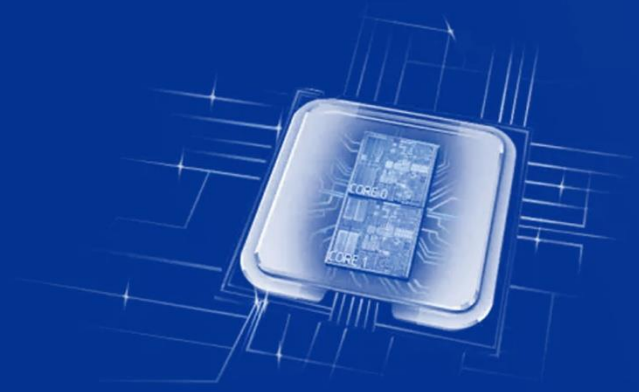
SCMCU



用户手册

SC8F301X 系列
触摸型 FLASH MCU

(包含了SC8F3013S、SC8F3013、SC8F3012、SC8F3010)

V1.0

请注意以下有关芯联发公司知识产权政策：

（一）芯联发公司已申请了专利，享有绝对的合法权益。与芯联发公司 MCU 或其他产品有关的专利权并未被同意授权使用，任何经由不当手段侵害芯联发公司专利权的公司、组织或个人，芯联发公司将采取一切可能的法律行动，遏止侵权者不当的侵权行为，并追讨芯联发公司因侵权行为所受的损失、或侵权者所得的不法利益。

（二）芯联发公司的名称和标识都是芯联发公司的注册商标。

（三）芯联发公司保留对规格书中产品在可靠性、功能和设计方面的改进作进一步说明的权利。然而芯联发公司对于规格内容的使用不负责任。文中提到的应用其目的仅仅是用来做说明，芯联发公司不保证和不表示这些应用没有更深入的修改就能适用，也不推荐它的产品使用在会由于故障或其它原因可能会对人身造成危害的地方。芯联发公司的产品不授权适用于救生、维生器件或系统中作为关键器件。芯联发公司拥有不事先通知而修改产品的权利。

目录

1. 产品概述

1.1 功能特性及选型表

型号	SC8F3010	SC8F3012	SC8F3013	SC8F3013S
脚位	8	16	20	24
封装形式	SOP8,DIP8	SOP16,DIP16	SOP20,DIP20	SSOP24
I/O	5+1	13+1	17+1	17+1
ROM(FLASH)	2K*16	2K*16	2K*16	2K*16
RAM	176 字节	176 字节	176 字节	176 字节
内部振荡	8MHZ/4MHZ/2MHZ/1MHZ/500KHZ/250KHZ/125KHZ/31KHZ	8MHZ/4MHZ/2MHZ/1MHZ/500KHZ/250KHZ/125KHZ/31KHZ	8MHZ/4MHZ/2MHZ/1MHZ/500KHZ/250KHZ/125KHZ/31KHZ	8MHZ/4MHZ/2MHZ/1MHZ/500KHZ/250KHZ/125KHZ/31KHZ
触摸按键	4	12	16	16
ADC	3	11	15	15
CCP	1	2	2	2
TIMER(8Bit)	2	2	2	2
TIMER(16Bit)	1	1	1	1
外部中断	2	2	2	2
内部中断	5	6	6	6
LCD 驱动 COM 口	0	4	4	4
IO 内部上拉电阻	PORTA\B\C	PORTA\B\C	PORTA\B\C	PORTA\B\C
唤醒	PORTA	PORTA	PORTA	PORTA
LVR	有	有	有	有
WDT	有	有	有	有
查表功能	有	有	有	有
堆栈	8 级	8 级	8 级	8 级
指令	49 条	49 条	49 条	49 条
工作电压	2.5V-5.5V	2.5V-5.5V	2.5V-5.5V	2.5V-5.5V
工作温度范围	-40℃-85℃	-40℃-85℃	-40℃-85℃	-40℃-85℃

表 1.1 特性及选型表

1.2 系统结构框图

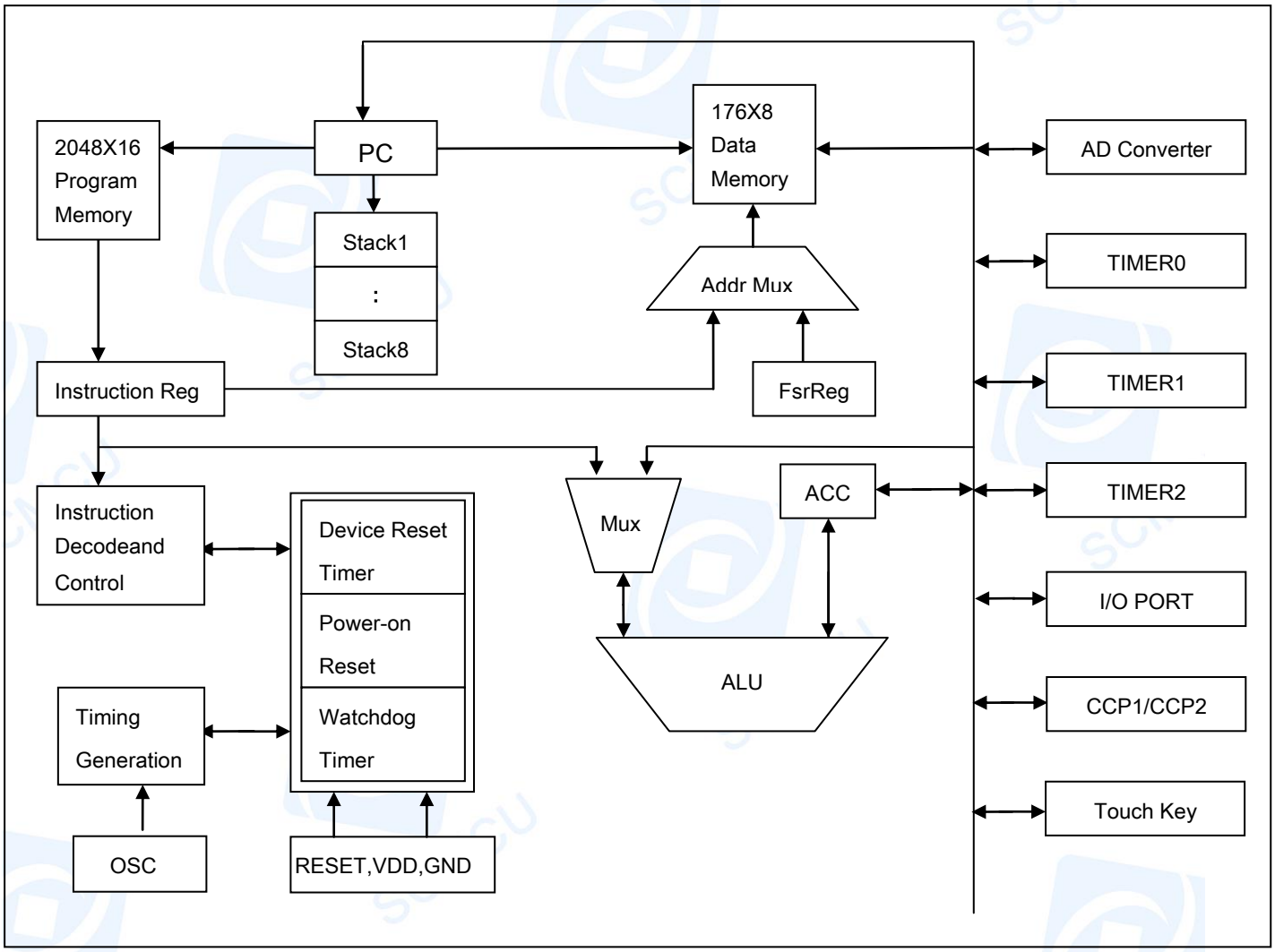


图 1.1 系统结构框图

1.3 管脚分布

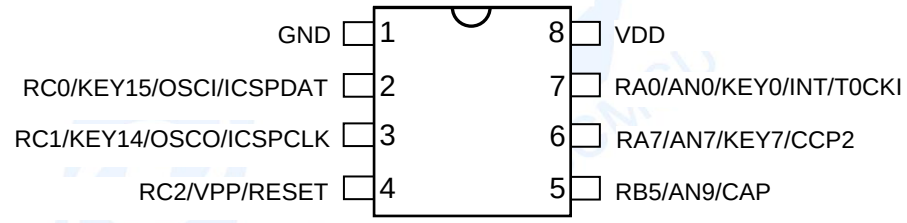


图 1.2 SC8F3010

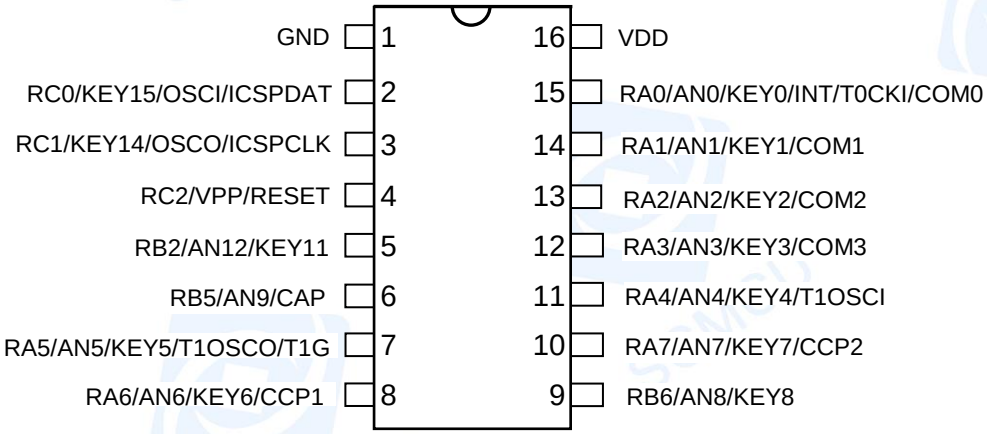


图 1.4 SC8F3012

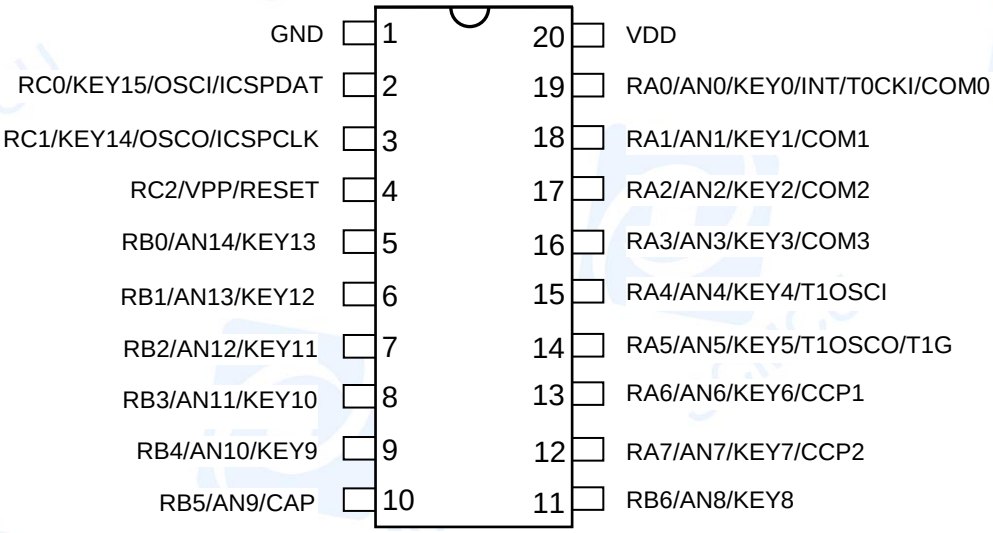


图 1.5 SC8F3013

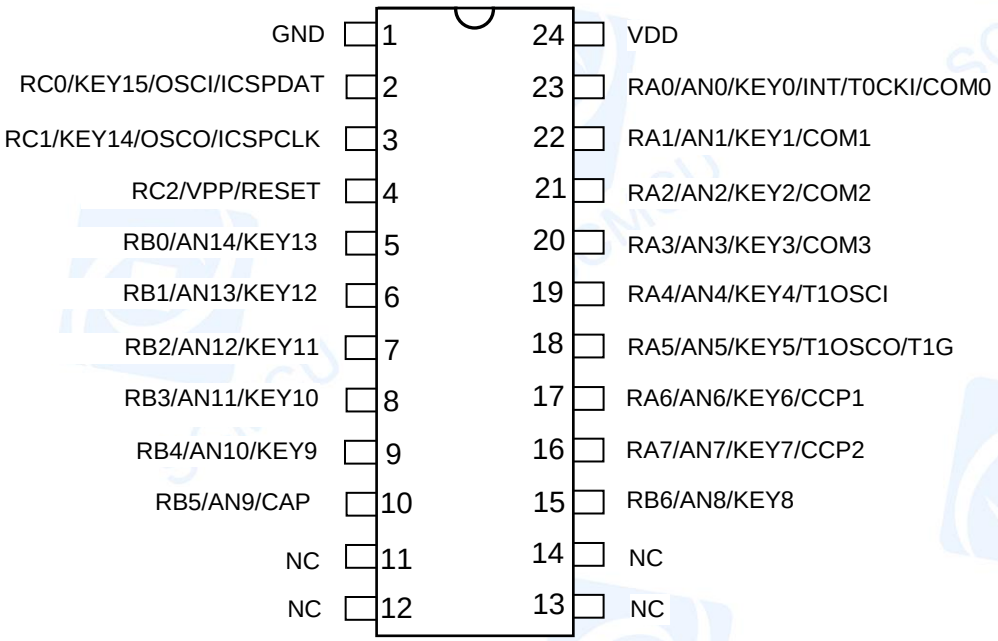


图 1.6　SC8F3013S

1.4 功能描述

管脚名称	IO 类型	管脚说明	共享引脚
RA0-RA7	I/O	可编程为：输入口，推挽输出口，带上拉电阻功能、电平变化中断功能； A/D 转换的模拟输入口，外部中断输入口，触摸按键输入口，捕捉/比较/PWM 输出口，带休眠唤醒功能，TIMER1 门控信号输入脚，TIMER0 外部时钟输入脚,LCD 驱动 COM 口	AN0-AN7 KEY0-KEY7 COM0-COM3 INT,T1OSCI,T1OSCO T1G,CCP1,CCP2,T0CKI
RB0-RB6	I/O	可编程为：输入口，A/D 转换的模拟输入口，推挽输出口，带上拉电阻功能；触摸按键灵敏度电容口，触摸按键输入口。	CAP,AN8-AN14 KEY8-KEY13
RC0-RC1	I/O	可编程为：输入口，推挽输出口，带上拉电阻功能；编程时钟输入口，触摸按键输入口，编程数据输出/输入口，晶振输入/输出口	KEY14-KEY15 ICSPDAT,ICSPCLK,OSCI OSCO
RC2	I/O	可编程为：输入口，开漏输出口，外部复位口，编程 VPP 输入口	RESET,VPP
VDD,GND	P	电源电压输入脚，接地脚	--
RESET	I	外部复位输入口	RC2
KEY0-KEY15	--	电容式触摸按键输入口	RA0-RA7,RB0-RB6,RC1-RC2
CAP	--	触摸按键灵敏度电容输入口	RB5
VPP	I	编程 VPP 输入口	RC2
T0CKI	I	TIMER0 外部时钟输入脚	RA0
AN0-AN14	I	12 位 ADC 输入脚	RA0-RA7,RB0-RB6
INT	I	外部中断输入口	RA0
T1G	I	TIMER1 门控信号输入脚	RA5
OSCI,OSCO	I/O	晶振输入/输出口	RC0,RC1
T1OSCI,T1OSCO	I/O	32.768K 晶振输入/输出口	RA4,RA5
CCP1	I/O	捕捉/比较/PWM1	RA6
CCP2	I/O	捕捉/比较/PWM2	RA7
ICSPDAT	I/O	编程数据输入输出	RC0
ICSPCLK	I	编程时钟输入	RC1

表 1.2 管脚功能描述

1.5 系统配置寄存器

系统配置寄存器(CONFIG)是 MCU 初始条件的 FLASH 选项。它只能被芯联发公司烧写器烧写，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 打开看门口定时器
- DISABLE 关闭看门狗定时器

2、PROTECT(加密)

- DISABLE FLASH 代码不加密
- ENABLE FLASH 代码加密，加密后读出来的值将不确定

3、OSC TIME(起振时间)

- 18ms
- 9ms
- 2ms
- 560us

4、LVR (低压侦测电路)

- ENABLE 打开低压侦测电路，选择内部复位，同时 RC2 口作为普通 IO
- DISABLE 关闭低压侦测电路，选择外部复位，同时 RC2 口作为复位口(低电平复位)

5、LVR _SEL(低压侦测电压选择)

- 1.8V(不建议使用)
- 2.5V
- 3.6V

6、OSC(振荡方式选择)

- INTRC 内部 RC 振荡
- XT 外表晶体振荡

7、RES_OUT(RC2 口开漏输出选择)

- DISABLE 禁止 RC2 口作为开漏输出口
- ENABLE 允许 RC2 口作为开漏输出口

注: 1、LVR_SEL 电压为设计电压，实际芯片的复位电压可能会有所浮动。

1.6 在线串行编程

可在最终应用电路中对 SC8F301X 单片机进行串行编程。编程可以简单地通过以下 5 根线完成:

- 电源线
- 接地线
- 编程电压线
- 数据线
- 时钟线

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

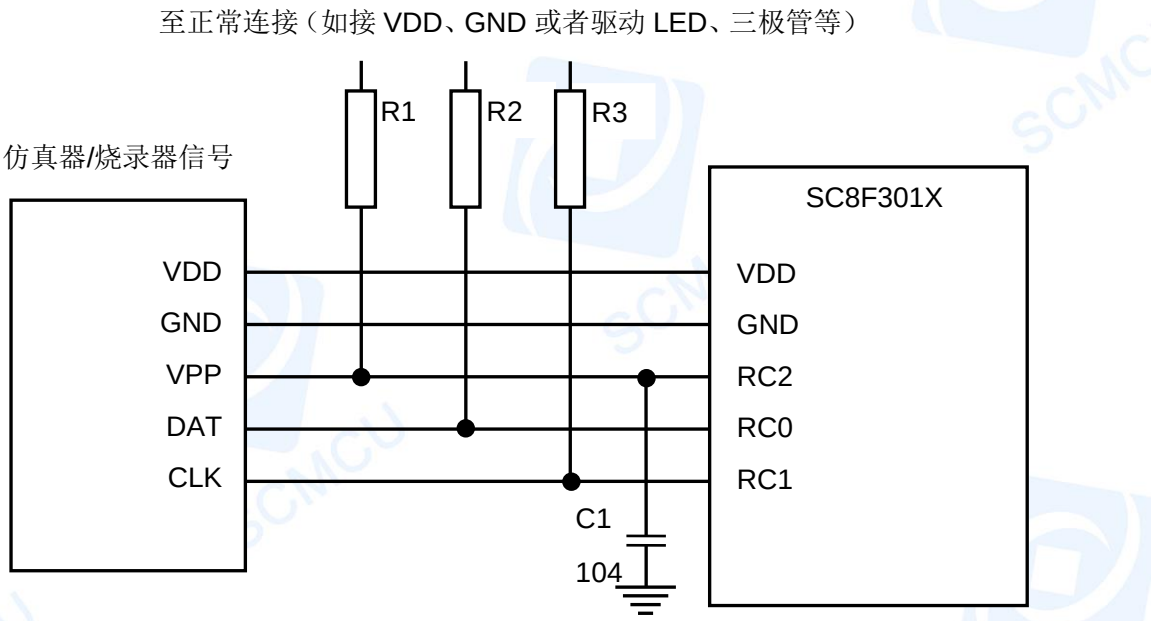


图 1.6　典型的在线串行编程连接方式

上图中，R1、R2、R3 为电气隔离器件，常以电阻代替，其阻值如下:

- R1≥30K,R2≥4.7K,R3≥4.7K。
- C1 为 VPP 滤波电容，建议为 0.1uF(104)大小的瓷片电容。
- 如果烧录的通信线太长，可以在 DAT、CLK 管脚接一个对地电容，以增加通信的稳定性，其容值不能大于 100pF(101)。

2. 中央处理器（CPU）

2.1 内存

2.1.1 程序内存

- FLASH：2K

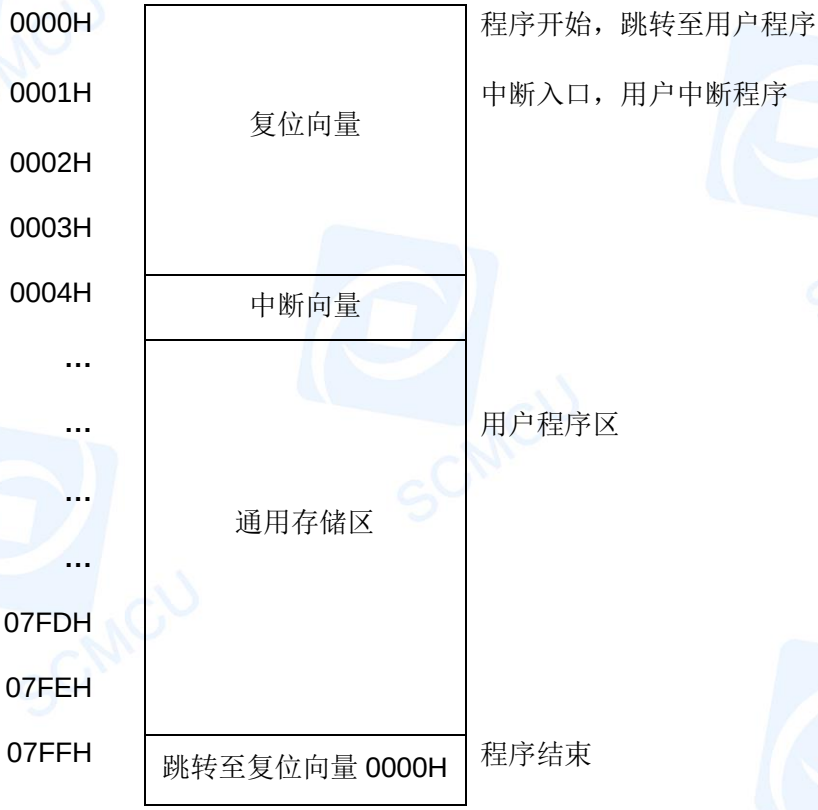


图 2.1.1 2K-FLASH 地址表

2.1.1.1 复位向量(0000H)

SC8F301X系列单片机具有一个字长的系统复位向量(0000H)。具有以下四种复位方式:

- 上电复位
- 看门狗复位
- 外部复位
- 低压复位(LVR)

发生上述任一种复位后，程序将从0000H处重新开始执行，系统寄存器也都将恢复为默认值。根据STATUS寄存器中的PD和TO标志位的内容可以判断系统复位方式。下面一段程序演示了如何定义FLASH中的复位向量。

★ 例：定义复位向量

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

注：由于 SC8F301X 系列芯片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:
        MOVWF   W_BAK       ;保存 W 至自定义寄存器 W_BAK
        SWAPF   STATUS,0    ;状态寄存器 STATUS 高低半字节互换
        MOVWF   STATUS_BAK  ;保存至自定义寄存器 STATUS_BAK
        RETURN              ;返回
```

★ 例：中断出口恢复现场

```
POP:
        SWAPF   STATUS_BAK,0 ;将保存至 STATUS_BAK 的数据高低半字节互换给 W
        MOVWF   STATUS       ;将 W 的值给状态寄存器 STATUS
        SWAPF   W_BAK,1      ;将保存至 W_BAK 的数据高低半字节互换
        SWAPF   W_BAK,0      ;将保存至 W_BAK 的数据高低半字节互换给 W
        RETURN               ;返回
```

2.1.1.3 查表

芯片具有查表功能，FLASH空间的任何地址都可做为查表使用。

相关指令:

- TABLE f把表格内容的低字节送给寄存器f，高字节送到寄存器TABLE_DATAH。
- TABLEA把表格内容的低字节送给累加器W，高字节送到寄存器TABLE_DATAH。

相关寄存器:

- TABLE_SPH(99H)可擦写寄存器，用来指明表格高3位地址。
- TABLE_SPL(9AH)可擦写寄存器，用来指明表格低8位地址。
- TABLE_DATAH(9BH)只读寄存器，存放表格高字节内容。

注: 在查表之前要先把表格地址写入TABLE_SPH和TABLE_SPL中。如果主程序和中断服务程序都用到查表指令，主程序中的TABLE_SPH的值可能会因为中断中执行的查表指令而发生变化，产生错误。也就是说要避免在主程序和中断服务程序中都使用查表指令。但如果必须这样做的话，我们可以在查表指令前先将中断禁止，在查表结束后再开放中断，以避免发生错误。

表格调用流程图

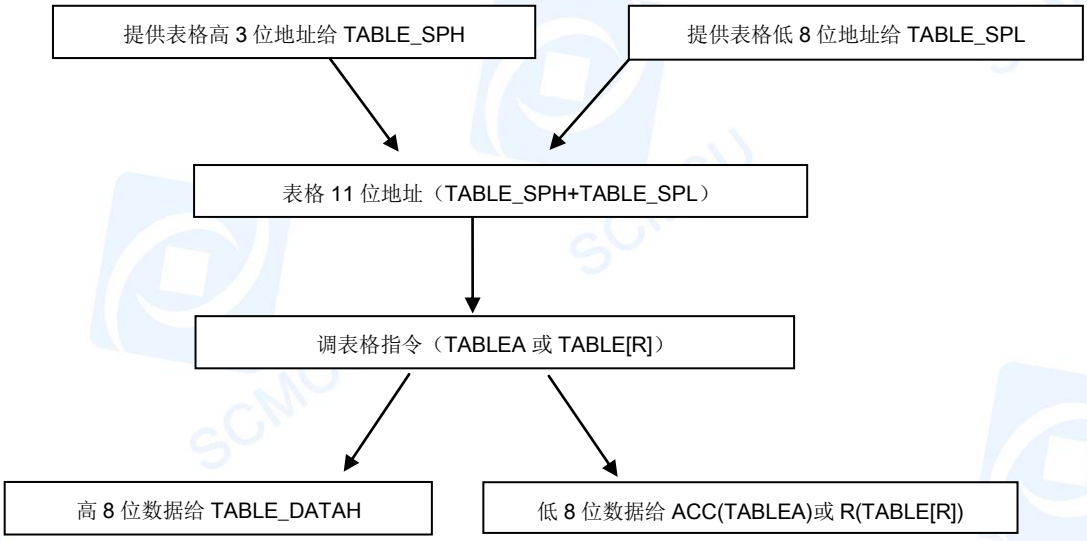


图 2.1.2　表格调用的流程图

下面例子给出了如何在程序中调用表格。

```
...                             ;上接用户程序
MOVLW       02H                 ;表格低位地址
MOVWF       TABLE_SPL
MOVLW       06H                 ;表格高位地址
MOVWF       TABLE_SPH
TABLE       R01                 ;表格指令，将表格低8位(56H)给自定义寄存器R01
MOVF        TABLE_DATAH,0       ;将查表结果的高8位(34H)给累加器W
MOVWF       R02                 ;将W值(34H)给自定义寄存器R02
...                             ;用户程序
ORG         0600H               ;表格起始地址
DW          1234H               ;0600H地址表格内容
DW          2345H               ;0601H地址表格内容
DW          3456H               ;0602H地址表格内容
DW          0000H               ;0603H地址表格内容
```

2.1.1.4 跳转表

跳转表能够实现多地址跳转功能。由于PCL和W的值相加即可得到新的PCL,因此，可以通过对PCL加上不同的W值来实现多地址跳转。W值若为N,PCL+W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＋W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W的值轻松实现多地址的跳转。PCLATH为PC高位缓冲寄存器，对PCL操作时，必须先对PCLATH进行赋值。

★ 例：正确的多地址跳转程序示例

```
FLASH 地址
                MOVLW   01H
                MOVWF   PCLATH      ;对 PCLATH 进行赋值
0110H:          ADDWF   PCL,1       ;W+PCL
0111H:          GOTO    LOOP1       ;W=0,跳转至 LOOP1
0112H:          GOTO    LOOP2       ;W=1,跳转至 LOOP2
0113H:          GOTO    LOOP3       ;W=2,跳转至 LOOP3
0114H:          GOTO    LOOP4       ;W=3,跳转至 LOOP4
0115H:          GOTO    LOOP5       ;W=4,跳转至 LOOP5
0116H:          GOTO    LOOP6       ;W=5,跳转至 LOOP6
```

★ 例：错误的多地址跳转程序示例

```
FLASH 地址
                CLRF    PCLATH
                ...     ...
00FCH:          ADDWF   PCL,1       ;W+PCL
00FDH:          GOTO    LOOP1       ;W=0,跳转至 LOOP1
00FEH:          GOTO    LOOP2       ;W=1,跳转至 LOOP2
00FFH:          GOTO    LOOP3       ;W=2,跳转至 LOOP3
0100H:          GOTO    LOOP4       ;W=3,跳转至 0000H
0101H:          GOTO    LOOP5       ;W=4,跳转至 0001H
0102H:          GOTO    LOOP6       ;W=5,跳转至 0002H
```

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，一定要注意该段程序一定不能放在FLASH空间的分页处。

2.1.2 数据存储器

★ RAM: 256字节（128字节）

地址	BANK0 STATUS.5=0	BANK1 STATUS.5=1	地址
0000H	系统寄存器区	系统寄存器区	0080H
0001H			0081H
…			…
…			…
001FH			009FH
0020H	通用寄存器区 96 字节	通用寄存器区 80 字节	00A0H
0021H			00A1H
0022H			00A2H
…			…
…		快速存储区 70H-7FH	…
007EH			00F0H
007FH			00FFH

图2.1.3 256字节RAM

数据存储器由 256×8 位组成，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从 00H-1FH，80-9FH。

注: 08H，09H，0DH，14H，18H，19H，1AH，88H，89H，8DH，9DH这11个地址没有特殊功能，可以当作通用数据寄存器使用

2.1.2.1 通用数据存储器

RAM 的 0020H～007FH 地址，00A0H～00FFH 地址以及 08H，09H，0DH，14H，18H，19H，1AH，88H，89H，8DH，9DH 这 11 个地址，属于用户可自由定义的通用寄存器区，在此区域的寄存器上电为随机值。当系统上电工作后，若发生意外复位(非上电复位)，此区域寄存器保持原来值不变。

2.1.2.2 系统专用数据存储器

特殊功能寄存器 Bank0		
地址	名称	说明
00H	INDF	间接寻址寄存器
01H	TMR0	TIMER0 数据寄存器
02H	PCL	程序指针 PC 低 8 位
03H	STATUS	系统状态标志寄存器
04H	FSR	间接寻址指针
05H	PORTA	PORTA 数据寄存器
06H	PORTB	PORTB 数据寄存器
07H	PORTC	PORTC 数据寄存器
0AH	PCLATH	程序计数器高 3 位的写缓冲器
0BH	INTCON	中断控制寄存器
0CH	PIR1	外设中断请求寄存器
0EH	TMR1L	16 位 TIMER1 寄存器低 8 位数据寄存器
0FH	TMR1H	16 位 TIMER1 寄存器高 8 位数据寄存器
10H	T1CON	TIMER1 控制寄存器
11H	TMR2	TIMER2 数据寄存器
12H	T2CON	TIMER2 控制寄存器
13H	PR2	TIMER2 周期寄存器
15H	CCPR1L	捕捉/比较/PWM 寄存器 1 的低 8 位
16H	CCPR1H	捕捉/比较/PWM 寄存器 1 的高 8 位
17H	CCP1CON	CCP1 预分频器
1BH	CCPR2L	捕捉/比较/PWM 寄存器 2 的低 8 位
1CH	CCPR2H	捕捉/比较/PWM 寄存器 2 的高 8 位
1DH	CCP2CON	CCP2 预分频器
1EH	ADRESH	A/D 结果寄存器高 8 位
1FH	ADCON0	AD 控制寄存器
特殊功能寄存器 Bank1		

地址	名称	说明
80H	INDF	间接寻址寄存器
81H	OPTION_REG	预分频器
82H	PCL	程序指针 PC 低 8 位
83H	STATUS	系统状态标志寄存器
84H	FSR	间接寻址指针
85H	TRISA	PORTA 方向寄存器
86H	TRISB	PORTB 方向寄存器
87H	TRISC	PORTC 方向寄存器
8AH	PCLATH	程序计数器高 3 位的写缓冲器
8BH	INTCON	中断控制寄存器
8CH	PIE1	外设中断允许寄存器
8EH	WDTCON	看门狗定时器控制寄存器
8FH	OSCCON	振荡器控制寄存器
90H	OSCTUNE	振荡器调节寄存器
91H	KEYCON0	触摸按键控制寄存器
92H	KEYCON1	触摸按键控制寄存器
93H	KEYDATAL	触摸按键结果寄存器低 8 位
94H	KEYDATAH	触摸按键结果寄存器高 8 位
95H	WPUA	PORTA 上拉电阻寄存器
96H	WPUB	PORTB 上拉电阻寄存器
97H	WPUC	PORTC 上拉电阻寄存器
98H	IOCA	PORTA 电平变化中断寄存器
99H	TABLE_SPH	表格高位指针
9AH	TABLE_SPL	表格低位指针
9BH	TABLE_DATAH	表格高位数据
9CH	LCDCON	LCD 控制寄存器
9EH	ADRESL	A/D 结果寄存器的低字节
9FH	ADCON1	AD 控制寄存器

表 2.0　系统专用数据存储器表

2.2 寻址方式

2.2.1 直接寻址

◆ 通过工作寄存器(W)来对 RAM 进行操作。

★ 例: W 的值送给 30H 寄存器

```
MOVWF   30H
```

★ 例: 30H 寄存器的值送给 W

```
MOVF    30H,0
```

2.2.2 立即寻址

◆ 把立即数传给工作寄存器(W)

★ 例：立即数 12H 送给 W

```
MOVLW   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，它会根据 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，将导致一个空运作。以下例子说明了程序中间接寻址的用法。

★ 例: FSR 及 INDF 的应用

```
        MOVLW   30H
        MOVWF   FSR             ;间接寻址指针指向 30H
        CLRF    INDF            ;清零 INDF 实际是清零 FSR 指向的 30H 地址 RAM
```

★ 例: 间接寻址清 RAM(30H-7FH)举例:

```
        MOVLW   2FH
        MOVWF   FSR             ;间接寻址指针指向 2FH
LOOP:
        INCF    FSR,1           ;地址加 1,初始地址为 30H
        CLRF    INDF            ;清零 FSR 所指向的地址
        MOVLW   7FH
        SUBWF   FSR,0
        BTFSS   STATUS,C        ;一直清零至 FSR 地址为 7FH
        GOTO    LOOP
```

2.3 堆栈

SC8F301X 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。

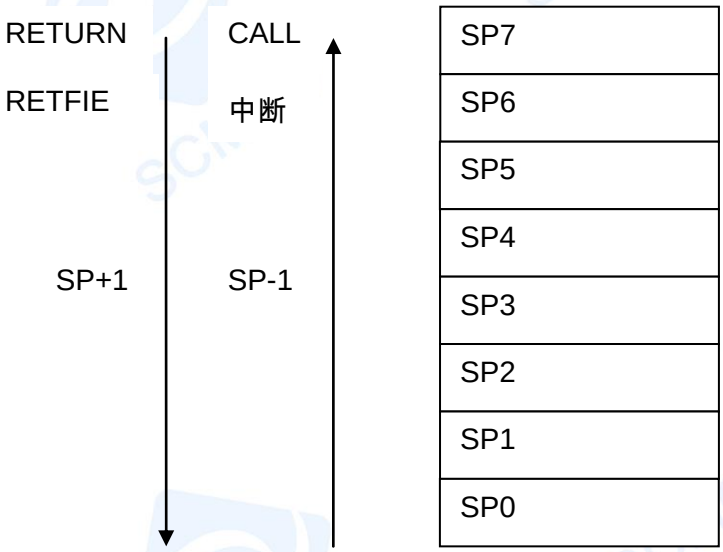


图 2.3.1 堆栈缓存器

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

2.4 工作寄存器(W)

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、移位元及逻辑运算；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。

W 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 W 应用

★ 例：用 W 做数据传送

```
MOVF    R01,0    ;将寄存器 R01 的值赋给 W
MOVWF   R02      ;将 W 的值赋给寄存器 R02
```

★ 例：用 W 做立即寻址目标操作数

```
MOVLW   30H      ;给 W 赋值 30H
ANDLW   30H      ;将当前 W 的值跟立即数 30H 进行“与”操作，结果放入 W
XORLW   30H      ;将当前 W 的值跟立即数 30H 进行“异或”操作，结果放入 W
```

★ 例：用 W 做双操作数指令的第二操作数

```
SUBWF   R01,0    ;R01-W,结果放入 W
SUBWF   R01,1    ;R01-W,结果放入 R01
```

2.5 程序状态寄存器(STATUS)

寄存器 STATUS 中包含 ALU 运算状态信息、系统复位状态信息、RAM 分页选择。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，CLRF 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 BCF、BSF、SWAPF 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	---	---	RP0	TO	PD	Z	DC	C
读写	---	---	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	1	1	X	X	X

表2.2　程序状态寄存器（STATUS）

BIT7-6　**未用**

BIT5　**RP0：寄存器存储区选择位**
1 = Bank 1
0 = Bank 0

BIT4　**TO：超时位**
1 =上电或是执行了CLRWDT指令或SLEEP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SLEEP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**（ADDWF、ADDLW、SUBLW或SUBWF指令）
1 = 发生了结果的第4低位向高位进位
0 = 结果的第4低位没有向高位进位

BIT0　**C：进位/ 借位位**（ADDWF、ADDLW、SUBLW或SUBWF指令）
1 = 结果的最高位发生了进位
0 = 结果的最高位没有发生进位

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
SLEEP 指令	1	0
CLRWDT 指令	1	1
休眠	1	0

表 2.3　影响 TO/PD 的事件表

TO	PD	复位原因
0	0	WDT 溢出唤醒休眠 MCU
0	1	WDT 溢出非休眠态
1	0	按键或外部复位唤醒休眠 MCU
1	1	电源上电

表 2.4　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：

- TIMER0/WDT 预分频器，
- TIMER0，
- PORTA 上拉电阻控制。

预分频器控制寄存器OPTION_REG（81H）

81H	BIT7	BIT6	BIT5	BIT4	BIT3	BIT2	BIT1	BIT0
OPTION_REG	RAPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
读写	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

表 2.5 预分频寄存器

BIT7 **RAPU:** PORTA 上拉使能位
1=禁止 PORTA 上拉
0=由端口的各个锁存值使能 PORTA 上拉

BIT6 **INTEDG:** 触发中断的边沿选择位
1=INT 引脚上升沿触发中断
0=INT 引脚下降沿触发中断

BIT5 **T0CS:** TIMER0 时钟源选择位
1=T0CKI 引脚上的跳变沿
0=内部指令周期时钟（FOSC/4）

BIT4 **T0SE:** TIMER0 时钟源边沿选择位
1=在 T0CKI 引脚信号从高电平跳变到低电平时递增
0=在 T0CKI 引脚信号从低电平跳变到高电平时递增

BIT3 **PSA:** 预分频器分配位
1=预分频器分配给 WDT
0=预分频器分配给 TIMER0 模块

BIT2～BIT0 **PS2～PS0:** 预分配参数配置位

PS2	PS1	PS0	TIMER0 分频比	WDT 分频比
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

预分频寄存器实际上是一个8位的计数器，用于监视寄存器WDT时，是作为一个后分频器；用于定时器/计数器时，作为一个预分频器，通常统称作预分频器。在片内只有一个物理的分频器，只能用于WDT/TIMER0两者之一，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: CLRF TIMER0,BSF TIMER0,1等)都会对预分频器清零。

由TIMER0还是WDT使用预分频器，完全由软件控制。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。

```
CLRF      TMR0          ;TMR0 清零
CLRWDT                  ;WDT 清零
MOVLW     B’00xx1xxx’   ;设置新的预分频器
OPTION                  ;预分频寄存器赋值
```

将预分频器从分配给 WDT 切换为分配给 TMR0 模块，应该执行以下指令。

```
CLRWDT                  ;WDT 清零
MOVLW     B’00xx0xxx’   ;设置新的预分频器
OPTION                  ;预分频寄存器赋值
```

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

2.7 程序计数器(PC)

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

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

程序计数器(PC)是 11BIT 宽度，低 8 位通过 PCL(02H)寄存器用户可以访问，高 3 位用户不能访问。可容纳 2Kx14 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);
RETLW k、RETURN、RETFIE 时	PC=堆栈出来的值;
操作 PCL 时	PC[10:8]不变，PC[7:0]=用户指定的值;
GOTO 时	PC=程序指定的值;
其它指令	PC=PC+1;

表 2.6 特殊情况 PC 值

2.8 看门狗计数器(WDT)

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

2.8.1 WDT 周期

WDT 与 TIMER0 共用 8 位预分频器。WDT 有一个基本的溢出周期 18ms(无预分频器)，假如你需要更长时间的 WDT 周期，可以把预分频器 OPTION_REG 分配给 WDT。WDT 的溢出周期将受到环境温度，电源电压等参数影响。

“CLRWDT”和“SLEEP”指令将清除 WDT 定时器以及预分频器里的计数值(当预分频器分配给 WDT 时)。WDT 一般用来防止系统失控，或者可以说是用来防止单片机程序失控。在正常情况下，WDT 应该在其溢出前被“CLRWDT”指令清零，以防止产生复位。如果程序由于某种干扰而失控，那么不能在 WDT 溢出前执行“CLRWDT”指令，就会使 WDT 溢出而产生复位。使系统重启而不至

于失去控制。若是 WDT 溢出产生的复位，则状态寄存器(STATUS)的“TO”位会被清零，用户可根据此位来判断复位是否是 WDT 溢出所造成的。

注：1.若使用 WDT 功能，一定要在程序的某些地方放置“CLRWDT”指令，以保证在 WDT 溢出前能被清零。否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 看门狗定时器控制寄存器 WDTCON

看门狗定时器控制寄存器WDTCON(8EH)

8EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WDTCON	----	----	----	WDTPS[3:0]				SWDTEN
R/W	----	----	----	R/W	R/W	R/W	R/W	R/W
复位值	----	----	----	0	1	0	0	0

表 2.7 看门狗定时器控制寄存器 WDTCON

Bit7-5 **未用， 读为0**

Bit4-1 **WDTPS[3:0] :** 看门狗定时器周期选择位

0000=1:32
0001=1:64
0010=1:128
0011=1:256
0100=1:512（默认值）
0101=1:1024
0110=1:2048
0111=1:4096
1000=1:8192
1001=1:16384
1010=1:32768
1011=1:65536
11xx=保留

Bit0 **SWDTEN:** 软件使能或禁止看门狗定时器位

1=使能WDT
0=禁止WDT（复位值）

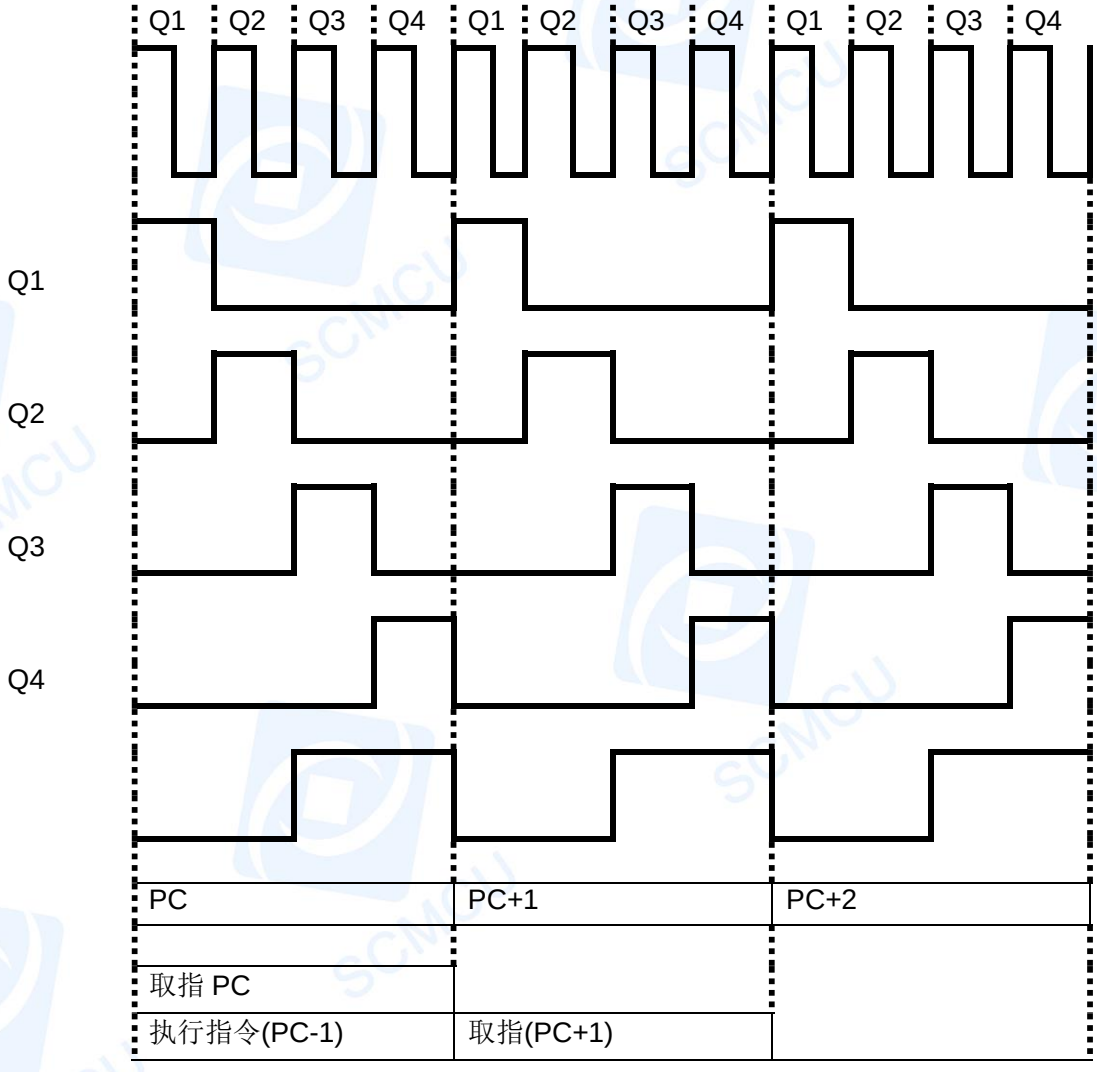
注 1: 如果 CONFIG 中 WDT 配置位=1，则 WDT 始终被使能，而与 SWDTEN 控制位的状态无关。如果 CONFIG 中 WDT 配置位=0，则可以使用 SWDTEN 控制位使能或禁止 WDT。

3. 系统时钟

3.1 概述

时钟信号从 OSCIN 引脚输入后(或者由内部振荡产生)，在片内产生 4 个非重迭正交时钟信号，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 GOTO)那么预取的指令操作码就无效，就需要两个指令周期来完成该条指令，这就是对 PC 操作指令都占用两个时钟周期的原因。



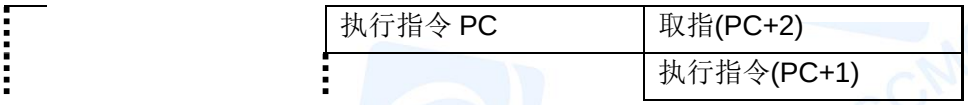


图 3.1　时钟与指令周期时序图

下面列出振荡频率与指令速度的关系

频率	双指令周期	单指令周期
1MHz	8uS	4uS
2MHz	4uS	2uS
4MHz	2uS	1uS
8MHz	1uS	500nS

表 3.1　震荡频率与指令速度关系表

3.2　系统振荡器

SC8F301X 只有 2 种振荡方式，内部 RC 振荡和外部 XT 振荡。

3.2.1　内部 RC 振荡

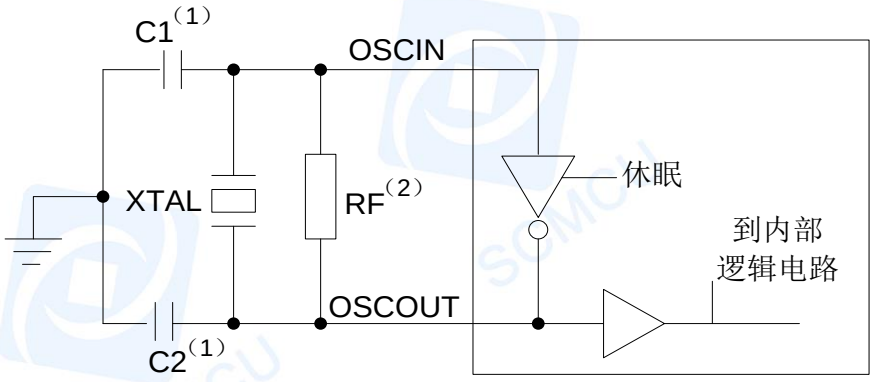
芯片默认的振荡方式为内部 RC 振荡，其振荡频率为 8MHz 可通过 OSCCON 寄存器设置芯片工作频率。振荡频率在出厂时校正，其误差在±3%以内。

当选择内部 RC 作为芯片的振荡器时，芯片的 OSCIN 和 OSCOUT 可以作为普通的 I/O 口。

3.2.2　外部 XT 振荡

在烧录时将 CONFIG 选项中的 OSC 选择成 XT，芯片工作在外部 XT 振荡模式下，此时内部 RC 振荡停止工作，OSCIN 和 OSCOUT 作为振荡口。

典型的 XT 振荡方式如下图所示:



建议参数:

类型	频率	建议值 RF	建议值 C1～C2
XT	455kHz	1MΩ	100PF～470PF
XT	2MHz	1MΩ	10PF～47PF
XT	4MHz	1MΩ	10PF～47PF
XT	8MHz	1MΩ	10PF～47PF

3.3 起振时间

起振时间(OSC TIME)是指从芯片复位到芯片振荡稳定这段时间，可由内部烧写 CONFIG 选项设置为 18ms、9ms、2ms、560us；具体设置参数请参照 1.5 烧写选项设定章节。

注：无论芯片是电源上电复位，还是其它原因引起的复位，都会存在这个起振时间。

3.4 振荡器控制寄存器

振荡器控制（OSCCON）寄存器控制系统时钟和频率选择，振荡器调节寄存器 OSCTUNE 可以用软件调节内部振荡频率。

振荡器控制寄存器OSCCON(8FH)

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

表 3.2 振荡器控制寄存器 OSCCON

Bit7 **未用，读为 0**

Bit6-4 **IRCF<2:0>:** 内部振荡器频率选择位

111=8MHz
110=4MHz（默认）
101=2MHz
100=1MHz
011=500kHz
010=250kHz
001=125kHz
000=31kHz（LFINTOSC）

Bit3-Bit1 未用

Bit0 **SCS:** 系统时钟选择位
1=内部振荡器用作系统时钟
0=时钟源由 CONFIG 定义

振荡器调节寄存器OSCTUNE(90H)

90H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCTUNE	----	----	----	TUN4	TUN3	TUN2	TUN1	TUN0
R/W	----	----	----	R/W	R/W	R/W	R/W	R/W
复位值	----	----	----	0	0	0	0	0

表 3.3 振荡器调节寄存器 OSCTUNE

Bit7-5 未用

Bit4-0 **TUN<4:0>:** 频率调节位
01111=最高频率
01110=
•
•
00001=
00000=振荡器模块以厂家校准后的频率运行
11111=
•
10000=最低频率

4. 复位

SC8F301X 可用如下 4 种复位方式:

- 上电复位
- 低电压复位(LVR 使能)
- 正常工作下的看门狗溢出复位
- 复位口低电平(LVR 禁止)

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 PD 和 TO 标志位能够给出系统复位状态的信息，(详见 STATUS 的说明)，用户可根据 PD 和 TO 的状态，控制程序运行路径。

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

4.1 上电复位

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

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

4.2 掉电复位

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

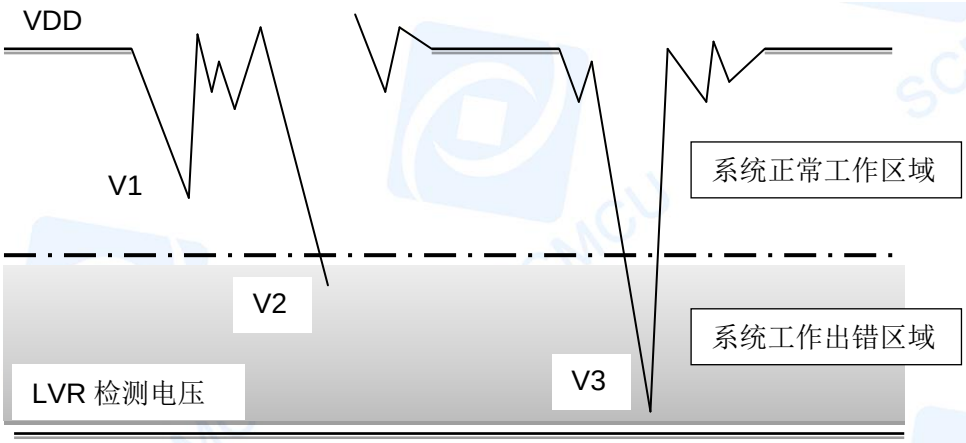


图4.1 掉电复位示意图

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

以下情况系统可能进入死区：

DC 运用中：

DC 运用中一般都采用电池供电，当电池电压过低或单片机驱动负载时，系统电压可能跌落并进入死区。这时，电源不会进一步下降到 LVD 检测电压，因此系统维持在死区。

AC 运用中：

系统采用 AC 供电时，DC 电压值受 AC 电源中的噪声影响。当外部负载过高，如驱动马达时，负载动作产生的干扰也影响到 DC 电源。VDD 若由于受到干扰而跌落至最低工作电压以下时，则系统将有可能进入不稳定工作状态。

在 AC 运用中，系统上、下电时间都较长。其中，上电时序保护使得系统正常上电，但下电过程却和 DC 运用中情形类似，AC 电源关断后，VDD 电压在缓慢下降的过程中易进入死区。

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

4.2.1 掉电复位的改进办法

如何改进系统掉电复位性能，以下给出几点建议：

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

看门狗定时器

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

降低系统的工作速度

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

增大电压下降斜率

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

4.3 看门狗复位

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

看门狗复位的时序如下：

- 看门狗定时器状态：系统检测看门狗定时器是否溢出，若溢出，则系统复位；
- 初始化：所有的系统寄存器被置为默认状态；
- 振荡器开始工作：振荡器开始提供系统时钟；
- 程序：复位结束，程序开始运行。
- 关于看门狗定时器的应用问题请参看2.8 WDT应用章节。

4.4 复位口电平复位

当 LVR 功能被屏蔽时，REST 口低电平会使 MCU 进入复位态，具体工作时序如下：

- REST口为低电平；
- REST口电平从低变为高，若一直为低，则芯片一直处于复位态。
- 初始化：所有的系统专用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。

4.5 基本外部复位电路

4.5.1 RC 复位电路

下图为一个由电阻 R1 和电容 C1 组成的基本 RC 复位电路，它在系统上电的过程中能够为复位引脚提供一个缓慢上升的复位信号。这个复位信号的上升速度低于 VDD 的上电速度，为系统提供合理的复位时序，当复位引脚检测到高电平时，系统复位结束，进入正常工作状态，当系统选择 LVR 禁止模式时，用户可选用此电路以提高复位的可靠性。

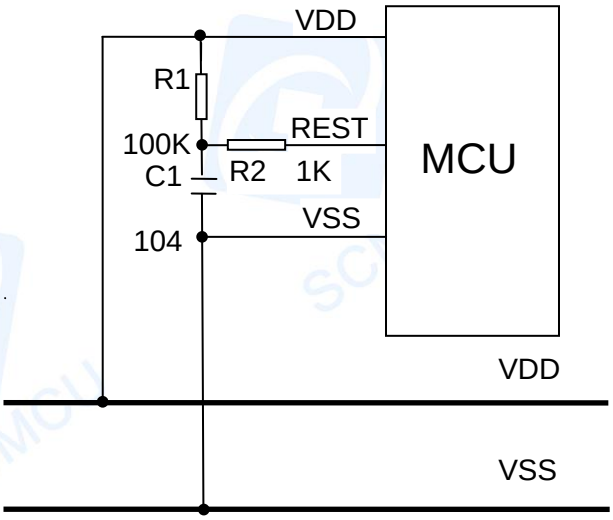


图 4.2　RC 复位电路

4.5.2　二极管及 RC 复位电路

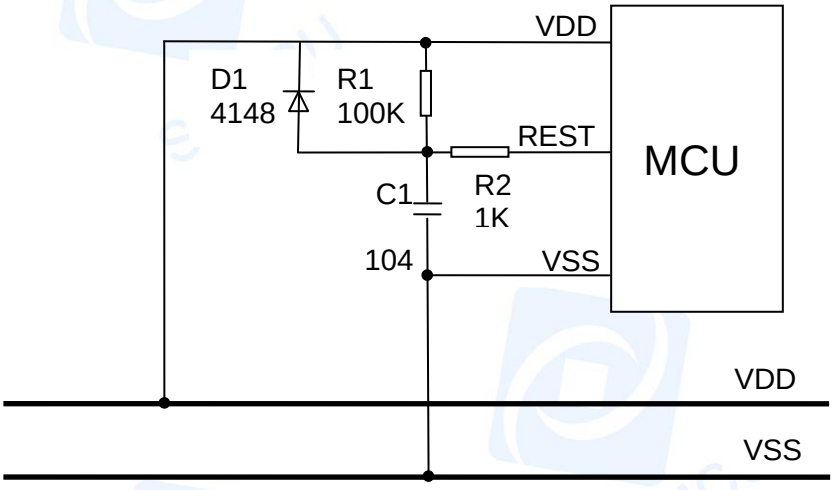


图 4.3　二极管及 RC 复位电路

上图中，R1 和 C1 同样是为复位引脚提供输入信号。对于电源异常情况，二极管正向导通使 C1 快速放电并与 VDD 保持一致，避免复位引脚持续高电平、系统无法正常复位。

4.5.3　三极管复位电路

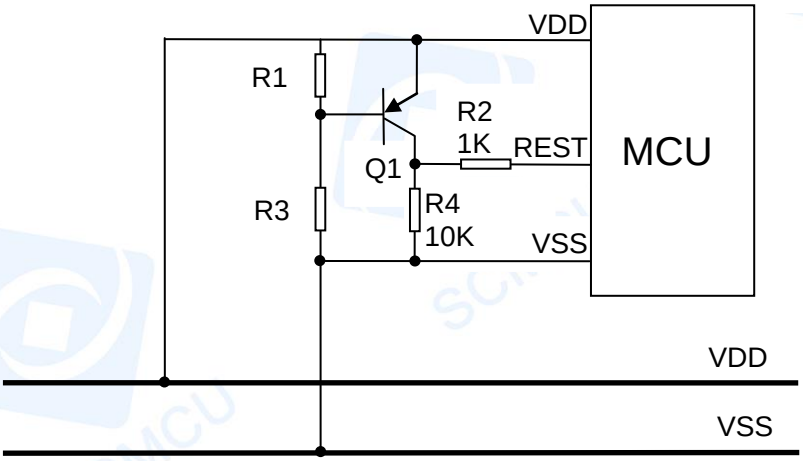


图 4.4　三极管复位电路

电压偏置复位电路是一种简单的 LVR 电路，基本上可以解决掉电复位问题。这种复位电路中，R1 和 R2 构成分压电路，当 VDD 高于和等于分压值“0.7V x (R1 + R3)/ R1”时，三极管集电极 C 输出高电平，单片机正常工作；VDD 低于“0.7V x (R1 + R3) / R1”时，集电极 C 输出低电平，单片机复位。对于不同应用需求，选择适当的分压电阻。单片机复位引脚上电压的变化与 VDD 电压变化之间的差值为 0.7V。如果 VDD 跌落并低于复位引脚复位检测值，那么系统将被复位。如果希望提升电路复位电平，可将分压电阻设置为 R3>R1，并选择 VDD 与集电极之间的结电压高于 0.7V。分压电阻 R1 和 R3 在电路中要耗电，此处的功耗必须计入整个系统的功耗中。

4.5.4　稳压二极管复位电路

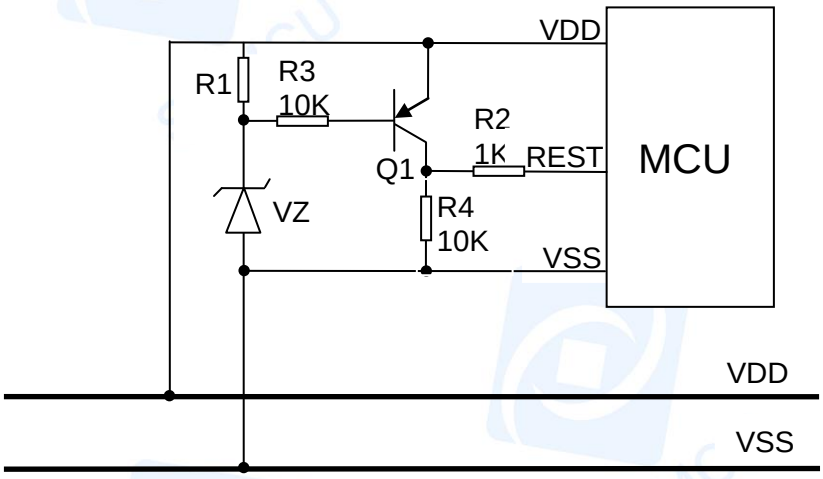


图 4.5　稳压管复位电路

稳压二极管复位电路是一种简单的 LVR 电路，基本上可以解决掉电复位问题。如上图电路中，利用稳压管的击穿电压作为电路复位检测值，当 VDD 高于“Vz + 0.7V”时，三极管集电极输出高电平，单片机正常工作；当 VDD 低于“Vz + 0.7V”时，三极管集电极输出低电平，单片机复位。稳压管规格不同则电路复位检测值不同，根据电路的要求选择合适的二极管。

注：上述外部复位电路中，R2电阻不能取消，以提高REST口的抗EMC及ESD能力。

5. 系统工作模式

SC8F301X 系列 MCU 存在两种工作模式，一种是正常工作模式，一种是休眠工作模式。在正常工作模式下，各个功能模块均处于工作状态，在休眠状态下，系统时钟停止，芯片保持原来的状态不变，此时 WDT 的功能若没有被烧写 CONFIG 选项禁止，则 WDT 定时器一直工作。

5.1 进入休眠模式

执行 SLEEP 指令可进入掉电模式。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 SLEEP 指令之前的状态（驱动为高电平、低电平或高阻态）。

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

5.2 从休眠状态唤醒

可以通过下列任一事件将器件从休眠状态唤醒：

1、看门狗定时器唤醒（WDT 强制使能）

2、PORTA 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 SLEEP 指令时被清零。TO 位在发生 WDT 唤醒时被清零。

当执行 SLEEP 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 SLEEP 指令之后的指令。如果 GIE 位被置 1（允许），器件执行 SLEEP 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 SLEEP 指令之后的指令，用户应该在 SLEEP 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 SLEEP 指令之前产生了中断，那么 SLEEP 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 SLEEP 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。SLEEP 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 SLEEP 指令之前检查到标志位为 0，它也可能在 SLEEP 指令执行完毕之前被置 1。要确定是否执行了 SLEEP 指令，可以测试 PD 位。如果 PD 位置 1，则说明 SLEEP 指令被作为一条 NOP 指令执行了。在执行 SLEEP 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 SLEEP 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 AD 模块及比较器模块；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 SLEEP 的处理程序

```
SLEEP_MODE:
        CLRF    INTCON      ;关断中断使能；
        MOVLW   B'00000000'
        MOVWF   TRISA
        MOVWF   TRISB       ;所有 I/O 设置为输出口；
        MOVWF   TRISC
        …                   关闭其它功能；
        MOVLW   0A5H
        MOVWF   SP_FLAG     ;置休眠状态记忆寄存器(用户自定义)；
```

```
CLRWDT                ;清零 WDT；
SLEEP                 ;执行 SLEEP 指令。
```

5.5 休眠模式唤醒时间

当 MCU 从休眠态被唤醒时，需要等待一个振荡稳定时间(RESET TIME)。

6. I/O 端口

SC8F301X 有三个 I/O 端口: PORTA、PORTB 和 PORTC(最多 18 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	20 脚	16 脚	8 脚	管脚描述	输入输出
PORT A	0	19	15	7	施密特触发输入，推挽式输出，AN0,KEY0,外部中断,COM0	I/O
	1	18	14	/	施密特触发输入，推挽式输出，AN1,KEY1,COM1	I/O
	2	17	13	/	施密特触发输入，推挽式输出，AN2,KEY2, COM2	I/O

	3	16	12	/	施密特触发输入，推挽式输出，AN3,KEY3,COM3	I/O
	4	15	11	/	施密特触发输入，推挽式输出，AN4,KEY4,T1OSCI	I/O
	5	14	7	/	施密特触发输入，推挽式输出，AN5,KEY5,T1OSCO,T1G	I/O
	6	13	8	/	施密特触发输入，推挽式输出，AN6,KEY6,CCP1	I/O
	7	12	10	6	施密特触发输入，推挽式输出，AN7,KEY7,CCP2	I/O
PORT B	0	5	/	/	施密特触发输入，推挽式输出，AN14,KEY13	I/O
	1	6	/	/	施密特触发输入，推挽式输出，AN13,KEY12	I/O
	2	7	5	/	施密特触发输入，推挽式输出，AN12,KEY11	I/O
	3	8	/	/	施密特触发输入，推挽式输出，AN11, KEY10	I/O
	4	9	/	/	施密特触发输入，推挽式输出，AN10, KEY9	I/O
	5	10	6	5	施密特触发输入，推挽式输出，AN9,CAP	I/O
	6	11	9	/	施密特触发输入，推挽式输出，AN8,KEY8	I/O
PORTC	0	2	2	2	施密特触发输入，推挽式输出，KEY15,OSCI,ICSPDAT	I/O
	1	3	3	3	施密特触发输入，推挽式输出，KEY14,OSCO,ICSCLK	I/O
	2	4	4	4	施密特触发输入，开漏输出，VPP,RESET	I/O

表 6.1　端口配置总概

I/O 口结构图

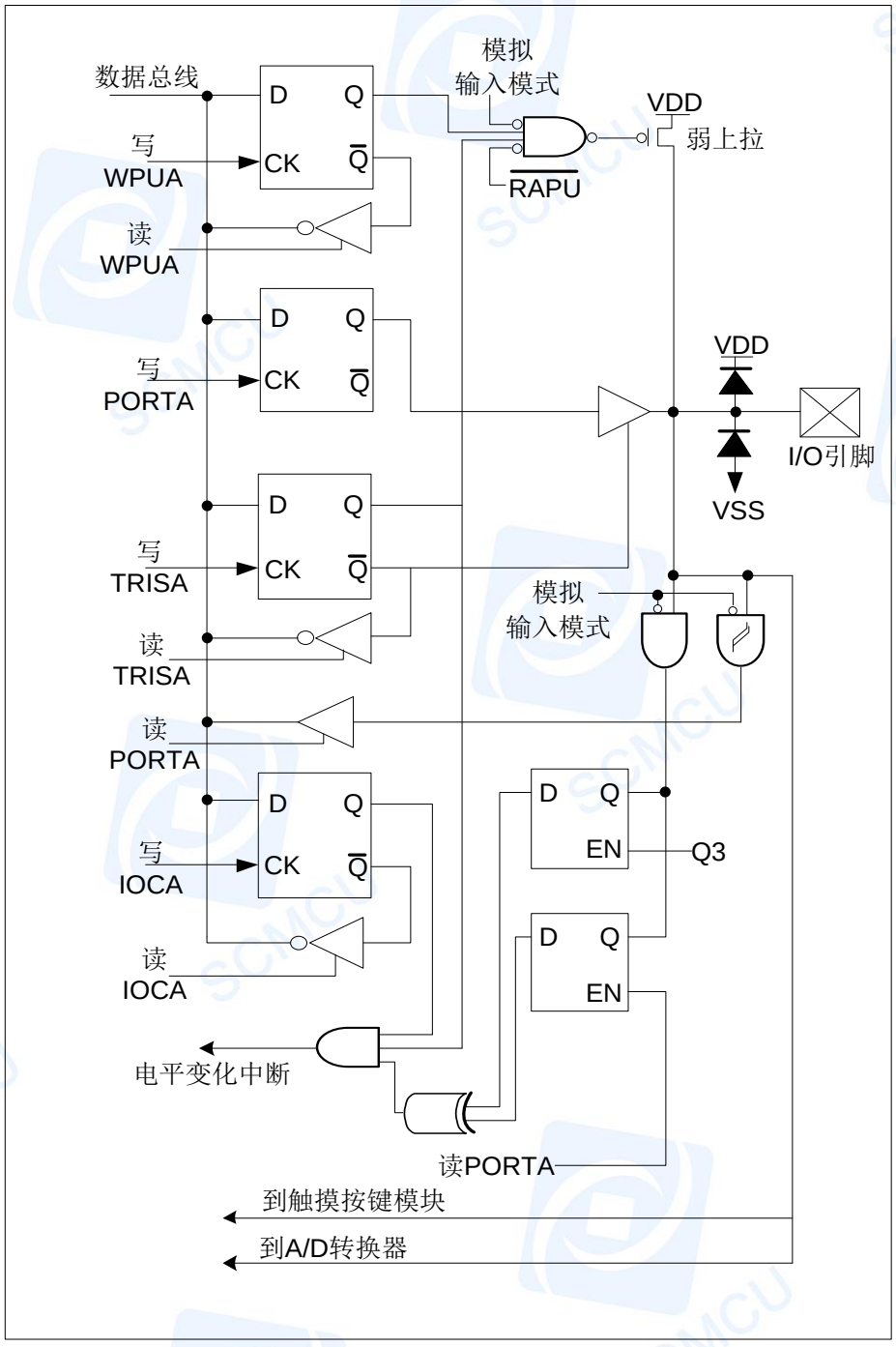


图 6.1　I/O 结构图 1

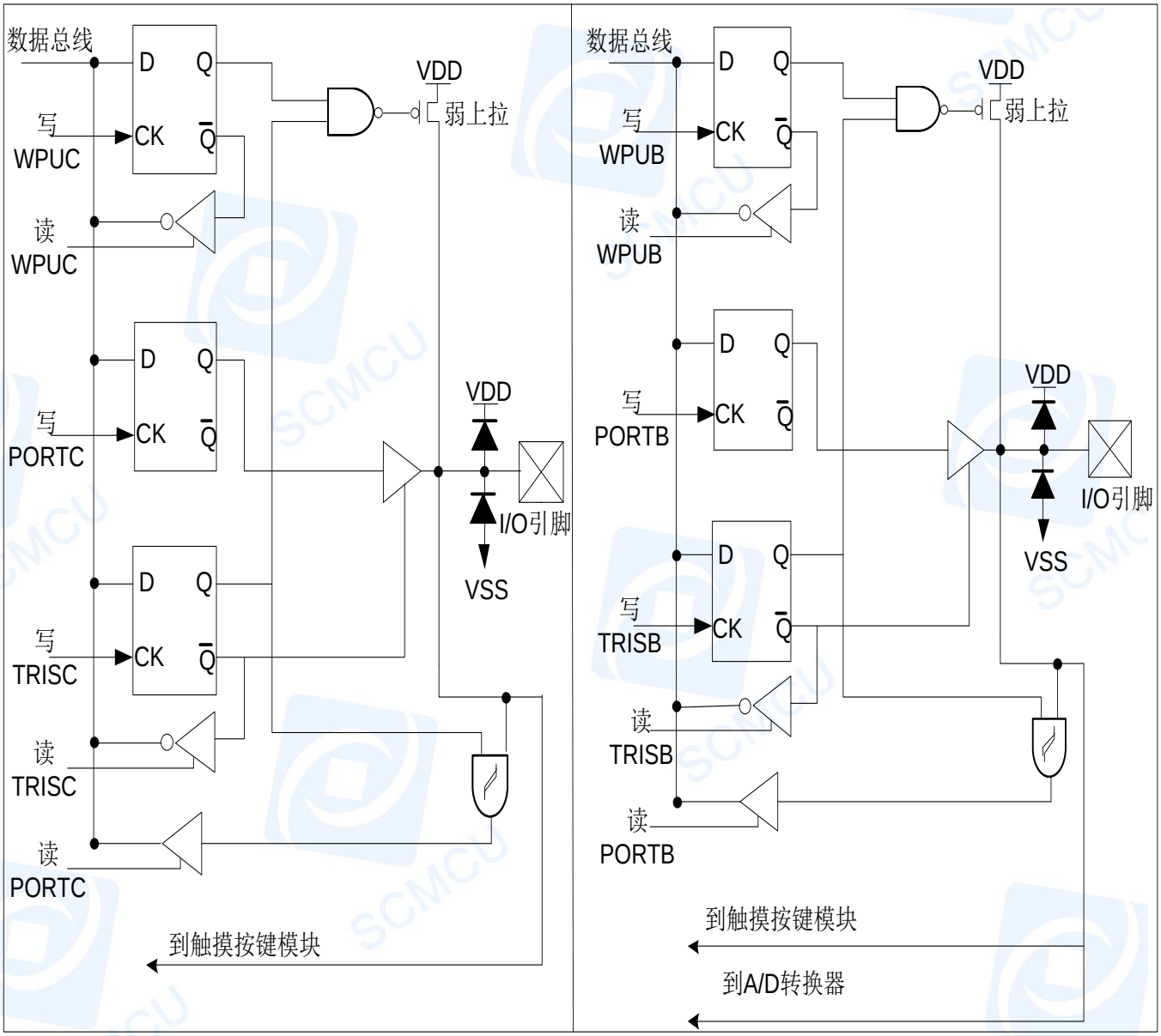

数据总线
D
Q
CK
Q
写
WPUC
读
WPUC
VDD
弱上拉
写
PORTC
I/O引脚
VSS
写
TRISC
读
TRISC
读
PORTC
到触摸按键模块
数据总线
写
WPUB
读
WPUB
写
PORTB
写
TRISB
读
TRISB
读
PORTB
到触摸按键模块
到A/D转换器

图 6.2　I/O 结构图 2

6.1 PORTA

6.1.1 PORTA 数据及方向控制

PORTA 是 8 位宽的双向端口。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。

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

PORTA口数据寄存器PORTA (05H)

05H	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

表 6.2 PORTA 口数据寄存器

Bit7-0 PORTA<7:0>: PORTA I/O 引脚位

1=端口引脚电平>VIH

0=端口引脚电平<VIL

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

表 6.3 PORTA 方向寄存器 TRISA

Bit7-0 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

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

6.1.2 PORTA 上拉电阻

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

PORTA上拉电阻寄存器WPUA(95H)

95H	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

表 6.4 PORTA 上拉电阻寄存器 WPUA

BIT7-BIT6 **WPUA<7:0>:** 弱上拉寄存器位

1=使能上拉

0=禁止上拉

注 1: 要单独使能任一个上拉，OPTION_REG 寄存器的全局 RAPU 位必须清零。

6.1.3 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 INTCON 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方式清除中断：

a)对 PORTA 进行读或写操作。这将结束引脚电平的不匹配状态。

b)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。锁存器将保持最后一次读取的值不受欠压复位的影响。在复位之后，如果不匹配仍然存在，RAIF 标志位将继续置 1。

PORTA电平变化中断寄存器IOCA(98H)

98H	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

表 6.5 PORTA 电平变化中断寄存器 IOCA

Bit7-0 **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

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。

6.2 PORTB

6.2.1 PORTB 数据及方向控制

PORTB 是一个 7 位宽的双向端口。对应的数据方向寄存器为 PORTB。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。

与 PORTB 口相关寄存器有 PORTB、TRISB、WPUB 等。

PORTB数据寄存器PORTB(06H)

06H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTB	----	RB6	RB5	RB4	RB3	RB2	RB1	RB0
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	x	x	x	x	x	x	x	x

表 6.6 PORTB 数据寄存器 PORTB

Bit7 未用

Bit6-0 **PORTB<6:0>:** PORTB I/O 引脚位

1=端口引脚电平>VIH

0=端口引脚电平<VIL

PORTB方向寄存器TRISB(86H)

86H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISB	----	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

表 6.7 PORTB 方向寄存器 TRISB

Bit7 未用

Bit6-0 **PORTB<6:0>:** PORTB 三态控制位

1=PORTB 引脚被配置为输入（三态）

0=PORTB 引脚被配置为输出

例：PORTB 口处理程序

```
CLRF    PORTB          ;清数据寄存器
MOVLW   B'00110000'    ;设置 PORTB<5:4>为输入口，其余为输出口
MOVWF   TRISB
```

6.2.2 PORTB 上拉电阻

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

PORTB上拉电阻寄存器WPUB(96H)

96H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUB	----	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
R/W	----	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

表 6.8 PORTB 上拉电阻寄存器 WPUB

Bit7 未用

Bit6-0 **WPUB<6:0>:** 弱上拉寄存器位

1=使能上拉

0=禁止上拉

6.3 PORTC

6.3.1 PORTC 数据及方向

PORTC 是一个 3 位宽的双向端口。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，RC2 口只能用于开漏输出口并 CONFIG 需使能。

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

PORTC数据寄存器PORTC(07H)

07H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PORTC	----	----	----	----	----	RC2	RC1	RC0
R/W	----	----	----	----	----	R/W	R/W	R/W
复位值	x	x	x	x	x	x	x	x

表 6.9 PORTC 数据寄存器 PORTC

Bit7-3 未用

Bit2-0 **PORTC<2:0>:** PORTC I/O 引脚位

1=端口引脚电平>VIH

0=端口引脚电平<VIL

PORTC方向寄存器TRISC(87H)

87H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TRISC	----	----	----	----	----	TRISC2	TRISC1	TRISC0
R/W	----	----	----	----	----	R/W	R/W	R/W
复位值	1	1	1	1	1	1	1	1

表 6.10　PORTC 方向寄存器 TRISC

Bit7-3　未用

Bit2-0　**TRISC<2:0>:** 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

注：在使用 RES_OUT 功能时，程序中必须上电延时超过 100ms 的时间，才能将 RES 口置成开漏输出态，否则芯片可能无法重复烧录。

例：PORTC 口处理程序

```
CLRF    PORTC           ;清数据寄存器
MOVLW   B'00000010'     ;设置 RC1 为输入口，其余为输出口
MOVWF   TRISC
```

6.3.2　PORTC 上拉电阻

每个 PORTC 引脚都有可单独配置的内部弱上拉。控制位 WPUC<1:0>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(97H)

97H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
WPUC	----	----	----	----	----	----	WPUC1	WPUC0
R/W	----	----	----	----	----	----	R/W	R/W
复位值	0	0	0	0	0	0	0	0

表 6.11　PORTC 上拉电阻寄存器 WPUC

Bit1-0　**WPUC<1:0>:** 弱上拉寄存器位

1=使能上拉

0=禁止上拉

注 1: 如果引脚被配置为输出，将自动禁止弱上拉。

6.4　I/O 口的使用

6.4.1　写 I/O 口

SC8F301X 系列芯片的 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

```
MOVWF   PORTA           ;W 值赋给 PORTA 口
BCF     PORTB,1         ;RB1 口置零
CLRF    PORTC           ; PORTC 口清零
SETF    PORTA           ; PORTA 所有输出口置 1
```

6.4.2 读 I/O 口

★ 例：读 I/O 口程序

```
MOVF    PORTA,0     ;PORTA 的值赋给 W
BTFSS   PORTA,1     ;判断 RA1 口是否为 1，为 1 跳过下一条语句
BTFSC   PORTA,1     ;判断 RA1 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。

6.5 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：

- 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。
- 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。
- 当 I/O 口为输入口时，其输入电平应在“VDD+0.7V”与”VSS-0.7V”之间。若输入口电压不在此范围内可采用如下图所示方法。

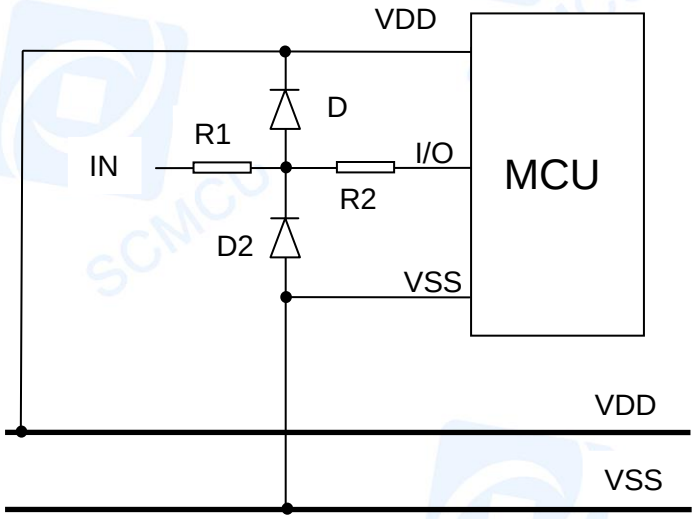


图 6.3 输入电压不在规定范围内采用电路

- 若在 I/O 口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

SC8F301X 具有以下多种中断源:

- TIMER0 溢出中断
- TIMER1 溢出中断
- TIMER2 匹配中断
- INT 中断
- PORTA 电平变化中断
- A/D 中断
- CCP1/CCP2 中断

中断控制寄存器（INTCON）和外设中断请求寄存器（PIR1）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。

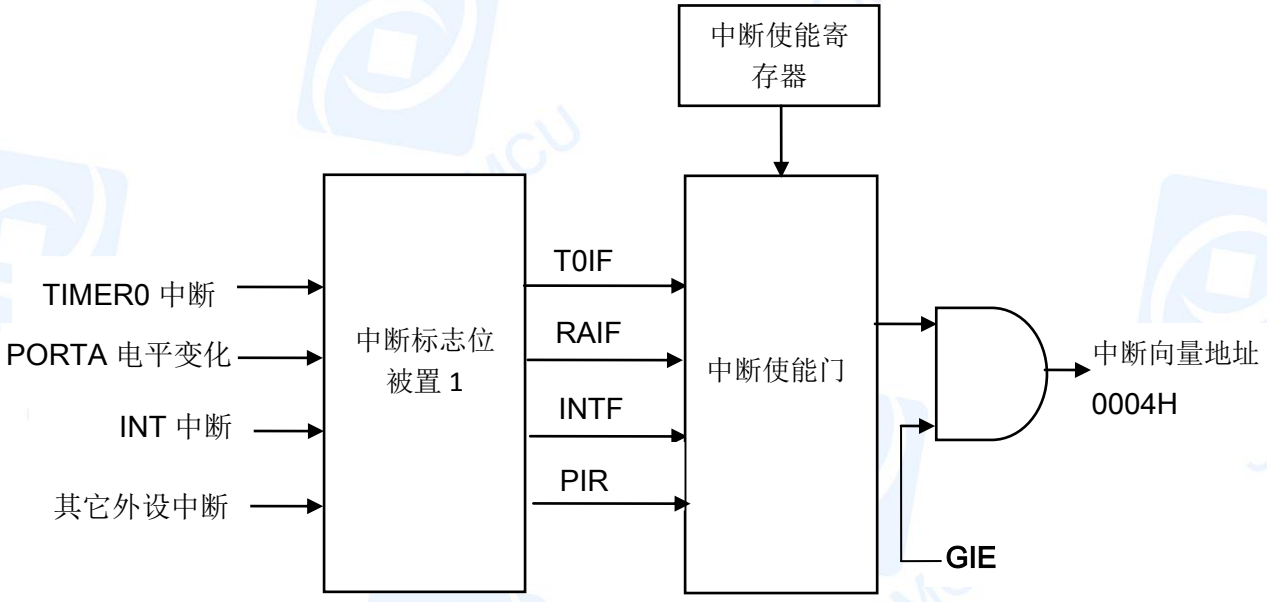


图7.1 中断系统

7.2 中断控制寄存器

7.2.1 中断控制寄存器

中断控制寄存器 INTCON 是可读写的寄存器，包含 TMR0 寄存器溢出、PORTA 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	RAIE	T0IF	INTF	RAIF
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

表 7.1 中断控制寄存器 INTCON

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 **RAIE:** PORTA电平变化中断允许位(1)
1=允许PORTA电平变化中断
0=禁止PORTA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 **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

注: (1)、IOCA 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 PIE1，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(8CH)

8CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIE1	----	ADIE	----	----	CCP2IE	CCP1IE	TMR2IE	TMR1IE
R/W	----	R/W	----	----	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

表 7.2 外设中断允许寄存器 PIE1

Bit7 **未用，读为0**

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

Bit5-4 **未用，读为0**

Bit3 **CCP2IE:** CCP2中断允许位
1=允许CCP2中断
0=禁止CCP2中断

Bit2 **CCP1IE:** CCP1中断允许位
1=允许CCP1中断
0=禁止CCP1中断

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

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

7.2.3 外设中断请求寄存器

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

外设中断请求寄存器PIR1(0CH)

0CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PIR1	----	ADIF	----	----	CCP2IF	CCP1IF	TMR2IF	TMR1IF
R/W	----	R/W	----	----	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

表 7.3 外设中断请求寄存器 PIR1

Bit7 **未用，读为0**

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

Bit5-4 **未用，读为0**

Bit3 **CCP2IF:** CCP2中断标志位

捕捉模式**:**

1=发生了TMR1寄存器的捕捉（必须由软件清零）

0=没有发生TMR1寄存器的捕捉

PWM模式**:**

在此模式下未用

Bit2 **CCP1IF:** CCP1中断标志位

捕捉模式**:**

1=发生了TMR1寄存器的捕捉（必须由软件清零）

0=没有发生TMR1寄存器的捕捉

PWM模式**:**

在此模式下未用

Bit1 **TMR2IF:** TIMER2与PR2匹配中断标志位

1=发生了TIMER2与PR2匹配（必须由软件清零）

0=TIMER2与PR2不匹配

Bit0 **TMR1IF:** TIMER1溢出中断标志位

1=TMR1寄存器溢出（必须由软件清零）

0=TMR1寄存器未溢出

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 W、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 W 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
        GOTO    START               ;用户程序起始地址
        ORG     0004H
        GOTO    INT_SERVICE         ;中断服务程序
        ORG     0008H
START:
        ...
        ...
INT_SERVICE
PUSH:                               ;中断服务程序入口，保存 W 及 STATUS
        MOVWF   W_BAK               ;保存 W 的值,(W_BAK 需自定义)
        SWAPF   STATUS,0
        MOVWF   STATUS_BAK          ;保存 STATUS,（STATUS_BAK 需自定义）
        ...
        ...
POP:                                ;中断服务程序出口，还原 ACC 及 STATUS
        SWAPF   STATUS_BAK,0
        MOVWF   STATUS              ;还原 STATUS 的值
        SWAPF   W_BAK,1             ;还原 W 的值
        SWAPF   W_BAK,0
        RETFIE
```

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“RETFIE”指令后，才能响应下一个中断。

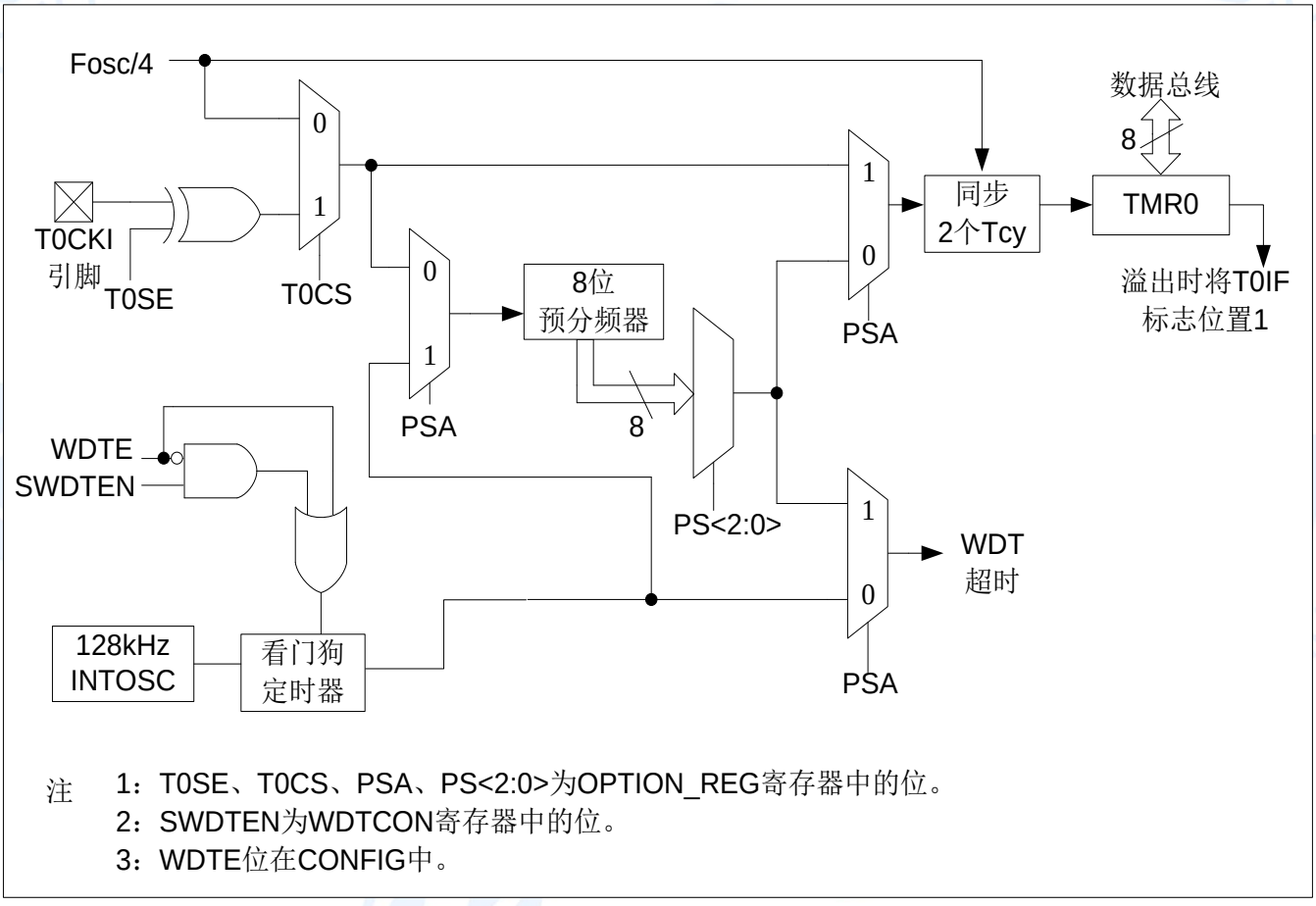
多个中断同时发生时，MCU 没有预置的中断优先级。首先，必须预先设定好各中断的优先权；其次，利用中断使能位和中断控制位，控制系统是否响应该中断。在程序中，必须对中断控制位和中断请求标志进行检测。

8. 定时计数器 TIMER0

8.1 定时计数器 TIMER0 概述

TMR0 由如下功能组成:

- 8 位定时器 / 计数器
- 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可选择
- 溢出中断



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

图 8.1 TMR0/WDT 结构图

8.2 TIMER0 的工作原理

TIMER0 模块既可用作 8 位定时器也可用作 8 位计数器。

8.2.1 8 位定时器模式

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

8.2.2 8 位计数器模式

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

8.2.3 软件可编程预分频器

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

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

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

8.2.4 在 TIMER0 和 WDT 模块间切换预分频器

将预分频器分配给 TIMER0 或 WDT 后，在切换预分频比时可能会产生无意的器件复位。要将预分频器从分配给 TIMER0 改为分配给 WDT 模块时，必须执行如例 8-1 所示的指令序列。

例 8-1 更改预分频器（TMR0-WDT）

```
CLRWDT
CLRF        TMR0
BSF         OPTION_REG,PSA      ;选择 WDT
CLRWDT
MOVLW       B’11111000’
ADDWF       OPTION_REG,0        ;低 3 位置 0
IORLW       B’00000101’         ;低 3 位置成 101，其余位不变
OPTION
```

要将预分频器从分配给 WDT 改为分配给 TIMER0 模块，必须执行以下指令序列（见例 8-2）。

例 8-2 更改预分频器（TMR0-WDT）

```
CLRWDT
MOVLW       B'11110000'
ADDWF       OPTION_REG,0        ;低 4 位置 0
IORLW       B'00000101'         ;低 4 位置成 0101，其余位不变
OPTION
```

8.2.5 TIMER0 中断

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

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

8.3 与 TIMER0 相关寄存器

有两个寄存器与 TMR0 相关，定时器 / 计数器(TMR0)，可编程控制寄存器(OPTION_REG)。TMR0 为一个 8 位可读写的定时/计数器，OPTION_REG 为一个 8 位只写寄存器，用户可改变 OPTION_REG 的值，来改变 TMR0 的工作模式等。请参看 2.6 关于预分频寄存器的应用。

8位定时器/计数器TMR0(01H)

01H	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

表 8.1　8 位定时器/计数器 TMR0

OPTION_REG寄存器（81H）

81H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OPTION_REG	RAPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
读写	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

表 8.2　OPTION_REG 寄存器

BIT7　**RAPU:** PORTA 上拉使能位
1=禁止 PORTA 上拉
0=由端口的各个锁存值使能 PORTA 上拉

BIT6　**INTEDG:** 触发中断的边沿选择位
1=INT 引脚上升沿触发中断
0=INT 引脚下降沿触发中断

BIT5　**T0CS:** TIMER0 时钟源选择位
1=T0CKI 引脚上的跳变沿

0=内部指令周期时钟（FOSC/4）

BIT4 **T0SE:** TIMER0 时钟源边沿选择位

1=在 T0CKI 引脚信号从高电平跳变到低电平时递增

0=在 T0CKI 引脚信号从低电平跳变到高电平时递增

BIT3 **PSA**：预分频器分配位

1=预分频器分配给 WDT

0=预分频器分配给 TIMER0 模块

BIT2～BIT0 **PS2～PS0**：预分配参数配置位：

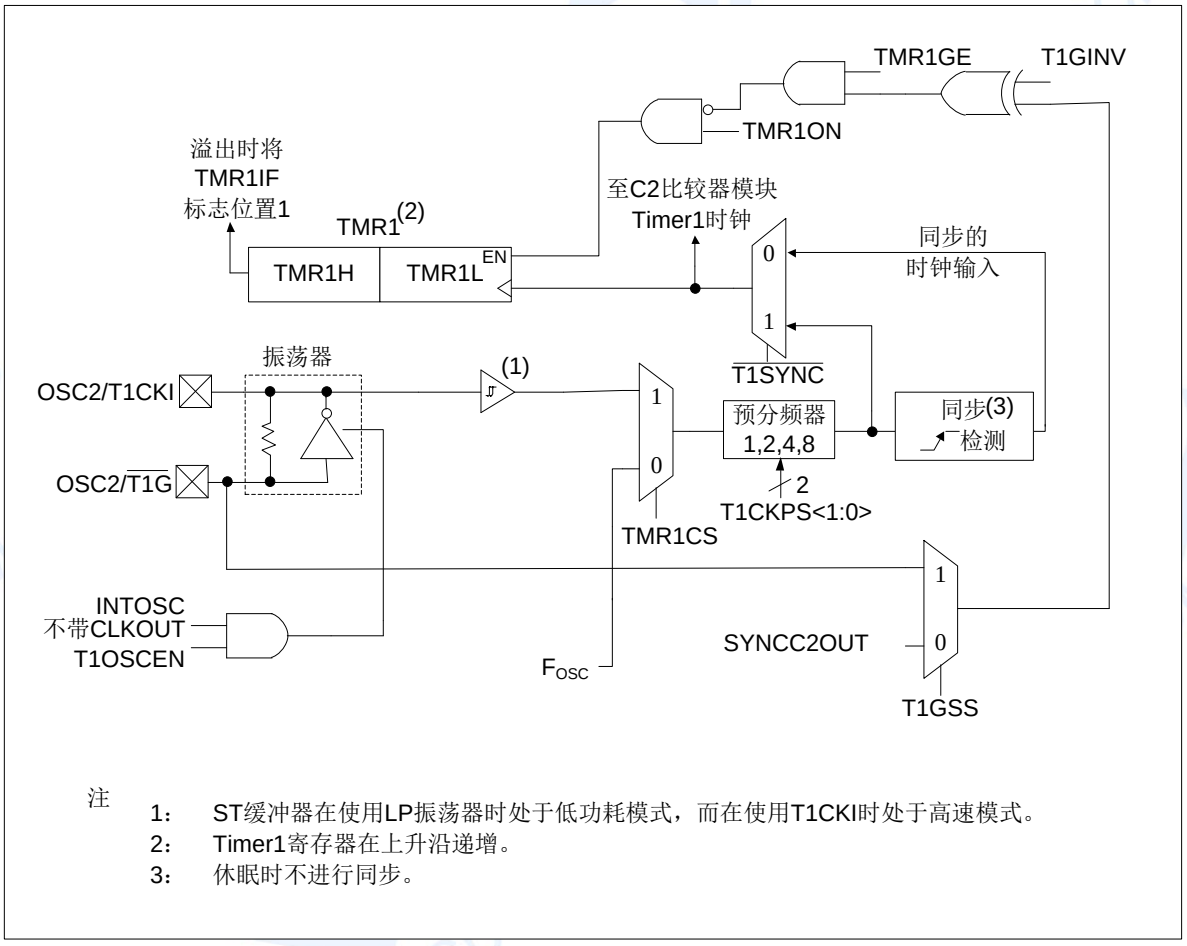
PS2	PS1	PS0	TIMER0 分频比	WDT 分频比
0	0	0	1:2	1:1
0	0	1	1:4	1:2
0	1	0	1:8	1:4
0	1	1	1:16	1:8
1	0	0	1:32	1:16
1	0	1	1:64	1:32
1	1	0	1:128	1:64
1	1	1	1:256	1:128

9. 定时计数器 TIMER1

9.1 TMR1 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）
- 可编程内部或外部时钟源
- 3 位预分频器
- 可选 LP 振荡器
- 同步或异步操作
- 通过 T1G 引脚门控 TIMER1（使能计数）
- 溢出中断
- 溢出时唤醒（仅外部时钟异步模式）
- 捕捉功能的时基



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。

图 9.1 TMR1 结构图

9.2 TIMER1 的工作原理

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

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

9.3 时钟源选择

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

时钟源	TMR1CS
Fosc	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 Fosc 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（且单片机正在使用不带 CLKOUT 的 INTOSC），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）：

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

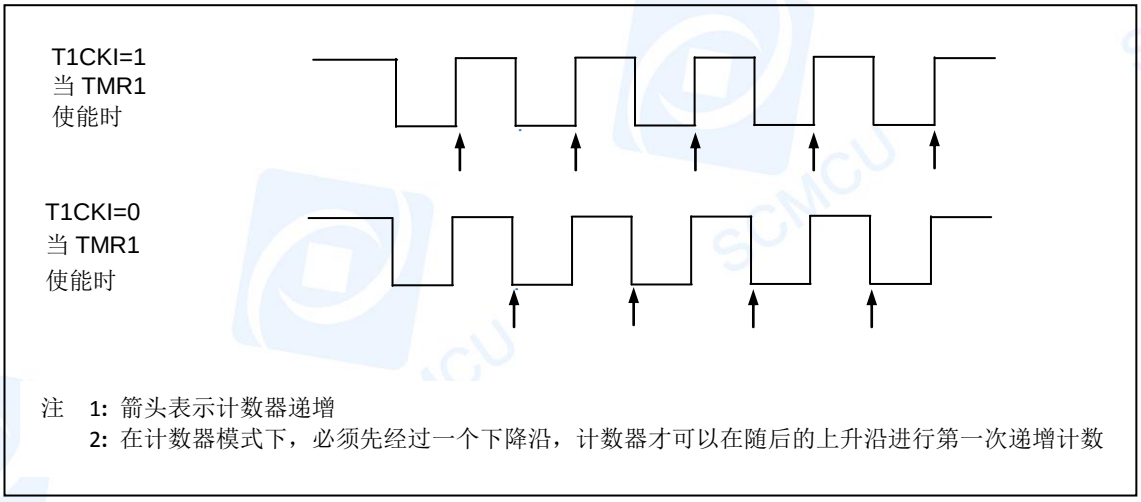


图 9.2 TIMER1 的递增边沿

9.4 TIMER1 预分频器

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

9.5 TIMER1 振荡器

在 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 TRISA6 和 TRISA7 位被置 1。RA6 和 RA7 位读为 0 且 TRISA6 和 TRISA7 位读为 1。

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

注：当从同步操作切换到异步操作时，有可能漏过一个递增。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:TTMR1L 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。有关如何选择 TIMER1 门控信号源的信息，请参见 2.1.2 数据存储器（表 2-1）。此功能部件可以仅仅是△-ΣA/D 转换器的软件，也可以是很多其他应用。

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

可使用 T1CON 寄存器的 T1GINV 位来设置 TIMER1 门控信号的极性，该位可将 TIMER1 配置为对两个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，器器件将调用中断服务程序（0004h）。

9.10 ECCP 捕捉时基

ECCP 模块使用 TMR1H:TMR1L 这对寄存器作为其工作在捕捉模式下的时基。

在捕捉模式下，TMR1H:TMR1L 这对寄存器的值在配置事件发生时被复制 CCPRxH:CCPRxL 这对寄存器中。

更多信息请参见第 12.0 节“捕捉/PWM 模块（CCP1 和 CCP2)”。

9.11 TIMER1 控制寄存器

TIMER1控制寄存器TICON(10H)

10H	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

表 9.1 TIMER1 控制寄存器 TICON

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

BIT5-4 **T1CKPS<1:0>:** TIMER1输入时钟预分频比选择位

11=1**:**8预分频比

10=1**:**4预分频比

01=1**:**2预分频比

00=1**:**1预分频比

BIT3 **T1OSCEN:**LP 振荡器使能控制位

1=使能 LP 振荡器作为 TIMER1 的时钟源

0=LP 振荡器关闭

BIT2 **T1SYNC:**TIMER1 外部时钟输入同步控制位

TMR1CS=1**:**

1=不与外部时钟输入同步

0=与外部时钟输入同步

TMR1CS=0**:**

忽略此位。TIMER1 使用内部时钟。

BIT1 **TMR1CS:** TIMER1时钟源选择位

1=来自T1CKI引脚的外部时钟源（上升沿触发）

0=内部时钟源F_{OSC}

BIT0 **TMR1ON:** TIMER1使能位

1=使能TIMER1

0=禁止TIMER1

10. 定时计数器 TIMER2

10.1 TIMER2 概述

TIMER2 模块是一个 8 位定时器/计数器，具有以下特性：

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

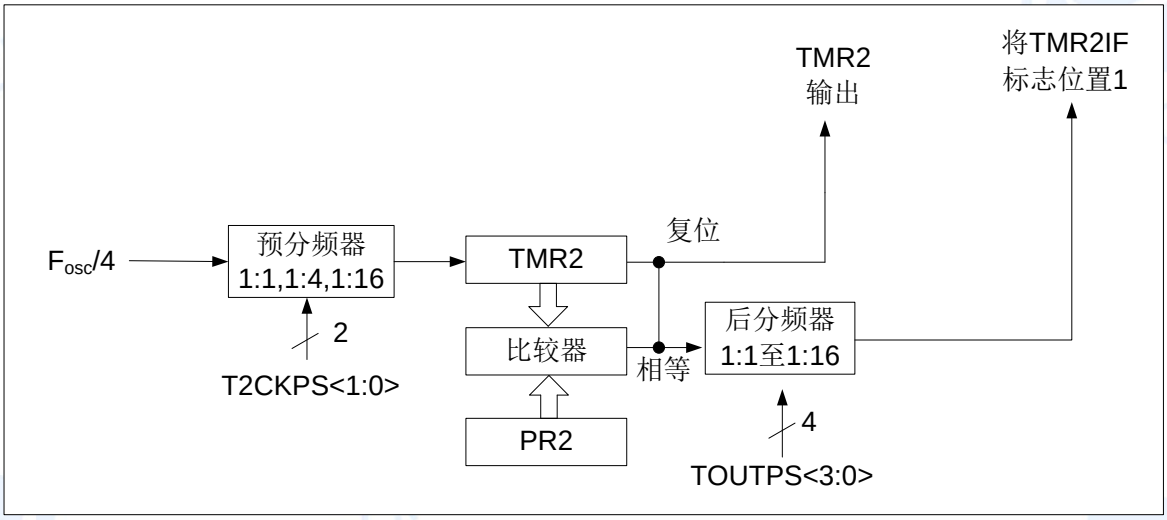


图 10.1 TIMER2 结构框图

10.2 TIMER2 的工作原理

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

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

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

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

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

通过将 T2CON 寄存器的 TMR2ON 位置 1 使能 TIMER2。

通过将 TMR2ON 位清零禁止 TIMER2。

TIMER2 预分频器由 T2CON 寄存器的 T2CKPS 位控制。

TIMER2 后分频器由 T2CON 寄存器的 TOUTPS 位控制。

预分步器和后分步器计数器在以下情况下被清零:

- 对 TMR2 寄存器执行写操作
- 对 T2CON 寄存器执行写操作

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。

10.3 TIMER2 相关的寄存器

有 2 个寄存器与 TIMER2 相关，分别是数据存储器 TMR2，控制寄存器 T2CON。

TIMER2数据寄存器TMR2（11H）

11H	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

表 10.1 TIMER2 数据寄存器 TMR2

TIMER2控制寄存器T2CON(12H)

12H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T2CON	----	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
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

表 10.2 TIMER2 数据寄存器 T2CON

Bit7 **未用，读为 0**

Bit6-3 **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

Bit1-0 **T2CKPS<1:0>:** TIMER2时钟预分频比选择位
00=预分频值为1
01=预分频值为4
1x=预分频值为16

11. 模数转换(ADC)

11.1 ADC 概述

模数转换器(Analog-to-Digital Converter,ADC)可以将模拟输入信号转换为12位二进制代码表示。该模块使用模拟信道通过多路开关连接到一个采样保持电路，采样保持电路的输出与转换器的输入相连接，转换器通过逐次逼近法产生 12 位的二进制结果，并将转换结果存入 ADRESL 跟 ADRESH 寄存器中，并产生中断请求信号 ADIF。

ADC 由如下功能组成:

- 最多 15 通道输入，ADC 参考电压始终为内部产生
- 12-bit 连续近似值寄存器
- 输出寄存器组成(ADRESL、ADRESH)
- 控制寄存器(ADCON0、ADCON1)
- 转换结束产生中断

ADC 结构框图如下:

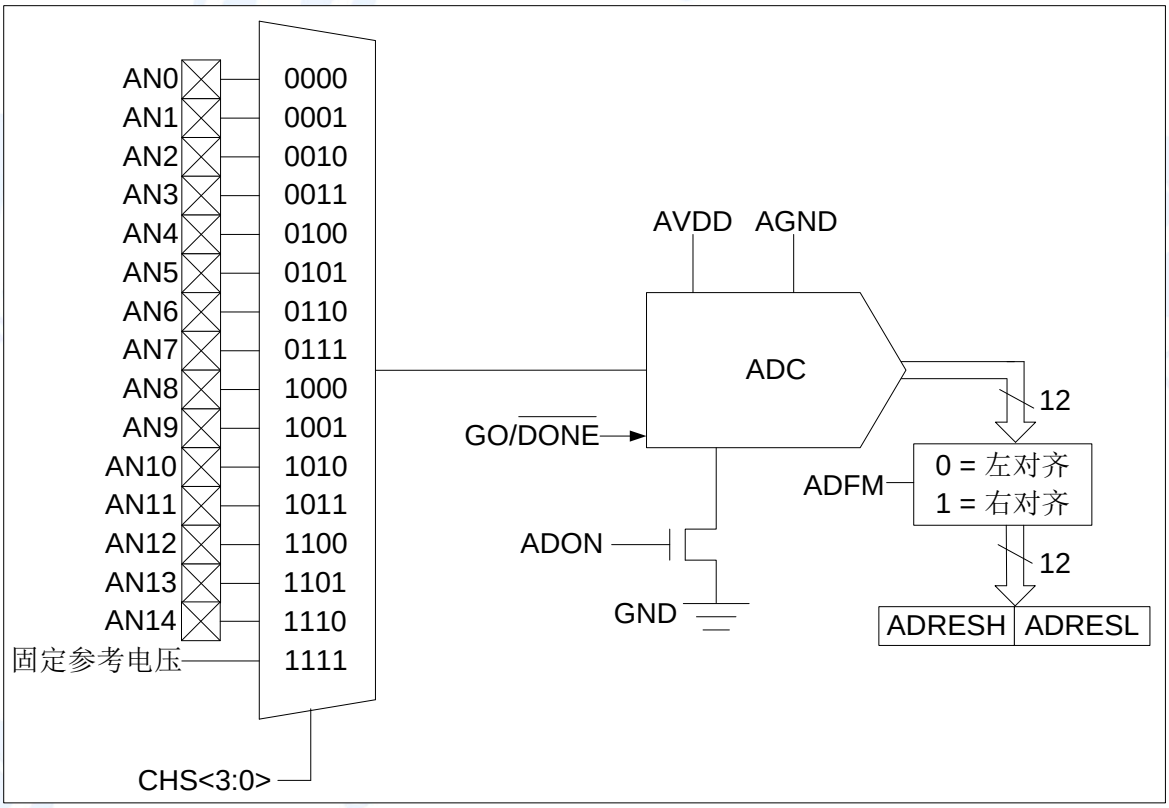


图 11.1 ADC 结构框图

11.2 ADC 配置

配置和使用 ADC 时，必须考虑如下因素:

- 端口配置
- 通道选择
- ADC 转换时钟源
- 中断控制
- 结果的存储格式

11.2.1 端口配置

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

11.2.2 通道选择

由 ADCON0 寄存器的 CHS 位决定将哪个通道连接到采样保持电路。

如果更改了通道，在下一次转换开始前需要一定的延迟。更多信息请参见第 11.3 节“ADC 工作原理”。

11.2.3 ADC 参考电压

ADC 的参考电压始终是由芯片的 VDD 和 GND 提供。

11.2.4 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：

- FOSC/8
- FOSC/16
- FOSC/32
- FRC（专用内部振荡器）

完成一位转换的时间定义为 TAD。一个完整的 12 位转换需要 49 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。下表给出了正确选择 ADC 时钟的示例。

注：除非使用 FRC，否则系统时钟频率的任何改变都会改变 ADC 时钟的频率，从而对 ADC 转换结果产生负面影响

图注: 建议不要使用阴影单元内的值

ADC 时钟周期		器件频率		
ADC 时钟源	ADCS<1:0>	8MHz	4MHz	1MHz
Fosc/8	00	49.0μs	98.0μs	392.0μs
Fosc/16	01	98.0μs	196.0μs	784.0μs
Fosc/32	10	196.0μs	392.0μs	1.5ms
FRC	11	1-3ms	1-3ms	1-3ms

表 11.1　ADC 时钟周期（TAD）与器件工作频率的关系（VDD=5.0V）

11.2.5 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。

不管器件处于工作模式还是休眠模式都可以产生中断。如果器件处于休眠模式，该中断可将器件唤醒。当将器件从休眠状态唤醒后，总是执行 SLEEP 指令后的下一条指令。如果用户尝试使器件从休眠模式唤醒并按顺序恢复代码执行，则必须禁止全局中断。如果允许全局中断，程序将跳转到中断服务程序处执行。

11.2.6 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 ADCON0 寄存器的 ADFM 位控制输出格式。

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

11.3 ADC 工作原理

11.3.1 启动转换

要使能 ADC 模块，必须将 ADCON0 寄存器的 ADON 位置 1，将 ADCON0 寄存器的 GO/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/DONE 位置 1

11.3.2 完成转换

当转换完成时，ADC 模块将：

- 清零 GO/DONE 位
- 将 ADIF 标志位置 1
- 用转换的新结果更新 ADRESH:ADRESL 寄存器

11.3.3 终止转换

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

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

11.3.4 ADC 在休眠模式下的工作原理

ADC 模块可以工作在休眠模式下。此操作需要将 ADC 时钟源设置为 FRC 选项。如果选择了 FRC 时钟源，ADC 在开始转换之前要多等待一个指令周期。从而允许执行 SLEEP 指令，以降低转换中的系统噪声。如果允许 ADC 中断，当转换结束时，将使器件从休眠模式唤醒。如果禁止 ADC 中断，即使 ADON 位保持置 1，则转换结束后也还是会关闭 ADC 模块。如果 ADC 时钟源不是 FRC，即使 ADON 位仍保持置 1，执行 SLEEP 指令还是会中止当前的转换并关闭 A/D 模块。

11.3.5 A/D 转换步骤

如下步骤给出了使用 ADC 进行模数转换的示例**:**

1、端口配置**:**
   - ◆ 禁止引脚输出驱动器（见 TRIS 寄存器）

2、设置 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、设置 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、GO/DONE 置 1 启动转换。

6、如下方法之一等待 ADC 转换结束:
   - ◆ 查询 GO/DONE 位
   - ◆ 等待 ADC 中断（允许中断）

7、ADC 结果

8、ADC 中断标志位清零（如果允许中断的话，需要进行此操作）。

注: 如果用户尝试在使器件从休眠模式唤醒后恢复顺序代码执行，则必须禁止全局中断。

例 AD 转换

```
MOVLW    B'10000000'
MOVWF    ADCON1
BSF      TRISA,0        ;设置 RA0 为输入口
MOVLW    B'11000001'
MOVWF    ADCON0
CALL     DELAY          ;延时一段时间
BSF      ADCON0,GO
BTFSC    ADCON0,GO      ;等待 AD 转换结束
GOTO     $-1
MOVF     ADRESH,0       ;保存 AD 转换结果高位
MOVWF    RESULTH
MOVF     ADRESL,0       ;保存 AD 转换结果低位
```

```
MOVWF       RESULTL
```

11.4 ADC 相关寄存器

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

AD控制寄存器ADCON0(1FH)

1FH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADCON0	ADCS1	ADCS0	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
读写	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

表 11.1 AD 控制寄存器 ADCON0

Bit7-6 **ADCS<1:0>:** A/D转换时钟选择位
00=FOSC/8
01=FOSC/16
10=FOSC/32
11=FRC（由专用的内部振荡器产生频率最高为32kHz的时钟）

Bit5-2 **CHS<3:0>:** 模拟通道选择位
0000=AN0
0001=AN1
0010=AN2
0011=AN3
0100=AN4
0101=AN5
0110=AN6
0111=AN7
1000=AN8
1001=AN9
1010=AN10
1011=AN11
1100=AN12
1101=AN13
1110=CVREF
1111=固定参考电压（0.6V固定参考电压）

Bit1 **GO/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。转换完成以后，该位自动清零。
0=A/D转换完成/或不在进行中

Bit0 **ADON:** ADC使能位
1=使能ADC

AD控制寄存器ADCON1(9FH)

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

表 11.2 AD 控制寄存器 ADCON1(9FH)

Bit7 **ADFM:** A/D转换结果格式选择
1=右对齐
0=左对齐

Bit6-0 **未用，读为0**

AD数据寄存器高位ADRESH(1EH)，ADFM=0

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	ADRES11	ADRES10	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4
读写	R	R	R	R	R	R	R	R
复位值	x	x	x	x	x	x	x	x

表 11.3 AD 数据寄存器高位 ADRESH

Bit7-0 **ADRES<11:4>:** ADC结果寄存器位
12位转换结果的高8位

AD数据寄存器低位ADRESL(9EH)，ADFM=0

9EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES3	ADRES2	ADRES1	ADRES0	----	----	----	----
读写	R	R	R	R	----	----	----	----
复位值	x	x	x	x	----	----	----	----

表 11.4 AD 数据寄存器低位 ADRESL

Bit7-4 **ADRES<3:0>:** ADC结果寄存器位
12位转换结果的低4位

Bit3-0 **未用**

AD数据寄存器高位ADRESH(1EH)，ADFM=1

1EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESH	----	----	----	----	----	----	ADRES11	ADRES10
读写	----	----	----	----	----	----	R	R
复位值	----	----	----	----	----	----	x	x

表 11.5 AD 数据寄存器高位 ADRESH

Bit7-2 **未用**

Bit1-0 **ADRES<11:10>:** ADC结果寄存器位
10位转换结果的高2位

AD数据寄存器低位ADRESL(9EH)，ADFM=1

9EH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
ADRESL	ADRES9	ADRES8	ADRES7	ADRES6	ADRES5	ADRES4	ADRES3	ADRES2
读写	R	R	R	R	R	R	R	R
复位值	x	x	x	x	x	x	x	x

表 11.6 AD 数据寄存器低位 ADRESL

Bit7-0 **ADRES<9:2>:** ADC结果寄存器位
10位转换结果的低8位

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

12. 捕捉/ PWM 模块（CCP1 和 CCP2）

SC8F301X 包含 2 个捕捉/ PWM（CCP1）和（CCP2）。CCP1 和 CCP2 模块的操作基本相同。

注：本文档中的 CCPRx 和 CCPx 分别指 CCPR1 或 CCPR2 和 CCP1 或 CCP2

12.1 捕捉/ PWM（CCPx）

捕捉/比较/PWM 模块是允许用户定时和控制不同事件的外设。在捕捉模式下，该外设能对事件的持续时间计时。捕捉模式允许用户在预先确定的定时时间结束后触发一个外部事件。PWM 模式可产生频率和占空比都可变化的脉宽调制信号。

当 CCP 用在捕捉模式下，需要用到定时器 TIMER1；当其用在 PWM 模式下则需要用定时器 TIMER2。

CCPx控制寄存器CCPxCON

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CCPxCON	----	----	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
读写	----	----	R/W	R/W	R/W	R/W	R/W	R/W
复位值	0	0	0	0	0	0	0	0

表 12.1 CCPx 控制寄存器 CCPxCON

Bit7-6 未用

Bit5-4 **DCxB<1:0>:** PWM占空比的低两位。

捕捉模式:

未使用

PWM模式:

这两位是10位PWM占空比的低2位。占空比的高8位在CCPRxL中。

Bit3-0 **CCPxM<3:0>:** CCPx模式选择位

0000=捕捉/比较/PWM关闭（复位CCPx模块）

0001=未使用（保留）

0010=未使用（保留）

0011=未使用（保留）

0100=捕捉模式，在每个下降沿发生捕捉

0101=捕捉模式，在每个上升沿发生捕捉

0110=捕捉模式，每4个上升沿发生捕捉

0111=捕捉模式，每16个上升沿发生捕捉

1x00=未使用（保留）

11xx=PWM模式

12.2 捕捉模式

在捕捉模式下，当对应的 CCPx 引脚发生事件时，CCPRxH:CCPRxL 这对寄存器捕捉 TMR1 寄存器的 16 位值。触发捕捉的事件可被定义为以下四者之一，并且由 CCP1CON 寄存器中的 CCP1M<3:0>位配置：

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

通过模式选择位 CCPxM3:CCPxM0（CCPxCON<3:0>）选择事件类型。当一个捕捉发生时，中断请求标志位 PIRx 寄存器中的 CCPxIF 置 1；它必须用软件清零。如果在 CCPRxH 和 CCPRxL 这对寄存器中的值被读取之前发生另一次捕捉，那么之前捕捉的值将被新捕捉的值覆盖（见图 12.1）。

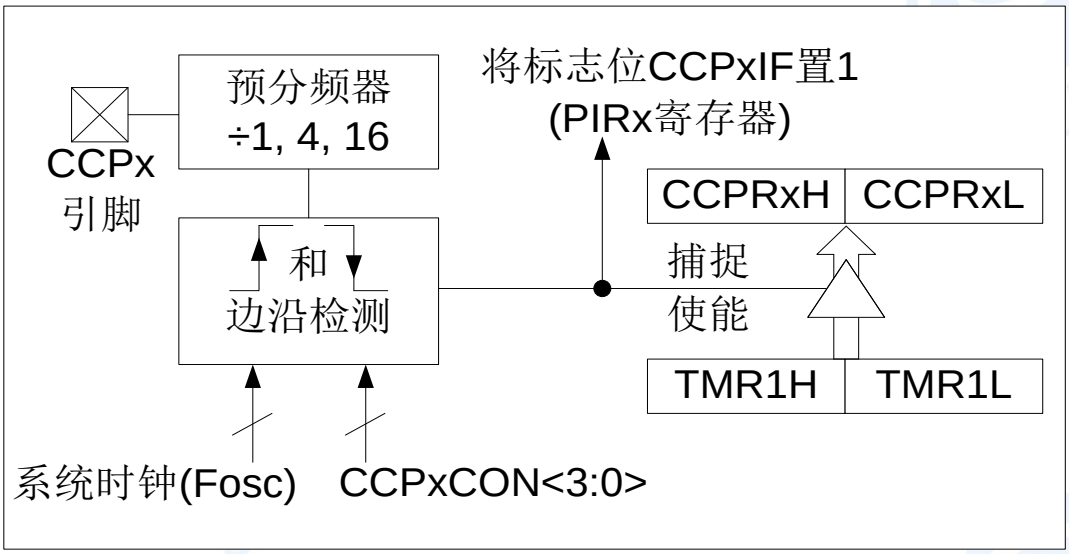


图 12.1 捕捉模式工作框图

12.2.1 CCP 引脚配置

在捕捉模式下，应通过将对应的 TRIS 控制位置 1 来将相应的 CCPx 引脚配置为输入。

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

12.2.2 TIMER1 模式选择

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

12.2.3 软件中断

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

12.2.4 CCP 预分频器

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

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

例 12.1 改变捕捉预分频比:

```
CLRF    CCP1CON          ;关闭 CCP1
MOVLW   B'00000101'
MOVWF   CCP1CON          ;给 CCP1 赋新值
```

12.3 PWM 模式

PWM 模式在 CCPx 引脚上产生脉宽调制信号。由以下寄存器确定占空比、周期和分辨率:

- PR2
- T2CON
- CCPRxL
- CCPxCON

在脉宽调制（PWM）模式下，CCP 模块可在 CCPx 引脚上输出分辨率高达 10 位的 PWM 信号。由于 CCPx 引脚与端口数据锁存器复用，必须清零相应的 TRIS 位才能使能 CCPx 引脚的输出驱动器。

注: 清零 CCPxCON 寄存器将放弃 CCPx 对 CCPx 引脚的控制权。

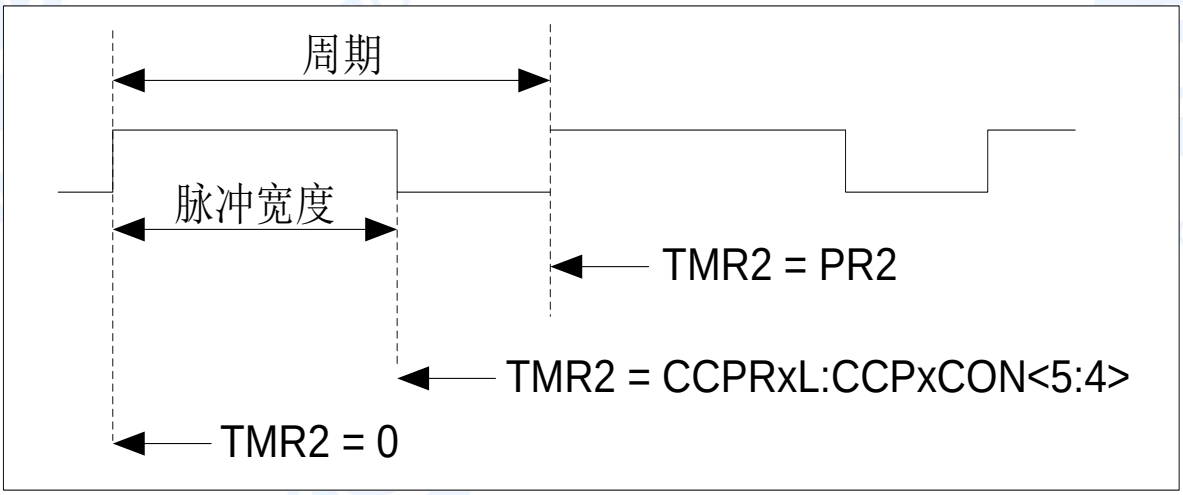


图 12.2 CCPPWM 输出

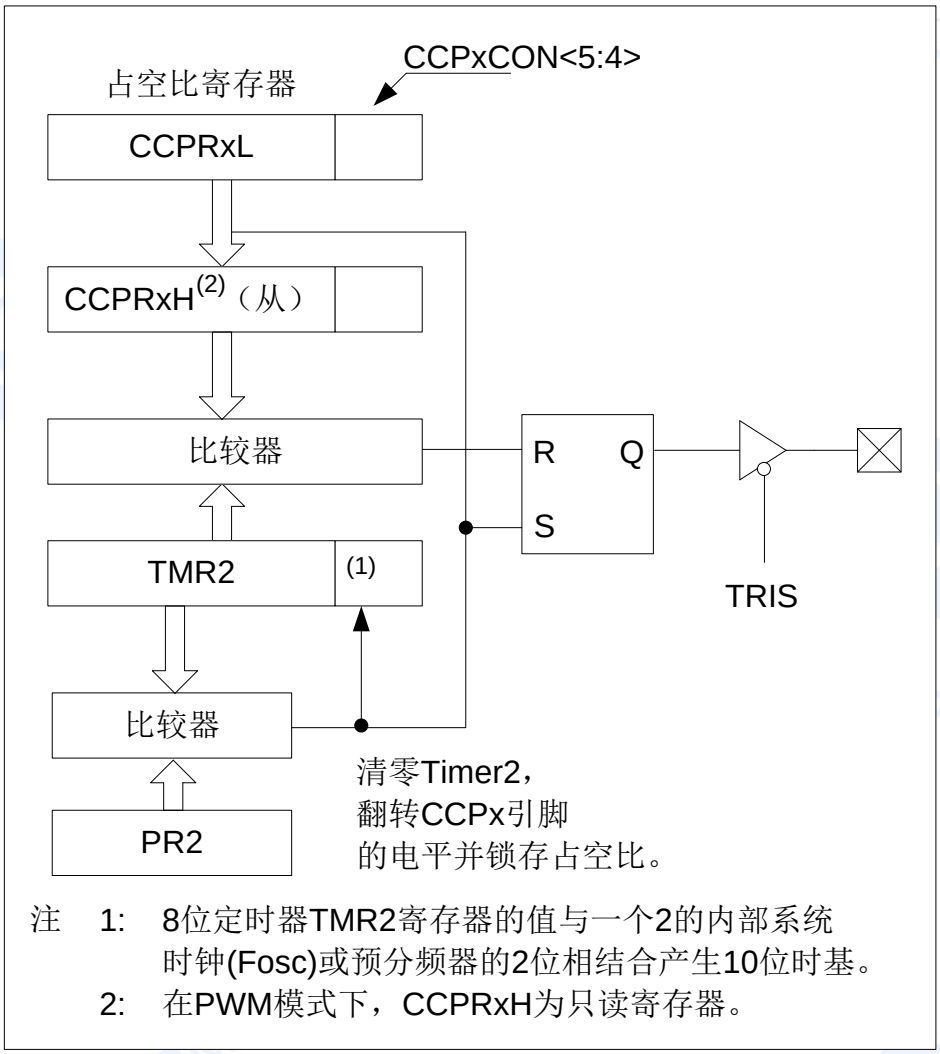


图 12.3 PWM 简化框图

12.3.1 PWM 周期

PWM 周期是通过写 TIMER2 的 PR2 寄存器来指定的。

可以使用如下公式计算 PWM 周期：

$$PWM周期 = [(PR2)+1] * 4 * T_{OSC} * (TMR2预分频值)$$

注：$T_{OSC} = 1/F_{OSC}$

当 TMR2 等于 PR2 时，在下一个递增计数周期中会发生以下 3 个事件：

- TMR2 被清零
- CCPx 引脚被置 1（例外情况：如果 PWM 占空比=0%，CCPx 引脚将不被置 1）
- PWM 占空比从 CCPRxL 被锁存到 CCPRxH

注：在确定 PWM 频率时不使用 TIMER2 后分频比（见第 10.2 节“TIMER2 的工作原理”）

12.3.2 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:CCPRxL 寄存器和 CCPxCON 寄存器的 DCxB<1:0>位。CCPRxL 保存占空比的高 8 位，而 CCPxCON 寄存器的 DCxB<1:0>位保存占空比的低 2 位。可以在任何时候写入 CCPRxL 和 CCPxCON 寄存器的 DCxB<1:0>位，但直到 PR2 和 TMR2 中的值匹配（即周期结束）时，占空比的值才被锁存到 CCPRxH 中。在 PWM 模式下，CCPRxH 是只读寄存器。

可以使用如下公式计算 PWM 脉冲宽度:

$$\text{脉冲宽度}=(\text{CCPRXL}:\text{CCPXCON}<5:4>)*T_{OSC}*(\text{TMR2预分频值})$$

可以使用如下公式计算 PWM 占空比:

$$\text{占空比}=\frac{(\text{CCPRxL}:\text{CCPxCON}<5:4>)}{4(\text{PR2}+1)}$$

CCPRxH 寄存器和一个 2 位的内部锁存器用于为 PWM 占空比提供双重缓冲。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。

8 位定时器 TMR2 寄存器的值与一个 2 位的内部系统时钟（FOSC）或预分频器的 2 位相结合，产生 10 位时基。当 TIMER2 预分频比为 1:1 时使用系统时钟。

当 10 位时基与 CCPRxH 和 2 位锁存器相结合的值匹配时，CCPx 引脚被清零（见上图 12.2）。

12.3.3 PWM 分辨率

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

当 PR2 为 255 时，PWM 的最大分辨率为 10 位。如下面公式所示，分辨率是 PR2 寄存器值的函数。

可以使用如下公式计算 PWM 分辨率：

$$\text{分辨率}=\frac{\log[4(\text{PR2}+1)]}{\log(2)}$$

注: 如果脉冲宽度大于周期值，指定的 PWM 引脚将保持不变。

下列表格给出了在 Fosc=8M 的情况下，PWM 的频率和分辨率的值。

PWM 频率（KHZ）	1.22	4.90	19.61	76.92	153.85	200.0
定时器预分频值（1、4 或 16）	16	4	1	1	1	1
PR2 值	0x65	0x65	0x65	0x19	0x0C	0x09
最高分辨率（位）	8	8	8	6	5	5

表 12.2 PWM 频率和分辨率示例（FOSC=8MHz）

12.3.4 休眠模式下的操作

在休眠模式下，TMR2 寄存器将不会递增并且模块的状态将保持不变。如果 CCPx 引脚有输出，将继续保持该输出值不变。当器件被唤醒时，TMR2 将从原先的状态继续工作。

12.3.5 系统时钟频率的改变

PWM 频率是由系统时钟频率产生的。系统时钟频率发生任何改变都会使 PWM 频率发生变化。

12.3.6 复位的影响

任何复位都会将所有端口强制为输入模式，并强制 CCP 寄存器进入其复位状态。

12.3.7 设置 PWM 操作

在将 CCP 模块配置为 PWM 操作模式时应该执行以下步骤:

1、通过将相应的 TRIS 位置 1，禁止 PWM 引脚（CCPx）的输出驱动器，使之成为输入引脚。

2、通过装载 PR2 寄存器设置 PWM 周期。

3、通过用适当的值装载 CCPxCON 寄存器配置 CCP 模块的 PWM 模式。

4、通过装载 CCPRxL 寄存器和 CCPxCON 寄存器中的 DCxB<1:0>位设置 PWM 占空比。

5、 配置并启动 TIMER2:

- 清零 PIR1 寄存器中的 TMR2IF 中断标志位。
- 通过装载 T2CON 寄存器的 T2CKPS 位来设置 TIMER2 预分频比。
- 通过将 T2CON 寄存器中的 TMR2ON 位置 1 来使能 TIMER2。

6、在新的 PWM 周期开始后，使能 PWM 输出:

- 等待 TIMER2 溢出（PIR1 寄存器中的 TMR2IF 位置 1）。
- 通过将相应的 TRIS 位清零，使能 CCPx 引脚输出驱动器。

13. 触摸按键模块

13.1 触摸按键模块概述

SC8F301X 的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:

- 1-16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

SC8F301X 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。

内部电路框图如下。

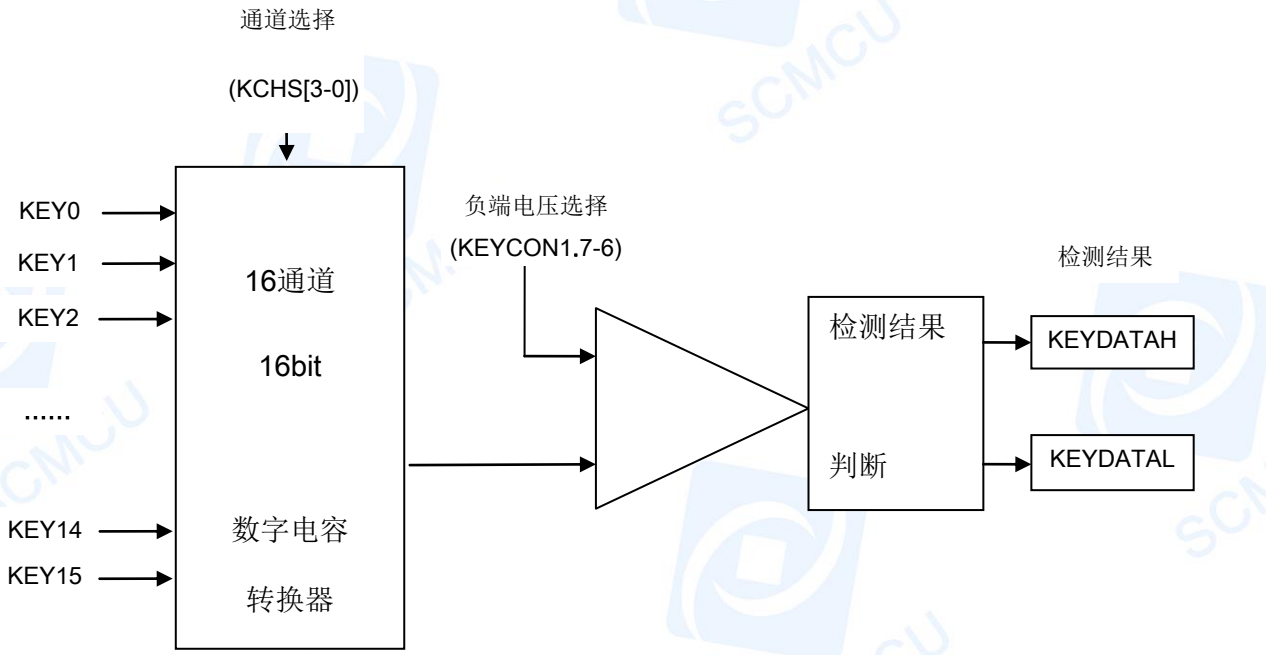


图 13.1 SC8F301X 触摸模块内部电路框图

13.2 与触摸按键相关的寄存器

有 4 个寄存器与触摸按键相关，分别是触摸控制寄存器 KEYCON0(91H),KEYCON1(92H)，触摸按键结果寄存器 KEYDATAL(93H),KEYDATAH(94H)

触摸按键结果寄存器KEYDATAL(93H)

93H	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

表13.1 触摸按键结果寄存器KEYDATAL

触摸按键结果寄存器KEYDATAH(94H)

94H	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

表13.2 触摸按键结果寄存器KEYDATAH

KEYDATAH 跟 KEYDATAL 是触摸按键结果寄存器，是只读寄存器，当完成触摸检测后可从 KEYDATAH 跟 KEYDATAL 读取检测结果。

触摸按键控制寄存器KEYCON0(91H)

91H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
KEYCON0	KDONE	----	CAPK[2:0]			KTOUT	KCAP	KEN
R/W	R	----	R/W	R/W	R/W	R	R/W	R/W
复位值	0	----	0	0	0	0	0	0

表13.3 触摸按键控制寄存器KEYCON0

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 **未用**

BIT5-BIT3 **CAPK[2:0]按键口内部并联电容选择（C≈0.4pF）**
000: 按键口不并联电容
001: 按键口并联一个 C×1 的电容
010: 按键口并联一个 C×2 的电容
011: 按键口并联一个 C×3 的电容
100: 按键口并联一个 C×4 的电容
101: 按键口并联一个 C×5 的电容
110: 按键口并联一个 C×6 的电容
111: 按键口并联一个 C×7 的电容

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（R∪管脚）**
0: 禁止
1: 使能

BIT0 **KEN: 触摸检测使能位**
0: 关闭触摸模块
1: 打开触摸模块开始检测

触摸按键控制寄存器KEYCON1(92H)

92H	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

表 13.4　触摸按键控制寄存器 KEYCON1

BIT7- Bit6 **KVREF：触摸按键内部比较器正端电压选择**
00: 0.4VDD
01: 0.5VDD
10: 0.6VDD
11: 0.7VDD

BIT5-BIT4 **KCLK：触摸按键时钟选择**
00: Fsys
01: Fsys/2
10: Fsys/4
11: Fsys/8

BIT3- BIT0 **KCHS：检测通道选择**
KCHS[3:0]:
0000：选择 KEY0 通道
0001：选择 KEY1 通道
0010：选择 KEY2 通道
0011：选择 KEY3 通道
…
…
…
1110：选择 KEY14 通道
1111：选择 KEY15 通道

13.3 触摸按键模块应用

13.3.1 用查询模式读取“按键数据值”流程

1、设置相应 IO 口(包括按键口和灵敏度调节电容口)为输入口
2、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、比较器正端电压设置)
3、设置按键控制寄存器 KEYCON0(使能触摸电容口，设置按键口是否需要并联电容)
4、开始检测按键: KEYCON0.0 位 KEN 从 0 到 1 变化
5、判断按键结束标志 KEYCON0.7 位 KDONE 是否为 1
6、读取 16 位数据
7、结束检测按键: KEN=0
8、返回第 2 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:

```
        MOVLW   00H
        MOVWF   INTCON      ;中断关闭
        MOVLW   B'00000001'
        MOVWF   TRISA       ;设置 RA0 口为按键检测口
        MOVLW   B'00100000'  ;设置 RB5 口为灵敏度电容口
        MOVWF   TRISB
        MOVLW   B'01010000'
        MOVWF   KEYCON1     ;设置比较器正端电压、触摸检测时钟、通道
        MOVLW   02H
        MOVWF   KEYCON0     ;设置检测通道、分频比、比较器电压
        BSF     KEYCON0,0   ;开始检测
WAIT:
        BTFSS   KEYCON0,7   ;等待检测结束
        GOTO    WAIT
        MOVF    KEYDATAH,0
        MOVWF   R01         ;保存高 8 位结果到用户自定义 RAM 里面
        MOVF    KEYDATAL,0
        MOVWF   R02         ;保存低 8 位结果到用户自定义 RAM 里面
        BCF     KEYCON0,0   ;结束检测
        GOTO    XXXX        ;转到其它程序
```

13.3.2 判断按键方法

1、判断基础: 没键按下---“数据”大；有键按下---“数据”小
2、当前的值比以前的值小到一定程度，可认为“有键”
3、在一定时间内，“数据”由大到小变化认为有键按下(参见下文例程)

13.4 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联机尽量短和细，如果 PCB 工艺允许尽量采用 5mil 的线宽。
- 感应盘到触摸芯片的联机不要跨越强干扰、高频的信号线。
- 感应盘到触摸芯片的联机周围 0.5mm 不要走其它信号线。

例：判断有没有按键举例：

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。

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

14. LCD 驱动功能

14.1 LCD 控制功能说明

SC8F2912\2913\2913S 由 4 个 I/O 口内置上下拉电阻，可用来做 1/2Bias 的 LCD 驱动，其控制流程为：

1、 将 LCDCON.7 位 LCD_EN 置 1；
2、 设置相应的 COM_EN 位为“1”，允许该管脚为 1/2Bias 的 COM 口；
3、 当其中一个 COM 口需要点亮做输出的时候，需要把该管脚的 TRIS 位置“0”，并将相应的 COM_EN 位置“0”。

相关寄存器如下

LCD 控制寄存器 LCDCON(9CH)

9CH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
LCDCON	LCD_EN	LCD_ISLE[1:0]		----	COM_EN[3:0]			
读写	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

表 13.1 LCD 控制寄存器 LCDCON

Bit7 **LCD_EN: LCD**使能位
0=LCD禁止
1=LCD使能

Bit6-5 **LCD_ISLE[1:0]: LCD COM**口电流控制位
00=25uA
01=50uA
10=100uA
11=200uA

Bit4 未用

Bit3-0 **COM_EN[3:0]: LCD COM**口使能控制位
0=对应COMx口为普通IO口；
1=对应COMx口为LCD功能的COM口；

15. 电气参数

15.1 DC 特性

符号	参数	测试条件		最小	典型	最大	单位
		VDD	条件				
VDD	工作电压	-	8MHz	3.0	-	5.5	V
		-	4MHz	2.5	-	5.5	V
Idd	工作电流	5V	ADC 使能	-	3	-	mA
		3V	ADC 使能	-	2	-	mA
Istb	静态电流	5V	----	-	0.1	10	μA
		3V	----	-	0.1	5	μA
Vil	低电平输入电压	-	----	-	-	0.3VDD	V
Vih	高电平输入电压	-	----	0.7VDD	-	-	V
Voh	高电平输出电压	-	不带负载	0.9VDD	-	-	V
VoL	低电平输出电压	-	不带负载	-	-	0.1VDD	V
VADI	AD 口输入电压	-	----	0	-	VDD	V
VAD	AD 模块工作电压	-	----	2.7	-	5.5	V
VEEPROM	EEPROM 模块工作电压	-	----	3.0	-	5.5	V
EAD	AD 转换误差	-	----	-	±2	-	-
Rph	上拉电阻阻值	5V	----	-	35	-	K
		3V	----	-	65	-	K
IoL	输入口灌电流	5V	Vol=0.3VDD	-	60	-	mA
		3V	Vol=0.3VDD	-	25	-	mA
IoH	输入口拉电流	5V	Voh=0.7VDD	-	15	-	mA
		3V	Voh=0.7VDD	-	10	-	mA

表 14.1 DC 特性

15.2 AC 特性

符号	参数	测试条件		最小	典型	最大	单位
		VDD	条件				
TWDT	WDT 复位时间	5V	---	-	18	-	ms
		3V	---	-	36	-	ms
TAD	AD 转换时间	5V	---	-	49	-	CLK
		3V	---	-	49	-	CLK
TEEPROM	EEPROM 写入时间	5V	---	-	2.5	-	ms
		3V	---	-	2.5	-	ms

表 14.2 AC 特性

15.3 内部 RC 振荡特性

15.3.1 内部 RC 振荡电压特性

测试条件	振荡频率(典型值)(HZ)
2.5V	8.0M
2.6V	8.1M
2.8V	8.2M
3.0V	8.3M
3.2V	8.3M
3.4V	8.2M
3.6V	8.2M
3.8V	8.2M
4.0V	8.1M
4.2V	8.1M
4.4V	8.1M
4.6V	8.0M
4.8V	8.0M
5.0V	8.0M
5.2V	8.0M
5.4V	7.9M
5.5V	7.8M

表 14.3 内部 RC 震荡电压特性

15.3.2 内部 RC 振荡温度特性

测试条件	-25℃	25℃	40℃	60℃	85℃
振荡频率(典型值)(Hz)	7.9M	8.0 M	8.0M	8.1M	8.1M

表 14.4 内部 RC 振荡温度特性

16. 指令

16.1 指令一览表

对于字节操作指令，f 为代表文件寄存器的指示符，而 d 为代表目标寄存器的指示符。文件寄存器指示符指定指令将会使用存储区内 0-FFH 地址的文件寄存器中的一个。

目标寄存器指示符指定操作结果的存放位置。如果 d 为 0，结果存回 W 寄存器，此时目标寄存器为 W。如果 d 为 1，操作结果存入指令指定的文件寄存器中，此时目标寄存器为指定寄存器。(即 d ∈[0,1])

对于位操作指令，b 为代表位域的指示符，用于选择操作所影响的位，而 f 则代表相应位所在的寄存器的地址。

对于立即数和控制操作指令，k 代表一个 8 位或 9 位常数或立即数值。

✧ **指令集汇总**

助记符	操作	指令周期	标志
控制类-4 条			
NOP	空操作	1	None
SLEEP	进入休眠模式	1	TO,PD
OPTION	装载 OPTION_REG 寄存器	1	None
CLRWDT	清零看门狗计数器	1	TO,PD
数据传送-3 条			
MOVWF f	将 W 数据传送到 f	1	NONE
MOVF f,d	将 f 数据传送到目标寄存器	1	Z
MOVLW k	立即数 k 传送给 W	1	NONE
逻辑运算-11 条			
CLRW	清零 W 寄存器	1	Z
SETF f	置位数据寄存器 f	1	NONE
CLRF f	清零数据存储器 f	1	Z
IORWF f,d	f 与 W 内容做“或”运算，结果存入目标寄存器	1	Z
ANDWF f,d	f 与 W 内容做“与”运算，结果存入目标寄存器	1	Z
XORWF f,d	f 与 W 内容做“异或”运算，结果存入目标寄存器	1	Z
SWAPF f,d	f 寄存器内容的高低半字节转换，结果存入目标寄存器	1	NONE
COMF f,d	f 寄存器内容取反，结果存入目标寄存器	1	Z
XORLW k	W 与立即数 k 做“异或”运算，结果存入 W	1	Z
ANDLW k	W 与立即数 k 做“与”运算，结果存入 W	1	Z
IORLW k	W 与立即数 k 做“或”运算，结果存入 W	1	Z
移位操作-4 条			
RRF f,d	数据存储器带进位循环右移一位，结果存入目标寄存器	1	C
RLF f,d	数据存储器带进位循环左移一位，结果存入目标寄存器	1	C
RRNCF f,d	数据存储器不带进位循环右移一位，结果存入目标寄存器	1	NONE

RLNCF f,d	数据存储器不带进位循环左移一位，结果存入目标寄存器	1	NONE
递增递减-2 条			
INCF f,d	递增数据存储器 f,结果存入目标寄存器	1	Z
DECF f,d	递减数据存储器 f,结果存入目标寄存器	1	Z
位操作-2 条			
BCF f,b	将数据存储器 f 中某位清零(第 b 位)	1	NONE
BSF f,b	将数据存储器 f 中某位置一(第 b 位)	1	NONE
查表-2 条			
TABLE f	读取 FLASH 内容结果放入 TABLE_DATAH 与 f	2	NONE
TABLEA	读取 FLASH 内容结果放入 TABLE_DATAH 与 W	2	NONE
数学运算-9 条			
ADDWF f,d	W+f→目标寄存器	1	Z,C,DC,OV
ADDLW k	W+k→W	1	Z,C,DC,OV
SUBWF f,d	f-W→目标寄存器	1	Z,C,DC,OV
SUBLW k	k-W→W	1	Z,C,DC,OV
ADDWFC f,d	W+f+C→目标寄存器	1	Z,C,DC,OV
SUBWFB f,d	f-W-C→目标寄存器	1	Z,C,DC,OV
HSUBWF f,d	当 d=0 时，W-f→W； 当 d=1 时，W-f-C→W	1	Z,C,DC,OV
HSUBWFB f,d	W-f- $\overline{C}$ →目标寄存器	1	Z,C,DC,OV
HSUBLW k	W-k→W	1	Z,C,DC,OV
无条件转移-5 条			
RETURN	从子程序返回，堆栈顶端的值会存放 PC 中	2	NONE
RETLW k	从子程序带参数返回，将立即数 k 存入 W	2	NONE
RETFIE	从中断程序中返回，堆栈顶的值会存入 PC 中，全局中断位置 1	2	NONE
CALL k	子程序调用，首先将返回地址((PC)＋1)压栈保护，再转入所调用的子程序入口地址执行	2	NONE
GOTO k	无条件跳转，绝对地址跳转，无需分页寻址	2	NONE
条件转移-6 条			
BTFSC f,b	如果数据存储器 f 的 b 位为“0”，则跳过下一条指令	1 or 2	NONE
BTFSS f,b	如果数据存储器 f 的 b 位为“1”，则跳过下一条指令	1 or 2	NONE
INCFSZ f,d	数据存储器 f 加“1”，若结果为”0”，则跳过下一条指令	1 or 2	NONE
DECFSZ f,d	数据存储器 f 减“1”，若结果为”0”，则跳过下一条指令	1 or 2	NONE
TSTFSZW f	数据存储器 f 送至 W,若内容为”0”，则跳过下一条指令	1 or 2	NONE
TSTFSZ f	数据存储器 f 内容为”0”，则跳过下一条指令	1 or 2	NONE

表 15.1　SCMCU 指令表

16.2 指令说明

ADDWF f,d

操作: 将 f 加 W 的值赋给目标寄存器。

说明: 将 W 寄存器的内容与寄存器 f 相加。如果 d 为 0,结果存入 W 寄存器；如果 d 为 1,结果存回 f 寄存器。

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
MOVLW   09H     ;给 W 赋值 09H
MOVWF   R01     ;将 W 的值(09H)赋给自定义寄存器 R01
MOVLW   077H    ;给 W 赋值 77H
ADDWF   R01,0   ;执行结果：W=09H + 77H =80H
```

ADDLW k

操作: 将立即数 k 加 W,结果放入 W

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
MOVLW   09H     ;给 W 赋值 09H
ADDLW   077H    ;执行结果：W = W(09H) + i(77H)=80H
```

ANDWF f,d

操作: 寄存器 R 跟 W 进行逻辑与运算。如果 d 为 0,结果存入 W 寄存器；如果 d 为 1,结果存回 f 寄存器。

周期: 1

影响标志位: Z

举例:

```
MOVLW   0FH     ;给 W 赋值 0FH
MOVWF   R01     ;将 W 的值(0FH)赋给寄存器 R01
MOVLW   77H     ;给 W 赋值 77H
ANDWF   R01,1   ;执行结果：R01=(0FH  and  77H)=07H
```

ANDLW k

操作: 将立即数 k 与 W 进行逻辑与运算，结果放入 W。

周期: 1

影响标志位: Z

举例:

```
MOVLW   0FH     ;给 W 赋值 0FH
ANDLW   77H     ;执行结果:W =(0FH  and  77H)=07H
```

CALL k

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
```

CLRW

操作: W 寄存器清零。

周期: 1

影响标志位: Z

举例:

```
CLRW            ;执行结果:W=0
```

CLRF f

操作: 寄存器 f 清零。

周期: 1

影响标志位: Z

举例:

```
CLRF    R01     ;执行结果：R01=0
```

BCF f,b

操作: 寄存器 f 的第 b 位清零。

周期: 1

影响标志位: 无

举例:

```
BCF     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
```

COMF f,d

操作: 寄存器 f 取反，结果放入目标寄存器。

周期: 1

影响标志位: Z

举例:

```
MOVLW   0AH     ;W 赋值 0AH
MOVWF   R01     ;将 W 的值(0AH)赋给寄存器 R01
COMF    R01,0   ;执行结果:W=0F5H
```

DECF f,d

操作: 寄存器 f 自减 1,结果放入目标寄存器。

周期: 1

影响标志位: Z

举例:

```
MOVLW   0AH     ;W 赋值 0AH
MOVWF   R01     ;将 W 的值(0AH)赋给寄存器 R01
DECF    R01,1   ;执行结果:R01=(0AH-1)=09H
```

INCF f,d

操作: 寄存器 f 自加 1,结果放入目标寄存器。

周期: 1

影响标志位: Z

举例:

```
MOVLW   0AH     ;W 赋值 0AH
MOVWF   R01     ;将 W 的值(0AH)赋给寄存器 R01
INCF    R01,0   ;执行结果：W=(0AH+1)=0BH
```

GOTO k

操作: 跳转到 k 地址。

周期: 2

影响标志位: 无

举例:

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

MOVF f,d

操作: 将 f 的值赋给目标寄存器。

周期: 1

影响标志位: Z

举例:

```
MOVF    R01,0     ;将寄存器 R01 的值赋给 W
MOVWF   R02       ;将 W 的值赋给寄存器 R02,实现了数据从 R01→R02 的移动
```

MOVWF f

操作: 将 W 的值赋给 f。

周期: 1

影响标志位: 无

举例:

```
MOVLW   09H       ;给 W 赋值 09H
MOVWF   R01       ;执行结果: R01=09H
```

MOVLW k

操作: 立即数 k 赋给 W。

周期: 1

影响标志位: 无

举例:

```
MOVLW   0AH       ;W 赋值 0AH
```

NOP

操作: 空指令。

周期: 1

影响标志位: 无

举例:

```
NOP
NOP
```

OPTION

操作: 写预分频器

周期: 1

影响标志位: 无

举例:

```
MOVLW   00H
OPTION            ;OPTION_REG 赋值 00H
```

IORLW k

操作: 立即数 k 与 W 进行逻辑或操作，结果赋给 W。

周期: 1

影响标志位: Z

举例:

```
MOVLW   0AH     ;W 赋值 0AH
IORLW   030H    ;执行结果：W =(0AH   or   30H)=3AH
```

IORWF f,d

操作: 寄存器 f 跟 W 进行逻辑或运算，结果放入目标寄存器。

周期: 1

影响标志位: Z

举例:

```
MOVLW   0AH     ;给 W 赋值 0AH
MOVWF   R01     ;将 W(0AH)赋给寄存器 R01
MOVLW   30H     ;给 W 赋值 30H
IORWF   R01,1   ;执行结果：R01=(0AH   or   30H)=3AH
```

RETURN

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             ;返回后将执行这条语句
        …               ;其它程序
LOOP:
        …               ;子程序
        RETURN          ;子程序返回
```

RETLW k

操作: 从子程序带参数返回，参数放入 W。

周期: 2

影响标志位: 无

举例:

```
        CALL    LOOP    ;调用子程序 LOOP
        NOP             ;返回后将执行这条语句
LOOP:
```

```
                ...                 ;子程序
                RETLW   35H         ;子程序返回，W=35H
```

RETFIE

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
                RETIFIE             ;中断返回
```

RLF f,d

操作: 寄存器 f 带进位 C 循环左移一位，结果放入目标寄存器。

周期: 1

影响标志位: C

举例:

```
                MOVLW   03H         ;W 赋值 03H
                MOVWF   R01         ;W 值赋给 R01,R01=03H
                RLF     R01,1       ;操作结果：R01=06H(C=0)；R01=07H(C=1)；C=0
```

RRF f,d

操作: 寄存器 R 带进位 C 循环右移一位，结果放入目标寄存器。

周期: 1

影响标志位: 无

举例:

```
                MOVLW   03H         ;W 赋值 03H
                MOVWF   R01         ;W 值赋给 R01,R01=03H
                RRF     R01,1       ;操作结果：R01=81H
```

BSF f,b

操作: 寄存器 f 的第 b 位置 1。

周期: 1

影响标志位: 无

举例:

```
                CLRF    R01         ;R01=0
                BSF     R01,3       ;操作结果：R01=08H
```

SLEEP

操作: 进入休眠状态。

周期: 1

影响标志位: TO,PD

举例:

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

SUBLW k

操作: 立即数 k 减 W,结果放入 W。

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
MOVLW   077H    ;W 赋值 77H
SUBLW   80H     ;操作结果：W=80H-77H=09H
```

SUBWF f,d

操作: 寄存器 f 减 W,结果放入目标寄存器。

周期: 1

影响标志位: C,DC,Z,OV

举例:

```
MOVLW   080H    ;W 赋值 80H
MOVWF   R01     ;W 的值赋给 R01,R01=80H
MOVLW   77H     ;W 赋值 77H
SUBWF   R01,1   ;操作结果：R01=80H-77H=09H
```

SWAPF f,d

操作: 寄存器 f 高低半字节交换，结果放入目标寄存器。

周期: 1

影响标志位: 无

举例:

```
MOVLW   035H    ;W 赋值 35H
MOVWF   R01     ;W 的值赋给 R01,R01=35H
SWAPF   R01,1   ;操作结果：R01=53H
```

BTFSC f,b

操作: 判断寄存器 f 的第 b 位，为 0 则跳过，否则顺序执行。

周期: 1 or 2

影响标志位: 无

举例:

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

BTFSS f,b

操作: 判断寄存器 f 的第 b 位，为 1 则跳过，否则顺序执行。

周期: 1 or 2

影响标志位: 无

举例:

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

INCFSZ f,d

操作: 将寄存器 f 自加 1，结果放入目标寄存器，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:

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

DECFSZ f,d

操作: 将寄存器 R 自减 1,结果放入目标寄存器，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:

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

XORLW k

操作: 立即数 k 与 W 进行逻辑异或运算，结果放入 W。

周期: 1

影响标志位: Z

举例:

```
MOVLW   0AH     ;W 赋值 0AH
```

```
        XORLW   0FH         ;执行结果: W=05H
```

XORWF f,d

操作: 寄存器 f 与 W 进行逻辑异或运算，结果放入目标寄存器。

周期: 1

影响标志位: Z

举例:

```
        MOVLW   0AH         ;W 赋值 0AH
        MOVWF   R01         ;W 值赋给 R01,R01=0AH
        MOVLW   0FH         ;W 赋值 0FH
        XORWF   R01,1       ;执行结果: R01=05H
```

SETF f

操作: 置位数据寄存器 f　f=0FFH。

周期: 1

影响标志位: NONE

举例:

```
        SETF    R0          R0=0FFH
```

RRNCF f,d

操作: 数据存储器不带进位循环右移一位，结果放入目标寄存器。

周期: 1

影响标志位: NONE

举例:

```
        BSF     STATUS,C    ;STATUS 的第 C 位置 1
        MOVLW   0AH         ;W 赋值 0AH
        MOVWF   R0          ;W 的值赋给 R0
        RRNCF   R0,0        ;执行结果: R0 右移一位，结果存入 W
```

RLNCF f,d

操作: 数据存储器不带进位循环左移一位，结果放入目标寄存器

周期: 1

影响标志位: NONE

举例:

```
        BSF     STATUS,C    ;STATUS 的第 C 位置 1
        MOVLW   0AH         ;W 赋值 0AH
        MOVWF   R0          ;W 的值赋给 R0
```

```
        RLNCF   R0,0        ;执行结果: R0 左移一位，结果存入 W
```

TABLE f

操作: 查表，查表结果低 8 位放入 W，高位放入专用寄存器 TABLE_SPH

周期: 2

影响标志位: NONE

举例:

```
        MOVLW   01H         ;W 赋值 01H
        MOVWF   TABLE_SPH   ;W 值赋给 TABLE_SPH
        MOVLW   015H        ;W 赋值 015H
        MOVWF   TABLE_SPL   ;W 值赋给 TABLE_SPL
        TABLE   R0          ;存入 R0
        MOVF    TABLE_DATAH,0   ;读取 TABLE_DATAH,存入 W
        MOVWF   R1          ;将 W 内容传送到 R1
        ORG     0115H       ;
        DW      1234H       ;
```

TABLEA

操作: 查表，查表结果低 8 位放入 W，高位放入专用寄存器 TABLE_SPH

周期: 2

影响标志位: NONE

举例:

```
        MOVLW   01H         ;W 赋值 01H
        MOVWF   TABLE_SPH   ;W 值赋给 TABLE_SPH
        MOVLW   015H        ;W 赋值 015H
        MOVWF   TABLE_SPL   ;W 值赋给 TABLE_SPL
        TABLEA              ;存入 W
        MOVWF   R0          ;将 W 内容传送到 R0
        MOVF    TABLE_DATAH,0   ;读取 TABLE_DATAH,存入 W
        MOVWF   R1          ;将 W 内容传送到 R1
        ORG     0115H
        DW      1234H
```

ADDWFC f,d

操作: W+f+C→目标寄存器。

周期: 1

影响标志位: Z,C,DC,OV

举例:

```
MOVLW   09H     ;W 赋值 09H
MOVWF   R01     ;W 值赋给 R01,R01=09H
MOVLW   077H    ;W 赋值 077H
ADDWFC  R01,1   ;执行结果:W+f+C→R01
```

SUBWFB f,d

操作: f-W-C→目标寄存器。

周期: 1

影响标志位: Z,C,DC,OV

举例:

```
MOVLW   080H    ;W 赋值 08H
MOVWF   R0      ;W 值赋给 R0,R0=08H
MOVLW   077H    ;W 赋值 077H
SUBWFB  R0,0    ;执行结果：f-W-C→W
```

HSUBWF f,d

操作: 当 d=0 时，W-f→W；当 d=1 时，W-f-C→W。

周期: 1

影响标志位: Z,C,DC,OV

举例:

```
MOVLW   077H    ;W 赋值 077H
MOVWF   R0      ;W 值赋给 R0,R0=077H
MOVLW   080H    ;W 赋值 080H
HSUBWF  R0,1    ;执行结果：d=1,W-f-C→W
```

HSUBWFB f,d

操作: W-f-$\overline{C}$ →目标寄存器。

周期: 1

影响标志位: Z,C,DC,OV

举例:

```
MOVLW   077H    ;W 赋值 077H
MOVWF   R0      ;W 值赋给 R0,R0=077H
MOVLW   080H    ;W 赋值 080H
HSUBWFB R0,1    ;执行结果：W-f-C →f
```

HSUBLW k

操作: W-k→W。

周期: 1

影响标志位: Z,C,DC,OV

举例:

```
MOVLW   077H    ;W 赋值 077H
HSUBLW  066H    ;W-k→W
```

DAW f

操作: 将加法运算中放入 W 的值调整为 10 进制数，并将结果存入 f。

周期: 1

影响标志位: C

举例:

```
MOVLW   077H    ;W 赋值 077H
MOVWF   R0      ;W 值赋给 R0,R0=077H
MOVLW   080H    ;W 赋值 080H
DAW     R0      ;执行结果存入 R0
```

TSTFSZW f

操作: 数据存储器 f 送至 W,若内容为"0"，则跳过下一条指令。

周期: 1 or 2

影响标志位: NONE

举例:

```
MOVLW   080H    ;W 赋值 080H
MOVWF   R01     ;W 值赋给 R01,R01=080H
TSTFSZW R01     ;将 R01 的值送至 W,判断是否为 0
GOTO    LOOP    ;执行结果:将执行这条指令
```

TSTFSZ f

操作: 数据存储器 f 内容为"0"，则跳过下一条指令。

周期: 1 or 2

影响标志位: NONE

举例:

```
MOVLW   080H    ;W 赋值 080H
MOVWF   R01     ;W 值赋给 R01,R01=080H
TSTFSZ  R01     ;查看 R01 内容是否为 0
GOTO    LOOP    ;执行结果: 不跳过
```

16.3 SCMCU 指令与 PIC 指令的区别

16.3.1 指令集概述

本节为芯联发公司所使用 SCMCU 指令集与 PIC 指令集功能对应的相关指令列举，设计者在使用 SCMCU 指令时能够找到与之对应的 PIC 指令，增强程序的复用性及移植性。本着这个原则，芯联发公司开发设计的 SCMCU 指令集兼容了 PIC16F5X、PIC16F8XX 的所有指令，并部分兼容 PIC18FXXX 系列的指令，并在此基础上增加了部分特定的 SCMCU 指令，能够更加方便的进行程序设计。

16.3.2 指令对应表

1. 没有涂颜色的—兼容 PIC16F5X 指令;
2. 涂红色的—兼容 PIC16F8XX 指令;
3. 涂绿色的—兼容 PIC18FXXX 指令;
4. 涂黄色的—PIC 不支持的指令。

指令对应表

SCMCU 指令	D 参数	PIC 指令	操作	指令周期	标志
控制类-4 条					
NOP		NOP	空操作	1	None
OPTION		OPTION	装载 OPTION 寄存器	1	None
SLEEP		SLEEP	进入休眠模式	1	TO,PD
CLRWDT		CLRWDT	清零看门狗计数器	1	TO,PD
数据传送-4 条					
MOVWF f		MOVWF f	将 W 内容传送到 f	1	NONE
MOVF f,d	d=0	MOVF f,d	将 f 内容传送到 W	1	Z
MOVF f,d	d=1	MOVF f,d	将数据存储器内容传给数据存储器	1	Z
MOVLW k		MOVLW k	立即数 k 送给 W	1	NONE
逻辑运算-16 条					
CLRW		CLRW	清零 W 寄存器	1	Z
SETF f		SETF f	置位数据存储器 f	1	NONE
CLRF f		CLRF f	清零数据存储器 f	1	Z

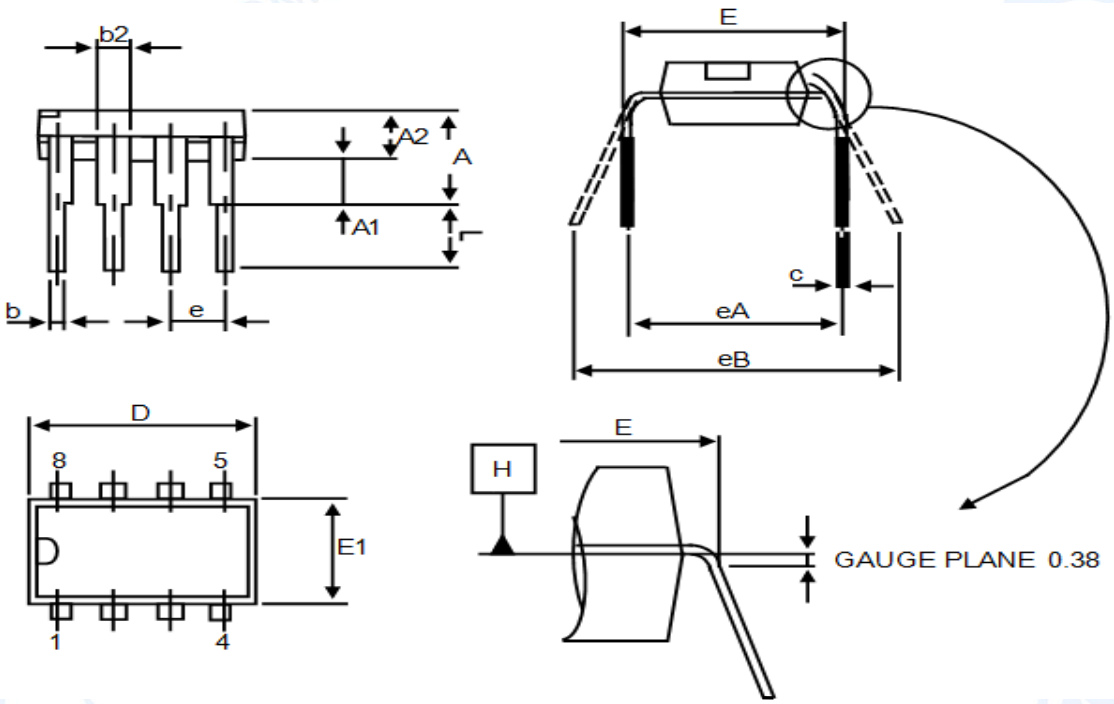
IORWF	f,d	d=0	IORWF	f,d	f 与 W 内容做“或”运算，结果存入 W	1	Z
IORWF	f,d	d=1	IORWF	f,d	f 与 W 内容做“或”运算，结果存入 f	1	Z
ANDWF	f,d	d=0	ANDWF	f,d	f 与 W 内容做“与”运算，结果存入 W	1	Z
ANDWF	f,d	d=1	ANDWF	f,d	f 与 W 内容做“与”运算，结果存入 f	1	Z
XORWF	f,d	d=0	XORWF	f,d	f 与 W 内容做“异或”运算，结果存入 W	1	Z
XORWF	f,d	d=1	XORWF	f,d	f 与 W 内容做“异或”运算，结果存入 f	1	Z
SWAPF	f,d	d=0	SWAPF	f,d	f 寄存器内容的高低半字节转换结果存入 W	1	NONE
SWAPF	f,d	d=1	SWAPF	f,d	f 寄存器内容的高低半字节转换,结果存入 f	1	NONE
COMF	f,d	d=0	COMF	f,d	f 寄存器内容取反，结果存入 W	1	Z
COMF	f,d	d=1	COMF	f,d	f 寄存器内容取反，结果存入 f	1	Z
XORLW	k		XORLW	k	W 与立即数 k 做“异或”运算，结果存入 W	1	Z
ANDLW	k		ANDLW	k	W 与立即数 k 做“与”运算，结果存入 W	1	Z
IORLW	k		IORLW	k	W 与立即数 k 做“或”运算，结果存入 W	1	Z
移位操作-8 条							
RRF	f,d	d=0	RRF	f,d	数据存储器带进位循环右移一位，结果存入 W	1	C
RRF	f,d	d=1	RRF	f,d	数据存储器带进位循环右移一位，结果存入 f	1	C
RLF	f,d	d=0	RLF	f,d	数据存储器带进位循环左移一位，结果存入 W	1	C
RLF	f,d	d=1	RLF	f,d	数据存储器带进位循环左移一位，结果存入 f	1	C
RRNCF	f,d	d=0	RRNCF	f,d	数据存储器不带进位循环左移一位,结果存入 W	1	NONE
RRNCF	f,d	d=1	RRNCF	f,d	数据存储器不带进位循环左移一位，结果存入 f	1	NONE
RLNCF	f,d	d=0	RLNCF	f,d	数据存储器不带进位循环右移一位,结果存入 W	1	NONE
RLNCF	f,d	d=1	RLNCF	f,d	数据存储器不带进位循环右移一位，结果存入 f	1	NONE
递增递减-4 条							
INCF	f,d	d=0	INCF	f,d	递增数据存储器 f,结果放入 W	1	Z
INCF	f,d	d=1	INCF	f,d	递增数据存储器 f,结果放入 f	1	Z
DECF	f,d	d=0	DECF	f,d	递减数据存储器 f,结果放入 W	1	Z
DECF	f,d	d=1	DECF	f,d	递减数据存储器 f,结果放入 f	1	Z
位操作-2 条							
BCF	f,b		BCF	f,b	将数据存储器 f 中某位清零	1	NONE
BSF	f,b		BSF	f,b	将数据存储器 f 中某位置一	1	NONE
查表-2 条							
TABLE	f		PIC 无该指令		读 FLASH 内容结果放入 TABLE_DATAH 与 f	2	NONE
TABLEA			PIC 无该指令		读 FLASH 内容结果放入 TABLE_DATAH 与 W	2	NONE
数学运算-16 条							
ADDWF	f,d	d=0	ADDWF	f,d	W+f→W	1	C,DC,Z,OV
ADDWF	f,d	d=1	ADDWF	f,d	W+f→f	1	C,DC,Z,OV
ADDWFC	f,d	d=0	ADDWFC	f,d	W+f+C→W	1	Z,C,DC,OV
ADDWFC	f,d	d=1	ADDWFC	f,d	W+f+C→f	1	Z,C,DC,OV

ADDLW k	------	ADDLW k	W+k→f	1	Z,C,DC,OV
SUBWF f,d	d=0	SUBWF f,d	f-W→W	1	C,DC,Z,OV
SUBWF f,d	d=1	SUBWF f,d	f-W→f	1	C,DC,Z,OV
SUBWFB f,d	d=0	SUBWFB f,d	f-W-C→W	1	Z,C,DC,OV
SUBWFB f,d	d=1	SUBWFB f,d	f-W-C→f	1	Z,C,DC,OV
SUBLW k	------	SUBLW k	I-W→W	1	Z,C,DC,OV
HSUBWF f,d	d=0	PIC 无该指令	W-f→W	1	Z,C,DC,OV
HSUBWF f,d	d=1	PIC 无该指令	W-f-C→W	1	Z,C,DC,OV
HSUBWFB f,d	d=0	PIC 无该指令	W-f- $\overline{C}$ →W	1	Z,C,DC,OV
HSUBWFB f,d	d=1	PIC 无该指令	W-f- $\overline{C}$ →f	1	Z,C,DC,OV
HSUBLW k		PIC 无该指令	W-I→W	1	Z,C,DC,OV
DAW f		PIC 无该指令	将加法运算中放入 W 的值调整为 10 进制数，并将结果存入 f	1	C
无条件转移-5 条					
RETURN		RETURN	从子程序返回	2	NONE
RETLW k		RETLW k	从子程序返回，并将立即数 I 存入 W	2	NONE
RETFIE		RETFIE	从中断返回	2	NONE
CALL k		CALL k	子程序调用	2	NONE
GOTO k		GOTO k	无条件跳转	2	NONE
条件转移-8 条					
BTFSC f,b		BTFSC f,b	如果数据存储器 f 的 b 位为“0”，则跳过下一条指令	1or 2	NONE
BTFSS f,b		BTFSS f,b	如果数据存储器 f 的 b 位为“1”，则跳过下一条指令	1or 2	NONE
TSTFSZW f		TSTFSZW f	数据存储器 f 送至 W,若内容为”0”，则跳过下一条指令	1or 2	NONE
TSTFSZ f		TSTFSZ f	数据存储器 f 内容为”0”，则跳过下一条指令	1or 2	NONE
INCFSZ f,d	d=0	INCFSZ f,d	数据存储器 f 加“1”，结果放入 W,若结果为”0”，则跳过下一条指令	1or 2	NONE
INCFSZ f,d	d=1	INCFSZ f,d	数据存储器 f 加“1”，若结果为”0”，则跳过下一条指令	1or 2	NONE
DECFSZ f,d	d=0	DECFSZ f,d	数据存储器 f 减“1”，结果放入 W,若结果为”0”，则跳过下一条指令	1or 2	NONE
DECFSZ f,d	d=1	DECFSZ f,d	数据存储器 f 减“1”，若结果为”0”，则跳过下一条指令	1or 2	NONE

17. 封装

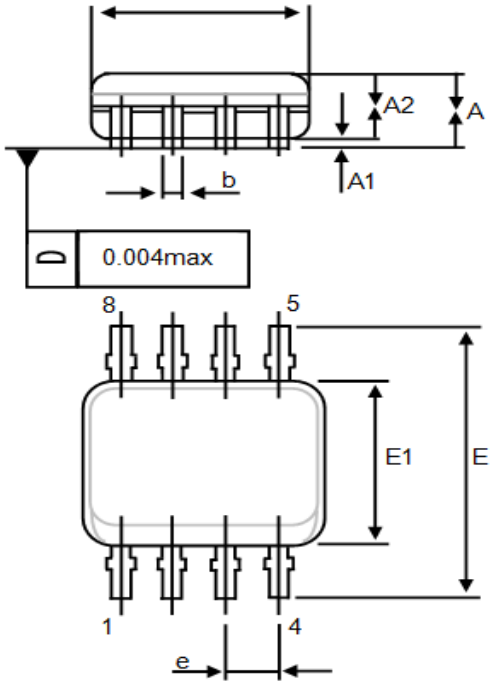
17.1 8pin 封装

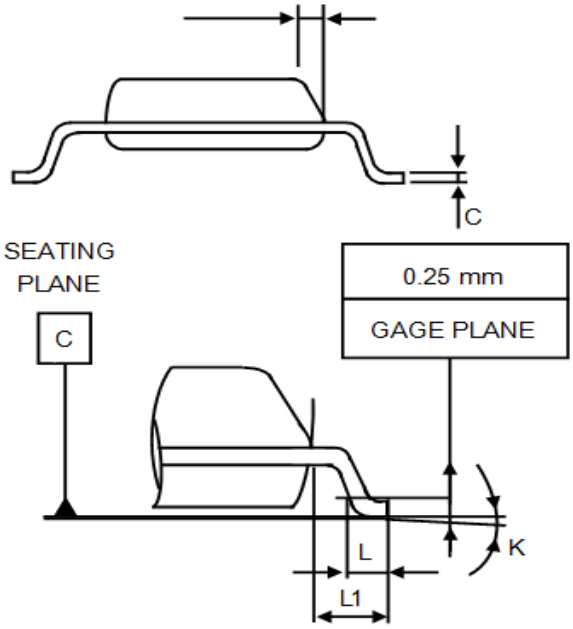
17.1.1 Dip8



Symbol	Dimensions					
	Millmeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A			5.33			0.210
A1	0.38			0.015		
A2	2.92	3.30	4.95	0.115	0.130	0.195
b	0.36	0.46	0.56	0.014	0.018	0.022
b2	1.14	1.52	1.78	0.045	0.060	0.070
c	0.20	0.25	0.36	0.008	0.010	0.014
D	9.02	9.27	10.16	0.355	0.365	0.400
E	7.62	7.87	8.26	0.300	0.310	0.325
E1	6.10	6.35	7.11	0.240	0.250	0.280
e		2.54			0.100	
eA		7.62			0.300	
eB			10.92			0.430
L	2.92	3.30	3.81	0.115	0.130	0.150

17.1.2 Sop8

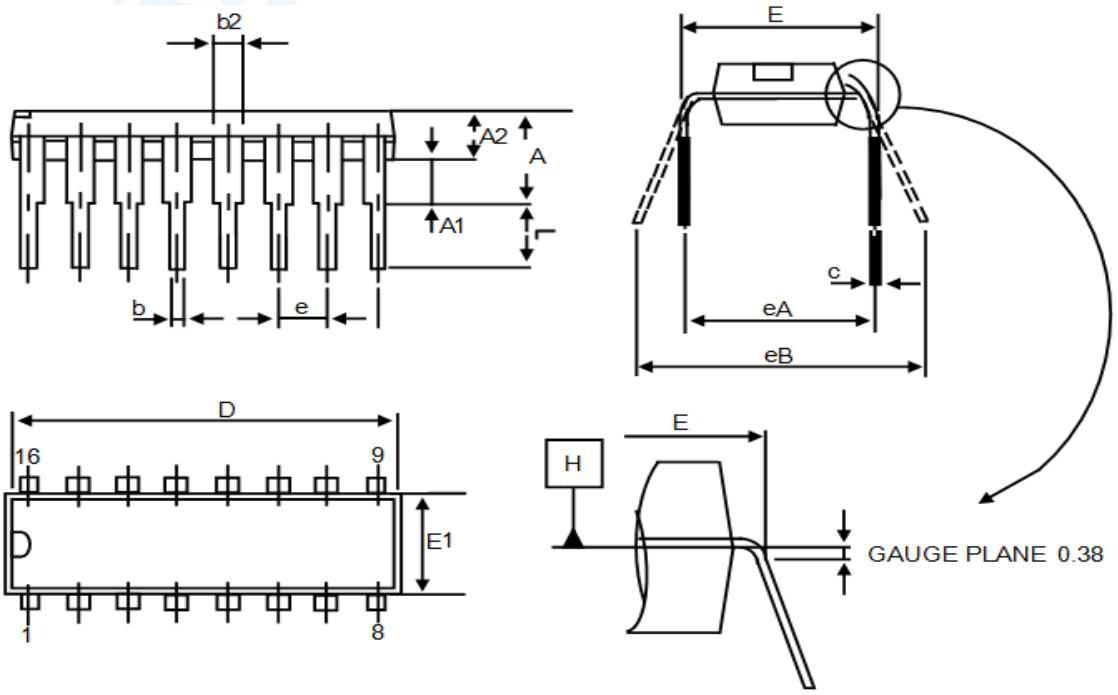




Symbol	Dimensions					
	Millmeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A			1.75			0.069
A1	0.10		0.25	0.004		0.010
A2	1.25			0.049		
b	0.28		0.48	0.011		0.019
c	0.17		0.23	0.007		0.010
D	4.80	4.90	5.00	0.189	0.193	0.197
E	5.80	6.00	6.20	0.228	0.236	0.244
E1	3.80	3.90	4.00	0.150	0.154	0.157
e		1.27			0.050	
h	0.25		0.50	0.010		0.020
L	0.40		1.27	0.016		0.050
L1		1.04			0.040	
K	0		8°	1°		8°

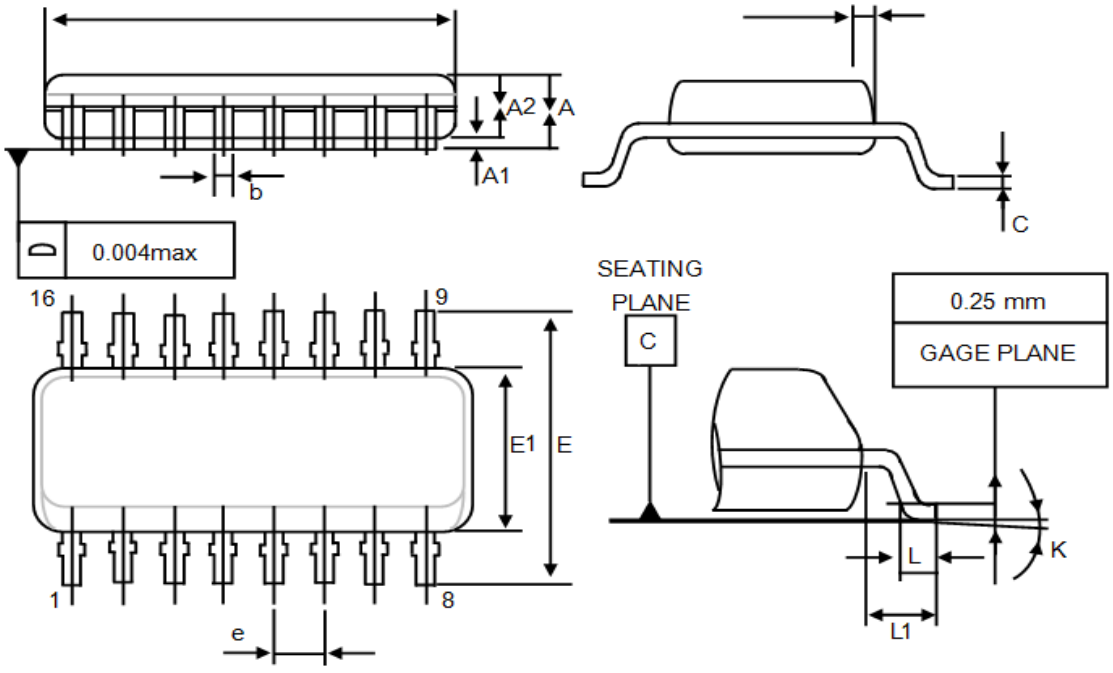
17.2 16pin 封装

17.2.1 Dip 16



Symbol	Dimensions					
	Millmeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A	3.60	3.80	4.00	0.142	0.150	0.157
A1	0.51			0.020		
A2	3.2	3.30	3.4	0.126	0.130	0.134
b	0.44		0.52	0.017		0.020
b2		1.52			0.060	
c	0.25		0.29	0.010		0.011
D	19.00	19.10	19.20	0.748	0.752	0.756
E	7.62	7.87	8.26	0.300	0.310	0.325
E1	6.25	6.35	6.45	0.246	0.25	0.254
e		2.54			0.1	
eA		7.62			0.3	
eB	7.62		9.30	0.3		0.366
L	3.00			0.118		

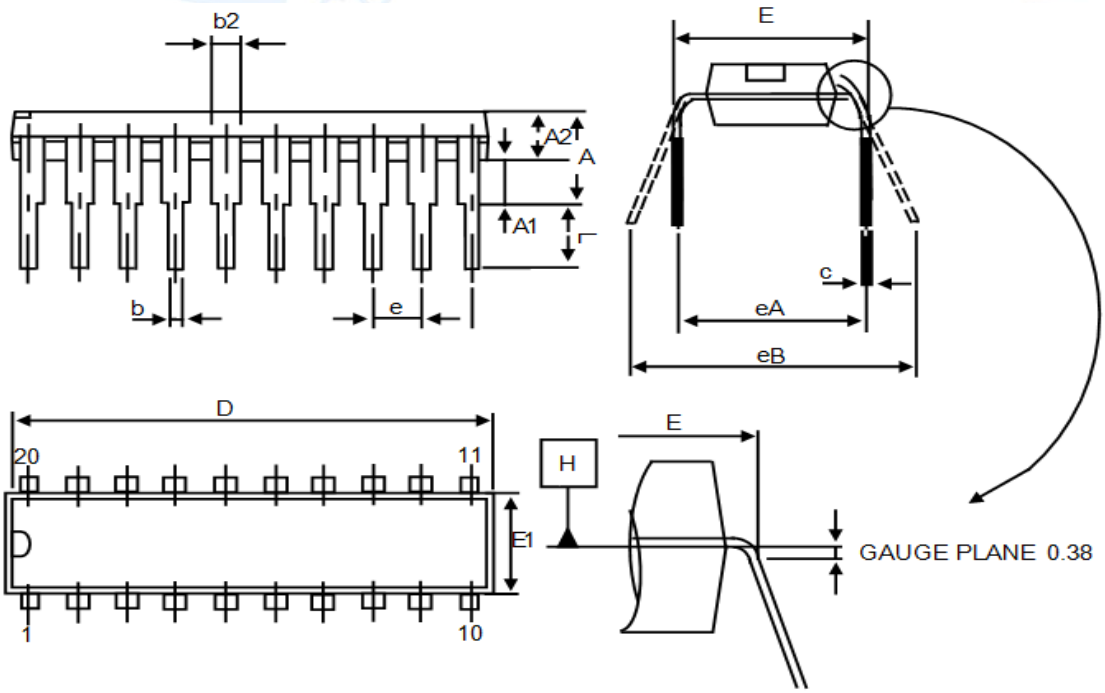
17.2.2 Sop16



Symbol	Dimensions					
	Millmeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A			1.75			0.069
A1	0.10		0.25	0.004		0.010
A2	1.25			0.049		
b	0.28		0.48	0.011		0.019
c	0.17		0.23	0.007		0.010
D	4.80	4.90	5.00	0.189	0.193	0.197
E	5.80	6.00	6.20	0.228	0.236	0.244
E1	3.80	3.90	4.00	0.150	0.154	0.157
e		1.27			0.050	
h	0.25		0.50	0.010		0.020
L	0.40		1.27	0.016		0.050
L1		1.04			0.040	
K	0		8°	1°		8°

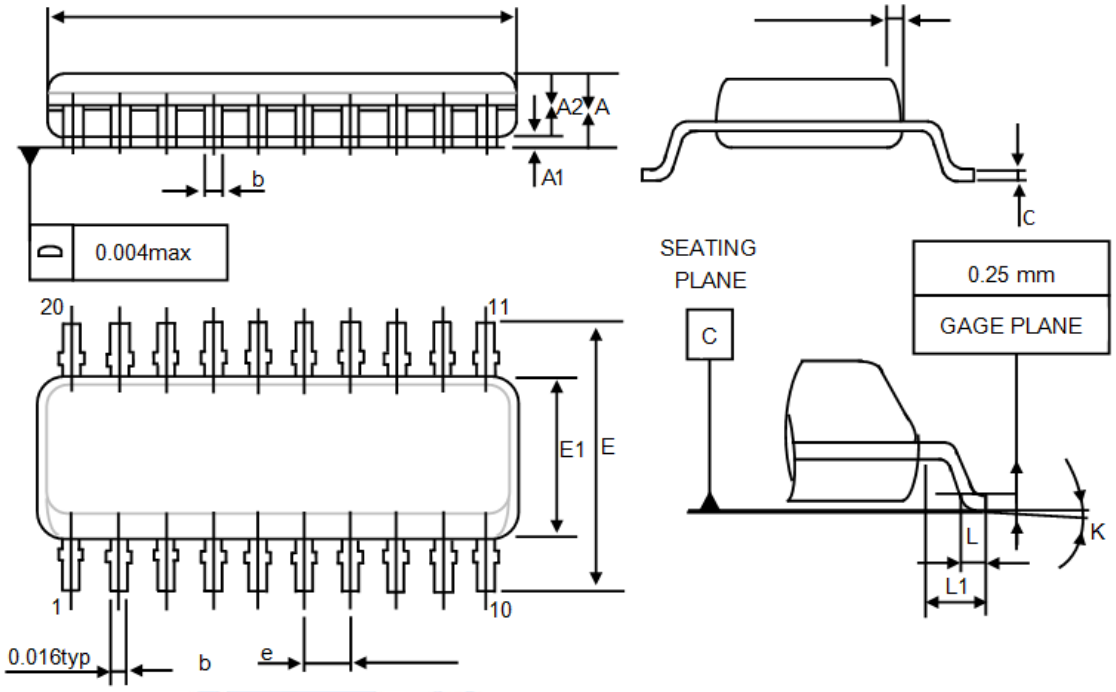
17.3 20pin 封装

17.3.1 Dip 20



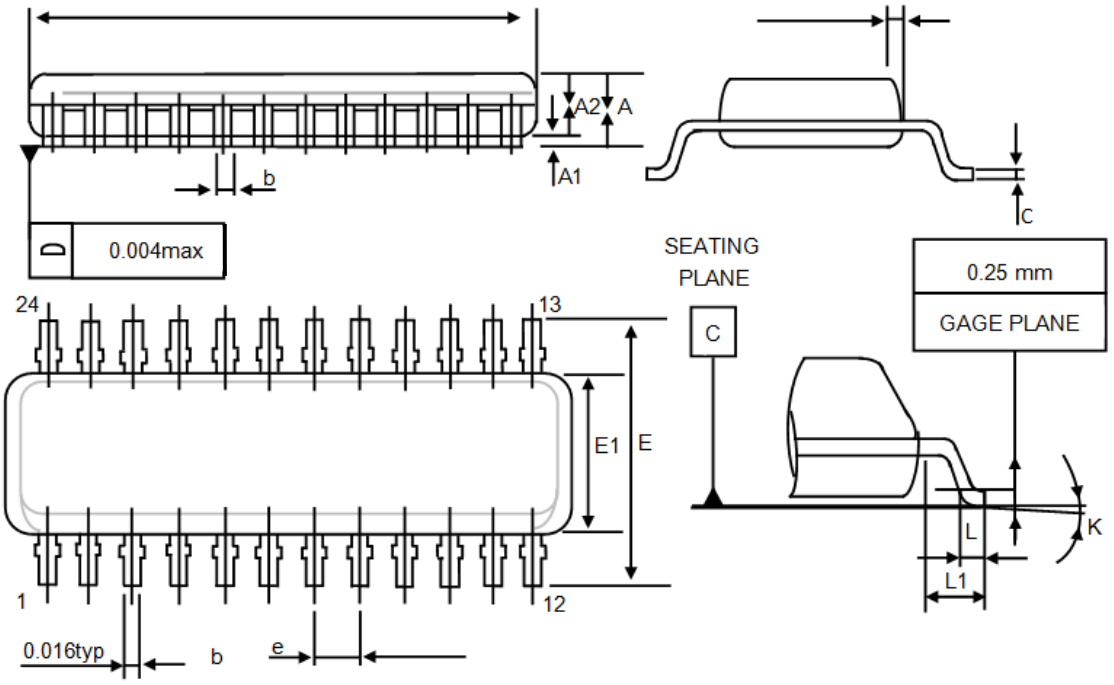
Symbol	Dimensions					
	Millmeters			Lnches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A	3.60	3.80	4.00	0.142	0.150	0.157
A1	0.51			0.020		
A2	3.2	3.30	3.4	0.126	0.130	0.134
b	0.44		0.52	0.017		0.020
b2		1.52			0.060	
c	0.25		0.29	0.010		0.011
D	25.80	25.90	26.00	1.016	1.020	1.024
E	7.62	7.87	8.26	0.300	0.310	0.325
E1	6.45	6.55	6.65	0.254	0.258	0.262
e		2.54			0.100	
eA		7.62			0.300	
eB	7.62		9.30	0.3		0.366
L	3.00			0.118		

17.3.2 Sop 20



Symbol	Dimensions					
	Millmeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A			2.65			0.104
A1	0.10		0.30	0.004		0.012
A2	2.25	2.30	2.35	0.089	0.091	0.093
b	0.35		0.43	0.014		0.017
c	0.25		0.29	0.010		0.011
D	12.70	12.80	12.90	0.5	0.504	0.508
E	10.10	10.30	10.50	0.398	0.406	0.413
E1	7.40	7.50	7.60	0.291	0.295	0.299
e		1.27			0.05	
L	0.70		1.00	0.028		0.039
L1		1.40			0.055	
K	0°		8°	0°		8°

17.4 SSOP24 封装



Symbol	Dimensions					
	Millmeters			Inches		
	Min.	Typ.	Max.	Min.	Typ.	Max.
A			1.75			0.069
A1	0.10	0.15	0.25	0.004	0.06	0.010
A2	1.30	1.40	1.50	0.051	0.055	0.059
b	0.23		0.31	0.009		0.012
c	0.20		0.24	0.008		0.009
D	8.55	8.65	8.75	0.337	0.341	0.344
E	5.80	6.00	6.20	0.228	0.236	0.244
E1	3.80	3.90	4.00	0.150	0.154	0.157
e		0.635			0.025	
h	0.30		0.50	0.012		0.020
L	0.50		0.80	0.020		0.031
L1		1.05			0.041	
K	0°		8°	0°		8°

18. 版本修订说明

版本号	时间	修改内容
V1.0	2016 年 9 月	初始版本