SN8P2524

用户参考手册

Version 1.1

SN8P2524

SONiX 8 位单片机

修改记录

版本	时间	修改说明
VER 1.0	2011.12	第一版。
VER 1.1	2012.09	1、用 SSOP24 的封装替代 SSOP28 的封装。 2、调整部分 IO 引脚的配置和 IO 章节的部分内容。

目录

1 产品简介

1.1 功能特性

- **存储器配置**

  ROM：2K * 16 位。

  RAM：256 * 8 位。

- **8 层堆栈缓存器**

- **6 个中断源**
  5 个内部中断：T0，TC1，CM0，SIO，MSP。
  1 个外部中断：INT0。

- **I/O 引脚配置**
  双向输入输出端口：P0，P1，P5。
  具有唤醒功能的端口：P0，P1 电平变换。
  具有上拉电阻的端口：P0，P1，P5。
  可编程的开漏引脚：P5.0~P5.2。
  比较器输入引脚：CM0N0~CM0N15。
  比较器输出引脚：CM0O。

- **Fcpu（指令周期）**
  Fcpu = Fpsc/1，Fpsc/2，Fosc/4，Fosc/8，Fosc/16。

- **功能强大的指令集**
  指令的长度为 1 个字长。
  大多数指令只需要一个周期。
  JMP/CALL 指令可寻址整个 ROM 区。
  查表指令 MOVC 可寻址整个 ROM 区。

- **1 个 8 位基本定时器 T0**
- **1 个 8 位定时器 TC1，具有占空比/周期可编程控制的 PWM 功能**
- **8 通道 LED PWM 驱动**
- **16 通道比较器**
- **SIO 串行输入/输出接口**
- **MSP 从动模式接口**
- **内置看门狗定时器，时钟源由内部低速 RC 时钟提供（16KHz @3V, 32KHz @5V）**

- **2 种系统时钟**
  内部高速时钟：RC，16MHz。
  内部低速时钟：RC，16KHz（3V），32KHz（5V）。

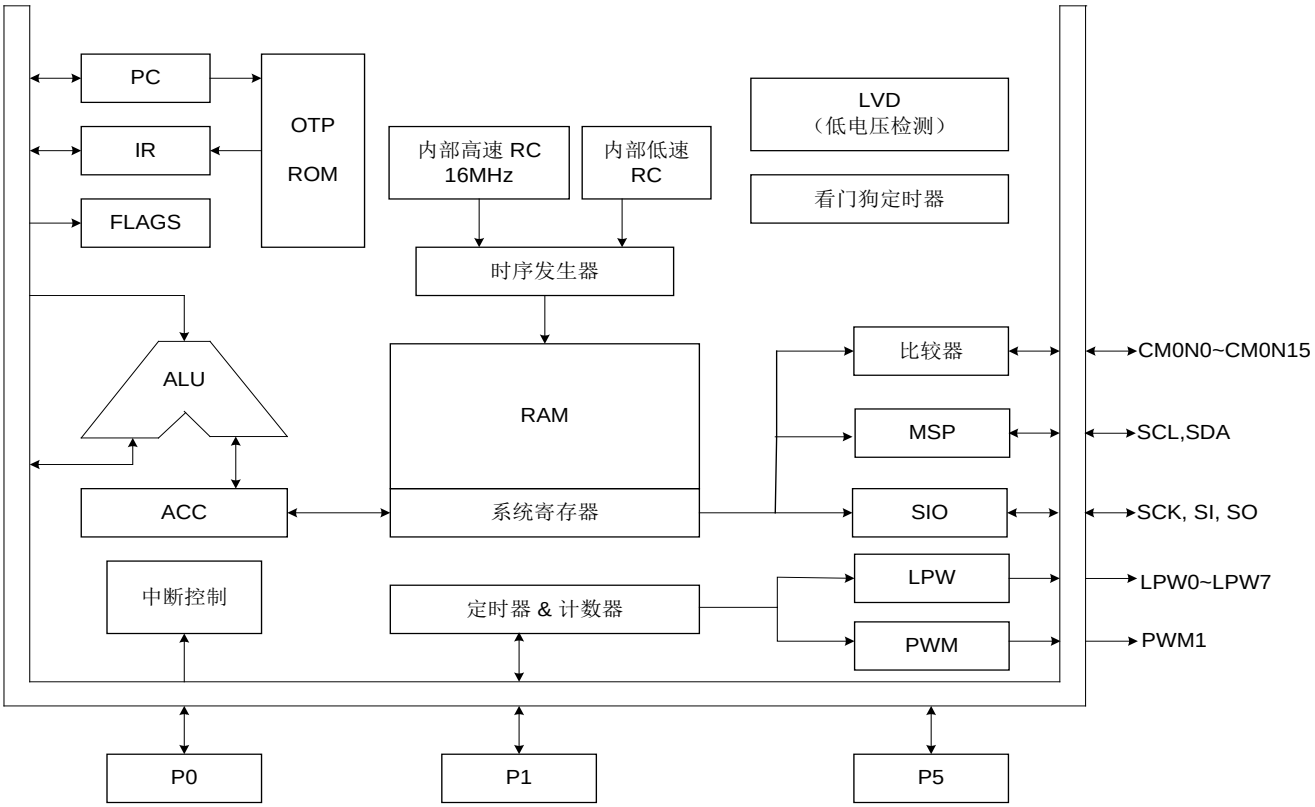
- **4 种工作模式**
  普通模式：高低速时钟正常工作。
  低速模式：仅低速时钟工作。
  睡眠模式：高低速时钟都停止工作。
  绿色模式：由定时器周期性的唤醒。

- **封装形式**
  SKDIP 24 pin
  SOP 24 pin
  SSOP 24 pin

☞ **产品性能列表**

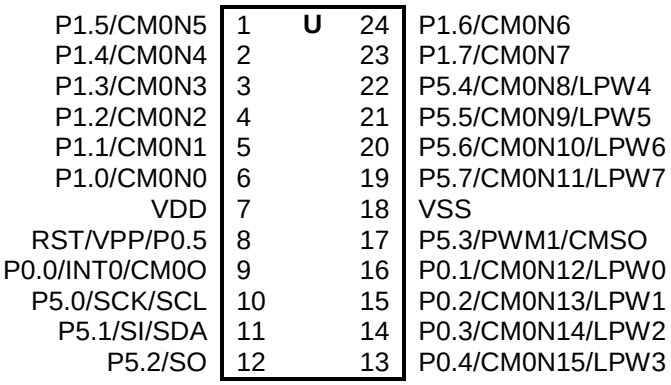
单片机名称	ROM	RAM	堆栈	定时器				SIO	MSP	I/O	比较器	PWM	唤醒功能引脚数目	封装形式
				T0	TC0	TC1	T1							
SN8P2522	2K*16	128	8	V	V	V	V	V	-	16	8-ch	2	9	DIP18/SOP18/SSOP20
SN8P2523	2K*16	256	8	V	-	V	-	V	V	22	12-ch	8+1	14	SKDIP24/SOP24/SSOP28
SN8P25231	2K*16	256	8	V	-	V	-	V	V	16	8-ch	2+1	12	DIP18/SOP18/SSOP20
SN8P2524	2K*16	256	8	V	-	V	-	V	V	22	16-ch	8+1	10	SKDIP24/SOP24/SSOP24

1.2 系统框图


PC
IR
FLAGS
OTP
ROM
内部高速 RC
16MHz
内部低速
RC
LVD
（低电压检测）
看门狗定时器
时序发生器
ALU
RAM
比较器
MSP
SIO
LPW
PWM
CM0N0~CM0N15
SCL,SDA
SCK, SI, SO
LPW0~LPW7
PWM1
ACC
系统寄存器
中断控制
定时器 & 计数器
P0
P1
P5

1.3 引脚配置

SN8P2524K (SKDIP 24 pins)
SN8P2524S (SOP 24 pins)
SN8P2524X (SSOP 24 pins)



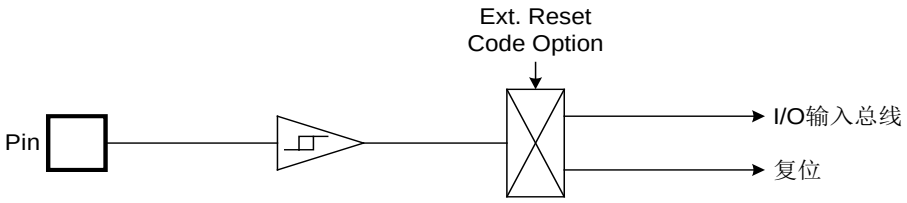
SN8P2524K
SN8P2524S
SN8P2524X

1.4 引脚说明

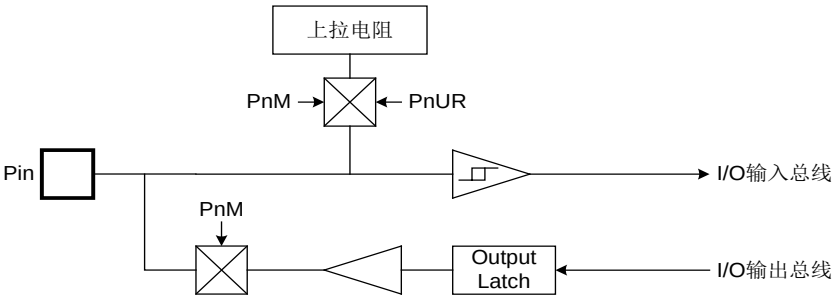
引脚名称	类型	功能说明
VDD, VSS	P	电源输入端。
P0.5/RST/VPP	I, P	RST：系统复位引脚，施密特触发，低电平有效，通常保持高电平。
		VPP：OTP 12.3V 烧录电源输入引脚。
		P0.5：单向输入引脚，施密特触发，具有唤醒功能，此时必须禁止外部复位功能。
P0.0/INT0/CM0O	I/O	P0.0：双向输入输出引脚，输入模式下施密特触发，内置上拉电阻，具有唤醒功能。
		INT0 触发引脚，施密特触发。
		CM0O：比较器输出引脚。
P0[4:1]/CM0N[15:12]/ LPW[3:0]	I/O	P0.1~P0.4：单向输出引脚，非施密特触发，无上拉电阻，无唤醒功能。
		CM0N[15:12]：比较器负极输入通道 12～15。
		LPW[3:0]：LED[3:0] PWM 输出引脚。
P1[7:0]/CM0N[7:0]	I/O	P1：双向输入输出引脚，输入模式下施密特触发，内置上拉电阻，具有唤醒功能。
		CM0N[7:0]：比较器负极输入通道 0～7。
P5.0/SCK/SCL	I/O	P5.0：双向输入输出引脚，输入模式下施密特触发，内置上拉电阻，可编程开漏引脚。
		SCK：SIO 时钟引脚。
		SCL：MSP 时钟引脚。
P5.1/SI/SDA	I/O	P5.1：双向输入输出引脚，输入模式下施密特触发，内置上拉电阻，可编程开漏引脚。
		SI：SIO 数据输入引脚。
		SDA：MSP 数据引脚。
P5.2/SO	I/O	P5.2：双向输入输出引脚，输入模式下施密特触发，内置上拉电阻，可编程开漏引脚。
		SO：SIO 数据输出引脚。
P5.3/PWM1/CMSO	I/O	P5.3：双向输入输出引脚，输入模式下施密特触发，内置上拉电阻。
		PWM1：PWM 输出引脚。
		CMSO：补偿输出引脚。
P5[7:4]/CM0N[11:8]/ LPW[7:4]	I/O	P5.4-5.7：双向输入输出引脚，输入模式下施密特触发，内置上拉电阻。
		CM0N[11:8]：比较器负极输入通道 8～11。
		LPW[7:4]：LED[7:4] PWM 输出引脚。

1.5 引脚电路结构图

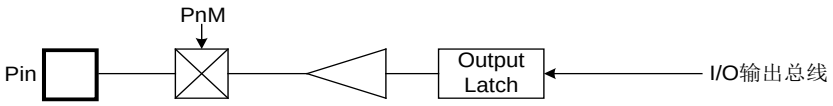
P0.5：



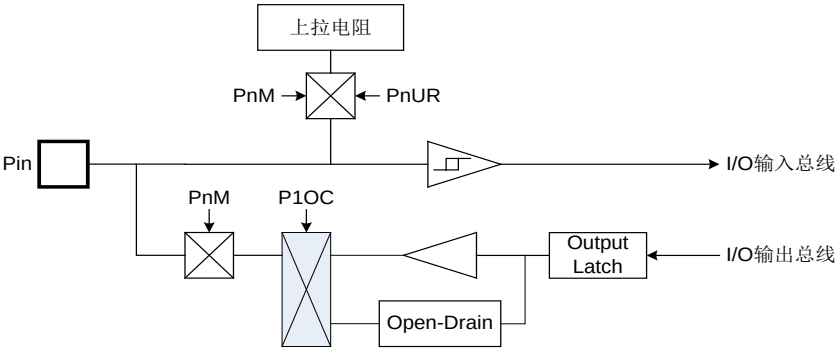
P0.0，P5.3：



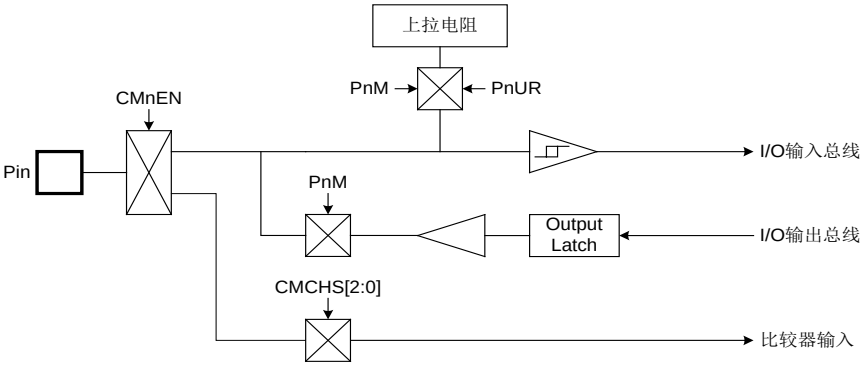
P0.1~0.4：



P5.0~ 5.2：



P1.0~ 1.7/P5.4~ 5.7：



2 中央处理器（CPU）

2.1 程序存储器（ROM）

☞ **ROM：2K**

ROM 包括复位向量，中断向量，通用存储区域和系统保留区域：复位向量是程序的开始地址；中断向量是中断服务程序的开始地址；通用存储区域则是程序存储区，包括主循环，子程序和数据表。

2.1.1 复位向量（0000H）

一个字长的系统复位向量（0000H）执行系统复位。

- **上电复位（NT0=1，NPD=0）；**
- **看门狗复位（NT0=0，NPD=0）；**
- **外部复位（NT0=1，NPD=1）。**

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 PFLAG 寄存器中的 NT0 和 NPD 标志位的内容可以判断系统复位方式。下面一段程序演示了如何定义 ROM 中的复位向量。

➢ **例：定义复位向量。**

```
        ORG     0               ;
        JMP     START           ; 跳至用户程序。
        …

        ORG     10H
START:                          ; 用户程序起始地址。
        …                       ; 用户程序。
        …
        ENDP                    ; 程序结束。
```

2.1.2 中断向量（0008H）

中断向量地址为 0008H。一旦有中断响应，程序计数器 PC 的当前值就会存入堆栈缓存器并跳转到 0008H 开始执行中断服务程序。用户必须定义中断向量，下面的示例程序说明了如何在程序存储器种定义中断向量。

✷ **注：通过“PUSH”、“POP”指令保存和恢复 ACC/PFLAG（不包括 NT0，NPD）寄存器，PUSH/POP 缓存器只有一层。**

➢ **例：定义中断向量，中断服务程序紧随 ORG 8 之后。**

```
.CODE
                ORG     0
                JMP     START           ; 跳到用户程序。
                …
                ORG     8               ; 中断向量。
                PUSH                    ; 保存 ACC 和 PFLAG 寄存器。
                …
                …
                POP                     ; 恢复 ACC 和 PFLAG 寄存器。
                RETI                    ; 中断服务程序结束。
                …
START:                                  ; 用户程序开始。
                …                       ; 用户程序。
                …
                JMP     START           ; 用户程序结束。
                …

                ENDP                    ; 程序结束。
```

➢ **例：定义中断向量，中断服务程序在用户程序之后。**

```
.CODE
                ORG     0
                JMP     START           ; 跳到用户程序。
                …
                ORG     8               ; 中断向量。
                JMP     MY_IRQ          ; 跳到中断服务程序。

                ORG     10H
START:                                  ; 用户程序开始。
                …                       ; 用户程序。
                …
                …
                JMP     START           ; 用户程序结束。
                …
MY_IRQ:                                 ; 中断服务程序开始。
                PUSH                    ; 保存 ACC 和 PFLAG 寄存器。
                …
                …
                POP                     ; 恢复 ACC 和 PFLAG 寄存器。
                RETI                    ; 中断服务程序结束。
                …
                ENDP                    ; 程序结束。
```

✷ **注：从上面的程序中容易得出 SONiX 的编程规则，有以下几点：**
1、地址 0000H 的“JMP”指令使程序从头开始执行；
2、地址 0008H 是中断向量；
3、用户的程序应该是一个循环。

2.1.3 查表

在 SONiX 单片机中，对 ROM 区中的数据进行查找，查表指针存放在寄存器 Y，Z 中。寄存器 Y 指向所找数据地址的中间字节（bit8~bit15），寄存器 Z 指向所找数据地址的低字节（bit0~bit7）。执行完 MOVC 指令后，所查找数据低字节内容被存入 ACC 中，而数据高字节内容被存入 R 寄存器。

➢ **例：查找存储在“TABLE1”中的数据**

```
        B0MOV   Y, #TABLE1$M    ; 设置 table1 的中间字节地址。
        B0MOV   Z, #TABLE1$L    ; 设置 table1 的低位字节地址。
        MOVC                    ; 查表，R = 00H，ACC = 35H。

                                ; 查找下一地址。
        INCMS   Z               ; Z+1。
        JMP     @F              ; Z 没有溢出。
        INCMS   Y               ; Z 溢出，Y=Y+1。
        NOP
                                ;
@@:     MOVC                    ; 查表，R = 51H，ACC = 05H。
        …                       ;
TABLE1: DW      0035H           ; 定义数据表数据（16-bit）。
        DW      5105H
        DW      2012H
        …
```

✷ **注：当寄存器 Z 溢出（从 FFH 变成 00H）时，Y 寄存器并不会自动加 1。用户必须注意这种情况以免查表错误。若 Z 溢出，Y 必须由程序加 1。下面的宏指令 INC_YZ 可以对 Y 和 Z 寄存器自动处理。**

➢ **例：宏指令 INC_YZ。**

```
INC_YZ  MACRO
        INCMS   Z               ; Z+1
        JMP     @F              ; 没有溢出。

        INCMS   Y               ; Y+1
        NOP                     ; 没有溢出。
@@:
        ENDM
```

➢ **例：通过宏指令“INC_YZ”对上例优化。**

```
        B0MOV   Y, #TABLE1$M    ; 设置 table1 的中间字节地址。
        B0MOV   Z, #TABLE1$L    ; 设置 table1 的低位字节地址。
        MOVC                    ; 查表，R = 00H，ACC = 35H。

        INC_YZ                  ; 查找下一地址。
                                ;
@@:     MOVC                    ; 查表，R = 51H，ACC = 05H。
        …                       ;
TABLE1: DW      0035H           ; 定义数据表数据（16-bit）。
        DW      5105H
        DW      2012H
        …
```

下面的程序通过累加器对 Y 和 Z 寄存器进行处理来实现查表功能，但需要特别注意进位时的处理。

➢ **例：通过指令 B0ADD/ADD 实现查表功能。**

```
        B0MOV   Y, #TABLE1$M    ; 设置 table1 的中间字节地址。
        B0MOV   Z, #TABLE1$L    ; 设置 table1 的低位字节地址。

        MOV     A, BUF          ; Z = Z + BUF。
        B0ADD   Z, A
        MOV     A, #1
        B0BTS0  FC              ; 检查进位标志。
        B0ADD   Y, A            ; FC = 1，Y+1。

GETDATA:                        ;
        MOVC                    ; 存储数据，如果 BUF = 0，数据为 35H。
                                ; 如果 BUF = 1，数据＝5105H。
                                ; 如果 BUF = 2，数据＝2012H
        …
TABLE1: DW      0035H           ; 定义数据表数据（16-bit）。
        DW      5105H
        DW      2012H
        …
```

2.1.4 跳转表

跳转表能够实现多地址跳转功能。由于 PCL 和 ACC 的值相加即可得到新的 PCL，因此，可以通过对 PCL 加上不同的 ACC 值来实现多地址跳转。ACC 值若为 n，PCL+ACC 即表示当前地址加 n，执行完当前指令后 PCL 值还会自加 1，可参考以下范例。如果 PCL＋ACC 后发生溢出，PCH 则自动加 1。由此得到的新的 PC 值再指向跳转指令列表中新的地址。这样，用户就可以通过修改 ACC 的值轻松实现多地址的跳转。

✷ **注：PCH 只支持 PC 增量运算，而不支持 PC 减量运算。当 PCL+ACC 后如有进位，PCH 的值会自动加 1。PCL-ACC 后若有借位，PCH 的值将保持不变，用户在设计应用时要加以注意。**

➢ **例：跳转表。**

```
        ORG     0100H           ; 跳转表从 ROM 前端开始。

        B0ADD   PCL, A          ; PCL = PCL + ACC，PCL 溢出时 PCH 加 1。
        JMP     A0POINT         ; ACC = 0，跳至 A0POINT。
        JMP     A1POINT         ; ACC = 1，跳至 A1POINT。
        JMP     A2POINT         ; ACC = 2，跳至 A2POINT。
        JMP     A3POINT         ; ACC = 3，跳至 A3POINT。
```

SONiX 单片机提供一个宏以保证可靠执行跳转表功能，它会自动检测 ROM 边界并将跳转表移至适当的位置。但采用该宏程序会占用部分 ROM 空间。

➢ **例：如果跳转表跨越 ROM 边界，将引起程序错误。**

```
@JMP_A  MACRO   VAL
        IF      (($+1) !& 0XFF00) !!= (($+(VAL)) !& 0XFF00)
        JMP     ($ | 0XFF)
        ORG     ($ | 0XFF)
        ENDIF
        ADD     PCL, A
        ENDM
```

✷ **注：“VAL”为跳转表列表中列表个数。**

➢ **例：宏“MACRO3.H”中，“@JMP_A”的应用。**

```
        B0MOV   A, BUF0         ; “BUF0”从 0 至 4。
        @JMP_A  5               ; 列表个数为 5。
        JMP     A0POINT         ; ACC = 0，跳至 A0POINT。
        JMP     A1POINT         ; ACC = 1，跳至 A1POINT。
        JMP     A2POINT         ; ACC = 2，跳至 A2POINT。
        JMP     A3POINT         ; ACC = 3，跳至 A3POINT。
        JMP     A4POINT         ; ACC = 4，跳至 A4POINT。
```

如果跳转表恰好位于 ROM BANK 边界处（00FFH~0100H），宏指令“@JMP_A”将调整跳转表到适当的位置（0100H）。

➢ **例：“@JMP_A”运用举例**

```
; 编译前
ROM 地址
                B0MOV     A, BUF0       ; “BUF0”从 0 到 4。
                @JMP_A    5             ; 列表个数为 5。
00FDH           JMP       A0POINT       ; ACC = 0，跳至 A0POINT。
00FEH           JMP       A1POINT       ; ACC = 1，跳至 A1POINT。
00FFH           JMP       A2POINT       ; ACC = 2，跳至 A2POINT。
0100H           JMP       A3POINT       ; ACC = 3，跳至 A3POINT。
0101H           JMP       A4POINT       ; ACC = 4，跳至 A4POINT。

; 编译后
ROM 地址
                B0MOV     A, BUF0       ; “BUF0”从 0 到 4。
                @JMP_A    5             ; 列表个数为 5。
0100H           JMP       A0POINT       ; ACC = 0，跳至 A0POINT。
0101H           JMP       A1POINT       ; ACC = 1，跳至 A1POINT。
0102H           JMP       A2POINT       ; ACC = 2，跳至 A2POINT。
0103H           JMP       A3POINT       ; ACC = 3，跳至 A3POINT。
0104H           JMP       A4POINT       ; ACC = 4，跳至 A4POINT。
```

2.1.5 CHECKSUM计算

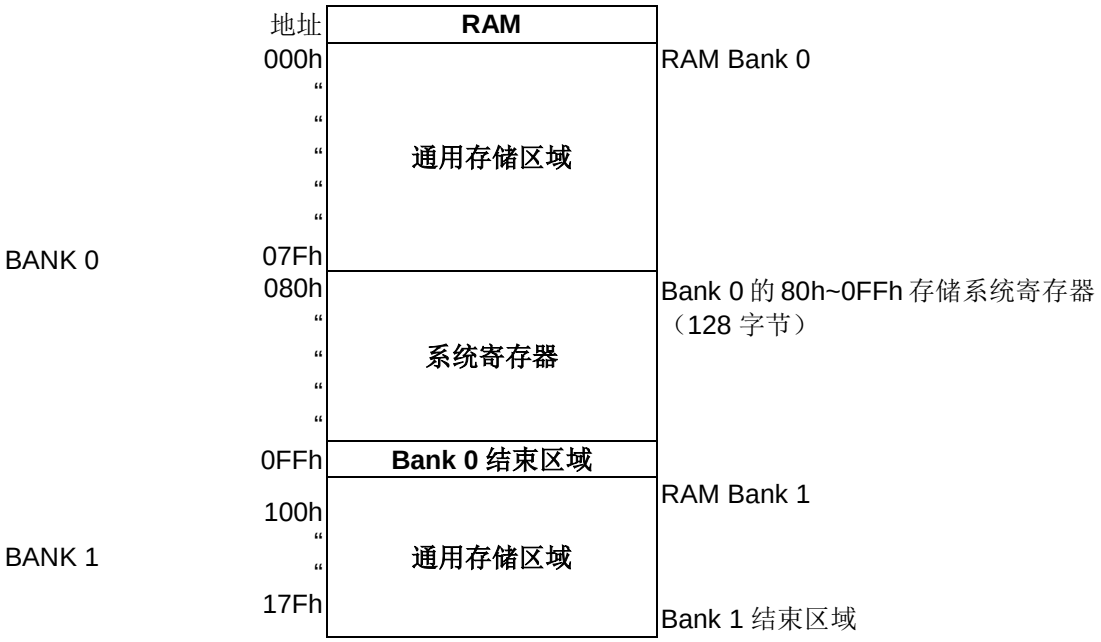
ROM 区末端位置的几个字限制使用，进行 Checksum 计算时，用户应避免对该单元的访问。

➢ **例：示例程序演示了如何对 00H 到用户程序结束进行 Checksum 计算。**

```
                MOV     A,#END_USER_CODE$L
                B0MOV   END_ADDR1, A        ; 用户程序结束地址低位地址存入 end_addr1。
                MOV     A,#END_USER_CODE$M
                B0MOV   END_ADDR2, A        ; 用户程序结束地址中间地址存入 end_addr2。
                CLR     Y                   ; 清 Y。
                CLR     Z                   ; 清 Z。
@@:
                MOVC
                B0BCLR  FC                  ; 清标志位 C。
                ADD     DATA1, A            ;
                MOV     A, R
                ADC     DATA2, A            ;
                JMP     END_CHECK           ; 检查 YZ 地址是否为代码的结束地址。
AAA:
                INCMS   Z
                JMP     @B                  ; 若 Z != 00H，进行下一个计算。
                JMP     Y_ADD_1             ; 若 Z = 00H，Y+1。
END_CHECK:
                MOV     A, END_ADDR1
                CMPRS   A, Z                ; 检查 Z 地址是否为用户程序结束地址低位地址。
                JMP     AAA                 ; 否，则进行 Checksum 计算。
                MOV     A, END_ADDR2
                CMPRS   A, Y                ; 是则检查 Y 的地址是否为用户程序结束地址中间地址。
                JMP     AAA                 ; 否，则进行 Checksum 计算。
                JMP     CHECKSUM_END        ; 是则 Checksum 计算结束。

Y_ADD_1:
                INCMS   Y                   ;
                NOP
                JMP     @B                  ; 跳转到 Checksum 计算。
CHECKSUM_END:
                …
                …
END_USER_CODE:                              ; 程序结束。
```

2.2 数据存储器（RAM）

☞ **RAM：256 * 8 bit**



2.2.1 系统寄存器

2.2.1.1 系统寄存器列表

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	L	H	R	Z	Y	-	PFLAG	RBANK	-	-	-	-	-	-	-	-
9	MSPSTAT	MSPM1	MSPBUF	MSPADR	-	-	-	-	-	-	-	-	CM0M	CM0M1	-	-
A	-	-	-	-	-	-	-	-	TC0R0	TC0R1	TC0R2	TC0R3	TC0R4	TC0R5	TC0R6	TC0R7
B	-	-	-	-	SIOM	SIOR	SIOB	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	-	-	-	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	LPWS	PCL	PCH
D	P0	P1	-	-	-	P5	-	-	T0M	T0C	TC0M	TC0C	TC1M	TC1C	TC1R	STKP
E	P0UR	P1UR	-	-	-	P5UR	@HL	@YZ	-	P1OC	TC1D	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2 系统寄存器说明

H, L = 工作寄存器，@HL 间接寻址寄存器
Y, Z = 工作寄存器，@YZ 间接寻址寄存器，ROM 地址寄存器
R = 工作寄存器，ROM 查表数据缓存器
PFLAG = ROM 页，特殊标志寄存器
CM0M = 比较器配置寄存器
CM0M1 = 比较器配置寄存器
SIOM = SIO 配置寄存器
SIOR = SIO 时钟寄存器
SIOB = SIO 数据缓存器
PnM = Pn 输入/输出模式寄存器
PEDGE = P0.0 触发方向控制寄存器
INTRQ = 中断请求寄存器
INTEN = 中断使能寄存器
OSCM = 振荡模式寄存器
WDTR = 看门狗清零定时器
PCH, PCL = 程序计数器
Pn = Pn 数据缓存器
PnUR = Pn 上拉电阻控制寄存器
T0M = T0 模式寄存器
T0C = T0 计数寄存器
TC0M = TC0 模式寄存器
TC0C = TC0 计数寄存器
TC0Rn = TC0 PWM 数据缓存器
LPWS = PWM 通道选择寄存器
TC1M = TC1 模式寄存器
TC1C = TC1 计数寄存器
TC1R = TC1 自动装载数据缓存器
TC1D = TC1 占空比控制寄存器
@HL = 间接寻址寄存器
@YZ = 间接寻址寄存器
STK0~STK7 = 堆栈缓存器
STKP = 堆栈指针
MSPSTAT = MSP 状态寄存器
MSPM1= MSP 模式寄存器
MSPBUF = MSP 数据缓存器
MSPADR= MSP 地址寄存器

2.2.1.3 系统寄存器的位定义

地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	注释
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	NT0	NPD	LVD36	LVD24		C	DC	Z	R/W	PFLAG
087H								RBNKS0	R/W	RBANK
090H		CKE	D_A	P	S	RED_WRT		BF	R/W	MSPSTAT
091H	WCOL	MSPOV	MSPENB	CKP	SLRXCKP	MSPWK	GCEN		R/W	MSPM1
092H	MSPBUF7	MSPBUF6	MSPBUF5	MSPBUF4	MSPBUF3	MSPBUF2	MSPBUF1	MSPBUF0	R/W	MSPBUF
093H	MSPADR7	MSPADR6	MSPADR5	MSPADR4	MSPADR3	MSPADR2	MSPADR1	MSPADR0	R/W	MSPADR
09CH	CM0EN	CM0OUT	CM0S1	CM0S0	CMCH3	CMCH2	CMCH1	CMCH0	R/W	CM0M
09DH					CMDB1	CMDB0	CM0OEN	CM0G	R/W	CM0M1
0A8H	TC0R07	TC0R06	TC0R05	TC0R04	TC0R03	TC0R02	TC0R01	TC0R00	W	TC0R0
0A9H	TC0R17	TC0R16	TC0R15	TC0R14	TC0R13	TC0R12	TC0R11	TC0R10	W	TC0R1
0AAH	TC0R27	TC0R26	TC0R25	TC0R24	TC0R23	TC0R22	TC0R21	TC0R20	W	TC0R2
0ABH	TC0R37	TC0R36	TC0R35	TC0R34	TC0R33	TC0R32	TC0R31	TC0R30	W	TC0R3
0ACH	TC0R47	TC0R46	TC0R45	TC0R44	TC0R43	TC0R42	TC0R41	TC0R40	W	TC0R4
0ADH	TC0R57	TC0R56	TC0R55	TC0R54	TC0R53	TC0R52	TC0R51	TC0R50	W	TC0R5
0AEH	TC0R67	TC0R66	TC0R65	TC0R64	TC0R63	TC0R62	TC0R61	TC0R60	W	TC0R6
0AFH	TC0R77	TC0R76	TC0R75	TC0R74	TC0R73	TC0R72	TC0R71	TC0R70	W	TC0R7
0B4H	SENB	START	SRATE1	SRATE0	MLSB	SCLKMD	CPOL	CPHA	R/W	SIOM
0B5H	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0	W	SIOR
0B6H	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0	R/W	SIOB
0B8H				P04M	P03M	P02M	P01M	P00M	R/W	P0M
0BFH				P00G1	P00G0				R/W	PEDGE
0C0H	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W	W	P1W
0C1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C5H	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C8H	CM0IRQ	TC1IRQ		T0IRQ	SIOIRQ		MSPIRQ	P00IRQ	R/W	INTRQ
0C9H	CM0IEN	TC1IEN		T0IEN	SIOIEN		MSPIEN	P00IEN	R/W	INTEN
0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	LPWS7	LPWS6	LPWS5	LPWS4	LPWS3	LPWS2	LPWS1	LPWS0	R/W	LPWS
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH						PC10	PC9	PC8	R/W	PCH
0D0H			P05	P04	P03	P02	P01	P00	R/W	P0
0D1H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D5H	P57	P56	P55	P54	P53	P52	P51	P50	R/W	P5
0D8H	T0ENB	T0rate2	T0rate1	T0rate0					R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0				PWMDR	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	TC1ENB	TC1rate2	TC1rate1	TC1rate0		TC1CKS0		PWM1OUT	R/W	TC1M
0DDH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
0DEH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	W	TC1R
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H								P00R	W	P0UR
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E5H	P57R	P56R	P55R	P54R	P53R	P52R	P51R	P50R	W	P5UR
0E6H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E9H				P52OC	P51OC	P50OC			W	P1OC
0EAH	TC1D7	TC1D6	TC1D5	TC1D4	TC1D3	TC1D2	TC1D1	TC1D0	R/W	TC1D
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H						S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H						S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H						S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H						S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H						S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH						S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH						S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH						S0PC10	S0PC9	S0PC8	R/W	STK0H

✸ **注：**
1、为了避免系统错误，在初始化时，请将上表所有寄存器的位都按照设计要求设置为确定的"1"或者"0"；
2、所有寄存器名都已在 SN8ASM 编译器中做了宣告；
3、用户使用 SN8ASM 编译器对寄存器的位进行操作时，须在该寄存器的位前加“F”；
4、指令“b0bset”，“b0bclr”，“bset”，“bclr”只能用于可读写的寄存器（“R/W”）。

2.2.2 累加器

8 位数据寄存器 ACC 用来执行 ALU 与数据存储器之间数据的传送操作。如果操作结果为零（Z）或有进位产生（C 或 DC），程序状态寄存器 PFLAG 中相应位会发生变化。

ACC 并不在 RAM 中，因此在立即寻址模式中不能用“B0MOV”指令对其进行读写。

➢ **例：读/写 ACC。**

```
; 将立即数写入 ACC。
                MOV         A, #0FH

; 把 ACC 中的数据存入 BUF 中。
                MOV         BUF, A
                B0MOV       BUF, A

; 把 BUF 中的数据送到 ACC 中。
                MOV         A, BUF
                B0MOV       A, BUF
```

系统执行中断时，不会自动保存 ACC 和 PFLAG，必须通过 PUSH、POP 指令来保存和恢复 ACC 和 PFLAG。

➢ **例：保存 ACC 和工作寄存器。**

```
INT_SERVICE:
                PUSH                    ; 保存 ACC 和 PFLAG。
                …
                …
                POP                     ; 恢复 ACC 和 PFLAG。

                RETI                    ; 退出中断。
```

2.2.3 程序状态寄存器PFLAG

寄存器 PFLAG 中包含 ALU 运算状态信息、系统复位状态信息和 LVD 低电压检测状态信息。其中，位 NT0 和 NPD 显示系统复位状态信息，包括上电复位、LVD 复位、外部复位和看门狗复位；位 C、DC 和 Z 显示 ALU 的运算信息；位 LVD24、LVD36 显示 LVD 检测电源电压的状态。

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	LVD36	LVD24	-	C	DC	Z
读/写	R/W	R/W	R	R	-	R/W	R/W	R/W
复位	-	-	0	0	-	0	0	0

Bit [7:6] **NT0, NPD：**复位状态标志位。

NT0	NPD	复位状态
0	0	看门狗定时器溢出
0	1	系统保留
1	0	LVD 复位
1	1	外部复位引脚复位

Bit 5 **LVD36：**LVD 3.6V 工作电压标志，LVD 编译选项为 LVD_H 时有效。
0 = 无效（VDD > 3.6V）；
1 = 有效（VDD <= 3.6V）。

Bit 4 **LVD24：**LVD 2.4V 工作电压标志，LVD 编译选项为 LVD_M 时有效。
0 = 无效（VDD > 2.4V）；
1 = 有效（VDD <= 2.4V）。

Bit 2 **C：**进位标志。
1 =加法运算后有进位、减法运算没有借位发生或移位后移出逻辑“1”或比较运算的结果≥0；
0 =加法运算后没有进位、减法运算有借位发生或移位后移出逻辑“0”或比较运算的结果<0。

Bit 1 **DC：**辅助进位标志。
1 =加法运算时低四位有进位，或减法运算后没有向高四位借位；
0 =加法运算时低四位没有进位，或减法运算后有向高四位借位。

Bit 0 **Z：**零标志。
1 =算术/逻辑/分支转移运算的结果为零；
0 =算术/逻辑/分支转移运算的结果非零。

✷ **注：关于标志位 C、DC 和 Z 的更多信息请参阅指令集相关内容。**

2.2.4 程序计数器

程序计数器 PC 是一个 11 位二进制程序地址寄存器，分高 3 位和低 8 位。专门用来存放下一条需要执行指令的内存地址。通常，程序计数器会随程序中指令的执行自动增加。

若程序执行 CALL 和 JMP 指令时，PC 指向特定的地址。

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC	-	-	-	-	-	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
复位后	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

☞ **单地址跳转**

在 SONiX 单片机里面，有 9 条指令（CMPRS、INCS、INCMS、DECS、DECMS、BTS0、BTS1、B0BTS0 和 B0BTS1）可完成单地址跳转功能。如果这些指令执行结果为真，那么 PC 值加 2 以跳过下一条指令。

如果位测试为真，PC 加 2。

```
                B0BTS1      FC          ; 若 Carry_flag = 1 则跳过下一条指令。
                JMP         C0STEP      ; 否则执行 C0STEP。
                …
C0STEP:         NOP

                B0MOV       A, BUF0     ; BUF0 送入 ACC。
                B0BTS0      FZ          ; Zero flag = 0 则跳过下一条指令。
                JMP         C1STEP      ; 否则执行 C1STEP。
                …
C1STEP:         NOP
```

如果 ACC 等于指定的立即数则 PC 值加 2，跳过下一条指令。

```
                CMPRS       A, #12H
                JMP         C0STEP
                …
C0STEP:         NOP
```

执行加 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

INCS：

```
                INCS        BUF0
                JMP         C0STEP      ;如果 ACC 不为“0”，则跳至 C0STEP。
                …
C0STEP:         NOP
```

INCMS：

```
                INCMS       BUF0
                JMP         C0STEP      ;如果 BUF0 不为“0”，则跳至 C0STEP。
                …
C0STEP:         NOP
```

执行减 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

DECS：

```
                DECS        BUF0
                JMP         C0STEP      ;如果 ACC 不为“0”，则跳至 C0STEP。
                …
C0STEP:         NOP
```

DECMS：

```
                DECMS       BUF0
                JMP         C0STEP      ;如果 BUF0 不为“0”，则跳至 C0STEP。
                …
C0STEP:         NOP
```

☞ **多地址跳转**

执行 JMP 或 ADD M,A（M=PCL）指令可实现多地址跳转。执行 ADD M, A、ADC M, A 或 B0ADD M, A 后，若 PCL 溢出，PCH 会自动进位。对于跳转表及其它应用，用户可以通过上述 3 条指令计算 PC 的值而不需要担心 PCL 溢出的问题。

✸ **注：PCH 仅支持 PC 的递增运算而不支持递减运算。当 PCL+ACC 执行完 PCL 有进位时，PCH 会自动加 1；但执行 PCL-ACC 有借位发生，PCH 的值会保持不变。**

➢ **例：PC = 0323H（PCH = 03H，PCL = 23H）。**

```
; PC = 0323H
                MOV         A, #28H
                B0MOV       PCL, A          ; 跳到地址 0328H。
                …

; PC = 0328H
                MOV         A, #00H
                B0MOV       PCL, A          ; 跳到地址 0300H。
                …
```

➢ **例：PC = 0323H（PCH = 03H，PCL = 23H）。**

```
; PC = 0323H
                B0ADD       PCL, A          ; PCL = PCL + ACC，PCH 的值不变。
                JMP         A0POINT         ; ACC = 0，跳到 A0POINT。
                JMP         A1POINT         ; ACC = 1，跳到 A1POINT。
                JMP         A2POINT         ; ACC = 2，跳到 A2POINT。
                JMP         A3POINT         ; ACC = 3，跳到 A3POINT。
                …
                …
```

2.2.5 H,L寄存器

寄存器 H 和 L 都是 8 位缓存器，主要有以下两个功能：

- **通用工作寄存器；**
- **RAM 数据寻址指针@HL。**

081H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

080H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
L	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

➢ **例：用 H、L 作为数据指针，访问 bank0 中 020H 处的内容。**

```
B0MOV   H, #00H
B0MOV   L, #20H
B0MOV   A, @HL
```

➢ **例：对 bank 0 中的数据进行清零处理。**

```
        CLR     H           ; H = 0，指向 bank 0。
        B0MOV   L, #7FH     ; L = 7FH。
CLR_HL_BUF:
        CLR     @HL         ; @HL 清零。
        DECMS   L           ; L – 1，如果 L = 0，程序结束。
        JMP     CLR_HL_BUF

        CLR     @HL
END_CLR:
        …
        …
```

2.2.6 Y, Z寄存器

寄存器 Y 和 Z 都是 8 位缓存器，主要用途如下：

- **通用工作寄存器；**
- **RAM 数据寻址指针@YZ；**
- **配合指令 MOVC 对 ROM 数据进行查表。**

084H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Y	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

083H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Z	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

➢ **例：用 Y、Z 作为数据指针，访问 bank0 中 025H 处的内容。**

```
B0MOV   Y, #00H     ; Y 指向 RAM bank 0。
B0MOV   Z, #25H     ; Z 指向 25H。
B0MOV   A, @YZ      ; 数据送入 ACC。
```

➢ **例：利用数据指针@YZ 对 RAM 数据清零。**

```
        B0MOV   Y, #0       ; Y = 0，指向 bank 0。
        B0MOV   Z, #7FH     ; Z = 7FH，RAM 区的最后单元。

CLR_YZ_BUF:
        CLR     @YZ         ; @YZ 清零。

        DECMS   Z           ;
        JMP     CLR_YZ_BUF  ; 不为零。

        CLR     @YZ
END_CLR:                    ;
        …
```

2.2.7 R寄存器

8 位缓存器 R 主要有以下两个功能：

- **通用工作寄存器使用；**
- **存储执行查表指令后的高字节数据。**

（执行 MOVC 指令，指定 ROM 单元的高字节数据会被存入 R 寄存器而低字节数据则存入 ACC。）

082H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	-	-	-	-	-

✷ **注：R 寄存器的详细内容请查看“查表”章节。**

2.3 寻址模式

2.3.1 立即寻址模式

将立即数直接送入 ACC 或指定的 RAM 单元。

- **例：立即数 12H 送入 ACC。**

```
MOV     A, #12H
```

- **例：立即数 12H 送入寄存器 R。**

```
B0MOV   R, #12H
```

✷ **注：立即寻址模式中，指定的 RAM 单元必须是 80H~87H 的工作寄存器。**

2.3.2 直接寻址模式

通过 ACC 对 RAM 单元数据进行操作。

- **例：地址 12H 处的内容送入 ACC。**

```
B0MOV   A, 12H
```

- **例：ACC 中数据写入 RAM 中 12H 单元。**

```
B0MOV   12H, A
```

2.3.3 间接寻址模式

通过数据指针（Y/Z）对数据存储单元进行读写。

- **例：通过指针@HL 间接寻址。**

```
B0MOV   H, #0       ; 清“H”以寻址 RAM bank 0。
B0MOV   L, #12H     ; 设定寄存器地址。
B0MOV   A, @HL
```

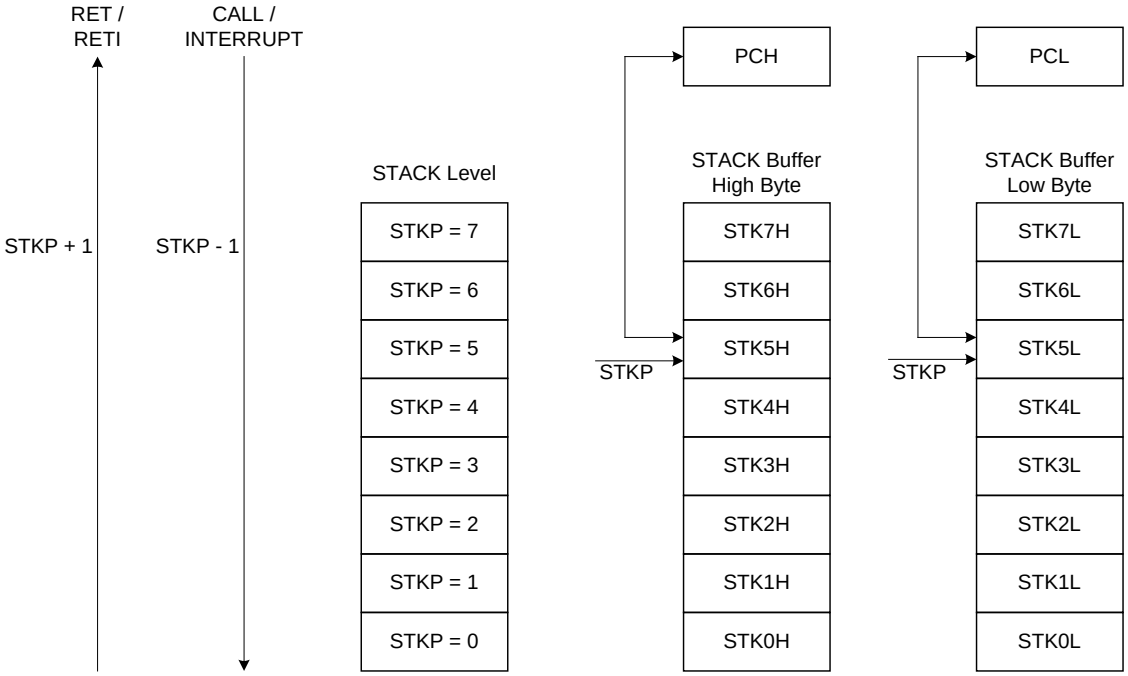
- **例：通过指针@YZ 间接寻址。**

```
B0MOV   Y, #0       ; 清“Y”以寻址 RAM bank 0。
B0MOV   Z, #12H     ; 设定寄存器地址。
B0MOV   A, @YZ
```

2.4 堆栈

2.4.1 概述

SN8P2524 的堆栈缓存器共有 8 层，程序进入中断或执行 CALL 指令时，用来存储程序计数器 PC 的值。寄存器 STKP 为堆栈指针，指向堆栈缓存器顶层，STKnH 和 STKnL 分别是各堆栈缓存器高、低字节。



2.4.2 堆栈寄存器

堆栈指针 STKP 是一个 3 位寄存器，存放被访问的堆栈单元地址，11 位数据存储器 STKnH 和 STKnL 用于暂存堆栈数据。以上寄存器都位于 bank 0。

使用入栈指令 PUSH 和出栈指令 POP 可对堆栈缓存器进行操作。堆栈操作遵循后进先出（LIFO）的原则，入栈时堆栈指针 STKP 的值减 1，出栈时 STKP 的值加 1，这样，STKP 总是指向堆栈缓存器顶层单元。

系统进入中断或执行 CALL 指令之前，程序计数器 PC 的值被存入堆栈缓存器中进行入栈保护。

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0
读/写	R/W	-	-	-	-	R/W	R/W	R/W
复位后	0	-	-	-	-	1	1	1

Bit[2:0] **STKPBn**：堆栈指针（n = 0 ~ 2）。

Bit 7 **GIE**：全局中断控制位。
0 = 禁止；
1 = 允许。

➢ **例：系统复位时，堆栈指针寄存器内容为默认值，但强烈建议在程序初始部分重新设定，如下面所示：**

```
MOV     A, #00000111B
B0MOV   STKP, A
```

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnH	-	-	-	-	-	SnPC10	SnPC9	SnPC8
读/写	-	-	-	-	-	R/W	R/W	R/W
复位后	-	-	-	-	-	0	0	0

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnL	SnPC7	SnPC6	SnPC5	SnPC4	SnPC3	SnPC2	SnPC1	SnPC0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

STKn = STKnH , STKnL (n = 7 ~ 0)

2.4.3 堆栈操作举例

执行程序调用指令 CALL 和响应中断服务时，堆栈指针 STKP 的值减 1，指针指向下一个堆栈缓存器。同时，对程序计数器 PC 的内容进行入栈保护。入栈操作如下表所示：

堆栈层数	STKP 寄存器			堆栈缓存器		说明
	STKPB2	STKPB1	STKPB0	高字节	低字节	
0	1	1	1	Free	Free	-
1	1	1	0	STK0H	STK0L	-
2	1	0	1	STK1H	STK1L	-
3	1	0	0	STK2H	STK2L	-
4	0	1	1	STK3H	STK3L	-
5	0	1	0	STK4H	STK4L	-
6	0	0	1	STK5H	STK5L	-
7	0	0	0	STK6H	STK6L	-
8	1	1	1	STK7H	STK7L	-
> 8	1	1	0	-	-	堆栈溢出，出错

对应每个入栈操作，都有一个出栈操作来恢复程序计数器 PC 的值。RETI 指令用于中断服务程序中，RET 用于子程序调用。出栈时，STKP 加 1 并指向下一个空闲堆栈缓存器。堆栈恢复操作如下表所示：

堆栈层数	STKP 寄存器			堆栈缓存器		说明
	STKPB2	STKPB1	STKPB0	高字节	低字节	
8	1	1	1	STK7H	STK7L	-
7	0	0	0	STK6H	STK6L	-
6	0	0	1	STK5H	STK5L	-
5	0	1	0	STK4H	STK4L	-
4	0	1	1	STK3H	STK3L	-
3	1	0	0	STK2H	STK2L	-
2	1	0	1	STK1H	STK1L	-
1	1	1	0	STK0H	STK0L	-
0	1	1	1	Free	Free	-

2.5 编译选项列表（CODE OPTION）

编译选项（CODE OPTION）是一种系统的硬件配置，包括杂讯滤波器的选项，看门狗定时器的操作，LVD 选项，复位引脚选项以及 OTP ROM 的安全控制。如下表所示：

编译选项	配置项目	功能说明
Watch_Dog	Always_On	始终开启看门狗定时器，即使在睡眠模式和绿色模式下也不例外。
	Enable	使能看门狗定时器，但在睡眠模式和绿色模式下处于关闭状态。
	Disable	关闭看门狗定时器。
Fcpu	Fhosc/1	指令周期为 1 个振荡时钟。
	Fhosc/2	指令周期为 2 个振荡时钟。
	Fhosc/4	指令周期为 4 个振荡时钟。
	Fhosc/8	指令周期为 8 个振荡时钟。
	Fhosc/16	指令周期为 16 个振荡时钟。
Reset_Pin	Reset	使能外部复位引脚。
	P05	使能 P0.5 的单向输入引脚功能，没有上拉电阻。
Security	Enable	ROM 代码加密。
	Disable	ROM 代码不加密。
LVD	LVD_L	如果 VDD 低于 2.0V，LVD 复位系统。
	LVD_M	如果 VDD 低于 2.0V，LVD 复位系统。 寄存器 PFLAG 的 LVD24 位作为 2.4V 低电压的监测器。
	LVD_H	如果 VDD 低于 2.4V，LVD 复位系统。 寄存器 PFLAG 的 LVD36 位作为 3.6V 低电压的监测器。
	LVD_MAX	如果 VDD 低于 3.6V，LVD 复位系统。

2.5.1 Fcpu编译选项

Fcpu 指在普通模式下的指令周期。低速模式下，系统时钟源由内部低速 RC 振荡电路提供，Fcpu 不受 Fcpu 编译选项的影响，固定为 Flosc/4（16KHz/4@3V，32KHz/4@5V）。

2.5.2 Reset_Pin编译选项

复位引脚与单向输入引脚共用，由编译选项控制。

- **Reset：使能外部复位引脚功能。当下降沿触发时，系统复位。**
- **P05：使能 P0.5 为单向输入引脚。此时禁止外部复位引脚功能。**

2.5.3 Security编译选项

Security 编译选项是对 OTP ROM 的一种保护，当使能 Security 编译选项，ROM 代码加密，可以保护 ROM 的内容。

3 复位

3.1 概述

SN8P2524 系列的单片机有以下几种复位方式：

- **上电复位；**
- **看门狗复位；**
- **掉电复位；**
- **外部复位（使能外部复位引脚时有效）。**

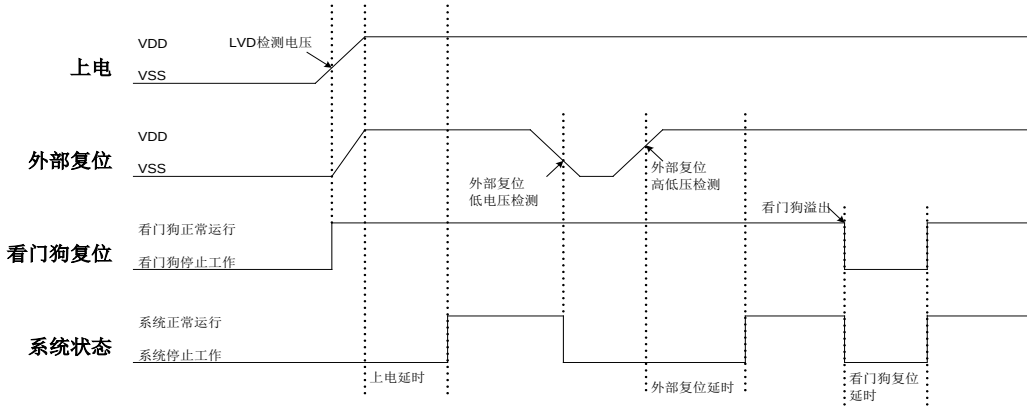
上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器 PC 清零。复位结束后，系统从向量 0000H 处重新开始运行。PFLAG 寄存器的 NT0 和 NPD 两个标志位能够给出系统复位状态的信息。用户可以根据 NT0 和 NPD 的状态，编程控制系统的运行路径。

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	LVD36	LVD24	-	C	DC	Z
读/写	R/W	R/W	R	R	-	R/W	R/W	R/W
复位	-	-	0	0	-	0	0	0

Bit [7:6] **NT0, NPD**：复位状态标志。

NT0	NPD	复位类型	复位条件
0	0	看门狗复位	看门狗定时器溢出
0	1	系统保留	-
1	0	上电复位和 LVD 复位	电源电压低于 LVD 检测电压
1	1	外部复位	外部复位引脚检测到低电平

任何一种复位方式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。对于不同类型的振荡器，完成复位所需要的时间也不同。因此，VDD 的上升速度和不同晶振的起振时间都不固定。RC 振荡器的起振时间最短，晶体振荡器的起振时间则较长。在用户使用的过程中，应考虑系统对上电复位时间的要求。系统复位时序图如下：



3.2 上电复位

上电复位与 LVD 操作密切相关。系统上电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；
- **外部复位（使能外部复位引脚时才有效）：**系统检测外部复位引脚状态。如果不为高电平，系统保持复位状态直到外部复位引脚的复位结束。
- **系统初始化：**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；
- **执行程序：**上电结束，程序开始运行。

3.3 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；
- **系统初始化：**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；
- **执行程序：**上电结束，程序开始运行。

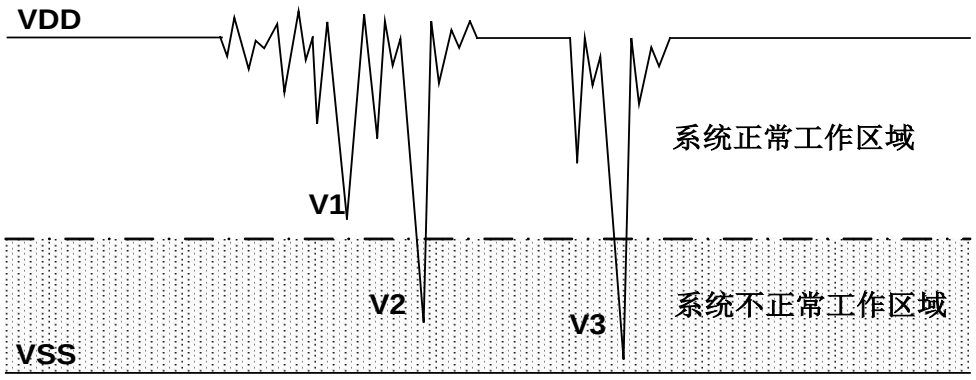
看门狗定时器应用注意事项如下：

- 对看门狗清零之前，检查 I/O 口的状态和 RAM 的内容可增强程序的可靠性；
- 不能在中断中对看门狗清零，否则无法侦测到主程序跑飞的状况；
- 程序中应该只在主程序中有一次清看门狗的动作，这种架构能够最大限度的发挥看门狗的保护功能。

✷ **注：关于看门狗定时器的详细内容，请参阅“看门狗定时器”有关章节。**

3.4 掉电复位

掉电复位针对外部因素引起的系统电压跌落情形（例如：干扰或外部负载的变化），掉电复位可能会引起系统工作状态不正常或程序执行错误。



掉电复位示意图

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

DC 运用中：

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

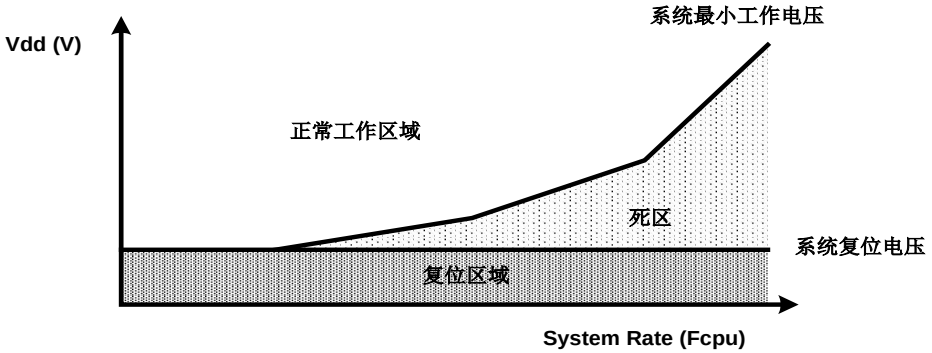
AC 运用中：

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

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

3.4.1 系统工作电压

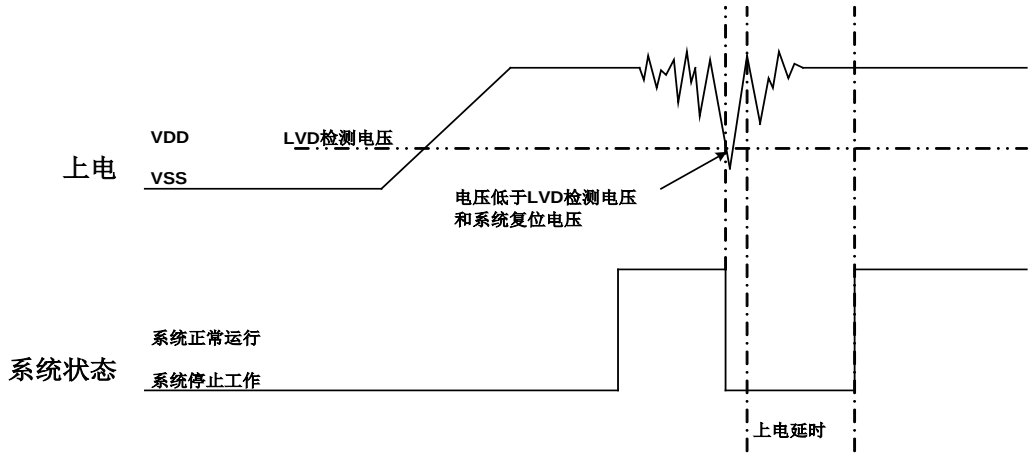
为了改善系统掉电复位的性能，首先必须明确系统具有的最低工作电压值。系统最低工作电压与系统执行速度有关，不同的执行速度下最低工作电压值也不同。



系统工作电压与执行速度关系图

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（LVD）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。

3.4.2 低电压检测（LVD）



低电压检测（LVD）是 SONiX 8 位单片机内置的掉电复位保护装置，当 VDD 跌落并低于 LVD 检测电压值时，LVD 被触发，系统复位。不同的单片机有不同的 LVD 检测电平，LVD 检测电平值仅为一个电压点，并不能覆盖所有死区范围。因此采用 LVD 依赖于系统要求和环境状况。电源变化较大时，LVD 能够起到保护作用，如果电源变化触发 LVD，系统工作仍出错，则 LVD 就不能起到保护作用，就需要采用其它复位方法。

LVD 设计为三层结构（2.0V/2.4V/3.6V），由 LVD 编译选项控制。对于上电复位和掉电复位，2.0V LVD 始终处于使能状态；2.4V LVD 具有 LVD 复位功能，并能通过标志位显示 VDD 状态；3.6V LVD 具有标记功能，可显示 VDD 的工作状态。LVD 标志功能只是一个低电压检测装置，标志位 LVD24 和 LVD36 给出 VDD 的电压情况。对于低电压检测应用，只需查看 LVD24 和 LVD36 的状态即可检测电池状况。

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	LVD36	LVD24	-	C	DC	Z
读/写	R/W	R/W	R	R	-	R/W	R/W	R/W
复位	-	-	0	0	-	0	0	0

Bit 5 **LVD36**：LVD 3.6V 工作电压标志，LVD 编译选项为 LVD_H 时有效。
0 = 无效（VDD > 3.6V）；
1 = 有效（VDD <= 3.6V）。

Bit 4 **LVD24**：LVD 2.4V 工作电压标志，LVD 编译选项为 LVD_M 时有效。
0 = 无效（VDD > 2.4V）；
1 = 有效（VDD <= 2.4V）。

LVD	LVD 编译选项			
	LVD_L	LVD_M	LVD_H	LVD_MAX
2.0V 复位	有效	有效	有效	有效
2.4V 标志	-	有效	-	-
2.4V 复位	-	-	有效	有效
3.6V 标志	-	-	有效	-
3.6V 复位	-	-	-	有效

LVD_L
如果 VDD < 2.0V，系统复位；
LVD24 和 LVD36 标志位无意义。
LVD_M
如果 VDD < 2.0V，系统复位。
使能 PFLAG 的 LVD24 标志位，VDD>2.4V，LVD24=0；VDD <=2.4V，LVD24=1。
LVD36 标志位无意义。
LVD_H
如果 VDD < 2.4V，系统复位。
使能 PFLAG 的 LVD36 标志位，VDD > 3.6V，LVD36＝0；VDD <= 3.6V，LVD36＝1。
LVD_MAX
如果 VDD < 3.6V，系统复位。

✷ **注：**
1、LVD 复位结束后，LVD24 和 LVD36 都将被清零；
2、LVD 2.4V 和 LVD3.6V 检测电平值仅作为设计参考，不能用作芯片工作电压值的精确检测。

3.4.3 掉电复位性能改进

如何改善系统掉电复位性能，有以下几点建议：

- **LVD 复位；**
- **看门狗复位；**
- **降低系统工作速度；**
- **采用外部复位电路（稳压二极管复位电路，电压偏移复位电路，外部 IC 复位电路）。**

✷ **注：“稳压二极管复位电路”、“电压偏移复位电路”和“外部 IC 复位电路”能够完全避免掉电复位出错。**

看门狗复位：

看门狗定时器用于保证系统正常工作。通常，会在主程序中将看门狗定时器清零，但不要在多个分支程序中清看门狗。若程序正常运行，看门狗不会复位。当系统进入死区或程序运行出错的时候，看门狗定时器继续计数直至溢出，系统复位。

如果看门狗复位后电源仍处于死区，则系统复位失败，保持复位状态，直到系统工作状态恢复到正常值。

降低系统工作速度：

系统工作速度越快最低工作电压值越高，从而加大工作死区的范围，因此降低系统工作速度不失为降低系统进入死区几率的有效措施。所以，可选择合适的工作速度以避免系统进入死区，这个方法需要调整整个程序使其满足系统要求。

附加外部复位电路：

外部复位也能够完全改善掉电复位性能。有三种外部复位方式可改善掉电复位性能：稳压二极管复位电路，电压偏移复位电路和外部 IC 复位电路。它们都采用外部复位信号控制单片机可靠复位。

3.5 外部复位

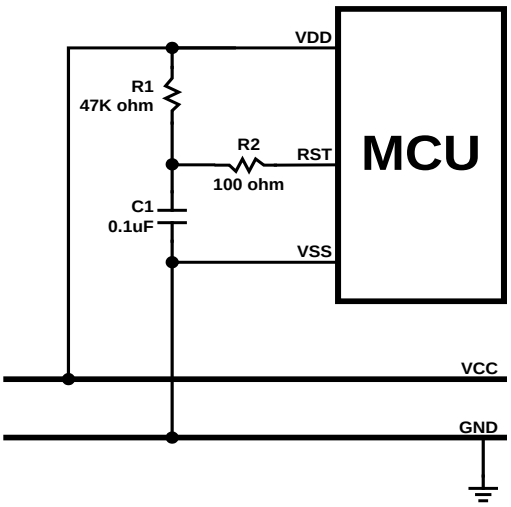
外部复位功能由编译选项“Reset_Pin”控制。将该编译选项置为“Reset”，可使能外部复位功能。外部复位引脚为施密特触发结构，低电平有效。复位引脚处于高电平时，系统正常运行。当复位引脚输入低电平信号时，系统复位。外部复位操作在上电和正常工作模式时有效。需要注意的是，在系统上电完成后，外部复位引脚必须输入高电平，否则系统将一直保持在复位状态。外部复位的时序如下：

- **外部复位（当且仅当外部复位引脚为使能状态）：**系统检测复位引脚的状态，如果复位引脚不为高电平，则系统会一直保持在复位状态，直到外部复位结束；
- **系统初始化：**所有的系统寄存器被置为初始状态；
- **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。

外部复位可以在上电过程中使系统复位。良好的外部复位电路可以保护系统以免进入未知的工作状态，如 AC 应用中的掉电复位等。

3.6 外部复位电路

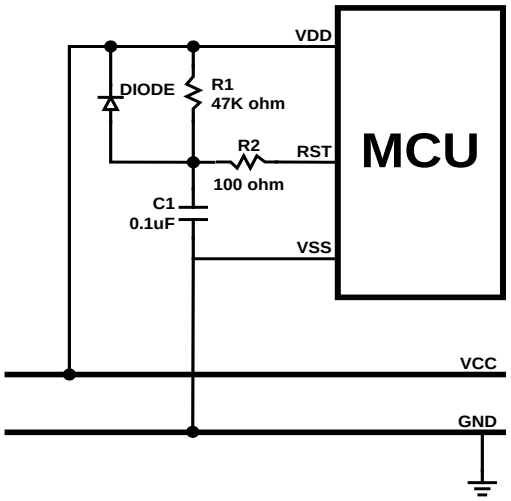
3.6.1 基本RC复位电路



上图为一个由电阻 R1 和电容 C1 组成的基本 RC 复位电路，它在系统上电的过程中能够为复位引脚提供一个缓慢上升的复位信号。这个复位信号的上升速度低于 VDD 的上电速度，为系统提供合理的复位时序，当复位引脚检测到高电平时，系统复位结束，进入正常工作状态。

✷ **注：此 RC 复位电路不能解决非正常上电和掉电复位问题。**

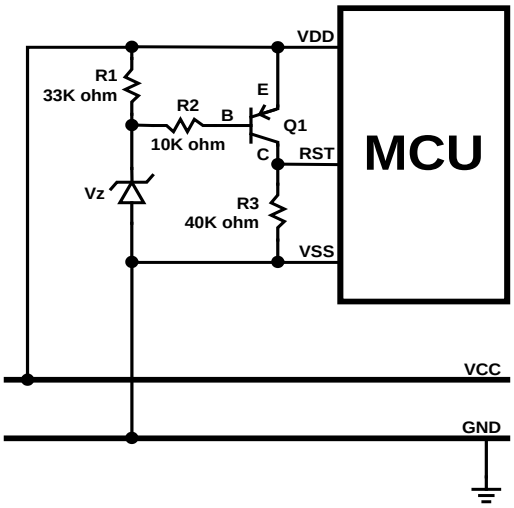
3.6.2 二极管&RC复位电路



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

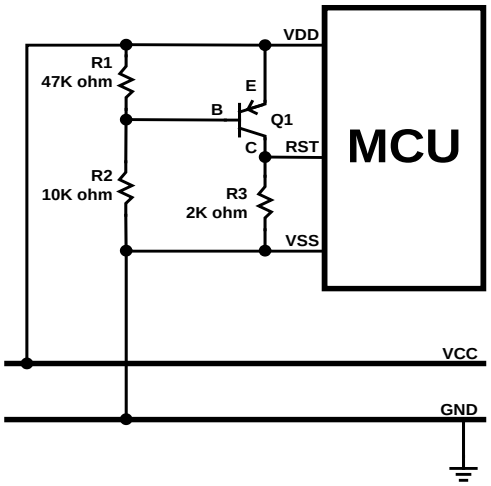
✷ **注：“基本 RC 复位电路”和“二极管及 RC 复位电路”中的电阻 R2 都是必不可少的限流电阻，以避免复位引脚 ESD（Electrostatic Discharge）或 EOS（Electrical Over-stress）击穿 。**

3.6.3 稳压二极管复位电路



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

3.6.4 电压偏置复位电路

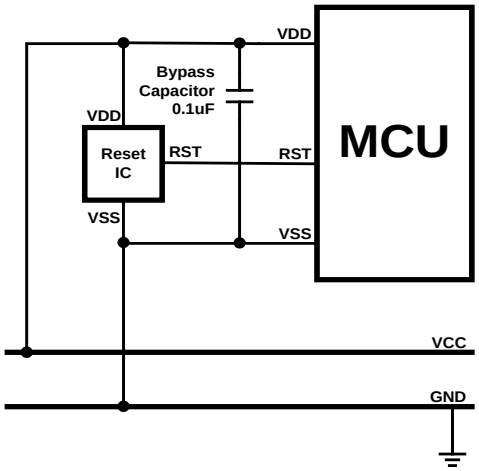


电压偏置复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，这种复位电路的检测电压值的精确度有所降低。电路中，R1 和 R2 构成分压电路，当 VDD 高于和等于分压值“0.7V x (R1 + R2) / R1”时，三极管集电极 C 输出高电平，单片机正常工作；VDD 低于“0.7V x (R1 + R2) / R1”时，集电极 C 输出低电平，单片机复位。

对于不同应用需求，选择适当的分压电阻。单片机复位引脚上电压的变化与 VDD 电压变化之间的差值为 0.7V。如果 VDD 跌落并低于复位引脚复位检测值，那么系统将被复位。如果希望提升电路复位电平，可将分压电阻设置为 R2>R1，并选择 VDD 与集电极之间的结电压高于 0.7V。分压电阻 R1 和 R2 在电路中要耗电，此处的功耗必须计入整个系统的功耗中。

✷ **注：在电源不稳定或掉电复位的情况下，“稳压二极管复位电路”和“偏压复位电路”能够保护电路在电压跌落时避免系统出错。当电压跌落至低于复位检测值时，系统将被复位。从而保证系统正常工作。**

3.6.5 外部IC复位电路



外部复位也可以选用 IC 进行外部复位，但是这样一来系统成本将会增加。针对不用的应用要求选择适当的复位 IC，如上图所示外部 IC 复位电路，能够有效的降低电源变化对系统的影响。

4 系统时钟

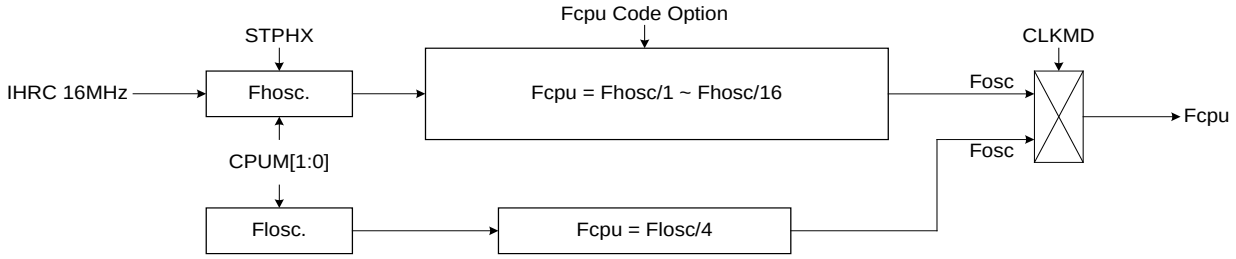
4.1 概述

SN8P2524 内置双时钟系统：高速时钟和低速时钟。高速时钟由内部高速振荡时钟提供。低速时钟由内部低速振荡器提供，由 OSCM 寄存器的 CLKMD 位控制，高、低速时钟都可以作为系统时钟源。

- **高速振荡器**
  内部高速振荡器：RC，高达 16MHz，称为 IHRC；

- **低速振荡器**
  内部低速振荡器：RC，16KHz@3V，32KHz@5V，称为 ILRC。

- **系统时钟框图**



- **Fhosc：内部高速 RC 时钟。**
- **Flosc：内部低速 RC 时钟频率（16KHz@3V，32KHz@5V）。**
- **Fosc：系统时钟频率。**
- **Fcpu：指令执行频率。**

4.2 指令周期（FCPU）

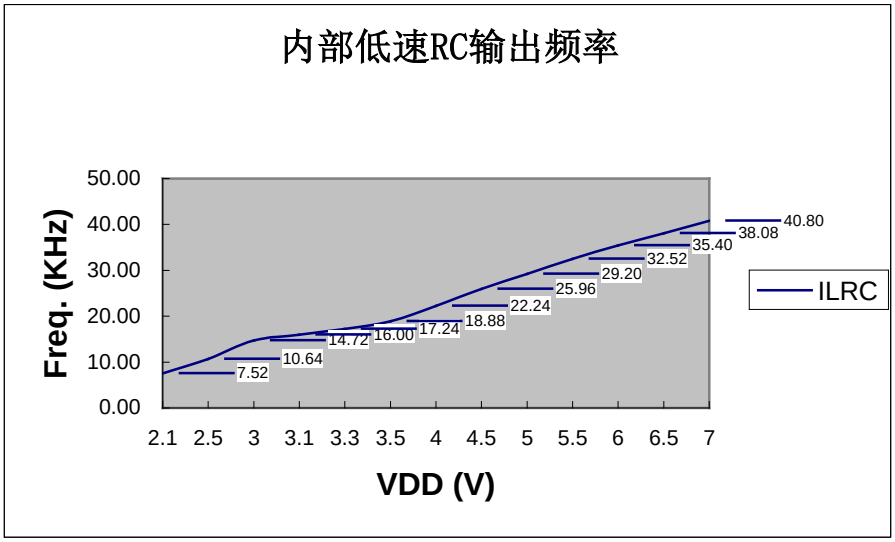
系统时钟速率，即指令周期（Fcpu），从系统时钟源分离出来，决定系统的工作速率。Fcpu 的速率由 Fcpu 编译选项决定，正常模式下，Fcpu=Fhosc/1~Fhosc/16。当 Fcpu 编译选项选择 Fhosc/4，则 Fcpu 频率为 16MHz/4=4MHz。低速模式下，Fcpu=Flosc/4，即 16KHz/4=4KHz@3V，32KHz/4=8KHz@5V。

4.3 系统高速时钟

系统高速时钟由内部高速 RC 振荡电路提供，可作为系统时钟源。内部高速时钟为 16MHz RC 振荡时钟，精度为±2%。

4.4 系统低速时钟

系统低速时钟源即内置的低速振荡器，采用 RC 振荡电路。低速时钟的输出频率受系统电压和环境温度的影响，通常为 5V 时输出 32KHZ，3V 时输出 16KHZ。输出频率与工作电压之间的关系如下图所示。



低速时钟可作为看门狗定时器以及系统低速模式的时钟源。由 OSCM 寄存器的 CLKMD 位来控制系统工作在低速模式。

- **Flosc = 内部低速 RC 振荡器（16KHz @3V，32KHz @5V）。**
- **低速模式 Fcpu = Flosc / 4。**

在睡眠模式下可以停掉内部低速 RC。

➢ **例：在睡眠模式下，停止内部低速振荡器。**

```
B0BSET      FCPUM0
```

✷ **注：不可以单独停止内部低速时钟；由寄存器 OSCM 的位 CPUM0 和 CPUM1 的设置决定内部低速时钟的状态。**

4.5 OSCM寄存器

寄存器 OSCM 控制振荡器的状态和系统的工作模式。

0CAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSCM	0	0	0	CPUM1	CPUM0	CLKMD	STPHX	0
读/写	-	-	-	R/W	R/W	R/W	R/W	-
复位	-	-	-	0	0	0	0	-

Bit 1 **STPHX**：内部高速振荡器控制位。
0 = 内部高速时钟正常运行；
1 = 内部高速振荡器停止，内部低速 RC 振荡器运行。

Bit 2 **CLKMD**：系统高/低速时钟模式控制位。
0 = 普通模式，系统采用高速时钟；
1 = 低速模式，系统采用内部低速时钟。

Bit[4:3] **CPUM[1:0]**：单片机工作模式控制位。
00 = 普通模式；
01 = 睡眠模式；
10 = 绿色模式；
11 = 系统保留。

STPHX 位为内部高速 RC 振荡器的控制位。当 STPHX=0，内部高速 RC 振荡器正常运行；当 STPHX=1，内部高速 RC 振荡器停止运行。

4.6 系统时钟测量

在设计过程中，用户可通过软件指令周期对系统时钟速度进行测试。

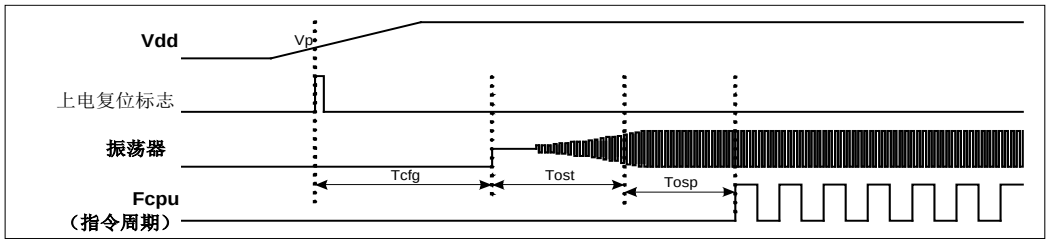
➢ **例：外部振荡器的 Fcpu 指令周期测试。**

```
        B0BSET      P0M.0           ; P0.0 置为输出模式以输出 Fcpu 的触发信号。
@@:
        B0BSET      P0.0
        B0BCLR      P0.0
        JMP         @B
```

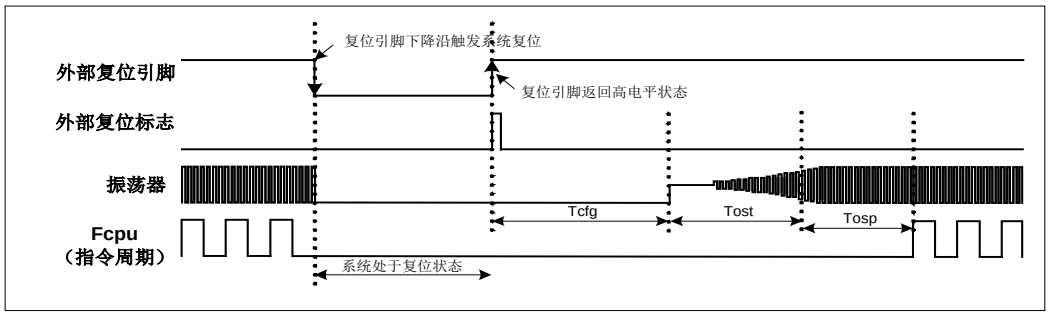
4.7 系统时钟时序

参数	符号	说明	典型值
硬件配置时间	Tcfg	2048*F_{ILRC}	64ms @ F_{ILRC} = 32KHz 128ms @ F_{ILRC} = 16KHz
振荡器启动时间	Tost	启动时间取决于振荡器的材料、工艺等。内部高速 RC 振荡器的启动时间非常短，几乎可以忽略不计。	-
振荡器起振时间	Tosp	复位情况下的振荡器起振时间为 2048*Fhosc （使能上电复位，LVD 复位，看门狗复位，外部复位引脚）	128us @ F_{hosc} = 16MHz
		睡眠模式唤醒情况的振荡器起振时间为： 32*F_{hosc}……内部高速RC振荡器。	2us @ F_{hosc} = 16MHz

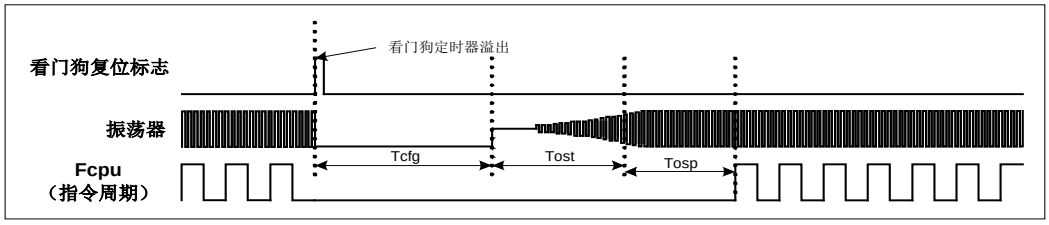
- 上电复位时序：



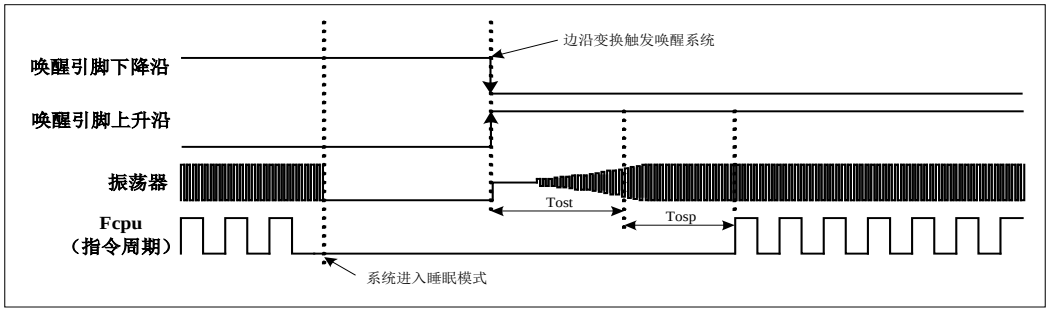
- 外部复位引脚复位时序：



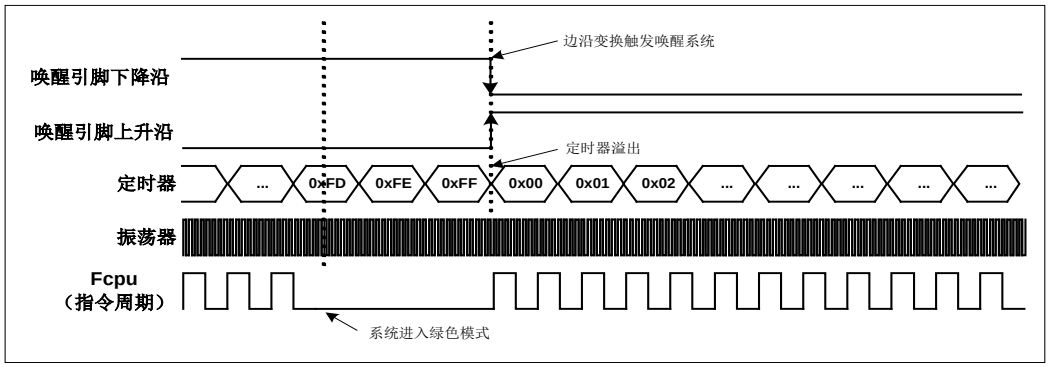
- 看门狗复位时序：



- 睡眠模式唤醒时序：

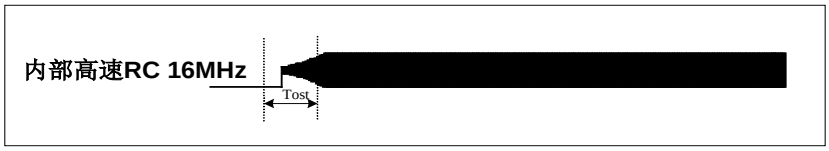


- **绿色模式唤醒时序：**



- **振荡器启动时序：**

  启动时间取决于振荡器的材料、工艺等。内部高速 RC 振荡器的启动时间非常短，几乎可以忽略不计。



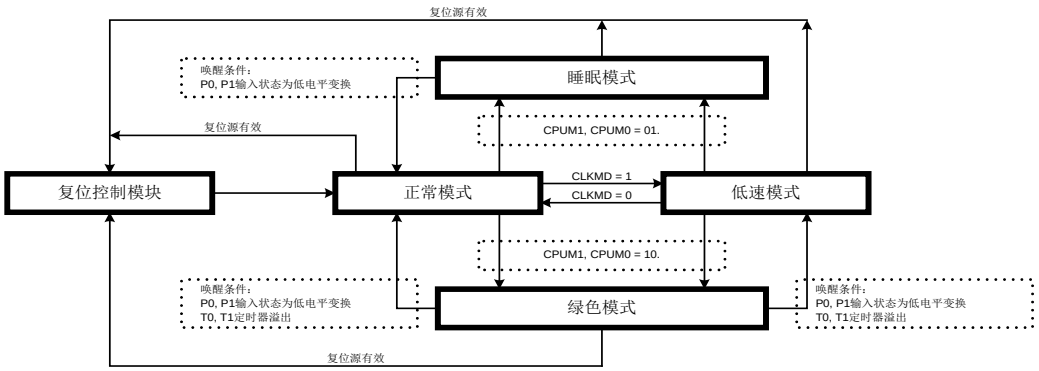
5 系统工作模式

5.1 概述

SN8P2524 可以在 4 种工作模式下以不同的时钟频率工作，这些模式可以控制振荡器的工作、程序的执行以及模拟电路的功能损耗。

- **普通模式：系统高速工作模式；**
- **低速模式：系统低速工作模式；**
- **省电模式：系统省电模式（睡眠模式）；**
- **绿色模式：系统理想模式。**

工作模式控制框图



工作模式时钟控制表

工作模式	普通模式	低速模式	绿色模式	睡眠模式
IHRC	运行	STPHX	STPHX	停止
ILRC	运行	运行	运行	停止
CPU 指令	执行	执行	停止	停止
T0 定时器	T0ENB	T0ENB	T0ENB	无效
TC0 定时器	TC0ENB	TC0ENB	TC0ENB （PWM 有效）	无效
TC1 定时器	TC1ENB	TC1ENB	TC1ENB （PWM 有效）	无效
T1 定时器	T1ENB	T1ENB	T1ENB	无效
看门狗定时器	Watch_Dog 编译选项	Watch_Dog 编译选项	Watch_Dog 编译选项	Watch_Dog 编译选项
内部中断	全部有效	全部有效	T0	全部无效
外部中断	全部有效	全部有效	全部有效	全部无效
唤醒功能	-	-	P0, P1, T0, CMP0, 复位	P0, P1, 复位

- **IHRC：内部高速 RC 振荡器。**
- **ILRC：内部低速 RC 振荡器。**

5.2 普通模式

普通模式是系统高速时钟正常工作模式，系统时钟源由高速振荡器提供。程序被执行。上电复位或任意一种复位触发后，系统进入普通模式执行程序。当系统从睡眠模式被唤醒后进入普通模式。普通模式下，高速振荡器正常工作，功耗最大。

- 程序被执行，所有的功能都可控制。
- 系统速率为高速。
- 内部高速振荡器和内部低速 RC 振荡器都正常工作。
- 通过 OSCM 寄存器，系统可以从普通模式切换到其它任何一种工作模式。
- 系统从睡眠模式唤醒后进入普通模式。
- 低速模式可以切换到普通模式。
- 从普通模式切换到绿色模式，唤醒后返回到普通模式。

5.3 低速模式

低速模式为系统低速时钟正常工作模式。系统时钟源由内部低速 RC 振荡器提供。低速模式由 OSCM 寄存器的 CLKMD 位控制。当 CLKMD=0 时，系统为普通模式；当 CLKMD=1 时，系统进入低速模式。切换进入低速模式后，不能自动禁止高速振荡器，必须通过 SPTHX 位来禁止以减少功耗。低速模式下，系统速率被固定为 Flosc/4（Flosc 为内部低速 RC 振荡器频率）。

- 程序被执行，所有的功能都可控制。
- 系统速率位低速（Flosc/4）。
- 内部低速 RC 振荡器正常工作，高速振荡器由 STPHX=1 控制。低速模式下，强烈建议停止高速振荡器。
- 通过 OSCM 寄存器，低速模式可以切换进入其它的工作模式。
- 从低速模式切换到睡眠模式，唤醒后返回到普通模式。
- 普通模式可以切换进入低速模式。
- 从低速模式切换到绿色模式，唤醒后返回到低速模式。

5.4 睡眠模式

睡眠模式是系统的理想状态，不执行程序，振荡器也停止工作。整个芯片的功耗低于 1uA。睡眠模式可以由 P0、P1 的电平变换触发唤醒。P1 的唤醒功能由 P1W 寄存器控制。从任何工作模式进入睡眠模式，被唤醒后都返回到普通模式。由 OSCM 寄存器的 CPUM0 位控制是否进入睡眠模式，当 CPUM0=1，系统进入睡眠模式。当系统从睡眠模式被唤醒后，CPUM0 被自动禁止（0 状态）。

- 程序停止执行，所有的功能被禁止。
- 所有的振荡器，包括内部高速振荡器和内部低速振荡器都停止工作。
- 功耗低于 1uA。
- 系统从睡眠模式被唤醒后进入普通模式。
- 睡眠模式的唤醒源为 P0 和 P1 电平变换触发。

✷ **注：普通模式下，设置 STPHX=1 禁止高速时钟振荡器，这样，无系统时钟在执行，此时系统进入睡眠模式，可以由 P0、P1 电平变换触发唤醒。**

5.5 绿色模式

绿色模式是另外的一种理想状态。在睡眠模式下，所有的功能和硬件设备都被禁止，但在绿色模式下，系统时钟保持工作，绿色模式下的功耗大于睡眠模式下的功耗。绿色模式下，不执行程序，但具有唤醒功能的定时器仍正常工作，定时器的时钟源为仍在工作的系统时钟。绿色模式下，有 2 种方式可以将系统唤醒：1、P0 和 P1 电平变换触发；2、具有唤醒功能的定时器溢出，这样，用户可以给定时器设定固定的周期，系统就在溢出时被唤醒。由 OSCM 寄存器 CPUM1 位决定是否进入绿色模式，当 CPUM1=1，系统进入绿色模式。当系统从绿色模式下被唤醒后，自动禁止 CPUM1（0 状态）。

- 程序停止执行，所有的功能被禁止。
- 具有唤醒功能的定时器正常工作。
- 作为系统时钟源的振荡器正常工作，其它的振荡器工作状态取决于系统工作模式的配置。
- 由普通模式切换到绿色模式，被唤醒后返回到普通模式。
- 由低速模式切换到绿色模式，被唤醒后返回到低速模式。
- 绿色模式下的唤醒方式为 P0、P1 电平变换触发唤醒或者指定的定时器溢出。
- 绿色模式下 PWM 和 Buzzer 功能仍然有效，但是定时器溢出时不能唤醒系统。

✷ **注：sonix 提供宏“GreenMode”来控制绿色模式的工作状态，必要时使用宏“GreeMode”进绿色模式。该宏共有 3 条指令。但在使用 BRANCH 指令（如 BTS0、BTS1、B0BTS0、B0BTS1、INCS、INCMS、DECS、DECMS、CMPRS、JMP）时必须注意宏的长度，否则程序会出错。**

5.6 工作模式控制宏

Sonix 提供工作模式控制宏以方便系统工作模式的切换。

宏名称	长度	说明
SleepMode	1-word	系统进入睡眠模式。
GreenMode	3-word	系统进入绿色模式。
SlowMode	2-word	系统进入低速模式并停止高速振荡器。
Slow2Normal	5-word	系统从低速模式返回到普通模式。该宏包括工作模式的切换，使能高速振荡器，高速振荡器唤醒延迟时间。

➢ **例：从普通模式/低速模式切换进入睡眠模式。**

```
SleepMode                       ; 直接宣告“SleepMode”宏。
```

➢ **例：从普通模式切换进入低速模式。**

```
SlowMode                        ; 直接宣告“SlowMode”宏。
```

➢ **例：从低速模式切换进入普通模式（外部高速振荡器停止工作）。**

```
Slow2Normal                     ; 直接宣告“Slow2Normal”宏。
```

➢ **例：从普通/低速模式切换进入绿色模式。**

```
GreenMode                       ; 直接宣告“GreenMode”宏。
```

➢ **例：从普通/低速模式切换进入绿色模式，并使能 T0 唤醒功能。**

```
; 设置定时器 T0 的唤醒功能。
        B0BCLR      FT0IEN      ; 禁止 T0 中断。
        B0BCLR      FT0ENB      ; 禁止 T0 定时器。
        MOV         A,#20H      ;
        B0MOV       T0M,A       ; 设置 T0 时钟= Fcpu / 64。
        MOV         A,#74H
        B0MOV       T0C,A       ; 设置 T0C 的初始值= 74H（设置 T0 间隔值 = 10 ms）。
        B0BCLR      FT0IEN      ; 禁止 T0 中断。
        B0BCLR      FT0IRQ      ; 清 T0 中断请求。
        B0BSET      FT0ENB      ; 使能 T0 定时器。

; 进入绿色模式。
        GreenMode               ; 直接宣告“GreenMode”宏。
```

5.7 系统唤醒

5.7.1 概述

睡眠模式和绿色模式下，系统并不执行程序。唤醒触发信号可以将系统唤醒进入普通模式或低速模式。唤醒触发信号包括：外部触发信号（P0、P1 的电平变换）和内部触发（T0 定时器溢出）。

- **从睡眠模式唤醒后只能进入普通模式，且将其唤醒的触发只能是外部触发信号（P0、P1 电平变化）；**
- **如果是将系统由绿色模式唤醒返回到上一个工作模式（普通模式或低速模式），则可以是外部触发信号（P0、P1 电平变换）和内部触发信号（T0 溢出）。**

5.7.2 唤醒时间

系统进入睡眠模式后，高速时钟振荡器停止运行。把系统从睡眠模式唤醒时，单片机需要等待 32 个内部高速时钟周期的时间，以等待振荡电路稳定工作，等待的这一段时间就称为唤醒时间。唤醒时间结束后，系统进入普通模式。

✷ **注：从绿色模式下唤醒系统不需要唤醒时间，因为系统时钟在绿色模式下仍然正常工作。**

唤醒时间的计算如下：

唤醒时间 = 1/Fosc * 32 (sec) + 高速时钟启动时间

➢ **例：睡眠模式下，系统被唤醒进入普通模式。唤醒时间的计算如下：**

唤醒时间 = 1/Fosc * 32 = 2us（Fosc = 16MHz）

5.7.3 P1W唤醒控制寄存器

在绿色模式和睡眠模式下，有唤醒功能的 I/O 口能够将系统唤醒到普通模式。P0 和 P1 都有唤醒功能，二者的区别在于，P0 的唤醒功能始终有效，而 P1 由寄存器 P1W 控制。

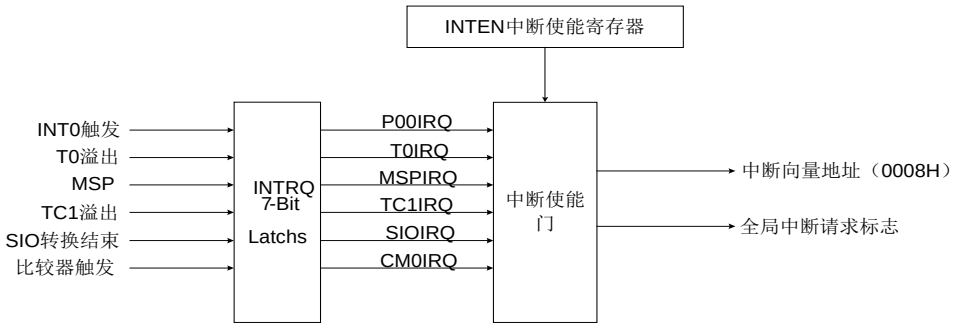
0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

Bit[3:0] **P10W~P17W**：P1 唤醒功能控制位。
0 = 禁止 P1n 唤醒功能；
1 = 使能 P1n 唤醒功能。

6 中断

6.1 概述

SN8P2524 提供 6 种中断源：5 个内部中断（T0/TC1/CM0/SIO/MSP）和 1 个外部中断（INT0）。外部中断可以将系统从睡眠模式唤醒进入高速模式，在返回高速模式前，外部中断请求被锁定。一旦程序进入中断，寄存器 STKP 的位 GIE 将被硬件自动清零以避免再次响应其它中断。系统退出中断，即执行完 RETI 指令后，硬件自动将 GIE 置“1”，以响应下一个中断。中断请求存放在寄存器 INTRQ 中。



✷ **注：程序响应中断时，位 GIE 必须处于有效状态。**

6.2 INTEN中断使能寄存器

中断使能寄存器 INTEN 包括所有中断的使能控制位。INTEN 的有效位被置为“1”就使能了其相应的中断请求功能。一旦中断发生，程序进行压栈并跳转到中断向量（0008H）处执行中断服务程序。程序运行到指令 RETI 时，中断结束，系统退出中断服务。

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	CM0IEN	TC1IEN	-	T0IEN	SIOIEN	-	MSPIEN	P00IEN
读/写	R/W	R/W	-	R/W	R/W	-	R/W	R/W
复位后	0	0	-	0	0	-	0	0

Bit 0 **P00IEN**：P0.0 外部中断（INT0）控制位。
0 = 禁止；
1 = 使能。

Bit 1 **MSPIEN**：MSP 中断控制位。
0 = 禁止；
1 = 使能。

Bit 3 **SIOIEN**：SIO 中断控制位。
0 = 禁止；
1 = 使能。

Bit 4 **T0IEN**：T0 中断控制位。
0 = 禁止；
1 = 使能。

Bit 6 **TC1IEN**：TC1 中断控制位。
0 = 禁止；
1 = 使能。

Bit 7 **CM0IEN**：CM0 中断控制位。
0 = 禁止；
1 = 使能。

6.3 INTRQ中断请求寄存器

中断请求寄存器 INTRQ 中存放各中断请求标志。一旦有中断请求发生，INTRQ 中的相应位将被置“1”，该请求被响应后，程序应将该标志位清零。根据 INTRQ 的状态，程序判断是否有中断发生，并执行相应的中断服务。

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	CM0IRQ	TC1IRQ	-	T0IRQ	SIOIRQ	-	MSPIRQ	P00IRQ
读/写	R/W	R/W	-	R/W	R/W	-	R/W	R/W
复位	0	0	-	0	0	-	0	0

Bit 0 **P00IRQ**：P0.0 中断（INT0）请求标志位。
0 = INT0 无中断请求；
1 = INT0 有中断请求。

Bit 1 **MSPIRQ**：MSP 中断请求标志位。
0 = MSP 无中断请求；
1 = MSP 有中断请求。

Bit 3 **SIOIRQ**：SIO 中断请求标志位。
0 = SIO 无中断请求；
1 = SIO 有中断请求。

Bit 4 **T0IRQ**：T0 中断请求标志位。
0 = T0 无中断请求；
1 = T0 有中断请求。

Bit 6 **TC1IRQ**：TC1 中断请求标志位。
0 = TC1 无中断请求；
1 = TC1 有中断请求。

Bit 7 **CM0IRQ**：CM0 中断请求标志位。
0 = CM0 无中断请求；
1 = CM0 有中断请求。

6.4 全局中断GIE

只有当全局中断控制位 GIE 置“1”的时候程序才能响应中断请求。一旦有中断发生，程序计数器（PC）指向中断向量地址（0008H），堆栈层数加 1。

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0
读/写	R/W	-	-	-	-	R/W	R/W	R/W
复位	0	-	-	-	-	1	1	1

Bit 7 **GIE**：全局中断控制位。
0 = 禁止全局中断；
1 = 使能全局中断。

➢ **例：设置全局中断控制位（GIE）。**

```
B0BSET      FGIE        ; 使能 GIE。
```

✷ **注：在所有中断中，GIE 都必须处于使能状态。**

6.5 PUSH, POP

有中断请求发生并被响应后，程序转至 0008H 执行中断子程序。在响应中断之前，必须保存 ACC 和 PFLAG 的内容。系统提供 PUSH 和 POP 指令进行入栈保护和出栈恢复。

✸ **注： PUSH、POP 指令保存和恢复 ACC/PFLAG（不包括 NT0、NPD）的内容。PUSH/POP 缓存器只有一层。**

➢ **例：用 PUSH、POP 指令来保护和恢复 ACC 和 PFLAG。**

```
                ORG     0
                JMP     START

                ORG     8
                JMP     INT_SERVICE

                ORG     10H
START:
                …

INT_SERVICE:
                PUSH                    ; 保存 ACC 和 PFLAG。
                …
                …
                POP                     ; 恢复 ACC 和 PFALG。

                RETI                    ; 退出中断。
                …
                ENDP
```

6.6 外部中断INT0

SN8P2524 只有 1 个外部中断 INT0。当外部中断被触发时，不管外部中断使能控制位是否使能，外部中断请求标志位都置 1，若使能外部中断控制位时，则程序跳到中断向量处开始执行中断服务程序。

外部中断内置唤醒锁存功能，就是指系统从睡眠模式唤醒，唤醒源为中断源（P0.0）。触发沿的方向与中断沿的配置相互配合。当触发沿被锁存后，系统被唤醒后先执行中断服务程序。

0BFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE	-	-	-	P00G1	P00G0	-	-	-
读/写	-	-	-	R/W	R/W	-	-	-
复位后	-	-	-	1	0	-	-	-

Bit[4:3] **P00G[1:0]**：INT0 触发方向选择位。
00 = 保留；
01 = 上升沿；
10 = 下降沿；
11 = 电平变换。

➢ **例：INT0 中断请求设置，电平触发。**

```
        MOV     A, #18H
        B0MOV   PEDGE, A        ; INT0 置为电平触发。

        B0BCLR  FP00IRQ         ; INT0 中断请求标志清零。
        B0BSET  FP00IEN         ; 使能 INT0 中断。
        B0BSET  FGIE            ; 使能 GIE。
```

➢ **例：INT0 中断。**

```
        ORG     8H              ;
        JMP     INT_SERVICE
INT_SERVICE:

        …                       ; ACC 和 PFLAG 入栈保护。

        B0BTS1  FP00IRQ         ; 检测 P00IRQ。
        JMP     EXIT_INT        ; P00IRQ = 0，退出中断。

        B0BCLR  FP00IRQ         ; P00IRQ 清零。
        …                       ; INT1 中断服务程序。
        …
EXIT_INT:
        …                       ; ACC 和 PFLAG 出栈恢复。

        RETI                    ; 退出中断。
```

6.7 T0 中断

T0C 溢出时，无论 T0IEN 处于何种状态，T0IRQ 都会置“1”。若 T0IEN 和 T0IRQ 都置“1”，系统就会响应 T0 的中断；若 T0IEN = 0，则无论 T0IRQ 是否置“1”，系统都不会响应 T0 中断。尤其需要注意多种中断下的情形。

➢ **例：设置 T0 中断。**

```
        B0BCLR      FT0IEN          ; 禁止 T0 中断。
        B0BCLR      FT0ENB          ; 关闭 T0。
        MOV         A, #20H         ;
        B0MOV       T0M, A          ; 设置 T0 时钟= Fcpu / 64。
        MOV         A, #74H         ; 初始化 T0C = 74H。
        B0MOV       T0C, A          ; 设置 T0 间隔时间= 10 ms。

        B0BCLR      FT0IRQ          ; T0IRQ 清零。
        B0BSET      FT0IEN          ; 使能 T0 中断。
        B0BSET      FT0ENB          ; 开启定时器 T0。

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：T0 中断服务程序。**

```
        ORG         8H
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; 保存 ACC 和 PFLAG。

        B0BTS1      FT0IRQ          ; 检查是否有 T0 中断请求标志。
        JMP         EXIT_INT        ; T0IRQ = 0，退出中断。

        B0BCLR      FT0IRQ          ; 清 T0IRQ。
        MOV         A, #74H
        B0MOV       T0C, A          ;
        …
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

6.8 TC1 中断

TC1C 溢出时，无论 TC1IEN 处于何种状态，TC1IRQ 都会置“1”。若 TC1IEN 和 TC1IRQ 都置“1”，系统就会响应 TC1 的中断；若 TC1IEN = 0，则无论 TC1IRQ 是否置“1”，系统都不会响应 TC1 中断。尤其需要注意多种中断下的情形。

➢ **例：TC1 中断请求设置。Fcpu = 16MHz / 16。**

```
        B0BCLR      FTC1IEN         ; 禁止 TC1 中断。
        B0BCLR      FTC1ENB         ;
        MOV         A, #20H         ; Fcpu=Fhosc/16=16MHz/16=1MHz。
        B0MOV       TC1M, A         ; TC1 时钟＝Fcpu / 64。
        MOV         A, # 64H        ; TC1C 初始值＝64H。
        B0MOV       TC1C, A         ; TC1 间隔= 10 ms。

        B0BCLR      FTC1IRQ         ; 清 TC1 中断请求标志。
        B0BSET      FTC1IEN         ; 使能 TC1 中断。
        B0BSET      FTC1ENB         ;

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：TC1 中断服务程序。**

```
        ORG         8H              ;
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; 保存 ACC 和 PFLAG。

        B0BTS1      FTC1IRQ         ; 检查是否有 TC1 中断请求标志。
        JMP         EXIT_INT        ; TC1IRQ = 0，退出中断。

        B0BCLR      FTC1IRQ         ; 清 TC1IRQ。
        MOV         A, #74H
        B0MOV       TC1C, A         ; TC1C 赋初始值。
        …                           ; TC1 中断程序。
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

6.9 SIO中断

当 SIO 完成数据传送后，无论 SIOIEN 处于何种状态，SIOIRQ 都会被置“1”。如果 SIOIRQ＝1 且 SIOIEN＝1，系统响应该中断；如果 SIOIRQ＝1 而 SIOIEN＝0，系统并不会执行中断服务。在处理多中断时尤其需要注意。

➢ **例：SIO 中断请求设置。**

```
        B0BCLR      FSIOIRQ         ; 清 SIO 中断请求标志。
        B0BSET      FSIOIEN         ; 使能 SIO 中断。
        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：SIO 中断服务程序。**

```
        ORG         8H              ;
        JMP         INT_SERVICE
INT_SERVICE:
        …                           ; ACC 和 PFLAG 入栈保护。

        B0BTS1      FSIOIRQ         ; 检查是否有 SIO 中断请求标志。
        JMP         EXIT_INT        ; SIOIRQ = 0，退出中断。

        B0BCLR      FSIOIRQ         ; 清 SIOIRQ。
        …                           ; SIO 中断服务程序。
        …
EXIT_INT:
        …                           ; ACC 和 PFLAG 出栈恢复

        RETI                        ; 退出中断
```

6.10 比较器中断（CMP0）

SN8P2524 内建 1 个比较器中断（CMP0）。比较器的中断触发方向由 CM0G 标志位决定，当比较器输出状态开始转变时，不管比较器中断使能控制位的状态，比较器的中断请求控制位都会置 1。当使能比较器中断使能控制位且比较器中断请求控制位置 1 时，程序会跳转到中断向量 0008 处开始执行中断服务程序。当禁止比较器中断使能控制位时，比较器中断请求控制位会处于无效状态，此时就不执行中断服务。

➢ **例：设置 CMP0 中断。**

```
        B0BSET      FCM0IEN         ; 使能 CMP0 中断。
        B0BCLR      FCM0IRQ         ; 清 CMP0 中断请求。
        B0BSET      FCM0EN
        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：CMP0 中断服务程序。**

```
        ORG         8               ; 中断向量。
        JMP         INT_SERVICE
INT_SERVICE:
        …                           ; 保存 ACC 和 PFLAG。

        B0BTS1      FCM0IRQ         ; 检查 CM0IRQ 状态。
        JMP         EXIT_INT        ; 若 CMP0IRQ = 0，退出中断。

        B0BCLR      FCM0IRQ         ; 复位 CM0IRQ。
        …                           ; CMP0 中断服务程序。
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。
        RETI                        ; 退出中断。
```

7 I/O口

7.1 概述

SN8P2524 共有 22 个 I/O 引脚，大多数 I/O 引脚与模拟引脚及特殊功能的引脚共用，详见下表：

I/O 引脚		共用引脚		引脚共用控制条件
引脚名称	引脚类型	引脚名称	引脚类型	
P0.0	I/O	INT0	DC	P00IEN=1
		CM0O	AC	CM0EN=1, CM0OEN=1
P0.1	O	LPW0	DC	TC0ENB=1, LPWS0=1
		CM0N12	AC	CM0EN=1, CMCH[3:0] = 1100
P0.2	O	LPW1	DC	TC0ENB=1, LPWS1=1
		CM0N13	AC	CM0EN=1, CMCH[3:0] = 1101
P0.3	O	LPW2	DC	TC0ENB=1, LPWS2=1
		CM0N14	AC	CM0EN=1, CMCH[3:0] = 1110
P0.4	O	LPW3	DC	TC0ENB=1, LPWS3=1
		CM0N15	AC	CM0EN=1, CMCH[3:0] = 1111
P0.5	I	RST	DC	Reset_Pin code option = Reset
		VPP	HV	OTP Programming
P1.0	I/O	CM0N0	AC	CM0EN=1, CMCH[3:0] = 0000
P1.1	I/O	CM0N1	AC	CM0EN=1, CMCH[3:0] = 0001
P1.2	I/O	CM0N2	AC	CM0EN=1, CMCH[3:0] = 0010
P1.3	I/O	CM0N3	AC	CM0EN=1, CMCH[3:0] = 0011
P1.4	I/O	CM0N4	AC	CM0EN=1, CMCH[3:0] = 0100
P1.5	I/O	CM0N5	AC	CM0EN=1, CMCH[3:0] = 0101
P1.6	I/O	CM0N6	AC	CM0EN=1, CMCH[3:0] = 0110
P1.7	I/O	CM0N7	AC	CM0EN=1, CMCH[3:0] = 0111
P5.0	I/O	SCK	DC	SENB=1
		SCL	DC	MSPENB=1
P5.1	I/O	SI	DC	SENB=1
		SDA	DC	MSPENB=1
P5.2	I/O	SO	DC	SENB=1
P5.3	I/O	PWM1	DC	TC1ENB=1, PWM1OUT=1
P5.4	I/O	LPW4	DC	TC0ENB=1, LPWS4=1
		CM0N8	AC	CM0EN=1, CMCH[3:0] = 1000
P5.5	I/O	LPW5	DC	TC0ENB=1, LPWS5=1
		CM0N9	AC	CM0EN=1, CMCH[3:0] = 1001
P5.6	I/O	LPW6	DC	TC0ENB=1, LPWS6=1
		CM0N10	AC	CM0EN=1, CMCH[3:0] = 1010
P5.7	I/O	LPW7	DC	TC0ENB=1, LPWS7=1
		CM0N11	AC	CM0EN=1, CMCH[3:0] = 1011

*** DC：数字特性；AC：模拟特性；HV：高压特性。**

7.2 I/O口模式

寄存器 PnM 控制 I/O 口的工作模式。

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	-	-	-	P04M	P03M	P02M	P01M	P00M
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1M	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **PnM[7:0]**：Pn 模式控制位（n = 0~5）。
0 = 输入模式；
1 = 输出模式。

✸ **注：用户可通过位操作指令（B0BSET、B0BCLR）对 I/O 口进行编程控制；**
✸ **注：P0.5 是单向输入引脚，P0M.5 保持为 1。**

➢ **例：I/O 模式选择。**

```
CLR     P0M         ; 设置为输入模式。
CLR     P1M
CLR     P5M

MOV     A, #0FFH    ; 设置为输出模式。
B0MOV   P0M, A
B0MOV   P1M,A
B0MOV   P5M, A

B0BCLR  P1M.0       ; P1.0 设为输入模式。

B0BSET  P1M.0       ; P1.0 设为输出模式。
```

7.3 I/O口上拉电阻寄存器

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	-	-	-	-	-	-	-	P00R
读/写	-	-	-	-	-	-	-	W
复位后	-	-	-	-	-	-	-	0

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0E5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5UR	P57R	P56R	P55R	P54R	P53R	P52R	P51R	P50R
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

✸ **注：P0.5 为单向输入引脚，无上拉电阻，故 P0UR.5 保持为 1。**

➢ **例：I/O 口的上拉电阻。**

```
MOV     A, #0FFH        ; 使能 P0、P1、P5 的上拉电阻。
B0MOV   P0UR, A         ;
B0MOV   P1UR,A
B0MOV   P5UR, A
```

7.4 I/O口数据寄存器

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	-	-	P05	P04	P03	P02	P01	P00
读/写	-	-	R	R/W	R/W	R/W	R/W	R/W
复位后	-	-	0	0	0	0	0	0

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1	P17	P16	P15	P14	P13	P12	P11	P10
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	P57	P56	P55	P54	P53	P52	P51	P50
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

✸ **注：当通过编译选项使能外部复位时，P05 保持为 1。**

➢ **例：读取输入口的数据。**

```
B0MOV   A, P0       ; 读取 P0、P1 和 P5 口的数据。
B0MOV   A, P1
B0MOV   A, P5
```

➢ **例：写入数据到输出端口。**

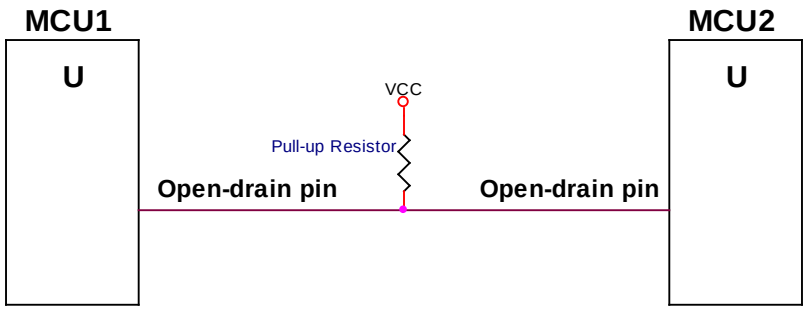
```
MOV     A, #0FFH    ; 写入数据 FFH 到 P0、P1 和 P5。
B0MOV   P0, A
B0MOV   P1, A
B0MOV   P5, A
```

➢ **例：写入 1 位数据到输出端口。**

```
B0BSET  P1.0        ; P1.0 和 P5.3 置 1。
B0BSET  P5.3

B0BCLR  P1.0        ; P1.0 和 P5.3 清 0。
B0BCLR  P5.3
```

7.5 I/O漏极开路寄存器

P5.0~P5.2 内置漏极开路功能，当使能该功能时，P5.0~P5.2 必须设置为输出模式。漏极开路的外部电路如下图所示：



上图中的上拉电阻必不可少，漏极开路的输出高电平由上拉电阻驱动。

0E9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1OC	-	-	-	P52OC	P51OC	P50OC	-	-
读/写	-	-	-	W	W	W	-	-
复位后	-	-	-	0	0	0	-	-

Bit 2 **P50OC**：P5.0 漏极开路控制位。
0 = 禁止；
1 = 使能。

Bit 3 **P51OC**：P5.1 漏极开路控制位。
0 = 禁止；
1 = 使能。

Bit 4 **P52OC**：P5.2 漏极开路控制位。
0 = 禁止；
1 = 使能。

➢ **例：开启 P5.0 的漏极开路功能并输出高电平。**

```
B0BSET      P5.0          ; P5.0 置高。

B0BSET      P50M          ; P5.0 设为输出模式。
MOV         A, #04H       ; 开启 P5.0 的漏极开路功能。
B0MOV       P1OC, A
```

✷ **注：P1OC 是只写寄存器，只能通过指令“MOV”设置 P1OC。**

➢ **例：禁止漏极开路功能。**

```
MOV         A, #0         ; 禁止漏极开路功能。
B0MOV       P1OC, A
```

✷ **注：禁止漏极开路功能后，I/O 引脚恢复到之前的 I/O 模式。**

8 定时器

8.1 看门狗定时器

看门狗定时器 WDT 是一个 4 位二进制计数器，用于监控程序的正常执行。如果由于干扰，程序进入了未知状态，看门狗定时器溢出，系统复位。看门狗的工作模式由编译选项控制，其时钟源由内部低速 RC 振荡器提供，它是将内部 RC 振荡器经 512 分频后送往看门狗定时器进行计数。

看门狗溢出时间 = 8192 /内部低速振荡器周期（sec）

VDD	内部低速 RC 频率	看门狗溢出时间
3V	16KHz	512ms
5V	32KHz	256ms

看门狗定时器的 3 种工作模式由编译选项“WatchDog”控制：

- **Disable**：禁止看门狗定时器功能。
- **Enable**：使能看门狗定时器功能，在普通模式和低速模式下有效，在睡眠模式和绿色模式下看门狗停止工作。
- **Always_On**：使能看门狗定时器功能，在睡眠模式和绿色模式下，看门狗仍会正常工作。

在高干扰环境下，强烈建议将看门狗设置为“Always_On”以确保系统在出错状态和重启时正常复位。

看门狗清零的方法是对看门狗计数器清零寄存器 WDTR 写入清零控制字 5AH。

0CCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WDTR	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

➢ **例：下面是对看门狗定时器操作的示例程序，在主程序的开始清看门狗定时器。**

```
Main:
        MOV     A, #5AH         ; 清看门狗定时器。
        B0MOV   WDTR, A
        …
        CALL    SUB1
        CALL    SUB2
        …
        JMP     Main
```

➢ **例：用宏指令@RST_WDT 清看门狗定时器。**

```
Main:
        @RST_WDT                ; 清看门狗定时器。
        …
        CALL    SUB1
        CALL    SUB2
        …
        JMP     Main
```

看门狗定时器应用注意事项如下：

- 对看门狗清零之前，检查 I/O 口的状态和 RAM 的内容可增强程序的可靠性；
- 不能在中断中对看门狗清零，否则无法侦测到主程序跑飞的状况；
- 程序中应该只在主程序中有一次清看门狗的动作，这种架构能够最大限度的发挥看门狗的保护功能。

➢ **例：下面是对看门狗定时器操作的示例程序，在主程序的开始清看门狗定时器。**

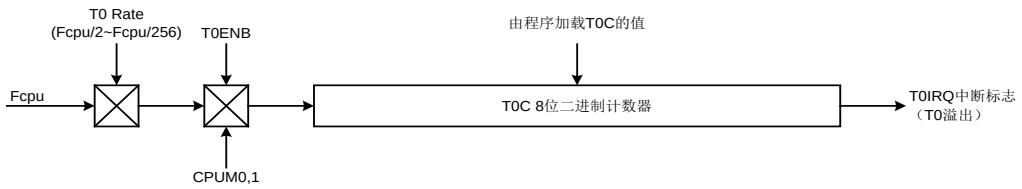
```
Main:
        …                       ; 检查 I/O 口的状态。
        …                       ; 检查 RAM 的内容。
Err:    JMP $                   ; I/O 口或 RAM 出错，不清看门狗等看门狗计时溢出。
Correct:                        ; I/O 口和 RAM 都正确，清看门狗定时器。
        MOV     A, #5AH         ; 清看门狗定时器。
        B0MOV   WDTR, A
        …
        CALL    SUB1
        CALL    SUB2
        …
        JMP     Main
```

8.2 8 位基本定时器T0

8.2.1 概述

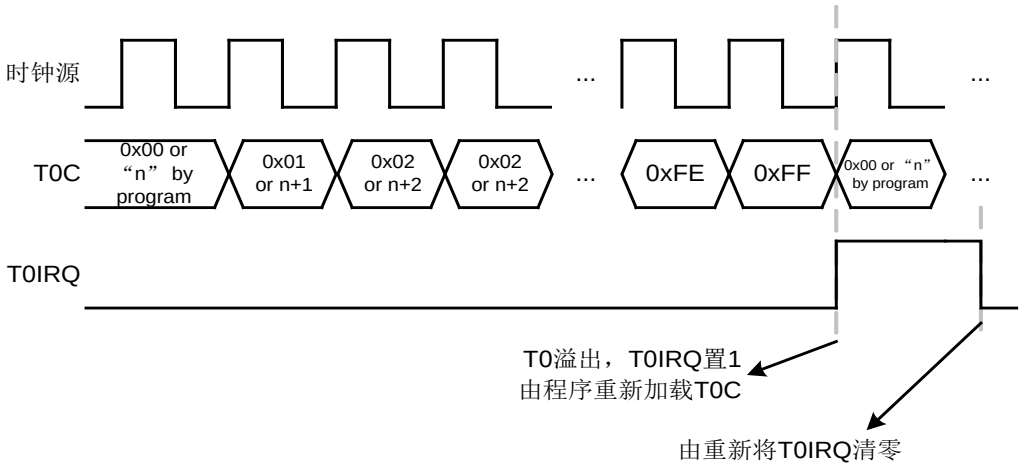
8 位二进制基本定时器 T0 具有定时器功能置：支持标志指示（T0IRQ）和中断操作（中断向量）。可以通过 T0M 和 T0C 寄存器控制间隔时间，具有在绿色模式下唤醒功能。在绿色模式下，T0 溢出，则将系统唤醒返回到上一个工作模式。

- **8 位可编程计数定时器：**根据选择的时钟频率周期性的产生中断请求。
- **中断功能：**T0 定时器支持中断功能，当 T0 溢出，T0IRQ 有效，程序计数器跳到中断向量地址执行中断。
- **绿色模式唤醒功能：**T0 定时器在绿色模式下正常工作，溢出时将系统从绿色模式下唤醒。



8.2.2 T0 操作

T0 定时器由 T0ENB 控制。当 T0ENB=0 时，T0 停止工作；当 T0ENB=1 时，T0 开始计数。T0C 溢出（从 0FFH 到 00H）时，T0IRQ 置 1 显示溢出状态并由程序清零。T0 溢出时由程序加载新值给 T0C，以选定合适的间隔时间。如果使能 T0 中断（T0IEN=1），T0 溢出后系统执行中断服务程序，在中断下必须由程序清 T0IRQ。T0 可以在普通模式、低速模式和绿色模式下工作，绿色模式下，T0 溢出时 T0IRQ 置 1，系统被唤醒。



T0 的时钟源为 Fcpu（指令周期），由 T0Rate[2:0]决定。详见下表：

T0rate[2:0]	T0 Clock	T0 Interval Time			
		Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=16MHz, Fcpu=Fhosc/16	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)
000b	Fcpu/256	16.384	64	65.536	256
001b	Fcpu/128	8.192	32	32.768	128
010b	Fcpu/64	4.096	16	16.384	64
011b	Fcpu/32	2.048	8	8.192	32
100b	Fcpu/16	1.024	4	4.096	16
101b	Fcpu/8	0.512	2	2.048	8
110b	Fcpu/4	0.256	1	1.024	4
111b	Fcpu/2	0.128	0.5	0.512	2

8.2.3 T0M模式寄存器

模式寄存器 T0M 设置 T0 的工作模式，包括 T0 前置分频器、时钟源等，这些设置必须在使能 T0 定时器之前完成。

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	-
读/写	R/W	R/W	R/W	R/W	-	-	-	-
复位后	0	0	0	0	-	-	-	-

Bit [6:4] **T0RATE[2:0]**：T0 分频选择位。
000 = fcpu/256；
001 = fcpu/128；
…；
110 = fcpu/4；
111 = fcpu/2。

Bit 7 **T0ENB**：T0 启动控制位。
0 = 禁止；
1 = 使能。

8.2.4 T0C计数寄存器

8 位计数器 T0C 溢出时，T0IRQ 置 1 并由程序清零，用来控制 T0 的中断间隔时间。首先须写入正确的值到 T0C 寄存器，并使能 T0 定时器以保证第一个周期正确。T0 溢出后，由程序装入正确的值到 T0C。

0D9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0C	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

T0C 初始值的计算公式如下：

T0C 初始值 = 256 -（T0 中断间隔时间 * 时钟 rate）

- **例：T0 的中断间隔时间为 10ms，高速时钟为内部 16MHz，Fcpu = Fosc/16=16MHz/16=1MHz，T0RATE = 001（Fcpu/128）。**

T0 间隔时间=10ms，T0 时钟 rate=16MHz/16/128

T0C 初始值 = 256 -（T0 中断间隔时间 * 输入时钟）
= 256 -（10ms * 16MHz / 16 / 128）
= 256 -（10-2 * 16 * 106 / 16 /128）
= B2H

8.2.5 T0 操作举例

- **T0 定时器：**

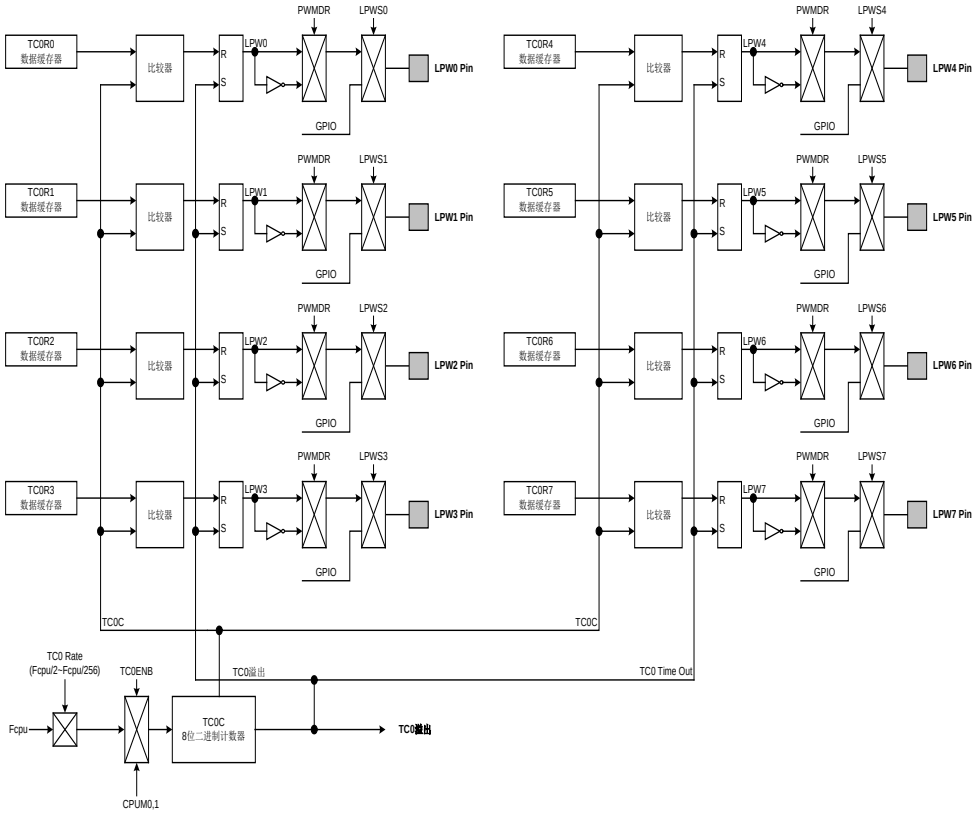
```
; 复位 T0 定时器。
        CLR     T0M             ; 清 T0M。

; 设置 T0 时钟源和 T0Rate。
        MOV     A, #0nnn0000b
        B0MOV   T0M, A

; 设置 T0C 寄存器获取 T0 间隔时间。
        MOV     A, #value
        B0MOV   T0C, A

; 清 T0IRQ。
        B0BCLR  FT0IRQ

; 使能 T0 定时器和中断功能。
        B0BSET  FT0IEN          ; 使能 T0 中断。
        B0BSET  FT0ENB          ; 使能 T0 定时器。
```

8.3 8 通道LED PWM驱动（TC0）

8.3.1 概述

8 位二进制定时器 TC0 的主要功能为产生 8 通道 PWM 输出。TC0 不支持普通定时器功能和中断功能。TC0C 256-step 决定 PWM 的周期，通过 TC0rate[2:0]设置 TC0 时钟 rate。TC0R0~TC0R7 决定 8 个 PWM 通道的占空比。TC0C=0 时，PWM 输出脉冲，TC0C=TC0R 时，PWM 切换为空闲状态，以产生 PWM 占空比。PWM 的相位由 PWMDR 位控制，PWMDR=0 时，PWM 的状态为低电平空闲，记录高电平占空比；PWMDR=1 时，PWM 的状态为高电平空闲，记录低电平的占空比。TC0 的主要功能如下：TC0Rn 非自动装载设计，如果在 PWM 输出时通过 TC0Rn 寄存器修改 PWM 的占空比，则 PWM 的输出波形会随之更改。

- **PWM 输出：**PWM 的占空比/周期由 TC0rate、TC0Rn 和 LPWS 寄存器编程控制。
- **绿色模式功能：**TC0 的 PWM 功能在绿色模式下依然有效。



8.3.2 TC0M控制寄存器

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	-	-	-	PWMDR
读/写	R/W	R/W	R/W	R/W	-	-	-	R/W
复位后	0	0	0	0	-	-	-	0

Bit 0 **PWMDR**：PWM 输出相位控制位。
0 = 高电平脉冲和低电平空闲状态；
1 = 低电平脉冲和高电平空闲状态。

Bit [6:4] **TC0RATE[2:0]**：TC0 定时器时钟源选择控制位。
000 = Fcpu/256；
001 = Fcpu/128；
010 = Fcpu/64；
011 = Fcpu/32；
100 = Fcpu/16；
101 = Fcpu/8；
110 = Fcpu/4；
111 = Fcpu/2。

Bit 7 **TC0ENB**：TC0 计数器控制位。
0 = 禁止；
1 = 使能。

0DBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0C	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0A8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R0	TC0R07	TC0R06	TC0R05	TC0R04	TC0R03	TC0R02	TC0R01	TC0R00
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0A9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R1	TC0R17	TC0R16	TC0R15	TC0R14	TC0R13	TC0R12	TC0R11	TC0R10
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0AAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R2	TC0R27	TC0R26	TC0R25	TC0R24	TC0R23	TC0R22	TC0R21	TC0R20
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0ABH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R3	TC0R37	TC0R36	TC0R35	TC0R34	TC0R33	TC0R32	TC0R31	TC0R30
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0ACH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R4	TC0R47	TC0R46	TC0R45	TC0R44	TC0R43	TC0R42	TC0R41	TC0R40
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0ADH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R5	TC0R57	TC0R56	TC0R55	TC0R54	TC0R53	TC0R52	TC0R51	TC0R50
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0AEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R6	TC0R67	TC0R66	TC0R65	TC0R64	TC0R63	TC0R62	TC0R61	TC0R60
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0AFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R7	TC0R77	TC0R76	TC0R75	TC0R74	TC0R73	TC0R72	TC0R71	TC0R70
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

TC0R 初始值 = 256 -（TC0 中断间隔时间 * TC0 时钟 rate）

- **例：TC0 的间隔时间为 10ms，时钟源来自 Fcpu= 16MHz/16=1MHz，TC0RATE=001（Fcpu/128）。**
  TC0 脉冲宽度为 10ms，TC0 时钟 rate=16MHz/16/128。

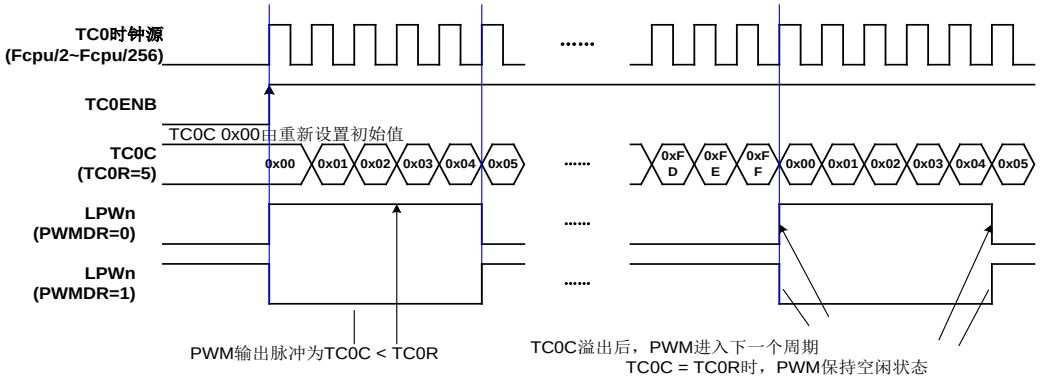
TC0R = 256 -（TC0 中断间隔时间 * 输入时钟）
= 256 -（10ms * 16MHz / 16 / 128）
= 256 -（10-2 * 16 * 106 / 16 / 128）
= B2H

0CDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LPWS	LPWS7	LPWS6	LPWS5	LPWS4	LPWS3	LPWS2	LPWS1	LPWS0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **LPWS [7:0]**：LED PWM 通道选择控制位。
LPWS0：LED PWM 通道 1。0=禁止，LED 引脚为 GPIO 模式；1=使能，LED 引脚输出 PWM 信号。
LPWS1：LED PWM 通道 2。0=禁止，LED 引脚为 GPIO 模式；1=使能，LED 引脚输出 PWM 信号。
LPWS2：LED PWM 通道 3。0=禁止，LED 引脚为 GPIO 模式；1=使能，LED 引脚输出 PWM 信号。
LPWS3：LED PWM 通道 4。0=禁止，LED 引脚为 GPIO 模式；1=使能，LED 引脚输出 PWM 信号。
LPWS4：LED PWM 通道 5。0=禁止，LED 引脚为 GPIO 模式；1=使能，LED 引脚输出 PWM 信号。
LPWS5：LED PWM 通道 6。0=禁止，LED 引脚为 GPIO 模式；1=使能，LED 引脚输出 PWM 信号。
LPWS6：LED PWM 通道 7。0=禁止，LED 引脚为 GPIO 模式；1=使能，LED 引脚输出 PWM 信号。
LPWS7：LED PWM 通道 8。0=禁止，LED 引脚为 GPIO 模式；1=使能，LED 引脚输出 PWM 信号。

8.3.3 TC0 PWM操作

TC0 PWM 的周期为 TC0C 00H~0FFH，由 TC0Rate[2:0]控制，无需设置 TC0C 初始值，只需在设置 TC0ENB 之前将 TC0C 清零。TC0 PWM 有 8 个通道，由 LPWS 寄存器控制。若 LPWS 寄存器的相关位置 1，则对应的通道切换到 LED PWM 输出模式，并忽略 GPIO 功能；若该位清零，则该通道返回到上一个 GPIO 模式。这样便于在 PWM 非输出模式下设置 PWM 空闲状态。



8.3.4 PWM操作举例

- **PWM：**

```
; 复位 TC0。
        CLR     TC0M            ; 请 TC0M。

; 设置 TC0 时钟 rate。
        MOV     A, #0nnn0000b   ; 设置 TC0RATE[2:0]。
        B0MOV   TC0M, A

        B0BCLR  FPWMDR          ; 高电平脉冲和低电平空闲状态。
; or
        B0BSET  FPWMDR          ; 低电平脉冲和高电平空闲状态。

; 设置 TC0Rn 寄存器获取 PWM 占空比。
        CLR     TC0C
        MOV     A, #value
        B0MOV   TC0Rn, A        ; 设置 PWM 占空比，n=0~7。
; 使能 PWM 和 TC0 定时器。
        B0BSET  FLPWSn          ; 使能 PWM 通道 n，n=0~7。
        B0BSET  FTC0ENB         ; 开启 TC0 定时器。
```

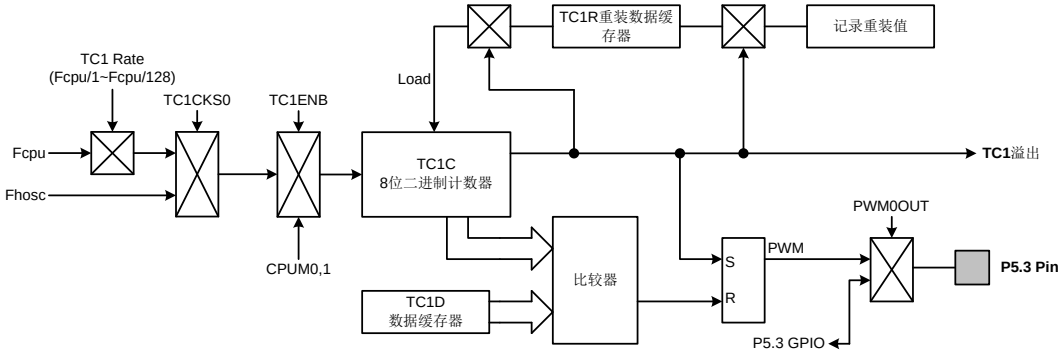
8.4 8 位定时器TC1

8.4.1 概述

8 位二进制定时器 TC1 具有基本定时器和 PWM 功能。基本定时器功能可以支持中断请求标志的显示（TC1IRQ）和中断操作（中断向量）。由 TC1M、TC1C、TC1R 寄存器控制 TC1 的中断间隔时间。TC1 还内置周期/占空比可编程控制的 PWM 功能，PWM 的周期和分辨率由 TC1 时钟周期、TC1R 和 TC1D 寄存器控制，故具有良好性能的 PWM 可以处理 IR 载波信号，马达控制和光度调节等。

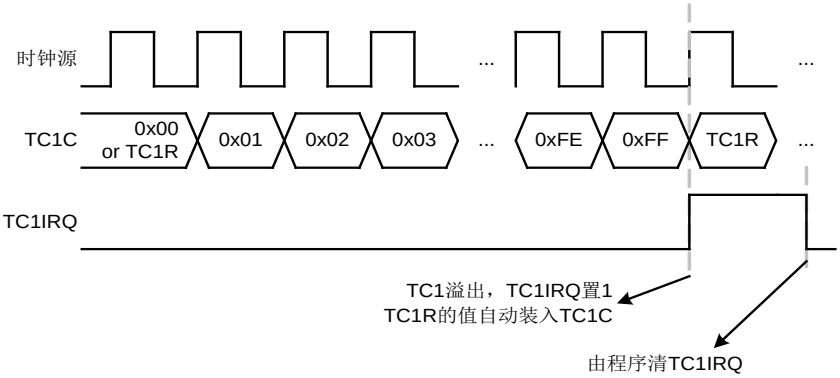
TC1 的主要用途如下：

- **8 位可编程定时器：**根据选择的时钟信号，产生周期性中断；
- **中断功能：**TC1 定时器支持中断，当 TC1 溢出时，TC1IRQ 置 1，系统执行中断；
- **可编程控制占空比/周期的 PWM 输出：**由 TC1R 和 TC1D 寄存器控制占空比/周期；
- **绿色模式功能：**绿色模式下，TC1 正常工作，但无唤醒功能。



8.4.2 TC1 操作

TC1 定时器由 TC1ENB 控制。当 TC1ENB=0 时，TC1 停止工作；当 TC1ENB=1 时，TC1 开始计数。使能 TC1 之前，先要设定好 TC1 的功能模式，如基本定时器、TC1 中断等。TC1C 溢出（从 0FFH 到 00H）时，TC1IRQ 置 1 以显示溢出状态并由程序清零。在不同的功能模式下，TC1C 不同的值对应不同的操作，若改变 TC1C 的值影响到操作，会导致功能出错。TC1 内置双重缓存器以避免此种状况的发生。在 TC1C 计数的过程中不断的刷新 TC1C，保证将最新的值存入 TC1R（重装缓存器）中，当 TC1 溢出后，新的 TC1R 值将自动装载到 TC1C。进入下一个周期后，TC1 进行新的工作状态。定时/计数器模式时，使能 TC1 时，自动使能自动重装功能。如果使能 TC1 中断功能（TC1IEN=1），在 TC1 溢出时系统执行中断服务程序，在中断时必须由程序清 TC1IRQ。TC1 可以在普通模式、低速模式和绿色模式下工作。但在绿色模式下，TC1 虽继续工作，但不能唤醒系统。



TC1 根据不同的时钟源选择不同的应用模式，TC1 的时钟源由 Fcpu（指令周期）和 Fhosc（高速振荡时钟）提供，由 TC1CKS0 控制。TC1CKS0 选择时钟源来自 Fcpu 或者 Fhosc，当 TC1CKS0=0 时，TC1 时钟源来自 Fcpu，可以由 TC1Rate[2:0]选择不同的分频，包括 Fcpu/1~Fcpu/128。当 TC1CKS0=1 时，TC1 时钟源来自 Fhosc，不分频，TC1Rate[2:0]处于无效状态。

TC1CKS0	TC1rate[2:0]	TC1 Clock	TC1 间隔时间 Fhosc=16MHz, Fcpu=Fhosc/2	
			max. (ms)	Unit (us)
0	000b	Fcpu/128	4.096	16
0	001b	Fcpu/64	2.048	8
0	010b	Fcpu/32	1.024	4
0	011b	Fcpu/16	0.512	2
0	100b	Fcpu/8	0.256	1
0	101b	Fcpu/4	0.128	0.5
0	110b	Fcpu/2	0.064	0.25
0	111b	Fcpu/1	0.032	0.125
1	useless	Fhosc	0.016	0.0625

8.4.3 TC1M模式寄存器

模式寄存器 TC1M 控制 TC1 的工作模式。

0DCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1M	TC1ENB	TC1rate2	TC1rate1	TC1rate0	-	TC1CKS0	-	PWM1OUT
读/写	R/W	R/W	R/W	R/W	-	R/W	-	R/W
复位后	0	0	0	0	-	0	-	0

Bit 0 **PWM1OUT**：PWM 输出控制位。
0 = 禁止 PWM 输出，P5.3 为普通 I/O 引脚；
1 = 允许 PWM 输出，P5.3 输出 PWM 信号。

Bit 2 **TC1CKS 0**：TC1 时钟源选择位。。
0 = Fcpu；
1 = Fhosc，**TC1Rate[2:0]处于无效状态**。

Bit [6:4] **TC1RATE[2:0]**：TC1 分频选择位。
000 = Fcpu/128；
001 = Fcpu/64；
010 = Fcpu/32
011 = Fcpu/16
100 = Fcpu/8
101 = Fcpu/4
110 = Fcpu/2；
111 = Fcpu/1。

Bit 7 **TC1ENB**：TC1 启动控制位。
0 = 关闭 TC1 定时器；
1 = 开启 TC1 定时器。

8.4.4 TC1C计数寄存器

8 位计数器 TC1C 溢出时，TC1IRQ 置 1 并由程序清零，用来控制 TC1 的中断间隔时间。首先须写入正确的值到 TC1C 和 TC1R 寄存器，并使能 TC1 定时器以保证第一个周期正确。TC1 溢出后，TC1R 的值自动装入 TC1C。

0DDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1C	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

TC1C 初始值的计算公式如下：

TC1C 初始值 = 256 -（TC1 中断间隔时间 * TC1 时钟 rate）

8.4.5 TC1R自动重装寄存器

TC1 内置自动重装功能，TC1R 寄存器存储重装数据。当 TC1C 溢出时，TC1R 的值自动装入 TC1C 中。TC1 定时器工作在计时模式时，要通过修改 TC1R 寄存器来修改 TC1 的间隔时间，而不是通过修改 TC1C 寄存器。在 TC1 定时器溢出后，新的 TC1C 值会被更新，TC1R 会将新的值装载到 TC1C 寄存器中。但在初次设置 TC1M 时，必须要在开启 TC1 定时器前把 TC1C 以及 TC1R 设置成相同的值。

TC1 为双重缓存器结构。若程序对 TC1R 进行了修改，那么修改后的 TC1R 值首先被暂存在 TC1R 的第一个缓存器中，TC1 溢出后，TC1R 的新值就会被存入 TC1R 缓存器中，从而避免 TC1 中断时间出错以及 PWM 误动作。

0DEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1R	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

TC1R 初始值计算公式如下：

TC1R 初始值 = 256 -（TC1 中断间隔时间 * TC1 时钟 rate）

- **例：计算 TC1C 和 TC1R 的值。TC1 的间隔时间为 10ms，时钟源来自 Fcpu= 16MHz/16=1MHz，TC1RATE=000（Fcpu/128）。**

TC1 中断间隔时间为 10ms，TC1 时钟 rate = 16MHz/16/128

TC1R = 256 -（TC1 中断间隔时间 * 输入时钟）
= 256 -（10ms * 16MHz / 16 / 128）
= 256 -（10-2 * 16 * 106 / 16 / 128）
= B2H

8.4.6 TC1D PWM占空比寄存器

TC1D 寄存器用来控制 PWM 的占空比。PWM 模式下，TC1R 控制 PWM 的周期，TC1D 控制 PWM 的占空比。TC1C=TC1D 时，PWM 切换为低电平。这样在应用中易于设置 TC1D 以选择合适的 PWM 占空比。

0EAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1D	TC1D7	TC1D6	TC1D5	TC1D4	TC1D3	TC1D2	TC1D1	TC1D0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

TC1D 初始值的计算方法如下：

TC1D 初始值 = TC1R +（PWM 脉冲高电平宽度周期 / TC1 时钟 rate）

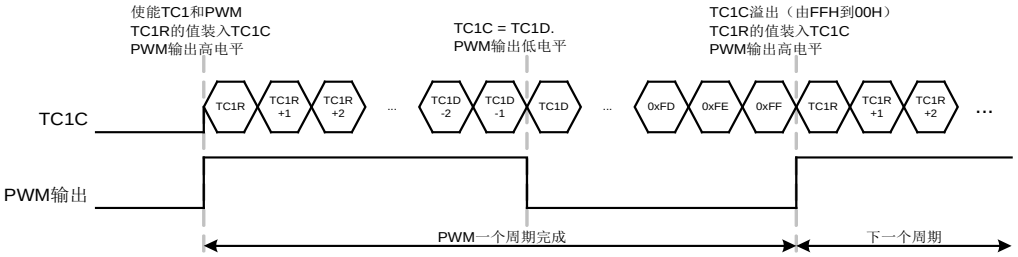
- **例：计算 TC1D 的值。1/3 占空比 PWM，TC1 时钟源 Fcpu=16MHz/16=1MHz，TC1RATE=000（Fcpu/128）。**

TC1R = B2H，TC1 间隔时间=10ms，PWM 周期频率为 100Hz，1/3 占空比条件下，PWM 高电平的宽度值约为 3.33ms。

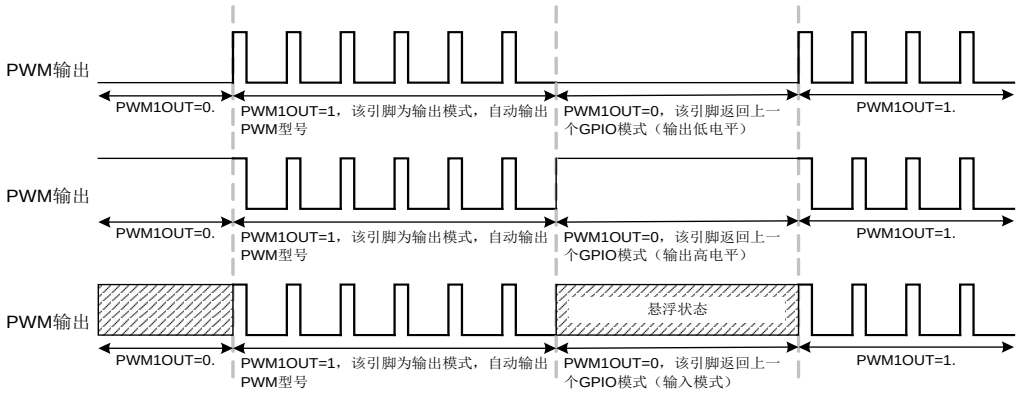
TC1D 初始值 = B2H +（PWM 脉冲高电平宽度值/TC0 时钟 Rate）
= B2H + (3.33ms * 16MHz / 16 / 128)
= B2H + 1AH
= CCH

8.4.7 脉冲宽度调制（PWM）

可编程控制占空比/周期的 PWM 可以提供不同的 PWM 信号。使能 TC1 定时器且 PWM1OUT=1 时，由 PWM 输出引脚（P5.3）输出 PWM 信号。PWM 首先输出高电平，然后输出低电平。TC1R 寄存器控制 PWM 的周期，TC1D 控制 PWM 的占空比（脉冲高电平的长度）。开启 TC1 定时器且定时器溢出后，TC1R 重装载 TC1C 的初始值。当 TC1C=TC1D 时，PWM 输出低电平；TC1 溢出时（TC1C 的值从 0FFH 到 00H），整个 PWM 周期完成，并进入下一个周期。TC1 溢出时，TC1R 的值自动装入 TC1C，PWM 的一个周期完成，以保持 PWM 的连贯性。在 PWM 输出的过程由程序更改 PWM 的占空比，则在下一个周期开始输出新的占空比的 PWM 信号。



PWM 的分辨率由 TC1R 决定，而 TC1R 的范围值为 00H~0FFH。当 TC1R=00H 时，PWM 的分辨率为 1/256；TC1R=80H 时，PWM 的分辨率为 1/128。TC1D 控制 PWM 高电平脉冲的宽度，即 PWM 的占空比。当 TC1C=TC1D 时，PWM 输出低电平，TC1D 的值必须大于 TC1R 的值，否则 PWM 的信号保持低电平状态。PWM 输出过程中，TC1 溢出时，TC1IRQ 有效，TC1IEN=1 时，则使能 TC1 中断。但强烈建议小心同时使用 PWM 和 TC1 定时器功能，保证两种功能都能正常工作。PWM 的输出引脚与 GPIO 共用，PWM1OUT=1 时，自动输出 PWM 信号；PWM1OUT=0，即禁止 PWM 时，该引脚自动返回到上一个 GPIO 模式。这样有利于处理 ON/OFF 操作的载波信号，而不控制 TC1ENB 位。



8.4.8 TC1 操作举例

- **TC1 定时器**

```
; 复位 TC1。
        CLR         TC1M            ; 清 TC1M。

; 设置 TC1 时钟源和 TC1Rate。
        MOV         A, #0nnn0n00b
        B0MOV       TC1M, A

; 设置 TC1C 和 TC1R 寄存器获得 TC1 间隔时间。
        MOV         A, #value       ; TC1C 必须和 TC1R 相等。
        B0MOV       TC1C, A
        B0MOV       TC1R, A

; 清 TC1IRQ。
        B0BCLR      FTC1IRQ

; 使能 TC1 定时器和中断功能。
        B0BSET      FTC1IEN         ; 使能 TC1 中断功能。
        B0BSET      FTC1ENB         ; 使能 TC1 定时器。
```

- **TC1 PWM**

```
; 复位 TC1。
        CLR         TC1M            ; 清 TC1M。

; 设置 TC1 时钟源和 TC1Rate。
        MOV         A, #0nnn0n00b
        B0MOV       TC1M, A

; 设置 TC1C 和 TC1R 寄存器获得 PWM 周期。
        MOV         A, #value1      ; TC1C 必须和 TC1R 相等。
        B0MOV       TC1C, A
        B0MOV       TC1R, A

; 设置 TC1D 寄存器获得 PWM 占空比。
        MOV         A, #value2      ; TC1D 的值必须大于 TC1R 的值。
        B0MOV       TC1D, A

; 使能 PWM 和 TC1 定时器。
        B0BSET      FTC1ENB         ; 使能 TC1 定时器。
        B0BSET      FPWM1OUT        ; 使能 PWM。
```

9 串行输入/输出收发器（SIO）

9.1 概述

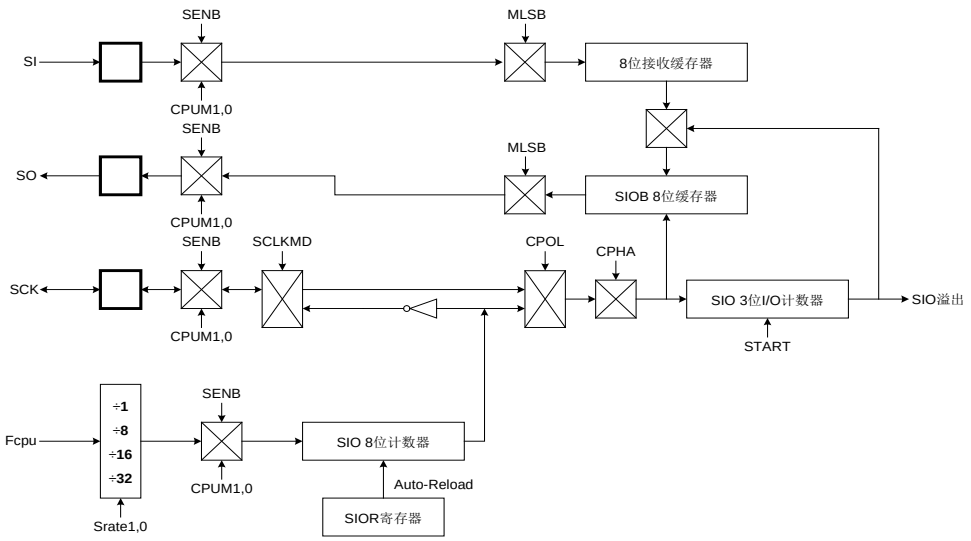
串行输入/输出收发器 SIO 允许数据在单片机与单片机，或单片机与其它外设之间进行传送。它为一个简单的 8 位接口，无规定的协议，封包及控制位。串行收发器模块有 3 个引脚，时钟引脚（SCK），数据输入引脚（SI），数据输出引脚（SO），这几个引脚使得数据可以在主机和从机之间传送。串行收发器具有 8 种数据发送格式，由 3 个位控制：时钟闲置状态，时钟相位和数据起始位方向。其支持大多数 SIO/SPI 通讯规范。

SIO 特性如下：

- **全双工 3 线同步传输；**
- **主控模式 (SCK 为时钟输出) 或从动模式 (SCK 为时钟输入) 操作；**
- **MSB/LSB 起始位传送；**
- **可以通过寄存器控制采样数据在第一个时钟相位有效或者第二个时钟相位有效；**
- **在多路从动装置应用时，SCK, SI, SO 均是可编程漏极开路输出引脚；**
- **主控模式时可设置数据传输速率；**
- **传送结束时产生 SIO 中断。**

9.2 SIO操作

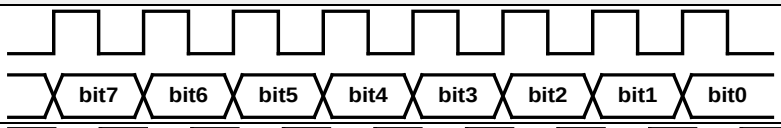
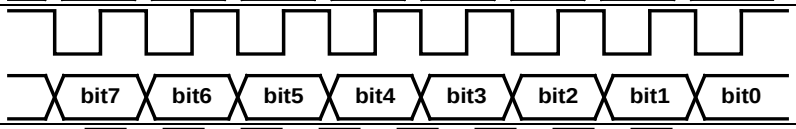
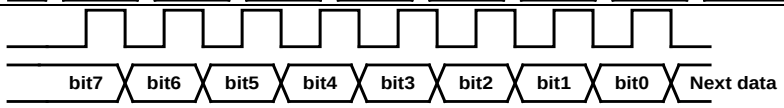
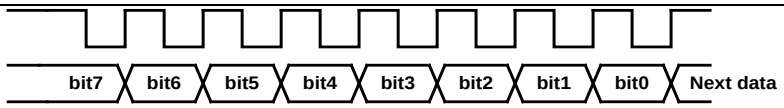
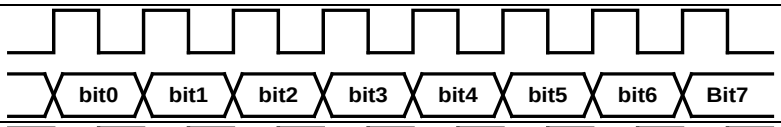
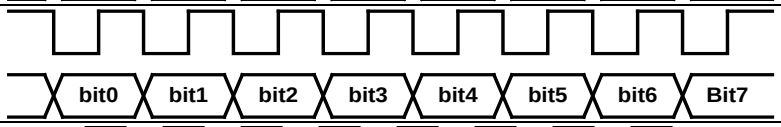
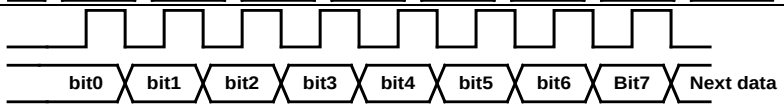
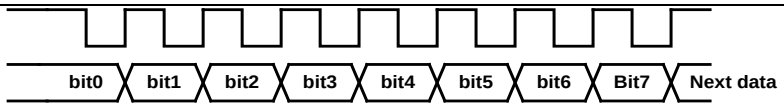
寄存器 SIOM 用来控制 SIO 功能，如发送/接收、时钟速率、触发边沿，时钟闲置状态，时钟相位，起始位优先发送权等。通过设置寄存器 SIOM 的 SENB 和 START 位，SIO 就可自动发送和接收 8 位数据。SIOB 是一个 8 位双层数据缓存器，用于存储发送/接收的数据，SIOC 和 SIOR 具有自动装载功能，能够产生 SIO 的时钟源。3 位的 I/O 计数器可以监控 SIO 的操作，每接收/发送 8 位数据后，会产生一个中断请求。一次发送或接收结束后，SIO 电路将自动禁止，可以通过重新编程 SIOM 寄存器启动下一次的数据传输。CPOL 位用来控制时钟闲置状态，CPHA 位用来控制数据接收的时钟沿方向，这两个位控制着串行收发器的数据的收发格式。SIO 数据传输的方向由 MLSB 位控制，可以先从低位发送，也可以先从高位开始发送。



SIO 接口电路图

串行收发器具有 8 种数据发送格式，有 3 个位控制：MLSB，CPOL 和 CPHA 。发送数据时由“数据传送边沿”控制。当设置为上升沿时，数据位在 SCK 上升沿发送和接收，在 SCK 下降沿发生数据转换；当设置为下降沿时，数据位在 SCK 下降沿发送和接收，在 SCK 上升沿发生数据转换。

CPHA 为时钟相位控制位，它控制有效数据采样的时钟相位。当 CPHA=1 时，在第一个 SCK 边沿转换数据，在第二个 SCK 边沿发送和接收数据；当 CPHA=0 时，第一个位被锁定，且在 SCK 第一个边沿发送和接收数据。SIO 数据传送时序图如下所示：

MLSB	CPOL	CPHA	示意图	说明
0	0	1		SCK 闲置状态 = Low 起始优先发送位= MSB SCK 数据传送边沿= 下降沿
0	1	1		SCK 闲置状态= High 起始优先发送位 = MSB SCK 数据传送边沿= 上升沿
0	0	0		SCK 闲置状态= Low 起始优先发送位 = MSB SCK 数据传送边沿= 上升沿
0	1	0		SCK 闲置状态= High 起始优先发送位 = MSB SCK 数据传送边沿= 下降沿
1	0	1		SCK 闲置状态= Low 起始优先发送位 = LSB SCK 数据传送边沿= 下降沿
1	1	1		SCK 闲置状态= High 起始优先发送位 = LSB SCK 数据传送边沿= 上升沿
1	0	0		SCK 闲置状态= Low 起始优先发送位 = LSB SCK 数据传送边沿= 上升沿
1	1	0		SCK 闲置状态= High 起始优先发送位 = LSB SCK 数据传送边沿= 下降沿

SIO 数据传输时序图

SIO 支持中断功能。SIOIEN 为 SIO 中断控制位。SIOIEN=0 时，禁止 SIO 中断；SIOIEN=1 时，使能 SIO 中断。SIO 产生中断时，程序计数器跳到中断向量（ORG 8）处，执行 SIO 中断服务程序。SIOIRQ 为 SIO 中断请求标志位，在 SIOIEN = 0 时也可以指示 SIO 的执行状态，但是必须用程序清零。SIO 操作完成后，SIOIRQ 置 1，是与 SIO “START” 控制位相反的状态位。

SIOIRQ 和 SIO 起始位指示 8 位数据传送后 SIO 的状态，由于从 SIO 传送结束到 SIOIRQ/START 有效的时间大约为“1/2*SIO 时钟周期”，这也就意味着，SIO 传送结束的指示标志位不能立即显示出来。

✷ **注：SIO 操作的第一步是先设置 SIO 各引脚的模式，使能 SENB 位，选择 CPOL 和 CPHA 位，这些位将控制 SIO 各引脚的模式。**

9.3 SIOM模式寄存器

0B4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOM	SENB	START	SRATE1	SRATE0	MLSB	SCKMD	CPOL	CPHA
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **SENB**：SIO 功能控制位。
0 = 禁止（P5.0~P5.2 作为普通输入输出引脚）；
1 = 使能（P5.0~P5.2 作为 SIO 引脚，SIO 引脚可以为推拉式结构和由 P1OC 寄存器控制的漏极开漏结构）。

Bit 6 **START**：SIO 状态控制位。
0 = 传送结束；
1 = 传送过程中。

Bit [5:4] **SRATE1,0**：SIO 传送时钟分频位，当 SCKMD=1，这两个位的功能是无效的。
00 = Fcpu；
01 = Fcpu/32；
10 = Fcpu/16；
11 = Fcpu/8。

Bit 3 **MLSB:** MSB/LSB 优先传送控制位。
0 = MSB 优先发送；
1 = LSB 优先发送。

Bit 2 **SCKMD**：SIO 时钟模式选择位。
0 = 内部时钟（主控模式）；
1 = 外部时钟（从动模式）。

Bit 1 **CPOL:** 时钟（SCK）闲置控制位。
0 = SCK 闲置输出低；
1 = SCK 闲置输出高。

Bit 0 **CPHA:** 数据采样有效的时钟相位选择控制位。
0 =在第一个时钟相位采样数据有效；
1 =在第二个时钟相位采样数据有效。

SIO 功能是与 P5 口共用：P5.0/SCK、P5.1/SI、P5.2/SO。下表列出了在使能或禁止 SIO 功能时，P5[2:0]的 I/O 口的模式状态和设置。

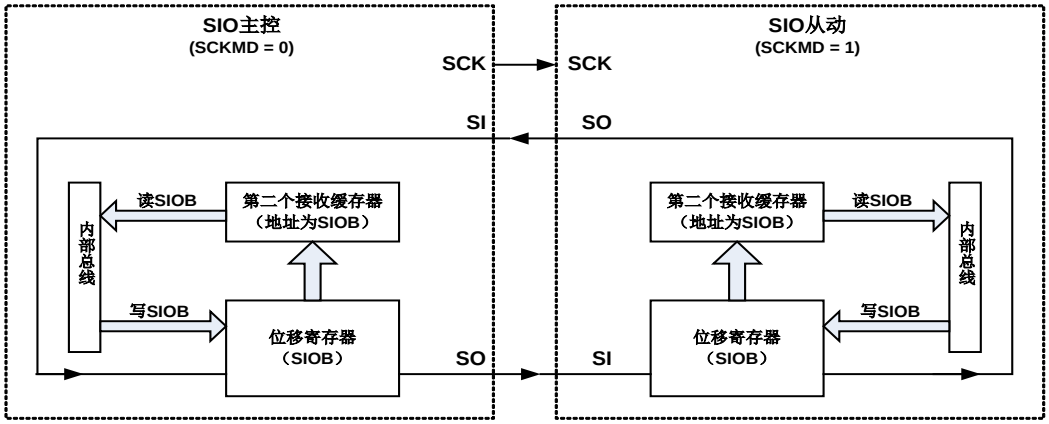
SENB=1 (SIO 使能 SIO 功能**)**		
P5.0/SCK	(SCKMD=1) 时钟来自外部时钟	P5.0 被自动设为输入模式，不管 P5M 如何设置
	(SCKMD=0) SIO 时钟源来自内部时钟	P5.0 被自动设为输出模式，不管 P5M 如何设置
P5.1/SI	P5.1 必须设为输入模式，否则 SIO 功能会出错	
P5.2/SO	SIO =发送/接收	P5.2 被自动设为输出模式，不管 P5M 如何设置
SENB=0 (禁止 SIO 功能)		
P5.0/P5.1/P5.2	禁止 SIO 功能时，由 P5M 完全控制 P5[2:0]的 I/O 模式	

✷ **注 1：在外部时钟模式时，若 SCKMD=1，则 SIO 处于从动模式；内部时钟时，若 SCKMD = 0 则为主控模式。**
✷ **注 2：不能同时设置 SENB 和 START 位为 1，否则会导致 SIO 传送出错。**
✷ **注 3：SIO 引脚可为推拉式结构，也可以为漏极开漏结构，由 P10C 寄存器控制。**

9.4 SIOB数据缓存器

0B6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOB	SIOB7	SIOB6	SIOB5	SIOB4	SIOB3	SIOB2	SIOB1	SIOB0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

SIOB为SIO的数据缓存寄存器，它存储发送/接收的数据。系统在传送时为一次缓存，而在接收时为两级缓存。也就是说在整个移位周期结束前，新的数据不能写入SIOB数据寄存器中；而在接收数据时，在新的数据完全移入前，必须从SIOB数据寄存器中读出接收的数据，否则，前一个数据将会丢失。下图是一个典型的单片机之间的数据通信。由主控单片机发送SCK启动数据传输，两个单片机必须有相同的时钟沿触发方式，并将在同一时刻发送和接收数据。



SIO 数据传送示意图

9.5 SIOR寄存器说明

0B5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SIOR	SIOR7	SIOR6	SIOR5	SIOR4	SIOR3	SIOR2	SIOR1	SIOR0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

寄存器SIOR用于SIO的自动装载，类似于后置分频器，能够提高SIO的时钟精度。用户可对SIOR进行写操作，从而控制SIO的通信速率。SIO的时钟频率计算如下：

SCK 频率 **=** （**SIO** 速率 **/**（**256 - SIOR**））**/2**

SIOR = 256 - （1 / 2*（ SCK 频率）* SIO 速率）

- **例：设置 SIO 时钟频率为 5KHz，Fhosc = 16MHz，SIO's 速率 = Fcpu = Fhosc/16。**

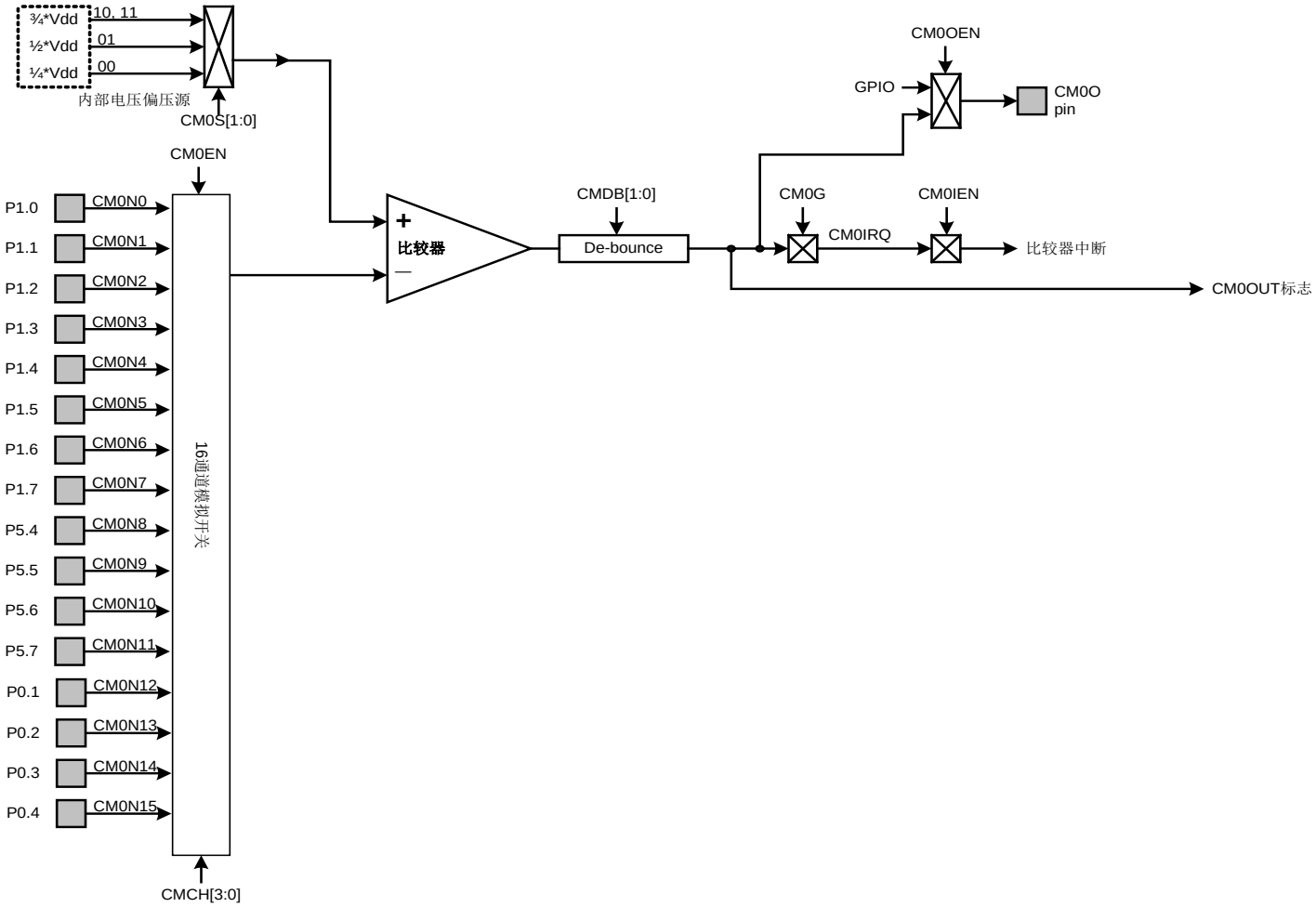
SIOR = 256 –（1/（2*5KHz）* 16MHz/16）
= 256 –（0.0001*1000000）
= 256 – 100
= 156

10 16 通道模拟比较器

10.1 概述

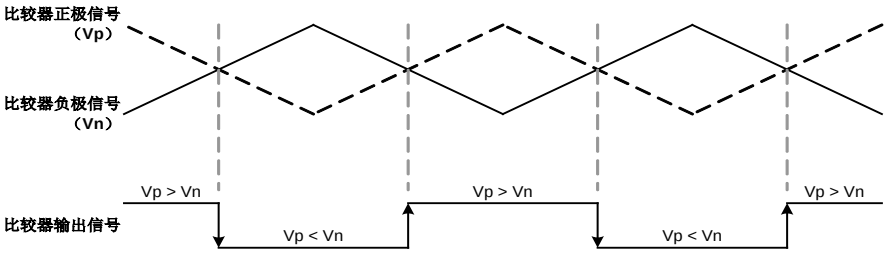
模拟比较器用来比较负极输入电压和正极输入电压，然后在输出端输出比较结果。对应不同的应用，比较器有不同的输入选择。比较器的负极输入端共有 16 个输入通道，由 CMCH[3:0]控制，正极输入端共有 3 个选项，由 CM0S[1:0]控制。比较器的输出端可连接到外部引脚 CM0O，并与内部通道相连，比较器的触发方向可编程控制。针对不同的应用，比较器提供比较结果标志位指示、中断功能和绿色模式下唤醒功能。

- **16 通道负极输入通道；**
- **比较器输出功能；**
- **可编程控制的内部参考电压与比较器的正极相连接；**
- **可编程控制的触发方向；**
- **中断功能；**
- **绿色模式下唤醒功能。**



10.2 比较器操作

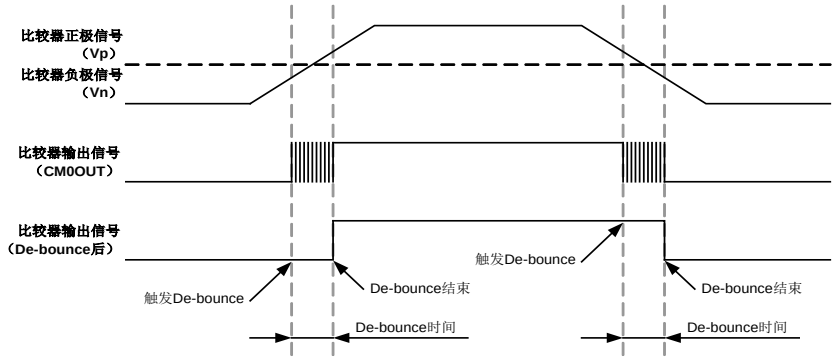
比较器用来比较比较器的正极输入电压和负极输入电压，当正极输入电压大于负极输入电压时，比较器输出高电平；当正极输入电压小于负极输入电压时，比较器输出低电平。



比较器内置中断功能，中断触发方向由 CM0G 控制，CM0G=0 时，上升沿触发；CM0G=1 时，下降沿触发。中断触发时，CM0IRQ 置 1，使能比较器中断功能，系统执行中断，由程序清 CM0IRQ。

比较器内置绿色模式唤醒功能，由电平变换触发唤醒。比较器的唤醒功能只能将系统从绿色模式下唤醒，而不能将系统从睡眠模式下唤醒。比较器唤醒触发时，系统从绿色模式下被唤醒。当唤醒系统的触发方向为中断触发方向时，CM0IRQ 会被置 1，如果使能了比较器中断功能，系统从绿色模式下被唤醒之后会立即执行比较器中断。

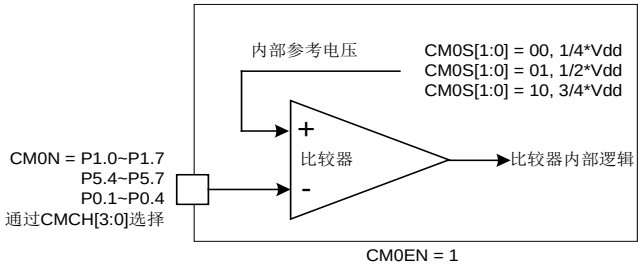
比较器工作中的一个至关重要的条件是比较器的负极输入电压与正极输入电压相等，而判断为正负极电压相等时的负极输入电压范围由比较器的共模输入补偿参数来决定，在该范围内，比较器的输出信号不稳定，并一直保持振荡，直到比较器的正负极电压不相等时为止。在这种情况下，比较器标志 CM0IRQ 锁存第一个变换状态和初始状态，而该状态为瞬时状态。故比较器内建了一个滤波器来对该不稳定状态进行滤波。比较器的输出信号通过 de-bounce 电路滤除比较器的瞬时状态。de-bounce 的时间由 CMDB[1:0]控制，即比较器的最小响应时间为、1*Fcpu、2*Fcpu、3*Fcpu 或者 no de-boune。De-bounce 的时间由信号的电压转换速率以及程序的设置共同决定。



比较器的正极输入端包括内部参考电压源，内部参考电压共有三种电压：CM0S[1:0]=00 时为 1/4 VDD，CM0S[1:0]=01 时为 1/2 VDD，CM0S[1:0]=10、11 时为 3/4 VDD。

比较器的负极输入端共有 16 个输入通道，由 CMCH[3:0]控制，0000=P1.0，0001=P1.1，0010=P1.2，0011=P1.3，0100=P1.4，0101=P1.5，0110=P1.6，0111=P1.7，1000=P5.4，1001=P5.5，1010=P5.6，1011=P5.7、1100=P0.1、1101=P0.2、1110=P0.3、1111=P0.4。只能在使能比较器（CM0EN=1）后，这 16 个通道才可以选择是否工作。当选择其中一个引脚作为输入通道时，该引脚切换为输入模式，并连接到比较器的负极输入端。当系统选择其它的引脚或者禁止比较器后，该引脚自动返回到上一个 GPIO 模式。

比较器的输出状态可由 CM0OEN 位来控制是否从 CM0O 引脚输出。CM0OEN=0 时，比较器输出引脚为 GPIO 模式；CM0OEN=1 时，CM0O 引脚输出比较器的输出结果，并禁止 GPIO 模式。比较器的输出端连接到内部通道，CM0OUT 标志能立即显示出比较器的比较结果，而 CM0IRQ 只是表征了比较器有比较结果输出。比较器的输出端通过 de-bounce 电路后产生一个触发沿，由 CM0G 控制，包括上升沿（CM0OUT 由低到高）和下降沿（CM0OUT 由高到低）。CM0IRQ=1，且 CM0IEN=1（使能比较器中断）时，执行比较器中断。



10.3 比较器控制寄存器

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM0M	CM0EN	CM0OUT	CM0S1	CM0S0	CMCH3	CMCH2	CMCH1	CMCH0
读/写	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	1	0	0	0	0	0	0

Bit 7 **CM0EN**：比较器控制位。
0 = 禁止，P1[7:0]和 P5[7:4]和 P0[4:1]为 GPIO 引脚；
1 = 使能，比较器的负极输入引脚由 CMCH[3:0]控制。

Bit 6 **CM0OUT**：比较器输出标志位。**禁止比较器时，比较器输出状态为 1。**
0 = 比较器内部参考电压小于 CM0N 电压；
1 = 比较器内部参考电压大于 CM0N 电压。

Bit [5:4] **CM0S[1:0]**：比较器正极输入端电压控制位。
00 = 内部 1/4*Vdd，使能内部参考电压发生器；
01 = 内部 1/2*Vdd，使能内部参考电压发生器；
10,11 = 内部 3/4*Vdd，使能内部参考电压发生器

Bit [3:0] **CMCH[3:0]**：比较器负极输入引脚控制位。
0000 = 比较器负极输入引脚 P1.0；
0001 = 比较器负极输入引脚 P1.1；
0010 = 比较器负极输入引脚 P1.2；
0011 = 比较器负极输入引脚 P1.3；
0100 = 比较器负极输入引脚 P1.4；
0101 = 比较器负极输入引脚 P1.5；
0110 = 比较器负极输入引脚 P1.6；
0111 = 比较器负极输入引脚 P1.7；
1000 = 比较器负极输入引脚 P5.4；
1001 = 比较器负极输入引脚 P5.5；
1010 = 比较器负极输入引脚 P5.6；
1011 = 比较器负极输入引脚 P5.7；
1100 = 比较器负极输入引脚 P0.1；
1101 = 比较器负极输入引脚 P0.2；
1110 = 比较器负极输入引脚 P0.3；
1111 = 比较器负极输入引脚 P0.4。

09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM0M1	-	-	-	-	CMDB1	CMDB0	CM0OEN	CM0G
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	0	0	0	0

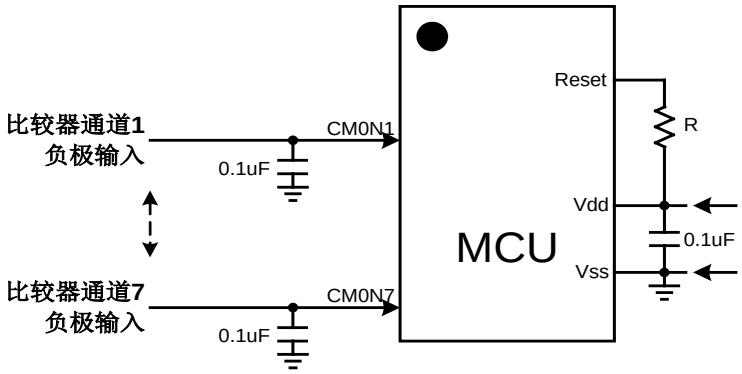
Bit [3:2] **CMDB[1:0]**：比较器输出 debounce 时间选择位。
00 = 1*Fcpu；
01 = 2*Fcpu；
10 = 3*Fcpu；
11 = No de-bounce。

Bit 1 **CM0OEN**：比较器输出引脚控制位。
0 = 禁止，CM0O 为 GPIO 模式；
1 = 使能比较器输出，P0.0 引脚切换为比较器输出引脚 CM0O，并禁止 GPIO 功能。

Bit 0 **CM0G**：比较器中断触发方向控制位。
0 = 上升沿触发，比较器内部参考电压> CM0N；
1 = 下降沿触发，比较器内部参考电压< CM0N。

10.4 比较器应用注意事项

比较器比较正极电压和负极电压并输出比较结果，正极输入端的内部参考电压和负极输入电压都是模拟信号。在硬件应用电路中，比较器输入引脚都必须连接一个 0.1uF 的电容，以减少电源干扰，使输入信号更稳定。应用电路如下所示：



➢ **例：使用比较器测量外部模拟信号，当模拟信号的电压小于 1/2 VDD 时执行中断服务程序。**
在该示例程序中，采用内部 1/2 VDD 参考电压作为比较器的正极输入电压，上升沿触发比较器中断。

```
; 比较器初始化。
        MOV     A, #00010000b   ; CM0S[1:0]=01b，选择比较器内部 1/2 VDD 参考电压为
        B0MOV   CM0M, A         ; 比较器的正极输入电压。
                                ; CMCH[3:0]=0000b，选择 CM0N0 为比较器的负极输入引脚。
        MOV     A, #00000000b   ; CM0G=0b，设置比较器的中断触发方向为上升沿触发。
        B0MOV   CM0M1, A
                                ; CMDB[1:0]=00b，没有进行 de-bounce。
        B0BSET  FCM0IEN         ; CM0IEN=1，使能比较器中断功能。
        B0BCLR  FCM0IRQ         ; CM0IRQ=0，清比较器中断请求标志位。

        B0BSET  FCM0EN          ; 使能比较器

Main:                           ;
        …
        JMP     MAIN
        …

; 中断服务程序。
ISR:
        PUSH                    ; 保存 ACC 和 PFLAG。
        B0BTS1  FCM0IRQ         ; 检查是否有比较器中断请求。
        JMP     ISR_EXIT        ; 没有中断请求，退出中断。

        B0BCLR  FCM0IRQ         ; 清比较器中断请求标志位。
        …                       ; 执行比较器中断。
        …
        …
        JMP     ISR_EXIT        ; 中断结束。

ISR_EXIT:                       ; 退出中断程序。
        POP                     ; 恢复 ACC 和 PFLAG。
        RETI                    ; 退出中断。
```

11 MSP

11.1 概述

MSP（Main Serial Port）是一个串行通信接口，用来进行几个单片机之间或者外围硬件之间的数据转换。外围设备可以是 EEPROM，ADC、显示设备等。

MSP 模块可以在下面模式下进行操作：

- **从动模式（带通用地址调用）。**

MSP 的主要性能有：

- **2 线同步数据发送/接收；**
- **从动（SCL 为时钟输入）操作；**
- **SCL、SDA 为可编程漏极开路输出引脚，可适合多种从动设备的应用；**
- **支持 400K 的时钟频率@Fcpu=4MIPs；**
- **发送/接收结束时产生 MSP 中断。**

11.2 MSP状态寄存器

090H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPSTAT	-	CKE	D_A	P	S	RED_WRT	-	BF
读/写	-	R/W	R	R	R	R	-	R
复位后	-	0	0	0	0	0	-	0

Bit 6 **CKE：**从机时钟边沿控制位。
从动模式下接收地址或数据字节。
0 = SCL 上升沿锁存数据（默认）；
1 = SCL 下降沿锁存数据。

✷ **注 1：从动模式下发送数据时，接收地址取决于 CKE 的设置；在 SCL 下降沿发送数据。**
✷ **注 2：从动模式下接收数据时，接收的地址和数据都由 CKE 设置。**

Bit 5 **D_A：**数据/地址标志位。
0 = 表示接收或者发送的上一字节是地址；
1 = 表示接收或者发送的上一字节是数据。

Bit 4 **P：**停止标志位。
0 = 没有检测到停止位；
1 = 检测到停止位。

✷ **注 1：检测到起始位后将停止位清零。**

Bit 3 **S：**起始标志位。
0 = 没有检测到起始位；
1 = 检测到起始位。

✷ **注 1：检测到停止位将起始位清零。**

Bit 2 **RED_WRT：**读/写标志位。该标志位在地址匹配成功之后开始有效，并锁存地址信息中的读写方向位，当接收到下一个起始位、停止位或 NO ACK 时，该标志位自动失效。
0 = 写；
1 = 读。

Bit 0 **BF**：缓存器状态位。
接收：1 = 接收完成，MSPBUF 填满；
0 = 接收没有完成，MSPBUF 空置。
发送：1 = 正在发送数据（不包括 ACK 和停止位），MSPBUF 填满；
0 = 完成发送数据（不包括 ACK 和停止位），MSPBUF 空置。

11.3 MSP模式寄存器

091H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPM1	WCOL	MSPOV	MSPENB	CKP	SLRXCKP	MSPWK	GCEN	-
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
复位后	0	0	0	0	0	0	0	-

Bit 7 **WCOL**：写冲突检测标志位
从动模式：
0 =未检测到写冲突；
1 = 正在发送上一个数据时，有写入新的数据到 MSPBUF。（WCOL 标志位必须由软件清零）

Bit 6 **MSPOV**：接收溢出标志位。
0 = 没有溢出；
1 =当 MSPBUF 仍存储着前一个接收到的字节时，接收到了新的字节时，则发生了接收溢出，此时 MSPOV 被置 1。在从机发送模式下 MSPOV 标志位可忽略。

Bit 5 **MSPENB**：MSP 通信使能位。
0 = 禁止 MSP，相关引脚为普通 I/O 引脚；
1 = 使能 MSP，相关引脚为 SCL、SDA 串口引脚。

注：禁止 MSP 后，MSP 状态寄存器会被清 0。故用户在使能 MSP 之前需重新设置 MSP 寄存器。

```
Ex:B0bclr    FMSPENB
     Call       MSP_init_setting
     B0bset     FMSPENB
```

Bit 4 **CKP**：SCL 时钟优先控制位。
MSP 从动模式：
0 = SCL 保持低电平（保证数据设置时间和从动设备准备就绪）；
1 = 释放 SCL 时钟（从动发送模式时始终使能 CKP 功能，从动接收模式时由 SLRXCKP 控制 CKP 功能）。

Bit 3 **SLRXCKP**：从动接收模式时 SCL 时钟优先控制位。
MSP 从动接收模式：
0 = 禁止 CKP 功能；
1 = 使能 CKP 功能。
MSP 从动发送模式下无效。

Bit 2 **MSPWK**：MSP 唤醒显示位。
0 = MSP 没有唤醒单片机；
1 = MSP 唤醒单片机。

✷ **注：在进入睡眠模式之前将 MSPWK 清零，以便在唤醒后知道是否由 MSP 唤醒。**

Bit 1 **GCEN**：通用调用使能位（仅在从动模式下有效）。
0 = 禁止通用地址调用；
1 = 接收到通用调用地址后使能中断。

11.4 MSP MSPBUF寄存器

MSPBUF 初始值 = 0000 0000

092H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPBUF	MSPBUF7	MSPBUF6	MSPBUF5	MSPBUF4	MSPBUF3	MSPBUF2	MSPBUF1	MSPBUF0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

11.5 MSP MSPADR寄存器

MSPADR 初始值 = 0000 0000

093H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSPADR	MSPADR7	MSPADR6	MSPADR5	MSPADR4	MSPADR3	MSPADR2	MSPADR1	MSPADR0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

11.6 从机模式操作

当接收到的地址匹配成功，或者寻址成功之后接收到数据，硬件将自动产生 ACK_信号，将接收到的数据从 MSPSR 装载到 MSPBUF（MSP 缓存寄存器）中。

在下列情形下，MSP 功能不会应答 ACK_信号：

- **数据缓存器全满：BF = 1 (MSPSTAT bit 0)时，接收到了另一组数据。**
- **数据溢出：MSPOV = 1 (MSPM1 bit 6)时，接收到了另一组数据。**

BF=1，即 MCU 还未读取 MSPBUF 的数据，故 MSPSR 中的数据还未装入 MSPBUF，但此时 MSPIRQ 和 MSPOV 位仍置 1，当读取 MSPBUF 寄存器后，BF 自动被清零，MSPOV 必须由软件清零。

11.6.1 寻址

使能 MSP 从动功能时，需等待产生一个 START 信号，START 信号产生后，8 位地址装入 MSPSR 寄存器，MSPSR[7:1] 的数据与 MSPADR 寄存器在 SCL 脉冲的第 8 个下降沿进行比较，若地址相同，BF 和 MSPOV 都清零，产生下列事件：

- **在 SCL 的第 8 个下降沿，MSPSR 寄存器的数据装入 MSPBUF。**
- **在 SCL 的第 8 个下降沿，缓存器状态位 BF 置 1。**
- **产生一个 ACK_信号。**
- **在 SCL 的第 9 个下降沿，MSP 中断请求 MSPIRQ 置 1。**

接收数据时的状态		MSPSR→ MSPBUF	响应 ACK_信号	设置 MSPIRQ
BF	MSPOV			
0	0	Yes	Yes	Yes
*0	*1	Yes	No	Yes
1	0	No	No	Yes
1	1	No	No	Yes

接收数据效果表

✷ **注：BF=0，MSPOV=1 表示软件没有清除数据溢出标志位。**

11.6.2 从机接收

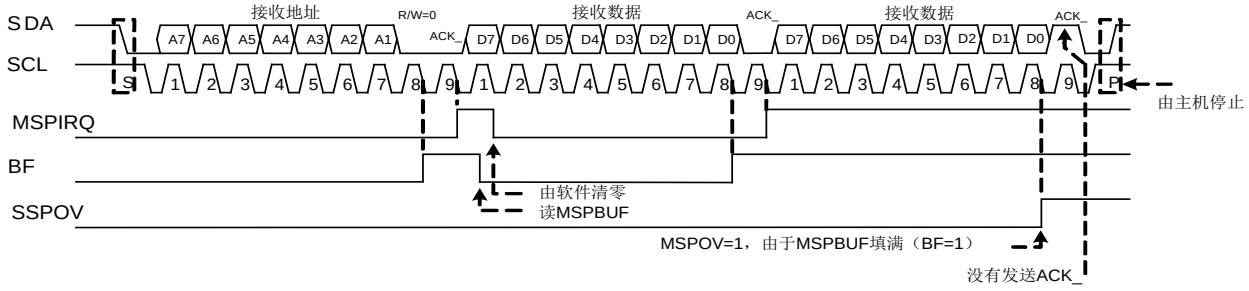
当硬件地址匹配，且地址中 R/W 位=1，那么 MSPSTAT 中的 RED_WRT 标志位被清零。同时地址载入 MSPBUF，在给出应答信号后，MSP 将每 8 个 CLK 接收一次数据，由 SLRXCKP 位控制是否使能 CKP 功能，由 CPE 位控制数据的锁存边沿－上升沿或下降沿。

当溢出时，不管 BF=1 还是 MSPOV=1，都没有应答信号。

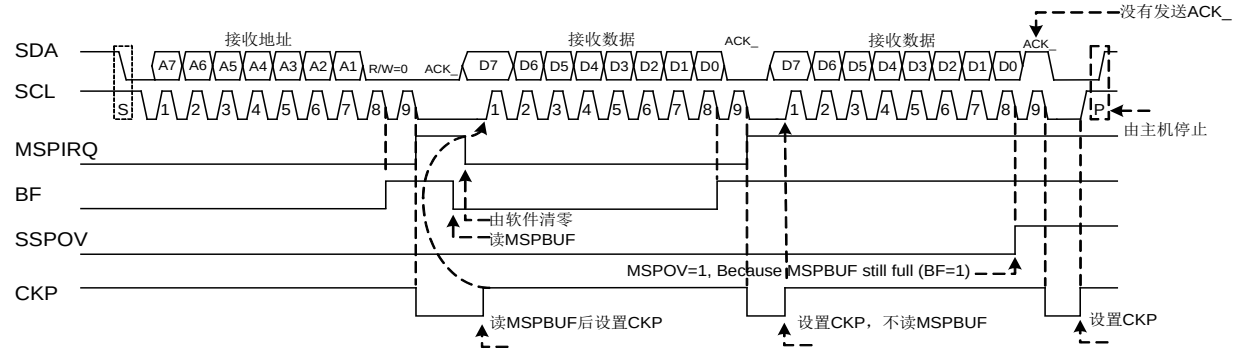
每发送一次数据会产生一次 MSP 中断，必须由软件清 MSPIRQ。

从动模式接收示意图如下所示：

SLRXCKP=0



SLRXCKP=1

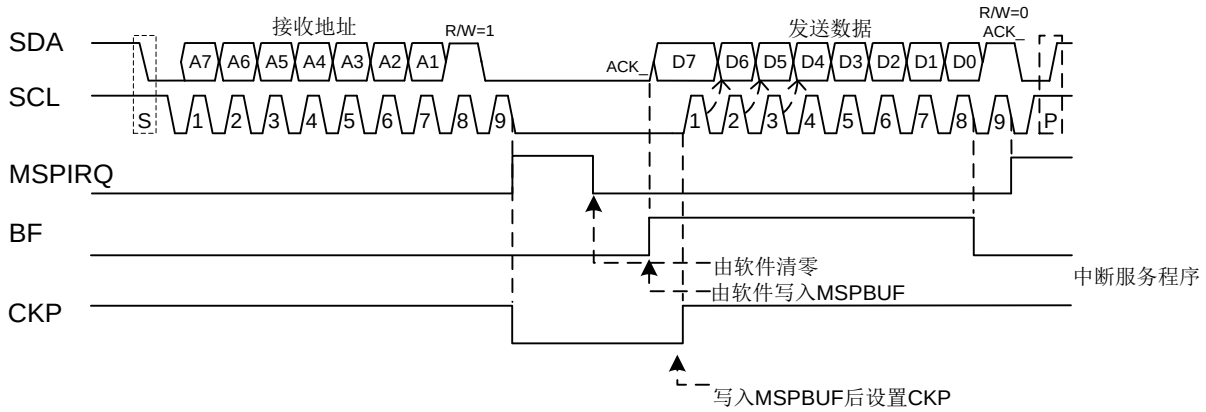


11.6.3 从机发送

匹配地址后，地址中 R/W 位=1，MSPSTAT bit 2 RED_WRT 置 1，接收到的地址装入 MSPBUF，然后在第 9 个时钟发送一个 ACK 信号，并将 SCL 保持为低电平。软件将发送的数据写入 MSPBUF 后，硬件会自动将数据写入 MSPSR 寄存器。主机应当要监控 SCL 引脚上的信号，因从机可能会通过设置 CKP 为 0 将 SCL 保持为低电平。在将数据载入 MSPBUF 后，设置 CKP 为 1，MSPBUF 数据将会再 SCL 信号的下降沿发送出去。这样可确保发送到 SDA 引脚的信号在 SCL 高电平周期内是有效的。

每个字节发送完后产生 MSP 中断，中断请求 MSPIRQ 在第 9 个 SCL 时钟置 1，由软件清零。MSPSTAT 寄存器可以监控数据发送的状态。

从动发送模式下，从主机接收的 ACK_信号在第 9 个 SCL 时钟的上升沿被锁存，如 ACK_为高电平，发送完成，从动设备逻辑复位并等待另一个 START 信号。如 ACK_为低电平，从设备应该重新导入 MSPBUF，设置 CKP=1，重新开始发送数据。



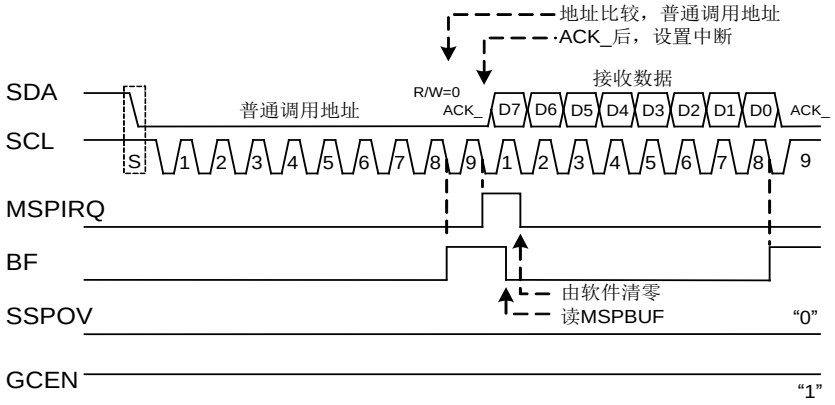
MSP 从动模式发送时序图

11.6.4 通用地址调用

MSP 总线的第一个字节的前 7 位为从动地址，只有该地址和 MSPADR 的数据相符时才会响应一个 ACK_信号。现有一个通用地址可以作为所有从动设备的硬件地址，发生该地址时，所有的从动设备都响应一个应答信号。

通用地址是一个全为 0 的特殊地址。通用地址功能由 GCEN 位控制。该位置 1 则使能通用地址功能，反之则禁止。当 GCEN=1，START 信号后，8 位地址装入 MSPSR，并与 MSPADR 进行比较，且通用地址由硬件固定。

如通用地址匹配，MSPSR 数据发送至 MSPBUF，BF 置 1，在第 9 个时钟（ACK_）下降沿请求 MSP 中断，执行中断时，读取 MSPBUF，检测该地址是否为通用地址或者特殊地址。

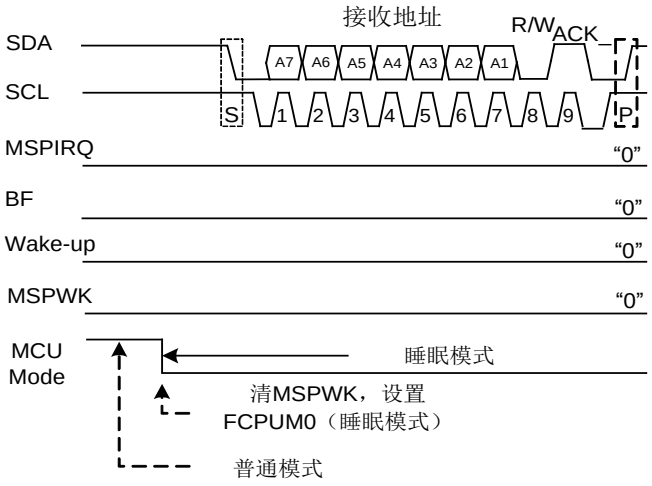


通用地址调用时序图

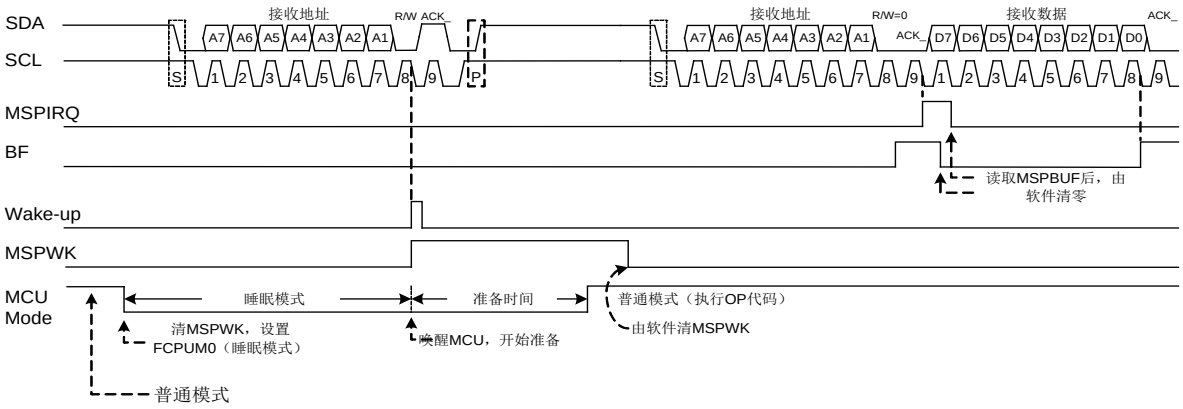
11.6.5 从机模式唤醒

睡眠模式下，若 MSPENB 置 1，则匹配的设备地址可以将单片机唤醒。

8 位 MSP 总线地址跟随 START 之后，装入 MSPSR，若地址匹配，在第 9 个 SCL 时钟时响应一个 NO ACK 信号，单片机被唤醒，MSPWK 置位，开始唤醒处理，但 MSPIRQ 不会置 1，且 MSPSR 数据也不会装入 MSPBUF。单片机被唤醒后，MSP 处于空闲状态，并等待主机的 START 信号。MCU 被唤醒之后，控制寄存器 BF、MSPIRQ、MSPOV 和 MSPBUF 的状态/数据与进入睡眠模式之前是一样的。若地址不匹配，在第 9 个 SCL 时钟时发送一个 NO ACK 信号，但不会唤醒单片机，系统仍处于睡眠状态。



MSP 唤醒时序示意图：地址不匹配



MSP 唤醒时序示意图：地址匹配

睡眠模式下，需要禁止 MSP，然后再使能 MSP 复位 MSP 功能，重新写入 I2C 的从动地址。

➢ **例：**

```
B0BSET      FCPUM0
B0BCLR      FMSPENB
NOP
B0BSET      FMSPENB
MOV         A, #0Xnn
B0MOV       MSPADR, A       ; 重新写入 I2C 的从动地址。
```

- **注 1：MSP 功能只能在普通模式下工作，当系统从睡眠模式被唤醒后，单片机必须在主机发送 START 信号之前进入普通模式下进行工作。**
- **注 2：MSP 唤醒时，若地址不匹配，则单片机仍处于睡眠模式。**
- **注 3：在进入睡眠模式之前，先由软件清 MSPWK，以显示唤醒状态。**

12 指令集

Field	Mnemonic		指令说明	C	DC	Z	Cycle
MOVE	MOV	A,M	A ← M	-	-	√	1
	MOV	M,A	M ← A	-	-	-	1
	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I，“M”指 80H~87H 的寄存器（如 PFLAG、R、Y、Z …）。	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
ARITHMETIC	ADC	A,M	A ← A + M + C，若有进位，则 C=1，否则 C=0。	√	√	√	1
	ADC	M,A	M ← A + M + C，若有进位，则 C=1，否则 C=0。	√	√	√	1+N
	ADD	A,M	A ← A + M，若有进位，则 C=1，否则 C=0。	√	√	√	1
	ADD	M,A	M ← A + M，若有进位，则 C=1，否则 C=0。	√	√	√	1+N
	B0ADD	M,A	M (bank 0) ← M (bank 0) + A，若有进位，则 C=1，否则 C=0。	√	√	√	1+N
	ADD	A,I	A ← A + I，若有进位，则 C=1，否则 C=0。	√	√	√	1
	SBC	A,M	A ← A - M - /C，若有借位，则 C=0，否则 C=1。	√	√	√	1
	SBC	M,A	M ← A - M - /C，若有借位，则 C=0，否则 C=1。	√	√	√	1+N
	SUB	A,M	A ← A - M，若有借位，则 C=0，否则 C=1。	√	√	√	1
	SUB	M,A	M ← A - M，若有借位，则 C=0，否则 C=1。	√	√	√	1+N
	SUB	A,I	A ← A - I，若有借位，则 C=0，否则 C=1。	√	√	√	1
LOGIC	AND	A,M	A ← A 与 M。	-	-	√	1
	AND	M,A	M ← A 与 M。	-	-	√	1+N
	AND	A,I	A ← A 与 I。	-	-	√	1
	OR	A,M	A ← A 或 M。	-	-	√	1
	OR	M,A	M ← A 或 M。	-	-	√	1+N
	OR	A,I	A ← A 或 I。	-	-	√	1
	XOR	A,M	A ← A 异或 M。	-	-	√	1
	XOR	M,A	M ← A 异或 M。	-	-	√	1+N
	XOR	A,I	A ← A 异或 I。	-	-	√	1
PROCESS	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
	RRC	M	A ← RRC M	√	-	-	1
	RRCM	M	M ← RRC M	√	-	-	1+N
	RLC	M	A ← RLC M	√	-	-	1
	RLCM	M	M ← RLC M	√	-	-	1+N
	CLR	M	M ← 0	-	-	-	1
	BCLR	M.b	M.b ← 0	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1+N
BRANCH	CMPRS	A,I	ZF,C ← A – I，如果 A = I，跳转到下一条指令。	√	-	√	1 + S
	CMPRS	A,M	ZF,C ← A – M，如果 A = M，跳转到下一条指令。	√	-	√	1 + S
	INCS	M	A ← M + 1，如果 A = 0，跳转到下一条指令。	-	-	-	1+ S
	INCMS	M	M ← M + 1，如果 M = 0，跳转到下一条指令。	-	-	-	1+N+S
	DECS	M	A ← M – 1，如果 A = 0，跳转到下一条指令。	-	-	-	1+ S
	DECMS	M	M ← M – 1，如果 M = 0，跳转到下一条指令。	-	-	-	1+N+S
	BTS0	M.b	如果 M.b = 0，跳转到下一条指令。	-	-	-	1 + S
	BTS1	M.b	如果 M.b = 1，跳转到下一条指令。	-	-	-	1 + S
	B0BTS0	M.b	如果 M(bank 0).b = 0，跳转到下一条指令。	-	-	-	1 + S
	B0BTS1	M.b	如果 M(bank 0).b = 1，跳转到下一条指令。	-	-	-	1 + S
	JMP	d	跳转指令，PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	子程序调用指令，Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
MISC	RET		子程序跳出指令，PC ← Stack	-	-	-	2
	RETI		中断程序跳出指令，PC ← Stack，使能全局中断。	-	-	-	2
	PUSH		保存工作寄存器。	-	-	-	1
	POP		恢复工作寄存器。	√	√	√	1
	NOP		空指令。	-	-	-	1

注： 1. M 指系统寄存器或者 RAM，如果 M 指系统寄存器，则 N=0，否则 N=1。

2. 条件跳转指令中，满足条件则 S=1，否则 S=0。

13 电气特性

13.1 极限参数

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8P2524K, SN8P2524S, SN8P2524X..... 0°C ~ + 70°C
SN8P2524KD, SN8P2524SD, SN8P2524XD..... –40°C ~ + 85°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

13.2 电气特性

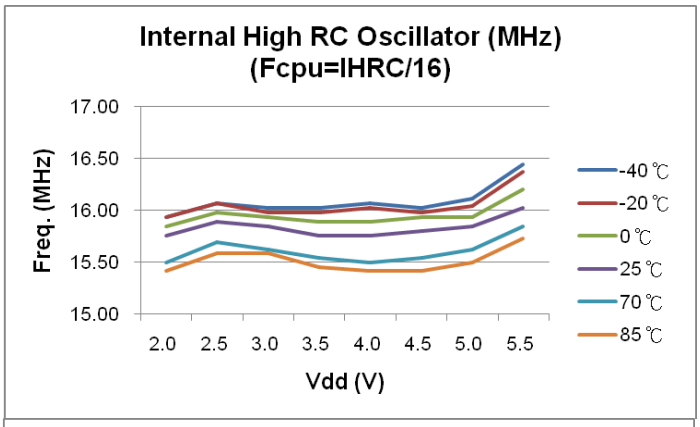
(All of voltages refer to Vss, Vdd = 5.0V, fosc = 16MHz,fcpu=1MHZ,ambient temperature is 25°C unless otherwise note.)

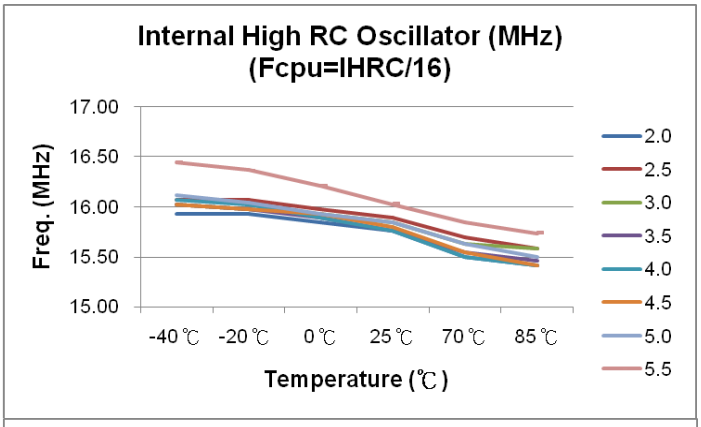
PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd, 25°C, Fcpu = 1MHz.		2.2	-	5.5	V
		Normal mode, Vpp = Vdd, -40°C~85°C		2.4	-	5.5	V
RAM Data Retention voltage	Vdr			1.5	-	-	V
*Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	All input ports		Vss	-	0.3Vdd	V
	ViL2	Reset pin		Vss	-	0.2Vdd	V
Input High Voltage	ViH1	All input ports		0.7Vdd	-	Vdd	V
	ViH2	Reset pin		0.9Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd		-	-	2	uA
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
		Vin = Vss , Vdd = 5V		50	100	150	
I/O output source current	IoH	Vop = Vdd – 0.5V		8	-	-	mA
sink current	IoL1	Vop = Vss + 0.5V		8	-	-	mA
	IoL2	Vop = Vss + 1.5V, P5.3, P5.4 only.		150	200	250	mA
*INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current (Disable Comparator)	Idd1	Run Mode (No loading, IHRC)	Vdd= 3V, Fcpu = 16MHz/2	-	5	-	mA
			Vdd= 5V, Fcpu = 16MHz/2	-	7	-	mA
			Vdd= 3V, Fcpu = 16MHz/4	-	2	-	mA
			Vdd= 5V, Fcpu = 16MHz/4	-	4	-	mA
			Vdd= 3V, Fcpu = 16MHz/16	-	1.5	-	mA
			Vdd= 5V, Fcpu = 16MHz/16	-	3	-	mA
	Idd2	Slow Mode (Internal low RC, Stop high clock)	Vdd= 3V, ILRC=16KHz	-	3.5	-	uA
			Vdd= 5V, ILRC=32KHz	-	10	-	uA
	Idd3	Sleep Mode	Vdd= 5V/3V	-	1	2	uA
	Idd4	Green Mode (No loading, Watchdog Disable)	Vdd= 3V, IHRC=16MHz	-	0.35	-	mA
			Vdd= 5V, IHRC=16MHz	-	0.55	-	mA
			Vdd= 3V, ILRC=16KHz	-	2	-	uA
			Vdd= 5V, ILRC=32KHz	-	5.5	-	uA
Internal High Oscillator Freq.	Fihrc	Internal High RC (IHRC)	25°C, Vdd=2.2V~ 5.5V Fcpu=Fhosc/2~Fhosc/16	15.68	16	16.32	MHz
			-40°C~85°C,Vdd=2.4V~ 5.5V Fcpu=Fhosc/2~Fhosc/16	15.2	16	16.8	MHz
*LVD Voltage	Vdet0	Low voltage reset level. 25°C		1.9	2.0	2.1	V
		Low voltage reset level. -40°C~85°C		1.8	2.0	2.3	V
	Vdet1	Low voltage reset/indicator level. 25°C		2.3	2.4	2.5	V
		Low voltage reset/indicator level. -40°C~85°C		2.2	2.4	2.7	V
	Vdet2	Low voltage reset/indicator level. 25°C		3.5	3.6	3.7	V
		Low voltage reset/indicator level. -40°C~85°C		3.3	3.6	3.9	V
Comparator Quiescent Current	Icmq	Iout=0		-	-	1	uA
Comparator Operating Current	Icmop	Internal reference disables. Vdd = 3V		-	30	-	uA
		Internal reference disables. Vdd = 5V		-	40	-	uA
		Internal reference enables. Vdd = 3V		-	50	-	uA
		Internal reference enables. Vdd = 5V		-	65	-	uA
Comparator Input Offset Voltage	*Vcmof	Vcm=Vss		-10	-	+10	mV
Common Mode Input Voltage Range	Vcm	Vdd=5.0V		Vss-0.3	-	Vdd+0.3	V

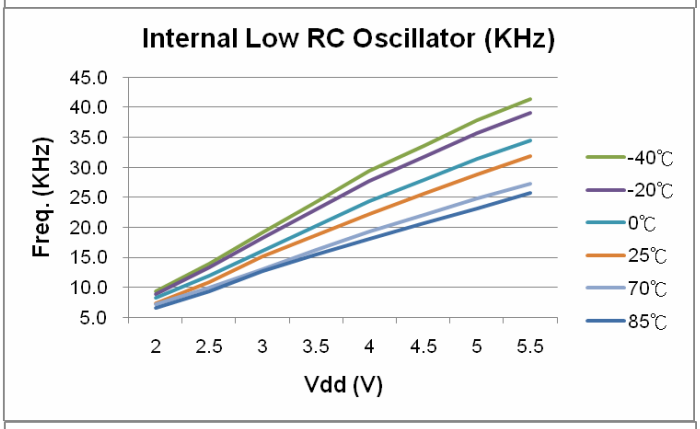
***These parameters are for design reference, not tested.**

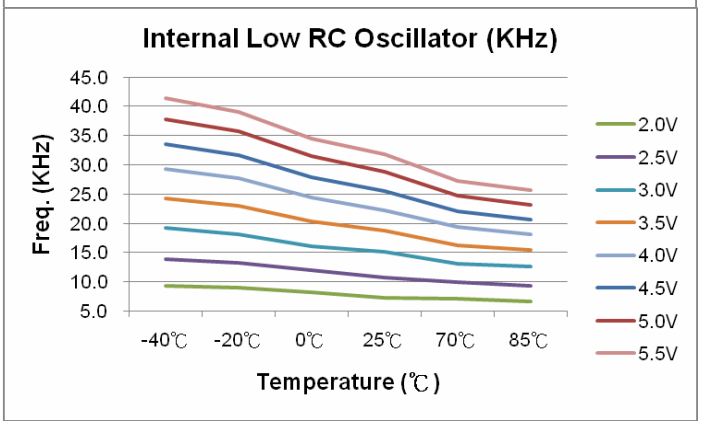
13.3 特性曲线图

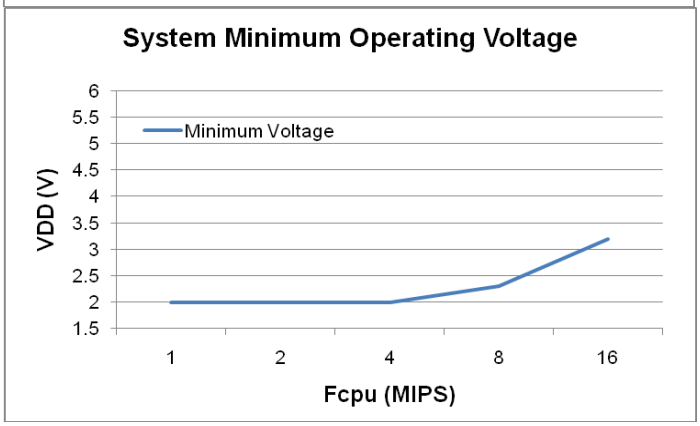
本章所列的各曲线图仅作设计参考，其中给出的部分数据可能超出了芯片指定的工作范围，为保证芯片的正常工作，请严格参照电气特性说明。











14 开发工具

在进行 SN8P2524 的开发时，SONIX 提供 ICE（在线仿真器），IDE（集成开发环境）和 EV-Kit 开发工具。ICE 和 EV-Kit 为外部硬件装置，IDE 有一个友好的用户界面进行韧件开发与仿真。各工具的版本如下所示：

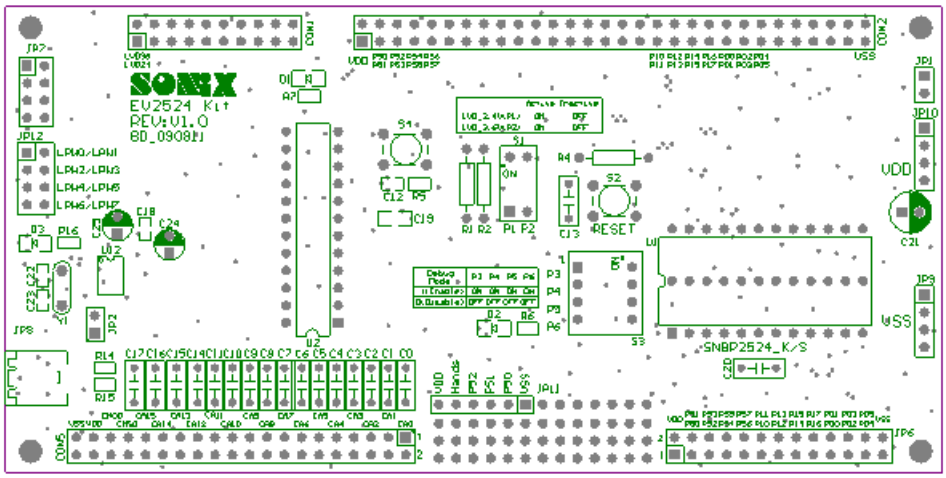
- **ICE： SN8ICE2K Plus 2。（在进行 SN8P2524 的仿真时，请在 ICE 选择 16MHz 的晶振。）**
- **ICE 最大仿真速度为：8MIPS @5V（如：16MHz 晶振，Fcpu=Fosc/2）。**
- **EV-kit：EV2524 Kit REV: V1.0。**
- **IDE：SONiX IDE M2IDE_V132 或更晚的版本。**
- **Writer：MPIII writer。**
- **Writer 转接板：SN8P2523。**

14.1 SN8P2524 EV-KIT

SN8P2524 EV- KIT 包括 ICE 接口，GPIO 接口和 EV-Chip，以及 PWM 模块。

- **EV-chip 模块：**仿真 12 通道比较器功能。
- **PWM 模块：**仿真 8 通道 PWM 功能。

SN8P2524 EV-kit PCB 如下所示：



- CON1, CON2：连接到 SN8ICE2K Plus2。
- JP6：GPIO 接口。
- U2：SN8P2524 EV-chip，用来仿真比较器功能。
- U1：SN8P2524 SKDIP/SOP 封装形式芯片接口，连接到用户目标板。
- S1：LVD24 和 LVD36 切换开关。
- C1~C11，C4~C17：16 通道比较器电容。
- CON5（CA0~CA15）：16 通道比较器接口。
- JP12：8 通道 PWM 接口。

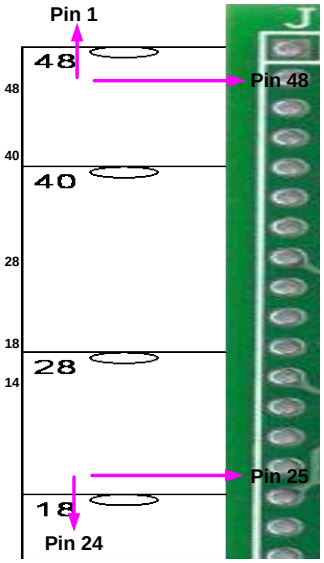
14.2 ICE和EV-KIT应用注意事项

- SN8P2524 EV- KIT 包括比较器仿真模块和 8-PWM 模块。
- SN8P2524 EV-Kit 上已经烧录好程序的 SN8P2524 芯片用来仿真比较器功能.
- 仿真比较器功能时，由 CA0~CA15 引脚输入比较器信号。
- 连接到 JP12（LPW0~LPW7）仿真 8-PWM 功能。
- 连接到 JP6 仿真 GPIO 功能。
- SN8P2524 EV-KIT 电源切换列表如下：

	JP1	JP2	JP10
ICE 电源	短接	断开	断开
外部电源	断开	断开	短接
USB 电源	断开	短接	断开

15 OTP烧录引脚

15.1 烧录器转接板引脚配置



JP3（连接 48-pin 接口）

DIP 1	1	48	DIP48
DIP 2	2	47	DIP47
DIP 3	3	46	DIP46
DIP 4	4	45	DIP45
DIP 5	5	44	DIP44
DIP 6	6	43	DIP43
DIP 7	7	42	DIP42
DIP 8	8	41	DIP41
DIP 9	9	40	DIP40
DIP10	10	39	DIP39
DIP11	11	38	DIP38
DIP12	12	37	DIP37
DIP13	13	36	DIP36
DIP14	14	35	DIP35
DIP15	15	34	DIP34
DIP16	16	33	DIP33
DIP17	17	32	DIP32
DIP18	18	31	DIP31
DIP19	19	30	DIP30
DIP20	20	29	DIP29
DIP21	21	28	DIP28
DIP22	22	27	DIP27
DIP23	23	26	DIP26
DIP24	24	25	DIP25

Writer JP1/JP2

VDD	1	2	VSS
CLK/PGCLK	3	4	CE
PGM/OTPCLK	5	6	OE/ShiftDat
D1	7	8	D0
D3	9	10	D2
D5	11	12	D4
D7	13	14	D6
VDD	15	16	VPP
HLS	17	18	RST
-	19	20	ALSB/PDB

JP1 连接烧录器转接板
JP2 连接 Dice 或大于 48pin 封装的芯片

15.2 烧录引脚配置

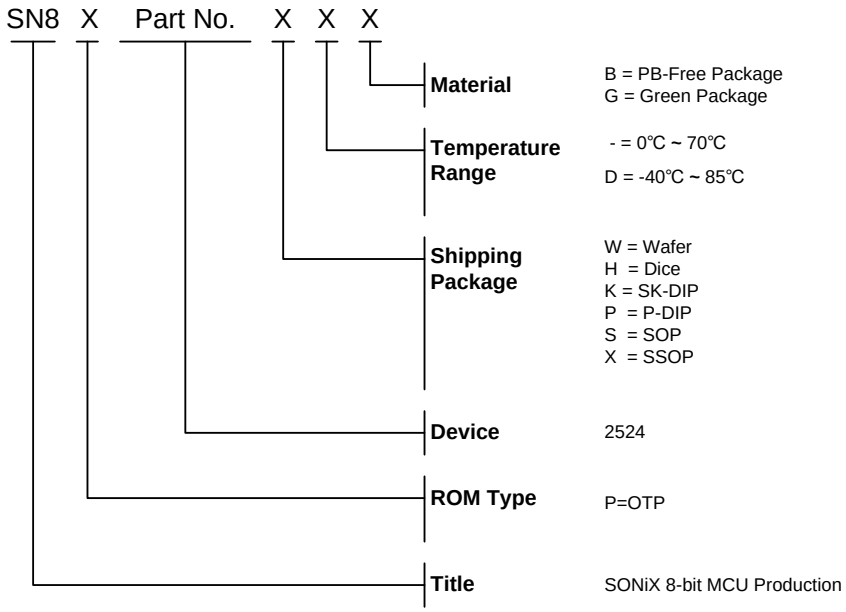
SN8P2524 系列的烧录引脚信息							
单片机名称		SN8P2524K/S(SKDIP/SOP)			SN8P2524X(SSOP)		
烧录器接口		IC and JP3 48-pin text tool 引脚配置					
JP1/JP2 引脚编号	JP1/JP2 引脚名称	IC 引脚编号	IC 引脚名称	JP3 引脚编号	IC 引脚编号	IC 引脚名称	JP3 引脚编号
1	VDD	7	VDD	19	9	VDD	19
2	GND	18	VSS	30	20	VSS	30
3	CLK	4	P1.2	16	6	P1.2	16
4	CE	-	-	-	-	-	-
5	PGM	6	P1.0	18	8	P1.0	18
6	OE	3	P1.3	15	5	P1.3	15
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP	8	RST	20	10	RST	20
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	5	P1.1	17	7	P1.1	17

16 单片机正印命名规则

16.1 概述

SONiX 8 位单片机产品具有多种型号，本章将给出所有 8 位单片机分类命名规则，适用于空片 OTP 型单片机。

16.2 单片机型号说明



16.3 命名举例

- **Wafer, Dice：**

名称	ROM 类型	器件名称（Device）	封装形式	温度范围	封装材料
S8P2524W	OTP	2524	Wafer	0℃~70℃	-
SN8P2524H	OTP	2524	Dice	0℃~70℃	-

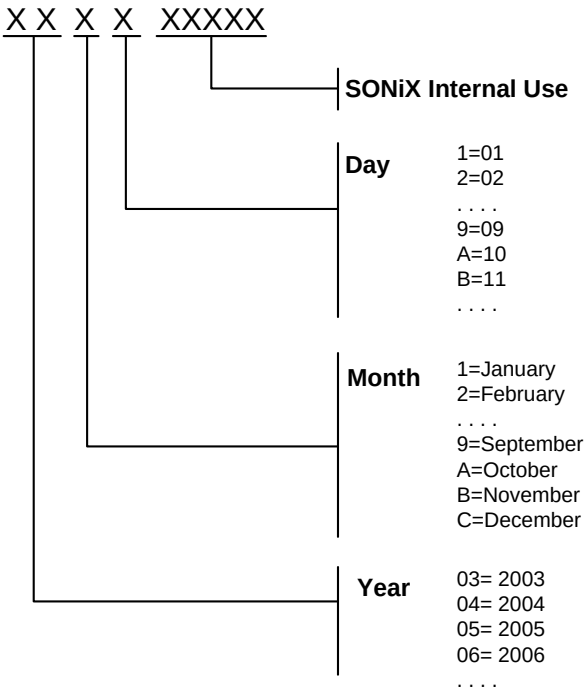
- **绿色封装：**

名称	ROM 类型	器件名称（Device）	封装形式	温度范围	封装材料
SN8P2524KG	OTP	2524	SK-DIP	0℃~70℃	绿色封装
SN8P2524SG	OTP	2524	SOP	0℃~70℃	绿色封装
SN8P2524XG	OTP	2524	SSOP	0℃~70℃	绿色封装
SN8P2524KDG	OTP	2524	SK-DIP	-40℃~85℃	绿色封装
SN8P2524SDG	OTP	2524	SOP	-40℃~85℃	绿色封装
SN8P2524XDG	OTP	2524	SSOP	-40℃~85℃	绿色封装

- **无铅封装：**

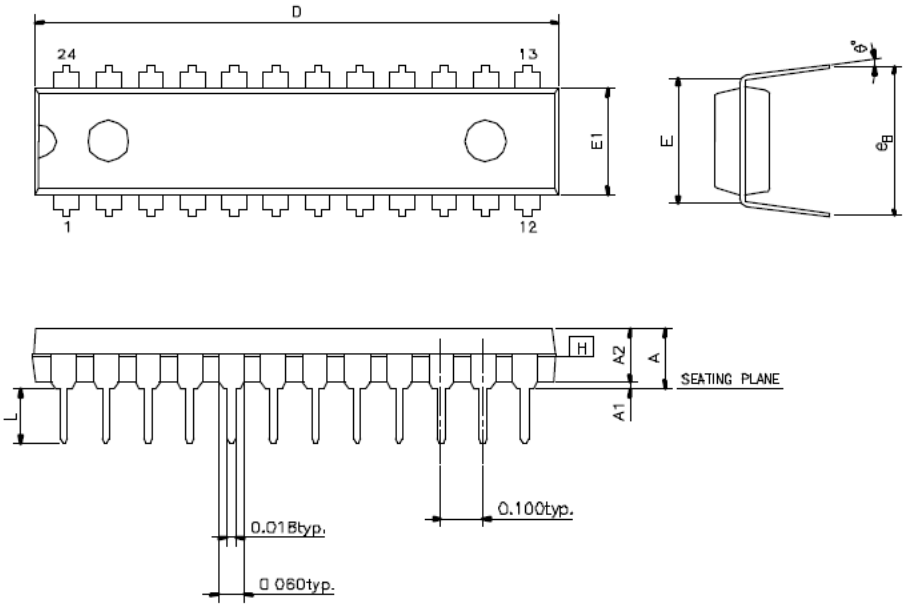
名称	ROM 类型	器件名称（Device）	封装形式	温度范围	封装材料
SN8P2524KB	OTP	2524	SK-DIP	0℃~70℃	无铅封装
SN8P2524SB	OTP	2524	SOP	0℃~70℃	无铅封装
SN8P2524XB	OTP	2524	SSOP	0℃~70℃	无铅封装
SN8P2524KDB	OTP	2524	SK-DIP	-40℃~85℃	无铅封装
SN8P2524SDB	OTP	2524	SOP	-40℃~85℃	无铅封装
SN8P2524XDB	OTP	2524	SSOP	-40℃~85℃	无铅封装

16.4 日期码规则


X X X X XXXXX
SONiX Internal Use
Day
1=01
2=02
. . . .
9=09
A=10
B=11
. . . .
Month
1=January
2=February
. . . .
9=September
A=October
B=November
C=December
Year
03= 2003
04= 2004
05= 2005
06= 2006
. . . .

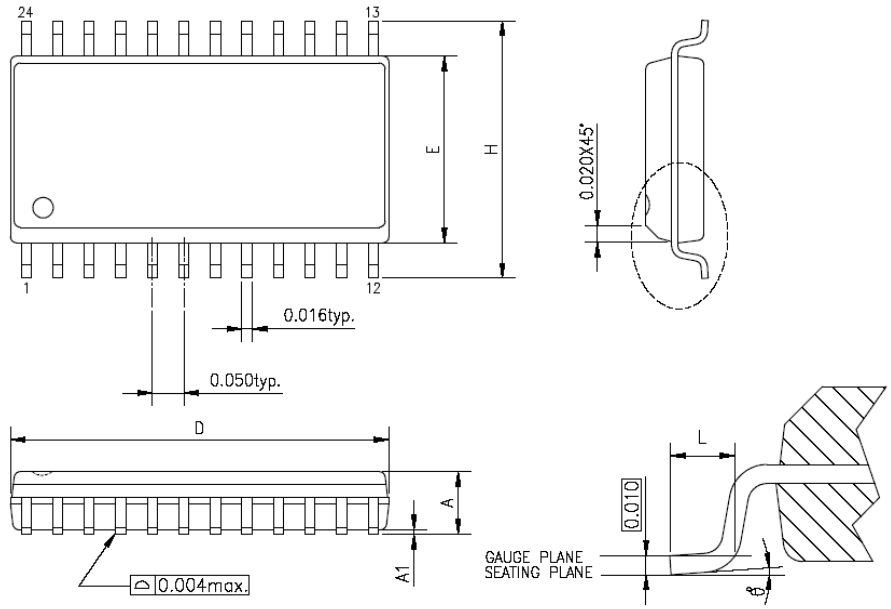
17 封裝信息

17.1 SK-DIP 24 PIN



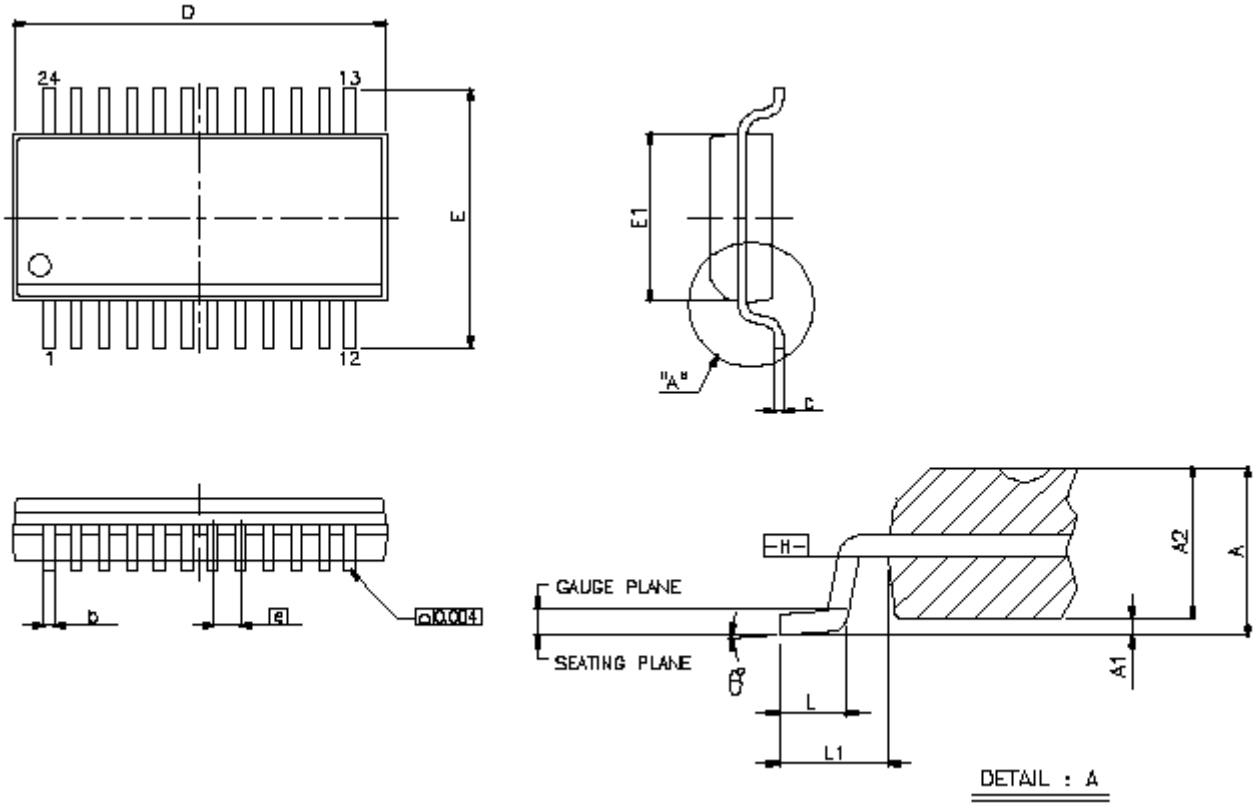
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-		0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	1.230	1.250	1.280	31.242	31.75	32.512
E	0.30 BSC			7.620 BSC		
E1	0.253	0.258	0.263	6.426	6.553	6.680
L	0.115	0.130	0.150	2.921	3.302	3.810
e B	0.335	0.355	0.375	8.509	9.017	9.525
θ°	0°	7°	15°	0°	7°	15°

17.2 SOP 24 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.069	-	-	1.753
A1	0.004	-	0.010	0.102	-	0.254
D	0.612	0.618	0.624	15.545	15.697	15.850
E	0.292	0.296	0.299	7.417	7.518	7.595
H	0.405	0.412	0.419	10.287	10.465	10.643
L	0.021	0.031	0.041	0.533	0.787	1.041
θ°	0°	4°	8°	0°	4°	8°

17.3 SSOP 24 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.053	0.064	0.069	1.346	1.625	1.752
A1	0.004	0.006	0.010	0.101	0.152	0.254
A2	-	-	0.059	-	-	1.499
D	0.337	0.341	0.344	8.559	8.661	8.737
E	0.228	0.236	0.244	5.791	5.994	6.197
E1	0.150	0.154	0.157	3.81	3.911	3.987
b	0.008	-	0.012	0.203	-	0.304
C	0.007	-	0.010	0.177	-	0.254
[e]	0.025 BASIC			0.635 BASIC		
L	0.016	0.025	0.050	0.406	0.635	1.27
L1	0.041 BASIC			1.041 BASIC		
θ°	0°	-	8°	0°	-	8°

SONiX 公司保留对以下所有产品在可靠性，功能和设计方面的改进作进一步说明的权利。SONiX 不承担由本手册所涉及的产品或电路的运用和使用所引起的任何责任，SONiX 的产品不是专门设计来应用于外科植入、生命维持和任何 SONiX 产品的故障会对个体造成伤害甚至死亡的领域。如果将 SONiX 的产品应用于上述领域，即使这些是由 SONIX 在产品设计和制造上的疏忽引起的，用户应赔偿所有费用、损失、合理的人身伤害或死亡所直接或间接产生的律师费用，并且用户保证 SONiX 及其雇员、子公司、分支机构和销售商与上述事宜无关。

总公司：
地址：台湾新竹县竹北市台元街 36 号 10 楼之一
电话：886-3-5600-888
传真：886-3-5600-889
台北办事处：
地址：台北市松德路 171 号 15 楼之 2
电话：886-2-2759 1980
传真：886-2-2759 8180
香港办事处：
地址：香港九龙湾宏开道 8 号其士商业中心 15 楼 1519 室
电话：852-2723 8086
传真：852-2723 9179
松翰科技（深圳）有限公司
地址：深圳市南山区高新技术产业园南区 T2-B 栋 2 层
电话：86-755-2671 9666
传真：86-755-2671 9786
技术支持：
Sn8fae@SONiX.com.tw