SC8F096 用户手册

增强型闪存 8 位 CMOS 单片机

Rev. 1.0.1

目录

1. 产品概述

1.1 功能特性

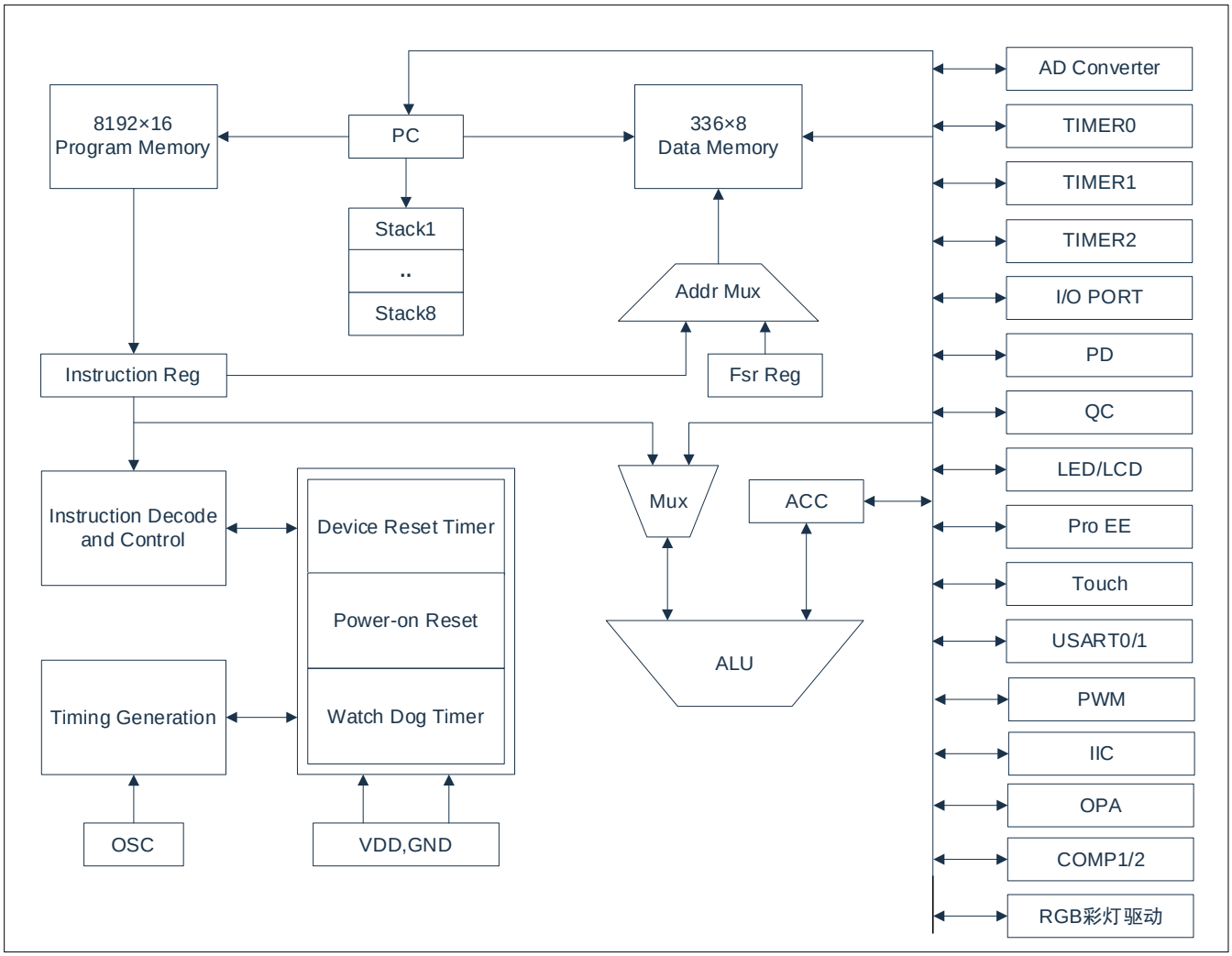
- 内存
  - ROM：8K×16Bit
  - 通用 RAM：336×8Bit
- 8 级堆栈缓存器
- 简洁实用的指令系统（66 条指令）
- 内置低压侦测电路
- 内置 WDT 定时器
- 中断源
  - 3 个定时中断
  - RA、RB 口电平变化中断
  - 其它外设中断
- 内置 256 字程序 EEPROM
  - 可重复擦写 1 万次
- 定时器
  - 8 位定时器 TIMER0，TIMER2
  - 16 位定时器 TIMER1
  - TIMER0、TIMER1、TIMER2 可选择外部 32.768Khz 振荡时钟源
- 内置 2 路比较器模块
  - COMP1 正端可选：RB1/电阻分压输出
  - COMP1 负端可选：RB1/RB2/RB4/RB5/BG/电阻分压输出
  - COMP2 正端可选：RA1/电阻分压出
  - COMP2 负端可选：RA1/RA2/RB0/RB1/BG/电阻分压输出
- 内置 1 路运放模块
- 内置触摸按键模块
  - 需要外挂触摸电容
  - 15 路触摸通道可选
- 内置 PD 模块
  - 仅支持 PD 通讯的 Sink 端
- 工作电压范围：2.5V~5.5V@16MHz/2T
  1.8V~5.5V@16MHz/4T
  工作温度范围：-40℃~85℃
- 内部 RC 振荡：设计频率 16MHz
- 指令周期（单指令或双指令）
- 内置 PWM 模块
  - 5 路 PWM，可设置 2 路互补输出，极性可选
  - 4 路 PWM 共用周期，独立占空比
  - 1 路 PWM 独立周期，独立占空比
  - 10 位 PWM 精度
- LVR 可选 1.8V/2V/2.5V/3V
- 高精度 12 位 ADC
  - 内建高精度 1.2V 基准电压
  - 可选择内部参考源：2.0V/2.4V/3.0V
- 内置 2 路 USART 通信模块
  - 支持同步主从模式和异步全双工模式
- 内置 1 路 IIC 通信 模块
  - 支持主控和从动模式(7 位地址)
  - 从动模式支持广播呼叫
- 内置正反推 LED 硬件驱动模块
  - LED 口灌电流可达 125mA
  - LED 口拉电流可灵活配置为 0~45mA
  - 最多可支持 10×9 个像素点
- 内置 LCD 硬件驱动模块
  - 1/2Bias 和 1/3Bias 输出可选
  - COM×SEG：6×24
  - 休眠态下可选 LSE 或 LSI 作时钟源继续运行
- 内置 RGB 彩灯驱动模块
- 内置 QC 模块

1.2 产品型号一览表

型号说明

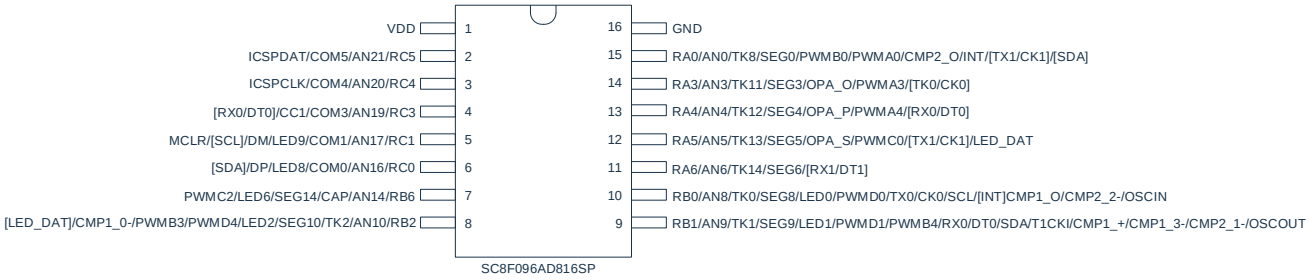
PRODUCT	ROM	RAM	Pro EE	PWM	ACOMP	OPA	ADC	Touch	USART	IIC	LED	LCD	PACKAGE
SC8F096AD816SP	8K×16	336×8	256×16	5	2	1	12Bit×14	8	2	1	7com×6seg	5com×9seg	SOP16
SC8F096AD820NPR	8K×16	336×8	256×16	5	2	1	12Bit×18	11	2	1	9com×8seg	6com×12seg	QFN20
SC8F096AD824NPR	8K×16	336×8	256×16	5	2	1	12Bit×22	15	2	1	10com×9seg	6com×16seg	QFN24
SC8F096AD824SS	8K×16	336×8	256×16	5	2	1	12Bit×22	15	2	1	10com×9seg	6com×16seg	SSOP24
SC8F096AD828SS	8K×16	336×8	256×16	5	2	1	12Bit×26	15	2	1	10com×9seg	6com×20seg	SSOP28
SC8F096AD832NPR	8K×16	336×8	256×16	5	2	1	12Bit×30	15	2	1	10com×9seg	6com×24seg	QFN32
SC8F096AD832FP	8K×16	336×8	256×16	5	2	1	12Bit×30	15	2	1	10com×9seg	6com×24seg	LQFP32

1.3 系统结构框图

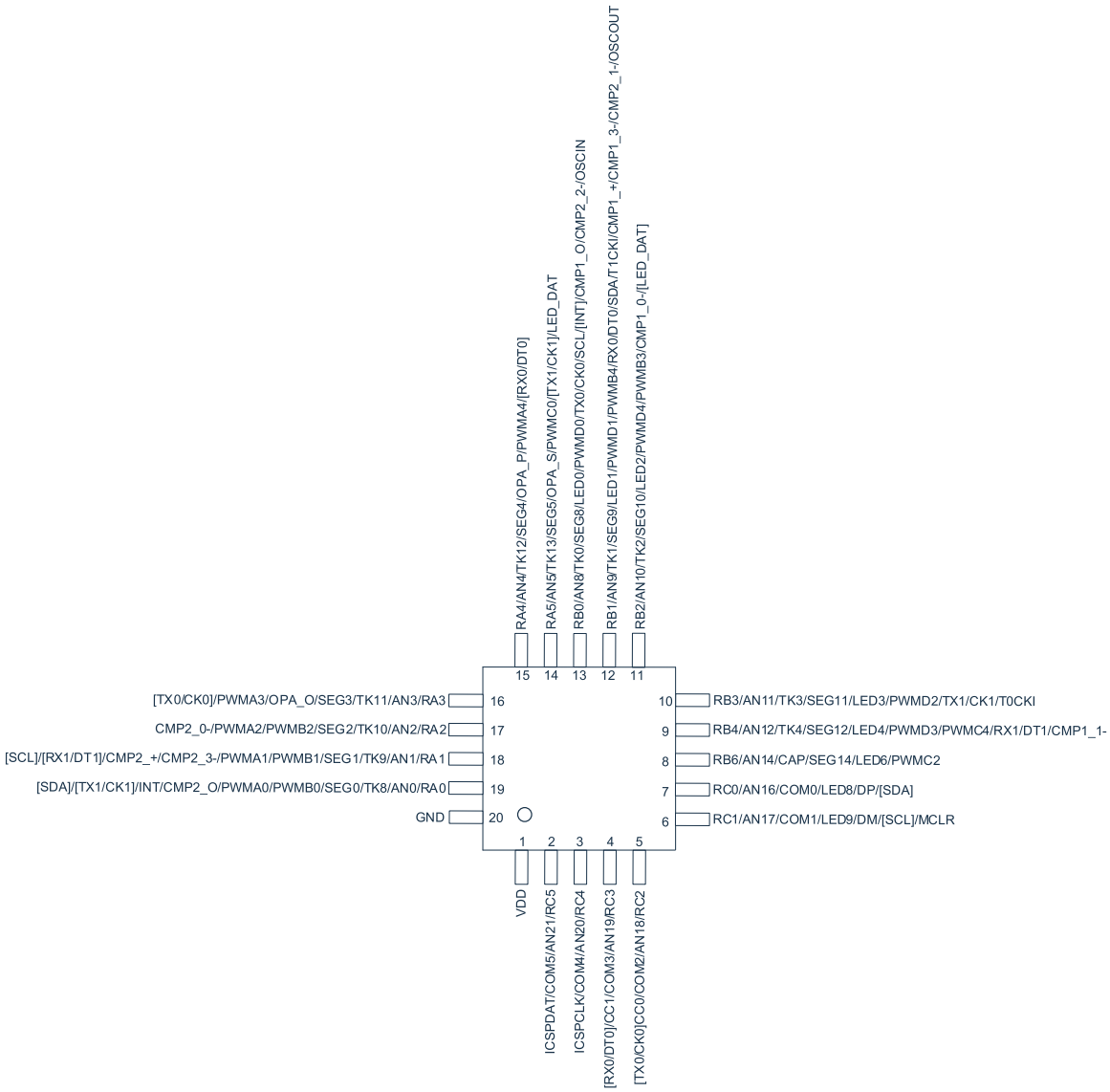

8192×16
Program Memory
PC
336×8
Data Memory
Stack1
..
Stack8
Addr Mux
Instruction Reg
Fsr Reg
Instruction Decode
and Control
Device Reset Timer
Power-on Reset
Watch Dog Timer
Timing Generation
OSC
VDD,GND
Mux
ACC
ALU
AD Converter
TIMER0
TIMER1
TIMER2
I/O PORT
PD
QC
LED/LCD
Pro EE
Touch
USART0/1
PWM
IIC
OPA
COMP1/2
RGB彩灯驱动

1.4 管脚分布

1.4.1 SC8F096AD816SP 管脚图

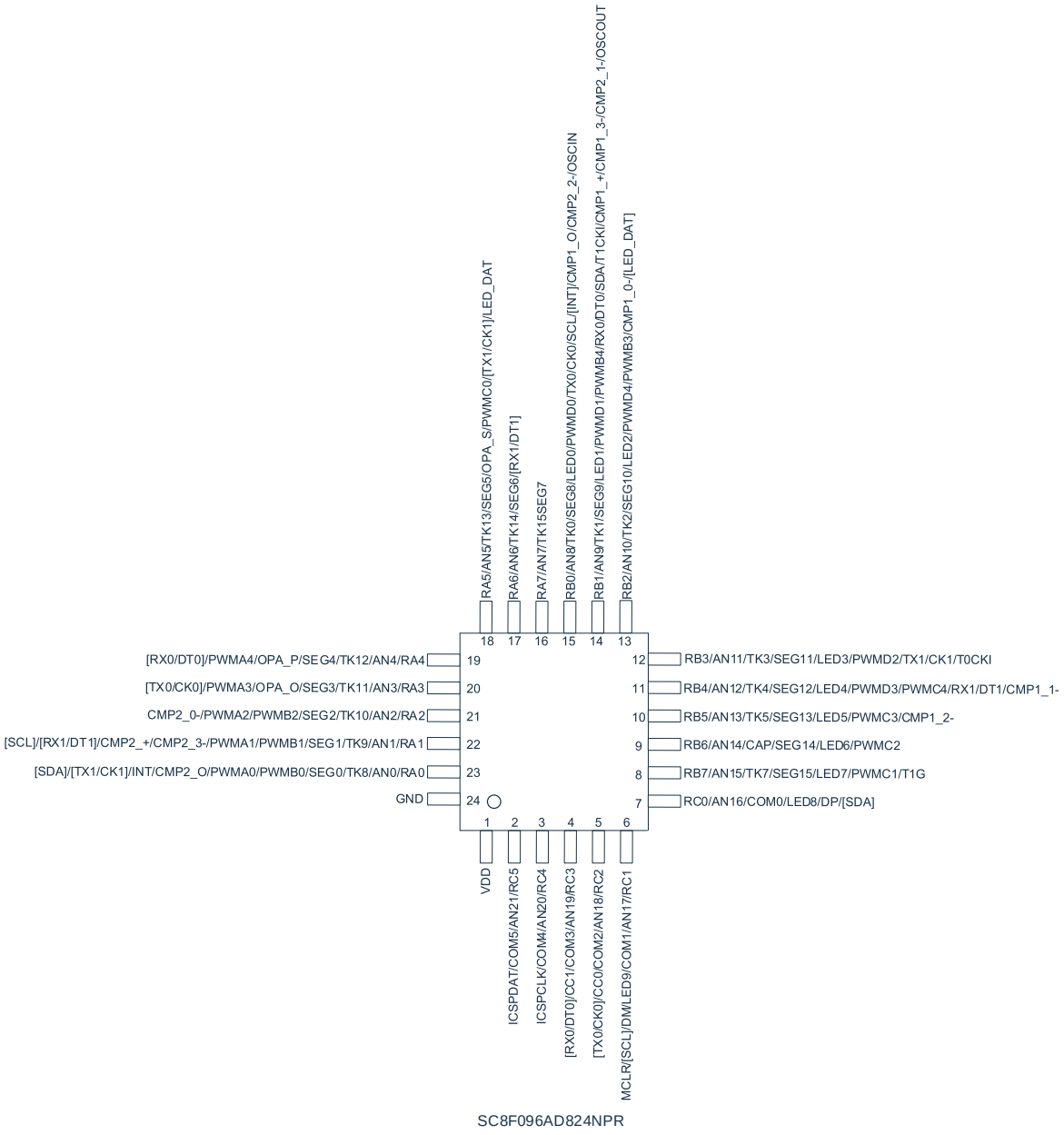


1.4.2 SC8F096AD820NPR 管脚图



SC8F096AD820NPR

1.4.3 SC8F096AD824NPR 管脚图

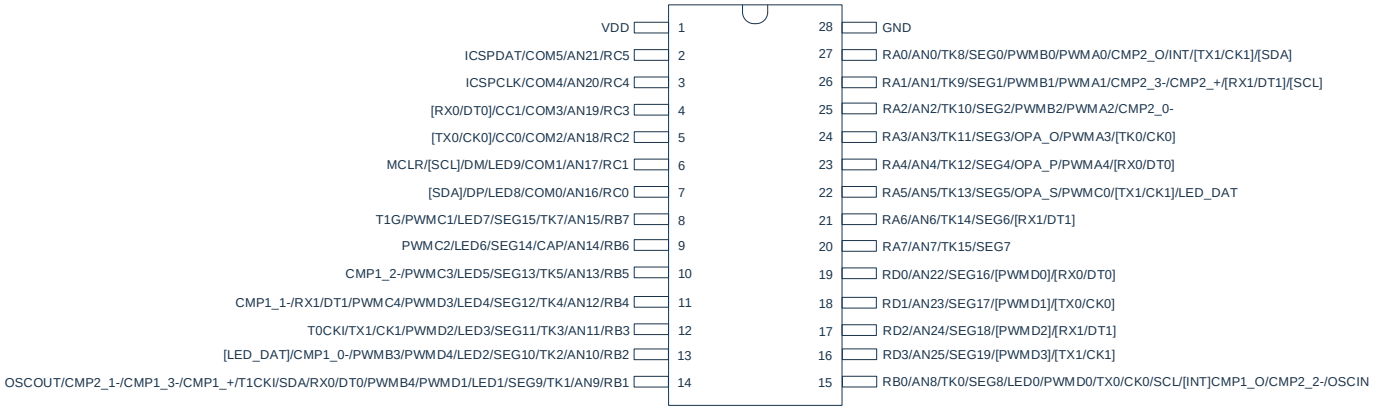


SC8F096AD824NPR

1.4.4 SC8F096AD824SS 管脚图

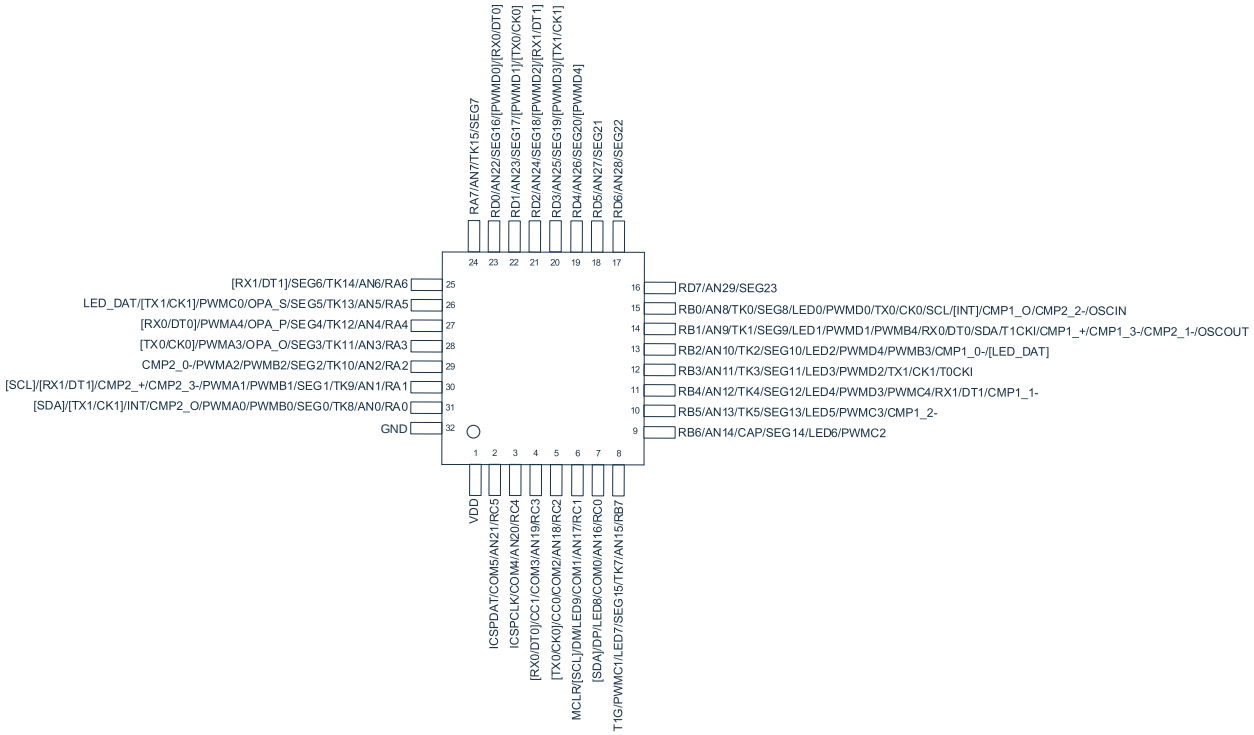
SC8F096AD824SS

1.4.5 SC8F096AD828SS 管脚图



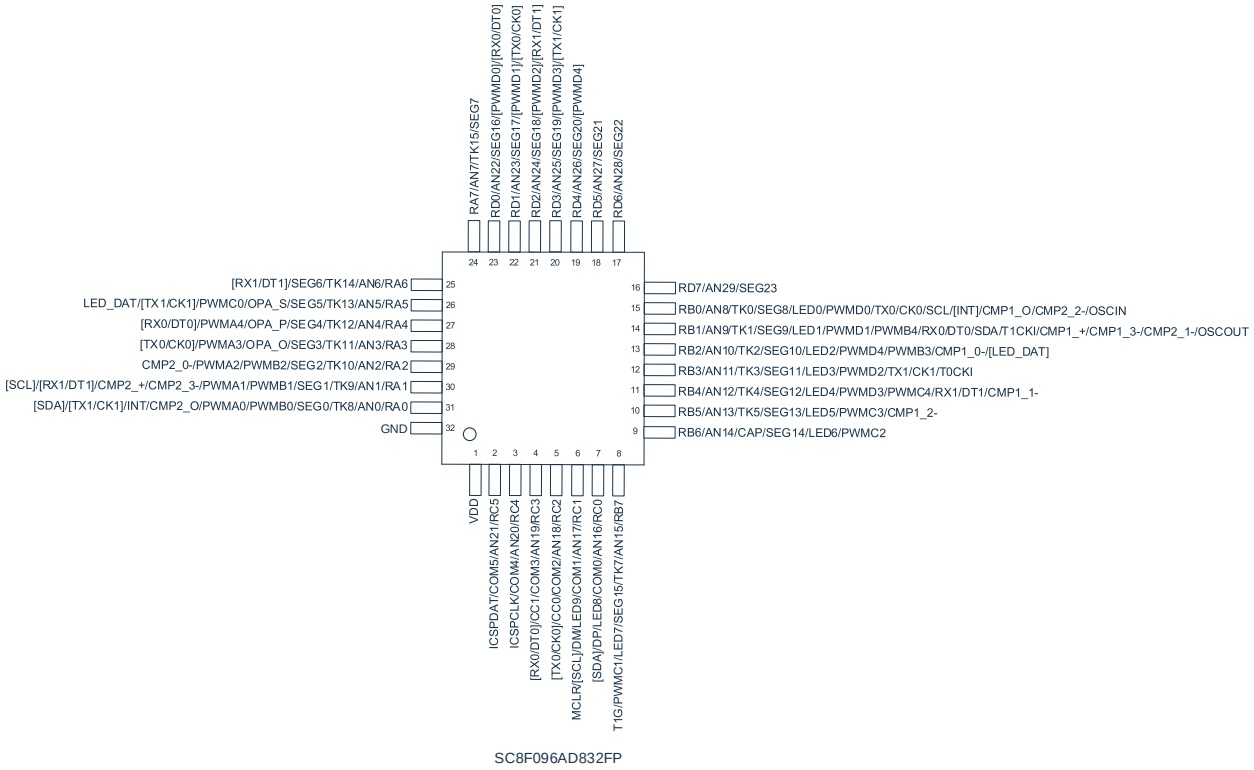
SC8F096AD828SS

1.4.6 SC8F096AD832NPR 管脚图



SC8F096AD832NPR

1.4.7 SC8F096AD832FP 管脚图



SC8F096AD832FP

注：
1) 串口 0、串口 1 的端口选择由 CONFIG 设置；
2) IIC 的端口选择由 CONFIG 设置；
3) 彩灯协议的发码端口选择由 CONFIG 设置；
4) PWMD 组端口选择由 CONFIG 配置。

SC8F096 引脚说明：

管脚名称	IO 类型	管脚说明
VDD,GND	P	电源电压输入脚，接地脚
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-RC5	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
RD0-RD7	I/O	可编程为输入脚，推挽输出脚，带上拉电阻功能
ICSPCLK/ICSPDAT	I/O	编程时钟/数据脚
PWMA0-PWMA4	O	PWMA 组输出脚
PWMB0-PWMB4	O	PWMB 组输出脚
PWMC0-PWMC4	O	PWMC 组输出脚
PWMD0-PWMD4	O	PWMD 组输出脚
AN0-AN29	I	12 位 ADC 输入脚
TK0-TK5，TK7~TK15	I	触摸按键输入脚
CAP	I	触摸电容输入脚
INT	I	外部中断输入脚
CMP1_+	I	比较器 1 正端输入脚
CMP2_+	I	比较器 2 正端输入脚
CMP1_0-， CMP1_1-，CMP1_2-，CMP1_3-	I	比较器 1 负端输入脚
CMP2_0-， CMP2_1-，CMP2_2-，CMP2_3-	I	比较器 2 负端输入脚
CMP1_O	O	比较器 1 结果输出脚
CMP2_O	O	比较器 2 结果输出脚
OPA_S	I	运放负端输入脚
OPA_P	I	运放正端输入脚
OPA_O	O	运放输出脚
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
TX0/CK0	I/O	USART0 异步发送输出/同步时钟输入/输出脚
RX0/DT0	I/O	USART0 异步接收输入脚/同步数据输入/输出脚
TX1/CK1	I/O	USART1 异步发送输出/同步时钟输入/输出脚
RX1/DT1	I/O	USART1 异步接收输入脚/同步数据输入/输出脚
SCL/SDA	I/O	IIC 时钟/数据脚
LED_DAT	O	LED 彩灯协议发码脚
CC0/CC1	I/O	PD 协议模块通讯脚
DM/DP	O	QC 协议模块驱动电压输出脚
COM0~COM5	O	LCD 模块 COM 口扫描输出脚
SEG0~SEG23	O	LCD 模块 SEG 口扫描输出脚
LED0~LED9	O	正反推 LED 模块扫描输出脚
OSCIN/OSCOUT	I/O	32.768K 晶振输入/输出脚
MCLR	I	外部复位输入脚

1.5 系统配置寄存器

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

1. WDT（看门狗选择）
   - ENABLE　　打开看门狗定时器
   - DISABLE　　关闭看门狗定时器
2. PROTECT（加密）
   - DISABLE　　ROM 代码不加密
   - ENABLE　　ROM 代码加密，加密后烧写仿真器读出来的值将不确定
3. LVR_SEL（低压侦测选择）
   - 1.8V
   - 2.0V
   - 2.5V
   - 3.0V
4. F_{CPU}_DIV（指令时钟分频）
   - 4T　　4 分频，$F_{CPU}=F_{SYS}/4$
   - 2T　　2 分频，$F_{CPU}=F_{SYS}/2$
5. WDT_DIV（WDT 预分频系数控制）
   - DISABLE　　通过 OPTION_REG 寄存器可选择 WDT 预分频从 1:128
   - ENABLE　　通过 OPTION_REG 寄存器可选择 WDT 预分频从 3:384
6. ICSP（仿真口功能选择）
   - 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
7. USART0_PORT_SEL（USART0 端口选择）
   - RB0/RB1　　选择 RB0 为 TX0 口，RB1 为 RX0 口
   - RC2/RC3　　选择 RC2 为 TX0 口，RC3 为 RX0 口
   - RD1/RD0　　选择 RD1 为 TX0 口，RD0 为 RX0 口
   - RA3/RA4　　选择 RA3 为 TX0 口，RA4 为 RX0 口
8. USART1_PORT_SEL（USART1 端口选择）
   - RB3/RB4　　选择 RB3 为 TX1 口，RB4 为 RX1 口
   - RA0/RA1　　选择 RA0 为 TX1 口，RA1 为 RX1 口
   - RD3/RD2　　选择 RD3 为 TX1 口，RD2 为 RX1 口
   - RA5/RA6　　选择 RA5 为 TX1 口，RA6 为 RX1 口
9. IIC_PORT_SEL（IIC 端口选择）
   - RB0/RB1　　选择 RB0 为 SCL 口，RB1 为 SDA 口
   - RA1/RA0　　选择 RA1 为 SCL 口，RA0 为 SDA 口
   - RC1/RC0　　选择 RC1 为 SCL 口，RC0 为 SDA 口
10. LED_DAT_PORT（LED 彩灯协议发码端口选择）
    - RA5　　选择 RA5 为 LED_DAT 口
    - RB2　　选择 RB2 为 LED_DAT 口
11. CMP_LV_SEL（比较器数字滤波时间选择）

- 1us~1.5us
- 关闭数字滤波

12. INT_PORT_SEL（INT 端口选择）
    - RA0　　选择 RA0 为 INT 口
    - RB0　　选择 RB0 为 INT 口
13. PWMD_PORT_SEL（PWMD 组端口选择）
    - PWMD0~PWMD4 分配在 RB0/RB1/RB3/RB4/RB2
    - PWMD0~PWMD4 分配在 RD0/RD1/RD2/RD3/RD4
14. EXT_RESET（外部复位口功能选择）
    - DISABLE　　禁止外部复位功能，RC1 为普通 IO 口
    - ENABLE　　使能外部复位功能，RC1 为外部复位口
15. LED_PORT_ISEL<9:0>（LED 口的低电平驱动能力选择位）
    - 0x3FF　　LED9~LED0 口的低电平驱动选择为 40mA
    - 0x000　　LED9~LED0 口的低电平驱动选择为 125mA

注：LED_PORT_ISEL<9:0>的 10 个 Bit 分别对应 LED9~LED0，每个 Bit 都可以单独配置为 0 或 1，配置为 0 时对应的 LED 口低电平驱动选择 125mA，配置为 1 时对应的 LED 口低电平驱动选择 40mA。

1.6 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

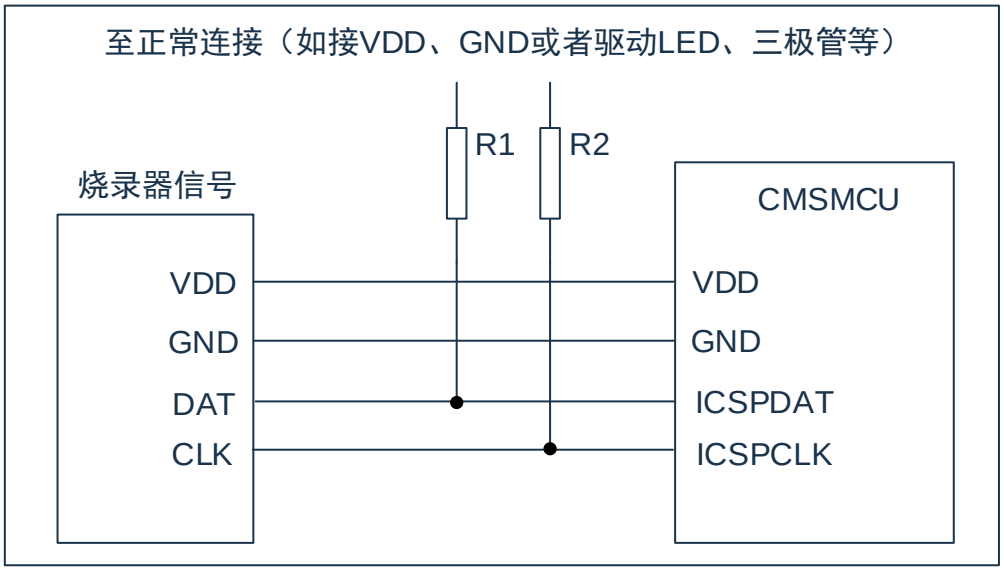


图 1-1：典型的在线串行编程连接方式

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。

1.7 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

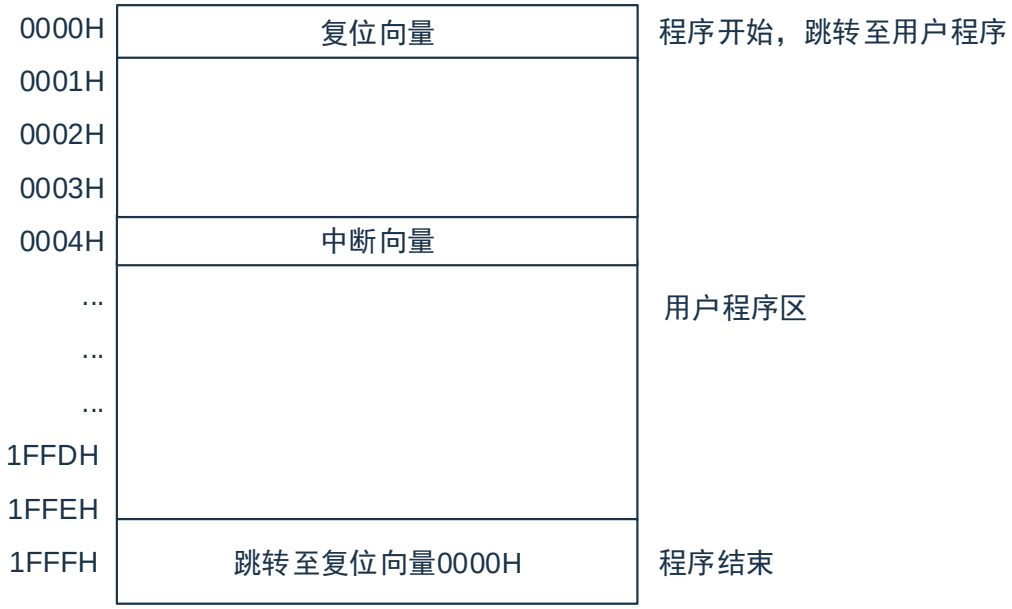
2. 中央处理器（CPU）

2.1 内存

2.1.1 程序内存

SC8F096 程序存储器空间

MTP：8K



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）。具有以下三种复位方式：

- 上电复位
- 看门狗复位
- 低压复位（LVR）
- 外部复位

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 MTP 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
```

注：由于单片机并未提供专门的出栈、压栈指令，故用户需自己保护中断现场。

例：中断入口保护现场

```
PUSH:
        LD      ACC_BAK,A       ;保存 ACC 至自定义寄存器 ACC_BAK
        SWAPA   STATUS          ;状态寄存器 STATUS 高低半字节互换
        LD      STATUS_BAK,A    ;保存至自定义寄存器 STATUS_BAK
        RET                     ;返回
```

例：中断出口恢复现场

```
POP:
        SWAPA   STATUS_BAK      ;将保存至 STATUS_BAK 的数据高低半字节互换给 ACC
        LD      STATUS,A        ;将 ACC 的值给状态寄存器 STATUS
        SWAPR   ACC_BAK         ;将保存至 ACC_BAK 的数据高低半字节互换
        SWAPA   ACC_BAK         ;将保存至 ACC_BAK 的数据高低半字节互换给 ACC
        RET                     ;返回
```

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

```
MTP 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

```
MTP 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 MTP 空间的分页处。

2.1.2 数据存储器

SC8F096 数据存储器列表

	地址		地址		地址		地址
INDF	00H	INDF	80H	INDF	100H	INDF	180H
OPTION_REG	01H	TMR0	81H	OPTION_REG	101H	SEGEN2	181H
PCL	02H	PCL	82H	PCL	102H	PCL	182H
STATUS	03H	STATUS	83H	STATUS	103H	STATUS	183H
FSR	04H	FSR	84H	FSR	104H	FSR	184H
TRISB	05H	TRISA	85H	TRISC	105H	TXSTA1	185H
PORTB	06H	PORTA	86H	PORTC	106H	RCSTA1	186H
WPDB	07H	WPDA	87H	PORTD	107H	TXREG1	187H
WPUB	08H	WPUA	88H	WPUC	108H	RCREG1	188H
IOCB	09H	IOCA	89H	ANSEL2	109H	SPBRG1	189H
PCLATH	0AH	PCLATH	8AH	PCLATH	10AH	PCLATH	18AH
INTCON	0BH	INTCON	8BH	INTCON	10BH	INTCON	18BH
QCCON	0CH	ANSEL3	8CH	TMR1L	10CH	OPACON	18CH
PIR1	0DH	EECON1	8DH	TMR1H	10DH	OPAADJ	18DH
PIE1	0EH	EECON2	8EH	T1CON	10EH	LEDCTR0	18EH
CMP1CON0	0FH	EEDAT	8FH	PIR2	10FH	LEDCTR1	18FH
CMP1CON1	10H	EEDATH	90H	PIE2	110H	IICCON	190H
PR2	11H	EEADR	91H	KEYCON0	111H	IICCON2	191H
TMR2	12H	EEADRH	92H	KEYCON1	112H	IICSTAT	192H
T2CON	13H	ANSEL0	93H	KEYCON2	113H	IICBUF	193H
OSCCON	14H	ANSEL1	94H	TRISD	114H	IICADD	194H
PWMCON0	15H	ADCON0	95H	WPUD	115H	CC0CON	195H
PWMCON1	16H	ADCON1	96H	KEYDATAL	116H	CC1CON	196H
PWMTL	17H	-----	97H	KEYDATAH	117H	PDCON0	197H
PWMTH	18H	ADRESL	98H	TXSTA0	118H	PDCON1	198H
PWMD0L	19H	ADRESH	99H	RCSTA0	119H	PDADD	199H
PWMD1L	1AH	CMP2CON0	9AH	SPBRG0	11AH	PDRDATA	19AH
PWMD4L	1BH	PWMD2L	9BH	TXREG0	11BH	PDSDATA	19BH
PWMT4L	1CH	PWMD3L	9CH	RCREG0	11CH	LCDADD	19CH
PWMCON2	1DH	PWM23DT	9DH	LCDCON0	11DH	LCDDATA	19DH
PWMDH	1EH	SEGEN0	9EH	LCDCON1	11EH	COMEN	19EH
PWM01DT	1FH	CMP2CON1	9FH	SEGEN1	11FH	SEGEN3	19FH
通用寄存器 96 字节	20H 7FH	通用寄存器 80 字节 快速存储区 70H-7FH	A0H EFH F0H -- FFH	通用寄存器 80 字节 快速存储区 70H-7FH	120H 16FH 170H -- 17FH	通用寄存器 80 字节 快速存储区 70H-7FH	1A0H 1EFH 1F0H -- 1FFH
BANK0		BANK1		BANK2		BANK3	

数据存储器分为两个功能区间：特殊功能寄存器和通用数据存储器。数据存储器单元大多数是可读/写的，但有些只读的。特殊功能寄存器地址为从 00H~1FH，80H~9FH，100H~11FH，180H~19FH。

SC8F096 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	INDF	寻址该单元会使用FSR的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
01H	OPTION_REG	T0LSE_EN	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	01111011
02H	PCL	程序计数器低字节								00000000
03H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
04H	FSR	间接数据存储器地址指针								xxxxxxxx
05H	TRISB	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0	11111111
06H	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxxxxxx
07H	WPDB	WPDB7	WPDB6	WPDB5	WPDB4	WPDB3	WPDB2	WPDB1	WPDB0	00000000
08H	WPUB	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0	00000000
09H	IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0	00000000
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	QCCON	----	----	DPDAT	DMDAT	DPSEL	DPEN	DMSEL	DMEN	--000000
0DH	PIR1	RC0IF	TX0IF	CMP1IF	PWMIF	RAIF	TMR1IF	TMR2IF	ADIF	--000-00
0EH	PIE1	RC0IE	TX0IE	CMP1IE	PWMIE	RAIE	TMR1IE	TMR2IE	ADIE	--000-00
0FH	CMP1CON0	CMP1EN	CMP1PS	CMP1NS2	CMP1NS1	CMP1NS0	CMP1NV	CMP1OUT	CMP1OEN	00000000
10H	CMP1CON1	CMP1IM	AN1_EN	RBIAS1_H	RBIAS1_L	LVDS1<3:0>				00000000
11H	PR2	TIMER2周期寄存器								11111111
12H	TMR2	TIMER2模块寄存器								00000000
13H	T2CON	CLK_SEL	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	00000000
14H	OSCCON	----	IRCF2	IRCF1	IRCF0	----	----	SWDTEN	-----	-101—1-
15H	PWMCON0	CLKDIV<2:0>			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN	00000000
16H	PWMCON1	PWMIO_SEL<1:0>		PWM2DTEN	PWM0DTEN	----	----	DT_DIV<1:0>		0000—00
17H	PWMTL	PWM0~PWM1周期低8位寄存器								00000000
18H	PWMTH	----	----	PWM4D<9:8>		PWM4T<9:8>		PWMT<9:8>		--000000
19H	PWMD0L	PWM0占空比低8位								00000000
1AH	PWMD1L	PWM1占空比低8位								00000000
1BH	PWMD4L	PWM4占空比低8位								00000000
1CH	PWMT4L	PWM4周期低8位寄存器								00000000
1DH	PWMCON2	----	----	----	PWM4DIR	PWM3DIR	PWM2DIR	PWM1DIR	PWM0DIR	---00000
1EH	PWMDH	PWMD3<9:8>		PWMD2<9:8>		PWMD1<9:8>		PWMD0<9:8>		00000000
1FH	PWM01DT	----	----	PWM01DT<5:0>						--000000

SC8F096 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
80H	INDF	寻址该单元会使用FSR的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
81H	TMR0	TIMER0数据寄存器								xxxxxxxx
82H	PCL	程序计数器低字节								00000000
83H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
84H	FSR	间接数据存储器地址指针								xxxxxxxx
85H	TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0	11111111
86H	PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0	xxxxxxxx
87H	WPDA	WPDA7	WPDA6	WPDA5	WPDA4	WPDA3	WPDA2	WPDA1	WPDA0	00000000
88H	WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0	00000000
89H	IOCA	IOCA7	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0	00000000
8AH	PCLATH	----	----	----	程序计数器高5位的写缓冲器					---00000
8BH	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	00000000
8CH	ANSEL3	ANS29	ANS28	ANS27	ANS26	ANS25	ANS24	ANS23	ANS22	00000000
8DH	EECON1	EEPGD	----	----	----	WRERR	WREN	WR	RD	0---0000
8EH	EECON2	EEPROM控制寄存器2（不是物理寄存器）								--------
8FH	EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0	xxxxxxxx
90H	EEDATH	EEDATH7	EEDATH6	EEDATH5	EEDATH4	EEDATH3	EEDATH2	EEDATH1	EEDATH0	xxxxxxxx
91H	EEADR	EEADR7	EEADR6	EEADR5	EEADR4	EEADR3	EEADR2	EEADR1	EEADR0	00000000
92H	EEADRH	----	----	----	EEADRH4	EEADRH3	EEADRH2	EEADRH1	EEADRH0	---00000
93H	ANSEL0	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0	00000000
94H	ANSEL1	ANS15	ANS14	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8	00000000
95H	ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/ $\overline{\text{DONE}}$	ADON	00000000
96H	ADCON1	ADFM	CHS4	----	----	----	LDO_EN	LDO_SEL1	LDO_SEL0	00---000
98H	ADRESL	A/D结果寄存器的低字节								xxxxxxxx
99H	ADRESH	A/D结果寄存器的高字节								xxxxxxxx
9AH	CMP2CON0	CMP2EN	CMP2PS	CMP2NS2	CMP2NS1	CMP2NS0	CMP2NV	CMP2OUT	CMP2OEN	00000000
9BH	PWMD2L	PWM2占空比低8位								00000000
9CH	PWMD3L	PWM3占空比低8位								00000000
9DH	PWM23DT	----	----	PWM23死区延时时间						--000000
9EH	SEGEN0	SEG7EN	SEG6EN	SEG5EN	SEG4EN	SEG3EN	SEG2EN	SEG1EN	SEG0EN	00000000
9FH	CMP2CON1	CMP2IM	AN2_EN	RBIAS2_H	RBIAS2_L	LVDS2<3:0>				00000000

SC8F096 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	INDF	寻址该单元会使用FSR的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
101H	OPTION_REG	T0LSE_EN	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	01111011
102H	PCL	程序计数器低字节								00000000
103H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
104H	FSR	间接数据存储器地址指针								xxxxxxxx
105H	TRISC	----	----	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	--111111
106H	PORTC	----	----	RC5	RC4	RC3	RC2	RC1	RC0	--xxxxx
107H	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxxxxxx
108H	WPUC	----	----	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0	--000000
109H	ANSEL2	----	----	ANS21	ANS20	ANS19	ANS18	ANS17	ANS16	--00000
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	TMR1L	16位TIMER1寄存器低字节的数据寄存器								xxxxxxxx
10DH	TMR1H	16位TIMER1寄存器高字节的数据寄存器								xxxxxxxx
10EH	T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	00000000
10FH	PIR2	BCLIF	TKIF	CMP2IF	IICIF	PDSIF	PDRIF	TX1IF	RC1IF	00000000
110H	PIE2	BCLIE	TKIE	CMP2IE	IICIE	PDSIE	PDRIE	TX1IE	RC1IE	00000000
111H	KEYCON0	KDONE	----	----	----	----	KTOUT	KCAP	KEN	0----000
112H	KEYCON1	KVREF<1:0>		KCLK<1:0>		KCHS<3:0>				00000000
113H	KEYCON2	CAP_LVBO<2:0>			TP_EN	TKLDOEN	----	TKLDO_SEL	TKEN	00000-00
114H	TRISD	TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0	11111111
115H	WPUD	WPUD7	WPUD6	WPUD5	WPUD4	WPUD3	WPUD2	WPUD1	WPUD0	00000000
116H	KEYDATAL	触摸结果寄存器低8位								00000000
117H	KEYDATAH	触摸结果寄存器高8位								00000000
118H	TXSTA0	CSRC0	TX9EN0	TXEN0	SYNC0	SCKP0	----	TRMT0	TX9D0	00000-10
119H	RCSTA0	SPEN0	RX9EN0	SREN0	CREN0	RCIDL0	FERR0	OERR0	RX9D0	00001000
11AH	SPBRG0	USART0波特率数据寄存器								00000000
11BH	TXREG0	USART0发送数据寄存器								00000000
11CH	RCREG0	USART0接收数据寄存器								00000000
11DH	LCDCON0	LCDEN	LEDEN	COMSEL<2:0>			LCDCLK<2:0>			00000000
11EH	LCDCON1	-----	BIAS	-----	-----	FCCTLM<1:0>		LCDF<1:0>		-0--0000
11FH	SEGEN1	SEG15EN	SEG14EN	SEG13EN	SEG12EN	SEG11EN	SEG10EN	SEG9EN	SEG8EN	00000000

SC8F096 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	INDF	寻址该单元会使用FSR的内容寻址数据存储器（不是物理寄存器）								xxxxxxxx
181H	SEGEN2	SEG23EN	SEG22EN	SEG21EN	SEG20EN	SEG19EN	SEG18EN	SEG17EN	SEG16EN	00000000
182H	PCL	程序计数器低字节								00000000
183H	STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C	00011xxx
184H	FSR	间接数据存储器地址指针								xxxxxxxx
185H	TXSTA1	CSRC1	TX9EN1	TXEN1	SYNC1	SCKP1	----	TRMT1	TX9D1	00000-10
186H	RCSTA1	SPEN1	RX9EN1	SREN1	CREN1	RCIDL1	FERR1	OERR1	RX9D1	00001000
187H	TXREG1	USART1发送数据寄存器								00000000
188H	RCREG1	USART1接收数据寄存器								00000000
189H	SPBRG1	USART1波特率数据寄存器								00000000
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	OPACON	OPA_EN	OPA_OEN	-----	OPA_ADC	----	-----	-----	OPA_FT	00-0---0
18DH	OPAADJ	OPADOUT	OPACOFM	OPACRS	OPAADJ<4:0>					000xxxxx
18EH	LEDCTR0	LEDSW	LEDSW_RT	IF_15	IF_30	RESET_T	LEDSW_CKSEL<1:0>		LEDSW_ST	00000000
18FH	LEDCTR1	LEDSW_ADR	-----	-----	LEDSW_ADD<4:0>					0--000000
190H	IICCON	IICWCOL	IICOV	IICEN	IICCKP	IICTOS<1:0>		IICM1	IICM0	00000000
191H	IICCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	00000000
192H	IICSTAT	-----	IDLE	D/A	P	S	R/W	IICTOF	BF	-0000000
193H	IICBUF	IICBUF<7:0>								00000000
194H	IICADD	IICADD<7:0>								00000000
195H	CC0CON	-----	-----	CC0_EN	CC0_HYS	CC0_CVS<1:0>		CC0_RD	CC0_CMPO	--00001x
196H	CC1CON	-----	-----	CC1_EN	CC1_HYS	CC1_CVS<1:0>		CC1_RD	CC1_CMPO	--00001x
197H	PDCON0	PDEN	PD_BUF	SEND_ST	CRC_RES	CC_SEL	PD_MODE	-----	REC_S	000000-1
198H	PDCON1	-----	PDSEN	SEND_SEL	PDCNT<4:0>					-0000000
199H	PDADD	PDSADD<2:0>			PDRADD<4:0>					00000000
19AH	PDRDATA	PDRDATA<7:0>								xxxxxxxx
19BH	PDSDATA	PDSDATA<7:0>								xxxxxxxx
19CH	LCDADD	LCDCS	B2RES<1:0>		LCDADD<4:0>					00000000
19DH	LCDDATA	LCDDATA<7:0>								xxxxxxxx
19EH	COMEN	-----	------	COM5EN	COM4EN	COM3EN	COM2EN	COM1EN	COM0EN	00000000
19FH	SEGEN3	SEGDRI<4:0>					-----	-----	LCDDATA<8>	00000--x

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
```

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 INDF 寄存器可间接寻址，INDF 不是物理寄存器。当对 INDF 进行存取时，它会把 FSR 寄存器内的值作为地址，并指向该地址的寄存器，因此在设置了 FSR 寄存器后，就可把 INDF 寄存器当作目的寄存器来存取。间接读取 INDF（FSR=0）将产生 00H。间接写入 INDF 寄存器，将导致一个空操作。以下例子说明了程序中间接寻址的用法。

例：FSR 及 INDF 的应用

```
        LDIA    30H
        LD      FSR,A       ;间接寻址指针指向 30H
        CLR     INDF        ;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)举例：

```
        LDIA    1FH
        LD      FSR,A       ;间接寻址指针指向 1FH
LOOP:
        INCR    FSR         ;地址加 1，初始地址为 20H
        CLR     INDF        ;清零 FSR 所指向的地址
        LDIA    7FH
        SUBA    FSR
        SNZB    STATUS,C    ;一直清零至 FSR 地址为 7FH
        JP      LOOP
```

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，且既不能被读出，也不能被写入。对它的操作通过堆栈指针（SP）来实现，堆栈指针（SP）也不能读出或写入，当系统复位后堆栈指针会指向堆栈顶部。当发生子程序调用及中断时的程序计数器（PC）值被压入堆栈缓存器，当从中断或子程序返回时将数值返回给程序计数器（PC），下图说明其工作原理。

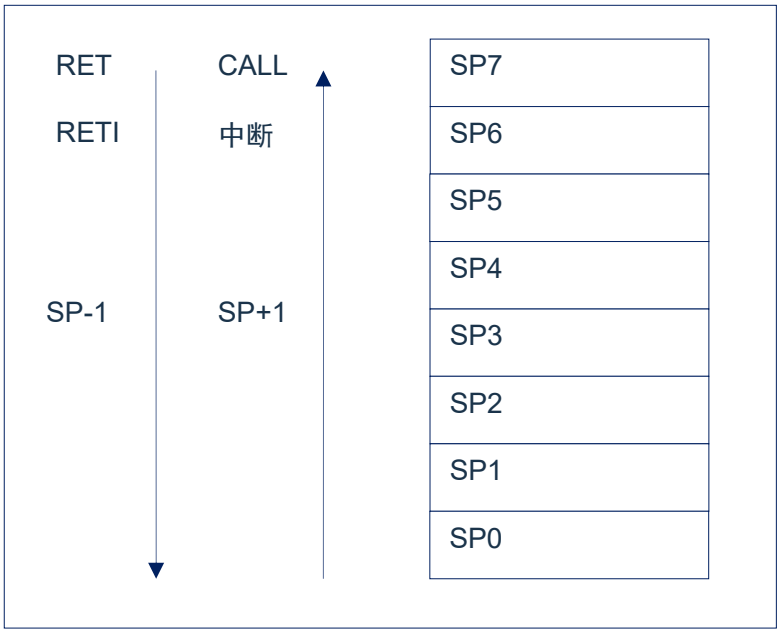


图 2-2：堆栈缓存器工作原理

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 8Bit 的寄存器，ALU 的运算结果可以存放在此，它并不属于数据存储器的一部分而是位于 CPU 中供 ALU 在运算中使用，因此不能被寻址，只能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 000uu1uu（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	IRP	RP1	RP0	TO	PD	Z	DC	C
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	1	1	X	X	X

Bit7 IRP: 寄存器存储器选择位(用与间接寻址)
1= Bank2和Bank3（100h~1FFh）
0= Bank0和Bank1（00h~FFh）

Bit6~Bit5 RP<1:0>: 存储区选择位
00= 选择Bank0
01= 选择Bank1
10= 选择Bank2
11= 选择Bank3

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= 发生了结果的低4位向高位进位或低4位没有发生借位
0= 结果的低4位没有向高位进位或低4位向高位借位

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	休眠态 WDT 溢出
0	1	非休眠态 WDT 溢出
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，包括各种控制位用于配置：

- WDT 预分频器。
- 外部中断触发边沿

预分频器控制寄存器 OPTION_REG（01H）

01H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	T0LSE_EN	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	1	1	1	1	0	1	1

Bit7 T0LSE_EN: TIMER0 时钟源选择 F_{LSE} 使能位
0= TIMER0 时钟源由 T0CS 决定
1= TIMER0 时钟源选择 F_{LSE}
Bit6 INTEDG: 触发中断的边沿选择位
0= INT 引脚下降沿触发中断
1= INT 引脚上升沿触发中断
Bit5 T0CS: TIMER0 时钟源选择位
0= 内部指令周期时钟（F_{CPU}）
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 (WDT_DIV=DISABLE)	WDT 分频比 (WDT_DIV=ENABLE)
0	0	0	1:2	1:1	1:3
0	0	1	1:4	1:2	1:6
0	1	0	1:8	1:4	1:12
0	1	1	1:16	1:8	1:24
1	0	0	1:32	1:16	1:48
1	0	1	1:64	1:32	1:96
1	1	0	1:128	1:64	1:192
1	1	1	1:256	1:128	1:384

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。

2.7 程序计数器（PC）

程序计数器（PC）控制程序内存 MTP 中的指令执行顺序，它可以寻址整个 MTP 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 13Bit 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 8Kx16 位程序地址。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）；
RET、RETI、RETI 时	PC=堆栈出来的值；
操作 PCL 时	PC[12:8]不变，PC[7:0]=用户指定的值；
JP 时	PC=程序指定的值；
其它指令	PC=PC+1；

2.8 看门狗计数器（WDT）

看门狗定时器（WatchDogTimer）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 使用 8 位预分频器。在所有复位后，WDT 溢出周期为 128ms，WDT 溢出周期计算方式为 16ms*分频系数，若需要改变 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。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。否者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. 看门狗计数器不同芯片的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 与看门狗控制相关的寄存器

振荡控制寄存器 OSCCON (14H)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	SWDTEN	---
R/W	---	R/W	R/W	R/W	---	---	R/W	---
复位值	---	1	0	1	---	---	1	---

Bit7 未用。

Bit6~Bit4 IRCF<2:0>: 内部振荡器频率选择位

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

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

101= $F_{SYS} = F_{HSI}/4$ (默认)

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

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

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

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

000= F_{SYS} = 32KHz(F_{LSI})

Bit3~Bit2 未用

Bit1 SWDTEN： 软件使能或禁止看门狗定时器位

1= 使能 WDT

0= 不使能 WDT

Bit0 未用

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

3. 系统时钟

3.1 概述

时钟信号由振荡器产生，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。

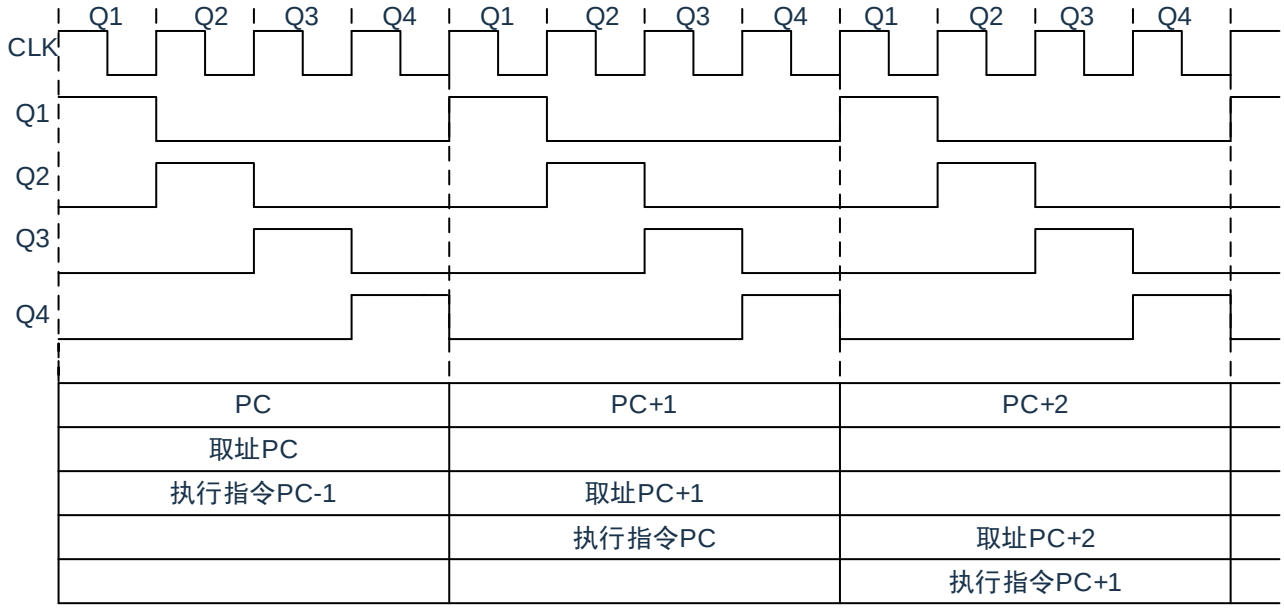
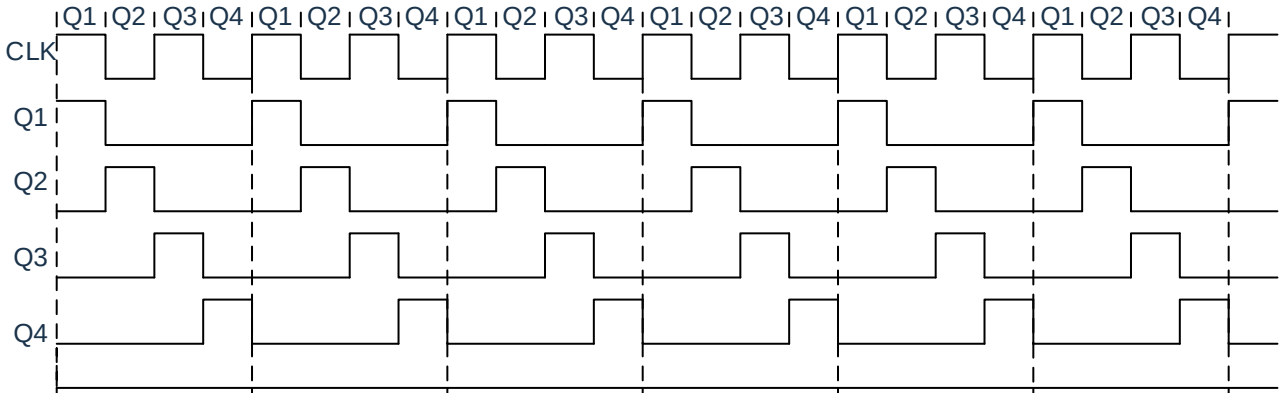


图 3-1：时钟与指令周期时序图(F_{CPU}_DIV=4T)

下面列出 F_{CPU}_DIV=4T 时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
1MHz	8μs	4μs
2MHz	4μs	2μs
4MHz	2μs	1μs
8MHz	1μs	500ns
16MHz	500ns	250ns



PC	PC+1	PC+2	PC+3	PC+4	PC+5	
取址PC						
执行指令PC-1	取址PC+1					
	执行指令PC	取址PC+2				
		执行指令PC+1	取址PC+3			
			执行指令PC+2	取址PC+4		
				执行指令PC+3	取址PC+5	
					执行指令PC+4	

图 3-2：时钟与指令周期时序图(F_{CPU}_DIV=2T)

下面列出 F_{CPU}_DIV=2T 时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
1MHz	4μs	2μs
2MHz	2μs	1μs
4MHz	1μs	500ns
8MHz	500ns	250ns
16MHz	250ns	125ns

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，振荡频率固定为 16MHz，在这个基础上可通过 OSCCON 寄存器设置芯片工作频率。

3.3 起振时间

起振时间（ResetTime）是指从芯片复位到芯片振荡稳定这段时间，其设计值为 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。

振荡控制寄存器 OSCCON (14H)

14H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCCON	---	IRCF2	IRCF1	IRCF0	---	---	SWDTEN	---
R/W	---	R/W	R/W	R/W	---	---	R/W	---
复位值	---	1	0	1	---	---	1	---

Bit7	未用	
Bit6~Bit4	IRCF<2:0>:	内部振荡器频率选择位
	111=	$F_{SYS} = F_{HSI}/1$
	110=	$F_{SYS} = F_{HSI}/2$
	101=	$F_{SYS} = F_{HSI}/4$ (默认)
	100=	$F_{SYS} = F_{HSI}/8$
	011=	$F_{SYS} = F_{HSI}/16$
	010=	$F_{SYS} = F_{HSI}/32$
	001=	$F_{SYS} = F_{HSI}/64$
	000=	F_{SYS} = 32KHz(LFINTOSC)
Bit3~Bit2	未用。	
Bit1	SWDTEN：	软件使能或禁止看门狗定时器位
	1=	使能 WDT
	0=	不使能 WD
Bit0	未用	

3.5 时钟框图

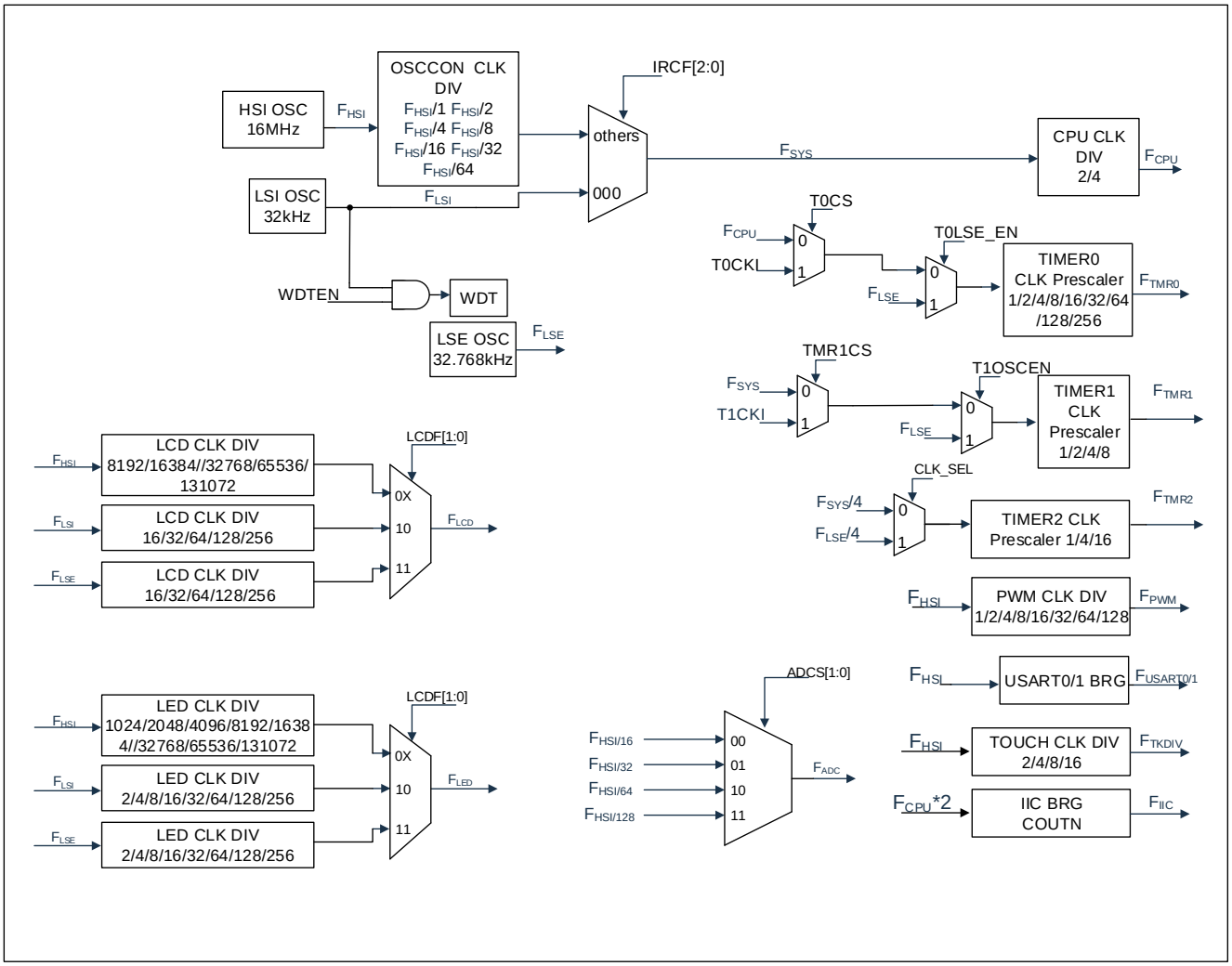


图 3-3：时钟框图

4. 复位

芯片可用如下 4 种复位方式：

- 上电复位
- 外部复位
- 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 外部复位

SC8F096 支持外部复位功能，可以通过 CONFIG 配置 RC1 作为复位口，此时 RC1 自动使能内部弱上拉。若 RC1 被拉低，芯片将会发生复位。

4.3 掉电复位

4.3.1 掉电复位概述

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。

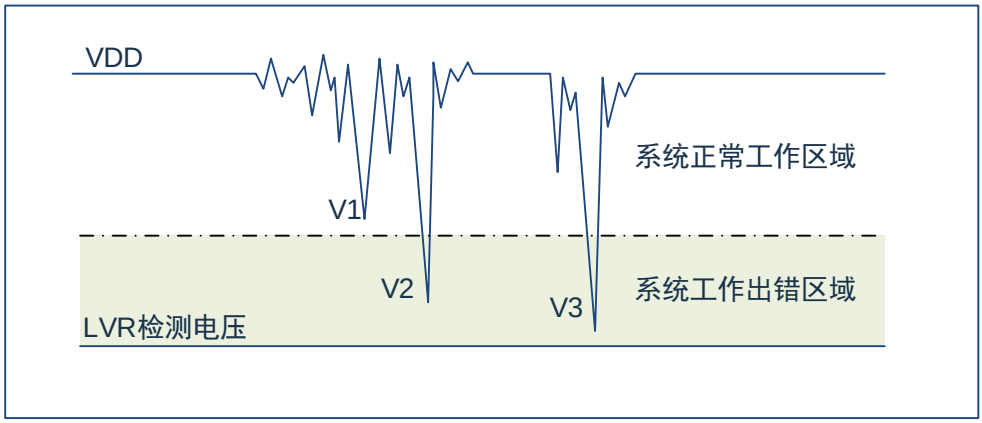


图 4-1：掉电复位示意图

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，系统上电、掉电时间都较长。其中，上电时序保护使得系统正常上电，但掉电过程却和 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。

4.3.2 掉电复位的改进办法

如何改进系统掉电复位性能，以下给出几点建议：

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

看门狗定时器

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

降低系统的工作速度

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

增大电压下降斜率

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

4.4 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；
2. INT 中断；
3. PORTB 电平变化中断；
4. PORTA 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，确保每一个输入口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 PWM 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      TRISB,A     ;所有 I/O 设置为输出口
        …                   ;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
        NOP
        NOP
```

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}）	$F_{SYS}=F_{HSI}$	$T_{WAIT}=264*1/F_{HSI}+16*1/F_{HSI}$
	$F_{SYS}= F_{HSI} /2$	$T_{WAIT}=264*2/F_{HSI}+16*1/F_{HSI}$
	…	…
	$F_{SYS}= F_{HSI} /64$	$T_{WAIT}=264*64/F_{HSI}+16*1/F_{HSI}$
内部低速 RC 振荡（$F_{LFINTOSC}$）	----	$T_{WAIT}=11/F_{LSI}$

6. I/O 端口

芯片有 4 个 I/O 端口：PORTA、PORTB、PORTC、PORTD（最多 30 个 I/O）。可读写端口数据寄存器可直接存取这些端口。

端口	位	管脚描述	I/O
PORTA	0	施密特触发输入，推挽输出，AN0，TK8，SEG0，TX1/CK1，SDA，PWMA0，PWMB0，INT，CMP2_O	I/O
	1	施密特触发输入，推挽输出，AN1，TK9，SEG1，RX1/DT1，SCL，PWMA1，PWMB1，CMP2_+，CMP2_3-	I/O
	2	施密特触发输入，推挽输出，AN2，TK10，SEG2，PWMA2，PWMB2，PWMD2，CMP2_0-	I/O
	3	施密特触发输入，推挽输出，AN3，TK11，SEG3，TX0/CK0，OPA_O，PWMA3	I/O
	4	施密特触发输入，推挽输出，AN4，TK12，SEG4，RX0/DT0，OPA_P，PWMA4，RX/DT	I/O
	5	施密特触发输入，推挽输出，AN5，TK13，SEG5，TX1/CK1，LED_DAT，OPA_S，PWMC0	I/O
	6	施密特触发输入，推挽输出，AN6，TK14，SEG6，RX1/DT1	I/O
	7	施密特触发输入，推挽输出，AN7，TK15，SEG7，	I/O
PORTB	0	施密特触发输入，推挽输出， AN8，TK0， SEG8，LED0，TX0/CK0，SCL，PWMD0， INT，CMP2_2-，CMP1_O，OSCIN	I/O
	1	施密特触发输入，推挽输出， AN9，TK1，SEG9，LED1，RX0/DT0，SDA， PWMB4，PWMD1，T1CKI，CMP2_1-，CMP1_3-，CMP1_+，OSCOUT	I/O
	2	施密特触发输入，推挽输出，AN10，TK2，SEG10，LED2，LED_DAT，PWMB3，PWMD4，CMP1_0-	I/O
	3	施密特触发输入，推挽输出，AN11，TK3，SEG11，LED3，TX1/CK1，PWMD2，T0CKI	I/O
	4	施密特触发输入，推挽输出，AN12，TK4，SEG12，LED4，RX1/DT1，PWMC4，PWMD3，CMP1_1-	I/O
	5	施密特触发输入，推挽输出，AN13，TK5，SEG13，LED5，PWMC3，CMP1_2-	I/O
	6	施密特触发输入，推挽输出，AN14，CAP，SEG14，LED6，PWMC2	I/O
	7	施密特触发输入，推挽输出，AN15，TK7，SEG15，LED7，PWMC1，T1G	I/O
PORTC	0	施密特触发输入，推挽输出， AN16，COM0，LED8，DP，SDA	I/O
	1	施密特触发输入，推挽输出， AN17，COM1，LED9，DM，SCL，MCLR	I/O
	2	施密特触发输入，推挽输出，AN18，COM2，CC0，TX0/CK0	I/O
	3	施密特触发输入，推挽输出，AN19，COM3，CC1，RX0/DT0	I/O
	4	施密特触发输入，推挽输出，编程时钟输入，AN20，COM4	I/O
	5	施密特触发输入，推挽输出，编程数据输入/输出，AN21，COM5	I/O
PORTD	0	施密特触发输入，推挽输出，AN22，SEG16，RX0/DT0，PWMD0	I/O
	1	施密特触发输入，推挽输出，AN23，SEG17，TX0/CK0，PWMD1	I/O
	2	施密特触发输入，推挽输出，AN24，SEG18，RX1/DT1，PWMD2	I/O
	3	施密特触发输入，推挽输出，AN25，SEG19，TX1/CK1，PWMD3	I/O
	4	施密特触发输入，推挽输出，AN26，SEG20， PWMD4	I/O
	5	施密特触发输入，推挽输出，AN27，SEG21	I/O
	6	施密特触发输入，推挽输出，AN28，SEG22	I/O
	7	施密特触发输入，推挽输出，AN29，SEG23	I/O

＜表 6-1：端口配置总概＞

6.1 I/O 口结构图

6.1.1 PORTA I/O 口结构图

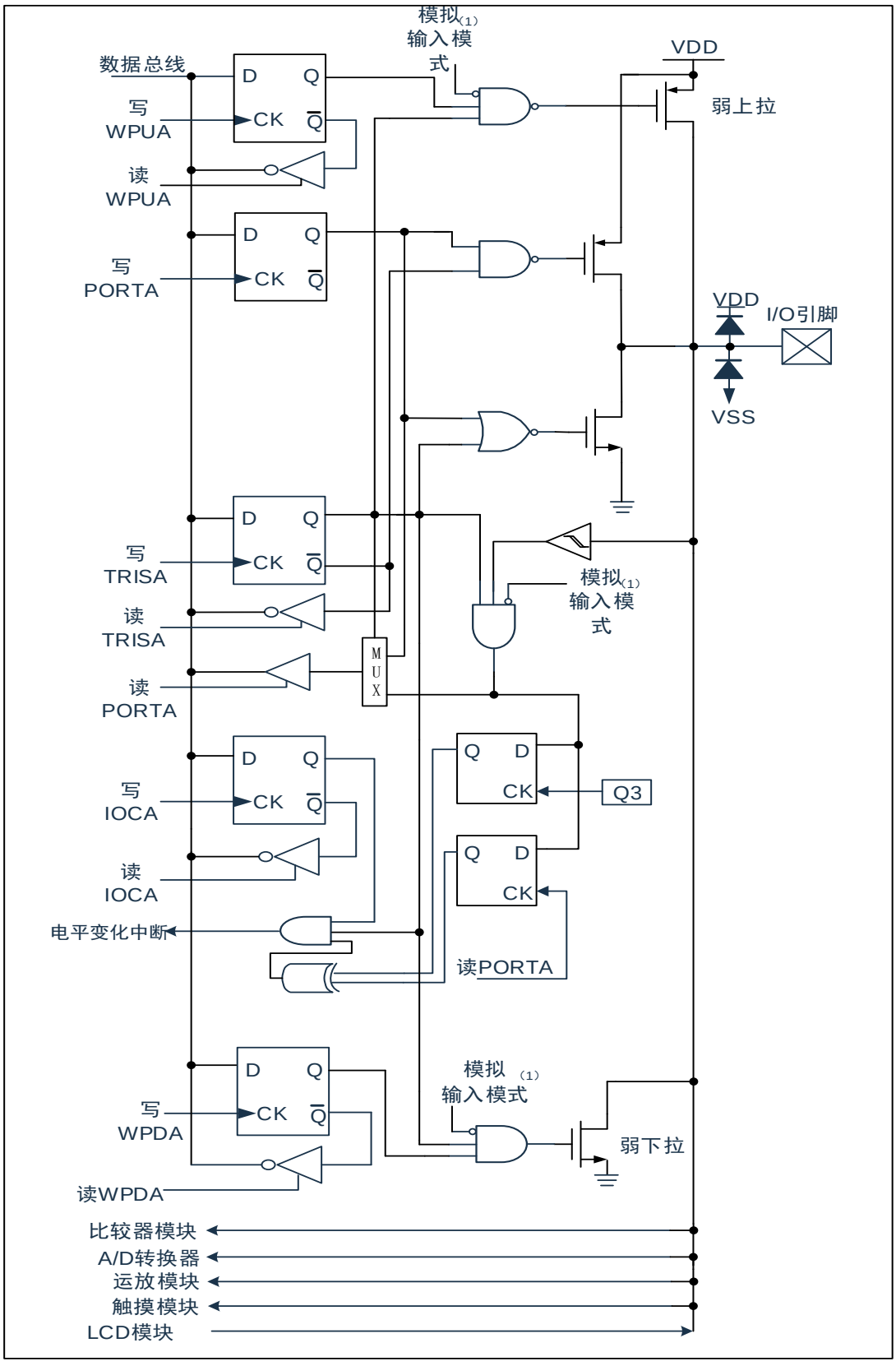


图 6-1：PORTA I/O 口结构图

6.1.2 PORTB I/O 口结构图

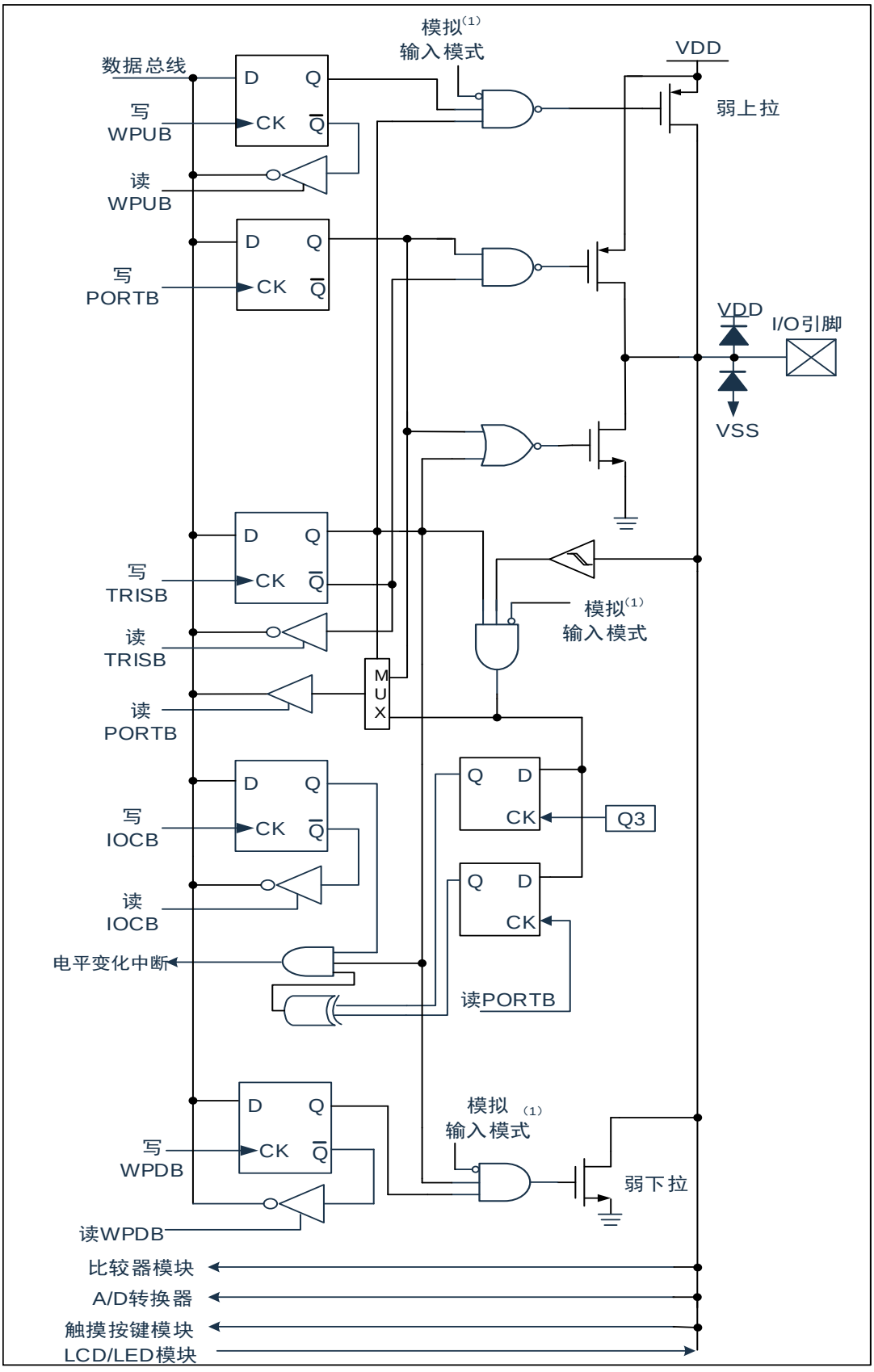


图 6-2：PORTB I/O 口结构图

6.1.3 PORTC I/O 口结构图

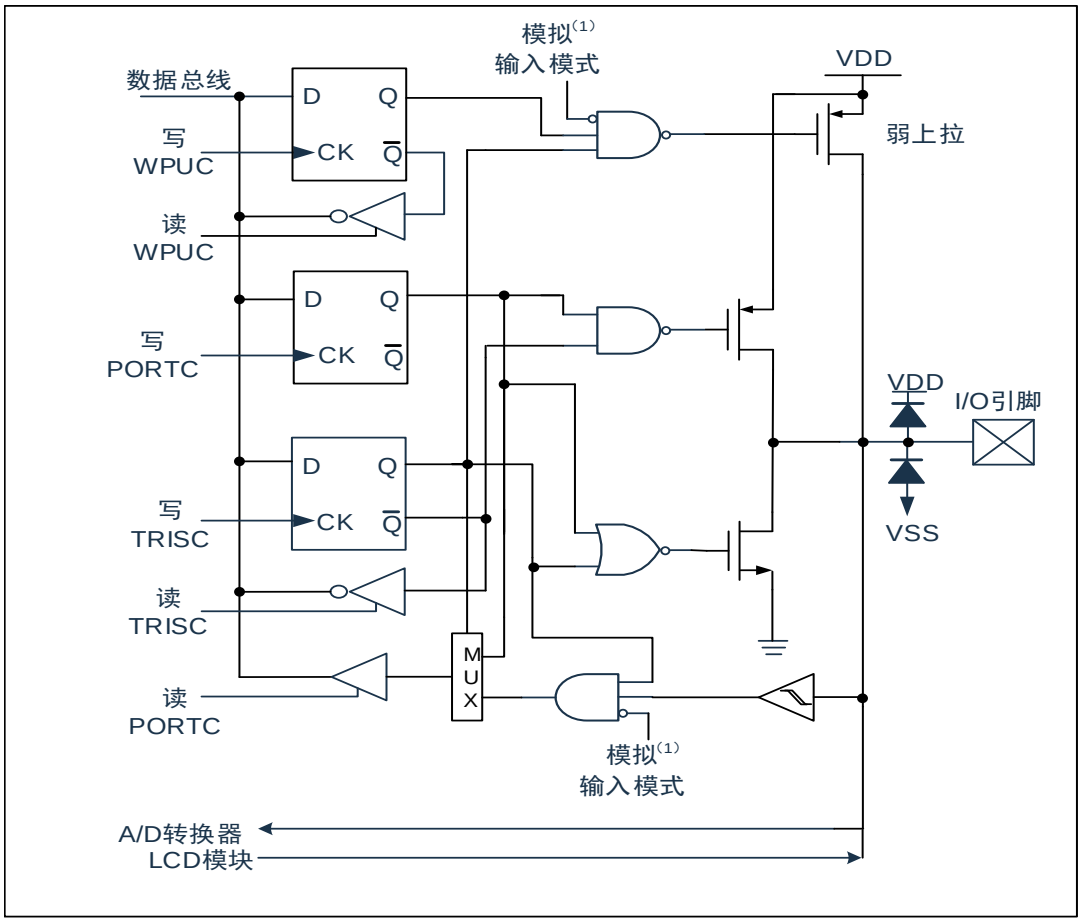


图 6-3：PORTC I/O 口结构图

注：ANx_EN 或 ANSx 决定模拟输入模式。

6.1.4 PORTD I/O 口结构图

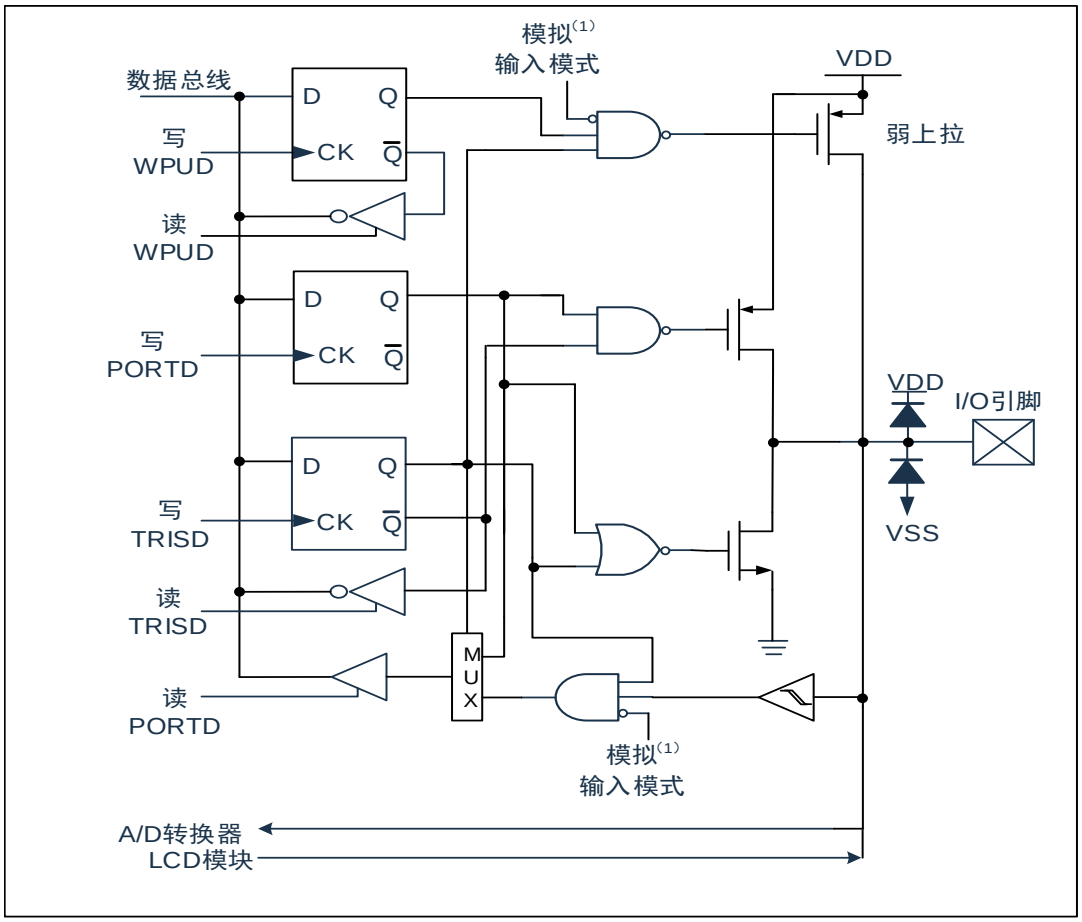


图 6-4：PORTD I/O 口结构图

6.2 PORTA

6.2.1 PORTA 数据及方向

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

读 PORTA 寄存器读的是引脚的状态而写该寄存器将会写入端口锁存器。所有写操作都是读－修改－写操作。因此，写一个端口就意味着先读该端口的引脚电平，修改读到的值，然后再将改好的值写入端口数据锁存器。即使在 PORTA 引脚用作模拟输入时，TRISA 寄存器仍然控制 PORTA 引脚的方向。当将 PORTA 引脚用作模拟输入时，用户必须确保 TRISA 寄存器中的位保持为置 1 状态。配置为模拟输入的 I/O 引脚总是读为 0。

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

PORTA 数据寄存器 PORTA(86H)

86H	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(85H)

85H	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'00110000'    ;设置PORTA<3:0>为输出口，PORTA<5:4>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<5:4>为输入口，所以赋0或1都没影响
```

6.2.2 PORTA 模拟选择控制

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

PORTA 模拟选择寄存器 ANSEL0(93H)

93H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL0	ANS7	ANS6	ANS5	ANS4	ANS3	ANS2	ANS1	ANS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

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

6.2.3 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或模拟输入时，其弱上拉会自动切断。

PORTA 上拉电阻寄存器 WPUA(88H)

88H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUA	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

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

注：如果引脚被配置为输出或模拟输入，将自动禁止弱上拉。

6.2.4 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或模拟输入时，其弱下拉会自动切断。

PORTA 下拉电阻寄存器 WPDA(87H)

87H	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或模拟输入，将自动禁止弱下拉。

6.2.5 PORTA 电平变化中断

所有的 PORTA 引脚都可以被单独配置为电平变化中断引脚。控制位 IOCA<7:0>允许或禁止每个引脚的该中断功能。上电复位时禁止引脚的电平变化中断功能。

对于已允许电平变化中断的引脚，则将该引脚上的值与上次读 PORTA 时锁存的旧值进行比较。将与上次读操作“不匹配”的输出一起进行逻辑或运算，以将 PIR1 寄存器中的 PORTA 电平变化中断标志位（RAIF）置 1。

该中断可将器件从休眠中唤醒，用户可在中断服务程序中通过以下步骤清除中断：

- 对 PORTA 进行读或写操作。这将结束引脚电平的不匹配状态。
- 将标志位 RAIF 清零。

不匹配状态会不断将 RAIF 标志位置 1。而读或写 PORTA 将结束不匹配状态，并且允许将 RAIF 标志位清零。

注：如果在执行读取操作时（Q2 周期的开始）I/O 引脚的电平发生变化，则 RAIF 中断标志位不会被置 1。此外，由于对端口的读或写影响到该端口的所有位，所以在电平变化中断模式下使用多个引脚的时候必须特别小心。在处理一个引脚电平变化的时候可能不会注意到另一个引脚上的电平变化。

PORTA 电平变化中断寄存器 IOCA(89H)

89H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCA	IOCA7	IOCA6	IOCA5	IOCA4	IOCA3	IOCA2	IOCA1	IOCA0
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　IOCA<7:0>　PORTA 的电平变化中断控制位
1=　允许电平变化中断
0=　禁止电平变化中断

6.3 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。配置为模拟输入的 I/O 引脚总是读为 0。

与 PORTB 口相关寄存器有 PORTB、TRISB、WUPB、WDPB、IOCB、ANSEL1 等。

PORTB 数据寄存器 PORTB(06H)

06H	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 (05H)

05H	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 模拟选择控制

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

PORTB 模拟选择寄存器 ANSEL1(94H)

94H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL1	ANS15	ANS14	ANS13	ANS12	ANS11	ANS10	ANS9	ANS8
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

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

6.3.3 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或模拟输入时，其弱上拉会自动切断。

PORTB 上拉电阻寄存器 WPUB(08H)

08H	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>:　PORTB 弱上拉使能位
1=　使能上拉
0=　禁止上拉

注：如果引脚被配置为输出或者模拟输入，将自动禁止弱上拉。

6.3.4 PORTB 下拉电阻

每个 PORTB 引脚都有可单独配置的内部弱下拉，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或模拟输入时，其弱下拉会自动切断。

PORTB 下拉电阻寄存器 WPDB(07H)

07H	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>:　PORTB 弱下拉使能位
1=　使能下拉
0=　禁止下拉

注：如果引脚被配置为输出或者模拟输入，将自动禁止弱下拉。

6.3.5 PORTB 电平变化中断

所有的 PORTB 引脚都可以被单独配置为电平变化中断引脚。控制位 IOCB<7:0>允许或禁止每个引脚的该中断功能。上电复位时禁止引脚的电平变化中断功能。

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

该中断可将器件从休眠中唤醒，用户可在中断服务程序中通过以下步骤清除中断：

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

不匹配状态会不断将 RBIF 标志位置 1。而读或写 PORTB 将结束不匹配状态，并且允许将 RBIF 标志位清零。

注：如果在执行读取操作时（Q2 周期的开始）I/O 引脚的电平发生变化，则 RBIF 中断标志位不会被置 1。此外，由于对端口的读或写影响到该端口的所有位，所以在电平变化中断模式下使用多个引脚的时候必须特别小心。在处理一个引脚电平变化的时候可能不会注意到另一个引脚上的电平变化。

PORTB 电平变化中断寄存器 IOCB(09H)

09H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IOCB	IOCB7	IOCB6	IOCB5	IOCB4	IOCB3	IOCB2	IOCB1	IOCB0
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　　IOCB<7:0>　PORTB 的电平变化中断控制位
1=　允许电平变化中断
0=　禁止电平变化中断

6.4 PORTC

6.4.1 PORTC 数据及方向

PORTC 是一个 6Bit 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。即使在 PORTC 引脚用作模拟输入时，TRISC 寄存器仍然控制 PORTC 引脚的方向。当将 PORTC 引脚用作模拟输入时，用户必须确保 TRISC 寄存器中的位保持为置 1 状态。配置为模拟输入的 I/O 引脚总是读为 0。

与 PORTC 口相关寄存器有 PORTC、TRISC、WUPC、ANSEL2 等。

PORTC 数据寄存器 PORTC(106H)

106H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTC	---	---	RC5	RC4	RC3	RC2	RC1	RC0
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　　未用

Bit5~Bit0　　PORTC<5:0>:　PORTC I/O 引脚位

1=　端口引脚电平>V_{IH}

0=　端口引脚电平<V_{IL}

注：RC2、RC3 为 PD 通讯的复用脚，分别为 CC0、CC1，芯片复位时默认开启 CC0、CC1 的下拉电阻；若要关闭 CC0、CC1 的下拉电阻，则需要在程序里将 CC0CON/CC1CON 寄存器的 CC0_RD/CC1_RD 位清零。

PORTC 方向寄存器 TRISC (105H)

105H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISC	---	---	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	---	1	1	1	1	1	1

Bit7~Bit6　　未用

Bit5~Bit0　　TRISC<5:0>:　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'00000001'    ;设置 PORTC<0>为输入口，PORTC<5:1>为输出口
LD      TRISC,A
```

6.4.2 PORTC 模拟选择控制

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

PORTC 模拟选择寄存器 ANSEL2(109H)

109H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL2	---	---	ANS21	ANS20	ANS19	ANS18	ANS17	ANS16
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	---	0	0	0	0	0	0

Bit7~Bit6　　未用

Bit5~Bit0　　ANS<21:16>:　模拟选择位,分别选择引脚 AN<21:16>的模拟或数字功能

1=　模拟输入，引脚被分配为模拟输入

0=　数字 I/O，引脚被分配给端口或特殊功能

6.4.3 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<5:0>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(108H)

108H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUC	---	---	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	---	0	0	0	0	0	0

Bit7~Bit6　　未用

Bit5~Bit0　　WPUC<5:0>:　PORTC 弱上拉使能位

1=　使能上拉

0=　禁止上拉

注：如果引脚被配置为输出或者模拟输入时，将自动禁止弱上拉。

6.5 PORTD

6.5.1 PORTD 数据及方向

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

读PORTD寄存器读的是引脚的状态而写该寄存器将会写入端口锁存器。所有写操作都是读-修改-写操作。因此，写一个端口就意味着先读该端口的引脚电平，修改读到的值，然后再将改好的值写入端口数据锁存器。即使在 PORTD 引脚用作模拟输入时，TRISD 寄存器仍然控制 PORTD 引脚的方向。当将 PORTD 引脚用作模拟输入时，用户必须确保 TRISD 寄存器中的位保持为置 1 状态。配置为模拟输入的 I/O 引脚总是读为 0。

与 PORTD 口相关寄存器有 PORTD、TRISD、WUPD、ANSEL3 等。

PORTD 数据寄存器 PORTD(107H)

107H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0
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　　PORTD<7:0>:　PORTD I/O 引脚位
1=　端口引脚电平>V_{IH}
0=　端口引脚电平<V_{IL}

PORTD 方向寄存器 TRISD (114H)

114H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISD	TRISD5	TRISD4	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0
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　　TRISD<7:0>:　PORTD 三态控制位
1=　PORTD引脚被配置为输入（三态）
0=　PORTD引脚被配置为输出

例：PORTD 口处理程序

```
CLR     PORTD          ;清数据寄存器
LDIA    B'00000001'    ;设置 PORTD<0>为输入口，PORTD<7:1>为输出口
LD      TRISD,A
```

6.5.2 PORTD 模拟选择控制

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

PORTD 模拟选择寄存器 ANSEL3(8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ANSEL3	ANS29	ANS28	ANS27	ANS26	ANS25	ANS24	ANS23	ANS22
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<29:22>:　模拟选择位,分别选择引脚 AN<29:22>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

6.5.3 PORTD 上拉电阻

每个 PORTD 引脚都有可单独配置的内部弱上拉。控制位 WPUD<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。

PORTD 上拉电阻寄存器 WPUD(115H)

115H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUD	WPUD7	WPUD6	WPUD5	WPUD4	WPUD3	WPUD2	WPUD1	WPUD0
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　WPUD<7:0>:　PORTD 弱上拉使能位
1=　使能上拉
0=　禁止上拉

注：如果引脚被配置为输出或者模拟输入时，将自动禁止弱上拉。

6.6 I/O 使用

6.6.1 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      PORTB,A     ;ACC 值赋给 PORTB 口
        CLRB    PORTB,1     ;PORTB.1 口清零
        SET     PORTB       ;PORTB 所有输出口置 1
        SETB    PORTB,1     ;PORTB.1 口置 1
```

6.6.2 读 I/O 口

例：读 I/O 口程序

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

6.7 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。

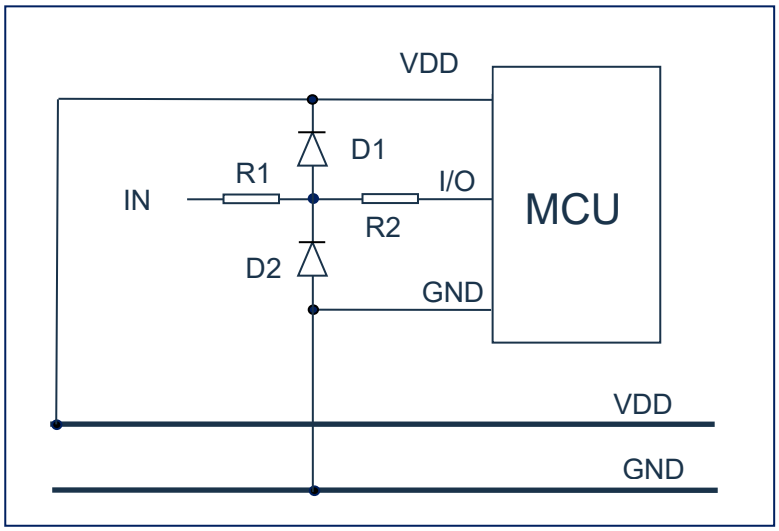


图 6-3：I/O 口注意事项连接图

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：

- PORTA 电平变化中断
- TIMER0 溢出中断
- TIMER2 匹配中断
- PWM 中断
- CMP1 中断
- USART0 接收中断
- USART1 接收中断
- IIC 总线冲突中断
- PD 发送中断
- A/D 中断
- PORTB 电平变化中断
- TIMER1 溢出中断
- 触摸中断
- INT 中断
- CMP2 中断
- USART0 发送中断
- USART1 发送中断
- IIC 中断
- PD 接收中断

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。

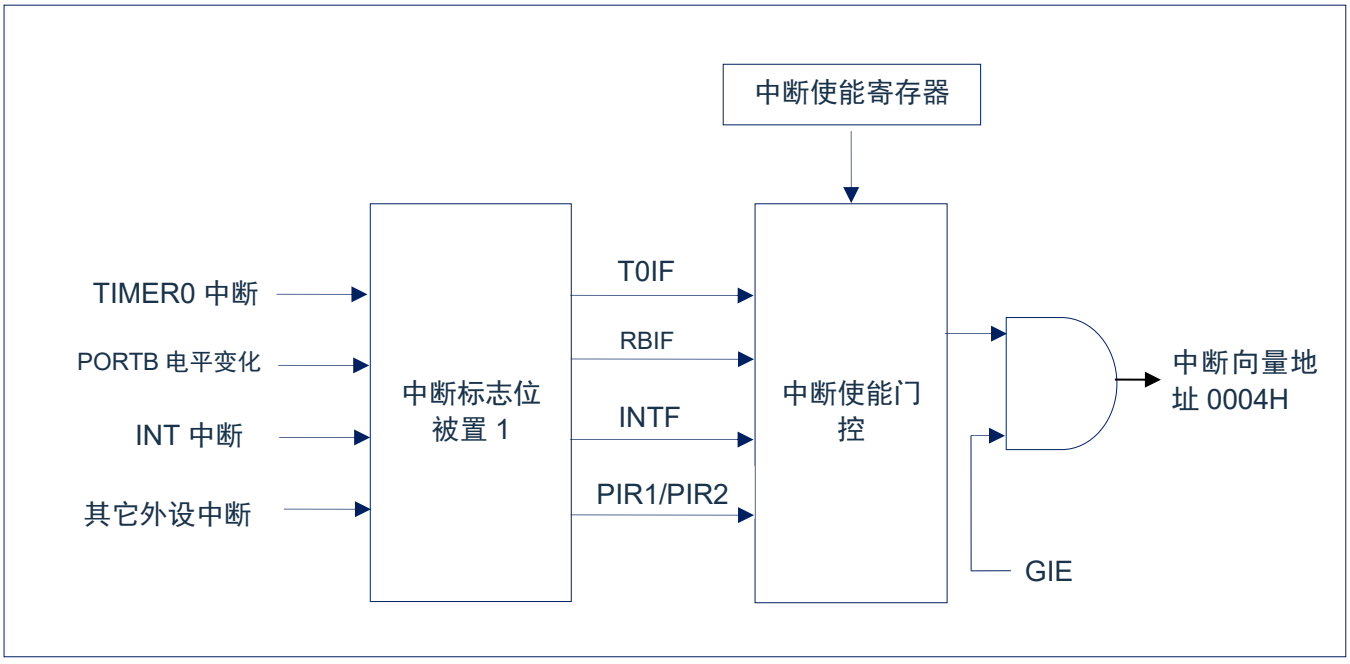


图 7-1：中断原理示意图

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 INT 中断、TIMER0 溢出中断、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. IOCB 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有 PIE1、PIE2，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(0EH)

0EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	RCIE	TXIE	CMP1IE	PWMIE	RAIE	TMR1IE	TMR2IE	ADIE
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 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

Bit6 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

Bit5 CMP1IE: 比较器1中断允许位
1= 允许比较器1中断
0= 禁止比较器1中断

Bit4 PWMIE: PWM中断允许位(PWM0/1/2/3)
1= 允许PWM中断
0= 禁止PWM中断

Bit3 RAIE: 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

Bit2 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

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 ADIE: ADC中断允许位
1= 允许ADC中断
0= 禁止ADC中断

外设中断允许寄存器 PIE2(110H)

110H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE2	BCLIE	TKIE	CMP2IE	IICIE	PDSIE	PDRIE	TX1IE	RC1IE
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 BCLIE : IIC总线冲突中断允许位
1= 允许IIC总线冲突中断
0= 禁止IIC总线冲突中断

Bit6 TKIE: 触摸检测结束中断允许位
1= 允许触摸检测结束中断
0= 禁止触摸检测结束中断

Bit5 CMP2IE: 比较器2中断允许位
1= 允许比较器2中断
0= 禁止比较器2中断

Bit4 IICIE: IIC中断允许位
1= 允许IIC中断
0= 禁止IIC中断

Bit3 PDSIE: PD数据发送中断允许位
1= 允许PD数据发送中断
0= 禁止PD数据发送中断

Bit2 PDRIE: PD数据接收中断允许位
1= 允许PD数据接收中断
0= 禁止PD数据接收中断

Bit1 TX1IE: USART1发送中断允许位
1= 允许USART1发送中断
0= 禁止USART1发送中断

Bit0 RC1IE: USART1接收中断允许位
1= 允许USART1接收中断
0= 禁止USART1接收中断

7.2.3 外设中断请求寄存器

外设中断请求寄存器有 PIR1、PIR2。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(0DH)

0DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR1	RCIF	TXIF	CMP1IF	PWMIF	RAIF	TMR1IF	TMR2IF	ADIF
R/W	R	R	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7 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

Bit6 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发送缓冲非空

Bit5 CMP1IF: 比较器1中断标志位(必须由软件清零)
1= 发生比较器1中断
0= 未发生比较器1中断

Bit4 PWMIF: PWM中断标志位(PWM0/1/2/3)(必须由软件清零)
1= 发生PWM中断
0= 未发生PWM中断

Bit3 RAIF: 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

Bit2 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

Bit1 TMR2IF: TIMER2与PR2匹配中断标志位(必须由软件清零)
1= 发生TMR2与PR2匹配中断
0= 未发生TMR2与PR2匹配中断

Bit0 ADIF: AD转换器中断标志位
1= AD转换完成（必须由软件清零）
0= AD转换未完成或尚未启动

外设中断请求寄存器 PIR2(10FH)

10FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR2	BCLIF	TKIF	CMP2IF	IICIF	PDSIF	PDRIF	TX1IF	RC1IF
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 BCLIF: IIC总线冲突中断标志位
1= 当配置为IIC主控模式时，IIC模块发生了总线冲突（必须由软件清零）
0= 未发生总线冲突

Bit6 TKIF: 触摸检测结束中断标志位
1= 产生了触摸检测结束中断（必须由软件清零）
0= 未产生触摸检测结束中断

Bit5 CMP2IF: 比较器2中断标志位(必须由软件清零)
1= 发生比较器2中断
0= 未发生比较器2中断

Bit5 CMP2IF: 比较器2中断标志位
1= 发生比较器2中断(必须由软件清零)
0= 未发生比较器2中断

Bit4 IICIF: IIC中断标志位
1= 产生了IIC中断条件，在从中断服务程序返回前必须由软件清零。使该位置1的条件有：
IIC从动/主控：
- 发生发送/接收

IIC主控：
- 发生的启动条件由IIC模块完成
- 发生的停止条件由IIC模块完成
- 发生的重新启动条件由IIC模块完成
- 发生的应答条件由IIC模块完成
- 当IIC模块空闲时发生启动条件（多主机系统）
- 当IIC模块空闲时发生停止条件（多主机系统）

0= 没有产生IIC中断条件

Bit3 PDSIF: PD数据发送中断标志位
1= 产生了PD数据发送完成中断（必须由软件清零）
0= 未产生PD数据发送完成中断

Bit2 PDRIF: PD数据接收中断标志位
1= 产生了PD数据接收完成中断（必须由软件清零）
0= 未产生PD数据接收完成中断

Bit1 TX1IF: USART1发送中断标志位
1= USART1发送缓冲器空（通过写TXREG1清零）
0= USART1发送缓冲器满

Bit0 RC1IF: USART1接收中断标志位
1= USART1接收缓冲器非空（通过读RCREG1清零）
0= USART1接收缓冲器空

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。响应中断之前，必须保存 ACC、STATUS 的内容。芯片没有提供专用的入栈保存和出栈恢复指令，用户需自己保护 ACC 和 STATUS 的内容，以避免中断结束后可能的程序运行错误。

例：对 ACC 与 STATUS 进行入栈保护

```
                ORG     0000H
                JP      START           ;用户程序起始地址
                ORG     0004H
                JP      INT_SERVICE     ;中断服务程序
                ORG     0008H
        START：
                ...
                ...
INT_SERVICE：
        PUSH：                          ;中断服务程序入口，保存 ACC 及 STATUS
                LD      ACC_BAK,A       ;保存 ACC 的值，(ACC_BAK 需自定义)
                SWAPA   STATUS
                LD      STATUS_BAK,A    ;保存 STATUS 的值，（STATUS_BAK 需自定义）
                ...
                ...
         POP：                          ;中断服务程序出口，还原 ACC 及 STATUS
                SWAPA   STATUS_BAK
                LD      STATUS,A        ;还原 STATUS 的值
                SWAPR   ACC_BAK         ;还原 ACC 的值
                SWAPA   ACC_BAK
                RETI
```

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；
- 可选外部 32.768K 振荡时钟（F_{LSE}）；
- 溢出中断。

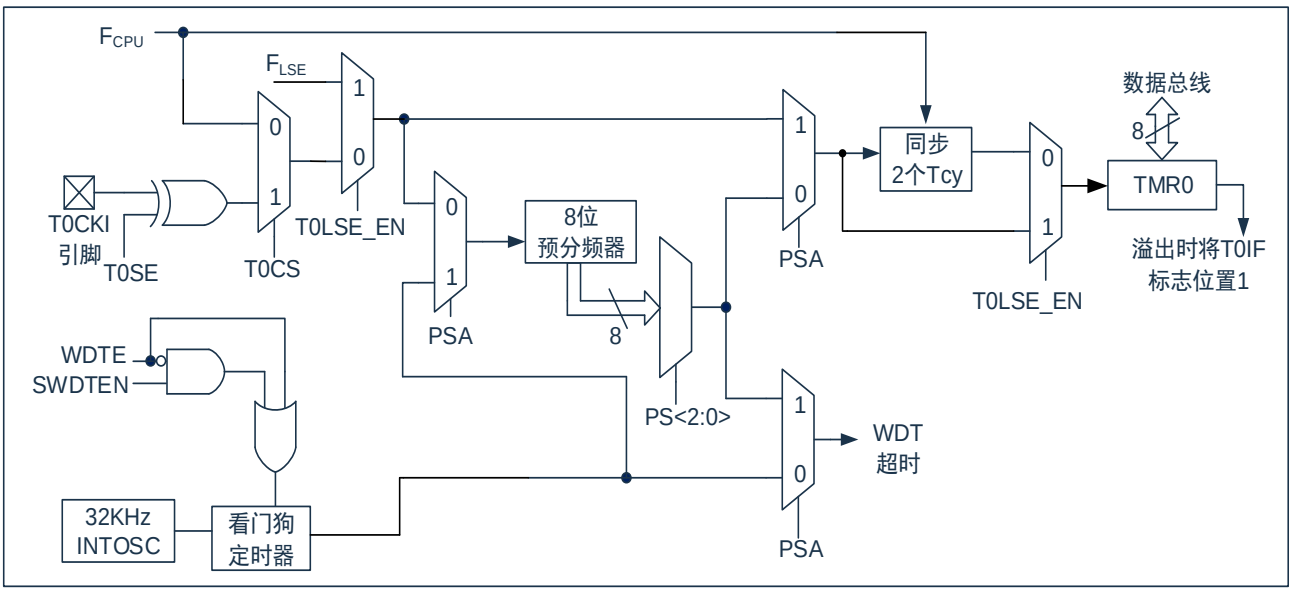


图 8-1：TIMER0/WDT 结构图

注：

1. T0SE、T0LSE_EN、T0CS、PSA、PS<2:0>为OPTION_REG寄存器中的位。
2. SWDTEN为OSCCON寄存器中的位。
3. WDTEN 位在 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

由 TIMER0 还是 WDT 使用预分频器，完全由软件控制，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。

```
CLR     TMR0            ;TMR0 清零
CLRWDT                  ;WDT 清零
LDIA    B'00xx1111'
LD      OPTION_REG,A
LDIA    B'00xx1xxx'     ;设置新的预分频器
LD      OPTION_REG,A
```

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
```

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：只有将时钟源选择 F_{LSE} 时， TIMER0 中断才能唤醒处理器。

8.3 TIMER0 相关寄存器

有两个寄存器与 TIMER0 相关，8 位定时器/计数器（TMR0）和 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 位可读写寄存器，用户可改变 OPTION_REG 的值，来改变 TIMER0 的工作模式等。请参看 2.6 关于预分频寄存器（OPTION_REG）的应用。

8 位定时器/计数器 TMR0（81H）

81H	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

预分频器控制寄存器 OPTION_REG（01H）

01H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	T0LSE_EN	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	1	1	1	1	0	1	1

Bit7 T0LSE_EN: TIMER0 时钟源选择 F_{LSE} 使能位
0= TIMER0 时钟源由 T0CS 决定
1= TIMER0 时钟源选择 F_{LSE}
Bit6 INTEDG: 触发中断的边沿选择位
0= INT 引脚下降沿触发中断
1= INT 引脚上升沿触发中断
Bit5 T0CS: TIMER0 时钟源选择位
0= 内部指令周期时钟（F_{CPU}）
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 (WDT_DIV=DISABLE)	WDT 分频比 (WDT_DIV=ENABLE)
0	0	0	1:2	1:1	1:3
0	0	1	1:4	1:2	1:6
0	1	0	1:8	1:4	1:12
0	1	1	1:16	1:8	1:24
1	0	0	1:32	1:16	1:48
1	0	1	1:64	1:32	1:96
1	1	0	1:128	1:64	1:192
1	1	1	1:256	1:128	1:384

9. 定时计数器 TIMER1

9.1 TIMER1 概述

TIMER1 模块是一个 16 位定时器/计数器，具有以下特性：

- 16 位定时器/计数器寄存器（TMR1H:TMR1L）
- 3 位预分频器
- 同步或异步操作
- 溢出时唤醒（仅外部时钟异步模式）
- 可编程内部或外部时钟源
- 可选 LP 振荡器(F_{LSE})
- 通过 T1G 引脚门控 TIMER1（使能计数）
- 溢出中断

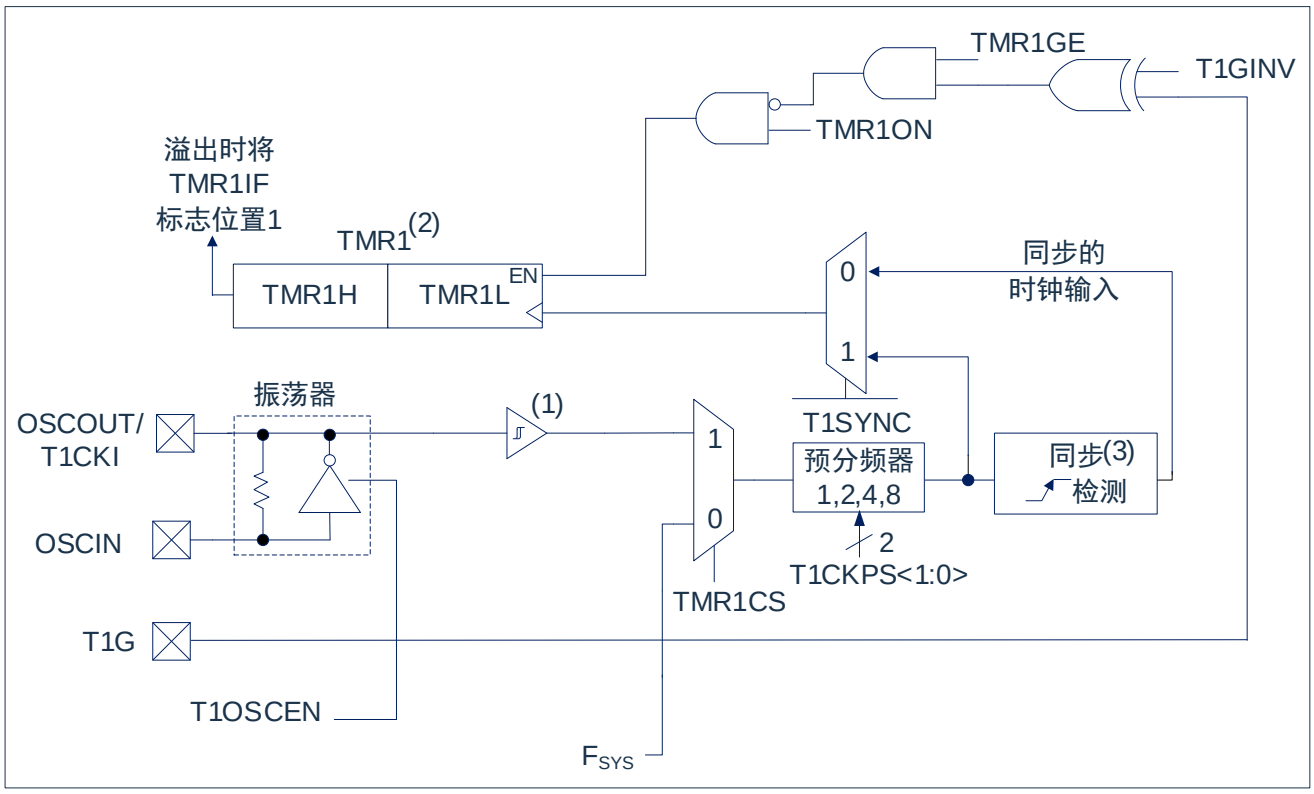


图9-1：TIMER1结构图

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，此模块用作定时器。当与外部时钟源一同使用时，此模块可用作定时器或计数器。

9.3 时钟源选择

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

时钟源	TMR1CS
F_{SYS}	0
T1CKI 引脚	1

9.3.1 内部时钟源

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

9.3.2 外部时钟源

选择外部时钟源后，TIMER1 模块可作为定时器或计数器。

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

如需一个外部时钟振荡器，TIMER1 可使用 LP 振荡器作为时钟源。

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

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

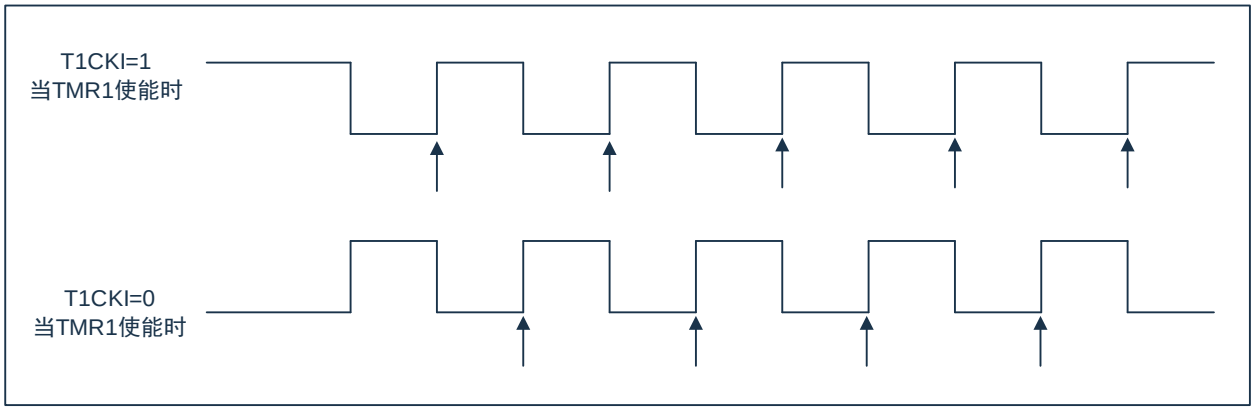


图 9-2：TIMER1 的递增边沿

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

9.4 TIMER1 预分频器

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

9.5 TIMER1 振荡器

在 T1OSI（输入）引脚和 T1OSO（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。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 PORTB0 和 PORTB1 被置为模拟输入。

> 注：振荡器需要经过一段起振和稳定时间后才能使用。因此，在使能 TIMER1 前应将 T1OSCEN 置 1 并经过适当的延时。

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，在用软件对定时器进行读/写操作时应该特别小心（请参见第 9.6.1 节“异步计数器模式下对 TIMER1 的读写”）。

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

9.6.1 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。

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，可使用外部晶振或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 相关寄存器

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

TIMER1 数据低位寄存器 TMR1L(10CH)

10CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR1L	TMR1<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0　　TMR1<7:0>:　TIMER1 数据低 8 位

TIMER1 数据高位寄存器 TMR1H(10DH)

10DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TMR1H	TMR1<15:8>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0　　TMR1<15:8>:　TIMER1 数据高 8 位

TIMER1 控制寄存器 T1CON(101H)

101H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
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　　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 LP 振荡器时钟源或来自 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：

1. 8 位定时器寄存器（TMR2）
2. 8 位周期寄存器（PR2）
3. TMR2 与 PR2 匹配时中断
4. 软件可编程预分频比（1:1、1:4 和 1:16）
5. 软件可编程后分频比（1:1 至 1:16）
6. 可选外部 32.768KHz 振荡时钟（F_{LSE}）

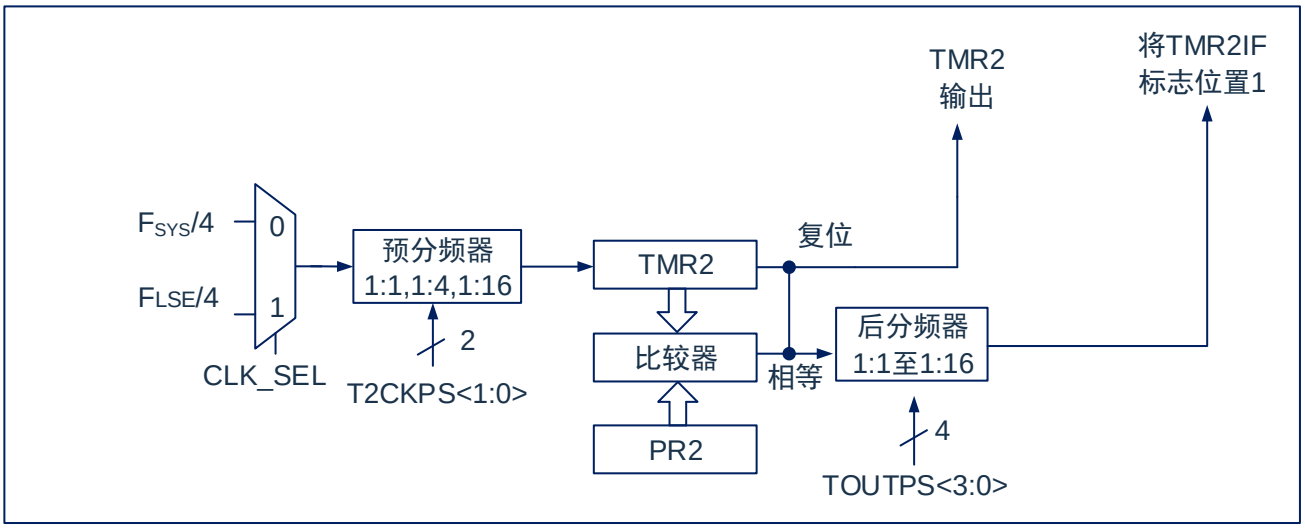


图 10-1：TIMER2 框图

10.2 TIMER2 的工作原理

TIMER2 模块的输入时钟是系统指令时钟（$F_{SYS}/4$）或外部 32.768kHz 振荡（$F_{LSE}/4$）。时钟被输入到 TIMER2 预分频器，有如下几种分频比可供选择：1:1、1:4 或 1:16。预分频器的输出随后用于使 TMR2 寄存器递增。

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

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

TIMER2 与 PR2 比较器的匹配输出随后输入给 TIMER2 的后分频器，后分频器具有 1:1 至 1:16 的后分频比可供选择。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：

1. 对 TMR2ON=0 时。
2. 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，分别是数据寄存器 TMR2，周期寄存器 PR2 和控制寄存器 T2CON。

TIMER2 数据寄存器 TMR2（12H）

12H	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（11H）

11H	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(13H)

13H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T2CON	CLK_SEL	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
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 CLK_SEL: 时钟源选择
1= 选择外部 F_{LSE}/4 作为 TMR2 时钟源（休眠态可继续计数）
0= 选择内部 F_{SYS}/4 作为 TMR2 时钟源

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. 10 位 PWM 模块

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

11.1 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。

11.2 相关寄存器说明

PWM 控制寄存器 PWMCON0(15H)

15H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON0	CLKDIV<2:0>			PWM4EN	PWM3EN	PWM2EN	PWM1EN	PWM0EN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

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

Bit4~Bit0 PWM0/1/2/3/4EN: PWM0/1/2/3/4使能位
1= 使能PWM0/1/2/3/4
0= 禁止PWM0/1/2/3/4

PWM 控制寄存器 PWMCON1(16H)

16H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON1	PWMIO_SEL<1:0>		PWM2DTEN	PWM0DTEN	---	---	DT_DIV<1:0>	
R/W	R/W	R/W	R/W	R/W	---	---	R/W	R/W
复位值	0	0	0	0	---	---	0	0

Bit7~Bit6 PWMIO_SEL： PWM IO分组选择
11= PWM分配在A组，PWM0-RA0,PWM1-RA1,PWM2-RA2,PWM3-RA3,PWM4-RA4
10= PWM分配在B组，PWM0-RA0,PWM1-RA1,PWM2-RA2,PWM3-RB2,PWM4-RB1
01= PWM分配在C组，PWM0-RA5,PWM1-RB7,PWM2-RB6,PWM3-RB5,PWM4-RB4
00= PWM分配在D组，PWM0-RB0,PWM1-RB1,PWM2-RB3,PWM3-RB4,PWM4-RB2
Bit5 PWM2DTEN： PWM2死区使能位
1= 使能PWM2死区功能，PWM2和PWM3组成一对互补输出
0= 禁止PWM2死区功能
Bit4 PWM0DTEN： PWM0死区使能位
1= 使能PWM0死区功能，PWM0和PWM1组成一对互补输出
0= 禁止PWM0死区功能
Bit3~Bit2 未用。
Bit1~Bit0 DT_DIV<1:0>: 死区时钟源分频
11= F_{HSI}/8
10= F_{HSI}/4
01= F_{HSI}/2
00= F_{HSI}/1

注：选择 PWMD 组时，PWM0~PWM4 可通过 config 配置在 RB0~RB4 或 RD0~RD4。

PWM 控制寄存器 PWMCON2(1DH)

1DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMCON2	---	---	---	PWM4DIR	PWM3DIR	PWM2DIR	PWM1DIR	PWM0DIR
R/W	---	---	---	R/W	R/W	R/W	R/W	R/W
复位值	---	---	---	0	0	0	0	0

Bit7~Bit5 未用
Bit4~Bit0 PWM0/1/2/3/4DIR PWM输出取反控制位
1= PWM0/1/2/3/4取反输出
0= PWM0/1/2/3/4正常输出

PWM0~PWM3 周期低位寄存器 PWMTL (17H)

17H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTL	PWMT<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMT<7:0>: PWM0~PWM3周期低8位

PWM4 周期低位寄存器 PWMT4L(1CH)

1CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMT4L	PWM4T<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 PWM4T<7:0>: PWM4周期低8位

周期高位寄存器 PWMTH (18H)

18H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMTH	---	---	PWMD4<9:8>		PWM4T<9:8>		PWMT<9:8>	
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	---	0	0	0	0	0	0

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

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

PWM0 占空比低位寄存器 PWMD0L (19H)

19H	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 (1AH)

1AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD1L	PWMD1<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD1<7:0>: PWM1占空比低8位

PWM2 占空比低位寄存器 PWMD2L (9BH)

9BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD2L	PWMD2<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD2<7:0>: PWM2占空比低8位

PWM3 占空比低位寄存器 PWMD3L (9CH)

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD3L	PWMD3<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD3<7:0>: PWM3占空比低8位

PWM4 占空比低位寄存器 PWMD4L (1BH)

1BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMD4L	PWMD4<7:0>							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit0 PWMD4<7:0>: PWM4占空比低8位

PWM0~PWM3 占空比高位寄存器 PWMDH (1EH)

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWMDH	PWMD3<9:8>		PWMD2<9:8>		PWMD1<9:8>		PWMD0<9:8>	
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 PWMD3<9:8>: PWM3占空比高2位
Bit5~Bit4 PWMD2<9:8>: PWM2占空比高2位
Bit3~Bit2 PWMD1<9:8>: PWM1占空比高2位
Bit1~Bit0 PWMD0<9:8>: PWM0占空比高2位

注：写入 PWMDx<9:8>并不能立即生效，需有写入 PWMDxL 操作后才能生效。（x=0，1，2，3）

PWM0 和 PWM1 死区时间寄存器 PWM01DT(1FH)

1FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM01DT	---	---	PWM01DT<5:0>					
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	---	0	0	0	0	0	0

Bit7~Bit6 未用。
Bit5~Bit0 PWM01DT<5:0>: PWM0和PWM1死区时间

PWM2 和 PWM3 死区时间寄存器 PWM23DT(9DH)

9DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PWM23DT	---	---	PWM23DT<5:0>					
R/W	---	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	---	---	0	0	0	0	0	0

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

11.3 10 位 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；

3）以上操作顺序只针对 PWM0、PWM1、PWM2、PWM3、PWM4 占空比寄存器。

11.4 10 位 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：

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

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 中断标志位；（PWM4 无中断）

11.5 10 位 PWM 占空比

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

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

公式 2：脉冲宽度计算公式：

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

公式3：PWM占空比计算公式：

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

PWM 占空比在芯片内部有双重缓冲。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。

11.6 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。

11.7 可编程的死区延时模式

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

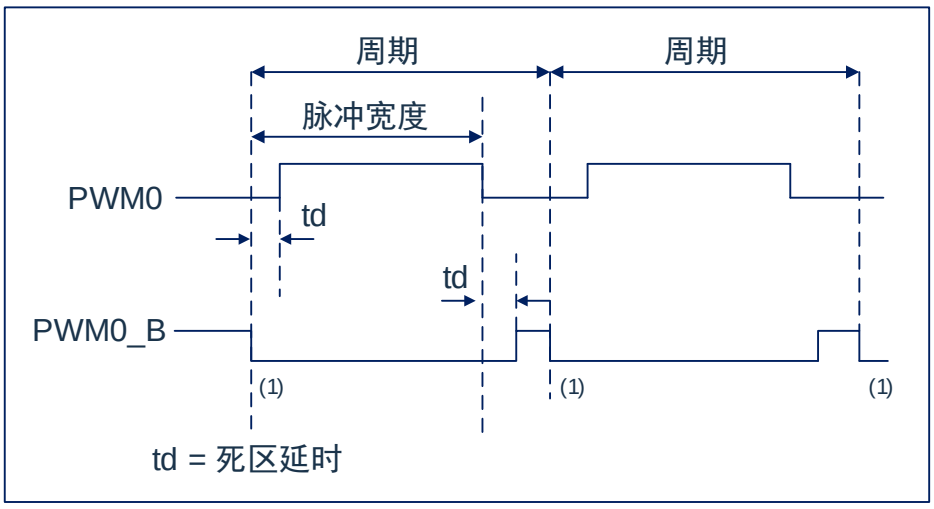


图11-1：PWM死区延时输出示例

死区时间计算公式为：

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

11.8 10 位 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，PWMTL 寄存器设置 PWM 周期。
3. 通过装载 PWMDxL，PWMDxxH 寄存器设置 PWM 占空比。
4. 清零 PWMIF 标志位。
5. 设置 PWMENx 位以使能相应 PWM 输出。
6. 在新的 PWM 周期开始后，使能 PWM 输出：
   - 等待 PWMIF 位置 1；
   - 通过将相应的 TRIS 位清零，使能 PWM 引脚输出驱动器。

12. 通用同步/异步收发器(USART0 和 USART1)

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

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

USARTx 模块包含如下功能：

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

以下图 12-1 和图 12-2 为 USARTx 收发器的框图。

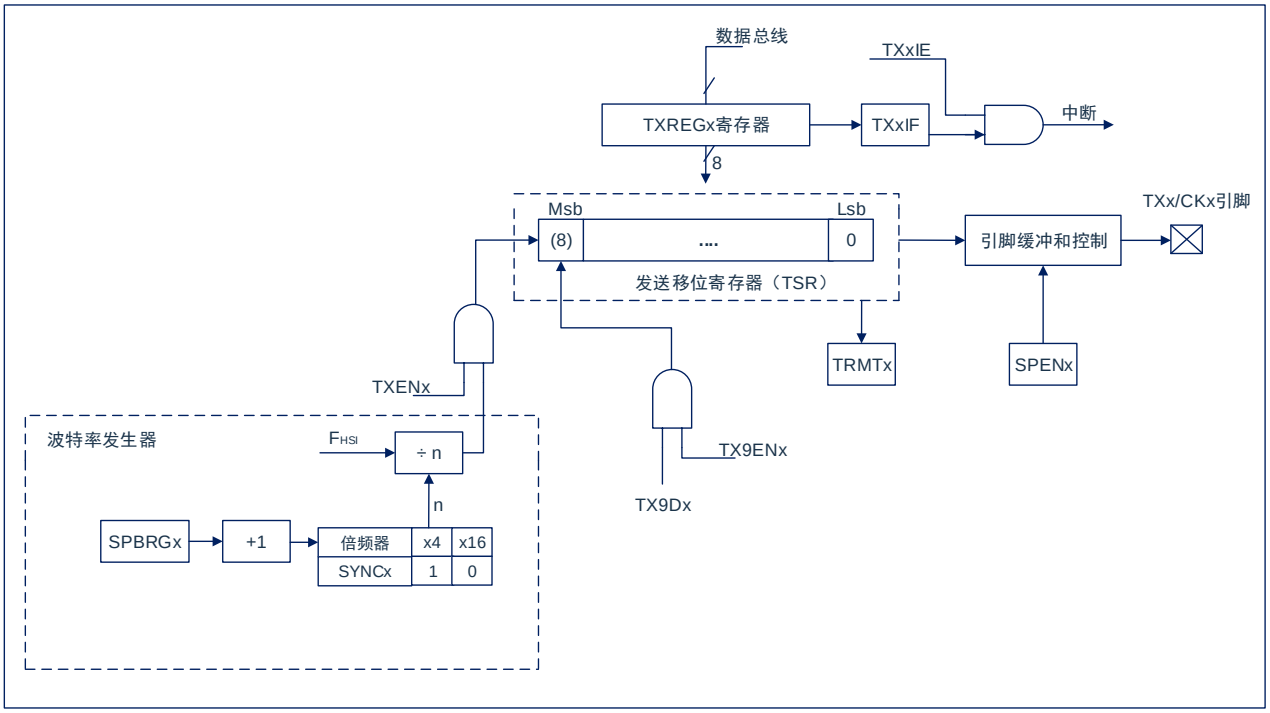


图12-1：USARTx发送框图

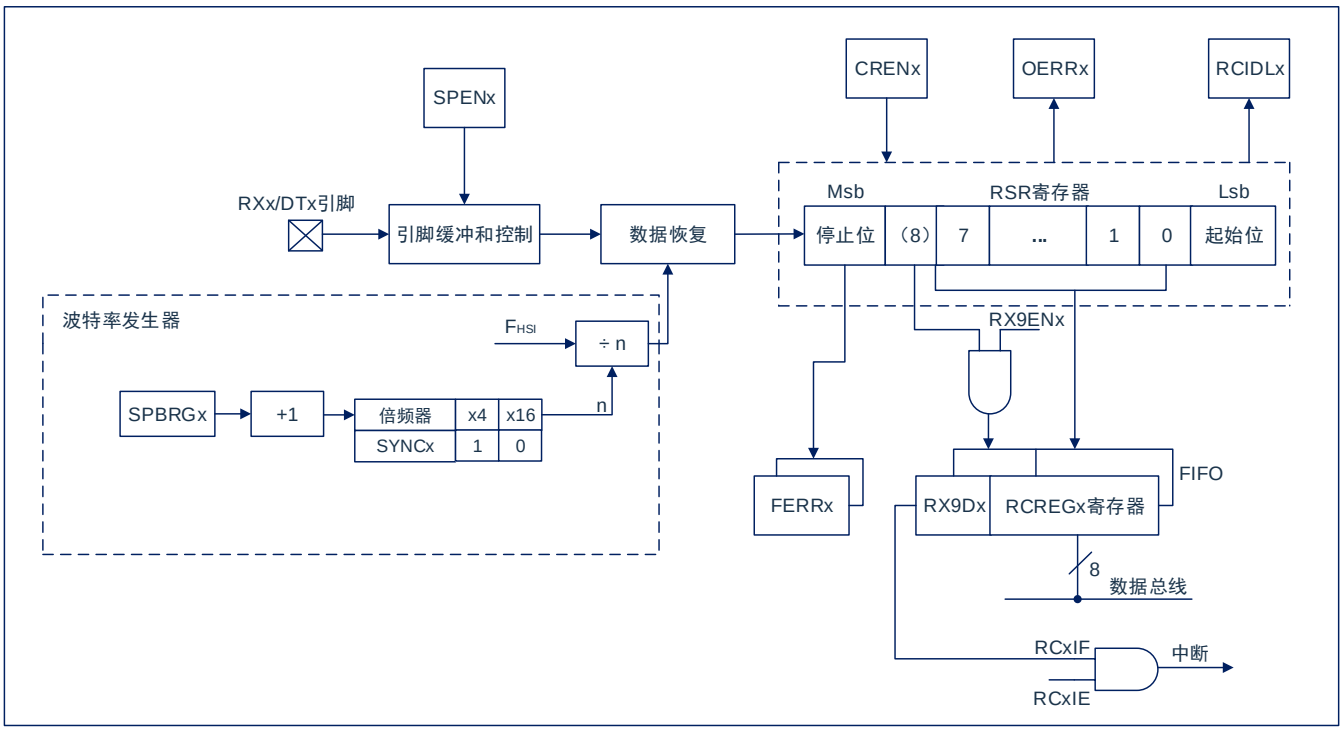


图12-2：USARTx接收框图

USARTx 模块的操作是通过 3 个寄存器控制的：

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

12.1 USARTx 异步模式

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

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

12.1.1 USARTx 异步发送器

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

12.1.1.1 使能发送器

通过配置如下三个控制位使能 USARTx 发送器，以用于异步操作：

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

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

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

12.1.1.2 发送数据

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

12.1.1.3 发送中断标志

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

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

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

12.1.1.4 TSR 状态

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

> 注：TSR 寄存器并未映射到数据存储器中，因此用户不能直接访问它。

12.1.1.5 发送 9 位字符

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

12.1.1.6 设置异步发送

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

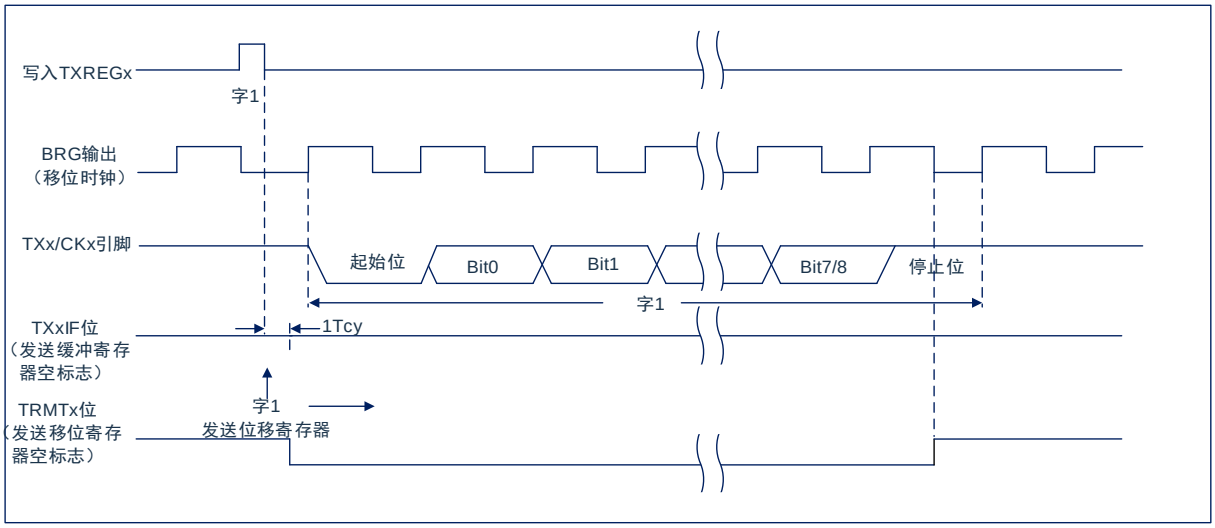


图 12-3：异步发送

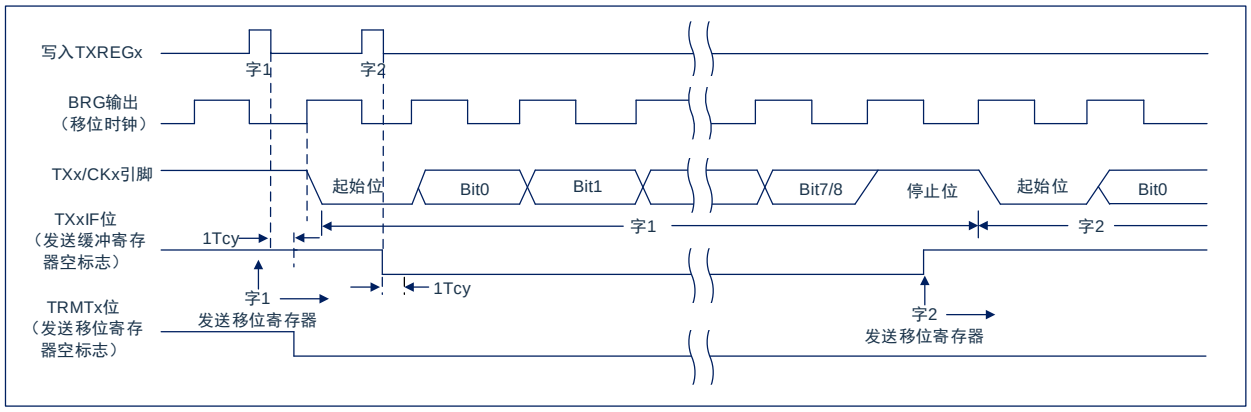


图 12-4：异步发送（背靠背）

注：本时序图显示了两次连续的发送。

12.1.2 USARTx 异步接收器

异步模式通常用于 RS-232 系统。图 12-2 给出了接收器的框图。在 RXx/DTx 引脚上接收数据和驱动数据恢复电路。数据恢复电路实际上是一个以 16 倍波特率为工作频率的高速移位器，而串行接收移位寄存器（ReceiveShiftRegister，RSR）则以比特率工作。当字符的全部 8 位或 9 位数据位被移入后，立即将它们传输到一个 2 字符的先入先出（FIFO）缓冲器。FIFO 缓冲器允许接收 2 个完整的字符和第 3 个字符的起始位，然后必须由软件将接收到的数据提供给 USARTx 接收器。FIFO 和 RSR 寄存器不能直接由软件访问。通过 RCREGx 寄存器访问接收到的数据。

12.1.2.1 使能接收器

通过配置如下三个控制位使能 USARTx 接收器，以用于异步操作。

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

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

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

12.1.2.2 接收数据

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

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

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

12.1.2.3 接收中断

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

通过将下列所有位均置 1 来允许 RCxIF 中断：

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

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

12.1.2.4 接收帧错误

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

由 RCSTAx 寄存器的 FERRx 位获取帧错误状态。

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

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

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

12.1.2.5 接收溢出错误

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

12.1.2.6 接收 9 位字符

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

12.1.2.7 异步接收设置

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

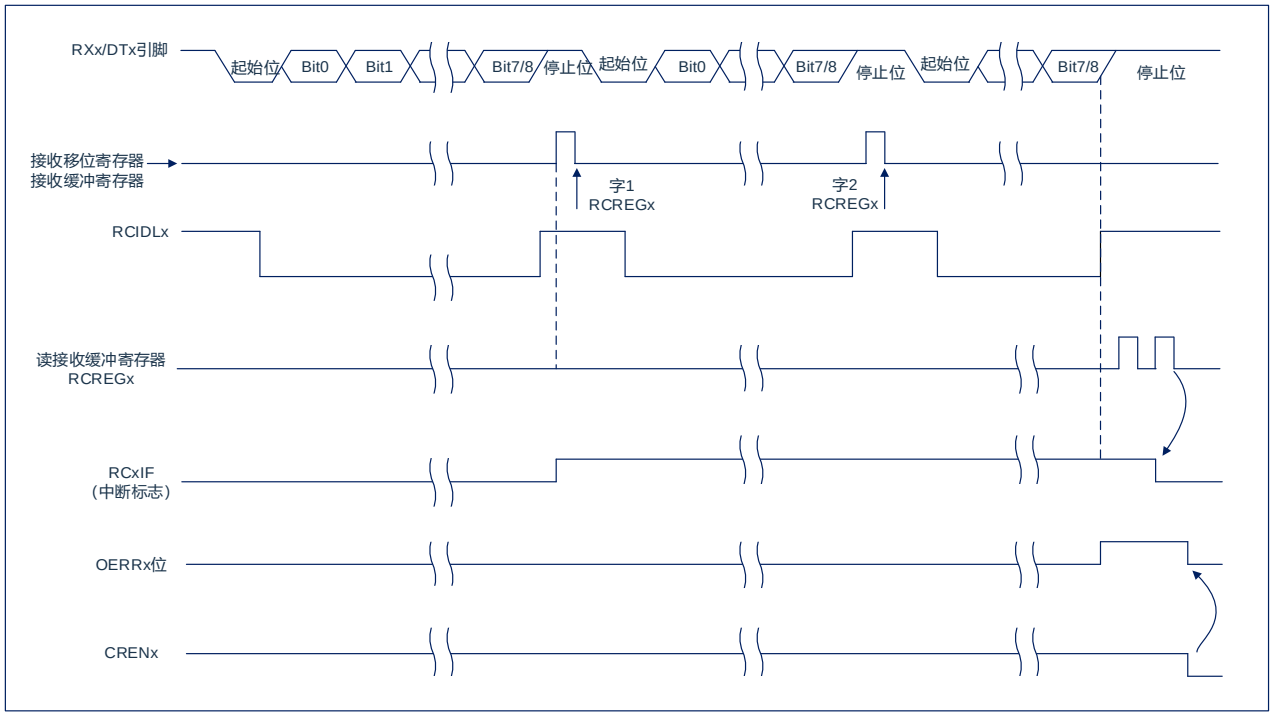


图 12-5：异步接收

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

12.2 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。

12.3 USARTx 相关寄存器

发送状态和控制寄存器 TXSTAx

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

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

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

Bit5 TXENx: 发送使能位(1)
1= 使能发送
0= 禁止发送

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

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

Bit2 未用

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

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

注：同步模式下，SRENx/CRENx 会覆盖 TXENx 的值。

接收状态和控制寄存器 RCSTAx

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

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

发送数据寄存器 TXREGx

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

接收数据寄存器 RCREGx

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

波特率数据寄存器 SPBRGx

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

12.4 USARTx 波特率发生器（BRG）

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

SPBRGx 寄存器对决定自由运行的波特率定时器的周期。

表 12-1 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 12-2 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 SPBRGx 寄存器对写入新值会导致 BRG 定时器复位（或清零）。这可以确保 BRG 无需等待定时器溢出就可以输出新的波特率。

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

公式 1：计算波特率误差

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

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

求解 SPBRGx：

$$X=\frac{\frac{F_{HSI}}{目标波特率}}{16}-1=\frac{\frac{160000000}{9600}}{16}-1=[103.16]=103$$

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

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

表 12-1：波特率公式

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

说明：n= SPBRGx 寄存器的值。

表 12-2：异步模式下的波特率

目标波特率	SYNCx=0		
	F_{HSI}=16.00MHz		
	实际波特率	误差（%）	SPBRGx 值
2400	----	----	----
9600	9615	0.16	103
10417	10417	0	95
14400	14492	0.63	68
19200	19230	0.16	51

12.5 USARTx 同步模式

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

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

同步发送无需使用起始位和停止位。

12.5.1 同步主控模式

下列位用来将 USARTx 配置为同步主控操作：

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

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

12.5.1.1 主控时钟

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

12.5.1.2 时钟极性

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

12.5.1.3 同步主控发送

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

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

注：TSR 寄存器并未映射到数据存储器中，因此用户不能直接访问它。

12.5.1.4 同步主控发送设置

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

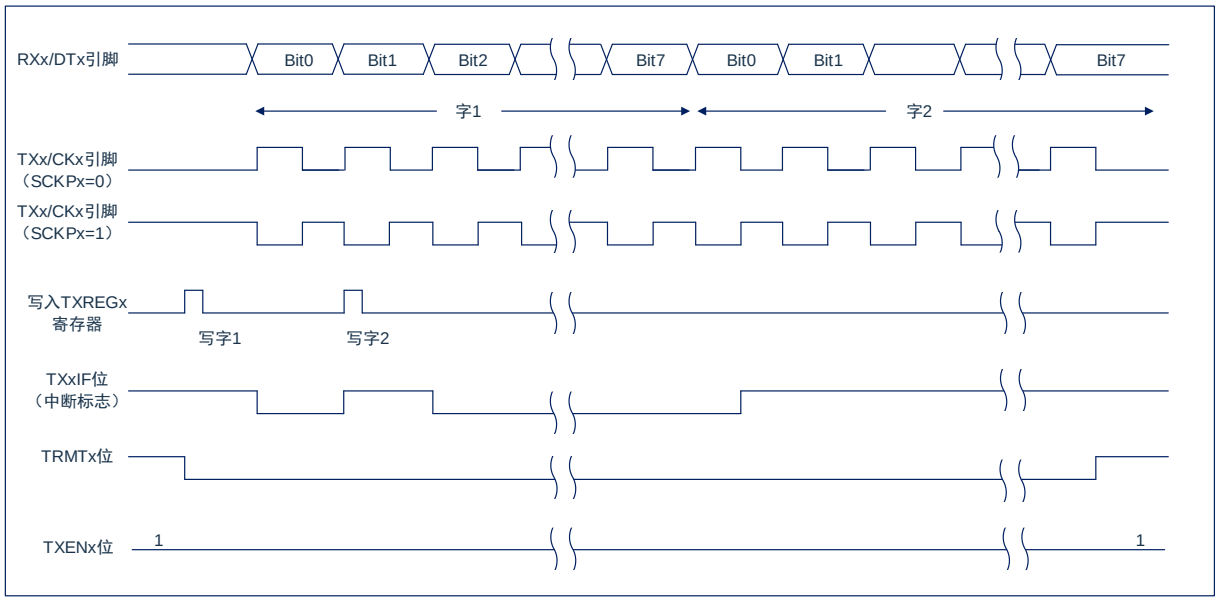


图 12-6：同步发送

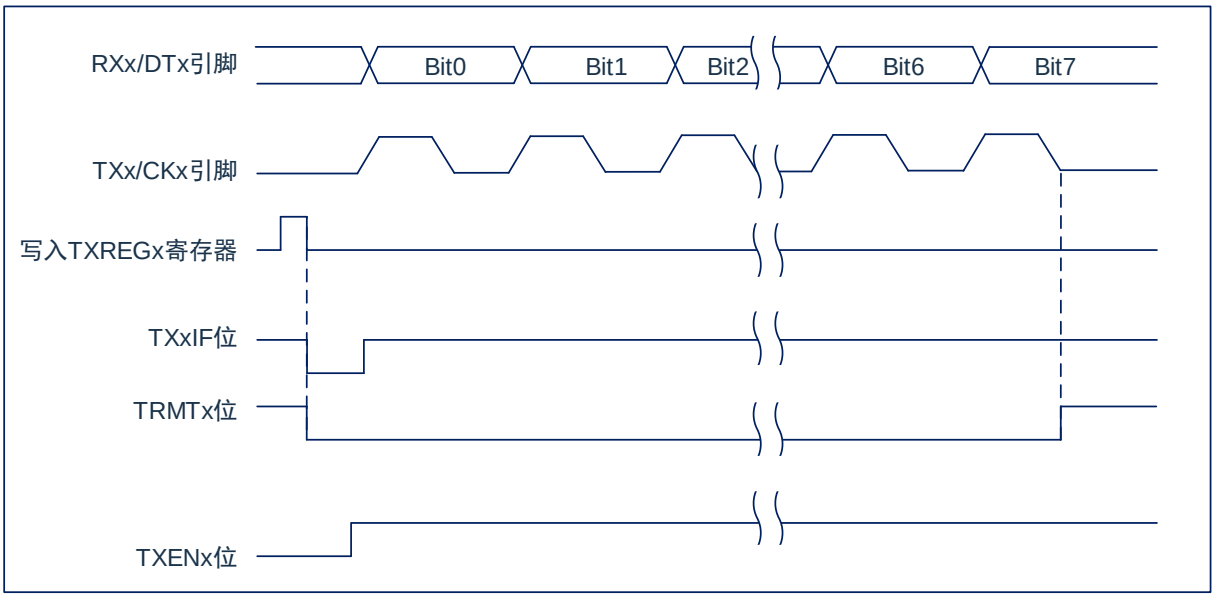


图 12-7：同步发送（通过 TXENx）

12.5.1.5 同步主控接收

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

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

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

12.5.1.6 从时钟

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

12.5.1.7 接收溢出错误

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

12.5.1.8 接收 9 位字符

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

12.5.1.9 同步主控接收设置

1. 初始化 SPBRGx 寄存器，以获得所需的波特率。（注：必须满足 SPBRGx>05H）
2. 将 SYNCx、SPENx 和 CSRCx 位置 1 使能同步主控串行端口。
3. 确保将 CRENx 和 SRENx 位清零。
4. 如果使用中断，将 INTCON 寄存器的 GIE 和 PEIE 位置 1，并将 PIE1 或 PIE2 寄存器的 RCxIE 位也置 1。
5. 如果需要接收 9 位字符，将 RX9ENx 位置 1。
6. 将 SRENx 位置 1，启动接收，或将 CRENx 位置 1 使能连续接收。
7. 当字符接收完毕后，将 RCxIF 中断标志位置 1。如果允许位 RCxIE 置 1，还会产生一个中断。
8. 读 RCREGx 寄存器获取接收到的 8 位数据。
9. 读 RCSTAx 寄存器以获取第 9 个数据位（使能 9 位接收时），并判断接收过程中是否产生错误。
10. 如果产生溢出错误，清零 RCSTAx 寄存器的 CRENx 位或清零 SPENx 以复位 USARTx 来清除错误。

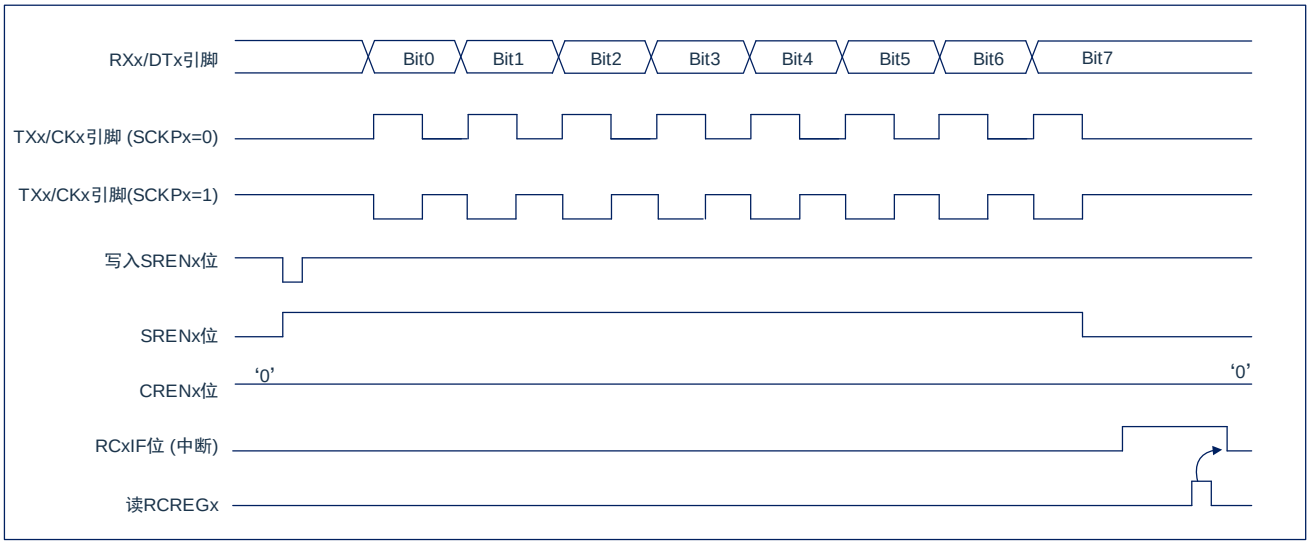


图 12-8：同步接收（主控模式，SRENx）

注：时序图说明了 SRENx=1 时的同步主控模式。

12.5.2 同步从动模式

下列位用来将 USARTx 配置为同步从动操作：

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

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

12.5.2.1 USARTx 同步从动发送

同步主控和从动模式的工作原理是相同的（见章节“同步主控发送”章节。）

12.5.2.2 同步从动发送设置

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

12.5.2.3 USART 同步从动接收

除了以下不同外，同步主控和从动模式的工作原理相同。

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

12.5.2.4 同步从动接收设置

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

13. 比较器（CMP1 和 CMP2）

芯片内置 2 个比较器 CMP1 和 CMP2，两个比较器的功能和性能一样，以下描述 x 的值为 1，2。

13.1 CMPx 的功能框图

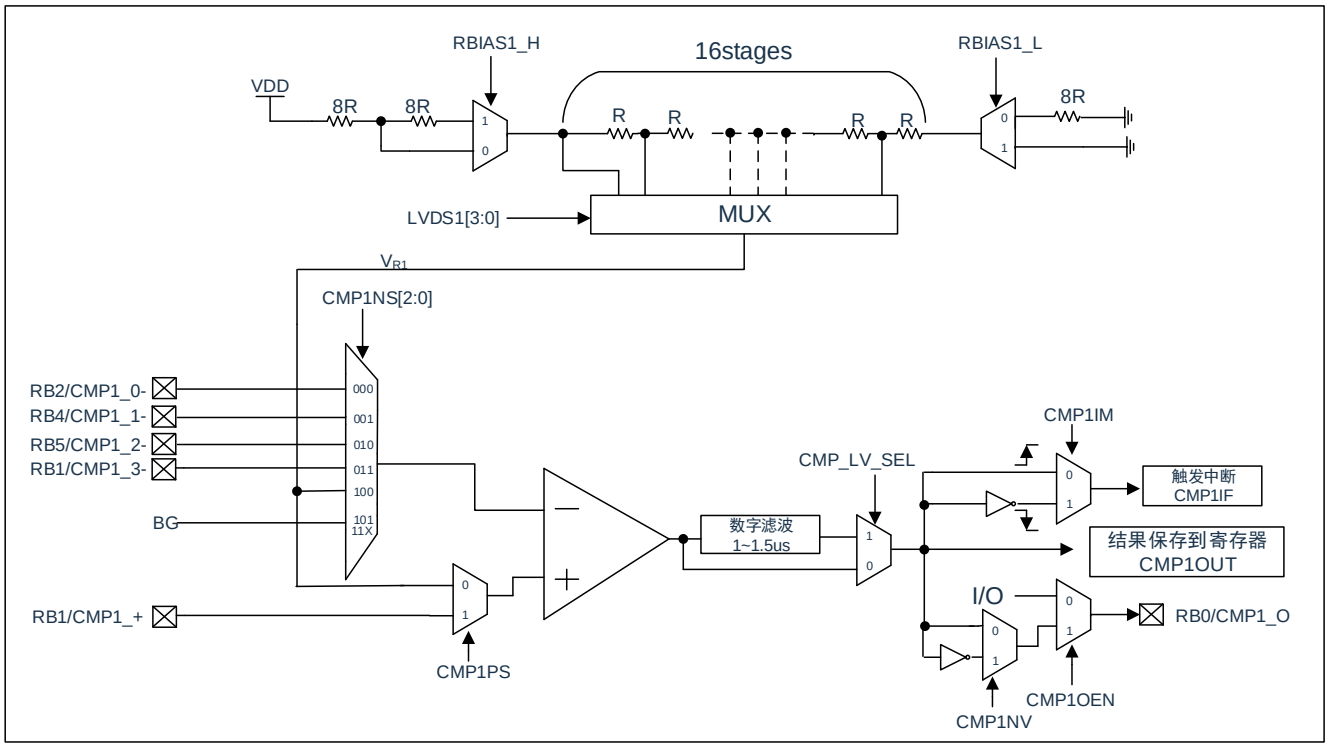


图 13-1：CMP1 的功能框图

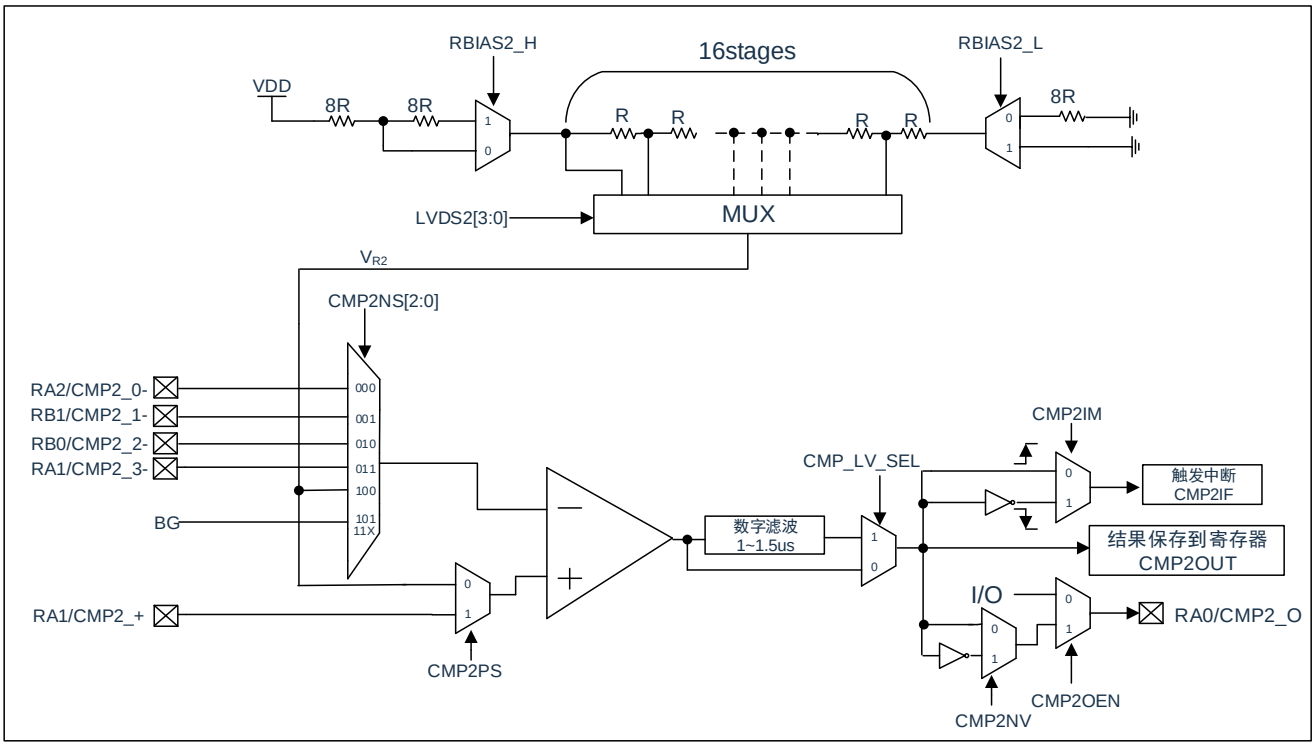


图 13-2：CMP2 的功能框图

13.2 CMPx 功能特性

- 比较器失调电压≤±13mv；
- 输入共模电压范围：0V~VDD-1.3V；
- 内置 1 个电阻分压模块，参考电压为 VDD；
- 比较器结果可选上升沿或下降沿触发中断；
- 比较器结果可选择从 IO 口输出，且支持取反输出。

13.3 CMPx 相关功能

13.3.1 CMPx 功能描述

CMPx 正端输入可通过配置 CMPxCON0 寄存器的 CMPxPS 位来选择 CMPx_+端口或者内部电阻分压输出信号 V_{Rx}；负端输入可通过配置 CMPxCON0 寄存器的 CMPxNS<2:0>位来选择 CMPx_m-端口、内部电阻分压输出信号 V_{Rx} 或者 1.2V BG 电压。当 CMPx 正端输入电压大于负端输入电压时，CMPx 经过数字滤波后输出 1；反之，如果 CMPx 正端输入电压小于负端输入电压，则 CMPx 经过数字滤波后输出 0。

注： CMPx_m-表示 CMPx_0-、CMPx_1-、CMPx_2-、CMPx_3-。

13.3.2 CMPx 内部电阻分压输出

CMPx 内部集成了一个电阻分压模块，参考电压为 VDD。可通过配置 CMPxCON1 寄存器的 RBIASx_H、RBIASx_L、LVDSx<3:0>等控制位的值来获得不同的电阻分压输出 V_{Rx}，根据 RBIASx_H、RBIASx_L 这两个位的不同值，V_{Rx} 有如下 4 种计算公式：

RBIASx_H	RBIASx_L	V_{Rx} 计算公式
0	0	$V_{Rx}=\frac{1}{4}*VDD+\frac{n+1}{32}*VDD$
0	1	$V_{Rx}=\frac{n+1}{24}*VDD$
1	0	$V_{Rx}=\frac{1}{5}*VDD+\frac{n+1}{40}*VDD$
1	1	$V_{Rx}=\frac{n+1}{32}*VDD$

注：n 为 LVDSx<3:0>的值，即 n=0，1，2……14，15。

13.3.3 CMPx 监测电源电压

根据 CMPx 功能框图和 13.3.2 里的公式可知，当 CMPx 的负端选择 BG 1.2V，正端选择内部电阻分压输出 V_{Rx} 时，可以通过 CMPx 来检测电源电压，当电源电压低于设定值时 CMPx 输出 0，电源电压高于设定值时 CMPx 输出 1，通过配置 RBIASx_H、RBIASx_L、LVDSx<3:0>的值可设定不同的电压监测点，具体如下表：

RBIASx_H	RBIASx_L	LVDSx[3:0]	监测值(V)	RBIASx_H	RBIASx_L	LVDSx[3:0]	监测值(V)	RBIASx_H	RBIASx_L	LVDSx[3:0]	监测值(V)
0	1	0101	4.80	0	0	0100	2.95	1	0	1101	2.18
1	0	0010	4.36	0	1	1001	2.88	0	0	1001	2.13
0	0	0000	4.27	1	0	1000	2.82	1	0	1110	2.09
0	1	0110	4.11	0	0	0101	2.74	0	1	1101	2.06
1	0	0011	4.00	1	0	1001	2.67	0	0	1010	2.02
0	0	0001	3.84	0	1	1010	2.62	1	0	1111	2.00
1	0	0100	3.69	0	0	0110	2.56	-	-	-	-
0	1	0111	3.60	1	0	1010	2.53	-	-	-	-
0	0	0010	3.49	0	0	0111	2.40	-	-	-	-
1	0	0101	3.43	1	0	1100	2.29	-	-	-	-
0	0	0011	3.20	0	0	1000	2.26	-	-	-	-
1	0	0111	3.00	0	1	1100	2.22	-	-	-	-

13.3.4 CMPx 的中断使用

若要使用 CMPx 的中断功能则可以通过以下配置步骤来开启：

- 配置寄存器 CMPxCON0 的 CMPxPS 位选择正端输入；
- 配置寄存器 CMPxCON0 的 CMPxNS<2:0>位选择负端输入；
- 配置寄存器 CMPxCON1 的 CMPxIM 位选择上升沿或者下降沿触发中断；
- 配置寄存器 CMPxCON0 的 CMPxEN 位使能比较器；
- 延时等待 10us；
- 清零 PIR1 或 PIR2 寄存器中的 CMPxIF 位；
- 置 1 PIE1 或 PIE2 寄存器的 CMPxIE 位，使能比较器中断；
- 置 1 INTCON 寄存器的 PEIE、GIE 位，开启外设中断和全局中断。

13.3.5 CMPx 中断休眠唤醒

CMPx 中断可将芯片从休眠模式下唤醒，具体程序配置可以参考以下 CMP1 的例程：

例：CMP1 中断休眠唤醒程序

```
SLEEP_MODE：
        LDIA    B'00000110'
        LD      TRISB,A         ;将 RB1/CMP1_+,RB2/CMP1_0-配置为输入口
        ...                     ;关闭其它功能
        LDIA    00H
        LD      CMP1CON0,A      ;CMP1NS<2:0>=000，负端选择 RB2/CMP1_0-
        SETB    CMP1CON0,6      ;CMP1PS=1，正端选择 RB1/CMP1_+
        SETB    CMP1CON1,6      ;AN1_EN=1，将 CMP1_+、CMP1_0-设为模拟口,以便降低休眠功耗
        SETB    CMP1CON1,7      ;CMP1IM=1，选择下降沿触发中断
        SETB    CMP1CON0,7      ;使能 CMP1
        CALL    DELAY10US       ;使能后延时等待比较器输出稳定
        CLRB    PIR1,5          ;清零 CMP1IF（必须）
        SETB    PIE1,5          ;使能 CMP1 中断
        SETB    INTCON,6        ;允许外设中断
        SETB    INTCON,7        ;开启总中断使能，唤醒后程序跳转到中断向量地址 0004H 执行
        CLRWDT                  ;清零 WDT
        STOP                    ;执行 STOP 指令
        NOP
        NOP
```

13.3.6 CMPx 结果输出引脚配置

CMPx 的输出经过数字滤波后，通过读寄存器 CMPxCON0 的 CMPxOUT 位得到当前的比较结果；还可以通过以下的配置步骤将比较结果输出到 CMPx_O 引脚：

- 将 CMPx_O 引脚配置为输出口；
- 配置寄存器 CMPxCON0 的 CMPxNV 位来选择正向输出或反向输出；
- 将寄存器 CMPxCON0 的 CMPxOEN 位置 1 来使能 CMPxOUT 输出到 CMPx_O 引脚。

13.4 CMPx 相关寄存器

CMPx 控制寄存器 CMPxCON0

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CMPxCON0	CMPxEN	CMPxPS	CMPxNS2	CMPxNS1	CMPxNS0	CMPxNV	CMPxOUT	CMPxOEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
复位值	0	0	0	0	0	0	0	0

Bit7 CMPxEN: CMPx使能位
1= 使能CMPx
0= 禁止CMPx
Bit6 CMPxPS: CMPx正端输入选择位
1= CMPx_+端口电压
0= VDD经过内部电阻分压后的电压
Bit5~Bit3 CMPxNS<2:0>: CMPx负端输入选择位
000= CMPx_0-端口电压
001= CMPx_1-端口电压
010= CMPx_2-端口电压
011= CMPx_3-端口电压
100= VDD经过内部电阻分压后的电压
101= BG
11x= BG
Bit2 CMPxNV: CMPx_O端口输出取反控制位
1= CMPxOUT在CMPx_O端口取反输出
0= CMPxOUT在CMPx_O端口正常输出
Bit1 CMPxOUT: CMPx结果位
Bit0 CMPxOEN: CMPx_O端口输出使能位
1= 使能CMPx_O端口输出CMPxOUT
0= 禁止CMPx_O端口输出CMPxOUT

CMPx 控制寄存器 CMPxCON1

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CMPxCON1	CMPxIM	ANx_EN	RBIASx_H	RBIASx_L	LVDSx<3:0>			
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7 CMPxIM: CMPx中断触发边沿选择
1= CMPx输出的下降沿触发中断
0= CMPx输出的上升沿触发中断
Bit6 ANx_EN: 模拟口使能位，使能CMPx_+、CMPx_m-的模拟口功能
1= 模拟口
0= 数字口
Bit5 RBIASx_H: 具体用法参考CMPx的功能框图
Bit4 RBIASx_L: 具体用法参考CMPx的功能框图
Bit3~Bit0 LVDSx<3:0>: 内部电阻分压比选择位

注：ANx_EN 位只对被选作比较器功能的 IO 端口有效，另外通过 ANSEL0、ANSEL1 寄存器也可以使能 CMPx_+、CMPx_m-的模拟口功能。

14. 模数转换（ADC）

14.1 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 参考电压可选择内部 LDO 或 VDD。ADC 在转换完成之后可以产生一个中断。

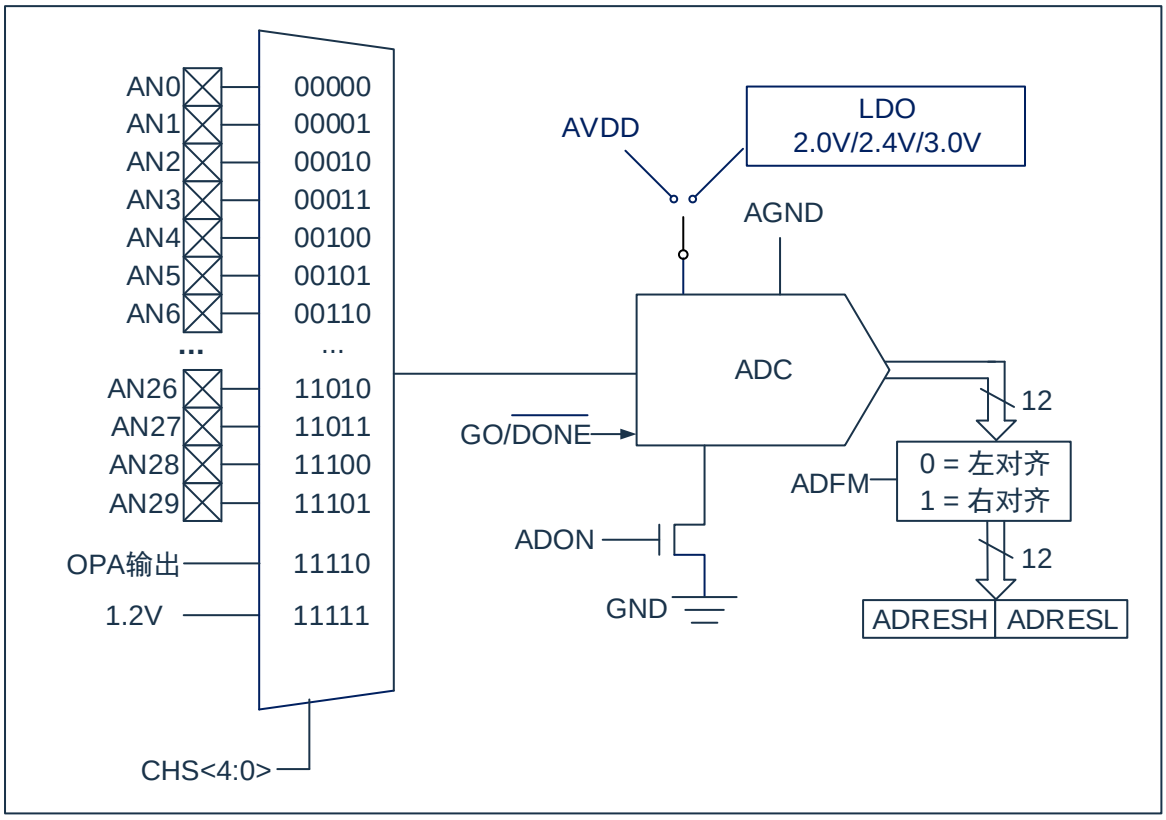


图 14-1：ADC 框图

14.2 ADC 配置

配置和使用 ADC 时，必须考虑如下因素：

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

14.2.1 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。

14.2.2 通道选择

由 ADCON0 和 ADCON1 寄存器的 CHS 位决定将哪个通道连接到采样保持电路。

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

14.2.3 ADC 内部基准电压

芯片内置基准电压，需要检测该基准电压时，需把设置 CHS<4:0>位为 11111。

14.2.4 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 2.0V/2.4/3.0V。

当选择内部参考电压时，需要选择较慢的转换时钟，参考转换时钟章节。

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

14.2.5 转换时钟

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

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

完成一位转换的时间定义为 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。
不同参考电压和不同 VDD 时，需要参考以下表格设置合理的分频。

参考电压（V）	工作电压（V）	最快分频设置	转换时间(us)
		F_{HSI}=16M	
VDD	4.0~5.5	F_{HSI}/16	16
VDD	2.7~4.0	F_{HSI}/32	32
3.0	4.0~5.5	F_{HSI}/32	32
3.0	3.3~4.0	F_{HSI}/64	64
2.4	4.0~5.5	F_{HSI}/32	32
2.4	2.7~4.0	F_{HSI}/64	64
2.0	4.0~5.5	F_{HSI}/64	64
2.0	2.7~4.0	F_{HSI}/128	128

14.2.6 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。

14.2.7 结果格式化

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

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

14.3 ADC 工作原理

14.3.1 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。

14.3.2 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。

14.3.3 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。

14.3.4 ADC 在休眠模式下的工作原理

ADC 模块不能在休眠模式下工作。

14.3.5 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 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    TRISA,0         ;设置 PORTA.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
```

14.4 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 和 ADCON1，数据寄存器 ADRESH 和 ADRESL。

AD 控制寄存器 ADCON0(95H)

95H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	ADON
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

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

Bit5~Bit2 CHS<3:0>: 模拟通道选择位低四位与CHS4组成五位通道选择
CHS<4:0>:
00000= AN0
00001= AN1
00010= AN2
00011= AN3
00100= AN4
00101= AN5
…
11100= AN28
11101= AN29
11110= OPA输出
11111= 1.2V（固定参考电压）
其他= 保留

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

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

AD 控制寄存器 ADCON1(96H)

96H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON1	ADFM	CHS4	---	---	---	LDO_EN	LDO_SEL1	LDO_SEL0
R/W	R/W	R/W	---	---	---	R/W	R/W	R/W
复位值	0	0	---	---	---	0	0	0

Bit7 ADFM: A/D转换结果格式选择位
1= 右对齐
0= 左对齐
Bit6 CHS4: 通道选择位
Bit5~Bit3 未用
Bit2 LDO_EN: 内部参考电压使能位
1= 使能ADC内部LDO参考电压
当选择内部LDO作参考电压时，ADC最大有效精度为8位
0= VDD作为ADC参考电压
Bit1~Bit0 LDO_SEL<1:0>: 参考电压选择位
11= 3.0V
10= 2.4V
0x= 2.0V

AD 数据寄存器高位 ADRESH(99H)，ADFM=0

99H	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(98H)，ADFM=0

98H	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(99H)，ADFM=1

99H	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(98H)，ADFM=1

98H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2
R/W	R	R	R	R	R	R	R	R
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0　　ADRES<9:2>:　ADC结果寄存器位

12位转换结果的第2-9位

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

15. 程序 EEPROM 和程序存储器控制

15.1 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，在所有地址范围内都是只读的；器件具有 256 字的程序 EEPROM，地址范围为 000h 到 0FFh，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 6 个 SFR 寄存器用于访问这些存储器：

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

当访问程序 EEPROM 时，EEDAT 和 EEDATH 寄存器形成一个双字节字用于存放 16 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取的 13 位程序存储器单元地址。

程序存储器允许以字为单位读取。程序 EEPROM 允许字读写。字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，只可读；
   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~5.5V，写电流为 5mA@VDD=5V.

15.2 相关寄存器

15.2.1 EEADR 和 EEADRH 寄存器

EEADR 和 EEADRH 寄存器能寻址最大 256 字的程序 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。

15.2.2 EECON1 和 EECON2 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。程序存储器是只读的。

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 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。

EECON2 不是物理寄存器。读 EECON2 得到的是全 0。

EECON2 寄存器仅在执行程序 EEPROM 写序列时使用。

EEPROM 数据寄存器 EEDAT(8FH)

8FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EEDAT	EEDAT7	EEDAT6	EEDAT5	EEDAT4	EEDAT3	EEDAT2	EEDAT1	EEDAT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	X	X	X	X	X	X	X	X

Bit7~Bit0　　EEDAT<7:0>:　要从程序EEPROM中读取或向程序EEPROM写入数据的低8位，或者要从程序存储器中读取数据的低8位

EEPROM 地址寄存器 EEADR(91H)

91H	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EEPROM读/写操作的地址的低8位，或程序存储器读操作的地址的低8位

EEPROM 数据寄存器 EEDATH(90H)

90H	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(92H)

92H	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>:　指定程序存储器读操作的高5位地址。

EEPROM 控制寄存器 EECON1(8DH)

8DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
EECON1	EEPGD	---	---	---	WRERR	WREN	WR	RD
R/W	R/W	---	---	---	R/W	R/W	R/W	R/W
复位值	0	---	---	---	X	0	0	0

Bit7　　EEPGD:　程序/程序EEPROM选择位
　　1=　操作程序存储器
　　0=　操作程序EEPROM
Bit6~Bit4　　未用。
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）
　　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，但不能清零）
　　0=　写周期完成
Bit0　　RD:　读控制位
　　1=　启动存储器读操作（由硬件清零RD，用软件只能将RD位置1，但不能清零）
　　0=　不启动存储器读操作

15.3 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 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 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      RDATAL,A
        LD      A,EEDATH
        LD      RDATAH,A
EEPDATA_READ_BACK:
        RET
```

15.4 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 和 EEDATH 寄存器。然后用户必须按特定顺序开始写入每个字。

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，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。

注：在写程序 EEPROM 期间，CPU 会停止工作，停止时间为 T_{EEPROM}。

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,WDATAL        ;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
        CLR     EECON1
        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
```

15.5 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
```

15.6 写程序存储器

程序存储器是只读的，不可写。

15.7 程序 EEPROM 操作注意事项

15.7.1 关于程序 EEPROM 的烧写时间

程序 EEPROM 烧写时间是大致固定的，烧写不同的数据需要的时间约为 4.6ms，并且在烧写期间 CPU 停止工作，程序需要做好相关处理。

15.7.2 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。

15.7.3 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

16. 触摸按键

16.1 触摸按键模块概述

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

技术参数：

- 1-15 个按键可选
- 灵敏度可通过外接电容调节
- 有效触摸反应时间<100ms

芯片使用 16Bit 高精度的 CDC（数字电容转换器）、IC 检测感应盘（电容传感器）上的电容变化来识别人手指的触摸动作。

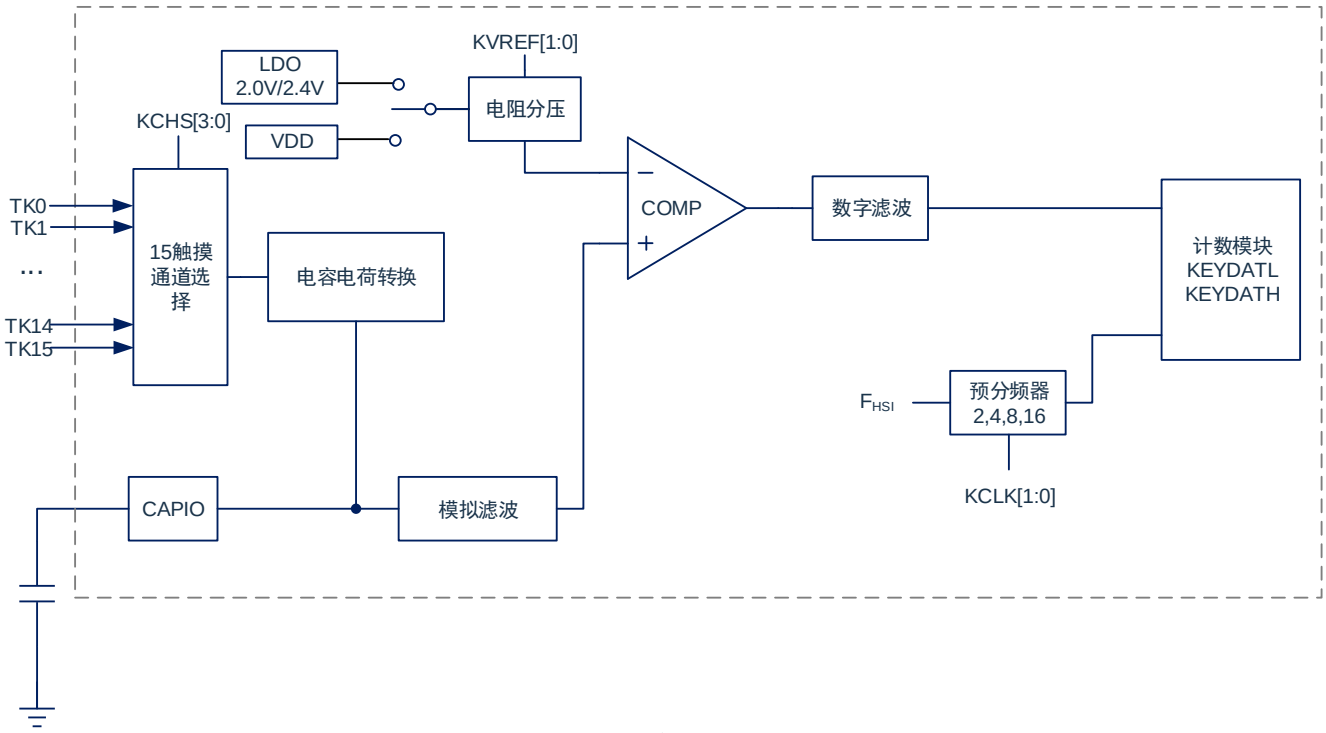


图16-1：内部电路框图

16.2 触摸按键相关寄存器

有 5 个寄存器与触摸按键相关，分别是触摸控制寄存器 KEYCON0、KEYCON1、KEYCON2，触摸按键结果寄存器 KEYDATL、KEYDATH。

触摸按键结果寄存器 KEYDATAL(116H)

116H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
KEYDATAL								
R/W	R	R	R	R	R	R	R	R
复位值	0	0	0	0	0	0	0	0

触摸按键结果寄存器 KEYDATAH(117H)

117H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
KEYDATAH								
R/W	R	R	R	R	R	R	R	R
复位值	0	0	0	0	0	0	0	0

摸按键控制寄存器 KEYCON0(111H)

111H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
KEYCON0	KDONE	----	----	----	----	KTOUT	KCAP	KEN
R/W	R	----	----	----	----	R	R/W	R/W
复位值	0	----	----	----	----	0	0	0

Bit7 KDONE: 触摸按键检测结束标志位
0= 转换未结束
1= 转换结束

Bit6~Bit3 保留，需写 0

Bit2 KTOUT: 按键结果计数器溢出标志位
0= 没有溢出
1= 有溢出

Bit1 KCAP: 触摸电容使能位（RB6 管脚）
0= 禁止
1= 使能

Bit0 KEN: 触摸检测启动位；
0= 停止触摸检测，当该位为 0 时，KEYDATAH 和 KEYDATAL 寄存器会自动清零
1= 开始触摸检测，检测过程中须保持为 1

触摸按键控制寄存器 KEYCON1(112H)

112H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
KEYCON1	KVREF<1:0>		KCLK<1:0>		KCHS<3:0>			
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

Bit7~Bit6 KVREF: 触摸按键内部比较器负端电压选择
0x= 0.5 TK_{VREF}
10= 0.6 TK_{VREF}
11= 0.7 TK_{VREF}

Bit5~Bit4 KCLK: 触摸按键时钟 F_{TKDIV} 分频选择
00= $F_{TKDIV}=F_{HSI}/2$
01= $F_{TKDIV}=F_{HSI}/4$
10= $F_{TKDIV}=F_{HSI}/8$
11= $F_{TKDIV}=F_{HSI}/16$

Bit3~Bit0 KCHS<3:0>: 检测通道选择。
0000:选择 TK0 通道
0001:选择 TK1 通道
0010:选择 TK2 通道
0011:选择 TK3 通道
0100:选择 TK4 通道
0101:选择 TK5 通道
0110:未用
0111:选择 TK7 通道
…
1101:选择 TK13 通道
1110:选择 TK14 通道
1111:选择 TK15 通道

触摸按键控制寄存器 KEYCON2(113H)

113H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
KEYCON2	CAP_LVBO<2:0>			TP_EN	TKLDOEN	----	TKLDO_SEL	TKEN
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~Bit5 CAP_LVBO<2:0> 结束标志的数字滤波时间选择(T_{TKDIV}=1/ F_{TKDIV})

000= 滤波 1* T_{TKDIV}

001= 滤波 2* T_{TKDIV}

010= 滤波 3* T_{TKDIV}

011= 滤波 4* T_{TKDIV}

100= 滤波 5* T_{TKDIV}

101= 滤波 6* T_{TKDIV}

110= 滤波 7* T_{TKDIV}

111= 滤波 8* T_{TKDIV}

Bit4 TP_EN: 触摸跳频使能位

0= 不使能触摸跳频

1= 使能触摸跳频

Bit3 TKLDOEN 触摸 LDO 使能位

0= 不使能触摸 LDO，用 VDD 电压作为触摸参考 TK_{VREF}

1= 使能触摸 LDO，用 LDO 电压作触摸参考 TK_{VREF}

Bit2 保留，需写 0

Bit1 TKLDO_SEL 触摸 LDO 电压选择位

0= 触摸 LDO 电压选择 2.0V

1= 触摸 LDO 电压选择 2.4V

Bit0 TKEN 触摸模块总使能位

0= 触摸模块未使

1= 触摸模块使能

16.3 触摸按键模块应用

16.3.1 用查询模式读取“按键数据值”流程

1. 设置灵敏度调节电容口和按键口为输出口且输出 0；
2. 设置 KEYCON2 寄存器 TKEN 位为 1；
3. 设置按键控制寄存器 KEYCON1（包括通道选择、触摸按键检测时钟设置、比较器负端电压设置）；
4. 设置 KEYCON2 寄存器（包括数字滤波选择，跳频选择、触摸 LDO 选择）；
5. 设置按键控制寄存器 KEYCON0（使能触摸电容口）；
6. KEYCON0.0 位 KEN 从 0 到 1 变化，开始检测按键；
7. 判断按键结束标志 KEYCON0.7 位 KDONE 是否为 1；
8. 读取 16 位数据；
9. 结束检测按键：KEN=0；
10. 返回第 3 步继续检测下一个按键。

例：查询模式的触摸按键键值(KEY0)检测程序

```
KEY_START：
        LDIA    00H
        LD      INTCON，A       ;中断关闭
        LDIA    B’00000000’     ;
        LD      TRISB，A        ;
        LDIA    B’00000000’
        LD      PORTB，A
        SETB    KEYCON2，0      ;设置触摸模块使能位有效
        LDIA    B’01010000’
        LD      KEYCON1，A      ;设置比较器负端电压、触摸检测时钟、通道
        LDIA    02H
        LD      KEYCON0，A      ;使能触摸电容口
        SETB    KEYCON0，0      ;开始检测
WAIT：
        SNZB    KEYCON0，7      ;等待检测结束
        JP      WAIT
        LD      A，KEYDATH
        LD      R01，A          ;保存高 8 位结果到用户自定义 RAM 里面
        LD      A，KEYDATL
        LD      R02，A          ;保存低 8 位结果到用户自定义 RAM 里面
        CLRB    KEYCON0，0      ;结束检测
        JP      XXXX            ;转到其它程序
```

16.3.2 判断按键方法

- 判断基础：无键按下---“数据”大；有键按下---“数据”小；
- 当前的值比以前的值小到一定程度，可认为“有键”；
- 在一定时间内，“数据”由大到小变化认为有键，按下。

例：判断有无按键举例

```
K_START:
        LD      A，KOLDH        ;开始判断，先判断高位
        SUBA    KDATAH          ;将新的键值减去旧的键值
        SZB     STATUS，Z
        JP      K_H_SAME        ;高位相等，判断低位
        SZB     STATUS，C
        JP      KNO             ;新值高位比旧值高位大没有按键
        SUBIA   01H
        SNZB    STATUS，C
        JP      KHAVE           ;新值高位比旧值高位小，并且小的值大于等于 2，有键
        LD      A，KDATAL
        SUBA    KOLDL
        SZB     STATUS，C
        JP      KHAVE           ;高位比旧值小 1，低位也小则有键
        SUBIA   0AH
        SZB     STATUS，C
        JP      KHAVE           ;总体比旧值小超过 10 认为有键
        JP      KNO             ;总体比旧值小不超过 10 认为没有键
K_H_SAME：
        LD      A，KDATAL
        SUBA    KOLDL
        SNZB    STATUS，C
        JP      K_NO            ;高位相等，低位比旧的值大没有按键
        SUBIA   0AH
        SZB     STATUS，C
        JP      KNO             ;新值比旧值小 10 个以上才认为有按键
KHAVE：                         ;有按键程序
        ...
        JP      XXXX            ;处理完有按键程序跳转
KNO:                            ;没有按键的处理程序
        ...
        JP      XXXX            ;处理完没有按键程序跳转
```

其中 KOLDH、KOLDL 存放检测到的旧值，KDATAH、KDATAL 存放检测到的新值，这里设定新值比旧值小 10 个值以上才认为有按键，实际应用中应根据具体情况设置该值。

16.4 触摸模块使用注意事项

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

17. I²C 模块

17.1 I²C 模块概述

IIC 模块可以实现所有的主控和从动功能（包括广播呼叫支持），并且用硬件提供起始位和停止位的中断来判断总线何时空闲（多主机功能）。IIC 模块实现了标准模式规范，以及 7 位寻址。

有两个引脚用于数据传输。它们是时钟引脚（SCL）和数据引脚（SDA）。使用 IIC 模块时，用户必须通过相应的 TRIS 位将这些引脚配置为输入引脚。

通过将 IICCON 寄存器的使能位 IICEN 置 1，使能 IIC 模块的功能。

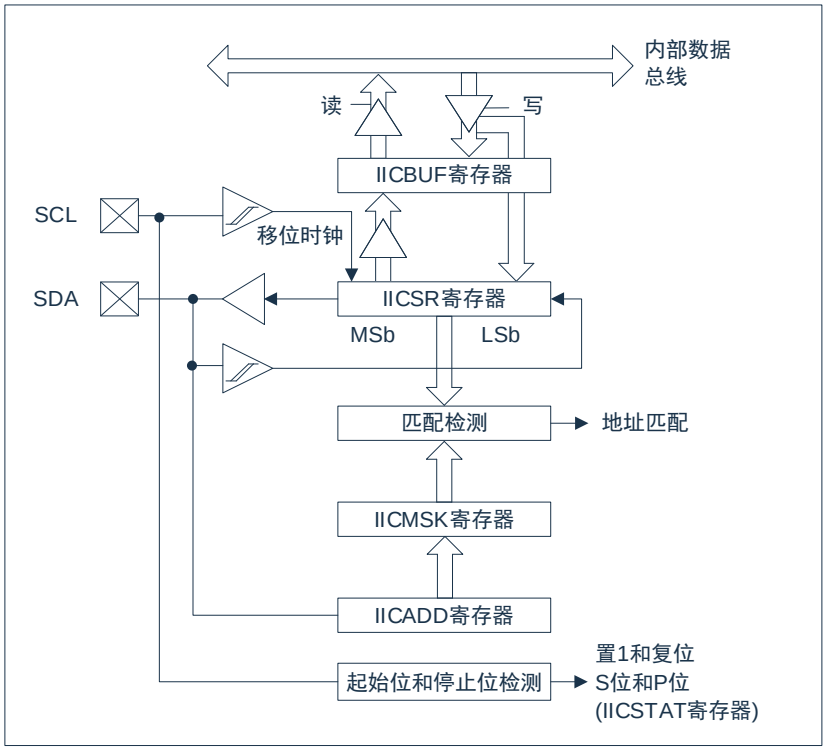


图 17-1：IIC 框图（I²C 模式）

注：I/O 引脚具有连接到 VDD 和 VSS 的保护二极管。

IIC 模块具有 7 个用于 I²C 操作的寄存器。它们是：

- IIC 控制寄存器 1（IICCON）
- IIC 控制寄存器 2（IICCON2）
- IIC 状态寄存器（IICSTAT）
- 串行接收/发送缓冲寄存器（IICBUF）
- IIC 移位寄存器（IICSR）：不能直接访问
- IIC 地址寄存器（IICADD）
- IIC 屏蔽寄存器（IICMSK）

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

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

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

17.2 I2C 相关寄存器说明

IIC 状态寄存器 IICSTAT(192H)

192H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IICSTAT	----	IDLE	D/A	P	S	R/W	IICTOF	BF
读写	----	R	R	R	R	R	W/R	R
复位值	----	0	0	0	0	0	0	0

Bit7 未用

Bit6 IDLE 主控模式空闲位
（仅主控模式有效），所有主控操作都可以通过该位来判断是否结束
1= 总线上没有主控操作
0= 总线上正在进行主控操作

Bit5 D/A 数据/地址位。
1= 表示最后接收或发送的字节是数据。
0= 表示最后接收或发送的字节是地址。

Bit4 P 停止位（禁止IIC模块（IICEN清零）时此位被清零）。
1= 表示最后检测到了停止位（复位时该位为0）。
0= 表示最后未检测到停止位。

Bit3 S 起始位（禁止IIC模块（IICEN 清零）时此位被清零）。
1= 表示最后检测到了起始位（复位时该位为0）。
0= 最后未检测到起始位。

Bit2 R/W 读/写位信息。
该位用来保存在最后一次地址匹配后的R/W位信息。该位仅在从地址匹配开始到下一个起始位、停止位或非ACK位时有效。

在I^2C从动模式下
1= 读。
0= 写。

在I^2C主控模式下
1= 正在发送。
0= 不在进行发送。
此位与SEN、RSEN、PEN、RCEN或ACKEN做逻辑或运算的结果将指示IIC是否在空闲模式下。

Bit1 IICTOF: 从动模式超时标志位
1= 发生了从动模式超时；
0= 未发生从动模式超时。

Bit0 BF 缓冲器满状态位。
接收:
1= 接收完成，IICBUF满。
0= 接收未完成，IICBUF空。
发送:
1 = 数据正在发送（不包括ACK和停止位），IICBUF满。
0 = 数据发送完成（不包括ACK和停止位），IICBUF空。

IIC 控制寄存器 IICCON(190H)

190H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IICCON	IICWCOL	IICOV	IICEN	IICCKP	IICTOS[1:0]		IICM1	IICM0
读写	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 IICWCOL: 写冲突检测位。
主控模式: 1= 在I^2C不满足开始发送数据的条件下，试图对IICBUF寄存器进行写操作。
0= 未发生冲突。
从动模式:
1= 正在发送前一个字时，又对IICBUF寄存器进行写操作（必须由软件清零）。
0= 未发生冲突。

Bit6 IICOV: 接收溢出指示位。(仅在从动接收模式下有限)
1= IICBUF寄存器仍保持前一数据时，又接收到一个新的字节。在发送模式下IICOV位可为任意值（该位必须由软件清零）。
0= 没有溢出。

Bit5 IICEN: IIC使能位（必须正确配置这些引脚为输入）。
1= 使能IIC并将SDA和SCL引脚配置为串行端口引脚。
0= 禁止IIC并将这些引脚配置为I/O端口引脚。

Bit4 IICCKP: 时钟极性选择位。
在I^2C从动模式下: SCK释放控制。
1 = 使能时钟。
0 = 保持时钟线为低电平（时钟延长）（用于确保数据建立时间）。
在I^2C主控模式下: 在此模式下未使用。

Bit3~Bit2 IICTOS[1:0]: IIC从动模式超时选择
00= 禁止IIC从动超时功能；
01= 使能IIC从动超时功能，超时时间为16ms，超时后复位IIC模块；
10= 使能IIC从动超时功能，超时时间为32ms，超时后复位IIC模块；
11= 使能IIC从动超时功能，超时时间为64ms，超时后复位IIC模块；

Bit1~Bit0 IICM<1:0>: IIC模式选择位。
00= I^2C主控模式，时钟= F_{CPU}/ (IICADD+4)；
01= I^2C从动模式，7位地址，不响应起始位和停止位中断；
10= I^2C从动模式，7位地址，并允许起始位和停止位中断；
11= 允许操作IICMSK寄存器

IIC 控制寄存器 IICCON2(191H)

191H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IICCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
读写	R/W	R/W	R	R	R	R	R	R
复位值	0	0	0	0	0	0	0	0

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

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

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

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

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

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

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

Bit0 SEN: 启动条件使能位。
在主控模式下:
1 = 在SDA和SCL引脚启动启动条件。由硬件自动清零。
0 = 启动条件空闲。
在从动模式下:
1 = 从发送和接收都会使能时钟延长（使能时钟延长）。
0 = 禁止时钟延长。

17.3 主控模式

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

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

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

17.3.1 I²C 主控模式支持

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

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

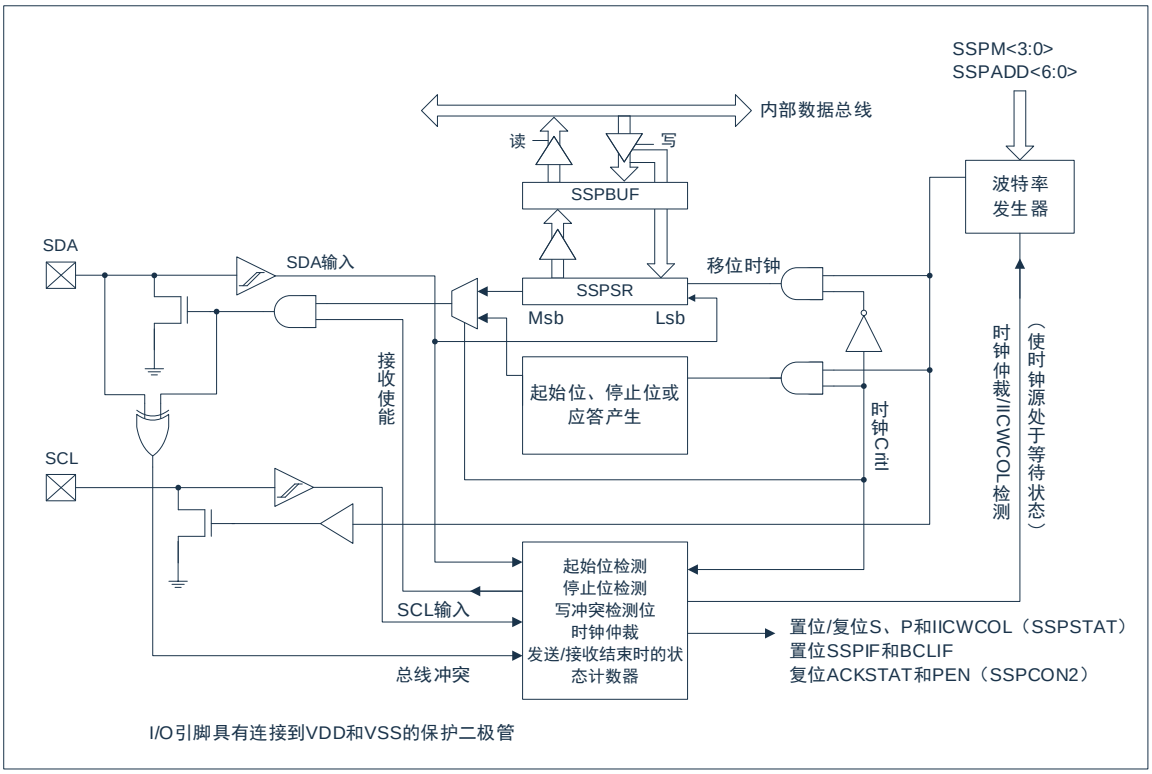


图18-2：IIC框图（I²C™主控模式）

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

17.3.1.1 I²C 主控模式操作

所有串行时钟脉冲和启动/ 停止条件均由主器件产生。停止条件或重复启动条件能结束传输。因为重复启动条件也是下一次串行传输的开始，因此不会释放 I²C 总线。在主控发送器模式下，串行数据通过 SDA 输出，而串行时钟由 SCL 输出。发送的第一个字节包括接收器件的地址（7 位）和读/ 写（R/W）位。在这种情况下，R/W 位将是逻辑 0。串行数据每次发送 8 位。每发送一个字节，会收到一个应答位。启动和停止条件的输出表明串行传输的开始和结束。

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

波特率发生器的重载值位于 IICADD 寄存器的低 7 位。当发生对 IICBUF 的写操作时，波特率发生器将自动开始计数。如果指定操作完成（即，发送的最后一个数据位后面跟着 ACK），内部时钟将自动停止计数，SCL 引脚将保持在其最后的状态。

下面是一个典型的发送事件序列：

- 用户通过将启动使能位 SEN（IICCON2 寄存器）置 1 产生启动条件。
- IICIF 位置 1。在进行任何其他操作前，IIC 模块将等待所需的启动时间。
- 用户将从器件地址装入 IICBUF 进行发送。
- 地址从 SDA 引脚移出，直到发送完所有 8 位为止。
- IIC 模块移入来自从器件的 ACK 位，并将它的值写入 IICCON2 寄存器的 ACKSTAT 位。
- IIC 模块在第 9 个时钟周期的末尾将 IICIF 位置 1，产生一个中断。
- 用户将 8 位数据装入 IICBUF。
- 数据从 SDA 引脚移出，直到发送完所有 8 位为止。
- IIC 模块移入来自从器件的 ACK 位，并将它的值写入 IICCON2 寄存器的 ACKSTAT 位。
- IIC 模块在第 9 个时钟的末尾将 IICIF 位置 1，产生一个中断。
- 用户通过将停止使能位（PEN）位（IICCON2 寄存器）置 1 产生停止条件。
- 一旦停止条件完成，将产生一个中断。

17.3.2 波特率发生器

在 I²C 主控模式下，波特率发生器的重载值位于 IICADD 寄存器的低 7 位（图 17-3）。当装载了该值后，波特率发生器将自动开始计数并递减至 0，然后停止直到下次重载为止。BRG 会在每个指令周期（TCY）中的 Q2 和 Q4 时钟周期上进行两次减计数。在 I²C 主控模式下，会自动重载 BRG。例如，在发生时钟仲裁时，BRG 将在 SCL 引脚采样到高电平时重载（图 17-4）。

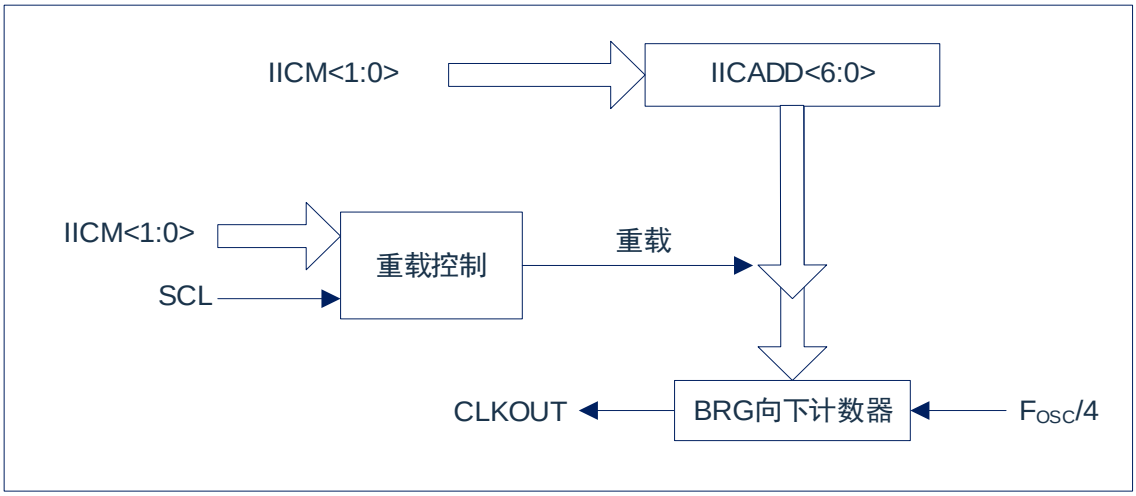


图17-3：波特率发生器框图

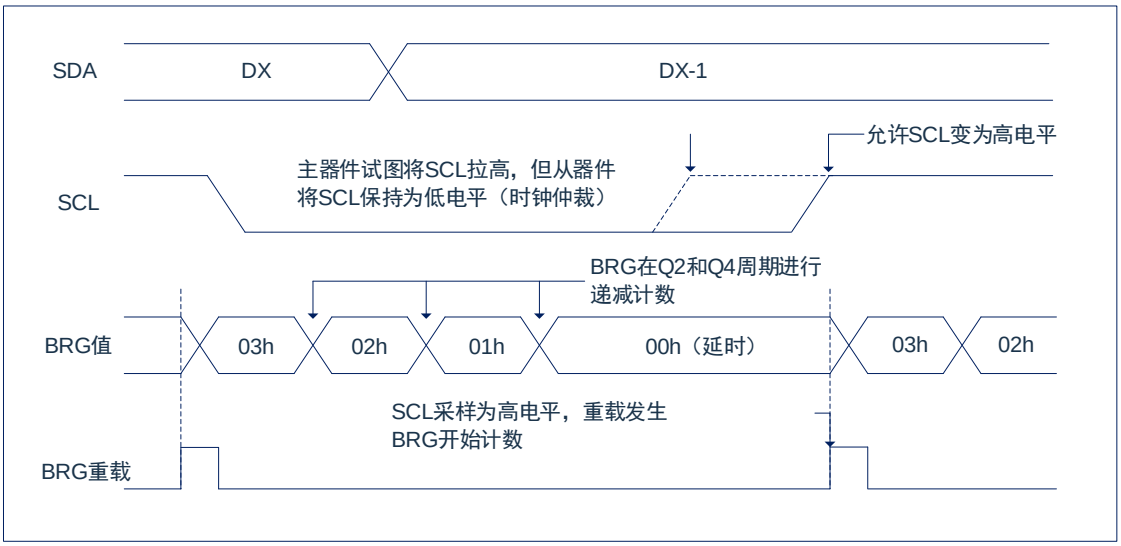


图17-4：带有时钟仲裁的波特率发生器时序

17.3.3 I²C 主控模式发送

发送一个数据字节、一个 7 位地址，都可以直接通过写一个值到 IICBUF 寄存器来实现。该操作将使缓冲器满标志位 BF 置 1，并且波特率发生器开始计数，同时启动下一次发送。在 SCL 的下降沿有效后，地址/数据的每一位将被移出至 SDA 引脚。在一个波特率发生器计满返回计数周期（T_{BRG}）内，SCL 保持低电平。数据应该在 SCL 释放为高电平前保持有效。当 SCL 引脚被释放为高电平时，它将在整个 T_{BRG} 中保持高电平状态。在此期间以及下一个 SCL 下降沿之后的一段时间内，SDA 引脚上的数据必须保持稳定。在第 8 位被移出（第 8 个时钟周期的下降沿）之后，BF 标志位清零，同时主器件释放 SDA。

此时如果发生地址匹配或是数据被正确接收，被寻址的从器件将在第 9 位的时间以一个 ACK 位响应。ACK 的状态在第 9 个时钟周期的下降沿写入 ACKDT 位。主器件接收到应答之后，应答状态位 ACKSTAT 会被清零；如果未收到应答，则该位被置 1。第 9 个时钟之后，IICIF 位会置 1，主控时钟（波特率发生器）暂停，直到下一个数据字节装入 IICBUF 为止，SCL 引脚保持低电平，SDA 保持不变。

在写 IICBUF 之后，地址的每一位在 SCL 的下降沿被移出，直至地址的所有 7 位和 R/W 位都被移出为止。在第 8 个时钟的下降沿，主器件将 SDA 引脚拉为高电平，以允许从器件发出应答响应。在第 9 个时钟的下降沿，主器件通过采样 SDA 引脚来判断地址是否被从器件识别。ACK 位的状态被装入 ACKSTAT 状态位（IICCON2 寄存器）。在发送地址的第 9 个时钟下降沿之后，IICIF 置 1，BF 标志位清零，波特率发生器关闭直到下一次写 IICBUF，且 SCL 引脚保持低电平，允许 SDA 引脚悬空。

17.3.3.1 BF 状态标志

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

17.3.3.2 IICWCOL 状态标志位

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

17.3.3.3 ACKSTAT 状态标志

在发送模式下，当从器件发送应答响应（ACK= 0）时，ACKSTAT 位（IICCON2 寄存器）清零；当从器件没有应答（ACK=1）时，该位置 1。从器件在识别出其地址（包括广播呼叫地址）或正确接收数据后，会发送一个应答。

17.3.4 I²C 主控模式接收

通过编程接收使能位 RCEN（IICCON2 寄存器）使能主控模式接收。波特率发生器开始计数，每次计满返回时，SCL 引脚的状态都发生改变（由高变低或由低变高），且数据被移入 IICSR。第 8 个时钟的下降沿之后，接收使能标志位自动清零，IICSR 的内容装入 IICBUF，BF 标志位置 1，IICIF 标志位置 1，波特率发生器暂停计数，SCL 保持为低电平。此时 IIC 处于空闲状态，等待下一条命令。当 CPU 读缓冲器时，BF 标志位将自动清零。通过将应答序列使能位 ACKEN （IICCON2 寄存器）置 1，用户可以在接收结束后发送应答位。

17.3.4.1 BF 状态标志

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

17.3.4.2 IICWCOL 状态标志

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

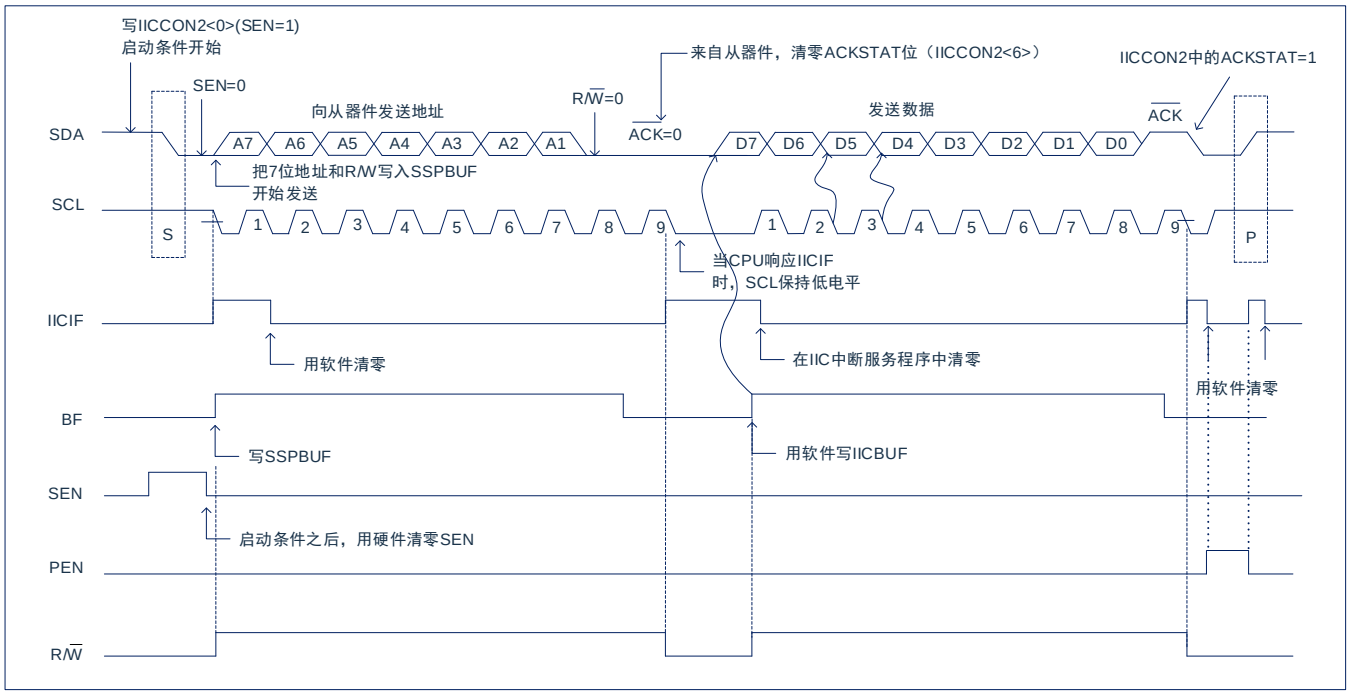


图17-5：I²C™主控模式发送时序

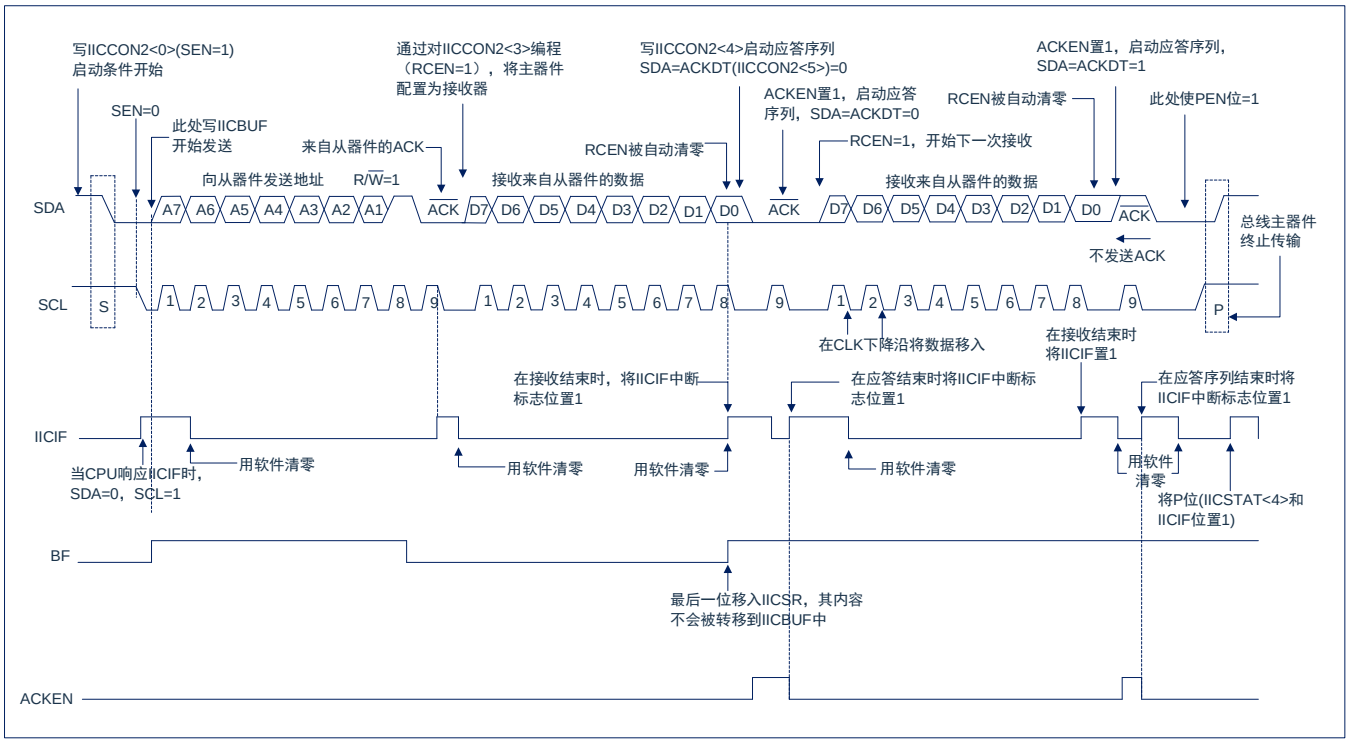


图17-6：I²C™主控模式接收时序

17.3.5 I²C 主控模式启动条件时序

要发起启动条件，用户应将 IICCON2 寄存器的启动条件使能位 SEN 置 1。当 SDA 和 SCL 引脚都采样为高电平时，波特率发生器重新装入 IICADD<6:0>的内容并开始计数。当波特率发生器发生超时（T_{BRG}）时，如果 SCL 和 SDA 都采样为高电平，则 SDA 引脚被驱动为低电平。当 SCL 为高电平时，将 SDA 驱动为低电平就是启动条件，将使 S 位（IICSTAT 寄存器）置 1。随后波特率发生器重新装入 IICADD<6:0>的内容并恢复计数。当波特率发生器超时（T_{BRG}）时，IICCON2 寄存器的 SEN 位将自动被硬件清零。波特率发生器暂停工作，SDA 线保持低电平，启动条件结束。

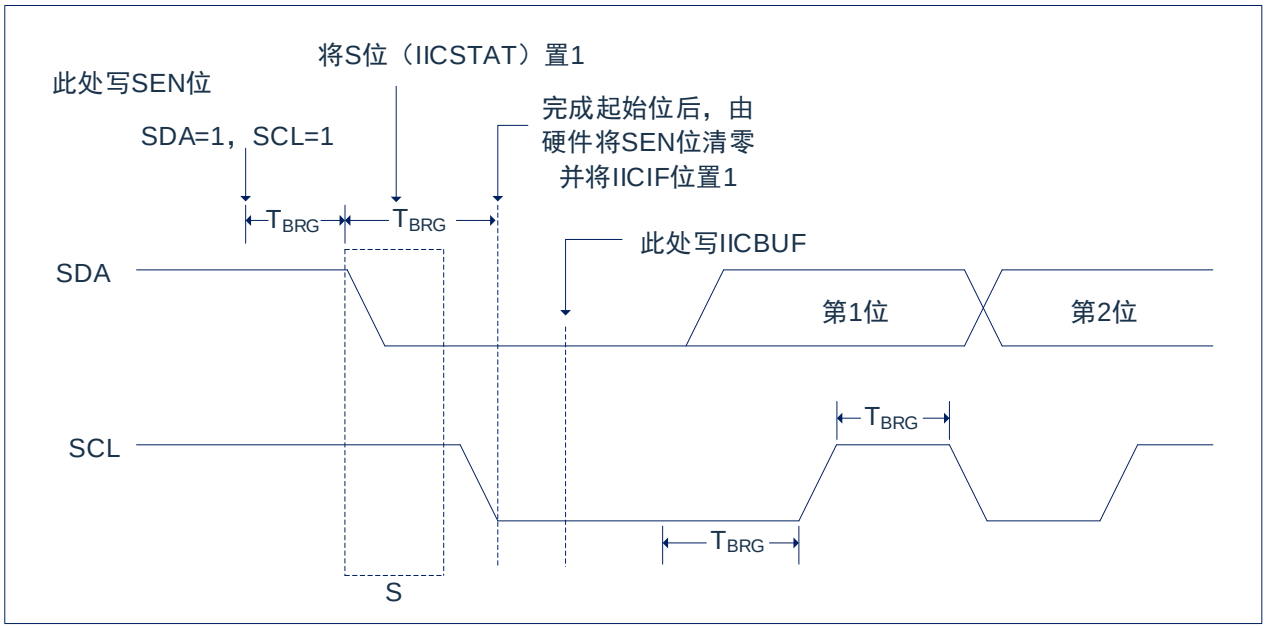


图17-7：第一个启动位时序

17.3.5.1 IICWCOL 状态标志

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

注：由于不允许事件排队，在启动条件结束之前，不能对 IICCON2 的低 5 位进行写操作。

17.3.6 I²C 主控模式重复启动条件时序

将 RSEN 位（IICCON2 寄存器）编程为高电平，并且 I²C 逻辑模块处于空闲状态时，就会产生重复启动条件。当 RSEN 位置 1 时，SCL 引脚被拉为低电平。当 SCL 引脚采样为低电平时，波特率发生器装入 IICADD<6:0>的内容，并开始计数。在一个波特率发生器计数周期（T_{BRG}）内 SDA 引脚被释放（其引脚电平被拉高）。当波特率发生器超时时，如果 SDA 采样为高电平，SCL 引脚将被拉高。当 SCL 引脚采样为高电平时，波特率发生器将被重新装入 IICADD<6:0>的内容并开始计数。SDA 和 SCL 必须在一个计数周期 T_{BRG} 内采样为高电平。随后将 SDA 引脚拉为低电平（SDA = 0）并保持一个计数周期 T_{BRG}，同时 SCL 为高电平。然后 RSEN 位（IICCON2 寄存器）将自动清零，波特率发生器不会重载，SDA 引脚保持低电平。一旦在 SDA 和 SCL 引脚上检测到启动条件，S 位（IICSTAT 寄存器）将被置 1。直到波特率发生器超时后，IICIF 位才会置 1。

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

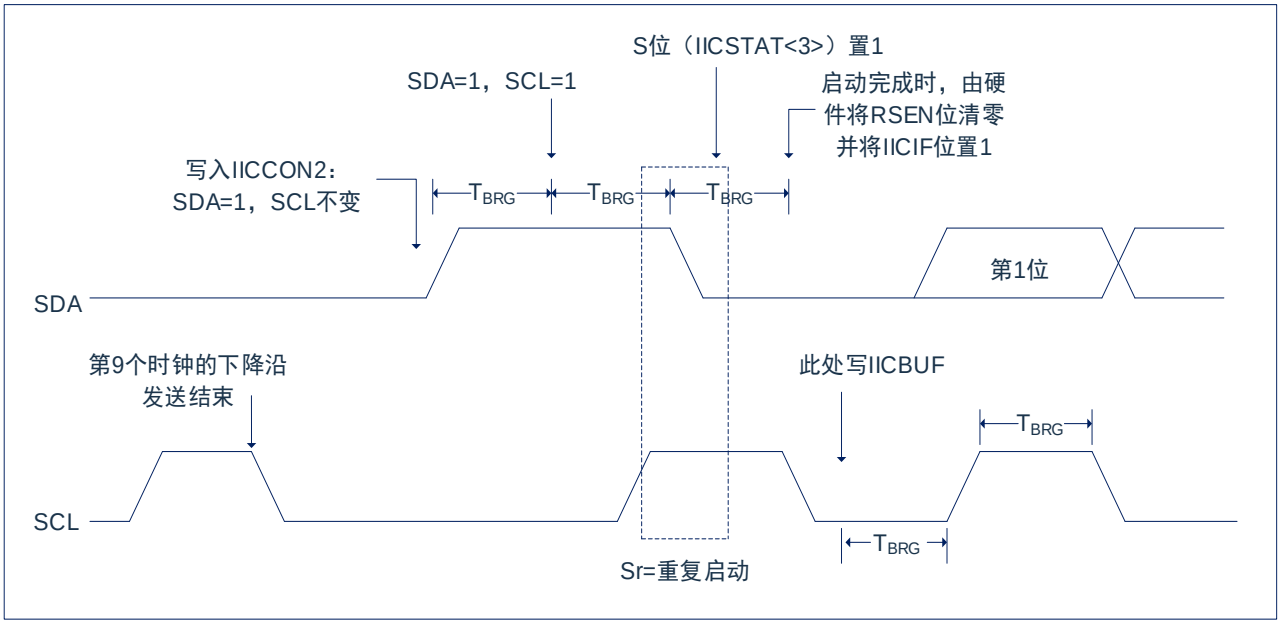


图17-8：重复启动条件时序波形

17.3.6.1 IICWCOL 状态标志

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

注：由于不允许事件排队，在重复启动条件结束之前，不能对IICCON2的低5位进行写操作。

17.3.7 应答序列时序

将应答序列使能位 ACKEN（IICCON2 寄存器）置 1 即可使能应答序列。当该位被置 1 时，SCL 引脚被拉低，应答数据位的内容出现在 SDA 引脚上。如果用户希望产生一个应答，则应该将 ACKDT 位清零；否则，用户应该在应答序列开始前将 ACKDT 位置 1。然后波特率发生器进行一个计满返回周期（T_{BRG}）的计数，随后 SCL 引脚电平被拉高。当 SCL 引脚采样为高电平时（时钟仲裁），波特率发生器再进行一个 T_{BRG} 周期的计数。然后 SCL 引脚被拉低。在这之后，ACKEN 位自动清零，波特率发生器关闭，IIC 模块进入空闲模式。

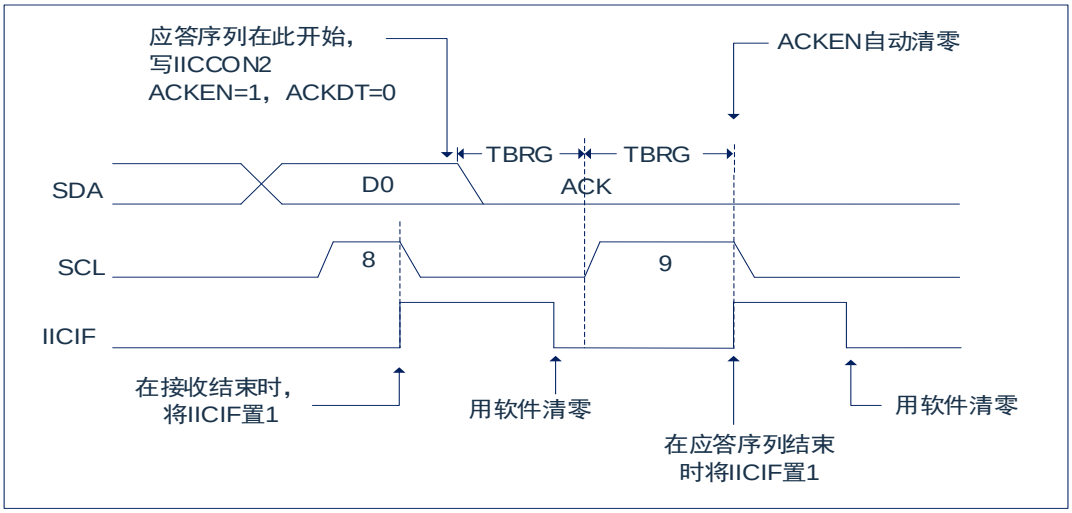


图 17-9：应答序列时序波形

注：T_{BRG}=一个波特率发生器周期。

17.3.7.1 IICWCOL 状态标志位

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

17.3.8 停止条件序列

在接收/发送结束时，通过置 1 停止序列的使能位，PEN（IICCON2 寄存器），SDA 引脚将产生一个停止位。在接收/发送结束时，SCL 引脚在第 9 个时钟的下降沿后保持低电平。当 PEN 位置 1 时，主控器件将 SDA 置为低电平。当 SDA 线采样为低电平时，波特率发生器被重新装入值并递减计数至 0。波特率发生器发生超时时，SCL 引脚被拉到高电平，且一个 T_{BRG}（波特率发生器计满回零）后，SDA 引脚被重新拉到高电平。当 SDA 引脚采样为高电平且 SCL 也是高电平时，P 位（IICSTAT 寄存器）置 1。一个 T_{BRG} 周期后，PEN 位清零且 IICIF 位置 1。

17.3.8.1 IICWCOL 状态标志

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

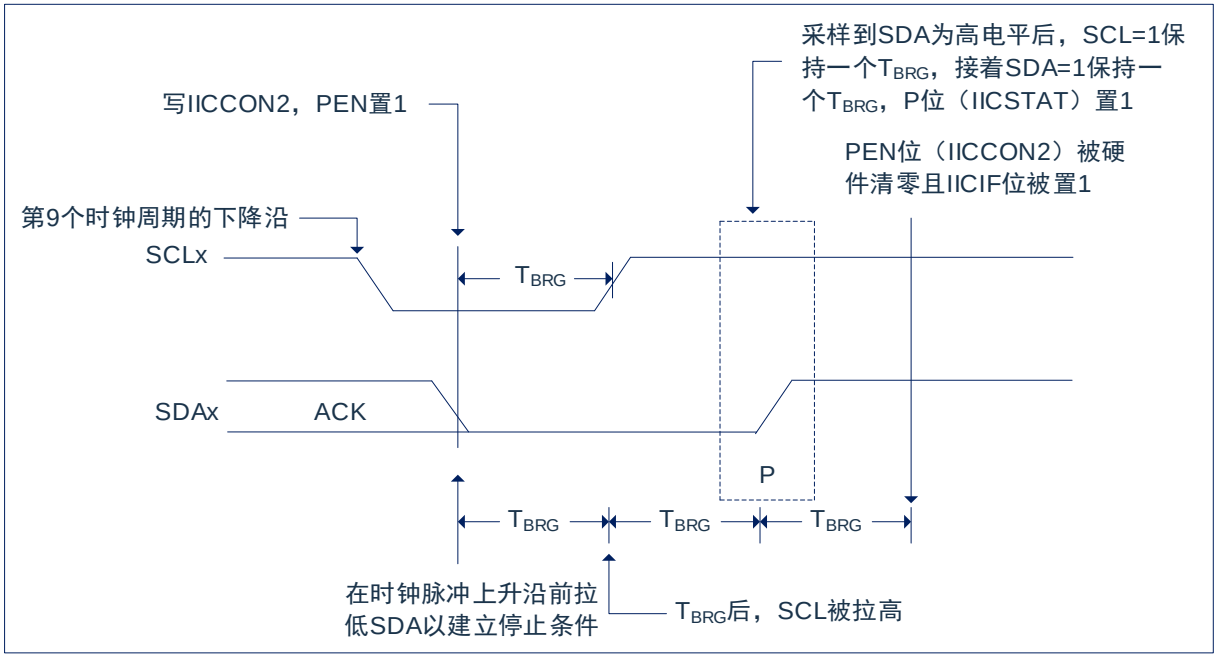


图 17-10：停止条件接收或发送模式

注：T_{BRG}=一个波特率发生器周期。

17.3.9 时钟仲裁

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

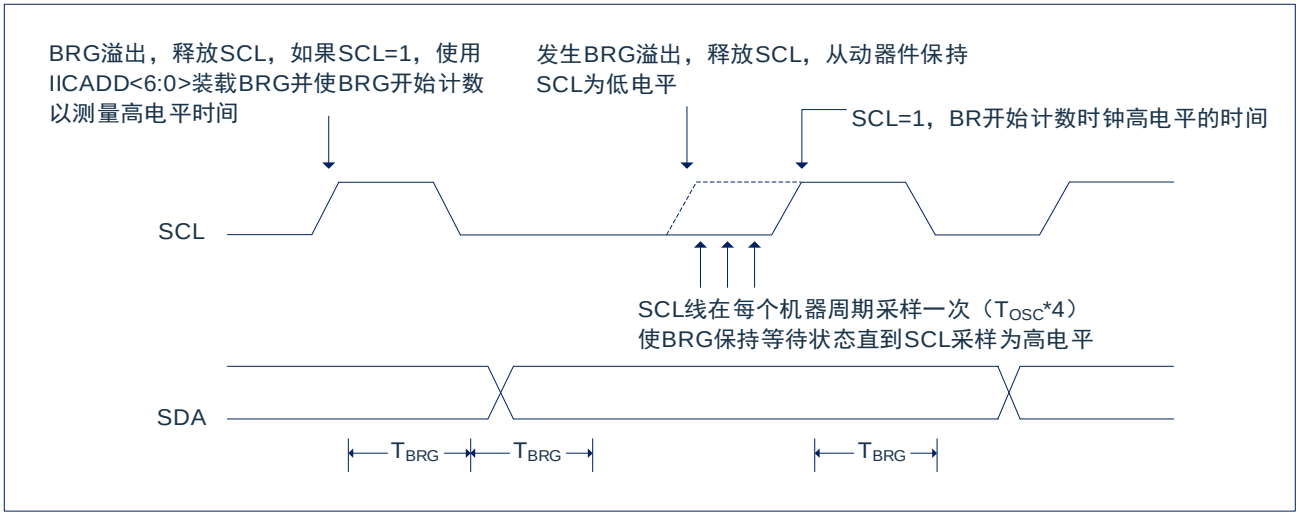


图 17-11：主控发送模式下的时钟仲裁时序

17.3.10 多主机模式

在多主机模式下，通过在检测到启动和停止条件时产生中断可以确定总线何时空闲。停止（P）位和启动（S）位在复位时或禁止 IIC 模块时清零。当 P 位置 1 时，可以取得 I²C 总线的控制权；否则总线处于空闲状态，且 P 位和 S 位清零。当总线忙时，如果出现停止条件，则将产生中断（若允许 IIC 中断）。

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

在以下状态下仲裁可能失败：

- 地址传输
- 数据传输
- 启动条件
- 重复启动条件
- 应答条件

17.3.11 多主机通信、总线冲突与总线仲裁

多主机模式是通过总线仲裁来支持的。当主器件将地址/ 数据位输出到 SDA 引脚时，如果一个主器件通过将 SDA 引脚悬空为高电平以在 SDA 上输出 1，而另一个主器件输出 0，就会发生总线仲裁。如果 SDA 引脚上期望的数据是 1，而实际在 SDA 引脚上采样到的数据是 0，则发生了总线冲突。主器件将把总线冲突中断标志位 BCLIF 置 1，并将 I^2C 端口复位到空闲状态。

如果在发送过程中发生总线冲突，则发送停止，BF 标志位清零，SDA 和 SCL 线被拉高，并且允许对 IICBUF 进行写操作。当执行完总线冲突中断服务程序后，如果 I^2C 总线空闲，用户可通过发出启动条件恢复通信。如果在启动、重复启动、停止或应答条件的进行过程中发生总线冲突，则该条件被中止，SDA 和 SCL 线被拉高，IICCON2 寄存器中的对应控制位清零。当执行完总线冲突中断服务程序后，如果 I^2C 总线空闲，用户可通过发出启动条件恢复通信。主器件将继续监视 SDA 和 SCL 引脚。如果出现停止条件，IICIF 位将被置 1。无论发生总线冲突时发送的进度如何，写 IICBUF 都会从第一个数据位开始发送数据。

在多主机模式下，通过在检测到启动和停止条件时产生中断可以确定总线何时空闲。P 位置 1 时，可以获取 I^2C 总线的控制权，否则总线空闲且 S 和 P 位清零。

17.4 从动模式

在从动模式下，SCL 引脚和 SDA 引脚必须被配置为输入。需要时（如从发送器）IIC 模块将用输出数据改写输入状态。

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

只要满足下列条件之一，IIC 模块就不会产生此 ACK 脉冲：

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

在这种情况下，IICSR 寄存器的值不会载入 IICBUF，但是 PIR2 寄存器的 IICIF 位会置 1。BF 位是通过读取 IICBUF 寄存器清零的，而 IICOV 位是通过软件清零的。

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

17.4.1 寻址

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

- IICSR 寄存器的值被装入 IICBUF 寄存器。
- 缓冲器满标志位 BF 置 1。
- 产生 ACK 脉冲。
- 在第 9 个 SCL 脉冲的下降沿，PIR2 寄存器的 IIC 中断标志位 IICIF 置 1（如果允许中断则产生中断）。

17.4.2 接收

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

当存在地址字节溢出条件时，则不会产生应答脉冲(ACK)。溢出条件是指 BF 位（IICSTAT 寄存器）置 1，或者 IICOV 位（IICCON 寄存器）置 1。每个数据传输字节都会产生一个 IIC 中断。必须用软件将 PIR2 寄存器的中断标志位 IICIF 清零。IICSTAT 寄存器用于确定该字节的状态。

17.4.3 发送

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

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

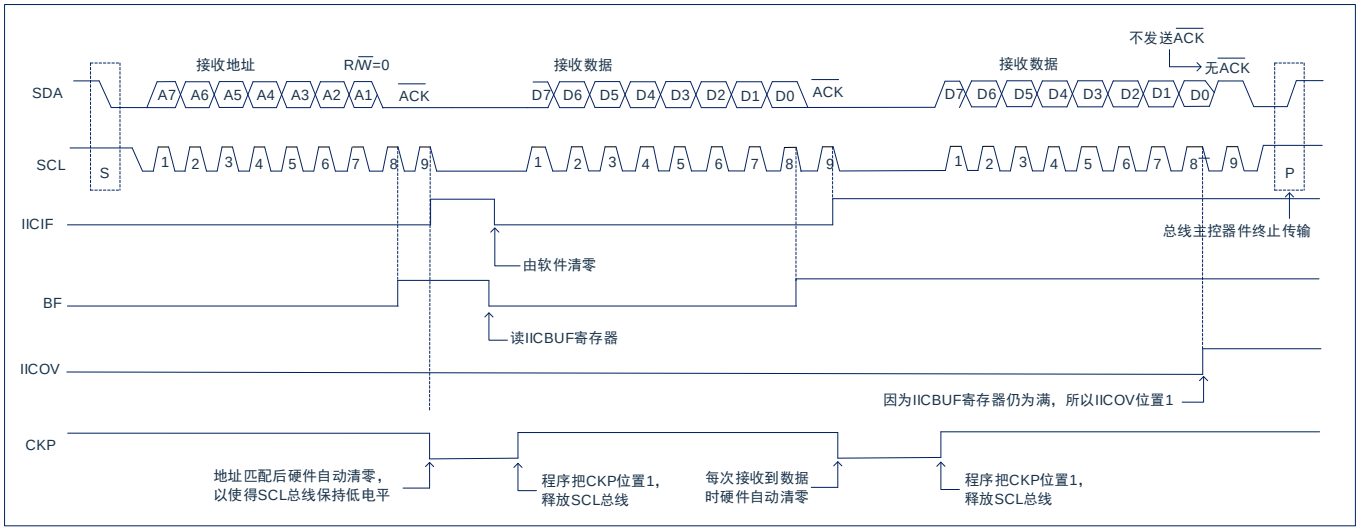


图 17-12：I²C™ 从动模式接收时序

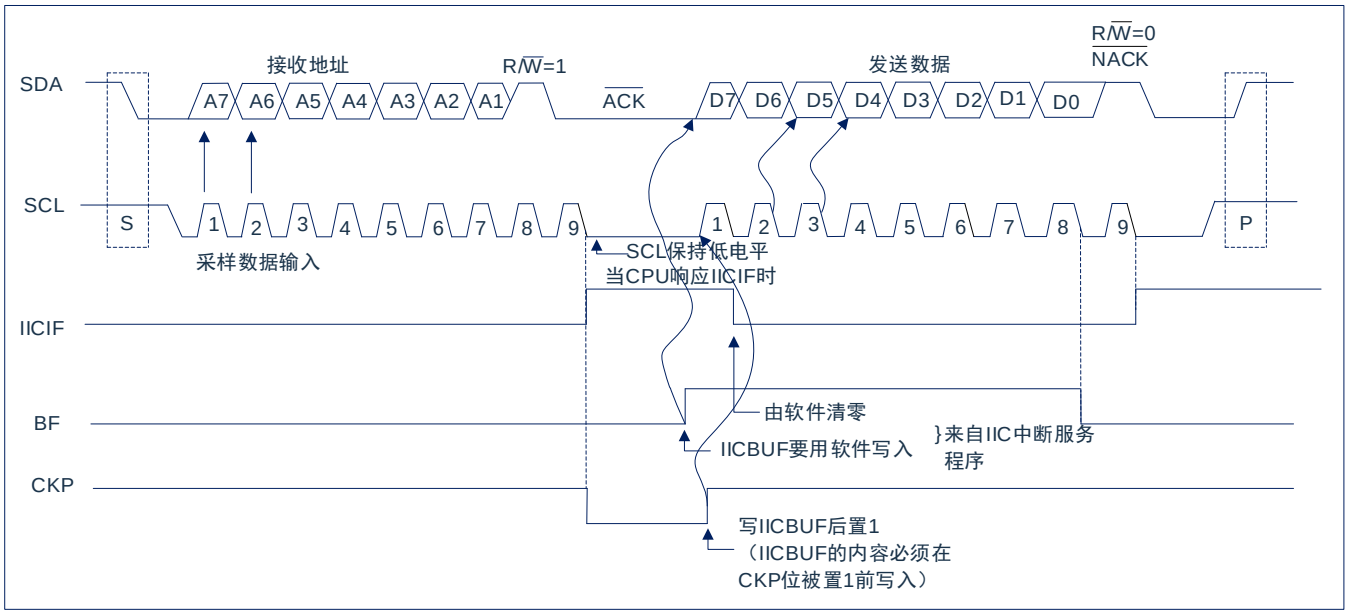


图 17-13：I²C™ 从动模式发送时序

17.4.4 I2C 屏蔽寄存器

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

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

IIC 屏蔽寄存器在以下情况下有效：

- 7 位地址模式：与 A<7:1>进行地址比较。

IIC 屏蔽在接收到地址的第一个（高）字节期间无效。

IICMSK:IIC 屏蔽寄存器(194H)

194H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
IICMSK	MSK7	MSK6	MSK5	MSK4	MSK3	MSK2	MSK1	---
读写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	---
复位值	1	1	1	1	1	1	1	---

Bit7~Bit1　　MSK<7:1>:　屏蔽位。

1=　接收到的地址的Bit n 与IICADD<n>比较以检测I²C的地址匹配情况。

0=　接收到的地址的Bit n 不用于检测I²C的地址匹配情况。

Bit 0　　未用

注：当IICCON位IICM<1:0> = 11时，对IICADD寄存器的读写操作实际上读写的是IICMSK寄存器。

17.4.5 I2C 从动超时保护

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

17.5 休眠模式下的操作

在休眠模式下，I²C 模块无法使用。

17.6 复位的影响

复位会禁止 I²C 模块并终止当前的传输。

18. LCD/LED 驱动模块

芯片内置 LCD/LED 驱动模块，它们共用寄存器。

LCD 模式可驱动 1/2Bias、1/3Bias 的 LCD，最大支持 6COM×24SEG，程序只需要把相关控制位和显示数据设置好，芯片管脚就能自动输出驱动 LCD 的波形（硬件驱动）。

LED 模式为正反推驱动 LED 灯的方式，通过 10 个引脚（LED0~LED9）最大支持 10COM×9SEG，程序只需要把相关控制位和显示数据设置好，芯片管脚就能自动输出驱动 LED 的波形（硬件驱动）。

18.1 LCD/LED 功能使能

将 LCDCON0 的第 7 位 LCDEN 置 1，第 6 位 LEDEN 清 0，允许 LCD 驱动功能；

将 LCDCON0 的第 6 位 LEDEN 置 1，第 7 位 LCDEN 清 0，允许 LED 驱动功能；

将 LCDEN 和 LEDEN 都置 0，关闭 LCD/LED 模块。

18.2 LCD/LED 功能管脚设置

若使能 LCD 驱动功能，必须设置相应的 SEG 口和 COM 口为输入态，即将相应的 TRIS 位置 1；

若使能 LED 驱动功能，必须设置相应的 LED 口为输入态，即将相应的 TRIS 位置 1。

18.3 LCD/LED 功能 COM 口设置

LCD 的 COM 口或者 LED 口设置方式如下：

COMSEL[2:0]	COM 口个数(LCD 模式)	LED 口个数(LED 模式)
000	2	3
001	3	4
010	4	5
011	5	6
100	6	7
101	6	8
110	6	9
111	6	10

设置 COMEN 寄存器，将相应管脚设置为 LCD 功能的 COM 口或者 LED 功能的 LED 口,LED8~LED9；LED0~LED7 口的设置在 SEGEN1 寄存器里。

在 LCD 模式下，用户在使用过程中，如果 COM 口不按照顺序排列，例如将 COM3-COM5 作为 LCD 功能的 COM 口，COM0-COM2 作为普通 I/O 口，可以做以下设置：

- 将 COM 口个数设置成 6 个 COM，COMSEL=100；
- 将 COMEN 寄存器的 COM3EN~COM5EN 置 1，COM0EN~COM2EN 清 0。

此时，COM3~COM5 为 LCD 功能的 COM 口，其输出占空比为 1/6，而 COM0~COM2 可作为普通 I/O 口。

在 LED 模式下，用户在使用过程中，如果 LED 口不按照顺序排列，例如将 LED6~LED9 作为 LED 功能的 LED 口，LED0~LED6 作为普通 I/O 口，可以做以下设置：

- 将 LED 口个数设置成 10 个，COMSEL=111；
- 将 COMEN 寄存器的 COM0EN~COM1EN 置 1，COM2EN~COM5EN 清 0；
- 将 SEGEN1 寄存器的 SEG14EN~SEG15EN 置 1，SEG8EN~SEG13EN 清 0。

此时，LED6~LED9 作为 LED 功能的 LED 口，其输出占空比为 1/10，而 LED0~LED5 可作为普通 I/O 口。

18.4 LCD 功能的 SEG 口设置

使能 LCD 功能的 SEG 口必须满足以下条件：

1. 设置 LCD 功能相应管脚为输入态；
2. 在 SEGEN0、SEGEN1、SEGEN2 寄存器中设置相应管脚为 LCD 驱动功能。

18.5 LED 功能的 LED 口设置

1. 设置 LED 功能相应管脚为输入态；
2. 在 SEGEN1 寄存器和 COMEN 寄存器的 COM0EN~COM1EN 位中设置相应管脚为 LED 驱动功能。

注：在正反推点 LED 灯时，LED 口既做 SEG 口又做 COM 口

18.6 LCD/LED 功能的数据设置

设置 LCD/LED 显示数据需以下步骤：

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

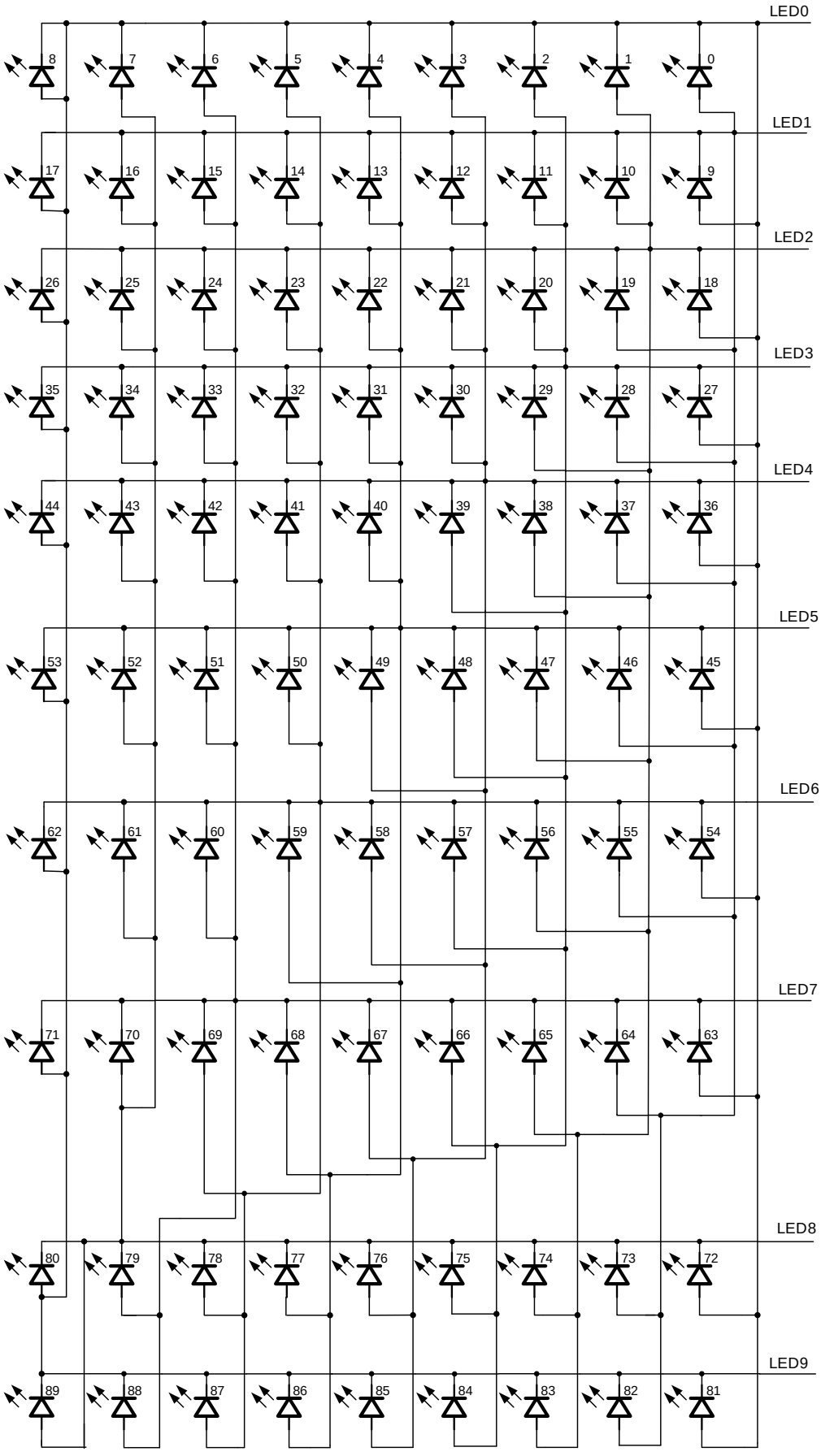
LCD 扫描模式的地址和数据对应关系如下表：

LCDADD	LCDDATA						
00H	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG0
01H	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG1
.	.	.	.	.	.	.	
.	.	.	.	.	.	.	
.	.	.	.	.	.	.	
.	.	.	.	.	.	.	
16H	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG22
17H	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	SEG23
	COM5	COM4	COM3	COM2	COM1	COM0	

LED 扫描模式的地址和数据对应关系如下表：

LCDADD	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
00H	8	7	6	5	4	3	2	1	0
01H	17	16	15	14	13	12	11	10	9
02H	26	25	24	23	22	21	20	19	18
03H	35	34	33	32	31	30	29	28	27
04H	44	43	42	41	40	39	38	37	36
05H	53	52	51	50	49	48	47	46	45
06H	62	61	60	59	58	57	56	55	54
07H	71	70	69	68	67	66	65	64	63
08H	80	79	78	77	76	75	74	73	72
09H	89	88	87	86	85	84	83	82	81

注：Bit8 为 SEGEN3 寄存器中的第 0 位，即 LCDDATA<8>。



LED10*9点矩阵对应像素点

18.7 LCD/LED 相关寄存器

LCD/LED 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、SEGEN3。

LCD/LED 控制寄存器 LCDCON0(11DH)

11DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCON0	LCDEN	LEDEN	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

Bit6 LEDEN: LED 模块使能
0: 禁止 LED 模块
1: 使能 LED 模块。

Bit5~Bit3 COMSEL[2:0]: LCD/LED 模块 COM 口个数选择

	LCD 扫描模式	LED 扫描模式
000:	2COM	3COM
001:	3COM	4COM
010:	4COM	5COM
011:	5COM	6COM
100:	6COM	7COM
101:	6COM	8COM
110:	6COM	9COM
111:	6COM	10COM

Bit2~Bit0 LCDCLK[2:0]: LCD/LED 扫描频率（根据 LCDCON1 的 LCDF<1:0>选择时钟源）

	LCD 扫描模式	LED 扫描模式
000：	F_{HSI}/8192||F_{LSI}/16||F_{LSE}/16	F_{HSI}/1024||F_{LSI}/2||F_{LSE}/2
001：	F_{HSI}/8192||F_{LSI}/16||F_{LSE}/16	F_{HSI}/2048||F_{LSI}/4||F_{LSE}/4
010：	F_{HSI}/8192||F_{LSI}/16||F_{LSE}/16	F_{HSI}/4096||F_{LSI}/8||F_{LSE}/8
011：	F_{HSI}/8192||F_{LSI}/16||F_{LSE}/16	F_{HSI}/8192||F_{LSI}/16||F_{LSE}/16
100：	F_{HSI}/16384||F_{LSI}/32||F_{LSE}/32	F_{HSI}/16384||F_{LSI}/32||F_{LSE}/32
101：	F_{HSI}/32768||F_{LSI}/64||F_{LSE}/64	F_{HSI}/32768||F_{LSI}/64||F_{LSE}/64
110：	F_{HSI}/65536||F_{LSI}/128||F_{LSE}/128	F_{HSI}/65536||F_{LSI}/128||F_{LSE}/128
111：	F_{HSI}/131072||F_{LSI}/256||F_{LSE}/256	F_{HSI}/131072||F_{LSI}/256||F_{LSE}/256

注：LCD/LED 帧频率=LCD/LED 扫描频率÷COM 口个数，LCD 模式下，建议将帧频率设置为 60Hz~120Hz。

LCD/LED 控制寄存器 LCDCON1(11EH)

11EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCON1	----	BIAS	----	----	FCCTLM<1:0>		LCDF<1:0>	
R/W	----	R/W	----	----	R/W	R/W	R/W	R/W
复位值	----	0	----	----	0	0	0	0

Bit7 未用

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

Bit5~Bit4 未用

Bit3~Bit2 FCCTLM<1:0>: LCD 模式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 LCDF<1:0>: LCD/LED 时钟源选择
0x: F_{HSI}
10: F_{LSI}
11: F_{LSE}

LCD/LED 地址寄存器 LCDADD(19CH)

19CH	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/LED 数据读写使能
0: 禁止读写 LCD/LED 数据
1: 允许读写 LCD/LED 数据

Bit6~Bit5 B2RES<1:0>: LCD 模式内部分压电阻选择
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/LED 地址选择
LCD/LED 地址范围 00H-017H

LCD/LED 数据寄存器 LCDDATA(19DH)

19DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDDATA	LCDDATA<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　　LCDDATA<7:0>:　LCD/LED 数据设置，写入 LCDADD 对应地址的数据

注：（1）LCD 模式下，只有 LCDDATA<5:0>数据位有效；
（2）LED 模式下，LCDDATA<8:0>数据位有效（LCDDATA<8>为 SEGEN3 寄存器里的 Bit0 位）；
（3）LED 模式下，写显存的步骤：写显存地址→写 LCDDATA<8>位数据→写 LCDDATA<7:0>位数据；读显存的步骤：写显存地址→读 LCDDATA<7:0>位数据→读 LCDDATA<8>位数据。

LCD/LED 功能 COM 口/LED 口控制寄存器 COMEN(19EH)

19EH	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
复位值	----	----	0	0	0	0	0	0

Bit7~Bit6　　未用
Bit5~Bit0　　COMxEN:　COM 口/LED 口功能设置
0:　对应 COMx 口/LEDy 口为普通 I/O 口(x=0~5，y=8~9)
1:　对应 COMx 口/LEDy 口为 LCD/LED 功能的 COM 口/LED 口(x=0~5，y=8~9)

LCD 功能 SEG 口控制寄存器 SEGEN0(9EH)

9EH	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~Bit0　　SEGxEN:　SEG 口功能设置
0:　对应 SEGx 口为普通 I/O 口(x=0~7)
1:　对应 SEGx 口为 LCD 功能的 SEG 口(x=0-7)

LCD/LED 功能 SEG 口/LED 口控制寄存器 SEGEN1(11FH)

11FH	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/LED 口功能设置；
0:　对应 SEGx 口/LEDy 口为普通 I/O 口(x=8-15，y=0~7)
1:　对应 SEGx 口/LEDy 口为 LCD/LED 功能的 SEG 口/LED 口(x=8-15，y=0~7)

LCD 功能 SEG 口控制寄存器 SEGEN2(181H)

181H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGEN2	SEG23EN	SEG22EN	SEG21EN	SEG20EN	SEG19EN	SEG18EN	SEG17EN	SEG16EN
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=16-23)
1: 对应 SEGx 口为 LCD 功能的 SEG 口(x=16-23)

LCD 功能 SEG 口控制寄存器 SEGEN3(19FH)

19FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SEGEN3	SEGDRI[4:0]					----	----	LCDDATA<8>
R/W	R/W	R/W	R/W	R/W	R/W	----	----	R/W
复位值	0	0	0	0	0	----	----	X

Bit7~Bit3 SEGDRI[4:0]: SEG 口驱动电流设置
00000: SEG 口驱动电流为 0mA
00001: SEG 口驱动电流为 1mA
00010: SEG 口驱动电流为 2mA
00011: SEG 口驱动电流为 3mA
00100: SEG 口驱动电流为 4mA
00101: SEG 口驱动电流为 5mA
00110: SEG 口驱动电流为 6mA
00111: SEG 口驱动电流为 7mA
01000: SEG 口驱动电流为 8mA
…
11101: SEG 口驱动电流为 29mA
11110: SEG 口驱动电流为 30mA
11111: SEG 口驱动电流为 45mA
Bit2~Bit1 未用
Bit0 LCDDATA<8>: LED 模式下，LED 数据的第 8 位

18.8 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”。

18.9 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 晶振或内部 32KHz 低速振荡；
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）使能 LCD；
6）执行“STOP”指令进入休眠态。

19. 运算放大器(OPA)

芯片内置 1 路运算放大器 OPA，下图为其基本功能框图。

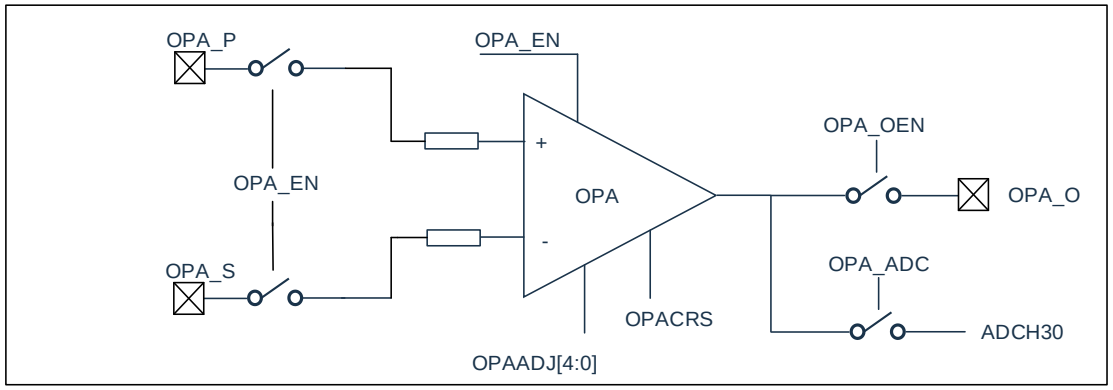


图 19-1：OPA 框图

19.1 OPA 概述

OPA 具有以下特性：

- 内部集成调零电路；
- 正端、负端可以接至 I/O 口；
- 输出端可接至 I/O 口或内部 ADC 检测通道。

19.1.1 OPA 使能

将寄存器 OPACON 的第 7 位 OPA_EN 置 1，使能运算放大器。将 OPA_EN 置 0，禁止运算放大器。使能运算放大器后，正端、负端自动连至 I/O 口。

19.1.2 OPA 端口选择

1. 使能运算放大器后，正端自动连至 I/O 口，负端自动连至 I/O 口；
2. 运放的输出可以从 OPAO 引脚输出，这是通过设置 OPACON 的第 6 位 OPA_OEN 来实现的；运放输出可以通过设置 OPACON 第 4 位 OPA_ADC 接到 ADCH30 通道。

注：OPA 相关的 I/O 口必须设置为输入态，包括运放输入和运放输出（需要连接到 IO 时）。

19.1.3 OPA 工作模式

芯片的内置运放具有 2 种工作模式，正常模式和调节模式。

寄存器 OPAADJ 的第 6 位 OPACOFM 置 0，运放进入正常工作模式。

寄存器 OPAADJ 的第 6 位 OPACOFM 置 1，运放进入调节模式。在调节模式下，运放的正负端内部短路在一起，并连接至运放的正端或者负端（通过 OPAADJ 的第 5 位 OPACRS 来选择）。调节模式的作用是将运放的失调电压调至最小。

调节模式工作流程：

1. 使能运放功能；
2. 设置运放进入调节模式；
3. 设置运放调节模式从正端输入或者负端输入，输入端不能悬空；
4. 将调节位 OPAADJ<4:0>设置成初始值，最大(1FH)或最小(00H)；
5. 延时一段时间，该时间和外部电容参数有关；
6. 读取运放输出；
7. 将调节位自减 1（初始值设置成最大 1FH）或者自加 1（初始值设置成 00H）；
8. 延时；
9. 读取运放输出，是否发生改变，如果没有改变，则继续执行步骤 7；
10. 读取值发生改变，调零结束，将 OPACOFM 清零，进入正常工作模式。

19.2 OPA 相关寄存器

有 2 个寄存器和 OPA 模块相关，分别是 OPACON、OPAADJ。

OPA 控制寄存器 OPACON (18CH)

18CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPACON	OPA_EN	OPA_OEN	----	OPA_ADC	----	----	----	OPA_FT
R/W	R/W	R/W	----	R/W	----	----	----	R/W
复位值	0	0	----	0	----	----	----	0

Bit7 OPA_EN: OPA使能位
1= 使能OPA
0= 关闭OPA
Bit6 OPA_OEN: OPA输出使能
1= OPA输出接至I/O口(OPA_O管脚)
0= OPA输出不接至I/O口
Bit5 未用
Bit4 OPA_ADC: ADC通道使能
1= OPA的输出接至ADC30通道
0= OPA的输出不接至ADC30通道
Bit3~Bit1 未用
Bit0 OPA_FT: 运放输出内部滤波选择
1= OPA输出内部接滤波电路
0= OPA输出内部不接滤波电路

OPA 控制寄存器 OPAADJ (18DH)

18DH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPAADJ	OPADOUT	OPACOFM	OPACRS	OPAADJ[4:0]				
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	x	x	x	x	x

Bit7 OPADOUT: OPA输出结果
1= 运放输出为高，正端电压高于负端电压
0= 运放输出为低，正端电压低于负端电压
Bit6 OPACOFM: OPA工作模式选择
1= OPA工作在调节模式
0= OPA工作在正常模式
Bit5 OPACRS: OPA调节模式输入端选择位
1= OPA调节模式正端输入
0= OPA调节模式负端输入
Bit4~Bit0 OPAADJ[4:0]: OPA失调电压调节位

20. RGB 彩灯驱动模块

芯片内置 1 路 RGB 彩灯驱动模块，具有 30 个字节的数据缓存空间，可驱动 10 级 RGB 彩灯，并且支持数据重发功能，开启数据重发功能后可将 RGB 彩灯的级数扩展至 15、20 级。

20.1 相关寄存器

RGB 彩灯驱动模块相关寄存器有：控制寄存器 LEDCTR0、LEDCTR1，数据寄存器 LEDSDATA。

RGB 彩灯驱动控制寄存器 LEDCTR0 (18EH)

18EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LEDCTR0	LEDSWE	LEDSW_RT	IF_15	IF_30	RESET_T	LEDSW_CKSEL<1:0>		LEDSW_ST
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 LEDSWE: RGB彩灯驱动模块使能位
1= 使能RGB彩灯驱动模块
0= 关闭RGB彩灯驱动模块

Bit6 LEDSW_RT: 重发模式使能位
1= 使能重发模式
0= 关闭重发模式

Bit5 IF_15: 发码完成标志位
1= 已完成发码15个字节（第1到15字节），需软件清零才能重发（第1到15字节）
0= 未完成发码15个字节

Bit4 IF_30: 发码完成标志位
1= 已完成发码30个字节（第1到30字节），需软件清零才能重发（第16到30字节）
0= 未完成发码30个字节

Bit3 RESET_T: RESET时间选择
1= 300us
0= 100us

Bit2~Bit1 LEDSW_CKSEL<1:0>: 码元格式选择
11= T0H=500ns，T0L=2000ns；T1H=2000ns，T1L=500ns
10= T0H=312.5ns，T0L=875ns；T1H=875ns，T1L=875ns
01= T0H=250ns，T0L=1000ns；T1H=875ns，T1L=375ns
00= T0H=312.5ns，T0L=625ns；T1H=625ns，T1L=312.5ns

Bit0 LEDSW_ST: 发码启动位
1= 启动发码，发码完成后硬件自动清零
0= 禁止发码

RGB 彩灯驱动控制寄存器 LEDCTR 1(18FH)

18FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LEDCTR1	LEDSW_ADR	----	----	LEDSW_ADD<4:0>				
R/W	R/W	----	----	R/W	R/W	R/W	R/W	R/W
复位值	0	----	----	0	0	0	0	0

Bit7 LEDSW_ADR: RGB彩灯驱动模块读写数据缓存使能位
1= 使能读写数据缓存
0= 禁止读写数据缓存
Bit6~Bit5 未用
Bit4~Bit0 LEDSW_ADD<4:0> 数据缓存地址选择
00000= 第1字节数据缓存地址
…
11101= 第30字节数据缓存地址

RGB 彩灯驱动模块数据寄存器 LEDSDATA(19BH)

19BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LEDSDATA	LEDSDATA<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 LEDSDATA<7:0>: 读取或写入LEDSW_ADD<4:0>所设置地址的数据缓存的数据

20.2 数据缓存的读写

可参考如下步骤对数据缓存进行读写操作：

1. 设置 LEDCTR1 寄存器的第 7 位 LEDSW_ADR=1，允许读写缓存数据；
2. 向 LEDCTR1 寄存器的 LEDSW_ADD<4:0>位写入缓存的地址；
3. 向 LEDSDATA 寄存器写入数据，或读取 LEDSDATA 寄存器的数据；
4. 重复步骤 2-3，读写其它地址的缓存数据；
5. 完成后将 LEDSW_ADR 位清 0，关闭数据缓存读写使能位。

注：

（1） RGB 彩灯驱动模块的数据缓存和 PD 模块的数据接收缓存共用一块物理缓存，因此应用过程中不能同时开启这两个模块（物理缓存空间的地址范围为：00H-1DH）；

（2） LEDSDATA 寄存器和 PDSDATA 寄存器映射的是同一个物理地址，访问 LEDSDATA 寄存器时必须先设置 LEDSW_ADR=1，访问 PDSDATA 寄存器时则必须先设置 LEDSW_ADR=0；

（3） 当 LED 彩灯发码模块正在进行数据发送（即 LEDSW_ST=1）时，只允许对 LEDSDATA 寄存器进行写操作，读操作无效。

20.3 缓存数据的发送

RGB 彩灯驱动模块仅支持单工通信，使能了该模块后，发送空闲时 LED_DAT 口为低电平；将 LEDSW_ST 位置 1 后，模块会以归零码的方式在 LED_DAT 口将 30 字节的缓存数据逐位发出；发送完第 15 个字节时 IF_15 位被硬件置 1，发送完第 30 个字节时 IF_30 位被硬件置 1；当数据位发送完成后会在末尾发送 RESET_T 时间的低电平作为 RESET 码，发送完 RESET 码后 LEDSW_ST 位会被硬件清 0，发送过程如下图所示：

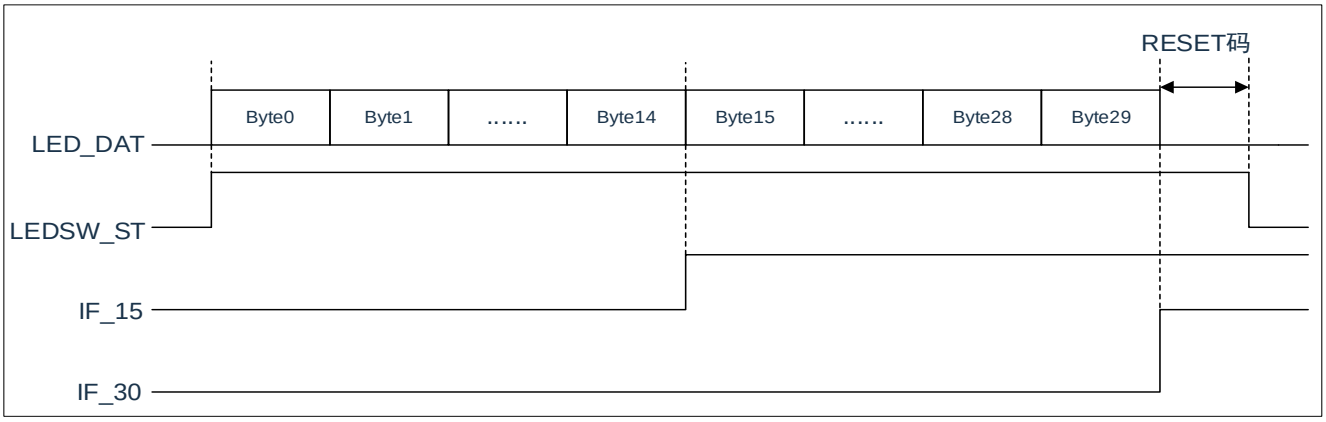


图 20-1：数据发送框图

注：

（1） LED_DAT 口通过 CONFIG 配置在 RA5 或 RB2，发送数据时需要将 LED_DAT 口设为输出口，即将对应的 TRIS 位清 0；

（2） 数据发送时是先发高位再发低位。

20.3.1 输出码元格式

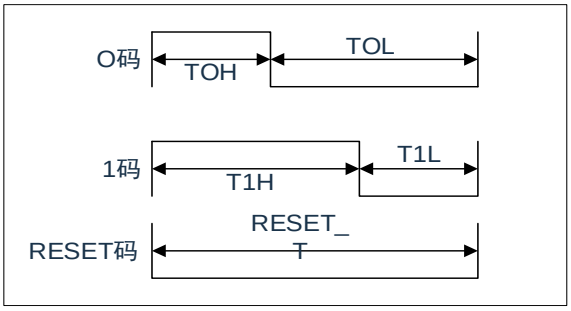


图 20-2：码元格式框图

20.3.2 输出码元时间

通过配置 LEDCTR0 寄存器里的 LEDSW_CKSEL<1:0>位可选择数据码元时间，相关配置对应的时间关系如下表所示：

LEDSW_CKSEL<1:0>	T0H	T0L	T1H	T1L
00	312.5ns	625 ns	625 ns	312.5 ns
01	250 ns	1000 ns	875 ns	375 ns
10	312.5 ns	875 ns	875 ns	875 ns
11	500 ns	2000 ns	2000 ns	500 ns

注：RESET 码的时间由 LEDCTR0 寄存器里的 RESET_T 位配置，RESET_T=1 时，RESET 码的时间为 300us，RESET_T=0 时，RESET 码的时间为 100us。

20.3.3 缓存数据发送的操作步骤

需要启动 RGB 彩灯驱动模块发送缓存数据时可参考如下操作步骤：

1. 将 LEDCTR0 寄存器的 LEDSWE 位置 1，使能 RGB 彩灯驱动模块；
2. 将 LED_DAT 口对应的 TRIS 位清 0，设置 LED_DAT 口为输出口；
3. 设置 LEDCTR0 寄存器里的 LEDSW_CKSEL<1:0>、RESET_T 位的值，配置好码元时间以及 RESET 码时间；
4. 设置 00H~1DH 地址缓存的数据（参考 20.2 章节）
5. 将 LEDCTR0 寄存器里的 LEDSW_ST 位置 1，启动发送；
6. 等待 LEDSW_ST=0；
7. 数据发送完成，将 LEDCTR0 寄存器的 LEDSWE 清 0，关闭 RGB 彩灯驱动模块。

20.4 数据重发功能

RGB 彩灯驱动模块在不使能重发模式时，每次只发送 30 字节的缓存数据；当使能了重发模式后，在第 30 个字节的最后 1 个位发送结束时芯片内部会检测 IF_15 位是否为 0，若为 0 则会再次发送 00H~0EH 地址的缓存数据，反之则停止数据发送；当第 15 字节的最后 1 个位发送结束时芯片内部会检测 IF_30 位是否为 0，若为 0 则会再次发送 0FH~1DH 地址的缓存数据，反之则停止数据发送。

20.4.1 重发 00H~0EH 地址缓存数据的操作步骤

1. 将 LEDCTR0 寄存器的 LEDSWE 位和 LEDSW_RT 位置 1，使能 RGB 彩灯驱动模块并使能重发模式；
2. 将 LED_DAT 口对应的 TRIS 位清 0，设置 LED_DAT 口为输出口；
3. 设置 LEDCTR0 寄存器里的 LEDSW_CKSEL<1:0>、RESET_T 位的值，配置好码元时间以及 RESET 码时间；
4. 设置 00H~1DH 地址缓存的数据（参考 20.2 章节）
5. 将 LEDCTR0 寄存器里的 LEDSW_ST 位置 1，启动发送；
6. 等待 IF_15=1；
7. 清零 IF_15，并更新 00H~0EH 地址缓存的数据（必须在 IF_30=1 之前将数据更新完成）；
8. 等待 LEDSW_ST=0；
9. 数据发送完成，将 LEDCTR0 寄存器的 LEDSWE 位和 LEDSW_RT 位清 0，关闭 RGB 彩灯驱动模块和重发模式。

20.4.2 重发 00H~1DH 地址缓存数据的操作步骤

1. 将 LEDCTR0 寄存器的 LEDSWE 位和 LEDSW_RT 位置 1，使能 RGB 彩灯驱动模块并使能重发模式；
2. 将 LED_DAT 口对应的 TRIS 位清 0，设置 LED_DAT 口为输出口；
3. 设置 LEDCTR0 寄存器里的 LEDSW_CKSEL<1:0>、RESET_T 位的值，配置好码元时间以及 RESET 码时间；
4. 设置 00H~1DH 地址缓存的数据（参考 20.2 章节）
5. 将 LEDCTR0 寄存器里的 LEDSW_ST 位置 1，启动发送；
6. 等待 IF_15=1；
7. 清零 IF_15，并更新 00H~0EH 地址缓存的数据（必须在 IF_30=1 之前将数据更新完成）；
8. 等待 IF_30=1；
9. 清零 IF_30，并更新 0FH~1DH 地址缓存的数据（必须在 IF_15=1 之前将数据更新完成）；
10. 等待 LEDSW_ST=0；
11. 数据发送完成，将 LEDCTR0 寄存器的 LEDSWE 位和 LEDSW_RT 位清 0，关闭 RGB 彩灯驱动模块和重发模式。

21. QC 模块

芯片内部集成了 1 个 QC 模块，内含 2 个独立的 LDO，可通过相关的寄存器位控制 LDO 在 DM、DP 口输出 0V/0.6V/3.3V 的电压。

当DMEN=1时,DM口自动变为模拟输出态,对DMDAT位写1输出0.6V(DMSEL=0)或者3.3V(DMSEL=1)；写 0 则输出 0V；当 DPEN=1 时，DP 口自动变为模拟输出态，对 DPDAT 位写 1 输出 0.6V(DPSEL=0)或者3.3V(DPSEL=1)；写 0 则输出 0V。

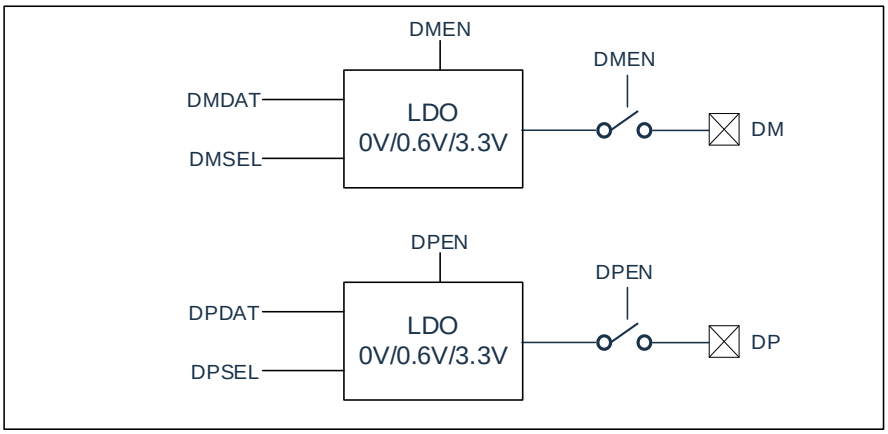


图 21-1：QC 模块功能框图

21.1 QC 模块相关寄存器

QC 模块控制寄存器 QCCON(0CH)

0CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
QCCON	----	----	DPDAT	DMDAT	DPSEL	DPEN	DMSEL	DMEN
R/W	----	----	R/W	R/W	R/W	R/W	R/W	R/W
复位值	----	----	0	0	0	0	0	0

Bit7~Bit6 未用

Bit5 DPDAT: DP口数据位
1= 3.3V或0.6V(由DPSEL选择)
0= 0V

Bit4 DMDAT: DM口数据位
1= 3.3V或0.6V(由DMSEL选择)
0= 0V

Bit3 DPSEL: DP口输出电压选择(DPEN=1才有效)
1= 3.3V
0= 0.6V

Bit2 DPEN: DP口控制位
1= 该I/O口作为DP电压输出口
0= 普通I/O

Bit1 DMSEL: DM口输出电压选择(DMEN=1才有效)
1= 3.3V
0= 0.6V

Bit0 DMEN: DM口控制位
1= 该I/O口作为DM电压输出口
0= 普通I/O

22. PD 模块

芯片内部集成 1 组 Type-C 口（CC0 和 CC1）和 PD 通讯模块，具有如下功能特性：

- 支持正反接和拔插检测；
- 仅支持 Sink/Consumer，不支持 Source/Provider；
- 仅支持 SOP 的 PD 包，不支持 SOP'和 SOP''等 PD 包，也不支持 PD 复位信号帧硬件复位；
- 内置 PD 收发器，自动 4b5b 编解码、BMC 编解码和 CRC；
- 接收器支持的最大包长度为 30 字节，发送器支持的最大包长度为 6 字节。

22.1 PD 功能框图

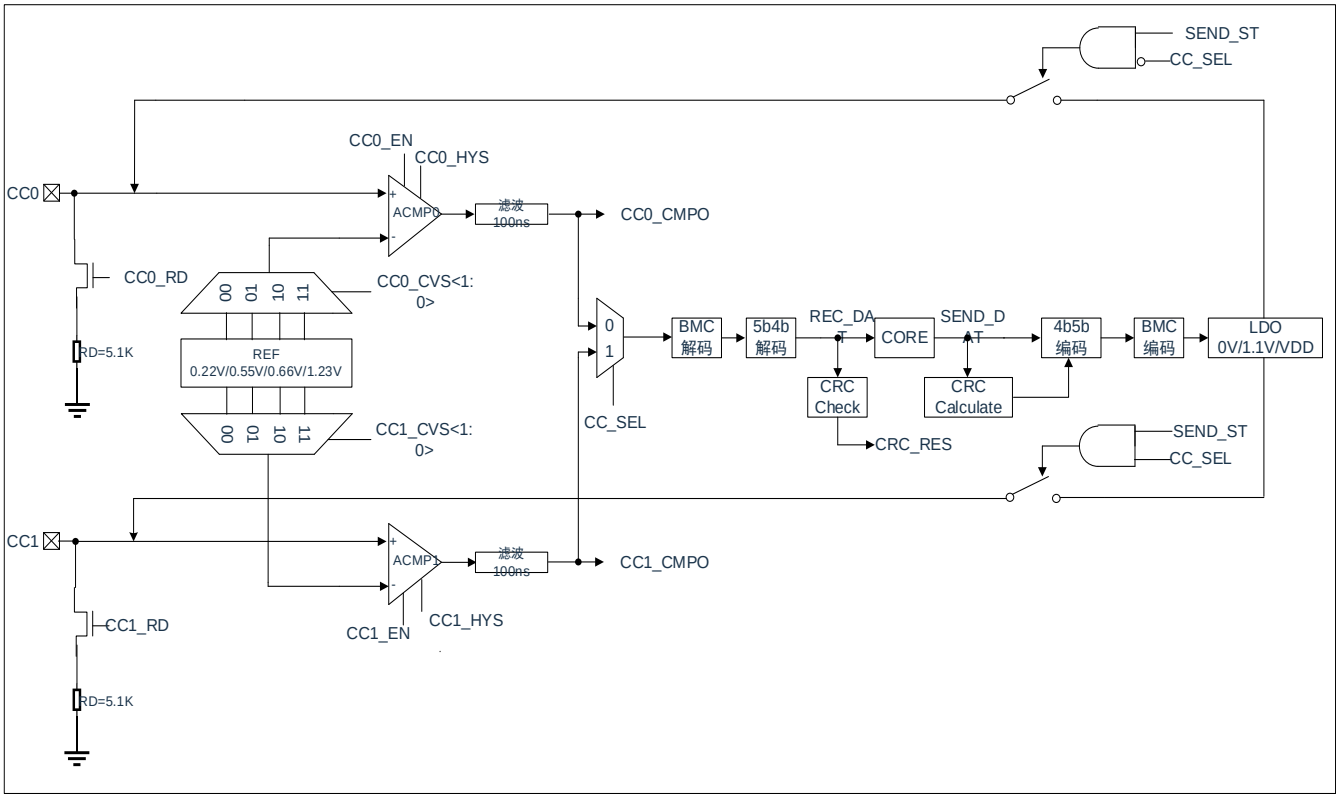


图 22-1：PD 功能框图

22.2 PD 通信端口配置

CC0 和 CC1 的端口功能一致，以下描述里 x 的值为 0，1；通过 CCxCON 寄存器的相关控制位可以配置 CCx 端口的功能。

通过将 CCxCON 寄存器里的 CCx_RD 置 1 可开启 CCx 端口的下拉电阻；配置 CCxCON 寄存器里 CCx_CVS<1:0>的值可选择 CCx 端口比较器负端的参考电压；将 CCx_HYS 位置 1 可开启比较器的迟滞电压，将 CCx_EN 位置 1 可使能 CCx 端口比较器；使能比较器后，当正端电压大于负端电压时 CCx_CMPO 位为 1，当正端电压小于负端电压时 CCx_CMPO 位为 0。

22.3 PD 数据接收

通过将 PDCON0 寄存器里的 PDEN 位置 1 使能 PD 通信模块，将 PD_MODE 位清 0 开启 PD 通信模块的接收模式，配置 CC_SEL 位的值可将接收端口选择在 CC0 或 CC1。当开始检测到有效的 SOP PD 数据包后 REC_S 位被拉低；数据接收过程中，每接收到 1 个字节的数据 PDCON1 寄存器里的 PDCNT<4:0>位自动加 1 并将数据写入接收缓存；数据接收完成后将发生以下事件：

1) 将 PDCON0 寄存器里的 REC_S 位拉高；
2) 将 PIR2 寄存器里的 PD 数据接收中断标志位 PDRIF 置 1；
3) 将接收到的数据进行 CRC 校验，若校验正确则将 PDCON0 寄存器里的 CRC_RES 位清零，若校验错误则将 CRC_RES 位置 1。

22.3.1 接收缓存数据的读取

PD 模块内含 30 字节的接收数据缓存，当数据接收完成后通过 PDCON1 寄存器 PDCNT<4:0>位的值可以知道一个共接收了多少字节的数据，并通过 PDRDATA 寄存器和 PDADD 寄存器将缓存数据读出，以下为读取接收缓存数据的步骤：

1) 读取 PDCON1 寄存器里 PDCNT<4:0>的值，根据该值获取需要读取的缓存地址范围；
2) 将缓存地址 00H 写入 PDADD 寄存器里的 PDRADD<4:0>位；
3) 读取 PDRDATA 寄存器，获取对应地址的缓存数据；
4) 将 PDRADD<4:0>的值加 1；
5) 判断 PDRADD<4:0>的值是否等于 PDCNT<4:0>，若不等则重复 3、4、5 的步骤，若相等则读取结束。

> 注：PD 模块的数据接收缓存和 RGB 彩灯驱动模块的数据缓存共用一块物理缓存，因此应用过程中不能同时开启这两个模块（物理缓存空间的地址范围为：00H-1DH）；

22.3.1 PD 数据接收的配置

可参考以下步骤启动 PD 模块进行数据接收：

1) 将 PDCON0 寄存器里 PDEN 位置 1，使能 PD 模块，将 PD_MODE 位清 0，设为数据接收模式；
2) 配置 PDCON0 寄存器里 CC_SEL 位的值，选择 PD 通信端口；
3) 通过 CC0CON 或者 CC1CON 寄存器配置通信端口（参考 22.2 章节）；
4) 等待 PIR2 寄存器里的 PDRIF 位被置 1；
5) 读取 CRC_RES 位的值，若为 0 则读取接收缓存里的数据（参考 22.3.1 章节），若为 1 则忽略本次接收的数据。

22.4 PD 数据发送

通过将 PDCON0 寄存器里的 PDEN 位置 1 使能 PD 通信模块，将 PD_MODE 位置 1 开启 PD 通信模块的发送模式，配置 CC_SEL 位的值可将发送端口选择在 CC0 或 CC1，配置 PD_LDO 位选择 LDO 的输出电压为 1.1V 或者 VDD，配置 PDCON1 寄存器里 SEND_SEL 位可选择发送数据的字节数据；最后将 PDCON0 寄存器里的 SEND_ST 位置 1，启动数据发送，此时发送器会先计算发送缓存数据的 CRC 结果，然后将发送缓存的数据和 CRC 结果发送出去，当发送完成后将发生以下事件：

1) 将 PDCON0 寄存器里的 SEND_ST 位清 0；
2) 将 PIR2 寄存器里的 PD 数据发送中断标志位 PDSIF 置 1。

22.4.1 发送缓存数据的读写

PD 模块内含 6 字节的发送数据缓存，在启动数据发送之前必须将需要发送的数据写入发送缓存，通过 PDSDATA 寄存器、PDADD 寄存器可读写发送缓存的数据，以下为读写发送缓存数据的步骤：

1) 将 PDCON1 寄存器里的 PDSEN 位置 1，允许写发送缓存数据；
2) 将缓存地址 00H 写入 PDADD 寄存器里的 PDSADD<2:0>位；
3) 读写 PDSDATA 寄存器，即读写对应地址的缓存数据；
4) 将 PDSADD<2:0>的值加 1；
5) 重复 3、4 的步骤，读写其他地址的缓存数据。

> 注：PDSDATA 寄存器和 LEDSDATA 寄存器映射的是同一个物理地址，访问 LEDSDATA 寄存器时必须先设置 LEDSW_ADR=1，访问 PDSDATA 寄存器时则必须先设置 LEDSW_ADR=0；

22.4.2 LDO 输出

数据经过 4b5b 编码和 BMC 编码后由 LDO 输出到 CC0 或 CC1 通信端口，BMC 编码输出为 0 时，LDO 输出电压为 0V；BMC 编码输出为 1 时，LDO 输出电压由 PDCON0 寄存器的 PD_LDO 位决定，若 PD_LDO=1，则输出电压为 1.1V，若 PD_LDO=0，则输出电压为 VDD。

22.4.3 PD 数据发送的配置

可参考以下步骤启动 PD 模块进行数据发送：

1) 将 PDCON0 寄存器里 PDEN 位置 1，使能 PD 模块，将 PD_MODE 位置 1，设为数据发送模式；
2) 配置 PDCON0 寄存器里 CC_SEL 位的值，选择 PD 通信端口；
3) 配置 PDCON1 寄存器里 SEND_SEL 位的值，选择要发送的字节数；
4) 将要发送的数据写入发送缓存（参考 22.4.1 章节）；
   将 PDCON0 寄存器里 SEND_ST 位置 1，启动数据发送；
5) 等待数据发送完成。

22.5 PD 相关寄存器

CC0 控制寄存器 CC0CON(195H)

195H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CC0CON	----	----	CC0_EN	CC0_HYS	CC0_CVS<1:0>		CC0_RD	CC0_CMPO
R/W	----	----	R/W	R/W	R/W	R/W	R/W	R
复位值	----	----	0	0	0	0	1	X

Bit7~Bit6 未用

Bit5 CC0_EN: CC0端口电压比较器使能位
1= 使能比较器
0= 禁止比较器

Bit4 CC0_HYS: CC0端口电压比较器迟滞电压选择位
1= 迟滞±20mV
0= 迟滞关闭

Bit3~Bit2 CC0_CVS<1:0>: CC0端口电压比较器的负端参考电压选择位
11= 1.23v
10= 0.66v
01= 0.55v
00= 0.22v

Bit1 CC0_RD: CC0端口下拉电阻使能位
1= 使能5.1K下拉电阻
0= 禁止下拉电阻

Bit0 CC0_CMPO CC0端口电压比较器结果位
1= CC0口电压大于负端参考电压
0= CC0口电压小于负端参考电压

CC1 控制寄存器 CC1CON(196H)

196H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CC1CON	----	----	CC1_EN	CC1_HYS	CC1_CVS<1:0>		CC1_RD	CC1_CMPO
R/W	----	----	R/W	R/W	R/W	R/W	R/W	R
复位值	----	----	0	0	0	0	1	X

Bit7~Bit6 未用

Bit5 CC1_EN: CC1端口电压比较器使能位
1= 使能比较器
0= 禁止比较器

Bit4 CC1_HYS: CC1端口电压比较器迟滞电压选择位
1= 迟滞±20mV
0= 迟滞关闭

Bit3~Bit2 CC1_CVS<1:0>: CC1端口电压比较器的负端参考电压选择位
11= 1.23v
10= 0.66v
01= 0.55v
00= 0.22v

Bit1 CC1_RD: CC1端口下拉电阻使能位
1= 使能5.1K下拉电阻
0= 禁止下拉电阻

Bit0 CC1_CMPO CC1端口电压比较器结果位

1= CC1口电压大于负端参考电压
0= CC1口电压小于负端参考电压

PD 控制寄存器 PDCON0(197H)

197H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PDCON0	PDEN	PD_LDO	SEND_ST	CRC_RES	CC_SEL	PD_MODE	-----	REC_S
R/W	R/W	R/W	R/W	R	R/W	R/W	----	R
复位值	0	0	0	0	0	0	----	1

Bit7 PDEN: PD 模块使能位
1= 使能 PD 模块
0= 禁止 PD 模块
Bit6 PD_LDO: PD LDO 输出电压选择位
1= 1.1V
0= VDD
Bit5 SEND_ST PD 发送启动位，在 PD_MODE=1 时有效
1= 启动发送，PHY 自动切为输出，发送完成后该位自动清零
0= 禁止 PD 发送数据
Bit4 CRC_RES: 接收数据 CRC 校验结果位
1= 接收数据 CRC 校验错误
0= 接收数据 CRC 校验正确
Bit3 CC_SEL: PD 通信数据收发端口选择位
1= CC1
0= CC0
Bit2 PD_MODE: PD 工作模式选择位
1= PD 为发送数据模式
0= PD 为接收数据模式
Bit1 未用
Bit0 REC_S PD 接收数据状态位
1= PD 未在进行接收数据
0= PD 正在进行接收数据

PD 控制寄存器 PDCON1(198H)

198H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PDCON1	----	PDSEN	SEND_SEL	PDCNT<4:0>				
R/W	R/W	R/W	R/W	R	R	R	R	R
复位值	0	0	0	0	0	0	0	0

Bit7 未用
Bit6 PDSEN: 写 PD 发送缓存数据寄存器使能位
1= 允许写 PD 发送缓存数据
0= 禁止写 PD 发送缓存数据
Bit5 SEND_SEL: PD 发送数据字节数选择位
1= 发送 6 个字节
0= 发送 2 个字节
Bit4~Bit0 PD_CNT<4:0> PD 接收数据字节计数寄存器

PD 地址寄存器 PDADD(199H)

199H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PDADD	PDSADD<2:0>			PDRADD<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~Bit5 PDSADD<2:0>: PD 发送缓存地址选择位
Bit4~Bit0 PDRADD<4:0>: PD 接收缓存地址选择位

PD 接收缓存数据寄存器 PDRDATA(19AH)

19AH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PDRDATA	PDRDATA<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 PDRDATA<7:0>: PD 接收缓存数据位

PD 发送缓存数据寄存器 PDSDATA(19BH)

19BH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PDSDATA	PDSDATA<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 PDSDATA<7:0>: PD 发送缓存数据位

23. 电气参数

23.1 极限参数

电源供应电压.....GND-0.3V~GND+5.5V
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。

23.2 直流电气特性

（VDD=5V，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}=16MHz/2T	2.5	-	5.5	V
		-	F_{SYS}=16MHz/4T	1.8	-	5.5	V
I_{DD}	工作电流	5V	F_{SYS}=16MHz, 所有模拟模块关闭	-	3.5	-	mA
		5V	F_{SYS}=8MHz, 所有模拟模块关闭	-	2	-	mA
		3V	F_{SYS}=16MHz, 所有模拟模块关闭	-	2	-	mA
		3V	F_{SYS}=8MHz, 所有模拟模块关闭	-	1.5	-	mA
		5V	烧写程序 EEPROM	-	6	-	mA
I_{STB}	静态电流	5V	LVR=DIS WDT=DIS	-	3	7	uA
		3V	LVR=DIS WDT=DIS	-	1.8	5	uA
		5V	LVR=DIS WDT=EN	-	6.5	14	uA
		3V	LVR=DIS WDT=EN	-	3	7	uA
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 模块工作电压	-	----	2.5	-	5.5	V
R_{PH}	上拉电阻阻值	5V	Vo=0.5VDD	-	36	-	KΩ
		3V	Vo=0.5VDD	-	56	-	KΩ
R_{PL}	下拉电阻阻值	5V	Vo=0.5VDD	-	36	-	KΩ
		3V	Vo=0.5VDD	-	60	-	KΩ
I_{OL}	输出口灌电流 (普通 I/O 口)	5V	V_{OL}=0.3VDD	-	40	-	mA
		3V	V_{OL}=0.3VDD	-	20	-	mA
I_{OH}	输出口拉电流 (普通 I/O 口)	5V	V_{OH}=0.7VDD	-	-20	-	mA
		3V	V_{OH}=0.7VDD	-	-10	-	mA
I_{OL}	输出口灌电流 (LED 口)	5V	V_{OL}=0.3VDD	-	125	-	mA
		3V	V_{OL}=0.3VDD	-	60	-	mA
I_{OH}	输出口拉电流 (LED 口最大电流)	5V	V_{OH}=0.7VDD	-	-45	-	mA
		3V	V_{OH}=0.7VDD	-	-20	-	mA
V_{BG}	内部基准电压 1.2V	VDD=2.5~5.5V T_A=25℃		-1.5%	1.2	+1.5%	V
		VDD=2.0~5.5V T_A=25℃		-2.5%	1.2	+2.5%	V
		VDD=2.5~5.5V T_A=-40~85℃		-2.0%	1.2	+2.0%	V
		VDD=2.0~5.5V T_A=-40~85℃		-3.0%	1.2	+3.0%	V

23.3 比较器特性

（TA= 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	工作电压范围	-	2.0	-	5.5	V
Iwork	工作电流	VDD=5V COMP+=2V COMP-=2V	-	34	46	uA
		VDD=3V COMP+=1V COMP-=1V	-	20	26	uA
I_{BG}	BG工作电流	VDD=5V	-	35	46	uA
		VDD=3V	-	33	44	uA
V_{IN}	输入共模电压范围	-	0	-	VDD-1.3	V
Vos	失调电压	-	-	-	±13	mV
LSB	最小分辨率	-	-	10	20	mV
Tr	响应时间	-	-	-	1.5	us
-	内部电阻分压误差	VDD=5V V_R>1V	-1%	-	＋1%	-
		VDD=5V V_R<1V	-2%	-	＋2%	-

备注：V_R 为内部电阻分压输出值

23.4 CC 比较器特性

（T_A= 25℃，除非另有说明）

符号	参数	条件	最小值	典型值	最大值	单位
V_{DD}	电源电压	-	2.5	-	5.5	V
T_A	工作温度	-	-40	25	85	℃
I_{work}	工作电流	VDD=2.5~5.5V, T_A =-40~85℃ (2 个比较器同时工作)	125	140	150	uA
V_{OS}	输入失调电压	-	-8.0	-	8.0	mV
V_{CM}	共模输入电压范围	-40℃~85℃	0	-	V_{DD}-1.3	V
V_{HYS}	输入迟滞电压	V_{DD}=2.5~5.5V, V_{IN+}=0.5V	-	±20	-	mV
PSRR	电源抑制比	V_{DD}=2.5~5.5V, V_{IN+}=1V, V_{SENSE}=0mV	-	80	-	dB
CMRR	共模抑制比	V_{DD}=2.5~5.5V, -40℃~85℃	-	80	-	dB
T_{STB}	稳定时间	-	-	-	2	μs
T_{PGD}	响应延时	V_{COM}=1V, V_{IN+}= V_{IN-} ±0.1V	-	100	200	ns

23.5 PD LDO 电气特性

（T_A= 25℃，除非另有说明）

符号	参数	条件	最小值	典型值	最大值	单位
V_{DD}	电源电压	-	2.5	-	5.5	V
T_A	工作温度	-	-40	-	85	℃
I_{work}	工作电流	VDD=2.5~5.5V, T_A =-40~85℃	150	180	210	uA
V_O	输出电压精度	VDD=2.5~5.5V, T_A =-40~85℃	-6%	1.1	+6%	V
I_O	输出驱动电流	VDD=2.5~5.5V, T_A =-40~85℃	-8%	500	+8%	uA
T_C	输出上升/下降时间	V_{DD}=5V, T_A =25℃ CC 线外接 330pF 电容	300	-	1300	nS
Rd	CC0/CC1 口下拉电阻	VDD=2.5~5.5v, T_A =-40~85℃	4.6	5.1	5.6	KΩ

23.6 QC LDO 电气特性

（T_A= 25℃，除非另有说明）

符号	参数	条件	最小值	典型值	最大值	单位
V_{DD}	电源电压	-	4.0	-	5.5	V
T_A	工作温度	-	-40	-	85	℃
I_{work}	工作电流	VDD=4.0~5.5V, T_A =-40~85℃ (2 个 LDO 同时工作)	620	1060	1100	uA
V_O	输出电压精度	VDD=4.0~5.5V，T_A =-40~85℃	-6%	0.6	+6%	V
			-6%	3.3	+6%	V
I_O	输出驱动电流能力	4.0V<V_{DD}<5.5V，T_A =-40~85℃	-8%	150	+8%	uA

23.7 OPA 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
DC 电气特性						
VDD	工作电压	-	2.5	-	5.5	V
T_A	工作温度	-	-40	-	85	℃
I_{DD}	工作电流	VDD=5.0V	-	380	-	uA
V_{OPOS}	输入失调电压	默认值 VDD=5V VCM=1V	-20	-	20	mV
		调零后 VDD=5V VCM=1V	-5	-	5	mV
VCM	共模电压范围	-	0	-	VDD-1.5V	V
PSRR	电源电压抑制比*	-	60	70	-	dB
CMRR	共模抑制比*	VDD=5V VCM=0~VDD-1.5V	90	100	-	dB
AC 电气特性						
AOL	开环增益*		90	100	-	dB
GBW	增益带宽*	RL=1MΩ，CL=100Pf	1.5	2	-	MHz

*表示由设计保证，未批量测试。

23.8 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_{ADC}	ADC 工作电压	V_{ADREF}= VDD,F_{ADCCLK}=1MHz	3.0		5.5	V
		V_{ADREF}= VDD,F_{ADCCLK}=500kHz	2.7		5.5	V
		V_{ADREF}=2.0V,F_{ADCCLK}=250kHz	2.7		5.5	V
		V_{ADREF}=2.4V,F_{ADCCLK}=250kHz	2.7		5.5	V
		V_{ADREF}=3.0V,F_{ADCCLK}=500kHz	3.3		5.5	V
I_{ADC}	ADC 转换电流	V_{ADC}=5V,V_{ADREF}= VDD, F_{ADCCLK}=500kHz			500	uA
		V_{ADC} =3V,V_{ADREF}= VDD, F_{ADCCLK}=500kHz			200	uA
V_{AIN}	ADC 输入电压	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=500kHz	0		V_{ADC}	V
DNL1	微分非线性误差	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=1MHz		±4		LSB
INL1	积分非线性误差	V_{ADC}=5V, V_{ADREF} = VDD, F_{ADCCLK}=1MHz		±8		LSB
DNL2	微分非线性误差	V_{ADC}=5V, V_{ADREF} = 3.0V, F_{ADCCLK}=500kHz,V_{AIN}<1V		±4		LSB
INL2	积分非线性误差	V_{ADC}=5V, V_{ADREF} = 3.0V, F_{ADCCLK}=500kHz,V_{AIN}<1V		±16		LSB
DNL3	微分非线性误差	V_{ADC}=5V, V_{ADREF} = 2.4V, F_{ADCCLK}=250kHz,V_{AIN}<0.8V		±4		LSB
INL3	积分非线性误差	V_{ADC}=5V, V_{ADREF} = 2.4V, F_{ADCCLK}=250kHz,V_{AIN}<0.8V		±16		LSB
DNL4	微分非线性误差	V_{ADC}=5V, V_{ADREF} = 2.0V, F_{ADCCLK}=250kHz,V_{AIN}<0.7V		±4		LSB
INL4	积分非线性误差	V_{ADC}=5V, V_{ADREF} = 2.0V, F_{ADCCLK}=250kHz,V_{AIN}<0.7V		±16		LSB
T_{ADC}	ADC 转换时间			16		T_{ADCCLK}

备注：低温规格值由设计保证，量产不测低温条件。

23.9 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_{LVR1}	LVR 设定电压=1.8V	VDD=1.6~5.5V	1.7	1.8	1.9	V
V_{LVR2}	LVR 设定电压=2.0V	VDD=1.8~5.5V	1.9	2.0	2.1	V
V_{LVR3}	LVR 设定电压=2.5V	VDD=2.3~5.5V	2.4	2.5	2.6	V
V_{LVR4}	LVR 设定电压=3.0V	VDD=2.8~5.5V	2.9	3.0	3.1	V

23.10 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	典型	最大	单位
		VDD	条件				
F_{WDT}	WDT 时钟源	VDD=1.8~5.5V	T_A=25℃	-30%	32	+30%	KHz
		VDD=1.8~5.5V	T_A=－40~85℃	-50%	32	+50%	KHz
T_{EEPROM}	EEPROM 编程时间	5V	F_{HSI}=16MHz	-	4.6	-	ms
		3V	F_{HSI}=16MHz	-	4.6	-	ms
F_{INTRC}	内振频率 16MHz	VDD=4.0~5.5V	TA=25℃	-1.5%	16	+1.5%	MHz
		VDD=2.5~5.5V	TA=25℃	-2.0%	16	+2.0%	MHz
		VDD=1.8~5.5V	TA=25℃	-2.5%	16	+2.5%	MHz
		VDD=4.0~5.5V	TA=－40~85℃	-2.5%	16	+2.5%	MHz
		VDD=2.5~5.5V	TA=－40~85℃	-3.5%	16	+3.5%	MHz
		VDD=1.8~5.5V	TA=－40~85℃	-5.0%	16	+5.0%	MHz

23.11 LSE 特性

（TA= 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	工作电压范围	-	1.8	-	5.5	V
F_{LSE}	LSE振荡频率	-	-	32.768	-	KHz
C1	OSCIN引脚匹配电容	-	-	22	-	pF
C2	OSCOUT引脚匹配电容	-	-	22	-	pF
I_{LSE}	LSE工作电流	VDD=5V C_1=22pF C_2=22pF	-	20	-	uA
		VDD=3V C_1=22pF C_2=22pF	-	8	-	uA
T_{LSE}	LSE稳定时间	VDD=5V C_1=22pF C_2=22pF	-	260	700	ms
		VDD=3V C_1=22pF C_2=22pF		300	1000	ms

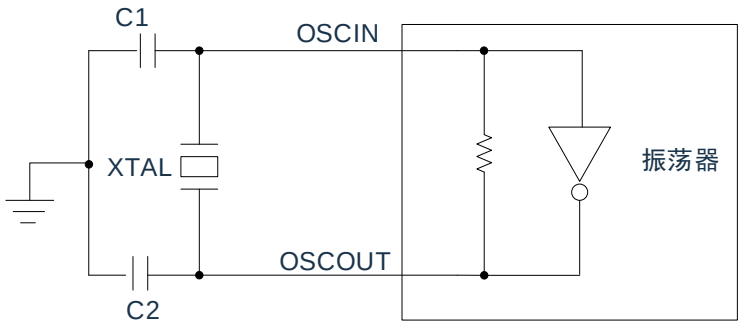


图 23-1：典型应用电路

23.12 EMC 特性

23.12.1 EFT 电气特性

符号	参数	测试条件	等级
V_{EFTB}	在VDD和VSS上通过耦合/去耦合网路施加一个瞬变电压的脉冲群（正向和反向）直到产生功能性错误。VDD和VSS入口有0.1μF旁路电容	T_A = + 25℃, F_{SYS}=8MHz, 符合IEC 61000-4-4	4

注：电快速瞬变脉冲群（EFT）抗扰度性能与系统设计（包括电源结构、电路设计、布局布线、芯片配置、程序结构等）密切相关。上述表格中的 EFT 参数是在 CMS 内部测试平台上所测得的结果，并非适用于所有应用环境，该测试数据仅作为参考。系统设计各方面均可能会对 EFT 性能造成影响，在 EFT 性能要求较高的应用中，设计时应注意尽量避免干扰源影响系统运行，建议分析干扰路径及优化设计以达到最佳的抗扰性能。

23.12.2 ESD 电气特性

符号	参数	测试条件	等级
V_{ESD}	静电放电 (人体放电模式HBM)	TA = + 25℃, ANSI/ESDA/JEDEC JS-001-2023	3A
	静电放电 (器件充电模型CDM)	TA = + 25℃, ANSI/ESDA/JEDEC JS-002-2022	C3

23.12.3 Latch-Up 电气特性

符号	参数	测试条件	测试类型
LU	静态 latch-up	JSED 78F	Class Ⅰ A (T_A = +25℃)

24. 指令

24.1 指令一览表

助记符		操作	指令周期	标志
控制类				
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
数据传送				
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
逻辑运算				
CLRA		清零 ACC	1	Z
SET	[R]	置位数据存储器 R	1	NONE
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				
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
RLA	[R]	数据存储器不带进位循环左移一位，结果存入 ACC	1	NONE
RLR	[R]	数据存储器不带进位循环左移一位，结果存入 R	1	NONE
RRA	[R]	数据存储器不带进位循环右移一位，结果存入 ACC	1	NONE
RRR	[R]	数据存储器不带进位循环右移一位，结果存入 R	1	NONE
递增递减				
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

助记符		操作	指令周期	标志
位操作				
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				
ADDA	[R]	ACC+[R]→ACC	1	C,DC,Z,OV
ADDR	[R]	ACC+[R]→R	1	C,DC,Z,OV
ADDCA	[R]	ACC+[R]+C→ACC	1	Z,C,DC,OV
ADDCR	[R]	ACC+[R]+C→R	1	Z,C,DC,OV
ADDIA	i	ACC+i→ACC	1	Z,C,DC,OV
SUBA	[R]	[R]-ACC→ACC	1	C,DC,Z,OV
SUBR	[R]	[R]-ACC→R	1	C,DC,Z,OV
SUBCA	[R]	[R]-ACC-C→ACC	1	Z,C,DC,OV
SUBCR	[R]	[R]-ACC-C→R	1	Z,C,DC,OV
SUBIA	i	i-ACC→ACC	1	Z,C,DC,OV
HSUBA	[R]	ACC-[R]→ACC	1	Z,C,DC,OV
HSUBR	[R]	ACC-[R]→R	1	Z,C,DC,OV
HSUBCA	[R]	ACC-[R]- $\overline{C}$ →ACC	1	Z,C,DC,OV
HSUBCR	[R]	ACC-[R]- $\overline{C}$ →R	1	Z,C,DC,OV
HSUBIA	i	ACC-i→ACC	1	Z,C,DC,OV
无条件转移				
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
JP	ADD	无条件跳转	2	NONE
条件转移				
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
SZA	[R]	数据存储器 R 送至 ACC，若内容为“0”，则跳过下一条指令	1 or 2	NONE
SZR	[R]	数据存储器 R 内容为“0”，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	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	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

24.2 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 跟 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 跟 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
```

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
```

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
```

HSUBA **[R]**

操作: ACC 减 R，结果放入 ACC

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
LDIA    077H    ;ACC 赋值 077H
LD      R01,A   ;将 ACC 的值(077H)赋给寄存器 R01
LDIA    080H    ;ACC 赋值 080H
HSUBA   R01     ;执行结果：ACC=(80H-77H)=09H
```

HSUBR **[R]**

操作: ACC 减 R，结果放入 R

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
LDIA    077H    ;ACC 赋值 077H
LD      R01,A   ;将 ACC 的值(077H)赋给寄存器 R01
LDIA    080H    ;ACC 赋值 080H
HSUBR   R01     ;执行结果：R01=(80H-77H)=09H
```

HSUBCA **[R]**

操作: ACC 减 R 减 $\overline{C}$，结果放入 ACC

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
LDIA    077H    ;ACC 赋值 077H
LD      R01,A   ;将 ACC 的值(077H)赋给寄存器 R01
LDIA    080H    ;ACC 赋值 080H
HSUBCA  R01     ;执行结果：ACC=(80H-77H-C)=08H(C=0)
                ;          ACC=(80H-77H-C)=09H(C=1)
```

HSUBCR **[R]**

操作: ACC 减 R 减 $\overline{C}$，结果放入 R

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
LDIA    077H    ;ACC 赋值 077H
LD      R01,A   ;将 ACC 的值(077H)赋给寄存器 R01
LDIA    080H    ;ACC 赋值 080H
HSUBCR  R01     ;执行结果：R01=(80H-77H-C)=08H(C=0)
                ;          R01=(80H-77H-C)=09H(C=1)
```

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
```

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
```

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
```

RLA **[R]**

操作: 寄存器 R 不带 C 循环左移一位，结果放入 ACC

周期: 1

影响标志位: 无

举例:

```
LDIA    03H         ;ACC 赋值 03H
LD      R01,A       ;ACC 值赋给 R01,R01=03H
RLA     R01         ;操作结果：ACC=06H
```

RLR **[R]**

操作: 寄存器 R 不带 C 循环左移一位，结果放入 R

周期: 1

影响标志位: 无

举例:

```
LDIA    03H         ;ACC 赋值 03H
LD      R01,A       ;ACC 值赋给 R01,R01=03H
RLR     R01         ;操作结果：R01=06H
```

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
```

RRA **[R]**

操作: 寄存器 R 不带 C 循环右移一位，结果放入 ACC

周期: 1

影响标志位: 无

举例:

```
LDIA    03H         ;ACC 赋值 03H
LD      R01,A       ;ACC 值赋给 R01,R01=03H
RRA     R01         ;操作结果：ACC=81H
```

RRR **[R]**

操作: 寄存器 R 不带 C 循环右移一位，结果放入 R

周期: 1

影响标志位: 无

举例:

```
LDIA    03H         ;ACC 赋值 03H
LD      R01,A       ;ACC 值赋给 R01,R01=03H
RRR     R01         ;操作结果：R01=81H
```

SET **[R]**

操作: 寄存器 R 所有位置 1

周期: 1

影响标志位: 无

举例:

```
SET     R01         ;操作结果：R01=0FFH
```

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
```

SUBCA **[R]**

操作: 寄存器 R 减 ACC 减 C，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
SUBCA   R01     ;操作结果：ACC=80H-77H-C=09H(C=0);
                          ACC=80H-77H-C=08H(C=1);
```

SUBCR **[R]**

操作: 寄存器 R 减 ACC 减 C，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
SUBCR   R01     ;操作结果：R01=80H-77H-C=09H(C=0)
                          R01=80H-77H-C=08H(C=1)
```

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 Add1
```

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

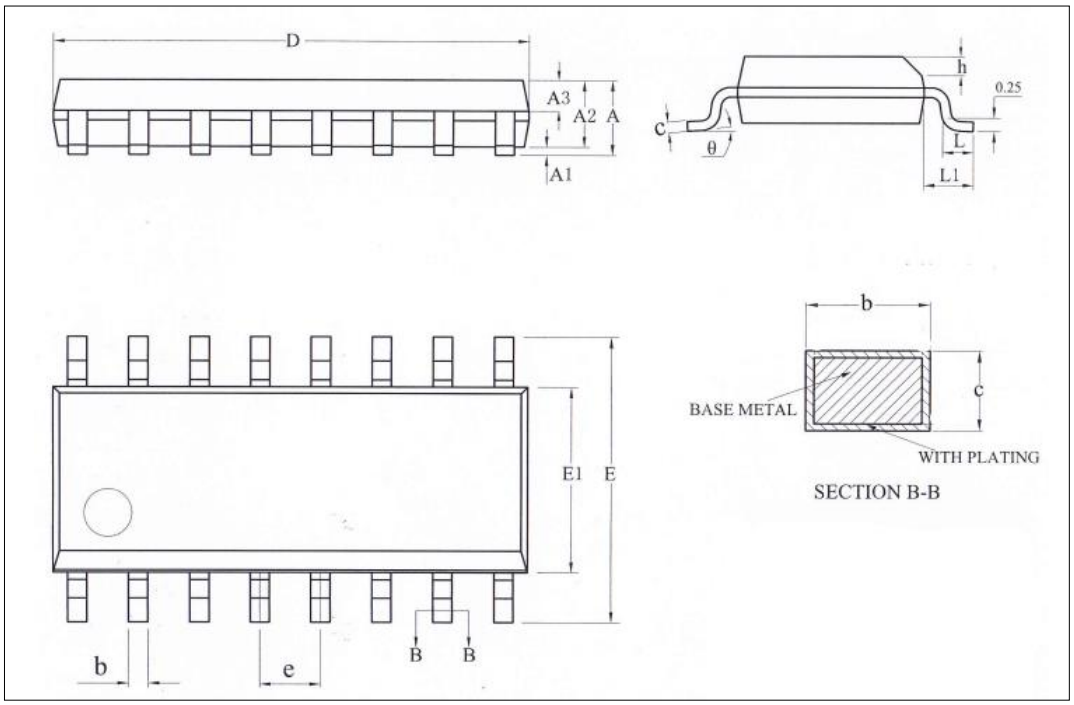
影响标志位: Z

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
```

25. 封装

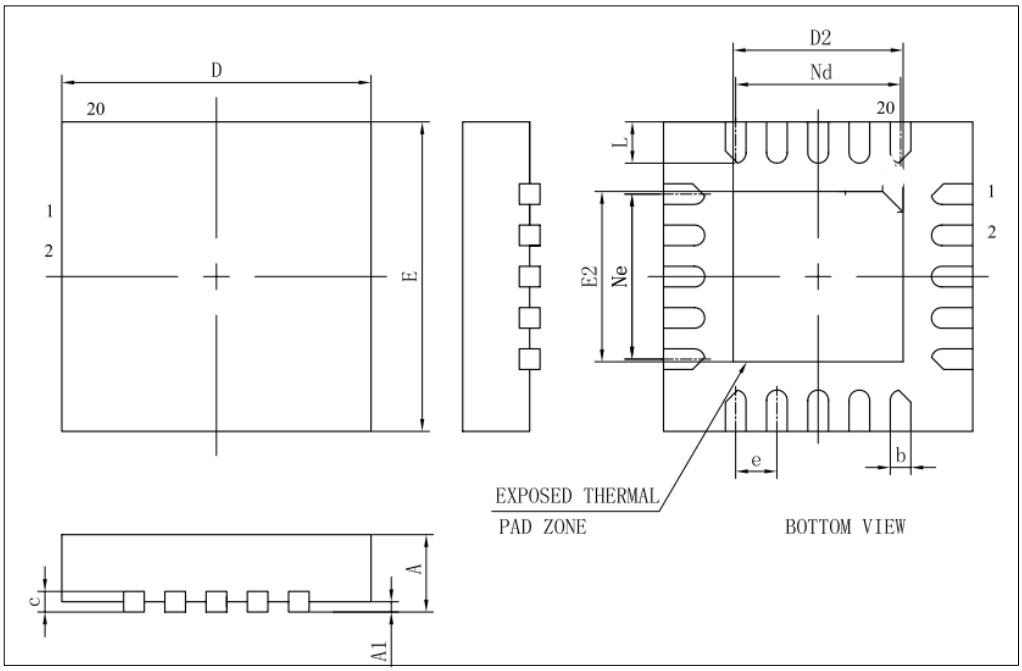
25.1 SOP16



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.85
A1	0.05	-	0.25
A2	1.30	-	1.60
A3	0.60	-	0.71
b	0.356	-	0.51
c	0.20	-	0.26
D	9.70	-	10.10
E	5.80	6.00	6.20
E1	3.70	-	4.10
e	1.27BSC		
h	0.25	-	0.50
L	0.40	-	0.80
L1	1.05REF		
θ	0	-	8°

注意：封装尺寸不包括模的毛边凸起或门毛刺。

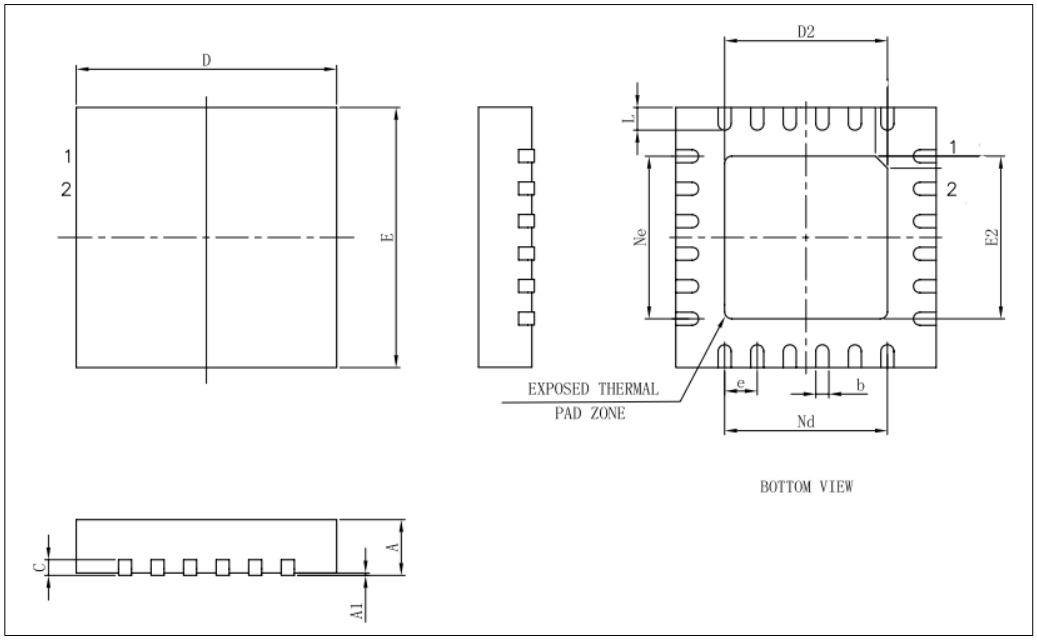
25.2 QFN20（3*3*0.75-0.40mm）



Symbol	Millimeter		
	Min	Nom	Max
A	0.65	0.75	0.85
A1	-	0.02	0.05
b	0.15	0.20	0.25
c	0.18	0.20	0.25
D	2.90	3.00	3.10
D2	1.55	-	2.00
e	0.40BSC		
Ne	1.60BSC		
Nd	1.60BSC		
E	2.90	3.00	3.10
E2	1.55	-	2.00
L	0.20	-	0.50

注意：封装尺寸不包括模的毛边凸起或门毛刺

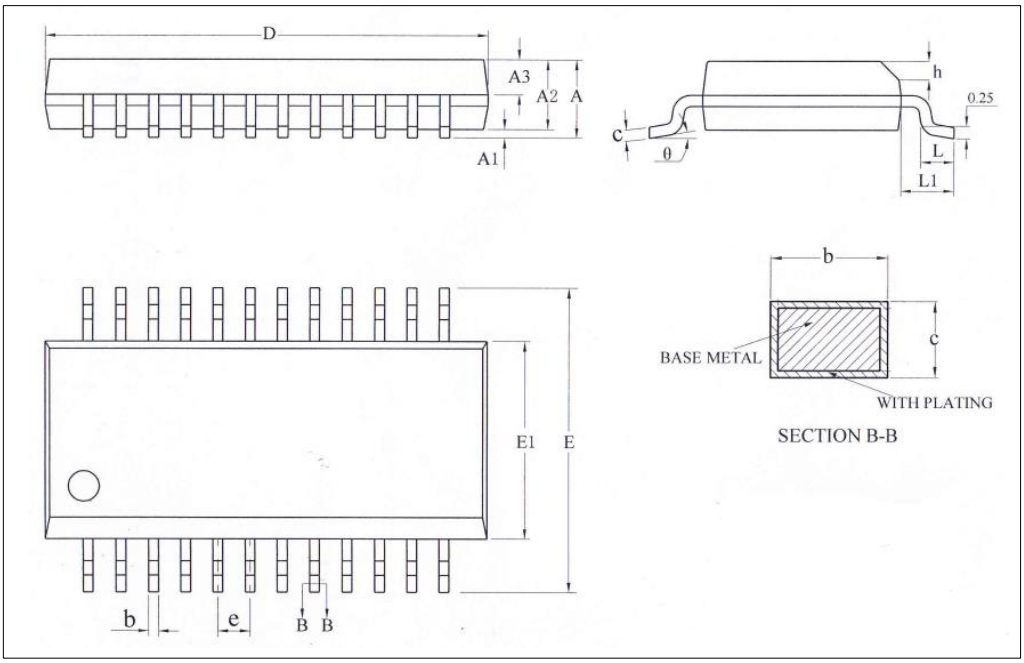
25.3 QFN24（4*4*0.75-0.5mm）



Symbol	Millimeter		
	Min	Nom	Max
A	0.70	0.75	0.80
A1	-	0.02	0.05
b	0.18	0.25	0.30
c	0.18	0.20	0.25
D	3.90	4.00	4.10
D2	2.20	-	2.80
e	0.50BSC		
Ne	2.50BSC		
Nd	2.50BSC		
E	3.90	4.00	4.10
E2	2.20	-	2.80
L	0.30	0.40	0.50
h	0.25	-	0.40

注意：封装尺寸不包括模的毛边凸起或门毛刺。

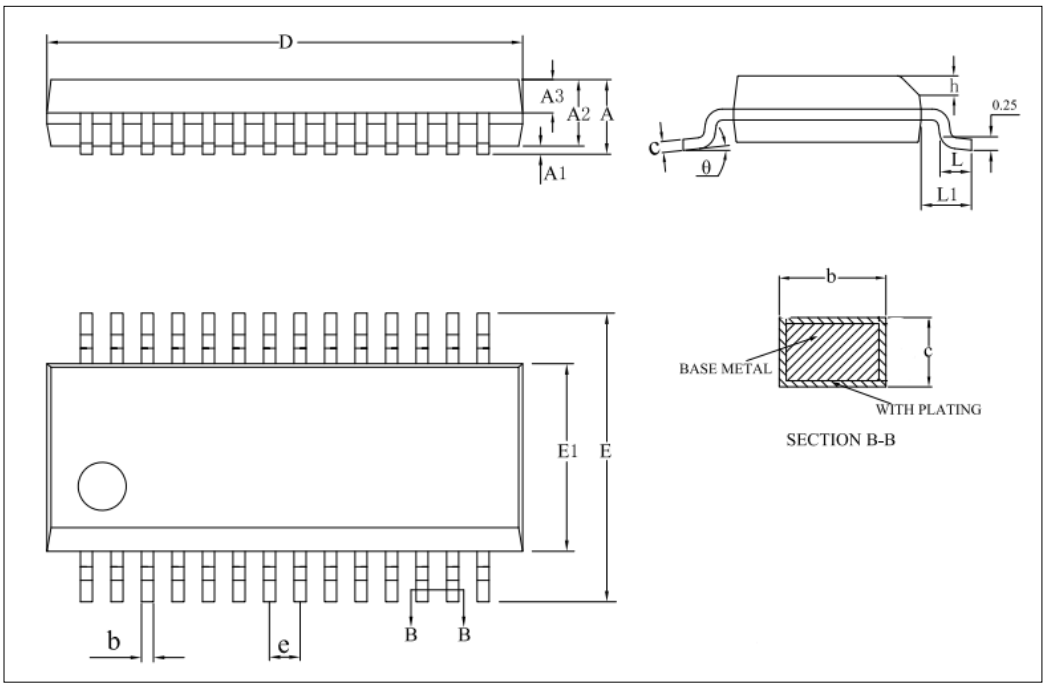
25.4 SSOP24（0.635mm）



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.80
A1	0.10	0.15	0.25
A2	1.30	-	1.55
A3	0.60	0.65	0.70
b	0.20	-	0.31
c	0.20	-	0.24
D	8.53	-	8.75
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	0.635BSC		
h	0.30	-	0.50
L	0.406	-	0.889
L1	1.05REF		
θ	0	-	8°

注意：封装尺寸不包括模的毛边凸起或门毛刺。

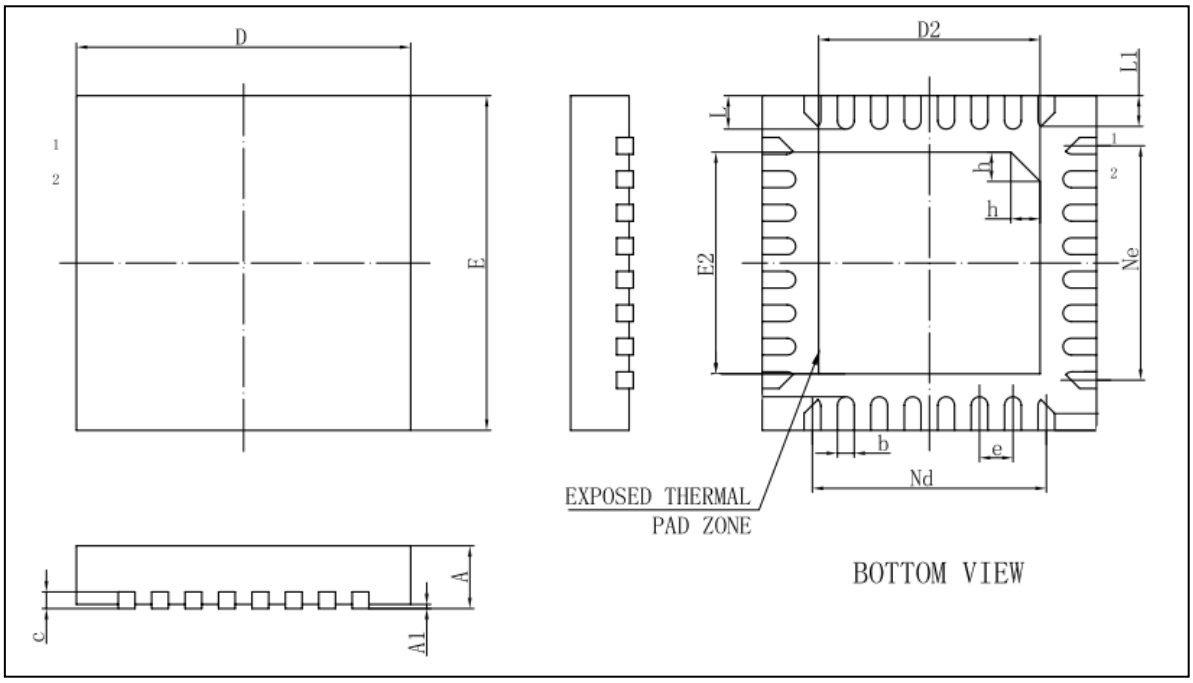
25.5 SSOP28（0.635mm）



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.725
A1	0.05	-	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.23	-	0.31
c	0.20	-	0.24
D	9.80	9.90	10.00
E	5.80	6.00	6.20
E1	3.80	3.90	4.00
e	0.635BSC		
h	0.25	-	0.50
L	0.50	-	0.80
L1	1.05REF		
θ	0°	-	8°

注意：封装尺寸不包括模的毛边凸起或门毛刺。

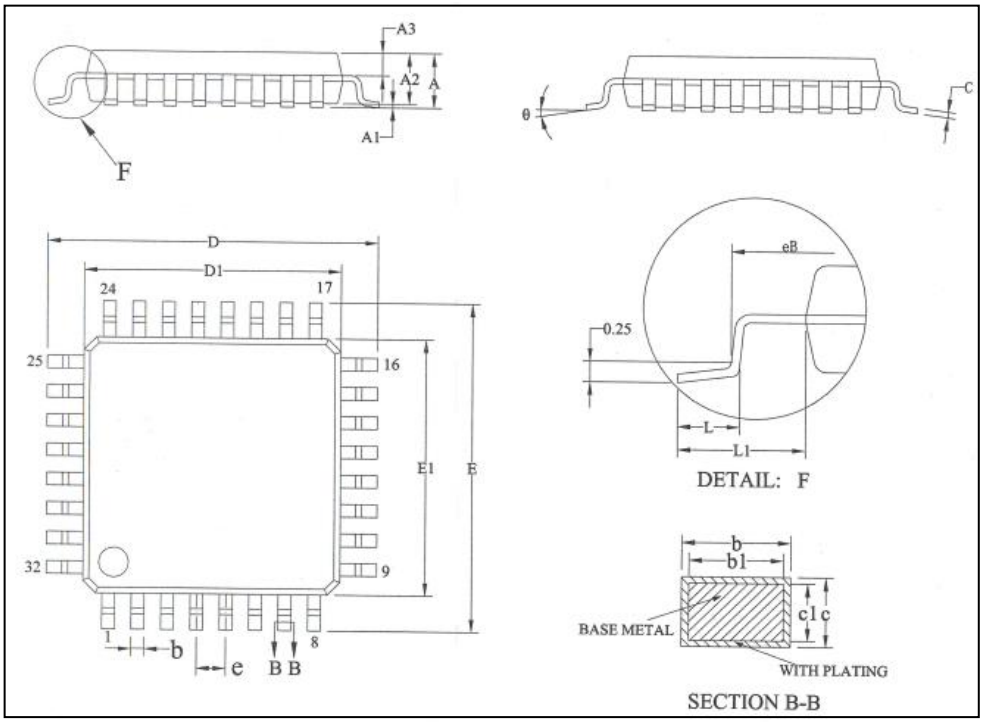
25.6 QFN32（4*4*0.75-0.40mm）



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。

25.7 LQFP32（7*7）



Symbol	Millimeter		
	Min	Nom	Max
A	-	-	1.60
A1	0.05	-	0.15
A2	1.35	1.40	1.45
A3	0.59	0.64	0.69
b	0.32	-	0.43
b1	0.31	-	0.39
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	0.80BSC		
L	0.45	-	0.75
L1	1.00REF		
θ	0	-	7°

注意：封装尺寸不包括模的毛边凸起或门毛刺。

26. 版本修订说明

版本号	时间	修订内容
V1.0.0	2025年3月	正式版本 1) 订正异步模式下的波特率表 2) 订正LCD/LED控制寄存器描述 3) 更新1.4.6章节标题
V1.0.1	2025年6月	1) 订正16MHz/2T的工作电压范围 2) 增加EMC特性章节