SN8P2735

用户参考手册

Version 1.0

SONiX 8 位单片机

修改记录

版本	日期	修改说明
VER 1.0	2009.5	初版。

目　　录

1 产品简介

1.1 功能特性

- **存储器配置**
  ROM：6K * 16 位。
  RAM：256 * 8 位。

- **8 层堆栈缓存器**

- **10 个中断源**
  7 个内部中断：T0、T1、TC0、ADC、CM0、CM1、CM2。
  3 个外部中断：INT0、INT1、INT2。

- **I/O 引脚配置**
  双向输入输出端口：P0、P1、P4、P5。
  具有唤醒功能的端口：P0、P1 的电平变化触发。
  内置上拉电阻的端口：P0、P1、P4、P5。
  Op-amp/比较器引脚：P1.0~P1.7、P5.0。
  ADC 输入引脚：P4.0~P4.7。

- **3 级 LVD**
  复位系统，监控电源。

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

- **功能强大的指令系统**
  指令的长度为一个字。
  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，具有外部事件计数，Buzzer 和 PWM 功能（TC0）。**
- **1 个 16 位抓取（capture）定时器 T1。**
- **6 通道 8/10/12 位 PWM。**
- **1 通道 8/10/12 位 PWM，具有死区和反向输出功能。**

- **8 通道 12 位 SAR ADC。**
- **3-set rail-to-rail OP 放大器。**
- **3-set rail-to-rail 比较器。**
- **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）。**

- **4 种系统时钟。**
  外部高速时钟：RC，高达 10MHz。
  外部高速时钟：晶振，高达 16MHz。
  内部高速时钟：RC，高达 16MHz。
  内部低速时钟：RC，16KHz（3V）、32KHz（5V）。

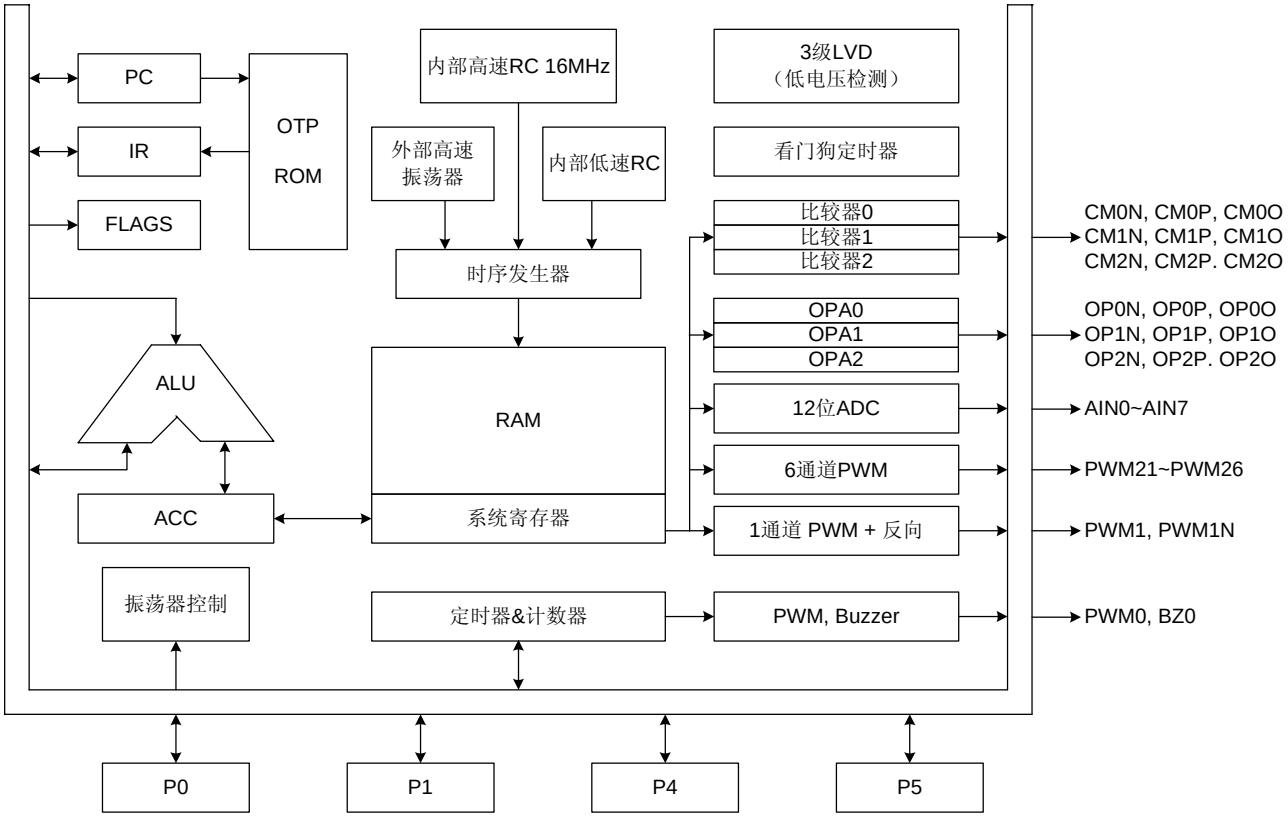
- **4 种工作模式**
  普通模式：高低速时钟都正常工作。
  低速模式：仅低速时钟工作。
  睡眠模式：高低速时钟都停止工作。
  绿色模式：由定时器周期性的唤醒。

- **封装形式**
  PDIP 32 pin
  LQFP 32 pin

☞ 产品性能比较表

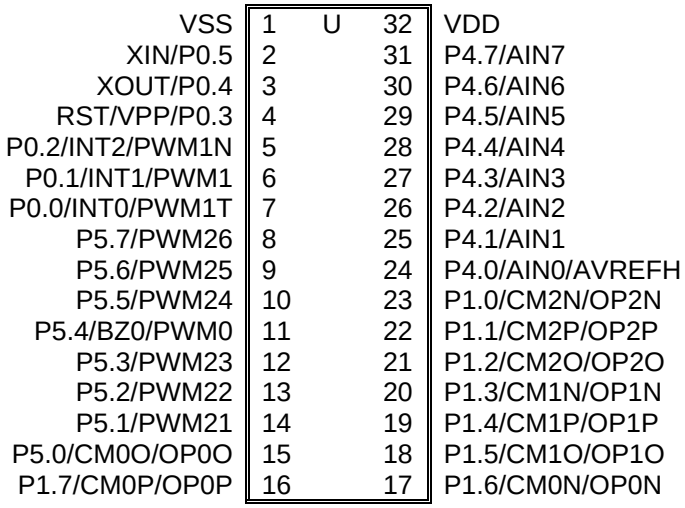
单片机名称	ROM	RAM	堆栈	定时器			I/O	PWM		ADC	OP 放大器	比较器	唤醒功能引脚输出	封装形式
				T0	TC0	T1		8 位	8~12 位					
SN8P2735	6K*16	256	8	V	V	V	30	1	7	8-ch	3	3	14	PDIP32/ LQFP32

1.2 系统框图

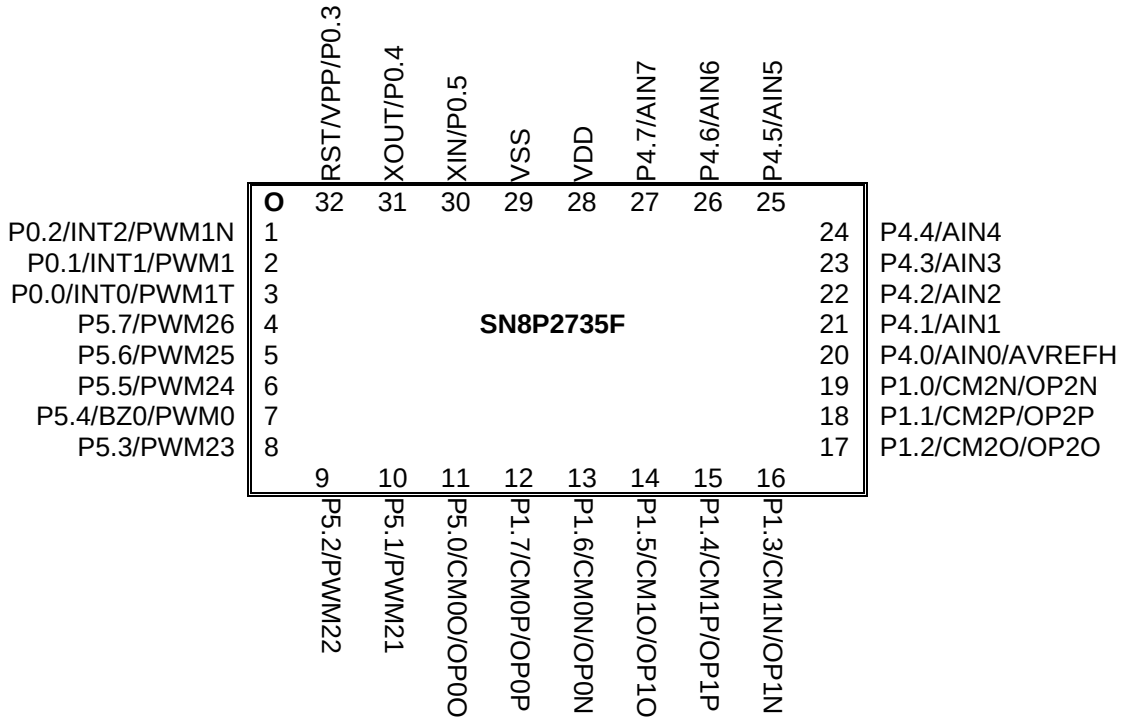

PC
IR
FLAGS
OTP
ROM
ALU
ACC
RAM
系统寄存器
振荡器控制
内部高速RC 16MHz
外部高速
振荡器
内部低速RC
时序发生器
定时器&计数器
3级LVD
（低电压检测）
看门狗定时器
比较器0
比较器1
比较器2
OPA0
OPA1
OPA2
12位ADC
6通道PWM
1通道 PWM + 反向
PWM, Buzzer
CM0N, CM0P, CM0O
CM1N, CM1P, CM1O
CM2N, CM2P. CM2O
OP0N, OP0P, OP0O
OP1N, OP1P, OP1O
OP2N, OP2P. OP2O
AIN0~AIN7
PWM21~PWM26
PWM1, PWM1N
PWM0, BZ0
P0
P1
P4
P5

1.3 引脚配置

SN8P2735P（PDIP 32 pins）



SN8P2735F（LQFP 32 pins）



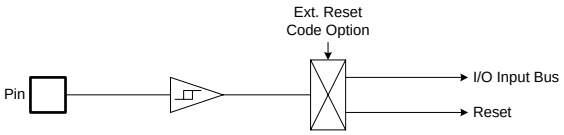
1.4 引脚说明

引脚名称	类型	功能说明
VDD, VSS	P	电源输入端。
P0.3/RST/VPP	I, P	RST：系统外部复位输入引脚，施密特触发，低电平有效，通常保持高电平。
		VPP：OTP 烧录引脚。
		P0.3：单向输入引脚，施密特触发，无上拉电阻。
XIN/P0.5	I/O	XIN：振荡信号输入引脚。
		P0.5：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
XOUT/P0.4	I/O	XOUT：振荡信号输出引脚。
		P0.4：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
P0.0/INT0/PWM1T	I/O	P0.0：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		INT0：外部中断 INT0 触发输入引脚。
		TC0 事件计数器输入引脚。
		PWM1T：PWM1 外部触发源。
P0.1/INT1/PWM1	I/O	P0.1：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		INT1：外部中断 INT1 触发输入引脚。
		PWM1：4 相加速 PWM 输出引脚。
P0.2/INT2/PWM1N	I/O	P0.2：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		INT2：外部中断 INT2 触发输入引脚。
		PWM1N：PWM1 负极输出引脚。
P1.0/CM2N/OP2N	I/O	P1.0：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		CM2N：比较器的负极输入引脚。
		OP2N：OP 放大器的负极输入引脚。
P1.1/CM2P/OP2P	I/O	P1.1：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		CM2P：比较器的正极输入引脚。
		OP2P：OP 放大器的正极输入引脚。
P1.2/CM2O/OP2O	I/O	P1.2：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		CM2O：比较器的输出引脚。
		OP2O：OP 放大器的输出引脚。
P1.3/CM1N/OP1N	I/O	P1.3：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		CM1N：比较器的负极输入引脚。
		OP1N：OP 放大器的负极输入引脚。
P1.4/CM1P/OP1P	I/O	P1.4：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		CM1P：比较器的正极输入引脚。
		OP1P：OP 放大器的正极输入引脚。
P1.5/CM1O/OP1O	I/O	P1.5：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		CM1O：比较器的输出引脚。
		OP1O：OP 放大器的输出引脚。
P1.6/CM0N/OP0N	I/O	P1.6：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		CM0N：比较器的负极输入引脚。
		OP0N：OP 放大器的负极输入引脚。
P1.7/CM0P/OP0P	I/O	P1.7：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		CM0P：比较器的正极输入引脚。
		OP0P：OP 放大器的正极输入引脚。
P5.0/CM0O/OP0O	I/O	P5.0：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		CM0O：比较器的输出引脚。
		OP0O：OP 放大器的输出引脚。
P5.1/PWM21	I/O	P5.1：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		PWM21：4 相加速 PWM2 通道 1 的输出引脚。
P5.2/ PWM22	I/O	P5.2：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		PWM22：4 相加速 PWM2 通道 2 的输出引脚。
P5.3/ PWM23	I/O	P5.3：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		PWM23：4 相加速 PWM2 通道 3 的输出引脚。

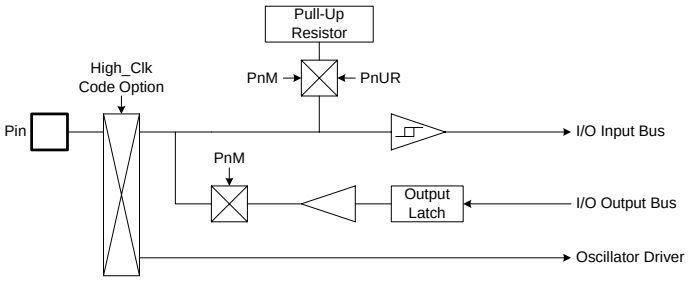
P5.4/BZ0/PWM0	I/O	P5.4：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		BZ0：可编程 Buzzer 输出引脚，信号来自 TC0/2。
		PWM0：可编程 PWM 输出引脚，信号来自 TC0。
P5.5/PWM24	I/O	P5.5：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		PWM24：4 相加速 PWM2 通道 4 的输出引脚。
P5.6/PWM25	I/O	P5.6：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		PWM25：4 相加速 PWM2 通道 5 的输出引脚。
P5.7/PWM26	I/O	P5.7：双向输入输出引脚，输入模式下为施密特触发，内置上拉电阻。
		PWM26：4 相加速 PWM2 通道 6 的输出引脚。
P4.0/AIN0/AVREFH	I/O	P4.0：双向输入输出引脚，无施密特触发，内置上拉电阻。
		AIN0：ADC 模拟输入引脚。
		AVREFH：ADC 参考高电压输入引脚。
P4.1/AIN1	I/O	P4.1：双向输入输出引脚，无施密特触发，内置上拉电阻。
		AIN1：ADC 模拟输入引脚。
P4.2/AIN2	I/O	P4.2：双向输入输出引脚，无施密特触发，内置上拉电阻。
		AIN2：ADC 模拟输入引脚。
P4.3/AIN3	I/O	P4.3：双向输入输出引脚，无施密特触发，内置上拉电阻。
		AIN3：ADC 模拟输入引脚。
P4.4/AIN4	I/O	P4.4：双向输入输出引脚，无施密特触发，内置上拉电阻。
		AIN4：ADC 模拟输入引脚。
P4.5/AIN5	I/O	P4.5：双向输入输出引脚，无施密特触发，内置上拉电阻。
		AIN5：ADC 模拟输入引脚。
P4.6/AIN6	I/O	P4.6：双向输入输出引脚，无施密特触发，内置上拉电阻。
		AIN6：ADC 模拟输入引脚。
P4.7/AIN7	I/O	P4.7：双向输入输出引脚，无施密特触发，内置上拉电阻。
		AIN7：ADC 模拟输入引脚。 请参考 ADC 章节内容。

1.5 引脚电路结构图

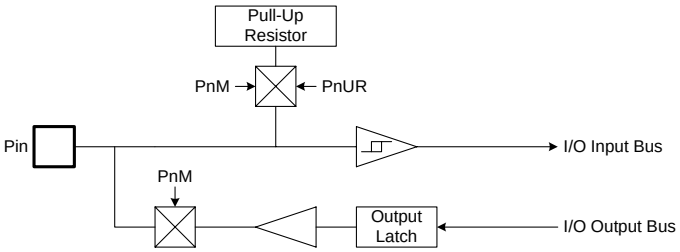
- **P0.3：**



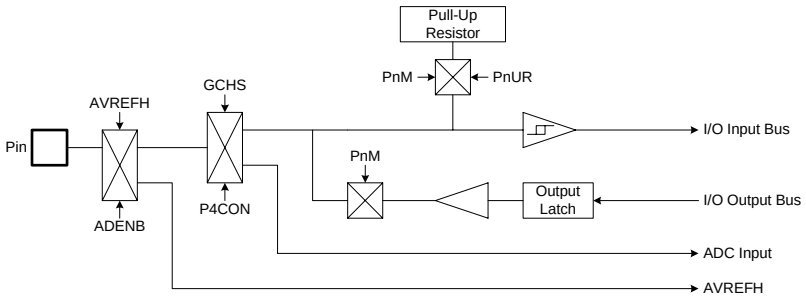
- **P0.4、0.5：**



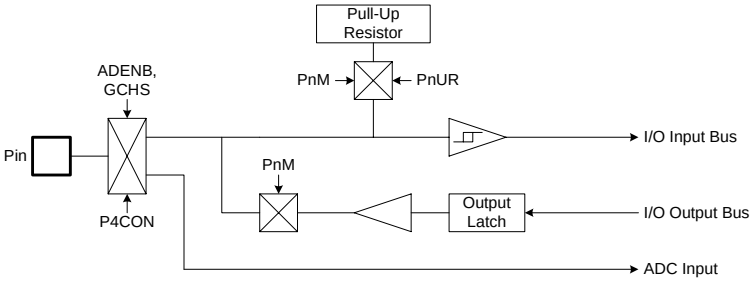
- **P0.0~P0.2、P5.1~P5.7：**



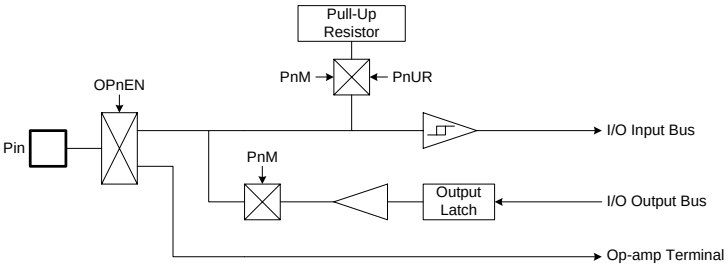
- **P4.0：**



- **P4.1~P4.7：**

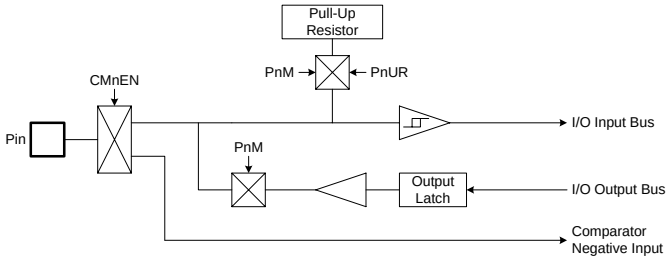


- **P1.0~P1.7、P5.0 OP 放大器共用引脚：**

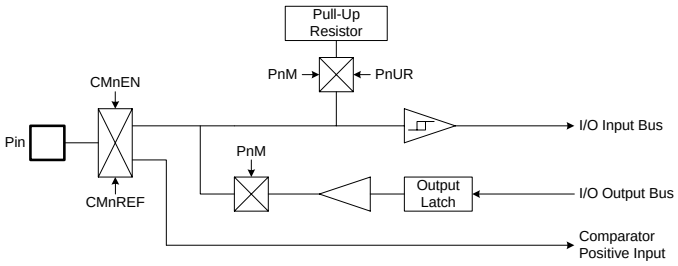


- **P1.0~P1.7, P5.0 比较器共用引脚：**

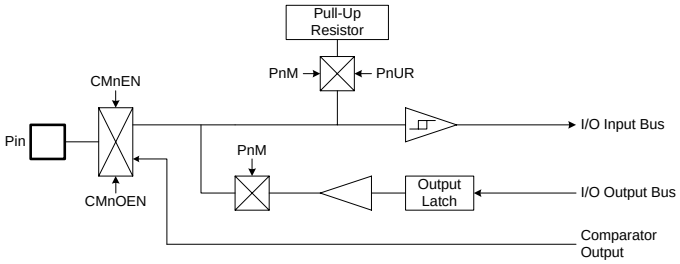
比较器负极输入引脚：



比较器正极输入引脚：



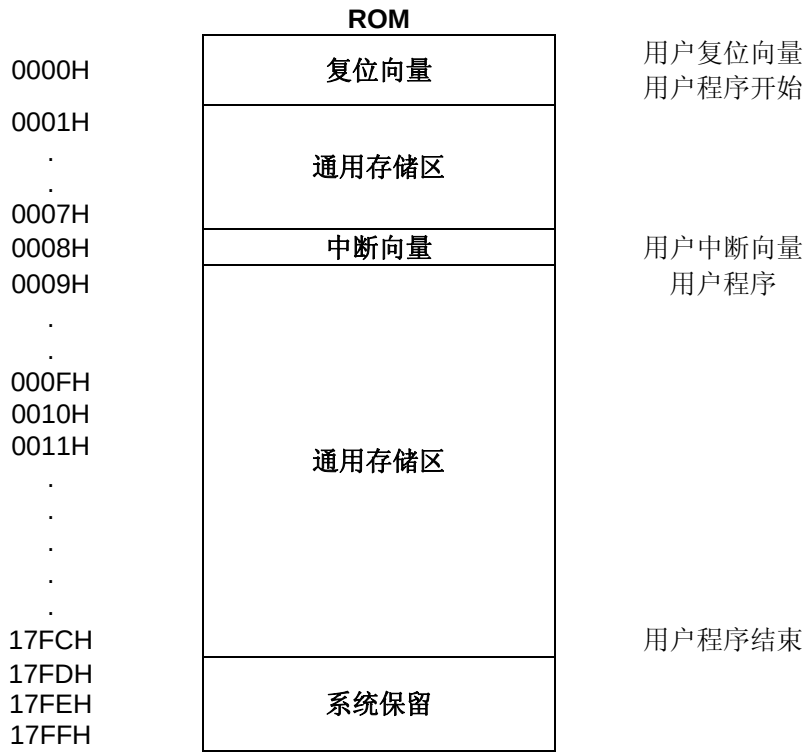
比较器输出引脚：



2 中央处理器（CPU）

2.1 程序存储器 ROM

☞ ***ROM：6K***



ROM 包含复位向量、中断向量、通用存储区域和系统保留部分。复位向量是程序的开始位置，中断向量则是中断服务程序的开始位置，而通用存储区域则保存主程序、子程序及数据表格。

2.1.1 复位向量（0000H）

具有一个字长的系统复位向量（0000H）。

☞ **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编写中断服务程序。

✷ **注：“PUSH”、“POP”指令只保存 ACC/PFLAG，无 NT0 和 NPD。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               ; 0000H。
                JMP     START           ; 跳转到用户程序。
                …

                ORG     8H              ; 中断向量。
                PUSH                    ; 保存 ACC 和 PFLAG。
                …
                …
                POP                     ; 恢复 ACC 和 PFLAG。
                RETI                    ; 中断返回。
                …

START:                                  ; 用户程序开始。
                …                       ; 用户程序。
                …
                JMP     START           ;
                …

                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               ; 0000H。
                JMP     START           ; 跳转到用户程序。
                …
                ORG     8H              ; 中断向量。
                JMP     MY_IRQ          ; 0008H，跳转到中断服务程序。

                ORG     10H
START:                                  ; 0010H，用户程序开始。
                …                       ; 用户程序。
                …
                …
                JMP     START           ;
                …
MY_IRQ:                                 ; 中断服务程序开始。
                PUSH                    ; 保存 ACC 和 PFLAG。
                …
                …
                POP                     ; 恢复 ACC 和 PFLAG。
                RETI                    ; 中断返回。
                …
                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 ROM 地址为“TABLE1”的值。**

```
        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。
        JMP     @F              ; Y 没有溢出。
        INCMS   X               ; Y 溢出，X=X+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。

        B0MOV   A, BUF          ; Z = Z + BUF。
        B0ADD   Z, A

        B0BTS1  FC              ; 检查进位标志。
        JMP     GETDATA         ; FC = 0。
        INCMS   Y               ; FC = 1，Y+1。
        NOP

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 字节**

	地址	RAM	
BANK 0	000h “ “ “ 07Fh	通用存储区	RAM Bank 0
	080h “ “ “ 0FFh	系统寄存器	080h~0FFh（Bank 0）为系统寄存器（128 bytes）。 Bank 0 结束区
BANK 1	100h “ “ “ 17Fh	通用存储区	RAM Bank 1 Bank 1 结束区

256 字节的通用存储区分别位于 Bank0 和 Bank1。由 RBANK 寄存器决定访问 RAM 的哪个 Bank 区，当 RBANK=0 时，PC 直接访问 Bank0；当 RBANK=1 时，PC 直接访问 Bank1。当在一个 Bank 区而需要访问另一个 Bank 区时，必须设置 RBANK 寄存器。SONIX 提供了“Bank0”指令（如 B0MOV，B0ADD，B0BTS1，B0BSET 等），可以在非 Bank0 区直接访问 Bank0。

➢ **例：在 Bank1 区直接访问 Bank0，将 Bank0（WK00）的值存入 Bank1（WK01）中。**

```
; Bank 1 (RBANK = 1)
        B0MOV   A, WK00         ; 使用 Bank 0 指令直接访问 Bank0 RAM。
        MOV     WK01,A
```

✷ **注：对应多 Bank RAM 程序，不易控制 RAM Bank 的选择。用户必须注意 RBANK 的控制条件，尤其在中断时。系统不会保存 RBANK，也不会将 RAM 转换到 Rank0，因此必须通过程序来设定。使用 Bank0 指令是一个比较好的方法。**

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	PW1NM	-	-	PW1M	PW2M	PW2CHS	PW1RH	PW1RL	PW2RH	PW2RL	-	-	CM0M	CM1M	CM2M	OPM
A	T1M	T1CL	T1CH	-	-	-	-	-	-	-	-	-	-	-	P4CON	-
B	-	ADM	ADB	ADR	ADT	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	-	-	P4M	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	P1	-	-	P4	P5	-	-	T0M	T0C	TC0M	TC0C	-	-	-	STKP
E	P0UR	P1UR	-	-	P4UR	P5UR	@HL	@YZ	-	-	-	-	-	-	-	-
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
R = 工作寄存器，ROM 查表数据寄存器
RBANK = RAM Bank 选择寄存器
PW1M = PWM 1 模式寄存器
PW2CHS = PWM 2 输出通道选择寄存器
PW2RH, L = PWM 2 重装缓存器
CM1M = 比较器 1 模式寄存器
OPM = OP 放大器 0~2 模式寄存器
T1CH, L = T1 计数寄存器
ADM = ADC 模式寄存器
ADR = ADC 精度选择寄存器
PEDGE = P0.0、P0.1、P0.2 边沿方向寄存器
INTEN = 中断使能寄存器
PnM = Pn 输入输出模式寄存器
PnUR = Pn 上拉电阻控制寄存器
PCH, PCL = 程序计数器
T0C = T0 计数寄存器
TC0C = TC0 计数寄存器
@HL = 间接寻址寄存器
STKP = 堆栈指针缓存器

Y, Z = 工作寄存器，@YZ 间接寻址寄存器，ROM 寻址寄存器
PFLAG = 特殊标志寄存器
PW1NM = 反向 PWM1 模式寄存器
PW2M = PWM 2 模式寄存器
PW1RH, L = PWM 1 重装缓存器
CM0M = 比较器 0 模式寄存器
CM2M = 比较器 2 模式寄存器
T1M = T1 模式寄存器
P4CON = P4 配置寄存器
ADB = ADC 数据缓存器
ADT = ADC 偏移校准寄存器
INTRQ = 中断请求寄存器
WDTR = 看门狗清零寄存器
Pn = Pn 数据寄存器
OSCM = 振荡模式控制寄存器
T0M = T0 模式寄存器
TC0M = TC0 模式寄存器
TC0R = TC0 自动重装缓存器
@YZ = 间接寻址寄存器
STK0~STK7 = 堆栈缓存器

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	PW1NEN	PW1D2	PW1D1	PW1D0	PW1DEN	PW1NV			R/W	PW1NM
093H	PW1EN	PW1rate2	PW1rate1	PW1rate0P	PW1CKS	PW1LN1	PW1LN0	PW1S	R/W	PW1M
094H	PW2EN	PW2rate2	PW2rate1	PW2rate0	PW2CKS	PW2LN1	PW2LN0		R/W	PW2M
095H			PW2CH6	PW2CH5	PW2CH4	PW2CH3	PW2CH2	PW2CH1	R/W	PW2CHS
096H	PW1GS	PW1GEN	PW1GD		PW1R11	PW1R10	PW1R9	PW1R8	R/W	PW1RH
097H	PW1R7	PW1R6	PW1R5	PW1R4	PW1R3	PW1R2	PW1R1	PW1R0	W	PW1RL
098H					PW2R11	PW2R10	PW2R9	PW2R8	R/W	PW2RH
099H	PW2R7	PW2R6	PW2R5	PW2R4	PW2R3	PW2R2	PW2R1	PW2R0	W	PW2RL
09CH	CM0EN	CM0IEN	CM0IRQ	CM0OEN	CM0REF	CM0OUT	CM0G1	CM0G0	R/W	CM0M
09DH	CM1EN	CM1IEN	CM1IRQ	CM1OEN	CM1REF	CM1OUT	CM1G1	CM1G0	R/W	CM1M
09EH	CM2EN	CM2IEN	CM2IRQ	CM2OEN	CM2REF	CM2OUT	CM2G1	CM2G0	R/W	CM2M
09FH						OP2EN	OP1EN	OP0EN	R/W	OPM
0A0H	T1ENB	T1rate2	T1rate1	T1rate0	CPTCKS	CPTStart	CPTG1	CPTG0	R/W	T1M
0A1H	T1CL7	T1CL6	T1CL5	T1CL4	T1CL3	T1CL2	T1CL1	T1CL0	R/W	T1CL
0A2H	T1CH7	T1CH6	T1CH5	T1CH4	T1CH3	T1CH2	T1CH1	T1CH0	R/W	T1CH
0AEH	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	W	P4CON
0B1H	ADENB	ADS	EOC	GCHS	AVREFH	CHS2	CHS1	CHS0	R/W	ADM
0B2H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R	ADB
0B3H		ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0	R/W	ADR
0B4H	ADTS1	ADTS0		ADT4	ADT3	ADT2	ADT1	ADT0	R/W	ADT
0B8H			P05M	P04M	-	P02M	P01M	P00M	R/W	P0M
0BFH			P02G1	P02G0	P01G1	P01G0	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
0C4H	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4M
0C5H	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C8H	ADCIRQ	T1IRQ	TC0IRQ	T0IRQ		P02IRQ	P01IRQ	P00IRQ	R/W	INTRQ
0C9H	ADCIEN	T1IEN	TC0IEN	T0IEN		P02IEN	P01IEN	P00IEN	R/W	INTEN
0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH				PC12	PC11	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
0D4H	P47	P46	P45	P44	P43	P42	P41	P40	R/W	P4
0D5H	P57	P56	P55	P54	P53	P52	P51	P50	R/W	P5
0D8H	T0ENB	T0rate2	T0rate1	T0rate0				T0TB	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H			P05R	P04R	-	P02R	P01R	P00R	W	P0UR
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E4H	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R	W	P4UR
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
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H				S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H				S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H				S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H				S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H				S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH				S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH				S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH				S0PC12	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

✷ 注：

1. 所有寄存器名都已在SN8ASM编译器中做了宣告；
2. 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”；
3. 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 ACC

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
```

系统执行中断操作时，ACC 和 PFLAG 的值不会自动存储，用户必须使用 PUSH 和 POP 指令来保存 ACC 和 PFLAG。

➢ **例：ACC 和工作寄存器中断保护操作。**

```
INT_SERVICE:
                PUSH                    ; 保存 ACC 和 PFALG。
                …
                …
                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。LVD24、LVD36显示LVD检测电压状态。

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
复位后	-	-	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	保留
1	0	LVD 复位
1	1	外部复位

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 PC

程序计数器 PC 是一个 13 位二进制程序地址寄存器，分高 5 位和低 8 位。专门用来存放下一条需要执行指令的地址。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	-	-	-	PC12	PC11	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	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：

- **普通工作寄存器；**
- **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：

- **作为工作寄存器使用；**
- **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	-	-	-	-	-	-	-	-

2.3 寻址模式

2.3.1 立即寻址

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

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

通过数据指针（H/L、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
```

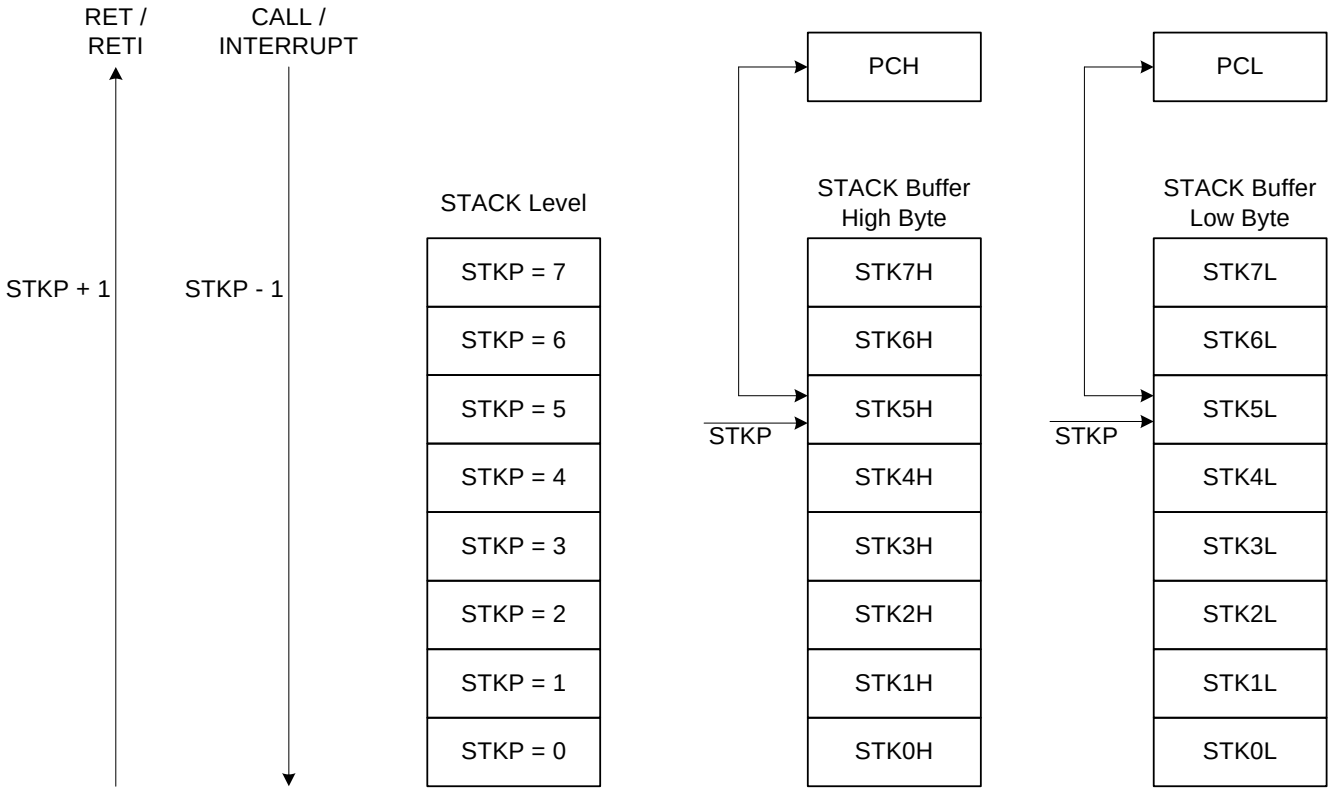
- **例：通过指针@YZ 间接寻址。**

```
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

SN8P2735 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，13 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	-	-	-	SnPC12	SnPC11	SnPC10	SnPC9	SnPC8
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	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		注释
	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		注释
	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（CODE OPTION）

编译选项（CODE OPTION）是一种系统的硬件配置，包括振荡器的类型，杂讯滤波器的选项。看门狗定时器的工作，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
High_Clk	IHRC_16M	高速内部 16MHz RC 振荡器，XIN/XOUT 作为普通的 I/O 引脚。
	IHRC_RTC	高速内部 16MHz RC 振荡器，XIN/XOUT 连接外部 32768Hz 晶振。
	RC	外部高速振荡器采用廉价的 RC 振荡电路，XIN 连接 RC 电路，XOUT 作为普通的 I/O 引脚。
	32K X'tal	外部高速振荡器采用低频振荡器（如 32768Hz）。
	12M X'tal	外部高速振荡器采用高频振荡器（如 12MHz）。
	4M X'tal	外部高速振荡器采用标准振荡器（如 4MHz）。
Fcpu	Fhosc/1	指令周期为 1 个振荡时钟。 注：在 Fosc/1 下，必须禁止杂讯滤波器。
	Fhosc/2	指令周期为 2 个振荡时钟。 注：在 Fosc/2 下，必须禁止杂讯滤波器。
	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。
Noise_Filter	Enable	使能杂讯滤波器，Fcpu = Fosc/4~Fosc/16。
	Disable	禁止杂讯滤波器，Fcpu = Fosc/2~Fosc/16。
Watch_Dog	Always_On	始终开启看门狗定时器，在睡眠模式和绿色模式下也处于开启状态。
	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。
Reset_Pin	Reset	使能外部复位引脚。
	P03	使能 P0.3 为单向输入引脚，无上拉电阻。
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。

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@3V，32KHz@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。
- **P03**：使能 P0.3 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 Secuity 编译选项，ROM 代码加密。

2.5.4 Noise Filter 编译选项

Noise Filter 编译选项是一个电源杂讯滤波器，用来减少噪音对系统时钟的影响。如果使能杂讯滤波器，Fcpu 不能选择 Fhosc/1 和 Fhosc/2 选项，Fcpu 的最快速率为 Fhosc/4；如果禁止杂讯滤波器，Fhosc/1 和 Fhosc/2 选项被释放。在高干扰环境下，使能杂讯滤波器，看门狗定时器以及选择一个适合的 LVD 选项可以使系统更好工作，避免出错。

3 复位

3.1 概述

SN8P2735 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。

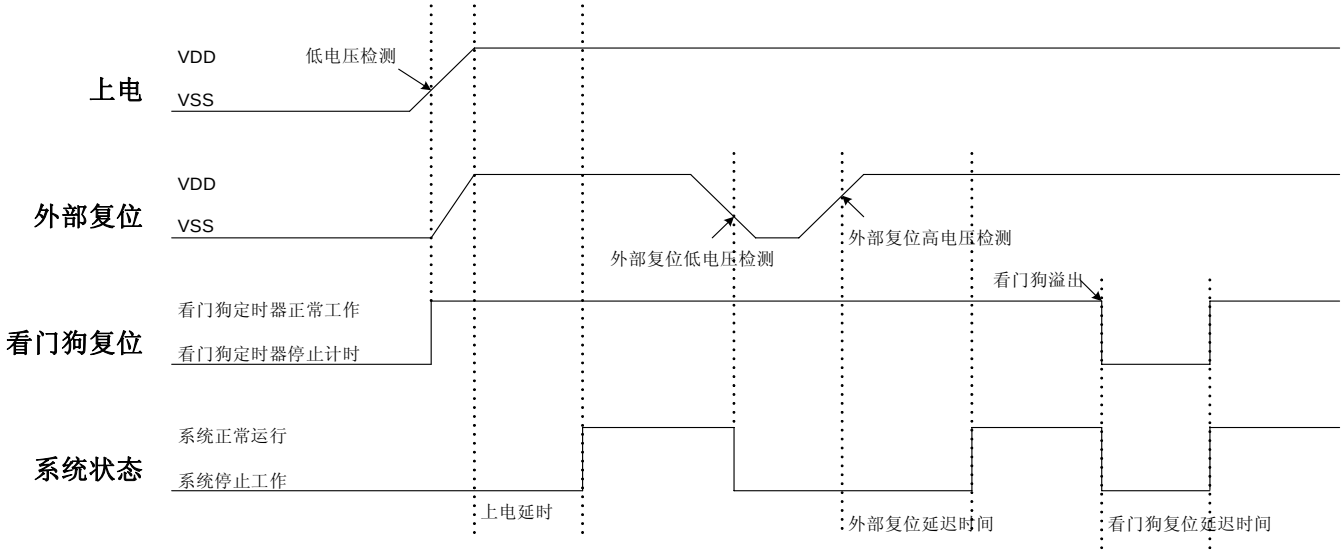
上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器 PC 清零。复位结束后，系统从向量 0000H 处重新开始运行。PFLAG 寄存器的 NT0 和 NPD 两个标志位能够给出系统复位状态的信息。用户可以编程控制 NT0 和 NPD，从而控制系统的运行路径。

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
复位后	-	-	0	0	-	0	0	0

Bit [7:6] **NT0, NPD**：复位状态标志位。

NT0	NPD	复位情况	说明
0	0	看门狗复位	看门狗定时器溢出
0	1	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。



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。

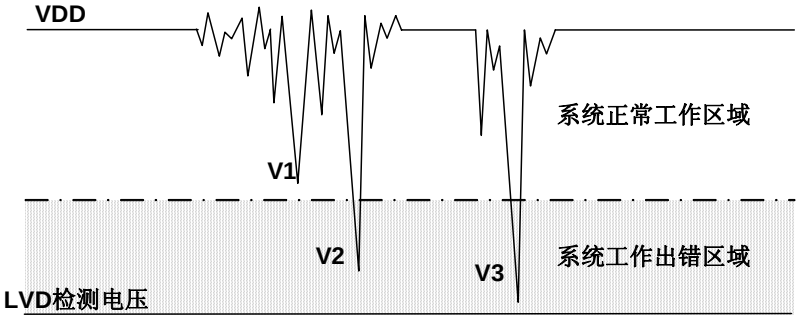
看门狗定时器应用注意事项如下：

- 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。

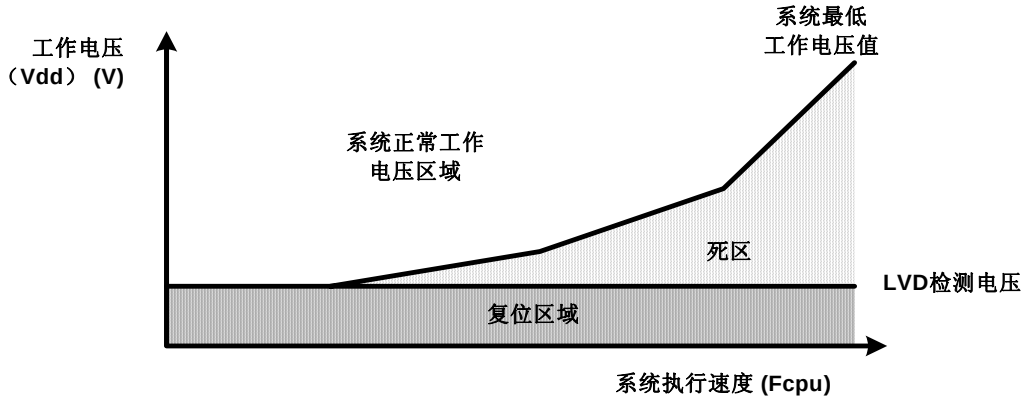
AC 运用中：

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。

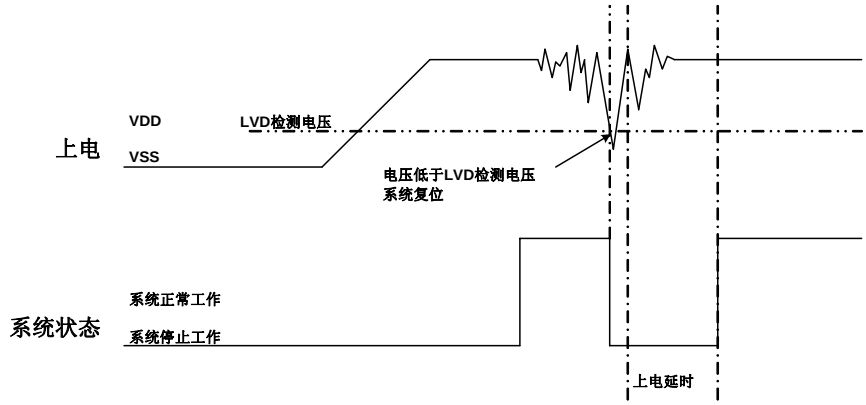


系统工作电压与执行速度关系图

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
复位后	-	-	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
2.0V 复位	有效	有效	有效
2.4V 标志	-	有效	-
2.4V 复位	-	-	有效
3.6V 标志	-	-	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；
LVD24：如果 VDD > 2.4V，LVD24 ＝0；如果 VDD <= 2.4V，LVD24＝1；
LVD36 标志位无意义。

LVD_H
如果 VDD < 2.4V，系统复位；
LVD36：如果 VDD > 3.6V，LVD36＝0；如果 VDD <= 3.6V，LVD36＝1；
LVD24 标志位无意义。

✷ **注：**
a）LVD 复位结束后，LVD24 和 LVD36 都将被清零；
b）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）。**

✷ **注：“稳压二极管复位电路”、“电压偏移复位电路”和“外部 IC 复位”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。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

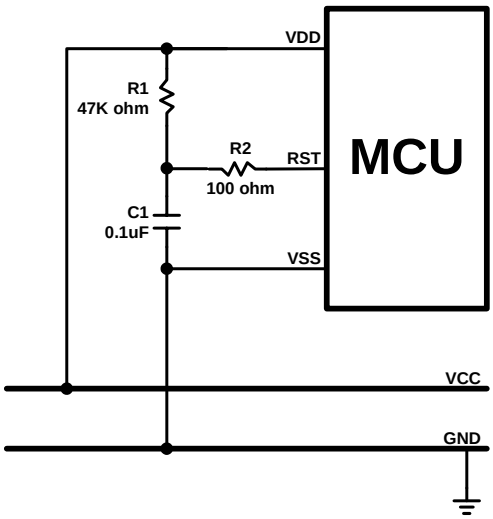
外部复位功能由编译选项“Reset_Pin”控制。将该编译选项置为“Reset”，可使能外部复位功能。外部复位引脚为施密特触发结构，低电平有效。复位引脚处于高电平时，系统正常运行。当复位引脚输入低电平信号时，系统复位。外部复位操作在上电和正常工作模式时有效。需要注意的是，在系统上电完成后，外部复位引脚必须输入高电平，否则系统将一直保持在复位状态。外部复位的时序如下：

- **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

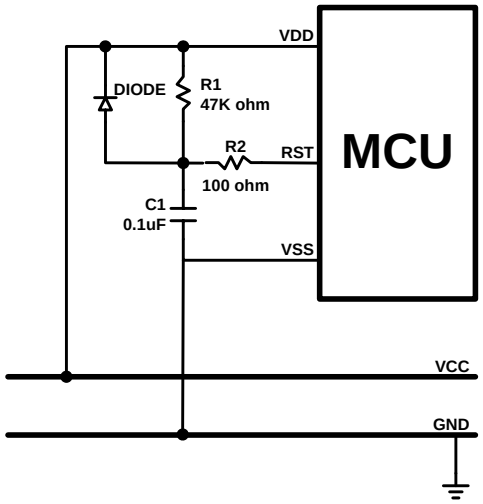
3.6.1 RC 复位电路



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。**

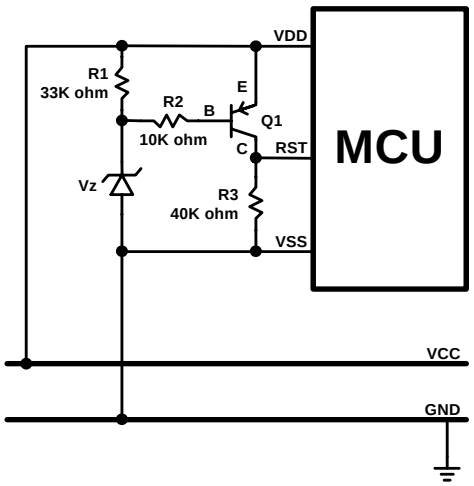
3.6.2 二极管及 RC 复位电路



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

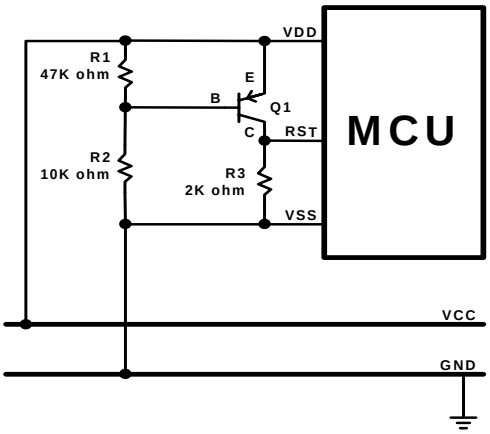
✷ **注："基本 RC 复位电路"和"二极管及 RC 复位电路"中的电阻 R2 都是必不可少的限流电阻，以避免复位引脚 ESD（Electrostatic Discharge）或 EOS（Electrical Over-stress）击穿 。**

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

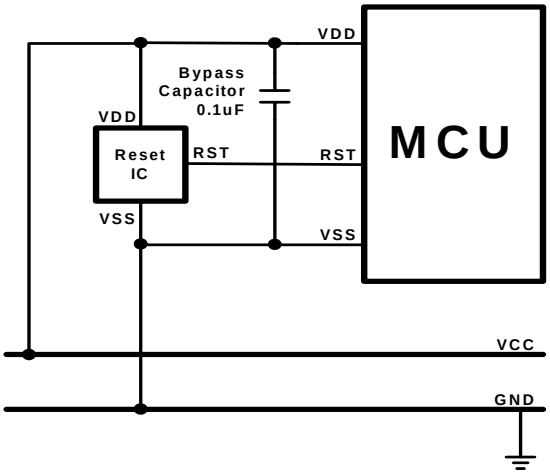


电压偏置复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，这种复位电路的检测电压值的精确度有所降低。电路中，R1 和 R2 构成分压电路，当 VDD 高于和等于分压值“0.7V x (R1 + R2) / R1”时，三极管集电极 C 输出高电平，单片机正常工作；VDD 低于“0.7V x (R1 + R2) / R1”时，集电极 C 输出低电平，单片机复位。

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

SN8P2735 内置双时钟系统：高速时钟和低速时钟。高速时钟包括内部高速振荡器和外部高速振荡器，由编译选项 High_CLK 选择。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。

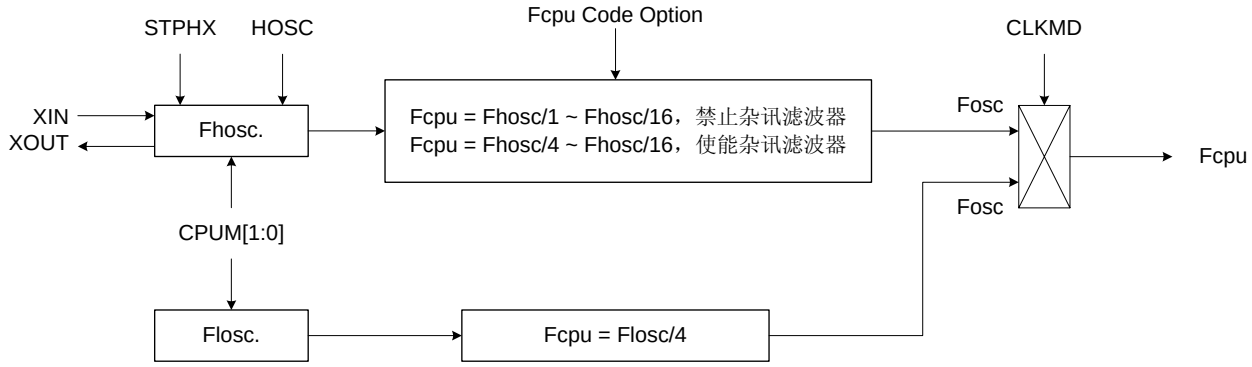
- **高速振荡器**

  内部高速振荡器：RC，高达 16MHz，称为 IHRC；
  外部高速振荡器：晶体/陶瓷（4MHz/12MHz/32KHz）和 RC。

- **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**



- HOSC：High_Clk 编译选项。
- Fhosc：外部高速时钟频率/内部高速时钟频率。
- 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。

在干扰较严重的运用条件下，SONiX 提供的杂讯滤波器能够对外部干扰进行隔离以保护系统的正常工作。但在杂讯滤波器有效时，高速时钟的 Fcpu 的最小选项为 Fcpu=Fhosc/4。

4.2 FCPU（指令周期）

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。当系统高速时钟源由外部 4MHz 晶振提供时，Fcpu 编译选项选择 Fhosc/4，则 Fcpu 频率为 4MHz/4=1MHz。低速模式下，Fcpu=Flosc/4，即 16KHz/4=4KHz@3V，32KHz/4=8KHz@5V。

Fcpu 的范围由高速时钟和杂讯滤波器编译选项控制。当高速时钟选项选择 RC、12M X'tal、4M X'tal 或者 32K X'tal 时，Fcpu=Fhosc/1~Fhosc/16；当高速时钟选项选择 IHRC_16M 或者 IHRC_RTC 时，Fcpu=Fhosc/2~Fhosc/16。若关闭杂讯滤波器，Fcpu 的范围取决于高速时钟的选项；若开启杂讯滤波器，Fcpu=Fhosc/4~Fhosc/16，以减少干扰。

4.3 杂讯滤波器（NOISE FILTER）

杂讯滤波器（由编译选项“Noise_Filter”控制）是一个低通滤波器，支持外部振荡器，包括 RC 和晶体模式。杂讯滤波器可以滤除来自外部振荡器的高干扰信号。

在高干扰环境下，强烈建议开启杂讯滤波器以减少干扰的影响。

4.4 系统高速时钟

系统高速时钟包括外部高速时钟和内部高速时钟。外部高速时钟又包括4MHz、12MHz、32KHz晶体/陶瓷和RC振荡器，高速时钟振荡器由编译选项High_CLK选择。内部高速时钟支持实时时钟（RTC）功能，在IHRC_RTC模式下，内部高速时钟和外部32KHz振荡器有效，内部高速时钟作为系统时钟源，而外部32KHz振荡器为RTC时钟源，提供一个精确的实时时钟频率。

4.4.1 HIGH_CLK 编译选项

对应不同的时钟功能，SONIX 提供多种高速时钟选项，由 High_CLK 选项控制。High_CLK 选项可以选择 IHRC_16M、IHRC_RTC、RC、32K X'tal、12M X'tal 和 4M X'tal，以支持不同带宽的振荡器。

- **IHRC_16M**：系统高速时钟源来自内部高速 16MHz RC 振荡器，XIN/XOUT 作为普通的 I/O 引脚，不连接任何外部中断设备。
- **IHRC_RTC**：系统高速时钟源来自内部高速 16MHz RC 振荡器，RTC 时钟源为外部低速 32768Hz 振荡器。XIN/XOUT 连接外部 32768Hz 晶振，其 I/O 功能被禁止。
- **RC**：系统高速时钟源来自廉价的 RC 振荡电路，RC 振荡电路只需要和 XIN 引脚连接，XOUT 作为普通的 I/O 引脚。
- **32K X'tal**：系统高速时钟源来自外部低频 32768Hz 振荡器。该选项仅支持 32768Hz 晶体振荡器，RTC 正常工作。
- **12M X'tal**：系统高速时钟源来自外部高频晶体/陶瓷振荡器，其带宽为 10MHz~16MHz。
- **4M X'tal**：系统高速时钟源来自外部高频晶体/陶瓷振荡器，其带宽为 1MHz~10MHz。

关于功耗，选择 IHRC_RTC 选项时，绿色模式下内部高速振荡器和内部低速振荡器都停止工作，仅外部 32768Hz 晶振正常工作，此时，看门狗定时器不能选择 Always_On 选项，否则内部低速振荡器会正常工作。

4.4.2 内部高速 RC 振荡器（IHRC）

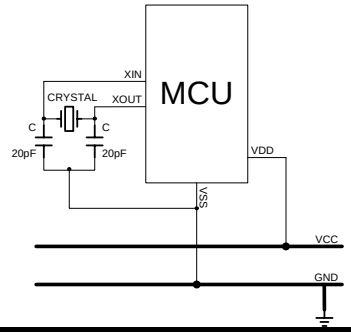
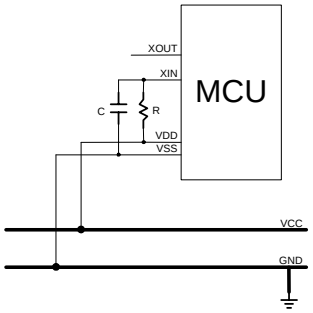
内部高速 16MHz RC 振荡器，普通环境下精确度为±2%，当选择 IHRC_16M 或者 IHRC_RTC 时，使能内部高速振荡器。

- **IHRC_16M**：系统高速时钟为内部 16MHz RC 振荡器，XIN/XOUT 为普通 I/O 引脚。
- **IHRC_RTC**：系统高速时钟为内部 16MHz RC 振荡器，外部 32768Hz 晶振作为试试时钟的时钟源，XIN/XOUT 连接外部 32768Hz 晶振。

4.4.3 外部高速振荡器

外部高速振荡器包括 4MHz，12MHz，32KHz 和 RC。4M、12M 和 32K 可以使用晶体和陶瓷振荡器，XIN/XOUT 和 GND 之间需连接一个 20pF 的电容。廉价的 RC 振荡电路只需要和 XIN 引脚连接，电容的容值不能低于 100pF，电阻的阻值由频率决定。

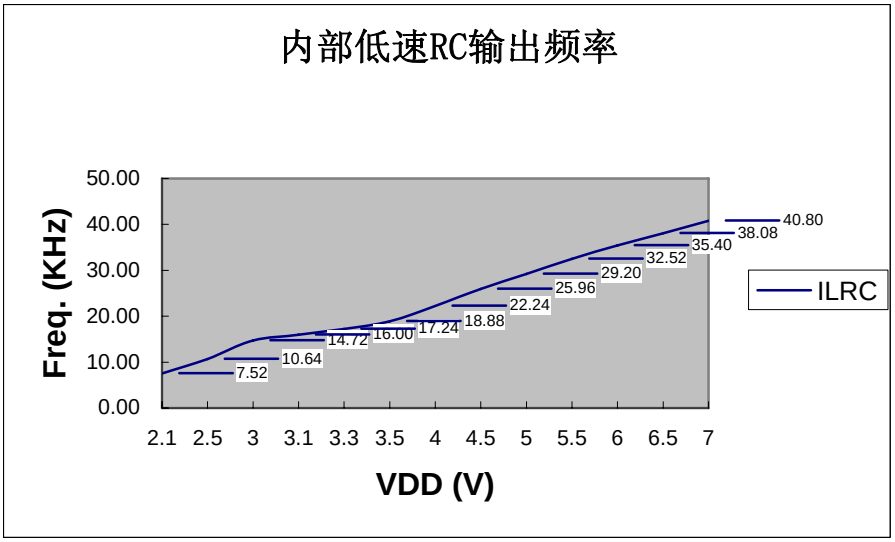
4.4.4 外部振荡应用电路

晶体/陶瓷	RC
	

✷ **注：晶体/陶瓷和电容 C 要尽可能的靠近单片机的 XIN/XOUT/VSS；电阻 R 和电容 C 要尽可能的靠近单片机的 VDD。**

4.5 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的时钟源。由 CLKMD 控制系统低速工作模式。

- 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。**

4.6 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	-
复位后	-	-	-	0	0	0	0	-

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

➢ **例：停止高速振荡器。**

```
B0BSET      FSTPHX
```

➢ **例：进入睡眠模式时，停止高速及低速振荡器。**

```
B0BSET      FCPUM0
```

STPHX 位为内部高速 RC 振荡器和外部振荡器的模式控制位。当 STPHX=0，外部振荡器和内部高速 RC 振荡器正常运行；当 STPHX=1，外部振荡器或内部高速 RC 振荡器停止运行。不同的高速时钟选项决定不同的 STPHX 功能。

- **IHRC_16M：STPHX=1，禁止内部高速 RC 振荡器；**
- **IHRC_RTC：STPHX=1，禁止内部高速 RC 振荡器和外部 32768Hz 振荡器；**
- **RC, 4M, 12M, 32K：STPHX=1，禁止外部振荡器。**

4.7 系统时钟测试

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。**

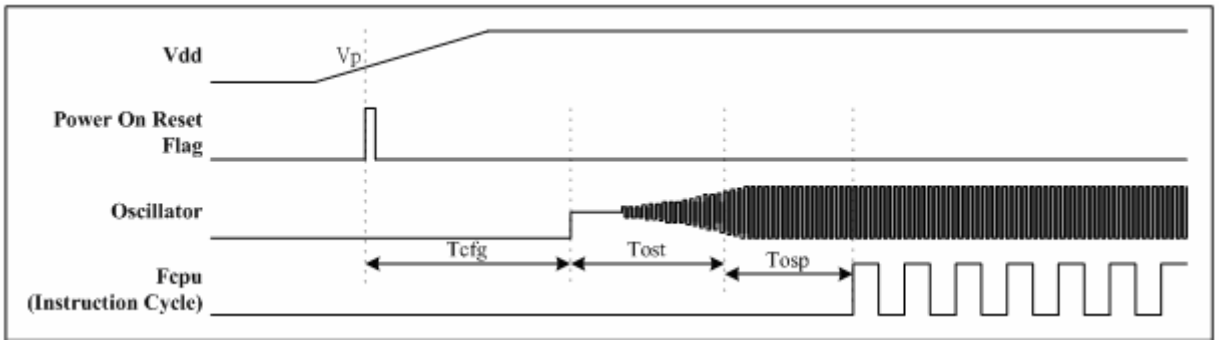
```
B0BSET      P0M.0           ; P0.0 置为输出模式以输出 Fcpu 的触发信号。

@@:
            B0BSET      P0.0
            B0BCLR      P0.0
            JMP         @B
```

✷ **注：不能直接从 XIN 引脚测试 RC 振荡频率，因为探针的连接会影响测试的准确性。**

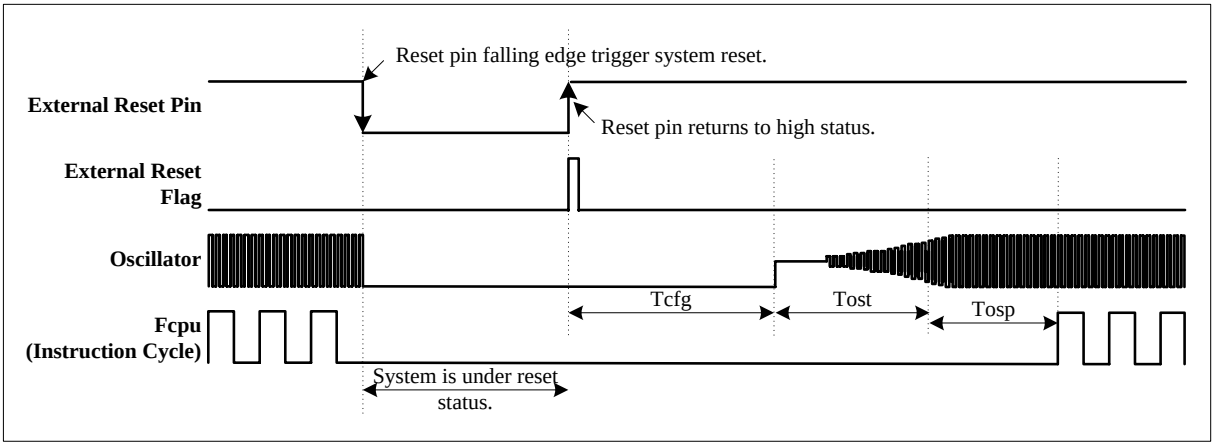
4.8 系统时钟时序

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。通常情况下，低速振荡器的启动时间要比高速振荡器的启动时间短，RC 振荡器的启动时间要比晶体/陶瓷振荡器的启动时间短。	-
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）	64ms @ F_{hosc} = 32KHz 512us @ F_{hosc} = 4MHz 128us @ F_{hosc} = 16MHz
		睡眠模式唤醒情况的振荡器起振时间为： 2048*F_{hosc} ……晶体/陶瓷振荡器，如 32768Hz 晶振，4MHz 晶振，16MHz 晶振等； 32*F_{hosc}……RC 振荡器，如外部 RC 振荡电路，内部高速 RC 振荡器。	X'tal: 64ms @ F_{hosc} = 32KHz 512us @ F_{hosc} = 4MHz 128us @ F_{hosc} = 16MHz RC: 8us @ F_{hosc} = 4MHz 2us @ F_{hosc} = 16MHz

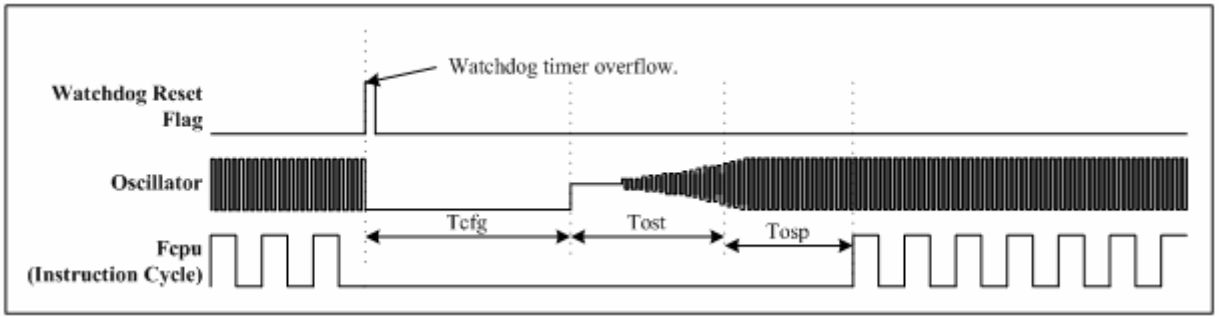
- 上电复位时序



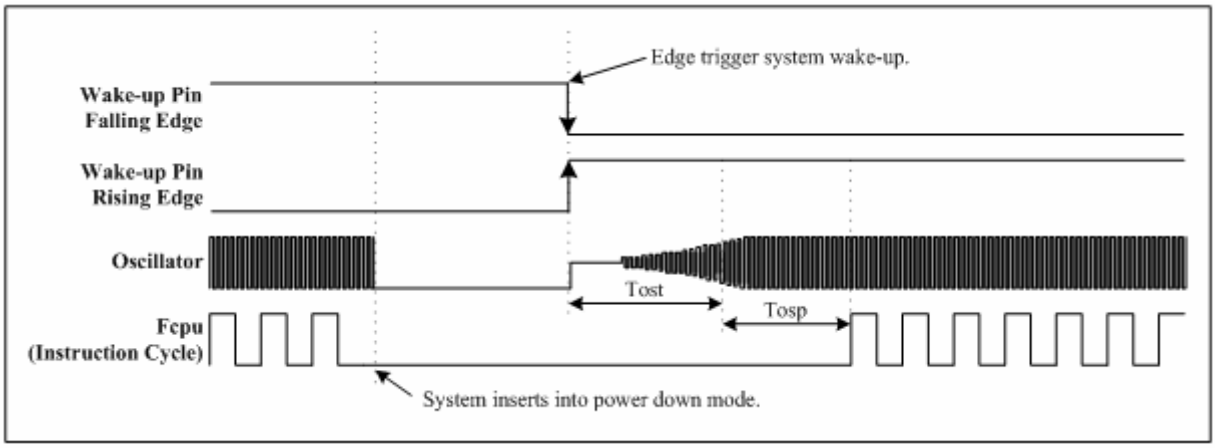
- 外部复位引脚复位时序



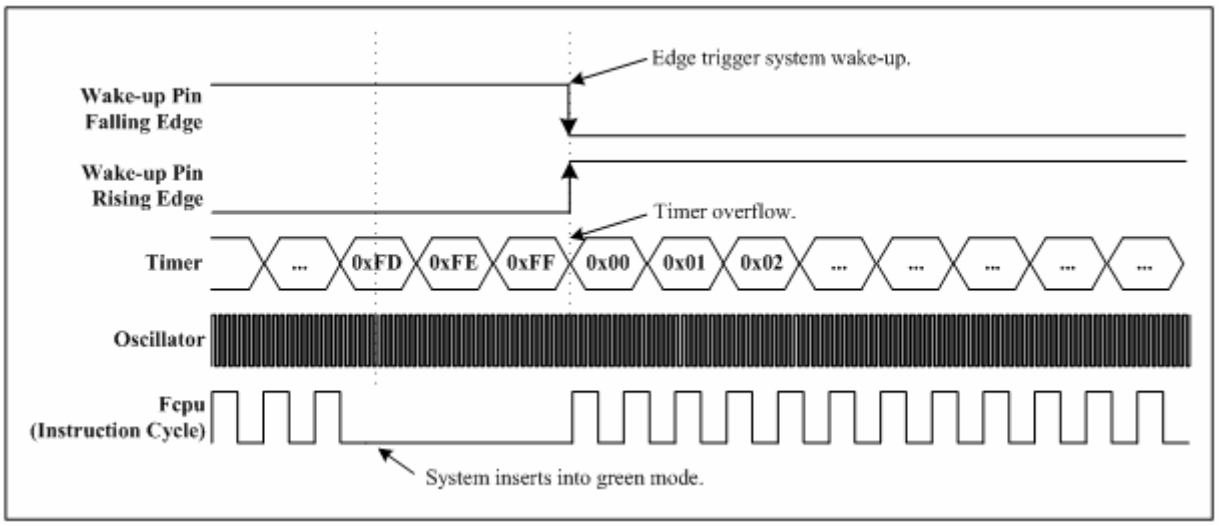
- 看门狗复位时序



- **睡眠模式唤醒时序**

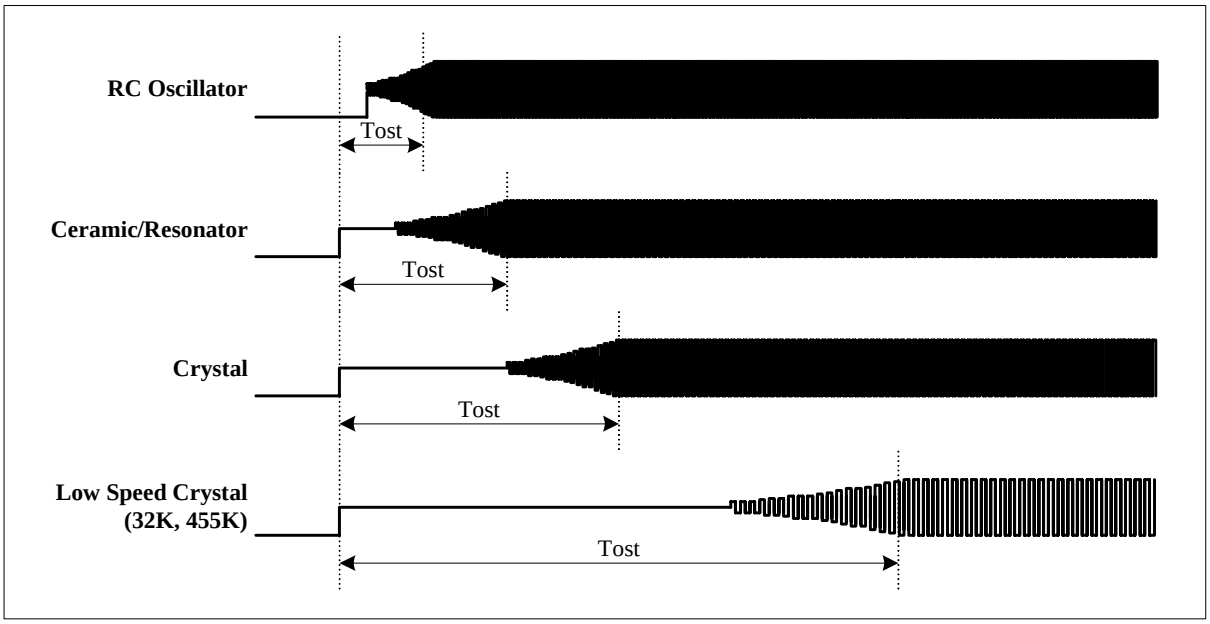


- **绿色模式唤醒时序**



- **振荡器启动时间**

启动时间取决于振荡器的材料、工艺等。通常情况下，低速振荡器的启动时间要比高速振荡器的启动时间短，RC 振荡器的启动时间要比晶体/陶瓷振荡器的启动时间短。



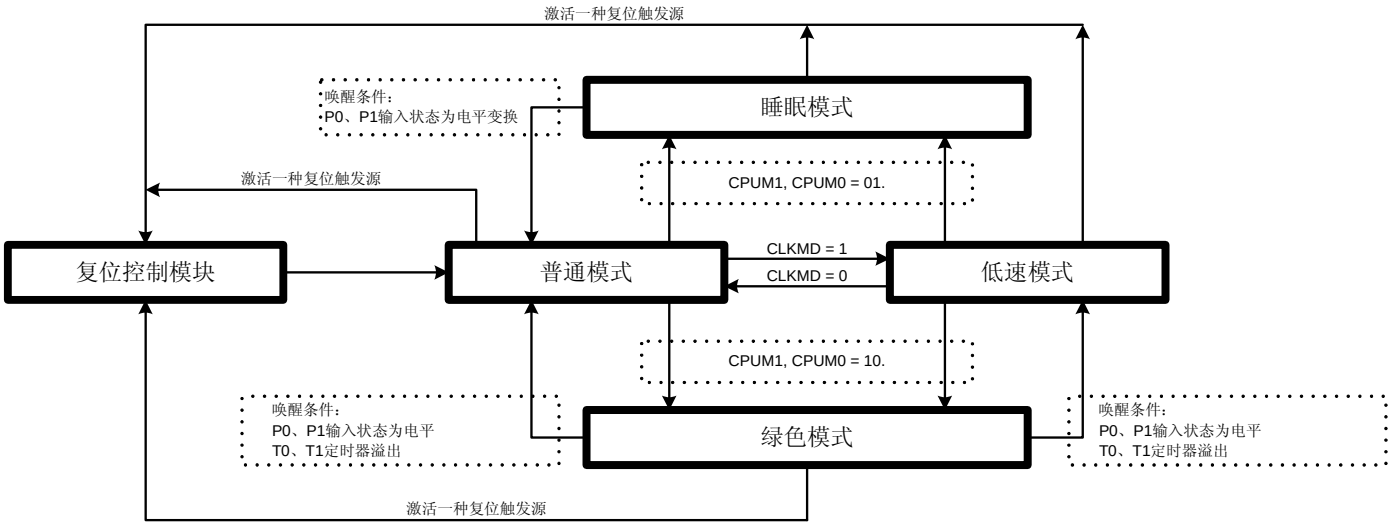
5 系统工作模式

5.1 概述

SN8P2735 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、程序的执行已经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
EHOSC	运行	STPHX	STPHX	停止
IHRC	运行	STPHX	STPHX	停止
ILRC	运行	运行	运行	停止
EHOSC with **RTC**	运行	STPHX	运行	停止
IHRC with **RTC**	运行	STPHX	停止	停止
ILRC with **RTC**	运行	运行	停止	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/Buzzer 有效）	无效
T1 定时器	T1ENB	T1ENB	T1ENB	无效
PWM1, PWM2	PWnEN	PWnEN	PWnEN	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, T1	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, T1, 复位	P0, P1, 复位

- **EHOSC**：外部高速振荡器（XIN/XOUT）。
- **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。
- 高速振荡器和内部低速 RC 振荡器都正常工作。
- 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，包括外部高速振荡器、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，可以确保系统进入睡眠模式，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，系统就在溢出是被唤醒。由 OSCM 寄存器 CPUM1 为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。
```

➢ **例：从普通/低速模式切换进入绿色模式，并使能 T0 的唤醒功能，带有 RTC 功能。**

```
    CLR       T0C               ; 清 T0 计数器。
    B0BSET    FT0TB             ; 使能 T0 RTC 功能。
    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/T1 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/T1 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 2048 个外部高速时钟周期和 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 * 2048（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 * 2048 = 0.512 ms（Fosc = 4MHz）

总的唤醒时间 = 0.512ms + 振荡器启动时间

内部高速 RC 时钟的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 = 2 ms（Fosc = 16MHz）

✷ **注：高速时钟的启动时间决定于 VDD 和振荡器的类型。**

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 都具有唤醒功能，二者区别在于：P0 的唤醒功能始终有效，而 P1 的唤醒功能则由寄存器 P1W 控制。

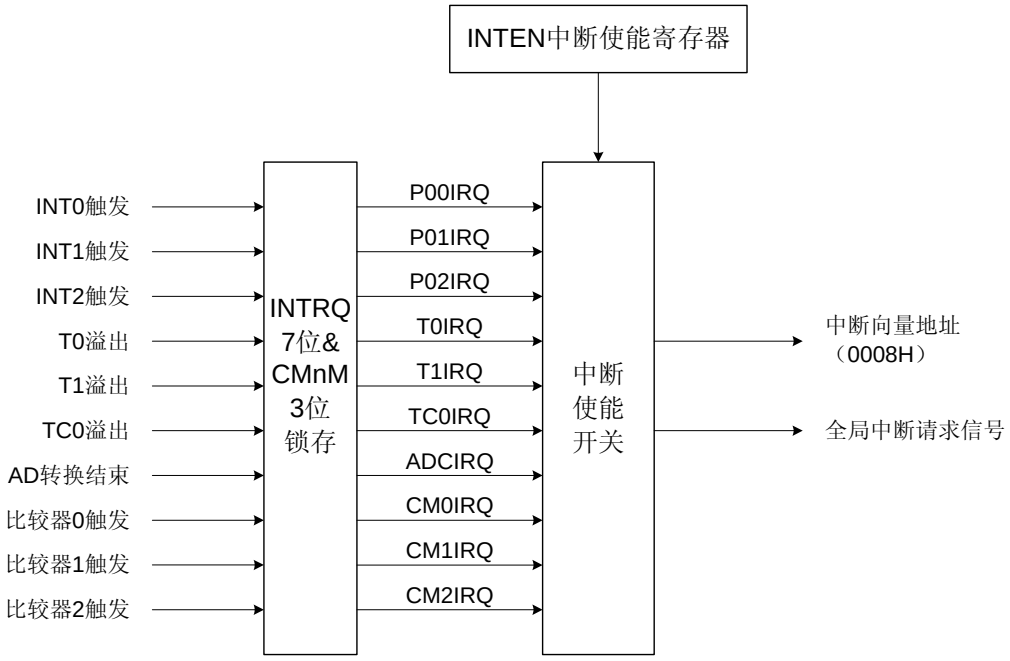
0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **P10W~P17W**：P1 唤醒功能控制位。
0 = 禁止；
1 = 使能。

6 中断

6.1 概述

SN8P2735 提供 10 种中断源：7 个内部中断（T0/T1/TC0/CM0/CM1/CM2/ADC）和 3 个外部中断（INT0/INT1/INT2）。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 中断使能寄存器 INTEN

中断请求控制寄存器 INTEN 包括所有中断的使能控制位。INTEN 的有效位被置为“1”，则系统进入该中断服务程序，程序计数器入栈，程序转至 0008H 即中断程序。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	ADCIEN	T1IEN	TC0IEN	T0IEN	-	P012EN	P01IEN	P00IEN
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 0 **P00IEN**：P0.0 外部中断（INT0）控制位。
0 = 禁止；
1 = 使能。

Bit 1 **P01IEN**：P0.1 外部中断（INT1）控制位。
0 = 禁止；
1 = 使能。

Bit 2 **P02IEN**：P0.2 外部中断（INT2）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 5 **TC0IEN**：TC0 中断控制位。
0 = 禁止；
1 = 使能。

Bit 6 **T1IEN**：T1 中断控制位。
0 = 禁止；
1 = 使能。

Bit 7 **ADCIEN**：ADC 中断控制位。
0 = 禁止；
1 = 使能。

CM0IEN（CM0M’s bit 6）：比较器 0 中断控制位。
0 = 禁止；
1 = 使能。

CM1IEN（CM1M’s bit 6）比较器 1 中断控制位。
0 = 禁止；
1 = 使能。

CM2IEN（CM2M’s bit 6）：比较器 2 中断控制位。
0 = 禁止；
1 = 使能。

6.3 中断请求寄存器 INTRQ

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	ADCIRQ	T1IRQ	TC0IRQ	T0IRQ	-	P02IRQ	P01IRQ	P00IRQ
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 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 **P01IRQ**：P0.1 中断（INT1）请求标志位。
0 = INT1 无中断请求；
1 = INT1 有中断请求。

Bit 2 **P02IRQ**：P0.2 中断（INT2）请求标志位。
0 = INT2 无中断请求；
1 = INT2 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 5 **TC0IRQ**：TC0 中断请求标志位。
0 = TC0 无中断请求；
1 = TC0 有中断请求。

Bit 6 **T1IRQ**：T1 中断请求标志位。
0 = TC1 无中断请求；
1 = TC1 有中断请求。

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

CM0IRQ（CM0M’s bit 5）：比较器 0 中断请求标志位。
0 = CM0 无中断请求；
1 = CM0 有中断请求。

CM1IRQ（CM1M’s bit 5）：比较器 1 中断请求标志位。
0 = CM1 无中断请求；
1 = CM1 有中断请求。

CM2IRQ（CM2M’s bit 5）：比较器 2 中断请求标志位。
0 = CM2 无中断请求；
1 = CM2 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
复位后	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（无 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。**

```
.DATA       ACCBUF      DS 1

.CODE
            ORG         0
            JMP         START

            ORG         8H
            JMP         INT_SERVICE

            ORG         10H
START:
            …

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

            RETI                    ; 退出中断。
            …
            ENDP
```

6.6 外部中断（INT0~INT2）

SN8P2735 共有 3 个外部中断：INT0、INT1、INT2。当外部中断被触发时，若使能外部中断使能控制位，则外部中断请求标志位被置 1。若使能外部中断使能控制位且外部中断请求位置 1 时，程序计数器跳转到中断向量地址 0008H 开始执行中断服务。若禁止外部中断使能控制位，则外部中断请求标志位无效，则不会执行中断服务程序。

外部中断内置唤醒锁存功能，就是指系统从睡眠模式唤醒，唤醒源位中断源（P0.0、P0.1 和 P0.2）。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	-	-	P02G1	P02G0	P01G1	P01G0	P00G1	P00G0
读/写	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	0	0	0	0	0	0

Bit[5:4] **P02G[1:0]**：INT2 中断触发控制位。
00 = 保留；
01 = 上升沿；
10 = 下降沿；
11 = 上升/下降沿（电平变换触发）。

Bit[3:2] **P01G[1:0]**：INT1 中断触发控制位。
00 = 保留；
01 = 上升沿；
10 = 下降沿；
11 = 上升/下降沿（电平变换触发）。

Bit[1:0] **P00G[1:0]**：INT0 中断触发控制位。
00 = 保留；
01 = 上升沿；
10 = 下降沿；
11 = 上升/下降沿（电平变换触发）。

➢ **例：INT0 中断请求设置，电平触发。**

```
        MOV     A, #98H
        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, #64H         ; 初始化 T0C = 64H。
        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, #64H
        B0MOV       T0C, A          ;
        …
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

✷ **注：RTC 模式下，不能在中断下复位 T0C。**

6.8 TC0 中断

TC0C 溢出时，无论 TC0IEN 处于何种状态，TC0IRQ 都会置“1”。若 TC0IEN 和 TC0IRQ 都置“1”，系统就会响应 TC0 的中断；若 TC0IEN = 0，则无论 TC0IRQ 是否置“1”，系统都不会响应 TC0 中断。尤其需要注意多种中断下的情形。

➢ **例：TC0 中断请求设置。**

```
        B0BCLR      FTC0IEN         ; 禁止 TC0 中断。
        B0BCLR      FTC0ENB         ;
        MOV         A, #20H         ;
        B0MOV       TC0M, A         ; TC0 时钟＝Fcpu / 64。
        MOV         A, # 64H        ; TC0C 初始值＝64H。
        B0MOV       TC0C, A         ; TC0 间隔= 10 ms。

        B0BCLR      FTC0IRQ         ; 清 TC0 中断请求标志。
        B0BSET      FTC0IEN         ; 使能 TC0 中断。
        B0BSET      FTC0ENB         ;

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：TC0 中断服务程序。**

```
        ORG         8H              ;
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; 保存 ACC 和 PFLAG。

        B0BTS1      FTC0IRQ         ; 检查是否有 TC0 中断请求标志。
        JMP         EXIT_INT        ; TC0IRQ = 0，退出中断。

        B0BCLR      FTC0IRQ         ; 清 TC0IRQ。
        MOV         A, #64H
        B0MOV       TC0C, A         ; 清 TC0C。
        …                           ; TC0 中断程序。
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

6.9 T1 中断

T1C（T1CH、T1CL）溢出时，无论 T1IEN 处于何种状态，T1IRQ 都会置“1”。若 T1IEN 和 T1IRQ 都置“1”，系统就会响应 T1 的中断；若 T1IEN = 0，则无论 T1IRQ 是否置“1”，系统都不会响应 T1 中断。尤其需要注意多种中断下的情形。

➢ **例：T1 中断请求设置。**

```
        B0BCLR      FT1IEN          ; 禁止 T1 中断。
        B0BCLR      FT1ENB          ; 禁止 T1 定时器。
        MOV         A, #20H         ;
        B0MOV       T1M, A          ; 设置 T1 时钟 = Fcpu / 64，下降沿触发。
        CLR         T1CH
        CLR         T1CL

        B0BCLR      FT1IRQ          ; 清 T1 中断请求。
        B0BSET      FT1IEN          ; 使能 T1 中断。
        B0BSET      FT1ENB          ; 使能 T1 定时器

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：T1 中断服务程序。**

```
        ORG         8H
        JMP         INT_SERVICE
INT_SERVICE:

        PUSH                        ; 保存 ACC 和 PFLAG。

        B0BTS1      FT1IRQ          ; 检查是否有 T1 中断请求标志。
        JMP         EXIT_INT        ; T1IRQ = 0，退出中断。

        B0BCLR      FT1IRQ          ; 清 T1IRQ。
        B0MOV       A, T1CH
        B0MOV       T1CHBUF, A
        B0MOV       A, T1CL
        B0MOV       T1CLBUF, A      ; 保存脉宽。
        CLR         T1CH
        CLR         T1CL
        ...                         ; T1 中断服务程序。
        …
EXIT_INT:
        POP                         ; 恢复 ACC 和 PFLAG。


        RETI                        ; 退出中断。
```

6.10 ADC 中断

当 ADC 转换完成后，无论 ADCIEN 是否使能，ADCIQR 都会置“1”。若 ADCIEN 和 ADCIQR 都置“1”，那么系统就会响应 ADC 中断。若 ADCIEN = 0，不管 ADCIRQ 是否置“1”，系统都不会进入 ADC 中断。用户应注意多种中断下的处理。

➢ **例： ADC 中断设置。**

```
        B0BCLR      FADCIEN         ; 禁止 ADC 中断。

        MOV         A, #10110000B   ;
        B0MOV       ADM, A          ; 允许 P4.0 ADC 输入，使能 ADC 功能。
        MOV         A, #00000000B   ; 设置 AD 转换速率 = Fcpu/16。
        B0MOV       ADR, A

        B0BCLR      FADCIRQ         ; 清除 ADC 中断请求标志。
        B0BSET      FADCIEN         ; 使能 ADC 中断。
        B0BSET      FGIE            ; 使能 GIE。

        B0BSET      FADS            ; 开始 AD 转换。
```

➢ **例：ADC 中断服务程序。**

```
        ORG         8H              ; 中断向量地址。
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; 保存 ACC 和 PFLAG。

        B0BTS1      FADCIRQ         ; 检查是否有 ADC 中断。
        JMP         EXIT_INT        ; ADCIRQ = 0，退出中断。

        B0BCLR      FADCIRQ         ; 清 ADCIRQ。
        …                           ; ADC 中断服务程序。
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

6.11 比较器中断（CMP0~CMP2）

SN8P2735 共有 3 个比较器中断（CMP0~CMP2）。寄存器 CM0M 的 CM0G[1:0]控制中断 CMP0 的中断触发方向，寄存器 CM1M 的 CM1G[1:0]控制中断 CMP1 的中断触发方向，寄存器 CM2M 的 CM2G[1:0]控制中断 CMP2 的中断触发方向。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，比较器振荡器的请求控制位都会置 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。

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM0M	CM0EN	CM0IEN	CM0IRQ	CM0OEN	CM0REF	CM0OUT	CM0G1	CM0G0
读/写	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 6 **CM0IEN**：CMP0 中断控制位。
0 = 禁止；
1 = 使能。

Bit 5 **CM0IRQ**：CMP0 中断请求控制位。
0 = CMP0 无中断请求；
1 = CMP0 有中断请求。

Bit [1:0] **CM0G[1:0]**：CMP0 中断触发方向控制位。
00 = 保留；
01 = 上升沿，CM0P > CM0N 或比较器内部参考电压；
10 = 下降沿，CM0P < CM0N 或比较器内部参考电压；
11 = 上升/下降沿（电平变换触发）。

09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM1M	CM1EN	CM1IEN	CM1IRQ	CM1OEN	CM1REF	CM1OUT	CM1G1	CM1G0
读/写	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 6 **CM1IEN**：CMP1 中断控制位。
0 = 禁止；
1 = 使能。

Bit 5 **CM1IRQ**：CMP1 中断请求控制位。
0 = CMP1 无中断请求；
1 = CMP1 有中断请求。

Bit [1:0] **CM1G[1:0]**：CMP1 中断触发方向控制位。
00 = 保留；
01 = 上升沿，CM1P > CM1N 或比较内部参考电压；
10 = 下降沿，CM1P < CM1N 或比较内部参考电压；
11 = 上升/下降沿（电平变换触发）。

09EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM2M	CM2EN	CM2IEN	CM2IRQ	CM2OEN	CM2REF	CM2OUT	CM2G1	CM2G0
读/写	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 6 **CM2IEN**：CMP2 中断控制位。
0 = 禁止；
1 = 使能。

Bit 5 **CM2IRQ**：CMP2 中断请求控制位。
0 = CMP2 无中断请求；
1 = CMP2 有中断请求。

Bit [1:0] **CM2G[1:0]**：CMP2 中断触发方向控制位。
00 = 保留；
01 = 上升沿，CM2P > CM2N 或比较内部参考电压；
10 = 下降沿，CM2P < CM2N 或比较内部参考电压；
11 = 上升/下降沿（电平变换触发）。

➢ **例：设置 CMP0 中断。**

```
        MOV     A, #03H
        B0MOV   CM0M, A         ; 设置 CMP0 中断为电平变换触发。

        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。
```

6.12 多中断操作

在同一时刻，系统中可能出现多个中断请求。此时，用户必须根据系统的要求对各中断进行优先权的设置。中断请求标志 IRQ 由中断事件触发，当 IRQ 处于有效值“1”时，系统并不一定会响应该中断。各中断触发事件如下表所示：

中断	有效触发
P00IRQ	由 PEDGE 控制
P01IRQ	由 PEDGE 控制
P02IRQ	由 PEDGE 控制
T0IRQ	T0C 溢出
T1IRQ	T1CH、T1CL 溢出
TC0IRQ	TC0C 溢出
ADCIRQ	AD 转换结束
CMP0IRQ	CMP0 输出电平转变
CMP1IRQ	CMP1 输出电平转变
CMP2IRQ	CMP2 输出电平转变

多个中断同时发生时，需要注意的是：首先，必须预先设定好各中断的优先权；其次，利用 IEN 和 IRQ 控制系统是否响应该中断。在程序中，必须对中断控制位和中断请求标志进行检测。

➢ **例：多中断条件下检测中断请求。**

```
        ORG     8H                  ;
        JMP     INT_SERVICE
INT_SERVICE:
        …                           ; 保存 ACC 和 PFLAG。
INTP00CHK:                          ; 检查是否有 INT0 中断请求。
        B0BTS1  FP00IEN             ; 检查是否使能 INT0 中断。
        JMP     INTT0CHK            ; 跳到下一个中断。
        B0BTS0  FP00IRQ             ; 检查是否有 INT0 中断请求。
        JMP     INTP00              ; 进入 INT0 中断。
INTT0CHK:                           ; 检查是否有 T0 中断请求。
        B0BTS1  FT0IEN              ; 检查是否使能 T0 中断。
        JMP     INTTC0CHK           ; 跳到下一个中断。
        B0BTS0  FT0IRQ              ; 检查是否有 T0 中断请求。
        JMP     INTT0               ; 进入 T0 中断。
INTTC0CHK:                          ; 检查是否有 TC0 中断请求。
        B0BTS1  FTC0IEN             ; 检查是否使能 TC0 中断。
        JMP     INTADCCHK           ; 跳到下一个中断。
        B0BTS0  FTC0IRQ             ; 检查是否有 TC0 中断请求。
        JMP     INTTC0              ; 进入 TC0 中断。
INTADCHK:                           ; 检查是否有 ADC 中断请求。
        B0BTS1  FADCIEN             ; 检查是否使能 ADC 中断。
        JMP     INT_EXIT            ;
        B0BTS0  FADCIRQ             ; 检查是否有 ADC 中断请求。
        JMP     INTADC              ; 进入 ADC 中断。
INT_EXIT:
        …                           ; 恢复 ACC 和 PFLAG。
        RETI                        ; 退出中断。
```

7 I/O 端口

7.1 概述

SN8P2735 共有 30 个 I/O 引脚，大多数 I/O 引脚与模拟引脚和特殊功能的引脚共用，详见下表：

I/O 引脚		共用引脚		引脚共用条件
名称	类型	名称	类型	
P0.0	I/O	INT0	DC	P00IEN=1
		PWM1T	DC	PW1EN=1，PW1GEN=1，PW1GS=0
P0.1	I/O	INT1	DC	P01IEN=1
		PWM1	DC	PW1EN=1.
P0.2	I/O	INT2	DC	P02IEN=1
		PWM1N	DC	PW1EN=1，PW1NEN=1
P0.3	I	RST	DC	Reset_Pin 编译选项为 Reset
		VPP	HV	OTP 烧录
P0.4	I/O	XOUT	AC	High_CLK 编译选项为 IHRC_RTC、32K、4M、12M
P0.5	I/O	XIN	AC	High_CLK 编译选项为 IHRC_RTC、RC、32K、4M、12M
P1.0	I/O	CM2N	AC	CM2EN=1
		OP2N	AC	OP2EN=1
P1.1	I/O	CM2P	AC	CM2EN=1，CM2REF=0
		OP2P	AC	OP2EN=1
P1.2	I/O	CM2O	AC	CM2EN=1，CM2OEN=1
		OP2O	AC	OP2EN=1
P1.3	I/O	CM1N	AC	CM1EN=1
		OP1N	AC	OP1EN=1
P1.4	I/O	CM1P	AC	CM1EN=1，CM1REF=0
		OP1P	AC	OP1EN=1
P1.5	I/O	CM1O	AC	CM1EN=1，CM1OEN=1
		OP1O	AC	OP1EN=1
P1.6	I/O	CM0N	AC	CM0EN=1
		OP0N	AC	OP0EN=1
P1.7	I/O	CM0P	AC	CM0EN=1，CM0REF=0
		OP0P	AC	OP0EN=1
P5.0	I/O	CM0O	AC	CM0EN=1，CM0OEN=1
		OP0O	AC	OP0EN=1
P5.1	I/O	PWM21	DC	PW2EN=1，PW2CH1=1
P5.2	I/O	PWM22	DC	PW2EN=1，PW2CH2=1
P5.3	I/O	PWM23	DC	PW2EN=1，PW2CH3=1
P5.4	I/O	BZ0/PWM0	DC	TC0ENB=1，TC0OUT=1 或者 PWM0OUT=1
P5.5	I/O	PWM24	DC	PW2EN=1，PW2CH4=1
P5.6	I/O	PWM25	DC	PW2EN=1，PW2CH5=1
P5.7	I/O	PWM26	DC	PW2EN=1，PW2CH6=1
P4.0	I/O	AIN0	AC	ADENB=1，GCHS=1，CHS[2:1] = 000b
		AVREFH	AC	ADENB=1，AVREFH=1
P4[7:1]	I/O	AIN[7:1]	AC	ADENB=1，GCHS=1，CHS[2:1] = 001b~111b

*** 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	-	-	P05M	P04M	-	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

0C4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4M	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M
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：P0.3 是单向输入引脚，P0M.3 未定义。**

➢ **例：I/O 模式选择。**

```
CLR     P0M         ; 设置为输入模式。
CLR     P4M
CLR     P5M

MOV     A, #0FFH    ; 设置为输出模式。
B0MOV   P0M, A
B0MOV   P4M,A
B0MOV   P5M, A

B0BCLR  P4M.0       ; P4.0 设为输入模式。

B0BSET  P4M.0       ; P4.0 设为输出模式。
```

7.3 I/O 口上拉电阻

输入模式下内置上拉电阻。PnUR 寄存器可以控制上拉电阻，当 PnUR 为 0 时，禁止上拉电阻，为 1 时使能上拉电阻。

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	-	-	P05R	P04R	-	P02R	P01R	P00R
读/写	-	-	W	W	-	W	W	W
复位后	-	-	0	0	-	0	0	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

0E4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4UR	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R
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：P0.3 为单向输入引脚，无上拉电阻，故 P0UR.3 未定义。**

➢ **例：I/O 口的上拉电阻。**

```
MOV     A, #0FFH        ; 使能 P0、P4、P5 的上拉电阻。
B0MOV   P0UR, A         ;
B0MOV   P4UR,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/W	R/W	R	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

0D4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4	P47	P46	P45	P44	P43	P42	P41	P40
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，P03 保持为 1。**

➢ **例：读取输入口的数据。**

```
B0MOV   A, P0       ; 读取 P0、P4 和 P5 口的数据。
B0MOV   A, P4
B0MOV   A, P5
```

➢ **例：写入数据到输出端口。**

```
MOV     A, #0FFH    ; 写入数据 FFH 到 P0、P4 和 P5。
B0MOV   P0, A
B0MOV   P4, A
B0MOV   P5, A
```

➢ **例：写入 1 位数据到输出端口。**

```
B0BSET  P4.0        ; P4.0 和 P5.3 置 1。
B0BSET  P5.3

B0BCLR  P4.0        ; P4.0 和 P5.3 清 0。
B0BCLR  P5.3
```

7.5 P4 与 ADC 共用引脚

P4 口和 ADC 的输入口共用，无施密特触发。同一时间只能设置 P4 口的一个引脚作为 ADC 的测量信号输入口（通过 ADM 寄存器来设置），其它引脚则作为普通 I/O 使用。具体应用中，当输入一个模拟信号到 CMOS 结构端口，尤其当模拟信号为 1/2 VDD 时，将可能产生额外的漏电流。同样，当 P4 口外接多个模拟信号时，也会产生额外的漏电流。在睡眠模式下，上述漏电流会严重影响到系统的整体功耗。P4CON 为 P4 口的配置寄存器。将 P4CON[7:0]置“1”，其对应的 P4 口将被设置为纯模拟信号输入口，从而避免上述漏电流的情况。

0AEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
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] **P4CON[7:0]**：P4.n 控制位。
0 = P4.n 作为模拟信号输入引脚（ADC 输入引脚）或普通 I/O 引脚；
1 = P4.n 只能作为模拟信号输入引脚，不能作为普通 I/O 引脚。

✷ 注：当 P4.n 作为普通 I/O 口而不是 ADC 输入引脚时，P4CON.n 必须置为 0，否则 P4.n 的普通 I/O 信号会被隔离开来。

P4 的 ADC 模拟输入由寄存器 ADM 的 GCHS 和 CHSn 位控制，若 GCHS = 0，P4.n 为普通的 I/O 引脚，若 GCHS = 1，CHSn 所对应的 P4.n 用作 ADC 模拟信号输入引脚。

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	AVREFH	CHS2	CHS1	CHS0
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 4 **GCHS**：ADC 输入通道控制位。
0 = 禁止 AIN 通道；
1 = 开启 AIN 通道。

Bit 3 **AVREFH**：ADC 外部参考电压高电平输入引脚控制位。
0 = ADC 参考电压由内部 VDD 提供，P4.0 作为 GPIO 或 AIN0 引脚；
1 = 使能 ADC 外部参考电压高电平由 P4.0 输入。

Bit[2:0] **CHS[2:0]**：ADC 输入通道选择位。
000 = AIN0； 001 = AIN1； …； 110 = AIN6；111 = AIN7。

✷ 注：在设置 P4.n 为普通的 I/O 引脚时，必须保证 P4.n 的 ADC 功能已经被禁止。否则当 GCHS=1 时，会将 CHS[2:0]对应的 P4.n 自动设为 ADC 模拟输入引脚。

➢ **例：设置 P4.1 为普通的输入引脚，P4CON.1 必须置为 0。**

```
; 检查 GCHS 和 CHS[2:0]的状态。
        B0BCLR      FGCHS       ; 若 CHS[2:0]指向 P4.1（CHS[2:0] = 001B），GCHS=0。
                                ; 若 CHS[2:0]没有指向 P4.1，则忽略 GCHS 的状态。
; 清 P4CON。
        B0BCLR      P4CON.1     ; 使能 P4.1 的普通 I/O 功能。
; P4.1 设为输入模式。
        B0BCLR      P4M.1       ; 设置 P4.1 为输入模式。
```

➢ **例：设置 P4.1 为普通的输出模式，P4CON.1 必须置为 0。**

```
; 检查 GCHS 和 CHS[2:0]的状态。
        B0BCLR      FGCHS       ; 若 CHS[2:0]指向 P4.1（CHS[2:0] = 001B），GCHS=0。
                                ; 若 CHS[2:0]没有指向 P4.1，则忽略 GCHS 的状态。
; 清 P4CON。
        B0BCLR      P4CON.1     ; 使能 P4.1 的普通 I/O 功能。
; 设置 P4.1 为输出模式以避免误操作。
        B0BSET      P4.1        ; 设置 P4.1 为 1。
; 或
        B0BCLR      P4.1        ; 设置 P4.1 为 0。
; P4.1 设为输出模式。
        B0BSET      P4M.1       ; 设置 P4.1 为输出模式。
```

P4.0 与 ADC 输入引脚（AIN0）和 ADC 外部参考电压高电平输入引脚共用。ADM 寄存器的 AVREFH 位为 ADC 参考电压高电平输入的控制位。若使能 AVREFH，禁止 P4.0 的普通 I/O 功能和 ADC 模拟输入功能（AIN0），P4.0 直接连接 ADC 参考电压高电平输入。

✷ **注：若需使能 P4.0 的普通 I/O 功能和 AIN0 功能，则 AVREFH 必须置 0。**

➢ **例：设置 P4.0 为普通输入引脚，AVREFH 和 P4CON.0 必须置 0。**

```
; 检查 AVREFH 的状态。
        B0BTS0      FAVREFH     ;
        B0BCLR      FAVREFH     ; AVREFH = 1，清 AVREFH，禁止外部 ADC 参考电压高电平输入。
                                ; AVREFH = 0，跳到下一条程序。

; 检查 GCHS 和 CHS[2:0]的状态。
        B0BCLR      FGCHS       ; 若 CHS[2:0]指向 P4.0（CHS[2:0]=000B），GCHS=0.
                                ; 若 CHS[2:0]没有指向 P4.0，则忽略 GCHS 的状态。

; 清 P4CON。
        B0BCLR      P4CON.0     ; 使能 P4.0 的普通 I/O 功能。

; 设置 P4.0 为输入模式。
        B0BCLR      P4M.0
```

➢ **例：设置 P4.0 为普通输出引脚，EVHENB 和 P4CON.0 必须置 0。**

```
; 检查 AVREFH 的状态。
        B0BTS0      FAVREFH     ;
        B0BCLR      FAVREFH     ; AVREFH = 1，清 AVREFH，禁止外部 ADC 参考电压高电平输入。
                                ; AVREFH = 0，跳到下一条程序。

; 检查 GCHS 和 CHS[2:0]的状态。
        B0BCLR      FGCHS       ; 若 CHS[2:0]指向 P4.0（CHS[2:0]=000B），GCHS=0.
                                ; 若 CHS[2:0]没有指向 P4.0，则忽略 GCHS 的状态。

; 清 P4CON。
        B0BCLR      P4CON.0     ; 使能 P4.0 的普通 I/O 功能。

; 设置 P4.0 为输出模式以避免误操作。
        B0BSET      P4.0        ; 设置 P4.0 为 1。
; 或
        B0BCLR      P4.0        ; 设置 P4.0 为 0。
; P4.0 设为输出模式。
        B0BSET      P4M.0
```

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。**

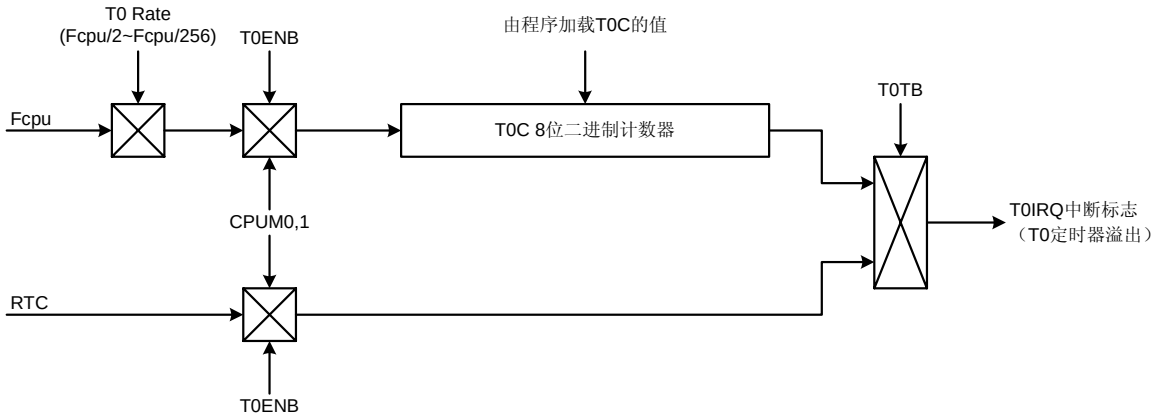
```
Main:
        …                       ; 检查 I/O 口的状态。
        …                       ; 检查 RAM 的内容。
Err:    JMP $                   ; I/O 口或 RAM 出错，不清看门狗等看门狗计时溢出。

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：支持中断请求标志的显示（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，支持 RTC 功能，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。

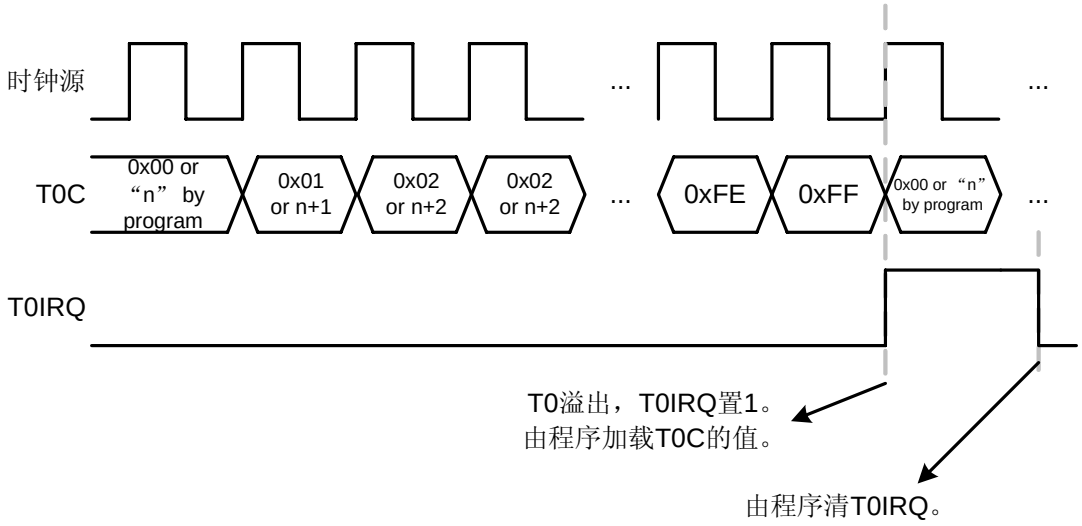
- **8 位可编程计数定时器：**根据选择的时钟频率周期性的产生中断请求。
- **中断功能：**T0 定时器支持中断功能，当 T0 溢出，T0IRQ 有效，程序计数器跳到中断向量地址执行中断。
- **RTC 功能：**T0 支持 RTC 功能，RTC 时钟源由外部低速 32K 振荡时钟提供，此时 T0TB=1。**RTC 功能只能在 High_Clk 选项选择 IHRC_RTC 时才有效。**
- **绿色模式唤醒功能：**T0 定时器在绿色模式下正常工作，溢出时将系统从绿色模式下唤醒。



✷ **注：RTC 模式下，T0 的间隔时间固定为 0.5S，T0C 计数 256 次。**

8.2.2 T0 定时器操作

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 时钟	T0 间隔时间					
		Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4		IHRC_RTC 模式	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)	max. (sec)	Unit (ms)
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	-	-
-	32768Hz/64	-	-	-	-	0.5	1.953

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	-	-	-	T0TB
读/写	R/W	R/W	R/W	R/W	-	-	-	R/W
复位后	0	0	0	0	-	-	-	0

Bit 0 **T0TB**：RTC 时钟源控制位。
0 = 禁止 RTC（T0 时钟源由 Fcpu 提供）；
1 = 使能 RTC。

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。

✷ **注：RTC 模式下 T0RATE 无效，T0 的间隔时间固定为 0.5S。**

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。

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 * 输入时钟）

➢ **例：T0 的中断间隔时间为 10ms，高速时钟为内部 4MHz，Fcpu = Fosc/4=4MHz/4=1MHz，T0RATE = 001（Fcpu/128）。**

T0C 初始值 = 256 -（T0 中断间隔时间 * 输入时钟）
= 256 -（10ms * 4MHz / 4 / 128）
= 256 -（10^{-2} * 4 * 10^{6} / 4 /128）
= B2H

✷ **注：RTC 模式下，T0C 计数 256 次，产生 0.5S 的中断间隔时间，不能在 RTC 模式下更改 T0C 的值。**

8.2.5 T0 定时器操作举例

- **T0 定时器设置**

```
; 复位 T0 定时器。
        MOV     A, #0x00        ; 清 T0M。
        B0MOV   T0M, A

; 设置 T0 时钟源和 T0Rate。
        MOV     A, #0nnn00000b
        B0MOV   T0M, A

; 设置 T0 中断间隔时间。
        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 中断功能。
        B0BSET  FT0ENB          ; 使能 T0 定时器。
```

- **T0 在 RTC 模式下工作**

```
; 复位 T0 定时器。
        MOV     A, #0x00        ; 清 T0M。
        B0MOV   T0M, A

; 设置 T0 RTC 功能。
        B0BSET  FT0TB

; 清 T0C。
        CLR     T0C

; 清 T0IRQ。
        B0BCLR  FT0IRQ

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

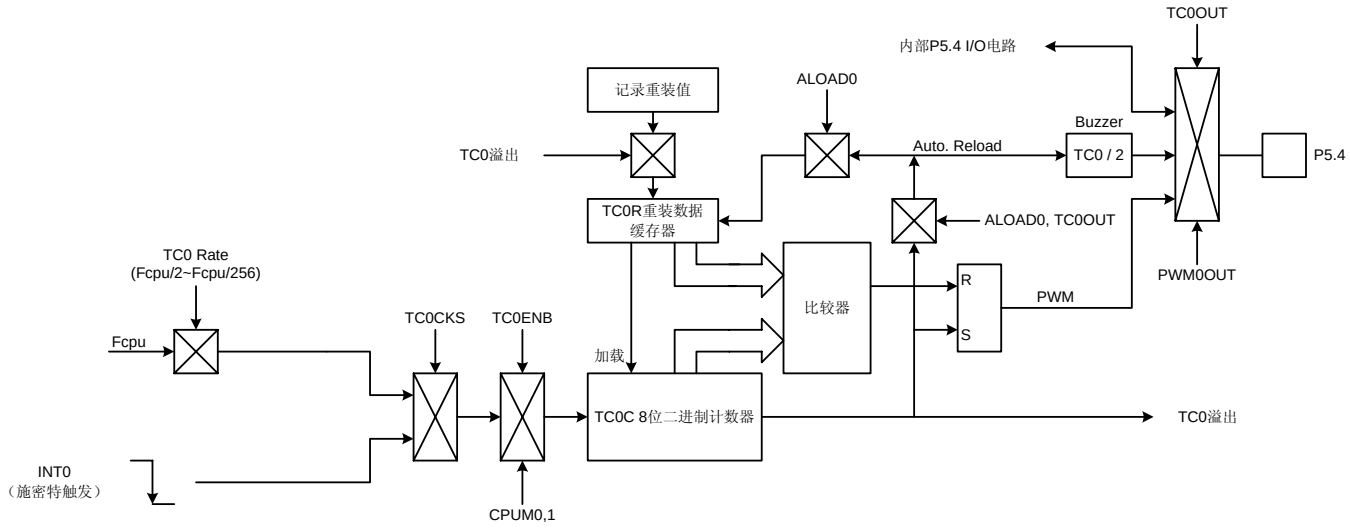
8.3 8 位定时/计数器 TC0

8.3.1 概述

8 位二进制定时/计数器具有基本定时器、事件计数器、Buzzer 和 PWM 功能。基本定时器功能可以支持中断请求标志的显示（TC0IRQ）和中断操作（中断向量）。由 TC0M、TC0C、TC0R 寄存器控制 TC0 的中断间隔时间。事件计数器可以将 TC0 时钟源由系统时钟更改为外部时钟信号（如连续的脉冲、R/C 振荡信号等）。TC0 作为计数器时记录外部时钟数目以进行测量应用。TC0 还内置 Buzzer 和 PWM 功能，其周期/分辨率由 TC0 定时器时钟频率和 TC0R 寄存器控制，故具有良好性能的 Buzzer 和 PWM 可以处理 IR 进位信号、马达控制和光度调节等。

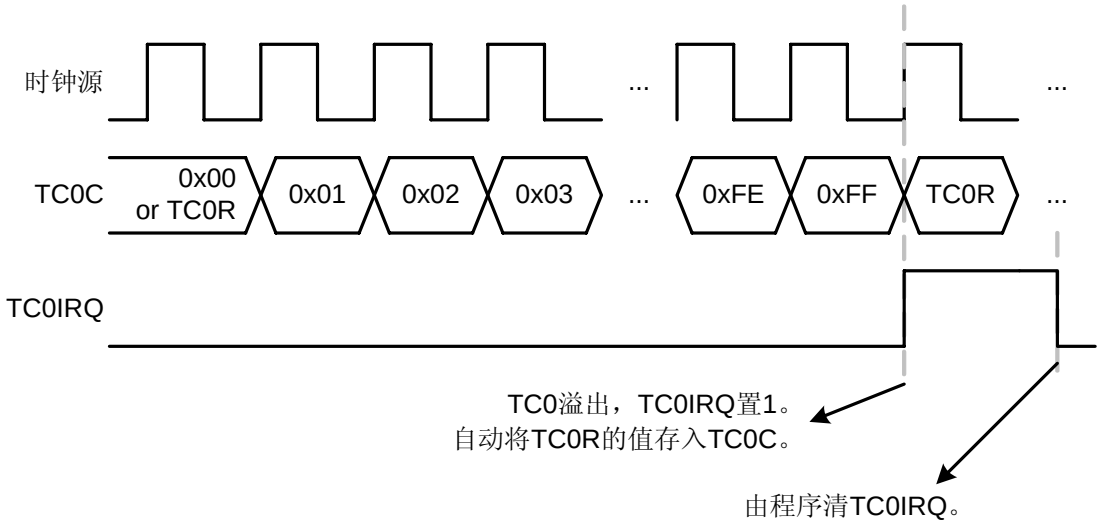
TC0 的主要通途如下：

- **8 位可编程定时器：**根据选择的时钟信号，产生周期中断；
- **中断功能：**TC0 定时器支持中断，当 TC0 溢出时，TC0IRQ 置 1，系统执行中断；
- **外部事件计数器：**对外部事件计数；
- **PWM 输出：**由 TC0rate 和 TC0R 寄存器控制占空比/周期；
- **Buzzer 输出：**Buzzer 的输出信号为 TC0 中断间隔时间的 1/2；
- **绿色模式功能：**绿色模式下，TC0 正常工作，但无唤醒功能。



8.3.2 TC0 定时器操作

TC0 定时器由 TC0ENB 控制。当 TC0ENB=0 时，TC0 停止工作；当 TC0ENB=1 时，TC0 开始计数。使能 TC0 之前，先要设定好 TC0 的功能模式，如基本定时器、TC0 中断等。TC0C 溢出（从 0FFH 到 00H）时，TC0IRQ 置 1 以显示溢出状态并由程序清零。在不同的功能模式下， TC0C 不同的值对应不同的操作，若改变 TC0C 的值影响到操作，会导致功能出错。TC0 内置双重缓存器以避免此种状况的发生。在 TC0C 计数的过程中不断的刷新 TC0C，保证将最新的值存入 TC0R（重装缓存器）中，当 TC0 溢出后，TC0R 的值由自动存入 TC0。进入下一个周期后，TC0 进行新的工作状态。定时/计数器模式时，自动重装功能由 ALOAD0 位控制，PWM 模式下自动使能自动重装功能并使能 TC0。如果使能 TC0 中断功能（TC0IEN=1），在 TC0 溢出时系统执行中断服务程序，在中断时必须由程序清 TC0IRQ。TC0 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，TC0 虽继续工作，但不能唤醒系统。



TC0 根据不同的时钟源选择不同的应用模式，TC0 的时钟源由 Fcpu（指令周期）和外部引脚输入（P0.0）提供，由 TC0CKS 控制。当 TC0CKS=0 时，TC0 时钟源来自 Fcpu，可以由 TC0Rate[2:0]选择不同的分频。当 TC0CKS=1 时，TC0 时钟源来自外部引脚输入，此时使能外部事件计数功能。

TC0rate[2:0]	TC0 时钟分频	TC0 中断间隔时间			
		Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
		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

8.3.3 TC0M 模式寄存器

模式寄存器 TC0M 控制 TC0 的工作模式。

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT
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 0 **PWM0OUT**：PWM 输出控制位。
0 = 禁止 PWM 输出，P5.4 为普通 I/O 引脚；
1 = 允许 PWM 输出，P5.4 输出 PWM 信号，PWM 的输出占空比由 TC0OUT 和 ALOAD0 控制。

Bit 1 **TC0OUT**：TC0 溢出信号输出控制位，仅当 PWM0OUT = 0 时有效。
0 = 禁止，P5.4 作为普通的 I/O 引脚；
1 = 使能，P5.4 输出 TC0OUT 信号。

Bit 2 **ALOAD0**：自动装载控制位，仅当 PWM0OUT = 0 时有效。
0 = 禁止；
1 = 使能。

Bit 3 **TC0CKS**：TC0 时钟信号控制位。
0 = 内部时钟 Fcpu；
1 = 外部时钟信号，使能事件计数器功能，此时 TC0Rate[2:0]无效。

Bit [6:4] **TC0RATE[2:0]**：TC0 分频选择位。
000 = fcpu/256.
001 = fcpu/128.
…
110 = fcpu/4.
111 = fcpu/2.

Bit 7 **TC0ENB**：TC0 启动控制位。
0 = 关闭 TC0 定时器；
1 = 开启 TC0 定时器。

✷ **注：若 TC0CKS=1，TC0 则用作外部事件计数器，此时不需考虑 TC0RATE 的设置。P0.0 无中断请求（P00IRQ＝0）。**

8.3.4 TC0C 计数寄存器

8 位计数器 TC0C 溢出时，TC0IRQ 置 1 并由程序清零，用来控制 TC0 的中断间隔时间。首先须写入正确的值到 TC0C 和 TC0R 寄存器，并使能 TC0 定时器以保证第一个周期正确。TC0 溢出后，TC0R 的值自动装入 TC0C。

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

TC0C 初始值的计算公式如下：

TC0C 初始值 = N -（TC0 中断间隔时间 * 输入时钟）

N 是 TC0 最大溢出值。TC0 的溢出时间和有效值见下表：

TC0CKS	PWM0	ALOAD0	TC0OUT	N	TC0R 有效值	TC0R 二进制有效范围	注释
0	0	x	x	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出
	1	0	0	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出
	1	0	1	64	00H~3FH	xx000000b~xx111111b	每计数 64 次溢出
	1	1	0	32	00H~1FH	xxx00000b~xxx11111b	每计数 32 次溢出
	1	1	1	16	00H~0FH	xxxx0000b~xxxx1111b	每计数 16 次溢出
1	-	-	-	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出

8.3.5 TC0R 自动重装寄存器

TC0 的自动重装功能由 TC0M 的 ALOAD0 位控制。当 TC0C 溢出时，TC0R 的值自动装入 TC0C 中。这样，用户在使用的过程中就不需要在中断中复位 TC0C。

TC0 为双重缓存器结构。若程序对 TC0R 进行了修改，那么修改后的 TC0R 值首先被暂存在 TC0R 的第一个缓存器中，TC0 溢出后，TC0R 的新值就会被存入 TC0R 缓存器中，从而避免 TC0 中断时间出错以及 PWM 和蜂鸣器误动作。

0CDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

TC0R 初始值计算公式如下：

TC0R 初始值 = 256 -（TC0 中断间隔时间 * 输入时钟）

TC0 的溢出时间和有效值见下表：

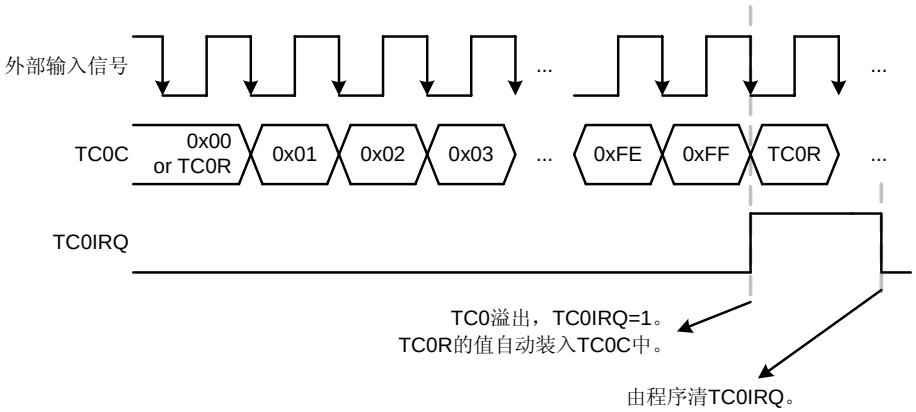
TC0CKS	PWM0	ALOAD0	TC0OUT	N	TC0R 有效值	TC0R 二进制有效范围
0	0	x	x	256	00H~0FFH	00000000b~11111111b
	1	0	0	256	00H~0FFH	00000000b~11111111b
	1	0	1	64	00H~3FH	xx000000b~xx111111b
	1	1	0	32	00H~1FH	xxx00000b~xxx11111b
	1	1	1	16	00H~0FH	xxxx0000b~xxxx1111b
	0	x	x	256	00H~0FFH	00000000b~11111111b
	1	0	0	256	00H~0FFH	00000000b~11111111b
	1	0	1	64	00H~3FH	xx000000b~xx111111b
	1	1	0	32	00H~1FH	xxx00000b~xxx11111b
	1	1	1	16	00H~0FH	xxxx0000b~xxxx1111b
1	-	-	-	256	00H~0FFH	00000000b~11111111b

➢ **例：TC0 的间隔时间为 10ms，时钟源来自 Fcpu，高速时钟为外部 4MHz，Fcpu=Fosc/4=4MHz/4=1MHz，TC0RATE=001（Fcpu/128）。**

TC0R = 256 -（TC0 中断间隔时间 * 输入时钟）
= 256 -（10ms * 4MHz / 4 / 128）
= 256 -（10^{-2} * 4 * 10^{6} / 4 / 128）
= B2H

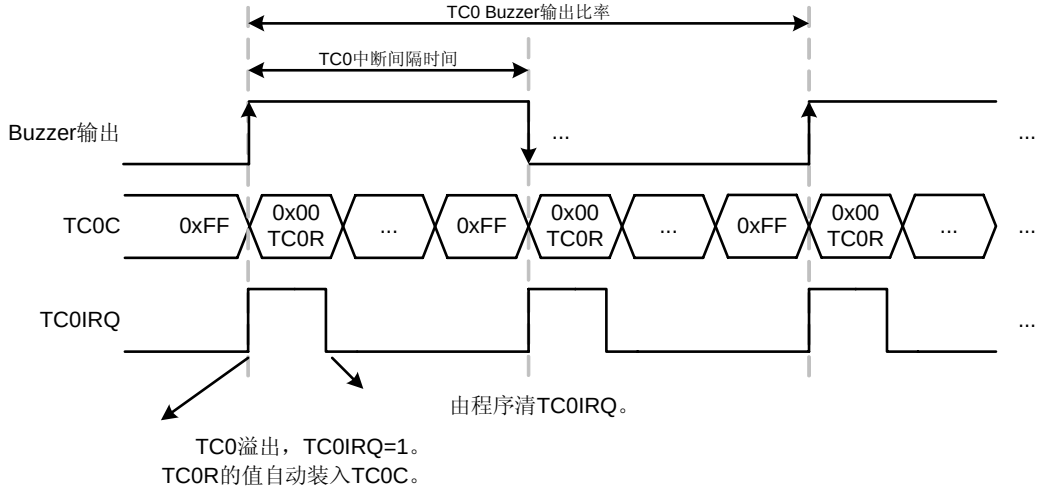
8.3.6 TC0 事件计数器

TC0 作为外部事件计数器时，其时钟源由外部输入引脚（P0.0）提供。当 TC0CKS1=1 时，TC0 的时钟源由外部输入引脚（P0.0）提供，下降沿触发。TC0C 溢出（从 FFH 到 00H）时，使能外部事件计数功能，同时禁止外部输入引脚的唤醒功能以避免外部事件的触发信号将系统唤醒而耗电。此时，P0.0 的外部中断功能也被禁止，即 P00IRQ=0。外部事件计数器通常用来测量外部连续信号的比率，如连续的脉冲信号，R/C 振荡信号等，外部信号的相位与 MCU 时钟的相位并不同步。



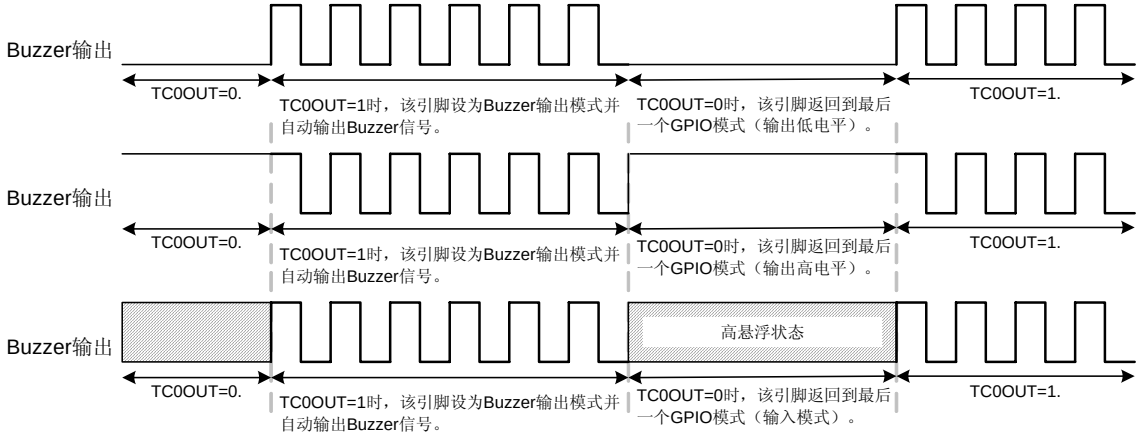
8.3.7 TC0 BUZZER 输出

Buzzer 输出是一个简单的 1/2 占空比信号输出，由 TC0 产生。当 TC0 溢出时，Buzzer 开始输出一个方波，中断间隔时间频率 2 分频后作为 Buzzer 输出的频率。Buzzer 输出的波形图如下所示：



TC0 溢出后，Buzzer 输出时，TC0IRQ 有效，且当 TC0IEN=1 时，使能 TC0 中断功能。但强烈建议小心同时使用 Buzzer 和 TC0 定时器，以确保两种功能都能正常工作。

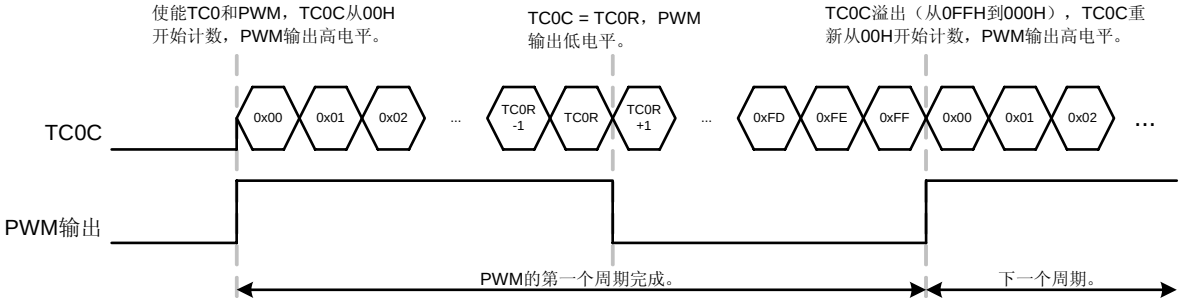
Buzzer 输出引脚与 GPIO 引脚共用，TC0OUT=1 时，该引脚自动设为 Buzzer 输出引脚。如清 TC0OUT 位以禁止 Buzzer 输出后，该引脚自动返回到最后一个 GPIO 模式。



✷ **注：PWM 模式下，由于是 TC0OUT 决定 PWM 的周期，故 Buzzer 输出时，PWM0OUT 必须置 0。**

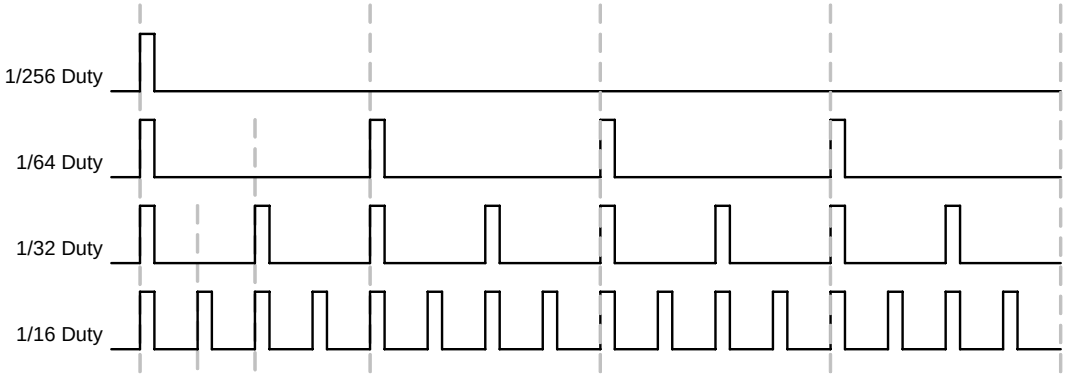
8.3.8 脉宽调制（PWM）

可编程控制占空比/周期的 PWM 可以提供不同的 PWM 信号。使能 TC0 定时器且 PWM0OUT=1 时，由 PWM 输出引脚（P5.4）输出 PWM 信号。PWM 首先输出高电平，然后输出低电平。TC0Rate[2:0]控制 PWM 的周期，ALOAD0 和 TC0OUT 决定 PWM 的分辨率，TC0R 控制 PWM 的占空比（脉冲高电平的长度）。TC0C 初始化后，使能 TC0 定时器，TC0 溢出。当 TC0C=TC0R 时，PWM 输出低电平；TC0 溢出时（TC0C 的值从 0FFH 到 00H），整个 PWM 周期完成，并进入下一个周期。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。



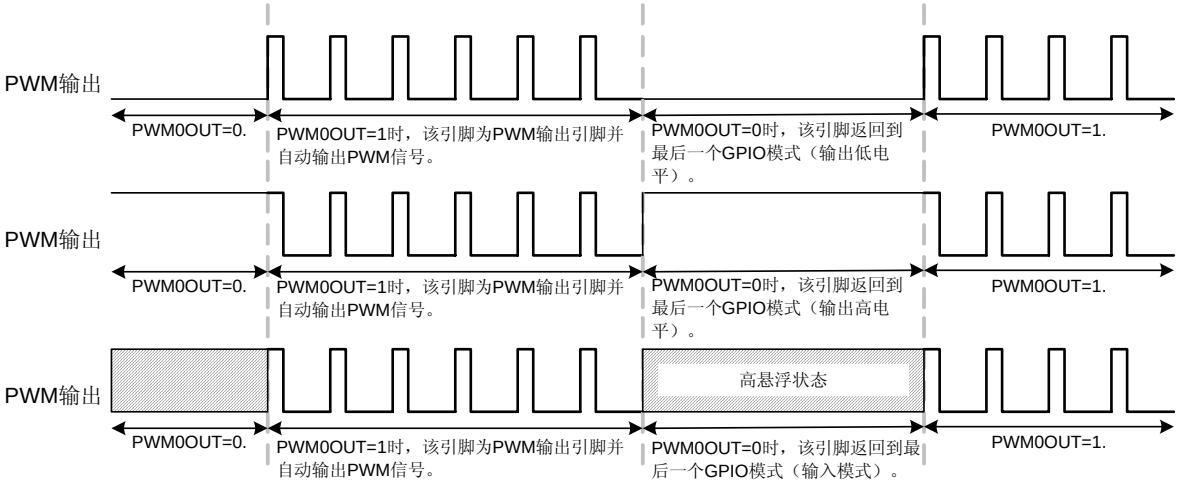
PWM 的分辨率包括 1/256、1/64、1/32、1/16，由 ALOAD0 和 TC0OUT 控制。当 ALOAD0、TC0OUT=00 时，PWM 分辨率为 1/256；ALOAD0、TC0OUT=01 时，PWM 分辨率为 1/64；ALOAD0、TC0OUT=10 时，PWM 分辨率为 1/32；ALOAD0、TC0OUT=11 时，PWM 分辨率为 1/16。调整 PWM 的分辨率时，TC0R PWM 的占空比也要跟着跳转以匹配新的 PWM 分辨率。TC0 溢出后，PWM 输出时，TC0IRQ 有效，且当 TC0IEN=1 时，使能 TC0 中断功能。但强烈建议小心同时使用 PWM 和 TC0 定时器，以确保两种功能都能正常工作。

ALOAD0	TC0OUT	PWM 分辨率	TC0R 有效值	TC0R 有效值（二进制）
0	0	256	00H~0FFH	00000000b~11111111b
0	1	64	00H~3FH	xx000000b~xx111111b
1	0	32	00H~1FH	xxx00000b~xxx11111b



1	1	16	00H~0FH	xxxx0000b~xxxx1111b

PWM 输出引脚和 GPIO 引脚共用，PWM0OUT=1 时，该引脚输出 PWM 信号。如果清 PWM0OUT 位以禁止 PWM 时，该引脚返回到最后一个 GPIO 模式。



8.3.9 TC0 定时器操作举例

- **TC0 定时器：**

```
; 复位 TC0。
        MOV         A, #0x00            ; 清 TC0M。
        B0MOV       TC0M, A

; 设置 TC0Rate 和自动重装功能。
        MOV         A, #0nnn0000b
        B0MOV       TC0M, A
        B0BSET      FALOAD0

; 设置 TC0C 和 TC0R 寄存器以获得 TC0 中断间隔时间。
        MOV         A, #value           ; TC0C 必须等于 TC0R。
        B0MOV       TC0C, A
        B0MOV       TC0R, A

; 清 TC0IRQ。
        B0BCLR      FTC0IRQ

; 使能 TC0 定时器和中断功能。
        B0BSET      FTC0IEN             ; 使能 TC0 中断功能。
        B0BSET      FTC0ENB             ; 使能 TC0 定时器。
```

- **TC0 事件计数器：**

```
; 复位 TC0。
        MOV         A, #0x00            ; 清 TC0M。
        B0MOV       TC0M, A

; 设置 TC0 自动重装功能。
        B0BSET      FALOAD0

; 使能 TC0 事件计数器。
        B0BSET      FTC0CKS             ; 设置 TC0 时钟源为外部输入引脚（P0.0）。

; 设置 TC0C 和 TC0R 寄存器以获得 TC0 中断间隔事件。
        MOV         A, #value           ; TC0C 必须等于 TC0R。
        B0MOV       TC0C, A
        B0MOV       TC0R, A

; 清 TC0IRQ。
        B0BCLR      FTC0IRQ

; 使能 TC0 定时器和中断功能。
        B0BSET      FTC0IEN             ; 使能 TC0 中断功能。
        B0BSET      FTC0ENB             ; 使能 TC0 定时器。
```

- **TC0 BUZZER 输出：**

```
; 复位 TC0。
        MOV         A, #0x00            ; 清 TC0M。
        B0MOV       TC0M, A

; 设置 TC0Rate 和自动重装功能。
        MOV         A, #0nnn0000b
        B0MOV       TC0M, A
        B0BSET      FALOAD0

; 设置 TC0C 和 TC0R 寄存器以获得 TC0 中断间隔事件。
        MOV         A, #value           ; TC0C 必须等于 TC0R。
        B0MOV       TC0C, A
        B0MOV       TC0R, A

; 使能 TC0 定时器和 Buzzer 输出功能。
        B0BSET      FTC0ENB             ; 使能 TC0 定时器。
        B0BSET      FTC0OUT             ; 使能 TC0 Buzzer 输出功能。
```

- **TC0 PWM：**

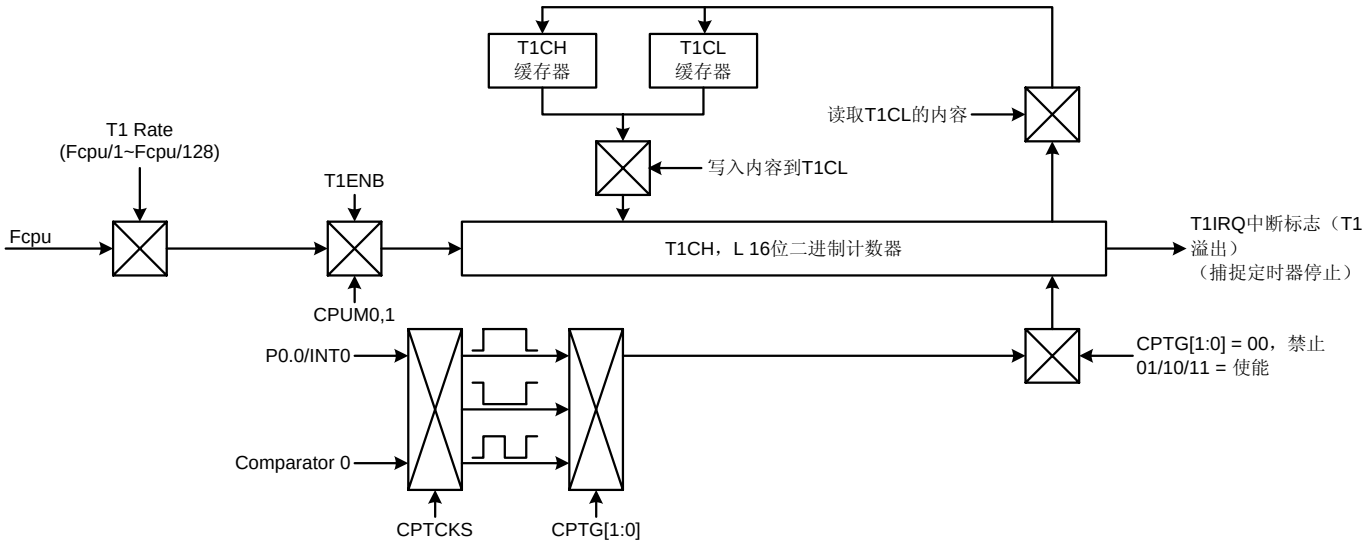
```
; 复位 TC0。
        MOV         A, #0x00            ; 清 TC0M。
        B0MOV       TC0M, A

; Set TC0Rate 以获取 PWM 周期。
        MOV         A, #0nnn0000b
        B0MOV       TC0M, A

; 设置 PWM 分辨率。
        MOV         A, #00000nn0b
        OR          TC0M, A

; 设置 TC0R 寄存器以获取 PWM 占空比。
        MOV         A, #value
        B0MOV       TC0R, A

; 清 TC0C。
        CLR         TC0C

; 使能 PWM 和 TC0 定时器。
        B0BSET      FTC0ENB             ; 使能 TC0 定时器。
        B0BSET      FPWM0OUT            ; 使能 PWM。
```

8.4 16 位定时/计数器 T1

8.4.1 概述

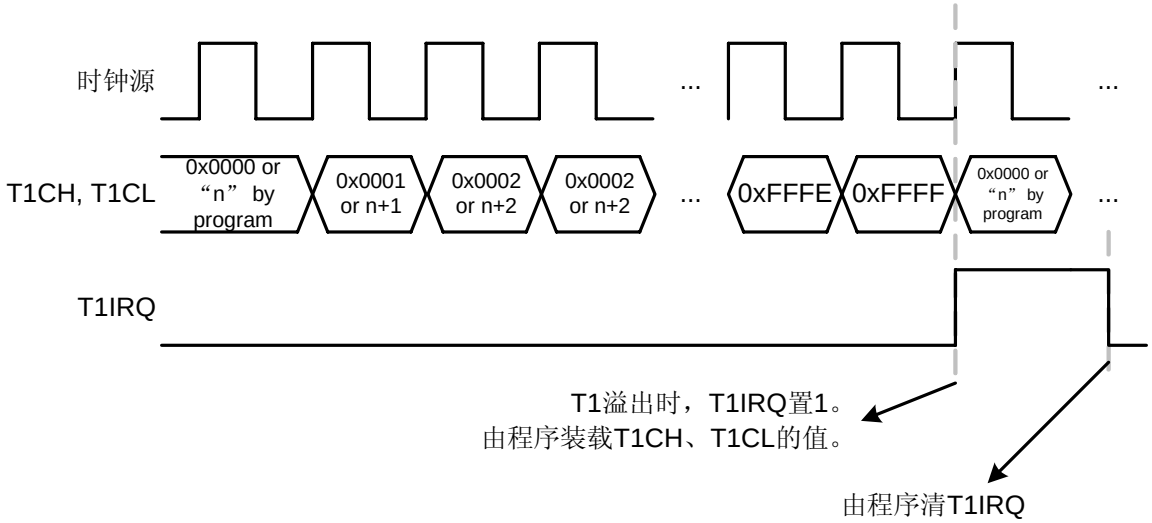
16 位二进制定时器 T1 具有基本定时器和捕捉定时器功能：基本定时器支持中断请求标志的显示（T1IRQ）和中断操作（中断向量），中断间隔时间由 T1M、T1CH/T1CL 计数寄存器控制；捕捉定时器支持脉冲高电平宽度测量、低电平测量和周期测量（由 P0.0 提供）以及比较器的输出信号（如连续的脉冲、R/C 振荡信号等）。T1 作为定时器使用时用来记录外部信号时间参数以进行测量应用。T1 的主要功能如下所示：

- **16 位可编程的计数定时器：**根据选择的时钟频率周期性的产生中断请求。
- **中断功能：**T1 定时器支持中断功能。当 T1 溢出，T1IRQ 有效，程序计数器跳到中断向量地址执行中断。
- **16 位捕捉定时器：**测量输入信号的脉冲宽度和周期，由根据决定捕捉定时器的分辨率的 T1 时钟决定。T1 内置可编程的触发边沿选择装置以决定开始-停止触发事件。
- **绿色模式功能：**T1 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。



8.4.2 T1 定时器操作

T1 定时器由 T1ENB 控制。当 T1ENB=0 时，T1 停止工作；当 T1ENB=1 时，T1 开始计数。使能 T1 之前，先设置 T1 的功能模式，如基本定时器、中断功能等。16 位计数器 T1（T1CH、T1CL）溢出（从 0FFFFH 到 0000H）时，T1IRQ 置 1 显示溢出状态并由程序清零，并由程序装载 T1CH、T1CL 的值以确定合适的中断间隔时间。若使能 T1 中断（T1IEN=1），T1 溢出时，程序计数器跳到中断向量地址开始执行中断服务程序，在中断下必须由程序清 T1IRQ。T1 可以在普通模式、低速模式和绿色模式下工作，在绿色模式下，T1 溢出时 T1IRQ 置 1，将系统唤醒。



T1 的时钟源为 Fcpu（指令周期），由 T1Rate[2:0]决定。详见下表：

T1rate[2:0]	T1 时钟	T1 中断间隔时间			
		Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)
000b	Fcpu/128	2097.152	32	8388.608	128
001b	Fcpu/64	1048.576	16	4194.304	64
010b	Fcpu/32	524.288	8	2097.152	32
011b	Fcpu/16	262.144	4	1048.576	16
100b	Fcpu/8	131.072	2	524.288	8
101b	Fcpu/4	65.536	1	262.144	4
110b	Fcpu/2	32.768	0.5	131.072	2
111b	Fcpu/1	16.384	0.25	65.536	1

8.4.3 T1M 模式寄存器

模式寄存器 T1M 设置 T1 的工作模式，包括 T1 前置分频器、时钟源和捕捉参数等，这些设置必须在使能 T1 定时器之前完成。

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1M	T1ENB	T1rate2	T1rate1	T1rate0	CPTCKS	CPTStart	CPTG1	CPTG0
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 [1:0] **CPTG[1:0]**：T1 捕捉定时器功能控制位。
00 = 禁止捕捉定时器功能；
01 = 测量脉冲高电平宽度；
10 = 测量脉冲低电平宽度；
11 = 测量周期。

Bit 2 **CPTStart**：T1 捕捉定时器操作控制位。
0 = 操作结束；
1 = 开始计数。

Bit 3 **CPTCKS**：T1 捕捉定时器输入源选择位。
0 = 外部器输入引脚（P0.0/INT0），使能捕捉定时器功能；
1 = 比较器输出引脚，使能捕捉定时器功能。

Bit [6:4] **T1RATE[2:0]**：T1 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 **T1ENB**：T1 启动控制位。
0 = 禁止；
1 = 使能。

8.4.4 T1CH, T1CL 16 位计数寄存器

16 位计数器 T1CH，T1CL 溢出时，T1IRQ 置 1 并由程序清零，用来控制 T1 的中断间隔时间。

0A1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CL	T1CL7	T1CL6	T1CL5	T1CL4	T1CL3	T1CL2	T1CL1	T1CL0
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

0A2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T1CH	T1CH7	T1CH6	T1CH5	T1CH4	T1CH3	T1CH2	T1CH1	T1CH0
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

16 位定时计数器 T1 为双重缓存器设计，但核心总线只有 8 位，故处理 16 位数据时须锁存标志，以免瞬间状态影响到 16 位数据而出错。写入数据时，写 T1CL 是锁存控制标志；读取数据时，读 T1CL 是锁存控制标志。故写入数据到 T1 时，要先写入数据到 T1CH，再写入数据到 T1CL，即执行写入数据到 T1CL 时，所有数据都已经写入 16 位计数器 T1 中。反之，读取 T1 的数据时，要先读取 T1CL 的数据，再读取 T1CH 的数据，即执行读取 T1CH 时，T1 中的所有数据都已经被读取完毕。

- **读取 T1 计数缓存器中的数据时，要先读取 T1CL 中的数据，再读取 T1CH 中的数据。**
- **写入数据到 T1 计数缓存器中时，要先写入数据到 T1CH，再写入数据到 T1CL。**

16 位计数器 T1（T1CH，T1CL）初始值的计算公式如下：

T1CH，T1 初始值 = 65536 -（T1 中断间隔时间 * T1 时钟比率 T1Rate）

- **例：通过计算 T1CH 和 T1CL 的值，获得 T0 的中断间隔时间为 500ms，T1 时钟源为 Fcpu =16MHz/16=1MHz，T1RATE = 000（Fcpu/128）。**

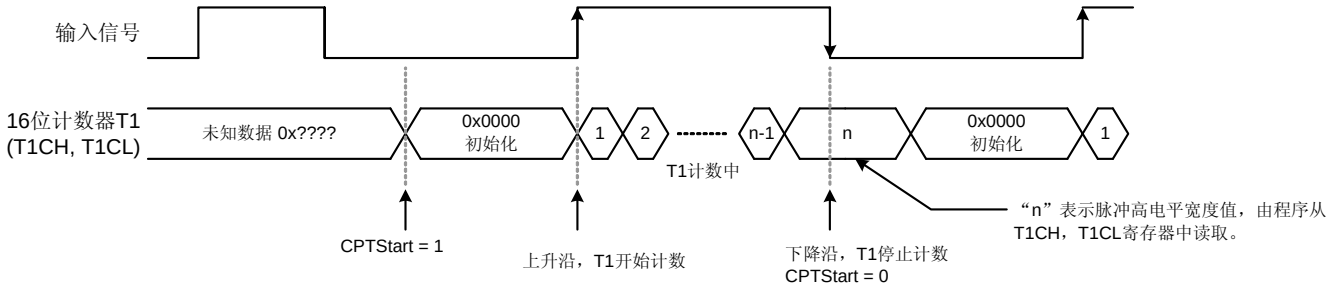
T1 中断间隔时间=500ms，T1Rate=16MHz/16/128

T1CH、T1CL 初始值 = 65536 -（T1 中断间隔时间 * 输入时钟）
= 65536 -（500ms * 16MHz / 16 / 128）
= 65536 -（$500*10^{-3} * 16 * 10^{6} / 16 / 128$）
= F0BDH（**T1CH = F0H，T1CL = BDH**）

8.4.5 T1 捕捉定时器

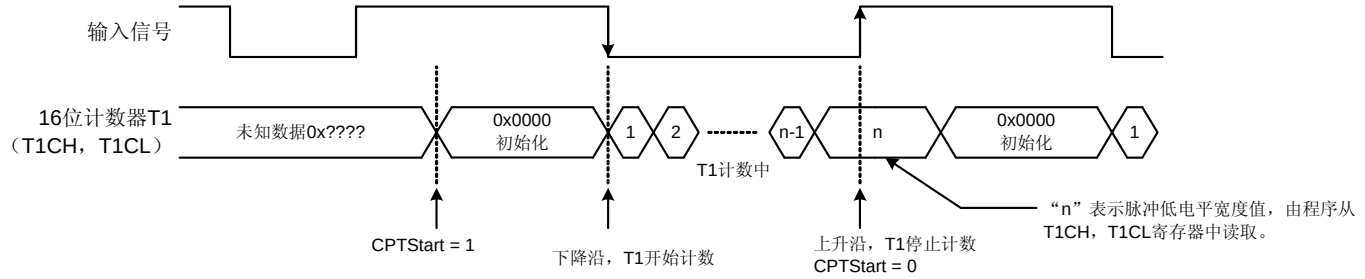
16 位捕捉定时器用来测量输入信号脉冲宽度和周期。当满足触发条件时，T1 开始和停止计数，捕捉定时器功能由 CPTG[1:0]位控制，当 CPTG[1:0]=00 时，禁止捕捉定时器功能；当 CPTG[1:0]=01/10/11 时，使能捕捉定时器功能，但必须要先使能 T1ENB。捕捉定时器可以测量输入脉冲的高电平宽度，输入脉冲的低电平宽度和输入脉冲的周期，由 CPTG[1:0]决定测量内容：当 CPTG[1:0]=01 时，测量输入脉冲的高电平宽度，CPTG[1:0]=10 时，测量输入脉冲低电平宽度，CPTG[1:0]=11 时，测量输入脉冲的周期。CPTG[1:0]只能选择捕捉定时器功能，而不能执行该功能。CPTStart 位是该功能的执行控制位，CPTStart=1 时，捕捉定时器等待合适的触发沿，开始工作；等到第二个合适的触发沿到来时，捕捉定时器停止工作，此时 CPTStart 位被清零，T1IRQ 被置 1。在设置 CPTStart 位之前，16 位计数器 T1 被清零以自动初始化捕捉定时器。

- **测量脉冲高电平宽度（T1ENB = 1，CPTG[1:0] = 01）：**



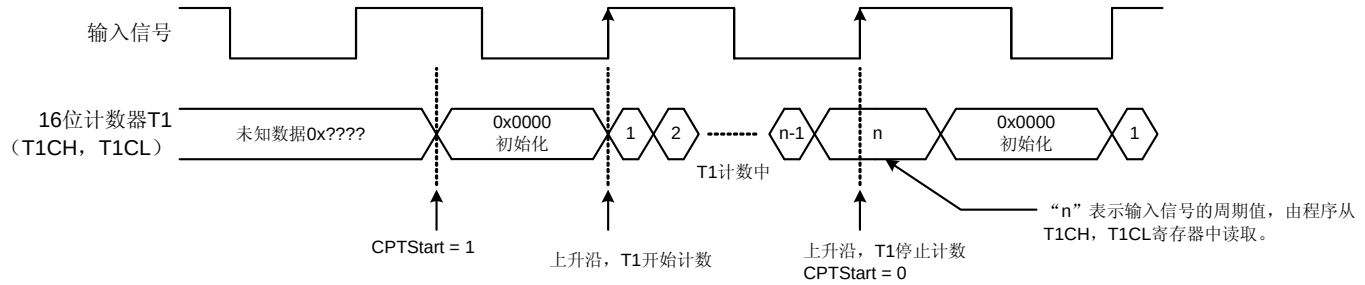
测量脉冲高电平宽度：上升沿时，T1 开始计数；下降沿时，T1 停止计数。如果在高电平期间设置 CPTStart 位，捕捉定时器会在下个上升沿时重新测量高电平宽度。在下降沿时 T1 停止计数后，16 位计数器 T1CH 和 T1CL 寄存器中的值则为脉冲高电平的宽度值。

- **测量脉冲低电平宽度（T1ENB = 1，CPTG[1:0] = 10）：**



测量脉冲低电平宽度：下降沿时，T1 开始计数；上升沿时，T1 停止计数。如果在低电平期间设置 CPTStart 位，捕捉定时器会在下个下降沿时重新测量低电平宽度。在上升沿时 T1 停止计数后，16 位计数器 T1CH 和 T1CL 寄存器中的值则为脉冲低电平的宽度值。

- **测量输入信号的周期（T1ENB = 1，CPTG[1:0] = 11）：**



测量输入信号的周期：上升沿时，T1 开始计数；下一个上升沿时，T1 停止计数。如果在高电平和低电平期间设置 CPTStart 位，捕捉定时器会在下个上升沿时重新开始测量。在上升沿时 T1 停止计数后，16 位计数器 T1CH 和 T1CL 寄存器中的值则为脉冲的周期值。

8.4.6 T1 定时器操作举例

- **T1 定时器：**

```
; 复位 T1。
        MOV         A, #0x00            ; 清 T1M。
        B0MOV       T1M, A

; 设置 T1Rate。
        MOV         A, #0nnn0000b
        B0MOV       T1M, A

; 设置 T1CH，T1CL 寄存器获取 T1 中断间隔时间。
        MOV         A, #value1          ; 先设置高字节。
        B0MOV       T1CH, A
        MOV         A, #value2          ; 设置低字节。
        B0MOV       T1CL, A

; 清 T1IRQ。
        B0BCLR      FT1IRQ

; 使能 T1 定时器和中断功能。
        B0BSET      FT1IEN              ; 使能 T1 中断功能。
        B0BSET      FT1ENB              ; 使能 T1 定时器。
```

- **T1 捕捉定时器，测量信号周期。**

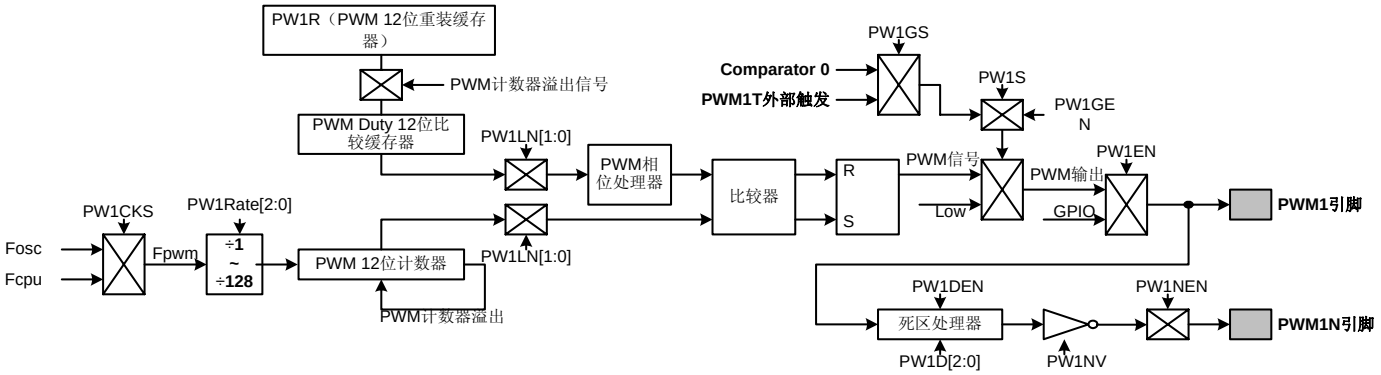
```
; 复位 T1。
        MOV         A, #0x00            ; 清 T1M。
        B0MOV       T1M, A

; 设置 T1Rate，选择输入源，选择/使能 T1 捕捉定时器功能。
        MOV         A, #0nnn00mmb       ; “nnn”代表 T1rate[2:0]。
        B0MOV       T1M, A              ; “mm”代表 CPTG[1:0]。
                                        ; CPTG[1:0] = 00b，禁止 T1 捕捉定时器功能。
                                        ; CPTG[1:0] = 01b，测量脉冲的高电平宽度。
                                        ; CPTG[1:0] = 10b，测量脉冲的低电平宽度。
                                        ; CPTG[1:0] = 11b，测量脉冲的周期。

; 选择 T1 捕捉源。
        B0BCLR      FCPTCKS             ; 捕捉源为 P0.0.。
; or
        B0BSET      FCPTCKS             ; 捕捉源为比较器输出。

; 清 T1CH，T1CL。
        CLR         T1CH                ; 先清除高字节。
        CLR         T1CL                ; 再清除低字节。

; 清 T1IRQ。
        B0BCLR      FT1IRQ

; 使能 T1 定时器和中断功能。
        B0BSET      FT1IEN              ; 使能 T1 中断功能。
        B0BSET      FT1ENB              ; 使能 T1 定时器。

; 设置捕捉定时器开始位。
        B0BSET      FCPTStart
```

9 多功能脉冲宽度调制（PWM1）

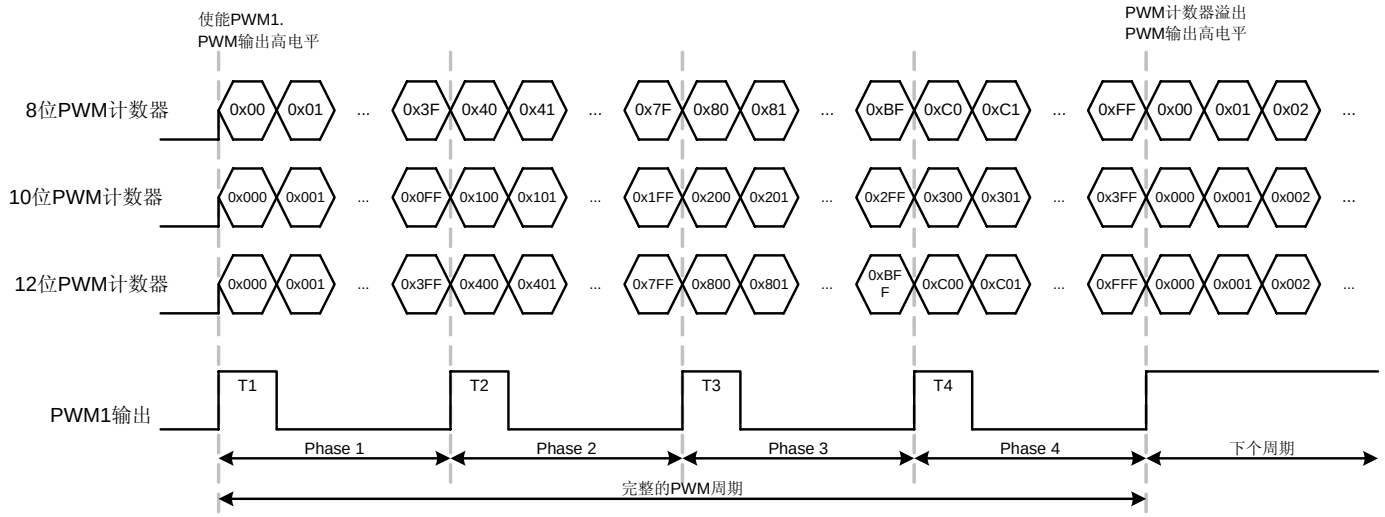
9.1 概述

多功能脉冲宽度调制（PWM1）是高性能设计，包括可编程的高分辨率，可编程的占空比/周期，高频，同步触发功能和反向输出功能。PWM 分辨率有 8 位，10 位和 12 位，PWM 周期由 PWM 时钟源、PWM 时钟 Rate 选择选项和 PWM 分辨率决定。PWM 时钟源由 Fcpu 和 Fhosc 提供，PWM 时钟 Rate 包括 Fosc/1~Fosc/128 和 Fcpu/1~Fcpu/128。PWM 利用新工艺完善 PWM 高频，减少纹波对输出信号的影响。反向输出功能输出一个完整的正常 PWM 信号的反向波形，且内置一个可编程的死区功能。同步触发功能触发 PWM 输出，在死区的反向 PWM 输出和同步触发功能可以实现 AC 零交叉处理应用。PWM 的输出引脚和 GPIO 共用。PWM 的主要功能如下：

- **高速 PWM。**
- **8/10/12 位可编程的分辨率。**
- **多种时钟源，包括 Fosc，Fcpu，Fpwm/1~Fpwm/128。**
- **可编程的死区反向输出功能。**
- **可编程的死区输出功能。**
- **同步触发功能控制 PWM 输出，触发源由比较器或者外部引脚提供，触发沿可编程控制，包括上升沿，下降沿和电平变换。**



9.2 PWM1 COMMON 操作



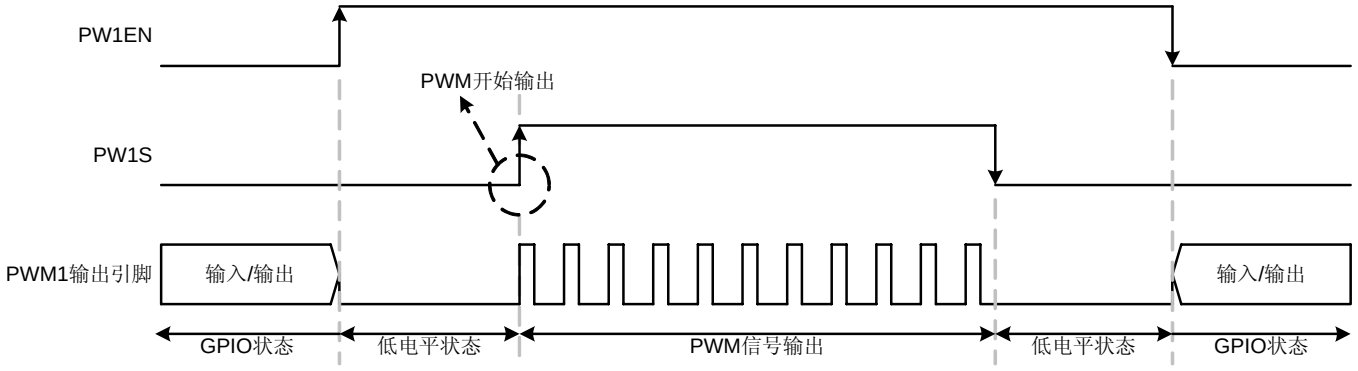
PWM 为 4 相设计，PWM 的一个周期包括 4 个小周期，PWM 信号保持最初的周期参数，但频率快一些，PWM 的频率与小周期的频率相等。上面 PWM 的周期为“（T1+T2+T3+T4）/（PWM 周期）”，4 相占空比的输出顺序是一个特殊的设计和更多的平衡。

PWM 的周期由 PWM 时钟源和 PWM 设置 Rate 选项控制，PW1CKS 位选择 PWM 时钟源（F_{PWM}）来自 Fosc 或者 Fcpu，PW1Rate[2:0]位选择 PWM 时钟 Rate，可以在 F_{PWM}/1~F_{PWM}/128 之间选择。

PWM 重装缓存器（PW1RH，PW1RL）决定 PWM 的占空比，通过 PWM 相位处理器来分配每相的占空比。PWM 具有自动装载功能，在 PWM 输出过程中，通过程序更改 PWM 的占空比，则在下一个周期时会产生新的占空比，而不是立即改变，这样可以避免出现过渡期的占空比。

PWM 分辨率包括 8 位，10 位和 12 位，由 PW1LN[1:0]控制，PW1LN[1:0]=00 时选择 8 位 PWM，PW1LN[1:0]=01 时选择 10 位 PWM，PW1LN[1:0]=10 时选择 12 位 PWM。根据实际需要选择合适的 PWM 分辨率。

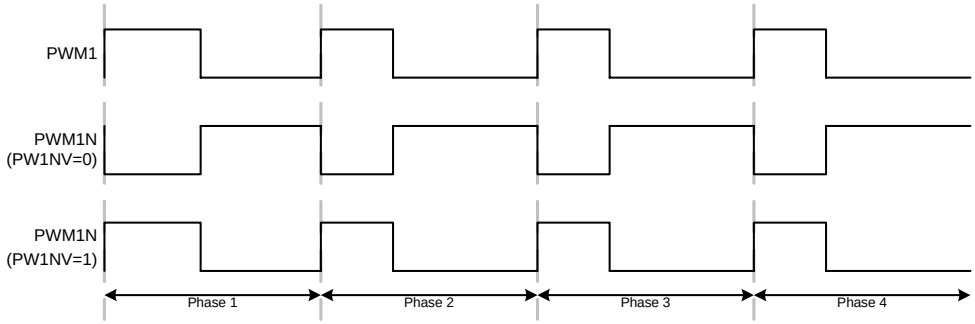
PWM 输出由 PW1EN 和 PW1S 位控制，PW1EN 位控制 PWM1 引脚为 GPIO 引脚或者 PWM 输出引脚，PW1EN=0 时，PWM1 为 GPIO 引脚；PW1EN=1 时，PWM1 为 PWM 输出引脚，初始状态为低电平。PW1S 位控制输出 PWM 信号，PW1EN=1 时有效，PW1S=0 时，禁止 PWM 信号输出，PWM1 为初始状态；PW1S=1 时，从 PWM1 引脚输出 PWM 信号。



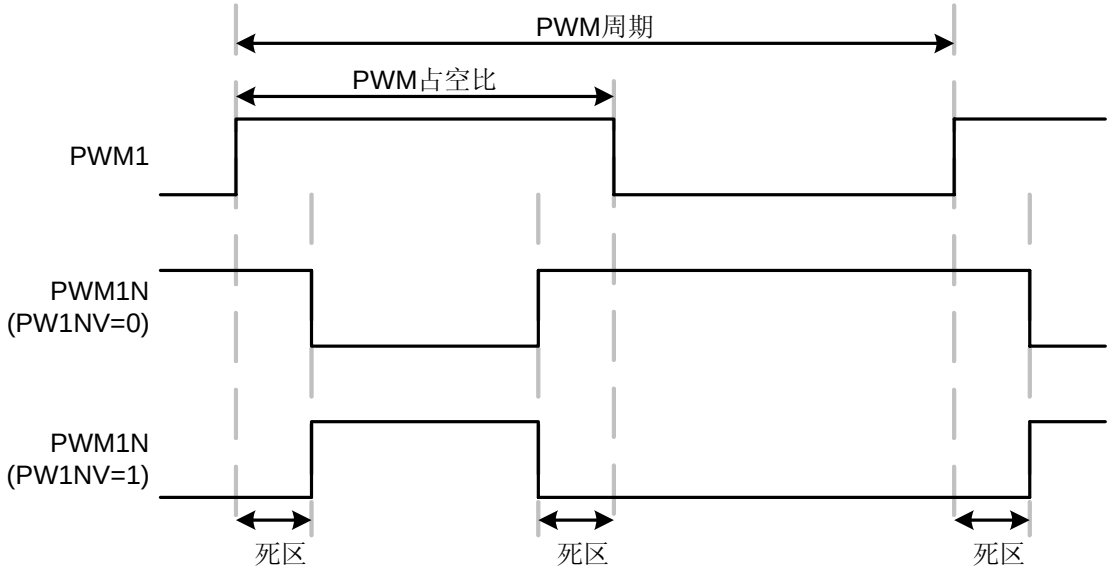
✷ 注：PW1S=1 ，PWM 开始输出，第一个 PWM 周期是一个完全的周期，在 PW1S 设置为上升沿时无延迟时间，也不会输出错误的 PWM 信号。

9.3 死区反向 PWM1 输出功能

PWM1 内置反向输出功能，由 PW1NEN 位控制，PW1NEN=0 时，PWM1N 引脚为 GPIO 引脚，PW1NEN=1 时，PWM1N 引脚为反向 PWM 输出引脚，并使用 PW1NV 位选择 PWM 输出相位。由 PWM1N 引脚输出的 PWM 信号由 PW1NV 位控制（此时 PW1NEN=1），PW1NV=0 时，PWM1N 引脚输出反向 PWM 信号；PWM1NV=1 时，PWM1N 引脚输出普通 PWM 信号。



反向 PWM 内置“死区”功能，其信号有延时。

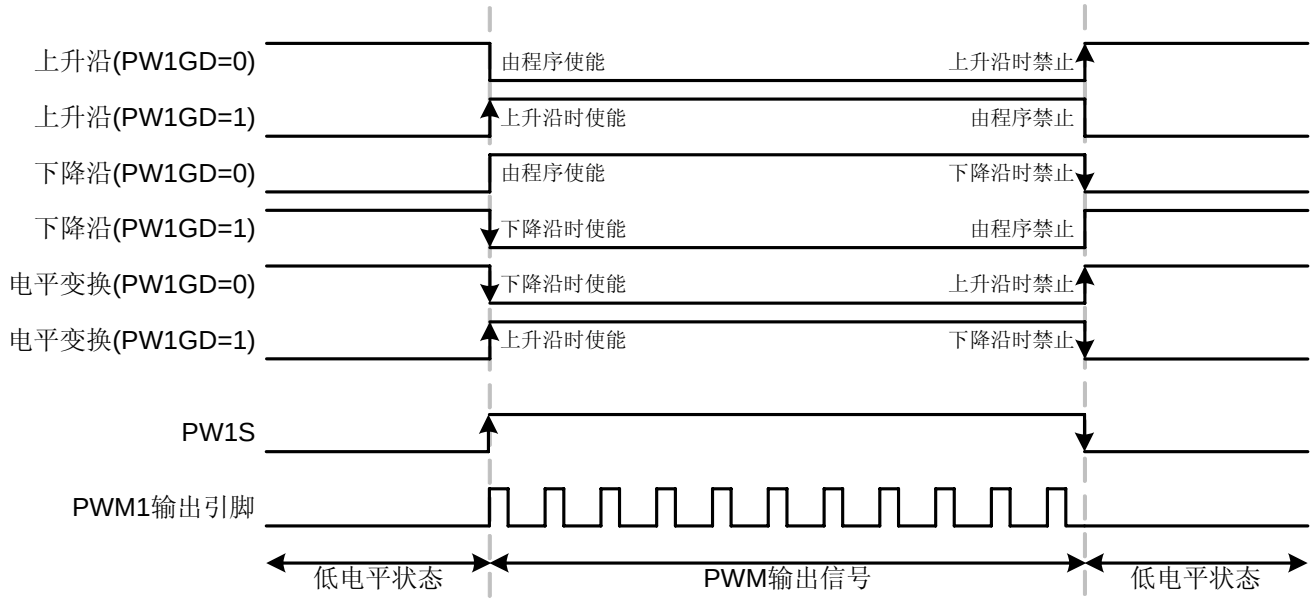


PWM 死区功能有 PW1DEN 位控制，PW1DEN=0 时，禁止 PWM1 死区功能；PW1DEN=1 时，使能 PWM1 死区功能，该位仅在使能 PWM 反向功能（PW1NEN=1）时有效。在 PWM1 脉冲高电平时会出现 PWM 死区反向功能，死区的时间可编程控制，死区在 PWM 高电平的两端对称出现，如死区时间设置为 1* Fpwm，出现在 PWM 高电平的左端，则 PWM 高电平的另一端会对称出现死区，故死区的总时间为 2* Fpwm。死区的时间由 PW1D[2:0]控制（000=1* Fpwm 时钟，001=2* Fpwm，010=3* Fpwm，011=4* Fpwm，100=5* Fpwm，101=6* Fpwm，110=7* Fpwm，111=8* Fpwm）。注意，在 PWM 高电平中出现死区是有必要的，但是死区的时间最好要小于 PWM 高电平的宽度时间，否则 PWM 的高电平会被屏蔽。

✷ **注：如果死区时间比 PWM 占空比长，则 PWM1N 无输出。**

9.4 PWM 同步触发功能

PWM1内置同步触发功能，触发源包括单片机的比较器0输出信号和PW1T外部触发引脚。通过该触发功能可以决定是否输出PWM信号，实际上PWM同步触发信号可以通过触发源控制PW1S位。PWM同步触发信号是由PW1GEN位控制的，当PW1GEN=0时，禁止PWM同步触发功能，PW1GEN=1时，使能PWM同步触发功能。



PWM1同步触发操作包括6种，由PW1GH位和触发源的方向共同决定。触发发生时，设置PW1GD位选择PWM操作：PW1GD=0，禁止PWM输出；PW1GD=1，使能PWM输出。如果是上升沿触发且PW1GD=0时，则在上升沿时停止PWM输出。触发源包括上升沿触发，下降沿触发和电平变换触发，上升沿和下降沿触发是单一的触发功能。触发发生时，必须由触发源设置好PW1S位，否则会由程序清零并做其它用途。电平变换触发使能和禁止PW1S位，即触发时，在第一个的电平开始时使能/禁止PW1S位，则在电平变换时自动禁止/使能该位。详见下表：

PW1GEN	PW1GD	CM0G[1:0] PW1GS=1	P00G[1:0] PW1GS=0	触发沿方向	PWM 同步触发操作	PW1S （PWM 起始位）
1	0	00	00	无触发	无触发	由程序控制
	1					
	0	01	01	上升沿	上升沿时禁止 PWM	0：由触发沿控制 1：由程序控制
	1				上升沿时使能 PWM	0：由程序控制 1：由触发沿控制
	0	10	10	下降沿	下降沿时禁止 PWM	0：由触发沿控制 1：由程序控制
	1				下降沿时使能 PWM	0：由程序控制 1：由触发沿控制
	0	11	11	电平变换触发	上升沿时禁止 PWM，下降沿时使能 PWM	0/1：由触发沿控制
	1				下降沿时禁止 PWM，上升沿时使能 PWM	0/1：由触发沿控制
0	-	-	-	-	禁止 PWM 同步触发功能	由程序控制

9.5 PWM1 模式寄存器

PWM1 模式寄存器包括 PW1M 和 PW1RH 寄存器，控制 PWM 的操作模式，包括 PWM 前置分频器，时钟源，PWM 分辨率，PWM 同步触发功能等。必须在使能 PWM1 功能之前设置好这些功能。

093H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1M	PW1EN	PW1rate2	PW1rate1	PW1rate0	PW1CKS	PW1LN1	PW1LN0	PW1S
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 **PW1EN**：PWM1 控制位。
0 = 禁止，PWM1（P0.1）和 PWM1N（P0.2）引脚作为 GPIO 引脚；
1 = 使能，PWM1（P0.1）输出 PWM 低电平。

Bit [6:4] **PW1rate[2:0]**：PWM1 时钟分频控制位。注：**Fpwm 是 PWM1 时钟源。**
000 = Fpwm/128，001 = Fpwm/64，010 = Fpwm/32，011 = Fpwm/16，100 = Fpwm/8，101 = Fpwm/4，110 = Fpwm/2，111 = Fpwm/1。

Bit 3 **PW1CKS**：PWM1 时钟源选择位。
0 = Fosc；
1 = Fcpu。

Bit [2:1] **PW1LN[1:0]**：PWM1 分辨率选择位。
00 = 8 位；01 = 10 位；10 = 12 位；11 = 保留。

Bit 0 **PW1S**：PWM1 开始输出控制，由程序或 PWM 同步触发源控制。
0 = 禁止 PWM 输出，PWM1 输出引脚为低电平状态；
1 = PWM 开始输出。

096H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1RH	PW1GS	PW1GEN	PW1GD	-	PW1R11	PW1R10	PW1R9	PW1R8
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 **PW1GS**：PWM1 同步触发源选择位，在使能 PWM 同步触发功能是可使用此功能。
0 = PWM1T 引脚（P0.0）；
1 = 比较器 0 输出信号。

Bit 6 **PW1GEN**：PWM1 同步触发功能控制位。
0 = 禁止；
1 = 使能。

Bit 5 **PW1GD**：PWM1 同步触发操作控制位。
0 = 禁止 PWM 输出；
1 = 使能 PWM 输出。

090H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1NM	PW1NEN	PW1D2	PW1D1	PW1D0	PW1DEN	PW1NV	-	-
读/写	R/W	R/W	R/W	R/W	R/W	R/W	-	-
复位后	0	0	0	0	0	0	-	-

Bit 7 **PW1NEN**：PWM1 反向输出功能控制位。
0 = 禁止，PWM1N（P0.2）为 GPIO 引脚；
1 = 使能 PWM1 反向输出功能（PW1EN=1），PWM1N 为 PWM1 反向输出引脚，并与 P0.2 GPIO 功能隔离。

Bit [6:4] **PW1D[2:0]**：PWM1 反向输出死区时间选择位。
000 = 1*Fpwm clock；001 = 2* Fpwm clock；010 = 3* Fpwm clock；011 = 4* Fpwm clock；
100 = 5*Fpwm clock；101 = 6* Fpwm clock；110 = 7* Fpwm clock；111 = 8* Fpwm clock。

Bit 3 **PW1DEN**：PWM1 死区反向输出控制位。
0 = 禁止；
1 = 使能。

Bit 2 **PW1NV**：PWM1 反向输出控制位。
0 = PW1N 引脚输出 PWM1 死区反向信号；
1 = PW1N 引脚输出 PWM1 死区信号。

9.6 PWM1 占空比寄存器

PWM1 的占空比由 PW1RH 和 PW1RL 寄存器控制，PWM 占空比寄存器的长度为 12 位，通过 PWM 计数器的值和 PWM 相位处理产生 PWM 输出信号。PWM 具有自动装载功能，在 PWM 输出过程中，通过程序更改 PWM 的占空比，则在下一个周期时会产生新的占空比，而不是立即改变，这样可以避免出现过渡期的占空比。

096H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1RH	PW1GS	PW1GEN	PW1GD	-	PW1R11	PW1R10	PW1R9	PW1R8
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 [3:0] **PW1R [11:8]** = PWM1 占空比配置位[11:8]。

097H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW1RL	PW1R7	PW1R6	PW1R5	PW1R4	PW1R3	PW1R2	PW1R1	PW1R0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **PW1R [7:0]** = PWM1 占空比配置位[7:0]。

PWM 占空比寄存器的长度为 12 位，包括 PW1RH 和 PW1RL。该定时计数器为双重缓存器设计，核心总线为 8 位，故处理 12 位数据需锁存标志以避免瞬时状态影响 12 位数据而出错。写入模式下，PW1RL 为锁存控制标志，要先写入数据到 PW1RH 寄存器，再写入 PW1RL 中，执行写入 PW1RL 后，所有数据都已经写入 12 位缓存器中。

➢ **写入 12 位数据到 PWM 占空比寄存器中时，要先写入 PW1RH 中，再写入 PW1RL 中。**

PWM 占空比的长度由 PWM 分辨率（8/10/12 位）决定，不同的 PWM 分辨率，PWM 占空比寄存器有效位也不同。

- **8 位分辨率：PWM 占空比缓存器有效位为 PW1R0~PW1R7，PW1R8~PW1R11 置 0。**
- **10 位分辨率：PWM 占空比缓存器有效位为 PW1R0~PW1R9，PW1R10~PW1R11 置 0。**
- **12 位分辨率：PWM 占空比缓存器有效位为 PW1R0~PW1R11.。**

✷ **注：PW1R[11:0]的写入顺序为：先写入 PW1R[11:8]，再写入 PW1R[7:0]，写入 PW1R[7:0]完成后表示数据全部写入 PW1R 寄存器中。**

9.7 PWM1 操作举例

● **PWM1：**

```
; 复位 PWM1。
        CLR     PW1M            ; 清 PWM1 模式寄存器。

; 设置 PWM1 时钟源，时钟 Rate 和分辨率。
        MOV     A, #0mmmnll0b   ; “mmm”代表 PWM 时钟 Rate 选择位。
        B0MOV   PW1M, A         ; “n”代表 PWM 时钟源选择位。
                                ; “ll”代表 PWM 分辨率选择位。

; 设置 PWM1 的占空比。
        MOV     A, #value1      ; 先设置高字节。
        B0MOV   PW1RH, A
        MOV     A, #value2      ; 再设置低字节。
        B0MOV   PW1RL, A

; 使能 PWM1 功能。
        B0BSET  FPW1EN

; 输出 PWM 信号。
        B0BSET  FPW1S
```

● **使能 PWM1 同步触发功能：（使能 PWM1 之前配置 PW1RH[5:7]位以使能 PWM1 同步触发功能。）**

```
; 参照上例配置 PWM1。
        …

; 设置 PWM1 同步触发源和触发操作。
        MOV     A, #n0m000000b  ; “n”代表 PWM1 同步触发源选择位。
        OR      PW1RH, A        ; “m”代表 PWM1 同步触发操作选择位。

; 使能 PWM1 同步触发功能。
        B0BSET  FPW1GEN

; 使能 PWM1 功能。
        B0BSET  FPW1EN

; 输出 PWM 信号。
        B0BSET  FPW1S
```

● **使能 PWM1 反向输出功能：（使能 PWM1 之前配置 PW1NM 模式寄存器以使能 PWM1 反向输出功能。）**

```
; 参照上例配置 PWM1。
        …

; 复位 PW1NM 寄存器。
        CLR     PW1NM

; 选择 PWM1N 引脚输出 PWM 信号。
        B0BCLR  FPW1NV          ; PWM1N 引脚输出 PWM1 反向信号。
or
        B0BSET  FPW1NV          ; PWM1N 引脚输出 PWM1 信号。

; 使能 PWM1 反向输出功能。
        B0BSET  FPW1NEN

; 使能 PWM1 功能。
        B0BSET  FPW1EN

; 输出 PWM 信号。
        B0BSET  FPW1S
```

● 使能 **PWM1** 死区功能：（使能 **PWM1** 之前设置 **PW1NM** 模式寄存器以使能 **PWM1** 死区功能。）

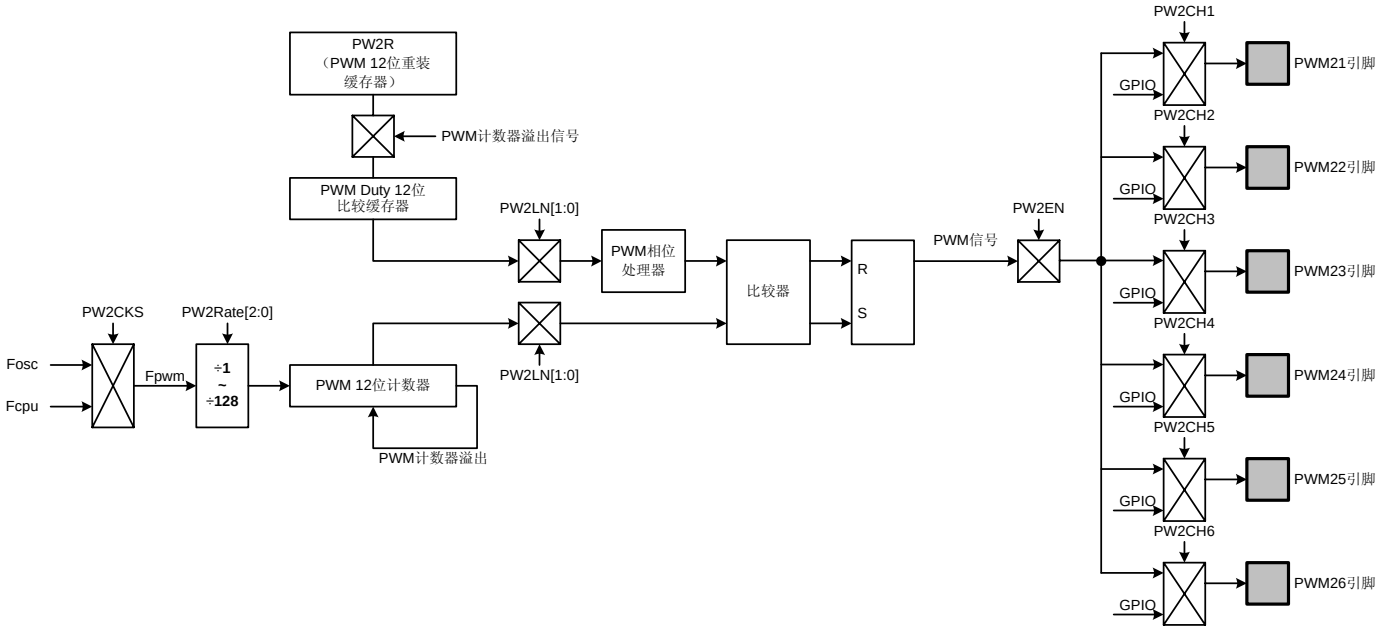
```
; 参照上例配置 PWM1。
        …

; 参照上例设置 PWM1 反向输出功能。
        …

; 设置 PWM1 死区时间。
        MOV     A, #0nnn0000b       ; “nnn” is PW1D[2:0] dead-band period selection.
        OR      PW1NM, A

; 使能 PWM1 死区功能。
        B0BSET  FPW1DEN

; 使能 PWM1 反向输出功能。
        B0BSET  FPW1NEN

; 使能 PWM1 功能。
        B0BSET  FPW1EN

; 输出 PWM 信号。
        B0BSET  FPW1S
```

10 6 通道脉冲宽度调制（PWM2）

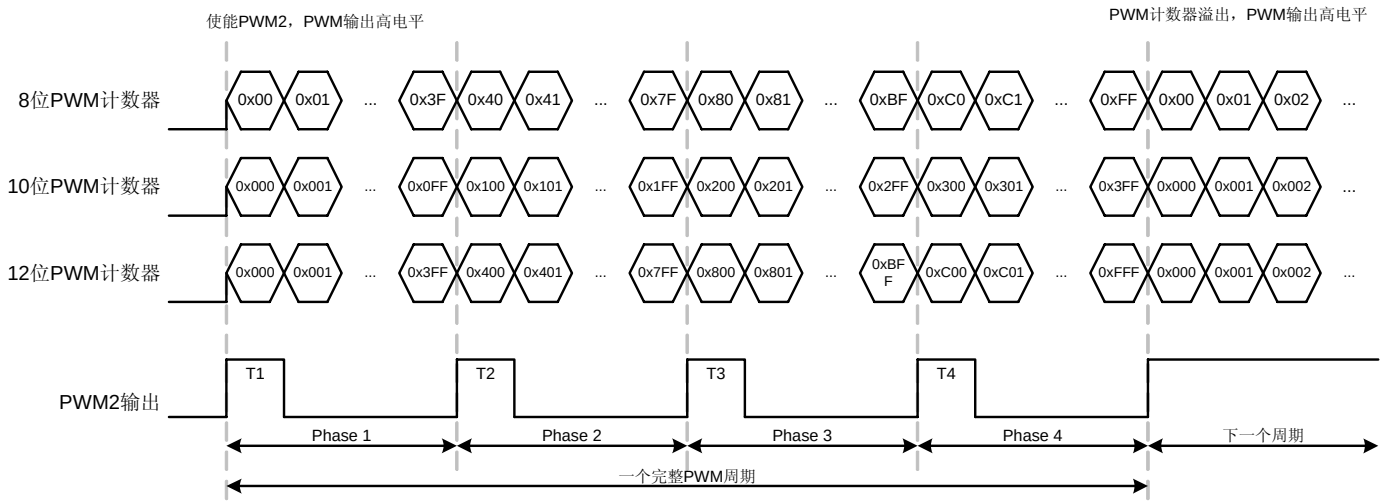
10.1 概述

6 通道脉冲宽度调制（PWM）是高性能设计，包括可编程的高分辨率，可编程的占空比/周期和高频功能。PWM 分辨率有 8 位，10 位和 12 位，PWM 周期由 PWM 时钟源、PWM 时钟 Rate 选择选项和 PWM 分辨率决定。PWM 时钟源由 Fcpu 和 Fhosc 提供，PWM 时钟 Rate 包括 Fosc/1~Fosc/128 和 Fcpu/1~Fcpu/128。PWM 利用新工艺完善 PWM 高频，减少纹波对输出信号的影响。PWM 的输出引脚和 GPIO 共用，由 PW2CHS 寄存器控制。6 通道 PWM 易于实现 3 相 DC 马达控制，如 BLDC 马达。PWM 的主要功能如下：

- **高速 PWM。**
- **8/10/12 位可编程的分辨率。**
- **多种时钟源，包括 Fosc，Fcpu，Fpwm/1~Fpwm/128。**
- **6 通道独立输出引脚。**



10.2 PWM2 COMMON 操作



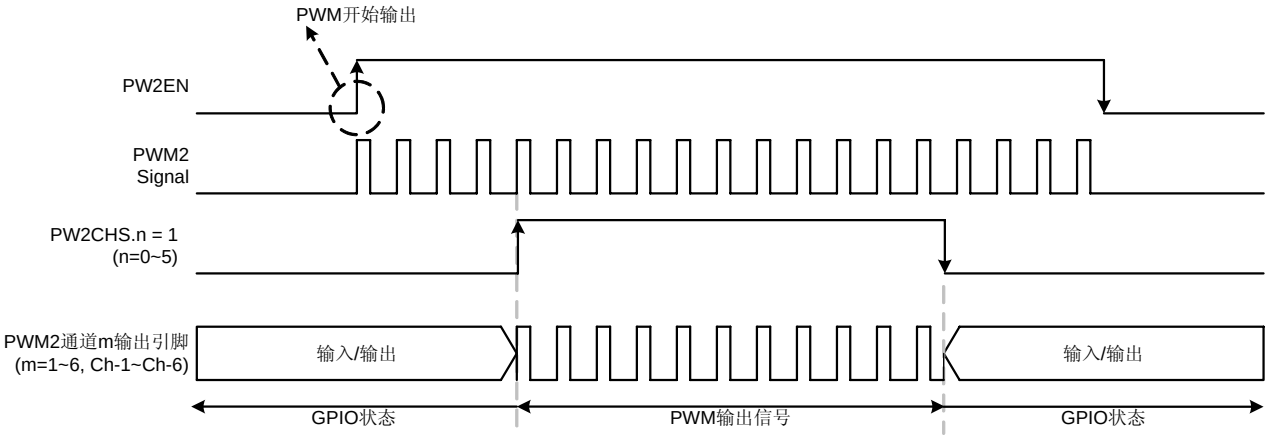
PWM 为 4 相设计，PWM 的一个周期包括 4 个小周期，PWM 信号保持最初的周期参数，但频率快一些，PWM 的频率与小周期的频率相等。上面 PWM 的周期为“（T1+T2+T3+T4）/（PWM 周期）”，4 相占空比的输出顺序是一个特殊的设计和更多的平衡。

PWM 的周期由 PWM 时钟源和 PWM 设置 Rate 选项控制，PW2CKS 位选择 PWM 时钟源（F_{PWM}）来自 Fosc 或者 Fcpu，PW2Rate[2:0]位选择 PWM 时钟 Rate，可以在 $F_{PWM}/1$~$F_{PWM}/128$ 之间选择。

PWM 重装缓存器（PW2RH，PW2RL）决定 PWM 的占空比，通过 PWM 相位处理器来分配每相的占空比。PWM 具有自动装载功能，在 PWM 输出过程中，通过程序更改 PWM 的占空比，则在下一个周期时会产生新的占空比，而不是立即改变，这样可以避免出现过渡期的占空比。

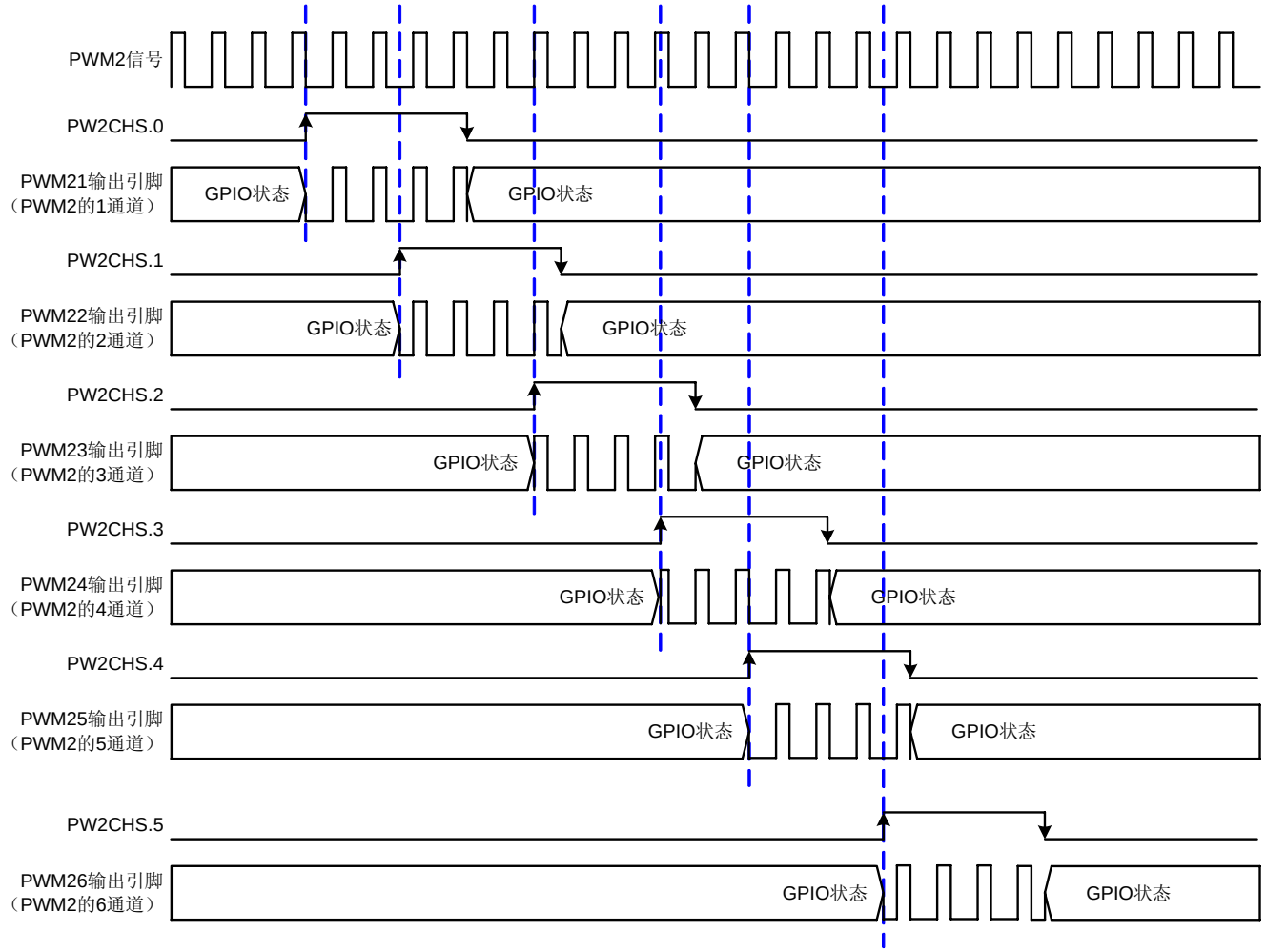
PWM 分辨率包括 8 位，10 位和 12 位，由 PW2LN[1:0]控制，PW2LN[1:0]=00 时选择 8 位 PWM，PW2LN[1:0]=01 时选择 10 位 PWM，PW2LN[1:0]=10 时选择 12 位 PWM。根据实际需要选择合适的 PWM 分辨率。

PWM 输出由 PW2EN 控制，PW2EN 位控制 PWM2 功能，PW2EN=0 时，禁止 PWM2 功能；PW2EN=1 时，PWM2 引脚输出 PWM 信号。PW2CHS 位决定 PWM 输出引脚的功能模式（GPIO 引脚和 PWM 输出引脚）。PW2CHS 的 0~6 位对应 PWM 的 6 个通道（PWM21~PWM26）。当 PW2CHS.n=0 时，对应的通道 n 的 PWM2 引脚为 GPIO 引脚；PWM2CHS.n=1 时，对应的通道 n 的 PWM2 引脚为 PWM 输出引脚。PW2CHS 寄存器在 PWM2EN=1 时才有效。



PWM2 输出引脚作为 GPIO 模式时可以设置为 PWM 信号的空闲状态，PWM 空闲状态为高时，GPIO 输出高电平；PWM 空闲状态为低时，GPIO 输出低电平；PWM 空闲状态为高阻抗状态时，GPIO 为输入模式。禁止 PWM 时选择合适的 PWM 空闲状态对转载控制信号很重要。

PWM 信号由内部 PWM 处理器和外部输出引脚（PWM21~PWM26）产生，外部输出引脚由 PWM2 通道选择位决定，内部和外部产生的 PWM 信号时相同的。通道选择位仅选择 PWM2 通道，并不处理 PWM 信号的相位。



10.3 PWM2 模式寄存器

PW2M 模式寄存器控制 PWM 的操作模式，包括 PWM 前置分频器，时钟源，PWM 分辨率等。必须在使能 PWM2 功能之前设置好这些功能。

094H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW2M	PW2EN	PW2rate2	PW2rate1	PW2rate0	PW2CKS	PW2LN1	PW2LN0	-
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 **PW2EN**：PWM2 控制位。
0 = 禁止；
1 = 使能。

Bit [6:4] **PW2rate[2:0]**：PWM2 时钟分频控制位。注：**Fpwm 是 PWM2 时钟源。**
000 = Fpwm/128，001 = Fpwm/64，010 = Fpwm/32，011 = Fpwm/16，100 = Fpwm/8，101 = Fpwm/4，110 = Fpwm/2，111 = Fpwm/1。

Bit 3 **PW2CKS**：PWM2 时钟源选择位。
0 = Fosc；
1 = Fcpu。

Bit [2:1] **PW2LN[1:0]**：PWM2 分辨率选择位。
00 = 8 位；01 = 10 位；10 = 12 位；11 = 保留。

10.4 PWM2 通道选择寄存器

通过 PW2CHS 寄存器，PWM2 可以选择 6 个通道，使能 PW2CHS 的相关位，则对应的引脚输出 PWM2 信号；如果禁止相关位，则对应的引脚返回到上一个 GPIO 模式。

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW2CHS	-	-	PW2CH6	PW2CH5	PW2CH4	PW2CH3	PW2CH2	PW2CH1
读/写	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	0	0	0	0	0	0

Bit 5 **PW2CH6**：PWM2 6 通道控制位。
0 = 禁止 PWM 输出，PWM26 引脚作为 P5.7 GPIO 引脚；
1 = PWM26 输出 PWM 信号，GPIO 功能被隔离。

Bit 4 **PW2CH5**：PWM2 5 通道控制位。
0 = 禁止 PWM 输出，PWM25 引脚作为 P5.6 GPIO 引脚；
1 = PWM25 输出 PWM 信号，GPIO 功能被隔离。

Bit 3 **PW2CH4**：PWM2 4 通道控制位。
0 = 禁止 PWM 输出，PWM24 引脚作为 P5.5 GPIO 引脚；
1 = PWM24 输出 PWM 信号，GPIO 功能被隔离。

Bit 2 **PW2CH3**：PWM2 3 通道控制位。
0 = 禁止 PWM 输出，PWM23 引脚作为 P5.3 GPIO 引脚；
1 = PWM23 输出 PWM 信号，GPIO 功能被隔离。

Bit 1 **PW2CH2**：PWM2 2 通道控制位。
0 = 禁止 PWM 输出，PWM22 引脚作为 P5.2 GPIO 引脚；
1 = PWM22 输出 PWM 信号，GPIO 功能被隔离。

Bit 0 **PW2CH1**：PWM2 1 通道控制位。
0 = 禁止 PWM 输出，PWM21 引脚作为 P5.1 GPIO 引脚；
1 = PWM21 输出 PWM 信号，GPIO 功能被隔离。

✷ **注：PW2CHS 寄存器仅在 PW2EN=1 时有效，否则 PWM 输出引脚作为 GPIO 引脚。**

10.5 PWM2 占空比寄存器

PWM2 的占空比由 PW2RH 和 PW2RL 寄存器控制，PWM 占空比寄存器的长度为 12 位，通过 PWM 计数器的值和 PWM 相位处理产生 PWM 输出信号。PWM 具有自动装载功能，在 PWM 输出过程中，通过程序更改 PWM 的占空比，则在下一个周期时会产生新的占空比，而不是立即改变，这样可以避免出现过渡期的占空比。

098H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW2RH	-	-	-	-	PW2R11	PW2R10	PW2R9	PW2R8
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	0	0	0	0

Bit [3:0] **PW2R [11:8]** = PWM2 占空比配置位[11:8]。

098H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PW2RL	PW2R7	PW2R6	PW2R5	PW2R4	PW2R3	PW2R2	PW2R1	PW2R0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

Bit [7:0] **PW2R [7:0]** = PWM2 占空比配置位[7:0]。

PWM 占空比寄存器的长度为 12 位，包括 PW2RH 和 PW2RL。该定时计数器为双重缓存器设计，核心总线为 8 位，故处理 12 位数据需锁存标志以避免瞬时状态影响 12 位数据而出错。写入模式下，PW2RL 为锁存控制标志，要先写入数据到 PW2RH 寄存器，再写入 PW2RL 中，执行写入 PW2RL 后，所有数据都已经写入 12 位缓存器中。

➢ **写入 12 位数据到 PWM 占空比寄存器中时，要先写入 PW2RH 中，再写入 PW2RL 中。**

PWM 占空比的长度由 PWM 分辨率（8/10/12 位）决定，不同的 PWM 分辨率，PWM 占空比寄存器有效位也不同。

- **8 位分辨率：PWM 占空比缓存器有效位为 PW2R0~PW2R7，PW2R8~PW2R11 置 0。**
- **10 位分辨率：PWM 占空比缓存器有效位为 PW2R0~PW2R9，PW2R10~PW2R11 置 0。**
- **12 位分辨率：PWM 占空比缓存器有效位为 PW2R0~PW2R11.。**

✷ **注：PW2R[11:0]的写入顺序为：先写入 PW2R[11:8]，再写入 PW2R[7:0]，写入 PW2R[7:0]完成后表示数据全部写入 PW2R 寄存器中。**

10.6 PWM2 操作举例

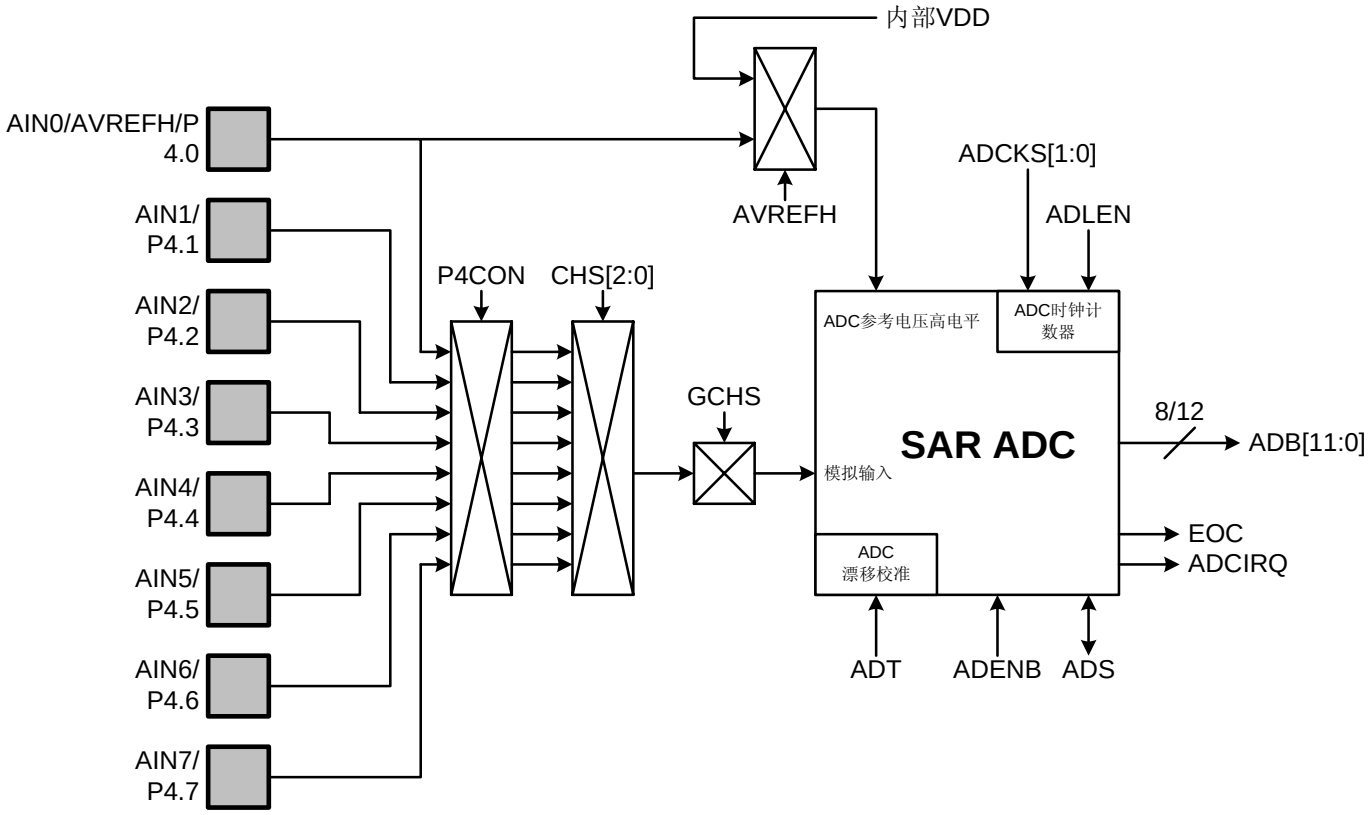
- **PWM2：**

```
; 复位 PWM2。
        CLR     PW2M            ; 清 PWM2 模式寄存器。

; 设置 PWM2 时钟源，时钟 Rate 和分辨率。
        MOV     A, #0mmmnll0b   ; “mmm”代表 PWM 时钟 Rate 选择位。
        B0MOV   PW2M, A         ; “n”代表 PWM 时钟源选择位。
                                ; “ll”代表 PWM 分辨率选择位。

; 设置 PWM2 的占空比。
        MOV     A, #value1      ; 先设置高字节。
        B0MOV   PW2RH, A
        MOV     A, #value2      ; 再设置低字节。
        B0MOV   PW2RL, A

; 设置 PWM 的空闲状态。
        MOV     A, #11101110b   ; PWM21~PWM26 空闲状态为高。
        OR      P5, A
        MOV     A, #11101110b
        OR      P5M, A
or
        MOV     A, #00010001b   ; PWM21~PWM26 空闲状态为低。
        AND     P5, A
        MOV     A, #11101110b
        OR      P5M, A

; 使能 PWM2 功能。
        B0BSET  FPW2EN

; 输出 PWM 信号。
        B0BSET  FPW2CH1
        B0BSET  FPW2CH2
        B0BSET  FPW2CH3
        B0BSET  FPW2CH4
        B0BSET  FPW2CH5
        B0BSET  FPW2CH6
```

11 8 通道模拟数字转换（ADC）

11.1 概述

模拟数字转换（ADC）是一个 SAR 结构，内置 8 个模拟通道（AIN0~AIN7），高达 4096 阶的分辨率，能将一个模拟信号转换成相应的 12 位数字信号。ADC 内置的 8 个模拟通道可以测量 8 种不同的模拟信号源，由 CHS[2:0]和 GCHS 位控制。通过 ADLEN 位可以选择 ADC 的分辨率为 8 位或者 12 位；可以通过 ADCKS[1:0]位选择 ADC 的转换速率以决定 ADC 的转换时间。ADC 参考电压的高电平有两种提供方式，由 AVREFH 位决定：其中一种是内部 VDD（AVREFH=0），另外一种是由 P4.0 输入的外部参考源（AVREFH=1）。ADC 内置 P4CON 寄存器来设置模拟输入引脚，必须由程序将 P4.0 设为带上拉电阻的输入引脚。设置好 ADENB 和 ADS 位后，ADC 开始转换，转换结束时，ADC 电路将 EOC 和 ADCIRQ 置 1，并将转换结构存入 ADB 和 ADR 寄存器中。若 ADCIEN=1，ADC 请求中断，AD 转换完成后，ADCIRQ=1 时，程序计数器跳转中断向量地址执行中断服务程序。中断时必须由程序将 ADCIRQ 清零。



11.2 ADC 模式寄存器

ADC 模式寄存器 ADM 设置 ADC 的相关配置：包括 ADC 启动，ADC 通道选择，ADC 参考电压高电平输入源和 ADC 处理状态显示等。必须在 AD 开始转换前将这些配置设置完毕。

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	AVREFH	CHS2	CHS1	CHS0
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 **ADENB**：ADC 控制位。**省电模式下，禁止 ADC 以省电。**
0 = 禁止 ADC；
1 = 使能 ADC。

Bit 6 **ADS**：ADC 启动控制位。**AD 转换完成后自动将 ADS 位清零。**
0 = 停止 AD 转换；
1 = 开始 AD 转换。

Bit 5 **EOC**：ADC 状态位。**ADC 开始之前必须由程序将 EOC 位清零。**
0 = AD 转换中；
1 = AD 转换结束，ADC 位复位。

Bit 4 **GCHS**：ADC 全局通道控制位。
0 = 禁止 AIN 通道；
1 = 使能 AIN 通道。

Bit 3 **AVREFH**：ADC 参考电压高电平输入控制位。
0 = 内部 VDD，P4.0 为 GPIO 或 AIN0 引脚；
1 = 由 P4.0 输入的外部参考电压。

Bit [2:0] **CHS[2:0]**：ADC 输入通道选择位。
000 = AIN0；001 = AIN1；010 = AIN2；011 = AIN3；100 = AIN4；101 = AIN5；110 = AIN6；111 = AIN7。

ADR 寄存器包括 ADC 模式控制和 ADC 低字节数据缓存器，ADC 配置包括 ADC 时钟速率和 ADC 分辨率。必须在启动 ADC 之前设置好这些配置。

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	-	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0
读/写	-	R/W	R/W	R/W	R	R	R	R
复位后	-	0	0	0	-	-	-	-

Bit 6,4 **ADCKS [1:0]**：ADC 时钟 rate 选择位。
00 = Fcpu/16；01 = Fcpu/8；10 = Fcpu/1；11 = Fcpu/2。

Bit 5 **ADLEN**：ADC 分辨率选择位。
0 = 8 位；
1 = 12 位。

11.3 ADC 数据缓存器

ADC 数据缓存器共 12 位，用来存储 AD 转换结果，ADB 存放结果的高字节（bit4~bit11），ADR（ADR[3:0]）存放低字节（bit0~bit3）。ADC 数据缓存器是只读寄存器，系统复位后处于未知状态。

ADB[11:4]：8 位 ADC 模式下，ADC 数据存放于 ADB 寄存器中。
ADB[11:0]：12 位 ADC 模式下，ADC 数据存放于 ADB 和 ADR 寄存器中。

0B2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADB	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4
读/写	R	R	R	R	R	R	R	R
复位后	-	-	-	-	-	-	-	-

Bit[7:0] **ADB[7:0]：**8 位 ADC 数据缓存器，存放 12 位 ADC 的高字节数据。

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	-	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0
读/写	-	R/W	R/W	R/W	R	R	R	R
复位后	-	0	0	0	-	-	-	-

Bit [3:0] **ADB [3:0]：**12 位 ADC 的低字节数据缓存器。

AIN 输入电压 v.s. ADB 输出数据

AIN n	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
0/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	0
1/4096*VREFH	0	0	0	0	0	0	0	0	0	0	0	1
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
4094/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	0
4095/4096*VREFH	1	1	1	1	1	1	1	1	1	1	1	1

针对不同的应用，用户可能需要精度介于 8 位到 12 位之间的 AD 转换器。对于这种情况，可以通过对保存在 ADR 和 ADB 中的转换结果进行处理得到。首先，用户必须选择 12 位分辨率的模式，进行 AD 转换，然后在转换结果中去掉最低的几位得到需要的结果。如下表所示：

ADC 分辨率	ADB								ADR			
	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
8-bit	O	O	O	O	O	O	O	O	x	x	x	x
9-bit	O	O	O	O	O	O	O	O	O	x	x	x
10-bit	O	O	O	O	O	O	O	O	O	O	x	x
11-bit	O	O	O	O	O	O	O	O	O	O	O	x
12-bit	O	O	O	O	O	O	O	O	O	O	O	O
O = 可选位，x = 无效位												

✷ **注：ADC 数据缓存器包括 ADB 和 ADR 低字节，系统复位后，ADC 数据缓存器的值是未知的。**

11.4 ADC 操作说明和注意事项

11.4.1 ADC 信号格式

ADC 采样电压范围为参考电压高/低电平之间，ADC 参考低电压为 VSS，高电压有两种，由 AVREFH 为决定：内部 VDD 和 P4.0/AVREFH 输入的外部参考电压。AVREFH=0 时，ADC 参考电压由内部 VDD（单片机电源电压）提供；AVREFH=1 时，ADC 参考电压由外部参考源提供。ADC 参考电压的范围为：**（ADC 参考高电压-ADC 参考低电压）≧ 2V**，ADC 参考低电压为 VSS=0V，故 **ADC 参考高电压范围为 2V~VDD**，外部参考电压需在此范围之内。

- **ADC 内部参考低电压=0V。**
- **ADC 内部参考高电压=VDD（AVREFH=0）。**
- **ADC 外部参考电压=2V~VDD（AVREFH=1）。**

ADC 采样输入信号电压必须在 ADC 参考低电压和 ADC 参考高电压之间，若 ADC 输入信号的电压不在此范围内，则 ADC 的转换结果会出错（满量程或者为 0）。

- **ADC 参考低电压 ≦ ADC 采用输入信号电压 ≦ ADC 参考高电压**

11.4.2 ADC 转换时间

ADC 转换时间是指从 ADS=1（开始 ADC）到 EOC=1（ADC 结束）所用的时间，由 ADC 分辨率和 ADC 时钟 Rate 控制，12 位 ADC 的转换时间为 1/（ADC 时钟/4）*16 S；8 位 ADC 的转换时间为 1/（ADC 时钟/4）*12 S。ADC 的时钟源为 Fcpu，包括 Fcpu/1，Fcpu/2，Fcpu/8，Fcpu/16，由 ADCKS[1:0]位控制。

ADC 的转换时间会影响 ADC 的性能，如果输入高 Rate 的模拟信号，必须要选择一个高 Rate 的 ADC 转换 Rate。如果 ADC 的转换时间比模拟信号的转换 Rate 慢，则 ADC 的结果出错。故选择合适的 ADC 时钟 Rat 和 ADC 分辨率才能得到合适的 ADC 转换 Rate。

12 位 ADC 转换时间 = 1/(ADC 时钟 Rate/4)*16 sec

ADLEN	ADCKS1, ADCKS0	ADC 时钟 Rate	Fcpu=4MHz		Fcpu=16MHz	
			ADC 转换时间	ADC 转换 Rate	ADC 转换时间	ADC 转换 Rate
1 (12-bit)	00	Fcpu/16	1/(4MHz/16/4)*16 = 256 us	3.906KHz	1/(16MHz/16/4)*16 = 64 us	15.625KHz
	01	Fcpu/8	1/(4MHz/8/4)*16 = 128 us	7.813KHz	1/(16MHz/8/4)*16 = 32 us	31.25KHz
	10	Fcpu	1/(4MHz/4)*16 = 16 us	62.5KHz	1/(16MHz/4)*16 = 4 us	250KHz
	11	Fcpu/2	1/(4MHz/2/4)*16 = 32 us	31.25KHz	1/(16MHz/2/4)*16 = 8 us	125KHz

8 位 ADC 转换时间 = 1/(ADC 时钟 Rate/4)*12 sec

ADLEN	ADCKS1, ADCKS0	ADC 时钟 Rate	Fcpu=4MHz		Fcpu=16MHz	
			ADC 转换时间	ADC 转换 Rate	ADC 转换时间	ADC 转换 Rate
0 (8-bit)	00	Fcpu/16	1/(4MHz/16/4)*12 = 192 us	5.208KHz	1/(16MHz/16/4)*12 = 48 us	20.833KHz
	01	Fcpu/8	1/(4MHz/8/4)*12 = 96 us	10.416KHz	1/(16MHz/8/4)*12 = 24 us	41.667KHz
	10	Fcpu	1/(4MHz/4)*12 = 12 us	83.333KHz	1/(16MHz/4)*12 = 3 us	333.333KHz
	11	Fcpu/2	1/(4MHz/2/4)*12 = 24 us	41.667KHz	1/(16MHz/2/4)*12 = 6 us	166.667KHz

11.4.3 ADC 引脚配置

ADC 输入引脚与 P4 口共用，ADC 输入通道的选择由 ADCHS[2:0]控制，ADCCHS[2:0]=000 时选择 AIN0，ADCCHS[2:0]=001 时选择 AIN1……同一时间设置 P4 口的一个引脚作为 ADC 的输入引脚，该引脚必须设置为输入引脚，禁止内部上拉，并首先由程序使能 P4CON 寄存器。通过 ADCHS[2:0]选择好 ADC 输入通道后，GCHS 置 1 以使能 ADC 功能。

- **ADC 输入引脚为 GPIO 引脚时必须设为输入模式。**
- **必须禁止 ADC 输入引脚的内部上拉电阻。**
- **ADC 输入通道的 P4CON 位必须置 1。**

AVREFH=1 时，P4.0/AIN0 可以作为 ADC 外部参考源的输入引脚，此时，该引脚必须设为输入模式，且禁止内部上拉。

- **ADC 外部参考电压的输入引脚必须设为输入模式。**
- **ADC 外部参考电压的输入引脚必须禁止内部上拉电阻。**

ADC 输入引脚与普通 I/O 引脚共用。当输入一个模拟信号到 CMOS 结构端口时，尤其当模拟信号为 1/2 VDD 时，可能产生额外的漏电流。当 P4 输入多个模拟信号时，也会产生额外的漏电流。睡眠模式下，上述漏电流会严重影响到系统的整体功耗。P4CON 为 P4 口的配置寄存器，将 P4CON[7:0]置 1，其对应的 P4 引脚将被设为纯模拟信号输入引脚，从而避免上述漏电流的产生。

0AEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

Bit[4:0] **P4CON[7:0]**：P4.n 配置控制位。
0 = P4.n 可以作为模拟输入（ADC 输入）引脚或者 GPIO 引脚；
1 = P4.n 只能作为模拟输入引脚，不能作为 GPIO 引脚。

✷ **注：P4.n 作为 GPIO 引脚而不是 ADC 输入引脚时，P4CON.你必须置 0，否则 P4.n 的普通 I/O 信号会被隔离。**

11.4.4 ADC 操作举例

- **ADC：**

```
; 复位 ADC。
        CLR     ADM             ; 清 ADM 寄存器。

; 设置 ADC 时钟 Rate 和 ADC 分辨率。
        MOV     A, #0nmn0000b   ; nn-DCKS[1:0]代表 ADC 时钟 Rate。
        B0MOV   ADR, A          ; m 代表 ADC 分辨率。

; 设置 ADC 参考高电压。
        B0BCLR  FAVREFH         ; 内部 VDD。
or
        B0BSET  FAVREFH         ; 外部参考源。

; 设置 ADC 输入通道。
        MOV     A, #value1      ; 设置 P4CON 选择 ADC 输入通道。
        B0MOV   P4CON, A
        MOV     A, #value2      ; 设置 ADC 输入通道为输入模式。
        B0MOV   P4M, A
        MOV     A, #value3      ; 禁止 ADC 输入通道的内部上拉电阻。
        B0MOV   P4UR, A

; 使能 ADC。
        B0BSET  FADCENB

; 执行 ADC 100us 启动时间延迟循环。
        CALL    100usDLY        ; 100us 延迟循环。

; 选择 ADC 输入通道。
        MOV     A, #value       ; 设置 ADCHS[2:0]选择 ADC 输入通道。
        OR      ADM, A

; 使能 ADC 输入通道。
        B0BSET  FGCHS

; 使能 ADC 中断功能。
        B0BCLR  FADCIRQ         ; 清 ADC 中断请求。
        B0BSET  FADCIEN         ; 使能 ADC 中断功能。

; 开始 AD 转换。
        B0BSET  FADS
```

✸ **注：**

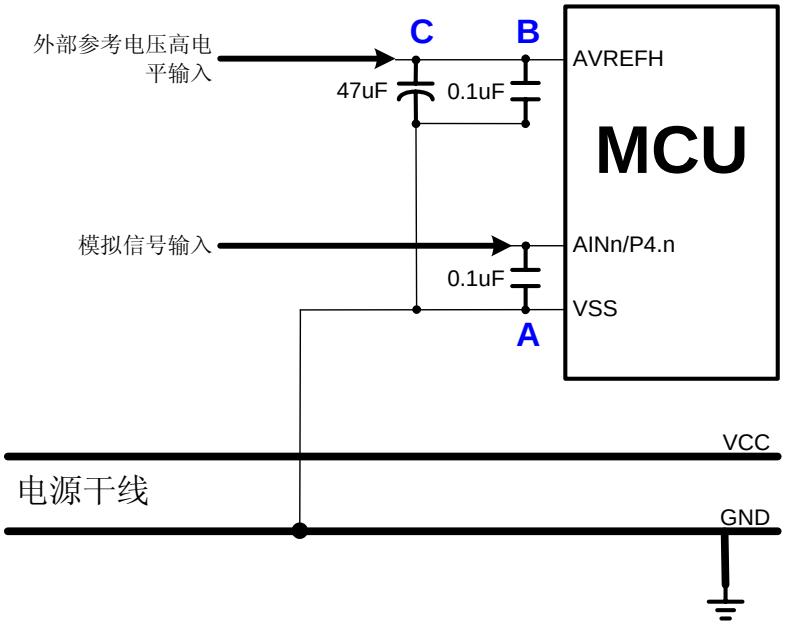
1. **使能 ADENB 后（不是使能 ADS），系统必须延迟 100us 等待启动 ADC，然后设置 ADS 开始 AD 转换，否则 ADC 的结果出错。系统正常运行时，设置 ADENB 一次，延时一次。**
2. **睡眠模式和绿色模式下禁止 ADC 以省电。**

● **ADC 转换：**

```
; 禁止 ADC 中断。
@@:
        B0BTS1      FEOC        ; 检查 ADC 转换标志。
        JMP         @B          ; EOC=0：ADC 转换过程中。
        B0MOV       A, ADB      ; EOC=1：ADC 转换结束，处理 ADC 结果。
        B0MOV       BUF1,A
        MOV         A, #00001111b
        AND         A, ADR
        B0MOV       BUF2,A
        …                       ; ADC 结果处理完成。
        CLR         FEOC        ; 清 ADC 转换标志开始下个 ADC 转换。

; 使能 ADC 中断。
        ORG 8                   ; 中断向量。
INT_SR:                         ; 中断服务程序。
        PUSH
        B0BTS1      FADCIRQ     ; 检查 ADC 中断请求标志。
        JMP         EXIT_INT    ; ADCIRQ=0：无 ADC 中断请求。
        B0MOV       A, ADB      ; ADCIRQ=1：ADC 转换结束，处理 ADC 结果。
        B0MOV       BUF1,A
        MOV         A, #00001111b
        AND         A, ADR
        B0MOV       BUF2,A
        …                       ; ADC 结果处理完成。
        CLR         FEOC        ; 清 ADC 转换标志开始下个 ADC 转换。
        JMP         INT_EXIT

INT_EXIT:
        POP
        RETI                    ; 退出中断。
```

✷ **注：ADC 转换结束后，自动清 ADS。EOC 实时显示 ADC 的转换过程，ADS=1 时被自动清零。用户不需要在程序中清 EOC。**

11.5 ADC 应用电路

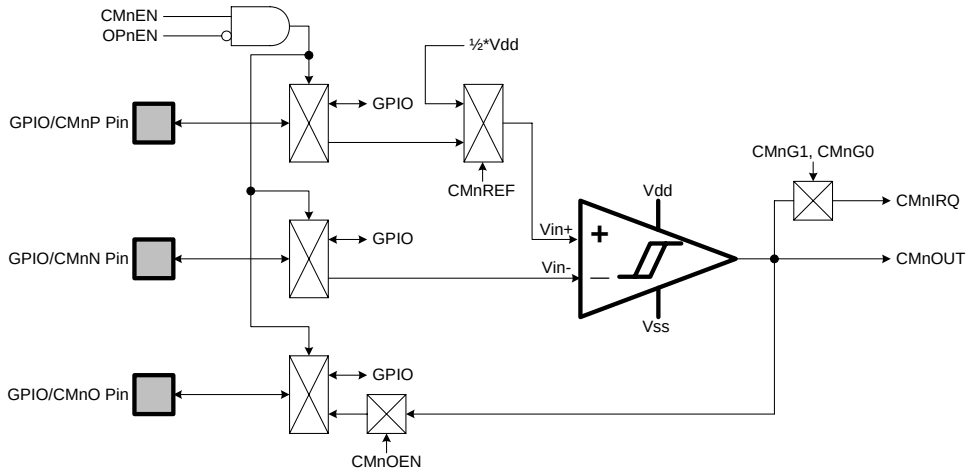


模拟信号从 ADC 输入引脚 AINn/P4.n 输入。在 ADC 输入引脚和 VSS 之间（A）必须连接一个 0.1uF 的电容，且要尽可能的靠近 ADC 输入引脚。不能将电容的 GND 直接连接到电源干线上的 GND，必须通过 VSS 引脚。该电容可以减少电源干扰对模拟信号的影响。

如果 ADC 参考高电压由外部参考源提供，则必须由 AVREFH/P4.0 引脚输入。且在 AVREFH 引脚和 VSS 之间连接电容，首先在图中 C 处连接一个 47uF 的电解电容，再在 B 处连接一个 0.1uF 的电容，且要尽可能的靠近 AVREFH 引脚。不能将电容的 GND 直接连接到电源干线上的 GND，必须通过 VSS 引脚。

12 RAIL TO RAIL 模拟比较器

12.1 概述



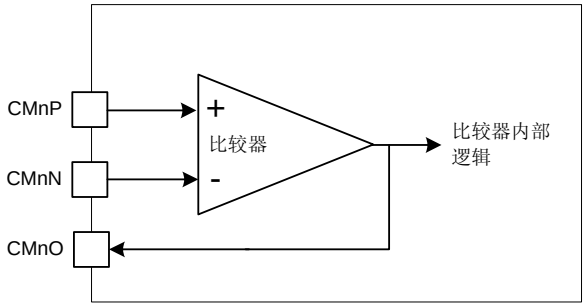
SN8P2735 内置 3 个 Rail-to-Rail 结构的比较器，即输入/输出电压的实际值在 VDD~VSS 之间。当输入电压的正极大于输入电压的负极时，比较器输出高电平；当输入电压的正极小于输入电压的负极时，比较器输出低电平。

CMnOUT 和 CMnIRQ 位都显示比较结果：不同的是 CMnOUT 立即显示出实时值，CMnIRQ 显示均值。均值情况由寄存器和触发沿控制，触发沿包括上升沿（CMnOUT 由低到高），下降沿（CMnOUT 由高到低）和电平变换（CMnOUT 的任何电平变换）。CMnIRQ=1 且 CMnIEN=1（比较器中断使能控制位）时执行比较器中断服务程序。

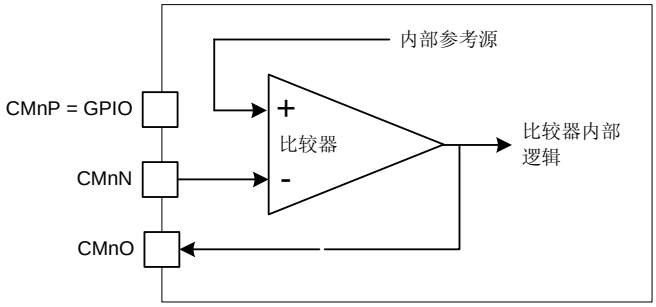
比较器内置内部参考源用来取代比较器的外部正极输入源，由 CMnREF 位控制，内部参考源的电压为 1/2 VDD。当 CMnREF=0 时，比较器的正极输入电压由外部参考源提供，由 CMnP 引脚输入；CMnREF=1 时，比较器的正极输入电压由内部 1/2 VDD 提供，CMnP 引脚为 GPIO 引脚。

✷ **注：CMnOUT 是比较器的自然输出结果，无锁存装置，随着比较结果的改变而改变；CMnIRQ 比较器的锁存输出结果，必须由程序清零**

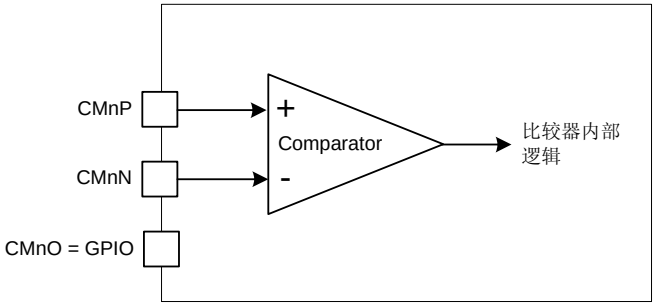
比较器引脚与 GPIO 引脚共用，由 CMnEN 位控制，CMnEN=1 时，CMnN 引脚为比较器的负极输入引脚。CMnOEN 控制比较器的输出引脚是否使能 GPIO 功能，CMnOEN=1 时，比较器的输出端连接至 GPIO 口，并禁止 GPIO 功能。CMnREF 控制比较器的正极输入源是由内部 1/2 VDD 提供还是由外部输入提供，CMnREF=1 时，比较器的正极输入源由内部 1/2 VDD 提供，并使能 CMnP 的 GPIO 功能。比较器引脚的功能列表如下所示：



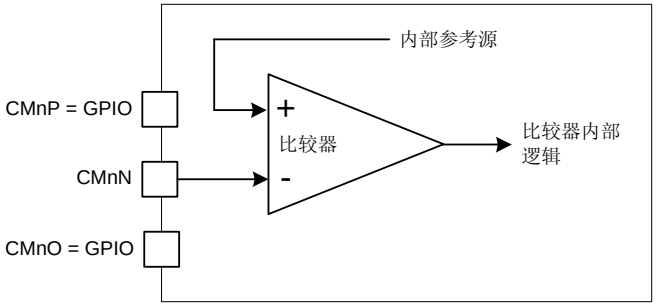
CMnEN = 1, CMnOEN = 1, CMnREF = 0



CMnEN = 1, CMnOEN = 1, CMnREF = 1



CMnEN = 1, CMnOEN = 0, CMnREF = 0



CMnEN = 1, CMnOEN = 0, CMnREF = 1

由于比较器的引脚和 OP 放大器的引脚相同，故比较器的控制信号和 OP 放大器的控制信号需要设定一定的条件以避免同时使能比较器和 OP 放大器。CMnEN 和 OPnEN 相同时（00 或者 11），使能引脚的 GPIO 功能，禁止比较器和 OP 放大器功能；CMnEN=1，OPnEN=0 时，使能比较器功能，禁止 OP 放大器功能；OPnEN=1，CMnEN=0 时，使能 OP 放大器功能，禁止比较器功能。

比较器编号	CMnEN	OPnEN	比较器负极输入引脚	比较器正极输入引脚		比较器输出引脚	
				CMnREF=0	CMnREF=1	CMnOEN=0	CMnOEN=1
CMP0	CM0EN=0	OP0EN=0	使能所有引脚的 GPIO 功能，禁止比较器和 OP 放大器功能。				
		OP0EN=1	使能 OP 放大器功能。				
	CM0EN=1	OP0EN=0	CM0N	CM0P	P1.7 GPIO	P5.0 GPIO	CM0O
		OP0EN=1	使能所有引脚的 GPIO 功能，禁止比较器和 OP 放大器功能。				
CMP1	CM1EN=0	OP1EN=0	使能所有引脚的 GPIO 功能，禁止比较器和 OP 放大器功能。				
		OP1EN=1	使能 OP 放大器功能。				
	CM1EN=1	OP1EN=0	CM1N	CM1P	P1.4 GPIO	P1.5 GPIO	CM1O
		OP1EN=1	使能所有引脚的 GPIO 功能，禁止比较器和 OP 放大器功能。				
CMP2	CM2EN=0	OP2EN=0	使能所有引脚的 GPIO 功能，禁止比较器和 OP 放大器功能。				
		OP2EN=1	使能 OP 放大器功能。				
	CM2EN=1	OP2EN=0	CM2N	CM2P	P1.1 GPIO	P1.2 GPIO	CM2O
		OP2EN=1	使能所有引脚的 GPIO 功能，禁止比较器和 OP 放大器功能。				

✷ **注：比较器的使能条件固定为 CMnEN=1 且 OPnEN=0，否则比较器的引脚为 GPIO 引脚，禁止比较器功能。**

12.2 比较器模式寄存器

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMP0M	CM0EN	CM0IEN	CM0IRQ	CM0OEN	CM0REF	CM0OUT	CM0G1	CM0G0
读/写	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **CM0EN**：比较器 0 控制位。
0 = 禁止，P1.6、P1.7、P5.0 为 GPIO 引脚；
1 = 使能，P1.6 为 CM0N 引脚。

Bit 6 **CM0IEN**：比较器 0 中断功能控制位。
0 = 禁止；
1 = 使能。

Bit 5 **CM0IRQ**：比较器 0 中断请求位。
0 = 无中断请求；
1 = 请求中断。

Bit 4 **CM0OEN**：比较器 0 输出引脚控制位。
0 = 禁止，CM0O 为 P5.0 GPIO 引脚；
1 = 使能，CM0O 为比较器输出引脚，并隔离 P5.0 的 GPIO 功能。

Bit 3 **CM0REF**：比较器内部参考电压源控制位。
0 = 禁止，CM0P 为比较器正极输入引脚，并隔离 P1.7 的 GPIO 功能。
1 = 使能，CM0P 为 P1.7 GPIO 引脚。

Bit 2 **CM0OUT**：比较器 0 输出标志位。
0 = CM0P 电压或比较器内部参考电压小于 CM0N 电压；
1 = CM0P 电压或比较器内部参考电压大于 CM0N 电压。

Bit [1:0] **CM0G[1:0]**：比较器中断触发方向控制位。
00 = 保留；
01 = 上升沿触发，CM0P > CM0N 或比较器内部参考电压；
10 = 下降沿触发，CM0P < CM0N 或比较器内部参考电压；
11 = 电平变换触发。

09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMP1M	CM1EN	CM1IEN	CM1IRQ	CM1OEN	CM1REF	CM1OUT	CM1G1	CM1G0
读/写	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **CM1EN**：比较器 1 控制位。
0 = 禁止，P1.3、P1.4、P1.5 为 GPIO 引脚；
1 = 使能，P1.3 为 CM1N 引脚。

Bit 6 **CM1IEN**：比较器 1 中断功能控制位。
0 = 禁止；
1 = 使能。

Bit 5 **CM1IRQ**：比较器 1 中断请求位。
0 = 无中断请求；
1 = 请求中断。

Bit 4 **CM1OEN**：比较器 1 输出引脚控制位。
0 = 禁止，CM1O 为 P1.5 GPIO 引脚；
1 = 使能，CM1O 为比较器输出引脚，并隔离 P1.5 的 GPIO 功能。

Bit 3 **CM1REF**：比较器内部参考电压源控制位。
0 = 禁止，CM1P 为比较器正极输入引脚，并隔离 P1.4 的 GPIO 功能。
1 = 使能，CM1P 为 P1.4 GPIO 引脚。

Bit 2 **CM1OUT**：比较器 1 输出标志位。
0 = CM1P 电压或比较器内部参考电压小于 CM1N 电压；
1 = CM1P 电压或比较器内部参考电压大于 CM1N 电压。

Bit [1:0] **CM1G[1:0]**：比较器中断触发方向控制位。
00 = 保留；
01 = 上升沿触发，CM1P > CM1N 或比较器内部参考电压；
10 = 下降沿触发，CM1P < CM1N 或比较器内部参考电压；
11 = 电平变换触发。

09EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMP2M	CM2EN	CM2IEN	CM2IRQ	CM2OEN	CM2REF	CM2OUT	CM2G1	CM2G0
读/写	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **CM2EN**：比较器 2 控制位。
0 = 禁止，P1.0、P1.1、P1.2 为 GPIO 引脚；
1 = 使能，P1.0 为 CM2N 引脚。

Bit 6 **CM2IEN**：比较器 2 中断功能控制位。
0 = 禁止；
1 = 使能。

Bit 5 **CM2IRQ**：比较器 2 中断请求位。
0 = 无中断请求；
1 = 请求中断。

Bit 4 **CM2OEN**：比较器 2 输出引脚控制位。
0 = 禁止，CM2O 为 P1.2 GPIO 引脚；
1 = 使能，CM2O 为比较器输出引脚，并隔离 P1.2 的 GPIO 功能。

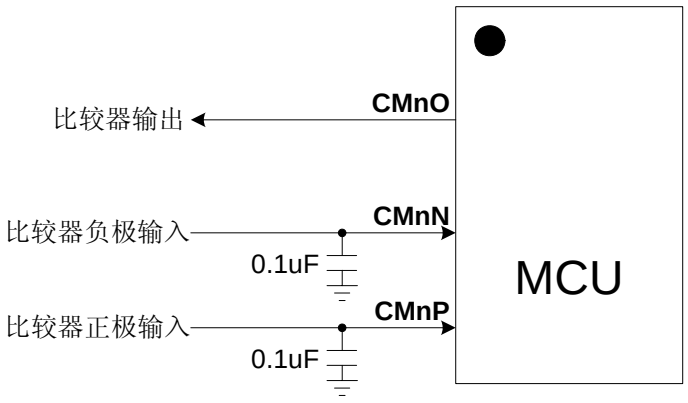
Bit 3 **CM2REF**：比较器内部参考电压源控制位。
0 = 禁止，CM2P 为比较器正极输入引脚，并隔离 P1.1 的 GPIO 功能。
1 = 使能，CM2P 为 P1.1 GPIO 引脚。

Bit 2 **CM2OUT**：比较器 2 输出标志位。
0 = CM2P 电压或比较器内部参考电压小于 CM2N 电压；
1 = CM2P 电压或比较器内部参考电压大于 CM2N 电压。

Bit [1:0] **CM2G[1:0]**：比较器中断触发方向控制位。
00 = 保留；
01 = 上升沿触发，CM2P > CM2N 或比较器内部参考电压；
10 = 下降沿触发，CM2P < CM2N 或比较器内部参考电压；
11 = 电平变换触发。

12.3 比较器应用注意事项

比较器是指比较正极电压和负极电压并将结果输出，正负极的信号都是模拟信号。在硬件应用电路中，比较器的输入引脚必须连接一个 0.1uF 的电容以减少电源干扰，确保输入信号的稳定。应用电路如下所示：

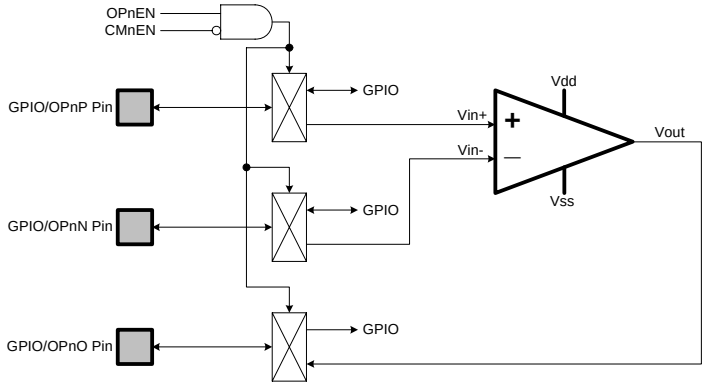


➢ **例：使用比较器 0 测量外部模拟信号，当模拟信号小于 1/2 VDD 时，执行中断服务程序。在这个演示程序中，选用比较器 1/2 VDD 作为比较器的正极输入电压源，中断触发条件为上升沿触发。**

```
; 初始化比较器 0。
        MOV     A, #01001001b   ; CM0REF=1，使能比较器内部 1/2VDD 参考电压作为
        B0MOV   CM0M, A         ; 比较器的正极输入源。
                                ; CM0G1, CM0G0=01，设置比较器的中断请求为上升沿触发。
                                ; CM0IEN=1，使能比较器 0 的中断功能。
                                ; CM0IRQ=0，清除比较器 0 的中断请求标志。

        B0BSET  FCM0EN          ; 使能比较器 0。

Main:                           ; 主循环。
        …
        …
        JMP     MAIN
        …
        …

; 中断服务程序。跳到中断向量（0008H）。
ISR:
        PUSH                    ; 保存 ACC 和 PFLAG。
        B0BTS1  FCM0IRQ         ; 检查是否有比较器 0 中断请求。
        JMP     ISR_EXIT        ; 无中断请求，退出中断。

        B0BCLR  FCM0IRQ         ; 清除比较器 0 的中断请求标志。
        …                       ; 执行中断。
        …
        …
        JMP     ISR_EXIT        ; 中断服务程序结束。

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

13 RAIL to RAIL OP 放大器

13.1 概述



SN8P2735 内置 3 个 Rail-to-Rail 结构的 OP 放大器，即输入/输出电压的实际值在 VDD~VSS 之间。

OP 放大器的引脚与 GPIO 引脚共用，由 OPnEN 位控制。OPnEN=1 时，GPIO 引脚切换到 OP 放大器引脚并隔离 GPIO 功能。OP 引脚功能的选择列表如下所示：

由于比较器的引脚和 OP 放大器的引脚相同，故比较器的控制信号和 OP 放大器的控制信号需要设定一定的条件以避免同时使能比较器和 OP 放大器。CMnEN 和 OPnEN 相同时（00 或者 11），使能引脚的 GPIO 功能，禁止比较器和 OP 放大器功能；CMnEN=1，OPnEN=0 时，使能比较器功能，禁止 OP 放大器功能；OPnEN=1，CMnEN=0 时，使能 OP 放大器功能，禁止比较器功能。

OP 编号	OPnEN	CMnEN	OP 正极输入引脚	OP 负极输入引脚	OP 输出引脚
OP0	OP0EN=0	CM0EN=0	使能所有引脚的 GPIO 功能，禁止比较器和 OP 放大器功能。		
		CM0EN=1	使能比较器功能。		
	OP0EN=1	CM0EN=0	OP0P (Vin+)	OP0N (Vin-)	OP0O (Vout)
		CM0EN=1	使能所有引脚的 GPIO 功能，禁止比较器和 OP 放大器功能。		
OP1	OP1EN=0	CM1EN=0	使能所有引脚的 GPIO 功能，禁止比较器和 OP 放大器功能。		
		CM1EN=1	使能比较器功能。		
	OP1EN=1	CM1EN=0	OP1P (Vin+)	OP1N (Vin-)	OP1O (Vout)
		CM1EN=1	使能所有引脚的 GPIO 功能，禁止比较器和 OP 放大器功能。		
OP2	OP2EN=0	CM2EN=0	使能所有引脚的 GPIO 功能，禁止比较器和 OP 放大器功能。		
		CM2EN=1	使能比较器功能。		
	OP2EN=1	CM2EN=0	OP2P (Vin+)	OP2N (Vin-)	OP2O (Vout)
		CM2EN=1	使能所有引脚的 GPIO 功能，禁止比较器和 OP 放大器功能。		

✷ **注：OP 放大器的使能条件固定为 OPnEN=1，且 CMnEN=0，否则 OP 放大器的引脚为 GPIO 引脚并禁止比较器功能。**

13.2 OP AMP 寄存器

09FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPM	-	-	-	-	-	OP2EN	OP1EN	OP0EN
读/写	-	-	-	-	-	R/W	R/W	R/W
复位后	-	-	-	-	-	0	0	0

Bit 2 **OP2EN**：OP Amp 2 控制位。
0 = 禁止，P1.0~P1.2 为 GPIO 引脚和比较器引脚；
1 = 使能，P1.0~P1.2 为 OP Amp 引脚。

Bit 1 **OP1EN**：OP Amp 1 控制位。
0 = 禁止，P1.3~P1.5 为 GPIO 引脚和比较器引脚；
1 = 使能，P1.3~P1.5 为 OP Amp 引脚。

Bit 0 **OP0EN**：OP Amp 0 控制位。
0 = 禁止，P1.6、P1.7 和 P5.0 为 GPIO 引脚和比较器引脚；
1 = 使能，P1.6、P1.7 和 P5.0 为 OP Amp 引脚。

14 指令集

指令	指令格式		指令说明	C	DC	Z	周期
	MOV	A,M	A ← M	-	-	√	1
M	MOV	M,A	M ← A	-	-	-	1
O	B0MOV	A,M	A ← M (bnak 0)	-	-	√	1
V	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
E	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I，（M 指工作寄存器 R、Y、Z、RBANK 和 PFLAG 等。）	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
	ADC	A,M	A ← A + M + C，如果产生进位，则 C=1，否则 C=0。	√	√	√	1
A	ADC	M,A	M ← A + M + C，如果产生进位，则 C=1，否则 C=0。	√	√	√	1+N
R	ADD	A,M	A ← A + M，如果产生进位，则 C=1，否则 C=0。	√	√	√	1
I	ADD	M,A	M ← A + M，如果产生进位，则 C=1，否则 C=0。	√	√	√	1+N
T	B0ADD	M,A	M (bank 0) ← M (bank 0) + A，如果产生进位，则 C=1，否则 C=0。	√	√	√	1+N
H	ADD	A,I	A ← A + I，如果产生进位，则 C=1，否则 C=0。	√	√	√	1
M	SBC	A,M	A ← A - M - /C，如果产生借位，则 C=0，否则 C=1。	√	√	√	1
E	SBC	M,A	M ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1+N
T	SUB	A,M	A ← A - M，如果产生借位，则 C=0，否则 C=1。	√	√	√	1
I	SUB	M,A	M ← A - M，如果产生借位，则 C=0，否则 C=1。	√	√	√	1+N
C	SUB	A,I	A ← A - I，如果产生借位，则 C=0，否则 C=1。	√	√	√	1
	DAA		将 ACC 中的数据由十六进制转换成十进制格式。	√	-	-	1
	MUL	A,M	R, A ← A * M，结果的 LB 存入 ACC，HB 存入 R 寄存器，ZF 受 ACC 的影响。	-	-	√	2
	AND	A,M	A ← A 与 M	-	-	√	1
L	AND	M,A	M ← A 与 M	-	-	√	1+N
O	AND	A,I	A ← A 与 I	-	-	√	1
G	OR	A,M	A ← A 或 M	-	-	√	1
I	OR	M,A	M ← A 或 M	-	-	√	1+N
C	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
	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
P	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
R	RRC	M	A ← RRC M	√	-	-	1
O	RRCM	M	M ← RRC M	√	-	-	1+N
C	RLC	M	A ← RLC M	√	-	-	1
E	RLCM	M	M ← RLC M	√	-	-	1+N
S	CLR	M	M ← 0	-	-	-	1
S	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
	CMPRS	A,I	ZF,C ← A – I，如果 A = I，跳转到下一条指令。	√	-	√	1 + S
B	CMPRS	A,M	ZF,C ← A – M，如果 A = M，跳转到下一条指令。	√	-	√	1 + S
R	INCS	M	A ← M + 1，如果 A = 0，跳转到下一条指令。	-	-	-	1+ S
A	INCMS	M	M ← M + 1，如果 M = 0，跳转到下一条指令。	-	-	-	1+N+S
N	DECS	M	A ← M – 1，如果 A = 0，跳转到下一条指令。	-	-	-	1+ S
C	DECMS	M	M ← M – 1，如果 M = 0，跳转到下一条指令。	-	-	-	1+N+S
H	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
M	RET		子程序跳出指令，PC ← Stack	-	-	-	2
I	RETI		中断程序跳出指令，PC ← Stack，使能全局中断。	-	-	-	2
S	PUSH		保存工作寄存器。	-	-	-	1
C	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。

15 电气特性

15.1 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)
SN8P2735P, SN8P2735F..... 0°C ~ + 70°C
SN8P2735PD, SN8P2735FD..... –40°C ~ + 85°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

15.2 电气特性

- **DC CHARACTERISTIC**

(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 4MHz,Fcpu=1MHz,ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd, *25℃, Fpu = 1MHz*		1.8	-	5.5	V
		Normal mode, Vpp = Vdd, *-40℃~85℃*		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	IoL	Vop = Vss + 0.5V		8	-	-	
*INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current (Disable ADC, OP-amp, Comparator)	Idd1	Run Mode (No loading)	Vdd= 3V, Fcpu = 8MHz	-	5	-	mA
			Vdd= 5V, Fcpu = 8MHz	-	7	-	mA
			Vdd= 3V, Fcpu = 4MHz	-	2	-	mA
			Vdd= 5V, Fcpu = 4MHz	-	4	-	mA
			Vdd= 3V, Fcpu = 1MHz	-	1.5	-	mA
			Vdd= 5V, Fcpu = 1MHz	-	3	-	mA
			Vdd= 3V, Fcpu = 32KHz	-	20	-	uA
			Vdd= 5V, Fcpu = 32KHz	-	45	-	uA
	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= 5V, IHRC 16MHz	-	0.35	-	mA
			Vdd= 3V, IHRC 16MHz	-	0.55	-	mA
			Vdd= 3V, Ext. 32KHz X'tal	-	6	-	uA
			Vdd= 5V, Ext. 32KHz X'tal	-	18	-	uA
			Vdd= 3V, ILRC=16KHz	-	3	-	uA
			Vdd= 5V, ILRC=32KHz	-	5.5	-	uA
Internal High Oscillator Freq.	Fihrc	Internal Hihg RC (IHRC)	*25℃*, Vdd=2.2V~ 5.5V Fcpu=Fhosc/2~Fhosc/16	15.68	16	16.32	MHz
			-40℃~85℃,Vdd=2.4V~ 5.5V Fcpu=Fhosc/2~Fhosc/16	15.2	16	16.8	MHz
LVD Voltage	Vdet0	Low voltage reset level. *25℃*		1.9	2.0	2.1	V
		Low voltage reset level. *-40℃~85℃*		1.8	2.0	2.3	V
	Vdet1	Low voltage reset/indicator level. *25℃*		2.3	2.4	2.5	V
		Low voltage reset/indicator level. *-40℃~85℃*		2.2	2.4	2.7	V
	Vdet2	Low voltage reset/indicator level. *25℃*		3.5	3.6	3.7	V
		Low voltage reset/indicator level. *-40℃~85℃*		3.3	3.6	3.9	V

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

ADC CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 4MHz,Fcpu=1MHz,ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
AIN0 ~ AIN7 input voltage	Vani	Vdd = 5.0V	0	-	Avrefh	V
ADC reference Voltage	Vref		2	-	-	V
*ADC enable time	Tast	Ready to start convert after set ADENB = "1"	100	-	-	us
*ADC current consumption	I_{ADC}	Vdd=5.0V	-	0.6	-	mA
		Vdd=3.0V	-	0.4	-	mA
ADC Clock Frequency	F_{ADCLK}	VDD=5.0V	-	-	8M	Hz
		VDD=3.0V	-	-	5M	Hz
ADC Conversion Cycle Time	F_{ADCYL}	VDD=2.4V~5.5V	64	-	-	$1/F_{ADCLK}$
ADC Sampling Rate (Set FADS=1 Frequency)	F_{ADSMP}	VDD=5.0V	-	-	125	K/sec
		VDD=3.0V	-	-	80	K/sec
Differential Nonlinearity	DNL	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K	±1	-	-	LSB
Integral Nonlinearity	INL	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K	±2	-	-	LSB
No Missing Code	NMC	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K	10	11	12	Bits
ADC offset Voltage	$V_{ADCoffset}$	Non-trimmed	-10	0	+10	mV
		Trimmed	-2	0	+2	mV

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

OP AMP CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 4MHz,Fcpu=1MHz,ambient temperature is 25℃ unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
*Supply Current	Iop	Iout=0, Vdd= 3V	-	130	-	uA
		Iout=0, Vdd= 5V	-	150	-	uA
*Quiescent Current	Iq	OP-amp disable.	-	-	1	uA
Common Mode Input Voltage Range	Vcmr	Vdd=5.0V	Vss-0.3	-	Vdd+0.3	V
Input Offset Voltage	Vos	Vcm=Vss	-3	-	+3	mV
Power Supply Rejection Ratio	PSRR	Vcm=Vss	50	-	70	dB
Common Mode Rejection Ratio	CMRR	Vcm=-0.3V~2.5V. Vdd=5V.	50	-	80	dB
Open-Loop Gain (Large Signal)	Aol	Vout=0.2V~Vdd-0.2V. Vcm=Vss.	90	110	-	dB
Maximum Output Voltage Swing	Vol, Voh	0.5V output overdrive.	Vss+15		Vdd-15	V
Output Short Current	Isc	Unit Gain Buffer. Vo = Vss~Vdd, Vdd=3V	-15	-	+15	mA
		Unit Gain Buffer. Vo = Vss~Vdd, Vdd=5V	-40	-	+40	mA
Output Slew Rate	Tosr	Vo = Vss to Vdd, Vdd to Vss.	3	-	5	us

COMPARATOR CHARACTERISTIC

(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 4MHz,Fcpu=1MHz,ambient temperature is 25℃ unless otherwise note.)

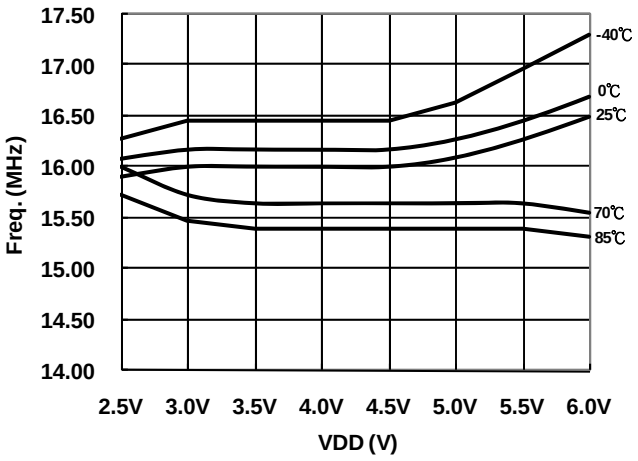
PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
*Supply Current	Icm	Iout=0, Vdd= 3V	-	130	-	uA
		Iout=0, Vdd= 5V	-	150	-	uA
*Quiescent Current	Iq	OP-amp disable.	-	-	1	uA
Input Offset Voltage	Vos	Vcm=Vss	-3		+3	mV
Common Mode Input Voltage Range	Vcmr	Vdd=5.0V	Vss-0.3		Vdd+0.3	V
Output Slew Rate	Tosr	Vo = Vss to Vdd, Vdd to Vss.	1	-	2	us

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

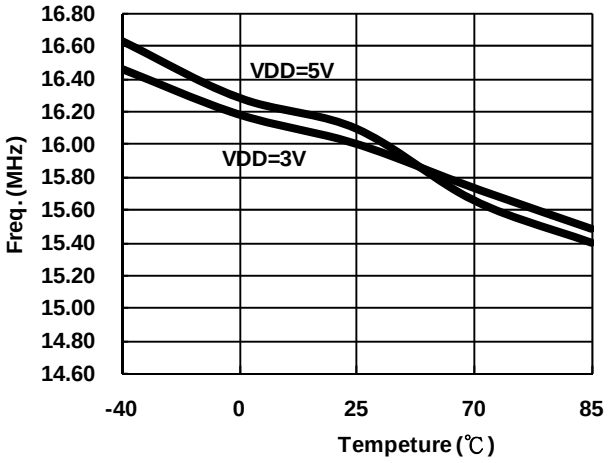
15.3 特性曲线

The Graphs in this section are for design guidance, not tested or guaranteed. In some graphs, the data presented are outside specified operating range. This is for information only and devices are guaranteed to operate properly only within the specified range.

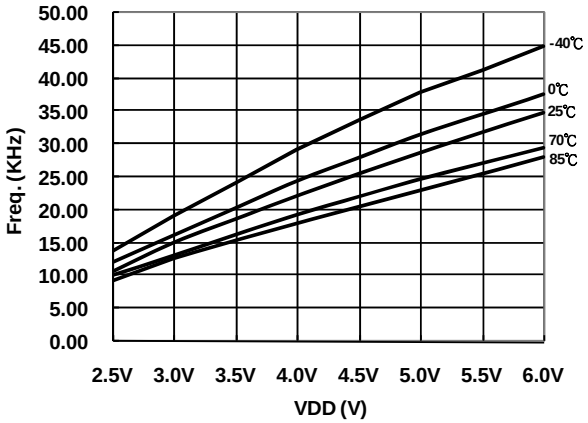
Internal High RC Oscillator (MHz)
(Fcpu = IHRC/2~IHRC/16)



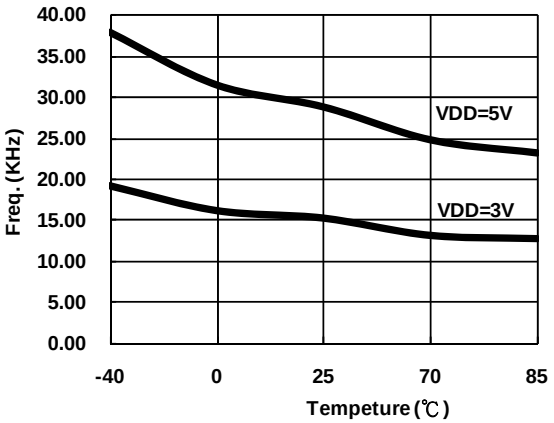
Internal High RC Oscillator (MHz)
(Fcpu=IHRC/2~IHRC/16)



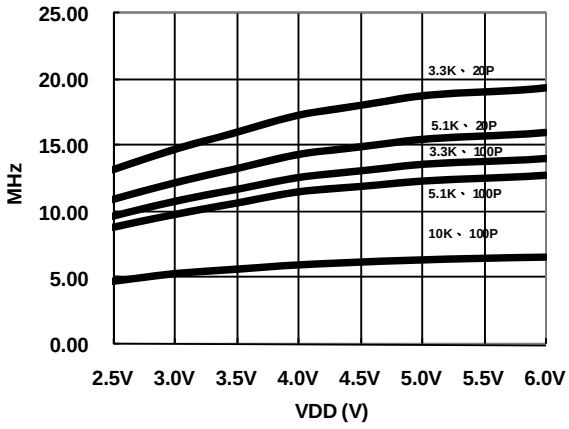
Internal Low RC Oscillator (KHz)



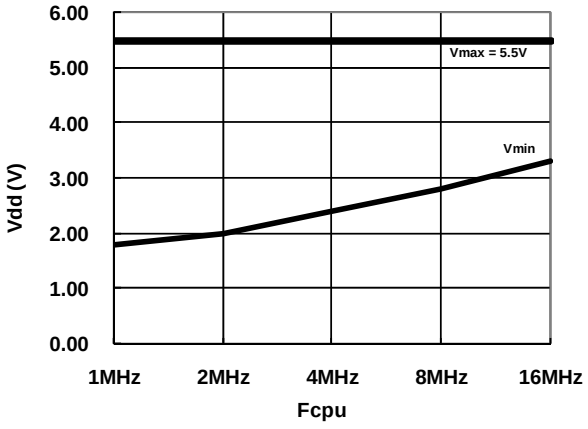
Internal Low RC Oscillator (KHz)



External RC Oscillator (25℃)



System Minmum Operating Voltage



16 开发工具

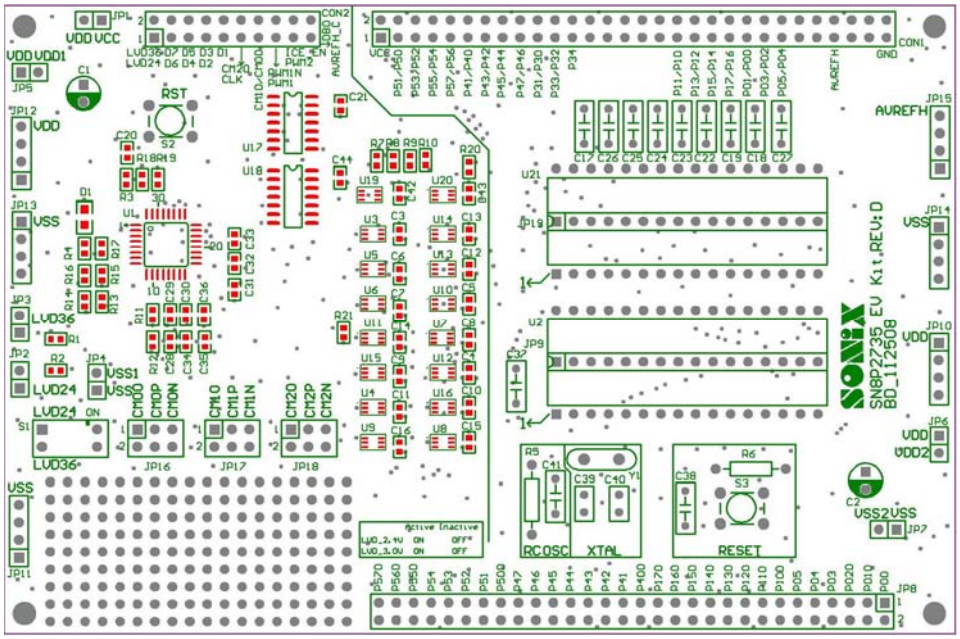
在进行 SN8P2735 的开发时，SONIX 提供 ICE（在线仿真器），IDE（集成开发环境）和 EV-Kit 开发工具。ICE 和 EV-Kit 为外部硬件装置，IDE 有一个友好的用户界面进行韧件开发与仿真。各工具的版本如下所示：

- **ICE：SN8ICE2K Plus 1，SN8ICE2K Plus 2。（在进行 IHRC_16M 和 IHRC_RTC 仿真时，请选择 16MHz 的晶振。）**
- **EV-kit：SN8P2735_EV kit Rev. D。**
- **IDE：SONiX IDE M2IDE_V119 或更晚的版本。**
- **Writer：MPIII writer。**

16.1 SN8P2735 EV-KIT

SONIX 提供 SN8P2735 EV-Kit 来仿真 SN8P2735 的所有功能，包括多路 PWM，ADC，比较器和 OP 模拟功能，并能进行 LVD2.4V/3.6V 的电路切换。

SN8P2735 EV-Kit PCB 如下所示：

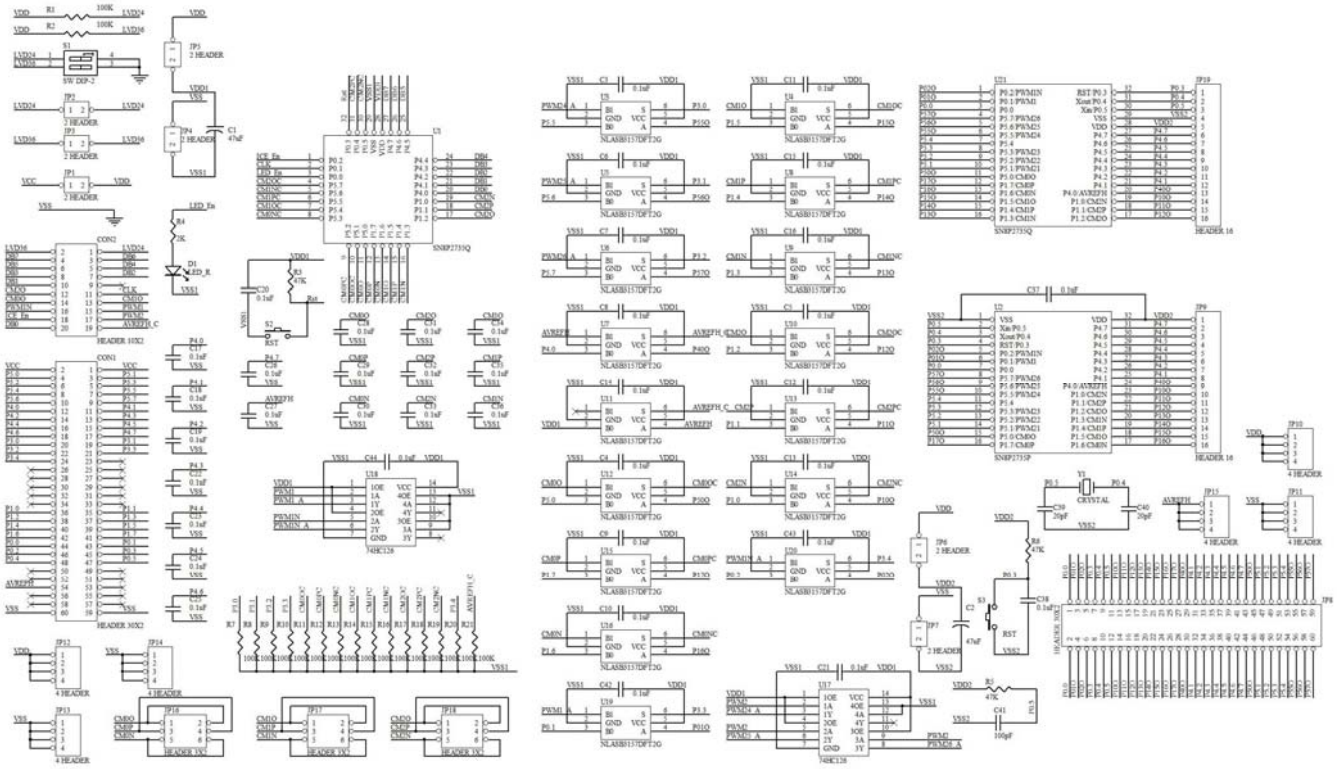


- CON1：连接到 SN8ICE2K Plus CON1（包括 GPIO、EV-Kit 控制信号等）。
- CON2：连接到 SN8ICE2K Plus JP3（ICE 与 EV-Kit 的通讯总线，控制信号等）。
- S1：LVD24V / LVD36V 控制开关，可以仿真 LVD2.4V 标志/ 复位功能和 LVD3.6V / 标志功能。

开关变换	ON	OFF
LVD24	LVD 2.4V 有效	LVD 2.4V 无效
LVD36	LVD 3.6V 有效	LVD 3.6V 无效

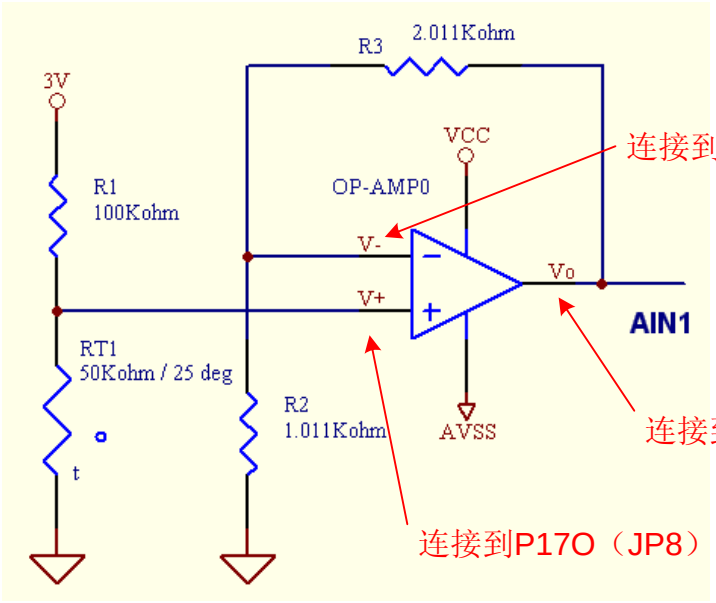
- JP8：GPIO 接口。
- U1：SN8P2735 EV 芯片，用来仿真模拟功能。
- U2：SN8P2735 DIP 封装形式的 IC 接口，用来连接用户目标板。
- U21：SN8P2735 LQFP 封装形式的 IC 接口，用来连接用户目标板。
- JP15：使用 ADC 功能之前，必须将 SN8ICE2K Plus 上的 AVREFH/VDD 的跳线断开，否则会使能 ADC 的外部参考源，JP15（AVREFH）或者 P40 作为外部参考源输入端。
- C17~C19,C22~C26：0.1uF 的旁路电容，连接到 AIN0~AIN7 引脚处。
- C27：0.1uF 的旁路电容，连接到 AVREFH 输入引脚处。
- S2：复位按键。如果 EV-Kit 不能工作，请按 S2 复位 EV-Kit 上的 EV 芯片（U1）。
- JP16 ~ JP18：观察 CMP0~CMP2 / OP-Amp0~OP-Amp2 输入/输出电压。
- C28~C36：比较器或者 OP 放大器的旁路电容。

SN8P2735 EV-kit 的原理图：



16.2 ICE 和 EV-KIT 应用注意事项

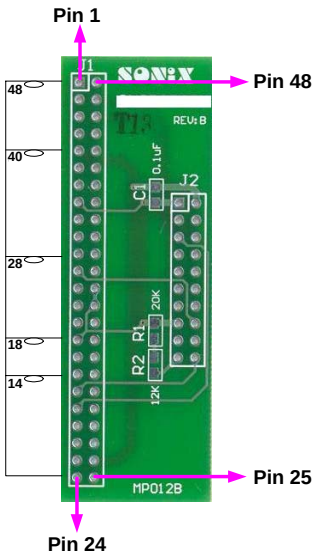
1. 连接 SN8P2735 EV-Kit 到 SN8ICE2K Plus 之前，必须将 SN8ICE2K Plus 上的电源开关关闭。
2. 将 EV-KIT 上的 CON1/CON2 连接到 ICE 上的 CON1/JP3。
3. 必须将 SN8ICE2K Plus 上的 AVREFH/VDD 跳线断开。
4. 用户完成 1、2、3 之后，将 SN8ICE2K Plus 上的电源开关打开。
5. **在 ICE 上仿真 IHRC_16M 功能时，必须连接 16MHz 的外部晶振。**
6. 使能 ADC 功能时，ADM 的 bit3（FAVREFH）置高时，P40O 或 JP15 作为外部参考电压输入引脚。
7. 使能 ADC 功能时，ADM 的 bit3（FAVREFH）置低时，P40O 作为模拟信号输入引脚，JP15（AVREFH）不能连接任何电源设备。
8. 在 JP15（AVREFH）观察 ADC 内部或外部参考电压。
9. SN8P2735 OP 放大器 0 的应用电路如下：



10. 如上图：使能 SN8P2735 的 OP-AMP0 功能时，P50O（JP8）是 OP-AMP0 的输出端，P17O（JP8）是 AMP0 的正极输入引脚，P16O（JP8）是 AMP0 的负极输入引脚。
11. 如上图：如果用户需测量 OP-AMP0 V+/V-/Vo 的电压，OP-AMP 的负极输入电压是 CM0N（JP16），正极输入电压是 CM0P（JP16），输出电压是 CM0O（JP16）。
12. 如果用户想用 OP-AMP1 替换上图中的 OP-AMP0，P15O（JP8）是 OP-AMP1 的输出端，P14O（JP8）是 OP-AMP1 的正极输入端，P13O（JP8）是 OP-AMP1 的负极输入端。
13. 如上图：如果用户需测量 OP-AMP1 V+/V-/Vo 的电压，OP-AMP 的负极输入电压是 CM1N（JP17），正极输入电压是 CM1P（JP17），输出电压是 CM1O（JP17）。
14. 如果用户想用 OP-AMP2 替换上图中的 OP-AMP0，P12O（JP8）是 OP-AMP2 的输出端，P11O（JP8）是 OP-AMP2 的正极输入端，P10O（JP8）是 OP-AMP2 的负极输入端。
15. 如上图：如果用户需测量 OP-AMP2 V+/V-/Vo 的电压，OP-AMP 的负极输入电压是 CM2N（JP18），正极输入电压是 CM2P（JP18），输出电压是 CM2O（JP18）。
16. 由于 OP-AMP 系列连接不同的模拟转换内部电阻（Ron），故测量时连接不同的接口。

17 OTP 烧录引脚

17.1 烧录器转接板引脚配置



JP3（配置 48-Pin 锁紧）

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

17.2 烧录引脚配置

SN8P2735 烧录引脚信息							
单片机名称		SN8P2735P（DIP）			SN8P2735F（LQFP）		
Writer 接口		IC 和 JP3 48-pin 锁紧引脚分频					
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	32	VDD	40	28	VDD	36
2	GND	1	VSS	9	29	VSS	37
3	CLK	24	P4.0	32	20	P4.0	28
4	CE	-	-	-	-	-	-
5	PGM	28	P4.4	36	24	P4.4	32
6	OE	25	P4.1	33	21	P4.1	29
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP	4	RST	12	32	RST	40
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	3	P0.4	11	31	P0.4	39

18 单片机正印命名规则

18.1 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。

18.2 单片机型号说明

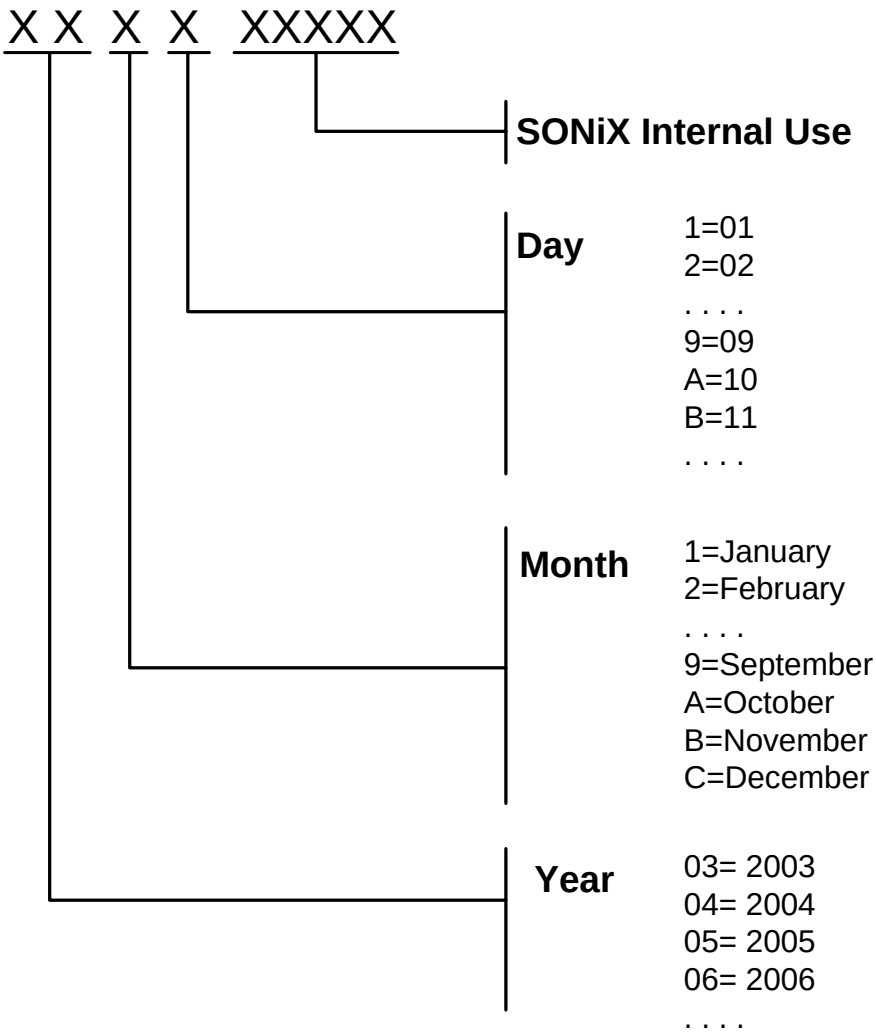
SN8 X PART No. X X X

Material	B = PB-Free Package G = Green Package
Temperature Range	- = 0℃ ~ 70℃ D = -40℃ ~ 85℃
Shipping Package	W = Wafer H = Dice P = P-DIP S = SOP X = SSOP
Device	2711
ROM Type	P=OTP
Title	SONiX 8-bit MCU Production

18.3 命名举例

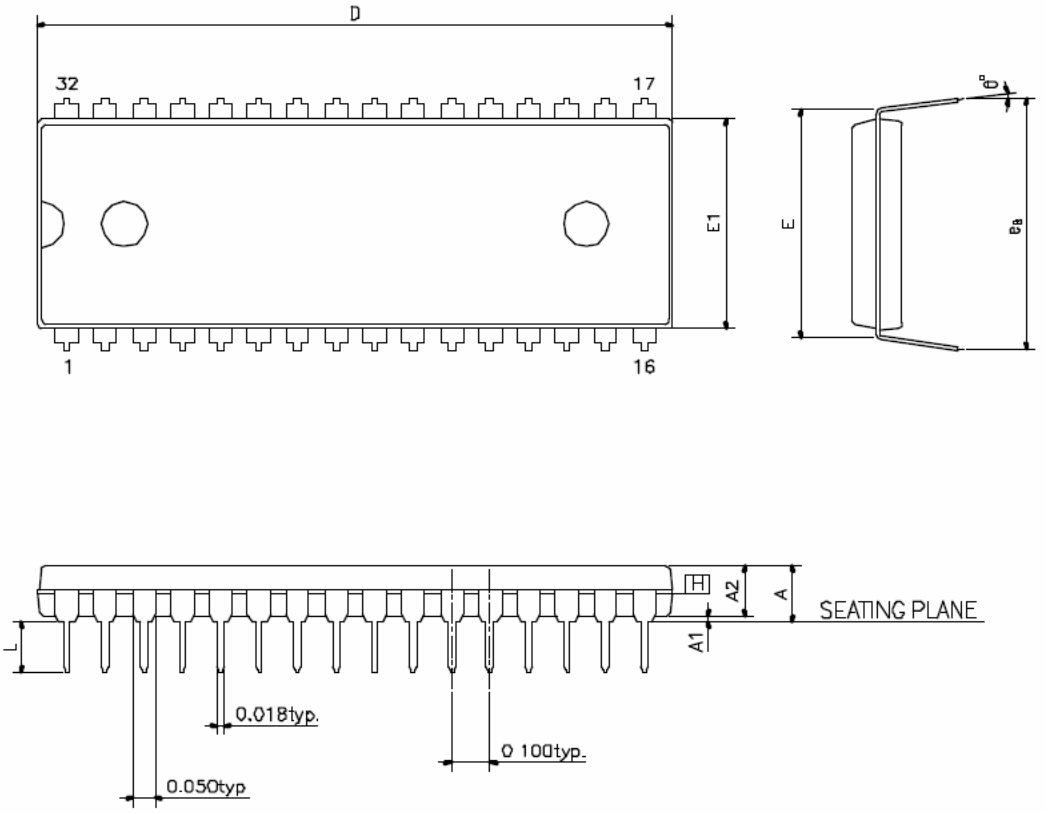
单片机名称	ROM 类型	器件（Device）	封装形式	温度范围	封装材料
SN8P2735PB	OTP	2735	P-DIP	0℃~70℃	无铅封装
SN8P2735FB	OTP	2735	LQFP	0℃~70℃	无铅封装
SN8P2735PG	OTP	2735	P-DIP	0℃~70℃	绿色封装
SN8P2735FG	OTP	2735	LQFP	0℃~70℃	绿色封装
SN8P2735PDB	OTP	2735	P-DIP	-40℃~85℃	无铅封装
SN8P2735FDB	OTP	2735	LQFP	-40℃~85℃	无铅封装
SN8P2735PDG	OTP	2735	P-DIP	-40℃~85℃	绿色封装
SN8P2735FDG	OTP	2735	LQFP	-40℃~85℃	绿色封装
SN8P2735W	OTP	2735	Wafer	0℃~70℃	-
SN8P2735H	OTP	2735	Dice	0℃~70℃	-

18.4 日期码规则



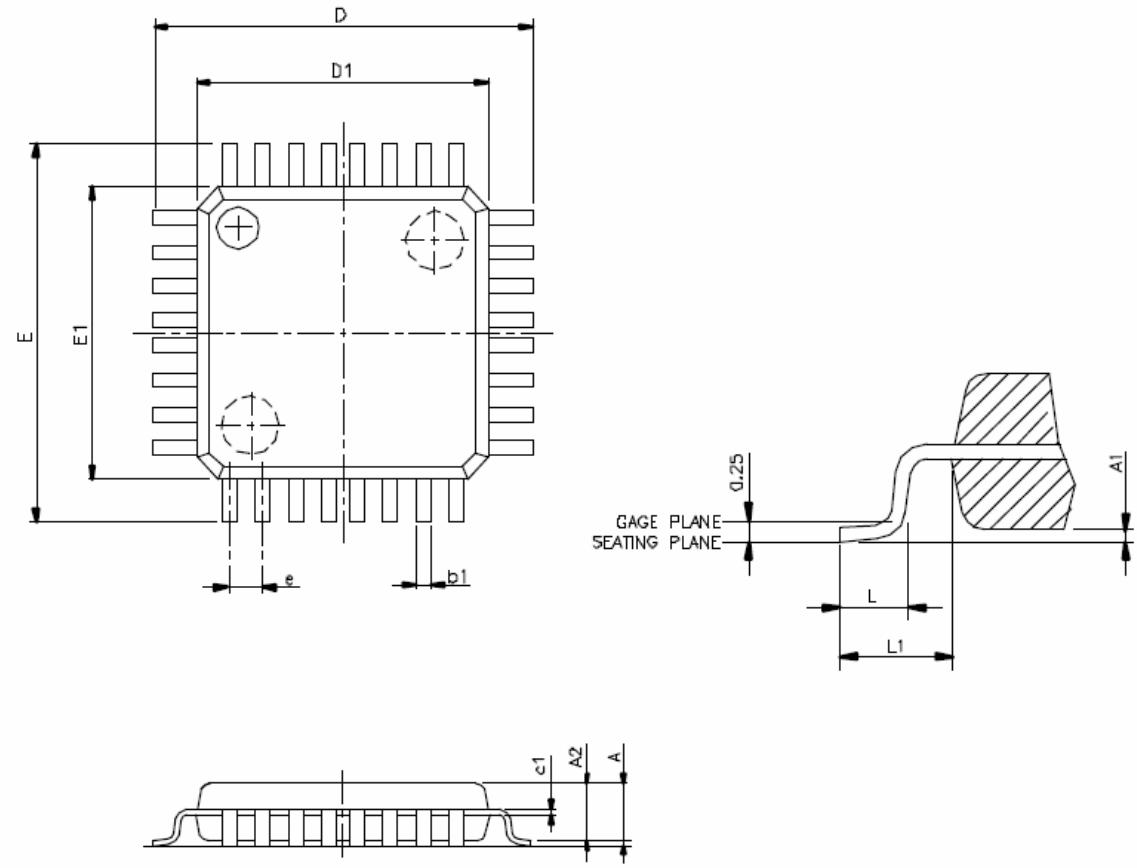
19 封裝信息

19.1 P-DIP 32 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.220	-	-	5.588
A1	0.015	-	-	0.381	-	-
A2	0.150	0.155	0.160	3.81	3.937	4.064
D	1.645	1.650	1.660	41.783	41.91	42.164
E	0.600 BSC			15.24 BSC		
E1	0.540	0.545	0.550	13.716	13.843	13.97
L	0.115	0.130	0.150	2.921	3.302	3.81
e_B	0.630	0.650	0.670	16.002	16.51	17.018
θ°	0°	7°	15°	0°	7°	15°

19.2 LQFP 32 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.063	-	-	1.6
A1	0.002	0.004	0.006	0.05	0.1	0.15
A2	0.053	0.055	0.057	1.35	1.4	1.45
c1	0.004	0.005	0.006	0.09	0.125	0.16
D	0.354 BSC			9 BSC		
D1	0.276 BSC			7 BSC		
BSC E	0.354 BSC			9 BSC		
E1	0.276 BSC			7 BSC		
e	0.031 BSC			0.8 BSC		
b	0.012	0.015	0.018	0.3	0.375	0.45
L	0.018	0.024	0.030	0.45	0.6	0.75
L1	0.039 REF			1 REF		

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：香港新界沙田沙田乡宁会路 138＃新城市中央广场第一座 7 楼 705 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