复位偶编号定时器（Timer2 和 Timer4 等）。如果对 TxERS 寄存器进行了相应设置，则当比较器输出变为高电平时定时器将复位。

36.13. 在休眠模式下工作

比较器模块可以在休眠模式下工作。比较器时钟源基于 Timer1 时钟源。如果 Timer1 时钟源为系统时钟（F_{OSC}）或指令时钟（$F_{OSC}/4$），Timer1 将不会在休眠期间工作，并且同步比较器输出将无法运行。

比较器中断可以将器件从休眠模式唤醒。PIE2 寄存器的 CxIE 位必须置 1 才能允许比较器中断。

36.14. 寄存器定义：比较器控制

36.14.1. CMOUT

名称： CMOUT
偏移量： 0xF38

比较器输出寄存器

位	7	6	5	4	3	2	1	0
							MC2OUT	MC1OUT
访问							RO	RO
复位							0	0

Bit 0, 1 – MCxOUT CxOUT 的镜像副本位

36.14.2. CMxCON0

名称： CMxCON0
偏移量： 0xF34,0xF30

比较器 x 控制寄存器 0

位	7	6	5	4	3	2	1	0
	EN	OUT		POL			HYS	SYNC
访问	R/W	RO		R/W			R/W	R/W
复位	0	0		0			0	0

Bit 7 – EN 比较器使能位

值	说明
1	使能比较器
0	禁止比较器，不消耗有功功率

Bit 6 – OUT 比较器输出位

值	条件	说明
1	如果 POL = 0（极性不反相）：	CxVP > CxVN
0	如果 POL = 0（极性不反相）：	CxVP < CxVN
1	如果 POL = 1（极性反相）：	CxVP < CxVN
0	如果 POL = 1（极性反相）：	CxVP > CxVN

Bit 4 – POL 比较器输出极性选择位

值	说明
1	比较器输出反相
0	比较器输出不反相

Bit 1 – HYS 比较器滞后使能位

值	说明
1	使能比较器滞后
0	禁止比较器滞后

Bit 0 – SYNC 比较器输出同步模式位
输出在预分频 Timer1 时钟源的下降沿进行更新。

值	说明
1	送到 Timer1 和 I/O 引脚的比较器输出与预分频 Timer1 时钟的变化同步
0	送到 Timer1 和 I/O 引脚的比较器输出是异步的

36.14.3. CMxCON1

名称： CMxCON1
偏移量： 0xF35,0xF31

比较器 x 控制寄存器 1

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

Bit 1 – INTP 比较器正向边沿中断允许位

值	说明
1	在 CxOUT 位的正向边沿，CxIF 中断标志将置 1
0	在 CxOUT 位的正向边沿，CxIF 中断标志不会置 1

Bit 0 – INTN 比较器负向边沿中断允许位

值	说明
1	在 CxOUT 位的负向边沿，CxIF 中断标志将置 1
0	在 CxOUT 位的负向边沿，CxIF 中断标志不会置 1

36.14.4. CMxNCH

名称： CMxNCH
偏移量： 0xF36,0xF32

比较器 x 反相通道选择寄存器

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

Bit 2:0 – NCH[2:0] 比较器反相输入通道选择位

NCH	反相输入源
111	V_{SS}
110	FVR 缓冲器 2
101	CxNCH 未连接
100	CxNCH 未连接
011	CxIN3-
010	CxIN2-
001	CxIN1-
000	CxIN0-

36.14.5. CMxPCH

名称： CMxPCH
偏移量： 0xF37,0xF33

比较器 x 同相通道选择寄存器

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

Bit 2:0 – PCH[2:0] 比较器同相输入通道选择位

PCH	同相输入源
111	V_{SS}
110	FVR 缓冲器 2
101	DAC 输出
100	CxPCH 未连接
011	CxPCH 未连接
010	CxPCH 未连接
001	CxIN1+
000	CxIN0+

36.15. 寄存器汇总——比较器

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0F2F	保留									
0x0F30	CM2CON0	7:0	EN	OUT		POL			HYS	SYNC
0x0F31	CM2CON1	7:0							INTP	INTN
0x0F32	CM2NCH	7:0						NCH[2:0]		
0x0F33	CM2PCH	7:0						PCH[2:0]		
0x0F34	CM1CON0	7:0	EN	OUT		POL			HYS	SYNC
0x0F35	CM1CON1	7:0							INTP	INTN
0x0F36	CM1NCH	7:0						NCH[2:0]		
0x0F37	CM1PCH	7:0						PCH[2:0]		
0x0F38	CMOUT	7:0							MC2OUT	MC1OUT

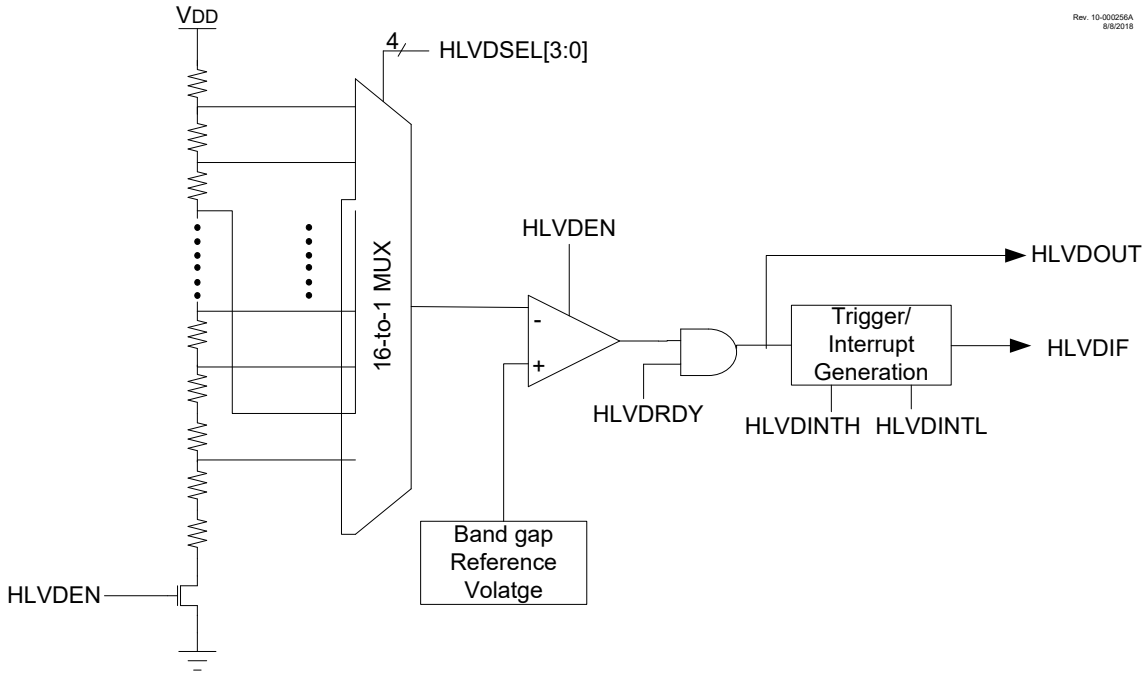
37. HLVD——高/低电压检测

HLVD 模块经配置后可监视器件电压。这在电池监视应用中非常有用。

使用 HLVDCON0 和 HLVDCON1 寄存器即可完全控制 HLVD 模块。

该模块的框图如下图所示。

图 37-1. HLVD 模块框图



由于可使用软件通过 HLVDEN 位使能 HLVD，因此将使能位置 1 和清零不会产生错误的 HLVD 事件毛刺。每次使能 HLVD 模块时，均可使用 RDY 位来检测模块何时达到稳定且准备就绪。

INTH 和 INTL 位用于决定模块的整体操作。当 INTH 置 1 时，模块会监视 V_{DD} 是否上升到由 SEL 位设置的跳变点以上。当 INTL 置 1 时，模块会监视 V_{DD} 是否下降到由 SEL 位设置的跳变点以下。当 INTH 和 INTL 位均置 1 时，模块会同时监视 VDD 是否上升/下降到由 SEL 位设置的跳变点以上/以下。

可通过读取 OUT 位来确定电压是否高于或低于选定跳变点。

37.1. 工作原理

使能 HLVD 模块时，比较器会使用内部生成的参考电压作为设定值。将设定值与跳变点进行比较，其中电阻分压器中的每个节点代表一个跳变点电压。“跳变点”电压是器件检测到高压或低压事件时的电压，具体取决于模块的配置。

当供电电压等于跳变点时，电阻阵列的节点电压等于由参考电压模块产生的内部参考电压。然后，比较器通过将 HLVDIF 位置 1 来产生中断信号。

可用软件将跳变点电压编程为 16 个值中的任意一个。通过编程 SEL 位可以选择跳变点。

37.2. 设置

要设置 HLVD 模块，请按以下步骤操作：

1. 通过向 HLVDCON1 寄存器的 SEL 位中写入值来选择所需的 HLVD 跳变点。

2. 根据要检测高电压、低电压还是同时检测两者，相应地将 INTH 和/或 INTL 位置 1。
3. 通过将 EN 位置 1，使能 HLVD 模块。
4. 清零 HLVD 中断标志（HLVDIF），该标志可能已由先前的中断置 1。
5. 如果需要中断，可通过将 HLVDIE 和 GIE 位置 1 来允许 HLVD 中断。
   待 RDY 位置 1 后，便可产生中断。

重要：在更改任何模块设置（中断和跳变点）之前，需先禁止模块（EN = 0），完成更改后再重新使能模块。这样可以防止产生错误的 HLVD 事件。

37.3. 电流消耗

当模块使能时，HLVD 比较器和分压器也会随之使能并产生静态电流消耗。有关模块使能时的总电流消耗，可参见电气规范参数 D206。

根据具体应用，HLVD 模块有时并不需要持续工作。为了降低电流消耗，可以仅在检测电压时短暂地使能模块，检测完成后即可禁止模块。

37.4. HLVD 启动时间

如果通过禁止 HLVD 或其他使用内部参考电压的电路来降低器件的电流消耗，则需要花费一段时间让参考电压电路稳定下来，之后才能可靠地检测低压或高压状况。启动时间 T_{FVRST} 与器件时钟速度无关，具体值请参见电气规范。

HLVD 中断标志需在 T_{FVRST} 结束且参考电压达到稳定之后才会使能。因此，这段时间可能无法检测到超出设定值的短暂偏离（见下图）。

图 37-2. 低电压检测工作原理（INTL = 1）

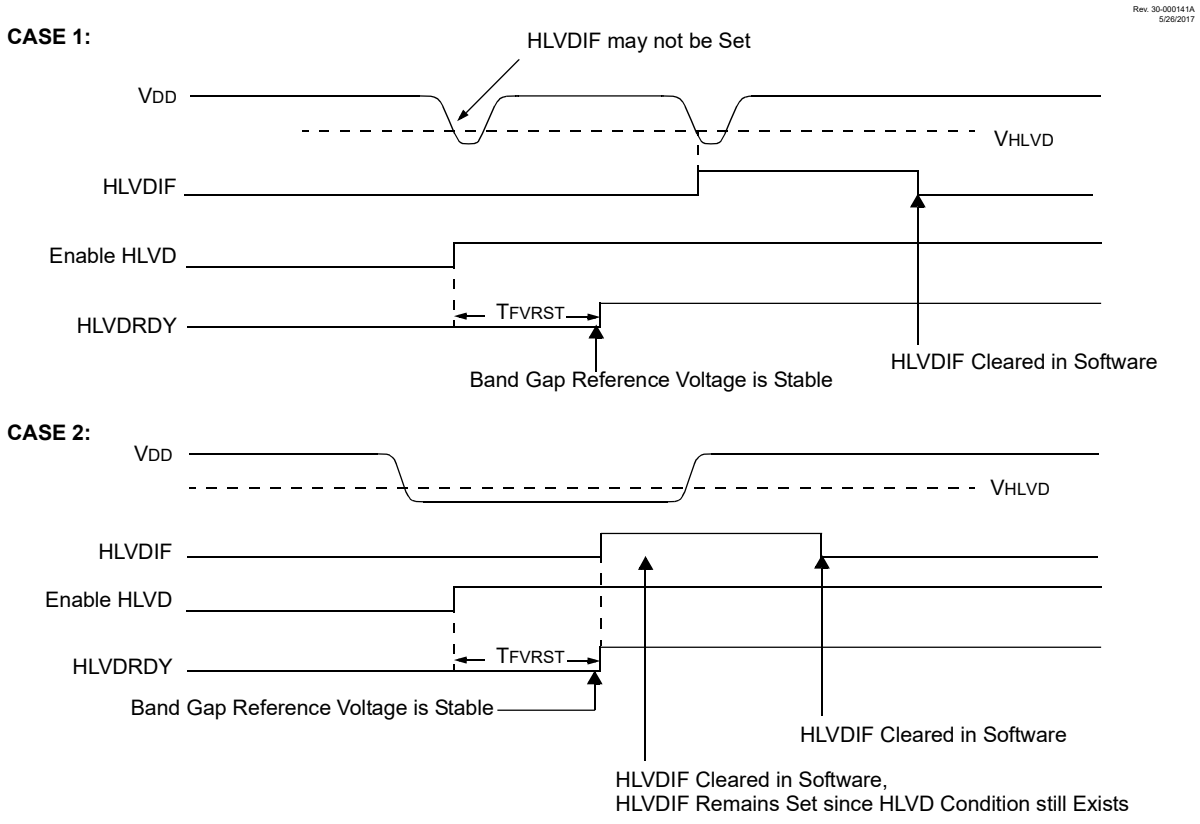
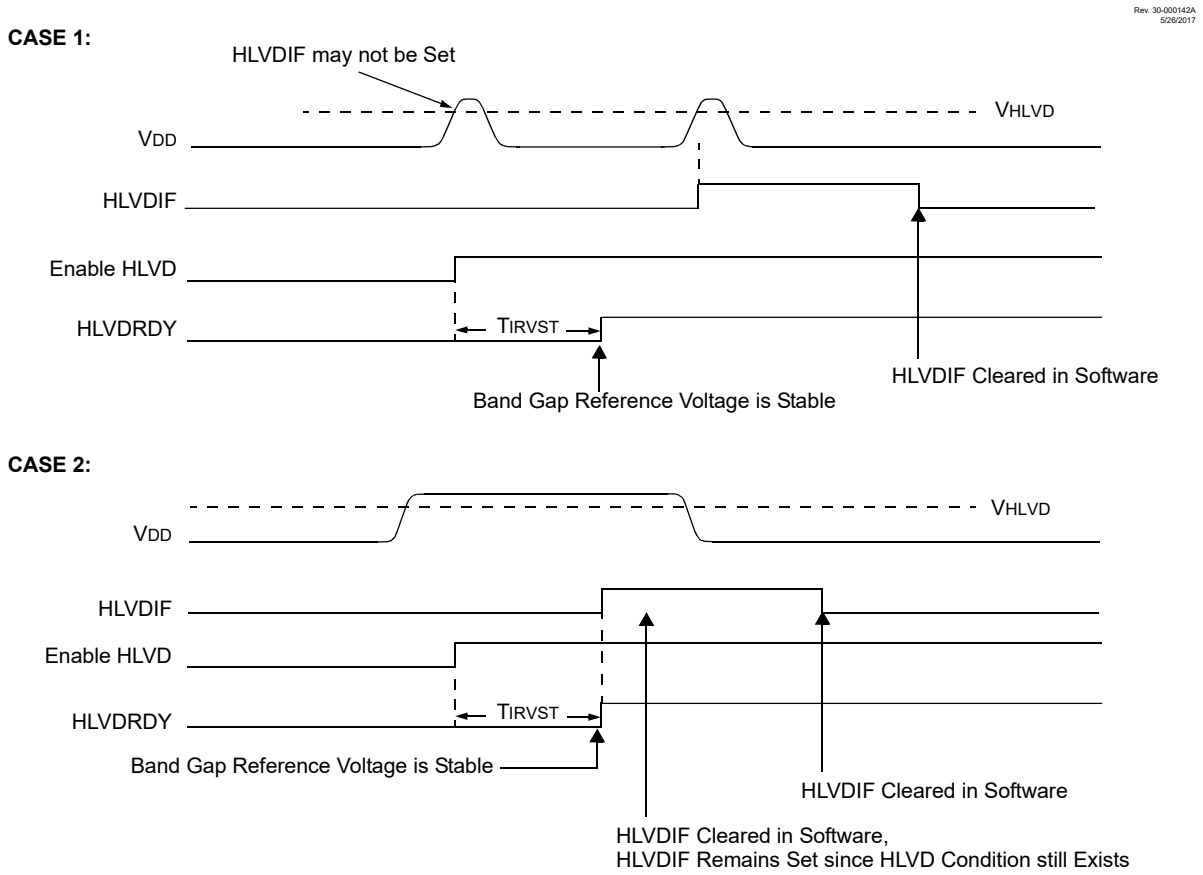


图 37-3. 高电压检测工作原理（INTH = 1）

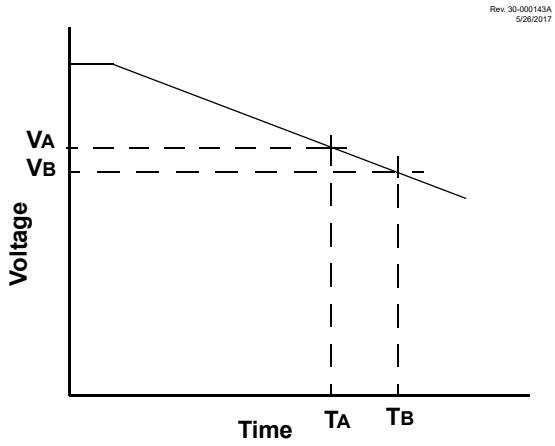


37.5. 应用

在许多应用中，当电压低于或高于某个电压阈值时，系统希望可以检测到该事件。例如，可以定期使能 HLVD 模块来检测通用串行总线（USB）的连接状态。这里假定断开连接时器件的供电电压低于 USB 电压。如果连接了 USB，将检测到 3.3V 到 5V 的高电压（USB 上的电压）；如果断开连接，情况正好相反。凭借此功能，可以在设计中省去一些额外的元件和连接信号（输入引脚）。

对于一般的电池应用，下图给出了一条可能的电压曲线。器件电压会随时间逐渐下降。当器件电压达到电压 V_A 时，HLVD 逻辑会在时间 T_A 产生中断。该中断可导致执行中断服务程序（ISR），从而使应用程序能在器件电压退出有效工作范围（对应的时间为 T_B）之前执行“后台处理任务”，并执行受控关闭。这样应用程序就能有一个时间窗口（用 T_A 和 T_B 的时间差表示）来安全退出。

图 37-4. 典型低电压检测应用



Legend: VA = HLVD trip point
VB = Minimum valid device operating voltage

37.6. 休眠期间的操作

使能的 HLVD 电路在休眠期间会继续工作。如果器件电压超过跳变点，则 HLVDIF 位将置 1，同时器件将从休眠模式唤醒。如果已全局允许中断，则器件将从中断向量地址继续执行。

37.7. 空闲模式和打盹模式期间的操作

在空闲模式和打盹模式下，模块处于工作状态，并且如果使能外设，则会生成相关事件。

37.8. 复位的影响

器件复位将强制所有寄存器为复位状态。这会强制关闭 HLVD 模块。

37.9. 寄存器定义：HLVD 控制

下表列出了 HLVD 外设的长位名称前缀。更多信息，请参见**“长位名称”**一节。

表 37-1. HLVD 长位名称前缀

外设	位名称前缀
HLVD	HLVD

37.9.1. HLVDCON0

名称： HLVDCON0
偏移量： 0xF2A

高/低电压检测控制寄存器 0

位	7	6	5	4	3	2	1	0
	EN		OUT	RDY			INTH	INTL
访问	R/W		RO	RO			R/W	R/W
复位	0		x	x			0	0

Bit 7 – EN 高/低电压检测功能使能位

值	说明
1	使能 HLVD 模块
0	禁止 HLVD 模块

Bit 5 – OUT HLVD 比较器输出位

值	说明
1	电压 ≤ 选定的检测限值（SEL）
0	电压 ≥ 选定的检测限值（SEL）

Bit 4 – RDY 带隙参考电压稳定状态标志位

值	说明
1	指示 HLVD 模块已就绪且输出稳定
0	指示 HLVD 模块未就绪

Bit 1 – INTH HLVD 正向（高电压）中断允许

值	说明
1	HLVDIF 将在电压 ≥ 选定的检测限值（SEL）时置 1
0	HLVDIF 不会置 1

Bit 0 – INTL HLVD 负向（低电压）中断允许

值	说明
1	HLVDIF 将在电压 ≤ 选定的检测限值（SEL）时置 1
0	HLVDIF 不会置 1

37.9.2. HLVDCON1

名称： HLVDCON1
偏移量： 0xF2B

低电压检测控制寄存器 1

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

Bit 3:0 – SEL[3:0] 高/低电压检测限值选择位

表 37-2. HLVD 检测限值

SEL	检测限值
1111	**保留**
1110 至 0000	请参见高/低电压检测特性

复位状态： POR/BOR = 0000
所有其他复位时的值 = uuuu

37.10. 寄存器汇总——HLVD

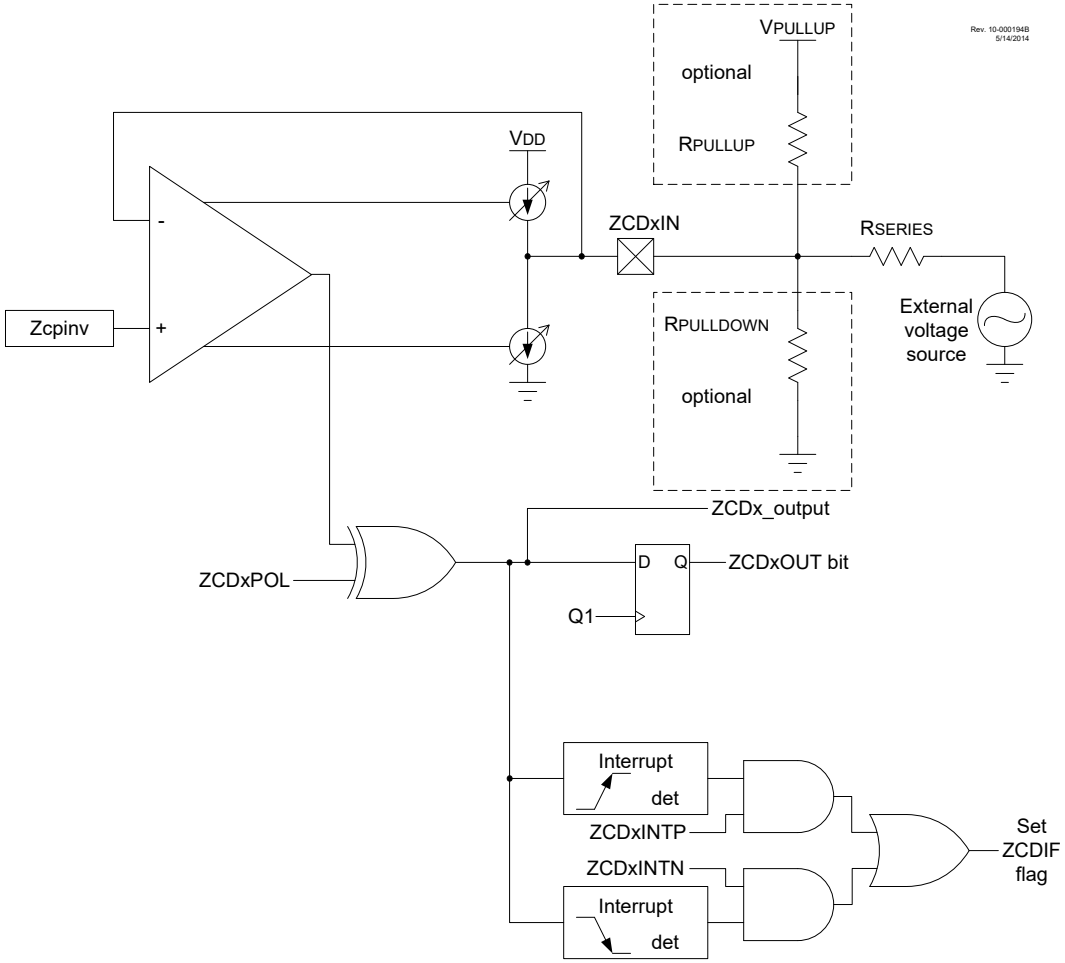
偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0F29	保留									
0x0F2A	HLVDCON0	7:0	EN		OUT	RDY			INTH	INTL
0x0F2B	HLVDCON1	7:0					SEL[3:0]			

38. ZCD——过零检测模块

过零检测（ZCD）模块用于检测交流信号何时越过地电位。实际过零阈值为过零参考电压 Z_{CPINV}，它通常比地电位高 0.75V。

要检测的信号通过一个串联限流电阻进行连接。该模块在 ZCD 引脚上施加拉电流或灌电流，以便在该引脚上维持恒定电压，从而防止引脚电压使 ESD 保护二极管正向偏置。当施加的电压大于参考电压时，模块产生灌电流。当施加的电压小于参考电压时，模块产生拉电流。拉电流和灌电流操作使引脚电压在所施加电压的完整范围内保持恒定。以下为 ZCD 模块的简化框图。

图 38-1. ZCD 简化框图



ZCD 模块可用于为以下用途（但不限于以下用途）监视交流波形：

- 交流周期测量
- 精确的长期时间测量
- 调光器相位延时驱动
- 低 EMI 周期切换

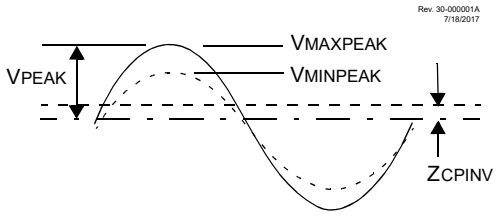
38.1. 外部电阻选择

ZCD 模块要求将一个限流电阻与外部电压源串联。该电阻的阻抗和额定值取决于外部源峰值电压。选择的电阻值需要在通过电阻的电流为 300 μA 标称值时降低全部峰值电压。确保已禁止 ZCD I/O 引脚的内部弱上拉功能，使其不会干扰拉电流和灌电流。

公式 38-1. 外部电阻

$$R_{SERIES} = \frac{V_{PEAK}}{3 \times 10^{-4}}$$

图 38-2. 外部电压源



38.2. ZCD 逻辑输出

ZCD 模块包含一个状态位，可以通过读取它来确定起作用的是拉电流还是灌电流。OUT 位在灌电流起作用时置 1，在拉电流起作用时清零。OUT 位受极性位影响。

OUT 信号也可用作其他模块的输入。这由相应模块的寄存器控制。OUT 可用作以下源：

- TMR1/3/5 的门控源
- TMR2/4/6 的时钟源
- TMR2/4/6 的复位源

38.3. ZCD 逻辑极性

POL 位会相对于拉电流和灌电流输出反转 OUT 位。当 POL 位置 1 时，OUT 高电平指示拉电流在起作用，低电平输出指示灌电流在起作用。

POL 位会影响 ZCD 中断。

38.4. ZCD 中断

如果相应的中断允许位置 1，则在 ZCD 逻辑输出改变时，将会产生中断。因此，ZCD 中具有一个上升沿检测器和一个下降沿检测器。

触发其中一个边沿检测器，且其相关的中断允许位置 1 时，PIR2 寄存器的 ZCDIF 位会置 1。ZCDxCON 寄存器中的 INTP 位用于允许上升沿中断，ZCDxCON 寄存器中的 INTN 位用于允许下降沿中断。

要完全允许中断，必须将以下位置 1：

- PIE2 寄存器的 ZCDIE 位
- INTP 位（对于上升沿检测）
- INTN 位（对于下降沿检测）
- INTCON 寄存器的 PEIE 和 GIE 位

无论 SEN 位的电平如何，更改 POL 位均会产生中断。

作为中断服务程序的一部分，必须用软件将 PIR2 寄存器的 ZCDIF 位清零。如果在该标志位清零期间又检测到一边沿，则在该序列结束时仍将该标志位置 1。

38.5. 校正 Z_{CPINV} 失调

ZCD 的实际开关电压是 ZCD 运放同相输入端的参考电压。对于外部电压源波形（方波除外），这一偏移零点的失调电压会导致过零事件过早或过晚出现。

38.5.1. 通过交流耦合校正

当外部电压源为正弦波时，除了使用降压电阻之外，还可以使用电容将外部电压源与 ZCD 引脚隔离，来消除 Z_{CPINV} 失调的影响。电容将引起相移，最终导致 ZCD 输出在实际过零事件之前切换。上升过零和下降过零的相移是相同的，通过定时器或其他方法延迟 CPU 对 ZCD 切换的响应可以补偿相移，或者也可以选择足够大的电容值，使相移可以忽略不计。

要确定该配置的串联电阻和电容值，首先应计算获得 300 µA 峰值电流所需的阻抗 Z。接下来，随意选择一个容值适当的无极性电容并使用外部电压源频率计算其电抗 X_C。最后，通过如下所示的公式计算串联电阻、电容峰值电压和相移。

如果在没有输入信号的情况下使用该技术，ZCD 往往会发生振荡。为避免这种振荡，可以使用高阻抗电阻（如 200K）将 ZCD 引脚连接到 V_{DD} 或 GND。

公式 38-2. R-C 公式

V_{PEAK} = 外部电压源峰值电压

f = 外部电压源频率

C = 串联电容

R = 串联电阻

V_C = 电容峰值电压

Φ = 电容引起的过零相位超前（以弧度为单位）

T_Φ = 在实际过零之前发生 ZC 事件的时间

$$Z = \frac{V_{PEAK}}{3 \times 10^{-4}}$$

$$X_C = \frac{1}{2\pi f C}$$

$$R = \sqrt{Z^2 - {X_C}^2}$$

$$V_C = X_C\left(3 \times 10^{-4}\right)$$

$$\Phi = \tan^{-1}\theta\left(\frac{X_C}{R}\right)$$

$$T_\Phi = \frac{\Phi}{2\pi f}$$

公式 38-3. R-C 计算示例

$$V_{rms} = 120$$

$$V_{PEAK} = V_{rms} \times \sqrt{2} = 169.7$$

$f = 60\ Hz$

$C = 0.1\ \mu F$

$Z = \frac{V_{PEAK}}{3 \times 10^{-4}} = \frac{169.7}{3 \times 10^{-4}} = 565.7\ k\Omega$

$X_C = \frac{1}{2\pi fC} = \frac{1}{2\pi \times 60 \times 10^{-7}} = 26.53\ k\Omega$

$R = \sqrt{Z^2 - X_C^2} = 565.1\ k\Omega$（计算结果）

$R_a = 560\ k\Omega$（使用值）

$Z_R = \sqrt{R_a^{\ 2} + X_C^{\ 2}} = 560.6\ k\Omega$

$I_{PEAK} = \frac{V_{PEAK}}{Z_R} = 302.7 \times 10^{-6} A$

$V_C = X_C \times I_{PEAK} = 8.0\ V$

$\Phi = \tan^{-1}\theta\left(\frac{X_C}{R}\right) = 0.047$ 弧度

$T_\Phi = \frac{\Phi}{2\pi f} = 125.6\ \mu s$

38.5.2. 通过电流失调校正

当波形相对 V_{SS} 发生变化时，若波形下降，则会过早检测到过零事件；若波形上升，则会过晚检测到过零事件。当波形相对 V_{DD} 发生变化时，若波形上升，则会过晚检测到过零事件；若波形下降，则会过早检测到过零事件。可根据如下所示的公式确定正弦波形的实际失调时间。

公式 38-4. ZCD 事件失调

当外部电压源相对 V_{SS} 发生变化时：

$$T_{offset} = \frac{\sin^{-1}\left(\frac{Z_{CPINV}}{V_{PEAK}}\right)}{2\pi f}$$

当外部电压源相对 V_{DD} 发生变化时：

$$T_{offset} = \frac{\sin^{-1}\left(\frac{V_{DD} - Z_{CPINV}}{V_{PEAK}}\right)}{2\pi f}$$

可通过向 ZCD 引脚添加上拉或下拉偏置电阻来补偿这段失调时间。当外部电压源相对 V_{SS} 发生变化时，使用上拉电阻。当电压相对 V_{DD} 发生变化时，使用下拉电阻。此电阻可为 ZCD 引脚添加偏置电压，这样目标外部电压源必须达到零点才能将引脚电压拉至 Z_{CPINV} 开关电压。可根据如下所示的公式来确定上拉或下拉电阻值。

公式 38-5. ZCD 上拉/下拉电阻

当外部电压源相对 V_{SS} 发生变化时：

$$R_{pullup} = \frac{R_{SERIES}(V_{pullup} - Z_{CPINV})}{Z_{CPINV}}$$

当外部电压源相对 V_{DD} 发生变化时：

$$R_{pulldoWn} = \frac{R_{SERIES}(Z_{CPINV})}{(V_{DD} - Z_{CPINV})}$$

38.6. 处理 V_{PEAK} 变化

如果预计外部电压的峰值幅值将发生变化，则必须选择使用串联电阻以保持 ZCD 拉电流和灌电流低于设计的最大范围±600 µA 并高于合理的最小范围。最大峰值电压不得超过最小峰值电压的六倍。为确保最大电流不超过±600 µA 且最小电流至少为±100 µA，请根据公式 38-6 来计算串联电阻。可根据公式 38-5 所示的公式来确定此串联电阻的补偿上拉电阻，因为上拉电阻值与峰值电压无关。

公式 38-6. 电压范围内的串联电阻

$$R_{SERIES} = \frac{V_{MAX\_PEAK} + V_{MIN\_PEAK}}{7 \times 10^{-4}}$$

38.7. 休眠期间的操作

ZCD 电流源和中断不受休眠影响。

38.8. 复位的影响

ZCD 电路可配置为上电复位（POR）时默认为有效或无效状态。当 $\overline{ZCD}$ 配置位清零时，ZCD 电路在 POR 时处于有效状态。当 $\overline{ZCD}$ 配置位置 1 时，必须将 SEN 位置 1 以使能 ZCD 模块。

38.9. 禁止 ZCD 模块

可以采用以下两种方式禁止 ZCD 模块：

1. 当 $\overline{ZCD}$ 配置位置 1 时，会禁止 ZCD 模块。在这种情况下，将 SEN 位置 1 将使能 ZCD 模块。当 $\overline{ZCD}$ 位清零时，ZCD 始终使能并且 SEN 位没有任何作用。
2. 此外，也可以使用 PMD3 寄存器的 ZCDMD 位禁止 ZCD。这会受到 $\overline{ZCD}$ 位状态的影响。

38.10. 寄存器定义：ZCD 控制

38.10.1. ZCDCON

名称： ZCDCON
偏移量： 0xF2D

过零检测控制寄存器

位	7	6	5	4	3	2	1	0
	SEN		OUT	POL			INTP	INTN
访问	R/W		RO	R/W			R/W	R/W
复位	0		x	0			0	0

Bit 7 – SEN 过零检测软件使能位
当 ZCD 熔丝清零时，会忽略该位。

值	条件	说明
X	ZCD 配置熔丝 = 0	始终使能过零检测。该位被忽略。
1	ZCD 配置熔丝 = 1	使能过零检测。ZCD 引脚被强制为输出拉电流和灌电流。
0	ZCD 配置熔丝 = 1	禁止过零检测。ZCD 引脚依照 PPS 和 TRIS 控制操作。

Bit 5 – OUT 过零检测数据输出位

值	条件	说明
1	POL = 0	ZCD 引脚灌电流
0	POL = 0	ZCD 引脚拉电流
1	POL = 1	ZCD 引脚拉电流
0	POL = 1	ZCD 引脚灌电流

Bit 4 – POL 过零检测极性位

值	说明
1	ZCD 逻辑输出反相
0	ZCD 逻辑输出同相

Bit 1 – INTP 过零检测正向边沿中断允许位

值	说明
1	当 ZCD_output 从低电平跳变为高电平时，ZCDIF 位置 1
0	当 ZCD_output 从低电平跳变为高电平时，ZCDIF 位不受影响

Bit 0 – INTN 过零检测负向边沿中断允许位

值	说明
1	当 ZCD_output 从高电平跳变为低电平时，ZCDIF 位置 1
0	当 ZCD_output 从高电平跳变为低电平时，ZCDIF 位不受影响

38.11. 寄存器汇总——ZCD 控制

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0F2C	保留									
0x0F2D	ZCDCON	7:0	SEN		OUT	POL			INTP	INTN

39. 编程

编程允许用户在生产电路板时使用未编程器件。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 和配置字进行编程。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。

39.1. 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_{IHH}，可将器件置于高电压编程模式。

39.2. 低电压编程进入模式

低电压编程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。当配置字 4 寄存器的 LVP 位设置为 1 时，将会使能低电压编程进入模式。要禁止低电压编程模式，必须将 LVP 位编程为 0。

进入低电压编程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，而在 CLK 引脚上输出时钟。

在密钥序列完成后，只要保持在编程/校验模式下，$\overline{MCLR}$ 就必须保持在 V_{IL} 电平。

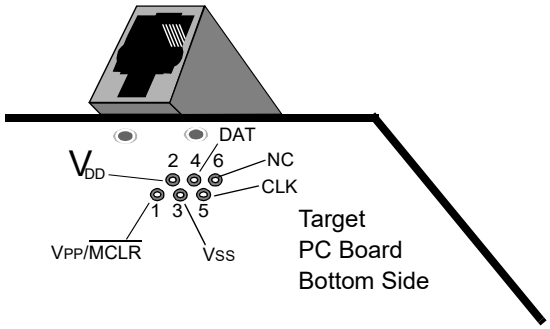
如果使能了低电压编程（LVP = 1），将自动使能 $\overline{MCLR}$ 复位功能，且无法禁止。更多信息，请参见 *$\overline{MCLR}$* 部分。

只能通过使用高电压编程模式将 LVP 位重新编程为 0。

39.3. 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 39-1。

图 39-1. 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

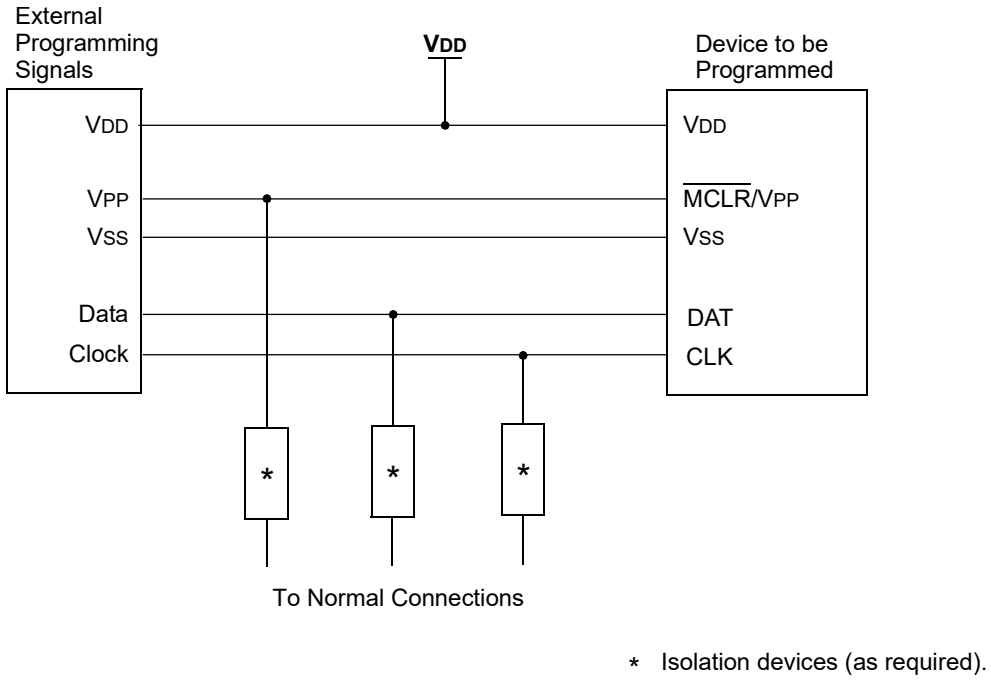
6 = 无连接

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 39-2。

注：

1. 6 引脚连接头（0.100"间距）可使用 0.025"方形引脚。

图 39-2. 编程的典型连接



* Isolation devices (as required).

40. 指令集汇总

CN2710 器件包含标准指令集（75 条指令）和扩展指令集（8 条新指令），用于优化递归或利用软件堆栈的代码。本节稍后将讨论扩展指令集。

40.1. 标准指令集

大部分指令的长度为一个程序存储字（16 位），但有四条指令需要两个程序存储单元。

每一条单字指令长 16 位，分为一个指定指令类型的操作码和进一步指定指令操作的一个或多个操作数。

整个指令集具有高度的正交性，分为以下 4 种基本类型：

- 面向字节的操作类指令
- 面向位的操作类指令
- 立即数操作类指令
- 控制操作类指令

表 40-2 是指令集的汇总，列出了面向字节的操作类指令、面向位的操作类指令、立即数操作类指令和控制操作类指令。 表 40-1 给出了操作码字段说明。

大多数面向字节的指令有三个操作数：

1. 文件寄存器（由 f 指定）。
2. 保存结果的目标寄存器（由 d 指定）。
3. 访问的存储器（由 a 指定）。

文件寄存器标识符 f 指定指令将会使用哪个文件寄存器。目标寄存器标识符 d 指定操作结果的存放位置。如果 d 为 0，则将操作结果存入 WREG 寄存器中。如果 d 为 1，则将操作结果存入指令指定的文件寄存器中。

所有面向位的指令有三个操作数：

1. 文件寄存器（由 f 指定）。
2. 文件寄存器中的位（由 b 指定）。
3. 访问的存储器（由 a 指定）。

位域标识符 b 用于选择操作所影响的位的编号，而文件寄存器标识符 f 代表位所在文件寄存器的编号。

立即数指令可以使用下列操作数：

- 装入到文件寄存器中的立即数（由 k 指定）
- 装入立即数的 FSR 寄存器（由 f 指定）
- 不需要操作数（由“—”指定）

控制操作类指令可使用以下操作数：

- 程序存储器地址（由 n 指定）
- CALL 或 RETURN 指令的模式（由 s 指定）
- 表读和表写指令的模式（由 m 指定）
- 不需要操作数（由“—”指定）

除了四个双字指令外，所有指令都是单字指令。这四个双字指令旨在包含所需的 32 位信息。在第二个字中，高 4 位全为 1。如果指令自身将第二个字当作一条指令来执行的话，它将作为一条 NOP 指令来执行。

所有单字指令都在一个指令周期内执行，除非条件测试为真或指令执行结果改变了程序计数器。对于上述两种特殊情况，指令执行需要两个指令周期，在额外的指令周期中执行 NOP 指令。

双字指令执行需要两个指令周期。

一个指令周期包含四个振荡器周期。因此，如果振荡器频率为 4 MHz，则正常的指令执行时间为 1 µs。如果条件测试为真或者指令执行结果改变了程序计数器的值，则指令执行时间为 2 µs。双字转移指令（如果为真）的执行时间为 3 µs。

图 40-1 显示指令的一般格式。所有示例均使用约定的“nnh”表示十六进制数。

指令集汇总（如表 40-2 所示）列出了标准指令。

标准指令集详解给出了每条指令的说明。

表 40-1. 操作码字段说明

字段	说明
a	RAM 访问位 a = 0：快速操作 RAM 中的 RAM 存储单元（忽略 BSR 寄存器） a = 1：RAM 存储区由 BSR 寄存器指定
bbb	8 位文件寄存器内的位地址（0 至 7）
BSR	存储区选择寄存器。用于选择当前 RAM 存储区。
C、DC、Z、OV 和 N	ALU 状态位：进位、半进位、全零、溢出和负标志位
d	目标选择位 d = 0：结果存入 WREG d = 1：结果存入文件寄存器 f
目标	目标：WREG 寄存器或指定的寄存器文件存储单元
f	8 位文件寄存器地址（00h 至 FFh）或 2 位 FSR 标识符（0h 至 3h）
f_s	12 位文件寄存器地址（000h 至 FFFh）。这是源地址。
f_d	12 位文件寄存器地址（000h 至 FFFh）。这是目标地址。
GIE	全局中断允许位
k	立即数字段，常量数据或标号（可以为 8 位、12 位或 20 位值）
label	标号名称
mm	用于表读和表写指令的 TBLPTR 寄存器的模式。 仅用于表读和表写指令：
*	不更改寄存器（例如用于表读和表写的 TBLPTR）
*+	后递增寄存器（例如用于表读和表写的 TBLPTR）
*-	后递减寄存器（例如用于表读和表写的 TBLPTR）
+*	预递增寄存器（例如用于表读和表写的 TBLPTR）
n	相对转移指令的相对地址（二进制补码）或 CALL/BRANCH 和 RETURN 指令的直接地址

表 40-1. 操作码字段说明 （续）

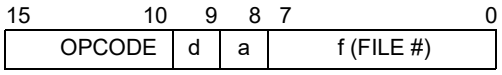
字段	说明
PC	程序计数器
PCL	程序计数器低字节
PCH	程序计数器高字节
PCLATH	程序计数器高字节锁存器
PCLATU	程序计数器最高字节锁存器
PD	掉电位
PRODH	乘积的高字节
PRODL	乘积的低字节
s	快速 Call/Return 模式选择位 s = 0：不会更新到影子寄存器/从影子寄存器更新 s = 1：某些寄存器装入影子寄存器/从影子寄存器装入某些寄存器（快速模式）
TBLPTR	21 位表指针（指向程序存储单元）
TABLAT	8 位表锁存器
TO	超时位
TOS	栈顶
u	未使用或未更改
WDT	看门狗定时器
WREG	工作寄存器（累加器）
x	无关（0 或 1）。
z_s	文件寄存器间接寻址的 7 位偏移值（源）
z_d	文件寄存器间接寻址的 7 位偏移值（目标）
{ }	可选参数
[text]	指示变址地址
(text)	text 的内容
[expr]<n>	指定由指针 expr 指示的寄存器的 bit n。

表 40-1. 操作码字段说明 （续）

字段	说明
→	赋值
`[ ]`	寄存器位域
∈	表示属于某个集合
斜体	用户定义项（字体是 Courier）

图 40-1. 指令的一般格式

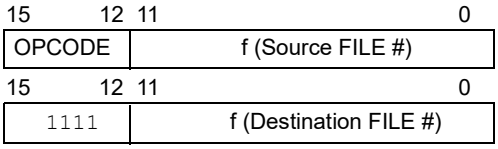
Byte-oriented file register operations — **Example Instruction**



`ADDWF MYREG, W, B`

d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

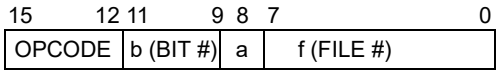
Byte to Byte move operations (2-word)



`MOVFF MYREG1, MYREG2`

f = 12-bit file register address

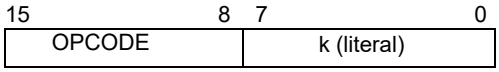
Bit-oriented file register operations



`BSF MYREG, bit, B`

b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

Literal operations

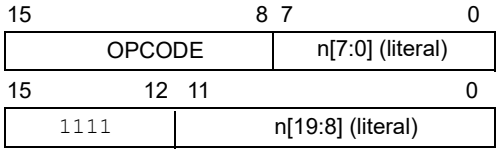


`MOVLW 7Fh`

k = 8-bit immediate value

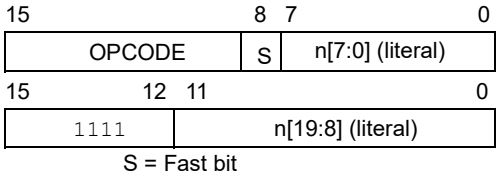
Control operations

CALL, GOTO and Branch operations



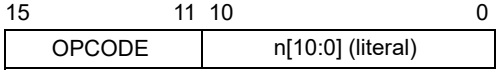
`GOTO Label`

n = 20-bit immediate value

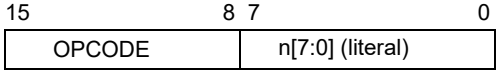


`CALL MYFUNC`

S = Fast bit



`BRA MYFUNC`



`BC MYFUNC`

表 40-2. 指令集

助记符和操作数		说明	指令周期数	16 位指令字				受影响的状态位	注
				MSb			LSb		
面向字节的操作类指令									
ADDWF	f, d, a	WREG 和 f 相加	1	0010	01da	ffff	ffff	C、DC、Z、OV 和 N	1, 2
ADDWFC	f, d, a	WREG 和 f 进行带进位的相加	1	0010	00da	ffff	ffff	C、DC、Z、OV 和 N	1, 2
ANDWF	f, d, a	WREG 和 f 作逻辑与运算	1	0001	01da	ffff	ffff	Z 和 N	1, 2
CLRF	f, a	将 f 清零	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	f 取反	1	0001	11da	ffff	ffff	Z 和 N	1, 2
CPFSEQ	f, a	比较 f 和 WREG，如果相等则跳过	1（2 或 3）	0110	001a	ffff	ffff	无	4
CPFSGT	f, a	比较 f 和 WREG，如果大于则跳过	1（2 或 3）	0110	010a	ffff	ffff	无	4
CPFSLT	f, a	比较 f 和 WREG，如果小于则跳过	1（2 或 3）	0110	000a	ffff	ffff	无	1, 2
DECF	f, d, a	f 递减 1	1	0000	01da	ffff	ffff	C、DC、Z、OV 和 N	1, 2, 3, 4
DECFSZ	f, d, a	f 递减 1，为 0 则跳过	1（2 或 3）	0010	11da	ffff	ffff	无	1, 2, 3, 4
DCFSNZ	f, d, a	f 递减 1，不为 0 则跳过	1（2 或 3）	0100	11da	ffff	ffff	无	1, 2
INCF	f, d, a	f 递增 1	1	0010	10da	ffff	ffff	C、DC、Z、OV 和 N	1, 2, 3, 4
INCFSZ	f, d, a	f 递增 1，为 0 则跳过	1（2 或 3）	0011	11da	ffff	ffff	无	4
INFSNZ	f, d, a	f 递增 1，不为 0 则跳过	1（2 或 3）	0100	11da	ffff	ffff	无	1, 2
IORWF	f, d, a	WREG 与 f 作逻辑或运算	1	0001	00da	ffff	ffff	Z 和 N	1, 2
MOVF	f, d, a	传送 f	1	0101	00da	ffff	ffff	Z 和 N	1
MOVFF	f_s, f_d	将 f_s（源）送入（第 1 个字）	2	1100	ffff	ffff	ffff	无	
		f_d（目标）（第 2 个字）		1111	ffff	ffff	ffff		
MOVWF	f, a	将 WREG 中的内容送入 f	1	0110	111a	ffff	ffff	无	
MULWF	f, a	WREG 和 f 相乘	1	0000	001a	ffff	ffff	无	1, 2
NEGF	f, a	对 f 内容求补	1	0110	110a	ffff	ffff	C、DC、Z、OV 和 N	

表 40-2. 指令集 （续）

助记符和操作数		说明	指令周期数	16 位指令字				受影响的状态位	注
				MSb			LSb		
RLCF	f, d, a	对 f 执行带进位的循环左移	1	0011	01da	ffff	ffff	C、Z 和 N	1, 2
RLNCF	f, d, a	f 循环左移（不带进位）	1	0100	01da	ffff	ffff	Z 和 N	
RRCF	f, d, a	对 f 执行带进位的循环右移	1	0011	00da	ffff	ffff	C、Z 和 N	
RRNCF	f, d, a	f 循环右移（不带进位）	1	0100	00da	ffff	ffff	Z 和 N	
SETF	f, a	将 f 置 1	1	0110	00da	ffff	ffff	无	1, 2
SUBFWB	f, d, a	WREG 减去 f（带借位）	1	0101	01da	ffff	ffff	C、DC、Z、OV 和 N	
SUBWF	f, d, a	f 减去 WREG	1	0101	11da	ffff	ffff	C、DC、Z、OV 和 N	1, 2
SUBWFB	f, d, a	f 减去 WREG（带借位）	1	0101	10da	ffff	ffff	C、DC、Z、OV 和 N	
SWAPF	f, d, a	将 f 中的两个半字节进行交换	1	0011	10da	ffff	ffff	无	4
TSTFSZ	f, a	测试 f，为 0 则跳过	1（2 或 3）	0110	011a	ffff	ffff	无	1, 2
XORWF	f, d, a	WREG 与 f 作逻辑异或运算	1	0001	10da	ffff	ffff	Z 和 N	
面向位的操作类指令									
BCF	f, b, a	将 f 中的指定位清零	1	1001	bbba	ffff	ffff	无	1, 2
BSF	f, b, a	将 f 中的指定位置 1	1	1000	bbba	ffff	ffff	无	1, 2
BTFSC	f, b, a	对 f 中的指定位进行测试，如果为零则跳过	1（2 或 3）	1011	bbba	ffff	ffff	无	3, 4
BTFSS	f, b, a	对 f 中的指定位进行测试，如果为 1 则跳过	1（2 或 3）	1010	bbba	ffff	ffff	无	3, 4
BTG	f, b, a	将 f 中的指定位取反	1	0111	bbba	ffff	ffff	无	1, 2
控制类操作									
BC	n	如果有进位则转移	1 (2)	1110	0010	nnnn	nnnn	无	4
BN	n	如果为负则转移	1 (2)	1110	0110	nnnn	nnnn	无	
BNC	n	如果没有进位则转移	1 (2)	1110	0011	nnnn	nnnn	无	
BNN	n	如果不为负则转移	1 (2)	1110	0111	nnnn	nnnn	无	
BNOV	n	如果未溢出则转移	1 (2)	1110	0101	nnnn	nnnn	无	

表 40-2. 指令集 （续）

助记符和操作数		说明	指令周期数	16 位指令字				受影响的状态位	注
				MSb			LSb		
BNZ	n	如果不为零则转移	1 (2)	1110	0001	nnnn	nnnn	无	4
BOV	n	如果溢出则转移	1 (2)	1110	0100	nnnn	nnnn	无	
BRA	n	无条件转移	2	1101	0nnn	nnnn	nnnn	无	
BZ	n	如果为零则转移	1 (2)	1110	0000	nnnn	nnnn	无	
CALL	k, s	调用 子程序（第 1 个字）	2	1110	110s	kkkk	kkkk	无	
		第二个字		1111	kkkk	kkkk	kkkk		
CLRWDT	—	将看门狗定时器清零	1	0000	0000	0000	0100	$\overline{TO}$ 和 $\overline{PD}$	
DAW	—	十进制调整 WREG	1	0000	0000	0000	0111	C	
GOTO	k	跳转到地址（第 1 个字）	2	1110	1111	kkkk	kkkk	无	
		第二个字		1111	kkkk	kkkk	kkkk		
NOP	—	空操作	1	0000	0000	0000	0000	无	
NOP	—	空操作	1	1111	xxxx	xxxx	xxxx	无	
POP	—	将返回栈顶（TOS）的内容弹出	1	0000	0000	0000	0110	无	
PUSH	—	将返回栈顶（TOS）的内容压入	1	0000	0000	0000	0101	无	
RCALL	n	相对调用	2	1101	1nnn	nnnn	nnnn	无	
RESET		软件器件复位	1	0000	0000	1111	1111	全部	
RETFIE	s	使能从中断返回	2	0000	0000	0001	000s	GIE/GIEH， PEIE/GIEL	
RETLW	k	返回并将立即数传送到 WREG	2	0000	1100	kkkk	kkkk	无	
RETURN	s	从子程序返回	2	0000	0000	0001	001s	无	
SLEEP	—	进入待机模式	1	0000	0000	0000	0011	$\overline{TO}$,$\overline{PD}$	
立即数操作									

表 40-2. 指令集 （续）

助记符和操作数		说明	指令周期数	16 位指令字				受影响的状态位	注
				MSb			LSb		
ADDLW	k	立即数和 WREG 相加	1	0000	1111	kkkk	kkkk	C、DC、Z、OV 和 N	
ANDLW	k	立即数与 WREG 作逻辑与运算	1	0000	1011	kkkk	kkkk	Z 和 N	
IORLW	k	立即数与 WREG 作逻辑或运算	1	0000	1001	kkkk	kkkk	Z 和 N	
LFSR	f, k	将立即数（12 位）（第 2 个字）送入	2	1110	1110	00ff	kkkk	无	
		FSR（f）（第 1 个字）		1111	0000	kkkk	kkkk		
MOVLB	k	将立即数送入 BSR[3:0]	1	0000	0001	0000	kkkk	无	
MOVLW	k	将立即数送入 WREG	1	0000	1110	kkkk	kkkk	无	
MULLW	k	立即数与 WREG 相乘	1	0000	1101	kkkk	kkkk	无	
RETLW	k	返回并将立即数传送到 WREG	2	0000	1100	kkkk	kkkk	无	
SUBLW	k	立即数减去 WREG	1	0000	1000	kkkk	kkkk	C、DC、Z、OV 和 N	
XORLW	k	立即数与 WREG 作逻辑异或运算	1	0000	1010	kkkk	kkkk	Z 和 N	
数据存储器 ↔ 程序存储器操作									
TBLRD*		表读	2	0000	0000	0000	1000	无	
TBLRD*+		带后递增的表读		0000	0000	0000	1001	无	
TBLRD*-		带后递减的表读		0000	0000	0000	1010	无	
TBLRD+*		带预递增的表读		0000	0000	0000	1011	无	
TBLWT*		表写	2	0000	0000	0000	1100	无	
TBLWT*+		带后递增的表写		0000	0000	0000	1101	无	
TBLWT*-		带后递减的表写		0000	0000	0000	1110	无	
TBLWT+*		带预递增的表写		0000	0000	0000	1111	无	

注：

1. 当 PORT 寄存器修改自身时（例如，MOVF PORTB, 1, 0），修改时使用的值是引脚上的当前值。例如，如果将一引脚配置为输入，其对应数据锁存器中的值为 1，但此时若有外部器件将该引脚驱动为低电平，则被写回数据锁存器的数据值将是 0。
2. 当对 TMR0 寄存器执行该指令（且在适用情况下，d = 1）时，如果已分配了预分频器，该预分频器将被清零。
3. 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。
4. 一些指令为双字指令。这些指令的第二个字将作为 NOP 执行，除非指令的第一个字检索嵌入这 16 位的信息。这样可确保所有程序存储单元均具有有效指令。

40.1.1. 标准指令集详解

ADDLW	立即数与 W 相加			
语法：	ADDLW k			
操作数：	0 ≤ k ≤ 255			
操作：	(W) + k → W			
受影响的状态位：	N、OV、C、DC 和 Z			
指令编码：	0000	1111	kkkk	kkkk
说明：	将 W 寄存器的内容与 8 位立即数 k 相加，结果存入 W 寄存器。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 k	处理数据	写入 W

示例：	ADDLW	15h
执行指令前 W = 10h 执行指令后 W = 25h		

ADDWF	W 与 f 相加			
语法：	ADDWF f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(W) + (f) → 目标寄存器			
受影响的状态位：	N、OV、C、DC 和 Z			
指令编码：	0010	01da	ffff	ffff
说明：	W 与寄存器 f 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（默认）。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			

标准指令集详解（续）

ADDWF	W 与 f 相加			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

示例：	ADDWF	REG,	0, 0
执行指令前 W = 17h REG = 0C2h 执行指令后 W = 0D9h REG = 0C2h			

重要：所有指令都可以在指令助记符之前采用可选的标号参数，以用于符号寻址。如果使用标号，则指令格式变为：{标号}指令参数。

ADDWFC	W 和 f 进行带进位的相加			
语法：	ADDWFC f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(W) + (f) + (C) → 目标寄存器			
受影响的状态位：	N、OV、C、DC 和 Z			
指令编码：	0010	00da	ffff	ffff
说明：	将 W 的内容、进位标志位与数据存储单元 f 的内容相加。如果 d 为 0，结果存放到 W 寄存器。如果 d 为 1，结果存放到数据存储单元 f。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

示例：	ADDWFC	REG,	0 和 1

执行指令前
进位位 = 1
REG = 02h
W = 4Dh
执行指令后
进位位 = 0
REG = 02h
W = 50h

ANDLW	立即数与 W 作逻辑与运算			
语法：	ANDLW k			
操作数：	0 ≤ k ≤ 255			
操作：	(W) .AND. k → W			
受影响的状态位：	N 和 Z			
指令编码：	0000	1011	kkkk	kkkk
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。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 k	处理数据	写入 W

示例：	ANDLW	05Fh

执行指令前
W = A3h
执行指令后
W = 03h

ANDWF	W 和 f 作逻辑与运算			
语法：	ANDWF f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(W) .AND.(f) → 目标寄存器			
受影响的状态位：	N 和 Z			
指令编码：	0001	01da	ffff	ffff
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（默认）。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4

译码	读取寄存器 f	处理数据	写入目标

示例：	ANDWF	REG,	0, 0
执行指令前 W = 17h REG = C2h 执行指令后 W = 02h REG = C2h			

BC	如果有进位则转移			
语法：	BC n			
操作数：	-128 ≤ n ≤ 127			
操作：	如果进位位为 1 (PC) + 2 + 2n → PC			
受影响的状态位：	无			
指令编码：	1110	0010	nnnn	nnnn
说明：	如果进位位为 1，则程序将转移。 二进制补码“2n”与 PC 的内容相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。			
指令字数：	1			
指令周期数：	1（2）			

Q 周期活动： 如果跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作
如果未跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	空操作

示例：	HERE	BC	5
执行指令前 PC = 地址（HERE） 执行指令后 如果进位位 = 1； PC = 地址（HERE + 12） 如果进位位 = 0； PC = 地址（HERE + 2）			

BCF	将 f 中的指定位清零
语法：	BCF f, b {,a}
操作数：	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]
操作：	0 → f<b>

标准指令集详解（续）

BCF	将 f 中的指定位清零			
受影响的状态位：	无			
指令编码：	1001	bbba	ffff	ffff
说明：	将寄存器 f 中的 bit b 清零。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入寄存器 f

示例：	BCF	FLAG_REG, 7, 0
执行指令前 FLAG_REG = C7h 执行指令后 FLAG_REG = 47h		

BN	如果为负则转移			
语法：	BN n			
操作数：	-128 ≤ n ≤ 127			
操作：	如果负标志位为 1 (PC) + 2 + 2n → PC			
受影响的状态位：	无			
指令编码：	1110	0110	nnnn	nnnn
说明：	如果负标志位为 1，则程序将转移。 二进制补码“2n”与 PC 的内容相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。			
指令字数：	1			
指令周期数：	1（2）			

Q 周期活动： 如果跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作
如果未跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	空操作

示例：	HERE	BN	Jump
执行指令前 PC = 地址（HERE） 执行指令后 如果负标志位 = 1；PC = 地址（Jump） 如果负标志位 = 0；PC = 地址（HERE + 2）			

BNC	如果没有进位则转移			
语法：	BNC n			
操作数：	-128 ≤ n ≤ 127			
操作：	如果进位位为 0 (PC) + 2 + 2n → PC			
受影响的状态位：	无			
指令编码：	1110	0011	nnnn	nnnn
说明：	如果进位位为 0，则程序将转移。 二进制补码“2n”与 PC 的内容相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。			
指令字数：	1			
指令周期数：	1（2）			

Q 周期活动： 如果跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作
如果未跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	空操作

示例：	HERE	BNC	Jump
执行指令前 PC = 地址（HERE） 执行指令后 如果进位位 = 0； PC = 地址（Jump） 如果进位位 = 1； PC = 地址（HERE + 2）			

BNN	如果不为负则转移			
语法：	BNN n			
操作数：	-128 ≤ n ≤ 127			
操作：	如果负标志位为 0 (PC) + 2 + 2n → PC			
受影响的状态位：	无			
指令编码：	1110	0111	nnnn	nnnn

标准指令集详解（续）

BNN	如果不为负则转移			
说明：	如果负标志位为 0，则程序将转移。 二进制补码“2n”与 PC 的内容相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。			
指令字数：	1			
指令周期数：	1（2）			

Q 周期活动： 如果跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作
如果未跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	空操作

示例：	HERE	BNN	Jump
执行指令前 PC = 地址（HERE） 执行指令后 如果负标志位 = 0； PC = 地址（Jump） 如果负标志位 = 1； PC = 地址（HERE + 2）			

BNOV	如果未溢出则转移			
语法：	BNOV n			
操作数：	-128 ≤ n ≤ 127			
操作：	如果溢出位为 0 (PC) + 2 + 2n → PC			
受影响的状态位：	无			
指令编码：	1110	0101	nnnn	nnnn
说明：	如果溢出位为 0，则程序将转移。 二进制补码“2n”与 PC 的内容相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。			
指令字数：	1			
指令周期数：	1（2）			

Q 周期活动： 如果跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作
如果未跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	空操作

示例：	HERE	BNOV	Jump
执行指令前 PC = 地址（HERE） 执行指令后 如果溢出位 = 0； PC = 地址（Jump） 如果溢出位 = 1； PC = 地址（HERE + 2）			

BNZ	如果不为零则转移			
语法：	BNZ n			
操作数：	-128 ≤ n ≤ 127			
操作：	如果全零位为 0 (PC) + 2 + 2n → PC			
受影响的状态位：	无			
指令编码：	1110	0001	nnnn	nnnn
说明：	如果全零位为 0，则程序将转移。 二进制补码“2n”与 PC 的内容相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。			
指令字数：	1			
指令周期数：	1（2）			

Q 周期活动： 如果跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作
如果未跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	空操作

示例：	HERE	BNZ	Jump
执行指令前 PC = 地址（HERE） 执行指令后 如果全零位 = 0； PC = 地址（Jump） 如果全零位 = 1； PC = 地址（HERE + 2）			

BRA	无条件转移			
语法：	BRA n			
操作数：	-1024 ≤ n ≤ 1023			
操作：	(PC) + 2 + 2n → PC			
受影响的状态位：	无			
指令编码：	1101	0nnn	nnnn	nnnn
说明：	二进制补码“2n”与 PC 的内容相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。			
指令字数：	1			

标准指令集详解（续）

BRA	无条件转移			
指令周期数：	2			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作

示例：	HERE	BRA	Jump
执行指令前 PC = 地址（HERE） 执行指令后 PC = 地址（Jump）			

BSF	将 f 中的指定位置 1			
语法：	BSF f, b {,a}			
操作数：	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]			
操作：	1 → f<b>			
受影响的状态位：	无			
指令编码：	1000	bbba	ffff	ffff
说明：	将寄存器 f 中的 bit b 置 1。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入寄存器 f

示例：	BSF	FLAG_REG, 7, 1
执行指令前 FLAG_REG = 0Ah 执行指令后 FLAG_REG = 8Ah		

BTFSC	测试文件中的某位，为 0 则跳过
语法：	BTFSC f, b {,a}
操作数：	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]

标准指令集详解（续）

BTFSC	测试文件中的某位，为 0 则跳过			
操作：	如果(f<b>) = 0，则跳过			
受影响的状态位：	无			
指令编码：	1011	bbba	ffff	ffff
说明：	如果寄存器 f 的 bit b 为 0，则跳过下一条指令。如果 bit b 为 0，则丢弃当前指令执行期间所取的下一条指令，代之执行一条 NOP 指令，使之成为一条双周期指令。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1（2） **注：**如果跳过且后跟一条双字指令，则为三个周期。			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	空操作

如果跳过：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
如果跳过且后跟一条双字指令：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：	HERE FALSE TRUE	BTFSC : :	FLAG, 1, 0
执行指令前 PC = 地址（HERE） 执行指令后 如果 FLAG<1> = 0； PC = 地址（TRUE） 如果 FLAG<1> = 1； PC = 地址（FALSE）			

BTFSS	测试文件中的某位，为 1 则跳过			
语法：	BTFSS f, b {,a}			
操作数：	0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1]			
操作：	如果(f<b>) = 1，则跳过			
受影响的状态位：	无			
指令编码：	1010	bbba	ffff	ffff

标准指令集详解（续）

BTFSS	测试文件中的某位，为 1 则跳过			
说明：	如果寄存器 f 的 bit b 为 1，则跳过下一条指令。如果 bit b 为 1，则丢弃当前指令执行期间所取的下一条指令，代之执行一条 NOP 指令，使之成为一条双周期指令。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1（2） **注：**如果跳过且后跟一条双字指令，则为三个周期。			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	空操作

如果跳过：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
如果跳过且后跟一条双字指令：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：	HERE FALSE TRUE	BTFSS : :	FLAG, 1, 0
执行指令前 PC = 地址（HERE） 执行指令后 如果 FLAG<1> = 0； PC = 地址（FALSE） 如果 FLAG<1> = 1； PC = 地址（TRUE）			

BTG	将 f 中的指定位取反			
语法：	BTG f, b {,a}			
操作数：	0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1]			
操作：	$(\overline{f<b>}) \rightarrow f<b>$			
受影响的状态位：	无			
指令编码：	0111	bbba	ffff	ffff
说明：	将数据存储单元 f 中的 bit b 取反。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			

标准指令集详解（续）

BTG	将 f 中的指定位取反			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入寄存器 f

示例：	`BTG`	`PORTC,`	`4, 0`
指令执行前： PORTC = 0111 0101 [75h] 指令执行后： PORTC = 0110 0101 [65h]			

BOV	如果溢出则转移			
语法：	BOV n			
操作数：	-128 ≤ n ≤ 127			
操作：	如果溢出位为 1 (PC) + 2 + 2n → PC			
受影响的状态位：	无			
指令编码：	1110	0100	nnnn	nnnn
说明：	如果溢出位为 1，则程序将转移。 二进制补码“2n”与 PC 的内容相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。			
指令字数：	1			
指令周期数：	1（2）			

Q 周期活动： 如果跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作
如果未跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	空操作

示例：	`HERE`	`BOV`	`Jump`
执行指令前 PC = 地址（HERE） 执行指令后 如果溢出位 = 1； PC = 地址（Jump） 如果溢出位 = 0； PC = 地址（HERE + 2）			

BZ	如果为零则转移			
语法：	BZ n			
操作数：	-128 ≤ n ≤ 127			
操作：	如果全零位为 1 (PC) + 2 + 2n → PC			
受影响的状态位：	无			
指令编码：	1110	0000	nnnn	nnnn
说明：	如果全零位为 1，则程序将转移。 二进制补码“2n”与 PC 的内容相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。			
指令字数：	1			
指令周期数：	1（2）			

Q 周期活动： 如果跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	写入 PC
空操作	空操作	空操作	空操作
如果未跳转：			
Q1	Q2	Q3	Q4
译码	读取立即数 n	处理数据	空操作

示例：	HERE	BZ	Jump
执行指令前 PC = 地址（HERE） 执行指令后 如果全零位 = 1； PC = 地址（Jump） 如果全零位 = 0； PC = 地址（HERE + 2）			

CALL	调用子程序			
语法：	CALL k {,s}			
操作数：	0 ≤ k ≤ 1048575 s ∈ [0,1]			
操作：	(PC) + 4 → TOS, k → PC<20:1>, 如果 s = 1 (W) → WS, (Status) → STATUSS, (BSR) → BSRS			
受影响的状态位：	无			
指令编码： 第 1 个字（k<7:0>） 第 2 个字（k<19:8>）	1110 1111	110s k_{19}kkk	$k_7$kkk kkkk	kkkk$_0$ kkkk$_8$

标准指令集详解（续）

CALL	调用子程序			
说明：	在整个 2 MB 存储器范围内调用子程序。首先，将返回地址（PC + 4）压入返回堆栈。如果 s = 1，则 W、Status 和 BSR 寄存器的内容也会被压入各自的影子寄存器 WS、STATUSS 和 BSRS。如果 s = 0，则不进行更新（默认）。然后，将 20 位值 k 装入 PC<20:1>。CALL 是一条双周期指令。			
指令字数：	2			
指令周期数：	2			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 k<7:0>，	将 PC 压入堆栈	读取立即数 k<19:8>，写入 PC
空操作	空操作	空操作	空操作

示例：	HERE	CALL	THERE, 1
执行指令前 PC = 地址（HERE） 执行指令后 PC = 地址（THERE） TOS = 地址（HERE + 4） WS = W BSRS = BSR STATUSS = Status			

CLRF	将 f 清零			
语法：	CLRF f {,a}			
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]			
操作：	000h → f 1 → Z			
受影响的状态位：	Z			
指令编码：	0110	101a	ffff	ffff
说明：	清零指定寄存器的内容。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入寄存器 f

示例：	CLRF	FLAG_REG, 1

执行指令前 FLAG_REG = 5Ah 执行指令后 FLAG_REG = 00h

CLRWDT	将看门狗定时器清零			
语法：	CLRWDT			
操作数：	无			
操作：	000h → WDT， 000h → WDT 后分频器， 1 →$\overline{TO}$ 1 →$\overline{PD}$			
受影响的状态位：	$\overline{TO}$ 和 $\overline{PD}$			
指令编码：	0000	0000	0000	0100
说明：	CLRWDT 指令复位看门狗定时器及其后分频器。状态位 $\overline{TO}$ 和 $\overline{PD}$ 均被置 1。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	空操作	处理数据	空操作

示例：	`CLRWDT`
执行指令前 WDT 计数器 = ? 执行指令后 WDT 计数器 = 00h WDT 后分频器 = 0 $\overline{TO}$= 1 $\overline{PD}$= 1	

COMF	f 取反			
语法：	COMF f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	($\bar{f}$) → 目标寄存器			
受影响的状态位：	N 和 Z			
指令编码：	0001	11da	ffff	ffff
说明：	将寄存器 f 的内容取反。如果 d 为 0，结果存入 W 寄存器。如果 d 为 1，结果存回寄存器 f（默认）。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

示例：	COMF	REG,	0, 0
执行指令前 REG = 13h 执行指令后 REG = 13h W = ECh			

CPFSEQ	**比较 f 和 W，如果 f = W 则跳过**			
语法：	CPFSEQ f {,a}			
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]			
操作：	(f) – (W)，如果(f) = (W)（无符号比较）则跳过			
受影响的状态位：	无			
指令编码：	0110	001a	ffff	ffff
说明：	通过执行无符号减法将数据存储单元 f 的内容与 W 的内容进行比较。 如果 f = W，则丢弃所取指令，代之执行一条 NOP 指令，使之成为一条双周期指令。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1（2） **注：**如果跳过且后跟一条双字指令，则为三个周期。			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	空操作

如果跳过：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
如果跳过且后跟一条双字指令：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：	

```
HERE     CPFSEQ REG, 0
NEQUAL   :
EQUAL    :
```

执行指令前
PC 地址 = HERE
W = ?
REG = ?
执行指令后
如果 REG = W；
PC = 地址（EQUAL）
如果 REG ≠ ；
PC = 地址（NEQUAL）

CPFSGT	比较 f 和 W，如果 f > W 则跳过			
语法：	CPFSGT f {,a}			
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]			
操作：	(f) – (W)，如果(f) > (W)（无符号比较）则跳过			
受影响的状态位：	无			
指令编码：	0110	010a	ffff	ffff
说明：	通过执行无符号减法将数据存储单元 f 的内容与 W 的内容进行比较。 如果 f 的内容大于 WREG 的内容，则丢弃所取指令，代之执行一条 NOP 指令，使之成为一条双周期指令。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1（2） **注：**如果跳过且后跟一条双字指令，则为三个周期。			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	空操作

如果跳过：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
如果跳过且后跟一条双字指令：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：	

```
HERE      CPFSGT REG, 0
NGREATER  :
GREATER   :
```

执行指令前
PC = 地址（HERE）
W = ?
执行指令后
如果 REG > W；
PC = 地址（GREATER）
如果 REG ≤ W；
PC = 地址（NGREATER）

CPFSLT	比较 f 和 W，如果 f < W 则跳过			
语法：	CPFSLT f {,a}			
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]			
操作：	(f) – (W)，如果(f) < (W)（无符号比较）则跳过			
受影响的状态位：	无			
指令编码：	0110	000a	ffff	ffff
说明：	通过执行无符号减法将数据存储单元 f 的内容与 W 的内容进行比较。 如果 f 的内容小于 W 的内容，则丢弃所取指令，代之执行一条 NOP 指令，使之成为一条双周期指令。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。			
指令字数：	1			
指令周期数：	1（2） **注：**如果跳过且后跟一条双字指令，则为三个周期。			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	空操作

如果跳过：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
如果跳过且后跟一条双字指令：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：

```
HERE    CPFSLT REG, 1
NLESS   :
LESS    :
```

执行指令前
PC = 地址（HERE）
W = ?
执行指令后
如果 REG < W；
PC = 地址（LESS）
如果 REG ≥ W；
PC = 地址（NLESS）

DAW	十进制调整 W 寄存器
语法：	DAW

标准指令集详解（续）

DAW	十进制调整 W 寄存器			
操作数：	无			
操作：	如果[W<3:0> > 9]或[DC = 1]，则 (W<3:0>) + 6 → W<3:0>； 否则(W<3:0>) → W<3:0>； 如果[W<7:4> + DC > 9]或[C = 1]，则 (W<7:4>) + 6 + DC → W<7:4>； 否则(W<7:4>) + DC → W<7:4>			
受影响的状态位：	C			
指令编码：	0000	0000	0000	0111
说明：	DAW 将调整 W 中的 8 位值（之前由两个打包的 BCD 格式的变量相加而得），并产生正确打包的 BCD 结果。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 W	处理数据	写入 W

例 1：	DAW
执行指令前 W = A5h C = 0 DC = 0 执行指令后 W = 05h C = 1 DC = 0 例 2： 执行指令前 W = CEh C = 0 DC = 0 执行指令后 W = 34h C = 1 DC = 0	

DECF	f 递减 1			
语法：	DECF f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(f) – 1 → 目标寄存器			
受影响的状态位：	C、DC、N、OV 和 Z			
指令编码：	0000	01da	ffff	ffff

标准指令集详解（续）

DECF	f 递减 1			
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（默认）。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

示例：	DECF CNT,	1, 0
执行指令前 CNT = 01h Z = 0 执行指令后 CNT = 00h Z = 1		

DECFSZ	f 递减 1，为 0 则跳过			
语法：	DECFSZ f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(f) – 1 → 目标寄存器，如果结果 = 0 则跳过			
受影响的状态位：	无			
指令编码：	0010	11da	ffff	ffff
说明：	将寄存器 f 的内容递减 1。如果 d 为 0，结果存入 W 寄存器。如果 d 为 1，结果存回寄存器 f（默认）。 如果结果为 0，则丢弃已取的下一条指令，代之执行一条 NOP 指令，使之成为一条双周期指令。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1（2） **注：**如果跳过且后跟一条双字指令，则为三个周期。			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

如果跳过：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
如果跳过且后跟一条双字指令：			

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：	
	HERE DECFSZ CNT, 1, 1 GOTO LOOP CONTINUE

执行指令前
PC = 地址（HERE）
执行指令后
CNT = CNT - 1
如果 CNT = 0；
PC = 地址（CONTINUE）
如果 CNT ≠ 0；
PC = 地址（HERE + 2）

DCFSNZ	f 递减 1，不为 0 则跳过			
语法：	DCFSNZ f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(f) – 1 → 目标寄存器，如果结果 ≠ 0 则跳过			
受影响的状态位：	无			
指令编码：	0100	11da	ffff	ffff
说明：	将寄存器 f 的内容递减 1。如果 d 为 0，结果存入 W 寄存器。如果 d 为 1，结果存回寄存器 f（默认）。 如果结果不为 0，则丢弃已取的下一条指令，代之执行一条 NOP 指令，使之成为一条双周期指令。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1（2） **注：**如果跳过且后跟一条双字指令，则为三个周期。			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

如果跳过：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作

如果跳过且后跟一条双字指令：

Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：	
	HERE DCFSNZ TEMP, 1, 0 ZERO : NZERO :

执行指令前 TEMP = ? 执行指令后 TEMP = TEMP – 1， 如果 TEMP = 0； PC = 地址（ZERO） 如果 TEMP ≠ 0； PC = 地址（NZERO）

GOTO	无条件转移			
语法：	GOTO k			
操作数：	0 ≤ k ≤ 1048575			
操作：	k → PC<20:1>			
受影响的状态位：	无			
指令编码： 第 1 个字（k<7:0>） 第 2 个字（k<19:8>）	1110 1111	1111 k_{19}kkk	$k_7$kkk kkkk	kkkk$_0$ kkkk$_8$
说明：	GOTO 允许在整个 2 MB 存储器范围内的任意位置执行无条件转移。20 位值 k 装入 PC<20:1>。GOTO 始终是一条双周期指令。			
指令字数：	2			
指令周期数：	2			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 k<7:0>，	空操作	读取立即数 k<19:8>，写入 PC
空操作	空操作	空操作	空操作

示例：	GOTO THERE
执行指令后 PC = 地址（THERE）	

INCF	f 递增 1			
语法：	INCF f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(f) + 1 → 目标寄存器			
受影响的状态位：	C、DC、N、OV 和 Z			
指令编码：	0010	10da	ffff	ffff
说明：	将寄存器 f 的内容递增 1。如果 d 为 0，结果存入 W 寄存器。如果 d 为 1，结果存回寄存器 f（默认）。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

示例：	INCF	CNT,	1, 0
执行指令前 CNT = FFh Z = 0 C = ? DC = ? 执行指令后 CNT = 00h Z = 1 C = 1 DC = 1			

INCFSZ	**f 递增 1，为 0 则跳过**			
语法：	INCFSZ f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(f) + 1 → 目标寄存器， 如果结果 = 0，则跳过			
受影响的状态位：	无			
指令编码：	0011	11da	ffff	ffff
说明：	将寄存器 f 的内容递增 1。如果 d 为 0，结果存入 W 寄存器。如果 d 为 1，结果存回寄存器 f（默认）。 如果结果为 0，则丢弃已取的下一条指令，代之执行一条 NOP 指令，使之成为一条双周期指令。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1（2） **注：**如果跳过且后跟一条双字指令，则为三个周期。			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

如果跳过：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
如果跳过且后跟一条双字指令：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：	
	HERE INCFSZ CNT, 1, 0 NZERO : ZERO :
执行指令前 PC = 地址（HERE） 执行指令后 CNT = CNT + 1 如果 CNT = 0； PC = 地址（ZERO） 如果 CNT ≠ 0； PC = 地址（NZERO）	

INFSNZ	f 递增 1，不为 0 则跳过			
语法：	INFSNZ f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(f) + 1 → 目标寄存器，如果结果 ≠ 0 则跳过			
受影响的状态位：	无			
指令编码：	0100	10da	ffff	ffff
说明：	将寄存器 f 的内容递增 1。如果 d 为 0，结果存入 W 寄存器。如果 d 为 1，结果存回寄存器 f（默认）。 如果结果不为 0，则丢弃已取的下一条指令，代之执行一条 NOP 指令，使之成为一条双周期指令。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1（2） **注：**如果跳过且后跟一条双字指令，则为三个周期。			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

如果跳过：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
如果跳过且后跟一条双字指令：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：	
	HERE INFSNZ REG, 1, 0 ZERO NZERO

执行指令前 PC = 地址（HERE） 执行指令后 REG = REG + 1 如果 REG ≠ 0； PC = 地址（NZERO） 如果 REG = 0； PC = 地址（ZERO）

IORLW	立即数与 W 作逻辑或运算			
语法：	IORLW k			
操作数：	0 ≤ k ≤ 255			
操作：	(W) .OR. k → W			
受影响的状态位：	N 和 Z			
指令编码：	0000	1001	kkkk	kkkk
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。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 k	处理数据	写入 W

示例：	IORLW	35h
执行指令前 W = 9Ah 执行指令后 W = BFh		

IORWF	W 与 f 作逻辑或运算			
语法：	IORWF f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(W) .OR.(f) → 目标寄存器			
受影响的状态位：	N 和 Z			
指令编码：	0001	00da	ffff	ffff
说明：	W 与寄存器 f 作逻辑或运算。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（默认）。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

示例：	`IORWF  RESULT, 0, 1`
执行指令前 RESULT = 13h W = 91h 执行指令后 RESULT = 13h W = 93h	

LFSR	装入 FSR			
语法：	LFSR f, k			
操作数：	0 ≤ f ≤ 2 0 ≤ k ≤ 4095			
操作：	k → FSRf			
受影响的状态位：	无			
指令编码：	1110 1111	1110 0000	00ff $k_7$kkk	k_{11}kkk kkkk
说明：	将 12 位立即数 k 装入 f 指向的文件选择寄存器。			
指令字数：	2			
指令周期数：	2			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 k MSB	处理数据	将立即数 k MSB 写入 FSRfH
译码	读取立即数 k LSB	处理数据	将立即数 k 写入 FSRfL

示例：	`LFSR     2, 3ABh`
执行指令后 FSR2H = 03h FSR2L = ABh	

MOVF	传送 f			
语法：	MOVF f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	f → 目标寄存器			
受影响的状态位：	N 和 Z			
指令编码：	0101	00da	ffff	ffff
说明：	根据 d 的状态，将寄存器 f 的内容送入目标寄存器。如果 d 为 0，结果存入 W 寄存器。如果 d 为 1，结果存回寄存器 f（默认）。存储单元 f 可以是 256 字节存储区中的任意位置。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入 W

示例：	MOVF REG, 0, 0
执行指令前 REG = 22h W = FFh 执行指令后 REG = 22h W = 22h	

MOVFF	将 f 中的内容送入 f			
语法：	MOVFF f_s,f_d			
操作数：	0 ≤ f_s ≤ 4095 0 ≤ f_d ≤ 4095			
操作：	(f_s) → f_d			
受影响的状态位：	无			
指令编码： 第 1 个字（源） 第 2 个字（目标）	1100 1111	ffff ffff	ffff ffff	$ffff_s$ $ffff_d$
说明：	将源寄存器 f_s 中的内容送入目标寄存器 f_d。源寄存器 f_s 的存储单元可以是 4096 字节数据空间（000h 至 FFFh）内的任意位置，目标寄存器 f_d 的存储单元可以是 000h 至 FFFh 范围内的任意位置。 源寄存器或目标寄存器可以是 W 寄存器（这是一种有用的特殊情况）。 MOVFF 对于将数据存储单元中的内容传输到外设寄存器（如发送缓冲区或 I/O 端口）特别有用。 MOVFF 指令不能使用 PCL、TOSU、TOSH 或 TOSL 作为目标寄存器。			
指令字数：	2			
指令周期数：	2（3）			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f（源）	处理数据	空操作
译码	空操作 无假读	空操作	写入寄存器 f（目标）

示例：	MOVFF REG1, REG2
执行指令前 REG1 = 33h REG2 = 11h 执行指令后 REG1 = 33h REG2 = 33h	

MOVLB	将立即数送入 BSR 中的低半字节
语法：	MOVLB k
操作数：	0 ≤ k ≤ 255

标准指令集详解（续）

MOVLB	将立即数送入 BSR 中的低半字节			
操作：	k → BSR			
受影响的状态位：	无			
指令编码：	0000	0001	kkkk	kkkk
说明：	将 8 位立即数 k 装入存储区选择寄存器（BSR）。无论 k_7:k_4 的值如何，BSR<7:4>的值始终保持为 0。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 k	处理数据	将立即数 k 写入 BSR

示例：	`MOVLB`	`5`
执行指令前 BSR 寄存器 = 02h 执行指令后 BSR 寄存器 = 05h		

MOVLW	将立即数传送到 W			
语法：	MOVLW k			
操作数：	0 ≤ k ≤ 255			
操作：	k → W			
受影响的状态位：	无			
指令编码：	0000	1110	kkkk	kkkk
说明：	将 8 位立即数 k 装入 W。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 k	处理数据	写入 W

示例：	`MOVLW`	`5Ah`
执行指令后 W = 5Ah		

MOVWF	将 W 的内容传送到 f			
语法：	MOVWF f {,a}			
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]			
操作：	(W) → f			
受影响的状态位：	无			
指令编码：	0110	111a	ffff	ffff

标准指令集详解（续）

MOVWF	将 W 的内容传送到 f			
说明：	将 W 的数据传送到寄存器 f。存储单元 f 可以是 256 字节存储区中的任意位置。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入寄存器 f

示例：	MOVWF	REG, 0
执行指令前 W = 4Fh REG = FFh 执行指令后 W = 4Fh REG = 4Fh		

MULLW	立即数与 W 的内容相乘			
语法：	MULLW k			
操作数：	0 ≤ k ≤ 255			
操作：	(W) x k → PRODH:PRODL			
受影响的状态位：	无			
指令编码：	0000	1101	kkkk	kkkk
说明：	将 W 的内容与 8 位立即数 k 作无符号乘法运算。16 位结果存入 PRODH:PRODL 寄存器对。PRODH 包含高字节。W 不变。 没有任何状态标志受到影响。 请注意，在此运算中既不能溢出也不能进位。结果可能为零，但无法检测到。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 k	处理数据	写入寄存器 PRODH:PRODL

示例：	MULLW 0C4h

执行指令前
W = E2h
PRODH = ?
PRODL = ?
执行指令后
W = E2h
PRODH = ADh
PRODL = 08h

MULWF	W 和 f 相乘			
语法：	MULWF f {,a}			
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]			
操作：	(W) x (f) → PRODH:PRODL			
受影响的状态位：	无			
指令编码：	0000	001a	ffff	ffff
说明：	将 W 的内容与寄存器文件存储单元 f 作无符号乘法运算。16 位结果存入 PRODH:PRODL 寄存器对。PRODH 包含高字节。W 和 f 不变。 没有任何状态标志受到影响。 请注意，在此运算中既不能溢出也不能进位。结果可能为零，但无法检测到。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入寄存器 PRODH:PRODL

示例：

```
MULWF   REG, 1
```

执行指令前
W = C4h
REG = B5h
PRODH = ?
PRODL = ?
执行指令后
W = C4h
REG = B5h
PRODH = 8Ah
PRODL = 94h

NEGF	对 f 内容求补
语法：	NEGF f {,a}
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]
操作：	(f̄) + 1 → f

标准指令集详解（续）

NEGF	对 f 内容求补			
受影响的状态位：	N、OV、C、DC 和 Z			
指令编码：	0110	110a	ffff	ffff
说明：	使用二进制补码对存储单元 f 求补。结果存入数据存储单元 f。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入寄存器 f

示例：	NEGF REG, 1
执行指令前 REG = 0011 1010 [3Ah] 执行指令后 REG = 1100 0110 [C6h]	

NOP	空操作			
语法：	NOP			
操作数：	无			
操作：	空操作			
受影响的状态位：	无			
指令编码：	0000 1111	0000 xxxx	0000 xxxx	0000 xxxx
说明：	空操作。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	空操作	空操作	空操作

示例：
无。

POP	将返回栈顶的内容弹出			
语法：	POP			
操作数：	无			
操作：	(TOS) → 位桶			
受影响的状态位：	无			
指令编码：	0000	0000	0000	0110

标准指令集详解（续）

POP	将返回栈顶的内容弹出			
说明：	TOS 值从返回堆栈弹出，然后被丢弃。TOS 值随后变为前一个压入返回堆栈的值。 该指令旨在帮助用户正确地管理返回堆栈以集成软件堆栈。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	空操作	弹出 TOS 值	空操作

示例：	POP GOTO	NEW
执行指令前 TOS = 0031A2h 堆栈（下移 1 级）= 014332h 执行指令后 TOS = 014332h PC = NEW		

PUSH	将值压入返回栈顶			
语法：	PUSH			
操作数：	无			
操作：	(PC + 2) → TOS			
受影响的状态位：	无			
指令编码：	0000	0000	0000	0101
说明：	将 PC + 2 的值压入返回栈顶。前一个 TOS 值在堆栈中下移。 该指令可通过以下方式实现软件堆栈：修改 TOS 值，然后将其压入返回堆栈。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	将 PC + 2 的值压入返回堆栈	空操作	空操作

示例：	PUSH
执行指令前 TOS = 345Ah PC = 0124h 执行指令后 PC = 0126h TOS = 0126h 堆栈（下移 1 级）= 345Ah	

RCALL	相对调用
语法：	RCALL n
操作数：	-1024 ≤ n ≤ 1023

标准指令集详解（续）

RCALL	相对调用			
操作：	(PC) + 2 → TOS， (PC) + 2 + 2n → PC			
受影响的状态位：	无			
指令编码：	1101	1nnn	nnnn	nnnn
说明：	调用子程序（最多跳转到距当前存储单元 1K 的位置）。首先，将返回地址（PC + 2）压入堆栈。然后，将二进制补码“2n”与 PC 的内容相加。由于 PC 将递增以便取出下一条指令，所以新地址将为 PC + 2 + 2n。该指令为一条双周期指令。			
指令字数：	1			
指令周期数：	2			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 n 将 PC 压入堆栈	处理数据	写入 PC
空操作	空操作	空操作	空操作

示例：	HERE	RCALL	Jump
执行指令前 PC = 地址（HERE） 执行指令后 PC = 地址（Jump） TOS = 地址（HERE + 2）			

RESET	复位			
语法：	RESET			
操作数：	无			
操作：	复位所有受 $\overline{\text{MCLR}}$ 复位影响的寄存器和标志位。			
受影响的状态位：	全部			
指令编码：	0000	0000	1111	1111
说明：	该指令可实现用软件执行 $\overline{\text{MCLR}}$ 复位。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	启动复位	空操作	空操作

示例：	RESET
执行指令后 寄存器 = 复位值 标志* = 复位值	

RETFIE	从中断返回
语法：	RETFIE {s}

标准指令集详解（续）

RETFIE	从中断返回			
操作数：	s ∈ [0,1]			
操作：	(TOS) → PC， 1 → GIE/GIEH 或 PEIE/GIEL， 如果 s = 1 (WS) → W， (STATUSS) → Status， (BSRS) → BSR， PCLATU 和 PCLATH 不变。			
受影响的状态位：	GIE/GIEH 和 PEIE/GIEL。			
指令编码：	0000	0000	0001	000s
说明：	从中断返回。执行出栈操作，将栈顶（TOS）的内容装入 PC。通过将高优先级或低优先级全局中断允许位置 1 来允许中断。如果 s = 1，则将影子寄存器 WS、STATUSS 和 BSRS 的内容装入其相应的寄存器 W、Status 和 BSR 中。如果 s = 0，则不更新这些寄存器（默认）。			
指令字数：	1			
指令周期数：	2			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	空操作	空操作	将 PC 的值从堆栈弹出 将 GIEH 或 GIEL 置 1
空操作	空操作	空操作	空操作

示例：	`RETFIE 1`
中断后 PC = TOS W = WS BSR = BSRS Status = STATUSS GIE/GIEH 和 PEIE/GIEL = 1	

RETLW	将立即数返回 W			
语法：	RETLW k			
操作数：	0 ≤ k ≤ 255			
操作：	k → W， (TOS) → PC， PCLATU 和 PCLATH 不变			
受影响的状态位：	无			
指令编码：	0000	1100	kkkk	kkkk
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。高地址锁存器（PCLATH）保持不变。			
指令字数：	1			
指令周期数：	2			

Q 周期活动：		

Q1	Q2	Q3	Q4
译码	读取立即数 k	处理数据	将 PC 的值从堆栈弹出，写入 W
空操作	空操作	空操作	空操作

示例：

```
      CALL TABLE            ;  contains table
; offset value
; W now has
; table value
   :
TABLE
ADDWF PCL             ; W = offset
RETLW k0             ; Begin table
RETLW k1             ;
:
:
RETLW kn             ; End of table
```

执行指令前
W = 07h

执行指令后

W = kn 的值

RETURN	从子程序返回			
语法：	RETURN {s}			
操作数：	s ∈ [0,1]			
操作：	(TOS) → PC， 如果 s = 1 (WS) → W， (STATUSS) → Status， (BSRS) → BSR， PCLATU 和 PCLATH 不变			
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。如果 s = 1，则将影子寄存器 WS、STATUSS 和 BSRS 的内容装入其相应的寄存器 W、Status 和 BSR 中。如果 s = 0，则不更新这些寄存器（默认）。			
指令字数：	1			
指令周期数：	2			

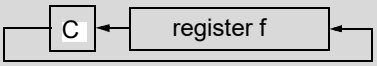
Q 周期活动：			
Q1	Q2	Q3	Q4
译码	空操作	处理数据	将 PC 的值从堆栈弹出
空操作	空操作	空操作	空操作

示例：	RETURN

指令执行后：
PC = TOS

RLCF	对 f 执行带进位的循环左移
语法：	RLCF f {,d {,a}}

标准指令集详解（续）

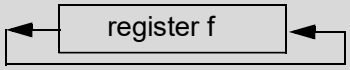
RLCF	对 f 执行带进位的循环左移			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(f<n>) → dest<n + 1>， (f<7>) → C， (C) → dest<0>			
受影响的状态位：	C、N 和 Z			
指令编码：	0011	01da	ffff	ffff
说明：	将寄存器 f 的内容连同进位标志位一起循环左移 1 位。如果 d 为 0，结果存入 W 寄存器。如果 d 为 1，结果存回寄存器 f（默认）。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。 			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

示例：	RLCF	REG, 0, 0
执行指令前 REG = 1110 0110 C = 0 执行指令后 REG = 1110 0110 W = 1100 1100 C = 1		

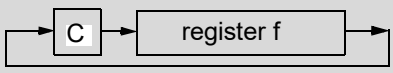
RLNCF	f 循环左移（不带进位）			
语法：	RLNCF f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(f<n>) → dest<n + 1>， (f<7>) → dest<0>			
受影响的状态位：	N 和 Z			
指令编码：	0100	01da	ffff	ffff

标准指令集详解 （续）

RLNCF	f 循环左移（不带进位）			
说明：	将寄存器 f 的内容循环左移 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）。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。 			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

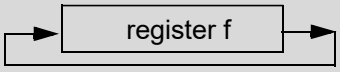
示例：	RLNCF	REG, 1, 0
执行指令前 REG = 1010 1011 执行指令后 REG = 0101 0111		

RRCF	对 f 执行带进位的循环右移			
语法：	RRCF f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(f<n>) → dest<n – 1>, (f<0>) → C, (C) → dest<7>			
受影响的状态位：	C、N 和 Z			
指令编码：	0011	00da	ffff	ffff
说明：	将寄存器 f 的内容连同进位标志位一起循环右移 1 位。如果 d 为 0，结果存入 W 寄存器。如果 d 为 1，结果存回寄存器 f（默认）。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。 			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4

译码	读取寄存器 f	处理数据	写入目标

示例：	RRCF	REG, 0, 0
执行指令前 REG = 1110 0110 C = 0 执行指令后 REG = 1110 0110 W = 0111 0011 C = 0		

RRNCF	f 循环右移（不带进位）			
语法：	RRNCF f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(f<n>) → dest<n – 1>, (f<0>) → dest<7>			
受影响的状态位：	N 和 Z			
指令编码：	0100	00da	ffff	ffff
说明：	将寄存器 f 的内容循环右移 1 位。如果 d 为 0，结果存入 W 寄存器。如果 d 为 1，结果存回寄存器 f（默认）。 如果 a 为 0，将选择快速操作存储区（默认），覆盖 BSR 值。如果 a 为 1，将根据 BSR 值选择快速操作存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。 			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

例 1：	RRNCF REG, 1, 0
执行指令前 REG = 1101 0111 执行指令后 REG = 1110 1011	
例 2：	RRNCF REG, 0, 0
执行指令前 W = ? REG = 1101 0111 执行指令后 W = 1110 1011 REG = 1101 0111	

SETF	将 f 置 1			
语法：	SETF f {,a}			
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]			
操作：	FFh → f			
受影响的状态位：	无			
指令编码：	0110	100a	ffff	ffff
说明：	将指定寄存器的内容设为 FFh。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入寄存器 f

示例：	SETF	REG, 1
执行指令前 REG = 5Ah 执行指令后 REG = FFh		

SLEEP	进入休眠模式			
语法：	SLEEP			
操作数：	无			
操作：	00h → WDT， 0 → WDT 后分频器， 1 → $\overline{TO}$， 0 → $\overline{PD}$			
受影响的状态位：	$\overline{TO}$ 和 $\overline{PD}$			
指令编码：	0000	0000	0000	0011
说明：	掉电（$\overline{PD}$）状态位清零。超时（$\overline{TO}$）状态位置 1。看门狗定时器及其后分频器被清零。 振荡器停振，处理器进入休眠模式。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	空操作	处理数据	进入休眠状态

示例：	SLEEP

执行指令前 $\overline{TO}$= ? $\overline{PD}$= ? 执行指令后 $\overline{TO}$= 1 † $\overline{PD}$= 0 † 如果 WDT 导致唤醒，则该位被清零。

SUBFWB	用 W 的内容减去 f 的内容（带借位）			
语法：	SUBFWB f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(W) – (f) – ($\overline{C}$) → 目标寄存器			
受影响的状态位：	N、OV、C、DC 和 Z			
指令编码：	0101	01da	ffff	ffff
说明：	用 W 的内容减去寄存器 f 的内容以及进位标志（借位）（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（默认）。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

例 1：	
	`SUBFWB  REG, 1, 0`
执行指令前 REG = 3 W = 2 C = 1 执行指令后 REG = FF W = 2 C = 0 Z = 0 N = 1；结果为负	
例 2：	`SUBFWB  REG, 0, 0`

执行指令前 REG = 2 W = 5 C = 1 执行指令后 REG = 2 W = 3 C = 1 Z = 0 N = 0；结果为正	
例 3：	`SUBFWB  REG, 1, 0`
执行指令前 REG = 1 W = 2 C = 0 执行指令后 REG = 0 W = 2 C = 1 Z = 1；结果为零 N = 0	

SUBLW	从立即数中减去 W 的内容			
语法：	SUBLW k			
操作数：	0 ≤ k ≤ 255			
操作：	k - (W) →			
受影响的状态位：	N、OV、C、DC 和 Z			
指令编码：	0000	1000	kkkk	kkkk
说明	用 8 位立即数 k 减去 W 的内容。结果存入 W 寄存器。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 k	处理数据	写入 W

例 1：	`SUBLW`	`02h`
执行指令前 W = 01h C = ? 执行指令后 W = 01h C = 1；结果为正 Z = 0 N = 0		
例 2：	`SUBLW`	`02h`

执行指令前 W = 02h C = ? 执行指令后 W = 00h C = 1；结果为零 Z = 1 N = 0		
例 3：	`SUBLW`	`02h`
执行指令前 W = 03h C = ? 执行指令后 W = FFh；（二进制补码） C = 0；结果为负 Z = 0 N = 1		

SUBWF	**f 减去 W**			
语法：	SUBWF f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(f) – (W) → 目标寄存器			
受影响的状态位：	N、OV、C、DC 和 Z			
指令编码：	0101	11da	ffff	ffff
说明：	用寄存器 f 的内容减去 W 的内容（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（默认）。 如果 a 为 `0`，则选择快速操作存储区。如果 a 为 `1`，则使用 BSR 选择 GPR 存储区。 如果 a 为 `0` 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

例 1：	`SUBWF   REG, 1, 0`

执行指令前 REG = 3 W = 2 C = ? 执行指令后 REG = 1 W = 2 C = 1；结果为正 Z = 0 N = 0	
例 2：	`SUBWF  REG, 0, 0`
执行指令前 REG = 2 W = 2 C = ? 执行指令后 REG = 2 W = 0 C = 1；结果为零 Z = 1 N = 0	
例 3：	`SUBWF  REG, 1, 0`
执行指令前 REG = 1 W = 2 C = ? 执行指令后 REG = FFh；（二进制补码） W = 2 C = 0；结果为负 Z = 0 N = 1	

SUBWFB	用 f 的内容减去 W 的内容（带借位）			
语法：	SUBWFB f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(f) – (W) – ($\overline{C}$) → 目标寄存器			
受影响的状态位：	N、OV、C、DC 和 Z			
指令编码：	0101	10da	ffff	ffff
说明：	用寄存器 f 的内容减去 W 的内容以及进位标志（借位）（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（默认）。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			

标准指令集详解（续）

SUBWFB	用 f 的内容减去 W 的内容（带借位）
指令周期数：	1

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

例 1：

```
SUBWFB  REG, 1, 0
```

执行指令前
REG = 19h（0001 1001）
W = 0Dh（0000 1101）
C = 1
执行指令后
REG = 0Ch（0000 1100）
W = 0Dh（0000 1101）
C = 1
Z = 0
N = 0；结果为正

例 2：

```
SUBWFB        REG, 0, 0
```

执行指令前
REG = 1Bh（0001 1011）
W = 1Ah（0001 1010）
C = 0
执行指令后
REG = 1Bh（0001 1011）
W = 00h
C = 1
Z = 1；结果为零
N = 0

例 3：

```
SUBWFB  REG, 1, 0
```

执行指令前
REG = 03h（0000 0011）
W = 0Eh（0000 1110）
C = 1
执行指令后
REG = F5h（1111 0101）
；[二进制补码]
W = 0Eh（0000 1110）
C = 0
Z = 0
N = 1；结果为负

SWAPF	将 f 中的两个半字节进行交换
语法：	SWAPF f {,d {,a}}

标准指令集详解（续）

SWAPF	将 f 中的两个半字节进行交换			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(f<3:0>) → dest<7:4>, (f<7:4>) → dest<3:0>			
受影响的状态位：	无			
指令编码：	0011	10da	ffff	ffff
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）。如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

示例：	`SWAPF`	`REG, 1, 0`
执行指令前 REG = 53h 执行指令后 REG = 35h		

TBLRD	表读			
语法：	TBLRD (*; *+; *-; +*)			
操作数：	无			
操作：	如果执行 TBLRD *， (程序存储器(TBLPTR)) → TABLAT； TBLPTR——无变化； 如果执行 TBLRD *+， (程序存储器(TBLPTR)) → TABLAT； (TBLPTR) + 1 → TBLPTR； 如果执行 TBLRD *-， (程序存储器(TBLPTR)) → TABLAT； (TBLPTR) – 1 → TBLPTR； 如果执行 TBLRD +*， (TBLPTR) + 1 → TBLPTR； (程序存储器(TBLPTR)) → TABLAT；			
受影响的状态位：	无			
指令编码：	0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*

标准指令集详解（续）

TBLRD	表读
说明：	该指令用于读取程序存储器（P.M.）的内容。要寻址程序存储器，需使用称为表指针（TBLPTR）的指针。TBLPTR（21 位指针）指向程序存储器中的每个字节。TBLPTR 的地址范围为 2 MB。 TBLPTR[0] = 0：程序存储器字的最低有效字节 TBLPTR[0] = 1：程序存储器字的最高有效字节 TBLRD 指令可以对 TBLPTR 的值进行如下修改： • 无变化 • 后递增 • 后递减 • 预递增
指令字数：	1
指令周期数：	2

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	空操作	空操作	空操作
空操作	空操作（读取程序存储器）	空操作	空操作（写入 TABLAT）

TBLRD	表读（续）
例 1：	`TBLRD *+ ;`
执行指令前 TABLAT = 55h TBLPTR = 00A356h 存储单元（00A356h）= 34h 执行指令后 TABLAT = 34h TBLPTR = 00A357h	
例 2：	`TBLRD +* ;`
执行指令前 TABLAT = AAh TBLPTR = 01A357h 存储单元（01A357h）= 12h 存储单元（01A358h）= 34h 执行指令后 TABLAT = 34h TBLPTR = 01A358h	

TBLWT（续）	表写
语法：	TBLWT (*; *+; *-; +*)
操作数：	无

标准指令集详解（续）

TBLWT（续）	表写			
操作：	如果执行 TBLWT*， (TABLAT) → 保持寄存器； TBLPTR——无变化； 如果执行 TBLWT*+， (TABLAT) → 保持寄存器； (TBLPTR) + 1 → TBLPTR； 如果执行 TBLWT*-， (TABLAT) → 保持寄存器； (TBLPTR) – 1 → TBLPTR； 如果执行 TBLWT+*， (TBLPTR) + 1 → TBLPTR； (TABLAT) → 保持寄存器；			
受影响的状态位：	无			
指令编码：	0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*
说明：	该指令使用 TBLPTR 的 LSB 来确定 TABLAT 写入哪个保持寄存器。保持寄存器用于编程程序存储器（P.M.）的内容。有关编程闪存的更多详细信息，请参见**“编程闪存”**一节。 TBLPTR（21 位指针）指向程序存储器中的每个字节。TBLPTR 的地址范围为 2 MB。TBLPTR 的 LSb 用于选择将访问程序存储单元的哪个字节。 TBLPTR[0] = 0：程序存储器字的最低有效字节 TBLPTR[0] = 1：程序存储器字的最高有效字节 TBLWT 指令可以对 TBLPTR 的值进行如下修改： • 无变化 • 后递增 • 后递减 • 预递增			
指令字数：	1			
指令周期数：	2			
Q 周期活动：				
	Q1	Q2	Q3	Q4
	译码	空操作	空操作	空操作
	空操作	空操作（读取 TABLAT）	空操作	空操作（写入保持寄存器）

TBLWT	表写（续）
例 1：	TBLWT *+;

执行指令前 TABLAT = 55h TBLPTR = 00A356h 保持寄存器（00A356h）= FFh 执行指令后（表写完成） TABLAT = 55h TBLPTR = 00A357h 保持寄存器（00A356h）= 55h	
例 2：	TBLWT +*;
执行指令前 TABLAT = 34h TBLPTR = 01389Ah 保持寄存器（01389Ah）= FFh 保持寄存器（01389Bh）= FFh 执行指令后（表写完成） TABLAT = 34h TBLPTR = 01389Bh 保持寄存器（01389Ah）= FFh 保持寄存器（01389Bh）= 34h	

TSTFSZ	**测试 f，为 0 则跳过**			
语法：	TSTFSZ f {,a}			
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]			
操作：	如果 f = 0 则跳过			
受影响的状态位：	无			
指令编码：	0110	011a	ffff	ffff
说明：	如果 f = 0，则丢弃当前指令执行期间所取的下一条指令，代之执行一条 NOP 指令，使之成为一条双周期指令。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1（2） **注：**如果跳过且后跟一条双字指令，则为三个周期。			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	空操作

如果跳过：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
如果跳过且后跟一条双字指令：			
Q1	Q2	Q3	Q4
空操作	空操作	空操作	空操作
空操作	空操作	空操作	空操作

示例：	
	HERE TSTFSZ CNT, 1 NZERO : ZERO :
执行指令前 PC = 地址（HERE） 执行指令后 如果 CNT = 00h， PC = 地址（ZERO） 如果 CNT ≠ 00h， PC = 地址（NZERO）	

XORLW	立即数与 W 作逻辑异或运算			
语法：	XORLW k			
操作数：	0 ≤ k ≤ 255			
操作：	(W) .XOR. k →			
受影响的状态位：	N 和 Z			
指令编码：	0000	1010	kkkk	kkkk
说明：	将 W 的内容与 8 位立即数 k 进行逻辑异或运算。结果存入 W 寄存器。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 k	处理数据	写入 W

示例：	XORLW	0AFh
执行指令前 W = B5h 执行指令后 W = 1Ah		

XORWF	W 与 f 作逻辑异或运算			
语法：	XORWF f {,d {,a}}			
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
操作：	(W) .XOR.(f) → 目标寄存器			
受影响的状态位：	N 和 Z			
指令编码：	0001	10da	ffff	ffff
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（默认）。 如果 a 为 0，则选择快速操作存储区。如果 a 为 1，则使用 BSR 选择 GPR 存储区。 如果 a 为 0 且使能了扩展指令集，则只要 f ≤ 95（5Fh），该指令便将在立即数变址寻址模式下工作。有关详细信息，请参见立即数变址寻址模式下面向字节和面向位的指令。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：		

Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

示例：	XORWF REG, 1, 0
执行指令前 REG = AFh W = B5h 执行指令后 REG = 1Ah W = B5h	

40.2. 扩展指令集

除指令集的 75 条标准指令外，CN2710 器件还为核心 CPU 功能提供了可选扩展指令。增加的功能包括八条附加指令，用于增强间接和变址寻址操作以及许多标准指令的立即数变址寻址模式的实现。

默认情况下，禁止扩展指令集的附加功能。要使能这些附加功能，用户必须将 XINST 配置位置 1。

扩展指令集中的指令都可以归类为立即数操作类指令，可以操作文件选择寄存器，也可以将它们用于变址寻址。其中的两条指令 ADDFSR 和 SUBFSR 各有一个使用 FSR2 的附加特殊实例。这两种指令形式（ADDULNK 和 SUBULNK）允许在执行后自动返回。

扩展指令专门用于优化以高级语言（特别是 C 语言）编写的可重入程序代码（即递归或使用软件堆栈的代码）。此外，这使得使用高级语言工作的用户可以更有效地对数据结构执行某些操作。其中包括：

- 进入和退出子程序时动态分配和释放软件堆栈空间
- 函数指针调用
- 软件堆栈指针操作
- 处理位于软件堆栈中的变量

扩展指令语法中提供了扩展指令集中的指令汇总。扩展指令集中给出了详细说明。标准指令集中的操作码字段说明同时适用于标准和扩展指令集。

重要：指令集扩展和立即数变址寻址模式旨在优化采用 C 语言编写的应用程序。用户很可能不会使用这些指令。这些命令的语法作为参考提供给可能需要复查由编译器生成的代码的用户。

40.2.1. 扩展指令语法

大多数扩展指令使用变址参数，使用一个文件选择寄存器和某个偏移量来指定源或目标寄存器。当指令的参数作为变址寻址的一部分时，将用方括号（“[ ]”）括起来。这样做是为了指示该参数用作变址或偏移量。

使能扩展指令集时，方括号也用于指示面向字节和面向位的指令中的变址参数。这是对其语法的其他更改的补充。有关详细信息，请参见标准命令的扩展指令语法。

重要：过去，方括号曾用于表示早期指令集中的可选参数。在本文和后续文档中，可选参数用花括号（“{}”）表示。

表 40-3. 指令集的扩展

助记符和操作数		说明	周期数	16 位指令字				受影响的状态位
				MSb			LSb	
ADDFSR	f, k	将立即数与 FSR 相加	1	1110	1000	ffkk	kkkk	无
ADDULNK	k	将立即数加到 FSR2 并返回	2	1110	1000	11kk	kkkk	无
CALLW		使用 WREG 寄存器调用子程序	2	0000	0000	0001	0100	无
MOVSF	z_s, f_d	将 z_s（源）中的内容送入（第 1 个字）	2	1110	1011	0zzz	zzzz	无
		f_d（目标）（第 2 个字）		1111	ffff	ffff	ffff	
MOVSS	z_s, z_d	将 z_s（源）中的内容送入（第 1 个字）	2	1110	1011	1zzz	zzzz	无
		z_d（目标）（第 2 个字）		1111	xxxx	xzzz	zzzz	
PUSHL	k	将立即数存入 FSR2，FSR2 递减 1	1	1110	1010	kkkk	kkkk	无
SUBFSR	f, k	从 FSR 中减去立即数	1	1110	1001	ffkk	kkkk	无
SUBULNK	k	从 FSR2 中减去立即数并返回	2	1110	1001	11kk	kkkk	无

40.2.2. 扩展指令集

ADDFSR	将立即数与 FSR 相加			
语法：	ADDFSR f, k			
操作数：	0 ≤ k ≤ 63 f ∈ [0, 1, 2]			
操作：	FSR(f) + k → FSR(f)			
受影响的状态位：	无			
指令编码：	1110	1000	ffkk	kkkk
说明：	将 6 位立即数 k 加到 f 指定的 FSR 的内容。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 k	处理数据	写入 FSR

示例：	ADDFSR 2, 23h
执行指令前 FSR2 = 03FFh 执行指令后 FSR2 = 0422h	

ADDULNK	将立即数加到 FSR2 并返回
语法：	ADDULNK k

扩展指令集（续）

ADDULNK	将立即数加到 FSR2 并返回			
操作数：	0 ≤ k ≤ 63			
操作：	FSR2 + k → FSR2， (TOS) → PC			
受影响的状态位：	无			
指令编码：	1110	1000	11kk	kkkk
说明：	将 6 位立即数 k 加到 FSR2 的内容。然后通过向 PC 装入 TOS 来执行 RETURN。 该指令需要两个周期来执行；在第二个周期执行 NOP。 可认为该指令是 ADDFSR 指令的特例，其中 f = 3（二进制 11）；它仅对 FSR2 进行操作。			
指令字数：	1			
指令周期数：	2			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取立即数 k	处理数据	写入 FSR
空操作	空操作	空操作	空操作

示例：	ADDULNK 23h
执行指令前 FSR2 = 03FFh PC = 0100h 执行指令后 FSR2 = 0422h PC = (TOS)	

重要：所有指令都可以在指令助记符之前采用可选的标号参数，以用于符号寻址。如果使用标号，则指令语法变为：{标号}指令参数。

CALLW	使用 WREG 调用子程序			
语法：	CALLW			
操作数：	无			
操作：	(PC + 2) → TOS， (W) → PCL， (PCLATH) → PCH， (PCLATU) → PCU			
受影响的状态位：	无			
指令编码：	0000	0000	0001	0100
说明	首先，将返回地址（PC + 2）压入返回堆栈。接下来，将 W 的内容写入 PCL 并丢弃现有值。然后，将 PCLATH 和 PCLATU 的内容分别锁存到 PCH 和 PCU 中。获取新的下一条指令时，在第二个周期执行 NOP 指令。 与 CALL 不同，没有更新 W、Status 或 BSR 的选项。			
指令字数：	1			
指令周期数：	2			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读 WREG	将 PC 压入堆栈	空操作

空操作	空操作	空操作	空操作

示例：	HERE	CALLW
执行指令前 PC = 地址（HERE） PCLATH = 10h PCLATU = 00h W = 06h 执行指令后 PC = 001006h TOS = 地址（HERE + 2） PCLATH = 10h PCLATU = 00h W = 06h		

MOVSF	将变址的内容送入 f			
语法：	MOVSF [z_s], f_d			
操作数：	0 ≤ z_s ≤ 127 0 ≤ f_d ≤ 4095			
操作：	((FSR2) + z_s) → f_d			
受影响的状态位：	无			
指令编码： 第 1 个字（源） 第 2 个字（目标）	1110 1111	1011 ffff	0zzz ffff	$zzzz_s$ $ffff_d$
说明：	将源寄存器的内容送入目标寄存器 f_d。通过将第一个字中的 7 位立即数偏移量 z_s 与 FSR2 的值相加来确定源寄存器的实际地址。目标寄存器的地址由第二个字中的 12 位立即数 f_d 指定。这两个地址可以是 4096 字节数据空间（000h 到 FFFh）中的任何位置。 MOVSF 指令不能使用 PCL、TOSU、TOSH 或 TOSL 作为目标寄存器。 如果结果源地址指向间接寻址寄存器，则返回的值将为 00h。			
指令字数：	2			
指令周期数：	2			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	确定源地址	确定源地址	读取源寄存器
译码	空操作 无假读	空操作	写入寄存器 f（目标）

示例：	MOVSF [05h], REG2

执行指令前
FSR2 = 80h
85h 的内容 = 33h
REG2 = 11h
执行指令后
FSR2 = 80h
85h 的内容 = 33h
REG2 = 33h

MOVSS	将变址的内容送入变址			
语法：	MOVSS [z_s], [z_d]			
操作数：	0 ≤ z_s ≤ 127 0 ≤ z_d ≤ 127			
操作：	((FSR2) + z_s) → ((FSR2) + z_d)			
受影响的状态位：	无			
指令编码： 第 1 个字（源） 第 2 个字（目标）	1110 1111	1011 xxxx	1zzz xzzz	$zzzz_s$ $zzzz_d$
说明	将源寄存器中的内容送入目标寄存器。通过将 7 位立即数偏移量 z_s 或 z_d 分别与 FSR2 的值相加来确定源寄存器和目标寄存器的地址。这两个寄存器可位于 4096 字节数据存储空间（000h 到 FFFh）中的任何位置。 MOVSS 指令不能使用 PCL、TOSU、TOSH 或 TOSL 作为目标寄存器。 如果结果源地址指向间接寻址寄存器，则返回的值将为 00h。如果结果目标地址指向间接寻址寄存器，则指令将作为 NOP 执行。			
指令字数：	2			
指令周期数：	2			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	确定源地址	确定源地址	读取源寄存器
译码	确定目标地址	确定目标地址	写入目标寄存器

示例：	`MOVSS [05h], [06h]`

执行指令前
FSR2 = 80h
85h 的内容 = 33h
86h 的内容 = 11h
执行指令后
FSR2 = 80h
85h 的内容 = 33h
86h 的内容 = 33h

PUSHL	将立即数存入 FSR2，FSR2 递减 1
语法：	PUSHL k
操作数：	0 ≤ k ≤ 255

扩展指令集（续）

PUSHL	将立即数存入 FSR2，FSR2 递减 1			
操作：	k → (FSR2)， FSR2 – 1 → FSR2			
受影响的状态位：	无			
指令编码：	1111	1010	kkkk	kkkk
说明：	将 8 位立即数 k 写入 FSR2 指定的数据存储器地址。在该操作后，将 FSR2 递减 1。 该指令允许用户将值压入软件堆栈。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取 k	处理数据	写入目标

示例：	`PUSHL 08h`
执行指令前 FSR2H:FSR2L = 01ECh 存储单元（01ECh）= 00h 执行指令后 FSR2H:FSR2L = 01EBh 存储单元（01ECh）= 08h	

SUBFSR	从 FSR 中减去立即数			
语法：	SUBFSR f, k			
操作数：	0 ≤ k ≤ 63			
	f ∈ [0, 1, 2]			
操作：	FSR(f) – k → FSRf			
受影响的状态位：	无			
指令编码：	1110	1001	ffkk	kkkk
说明：	从 f 指定的 FSR 的内容中减去 6 位立即数 k。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

示例：	SUBFSR 2, 23h
执行指令前 FSR2 = 03FFh 执行指令后 FSR2 = 03DCh	

SUBULNK	从 FSR2 中减去立即数并返回
语法：	SUBULNK k
操作数：	0 ≤ k ≤ 63

扩展指令集（续）				
SUBULNK	**从 FSR2 中减去立即数并返回**			
操作：	FSR2 – k → FSR2			
	(TOS) → PC			
受影响的状态位：	无			
指令编码：	1110	1001	11kk	kkkk
说明：	从 FSR2 的内容中减去 6 位立即数 k。然后通过向 PC 装入 TOS 来执行 RETURN。 该指令需要两个周期来执行；在第二个周期执行 NOP。 可认为该指令是 SUBFSR 指令的特例，其中 f = 3（二进制 11）；它仅对 FSR2 进行操作。			
指令字数：	1			
指令周期数：	2			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标
空操作	空操作	空操作	空操作

示例：	SUBULNK 23h
执行指令前 FSR2 = 03FFh PC = 0100h 执行指令后 FSR2 = 03DCh PC = (TOS)	

40.2.3. 立即数变址寻址模式下面向字节和面向位的指令

重要：使能指令集扩展可能会导致旧应用程序表现异常或完全失效。

除扩展指令集中的 8 个新命令外，使能扩展指令集还将使能立即数变址寻址模式（见“立即数变址寻址”章节）。这会显著影响解释标准指令集的许多命令的方式。

禁止扩展指令集时，嵌入在操作码中的地址被视为立即数存储单元：快速操作存储区中的存储单元（“a” = 0），或者由 BSR 指定的 GPR 存储区（“a” = 1）。但是，当使能扩展指令集且“a” = 0 时，5Fh 或更小的文件寄存器参数被解释为 FSR2 中指针值的偏移量，而不是立即数地址。在实际应用中，这意味着所有使用快速操作 RAM 位作为参数的指令（即所有面向字节和面向位的指令，或几乎一半的指令）在使能扩展指令集时的行为会有所不同。

当 FSR2 的内容为 00h 时，快速操作 RAM 的边界基本上会重映射到其原始值。这在创建向后兼容代码时可能很有用。如果使用此技术，可能需要保存 FSR2 的值，并在 C 和汇编程序之间来回切换时将其恢复，以便保存堆栈指针。用户还必须牢记扩展指令集的语法要求（见标准命令的扩展指令语法）。

尽管立即数变址寻址模式对于动态堆栈和指针操作非常有用，但如果在错误的寄存器上执行简单的算术运算，就会变得非常麻烦。当使能扩展指令集时，5Fh 或更小的寄存器地址用于立即数变址寻址。

下面提供了立即数变址寻址模式中典型的面向字节和面向位的指令的代表示例，以说明如何对执行产生影响。示例中显示的操作数条件适用于这些类型的所有指令。

40.2.3.1. 标准命令的扩展指令语法

当使能扩展指令集时，标准的面向字节和面向位的命令中的文件寄存器参数“f”将替换为立即数偏移值“k”。如前所述，只有当“f”小于或等于 5Fh 时才会发生这种情况。使用偏移值时，必须用方括号

（“[]”）表示。与扩展指令一样，通过使用方括号向编译器指示该值被解释为变址或偏移量。省略方括号或在方括号内使用大于 5Fh 的值将生成错误。

如果立即数变址寻址的变址参数正确括在方括号中，则永远无需指定快速操作 RAM 参数，它会自动假定为 0。当基于目标地址设置“a”时，这与标准操作（禁止扩展指令集）不同。在此模式下声明快速操作 RAM 位也会生成错误。

目标参数“d”的功能与之前相同。

40.2.4. 使能扩展指令集时的注意事项

需要注意的是，指令集扩展可能并不是对所有用户都有利，特别是不编写使用软件堆栈的代码的用户。

此外，立即数变址寻址模式可能对旧应用程序造成问题。原因在于，旧代码中的指令可能试图在 5Fh 以下的快速操作存储区中寻址寄存器。由于在使能指令集扩展时这些地址被解释为 FSR2 的立即数偏移，因此应用程序可能会读取或写入错误的数据地址。

将应用程序移植到 CN2710 时，考虑代码类型非常重要。使用指令集扩展时，使用“C”编写且受益于高效编译的大型可重入应用程序将顺利运行。大量使用快速操作存储区的旧应用程序很可能无法从使用扩展指令集中受益。

ADDWF	将 W 与变址相加（立即数变址寻址模式）			
语法：	ADDWF [k] {,d}			
操作数：	0 ≤ k ≤ 95 d ∈ [0,1]			
操作：	(W) + ((FSR2) + k) → 目标寄存器			
受影响的状态位：	N、OV、C、DC 和 Z			
指令编码：	0010	01d0	kkkk	kkkk
说明：	将 W 的内容与 FSR2 内容加偏移量 k 指示的寄存器的内容相加。 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（默认）。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取 k	处理数据	写入目标

示例：	ADDWF	[OFST]	, 0
执行指令前 W = 17h OFST = 2Ch FSR2 = 0A00h 0A2Ch 的内容 = 20h 执行指令后 W = 37h 0A2Ch 的内容 = 20h			

BSF	位置 1 变址（立即数变址寻址模式）
语法：	BSF [k], b
操作数：	0 ≤ f ≤ 95 0 ≤ b ≤ 7
操作：	1 → ((FSR2) + k)<b>

使能扩展指令集时的注意事项（续）				
BSF	**位置 1 变址（立即数变址寻址模式）**			
受影响的状态位：	无			
指令编码：	1000	bbb0	kkkk	kkkk
说明：	将 FSR2 内容加偏移量 k 指示的寄存器的内容的 bit b 置 1。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取寄存器 f	处理数据	写入目标

示例：	`BSF`	`[FLAG_OFST], 7`
执行指令前 FLAG_OFST = 0Ah FSR2 = 0A00h 0A0Ah 的内容 = 55h 执行指令后 0A0Ah 的内容 = D5h		

SETF	**置 1 变址（立即数变址寻址模式）**			
语法：	SETF [k]			
操作数：	0 ≤ k ≤ 95			
操作：	FFh → ((FSR2) + k)			
受影响的状态位：	无			
指令编码：	0110	1000	kkkk	kkkk
说明：	将 FSR2 内容加偏移量 k 指示的寄存器的内容设置为 FFh。			
指令字数：	1			
指令周期数：	1			

Q 周期活动：			
Q1	Q2	Q3	Q4
译码	读取 k	处理数据	写入 寄存器

示例：	`SETF`	`[OFST]`
执行指令前 OFST = 2Ch FSR2 = 0A00h 0A2Ch 的内容 = 00h 执行指令后 0A2Ch 的内容 = FFh		

40.2.5. 编程工具的特殊注意事项

要为扩展指令集开发软件，用户必须在其语言工具中使能对指令和变址寻址模式的支持。根据所使用的环境，可以通过以下几种方式完成该操作：

- 环境中的菜单选项或对话框，允许用户为项目配置语言工具及其设置

- 命令行选项
- 源代码中的伪指令

这些选项因编译器、汇编器和开发环境而异。建议用户查看其开发系统随附的文档，以获取相应的信息。

41. 寄存器汇总

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x00 ... 0x0E1E	保留									
0x0E1F	CLCIN0PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0E20	CLCIN1PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0E21	CLCIN2PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0E22	CLCIN3PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0E23	CLCIN4PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0E24	CLCIN5PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0E25	CLCIN6PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0E26	CLCIN7PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0E27	CLC1CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E28	CLC1POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E29	CLC1SEL0	7:0			D1S[5:0]					
0x0E2A	CLC1SEL1	7:0			D2S[5:0]					
0x0E2B	CLC1SEL2	7:0			D3S[5:0]					
0x0E2C	CLC1SEL3	7:0			D4S[5:0]					
0x0E2D	CLC1GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E2E	CLC1GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E2F	CLC1GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E30	CLC1GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E31	CLC2CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E32	CLC2POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E33	CLC2SEL0	7:0			D1S[5:0]					
0x0E34	CLC2SEL1	7:0			D2S[5:0]					
0x0E35	CLC2SEL2	7:0			D3S[5:0]					
0x0E36	CLC2SEL3	7:0			D4S[5:0]					
0x0E37	CLC2GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E38	CLC2GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E39	CLC2GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E3A	CLC2GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E3B	CLC3CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E3C	CLC3POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E3D	CLC3SEL0	7:0			D1S[5:0]					
0x0E3E	CLC3SEL1	7:0			D2S[5:0]					
0x0E3F	CLC3SEL2	7:0			D3S[5:0]					
0x0E40	CLC3SEL3	7:0			D4S[5:0]					
0x0E41	CLC3GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E42	CLC3GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E43	CLC3GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E44	CLC3GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E45	CLC4CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E46	CLC4POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E47	CLC4SEL0	7:0			D1S[5:0]					
0x0E48	CLC4SEL1	7:0			D2S[5:0]					
0x0E49	CLC4SEL2	7:0			D3S[5:0]					
0x0E4A	CLC4SEL3	7:0			D4S[5:0]					
0x0E4B	CLC4GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E4C	CLC4GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E4D	CLC4GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E4E	CLC4GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E4F	CLC5CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E50	CLC5POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E51	CLC5SEL0	7:0			D1S[5:0]					
0x0E52	CLC5SEL1	7:0			D2S[5:0]					
0x0E53	CLC5SEL2	7:0			D3S[5:0]					
0x0E54	CLC5SEL3	7:0			D4S[5:0]					

寄存器汇总（续）

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x0E55	CLC5GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E56	CLC5GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E57	CLC5GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E58	CLC5GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E59	CLC6CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E5A	CLC6POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E5B	CLC6SEL0	7:0			D1S[5:0]					
0x0E5C	CLC6SEL1	7:0			D2S[5:0]					
0x0E5D	CLC6SEL2	7:0			D3S[5:0]					
0x0E5E	CLC6SEL3	7:0			D4S[5:0]					
0x0E5F	CLC6GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E60	CLC6GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E61	CLC6GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E62	CLC6GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E63	CLC7CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E64	CLC7POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E65	CLC7SEL0	7:0			D1S[5:0]					
0x0E66	CLC7SEL1	7:0			D2S[5:0]					
0x0E67	CLC7SEL2	7:0			D3S[5:0]					
0x0E68	CLC7SEL3	7:0			D4S[5:0]					
0x0E69	CLC7GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E6A	CLC7GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E6B	CLC7GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E6C	CLC7GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E6D	CLC8CON	7:0	EN		OUT	INTP	INTN		MODE[2:0]	
0x0E6E	CLC8POL	7:0	POL				G4POL	G3POL	G2POL	G1POL
0x0E6F	CLC8SEL0	7:0			D1S[5:0]					
0x0E70	CLC8SEL1	7:0			D2S[5:0]					
0x0E71	CLC8SEL2	7:0			D3S[5:0]					
0x0E72	CLC8SEL3	7:0			D4S[5:0]					
0x0E73	CLC8GLS0	7:0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N
0x0E74	CLC8GLS1	7:0	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N
0x0E75	CLC8GLS2	7:0	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N
0x0E76	CLC8GLS3	7:0	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
0x0E77	CLCDATA	7:0	MLC8OUT	MLC7OUT	MLC6OUT	MLC5OUT	MLC4OUT	MLC3OUT	MLC2OUT	MLC1OUT
0x0E78 … 0x0E87	保留									
0x0E88	RX2PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0E89	CK2PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0E8A	SSP2CLKPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0E8B	SSP2DATPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0E8C	SSP2SSPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0E8D	SSP2BUF	7:0	BUF[7:0]							
0x0E8E	SSP2ADD	7:0	ADD[7:0]							
0x0E8F	SSP2MSK	7:0	MSK[6:0]							MSK0
0x0E90	SSP2STAT	7:0	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
0x0E91	SSP2CON1	7:0	WCOL	SSPOV	SSPEN	CKP	SSPM[3:0]			
0x0E92	SSP2CON2	7:0	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
0x0E93	SSP2CON3	7:0	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
0x0E94	RC2REG	7:0	RCREG[7:0]							
0x0E95	TX2REG	7:0	TXREG[7:0]							
0x0E96	SP2BRG	7:0	SPBRG[7:0]							
		15:8	SPBRG[15:8]							
0x0E98	RC2STA	7:0	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
0x0E99	TX2STA	7:0	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
0x0E9A	BAUD2CON	7:0	ABDOVF	RCIDL		SCKP	BRG16		WUE	ABDEN
0x0E9B	PPSLOCK	7:0								PPSLOCKED
0x0E9C	INT0PPS	7:0					PORT		PIN[2:0]	
0x0E9D	INT1PPS	7:0					PORT		PIN[2:0]	

寄存器汇总（续）

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x0E9E	INT2PPS	7:0					PORT		PIN[2:0]	
0x0E9F	T0CKIPPS	7:0					PORT		PIN[2:0]	
0x0EA0	T1CKIPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EA1	T1GPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EA2	T3CKIPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EA3	T3GPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EA4	T5CKIPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EA5	T5GPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EA6	T2INPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EA7	T4INPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EA8	T6INPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EA9	ADACTPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EAA	CCP1PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EAB	CCP2PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EAC	CWG1PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EAD	MDCARLPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EAE	MDCARHPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EAF	MDSRCPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EB0	RX1PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EB1	CK1PPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EB2	SSP1CLKPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EB3	SSP1DATPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EB4	SSP1SSPPS	7:0				PORT[1:0]			PIN[2:0]	
0x0EB5	IPR0	7:0			TMR0IP	IOCIP		INT2IP	INT1IP	INT0IP
0x0EB6	IPR1	7:0	OSCFIP	CSWIP					ADTIP	ADIP
0x0EB7	IPR2	7:0	HLVDIP	ZCDIP					C2IP	C1IP
0x0EB8	IPR3	7:0	RC2IP	TX2IP	RC1IP	TX1IP	BCL2IP	SSP2IP	BCL1IP	SSP1IP
0x0EB9	IPR4	7:0			TMR6IP	TMR5IP	TMR4IP	TMR3IP	TMR2IP	TMR1IP
0x0EBA	IPR5	7:0	CLC4IP	CLC3IP	CLC2IP	CLC1IP		TMR5GIP	TMR3GIP	TMR1GIP
0x0EBB	IPR6	7:0	CLC8IP	CLC7IP	CLC6IP	CLC5IP			CCP2IP	CCP1IP
0x0EBC	IPR7	7:0	SCANIP	CRCIP	NVMIP					CWG1IP
0x0EBD	PIE0	7:0			TMR0IE	IOCIE		INT2IE	INT1IE	INT0IE
0x0EBE	PIE1	7:0	OSCFIE	CSWIE					ADTIE	ADIE
0x0EBF	PIE2	7:0	HLVDIE	ZCDIE					C2IE	C1IE
0x0EC0	PIE3	7:0	RC2IE	TX2IE	RC1IE	TX1IE	BCL2IE	SSP2IE	BCL1IE	SSP1IE
0x0EC1	PIE4	7:0			TMR6IE	TMR5IE	TMR4IE	TMR3IE	TMR2IE	TMR1IE
0x0EC2	PIE5	7:0	CLC4IE	CLC3IE	CLC2IE	CLC1IE		TMR5GIE	TMR3GIE	TMR1GIE
0x0EC3	PIE6	7:0	CLC8IE	CLC7IE	CLC6IE	CLC5IE			CCP2IE	CCP1IE
0x0EC4	PIE7	7:0	SCANIE	CRCIE	NVMIE					CWG1IE
0x0EC5	PIR0	7:0			TMR0IF	IOCIF		INT2IF	INT1IF	INT0IF
0x0EC6	PIR1	7:0	OSCFIF	CSWIF					ADTIF	ADIF
0x0EC7	PIR2	7:0	HLVDIF	ZCDIF					C2IF	C1IF
0x0EC8	PIR3	7:0	RC2IF	TX2IF	RC1IF	TX1IF	BCL2IF	SSP2IF	BCL1IF	SSP1IF
0x0EC9	PIR4	7:0			TMR6IF	TMR5IF	TMR4IF	TMR3IF	TMR2IF	TMR1IF
0x0ECA	PIR5	7:0	CLC4IF	CLC3IF	CLC2IF	CLC1IF		TMR5GIF	TMR3GIF	TMR1GIF
0x0ECB	PIR6	7:0	CLC8IF	CLC7IF	CLC6IF	CLC5IF			CCP2IF	CCP1IF
0x0ECC	PIR7	7:0	SCANIF	CRCIF	NVMIF					CWG1IF
0x0ECD	WDTCON0	7:0			WDTPS[4:0]					SEN
0x0ECE	WDTCON1	7:0		WDTCS[2:0]				WINDOW[2:0]		
0x0ECF	WDTPSL	7:0	PSCNTL[7:0]							
0x0ED0	WDTPSH	7:0	PSCNTH[7:0]							
0x0ED1	WDTTMR	7:0	WDTTMR[4:0]					STATE	PSCNT[1:0]	
0x0ED2	CPUDOZE	7:0	IDLEN	DOZEN	ROI	DOE		DOZE[2:0]		
0x0ED3	OSCCON1	7:0		NOSC[2:0]			NDIV[3:0]			
0x0ED4	OSCCON2	7:0		COSC[2:0]			CDIV[3:0]			
0x0ED5	OSCCON3	7:0	CSWHOLD	SOSCPWR		ORDY	NOSCR			
0x0ED6	OSCSTAT	7:0	EXTOR	HFOR	MFOR	LFOR	SOR	ADOR		PLLR
0x0ED7	OSCEN	7:0	EXTOEN	HFOEN	MFOEN	LFOEN	SOSCEN	ADOEN		
0x0ED8	OSCTUNE	7:0	HFTUN[5:0]							
0x0ED9	OSCFRQ	7:0					HFFRQ[3:0]			

寄存器汇总（续）

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x0EDA	VREGCON	7:0			PMSYS[1:0]				VREGPM[1:0]	
0x0EDB	BORCON	7:0	SBOREN							BORRDY
0x0EDC	PMD0	7:0	SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD
0x0EDD	PMD1	7:0		TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD
0x0EDE	PMD2	7:0		DACMD	ADCMD			CMP2MD	CMP1MD	ZCDMD
0x0EDF	PMD3	7:0	CLC8MD	CLC7MD	CLC6MD	CLC5MD	PWM4MD	PWM3MD	CCP2MD	CCP1MD
0x0EE0	PMD4	7:0	UART2MD	UART1MD	MSSP2MD	MSSP1MD				CWG1MD
0x0EE1	PMD5	7:0	CLC4MD	CLC3MD	CLC2MD	CLC1MD				DSMMD
0x0EE2	RA0PPS	7:0						PPS[4:0]		
0x0EE3	RA1PPS	7:0						PPS[4:0]		
0x0EE4	RA2PPS	7:0						PPS[4:0]		
0x0EE5	RA3PPS	7:0						PPS[4:0]		
0x0EE6	RA4PPS	7:0						PPS[4:0]		
0x0EE7	RA5PPS	7:0						PPS[4:0]		
0x0EE8	RA6PPS	7:0						PPS[4:0]		
0x0EE9	RA7PPS	7:0						PPS[4:0]		
0x0EEA	RB0PPS	7:0						PPS[4:0]		
0x0EEB	RB1PPS	7:0						PPS[4:0]		
0x0EEC	RB2PPS	7:0						PPS[4:0]		
0x0EED	RB3PPS	7:0						PPS[4:0]		
0x0EEE	RB4PPS	7:0						PPS[4:0]		
0x0EEF	RB5PPS	7:0						PPS[4:0]		
0x0EF0	RB6PPS	7:0						PPS[4:0]		
0x0EF1	RB7PPS	7:0						PPS[4:0]		
0x0EF2	RC0PPS	7:0						PPS[4:0]		
0x0EF3	RC1PPS	7:0						PPS[4:0]		
0x0EF4	RC2PPS	7:0						PPS[4:0]		
0x0EF5	RC3PPS	7:0						PPS[4:0]		
0x0EF6	RC4PPS	7:0						PPS[4:0]		
0x0EF7	RC5PPS	7:0						PPS[4:0]		
0x0EF8	RC6PPS	7:0						PPS[4:0]		
0x0EF9	RC7PPS	7:0						PPS[4:0]		
0x0EFA	RD0PPS	7:0						PPS[4:0]		
0x0EFB	RD1PPS	7:0						PPS[4:0]		
0x0EFC	RD2PPS	7:0						PPS[4:0]		
0x0EFD	RD3PPS	7:0						PPS[4:0]		
0x0EFE	RD4PPS	7:0						PPS[4:0]		
0x0EFF	RD5PPS	7:0						PPS[4:0]		
0x0F00	RD6PPS	7:0						PPS[4:0]		
0x0F01	RD7PPS	7:0						PPS[4:0]		
0x0F02	RE0PPS	7:0						PPS[4:0]		
0x0F03	RE1PPS	7:0						PPS[4:0]		
0x0F04	RE2PPS	7:0						PPS[4:0]		
0x0F05	IOCAF	7:0	IOCAF7	IOCAF6	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0
0x0F06	IOCAN	7:0	IOCAN7	IOCAN6	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0
0x0F07	IOCAP	7:0	IOCAP7	IOCAP6	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0
0x0F08	INLVLA	7:0	INLVLA7	INLVLA6	INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0
0x0F09	SLRCONA	7:0	SLRA7	SLRA6	SLRA5	SLRA4	SLRA3	SLRA2	SLRA1	SLRA0
0x0F0A	ODCONA	7:0	ODCA7	ODCA6	ODCA5	ODCA4	ODCA3	ODCA2	ODCA1	ODCA0
0x0F0B	WPUA	7:0	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
0x0F0C	ANSELA	7:0	ANSELA7	ANSELA6	ANSELA5	ANSELA4	ANSELA3	ANSELA2	ANSELA1	ANSELA0
0x0F0D	IOCBF	7:0	IOCBF7	IOCBF6	IOCBF5	IOCBF4	IOCBF3	IOCBF2	IOCBF1	IOCBF0
0x0F0E	IOCBN	7:0	IOCBN7	IOCBN6	IOCBN5	IOCBN4	IOCBN3	IOCBN2	IOCBN1	IOCBN0
0x0F0F	IOCBP	7:0	IOCBP7	IOCBP6	IOCBP5	IOCBP4	IOCBP3	IOCBP2	IOCBP1	IOCBP0
0x0F10	INLVLB	7:0	INLVLB7	INLVLB6	INLVLB5	INLVLB4	INLVLB3	INLVLB2	INLVLB1	INLVLB0
0x0F11	SLRCONB	7:0	SLRB7	SLRB6	SLRB5	SLRB4	SLRB3	SLRB2	SLRB1	SLRB0
0x0F12	ODCONB	7:0	ODCB7	ODCB6	ODCB5	ODCB4	ODCB3	ODCB2	ODCB1	ODCB0
0x0F13	WPUB	7:0	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
0x0F14	ANSELB	7:0	ANSELB7	ANSELB6	ANSELB5	ANSELB4	ANSELB3	ANSELB2	ANSELB1	ANSELB0
0x0F15	IOCCF	7:0	IOCCF7	IOCCF6	IOCCF5	IOCCF4	IOCCF3	IOCCF2	IOCCF1	IOCCF0

寄存器汇总（续）

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x0F16	IOCCN	7:0	IOCCN7	IOCCN6	IOCCN5	IOCCN4	IOCCN3	IOCCN2	IOCCN1	IOCCN0
0x0F17	IOCCP	7:0	IOCCP7	IOCCP6	IOCCP5	IOCCP4	IOCCP3	IOCCP2	IOCCP1	IOCCP0
0x0F18	INLVLC	7:0	INLVLC7	INLVLC6	INLVLC5	INLVLC4	INLVLC3	INLVLC2	INLVLC1	INLVLC0
0x0F19	SLRCONC	7:0	SLRC7	SLRC6	SLRC5	SLRC4	SLRC3	SLRC2	SLRC1	SLRC0
0x0F1A	ODCONC	7:0	ODCC7	ODCC6	ODCC5	ODCC4	ODCC3	ODCC2	ODCC1	ODCC0
0x0F1B	WPUC	7:0	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
0x0F1C	ANSELC	7:0	ANSELC7	ANSELC6	ANSELC5	ANSELC4	ANSELC3	ANSELC2	ANSELC1	ANSELC0
0x0F1D … 0x0F21	保留									
0x0F22	IOCEF	7:0					IOCEF3			
0x0F23	IOCEN	7:0					IOCEN3			
0x0F24	IOCEP	7:0					IOCEP3			
0x0F25	INLVLE	7:0					INLVLE3			
0x0F26 … 0x0F27	保留									
0x0F28	WPUE	7:0					WPUE3			
0x0F29	保留									
0x0F2A	HLVDCON0	7:0	EN		OUT	RDY			INTH	INTL
0x0F2B	HLVDCON1	7:0					SEL[3:0]			
0x0F2C	FVRCON	7:0	FVREN	FVRRDY	TSEN	TSRNG	CDAFVR[1:0]		ADFVR[1:0]	
0x0F2D	ZCDCON	7:0	SEN		OUT	POL			INTP	INTN
0x0F2E	DAC1CON0	7:0	EN		OE1	OE2	PSS[1:0]			NSS
0x0F2F	DAC1CON1	7:0				DAC1R[4:0]				
0x0F30	CM2CON0	7:0	EN	OUT		POL			HYS	SYNC
0x0F31	CM2CON1	7:0							INTP	INTN
0x0F32	CM2NCH	7:0						NCH[2:0]		
0x0F33	CM2PCH	7:0						PCH[2:0]		
0x0F34	CM1CON0	7:0	EN	OUT		POL			HYS	SYNC
0x0F35	CM1CON1	7:0							INTP	INTN
0x0F36	CM1NCH	7:0						NCH[2:0]		
0x0F37	CM1PCH	7:0						PCH[2:0]		
0x0F38	CMOUT	7:0							MC2OUT	MC1OUT
0x0F39	CLKRCON	7:0	EN			DC[1:0]		DIV[2:0]		
0x0F3A	CLKRCLK	7:0					CLK[3:0]			
0x0F3B	CWG1CLK	7:0								CS
0x0F3C	CWG1ISM	7:0					ISM[3:0]			
0x0F3D	CWG1DBR	7:0			DBR[5:0]					
0x0F3E	CWG1DBF	7:0			DBF[5:0]					
0x0F3F	CWG1CON0	7:0	EN	LD				MODE[2:0]		
0x0F40	CWG1CON1	7:0			IN		POLD	POLC	POLB	POLA
0x0F41	CWG1AS0	7:0	SHUTDOWN	REN	LSBD[1:0]		LSAC[1:0]			
0x0F42	CWG1AS1	7:0	AS7E	AS6E	AS5E	AS4E	AS3E	AS2E	AS1E	AS0E
0x0F43	CWG1STR	7:0	OVRD	OVRC	OVRB	OVRA	STRD	STRC	STRB	STRA
0x0F44	SCANLADR	7:0	SCANLADRL[7:0]							
		15:8	SCANLADRH[7:0]							
		23:16			SCANLADRU[5:0]					
0x0F47	SCANHADR	7:0	SCANHADRL[7:0]							
		15:8	SCANHADRH[7:0]							
		23:16			SCANHADRU[5:0]					
0x0F4A	SCANCON0	7:0	SCANEN	SCANGO	BUSY	INVALID	INTM		MODE[1:0]	
0x0F4B	SCANTRIG	7:0				TSEL[4:0]				
0x0F4C	MDCON0	7:0	EN		OUT	OPOL				BIT
0x0F4D	MDCON1	7:0			CHPOL	CHSYNC			CLPOL	CLSYNC
0x0F4E	MDSRC	7:0				SRCS[4:0]				
0x0F4F	MDCARL	7:0					CLS[3:0]			
0x0F50	MDCARH	7:0					CHS[3:0]			
0x0F51	ADACT	7:0				ACT[4:0]				
0x0F52	ADCLK	7:0			CS[5:0]					

寄存器汇总（续）

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x0F53	ADREF	7:0				NREF			PREF[1:0]	
0x0F54	ADCON1	7:0	PPOL	IPEN	GPOL					DSEN
0x0F55	ADCON2	7:0	PSIS		CRS[2:0]		ACLR		MD[2:0]	
0x0F56	ADCON3	7:0			CALC[2:0]		SOI		TMD[2:0]	
0x0F57	ADACQ	7:0	ACQ[7:0]							
		15:8				ACQ[12:8]				
0x0F58	ADCAP	7:0				CAP[4:0]				
0x0F59	ADPRE	7:0	PRE[7:0]							
		15:8				PRE[12:8]				
0x0F5A	ADPCH	7:0			PCH[5:0]					
0x0F5B	ADCON0	7:0	ON	CONT		CS		FM		GO
0x0F5C	ADPREV	7:0	PREV[7:0]							
		15:8	PREV[15:8]							
0x0F5E	ADRES	7:0	RES[7:0]							
		15:8	RES[15:8]							
0x0F60	ADSTAT	7:0	AOV	UTHR	LTHR	MATH		STAT[2:0]		
0x0F61	ADRPT	7:0	RPT[7:0]							
0x0F62	ADCNT	7:0	CNT[7:0]							
0x0F63	ADSTPT	7:0	STPT[7:0]							
		15:8	STPT[15:8]							
0x0F65	ADLTH	7:0	LTH[7:0]							
		15:8	LTH[15:8]							
0x0F67	ADUTH	7:0	UTH[7:0]							
		15:8	UTH[15:8]							
0x0F69	ADERR	7:0	ERR[7:0]							
		15:8	ERR[15:8]							
0x0F6B	ADACC	7:0	ACC[7:0]							
		15:8	ACC[15:8]							
		23:16							ACC[17:16]	
0x0F6D	ADFLTR	7:0	FLTR[7:0]							
		15:8	FLTR[15:8]							
0x0F6F	CRCDAT	7:0	CRCDATL[7:0]							
		15:8	CRCDATH[7:0]							
0x0F71	CRCACC	7:0	CRCACCL[7:0]							
		15:8	CRCACCH[7:0]							
0x0F73	CRCSHIFT	7:0	CRCSHIFTL[7:0]							
		15:8	CRCSHIFTH[7:0]							
0x0F75	CRCXOR	7:0	CRCXORL[6:0]							CRCXORL0
		15:8	CRCXORH[7:0]							
0x0F77	CRCCON0	7:0	EN	GO	BUSY	ACCM			SHIFTM	FULL
0x0F78	CRCCON1	7:0	DLEN[3:0]				PLEN[3:0]			
0x0F79	NVMADR	7:0	NVMADRL[7:0]							
		15:8	NVMADRH[7:0]							
		23:16			NVMADRU[5:0]					
0x0F7C	NVMDAT	7:0	NVMDATL[7:0]							
		15:8	NVMDATH[7:0]							
0x0F7E	保留									
0x0F7F	NVMCON0	7:0	NVMEN			NVMERR	保留			
0x0F80	NVMCON1	7:0		SECER	SECWR	WR			SECRD	RD
0x0F81	NVMCON2	7:0	NVMCON2[7:0]							
0x0F82	LATA	7:0	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0
0x0F83	LATB	7:0	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
0x0F84	LATC	7:0	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0
0x0F85 ... 0x0F86	保留									
0x0F87	TRISA	7:0	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
0x0F88	TRISB	7:0	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
0x0F89	TRISC	7:0	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0

寄存器汇总（续）

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x0F8A ... 0x0F8B	保留									
0x0F8C	PORTA	7:0	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
0x0F8D	PORTB	7:0	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
0x0F8E	PORTC	7:0	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
0x0F8F	保留									
0x0F90	PORTE	7:0					RE3			
0x0F91	SSP1BUF	7:0	BUF[7:0]							
0x0F92	SSP1ADD	7:0	ADD[7:0]							
0x0F93	SSP1MSK	7:0	MSK[6:0]							MSK0
0x0F94	SSP1STAT	7:0	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
0x0F95	SSP1CON1	7:0	WCOL	SSPOV	SSPEN	CKP	SSPM[3:0]			
0x0F96	SSP1CON2	7:0	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
0x0F97	SSP1CON3	7:0	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
0x0F98	RC1REG	7:0	RCREG[7:0]							
0x0F99	TX1REG	7:0	TXREG[7:0]							
0x0F9A	SP1BRG	7:0	SPBRG[7:0]							
		15:8	SPBRG[15:8]							
0x0F9C	RC1STA	7:0	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
0x0F9D	TX1STA	7:0	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
0x0F9E	BAUD1CON	7:0	ABDOVF	RCIDL		SCKP	BRG16		WUE	ABDEN
0x0F9F	PWM4DC	7:0	DCL[1:0]							
		15:8	DCH[7:0]							
0x0FA1	PWM4CON	7:0	EN		OUT	POL				
0x0FA2	PWM3DC	7:0	DCL[1:0]							
		15:8	DCH[7:0]							
0x0FA4	PWM3CON	7:0	EN		OUT	POL				
0x0FA5	CCPR2	7:0	CCPR[7:0]							
		15:8	CCPR[15:8]							
0x0FA7	CCP2CON	7:0	EN		OUT	FMT	MODE[3:0]			
0x0FA8	CCP2CAP	7:0					CTS[3:0]			
0x0FA9	CCPR1	7:0	CCPR[7:0]							
		15:8	CCPR[15:8]							
0x0FAB	CCP1CON	7:0	EN		OUT	FMT	MODE[3:0]			
0x0FAC	CCP1CAP	7:0					CTS[3:0]			
0x0FAD	CCPTMRS	7:0	P4TSEL[1:0]		P3TSEL[1:0]		C2TSEL[1:0]		C1TSEL[1:0]	
0x0FAE	T6TMR	7:0	T6TMR[7:0]							
0x0FAF	T6PR	7:0	T6PR[7:0]							
0x0FB0	T6CON	7:0	ON		CKPS[2:0]		OUTPS[3:0]			
0x0FB1	T6HLT	7:0	PSYNC	CPOL	CSYNC	MODE[4:0]				
0x0FB2	T6CLKCON	7:0				CS[4:0]				
0x0FB3	T6RST	7:0				RSEL[4:0]				
0x0FB4	T4TMR	7:0	T4TMR[7:0]							
0x0FB5	T4PR	7:0	T4PR[7:0]							
0x0FB6	T4CON	7:0	ON		CKPS[2:0]		OUTPS[3:0]			
0x0FB7	T4HLT	7:0	PSYNC	CPOL	CSYNC	MODE[4:0]				
0x0FB8	T4CLKCON	7:0				CS[4:0]				
0x0FB9	T4RST	7:0				RSEL[4:0]				
0x0FBA	T2TMR	7:0	T2TMR[7:0]							
0x0FBB	T2PR	7:0	T2PR[7:0]							
0x0FBC	T2CON	7:0	ON		CKPS[2:0]		OUTPS[3:0]			
0x0FBD	T2HLT	7:0	PSYNC	CPOL	CSYNC	MODE[4:0]				
0x0FBE	T2CLKCON	7:0				CS[4:0]				
0x0FBF	T2RST	7:0				RSEL[4:0]				
0x0FC0	TMR5	7:0	TMR5[7:0]							
		15:8	TMR5[15:8]							
0x0FC2	T5CON	7:0			CKPS[1:0]			$\overline{SYNC}$	RD16	ON
0x0FC3	T5GCON	7:0	GE	GPOL	GTM	GSPM	GGO/$\overline{DONE}$	GVAL		
0x0FC4	T5GATE	7:0				GSS[4:0]				

寄存器汇总（续）

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x0FC5	T5CLK	7:0				CS[4:0]				
0x0FC6	TMR3	7:0	TMR3[7:0]							
		15:8	TMR3[15:8]							
0x0FC8	T3CON	7:0			CKPS[1:0]			SYNC	RD16	ON
0x0FC9	T3GCON	7:0	GE	GPOL	GTM	GSPM	GGO/DONE	GVAL		
0x0FCA	T3GATE	7:0				GSS[4:0]				
0x0FCB	T3CLK	7:0				CS[4:0]				
0x0FCC	TMR1	7:0	TMR1[7:0]							
		15:8	TMR1[15:8]							
0x0FCE	T1CON	7:0			CKPS[1:0]			SYNC	RD16	ON
0x0FCF	T1GCON	7:0	GE	GPOL	GTM	GSPM	GGO/DONE	GVAL		
0x0FD0	T1GATE	7:0				GSS[4:0]				
0x0FD1	T1CLK	7:0				CS[4:0]				
0x0FD2	TMR0L	7:0	TMR0L[7:0]							
0x0FD3	TMR0H	7:0	TMR0H[7:0]							
0x0FD4	T0CON0	7:0	EN		OUT	MD16	OUTPS[3:0]			
0x0FD5	T0CON1	7:0	CS[2:0]			ASYNC	CKPS[3:0]			
0x0FD6	PCON1	7:0						RVREG		RCM
0x0FD7	PCON0	7:0	STKOVF	STKUNF	WDTWV	RWDT	RMCLR	RI	POR	BOR
0x0FD8	STATUS	7:0		TO	PD	N	OV	Z	DC	C
0x0FD9	FSR2	7:0	FSRL[7:0]							
		15:8					FSRH[3:0]			
0x0FDB	PLUSW2	7:0	PLUSW[7:0]							
0x0FDC	PREINC2	7:0	PREINC[7:0]							
0x0FDD	POSTDEC2	7:0	POSTDEC[7:0]							
0x0FDE	POSTINC2	7:0	POSTINC[7:0]							
0x0FDF	INDF2	7:0	INDF[7:0]							
0x0FE0	BSR	7:0					BSR[3:0]			
0x0FE1	FSR1	7:0	FSRL[7:0]							
		15:8					FSRH[3:0]			
0x0FE3	PLUSW1	7:0	PLUSW[7:0]							
0x0FE4	PREINC1	7:0	PREINC[7:0]							
0x0FE5	POSTDEC1	7:0	POSTDEC[7:0]							
0x0FE6	POSTINC1	7:0	POSTINC[7:0]							
0x0FE7	INDF1	7:0	INDF[7:0]							
0x0FE8	WREG	7:0	WREG[7:0]							
0x0FE9	FSR0	7:0	FSRL[7:0]							
		15:8					FSRH[3:0]			
0x0FEB	PLUSW0	7:0	PLUSW[7:0]							
0x0FEC	PREINC0	7:0	PREINC[7:0]							
0x0FED	POSTDEC0	7:0	POSTDEC[7:0]							
0x0FEE	POSTINC0	7:0	POSTINC[7:0]							
0x0FEF	INDF0	7:0	INDF[7:0]							
0x0FF0 ... 0x0FF1	保留									
0x0FF2	INTCON	7:0	GIE/GIEH	PEIE/GIEL	IPEN			INT2EDG	INT1EDG	INT0EDG
0x0FF3	PROD	7:0	PRODL[7:0]							
		15:8	PRODH[7:0]							
0x0FF5	TABLAT	7:0	TABLAT[7:0]							
0x0FF6	TBLPTR	7:0	TBLPTRL[7:0]							
		15:8	TBLPTRH[7:0]							
		23:16			TBLPTR21	TBLPTRU[4:0]				
0x0FF9	PCL	7:0	PCL[7:0]							
0x0FFA	PCLAT	7:0	PCLATH[7:0]							
		15:8				PCLATU[4:0]				
0x0FFC	STKPTR	7:0				STKPTR[4:0]				
0x0FFD	TOS	7:0	TOSL[7:0]							
		15:8	TOSH[7:0]							
		23:16				TOSU[4:0]				

寄存器汇总（续）

偏移量	名称	位位置	7	6	5	4	3	2	1	0
0x1000 ... 0x2FFFFF	保留									
0x300000	CONFIG1	7:0		RSTOSC[2:0]				FEXTOSC[2:0]		
		15:8			FCMEN		CSWEN			$\overline{\text{CLKOUTEN}}$
0x300002	CONFIG2	7:0	BOREN[1:0]		$\overline{\text{LPBOREN}}$				$\overline{\text{PWRTE}}$	MCLRE
		15:8	$\overline{\text{XINST}}$		$\overline{\text{DEBUG}}$	STVREN	PPS1WAY	$\overline{\text{ZCD}}$	BORV[1:0]	
0x300004	CONFIG3	7:0		WDTE[1:0]				WDTCPS[4:0]		
		15:8				WDTCCS[2:0]			WDTCWS[2:0]	
0x300006	CONFIG4	7:0					WRT3	WRT2	WRT1	WRT0
		15:8			LVP	SCANE		WRTD	WRTB	WRTC
0x300008	CONFIG5	7:0							$\overline{\text{CPD}}$	$\overline{\text{CP}}$
		15:8								
0x30000A	CONFIG6	7:0					EBTR3	EBTR2	EBTR1	EBTR0
		15:8							EBTRB	
0x30000C ... 0x3FFFFB	保留									
0x3FFFFC	版本 ID	7:0	MJRREV[1:0]		MNRREV[5:0]					
		15:8	1010[3:0]				MJRREV[5:2]			
0x3FFFFE	器件 ID	7:0	DEV[7:0]							
		15:8	DEV[15:8]							

42. 电气规范

42.1. 绝对最大额定值

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(1)		
• **V_{SS} 引脚**	-40°C ≤ T_A ≤ +85°C	350 mA
• **V_{DD} 引脚（28 引脚器件）**	-40°C ≤ T_A ≤ +85°C	250 mA
• **任何标准 I/O 引脚**		±50 mA
钳位电流，I_K（V_{PIN} < 0 或 V_{PIN} > V_{DD}）		±20 mA
总功耗(2)		800 mW

重要：

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，请参见热特性表来计算器件规范值。
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。

42.2. 标准工作条件

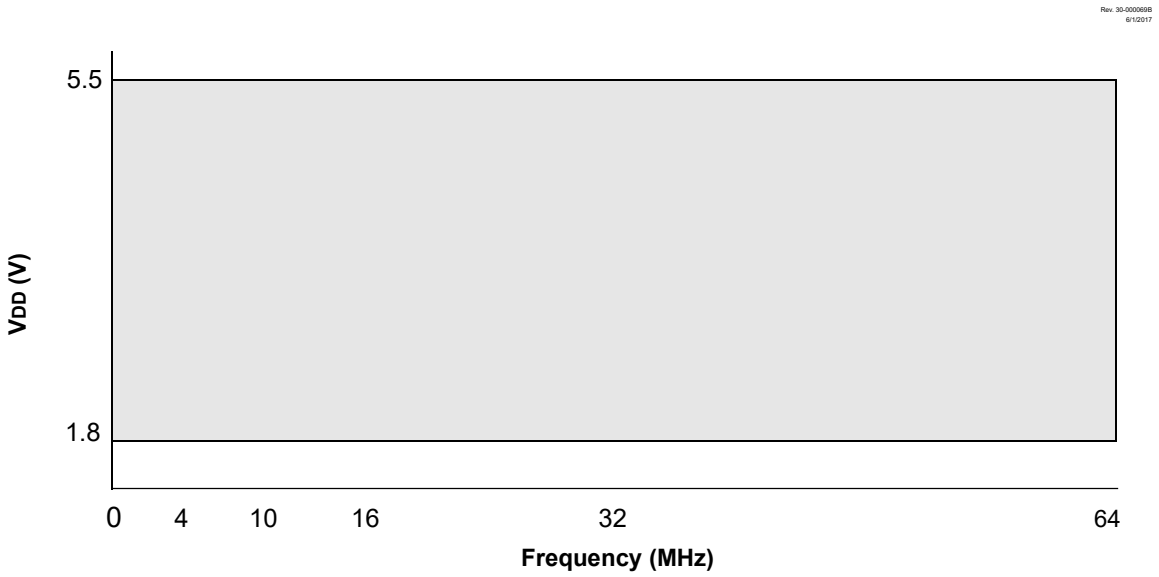
器件的标准工作条件定义如下：

工作电压： $V_{DDMIN} \leqslant V_{DD} \leqslant V_{DDMAX}$

工作温度： $T_{A\_MIN} \leqslant T_A \leqslant T_{A\_MAX}$

参数		额定值
V_{DD}——工作电源电压(1)	V_{DDMIN}	+1.8V
	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

图 42-1. 电压—频率关系图，-40°C ≤ T_A ≤ +85°C



注：

1. 阴影区域表示允许的电压和频率的组合。
2. 请参见外部时钟/振荡器时序要求了解每种振荡器模式所支持的频率。

42.3. 直流特性

42.3.1. 电源电压

表 42-1.

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	
RAM 数据保持电压(1)							
D003	V_{DR}		1.7	—	—	V	器件处于休眠模式
上电复位释放电压(2)							

表 42-1.（续）

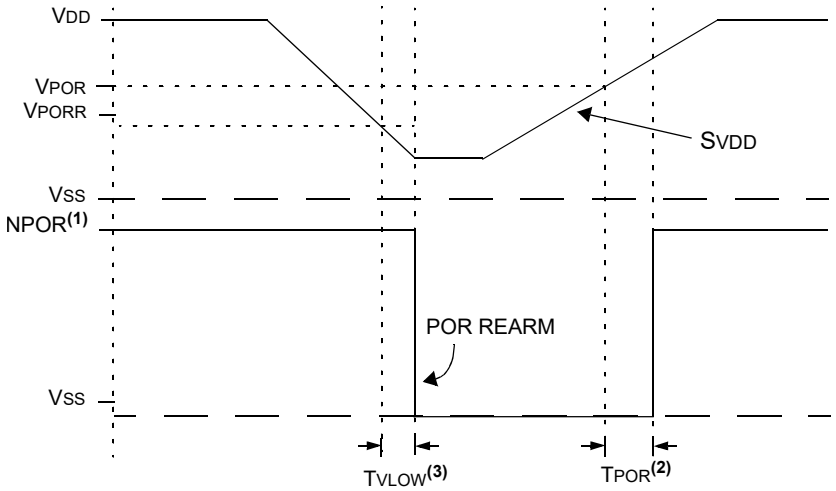
标准工作条件（除非另外声明）							
参数编号	**符号**	**特性**	**最小值**	**典型值†**	**最大值**	**单位**	**条件**
D004	V_{POR}		—	1.6	—	V	禁止 BOR 和 LPBOR(3)
上电复位重新激活电压(2)							
D005	V_{PORR}		—	1	—	V	禁止 BOR 和 LPBOR(3)
确保内部上电复位信号的 V_{DD} 上升速率(2)							
D006	S_{VDD}		0.05	—	—	V/ms	禁止 BOR 和 LPBOR(3)

†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 和 LPBOR 跳变点的信息，请参见复位、WDT、振荡器起振定时器、上电定时器、欠压复位和低功耗欠压复位规范。

图 42-2. 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。
2. T_{POR} 典型值为 1 µs。
3. T_{VLOW} 典型值为 2.7 µs。

42.3.2. 供电电流（I_{DD}）(1,2,4)

表 42-2.

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**
D100	$I_{DD_{XT4}}$	XT = 4 MHz	—	490	—	µA	3.0V	
D100A	$I_{DD_{XT4}}$	XT = 4 MHz	—	410	—	µA	3.0V	所有 PMD 位均为 1
D101	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	1.3	—	mA	3.0V	
D101A	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	1.1	—	mA	3.0V	所有 PMD 位均为 1
D102	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	4.4	—	mA	3.0V	
D102A	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	3.4	—	mA	3.0V	所有 PMD 位均为 1
D103	$I_{DD_{HSPLL64}}$	HS+PLL = 64 MHz	—	4.3	—	mA	3.0V	

表 42-2.（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_{DD}	注
D103A	$I_{DD_{HSPLL64}}$	HS+PLL = 64 MHz	—	3.3	—	mA	3.0V	所有 PMD 位均为 1
D104	$I_{DD_{IDLE}}$	空闲模式，HFINTOSC = 16 MHz	—	0.6	—	mA	3.0V	
D105	$I_{DD_{DOZE}}$ **(3)**	打盹模式，HFINTOSC = 16 MHz，打盹模式时钟分频比 = 16	—	0.7	—	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：OSC1 = 外部方波，轨到轨满幅；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。
3. $I_{DD_{DOZE}}$ = [$I_{DD_{IDLE}}$*(N-1)/N] + $I_{DD_{HFO}}$16/N，其中 N = 打盹模式时钟分频比（见 *CPUDOZE* 寄存器）。
4. PMD 位均处于默认状态，不禁止任何模块。

42.3.3. 掉电电流（I_{PD}）(1,2)

表 42-3.

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}	VREGPM	注
D200	I_{PD}	基本 I_{PD} 电流	—	—	—	—	—	b'11'	保留
D200A	I_{PD}	基本 I_{PD} 电流	—	0.6	—	μA	3.0V	b'10'	
D200B			—	50	—	μA	3.0V	b'01'	
D200C			—	170	—	μA	3.0V	b'00'	
D201	I_{PD_WDT}	低频内部振荡器/WDT	—	0.9	—	μA	3.0V	b'10'	
D202*	I_{PD_SOSC}	辅助振荡器（S_{OSC}）	—	1.6	—	μA	3.0V	b'10'	SOSCPWR = 0
D203	I_{PD_LPBOR}	低功耗欠压复位（LPBOR）	—	0.9	—	μA	3.0V	b'10'	
D204	I_{PD_FVR}	FVR	—	70	—	μA	3.0V	b'10'或 b'01'	FVRCON = 0x81 或 0x84
D205	I_{PD_BOR}	欠压复位（BOR）	—	30	—	μA	3.0V	b'10'	
D206	I_{PD_HLVD}	高/低电压检测（HLVD）	—	22	—	μA	3.0V	b'10'	
D207	I_{PD_ADCA}	ADC——工作	—	330	—	μA	3.0V	b'10'或 b'01'	ADC 正在进行转换**(4)**
D208	I_{PD_CMP}	比较器	—	60	—	μA	3.0V	b'10'	

表 42-3.（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 +85°C	单位	条件		
							V_{DD}	VREGPM	注

* 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. 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，则列出的所有外设电流均是基于每个外设的。
4. ADC 时钟源为 FRC。

42.3.4. I/O 端口

表 42-4.

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
输入低电平电压							
	V_{IL}	I/O 端口：					
D300		• 带 TTL 缓冲器	—	—	0.8	V	4.5V≤V_{DD}≤5.5V
D301			—	—	0.15 V_{DD}	V	1.8V≤V_{DD}<4.5V
D302		• 带施密特触发缓冲器	—	—	0.2 V_{DD}	V	2.0V≤V_{DD}≤5.5V
D303		• 带 I^2C 电平	—	—	0.3 V_{DD}	V	
D304		• 带 SMBus 电平	—	—	0.8	V	2.7V≤V_{DD}≤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≤V_{DD}≤5.5V
D321			0.25 V_{DD}+0.8	—	—	V	1.8V≤V_{DD}<4.5V
D322		• 带施密特触发缓冲器	0.8V_{DD}	—	—	V	2.0V≤V_{DD}≤5.5V
D323		• 带 I^2C 电平	0.7 V_{DD}	—	—	V	
D324		• 带 SMBus 电平	2.1	—	—	V	2.7V≤V_{DD}≤5.5V
D325		$\overline{MCLR}$	0.7 V_{DD}	—	—	V	
输入泄漏电流(1)							
D340	I_{IL}	I/O 端口	—	±5	—	nA	V_{SS}≤V_{PIN}≤V_{DD}，引脚处于高阻态，85°C
D342		$\overline{MCLR}$(2)	—	±50	—	nA	V_{SS}≤V_{PIN}≤V_{DD}，引脚处于高阻态，85°C
弱上拉电流							
D350	I_{PUR}		—	140	—	μA	V_{DD}=3.0V，V_{PIN}=V_{SS}
输出低电平电压							
D360	V_{OL}	I/O 端口	—	—	0.6	V	I_{OL}=10.0 mA，V_{DD}=3.0V
输出高电平电压							
D370	V_{OH}	I/O 端口	V_{DD}-0.7	—	—	V	I_{OH}=6.0 mA，V_{DD}=3.0V
所有 I/O 引脚							
D380	C_{IO}		—	5	—	pF	

表 42-4.（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

† 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。

42.3.5. 存储器编程规范

表 42-5.

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	单位	条件
数据 EEPROM 存储器规范							
MEM20	E_D	数据 EEPROM 字节耐擦写能力	100k	—	—	E/W	-40°C≤T_A≤+85°C
MEM21	T_{D_RET}	特性保持时间	—	40	—	年	假设未违反其他规范
MEM22	N_{D_REF}	刷新前的总擦除/写次数	—	10M	—	E/W	-40°C≤ T_A≤+85°C
MEM23	V_{D_RW}	用于读或擦除/写操作的 V_{DD}	V_{DDMIN}	—	V_{DDMAX}	V	
MEM24	T_{D_BEW}	字节擦除和写周期时间	—	10	—	ms	
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≤Ta≤+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}	自定时扇区写入	—	6	—	ms	
MEM36	T_{SE}	自定时扇区擦除	—	10	—	ms	
MEM37	T_{P_WRD}	自定时字写入	—	50	—	μs	

† 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：一个行擦除操作和一个自定时写操作。

42.3.6. 热特性

表 42-6.

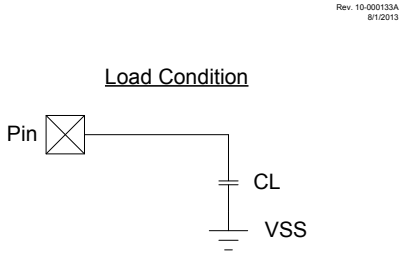
标准工作条件（除非另外声明）					
参数编号	符号	特性	典型值	单位	条件
TH01	θ_{JA}	结到环境的热阻	90	°C/W	28 引脚 SSOP 封装
TH03	T_{JMAX}	最高结温	150	°C	

注：

1. 有关总功耗，请参见*绝对最大额定值*。

42.4. 交流特性

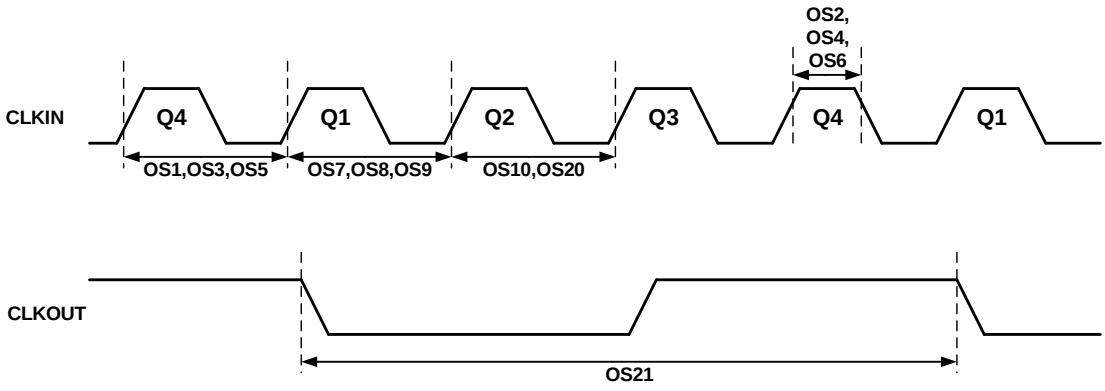
图 42-3. 负载条件



Legend: CL = 50 pF for all pins

42.4.1. 外部时钟/振荡器时序要求

图 42-4. 时钟时序



注：请参见下表。

表 42-7.

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 外部时钟							
OS1	F_{ECL}	时钟频率	—	—	1	MHz	
OS2	T_{ECL_DC}	时钟占空比	40	—	60	%	
ECM 外部时钟							
OS3	F_{ECM}	时钟频率	—	—	16	MHz	
OS4	T_{ECM_DC}	时钟占空比	40	—	60	%	
ECH 外部时钟							
OS5	F_{ECH}	时钟频率	—	—	64	MHz	
OS6	T_{ECH_DC}	时钟占空比	40	—	60	%	
LP 振荡器							
OS7	F_{LP}	时钟频率	—	—	100	kHz	**注 4**

表 42-7.（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**
XT 振荡器							
OS8	F_{XT}	时钟频率	—	—	4	MHz	**注 4**
HS 振荡器							
OS9	F_{HS}	时钟频率	—	—	20	MHz	**V_{DD}>2.5V（注 4）**
辅助振荡器							
OS10	F_{SEC}	时钟频率	—	32.768	—	kHz	**注 4**
系统振荡器							
OS20	F_{OSC}	系统时钟频率	—	—	64	MHz	（**注 2 和注 3**）
OS21	F_{CY}	指令频率	—	F_{OSC}/4	—	MHz	
OS22	T_{CY}	指令周期	62.5	1/F_{CY}	—	ns	

注：

1. 指令周期（TCY）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 OSC1 引脚连接了外部时钟。当使用了外部时钟输入时，所有器件的“最大”周期时间限制为“DC”（没有时钟）。
2. 系统时钟频率（FOSC）通过“主时钟切换控制”选择，如*“节能工作模式”*部分所述。
3. 系统时钟频率（FOSC）必须满足*标准工作条件*部分中定义的电压要求。
4. LP、XT 和 HS 振荡器模式要求将一个适当的晶振或谐振器连接到器件。当通过外部方波为器件提供时钟时，必须选择其中一种 EC 模式。

42.4.2. 内部振荡器参数(1)

表 42-8.

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 12 16 32 48 64	—	MHz	（**注 1 和注 2**）
OS51	$F_{HFOSCLP}$	针对低功耗优化的 HFINTOSC 频率	— —	1 2	— —	MHz MHz	基频 1 MHz 基频 2 MHz
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(3)从休眠模式唤醒的起振时间	— —	14 100	— —	µs µs	VREGPM =0x VREGPM =1x 系统时钟为 4 MHz
OS56	$T_{LFOSCST}$	LFINTOSC 从休眠模式唤醒的起振时间	—	0.3	—	ms	

表 42-8.（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**

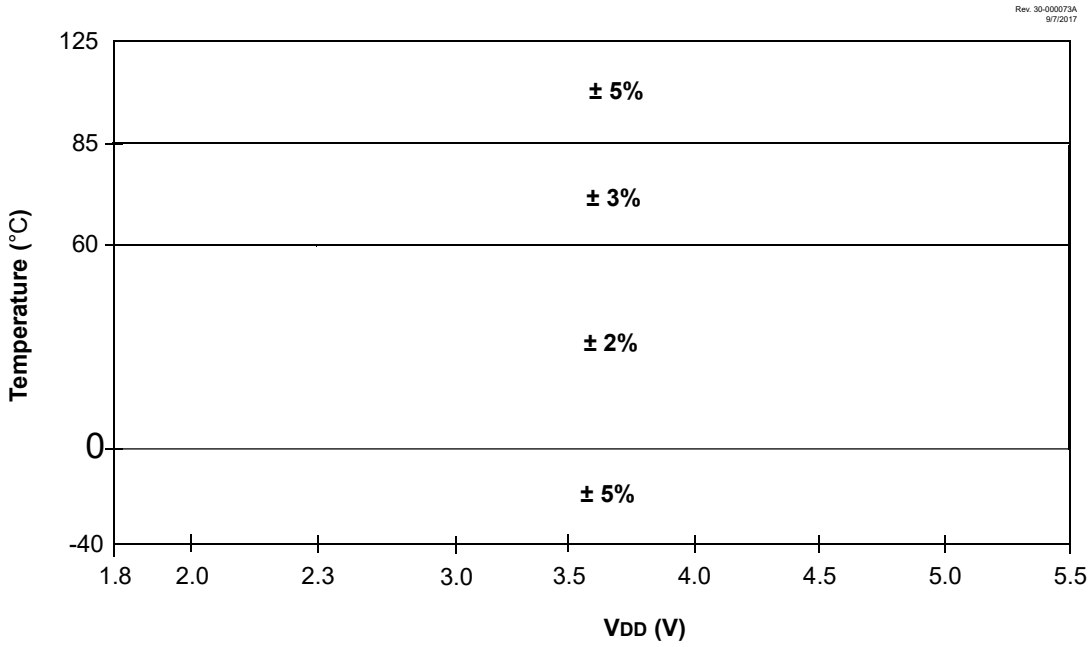
***** 这些参数通过表征确定，但未经测试。

† 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. 为了确保振荡器频率容差，必须尽可能靠近器件、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. 请参见下图。
3. HFINTOSC 时钟 = 4 MHz

图 42-5. 器件 V_{DD} 和温度范围内的 HFINTOSC 频率精度（已校准）



42.4.3. PLL 规范

表 42-9.

标准工作条件（除非另外声明） V_{DD} > = 2.5V							
参数编号	**符号**	**特性**	**最小值**	**典型值†**	**最大值**	**单位**	**条件**
PLL01	F_{PLLIN}	PLL 输入频率范围	4	—	16	MHz	
PLL02	F_{PLLOUT}	PLL 输出频率范围	16	—	64	MHz	（**注 1**）
PLL03*	F_{PLLST}	PLL 自起振的锁定时间	—	200	—	µs	
PLL04*	F_{PLLJIT}	PLL 输出频率稳定性（抗抖动性）	-0.25	—	0.25	%	

***** 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. PLL 的输出频率必须满足参数 D002 中列出的 F_{OSC} 要求。

42.4.4. I/O 和 CLKOUT 时序规范

图 42-6. CLKOUT 和 I/O 时序

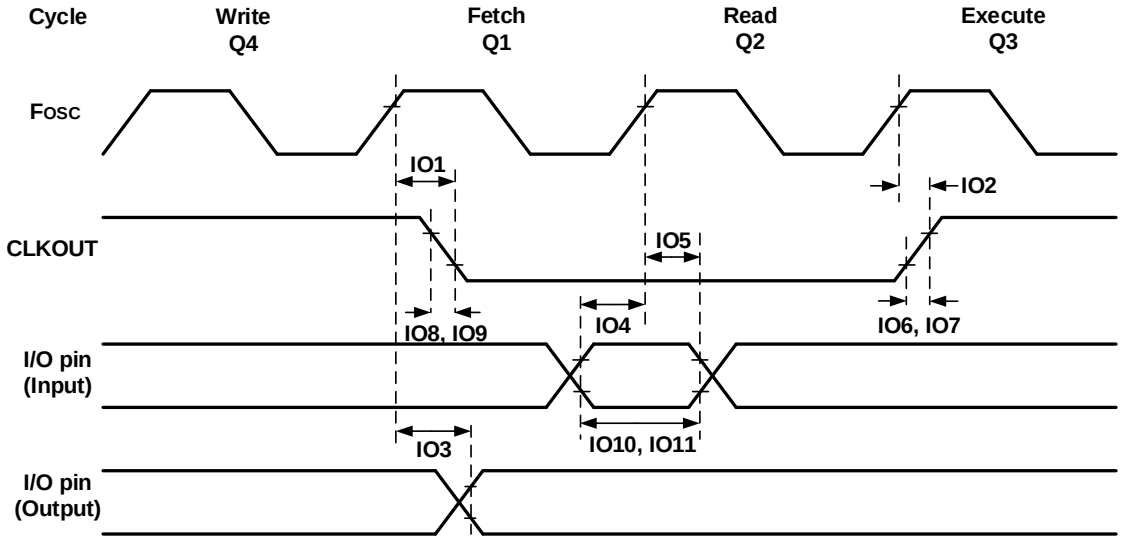
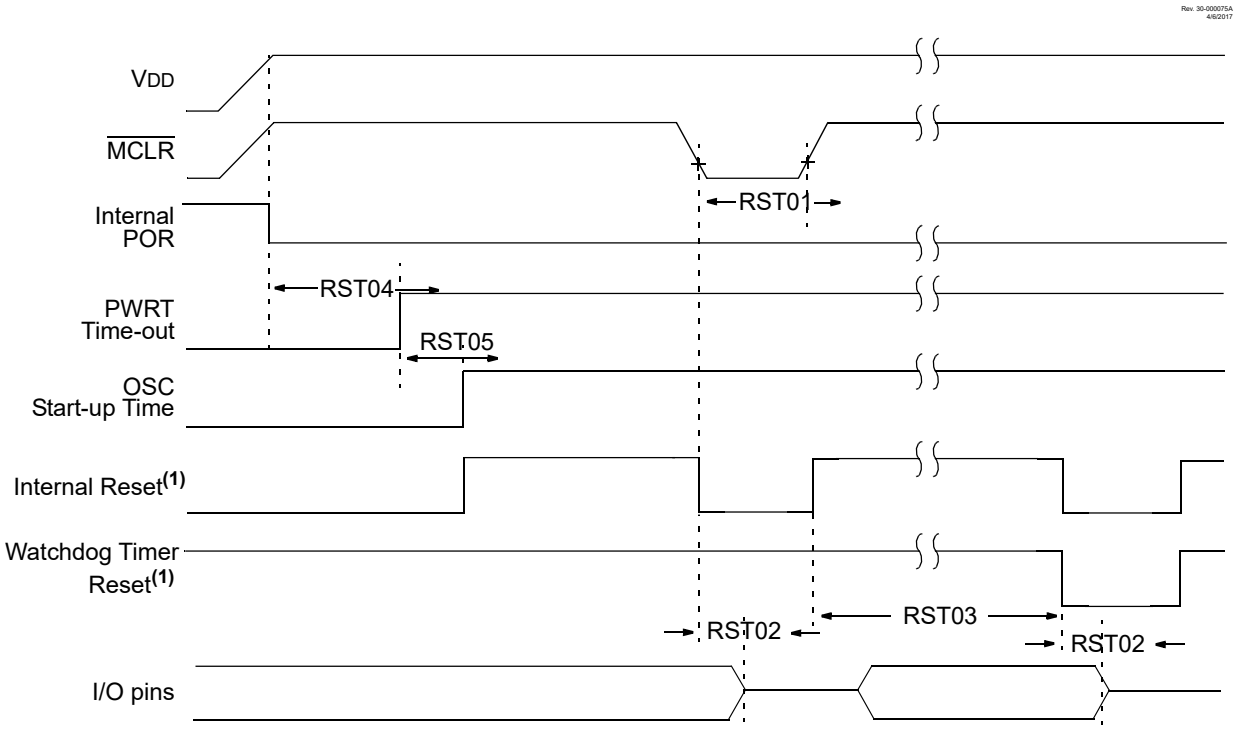


表 42-10. I/O 和 CLKOUT 时序规范

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	70	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。							

42.4.5. 复位、WDT、振荡器起振定时器、上电定时器、欠压复位和低功耗欠压复位规范

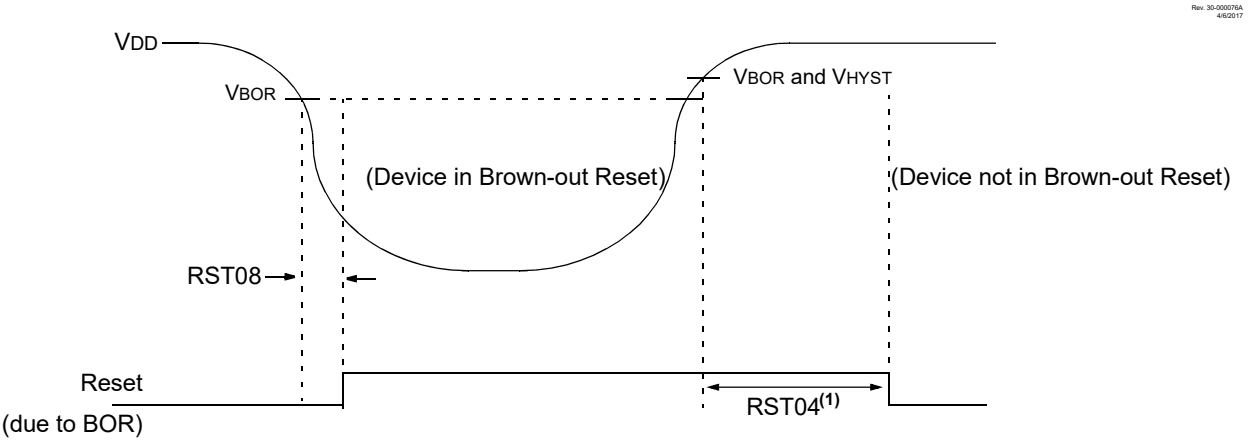
图 42-7. 复位、看门狗定时器、振荡器起振定时器和上电延时定时器时序



注：

1. 低电平有效。

图 42-8. 欠压复位时序和特性



注：

1. 仅当配置字寄存器中的 $\overline{\text{PWRTE}}$ 位被设置为 1 时；如果 $\overline{\text{PWRTE}}$= 0，则为 2 ms 延时。

表 42-11.

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

表 42-11.（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**
RST04*	T_{PWRT}	上电延时定时器周期	—	65	—	ms	
RST05	T_{OST}	振荡器起振定时器周期(1,2)	—	1024	—	T_{OSC}	
RST06	V_{BOR}	欠压复位电压	—	2.85	—	V	BORV = 00
			—	2.7	—	V	BORV = 01
			—	2.45	—	V	BORV = 10
			—	1.9	—	V	BORV = 11
RST07	V_{BORHYS}	欠压复位滞后	—	60	—	mV	BORV = 00
RST08	T_{BORDC}	欠压复位响应时间	—	3	—	μs	
RST09	V_{LPBOR}	低功耗欠压复位电压	—	1.9	—	V	

* 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. 根据设计，振荡器起振定时器（OST）在提供系统频率之前首先计数 1024 个周期，与频率无关。
2. 为了确保这些电压容差，必须尽可能靠近器件、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。

42.4.6. 高/低电压检测特性

表 42-12.

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**
HLVD01	V_{DET}	电压检测	—	1.90	—	V	HLVDSEL = b'0000'
			—	2.10	—	V	HLVDSEL = b'0001'
			—	2.25	—	V	HLVDSEL = b'0010'
			—	2.50	—	V	HLVDSEL = b'0011'
			—	2.60	—	V	HLVDSEL = b'0100'
			—	2.75	—	V	HLVDSEL = b'0101'
			—	2.90	—	V	HLVDSEL = b'0110'
			—	3.15	—	V	HLVDSEL = b'0111'
			—	3.35	—	V	HLVDSEL = b'1000'
			—	3.60	—	V	HLVDSEL = b'1001'
			—	3.75	—	V	HLVDSEL = b'1010'
			—	4.00	—	V	HLVDSEL = b'1011'
			—	4.20	—	V	HLVDSEL = b'1100'
			—	4.35	—	V	HLVDSEL = b'1101'
			—	4.65	—	V	HLVDSEL = b'1110'

42.4.7. 模数转换器（ADC）精度规范(1,2)

表 42-13.

标准工作条件（除非另外声明） V_{DD} = 3.0V，T_A = 25°C，T_{AD} = 1μs							
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	—	±0.1	—	LSb	ADC_{REF}+ = 3.0V，ADC_{REF}- = 0V
AD03	E_{DL}	微分误差	—	±0.1	—	LSb	ADC_{REF}+ = 3.0V，ADC_{REF}- = 0V
AD04	E_{OFF}	失调误差	—	0.5	—	LSb	ADC_{REF}+ = 3.0V，ADC_{REF}- = 0V
AD05	E_{GN}	增益误差	—	±0.2	—	LSb	ADC_{REF}+ = 3.0V，ADC_{REF}- = 0V
AD06	V_{ADREF}	ADC 参考电压（AD_{REF}+ - AD_{REF}-）	1.8	—	V_{DD}	V	

表 42-13.（续）

标准工作条件（除非另外声明） V_{DD} = 3.0V，T_A = 25°C，T_{AD} = 1μs							
参数编号	**符号**	**特性**	**最小值**	**典型值†**	**最大值**	**单位**	**条件**
AD07	V_{AIN}	满量程范围	AD_{REF}-	—	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	（**注 3**）

* 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。
3. 这是选择外部参考焊盘时 V_{REF} 焊盘的阻抗。

42.4.8. 模数转换器（ADC）转换时序规范

表 42-14.

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	1	—	9	μs	使用 F_{OSC} 作为 ADC 时钟源，ADOCS = 0
AD21			1	2	6	μs	使用 F_{RC} 作为 ADC 时钟源，ADOCS = 1
AD22	T_{CNV}	转换时间(1)	—	11+3T_{CY}	—	T_{AD}	GO/$\overline{DONE}$ 位置 1 到 GO/$\overline{DONE}$ 位清零的时间
AD23	T_{ACQ}	采集时间	—	2	—	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。

注：

1. 不适用于 ADCRC 振荡器。

图 42-9. ADC 转换时序（ADC 时钟基于 F_{OSC}）

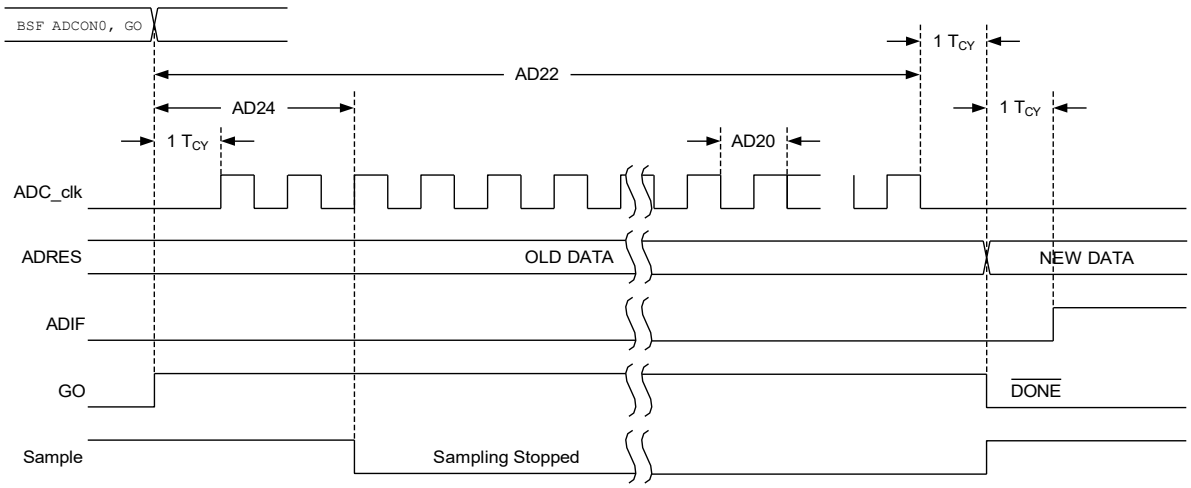
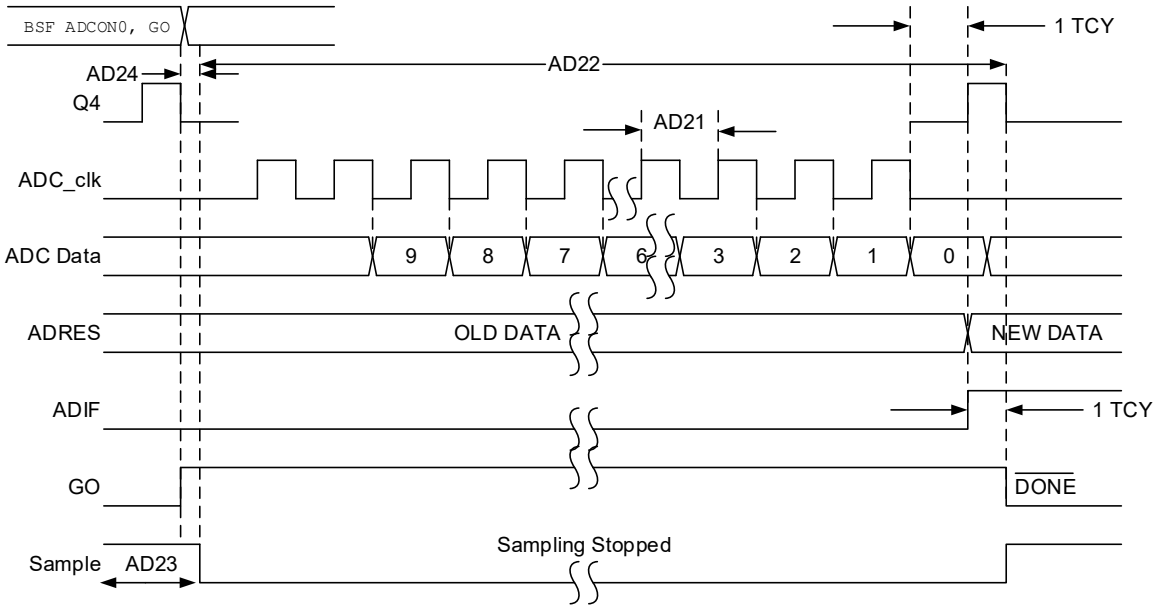


图 42-10. ADC 转换时序（ADC 时钟来自 F_{RC}）



注：

1. 如果选择 F_{RC} 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。

42.4.9. 比较器规范

表 42-15.

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**
CM01	V_{IOFF}	输入失调电压	—	—	±30	mV	$V_{ICM} = V_{DD}/2$
CM02	V_{ICM}	输入共模范围	GND	—	V_{DD}	V	
CM03	CMRR	共模输入抑制比	—	50	—	dB	
CM04	V_{HYST}	比较器滞后	—	25	—	mV	
CM05	T_{RESP}(1)	响应时间上升沿	—	300	—	ns	
		响应时间下降沿	—	220	—	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。

注：

1. 响应时间是在比较器的一个输入端电压为 $V_{DD}/2$，而另一个输入端从 V_{SS} 跳变到 V_{DD} 时测得的。

42.4.10. 5 位 DAC 规范

表 42-16.

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**
DSB01	V_{LSB}	步长	—	$(V_{DACREF+}-V_{DACREF-})/32$	—	V	
DSB02	V_{ACC}	绝对精度	—	—	±0.5	LSb	
DSB03*	R_{UNIT}	单位电阻值	—	5000	—	Ω	

表 42-16.（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**
DSB04*	T_{ST}	稳定时间(1)	—	—	10	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。

注：

1. 稳定时间是在 DACR<4:0>从 00000 跳变到 01111 时测得的。

42.4.11. 固定参考电压（FVR）规范

表 42-17.

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**
FVR01	$V_{FVR}1$	1x 增益（1.024V）	-4	—	+4	%	VDD ≥ 2.5V，-40°C 至 85°C
FVR02	$V_{FVR}2$	2x 增益（2.048V）	-4	—	+4	%	VDD ≥ 2.5V，-40°C 至 85°C
FVR03	$V_{FVR}4$	4x 增益（4.096V）	-4	—	+4	%	VDD ≥ 4.75V，-40°C 至 85°C
FVR04	T_{FVRST}	FVR 启动时间	—	25	—	μs	

42.4.12. 过零检测（ZCD）规范

表 42-18.

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**
ZC01	V_{PINZC}	过零引脚上的电压	—	0.9	—	V	
ZC02	I_{ZCD_MAX}	最大拉电流或灌电流	—	—	600	μA	
ZC03	T_{RESPH}	响应时间上升沿	—	1	—	μs	
	T_{RESPL}	响应时间下降沿	—	1	—	μs	

† 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。

42.4.13. Timer0 和 Timer1 外部时钟要求

表 42-19.

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		取如下二者中较大值：20 或(T_{CY}+40)/N	—	—	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	

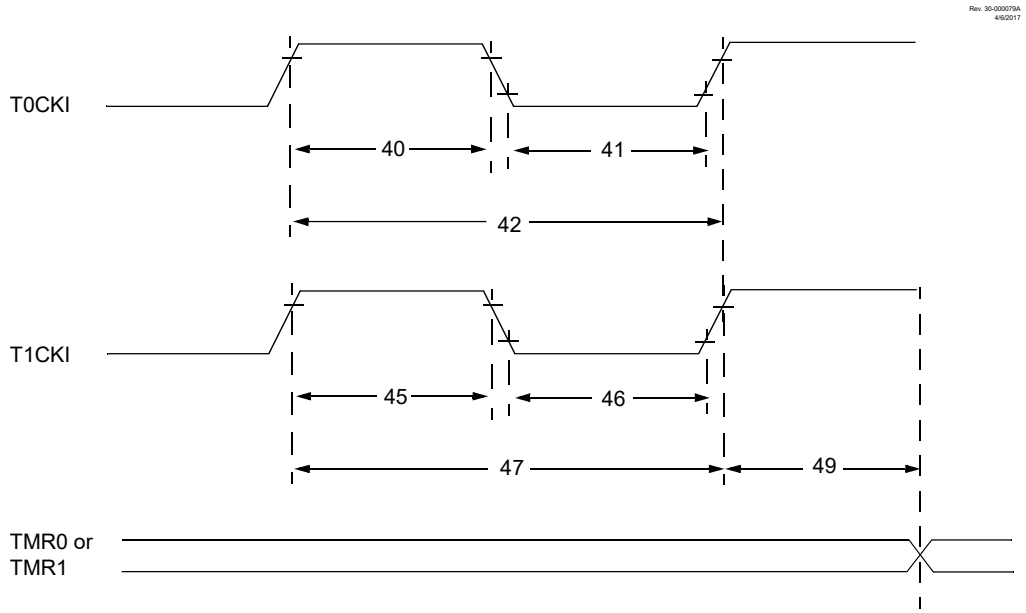
表 42-19.（续）

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**
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	取如下二者中较大值：30 或$(T_{CY}+40)/N$	—	—	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		2 T_{OSC}	—	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 42-11. Timer0 和 Timer1 外部时钟时序



42.4.14. 捕捉/比较/PWM（CCP）要求

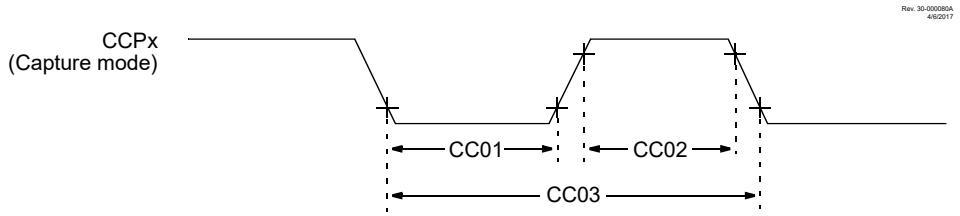
表 42-20.

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 42-12. 捕捉/比较/PWM（CCP）时序



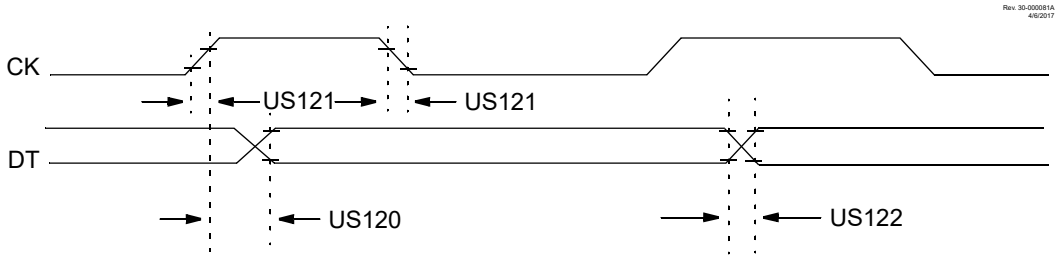
注：负载条件请参见图 42-3。

42.4.15. EUSART 同步发送要求

表 42-21.

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**
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	$3.0V \leqslant V_{DD} \leqslant 5.5V$
			—	100	ns	$1.8V \leqslant V_{DD} \leqslant 5.5V$
US121	T_{CKRF}	时钟输出上升和下降时间 （主模式）	—	45	ns	$3.0V \leqslant V_{DD} \leqslant 5.5V$
			—	50	ns	$1.8V \leqslant V_{DD} \leqslant 5.5V$
US122	T_{DTRF}	数据输出上升和下降时间	—	45	ns	$3.0V \leqslant V_{DD} \leqslant 5.5V$
			—	50	ns	$1.8V \leqslant V_{DD} \leqslant 5.5V$

图 42-13. EUSART 同步发送（主/从）时序



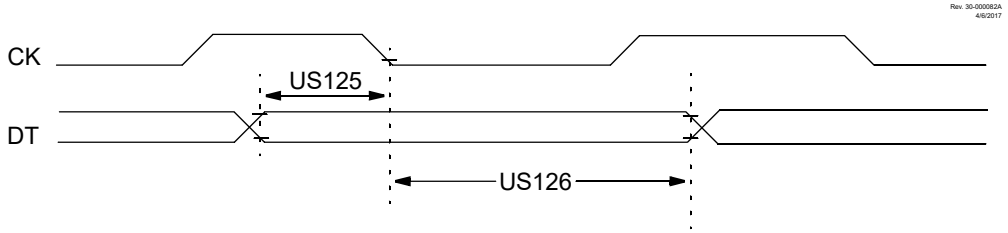
注：负载条件请参见图 42-3。

42.4.16. EUSART 同步接收要求

表 42-22.

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 42-14. EUSART 同步接收（主/从）时序



注：负载条件请参见图 42-3。

42.4.17. SPI 模式要求

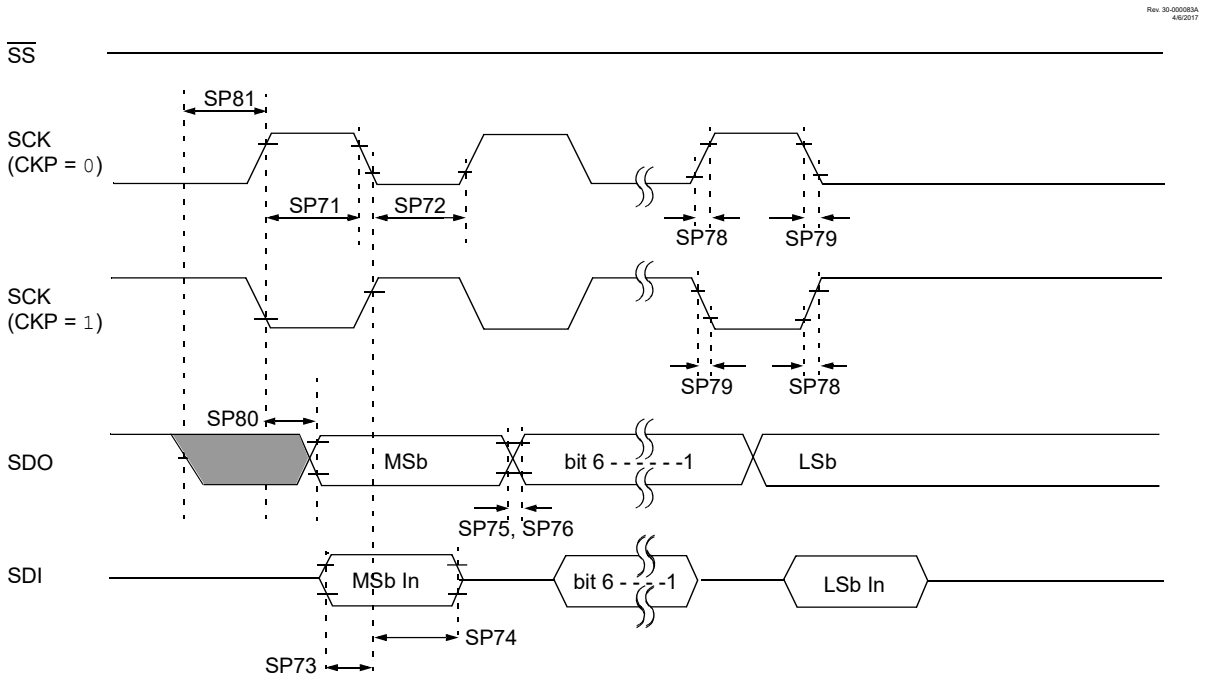
表 42-23.

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	
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	—	10	—	ns	3.0V≤V_{DD}≤5.5V
			—	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	10	—	50	ns	
SP78*	$T_{SC}R$	SCK 输出上升时间	—	10	—	ns	3.0V≤V_{DD}≤5.5V
			—	25	—	ns	1.8V≤V_{DD}≤5.5V
SP79*	$T_{SC}F$	SCK 输出下降时间	—	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	—	—	50	ns	3.0V≤V_{DD}≤5.5V
			—	—	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*	$T_{SC}H2_{SS}H$，$T_{SC}L2_{SS}H$	在 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。

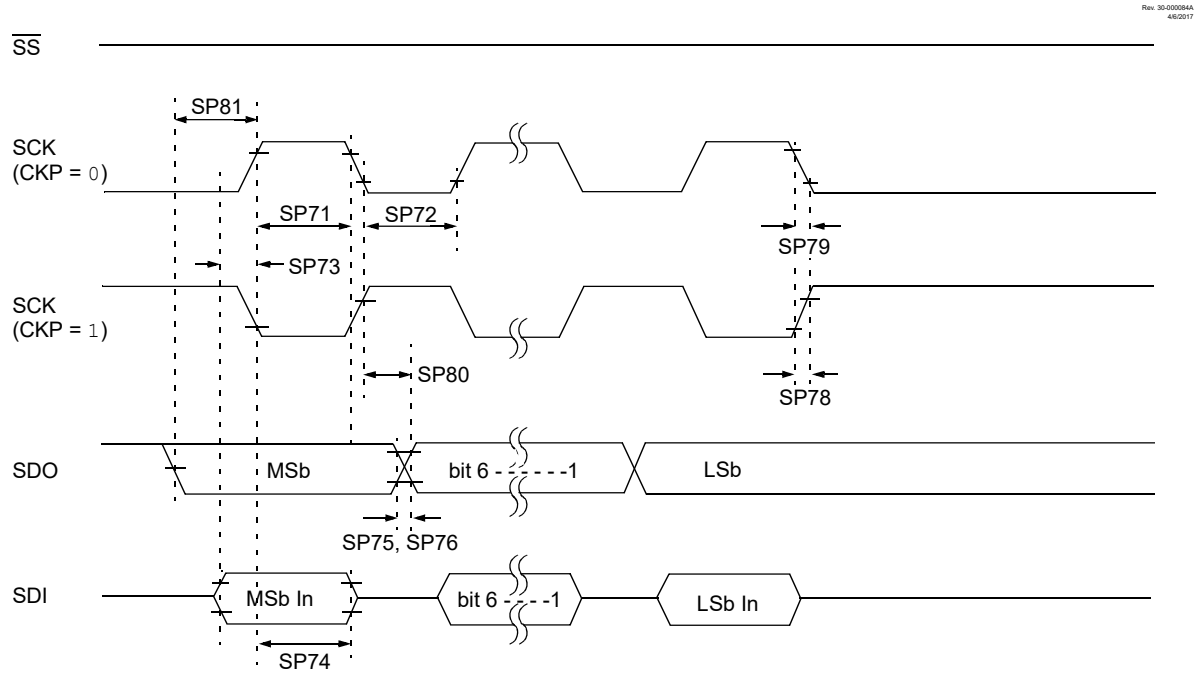
† 除非另外声明，否则“典型值”栏中的数据均为 3.0V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

图 42-15. SPI 主模式时序（CKE = 0，SMP = 0）



注：负载条件请参见图 42-3。

图 42-16. SPI 主模式时序（CKE = 1，SMP = 1）



注：负载条件请参见图 42-3。

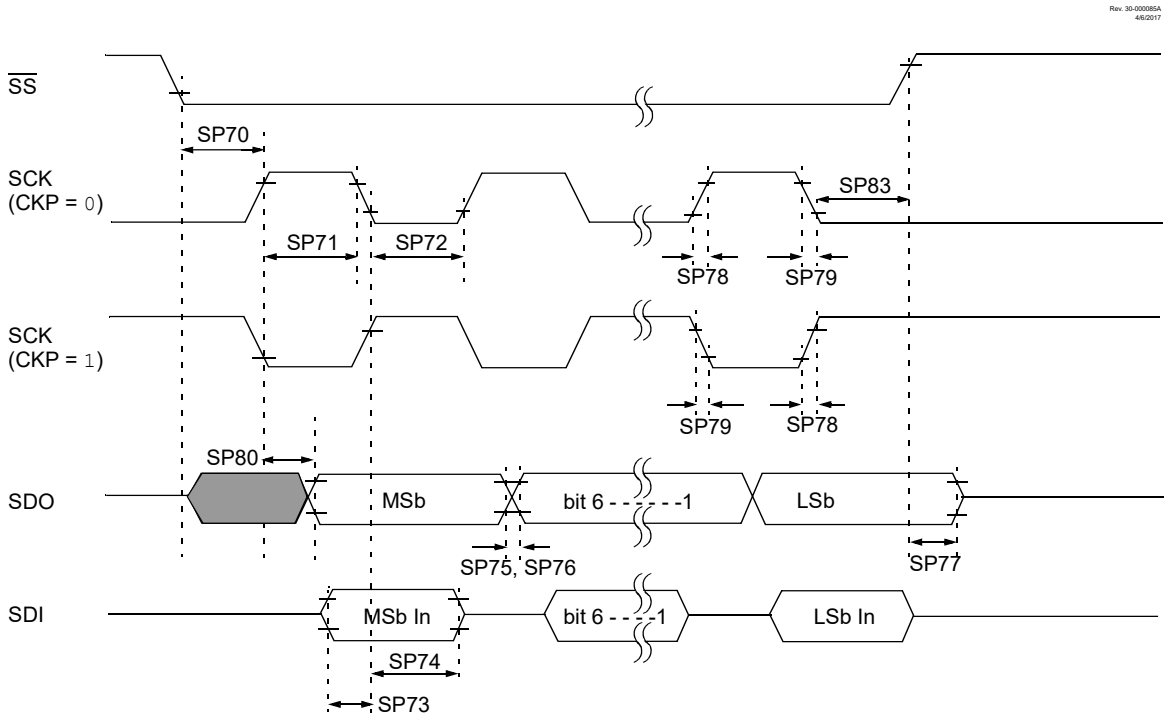
表 42-24.

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	
SP71*	$T_{SC}H$	SCK 输入高电平时间	T_{CY} + 20	—	—	ns	
SP72*	$T_{SC}L$	SCK 输入低电平时间	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	—	10	—	ns	3.0V≤V_{DD}≤5.5V
			—	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	10	—	50	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	—	—	50	ns	3.0V≤V_{DD}≤5.5V
			—	—	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*	$T_{SC}H2_{SS}H$，$T_{SC}L2_{SS}H$	在 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。

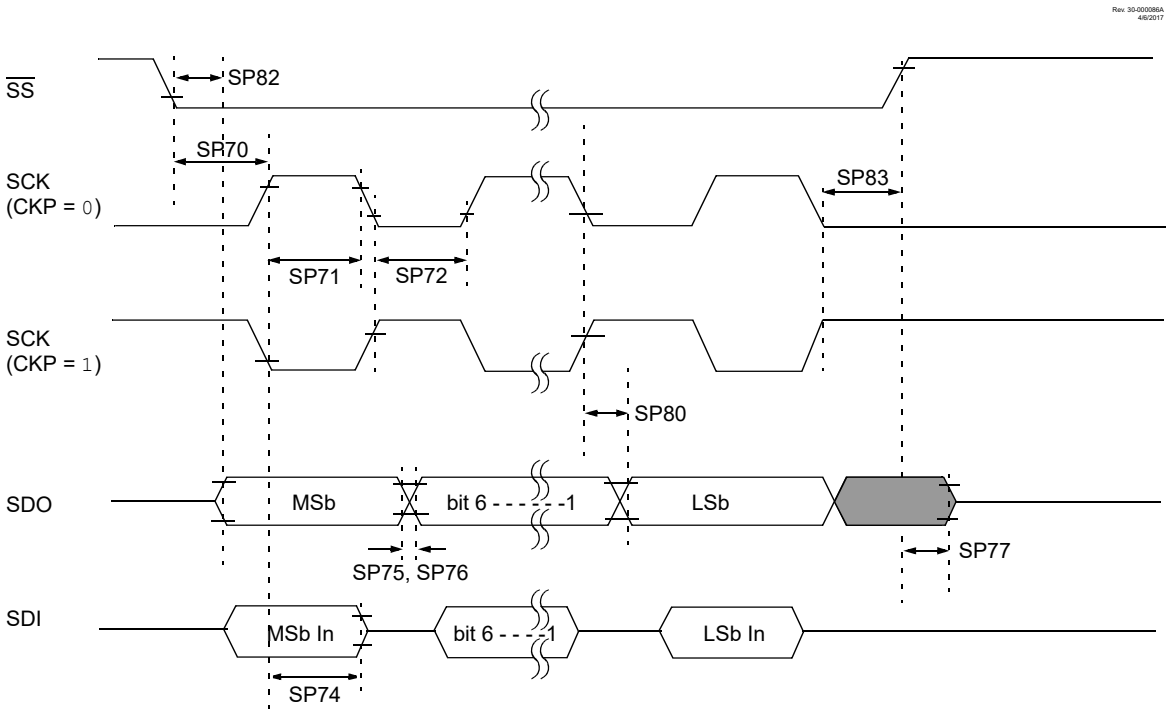
† 除非另外声明，否则“典型值”栏中的数据均为 3.0V 和 25°C 条件下的值。这些参数仅供设计参考，未经测试。

图 42-17. SPI 从模式时序（CKE = 0）



注：负载条件请参见图 42-3。

图 42-18. SPI 从模式时序（CKE = 1）



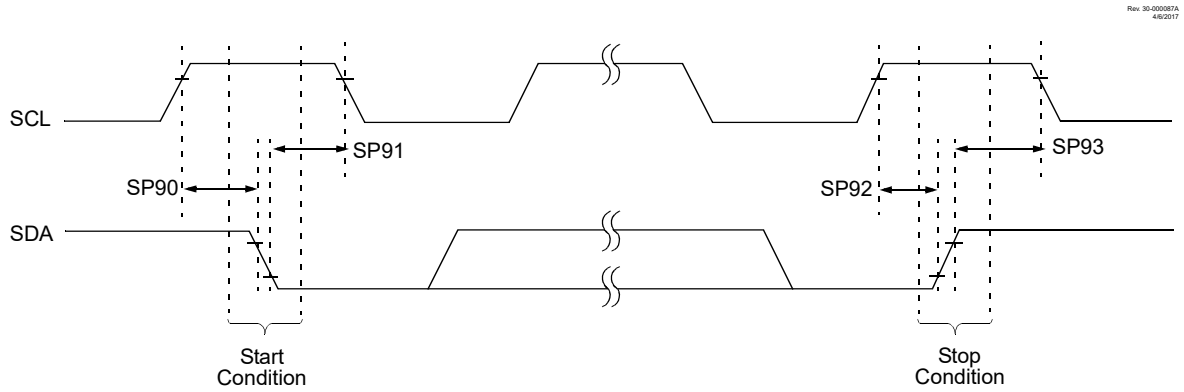
注：负载条件请参见图 42-3。

42.4.18. I²C 总线启动位/停止位要求

表 42-25.

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	仅与重复启动条件相关
			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 42-19. I²C 总线启动/停止位时序



注：负载条件请参见图 42-3。

42.4.19. I²C 总线数据要求

表 42-26.

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

表 42-26.（续）

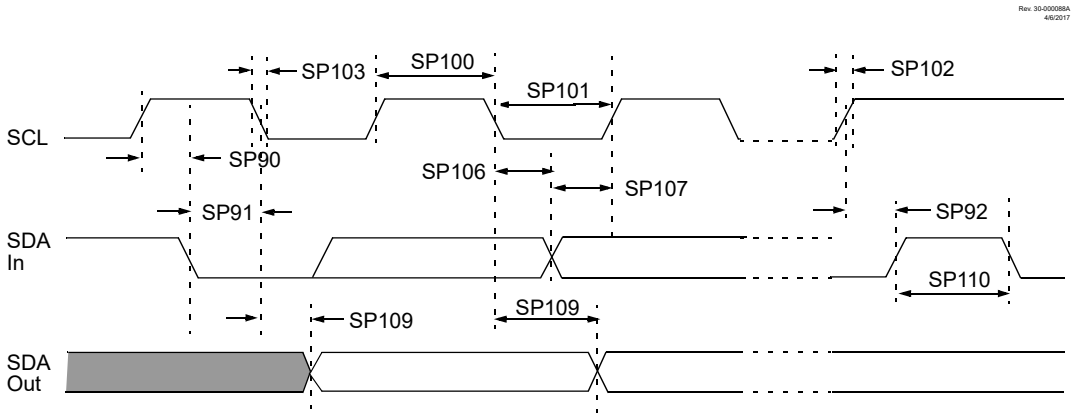
标准工作条件（除非另外声明）							
参数编号	**符号**	**特性**		**最小值**	**最大值**	**单位**	**条件**
SP103*	T_F	SDA 和 SCL 下降时间	100 kHz 模式	—	250	ns	
			400 kHz 模式	20 + 0.1C_B	250	ns	C_B 值规定在 10-400 pF 之间
SP106*	$T_{HD:DAT}$	数据输入保持时间	100 kHz 模式	0	—	ns	
			400 kHz 模式	0	0.9	µs	
SP107*	$T_{SU:DAT}$	数据输入建立时间	100 kHz 模式	250	—	ns	（**注 2**）
			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		—		pF	

* 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 $T_{SU:DAT}$≥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 42-20. I²C 总线数据时序



注：负载条件请参见图 42-3。

42.4.20. 可配置逻辑单元（CLC）特性

表 42-27.

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**
CLC01*	T_{CLCIN}	CLC 输入时间		—	7	—	ns	（**注 1**）
CLC02*	T_{CLC}	CLC 模块输入至输出的传播时间		—	24	—	ns	V_{DD} = 1.8V
				—	12	—	ns	V_{DD} > 3.6V
CLC03*	T_{CLCOUT}	CLC 输出时间	上升时间	—	OS18	—	—	（**注 1**）
			下降时间	—	OS19	—	—	（**注 1**）

表 42-27.（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**
CLC04*	F_{CLCMAX}	CLC 最大开关频率	—	—	OS20	—	

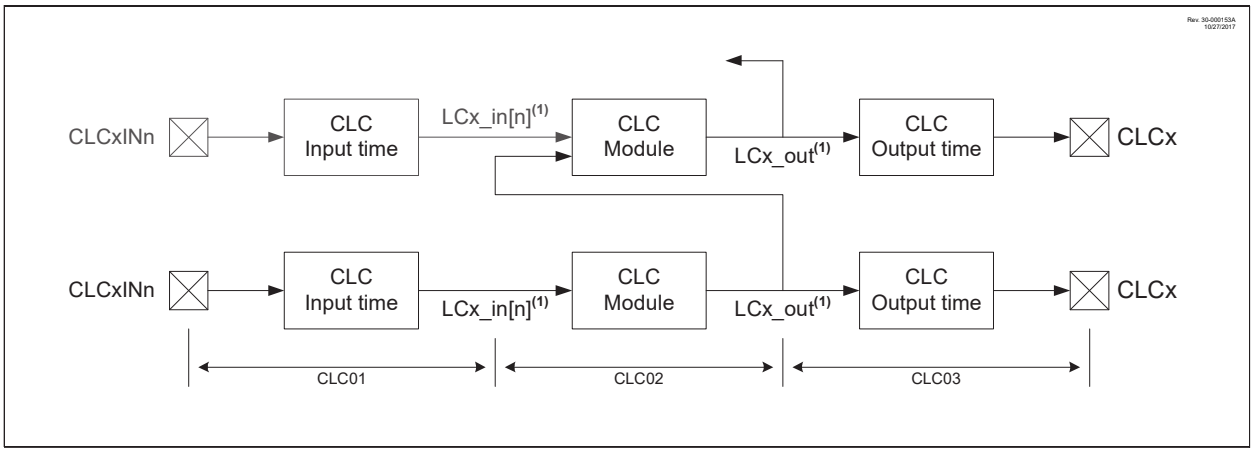
***** 这些参数通过表征确定，但未经测试。

† 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. 关于 OS7、OS8 和 OS9 的上升和下降时间，请参见 *“I/O 和 CLKOUT 时序规范”*。

图 42-21. CLC 传播时序

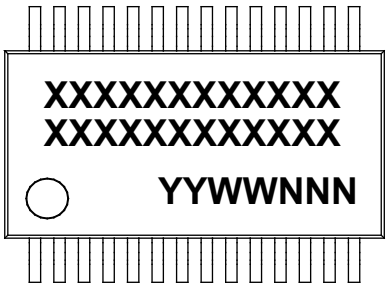


43. 封装信息

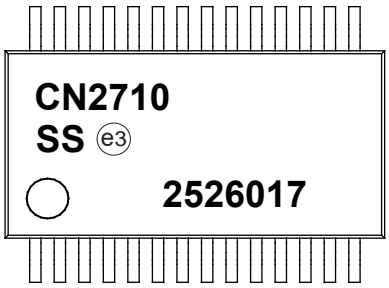
封装标识信息

图注：	XX...X	客户指定信息或器件编号
	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

28引脚SSOP（5.30 mm）



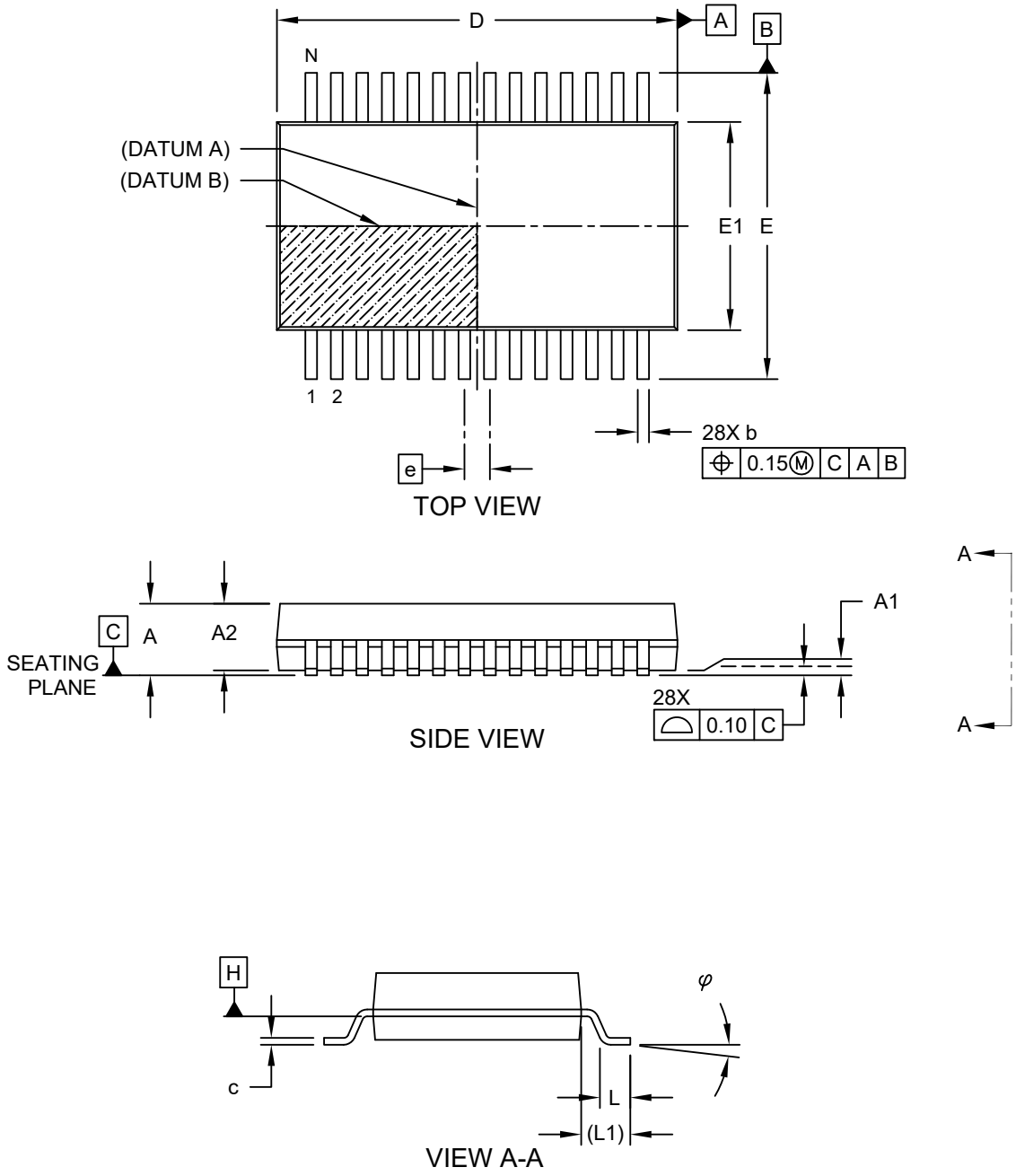
示例



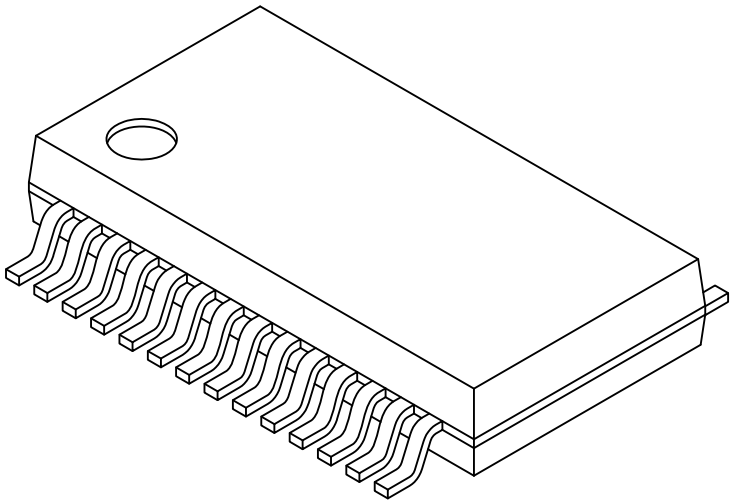
43.1. 封装详细信息

以下部分给出了封装的技术详细信息。

28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]



28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		0.65 BSC	
Overall Height	A	-	-	2.00
Molded Package Thickness	A2	1.65	1.75	1.85
Standoff	A1	0.05	-	-
Overall Width	E	7.40	7.80	8.20
Molded Package Width	E1	5.00	5.30	5.60
Overall Length	D	9.90	10.20	10.50
Foot Length	L	0.55	0.75	0.95
Footprint	L1		1.25 REF	
Lead Thickness	c	0.09	-	0.25
Foot Angle	φ	0°	4°	8°
Lead Width	b	0.22	-	0.38

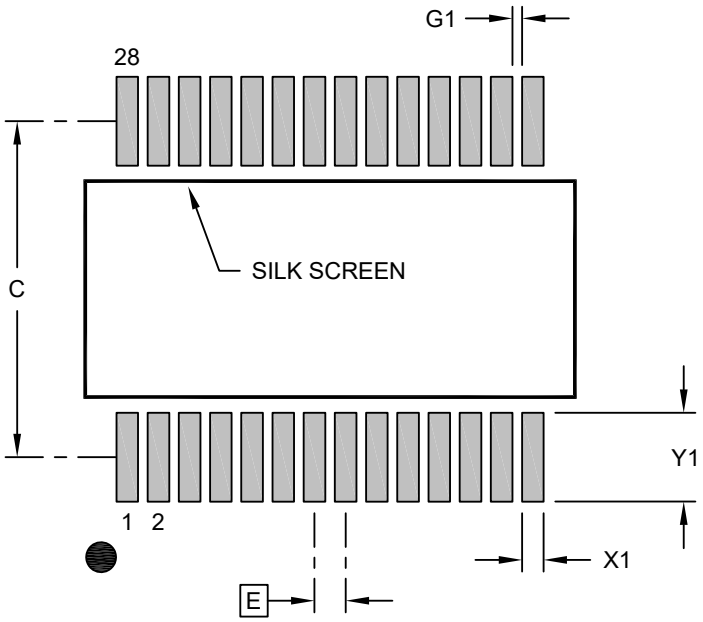
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]



RECOMMENDED LAND PATTERN

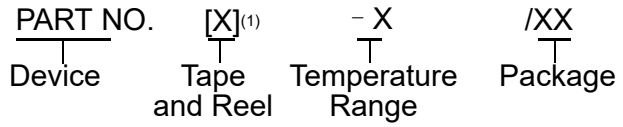
Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E	0.65 BSC		
Contact Pad Spacing	C		7.00	
Contact Pad Width (X28)	X1			0.45
Contact Pad Length (X28)	Y1			1.85
Contact Pad to Center Pad (X26)	G1	0.20		

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

产品标识体系

欲订货或获取信息，请访问 www.weixinsemi.com。



器件：	CN2510、CN2610 和 CN2710	
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)：	SS	= 28 引脚 SSOP
封装类型：	空白	标准

示例：

- CN2510T-I/SS：卷带式，工业级温度，28 引脚 SSOP
- CN2710-I/ST：工业级温度，28 引脚 SSOP

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