CMS8H341x 用户手册

增强型闪存8位CMOS单片机

Rev. 0.9.1

目录

1. 产品概述

1.1 功能特性

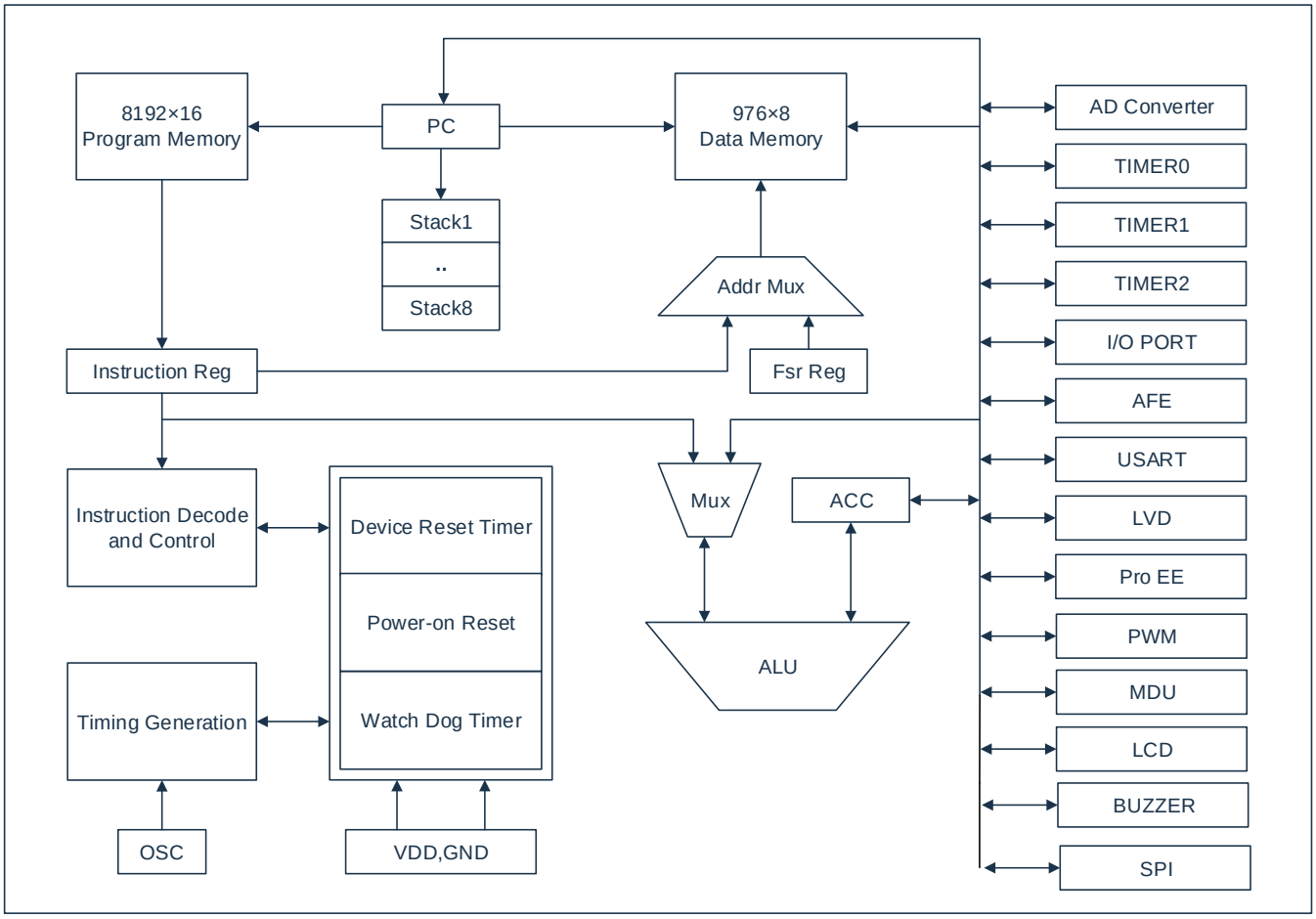
- 内存
  - ROM：8Kx16
  - 通用 RAM：976x8
- 8 级堆栈缓存器
- 简洁实用的指令系统（72 条指令）
- 间接查表功能
- 内置 WDT 定时器
- 内置低压侦测电路（侦测电压 2.2V）
- 中断源
  - 3 个定时中断
  - RA/RB 口电平变化中断
  - 其它外设中断
- 定时器
  - 8 位定时器 TIMER0，TIMER2
  - 16 位定时器 TIMER1
- 内置 128 字节程序 EEPROM
  - 可重复擦写 1 万次
- 内置 USART 通信模块
  - 支持同步主从模式和异步全双工模式
  - 支持 9600~115200 波特率
- 内置 BUZZER 蜂鸣器输出模块
- AFE 模块
  - 内置 24Bit Sigma-Delta ADC
  - 内置人体阻抗测量模块 BIM
- 内置硬件乘除法器
  - 支持 24*24 硬件乘法器
  - 支持 48/24 硬件除法器
- 工作电压范围：2.2V~4.5V@24MHz
  工作温度范围：-40℃~85℃
- 一种振荡方式
  - 内部 RC 振荡：设计频率 24MHz
- 指令周期（单指令或双指令）
- 内置 LVD 模块
  - 支持多种电压检测模式
  - 支持多种电压可选
- 高精度 12 位 ADC
  - 内建高精度 1.2V 基准电压
  - ±1.5% @VDD=2.2V~4.5V T_A=25℃
  - ±2% @VDD=2.2V~4.5V T_A= -40℃~85℃
  - 可选择内部参考源：2.4V/2.6V/3.0V/3.3V
  - 最快转换速率可达 250ksps
- 内置 LED 驱动模块
  - PORTA/PORTB 都可选 SEG/ COM 输出
  - PORTA/PORTB 驱动电流可选
- PWM 模块
  - 10 位 PWM 精度
  - 2 路 PWM 可设置独立周期和占空比
- 内置 LCD 驱动模块
  - 支持 23SEG*4COM 或者 21SEG*6COM
  - 可选 1/2、1/3 偏压
  - 可选 1/2、1/3、1/4、1/5、1/6 占空比

型号说明

PRODUCT	ROM	RAM	Pro EE	I/O	24bit ADC	12bit ADC	USART	PWM	BUZ	PACKAGE
CMS8H3415	8Kx16	976x8	128x8	16	1	12Bitx6	1	1	1	QFN32
CMS8H3416	8Kx16	976x8	128x8	24	1	12Bitx8	1	2	1	LQFP48

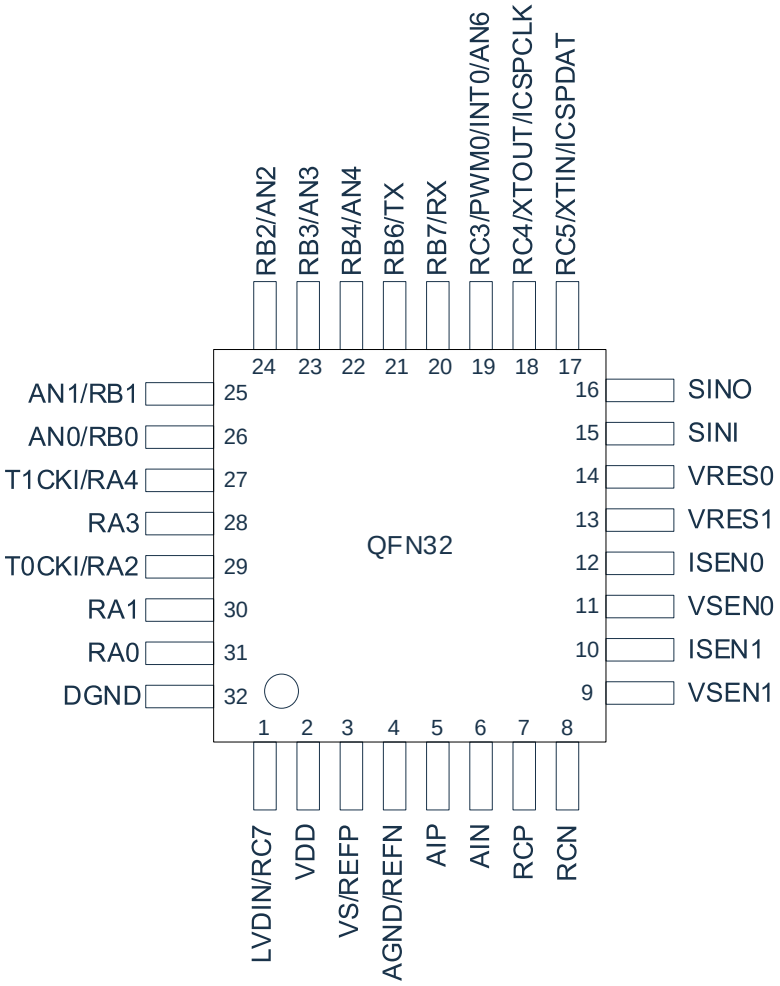
注：ROM----程序存储器　　Pro EE----程序EEPROM

1.2 系统结构框图

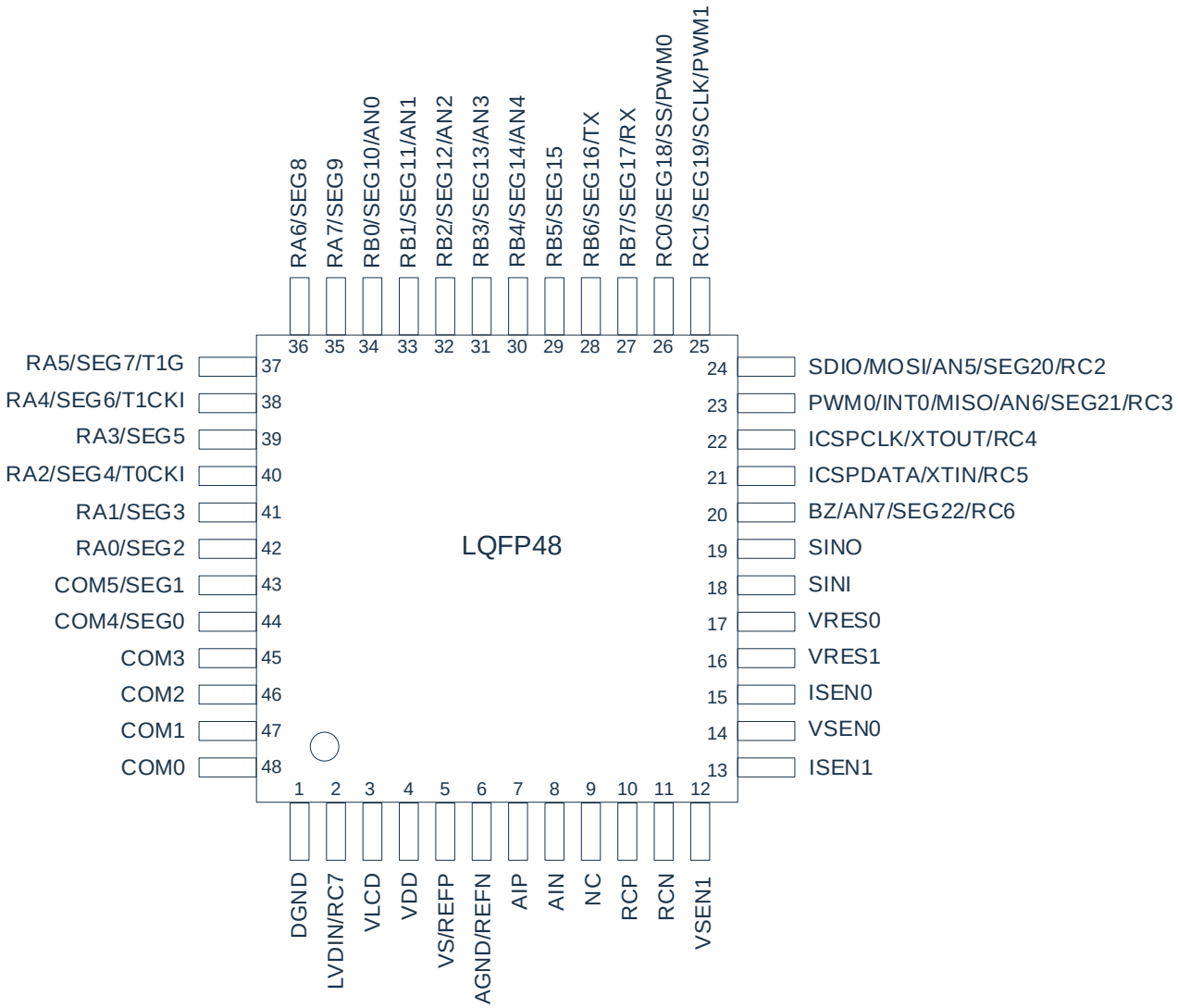

8192×16
Program Memory
PC
976×8
Data Memory
AD Converter
TIMER0
TIMER1
TIMER2
I/O PORT
AFE
USART
LVD
Pro EE
PWM
MDU
LCD
BUZZER
SPI
Stack1
..
Stack8
Addr Mux
Fsr Reg
Instruction Reg
Instruction Decode
and Control
Device Reset Timer
Power-on Reset
Watch Dog Timer
Timing Generation
OSC
VDD,GND
Mux
ACC
ALU

1.3 管脚分布

1.3.1 CMS8H3415 引脚图



1.3.2 CMS8H3416 引脚图



CMS8H341x 引脚说明：

管脚名称	IO 类型	管脚说明
VDD	P	电源电压输入脚
RA0-RA7	I/O	可编程为输入脚，推挽输出脚，带上拉、下拉电阻功能、电平变化中断功能
RB0-RB7	I/O	可编程为输入脚，推挽输出脚，带上拉、下拉电阻功能、电平变化中断功能
RC0-RC6	I/O	可编程为输入脚，推挽输出脚，带上拉、下拉电阻功能
ICSPCLK/ICSPDAT	I/O	编程时钟/数据脚
AN0-AN7	I	12 位 ADC 输入脚
T0CKI	I	TIMER0 外部时钟输入脚
T1CKI	I	TIMER1 外部时钟输入脚
T1G	I	TIMER1 门控输入脚
TX/CK	I/O	USART 异步发送/同步时钟口
RX/DT	I/O	USART 异步接收/同步数据口
PWM0-PWM1	O	PWM 输出引脚
INT0	I	外部中断输入
BUZ	O	蜂鸣器频率输出脚
XTIN/XTOUT	I/O	32.768KHz 晶振输入/输出引脚
VLCD	O	升压泵 CPR 输出口
SEG22-SEG0	O	LCD 驱动端输出
COM0-COM5	O	LCD 驱动公共端
LVDIN	I	LVD 输入脚
AIP, AIN	I	AFE 的差分输入口
VS/REFP	O	AFE 的参考电压输入正端，与 LDO 输出在片外短接
RCP	O	整流输出正端
RCN	O	整流输出负端
VSEN1	I	电压检测电极输入通道 1
ISEN1	O	激励电流电极通道 1
VSEN0	I	电压检测电极输入通道 0
ISEN0	O	激励电流电极通道 0
VRES1	I	参考电阻 1 接入通道
VRES0	I	参考电阻 0 接入通道
SINI	I	正弦激励输入端
SINO	O	正弦激励输出端
AGND/REFN	P	AFE 的参考电压输入负端，与模拟地在片外短接
DGND	P	数字地
NC	/	悬空，无连接
MISO	I/O	SPI 主控数据输入、从动数据输出引脚
MOSI	I/O	SPI 主控数据输出、从动数据输入引脚
SS	I	SPI 从动使能引脚
SCLK	I/O	SPI 时钟引脚

1.4 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，用户不能通过程序访问及操作。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. ICSP（仿真端口功能选择）
   - ICSP　　ICSPCLK、DAT 口一直保持为仿真口，所有功能均不能使用
   - NORMAL　　ICSPCLK、DAT 口为普通功能口
4. URSECPR(ROM 自写分区保护)
   - 11_1111_1111_1111(默认)

1.5 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。

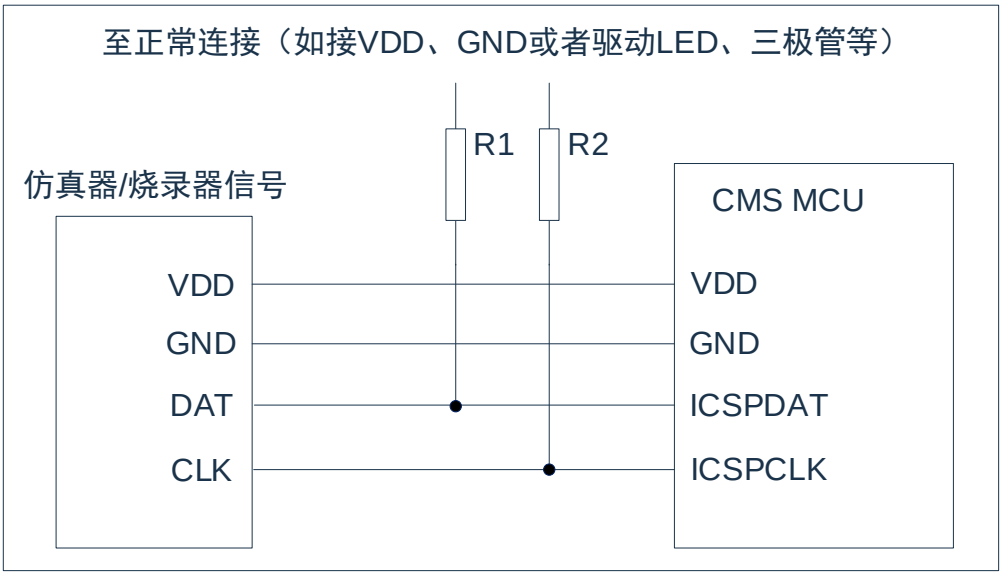


图 1-1：典型的在线串行编程连接方法

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。

1.6 集成开发环境

- 片上调试（OCD），ISP；
- 4 个硬件断点；
- 软件复位，暂停，单步，运行等。

2. 中央处理器（CPU）

2.1 内存

2.1.1 程序内存

CMS341x 程序存储器空间

ROM:8K

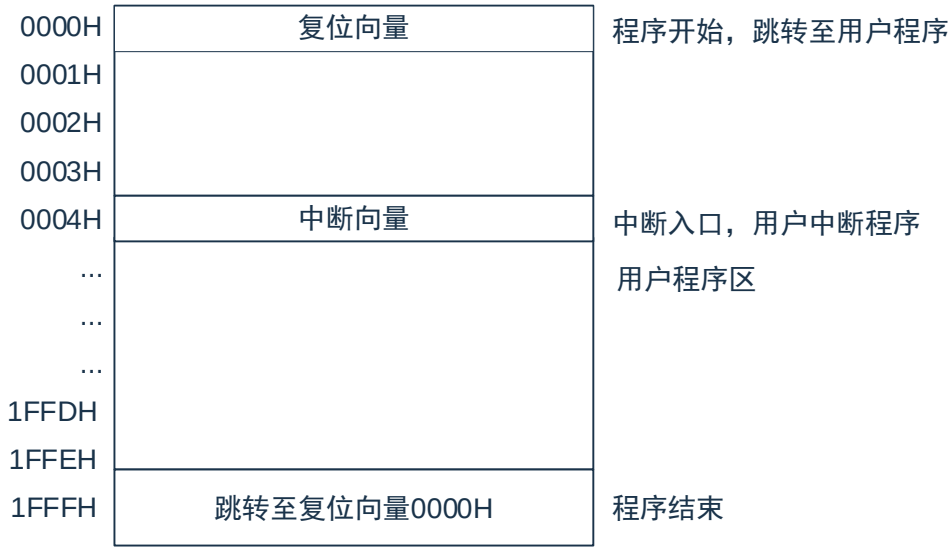


图 2-1：ROM空间示意图

2.1.1.1 复位向量（0000H）

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 STATUS 寄存器中的 PD 和 TO 标志位的内容可以判断系统复位方式。下面一段程序演示了如何定义 FLASH 中的复位向量。

例：定义复位向量

```
        ORG     0000H       ;系统复位向量
        JP      START
        ORG     0010H       ;用户程序起始
START:
        …                   ;用户程序
        …
        END                 ;程序结束
```

2.1.1.2 中断向量

中断向量地址为 0004H。一旦有中断响应，程序计数器 PC 的当前值就会存入堆栈缓存器，并且内核寄存器的值也会存入相应的影子寄存器中并跳转到 0004H 开始执行中断服务程序。所有中断都会进入 0004H 这个中断向量，具体执行哪个中断将由用户根据中断请求标志位寄存器的位决定。下面的示例程序说明了如何编写中断服务程序。

例：定义中断向量，中断程序放在用户程序之后

```
        ORG     0000H       ;系统复位向量
        JP      START
        ORG     0004H       ;用户程序起始
INT_START:
        CALL    PUSH        ;保存 ACC 跟 STATUS
        ...                 ;用户中断程序
        ...
INT_BACK:
        CALL    POP         ;返回 ACC 跟 STATUS
        RETI                ;中断返回
START:
        ...                 ;用户程序
        ...
        END                 ;程序结束
```

2.1.1.3 跳转表

跳转表能够实现多地址跳转功能。由于 PCL 和 ACC 的值相加即可得到新的 PCL，因此，可以通过对 PCL 加上不同的 ACC 值来实现多地址跳转。ACC 值若为 n，PCL+ACC 即表示当前地址加 n，执行完当前指令后 PCL 值还会自加 1，可参考以下范例。如果 PCL+ACC 后发生溢出，PC 不会自动进位，故编写程序时应注意。这样，用户就可以通过修改 ACC 的值轻松实现多地址的跳转。

PCLATH 为 PC 高位缓冲寄存器，对 PCL 操作时，必须先对 PCLATH 进行赋值。

例：正确的多地址跳转程序示例

```
FLASH 地址
                LDIA        01H
                LD          PCLATH,A        ;必须对 PCLATH 进行赋值
                …
0110H:          ADDR        PCL             ;ACC+PCL
0111H:          JP          LOOP1           ;ACC=0，跳转至 LOOP1
0112H:          JP          LOOP2           ;ACC=1，跳转至 LOOP2
0113H:          JP          LOOP3           ;ACC=2，跳转至 LOOP3
0114H:          JP          LOOP4           ;ACC=3，跳转至 LOOP4
0115H:          JP          LOOP5           ;ACC=4，跳转至 LOOP5
0116H:          JP          LOOP6           ;ACC=5，跳转至 LOOP6
```

例：错误的多地址跳转程序示例

```
FLASH 地址
                CLR         PCLATH
                …
00FCH:          ADDR        PCL             ;ACC+PCL
00FDH:          JP          LOOP1           ;ACC=0，跳转至 LOOP1
00FEH:          JP          LOOP2           ;ACC=1，跳转至 LOOP2
00FFH:          JP          LOOP3           ;ACC=2，跳转至 LOOP3
0100H:          JP          LOOP4           ;ACC=3，跳转至 0000H 地址
0101H:          JP          LOOP5           ;ACC=4，跳转至 0001H 地址
0102H:          JP          LOOP6           ;ACC=5，跳转至 0002H 地址
```

注：由于 PCL 溢出不会自动向高位进位，故在利用 PCL 作多地址跳转时，需要注意该段程序一定不能放在 FLASH 空间的分页处。

2.1.2 数据存储器

CMS341x 数据存储器列表

	地址		地址		地址		地址		地址
INDF0	00H	INDF0	80H	INDF0	100H	INDF0	180H	INDF0	200H
INDF1	01H	INDF1	81H	INDF1	101H	INDF1	181H	INDF1	201H
PCL	02H	PCL	82H	PCL	102H	PCL	182H	PCL	202H
STATUS	03H	STATUS	83H	STATUS	103H	STATUS	183H	STATUS	203H
FSR0L	04H	FSR0L	84H	FSR0L	104H	FSR0L	184H	FSR0L	204H
FSR0H	05H	FSR0H	85H	FSR0H	105H	FSR0H	185H	FSR0H	205H
FSR1L	06H	FSR1L	86H	FSR1L	106H	FSR1L	186H	FSR1L	206H
FSR1H	07H	FSR1H	87H	FSR1H	107H	FSR1H	187H	FSR1H	207H
BSR	08H	BSR	88H	BSR	108H	BSR	188H	BSR	208H
WREG	09H	WREG	89H	WREG	109H	WREG	189H	WREG	209H
PCLATH	0AH	PCLATH	8AH	PCLATH	10AH	PCLATH	18AH	PCLATH	20AH
INTCON	0BH	INTCON	8BH	INTCON	10BH	INTCON	18BH	INTCON	20BH
PORTA	0CH	TRISA	8CH	WPUA	10CH	ANSEL	18CH	MDUCON	20CH
PORTB	0DH	TRISB	8DH	WPUB	10DH	PAPI	18DH	MDUDT0	20DH
PORTC	0EH	TRISC	8EH	WPUC	10EH	PBPI	18EH	MDUDT1	20EH
PAWP	0FH	WPDC	8FH	WPDA	10FH	COMENA	18FH	MDUDT2	20FH
PBWP	10H		90H	WPDB	110H	COMENB	190H	MDUDT3	210H
PIR1	11H	PIE1	91H	EECON3	111H	SEGENA	191H	MDUDT4	211H
PIR2	12H	PIE2	92H	PWMCON0	112H	SEGENB	192H	MDUDT5	212H
PAWE	13H	TABLE_DATAH	93H	PWMCON1	113H	IOSR	193H	MDUDT6	213H
PBWE	14H	SPIBUF	94H	PWMTH	114H	BUZCON0	194H	MDUDT7	214H
TMR0	15H	SPICON	95H	PWM0TL	115H	BUZCON1	195H	MDUDT8	215H
OPTION_REG	16H	SPICON2	96H	PWM1TL	116H	AFECON1	196H	LCDCON0	216H
TMR1L	17H	RCSTA	97H	PWMD0L	117H	AFECON2	197H	LCDCON1	217H
TMR1H	18H	TXSTA	98H	PWMD1L	118H	AFECON3	198H	LCDADD	218H
T1CON	19H	SPBRG	99H	PWMD01H	119H	AFECON4	199H	LCDDATA	219H
TMR2	1AH	TXREG	9AH	EEDAT	11AH	AFERES0	19AH	COMEN	21AH
PR2	1BH	RCREG	9BH	EEADR	11BH	AFERES1	19BH	SEGEN0	21BH
T2CON	1CH	ADCON0	9CH	EEDATH	11CH	AFERES2	19CH	SEGEN1	21CH
WDTCON	1DH	ADCON1	9DH	EEADRH	11DH	BIMCON1	19DH	SEGEN2	21DH
OSCCON	1EH	ADRESL	9EH	EECON1	11EH	BIMCON2	19EH		21EH
LVDCON	1FH	ADRESH	9FH	EECON2	11FH		19FH		21FH
	20H		A0H		120H		1A0H		220H
通用寄存器 80 字节		通用寄存器 80 字节		通用寄存器 80 字节		通用寄存器 80 字节		通用寄存器 80 字节	
	6FH		EFH		16FH		1EFH		26FH
快速存储区 70H-7FH	70H -- 7FH	快速存储区 70H-7FH	F0H -- FFH	快速存储区 70H-7FH	170H -- 17FH	快速存储区 70H-7FH	1F0H -- 1FFH	快速存储区 70H-7FH	270H -- 27FH
BANK0		BANK1		BANK2		BANK3		BANK4	

	地址		地址		地址		地址		地址		地址		地址
INDF0	280H	INDF0	300H	INDF0	380H	INDF0	400H	INDF0	480H	INDF0	500H	INDF0	580H
INDF1	281H	INDF1	301H	INDF1	381H	INDF1	401H	INDF1	481H	INDF1	501H	INDF1	581H
PCL	282H	PCL	302H	PCL	382H	PCL	402H	PCL	482H	PCL	502H	PCL	582H
STATUS	283H	STATUS	303H	STATUS	383H	STATUS	403H	STATUS	483H	STATUS	503H	STATUS	583H
FSR0L	284H	FSR0L	304H	FSR0L	384H	FSR0L	404H	FSR0L	484H	FSR0L	504H	FSR0L	584H
FSR0H	285H	FSR0H	305H	FSR0H	385H	FSR0H	405H	FSR0H	485H	FSR0H	505H	FSR0H	585H
FSR1L	286H	FSR1L	306H	FSR1L	386H	FSR1L	406H	FSR1L	486H	FSR1L	506H	FSR1L	586H
FSR1H	287H	FSR1H	307H	FSR1H	387H	FSR1H	407H	FSR1H	487H	FSR1H	507H	FSR1H	587H
BSR	288H	BSR	308H	BSR	388H	BSR	408H	BSR	488H	BSR	508H	BSR	588H
WREG	289H	WREG	309H	WREG	389H	WREG	409H	WREG	489H	WREG	509H	WREG	589H
PCLATH	28AH	PCLATH	30AH	PCLATH	38AH	PCLATH	40AH	PCLATH	48AH	PCLATH	50AH	PCLATH	58AH
INTCON	28BH	INTCON	30BH	INTCON	38BH	INTCON	40BH	INTCON	48BH	INTCON	50BH	INTCON	58BH
	28CH		30CH		38CH		40CH		48CH		50CH		58CH
	28DH		30DH		38DH		40DH		48DH		50DH		58DH
	28EH		30EH		38EH		40EH		48EH		50EH		58EH
	28FH		30FH		38FH		40FH		48FH		50FH		58FH
	290H		310H		390H		410H		490H		510H		590H
	291H		311H		391H		411H		491H		511H		591H
	292H		312H		392H		412H		492H		512H		592H
	293H		313H		393H		413H		493H		513H		593H
	294H		314H		394H		414H		494H		514H		594H
	295H		315H		395H		415H		495H		515H		595H
	296H		316H		396H		416H		496H		516H		596H
	297H		317H		397H		417H		497H		517H		597H
	298H		318H		398H		418H		498H		518H		598H
	299H		319H		399H		419H		499H		519H		599H
	29AH		31AH		39AH		41AH		49AH		51AH		59AH
	29BH		31BH		39BH		41BH		49BH		51BH		59BH
	29CH		31CH		39CH		41CH		49CH		51CH		59CH
	29DH		31DH		39DH		41DH		49DH		51DH		59DH
	29EH		31EH		39EH		41EH		49EH		51EH		59EH
	29FH		31FH		39FH		41FH		49FH		51FH		59FH
	2A0H		320H		3A0H		420H		4A0H		520H		5A0H
通用寄存器 80 字节		通用寄存器 80 字节		通用寄存器 80 字节		通用寄存器 80 字节		通用寄存器 80 字节		通用寄存器 80 字节		通用寄存器 80 字节	
	2EFH		36FH		3EFH		46FH		4EFH		56FH		5EFH
快速存储区 70H-7FH	2F0H -- 2FFH	快速存储区 70H-7FH	370H -- 37FH	快速存储区 70H-7FH	3F0H -- 3FFH	快速存储区 70H-7FH	470H -- 47FH	快速存储区 70H-7FH	4F0H -- 4FFH	快速存储区 70H-7FH	570H -- 57FH	快速存储区 70H-7FH	5F0H -- 5FFH
BANK5		BANK6		BANK7		BANK8		BANK9		BANK10		BANK11	

	地址		地址		地址		地址		地址		地址		地址
INDF0	600H	INDF0	680H	INDF0	700H	INDF0	780H	INDF0	800H	INDF0	880H	INDF0	900H
INDF1	601H	INDF1	681H	INDF1	701H	INDF1	781H	INDF1	801H	INDF1	881H	INDF1	901H
PCL	602H	PCL	682H	PCL	702H	PCL	782H	PCL	802H	PCL	882H	PCL	902H
STATUS	603H	STATUS	683H	STATUS	703H	STATUS	783H	STATUS	803H	STATUS	883H	STATUS	903H
FSR0L	604H	FSR0L	684H	FSR0L	704H	FSR0L	784H	FSR0L	804H	FSR0L	884H	FSR0L	904H
FSR0H	605H	FSR0H	685H	FSR0H	705H	FSR0H	785H	FSR0H	805H	FSR0H	885H	FSR0H	905H
FSR1L	606H	FSR1L	686H	FSR1L	706H	FSR1L	786H	FSR1L	806H	FSR1L	886H	FSR1L	906H
FSR1H	607H	FSR1H	687H	FSR1H	707H	FSR1H	787H	FSR1H	807H	FSR1H	887H	FSR1H	907H
BSR	608H	BSR	688H	BSR	708H	BSR	788H	BSR	808H	BSR	888H	BSR	908H
WREG	609H	WREG	689H	WREG	709H	WREG	789H	WREG	809H	WREG	889H	WREG	909H
PCLATH	60AH	PCLATH	68AH	PCLATH	70AH	PCLATH	78AH	PCLATH	80AH	PCLATH	88AH	PCLATH	90AH
INTCON	60BH	INTCON	68BH	INTCON	70BH	INTCON	78BH	INTCON	80BH	INTCON	88BH	INTCON	90BH
	60CH		68CH		70CH		78CH		80CH		88CH		90CH
	60DH		68DH		70DH		78DH		80DH		88DH		90DH
	60EH		68EH		70EH		78EH		80EH		88EH		90EH
	60FH		68FH		70FH		78FH		80FH		88FH		90FH
	610H		690H		710H		790H		810H		890H		910H
	611H		691H		711H		791H		811H		891H		911H
	612H		692H		712H		792H		812H		892H		912H
	613H		693H		713H		793H		813H		893H		913H
	614H		694H		714H		794H		814H		894H		914H
	615H		695H		715H		795H		815H		895H		915H
	616H		696H		716H		796H		816H		896H		916H
	617H		697H		717H		797H		817H		897H		917H
	618H		698H		718H		798H		818H		898H		918H
	619H		699H		719H		799H		819H		899H		919H
	61AH		69AH		71AH		79AH		81AH		89AH		91AH
	61BH		69BH		71BH		79BH		81BH		89BH		91BH
	61CH		69CH		71CH		79CH		81CH		89CH		91CH
	61DH		69DH		71DH		79DH		81DH		89DH		91DH
	61EH		69EH		71EH		79EH		81EH		89EH		91EH
	61FH		69FH		71FH		79FH		81FH		89FH		91FH
未用	620H	未用	6A0H	未用	720H	未用	7A0H	未用	820H	未用	8A0H	未用	920H
	66FH		6EFH		76FH		7EFH		86FH		8EFH		96FH
快速存储区	670H	快速存储区	6F0H	快速存储区	770H	快速存储区	7F0H	快速存储区	870H	快速存储区	8F0H	快速存储区	970H
70H-7FH	--	70H-7FH	--	70H-7FH	--	70H-7FH	--	70H-7FH	--	70H-7FH	--	70H-7FH	--
	67FH		6FFH		77FH		7FFH		87FH		8FFH		97FH
BANK12		BANK13		BANK14		BANK15		BANK16		BANK17		BANK18	

地址		地址		地址		地址		地址		地址		地址	
INDF0	980H	INDF0	A00H	INDF0	A80H	INDF0	B00H	INDF0	B80H	INDF0	C00H	INDF0	C80H
INDF1	981H	INDF1	A01H	INDF1	A81H	INDF1	B01H	INDF1	B81H	INDF1	C01H	INDF1	C81H
PCL	982H	PCL	A02H	PCL	A82H	PCL	B02H	PCL	B82H	PCL	C02H	PCL	C82H
STATUS	983H	STATUS	A03H	STATUS	A83H	STATUS	B03H	STATUS	B83H	STATUS	C03H	STATUS	C83H
FSR0L	984H	FSR0L	A04H	FSR0L	A84H	FSR0L	B04H	FSR0L	B84H	FSR0L	C04H	FSR0L	C84H
FSR0H	985H	FSR0H	A05H	FSR0H	A85H	FSR0H	B05H	FSR0H	B85H	FSR0H	C05H	FSR0H	C85H
FSR1L	986H	FSR1L	A06H	FSR1L	A86H	FSR1L	B06H	FSR1L	B86H	FSR1L	C06H	FSR1L	C86H
FSR1H	987H	FSR1H	A07H	FSR1H	A87H	FSR1H	B07H	FSR1H	B87H	FSR1H	C07H	FSR1H	C87H
BSR	988H	BSR	A08H	BSR	A88H	BSR	B08H	BSR	B88H	BSR	C08H	BSR	C88H
WREG	989H	WREG	A09H	WREG	A89H	WREG	B09H	WREG	B89H	WREG	C09H	WREG	C89H
PCLATH	98AH	PCLATH	A0AH	PCLATH	A8AH	PCLATH	B0AH	PCLATH	B8AH	PCLATH	C0AH	PCLATH	C8AH
INTCON	98BH	INTCON	A0BH	INTCON	A8BH	INTCON	B0BH	INTCON	B8BH	INTCON	C0BH	INTCON	C8BH
	98CH		A0CH		A8CH		B0CH		B8CH		C0CH		C8CH
	98DH		A0DH		A8DH		B0DH		B8DH		C0DH		C8DH
	98EH		A0EH		A8EH		B0EH		B8EH		C0EH		C8EH
	98FH		A0FH		A8FH		B0FH		B8FH		C0FH		C8FH
	990H		A10H		A90H		B10H		B90H		C10H		C90H
	991H		A11H		A91H		B11H		B91H		C11H		C91H
	992H		A12H		A92H		B12H		B92H		C12H		C92H
	993H		A13H		A93H		B13H		B93H		C13H		C93H
	994H		A14H		A94H		B14H		B94H		C14H		C94H
	995H		A15H		A95H		B15H		B95H		C15H		C95H
	996H		A16H		A96H		B16H		B96H		C16H		C96H
	997H		A17H		A97H		B17H		B97H		C17H		C97H
	998H		A18H		A98H		B18H		B98H		C18H		C98H
	999H		A19H		A99H		B19H		B99H		C19H		C99H
	99AH		A1AH		A9AH		B1AH		B9AH		C1AH		C9AH
	99BH		A1BH		A9BH		B1BH		B9BH		C1BH		C9BH
	99CH		A1CH		A9CH		B1CH		B9CH		C1CH		C9CH
	99DH		A1DH		A9DH		B1DH		B9DH		C1DH		C9DH
	99EH		A1EH		A9EH		B1EH		B9EH		C1EH		C9EH
	99FH		A1FH		A9FH		B1FH		B9FH		C1FH		C9FH
未用	9A0H	未用	A20H	未用	AA0H	未用	B20H	未用	BA0H	未用	C20H	未用	CA0H
	9EFH		A6FH		AEFH		B6FH		BEFH		C6FH		CEFH
快速存储区	9F0H	快速存储区	A70H	快速存储区	AF0H	快速存储区	B70H	快速存储区	BF0H	快速存储区	C70H	快速存储区	CF0H
70H-7FH	--	70H-7FH	--	70H-7FH	--	70H-7FH	--	70H-7FH	--	70H-7FH	--	70H-7FH	--
	9FFH		A7FH		AFFH		B7FH		BFFH		C7FH		CFFH
BANK19		BANK20		BANK21		BANK22		BANK23		BANK24		BANK25	

	地址		地址		地址		地址		地址		地址
INDF0	D00H	INDF0	D80H	INDF0	E00H	INDF0	E80H	INDF0	F00H	INDF0	F80H
INDF1	D01H	INDF1	D81H	INDF1	E01H	INDF1	E81H	INDF1	F01H	INDF1	F81H
PCL	D02H	PCL	D82H	PCL	E02H	PCL	E82H	PCL	F02H	PCL	F82H
STATUS	D03H	STATUS	D83H	STATUS	E03H	STATUS	E83H	STATUS	F03H	STATUS	F83H
FSR0L	D04H	FSR0L	D84H	FSR0L	E04H	FSR0L	E84H	FSR0L	F04H	FSR0L	F84H
FSR0H	D05H	FSR0H	D85H	FSR0H	E05H	FSR0H	E85H	FSR0H	F05H	FSR0H	F85H
FSR1L	D06H	FSR1L	D86H	FSR1L	E06H	FSR1L	E86H	FSR1L	F06H	FSR1L	F86H
FSR1H	D07H	FSR1H	D87H	FSR1H	E07H	FSR1H	E87H	FSR1H	F07H	FSR1H	F87H
BSR	D08H	BSR	D88H	BSR	E08H	BSR	E88H	BSR	F08H	BSR	F88H
WREG	D09H	WREG	D89H	WREG	E09H	WREG	E89H	WREG	F09H	WREG	F89H
PCLATH	D0AH	PCLATH	D8AH	PCLATH	E0AH	PCLATH	E8AH	PCLATH	F0AH	PCLATH	F8AH
INTCON	D0BH	INTCON	D8BH	INTCON	E0BH	INTCON	E8BH	INTCON	F0BH	INTCON	F8BH
	D0CH		D8CH		E0CH		E8CH		F0CH	未用	F8CH
	D0DH		D8DH		E0DH		E8DH		F0DH		
	D0EH		D8EH		E0EH		E8EH		F0EH		
	D0FH		D8FH		E0FH		E8FH		F0FH		
	D10H		690H		E10H		E90H		F10H		
	D11H		D91H		E11H		E91H		F11H		
	D12H		D92H		E12H		E92H		F12H		
	D13H		D93H		E13H		E93H		F13H		
	D14H		D94H		E14H		E94H		F14H		
	D15H		D95H		E15H		E95H		F15H		
	D16H		D96H		E16H		E96H		F16H		
	D17H		D97H		E17H		E97H		F17H		
	D18H		D98H		E18H		E98H		F18H		
	D19H		D99H		E19H		E99H		F19H		
	D1AH		D9AH		E1AH		E9AH		F1AH		FE3H
	D1BH		D9BH		E1BH		E9BH		F1BH	STATUS_SHAD	FE4H
	D1CH		D9CH		E1CH		E9CH		F1CH	WREG_SHAD	FE5H
	D1DH		D9DH		E1DH		E9DH		F1DH	BSR_SHAD	FE6H
	D1EH		D9EH		E1EH		E9EH		F1EH	PCLATH_SHAD	FE7H
	D1FH		D9FH		E1FH		E9FH		F1FH	FSR0L_SHAD	FE8H
	D20H		DA0H		E20H		EA0H		F20H	FSR0H_SHAD	FE9H
										FSR1L_SHAD	FEAH
										FSR1H_SHAD	FEBH
未用		未用		未用		未用		未用		----	FECH
										STKPTR	FEDH
										TOSL	FEEH
	D6FH		DEFH		E6FH		EEFH		F6FH	TOSH	FEFH
快速存储区	D70H	快速存储区	DF0H	快速存储区	E70H	快速存储区	EF0H	快速存储区	F70H	快速存储区	FF0H
70H-7FH	--	70H-7FH	--	70H-7FH	--	70H-7FH	--	70H-7FH	--	70H-7FH	--
	D7FH		DFFH		E7FH		EFFH		F7FH		FFFH
BANK26		BANK27		BANK28		BANK29		BANK30		BANK31	

2.1.3 数据存储器构成

数据存储器可划分为 32 个存储区，每个存储区 128 个字节。每个存储区包含：

- 12 个内核寄存器
- 最多 20 个特殊功能寄存器（SFR）
- 最多 80 字节的通用 RAM（GPR）
- 16 个公共 RAM

可通过将存储区号写入存储区选择寄存器来选择有效存储区，未实现的存储器将读为未知值。所有的数据存储器都可以直接访问（通过文件寄存器指令）。

BANK 选择寄存器 BSR(08H)

08H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BSR				PAGE<2:0>			BSR1	BSR0
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~Bit2　　PAGE<2:0>:　页选择

000=　选择BANK0-BANK3

001=　选择BANK4-BANK7

010=　选择BANK8-BANK11

…

111=　选择BANK28-BANK31

Bit1~Bit0　　BSR<1:0>　页选择（仅C语言程序有效）

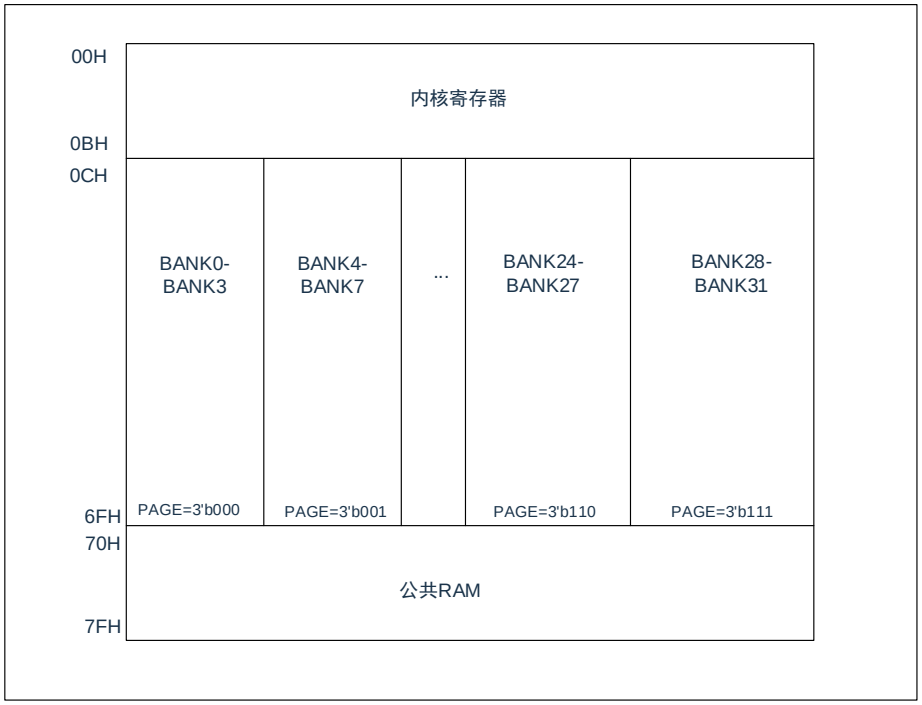


图 2-2：数据存储器（汇编语言程序）

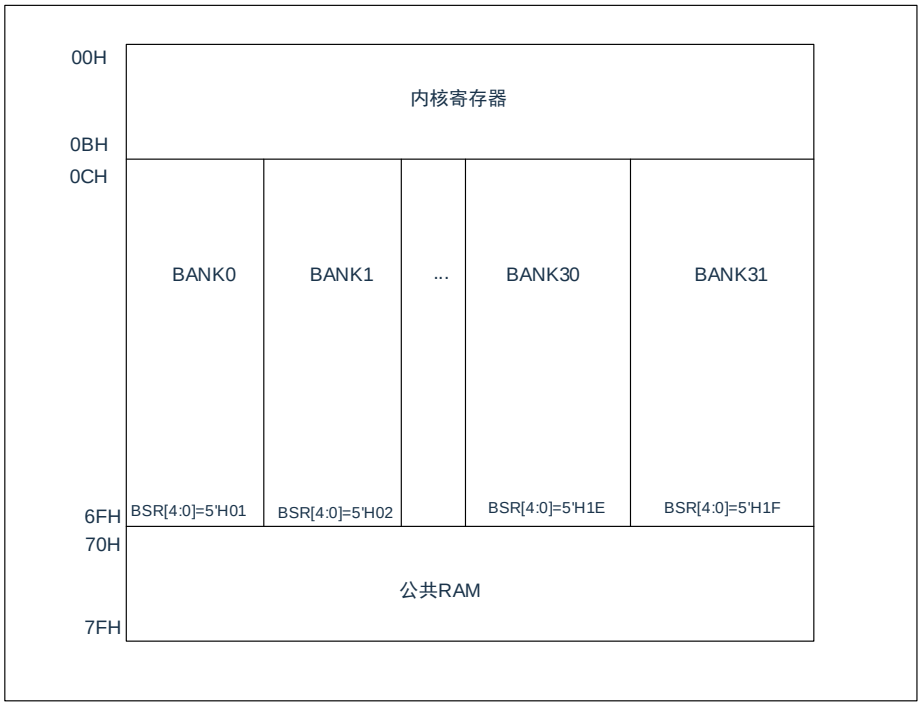


图 2-3：数据存储器（C 语言程序）

注：
1) 图 2-2，图 2-3 纵坐标指的是每个 BANK 的低 7 位地址偏移量；
2) 图 2-2 数据存储器仅对汇编程序有效，图 2-3 数据存储器仅对 C 语言程序有效；

每个 BANK 的地址分配	起始地址	结束地址
内核寄存器	00H	0BH
特殊功能寄存器（SFR）	0CH	1FH
通用 RAM（GPR）	20H	6FH
公共 RAM	70H	7FH

在直接寻址或者间接寻址模式下，内核寄存器地址和公共 RAM 地址不受 BSR 寄存器的控制；在直接寻址模式下，特殊功能寄存器地址和通用 RAM 地址受 BSR 寄存器的控制。

首先在汇编语言程序中，特殊功能寄存器地址和通用 RAM 地址受 PAGE（BSR<4:2>）控制的，直接寻址访问相应的 BANK 存储区地址之前需要正确设置 PAGE 值；PAGE 控制 BANK 存储区示意图可参考图 2-2。

在 C 语言程序中，特殊功能寄存器地址和通用 RAM 地址受 BSR<4:0>控制的，直接寻址访问相应的 BANK 存储区地址之前需要正确设置 BSR 值（实际程序并不需要，编译软件自动插入指令，赋值给 BSR 寄存器）；BSR 控制 BANK 存储区示意图可参考图 2-3。

CMS341x 特殊功能寄存器汇总 Bank0

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
00H	INDF0	寻址该单元会使用FSR0H/FSR0L的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
01H	INDF1	寻址该单元会使用FSR1H/FSR1L的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
02H	PCL	程序计数器低字节								00000000
03H	STATUS	----	----	----	TO	PD	Z	DC	C	---11xxx
04H	FSR0L	间接数据存储器地址0低字节指针								xxxxxxxx
05H	FSR0H	间接数据存储器地址0高字节指针								xxxxxxxx
06H	FSR1L	间接数据存储器地址1低字节指针								xxxxxxxx
07H	FSR1H	间接数据存储器地址1高字节指针								xxxxxxxx
08H	BSR	----	----	----	PAGE<2:0>			BSR1	BSR0	---00000
09H	WREG	工作寄存器								xxxxxxxx
0AH	PCLATH	----	---	----	程序计数器高5位的写缓冲器					---00000
0BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
0CH	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxxxxxx
0DH	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxxxxxx
0EH	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxxxxxx
0FH	PAWP	PAWP7	PAWP6	PAWP5	PAWP4	PAWP3	PAWP2	PAWP1	PAWP0	00000000
10H	PBWP	PBWP7	PBWP6	PBWP5	PBWP4	PBWP3	PBWP2	PBWP1	PBWP0	00000000
11H	PIR1	SPIIF	PWM0IF	RCIF	TXIF	EEIF	ADIF	TMR2IF	TMR1IF	00000000
12H	PIR2	----	---	RACIF	----	----	AFEIF	----	LVDIF	--0—0-0
13H	PAWE	PAWE7	PAWE6	PAWE5	PAWE4	PAWE3	PAWE2	PAWE1	PAWE0	00000000
14H	PBWE	PBWE7	PBWE6	PBWE5	PBWE4	PBWE3	PBWE2	PBWE1	PBWE0	00000000
15H	TMR0	TIMER0数据寄存器								xxxxxxxx
16H	OPTION_REG	----	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	-1111011
17H	TMR1L	16位TIMER1寄存器低字节的数据寄存器								xxxxxxxx
18H	TMR1H	16位TIMER1寄存器高字节的数据寄存器								xxxxxxxx
19H	T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T0OSCEN	T1SYNC	TMR1CS	TMR1ON	00000000
1AH	TMR2	TIMER2模块寄存器								00000000
1BH	PR2	TIMER2周期寄存器								11111111
1CH	T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-0000000
1DH	WDTCON	----	----	----	----	----	----	----	SWDTEN	-------0
1EH	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	----	----	-110----
1FH	LVDCON	LVD_RES	LVD_MODE<1:0>		----	LVD_SEL<2:0>			LVDEN	000-0000

CMS341x 特殊功能寄存器汇总 Bank1

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
080H	INDF0	寻址该单元会使用FSR0H/FSR0L的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
081H	INDF1	寻址该单元会使用FSR1H/FSR1L的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
082H	PCL	程序计数器低字节								00000000
083H	STATUS	----	----	----	TO	PD	Z	DC	C	---11xxx
084H	FSR0L	间接数据存储器地址0低字节指针								xxxxxxxx
085H	FSR0H	间接数据存储器地址0高字节指针								xxxxxxxx
086H	FSR1L	间接数据存储器地址1低字节指针								xxxxxxxx
087H	FSR1H	间接数据存储器地址1高字节指针								xxxxxxxx
088H	BSR	----	----	----	PAGE<2:0>			BSR1	BSR0	---00000
089H	WREG	工作寄存器								xxxxxxxx
08AH	PCLATH	----	---	----	程序计数器高5位的写缓冲器					---00000
08BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
08CH	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	11111111
08DH	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	11111111
08EH	TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	11111111
08FH	WPDC	WPDC7	WPDC6	WPDC5	WPDC4	WPDC3	WPDC2	WPDC1	WPDC0	00000000
091H	PIE1	SPIIE	PWMIE	RCIE	TXIE	EEIE	ADIE	TMR2IE	TMR1IE	00000000
092H	PIE2	----	---	RACIE	----	----	AFEIE	----	LVDIE	--0—0-0
093H	TABLE_DATAH	间接寻址ROM空间地址的高8位数据								xxxxxxxx
094H	SPIBUF	SPI发送/接收寄存器								xxxxxxxx
095H	SPICON	SPIWCOL	SPIOV	SPIEN	SPICKP	SPIM<3:0>				00000000
096H	SPICON2	----	CKE	MODE	----	----	----	----	SPIBF	-00----0
097H	RCSTA	SPEN	RX9EN	SREN	CREN	RCIDL	FERR	OERR	RX9D	00001000
098H	TXSTA	CSRC	TX9EN	TXEN	SYNC	SCKP	----	TRMT	TX9D	00000010
099H	SPBRG	USART波特率低8位寄存器								00000000
09AH	TXREG	USART发送数据寄存器								00000000
09BH	RCREG	USART接收数据寄存器								xxxxxxxx
09CH	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON	00000000
09DH	ADCON1	ADFM	----	ADCS2	----	----	----	----	----	0-0-----
09EH	ADRESL	SAR ADC结果寄存器的低字节								xxxxxxxx
09FH	ADRESH	SAR ADC结果寄存器的高字节								xxxxxxxx

CMS341x 特殊功能寄存器汇总 Bank2

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
100H	INDF0	寻址该单元会使用FSR0H/FSR0L的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
101H	INDF1	寻址该单元会使用FSR1H/FSR1L的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
102H	PCL	程序计数器低字节								00000000
103H	STATUS	----	----	----	TO	PD	Z	DC	C	---11xxx
104H	FSR0L	间接数据存储器地址0低字节指针								xxxxxxxx
105H	FSR0H	间接数据存储器地址0高字节指针								xxxxxxxx
106H	FSR1L	间接数据存储器地址1低字节指针								xxxxxxxx
107H	FSR1H	间接数据存储器地址1高字节指针								xxxxxxxx
108H	BSR	----	----	----	PAGE<2:0>			BSR1	BSR0	---00000
109H	WREG	工作寄存器								xxxxxxxx
10AH	PCLATH	----	---	----	程序计数器高5位的写缓冲器					---00000
10BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
10CH	WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	00000000
10DH	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	00000000
10EH	WPUC	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0	00000000
10FH	WPDA	WPDA7	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0	00000000
110H	WPDB	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0	00000000
111H	EECON3	ROM自写使能寄存器								00000000
112H	PWMCON0	CLKDIV<2:0>			----	----	----	PWM1EN	PWM0EN	000---00
113H	PWMCON1	----	PWMSEL	----	----	----	----	PWM1DIR	PWM0DIR	-0—0000
114H	PWMTH	----	----	PWM1T<9:8>		----	----	PWM0T<9:8>		--00—00
115H	PWM0TL	PWM0周期低位寄存器								00000000
116H	PWM1TL	PWM1周期低位寄存器								00000000
117H	PWMD0L	PWM0占空比低位寄存器								00000000
118H	PWMD1L	PWM1占空比低位寄存器								00000000
119H	PWMD01H	----	----	PWMD1<9:8>		----	----	PWMD0<9:8>		--00—00
11AH	EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0	xxxxxxxx
11BH	EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	00000000
11CH	EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0	xxxxxxxx
11DH	EEADRH	----	----	----	EEADRH4	EEADRH3	EEADRH2	EEADRH1	EEADRH0	---00000
11EH	EECON1	EEPGD	----	EETIME1	EETIME0	WRERR	WREN	WR	RD	0-00x000
11FH	EECON2	EEPROM控制寄存器2（不是物理寄存器）								--------

CMS341x 特殊功能寄存器汇总 Bank3

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
180H	INDF0	寻址该单元会使用FSR0H/FSR0L的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
181H	INDF1	寻址该单元会使用FSR1H/FSR1L的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
182H	PCL	程序计数器低字节								00000000
183H	STATUS	----	----	----	TO	PD	Z	DC	C	---11xxx
184H	FSR0L	间接数据存储器地址0低字节指针								xxxxxxxx
185H	FSR0H	间接数据存储器地址0高字节指针								xxxxxxxx
186H	FSR1L	间接数据存储器地址1低字节指针								xxxxxxxx
187H	FSR1H	间接数据存储器地址1高字节指针								xxxxxxxx
188H	BSR	----	----	----	PAGE<2:0>			BSR1	BSR0	---00000
189H	WREG	工作寄存器								xxxxxxxx
18AH	PCLATH	----	---	----	程序计数器高5位的写缓冲器					---00000
18BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
18CH	ANSEL	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	00000000
18DH	PAPI	PORTA拉电流控制寄存器								11111111
18EH	PBPI	PORTB拉电流控制寄存器								11111111
18FH	COMENA	PORTA灌电流控制寄存器								00000000
190H	COMENB	PORTB灌电流控制寄存器								00000000
191H	SEGENA	PORTA拉电流使能控制寄存器								00000000
192H	SEGENB	PORTB拉电流使能控制寄存器								00000000
193H	IOSR	----	----	----	----	IOSRBH	IOSRBL	IOSRAH	IOSRAL	----0000
194H	BUZCON0	BUZZER输出周期数据								00000000
195H	BUZCON1	BUZEN	----	----	----	----	----	BUZDIV<1:0>		0-----00
196H	AFECON1	BYPASSLDO	ENBLDO	SET_LDO<1:0>		OCP_DIS_LDO	ADC_ST	CLK_DIV	RDY	10000000
197H	AFECON2	ENBAFE	LPWRPGA	PGA_SEL<2:0>			ENSAR<1:0>		ENCHOPB	10110000
198H	AFECON3	CH_SEL<3:0>				----	----	----	EN_ADTESTGN	0000---0
199H	AFECON4	OSR<2:0>			FCHOP_ADC	FADC	LPWR	----	----	110011--
19AH	AFERES0	AFE结果数据寄存器第7位到第0位								xxxxxxxx
19BH	AFERES1	AFE结果数据寄存器第15位到8位								xxxxxxxx
19CH	AFERES2	AFE结果数据寄存器第23位到第16位								xxxxxxxx
19DH	BIMCON1	EN_BIM	ENB_SIN	BIMMODE<1:0>		ENB_CHOPBIM	BW_BIM	ISIN	VSEN	00000000
19EH	BIMCON2	ENB_DDL	EN_BIMBW	SEL_CLK<1:0>			EN_SINRES	FREQ_SIN<2:0>		00110000

CMS341x 特殊功能寄存器汇总 Bank4

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
200H	INDF0	寻址该单元会使用FSR0H/FSR0L的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
201H	INDF1	寻址该单元会使用FSR1H/FSR1L的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
202H	PCL	程序计数器低字节								00000000
203H	STATUS	----	----	----	TO	PD	Z	DC	C	---11xxx
204H	FSR0L	间接数据存储器地址0低字节指针								xxxxxxxx
205H	FSR0H	间接数据存储器地址0高字节指针								xxxxxxxx
206H	FSR1L	间接数据存储器地址1低字节指针								xxxxxxxx
207H	FSR1H	间接数据存储器地址1高字节指针								xxxxxxxx
208H	BSR	----	----	----	PAGE<2:0>			BSR1	BSR0	---00000
209H	WREG	工作寄存器								xxxxxxxx
20AH	PCLATH	----	---	----	程序计数器高5位的写缓冲器					---00000
20BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
20CH	MDUCON	MDUOV	----	----	----	----	----	DIV48EN	MUL24EN	0-----00
20DH	MDUDT0	MDU数据寄存器								xxxxxxxx
20EH	MDUDT1	MDU数据寄存器								xxxxxxxx
20FH	MDUDT2	MDU数据寄存器								xxxxxxxx
210H	MDUDT3	MDU数据寄存器								xxxxxxxx
211H	MDUDT4	MDU数据寄存器								xxxxxxxx
212H	MDUDT5	MDU数据寄存器								xxxxxxxx
213H	MDUDT6	MDU数据寄存器								xxxxxxxx
214H	MDUDT7	MDU数据寄存器								xxxxxxxx
215H	MDUDT8	MDU数据寄存器								xxxxxxxx
216H	LCDCON0	LCDEN	BIAS	COMSEL<2:0>			LCDCLK<2:0>			00000000
217H	LCDCON1	CPREN	LCDF	SEGOUT<1:0>		FCCTLM<1:0>		VLCDX<1:0>		00000000
218H	LCDADD	LCDCS	B2RES<1:0>	LCDADD<4:0>						00000000
219H	LCDDATA	----	----	LCDDATA<5:0>						--xxxxxx
21AH	COMEN			COM5EN	COM4EN	COM3EN	COM2EN	COM1EN	COM0EN	00000000
21BH	SEGEN0	SEG7EN	SEG6EN	SEG5EN	SEG4EN	SEG3EN	SEG2EN	SEG1EN	SEG0EN	00000000
21CH	SEGEN1	SEG15EN	SEG14EN	SEG13EN	SEG12EN	SEG11EN	SEG10EN	SEG9EN	SEG8EN	00000000
21DH	SEGEN2	----	SEG22EN	SEG21EN	SEG20EN	SEG19EN	SEG18EN	SEG17EN	SEG16EN	-0000000

CMS341x 特殊功能寄存器汇总 Bank31

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
F10H	INDF0	寻址该单元会使用FSR0H/FSR0L的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
F11H	INDF1	寻址该单元会使用FSR1H/FSR1L的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
F12H	PCL	程序计数器低字节								00000000
F13H	STATUS	----	----	----	TO	PD	Z	DC	C	---11xxx
F14H	FSR0L	间接数据存储器地址0低字节指针								xxxxxxxx
F15H	FSR0H	间接数据存储器地址0高字节指针								xxxxxxxx
F16H	FSR1L	间接数据存储器地址1低字节指针								xxxxxxxx
F17H	FSR1H	间接数据存储器地址1高字节指针								xxxxxxxx
F18H	BSR	----	----	----	PAGE<2:0>			BSR1	BSR0	---00000
F19H	WREG	工作寄存器								xxxxxxxx
F1AH	PCLATH	----	---	----	程序计数器高5位的写缓冲器					---00000
F1BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
FE4H	STATUS_SHAD	----	---	----	----	---	Z_SHAD	DC_SHAD	C_SHAD	-----xxx
FE5H	WREG_SHAD	工作寄存器（ACC）影子寄存器								xxxxxxxx
FE6H	BSR_SHAD	----	---	----	存储区选择寄存器影子寄存器					---xxxxx
FE7H	PCLATH_SHAD	----	---	----	程序计数器高字节锁存寄存器影子寄存器					---xxxxx
FE8H	FSR0L_SHAD	间接数据存储器地址0低字节指针影子寄存器								xxxxxxxx
FE9H	FSR0H_SHAD	间接数据存储器地址0高字节指针影子寄存器								xxxxxxxx
FEAH	FSR1L_SHAD	间接数据存储器地址1低字节指针影子寄存器								xxxxxxxx
FEBH	FSR1H_SHAD	间接数据存储器地址1高字节指针影子寄存器								xxxxxxxx
FEDH	STKPTR	----	---	----	----	----	当前堆栈指针			-----111
FEEH	TOSL	栈顶低字节								xxxxxxxx
FEFH	TOSH	----	---	----	栈顶高字节					---xxxxx

2.2 寻址方式

2.2.1 直接寻址

通过工作寄存器（ACC）来对 RAM 进行操作。

例：ACC 的值送给 30H 寄存器

```
LD          30H,A
```

例：30H 寄存器的值送给 ACC

```
LD          A,30H
```

例：访问不同 BANK 存储区数据

```
PAGE0       RAM1 EQU ?      ;定义位于 BANK0-BANK3 的数据变量
PAGE1       RAM2 EQU ?      ;定义位于 BANK4-BANK7 的数据变量
……
LDIB        00H             ;PAGE=3'b000，指向 BANK0-BANK3 存储区
LD          A,RAM1          ;把 RAM1 地址的值送给 ACC
……
LDIB        04H             ;PAGE=3'b001，指向 BANK4-BANK7 存储区
LD          A,RAM2          把 RAM2 地址的值送给 ACC
```

2.2.2 立即寻址

把立即数传给工作寄存器（ACC）。

例：立即数 12H 送给 ACC

```
LDIA        12H
```

2.2.3 间接寻址

数据存储器能被直接或间接寻址。通过 INDFn 寄存器可间接寻址，INDFn 不是物理寄存器。当对 INDFn 进行存取时，它会根据 FSRn 寄存器（FSRnL,FSRnH）内的值作为地址，并指向该地址的寄存器，因此在设置了 FSRn 寄存器后，就可把 INDFn 寄存器当作目的寄存器来存取。间接读取 INDFn（FSRn=0）将产生 00H。间接写入 INDFn 寄存器，将导致一个空操作。

FSR 寄存器形成的 16 位地址可以对 65536 个地址单元的空间进行寻址。这些地址单元可划分为 3 个存储区：

- 传统数据存储区
- 线性数据存储区
- 程序存储器

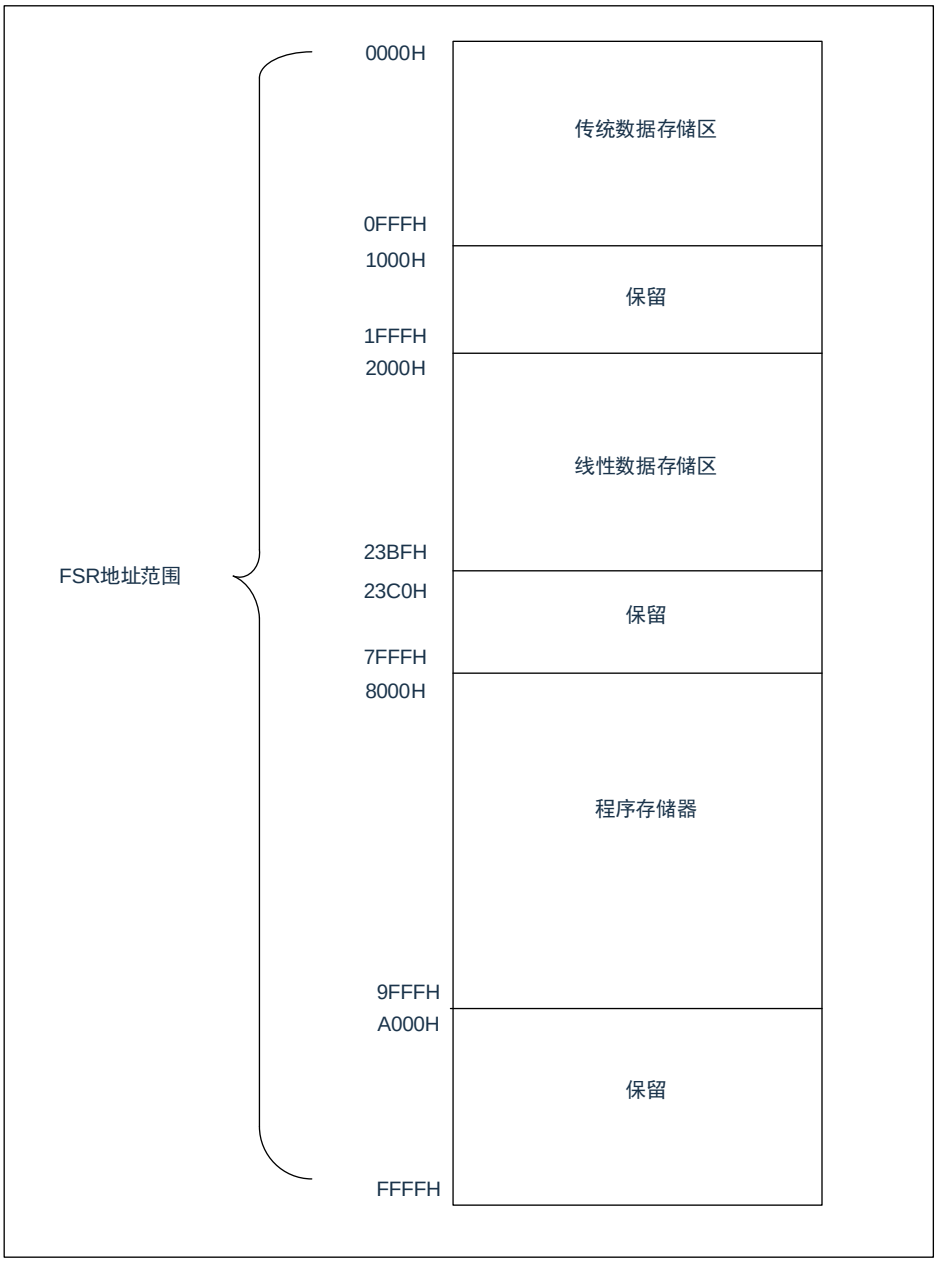


图 2-4：间接寻址地址空间

2.2.3.1 传统数据存储区

传统数据存储区指的是从 FSR 地址 0000H 到 FSR 地址 0FFFH 的区域。此地址对应于所有 SFR、GPR 和公共 RAM 的绝对地址。

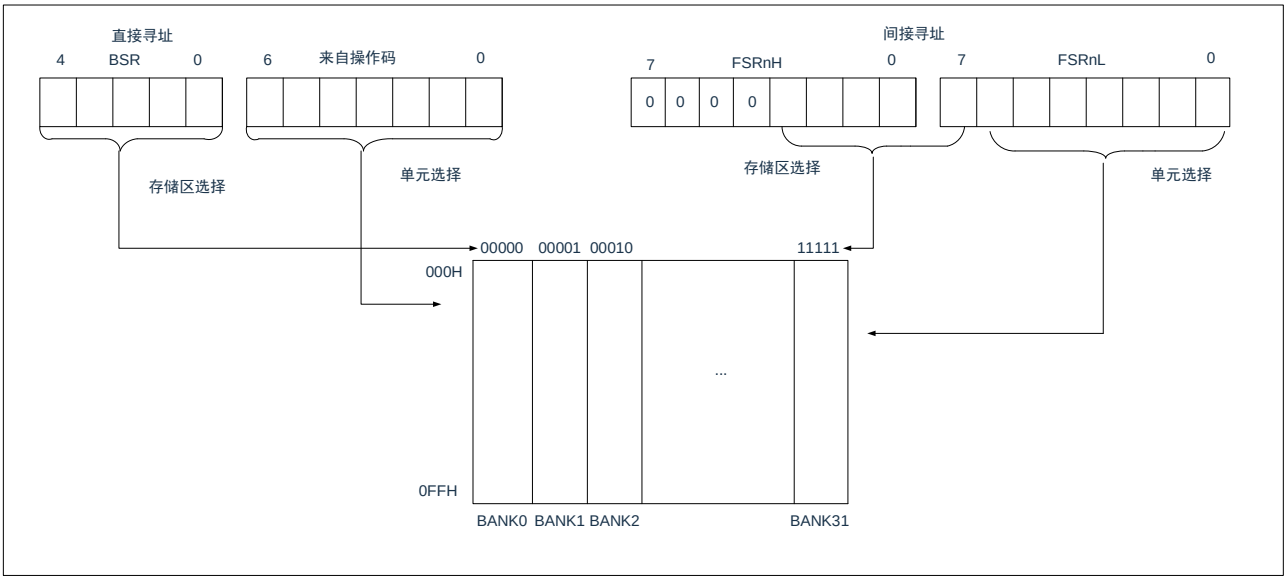


图 2-5：传统数据存储区映射

2.2.3.2 线性数据存储区

线性数存储区指的是从 FSR 地址 2000H 到 FSR 地址 23BFH 的区域。该区域为虚拟区域，它指向所有存储区中 80 字节的 GPR 存储区域。

使用线性数据存储器区域允许缓冲区大于80 字节，因为当FSR增大到超过一个存储区时，会直接转到下一个存储区的GPR存储区。

线性数据存储器区域不包括 16 字节的公共存储器。

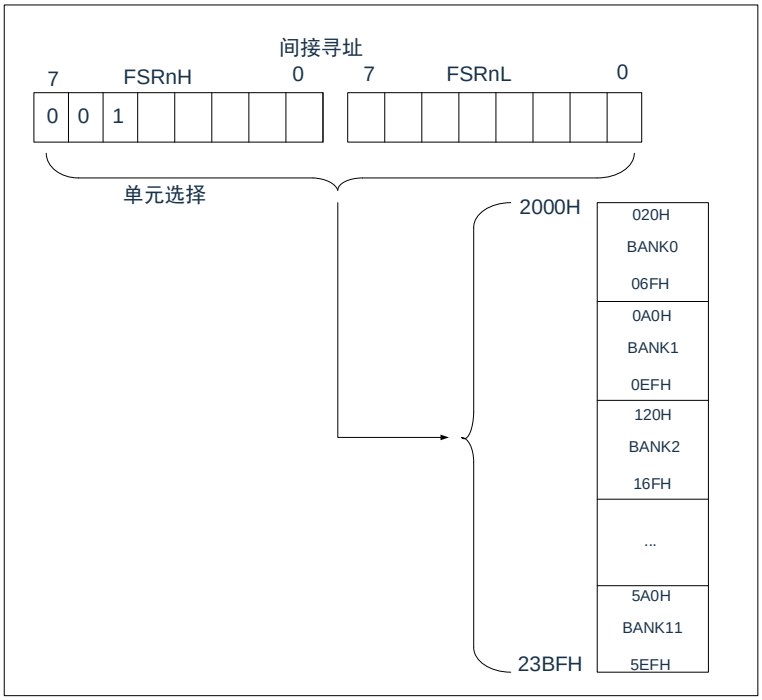


图 2-6：线性数据存储区映射

2.2.3.3 程序存储器

要使常数的访问更为容易，可将整个闪存程序存储器映射到 FSRn 地址空间的高半部分。当 FSRnH 的 MSB 置 1 时，低 13 位就是可通过 INDFn 进行访问的程序存储器的地址。每个存储单元的低 8 位可通过 INDF 进行访问，高 8 位存放在 TABLE_DATAH 寄存器。通过 FSR/INDF 接口无法对程序存储器执行写操作。所有通过 FSR/INDF 接口对程序存储器进行访问的指令都需要一个额外的指令周期才能完成。

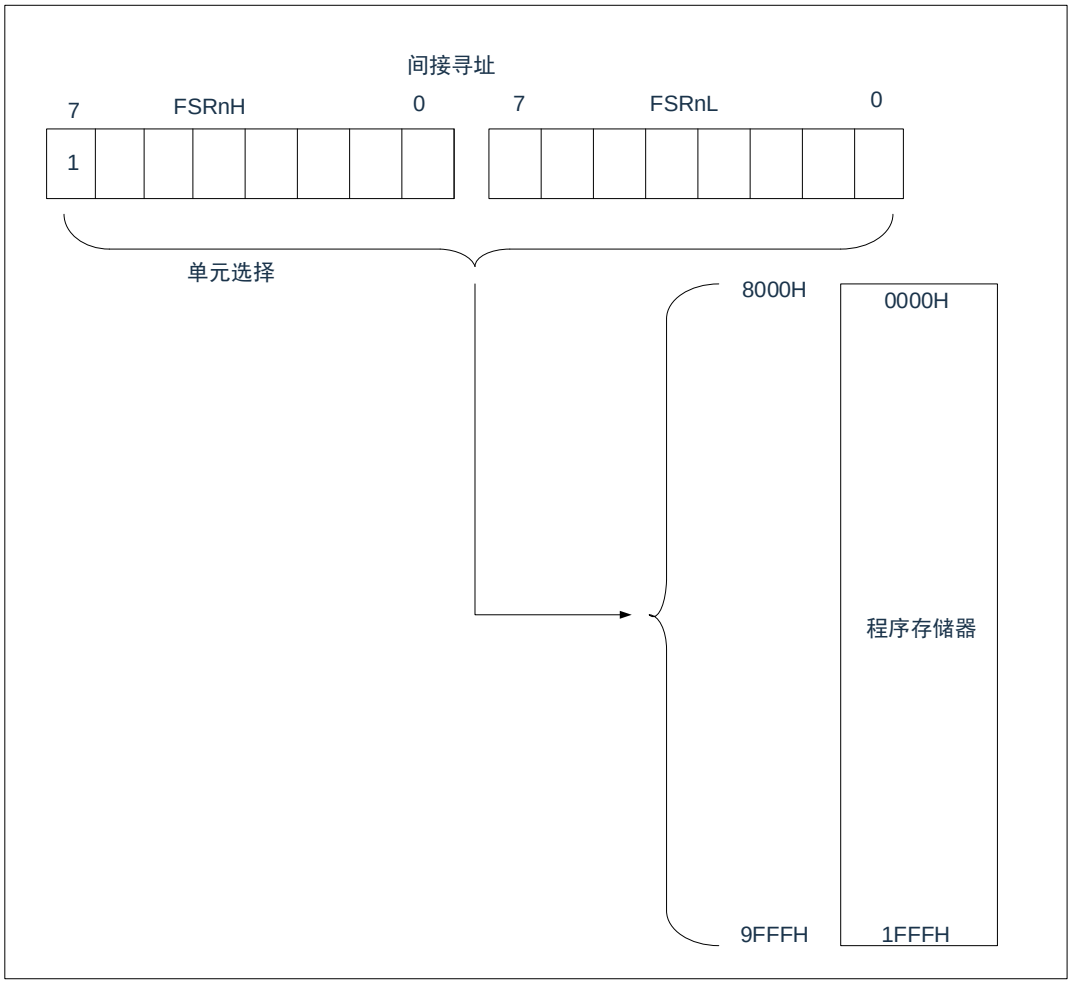


图 2-7：程序存储器映射

以下例子说明了程序中间接寻址的用法。

例：FSR 及 INDF 的应用

```
        CLR     FSR0H
        LDIA    30H
        LD      FSR0L,A         ;间接寻址指针指向 30H
        CLR     INDF0           ;清零 INDF 实际是清零 FSR 指向的 30H 地址 RAM
```

例：间接寻址清 RAM(20H-7FH 和 A0H-EFH)举例：

```
        LDIA    020H
        LD      FSR1H,A
        LDIA    00H
        LD      FSR1L,A     ;间接寻址指针指向线性存储区的 00H 地址，对应传统数据区的 20H 地址
LOOP:
        CLR     INDF1       ;清零 FSR 所指向的地址
        INCR    FSR1L       ;地址加 1
        LDIA    0A0H
        SUBA    FSR1L
        SNZB    STATUS,Z    ;一直清零至 FSR 地址为 9FH
        JP      LOOP
```

例：间接寻址访问程序存储器的 1FF0H 地址举例：

```
        LDIA    09FH
        LD      FSR0H,A
        LDIA    0F0H
        LD      FSR0L,A     ;间接寻址指针指向程序存储器的 01FF0H 地址
        LD      A,INDF0     ;将地址 1FF0H 的低 8 位数据（34H）给 ACC,高 8 位数据（12H）给 TABLE_DATAH 寄存器
        …
        ORG     1FF0H       ;表格起始地址
        DW      1234H       ;1FF0H 地址表格内容
```

2.3 堆栈

芯片的堆栈缓存器共 8 层，堆栈缓存器既不是数据存储器的一部分，也不是程序内存的一部分。对它的操作通过堆栈指针（STKPTR）来实现，当系统复位后堆栈指针会指向堆栈顶部。可通过 TOSH,TOSL 和 STKPTR 寄存器来使用堆栈，TOSH:TOSL 寄存器对指向栈顶，这两个寄存器只可读不能写。当发生子程序调用（CALL,CALLA 指令）及中断时的程序计数器（PC）值被压入堆栈缓存器，STKPTR 值递增 1；当从中断或子程序返回时将数值返回给程序计数器（PC），STKPTR 值递减 1。任何时候都可以检查 STKPTR，以查看可用堆栈空间。STKPTR 总是指向堆栈中的当前使用单元。以下图示说明其工作原理。

访问堆栈实例 1

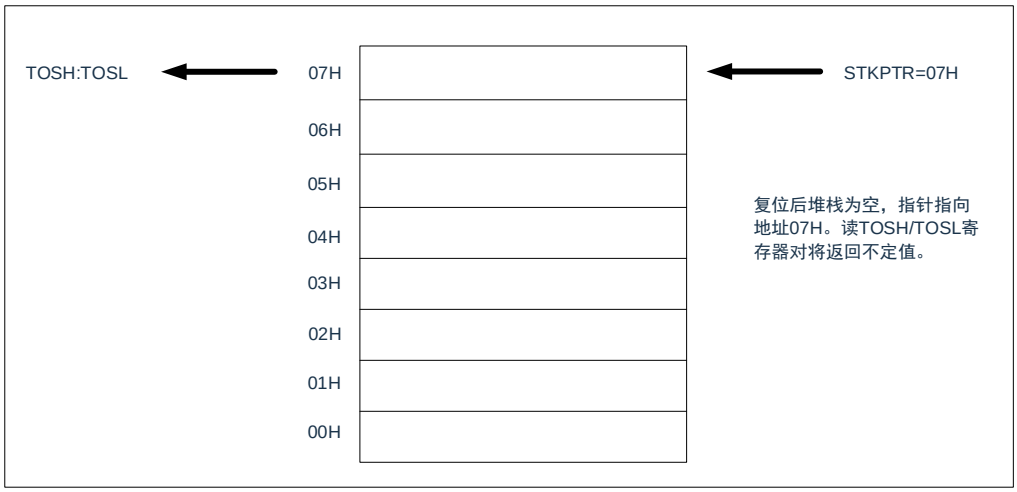


图 2-8：访问堆栈实例 1

访问堆栈实例 2

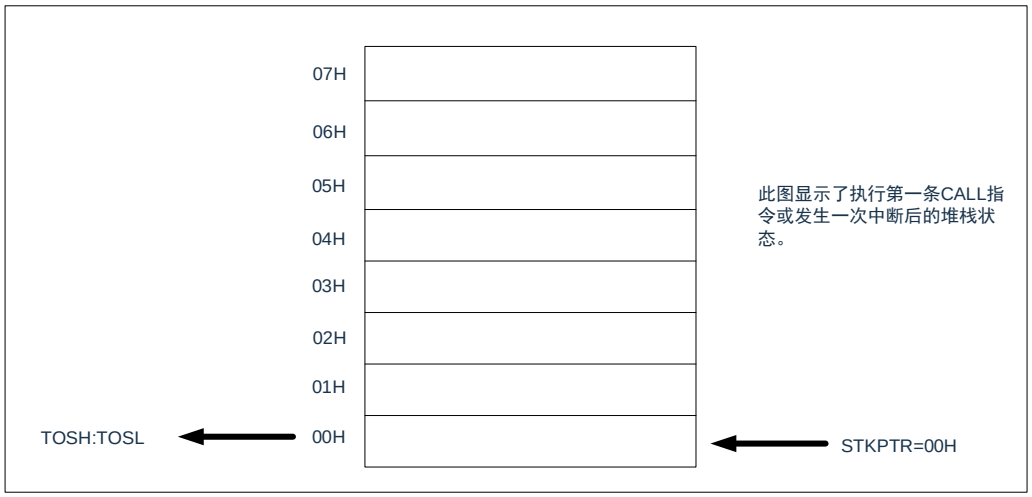


图 2-9：访问堆栈实例 2

访问堆栈实例 3

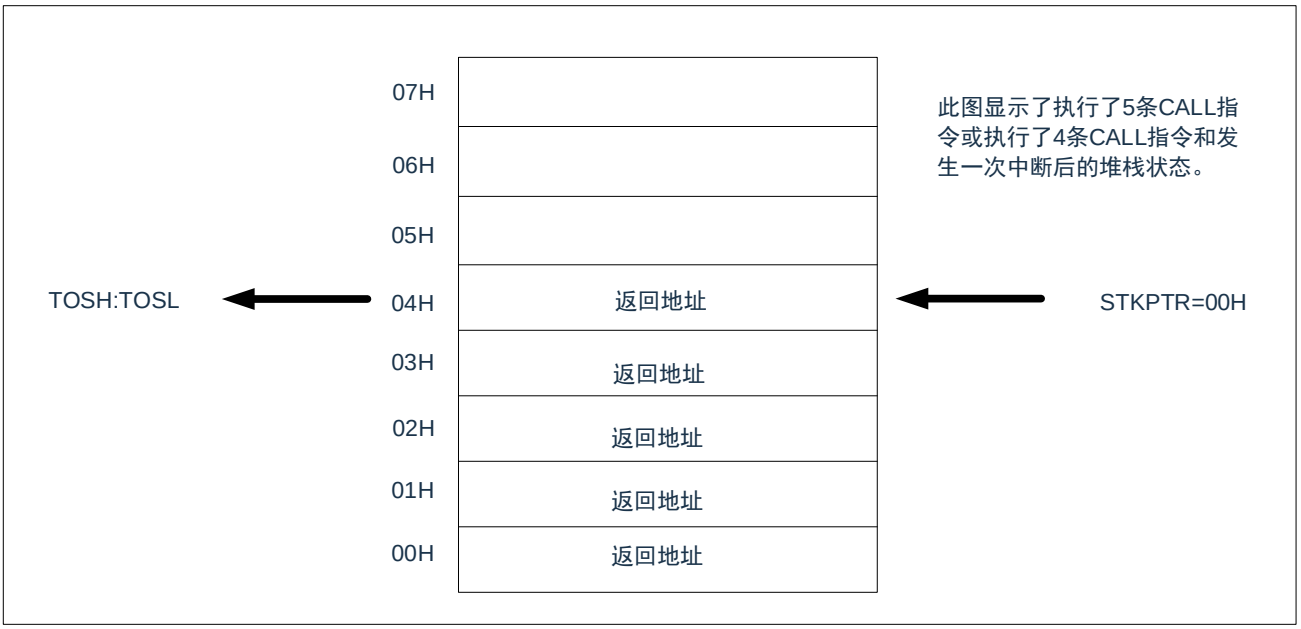


图 2-10：访问堆栈实例 3

堆栈缓存器的使用将遵循一个原则“先进后出”。

注：堆栈缓存器只有 8 层，如果堆栈已满，并且发生不可屏蔽的中断，那么只有中断标志位会被记录下来，而中断响应则会被抑制，直到堆栈指针发生递减，中断才会被响应，这个功能可以防止中断使堆栈溢出，同样如果堆栈已满，并且发生子程序调用，那么堆栈将会发生溢出，首先进入堆栈的内容将会丢失，只有最后 8 个返回地址被保留，故用户在写程序时应注意此点，以免发生程序走飞。

2.4 工作寄存器（ACC）

2.4.1 概述

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（STATUS 状态寄存器中），用来表示运算结果的状态。

ACC 寄存器是一个 8-Bit 的寄存器，ALU 的运算结果可以存放在此，它属于数据存储器的一部分，又叫做 WREG 寄存器，在运算中供 ALU 使用；因此 ACC 可以被寻址，也可以通过所提供的指令来使用。

2.4.2 ACC 应用

例：用 ACC 做数据传送

```
LD      A,R01       ;将寄存器 R01 的值赋给 ACC
LD      R02,A       ;将 ACC 的值赋给寄存器 R02
```

例：用 ACC 做立即寻址目标操作数

```
LDIA    30H         ;给 ACC 赋值 30H
ANDIA   30H         ;将当前 ACC 的值跟立即数 30H 进行“与”操作，结果放入 ACC
XORIA   30H         ;将当前 ACC 的值跟立即数 30H 进行“异或”操作，结果放入 ACC
```

例：用 ACC 做双操作数指令的第一操作数

```
HSUBA   R01         ;ACC-R01，结果放入 ACC
HSUBR   R01         ;ACC-R01，结果放入 R01
```

例：用 ACC 做双操作数指令的第二操作数

```
SUBA    R01         ;R01-ACC，结果放入 ACC
SUBR    R01         ; R01-ACC，结果放入 R01
```

2.5 程序状态寄存器（STATUS）

STATUS 寄存器如下表所示，包含：

- ALU 的算术状态。
- 复位状态。

与其他寄存器一样，STATUS 寄存器可以是任何指令的目标寄存器。如果一条影响 Z、DC 或 C 位的指令以 STATUS 寄存器作为目标寄存器，则不能写这 3 个状态位。这些位根据器件逻辑被置 1 或清零。而且也不能写 TO 和 PD 位。因此将 STATUS 作为目标寄存器的指令可能无法得到预期的结果。

例如，CLR STATUS 会清零高 3 位，并将 Z 位置 1。这样 STATUS 的值将为 000u u1uu（其中 u=不变）。因此，建议仅使用 CLRB、SETB、SWAPA、SWAPR 指令来改变 STATUS 寄存器，因为这些指令不会影响任何状态位。

程序状态寄存器 STATUS(03H)

03H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STATUS	---	---	---	TO	PD	Z	DC	C
R/W	---	---	---	R/W	R/W	R/W	R/W	R/W
复位值	---	---	---	1	1	X	X	X

Bit7~Bit5 未用

Bit4 TO: 超时位
1= 上电或执行了CLRWDT指令或STOP指令
0= 发生了WDT超时

Bit3 PD: 掉电位
1= 上电或执行了CLRWDT指令
0= 执行了STOP指令

Bit2 Z: 结果为零位
1= 算术或逻辑运算的结果为零
0= 算术或逻辑运算的结果不为零

Bit1 DC: 半进位/借位位
1= 发生了结果的第4低位向高位进位
0= 结果的第4低位没有向高位进位

Bit0 C: 进位/借位位
1= 结果的最高位发生了进位或没有发生借位
0= 结果的最高位没有发生进位或发生了借位

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
STOP 指令	1	0
CLRWDT 指令	1	1
休眠	1	0

影响 PD、TO 的事件表

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

复位后 TO/PD 的状态

2.6 预分频器（OPTION_REG）

OPTION_REG 寄存器是可读写的寄存器，各个控制位用于配置：

- TIMER0/WDT 预分频器。
- TIMER0。

预分频器 OPTION_REG(81H)

81H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	---	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W	---	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	1	1	1	1	0	1	1

Bit7 未用

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

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: 预分配参数配置位

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，CLRWDT 指令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,SETB TMR0,1 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，应该执行以下指令。

```
CLRB    INTCON,GIE          ;关中断总使能位,避免在执行以下特定时序时进入中断程序
LDIA    B'00000111'
ORR     OPTION_REG,A        ;预分频器设置为最大值
CLR     TMR0                ;TMR0 清零
SETB    OPTION_REG,PSA      ;设置预分频器分配给 WDT
CLRWDT                      ;WDT 清零
LDIA    B'xxxx1xxx'         ;设置新的预分频器
LD      OPTION_REG,A
CLRWDT                      ;WDT 清零
SETB    INTCON,GIE          ;若程序需要用到中断,此处重新打开总使能位
```

将预分频器从分配给 WDT 切换为分配给 TIMER0 模块，应该执行以下指令

```
CLRWDT                      ;WDT 清零
LDIA    B'00xx0xxx'         ;设置新的预分频器
LD      OPTION_REG,A
```

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。

2.7 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。

程序计数器（PC）是 13-Bit 宽度，低 8 位通过 PCL（02H）寄存器用户可以访问，高 5 位用户不能访问。可容纳 8K×16Bit 程序地址。对 PCL 赋值将会产生一个短跳转动作，跳转范围为当前页的 256 个地址。

注：当程序员在利用 PCL 作短跳转时，要先对 PC 高位缓冲寄存器 PCLATH 进行赋值。

下面给出几种特殊情况的 PC 值。

复位时	PC=0000；
中断时	PC=0004(原来的 PC+1 会被自动压入堆栈)；
CALL 时	PC=程序指定地址(原来的 PC+1 会被自动压入堆栈)；
CALLA 时	PC<12:8>=PCLATH 的值,PC<7:0>=ACC 的值(原来的 PC+1 会被自动压入堆栈)；
RET、RETI、RET i 时	PC=堆栈出来的值；
操作 PCL 时	PC<12:8>=PCLATH 的值，PC<7:0>=用户指定的值；
JP 时	PC=程序指定的值；
JPI 时	PC=程序指定的值；
JPA 时	PC=PC+1+ACC 的值；
其它指令	PC=PC+1；

2.8 看门狗计数器（WDT）

看门狗定时器（Watchdog 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。

2.8.1 WDT 周期

WDT 与 TIMER0 共用 8 位预分频器。芯片复位后，WDT 默认溢出周期为 128ms，WDT 溢出周期为 16ms* 分频系数，假如需要改变 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。

“CLRWDT”和“STOP”指令将清除 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 应该在其溢出前被“CLRWDT”指令清零，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 溢出前执行“CLRWDT”指令，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，则状态寄存器（STATUS）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. 若使用 WDT 功能，一定要在程序的某些地方放置“CLRWDT”指令，以保证在 WDT 溢出前能被清零。否则会使芯片不停的复位，造成系统无法正常工作。
2. 不能在中断程序中对 WDT 进行清零，否则无法侦测到主程序“跑飞”的情况。
3. 程序中应在主程序中有一次清 WDT 的操作，尽量不要在多个分支中清零 WDT，这种架构能最大限度发挥看门狗计数器的保护功能。
4. 不同芯片的看门狗计数器溢出时间有一定差异，所以设置清 WDT 时间时，应与 WDT 的溢出时间有较大的冗余，以避免出现不必要的 WDT 复位。

2.8.2 看门狗定时器控制寄存器 WDTCON

看门狗定时器控制寄存器 WDTCON(105H)

105H	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 配置位为 1，则 WDT 始终被使能，而与 SWDTEN 控制位的状态无关。如果 CONFIG 中 WDT 配置位为 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

时钟信号从 OSCIN 引脚输入后（或者由内部振荡产生），在片内产生 4 个非重叠正交时钟信号，分别称作 Q1、Q2、Q3、Q4。在 IC 内部每个 Q1 使程序计数器（PC）增量加一，Q4 从程序存储单元中取出该指令，并将其锁存在指令寄存器中。在下一个 Q1 到 Q4 之间对取出的指令进行译码和执行，也就是说 4 个时钟周期才会执行一条指令。下图表示时钟与指令周期执行时序图。

一个指令周期含有 4 个 Q 周期，指令的执行和获取是采用流水线结构，取指占用一个指令周期，而译码和执行占用另一个指令周期，但是由于流水线结构，从宏观上看，每条指令的有效执行时间是一个指令周期。如果一条指令引起程序计数器地址发生改变（例如 JP）那么预取的指令操作码就无效，就需要两个指令周期来完成该条指令，这就是对 PC 操作指令都占用两个时钟周期的原因。

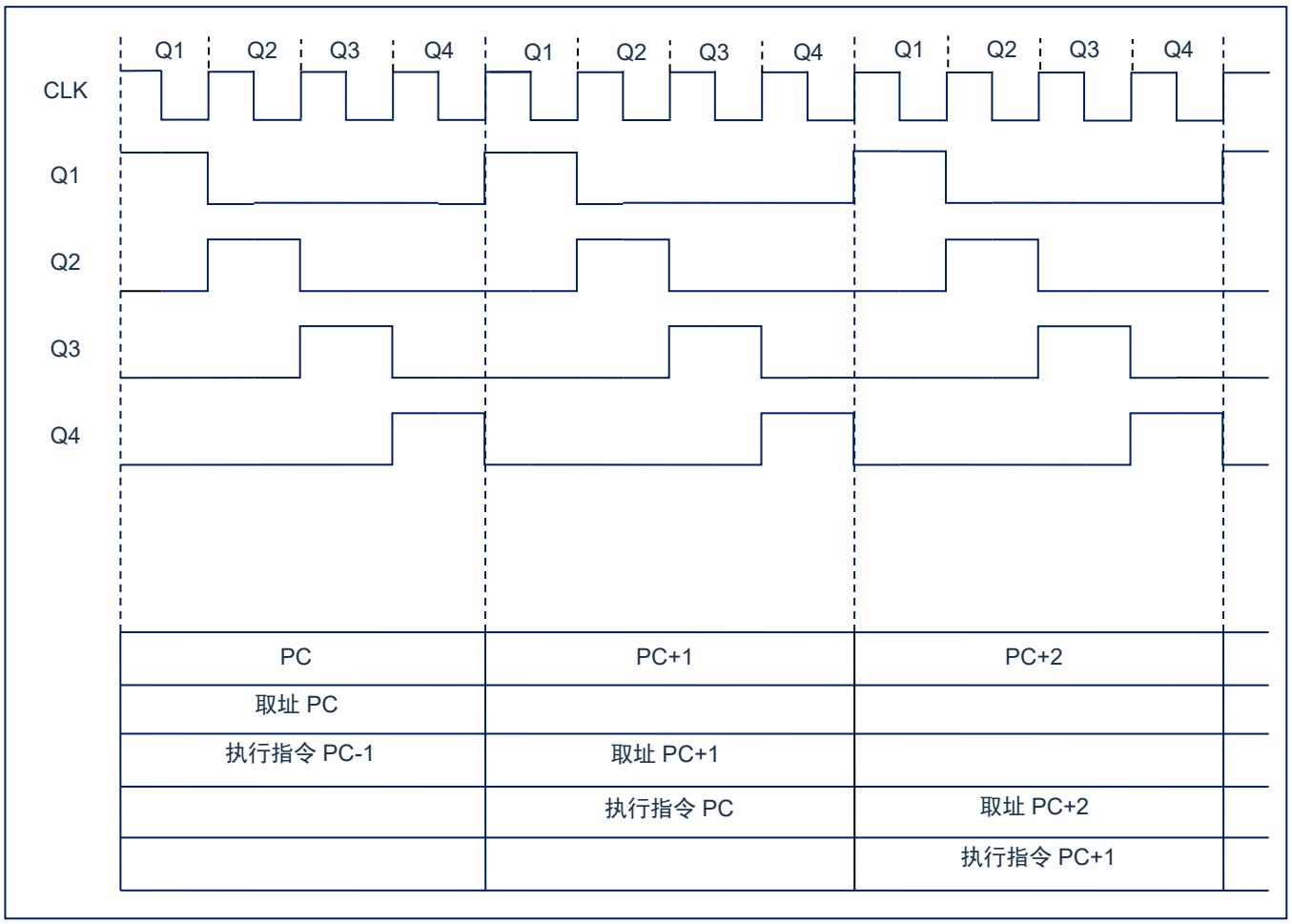


图 3-1：时钟与指令周期时序图

下面列出系统工作频率与指令速度的关系：

系统工作频率(F_{SYS})	双指令周期	单指令周期
2MHz	4μs	2μs
4MHz	2μs	1μs
8MHz	1μs	500ns
16MHz	500ns	250ns

3.2 系统振荡器

芯片有 1 种振荡方式，内部 RC 振荡。

3.2.1 内部 RC 振荡

芯片默认的振荡方式为内部 RC 振荡，其振荡频率为 24MHz，可通过 OSCCON 寄存器设置芯片工作频率。

3.3 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。

3.4 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(8FH)

8FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	---	---
R/W	---	R/W	R/W	R/W	---	---	---	---
复位值	---	1	1	0	---	---	---	---

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（默认）

101=　$F_{SYS} = F_{HSI}$ /4

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=　F_{SYS} = 32kHz（LSI）

Bit3~Bit0　　未用。

注：F_{HSI} 为内部高速振荡器频率 24MHz；F_{SYS} 为系统工作频率。

3.5 时钟框图

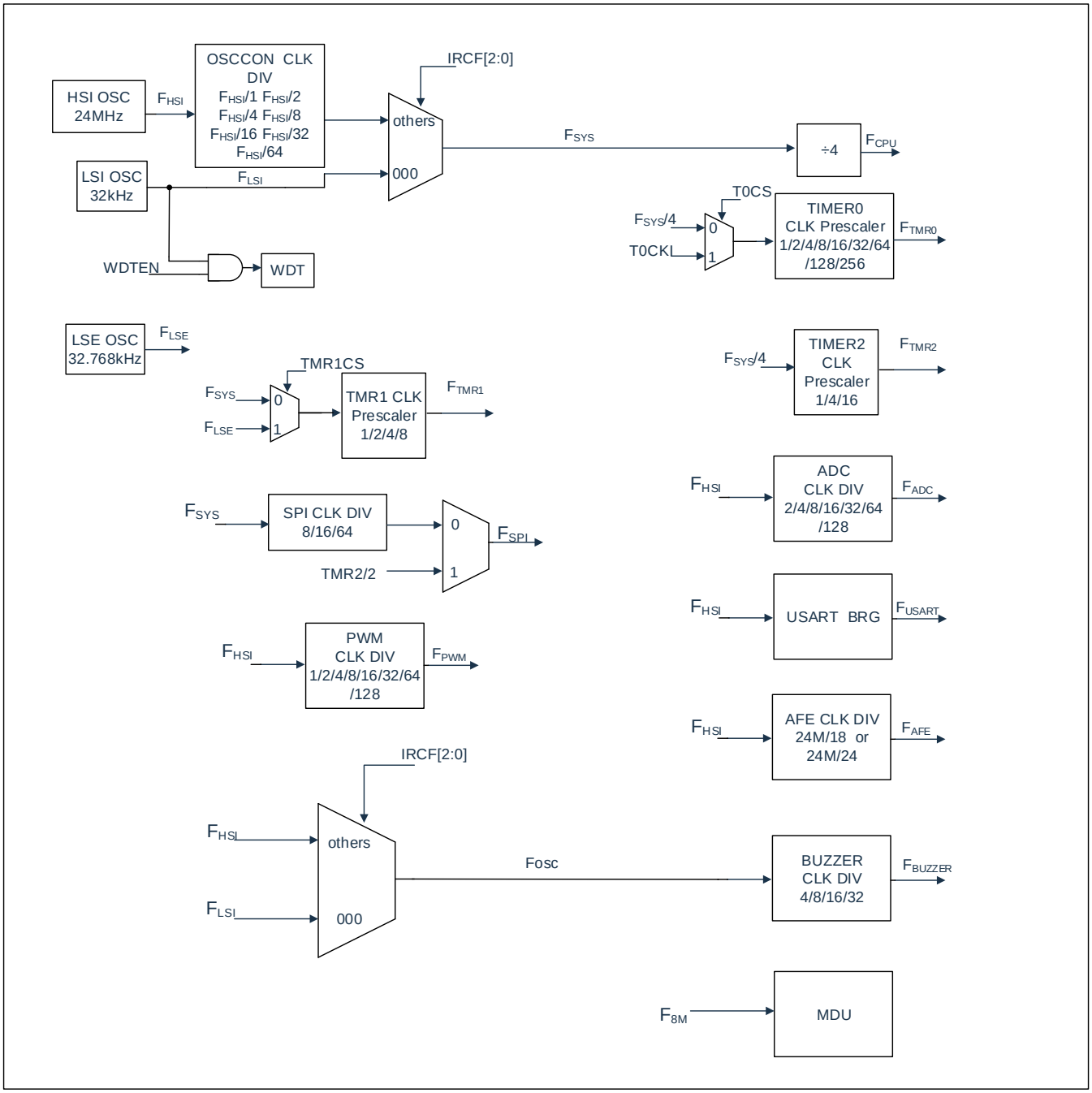


图 3-2：时钟框图

4. 复位

芯片可用如下 3 种复位方式：

- 上电复位；
- LVR 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。

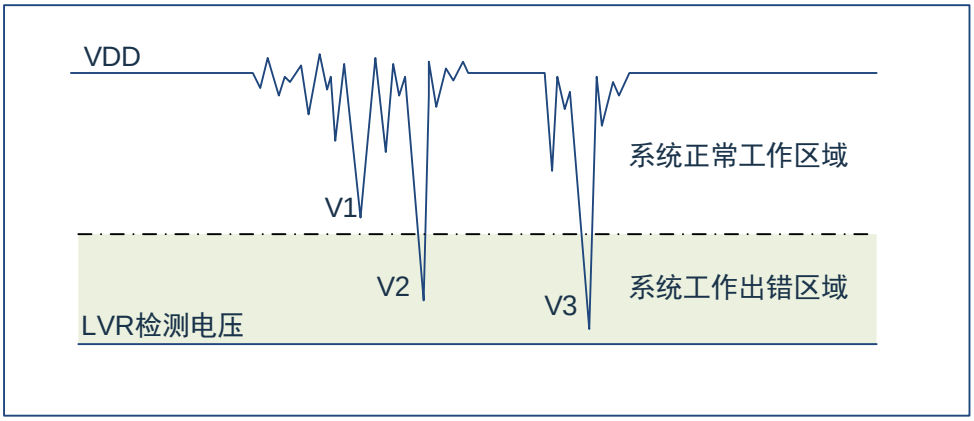


图4-1：掉电复位示意图

上图是一个典型的掉电复位示意图。图中，VDD 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 VDD 跌至 V1 时，系统仍处于正常状态；当 VDD 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。VDD 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，VDD 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：

- 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几率。

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 2.8WDT 应用章节。

5. 休眠模式

5.1 进入休眠模式

执行 STOP 指令可进入休眠模式。如果 WDT 使能，那么：

- WDT 将被清零并继续运行。
- STATUS 寄存器中的 PD 位被清零。
- TO 位被置 1。
- 关闭振荡器驱动器。
- I/O 端口保持执行 STOP 指令之前的状态（驱动为高电平、低电平或高阻态）。

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。

5.2 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. PORTA/PORTB 电平变化中断或外设中断。

上述两种事件被认为是程序执行的延续，STATUS 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，而在执行 STOP 指令时被清零。TO 位在发生 WDT 唤醒时被清零。

当执行 STOP 指令时，下一条指令（PC+1）被预先取出。如果希望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。如果 GIE 位被清零（禁止），器件将继续执行 STOP 指令之后的指令。如果 GIE 位被置 1（允许），器件执行 STOP 指令之后的指令，然后跳转到中断地址（0004h）处执行代码。如果不想执行 STOP 指令之后的指令，用户应该在 STOP 指令后面放置一条 NOP 指令。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 使用中断唤醒

当禁止全局中断（GIE 被清零）时，并且有任一中断源将其中断允许位和中断标志位置 1，将会发生下列事件之一：

- 如果在执行 STOP 指令之前产生了中断，那么 STOP 指令将被作为一条 NOP 指令执行。因此，WDT 及其预分频器和后分频器（如果使能）将不会被清零，并且 TO 位将不会被置 1，同时 PD 也不会被清零。
- 如果在执行 STOP 指令期间或之后产生了中断，那么器件将被立即从休眠模式唤醒。STOP 指令将在唤醒之前执行完毕。因此，WDT 及其预分频器和后分频器（如果使能）将被清零，并且 TO 位将被置 1，同时 PD 也将被清零。即使在执行 STOP 指令之前检查到标志位为 0，它也可能在 STOP 指令执行完毕之前被置 1。要确定是否执行了 STOP 指令，可以测试 PD 位。如果 PD 位置 1，则说明 STOP 指令被作为一条 NOP 指令执行了。在执行 STOP 指令之前，必须先执行一条 CLRWDT 指令，来确保将 WDT 清零。

5.4 休眠模式应用举例

系统在进入休眠模式之前，若用户需要获得较小的休眠电流，请先确认所有 I/O 的状态，若用户方案中存在悬空的 I/O 口，把所有悬空口都设置为输出口，确保每一个 I/O 都有一个固定的状态，以避免 I/O 为输入状态时，口线电平处于不定态而增大休眠电流；关断 AD 等其它外设模块；根据实际方案的功能需求可禁止 WDT 功能来减小休眠电流。

例：进入休眠的处理程序

```
SLEEP_MODE：
        CLR     INTCON      ;关断中断使能
        LDIA    B'00000000'
        LD      TRISA,A
        LD      TRISB,A     ;所有 I/O 设置为输出口
        LD      TRISC,A
        ...                 ;关闭其它功能
        LDIA    0A5H
        LD      SP_FLAG,A   ;置休眠状态记忆寄存器（用户自定义）
        CLRWDT              ;清零 WDT
        STOP                ;执行 STOP 指令
```

5.5 休眠模式唤醒时间

当 MCU 从休眠态被唤醒时，需要等待一个振荡稳定时间（Reset Time），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}）	$F_{SYS}=F_{HSI}$	$T_{WAIT}=520*1/F_{HSI}+16*1/F_{HSI}$
	$F_{SYS}= F_{HSI} /2$	$T_{WAIT}=520*2/F_{HSI}+16*1/F_{HSI}$
	…	…
	$F_{SYS}= F_{HSI} /64$	$T_{WAIT}=520*64/F_{HSI}+16*1/F_{HSI}$
内部低速 RC 振荡（F_{LSI}）	----	$T_{WAIT}=11/F_{LSI}$

6. I/O 端口

芯片有 3 个 I/O 端口：PORTA、PORTB、PORTC（最多 24 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
PROTA	0	施密特触发输入，推挽式输出，SEG2	I/O
	1	施密特触发输入，推挽式输出，SEG3	I/O
	2	施密特触发输入，推挽式输出，TMR0 外部时钟输入，SEG4	I/O
	3	施密特触发输入，推挽式输出，SEG5	I/O
	4	施密特触发输入，推挽式输出，TMR1 外部时钟输入，SEG6	I/O
	5	施密特触发输入，推挽式输出，TMR1 门控输入，SEG7	I/O
	6	施密特触发输入，推挽式输出，SEG8	I/O
	7	施密特触发输入，推挽式输出，SEG9	I/O
PORTB	0	施密特触发输入，推挽式输出，AN0，SEG10	I/O
	1	施密特触发输入，推挽式输出，AN1，SEG11	I/O
	2	施密特触发输入，推挽式输出，AN2，SEG12	I/O
	3	施密特触发输入，推挽式输出，AN3，SEG13	I/O
	4	施密特触发输入，推挽式输出，AN4，SEG14	I/O
	5	施密特触发输入，推挽式输出，SEG15	I/O
	6	施密特触发输入，推挽式输出，USART 的 TX 口，SEG16	I/O
	7	施密特触发输入，推挽式输出，USART 的 RX 口，SEG17	I/O
PORTC	0	施密特触发输入，推挽式输出，PWM0，SPI 从动模式控制输入，SEG18	I/O
	1	施密特触发输入，推挽式输出，PWM1，SPI 时钟口，SEG19	I/O
	2	施密特触发输入，推挽式输出，AN5，SPI 的 MOSI 口，SEG20	I/O
	3	施密特触发输入，推挽式输出，AN6，SPI 的 MISO 口，PWM0，INT0，SEG21	I/O
	4	施密特触发输入，推挽式输出，编程时钟输入，XTOUT	I/O
	5	施密特触发输入，推挽式输出，编程数据输入/输出，XTIN	I/O
	6	施密特触发输入，推挽式输出，BZ，AN7，SEG22	I/O
	7	施密特触发输入，推挽式输出，LVDIN	

＜表 6-1：端口配置总概＞

6.1 I/O 口结构图

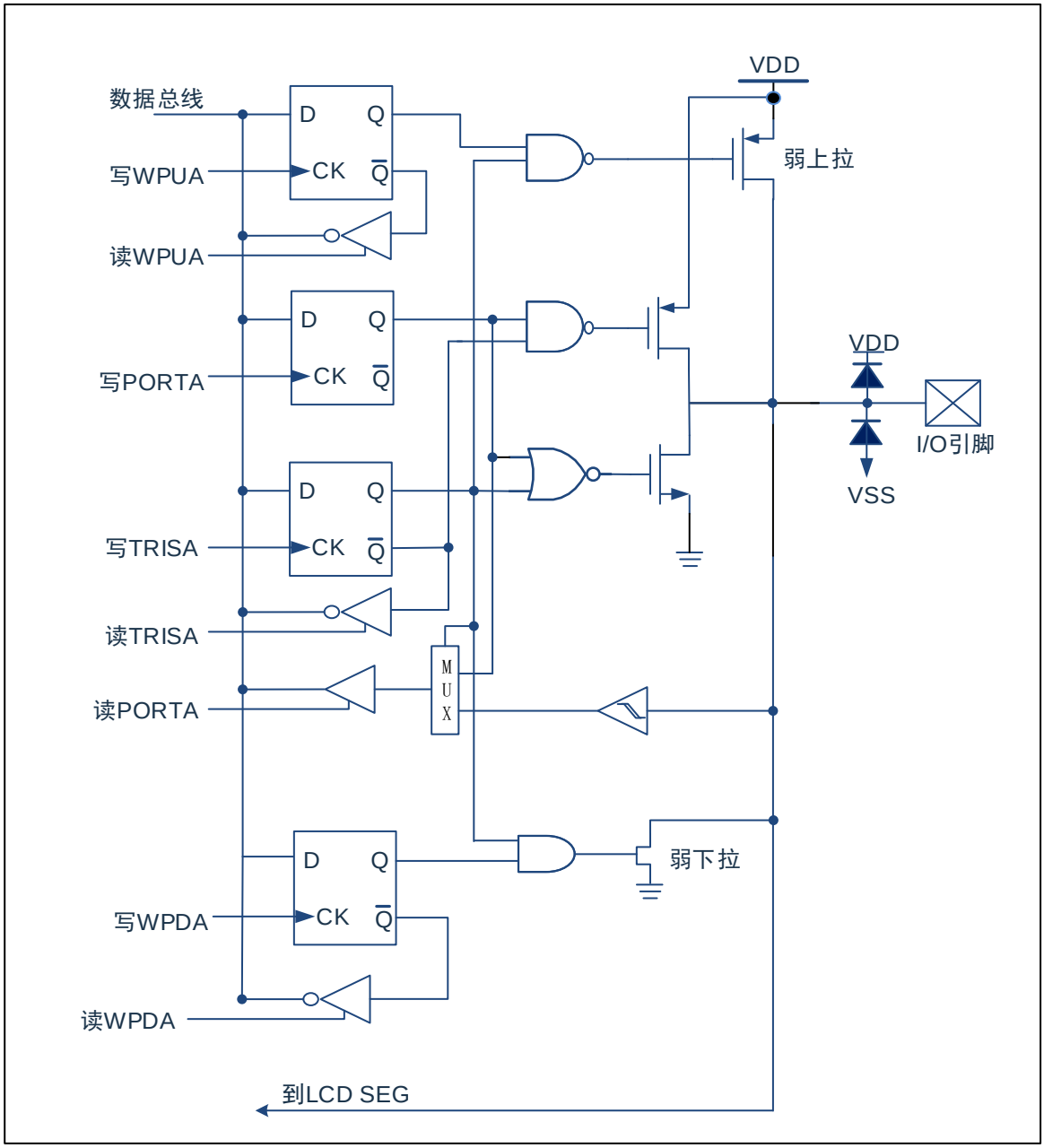


图 6-1：I/O 口结构图—PORTA

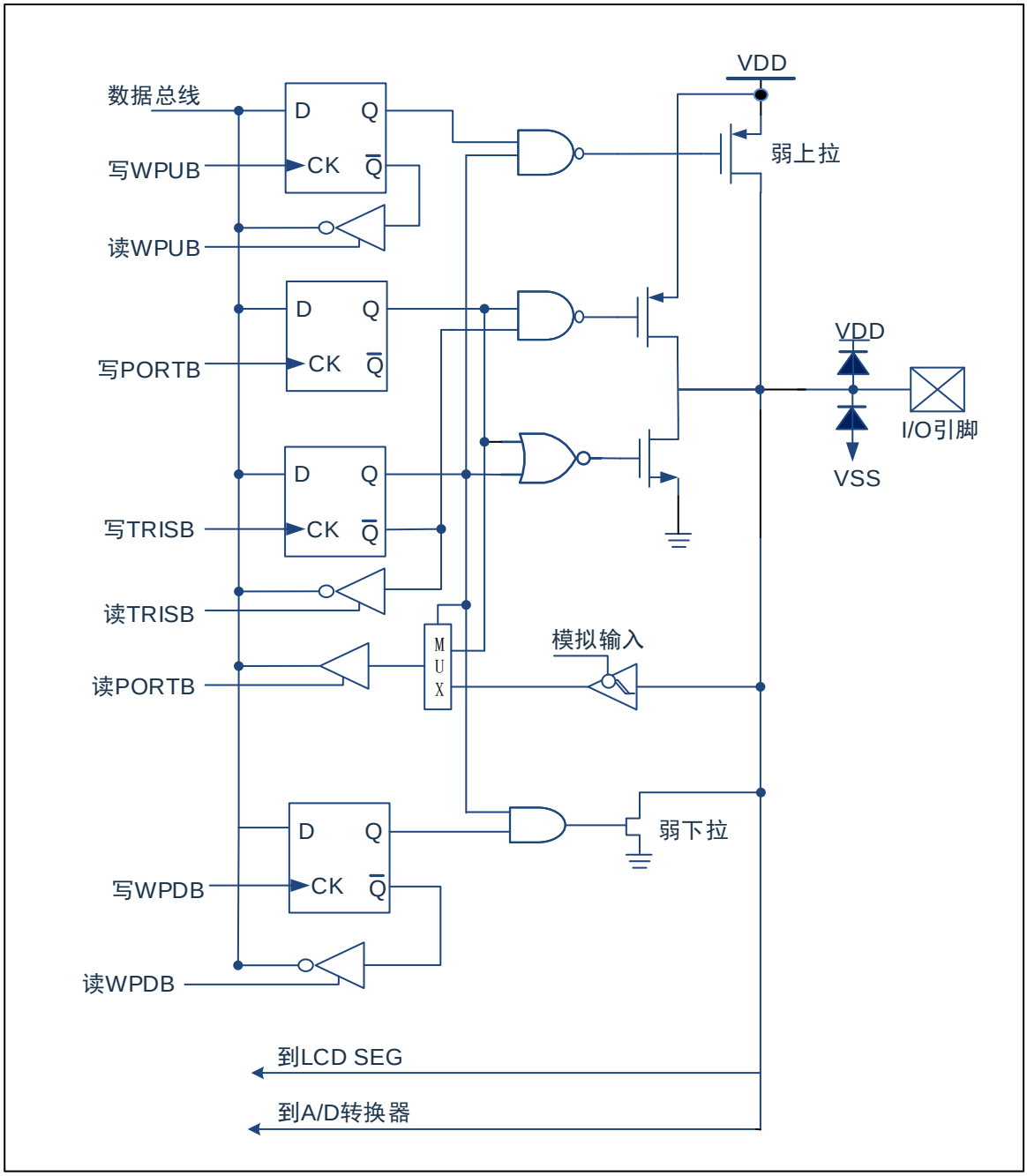


图 6-2：I/O 口结构图—PORTB

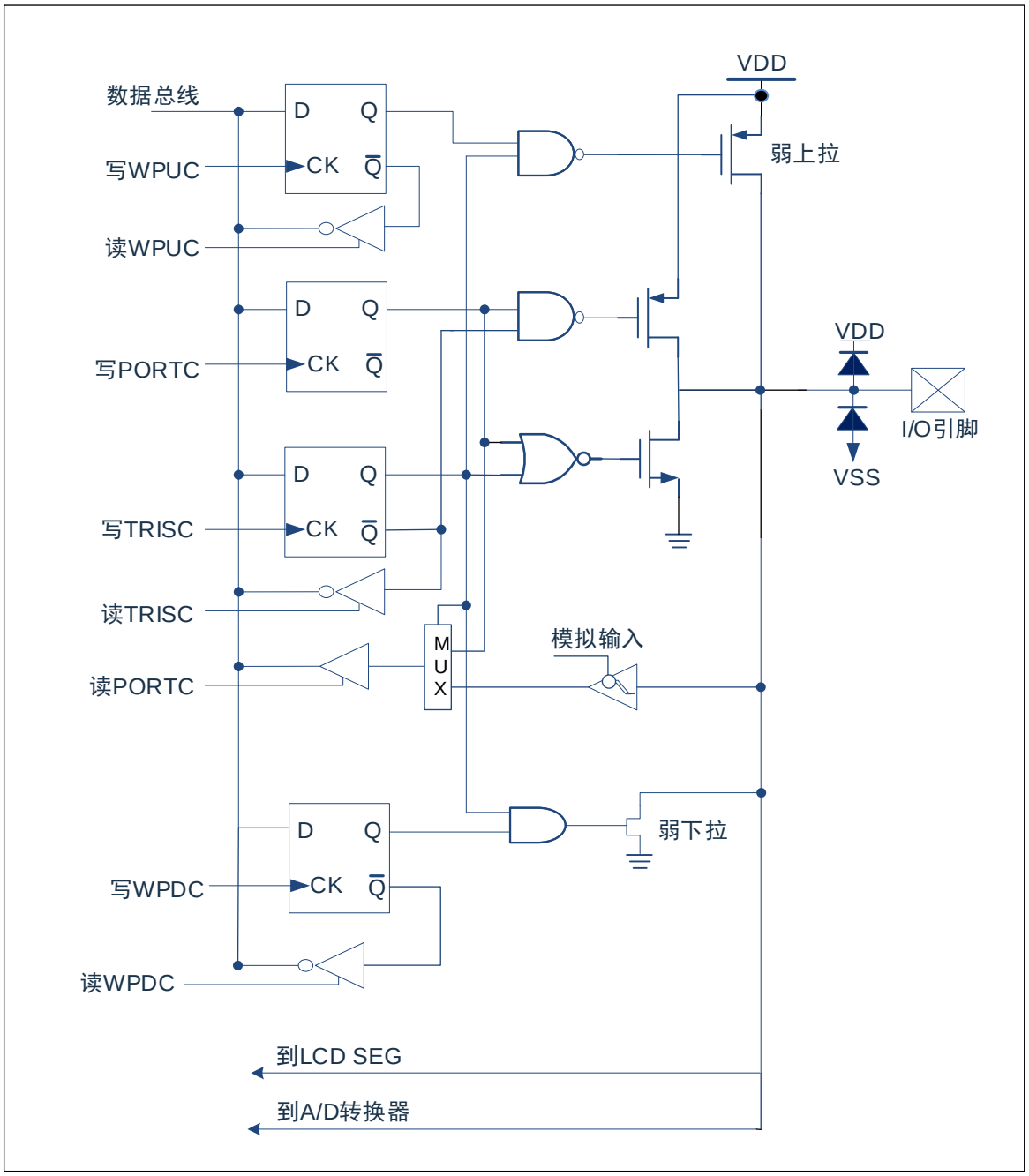


图 6-3：I/O 口结构图—PORTC

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。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、WPDA、PAPI、COMENA、SEGENA、IOSR、PAWP、PAWE 等。

PORTA 数据寄存器 PORTA(0CH)

0CH	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
1=　端口引脚电平>V_{IH}
0=　端口引脚电平<V_{IL}

PORTA 方向寄存器 TRISA(8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISA	TRISA7	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

例：PORTA 口处理程序

```
LDIA    B'11110000'     ;设置PORTA<3:0>为输出口，PORTA<7:4>为输入口
LD      TRISA,A
LDIA    03H             ;PORTA<1:0>输出高电平，PORTA<3:2>输出低电平
LD      PORTA,A         ;由于PORTA<7:4>为输入口，所以赋0或1都没影响
```

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(10CH)

10CH	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，将自动禁止弱上拉。

6.2.3 PORTA 下拉电阻

每个 PORTA 引脚都有可单独配置的内部弱下拉。控制位 WPDA<7:0>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。

PORTA 下拉电阻寄存器 WPDA（10FH）

10FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDA	WPDA7	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0
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 WPDA<7:0>: 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，将自动禁止弱下拉。

6.2.4 PORTA 拉电流控制

PORTA 拉电流控制寄存器 PAPI（116H）

116H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PAPI	PAPI<7:4>				PAPI<3:0>			
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~Bit4 PAPI<7:4>: PORTA<7:4>拉电流设置

0000=	0mA	1000=	24mA
0001=	3mA	1001=	27mA
0010=	6mA	1010=	30mA
0011=	9mA	1011=	33mA
0100=	12mA	1100=	36mA
0101=	15mA	1101=	39mA
0110=	18mA	1110=	42mA
0111=	21mA	1111=	45mA

Bit3~Bit0 PAPI<3:0>: PORTA<3:0>拉电流设置

0000=	0mA	1000=	24mA
0001=	3mA	1001=	27mA
0010=	6mA	1010=	30mA
0011=	9mA	1011=	33mA
0100=	12mA	1100=	36mA
0101=	15mA	1101=	39mA
0110=	18mA	1110=	42mA
0111=	21mA	1111=	45mA

6.2.5 PORTA 拉电流使能控制

PORTA 拉电流使能控制寄存器 SEGENA（191H）

191H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGENA	SEGENA<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 SEGENA<7:0>: PORTA 拉电流使能控制

0= 对应 I/O 的拉电流不能设置大小，I/O 输出为最大拉电流

1= 使能对应的 I/O 拉电流可设置

6.2.6 PORTA 灌电流控制

PORTA 灌电流控制寄存器 COMENA（18FH）

18FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
COMENA	COMENA<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 COMENA<7:0>: PORTA 灌电流设置

0= 82mA

1= 164mA

6.2.7 IO 斜率控制

IO 斜率控制寄存器 IOSR（193H）

193H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOSR	---	---	---	---	IOSRBH	IOSRBL	IOSRAH	IOSRAL
R/W	---	---	---	---	R/W	R/W	R/W	R/W
复位值	---	---	---	---	0	0	0	0

Bit7~Bit4 未用

Bit3 IOSETBH PORTB<7:4>斜率控制
0= 15ns
1= 100ns

Bit2 IOSETBL PORTB<3:0>斜率控制
0= 15ns
1= 100ns

Bit1 IOSETAH PORTA<7:4>斜率控制
0= 15ns
1= 100ns

Bit0 IOSETAL PORTA<3:0>斜率控制
0= 15ns
1= 100ns

6.2.8 PORTA 电平变化唤醒

所有的 PORTA 引脚都可以被单独配置为电平变化唤醒引脚。控制位 PAWP<7:0>允许或禁止每个引脚的该唤醒功能。上电复位时禁止引脚的电平变化唤醒功能。

对于已允许电平变化唤醒的引脚：

- 若器件处于正常工作态，该引脚电平变化（相应的唤醒引脚需设置为输入态，PAWE 寄存器设置电平变化方式）将会把 PIR2 寄存器中的 RAIF 位置 1，如果中断相关使能位有效，则触发 PORTA 电平变化中断。
- 若 MCU 处于休眠态，该引脚电平变化（相应的唤醒引脚需设置为输入态，PAWE 寄存器设置电平变化方式）将会直接把器件从休眠态唤醒，同时 PIR2 寄存器中的 RAIF 位置 1。如果中断相关使能位有效，则器件唤醒后触发 PORTA 电平变化中断。

注：如需打开 PORTA 电平变化中断功能，必须把 INTCON 寄存器的 GIE 位和 PEIE 位，以及 RACIE 置 1

PORTA 唤醒控制寄存器 PAWP(0FH)

0FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PAWP	PAWP7	PAWP6	PAWP5	PAWP4	PAWP3	PAWP2	PAWP1	PAWP0
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　PAWP<7:0>　PORTA 的唤醒控制位
1=　允许电平变化唤醒
0=　禁止电平变化唤醒

PORTA 唤醒控制寄存器 PAWE(13H)

13H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PAWE	PAWE7	PAWE6	PAWE5	PAWE4	PAWE3	PAWE2	PAWE1	PAWE0
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　PAWE<7:0>　PORTA 的唤醒电平变化方式控制位
1=　上升沿
0=　下降沿

6.3 PORTB

6.3.1 PORTB 数据及方向

PORTB 是一个 8Bit 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、WPDB、PBPI、COMENB、SEGENB、IOSR、ANSEL、PBWP、PBWE 等。

PORTB 数据寄存器 PORTB(0DH)

0DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
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 PORTB<7:0>: PORTBI/O 引脚位
1= 端口引脚电平>V_{IH}
0= 端口引脚电平<V_{IL}

PORTB 方向寄存器 TRISB (8DH)

8DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
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 TRISB<7:0>: 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

例：PORTB 口处理程序

```
CLR     PORTB          ;清数据寄存器
LDIA    B'00110000'    ;设置 PORTB<5:4>为输入口，其余为输出口
LD      TRISB,A
```

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<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。

PORTB 上拉电阻寄存器 WPUB(10DH)

10DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
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 WPUB<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，将自动禁止弱上拉。

6.3.3 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<7:0>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（110H）

110H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDB	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0
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 WPDB<7:0>: 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，将自动禁止弱下拉。

6.3.4 PORTB 拉电流控制

PORTB 拉电流控制寄存器 PBPI（18EH）

18EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PBPI	PBPI<7:4>				PBPI<3:0>			
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~Bit4 PBPI<7:4>: PORTB<7:4>拉电流设置

0000=	0mA	1000=	24mA
0001=	3mA	1001=	27mA
0010=	6mA	1010=	30mA
0011=	9mA	1011=	33mA
0100=	12mA	1100=	36mA
0101=	15mA	1101=	39mA
0110=	18mA	1110=	42mA
0111=	21mA	1111=	45mA

Bit3~Bit0 PBPI<3:0>: PORTB<3:0>拉电流设置

0000=	0mA	1000=	24mA
0001=	3mA	1001=	27mA
0010=	6mA	1010=	30mA
0011=	9mA	1011=	33mA
0100=	12mA	1100=	36mA
0101=	15mA	1101=	39mA
0110=	18mA	1110=	42mA
0111=	21mA	1111=	45mA

6.3.5 PORTB 拉电流使能控制

PORTB 拉电流使能控制寄存器 SEGENB（192H）

192H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGENB	SEGENB<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 SEGENB<7:0>: PORTB 拉电流使能控制

0= 对应 I/O 的拉电流不能设置大小，I/O 输出为最大拉电流

1= 使能对应的 I/O 拉电流可设置

6.3.6 PORTB 灌电流控制

PORTB 下拉电阻寄存器 COMENB（190H）

190H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
COMENB	COMENB<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 COMENB<7:0>: PORTB 灌电流设置
0= 82mA
1= 164mA

6.3.7 PORTB 电平变化唤醒

所有的 PORTB 引脚都可以被单独配置为电平变化唤醒引脚。控制位 PBWP<7:0>允许或禁止每个引脚的该唤醒功能。上电复位时禁止引脚的电平变化唤醒功能。

对于已允许电平变化唤醒的引脚：

- 若器件处于正常工作态，该引脚电平变化（PBWE 寄存器设置电平变化方式）将会把 INTCON 寄存器中的 RBIF 位置 1，如果中断相关使能位有效，则触发 PORTB 电平变化中断。
- 若 MCU 处于休眠态，该引脚电平变化（PBWE 寄存器设置电平变化方式）将会直接把器件从休眠态唤醒，同时 INTCON 寄存器中的 RBIF 位置 1。如果中断相关使能位有效，则器件唤醒后触发 PORTB 电平变化中断。

注：如需打开 PORTB 电平变化中断功能，必须把 INTCON 寄存器的 GIE 位和 RBIE 置 1

PORTA 唤醒控制寄存器 PBWP(10H)

10H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PBWP	PBWP7	PBWP6	PBWP5	PBWP4	PBWP3	PBWP2	PBWP1	PBWP0
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 PBWP<7:0> PORTB 的唤醒控制位
1= 允许电平变化唤醒
0= 禁止电平变化唤醒

PORTB 唤醒控制寄存器 PBWE(14H)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PBWE	PBWE7	PBWE6	PBWE5	PBWE4	PBWE3	PBWE2	PBWE1	PBWE0
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 PBWE<7:0> PORTB 的唤醒电平变化方式控制位
1= 上升沿
0= 下降沿

6.3.8 PORTB 模拟选择控制

ANSEL 寄存器用于将 I/O 引脚的输入模式配置为模拟模式。将 ANSEL 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。ANSEL 位的状态对数字输出功能没有影响。TRIS 清零且 ANSEL 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，PORTC 模拟选择寄存器 ANSEL(18CH)

18CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL	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

6.4 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(0EH)

0EH	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
1=　端口引脚电平>V_{IH}
0=　端口引脚电平<V_{IL}

PORTC 方向寄存器 TRISC(8EH)

8EH	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

例：PORTC 口处理程序

```
CLR     PORTC          ;清数据寄存器
LDIA    B'00000010'    ;设置 PORTC<0>为输出口，PORTC<1>为输入口
LD      TRISC,A
```

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(10EH)

10EH	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，将自动禁止弱上拉。

6.4.3 PORTC 下拉电阻

每个 PORTC 引脚都有可单独配置的内部弱下拉。控制位 WPDC<7:0>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。

PORTC 下拉电阻寄存器 WPDC(8FH)

8FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPDC	WPDC7	WPDC6	WPDC5	WPDC4	WPDC3	WPDC2	WPDC1	WPDC0
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　　WPDC<7:0>:　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，将自动禁止弱下拉。

6.4.4 PORTC 模拟选择控制

ANSEL 寄存器用于将 I/O 引脚的输入模式配置为模拟模式。将 ANSEL 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。ANSEL 位的状态对数字输出功能没有影响。TRIS 清零且 ANSEL 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，PORTC 模拟选择寄存器 ANSEL(18CH)

18CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL	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

6.5 I/O 使用

6.5.1 写 I/O 口

芯片的 I/O 口寄存器，和一般通用寄存器一样，可以通过数据传输指令，位操作指令等进行写操作。

例：写 I/O 口程序

```
LD      PORTA,A     ;ACC 值赋给 PORTA 口
CLRB    PORTB,1     ;PORTB.1 口置零
CLR     PORTC       ;PORTC 口清零
SET     PORTA       ;PORTA 所有输出口置 1
SETB    PORTB,1     ;PORTB.1 口置 1
```

6.5.2 读 I/O 口

例：读 I/O 口程序

```
LD      A,PORTA     ;PORTA 的值赋给 ACC
SNZB    PORTA,1     ;判断 PORTA,1 口是否为 1，为 1 跳过下一条语句
SZB     PORTA,1     ;判断 PORTA,1 口是否为 0，为 0 跳过下一条语句
```

注：当用户读一个 I/O 口状态时，若此 I/O 口为输入口，则用户读回的数据将是此口线外部电平的状态，若此 I/O 口为输出口那么读出的值将会是此口线内部输出寄存器的数据。

6.6 I/O 口使用注意事项

在操作 I/O 口时，应注意以下几个方面：

1. 当 I/O 从输出转换为输入时，要等待几个指令周期的时间，以便 I/O 口状态稳定。
2. 若使用内部上拉电阻，那么当 I/O 从输出转换为输入时，内部电平的稳定时间，与接在 I/O 口上的电容有关，用户应根据实际情况，设置等待时间，以防止 I/O 口误扫描电平。
3. 当 I/O 口为输入口时，其输入电平应在“VDD+0.3V”与“GND-0.3V”之间。若输入口电压不在此范围内可采用如下图所示方法。

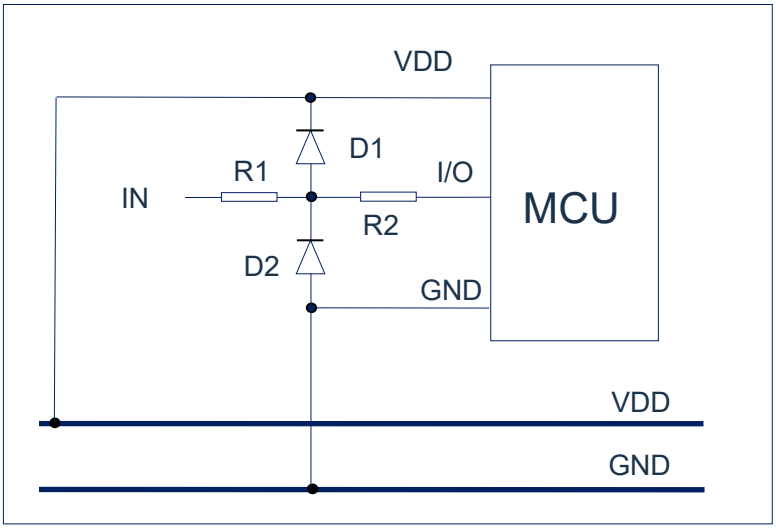


图 6-3：输入电压不在规定范围内采用电路

4. 若在 I/O 口所在线串入较长的连接线，请在靠近芯片 I/O 的地方加上限流电阻以增强 MCU 抗 EMC 能力。

7. 中断

7.1 中断概述

芯片具有以下多种中断源：

- TIMER0 溢出中断
- TIMER2 匹配中断
- SAR ADC 中断
- USART 接收/发送中断
- LVD 中断
- PWM0 中断
- INT 中断
- PORTA 电平变化中断
- PORTB 电平变化中断
- 程序 EEPROM 写操作中断
- AFE 中断
- TIMER1 溢出中断

中断控制寄存器（INTCON）和外设中断请求寄存器（PIR1、PIR2）在各自的标志位中记录各种中断请求。INTCON 寄存器还包括各个中断允许位和全局中断允许位。

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 INTCON、PIE1、PIE2 寄存器中相应的允许位来禁止各个中断。复位时 GIE 被清零。

执行“从中断返回”指令 RETI 将退出中断服务程序，从影子寄存器恢复保存的内容并将 GIE 位置 1，从而重新允许未屏蔽的中断。

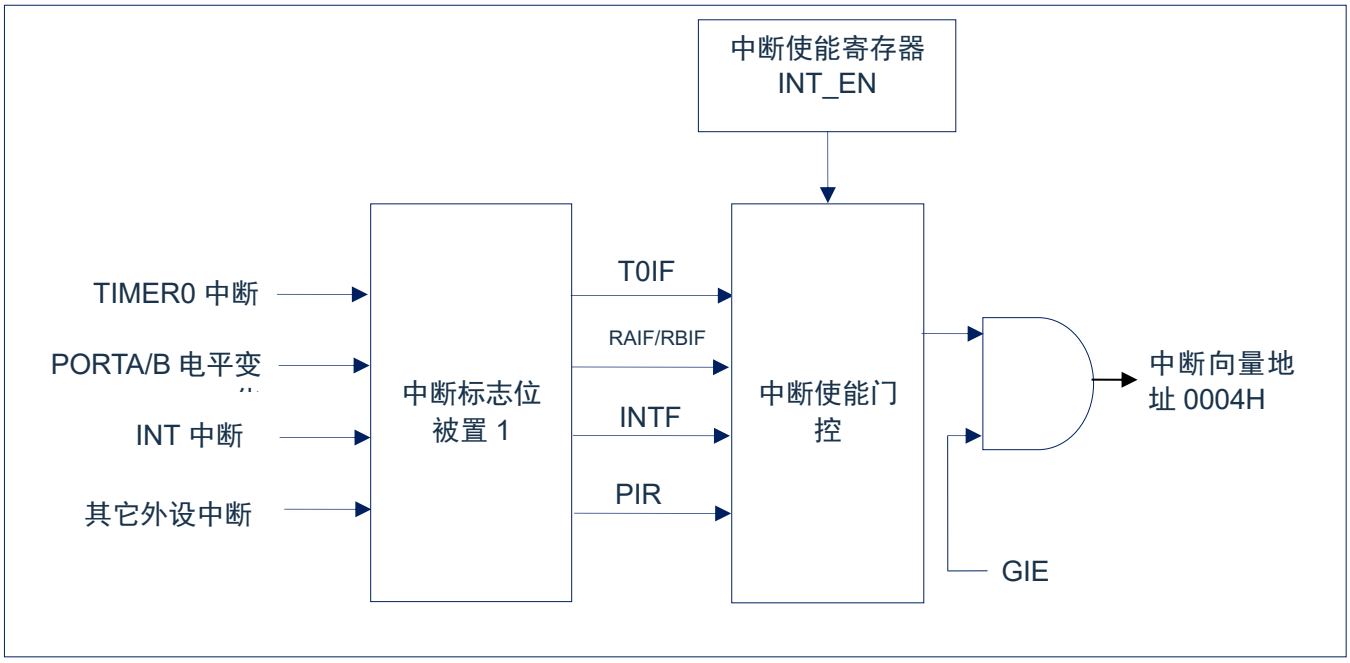


图 7-1：中断原理示意图

7.2 中断控制寄存器

7.2.1 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 TMR0 寄存器溢出、PORTB 端口电平变化中断等的允许和标志位。

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（0BH）

0BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
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 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 PEIE: 外设中断允许位
1= 允许所有未被屏蔽的外设中断
0= 禁止所有外设中断
Bit5 T0IE: TIMER0溢出中断允许位
1= 允许TIMER0中断
0= 禁止TIMER0中断
Bit4 INTE: INT外部中断允许位
1= 允许INT外部中断
0= 禁止INT外部中断
Bit3 RBIE: PORTB电平变化中断允许位（1）
1= 允许PORTB电平变化中断
0= 禁止PORTB电平变化中断
Bit2 T0IF: TIMER0溢出中断标志位（2）
1= TMR0寄存器已经溢出（必须由软件清零）
0= TMR0寄存器未发生溢出
Bit1 INTF: 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：
1. PBWP 寄存器也必须使能，相应的口线需设置为输入态。
2. T0IF 位在 TMR0 计满归 0 时置 1。复位不会使 TMR0 发生改变，应在将 T0IF 位清零前对其进行初始化。

7.2.2 外设中断允许寄存器

外设中断允许寄存器为 PIE，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(91H)

91H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	SPIIE	PWM0IE	RCIE	TXIE	EEIE	ADIE	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 SPIIE: SPI中断允许位
1= 允许SPI中断
0= 禁止SPI中断

Bit6 PWM0IE: PWM0中断允许位
1= 允许PWM中断
0= 禁止PWM中断

Bit5 RCIE: USART接收中断允许位
1= 允许USART接收中断
0= 禁止USART接收中断

Bit4 TXIE: USART发送中断允许位
1= 允许USART发送中断
0= 禁止USART发送中断

Bit3 EEIE: 数据EEPROM写操作中断允许位
1= 允许数据EEPROM写操作中断
0= 禁止数据EEPROM写操作中断

Bit2 ADIE: A/D转换器（SAR ADC）中断允许位
1= 允许SAR ADC中断
0= 禁止SAR ADC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(92H)

92H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE2	---	---	RACIE	---	---	AFEIE	---	LVDIE
R/W	---	---	R/W	---	---	R/W	---	R/W
复位值	---	---	0	---	---	0	---	0

Bit7~Bit6 未用

Bit5 RACIE: PORTA电平变化中断允许位
1= 允许PORTA电平变化中断
0= 禁止PORTA电平变化中断

Bit4~Bit3 未用

Bit2 AFEIE: AFE中断允许位
1= 允许AFE中断
0= 禁止AFE中断

Bit1 未用

Bit0 LVDIE: LVD中断允许位
1= 允许LVD中断
0= 禁止LVD中断

7.2.3 外设中断请求寄存器

外设中断请求寄存器为 PIR1。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(11H)

11H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR1	SPIIF	PWM0IF	RCIF	TXIF	EEIF	ADIF	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 SPIIF: SPI中断标志位
1= 发生了发送/接收中断（必须由软件清零）
0= 没有产生SPI中断条件

Bit6 PWM0IF: PWM0中断标志位
1= 发生了PWM0中断(必须由软件清零)
0= 未发生PWM0中断

Bit5 RCIF: USART接收中断标志位
1= USART接收缓冲器非空（通过读RCREG清零）
0= USART接收缓冲器空

Bit4 TXIF: USART发送中断标志位
1= USART发送缓冲器空（通过写TXREG清零）
0= USART发送缓冲器非空

Bit3 EEIF: EE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

Bit2 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

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

外设中断请求寄存器 PIR2(12H)

12H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR2	---	---	RACIF	---	---	AFEIF	---	LVDIF
R/W	---	---	R/W	---	---	R/W	---	R/W
复位值	---	---	0	---	---	0	---	0

Bit7~Bit6 未用。

Bit5 RACIF: PORTA电平变化中断标志位

1= PORTA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PORTA通用I/O引脚的状态发生了改变

Bit4~Bit3 未用。

Bit2 AFEIF: AFE中断标志位

1= Sigma-Delta ADC 转换完成（必须由软件清零）

0= Sigma-Delta ADC 转换未完成或尚未启动

Bit1 未用

Bit0 LVDIF: LVD中断标志位

1= 电源电压低于LVD设定的电压点（必须由软件清零）

0= 电源电压高于LVD设定的电压点

7.3 中断现场的保护方法

有中断请求发生并被响应后，程序转至 0004H 执行中断子程序。

中断事件的发生会引发以下事件：

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

进入中断时，将返回的PC地址保存在堆栈中，而且以下寄存器的内容自动保存在影子寄存器中：

- ACC工作寄存器
- STATUS寄存器（TO和PD状态标志位除外）
- BSR寄存器
- FSR寄存器
- PCLATCH寄存器

退出中断服务程序时，这些寄存器将自动恢复。影子寄存器在Bank 31中，只可读不可写。根据用户应用程序的要求，可能还需要保存其他寄存器。

7.4 中断的优先级，及多中断嵌套

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 没有预置的中断优先级。首先，必须预先设定好各中断的优先权；其次，利用中断使能位和中断控制位，控制系统是否响应该中断。在程序中，必须对中断控制位和中断请求标志进行检测。

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。

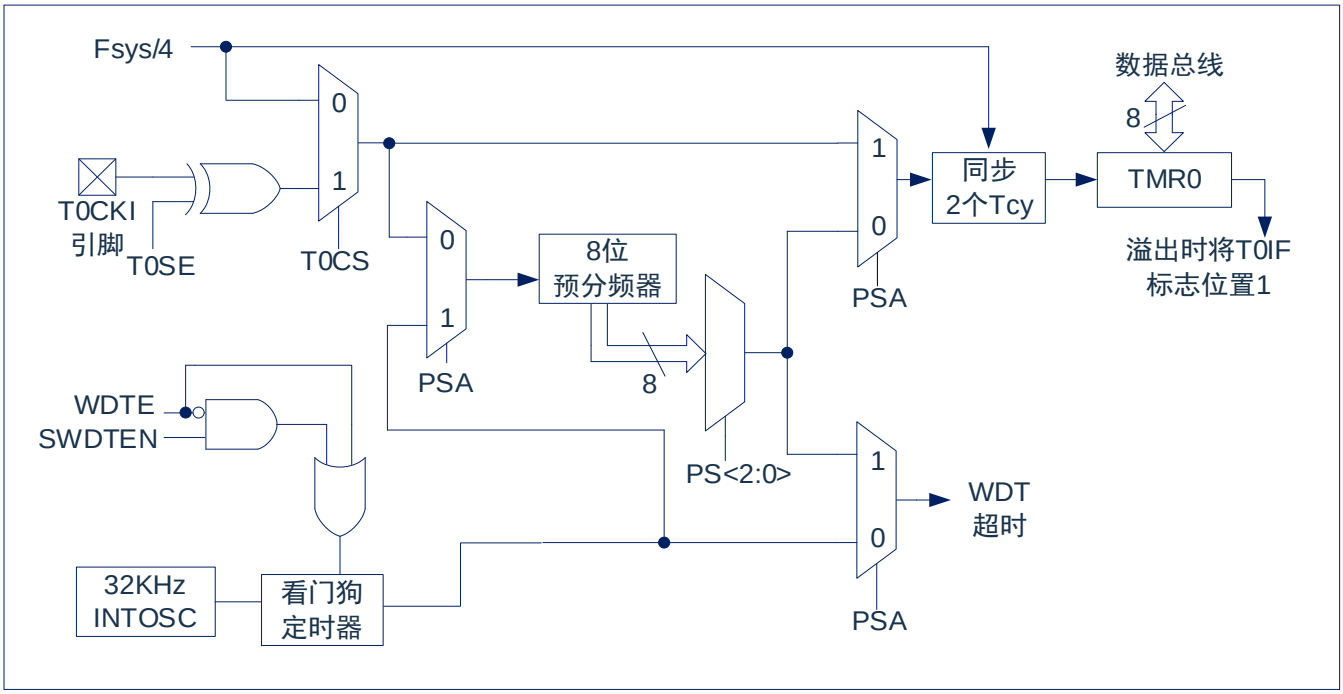


图 8-1：TIMER0/WDT 模块结构图

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. WDTE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，必须执行如下所示的指令序列。

更改预分频器（TMR0-WDT）

```
        CLRB    INTCON,GIE      ;关中断总使能位,避免在执行以下特定时序时进入中断程序
        LDIA    B'00000111'
        ORR     OPTION_REG,A    ;预分频器设置为最大值
        CLR     TMR0            ;TMR0 清零
        SETB    OPTION_REG,PSA  ;设置预分频器分配给 WDT
        CLRWDT                  ;WDT 清零
        LDIA    B'xxxx1xxx'     ;设置新的预分频器
        LD      OPTION_REG,A
        CLRWDT                  ;WDT 清零
        SETB    INTCON,GIE      ;若程序需要用到中断,此处重新打开总使能位
```

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。

更改预分频器（WDT-TMR0）

```
        CLRWDT                  ;WDT 清零
        LDIA    B'00xx0xxx'     ;设置新的预分频器
        LD      OPTION_REG,A
```

8.2.5 TIMER0 中断

当 TMR0 寄存器从 FFh 溢出至 00h 时，产生 TIMER0 中断。每次 TMR0 寄存器溢出时，不论是否允许 TIMER0 中断，INTCON 寄存器的 T0IF 中断标志位都会置 1。T0IF 位必须在软件中清零。TIMER0 中断允许位是 INTCON 寄存器的 T0IE 位。

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 TMR0 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 位只写寄存器，用户可改变 OPTION_REG 的值，来改变 TMR0 的工作模式等。请参看 2.6 关于预分频寄存器（OPTION_REG）的应用。

8 位定时器/计数器 TMR0(15H)

15H	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

OPTION_REG 寄存器(16H)

16H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	---	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W	---	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	1	1	1	1	0	1	1

Bit7　未用。

Bit6　INTEDG:　中断边沿选择位
- 1= INT 引脚的上升沿触发中断
- 0= INT 引脚的下降沿触发中断

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

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

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

Bit2~Bit0　PS2~PS0:　预分配参数配置位

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 定时计数器 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
- 通过比较器或 T1G 引脚门控 TIMER1(使能计数）
- 可编程内部或外部时钟源
- 溢出中断
- 溢出时唤醒（仅外部时钟异步模式）

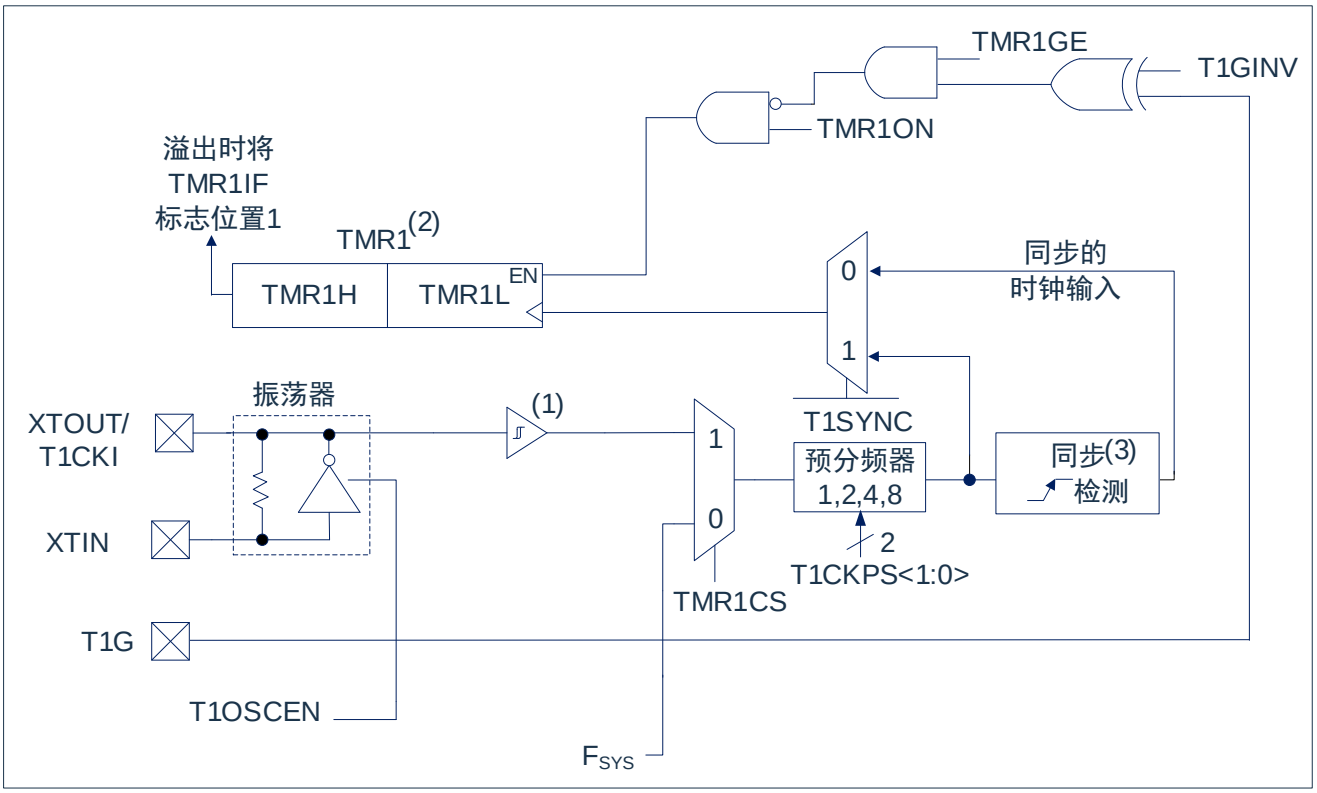


图9-1：TIMER1结构图

注：

1. ST 缓冲器在使用 LP 振荡器时处于低功耗模式，而在使用 T1CKI 时处于高速模式。
2. Timer1 寄存器在上升沿递增。
3. 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 FSYS 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）：

- 在 POR 或 BOR 复位后使能 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。

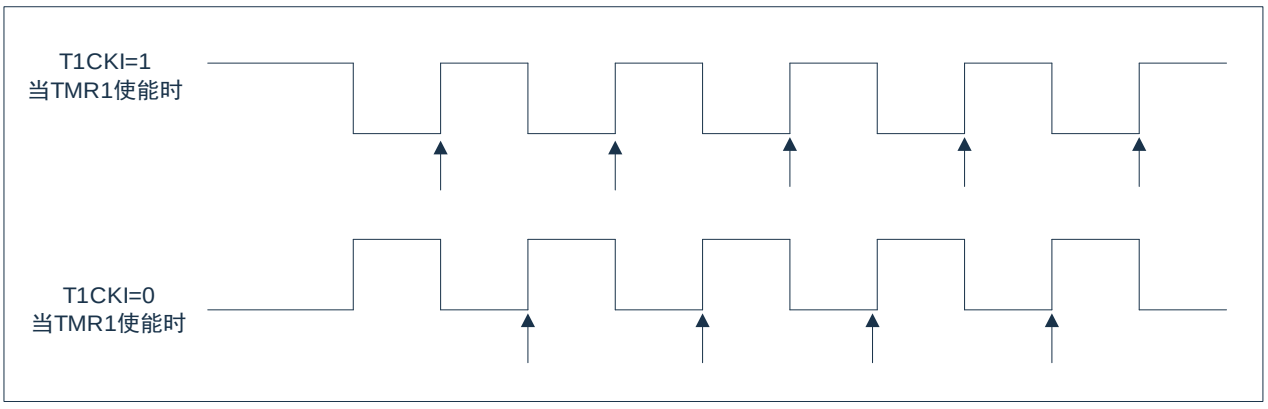


图 9-2：TIMER1 的递增边沿

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，允许对时钟输入进行 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 TIMER1 振荡器

在 XTIN（输入）引脚和 XTOUT（放大器输出）引脚之间连接有一个内置的低功耗 32.768KHz 振荡器。该内部电路要与外部 32.768 kHz 晶振结合使用。将 T1CON 寄存器的 T1OSCEN 控制位置 1 可使能该振荡器。此振荡器将在休眠模式下继续运行，但是必须使 TIMER1 选择为异步计数模式。

TIMER1 振荡器与 LP 振荡器完全相同。用户必须提供软件延时，以保证振荡器正常振荡。

使能 TIMER1 振荡器时 RC4 和 PC5 被置为模拟输入。

9.6 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“9.6.1 异步计数器模式下对 TIMER1 的读写操作”章节）。

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。

9.6.1 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。

9.7 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。

9.8 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。

9.9 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（0004h）。

9.10 TIMER1 控制寄存器

TIMER1 控制寄存器 T1CON(19H)

19H	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 T1OSCEN: LP 振荡器使能控制位
1= 使能 LP 振荡器作为 TIMER1 的时钟源
0= LP 振荡器关闭

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）；

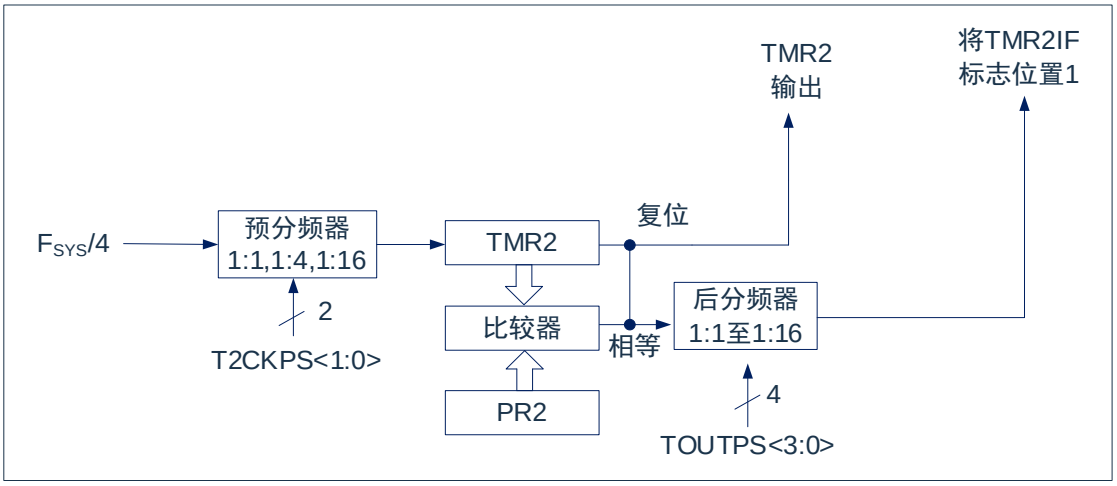


图 10-1：TIMER2 框图

10.2 TIMER2 的工作原理

TIMER2 模块的输入时钟是系统指令时钟（F_{CPU}）。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(1AH)

1AH	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	0	0	0	0	0	0	0	0

TIMER2 周期寄存器 PR2(1BH)

1BH	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(1CH)

1CH	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	
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（SAR 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保持电路。采样保持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 在转换完成之后可以产生一个中断。ADC 模块的参考电压可选 VDD 或者 LDO，LDO 电压可选 2.4V/2.6/3.0V/3.3V。

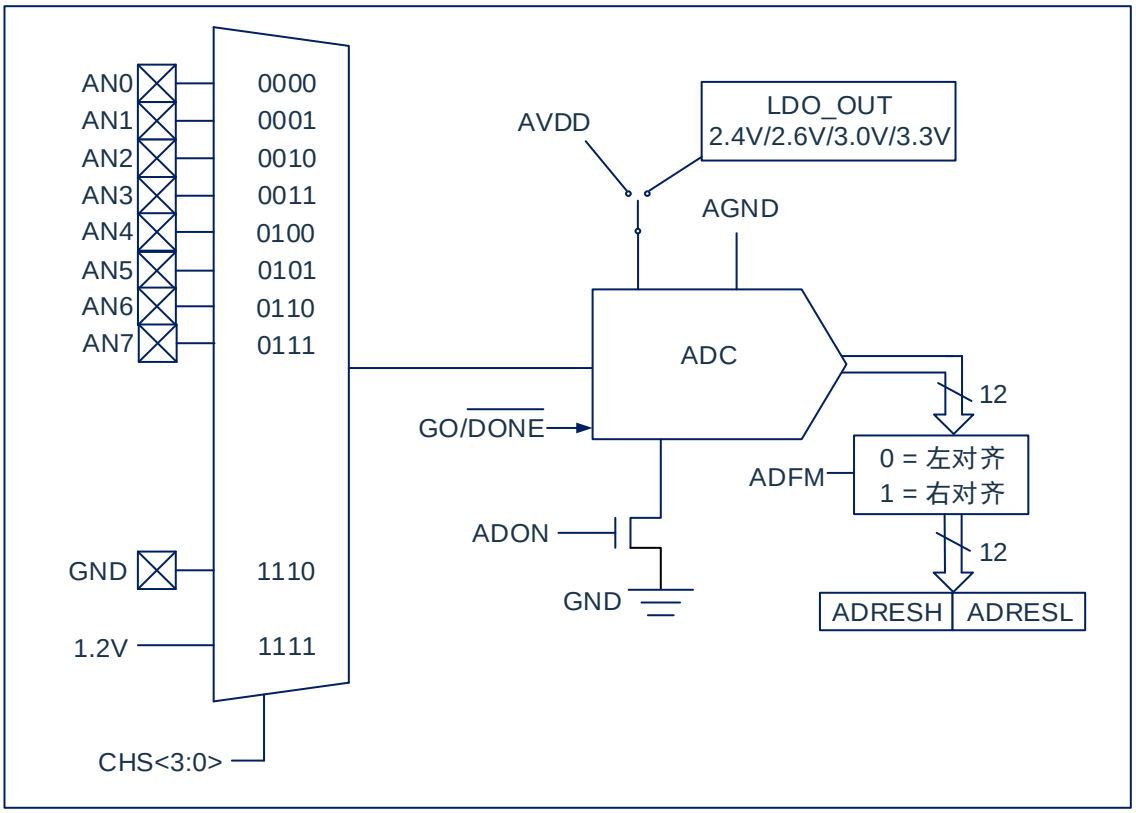


图 11-1：ADC 框图

11.2 ADC 配置

配置和使用 ADC 时，必须考虑如下因素：

- 端口配置；
- 通道选择；
- ADC 参考电压；
- 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“**错误!未找到引用源。**”章节。

11.2.3 ADC 内部基准电压

芯片内置 1.2V 基准电压，需要检测该基准电压时，需把设置 CHS<3:0>位为 1111。

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/2.6V/3.0V/3.3V，ADC 参考电压选择由 AFECON1 寄存器控制。

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。

11.2.5 AD 转换时钟

可以通过软件设置 ADCON0 和 ADCON1 寄存器的 ADCS 位来选择转换的时钟源。有以下 7 种可能的时钟频率可供选择：

- 4MHz
- 2MHz
- 1MHz
- 500KHz
- 250KHz
- 125KHz
- 62.5KHz

完成一位转换的时间定义为 TAD。一个完整的 12 位转换需要 16 个 TAD 周期。

必须符合相应的 TAD 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。

ADC 时钟周期（TAD）与器件工作频率的关系（VDD=3.3V）

ADC 时钟选择		一次 AD 转换时间
ADC 时钟源	ADCS<2:0>	F_{HSI} = 24MHz
4MHz	001	4μs
2MHz	010	8μs
1MHz	011	16μs
500KHz	100	32μs
250KHz	101	64μs
125KHz	110	128μs
62.5KHz	111	256μs

不同参考电压和不同 VDD 时，需要参考以下表格设置合理的时钟频率。

参考电压（V）	工作电压（V）	AD 时钟频率设置	转换速率（ksps）
		F_{HSI} = 24MHz	
VDD	2.5~4.5	4MHz	250
VDD	2.4~2.5	2MHz	125
2.4	2.6~4.5	2MHz	125
2.6	2.8~4.5	4MHz	250
3.0	3.2~4.5	4MHz	250
3.3	3.5~4.5	4MHz	250

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，A/D 转换结果左对齐，A/D 转换结果为 12Bit；当 ADFM=1 时，A/D 转换结果右对齐，A/D 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/$\overline{\text{DONE}}$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/$\overline{\text{DONE}}$位置 1。

11.3.2 完成转换

当转换完成时，ADC 模块将：

- 清零 GO/$\overline{\text{DONE}}$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/$\overline{\text{DONE}}$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 TAD 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 在休眠模式下的工作

ADC 模块休眠模式下不可以工作。

11.3.5 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 参考电压，如果是从 VDD 切换到内部 2.4V/2.6V/3.0V/3.3V 电压，需等待至少 200us 才能开始检测 AD；
   - 选择 AD 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）。

例：AD 转换

```
        LDIA    B'10000000'
        LD      ADCON1,A
        SETB    TRISB,0         ;设置 PORTB.0 为输入口
        LDIA    B'11000001'
        LD      ADCON0,A
        CALL    DELAY           ;延时一段时间
        SETB    ADCON0,GO
        SZB     ADCON0,GO       ;等待 AD 转换结束
        JP      $-1
        LD      A,ADRESH        ;保存 AD 转换结果高位
        LD      RESULTH,A
        LD      A,ADRESL        ;保存 AD 转换结果低位
        LD      RESULTL,A
```

11.4 ADC 相关寄存器

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

AD 控制寄存器 ADCON0(9CH)

9CH	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
ADCS<2:0> ADCS2在ADCON1寄存器
000= 保留
001= 4MHz
010= 2MHz
011= 1MHz
100= 500KHz
101= 250KHz
110= 125KHz
111= 62.5KHz

Bit5~Bit2 CHS<3:0>: 模拟通道选择位
0000= AN0
0001= AN1
0010= AN2
0011= AN3
0100= AN4
0101= AN5
0110= AN6
0111= AN7

1110= ADC输入内部接GND（不需要使能ADON）
1111= 1.2V(内部基准电压)

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，不消耗电流

AD 控制寄存器 ADCON1(9DH)

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	ADFM	---	ADCS2	---	---	---	---	---
R/W	R/W	---	R/W	---	---	---	---	---
复位值	0	---	0	---	---	---	---	---

Bit7 ADFM: A/D转换结果格式选择位
1= 右对齐
0= 左对齐
Bit6 未用
Bit5 ADCS2 与ADCON0的ADCS1~0组合使能
Bit4~Bit0 未用

AD 数据寄存器高位 ADRESH(9FH)，ADFM=0

9FH	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(9EH)，ADFM=0

9EH	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(9FH)，ADFM=1

9FH	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(9EH)，ADFM=1

9EH	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9-2位

注：在 ADFM 等于 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. AFE 模块（Sigma-Delta ADC）

12.1 AFE 概述

AFE 模块内部集成了多路选择器、PGA、24Bit Sigma-Delta ADC、温度传感器、BIM 模块、带隙基准（BG）及 LDO。低噪声 PGA 采用全差分结构，放大倍数可选：2、4、12、16、48、64、128、384。BIM 模块包含正弦波发生器、信号调理及幅值相角检测等功能。芯片还集成了内部温度传感器，BG， LDO 等模块。通过配置多路选择器，可以将不同的信号源（如模拟差分输入通道，BIM 模块输出电压，温感，BG 等）输入给 PGA，其输出可以被 Sigma-Delta ADC 或者 SAR ADC 转换。Sigma-Delta ADC 可以将模拟输入差分信号转换为表示该信号的一个 24 位二进制数，并将该结果保存在 ADC 结果寄存器（AFERES2，AFERES1，AFERES0）中。AFE 模块的电源可由 VDD 或者低噪声 LDO 供电，其中 LDO 可为内部模拟电路、外部电路提供 20mA 电流。

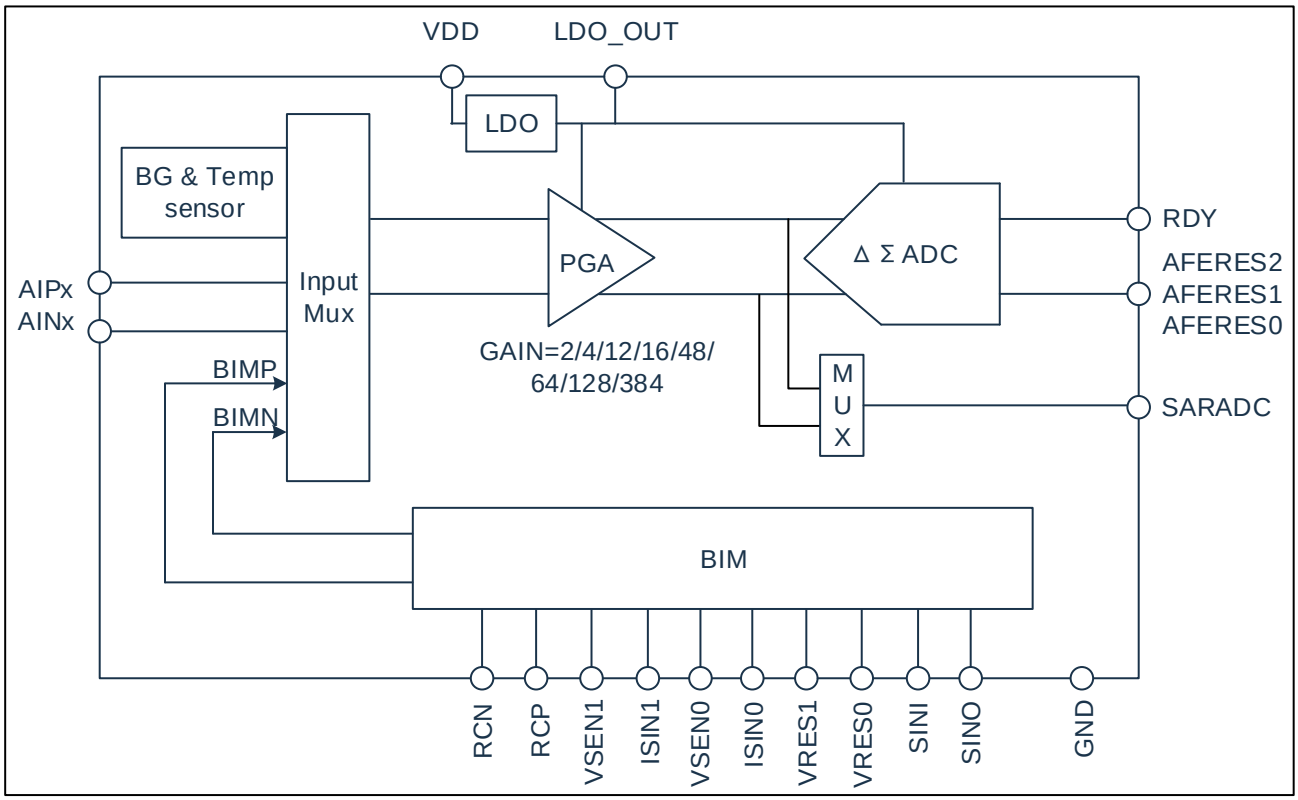


图 12-1：AFE 框图

12.2 LDO

AFE 包含一路 LDO 给片内模拟模块供电，同时也可以为片外电路提供 20mA 电流。VS/VREFP 端口需外接至少 1uF 电容。可以通过配置 AFECON1 寄存器控制 LDO 的各个功能。通过 SET_LDO<1:0>来设置 LDO 输出电压，可以选择 2.4V、2.6V、3V、3.3V。通过设置 BYPASSLDO 来配置 LDO 工作为开关状态或者线性稳压状态。当 ENBLDO 设置为 1，ENBAFE 设置为 1，且将 BYPASSLDO 设置为 1，此时 VS/VREFP 输出 0V，AFE 才能进入深度休眠态。通过 OCP_DIS_LDO 可以使能 LDO 的过流保护功能，当输出电流超过大约 100mA 以上时，过流保护功能被触发。

12.3 PGA 与 Sigma-Delta ADC

AFE 包含一路基于斩波技术的低噪声、低漂移的 PGA 放大器。可以通过配置 AFECON2，AFECON3 来配置 PGA 各项功能。可将 LOWPGA 设置为 1，获得更低的功耗。可通过 PGA_SEL<2:0>来配置 2 、4、12、16、48、64 、128、384 等不同的增益。当使用 PGA=2，4，12 时，第一级低噪声 PGA 放大器会被关断以节省功耗。当 PGA 增益被设置为 16、48 时，第一级低噪声 PGA 的增益为 4；当 PGA 增益被设置为 64、128、384 时，第一级低噪声 PGA 的增益为 32。当使用低噪声 PGA 放大器时，应保证输入信号范围以及第一级输出在 GND+0.75V 到 VDD-1V 之间，超出这个范围，会导致实际性能下降。当模拟输入跳过 PGA，直接输入至 ADC 时，增益为 1。可以通过 CHSEL <3:0> 选择接入 PGA 的不同通道。

通过配置 AFECON4，AFECON4 可以配置 Sigma-Delta ADC 的输出速率，详见寄存器说明。

不同 PGA 增益和 Sigma-Delta ADC 输出速率下的有效分辨率可见 AFE 电气参数章节。

12.4 PGA 与 SAR ADC

该芯片支持直接由 SAR ADC 转换 PGA 的输出信号。该功能可以满足中等精度、中等速度、低功耗的信号检测需求。例如借助该功能完成电子秤周期性的快速上秤监测功能，可以显著降低电子秤的平均功耗。

使用该功能时，须将 ENCHOPB 置 1，关闭 PGA 的斩波功能。将 PGA 设置为适合的增益后，等待 100us PGA 的建立时间。之后启动 SAR ADC，分别将 ENSAR<1:0>设置为 11、10，即转换得出 PGA 的同相输出和反相输出电压，两者相减即可得到 PGA 的差分输出信号。需要注意的是，由于关闭了 PGA 斩波功能，此时 PGA 的失调电压较大；如需消除失调电压，则可以先转换出 CH_SEL<2:0>=000 时的差分信号，再通过配置 CH_SEL<2:0>=001，使能系统斩波功能，得出另一个差分输出信号；将前者减去后者后再除以 2，即可得到极低失调电压的转换结果。

12.5 温度传感器

芯片内部提供温度测量功能。建议采用 PGA 增益 4，ADC 速度为 640Hz 的配置。温度传感器需要进行单点校正。校正方法：已知某个温度点 A 下，使用温度传感器进行测量得到温度电压值为 Ya。测得其他温度点 B 对应的温度电压值为 Yb，则有两种方法得出实际温度 B，一种为 A+（Yb-Ya）/275u；另一种为 Yb*(273.15+A)/Ya-273.15。A 温度单位是摄氏度。

对于参考电压可能变化的应用场合（如 Ratio Metric 测量，参考电压的绝对值不重要），可选择通过测量 BG 间接测量出实时的参考电压，进而计算出实时的温度。

12.6 人体阻抗测量（BIM)

芯片可提供人体交流阻抗及相角测量功能，其原理是将人体等同为一个阻容网络，然后让电流流过该网络，产生一个和该网络阻抗成正比的电压，通过 ADC 测量该电压来换算该网络的等效阻抗。BIM 模块结构如图 12-2 所示，正弦发生器产生的正弦信号经过 C_0、R_0 隔直限流转为正弦电流后，通过接触人体两个不同部位的激励电极 ISIN0、ISIN1，在人体等效阻容网络形成一个电压降，进而通过 VSEN0，VSEN1 测量电极量取该压降后，经过整流滤波处理后送入 ADC 转换为数字信号，从而换算得出人体阻抗。本芯片支持 IQ 模式测量阻容网络的相角。

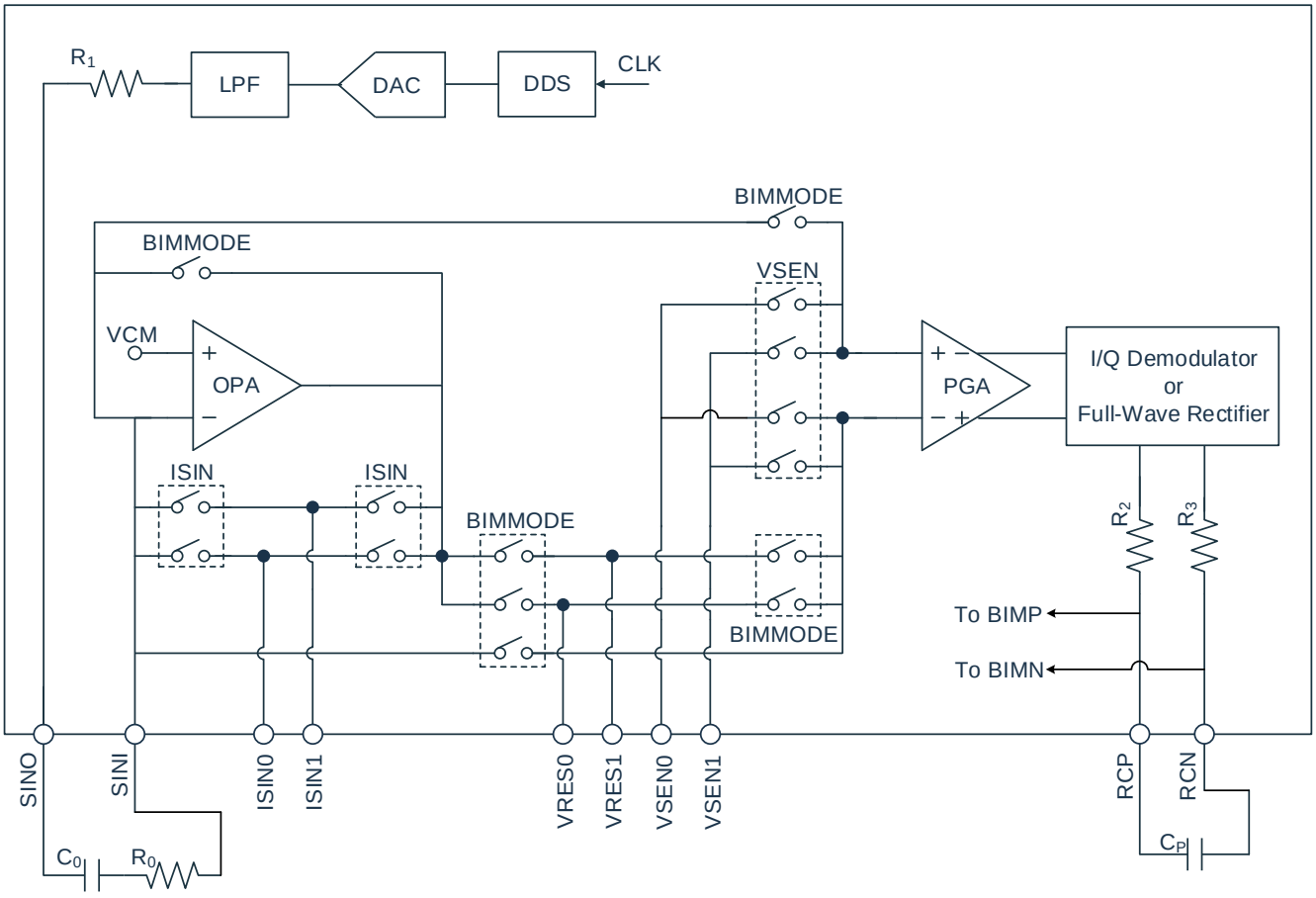


图 12-2：BIM 模块结构图

12.7 AFE 的使能和关闭步骤

使能步骤：

1）SET_LDO=01（2.4V）

2）BYPASSLDO=0

3）ENBLDO=0，延时 100us

4）SET_LDO=xx（设置成实际需要的 LDO 电压，若设置的电压和 2.4V 不同，需再延时 50us）

5）ENBAFE=0

关闭步骤：

1）ENBAFE=1

2）ENBLDO=1

3）BYPASSLDO=1

注：LDO 从关闭态切换到使能态时，由于 VS/VREFP 管脚上的电容需从 0V 充电到 LDO 设定电压，VDD 管脚的瞬间流入电流会很大，为避免 VDD 电压被拉低导致芯片复位，建议在 PCB 板上靠近芯片 VDD 管脚的位置放 4.7uF 以上的电容，同时尽可能减小 VDD/GND 与电池（电源）之间的连线电阻。

12.8 AFE 相关寄存器

AFE 控制寄存器 AFECON1(196H)

196H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
AFECON1	BYPASSLDO	ENBLDO	SET_LDO<1:0>		OCP_DIS_LDO	ADC_ST	CLK_DIV	RDY
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	0	0	0	0	0	0	0

Bit7 BYPASSLDO VS/VREFP输出选择位（**当ENBLDO=1时，BYPASSLDO必须为1，AFE才进入低功耗休眠态**）
0= VS/VREFP输出稳压，由SET_LDO位决定
1= ENBLDO=0时，VS/VREFP输出VDD；ENBLDO=1时，VS/VREFP输出0V

Bit6 ENBLDO LDO使能
0= 使能LDO，VS/VREFP输出稳压或VDD
1= 禁止LDO，VS/VREFP输出0V

Bit5~Bit4 SET_LDO<1:0> LDO电压设置
00= 3.0V
01= 2.4V
10= 2.6V
11= 3.3V

Bit3 OCP_DIS_LDO LDO输出过流保护使能控制
0= 使能
1= 禁止

Bit2 ADC_ST ADC时钟使能位
0= 禁止
1= 使能，开始转换

Bit1 CLK_DIV 时钟分频
0= 8M/6
1= 8M/8

Bit0 RDY ADC转换状态位
0= 正在转换
1= 转换完成，需软件清零

AFE 控制寄存器 AFECON2(197H)

197H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
AFECON2	ENBAFE	LPWRPGA	PGA_SEL<2:0>			ENSAR<1:0>		ENCHOPB
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	0	1	1	0	0	0	0

Bit7 ENBAFE 模拟前端使能位（**使能模拟前端应滞后于使能LDO**）

0= 使能，若使用SAR ADC转换PGA输出，则需等待100us的PGA建立时间后，才使能ADC时钟开始转换；若使用Sigma-Delta ADC，则无需设定额外等待时间。

1= 禁止

Bit6 LPWRPGA PGA功耗选择

0= 正常功耗

1= 低功耗

Bit5~Bit3 PGA_SEL<2:0> PGA增益选择

0000= 2

0001= 4

0010= 12

0011= 16

0100= 48

0101= 64

0110= 128

0111= 384

Bit2~Bit1 ENSAR<1:0> PGA输出连接设置

0x= PGA输出至Sigma-Delta ADC

10= PGA反相输出端至SAR ADC，自动屏蔽其它ADC通道

11= PGA同相输入端至SAR ADC，自动屏蔽其它ADC通道

Bit0 ENCHOPB 斩波使能

0= 打开斩波功能

1= 关闭斩波功能

AFE 控制寄存器 AFECON3(198H)

198H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
AFECON3	CH_SEL<3:0>				---	---	---	EN_ADTESTGN
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 CH_SEL<3:0> 通道选择
0000= 通道AIP/AIN
0001= 通道AIP/AIN正负交换（系统斩波）
0010= 温度
0011= 内短
0100= 通道AIP/AIN直通ADC并关PGA
0101= 通道AIP/AIN正负交换直通ADC并关PGA
0110= BG
0111= 内短直通ADC
1000= 通道BIM
1001= 通道BIM正负交换（系统斩波）
1010= 温度
1011= 内短
1100= 通道BIM直通ADC并关PGA
1101= 通道BIM正负交换直通ADC并关PGA
1110= BG
1111= 内短直通ADC

Bit3 未用。

Bit2 保留，需为0

Bit1 未用。

Bit0 EN_ADTESTGN 接入1/600（VREFP-VREFN）信号至输入端口
0= 不使能
1= 使能

AFE 控制寄存器 AFECON4(199H)

199H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
AFECON4	OSR<2:0>			FCHOP_ADC	FADC	LPWR	---	---
R/W	R/W	R/W	R/W	R/W	R/W	R/W	---	---
复位值	1	1	0	0	1	1	---	---

Bit7~Bit5 OSR<2:0> OSR设置，OSR和FADC共同决定ADC的输出数据速率ODR
000= FADC=1, ODR=5.2ksps；FADC=0，ODR=10.4ksps
001= FADC=1, ODR=2.6ksps；FADC=0，ODR=5.2ksps
010= FADC=1, ODR=1.3ksps；FADC=0，ODR=2.6ksps
011= FADC=1, ODR=651sps；FADC=0，ODR=1.3ksps
100= FADC=1, ODR=163sps；FADC=0，ODR=326sps
101= FADC=1, ODR=40.7sps；FADC=0，ODR=81.4sps
110= FADC=1, ODR=20.3sps；FADC=0，ODR=40.7sps
111= FADC=1, ODR=10.2sps；FADC=0，ODR=20.3sps

Bit4 FCHOP_ADC ADC斩波频率选择
0= 16分频
1= 32分频

Bit3 FADC ADC采样速率
0= 1333.3kHz
1= 666.7kHz

Bit2 LPWR ADC功耗选择位
0= ADC 正常功耗
1= ADC 低功耗

Bit1~Bit0 未用

AFE 数据寄存器 AFERES0(19AH)

19AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
AFERES0	AFERES<7:0>							
R/W	R	R	R	R	R	R	R	R
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0 ADRES<7:0>: AFE结果寄存器位
24位转换结果的低8位

AFE 数据寄存器 AFERES1(19BH)

19BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
AFERES1	AFERES<15:8>							
R/W	R	R	R	R	R	R	R	R
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0 AFERES<15:8>: AFE结果寄存器位
24位转换结果的第15位到第8位

AFE 数据寄存器 AFERES2(19CH)

19CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
AFERES2	AFERES<23:16>							
R/W	R	R	R	R	R	R	R	R
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0 AFERES<23:16>: AFE结果寄存器位

24位转换结果的高8位，其中AFERES<23>为符号位，0为正，1为负

AFE 控制寄存器 BIMCON1(19DH)

19DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BIMCON1	EN_BIM	ENB_SIN	BIMMODE<1:0>		ENB_CHOPBIM	BW_BIM	ISIN	VSEN
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 EN_BIM BIM使能位
0= 关闭BIM模块(默认)
1= 使能BIM模块
Bit6 ENB_SIN 正弦发生器控制位
0= 正弦发生器使能位由EN_BIM控制
1= 正弦发生器关闭
Bit5~Bit4 BIMMODE<1:0> BIM工作模式
00= 测量模式
01= 校准电阻0模式
10= 校准电阻1模式
11= BIM上称检测
Bit3 ENB_CHOPBIM BIM全差分放大器斩波使能
0= 使能
1= 禁止
Bit2 BW_BIM BIM全差分运放带宽选择
0= 高带宽
1= 低带宽
Bit1 ISIN 正弦电流通道选择位
0= ISEN0输出，ISEN1输入
1= ISEN0输入，ISEN1输出
Bit0 VSEN 电压检测通道选择位
0= VSEN0为正输入端，VSEN1为负输入端
1= VSEN0为负输入端，VSEN1为正输入端

AFE 控制寄存器 BIMCON2(19EH)

19EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BIMCON2	ENB_DDL	EN_BIMBW	SEL_CLK<1:0>		EN_SINRES	FREQ_SIN<2:0>		
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	0	0	0	0

Bit7 ENB_DDL: 自适应比较器DDL使能位
0= 使能自适应比较器DLL模块
1= 不使能自适应比较器DDL模块
Bit6 EN_BIMBW: BIM模块放大器带宽设置
0= 高带宽
1= 低带宽
Bit5~Bit4 SEL_CLK<1:0>: BIM调制器的时钟源选择位
00= 同相时钟
01= 正交时钟
1x= 全波整流时钟
Bit3 EN_SINRES: BIM正弦波电压转电流内置电阻的使能信号
0= 不使能
1= 使能
Bit2~Bit0 FREQ_SIN<2:0> 正弦波输出频率设置位
000= 5K
001= 10K
010= 25K
011= 50K
100= 100K
101= 125K
110= 250K
111= 500K

13. 通用同步/异步收发器(USART)

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。

USART 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 USART 收发器的框图。

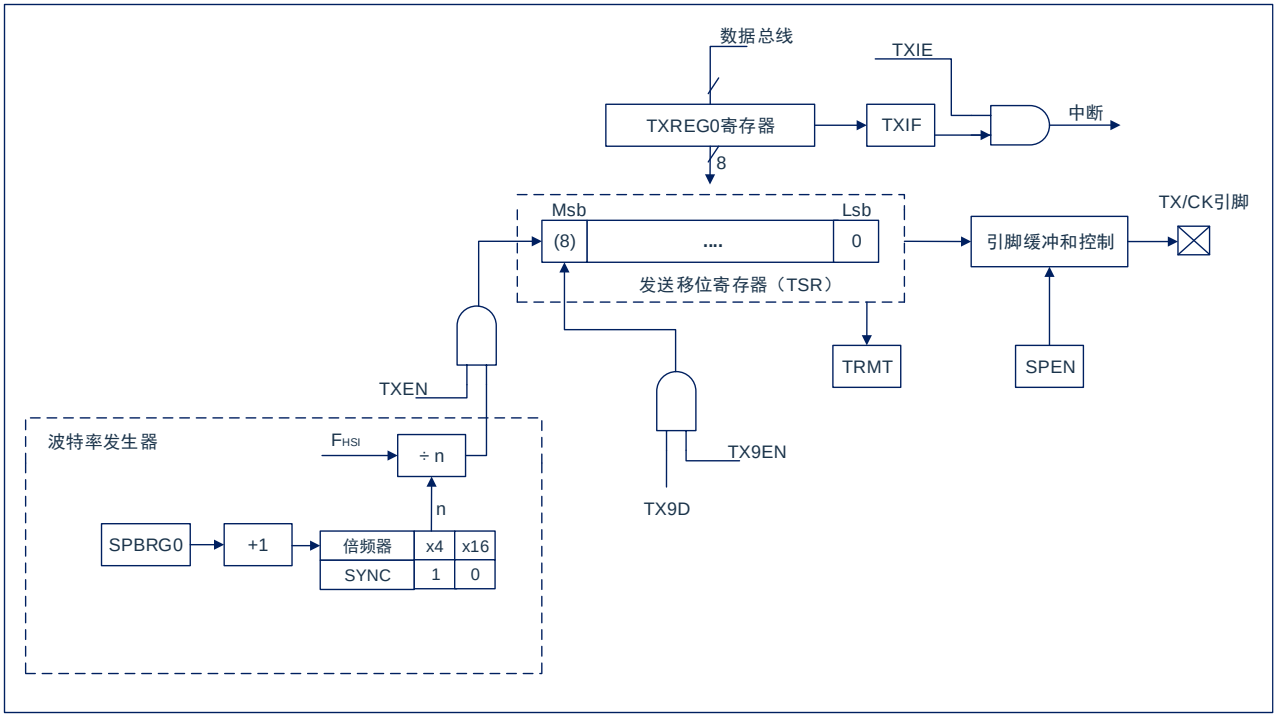


图13-1：USART发送框图

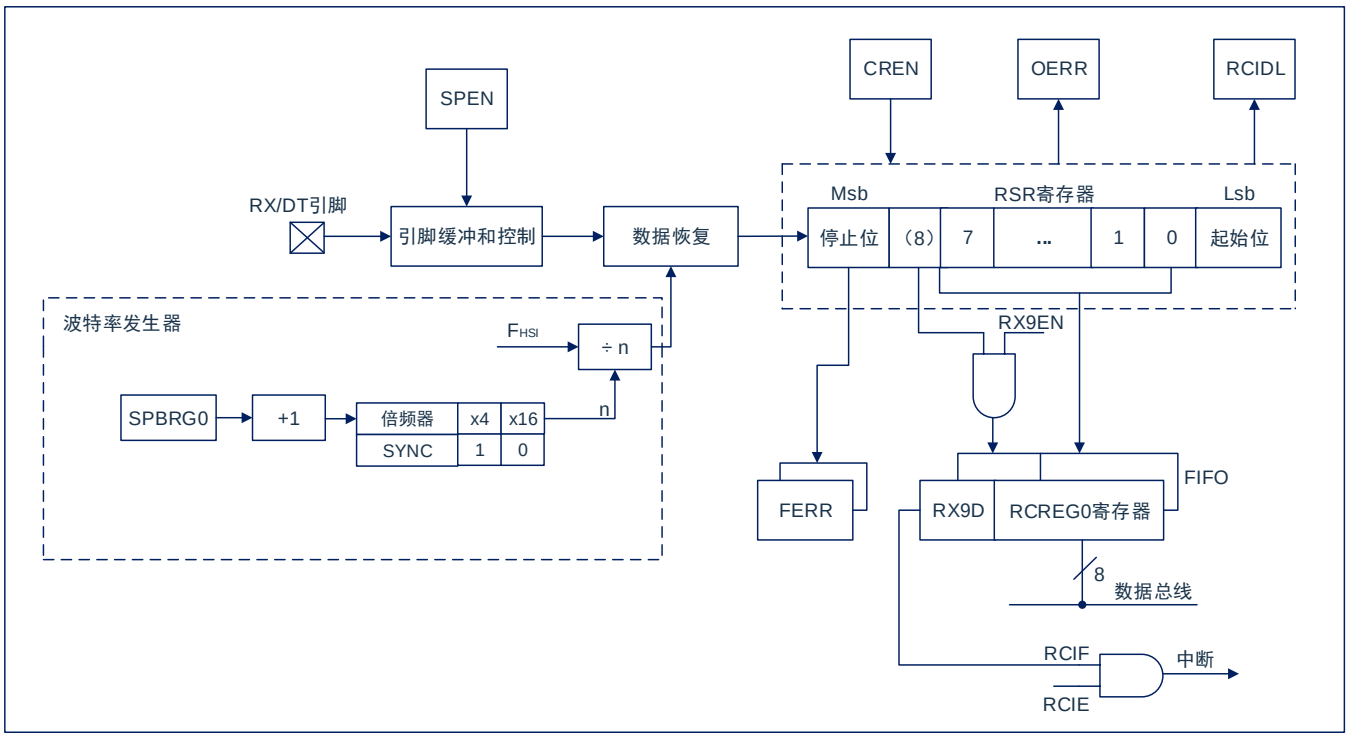


图13-2：USART接收框图

USART 模块的操作是通过 2 个寄存器控制的：

- 发送状态和控制寄存器（TXSTA）
- 接收状态和控制寄存器（RCSTA）

13.1 USART 异步模式

USART 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。

USART 首先发送和接收 Lsb。USART 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 USART 异步发生器

图 13-1 所示为 USART 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 TXREG 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 USART 发送器，以用于异步操作：

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

假设所有其他 USART 控制位处于其默认状态。

将 TXSTA 寄存器的 TXEN 位置 1，使能 USART 发送器电路。将 TXSTA 寄存器的 SYNC 位清零，将 USART 配置用于异步操作。

注：
1. 当将 SPEN 位和 TXEN 位置 1，SYNC 位清零，TX/CK 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 SPEN 位和 CREN 位置 1，SYNC 位清零，RX/DT 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 TXREG 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，TXREG 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 TXREG 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，TXREG 中待处理的数据将被传输到 TSR。当数据从 TXREG 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 USART 发送器且 TXREG 中没有待发送数据，就将 PIR1 寄存器的 TXIF 中断标志位置 1。换句话说，只有当 TSR 忙于处理字符和 TXREG 中有排队等待发送的新字符时，TXIF 位才处于清零状态。写 TXREG 时，不立即清零 TXIF 标志位。TXIF 在写指令后的第 2 个指令周期清零。在写 TXREG 后立即查询 TXIF 会返回无效结果。TXIF 为只读位，不能由软件置 1 或清零。

可通过将 PIE1 寄存器的 TXIE 中断允许位置 1 允许 TXIF 中断。然而，只要 TXREG 为空，不管 TXIE 允许位的状态如何都会将 TXIF 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 TXIE 位置 1。当将待发送的最后一个字符写入 TXREG 后，将 TXIE 中断允许位清零。

13.1.1.4 TSR 状态

TXSTA 寄存器的 TRMT 位指示 TSR 寄存器的状态。TRMT 位为只读位。当 TSR 寄存器为空时，TRMT 位被置 1，当有字符从 TXREG 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

USART 支持 9 位字符发送。当 TXSTA 寄存器的 TX9EN 位置 1 时，USART 将移出每个待发送字符的 9 位。TXSTA 寄存器的 TX9D 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 TXREG 之前，写 TX9D 数据位。在写入 TXREG 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 SPBRG 寄存器，以获得所需的波特率（请参见“USART 波特率发生器（BRG）”章节）。
2. 通过将 SYNC 位清零并将 SPEN 位置 1 使能异步串口。
3. 如果需要 9 位发送，将 TX9EN 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 TXEN 控制位置 1，使能发送；这将导致 TXIF 中断标志位置 1。
5. 如果需要中断，将 PIE1 寄存器中的 TXIE 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 TX9D 数据位。
7. 将 8 位数据装入 TXREG 寄存器开始发送数据。

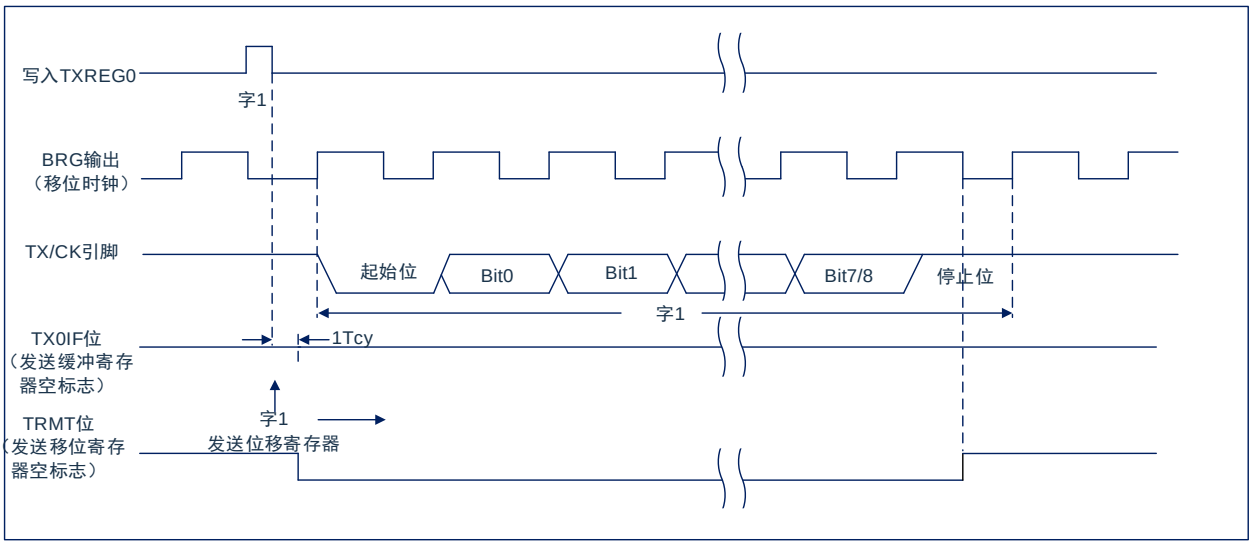


图 13-3：异步发送

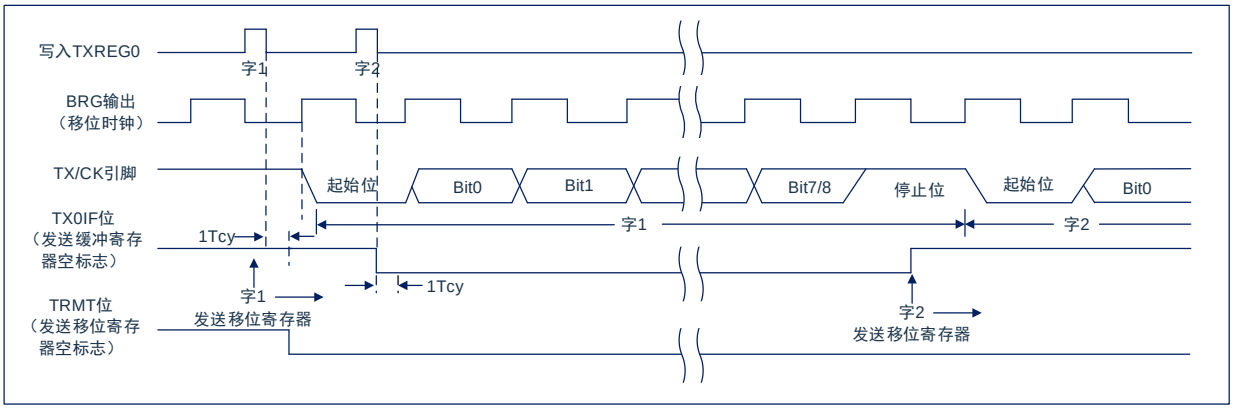


图 13-4：异步发送（背靠背）

注：本时序图显示了两次连续的发送。

13.1.2 USART 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 RX/DT 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 USART 接收器。FIFO 和 RSR 寄存器不能直接由软件访问。通过 RCREG 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 USART 接收器，以用于异步操作。

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

假设所有其他 USART 控制位都处于默认状态。将 RCSTA 寄存器的 CREN 位置 1，使能 USART 接收器电路。将 TXSTA 寄存器的 SYNC 位清零，配置 USART 以用于异步操作。

注：
1. 当将 SPEN 位和 TXEN 位置 1，SYNC 位清零，TX/CK 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 SPEN 位和 CREN 位置 1，SYNC 位清零，RX/DT 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 USART 的接收 FIFO 并将 PIR1 寄存器的 RCIF 中断标志位置 1。通过读 RCREG 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 USART 接收器且在接收 FIFO 中没有未读数据，PIR1 寄存器中的 RCIF 中断标志位就会清零。RCIF 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 RCIF 中断：

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

如果 FIFO 中有未读数据，无论中断允许位的状态如何，都会将 RCIF 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 RCSTA 寄存器的 FERR 位获取帧错误状态。必须在读 RCREG 寄存器之后读 FERR 位。

帧错误（FERR=1）并不会阻止接收更多的字符。无需清零 FERR 位。

清零 RCSTA 寄存器的 SPEN 位会复位 USART，并强制清零 FERR 位。清零 RCSTA 寄存器的 CREN 位会强制清零 FERR 位。帧错误本身不会产生中断。

注：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 FERR 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，RCSTA 寄存器的 OERR 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 RCSTA 寄存器的 CREN 位或通过清零 RCSTA 寄存器的 SPEN 位使 USART 复位来清除错误。

13.1.2.6 接收 9 位字符

USART 支持 9 位数据接收。将 RCSTA 寄存器的 RX9EN 位置 1 时，USART 将接收到的每个字符的 9 位移入 RSR。必须在读 RCREG 中的低 8 位之后，读取 RX9D 数据位。

13.1.2.7 异步接收设置

1. 初始化 SPBRG 寄存器，以获得所需的波特率。
   （请参见“USART 波特率发生器（BRG）”章节）
2. 将 SPEN 位置 1，使能串行端口。必须清零 SYNC 位以执行异步操作。
3. 如果需要中断，将 PIE1 寄存器中的 RCIE 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 RX9EN 位置 1。
5. 将 CREN 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 RCIF 中断标志位置 1。如果 RCIE 中断允许位也置 1 还将产生中断。
7. 读 RCREG 寄存器，从接收缓冲器获取接收到的 8 个低数据位。
8. 读 RCSTA 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 CREN 接收器使能位清零 OERR 标志。

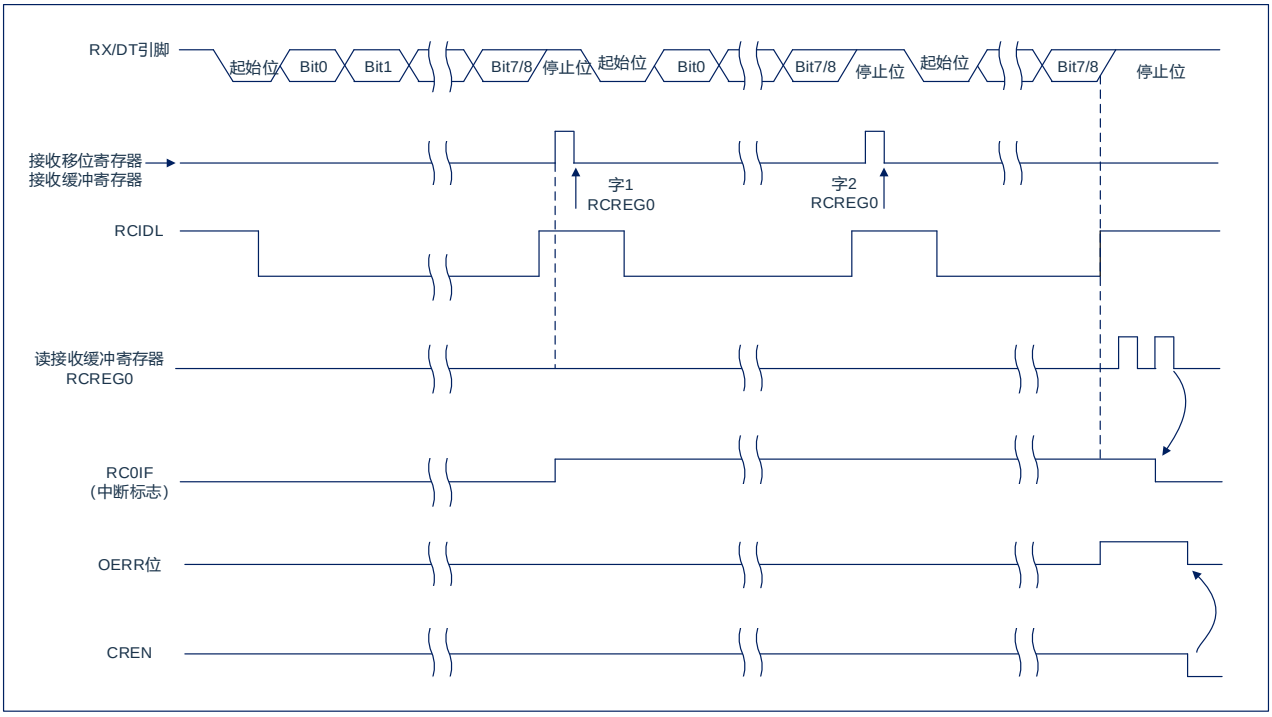


图 13-5：异步接收

注：本时序图显示出了在 RX 输入引脚接收三个字的情况，在第 3 个字后读取 RCREG（接收缓冲器），导致 OERR（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 USART 相关寄存器

TXSTA：发送状态和控制寄存器(98H)

98H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TXSTA	CSRC	TX9EN	TXEN(1)	SYNC	SCKP	---	TRMT	TX9D
R/W	R/W	R/W	R/W	R/W	R/W	---	R	R/W
复位值	0	0	0	0	0	0	1	0

Bit7 CSRC: 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 TX9: 9位发送使能位
1= 选择9位发送
0= 选择8位发送
Bit5 TXEN: 发送使能位(1)
1= 使能发送
0= 禁止发送
Bit4 SYNC: USART模式选择位
1= 同步模式
0= 异步模式
Bit3 SCKP: 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TX/CK引脚
0= 直接将数据字符发送到TX/CK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 保留，需为0
Bit1 TRMT: 发送移位寄存器状态位
1= TSR为空
0= TSR为满
Bit0 TX9D: 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，SREN/CREN 会覆盖 TXEN 的值。

RCSTA：接收状态和控制寄存器(97H)

97H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RCSTA	SPEN	RX9EN	SREN	CREN	RCIDL	FERR	OERR	RX9D
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 SPEN: 串行端口使能位
1= 使能串行端口（将RX/DT和TX/CK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 RX9: 9位接收使能位
1= 选择9位接收
0= 选择8位接收

Bit5 SREN: 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 CREN: 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CREN使能位（CREN覆盖SREN）
0=禁止连续接收

Bit3 RCIDL: 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 FERR: 帧错误位
1= 帧错误（可通过读RCREG寄存器更新并接收下一个有效字节）
0= 没有帧错误

Bit1 OERR: 溢出错误位
1= 溢出错误（可通过清零CREN位清零）
0= 没有溢出错误

Bit0 RX9D: 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

13.4 USART 波特率发生器（BRG）

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

SPBRG 寄存器对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 为一个计算波特率和波特率误差的示例。

表 13-2 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 SPBRG 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 RCIDL 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 8MHz，目标波特率为 9600bps，异步模式采用 8 位 BRG 的器件：

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

求解 SPBRG：

$$X=\frac{\frac{F_{HSI}}{目标波特率}}{16}-1=\frac{\frac{8000000}{9600}}{16}-1=[51.08]=51$$

$$计算波特率=\frac{8000000}{16（51+1）}=9615$$

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

表 13-1：波特率公式

配置位	BRG/USART 模式	波特率公式
SYNC		
0	8 位/异步	$F_{HSI}/[16(n+1)]$
1	8 位/同步	$F_{HSI}/[4(n+1)]$

说明：n= SPBRG 寄存器的值。

表 13-2：异步模式下的波特率

目标波特率	SYNC=0		
	F_{HSI}=24.00MHz		
	实际波特率	误差（%）	SPBRG 值
9600	9615	0.16	155
10417	10417	0	143
14400	14286	-0.8	105
19200	19230	0.16	78
115200	115384	0.16	12

13.5 USART 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。USART 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 USART 配置为同步主控操作：

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

将 TXSTA 寄存器的 SYNC 位置 1，可将 USART 配置用于同步操作。将 TXSTA 寄存器的 CSRC 位置 1，将器件配置为主控器件。将 RCSTA 寄存器的 SREN 和 CREN 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 RCSTA 寄存器的 SPEN 位置 1，使能 USART。

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 TX/CK 引脚发送时钟信号。当 USART 被配置为同步发送或接收操作时，TX/CK 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 TXSTA 寄存器的 SCKP 位选择时钟极性。将 SCKP 位置 1 将时钟空闲状态设置为高电平。当 SCKP 位置 1 时，数据在每个时钟的下降沿发生改变。清零 SCKP 位，将时钟空闲状态设置为低电平。当清零 SCKP 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 RX/DT 引脚输出数据。当 USART 配置为同步主控发送操作时，器件的 RX/DT 和 TX/CK 输出引脚自动使能。

向 TXREG 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 TXREG 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 TXREG 中的数据会被立即传输到 TSR 寄存器。当字符从 TXREG 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 SPBRG 寄存器，以获得所需的波特率。
   （请参见“USART 波特率发生器（BRG）”章节）
2. 将 SYNC、SPEN 和 CSRC 位置 1，使能同步主控串行端口。
3. 将 SREN 和 CREN 位清零，禁止接收模式。
4. 将 TXEN 位置 1 使能发送模式。
5. 如果需要发送 9 位字符，将 TX9EN 置 1。
6. 若需要中断，将 PIE2 寄存器中的 TXIE 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 TX9D 位。
8. 通过将数据装入 TXREG 寄存器启动发送。

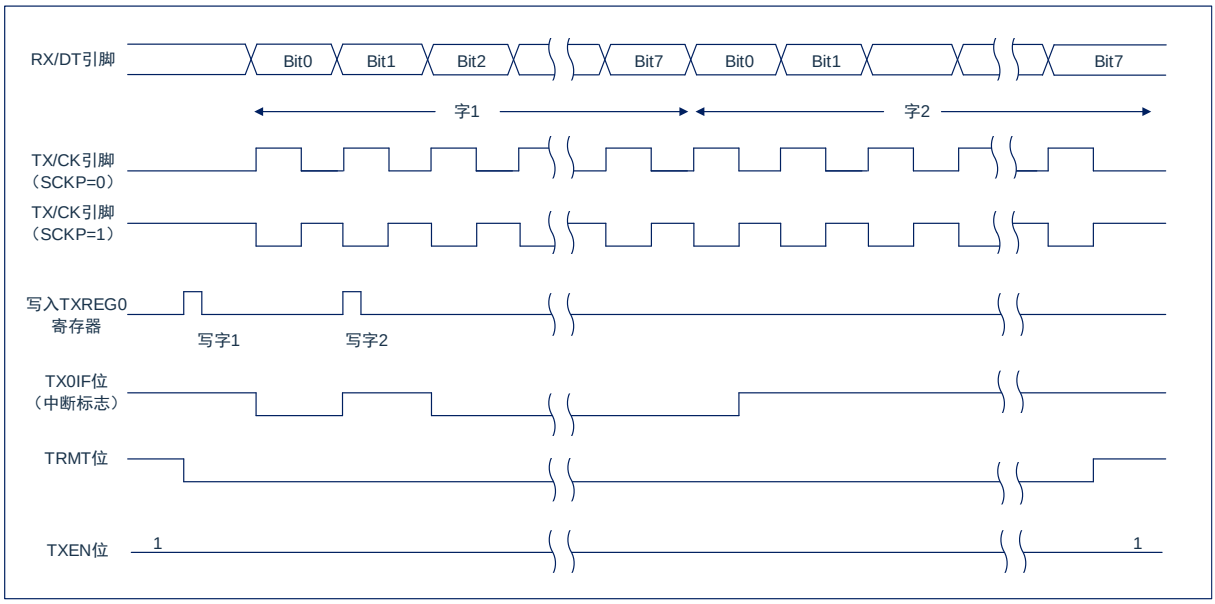


图 13-6：同步发送

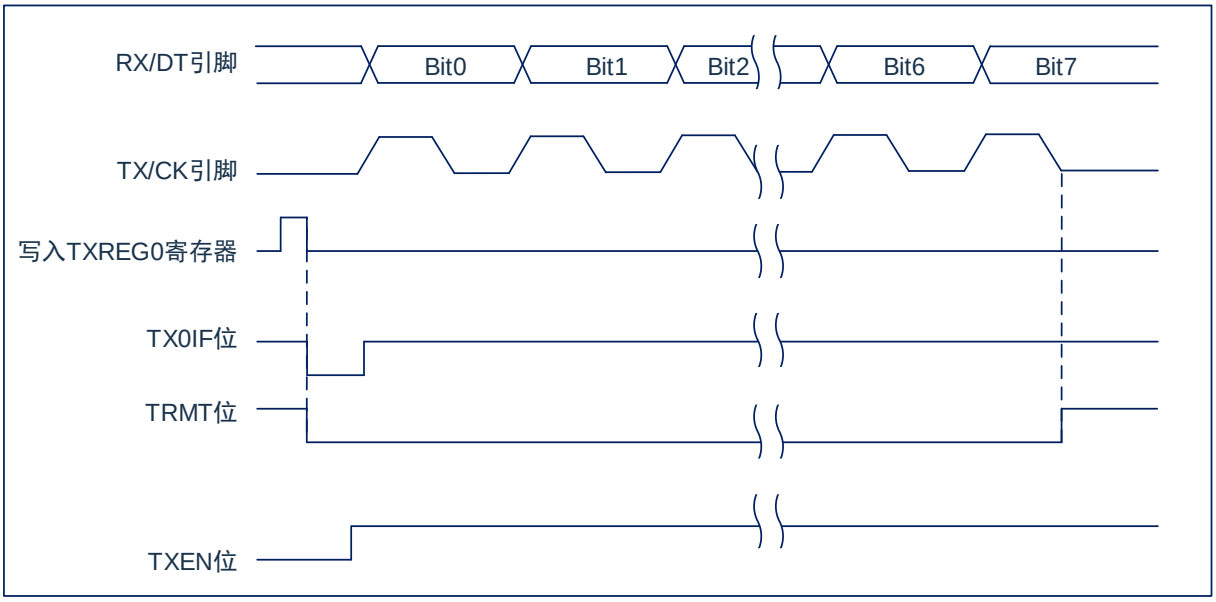


图 13-7：同步发送（通过 TXEN）

13.5.1.5 同步主控接收

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

在同步模式下，将单字接收使能位（RCSTA 寄存器的 SREN 位）或连续接收使能位（RCSTA 寄存器的 CREN 位）置 1 使能接收。当将 SREN 置 1，CREN 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 SREN 位。当 CREN 置 1 时，将产生连续时钟，直到清零 CREN 为止。如果 CREN 在一个字符的传输过程中清零，则 CK 时钟立即停止，并丢弃该不完整的字符。如果 SREN 和 CREN 都置 1，则当第一个字符传输完成时，SREN 位被清零，CREN 优先。

将 SREN 或 CREN 位置 1，启动接收。在 TX/CK 时钟引脚信号的下降沿采样 RX/DT 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 RCIF 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 RCREG 读取。只要接收 FIFO 中仍有未读字符，则 RCIF 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 TX/CK 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，TX/CK 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 RCREG 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，RCSTA 寄存器的 OERR 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 OERR 位清零。如果发生溢出时，SREN 位为置 1 状态，CREN 位为清零状态，则通过读 RCREG 寄存器清除错误。如果溢出时，CREN 为置 1 状态，则可以清零 RCSTA 寄存器的 CREN 位或清零 SPEN 位以复位 USART，从而清除错误。

13.5.1.8 接收 9 位字符

USART 支持接收 9 位字符。当 RCSTA 寄存器的 RX9EN 位置 1 时，USART 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 RCREG 的 8 个低位之后，读取 RX9D 数据位。

13.5.1.9 同步主控接收设置

1. 初始化 SPBRG 寄存器，以获得所需的波特率。（注：必须满足 SPBRG>05H）
2. 将 SYNC、SPEN 和 CSRC 位置 1 使能同步主控串行端口。
3. 确保将 CREN 和 SREN 位清零。
4. 如果使用中断，将 INTCON 寄存器的 GIE 和 PEIE 位置 1，并将 PIE2 寄存器的 RCIE 位也置 1。
5. 如果需要接收 9 位字符，将 RX9EN 位置 1。
6. 将 SREN 位置 1，启动接收，或将 CREN 位置 1 使能连续接收。
7. 当字符接收完毕后，将 RCIF 中断标志位置 1。如果允许位 RCIE 置 1，还会产生一个中断。
8. 读 RCREG 寄存器获取接收到的 8 位数据。
9. 读 RCSTA 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 RCSTA 寄存器的 CREN 位或清零 SPEN 以复位 USART 来清除错误。

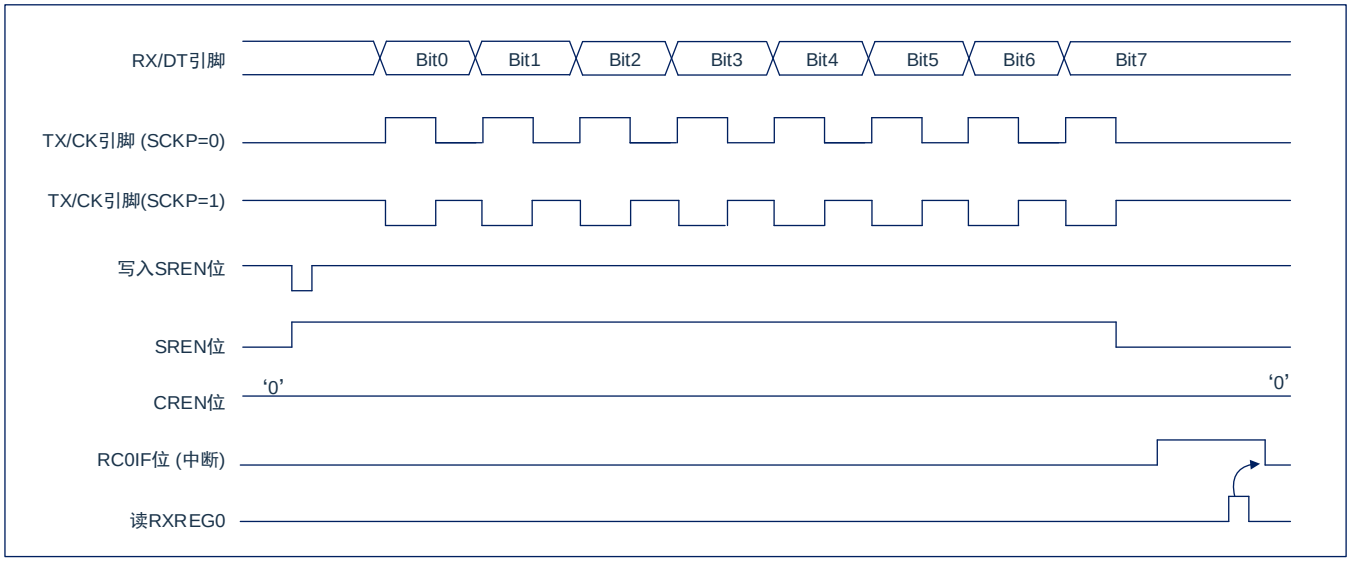


图 13-8：同步接收（主控模式，SREN）

注：时序图说明了 SREN=1 时的同步主控模式。

13.5.2 同步从动模式

下列位用来将 USART 配置为同步从动操作：

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

将 TXSTA 寄存器的 SYNC 位置 1，可将器件配置用于同步操作。将 TXSTA 寄存器的 CSRC 位置 0，将器件配置为从动器件。将 RCSTA 寄存器的 SREN 和 CREN 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 RCSTA 寄存器的 SPEN 位置 1，使能 USART。

13.5.2.1 USART 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 SYNC 和 SPEN 位置 1 并将 CSRC 位清零。
2. 将 CREN 和 SREN 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 寄存器的 TXIE 位也置 1。
4. 如果需要发送 9 位数据，将 TX9EN 位置 1。
5. 将 TXEN 位置 1 使能发送。
6. 若选择发送 9 位数据，将最高位写入 TX9D 位。
7. 将低 8 位数据写入 TXREG 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. CREN 位总是置 1，因此接收器不能进入空闲状态。
2. SREN 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 SYNC 和 SPEN 位置 1 并将 CSRC 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 寄存器的 RCIE 位也置 1。
3. 如果需要接收 9 位字符，将 RX9EN 位置 1。
4. 将 CREN 位置 1，使能接收。
5. 当接收完成后，将 RCIF 位置 1。如果 RCIE 已置 1，还会产生一个中断。
6. 读 RCREG 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 RCSTA 寄存器的 RX9D 位获取最高位。
8. 如果产生溢出错误，清零 RCSTA 寄存器的 CREN 位或清零 SPEN 位以复位 USART 来清除错误。

14. 蜂鸣器输出模块（BUZZER）

14.1 BUZZER 概述

芯片内置一个可编程 8 位周期的 BUZZER 模块，4 种分频选择可供选择，产生 50%的占空比方波，其频率覆盖一个较宽的范围；由 BUZCON1 寄存器中 BUZEN 控制。

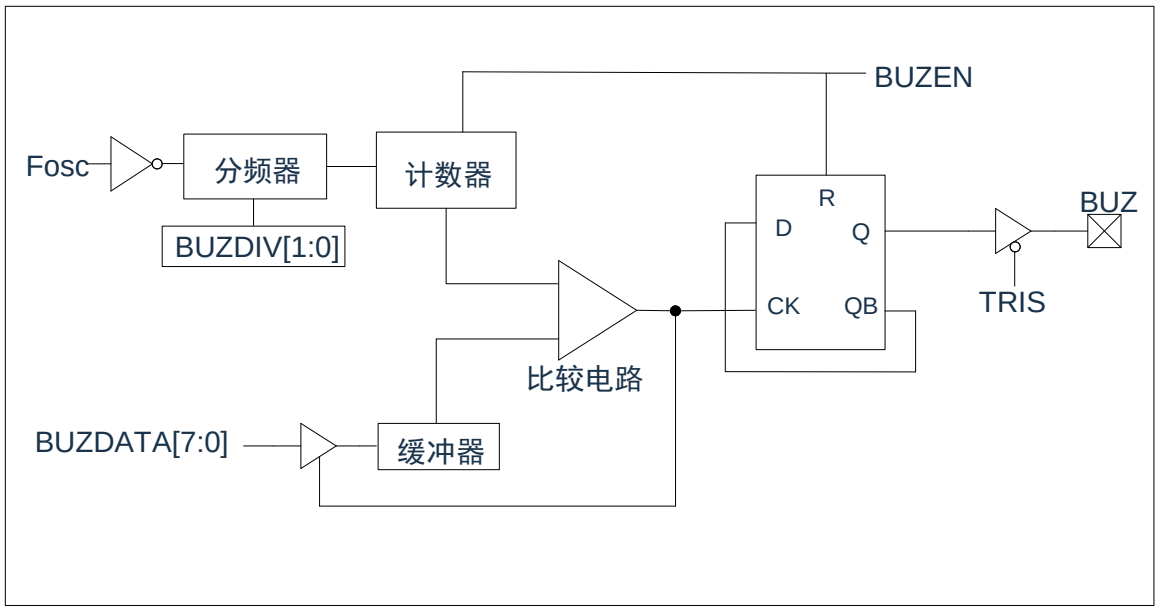


图14-1：BUZZER结构框图

注：当BUZDATA<7:0>=8’b00000000时，BUZ禁止使能。

14.2 相关寄存器

BUZ 控制寄存器 BUZCON0 (194H)

194H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BUZCON0	BUZDATA<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 BUZDATA<7:0>: BUZZER输出周期数据

BUZZER 控制寄存器 BUZCON1 (195H)

195H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BUZCON1	BUZEN	----	----	----	----	----	BUZDIV<1:0>	
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 BUZEN BUZ 使能
0= 禁止
1= 使能

Bit6~Bit2 未用。

Bit1~Bit0 BUZDIV<1:0>: 蜂鸣器时钟分频
00= Fosc/4
01= Fosc/8
10= Fosc/16
11= Fosc/32

14.3 BUZZER 周期

BUZZER 周期是通过写 BUZCON 寄存器的 BUZDATA<7:0>来指定的。

BUZZER 周期计算公式：

BUZZER 周期=(BUZDATA<7:0>+1)*Tosc*(CLKDIV 分频值)*2 **注：$T_{osc}=1/F_{osc}$**

当 BUZZER 周期计数器等于 BUZDATA<7:0>时，BUZ 引脚翻转。

14.4 BUZ 设置

使用 BUZZER 模块时应该执行以下步骤：

1. 通过将相应的 TRIS 位置 1，使之成为输入引脚。
2. 通过装载 BUZCON 寄存器设置 BUZZER 周期和分频。
3. 设置 BUZCON1 寄存器，使能 BUZZER 输出。
4. 通过将相应的 TRIS 位清零，使能 BUZ 引脚输出驱动器。

15. PWM 模块

芯片包含一个 10 位 PWM 模块，可配置为 2 路独立周期、独立占空比的输出。

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(112H)

112H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON0	CLKDIV<2:0>			---	---	---	PWM1EN	PWM0EN
R/W	R/W	R/W	R/W	---	---	---	R/W	R/W
复位值	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~Bit2 未用
Bit1~Bit0 PWMxEN: PWMx使能位。
1= 使能PWMx
0= 禁止PWMx

PWM 控制寄存器 PWMCON1(113H)

113H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON1	---	PWMSEL	---	---	---	---	PWM1DIR	PWM0DIR
R/W	---	R/W	---	---	---	---	R/W	R/W
复位值	---	0	---	---	---	---	0	0

Bit7 未用
Bit6 PWMSEL: PWM0 输出IO选择
1= RC3输出PWM0
0= RC0输出PWM0
Bit5~Bit2 未用
Bit1~Bit0 PWMxDIR: PWM输出取反控制位
1= PWMx取反输出
0= PWMx正常输出

PWM0 周期低位寄存器 PWM0TL(115H)

115H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM0TL	PWM0T<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 PWM0T<7:0>: PWM0周期低8位

PWM1 周期低位寄存器 PWM1TL(116H)

116H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM1TL	PWM1T<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 PWM1T<7:0>: PWM1周期低8位

周期高位寄存器 PWMTH (114H)

114H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTH	---	---	PWM1T<9:8>		---	---	PWM0T<9:8>	
R/W	---	---	R/W	R/W	---	---	R/W	R/W
复位值	---	---	0	0	---	---	0	0

Bit7~Bit6 未用
Bit5~Bit4 PWM1T<9:8>: PWM1周期高2位
Bit3~Bit2 未用
Bit1~Bit0 PWM0T<9:8>: PWM0周期高2位

PWM0 占空比低位寄存器 PWMD0L (117H)

117H	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 (118H)

118H	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

PWM0 和 PWM1 占空比高位寄存器 PWMD01H (119H)

119H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD01H	---	---	PWMD1<9:8>		---	---	PWMD0<9:8>	
R/W	---	---	R/W	R/W	---	----	R/W	R/W
复位值	---	---	0	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。

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 周期中断标志位；

注：只有 PWM0 能够产生 PWM 周期中断标志位。

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：

$$\text{脉冲宽度} = (PWMDx<9:0>+1)*T_{OSC}*(CLKDIV\ \text{分频值})$$

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 频率只与芯片振荡时钟有关，系统时钟频率发生任何改变都不会影响 PWM 频率。

15.7 PWM 设置

使用 PWM 模块时应该执行以下步骤：

1. 通过将相应的 TRIS 位置 1，使之成为输入引脚；
2. 通过装载 PWMTH，PWMxTL 寄存器设置 PWM 周期；
3. 通过装载 PWMD01H，PWMDxL 寄存器设置 PWM 占空比；
4. 若需要 PWM 取反输出，则需将 PWMCON1<1:0>的相应位置 1
5. 清零 PWMIF 标志位；
6. 设置 PWMCN0<1:0>的相应位以使能相应 PWM 输出；
7. 通过将相应的 TRIS 位清零，使能 PWM 引脚输出驱动器。

16. 程序 EEPROM 和程序存储器控制

16.1 概述

该系列中器件具有 8K 字的程序存储器，地址范围从 0000h 到 1FFFh，在 0000h 到 03FFh 地址范围内都是只读的，在 0400h 到 1FFFh 地址范围内都是可读写的；器件具有 128 字节的程序 EEPROM，地址范围为 0h 到 07Fh，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文件空间，而是通过特殊功能寄存器（SFR）对其进行间接寻址。共有 7 个 SFR 寄存器用于访问这些存储器：

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

当访问程序 EEPROM 时，EEDAT 寄存器存放 8 位读写的数据，而 EEADR 寄存器存放被访问的程序 EEPROM 单元的地址。

当访问器件的程序存储器时，EEDAT 和 EEDATH 寄存器形成一个双字节字用于保存要读写的 16 位数据，EEADR 和 EEADRH 寄存器组成一个双字节字用于保存待读写的 13 位程序存储器地址。

程序存储器允许以字为单位读写。程序 EEPROM 允许字节读写。字写或者字节写操作可自动擦除目标单元并写入新数据（在写入前擦除）。

写入时间由片上定时器控制。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，用于进行字节或字操作。

当器件受代码保护时，CPU 仍可继续读写程序 EEPROM 和程序存储器。代码保护时，器件编程器将不再能访问程序 EEPROM 或程序存储器。

注：
1. 程序存储器指 ROM 空间，即指令代码存储的空间，可读写；
   程序 EEPROM 是可存储用户数据的空间，可读写。
2. 程序 EEPROM 正常写电压范围为 2.5V~4.5V，写电流为 10mA@VDD=3.3V。

16.2 相关寄存器

16.2.1 EEADR 和 EEADRH 寄存器

EEADR 和 EEADRH 寄存器能寻址最大 128 字节的程序 EEPROM 或最大 8K 字的程序存储器。

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。当选择程序 EEPROM 地址值时，只将地址的低字节写入 EEADR 寄存器。

16.2.2 EECON1,EECON2 和 EECON3 寄存器

EECON1 是访问程序 EEPROM 的控制寄存器。

控制位 EEPGD 决定访问的是程序存储器还是程序 EEPROM。该位被清零时，和复位时一样，任何后续操作都将针对程序 EEPROM 进行。该位置 1 时，任何后续操作都将针对程序存储器进行。

控制位 RD 和 WR 分别启动读和写。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。由于无法用软件将 WR 位清零，从而可避免意外地过早终止写操作。

- 当 WREN 置 1 时，允许对程序存储器或者程序 EEPROM 执行写操作。上电时，WREN 位被清零。当正常的写入操作被 LVR 复位时，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。
- 当写操作完成时 PIR1 寄存器中的中断标志位 EEIF 被置 1。此标志位必须用软件清零。

EECON2 不是物理寄存器。读 EECON2 得到的是全 0。

EECON2 寄存器在执行程序存储器和程序 EEPROM 写序列时使用。

EECON3 寄存器仅在执行程序存储器写序列时使用。

EEPROM 数据寄存器 EEDAT(11AH)

11AH	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>: 要向程序存储器或者程序EEPROM写入数据的低8位，或者要从程序存储器中或者程序EEPROM中读取数据的低8位

EEPROM 地址寄存器 EEADR(11BH)

11BH	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>: 指定程序存储器或者程序EEPROM读/写操作的地址的低8位

EEPROM 数据寄存器 EEDATH(11CH)

11CH	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>:　从程序EEPROM/程序存储器读出的数据的高8位

EEPROM 地址寄存器 EEADRH(11DH)

11DH	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/写操作的高5位地址

EEPROM 控制寄存器 EECON1(11EH)

11EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON1	EEPGD	---	EETIME1	EETIME0	WRERR	WREN	WR	RD
R/W	R/W	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	---	0	0	X	0	0	0

Bit7　　EEPGD:　程序存储器/程序EEPROM选择位
1=　操作程序存储器
0=　操作程序EEPROM
Bit6　　未用
Bit5~Bit4　　EETIME<1:0>　最长烧写等待时间；**（更多 EETIME 信息请参考图 16-1）**
00=　2.5ms（VDD=4.0~4.5V，TEMP=0~75℃建议值）
01=　2.5ms（VDD=4.0~4.5V，TEMP=0~75℃建议值）
10=　5ms
11=　10ms（除 2.5ms 外的其他条件建议值）
Bit3　　WRERR:　EEPROM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，或EETIME设定的时间到了但自校验还未成功）
0=　写操作完成
Bit2　　WREN:　EEPROM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
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

16.3 读程序 EEPROM

要读取程序 EEPROM 单元，用户必须将地址写入 EEADR 寄存器，清零 EECON1 寄存器的 EEPGD 控制位，然后将控制位 RD 置 1。一旦设置好读控制位，程序 EEPROM 控制器将使用第二个指令周期来读数据。这会导致紧随”SETB　EECON1,RD”指令的第二条指令被忽略(1)。在紧接下来的一个时钟周期，程序 EEPROM 相应地址的值会被锁存到 EEDAT 寄存器中，用户可在随后的指令中读取这两个寄存器。EEDAT 将保存此值直至下一次用户向该单元读取或写入数据时为止。

注：程序存储器读操作后的两条指令必须为 NOP。这可阻止用户在 RD 位置 1 后的下一条指令执行双周期指令。

例：读程序 EEPROM

```
EEPDATA_READ:
        LD          A,RADDR         ;将要读取的地址放入 EEADR 寄存器
        LD          EEADR,A
        CLRB        EECON1,EEPGD    ;访问数据存储器
        SETB        EECON1,RD       ;启动读操作
        NOP
        NOP
        LD          A,EEDAT         ;读取数据到 ACC
        LD          RDATA,A
EEPDATA_READ_BACK:
        RET
```

16.4 写程序 EEPROM

要写程序 EEPROM 存储单元，用户应首先将该单元的地址写入 EEADR 寄存器并将数据写入 EEDAT 寄存器。然后用户必须按特定顺序开始写入每个字节。

如果没有完全按照下面的指令顺序（即首先将 55h 写入 EECON2，随后将 AAh 写入 EECON2，最后将 WR 位置 1）写每个字节，将不会启动写操作。在该代码段中应禁止中断。

此外，必须将 EECON1 中的 WREN 位置 1 以使能写操作。这种机制可防止由于代码执行错误（异常）（即程序跑飞）导致误写 EEPROM。在不更新 EEPROM 时，用户应该始终保持 WREN 位清零。WREN 位不能被硬件清零。

一个写过程启动后，将 WREN 位清零将不会影响此写周期。除非 WREN 位置 1，否则 WR 位将无法置 1。写周期完成时，WR 位由硬件清零并且 EE 写完成中断标志位（EEIF）置 1。用户可以允许此中断或查询此位。EEIF 必须用软件清零。

注：在写程序 EEPROM 期间，CPU 会停止工作，需在写操作开始前执行 CLRWDT 命令，以避免在此期间 WDT 溢出复位芯片。

例：写程序 EEPROM

```
EEPDATA_WRITE:
        LD      A,WADDR         ;将要写入的地址放入 EEADR 寄存器
        LD      EEADR,A
        LD      A,WDATA         ;将要写入的数据给 EEDAT 寄存器
        LD      EEDAT,A
        CLRWDT
        CLR     EECON1
        SETB    EECON1,EETIME0
        SETB    EECON1,EETIME1  ;EE 烧写时间 10ms，用户可自定义
        CLRB    EECON1,EEPGD    ;访问数据存储器
        SETB    EECON1,WREN     ;允许写周期
        CLRB    F_GIE_ON        ;保存中断开启状态
        SZB     INTCON,GIE
        SETB    F_GIE_ON
        CLRB    INTCON,GIE      ;关闭中断
        SZB     INTCON,GIE      ;确保中断已关闭
        JP      $-2

        LDIA    055H
        LD      EECON2,A
        LDIA    0AAH
        LD      EECON2,A
        SETB    EECON1,WR       ;启动写操作
        NOP
        NOP
        CLRWDT
        CLRB    EECON1,WREN     ;写结束，关闭写使能位

        SZB     F_GIE_ON        ;恢复中断开启状态
        SETB    INTCON,GIE

        SNZB    EECON1,WRERR    ;判断 EEPROM 写操作是否出错
        JP      EEPDATA_WRITE_BACK
        SZDECR  WERR_C          ;计数超时则退出，用户可自定义
        JP      EEPDATA_WRITE   ;EEPROM 写操作出错则重写
EEPDATA_WRITE_BACK:
        RET
```

16.5 读程序存储器

要读取程序存储器单元,用户必须将地址的高位和低位分别写入 EEADR 和 EEADRH 寄存器,将 EECON1 寄存器的 EEPGD 位置 1，然后将控制位 RD 置 1。一旦设置好读控制位，程序存储器控制器将使用第二个指令周期来读数据。这会导致紧随“SETB EECON1,RD”指令的第二条指令被忽略。在紧接下来的一个时钟周期，程序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。EEDAT 和 EEDATH 寄存器将保存此值直至下一次用户向该单元读取或写入数据时为止。

注：
1. 程序存储器读操作后的两条指令必须为 NOP。这可阻止用户在 RD 位置 1 后的下一条指令执行双周期指令。
2. 当 EEPGD=1 时如果 WR 位置 1，它会立即复位为 0，而不执行任何操作。

例：读闪存程序存储器

```
        LD      A,RADDRL        ;将要读取的地址放入 EEADR 寄存器
        LD      EEADR,A
        LD      A,RADDRH        ;将要读取的地址高位放入 EEADRH 寄存器
        LD      EEADRH,A
        SETB    EECON1,EEPGD    ;选择操作程序存储器
        SETB    EECON1,RD       ;允许读操作
        NOP
        NOP
        LD      A,EEDAT         ;保存读取的数据
        LD      RDATL,A
        LD      A,EEDATH
        LD      RDATH,A
```

16.6 写程序存储器

要写程序 EEPROM 存储单元，用户应首先完成写使能序列。

写使能序列需要按照下面的指令顺序（将 5Ah 写入 EECON3），才使能写程序存储器。

接着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。然后用户接着完成写启动序列，才能启动写过程。

写启动序列需要完全按照下面的指令顺序（即首先将 69h 写入 EECON2，随后将 96h 写入 EECON2，最后将 WR 位置 1），才启动写操作。在该代码段中应禁止中断。

此外，必须将 EECON1 中的 WREN 位置 1 以使能写操作。这种机制可防止由于代码执行错误（异常）（即程序跑飞）导致误写程序存储器。在不更新程序存储器时，用户应该始终保持 WREN 位清零。WREN 位不能被硬件清零。

一个写过程启动后，将 WREN 位清零将不会影响此写周期。除非 WREN 位置 1，否则 WR 位将无法置 1。写周期完成时，WR 位由硬件清零并且 EE 写完成中断标志位（EEIF）置 1。用户可以允许此中断或查询此位。EEIF 必须用软件清零。

注：

1. 在写程序存储器或者程序 EEPROM 期间，CPU 会停止工作，需在写操作开始前执行 CLRWDT 命令，以避免在此期间 WDT 溢出复位芯片。
2. 写程序存储器之前需要正确配置 ROM 自写分区保护位（CONFIG 的 URSECPR 的值）

例：写程序存储器

```
EEPROM_ENABLE:
        SETB    EECON1,7        ;访问程序存储器
        LDIA    05AH
        LD      EECON3,A
        RET
        …
EEPROM_WRITE:
        CALL    EEPROM_ENABLE
        LD      A,WADDR         ;将要写入的地址低 8 位放入 EEADR 寄存器
        LD      EEADR,A
        LD      A,WADDRH        ;将要写入的地址高 8 位放入 EEADRH 寄存器
        LD      EEADRH,A
        LD      A,WDATA         ;将要写入的数据低 8 位给 EEDAT 寄存器
        LD      EEDAT,A
        LD      A,WDATAH        ;将要写入的数据高 8 位给 EEDATH 寄存器
        LD      EEDATH,A
        CLRWDT
        CLR     EECON1
        SETB    EECON1,EETIME0
        SETB    EECON1,EETIME1  ;EE 烧写时间 10ms，用户可自定义
        SETB    EECON1,EEPGD    ;访问程序存储器
        SETB    EECON1,WREN     ;允许写周期
        CLRB    F_GIE_ON        ;保存中断开启状态
        SZB     INTCON,GIE
        SETB    F_GIE_ON
        CLRB    INTCON,GIE      ;关闭中断
        SZB     INTCON,GIE      ;确保中断已关闭
        JP      $-2

        LDIA    069H
        LD      EECON2,A
        LDIA    096H
        LD      EECON2,A
        SETB    EECON1,WR       ;启动写操作
        NOP
        NOP
        CLRWDT
        CLRB    EECON1,WREN     ;写结束，关闭写使能位

        SZB     F_GIE_ON        ;恢复中断开启状态
        SETB    INTCON,GIE

        SNZB    EECON1,WRERR    ;判断程序存储器写操作是否出错
        JP      EEPROM_WRITE_BACK
        SZDECR  WERR_C          ;计数超时则退出，用户可自定义
        JP      EEPROM_WRITE    ;程序存储器写操作出错则重写
EEPROM_WRITE_BACK:
        RET
```

16.7 程序 EEPROM 操作注意事项

16.7.1 关于程序 EEPROM 的烧写时间

程序 EEPROM 烧写时间不是固定的，烧写不同的数据需要的时间不一样，从 100us~10ms 不等，EECON1 寄存器的 EETIME 位决定程序 EEPROM 烧写的最长时间，程序 EEPROM 模块内置自校验功能，在烧写过程中，自校验成功或 EETIME 设定的时间已到，满足其中一个条件就会结束写操作。在烧写期间 CPU 停止工作，外设模块正常工作，程序需要做好相关处理。

16.7.2 关于程序 EEPROM 的烧写次数

程序 EEPROM 的烧写次数，与 EETIME 设置的烧写时间，还有电压、温度相关。

16.7.3 写校验

根据具体的应用，好的编程习惯一般要求将写入程序 EEPROM 的值对照期望值进行校验。

16.7.4 避免误写的保护

有些情况下，用户可能不希望向程序 EEPROM 写入数据。为防止误写 EEPROM，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 EEPROM 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

17. LVD 低电压检测

17.1 LVD 模块概述

CMS341x 系列单片机具有低电压检测功能，可以用于监测电源电压和外部输入电压，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。

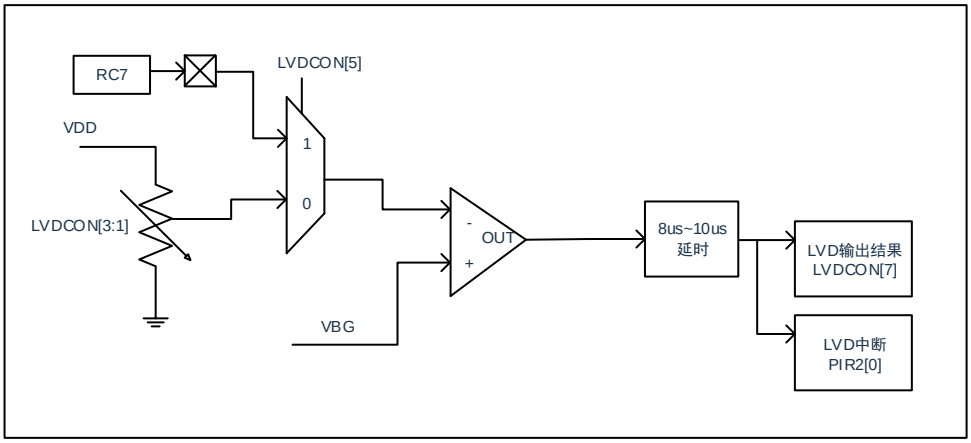


图 17-1：LVD 框图

17.2 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(1FH)

1FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LVDCON	LVD_RES	LVD_MODE<1:0>		---	LVD_SEL<2:0>			LVDEN
R/W	R	R/W	R/W	---	R/W	R/W	R/W	R/W
复位值	X	0	0	---	0	0	0	0

Bit7 LVD_RES: LVD 输出结果
LVD_MODE=00
0= VDD>设定的 LVD 电压
1= VDD<设定的 LVD 电压
LVD_MODE=01
0= LVD 引脚电压>V_{BG}
1= LVD 引脚电压<V_{BG}

Bit6~Bit5 LVD_MODE<1:0>: LVD 模式
00= 检测 VDD 电压
01= 检测 LVD 引脚电压
1x= 未用

Bit4 未用

Bit3~Bit1 LVD_SEL<2:0>: LVD 电压选择（LVD_MODE=00 模式下有效）
000= 2.2V
001= 2.4V
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

17.3 LVD 操作

LVD 可以实现 VDD 或外部输入的电压检测，通过寄存器位 LVD_MODE 选择 LVD 检测模式，通过寄存器位 LVD_SEL 选择电压检测量值。当 LVD_MODE=00 时为检测 VDD 模式 ，实际上是 VDD 通过内部电阻分压与内部电压基准 1.2V 作比较；当 LVD_MODE=01 时为检测 LVD 引脚功能，实际上是 LVD 引脚电压和内部电压基准 1.2V 作比较。

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 不可以用于中断唤醒模式。

18. SPI 模块

18.1 SPI 模块概述

SPI 模式允许同时同步发送和接收 8 位数据。支持 SPI 的主控 4 种模式和从动 2 种模式。另外，可选择 3 线模式或 4 线模式。

4 线模式下使用以下三个引脚来完成通信：

- 主控数据输入/从动数据输出（MISO）
- 主控数据输出/从动数据输入（MOSI）
- 串行时钟（SCK）
- 从动选择（SS）

3 线模式下使用以下三个引脚来完成通信：

- 串行数据输入/输出（SDIO）
- 串行时钟（SCK）
- 从动选择（SS）

注：以下描述中，SDI 是代表主控模式的 MISO 引脚和从动模式的 MOSI 引脚；SDO 是代表主控模式的 MOSI 引脚和从动模式的 MISO 引脚。

18.2 SPI 相关寄存器

SPI 控制寄存器 SPICON2(96H)

96H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPICON2	---	CKE	MODE	---	---	---	---	SPIBF
R/W	---	R/W	R/W	---	---	---	---	R
复位值	---	0	0	---	---	---	---	0

Bit7 保留 需写0

Bit 6 CKE: SPI时钟边沿选择位。（注：在从动模式下，CKE必须设置为0）

SPICKP= 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

SPICKP = 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~Bit1 未用。

Bit0 SPIBF: 缓冲器满状态位

1= 接收完成，SPIBUF满

0= 接收未完成，SPIBUF空

SPI 控制寄存器 SPICON(95H)

95H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPICON	SPIWCOL	SPIOV	SPIEN	SPICKP	SPIM3	SPIM2	SPIM1	SPIM0
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 SPIWCOL: 写冲突标志位
1= 在发送/接收数据过程中，试图对 SPIBUF 寄存器进行写操作
0= 未发生冲突

Bit6 SPIOV: 接收溢出指示位
1= SPIBUF仍保持前一数据时，又收到一个新的字节。出现溢出时，SPISR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SPIBUF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SPIBUF寄存器来启动（该位必须由软件清零）
0= 没有溢出

Bit5 SPIEN: SPI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 SPICKP: 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 SPIM<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
其他= 保留

18.3 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（SPICON<5:0>和 SPICON2<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）

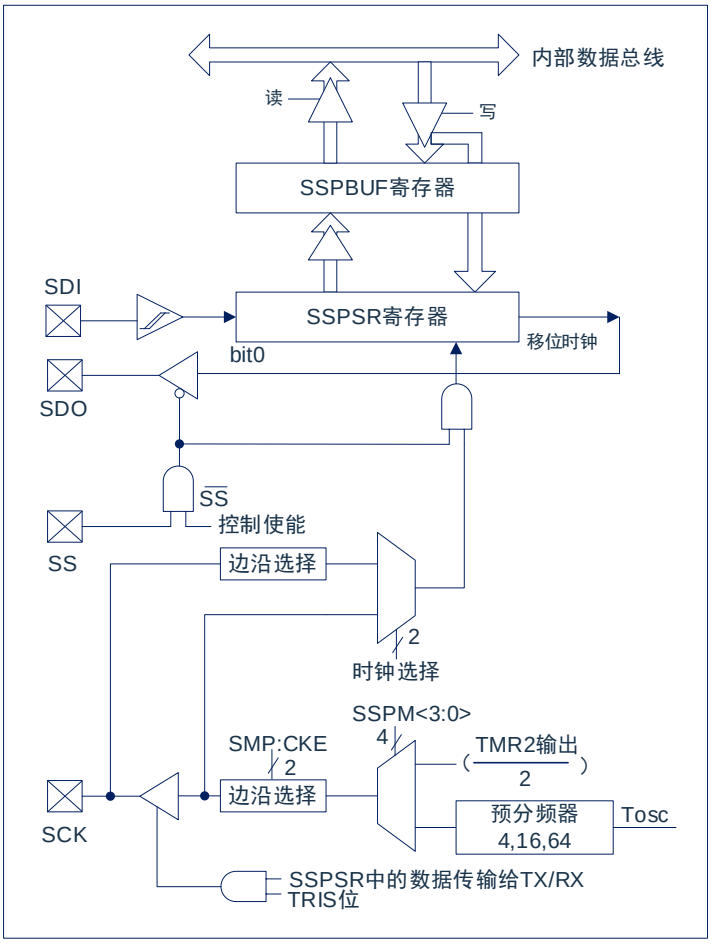


图 18-1：SPI 模块框图

注：I/O 引脚具有对 VDD 和 VSS 的二极管保护。

SPI 模块由一个发送/ 接收移位寄存器（SPISR）和一个缓冲寄存器（SPIBUF）组成。SPISR 将数据移入和移出器件，最高有效位在前。SPIBUF 保存上次写入 SPISR 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 SPIBUF 寄存器。然后，PIR1 寄存器的中断标志位 SPIIF 被置 1。这种双重缓冲数据接收方式（SPIBUF）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 SPIBUF 寄存器的操作都会被忽略，并将 SPICON 寄存器的写冲突检测位 SPIWCOL 置 1。此时用户必须用软件将 SPIWCOL 位清零，否则无法判别下一次对 SPIBUF 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 SPIBUF 之前，将 SPIBUF 中的前一个数据读出。

18.4 使能 SPI I/O

要使能串行端口，SPICON 寄存器的使能位 SPIEN 必须置 1。要复位或重新配置 SPI 模式，要先将 SPIEN 位清零，重新初始化 SPICON 寄存器，然后将 SPIEN 位置 1。这将把 MOSI、MISO、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 MOSI 的 TRIS 位清零(主控模式)；
- 必须将 MISO 的 TRIS 位清零(从动模式)；
- 必须将 SCK（主控模式）的 TRIS 位清零；
- 必须将 SCK（从动模式）的 TRIS 位置 1；
- 必须将 SS 的 TRIS 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。

18.5 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 SPIBUF 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）。SPISR 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 SPIBUF 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 SPICON 寄存器的 SPICKP 位进行相应的编程来选择时钟极性。图 18-2、图 18-3 和图 18-4 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}/8（或 2.TCY）
- F_{SYS}/16（或 4.TCY）
- F_{SYS}/64（或 16.TCY）
- TIMER2 输出/2

图 18-2 为主控模式的波形图。当 SPICON2 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 SPIBUF 的时间。

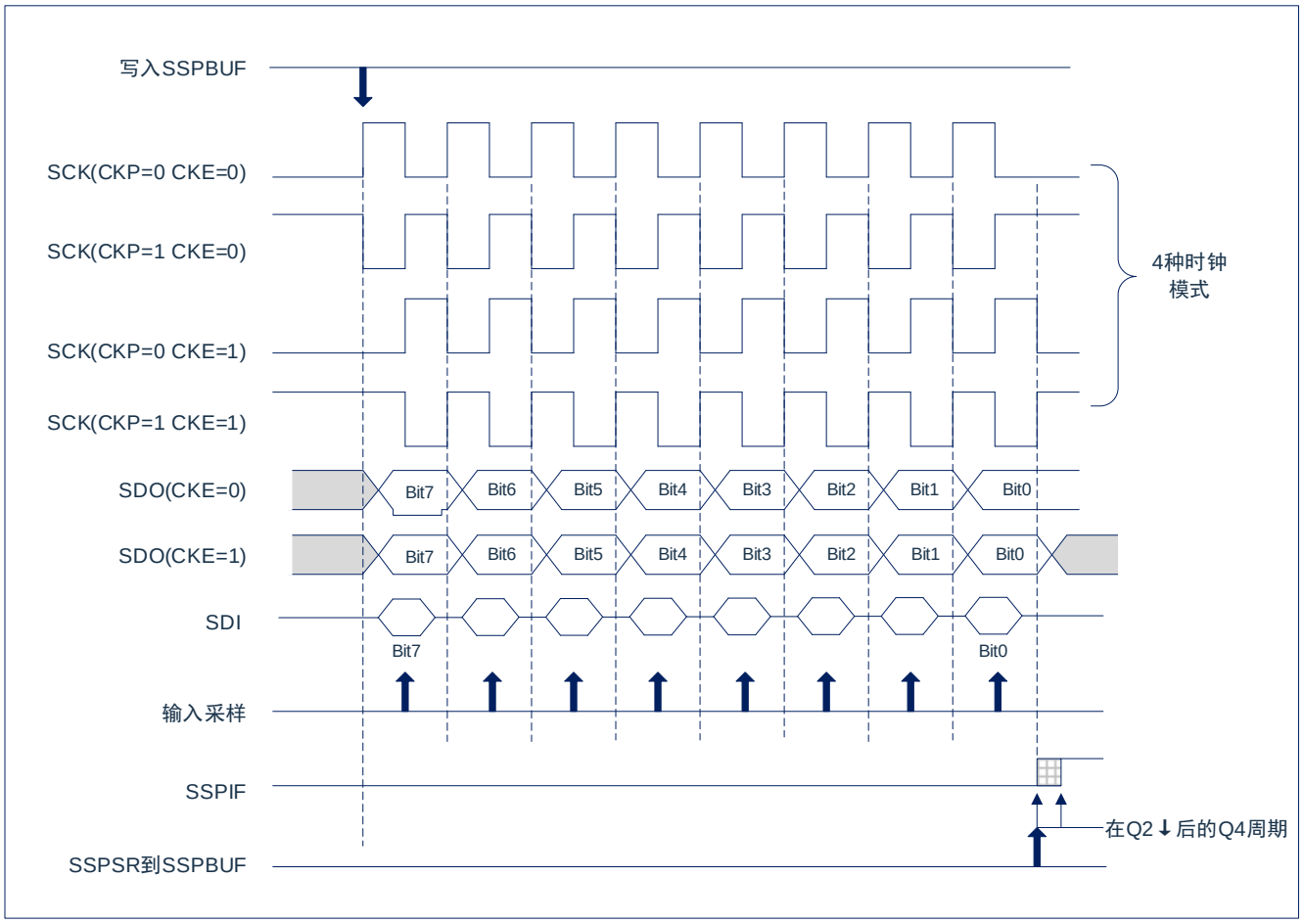


图 18-2：SPI 模式的波形（主控模式）

18.6 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 SPIIF 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。

18.7 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 SPICON<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 SPIEN 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（SPIxCON<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（SPICON2 寄存器）的从动模式下使用 SPI，则必须使能 SS 引脚控制。

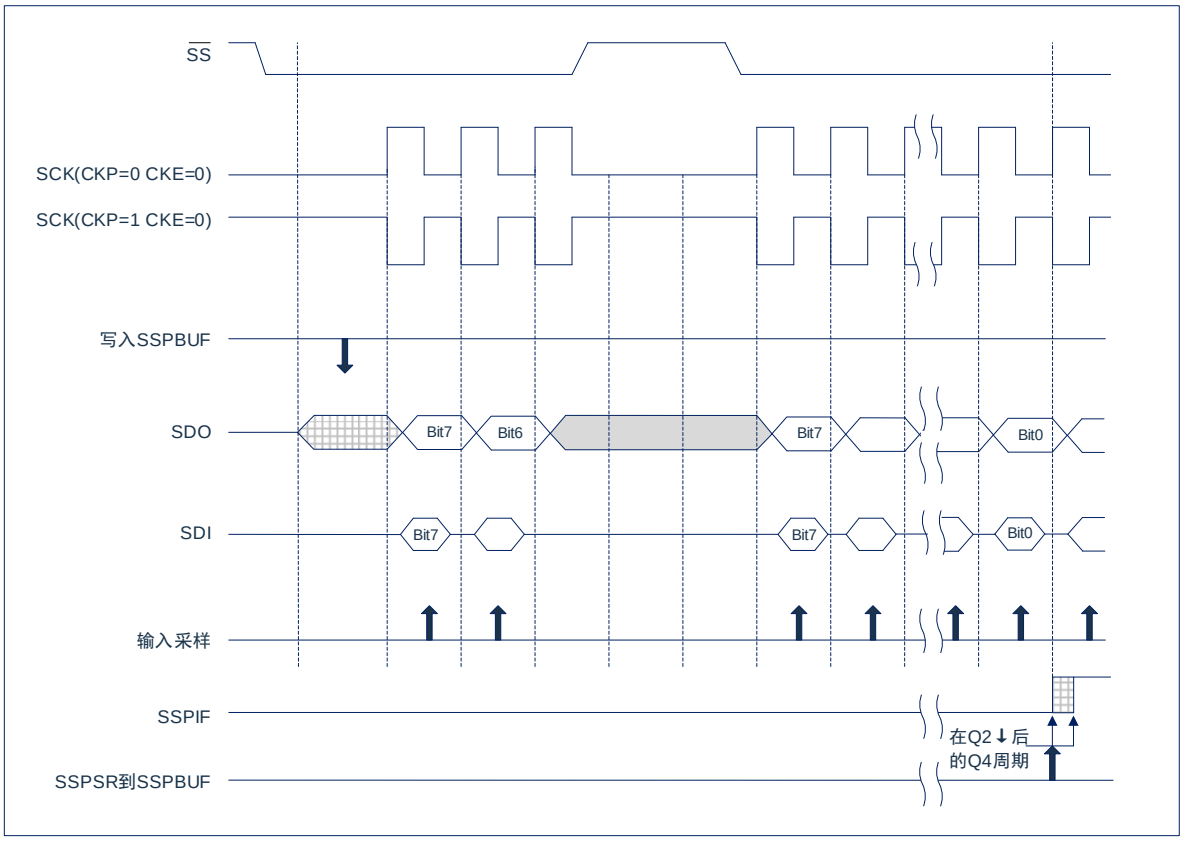


图 18-3：从动同步波形

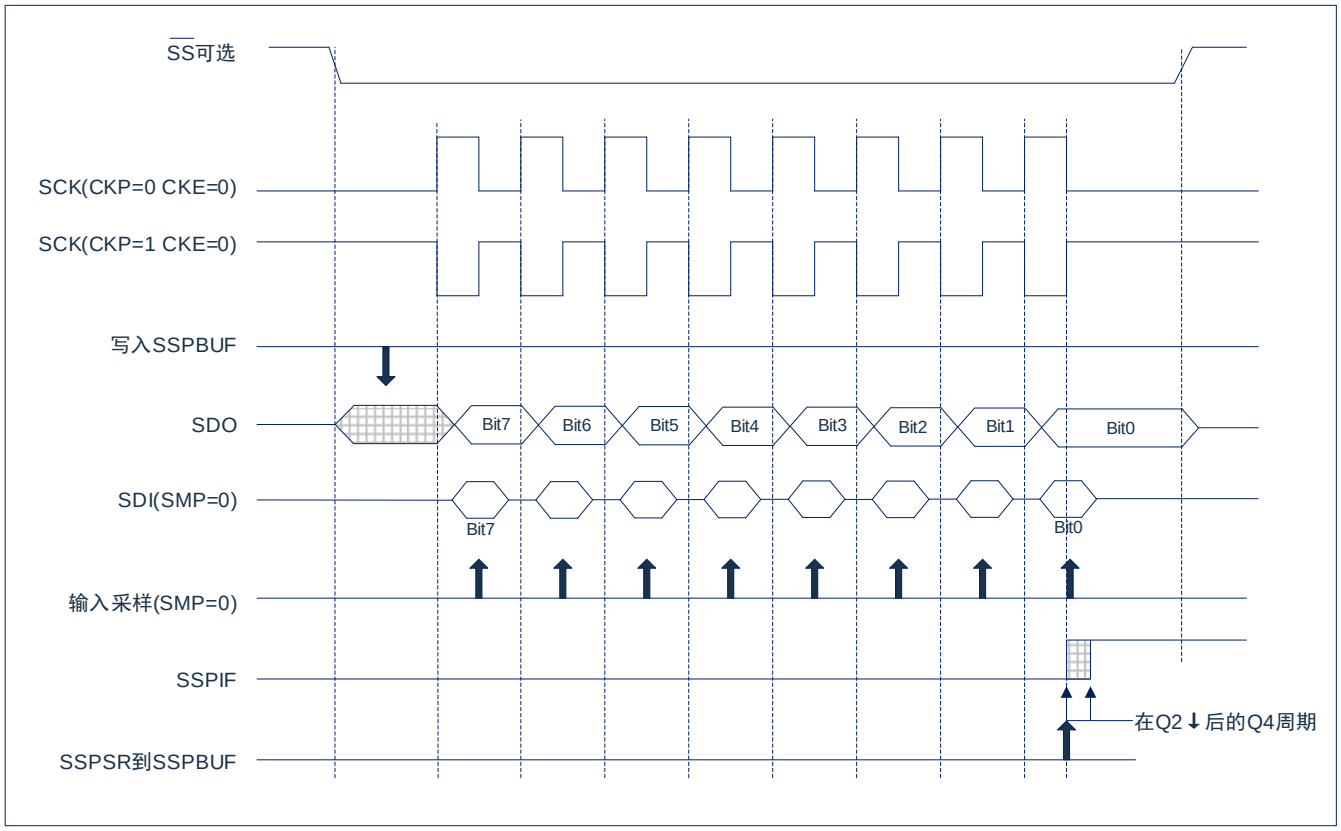


图 18-4：SPI 模式波形（从动模式，CKE=0）

18.8 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。

18.9 复位的影响

复位会禁止 SPI 模块并终止当前的传输。

19. MDU 模块

19.1 MDU 概述

MDU 模块支持 24 位无符号乘法和 48 位/24 位无符号除法器。该硬件乘除法器可取代软件乘除法运算，节省大量运算时间，程序存储器和数据存储器空间。

19.2 MDU 相关寄存器

MDU 数据寄存器 MDUDTn

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MDUDTn	MDUDTn<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 MDUDTn<7:0> MDU数据寄存器n

MDU 控制寄存器 MDUCON(20CH)

20CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
MDUCON	MDUOV	---	---	---	---	---	DIV48EN	MUL24EN
R/W	R	---	---	---	---		R/W	R/W
复位值	0	---	---	---	---	---	0	0

Bit7 MDUOV: 运算溢出标志位
0= 溢出未发生
1= 除数为0
Bit6~Bit2 未用
Bit1 DIV48EN: 除法使能标志，该标志只能写 1，运算完成后自动清零
0= 48-bit 除法未使能或运算完成
1= 48-bit 除法使能，使能期间写 MDUDTn 寄存器无效
Bit0 MUL24EN: 24-bit 乘法使能标志，该标志只能写 1，运算完成后自动清零
0= 24-bit 乘法未使能或运算完成
1= 24-bit 乘法使能，使能期间写 MDUDTn 寄存器无效

注：使能位只能 1 位有效，若多位使能，则优先级关系如下：MUL24EN>DIV48EN

19.3 24-bit 乘法操作

1）将被乘数写入 MDUDT5~MDUDT3;
2）将乘数写入 MDUDT2~MDUDT0;
3）使能 MUL24EN;
4）等待计算完成后，从 MDUDT5~MDUDT0 寄存器中读取乘积；

24-bit 乘法运算时间：$26*T_{MDUCK}$ (T_{MDUCK}=125ns)

19.4 48-bit 除法操作

1）将被除数写入 MDUDT8~MDUDT3;
2）将除数写入 MDUDT2~MDUDT0;
3）使能 DIV48EN;
4）等待计算完成后，从 MDUDT8~MDUDT3 寄存器中读取商，从 MDUDT2~MDUDT0 寄存器中读取余数；

48-bit 除法运算时间：$50*T_{MDUCK}$ (T_{MDUCK}=125ns)

20. LCD 驱动模块

CMS8H341x 可驱动 1/2Bias、1/3Bias 的 LCD，在设置完 LCD 数据并使能 LCD 控制位后，芯片可自动输出驱动 LCD。最大支持 23SEG×4COM 或则 21SEG×6COM。

20.1 LCD 功能管脚设置

若使能 RA0~RA7、RB0~RB7 以及 RC0~RC3、RC6 的 LCD 驱动功能，必须将其设置为输入态，即将相应的 TRIS 位置“1”。

20.2 LCD 功能 COM 口设置

COMSEL<2:0>	COM 口个数	Bias
000	2	1/2 or 1/3
001	2	1/2 or 1/3
010	3	1/2 or 1/3
011	4	1/2 or 1/3
10x	5	1/2 or 1/3
11x	6	1/2 or 1/3

设置 COM_EN 寄存器，将相应管脚设置为 LCD 功能的 COM 口。

20.3 LCD 功能 SEG 口设置

1. 设置相应管脚状态，如使用到 RA0~RA7、RB0~RB7 以及 RC0~RC3、RC6 的 LCD 功能设置，则管脚应设置为输入态；
2. 设置 SEGEN0、SEGEN1、SEGEN2 寄存器中相应管脚为 LCD 驱动功能。

20.4 LCD 功能的数据设置

设置 LCD 显示数据需以下步骤：

1. 设置 LCDCON1 寄存器的 SEGOUT<1:0>位为“1x”；
2. 设置 LCDADD 寄存器的第 7 位 LCDCS=1，允许读写数据；
3. 设置 LCDADD 的 0-6 位数据地址；
4. 设置 LCDDATA 数据（没有用作 LCD 功能用的管脚，其相应的 LCDDATA 位需设置为“0”）；
5. 重复步骤 3-4 设置其它地址数据；
6. 设置完成后关闭数据读写位 LCDCS=0。

LCD 地址和数据对应关系如下表：

LCDADD	LCDDATA								
00H	-	-	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG0
01H	-	-	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG1
.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.
15H	-	-	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG21
16H	-	-	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG22
			COM5	COM4	COM3	COM2	COM1	COM0	

20.5 LCD 相关寄存器

LCD 驱动功能相关寄存器有：控制寄存器 LCDCON0、LCDCON1；地址寄存器 LCDADD；数据寄存器 LCDDATA；口线设置寄存器 COMEN、SEGEN0、SEGEN1、SEGEN2。

LCD 控制寄存器 LCDCON0(216H)

216H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCON0	LCDEN	BIAS	COMSEL<2:0>			LCDCLK<2:0>		
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 LCDEN: LCD模块使能
0= 禁止LCD模块
1= 使能LCD模块
（注：需要把LCD相关COM、SEG、时钟等设置好后，才能使能该位）

Bit6 BIAS: LCD偏置电压选择
0= 1/2 Bias
1= 1/3 Bias

Bit5~Bit3 COMSEL<2:0>: LCD模块COM口个数选择
000= 2COM
001= 2COM
010= 3COM
011= 4COM
10x= 5COM
11x= 6COM

Bit2~Bit0 LCDCLK<2:0>: LCD扫描频率选择（根据LCDCON1第6位LCDF选择时钟源）

	LCDF=0	LCDF=1
000=	1.953KHz	1.985KHz
001=	1.953KHz	1.985KHz
010=	1.953KHz	1.985KHz
011=	1.953KHz	1.985KHz
100=	976.563Hz	992.75Hz
101=	488.281Hz	496.375Hz
110=	244.141Hz	248.188Hz
111=	122.07Hz	124.093Hz

注：LCD扫描频率=LCD帧频率 * COM口个数
建议LCD帧频率设置为60~120Hz

LCD 控制寄存器 LCDCON1(217H)

217H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCON1	CPREN	LCDF	SEGOUT<1:0>		FCCTLM<1:0>		VLCDX<1:0>	
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 CPREN: VLCD 升压泵CPR使能
0= CPR关闭，VLCD内部连接VDD，由VDD为LCD 驱动供电
1= CPR开启，VLCD连接CPR的输出，由CPR为LCD驱动供电
Bit6 LCDF: LCD时钟源选择
0= 内部时钟
1= 外部 32.768kHz振荡时钟
Bit5~Bit4 SEGOUT: SEG口输出模式选择；
00= SEG口输出全为0；
01= SEG口输出全为1；
1x= SEG口输出为LCDDATA中的数据。
Bit3~Bit2 FCCTLM<1:0>: 快速充电模式时间控制位
00= 1/8 COM周期
01= 1/16 COM周期
10= 1/32 COM周期
11= 1/64 COM周期
Bit1~Bit0 VLCDX<1:0>: VLCD输出电压选择
00= 2.6V
01= 2.8V
10= 3.0V
11= 3.3V

LCD 地址寄存器 LCDADD(218H)

218H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDADD	LCDCS	B2RES<1:0>		LCDADD<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 LCDCS: LCD数据读写使能；
0= 禁止读写LCD数据；
1= 允许读写LCD数据。
Bit6~Bit5 B2RES<1:0>: LCD功能内部分压电阻选择
00= R = 10K
01= R = 50K
10= R =200K（自动使能快速充电模式，在一个COM周期开始时50K、200K分压电阻同时有效，经设定时间后50K分压电阻关闭，200K分压电阻保持有效）
11= R =900K（自动使能快速充电模式，在一个COM周期开始时50K、900K分压电阻同时有效，经设定时间后50K分压电阻关闭，900K分压电阻保持有效）
Bit4~Bit0 LCDADD<4:0>: LCD地址选择；
LCD地址范围00H-16H

LCD 数据寄存器 LCDDATA(219H)

219H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDDATA	---	---	LCDDATA<5:0>					
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	---	x	x	x	x	x	x

Bit7~Bit6 保留

Bit5~Bit0 LCDDATA<5:0>: LCD 数据设置，写入 LCDADD 对应地址的数据

LCD 功能 COM 口控制寄存器 COMEN(21AH)

21AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
COMEN			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~Bit6 保留,须写 0

Bit5~Bit0 COMxEN: COMx 口使能位

0= 不使能

1= 使能

LCD 功能 SEG 口控制寄存器 SEGEN0(21BH)

21BH	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~Bit2 SEGxEN: SEG 口功能设置

0= 对应 SEGx 口为普通 I/O 口(x=2~7)

1= 对应 SEGx 口为 LCD 功能的 SEG 口(x=2~7)

Bit1~Bit0 SEGxEN: SEGx 口使能位（x=0~1）

0= 不使能

1= 使能

LCD 功能 SEG 口控制寄存器 SEGEN1(21CH)

21CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGEN1	SEG15EN	SEG14EN	SEG13EN	SEG12EN	SEG11EN	SEG10EN	SEG9EN	SEG8EN
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 SEGxEN: SEG 口功能设置；

0= 对应 SEGx 口为普通 I/O 口(x=8~15)

1= 对应 SEGx 口为 LCD 功能的 SEG 口(x=8~15)

LCD 功能 SEG 口控制寄存器 SEGEN2(21DH)

21DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGEN2	---	SEG22EN	SEG21EN	SEG20EN	SEG19EN	SEG18EN	SEG17EN	SEG16EN
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　　SEGxEN:　SEG 口功能设置

0=　对应 SEGx 口为普通 I/O 口(x=16~22)

1=　对应 SEGx 口为 LCD 功能的 SEG 口(x=16~22)

20.6 快速充电模式设置

当 LCD 内部分压电阻选择 200K 或者 900K 档位时，自动使能快速充电模式。此时 LCD 的时钟分频和快速充电模式时间的组合具有一定的关系，具体关系详见以下表格：

LCDCLK<2:0>	FCCTLM<1:0>=00	FCCTLM<1:0>=01	FCCTLM<1:0>=10	FCCTLM<1:0>=11
0xx				
100				
101				
110				
111				

如上表所示，阴影部分的组合是不允许的。比如当 LCDCLK<2:0>=000，FCCTLM<1:0>就不允许设置为“10”或者“11”。

20.7 LCD 休眠态工作设置

休眠态下 LCD 可以工作在低功耗模式，可参考以下步骤进行操作：

1) 设置 LCD 的 COM 口，SEG 口；
2) 设置 LCD 时钟源，选择外部 32.768KHz 晶振；
3) 设置 LCD 的显示数据，LCD 时钟分频，LCD 偏置电压；
4) 设置 LCD 分压电阻，选择 200K 或者 900K 档位以及设置快速充电时间；
5) 关闭 VLCD 升压泵，由 VDD 为 LCD 供电；
6) 使能 LCD；
7) 执行“STOP”指令进入休眠态。

21. 电气参数

21.1 极限参数

电源供应电压.....GND-0.3V~GND+5.0V
存储温度.....-50℃~125℃
工作温度.....-40℃~85℃
端口输入电压.....GND-0.3V~VDD+0.3V
所有端口最大灌电流.....200mA
所有端口最大拉电流.....-150mA

注：如果器件工作条件超过上述“极限参数”，可能会对器件造成永久性损坏。上述值仅为运行条件极大值，我们不建议器件在该规范规定的范围以外运行。器件长时间工作在极限值条件下，其稳定性会受到影响。

21.2 直流电气特性

（VDD=3.3V，T_A= 25℃，除非另有说明）

符号	参数	测试条件		最小值	典型值	最大值	单位
		VDD	条件				
VDD	工作电压		F_{SYS}=24MHz	2.2		4.5	V
I_{DD}	工作电流	3.3V	F_{SYS}=24MHz		3.6		mA
		2.5V	F_{SYS}=24MHz		3.0		mA
		3.3V	F_{SYS}=32KHz		20		uA
		2.5V	F_{SYS}=32KHz		15		uA
		3V	烧写程序 EEPROM	-	10	-	mA
I_{STB}	静态电流	3.3V	LVR=DIS, WDT=DIS LPLVR=EN			4.0	μA
		2.5V				3.0	μA
		3.3V	LVR=DIS, WDT=EN LPLVR=EN			12.0	μA
		2.5V				11.0	μA
V_{IL}	低电平输入电压		----			0.3VDD	V
V_{IH}	高电平输入电压		----	0.7VDD			V
V_{OH}	高电平输出电压		不带负载	0.9VDD			V
V_{OL}	低电平输出电压		不带负载			0.1VDD	V
V_{EEPROM}	EEPROM 模块擦写电压		----	2.5		4.5	V
R_{PH}	上拉电阻阻值	3.3V	V_O=0.5VDD		30		KΩ
		2.5V	V_O=0.5VDD		34		KΩ
R_{PL}	下拉电阻阻值	3.3V	V_O=0.5VDD		28		KΩ
		2.5V	V_O=0.5VDD		30		KΩ
I_{OL1}	输出口灌电流（普通 I/O 口）	3.3V	V_{OL}=0.3VDD		82		mA
		2.5V	V_{OL}=0.3VDD		54		mA
I_{OL2}	输出口灌电流（COMENA/COMENB 位置 1）	3.3V	V_{OL}=0.3VDD		164		mA
		2.5V	V_{OL}=0.3VDD		110		mA
I_{OH1}	输出口拉电流（普通 I/O 口）	3.3V	V_{OH}=0.7VDD		-45		mA
		2.5V	V_{OH}=0.7VDD		-27		mA
I_{OH2}	输出口拉电流	3.3V	V_{OH}=0.7VDD		-3.6		mA

	（SEGENA/SEGENB 位置 1）		PxPI<7:4>/PxPI<3:0> =0001				
		3.3V	V_{OH}=0.7VDD PxPI<7:4>/PxPI<3:0> =0010		-7.2		mA
		3.3V	V_{OH}=0.7VDD PxPI<7:4>/PxPI<3:0> =0100		-14		mA
		3.3V	V_{OH}=0.7VDD PxPI<7:4>/PxPI<3:0> =1000		-24		mA
		3.3V	V_{OH}=0.7VDD PxPI<7:4>/PxPI<3:0> =1111		-45		mA
V_{BG}	内部基准电压 1.2V	VDD=2.2~4.5V T_A=25℃		-1.5%	1.2	1.5%	V
		VDD=2.2~4.5V T_A= -40~85℃		-2.0%	1.2	2.0%	V

备注：低温规格值由设计保证，量产不测低温条件。

21.3 SAR ADC 电气特性

（T_A= 25℃，除非另有说明）

符号	参数	测试条件	最小值	典型值	最大值	单位
V_{AD}	工作电压范围	-	2.4		4.5	V
NR	精度	GND≤VAIN≤VDD or ADVREF		12		bit
V_{AIN}	输入电压范围	-	0		VDD AD_{VREF}	V
I_{ADC}	-	VDD=3.3V，参考电压=VDD，V_{AIN}=1V		2		mA
DNL1	微分非线性误差 1	VDD=3.3V，参考电压=VDD，AD_CLK=4MHz			±2	LSB
INL1	积分非线性误差 1	VDD=3.3V，参考电压=VDD，AD_CLK=4MHz			±2	LSB
DNL2	微分非线性误差 2	VDD=3.6V，参考电压=3.3V，AD_CLK=4MHz			±3	LSB
INL2	积分非线性误差 2	VDD=3.6V，参考电压=3.3V，AD_CLK=4MHz			±3	LSB
DNL3	微分非线性误差 3	VDD=3.3V，参考电压=3.0V，AD_CLK=4MHz			±3	LSB
INL3	积分非线性误差 3	VDD=3.3V，参考电压=3.0V，AD_CLK=4MHz			±3	LSB
DNL4	微分非线性误差 4	VDD=3.3V，参考电压=2.6V，AD_CLK=4MHz			±3	LSB
INL4	积分非线性误差 4	VDD=3.3V，参考电压=2.6V，AD_CLK=4MHz			±3	LSB
DNL5	微分非线性误差 5	VDD=3.3V，参考电压=2.4V，AD_CLK=2MHz			±3	LSB
INL5	积分非线性误差 5	VDD=3.3V，参考电压=2.4V，AD_CLK=2MHz			±3	LSB
F_{ADC}	转换时钟	VDD=3.3V，参考电压=VDD			4	MHz
		VDD=2.2V，参考电压=VDD			2	MHz
		VDD=3.6V，参考电压=3.3V			4	MHz
		VDD=3.3V，参考电压=3.0V			4	MHz
		VDD=3.3V，参考电压=2.6V			4	MHz
		VDD=3.3V，参考电压=2.4V			2	MHz
T_{ADC}	转换时间	AD_CLK=2MHz		8		uS
AD_{VREF1}	VREF1	VDD=2.6~4.5	-4%	2.4	+4%	V
AD_{VREF2}	VREF2	VDD=2.8~4.5	-4%	2.6	+4%	V
AD_{VREF3}	VREF3	VDD=3.2~4.5	-4%	3.0	+4%	V
AD_{VREF4}	VREF4	VDD=3.5~4.5	-4%	3.3	+4%	V

备注：低温规格值由设计保证，量产不测低温条件。

21.4 AFE 电气特性

参数	条件	最小值	典型值	最大值	单位
模拟输入					
满幅差分输入电压	-	-VS/PGA	-	VS/PGA	V
共模输入电压	-	GND+0.75	-	VS-1	V
差分输入阻抗	-	-	250	-	Mohm
系统性能					
分辨率	无失码数据	-	24	-	bits
输出速率	-	10.2	10.2	10.4K	Hz
建立时间	全建立	-	-	3	转换周期
等效输入噪声	PGA=128, 10Hz, LPWRPGA=1, LDO=2.4V	-	40	-	nVrms
有效分辨率	PGA=128, 10Hz, LPWRPGA=1, LDO=2.4V	-	19.9	-	bits
失调误差	PGA=64,128	-	2.5	-	uV
失调误差漂移	PGA=64,128	-	30	-	nV/℃
增益误差	PGA=64,128	-	±1.5	-	%
增益误差漂移	PGA=64,128	-	16	-	ppm/℃
参考电压输入	-	0.5	VS	VS	V
温感	-	-	±3	-	℃
带隙基准电压	VDD=3.3V	-	1.20	-	V
BIM 性能					
DAC 分辨率		-	8	-	Bits
DAC 参考电压		-	VS	-	V
限流电阻 R0		-	1000	-	Ohm
正弦激励波频率		5		500	KHz
正弦激励电流		-	322	-	uArms
人体阻抗测量范围		100		2K	Ohm
100~2KΩ电阻测试误差	FWR 模式，校准后	-	0.5	-	%
LDO 电气特性					
输出电压	SET_LDO<1:0>=00	-	3.03	-	V
	SET_LDO<1:0>=10	-	2.62	-	V
带载能力	VDD=3.3V	-	20	-	mA
电源电气特性					
电源电压	-	2.4	3.3	4.4	V
正常工作电流	PGA=128, LPWRPGA=1	-	1	-	mA

备注：低温规格值由设计保证，量产不测低温条件。

表 21-1 是 ADC 在不同的输出速率、PGA 增益条件下的有效分辨率（Effective Resolution）。测试条件：电源电压 3.3V，温度 27 度，LDO 设定为 2.4V 输出，参考电压为 LDO 输出电压，输入共模电压为 0.5 倍 LDO 输出电压，输入差分电压为 3mV，单颗芯片每种设置下的数据总量为 1000。

Effective Resolution=Log2 (2*REFIN /RMS_Noise)

表 21-1：DeltaSigma ADC 的有效分辨率

Effective Resolution	FADC	666.7K (FADC=1)							
	ODR	10.2	20.3	40.7	163	651	1302	2604	5208
PGA Gain	2	22.5	22.1	21.5	20.5	18.0	15.6	13.9	12.0
	4	22.3	21.9	21.4	20.1	18.7	16.2	13.5	11.8
	12	21.6	21.2	20.8	19.3	18.1	16.8	14.5	11.2
	16	21.9	21.5	21.0	19.1	18.0	16.7	14.4	11.8
	48	20.6	20.1	19.8	17.6	16.6	16.1	14.4	12.0
	64	20.8	20.5	20.2	17.3	16.3	15.9	14.4	12.0
	128	19.9	19.5	19.2	16.3	15.3	15.0	14.2	11.9
	384	18.0	17.8	17.5	14.7	13.7	13.5	13.2	11.9

21.5 上电复位特性

（T_A= 25℃，除非另有说明）

符号	参数	测试条件	最小值	典型值	最大值	单位
T_{VDD}	VDD 上升速率	-	0.05	-	-	V/ms
V_{LVR}	LVR 电压=2.2V	VDD=1.8~4.5V	2.0	2.2	2.3	V

备注：低温规格值由设计保证，量产不测低温条件。

21.6 LVD 电气特性

（T_A= 25℃，除非另有说明）

符号	参数	测试条件	最小值	典型值	最大值	单位
V_{LVD}	工作电压	-	2.2	-	4.5	V
-	精度	VDD=2.2~4.5V, T_A= -40~85℃	-5%	V_{SET}	+5%	V

备注：低温规格值由设计保证，量产不测低温条件。

21.7 交流电气特性

（T_A=25℃，除非另有说明）

符号	参数	测试条件		最小值	典型值	最大值	单位
		VDD	条件				
T_{WDT}	WDT 复位时间	3.3V	-	-	16	-	ms
		2.5V	-	-	16	-	ms
T_{EEPROM}	EEPROM 编程时间	3.3V	F_{HSI}=24MHz	-	-	10	ms
		2.5V	F_{HSI}=24MHz			10	ms
F_{RC}	内振频率稳定性	VDD=2.4~4.5V T_A=25℃		-1.6%	24	+1.6%	MHz
		VDD=2.2~4.5V T_A=25℃		-2%	24	+2%	MHz
		VDD=2.4~4.5V T_A= -40~85℃		-2.5%	24	+2.5%	MHz
		VDD=2.2~4.5V T_A= -40~85℃		-3.5%	24	+3.5%	MHz

备注：低温规格值由设计保证，量产不测低温条件。

22. 指令

22.1 指令一览表

助记符		操作	指令周期	标志
控制类 4				
NOP		空操作	1	None
STOP		进入休眠模式	1	TO,PD
CLRWDT		清零看门狗计数器	1	TO,PD
RESET		软件器件复位	1	None
数据传送 6				
LD	[R],A	将 ACC 内容传送到 R	1	NONE
LD	A,[R]	将 R 内容传送到 ACC	1	Z
TESTZ	[R]	将数据存储器内容传给数据存储器	1	Z
LDIA	i	立即数 i 送给 ACC	1	NONE
LDIB	i	立即数 i 送给 BSR 寄存器	1	NONE
LDIP	i	立即数 i 送给 PCLATH 寄存器	1	NONE
逻辑运算 15				
CLRA		清零 ACC	1	Z
CLR	[R]	清零数据存储器 R	1	Z
ORA	[R]	R 与 ACC 内容做“或”运算，结果存入 ACC	1	Z
ORR	[R]	R 与 ACC 内容做“或”运算，结果存入 R	1	Z
ANDA	[R]	R 与 ACC 内容做“与”运算，结果存入 ACC	1	Z
ANDR	[R]	R 与 ACC 内容做“与”运算，结果存入 R	1	Z
XORA	[R]	R 与 ACC 内容做“异或”运算，结果存入 ACC	1	Z
XORR	[R]	R 与 ACC 内容做“异或”运算，结果存入 R	1	Z
SWAPA	[R]	R 寄存器内容的高低半字节转换，结果存入 ACC	1	NONE
SWAPR	[R]	R 寄存器内容的高低半字节转换，结果存入 R	1	NONE
COMA	[R]	R 寄存器内容取反，结果存入 ACC	1	Z
COMR	[R]	R 寄存器内容取反，结果存入 R	1	Z
XORIA	i	ACC 与立即数 i 做“异或”运算，结果存入 ACC	1	Z
ANDIA	i	ACC 与立即数 i 做“与”运算，结果存入 ACC	1	Z
ORIA	i	ACC 与立即数 i 做“或”运算，结果存入 ACC	1	Z
移位操作 10				
RRCA	[R]	数据存储器带进位循环右移一位，结果存入 ACC	1	C
RRCR	[R]	数据存储器带进位循环右移一位，结果存入 R	1	C
RLCA	[R]	数据存储器带进位循环左移一位，结果存入 ACC	1	C
RLCR	[R]	数据存储器带进位循环左移一位，结果存入 R	1	C
ARCA	[R]	数据存储器带进位算术右移一位，结果存入 ACC	1	C,Z
ARCR	[R]	数据存储器带进位算术右移一位，结果存入 R	1	C,Z
LLCA	[R]	数据存储器带进位逻辑左移一位，结果存入 ACC	1	C,Z
LLCR	[R]	数据存储器带进位逻辑左移一位，结果存入 R	1	C,Z
LRCA	[R]	数据存储器带进位逻辑右移一位，结果存入 ACC	1	C,Z
LRCR	[R]	数据存储器带进位逻辑右移一位，结果存入 R	1	C,Z
递增递减 4				

助记符		操作	指令周期	标志
INCA	[R]	递增数据存储器 R，结果放入 ACC	1	Z
INCR	[R]	递增数据存储器 R，结果放入 R	1	Z
DECA	[R]	递减数据存储器 R，结果放入 ACC	1	Z
DECR	[R]	递减数据存储器 R，结果放入 R	1	Z
位操作 2				
CLRB	[R],b	将数据存储器 R 中某位清零	1	NONE
SETB	[R],b	将数据存储器 R 中某位置一	1	NONE
数学运算 12				
ADDA	[R]	ACC+[R]→ACC	1	C,DC,Z
ADDR	[R]	ACC+[R]→R	1	C,DC,Z
ADDCA	[R]	ACC+[R]+C→ACC	1	Z,C,DC
ADDCR	[R]	ACC+[R]+C→R	1	Z,C,DC
ADDIA	i	ACC+i→ACC	1	Z,C,DC
SUBA	[R]	[R]-ACC→ACC	1	C,DC,Z
SUBR	[R]	[R]-ACC→R	1	C,DC,Z
SUBNCA	[R]	[R]-ACC- $\overline{C}$ →ACC	1	Z,C,DC
SUBNCR	[R]	[R]-ACC- $\overline{C}$ →R	1	Z,C,DC
SUBIA	i	i-ACC→ACC	1	Z,C,DC
DAA		将加法运算中放入ACC 的值调整为十进制数，并将结果放入ACC寄存器中	1	C
DAS		将减法运算中放入ACC 的值调整为十进制数，并将结果放入ACC寄存器中	1	C
无条件转移 8				
RET		从子程序返回	2	NONE
RET	i	从子程序返回，并将立即数 I 存入 ACC	2	NONE
RETI		从中断返回	2	NONE
CALL	ADD	子程序调用	2	NONE
CALLA		调用地址由 ACC 寄存器指定的子程序	2	NONE
JP	ADD	无条件跳转	2	NONE
JPI	ADD	无条件跳转	2	NONE
JPA		将 ACC 寄存器的内容作为偏移量进行相对跳转	2	NONE
条件转移 6				
SZB	[R],b	如果数据存储器 R 的 b 位为“0”，则跳过下一条指令	1 or 2	NONE
SNZB	[R],b	如果数据存储器 R 的 b 位为“1”，则跳过下一条指令	1 or 2	NONE
SZINCA	[R]	数据存储器 R 加“1”，结果放入 ACC，若结果为“0”，则跳过下一条指令	1 or 2	NONE
SZINCR	[R]	数据存储器 R 加“1”，结果放入 R，若结果为“0”，则跳过下一条指令	1 or 2	NONE
SZDECA	[R]	数据存储器 R 减“1”，结果放入 ACC，若结果为“0”，则跳过下一条指令	1 or 2	NONE
SZDECR	[R]	数据存储器 R 减“1”，结果放入 R，若结果为“0”，则跳过下一条指令	1 or 2	NONE
C 编译器优化 5				
ADDFSR	n,i	立即数 i 与 FSRn 相加，数据放入 FSRn	1	NONE
LDINA	n,mm	将间接寄存器 INDFn 的数据放入 ACC，带预/后递增/递减	1	NONE
LDINAI	n,i	将间接寄存器 INDFn 的数据放入 ACC，采用变址间接寻址模式	1	Z
LDAIN	n,mm	将 ACC 的数据放入间接寄存器 INDFn，带预/后递增/递减	1	NONE
LDAINI	n,i	将 ACC 的数据放入间接寄存器 INDFn，采用变址间接寻址模式	1	Z

22.2 指令说明

ADDA **[R]**

操作: 将 R 加 ACC，结果放入 ACC

周期: 1

影响标志位: C，DC，Z，OV

举例:

```
LDIA    09H     ;给 ACC 赋值 09H
LD      R01,A   ;将 ACC 的值（09H）赋给自定义寄存器 R01
LDIA    077H    ;给 ACC 赋值 77H
ADDA    R01     ;执行结果：ACC=09H + 77H =80H
```

ADDR **[R]**

操作: 将 R 加 ACC，结果放入 R

周期: 1

影响标志位: C，DC，Z，OV

举例:

```
LDIA    09H     ;给 ACC 赋值 09H
LD      R01,A   ;将 ACC 的值（09H）赋给自定义寄存器 R01
LDIA    077H    ;给 ACC 赋值 77H
ADDR    R01     ;执行结果：R01=09H + 77H =80H
```

ADDCA **[R]**

操作: 将 R 加 ACC 加 C 位，结果放入 ACC

周期: 1

影响标志位: C，DC，Z，OV

举例:

```
LDIA    09H     ;给 ACC 赋值 09H
LD      R01,A   ;将 ACC 的值（09H）赋给自定义寄存器 R01
LDIA    077H    ;给 ACC 赋值 77H
ADDCA   R01     ;执行结果：ACC= 09H + 77H + C=80H (C=0)
                           ACC= 09H + 77H + C=81H (C=1)
```

ADDCR **[R]**

操作: 将 R 加 ACC 加 C 位，结果放入 R

周期: 1

影响标志位: C，DC，Z，OV

举例:

```
LDIA    09H     ;给 ACC 赋值 09H
LD      R01,A   ;将 ACC 的值（09H）赋给自定义寄存器 R01
LDIA    077H    ;给 ACC 赋值 77H
ADDCR   R01     ;执行结果：R01 = 09H + 77H + C=80H (C=0)
                           R01 = 09H + 77H + C=81H (C=1)
```

ADDIA **i**

操作: 将立即数 i 加 ACC，结果放入 ACC

周期: 1

影响标志位: C，DC，Z，OV

举例:

```
LDIA    09H     ;给 ACC 赋值 09H
ADDIA   077H    ;执行结果：ACC = ACC(09H) + i(77H)=80H
```

ANDA **[R]**

操作: 寄存器 R 和 ACC 进行逻辑与运算，结果放入 ACC

周期: 1

影响标志位: Z

举例:

```
LDIA    0FH     ;给 ACC 赋值 0FH
LD      R01,A   ;将 ACC 的值(0FH)赋给寄存器 R01
LDIA    77H     ;给 ACC 赋值 77H
ANDA    R01     ;执行结果：ACC=(0FH  and  77H)=07H
```

ANDR **[R]**

操作: 寄存器 R 和 ACC 进行逻辑与运算，结果放入 R

周期: 1

影响标志位: Z

举例:

```
LDIA    0FH     ;给 ACC 赋值 0FH
LD      R01,A   ;将 ACC 的值(0FH)赋给寄存器 R01
LDIA    77H     ;给 ACC 赋值 77H
ANDR    R01     ;执行结果：R01=(0FH  and  77H)=07H
```

ANDIA **i**

操作: 将立即数 i 与 ACC 进行逻辑与运算，结果放入 ACC

周期: 1

影响标志位: Z

举例:

```
LDIA    0FH     ;给 ACC 赋值 0FH
ANDIA   77H     ;执行结果：ACC =(0FH  and  77H)=07H
```

ARCA **[R]**

操作: 寄存器 R 带 C 算术右移一位，结果放入 ACC

C=R<0>
ACC<6:0>=R<7:1>
ACC<7>=R<7>

周期: 1

影响标志位: C,Z

举例:

```
LDIA    083H        ;ACC 赋值 03H
LD      R01,A       ;ACC 值赋给 R01,R01=083H
ARCA    R01         ;操作结果：ACC=0C1H，C=1;
```

操作说明 将寄存器 R 的内容连同进位标志位一起右移 1 位，MSb 保持不变。

ARCR **[R]**

操作: 寄存器 R 带 C 算术右移一位，结果放入 R

C=R<0>
R<6:0>=R<7:1>
R<7>=R<7>

周期: 1

影响标志位: C,Z

举例:

```
LDIA    083H        ;ACC 赋值 03H
LD      R01,A       ;ACC 值赋给 R01,R01=083H
ARCR    R01         ;操作结果：R01=0C1H，C=1;
```

操作说明 将寄存器 R 的内容连同进位标志位一起右移 1 位，MSb 保持不变。

ADDFSR **n,i**

操作: 立即数 i 与 FSRn 相加，结果放入 R

n=0,1,FSRn 或者 INDFn，i 的范围为-32≤i＜32

周期: 1

影响标志位: 无

举例:

```
ADDFSR  0,1         ;1 和 FSR0 的内容相加，结果放入 FSR0
ADDFSR  FSR1,2      ;2 和 FSR1 的内容相加，结果放入 FSR1
ADDFSR  INDF1,-4    ; FSR1 的内容与 4 相减，结果放入 FSR1
```

操作说明 将寄存器 R 的内容连同进位标志位一起右移 1 位，MSb 保持不变。

CALL **add**

操作: 调用子程序

周期: 2

影响标志位: 无

举例:

```
CALL    LOOP        ;调用名称定义为”LOOP”的子程序地址
```

CALLA **add**

操作: 调用由 ACC 寄存器指定的子程序地址

周期: 2

影响标志位: 无

举例:

```
CALLA                   ;调用 ACC 寄存器内容指定的子程序地址
```

CLRA

操作: ACC 清零

周期: 1

影响标志位: Z

举例:

```
CLRA                    ;执行结果：ACC=0
```

CLR **[R]**

操作: 寄存器 R 清零

周期: 1

影响标志位: Z

举例:

```
CLR     R01             ;执行结果：R01=0
```

CLRB **[R],b**

操作: 寄存器 R 的第 b 位清零

周期: 1

影响标志位: 无

举例:

```
CLRB    R01,3           ;执行结果：R01 的第 3 位为零
```

CLRWDT

操作: 清零看门狗计数器

周期: 1

影响标志位: TO，PD

举例:

```
CLRWDT                  ;看门狗计数器清零
```

COMA **[R]**

操作: 寄存器 R 取反，结果放入 ACC

周期: 1

影响标志位: Z

举例:

```
LDIA    0AH             ;ACC 赋值 0AH
LD      R01,A           ;将 ACC 的值(0AH)赋给寄存器 R01
COMA    R01             ;执行结果：ACC=0F5H
```

COMR **[R]**

操作: 寄存器 R 取反，结果放入 R

周期: 1

影响标志位: Z

举例:

```
LDIA    0AH     ;ACC 赋值 0AH
LD      R01,A   ;将 ACC 的值(0AH)赋给寄存器 R01
COMR    R01     ;执行结果：R01=0F5H
```

DECA **[R]**

操作: 寄存器 R 自减 1，结果放入 ACC

周期: 1

影响标志位: Z

举例:

```
LDIA    0AH     ;ACC 赋值 0AH
LD      R01,A   ;将 ACC 的值(0AH)赋给寄存器 R01
DECA    R01     ;执行结果：ACC=(0AH-1)=09H
```

DECR **[R]**

操作: 寄存器 R 自减 1，结果放入 R

周期: 1

影响标志位: Z

举例:

```
LDIA    0AH     ;ACC 赋值 0AH
LD      R01,A   ;将 ACC 的值(0AH)赋给寄存器 R01
DECR    R01     ;执行结果：R01=(0AH-1)=09H
```

DAA

操作: 将加法运算中放入ACC 的值调整为十进制数，并将结果放入ACC寄存器中

周期: 1

影响标志位: C

举例:

```
LDIA    09H     ;ACC 赋值 09H
LD      R01,A   ;将 ACC 的值(09H)赋给寄存器 R01
LDIA    02H     ;ACC 赋值 02H
ADDA    R01     ;执行结果，ACC=0BH
DAA             ;执行结果，ACC=11H
```

DAS

操作: 将减法运算中放入ACC 的值调整为十进制数，并将结果放入ACC寄存器中

周期: 1

影响标志位: C

举例:

```
LDIA    12H     ;ACC 赋值 12H
LD      R01,A   ;将 ACC 的值(12H)赋给寄存器 R01
LDIA    03H     ;ACC 赋值 03H
SUBA    R01     ;执行结果，ACC=0FH
DAS             ;执行结果，ACC=09H
```

INCA [R]

操作: 寄存器 R 自加 1，结果放入 ACC

周期: 1

影响标志位: Z

举例:

```
LDIA    0AH     ;ACC 赋值 0AH
LD      R01,A   ;将 ACC 的值(0AH)赋给寄存器 R01
INCA    R01     ;执行结果：ACC=(0AH+1)=0BH
```

INCR [R]

操作: 寄存器 R 自加 1，结果放入 R

周期: 1

影响标志位: Z

举例:

```
LDIA    0AH     ;ACC 赋值 0AH
LD      R01,A   ;将 ACC 的值(0AH)赋给寄存器 R01
INCR    R01     ;执行结果：R01=(0AH+1)=0BH
```

JP add

操作: 跳转到 add 地址

周期: 2

影响标志位: 无

举例:

```
JP      LOOP    ;跳转至名称定义为”LOOP”的子程序地址
```

JPI add

操作: 跳转到 add 地址,相对跳转

周期: 2

影响标志位: 无

举例:

```
JPI     LOOP    ;跳转至名称定义为”LOOP”的子程序地址
```

JPA

操作: 将 ACC 寄存器的内容作为偏移量进行相对跳转
周期: 2
影响标志位: 无
举例:

```
JPA                 ;跳转到 ACC 内容指定的子程序地址
```

LD A,[R]

操作: 将 R 的值赋给 ACC
周期: 1
影响标志位: Z
举例:

```
LD      A,R01       ;将寄存器 R0 的值赋给 ACC
LD      R02,A       ;将 ACC 的值赋给寄存器 R02，实现了数据从 R01➞R02 的移动
```

LD [R],A

操作: 将 ACC 的值赋给 R
周期: 1
影响标志位: 无
举例:

```
LDIA    09H         ;给 ACC 赋值 09H
LD      R01,A       ;执行结果：R01=09H
```

LDIA i

操作: 立即数 i 赋给 ACC
周期: 1
影响标志位: 无
举例:

```
LDIA    0AH         ;ACC 赋值 0AH
```

LDINA n,mm

操作: 将间接寄存器 INDFn 的数据放入 ACC，带预/后递增/递减
周期: 1
影响标志位: 无
举例:

```
LDINA   ++INDF0     将间接寄存器 INDF0 的内容放入 ACC，有效地址为 FSR0+1,执行指令后，FSR0 加 1
LDINA   ++FSR0      与上一条指令执行效果一样
LDINA   --INDF0     将间接寄存器 INDF0 的内容放入 ACC，有效地址为 FSR0-1,执行指令后，FSR0 减 1
LDINA   --FSR0      与上一条指令执行效果一样
LDINA   INDF0++     将间接寄存器 INDF0 的内容放入 ACC，有效地址为 FSR0,执行指令后，FSR0 加 1
LDINA   FSR0++      与上一条指令执行效果一样
LDINA   INDF0--     将间接寄存器 INDF0 的内容放入 ACC，有效地址为 FSR0,执行指令后，FSR0 减 1
LDINA   FSR0--      与上一条指令执行效果一样
```

LDINAI **n,i**

操作: 将间接寄存器 INDFn 的数据放入 ACC，采用变址间接寻址方式

n=0,1,INDFn,FSRn,i 的范围为-32≤i＜32

周期: 1

影响标志位: Z

举例:

LDINAI	INDF0,5	将间接寄存器 INDF0 的内容放入 ACC，有效地址为 FSR0+5,执行指令后，FSR0 不变
LDINAI	FSR0,5	与上一条指令执行效果一样
LDINAI	0,5	与上一条指令执行效果一样
LDINAI	INDF0,-32	将间接寄存器 INDF0 的内容放入 ACC，有效地址为 FSR0-32,执行指令后，FSR0 不变

LDAIN **n,mm**

操作: 将 ACC 的数据放入间接寄存器 INDFn，带预/后递增/递减

周期: 1

影响标志位: 无

举例:

LDAIN	++INDF0	将 ACC 的数据放入间接寄存器 INDFn，有效地址为 FSR0+1,执行指令后，FSR0 加 1
LDAIN	++FSR0	与上一条指令执行效果一样
LDAIN	--INDF0	将 ACC 的数据放入间接寄存器 INDFn，有效地址为 FSR0-1,执行指令后，FSR0 减 1
LDAIN	--FSR0	与上一条指令执行效果一样
LDAIN	INDF0++	将 ACC 的数据放入间接寄存器 INDFn，有效地址为 FSR0,执行指令后，FSR0 加 1
LDAIN	FSR0++	与上一条指令执行效果一样
LDAIN	INDF0--	将 ACC 的数据放入间接寄存器 INDFn，有效地址为 FSR0,执行指令后，FSR0 减 1
LDAIN	FSR0--	与上一条指令执行效果一样

LDAINI **n,i**

操作: 将 ACC 的数据放入间接寄存器 INDFn，采用变址间接寻址方式

n=0,1,INDFn,FSRn,i 的范围为-32≤i＜32

周期: 1

影响标志位: Z

举例:

LDAINI	INDF0,5	将 ACC 的数据放入间接寄存器 INDFn，有效地址为 FSR0+5,执行指令后，FSR0 不变
LDAINI	FSR0,5	与上一条指令执行效果一样
LDAINI	0,5	与上一条指令执行效果一样
LDAINI	INDF0,-32	将 ACC 的数据放入间接寄存器 INDFn，有效地址为 FSR0-32,执行指令后，FSR0 不变

LLCA **[R]**

操作: 数据存储器带进位逻辑左移一位，结果存入 ACC

C=R<7>
ACC<7:1>=R<6:0>
ACC<0>=0

周期: 1

影响标志位: C,Z

举例:

```
LDIA    03H     ;ACC 赋值 03H
LD      R01,A   ;ACC 值赋给 R01,R01=03H
LLCA    R01     ;操作结果：ACC=04H,C=0

LDIA    09H     ;ACC 赋值 09H
LD      R01,A   ;ACC 值赋给 R01,R01=09H
LLCA    R01     ;操作结果：ACC=02H,C=1
```

LLCR **[R]**

操作: 数据存储器带进位循环左移一位，结果存入 R

C=R<7>
R<7:1>=R<6:0>
R<0>=0

周期: 1

影响标志位: C,Z

举例:

```
LDIA    03H     ;ACC 赋值 03H
LD      R01,A   ;ACC 值赋给 R01,R01=03H
LLCR    R01     ;操作结果：R=04H,C=0

LDIA    09H     ;ACC 赋值 09H
LD      R01,A   ;ACC 值赋给 R01,R01=09H
LLCR    R01     ;操作结果：R01=02H,C=1
```

LRCA **[R]**

操作: 数据存储器带进位逻辑右移一位，结果存入 ACC

C=R<0>
ACC<6:0>=R<7:1>
ACC<7>=0

周期: 1

影响标志位: C,Z

举例:

```
LDIA    03H     ;ACC 赋值 03H
LD      R01,A   ;ACC 值赋给 R01,R01=03H
LRCA    R01     ;操作结果：ACC=01H,C=1

LDIA    080H    ;ACC 赋值 80H
LD      R01,A   ;ACC 值赋给 R01,R01=80H
LRCA    R01     ;操作结果：ACC=40H,C=0
```

LRCR **[R]**

操作: 数据存储器带进位逻辑右移一位，结果存入 ACC

C=R<0>
R<6:0>=R<7:1>
R<7>=0

周期: 1

影响标志位: C,Z

举例:

```
LDIA    03H     ;ACC 赋值 03H
LD      R01,A   ;ACC 值赋给 R01,R01=03H
LRCR    R01     ;操作结果：R01=01H,C=1
LDIA    080H    ;ACC 赋值 80H
LD      R01,A   ;ACC 值赋给 R01,R01=80H
LRCR    R01     ;操作结果：R01=40H,C=0
```

NOP

操作: 空指令

周期: 1

影响标志位: 无

举例:

```
NOP
NOP
```

ORIA **i**

操作: 立即数与 ACC 进行逻辑或操作，结果赋给 ACC

周期: 1

影响标志位: Z

举例:

```
LDIA    0AH     ;ACC 赋值 0AH
ORIA    030H    ;执行结果：ACC =(0AH  or  30H)=3AH
```

ORA **[R]**

操作: 寄存器 R 跟 ACC 进行逻辑或运算，结果放入 ACC

周期: 1

影响标志位: Z

举例:

```
LDIA    0AH     ;给 ACC 赋值 0AH
LD      R01,A   ;将 ACC(0AH)赋给寄存器 R01
LDIA    30H     ;给 ACC 赋值 30H
ORA     R01     ;执行结果：ACC=(0AH  or  30H)=3AH
```

ORR **[R]**

操作: 寄存器 R 跟 ACC 进行逻辑或运算，结果放入 R

周期: 1

影响标志位: Z

举例:

```
        LDIA    0AH         ;给 ACC 赋值 0AH
        LD      R01,A       ;将 ACC(0AH)赋给寄存器 R01
        LDIA    30H         ;给 ACC 赋值 30H
        ORR     R01         ;执行结果：R01=(0AH  or  30H)=3AH
```

RET

操作: 从子程序返回

周期: 2

影响标志位: 无

举例:

```
        CALL    LOOP        ;调用子程序 LOOP
        NOP                 ;RET 指令返回后将执行这条语句
        …                   ;其它程序
LOOP:
        …                   ;子程序
        RET                 ;子程序返回
```

RET **i**

操作: 从子程序带参数返回，参数放入 ACC

周期: 2

影响标志位: 无

举例:

```
        CALL    LOOP        ;调用子程序 LOOP
        NOP                 ;RET 指令返回后将执行这条语句
        …                   ;其它程序
LOOP:
        …                   ;子程序
        RET     35H         ;子程序返回,ACC=35H
```

RETI

操作: 中断返回

周期: 2

影响标志位: 无

举例:

```
INT_START                   ;中断程序入口
        …                   ;中断处理程序
        RETI                ;中断返回
```

RLCA **[R]**

操作: 寄存器 R 带 C 循环左移一位，结果放入 ACC

周期: 1

影响标志位: C

举例:

```
LDIA    03H         ;ACC 赋值 03H
LD      R01,A       ;ACC 值赋给 R01,R01=03H
RLCA    R01         ;操作结果：ACC=06H(C=0);
                            ACC=07H(C=1)
                            C=0
```

RLCR **[R]**

操作: 寄存器 R 带 C 循环左移一位，结果放入 R

周期: 1

影响标志位: C

举例:

```
LDIA    03H         ;ACC 赋值 03H
LD      R01,A       ;ACC 值赋给 R01,R01=03H
RLCR    R01         ;操作结果：R01=06H(C=0);
                            R01=07H(C=1);
                            C=0
```

RRCA **[R]**

操作: 寄存器 R 带 C 循环右移一位，结果放入 ACC

周期: 1

影响标志位: C

举例:

```
LDIA    03H         ;ACC 赋值 03H
LD      R01,A       ;ACC 值赋给 R01,R01=03H
RRCA    R01         ;操作结果：ACC=01H(C=0);
                            ACC=081H(C=1);
                            C=1
```

RRCR **[R]**

操作: 寄存器 R 带 C 循环右移一位，结果放入 R

周期: 1

影响标志位: C

举例:

```
LDIA    03H         ;ACC 赋值 03H
LD      R01,A       ;ACC 值赋给 R01,R01=03H
RRCR    R01         ;操作结果：R01=01H(C=0);
                            R01=81H(C=1);
                            C=1
```

RESET

操作: 软件复位器件

周期: 1

影响标志位: 无

举例:

```
RESET                   ;复位器件
```

SETB [R],b

操作: 寄存器 R 的第 b 位置 1

周期: 1

影响标志位: 无

举例:

```
CLR     R01             ;R01=0
SETB    R01,3           ;操作结果：R01=08H
```

STOP

操作: 进入休眠状态

周期: 1

影响标志位: TO，PD

举例:

```
STOP                    ;芯片进入省电模式，CPU、振荡器停止工作，IO 口保持原来状态
```

SUBIA i

操作: 立即数 i 减 ACC，结果放入 ACC

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
LDIA    077H            ;ACC 赋值 77H
SUBIA   80H             ;操作结果：ACC=80H-77H=09H
```

SUBA [R]

操作: 寄存器 R 减 ACC，结果放入 ACC

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
LDIA    080H            ;ACC 赋值 80H
LD      R01,A           ;ACC 的值赋给 R01，R01=80H
LDIA    77H             ;ACC 赋值 77H
SUBA    R01             ;操作结果：ACC=80H-77H=09H
```

SUBR **[R]**

操作: 寄存器 R 减 ACC，结果放入 R

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
LDIA    080H    ;ACC 赋值 80H
LD      R01,A   ;ACC 的值赋给 R01，R01=80H
LDIA    77H     ;ACC 赋值 77H
SUBR    R01     ;操作结果：R01=80H-77H=09H
```

SUBNCA **[R]**

操作: 寄存器 R 减 ACC 减 C 的反，结果放入 ACC

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
LDIA    080H    ;ACC 赋值 80H
LD      R01,A   ;ACC 的值赋给 R01，R01=80H
LDIA    77H     ;ACC 赋值 77H
SUBNCA  R01     ;操作结果：ACC=80H-77H-C=09H(C=1);
                          ACC=80H-77H-C=08H(C=0);
```

SUBNCR **[R]**

操作: 寄存器 R 减 ACC 减 C 的反，结果放入 R

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
LDIA    080H    ;ACC 赋值 80H
LD      R01,A   ;ACC 的值赋给 R01，R01=80H
LDIA    77H     ;ACC 赋值 77H
SUBNCR  R01     ;操作结果：R01=80H-77H-C=09H(C=1)
                          R01=80H-77H-C=08H(C=0)
```

SWAPA **[R]**

操作: 寄存器 R 高低半字节交换，结果放入 ACC

周期: 1

影响标志位: 无

举例:

```
LDIA    035H    ;ACC 赋值 35H
LD      R01,A   ;ACC 的值赋给 R01，R01=35H
SWAPA   R01     ;操作结果：ACC=53H
```

SWAPR **[R]**

操作: 寄存器 R 高低半字节交换，结果放入 R

周期: 1

影响标志位: 无

举例:

```
LDIA    035H    ;ACC 赋值 35H
LD      R01,A   ;ACC 的值赋给 R01，R01=35H
SWAPR   R01     ;操作结果：R01=53H
```

SZB **[R],b**

操作: 判断寄存器 R 的第 b 位，为 0 间跳，否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```
SZB     R01,3   ;判断寄存器 R01 的第 3 位
JP      LOOP    ;R01 的第 3 位为 1 才执行这条语句，跳转至 LOOP
JP      LOOP1   ;R01 的第 3 位为 0 时间跳，执行这条语句，跳转至 LOOP1
```

SNZB **[R],b**

操作: 判断寄存器 R 的第 b 位，为 1 间跳，否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```
SNZB    R01,3   ;判断寄存器 R01 的第 3 位
JP      LOOP    ;R01 的第 3 位为 0 才执行这条语句，跳转至 LOOP
JP      LOOP1   ;R01 的第 3 位为 1 时间跳，执行这条语句，跳转至 LOOP1
```

SZA **[R]**

操作: 将寄存器 R 的值赋给 ACC，若 R 为 0 则间跳，否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```
SZA     R01     ;R01→ACC
JP      LOOP    ;R01 不为 0 时执行这条语句，跳转至 LOOP
JP      LOOP1   ;R01 为 0 时间跳，执行这条语句，跳转至 LOOP1
```

SZR **[R]**

操作: 将寄存器 R 的值赋给 R，若 R 为 0 则间跳，否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```
SZR     R01     ;R01→R01
JP      LOOP    ;R01 不为 0 时执行这条语句，跳转至 LOOP
JP      LOOP1   ;R01 为 0 时间跳执行这条语句，跳转至 LOOP1
```

SZINCA **[R]**

操作: 将寄存器 R 自加 1，结果放入 ACC，若结果为 0，则跳过下一条语句，否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```
SZINCA    R01     ;R01+1→ACC
JP        LOOP    ;ACC 不为 0 时执行这条语句，跳转至 LOOP
JP        LOOP1   ;ACC 为 0 时执行这条语句，跳转至 LOOP1
```

SZINCR **[R]**

操作: 将寄存器 R 自加 1，结果放入 R，若结果为 0，则跳过下一条语句，否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```
SZINCR    R01     ;R01+1→R01
JP        LOOP    ; R01 不为 0 时执行这条语句，跳转至 LOOP
JP        LOOP1   ; R01 为 0 时执行这条语句，跳转至 LOOP1
```

SZDECA **[R]**

操作: 将寄存器 R 自减 1，结果放入 ACC，若结果为 0，则跳过下一条语句，否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```
SZDECA    R01     ;R01-1→ACC
JP        LOOP    ;ACC 不为 0 时执行这条语句，跳转至 LOOP
JP        LOOP1   ;ACC 为 0 时执行这条语句，跳转至 LOOP1
```

SZDECR **[R]**

操作: 将寄存器 R 自减 1，结果放入 R，若结果为 0，则跳过下一条语句，否则顺序执行

周期: 1 or 2

影响标志位: 无

举例:

```
SZDECR    R01     ;R01-1→R01
JP        LOOP    ; R01 不为 0 时执行这条语句，跳转至 LOOP
JP        LOOP1   ; R01 为 0 时执行这条语句，跳转至 LOOP1
```

TESTZ **[R]**

操作: 将 R 的值赋给 R,用以影响 Z 标志位

周期: 1

影响标志位: Z

举例:

```
TESTZ   R0          ;将寄存器 R0 的值赋给 R0，用于影响 Z 标志位
SZB     STATUS,Z    ;判断 Z 标志位，为 0 间跳
JP      Add1        ;当寄存器 R0 为 0 的时候跳转至地址 Add1
JP      Add2        ;当寄存器 R0 不为 0 的时候跳转至地址 Add2
```

XORIA **i**

操作: 立即数与 ACC 进行逻辑异或运算，结果放入 ACC

周期: 1

影响标志位: Z

举例:

```
LDIA    0AH         ;ACC 赋值 0AH
XORIA   0FH         ;执行结果：ACC=05H
```

XORA **[R]**

操作: 寄存器 R 与 ACC 进行逻辑异或运算，结果放入 ACC

周期: 1

影响标志位: Z

举例:

```
LDIA    0AH         ;ACC 赋值 0AH
LD      R01,A       ;ACC 值赋给 R01,R01=0AH
LDIA    0FH         ;ACC 赋值 0FH
XORA    R01         ;执行结果：ACC=05H
```

XORR **[R]**

操作: 寄存器 R 与 ACC 进行逻辑异或运算，结果放入 R

周期: 1

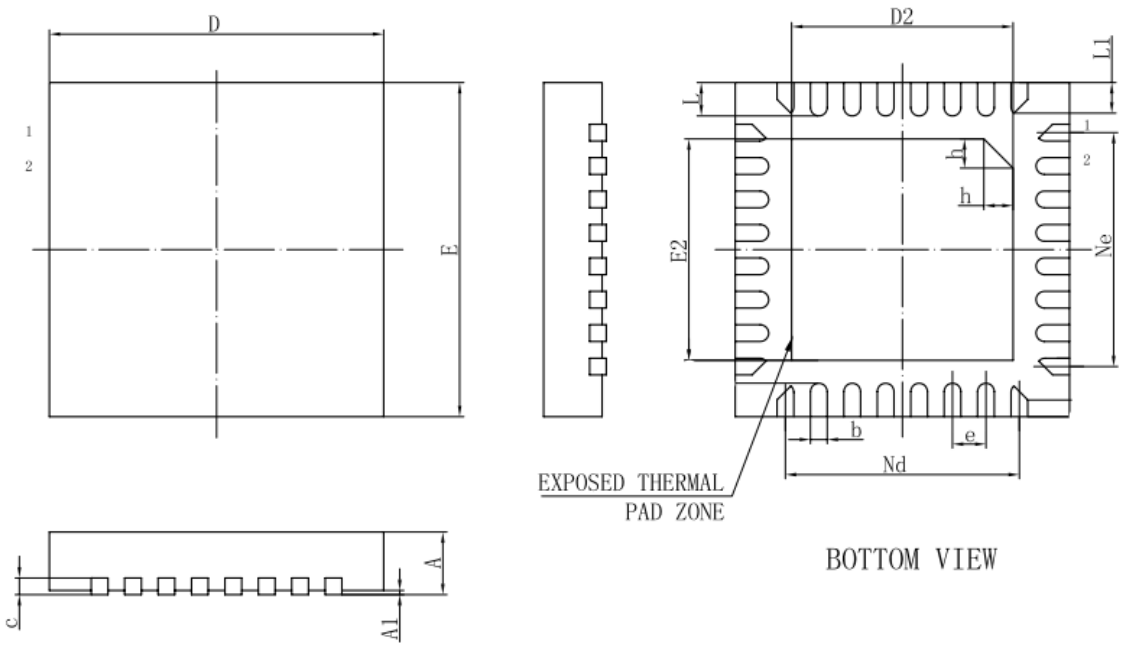
影响标志位: Z

举例:

```
LDIA    0AH         ;ACC 赋值 0AH
LD      R01,A       ;ACC 值赋给 R01,R01=0AH
LDIA    0FH         ;ACC 赋值 0FH
XORR    R01         ;执行结果：R01=05H
```

23. 封装

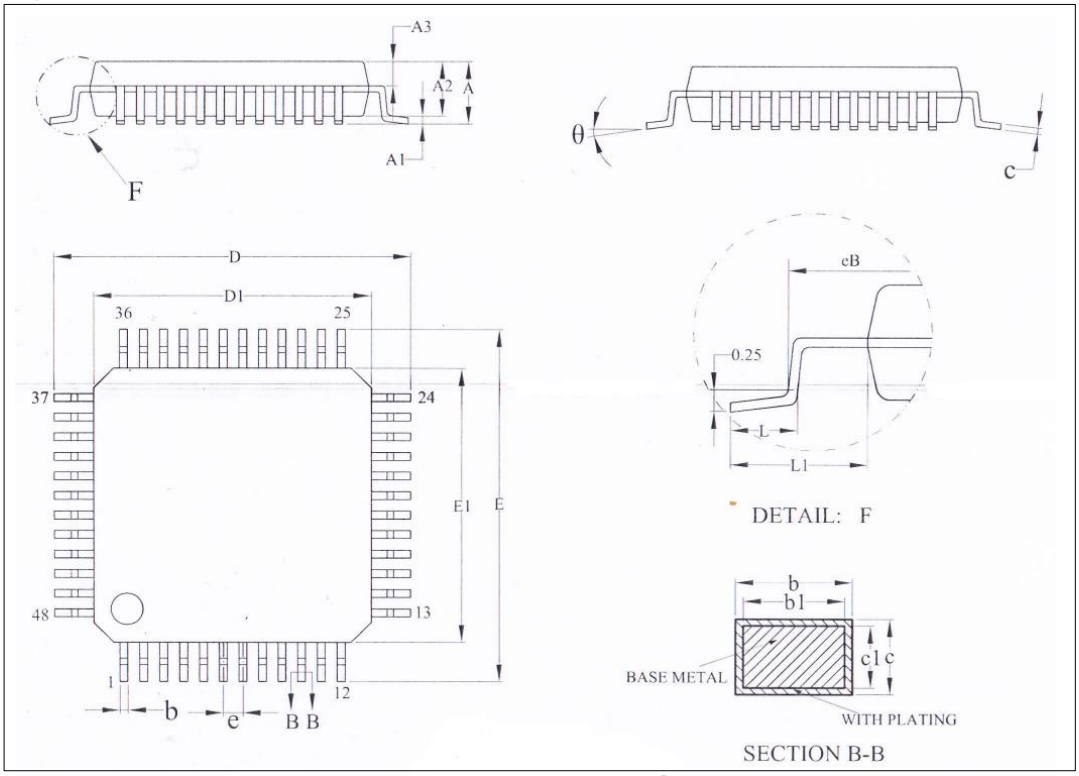
23.1 QFN32(4x4x0.75-0.4)



Symbol	Millimeter		
	Min	Nom	Max
A	0.70	0.75	0.80
A1	0	0.02	0.05
b	0.15	0.20	0.25
c	0.18	0.20	0.25
D	3.90	4.00	4.10
D2	2.60	-	2.80
e	0.40BSC		
Ne	2.80BSC		
Nd	2.80BSC		
E	3.90	4.00	4.10
E2	2.60	-	2.80
L	0.30	0.40	0.45
L1	0.29	0.35	0.40
h	0.30	0.35	0.40

注意：封装尺寸不包括模的毛边凸起或门毛刺。

23.2 LQFP48



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.60
A1	0.05	-	0.15
A2	1.30	1.40	1.50
A3	0.59	0.64	0.69
b	0.18	-	0.26
b1	0.17	0.20	0.23
c	0.13	-	0.18
c1	0.12	0.13	0.14
D	8.80	9.00	9.20
D1	6.90	7.00	7.10
E	8.80	9.00	9.20
E1	6.90	7.00	7.10
eB	8.10	-	8.25
e	0.50BSC		
L	0.43	-	0.75
L1	1.00REF		
θ	0	-	8°

注意：封装尺寸不包括模的毛边凸起或门毛刺。

24. 版本修订说明

版本号	时间	修改内容
V0.1.0	2023 年 7 月	初始版本
V0.1.1	2024 年 4 月	订正 AFE 数据寄存器 AFERES2 描述
V0.9.0	2024 年 6 月	修改 21.2 章节中静态电流最大值
V0.9.1	2024 年 10 月	修改 QFN32/LQFP48 封装尺寸信息