CMS80F732x用户手册

增强型闪存8位CMOS单片机

Rev. 0.1.5

目录

1. 产品概述

1.1 系统配置寄存器

系统配置寄存器（CONFIG）是 MCU 初始条件的 FLASH 选项。它只能被 CMS 烧写器烧写，用户不能访问及操作。它包含了以下内容：

1. WDT（看门狗选择）
   - ENABLE 打开看门狗定时器
   - DISABLE 关闭看门狗定时器
2. PROTECT（加密）
   - DISABLE FLASH 代码不加密
   - ENABLE FLASH 代码加密，加密后烧写仿真器读出来的值将不确定
3. LVR_SEL（低压侦测电压选择）
   - 2.5V
   - 3.0V
   - 3.5V
4. DEBUG_EN（调试口功能选择）
   - ENABLE DSCK、DSDA 口一直保持为调试口，所有功能均不能使用
   - DISABLE DSCK、DSDA 口为普通功能口
5. USART1_SEL（TX1/RX1）（USART1 端口选择）
   - RC7/RC6 选择 RC7 为 TX1 口，RC6 为 RX1 口
   - RB4/RB3 选择 RB4 为 TX1 口，RB3 为 RX1 口
6. IIC_SEL（IIC 端口选择）
   - RD1/RD0 选择 RD1 为 SCL 口，RD0 为 SDA 口
   - RC6/RC7 选择 RC6 为 SCL 口，RC7 为 SDA 口
   - RA5/RA6 选择 RA5 为 SCL 口，RA6 为 SDA 口
7. WRPR[15:0]（程序存储器写保护位，0x0000H~0x3FFFH，每一位保护 1K 空间）
   - DISABLE 程序区间不受写保护
   - ENABLE 程序区间受写保护，自写 ROM 时无法写入该区间
8. WAIT_NUM（ROM 读取等待）
   - DISABLE 无等待周期
   - ENABLE 一个等待周期

1.2 在线串行编程

可在最终应用电路中对单片机进行串行编程。编程可以简单地通过以下 4 根线完成：

- 电源线
- 接地线
- 数据线
- 时钟线

这使用户可使用未编程的器件制造电路板，而仅在产品交付前才对单片机进行编程。从而可以将最新版本的固件或者定制固件烧写到单片机中。

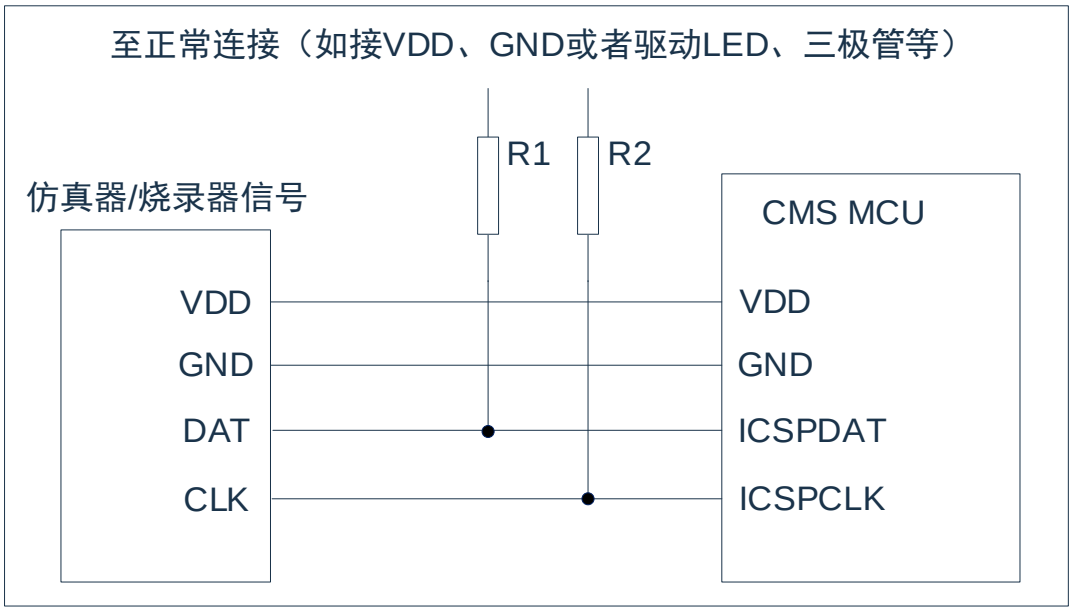


图 1-1：典型的在线串行编程连接方法

上图中，R1、R2 为电气隔离器件，常以电阻代替，其阻值如下：R1≥4.7K、R2≥4.7K。

注意在编程和调试时，ICSPDAT 禁止连接下拉电阻。如果实际电路需要接下拉电阻，建议利用跳线结构，在编程/调试时断开下拉电阻，完成之后再接入下拉电阻。

1.3 集成开发环境

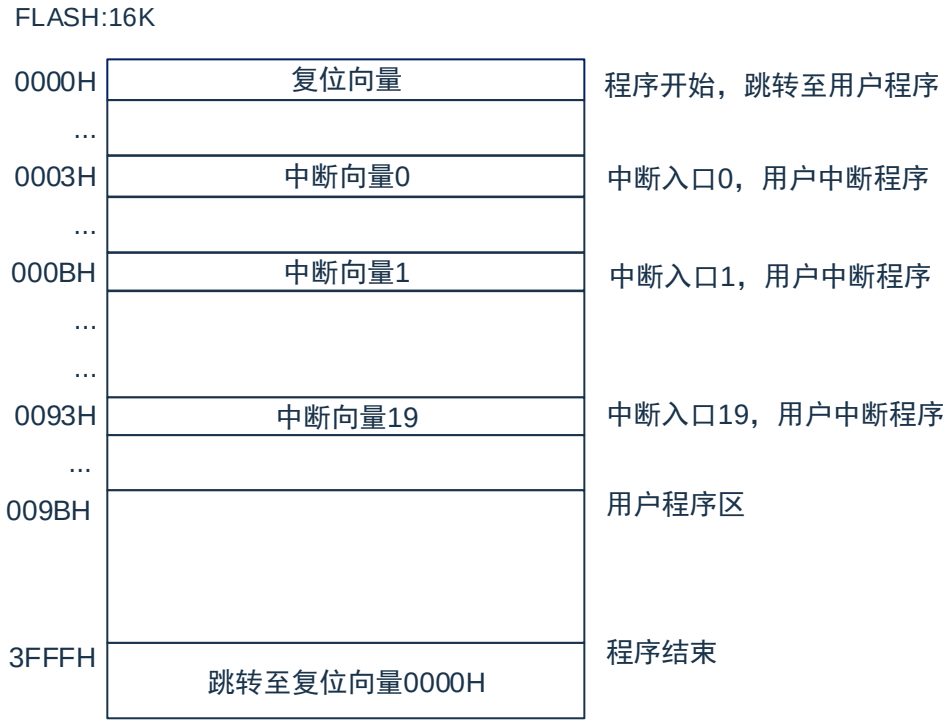
- 芯片支持 2 线调试功能。

2. 中央处理器（CPU）

2.1 内存

2.1.1 程序内存

该芯片具有一个 16KB 的 FLASH 存储空间, FLASH 空间分配结构框图如下：



2.1.1.1 复位向量

单片机具有一个字长的系统复位向量（0000H）。具有以下 3 种复位方式：

- 上电复位
- 看门狗复位
- 低压复位（LVR）

发生上述任一种复位后，程序将从 0000H 处重新开始执行，系统寄存器也都将恢复为默认值。根据 RSTF 寄存器中的 PD 和 TO 标志位的内容可以判断系统复位方式。

2.1.1.2 中断向量

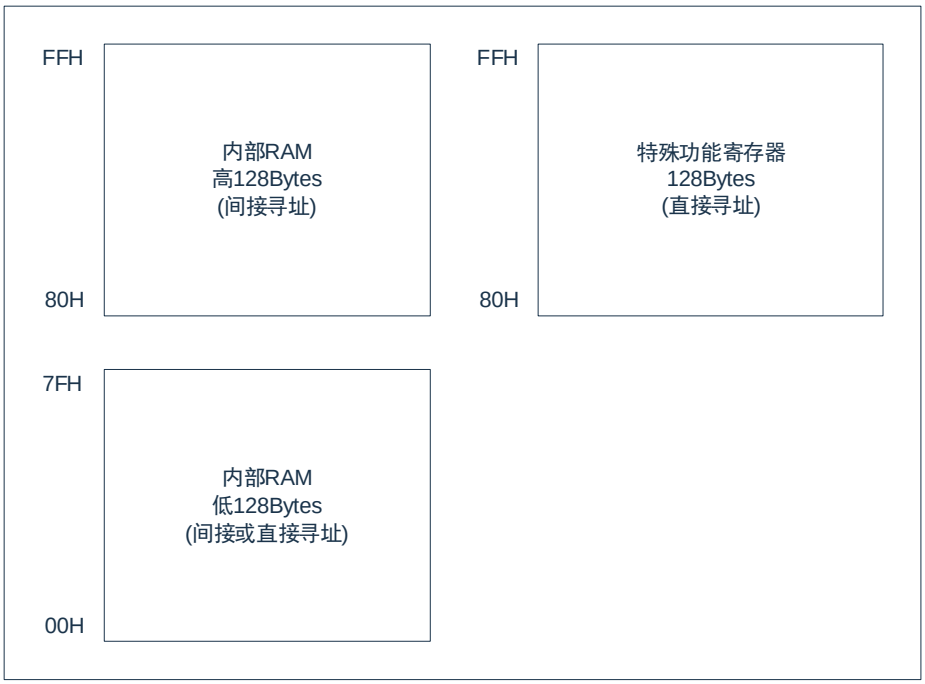
芯片具有 19 个中断源及中断向量：

中断源	中断描述	中断向量	同级优先序列
TMR1IF	Timer1 溢出中断	0-0x0003	1
TMR2IF	Timer2 与 PR2 匹配中断	1-0x000B	2
CPTIF	CPT 中断	2-0x0013	3
SSPIF	MSSP 中断	3-0x001B	4
TX0IF	USART0 发送中断	4-0x0023	5
RX0IF	USART0 接收中断	5-0x002B	6
ADIF	ADC 中断	6-0x0033	7
PWMIF	PWM 中断	7-0x003B	8
---	---	---	---
RX1IF	USART1 接收中断	9-0x004B	10
TX1IF	USART1 发送中断	10-0x0053	11
BCLIF	I^2C 总线冲突中断	11-0x005B	12
EEIF	Data Flash 写操作中断	12-0x0063	13
---	---	---	---
TKIF	触摸检测结束中断	14-0x0073	15
LVDIF	LVD 中断	15-0x007B	16
RBIF	PORTB 电平变化中断	16-0x0083	17
INTF	外部中断	17-0x008B	18
T0IF	Timer0 溢出中断	18-0x0093	19

2.1.2 数据存储器

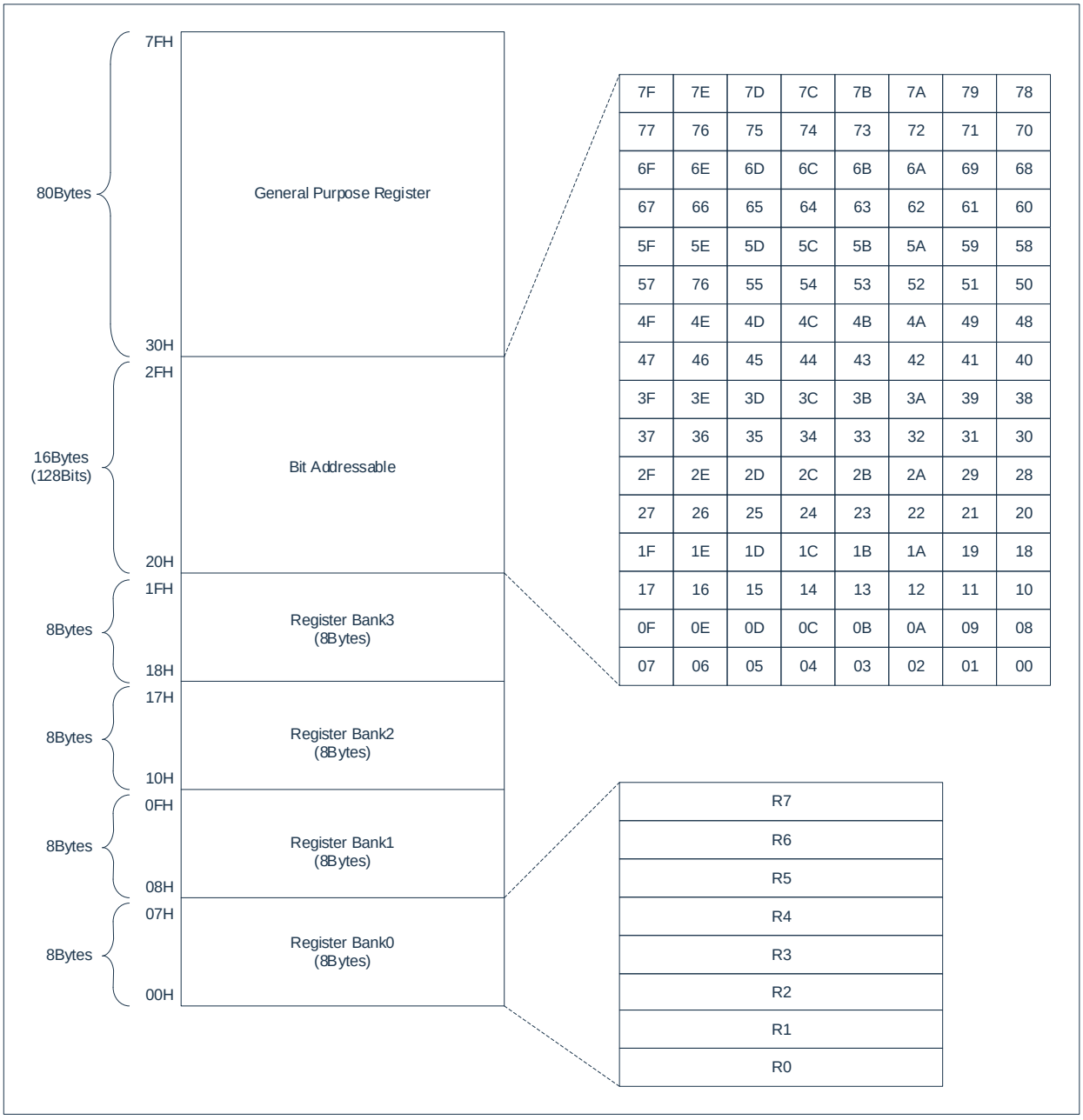
2.1.2.1 通用数据存储器 RAM

内部数据存储器分为 3 个部分：低 128Bytes、高 128Bytes、特殊功能寄存器 SFR。RAM 空间分配结构框图如下图所示：



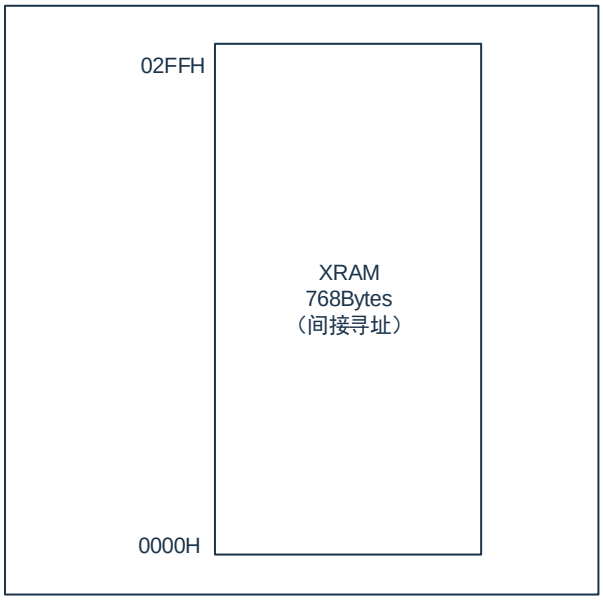
上图所示的高 128Bytes 和 SFR 占用相同的区域（80H~FFH），但它们本身却是独立的。直接寻址高于 7FH 的存储空间（SFR）和间接寻址高于 7FH（高 128Bytes）的存储空间进入到不同的存储空间。

上图所示的低 128Bytes 空间寄存器分配如下图所示。最低的 32 字节（00H~1FH）组成了 4 个寄存器组，每组 8 个存储单元，以 R0~R7 作为单元编号，用于保存操作数及中间结果等。复位后，默认选择 0 组，如果选择其他寄存器组，需通过改变程序状态来决定。寄存器组后边的 16Bytes（20H~2FH）组成了可位寻址的存储空间，该区域的 RAM 单元既可以按字节操作，也可以对单元中的每一位直接位操作。剩余的 80 个存储单元（30H~7FH），用户可设置堆栈区和存储中间数据。


7FH
80Bytes
General Purpose Register
30H
2FH
16Bytes
(128Bits)
Bit Addressable
20H
1FH
8Bytes
Register Bank3
(8Bytes)
18H
17H
8Bytes
Register Bank2
(8Bytes)
10H
0FH
8Bytes
Register Bank1
(8Bytes)
08H
07H
8Bytes
Register Bank0
(8Bytes)
00H
7F 7E 7D 7C 7B 7A 79 78
77 76 75 74 73 72 71 70
6F 6E 6D 6C 6B 6A 69 68
67 66 65 64 63 62 61 60
5F 5E 5D 5C 5B 5A 59 58
57 76 55 54 53 52 51 50
4F 4E 4D 4C 4B 4A 49 48
47 46 45 44 43 42 41 40
3F 3E 3D 3C 3B 3A 39 38
37 36 35 34 33 32 31 30
2F 2E 2D 2C 2B 2A 29 28
27 26 25 24 23 22 21 20
1F 1E 1D 1C 1B 1A 19 18
17 16 15 14 13 12 11 10
0F 0E 0D 0C 0B 0A 09 08
07 06 05 04 03 02 01 00
R7
R6
R5
R4
R3
R2
R1
R0

2.1.2.2 通用外部数据寄存器 XRAM

芯片内部有最大 768Bytes XRAM 区域，该区域与 FLASH/RAM 没有联系，XRAM 空间分配结构框图如下图所示：



XRAM/XSFR 空间访问通过 DPTR 数据指针操作，DPTR 包括两组指针：DPTR0，DPTR1，由 DPS 寄存器选择。例如通过 MOVX 间接寻址操作，汇编代码如下：

```
MOV     R0,#01H
MOV     A,#5AH
MOVX    @R0,A           ;将A中的数据写入XRAM地址01H中，高8位地址由DPH0/1决定
```

在 Keil51 中将 Target-->Memory Model 设置为 Large 后，C 编译器将采用 XRAM 作为变量地址。一般用 DPTR 进行 XRAM/XSFR 的操作。

2.1.2.3 特殊功能寄存器 SFR

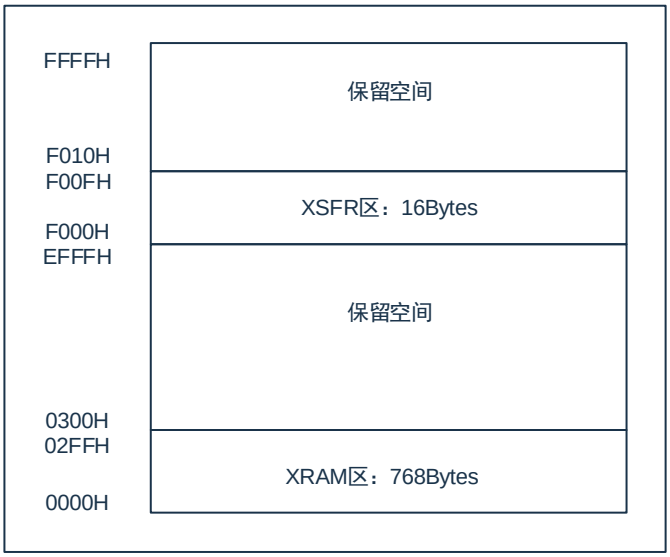
特殊功能寄存器是指有特殊用途的寄存器集合，本质上是一些具有特殊功能的片内 RAM 单元，离散地分布在地址范围 80H~FFH 内。用户可以通过直接寻址指令对它们进行字节存取，地址低四位为 0000 或 1000 的可进行位寻址。

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
0xF8	--	EEDAT	EEDATH	EEADR	EEADRH	EECON1	EECON2	EECON3
0xF0	B	INTCON	PIE1	PIE2	PIE3	PIR1	PIR2	PIR3
0xE8	--	--	--	--	--	--	--	--
0xE0	ACC	--	--	--	--	--	--	--
0xD8	--	--	--	--	ADRESL	ADRESH	ADCON0	ADCON1
0xD0	PSW	--	BRG_ADJ1	RCSTA1	TXREG1	RCREG1	TXSTA1	SPBRG1
0xC8	--	--	BRG_ADJ0	RCSTA0	TXREG0	RCREG0	TXSTA0	SPBRG0
0xC0	--	IOCB	WPDB	SSPBUF	SSPCON	SSPCON2	SSPADD	SSPSTAT
0xB8	--	WPUA	WPUB	WPUC	WPUD	ANSEL0	ANSEL1	ANSEL2
0xB0	PORTD	--	--	--	--	--	ANSEL3	RSTF
0xA8	--	LEDCON0	LEDADD	LEDDATA	SEGEN2	SEGEN1	SEGEN0	COMEN
0xA0	PORTC	IREMAP	CPTCON	CPTRL	CPTRH	TMR2	T2CON	PR2
0x98	--	TRISA	TRISB	TRISC	TRISD	--	--	LVDCON
0x90	PORTB	--	--	--	--	WDTCLR	TA	WDTCON
0x88	--	TMR0	OPTION_REG	TMR1L	TMR1H	T1CON	--	OSCCON
0x80	PORTA	SP	DPL0	DPH0	--	--	DPS	PCON

注：写 TMR0、TMR1L、TMR1H、TMR2、PIR1、PIR2、PIR3、EEADR、EEADRH、SEGEN0、SEGEN1、SEGEN2、COMEN、LEDDATA、LEDADD 等寄存器的指令后面需要加一条 NOP 指令。

2.1.2.4 外部特殊功能寄存器 XSFR

XSFR 是寻址空间与 XRAM 共用的特殊寄存器，主要包括：PWM 功能控制寄存器。其寻址范围如下图所示：



外部特殊功能寄存器列表如下：

地址	寄存器	寄存器描述
F000H	PWMD0L	
F001H	PWMD1L	
F002H	PWMD2L	
F003H	PWMD3L	
F004H	PWMD4L	
F005H	PWMD01H	
F006H	PWMD23H	
F007H	PWMCON0	
F008H	PWMCON1	
F009H	PWMCON2	
F00AH	PWM4TL	
F00BH	PWMTL	
F00CH	PWMTH	
F00DH	PWM01DT	
F00EH	PWM23DT	
F00FH	-	

2.2 累加器（ACC）

ALU 是 8Bit 宽的算术逻辑单元，MCU 所有的数学、逻辑运算均通过它来完成。它可以对数据进行加、减、移位及逻辑运算；ALU 也控制状态位（PSW 状态寄存器中），用来表示运算结果的状态。

ACC 寄存器是一个 8Bit 的寄存器，ALU 的运算结果可以存放在此。

2.3 B 寄存器（B）

B 寄存器在使用乘法和除法指令时使用。如不使用乘除法指令，也可作为通用寄存器使用。

2.4 堆栈指针寄存器（SP）

SP 寄存器指向堆栈的地址，复位后默认值为 0x07，意味着堆栈的区域从 RAM 地址的 08H 开始。该 SP 的值可以修改，如果将堆栈区域设置为 0xC0 开始，则系统复位后需要将 SP 的值设置为 0xBF。

影响 SP 的操作有：指令 PUSH、LCALL、ACALL、POP、RET、RETI 以及进入中断。

PUSH 指令占用堆栈中一个字节，LCALL，ACALL 及中断占用堆栈中两个字节，POP 指令释放一个字节，RET/RETI 指令释放两个字节。

使用 PUSH 指令会将被操作的寄存器的当前值自动保存到 RAM 中。

2.5 数据指针寄存器（DPTR）

数据指针主要用在 MOVX，MOVC 指令中，其作用是定位 XRAM 与 ROM 的地址。芯片内部有 1 个数据指针寄存器 DPTR

每组指针包括两个 8 位寄存器：DPTR0={DPH,DPL}；

例如操作 XRAM 的汇编代码如下：

```
        MOV     DPTR,#0001H
        MOV     A,#5AH
        MOVX    @DPTR,A             ;将A中的数据写入XRAM地址0001H中
```

2.6 数据指针选择寄存器（DPS）

数据指针选择寄存器 DPS

0x86	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
DPS	ID1	ID0	TSL	AU	--	--	--	--
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit6 ID<1:0>: 自减/自加功能选择
00= DPTR加1
01= DPTR减1
10= DPTR加1
11= DPTR减1
Bit5 TSL: 翻转选择使能
1= 执行DPTR指令后，SEL位会自动翻转
0= DPTR相关指令不影响SEL位
Bit4 AU: 自加/减使能位
1= 允许MOVX @DPTR或者MOVC @DPTR指令运行后，执行自减/自加的操作（由ID1-ID0决定）
0= DPTR相关指令不影响DPTR本身
Bit3~Bit0 -- 保留，须均为0

2.7 程序状态寄存器（PSW）

程序状态寄存器 PSW

0xD0	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PSW	CY	AC	F0	RS1	RS0	OV	--	P
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7 CY: 进位标志位
1= 有进位
0= 无进位
Bit6 AC: 辅助进位标志位（半进位标志位）
1= 有进位
0= 无进位
Bit5 F0: 通用标志位
Bit4~Bit3 RS<1:0>: 工作寄存器BANK选择位
00= 选择Bank0
01= 选择Bank1
10= 选择Bank2
11= 选择Bank3
Bit2 OV: 溢出标志位
1= 算术或逻辑运算有溢出
0= 算术或逻辑运算无溢出
Bit1 -- 保留，须为0
Bit0 P: 校验位
1= 结果的最高位发生了进位
0= 结果的最高位没有发生进位

.

2.8 程序计数器（PC）

程序计数器（PC）控制程序内存 FLASH 中的指令执行顺序，它可以寻址整个 FLASH 的范围，取得指令码后，程序计数器（PC）会自动加一，指向下一个指令码的地址。但如果执行跳转、条件跳转、向 PCL 赋值、子程序调用、初始化复位、中断、中断返回、子程序返回等操作时，PC 会加载与指令相关的地址而不是下一条指令的地址。

当遇到条件跳转指令且符合跳转条件时，当前指令执行过程中读取的下一条指令将会被丢弃，且会插入一个空指令操作周期，随后才能取得正确的指令。反之，就会顺序执行下一条指令。

2.9 时序存取寄存器（TA）

时序存取寄存器 TA

0x96	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TA	TA7	TA6	TA5	TA4	TA3	TA2	TA1	TA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0　　TA<7:0>:　时序存取控制位。

某些被保护的寄存器必须在对TA进行如下操作之前才能写入。

```
MOV     TA, #0AAH
MOV     TA, #055H
```

中间不能插入其他任何指令，再次修改时需要重新执行此序列。

被保护的寄存器:　IREMAP。

2.10 复位状态寄存器（RSTF）

RSTF 寄存器可以是任何指令的目标寄存器。这些位指示了器件复位状态。

程序状态寄存器 RSTF

0xB7	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RSTF	TO	PD	--	--	--	--	--	--
R/W	R/W	R/W	--	--	--	--	--	--
复位值	1	1	--	--	--	--	--	--

Bit7 TO: 超时位
1= 上电或执行了清看门狗序列或进入休眠
0= 发生了WDT超时
Bit6 PD: 掉电位
1= 上电或执行了清看门狗序列
0= 进入休眠
Bit5~0 未用

TO 和 PD 标志位可反映出芯片复位的原因，下面列出影响 TO、PD 的事件及各种复位后 TO、PD 的状态。

事件	TO	PD
电源上电	1	1
WDT 溢出	0	X
进入休眠	1	0
清看门狗序列	1	1

影响 TO/PD 的事件表

TO	PD	复位原因
0	0	WDT 溢出唤醒休眠 MCU
0	1	WDT 溢出非休眠模式
1	1	电源上电

复位后 TO/PD 的状态

2.11 预分频器（OPTION_REG）

OPTION_REG 寄存器为可读写的寄存器，各个控制位用于配置：

- TIMER0/WDT 预分频器。
- TIMER0。

预分频器 OPTION_REG

0x8A	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	0	1	1

Bit7 RBPU: PORTB 上拉使能位
0= 由端口的各个锁存值使能 PORTB 上拉
1= 禁止 PORTB 上拉
Bit6 INTEDG: 触发中断的边沿选择位。
0= INT 引脚下降沿触发中断。
1= INT 引脚上升沿触发中断。
Bit5 T0CS: TIMER0 时钟源选择位
0= 内部指令周期时钟（F_{SYS}/4）
1= T0CKI 引脚上的跳变沿
Bit4 T0SE: TIMER0 时钟源边沿选择位
0= 在 T0CKI 引脚信号从低电平跳变到高电平时递增
1= 在 T0CKI 引脚信号从高电平跳变到低电平时递增
Bit3 PSA: 预分频器分配位
0= 预分频器分配给 TIMER0 模块
1= 预分频器分配给 WDT
Bit2~Bit0 PS2~PS0: 预分频参数配置位

PS2	PS1	PS0	TMR0 分频比	WDT 分频比
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

预分频寄存器实际上是一个 8 位的计数器，用于监视寄存器 WDT 时，是作为一个后分频器；用于定时器/计数器时，作为一个预分频器，通常统称作预分频器。在片内只有一个物理的分频器，只能用于 WDT 或 TIMER0，两者不能同时使用。也就是说，若用于 TIMER0，WDT 就不能使用预分频器，反之亦然。

当用于 WDT 时，写 WDTCLR 寄存器将同时对预分频器和 WDT 定时器清零。

当用于 TIMER0 时，有关写入 TIMER0 的所有指令（如：CLR TMR0,MOV TMR0,A 等）都会对预分频器清零。

由 TIMER0 还是 WDT 使用预分频器，完全由软件控制。它可以动态改变。为了避免出现不该有的芯片复位，当从 TIMER0 换为 WDT 使用时，应该按如下顺序执行：

1. 关闭中断，避免在执行特定时序时进入中断程序
2. 将预分频器设置为最大值
3. 清零 TMR0
4. 设置预分频器分配给 WDT
5. 写 WDTCLR 清 WDT
6. 设置新的预分频比
7. 开启中断

要将预分频器从分配给 WDT 改为分配给 TIMER0 模块，应该按如下顺序执行：

1. 写 WDTCLR 清 WDT
2. 设置预分频器分配给 TIMER0，并设定分频比

注：要使 TIMER0 获取 1:1 的预分频比配置，可通过将选项寄存器的 PSA 位置 1 将预分频器分配给 WDT。

2.12 看门狗计数器（WDT）

看门狗定时器（Watch Dog Timer）是一个片内自振式的 RC 振荡定时器，无需任何外围组件，即使芯片的主时钟停止工作，WDT 也能保持计时。WDT 计时溢出将产生复位。

2.12.1 WDT 周期

WDT 与 TIMER0 共用 8 位预分频器。在所有复位后，WDT 默认溢出周期为 128ms，WDT 溢出周期计算方式为 16ms*分频系数，假如你需要改变的 WDT 周期，可以设置 OPTION_REG 寄存器。WDT 的溢出周期将受到环境温度、电源电压等参数影响。

写 WDTCLR 寄存器或者 PCON.STOP 置 1 将清除 WDT 定时器以及预分频器里的计数值（当预分频器分配给 WDT 时）。WDT 一般用来防止系统失控，或者可以说是用来防止单片机程序失控。

在正常情况下，WDT 应该在其溢出前清零，以防止产生复位。如果程序由于某种干扰而失控，那么不能在 WDT 溢出前写 WDTCLR 寄存器，就会使 WDT 溢出而产生复位。使系统重启而不至于失去控制。若是 WDT 溢出产生的复位，则复位状态寄存器（RSTF）的“TO”位会被清零，用户可根据此位来判断复位是否是 WDT 溢出所造成的。

注：

1) 看门狗计数器的溢出时间有一定波动，设置清 WDT 时间，应留有足够的余量。
2) 建议在主程序中定期清除 WDT，避免在多个分支中或者中断程序中清 WDT，最大限度发挥看门狗计数器的保护功能。

2.12.2 看门狗计数器清除寄存器 WDTCLR

WDT 清除寄存器 WDTCLR

0x95	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WDTCLR	CLR_WDT[7:0]							
R/W	R0/W	R0/W	R0/W	R0/W	R0/W	R0/W	R0/W	R0/W
复位值	0	0	0	0	0	0	0	0

Bit7~0 CLR_WDT[7:0]:
依次写入 0x5a,0x69: 清除 WDT 计数器
其他: 无操作

2.12.3 看门狗定时器控制寄存器 WDTCON

看门狗定时器控制寄存器 WDTCON

0x97	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WDTCON	---	---	---	---	---	---	---	SWDTEN
R/W	---	---	---	---	---	---	---	R/W
复位值	---	---	---	---	---	---	---	0

Bit7~Bit1　　未用，读为 0

Bit0　　SWDTEN:　软件使能或禁止看门狗定时器位

1=　使能 WDT

0=　禁止 WDT（复位值）

注：如果 CONFIG 中 WDT 配置位 =1，则 WDT 始终被使能，而与 SWDTEN 控制位的状态无关。如果 CONFIG 中 WDT 配置位=0，则可以使用 SWDTEN 控制位使能或禁止 WDT。

3. 系统时钟

3.1 概述系统振荡器

芯片有 1 种振荡方式：内部 RC 振荡。

3.1.1 内部 RC 振荡

芯片默认的振荡方式为内部 RC 振荡，其振荡频率 32MHz，可通过 OSCCON 寄存器设置芯片工作频率。

3.2 起振时间

起振时间（Reset Time）是指从芯片复位到芯片振荡稳定这段时间，其设计值约为 16ms。

注：无论芯片是电源上电复位，还是其它原因引起的复位，都会存在这个起振时间。

3.3 振荡器控制寄存器

振荡器控制（OSCCON）寄存器控制系统时钟和频率选择。

振荡器控制寄存器 OSCCON

0x8F	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	---	---
R/W	---	R/W	R/W	R/W	---	---	---	---
复位值	---	1	0	1	---	---	---	---

Bit7　　未用，读为 0

Bit6~Bit4　　IRCF<2:0>:　内部振荡器分频选择位。

111= $F_{SYS} = F_{HSI}$ /1

110= $F_{SYS} = F_{HSI}$ /2

101= $F_{SYS} = F_{HSI}$ /4（默认）

100= $F_{SYS} = F_{HSI}$ /8

011= $F_{SYS} = F_{HSI}$ /16

010= $F_{SYS} = F_{HSI}$ /32

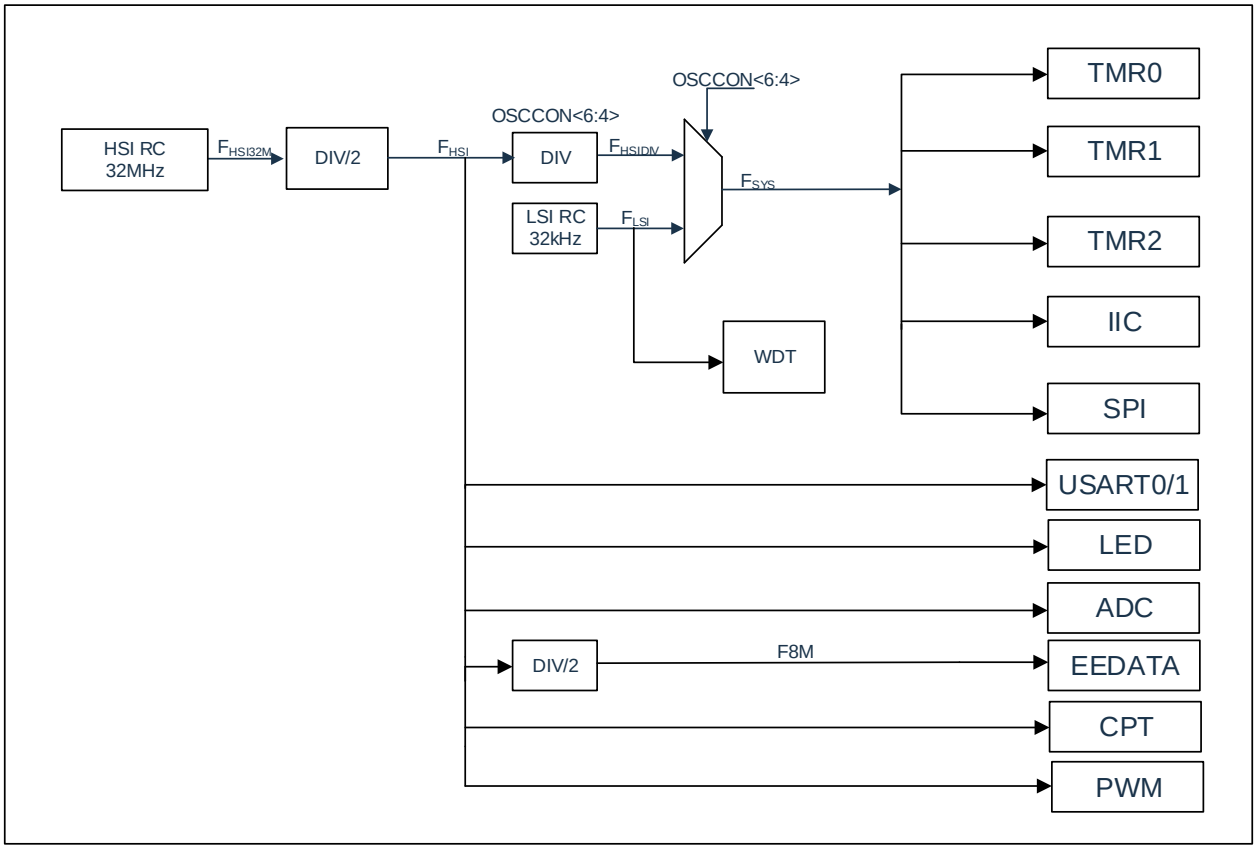
001= $F_{SYS} = F_{HSI}$ /64

000= FSYS = 32kHz（LSI）

Bit3~Bit0　　未用，读为0

注：F_{HSI} 为内部振荡器频率的 2 分频，固定为 16MHz；F_{SYS} 为系统工作频率。

3.4 时钟框图


HSI RC
32MHz
FHSI32M
DIV/2
FHSI
OSCCON<6:4>
DIV
FHSIDIV
OSCCON<6:4>
LSI RC
32kHz
FLSI
FSYS
WDT
TMR0
TMR1
TMR2
IIC
SPI
USART0/1
LED
ADC
DIV/2
F8M
EEDATA
CPT
PWM

4. 复位

芯片可用如下 3 种复位方式：

- 上电复位；
- 低电压复位；
- 正常工作下的看门狗溢出复位；

上述任意一种复位发生时，所有的系统寄存器将恢复默认状态，程序停止运行，同时程序计数器 PC 清零，复位结束后程序从复位向量 0000H 开始运行。STATUS 的 TO 和 PD 标志位能够给出系统复位状态的信息，（详见 STATUS 的说明），用户可根据 PD 和 TO 的状态，控制程序运行路径。

任何一种复位情况都需要一定的响应时间，系统提供完善的复位流程以保证复位动作的顺利进行。

4.1 上电复位

上电复位与 LVR 操作密切相关。系统上电的过程呈逐渐上升的曲线形式，需要一定时间才能达到正常电平值。下面给出上电复位的正常时序：

- 上电：系统检测到电源电压上升并等待其稳定；
- 系统初始化：所有的系统寄存器被置为初始值；
- 振荡器开始工作：振荡器开始提供系统时钟；
- 执行程序：上电结束，程序开始运行。

4.2 掉电复位

4.2.1 概述

掉电复位针对外部因素引起的系统电压跌落情形（例如，干扰或外部负载的变化）。电压跌落可能会进入系统死区，系统死区意味着电源不能满足系统的最小工作电压要求。

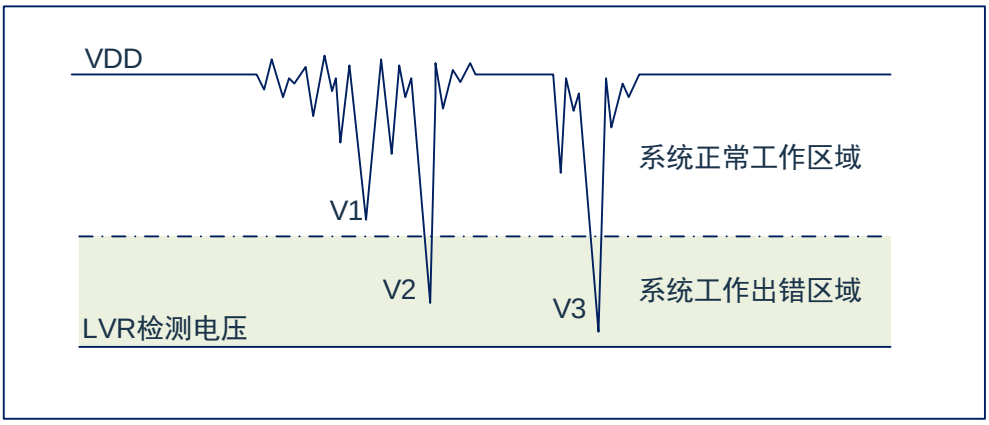


图4-1：掉电复位示意图

上图是一个典型的掉电复位示意图。图中，V_{DD} 受到严重的干扰，电压值降的非常低。虚线以上区域系统正常工作，在虚线以下的区域内，系统进入未知的工作状态，这个区域称作死区。当 V_{DD} 跌至 V1 时，系统仍处于正常状态；当 V_{DD} 跌至 V2 和 V3 时，系统进入死区，则容易导致出错。

以下情况系统可能进入死区：

- DC 运用中：
  - DC 运用中一般都采用电池供电，当电池电压过低或单片机驱动负载时，系统电压可能跌落并进入死区。这时，电源不会进一步下降到 LVD 检测电压，因此系统维持在死区。
- AC 运用中：
  - 系统采用 AC 供电时，DC 电压值受 AC 电源中的噪声影响。当外部负载过高，如驱动马达时，负载动作产生的干扰也影响到 DC 电源。V_{DD} 若由于受到干扰而跌落至最低工作电压以下时，则系统将有可能进入不稳定工作状态。
  - 在 AC 运用中，系统上、下电时间都较长。其中，上电时序保护使得系统正常上电，但下电过程却和 DC 运用中情形类似，AC 电源关断后，V_{DD} 电压在缓慢下降的过程中易进入死区。

如上图所示，系统正常工作电压区域一般高于系统复位电压，同时复位电压由低电压检测（LVR）电平决定。当系统执行速度提高时，系统最低工作电压也相应提高，但由于系统复位电压是固定的，因此在系统最低工作电压与系统复位电压之间就会出现一个电压区域，系统不能正常工作，也不会复位，这个区域即为死区。

4.2.2 掉电复位的改进办法

如何改进系统掉电复位性能，以下给出几点建议：

- 选择较高的 LVR 电压，有助于复位更可靠；
- 开启看门狗定时器；
- 降低系统的工作频率；
- 增大电压下降斜率。

看门狗定时器

看门狗定时器用于保证程序正常运行，当系统进入工作死区或者程序运行出错时，看门狗定时器会溢出，系统复位。

降低系统的工作速度

系统工作频率越快，系统最低工作电压越高。从而增大了工作死区的范围，降低系统工作速度就可以降低最低工作电压，从而有效的减小系统工作在死区电压的机率。

增大电压下降斜率

此方法可用于系统工作在 AC 供电的环境，一般 AC 供电系统，系统电压在掉电过程中下降很缓慢，这就会造成芯片较长时间工作在死区电压，此时若系统重新上电，芯片工作状态可能出错，建议在芯片电源与地线间加一个放电电阻，以便让 MCU 快速通过死区，进入复位区，避免芯片上电出错可能性。

4.3 看门狗复位

看门狗复位是系统的一种保护设置。在正常状态下，由程序将看门狗定时器清零。若出错，系统处于未知状态，看门狗定时器溢出，此时系统复位。看门狗复位后，系统重启进入正常状态。

看门狗复位的时序如下：

- 看门狗定时器状态：系统检测看门狗定时器是否溢出，若溢出，则系统复位；
- 初始化：所有的系统寄存器被置为默认状态；
- 振荡器开始工作：振荡器开始提供系统时钟；
- 程序：复位结束，程序开始运行。

关于看门狗定时器的应用问题请参看 2.12WDT 应用章节。

5. 电源管理

低功耗模式分为 2 类：

- IDLE：空闲模式
- STOP：休眠模式

用户利用 C 语言进行程序开发时，强烈建议使用 IDLE 和 STOP 宏指令来控制系统模式，不要直接设置 IDLE 和 STOP 位。宏指令如下：

进入空闲模式：IDLE();

进入休眠模式：STOP();

5.1 电源管理寄存器 PCON

电源管理寄存器寄存器 PCON

0x87	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PCON	--	--	--	--	--	--	STOP	IDLE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit2	未用	
Bit1	STOP:	休眠状态控制位
	0=	未进入休眠状态
	1=	进入休眠状态（退出 STOP 模式自动清零）
Bit0	IDLE:	空闲状态控制位
	0=	未进入空闲状态
	1=	进入空闲状态（退出 IDLE 模式自动清零）

5.2 IDLE 空闲模式

在此模式下，只有 CPU 时钟源被关闭。因此，在这种状态下， 时钟发生器 HSI 和部分外设功能（定时器，PWM）仍然正常工作。

系统进入空闲模式后，可由任意中断唤醒，唤醒后进入中断处理程序，中断返回后，继续执行休眠操作后指令。如果在中断服务程序中进入空闲模式，则只能由优先级较高的中断唤醒系统。

注：系统从空闲模式唤醒时 CPU 时钟源和外设时钟源会暂停，暂停时间与 5.3.2 章节里的休眠唤醒等待时间相同。

5.3 STOP 休眠模式

在此模式下，系统处于低功耗模式，数字电路均不工作。

在休眠模式下，为了尽量降低电流消耗，所有 I/O 引脚都应该保持为 VDD 或 GND，没有外部电路从 I/O 引脚消耗电流。为了避免输入引脚悬空而引入开关电流，应在外部将高阻输入的 I/O 引脚拉为高电平或低电平。为了将电流消耗降至最低，还应考虑芯片内部上拉电阻的影响。

5.3.1 从休眠状态唤醒

可以通过下列任一事件将器件从休眠状态唤醒：

1. 看门狗定时器唤醒（WDT 强制使能）；
2. PORTB 电平变化中断；
3. 外部中断。

如果通过看门狗定时器唤醒器件，RSTF 寄存器中的 TO 和 PD 位用于确定器件复位的原因。PD 位在上电时被置 1，而在进入休眠时被清零。TO 位在发生 WDT 唤醒时被清零。

如果通过中断事件唤醒器件，则必须将相应的中断允许位置 1（允许）。唤醒与 GIE 位的状态无关。器件从休眠状态唤醒时，WDT 都将被清零，而与唤醒的原因无关。

5.3.2 休眠模式唤醒时间

在中断产生或定时时间到后，MCU 从休眠态被唤醒需要等待一段时间才能唤醒系统，执行程序的下一条指令。MCU 被唤醒时，系统振荡器启动，但振荡频率还未稳定，CPU 未工作，PC 仍停止在休眠状态，系统需要等待一段时间才将时钟提供给 CPU。唤醒等待时间过后，MCU 认为系统时钟已经稳定，才将时钟提供给 CPU，程序继续执行。具体关系如下表所示。

系统主频时钟源	系统时钟分频选择(IRCF<2:0>)	休眠唤醒等待时间 T_{WAIT}
内部高速 RC 振荡（F_{HSI}）	F_{SYS}= F_{HSI}	T_{WAIT}=511*1/ F_{HSI}+3*1/ F_{SYS}
内部低速 RC 振荡（F_{LSI}）	----	T_{WAIT}=3*1/F_{LSI}

6. I/O 端口

芯片有四个 I/O 端口：PORTA、PORTB、PORTC、PORTD（最多 26 个 I/O）。可读写端口数据寄存器可直接存取这些端口。

端口	位	管脚描述	I/O
PORTA	0	施密特触发输入，推挽式输出，AN0，TK0，LED 驱动 SEG 口	I/O
	1	施密特触发输入，推挽式输出，AN1，TK1，LED 驱动 SEG 口	I/O
	2	施密特触发输入，推挽式输出，AN2，TK2，LED 驱动 SEG 口	I/O
	3	施密特触发输入，推挽式输出，AN3，TK3，LED 驱动 SEG 口	I/O
	4	施密特触发输入，推挽式输出，AN4，TK4，LED 驱动 SEG 口，T0CKI	I/O
	5	施密特触发输入，推挽式输出，AN5，TK5，LED 驱动 SEG 口	I/O
	6	施密特触发输入，推挽式输出，AN6，TK6，LED 驱动 SEG 口	I/O
	7	施密特触发输入，推挽式输出，AN7，TK7，LED 驱动 SEG 口	I/O
PORTB	0	施密特触发输入，推挽式输出，AN8，TK8，LED 驱动 SEG 口	I/O
	1	施密特触发输入，推挽式输出，AN9，TK9，LED 驱动 SEG 口，外部中断输入	I/O
	2	施密特触发输入，推挽式输出，AN10，TK10，LED 驱动 SEG 口	I/O
	3	施密特触发输入，推挽式输出，AN11，TK11，CPT，RX1/DT1	I/O
	4	施密特触发输入，推挽式输出，AN12，TK12，CPT，TX1/CK1，T1G	I/O
	5	施密特触发输入，推挽式输出，AN13，TK13，T1CKI，OSCO	I/O
	6	施密特触发输入，推挽式输出，AN14，TK14，OSCI	I/O
PORTC	0	施密特触发输入，推挽式输出，AN23，TK23，LED 驱动 COM 口	I/O
	1	施密特触发输入，推挽式输出，AN22，TK22，LED 驱动 COM 口	I/O
	2	施密特触发输入，推挽式输出，AN21，TK21，LED 驱动 COM 口	I/O
	3	施密特触发输入，推挽式输出，AN20，TK20，LED 驱动 COM 口	I/O
	4	施密特触发输入，推挽式输出，AN19，TK19，LED 驱动 COM 口	I/O
	5	施密特触发输入，推挽式输出，AN18，TK18，LED 驱动 COM 口	I/O
	6	施密特触发输入，推挽式输出，AN17，TK17，LED 驱动 COM 口，CPT，RX1/DT1，SCL/SCK	I/O
	7	施密特触发输入，推挽式输出，AN16，TK16，LED 驱动 COM 口，CPT，TX1/CK1，SDA/SDI	I/O
PORTD	0	施密特触发输入，推挽式输出，AN25，TK25，编程时钟输入，RX0/DT0，SDA	I/O
	1	施密特触发输入，推挽式输出，AN24，TK24，编程数据输入/输出，TX0/CK0，SCL	I/O
	2	施密特触发输入，推挽式输出，AN15，TK15	I/O

＜表 6-1：端口配置总概＞

6.1 I/O 口结构图

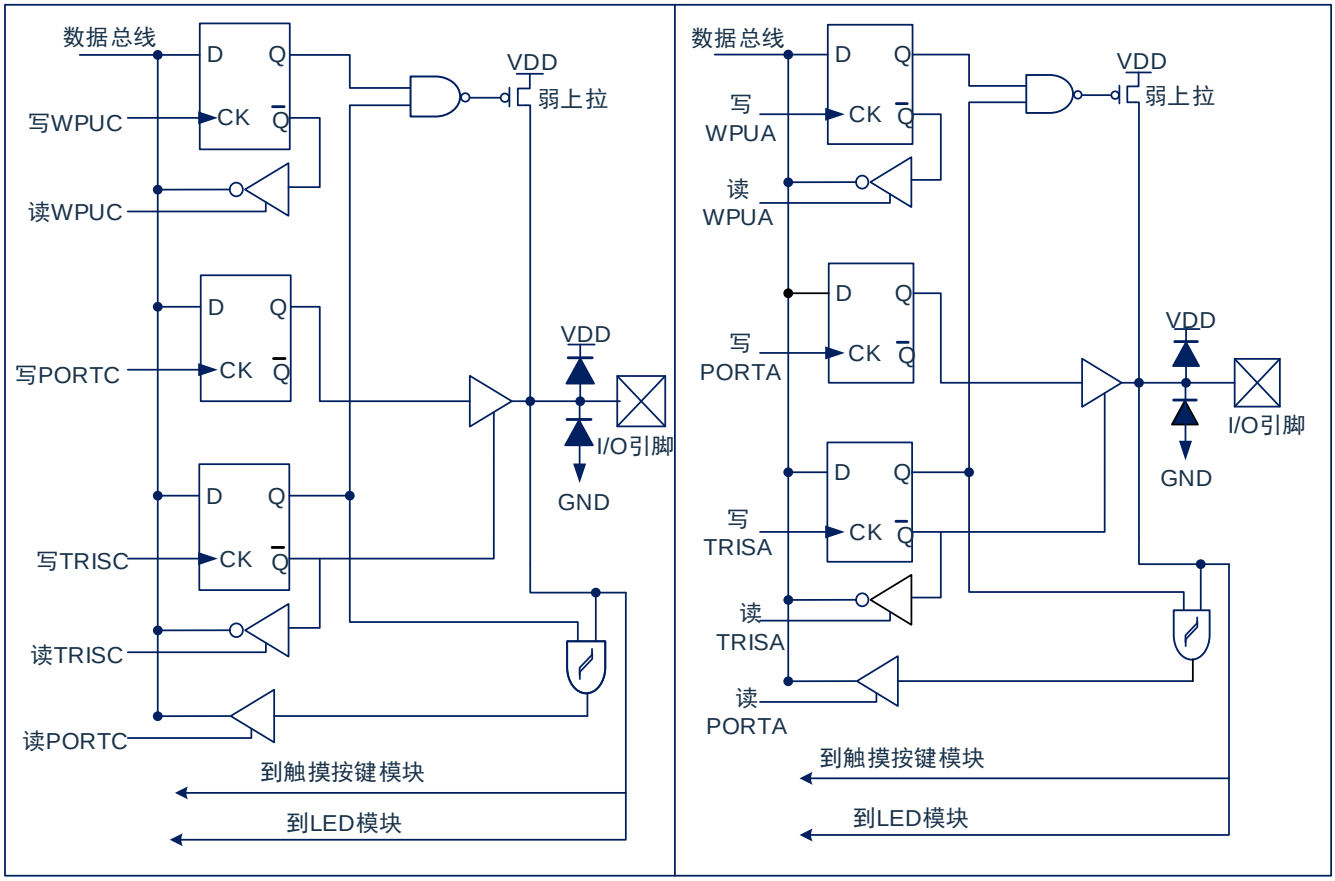


图 6-1：I/O 口结构图（1）

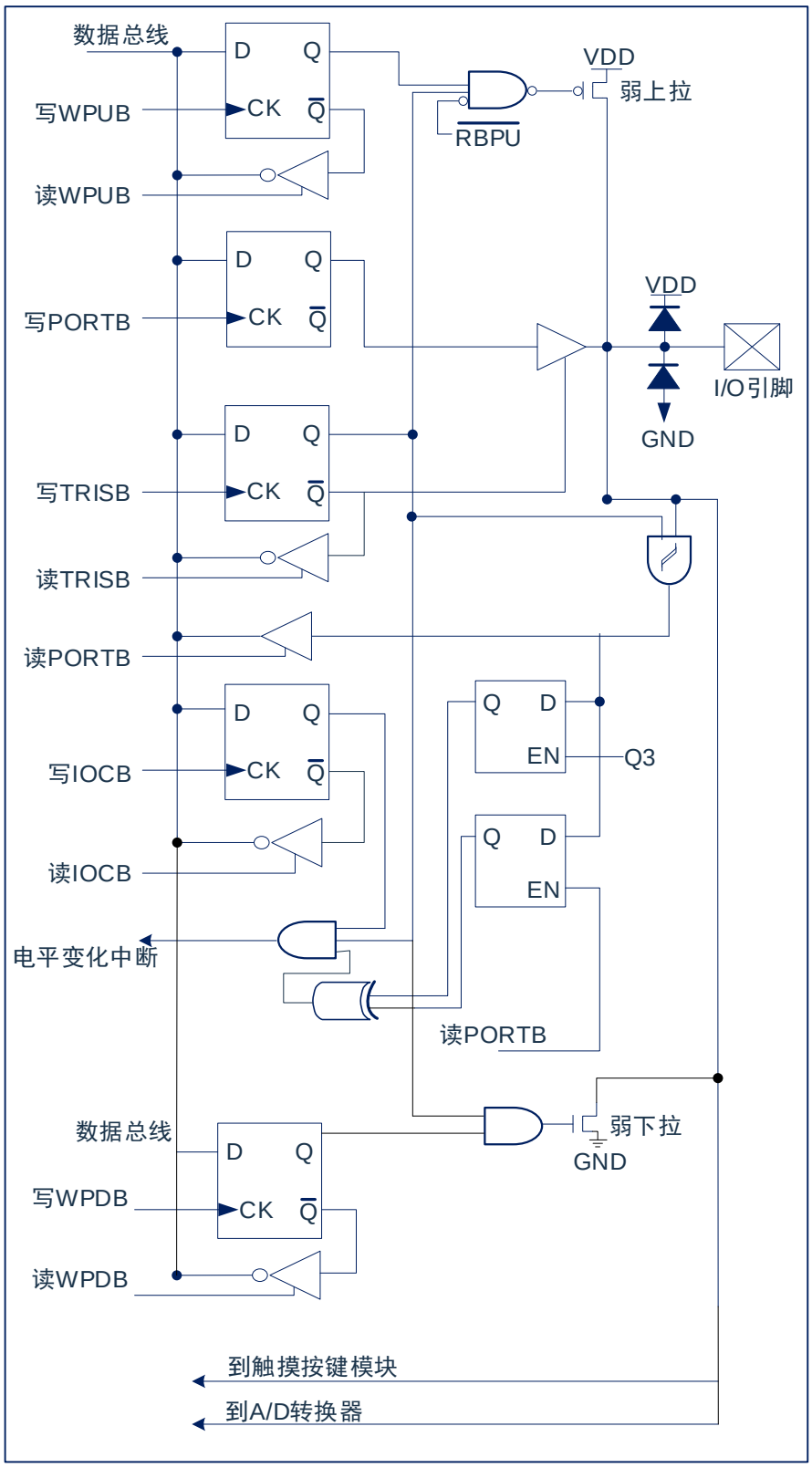

数据总线
D
Q
CK
Q
写WPUB
读WPUB
RBPU
VDD
弱上拉
写PORTB
VDD
I/O引脚
GND
写TRISB
读TRISB
读PORTB
写IOCB
读IOCB
Q
D
EN
Q3
电平变化中断
读PORTB
数据总线
弱下拉
GND
写WPDB
读WPDB
到触摸按键模块
到A/D转换器

图 6-2：I/O 口结构图（2）

6.2 PORTA

6.2.1 PORTA 数据及方向控制

PORTA 是 8Bit 宽的双向端口。它所对应的数据方向寄存器是 TRISA。将 TRISA 的一个位置 1（=1）可以将相应的引脚配置为输入。清零 TRISA 的一个位（=0）可将相应的 PORTA 引脚配置为输出。

读 PORTA 寄存器读的是引脚的状态而写该寄存器将会写入端口锁存器。所有写操作都是读－修改－写操作。因此，写一个端口就意味着先读该端口的引脚电平，修改读到的值，然后再将改好的值写入端口数据锁存器。即使在 PORTA 引脚用作模拟输入时，TRISA 寄存器仍然控制 PORTA 引脚的方向。当将 PORTA 引脚用作模拟输入时，用户必须确保 TRISA 寄存器中的位保持为置 1 状态。

与 PORTA 口相关寄存器有 PORTA、TRISA、WPUA、ANSEL0 等。

PORTA 数据寄存器 PORTA

80H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0　　PORTA<7:0>:　PORTAI/O 引脚位

当TRISAx=1时

1=　端口引脚电平>V_{IH}

0=　端口引脚电平<V_{IL}

当TRISAx=0时

1=　端口输出高电平

0=　端口输出低电平

PORTA 方向寄存器 TRISA

99H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISA	TRISA6	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

Bit7~Bit0　　TRISA<7:0>:　PORTA 三态控制位

1=　PORTA 引脚被配置为输入（三态）

0=　PORTA 引脚被配置为输出

6.2.2 PORTA 上拉电阻

每个 PORTA 引脚都有可单独配置的内部弱上拉。控制位 WPUA<7:0>使能或禁止每个弱上拉。当将端口引脚配置为输出时，其弱上拉会自动切断。

PORTA 上拉电阻寄存器 WPUA

B9H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 WPUA<7:0>: 弱上拉寄存器位
1= 使能上拉
0= 禁止上拉

注：如果引脚被配置为输出或模拟输入，将自动禁止弱上拉。

6.2.3 PORTA 模拟选择控制

ANSEL0 寄存器用于将 I/O 引脚的输入模式配置为模拟模式。将 ANSEL0 中适当的位置 1 将导致对相应引脚的所有数字读操作返回 0，并使引脚的模拟功能正常工作。ANSEL0 位的状态对数字输出功能没有影响。TRIS 清零且 ANSEL0 置 1 的引脚仍作为数字输出，但输入模式将成为模拟模式。这会导致在受影响的端口上执行读－修改－写操作时产生不可预计的结果。

PORTA 模拟选择寄存器 ANSEL0

BDH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL0	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 ANS<7:0>: 模拟选择位,分别选择引脚 AN<7:0>的模拟或数字功能
1= 模拟输入。引脚被分配为模拟输入
0= 数字 I/O。引脚被分配给端口或特殊功能

6.3 PORTB

6.3.1 PORTB 数据及方向

PORTB 是一个 7Bit 宽的双向端口。对应的数据方向寄存器为 TRISB。将 TRISB 中的某个位置 1（=1）可以使对应的 PORTB 引脚作为输入引脚。将 TRISB 中的某个位清零（=0）将使对应的 PORTB 引脚作为输出引脚。

读 PORTB 寄存器读的是引脚的状态而写该寄存器将会写入端口锁存器。所有写操作都是读－修改－写操作。因此，写一个端口就意味着先读该端口的引脚电平，修改读到的值，然后再将改好的值写入端口数据锁存器。即使在 PORTB 引脚用作模拟输入时，TRISB 寄存器仍然控制 PORTB 引脚的方向。当将 PORTB 引脚用作模拟输入时，用户必须确保 TRISB 寄存器中的位保持为置 1 状态。

与 PORTB 口相关寄存器有 PORTB、TRISB、WPUB、IOCB、WPDB、ANSEL1 等。

PORTB 数据寄存器 PORTB

90H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTB	----	RB6	RB5	RB4	RB3	RB2	RB1	RB0
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	----	X	X	X	X	X	X	X

Bit7 未用

Bit6~Bit0 PORTB<6:0>: PORTBI/O 引脚位

当TRISBx=1时

1= 端口引脚电平>V_{IH}

0= 端口引脚电平<V_{IL}

当TRISBx=0时

1= 端口输出高电平

0= 端口输出低电平

PORTB 方向寄存器 TRISB

9AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISB	----	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	----	1	1	1	1	1	1	1

Bit7 未用

Bit6~Bit0 TRISB<6:0>: PORTB 三态控制位

1= PORTB引脚被配置为输入（三态）

0= PORTB引脚被配置为输出

6.3.2 PORTB 上拉电阻

每个 PORTB 引脚都有可单独配置的内部弱上拉。控制位 WPUB<6:0>使能或禁止每个弱上拉。当将端口引脚配置为输出时，其弱上拉会自动切断。在上电复位时，弱上拉由 OPTION_REG 寄存器的 RBPU 位禁止。

PORTB 上拉电阻寄存器 WPUB

BAH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUB	----	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	----	0	0	0	0	0	0	0

Bit7 未用

Bit6~Bit0 WPUB<6:0>: 弱上拉寄存器位

1= 使能上拉

0= 禁止上拉

注：
1) 要单独使能任一个上拉，OPTION_REG 寄存器的全局 RBPU 位必须清零。
2) 如果引脚被配置为输出或者模拟输入，将自动禁止弱上拉。

6.3.3 PORTB 模拟选择控制

ANSEL1 寄存器用于将 I/O 引脚的输入模式配置为模拟模式。将 ANSEL1 中适当的位置 1 将导致对相应引脚的所有数字读操作返回 0，并使引脚的模拟功能正常工作。ANSEL1 位的状态对数字输出功能没有影响。TRIS 清零且 ANSEL1 置 1 的引脚仍作为数字输出，但输入模式将成为模拟模式。这会导致在受影响的端口上执行读－修改－写操作时产生不可预计的结果。

PORTB 模拟选择寄存器 ANSEL1

BEH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL1	ANS15	ANS14	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 ANS<15:8>: 模拟选择位,分别选择引脚 AN<15:8>的模拟或数字功能
1= 模拟输入。引脚被分配为模拟输入
0= 数字 I/O。引脚被分配给端口或特殊功能

6.3.4 PORTB 电平变化中断

所有的 PORTB 引脚都可以被单独配置为电平变化中断引脚。控制位 IOCB<6:0>允许或禁止每个引脚的该中断功能。上电复位时禁止引脚的电平变化中断功能。

对于已允许电平变化中断的引脚，则将该引脚上的值与上次读 PORTB 时锁存的旧值进行比较。将与上次读操作“不匹配”的输出一起进行逻辑或运算，以将 INTCON 寄存器中的 PORTB 电平变化中断标志位（RBIF）置 1。

该中断可将器件从休眠中唤醒，用户可在中断服务程序中通过以下步骤清除中断：

- 对 PORTB 进行读操作。这将结束引脚电平的不匹配状态。
- 将标志位 RBIF 清零。

不匹配状态会不断将 RBIF 标志位置 1。而读 PORTB 将结束不匹配状态，并且允许将 RBIF 标志位清零。锁存器将保持最后一次读取的值不受欠压复位的影响。在复位之后，如果不匹配仍然存在，RBIF 标志位将继续置 1。

> 注：由于对端口的读影响到该端口的所有位，所以在电平变化中断模式下使用多个引脚的时候必须特别小心，在处理一个引脚电平变化的时候可能不会注意到另一个引脚上的电平变化。

PORTB 电平变化中断寄存器 IOCB

C1H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCB	----	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	----	0	0	0	0	0	0	0

Bit7 未用
Bit6~Bit0 IOCB<6:0> PORTB 的电平变化中断控制位
1= 允许电平变化中断
0= 禁止电平变化中断

6.3.5 PORTB 下拉电阻

每个 PORTB 引脚都有可单独配置的内部弱下拉。控制位 WPDB<6:0>使能或禁止每个弱下拉。当将端口引脚配置为输出时，其弱下拉会自动切断。

PORTB 下拉电阻寄存器 WPDB

C2H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDB	----	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	----	0	0	0	0	0	0	0

Bit7 未用

Bit6~Bit0 WPDB<6:0>: 弱下拉寄存器位

1= 使能下拉

0= 禁止下拉

注：如果引脚被配置为输出或者模拟输入，将自动禁止弱下拉。

6.4 PORTC

6.4.1 PORTC 数据及方向

PORTC 是一个 8Bit 宽的双向端口。对应的数据方向寄存器为 TRISC。将 TRISC 中的某个位置 1（=1）可以使对应的 PORTC 引脚作为输入引脚。将 TRISC 中的某个位清零（=0）将使对应的 PORTC 引脚作为输出引脚。

读 PORTC 寄存器读的是引脚的状态而写该寄存器将会写入端口锁存器。所有写操作都是读－修改－写操作。因此，写一个端口就意味着先读该端口的引脚电平，修改读到的值，然后再将改好的值写入端口数据锁存器。

PORTC 数据寄存器 PORTC

A0H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0 PORTC<7:0> PORTC I/O 引脚位

当TRISCx=1时

1= 端口引脚电平>V_{IH}

0= 端口引脚电平<V_{IL}

当TRISCx=0时

1= 端口输出高电平

0= 端口输出低电平

PORTC 方向寄存器 TRISC

9BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

Bit7~Bit0 TRISC<7:0>: PORTC 三态控制位

1= PORTC 引脚被配置为输入（三态）

0= PORTC 引脚被配置为输出

6.4.2 PORTC 上拉电阻

每个 PORTC 引脚都有可单独配置的内部弱上拉。控制位 WPUC<7:0>使能或禁止每个弱上拉。当将端口引脚配置为输出时，其弱上拉会自动切断。

PORTC 上拉电阻寄存器 WPUC

BBH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUC	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0　　WPUC<7:0>:　弱上拉寄存器位。
1=　使能上拉。
0=　禁止上拉。

注：如果引脚被配置为输出或者模拟输入，将自动禁止弱上拉。

6.4.3 PORTC 模拟选择控制

ANSEL2 寄存器用于将 I/O 引脚的输入模式配置为模拟模式。将 ANSEL2 中适当的位置 1 将导致对相应引脚的所有数字读操作返回 0，并使引脚的模拟功能正常工作。ANSEL2 位的状态对数字输出功能没有影响。TRIS 清零且 ANSEL2 置 1 的引脚仍作为数字输出，但输入模式将成为模拟模式。这会导致在受影响的端口上执行读－修改－写操作时产生不可预计的结果。

PORTB 模拟选择寄存器 ANSEL2

BFH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL2	ANS23	ANS22	ANS21	ANS20	ANS19	ANS18	ANS17	ANS16
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0　　ANS<23:16>:　模拟选择位，分别选择引脚 AN<23:16>的模拟或数字功能
1=　模拟输入。引脚被分配为模拟输入
0=　数字 I/O。引脚被分配给端口或特殊功能

6.5 PORTD

6.5.1 PORTD 数据及方向

PORTD 是一个 3 位宽的双向端口。对应的数据方向寄存器为 TRISD。将 TRISD 中的某个位置 1（=1）可以使对应的 PORTD 引脚作为输入引脚。将 TRISD 中的某个位清零（=0）将使对应的 PORTD 引脚作为输出引脚。

读 PORTD 寄存器读的是引脚的状态而写该寄存器将会写入端口锁存器。所有写操作都是读－修改－写操作。因此，写一个端口就意味着先读该端口的引脚电平，修改读到的值，然后再将改好的值写入端口数据锁存器。

PORTD 数据寄存器 PORTD

B0H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTD	----	----	----	----	----	RD2	RD1	RD0
R/W	----	----	----	----	----	R/W	R/W	R/W
复位值	----	----	----	----	----	X	X	X

Bit7~Bit3 未用

Bit2~Bit0 PORTD<2:0>: PORTDI/O 引脚位

当TRISDx=1时

1= 端口引脚电平>V_{IH}

0= 端口引脚电平<V_{IL}

当TRISDx=0时

1= 端口输出高电平

0= 端口输出低电平

PORTD 方向寄存器 TRISD

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISD	----	----	----	----	----	TRISD2	TRISD1	TRISD0
R/W	----	----	----	----	----	R/W	R/W	R/W
复位值	----	----	----	----	----	1	1	1

Bit7~Bit3 未用

Bit2~Bit0 TRISD<2:0>: PORTD 三态控制位

1= PORTD 引脚被配置为输入（三态）

0= PORTD 引脚被配置为输出

6.5.2 PORTD 上拉电阻

每个 PORTD 引脚都有可单独配置的内部弱上拉。控制位 WPUD<2:0>使能或禁止每个弱上拉。当将端口引脚配置为输出时，其弱上拉会自动切断。

PORTD 上拉电阻寄存器 WPUD

BCH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUD	----	----	----	----	----	WPUD2	WPUD1	WPUD0
R/W	----	----	----	----	----	R/W	R/W	R/W
复位值	----	----	----	----	----	0	0	0

Bit7~Bit3　　未用

Bit2~Bit0　　WPUD<2:0>:　弱上拉寄存器位

1=　使能上拉

0=　禁止上拉

注：如果引脚被配置为输出或模拟输入，将自动禁止弱上拉。

6.5.3 PORTD 模拟选择控制

ANSEL3 寄存器用于将 I/O 引脚的输入模式配置为模拟模式。将 ANSEL3 中适当的位置 1 将导致对相应引脚的所有数字读操作返回 0，并使引脚的模拟功能正常工作。ANSEL3 位的状态对数字输出功能没有影响。TRIS 清零且 ANSEL3 置 1 的引脚仍作为数字输出，但输入模式将成为模拟模式。这会导致在受影响的端口上执行读－修改－写操作时产生不可预计的结果。

PORTD 模拟选择寄存器 ANSEL3

B6H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL3	---	---	---	---	---	---	ANS25	ANS24
R/W	---	---	---	---	---	---	R/W	R/W
复位值	---	---	---	---	---	---	0	0

Bit7~Bit2　　未用

Bit1~Bit0　　ANS<25:24>:　模拟选择位，分别选择引脚 AN<25:24>的模拟或数字功能

1=　模拟输入。引脚被分配为模拟输入

0=　数字 I/O。引脚被分配给端口或特殊功能

7. 中断

7.1 中断概述

芯片具有 19 个中断源及中断向量：

中断源	中断描述	中断向量	同级优先序列
TMR1IF	Timer1 溢出中断	0-0x0003	1
TMR2IF	Timer2 与 PR2 匹配中断	1-0x000B	2
CPTIF	CPT 中断	2-0x0013	3
SSPIF	MSSP 中断	3-0x001B	4
TX0IF	USART0 发送中断	4-0x0023	5
RX0IF	USART0 接收中断	5-0x002B	6
ADIF	ADC 中断	6-0x0033	7
PWMIF	PWM 中断	7-0x003B	8
---	---	---	---
RX1IF	USART1 接收中断	9-0x004B	10
TX1IF	USART1 发送中断	10-0x0053	11
BCLIF	I^2C 总线冲突中断	11-0x005B	12
EEIF	Data Flash 写操作中断	12-0x0063	13
---	---	---	---
TKIF	触摸检测结束中断	14-0x0073	15
LVDIF	LVD 中断	15-0x007B	16
RBIF	PORTB 电平变化中断	16-0x0083	17
INTF	外部中断	17-0x008B	18
T0IF	Timer0 溢出中断	18-0x0093	19

芯片规定两个中断优先级，可实现两级中断嵌套。当一个中断已经响应，若有高级别中断发出请求，后者可以中断前者，实现中断嵌套。

当芯片没有设置高优先级中断时，芯片的各个中断的优先级是平等的，当一个中断正在进行的时候，不会响应另外一个中断，只有执行“RETI”指令后，才能响应下一个中断。多个中断同时发生时，MCU 将按预置的中断优先级响应中断。

7.2 中断重映射

系统默认中断向量偏移基地址为 0x0000，但某些应用场景下需要将偏移基地址设置为其他值，此功能可通过修改中断偏移地址寄存器 IREMAP 来实现。

IREMAP 寄存器为设置为 0x01 时，0X01<<9 =0X0200，即中断偏移基地址为 0X0200，则其他所有中断地址需加上偏移基地址才是实际的中断向量地址。比如 TIMER1 的向量地址重映射之后应为：0x0200+0x0003=0x0203

由上述描述可知，中断向量偏移基地址最小的偏移单元为 0.5KB。

中断重映射使用实例：

在程序空间最后 2KB 区域实现 ISP 功能，上电后，程序跳转至最后 2KB 起始地址，之后设置 IRMAP 为 0x0c。即中断偏移的基地址为 0x1800。在此 2KB 区域实现完成相应的 ISP 功能或者跳过 ISP 后，关闭所有的中断使能，跳转至主程序区域，然后将 IREMAP 设置为 0X00，中断偏移的基地址即为 0x0000。

外设中断请求寄存器（PIR1、PIR2、PIR3）在各自的标志位中记录各种中断请求。全局中断允许位GIE（INTCON<7>）在置 1 时允许所有未屏蔽的中断，而在清零时，禁止所有中断。可以通过 PIE1、PIE2、PIE3 寄存器中相应的允许位来禁止各个中断。复位时 GIE 被清零。

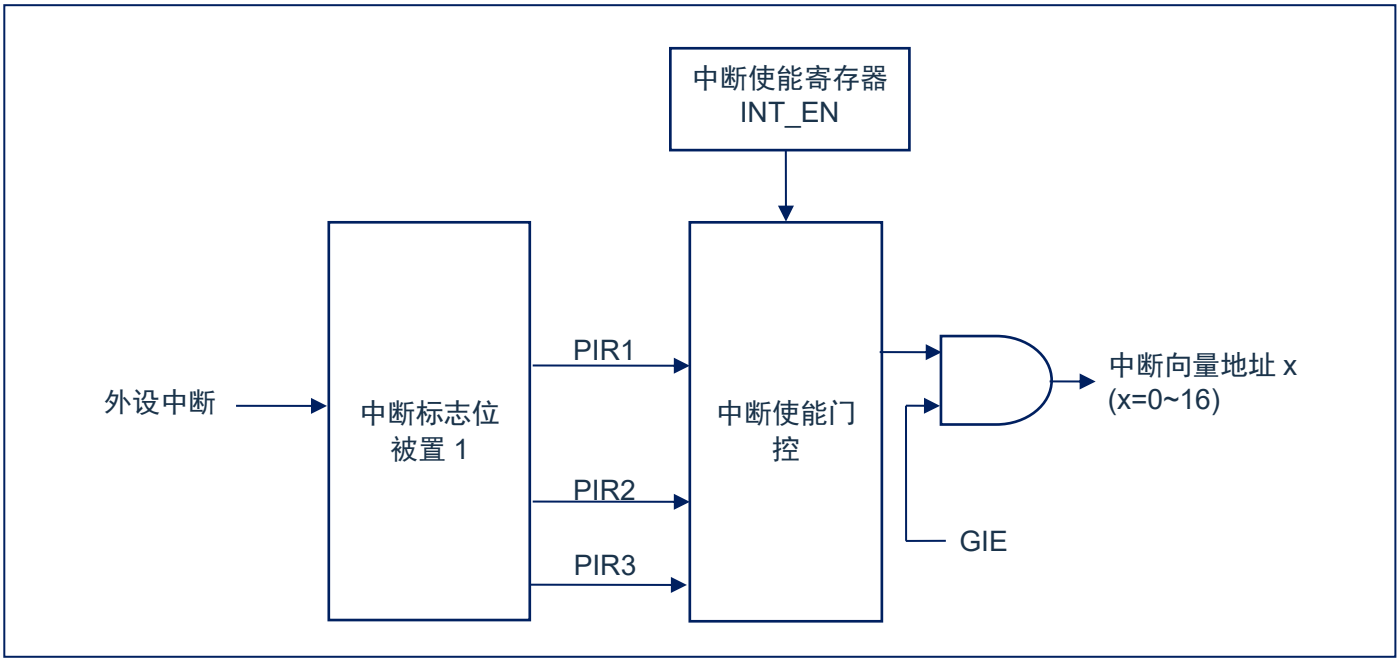


图 7-1：中断原理示意图

7.3 中断控制寄存器

7.3.1 中断控制寄存器

中断控制寄存器 INTCON 是可读写的寄存器，包含两个中断优先级控制；

当有中断条件产生时，无论对应的中断允许位或（INTCON 寄存器中的）全局允许位 GIE 的状态如何，中断标志位都将置 1。用户软件应在允许一个中断之前，确保先将相应的中断标志位清零。

中断控制寄存器 INTCON

0xF1	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTCON	GIE	HPRIIF	INTPS5	INTPS4	INTPS3	INTPS2	INTPS1	INTPS0
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	1	1	1	1	1	1

Bit7 GIE: 全局中断允许位
1= 允许所有未被屏蔽的中断
0= 禁止所有中断

Bit6 HPRIIF: 高优先级中断标志位
1= 当前响应的中断为高优先级中断
0= 当前响应的中断为低优先级中断

Bit5~Bit0 INTPS[5:0] 高优先级中断允许位
000000= PIR1[0]为高优先级中断
000001= PIR1[1]为高优先级中断
000010= PIR1[2]为高优先级中断
000011= PIR1[3]为高优先级中断
000100= PIR1[4]为高优先级中断
000101= PIR1[5]为高优先级中断
000110= PIR1[6]为高优先级中断
000111= PIR1[7]为高优先级中断
001000= PIR2[0]为高优先级中断
001001= PIR2[1]为高优先级中断
001010= PIR2[2]为高优先级中断
001011= PIR2[3]为高优先级中断
001100= PIR2[4]为高优先级中断
001101= PIR2[5]为高优先级中断
001110= PIR2[6]为高优先级中断
001111= PIR2[7]为高优先级中断
010000= PIR3[0]为高优先级中断
010001= PIR3[1]为高优先级中断
010010= PIR3[2]为高优先级中断
其他= 未用

7.3.2 外设中断允许寄存器

外设中断允许寄存器有 PIE1、PIE2 和 PIE3，在允许任何外设中断前，必须先将 INTCON 寄存器的 PEIE 位置 1。

外设中断允许寄存器 PIE1

0XF2	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	PWMIE	ADIE	RX0IE	TX0IE	SSPIE	CPTIE	TMR2IE	TMR1IE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7 PWMIE: PWM中断允许位
1= 允许PWM中断
0= 禁止PWM中断

Bit6 ADIE: A/D转换器（ADC）中断允许位
1= 允许ADC中断
0= 禁止ADC中断

Bit5 RX0IE: USART0接收中断允许位
1= 允许USART0接收中断
0= 禁止USART0接收中断

Bit4 TX0IE: USART0发送中断允许位
1= 允许USART0发送中断
0= 禁止USART0发送中断

Bit3 SSPIE: 主同步串行端口（MSSP）中断允许位
1= 允许MSSP中断
0= 禁止MSSP中断

Bit2 CPTIE: CPT中断允许位
1= 允许CPT中断
0= 禁止CPT中断

Bit1 TMR2IE: TIMER2与PR2匹配中断允许位
1= 允许TMR2与PR2匹配中断
0= 禁止TMR2与PR2匹配中断

Bit0 TMR1IE: TIMER1溢出中断允许位
1= 允许TIMER1溢出中断
0= 禁止TIMER1溢出中断

外设中断允许寄存器 PIE2

0xF3	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE2	LVDIE	TKIE	---	EEIE	BCLIE	TX1IE	RX1IE	---
R/W	R/W	R/W	---	R/W	R/W	R/W	R/W	---
复位值	0	0	---	0	0	0	0	---

Bit7 LVDIE LVD中断允许位
1= 允许LVD中断
0= 禁止LVD中断
Bit6 TKIE: 触摸检测结束中断允许位
1= 允许触摸检测结束中断
0= 禁止触摸检测结束中断
Bit5 未用
Bit4 EEIE: 程序EEPROM写操作中断允许位
1= 允许程序EEPROM写操作中断
0= 禁止程序EEPROM写操作中断
Bit3 BCLIE : 总线冲突中断允许位
1= 允许总线冲突中断
0= 禁止总线冲突中断
Bit2 TX1IE: USART1发送中断允许位
1= 允许USART1发送中断
0= 禁止USART1发送中断
Bit1 RC1IE: USART1接收中断允许位
1= 允许USART1接收中断
0= 禁止USART1接收中断
Bit0 未用

外设中断允许寄存器 PIE3

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE3	---	---	---	---	---	T0IE	INTIE	RBIE
R/W	---	---	---	---	---	R/W	R/W	R/W
复位值	---	---	---	---	---	0	0	0

Bit7~Bit3 未用
Bit2 T0IE: TIMER0溢出中断允许位
1= 允许TIMER0溢出中断
0= 禁止TIMER0溢出中断
Bit1 INTIE: INT外部中断允许位
1= 允许INT外部中断
0= 禁止INT外部中断
Bit0 RBIE: PORTB电平变化中断允许位
1= 允许PORTB电平变化中断
0= 禁止PORTB电平变化中断

7.3.3 外设中断请求寄存器

外设中断请求寄存器为 PIR1、PIR2 和 PIR3。当有中断条件产生时，无论对应的中断允许位或全局允许位 GIE 的状态如何，中断标志位都将置 1。用户软件应在允许一个中断之前，确保先将相应的中断标志位清零。

外设中断请求寄存器 PIR1

0xF5	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR1	PWMIF	ADIF	RX0IF	TX0IF	SSPIF	CPTIF	TMR2IF	TMR1IF
R/W	R/W	R/W	R	R	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7 PWMIF: PWM中断标志位
1= 发生了PWM中断（必须由软件清零）
0= 未发生PWM中断

Bit6 ADIF: A/D转换器中断标志位
1= A/D转换完成（必须由软件清零）
0= A/D转换未完成或尚未启动

Bit5 RX0IF: USART0接收中断标志位
1= USART0接收缓冲器非空（通过读RCREG0清零）
0= USART0接收缓冲器空

Bit4 TX0IF: USART0发送中断标志位
1= USART0发送缓冲器空（通过写TXREG0清零）
0= USART0发送缓冲器非空

Bit3 SSPIF: 主同步串行端口（MSSP）中断标志位
1= 产生了MSSP中断条件，在从中断服务程序返回前必须由软件清零。使该位置1的条件有：
- SPI
- 发生发送/接收
- I^2C从动/主控
- 发生发送/ 接收
- I^2C主控
- 发生的启动条件由MSSP模块完成
- 发生的停止条件由MSSP模块完成
- 发生的重新启动条件由MSSP模块完成
- 发生的应答条件由MSSP模块完成
- 当MSSP模块空闲时发生启动条件（多主机系统）

当MSSP模块空闲时发生停止条件（多主机系统）
0= 没有产生MSSP中断条件

Bit2 CPTIF: CPT中断标志位
1= 发生了TMR1寄存器的捕捉（必须由软件清零）
0= 没有发生TMR1寄存器的捕捉

Bit1 TMR2IF: TIMER2与PR2匹配中断标志位
1= 发生了TIMER2与PR2匹配（必须由软件清零）
0= TIMER2与PR2不匹配

Bit0 TMR1IF: TIMER1溢出中断标志位
1= TMR1寄存器溢出（必须由软件清零）
0= TMR1寄存器未溢出

注：写 PIR1 寄存器指令后面需要加一条 NOP 指令

外设中断请求寄存器 PIR2

0xF6	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR2	LVDIF	TKIF	---	EEIF	BCLIF	TX1IF	RC1IF	---
R/W	R/W	R/W	---	R/W	R/W	R	R	---
复位值	0	0	---	0	0	0	0	---

Bit7 LVDIF LVD中断标志位
1= 发生了LVD中断
0= 未发生LVD中断
Bit6 TKIF 触摸检测结束中断标志位（必须由软件清零）
1= 触摸检测已结束
0= 触摸检测未结束
Bit5 未用
Bit4 EEIF: 程序EEPROM写操作中断标志位
1= 写操作完成（必须由软件清零）
0= 写操作未完成或尚未启动
Bit3 BCLIF : 总线冲突中断标志位
1= 当配置为I^2C主控模式时，MSSP中发生了总线冲突
0= 未发生总线冲突
Bit2 TX1IF: USART1发送中断标志位
1= USART1发送缓冲器空（通过写TXREG1清零）
0= USART1发送缓冲器非空
Bit1 RC1IF: USART1接收中断标志位
1= USART1接收缓冲器非空（通过读RCREG1清零）
0= USART1接收缓冲器空
Bit0 未用

注：1.T0IF 位在 TMR0 计满归 0 时置 1。复位该标志位不会使 TMR0 计数值发生改变，可在将 T0IF 位清零前写 TMR0 寄存器，对计数初始值进行重载；
2.写 PIR2 寄存器指令后面需要加一条 NOP 指令。

外设中断允许寄存器 PIR3

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR3	---	---	---	---	---	T0IF	INTIF	RBIF
R/W	---	---	---	---	---	R/W	R/W	R/W
复位值	---	---	---	---	---	0	0	0

Bit7~Bit3 未用。
Bit2 T0IF: TIMER0溢出中断标志位
1= TMR0寄存器已经溢出（必须由软件清零）
0= TMR0寄存器未溢出
Bit1 INTIF: INT外部中断标志位
1= 发生INT外部中断（必须由软件清零）
0= 未发生INT外部中断
Bit0 RBIF: PORTB电平变化中断标志位
1= PORTB端口中至少有一个引脚的电平状态发生了改变（必须由软件清零）
0= 没有一个PORTB通用I/O引脚的状态发生了改变

注：写 PIR3 寄存器指令后面需要加一条 NOP 指令

7.3.4 中断偏移基地址寄存器 IREMAP

0xA1	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IREMAP	---	---	---	---	MAP03	MAP02	MAP01	MAP00
R/W	R	R	R	R	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

注：IREMAP为受保护寄存器。

Bit7-4 须写0 保留位

Bit3-0 MAP<3:0> 中断偏移地址

0X00= 中断偏移地址为0X0000

0X01= 中断偏移地址为(0X01<<9)，0X0200

0X02= 中断偏移地址为(0X02<<9)，0X0400

0X03= 中断偏移地址为(0X03<<9)，0X0600

0X04= 中断偏移地址为(0X04<<9)，0X0800

0X05= 中断偏移地址为(0X05<<9)，0X0A00

0X06= 中断偏移地址为(0X06<<9)，0X0C00

0X07= 中断偏移地址为(0X07<<9)，0X0E00

0X08= 中断偏移地址为(0X08<<9)，0X1000

0X09= 中断偏移地址为(0X09<<9)，0X1200

0X0a= 中断偏移地址为(0X0a<<9)，0X1400

0X0b= 中断偏移地址为(0X0b<<9)，0X1600

0X0c= 中断偏移地址为(0X0c<<9)，0X1800

0X0d= 中断偏移地址为(0X0d<<9)，0X1A00

0X0e= 中断偏移地址为(0X0e<<9)，0X1C00

0X0f= 中断偏移地址为(0X0f<<9)，0X1E00

修改 IREMAP 需要的指令序列（中间不能插入其他任何指令）：

```
CLR     INTCON
MOV     TA,#0AAH
MOV     TA,#055H
ORL     IREMAP,#01H
```

8. 定时计数器 TIMER0

8.1 定时计数器 TIMER0 概述

TIMER0 由如下功能组成：

- 8 位定时器/计数器寄存器（TMR0）；
- 8 位预分频器（与看门狗定时器共用）；
- 可编程内部或外部时钟源；
- 可编程外部时钟边沿选择；
- 溢出中断。

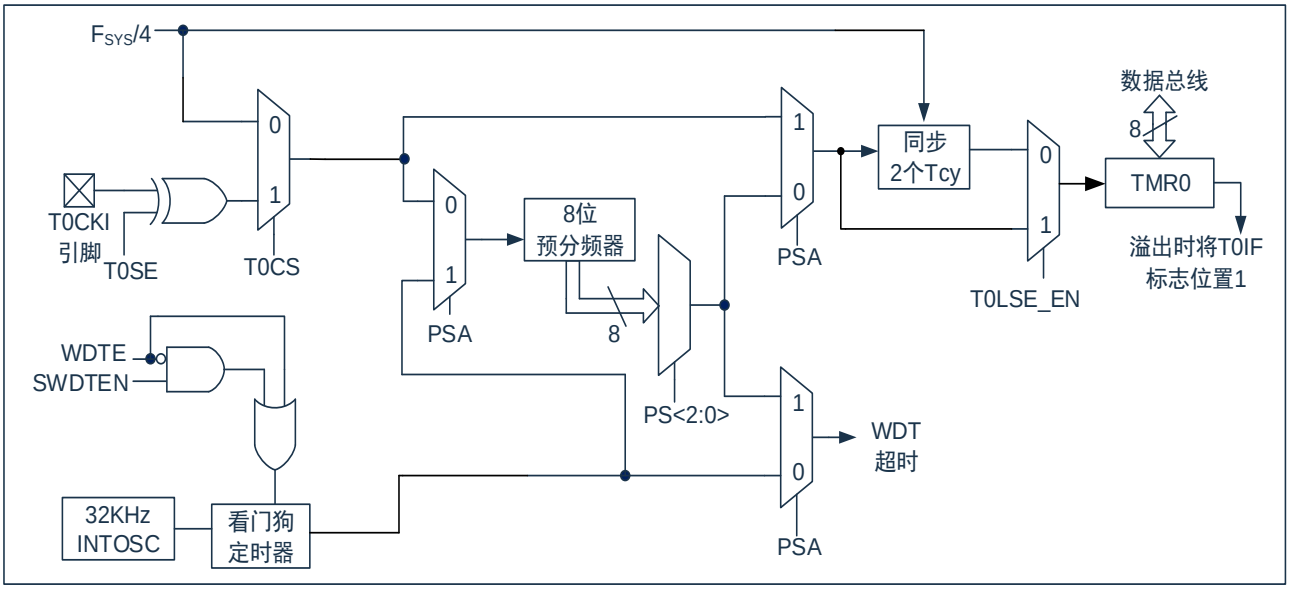


图 8-1：TIMER0/WDT 模块结构图

注：
1. T0SE、T0CS、PSA、PS<2:0>为OPTION_REG寄存器中的位。
2. SWDTEN为WDTCON寄存器中的位。
3. WDTEN位CONFIG中。

8.2 TIMER0 的工作原理

TIMER0 模块既可用作 8 位定时器也可用作 8 位计数器。

8.2.1 8 位定时器模式

用作定时器时，TIMER0 模块将在每个指令周期递增（不带预分频器）。通过将 OPTION_REG 寄存器的 T0CS 位清 0 可选择定时器模式。如果对 TMR0 寄存器执行写操作，则在接下来的两个指令周期将禁止递增。可调整写入 TMR0 寄存器的值，使得在写入 TMR0 时计入两个指令周期的延时。

8.2.2 8 位计数器模式

用作计数器时，TIMER0 模块将在 T0CKI 引脚的每个上升沿或下降沿递增。递增的边沿取决于 OPTION_REG 寄存器的 T0SE 位。通过将 OPTION_REG 寄存器的 T0CS 位置 1 可选择计数器模式。

8.2.3 软件可编程预分频器

TIMER0 和看门狗定时器（WDT）共用一个软件可编程预分频器，但不能同时使用。预分频器的分配由 OPTION_REG 寄存器的 PSA 位控制。要将预分频器分配给 TIMER0，PSA 位必须清 0。

TIMER0 模块具有 8 种预分频比选择，范围为 1:2 至 1:256。可通过 OPTION_REG 寄存器的 PS<2:0> 位选择预分频比。要使 TIMER0 模块具有 1:1 的预分频比，必须将预分频器分配给 WDT 模块。

预分频器不可读写。当预分频器分配给 TIMER0 模块时，所有写入 TMR0 寄存器的指令都将使预分频器清零。当预分频器分配给 WDT 时，CLRWDT 指令将同时清零预分频器和 WDT。

8.2.4 在 TIMER0 和 WDT 模块间切换预分频器

将预分频器分配给 TIMER0 或 WDT 后，在切换预分频比时可能会产生无意的器件复位。

要将预分频器从分配给 TIMER0 改为分配给 WDT 模块时，需要按如下顺序执行：

1. 关闭中断，避免在执行特定时序时进入中断程序
2. 将预分频器设置为最大值
3. 清零 TMR0
4. 设置预分频器分配给 WDT
5. 写 WDTCLR 清 WDT
6. 设置新的预分频比
7. 开启中断

要将预分频器从分配给 WDT 改为分配给 TIMER0 模块，必须执行以下指令序列。

1. 写 WDTCLR 清 WDT
2. 设置预分频器分配给 TIMER0，并设定分频比

8.2.5 TIMER0 中断

当 TMR0 寄存器从 FFh 溢出至 00h 时，不论是否允许 TIMER0 中断，PIR2 寄存器的 T0IF 中断标志位都会置 1，允许 TIMER0 中断时将发起中断请求。T0IF 位需由软件清零。

注：由于在休眠状态下定时器是关闭的，所以 TIMER0 中断无法唤醒处理器。

8.3 与 TIMER0 相关寄存器

有两个寄存器与 TIMER0 相关，8 位定时器/计数器（TMR0），8 位可编程控制寄存器（OPTION_REG）。

TMR0 为一个 8 位可读写的定时/计数器，OPTION_REG 为一个 8 位读写寄存器，用户可改变 OPTION_REG 的值，来改变 TIMER0 的工作模式等。请参看预分频寄存器（OPTION_REG）的应用。

8 位定时器/计数器 TMR0

0x89	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR0								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

注：写 TMR0 寄存器指令后面需要加一条 NOP 指令。

预分频器 OPTION_REG

0x8A	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	1	1	1	0	1	1

Bit7 RBPU: PORTB 上拉使能位
0= 由端口的各个锁存值使能 PORTB 上拉
1= 禁止 PORTB 上拉

Bit6 INTEDG: 触发中断的边沿选择位
0= INT 引脚下降沿触发中断
1= INT 引脚上升沿触发中断

Bit5 T0CS: TIMER0 时钟源选择位
0= 内部指令周期时钟（F_{SYS}/4）
1= T0CKI 引脚上的跳变沿

Bit4 T0SE: TIMER0 时钟源边沿选择位
0= 在 T0CKI 引脚信号从低电平跳变到高电平时递增
1= 在 T0CKI 引脚信号从高电平跳变到低电平时递增

Bit3 PSA: 预分频器分配位
0= 预分频器分配给 TIMER0 模块
1= 预分频器分配给 WDT

Bit2~Bit0 PS2~PS0: 预分频参数配置位

PS2	PS1	PS0	TMR0 分频比	WDT 分频比
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

9. 定时计数器 TIMER1

9.1 TIMER1 概述

TIMER1 模块是一个 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（TMR1H:TMR1L）
- 3 位预分频器
- 同步或异步操作
- 溢出时唤醒（仅外部时钟异步模式）
- 可编程内部或外部时钟源
- 通过 T1G 引脚门控 TIMER1（使能计数）
- 溢出中断
- 捕捉功能的时基

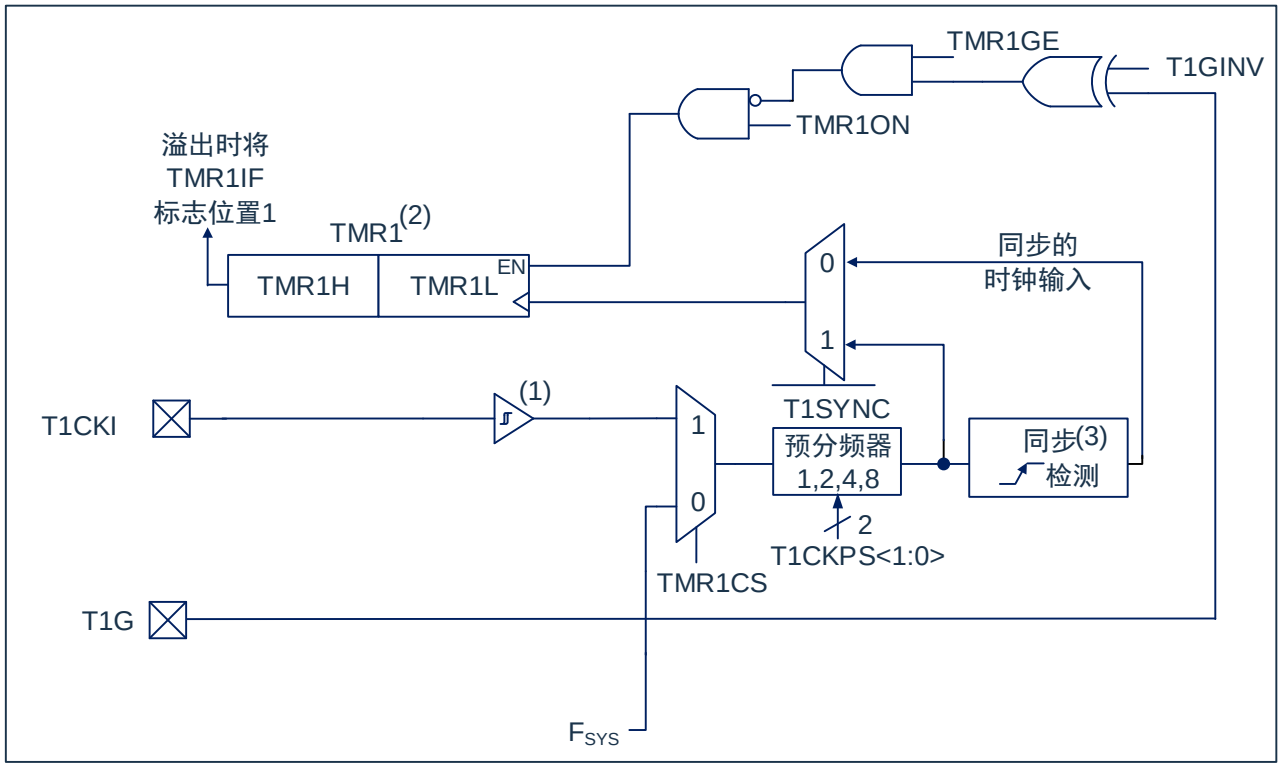


图9-1：TIMER1结构图

注：

1. Timer1 寄存器在上升沿递增。
2. 休眠时不进行同步。

9.2 TIMER1 的工作原理

TIMER1 模块是一个通过一对寄存器 TMR1H：TMR1L 访问的 16 位递增计数器。写入 TMR1H 或 TMR1L 可直接更新该计数器。

当与内部时钟源一同使用时，此模块用作计数器。当与外部时钟源一同使用时，此模块可用作定时器或计数器。

9.3 时钟源选择

T1CON 寄存器的 TMR1CS 位用于选择时钟源。当 TMR1CS=0 时，时钟源的频率为 F_{SYS}。当 TMR1CS=1 时，时钟源由外部提供。

时钟源	TMR1CS
F_{SYS}	0
T1CKI 引脚	1

9.3.1 内部时钟源

选择内部时钟源后，TMR1H:TMR1L 寄存器将以 F_{SYS} 的倍数为频率递增，具体倍数由 TIMER1 预分频器决定。

9.3.2 外部时钟源

选择外部时钟源后，TIMER1 模块可作为定时器或计数器。

计数时，TIMER1 在外部时钟输入 T1CKI 的上升沿递增。此外，计数器模式下的时钟可与单片机系统时钟同步或异步。

在计数器模式下，在出现以下一个或多个条件时，必须先经过一个下降沿，计数器才可以在随后的上升沿进行第一次递增计数（见图 9-2）：

- 使能 TIMER1。
- 对 TMR1H 或 TMR1L 执行了写操作。
- 禁止 TIMER1 时，T1CKI 为高电平；当重新使能 TIMER1 时，T1CKI 为低电平。

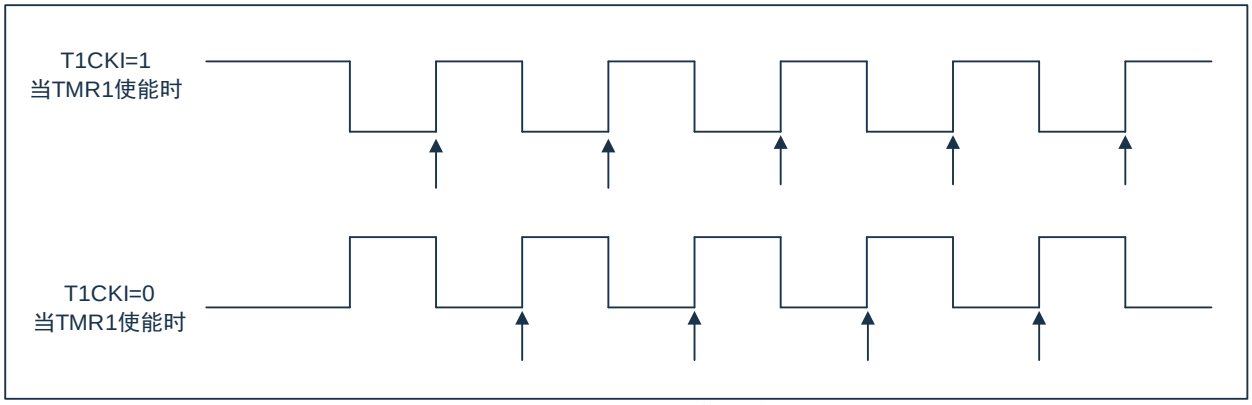


图 9-2：TIMER1 的递增边沿

注：
1) 箭头表示计数器递增。
2) 在计数器模式下，必须先经过一个下降沿，计数器才可以在随后的上升沿进行第一次递增计数。

9.4 TIMER1 预分频器

TIMER1 具有四种预分频比选择，允许对输入时钟进行 1、2、4 或 8 分频。T1CON 寄存器的 T1CKPS 位控制预分频计数器。不能直接对预分频计数器进行读或写操作；但是，通过写入 TMR1H 或 TMR1L 可清零预分频计数器。

9.5 在异步计数器模式下的 TIMER1 工作原理

如果 T1CON 寄存器中的控制位 T1SYNC 被置 1，外部时钟输入就不同步。定时器继续进行与内部相位时钟异步的递增计数。在休眠状态下定时器仍将继续运行，并在溢出时产生中断，从而唤醒处理器。但是，再用软件对定时器进行读/写操作时应该特别小心（请参见第 9.5.1 节“异步计数器模式下对 TIMER1 的读写”）。

注：
1) 当从同步操作切换到异步操作时，有可能漏过一个递增。
2) 当从异步操作切换到同步操作时，有可能产生一个误递增。

9.5.1 异步计数器模式下对 TIMER1 的读写操作

当定时器采用外部异步时钟工作时，对 TMR1H 或 TMR1L 的读操作将确保有效（由硬件负责）。但用户应牢记，用读两个 8 位值来读一个 16 位定时器本身就存在问题，这是因为在两次读操作之间定时器可能会溢出。

对于写操作，建议用户停止定时器后再写入所需数值。当寄存器正在递增计数时，向定时器的寄存器写入数据可能会产生写争用。从而会在 TMR1H:TMR1L 这对寄存器中产生不可预测的值。

9.6 TIMER1 门控

可用软件将 TIMER1 门控信号源配置为 T1G 引脚，这让器件可以直接使用 T1G 为外部事件定时。

注：必须将 T1CON 寄存器的 TMR1GE 位置 1 以使用 TIMER1 的门控信号。

可使用 T1CON 寄存器的 T1GINV 位来设置 TIMER1 门控信号的极性，门控信号可以来自 T1G 引脚。该位可将 TIMER1 配置为对事件之间的高电平时间或低电平时间进行计时。

9.7 TIMER1 中断

一对 TIMER1 寄存器（TMR1H:TMR1L）递增计数到 FFFFH 后，将溢出返回 0000H。当 TIMER1 溢出时，PIR1 寄存器的 TIMER1 中断标志位被置 1。要允许该溢出中断，用户应将以下位置 1：

- PIE1 寄存器中的 TIMER1 中断允许位；
- INTCON 寄存器中的 PEIE 位；
- INTCON 寄存器中的 GIE 位。

在中断服务程序中将 TMR1IF 位清零可以清除该中断。

注：再次允许该中断前，应将 TMR1H:TMR1L 这对寄存器以及 TMR1IF 位清零。

9.8 休眠期间的 TIMER1 工作原理

只有设置为异步计数器模式时，TIMER1 才可在休眠模式下工作。在该模式下，可使用时钟源使计数器进行递增计数。通过如下设置使定时器能够唤醒器件：

- T1CON 寄存器中的 TMR1ON 位必须置 1；
- PIE1 寄存器中的 TMR1IE 位必须置 1；
- INTCON 寄存器中的 PEIE 位必须置 1。

器件将在溢出时被唤醒并执行下一条指令。如果 INTCON 寄存器中的 GIE 位置 1，器件将调用中断服务程序。

9.9 TIMER1 相关寄存器

有 3 个寄存器与 TIMER1 相关，分别是数据寄存器 TMR1L、TMR1H，控制寄存器 T1CON。

IMER1 数据低位寄存器 TMR1L(8BH)

8BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR1L	TMR1<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0　　TMR1<7:0>:　　TIMER1 数据低 8 位

TIMER1 数据高位寄存器 TMR1H(8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR1H	TMR1<15:8>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0　　TMR1<15:8>:　　TIMER1 数据高 8 位

注：写 TMR1L 或者 TMR1H 寄存器指令后面需要加一条 NOP 指令

TIMER1 控制寄存器 T1CON

0x8D	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	----	T1SYNC	TMR1CS	TMR1ON
R/W	R/W	R/W	R/W	R/W	----	R/W	R/W	R/W
复位值	0	0	0	0	----	0	0	0

Bit7　　T1GINV:　　TIMER1 门控信号极性位
1=　TIMER1 门控信号高电平有效（当门控信号为高电平时 TIMER1 计数）
0=　TIMER1 门控信号低电平有效（当门控信号为低电平时 TIMER1 计数）

Bit6　　TMR1GE:　　TIMER1 门控使能位
如果TMR1ON=0，此位被忽略
如果 TMR1ON=1:　　1=TIMER1 计数由 TIMER1 门控功能控制
0=TIMER1 始终计数

Bit5~Bit4　　T1CKPS<1:0>:　　TIMER1 输入时钟预分频比选择位
11=　1:8 预分频比
10=　1:4 预分频比
01=　1:2 预分频比
00=　1:1 预分频比

Bit3　　未用

Bit2　　T1SYNC:　　TIMER1 外部时钟输入同步控制位
TMR1CS=1:　　1= 不与外部时钟输入同步
0= 与外部时钟输入同步
TMR1CS=0：忽略此位，TIMER1 使用内部时钟

Bit1　　TMR1CS:　　TIMER1 时钟源选择位
1=　来自 T1CKI 引脚的时钟源（上升沿触发）
0=　内部时钟源 F_{SYS}

Bit0　　TMR1ON:　　TIMER1 使能位
1=　使能 TIMER1
0=　禁止 TIMER1

10. 定时计数器 TIMER2

10.1 TIMER2 概述

TIMER2 模块是一个 8 位定时器/ 计数器，具有以下特性：

- 8 位定时器寄存器（TMR2）；
- 8 位周期寄存器（PR2）；
- TMR2 与 PR2 匹配时中断；
- 软件可编程预分频比（1:1，1:4 和 1:16）；
- 软件可编程后分频比（1:1 至 1:16）。

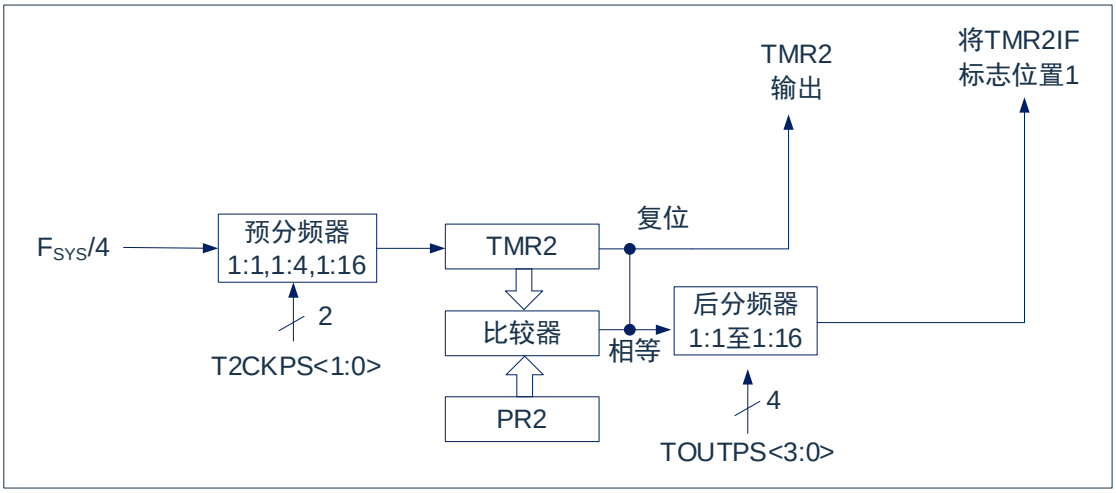


图 10-1：TIMER2 框图

10.2 TIMER2 的工作原理

TIMER2 模块的输入时钟是系统指令时钟（F_{SYS}/4）。时钟被输入到 TIMER2 预分频器，有如下几种分频比可供选择：1:1、1:4 或 1:16。预分频器的输出随后用于使 TMR2 寄存器递增。

持续将 TMR2 和 PR2 的值做比较以确定它们何时匹配。TMR2 将从 00h 开始递增直至与 PR2 中的值匹配。匹配发生时，会发生以下两个事件：

- TMR2 在下一递增周期被复位为 00h；
- TIMER2 后分频器递增。

TIMER2 与 PR2 比较器的匹配输出随后输入给 TIMER2 的后分频器。后分频器具有 1:1 至 1:16 的预分频比可供选择。TIMER2 后分频器的输出用于使 PIR1 寄存器的 TMR2IF 中断标志位置 1。

TMR2 和 PR2 寄存器均可读写。任何复位时，TMR2 寄存器均被设置为 00h 且 PR2 寄存器被设置为 FFh。

通过将 T2CON 寄存器的 TMR2ON 位置 1 使能 TIMER2；通过将 TMR2ON 位清零禁止 TIMER2。

TIMER2 预分频器由 T2CON 寄存器的 T2CKPS 位控制;TIMER2 后分频器由 T2CON 寄存器的 TOUTPS 位控制。

预分频器和后分频器计数器在以下情况下被清零：

- TMR2ON=0 时
- 发生任何器件复位（上电复位、看门狗定时器复位或欠压复位）。

注：写 T2CON 不会将 TMR2 清零，在 TMR2ON=0 时，TMR2 寄存器不能进行写操作。

10.3 TIMER2 相关的寄存器

有 3 个寄存器与 TIMER2 相关，分别是数据存储器 TMR2、周期寄存器 PR2 和控制寄存器 T2CON。

TIMER2 数据寄存器 TMR2

A5H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR2								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

注：写 TMR2 寄存器指令后面需要加一条 NOP 指令。

TIMER2 周期寄存器 PR2

A7H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PR2								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

TIMER2 控制寄存器 T2CON

A6H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	----	0	0	0	0	0	0	0

Bit7　　未用，读为 0

Bit6~Bit3　　TOUTPS<3:0>:　　TIMER2 输出后分频比选择位

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

Bit2　　TMR2ON:　　TIMER2 使能位

1=　使能 TIMER2
0=　禁止 TIMER2

Bit1~Bit0　　T2CKPS<1:0>:　　TIMER2 时钟预分频比选择位

00=　预分频值为 1
01=　预分频值为 4
1x=　预分频值为 16

11. 模数转换（ADC）

11.1 ADC 概述

模数转换器（ADC）可以将模拟输入信号转换为表示该信号的一个 12 位二进制数。器件使用的模拟输入通道共用一个采样电路。采样电路的输出与模数转换器的输入相连。模数转换器采用逐次逼近法产生一个 12 位二进制结果，并将该结果保存在 ADC 结果寄存器（ADRESL 和 ADRESH）中。

ADC 参考电压始终为内部产生。ADC 在转换完成之后可以产生一个中断。

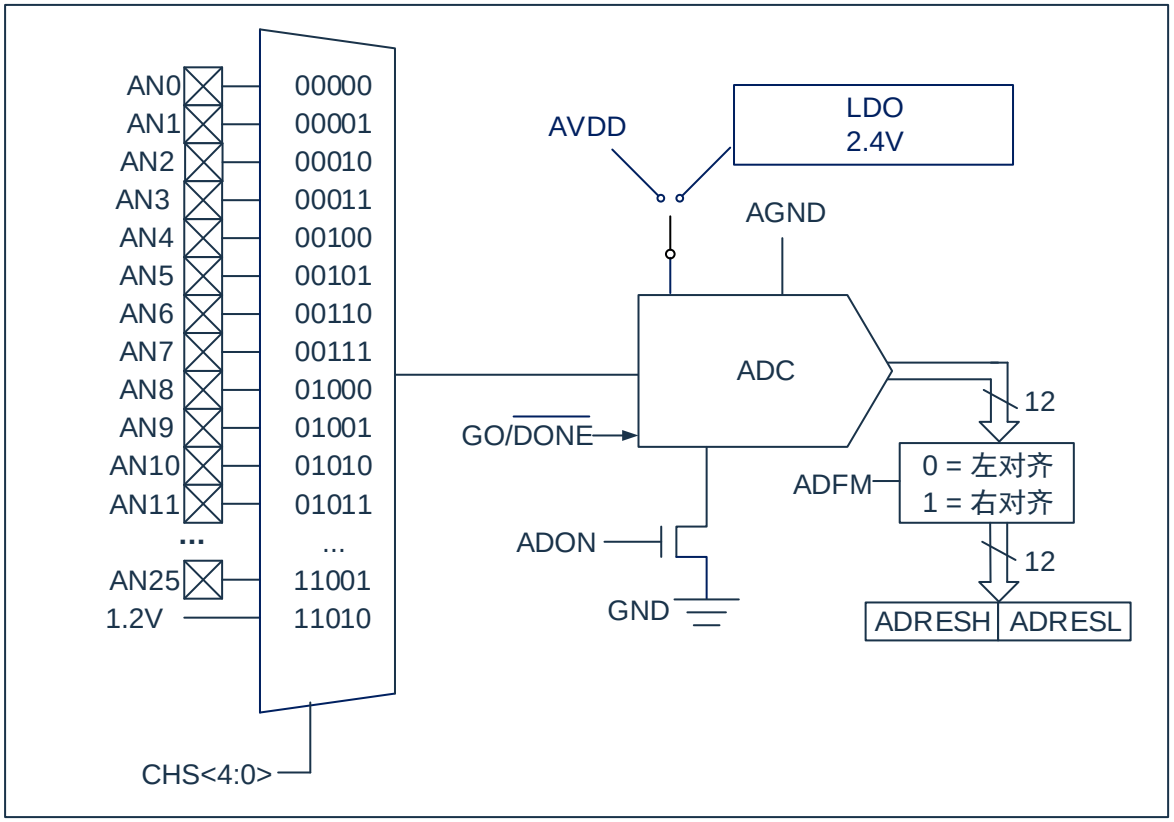


图 11-1：ADC 框图

11.2 ADC 配置

配置和使用 ADC 时，必须考虑如下因素：

- 端口配置；
- 参考电压选择；
- 通道选择；
- ADC 转换时钟源；
- 中断控制；
- 结果的存储格式。

11.2.1 端口配置

ADC 既可以转换模拟信号，又可以转换数字信号。当转换模拟信号时，应该通过将相应的 TRIS 位置 1，将 I/O 引脚配置为模拟输入引脚。更多信息请参见相应的端口章节。

注：对定义为数字输入的引脚施加模拟电压可能导致输入缓冲器出现过电流。

11.2.2 通道选择

由 ADCON0 寄存器的 CHS 位决定将哪个通道连接到采样保持电路。

如果更改了通道，在下一次转换开始前需要一定的延迟。更多信息请参见“ADC 工作原理”章节。

11.2.3 ADC 内部基准电压

芯片内置基准电压，需要检测该基准电压时，需把设置 CHS<4:0>位为 11010。

11.2.4 ADC 参考电压

ADC 的参考电压可选择内部 LDO 输出或芯片的 VDD 和 GND 提供。内部参考电压可选 2.4V。

当选择内部参考电压时，需要选择较慢的转换时钟，参考转换时钟章节。

注：当选择内部 LDO 作为参考电压时，ADC 有效精度会下降。检测电压越低，得到的 ADC 精度越高，建议输入电压设置为<1V。

11.2.5 转换时钟

可以通过软件设置 ADCON0 寄存器的 ADCS 位来选择转换的时钟源。有以下 4 种可能的时钟分频可供选择：

- F_{HSI}/16
- F_{HSI}/32
- F_{HSI}/64
- F_{HSI}/128

完成一位转换的时间定义为 T_{AD}。一个完整的 12 位转换需要 16 个 T_{AD} 周期。
必须符合相应的 T_{AD} 规范，才能获得正确的转换结果，下表为正确选择 ADC 时钟的示例。

不同参考电压和不同 V_{DD} 时，需要参考以下表格设置合理的分频。

参考电压	工作电压 (V)	最快分频设置	转换时间 (us)
		F_{SYS} = 16MHz	
V_{DD}	4.0~5.5	F_{HSI}/16	16
V_{DD}	2.7~5.5	F_{HSI}/32	32
2.4	4.0~5.5	F_{HSI}/32	32
2.4	2.7~4.0	F_{HSI}/64	64

11.2.6 ADC 中断

ADC 模块允许在完成模数转换后产生一个中断。ADC 中断标志位是 PIR1 寄存器中的 ADIF 位。ADC 中断允许位是 PIE1 寄存器中的 ADIE 位。ADIF 位必须用软件清零。每次转换结束后 ADIF 位都会被置 1，与是否允许 ADC 中断无关。

11.2.7 结果格式化

12 位 A/D 转换的结果可采用两种格式：左对齐或右对齐。由 ADCON1 寄存器的 ADFM 位控制输出格式。

当 ADFM=0 时，AD 转换结果左对齐，AD 转换结果为 12Bit；当 ADFM=1 时，AD 转换结果右对齐，AD 转换结果为 10Bit。

11.3 ADC 工作原理

11.3.1 启动转换

要使能 ADC 模块，必须将 ADCON0 寄存器的 ADON 位置 1，将 ADCON0 寄存器的 GO/$\overline{DONE}$位置 1 开始模数转换。

> 注：不能用开启 A/D 模块的同一指令将 GO/$\overline{DONE}$位置 1。

11.3.2 完成转换

当转换完成时，ADC 模块将：

- 清零 GO/$\overline{DONE}$位；
- 将 ADIF 标志位置 1；
- 用转换的新结果更新 ADRESH:ADRESL 寄存器。

11.3.3 终止转换

如果必须要在转换完成前终止转换，则可用软件清零 GO/$\overline{DONE}$位。不会用尚未完成的模数转换结果更新 ADRESH:ADRESL 寄存器。因此，ADRESH:ADRESL 寄存器将保持上次转换所得到的值。此外，在 A/D 转换终止以后，必须经过 2 个 T_{AD} 的延时才能开始下一次采集。延时过后，将自动开始对选定通道的输入信号进行采集。

> 注：器件复位将强制所有寄存器进入复位状态。因此，复位会关闭ADC模块并且终止任何待处理的转换。

11.3.4 ADC 在空闲模式下的工作原理

ADC 模块可以工作在空闲模式下。进入空闲模式之前，使能 ADC 转换，进入空闲模式后，ADC 依然可以继续工作，直到 ADC 转换完成；如果此时 ADC 中断标志位被置一，芯片将会被中断唤醒。

11.3.5 ADC 在休眠模式下的工作原理

ADC 模块不可以工作在休眠模式下。休眠模式下 ADC 输入时钟被关闭。

11.3.6 A/D 转换步骤

如下步骤给出了使用 ADC 进行模数转换的示例：

1. 端口配置：
   - 将引脚配置为输入引脚（见 TRIS 寄存器）。
2. 配置 ADC 模块：
   - 选择 ADC 转换时钟；
   - 选择 ADC 输入通道；
   - 选择结果的格式；
   - 启动 ADC 模块。
3. 配置 ADC 中断（可选）：
   - 清零 ADC 中断标志位；
   - 允许 ADC 中断；
   - 允许外设中断；
   - 允许全局中断。
4. 等待所需的采集时间。
5. 将 GO/$\overline{\text{DONE}}$置 1 启动转换。
6. 由如下方法之一等待 ADC 转换结束：
   - 查询 GO/$\overline{\text{DONE}}$位；
   - 等待 ADC 中断（允许中断）。
7. 读 ADC 结果。
8. 将 ADC 中断标志位清零（如果允许中断的话，需要进行此操作）。

11.4 ADC 相关寄存器

主要有 4 个 RAM 与 AD 转换相关，分别是控制寄存器 ADCON0 和 ADCON1，数据寄存器 ADRESH 和 ADRESL。

AD 控制寄存器 ADCON0

9EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	ADON
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit6 ADCS<1:0>: A/D转换时钟选择位
00= F_{HSI}/16
01= F_{HSI}/32
10= F_{HSI}/64
11= F_{HSI}/128

Bit5~Bit2 CHS<3:0>: 模拟通道选择位低四位与CHS4组成五位通道选择
CHS<4:0>:
00000= AN0
00001= AN1
00010= AN2
00011= AN3
00100= AN4
00101= AN5
00110= AN6
00111= AN7
01000= AN8
01001= AN9
01010= AN10
01011= AN11
…
10111= AN23
11000= AN24
11001= AN25
11010= 1.2V（固定参考电压）

Bit1 GO/$\overline{DONE}$: A/D转换状态位
1= A/D转换正在进行。将该位置1启动A/D转换。当A/D转换完成以后，该位由硬件自动清零
0= A/D转换完成/或不在进行中

Bit0 ADON: ADC使能位
1= 使能ADC
0= 禁止ADC，不消耗工作电流

AD 数据寄存器高位 ADCON1

9FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	ADFM	CHS4	----	----	----	LDO_EN	LDO_SEL1	LDO_SEL0
R/W	R/W	R/W	----	----	----	R/W	R/W	R/W
复位值	0	0	----	----	----	0	0	0

Bit7 ADFM: A/D转换结果格式选择位
1= 右对齐
0= 左对齐
Bit6 CHS4: 通道选择位
Bit5~Bit3 未用
Bit2 LDO_EN: 内部参考电压使能位
1= 使能ADC内部LDO参考电压
当选择内部LDO作参考电压时，ADC最大有效精度为8位
0= VDD作为ADC参考电压
Bit1~Bit0 LDO_SEL<1:0>: 参考电压选择位
10= 2.4V
其他 保留

AD 数据寄存器高位 ADRESH，ADFM=0

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	ADRES11	ADRES10	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4
R/W	R	R	R	R	R	R	R	R
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0 ADRES<11:4>: ADC结果寄存器位
12位转换结果的高8位

AD 数据寄存器低位 ADRESL，ADFM=0

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES3	ADRES2	ADRES1	ADRES0	----	----	----	----
R/W	R	R	R	R	----	----	----	----
复位值	X	X	X	X	----	----	----	----

Bit7~Bit4 ADRES<3:0>: ADC结果寄存器位
12位转换结果的低4位
Bit3~Bit0 未用

AD 数据寄存器高位 ADRESH，ADFM=1

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	----	----	----	----	----	----	ADRES11	ADRES10
R/W	----	----	----	----	----	----	R	R
复位值	----	----	----	----	----	----	X	X

Bit7~Bit2　　未用

Bit1~Bit0　　ADRES<11:10>:　ADC结果寄存器位

12位转换结果的高2位

AD 数据寄存器低位 ADRESL，ADFM=1

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2
R/W	R	R	R	R	R	R	R	R
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0　　ADRES<9:2>:　ADC结果寄存器位

12位转换结果的第2-9位

注：在 ADFM=1 的情况下 AD 转换结果只保存 12 位结果的高 10 位，其中 ADRESH 保存高 2 位，ADRESL 保存第 2 位至第 9 位。

12. LED 驱动模块

芯片内置 LED 驱动模块，配置了两种驱动模式：LED 矩阵模式和 LED 点阵扫描模式。

LED 模块的矩阵扫描模式最大可驱动 11*8 的数码管，点阵模式仅利用 9 个引脚就可驱动 8*9 的数码管，程序只需要把相关控制位和显示数据设置好，芯片管脚自动输出驱动 LED 的波形（硬件驱动）。

12.1 LED 功能使能

将 LEDCON0(113H)的第 6 位 LEDEN 置 1，允许 LED 驱动功能；LEDADD 的第 6 位置 0，选择矩阵扫描模式，否则选择点阵扫描模式；

将 LEDEN 都置 0，关闭 LED 模块。

12.2 LED 功能管脚设置

若使能 LED 矩阵扫描模式，必须设置相应的 SEG 口和 COM 口为输出态，并输出“0”，即将相应的 TRIS 位和 PORT 位置“0”。

若使能 LED 点阵扫描模式，必须设置相应的 LED 口为输入态，即将相应的 TRIS 位置“1”。

12.3 LED 功能 COM 口设置

LED 的 COM 口或者 LED 口设置方式如下：

设置 I/O 口方向和数据寄存器，LED 功能设置相应管脚为输出态并输出低电平。

COMSEL[2:0]	COM 口个数(矩阵扫描模式)	LED 口个数(点阵扫描模式)
000	4	5
001	5	6
010	6	7
011	8	8
100	2	9
101	3	9
110	3	9
111	4	9

设置 COMEN 寄存器，将相应管脚设置为 LED 功能的 COM 口或者 LED 口,LED0-LED7；LED8 口的设置在 SEGEN0 寄存器里。

在传统 LED 扫描模式，如果用户在使用过程中，COM 口不按照顺序排列。例如将 COM3-COM6 作为 LED 功能的 COM 口，COM0-COM2 作为普通 I/O 口，可以做以下设置：

- 将 COM 口个数设置成 8 个 COM，COMSEL=“011”；
- 将 COM_EN 寄存器的 COM3-COM6 置 1，COM0-COM2、COM7 清 0。

此时，COM3-COM6 为 LED 功能的 COM 口，其输出占空比为 1/8。而 COM0-COM2、COM7 可作为普通 I/O 口。

在 LED 点阵模式下，LED 口的个数设置和上述的原理一样。

12.4 LED 功能的 SEG 口设置

使能 LED 功能的 SEG 口必须满足以下条件：

1. 设置相应管脚状态，LED 功能设置相应管脚为输出态并输出“0”（仅 LED 矩阵扫描模式）；
2. 设置 SEGEN0、SEGEN1、SEGEN2 寄存器中相应管脚为 LED 驱动功能（仅 LED 矩阵扫描模式）；
3. 设置 SEGEN2 寄存器中 SEG 口输出电流或者 LED 口的拉电流。

注：在 LED 点阵模式下，LED 口既做 SEG 口又做 COM 口

12.5 LED 功能的数据设置

设置 LED 显示数据需以下步骤：

1. 设置 LEDADD 寄存器的第 7 位 LEDCS=1，允许读写数据；
2. 设置 LEDADD 的 0-6 位数据地址；
3. 设置 LEDDATA 数据（没有用作 LED 功能用的管脚，其相应的 LEDDATA 位需设置为“0”）；
4. 重复步骤 2-3 设置其它地址数据；
5. 设置完成后关闭数据读写位 LEDCS=0。

LED 矩阵扫描模式的地址和数据对应关系如下表：

LEDADD	LEDDATA								
00H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG0
01H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG1
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
16H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG22
17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG23
	COM7	COM6	COM5	COM4	COM3	COM2	COM1	COM0	

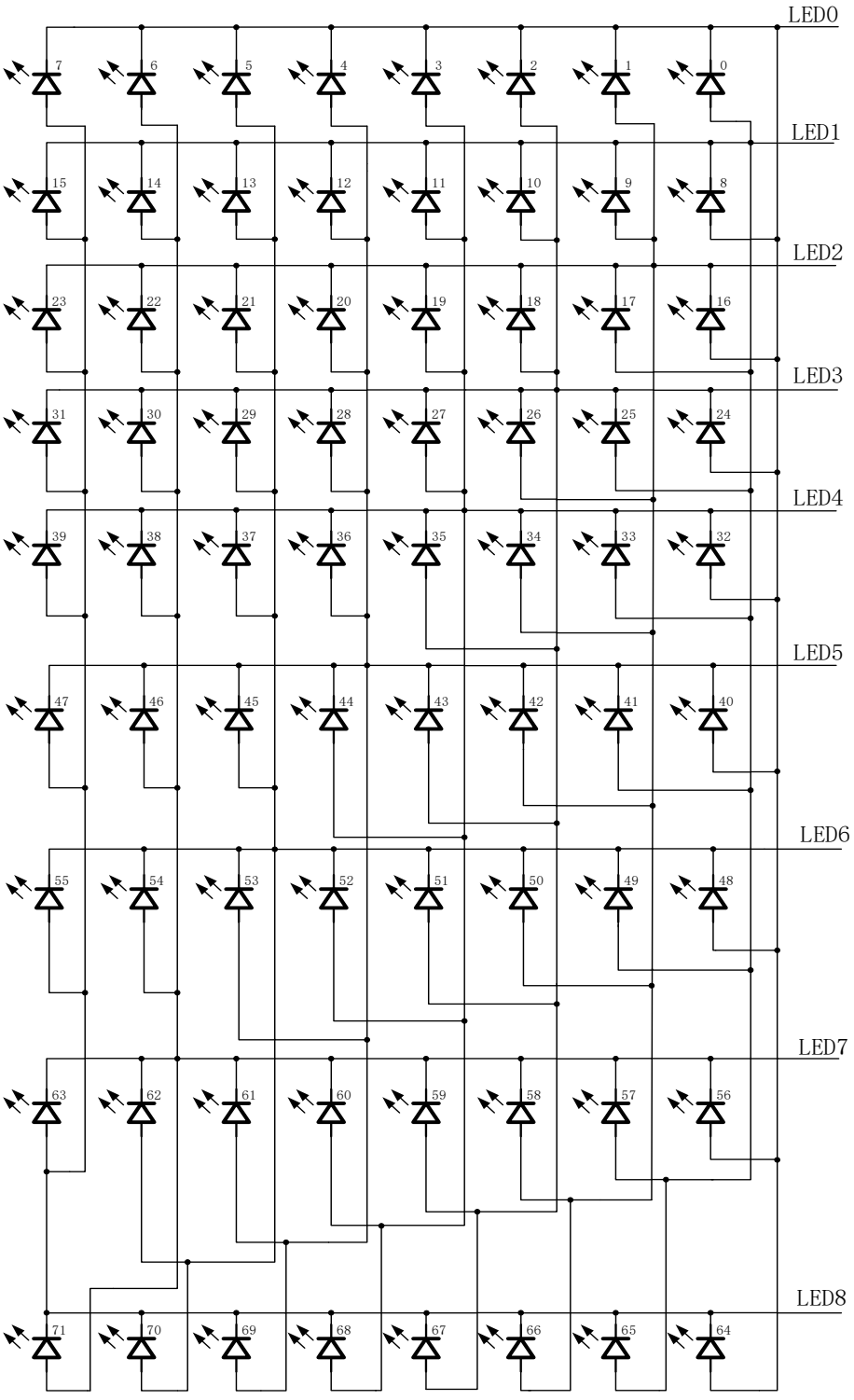
LED 点阵扫描模式的地址和数据对应关系如下表：

LEDADD	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
00H	7	6	5	4	3	2	1	0
01H	15	14	13	12	11	10	9	8
02H	23	22	21	20	19	18	17	16
03H	31	30	29	28	27	26	25	24
04H	39	38	37	36	35	34	33	32
05H	47	46	45	44	43	42	41	40
06H	55	54	53	52	51	50	49	48
07H	63	62	61	60	59	58	57	56
08H	71	70	69	68	67	66	65	64

12.6 LED 点阵扫描接线示意图

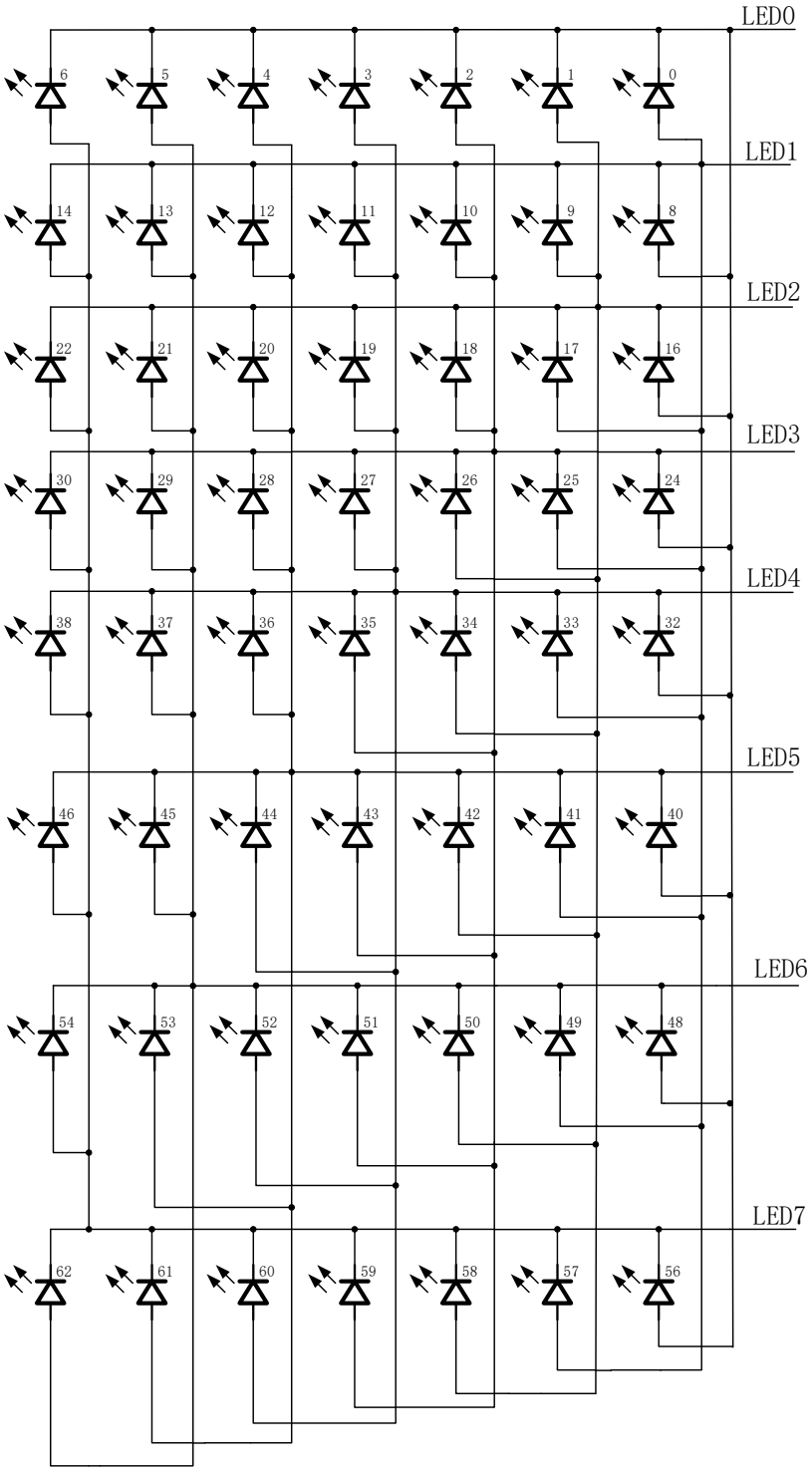
LED 点阵扫描最大支持 9*8=72 个灯，扫描端口为 LED0~LED8，分别对应 00H~08H 的显存地址，使能后按 LED0→LED8 的顺序逐一扫描。通过配置 COMSEL<2:0>的值可选择 5*4、6*5、7*6、8*7、9*8 等不同大小的 LED 灯点阵。

12.6.1 9*8 点阵示意图



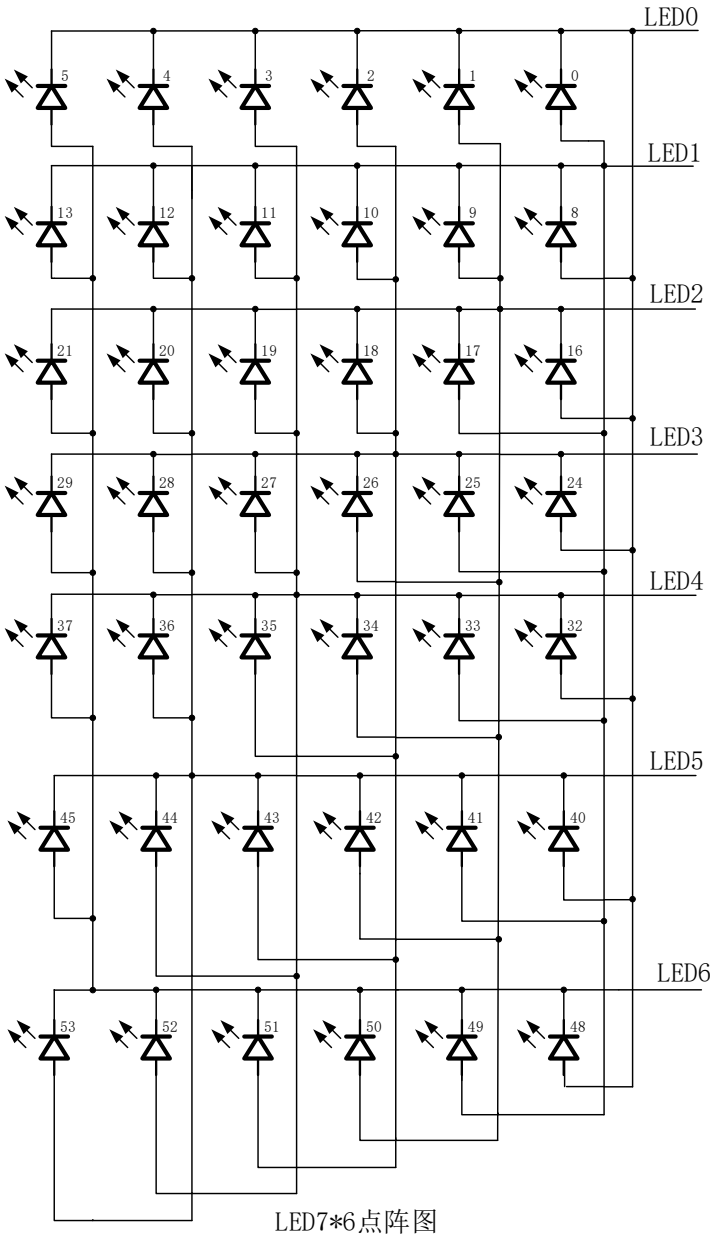
LED9*8点阵图

12.6.2 8*7 点阵示意图



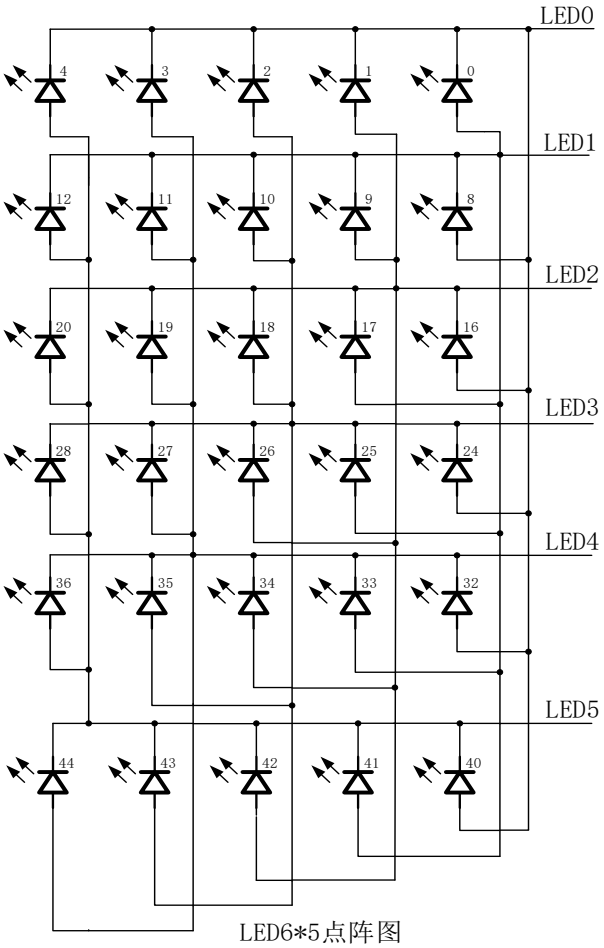
LED8*7点阵图

12.6.3 7*6 点阵示意图



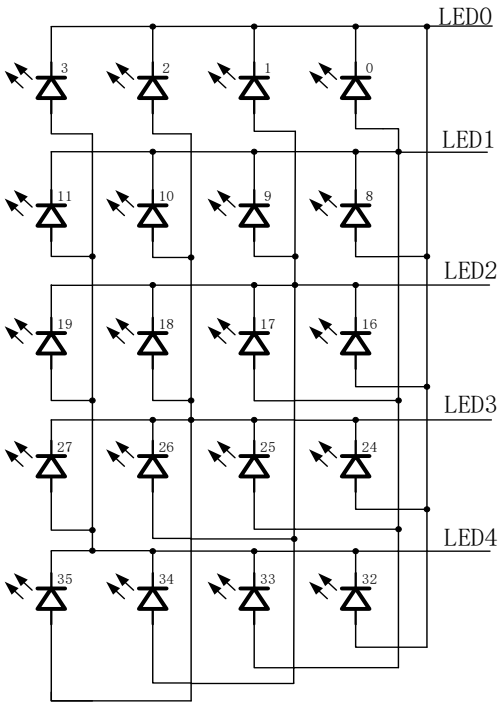
LED7*6点阵图

12.6.4 6*5 点阵示意图



LED6*5点阵图

12.6.5 5*4 点阵示意图



LED5*4点阵图

12.7 LED 点阵扫描的数据输出方式

LED 点阵扫描模式下，每个显存里的 8bit 数据分别对应 LED 扫描端口上的 8 个 LED 灯，数据为 1 时 LED 灯点亮，为 0 时 LED 灯熄灭。以 9*8 点阵为例，显存数据与 LED 灯有如下对应关系：

（1） 地址为 00H 显存的 Bit0~Bit7 分别对应 LED0 端口上标号为 0~7 的 LED 灯；

（2） 地址为 01H 显存的 Bit0~Bit7 分别对应 LED1 端口上标号为 8~15 的 LED 灯；

（3） 地址为 02H 显存的 Bit0~Bit7 分别对应 LED2 端口上标号为 16~23 的 LED 灯；

（4） 地址为 03H 显存的 Bit0~Bit7 分别对应 LED3 端口上标号为 24~31 的 LED 灯；

（5） 地址为 04H 显存的 Bit0~Bit7 分别对应 LED4 端口上标号为 32~39 的 LED 灯；

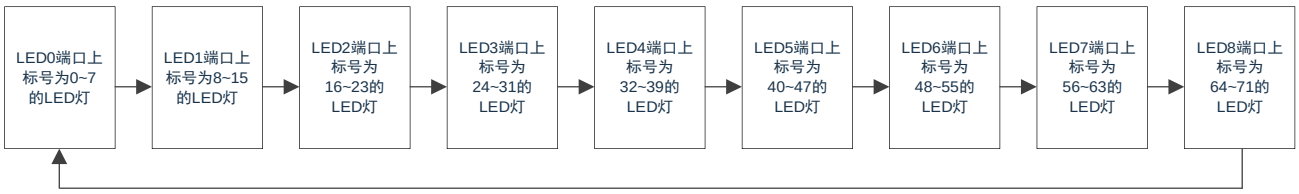
（6） 地址为 05H 显存的 Bit0~Bit7 分别对应 LED5 端口上标号为 40~47 的 LED 灯；

（7） 地址为 06H 显存的 Bit0~Bit7 分别对应 LED6 端口上标号为 48~55 的 LED 灯；

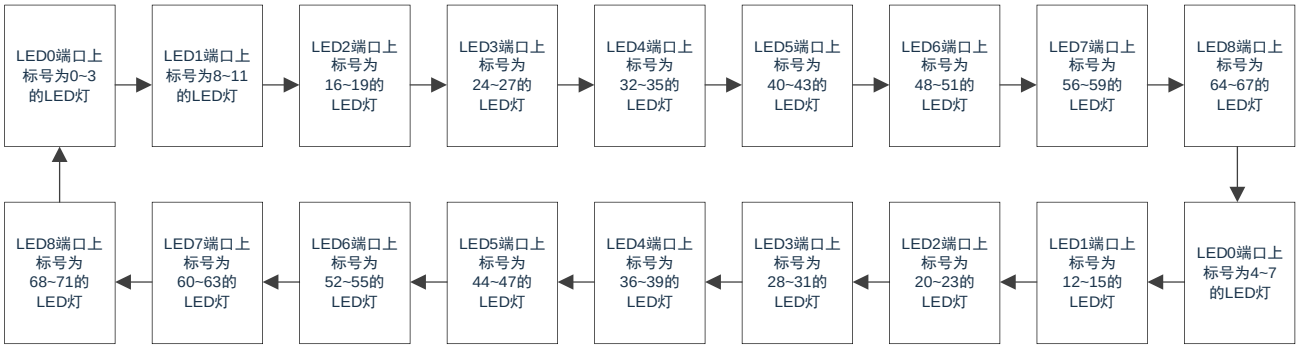
（8） 地址为 07H 显存的 Bit0~Bit7 分别对应 LED7 端口上标号为 56~63 的 LED 灯；

（9） 地址为 08H 显存的 Bit0~Bit7 分别对应 LED8 端口上标号为 64~71 的 LED 灯。

LED 点阵扫描模式下，通过 LEDCON0 寄存器的 LEDDATM 位可以选择对应地址的数据输出方式，LEDDATM=0 时，一个扫描周期内每个地址扫描 1 次，一次输出 8 位数据，即一次最多点亮 8 个 LED 灯，以 9*8 点阵为例，扫描流程如下图所示：



LEDDATM=1 时，一个扫描周期内每个地址扫描 2 次，第一次输出低 4 位数据，第二次输出高 4 位数据，即一次最多点亮 4 个 LED 灯，以 9*8 点阵为例，扫描流程如下图所示：



12.8 LED 相关寄存器

LED 驱动功能相关寄存器有：控制寄存器 LEDCON0，地址寄存器 LEDADD，数据寄存器 LEDDATA，口线设置寄存器 COMEN、SEGEN0、SEGEN1、SEGEN2。

LED 控制寄存器 LEDCON0

A9H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LEDCON0	LEDDATM	LEDEN	COMSEL[1:0]		LEDCLK[3:0]			
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7 LEDDATM: 点阵扫描模式下地址的数据输出模式选择

0: 一个扫描周期内每个地址扫描 1 次，一次输出 8 位数据

1: 一个扫描周期内每个地址扫描 2 次，第一次输出低 4 位数据，第二次输出高 4 位数据

Bit6 LEDEN: LED 模块使能

0: 禁止 LED 模块

1: 使能 LED 模块

Bit5~Bit4 COMSEL[1:0]: LED 模块 COM 口个数选择（COMSEL[2]在 LEDADD 寄存器第 5 位）

	传统的扫描模式	点阵扫描模式
000:	4COM	5COM
001:	5COM	6COM
010:	6COM	7COM
011:	8COM	8COM
100:	2COM	9COM
101:	3COM	9COM
110:	3COM	9COM
111:	4COM	9COM

Bit3~Bit0 LEDCLK[3:0]: LED 频率选择

	矩阵扫描模式	点阵扫描模式	
		LEDDATM=0	LEDDATM=1
0000:	16MHz/64	16MHz/1024	16MHz/512
0001:	16MHz/128	16MHz/1024	16MHz/512
0010:	16MHz/256	16MHz/1024	16MHz/512
0011:	16MHz/512	16MHz/1024	16MHz/512
0100:	16MHz/1024	16MHz/1024	16MHz/512
0101:	16MHz/2048	16MHz/2048	16MHz/1024
0110:	16MHz/4096	16MHz/4096	16MHz/2048
0111:	16MHz/8192	16MHz/8192	16MHz/4096
1x00:	16MHz/16384	16MHz/16384	16MHz/8192
1x01:	16MHz/32768	16MHz/32768	16MHz/16384
1x10:	16MHz/65536	16MHz/65536	16MHz/32768
1x11:	16MHz/131072	16MHz/131072	16MHz/65536

LED 地址寄存器 LEDADD

AAH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LEDADD	LEDCS	LEDM	COMSEL[2]	LEDADD[4:0]				
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7 LEDCS: LED 数据读写使能
0: 禁止读写 LED 数据
1: 允许读写 LED 数据
Bit6 LEDM: LED 扫描模式选择
0: 矩阵扫描模式
1: 点阵扫描模式
Bit5 COMSEL[2]: COM 选择最高位，COMSEL[2:0]设置 COM 口数量
Bit4~Bit0 LEDADD[4:0]: LED 地址选择
LED 地址范围 00H-0AH

注：写 LEDADD 寄存器指令后面需要加一条 NOP 指令。

LED 数据寄存器 LEDDATA

ABH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LEDDATA	LEDDATA[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0 LEDDATA[7:0]: LED 数据设置，写入 LEDADD 对应地址的数据

注：写 LEDDATA 寄存器指令后面需要加一条 NOP 指令。

LED 功能矩阵扫描模式 COM 口/点阵扫描模式 LED 口控制寄存器 COMEN

AFH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
COMEN	COM7EN	COM6EN	COM5EN	COM4EN	COM3EN	COM2EN	COM1EN	COM0EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 COMxEN: COM 口/LED 口功能设置
0: 对应 COMx 口/LEDx 口为普通 I/O 口(x=0-7)
1: 对应 COMx 口/LEDx 口为 LED 功能的 COM 口/LED 口(x=0-7)

注：写 COMEN 寄存器指令后面需要加一条 NOP 指令。

LED 功能 SEG 口控制寄存器 SEGEN0

AEH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGEN0	SEG7EN	SEG6EN	SEG5EN	SEG4EN	SEG3EN	SEG2EN	SEG1EN	SEG0EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit1 SEGxEN: SEG 口功能设置
0: 对应 SEGx 口为普通 I/O 口(x=1-7)
1: 对应 SEGx 口为 LED 功能的 SEG 口(x=1-7)
Bit0 SEG0EN: SEG0 口/LED8 口功能设
0: 对应 SEG0 口/LED8 口为普通 I/O 口
1: 对应 SEG0 口/LED8 口为 LED 功能的 SEG 口/LED8 口

注：写 SEGEN0 寄存器指令后面需要加一条 NOP 指令。

LED 功能 SEG 口控制寄存器 SEGEN1

ADH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGEN1	----	----	----	----	----	SEG10EN	SEG9EN	SEG8EN
R/W	----	----	----	----	----	R/W	R/W	R/W
复位值	----	----	----	----	----	0	0	0

Bit7~Bit3 未用
Bit2~Bit0 SEGxEN: SEG 口功能设置
0: 对应 SEGx 口为普通 I/O 口(x=8-10)
1: 对应 SEGx 口为 LED 功能的 SEG 口(x=8-10)

注：写 SEGEN1 寄存器指令后面需要加一条 NOP 指令。

LED 功能 SEG 口/LED 口拉电流控制寄存器 SEGEN2

ACH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGEN2	SEGDRI[3:0]				----	----	LED_DT <1:0>	
R/W	R/W	R/W	R/W	R/W	----	----	R/W	R/W
复位值	0	0	0	0	----	----	0	0

Bit7~Bit4 SEGDRI[3:0]: SEG 口驱动电流/LED 口拉电流设置
0000: SEG 口驱动电流/LED 口拉电流为 0
0001: SEG 口驱动电流/LED 口拉电流为 2mA
0010: SEG 口驱动电流/LED 口拉电流为 4mA
0011: SEG 口驱动电流/LED 口拉电流为 6mA
…
1110: SEG 口驱动电流/LED 口拉电流为 28mA
1111: SEG 口驱动电流/LED 口拉电流为 30mA
Bit3~Bit2 未用
Bit1~Bit0 LED_DT<1:0> LED 点阵模式 SEG 口死区放电时间选择位
00= 保留
01= 1us~2us
10= 2us~3us
11= 4us~5us

注：写 SEGEN2 寄存器指令后面需要加一条 NOP 指令。

13. 通用同步/异步收发器(USART0 和 USART1)

通用同步/异步收发器（USART）模块是一个串行 I/O 通信外设。该模块包括所有执行与器件程序执行无关的输入或输出串行数据传输所必需的时钟发生器、移位寄存器和数据缓冲器。USART 也可称为串行通信接口（SerialCommunicationsInterface，SCI），它可被配置为能与 CRT 终端和个人计算机等外设通信的全双工异步系统；也可以被配置为能与 A/D 或 D/A 集成电路、串行 EEPROM 等外设或其他单片机通信的半双工同步系统。与之通信的单片机通常不具有产生波特率的内部时钟，它需要主控同步器件提供外部时钟信号。

注：USART0 和 USART1 功能完全一样，以下章节描述中，x 值为 0，1。

USARTx 模块包含如下功能：

- 全双工异步发送和接收
- 单字符输出缓冲器
- 双字符输入缓冲器
- 接收到字符的帧错误检测
- 半双工同步从动模式
- 可将字符长度编程为 8 位或 9 位
- 输入缓冲溢出错误检测
- 半双工同步主控模式
- 同步模式下，可编程时钟极性

以下图 13-1 和图 13-2 为 USARTx 收发器的框图。

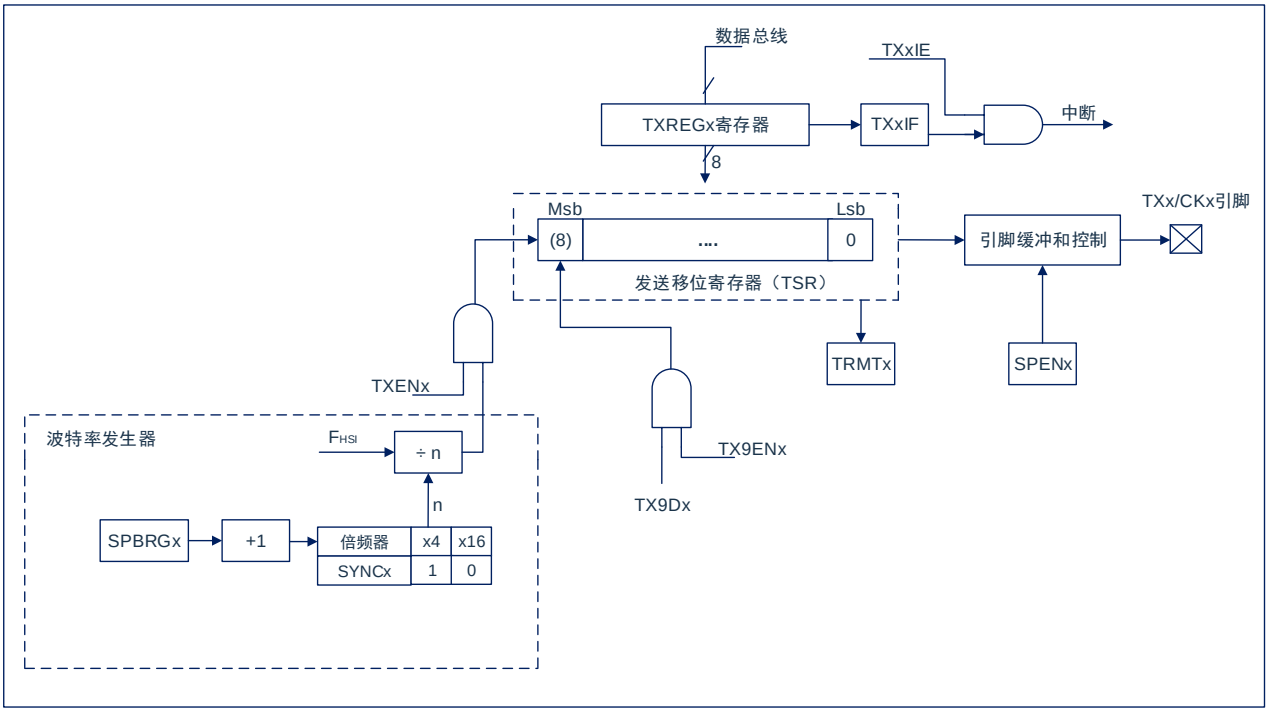


图13-1：USARTx发送框图

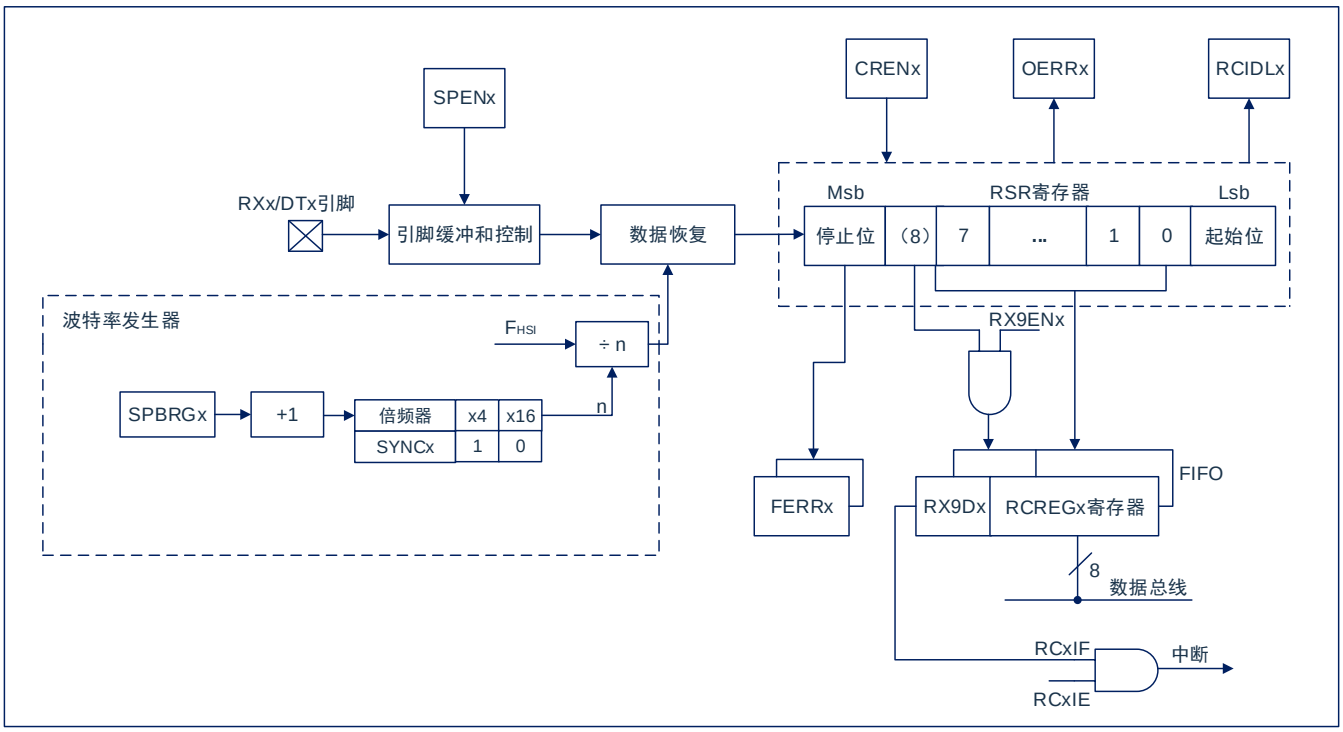


图13-2：USARTx接收框图

USARTx 模块的操作是通过 3 个寄存器控制的：

- 发送状态和控制寄存器（TXSTAx）
- 接收状态和控制寄存器（RCSTAx）
- 波特率控制寄存器（SPBRGx）

13.1 USARTx 异步模式

USARTx 使用标准不归零码（non-return-to-zero，NRZ）格式发送和接收数据。使用 2 种电平实现 NRZ：代表 1 数据位的 VOH 标号状态（markstate），和代表 0 数据位的 VOL 空格状态（spacestate）。采用 NRZ 格式连续发送相同值的数据位时，输出电平将保持该位的电平，而不会在发送完每个位后返回中间电平值。NRZ 发送端口在标号状态空闲。每个发送的字符都包括一个起始位，后面跟有 8 个或 9 个数据位和一个或多个终止字符发送的停止位。起始位总是处于空格状态，停止位总是处于标号状态。最常用的数据格式为 8 位。每个发送位的持续时间为 1/（波特率）。片上专用 8 位/16 位波特率发生器可用于通过系统振荡器产生标准波特率频率。

USARTx 首先发送和接收 Lsb。USARTx 的发送器和接收器在功能上是相互独立的，但采用相同的数据格式和波特率。硬件不支持奇偶校验，但可以用软件实现（奇偶校验位是第 9 个数据位）。

13.1.1 USARTx 异步发送器

图 12-1 所示为 USARTx 发送器的框图。发送器的核心是串行发送移位寄存器（TSR），该寄存器不能由软件直接访问。TSR 从 TXREGx 发送缓冲寄存器获取数据。

13.1.1.1 使能发送器

通过配置如下三个控制位使能 USARTx 发送器，以用于异步操作：

- TXENx=1
- SYNCx=0
- SPENx=1
- 假设所有其他 USARTx 控制位处于其默认状态。

将 TXSTAx 寄存器的 TXENx 位置 1，使能 USARTx 发送器电路。将 TXSTAx 寄存器的 SYNCx 位清零，将 USARTx 配置用于异步操作。

注：

1) 当将 SPENx 位和 TXENx 位置 1，SYNCx 位清零，TXx/CKx 引脚被自动配置为输出引脚，无需考虑相应 TRIS 位的状态。
2) 当将 SPENx 位和 CRENx 位置 1，SYNCx 位清零，RXx/DTx 引脚被自动配置为输入引脚，无需考虑相应 TRIS 位的状态。

13.1.1.2 发送数据

向 TXREGx 寄存器写入一个字符，以启动发送。如果这是第一个字符，或者前一个字符已经完全从 TSR 中移出，TXREGx 中的数据会立即发送给 TSR 寄存器。如果 TSR 中仍保存全部或部分前一字符，新的字符数据将保存在 TXREGx 中，直到发送完前一字符的停止位为止。然后，在停止位发送完毕后经过一个 TCY，TXREGx 中待处理的数据将被传输到 TSR。当数据从 TXREGx 传输至 TSR 后，立即开始进行起始位、数据位和停止位序列的发送。

13.1.1.3 发送中断标志

只要使能 USARTx 发送器且 TXREGx 中没有待发送数据，就将 TXxIF 中断标志位置 1。换句话说，只有当 TSR 忙于处理字符和 TXREGx 中有排队等待发送的新字符时，TXxIF 位才处于清零状态。写 TXREGx 时，不立即清零 TXxIF 标志位。TXxIF 在写指令后的第 2 个指令周期清零。在写 TXREGx 后立即查询 TXxIF 会返回无效结果。TXxIF 为只读位，不能由软件置 1 或清零。

可通过将 TXxIE 中断允许位置 1 允许 TXxIF 中断。然而，只要 TXREGx 为空，不管 TXxIE 允许位的状态如何都会将 TXxIF 标志位置 1。

如果要在发送数据时使用中断，只在有待发送数据时，才将 TXxIE 位置 1。当将待发送的最后一个字符写入 TXREGx 后，将 TXxIE 中断允许位清零。

13.1.1.4 TSR 状态

TXSTAx 寄存器的 TRMTx 位指示 TSR 寄存器的状态。TRMTx 位为只读位。当 TSR 寄存器为空时，TRMTx 位被置 1，当有字符从 TXREGx 传输到 TSR 寄存器时，TRMT 被清零。TRMT 位保持清零状态，直到所有位从 TSR 寄存器移出为止。没有任何中断逻辑与该位有关，所以用户必须查询该位来确定 TSR 的状态。

> 注：TSR 寄存器并未映射到数据存储器中，因此用户不能直接访问它。

13.1.1.5 发送 9 位字符

USARTx 支持 9 位字符发送。当 TXSTAx 寄存器的 TX9ENx 位置 1 时，USARTx 将移出每个待发送字符的 9 位。TXSTAx 寄存器的 TX9Dx 位为第 9 位，即最高数据位。当发送 9 位数据时，必须在将 8 个最低位写入 TXREGx 之前，写 TX9Dx 数据位。在写入 TXREGx 寄存器后会立即将 9 个数据位传输到 TSR 移位寄存器。

13.1.1.6 设置异步发送

1. 初始化 SPBRGx 寄存器，以获得所需的波特率（请参见“USARTx 波特率发生器（BRG）”章节）。
2. 通过将 SYNCx 位清零并将 SPENx 位置 1 使能异步串口。
3. 如果需要 9 位发送，将 TX9ENx 控制位置 1。当接收器被设置为进行地址检测时，将数据位的第 9 位置 1，指示 8 个最低数据位为地址。
4. 将 TXENx 控制位置 1，使能发送；这将导致 TXxIF 中断标志位置 1。
5. 如果需要中断，将 TXxIE 中断允许位置 1；如果 INTCON 寄存器的 GIE 和 PEIE 位也置 1 将立即产生中断。
6. 若选择发送 9 位数据，第 9 位应该被装入 TX9Dx 数据位。
7. 将 8 位数据装入 TXREGx 寄存器开始发送数据。

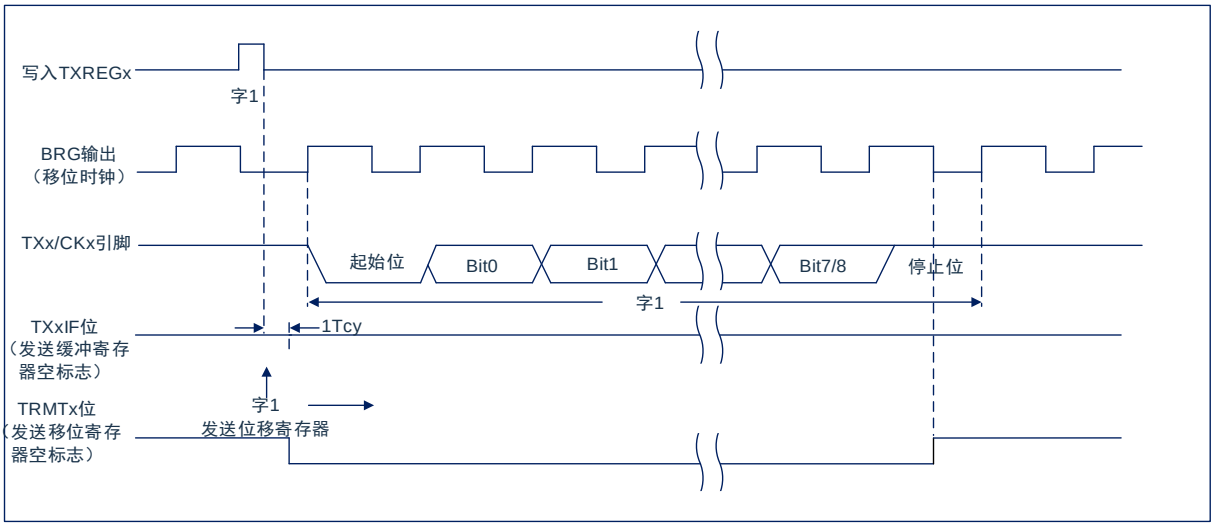


图 13-3：异步发送

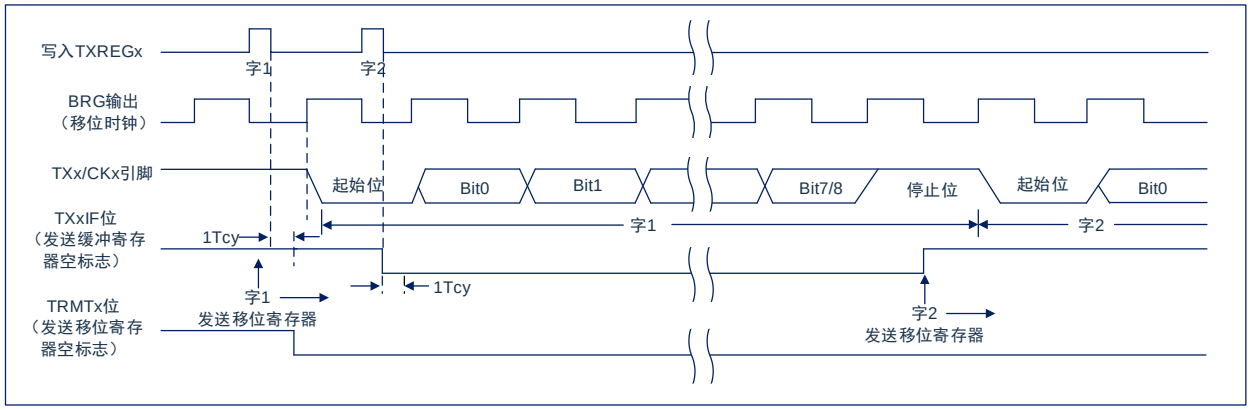


图 13-4：异步发送（背靠背）

注：本时序图显示了两次连续的发送。
在异步发送的背靠背模式下，使用 TXxIF 标志来判断是否可以往 TXREGx 写入内容时，则需要在写入 TXREGx 后，再对 TRMTx 位进行判断，如果为 1 再对 TXREGx 进行二次写入。

例：异步发送(背靠背)

```
INTERRUPT_USART_SEND:                   中断异步发送子函数
        MOV     A,#0X10
        ANL     A,PIR1
        JNZ     $+2                     判断 TX0IF 标志是否为 1，为 0 则跳出该中断函数
        RETI
        MOV     A,#0X40                 040H 为发送的数据
        MOV     TXREG0,A                将需要发送到内容写入 TXREG0
        MOV     A,#0X02
        ANL     A,TXSTA0
        JNZ     $+2                     判断 TRMT0 标志是否为 1，为 0 则跳出该中断函数
        RETI
        MOV     A,#0X40
        MOV     TXREG0,A                将 040H 重新加载到 TXREG0 中，进行二次写入
        RETI
```

13.1.2 USARTx 异步接收器

异步模式通常用于 RS-232 系统。图 13-2 给出了接收器的框图。在 RXx/DTx 引脚上接收数据和驱动数据恢复电路。数据恢复电路实际上是一个以 16 倍波特率为工作频率的高速移位器，而串行接收移位寄存器（ReceiveShiftRegister，RSR）则以比特率工作。当字符的全部 8 位或 9 位数据位被移入后，立即将它们传输到一个 2 字符的先入先出（FIFO）缓冲器。FIFO 缓冲器允许接收 2 个完整的字符和第 3 个字符的起始位，然后必须由软件将接收到的数据提供给 USARTx 接收器。FIFO 和 RSR 寄存器不能直接由软件访问。通过 RCREGx 寄存器访问接收到的数据。

13.1.2.1 使能接收器

通过配置如下三个控制位使能 USARTx 接收器，以用于异步操作。

- CRENx=1
- SYNCx=0
- SPENx=1

假设所有其他 USARTx 控制位都处于默认状态。将 RCSTAx 寄存器的 CRENx 位置 1，使能 USARTx 接收器电路。将 TXSTAx 寄存器的 SYNCx 位清零，配置 USARTx 以用于异步操作。

> 注：
> 1) 当将 SPENx 位和 TXENx 位置 1，SYNCx 位清零，TXx/CKx 引脚被自动配置为输出引脚，无需考虑相应 TRIS 位的状态。
> 2) 当将 SPENx 位和 CRENx 位置 1，SYNCx 位清零，RXx/DTx 引脚被自动配置为输入引脚，无需考虑相应 TRIS 位的状态。

13.1.2.2 接收数据

接收器数据恢复电路在第一个位的下降沿开始接收字符。第一个位，通常称为起始位，始终为 0。由数据恢复电路计数半个位时间，到起始位的中心位置，校验该位是否仍为零。如果该位不为零，数据恢复电路放弃接收该字符，而不会产生错误，并且继续查找起始位的下降沿。如果起始位零校验通过，则数据恢复电路计数一个完整的位时间，到达下一位的中心位置。由择多检测电路对该位进行采样，将相应的采样结果 0 或 1 移入 RSR。重复该过程，直到完成所有数据位的采样并将其全部移入 RSR 寄存器。测量最后一个位的时间并采样其电平。此位为停止位，总是为 1。如果数据恢复电路在停止位的位置采样到 0，则该字符的帧错误标志将置 1，反之，该字符的帧错误标志会清零。

当接收到所有数据位和停止位后，RSR 中的字符会被立即传输到 USARTx 的接收 FIFO 并将 RCxIF 中断标志位置 1。通过读 RCREGx 寄存器将 FIFO 最顶端的字符移出 FIFO。

> 注：如果接收 FIFO 溢出，则不能再继续接收其他字符，直到溢出条件被清除。

13.1.2.3 接收中断

只要使能 USARTx 接收器且在接收 FIFO 中没有未读数据， RCxIF 中断标志位就会置 0。RCxIF 中断标志位为只读，不能由软件置 1 或清零。

通过将下列所有位均置 1 来允许 RCxIF 中断：

- PIE1 或 PIE2 寄存器的 RCxIE 中断允许位；
- INTCON 寄存器的 PEIE 外设中断允许位；
- INTCON 寄存器的 GIE 全局中断允许位。

如果 FIFO 中有未读数据，无论中断允许位的状态如何，都会将 RCxIF 中断标志位置 1。

13.1.2.4 接收帧错误

接收 FIFO 缓冲器中的每个字符都有一个相应的帧错误状态位。帧错误指示未在预期的时间内接收到停止位。

由 RCSTAx 寄存器的 FERRx 位获取帧错误状态。

帧错误（FERRx=1）并不会阻止接收更多的字符。无需清零 FERRx 位。

清零 RCSTAx 寄存器的 SPENx 位会复位 USARTx，并强制清零 FERRx 位。清零 RCSTAx 寄存器的 CRENx 位会强制清零 FERRx 位。帧错误本身不会产生中断。

注：
1）要获取帧错误状态，必须在读 RCREGx 寄存器之后再读 FERRx 位；
2）如果接收 FIFO 缓冲器中所有接收到的字符都有帧错误，重复读 RCREG 不会清零 FERRx 位。

13.1.2.5 接收溢出错误

接收 FIFO 缓冲器可以保存 2 个字符。但如果在访问 FIFO 之前，接收到完整的第 3 个字符，则会产生溢出错误。此时，RCSTAx 寄存器的 OERRx 位会置 1。可以读取 FIFO 缓冲器内的字符，但是在错误清除之前，不能再接收其他字符。可以通过清零 RCSTAx 寄存器的 CRENx 位或通过清零 RCSTAx 寄存器的 SPENx 位使 USARTx 复位来清除错误。

13.1.2.6 接收 9 位字符

USARTx 支持 9 位数据接收。将 RCSTAx 寄存器的 RX9ENx 位置 1 时，USARTx 将接收到的每个字符的 9 位移入 RSR。必须在读 RCREGx 中的低 8 位之后，读取 RX9Dx 数据位。

13.1.2.7 异步接收设置

1. 初始化 SPBRGx 寄存器，以获得所需的波特率。
   （请参见“USARTx 波特率发生器（BRG）”章节）
2. 将 SPENx 位置 1，使能串行端口。必须清零 SYNCx 位以执行异步操作。
3. 如果需要中断，将 PIE1 或 PIE2 寄存器中的 RCxIE 位和 INTCON 寄存器的 GIE 和 PEIE 位置 1。
4. 如果需要接收 9 位数据，将 RX9ENx 位置 1。
5. 将 CRENx 位置 1 使能接收。
6. 当一个字符从 RSR 传输到接收缓冲器时，将 RCxIF 中断标志位置 1。如果 RCxIE 中断允许位也置 1 还将产生中断。
7. 读 RCREGx 寄存器，从接收缓冲器获取接收到的 8 个低数据位。
8. 读 RCSTAx 寄存器获取错误标志位和第 9 位数据位（如果使能 9 位数据接收）。
9. 如果发生溢出，通过清零 CRENx 接收器使能位清零 OERRx 标志。

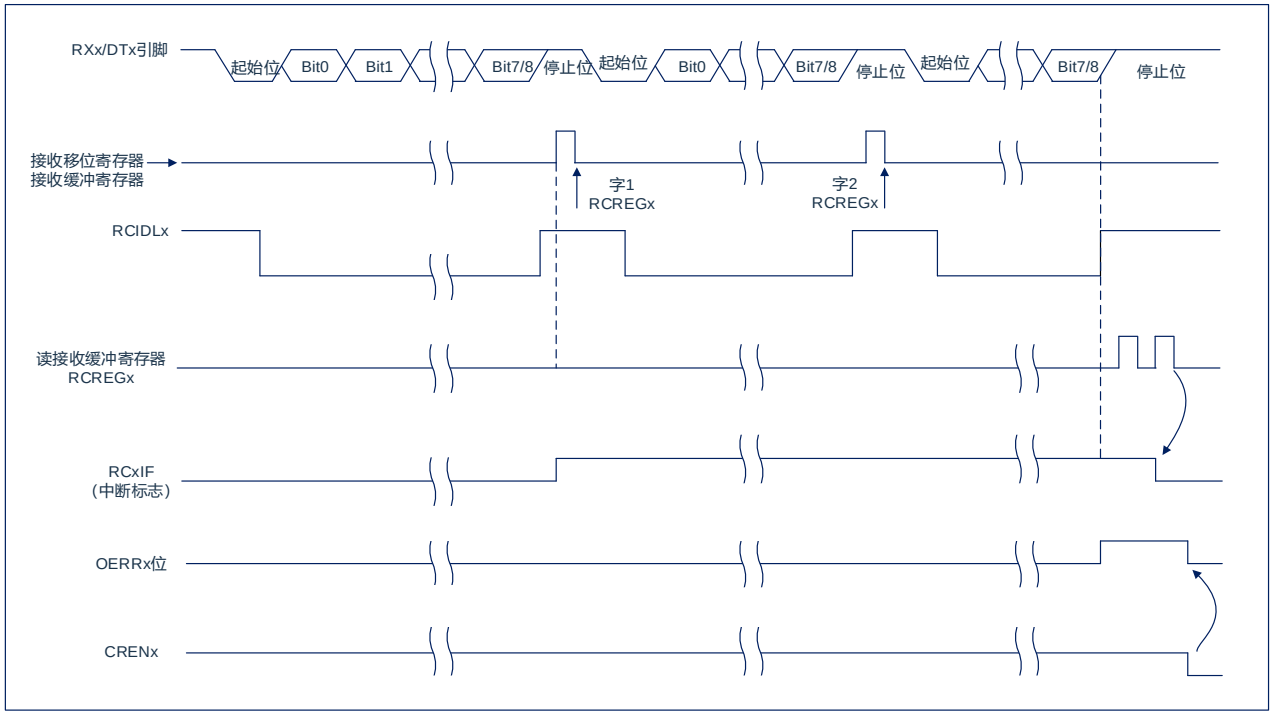


图 13-5：异步接收

注：本时序图显示出了在 RXx/DTx 输入引脚接收三个字的情况，在第 3 个字后读取 RCREGx（接收缓冲器），导致 OERRx（溢出）位置 1。

13.2 异步操作时的时钟准确度

由厂家校准内部振荡电路（INTOSC）的输出。但在 VDD 或温度变化时，INTOSC 会发生频率漂移，从而会直接影响异步波特率。

13.3 USARTx 相关寄存器

发送状态和控制寄存器 TXSTAx

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TXSTAx	CSRCx	TX9ENx	TXENx	SYNCx	SCKPx	----	TRMTx	TX9Dx
R/W	R/W	R/W	R/W	R/W	R/W	----	R	R/W
复位值	0	0	0	0	0	----	1	0

Bit7 CSRCx: 时钟源选择位
异步模式: 任意值
同步模式:
1=主控模式（由内部 BRG 产生时钟信号）
0=从动模式（由外部时钟源产生时钟）

Bit6 TX9ENx: 9 位发送使能位
1= 选择 9 位发送
0= 选择 8 位发送

Bit5 TXENx: 发送使能位(1)
1= 使能发送
0= 禁止发送

Bit4 SYNCx: USART 模式选择位
1= 同步模式
0= 异步模式

Bit3 SCKPx: 同步时钟极性选择位
异步模式:
1=将数据字符的电平取反后发送到 TXx/CKx 引脚
0=直接将数据字符发送到 TXx/CKx 引脚
同步模式:
0=在时钟上升沿传输数据
1=在时钟下降沿传输数据

Bit2 未用

Bit1 TRMTx: 发送移位寄存器状态位
1= TSR 为空
0= TSR 为满

Bit0 TX9Dx: 发送数据的第 9 位
可以是地址/数据位或奇偶校验位

注：同步模式下，SRENx/CRENx 会覆盖 TXENx 的值。

接收状态和控制寄存器 RCSTAx

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RCSTAx	SPENx	RX9ENx	SRENx	CRENx	RCIDLx	FERRx	OERRx	RX9Dx
R/W	R/W	R/W	R/W	R/W	R	R	R	R
复位值	0	0	0	0	1	0	0	0

Bit7 SPENx: 串行端口使能位
1= 使能串行端口（将 RXx/DTx 和 TXx/CKx 引脚配置为串行端口引脚）
0= 禁止串行端口（保持在复位状态）
Bit6 RX9ENx: 9 位接收使能位
1= 选择 9 位接收
0= 选择 8 位接收
Bit5 SRENx: 单字节接收使能位
异步模式: 任意值
同步主控模式:
1=使能单字节接收
0=禁止单字节接收
接收完成后清零该位
同步从动模式: 任意值
Bit4 CRENx: 连续接收使能位
异步模式:
1=使能接收
0=禁止接收
同步模式:
1=使能连续接收直到清零 CRENx 使能位（CRENx 覆盖 SRENx）
0=禁止连续接收
Bit3 RCIDLx: 接收空闲标志位
异步模式: 1=接收器空闲
0=已接收到起始位，接收器正在接收数据
同步模式: 任意值
Bit2 FERRx: 帧错误位
1= 帧错误（可通过读 RCREGx 寄存器更新并接收下一个有效字节）
0= 没有帧错误
Bit1 OERRx: 溢出错误位
1= 溢出错误（可通过清零 CRENx 位清零）
0= 没有溢出错误
Bit0 RX9Dx: 接收到数据的第 9 位
此位可以是地址/数据位或奇偶校验位，必须由用户固件计算得到

发送数据寄存器 TXREGx

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TXREGx								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

接收数据寄存器 RCREGx

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RCREGx								
R/W	R	R	R	R	R	R	R	R
复位值	0	0	0	0	0	0	0	0

波特率数据寄存器 SPBRGx

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPBRGx								
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

波特率微调寄存器 BRG_ADJx

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BRG_ADJx	EN_ADJx	---	---	ADJx <4:0>				
R/W	R/W	---	---	R/W	R/W	R/W	R/W	R/W
复位值	0	---	---	0	0	0	0	0

Bit7 EN_ADJx: 波特率微调值使能位
0: 不使能
1: 使能

Bit6~Bit5 未用

Bit4~Bit0 ADJx<4:0> 波特率微调值

注：波特率微调功能只有在异步模式下将 EN_ADJx=1 才有效，同步模式下无效。

13.4 USARTx 波特率发生器（BRG）

波特率发生器（BRG）是一个 8 位，专用于支持 USARTx 的异步和同步工作模式。

SPBRGx 寄存器和 BRG_ADJx 寄存器的低 4 位共同决定自由运行的波特率定时器的周期。

表 13-1 包含了计算波特率的公式。公式 1 为不开启波特率微调功能时的一个计算波特率和波特率误差的示例

表 13-2 和表 13-3 中给出了已经计算好的各种异步模式下的典型波特率和波特率误差值，可便于您使用。

向 SPBRGx 寄存器对写入新值会导致 BRG 定时器复位（或清零）。这可以确保 BRG 无需等待定时器溢出就可以输出新的波特率。

如果系统时钟在有效的接收过程中发生了变化，可能会产生接收错误或导致数据丢失。为了避免此问题，应该检查 RCIDLx 位的状态以确保改变系统时钟之前，接收操作处于空闲状态。

公式 1：计算波特率误差

对于 F_{HSI} 为 16MHz，目标波特率为 9600bps，异步模式采用 8 位 BRG 的器件：

$$目标波特率=\frac{F_{HSI}}{16([SPBRGx]+1)}$$

求解 SPBRGx：

$$X=\frac{\frac{F_{HSI}}{目标波特率}}{16}-1=\frac{\frac{16000000}{9600}}{16}-1=[103.16]=103$$

$$计算波特率=\frac{16000000}{16（103+1）}=9615$$

$$误差=\frac{计算波特率-目标波特率}{目标波特率}=\frac{（9615-9600）}{9600}=0.16\%$$

表 13-1：波特率公式

配置位		USARTx 模式	波特率公式
SYNCx	EN_ADJx		
0	0	异步	$F_{HSI}/[16(n+1)]$
0	1	异步	$F_{HSI}/[16(n+1)+y]$
1	x	同步	$F_{HSI}/[4(n+1)]$

说明：n= SPBRGx 寄存器的值，y=ADJx<4:0>的值。

表 13-2：异步模式下的波特率（EN_ADJx=0）

目标波特率	SYNCx=0		
	F_{HSI}=16.00MHz		
	实际波特率	误差（%）	SPBRGx 值
2400	----	----	----
9600	9615	0.16	103
10417	10417	0	95
14400	14492	0.83	68
19200	19230	0.16	51

表 13-3：异步模式下的波特率（EN_ADJx=1）

目标波特率	SYNCx=0			
	F_{HSI}=16.00MHz			
	实际波特率	误差（%）	SPBRGx 值	ADJx<4:0>值
4800	4800.48	0.01	207	5
9600	9603.84	0.04	103	2
19200	19207.68	0.04	51	1
115200	115107.91	-0.08	7	11

13.5 USARTx 同步模式

同步串行通信通常用在具有一个主控器件和一个或多个从动器件的系统中。主控器件包含产生波特率时钟所必需的电路，并为系统中的所有器件提供时钟。从动器件可以使用主控时钟，因此无需内部时钟发生电路。

在同步模式下，有 2 条信号线：双向数据线和时钟线。从动器件使用主控器件提供的外部时钟，将数据串行移入或移出相应的接收和发送移位寄存器。因为使用双向数据线，所以同步操作只能采用半双工方式。半双工是指：主控器件和从动器件都可以接收和发送数据，但是不能同时进行接收或发送。USARTx 既可以作为主控器件，也可以作为从动器件。

同步发送无需使用起始位和停止位。

13.5.1 同步主控模式

下列位用来将 USARTx 配置为同步主控操作：

- SYNCx=1
- CSRCx=1
- SRENx=0（用于发送）；SRENx=1（用于接收）
- CRENx=0（用于发送）；CRENx=1（用于接收）
- SPENx=1

将 TXSTAx 寄存器的 SYNCx 位置 1，可将 USARTx 配置用于同步操作。将 TXSTAx 寄存器的 CSRCx 位置 1，将器件配置为主控器件。将 RCSTAx 寄存器的 SRENx 和 CRENx 位清零，以确保器件处于发送模式，否则器件配置为接收模式。将 RCSTAx 寄存器的 SPENx 位置 1，使能 USARTx。

13.5.1.1 主控时钟

同步数据传输使用独立的时钟线同步传输数据。配置为主控器件的器件在 TXx/CKx 引脚发送时钟信号。当 USARTx 被配置为同步发送或接收操作时，TXx/CKx 输出驱动器自动使能。串行数据位在每个时钟的上升沿发生改变，以确保它们在下降沿有效。每个数据位的时间为一个时钟周期，有多少数据位就只能产生多少个时钟周期。

13.5.1.2 时钟极性

器件提供时钟极性选项以与 Microwire 兼容。由 TXSTAx 寄存器的 SCKPx 位选择时钟极性。将 SCKPx 位置 1 将时钟空闲状态设置为高电平。当 SCKPx 位置 1 时，数据在每个时钟的下降沿发生改变。清零 SCKPx 位，将时钟空闲状态设置为低电平。当清零 SCKPx 位时，数据在每个时钟的上升沿发生改变。

13.5.1.3 同步主控发送

由器件的 RXx/DTx 引脚输出数据。当 USARTx 配置为同步主控发送操作时，器件的 RXx/DTx 和 TXx/CKx 输出引脚自动使能。

向 TXREGx 寄存器写入一个字符开始发送。如果 TSR 中仍保存全部或部分前一字符，新的字符数据保存在 TXREGx 中，直到发送完前一字符的停止位为止。如果这是第一个字符，或者前一个字符已经完全从 TSR 中移出，则 TXREGx 中的数据会被立即传输到 TSR 寄存器。当字符从 TXREGx 传输到 TSR 后会立即开始发送数据。每个数据位在主控时钟的上升沿发生改变，并保持有效，直至下一个时钟的上升沿为止。

注：TSR 寄存器并未映射到数据存储器中，因此用户不能直接访问它。

13.5.1.4 同步主控发送设置

1. 初始化 SPBRGx 寄存器，以获得所需的波特率。
   （请参见“USARTx 波特率发生器（BRG）”章节）
2. 将 SYNCx、SPENx 和 CSRCx 位置 1，使能同步主控串行端口。
3. 将 SRENx 和 CRENx 位清零，禁止接收模式。
4. 将 TXENx 位置 1 使能发送模式。
5. 如果需要发送 9 位字符，将 TX9ENx 置 1。
6. 若需要中断，将 PIE1 或 PIE2 寄存器中的 TXxIE 位，以及 INTCON 寄存器中的 GIE 和 PEIE 位置 1。
7. 如果选择发送 9 位字符，应该将第 9 位数据装入 TX9Dx 位。
8. 通过将数据装入 TXREGx 寄存器启动发送。

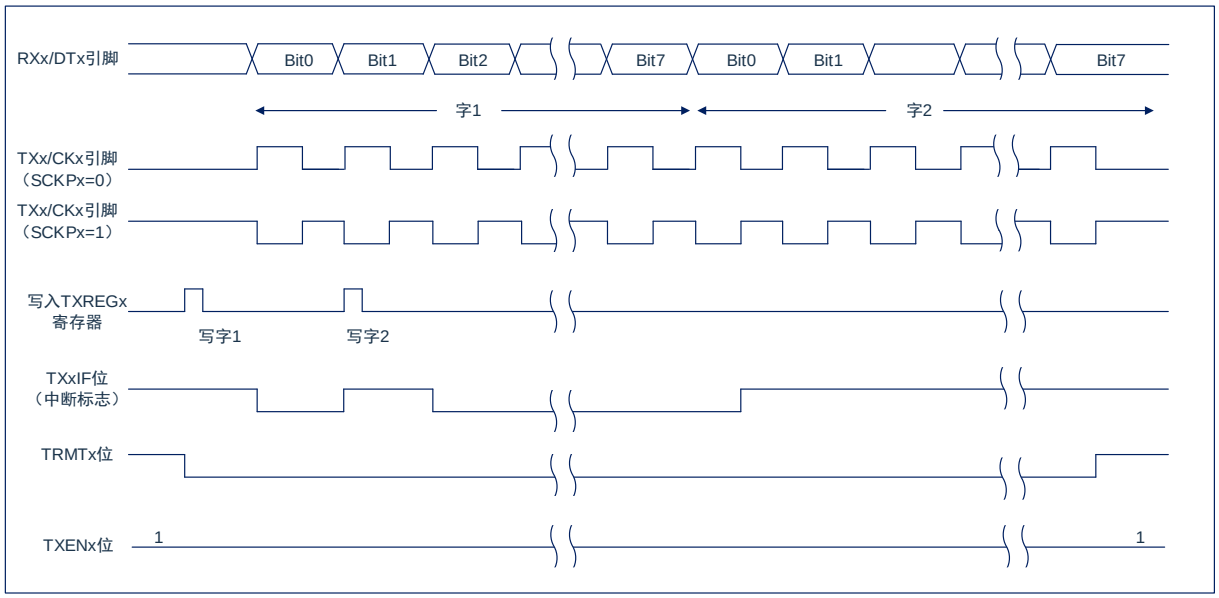


图 13-6：同步发送

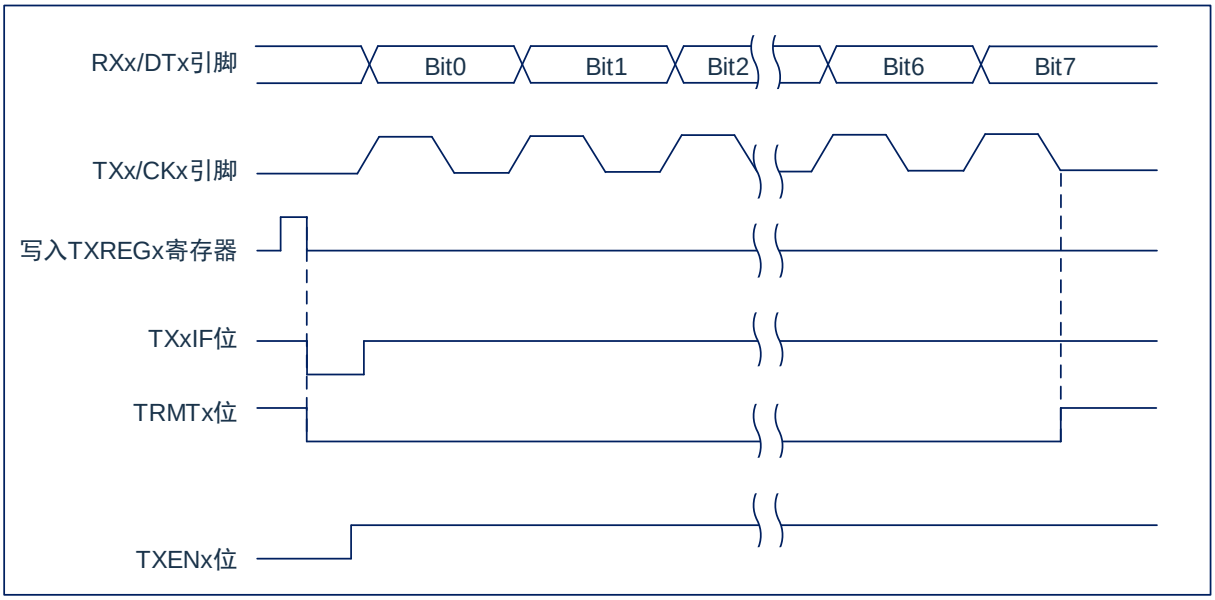


图 13-7：同步发送（通过 TXENx）

13.5.1.5 同步主控接收

在 RXx/DTx 引脚接收数据。当 USARTx 配置为同步主控接收时，自动禁止器件的 RXx/DTx 引脚的输出驱动器。

在同步模式下，将单字接收使能位（RCSTAx 寄存器的 SRENx 位）或连续接收使能位（RCSTAx 寄存器的 CRENx 位）置 1 使能接收。当将 SRENx 置 1，CRENx 位清零时，一个单字符中有多少数据位就只能产生多少时钟周期。一个字符传输结束后，自动清零 SRENx 位。当 CRENx 置 1 时，将产生连续时钟，直到清零 CRENx 为止。如果 CRENx 在一个字符的传输过程中清零，则 CKx 时钟立即停止，并丢弃该不完整的字符。如果 SRENx 和 CRENx 都置 1，则当第一个字符传输完成时，SRENx 位被清零，CRENx 优先。

将 SRENx 或 CRENx 位置 1，启动接收。在 TXx/CKx 时钟引脚信号的下降沿采样 RXx/DTx 引脚上的数据，并将采样到的数据移入接收移位寄存器（RSR）。当 RSR 接收到一个完整字符时，将 RCxIF 位置 1，字符自动移入 2 字节接收 FIFO。接收 FIFO 中最顶端字符的低 8 位可通过 RCREGx 读取。只要接收 FIFO 中仍有未读字符，则 RCxIF 位就保持置 1 状态。

13.5.1.6 从时钟

同步数据传输使用与数据线通读的独立时钟线。配置为从器件的器件接收 TXx/CKx 线上的时钟信号。当器件被配置为同步从发送或接收操作时，TXx/CKx 引脚的输出驱动器自动被禁止。串行数据位在时钟信号的前沿改变，以确保其在每个时钟的后沿有效。每个时钟周期只能传输一位数据，因此有多少数据位要传输就必须接收多少个时钟。

13.5.1.7 接收溢出错误

接收 FIFO 缓冲器可以保存 2 个字符。在读 RCREGx 以访问 FIFO 之前，若完整地接收到第 3 个字符，则产生溢出错误。此时，RCSTAx 寄存器的 OERRx 位会置 1。FIFO 中先前的数据不会被改写。可以读取 FIFO 缓冲器内的 2 个字符，但是在错误被清除前，不能再接收其他字符。只能通过清除溢出条件，将 OERRx 位清零。如果发生溢出时，SRENx 位为置 1 状态，CRENx 位为清零状态，则通过读 RCREGx 寄存器清除错误。如果溢出时，CRENx 为置 1 状态，则可以清零 RCSTAx 寄存器的 CRENx 位或清零 SPENx 位以复位 USARTx，从而清除错误。

13.5.1.8 接收 9 位字符

USARTx 支持接收 9 位字符。当 RCSTAx 寄存器的 RX9ENx 位置 1 时，USARTx 将接收到的每个字符的 9 位数据移入 RSR。当从接收 FIFO 缓冲器读取 9 位数据时，必须在读 RCREGx 的 8 个低位之后，读取 RX9Dx 数据位。

13.5.1.9 同步主控接收设置

1. 初始化 SPBRGx 寄存器，以获得所需的波特率。（注：必须满足 SPBRGx>05H）
2. 将 SYNCx、SPENx 和 CSRCx 位置 1 使能同步主控串行端口。
3. 确保将 CRENx 和 SRENx 位清零。
4. 如果使用中断，将 INTCON 寄存器的 GIE 和 PEIE 位置 1，并将 PIE1 或 PIE2 寄存器的 RCxIE 位也置 1。
5. 如果需要接收 9 位字符，将 RX9ENx 位置 1。
6. 将 SRENx 位置 1，启动接收，或将 CRENx 位置 1 使能连续接收。
7. 当字符接收完毕后，将 RCxIF 中断标志位置 1。如果允许位 RCxIE 置 1，还会产生一个中断。

8. 读 RCREGx 寄存器获取接收到的 8 位数据。
9. 读 RCSTAx 寄存器以获取第 9 个数据位（使能 9 位接收时），并判断接收过程中是否产生错误。
10. 如果产生溢出错误，清零 RCSTAx 寄存器的 CRENx 位或清零 SPENx 以复位 USARTx 来清除错误。

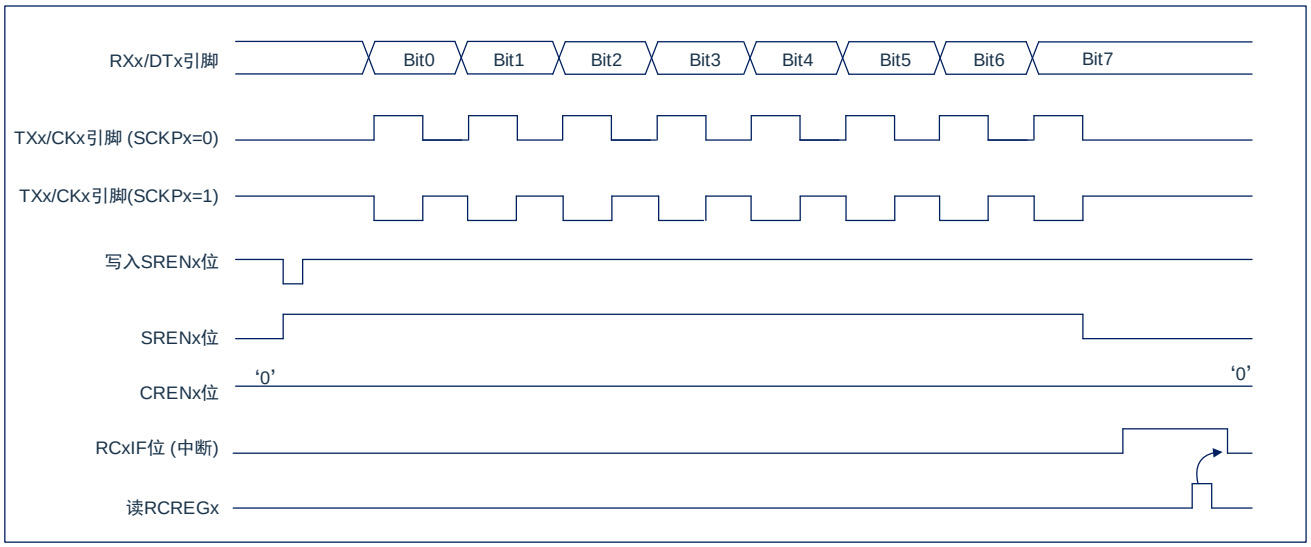


图 13-8：同步接收（主控模式，SRENx）

注：时序图说明了 SRENx=1 时的同步主控模式。

13.5.2 同步从动模式

下列位用来将 USARTx 配置为同步从动操作：

- SYNCx=1
- CSRCx=0
- SRENx=0（用于发送）；SRENx=1（用于接收）
- CRENx=0（用于发送）；CRENx=1（用于接收）
- SPENx=1

将 TXSTAx 寄存器的 SYNCx 位置 1，可将器件配置用于同步操作。将 TXSTAx 寄存器的 CSRCx 位置 0，将器件配置为从动器件。将 RCSTAx 寄存器的 SRENx 和 CRENx 位清零，以确保器件处于发送模式，否则器件将被配置为接收模式。将 RCSTAx 寄存器的 SPENx 位置 1，使能 USARTx。

13.5.2.1 USARTx 同步从动发送

同步主控和从动模式的工作原理是相同的（见章节“同步主控发送”章节。）

13.5.2.2 同步从动发送设置

1. 将 SYNCx 和 SPENx 位置 1 并将 CSRCx 位清零。
2. 将 CRENx 和 SRENx 位清零。
3. 如果使用中断，将 INTCON 寄存器的 GIE 和 PEIE 位置 1，并将 PIE1 或 PIE2 寄存器的 TXxIE 位也置 1。
4. 如果需要发送 9 位数据，将 TX9ENx 位置 1。
5. 将 TXENx 位置 1 使能发送。
6. 若选择发送 9 位数据，将最高位写入 TX9Dx 位。
7. 将低 8 位数据写入 TXREGx 寄存器开始传输。

13.5.2.3 USART 同步从动接收

除了以下不同外，同步主控和从动模式的工作原理相同。

1. CRENx 位总是置 1，因此接收器不能进入空闲状态。
2. SRENx 位，在从动模式可为“任意值”。

13.5.2.4 同步从动接收设置

1. 将 SYNCx 和 SPENx 位置 1 并将 CSRCx 位清零。
2. 如果使用中断，将 INTCON 寄存器的 GIE 和 PEIE 位置 1，并将 PIE1 或 PIE2 寄存器的 RCxIE 位也置 1。
3. 如果需要接收 9 位字符，将 RX9ENx 位置 1。
4. 将 CRENx 位置 1，使能接收。
5. 当接收完成后，将 RCxIF 位置 1。如果 RCxIE 已置 1，还会产生一个中断。
6. 读 RCREGx 寄存器，从接收 FIFO 缓冲器获取接收到的 8 个低数据位。
7. 如果使能 9 位模式，从 RCSTAx 寄存器的 RX9Dx 位获取最高位。
8. 如果产生溢出错误，清零 RCSTAx 寄存器的 CRENx 位或清零 SPENx 位以复位 USARTx 来清除错误。

14. 捕捉模块（CPT）

捕捉模块是允许用户定时和控制不同事件的外设。在捕捉模式下，该外设能对事件的持续时间计时。在捕捉模式下，需要用到定时器 TIMER1。

CPT 控制寄存器 CPTCON

A2H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CPTCON	---	---	CPTIO_SEL[1:0]		CPTM3	CPTM2	CPTM1	CPTM0
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	---	0	0	0	0	0	0

Bit7~Bit6 未用

Bit5~Bit4 CPTIO_SEL[1:0]: 捕捉模块输入引脚控制位

00= CPT1作为捕捉输入引脚

01= CPT2作为捕捉输入引脚

10= CPT3作为捕捉输入引脚

11= CPT4作为捕捉输入引脚

Bit3~Bit0 CPTM<3:0>: CPT模式选择位

0000= 捕捉模块关闭（复位CPT模块）

0001= 未使用（保留）

0010= 未使用（保留）

0011= 未使用（保留）

0100= 捕捉模式，在每个下降沿发生捕捉

0101= 捕捉模式，在每个上升沿发生捕捉

0110= 捕捉模式，每4个上升沿发生捕捉

0111= 捕捉模式，每16个上升沿发生捕捉

1xxx= 未使用（保留）

14.1 捕捉模式

在捕捉模式下，当对应的 CPTx 引脚发生事件时，CPTRH:CPTRL 这对寄存器捕捉 TMR1 寄存器的 16 位值。触发捕捉的事件可被定义为以下四者之一，并且由 CPTCON 寄存器中的 CPTM<3:0>位配置：

- 每个下降沿；
- 每个上升沿；
- 每 4 个上升沿；
- 每 16 个上升沿。

通过模式选择位 CPTM3:CPTM0（CPTCON<3:0>）选择事件类型。当一个捕捉发生时，中断请求标志位 PIR1 寄存器中的 CPTIF 置 1；它必须用软件清零。如果在 CPTRH 和 CPTRL 这对寄存器中的值被读取之前发生另一次捕捉，那么之前捕捉的值将被新捕捉的值覆盖（见图 14-1）。

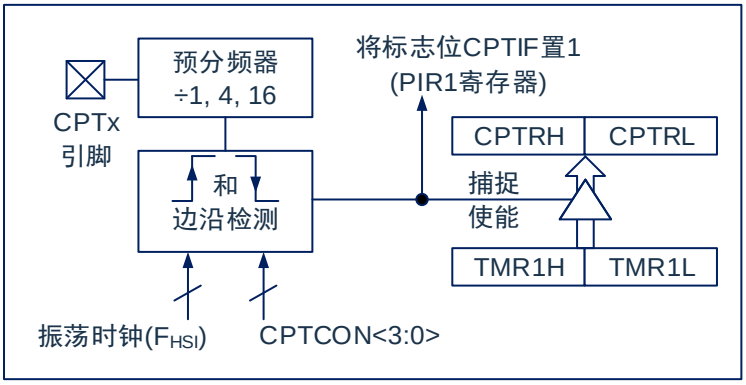


图14-1：捕捉模式工作框图

14.2 CPT 引脚配置

在捕捉模式下，应通过将对应的 TRIS 控制位置 0 来将相应的 CPTx 引脚配置为输入。

注：如果 CPTx 引脚被配置为输出，对该端口的写操作可能引发一个捕捉事件。

14.3 TIMER1 模式选择

TIMER1 必须运行在定时器模式或同步计数器模式下 CPT 模块才能使用捕捉功能。在异步计数器模式下无法进行捕捉操作。

14.4 软件中断

当捕捉模式改变时，可能会产生错误的捕捉中断。用户应该保持 PIE1 寄存器中的 CPTIE 中断允许位清零以避免产生误中断。在操作模式发生任何改变之后也应清零 PIR1 寄存器中的中断标志位 CPTIF。

14.5 CPT 预分频器

CPTCON 寄存器中的 CPTM<3:0>位指定了 4 种预分频器设置，每当关闭 CPT 模块或禁止捕捉模式时，就会清零预分频器计数器。这意味着任何复位都将清零预分频计数器。

从一种捕捉预分频比切换到另一种捕捉预分频比不会将预分频计数器清零，但可能会产生误中断。要避免出现这种不期望的操作，应在改变预分频比前通过将 CPTCON 寄存器清零关闭该模块。

15. PWM 模块

芯片包含一个 10 位 PWM 模块，可配置为 4 路共用周期、独立占空比的输出+1 路独立输出，或 2 组互补输出+1 路独立输出。

可通过寄存器选择 PWM 输出为 RB0-RB4 或 RC3-RC7。其中，PWM0/PWM1，PWM2/PWM3 可配置成带互补的正反向输出。

15.1 引脚配置

应通过将对应的 TRIS 控制位置 0 来将相应的 PWM 引脚配置为输出。

15.2 相关寄存器说明

PWM 控制寄存器 PWMCON0

F007H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON0	CLKDIV[2:0]			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit5 CLKDIV[2:0]: PWM时钟分频
111= F_{HSI} /128
110= F_{HSI} /64
101= F_{HSI} /32
100= F_{HSI} /16
011= F_{HSI} /8
010= F_{HSI} /4
001= F_{HSI} /2
000= F_{HSI} /1

Bit4~Bit0 PWMxEN: PWMx使能位
1= 使能PWMx
0= 禁止PWMx

PWM 控制寄存器 PWMCON1

F008H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON1	---	PWMIO_SEL	PWM2DTEN	PWM0DTEN	---	---	DT_DIV[1:0]	
R/W	---	R/W	R/W	R/W	---	---	R/W	R/W
复位值	---	0	0	0	---	---	0	0

Bit7 未用
Bit6 PWMIO_SEL: PWM IO选择
1= PWM分配在B组，PWM0-RC3,PWM1-RC4,PWM2-RC5,PWM3-RC6,PWM4-RC7
0= PWM分配在A组，PWM0-RB0,PWM1-RB1,PWM2-RB2,PWM3-RB3,PWM4-RB4
Bit5 PWM2DTEN: PWM2死区使能位
1= 使能PWM2死区功能，PWM2和PWM3组成一对互补输出
0= 禁止PWM2死区功能
Bit4 PWM0DTEN: PWM0死区使能位
1= 使能PWM0死区功能，PWM0和PWM1组成一对互补输出
0= 禁止PWM0死区功能
Bit3~Bit2 未用
Bit1~Bit0 DT_DIV[1:0] 死区时钟源分频
11= F_{HSI} /8
10= F_{HSI} /4
01= F_{HSI} /2
00= F_{HSI} /1

PWM 控制寄存器 PWMCON2

F009H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON2	---	---	---	PWM4DIR	PWM3DIR	PWM2DIR	PWM1DIR	PWM0DIR
R/W	---	---	---	R/W	R/W	R/W	R/W	R/W
复位值	---	---	---	0	0	0	0	0

Bit7~Bit5 未用
Bit4~Bit0 PWMxDIR PWM输出取反控制位
1= PWMx取反输出
0= PWMx正常输出

PWM4 周期低位寄存器 PWM4TL

F00AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM4TL	PWM4T[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 PWM4T[7:0]: PWM4周期低8位

PWM0~PWM3 周期低位寄存器 PWMTL

F00BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTL	PWMT[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMT[7:0]: PWM0~PWM3周期低8位

周期高位寄存器 PWMTH

F00CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTH	---	---	PWMD4[9:8]		PWM4T[9:8]		PWMT[9:8]	
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	---	0	0	0	0	0	0

Bit7~Bit6 未用
Bit5~Bit4 PWMD4[9:8]: PWM4占空比高2位
Bit3~Bit2 PWM4T[9:8]: PWM4周期高2位
Bit1~Bit0 PWMT[9:8]: PWM0~PWM3周期高2位

注：写入 PWMD4[9:8]并不能立即生效，需有写入 PWMD4L 操作后才能生效。

PWM0 占空比低位寄存器 PWMD0L

F000H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD0L	PWMD0[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD0[7:0]: PWM0占空比低8位

PWM1 占空比低位寄存器 PWMD1L

F001H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD1L	PWMD1[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD1[7:0]: PWM1占空比低8位

PWM2 占空比低位寄存器 PWMD2L

F002H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD2L	PWMD2[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD2[7:0]: PWM2占空比低8位

PWM3 占空比低位寄存器 PWMD3L

F003H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD3L	PWMD3[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD3[7:0]: PWM3占空比低8位

PWM4 占空比低位寄存器 PWMD4L

F004H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD4L	PWMD4[7:0]							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD4[7:0]: PWM4占空比低8位

PWM0 和 PWM1 占空比高位寄存器 PWMD01H

F005H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD01H	---	---	PWMD1[9:8]		---	---	PWMD0[9:8]	
R/W	---	---	R/W	---	---	----	R/W	R/W
复位值	---	---	0	---	---	----	0	0

Bit7~Bit6 未用
Bit5~Bit4 PWMD1[9:8]: PWM1占空比高2位
Bit3~Bit2 未用
Bit1~Bit0 PWMD0[9:8]: PWM0占空比高2位

注：写入 PWMD1[9:8]并不能立即生效，需有写入 PWMD1L 操作后才能生效。
写入 PWMD0[9:8]并不能立即生效，需有写入 PWMD0L 操作后才能生效。

PWM2 和 PWM3 占空比高位寄存器 PWMD23H

F006H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD23H	---	---	PWMD3[9:8]		---	---	PWMD2[9:8]	
R/W	---	---	R/W	---	---	----	R/W	R/W
复位值	---	---	0	---	---	----	0	0

Bit7~Bit6 未用
Bit5~Bit4 PWMD3[9:8]: PWM3占空比高2位
Bit3~Bit2 未用
Bit1~Bit0 PWMD2[9:8]: PWM2占空比高2位

注：写入 PWMD3[9:8]并不能立即生效，需有写入 PWMD3L 操作后才能生效。写入 PWMD2[9:8]并不能立即生效，需有写入 PWMD2L 操作后才能生效。

PWM0 和 PWM1 死区时间寄存器 PWM01DT

F00DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM01DT	---	---	PWM01DT[5:0]					
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	---	0	0	0	0	0	0

Bit7~Bit6 未用
Bit5~Bit0 PWM01DT[5:0]: PWM0和PWM1死区时间

PWM2 和 PWM3 死区时间寄存器 PWM23DT

F00EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM23DT	---	---	PWM23DT[5:0]					
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	---	0	0	0	0	0	0

Bit7~Bit6 未用
Bit5~Bit0 PWM23DT[5:0]: PWM2和PWM3死区时间

15.3 PWM 寄存器写操作顺序

由于 10 位 PWM 占空比数值分配在两个寄存器中，在修改占空比时，程序总是先后修改这两个寄存器，为了保证占空比数值的正确性，芯片内部设计了缓存加载功能。操作 10 位占空比数值需严格按照以下顺序进行：

1）写高 2 位数值，此时高 2 位数值只是写入内部的缓存；

2）写低 8 位数值，此时完整的 10 位占空比数值被锁存。

15.4 PWM 周期

PWM 周期是通过写 PWMTH 和 PWMTL 寄存器来指定的。

公式 1：PWM 周期计算公式：

$$\text{PWM 周期}=[PWMT+1]*T_{HSI}*(CLKDIV\ \text{分频值})$$

注：$T_{HSI}=1/F_{HSI}$

当 PWM 周期计数器等于 PWMT 时，在下一个递增计数周期中会发生以下 5 个事件：

- PWM 周期计数器被清零；
- PWMx 引脚被置 1；
- PWM 新周期值被锁存；
- PWM 新占空比值被锁存；
- 产生 PWM 中断标志位；

15.5 PWM 占空比

可通过将一个 10 位值写入以下多个寄存器来指定 PWM 占空比：PWMDxL、PWMDxxH。

可以在任何时候写入 PWMDxL 和 PWMDxxH 寄存器，但直到 PWM 周期计数器等于 PWMT（即周期结束）时，占空比的值才被更新到内部锁存器中。

公式 2：脉冲宽度计算公式：

$$\text{脉冲宽度} = (PWMDx[9:0]+1)*T_{HSI}*(CLKDIV\ \text{分频值})$$

公式3：PWM占空比计算公式：

$$\text{占空比}=\frac{PWMDx[9:0]+1}{PWMT[9:0]+1}$$

PWM 周期和 PWM 占空比在芯片内部都有双重缓冲。这种双重缓冲结构极其重要，可以避免在 PWM 操作过程中产生毛刺。

15.6 系统时钟频率的改变

PWM 频率与芯片系统时钟频率有关，系统时钟频率发生任何改变会影响 PWM 频率。

15.7 可编程的死区延时模式

可以通过设置 PWMxDT_EN 使能互补输出模式，使能互补输出后自动使能死区延时功能。

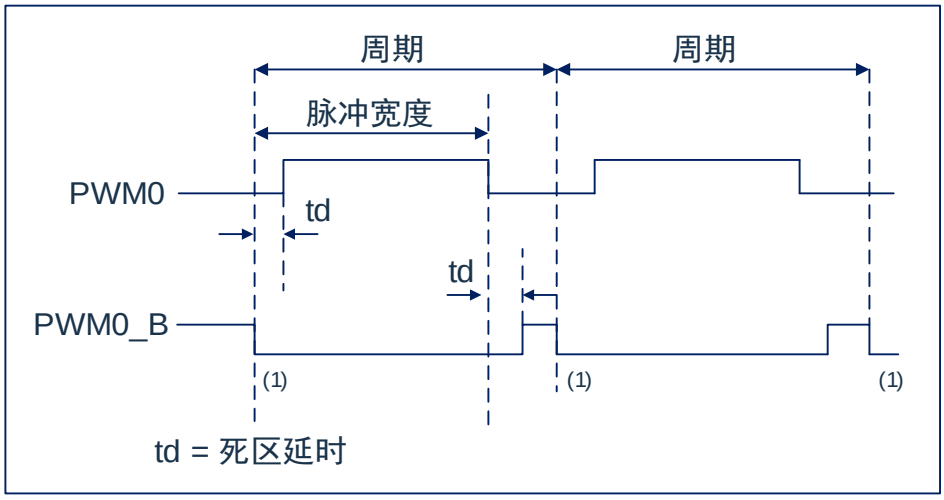


图15-1：PWM死区延时输出示例

死区时间计算公式为：

$$td=(PWMxxDT[5:0]+1)*T_{HSI}*(DT\_DIV\ 分频值)$$

15.8 PWM 设置

使用 PWM 模块时应该执行以下步骤：

1. 设置 PWMIO_SEL 控制位，选择 PWM 输出 IO 口；
2. 通过将相应的 TRIS 位置 1，使之成为输入引脚；
3. 通过装载 PWMTH，PWMTL 寄存器设置 PWM 周期；
4. 通过装载 PWMDxxH，PWMDxL 寄存器设置 PWM 占空比；
5. 若需要使用互补输出模式，需设置 PWMCON1[6:5]位，并装载 PWMxxDT 寄存器设置死区时间；
6. 清零 PWMIF 标志位；
7. 设置 PWMCN0[4:0]位以使能相应 PWM 输出；
8. 在新的 PWM 周期开始后，使能 PWM 输出：
   - 等待 PWMIF 位置 1；
   - 通过将相应的 TRIS 位清 0，使能 PWM 引脚输出驱动器。

16. 主控同步串行端口（MSSP）模块

16.1 主控 SSP（MSSP）模块概述

主控同步串行端口（Master Synchronous Serial Port，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、显示驱动器或 A/D 转换器等。

MSSP 模块有下列两种工作模式：

- 串行外设接口（SPI）。
- I^2C。
  - 全主控模式。
  - 从动模式（支持广播地址呼叫）。

I^2C 接口在硬件上支持下列模式：

- 主控模式。
- 多主机模式。
- 从动模式。

16.2 SPI 模式

SPI 模式允许同时同步发送和接收 8 位数据。SPI 支持 3 线模式和 4 线模式通信。

3 线模块下使用以下三个引脚：

- 串行数据输入（SDIO）——RC7/SDIO
- 串行时钟（SCK）——RC6/SCK
- 从动选择（SS）——RC4/SS

4 线模块下使用以下三个引脚：

- 串行数据输出（SDO）——RC5/SDO
- 串行数据输入（SDI）——RC7/SDI
- 串行时钟（SCK）——RC6/SCK
- 从动选择（SS）——RC4/SS

16.2.1 SPI 相关寄存器

SSPSTAT：SSP 状态寄存器

C7H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPSTAT	---	CKE	MODE	---	---	---	---	---
R/W	---	R/W	R/W	---	---	---	---	---
复位值	---	0	0	---	---	---	---	---

Bit7 保留 需写0

Bit 6 CKE: SPI时钟边沿选择位。

CKP= 0

0= 在SCK引脚的上升沿发送数据；

1= 在SCK引脚的下降沿发送数据。

CKP = 1

0= 在SCK引脚的下降沿发送数据；

1= 在SCK引脚的上升沿发送数据。

Bit5 MODE: 模式选择

1=3线模式 (当需要发送时，SDIO口TRIS位需清0；当需要接收时，SDIO口TRIS需置1)

0=4线模式

Bit4~Bit0 SPI模式下未用。

SSPCON：SSP 控制寄存器

C4H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7 WCOL: 写冲突检测位
1= 在发送/接收数据过程中，试图对SSPBUF寄存器进行写操作
0= 未发生冲突

Bit6 SSPOV: 接收溢出指示位
1= SSPBUF仍保持前一数据时，又收到一个新的字节。出现溢出时，SSPSR中的数据会丢失。溢出只会在从动模式下发生。在从动模式中，即使仅发送数据，用户也必须读SSPBUF以避免溢出。在主控模式中，溢出位不被置1，因为每次接收或发送新数据，都要通过写SSPBUF寄存器来启动（该位必须由软件清零）
0= 没有溢出

Bit5 SSPEN: 同步串行端口使能位
1= 使能串行端口并将SCK、SDO、SDI和SS配置为串行端口引脚
0= 禁止串行端口并将这些引脚配置为I/O端口引脚

Bit4 CKP: 时钟极性选择位
1= 时钟空闲状态为高电平
0= 时钟空闲状态为低电平

Bit3~Bit0 SSPM<3:0>: 同步串行端口模式选择位
0000= SPI主控模式，时钟= $F_{SYS}/8$
0001= SPI主控模式，时钟= $F_{SYS}/16$
0010= SPI主控模式，时钟= $F_{SYS}/64$
0011= SPI主控模式，时钟= TMR2输出/2
0100= SPI从动模式，时钟= SCK引脚，使能SS引脚控制
0101= SPI从动模式，时钟= SCK引脚，禁止SS引脚控制，SS可用作I/O引脚
0110= 保留
0111= 保留
1000= I^2C主控模式，时钟= $F_{SYS}/(4 * (SSPADD+1))$
1001= 禁止装载功能
1010= 保留
1011= 保留
1100= 保留
1101= 保留
1110= I^2C从动模式，7位地址，并允许起始位和停止位中断
1111= 保留

16.2.2 SPI 工作原理

当初始化 SPI 时，需要指定几个选项。可以通过对相应的控制位（SSPCON<5:0>和 SSPSTAT<7:6>）编程来指定。这些控制位用于指定以下选项：

- 主控模式（SCK 作为时钟输出）
- 时钟极性（SCK 的空闲状态）
- 时钟速率（仅限主控模式）
- 从动选择模式（仅限于从动模式）
- 从动模式（SCK 作为时钟输入）
- 输入数据的采样相位（数据输出时间的中间或末尾）
- 时钟边沿（在 SCK 的上升沿/下降沿输出数据）

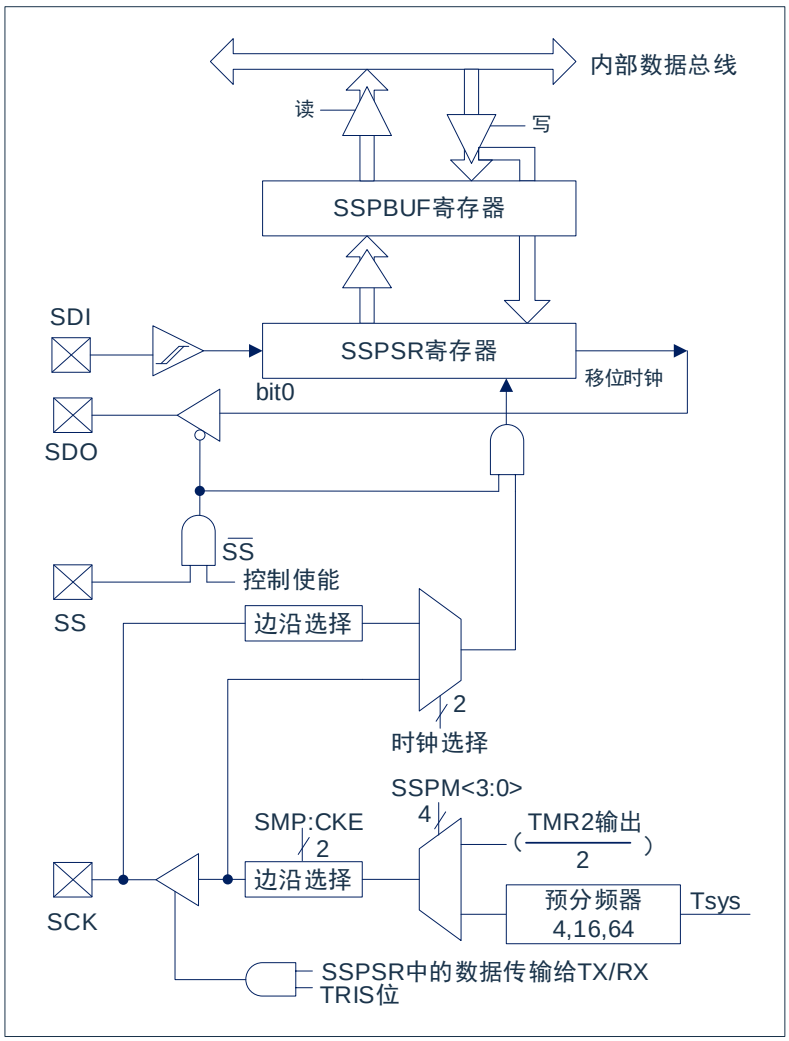


图 16-1：在 SPI 模式下 MSSP 模块的框图

注：I/O 引脚具有对 VDD 和 VSS 的二极管保护。

MSSP 模块由一个发送/ 接收移位寄存器（SSPSR）和一个缓冲寄存器（SSPBUF）组成。SSPSR 将数据移入和移出器件，最高有效位在前。SSPBUF 保存上次写入 SSPSR 的数据直到新接收到的数据就绪为止。一旦 8 位数据接收完毕，该字节就被移入 SSPBUF 寄存器。然后，PIR1 寄存器的中断标志位 SSPIF 被置 1。这种双重缓冲数据接收方式（SSPBUF）允许在读取刚接收的数据之前，就开始接收下一个字节。在数据发送/ 接收期间，任何试图写 SSPBUF 寄存器的操作都会被忽略，并将 SPCON 寄存器的写冲突检测位 WCOL

置 1。此时用户必须用软件将 WCOL 位清零，否则无法判别下一次对 SSPBUF 的写操作是否成功完成。

当应用软件等待接收有效数据时，应在下一个要传输的数据字节写入 SSPBUF 之前，将 SSPBUF 中的前一个数据读出。缓冲器满标志位 BF（SSPSTAT 寄存器）用于表示何时 SSPBUF 已经载入了接收到的数据（发送完成）。如果 SPI 仅仅作为发送器，则不必理会接收的数据。通常可用 MSSP 中断来判断发送或接收何时完成。如果不使用中断来处理数据的收发，用软件查询方法同样可确保不会发生写冲突。

16.2.3 使能 SPI I/O

要使能串行端口，SSPCON 寄存器的 MSSP 使能位 SSPEN 必须置 1。要复位或重新配置 SPI 模式，要先将 SSPEN 位清零，重新初始化 SSPCON 寄存器，然后将 SSPEN 位置 1。这将把 SDI、SDO、SCK 和 SS 引脚配置为串行端口引脚。要将这些引脚用作串行端口，还必须将其数据方向位（在 TRIS 寄存器中）正确编程，方法如下：

- SDI 由 SPI 模块自动控制；
- 必须将 SDO 的 TRISC<5>清零；
- 必须将 SCK（主控模式）的 TRISC<6>位清零；
- 必须将 SCK（从动模式）的 TRISC<6>位置 1；
- 必须将 SS（从动模式）的 TRISC<4>置 1。

对于任何不想要的串行端口功能，可通过将对应的数据方向（TRIS）寄存器设置为相反值来跳过。

16.2.4 主控模式

主器件控制 SCK，因此可以随时启动数据传输。主器件根据软件协议确定从器件应在何时广播数据。

在主控模式下，数据一旦写入 SSPBUF 寄存器就开始发送或接收。如果 SPI 仅作为接收器，则可以禁止 SDO 输出（将其编程设定为输入）。SSPSR 寄存器按设置的时钟速率对 SDI 引脚上的信号进行连续移位输入。每个字节接收完后，都会被当作普通的接收字节装入 SSPBUF 寄存器（相应的中断和状态位置 1）。这可以在接收器应用中作为“线路活动监控（Line Activity Monitor）”模式，是很有用的。

可通过对 SSPCON 寄存器的 CKP 位进行相应的编程来选择时钟极性。图 16-2、图 16-3、图 16-4 和图 15-5 给出了 SPI 通信的波形图，其中 MSb 被首先发送。在主控模式下，SPI 时钟速率（比特率）可由用户编程设定为下列速率之一：

- F_{SYS}/4（或 TCY）
- F_{SYS}/16（或 4.TCY）
- F_{SYS}/64（或 16.TCY）
- TIMER2 输出/2

图 16-2 为主控模式的波形图。当 SSPSTAT 寄存器的 CKE 位置 1 时，SDO 数据在 SCK 上出现时钟边沿前就有效。图中指出了将接收到的数据装入 SSPBUF 的时间。

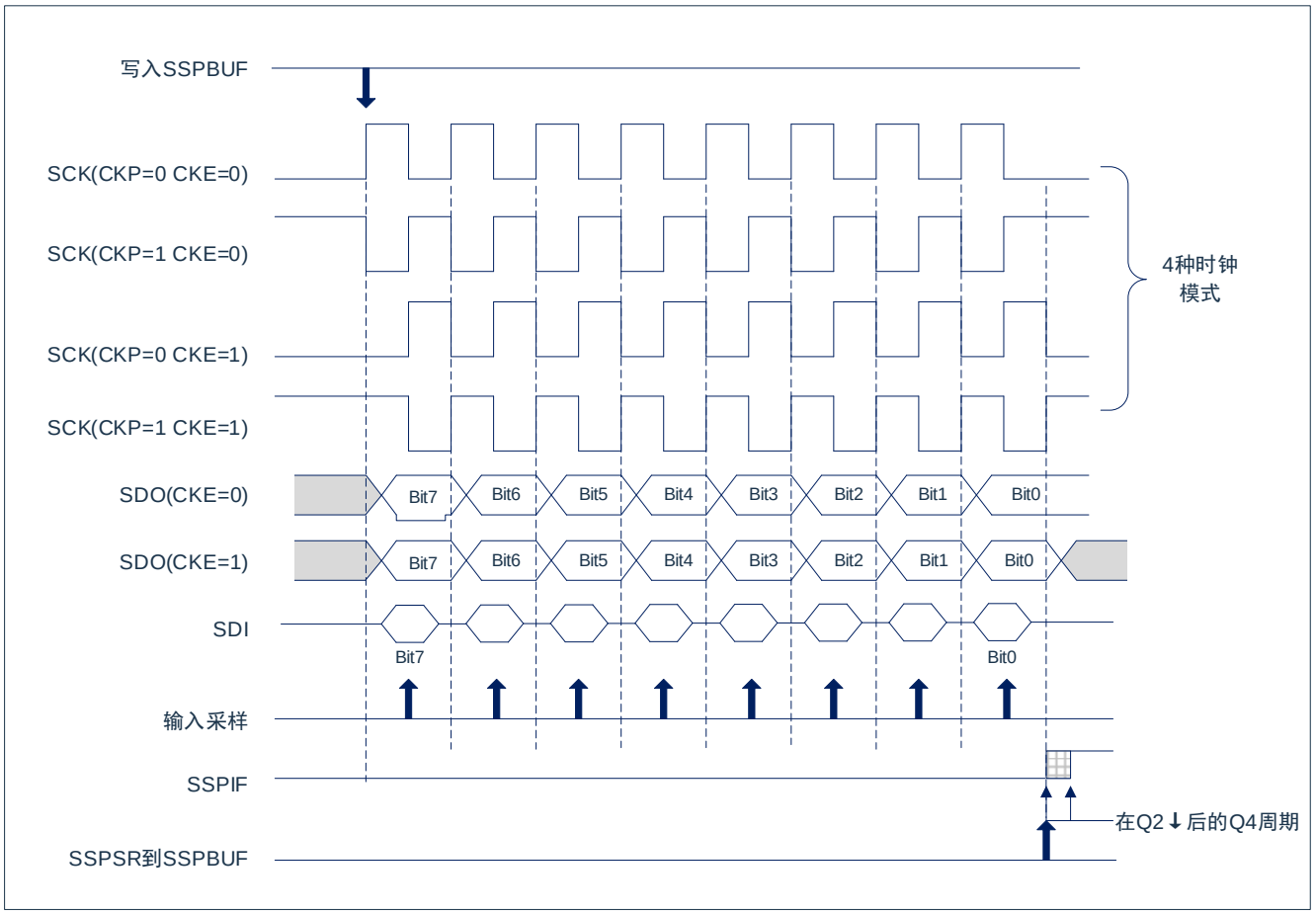


图 16-2：SPI 模式的波形（主控模式）

16.2.5 从动模式

在从动模式下，当 SCK 引脚上出现外部时钟脉冲时，发送和接收数据。当最后一位数据被锁存后，PIR1 寄存器的 SSPIF 中断标志位置 1。

在从动模式下，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。

16.2.6 从动选择同步

SS 引脚允许器件工作在同步从动模式。SPI 必须工作在从动模式下，并使能 SS 引脚控制 SSPxCON1<3:0> = 04h）。要使 SS 引脚用作输入引脚，不能将该引脚驱动为低电平。当 SS 引脚为低电平时，使能数据的发送和接收，同时 SDO 引脚被驱动。当 SS 引脚为高电平时，即使是在数据的发送过程中，SDO 引脚也不再被驱动，而是变成悬空输出。根据应用的需要，可外接上拉/ 下拉电阻。

当 SPI 模块复位后，位计数器被强制归 0。这可以通过强制将 SS 引脚拉为高电平或将 SSPEN 位清零来实现。将 SDO 引脚和 SDI 引脚相连可以仿真双线制通信。当 SPI 需要作为接收器工作时，SDO 引脚可以被配置为输入。这样就禁止了从 SDO 发送数据。因为 SDI 不会引起总线冲突，因而总是可以将其保留为输入（SDI 功能）。

注:

1. 当 SPI 工作在从动模式下，并且 SS 引脚控制使能（SSPxCON1<3:0> = 0100）时，如果 SS 引脚置为 VDD 电平，SPI 模块将被复位。
2. 如果在 CKE 置 1（SSPSTAT 寄存器）的从动模式下使用 SPI，则必须使能 SS 引脚控制。

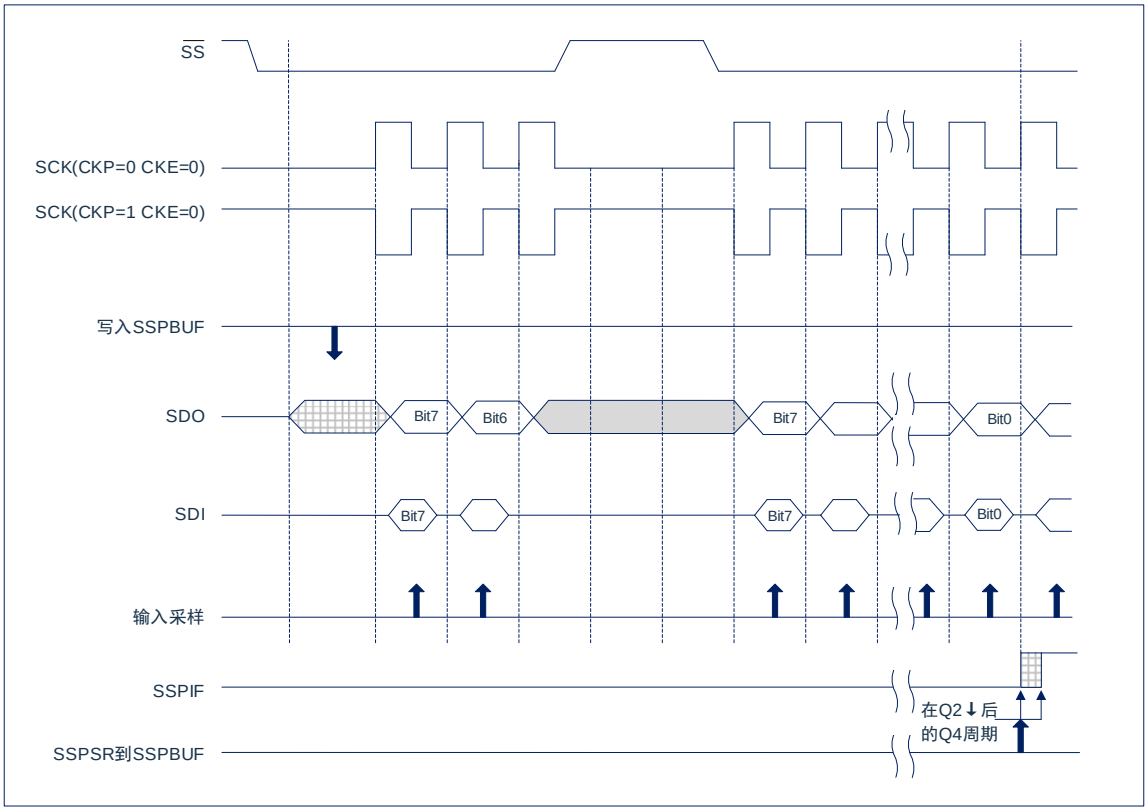


图 16-3：从动同步波形

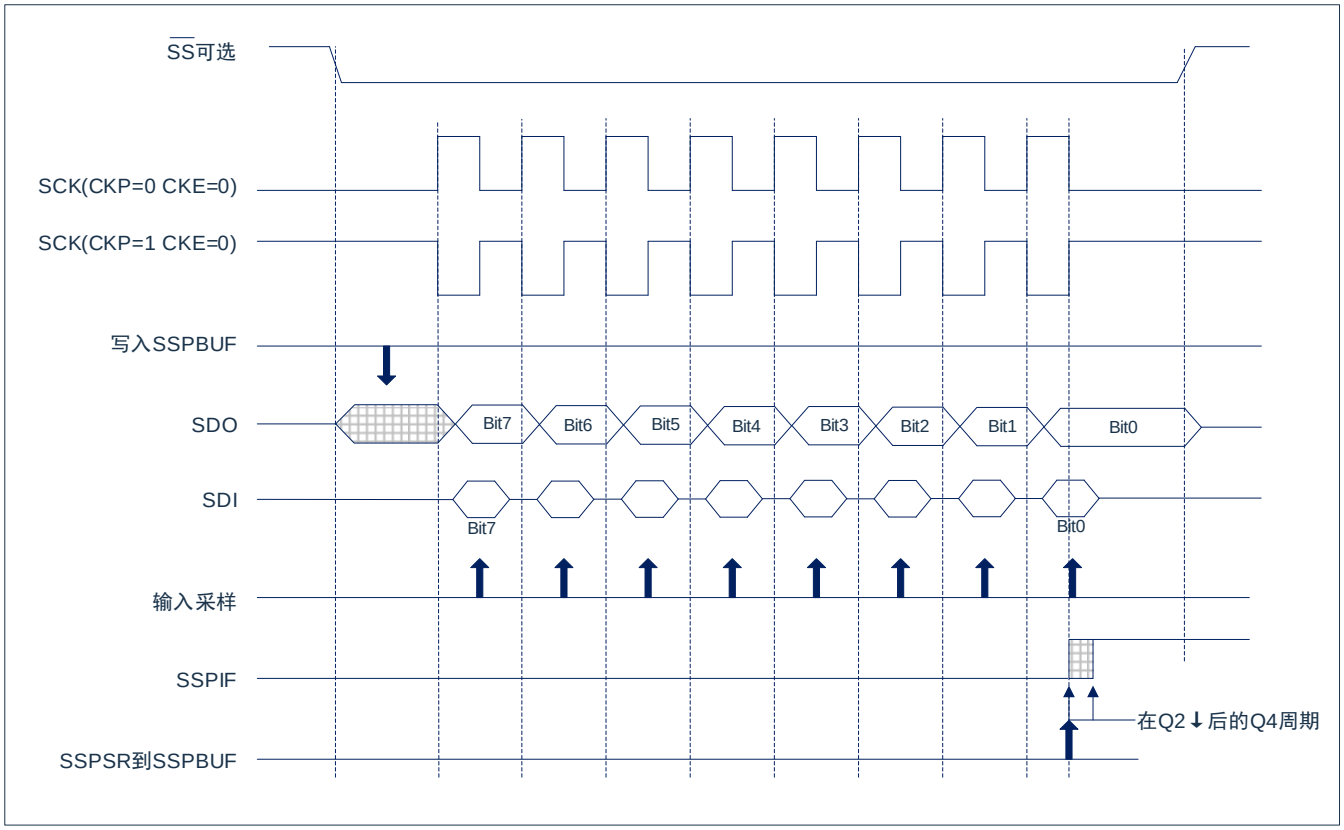


图 16-4：SPI 模式波形（从动模式，CKE=0）

16.2.7 休眠操作

在休眠模式，所有模块的时钟都将停止，并且在器件被唤醒前，发送/接收将保持此停滞状态。当器件返回到运行模式后，该模块将恢复发送和接收数据。

16.2.8 复位的影响

复位会禁止 MSSP 模块并终止当前的传输。

16.3 I²C 模块

MSSP 模块工作在 I²C 模式时，可以实现所有的主控和从动功能（包括广播呼叫支持），并且用硬件提供起始位和停止位的中断来判断总线何时空闲（多主机功能）。

有两个引脚用于数据传输，可以在系统配置寄存器中选择 I²C 功能的引脚分配在 RA5/RA6, RD0/RD1 或 RC6/RC7。使用 I²C 功能时，用户必须通过 TRISD<1:0>或 TRISC<7:6>位将这些引脚配置为输入引脚。通过将 SSPCON 寄存器的 MSSP 使能位 SSPEN 置 1，使能 MSSP 模块的功能。

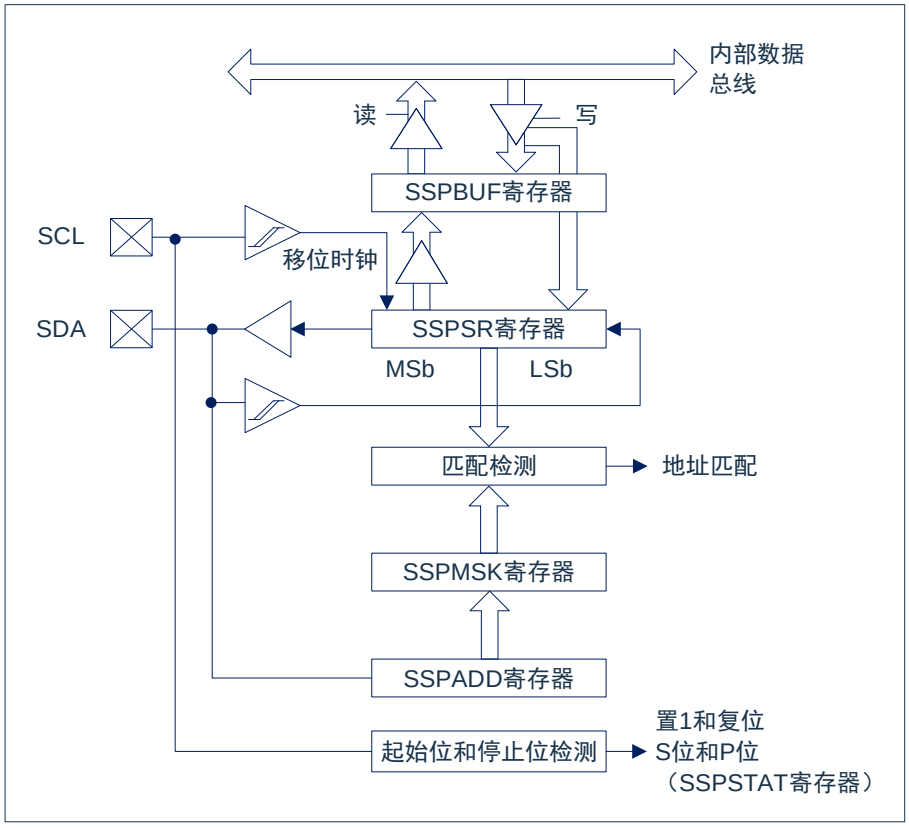


图 16-5：MSSP 框图（I²C 模式）

注：I/O 引脚具有连接到 VDD 和 VSS 的保护二极管。

MSSP 模块具有 7 个用于 I²C 操作的寄存器。它们是：

- MSSP 控制寄存器 1（SSPCON）
- MSSP 状态寄存器（SSPSTAT）
- MSSP 移位寄存器（SSPSR）：不能直接访问
- MSSP 屏蔽寄存器（SSPMSK）
- MSSP 控制寄存器 2（SSPCON2）
- 串行接收/发送缓冲寄存器（SSPBUF）
- MSSP 地址寄存器（SSPADD）

可使用 SSPCON 寄存器控制 I²C 的操作。可使用 SSPM<3:0>模式选择位（SSPCON 寄存器）选择以下 I²C 模式之一：

- I²C 从动模式，7 位地址，允许起始位和停止位中断
- I²C 主控模式，时钟=F_{SYS}/(4*(SSPADD+1))

如果已将 SCL 和 SDA 引脚编程为输入引脚（将相应的 TRIS 位置 1），选择任何 I²C 模式且 SSPEN 位置 1 将强制 SCL 和 SDA 引脚为漏极开路。

16.3.1 相关寄存器说明

SSPSTAT：SSP 状态寄存器

C7H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPSTAT	----	IDLE	D/A	P	S	R/W	IICTOF	BF
R/W	----	R	R	R	R	R	R/W	R
复位值	----	1	0	0	0	0	0	0

Bit7 未用 I^2C模式下无效

Bit6 IDLE 主控模式空闲位

（仅主控模式有效，所有主控操作都可以通过该位来判断是否结束）

1= 总线上没有主控操作

0= 总线上正在进行主控操作

Bit5 D/A: 数据/地址位

1= 表示最后接收或发送的字节是数据

0= 表示最后接收或发送的字节是地址

Bit4 P: 停止位（禁止MSSP模块（SSPEN清零）时此位被清零）

1= 表示最后检测到了停止位（复位时该位为0）

0= 表示最后未检测到停止位

Bit3 S: 起始位（禁止MSSP模块（SSPEN 清零）时此位被清零）

1= 表示最后检测到了起始位（复位时该位为0）

0= 最后未检测到起始位

Bit2 R/W: 读/写位信息

该位用来保存在最后一次地址匹配后的R/W位信息。该位仅在从地址匹配开始到下一个起始位、停止位或非ACK位时有效

在I^2C从动模式下:

1= 读

0= 写

在I^2C主控模式下:

1= 正在发送

0= 不在进行发送

此位与SEN、RSEN、PEN、RCEN或ACKEN做逻辑或运算的结果将指示MSSP是否在空闲模式下

Bit1 IICTOF: 从动模式超时标志位

1= 发生了从动模式超时

0= 未发生从动模式超时，写0清零此位

Bit0 BF 缓冲器满状态位

接收:

1= 接收完成，SSPBUF满

0= 接收未完成，SSPBUF空

发送:

1 = 数据正在发送（不包括ACK和停止位），SSPBUF满

0 = 数据发送完成（不包括ACK和停止位），SSPBUF空

SSPCON：SSP 控制寄存器

C4H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7 WCOL: 写冲突检测位

主控模式: 1= 在I^2C不满足开始发送数据的条件下，试图对SSPBUF寄存器进行写操作。

0= 未发生冲突

从动模式:

1= 正在发送前一个字时，又对SSPBUF寄存器进行写操作（必须由软件清零）

0= 未发生冲突

Bit6 SSPOV: 接收溢出指示位(仅在从动接收模式下有限)

1= SSPBUF寄存器仍保持前一数据时，又接收到一个新的字节。在发送模式下SSPOV位可为任意值（该位必须由软件清零）

0= 没有溢出

Bit5 SSPEN: 同步串行端口使能位（必须正确配置这些引脚为输入或输出引脚）

1= 使能串行端口并将SDA和SCL引脚配置为串行端口引脚

0= 禁止串行端口并将这些引脚配置为I/O端口引脚

Bit4 CKP: 时钟极性选择位

在I^2C从动模式下: SCK释放控制

1 = 使能时钟

0 = 保持时钟线为低电平（时钟延长）（用于确保数据建立时间）

在I^2C主控模式下: 在此模式下未使用

Bit3~Bit0 SSPM<3:0>: 同步串行端口模式选择位。

0000= SPI主控模式，时钟= $F_{SYS}/4$

0001= SPI主控模式，时钟= $F_{SYS}/16$

0010= SPI主控模式，时钟= $F_{SYS}/64$

0011= SPI主控模式，时钟= TMR2输出/2

0100= SPI从动模式，时钟= SCK引脚，使能SS引脚控制。

0101= SPI从动模式，时钟= SCK引脚，禁止SS引脚控制，SS可用作I/O引脚

0110= 保留

0111= 保留

1000= I^2C主控模式，时钟= $F_{SYS}/(4 * (SSPADD+1))$

1001= 禁止装载功能

1010= 保留

1011= 保留

1100= 保留

1101= 保留

1110= I^2C从动模式，7位地址，并允许起始位和停止位中断

1111= 保留

SSPCON2: SSP 控制寄存器 2

C5H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	1	0	0	0	0	0	0

Bit7 GCEN: 广播呼叫使能位（仅限I²C从动模式）
1= 允许在SSPSR中接收到广播呼叫地址（0000h）时产生中断
0= 禁止广播呼叫地址

Bit6 ACKSTAT: 应答状态位（仅限于I²C主控模式）
在主控发送模式下:
1 = 未接收到来自从动器件的应答
0 = 已接收到来自从动器件的应答

Bit5 ACKDT: 应答数据位（仅限于I²C主控模式）
在主控接收模式下: 用户在接收完成后发送的应答序列的值
1 = 不应答
0 = 应答

Bit4 ACKEN: 应答序列使能位（仅限I²C主控模式）
在主控接收模式下:
1 = 在SDA和SCL引脚启动应答序列，发送ACKDT数据位。由硬件自动清零
0 = 应答序列空闲

Bit3 RCEN: 接收使能位（仅限I²C主控模式）
1= 使能I²C接收模式
0= 接收空闲

Bit2 PEN: 停止条件使能位（仅限于I²C主控模式）
1 = 在SDA和SCL引脚启动停止条件。由硬件自动清零
0 = 停止条件空闲

Bit1 RSEN: 重复启动条件使能位（仅限I²C主控模式）
1= 在SDA和SCL引脚启动重复启动条件。由硬件自动清零
0= 重复启动条件空闲

Bit0 SEN: 启动条件使能位
在主控模式下:
1 = 在SDA和SCL引脚启动启动条件。由硬件自动清零
0 = 启动条件空闲

注：写入 SSPCON2 寄存器之前，必须把 SSPM[3:0]设置为 1000 或者 1110；如果退出 IIC 模式，SSPCON2 寄存器会被强制复位,无法写入 。

16.3.2 主控模式

主控模式通过在检测到启动和停止条件时产生中断来工作。停止（P）位和起始（S）位在复位时或禁止 MSSP 模块时清零。当 P 位置 1 时，可以取得 I²C 总线的控制权；否则总线处于空闲状态，且 P 位和 S 位都为零。

在主控模式中，SCL 线由 MSSP 硬件操纵，SDA 引脚必须被配置为输入（相应管脚 TRIS 位置 1）。下列事件会使 MSSP 中断标志位 SSPIF 置 1（如果允许 MSSP 中断，则产生中断）：

- 启动条件
- 停止条件
- 数据传输字节已发送/已接收
- 应答发送
- 重复启动条件

16.3.3 I²C 主控模式支持

通过将 SSPCON 中相应的 SSPM 位置 1 或清零并将 SSPEN 位置 1 可使能主控模式。一旦使能主控模式，用户即可选择以下 6 项操作：

1. 在 SDA 和 SCL 上发出一个启动条件。
2. 在 SDA 和 SCL 上发出一个重复启动条件。
3. 写入 SSPBUF 寄存器，开始数据/ 地址的发送。
4. 在 SDA 和 SCL 上产生停止条件。
5. 将 I²C 端口配置为接收数据。
6. 在接收到数据字节后产生应答条件。

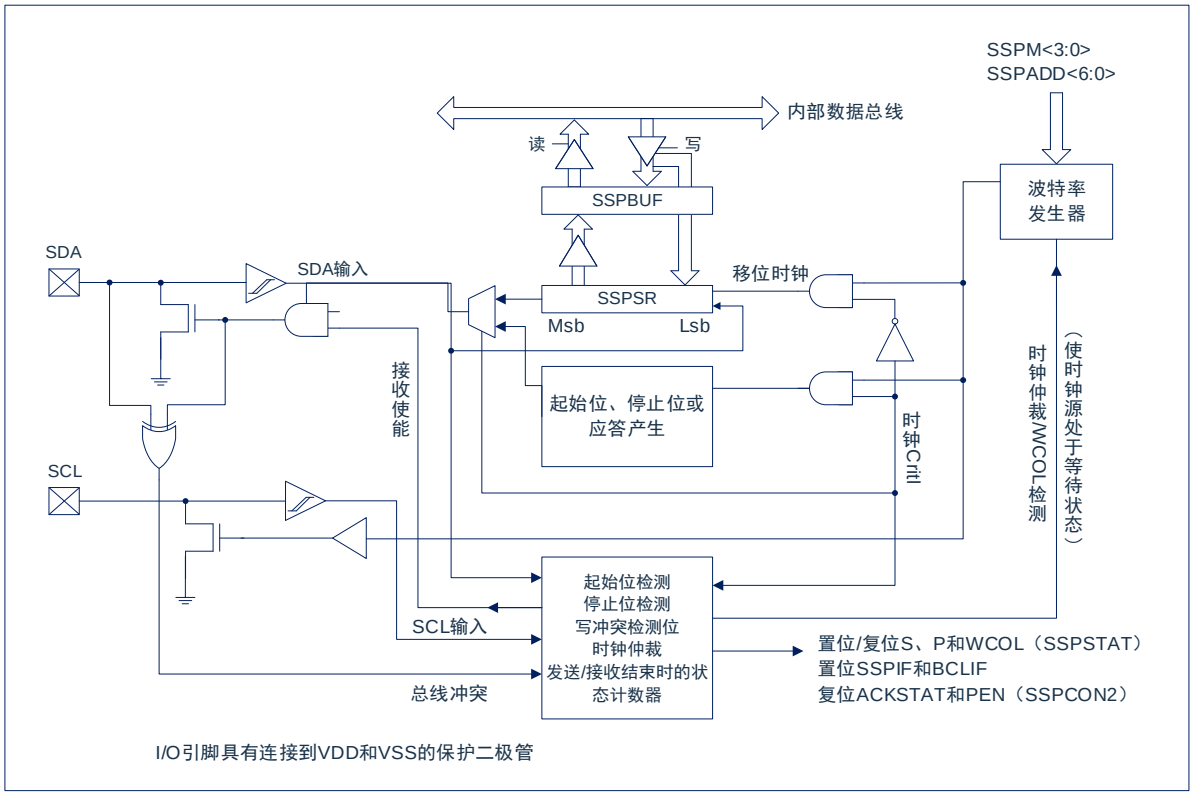


图16-6：MSSP框图（I²C™主控模式）

注：当配置为 I²C 主控模式时，MSSP 模块不允许事件排队。例如，在启动条件结束前，不允许用户发出另一个启动条件并立即写 SSPBUF 寄存器以发起传输。这种情况下，将不会写入 SSPBUF，WCOL 位将被置 1，这表明没有发生对 SSPBUF 的写操作。

16.3.3.1 I²C 主控模式操作

所有串行时钟脉冲和启动/ 停止条件均由主器件产生。停止条件或重复启动条件能结束传输。因为重复启动条件也是下一次串行传输的开始，因此不会释放 I²C 总线。在主控发送器模式下，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地址（7 位）和读/ 写（R/W）位。在这种情况下，R/W 位将是逻辑 0。串行数据每次发送 8 位。每发送一个字节，会收到一个应答位。启动和停止条件的输出表明串行传输的开始和结束。

在主控接收模式下，发送的第一个字节包括发送器件的地址（7 位）和 R/W 位。在这种情况下，R/W 位将是逻辑 1。因此，发送的第一个字节是一个 7 位从器件地址，后面跟 1 表示接收。串行数据通过 SDA 接收，而串行时钟由 SCL 输出。每次接收 8 位串行数据。每接收到一个字节，都会发送一个应答位。启动和停止条件分别表明发送的开始和结束。

在 I²C 模式下，在 SPI 模式中使用的波特率发生器被用于将 SCL 时钟频率设置为 100KHz、400KHz 或 1MHz。波特率发生器的重载值位于 SSPADD 寄存器的低 7 位。当发生对 SSPBUF 的写操作时，波特率发生器将自动开始计数。如果指定操作完成（即，发送的最后一个数据位后面跟着 ACK），内部时钟将自动停止计数，SCL 引脚将保持在其最后的状态。

下面是一个典型的发送事件序列：

- 用户通过将启动使能位 SEN（SSPCON2 寄存器）置 1 产生启动条件。
- SSPIF 位置 1。在进行任何其他操作前，MSSP 模块将等待所需的启动时间。
- 用户将从器件地址装入 SSPBUF 进行发送。
- 地址从 SDA 引脚移出，直到发送完所有 8 位为止。
- MSSP 模块移入来自从器件的 ACK 位，并将它的值写入 SSPCON2 寄存器的 ACKSTAT 位。
- MSSP 模块在第 9 个时钟周期的末尾将 SSPIF 位置 1，产生一个中断。
- 用户将 8 位数据装入 SSPBUF。
- 数据从 SDA 引脚移出，直到发送完所有 8 位为止。
- MSSP 模块移入来自从器件的 ACK 位，并将它的值写入 SSPCON2 寄存器的 ACKSTAT 位。
- MSSP 模块在第 9 个时钟的末尾将 SSPIF 位置 1，产生一个中断。
- 用户通过将停止使能位（PEN）位（SSPCON2 寄存器）置 1 产生停止条件。
- 一旦停止条件完成，将产生一个中断。

16.3.4 波特率发生器

在 I²C 主控模式下，波特率发生器的重载值位于 SSPADD 寄存器的低 7 位（图 16-7）。当装载了该值后，波特率发生器将自动开始计数并递减至 0，然后停止直到下次重载为止。BRG 会在每个指令周期（TCY）中的 Q2 和 Q4 时钟周期上进行两次减计数。在 I²C 主控模式下，会自动重载 BRG。例如，在发生时钟仲裁时，BRG 将在 SCL 引脚采样到高电平时重载（图 16-8）。

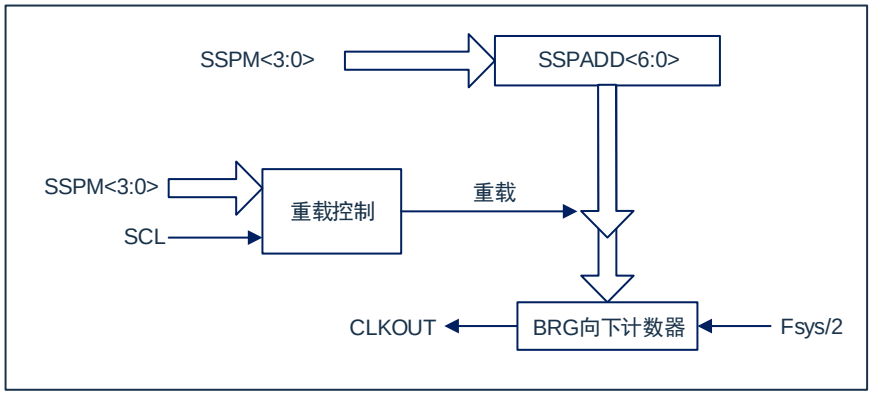


图16-7：波特率发生器框图

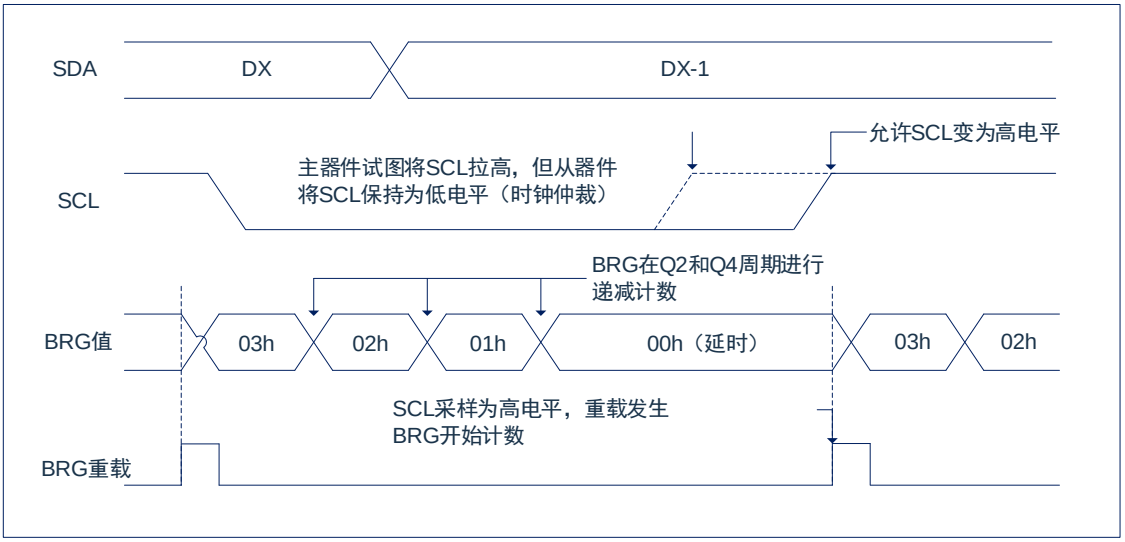


图16-8：带有时钟仲裁的波特率发生器时序

16.3.5 I²C 主控模式发送

发送一个数据字节、一个 7 位地址，都可以直接通过写一个值到 SSPBUF 寄存器来实现。该操作将使缓冲器满标志位 BF 置 1，并且波特率发生器开始计数，同时启动下一次发送。在 SCL 的下降沿有效后，地址/数据的每一位将被移出至 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。数据应该在 SCL 释放为高电平前保持有效。当 SCL 引脚被释放为高电平时，它将在整个 T_{BRG} 中保持高电平状态。在此期间以及下一个 SCL 下降沿之后的一段时间内，SDA 引脚上的数据必须保持稳定。在第 8 位被移出（第 8 个时钟周期的下降沿）之后，BF 标志位清零，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 位的时间以一个 ACK 位响应。ACK 的状态在第 9 个时钟周期的下降沿写入 ACKDT 位。主器件接收到应答之后，应答状态位 ACKSTAT 会被清零；如果未收到应答，则该位被置 1。第 9 个时钟之后，SSPIF 位会置 1，主控时钟（波特率发生器）暂停，直到下一个数据字节装入 SSPBUF 为止，SCL 引脚保持低电平，SDA 保持不变。

在写 SSPBUF 之后，地址的每一位在 SCL 的下降沿被移出，直至地址的所有 7 位和 R/W 位都被移出为止。在第 8 个时钟的下降沿，主器件将 SDA 引脚拉为高电平，以允许从器件发出应答响应。在第 9 个时钟的下降沿，主器件通过采样 SDA 引脚来判断地址是否被从器件识别。ACK 位的状态被装入 ACKSTAT 状态位（SSPCON2 寄存器）。在发送地址的第 9 个时钟下降沿之后，SSPIF 置 1，BF 标志位清零，波特率发生器关闭直到下一次写 SSPBUF，且 SCL 引脚保持低电平，允许 SDA 引脚悬空。

16.3.5.1 BF 状态标志

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

16.3.5.2 WCOL 状态标志位

如果用户在发送过程中（即，SSPSR 仍在移出数据字节时）写 SSPBUF，则 WCOL 置 1 且缓冲器的内容保持不变（未发生写操作）。WCOL 必须由软件清零。

16.3.5.3 ACKSTAT 状态标志

在发送模式下，当从器件发送应答响应（ACK= 0）时，ACKSTAT 位（SSPCON2 寄存器）清零；当从器件没有应答（ACK=1）时，该位置 1。从器件在识别出其地址（包括广播呼叫地址）或正确接收数据后，会发送一个应答。

16.3.6 I²C 主控模式接收

通过编程接收使能位 RCEN（SSPCON2 寄存器）使能主控模式接收。波特率发生器开始计数，每次计满返回时，SCL 引脚的状态都发生改变（由高变低或由低变高），且数据被移入 SSPSR。第 8 个时钟的下降沿之后，接收使能标志位自动清零，SSPSR 的内容装入 SSPBUF，BF 标志位置 1，SSPIF 标志位置 1，波特率发生器暂停计数，SCL 保持为低电平。此时 MSSP 处于空闲状态，等待下一条命令。当 CPU 读缓冲器时，BF 标志位将自动清零。通过将应答序列使能位 ACKEN （SSPCON2 寄存器）置 1，用户可以在接收结束后发送应答位。

16.3.6.1 BF 状态标志

接收时，当将地址或数据字节从 SSPSR 装入 SSPBUF 时，BF 位置 1；在读 SSPBUF 寄存器时 BF 位清零。

16.3.6.2 WCOL 状态标志

如果用户在接收过程中（即 SSPSR 仍在移入数据字节时）写 SSPBUF，则 WCOL 位置 1，缓冲器内容不变（未发生写操作）。

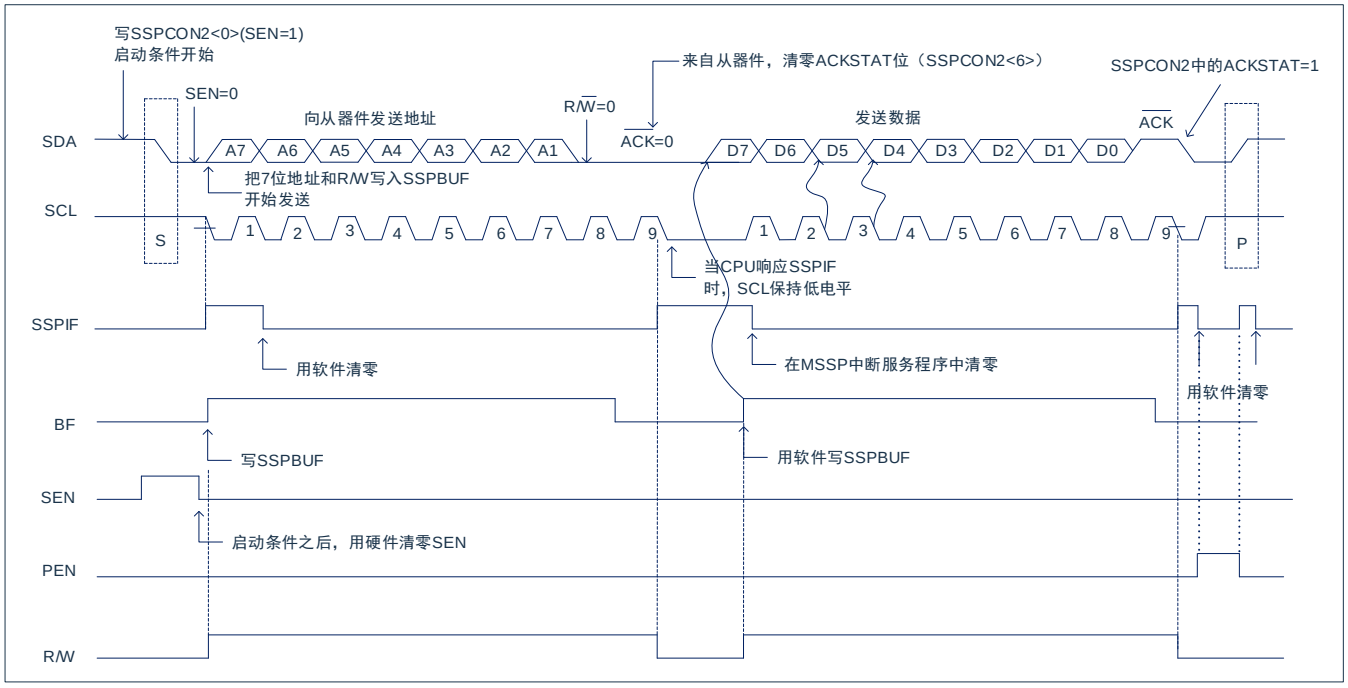


图16-9：I²C™主控模式发送时序

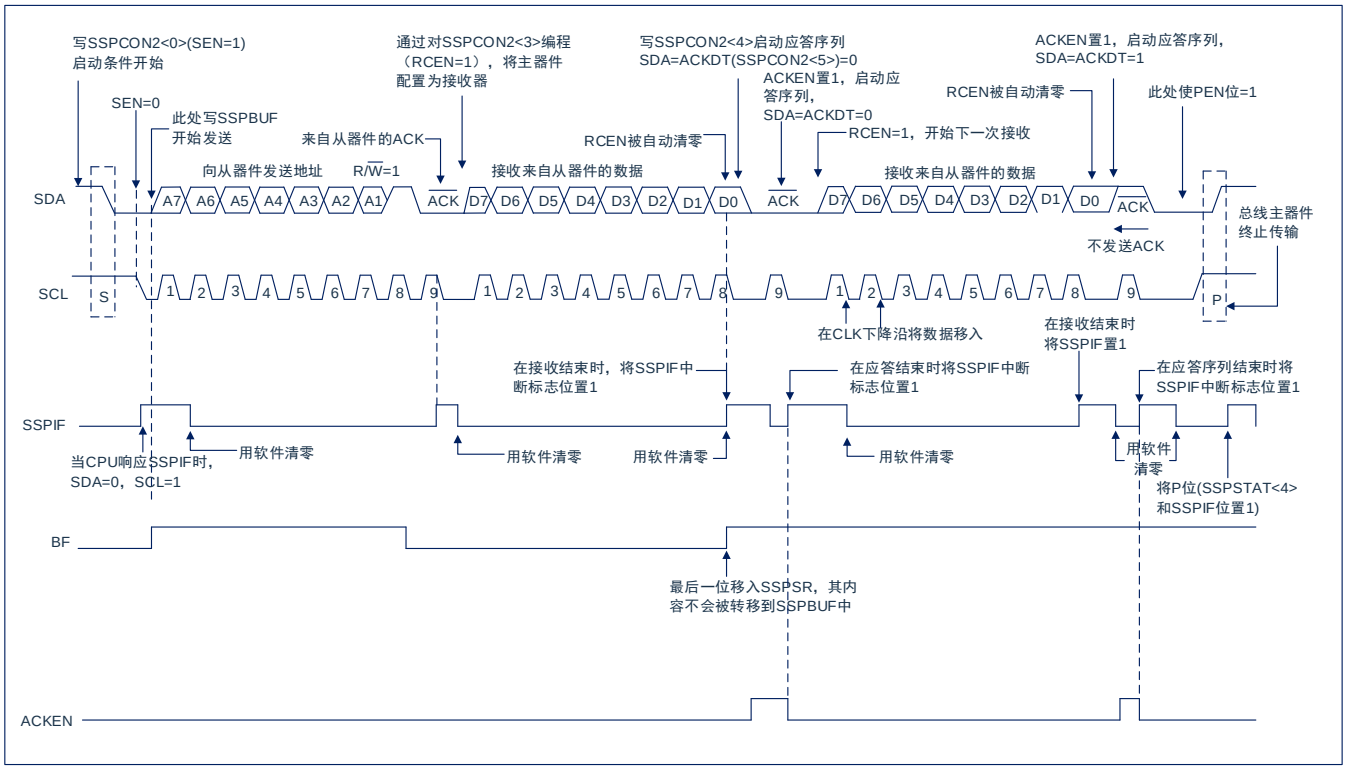


图16-10：I²C™主控模式接收时序（7位地址）

16.3.7 I²C 主控模式启动条件时序

要发起启动条件，用户应将 SSPCON2 寄存器的启动条件使能位 SEN 置 1。当 SDA 和 SCL 引脚都采样为高电平时，波特率发生器重新装入 SSPADD<6:0>的内容并开始计数。当波特率发生器发生超时（T_{BRG}）时，如果 SCL 和 SDA 都采样为高电平，则 SDA 引脚被驱动为低电平。当 SCL 为高电平时，将 SDA 驱动为低电平就是启动条件，将使 S 位（SSPSTAT 寄存器）置 1。随后波特率发生器重新装入 SSPADD<6:0>的内容并恢复计数。当波特率发生器超时（T_{BRG}）时，SSPCON2 寄存器的 SEN 位将自动被硬件清零。波特率发生器暂停工作，SDA 线保持低电平，启动条件结束。

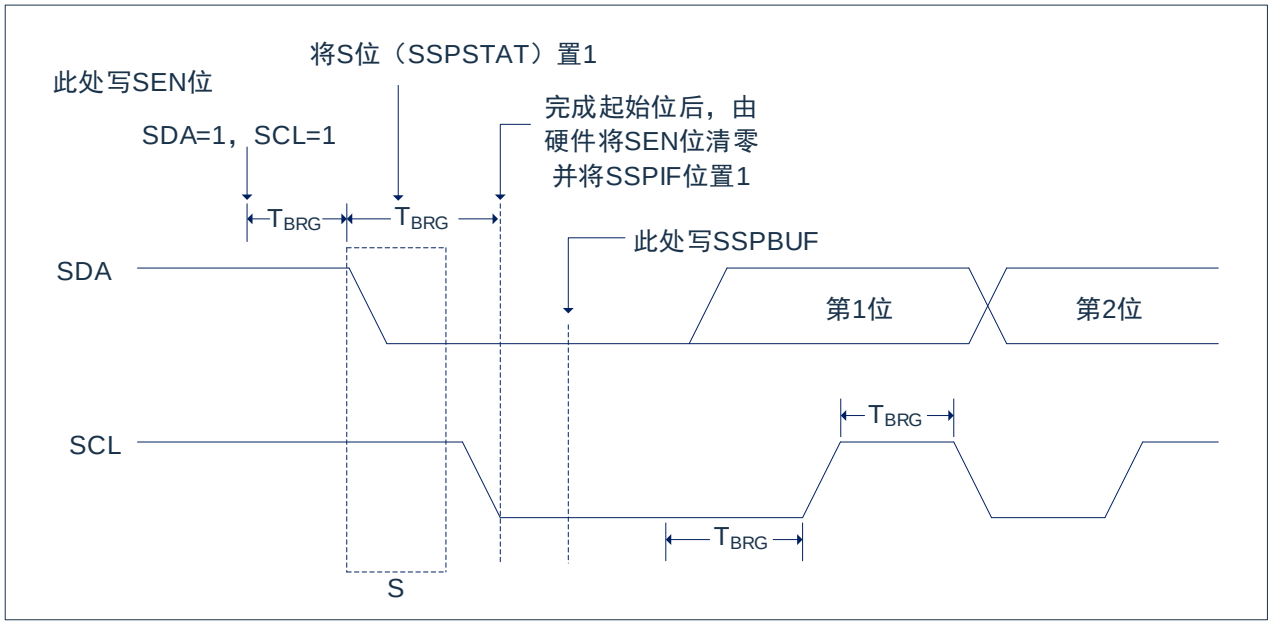


图16-11：第一个启动位时序

16.3.7.1 WCOL 状态标志

当启动序列进行时，如果用户写 SSPBUF，则 WCOL 被置 1，同时缓冲器内容不变（未发生写操作）。

注：由于不允许事件排队，在启动条件结束之前，不能对 SSPCON2 的低 5 位进行写操作。

16.3.8 I²C 主控模式重复启动条件时序

将 RSEN 位（SSPCON2 寄存器）编程为高电平，并且 I²C 逻辑模块处于空闲状态时，就会产生重复启动条件。当 RSEN 位置 1 时，SCL 引脚被拉为低电平。当 SCL 引脚采样为低电平时，波特率发生器装入 SSPADD<6:0>的内容，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（其引脚电平被拉高）。当波特率发生器超时时，如果 SDA 采样为高电平，SCL 引脚将被拉高。当 SCL 引脚采样为高电平时，波特率发生器将被重新装入 SSPADD<6:0>的内容并开始计数。SDA 和 SCL 必须在一个计数周期 T_{BRG} 内采样为高电平。随后将 SDA 引脚拉为低电平（SDA = 0）并保持一个计数周期 T_{BRG}，同时 SCL 为高电平。然后 RSEN 位（SSPCON2 寄存器）将自动清零，波特率发生器不会重载，SDA 引脚保持低电平。一旦在 SDA 和 SCL 引脚上检测到启动条件，S 位（SSPSTAT 寄存器）将被置 1。直到波特率发生器超时后，SSPIF 位才会置 1。

一旦 SSPIF 位被置 1，用户便可以将 7 位地址写入 SSPBUF，。当发送完第一个 8 位并接收到一个 ACK 后，用户可以发送 8 位数据。

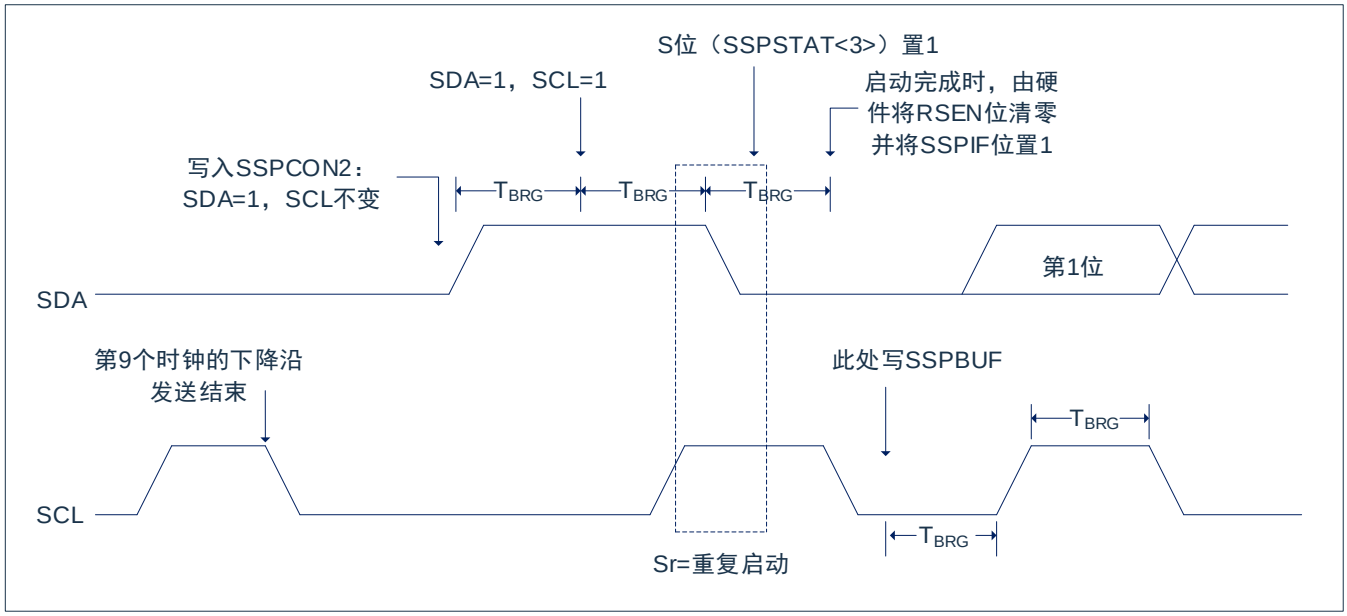


图16-12：重复启动条件时序波形

16.3.8.1 WCOL 状态标志

当重复启动序列进行时，如果用户写 SSPBUF，则 WCOL 被置 1，同时缓冲器内容不变（未发生写操作）。

注：由于不允许事件排队，在重复启动条件结束之前，不能对SSPCON2的低5位进行写操作。

16.3.9 应答序列时序

将应答序列使能位 ACKEN（SSPCON2 寄存器）置 1 即可使能应答序列。当该位被置 1 时，SCL 引脚被拉低，应答数据位的内容出现在 SDA 引脚上。如果用户希望产生一个应答，则应该将 ACKDT 位清零；否则，用户应该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，随后 SCL 引脚电平被拉高。当 SCL 引脚采样为高电平时（时钟仲裁），波特率发生器再进行一个 T_{BRG} 周期的计数。然后 SCL 引脚被拉低。在这之后，ACKEN 位自动清零，波特率发生器关闭，MSSP 模块进入空闲模式。

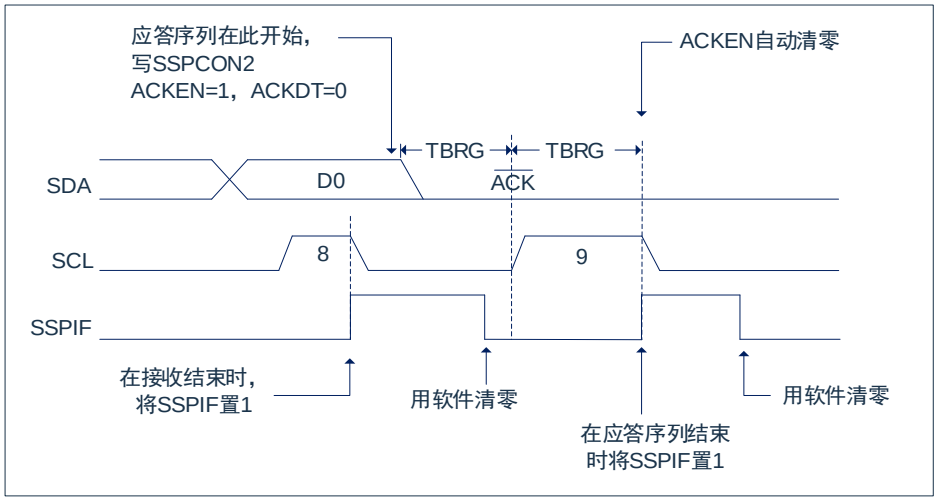


图 16-13：应答序列时序波形

注：T_{BRG}=一个波特率发生器周期。

16.3.9.1 WCOL 状态标志位

如果用户在应答序列正在进行时写 SSPBUF，WCOL 将被置 1 且缓冲器的内容保持不变（未发生写操作）。

16.3.10 停止条件序列

在接收/发送结束时，通过置 1 停止序列的使能位，PEN（SSPCON2 寄存器），SDA 引脚将产生一个停止位。在接收/发送结束时，SCL 引脚在第 9 个时钟的下降沿后保持低电平。当 PEN 位置 1 时，主控器件将 SDA 置为低电平。当 SDA 线采样为低电平时，波特率发生器被重新装入值并递减计数至 0。波特率发生器发生超时时，SCL 引脚被拉到高电平，且一个 T_{BRG}（波特率发生器计满回零）后，SDA 引脚被重新拉到高电平。当 SDA 引脚采样为高电平且 SCL 也是高电平时，P 位（SSPSTAT 寄存器）置 1。一个 T_{BRG} 周期后，PEN 位清零且 SSPIF 位置 1。

16.3.10.1 WCOL 状态标志

如果用户在停止序列进行过程中试图写 SSPBUF，则 WCOL 位将置 1，缓冲器的内容不会改变（未发生写操作）。

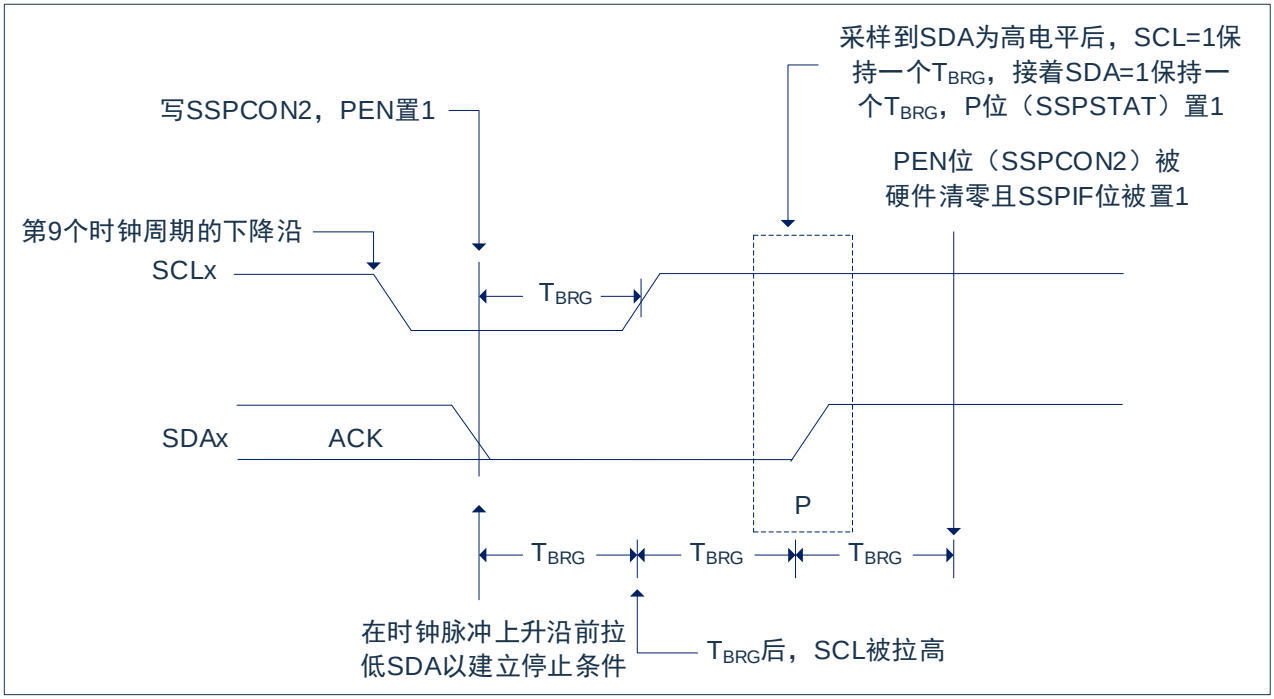


图 16-14：停止条件接收或发送模式

注：T_{BRG}=一个波特率发生器周期。

16.3.11 时钟仲裁

如果在任何接收、发送或重复启动/ 停止条件期间，主器件拉高了 SCL 引脚（允许 SCL 引脚悬空为高电平），就会发生时钟仲裁。如果允许 SCL 引脚悬空为高电平，波特率发生器（BRG）将暂停计数，直到实际采样到 SCL 引脚为高电平为止。当 SCL 引脚采样为高电平时，波特率发生器中将被重新装入 SSPADD<6:0>的内容并开始计数。这可以保证当外部器件将时钟拉低时，SCL 始终保持至少一个 BRG 计满返回周期的高电平。

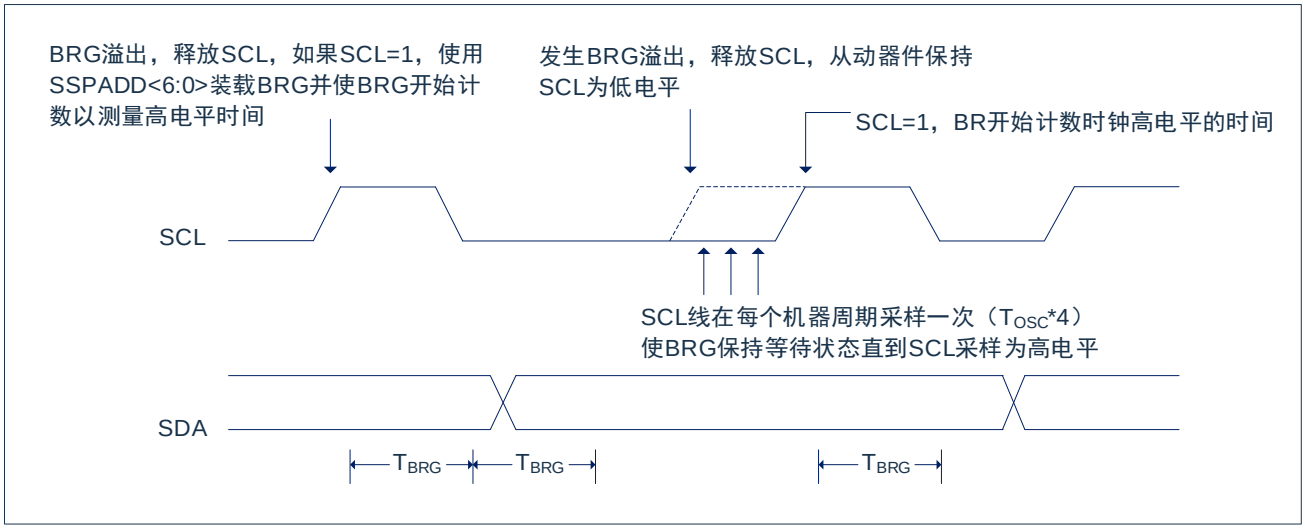


图 15-15：主控发送模式下的时钟仲裁时序

16.3.12 多主机模式

在多主机模式下，通过在检测到启动和停止条件时产生中断可以确定总线何时空闲。停止（P）位和启动（S）位在复位时或禁止 MSSP 模块时清零。当 P 位置 1 时，可以取得 I^2C 总线的控制权；否则总线处于空闲状态，且 P 位和 S 位清零。当总线忙时，如果出现停止条件，则将产生中断（若允许 MSSP 中断）。

在多主机模式下工作时，必须监视 SDA 线来进行仲裁，查看信号电平是否为期望的输出电平。此检查由硬件完成，其结果放在 BCLIF 位。

在以下状态下仲裁可能失败：

- 地址传输
- 数据传输
- 启动条件
- 重复启动条件
- 应答条件

16.3.13 多主机通信、总线冲突与总线仲裁

多主机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，如果一个主器件通过将 SDA 引脚悬空为高电平以在 SDA 上输出 1，而另一个主器件输出 0，就会发生总线仲裁。如果 SDA 引脚上期望的数据是 1，而实际在 SDA 引脚上采样到的数据是 0，则发生了总线冲突。主器件将把总线冲突中断标志位 BCLIF 置 1，并将 I²C 端口复位到空闲状态。

如果在发送过程中发生总线冲突，则发送停止，BF 标志位清零，SDA 和 SCL 线被拉高，并且允许对 SSPBUF 进行写操作。当执行完总线冲突中断服务程序后，如果 I²C 总线空闲，用户可通过发出启动条件恢复通信。如果在启动、重复启动、停止或应答条件的进行过程中发生总线冲突，则该条件被中止，SDA 和 SCL 线被拉高，SSPCON2 寄存器中的对应控制位清零。当执行完总线冲突中断服务程序后，如果 I²C 总线空闲，用户可通过发出启动条件恢复通信。主器件将继续监视 SDA 和 SCL 引脚。如果出现停止条件，SSPIF 位将被置 1。无论发生总线冲突时发送的进度如何，写 SSPBUF 都会从第一个数据位开始发送数据。

在多主机模式下，通过在检测到启动和停止条件时产生中断可以确定总线何时空闲。P 位置 1 时，可以获取 I²C 总线的控制权，否则总线空闲且 S 和 P 位清零。

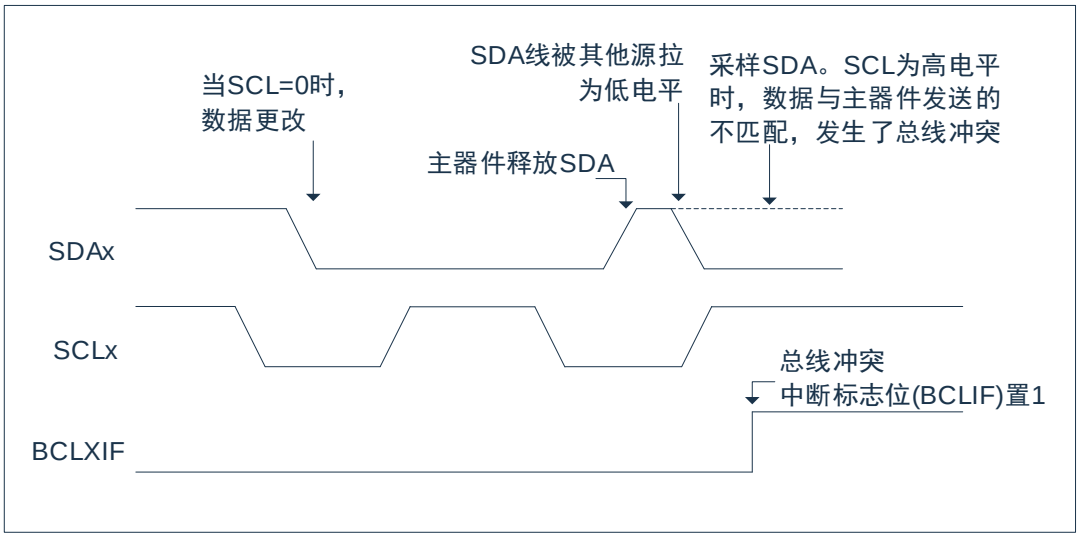


图 16-16：发送和应答时的总线冲突时序

16.3.14 从动模式

在从动模式下，SCL 引脚和 SDA 引脚必须被配置为输入（TRISC<7:6>置 1）。需要时（如从发送器）MSSP 模块将用输出数据改写输入状态。

当地址匹配时或在地址匹配后传输的数据被接收时，硬件会自动产生一个应答（ACK）脉冲，并把当时 SSPSR 寄存器中接收到的数据装入 SSPBUF 寄存器。

只要满足下列条件之一，MSSP 模块就不会产生此 ACK 脉冲：

- 缓冲器满标志位 BF（SSPCON 寄存器）在接收到传输的数据前置 1。
- 在接收到传输的数据之前，溢出标志位 SSPOV（SSPCON 寄存器）已被置 1。

在这种情况下，SSPSR 寄存器的值不会载入 SSPBUF，但是 PIR1 寄存器的 SSPIF 位会置 1。BF 位是通过读取 SSPBUF 寄存器清零的，而 SSPOV 位是通过软件清零的。

为确保正常工作，SCL 时钟输入必须满足最小高电平时间和最小低电平时间要求。

16.3.14.1 寻址

一旦使能了 MSSP 模块，它就会等待启动条件产生。在启动条件出现后，8 位数据被移入 SSPSR 寄存器。在时钟（SCL）线的上升沿采样所有的输入位。寄存器 SSPSR<7:1>的值会和寄存器 SSPADD<7:1>的值比较，该比较是在第 8 个时钟脉冲（SCL）的下降沿进行的。如果地址匹配，并且 BF 位和 SSPOV 位为零，会发生下列事件：

- SSPSR 寄存器的值被装入 SSPBUF 寄存器。
- 缓冲器满标志位 BF 置 1。
- 产生 ACK 脉冲。
- 在第 9 个 SCL 脉冲的下降沿，PIR1 寄存器的 MSSP 中断标志位 SSPIF 置 1（如果允许中断则产生中断）。

16.3.14.2 接收

当地址字节的 R/W 位清零并发生地址匹配时，SSPSTAT 寄存器的 R/W 位清零。接收到的地址被装入 SSPBUF 寄存器。

当存在地址字节溢出条件时，则不会产生应答脉冲(ACK)。溢出条件是指 BF 位（SSPSTAT 寄存器）置 1，或者 SSPOV 位（SSPCON 寄存器）置 1。每个数据传输字节都会产生一个 MSSP 中断。必须用软件将 PIR1 寄存器的中断标志位 SSPIF 清零。SSPSTAT 寄存器用于确定该字节的状态。

16.3.14.3 发送

当接收的地址字节的 R/W 位置 1 并发生地址匹配时，SSPSTAT 寄存器的 R/W 位置 1。接收到的地址被装入 SSPBUF 寄存器。ACK 脉冲在第 9 位上发送，同时 SDA 引脚保持低电平。传输的数据必须装入 SSPBUF 寄存器，同时也被装入 SSPSR 寄存器。随后应通过将 CKP 位（SSPCON 寄存器）置 1 使能 SCL 引脚。在发送另一个时钟脉冲前，主控器件必须监视 SCL 引脚。从动器件可以通过延长时钟，暂停与主控器件的数据传输。8 个数据位在 SCL 输入的下降沿移出。这可确保在 SCL 为高电平期间 SDA 信号是有效的。

每个数据传输字节都会产生一个 MSSP 中断。SSPIF 标志位必须由软件清零，SSPSTAT 寄存器用于确定字节的状态。SSPIF 位在第 9 个时钟脉冲的下降沿被置 1。来自主接收器的 ACK 脉冲将在 SCL 输入第 9 个脉冲的上升沿锁存。如果 SDA 线为高电平（无 ACK），那么表示数据传输已完成。在这种情况下，如果从器件锁存了 ACK，将复位从动逻辑（复位 SSPSTAT 寄存器），同时从器件监视下一个起始位的出现。如果 SDA 线为低电平（ACK），则必须将接下去要发送的数据装入 SSPBUF 寄存器，这也将装载 SSPSR 寄存器。应将 CKP 置 1 使能 SCL。

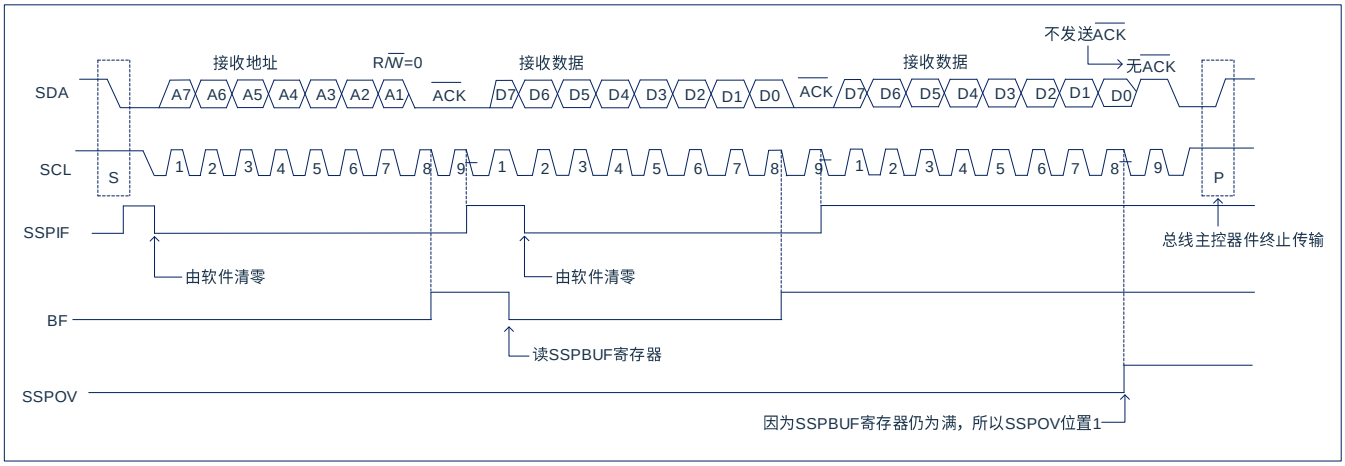


图 16-17：I²C™ 从动模式接收时序（7 位地址）

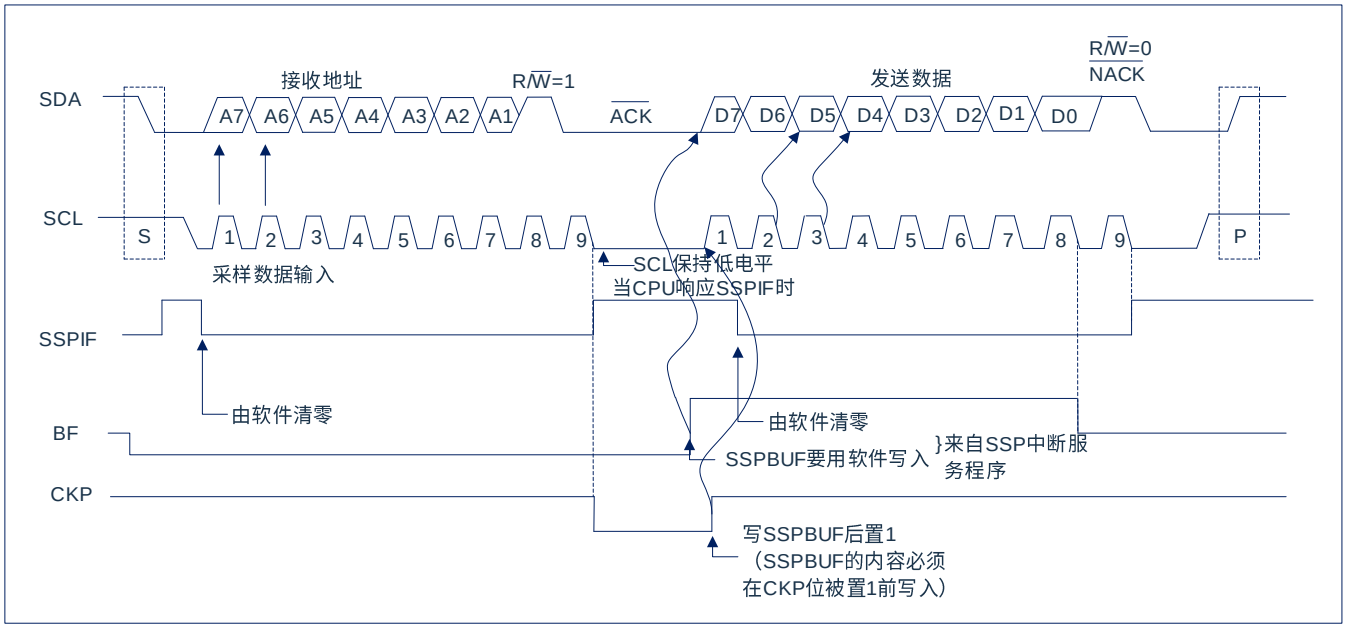


图 16-18：I²C™ 从动模式发送时序（7 位地址）

16.3.14.4 I2C 从动超时保护

在从动模式下，可以开启超时保护功能。超时计数器在地址匹配后开始起作用，在 SCL 下降沿清零，当下一个 SCL 下降沿来的时候，若计数时钟超过 64ms，则触发超时保护功能，此时自动复位 IIC 从动接收模块，总线恢复空闲状态，同时 IICTOF 标志位置 1（通过软件清零）。

16.3.15 SSP 屏蔽寄存器

在 I²C 从动模式下，SSP 屏蔽（SSPMSK）寄存器用于在地址比较操作下屏蔽 SSPSR 寄存器中的值。SSPMSK 寄存器中某位为 0 会使 SSPSR 寄存器中相应的位成为“无关位”。

此寄存器在任何复位条件发生时均复位为全 1，因此，在写入屏蔽值前，它对标准 SSP 操作没有影响。必须在通过设置 SSPM<3:0>位以选择 I²C 从动模式之前对此寄存器进行初始化。只有通过 SSPCON 的 SSPM<3:0>位选择了适当的模式后才可访问此寄存器。

SSP 屏蔽寄存器在以下情况下有效：

- 7 位地址模式：与 A<7:1>进行地址比较。
- 10 位地址模式：仅与 A<7:0>进行地址比较。

SSP 屏蔽在接收到地址的第一个（高）字节期间无效。

SSPMSK:SSP 屏蔽寄存器(93H) (1)

93H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SSPMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	MSK0(2)
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

Bit7~Bit1　　MSK<7:1>:　屏蔽位
1= 接收到的地址的Bit n 与SSPADD<n>比较以检测I²C的地址匹配情况
0= 接收到的地址的Bit n 不用于检测I²C的地址匹配情况

Bit 0　　未用

注：
1) 当SSPCON位SSPM<3:0> = 1001时，任何对SSPADDSFR地址的读或写操作都通过SSPMSK寄存进行。
2) 在所有其他 SSP 模式下，此位无效。

16.3.16 休眠模式下的操作

在休眠模式下，I²C 模块无法使用。

16.3.17 复位的影响

复位会禁止 MSSP 模块并终止当前的传输。

17. Flash 存储器

17.1 概述

该系列中器件具有 16K 字节的程序存储器，地址范围从 000h 到 3FFFh，在所有地址范围内都是可读写的；注意的是读写程序存储器时，是按字读写的，也就是说每次读写 2 个字节数据。器件具有 256 字的非易失性数据存储器，地址范围为 00h 到 FFh，在所有地址范围内都是可读写的。

这些存储器并不直接映射到寄存器空间，而是通过特殊功能寄存器（SFR）对其进行间接寻址。以下 SFR 寄存器用于访问这些存储器空间：

- EECON1
- EECON2
- EECON3
- EEDAT
- EEDATH
- EEADR
- EEADRH

存储器支持字读写，字写操作可自动擦除目标单元并写入新数据；存储器还支持页操作(擦除/烧写)，在使用页操作改写存储器数据时，需要先将目标页擦除，再对目标页进行页烧写。写入时间由片上定时器控制。写入和擦除电压是由片上电荷泵产生的，此电荷泵额定工作在器件的电压范围内，用于进行字操作或页操作。

当器件受代码保护时，CPU 仍可继续读写存储器，器件编程器将不能再访问存储器。

注：Flash 存储器正常写电压范围为 2.5V~5.5V，写电流为 6mA@VDD=5V.

17.2 相关寄存器

17.2.1 Flash 存储器控制寄存器

Flash 存储器控制寄存器 EECON1

0xFD	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON1	EEPGD[1]	EEPGD[0]	---	SERIOK	WRERR	WREN	WR	RD
R/W	R/W	R/W	---	R	R/W	R/W	R/W	R/W
复位值	0	0	---	0	X	0	0	0

Bit7~6	EEPGD[1:0]	程序/数据存储器选择位，
	1x=	操作程序存储器
	01=	未用
	00=	操作数据存储器
Bit5	未用	
Bit4	SERIOK:	存储器修改操作使能序列匹配状态位
	1:	匹配
	0:	未匹配（对存储器的修改操作均无法启动(WR/HVPLEN/START)）
Bit3	WRERR:	写错误标志位；
	1=	写操作出错（正常工作期间的任何WDT复位或欠压复位）
	0=	写操作完成
Bit2	WREN:	写使能位
	1=	允许写周期
	0=	禁止写入存储器
Bit1	WR:	写控制位
	1=	启动写周期（写操作一旦完成由硬件清零该位，用软件只能将WR位置1，但不能清零）；
	0=	写周期完成
Bit0	RD:	读控制位
	1=	启动存储器读操作（由硬件清零RD，用软件只能将RD位置1，但不能清零）
	0=	不启动存储器读操作

注：将 WR 位或 RD 位置 1 后需要等待 6 条 NOP 指令后才能进行其他操作。

EECON1 控制位 EEPGD[1:0]决定访问的是程序存储器，还是数据存储器。EEPGD[1:0]=00 时，任何后续操作都将针对数据存储器进行。EEPGD[1:0]=1x 时，任何后续操作都将针对程序存储器进行。

EECON1 控制位 RD 和 WR 分别启动字读和字写。用软件只能将这些位置 1 而无法清零。在读或写操作完成后，由硬件将它们清零。

EECON1 控制位 WREN 置 1 时，允许对存储器执行写操作。上电时，WREN 位被清零。当正常的写入操作被 LVR 复位，WRERR 位会置 1。在这些情况下，复位后用户可以检查 WRERR 位并重写相应的单元。

Flash 存储器控制寄存器 EECON2

0xFE	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON2	SERI[7:0]							
R/W	W	W	W	W	W	W	W	W
复位值	0	0	0	0	0	0	0	0

Bit7~0 SERI[7:0]: MTP修改操作使能序列
选择数据存储器时,先写0x55,再写0xaa: 使能数据存储器修改操作（写入WR/START均有效）
选择程序存储器时,先写0x69,再写0x96: 使能程序存储器修改操作（写入WR/START均有效）
写入其余值: 禁止存储器修改操作（写入WR/START均无效）

EECON2 不是物理寄存器，读 EECON2 得到的是全 0。该寄存器在需要修改目标存储器数据时使用，需要按访问的存储器类型写入特定序列；可通过 EECON1 的标志位 SERIOK 读取当前序列匹配状态。

Flash 存储器控制寄存器 EECON3

0xFF	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON3	---	---	PAGE_MODE	START	HVPLEN	---	OP[1]	OP[0]
R/W	R/W	R/W	R/W	R/W	R/W	---	R/W	R/W
复位值	0	0	0	0	0	---	0	0

Bit7~Bit6 未用
Bit5 PAGE_MODE: 页操作模式控制位
1= 使能页操作模式
0= 禁止页操作模式
Bit4 START: 启动 ERASE/PROG 操作
1= 启动
0= 停止；硬件自动清零，结束 ERASE/PROG 操作
Bit3 HVPLEN: 写缓存使能
1= 使能写缓存，使能后会有以下动作
1. HVPLEN从0->1时，硬件会自动清除缓存，并使能硬件计数器
2. 由硬件计数器控制缓存地址（16个缓存地址0~f）
3. 写EEDAT寄存器时，数据更新至缓存，后硬件计数器加1

0= 禁止写缓存
Bit2 未用
Bit1~Bit0 OP[1:0]: 操作选择位
11= 未用
10= ERASE
01= PROG
00= 未用

注：将 START 位或 HVPLEN 位置 1 后需要等待 6 条 NOP 指令后才能进行其他操作。

EECON3 控制位 HVPLEN 控制页缓存使能，使能时硬件将自动清除存储器缓存并使能内部硬件计数器。此时写 EEDAT 寄存器会将 EEDATH 和 EEDAT 的数据更新至硬件计数器对应地址的缓存空间，完成后硬件计数器自动加 1，重复将 16 个缓存空间写满，等待页操作执行。

EECON3 控制位 OP 选择页操作的实际动作（擦除/烧写）

EECON3 控制位 START 启动页操作，结束后该位由硬件自动清零，并自动清除缓存和重置硬件计数器；此时由 EEADRH 和 EEADR 选择存储器页操作的页。

17.2.2 Flash 存储器地址寄存器

EEADR 和 EEADRH 寄存器能寻址最大 256 字的数据存储器或最大 16K 字节的程序存储器。

当选择程序存储器地址值时，地址的高字节被写入 EEADRH 寄存器而低字节被写入 EEADR 寄存器；注意的是程序存储器是按字读写的，每次读写会连续读写两个地址。

程序存储器读写地址设置示例：

当处于程序存储器区时，EEADR=0x00，EEADRH=0x00，实际读写地址为 0x0000~0x0001；当 EEADR=0x01，EEADRH=0x00，实际读写地址为 0x0002~0x0003；以此类推。EEDAT 为低位地址的读写数据寄存器，EEDATH 为高位地址的读写数据寄存器。

当选择数据存储器地址值时，只将地址的低字节写入 EEADR 寄存器。

Flash 存储器地址寄存器 EEADR

0xFB	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0	EEADR<7:0>:	指定存储器读/写操作的地址的低8位 EEADR<7:4>指定存储器页操作的地址低4位

注：写 EEADR 寄存器指令后面需要加一条 NOP 指令

Flash 存储器地址寄存器 EEADRH

0xFC	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEADRH	---	---	---	EEADRH4	EEADRH3	EEADRH2	EEADRH1	EEADRH0
R/W	---	---	---	R/W	R/W	R/W	R/W	R/W
复位值	---	---	---	0	0	0	0	0

Bit7~Bit5	未用，读为0	
Bit4~Bit0	EEADRH<4:0>:	指定程序存储器读/写操作的地址高5位 指定程序存储器页操作的地址高5位

注：写 EEADRH 寄存器指令后面需要加一条 NOP 指令

17.2.3 Flash 存储器数据寄存器

Flash 存储器数据寄存器 EEDAT

0xF9	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0　　EEDAT<7:0>:　要从存储器中读取或向存储器写入数据的低8位，或者要向缓存写入数据的低8位

注：页操作模式下写缓存时，写 EEDAT 寄存器后需要等待 6 条 NOP 指令后才能进行其他操作。

Flash 存储器数据寄存器 EEDATH

0xFA	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0　　EEDATH<7:0>:　要从存储器中读取或向存储器写入数据的高8位，或者要向缓存写入数据的高8位

17.3 读 Flash 存储器

要读取 Flash 存储器内的数据，用户必须将地址的低位和高位分别写入 EEADR 和 EEADRH 寄存器，写 EECON1 寄存器的 EEPGD 控制位选择目标存储器类型，然后将控制位 RD 置 1。FLASH 存储器读操作时，CPU 处于暂停状态，操作完成时，CPU 继续运行指令；存储器相应地址的值会被锁存到 EEDAT 和 EEDATH 寄存器中，用户可在随后的指令中读取这两个寄存器。

注：将 RD 位置 1 后需要等待 6 条 NOP 指令后才能进行其他操作。

读程序/数据存储器数据流程示例：

1. 写 EECON1.EEPGD 控制位选择目标存储器的类型；
2. 将目标地址写入 EEADR 和 EEADRH 寄存器；
3. 将 EECON1.RD 置 1，启动读操作（CPU 将暂停至操作完成）；
4. 等待 6 条 NOP 指令后，判断等待 RD 是否清零；
5. 读取 EEDAT 和 EEDATH 寄存器获取数据。

17.4 写 Flash 存储器

对 Flash 存储器进行修改操作前，需要先设置 EECON1 寄存器的 EEPGD，选择程序或者数据存储器，再向 EECON2 写入特定序列以使能存储器的修改操作；

修改 Flash 存储器数据有两种方式：字写和页操作；字写操作是集成了清缓存、单次写缓存、擦除和烧写，其特点是能写特定地址的 Flash 存储器；而页操作则是用于为了批量操作 Flash 存储器，提高修改效率的一种方式。在实际应用中，按需求选用两种方式。

Flash 存储器进行修改操作时，CPU 处于暂停状态，操作完成后，CPU 继续运行指令；

17.4.1 字写

Flash 存储器的字写操作是通过使能 EECON1 控制位 WR 实现的；用户首先应选择存储器的类型，再将该单元的地址写入 EEADR 和 EEADRH 寄存器并将数据写入 EEDAT 和 EEDATH 寄存器，然后将 EECON1 中的 WREN 位置 1 以使能写操作，向 EECON2 写入特定序列后将 EECON1 控制位 WR 置 1 启动写操作。在该代码段中应禁止中断，避免写操作被中断。

在不更新存储器数据时，用户应该始终保持 WREN 位清零，这种机制可防止由于代码执行错误（即程序出现异常跑飞）导致误写 Flash 存储器。一个写过程启动后，将 WREN 位清零将不会影响此写周期。

注：将 WR 位置 1 后需要等待 6 条 NOP 指令后才能进行其他操作。

写数据存储器流程示例：

1. 设置 EECON1.EEPGD 选择数据存储器；
2. 将目标地址写入 EEADR 寄存器；
3. 将数据写入 EEDAT 和 EEDATH 寄存器；
4. 将 EECON1.WREN 置 1，允许写周期；
5. 关闭中断；
6. 向 EECON2 写入特定的序列（先写 0x55，再写 0xAA）；
7. 将 EECON1.WR 置 1，启动写操作（CPU 将暂停至操作完成）；
8. 等待 6 条 NOP 指令后，判断等待 WR 是否清零；
9. 将 EECON1.WREN 清零；
10. 恢复中断；
11. 读取 EECON1.WRERR 判断是否出错，出错需再次执行写操作。

写程序存储器流程示例：

1. 设置 EECON1 控制位 EEPGD 选择程序存储器；
2. 将目标地址写入 EEADR 和 EEADRH 寄存器；
3. 将数据写入 EEDAT 和 EEDATH 寄存器；
4. 将 EECON1 控制位 WREN 置 1，允许写周期；
5. 关闭中断；
6. 向 EECON2 写入特定的序列（先写 0x69，再写 0x96）；
7. 将 EECON1 控制位 WR 置 1，启动写操作（CPU 将暂停至操作完成）；
8. 等待 6 条 NOP 指令后，判断等待 WR 是否清零；
9. 将 EECON1 控制位 WREN 清零；
10. 恢复中断；
11. 读取 EECON1 标志位 WRERR 判断是否出错，出错需再次执行写操作。

注：当系统配置中的程序存储器写保护位有效时，对该程序存储器区间无法启动写操作。

17.4.2 页操作

Flash 存储器的页操作有 4 种：清缓存、写缓存、页擦除和页烧写；在执行页操作前，需要选定存储器的类型和向 EECON2 寄存器写入特定序列以使能 Flash 存储器对应空间的修改操作；页操作时 CPU 处于暂停状态，操作完成后，CPU 继续运行指令。页操作的时间是由内部定时控制的。

17.4.2.1 清缓存

Flash 存储器的清缓存操作由硬件自动启动，有以下两种方式产生清缓存操作：

1. 将 EECON3.PAGE_MODE 置 1 使能页操作模式，选择操作存储器的类型，然后向 EECON2 写入特定序列后，使能 EECON3 控制位 HVPLEN 以使能缓存，此时硬件会自动启动清缓存操作，并使能硬件计数器；

注：将 HVPLEN 位置 1 后需要等待 6 条 NOP 指令后才能进行其他操作。

2. 保持 EECON3 控制位 HVPLEN 为 1，在 Flash 存储器页擦除或页烧写结束前，硬件均会自动启动清缓存操作和重置硬件计数器。

17.4.2.2 写缓存

Flash 存储器的写缓存操作是在缓存使能时（HVPLEN 为 1），由写 EEDAT 寄存器启动；此时写 EEDAT 寄存器会将 EEDATH 和 EEDAT 寄存器内的数据更新至硬件计数器对应地址的缓存空间，完成后硬件计数器自动加 1，重复将 16 个缓存空间（0x0~0xf）写满，等待页擦除或页烧写启动。在缓存写满后，在清缓存前不得再往缓存中写入任意数据，否则将可能导致缓存数据异常。

注：写 EEDAT 寄存器后需要等待 6 条 NOP 指令后才能进行其他操作。

17.4.2.3 页擦除

Flash 存储器的页擦除操作是在清缓存和写缓存后，由 EEADRH<4:0>和 EEADR<7:4>组合选择页，通过 EECON3 选择位 OP 选择擦除，并使能 EECON3 控制位 START 启动，擦除结束后 START 自动清零。页擦除结束前，硬件会自动启动清缓存操作和重置硬件计数器。

注：将 START 位置 1 后需要等待 6 条 NOP 指令后才能进行其他操作。

17.4.2.4 页烧写

Flash 存储器的页烧写操作是，在页擦除和写缓存后，由 EEADRH<4:0>和 EEADR<7:4>组合选择页，通过 EECON3 选择位 OP 选择烧写，并使能 EECON3 控制位 START 启动，烧写结束后 START 自动清零。页烧写结束前，硬件会自动启动清缓存操作和重置硬件计数器。

注：将 START 位置 1 后需要等待 6 条 NOP 指令后才能进行其他操作。

页操作程序存储器的流程示例：

1. 将 EECON3.PAGE_MODE 置 1，使能页操作模式；
2. 设置 EECON1 控制位 EEPGD 选择程序存储器；
3. 关闭中断；
4. 向 EECON2 写入特定的序列（先写 0x69，再写 0x96）；
5. 使能 EECON3.HVPLEN，硬件自动清除缓存并使能硬件计数器；
6. 等待 6 条 NOP 指令
7. 写 EEADRH/EEADR 选择页；
8. EECON3.OP[1:0]选择擦除操作；
9. 先写 EEDATH，再写 EEDAT 启动写缓存；
10. 等待 6 条 NOP 指令；
11. 重复第 9~10 步，将依次写满 16 个缓存空间；
12. 写 EECON3.START 启动擦除；
13. 等待 6 个 NOP 指令后，判断等待 START 是否清零；
14. EECON3.OP[1:0]选择烧写操作
15. 先写 EEDATH，再写 EEDAT 启动写缓存；
16. 等待 6 条 NOP 指令；
17. 重复第 15~16 步，将依次写满 16 个缓存空间；
18. 写 EECON3.START 启动烧写；
19. 等待 6 个 NOP 指令后，判断等待 START 是否清零；
20. 清零 EECON3.OP[1:0];
21. 清零 EECON3.HVPLEN；
22. 恢复中断。

注：当系统配置中的程序存储器写保护位有效时，对该程序存储器区间无法启动写操作。

17.5 Flash 存储器操作注意事项

17.5.1 关于 Flash 存储器的操作时间

Flash 存储器的操作时间是由内部定时器控制的，具体时间如下表所示：

操作类型	定时器时钟	操作时间	单位
清缓存	F8M=8MHz	0.75	us
写缓存（单次）		1.25	us
页擦除		2.82	ms
页烧写		1.80	ms
字写		4.62	ms

由上述时间关系可以看出，字写和单次页写所需要的总时间几乎是一样的。

17.5.2 写校验

根据具体的应用，一般要求进行软件写校验。

17.5.3 避免误写的保护

有些情况下，用户可能不希望向 Flash 存储器写入数据。为防止发生误写，芯片内嵌了各种保护机制。上电时清零 WREN 位。而且，上电延时定时器（延迟时间为 16ms）会防止对 Flash 存储器执行写操作。

写操作的启动序列以及 WREN 位将共同防止在以下情况下发生误写操作：

- 欠压
- 电源毛刺
- 软件故障

18. LVD 低电压检测

芯片具有低电压检测功能，可以用于监测电源电压，如果电源电压低于设定的值，可以产生一个中断信号；程序可实时读取 LVD 输出标志位。

18.1 LVD 相关的寄存器

有 1 个寄存器与 LVD 模块相关。

LVD 控制寄存器 LVDCON

9FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LVDCON	LVD_RES	—	—	—	LVD_SEL[2:0]			LVDEN
R/W	R	—	—	—	R/W	R/W	R/W	R/W
复位值	X	—	—	—	0	0	0	0

Bit7	LVD_RES:	LVD 输出结果
	0=	VDD>设定的 LVD 电压
	1=	VDD<设定的 LVD 电压
Bit6~Bit4	未用	
Bit3~Bit1	LVD_SEL[2:0]:	LVD 电压选择
	000=	保留
	001=	保留
	010=	2.7V
	011=	3.0V
	100=	3.3V
	101=	3.7V
	110=	4.0V
	111=	4.3V
Bit0	LVDEN:	LVD 使能位
	0=	禁止
	1=	使能

18.2 LVD 操作

通过设定 LVDCON 寄存器中的 LVD 电压值，使能 LVDEN 之后，当电源电压低于设定的电压值，LVDCON 寄存器中的 LVD_RES 位被置高。当 LVD 模块使能后，需要延时 10us 的时间才能够读取 LVD_RES 位，因为内部做了滤波处理，以减少在 VLVD 电压值附近时，LVD 输出结果的频繁波动。

LVD 模块有自己的中断标志位，当设定好相关的中断使能位，电源电压低于设定的电压值时，会产生 LVD 中断，中断标志位 LVDIF 将被置 1，中断产生。LVD 也可以用于中断唤醒模式。

19. 触摸按键

19.1 触摸按键模块概述

触摸检测模块是为实现人体触摸接口而设计的集成电路。可替代机械式轻触按键，实现防水防尘、密封隔离、坚固美观的操作接口。

技术参数：

- 1-26 个按键可选，所有 I/O 均可以配置成触摸通道
- 无需外部触摸电容
- 高抗干扰性能，可轻松通过静态 10V，动态 3V 传导测试

19.2 触摸模块使用注意事项

- 触摸按键检测部分的地线应该单独连接成一个独立的地，再有一个点连接到整机的共地。
- 避免高压、大电流、高频操作的主板与触摸电路板上下重迭安置。如无法避免，应尽量远离高压大电流的期间区域或在主板上加屏蔽。
- 感应盘到触摸芯片的连线尽量短和细，如果 PCB 工艺允许尽量采用 0.1mm 的线宽。
- 感应盘到触摸芯片的连线不要跨越强干扰、高频的信号线。
- 感应盘到触摸芯片的连线周围 0.5mm 不要走其它信号线。

20. 指令

汇编指令总共包括 5 类：算术运算、逻辑运算、数据传送运算、布尔操作和程序分支指令，这些指令全部都与标准 8051 兼容。

20.1 符号说明

符号	说明
Rn	工作寄存器 R0-R7
Direct	内部数据存储器 RAM 的单元地址（00H-FFH）或特殊功能寄存器 SFR 中的地址
@Ri	间接寻址寄存器（@R0 或 @R1）
#data	8 位二进制常数
#data16	在指令中的 16 位二进制常数
Bit	内部数据存储器 RAM 或特殊功能寄存器 SFR 中的位地址
Addr16	16 位地址，地址范围 0-64KB 地址空间
Addr11	11 位地址，地址范围 0-2KB 地址空间
Rel	相对地址
A	累加器

20.2 指令一览表

助记符		描述
运算类		
ADD	A,Rn	累加器加寄存器
ADD	A,direct	累加器加直接寻址单元
ADD	A,@Ri	累加器加间接寻址 RAM
ADD	A,#data	累加器加立即数
ADDC	A,Rn	累加器加寄存器和进位标志
ADDC	A,direct	累加器加直接寻址单元和进位标志
ADDC	A,@Ri	累加器加间接寻址 RAM 和进位标志
ADDC	A,#data	累加器加立即数和进位标志
SUBB	A,Rn	累加器减寄存器和进位标志
SUBB	A,direct	累加器减直接寻址单元和进位标志
SUBB	A,@Ri	累加器减间接寻址 RAM 和进位标志
SUBB	A,#data	累加器减立即数和进位标志
INC	A	累加器加 1
INC	Rn	寄存器加 1
INC	direct	直接寻址单元加 1
INC	@Ri	间接寻址 RAM 加 1
INC	DPTR	数据指针加 1
DEC	A	累加器减 1
DEC	Rn	寄存器减 1
DEC	direct	直接寻址单元减 1
DEC	@Ri	间接寻址 RAM 减 1
MUL	A,B	累加器乘寄存器 B
DIV	A,B	累加器除以寄存器 B
DA	A	十进制调整
逻辑运算类		
ANL	A,Rn	累加器与寄存器
ANL	A,direct	累加器与直接寻址单元
ANL	A,@Ri	累加器与间接寻址 RAM
ANL	A,#data	累加器与立即数
ANL	direct,A	直接寻址单元与累加器
ANL	direct,#data	直接寻址单元与立即数
ORL	A,Rn	累加器或寄存器
ORL	A, direct	累加器或直接寻址单元
ORL	A,@Ri	累加器或间接寻址 RAM
ORL	A, #data	累加器或立即数
ORL	direct,A	直接寻址单元或累加器
ORL	direct,#data	直接寻址单元或立即数
XRL	A,Rn	累加器异或寄存器
XRL	A,direct	累加器异或直接寻址单元
XRL	A,@Ri	累加器异或间接寻址 RAM
XRL	A,#data	累加器异或立即数
XRL	direct,A	直接寻址单元异或累加器
XRL	direct,#data	直接寻址单元异或立即数
CLR	A	累加器清 0
CPL	A	累加器取反

助记符		描述
RL	A	累加器左循环移位
RLC	A	累加器连进位标志左循环移位
RR	A	累加器右循环移位
RRC	A	累加器连进位标志右循环移位
SWAP	A	累加器高 4 位与低 4 位交换
数据传输类		
MOV	A,Rn	寄存器传送到累加器
MOV	A,direct	直接寻址单元传送到累加器
MOV	A,@Ri	间接寻址 RAM 送累加器
MOV	A,#data	立即数送累加器
MOV	Rn,A	累加器送寄存器
MOV	Rn,direct	直接寻址单元送寄存器
MOV	Rn,#data	立即数送寄存器
MOV	direct,A	累加器送直接寻址单元
MOV	direct,Rn	寄存器送直接寻址单元
MOV	direct1,direct2	直接地址单元传送到直接寻址单元
MOV	direct,@Ri	间接寻址 RAM 送直接寻址单元
MOV	direct,#data	立即数送直接寻址单元
MOV	@Ri,A	累加器送间接寻址 RAM
MOV	@Ri,direct	直接寻址单元送间接寻址 RAM
MOV	@Ri,#data	立即数送间接寻址 RAM
MOV	DPTR,#data16	16 位立即数送数据指针
MOVC	A,@A+DPTR	查表数据送累加器（DPTR 为基址）
MOVC	A,@A+PC	查表数据送累加器（PC 为基址）
MOVX	A,@Ri	外部 RAM 单元送累加器（8 位地址）
MOVX	A,@DPTR	外部 RAM 单元送累加器（16 位地址）
MOVX	@Ri,A	累加器送外部 RAM 单元（8 位地址）
MOVX	@DPTR,A	累加器送外部 RAM 单元（16 位地址）
PUSH	direct	直接寻址单元压入栈顶
POP	direct	栈顶弹出直接寻址单元
XCH	A,Rn	累加器与寄存器交换
XCH	A, direct	累加器与直接寻址单元 RAM 交换
XCH	A,@Ri	累加器与间接寻址单元 RAM 交换
XCHD	A,@Ri	累加器与间接寻址单元 RAM 交换低 4 位
布尔运算类		
CLR	C	C 清零
CLR	bit	直接寻址位清零
SETB	C	C 置位
SETB	bit	直接寻址位置位
CPL	C	C 取反
CPL	bit	直接寻址位取反
ANL	C,bit	C 逻辑与直接寻址位
ANL	C,/bit	C 逻辑与直接寻址位的反
ORL	C,bit	C 逻辑或直接寻址位
ORL	C,/bit	C 逻辑或直接寻址位的反
MOV	C,bit	直接寻址位送 C
MOV	bit,C	C 送直接寻址位
程序跳转类		

助记符		描述
ACALL	addr11	2K 地址范围内绝对调用
LCALL	addr16	64K 地址范围内长调用
RET		子程序返回
RETI		中断返回
AJMP	addr11	2K 地址范围内绝对转移
LJMP	addr16	64K 地址范围内长转移
SJMP	rel	相对短转移
JMP	@A+DPTR	相对长转移
JZ	rel	累加器为 0 转移
JNZ	rel	累加器不为 0 转移
JC	rel	C 为 1 转移
JNC	rel	C 为 0 转移
JB	bit,rel	直接寻址位为 1 转移
JNB	bit,rel	直接寻址位为 0 转移
JBC	bit,rel	直接寻址位为 1 转移，并清该位
CJNE	A,direct,rel	累加器与直接寻址单元不等转移
CJNE	A,#data,rel	累加器与立即数不等转移
CJNE	Rn,#data,rel	寄存器与立即数不等转移
CJNE	@Ri,#data,rel	间接寻址单元 RAM 与立即数不等转移
DJNZ	Rn,rel	寄存器减 1 不为 0 转移
DJNZ	direct,rel	直接寻址单元减 1 不为 0 转移
NOP		空指令
读取—修改—写入指令（Read-Modify-Write）		
ANL		逻辑（ANL direct, A 与 ANL direct, #data)
ORL		逻辑或(ORL direct, A 与 ORL direct, #data)
XRL		逻辑异或（XRL direct, A 与 XRL direct, #data)
JBC		直接寻址位为 1 转移，并清该位（JBC bit, rel)
CPL		取反（CPL bit)
INC		加 1 (INC direct)
DEC		减 1. (DEC direct)
DJNZ		减 1 不为 0 转移(DJNZ direct, rel)
MOV	bit,C	C 送直接寻址位
CLR	bit	直接寻址位清零
SETB	bit	直接寻址位置位

21. 版本修订说明

版本号	时间	修改内容
V0.0.0	2025 年 2 月	初始内部版本
V0.1.0	2025 年 2 月	格式优化
V0.1.1	2025 年 2 月	1) 订正 PWM 里计算公式中的 Tosc 为 THIS 2) 17.1 章节注释处的“程序 EEPROM”改为“Flash 存储器”
V0.1.2	2025 年 5 月	1) 订正异步模式下的波特率表 2) 修改 6.3.4/9.9 章节内容 3) 7.3.3/8.3/10.3/12.8 章节增加备注
V0.1.3	2025 年 5 月	增加 12.6、12.7 章节的内容
V0.1.4	2025 年 7 月	1) 第 17 章节里增加字读、字写、清缓存、写缓存、页擦除、页写等操作时需要等待 6 条 NOP 指令的说明 2) 第 6 章节删除 PORTA、PORTB、PORTC、PORTD 端口处理程序 3) 增加 13.1.1.6 章节里的串口异步背靠背发送注意事项及范例程序
V0.1.5	2025 年 7 月	1） 修正 TIMER1 外部时钟源注意事项描述里的错别字，“技数”改成“计数”。 2） 5.2 章节里增加从空闲模式唤醒时 CPU 和外设时钟源暂停的说明，并修改了 5.3.2 章节表格里休眠唤醒等待时间计算公式 3） 修改 2.1.1 章节中 FLASH 空间结构图