# CMS8M3310用户手册

## 增强型1T 8051 电机微控制器

**Rev. 1.0.1**

# 目录

# 1. 产品概述

## 1.1 系统配置寄存器

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

1. WDT（看门狗选择）
   - ENABLE 打开看门狗定时器
   - DISABLE 关闭看门狗定时器
2. PROTECT（加密）
   - DISABLE FLASH 代码不加密
   - ENABLE FLASH 代码加密，加密后烧写仿真器读出来的值将不确定
3. LVR_SEL（低压侦测电压选择）
   - 2.5V
   - 3.0V
   - 3.5V
4. DEBUG_EN（调试口功能选择）
   - ENABLE DSCK、DSDA 口一直保持为调试口，所有功能均不能使用
   - DISABLE DSCK、DSDA 口为普通功能口
5. WRPR[15:0]（程序存储器写保护位，0x0000H~0x3FFFH，每一位保护 1K 空间）
   - DISABLE 程序区间不受写保护
   - ENABLE 程序区间受写保护，自写 ROM 时无法写入该区间
6. WAIT_NUM（ROM 读取等待）
   - DISABLE 无等待周期
   - ENABLE 一个等待周期
7. USART_SEL（TX/RX）（USART 端口选择）
   - P10/P21 选择 P10 为 TX 口，P21 为 RX 口
   - P14/P13 选择 P14 为 TX 口，P13 为 RX 口
8. PWM0SEL（PWM0 的输出通道选择）
   - P00
   - P01
   - P02
   - P03
   - P04
   - P05
9. PWM1SEL（PWM1 的输出通道选择）
   - P00
   - P01
   - P02
   - P03
   - P04
   - P05

10. PWM2SEL（PWM2 的输出通道选择）
    - P00
    - P01
    - P02
    - P03
    - P04
    - P05
11. PWM3SEL（PWM3 的输出通道选择）
    - P00
    - P01
    - P02
    - P03
    - P04
    - P05
12. PWM4SEL（PWM4 的输出通道选择）
    - P00
    - P01
    - P02
    - P03
    - P04
    - P05
13. PWM5SEL（PWM5 的输出通道选择）
    - P00
    - P01
    - P02
    - P03
    - P04
    - P05

## 1.2 在线串行编程

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

- 电源线
- 接地线
- 数据线
- 时钟线

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

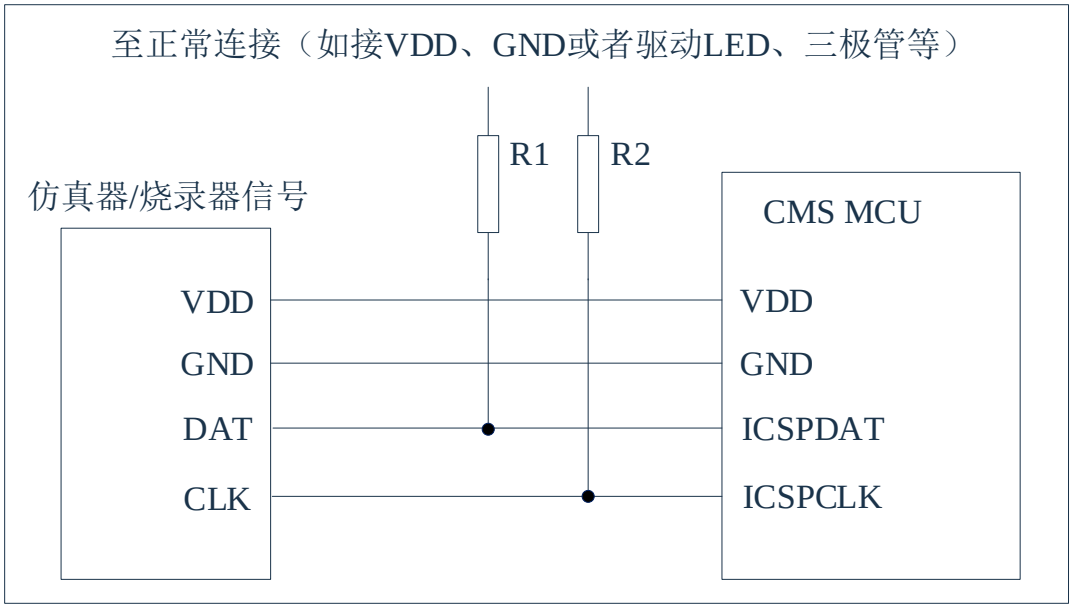



图 1-1：典型的在线串行编程连接方法

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

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

## 1.3 集成开发环境

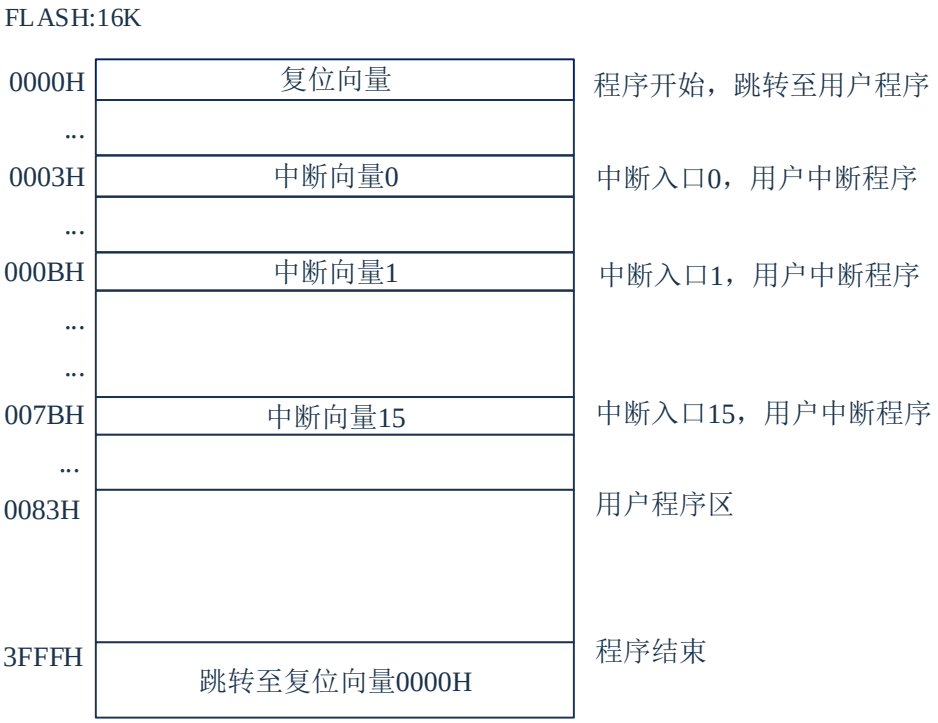
- 芯片支持 2 线调试功能。

# 2. 中央处理器（CPU）

## 2.1 内存

### 2.1.1 程序内存

程序存储器空间：




#### 2.1.1.1 复位向量

单片机具有一个字长的系统复位向量（0000H）。具有以下 3 种复位方式：

- 上电复位
- 看门狗复位
- 低压复位（LVR）
- 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 RSTF 寄存器中的 PD 和 TO 标志位的内容可以判断系统复位方式。

#### 2.1.1.2 中断向量

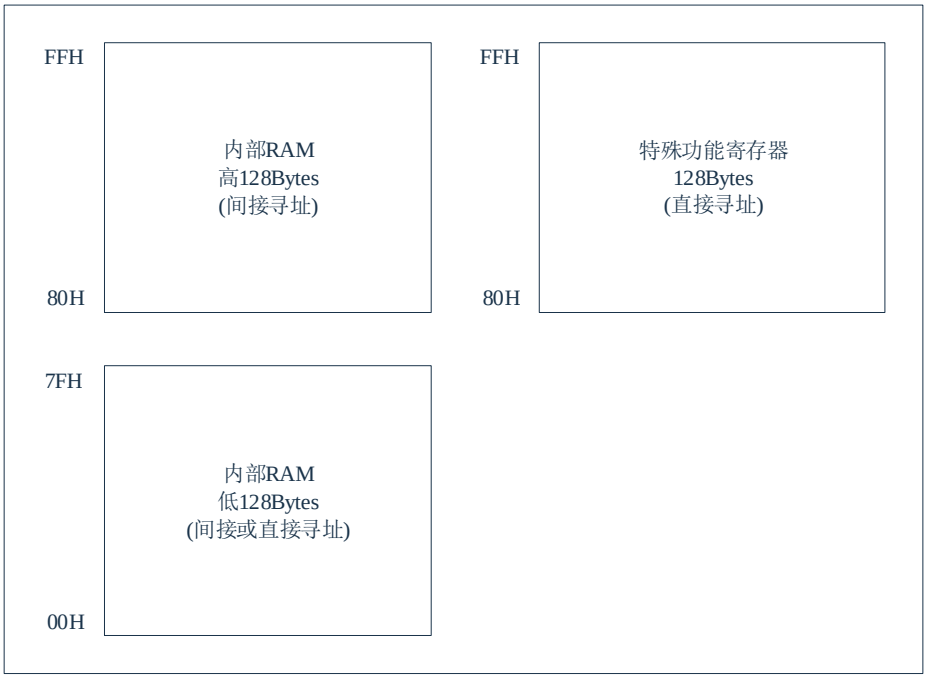
芯片具有 14 个中断源及中断向量：

| 中断源 | 中断描述 | 中断向量 | 同级优先序列 |
|---|---|---|---|
| TMR1 | Timer1 溢出中断 | 0-0x0003 | 1 |
| TMR2 | Timer2 溢出中断 | 1-0x000B | 2 |
| TMR3 | Timer3 溢出中断 | 2-0x0013 | 3 |
| TMR0 | Timer0 溢出中断 | 3-0x001B | 4 |
| TXIF | USART 发送中断 | 4-0x0023 | 5 |
| RCIF | USART 接收中断 | 5-0x002B | 6 |
| ADC | ADC 中断 | 6-0x0033 | 7 |
| P1IF | P1 电平变化中断 | 7-0x003B | 8 |
| ---- | ---- | ---- | ---- |
| PWMZ | PWM 零点中断 | 9-0x004B | 10 |
| PWM | PWM 周期点中断 | 10-0x0053 | 11 |
| ---- | ---- | ---- | ---- |
| EEPROM | 程序 EEPROM 写操作中断 | 12-0x0063 | 13 |
| ACMP0 | 模拟比较器 0 中断 | 13-0x006B | 14 |
| ACMP1 | 模拟比较器 1 中断 | 14-0x0073 | 15 |
| CAP | 捕捉模式中断 | 15-0x007B | 16 |

### 2.1.2 数据存储器

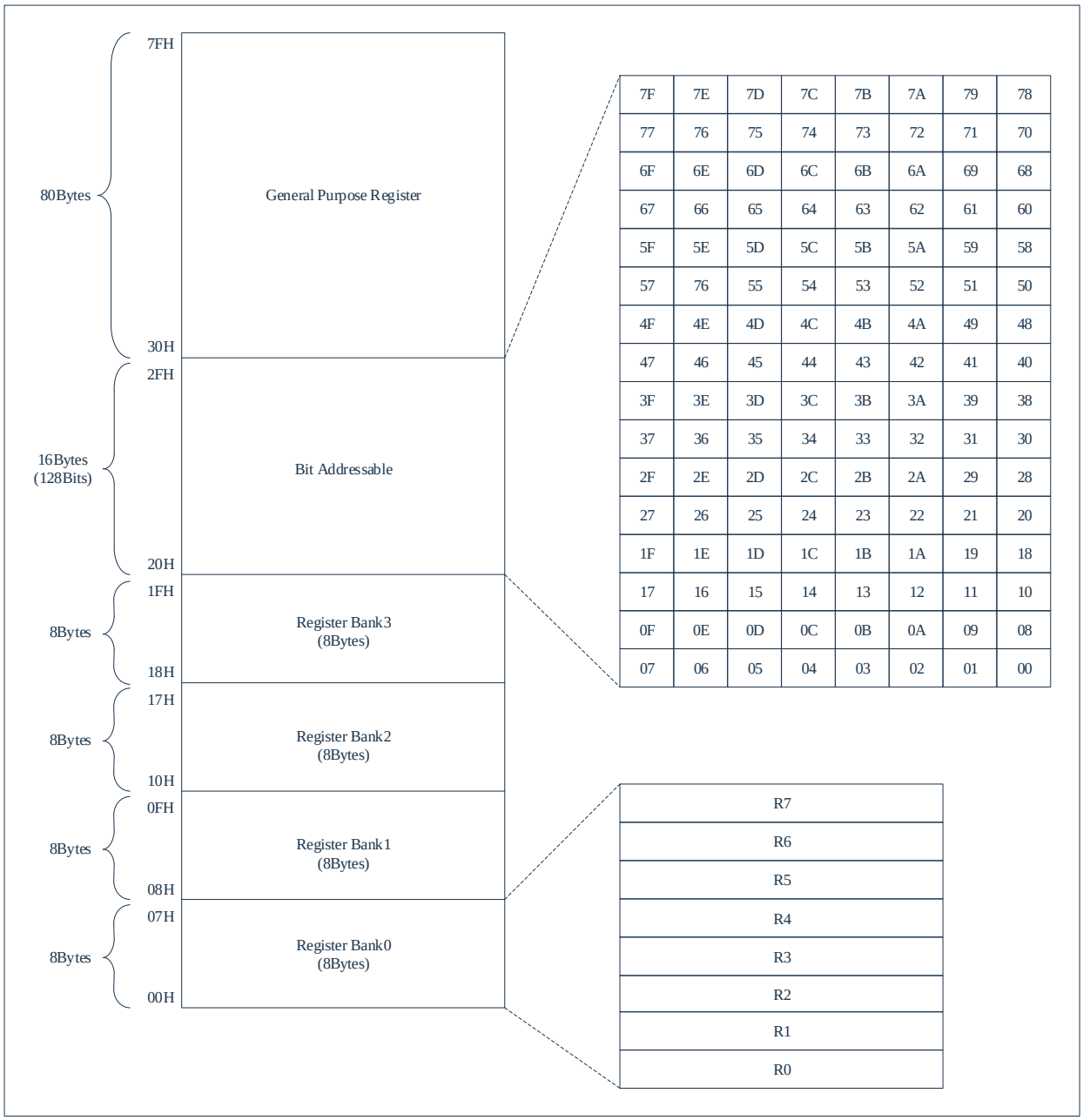
#### 2.1.2.1 通用数据存储器 RAM

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



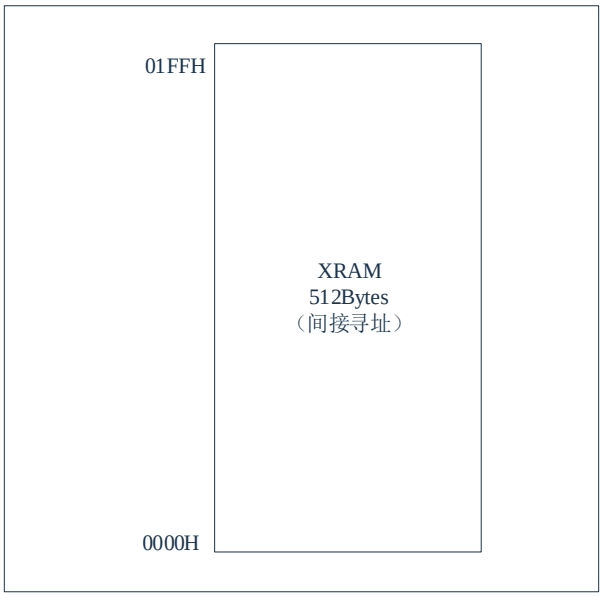

上图所示的高 128Bytes 和 SFR 占用相同的区域（80H~FFH），但它们本身却是独立的。直接寻址高于 7FH 的存储空间（SFR）和间接寻址高于 7FH（高 128Bytes）的存储空间进入到不同的存储空间。

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


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

#### 2.1.2.2 通用外部数据寄存器 XRAM

芯片内部有最大 512Bytes XRAM 区域，该区域与 FLASH/RAM 没有联系，XRAM 空间分配结构框图如下图所示：




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

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

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

#### 2.1.2.3 特殊功能寄存器 SFR

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

| | 0/8 | 1/9 | 2/A | 3/B | 4/C | 5/D | 6/E | 7/F |
|---|---|---|---|---|---|---|---|---|
| 0xF8 | -- | EEDAT | EEDATH | EEADR | EEADRH | EECON1 | EECON2 | EECON3 |
| 0xF0 | B | INTCON | PIE1 | PIE2 | | | | |
| 0xE8 | ACMP0CON0 | ACMP0CON1 | ACMP1CON1 | ACMPFLT | ACMPEVCON | ACMP0VRCON | PGACON0 | |
| 0xE0 | ACC | | | | | | | |
| 0xD8 | ACMP1CON0 | | | | ADRESL | ADRESH | ADCON0 | ADCON1 |
| 0xD0 | PSW | PWMFBKD | BRG_ADJ | RCSTA | TXREG | RCREG | TXSTA | SPBRG |
| 0xC8 | PWMCON0 | PWMCON1 | PWMMASKE | PWMMASKD | PWMDTL | PWMDTR | PWMTL | PWMTH |
| 0xC0 | PIR2 | PWMCON2 | PWMD01L | PWMD01H | PWMD23L | PWMD23H | PWMD45L | PWMD45H |
| 0xB8 | PIR1 | | | | | | | |
| 0xB0 | CAPCON | CAPUL | CAPUH | CAPDL | CAPDH | | | RSTF |
| 0xA8 | -- | | | | | ANSEL0 | ANSEL1 | ANSEL2 |
| 0xA0 | P2 | IREMAP | P0UP | P0RDN | P2RDN | TMR2 | T2CON | PR2 |
| 0x98 | -- | P0TRIS | P1TRIS | P2TRIS | P1INT | P1RDN | P1UP | P2UP |
| 0x90 | P1 | TMR3L | TMR3H | T3CON | | WDTCLR | TA | WDTCON |
| 0x88 | -- | TMR0 | OPTION_REG | TMR1L | TMR1H | T1CON | -- | OSCCON |
| 0x80 | P0 | SP | DPL0 | DPH0 | -- | -- | DPS | PCON |

注：写 TMR0、TMR1L、TMR1H、TMR2、TMR3L、TMR3H、PIR1、PIR2、EEADR、EEADRH、PWMTH、PWMD01H、PWMD23H、PWMD45H 等寄存器的指令后面需要加一条 NOP 指令。

## 2.2 累加器（ACC）

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

ACC 寄存器是一个 8Bit 的寄存器，ALU 的运算结果可以存放在此。

## 2.3 B 寄存器（B）

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

## 2.4 堆栈指针寄存器（SP）

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

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

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

使用 PUSH 指令会将被操作的寄存器的当前值自动保存到 RAM 中。

## 2.5 数据指针寄存器（DPTR）

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

每组指针包括两个 8 位寄存器：DPTR0={DPH,DPL}；

例如操作 XRAM 的汇编代码如下：

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

## 2.6 数据指针选择寄存器（DPS）

数据指针选择寄存器 DPS

| 0x86 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| DPS | ID1 | ID0 | TSL | AU | -- | -- | -- | -- |
| 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 |

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

## 2.7 程序状态寄存器（PSW）

程序状态寄存器 PSW

| 0xD0 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PSW | CY | AC | F0 | RS1 | RS0 | OV | -- | P |
| 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 |

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

.

## 2.8 程序计数器（PC）

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

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

## 2.9 时序存取寄存器（TA）

时序存取寄存器 TA

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

Bit7~Bit0　　TA<7:0>:　时序存取控制位

某些被保护的寄存器必须在对TA进行如下操作之前才能写入

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

中间不能插入其他任何指令，再次修改时需要重新执行此序列

被保护的寄存器:　IREMAP

## 2.10 复位状态寄存器（RSTF）

RSTF 寄存器可以是任何指令的目标寄存器。这些位指示了器件复位状态。

程序状态寄存器 RSTF

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

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

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

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

影响 TO/PD 的事件表

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

复位后 TO/PD 的状态

## 2.11 预分频器（OPTION_REG）

OPTION_REG 寄存器为可读写的寄存器，各个控制位用于配置：

- TIMER0/WDT 预分频器。
- TIMER0。

预分频器 OPTION_REG

| 0x8A | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| OPTION_REG | --- | --- | --- | --- | PSA | PS2 | PS1 | PS0 |
| R/W | --- | --- | --- | --- | R/W | R/W | R/W | R/W |
| 复位值 | --- | --- | --- | --- | 0 | 0 | 1 | 1 |

Bit7~Bit3 未用 未用

Bit3 PSA: 预分频器分配位

0= 预分频器分配给 TIMER0 模块

1= 预分频器分配给 WDT

Bit2~Bit0 PS2~PS0: 预分配参数配置位

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

预分频寄存器实际上是一个 8 位的计数器，用于监视寄存器 WDT 时，是作为一个后分频器。

写 WDTCLR 寄存器将同时对预分频器和 WDT 定时器清零。

## 2.12 看门狗计数器（WDT）

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

### 2.12.1 WDT 周期

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

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

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

注：

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

### 2.12.2 看门狗计数器清除寄存器 WDTCLR

WDT 清除寄存器 WDTCLR

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

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

### 2.12.3 看门狗定时器控制寄存器 WDTCON

看门狗定时器控制寄存器 WDTCON

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

Bit7~Bit1　　未用，读为 0

Bit0　　SWDTEN:　软件使能或禁止看门狗定时器位

1=　使能 WDT

0=　禁止 WDT（复位值）

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

# 3. 系统时钟

## 3.1 概述系统振荡器

芯片只有 1 种振荡方式，内部 RC 振荡。

### 3.1.1 内部 RC 振荡

芯片默认的振荡方式为内部 RC 振荡，其振荡频率 16MHz，可通过 OSCCON 寄存器设置芯片工作频率。

## 3.2 起振时间

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

注：无论芯片是电源上电复位，还是其它原因引起的复位，都会存在这个起振时间。

## 3.3 振荡器控制寄存器

振荡器控制（OSCCON）寄存器控制系统时钟和频率选择。

振荡器控制寄存器 OSCCON

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

Bit7　　未用，读为 0

Bit6~Bit4　　IRCF<2:0>:　内部振荡器分频选择位

111=　$F_{SYS} = F_{HSI}$ /1

110=　$F_{SYS} = F_{HSI}$ /2

101=　$F_{SYS} = F_{HSI}$ /4（默认）

100=　$F_{SYS} = F_{HSI}$ /8

011=　$F_{SYS} = F_{HSI}$ /16

010=　$F_{SYS} = F_{HSI}$ /32

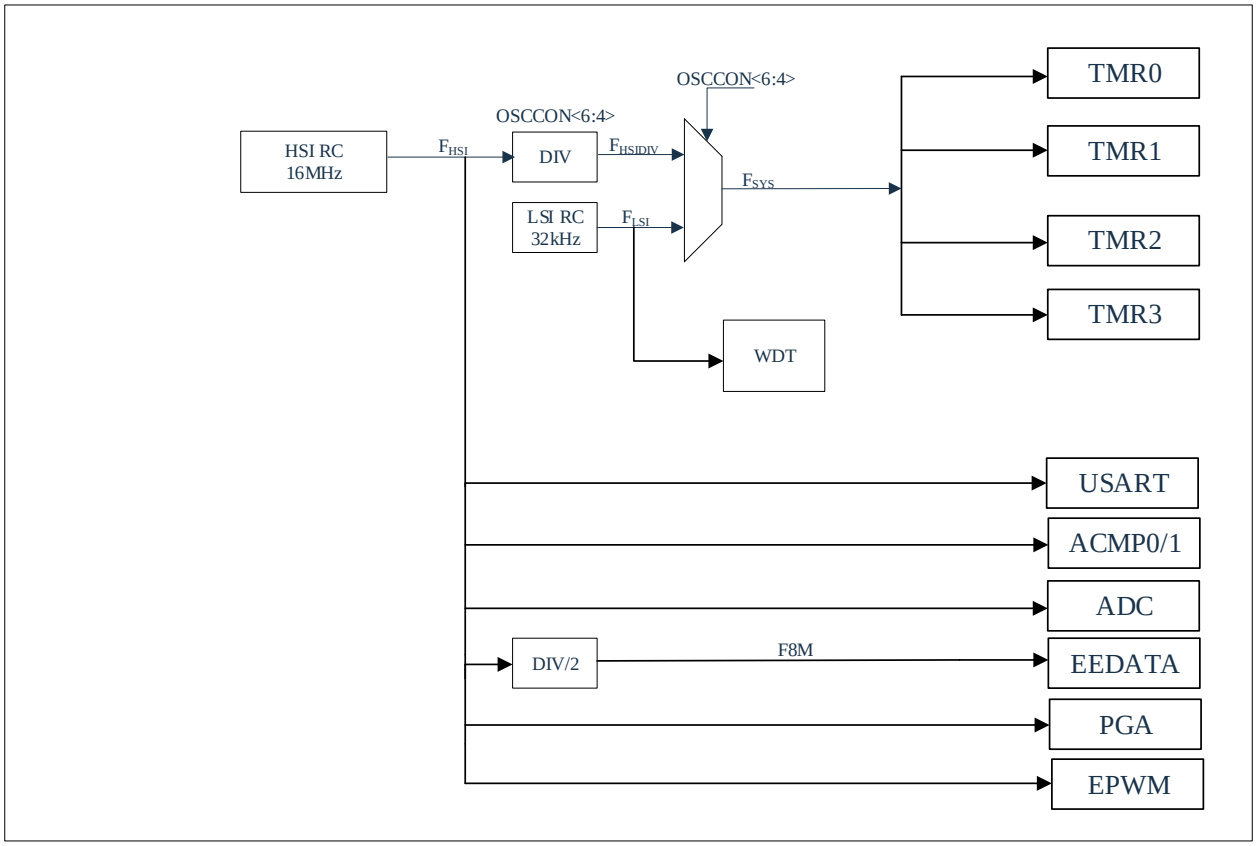
001=　$F_{SYS} = F_{HSI}$ /64

000=　FSYS = 32kHz（LSI）

Bit3~Bit0　　未用，读为0

注：$F_{HSI}$ 为内部振荡器频率 16MHz；$F_{SYS}$ 为系统工作频率。

## 3.4 时钟框图


OSCCON<6:4>
OSCCON<6:4>
HSI RC
16MHz
FHSI
DIV
FHSIDIV
FSYS
LSI RC
32kHz
FLSI
TMR0
TMR1
TMR2
TMR3
WDT
USART
ACMP0/1
ADC
DIV/2
F8M
EEDATA
PGA
EPWM

# 4. 复位

芯片可用如下 4 种复位方式：

- 上电复位；
- 低电压复位；
- 正常工作下的看门狗溢出复位；
- 外部复位；

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

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

## 4.1 上电复位

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

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

## 4.2 掉电复位

### 4.2.1 概述

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

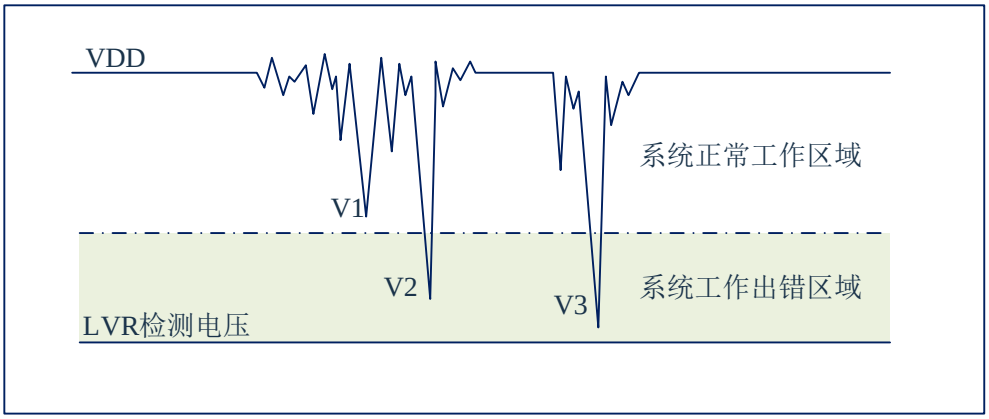



图4-1：掉电复位示意图

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

以下情况系统可能进入死区：

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

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

### 4.2.2 掉电复位的改进办法

如何改进系统掉电复位性能，以下给出几点建议：

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

**看门狗定时器**

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

**降低系统的工作速度**

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

**增大电压下降斜率**

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

## 4.3 看门狗复位

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

看门狗复位的时序如下：

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

关于看门狗定时器的应用问题请参看 2.12WDT 应用章节。

# 5. 低功耗模式

低功耗模式分为 2 类：

- IDLE：空闲模式
- STOP：休眠模式

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

进入空闲模式：IDLE();

进入休眠模式：STOP();

## 5.1 电源管理寄存器 PCON

电源管理寄存器寄存器 PCON

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

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

## 5.2 IDLE 空闲模式

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

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

## 5.3 STOP 休眠模式

在此模式下，系统处于低功耗模式，数字电路均不工作。

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

### 5.3.1 从休眠状态唤醒

可以通过下列任一事件将器件从休眠状态唤醒：

1. 看门狗定时器唤醒（WDT 强制使能）；
2. GPIO 中断或部分外设中断。

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

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

### 5.3.2 休眠模式唤醒时间

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

| 系统主频时钟源 | 系统时钟分频选择(IRCF<2:0>) | 休眠唤醒等待时间 $T_{WAIT}$ |
|---|---|---|
| 内部高速 RC 振荡（$F_{HSI}$） | $F_{SYS}$= $F_{HSI}$ | $T_{WAIT}$=1024*1/ $F_{HSI}$+4*1/ $F_{SYS}$ |
| 内部低速 RC 振荡（$F_{LSI}$） | ---- | $T_{WAIT}$=11/$F_{LSI}$ |

# 6. I/O 端口

芯片有四个 I/O 端口：P0、P1、P2（最多 22 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 |
|---|---|---|---|
| P0 | 0 | 施密特触发输入，推挽式输出，上/下拉输入，AN0，EPWMn | I/O |
| | 1 | 施密特触发输入，推挽式输出，上/下拉输入，AN1，EPWMn | I/O |
| | 2 | 施密特触发输入，推挽式输出，上/下拉输入，AN2，EPWMn | I/O |
| | 3 | 施密特触发输入，推挽式输出，上/下拉输入，AN3，EPWMn | I/O |
| | 4 | 施密特触发输入，推挽式输出，上/下拉输入，AN4，EPWMn | I/O |
| | 5 | 施密特触发输入，推挽式输出，上/下拉输入，AN5，EPWMn | I/O |
| P1 | 0 | 施密特触发输入，推挽式输出，上/下拉输入，AN6，IOINT，TX_A，DSCK | I/O |
| | 1 | 施密特触发输入，推挽式输出，上/下拉输入，AN7，IOINT | I/O |
| | 2 | 施密特触发输入，推挽式输出，上/下拉输入，AN8，IOINT | I/O |
| | 3 | 施密特触发输入，推挽式输出，上/下拉输入，AN9，IOINT，RX_B | I/O |
| | 4 | 施密特触发输入，推挽式输出，上/下拉输入，AN10，IOINT，TX_B | I/O |
| | 5 | 施密特触发输入，推挽式输出，上/下拉输入，AN11，IOINT，PGATO | I/O |
| | 6 | 施密特触发输入，推挽式输出，上/下拉输入，AN12，IOINT， | I/O |
| | 7 | 施密特触发输入，推挽式输出，上/下拉输入，AN13，IOINT，PGAP | I/O |
| P2 | 0 | 施密特触发输入，推挽式输出，上/下拉输入，AN14，PGAGND | I/O |
| | 1 | 施密特触发输入，推挽式输出，上/下拉输入，AN15，RX_A，T1CAP，DSDA | I/O |
| | 2 | 施密特触发输入，推挽式输出，上/下拉输入，AN16，C0_O，C1_O | I/O |
| | 3 | 施密特触发输入，推挽式输出，上/下拉输入，AN17，C1N | I/O |
| | 4 | 施密特触发输入，推挽式输出，上/下拉输入，AN18，C1P0 | I/O |
| | 5 | 施密特触发输入，推挽式输出，上/下拉输入，AN19，C1P1 | I/O |
| | 6 | 施密特触发输入，推挽式输出，上/下拉输入，AN20，C1P2 | I/O |
| | 7 | 施密特触发输入，推挽式输出，上/下拉输入，AN21，C0P0 | I/O |

＜表 6-1：端口配置总概＞

## 6.1 I/O 口结构图

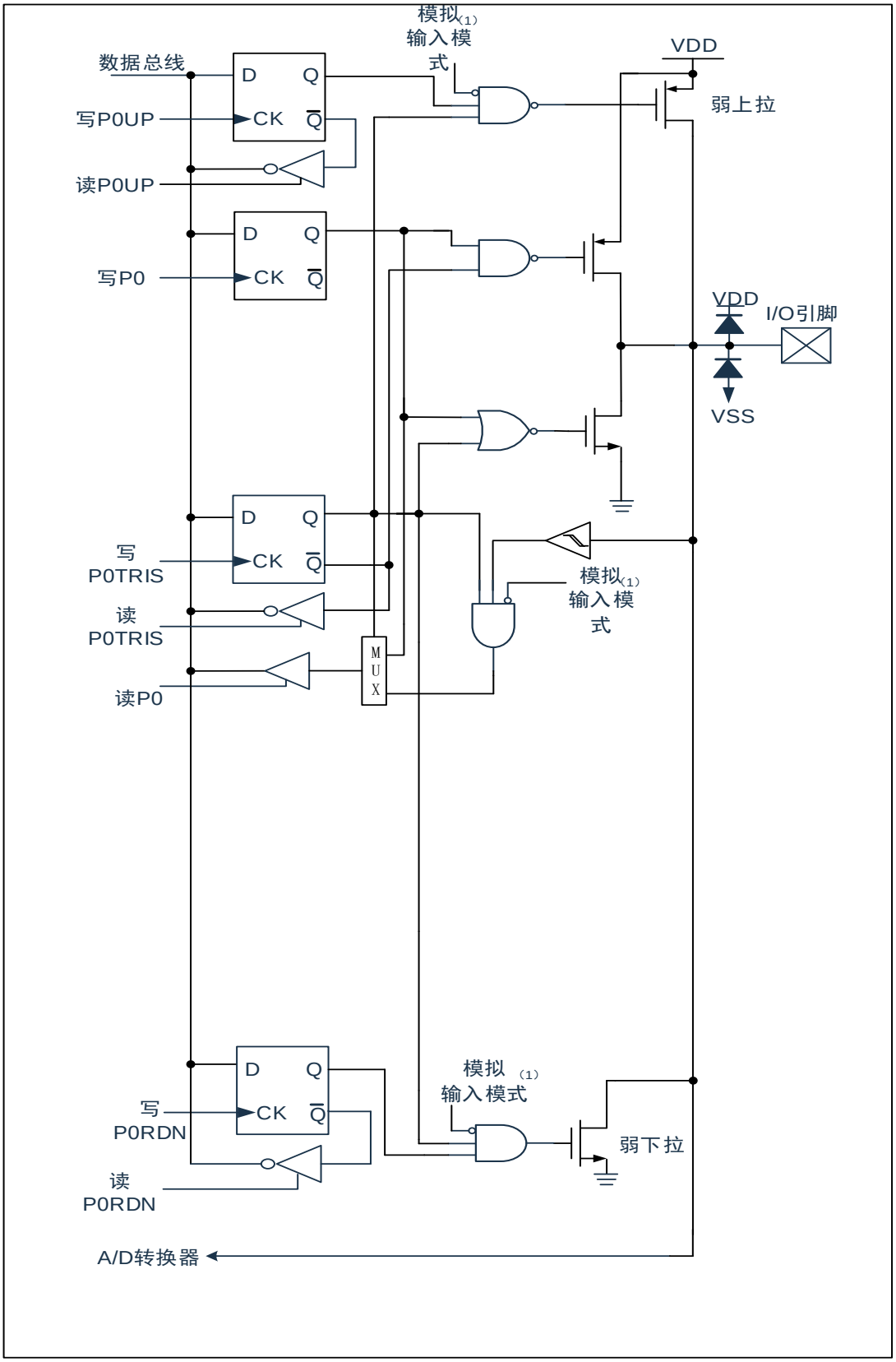



图 6-1：I/O 口结构图

## 6.2 P0

### 6.2.1 P0 数据及方向

P0 是 5Bit 宽的双向端口。它所对应的数据方向寄存器是 P0TRIS。将 P0TRIS 的一个位置 0（=0）可以将相应的引脚配置为输入。将 P0TRIS 的一个位置 1（=1）可将相应的 P0 引脚配置为输出。

读 P0 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 P0 引脚用作模拟输入时，P0TRIS 寄存器仍然控制 P0 引脚的方向。当将 P0 引脚用作模拟输入时，用户必须确保 P0TRIS 寄存器中的位保持为置 1 状态。

与 P0 口相关寄存器有 P0、P0TRIS、P0UP、P0RDN、ANSEL0 等。

P0 数据寄存器 P0

| 0x80 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| P0 | --- | --- | P05 | P04 | P03 | P02 | P01 | P00 |
| 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　　P0<5:0>:　P0I/O 引脚位

当P0TRISx=1时

1=　端口输出高电平

0=　端口输出低电平

当P0TRISx=0时

1=　端口引脚电平>$V_{IH}$

0=　端口引脚电平<$V_{IL}$

P0 方向寄存器 P0TRIS

| 0x99 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| P0TRIS | --- | --- | P0TRIS5 | P0TRIS4 | P0TRIS3 | P0TRIS2 | P0TRIS1 | P0TRIS0 |
| 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　　P0TRIS<5:0>:　P0 三态控制位

1=　P0 引脚被配置为输出

0=　P0 引脚被配置为输入（三态）

### 6.2.2 P0 上拉电阻

每个 P0 引脚都有可单独配置的内部弱上拉。控制位 P0UP<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。

P0 上拉电阻寄存器 P0UP

| 0xA2 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| P0UP | --- | --- | P0UP5 | P0UP4 | P0UP3 | P0UP2 | P0UP1 | P0UP0 |
| 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　　P0UP<5:0>:　弱上拉寄存器位

1=　使能上拉

0=　禁止上拉

注：如果引脚被配置为输出或者模拟输入，将自动禁止弱上拉。

### 6.2.3 P0 下拉电阻

每个 P0 引脚都有可单独配置的内部弱下拉。控制位 P0RDN<7:0>使能或禁止每个弱下拉。当将端口引脚配置为输出时，其弱下拉会自动切断。

P0 下拉电阻寄存器 P0RDN

| 0xA3 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| P0RDN | --- | --- | P0RDN5 | P0RDN4 | P0RDN3 | P0RDN2 | P0RDN1 | P0RDN0 |
| 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　　P0RDN<5:0>:　弱下拉寄存器位

1=　使能下拉

0=　禁止下拉

注：如果引脚被配置为输出或者模拟输入，将自动禁止弱下拉。

### 6.2.4 P0 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 置 1 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。

P0 模拟选择寄存器 ANSEL0

| 0xAD | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ANSEL0 | --- | --- | ANS5 | ANS4 | ANS3 | ANS2 | ANS1 | ANS0 |
| 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<5:0>　模拟选择位,分别选择引脚 AN<5:0>的模拟或数字功能

1=　模拟输入，引脚被分配为模拟输入

0=　数字 I/O，引脚被分配给端口或特殊功能

# 6.3 P1

## 6.3.1 P1 数据及方向

P1 是一个 8Bit 宽的双向端口。对应的数据方向寄存器为 P1TRIS。将 P1TRIS 中的某个位置 0（=0）可以使对应的 P1 引脚作为输入引脚。将 P1TRIS 中的某个位置 1（=1）将使对应的 P1 引脚作为输出引脚。

P1TRIS 置 0 时，读 P1 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 P1 引脚用作模拟输入时，P1TRIS 寄存器仍然控制 P1 引脚的方向。当将 P1 引脚用作模拟输入时，用户必须确保 P1TRIS 寄存器中的位保持为置 0 状态。

与 P1 口相关寄存器有 P1、P1TRIS、P1UP、P1RDN、P1INT、ANSEL1 等。

P1 数据寄存器 P1

| 0x90 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| P1 | P17 | P16 | P15 | P14 | P13 | P12 | P11 | P10 |
| 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~Bit3　　P1<7:0>:　P1 I/O 引脚位

当P1TRISx=1时

1=　端口输出高电平

0=　端口输出低电平

当P1TRISx=0时

1=　端口引脚电平>$V_{IH}$

0=　端口引脚电平<$V_{IL}$

P1 方向寄存器 P1TRIS

| 0x9A | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| P1TRIS | P1TRIS7 | P1TRIS6 | P1TRIS5 | P1TRIS4 | P1TRIS3 | P1TRIS2 | P1TRIS1 | P1TRIS0 |
| 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　　P1TRIS<7:0>:　P1 三态控制位

1=　P1引脚被配置为输出

0=　P1引脚被配置为输入（三态）

### 6.3.2 P1 上拉电阻

每个 P1 引脚都有可单独配置的内部弱上拉。控制位 P1UP<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。

P1 上拉电阻寄存器 P1UP

| 0x9E | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| P1UP | P1UP7 | P1UP6 | P1UP5 | P1UP4 | P1UP3 | P1UP2 | P1UP1 | P1UP0 |
| 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　　P1UP<7:0>:　弱上拉寄存器位
1=　使能上拉
0=　禁止上拉

注：如果引脚被配置为输出或者模拟输入，将自动禁止弱上拉。

### 6.3.3 P1 下拉电阻

每个 P1 引脚都有可单独配置的内部弱下拉。控制位 P1RDN<7:0>使能或禁止每个弱下拉。当将端口引脚配置为输出时，其弱下拉会自动切断。

P1 下拉电阻寄存器 P1RDN

| 0x9D | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| P1RDN | P1RDN7 | P1RDN6 | P1RDN5 | P1RDN4 | P1RDN3 | P1RDN2 | P1RDN1 | P1RDN0 |
| 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　　P1RDN<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者模拟输入，将自动禁止弱下拉。

### 6.3.4 P1 电平变化中断

所有的 P1 引脚都可以被单独配置为电平变化中断引脚。控制位 P1INT<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 P1 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 PIE1 寄存器中的 P1 电平变化中断标志位（P1IF）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 P1 进行读操作。这将结束引脚电平的不匹配状态。
- 将标志位 P1IF 清零。

不匹配状态会不断将 P1IF 标志位置 1。而读 P1 将结束不匹配状态，并且允许将 P1IF 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，P1IF 标志位将继续置 1。

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

P1 电平变化中断寄存器 P1INT

| 0x9C | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| P1INT | P1INT7 | P1INT6 | P1INT5 | P1INT4 | P1INT3 | P1INT2 | P1INT1 | P1INT0 |
| 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 P1INT<7:0> P1 的电平变化中断控制位
1= 允许电平变化中断
0= 禁止电平变化中断

### 6.3.5 P1 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 置 1 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。

P1 模拟选择寄存器 ANSEL1

| 0xAE | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ANSEL1 | ANS13 | ANS12 | ANS11 | ANS10 | ANS9 | ANS8 | ANS7 | ANS6 |
| 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<13:6> 模拟选择位,分别选择引脚 AN<13:6>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 P2

## 6.4.1 P2 数据及方向

P2 是一个 8Bit 宽的双向端口。对应的数据方向寄存器为 P2TRIS。将 P2TRIS 中的某个位置 0（=0）可以使对应的 P2 引脚作为输入引脚。将 P2TRIS 中的某个位置 1（=1）将使对应的 P2 引脚作为输出引脚。

当 P2TRIS 置 0 时，读 P2 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 P2 口相关寄存器有 P2、P2TRIS、P2UP、P2RDN、ANSEL2 等。

P2 数据寄存器 P2

| 0xA0 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| P2 | P27 | P26 | P25 | P24 | P23 | P22 | P21 | P20 |
| 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 P2<7:0> P2 I/O 引脚位

当P2TRISx=1时

1= 端口输出高电平

0= 端口输出低电平

当P2TRISx=0时

1= 端口引脚电平>$V_{IH}$

0= 端口引脚电平<$V_{IL}$

P2 方向寄存器 P2TRIS

| 0x9B | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| P2TRIS | P2TRIS7 | P2TRIS6 | P2TRIS5 | P2TRIS4 | P2TRIS3 | P2TRIS2 | P2TRIS1 | P2TRIS0 |
| 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 P2TRIS<7:0>: P2 三态控制位

1= P2 引脚被配置为输出

0= P2 引脚被配置为输入（三态）

### 6.4.2 P2 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 置 1 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。

P2 模拟选择寄存器 ANSEL2

| 0xAF | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ANSEL2 | ANS21 | ANS20 | ANS19 | ANS18 | ANS17 | ANS16 | ANS15 | ANS14 |
| 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<21:14>:　模拟选择位，分别选择引脚 AN<21:14>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 P2 上拉电阻

每个 P2 引脚都有可单独配置的内部弱上拉。控制位 P2UP<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。

P2 上拉电阻寄存器 P2UP

| 0x9F | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| P2UP | P2UP7 | P2UP6 | P2UP5 | P2UP4 | P2UP3 | P2UP2 | P2UP1 | P2UP0 |
| 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　　P2UP<7:0>:　弱上拉寄存器位
1=　使能上拉
0=　禁止上拉

注：如果引脚被配置为输出或者模拟输入，将自动禁止弱上拉。

### 6.4.4 P2 下拉电阻

每个 P2 引脚都有可单独配置的内部弱下拉。控制位 P2RDN<7:0>使能或禁止每个弱下拉。当将端口引脚配置为输出时，其弱下拉会自动切断。

P2 下拉电阻寄存器 P2RDN

| 0xA4 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| P2RDN | P2RDN7 | P2RDN6 | P2RDN5 | P2RDN4 | P2RDN3 | P2RDN2 | P2RDN1 | P2RDN0 |
| 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 P2RDN<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者模拟输入，将自动禁止弱下拉。

## 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：

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“$V_{DD}$+0.3V”与“GND-0.3V”之间。若输入口电压不在此范围内可采用如下图所示方法。

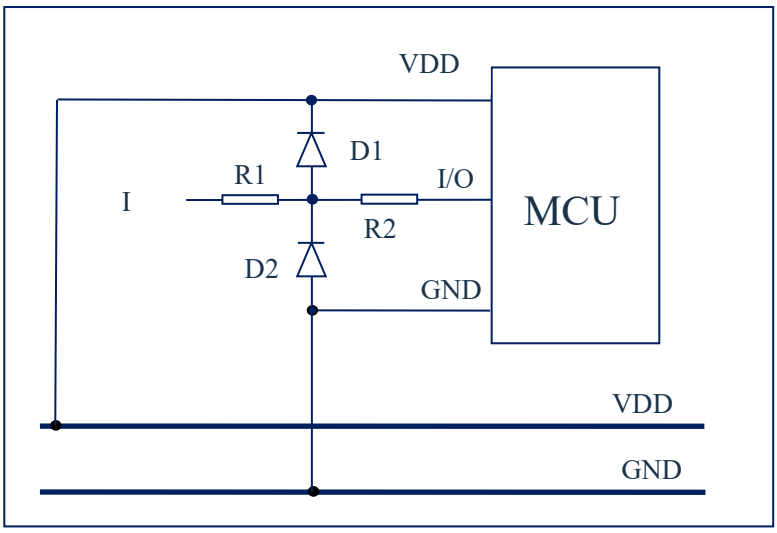



图 6-3：输入电压不在规定范围内采用电路

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

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。

# 7. 中断

## 7.1 中断概述

芯片具有 14 个中断源及中断向量：

| 中断源 | 中断描述 | 中断向量 | 同级优先序列 |
|---|---|---|---|
| TMR1 | Timer1 溢出中断 | 0-0x0003 | 1 |
| TMR2 | Timer2 溢出中断 | 1-0x000B | 2 |
| TMR3 | Timer3 溢出中断 | 2-0x0013 | 3 |
| TMR0 | Timer0 溢出中断 | 3-0x001B | 4 |
| TXIF | USART 发送中断 | 4-0x0023 | 5 |
| RCIF | USART 接收中断 | 5-0x002B | 6 |
| ADC | ADC 中断 | 6-0x0033 | 7 |
| P1IF | P1 电平变化中断 | 7-0x003B | 8 |
| ---- | ---- | ---- | ---- |
| PWMZ | PWM 零点中断 | 9-0x004B | 10 |
| PWM | PWM 周期点中断 | 10-0x0053 | 11 |
| ---- | ---- | ---- | ---- |
| EEPROM | 程序 EEPROM 写操作中断 | 12-0x0063 | 13 |
| ACMP0 | 模拟比较器 0 中断 | 13-0x006B | 14 |
| ACMP1 | 模拟比较器 1 中断 | 14-0x0073 | 15 |
| CAP | 捕捉模式中断 | 15-0x007B | 16 |

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

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

## 7.2 中断重映射

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

IREMAP 寄存器为设置为 0x01 时，0X01<<9 =0X0200，即中断偏移基地址为 0X0200，则其他所有中断地址需加上偏移基地址才是实际的中断向量地址。比如 INT0 的向量地址重映射之后应为：0x0200+0x0003=0x0203

由上述描述可知，中断向量偏移基地址最小的偏移单元为 0.5KB。

中断重映射使用实例：

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

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

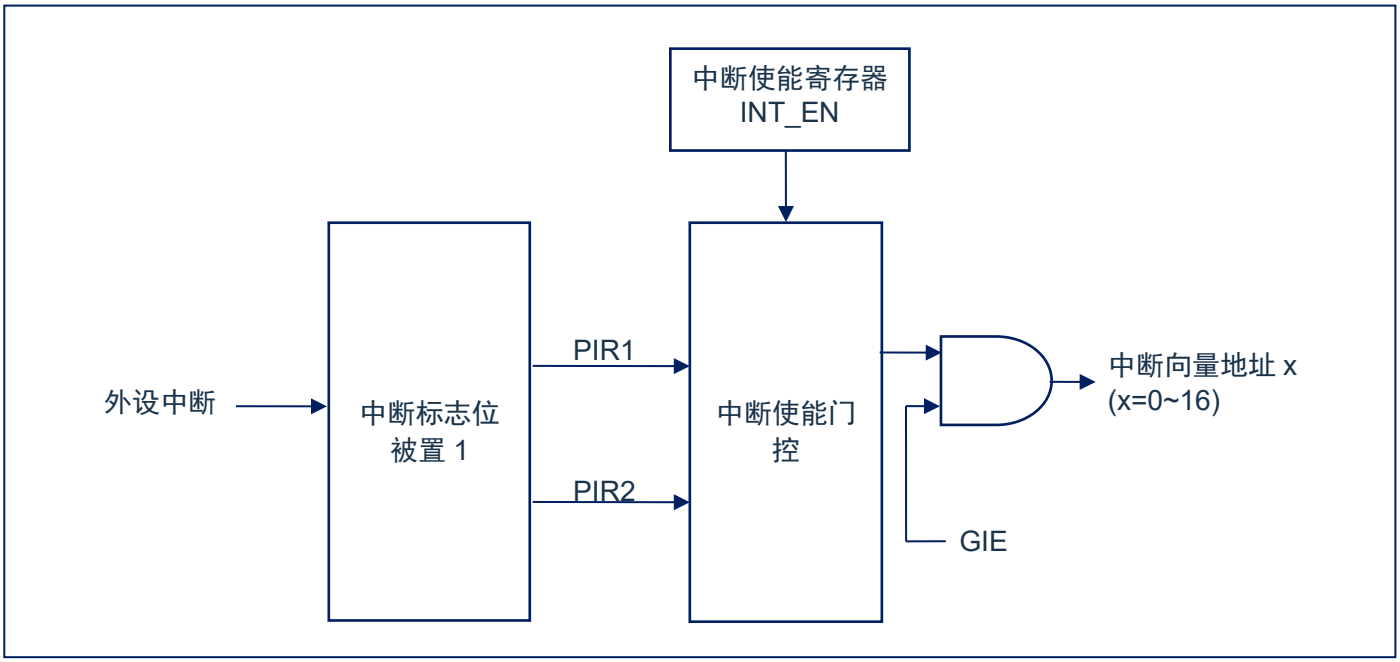



图 7-1：中断原理示意图

## 7.3 中断控制寄存器

### 7.3.1 中断控制寄存器

中断控制寄存器 INTCON 是可读写的寄存器，包含两个中断优先级控制；

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

中断控制寄存器 INTCON

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

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

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

Bit5 未用 未用

Bit4~Bit0 INTPS[4:0] 高优先级中断允许位
0000= PIR1[0]为高优先级中断
0001= PIR1[1]为高优先级中断
0010= PIR1[2]为高优先级中断
0011= PIR1[3]为高优先级中断
0100= PIR1[4]为高优先级中断
0101= PIR1[5]为高优先级中断
0110= PIR1[6]为高优先级中断
0111= PIR1[7]为高优先级中断
1000= 未用
1001= PIR2[1]为高优先级中断
1010= PIR2[2]为高优先级中断
1011= 未用
1100= PIR2[4]为高优先级中断
1101= PIR2[5]为高优先级中断
1110= PIR2[6]为高优先级中断
1111= PIR2[7]为高优先级中断

### 7.3.2 外设中断允许寄存器

外设中断允许寄存器有 PIE1 和 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

| 0XF2 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PIE1 | P1IE | ADIE | RXIE | TXIE | TMR0IE | TMR3IE | TMR2IE | TMR1IE |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 复位值 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit7 P1IE: P1电平变化中断允许位
允许P1电平变化中断
禁止P1电平变化中断

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

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

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

Bit3 TMR0IE: TIMER0溢出中断允许位
1= 允许TIMER0溢出中断
0= 禁止TIMER0溢出中断

Bit2 TMR3IE: TIMER3溢出中断允许位
1= 允许TIMER3溢出点中断
0= 禁止TIMER3溢出点中断

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

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

外设中断允许寄存器 PIE2

| 0xF3 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PIE2 | CAPIE | ACMP1IE | ACMP0IE | EEIE | ---- | PWMIE | PWMZIE | ---- |
| R/W | R/W | R/W | R/W | R/W | ---- | R/W | R/W | ---- |
| 复位值 | 0 | 0 | 0 | 0 | ---- | 0 | 0 | ---- |

Bit7 CAPIE: 捕捉中断允许位
1= 允许捕捉中断
0= 禁止捕捉中断

Bit6 ACMP1IE: 模拟比较器1中断允许位
1= 允许模拟比较器1产生中断
0= 禁止模拟比较器1产生中断

Bit5 ACMP0IE: 模拟比较器0中断允许位
1= 允许模拟比较器0产生中断
0= 禁止模拟比较器0产生中断

Bit4 EEIE: 程序EEPROM写操作中断允许位
1= 允许程序EEPROM写操作中断
0= 禁止程序EEPROM写操作中断

Bit3 未用

Bit2 PWMIE: PWM周期点中断允许位
1= 允许PWM周期点中断
0= 禁止PWM周期点中断

Bit1 PWMZIE: PWM零点中断允许位
1= 允许PWM零点中断
0= 禁止PWM零点中断

Bit0 未用

### 7.3.3 外设中断请求寄存器

外设中断请求寄存器为 PIR1 和 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

| 0xB8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PIR1 | P1IF | ADIF | RXIF | TXIF | TMR0IF | TMR3IF | TMR2IF | TMR1IF |
| R/W | R/W | R/W | R | R | R/W | R/W | R/W | R/W |
| 复位值 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit7 P1IF: P1电平变化中断标志位
1= P1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P1通用I/O引脚的状态发生了改变

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

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

Bit3 TMR0IF: TIMER0溢出中断标志位
1= TMR0寄存器溢出（必须由软件清零）
0= TMR0寄存器未溢出

Bit2 TMR3IF: TIMER3溢出中断标志位
1= 已产生TIMER3溢出中断（必须由软件清零）
0= 未产生TIMER3溢出中断

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

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

注：T0IF 位在 TMR0 计满归 0 时置 1。复位该标志位不会使 TMR0 计数值发生改变，可在将 T0IF 位清零前写 TMR0 寄存器，对计数初始值进行重载。

外设中断请求寄存器 PIR2

| 0xC0 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PIR2 | CAPIF | ACMP1IF | ACMP0IF | EEIF | -- | PWMIF | PWMZIF | -- |
| 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 CAPIF 捕捉中断标志位
1= 已产生中断
0= 未产生中断
Bit6 ACMP1IF 模拟比较器1中断标志位
1= 模拟比较器1已产生中断
0= 模拟比较器1未产生中断
Bit5 ACMP0IF 模拟比较器0中断标志位
1= 模拟比较器0已产生中断
0= 模拟比较器0未产生中断
Bit4 EEIF: 程序EEPROM写操作中断标志位
1= 写操作完成（必须由软件清零）
0= 写操作未完成或尚未启动
Bit3 未用
Bit2 PWMIF PWM周期点中断标志位
1= 已产生PWM周期点中断
0= 未产生PWM周期点中断
Bit1 PWMZIF PWM零点中断标志位
1= 已产生PWM零点中断
0= 未产生PWM零点中断
Bit0 未用

### 7.3.4 中断偏移基地址寄存器 IREMAP

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

注：IREMAP为受保护寄存器。

Bit7-4 须写0 保留位

Bit3-0 MAP<3:0> 中断偏移地址

0X00= 中断偏移地址为0X0000

0X01= 中断偏移地址为(0X01<<9)，0X0200

0X02= 中断偏移地址为(0X02<<9)，0X0400

0X03= 中断偏移地址为(0X03<<9)，0X0600

0X04= 中断偏移地址为(0X04<<9)，0X0800

0X05= 中断偏移地址为(0X05<<9)，0X0A00

0X06= 中断偏移地址为(0X06<<9)，0X0C00

0X07= 中断偏移地址为(0X07<<9)，0X0E00

0X08= 中断偏移地址为(0X08<<9)，0X1000

0X09= 中断偏移地址为(0X09<<9)，0X1200

0X0a= 中断偏移地址为(0X0a<<9)，0X1400

0X0b= 中断偏移地址为(0X0b<<9)，0X1600

0X0c= 中断偏移地址为(0X0c<<9)，0X1800

0X0d= 中断偏移地址为(0X0d<<9)，0X1A00

0X0e= 中断偏移地址为(0X0e<<9)，0X1C00

0X0f= 中断偏移地址为(0X0f<<9)，0X1E00

修改 IREMAP 需要的指令序列（中间不能插入其他任何指令）：

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

# 8. 定时计数器 TIMER0

## 8.1 定时计数器 TIMER0 概述

TIMER0 由如下功能组成：

- 8 位定时器/计数器寄存器（TMR0）；
- 8 位预分频器（与看门狗定时器共用）；
- 溢出中断。

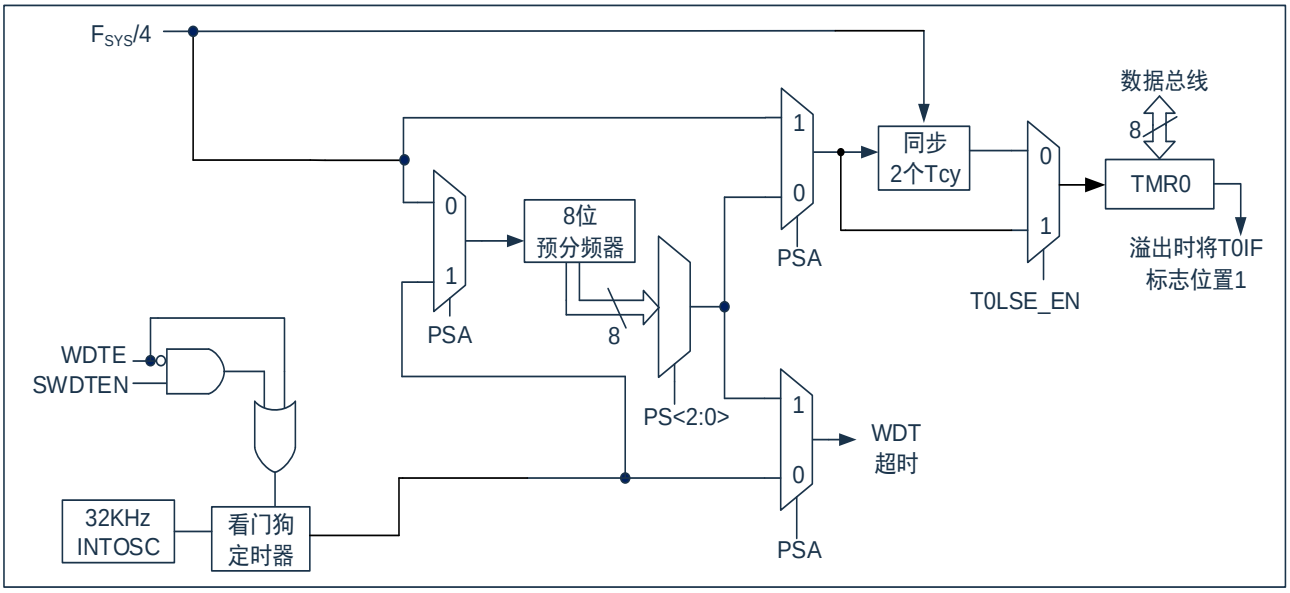



图 8-1：TIMER0/WDT 模块结构图

注：

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

## 8.2 TIMER0 的工作原理

TIMER0 模块既可用作 8 位定时器也可用作 8 位计数器。

### 8.2.1 8 位定时器模式

用作定时器时，TIMER0 模块将在每个指令周期递增（不带预分频器）。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 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。

### 8.2.3 在 TIMER0 和 WDT 模块间切换预分频器

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

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

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

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

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

### 8.2.4 TIMER0 中断

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

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

## 8.3 与 TIMER0 相关寄存器

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

TMR0 为一个 8 位可读写的定时/计数器，OPTION_REG 为一个 8 位读写寄存器，用户可改变 OPTION_REG 的值，来改变 TIMER0 的工作模式等。请参看 0 预分频寄存器（OPTION_REG）的应用。

8 位定时器/计数器 TMR0

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

预分频器 OPTION_REG

| 0x8A | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| OPTION_REG | ---- | ---- | ---- | ---- | PSA | PS2 | PS1 | PS0 |
| R/W | ---- | ---- | ---- | ---- | R/W | R/W | R/W | R/W |
| 复位值 | ---- | ---- | ---- | ---- | 1 | 0 | 1 | 1 |

Bit7~Bit4 未用

Bit3 PSA: 预分频器分配位

0= 预分频器分配给 TIMER0 模块

1= 预分频器分配给 WDT

Bit2~Bit0 PS2~PS0: 预分配参数配置位

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

# 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
- 可编程内部时钟源
- 溢出中断

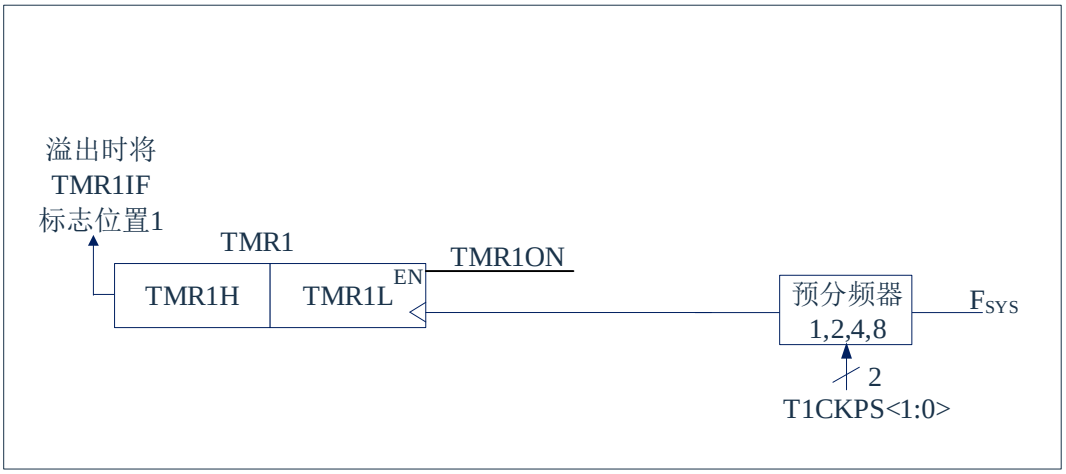



图9-1：TIMER1结构图

## 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。

## 9.3 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。

## 9.4 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 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。

## 9.5 TIMER1 控制寄存器

TIMER1 控制寄存器 T1CON

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

Bit7~Bit6 未用

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~Bit1 未用

Bit0 TMR1ON: TIMER1 使能位

1= 使能 TIMER1

0= 禁止 TIMER1

# 10. 定时计数器 TIMER2

## 10.1 TIMER2 概述

TIMER2 模块是一个 8 位定时器/ 计数器，具有以下特性：

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

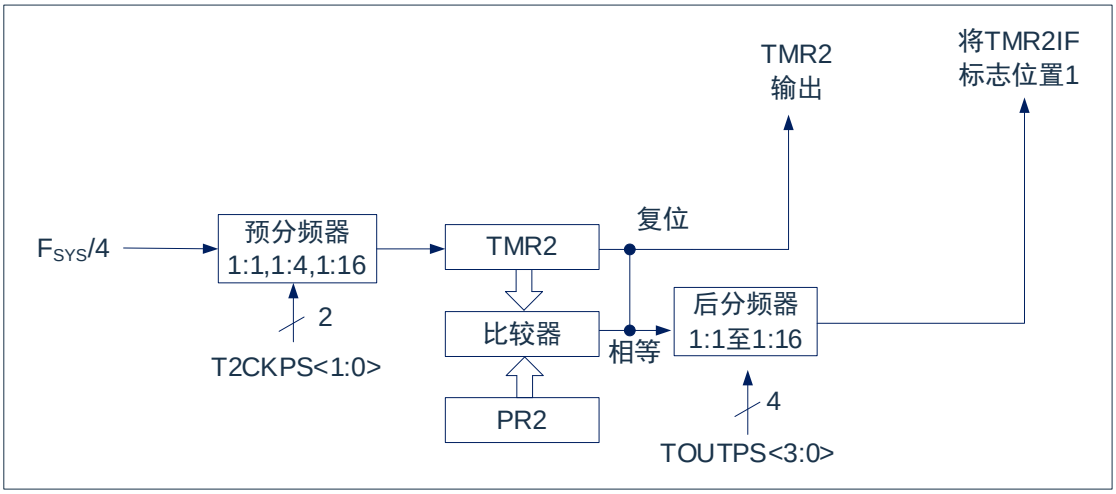



图 10-1：TIMER2 框图

## 10.2 TIMER2 的工作原理

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

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

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

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

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

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

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

预分频器和后分频器计数器在以下情况下被清零：

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

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

## 10.3 TIMER2 相关的寄存器

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

TIMER2 数据寄存器 TMR2

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

TIMER2 周期寄存器 PR2

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

TIMER2 控制寄存器 T2CON

| A6H | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| T2CON | CLK_OUT | TOUTPS3 | TOUTPS2 | TOUTPS1 | TOUTPS0 | TMR2ON | T2CKPS1 | T2CKPS0 |
| 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 |

Bit7 CLK_OUT TIMER2 时钟输出
1= 使能 TIMER2 输出
0= 禁止 TIMER2 输出

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. 定时计数器 TIMER3

## 11.1 TIMER3 概述

TIMER3 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（TMR3H:TMR3L）
- 3 位预分频器
- 支持自动捕获，捕获引脚可选
- 可编程内部时钟源
- 溢出中断
- 可自动捕获高低电平的脉宽

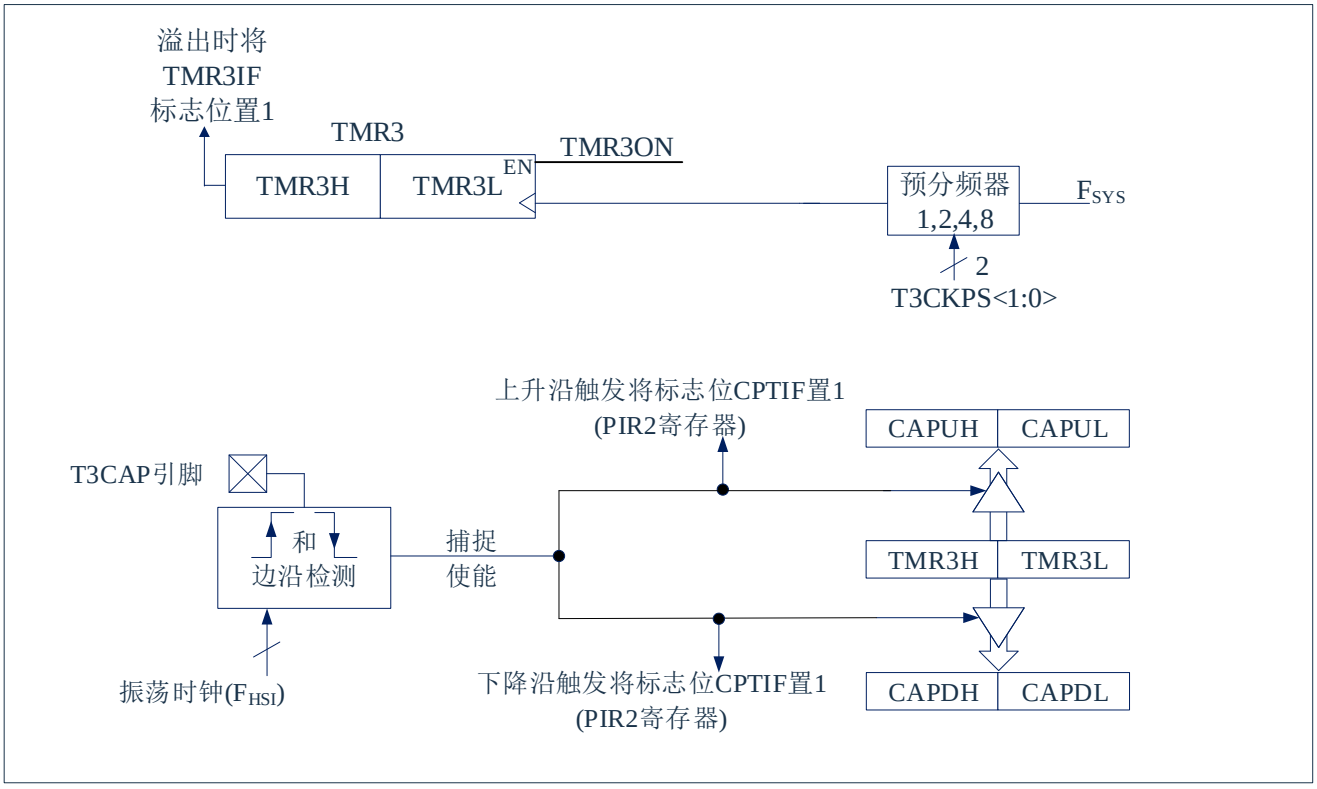



图11-1：TIMER3结构图

## 11.2 TIMER3 的工作原理

TIMER3 模块是一个通过一对寄存器 TMR3H：TMR3L 访问的 16 位递增计数器。写入 TMR3H 或 TMR3L 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。

## 11.3 TIMER3 预分频器

TIMER3 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。T3CON 寄存器的 T3CKPS 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 TMR3H 或 TMR3L 可清零预分频计数器。

## 11.4 TIMER3 中断

一对 TIMER3 寄存器（TMR3H:TMR3L）递增计数到 FFFFH 后，将溢出返回 0000H。当 TIMER3 溢出时，PIR2 寄存器的 TIMER3 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：

- PIE2 寄存器中的 TIMER3 中断允许位；
- INTCON 寄存器中的 GIE 位。

在中断服务程序中将 TMR3IF 位清零可以清除该中断。

注：再次允许该中断前，应将 TMR3H:TMR3L 这对寄存器以及 TMR3IF 位清零。

## 11.5 TIMER3 相关寄存器

TIMER3 控制寄存器 T3CON

| 0x93 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| T3CON | -- | -- | T3CKPS1 | T3CKPS0 | -- | -- | -- | TMR3ON |
| R/W | R/W | R/W | R/W | R/W | R | R | R | R/W |
| 复位值 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit7~Bit6 未用

Bit5~Bit4 T3CKPS<1:0>: TIMER3 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~Bit1 未用

Bit0 TMR3ON: TIMER3 使能位

1= 使能 TIMER3

0= 禁止 TIMER3

# 12. 模数转换（ADC）

## 12.1 ADC 概述

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

ADC 参考电压始终为内部产生。ADC 在转换完成之后可以产生一个中断。

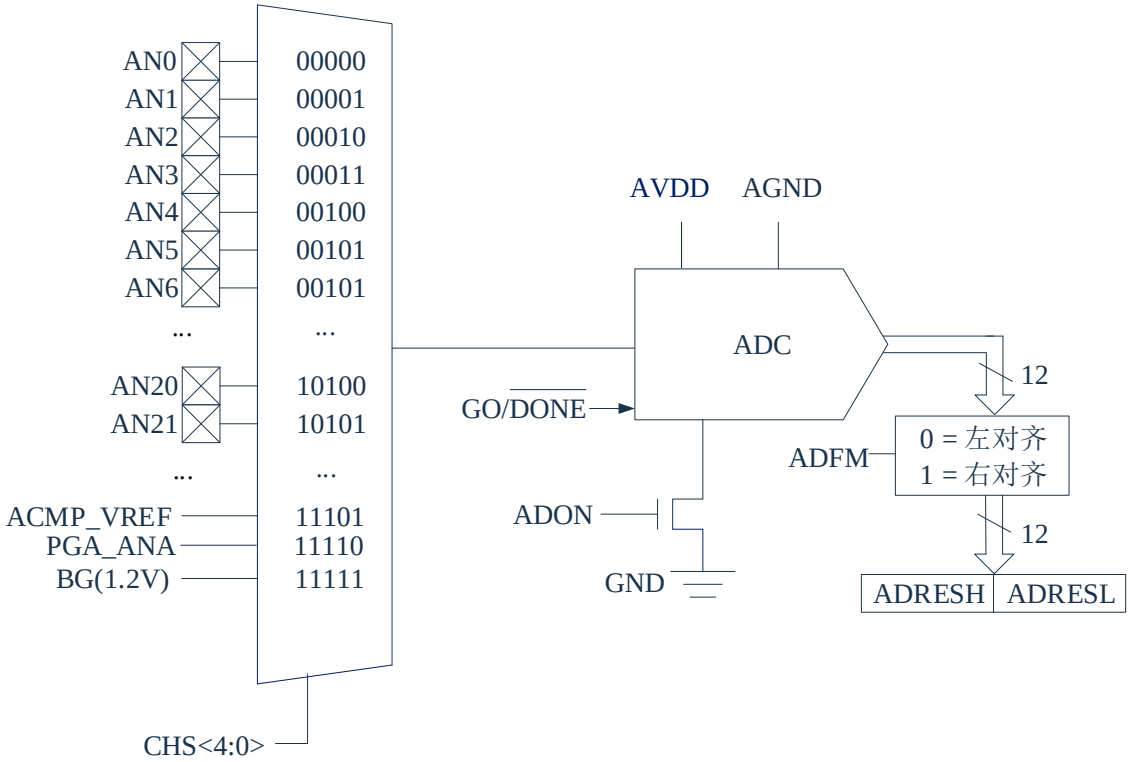



图 12-1：ADC 框图

## 12.2 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。

### 12.2.1 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 0，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。

### 12.2.2 通道选择

由 ADCON0 寄存器的 CHS 位决定将哪个通道连接到采样保持电路。

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

### 12.2.3 ADC 参考电压

ADC 的参考电压始终是由芯片的 $V_{DD}$ 和 GND 提供。

### 12.2.4 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 种可能的时钟分频可供选择：

- $F_{SYS}/16$
- $F_{SYS}/32$
- $F_{SYS}/64$
- $F_{SYS}/128$

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

不同参考电压和不同 $V_{DD}$ 时，需要参考以下表格设置合理的分频。

| 参考电压 | 工作电压 (V) | 最快分频设置 | 转换时间 (us) |
|---|---|---|---|
| | | $F_{SYS}$ = 16MHz | |
| $V_{DD}$ | 4.0~5.5 | 1MHz | 16 |
| $V_{DD}$ | 2.5~5.5 | 500KHz | 32 |
| $V_{DD}$ | 2.5~5.5 | 250KHz | 64 |
| $V_{DD}$ | 2.5~5.5 | 125KHz | 128 |

### 12.2.5 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。

### 12.2.6 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 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。

## 12.3 ADC 工作原理

### 12.3.1 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。

### 12.3.2 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。

### 12.3.3 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 $T_{AD}$ 的延时才能开始下一次采集。延时过后，将自动开始对选定通道的输入信号进行采集。

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

### 12.3.4 ADC 在空闲模式下的工作原理

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

### 12.3.5 ADC 在休眠模式下的工作原理

ADC 模块不可以工作在休眠模式下。休眠模式下 ADC 输入时钟被关闭。

### 12.3.6 A/D 转换步骤

如下步骤给出了使用 ADC 进行模数转换的示例：

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

## 12.4 ADC 硬件触发启动

除了软件触发 ADC 转换，该 ADC 模块还提供了硬件触发启动的方式。PWM 的边沿或周期触发方式。

使用硬件触发 ADC 需要将 ADCEX 置 1，即使能外部触发 ADC 功能。硬件触发信号经过一定延时后将 ADGO 位置 1，转换完毕后将自动清零。开启硬件触发功能后，不会关闭软件触发功能，ADC 空闲状态下，写 1 到 ADGO 位也能启动 AD 转换。

### 12.4.1 PWM 触发 ADC

PWM 可选择是周期点或者零点触发 ADC 转换。ADEGS[1:0]可选择触发方式。

## 12.5 ADC 相关寄存器

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

AD 控制寄存器 ADCON0

| 0xDE | 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= $F_{HSI}$/8
01= $F_{HSI}$/16
10= $F_{HSI}$/32
11= $F_{HSI}$/64

Bit5~Bit2 CHS<4:0>: 模拟通道选择位低四位与CHS4组成五位通道选择
00000= AN0
00001= AN1
00010= AN2
00011= AN3
00100= AN4
00101= AN5
00110= AN6
… …
10100 AN20
10101 AN21
… …
10110~11100= 未用
11101= ACMP_VREF(比较器的负端参考电压，详见ACMP章节)
11110= PGA_ANA(PGA内部输出信号，详见PGA模块章节)
11111= 1.2V(固定参考电压)

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

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

AD 数据寄存器高位 ADCON1

| 0xDF | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ADCON1 | ADFM | CHS4 | ---- | ADEGS | ADCEX | ---- | ---- | ---- |
| R/W | R/W | R/W | R | R/W | R/W | R | R | R |
| 复位值 | 0 | 0 | 0 | 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 模拟通道选择位

Bit5 未用

Bit4 ADEGS: ADC 硬件触发边沿选择位

0= PWM周期的周期点

1= PWM周期的零点

Bit3 ADCEX: ADC硬件触发使能位

1= 使能

0= 禁止

Bit2~ Bit0 未用

AD 数据寄存器高位 ADRESH，ADFM=0

| 0xDD | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ADRESH | ADRES11 | ADRES10 | ADRES9 | ADRES8 | ADRES7 | ADRES6 | ADRES5 | ADRES4 |
| R/W | R | R | R | R | R | R | R | R |
| 复位值 | X | X | X | X | X | X | X | X |

Bit7~Bit0 ADRES<11:4>: ADC结果寄存器位

12位转换结果的高8位

AD 数据寄存器低位 ADRESL，ADFM=0

| 0xDC | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ADRESL | ADRES3 | ADRES2 | ADRES1 | ADRES0 | ---- | ---- | ---- | ---- |
| R/W | R | R | R | R | ---- | ---- | ---- | ---- |
| 复位值 | X | X | X | X | ---- | ---- | ---- | ---- |

Bit7~Bit4 ADRES<3:0>: ADC结果寄存器位

12位转换结果的低4位

Bit3~Bit0 未用

AD 数据寄存器高位 ADRESH，ADFM=1

| 0xDD | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ADRESH | ---- | ---- | ---- | ---- | ---- | ---- | ADRES11 | ADRES10 |
| R/W | ---- | ---- | ---- | ---- | ---- | ---- | R | R |
| 复位值 | ---- | ---- | ---- | ---- | ---- | ---- | X | X |

Bit7~Bit2　　未用

Bit1~Bit0　　ADRES<11:10>:　ADC结果寄存器位

12位转换结果的高2位

AD 数据寄存器低位 ADRESL，ADFM=1

| 0xDC | 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。

# 13. 通用同步/异步收发器（USART）

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

USART 模块包含如下功能：

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

以下图 13-1 和图 13-2 为 USART 收发器的框图。

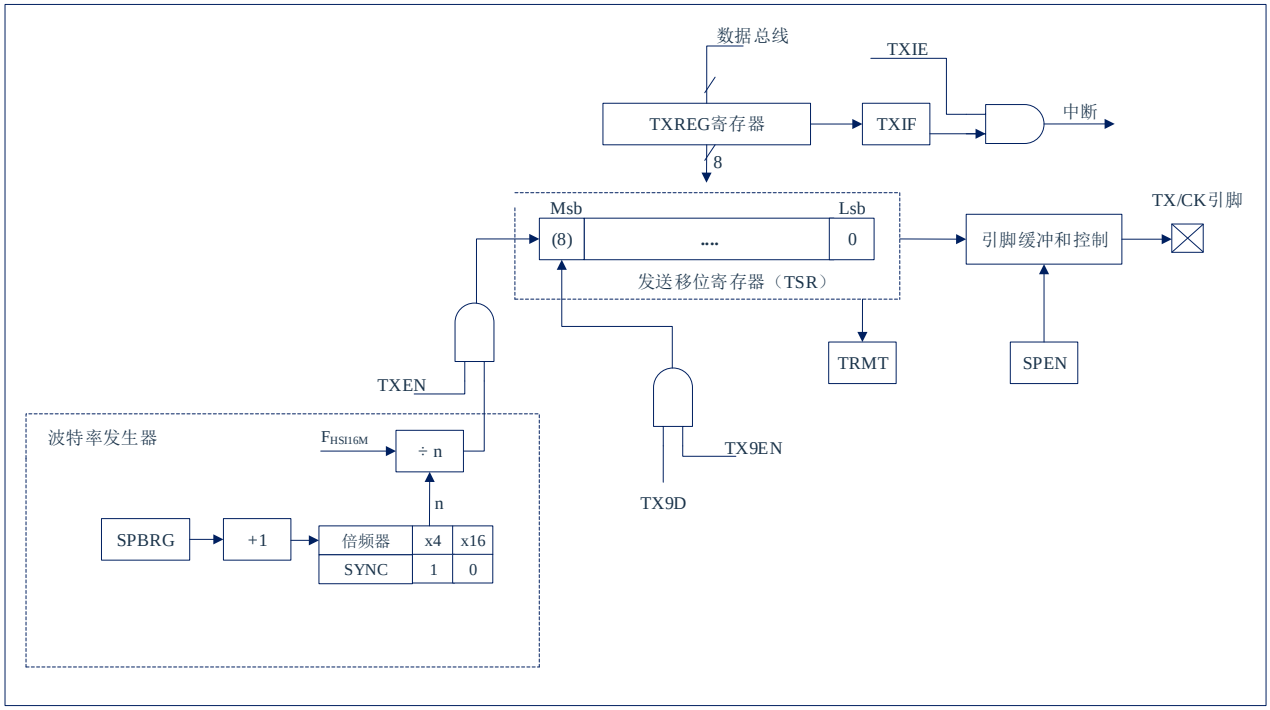



图13-1：USART发送框图

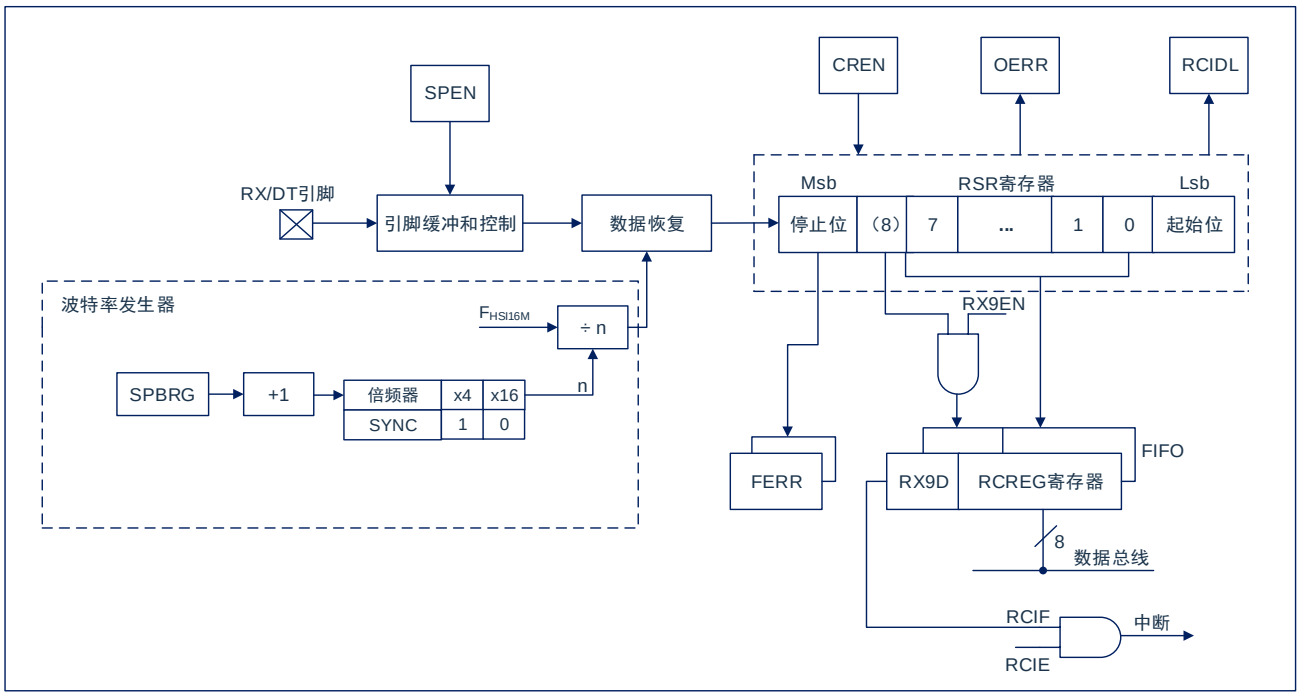



图13-2：USART接收框图

USART 模块的操作是通过 3 个寄存器控制的：

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

# 13.1 USART 异步模式

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

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

## 13.1.1 USART 异步发生器

图 11-1 所示为 USART 发送器的框图。发送器的核心是串行发送移位寄存器（TSR），该寄存器不能由软件直接访问。TSR 从 TXREG 发送缓冲寄存器获取数据。

### 13.1.1.1 使能发送器

通过配置如下三个控制位使能 USART 发送器，以用于异步操作：

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

假设所有其他 USART 控制位处于其默认状态。

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

注：
1) 当将 SPEN 位和 TXEN 位置 1，SYNC 位清零，TX/CK I/O 引脚被自动配置为输出引脚，无需考虑相应 TRIS 位的状态。
2) 当将 SPEN 位和 CREN 位置 1，SYNC 位清零，RX/DT I/O 引脚被自动配置为输入引脚，无需考虑相应 TRIS 位的状态。

### 13.1.1.2 发送数据

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

#### 13.1.1.3 发送中断标志

只要使能 USART 发送器且 TXREG 中没有待发送数据，就将 PIR1 寄存器的 TXIF 中断标志位置 1。换句话说，只有当 TSR 忙于处理字符和 TXREG 中有排队等待发送的新字符时，TXIF 位才处于清零状态。写 TXREG 时，不立即清零 TXIF 标志位。TXIF 在写指令后的第 2 个指令周期清零。在写 TXREG 后立即查询 TXIF 会返回无效结果。TXIF 为只读位，不能由软件置 1 或清零。

可通过将 PIE1 寄存器的 TXIE 中断允许位置 1 允许 TXIF 中断。然而，只要 TXREG 为空，不管 TXIE 允许位的状态如何都会将 TXIF 标志位置 1。

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

#### 13.1.1.4 TSR 状态

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

| 注：TSR 寄存器并未映射到数据存储器中，因此用户不能直接访问它。 |
|---|

#### 13.1.1.5 发送 9 位字符

USART 支持 9 位字符发送。当 TXSTA 寄存器的 TX9EN 位置 1 时，USART 将移出每个待发送字符的 9 位。TXSTA 寄存器的 TX9D 位为第 9 位，即最高数据位。当发送 9 位数据时，必须在将 8 个最低位写入 TXREG 之前，写 TX9D 数据位。在写入 TXREG 寄存器后会立即将 9 个数据位传输到 TSR 移位寄存器。

#### 13.1.1.6 设置异步发送

1. 初始化 SPBRG 寄存器，以获得所需的波特率（请参见“USART 波特率发生器（BRG）”
2. 通过将 SYNC 位清零并将 SPEN 位置 1 使能异步串口。
3. 如果需要 9 位发送，将 TX9EN 控制位置 1。当接收器被设置为进行地址检测时，将数据位的第 9 位置 1，指示 8 个最低数据位为地址。
4. 将 TXEN 控制位置 1，使能发送；这将导致 TXIF 中断标志位置 1。
5. 如果需要中断，将 PIE1 寄存器中的 TXIE 中断允许位置 1；如果 INTCON 寄存器的 GIE 和 PEIE 位也置 1 将立即产生中断。
6. 若选择发送 9 位数据，第 9 位应该被装入 TX9D 数据位。
7. 将 8 位数据装入 TXREG 寄存器开始发送数据。

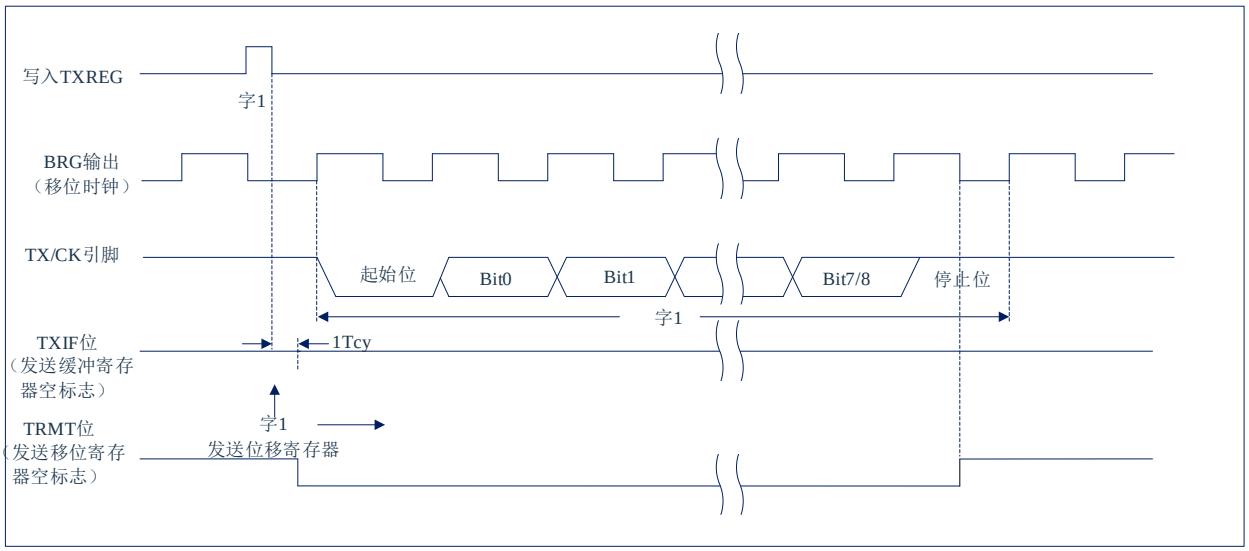



图 13-3：异步发送

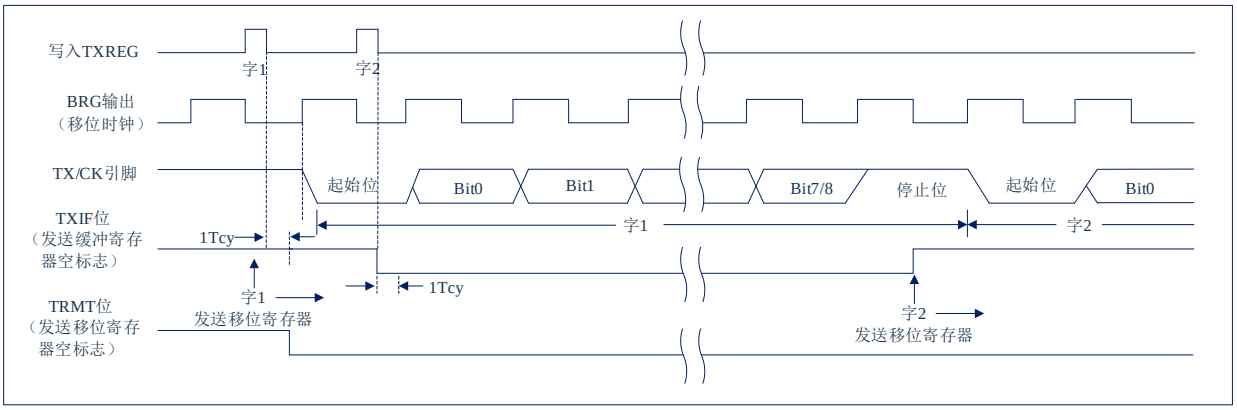



图 13-4：异步发送（背靠背）

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

例 1(汇编)：异步发送(背靠背)

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

例 2(C 语言)：异步发送(背靠背)

```
INTERRUPT_USART_SEND()          中断异步发送子函数
{
        if(TXIF)                判断 TXIF 标志，为 0 则跳出该中断函数，为 1 则 UASRT 发送缓冲器为空。
        {
        TXREG = 0x40;           040H 为发送的数据，将需要发送到内容写入 TXREG
        if(TRMT)                判断 TRMT 标志，为 0 则说明该次写入成功跳出该中断函数，为 1 则说明该次
                                写入失败需要再次写入
        TXREG = 0x40;           将 040H 重新加载到 TXREG 中，进行二次写入
        }
}
```

### 13.1.2 USART 异步接收器

异步模式通常用于 RS-232 系统。图 14-2 给出了接收器的框图。在 RX/DT 引脚上接收数据和驱动数据恢复电路。数据恢复电路实际上是一个以 16 倍波特率为工作频率的高速移位器，而串行接收移位寄存器（ReceiveShiftRegister，RSR）则以比特率工作。当字符的全部 8 位或 9 位数据位被移入后，立即将它们传输到一个 2 字符的先入先出（FIFO）缓冲器。FIFO 缓冲器允许接收 2 个完整的字符和第 3 个字符的起始位，然后必须由软件将接收到的数据提供给 USART 接收器。FIFO 和 RSR 寄存器不能直接由软件访问。通过 RCREG 寄存器访问接收到的数据。

#### 13.1.2.1 使能接收器

通过配置如下三个控制位使能 USART 接收器，以用于异步操作。

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

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

注：

1) 当将 SPEN 位和 TXEN 位置 1，SYNC 位清零，TX/CKI/O 引脚被自动配置为输出引脚，无需考虑相应 TRIS 位的状态。
2) 当将 SPEN 位和 CREN 位置 1，SYNC 位清零，RX/DTI/O 引脚被自动配置为输入引脚，无需考虑相应 TRIS 位的状态。

#### 13.1.2.2 接收数据

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

当接收到所有数据位和停止位后，RSR 中的字符会被立即传输到 USART 的接收 FIFO 并将 PIR1 寄存器的 RCIF 中断标志位置 1。通过读 RCREG 寄存器将 FIFO 最顶端的字符移出 FIFO。

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

#### 13.1.2.3 接收中断

只要使能 USART 接收器且在接收 FIFO 中没有未读数据，PIR1 寄存器中的 RCIF 中断标志位就会清零。RCIF 中断标志位为只读，不能由软件置 1 或清零。

通过将下列所有位均置 1 来允许 RCIF 中断：

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

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

#### 13.1.2.4 接收帧错误

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

由 RCSTA 寄存器的 FERR 位获取帧错误状态。必须在读 RCREG 寄存器之后读 FERR 位。

帧错误（FERR=1）并不会阻止接收更多的字符。无需清零 FERR 位。

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

注：如果接收 FIFO 缓冲器中所有接收到的字符都有帧错误，重复读 RCREG 不会清零 FERR 位。

#### 13.1.2.5 接收溢出错误

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

#### 13.1.2.6 接收 9 位字符

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

#### 13.1.2.7 异步接收设置

1. 初始化 SPBRG 寄存器，以获得所需的波特率。
   （请参见“USART 波特率发生器（BRG）”章节。
2. 将 SPEN 位置 1，使能串行端口。必须清零 SYNC 位以执行异步操作。
3. 如果需要中断，将 PIE1 寄存器中的 RCIE 位和 INTCON 寄存器的 GIE 和 PEIE 位置 1。
4. 如果需要接收 9 位数据，将 RX9EN 位置 1。
5. 将 CREN 位置 1 使能接收。
6. 当一个字符从 RSR 传输到接收缓冲器时，将 RCIF 中断标志位置 1。如果 RCIE 中断允许位也置 1 还将产生中断。
7. 读 RCREG 寄存器，从接收缓冲器获取接收到的 8 个低数据位。
8. 读 RCSTA 寄存器获取错误标志位和第 9 位数据位（如果使能 9 位数据接收）。
9. 如果发生溢出，通过清零 CREN 接收器使能位清零 OERR 标志。

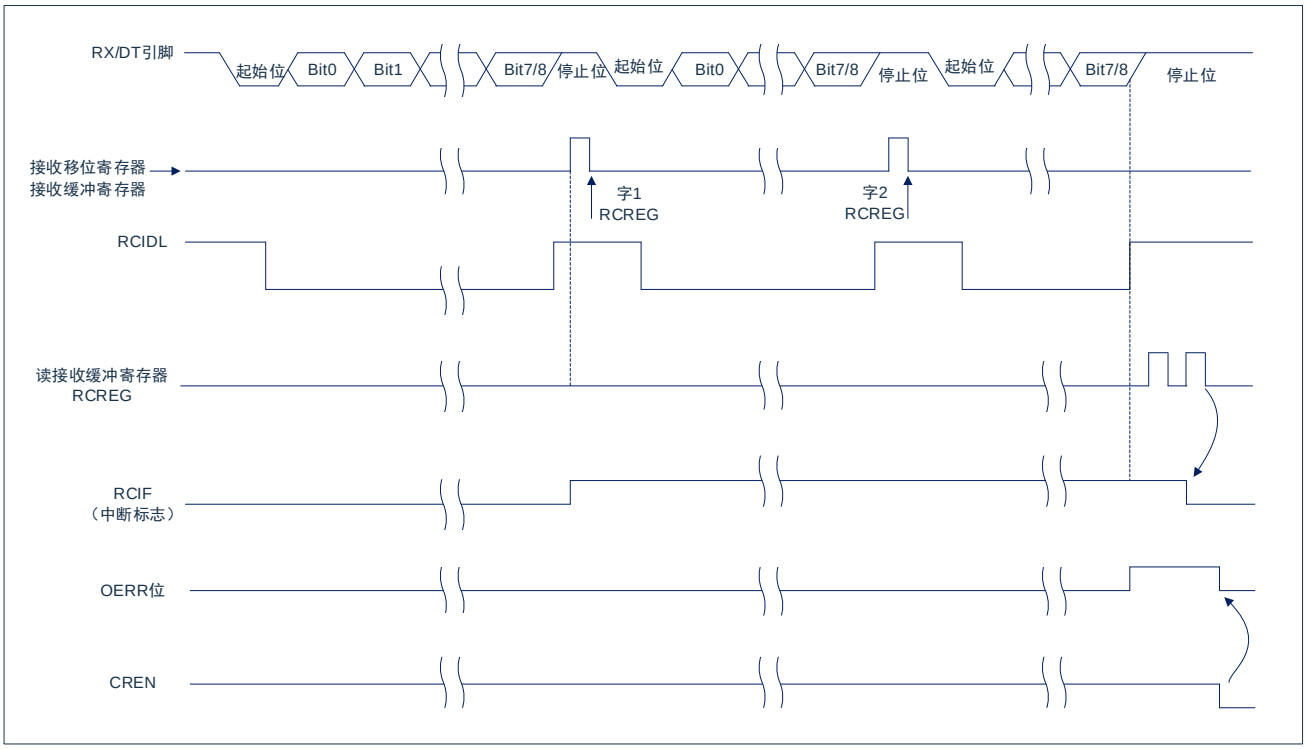



图 13-5：异步接收

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

## 13.2 异步操作时的时钟准确度

由厂家校准内部振荡电路（INTOSC）的输出。但在 VDD 或温度变化时，INTOSC 会发生频率漂移，从而会直接影响异步波特率。可通过以下方法调整波特率时钟，但需要某种类型的参考时钟源。

## 13.3 USART 相关寄存器

TXSTA：发送状态和控制寄存器

| 0xD6 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| TXSTA | CSRC | TX9EN | TXEN | SYNC | SCKP | -- | TRMT | TX9D |
| R/W | R/W | R/W | R/W | R/W | R/W | -- | R | R/W |
| 复位值 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

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

Bit6 TX9EN: 9位发送使能位
1= 选择9位发送
0= 选择8位发送

Bit5 TXEN: 发送使能位(1)
1= 使能发送
0= 禁止发送

Bit4 SYNC: USART模式选择位
1= 同步模式
0= 异步模式

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

Bit2 未用

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

Bit0 TX9D: 发送数据的第9位
可以是地址/数据位或奇偶校验位

注：同步模式下，SREN/CREN 会颠覆 TXEN 的值。

RCSTA：接收状态和控制寄存器

| 0xD3 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| RCSTA | SPEN | RX9EN | SREN | CREN | RCIDL | FERR | OERR | RX9D |
| R/W | R/W | R/W | R/W | R/W | R | R | R | R |
| 复位值 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

Bit7 SPEN: 串行端口使能位
1= 使能串行端口（将RX/DT和TX/CK引脚配置为串行端口引脚）
0= 禁止串行端口（保持在复位状态）
Bit6 RX9EN: 9位接收使能位
1= 选择9位接收
0= 选择8位接收
Bit5 SREN: 单字节接收使能位
异步模式: 任意值
同步主控模式:
1=使能单字节接收
0=禁止单字节接收
接收完成后清零该位
同步从动模式: 任意值
Bit4 CREN: 连续接收使能位
异步模式:
1=使能接收
0=禁止接收
同步模式:
1=使能连续接收直到清零CREN使能位（CREN覆盖SREN）
0=禁止连续接收
Bit3 RCIDL: 接收空闲标志位
异步模式: 1=接收器空闲
0=已接收到起始位，接收器正在接收数据
同步模式: 任意值
Bit2 FERR: 帧错误位
1= 帧错误（可通过读RCREG寄存器更新并接收下一个有效字节）
0= 没有帧错误
Bit1 OERR: 溢出错误位
1= 溢出错误（可通过清零CREN位清零）
0= 没有溢出错误
Bit0 RX9D: 接收到数据的第9位
此位可以是地址/数据位或奇偶校验位，必须由用户固件计算得到

发送数据寄存器 TXREG

| 0xD4 | 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 RCREG

| 0xD5 | 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 SPBRG

| 0xD7 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| SPBRGx | | | | | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| 复位值 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

波特率微调寄存器 BRG_ADJ

| 0xD2 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| BRG_ADJ | EN_ADJx | --- | --- | ADJx <4:0> | | | | |
| R/W | R/W | --- | --- | R/W | R/W | R/W | R/W | R/W |
| 复位值 | 0 | --- | --- | 0 | 0 | 0 | 0 | 0 |

Bit7 EN_ADJx: 波特率微调值使能位
0: 不使能
1: 使能

Bit6~Bit5 未用

Bit4~Bit0 ADJx<4:0> 波特率微调值

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

## 13.4 USART 波特率发生器（BRG）

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

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

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

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

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

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

公式 1：计算波特率误差

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

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

求解 SPBRGx：

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

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

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

表 13-1：波特率公式

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

说明：n= SPBRGx 寄存器的值，y=ADJx<4:0>的值。

表 13-2：异步模式下的波特率（EN_ADJx=0）

| 目标波特率 | SYNCx=0 | | |
|---|---|---|---|
| | $F_{HSI}$=16.00MHz | | |
| | 实际波特率 | 误差（%） | SPBRGx 值 |
| 2400 | ---- | ---- | ---- |
| 9600 | 9615 | 0.16 | 103 |
| 10417 | 10417 | 0 | 95 |
| 14400 | 14492 | 0.64 | 68 |
| 19200 | 19230 | 0.16 | 51 |

表 13-3：异步模式下的波特率（EN_ADJx=1）

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

## 13.5 USART 同步模式

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

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

同步发送无需使用起始位和停止位。

### 13.5.1 同步主控模式

下列位用来将 USART 配置为同步主控操作：

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

将 TXSTA 寄存器的 SYNC 位置 1，可将 USART 配置用于同步操作。将 TXSTA 寄存器的 CSRC 位置 1，将器件配置为主控器件。将 RCSTA 寄存器的 SREN 和 CREN 位清零，以确保器件处于发送模式，否则器件配置为接收模式。将 RCSTA 寄存器的 SPEN 位置 1，使能 USART。

#### 13.5.1.1 主控时钟

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

#### 13.5.1.2 时钟极性

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

#### 13.5.1.3 同步主控发送

由器件的 RX/DT 引脚输出数据。当 USART 配置为同步主控发送操作时，器件的 RX/DT 和 TX/CK 输出引脚自动使能。

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

注：TSR 寄存器并未映射到数据存储器中，因此用户不能直接访问它。

#### 13.5.1.4 同步主控发送设置

1. 初始化 SPBRG 寄存器，以获得所需的波特率。
   （请参见“USART 波特率发生器（BRG）”章节。
2. 将 SYNC、SPEN 和 CSRC 位置 1，使能同步主控串行端口。
3. 将 SREN 和 CREN 位清零，禁止接收模式。
4. 将 TXEN 位置 1 使能发送模式。
5. 如果需要发送 9 位字符，将 TX9EN 置 1。
6. 若需要中断，将 PIE1 寄存器中的 TXIE 位，以及 INTCON 寄存器中的 GIE 和 PEIE 位置 1。
7. 如果选择发送 9 位字符，应该将第 9 位数据装入 TX9D 位。
8. 通过将数据装入 TXREG 寄存器启动发送。

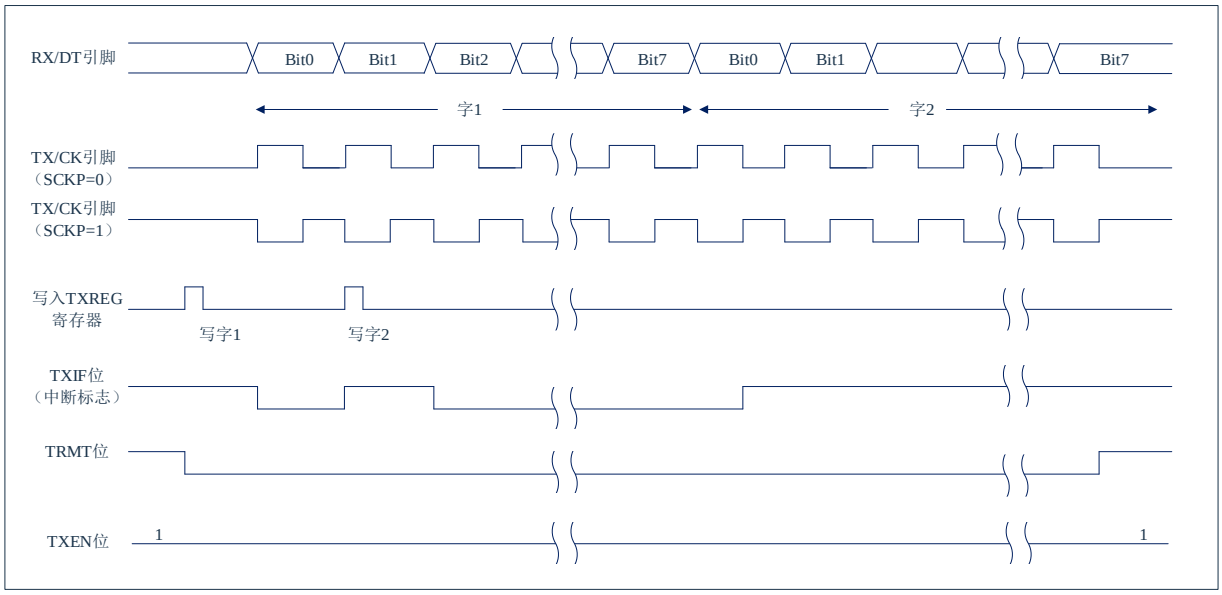



图 13-6：同步发送

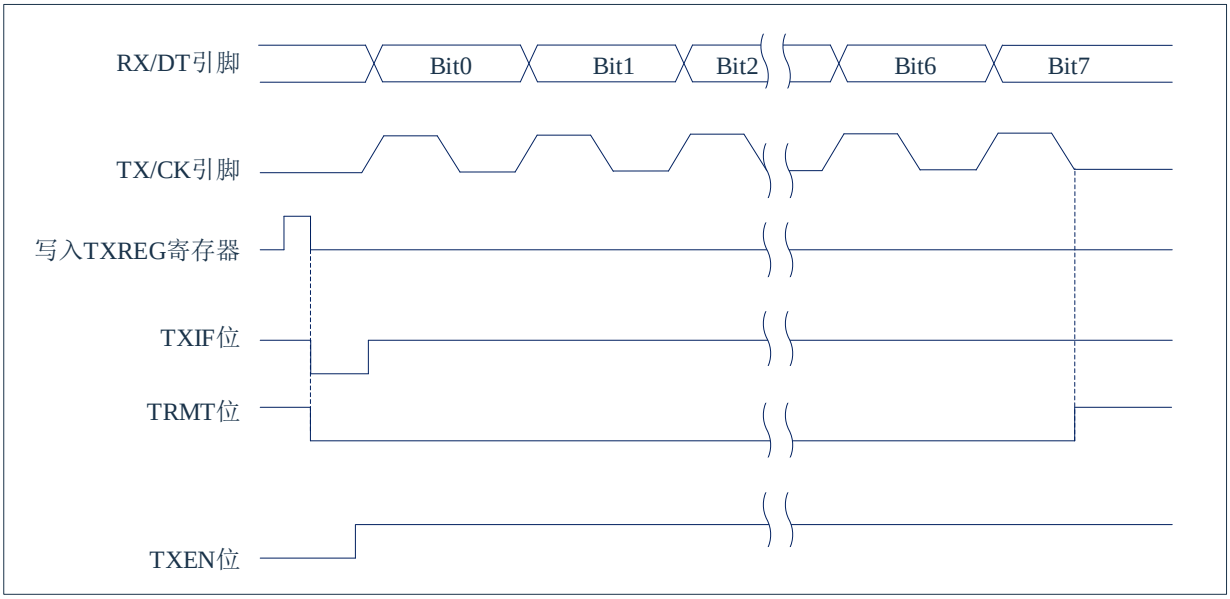



图 13-7：同步发送（通过 TXEN）

#### 13.5.1.5 同步主控接收

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

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

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

#### 13.5.1.6 从时钟

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

#### 13.5.1.7 接收溢出错误

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

#### 13.5.1.8 接收 9 位字符

USART 支持接收 9 位字符。当 RCSTA 寄存器的 RX9EN 位置 1 时，USART 将接收到的每个字符的 9 位数据移入 RSR。当从接收 FIFO 缓冲器读取 9 位数据时，必须在读 RCREG 的 8 个低位之后，读取 RX9D 数据位。

#### 13.5.1.9 同步主控接收设置

1. 初始化 SPBRG 寄存器，以获得所需的波特率。（注：必须满足 SPBRG>05H）
2. 将 SYNC、SPEN 和 CSRC 位置 1 使能同步主控串行端口。
3. 确保将 CREN 和 SREN 位清零。
4. 如果使用中断，将 INTCON 寄存器的 GIE 和 PEIE 位置 1，并将 PIE1 寄存器的 RCIE 位也置 1。
5. 如果需要接收 9 位字符，将 RX9EN 位置 1。
6. 将 SREN 位置 1，启动接收，或将 CREN 位置 1 使能连续接收。
7. 当字符接收完毕后，将 RCIF 中断标志位置 1。如果允许位 RCIE 置 1，还会产生一个中断。
8. 读 RCREG 寄存器获取接收到的 8 位数据。
9. 读 RCSTA 寄存器以获取第 9 个数据位（使能 9 位接收时），并判断接收过程中是否产生错误。
10. 如果产生溢出错误，清零 RCSTA 寄存器的 CREN 位或清零 SPEN 以复位 USART 来清除错误。

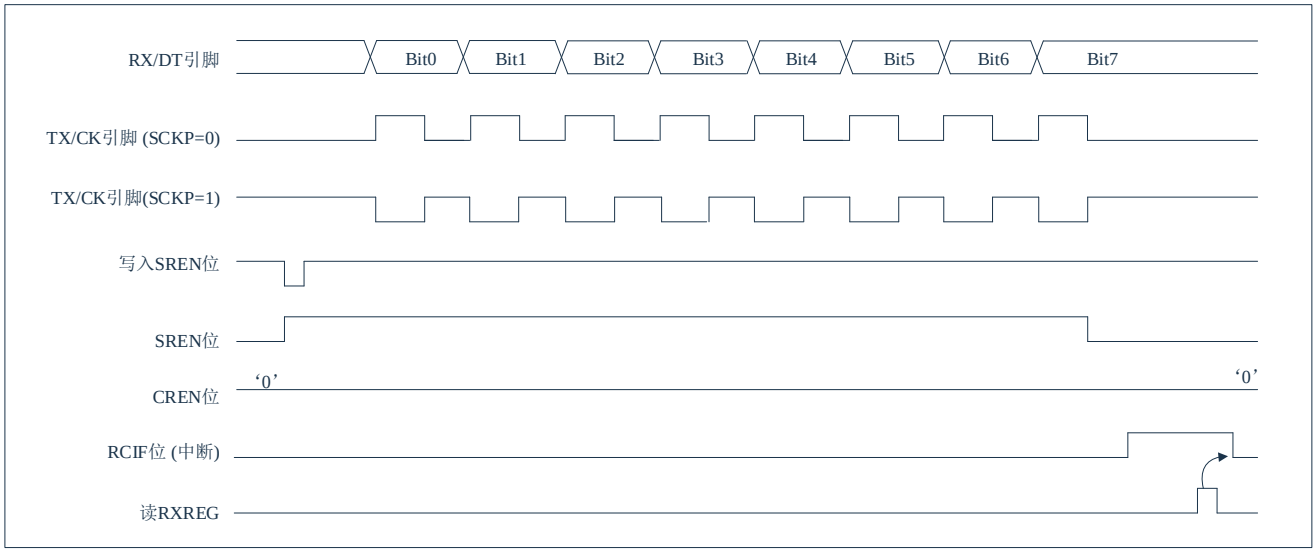



图 13-8：同步接收（主控模式，SREN）

注：时序图说明了 SREN=1 时的同步主控模式。

### 13.5.2 同步从动模式

下列位用来将 USART 配置为同步从动操作：

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

将 TXSTA 寄存器的 SYNC 位置 1，可将器件配置用于同步操作。将 TXSTA 寄存器的 CSRC 位置 1，将器件配置为从动器件。将 RCSTA 寄存器的 SREN 和 CREN 位清零，以确保器件处于发送模式，否则器件将被配置为接收模式。将 RCSTA 寄存器的 SPEN 位置 1，使能 USART。

#### 13.5.2.1 USART 同步从动发送

同步主控和从动模式的工作原理是相同的（见章节“同步主控发送”）。

#### 13.5.2.2 同步从动发送设置

1. 将 SYNC 和 SPEN 位置 1 并将 CSRC 位清零。
2. 将 CREN 和 SREN 位清零。
3. 如果使用中断，将 INTCON 寄存器的 GIE 和 PEIE 位置 1，并将 PIE1 寄存器的 TXIE 位也置 1。
4. 如果需要发送 9 位数据，将 TX9EN 位置 1。
5. 将 TXEN 位置 1 使能发送。
6. 若选择发送 9 位数据，将最高位写入 TX9D 位。
7. 将低 8 位数据写入 TXREG 寄存器开始传输。

#### 13.5.2.3 USART 同步从动接收

除了以下不同外，同步主控和从动模式的工作原理相同。

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

#### 13.5.2.4 同步从动接收设置

1. 将 SYNC 和 SPEN 位置 1 并将 CSRC 位清零。
2. 如果使用中断，将 INTCON 寄存器的 GIE 和 PEIE 位置 1，并将 PIE1 寄存器的 RCIE 位也置 1。
3. 如果需要接收 9 位字符，将 RX9EN 位置 1。
4. 将 CREN 位置 1，使能接收。
5. 当接收完成后，将 RCIF 位置 1。如果 RCIE 已置 1，还会产生一个中断。
6. 读 RCREG 寄存器，从接收 FIFO 缓冲器获取接收到的 8 个低数据位。
7. 如果使能 9 位模式，从 RCSTA 寄存器的 RX9D 位获取最高位。

如果产生溢出错误，清零 RCSTA 寄存器的 CREN 位或清零 SPEN 位以复位 USART 来清除错误。

# 14. 捕捉模块（CPT）

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

在捕捉模式下，需要用到定时器 TIMER3。

## 14.1 捕捉模式

该捕捉模式为外部捕捉。该模式主要用来捕捉外部输入的 PWM 波形信息，所以在使能捕捉模式时，需要将对应管脚方向配置为输入模式。

当使能捕捉模式后，会自动将计数器清零，直到出现第一次边沿才开始计数。16 位计数从 0 开始向上计数。如果溢出且没有产生捕获操作，则计数器清零继续向上计数。

在使能捕捉模式后，当捕捉管脚出现上升沿，会把定时器的 16 位值分别放到 CAPUH 和 CAPUL 寄存器中，并将定时器复位重新计数，当捕捉管脚出现下降沿，会把定时器的 16 位值分别放到 CAPDH 和 CAPDL 寄存器中，并将定时器复位重新计数。发生捕捉会产生捕捉中断，若开启了中断则会进入相应的处理程序，否则值置位相关标志。

用户可以读取 CAPUH/CAPUL 和 CAPDH/CAPDL 等寄存器的值来计算捕获信号的频率和占空比。

在设置为捕捉模式下，也可以通过读取口线数据寄存器来判断口线的高或低电平状态。

## 14.2 CPT 引脚配置

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

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

## 14.3 软件中断

用户应该保持 PIE2 寄存器中的 CAPIE 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 PIR2 寄存器中的中断标志位 CAPIF。

## 14.4 捕捉相关寄存器

捕捉功能控制寄存器 CAPCON

| 0xB0 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| CAPCON | CAPM_EN | CAPIO_SEL | --- | --- | --- | --- | CPT_UP | CPT_DOWN |
| 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 CAPM_EN: 捕捉模式使能位
1= 使能捕捉模式
0= 关闭捕捉模式
Bit6 CAPIO_SEL: 捕捉功能 IO 输入选择
1= P21
0= P13
Bit5~Bit2 未用 未用
Bit0 CAP_UP 捕捉上升沿信号标志
1= 已捕捉到上升沿信号
0= 未捕捉到上升沿信号
Bit0 CAP_DOWN 捕捉下降沿信号标志
1= 已捕捉到下降沿信号
0= 未捕捉到下降沿信号

上升沿捕捉数据低位寄存器 CAPUL

| 0xB1 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| CAPUL | CAPU<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 |

上升沿捕捉数据高位寄存器 CAPUH

| 0xB2 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| CAPUH | CAPU<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 |

下降沿捕捉数据低位寄存器 CAPDL

| 0xB3 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| CAPDL | CAPD<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 |

下降沿捕捉数据低位寄存器 CAPDH

| 0xB4 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| CAPDH | CAPD<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 |

# 15. 增强型 PWM 模块

## 15.1 概述

### 15.1.1 功能

增强型 PWM 模块支持 6 路 PWM 发生器，可以配置成 3 对分别带有编程死区发生器的互补 PWM（PWM0-PWM1，PWM2-PWM3，PWM4-PWM5）。

可通过寄存器选择 PWM0~PWM5 可输出到任意 PWMn 的管脚中。

### 15.1.2 特性

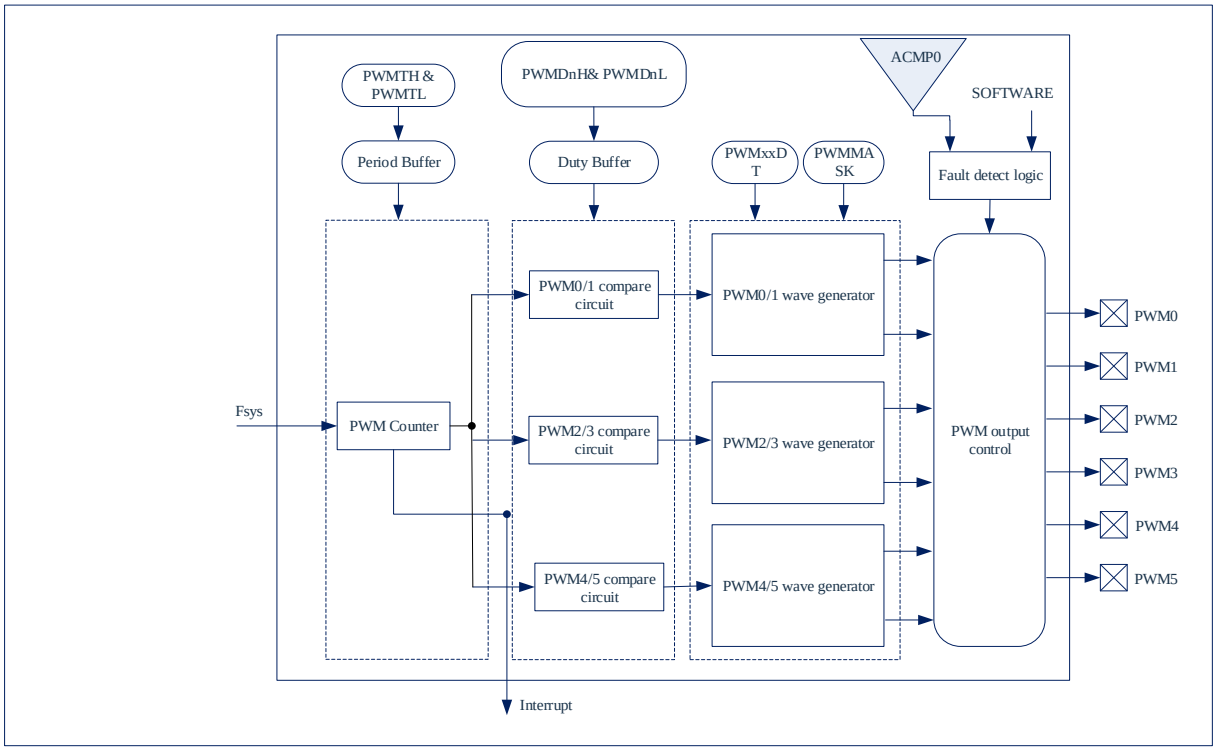
增强型 PWM 模块具有如下特性：

- 3 组互补 PWM 对输出：（PWM0-PWM1）、（PWM2-PWM3）、（PWM4-PWM5）
- 支持边沿对齐，中心对齐 2 种模式。
- 互补的 PWM 中，支持可编程死区发生器。
- 每路 PWM 有独立的极性控制。
- 硬件刹车保护（支持软件触发）。
- 比较事件可触发硬件刹车保护。
- PWM 零点或周期可触发启动 AD 转换。

## 15.2 配置

### 15.2.1 功能框图

增强型 PWM 功能框图如下图所示：



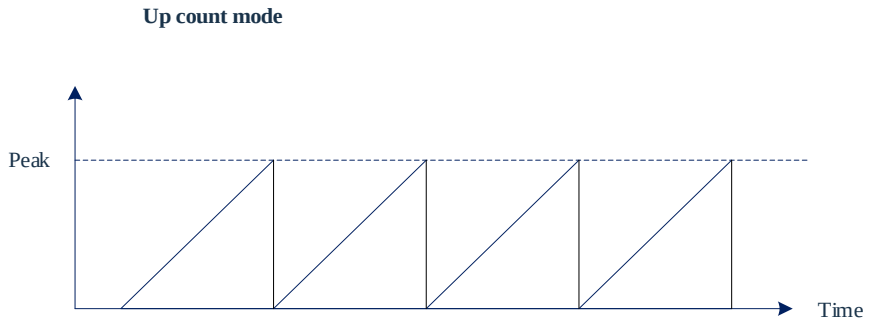
### 15.2.2 各功能模块描述

增强型 PWM 模块是由 PWM 计数器模块、输出比较单元、波形发生器、故障检测和输出控制器组成。

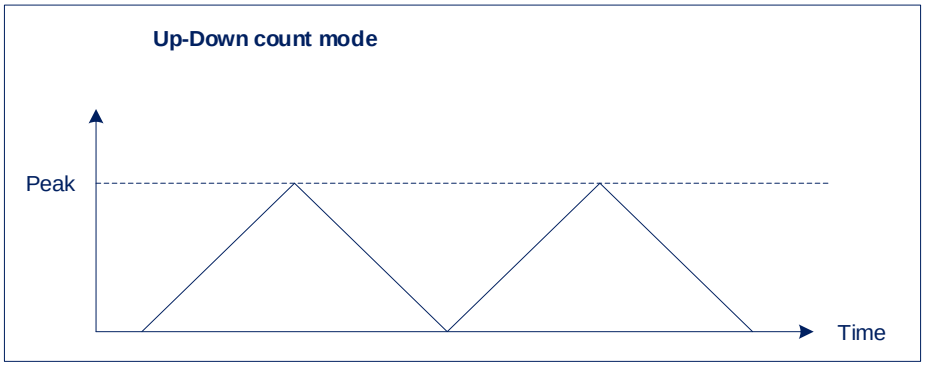
**PWM 计数器：**

系统时钟输入增强型 PWM 模块；周期寄存器（PWMPH，PWMPL）组成的 10 位控制寄存器用于设置 6 个 PWM 的共用周期。为了防止在 PWM 运行的过程中随意修改 PWM 的周期设置，采用缓冲寄存器（Period Buffer）。如果 PWM 设置为连续运行模式（PWMnCNTM=1），在每个 PWM 的零点会自动将周期寄存器的值加载到缓冲寄存器（Period Buffer）当中。

PWM 计数器有两种计数模式：向上计数（Up count mode）和上下计数（Up-Down count mode）。向上计数模式如下图所示：




上下计数模式如下图所示：




**OCU：**

输出比较单元（OCU）是由占空比寄存器（PWMDnH，PWMDnL）组成，用于设置 6 个通道的 PWM 占空比。同样地，为了防止在 PWM 运行的过程中随意修改 PWM 的占空比设置，采用缓冲寄存器（Duty Buffer）和 PWM 计数器进行比较以进行输出电平的翻转。在每个 PWM 的零点会自动将占空比寄存器的值加载到缓冲寄存器（Duty Buffer）当中。

**WFG：**

波形发生器是由死区控制单元和输出极性控制单元组成。针对带死区的互补输出，PWMDT0/ PWMDT1 用于分别设置 PWM 上升沿和下降沿的死区时间；再结合极性控制寄存器（PWMPINV）对 PWM 的输出极性进行控制。

**故障检测（刹车功能）：**

故障检测模块内嵌在增强型 PWM 电路中，配置为输入故障侦测，是为了保护系统防止器件损坏。一旦检测到有效的故障信号输入，则强制关断 PWM 的输出。为了适应不同的驱动要求，关断的电平可以通过 PWM 刹车数据寄存器：PWMFBKD 进行配置。

**掩码输出：**

针对类似方波电机控制这种特殊的应用场合，掩码输出显得尤为重要。PWM 每个通道都有单独的掩码控制位和掩码数据位，通过掩码控制寄存器 PWMMASKE 和掩码数据寄存器 PWMMASKD 设置。

当掩码输出禁止 PWMnMASKE=0 时，PWMn 输出正常的 PWM 波形；

当掩码输出使能 PWMnMASKE=1 时，PWMn 输出掩码寄存器 PWMnMASKD 的数据。

**输出控制器：**

输出控制器，用于对 PWM 的输出状态进行控制。PWM 输出使能控制寄存器 PWMCON1 用于设置各通道的输出使能。发生故障需要强制关断 PWM 时，MCU 可根据刹车数据寄存器 PWMFBKD 中的设置输出相应的电平以适应不同外设的需求。

### 15.2.3 相关 IO 口描述

使用增强型 PWM 模块前需要先将相关端口配置成输出通道，PWM 通道在引脚分配图上用 PWMn 来标注，PWM0~PWM5 均可通过 config 配置来进行输出到任意一个通道，但优先级会按照 PWM0 至 PWM5 进行排序。例如：

当 PWM0 和 PWM1 同时分配到一个 PWMn 时，会优先 PWM0 从该通道输出，PWM1 无法从该通道输出

当 PWM1 和 PWM2 同时分配到一个 PWMn 时，会优先 PWM1 从该通道输出，PWM2 无法从该通道输出

当 PWM0 和 PWM2 同时分配到一个 PWMn 时，会优先 PWM0 从该通道输出，PWM2 无法从该通道输出

当 PWM1 和 PWM3 同时分配到一个 PWMn 时，会优先 PWM1 从该通道输出，PWM3 无法从该通道输出

如此类推。

## 15.3 相关寄存器说明

PWM 控制寄存器 PWMCON0

| 0xC8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PWMCON0 | PWMLE | PWMCNTE | ---- | PWMFBKSW | PWMFB | CNTTYPE | CLKDIV | |
| R/W | R/W | R/W | R | R/W | R | R/W | R/W | R/W |
| 复位值 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Bit7 PWMLE: PWM通道的数据加载使能位（加载完成后硬件清零）
1= 使能加载周期，占空比数据(PERIODn、CMPn、CMPDn)
0= 写0无效

Bit6 PWMCNTE: PWM计数器使能控制位
1= PWM计数器开启（PWM开始输出）
0= PWM计数器停止（软件写0则计数器停止并清掉计数器值）

Bit5 未用

Bit4 PWMFBKSW: PWM 软件刹车信号启动位
1= PWM产生软件刹车信号
0= 禁止（通过PWMCNTE位写0来清零）

Bit3 PWMFB: 刹车标志
1= 已产生刹车动作
0= 未产生刹车动作（通过PWMCNTE位写0来清零）

Bit2 CNTTYPE: PWM计数对齐方式选择位
1= 中心对齐方式
0= 边沿对齐方式

Bit1~Bit0 CLKDIV[1:0]: PWM时钟分频
00 $F_{HSI}/1$
01 $F_{HSI}/2$
10 $F_{HSI}/4$
11 $F_{HSI}/8$

PWM 输出使能控制寄存器 PWMCON1

| 0xC9 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PWMCON1 | -- | -- | PWM5OE | PWM4OE | PWM3OE | PWM2OE | PWM1OE | PWM0OE |
| 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 未用

Bit5 PWM5OE: PWM通道5的输出使能位
1= 使能
0= 禁止

Bit4 PWM4OE: PWM通道4的输出使能位
1= 使能
0= 禁止

Bit3 PWM3OE: PWM通道3的输出使能位
1= 使能
0= 禁止

Bit2 PWM2OE: PWM通道2的输出使能位
1= 使能
0= 禁止

Bit1 PWM1OE: PWM通道1的输出使能位
1= 使能
0= 禁止

Bit0 PWM0OE: PWM通道0的输出使能位
1= 使能
0= 禁止

PWM 输出极性控制寄存器 PWMCON2

| 0xC1 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PWMCON2 | -- | -- | PWM5DIR | PWM4DIR | PWM3DIR | PWM2DIR | PWM1DIR | PWM0DIR |
| 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 未用

Bit5~ Bit0 PWMnDIR: PWM通道n输出极性控制位（n=0-5）
1= 反向输出
0= 正常输出

PWM 周期数据寄存器低 8 位 PWMTL

| 0xCE | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PWMTL | PWMTL7 | PWMTL6 | PWMTL5 | PWMTL4 | PWMTL3 | PWMTL2 | PWMTL1 | PWMTL0 |
| 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　　PWMPL<7:0>:　PWM周期数据寄存器低8位

PWM 周期数据寄存器高 8 位 PWMTH

| 0xCF | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PWMTH | | | | | | | PWMTL1 | PWMTL0 |
| 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　　PWMPH<7:0>:　PWM周期数据寄存器高2位

PWM 占空比寄存器低 8 位 PWMDnL(n=01、23、45)

| | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PWMDnL | PWMDnL7 | PWMDnL6 | PWMDnL5 | PWMDnL4 | PWMDnL3 | PWMDnL2 | PWMDnL1 | PWMDnL0 |
| 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　　PWMDnL<7:0>:　PWM通道n比较数据（占空比数据）寄存器低8位

PWM 占空比寄存器高 8 位 PWMDnH (n=01、23、45)

| | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PWMDnH | | | | | | | PWMDnH1 | PWMDnH0 |
| 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　　PWMDnH<7:0>:　PWM通道n比较数据（占空比数据）寄存器高2位

PWM 上升沿死区延时数据寄存器 PWMDTL

| 0xCC | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PWMDTL | | | PWMDT0<5:0> | | | | | |
| 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　　未用

Bit5~Bit0　　PWMDTL<5:0>:　PWM左侧死区延时数据寄存器

PWM 下降沿死区延时数据寄存器 PWMDTR

| 0xCD | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PWMDTR | | | PWMDT1<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~Bit6 未用

Bit5~Bit0 PWMDTR<5:0>: PWM右侧死区延时数据寄存器

PWM 掩码控制寄存器 PWMMASKE

| 0xCA | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PWMMASKE | -- | -- | PWM5MASKE | PWM4MASKE | PWM3MASKE | PWM2MASKE | PWM1MASKE | PWM0MASKE |
| 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 未用

Bit5~Bit0 PWMnMASKE: PWM通道n掩码控制使能位 (n=0-5)

1= PWMn通道使能掩码数据输出

0= PWMn通道禁止掩码数据输出（正常输出PWM波形）

PWM 掩码数据寄存器 PWMMASKD

| 0xCB | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PWMMASKD | -- | -- | PWM5MASKD | PWM4MASKD | PWM3MASKD | PWM2MASKD | PWM1MASKD | PWM0MASKD |
| 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 未用

Bit5~Bit0 PWMnMASKD: PWM通道n掩码数据位 (n=0-5)

1= PWMn通道输出高

0= PWMn通道输出低

PWM 刹车数据寄存器 PWMFBKD

| 0xD1 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PWMFBKD | -- | -- | PWM5FBKD | PWM4FBKD | PWM3FBKD | PWM2FBKD | PWM1FBKD | PWM0FBKD |
| 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 未用

Bit5~Bit0 PWMnFBKD: PWM通道n刹车数据位 (n=0-5)

1= PWMn通道产生刹车操作后输出高

0= PWMn通道产生刹车操作后输出低

## 15.4 增强型 PWM 操作

### 15.4.1 加载更新模式

由于 PWM 存在双缓存结构，在 PWM 运行的过程中，改变相关运行寄存器：PWMTL/PWMTH/PWMDnL/PWMDnH 的值，PWM 输出波形不会立即改变，只有在零点时这些寄存器的值才会加载到相应的缓存中。这样的结构在改变周期占空比数据后，不会立即改变当前 PWM 周期的输出波形，在下个周期 PWM 波形才会做出相应的变化。即任何 PWM 相关数据的改变不会影响当前一个完整 PWM 周期。

在高速的应用中，有可能会出现加载点已经到来，但写入运行寄存器的操作还未完成。此时不期望出现部分运行数据已经加载，另外一部分运行数据没有加载的情况。针对该高速应用情况，PWM 模块提供了加载使能位。

当改变相关运行寄存器后，需要将加载寄存器 PWMCON0 的使能位 PWMLE 置 1，周期和所有通道的占空比进行加载，加载完毕后 PWMLE 位自动清零。即可以通过读取该位来判断是否将相关寄存器的值加载到实际电路中。如果 PWMLE=0，则表示已经加载，将影响正在输出的 PWM 波形；如果 PWMLE=1，则表示还未加载，当前的 PWM 波形还未发生变化，将在下一个加载点加载之前改变的寄存器的值。如果再次改变相关运行寄存器的值，也需重新将 PWMLE=1 置 1。

注：当 PWMLE=1 时，对周期和占空比寄存器内容的更改，可能引发无法预测的结果。
建议先更改周期和占空比寄存器内容，再将加载使能位 PWMLE 置 1，最后等待加载完成（PWMLE=0）。

### 15.4.2 边沿对齐模式

边沿对齐模式下，PWM 计数器采用向上计数模式（Up count）：10 位 PWM 计数器 CNTn 的初始值为 0，以此开始向上计数直至计数值变为 PWMTH/PWMTL，此时 MCU 自动将计数器清零继而开始下一个 PWM 周期的计数。

当 CNTn 的值与占空比寄存器 PWMDnH/PWMDnL 的值相等时，PWMn 输出低电平；CNTn 继续向上计数至 PWMTH/PWMTL，此时 PWMn 将输出高电平（PWM 选择为反相输出时，输出电平正好与上述描述相反）。

边沿对齐相关参数如下：

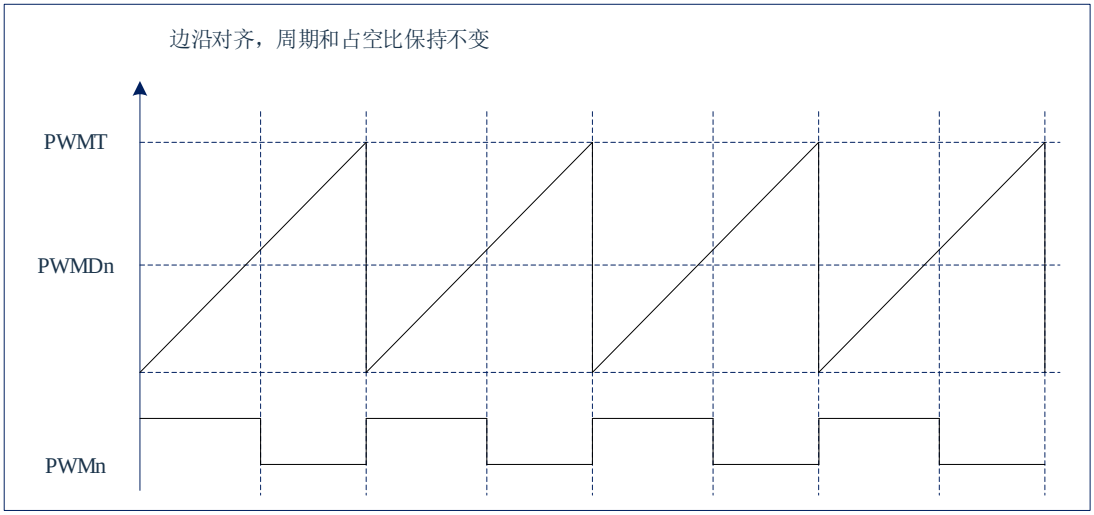
$$\text{高电平时间} = (\text{(PWMDnH:PWMDnL)}+1) \times T_{pwm}$$

$$\text{周期} = (\text{(PWMTH:PWMTL)}+1) \times T_{pwm}$$

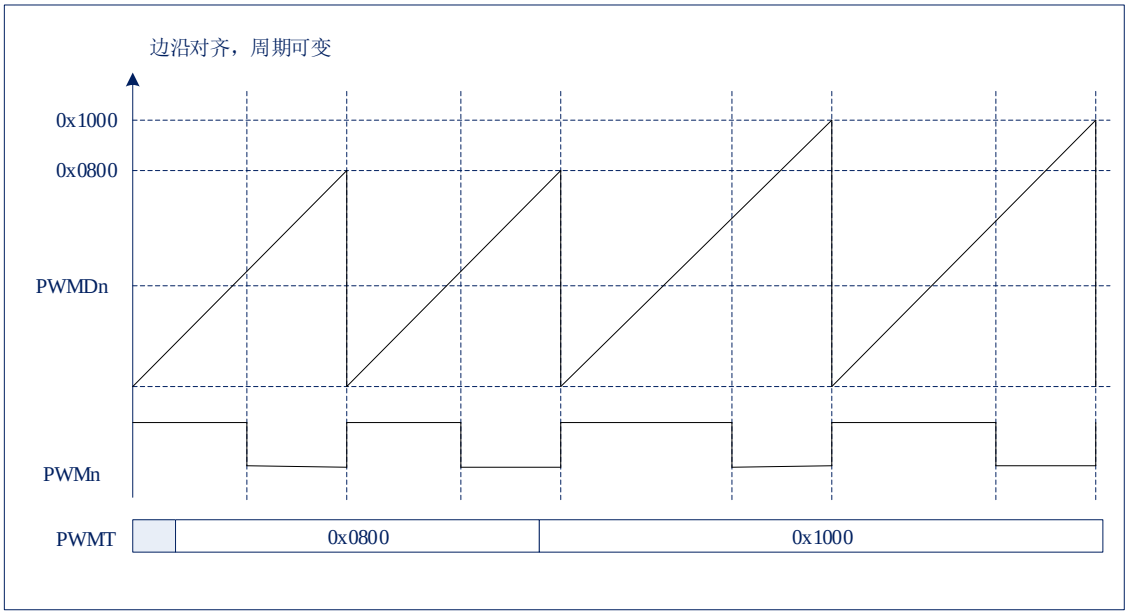
$$\text{占空比} = \frac{\text{(PWMDnH:PWMDnL)}+1}{\text{(PWMTH:PWMTL)}+1}$$

PWMDn=0 时，占空比为 0%。

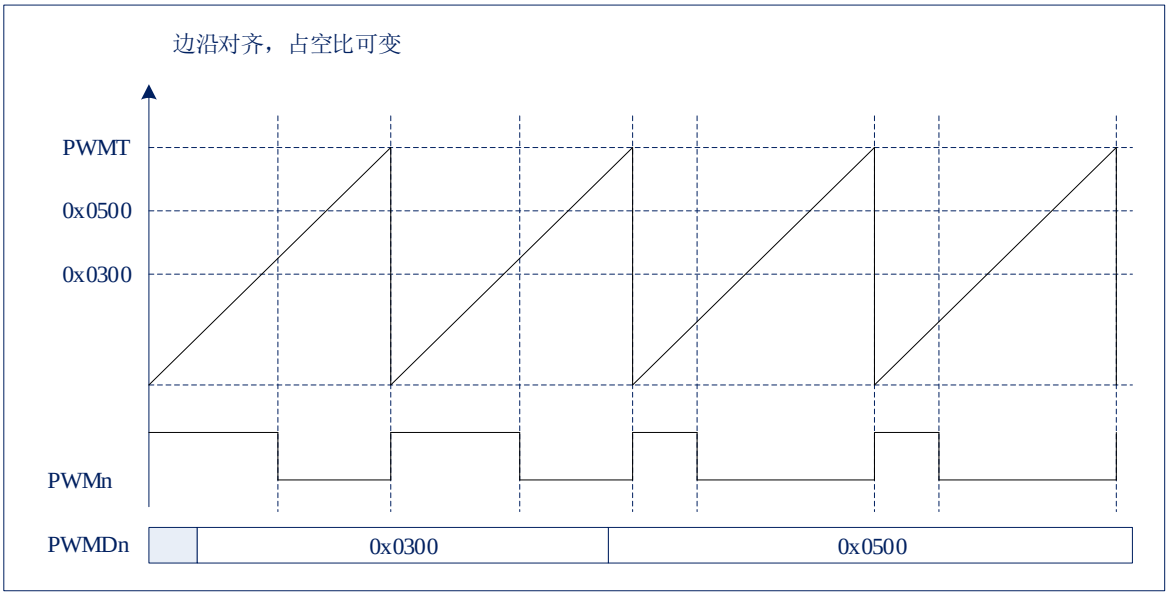
边沿对齐，周期和占空比不变的时序图如下图所示：



边沿对齐，周期变化时序图如下图所示：




边沿对齐，占空比变化时序图如下图所示：



### 15.4.3 中心对齐模式

#### 15.4.3.1 对称计数

中心对齐对称计数模式下，PWM 计数器采用上下计数模式（Up-Down count），10 位 PWM 计数器 CNTn 从 0 开始向上计数，当 CNTn = PWMTH:PWMDL 后又自动开始向上计数直到 0，后续的 PWM 周期重复这样的计数操作。

在向上计数边沿，当 CNTn 的值与占空比寄存器 PWMDn 的值相等时，PWMn 的电平发生翻转，变成高电平；在向下计数边沿，当 CNTn 的值与占空比寄存器 PWMDn 的值相等时，PWMn 的输出电平发生翻转，变成低电平（PWM 选择为反相输出时，输出电平正好与上述描述相反）。

对称计数下相关参数如下：

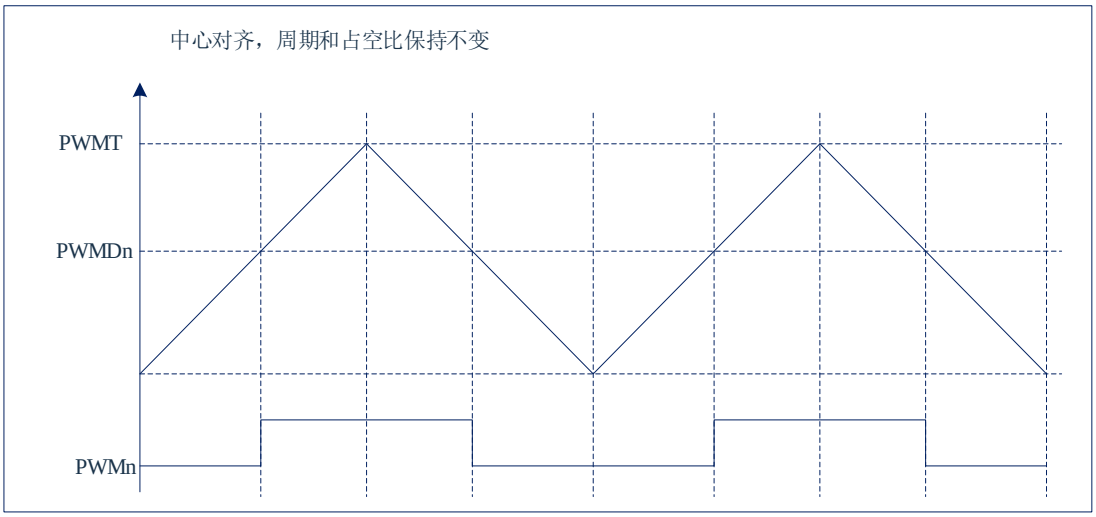
$$\text{高电平时间} = ((\text{PWMTH:PWMTL})\times 2-(\text{PWMDnH:PWMDnL})\times 2)\times \text{Tpwm}$$

$$\text{周期} = (\text{PWMTH:PWMTL})\times 2\times \text{Tpwm}$$

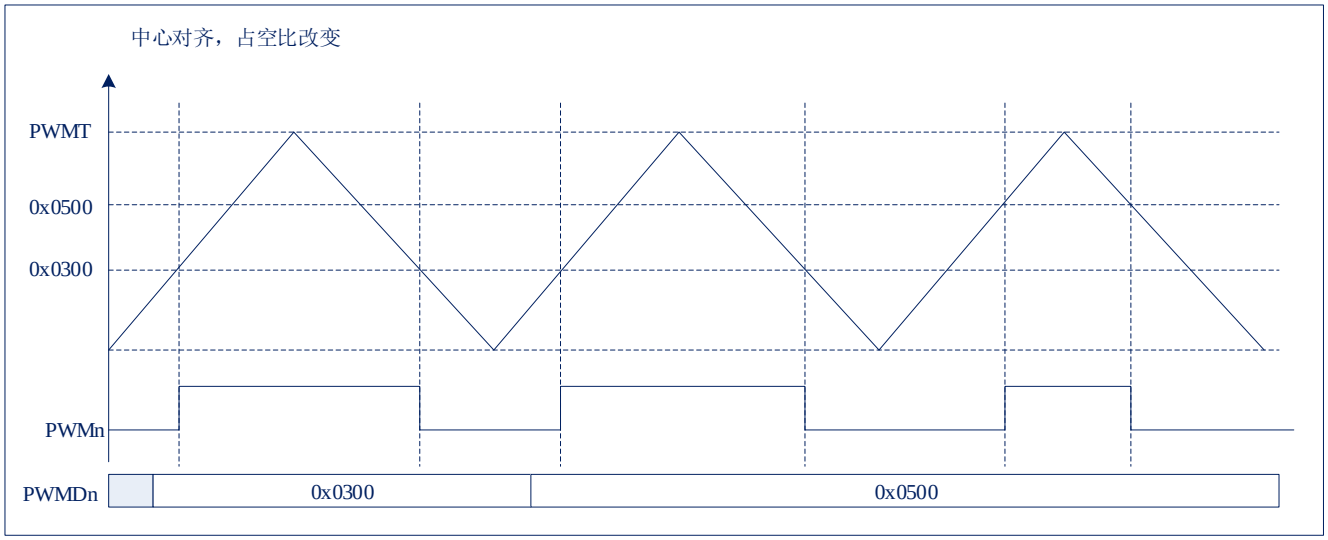
$$\text{占空比} = \frac{(\text{PWMTH:PWMTL})\times 2-(\text{PWMDnH:PWMDnL})\times 2}{(\text{PWMTH:PWMTL})\times 2}$$

PWMDn=0 时，占空比为 100%。

中心对齐对称计数，周期和占空比不变的时序图如下图所示：




中心对齐对称计数，占空比改变的时序图如下图所示：

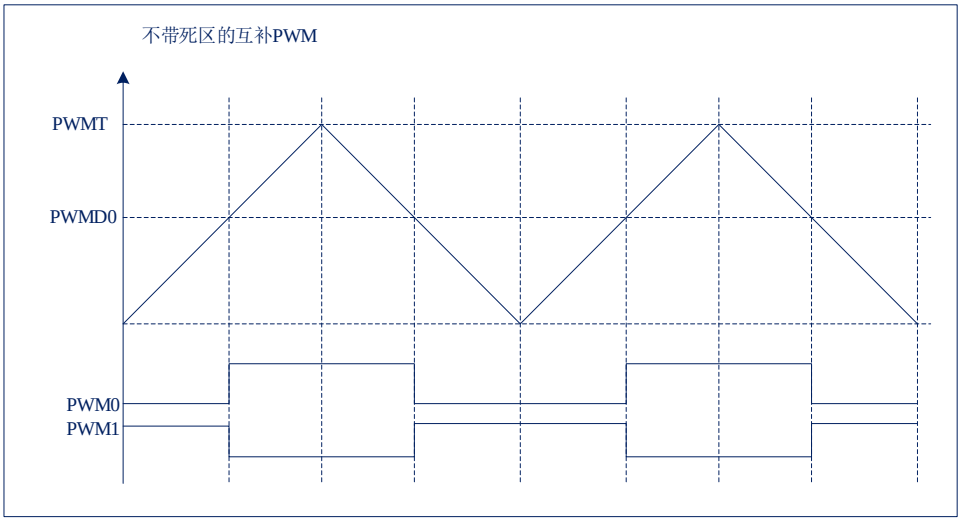


### 15.4.4 带死区的互补模式

在实际的电机控制应用中，用于驱动逆变桥的 PWM 信号需要具备互补输出的模式，即上桥臂的驱动信号正好和下桥臂的驱动信号反相。

在增强型 PWM 模块中，6 通道 PWM 可设置为 3 对互补信号：PWM0 和 PWM1，PWM2 和 PWM3，PWM4 和 PWM5，PWM1，PWM3，PWM5 的周期与占空比分别由 PWM0，PWM2，PWM4 相关寄存器决定。

不带死区的互补模式时序图如下图所示：




在电机控制应用当中，理想的 PWM 信号是在同一时刻发生电平的翻转，由于 MOS 管的开通和关断存在着延时，这样就容易造成电源对地直通，从而损坏功率管。为了避免这种现象，带死区时间的 PWM 就显得尤为重要。在互补模式下，每组互补 PWM 对均支持插入死区时间，插入的死区时间如下：

PWMn 上升沿死区时间：（PWMDTL+1）*TPWM

PWMn 下降沿死区时间：（PWMDTR+1）*TPWM

PWM(n+1)上升沿死区时间：（PWMDTR+1）*TPWM

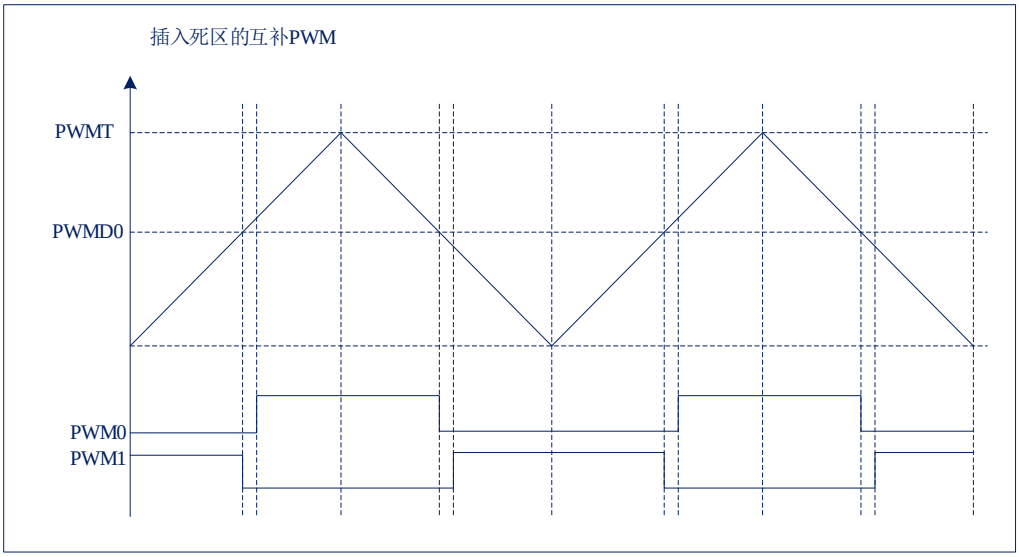
PWM(n+1)下降沿死区时间：（PWMDTL+1）*TPWM

n 为 0、2 和 4 通道

TPWM 为 PWM 的时钟源周期。

| 注：中心对齐与边沿对齐均支持互补模式。 |
|---|

插入死区的互补 PWM 波形如下图所示：


插入死区的互补PWM
PWMT
PWMD0
PWM0
PWM1

### 15.4.5 刹车功能

可触发 PWM 刹车的信号源有：

- 比较器 0 的输出。

触发刹车后，刹车标志位 PWMFB 置 1，所有通道的计数器使能位清零，且 PWM 输出预设的刹车数据。

如需恢复正常输出，则需要将刹车标志清零，并且将 PWM 通道计数器重新使能（刹车相关的配置请参见刹车控制寄存器 PWMFBKC 和刹车数据寄存器 PWMFBKD 的说明）。

### 15.4.6 中断功能

增强型 PWM 总共具有 2 个中断标志，其中 1 个周期中断标志，1 个零点中断标志，中断标志位的产生与对应中断使能位是否开启无关。开启 PWM 任何一种类型的中断均需打开全局中断使能位（EA=1）、PWM 总中断使能位 PWMIE，才能成功配置 PWM 中断功能。PWM 的所有中断共用一个中断向量入口，故进入中断服务程序后用户可通过中断标志位判断是哪种类型中断产生。

对于中心对齐方式和边沿对齐方式，有 2 种中断类型：零点中断，周期中断。其中周期中断和零点中断相同。

如边沿对齐模式的中断信号时序图如下图所示：

# 16. 模拟比较器（ACMP0/1）

## 16.1 概述

芯片内部包含两个模拟比较器。可按照比较器的配置适用于不同的应用场合。当正端电压大于负端电压时，比较器输出逻辑 1，反之输出 0，也可以通过输出极性选择位进行改变。当比较器输出值发生改变时，每路比较器都可通过配置产生中断。

## 16.2 结构框图

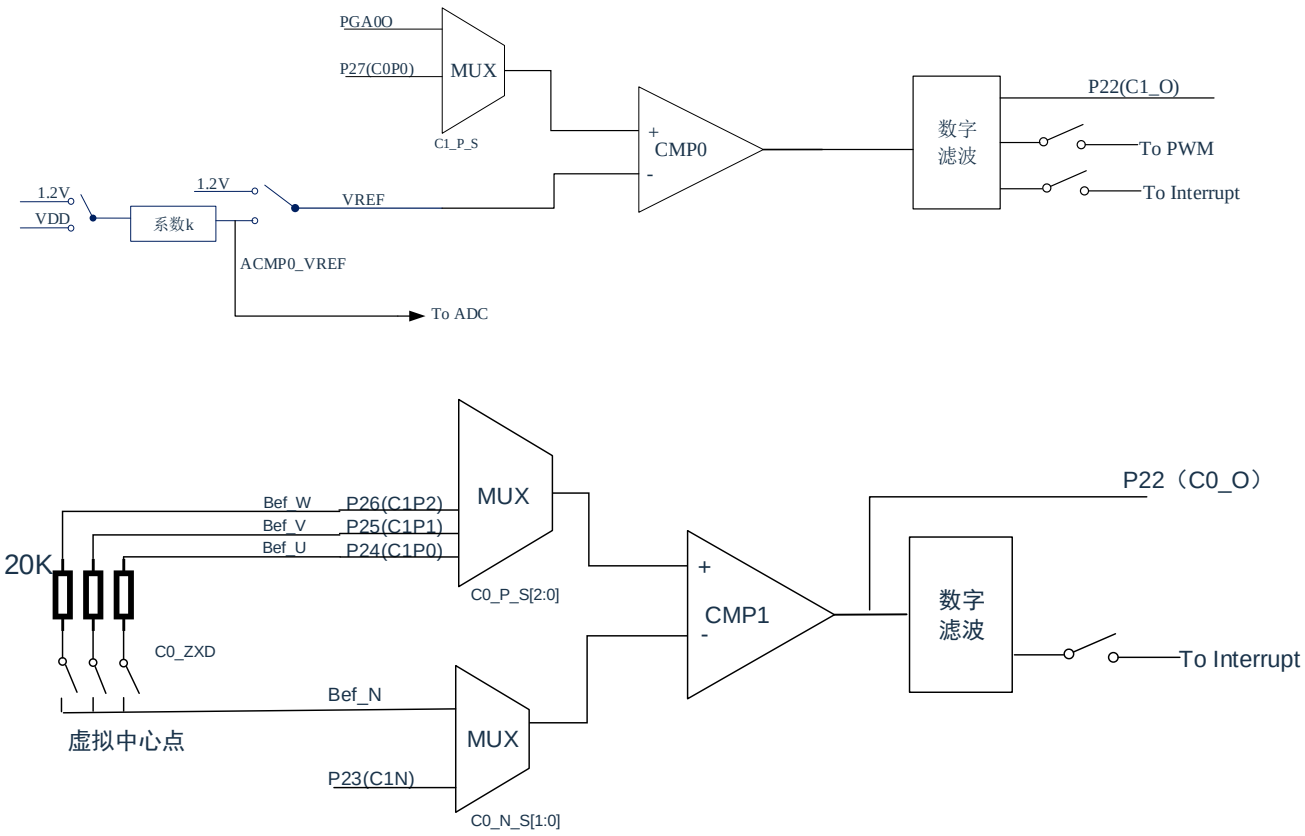



图 16-1：比较器结构框图

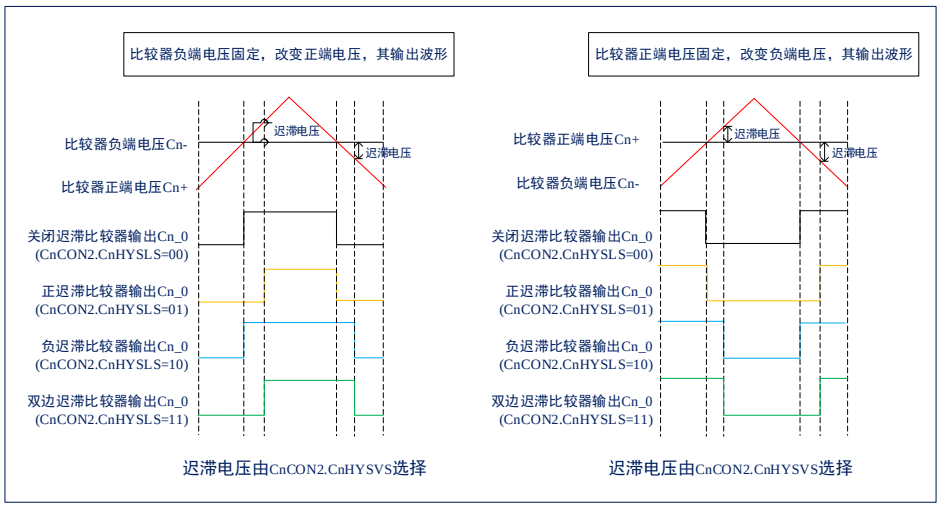



图16-1：比较器迟滞功能结构框图

## 16.3 特性

- 模拟输入电压范围：(0~VDD)V。
- 支持单边/双边迟滞功能。
- 支持迟滞电压选择(0/±10/±20/±60mV)
- 输出可滤波时间可选择：0~512*Tsys。
- 比较器 0 事件输出可作为增强型 PWM 的刹车触发信号。
- 输出改变可产生中断。
- 比较器 0 负端输入可选择 16 级可调的参考电压
- 比较器 1 支持虚拟中心点输入

## 16.4 功能说明

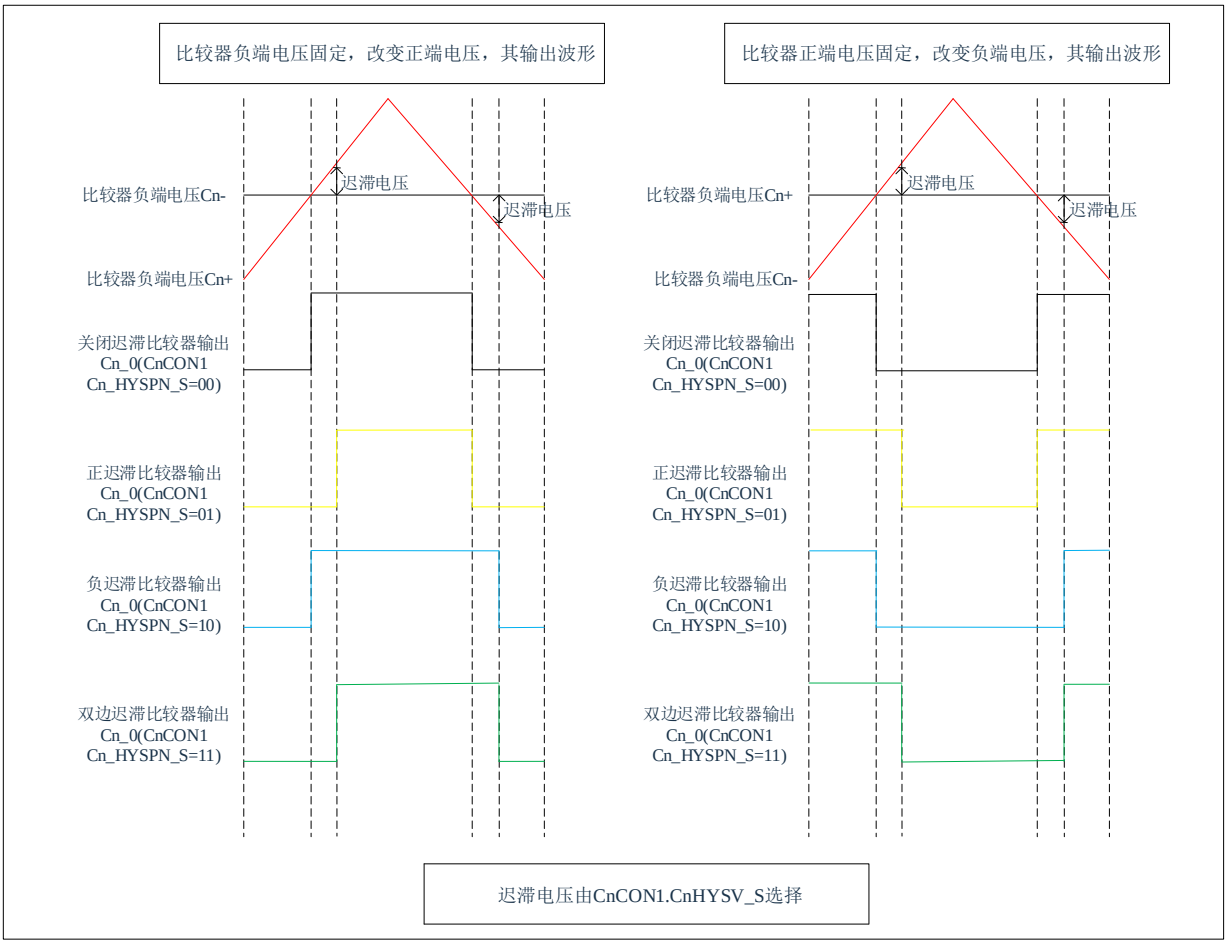



图 16-2：比较器迟滞功能结构框图

## 16.5 相关寄存器说明

ACMP0 控制寄存器 ACMP0CON0

| 0xE8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ACMP0CON0 | C0_OUT | -- | C0_OEN | -- | -- | -- | C0_P_S | C0_EN |
| R/W | R | 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 C0_OUT 模拟比较器0结果位(只读)

BIT6 未用

BIT5 C0_OEN 模拟比较器0输出使能位

0: 禁止

1: 使能

BIT4~ BIT2 未用

BIT1 C0_P_S 模拟比较器0正端通道选择

0: C0P0

1: PGAO

BIT0 C0_EN 模拟比较器0使能位

0: 禁止

1: 使能

ACMP0 控制寄存器 ACMP0CON1

| 0xE9 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ACMP0CON1 | -- | -- | C0_HYSPN_S | | C0_HYSV_S | | C0_POS | -- |
| 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 未用 未用

BIT5~BIT4 C0_HYSPN_S 模拟比较器0迟滞方式选择

00: 无迟滞

01: 正迟滞

10: 负迟滞

11: 正负迟滞

BIT3~BIT2 C0_HYSV_S 模拟比较器0迟滞电压选择

00: 无迟滞

01: 10mV

10: 20mV

11: 60mV

BIT1 C0_POS 模拟比较器0输出极性选择位

0: 正常输出

1: 反相输出

BIT0 未用

ACMP1 控制寄存器 ACMP1CON0

| 0xD8 | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ACMP1CON0 | C1_OUT | C1_ZXD | C1_OEN | -- | C1_N_S | C1_P_S | | C1_EN |
| R/W | R | 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 C1_OUT 模拟比较器1结果位(只读)

BIT6 C1_ZXD 模拟比较器1中心点选择使能位

0: 禁止

1: 使能

BIT5 C1_OEN 模拟比较器1输出使能位

0: 禁止

1: 使能

BIT4 未用 未用

BIT3 C1_N_S 模拟比较器1负端通道选择

0: C1N

1: Bef_N虚拟中心点(比较器内部信号)

BIT2~BIT1 C1_P_S 模拟比较器1正端通道选择

00: C1P0

01: C1P1

10/11: C1P2

BIT0 C1_EN 模拟比较器1使能位

0: 禁止

1: 使能

ACMP1 控制寄存器 ACMP1CON1

| 0xEA | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ACMP1CON1 | -- | -- | C1_HYSPN_S | | C1_HYSV_S | | C1_POS | -- |
| 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 未用 未用

BIT5~BIT4 C1_HYSPN_S 模拟比较器1迟滞方式选择

00: 无迟滞

01: 正迟滞

10: 负迟滞

11: 正负迟滞

BIT3~BIT2 C1_HYSV_S 模拟比较器1迟滞电压选择

00: 无迟滞

01: 10mV

10: 20mV

11: 60mV

BIT1 C1_POS 模拟比较器1输出极性选择位

0: 正常输出

1: 反相输出

BIT0 未用

ACMP1 控制寄存器 ACMPFLT

| 0xEB | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ACMP1CON2 | --- | C1_FS | | | --- | C0_FS | | |
| 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 | 未用 | |
| BIT6~BIT4 | C1_FS | 模拟比较器1输出滤波时间选择位 |
| | 000: | (0~1)*Tpclk |
| | 001: | (1~2)*Tpclk |
| | 010: | (2~3)*Tpclk |
| | 011: | (4~5)*Tpclk |
| | 100: | (8~9)*Tpclk |
| | 101: | (16~17)*Tpclk |
| | 110: | (32~33)*Tpclk |
| | 111: | (64~65)*Tpclk |
| BIT3 | 未用 | |
| BIT2~BIT0 | C0_FS | 模拟比较器0输出滤波时间选择位 |
| | 000: | (0~1)*Tpclk |
| | 001: | (1~2)*Tpclk |
| | 010: | (2~3)*Tpclk |
| | 011: | (4~5)*Tpclk |
| | 100: | (8~9)*Tpclk |
| | 101: | (16~17)*Tpclk |
| | 110: | (32~33)*Tpclk |
| | 111: | (64~65)*Tpclk |

ACMP 事件控制寄存器 ACMPEVCON

| 0xEC | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ACMPEVCON | -- | -- | -- | EVE0 | EVS1 | | EVS0 | |
| 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 未用

BIT4 EVE0 模拟比较器0输出刹车使能位(不影响中断产生)

0: 禁止

1: 使能

BIT3~BIT2 EVS1 模拟比较器 1 中断边沿选择

00: 上升沿

01: 下降沿

1x: 双沿

BIT1~BIT0 EVS0 模拟比较器 0 中断边沿选择

00: 上升沿

01: 下降沿

1x: 双沿

ACMP0 参考电压控制寄存器 ACMP0VRCON

| 0xED | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| ACMP0VRCON | -- | -- | ACMPDIVS | ACMPSVR | ACMPVS<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 未用

BIT5 ACMPDIVS ACMP_VREF参考源选择位

0: 选择VDD进行分压

1: 选择BG(1.2V)进行分压

BIT4 ACMPSVR 比较器0负端电压内部电压VREF选择位

0= 选择BG(1.2V)

1= 选择ACMP_VREF(分压电路开启，独立于比较器模块)

BIT3~BIT0 ACMPVS<3:0> ACMP_VREF参考源分压系数k选择位

0000~1111= 2/20~17/20

# 17. 可编程增益放大器

## 17.1 特性

**PGA (可编程增益放大器)**

- 增益可调节：4/8/12/16。
- 支持伪差分结构，反馈地可选择从外部端口接入。
- PGA0 可选择多种输出方式：
  （1） 可直接输出到 ADC 通道
  （2） 可直接输出到 PAD（A0O）
  （3） 可通过 10K 电阻后输出到 PAD（A0O）

## 17.2 结构框图

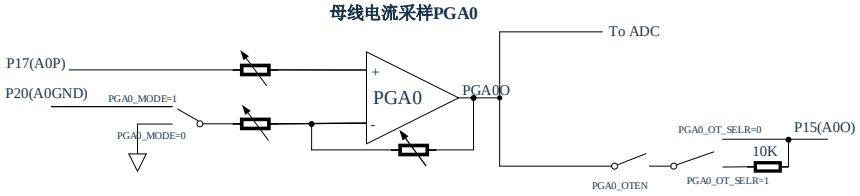



图 17-3：PGA 结构图

## 17.3 寄存器说明

PGA 控制寄存器 PGACON0

| 0xEE | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| PGACON0 | PGA_EN | PGA_OTEN | PGA_MODE | --- | PGA_OT_SELR | --- | PGA_S | |
| R/W | R | R | R | --- | R | --- | R | R |
| 复位值 | 0 | 0 | 0 | --- | 0 | --- | 0 | 0 |

| | | |
|---|---|---|
| BIT7 | PGA_EN | PGA使能位 |
| | 0: | 禁止 |
| | 1: | 使能 |
| BIT6 | PGA_OTEN | PGA输出到PAD通道使能 |
| | 0: | 禁止 |
| | 1: | 使能 |
| BIT5 | PGA_MODE | PGA模式选择 |
| | 0: | 单端模式 |
| | 1: | 伪差分模式 |
| BIT4 | 未用 | |
| BIT3 | PGA_OT_SELR | PGA0输出到PAD串联电阻选择 |
| | 0: | 内部不串联电阻 |
| | 1: | 内部串联10K电阻 |
| BIT2 | 未用 | |
| BIT1~BIT0 | PGA_S | PGA增益选择 |
| | 00: | 4X |
| | 01: | 8X |
| | 10: | 12X |
| | 11: | 16X |

# 18. Flash 存储器

## 18.1 概述

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

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

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

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

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

| 注：程序 EEPROM 正常写电压范围为 2.5V~5.5V，写电流为 6mA@VDD=5V. |
|---|

## 18.2 相关寄存器

### 18.2.1 Flash 存储器控制寄存器

Flash 存储器控制寄存器 EECON1

| 0xFD | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|
| EECON1 | EEPGD[1] | EEPGD[0] | --- | SERIOK | WRERR | WREN | WR | RD |
| R/W | R/W | R/W | --- | R | R/W | R/W | R/W1 | R/W1 |
| 复位值 | 0 | 0 | --- | 0 | X | 0 | 0 | 0 |

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

EECON1 控制位 EEPGD[1:0]决定访问的是程序存储器，UID 还是数据存储器。EEPGD[1:0]=00 时，任何后续操作都将针对数据存储器进行。EEPGD[1:0]=01 时，任何后续操作都将针对 UID 区域进行。EEPGD[1:0]=1x 时，任何后续操作都将针对程序存储器进行。

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

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

Flash 存储器控制寄存器 EECON2

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

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

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

Flash 存储器控制寄存器 EECON3

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

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

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

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

EECON3 控制位 OP 选择页操作的实际动作（擦除/烧写）

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

### 18.2.2 Flash 存储器地址寄存器

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

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

程序存储器读写地址设置示例：

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

当选择数据存储器地址值时，只将地址的低字节写入 EEADR 寄存器。

Flash 存储器地址寄存器 EEADR

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

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

Flash 存储器地址寄存器 EEADRH

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

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

### 18.2.4 Flash 存储器数据寄存器

Flash 存储器数据寄存器 EEDAT

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

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

Flash 存储器数据寄存器 EEDATH

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

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

## 18.3 读 UID

每颗芯片拥有不同 96 位唯一身份识别号，即唯一 ID（Unique identification）。出厂时已经设置，用户不能修改。UID 的数据放在内部 CONFIG 区域，地址为 0x20-0x25 对应的 6 个字。

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

读 UID 数据流程示例：

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

## 18.4 读 Flash 存储器

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

读程序/数据存储器数据流程示例：

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

## 18.5 写 Flash 存储器

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

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

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

### 18.5.1 字写

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

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

写数据存储器流程示例：

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

写程序存储器流程示例：

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

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

### 18.5.2 页操作

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

#### 18.5.2.1 清缓存

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

1. 选择操作存储器的类型，向 EECON2 写入特定序列后，使能 EECON3 控制位 HVPLEN 以使能缓存，此时硬件会自动启动清缓存操作，并使能硬件计数器。
2. 保持 EECON3 控制位 HVPLEN 为 1，在 Flash 存储器页擦除或页烧写结束前，硬件均会自动启动清缓存操作和重置硬件计数器；

#### 18.5.2.2 写缓存

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

#### 18.5.2.3 页擦除

Flash 存储器的页擦除操作是在清缓存和写缓存后，由 EEADRH<3:0>和 EEADR<7:4>组合选择页，通过 EECON3 选择位 OP 选择擦除，并使能 EECON3 控制位 START 启动，擦除结束后 START 自动清零。页擦除结束前，硬件会自动启动清缓存操作和重置硬件计数器。

#### 18.5.2.4 页烧写

Flash 存储器的页烧写操作是，在页擦除和写缓存后，由 EEADRH<3:0>和 EEADR<7:4>组合选择页，通过 EECON3 选择位 OP 选择烧写，并使能 EECON3 控制位 START 启动，烧写结束后 START 自动清零。页烧写结束前，硬件会自动启动清缓存操作和重置硬件计数器。

页操作程序存储器的流程示例：

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

**注意：当系统配置中的程序存储器写保护位有效时，对该程序存储器区间无法启动页擦除/页烧写操作；**

## 18.6 Flash 存储器操作注意事项

### 18.6.1 关于 Flash 存储器的操作时间

Flash 存储器的操作时间是由内部定时器控制的，具体时间如下表所示：

| 操作类型 | 定时器时钟 | 操作时间 | 单位 |
|---|---|---|---|
| 清缓存 | $F_{8M}$=8MHz | 0.75 | us |
| 写缓存（单次） | | 1.25 | us |
| 页擦除 | | 2.82 | ms |
| 页烧写 | | 1.80 | ms |
| 字写 | | 4.62 | ms |

由上述时间关系可以看出，字写和单次页写所需要的总时间几乎是一样的。

### 18.6.2 写校验

根据具体的应用，一般要求进行软件写校验。

### 18.6.3 避免误写的保护

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

# 19. 指令

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

## 19.1 符号说明

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

## 19.2 指令一览表

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

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

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

# 20. 版本修订说明

| 版本号 | 时间 | 修改内容 |
|---|---|---|
| V1.0.0 | 2025 年 8 月 | 正式版本<br>1) 新增读 EE 操作需要加 6 个 NOP 在判断 RD 位是否清零<br>2) 新增 USART 背靠背注意事项的 C 语言示例 |
| V1.0.1 | 2025 年 9 月 | 1) 订正 USART 波特率发生器章节里 SPBRG 计算公式<br>2) 订正捕捉相关寄存器里寄存器 CAPDL 和 CAPDH 的错误描述 |