SN8P2711B

用户参考手册

SONiX 8 位单片机

修正记录

版本号	日期	内容
VER 1.0	2012 年 11 月	初版。

目　　录

1 产品简介

1.1 功能特性

- **存储器配置**
  OTP ROM 空间：1K * 16 位。
  RAM 空间：64 字节。

- **4 层堆栈缓存器**

- **I/O 引脚配置**
  输入输出双向端口：P0、P4、P5。
  单向输入引脚：P0.4。
  具有唤醒功能的端口：P0 电平触发。
  内置上拉电阻端口：P0、P4、P5。
  ADC 输入硬件：P4.0~P4.4。
  外部中断引脚：
  P0.0：由寄存器 PEDGE 控制；
  P0.1：下降沿触发。

- **3 级低电压检测系统（LVD）**
  系统复位，监控系统电源。

- **5 个中断源**
  3 个内部中断：TC0、TC1、ADC。
  2 个外部中断：INT0、INT1。

- **强大的指令系统**
  指令的长度为一个字。
  大部分指令只需要一个时钟周期。
  跳转指令 JMP 可在整个 ROM 区执行。
  调用指令 CALL 可在整个 ROM 区执行。
  查表指令 MOVC 可寻址整个 ROM 区。

- **5+1 通道 12 位 SAR ADC**
  5 个外部 ADC 输入。
  一个内部电池检测。
  内部 AD 参考电压（VDD、4V、3V、2V）。

- **两个 8 位定时/计数器**
  TC0：外部事件计数器/PWM0/ Buzzer 输出。
  TC1：外部事件计数器/PWM1/ Buzzer 输出。

- **内置看门狗定时器，其时钟源由内部低速 RC 振荡器提供（16KHz @3V，32KHz @5V）**

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

- **工作模式**
  普通模式：高、低速时钟同时工作。
  低速模式：只有低速时钟工作。
  睡眠模式：高、低速时钟都停止工作。
  绿色模式：由 TC0 周期性的唤醒。

- **封装形式**
  P-DIP 14 pins。
  SOP 14 pins。
  MSOP 10 pins。

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

☞ **特性列表**

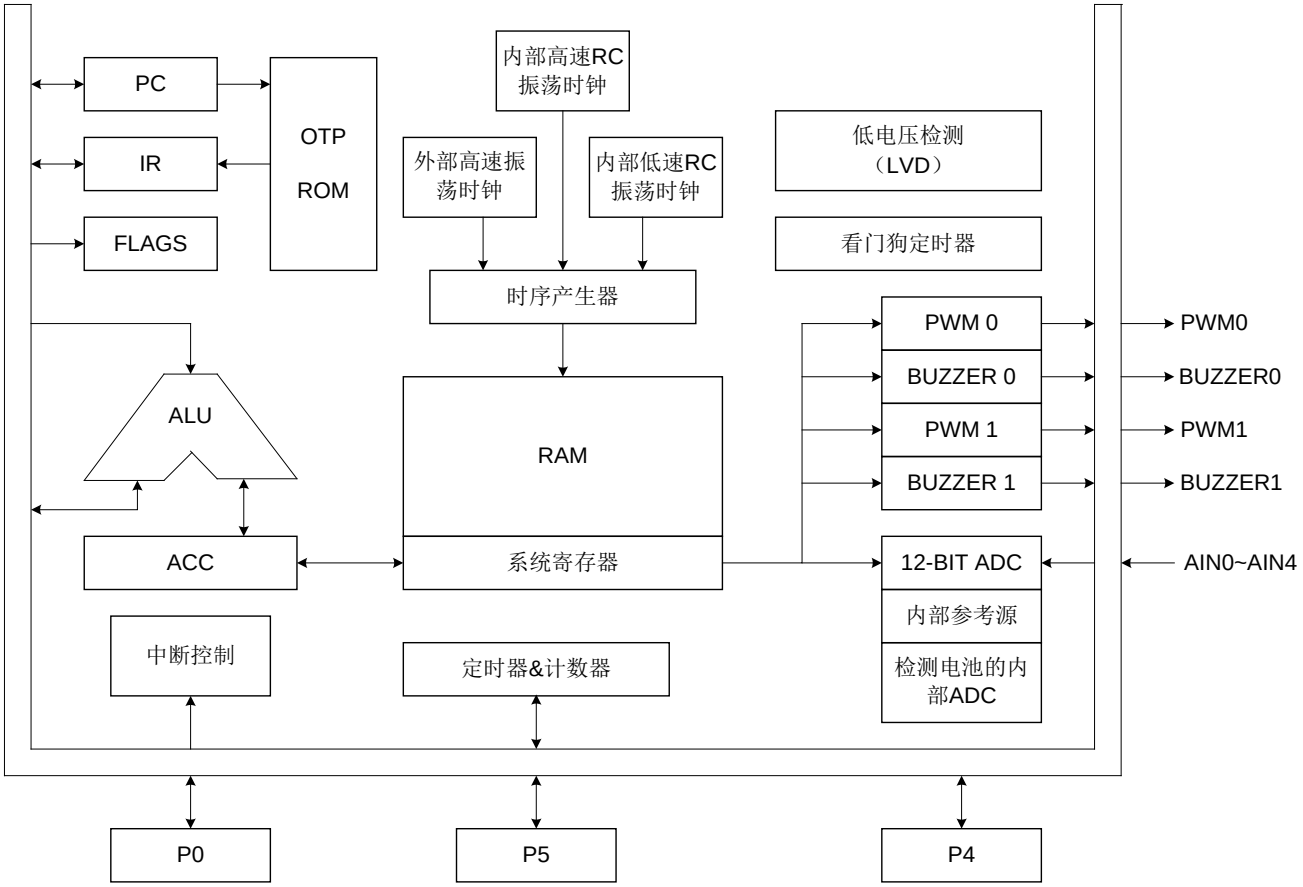
单片机型号	ROM	RAM	堆栈	定时器		I/O	ADC	ADC Int.Ref	绿色模式	PWM	唤醒功能引脚数目	封装形式
				TC0	TC1					Buzzer		
SN8P2711B	1K*16	64	4	V	V	12	**5+1 ch**	**V**	**V**	**2**	**5**	**P-DIP 14/SOP 14/MSOP 10**
SN8P2711A	1K*16	64	4	V	V	12	**5+1 ch**	**V**	**V**	**2**	**5**	**P-DIP 14/SOP 14/SSOP 16**

☞ **SN8P2711 升级为 SN8P2711A 注意事项**

项目	SN8P2711B	SN8P2711A
P4	P4：输入模式时施密特触发	P4：无施密特触发
32K 振荡器连接 XIN/XOUT 引脚的电容	15pF，连接到 GND	20pF，连接到 GND
绿色模式功耗（IHRC 编译选项）	SN8P2711B 绿色模式功耗（IHRC 编译选项）为 SN8P2711A 的 60%～70%。	
ADC 零漂补偿功能	增加 ADC 零漂补偿功能	无

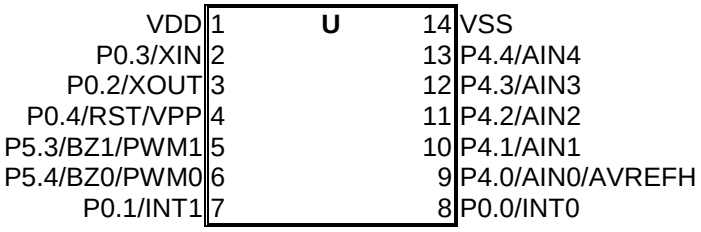
➢ **SN8P2711A 引脚配置（P-DIP，SOP）兼容 SN8P2711B。**
➢ **SN8P2711A 代码可以转化为 SN8P2711B。**
源代码烧录：通过 Writer 选择芯片名称为 SN8P2711B，可以直接烧录 SN8P2711A 的 SN8 文件到 SN8P2711B 中。
源代码：直接将 SN8P2711A 的 ASM 文件宣告为 SN8P2711B，重新编译即可。

1.2 系统结构框图


PC
IR
FLAGS
OTP
ROM
内部高速RC
振荡时钟
外部高速振
荡时钟
内部低速RC
振荡时钟
低电压检测
（LVD）
看门狗定时器
时序产生器
ALU
RAM
ACC
系统寄存器
PWM 0
BUZZER 0
PWM 1
BUZZER 1
12-BIT ADC
内部参考源
检测电池的内
部ADC
PWM0
BUZZER0
PWM1
BUZZER1
AIN0~AIN4
中断控制
定时器&计数器
P0
P5
P4

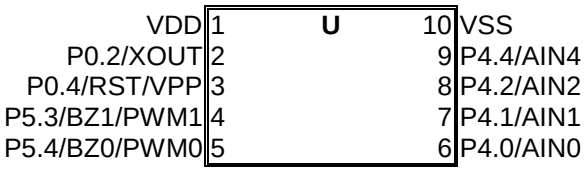
1.3 引脚配置

SN8P2711BP（P-DIP 14 pins）
SN8P2711BS（SOP 14 pins）



SN8P2711BP
SN8P2711BS

SN8P27113BA（MSOP 10 pins）



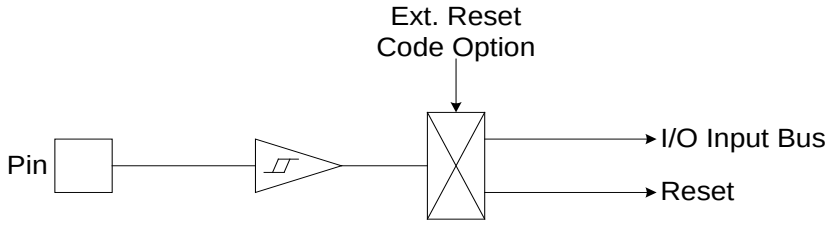
SN8P27113BA

1.4 引脚说明

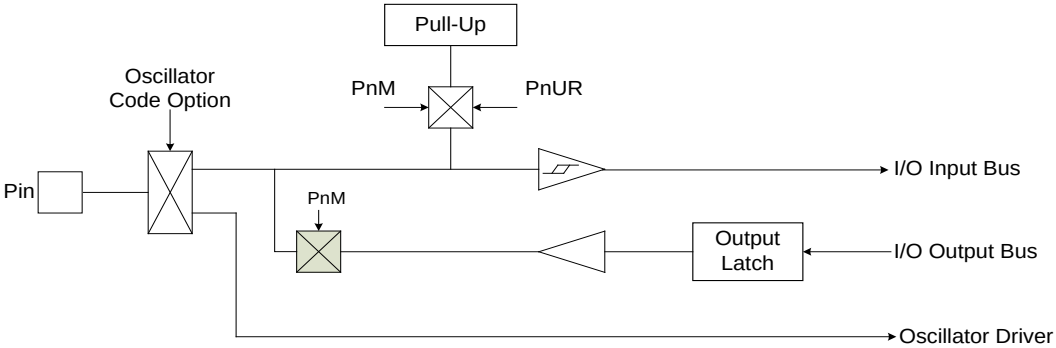
引脚名称	类型	功能说明
VDD, VSS	P	电源输入端。
P0.4/RST/VPP	I, P	P0.4：单向输入引脚，施密特触发，无内置上拉电阻，具有唤醒功能。 RST：系统复位输入引脚，施密特结构，低电平触发，通常保持高电平。 VPP：OTP 烧录引脚。
P0.3/XIN	I/O	P0.3：双向输入/输出引脚，输入模式时施密特触发，内置上拉电阻，具有唤醒功能。 XIN：使能外部振荡电路（晶体/RC 振荡电路）时为振荡信号输入引脚。
P0.2/XOUT	I/O	P0.2：双向输入/输出引脚，输入模式时施密特触发，内置上拉电阻，具有唤醒功能。 XOUT：使能外部晶体振荡器时为振荡器输出引脚。
P0.0/INT0	I/O	P0.0：双向输入/输出引脚，输入模式时施密特触发，内置上拉电阻，具有唤醒功能。 INT0：外部中断输入引脚。
P0.1/INT1	I/O	P0.1：双向输入/输出引脚，输入模式时施密特触发，内置上拉电阻，具有唤醒功能。 INT1：外部中断输入引脚。
P4.0/AIN0/AVREFH	I/O	P4.0：双向输入/输出引脚，输入模式时施密特触发，内置上拉电阻。 AIN0：ADC 输入通道。 AVERFH：ADC 参考电压的高电平输入引脚。
P4.[4:1]/AIN[4:1]	I/O	P4 [4:1]：双向输入/输出引脚，输入模式时施密特触发，内置上拉电阻。 AIN[4:1]：ADC 输入通道。
P5.3/PWM1/BZ1	I/O	P5.3：双向输入/输出引脚，输入模式时施密特触发，内置上拉电阻。 PWM1：PWM 输出引脚。 BZ1：Buzzer TC1/2 输出引脚。
P5.4/PWM0/BZ0	I/O	P5.4：双向输入/输出引脚，输入模式时施密特触发，内置上拉电阻。 PWM0：PWM 输出引脚。 BZ0：Buzzer TC0/2 输出引脚。

1.5 引脚电路结构图

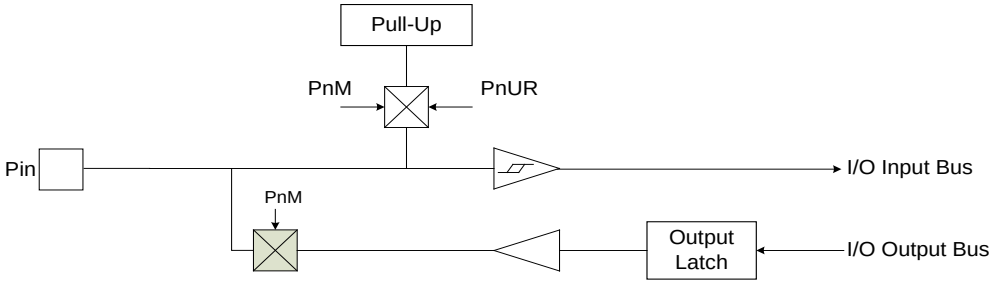
- 复位引脚：



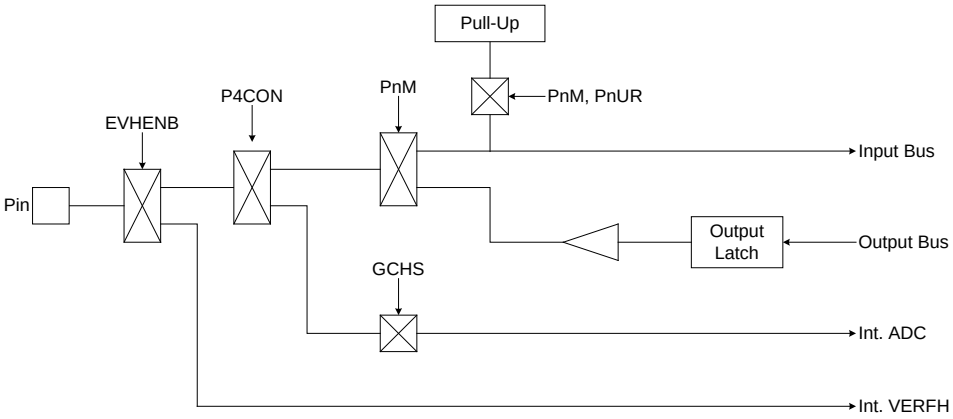
- 振荡器引脚



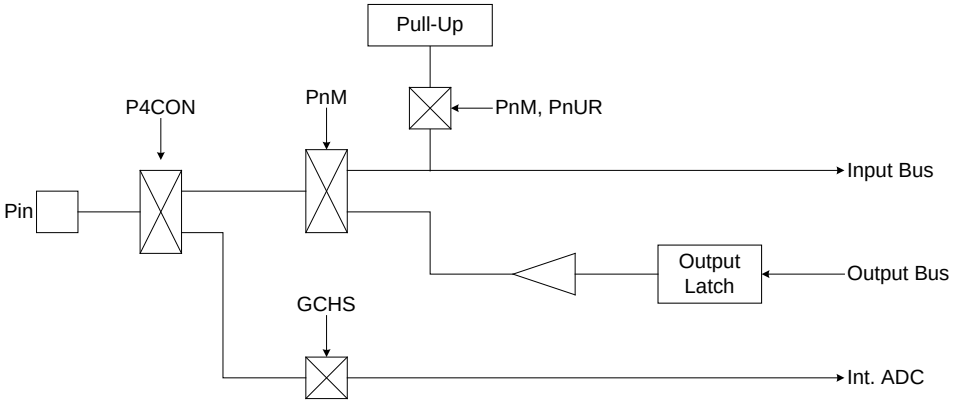
- GPIO 引脚：



- ADC 引脚，与参考电压高电平引脚共用：



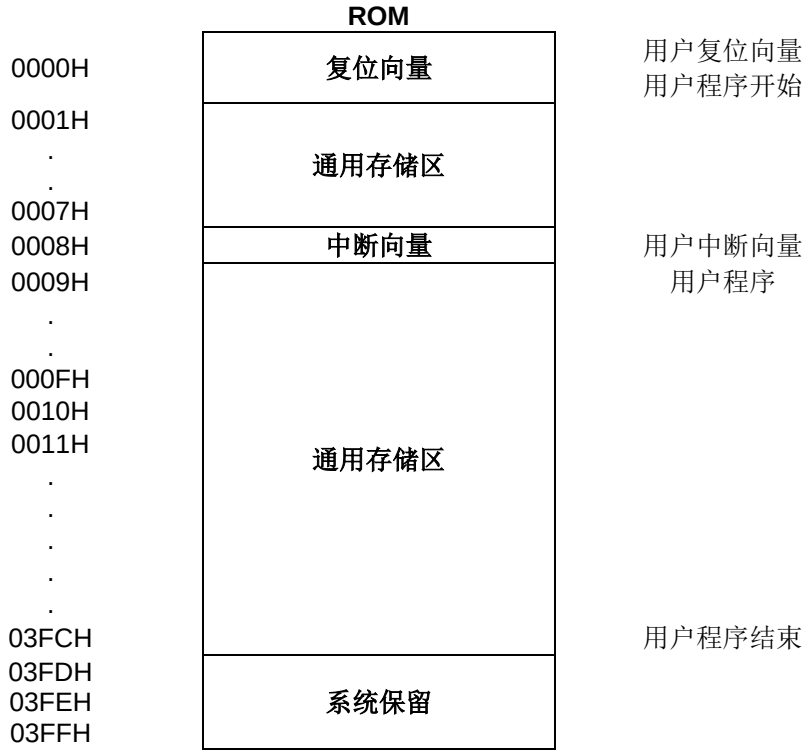
- ADC 引脚：



2 中央处理器（CPU）

2.1 程序存储器（ROM）

☞ **ROM：1K**



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。下面的示例程序说明了如何编写中断服务程序。

✸ 注：“PUSH”，“POP”指令用于存储和恢复 ACC/PFLAG，NT0、NTD 不受影响。PUSH/POP 缓存器是唯一的，且仅有一层。

➢ **例：定义中断向量，中断服务程序紧随 ORG 8H 之后。**

```
.CODE
                ORG         0
                JMP         START           ; 跳至用户程序。
                …
                ORG         8H              ; 中断向量。
                PUSH                        ; 保存 ACC 和 PFLAG。
                …
                POP                         ; 恢复 ACC 和 PFLAG。
                RETI                        ; 中断结束。
                …
START:                                      ; 用户程序开始。
                …                           ;
                JMP         START           ; 用户程序结束。
                …
                ENDP                        ; 程序结束。
```

➢ **例：定义中断向量，中断程序在用户程序之后。**

```
.CODE
                ORG         0
                JMP         START           ; 跳至用户程序。
                …
                ORG         8H              ; 中断向量。
                JMP         MY_IRQ          ; 跳至中断程序。

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

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

2.1.3 查表

在 SONiX 单片机中，对 ROM 区中的数据进行查找，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$M    ; 设置 TABLE1 地址高字节。
                B0MOV     Z, #TABLE1$L    ; 设置 TABLE1 地址低字节。
                MOVC                      ; 查表，R = 00H，ACC = 35H。

                                          ; 查找下一地址。
                INCMS     Z
                JMP       @F              ; Z 没有溢出。
                INCMS     Y               ; Z 溢出（FFH → 00），→ Y=Y+1
                NOP                       ;
                                          ;
@@:             MOVC                      ; 查表，R = 51H，ACC = 05H。
                …                         ;
TABLE1:         DW        0035H           ; 定义数据表（16 位）数据。
                DW        5105H
                DW        2012H
                …
```

✷ 注：当寄存器 Z 溢出（从 0FFH 变为 00H）时，寄存器 Y 并不会自动加 1。因此， Z 溢出时，Y 必须由程序加 1，下面的宏 INC_YZ 能够对 Y 和 Z 寄存器自动处理。

➢ **例：宏 INC_YZ。**

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

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

➢ **例：通过“INC_YZ”对上例进行优化。**

```
                B0MOV     Y, #TABLE1$M    ; 设置 TABLE1 地址中间字节。
                B0MOV     Z, #TABLE1$L    ; 设置 TABLE1 地址低字节。
                MOVC                      ; 查表，R = 00H，ACC = 35H。

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

下面的程序通过累加器对 Y，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：由指令 B0ADD/ADD 对 Y 和 Z 寄存器加 1。**

```
                B0MOV     Y, #TABLE1$M    ; 设置 TABLE1 地址中间字节。
                B0MOV     Z, #TABLE1$L    ; 设置 TABLE1 地址低字节。

                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。
                NOP
GETDATA:                                  ;
                MOVC                      ; 存储数据，如果 BUF = 0，数据为 0035H。
                                          ; 如果 BUF = 1，数据=5105H。
                                          ; 如果 BUF = 2，数据=2012H。
                …
TABLE1:         DW        0035H           ; 定义数据表（16 位）数据。
                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
        B0ADD   B0PCL, 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

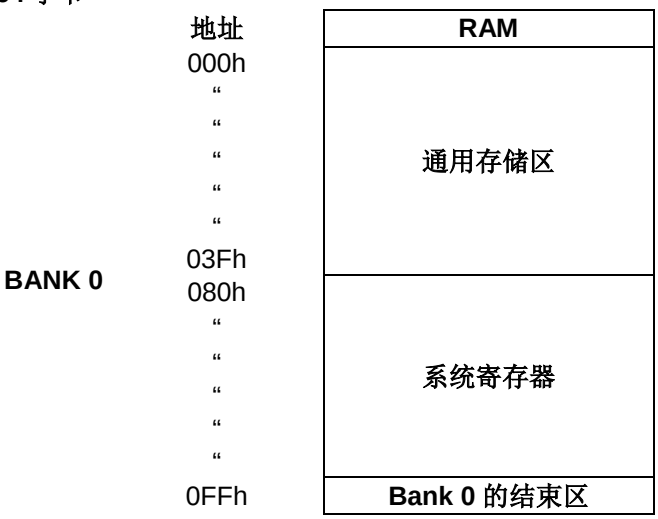
ROM 的最后一个地址是系统保留区，用户应该在计算 Checksum 时跳过该区域。

➢ **例：下面的程序说明如何从 00H 至用户代码结束的区域内进行 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：64 字节**



Bank0 的 080H~0FFH 是系统寄存器区（128 字节）。

64byte 的通用 RAM 定义在 BANK0。松翰提供“Bank0”操作指令（比如 B0MOV,B0ADD,B0BTS1,B0BSET...）来直接操作 BANK0 中的 RAM，而不管当前程序是否设置在 BANK0 区域

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	-	-	R	Z	Y	-	PFLAG	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P4CON	VREFH
B	-	ADM	ADB	ADR	ADT	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	-	-	-	-	P4M	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	-	-	-	P4	P5	-	-	T0M	-	TC0M	TC0C	TC1M	TC1C	TC1R	STKP
E	P0UR	-	-	-	P4UR	P5UR	-	@YZ	-	-	-	-	-	-	-	-
F	-	-	-	-	-	-	-	-	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2 系统寄存器说明

R = 工作寄存器和 ROM 查表数据缓存器
PFLAG = ROM 页和特殊标志寄存器
VERFH = ADC 参考电压寄存器
ADB = ADC 数据缓存器
PEDGE = P0.0 模式控制寄存器
INTRQ = 中断请求寄存器
OSCM = 振荡模式寄存器
TC0R = TC0 自动装载数据缓存器
Pn = Pn 数据缓存器
TC0M = TC0 模式寄存器
TC1M = TC1 模式寄存器
TC1R = TC1 自动装载实际缓存器
PnUR = Pn 上拉电阻控制寄存器
STK0~STK3 = 堆栈寄存器

Y, Z = 专用寄存器，@YZ 间接寻址寄存器，ROM 寻址寄存器
P4CON = P4 配置控制寄存器
ADM = ADC 模式寄存器
ADR = ADC 精度选择寄存器
ADT = ADC 零漂寄存器
PnM = Pn 模式控制寄存器
INTEN = 中断使能寄存器
WDTR = 看门狗清零寄存器
PCH, PCL = 程序计数器
T0M = TC0/TC1 加速和 TC0 唤醒功能寄存器
TC0C = TC0 计数寄存器
TC1C = TC1 计数寄存器
STKP = 堆栈指针
@YZ = 间接寻址寄存器

2.2.1.3 系统寄存器的位定义

地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	注释
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
0AEH				P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	R/W	P4CON
0AFH	EVHENB						VHS1	VHS2	R/W	VREFH
0B1H	ADENB	ADS	EOC	GCHS		CHS2	CHS1	CHS0	R/W	ADM
0B2H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R	ADB
0B3H		ADCKS1		ADCKS0	ADB3	ADB2	ADB1	ADB0	R/W	ADR
0B4H	ADTS1	ADTS0		ADT4	ADT3	ADT2	ADT1	ADT0	R/W	ADT
0B8H					P03M	P02M	P01M	P00M	R/W	P0M
0BFH				P00G1	P00G0				R/W	PEDGE
0C4H				P44M	P43M	P42M	P41M	P40M	R/W	P4M
0C5H				P54M	P53M				R/W	P5M
0C8H	ADCIRQ	TC1IRQ	TC0IRQ				P01IRQ	P00IRQ	R/W	INTRQ
0C9H	ADCIEN	TC1IEN	TC0IEN				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							PC9	PC8	R/W	PCH
0D0H				P04	P03	P02	P01	P00	R/W	P0
0D4H				P44	P43	P42	P41	P40	R/W	P4
0D5H				P54	P53				R/W	P5
0D8H					TC1X8	TC0X8	TC0GN		R/W	T0M
0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD1	TC1OUT	PWM1OUT	R/W	TC1M
0DDH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
0DEH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	W	TC1R
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H					P03R	P02R	P01R	P00R	W	P0UR
0E4H				P44R	P43R	P42R	P41R	P40R	W	P4UR
0E5H				P54R	P53R				W	P5UR
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H							S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH							S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH							S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH							S0PC9	S0PC8	R/W	STK0H

✷ 注:
1、为了避免系统错误，在初始化时，请将上表所有寄存器的位都按照设计要求设置为确定的"1"或者"0"；
2、所有寄存器的名称在 SN8ASM 编译器中做了宣告；
3、寄存器中各位的名称已在 SN8ASM 编译器中以“F”为前缀定义过；
4、指令“B0BSET”、“B0BCLR”、“BSET”、“BCLR”只能用于“R/W”寄存器。

2.2.2 累加器

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

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

➢ **例：读/写 ACC。**

```
; 数据写入 ACC。
                MOV         A, #0FH

; 读取 ACC 中的数据并存入 BUF。
                MOV         BUF, A
                B0MOV       BUF, A

; BUF 中的数据写入 ACC。
                MOV         A, BUF
                B0MOV       A, BUF
```

系统执行中断操作时，ACC 和 PFLAG 中的数据不会自动存储，用户需通过程序将中断入口处的 ACC 和 PFLAG 中的数据送入存储器进行保存。可通过“PUSH”和“POP”指令对 ACC 和 PFLAG 等系统寄存器进行存储及恢复。

➢ **例：ACC 和工作寄存器中断保护操作。**

```
INT_SERVICE:
                PUSH                    ; 保存 PFLAG 和 ACC。
                …           .
                …
                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 检测信息，其中，位 NT0 和 NPD 显示系统复位状态信息，包括上电复位、LVD 复位、外部复位和看门狗复位；位 C、DC 和 Z 显示 ALU 的运算信息。位 LVD24 和 LVD36 显示了单片机供电电压状况。

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	X	X	0	0	-	0	0	0

Bit [7:6] **NT0, NPD**：复位状态标志。

NT0	NPD	复位状态
0	0	看门狗复位
0	1	保留
1	0	LVD 复位
1	1	外部复位

Bit 5 **LVD36**：3.6V LVD 工作电压标志， LVD 编译选项为 LVD_H 时有效。
0 =系统工作电压 VDD 超过 3.6V，低电压检测器没有工作；
1 =系统工作电压 VDD 低于 3.6V，说明此时低电压检测器已处于监控状态。

Bit 4 **LVD24**：2.4V LVD 工作电压标志，LVD 编译选项为 LVD_M 时有效。
0 =系统工作电压 VDD 超过 2.4V，低电压检测器没有工作；
1 =系统工作电压 VDD 低于 2.4V，说明此时低电压检测器已处于监控状态。

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/分支运算的结果为零；
0 = 算术/逻辑/分支运算的结果非零。

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

2.2.4 程序计数器

程序计数器 PC 是一个 10 位二进制程序地址寄存器，分高 2 位和低 8 位。专门用来存放下一条需要执行指令的内存地址。通常，程序计数器会随程序中指令的执行自动增加。

若程序执行 CALL 和 JMP 指令时，PC 指向特定的地址。

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC	-	-	-	-	-	-	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
复位后	-	-	-	-	-	-	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     ; 如果 ACC = 12H，则跳过下一条指令。
                JMP         C0STEP      ; 否则跳至 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
                …
C0STEP:         NOP
```

INCMS：

```
                INCMS       BUF0
                JMP         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
                …
C0STEP:         NOP
```

DECMS：

```
                DECMS       BUF0
                JMP         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 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	X	X	X	X	X	X	X	X

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	X	X	X	X	X	X	X	X

➢ **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:                    ;
        …
```

2.2.6 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	X	X	X	X	X	X	X	X

2.3 寻址模式

2.3.1 立即寻址

将立即数送入 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：立即数寻址中，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 的 12H 单元。**

```
B0MOV   12H, A
```

2.3.3 间接寻址

通过指针寄存器（Y/Z）访问 RAM 数据。

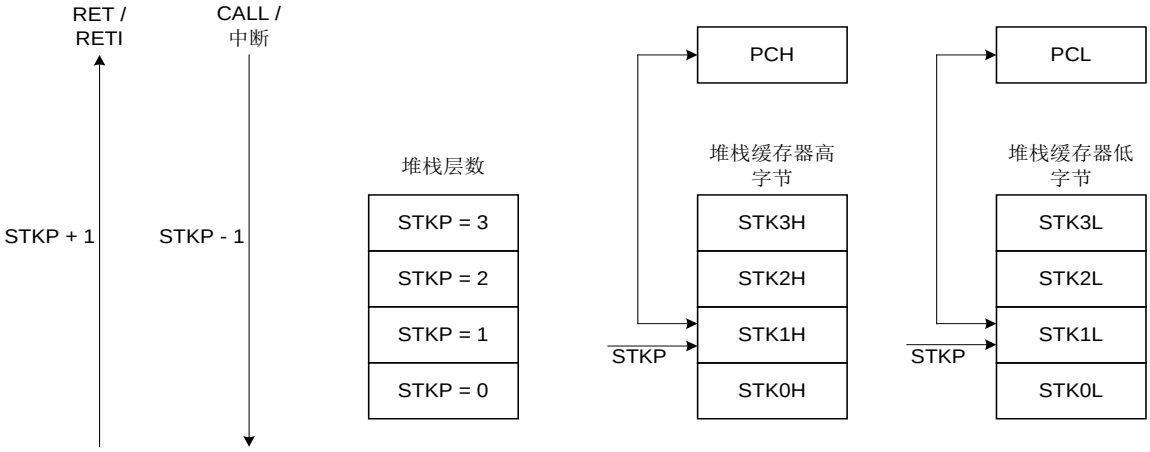
- **例：用 @YZ 实现间接寻址。**

```
B0MOV   Y, #0       ; Y 清零以寻址 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

SN8P2711B 的堆栈缓存器共 4 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，STKnH 和 STKnL 分别是各堆栈缓存器的高、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，10 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。

通过入栈指令 PUSH 和出栈指令 POP 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 = 使能。

➢ **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	-	-	-	-	-	-	SnPC9	SnPC8
读/写	-	-	-	-	-	-	R/W	R/W
复位后	-	-	-	-	-	-	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 = 3 ~ 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			缓存器		备注
	STKPB2	STKPB1	STKPB0	高字节	低字节	
0	1	1	1	保留	保留	-
1	1	1	0	STK0H	STK0L	-
2	1	0	1	STK1H	STK1L	-
3	1	0	0	STK2H	STK2L	-
4	0	1	1	STK3H	STK3L	-
> 4	0	1	0	-	-	**缓存器已满，错误**

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			缓存器		备注
	STKPB2	STKPB1	STKPB0	高字节	低字节	
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	保留	保留	-

2.5 编译选项表（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，看门狗定时器的操作，LVD 选项，复位引脚选项以及 OTP ROM 的安全控制。如下表所示：

编译选项	内容	功能说明
High_Clk	IHRC_16M	高速时钟采用内部 16MHz RC 振荡电路，XIN/XOUT（P0.3/P0.2）为普通的 I/O 引脚。
	RC	外部高速时钟振荡器采用廉价的 RC 振荡电路，XOUT（P0.2）为普通的 I/O 引脚。
	32K X'tal	外部高速时钟振荡器采用低频晶体/陶瓷振荡器（如 32.768KHz）。
	12M X'tal	外部高速时钟振荡器采用高频晶体/陶瓷振荡器（如 12MHz）。
	4M X'tal	外部高速时钟振荡器采用标准晶体/陶瓷振荡器（如 4MHz）。
Watch_Dog	Always_On	始终开启看门狗定时器，即使在睡眠模式和绿色模式下也处于开启状态。
	Enable	开启看门狗定时器，但在睡眠模式和绿色模式下关闭。
	Disable	关闭看门狗定时器。
Fcpu	Fhosc/1	指令周期 = 1 个时钟周期，必须关闭杂讯滤波功能。
	Fhosc/2	指令周期 = 2 个时钟周期，必须关闭杂讯滤波功能。
	Fhosc/4	指令周期 = 4 个时钟周期。
	Fhosc/8	指令周期 = 8 个时钟周期。
	Fhosc/16	指令周期 = 16 个时钟周期。
Reset_Pin	Reset	使能外部复位引脚。
	P04	P0.4 为单向输入引脚，无上拉电阻。
Security	Enable	ROM 程序加密。
	Disable	ROM 程序不加密。
Noise_Filter	Enable	使能杂讯滤除功能，Fcpu = Fosc/4~Fosc/16。
	Disable	禁止杂讯滤除功能，Fcpu = Fosc/1~Fosc/16。
LVD	LVD_L	VDD 低于 2.0V 时，系统复位。
	LVD_M	VDD 低于 2.0V 时，系统复位； PFLAG 寄存器的 LVD24 位作为 2.4V 低电压监测器。
	LVD_H	VDD 低于 2.4V 时，系统复位； PFLAG 寄存器的 LVD36 位作为 3.6V 低电压监测器。

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/4@3V，32KHz/4@5V）。

2.5.2 Reset_Pin编译选项

复位引脚与单向输入引脚共用，由编译选项控制。

- **Reset**：使能外部复位引脚功能。当下降沿触发时，系统复位。
- **P04**：使能 P0.4 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，可以保护 ROM 的内容。

2.5.4 Noise Filter编译选项

Noise Filter 编译选项是强杂讯滤除功能以减少杂讯对系统时钟的影响。如使能杂讯滤波器，在高干扰环境下，使能看门狗定时器且选择一个合适的 LVD 选项可以使整个系统更好的工作。

3 复位

3.1 概述

SN8P2711B 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（仅在外部复位引脚处于使能状态）。**

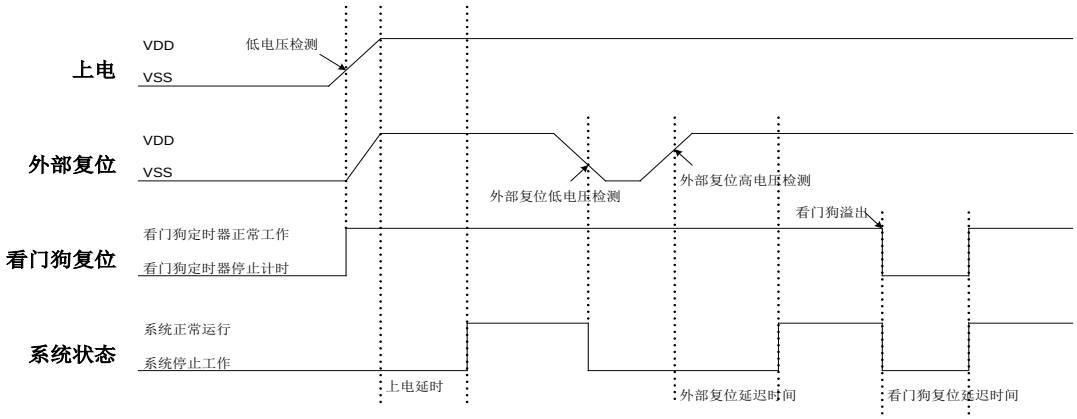
上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器 PC 清零。复位结束后，系统从向量 0000H 处重新开始运行。PFLAG 寄存器的 NT0 和 NPD 两个标志位能够给出系统复位状态的信息。用户可以编程控制 NT0 和 NPD，从而控制系统的运行路径。

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	X	X	0	0	-	0	0	0

Bit [7:6] **NT0, NPD：**复位状态标志。

NT0	NPD	复位情况	说明
0	0	看门狗复位	看门狗溢出
0	1	保留	-
1	0	上电及 LVD 复位	电源电压低于 LVD 检测值
1	1	外部复位	外部复位引脚检测到低电平

任何一种复位情况都需要一定的响应时间，系统提供完善的复位流程以保证复位动作的顺利进行。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。在用户终端使用的过程中，应注意考虑主机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。系统上电的过程呈逐渐上升的曲线形式，需要一定时间才能达到正常电平值。下面给出上电复位的正常时序：

- **上电**：系统检测到电源电压上升并等待其稳定；
- **外部复位（仅限于外部复位引脚使能状态）**：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释放；
- **系统初始化**：所有的系统寄存器被置为初始值；
- **振荡器开始工作**：振荡器开始提供系统时钟；
- **执行程序**：上电结束，程序开始运行。

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：

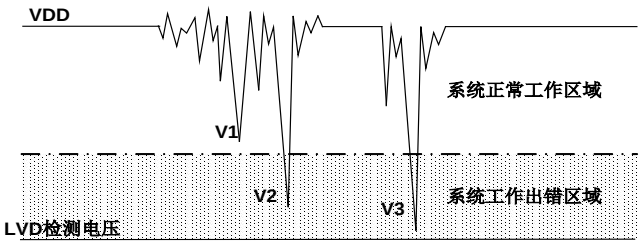
- 对看门狗清零之前，检查 I/O 口的状态和 RAM 的内容可增强程序的可靠性；
- 不能在中断中对看门狗清零，否则无法侦测到主程序跑飞的状况；
- 程序中应该只在主程序中有一次清看门狗的动作，这种架构能够最大限度的发挥看门狗的保护功能。

✷ **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

3.4.1 概述

掉电复位针对外部因素引起的系统电压跌落情形（例如，干扰或外部负载的变化），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。

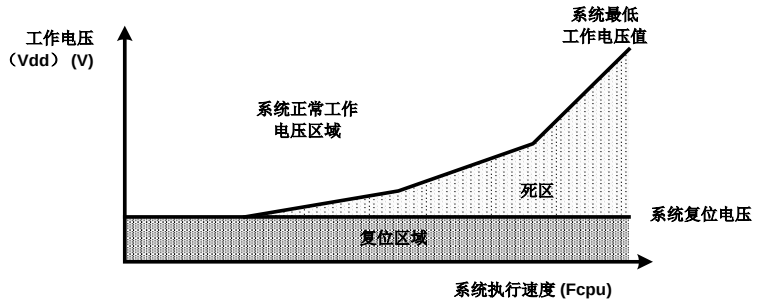
AC 运用中：

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。

3.4.2 系统工作电压

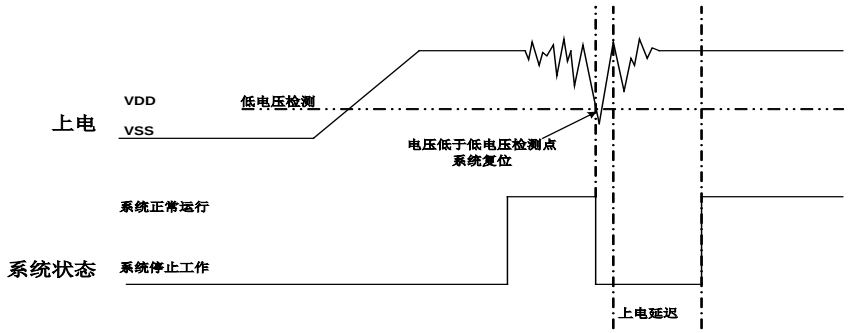
为了改善系统掉电复位的性能，首先必须明确系统具有的最低工作电压值。系统最低工作电压与系统执行速度有关，不同的执行速度下最低工作电压值也不同。



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。

3.4.3 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。如果电源跌落剧烈，远低于 LVD 触发点，LVD 能够起到保护作用，让系统正常复位；如果电源电压跌落不是很剧烈，仅仅是接近 LVD 触发点而造成的系统错误，则 LVD 就不能起到保护作用让系统复位。更多的关于 LVD 电压点的细节请参考电气特性章节。

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	X	X	0	0	-	0	0	0

Bit 5 **LVD36**：3.6V LVD 工作电压标志， LVD 编译选项为 LVD_H 时有效。
0 =系统工作电压 VDD 超过 3.6V，低电压检测器没有工作；
1 =系统工作电压 VDD 低于 3.6V，说明此时低电压检测器已处于监控状态。

Bit 4 **LVD24**：2.4V LVD 工作电压标志，LVD 编译选项为 LVD_M 时有效。
0 =系统工作电压 VDD 超过 2.4V，低电压检测器没有工作；
1 =系统工作电压 VDD 低于 2.4V，说明此时低电压检测器已处于监控状态。

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；

✷ 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。

3.4.4 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

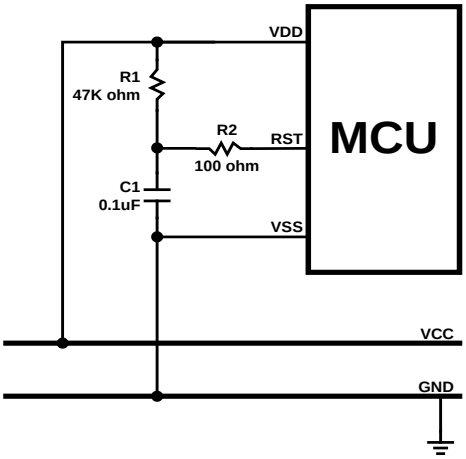
外部复位功能由编译选项“Reset_Pin”控制。将该编译选项置为“Reset”，可使能外部复位功能。外部复位引脚为施密特触发结构，低电平有效。复位引脚处于高电平时，系统正常运行。当复位引脚输入低电平信号时，系统复位。外部复位操作在上电和正常工作模式时有效。需要注意的是，在系统上电完成后，外部复位引脚必须输入高电平，否则系统将一直保持在复位状态。外部复位的时序如下：

- **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：**初始化所有的系统寄存器；
- **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

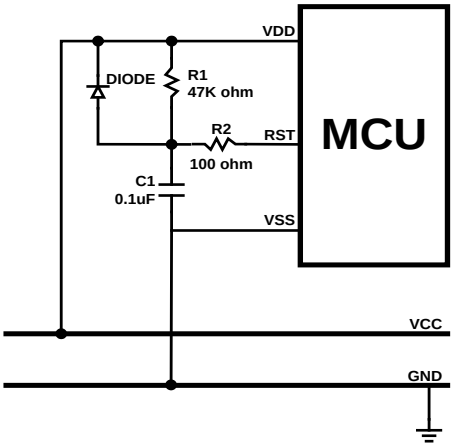
3.6.1 RC复位电路



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。**

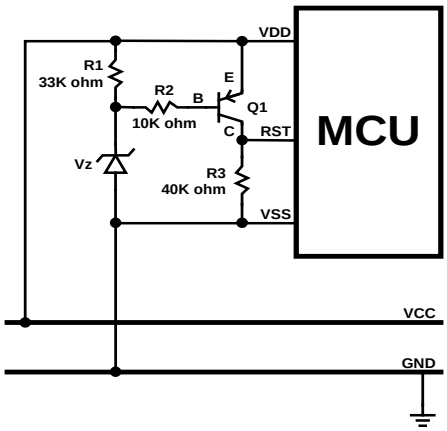
3.6.2 二极管及RC复位电路



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。

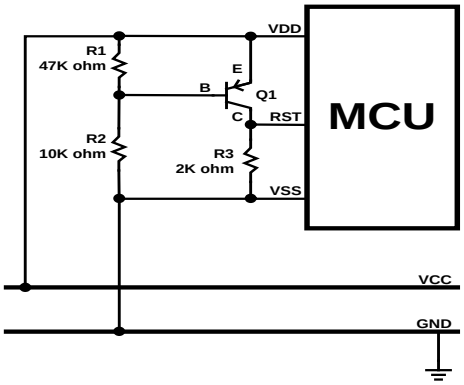
✷ **注：“基本 RC 复位电路”和“二极管及 RC 复位电路”中的电阻 R2 都是必不可少的限流电阻，以避免复位引脚 ESD（Electrostatic Discharge）或 EOS（Electrical Over-stress）击穿 。**

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

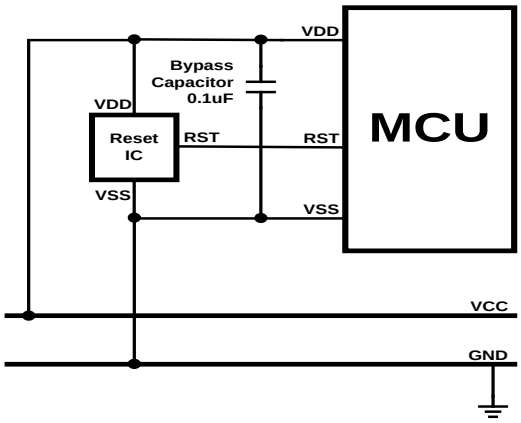


电压偏置复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，这种复位电路的检测电压值的精确度有所降低。电路中，R1 和 R2 构成分压电路，当 VDD 高于和等于分压值“0.7V x (R1 + R2) / R1”时，三极管集电极 C 输出高电平，单片机正常工作；VDD 低于“0.7V x (R1 + R2) / R1”时，集电极 C 输出低电平，单片机复位。

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



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

SN8P2711B 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时钟和外部高速时钟，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。

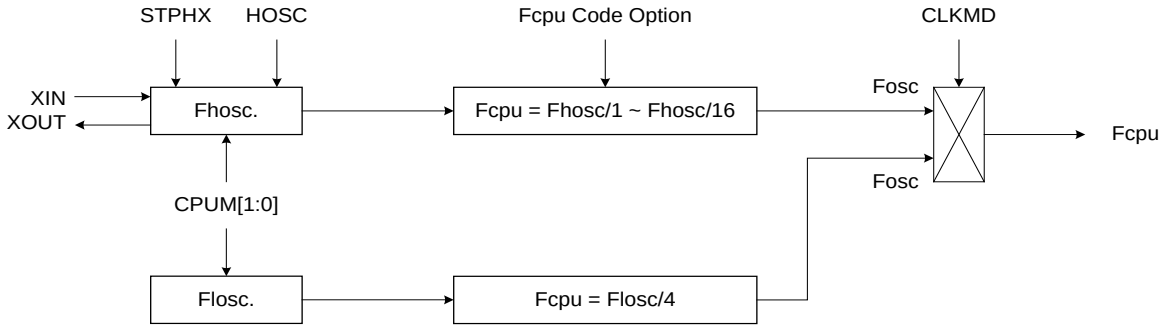
- **高速振荡器**

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

- **低速振荡器**

  内部低速振荡器：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/内部 RC 振荡器时钟频率。
- **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。

SONIX 提供“Noise Filter”，由编译选项控制。高干扰环境下，Noise Fliter 可以滤除来自外部振荡器的高干扰信号以使系统正常工作。使能 Noise Filter 时，高速时钟下 Fcpu 被限制为 Fhosc/4。

4.2 指令周期Fcpu

系统时钟速率，即指令周期（Fcpu），从系统时钟源分离出来，决定系统的工作速率。Fcpu 的速率由 Fcpu 编译选项决定，正常模式下，Fcpu=Fhosc/1~Fhosc/16。若高速时钟源为外部 4MHz 振荡器，则 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。

高干扰环境下，强烈建议设置 code option 的 Fcpu 选项在 Fhosc/4 以下，以减少干扰的影响。

4.3 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。

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、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，不连接任何外部振荡设备。
- **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。
- **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。

4.4.2 内部高速RC振荡器（IHRC）

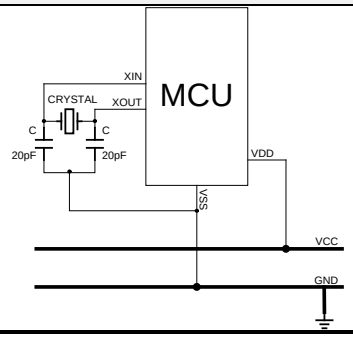
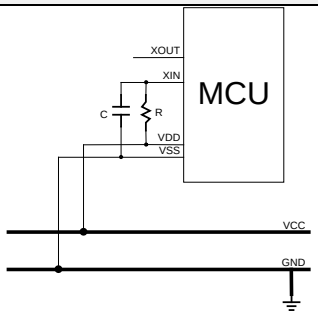
内部高速 16MHz RC 振荡器，普通环境下精确度为±2%，当选择 IHRC_16M 时，使能内部高速振荡器。

- **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。

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，电阻的阻值决定频率。

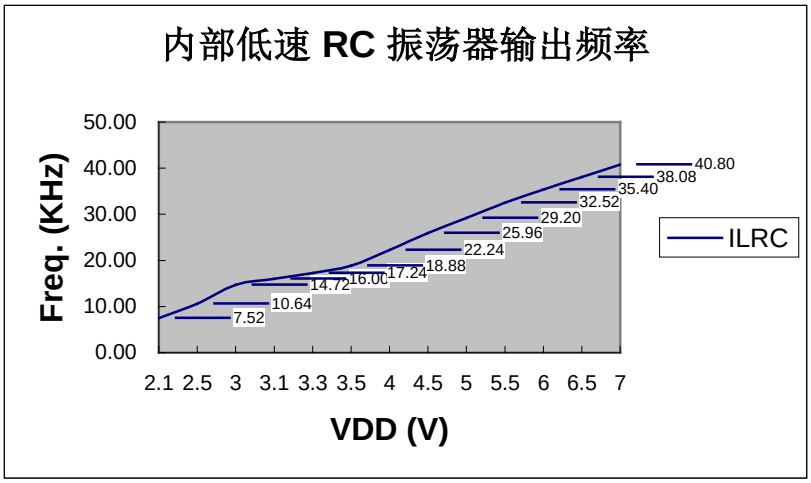
4.4.4 外部振荡应用电路

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。**

系统工作在睡眠模式下，可以停止低速 RC 振荡器。

- ➢ **例：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	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-
读/写	-	-	-	R/W	R/W	R/W	R/W	-
复位后	-	-	-	0	0	0	0	-

Bit 1 **STPHX**：高速振荡器控制位。
0 = 运行；
1 = 停止，内部低速 RC 振荡器仍然运行。

Bit 2 **CLKMD**：系统时钟模式控制位。
0 = 普通（双时钟）模式，高速时钟作为系统时钟；
1 = 低速模式，低速时钟作为系统时钟。

Bit[4:3] **CPUM[1:0]**：CPU 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          ; 停止外部高速振荡器。
```

STPHX 位为内部高速 RC 振荡器和外部高速振荡器的控制位。当 STPHX=0，内部高速 RC 振荡器和外部高速振荡器正常运行；当 STPHX=1，外部高速振荡器和内部高速 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；**
- **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 Fcpu 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。**

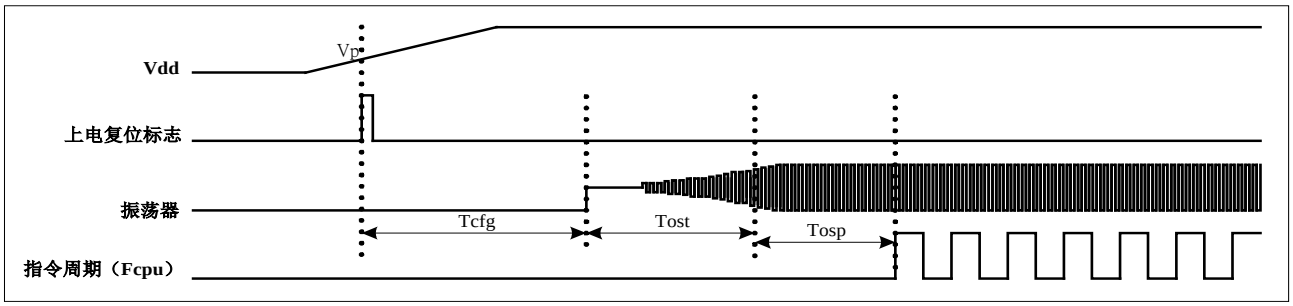
```
        B0BSET      P0M.0           ; P0.0 置为输出模式以输出 Fcpu 的触发信号。
@@:
        B0BSET      P0.0
        B0BCLR      P0.0
        JMP         @B
```

✷ **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。**

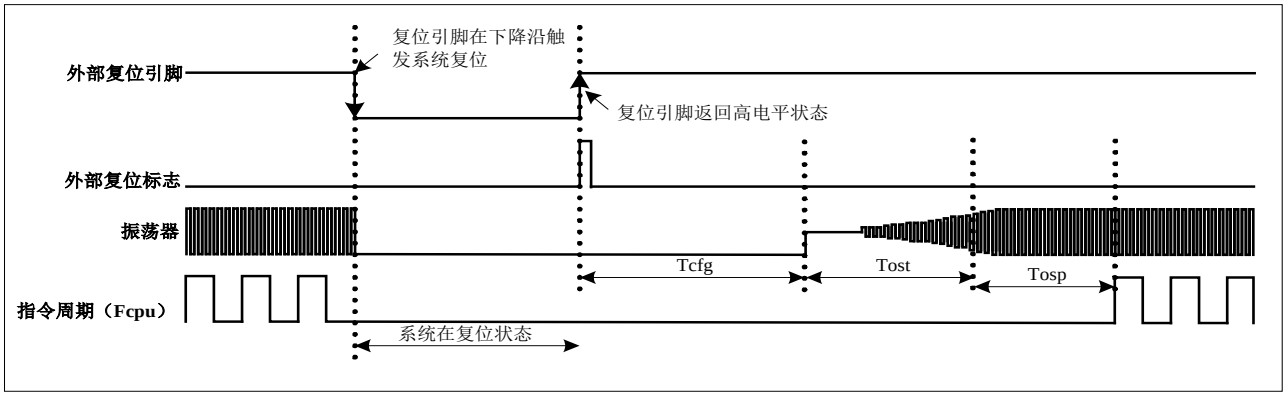
4.8 系统时钟时序

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慢，RC 振荡器的启动时间要比晶体/陶瓷振荡器的启动时间快。	-
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）	48ms @ F_{hosc} = 32KHz 384us @ F_{hosc} = 4MHz 96us @ 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: 2us @ F_{hosc} = 4MHz 0.5us @ F_{hosc} = 16MHz

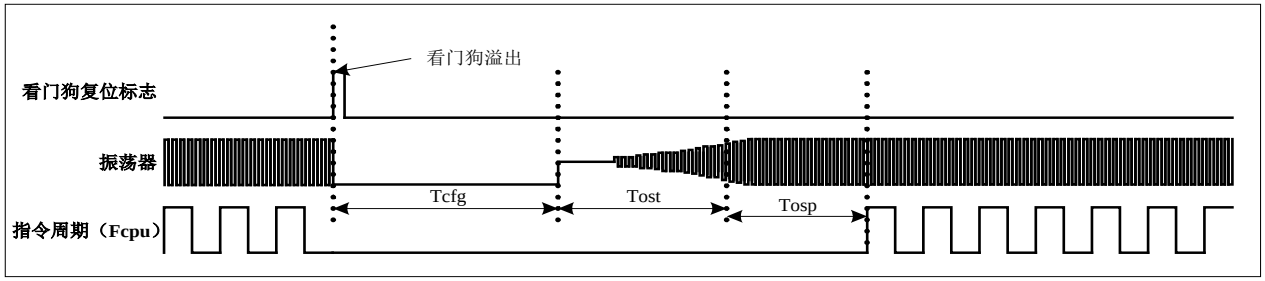
- 上电复位时序：



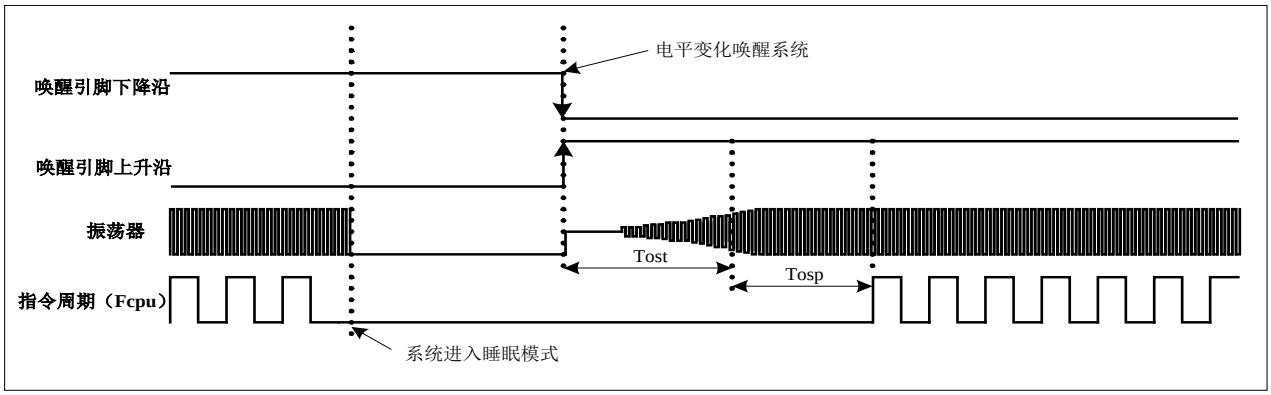
- 外部复位引脚复位时序：



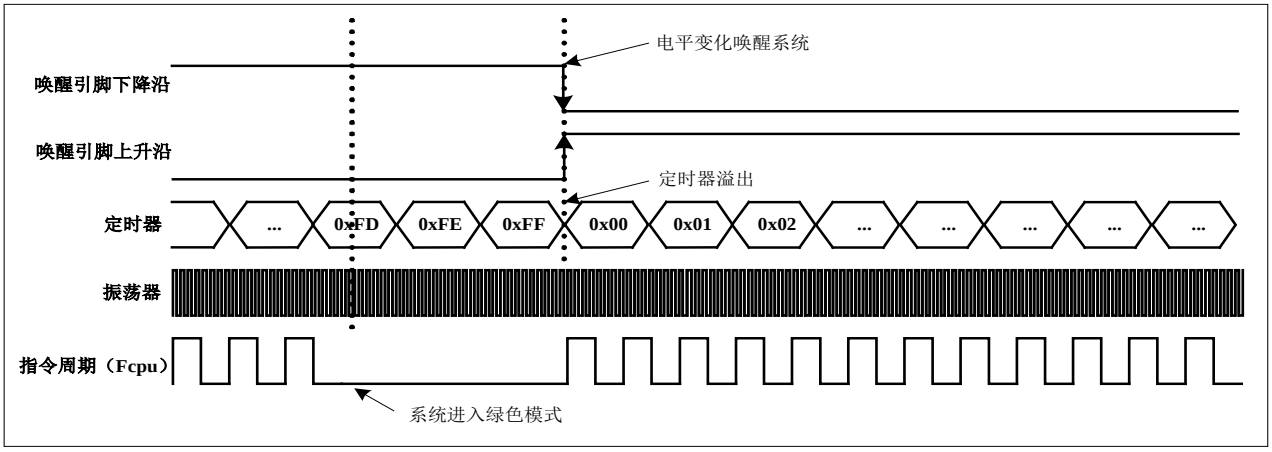
- 看门狗复位时序：



- **睡眠模式唤醒时序：**

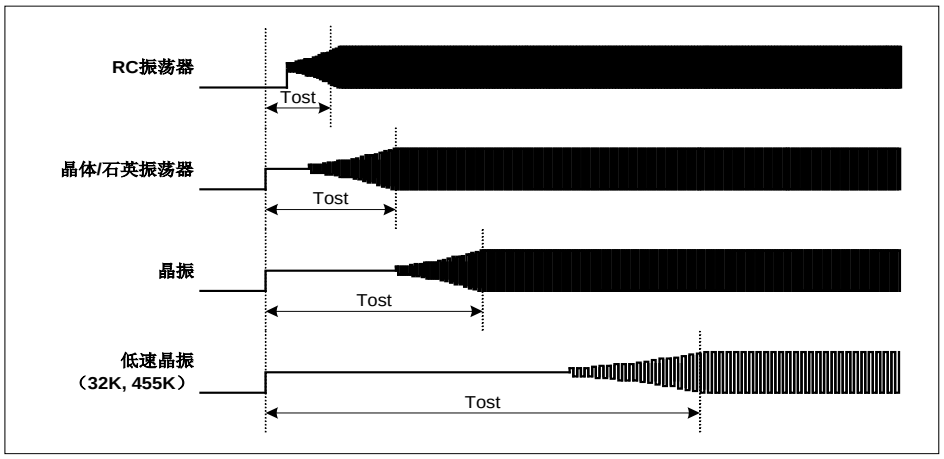


- **绿色模式唤醒时序：**



- **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慢，RC 振荡器的启动时间要比晶体/陶瓷振荡器的启动时间快。



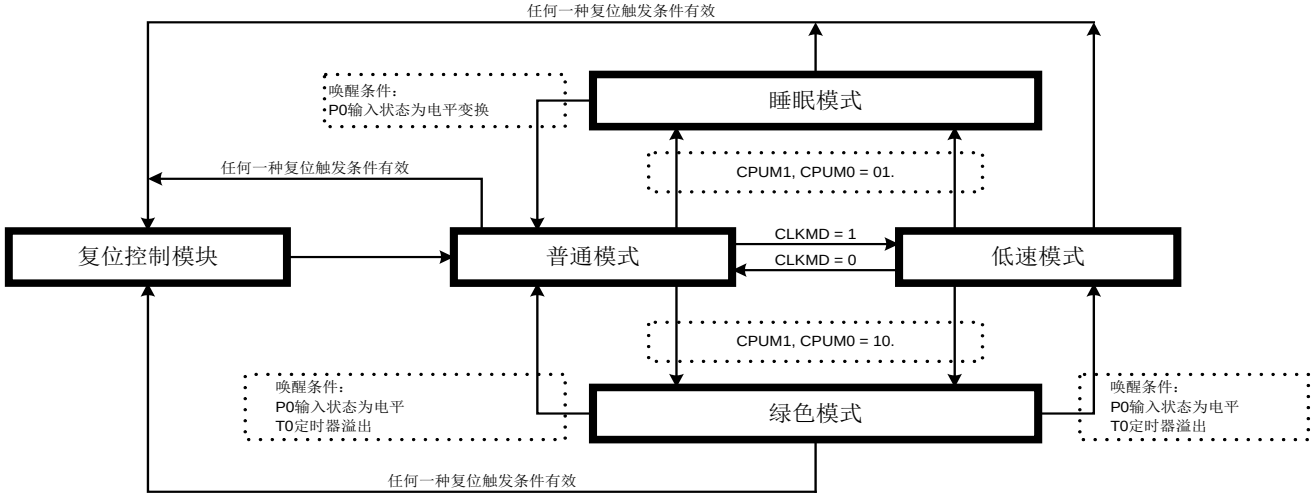
5 系统工作模式

5.1 概述

SN8P2711B 可以在 4 种工作模式下以不同的时钟频率工作，这些模式可以控制振荡器的工作、程序的执行以及模拟电路的功能损耗。

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

工作模式控制框图



工作模式说明

工作模式	普通模式	低速模式	绿色模式	睡眠模式
EHOSC	运行	STPHX 控制	STPHX 控制	停止
IHRC	运行	STPHX 控制	STPHX 控制	停止
ILRC	运行	运行	运行	停止
CPU 指令	执行	执行	停止	停止
TC0	TC0ENB 控制	TC0ENB 控制	TC0ENB 控制 仅 PWM/Buzzer 有效	无效
TC1	TC1ENB 控制	TC1ENB 控制	TC1ENB 控制 仅 PWM/Buzzer 有效	无效
看门狗定时器	Watch_Dog 编译选项控制	Watch_Dog 编译选项控制	Watch_Dog 编译选项控制	Watch_Dog 编译选项控制
内部中断	全部有效	全部有效	TC0	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，TC0，复位	P0，复位

- **EHOSC**：外部高速时钟（XIN/XOUT）。
- **IHRC**：内部高速时钟（RC 振荡器）。
- **ILRC**：内部低速时钟（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。
- 通过 OSCM 寄存器，系统可以从普通模式切换到其它任何一种工作模式。
- 系统从睡眠模式唤醒后进入普通模式。
- 低速模式可以切换到普通模式。
- 从普通模式切换到绿色模式，唤醒后返回到普通模式。

5.3 低速模式

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

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

5.4 睡眠模式

睡眠模式是系统的理想状态，不执行程序，振荡器也停止工作。整个芯片的功耗低于 1uA。睡眠模式可以由 P0 的电平变换触发唤醒。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 电平变换触发。

✷ **注：普通模式下，设置 STPHX=1 禁止高速时钟振荡器，这样，无系统时钟在执行，此时系统进入睡眠模式，可以由 P0 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 电平变换触发；2、具有唤醒功能的定时器溢出，这样，用户可以给定时器设定固定的周期，系统就在溢出时被唤醒。由 OSCM 寄存器 CPUM1 位决定是否进入绿色模式，当 CPUM1=1，系统进入绿色模式。当系统从绿色模式下被唤醒后，自动禁止 CPUM1（0 状态）。

- 程序停止执行，所有的功能被禁止。
- 具有唤醒功能的定时器正常工作。
- 作为系统时钟源的振荡器正常工作，其它的振荡器工作状态取决于系统工作模式的配置。
- 由普通模式切换到绿色模式，被唤醒后返回到普通模式。
- 由低速模式切换到绿色模式，被唤醒后返回到低速模式。
- 绿色模式下的唤醒方式为 P0 电平变换触发唤醒和指定的定时器溢出。
- 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 TC0 唤醒功能。**

```
; 设置定时器 TC0 的唤醒功能。
    B0BCLR      FTC0IEN         ; 禁止 TC0 中断。
    B0BCLR      FTC0ENB         ; 禁止 TC0 定时器。
    MOV         A,#20H          ;
    B0MOV       TC0M,A          ; 设置 TC0 时钟= Fcpu / 64。
    MOV         A,#64H
    B0MOV       TC0C,A          ; 设置 TC0C 的初始值=64H（设置 T0 间隔值 = 10 ms）。
    B0BCLR      FTC0IEN         ; 禁止 TC0 中断。
    B0BCLR      FTC0IRQ         ; 清 TC0 中断请求。
    B0BSET      FTC0ENB         ; 使能 TC0 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 的电平变换）和内部触发（TC0 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 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），唤醒触发信号可以是外部触发信号（P0 电平变换）和内部触发信号（TC0 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，单片机需要等待一段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。**

外部高速振荡器的唤醒时间的计算如下：

唤醒时间 = 1/Fhosc * 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/Fhosc * 2048 = 0.512 ms（4 MHz）
总的唤醒时间 = 0.512 ms + 振荡器启动时间

内部高速 RC 振荡器的唤醒时间的计算如下：

唤醒时间 = 1/Fhosc * 8(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：**

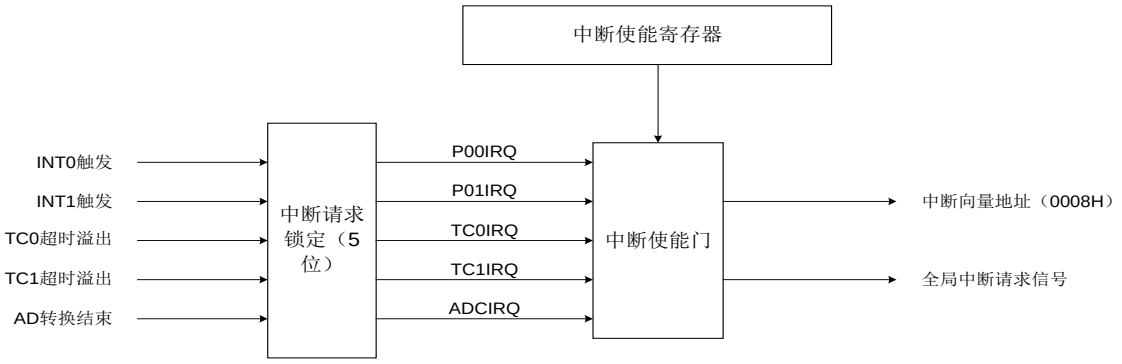
唤醒时间 = 1/Fosc * 8 = 0.5 us （Fhosc=16MHz）

✷ **注：高速时钟的启动时间与 VDD 和振荡器类型有关。**

6 中断

6.1 概述

SN8P2711B 提供 5 个中断源：3 个内部中断（TC0/TC1/ADC）和 2 个外部中断（INT0/INT1）。外部中断可以将系统从睡眠模式中唤醒进入高速模式，在返回到高速模式前，中断请求被锁定。一旦程序进入中断，寄存器 STKP 的位 GIE 被硬件自动清零以避免响应其它中断。系统退出中断后，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，必须开启全局中断控制位 GIE。**

6.2 中断请求使能寄存器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	TC1IEN	TC0IEN	-	-	-	P01IEN	P00IEN
读/写	R/W	R/W	R/W	-	-	-	R/W	R/W
复位后	0	0	0	-	-	-	0	0

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

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

Bit 5 **TC0IEN**：TC0 中断控制位。
0 = 无效；
1 = 有效。

Bit 6 **TC1IEN**：TC1 中断控制位。
0 = 无效；
1 = 有效。

Bit 7 **ADCIEN**：ADC 中断控制位。
0 = 无效；
1 = 有效。

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，则 INTRQ 中对应位将被置“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	TC1IRQ	TC0IRQ	-	-	-	P01IRQ	P00IRQ
读/写	R/W	R/W	R/W	-	-	-	R/W	R/W
复位后	0	0	0	-	-	-	0	0

Bit 0 **P00IRQ**：P0.0 中断（INT0）请求标志。
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。
0 = INT1 无中断请求；
1 = INT1 有中断请求。

Bit 5 **TC0IRQ**：TC0 中断请求标志。
0 = TC0 无中断请求；
1 = TC0 有中断请求。

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

Bit 7 **TC0IRQ**：ADC 中断请求标志。
0 = ADC 无中断请求；
1 = ADC 有中断请求。

6.4 GIE全局中断

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（ORG8），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处理

有中断请求发生并被响应后，程序转至 0008H 执行中断子程序。响应中断之前，必须保存 ACC、PFLAG 的内容。芯片提供 PUSH 和 POP 指令进行入栈保存和出栈恢复，从而避免中断结束后可能的程序运行错误。

✷ **注：“PUSH”、“POP”指令仅对 ACC 和 PFLAG 作中断保护，而不包括 NT0 和 NPD。PUSH/POP 缓存器是唯一的且仅有一层。**

➢ **例：对 ACC 和 PAFLG 进行入栈保护。**

```
                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 INT0（P0.0）中断

INT0 被触发，则无论 P00IEN 处于何种状态，P00IRQ 都会被置“1”。如果 P00IRQ=1 且 P00IEN=1，系统响应该中断；如果 P00IRQ=1 而 P00IEN=0，系统并不会执行中断服务。在处理多中断时尤其需要注意。

如果中断的触发方向和唤醒功能的触发方向是一样的，则在系统由 P0.0 从睡眠模式和绿色模式唤醒时，INT0 的中断请求（INT0IRQ）就会被锁定。系统会在唤醒后马上进入中断向量地址执行中断服务程序。

✷ **注：INT0 的中断请求被 P0.0 的唤醒触发功能锁定。**

✷ **注：P0.0 的中断触发边沿由 PEDGE 控制。**

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

Bit[4:3] **P00G[1:0]**：P0.0 中断触发控制位。
00 = 保留；
01 = 上升沿触发；
10 = 下降沿触发；
11 = 上升/下降沿触发（电平触发）。

➢ **例：INT0 中断请求设置，电平触发。**

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

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

➢ **例：INT0 中断。**

```
        ORG     8H              ;
        JMP     INT_SERVICE
INT_SERVICE:

        …                       ; 保存 ACC 和 PFLAG。

        B0BTS1  FP00IRQ         ; 检查是否有 P00 中断请求标志。
        JMP     EXIT_INT        ; P00IRQ = 0，退出中断。

        B0BCLR  FP00IRQ         ; 清 P00IRQ。
        …                       ; INT0 中断服务程序。
        …
EXIT_INT:
        …                       ; 恢复 ACC 和 PFLAG。

        RETI                    ; 退出中断。
```

6.7 INT1（P0.1）中断

INT1 被触发，则无论 P01IEN 处于何种状态，P01IRQ 都会被置“1”。如果 P01IRQ = 1 且 P01IEN = 1，系统响应该中断；如果 P01IRQ = 1 而 P01IEN = 0，系统并不会执行中断服务。在处理多中断时尤其需要注意。

如果中断的触发方向和唤醒功能的触发方向是一样的，则在系统由 P0.1 从睡眠模式和绿色模式唤醒时，INT1 的中断请求（INT1IRQ）就会被锁定。系统会在唤醒后马上进入中断向量地址执行中断服务程序。

✷ **注：INT1 的中断请求被 P0.1 的唤醒触发功能锁定。**

✷ **注：P0.1 中断由下降沿触发。**

➢ **例：INT1 中断请求设置。**

```
        B0BCLR      FP01IRQ     ; 清 INT1 中断请求标志。
        B0BSET      FP01IEN     ; 使能 INT1 中断。
        B0BSET      FGIE        ; 使能 GIE。
```

➢ **例：INT1 中断。**

```
        ORG         8H          ;
        JMP         INT_SERVICE
INT_SERVICE:
        …                       ; 保存 ACC 和 PFLAG。
        B0BTS1      FP01IRQ     ; 检查是否有 P01 中断请求标志。
        JMP         EXIT_INT    ; P01IRQ = 0，退出中断。
        B0BCLR      FP01IRQ     ; 清 P01IRQ。
        …                       ; INT1 中断服务程序。
        …
EXIT_INT:
        …                       ; 恢复 ACC 和 PFLAG。
        RETI                    ; 退出中断。
```

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 TC1 中断

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

➢ **例：设置 TC1 中断请求。**

```
        B0BCLR      FTC1IEN         ; 禁止 TC1 中断。
        B0BCLR      FTC1ENB         ; 关闭 TC1 定时器。
        MOV         A, # 20H        ;
        B0MOV       TC1M, A         ; 设置 TC1 时钟=Fcpu / 64。
        MOV         A, # 64H        ; 设置 TC1C 初始值=64H。
        B0MOV       TC1C, A         ; 设置 TC1 间隔时间=10 ms。

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

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：TC1 中断服务程序。**

```
        ORG         8H              ;
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; 保存 ACC 和 PFLAG。

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

        B0BCLR      FTC1IRQ         ; 清 TC1IRQ。
        MOV         A, #64H
        B0MOV       TC1C, A         ; 清 TC1C。
        …                           ; TC1 中断服务程序。
        …
EXIT_INT:
        …                           ; 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 多中断操作举例

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	下降沿触发
TC0IRQ	TC0C 溢出
TC1IRQ	TC1C 溢出
ADCIRQ	AD 转换完成

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级。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 P00 中断请求。
                B0BTS1      FP00IEN         ; 检查是否使能 P00 中断。
                JMP         INTP01CHK       ; 跳到下一个中断。
                B0BTS0      FP00IRQ         ; 检查是否有 P00 中断请求。
                JMP         INTP00          ; 进入 INT0 中断。
INTP01CHK:                                  ; 检查是否有 P01 中断请求。
                B0BTS1      FP01IEN         ; 检查是否使能 P01 中断。
                JMP         INTTC0CHK       ; 跳到下一个中断。
                B0BTS0      FP01IRQ         ; 检查是否有 P01 中断请求。
                JMP         INTP01          ; 进入 INT1 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         INTTC1CHK       ; 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。
INTTC1CHK:                                  ; 检查是否有 TC1 中断请求。
                B0BTS1      FTC1IEN         ; 检查是否使能 TC1 中断。
                JMP         INTADCHK        ; 跳到下一个中断。
                B0BTS0      FTC1IRQ         ; 检查是否有 TC1 中断请求。
                JMP         INTTC1          ; 进入 TC1 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口

7.1 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	-	-	-	-	P03M	P02M	P01M	P00M
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	0	0	0	0

0C4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4M	-	-	-	P44M	P43M	P42M	P42M	P40M
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	-	-	P54M	P53M	-	-	-
读/写	-	-	-	R/W	R/W	-	-	-
复位后	-	-	-	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：
1、用户可以用位操作指令（B0BSET，B0BCLR）对 I/O 口进行操作；
2、P0.4 是单向输入引脚， P0.4M = 1。

➢ **例：I/O 模式选择。**

```
CLR     P0M         ; 所有端口置为输入模式。
CLR     P4M
CLR     P5M

MOV     A, #0FFH    ; 所有端口置为输出模式。
B0MOV   P0M, A
B0MOV   P4M,A
B0MOV   P5M, A

B0BCLR  P4M.0       ; P4.0 置为输入模式。

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

7.2 I/O上拉电阻寄存器

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	-	-	-	-	P03R	P02R	P01R	P00R
读/写	-	-	-	-	W	W	W	W
复位后	-	-	-	-	0	0	0	0

0E4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4UR	-	-	-	P44R	P43R	P42R	P41R	P40R
读/写	-	-	-	W	W	W	W	W
复位后	-	-	-	0	0	0	0	0

0E5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5UR	-	-	-	P54R	P53R	-	-	-
读/写	-	-	-	W	W	-	-	-
复位后	-	-	-	0	0	-	-	-

✷ **注：P0.4 是单向输入引脚，无上拉电阻，因此 P0UR.4 始终为“1”。**

➢ **例：I/O 上拉电阻。**

```
MOV     A, #0FFH        ; 使能 P0、4、5 上拉电阻。
B0MOV   P0UR, A         ;
B0MOV   P4UR,A
B0MOV   P5UR, A
```

7.3 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	-	-	-	P04	P03	P02	P01	P00
读/写	-	-	-	R	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

0D4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4	-	-	-	P44	P43	P42	P41	P40
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	-	-	-	P54	P53	-	-	-
读/写	-	-	-	R/W	R/W	-	-	-
复位后	-	-	-	0	0	-	-	-

✷ **注：使能外部复位时，P04 的值保持为“1”。**

➢ **例：从输入端口读取数据。**

```
B0MOV   A, P0     ; 读 P0 口的数据。
B0MOV   A, P4     ; 读 P4 口的数据。
B0MOV   A, P5     ; 读 P5 口的数据。
```

➢ **例：写数据到输出端口。**

```
MOV     A, #0FFH  ; 写 0FFH 到所有的端口。
B0MOV   P0, A
B0MOV   P4, A
B0MOV   P5, A
```

➢ **例：写 1 位数据到输出端口。**

```
B0BSET  P4.0      ; P4.0 和 P5.3 设为“1”。
B0BSET  P5.3

B0BCLR  P4.0      ; P4.0 和 P5.3 为设“0”。
B0BCLR  P5.3
```

7.4 P4 口ADC共用引脚

P4 口和 ADC 的输入口共用，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	-	-	-	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

Bit[4:0] **P4CON[4:0]**：P4.n 控制位。
0 = P4.n 作为模拟信号输入或普通 I/O 引脚；
1 = P4.n 作为仅作模拟信号输入引脚。

✷ **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	-	CHS2	CHS1	CHS0
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 4 **GCHS**：ADC 输入通道控制位。
0 = 禁止 AIN 通道；
1 = 开启 AIN 通道。

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。

✷ **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，CHS[2:0]所指向的 P4.n 会被自动设为 ADC 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       ; CHS[2:0]指向 P4.1（CHS[2:0] = 001B），则 GCHS=0。
                                ; CHS[2:0]没指向 P4.1（CHS[2:0] ≠ 001B），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       ; CHS[2:0]指向 P4.1（CHS[2:0]=001B），则 GCHS=0。
                                ; CHS[2:0]不指向 P4.1（CHS[2:0]≠001B），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           ; P4.1 设为输入模式。
```

P4.0 可作为普通的 I/O 引脚，ADC 输入（AIN0）和 ADC 外部参考电压的高电平输入端 。VERFH 寄存器的 EVHENB 位是 ADC 的外部参考电压的高电平输入控制位。若使能 EVHENB，P4.0 的普通 I/O 功能和 ADC 输入（AIN0）功能被禁止。P4.0 和 ADC 的参考电压输入端直接相连。

✷ 注：若想使能 P4.0 的普通 I/O 功能和 AIN0 功能，必须将 EVHENB 设置为“0”。

0AFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
VREFH	EVHENB	-	-	-	-	-	VHS1	VHS0
读/写	R/W	-	-	-	-	-	R/W	R/W
复位后	0	-	-	-	-	-	0	0

Bit 7 **EVHENB：** ADC 外部参考电压的高电平输入控制位。
0 = 禁止 ADC 参考电压的高电平输入；
1 = 允许 ADC 参考电压的高电平输入。

➢ **例：设置 P4.0 为普通输入模式， EVHENB 和 P4CON.0 必须置 0。**

; 检查 EVHENB 状态。

```
        B0BTS0      FEVHENB         ;
        B0BCLR      FEVHENB         ; EVHENB=1，清 EVHENB 并禁止 ADC 外部参考电压的高电平输入。
                                    ; EVHENB = 0，执行下一段程序。
```

; 检查 GCHS 和 CHS[2:0]的状态。

```
        B0BCLR      FGCHS           ; CHS[2:0]指向 P4.0（CHS[2:0] = 000B），则 GCHS=0。
                                    ; CHS[2:0]不指向 P4.0（CHS[2:0]≠000B），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           ; P4.0 置为输入模式。
```

➢ **例：设置 P4.0 为普通输出模式，EVHENB 和 P4CON.0 位必须置 0。**

; 检查 EVHENB 的状态。

```
        B0BTS0      FEVHENB         ;
        B0BCLR      FEVHENB         ; EVHENB=1，清 EVHENB 并禁止 ADC 外部参考电压的高电平输入。
                                    ; EVHENB = 0，执行下一段程序。
```

; 检查 GCHS 和 CHS[2:0]的状态。

```
        B0BCLR      FGCHS           ; CHS[2:0]指向 P4.0（CHS[2:0] = 000B），设置 GCHS=0。
                                    ; CHS[2:0]不指向 P4.0（CHS[2:0]≠000B），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。

```
        B0BSET      P4.0            ; 设置 P4.0 为 1。
```

; 或

```
        B0BCLR      P4.0            ; 设置 P4.0 为 0。
```

; 设置 P4.0 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（16KHz @3V，32KHz @5V）提供。

看门狗溢出时间 = 8192 /内部低速振荡器周期（sec）

VDD	内部低速 RC Freq.	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：如下是对看门狗定时器的操作，在主程序开头对看门狗清零。**

```
        MOV     A,#5AH          ; 看门狗定时器清零。
        B0MOV   WDTR,A
        …
        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：如下是对看门狗定时器的操作，在主程序开头对看门狗清零。**

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

Correct:                        ; I/O 和 RAM 正常，看门狗清零 。
                                ;
        MOV     A, #5AH         ; 在整个程序中只有一处地方清看门狗。
        B0MOV   WDTR, A
        …
        CALL    SUB1
        CALL    SUB2
        …
        …
        JMP     MAIN
```

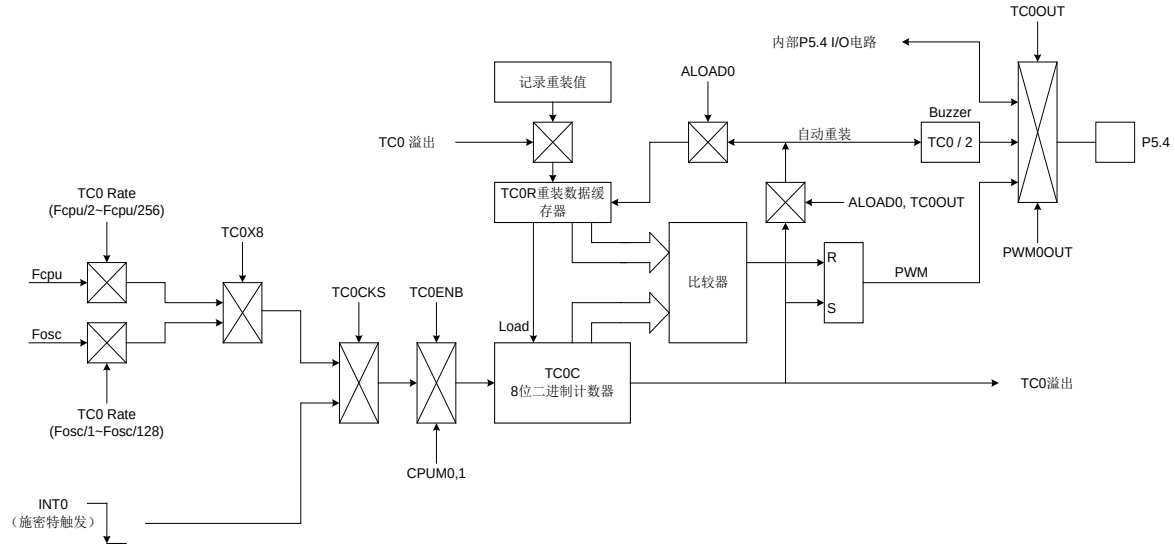
8.2 定时/计数器TC0

8.2.1 概述

8 位二进制定时/计数器具有基本定时器、事件计数器、PWM 和 Buzeer 功能。基本定时器功能可以支持标志显示（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 还内置周期/占空比可编程控制的 PWM 功能，PWM 的周期和分辨率由 TC0M 和 TC0R 寄存器控制。TC0 还内置 Buzzer 功能，以输出 TC0/2 信号。TC0 支持自动重装功能。TC0 溢出时，TC0R 的值自动装入 TC0C。TC0 内置绿色模式唤醒功能，由 TC0GN 控制。

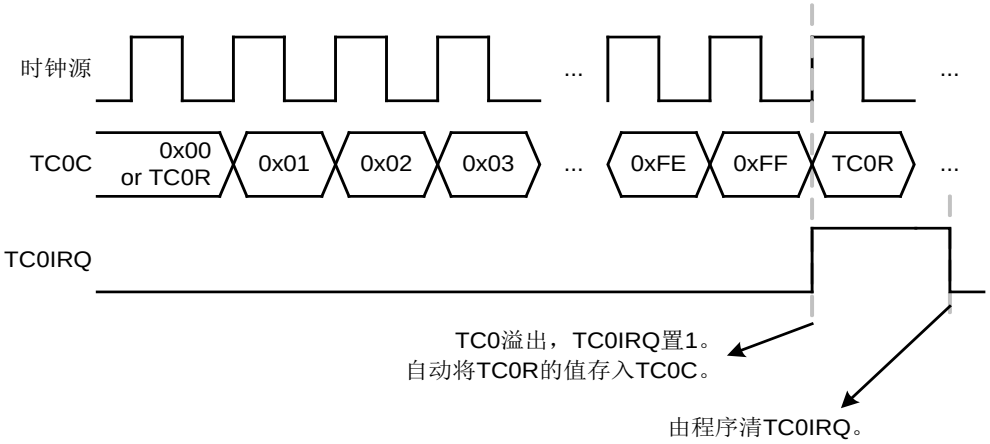
TC0 的主要用途如下：

- **8 位可编程定时器：**根据选择的时钟信号，产生周期中断；
- **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，TC0R 寄存器和 TC0M 寄存器的 ALOAD0 和 TC0OUT 位控制占空比/周期；
- **Buzzer 输出：**Buzzer 输出信号为 TC0 间隔时间的 1/2 周期；
- **绿色模式功能：**TC0 溢出时，TC0 内置绿色模式唤醒功能，由 TC0GN 控制。



8.2.2 TC0 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 TC0C。进入下一个周期后，TC0 按新的配置工作。使能 TC0 时，自动使能 TC0 的自动重装功能。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，设置 TC0IRQ 和 PWM 输出、Buzzer 功能，由 TC0GN 控制将系统唤醒。



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）、Fhosc（高速振荡时钟）和外部引脚输入（P0.0）提供，由 TC0CKS 和 TC0X8 控制。TC0X8 选择时钟源来自 Fcpu 或者 Fhosc，当 TC0X8=0 时，TC0 时钟源来自 Fcpu，可以由 TC0Rate[2:0]选择不同的分频。当 TC0X8=1 时，TC0 时钟源来自 Fhosc，可以由 TC0Rate[2:0]选择不同的分频。TC0CKS 决定时钟源由外部引脚输入或者由 TC0X8 控制，TC0CKS=0 时，TC0 的时钟源由 TC0X8 控制，TC0CKS=1 时，TC0 时钟源由外部输入引脚提供，此时使能外部事件计数功能。TC0X8=1 时，TC0Rate[2:0]处于无效状态。

TC0CKS1	TC0rate[2:0]	TC0 时钟	TC0 间隔时间			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/256	16.384	64	65.536	256
0	001b	Fcpu/128	8.192	32	32.768	128
0	010b	Fcpu/64	4.096	16	16.384	64
0	011b	Fcpu/32	2.048	8	8.192	32
0	100b	Fcpu/16	1.024	4	4.096	16
0	101b	Fcpu/8	0.512	2	2.048	8
0	110b	Fcpu/4	0.256	1	1.024	4
0	111b	Fcpu/2	0.128	0.5	0.512	2
1	000b	Fhosc/128	2.048	8	8.192	32
1	001b	Fhosc/64	1.024	4	4.096	16
1	010b	Fhosc/32	0.512	2	2.048	8
1	011b	Fhosc/16	0.256	1	1.024	4
1	100b	Fhosc/8	0.128	0.5	0.512	2
1	101b	Fhosc/4	0.064	0.25	0.256	1
1	110b	Fhosc/2	0.032	0.125	0.128	0.5
1	111b	Fhosc/1	0.016	0.0625	0.064	0.25

8.2.3 TC0M模式寄存器

模式寄存器 TC0M 控制 TC0 的工作模式，包括 TC0 分频、时钟源和 PWM 功能等。 这些设置必须在使能 TC0 定时器之前完成。

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。
0 = 禁止 PWM 输出；
1 = 使能 PWM 输出，PWM 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 作为输入/输出口；
1 = 允许，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 TC0 自动重装；
1 =允许 TC0 自动重装。

Bit 3 **TC0CKS**：TC0 时钟信号控制位。
0 = 内部时钟（Fcpu 或 Fosc）；
1 = 外部时钟，由 P0.0/INT0 输入。

Bit [6:4] **TC0RATE[2:0]**：TC0 分频选择位。

TC0RATE [2:0]	TC0X8 = 0	TC0X8 = 1
000	Fcpu / 256	Fosc / 128
001	Fcpu / 128	Fosc / 64
010	Fcpu / 64	Fosc / 32
011	Fcpu / 32	Fosc / 16
100	Fcpu / 16	Fosc / 8
101	Fcpu / 8	Fosc / 4
110	Fcpu / 4	Fosc / 2
111	Fcpu / 2	Fosc / 1

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

✷ **注：若 TC0CKS=1，则 TC0 用作外部事件计数器，此时不需要考虑 TC0RATE 的设置，P0.0 口无中断信号（P00IRQ=0）。**

8.2.4 TC0X8, TC0GN标志

TC0 时钟源现在和绿色模式唤醒功能有 T0M 寄存器控制。

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	-	-	-	-	TC1X8	TC0X8	TC0GN	-
读/写	-	-	-	-	R/W	R/W	R/W	-
复位后	-	-	-	-	0	0	0	-

Bit 1 **TC0GN**：TC0 绿色模式唤醒功能控制位。
0 = 禁止 TC0 的唤醒功能；
1 = 允许 TC0 的唤醒功能。

Bit 2 **TC0X8**：TC0 内部时钟选择控制位。
0 = TC0 内部时钟来自 Fcpu，TC0RATE = Fcpu/2~Fcpu/256；
1 = TC0 内部时钟来自 Fhosc，TC0RATE = Fosc/1~Fosc/128。

✷ **注：TC0CKS = 1 时，TC0X8 和 TC0RATE 无效。**

8.2.5 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 初始值计算公式如下：

TC0C 初始值 = N -（TC0 中断间隔时间 * 输入时钟）

N 为 TC0 二进制计数范围。各模式下参数的设定如下表所示：

TC0CKS	TC0X8	PWM0	ALOAD0	TC0OUT	N	TC0C 有效值	TC0C 二进制计数范围	备注
0	0 (Fcpu/2~Fcpu/256)	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 (Fosc/1~Fosc/128)	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

8.2.6 TC0R自动装载寄存器

TC0 内置自动重装功能，TC0R 寄存器存储重装值。TC0C 溢出时，TC0R 的值自动装入 TC0C 中。TC0 定时器工作在计时模式时，要通过修改 TC0R 寄存器来修改 TC0 的间隔时间，而不是通过修改 TC0C 寄存器。在 TC0 定时器溢出后，新的 TC0C 值会被更新，TC0R 会将新的值装载到 TC0C 寄存器中。但在初次设置 TC0M 时，必须要在开启 TC0 定时器前把 TC0C 以及 TC0R 设置成相同的值。

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 误动作。

✷ **注：在 PWM 模式下，系统自动开启重装功能，ALOAD0 用于控制溢出范围。**

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 = 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：

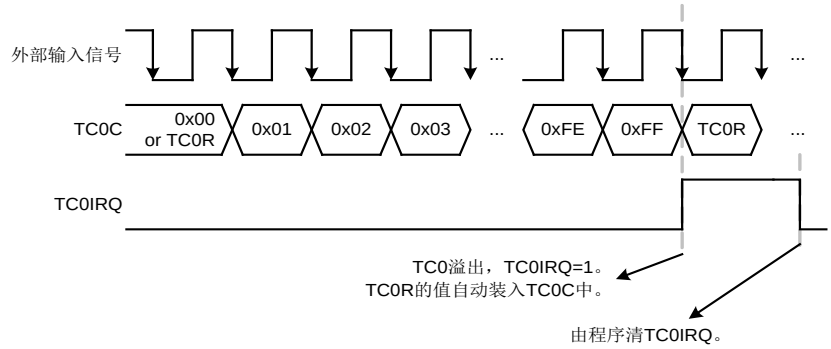
TC0CKS	TC0X8	PWM0	ALOAD0	TC0OUT	N	TC0R 有效值	TC0R 二进制有效范围
0	0 (Fcpu/2~Fcpu/256)	0	x	x	256	00H~0FFH	00000000b~11111111b
		1	0	0	256	00H~0FFH	00000000b~11111111b
		1	0	1	64	00H~3FH	xx000000b~xx111111b
		1	1	0	32	00H~1FH	xxx00000b~xxx11111b
		1	1	1	16	00H~0FH	xxxx0000b~xxxx1111b
	1 (Fosc/1~Fosc/128)	0	x	x	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 中断间隔时间设置为 10ms，时钟源选 Fcpu（TC0KS=0，TC0X8 = 0），无 PWM 输出（PWM0=0），高速时钟为外部 4MHz，Fcpu=Fosc/4，TC0RATE=010（Fcpu/64）。**

TC0R = N -（TC0 中断间隔时间 * 输入时钟）
= 256 -（10ms * 4MHz / 4 / 64）
= 256 -（10^{-2} * 4 * 10^{6} / 4 / 64）
= 100
= 64H

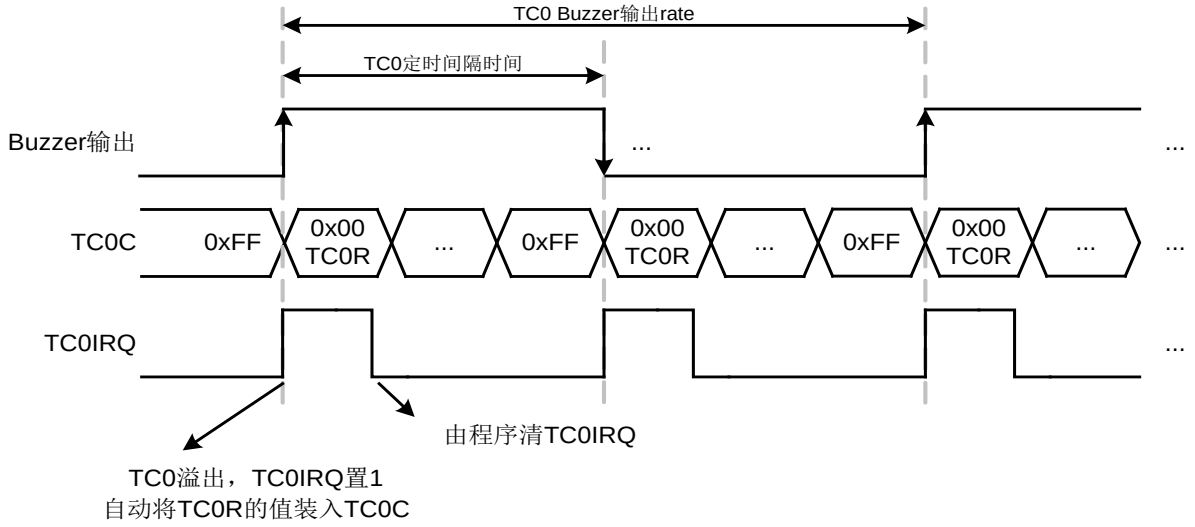
8.2.7 TC0 事件计数器

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 TC0CKS=1 时，TC0 的时钟源由外部输入引脚（P0.0）提供，下降沿触发，下降沿触发时，TC0C 开始计数。TC0C 溢出（从 FFH 到 00H）时， TC0 触发事件计数器溢出。使能外部事件计数功能，同时外部输入引脚的唤醒功能被禁止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，通过 TC0 事件计数器的测量和计算以达到不同的应用。



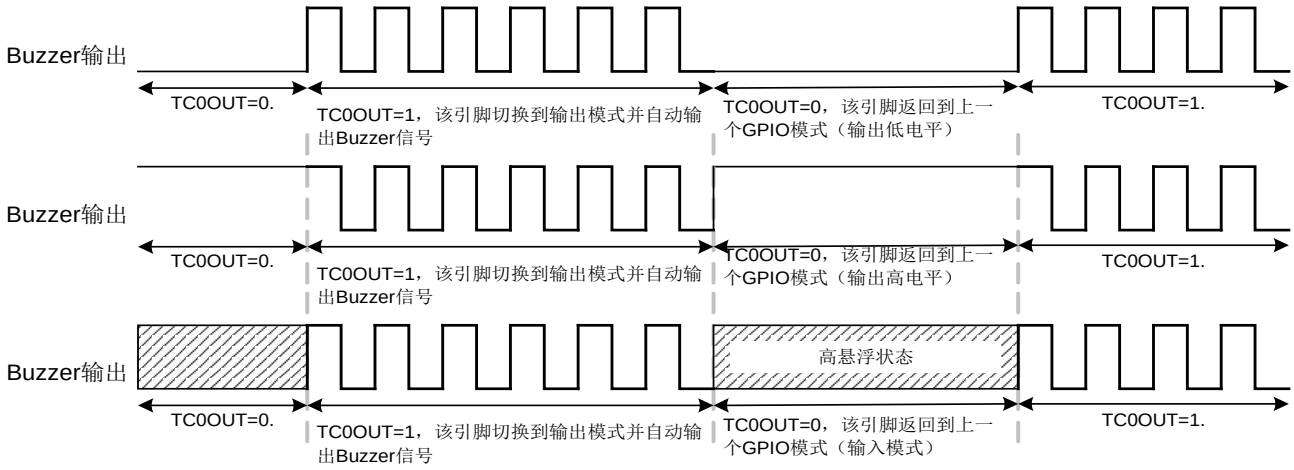
8.2.8 TC0 时钟频率输出（BUZZER）

Buzzer 输出（TC0OUT）为定时/计数器 TC0 频率输出功能，通过设置 TC0 时钟频率，时钟信号输出到 P5.4，此时自动禁止 P5.4 的普通 I/O 功能。TC0 间隔时间 2 分频后作为 TC0OUT 频率。通过 TC0 时钟可以获得不同的频率。TC0OUT 频率波形图如下所示：



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。



若外部高速时钟选择 4MHz，系统时钟源采用外部时钟 Fosc/4，程序中设置 TC0RATE2~TC0RATE1 = 110， TC0C = TC0R = 131，则 TC0 的溢出频率为 2KHz，TC0OUT 的输出频率为 1KHz。下面给出范例程序。

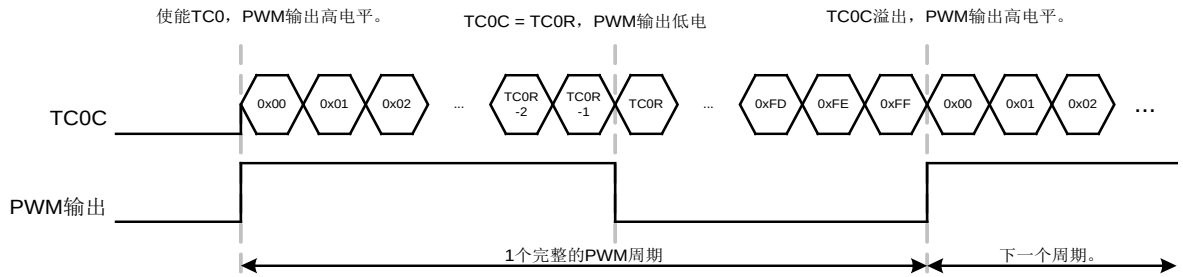
➢ **例：设置 TC0OUT（P5.4）。**

```
MOV     A,#01100000B
B0MOV   TC0M,A          ; TC0 速率=Fcpu/4。

MOV     A,#131          ; 自动加载参考值设置。
B0MOV   TC0C,A
B0MOV   TC0R,A

B0BSET  FTC0OUT         ; TC0 的输出信号由 P5.4 输出，禁止 P5.4 的普通 I/O 功能。
B0BSET  FALOAD0         ; 使能 TC0 自动装载功能。
B0BSET  FTC0ENB         ; 开启 TC0 定时器。
```

✷ **注：蜂鸣器的输出有效时，“PWM0OUT”必须被置为 0。**

8.2.9 脉冲宽度调制（PWM）

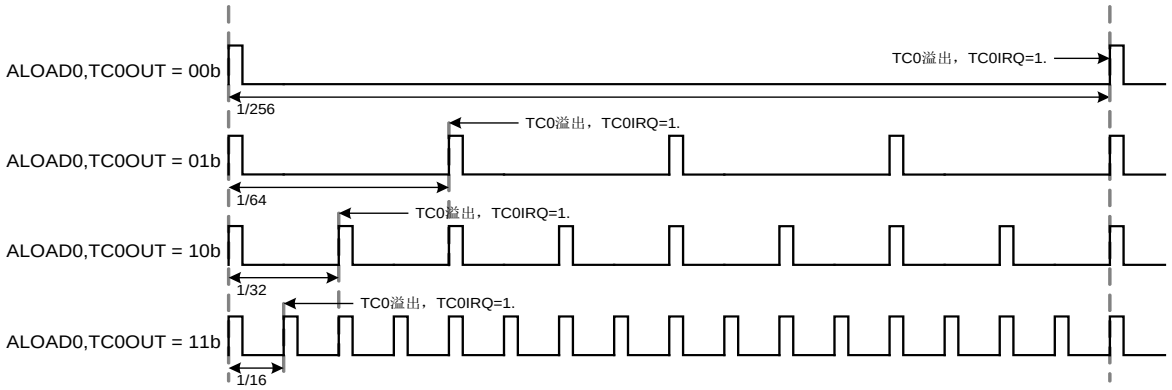
可编程控制占空比/周期的 PWM 可以提供不同的 PWM 信号。使能 TC0 定时器且 PWM0OUT=1 时，由 PWM 输出引脚输出 PWM 信号。PWM 首先输出高电平，然后输出低电平。TC0RATE、ALOAD0 和 TC0OUT 位控制 PWM 的周期，TC0R 控制 PWM 的占空比（脉冲高电平的长度）。使能 TC0 定时器时，设置 TC0C 的初始值为 0。当 TC0C=TC0R 时，PWM 输出低电平；TC0 溢出时（TC0C 的值从 0FFH 到 00H），整个 PWM 周期完成，并进入下一个周期。在 PWM 输出的过程由程序更改 PWM 的周期，则在下一个周期开始输出新的占空比的 PWM 信号。



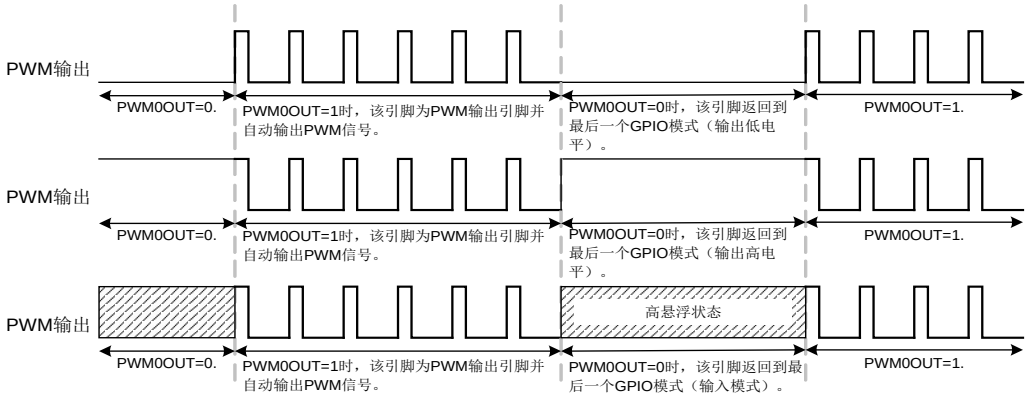
PWM 内置 4 种可编程控制的分辨率（1/256、1/64、1/32、1/16），在 PWM0OUT = 1 时由 ALOAD0 和 TC0OUT 位控制。

PWM0	ALOAD0	TC0OUT	PWM 分辨率	TC0R 有效值	TC0R 值（二进制）
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

PWM 的高电平宽度（PWM 占空比）由 TC0R 控制。TC0C = TC0R 时，PWM 输出低电平。PWM 输出过程中，TC0 溢出时，TC0IRQ 有效，TC0IEN=1 时，即使能 TC0 中断时，PWM 模式下的 TC0 中断间隔时间与 PWM 的周期相等，即 TC0 中断区域有 4 种不同的分辨率和 ALOAD0、TC0OUT 值。但强烈建议小心同时使用 PWM 和 TC0 定时器功能，保证两种功能都能正常工作。



PWM 的输出引脚与 GPIO 共用，PWM0OUT=1 时，自动输出 PWM 信号；PWM0OUT=0，即禁止 PWM 时，该引脚自动返回到上一个 GPIO 模式。这样有利于处理 ON/OFF 操作的载波信号，而不控制 TC0ENB 位。



8.2.10 TC0 操作举例

- **TC0 定时器**

```
; 复位 TC0。
        MOV     A, #00H         ; 清 TC0M。
        B0MOV   TC0M, A

; 设置 TC0Rate 和自动重装功能。
        MOV     A, #0nnn0000b   ; 设置 TC0RATE[2:0]。
        B0MOV   TC0M, A
        B0BCLR  FTC0AD0

; 设置 TC0C 和 TC0R 获得 TC0 的间隔时间。
        MOV     A, #value
        B0MOV   TC0C, A
        B0MOV   TC0R, A

; 清 TC0IRQ。
        B0BCLR  FTC0IRQ

; 选择 TC0 Fcpu / Fhosc 内部时钟源。
        B0BCLR  FTC0X8          ; 选择 Fcpu。
或
        B0BSET  FTC0X8          ; 选择 Fhosc。

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

- **TC0 事件计数器**

```
; 复位 TC0。
        MOV     A, #00H         ; 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 的时钟源由外部输入引脚（P0.0）提供。

; 设置 TC0C 和 TC0R 寄存器获得 TC0 的间隔时间。
        MOV     A, #value       ; TC0C 必须和 TC0R 相等。
        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。
        B0BSET  FTC0ENB         ; 使能 TC0 定时器。
```

- **TC0 BUZZER 输出**

```
; 复位 TC0。
        MOV     A, #00H         ; 清 TC0M。
        B0MOV   TC0M, A

; 设置 TC0Rate 和自动重装功能
        MOV     A, #0nnn0000b   ; 设置 TC0RATE[2:0]。
        B0MOV   TC0M, A
        B0BSET  FALOAD0

; 设置 TC0C 和 TC0R 获得 TC0 的间隔时间。
        MOV     A, #value
        B0MOV   TC0C, A
        B0MOV   TC0R, A

; 使能 Buzzer 和 TC0 定时器。
        B0BSET  FTC0OUT         ; 使能 Buzzer。
        B0BSET  FTC0ENB         ; 使能 TC0 定时器。
```

- **TC0 PWM**

```
; 复位 TC0。
        MOV     A, #00H         ; 清 TC0M。
        B0MOV   TC0M, A

; 设置 TC0Rate 和 PWM 周期
        MOV     A, #0nnn0000b   ; 设置 TC0RATE[2:0]。
        B0MOV   TC0M, A

; 设置 PWM 分辨率。
        MOV     A, #00000nn0b   ; ALOAD0 和 TC0OUT 位。
        OR      TC0M, A

; 设置 TC0R 寄存器获得 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。
```

- **设置 TC0 绿色模式唤醒功能**

```
        B0BSET  FTC0GN          ; 使能 TC0 绿色模式唤醒功能。
```

✷ **注：TC0 外部时钟源模式下，TC0X8 处于无效状态。**

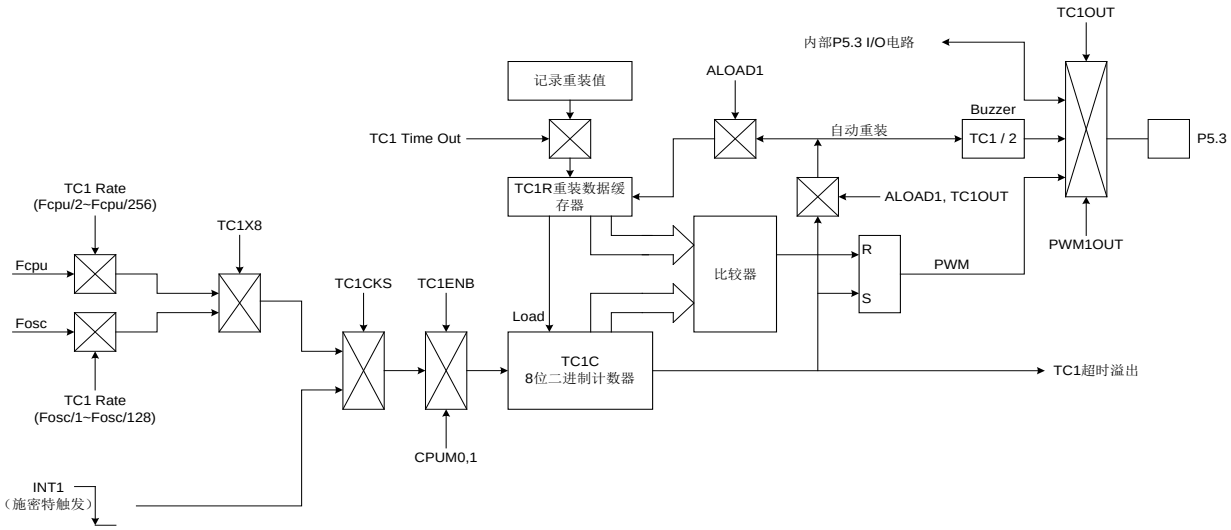
8.3 定时/计数器TC1

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、PWM 和 Buzeer 功能。基本定时器功能可以支持标志显示（TC1IRQ）和中断操作（中断向量）。由 TC1M、TC1C、TC1R 寄存器控制 TC1 的中断间隔时间。事件计数器可以将 TC1 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）。TC1 作为计数器时记录外部时钟数目以进行测量应用。TC1 还内置周期/占空比可编程控制的 PWM 功能，PWM 的周期和分辨率由 TC1M 和 TC1R 寄存器控制。TC1 还内置 Buzzer 功能，以输出 TC1/2 信号。TC1 支持自动重装功能。TC1 溢出时，TC1R 的值自动装入 TC1C。

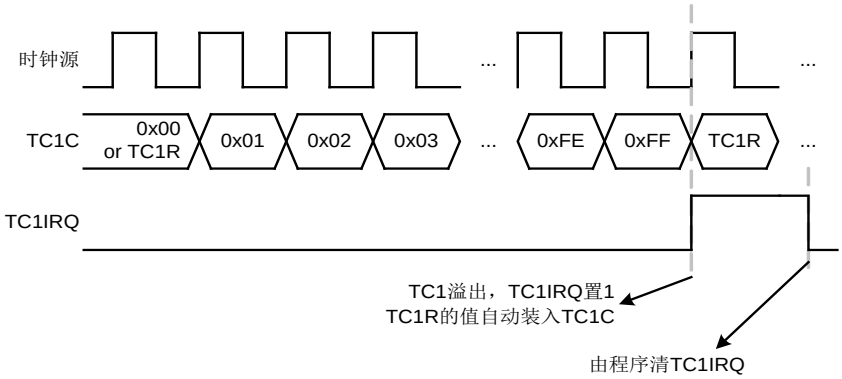
TC0 的主要用途如下：

- **8 位可编程定时器：**根据选择的时钟信号，产生周期中断；
- **中断功能：**TC1 定时器支持中断，当 TC1 溢出时，TC1IRQ 置 1，系统执行中断；
- **外部事件计数器：**对外部事件计数；
- **PWM 输出：**由 TC1rate，TC1R 寄存器和 TC1M 寄存器的 ALOAD1 和 TC1OUT 位控制占空比/周期；
- **Buzzer 输出：**Buzzer 输出信号为 TC1 间隔时间的 1/2 周期；
- **绿色模式功能：**绿色模式下，TC1 正常工作，但不能将系统唤醒。



8.3.2 TC1 操作

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



TC1 根据不同的时钟源选择不同的应用模式，TC1 的时钟源由 Fcpu（指令周期）、Fhosc（高速振荡时钟）提供和外部输入引脚 P0.1 提供，由 TC1CKS 和 TC1X8 位控制。TC1X8 选择时钟源来自 Fcpu 或者 Fhosc，当 TC1X8=0 时，TC1 时钟源来自 Fcpu，可以由 TC1Rate[2:0]选择不同的分频，包括 Fcpu/2~Fcpu/256。当 TC1X8=1 时，TC1 时钟源来自 Fhosc，可以由 TC1Rate[2:0]选择不同的分频，包括 Fcpu/1~Fcpu/128。TC1CKS 选择时钟源来自外部输入引脚或由 TC1X8 位控制，TC1CKS=0 时，TC1 的时钟源由 TC1X8 位控制；TC1CKS=1 时，TC1 的时钟源由外部输入引脚提供，即使能事件计数器功能，此时 TC1Rate[2:0]处于无效状态。

TC1CKS	TC1X8	TC1rate[2:0]	TC1 时钟	TC1 间隔时间			
				Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
				max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	0	000b	Fcpu/256	16.384	64	65.536	256
	0	001b	Fcpu/128	8.192	32	32.768	128
	0	010b	Fcpu/64	4.096	16	16.384	64
	0	011b	Fcpu/32	2.048	8	8.192	32
	0	100b	Fcpu/16	1.024	4	4.096	16
	0	101b	Fcpu/8	0.512	2	2.048	8
	0	110b	Fcpu/4	0.256	1	1.024	4
	0	111b	Fcpu/2	0.128	0.5	0.512	2
	1	000b	Fcpu/128	2.048	8	8.192	32
	1	001b	Fcpu/64	1.024	4	4.096	16
	1	010b	Fcpu/32	0.512	2	2.048	8
	1	011b	Fcpu/16	0.256	1	1.024	4
	1	100b	Fcpu/8	0.128	0.5	0.512	2
	1	101b	Fcpu/4	0.064	0.25	0.256	1
	1	110b	Fcpu/2	0.032	0.125	0.128	0.5
	1	111b	Fcpu/1	0.016	0.0625	0.064	0.25

8.3.3 TC1M模式寄存器

模式寄存器 TC2M 控制 TC2 的工作模式，包括 TC2 分频、时钟源和 PWM 功能等。 这些设置必须在使能 TC2 定时器之前完成。

0DCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1M	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD1	TC1OUT	PWM1OUT
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 **PWM1OUT**：PWM 输出控制位。
0 = 禁止 PWM 输出，P5.3 为 GPIO 引脚；
1 = 使能 PWM 输出，P5.3 输出 PWM 信号，PWM 输出占空比由 TC1OUT 和 ALOAD1 控制。

Bit 1 **TC1OUT**：TC1 超时输出信号控制。仅当 PWM1OUT = 0 时有效。
0 = 禁止，P5.3 为 GPIO 引脚；
1 = 使能，P5.3 输出 TC1/2 Buzzer 信号。

Bit 2 **ALOAD1**：自动装载控制位。仅当 PWM1OUT = 0 时有效。
0 = 禁止 TC1 自动装载；
1 = 使能 TC1 自动装载。

Bit 3 **TC1CKS**：TC1 时钟源控制位。
0 = 内部时钟（Fcpu 或 Fhosc，由 TC1X8 位控制）；
1 = 外部时钟，由 P0.1/INT1 输入，使能时间计数器功能。**TC1Rate[2:0]位处于无效状态。**

Bit [6:4] **TC1RATE[2:0]**：TC1 分频选择位。

TC1RATE [2:0]	TC1X8 = 0	TC1X8 = 1
000	Fcpu / 256	Fosc / 128
001	Fcpu / 128	Fosc / 64
010	Fcpu / 64	Fosc / 32
011	Fcpu / 32	Fosc / 16
100	Fcpu / 16	Fosc / 8
101	Fcpu / 8	Fosc / 4
110	Fcpu / 4	Fosc / 2
111	Fcpu / 2	Fosc / 1

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

✷ **注：若 TC1CKS=1，则 TC1 用作外部事件计数器，此时不需要考虑 TC1RATE 的设置，P0.1 口无中断信号 （P0.1IRQ=0）。**

8.3.4 TC1X8 标志

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	-	-	-	-	TC1X8	-	-	-
读/写	-	-	-	-	R/W	-	-	-
复位后	-	-	-	-	0	-	-	-

Bit 3 **TC1X8**：TC1 内部时钟选择控制位。
0 = TC1 内部时钟来自 Fcpu，TC1RATE = Fcpu/2~Fcpu/256；
1 = TC1 内部时钟来自 Fhosc，TC1RATE = Fosc/1~Fosc/128。

✷ **注：TC1CKS = 1 时，TC1X8 和 TC1RATE 无效。**

8.3.5 TC1C计数寄存器

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

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

TC1C 初始值的计算公式如下：

TC1C 初始值 = N -（TC1 中断间隔时间 * 输入时钟）

N 为 TC1 二进制计数范围。各模式下参数的设定如下表所示：

TC1CKS	TC1X8	PWM1	ALOAD1	TC1OUT	N	TC1C 有效值	TC1C 二进制计数范围	备注
0	0 (Fcpu/2~ Fcpu/256)	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 (Fosc/1~ Fosc/128)	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

8.3.6 TC1R自动装载寄存器

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

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

✷ **注：在 PWM 模式下，系统自动开启自动重装功能，ALOAD1 用于控制溢出范围。**

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

TC1R 初始值计算公式如下：

TC1R 初始值 = N -（TC1 中断间隔时间 * 输入时钟）

N 是 TC1 最大溢出值。TC1 的溢出时间和有效值见下表：

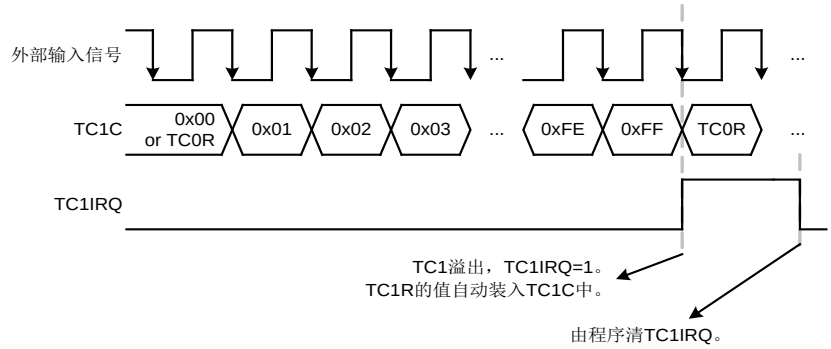
TC1CKS	TC1X8	PWM1	ALOAD1	TC1OUT	N	TC1R 有效值	TC1R 二进制有效范围
0	0 (Fcpu/2~Fcpu/256)	0	x	x	256	00H~0FFH	00000000b~11111111b
		1	0	0	256	00H~0FFH	00000000b~11111111b
		1	0	1	64	00H~3FH	xx000000b~xx111111b
		1	1	0	32	00H~1FH	xxx00000b~xxx11111b
		1	1	1	16	00H~0FH	xxxx0000b~xxxx1111b
	1 (Fosc/1~Fosc/128)	0	x	x	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：TC1 中断间隔时间设置为 10ms，时钟源选 Fcpu（TC1CKS=0，TC1X8 = 0），无 PWM 输出（PWM1=0），高速时钟为外部 4MHz，Fcpu=Fosc/4，TC1RATE=010（Fcpu/64）。**

TC1R 有效值 = N -（TC1 中断间隔时间 * 输入时钟）
= 256 -（10ms * 4MHz / 4 / 64）
= 256 -（10^{-2} * 4 * 10^{6} / 4 / 64）
= 100
= 64H

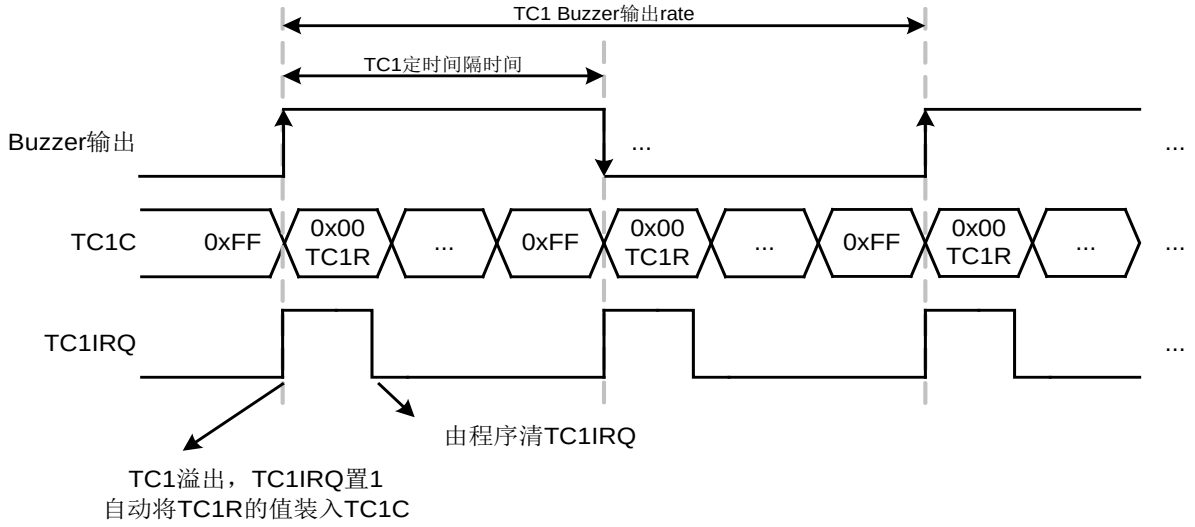
8.3.7 TC1 事件计数器

TC1 作为外部事件计数器时，其时钟源由外部输入引脚（P0.1）提供。当 TC1CKS=1 时，TC1 的时钟源由外部输入引脚（P0.1）提供，下降沿触发，下降沿触发时，TC1C 开始计数。TC1C 溢出（从 FFH 到 00H）时， TC1 触发事件计数器溢出。使能外部事件计数功能，同时外部输入引脚的唤醒功能被禁止以避免外部事件的触发信号将系统唤醒而耗电。此时，P0.1 的外部中断功能也被禁止，即 P01IRQ=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，通过 TC1 事件计数器的测量和计算以达到不同的应用。



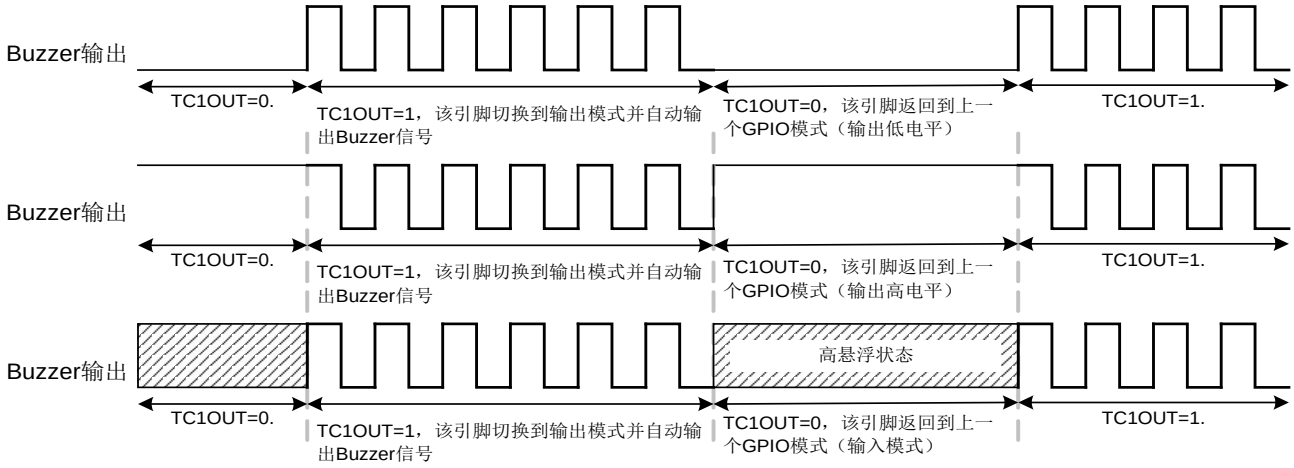
8.3.8 TC1 时钟频率输出（BUZZER）

Buzzer 输出（TC1OUT）为定时/计数器 TC1 频率输出功能，通过设置 TC1 时钟频率，时钟信号输出到 P5.3，此时自动禁止 P5.3 的普通 I/O 功能。TC1 间隔时间 2 分频后作为 TC1OUT 频率。通过 TC1 时钟可以获得不同的频率。TC1OUT 频率波形图如下所示：



TC1 溢出后，Buzzer 输出时，TC1IRQ 有效，且当 TC1IEN=1 时，使能 TC1 中断功能。但强烈建议小心同时使用 Buzzer 和 TC1 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，TC1OUT=1 时，该引脚自动设为 Buzzer 输出引脚。如清 TC1OUT 位以禁止 Buzzer 输出后，该引脚自动返回到最后一个 GPIO 模式。



若外部高速时钟选择 4MHz，系统时钟源采用外部时钟 Fosc/4，程序中设置 TC1RATE2~TC1RATE1 = 110， TC1C = TC1R = 131，则 TC1 的溢出频率为 2KHz，TC1OUT 的输出频率为 1KHz。下面给出范例程序。

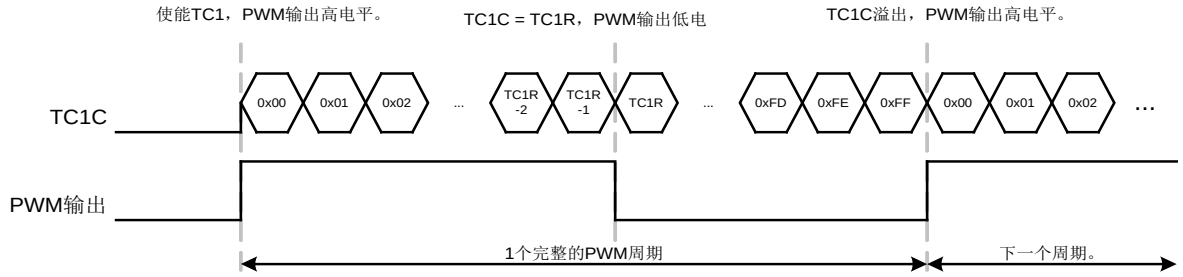
➢ **例：设置 TC1OUT（P5.3）。**

```
MOV     A,#01100000B
B0MOV   TC1M,A          ; TC1 速率=Fcpu/4。

MOV     A,#131          ; 自动加载参考值设置。
B0MOV   TC1C,A
B0MOV   TC1R,A

B0BSET  FTC1OUT         ; TC1 的输出信号由 P5.3 输出，禁止 P5.3 的普通 I/O 功能。
B0BSET  FALOAD1         ; 使能 TC1 自动装载功能。
B0BSET  FTC1ENB         ; 开启 TC1 定时器。
```

✷ **注：蜂鸣器的输出有效时，“PWM1OUT”必须被置为 0。**

8.3.9 脉冲宽度调制（PWM）

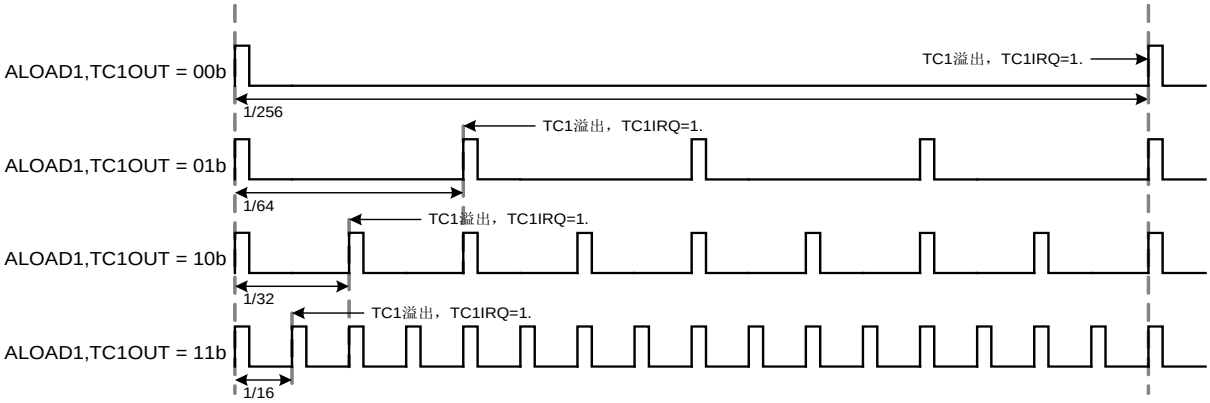
可编程控制占空比/周期的 PWM 可以提供不同的 PWM 信号。使能 TC1 定时器且 PWM1OUT=1 时，由 PWM 输出引脚（P5.3）输出 PWM 信号。PWM 首先输出高电平，然后输出低电平。TC1RATE、ALOAD1 和 TC1OUT 位控制 PWM 的周期，TC1R 控制 PWM 的占空比（脉冲高电平的长度）。使能 TC1 定时器时，设置 TC1C 的初始值为 0。当 TC1C=TC1R 时，PWM 输出低电平；TC1 溢出时（TC1C 的值从 0FFH 到 00H），整个 PWM 周期完成，并进入下一个周期。在 PWM 输出的过程由程序更改 PWM 的周期，则在下一个周期开始输出新的占空比的 PWM 信号。



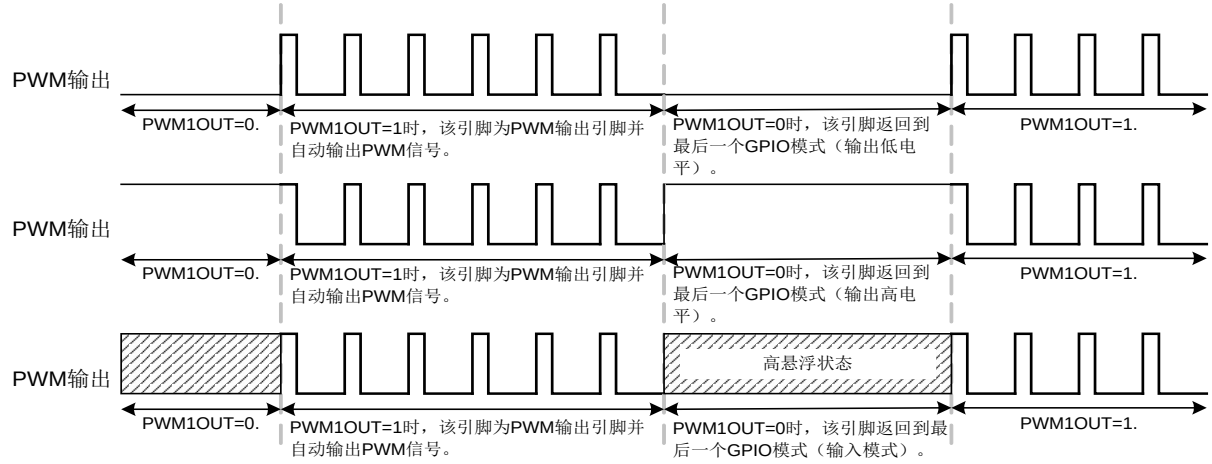
PWM 内置 4 种可编程控制的分辨率（1/256、1/64、1/32、1/16），在 PWM1OUT = 1 时由 ALOAD1 和 TC1OUT 位控制。

PWM1	ALOAD1	TC1OUT	PWM 分辨率	TC1R 有效值	TC1R 值（二进制）
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

PWM 的高电平宽度（PWM 占空比）由 TC1R 控制。TC1C = TC1R 时，PWM 输出低电平。PWM 输出过程中，TC1 溢出时，TC1IRQ 有效，TC1IEN=1 时，即使能 TC1 中断时，PWM 模式下的 TC1 中断间隔时间与 PWM 的周期相等，即 TC1 中断区域有 4 种不同的分辨率和 ALOAD1、TC1OUT 值。但强烈建议小心同时使用 PWM 和 TC0 定时器功能，保证两种功能都能正常工作。



PWM 的输出引脚与 GPIO 共用，PWM1OUT=1 时，自动输出 PWM 信号；PWM1OUT=0，即禁止 PWM 时，该引脚自动返回到上一个 GPIO 模式。这样有利于处理 ON/OFF 操作的载波信号，而不控制 TC1ENB 位。



8.3.10 TC1 操作举例

- **TC1 定时器**

```
; 复位 TC1。
        MOV     A, #00H         ; 清 TC1M。
        B0MOV   TC1M, A

; 设置 TC1Rate 和自动重装功能。
        MOV     A, #0nnn0000b   ; 设置 TC1RATE[2:0]。
        B0MOV   TC1M, A
        B0BCLR  FTC1AD1

; 设置 TC1C 和 TC1R 获得 TC1 的间隔时间。
        MOV     A, #value
        B0MOV   TC1C, A
        B0MOV   TC1R, A

; 清 TC1IRQ。
        B0BCLR  FTC1IRQ

; 选择 TC1 Fcpu / Fhosc 内部时钟源。
        B0BCLR  FTC1X8          ; 选择 Fcpu。
或
        B0BSET  FTC1X8          ; 选择 Fhosc。

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

- **TC1 事件计数器**

```
; 复位 TC1。
        MOV     A, #00H         ; 清 TC1M。
        B0MOV   TC1M, A

; 设置 TC1 自动重装功能。
        B0BSET  FALOAD1

; 使能 TC1 事件计数器。
        B0BSET  FTC1CKS         ; 设置 TC1 的时钟源由外部输入引脚（P0.1）提供。

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

; 清 TC1IRQ。
        B0BCLR  FTC1IRQ

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

● **TC1 BUZZER 输出**

```
; 复位 TC1。
        MOV         A, #00H             ; 清 TC1M。
        B0MOV       TC1M, A

; 设置 TC1Rate 和自动重装功能
        MOV         A, #0nnn0000b       ; 设置 TC1RATE[2:0]。
        B0MOV       TC1M, A
        B0BSET      FALOAD1

; 设置 TC1C 和 TC1R 获得 TC1 的间隔时间。
        MOV         A, #value           ; TC1C 和 TC1R 的值必须相等。
        B0MOV       TC1C, A
        B0MOV       TC1R, A

; 使能 Buzzer 和 TC1 定时器。
        B0BSET      FTC1OUT             ; 使能 Buzzer。
        B0BSET      FTC1ENB             ; 使能 TC1 定时器。
```

● **TC1 PWM**

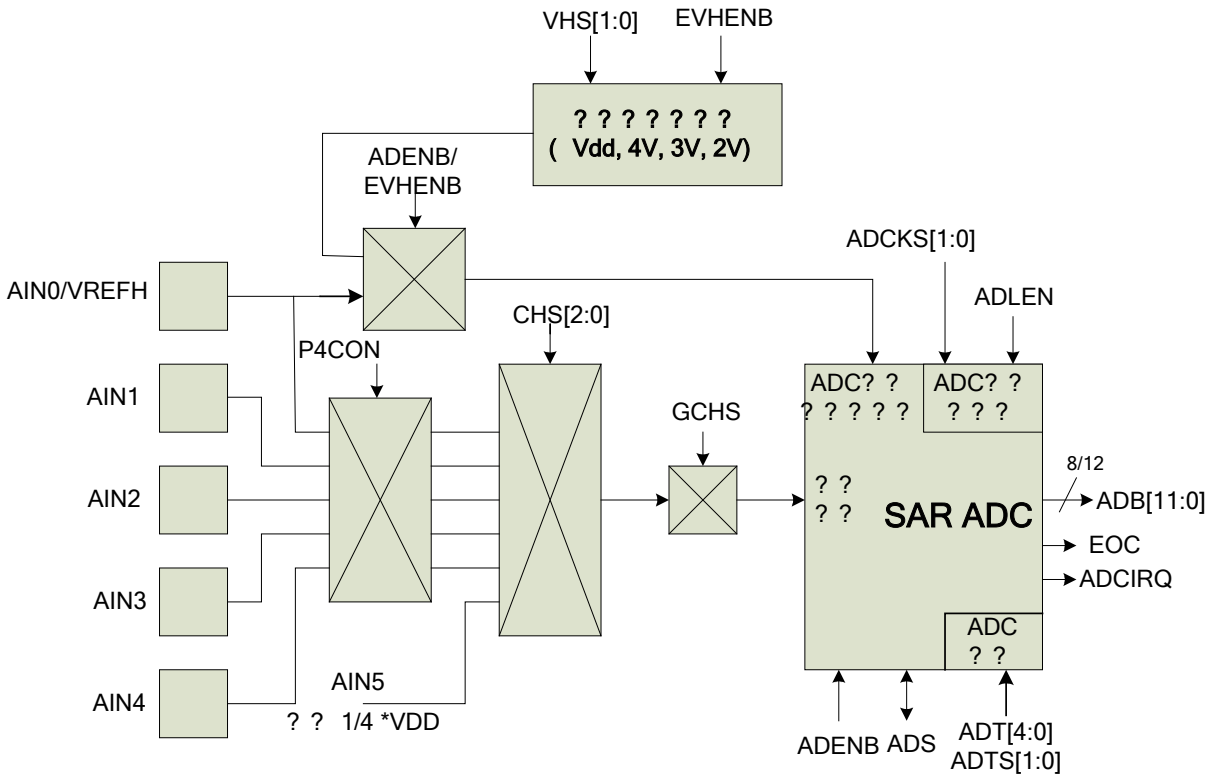
```
; 复位 TC1。
        MOV         A, #00H             ; 清 TC1M。
        B0MOV       TC1M, A

; 设置 TC0Rate 和 PWM 周期
        MOV         A, #0nnn0000b       ; 设置 TC1RATE[2:0]。
        B0MOV       TC1M, A

; 设置 PWM 分辨率。
        MOV         A, #00000nn0b       ; ALOAD0 和 TC0OUT 位。
        OR          TC1M, A

; 设置 TC1R 寄存器获得 PWM 占空比。
        MOV         A, #value
        B0MOV       TC1R, A

; 清 TC1C。
        CLR         TC1C

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

✷ **注：TC1 外部时钟源模式下，TC1X8 处于无效状态。**

9 5+1 通道ADC

9.1 概述

模拟数字转换（ADC）是一个 SAR 结构，内置 6 个模拟通道，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。通过 CHS[2:0]现在模拟信号输入引脚（AIN 引脚），内部 1/4*Vdd 电压源，GCHS 位使能全部 ADC 通道，模拟信号输入至 SAR ADC。ADC 的分辨率为 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 参考电压的高电平包括 2 种，内部参考源，包括 Vdd、4V、3V、2V（EBHENB=0），外部参考源，由 P4.0 提供（EVHENB=1）。ADC 内置 P4CON 寄存器来设置模拟输入引脚，必须由程序将 ADC 输入引脚设为不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，并将转换结果存入 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（ORG 0008H）执行中断服务程序。



✸ 注：
1、设置 ADS 输入引脚为不带上拉电阻的输入模式；
2、进入睡眠模式前禁止 ADC（ADENB=0）以省电；
3、睡眠模式下设置 P4CON 寄存器的相关位以避免额外的功耗；
4、使能 ADC 后（ADENB=1）延时 100us 以等待 ADC 电路稳定。

9.2 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 的参考源选择和 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	-	CHS2	CHS1	CHS0
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 **ADENB**：ADC 控制位。**睡眠模式下，禁止 ADC 以省电。**
0 = 禁止；
1 = 使能。

Bit 6 **ADS**：ADC 启动位。 **ADC 处理完成后，ADS 位自动清零。**
0 = 停止；
1 = 开始。

Bit 5 **EOC**：ADC 状态控制位 。
0 = 转换过程中；
1 = 转换结束，ADS 复位。

Bit 4 **GCHS**：通道选择位。
0 = 禁止 AIN 通道；
1 = 使能 AIN 通道。

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。

AIN5 是内部 1/4 VDD 输入通道，外部没有输入引脚。AIN5 可以作为电池系统的电池检测器。为了选择合适的内部 VREFH 电平并进行比较，系统内置了这个高性能、廉价的低电池检测器。

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	-	ADCKS0	ADB3	ADB2	ADB1	ADB0
读/写	-	R/W	-	R/W	R	R	R	R
复位后	-	0	-	0	X	X	X	X

Bit[6,4] ADCKS1，ADCKS0：**ADC 时钟源选择位。**

ADCKS1	ADCKS0	ADC 时钟源
0	0	Fcpu/16
0	1	Fcpu/8
1	0	Fcpu
1	1	Fcpu/2

9.3 ADB数据缓存器

ADC 数据缓存器共 12 位，用来存储 AD 转换结果，8 位只读寄存器 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[3:0]：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	ADB15	ADB14	ADB13	ADB12	ADB11	ADB10	ADB9	ADB8
读/写	R	R	R	R	R	R	R	R
复位后	X	X	X	X	X	X	X	X

Bit[7:0] **ADB[7:0]**：ADC 12 位分辨率的高字节数据缓存器。

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

Bit[3:0] **ADB[3:0]**：ADC 12 位分辨率的低字节数据缓存器。

AIN 的输入电压 v.s. ADB 的输出数据

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 = 未使用的位												

✷ **注：ADC 缓存器 ADB 在复位后的初始值是未知的。**

9.4 ADC参考电压寄存器

ADC 内置 5 种参考电压，由 VREFH 寄存器控制：包括 1 个外部参考电压和 4 个内部参考源（VDD、4V、3V、2V）。EVHENB = 1 时，ADC 参考电压由外部参考源提供（P4.0），必须输入一个电压作为 ADC 参考电压的高电平，且不能低于 2V。EVHENB = 0 时，ADC 参考电压由内部参考源提供，并由 VHS[1:0]选择控制。VHS[1:0] = 11 时，ADC 参考源选择 VDD；VHS[1:0] = 10 时，ADC 参考源选择 4V；VHS[1:0] = 01 时，ADC 参考源选择 3V；VHS[1:0] = 00 时，ADC 参考源选择 2V。外部参考源的限制条件为，最高为 VDD，最低为内部最低电平，否则默认为 VDD。

0AFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
VREFH	EVHENB	-	-	-	-	-	VHS1	VHS0
读/写	R/W	-	-	-	-	-	R/W	R/W
复位后	0	-	-	-	-	-	0	0

Bit[1:0] **VHS[1:0]：ADC 内部参考电压选择位。**

VHS1	VHS0	内部 VREFH 电压
1	**1**	**VDD**
1	**0**	**4.0V**
0	**1**	**3.0V**
0	**0**	**2.0V**

Bit[7] **EVHENB**：ADC 内部参考电压控制位。

0 = 允许 ADC 内部 VREFH 功能，VREFH 引脚是 P4.0/AIN0 引脚；

1 = 禁止 ADC 内部 VREFH 功能，P4.0/AIN0/VREFH 引脚来自外部 VREFH 输入引脚。

9.5 ADC操作说明和注意事项

9.5.1 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，高电压包括 VDD/4V/3V/2V，外部参考电压由 P4.0/AVREFH 引脚提供（由 EVHENB 控制）。EVHENB = 0 时，ADC 参考电压选择内部参考源；EVHENB = 1 时，ADC 参考电压选择外部参考源（P4.0/AVREFH）。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 内部端口低电压=VDD/4V/3V/2V。（EVHENB = 0）**
- **ADC 外部参考电压=2V~VDD。（EVHENB = 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**

9.5.2 AD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	
			AD 转换时间	AD 转换 Rate	AD 转换时间	AD 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	
			AD 转换时间	AD 转换 Rate	AD 转换时间	AD 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

9.5.3 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。**

EVHENB = 1 时，P4.0/AIN0 为 ADC 外部参考源的输入引脚，此时，P4.0 必须设为输入模式，并禁止其上拉电阻。

- **ADC 外部参考源输入引脚为 GPIO 引脚时必须设为输入模式。**
- **必须禁止 ADC 外部参考源输入引脚的内部上拉电阻。**

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[4:0]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	-	-	-	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
读/写	-	-	-	W	W	W	W	W
复位后	-	-	-	0	0	0	0	0

Bit[4:0] **P4CON[4:0]**：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。

9.6 ADC操作实例

- **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 参考电压高电平源。
        B0BSET  FEVHENB         ; 外部参考源。
Or
        MOV     A, #000000nnb   ; 内部 VDD。
                                ; “nn”选择内部参考源的电平。
                                ; 11 = VDD，10 = 4V，01 = 3V，00 = 2V。

; 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
```

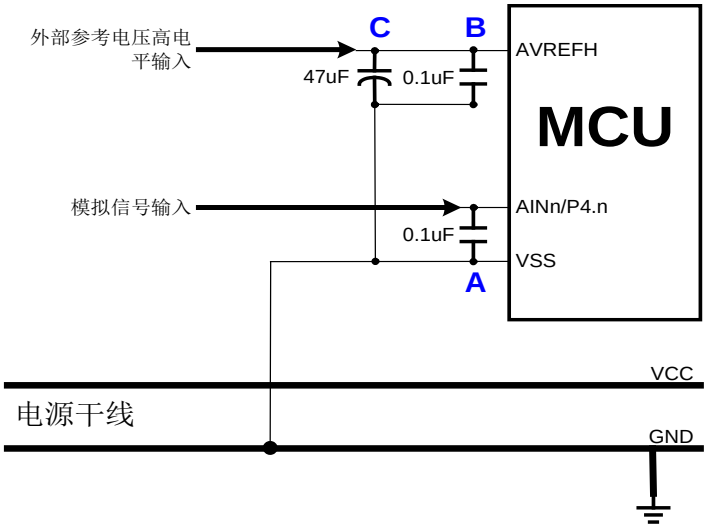
✷ 注：
1、使能 ADENB 后（不是使能 ADS），系统必须延迟 100us 等待启动 ADC，然后设置 ADS 开始 AD 转换，否则 ADC 的结果出错。系统正常运行时，设置 ADENB 一次，延时一次。
2、睡眠模式和绿色模式下禁止 ADC 以省电。

● **ADC 转换：**

```
; 禁止 ADC 中断模式。
@@:
        B0BTS1      FEOC            ; 检查 ADC 状态标志。
        JMP         @B              ; EOC=0：AD 转换中。
        B0MOV       A, ADB          ; EOC=1：AD 转换结束，处理 AD 转换结果。
        B0MOV       BUF1,A
        MOV         A, #00001111b
        AND         A, ADR
        B0MOV       BUF2,A
        …                           ; AD 转换结果处理完成。
        CLR         FEOC            ; 清除 ADC 状态标志以准备下一次 ADC。

; 使能 ADC 中断模式。
        ORG         8               ; 中断向量。
INT_SR:                             ; 中断服务程序。
        PUSH
        B0BTS1      FADCIRQ         ; 检查 ADC 中断标志。
        JMP         EXIT_INT        ; ADCIRQ=0：没有 ADC 中断发生。
        B0MOV       A, ADB          ; ADCIRQ=1：AD 转换结束，处理 AD 转换结果。
        B0MOV       BUF1,A
        MOV         A, #00001111b
        AND         A, ADR
        B0MOV       BUF2,A
        …                           ; AD 转换结果处理完成。
        CLR         FEOC            ; 清除 ADC 状态标志以准备下一次 ADC。
        JMP         INT_EXIT

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

✷ **注：AD 转换结束时自动将 ADS 清零，EOC 即时反映 ADC 的转换状态，ADS=1 时自动清除 EOC，用户无需通过程序来清零。**

9.7 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。

ADC 参考电压高电平由外部参考源提供，外部参考源连接到 AVREFH 引脚（P4.0）。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。

10 指令集

指令	指令格式		描述	C	DC	Z	周期
M O V E	MOV	A,M	A ← M。	-	-	√	1
	MOV	M,A	M ← A。	-	-	-	1
	B0MOV	A,M	A ← M (bank 0)。	-	-	√	1
	B0MOV	M,A	M (bank 0) ← A。	-	-	-	1
	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I。（M 仅适用于系统寄存器 R、Y、Z、RBANK、PFLAG。）	-	-	-	1
	XCH	A,M	A ←→M。	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)。	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]。	-	-	-	2
A R I T H M E T I C	ADC	A,M	A ← A + M + C，如果产生进位则 C=1，否则 C=0。	√	√	√	1
	ADC	M,A	M ← A + M + C，如果产生进位则 C=1，否则 C=0。	√	√	√	1+N
	ADD	A,M	A ← A + M，如果产生进位则 C=1，否则 C=0。	√	√	√	1
	ADD	M,A	M ← A + M，如果产生进位则 C=1，否则 C=0。	√	√	√	1+N
	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
	ADD	A,I	A ← A + I，如果产生进位则 C=1，否则 C=0。	√	√	√	1
	SBC	A,M	A ← A - M - /C，如果产生借位则 C=0，否则 C=1。	√	√	√	1
	SBC	M,A	M ← A - M - /C，如果产生借位则 C=0，否则 C=1。	√	√	√	1+N
	SUB	A,M	A ← A - M，如果产生借位则 C=0，否则 C=1。	√	√	√	1
	SUB	M,A	M ← A - M，如果产生借位则 C=0，否则 C=1。	√	√	√	1+N
	SUB	A,I	A ← A - I，如果产生借位则 C=0，否则 C=1。	√	√	√	1
L O G I C	AND	A,M	A ← A 与 M。	-	-	√	1
	AND	M,A	M ← A 与 M。	-	-	√	1+N
	AND	A,I	A ← A 与 I。	-	-	√	1
	OR	A,M	A ← A 或 M。	-	-	√	1
	OR	M,A	M ← A 或 M。	-	-	√	1+N
	OR	A,I	A ← A 或 I。	-	-	√	1
	XOR	A,M	A ← A 异或 M。	-	-	√	1
	XOR	M,A	M ← A 异或 M。	-	-	√	1+N
	XOR	A,I	A ← A 异或 I。	-	-	√	1
P R O C E S S	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)。	-	-	-	1
	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)。	-	-	-	1+N
	RRC	M	A ← M 带进位右移。	√	-	-	1
	RRCM	M	M ← M 带进位右移。	√	-	-	1+N
	RLC	M	A ← M 带进位左移。	√	-	-	1
	RLCM	M	M ← M 带进位左移。	√	-	-	1+N
	CLR	M	M ← 0。	-	-	-	1
	BCLR	M.b	M.b ← 0。	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0。	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1。	-	-	-	1+N
B R A N C H	CMPRS	A,I	比较，如果相等则跳过下一条指令 C 与 ZF 标志位可能受影响。	√	-	√	1 + S
	CMPRS	A,M	比较，如果相等则跳过下一条指令 C 与 ZF 标志位可能受影响。	√	-	√	1 + S
	INCS	M	A ← M + 1，如果 A = 0，则跳过下一条指令。	-	-	-	1+ S
	INCMS	M	M ← M + 1，如果 M = 0，则跳过下一条指令。	-	-	-	1+N+S
	DECS	M	A ← M - 1，如果 A = 0，则跳过下一条指令。	-	-	-	1+ S
	DECMS	M	M ← M - 1，如果 M = 0，则跳过下一条指令。	-	-	-	1+N+S
	BTS0	M.b	如果 M.b = 0，则跳过下一条指令。	-	-	-	1 + S
	BTS1	M.b	如果 M.b = 1，则跳过下一条指令。	-	-	-	1 + S
	B0BTS0	M.b	如果 M(bank 0).b = 0，则跳过下一条指令。	-	-	-	1 + S
	B0BTS1	M.b	如果 M(bank 0).b = 1，则跳过下一条指令。	-	-	-	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 I S C	RET		子程序跳出指令，PC ← Stack。	-	-	-	2
	RETI		中断处理程序跳出指令，PC ← Stack，使能全局中断控制位。	-	-	-	2
	PUSH		进栈指令，保存 ACC 和工作寄存器。	-	-	-	1
	POP		出栈指令，恢复 ACC 和工作寄存器。	√	√	√	1
	NOP		空指令，无特别意义。	-	-	-	1

注： **1. “M”是系统寄存器或 RAM，M 为系统寄存器时 N = 0，否则 N = 1。**
2.条件跳转指令的条件为真，则 S = 1，否则 S = 0。

11 电气特性

11.1 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)
SN8P2711BP, SN8P2711BS, SN8P27113BA..... 0°C ~ + 70°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

11.2 电气特性

- **DC CHARACTERISTIC**

(All of voltages refer to Vss, Vdd = 5.0V, fosc = 4MHz,fcpu=1MHZ,ambient temperature is 25°C unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd, 25°C		2.4	5.0	5.5	V
		Normal mode, Vpp = Vdd, -40°C~85°C		2.5	5.0	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		0.8Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	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 port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O output source current	IoH	Vop = Vdd – 0.5V		8	12	-	mA
sink current	IoL	Vop = Vss + 0.5V		8	15	-	
* INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current (Disable ADC)	Idd1	Run Mode (No loading, Fcpu = Fosc/4)	Vdd= 5V, 4Mhz	-	2.5	5	mA
			Vdd= 3V, 4Mhz	-	1	2	mA
	Idd2	Slow Mode (Internal low RC, Stop high clock)	Vdd= 5V, 32Khz	-	10	20	uA
			Vdd= 3V, 16Khz	-	5	10	uA
	Idd3	Sleep Mode	Vdd= 5V, 25°C	-	0.8	1.6	uA
			Vdd= 3V, 25°C	-	0.7	1.4	uA
			Vdd= 5V, -40°C~ 85°C	-	10	21	uA
			Vdd= 3V, -40°C~ 85°C	-	10	21	uA
	Idd4	Green Mode (No loading,Fcpu = Fosc/4 Watchdog Disable)	Vdd= 5V, 4Mhz	-	0.75	1.5	mA
			Vdd= 3V, 4Mhz	-	0.35	0.7	mA
			Vdd=5V, ILRC 32Khz	-	5	10	uA
			Vdd=3V, ILRC 16Khz ,	-	2	4	uA
Internal High Oscillator Freq.	Fihrc	Internal Hihg RC (IHRC)	25°C, Vdd= 5V, Fcpu = 1MHz	15.68	16	16.32	Mhz
			-40°C~85°C, Vdd= 2.4V~5.5V, Fcpu = 1MHz~16 MHz	15.2	16	16.8	Mhz
LVD Voltage	Vdet0	Low voltage reset level. 25°C		1.9	2.0	2.1	V
		Low voltage reset level. -40°C~ 85°C		1.8	2.0	2.3	V
	Vdet1	Low voltage reset/indicator level. 25°C		2.3	2.4	2.5	V
		Low voltage reset/indicator level. -40°C~ 85°C		2.2	2.4	2.7	V
	Vdet2	Low voltage reset/indicator level. 25°C		3.5	3.6	3.7	V
		Low voltage reset/indicator level. -40°C~ 85°C		3.3	3.6	3.9	V

*These parameters are for design reference, not tested.

● ADC CHARACTERISTIC

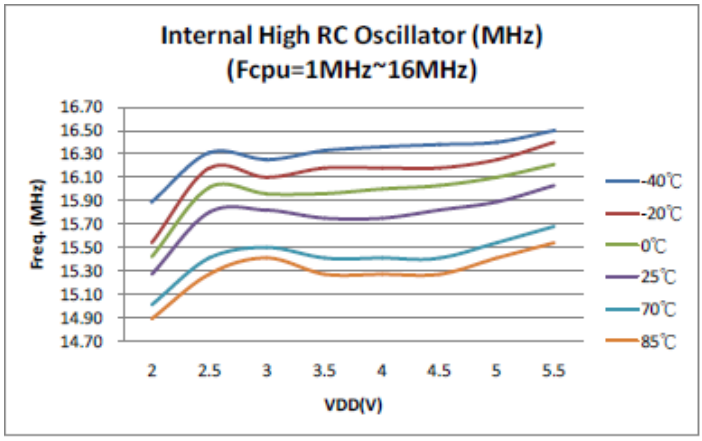
(All of voltages refer to Vss, Vdd = 5.0V, fosc = 4MHz,fcpu=1MHZ,ambient temperature is 25°C unless otherwise note.)

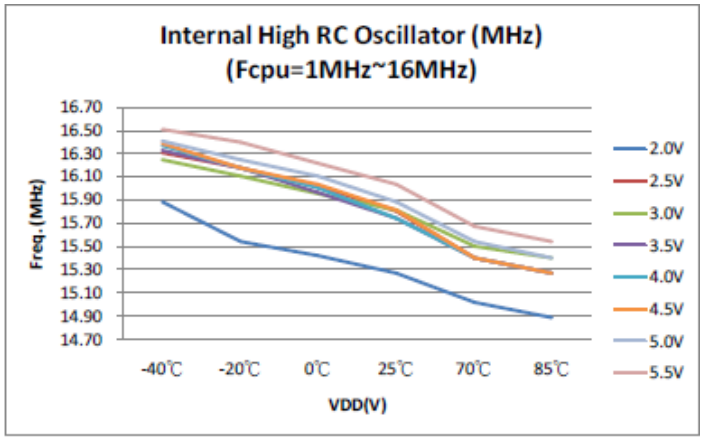
PARAMETER	SYM.	DESCRIPTION	MIN,	TYP.	MAX.	UNIT
VREFH input voltage	Vref	External reference voltage, Vdd = 5.0V.	2V	-	Vdd	V
	Virf1	Internal VDD reference voltage, Vdd = 5V.	-	Vdd*	-	V
	*Virf2	Internal 4V reference voltage, Vdd = 5V.	3.9	4*	4.1	V
	*Virf3	Internal 3V reference voltage, Vdd = 5V.	2.9	3*	3.1	V
	*Virf4	Internal 2V reference voltage, Vdd = 5V.	1.9	2*	2.1	V
Internal reference supply power	*Vprf	Internal 4/3/2V reference voltage enable.	Virf+0.5	-	-	v
AIN0 ~ AIN5 input voltage	Vani	Vdd = 5.0V	0	-	Vrefh1~5	V
ADC enable time	Tast	Ready to start convert after set ADENB = “1”	100	-	-	us
ADC current consumption	IADC	Vdd=5.0V	-	0.3	-	mA
		Vdd=3.0V	-	0.25	-	Ma
ADC Clock Frequency	FADCLK	VDD=5.0V	-		8M	Hz
		VDD=3.0V	-		5M	Hz
ADC Conversion Cycle Time	FADCYL	VDD=2.4V~5.5V	64			1/FADCLK
ADC Sampling Rate (Set FADS=1 Frequency)	FADSMP	VDD=5.0V	-	-	125	K/sec
		VDD=3.0V	-	-	80	K/sec
1/4 * VDD AIN channel input voltage	Vin	VDD=5.0V	1.187	1.25	1.313	V
Differential Nonlinearity(DNL)	DNL1	VDD=5.0V , AVREFH=2.4V, FADSMP =62.5K (12bit)	±1	-	-	LSB
Integral Nonlinearity(INL)	INL1	VDD=5.0V , AVREFH=2.4V, FADSMP =62.5K (12bit)	±2	-	-	LSB
No Missing Code	NMC	VDD=5.0V , AVREFH=2.4V, FADSMP =62.5K (12bit)	10	11	12	Bits
ADC offset Voltage	Vadcoffset	Non-trimmed	-10	0	+10	mV
		Trimmed	-2	0	+2	mV

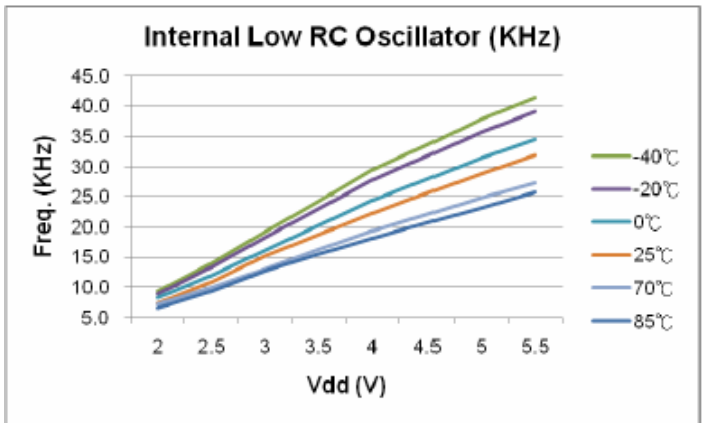
*These parameters are for design reference, not tested.

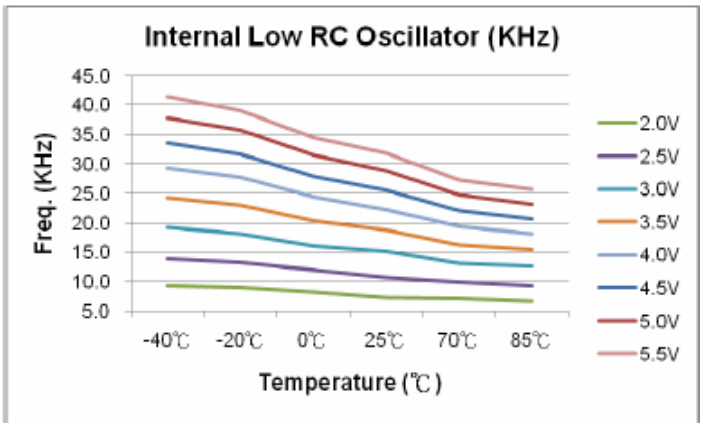
11.3 特性曲线图

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









12 开发工具

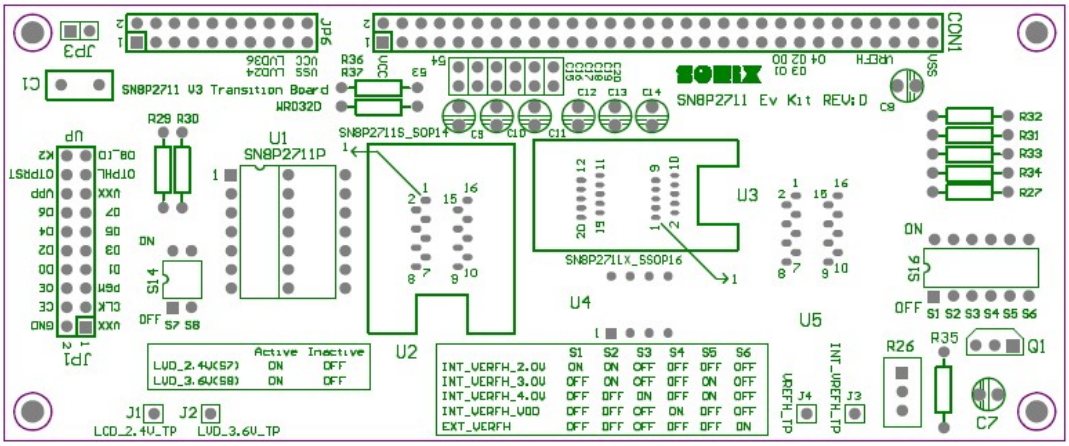
在进行 SN8P2711B 的开发时，SONIX 提供 ICE（在线仿真器），IDE（集成开发环境）和 EV-Kit 开发工具。ICE 和 EV-Kit 为外部硬件装置，IDE 有一个友好的用户界面进行韧件开发与仿真。各工具的版本如下所示：

- **ICE：SN8ICE2K Plus 2。（在仿真 IHRC 过年时，请在 ICE 选择 16MHz 的晶振。）**
- **ICE 的最高仿真速度为：8MIPS @5V（如 16MHz 晶振，Fcpu=Fosc/2）。**
- **EV-kit：SN8P2711B EV KIT REV.D。**
- **IDE：SONiX IDE M2IDE_V129 或更晚的版本。**
- **Writer：MPIII writer。**
- **Writer 转接板：SN8P2711B。**

12.1 SN8P2711B EVKIT

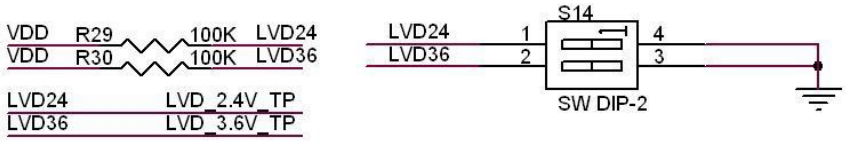
SONIX 提供 SN8P2711B EVKIT，包括 PWM 和 ADC 模拟功能。SN8ICE2K Plus II 不能给 ADC 模拟功能提供参考电源。还能通过 EVKIT 仿真 LVD 功能。

SN8P2711B EVKIT PCB 布局图如下所示：



- **CON1**：I/O 口和 ADC 参考电压输入口，连接到 SN8ICE2K Plus II CON1。
- **JP6**：LVD2.4V、3.6V 输入引脚，连接到 SN8ICE2K Plus II JP3。
- **S14**：LVD2.4V/3.6V 控制开关，反正 LVD 2.4V 标志/复位功能和 LVD 3.6V 标志功能。

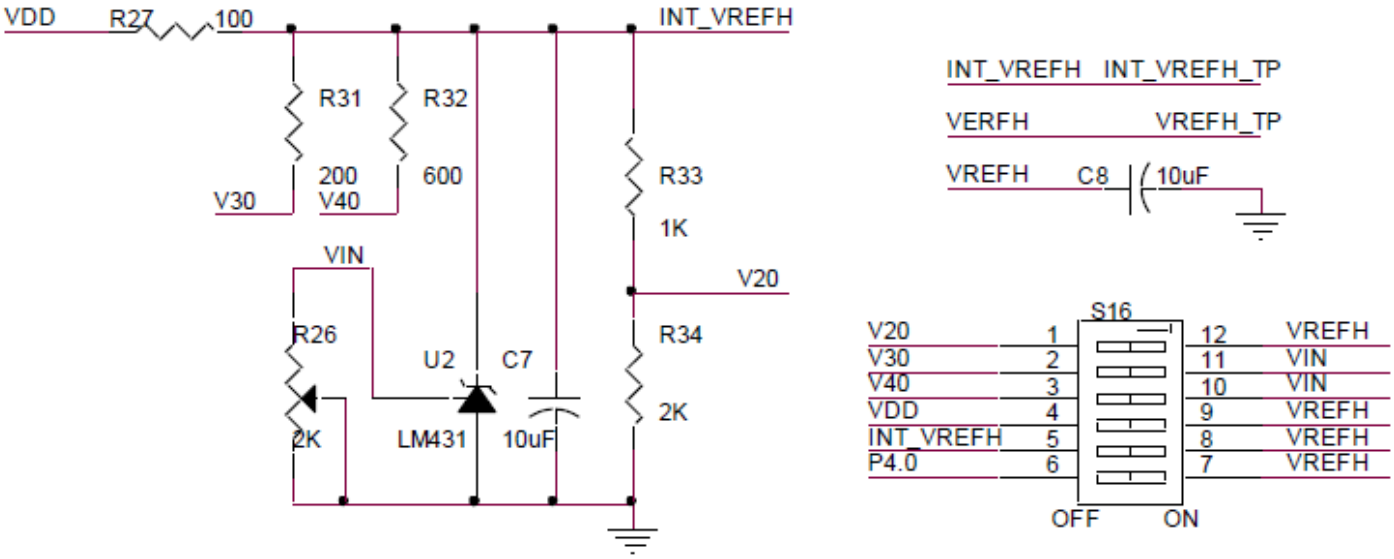
Switch No.	ON	OFF
S7	LVD 2.4V 有效	LVD 2.4V 无效
S8	LVD 3.6V 有效	LVD 3.6V 无效



- **S16**：ADC 参考电压选择项。参考电压连接到 CON1 的 VREFH 引脚。最大参考电压为 VDD，若 VDD<INT_VRERH_4.0V，ADC 参考电压则为 VDD。EXT_VREFH 是外部参考电压，由 P4.0 输入。内部参考电压模式下时，P4.0 为 GPIO 引脚或者 ADC 模拟输入引脚。

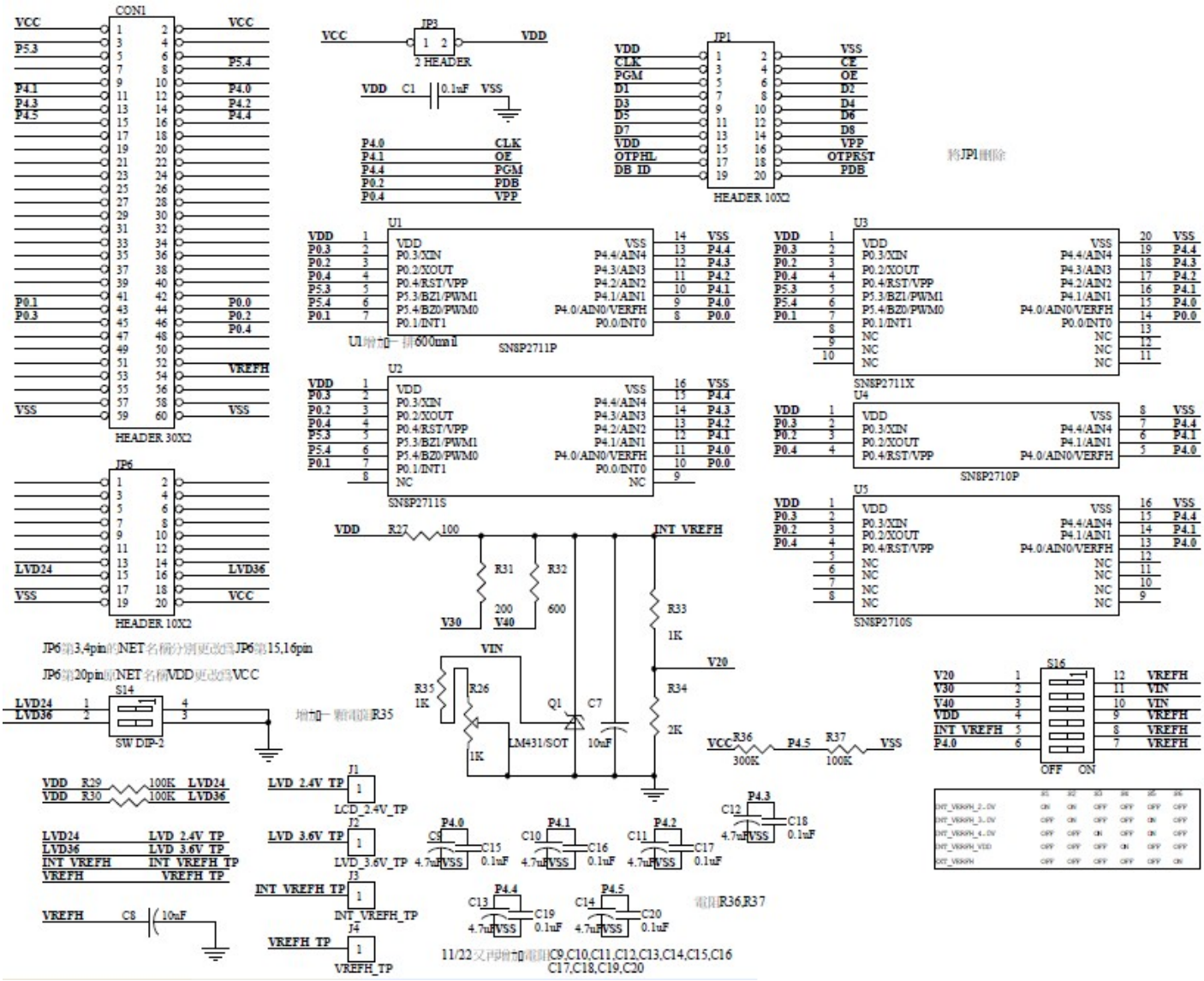
开关编号	S1	S2	S3	S4	S5	S6
INT_VREFH_2.0V	ON	ON	OFF	OFF	OFF	OFF
INT_VREFH_3.0V	OFF	ON	OFF	OFF	ON	OFF
INT_VREFH_4.0V	OFF	OFF	ON	OFF	ON	OFF
INT_VREFH_VDD	OFF	OFF	OFF	ON	OFF	OFF
EXT_VREFH	OFF	OFF	OFF	OFF	OFF	ON

- **R26**：2KΩ VR 用来条件 ADC 内部参考电压。用户必须选择正确的内部参考电压。设置 S16 为 INT_VREFH_4.0V 模式，输入电压 VDD=5V，通过 J4 测量内部参考电压，条件 R26 使得 J4 的电压为 4.0V。



- **R36, P37**：R36=300K 欧姆，R37= 100K 欧姆，偏压等于 1/4VDD。通过 ADC 通道 5 仿真低电压检测功能。
- **C9~C14**：47uF 电容，连接至 ADC 的 0～5 通道旁路电容，即 AIN0~AIN5 输入口。
- **C15~C20**：0.1uF 电容，连接至 ADC 的 0～5 通道旁路电容，即 AIN0~AIN5 输入口。

SN8P2711B EVKIT 原理图如下所示：



12.2 ICE和EVKIT应用注意事项

- 连接 SN8P2711B EVKIT 到 SN8ICE2K Plus II 之前，必须将 SN8ICE2K PlusII 的电源开关关闭。
- EVKIT 上的 JP6/CON1 连接到仿真器的 JP3/CON1。
- 必须将 SN8CE2K PlusII 上的 AVREFH/VDD 跳线帽断开。
- 前面 3 项完成后，打开 SN8ICE2K Plus II 的电源开关。
- **在 IHRC_16M 模式下仿真时，必须使用 16MHz 晶振，SN8ICE2K Plus 2 不支持超过 8M 的指令周期，但实际芯片可以。**
- 注意 ADC 内部或外部参考电压为 J4（VREFH_TP）。
- 用户需要 ADC 外部参考电压时，打开 S6（EXT_VERFH 模式），从 J4（VREFH_TP）输入参考电压。
- 用户需要内部 VDD 参考电压时，打开 S4（INT_VERFH_VDD 模式），从 J4（VREFH_TP）测量内部 VDD 参考电压。
- 用户需要内部 4V 参考电压时，打开 S3/S5（INT）VREFH_4.0V 模式），从 J4（VREFH_TP）测量内部参考电压。调节 R26 使 JP4（VREFH_TP）的电压为 4.0V。
- 用户需要内部 3V 参考电压时，打开 S2/S5（INT）VREFH_3.0V 模式），从 J4（VREFH_TP）测量内部参考电压。调节 R26 使 JP4（VREFH_TP）的电压为 3.0V。
- 用户需要内部 2V 参考电压时，打开 S1/S2（INT）VREFH_2.0V 模式），从 J4（VREFH_TP）测量内部参考电压。调节 R26 使 JP4（VREFH_TP）的电压为 2.0V。

13 OTP烧录信息

13.1 烧录转接板信息

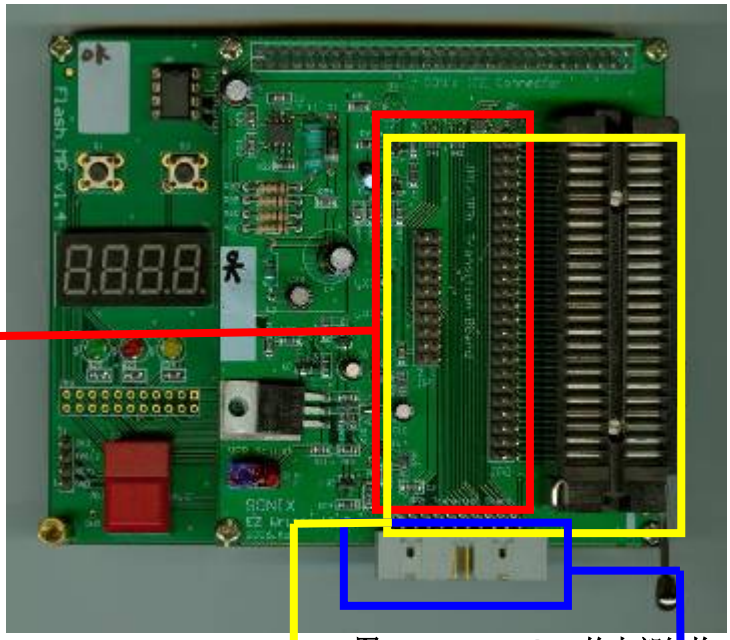
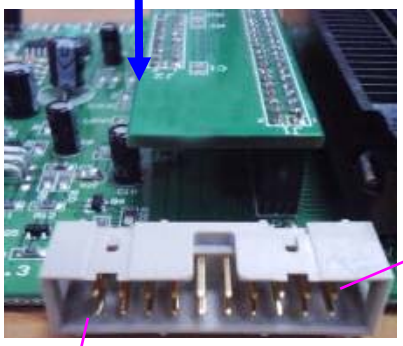


图 1　MPIII Writer 的内部结构

Writer 上板 JP1/JP3

Writer 上板 JP1/JP3



Writer 上板 JP2

注 1：JP1 连接 MP 烧录转接板，JP3 连接 OTP MCU。

注 2：JP2 连接外部烧录转接板。当 OTP MCU 的 PIN 超过 48PIN，或者烧录 Dice MCU 时，请采用外部烧录转接板，连接到 JP2 进行烧录。

下面两个图演示了如何焊接烧录转接板。

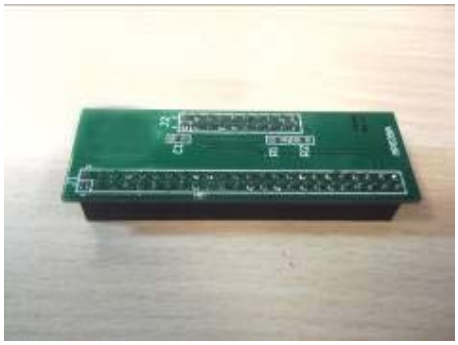


图 2

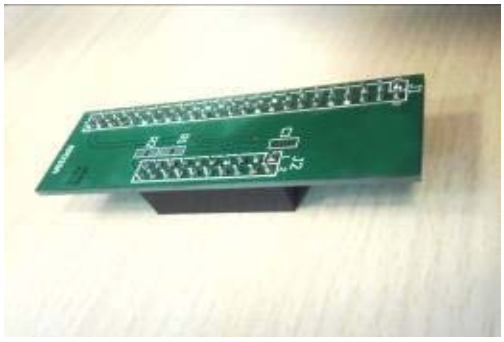


图 3

✸ **注：**

1、印有 IC 型号的这一面为转接板的正面。

2、180 度的母座必须焊接在 MP 转接板的背面。请参考图 2 和图 3。

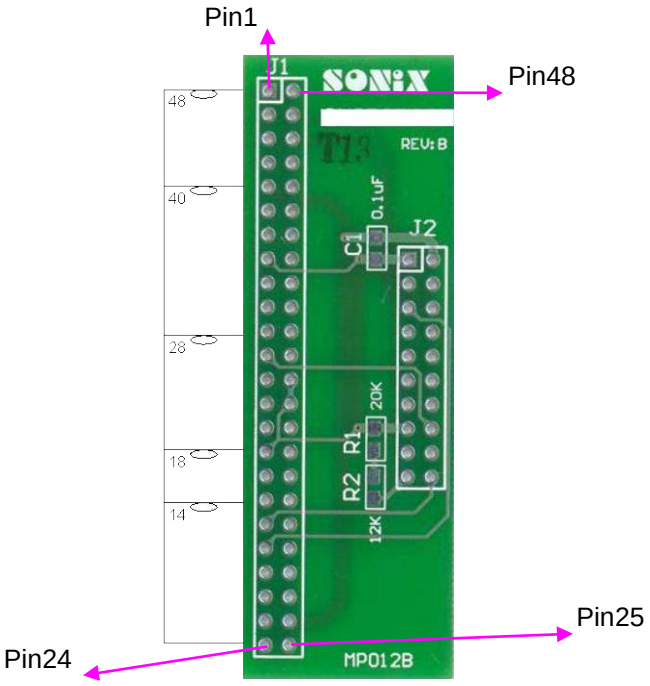


图 4　MP 转接板（连接到 JP1&JP3）

JP3（连接 48-pin text tool）

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

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 MP 烧录转接板

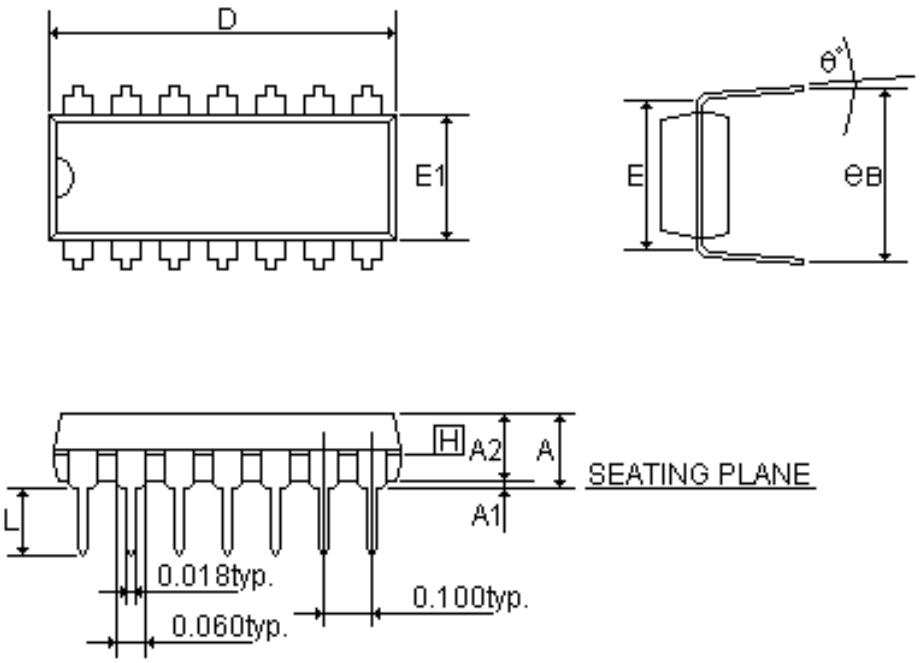
JP2 连接外部转接板

13.2 烧录引脚信息

SN8P2711B 系列烧录信息							
单片机名称		SN8P2711BP,S					
MPIII Writer		OTP IC / JP3 引脚配置					
JP1/JP2 Pin Number	JP1/JP2 Pin Name	IC Pin Number	IC Pin Number	JP3 Pin Number	IC Pin Number	IC Pin Number	JP3 Pin Number
1	VDD	1	VDD	18		VDD	
2	GND	14	VSS	31		VSS	
3	CLK	9	P4.0	26		P4.0	
4	CE	-	-	-	-	-	-
5	PGM	13	P4.4	30		P4.4	
6	OE	10	P4.1	27		P4.1	
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	1	VDD	18		VDD	
16	VPP	4	RST	21		RST	
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	3	P0.2	20		P0.2	

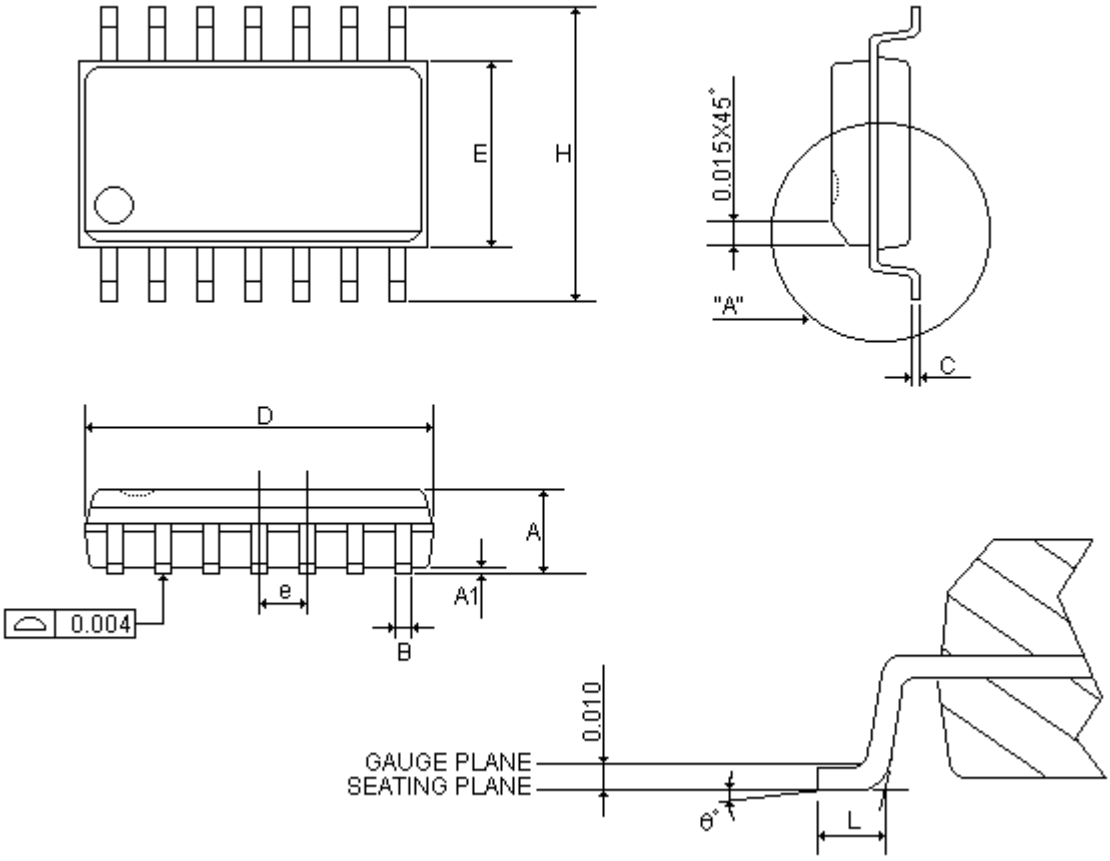
14 封裝信息

14.1 P-DIP 14 PIN



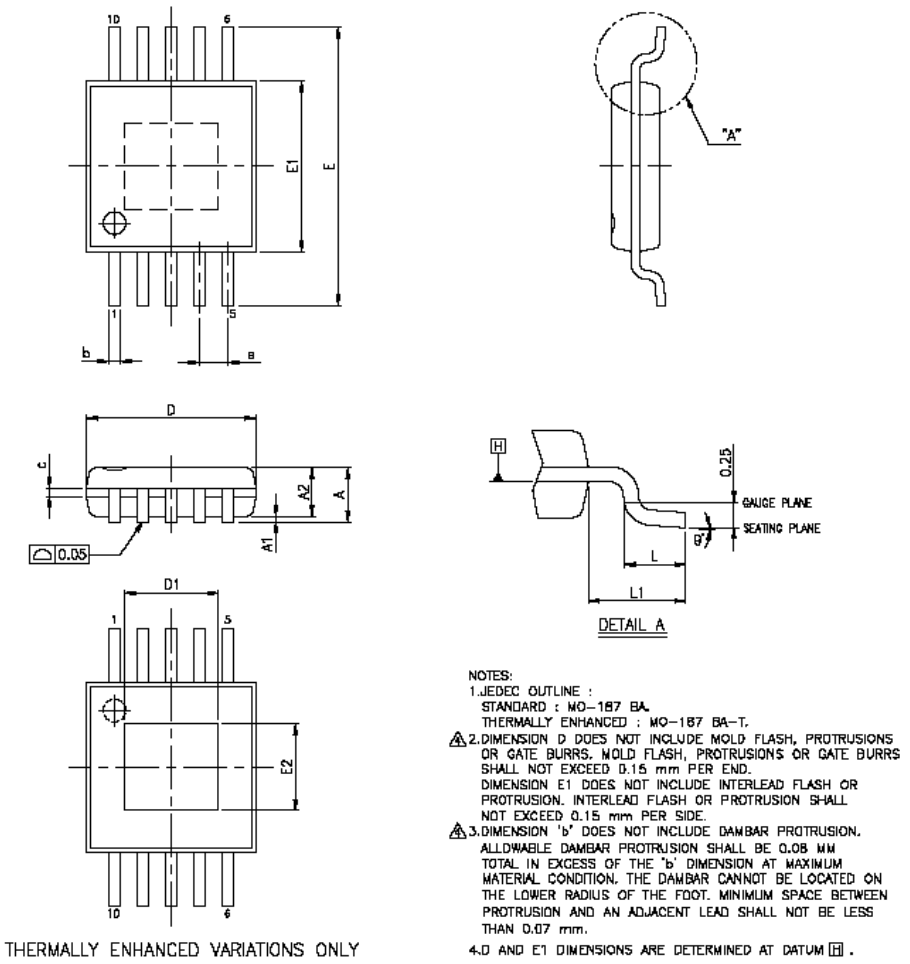
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.735	0.075	0.775	18.669	1.905	19.685
E		0.300			7.62	
E1	0.245	0.250	0.255	6.223	6.35	6.477
L	0.115	0.130	0.150	2.921	3.302	3.810
e B	0.335	0.355	0.375	8.509	9.017	9.525
θ°	0°	7°	15°	0°	7°	15°

14.2 SOP 14 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.058	0.064	0.068	1.4732	1.6256	1.7272
A1	0.004	-	0.010	0.1016	-	0.254
B	0.013	0.016	0.020	0.3302	0.4064	0.508
C	0.0075	0.008	0.0098	0.1905	0.2032	0.2490
D	0.336	0.341	0.344	8.5344	8.6614	8.7376
E	0.150	0.154	0.157	3.81	3.9116	3.9878
e	-	0.050	-	-	1.27	-
H	0.228	0.236	0.244	5.7912	5.9944	6.1976
L	0.015	0.025	0.050	0.381	0.635	1.27
θ°	0°	-	8°	0°	-	8°

14.3 MSOP 10 PIN



NOTES:
1. JEDEC OUTLINE :
   STANDARD : MO-187 BA.
   THERMALLY ENHANCED : MO-187 BA-T.
2. DIMENSION D DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS. MOLD FLASH, PROTRUSIONS OR GATE BURRS SHALL NOT EXCEED 0.15 mm PER END.
   DIMENSION E1 DOES NOT INCLUDE INTERLEAD FLASH OR PROTRUSION. INTERLEAD FLASH OR PROTRUSION SHALL NOT EXCEED 0.15 mm PER SIDE.
3. DIMENSION 'b' DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 MM TOTAL IN EXCESS OF THE 'b' DIMENSION AT MAXIMUM MATERIAL CONDITION. THE DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OF THE FOOT. MINIMUM SPACE BETWEEN PROTRUSION AND AN ADJACENT LEAD SHALL NOT BE LESS THAN 0.07 mm.
4. D AND E1 DIMENSIONS ARE DETERMINED AT DATUM H.

SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.043	-	-	1.10
A1	0.000	-	0.006	0.00	-	0.15
A2	0.030	0.033	0.037	0.75	0.85	0.95
b	0.007	-	0.011	0.17	-	0.27
c	0.003	-		0.08	-	0.23
D	0.118 BSC			3.00 BSC		
E	0.193 BSC			4.90 BSC		
E1	0.118 BSC			3.00 BSC		
e	0.197 BSC			0.50 BSC		
L	0.016	0.024	0.031	0.40	0.60	0.80
L1	0.374 REF			0.95 REF		
θ	0	-	8	0	-	8

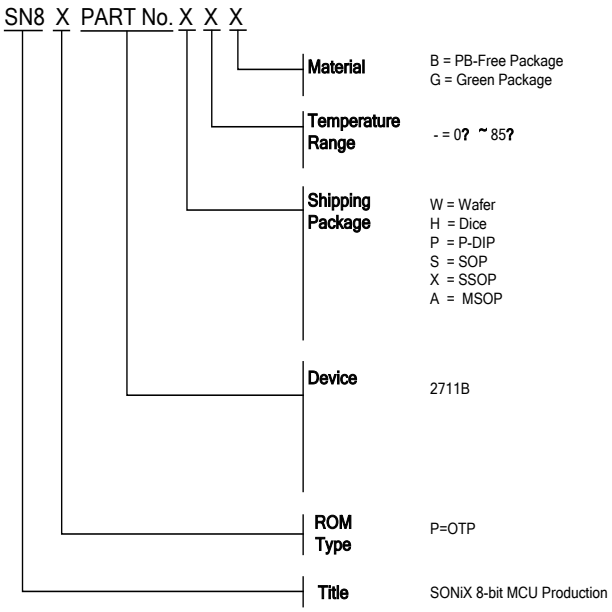
	E2 (mm)		D1 (mm)	
PAD SIZE	MIN	MAX	MIN	MAX
75x70E	1.52	1.91	1.42	1.78

15 芯片正印命名规则

15.1 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。

15.2 芯片型号说明



15.3 命名举例

- **Wafer，Dice：**

名称	ROM 类型	器件（Device）	封装形式	温度范围	材料
SN8P2711BW	OTP	2711B	Wafer	0℃~70℃	-
SN8P2711BH	OTP	2711B	Dice	0℃~70℃	-

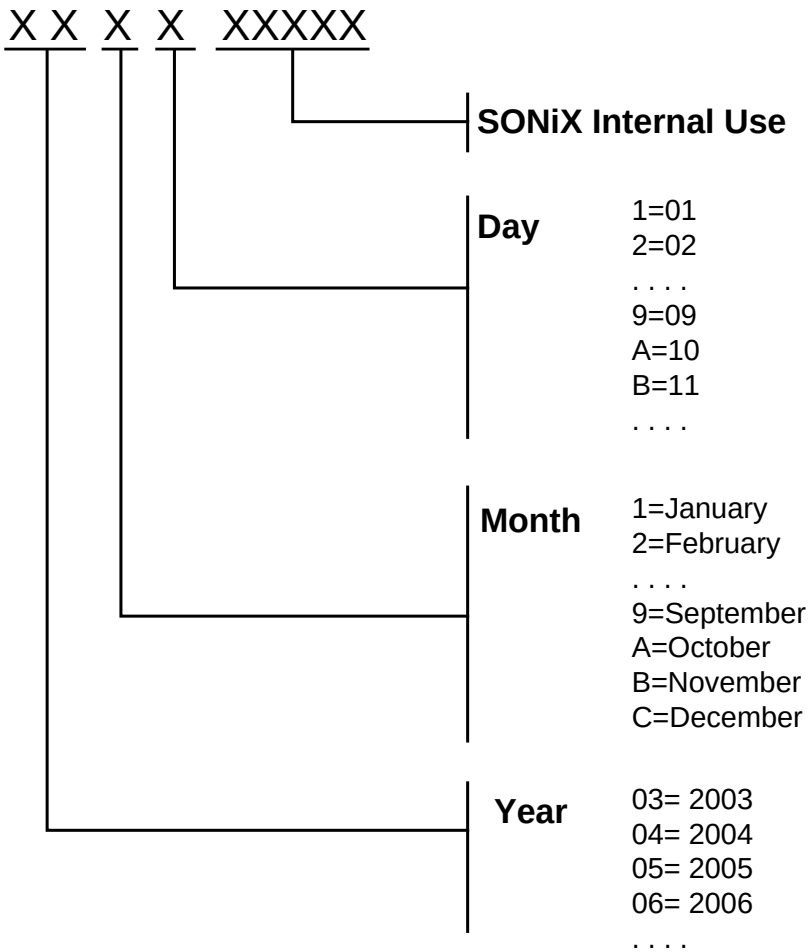
- **绿色封装：**

名称	ROM 类型	器件（Device）	封装形式	温度范围	材料
SN8P2711BPG	OTP	2711B	P-DIP	0℃~70℃	绿色封装
SN8P2711BSG	OTP	2711B	SOP	0℃~70℃	绿色封装
SN8P27113BAG	OTP	2711B	MSOP	0℃~70℃	绿色封装

- **无铅封装：**

名称	ROM 类型	器件（Device）	封装形式	温度范围	材料
SN8P2711BPB	OTP	2711B	P-DIP	0℃~70℃	无铅封装（PB-Free Package）
SN8P2711BSB	OTP	2711B	SOP	0℃~70℃	无铅封装（PB-Free Package）
SN8P27113BAB	OTP	2711B	MSOP	0℃~70℃	无铅封装（PB-Free Package）

15.4 日期码规则



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：香港新界沙田火炭禾盛街 11 号，中建电讯大厦 26 楼 03 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