SN8P2602C

用户参考手册

Version 1.4

SN8P2602C

SONiX 8 位单片机

修改记录

版本	时间	修改记录
VER 1.0	2010/07	第一版。
VER 1.1	2011/06	1、增加 Code Option Low_Power 说明。 2、增加开发工具的说明。 3、增加电气特性曲线图说明：-40℃~+85℃曲线图。 4、调整开发工具章节的部分内容。
VER 1.4	2016/06	修改电气特性的部分内容。

目　　录

1 产品简介

1.1 功能特性

- **存储器配置**
  ROM：1K * 16 位。
  RAM：48 * 8 位。

- **4 层堆栈缓存器**

- **3 个中断源**
  2 个内部中断：T0，TC0。
  1 个外部中断：INT0。

- **I/O 引脚配置**
  双向输入输出端口：P0，P1，P5。
  单向输入引脚：P1.5。
  具有上拉电阻的端口：P0，P1，P5。
  具有下拉电阻的引脚：P5.0~P5.3。
  具有唤醒功能的端口：P0、P1 电平变换。
  40mA sink 引脚：P5.0~P5.3，P5.5。
  可编程的开漏引脚：P1.0。
  外部中断触发沿：
  P0.0，由 PEDGE 寄存器控制。

- **3 级 LVD**
  复位系统，监控系统电源。

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

- **Fcpu（指令周期）**
  Fcpu = Fosc/1，Fosc/2，Fosc/4，Fosc/8，Fosc/16，Fosc/32，Fosc/64，Fosc/128。

- **1 个 8 位基本定时器 T0，具有 RTC 功能（0.5S）**
- **1 个 8 位定时器 TC0，具有外部事件计数器功能 Buzzer 和 PWM 功能**
- **1 通道 2K/4K buzzer 输出。**
- **内置看门狗定时器，时钟源由你低速 RC 时钟提供 (16KHz @3V, 32KHz @5V)**

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

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

- **封装形式**
  DIP 18 pin
  SOP 18 pin
  SSOP 20 pin

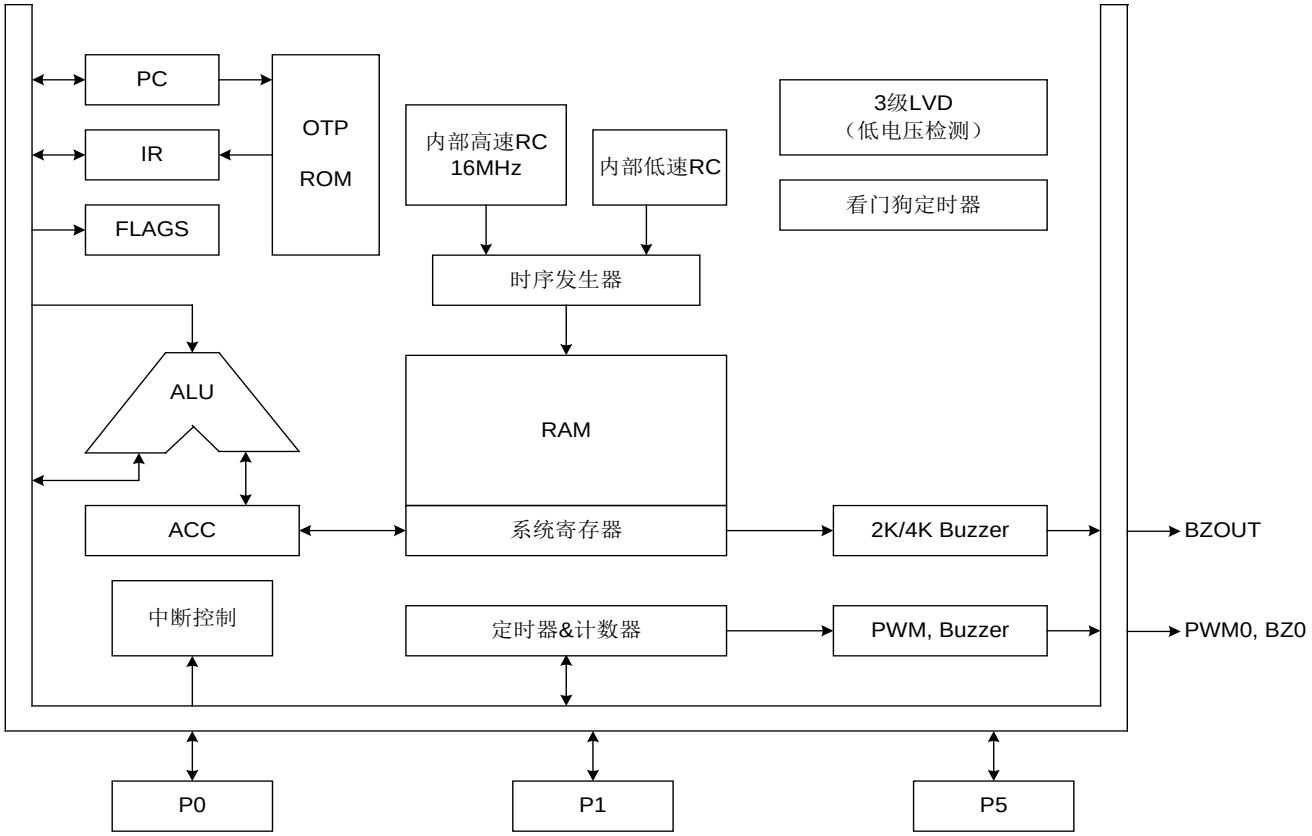
☞ **产品性能列表**

单片机名称	ROM (word)	RAM (Byte)	堆栈	定时器		LVD 级数	IHRC	I/O	绿色模式	低速模式	PWM/ Buzzer	2K/4K Buzzer	唤醒功能引脚数目	封装形式
				T0	TC0									
SN8P2602B	1K	48	4	V	V	3	-	15	V	V	V	-	7	DIP18/SOP18/SSOP20
SN8P2602C	1K	48	4	V	V	3	V	16	V	V	V	V	8	DIP18/SOP18/SSOP20
SN8P2612	2K	64	4	V	V	3	V	16	V	V	V	-	8	DIP18/SOP18/SSOP20
SN8P2613	2K	64	4	V	V	3	V	18	V	V	V	-	10	DIP20/SOP20/SSOP20

☞ **产品资源配置改良表（从 SN8P2602B 到 SN8P2602C）**

项目	SN8P2602B	SN8P2602C
XIN/XOUT 引脚	XIN 引脚不支持 GPIO 功能，XOUT 引脚支持 GPIO 功能	与 P1.4/P1.6 GPIO 引脚共用
下拉电阻	无	P5.0~P5.3 内置下拉电阻
Sink 电流	15mA	P5.0~P5.3，P5.5 内置 15mA/40mA sink 电流，由 CODE OPTION 选择
IHRC 16MHz	无	内置 IHRC 16MHz
Fcpu	Fosc/1~Fosc/8	Fosc/1~Fosc/128
T0 定时器	无 RTC 功能	具有 RTC 功能
2K/4K buzzer 功能	无	支持 2K/4K buzzer 功能
低功耗模式	无低功耗选项	内置低功耗选项以省电

1.2 系统框图



1.3 引脚配置

SN8P2602CP (PDIP 18 pins)
SN8P2602CS (SOP 18 pins)

P1.2	1	U	18	P1.1
P1.3	2		17	P1.0
P0.0/INT0	3		16	XIN/P1.6
P1.5/RST/VPP	4		15	XOUT/P1.4
VSS	5		14	VDD
P5.0	6		13	P5.7
P5.1	7		12	P5.6
P5.2	8		11	P5.5/BZOUT
P5.3	9		10	P5.4/BZ0/PWM0

SN8P2602CX (SSOP 20 pins)

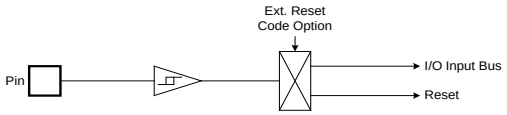
P1.2	1	U	20	P1.1
P1.3	2		19	P1.0
P0.0/INT0	3		18	XIN/P1.6
P1.5/RST/VPP	4		17	XOUT/P1.4
VSS	5		16	VDD
VSS	6		15	VDD
P5.0	7		14	P5.7
P5.1	8		13	P5.6
P5.2	9		12	P5.5/BZOUT
P5.3	10		11	P5.4/BZ0/PWM0

1.4 引脚说明

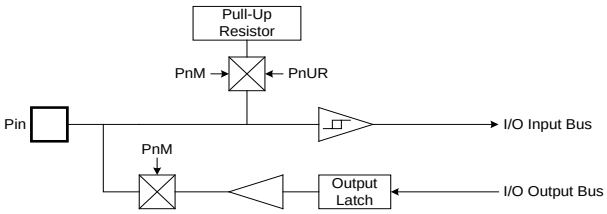
引脚名称	类型	功能说明
VDD, VSS	P	电源输入端。
P1.5/RST/VPP	I, P	RST：系统外部复位输入引脚，施密特触发，低电平有效，通常保持高电平。
		VPP：OTP 12.3V 烧录电源输入引脚。
		P1.5：单向输入引脚，施密特触发，无上拉电阻，电平变换时唤醒系统。
P0.0/INT0	I/O	P0.0：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时唤醒系统。
		INT0：外部中断输入引脚。
P1.0	I/O	双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时唤醒系统，可编程开漏引脚。
P1[3:1]	I/O	双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时唤醒系统。
XIN/P1.6	I/O	XIN：振荡器输入引脚。
		P1.6：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时唤醒系统。
XOUT/P1.4	I/O	XOUT：振荡器输出引脚。
		P1.4：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，电平变换时唤醒系统。
P5[3:0]	I/O	双向输入输出引脚，输入模式时施密特触发，内置上拉电阻和下拉电阻，可编程 15mA/40mA sink 电流。
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。
P5.5/BZOUT	I/O	P5.5：双向输入输出引脚，输入模式时施密特触发，内置上拉电阻，可编程的 15mA/40mA sink 电流。
		BZOUT：2K/4K buzzer 输出引脚。
P5[7:6]	I/O	双向输入输出引脚，输入模式时施密特触发，内置上拉电阻。

1.5 引脚电路结构图

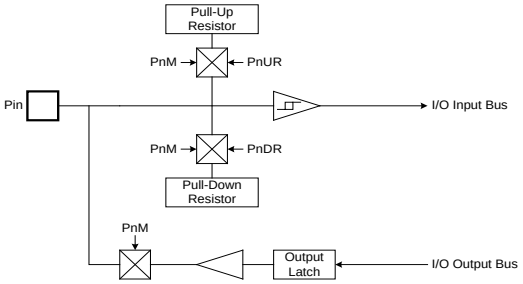
- **P1.5 复位引脚：**



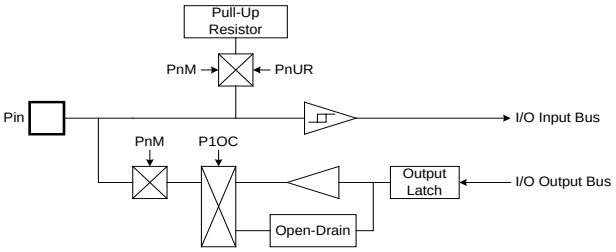
- **P0，P1，P5.4~P5.7 GPIO 引脚：**



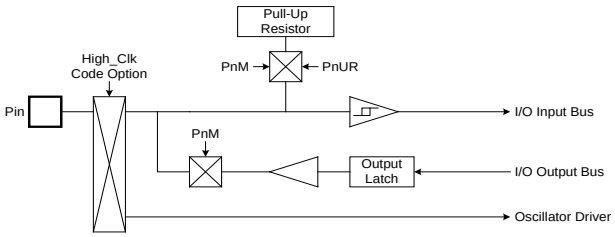
- **P5.0~P5.3 GPIO 引脚：**



- **P1.0：**



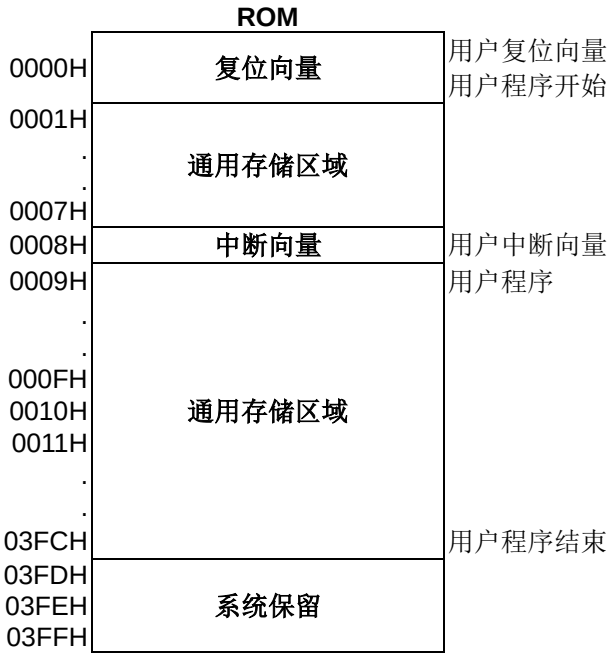
- **P1.4，P1.6：**



2 中央处理器（CPU）

2.1 程序存储器（ROM）

☞ **ROM：1K**



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

2.1.1 复位向量（0000H）

具有一个字长的系统复位向量（0000H）。

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

发生上述任一种复位后，程序将从 0000H 处重新开始执行，系统寄存器也都将恢复为默认值。根据 PFLAG 寄存器中的 NT0 和 NPD 标志位的内容可以判断系统复位方式。下面一段程序演示了如何定义 ROM 中的复位向量。

➢ **例：定义复位向量。**

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

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

2.1.2 中断向量（0008H）

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

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

➢ **例：定义中断向量，中断服务程序紧随 ORG 8 之后。**

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

                ENDP                    ; 程序结束。
```

➢ **例：定义中断向量，中断服务程序在用户程序之后。**

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

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

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

2.1.3 查表

在 SONiX 单片机中，对 ROM 区中的数据进行查找，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   PCL, A
        ENDM
```

✷ 注：“VAL”为跳转表列表中列表个数。

➢ **例：宏“MACRO3.H”中，“@JMP_A”的应用。**

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

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

➢ **例：“@JMP_A”运用举例**

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

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

2.1.5 CHECKSUM计算

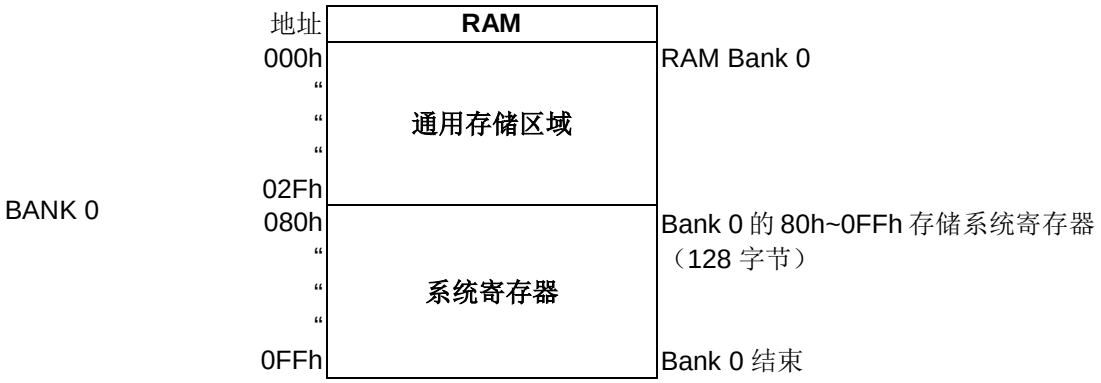
ROM 的末端位置的几个字限制使用，进行 Checksum 计算时，用户应避免对该单元的访问。

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

```
                MOV         A,#END_USER_CODE$L
                B0MOV       END_ADDR1, A            ; 用户程序终端地址低位字节存入 end_addr1。
                MOV         A,#END_USER_CODE$M
                B0MOV       END_ADDR2, A            ; 用户程序终端地址中间字节存入 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                      ; 如果 Z != 00H，进行下一个计算。
                JMP         Y_ADD_1                 ; 如果 Z = 00H，Y 加 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                      ; 转至 checksum 计算。
CHECKSUM_END:
                …
                …
END_USER_CODE:
```

2.2 数据存储器（RAM）

☞ **RAM：48*8bit**



48 字节的通用存储区域位于 RAM Bank 0 中，Sonix 提供“Bank 0”型的指令（如 B0MOV、B0ADD、B0BTS1、B0BTS0 等）直接访问 Bank 0 RAM。

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	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	-	-	-	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	P1	-	-	-	P5	-	-	T0M	T0C	TC0M	TC0C	BZM	-	-	STKP
E	P0UR	P1UR	-	-	-	P5UR	P5DR	@YZ	-	P1OC	-	-	-	-	-	-
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 = 特殊标志寄存器
INTRQ = 中断请求寄存器
WDTR = 看门狗定时器清零寄存器
PnM = Pn 模式控制寄存器
PnUR = Pn 上拉电阻控制寄存器
PCH, PCL = 程序计数器
T0C = T0 计数寄存器
TC0C = TC0 计数寄存器
P1OC = P1 漏极开路控制寄存器
STKP = 堆栈指针

Y, Z = 专用寄存器，@YZ 间接寻址寄存器，ROM 寻址寄存器
PEDGE = P0.0 触发方向寄存器
INTEN = 中断使能寄存器
Pn = Pn 数据缓存器
OSCM = 振荡器模式寄存器
PnDR= Pn 下拉电阻控制寄存器
T0M = T0 模式寄存器
TC0M = TC0 模式寄存器
TC0R = TC0 自动重装寄存器
BZM= 2K/4K buzzer 控制寄存器
@YZ = 间接寻址寄存器
STK0~STK3 = 堆栈缓存器

2.2.1.3 系统寄存器位定义

地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	备注
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
0B8H								P00M	R/W	P0M
0BFH				P00G1	P00G0				R/W	PEDGE
0C0H		P16W	P15W	P14W	P13W	P12W	P11W	P10W	W	P1W
0C1H		P16M		P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C5H	P57M	P56M	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C8H			TC0IRQ	T0IRQ				P00IRQ	R/W	INTRQ
0C9H			TC0IEN	T0IEN				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								P00	R/W	P0
0D1H		P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D5H	P57	P56	P55	P54	P53	P52	P51	P50	R/W	P5
0D8H	T0ENB	T0rate2	T0rate1	T0rate0				T0TB	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	BZEN	BZrate1	BZrate0						R/W	BZM
0DFH	GIE						STKPB1	STKPB0	R/W	STKP
0E0H								P00R	W	P0UR
0E1H		P16R		P14R	P13R	P12R	P11R	P10R	W	P1UR
0E5H	P57R	P56R	P55R	P54R	P53R	P52R	P51R	P50R	W	P5UR
0E6H					P53DR	P52DR	P51DR	P50DR	W	P5DR
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E9H								P10OC	W	P1OC
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、所有寄存器名都已在 SN8ASM 编译器中做了宣告；
3、用户使用 SN8ASM 编译器对寄存器的位进行操作时，须在该寄存器的位前加“F”；
4、指令“b0bset”，“b0bclr”，“bset”，“bclr”只能用于可读写的寄存器（“R/W”）。

2.2.2 累加器ACC

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

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

➢ **例：读/写 ACC。**

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

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

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

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

➢ **例：保存 ACC 和工作寄存器。**

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

        RETI            ; 退出中断。
```

2.2.3 程序状态寄存器PFLAG

寄存器 PFLAG 中包含 ALU 运算状态信息、系统复位状态信息和 LVD 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	-	-	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 = 无效（VDD>3.6V）；
1 = 有效（VDD≦3.6V）。

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

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

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

Bit 0 **Z**：零标志。
1 = 算术/逻辑/分支运算的结果为零；
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	-	-	-	-	-	-	-	-

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

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

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

➢ **例：利用数据指针@YZ 对 RAM 数据清零。**

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

CLR_YZ_BUF:
        CLR     @YZ         ; @YZ 清零。

        DECMS   Z           ;
        JMP     CLR_YZ_BUF  ; 不为零。

        CLR     @YZ
END_CLR:                    ;
        …
```

2.2.6 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	-	-	-	-	-	-	-	-

✷ **注：关于 R 寄存器查表应用请参考查表章节。**

2.3 寻址模式

2.3.1 立即寻址模式

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 直接寻址模式

通过 ACC 对 RAM 单元数据进行操作。

- **例：地址 12H 处的内容送入 ACC。**

```
B0MOV   A, 12H
```

- **例：ACC 中数据写入 RAM 的 12H 单元。**

```
B0MOV   12H, A
```

2.3.3 间接寻址模式

通过指针寄存器（Y/Z）访问 RAM 数据。

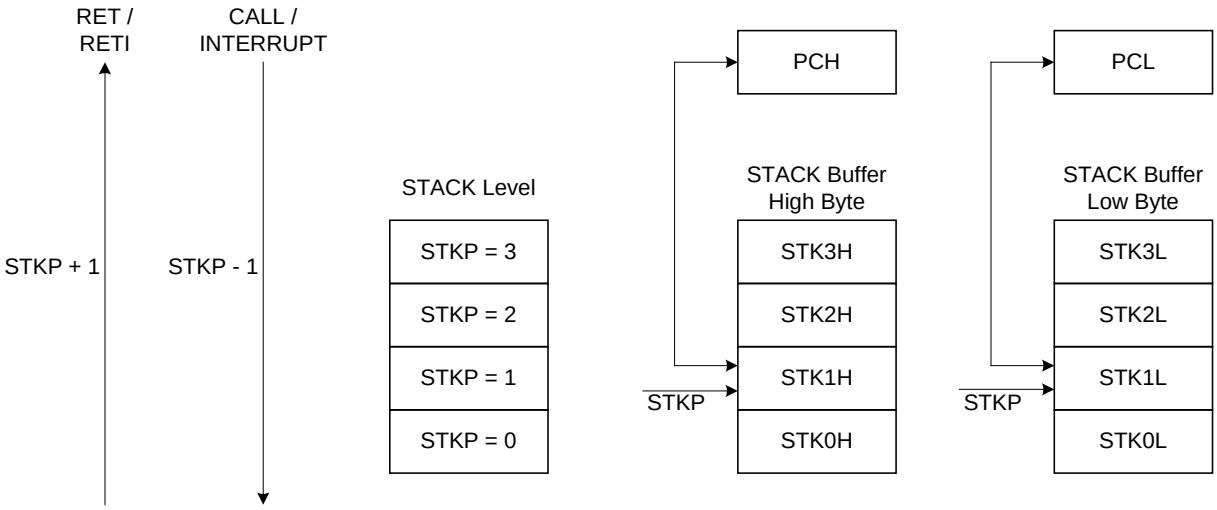
- **例：用 @YZ 实现间接寻址。**

```
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

SN8P2602C 的堆栈缓存器共有 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，指向堆栈缓存器顶层，STKnH 和 STKnL 分别是各堆栈缓存器高、低字节。



2.4.2 堆栈寄存器

堆栈指针 STKP 是一个 2 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。

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	-	-	-	-	-	STKPB1	STKPB0
读/写	R/W	-	-	-	-	-	R/W	R/W
复位后	0	-	-	-	-	-	1	1

Bit[1:0] **STKPBn**：堆栈指针（n = 0 ~ 1）。

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, #00000011B
B0MOV   STKP, A
```

0F0H~0F8H	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~0F8H	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 的内容进行入栈保存。

堆栈层数	STKP		堆栈缓存器		备注
	STKPB1	STKPB0	高字节	低字节	
0	1	1	Free	Free	-
1	1	0	STK0H	STK0L	-
2	0	1	STK1H	STK1L	-
3	0	0	STK2H	STK2L	-
4	1	1	STK3H	STK3L	-
> 4	1	0	-	-	溢出，出错

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

堆栈层数	STKP		堆栈缓存器		备注
	STKPB1	STKPB0	高字节	低字节	
4	1	1	STK3H	STK3L	-
3	0	0	STK2H	STK2L	-
2	0	1	STK1H	STK1L	-
1	1	0	STK0H	STK0L	-
0	1	1	Free	Free	-

2.5 编译选项列表（CODE OPTION）

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

编译选项	配置项目	功能说明
Noise_Filter	Enable	使能杂讯滤波器，Fcpu 选择 Fosc/4~Fosc/128。
	Disable	禁止杂讯滤波器，Fcpu 选择 Fosc/1~Fosc/128。
Fcpu	Fhosc/1	指令周期为 1 个振荡时钟，必须禁止杂讯滤波器。
	Fhosc/2	指令周期为 2 个振荡时钟，必须禁止杂讯滤波器。
	Fhosc/4	指令周期为 4 个振荡时钟。
	Fhosc/8	指令周期为 8 个振荡时钟。
	Fhosc/16	指令周期为 16 个振荡时钟。
	Fhosc/32	指令周期为 32 个振荡时钟。
	Fhosc/64	指令周期为 64 个振荡时钟。
	Fhosc/128	指令周期为 128 个振荡时钟。
High_Clk	IHRC_16M	高速内部 16MHz RC，XIN/XOUT 引脚为 GPIO 引脚。
	IHRC_RTC	高速内部 16MHz RC，XIN/XOUT 引脚外接 32768Hz 晶振。
	RC	外部高速振荡器采用廉价的 RC 振荡电路，XIN 引脚外接 RC 电路，XOUT 引脚为 GPIO 引脚。
	32K X'tal	外部高速振荡器采用低频、低功耗的晶体振荡器（如 32.768KHz）。
	12M X'tal	外部高速振荡器采用高速陶瓷/石英振荡器（如 12MHz）。
	4M X'tal	外部高速振荡器采用标准陶瓷/石英振荡器（如 4MHz）。
Watch_Dog	Always_On	始终开启看门狗定时器，睡眠模式和绿色模式下也不例外。.
	Enable	开启看门狗定时器，但在睡眠模式和绿色模式会处于关闭状态。
	Disable	关闭看门狗定时器。
Reset_Pin	Reset	使能外部复位引脚。
	P15	使能 P1.5 的单向输入功能，无上拉电阻。
Security	Enable	ROM 代码加密。
	Disable	ROM 代码不加密。
Low_Power	Enable	使能低功耗功能以省电。
	Disable	禁止低功耗功能。
LVD	LVD_L	如果 VDD 低于 2.0V，LVD 复位系统。
	LVD_M	如果 VDD 低于 2.0V，LVD 复位系统。 寄存器 PFLAG 的 LVD24 位作为 2.4V 低电压的监测器。
	LVD_H	如果 VDD 低于 2.4V，LVD 复位系统。 寄存器 PFLAG 的 LVD36 位作为 3.6V 低电压的监测器。
	LVD_MAX	如果 VDD 低于 3.6V，LVD 复位系统。
P5_SINK	40mA	P5.0~P5.3，P5.5 sink 电流为 40mA。
	15mA	P5.0~P5.3，P5.5 sink 电流为 15mA。

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。
- **P15**：使能 P1.5 为单向输入引脚。此时禁止外部复位引脚功能。

2.5.3 Security编译选项

Security 编译选项是对 OTP ROM 的一种保护，当使能 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。如果使能 Noise Filter，Fcpu 被限制为 Fhosc/2 以下，即 Fcpu 最快只能是 Fhosc/4；若禁止 Noise Filter，Fcpu 可以设置为 Fhosc/1 和 Fhosc/2。在高干扰环境下，使能杂讯滤波器和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。

2.5.5 Low_Power编译选项

Low_Power 编译选项可以减少工作电流，系统时钟 rate 小于或等于 2mips。

- **Fcpu，Noise_Filter & Low_Power 特性选项表：**

Code Option					
Low_Power	**High_Clk Clocl**	**频率（Hz）**	**Noise_Filter**	**Fcpu（Limit）**	**备注**
Enable	IHRC_16M & IHRC_RTC	16M	-	Fhosc/8~Fhosc/128	Fcpu 应该小于或等于 2MIPS
	外部晶振或者 RC	Fhosc≤8M	Enalbe	Fhosc/4~Fhosc/128	
		8M<Fhosc≤16M		Fhosc/8~Fhosc/128	
	外部晶振或者 RC	Fhosc≤2M	Disable	Fhosc/1~Fhosc/128	
		2M<Fhosc≤4M		Fhosc/2~Fhosc/128	
		4M<Fhosc≤8M		Fhosc/4~Fhosc/128	
		8M<Fhosc≤16M		Fhosc/8~Fhosc/128	
Disable	IHRC_16M & IHRC_RTC	-	Enable	Fhosc/4~Fhosc/128	
	外部晶振或者 RC				
	IHRC_16M & IHRC_RTC	-	Disable	Fhosc/1~Fhosc/128	
	外部晶振或者 RC				

3 复位

3.1 概述

SN8P2602C 系列的单片机有以下几种复位方式：

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

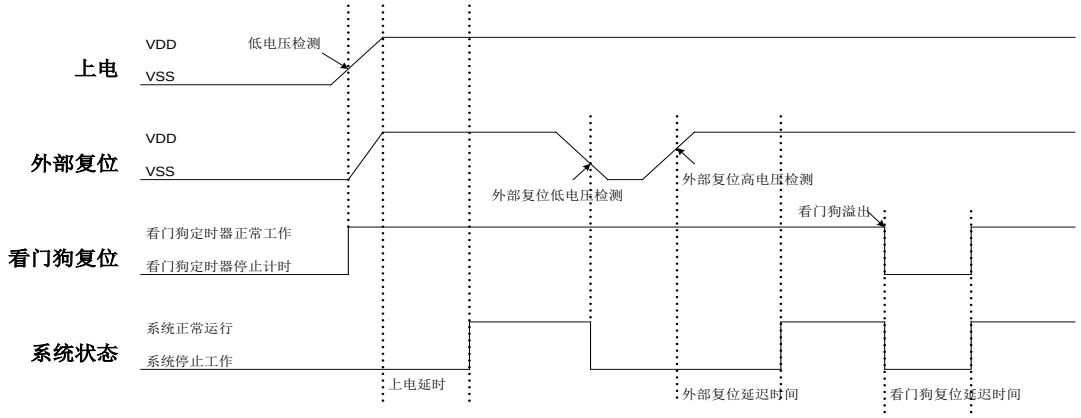
上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器 PC 清零。复位结束后，系统从向量 0000H 处重新开始运行。PFLAG 寄存器的 NT0 和 NPD 两个标志位能够给出系统复位状态的信息。用户可以根据 NT0 和 NPD 的状态，编程控制系统的运行路径。

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

Bit [7:6] NT0, NPD：复位状态标志。

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

任何一种复位方式都需要一定的响应时间，系统提供完善的复位流程以保证复位动作的顺利进行。对于不同类型的振荡器，完成复位所需要的时间也不同。因此，VDD 的上升速度和不同晶振的起振时间都不固定。RC 振荡器的起振时间最短，晶体振荡器的起振时间则较长。在用户使用的过程中，应考虑系统对上电复位时间的要求。系统复位时序图如下：



3.2 上电复位

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

- **上电：**系统检测到电源电压上升并等待其稳定；
- **外部复位（使能外部复位引脚时才有效）：**系统检测外部复位引脚状态。如果不为高电平，系统保持复位状态直到外部复位引脚的复位结束。
- **系统初始化：**所有的系统寄存器被置为默认状态；
- **振荡器开始工作：**振荡器开始提供系统时钟；
- **执行程序：**上电结束，程序开始运行。

3.3 看门狗复位

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

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

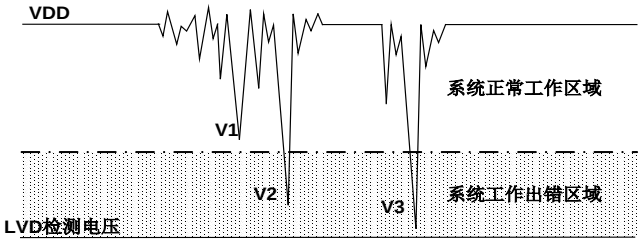
看门狗定时器应用注意事项如下：

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

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

3.4 掉电复位

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



掉电复位示意图

电压跌落可能会进入系统死区。系统死区意味着电源不能满足系统的最小工作电压要求。上图是一个典型的掉电复位示意图。图中，VDD 受到严重的干扰，电压值降的非常低。虚线以上区域系统正常工作，在虚线以下的区域内，系统进入未知的工作状态，这个区域称作死区。当 VDD 跌至 V1 时，系统仍处于正常状态；当 VDD 跌至 V2 和 V3 时，系统进入死区，则容易导致出错。以下情况系统可能进入死区：

DC 运用中：

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

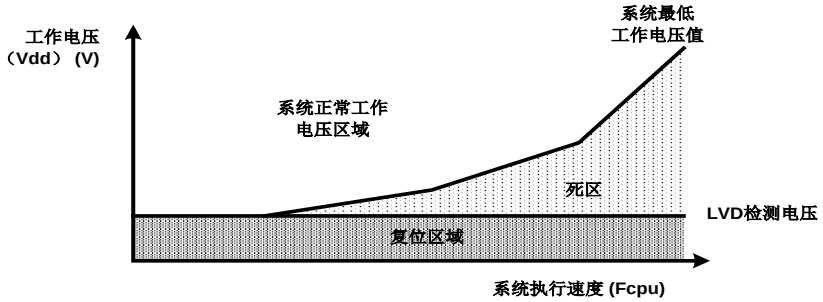
AC 运用中：

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

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

3.4.1 系统工作电压

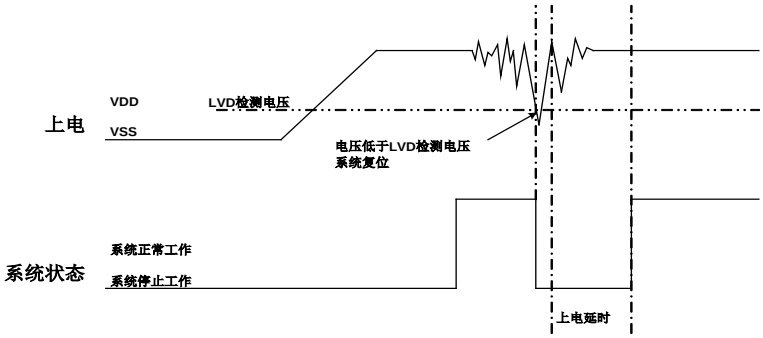
为了改善系统掉电复位的性能，首先必须明确系统具有的最低工作电压值。系统最低工作电压与系统执行速度有关，不同的执行速度下最低工作电压值也不同。



系统工作电压与执行速度关系图

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

3.4.2 低电压检测（LVD）



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

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

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

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

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

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

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

✷ 注：
a) LVD 复位结束后，LVD24 和 LVD36 都将被清零；
b) LVD 2.4V 和 LVD 3.6V 检测电平值仅作为设计参考，不能用作芯片工作电压值的精确检测。

3.4.3 掉电复位性能改进

如何改善系统掉电复位性能，有以下几点建议：

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

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

看门狗复位：

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

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

降低系统工作速度：

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

附加外部复位电路：

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

3.5 外部复位

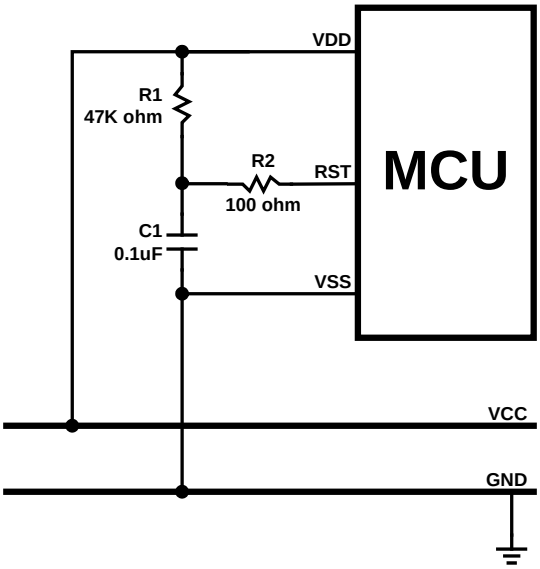
外部复位功能由编译选项“Reset_Pin”控制。将该编译选项置为“Reset”，可使能外部复位功能。外部复位引脚为施密特触发结构，低电平有效。复位引脚处于高电平时，系统正常运行。当复位引脚输入低电平信号时，系统复位。外部复位操作在上电和正常工作模式时有效。需要注意的是，在系统上电完成后，外部复位引脚必须输入高电平，否则系统将一直保持在复位状态。外部复位的时序如下：

- **外部复位（当且仅当外部复位引脚为使能状态）：**系统检测复位引脚的状态，如果复位引脚不为高电平，则系统会一直保持在复位状态，直到外部复位结束；
- **系统初始化：**所有的系统寄存器被置为初始状态；
- **振荡器开始工作：**振荡器开始提供系统时钟；
- **执行程序：**上电结束，程序开始运行。

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

3.6 外部复位电路

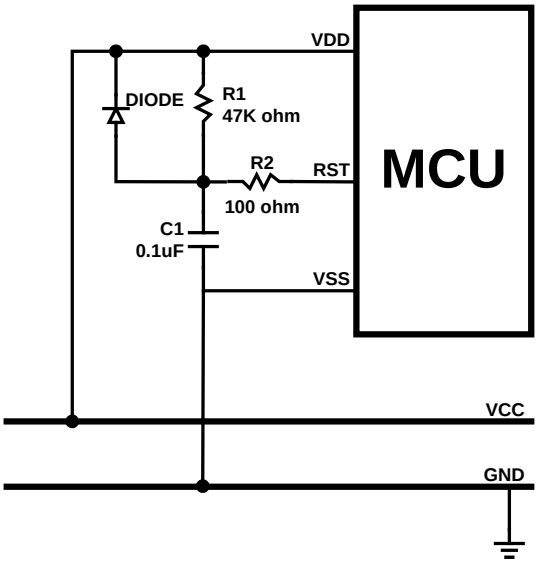
3.6.1 基本RC复位电路



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

✷ **注：此 RC 复位电路不能解决非正常上电和掉电复位问题。**

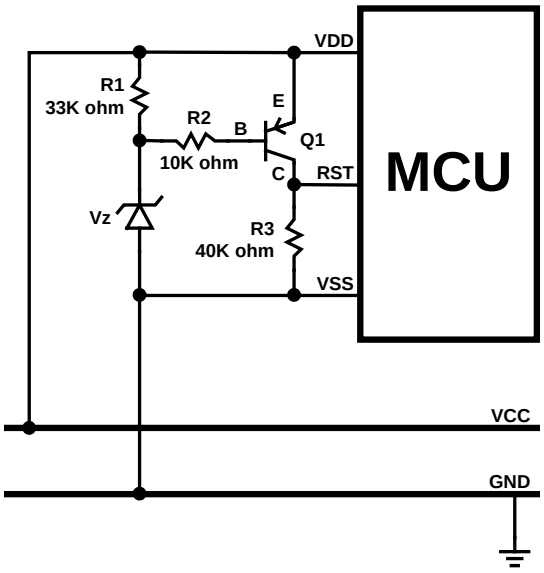
3.6.2 二极管&RC复位电路



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

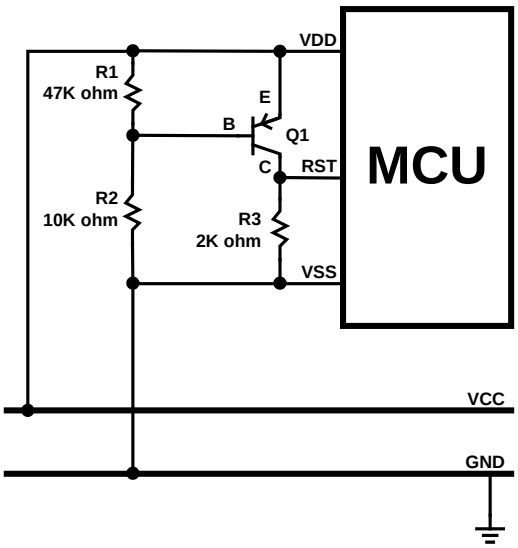
✷ **注：“基本 RC 复位电路”和“二极管及 RC 复位电路”中的电阻 R2 都是必不可少的限流电阻，以避免复位引脚 ESD（Electrostatic Discharge）或 EOS（Electrical Over-stress）击穿。**

3.6.3 稳压二极管复位电路



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

3.6.4 电压偏置复位电路

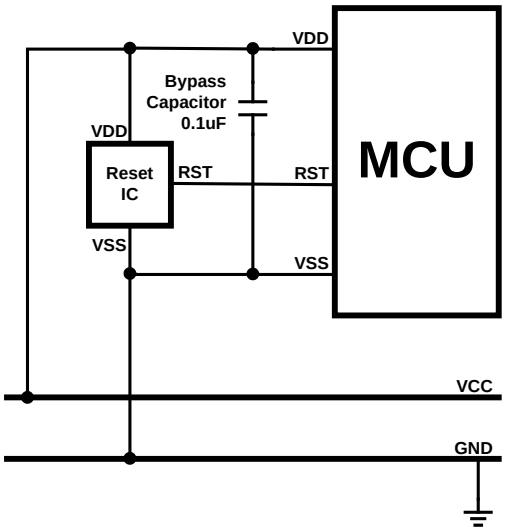


电压偏置复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，这种复位电路的检测电压值的精确度有所降低。电路中，R1 和 R2 构成分压电路，当 VDD 高于和等于分压值“0.7V x (R1 + R2) / R1”时，三极管集电极 C 输出高电平，单片机正常工作；VDD 低于“0.7V x (R1 + R2) / R1”时，集电极 C 输出低电平，单片机复位。

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

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

3.6.5 外部IC复位电路



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

4 系统时钟

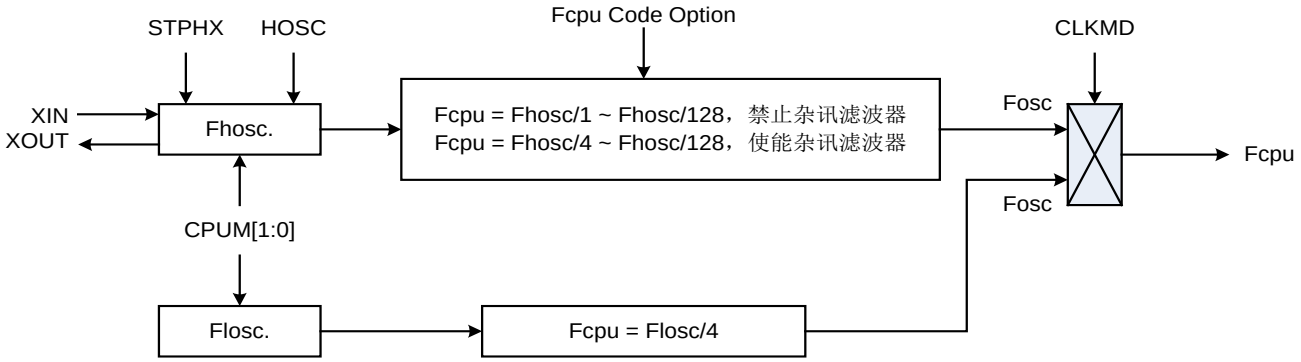
4.1 概述

SN8P2602C 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：外部高速时钟/内部高速 RC 时钟。
- Flosc：内部低速 RC 时钟（16KHz@3V，32KHz@5V）。
- Fosc：系统时钟源。
- Fcpu：指令周期。

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/128。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。

Fcpu 的范围由 Noise Filter 编译选项限定：禁止 Noise Filter 时，Fcpu=Fhosc/1~Fhosc/128；使能 Noise Filter 时，Fcpu=Fhosc/4~Fhosc/128，以减少杂讯的影响。

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。内部高速时钟支持实时时钟（RTC）功能，在IHRC_RTC模式下，内部高速时钟和外部32KHz振荡器有效，内部高速时钟作为系统时钟源，而外部32KHz振荡器为RTC时钟源，提供一个精确的实时时钟频率。

4.5 HIGH_CLK编译选项

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

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

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

4.5.1 内部高速RC振荡器（IHRC）

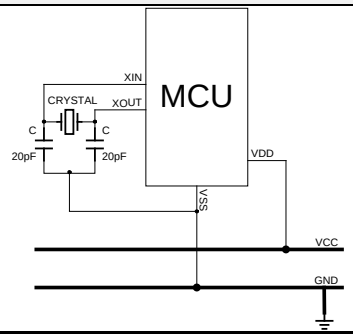
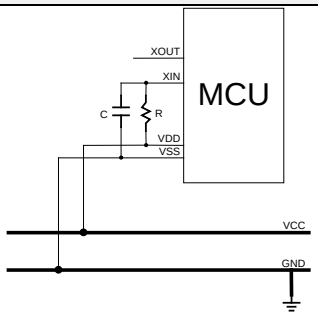
内部高速 16MHz RC 振荡器，普通环境下精确度为±2%，当选择 IHRC_16M 或者 IHRC_RTC 时，使能内部高速振荡器。

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

4.5.2 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。

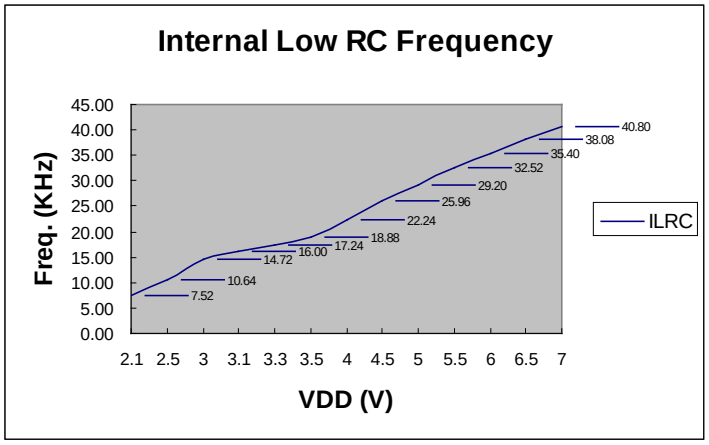
4.5.3 外部振荡应用电路

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。**

4.6 系统低速时钟

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



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

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

在睡眠模式下可以停掉内部低速 RC。

➢ **例：在睡眠模式下，停止内部低速振荡器。**

```
B0BSET      FCPUM0
```

✷ **注：不可以单独停止内部低速时钟；由寄存器 OSCM 的位 CPUM0 和 CPUM1（32K，禁止看门狗）的设置决定内部低速时钟的状态。**

4.7 OSCM寄存器

寄存器 OSCM 控制振荡器的状态和系统的工作模式。

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSCM	0	0	0	CPUM1	CPUM0	CLKMD	STPHX	0
读/写	-	-	-	R/W	R/W	R/W	R/W	-
复位后	-	-	-	0	0	0	0	-

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

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

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

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

4.8 系统时钟测试

在设计过程中，用户可通过软件指令周期对系统时钟速度进行测试。

➢ **例：外部振荡器的 Fcpu 指令周期测试。**

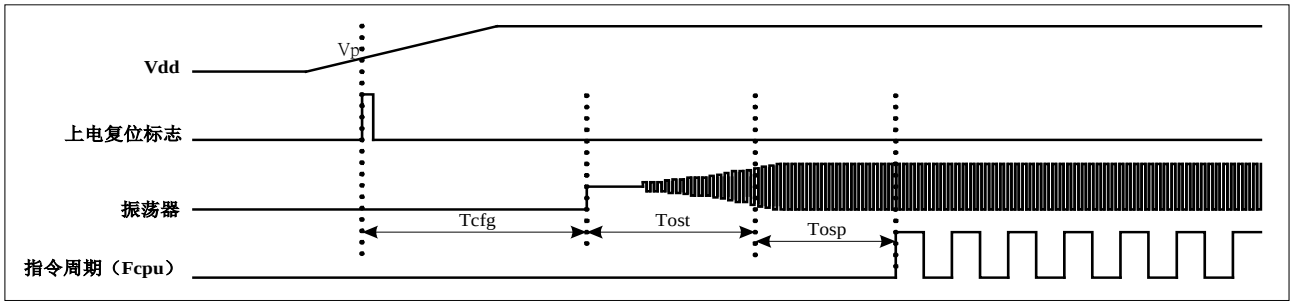
```
        B0BSET    P0M.0          ; P0.0 置为输出模式以输出 Fcpu 的触发信号。
@@:
        B0BSET    P0.0
        B0BCLR    P0.0
        JMP       @B
```

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

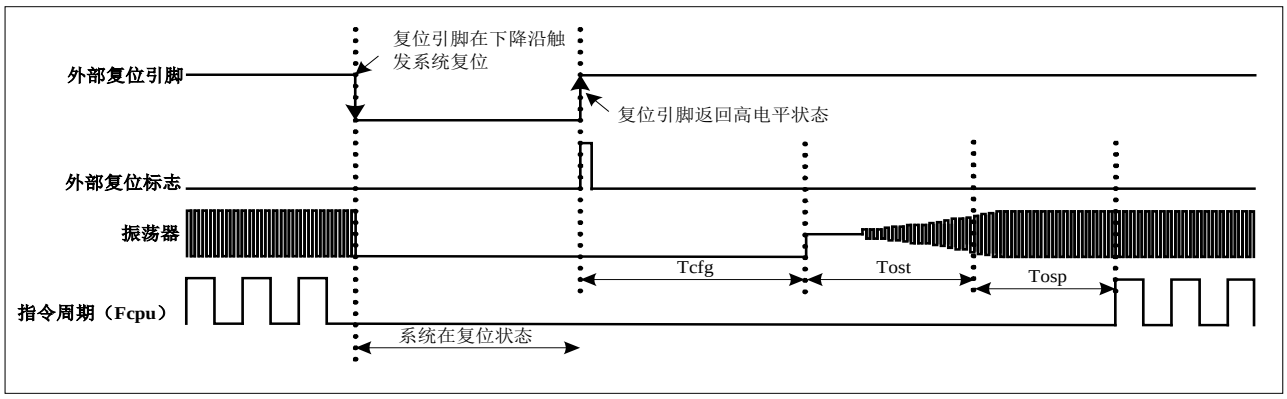
4.9 系统时钟时序

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

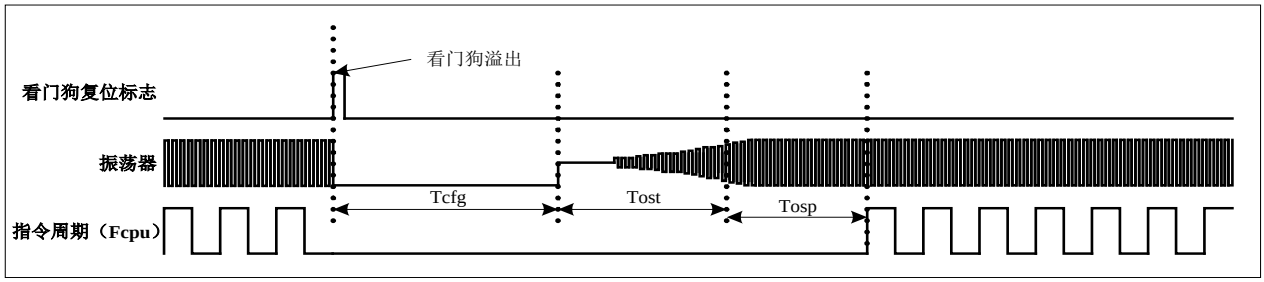
- 上电复位时序：



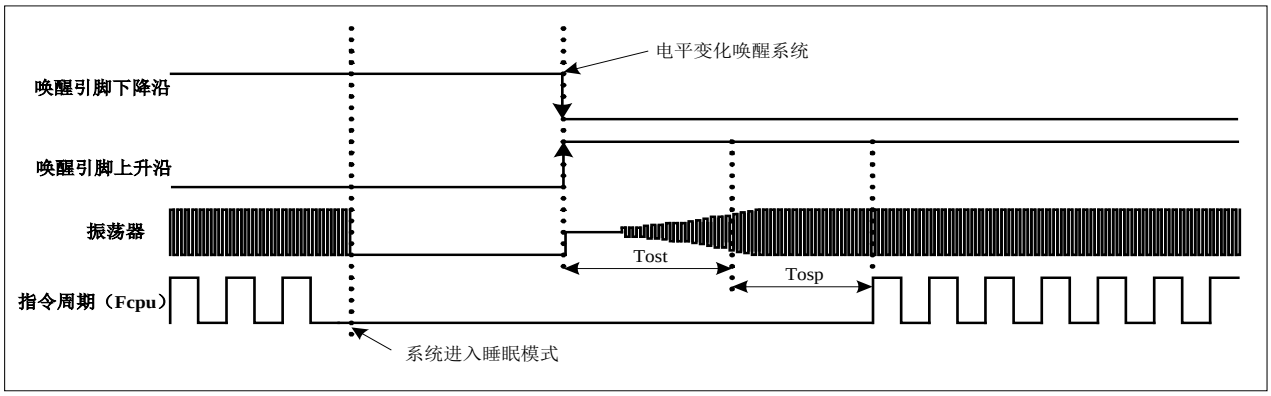
- 外部复位引脚复位时序：



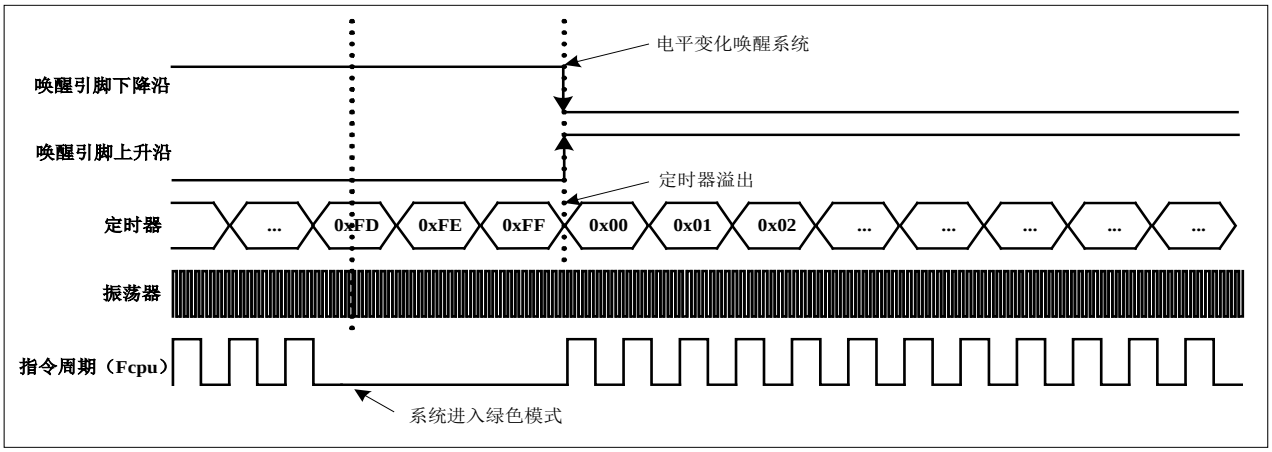
- 看门狗复位时序：



- **睡眠模式唤醒时序：**

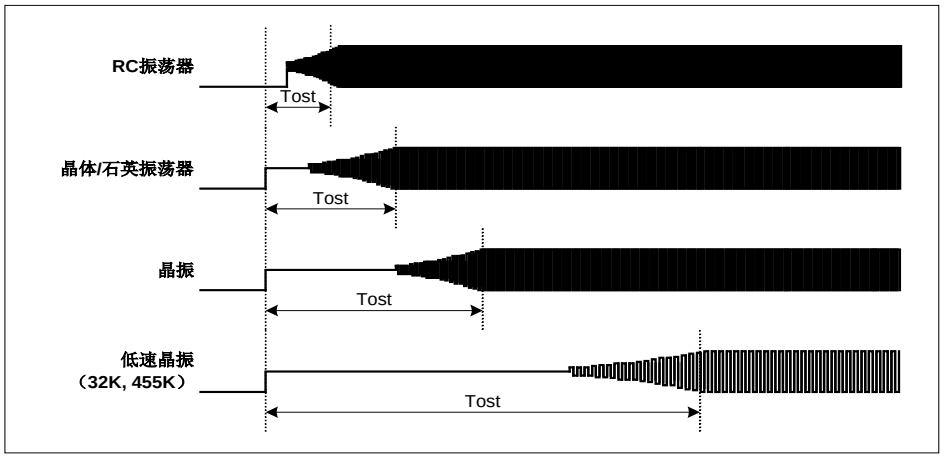


- **绿色模式唤醒时序：**



- **振荡器启动时间**

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



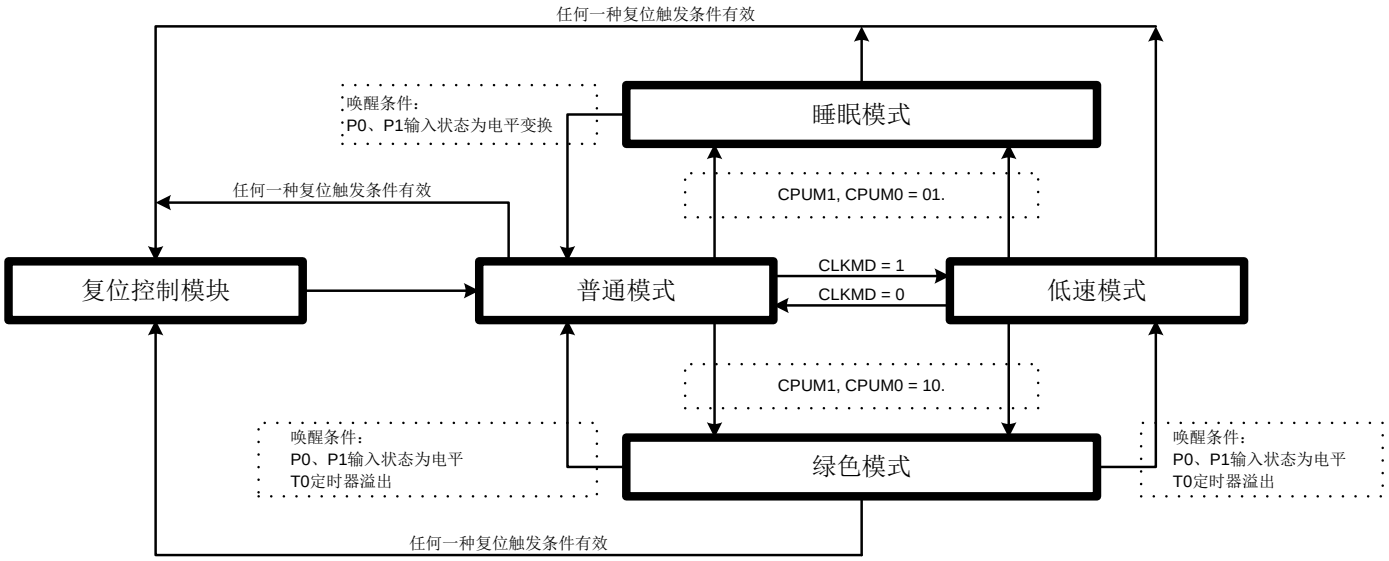
5 系统工作模式

5.1 概述

SN8P2602C 可以在 4 种工作模式下以不同的时钟频率工作，这些模式可以控制振荡器的工作、程序的执行以及模拟电路的功能损耗。

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

工作模式控制框图



工作模式时钟控制表

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

- EHOSC：外部高速振荡器（XIN/XOUT）。
- IHRC：内部高速 RC 振荡器。
- ILRC：内部低速 RC 振荡器。

5.2 普通模式

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

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

5.3 低速模式

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

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

5.4 睡眠模式

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

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

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

5.5 绿色模式

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

- 程序停止执行，所有的功能被禁止。
- 具有唤醒功能的定时器正常工作。
- 作为系统时钟源的振荡器正常工作，其它的振荡器工作状态取决于系统工作模式的配置。
- 由普通模式切换到绿色模式，被唤醒后返回到普通模式。
- 由低速模式切换到绿色模式，被唤醒后返回到低速模式。
- 绿色模式下的唤醒方式为 P0、P1 电平变换触发唤醒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 T0 唤醒功能。**

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

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

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

```
        CLR         T0C         ; 清 T0 计数器。
        B0BSET      FT0TB       ; 使能 T0 RTC 功能。
        B0BSET      FT0ENB      ; 使能 T0 定时器。

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

5.7 系统唤醒

5.7.1 概述

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

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

5.7.2 唤醒时间

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

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

唤醒时间的计算如下：

唤醒时间 = 1/Fosc * 32 (sec) + 高速时钟启动时间

➢ **例：睡眠模式下，系统被唤醒进入普通模式。唤醒时间的计算如下：**

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

✷ **注：高速时钟的启动时间与 VDD 和振荡器类型有关。**

5.7.3 P1W唤醒控制寄存器

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

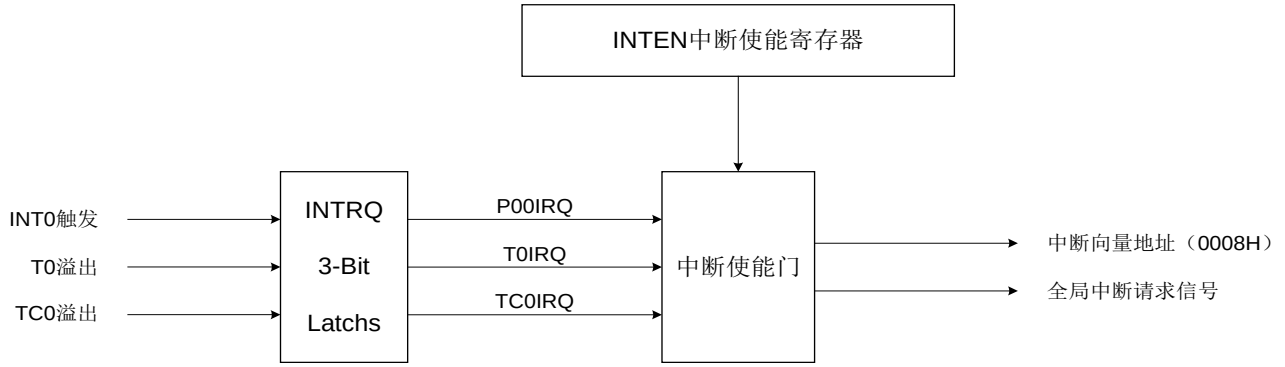
0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	-	P16W	-	P14W	P13W	P12W	P11W	P10W
读/写	-	W	-	W	W	W	W	W
复位后	-	0	-	0	0	0	0	0

Bit[3:0] **P10W~P16W**：P1 唤醒功能控制位。
0 = 禁止 P1n 唤醒功能；
1 = 使能 P1n 唤醒功能。

6 中断

6.1 概述

SN8P2602C 提供 3 个中断源：2 个内部中断（T0/TC0）和 1 个外部中断（INT0）。系统从睡眠模式进入高速普通模式时，外部中断能够将单片机唤醒。一旦程序进入中断，寄存器 STKP 的位 GIE 将被硬件自动清零以避免再次响应其它中断。系统退出中断后，硬件自动将 GIE 置“1”，以响应下一个中断。中断请求存放在寄存器 INTRQ 中。



✷ **注：程序响应中断时，位 GIE 必须处于有效状态。**

6.2 中断使能寄存器INTEN

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

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	-	-	TC0IEN	T0IEN	-	-	-	P00IEN
读/写	-	-	R/W	R/W	-	-	-	R/W
复位后	-	-	0	0	-	-	-	0

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

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

Bit 5 **TC0IEN**：TC0 中断控制位。
0 = 禁止；
1 = 使能。

6.3 中断请求寄存器INTRQ

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

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	-	-	TC0IRQ	T0IRQ	-	-	-	P00IRQ
读/写	-	-	R/W	R/W	-	-	-	R/W
复位后	-	-	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 4 **T0IRQ**：T0 中断请求标志。
0 = T0 无中断请求；
1 = T0 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。

6.4 全局中断GIE

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

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

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

➢ **例：设置全局中断控制位（GIE）。**

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

✸ **注：在所有中断中，GIE 都必须处于使能状态。**

6.5 PUSH, POP

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

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

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

```
                ORG     0
                JMP     START

                ORG     8
                JMP     INT_SERVICE

                ORG     10H
START:
                …

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

                RETI                    ; 退出中断。
                …
                ENDP
```

6.6 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。

✷ **注：P0.0 的中断触发方式由 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        ; INT0 置为电平触发。

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

➢ **例：INT0 中断。**

```
        ORG     8H              ;
        JMP     INT_SERVICE
INT_SERVICE:

        …                       ; ACC 和 PFLAG 入栈保护。

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

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

        RETI                    ; 退出中断。
```

6.7 T0 中断

T0C 计数器溢出时，不管 T0IEN 是否使能，T0IRQ 会被置“1”，此时若 T0IEN＝1，则系统响应 T0 中断；若此时 T0IEN=0，则系统并不会响应 T0 中断。

➢ **例：T0 中断请求设置。Fcpu = 4MHz / 4。**

```
        B0BCLR      FT0IEN          ; 禁止 T0 中断。
        B0BCLR      FT0ENB          ;
        MOV         A, #20H         ;
        B0MOV       T0M, A          ; T0 时钟= Fcpu / 64。
        MOV         A, # 64H        ; T0C 初始值置为 64H。
        B0MOV       T0C, A          ; T0 间隔为 10 ms。

        B0BCLR      FT0IRQ          ; T0 中断请求标志清零。
        B0BSET      FT0IEN          ; 允许响应 T0 中断。
        B0BSET      FT0ENB          ;

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：T0 中断程序。**

```
        ORG         8H              ;
        JMP         INT_SERVICE
INT_SERVICE:
        …                           ; ACC 和 PFLAG 入栈保存。
        B0BTS1      FT0IRQ          ; 检查是否有 T0 中断请求标志。
        JMP         EXIT_INT        ;

        B0BCLR      FT0IRQ          ; 清 T0IRQ。
        MOV         A, #64H
        B0MOV       T0C, A          ;
        …                           ; T0 中断程序。
EXIT_INT:
        …                           ; ACC 和 PFLAG 出栈恢复。

        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, # 74H        ; TC0C 初始值＝74H。
        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, #74H
        B0MOV       TC0C, A         ; 清 TC0C。
        …                           ; TC0 中断程序。
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。

        RETI                        ; 退出中断。
```

6.9 多中断操作

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

中断	有效触发
P00IRQ	由 PEDGE 控制
T0IRQ	T0C 溢出
TC0IRQ	TC0C 溢出

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

➢ **例：多中断条件下检测中断请求。**

```
            ORG         8H              ;
            JMP         INT_SERVICE
INT_SERVICE:
            …                           ; 保存 ACC 和 PFLAG。
INTP00CHK:                              ; 检查是否有 INT0 中断请求。
            B0BTS1      FP00IEN         ; 检查是否使能 INT0 中断。
            JMP         INTP01CHK       ; 跳到下一个中断。
            B0BTS0      FP00IRQ         ; 检查是否有 INT0 中断请求。
            JMP         INTP00          ; 进入 INT0 中断。
INTT0CHK:                               ; 检查是否有 T0 中断请求。
            B0BTS1      FT0IEN          ; 检查是否使能 T0 中断。
            JMP         INTTC0CHK       ; 跳到下一个中断。
            B0BTS0      FT0IRQ          ; 检查是否有 T0 中断请求。
            JMP         INTT0           ; 进入 T0 中断。
INTTC0CHK:                              ; 检查是否有 TC0 中断请求。
            B0BTS1      FTC0IEN         ; 检查是否使能 TC0 中断。
            JMP         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。
INT_EXIT:
            …                           ; 恢复 ACC 和 PFLAG。
            RETI                        ; 退出中断。
```

7 I/O口

7.1 概述

SN8P2602C 共有 16 个 I/O 引脚，大多数 I/O 引脚与模拟引脚及特殊功能的引脚共用，详见下表：

I/O 引脚		共用引脚		引脚共用控制条件
引脚名称	引脚类型	引脚名称	引脚类型	
P0.0	I/O	INT0	DC	P00IEN=1
P1.5	I	RST	DC	Reset_Pin code option = Reset
		VPP	HV	OTP Programming
P1.6	I/O	XIN	AC	High_CLK code option = IHRC_RTC, RC, 32K, 4M, 12M
P1.4	I/O	XOUT	AC	High_CLK code option = IHRC_RTC, 32K, 4M, 12M
P5.4	I/O	BZ0/PWM0	DC	TC0ENB=1, TC0OUT=1 or PWM0OUT=1
P5.5	I/O	BZOUT	DC	BZEN=1

*** DC：数字特性；AC：模拟特性；HV：高压特性。**

7.2 I/O口模式

寄存器 PnM 控制 I/O 口的工作模式。

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

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

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

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

✸ **注：用户可通过位操作指令（B0BSET、B0BCLR）对 I/O 口进行编程控制；**
✸ **注：P1.5 是单向输入引脚，P1M.5 未定义。**

➢ **例：I/O 模式选择。**

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

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

B0BCLR  P1M.0           ; P1.0 设为输入模式。

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

7.3 I/O口上拉电阻寄存器

I/O 引脚内置上拉电阻，仅在输入模式时有效，可通过 PnUR 寄存器编程控制。当 PnUR 寄存器的相关位置 0 时，禁止上拉电阻，置 1 时，使能上拉电阻。

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

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

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

✷ **注：P1.5 为单向输入引脚，无上拉电阻，故 P1UR.5 未定义。**

➢ **例：I/O 口的上拉电阻。**

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

7.4 I/O口下拉电阻寄存器

P5.0~P5.3 内置下拉电阻，仅在输入模式时有效，可通过下拉电阻寄存器 P5DR 编程控制。当 P5DR 寄存器的相关位置 0 时，禁止下拉电阻；置 1 时，使能下拉电阻。

0E6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5DR	-	-	-	-	P53DR	P52DR	P51DR	P50DR
读/写	-	-	-	-	W	W	W	W
复位后	-	-	-	-	0	0	0	0

➢ **例：I/O 口的下拉电阻。**

```
MOV     A, #0FH         ; 使能 P5.0~P5.3 的下拉电阻。
B0MOV   P5DR, A         ;
```

7.5 I/O口数据缓存器

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

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

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

✷ **注：当使能外部复位时， P15 的值保持为“1”。**

➢ **例：从输入口读取数据。**

```
B0MOV   A, P0     ; 从 P0 读数据。
B0MOV   A, P1     ; 从 P1 读数据。
B0MOV   A, P5     ; 从 P5 读数据。
```

➢ **例：写数据到输出端。**

```
MOV     A, #0FFH  ; 立即数 0FFH 写入所有输出口。
B0MOV   P0, A
B0MOV   P1, A
B0MOV   P5, A
```

➢ **例：写 1 位数据到输出口。**

```
B0BSET  P1.0      ; P1.0 和 P1.3 置“1”。
B0BSET  P1.3

B0BCLR  P1.0      ; P1.0 和 P1.3 置“0”。
B0BCLR  P1.3
```

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
```

➢ **例：用宏指令@RST_WDT 清看门狗定时器。**

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

看门狗定时器应用注意事项如下：

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

➢ **例：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。**

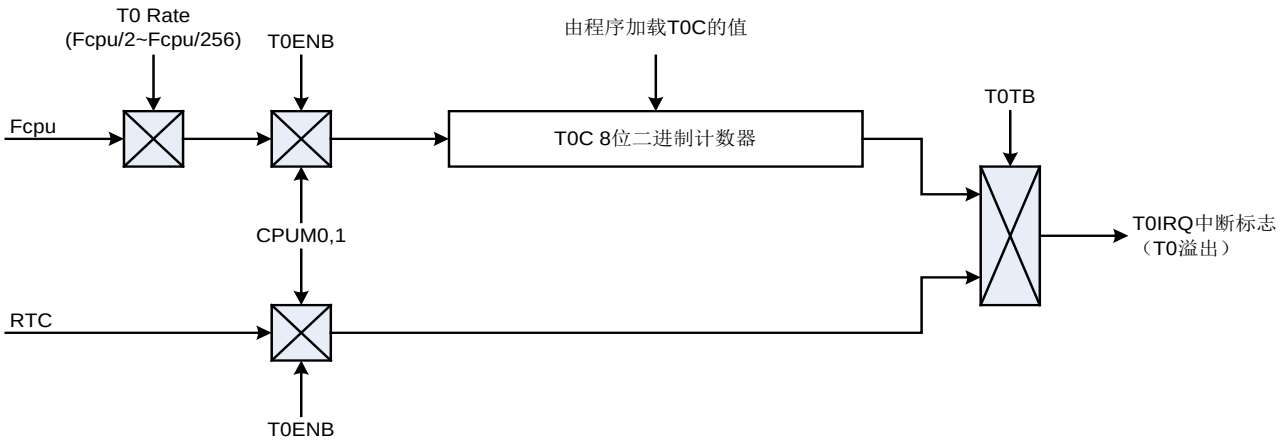
```
main:
        …                       ; 检测 I/O 口的状态。
        …                       ; 检测 RAM 的内容。
Err:    JMP $                   ; I/O 或 RAM 出错，不清看门狗等看门狗计时溢出。

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 8 位基本定时器T0

8.2.1 概述

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

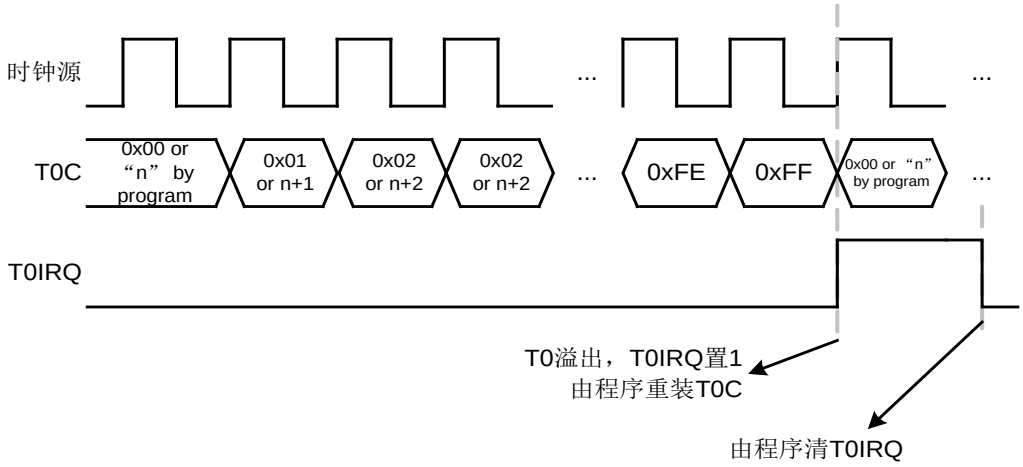
- **8 位可编程计数定时器：**根据选择的时钟频率周期性的产生中断请求。
- **中断功能：**T0 定时器支持中断功能，当 T0 溢出，T0IRQ 有效，程序计数器跳到中断向量地址执行中断。
- **RTC 功能：**T0 支持 RTC 功能。T0TB=1 时，RTC 时钟源由外部低速 32K 振荡器提供。RTC 功能仅在 High_Clk 选择 IHRC_RTC 时有效。
- **绿色模式唤醒功能：**T0 定时器在绿色模式下正常工作，溢出时将系统从绿色模式下唤醒。



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

8.2.2 T0 操作

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



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

T0rate[2:0]	T0 时钟	T0 间隔时间					
		Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4		IHRC_RTC 模式	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)	max. (sec)	Unit (ms)
000b	Fcpu/256	16.384	64	65.536	256	-	-
001b	Fcpu/128	8.192	32	32.768	128	-	-
010b	Fcpu/64	4.096	16	16.384	64	-	-
011b	Fcpu/32	2.048	8	8.192	32	-	-
100b	Fcpu/16	1.024	4	4.096	16	-	-
101b	Fcpu/8	0.512	2	2.048	8	-	-
110b	Fcpu/4	0.256	1	1.024	4	-	-
111b	Fcpu/2	0.128	0.5	0.512	2	-	-
-	32768Hz/64	-	-	-	-	0.5	1.953

8.2.3 T0M模式寄存器

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

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	T0TB
读/写	R/W	R/W	R/W	R/W	-	-	-	R/W
复位后	0	0	0	0	-	-	-	0

Bit0 T0TB：RTC 时钟源选择控制位。
0 = 禁止 RTC（T0 时钟源为 Fcpu）；
1 = 使能 RTC。

Bit [6:4] **T0RATE[2:0]**：T0 分频选择位。
000 = Fcpu/256；001 = Fcpu/128；010 = Fcpu/64；011 = Fcpu/32；100 = Fcpu/16；101 = Fcpu/8；110 = Fcpu/4；111 = Fcpu/2。

Bit 7 T0ENB：T0 启动控制位。
0 = 禁止；
1 = 使能。

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

8.2.4 T0C计数寄存器

8 位计数器 T0C 溢出时，T0IRQ 置 1 并由程序清零，用来控制 T0 的中断间隔时间。必须保证写入正确的值到 T0C 寄存器，然后使能 T0 定时器以保证第一个周期准确无误。T0 溢出后，由程序加载一个正确的值到 T0C 寄存器。

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

T0C 初始值的计算公式如下：

T0C 初始值 = 256 -（T0 中断间隔时间 * T0 时钟 RATE）

➢ **例：T0 的中断间隔时间为 10ms，T0 时钟源为 Fcpu = 4MHz/4/=1MHz，T0RATE = 001（Fcpu/128）。**
T0 中断间隔时间为 10ms，T0 时钟 rate＝4MHz/4/128

T0C 初始值 = 256 -（T0 中断间隔时间 * T0 时钟 rate）
= 256 -（10ms * 4MHz / 4/ 128）
= 256 -（10-2 * 4* 106 / 4/128）
= B2H

✷ **注：RTC 模式下，T0C 为 256，T0 的间隔时间为 0.5S。不能在 RTC 模式下修改 T0C 的值。**

8.2.5 T0 操作举例

- **T0 定时器：**

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

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

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

; 清 T0IRQ。
        B0BCLR  FT0IRQ

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

- **T0 在 RTC 模式下工作：**

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

; 设置 T0 RTC 功能。
        B0BSET  FT0TB

; 清 T0C。
        CLR     T0C

; 清 T0IRQ。
        B0BCLR  FT0IRQ

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

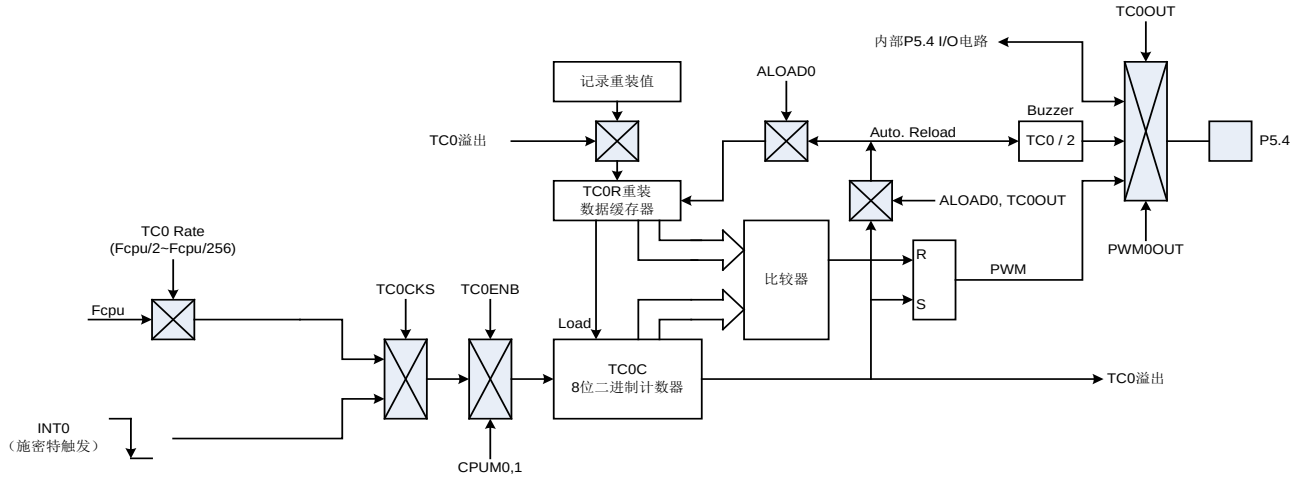
8.3 8 位定时/计数器TC0

8.3.1 概述

8 位二进制定时/计数器具有基本定时器、事件计数器、Buzzer 和 PWM 功能。基本定时器功能可以支持标志显示（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 Buzzer 和 PWM 功能，Buzzer 和 PWM 的周期和分辨率由 TC0 时钟 Rate、TC0R 寄存器控制，故具有良好性能的 Buzzer 和 PWM 可以处理 IR 载波信号，马达控制和光度调节等。

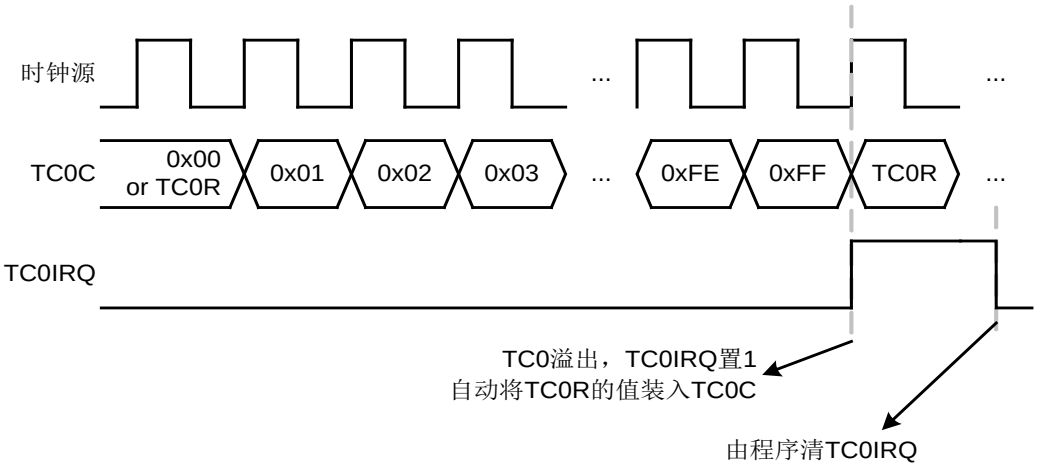
TC0 的主要用途如下：

- **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 和 TC0R 寄存器控制占空比/周期；
- **Buzzer 输出：**Buzzer 的输出信号是 TC0 间隔时间的 1/2 个周期；
- **绿色模式功能：**绿色模式下，TC0 正常工作，但无唤醒功能。



8.3.2 TC0 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 进入新的工作状态。定时器/计数器模式下，由 ALOAD0 控制自动重装功能；PWM 模式下，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，但不能唤醒系统。



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

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)
000b	Fcpu/256	16.384	64	65.536	256
001b	Fcpu/128	8.192	32	32.768	128
010b	Fcpu/64	4.096	16	16.384	64
011b	Fcpu/32	2.048	8	8.192	32
100b	Fcpu/16	1.024	4	4.096	16
101b	Fcpu/8	0.512	2	2.048	8
110b	Fcpu/4	0.256	1	1.024	4
111b	Fcpu/2	0.128	0.5	0.512	2

8.3.3 TC0M模式寄存器

模式寄存器 TC0M 控制 TC0 的工作模式：包括 TC0 前置分频，时钟源，PWM 功能等，这些功能必须在使能 TC0 定时器之前设置完成。

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 输出控制位。
0 = 禁止 PWM 输出，P5.4 为普通 I/O 引脚；
1 = 允许 PWM 输出，P5.4 输出 PWM 信号。PWM 占空比由 TC0OUT、ALOAD0 位控制。

Bit 1 **TC0OUT**：TC0 定时器触发信号输出控制位，仅在 PWM0OUT=0 时有效。
0 = 禁止，P5.4 为普通 I/O 引脚；
1 = 使能，P5.4 输出 TC0OUT 信号。

Bit 2 **ALOAD0**：自动重装功能控制位。仅在 PWM0OUT = 0 时有效。
0 = 禁止；
1 = 使能。

Bit 3 **TC0CKS**：TC0 时钟源选择位。
0 = 内部时钟（Fcpu）；
1 = 外部时钟信号（P0.0/INT0），使能事件计数器功能，TC0Rate[2:0]处于无效状态。

Bit [6:4] **TC0RATE[2:0]**：TC0 分频选择位。
000 = fcpu/256；001 = fcpu/128；010 = Fcpu/64；011 = Fcpu/32；100 = Fcpu/16；101 = Fcpu/8；
110 = Fcpu/4；111 = Fcpu/2。

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

✷ **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）。**

8.3.4 TC0C计数寄存器

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

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

TC0C 初始值的计算公式如下：

TC0C 初始值 = N -（TC0 中断间隔时间 * TC0 时钟 rate）

N 为 TC0 十进制计数范围。各模式下参数的设定如下表所示：

TC0CKS	PWM0	ALOAD0	TC0OUT	N	TC0C 范围	TC0C 二进制计数范围	注释
0	0	x	x	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出
	1	0	0	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出
	1	0	1	64	00H~3FH	xx000000b~xx111111b	每计数 64 次溢出
	1	1	0	32	00H~1FH	xxx00000b~xxx11111b	每计数 32 次溢出
	1	1	1	16	00H~0FH	xxxx0000b~xxxx1111b	每计数 16 次溢出
1	-	-	-	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出

8.3.5 TC0R自动重装寄存器

TC0 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。

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 初始值的计算公式如下：

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

N 为 TC0 十进制计数范围。各模式下参数的设定如下表所示：

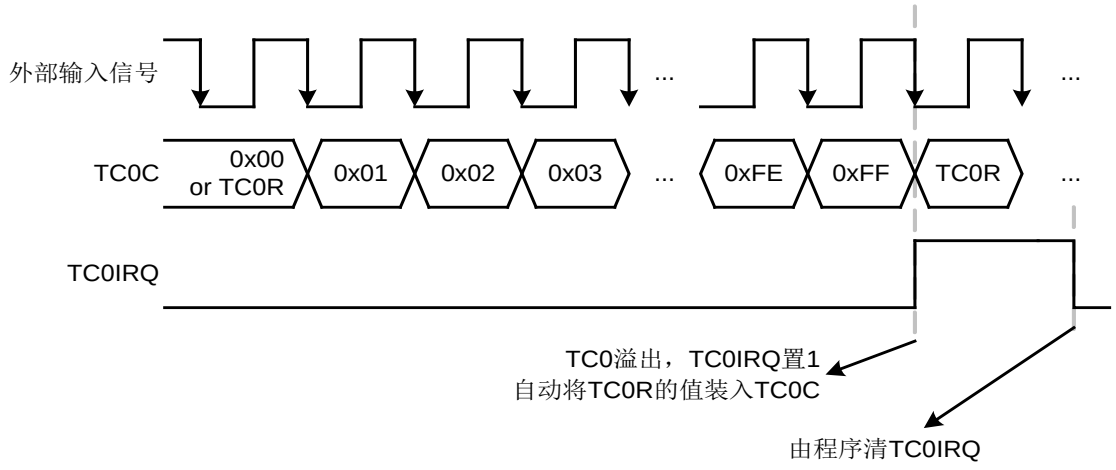
TC0CKS	PWM0	ALOAD0	TC0OUT	N	TC0C 范围	TC0C 二进制计数范围
0	0	x	x	256	00H~0FFH	00000000b~11111111b
	1	0	0	256	00H~0FFH	00000000b~11111111b
	1	0	1	64	00H~3FH	xx000000b~xx111111b
	1	1	0	32	00H~1FH	xxx00000b~xxx11111b
	1	1	1	16	00H~0FH	xxxx0000b~xxxx1111b
1	-	-	-	256	00H~0FFH	00000000b~11111111b

➢ **例：计算 TC0C 和 TC0R 的值，TC0 间隔时间为 10ms，时钟源 Fcpu=4MHz/4=1MHz，TC0RATE = 001（Fcpu/128）。TC0 间隔时间为 10ms，TC0 时钟 rate 为 4Hz/428**

TC0R 初始值 = 256 -（TC0 中断间隔时间 * 输入时钟源）
= 256 -（10ms * 4Hz / 4/ 128）
= 256 -（10-2 * 4* 106 / 4/ 128）
= 0B2H

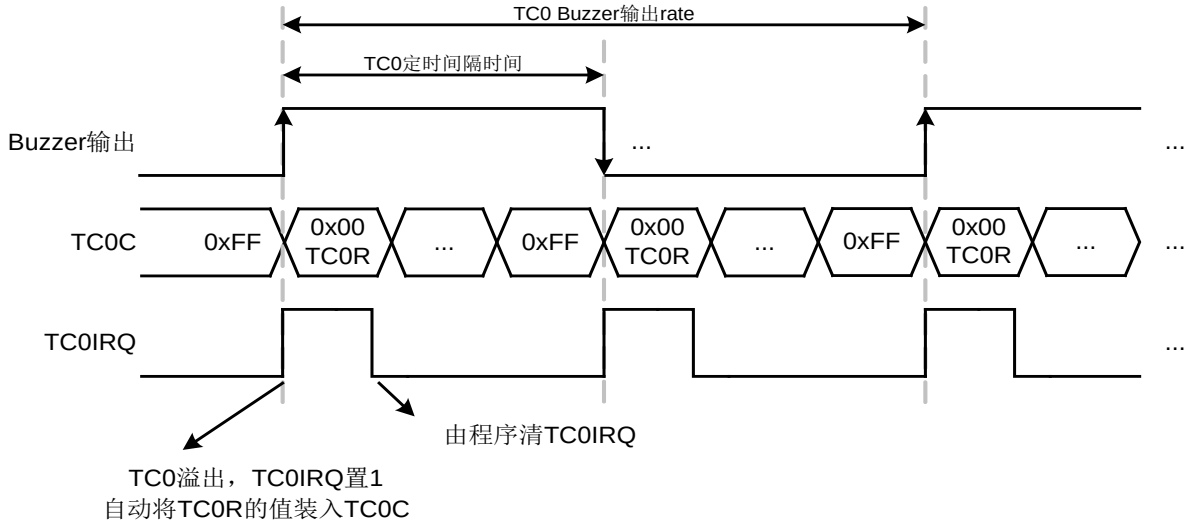
8.3.6 TC0 事件计数器

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



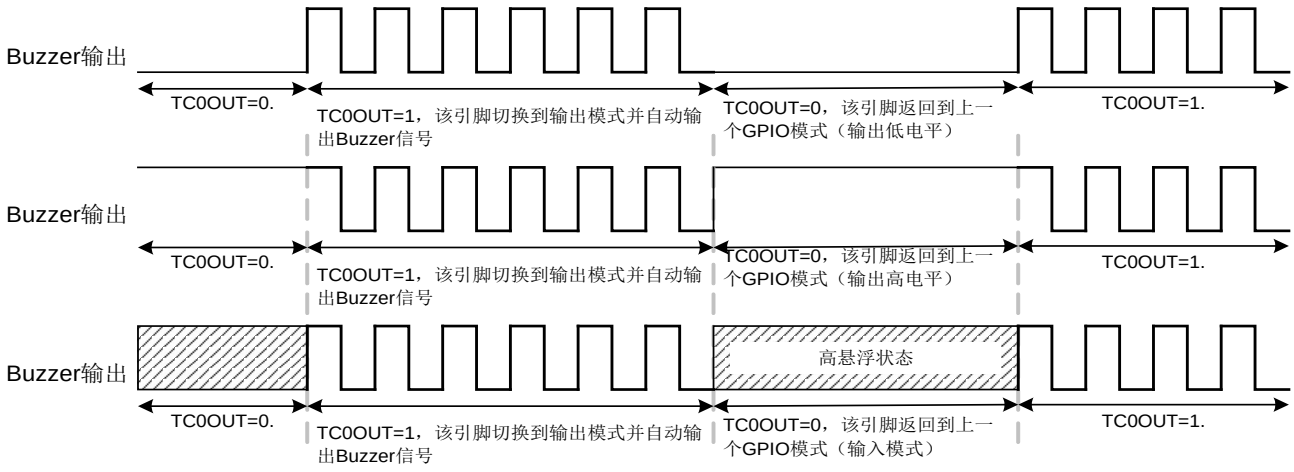
8.3.7 TC0 BUZZER输出

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



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

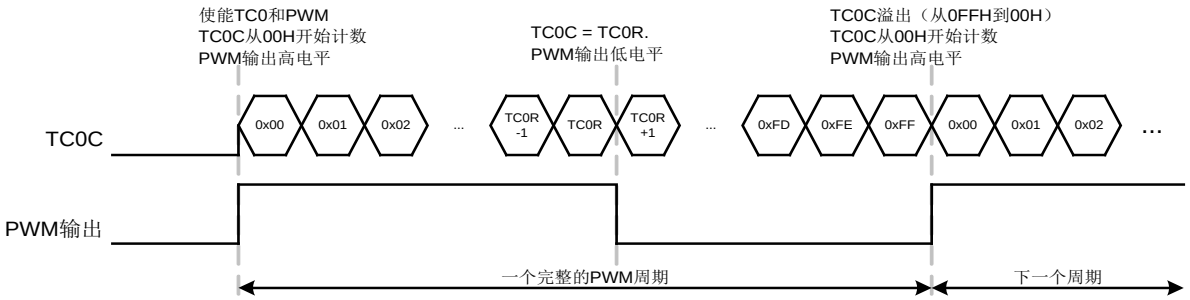
Buzzer 输出引脚与 GPIO 引脚共用，TC0OUT=1 时，该引脚自动设为 Buzzer 输出引脚。如清 TC0OUT 位以禁止 Buzzer 输出后，该引脚自动返回到最后一个 GPIO 模式。



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

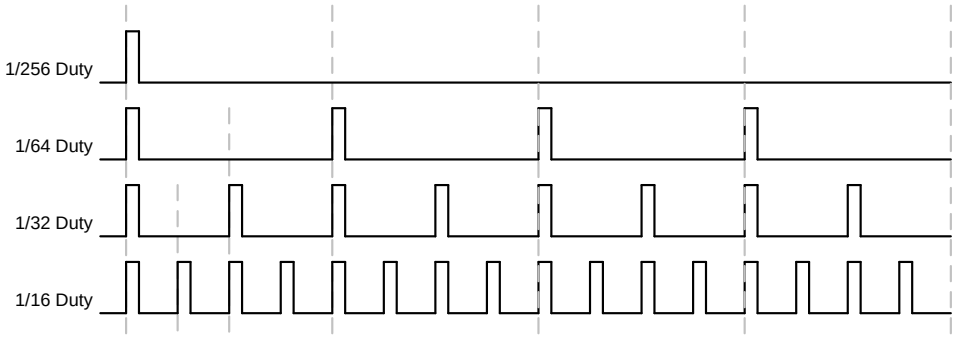
8.3.8 脉宽调制（PWM）

可编程控制占空比/周期的 PWM 可以提供不同的 PWM 信号。使能 TC0 定时器且 PWM0OUT=1 时，由 PWM 输出引脚（P5.4）输出 PWM 信号。PWM 首先输出高电平，然后输出低电平。TC0Rate[2:0]控制 PWM 的周期，ALOAD0 和 TC0OUT 决定 PWM 的分辨率，TC0R 寄存器决定 PWM 的占空比（脉冲高电平的长度）。开启 TC0 定时器且定时器溢出后，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。TC0 溢出时，PWM 的一个周期完成。在 PWM 输出的过程由程序更改 PWM 的占空比，则在下一个周期开始输出新的占空比的 PWM 信号。

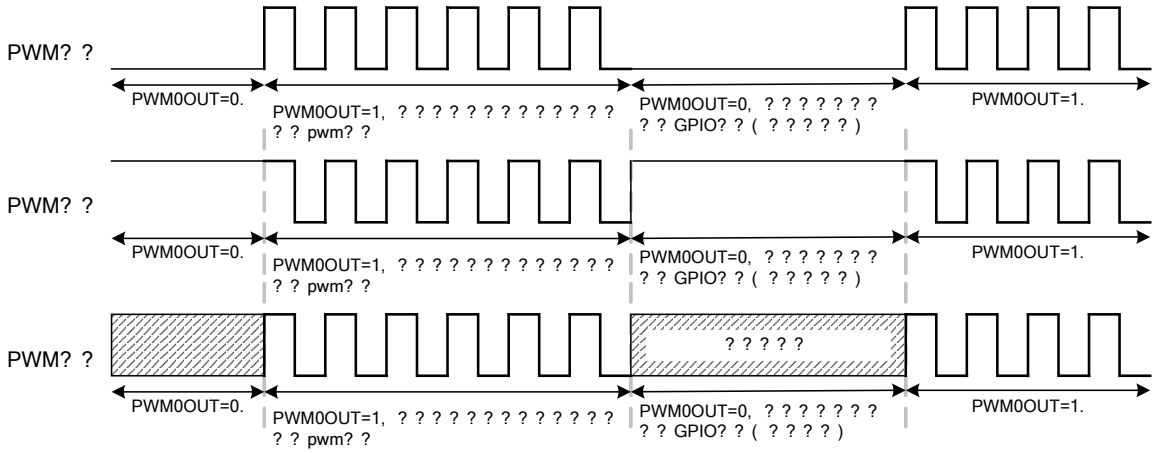


PWM 的分辨率由 ALOAD0 和 TC0OUT 决定，以实现高速 PWM 信号。当 ALOAD0、TC0OUT = 00 时，PWM 的分辨率为 1/256；ALOAD0、TC0OUT = 01 时，PWM 的分辨率为 1/64；ALOAD0、TC0OUT = 10 时，PWM 的分辨率为 1/32；ALOAD0、TC0OUT = 11 时，PWM 的分辨率为 1/16。若需调制 PWM 的分辨率，TC0R PWM 的占空比控制范围必须调制到一个合适的分辨率。PWM 输出过程中，TC0 溢出时，TC0IRQ 有效，TC0IEN=1 时，则使能 TC0 中断。但强烈建议小心同时使用 PWM 和 TC0 定时器功能，保证两种功能都能正常工作。

ALOAD0	TC0OUT	PWM 分辨率	TC0R 有效值	TC0R 有效值（二进制）
0	0	256	00H~FFH	00000000b~11111111b
0	1	64	00H~3FH	xx000000b~xx111111b
1	0	32	00H~1FH	xxx00000b~xxx11111b
1	1	16	00H~0FH	xxxx0000b~xxxx1111b



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



8.3.9 TC0 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   ; TC0rate[2:0]。
        B0MOV       TC0M, A
        B0BSET      FALOAD0

; 设置 TC0C 和 TC0R 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 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   ; TC0rate[2:0]。
        B0MOV       TC0M, A
        B0BSET      FALOAD0

; 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

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

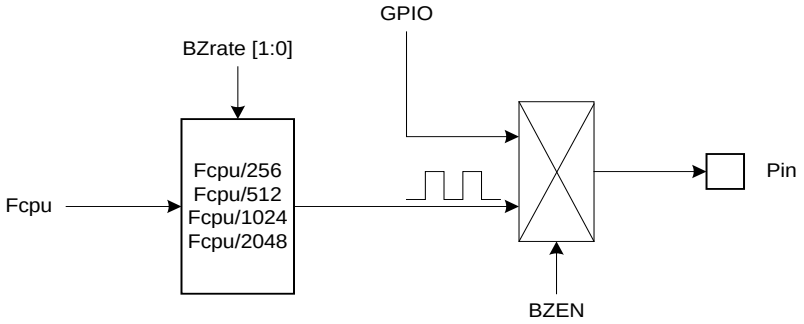
- **TC0 PWM**

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

; 设置 TC0Rate 和 PWM 周期。
        MOV         A, #0nnn0000b       ; TC0rate[2:0]。
        B0MOV       TC0M, A

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

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

; 清 TC0C。
        CLR         TC0C

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

9 2K/4K BUZZER产生器

9.1 概述

SN8P2602C 内置 Buzzer 产生器，用来驱动外部 buzzer 装置，可以驱动 2KHz 或者 4KHz buzzer。通过 BZM 寄存器调整 buzzer 输出频率。Buzzer 输出引脚与 GPIO 引脚共用，BZEN=1 时，该引脚输出 buzzer 载波信号；BZEN=0 时，该引脚返回到上一个 GPIO 模式（输入模式、输出高或输出低模式）。



Buzzer 频率由 Fcpu（指令周期）分频得到，由 BZRate 位控制，即 Fcpu 决定 buzzer 的频率。具体见下表：

BZrate [1:0]	Buzzer Rate Division	Buzzer Rate		
		Fcpu = 1MHz	**Fcpu = 2MHz**	**Fcpu = 4MHz**
00	Fcpu/256	**4KHz**	8KHz	16KHz
01	Fcpu/512	**2KHz**	**4KHz**	8KHz
10	Fcpu/1024	1KHz	**2KHz**	**4KHz**
11	Fcpu/2048	0.5KHz	1KHz	**2KHz**

Buzzer 的目标频率是 2KHz 和 4KHz，故必须选择合适的 Fcpu rate 已获取正确的 buzzer 频率。上表显示了 2KHz/4KHz buzzer 频率的配置。

9.2 BZM寄存器

0DCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
BZM	BZEN	BZrate1	BZrate0	-	-	-	-	-
读/写	R/W	R/W	R/W	-	-	-	-	-
复位后	0	0	0	-	-	-	-	-

Bit 7 **BZEN**：Buzzer 输出控制位。
0 = 禁止 BZ 输出，BZ 输出引脚返回到上一个 GPIO 模式；
1 = 使能 BZ 输出，禁止 BZ 输出引脚的 GPIO 功能。

Bit[6:5] **BZrate[1:0]**：Buzzer rate 控制位。
00 = Fcpu/256；
01 = Fcpu/512；
10 = Fcpu/1024；
11 = Fcpu/2048。

✷ **注：**
1、若 BZEN=0，禁止 buzzer 输出后，BZ 输出引脚返回到上一个 GPIO 模式。
2、若 BZEN=1，使能 buzzer 输出引脚的 buzzer 输出功能，禁止该引脚的 GPIO 功能。

10 指令集

Field	指令格式		描述	C	DC	Z	周期
MOVE	MOV	A,M	A ← M	-	-	√	1
	MOV	M,A	M ← A	-	-	-	1
	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I。(M 仅适用地址是 0x80~0x87 的系统寄存器，如 R、Y、Z...，。)	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)。	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
ARITHMETIC	ADC	A,M	A ← A + M + C，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
LOGIC	AND	A,M	A ← A 与 M。	-	-	√	1
	AND	M,A	M ← A 与 M。	-	-	√	1+N
	AND	A,I	A ← A 与 I。	-	-	√	1
	OR	A,M	A ← A 或 M。	-	-	√	1
	OR	M,A	M ← A 或 M。	-	-	√	1+N
	OR	A,I	A ← A 或 I。	-	-	√	1
	XOR	A,M	A ← A 异或 M。	-	-	√	1
	XOR	M,A	M ← A 异或 M。	-	-	√	1+N
	XOR	A,I	A ← A 异或 I。	-	-	√	1
PROCESS	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)。	-	-	-	1
	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)。	-	-	-	1+N
	RRC	M	A ← 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
BRANCH	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
MISC	RET		子程序跳出指令，PC ← Stack。	-	-	-	2
	RETI		中断处理程序跳出指令，PC ← Stack，并使能全局中断控制位。	-	-	-	2
	PUSH		保存 ACC 和 PFLAG（不包括 NT0 和 NPD）。	-	-	-	1
	POP		恢复 ACC 和 PFLAG（不包括 NT0 和 NPD）。	√	√	√	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)
SN8P2602CP, SN8P2602CS, SN8P2602CX .....-10°C ~ + 70°C
SN8P2602CPD, SN8P2602CSD, SN8P2602CXD ..... –40°C ~ + 85°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

11.2 电气特性

- **DC CHARACTERISTIC**

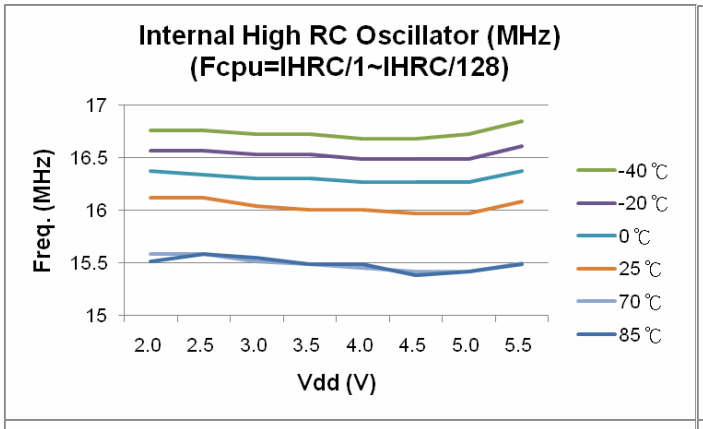
(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 4MHz,Fcpu=1MHz,ambient temperature is 25°C unless otherwise note.)

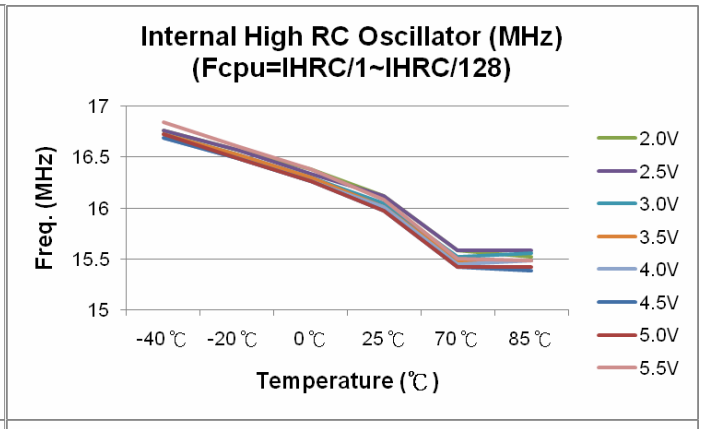
PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd, 25°C, Fcpu = 1MHz		2.2	-	5.5	V
		Normal mode, Vpp = Vdd, -40°C~85°C		2.4	-	5.5	V
RAM Data Retention voltage	Vdr			1.5	-	-	V
*Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	All input ports		Vss	-	0.3Vdd	V
	ViL2	Reset pin		Vss	-	0.2Vdd	V
Input High Voltage	ViH1	All input ports		0.7Vdd	-	Vdd	V
	ViH2	Reset pin		0.9Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd		-	-	2	uA
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
		Vin = Vss , Vdd = 5V		50	100	150	
I/O output source current sink current	IoH	Vop = Vdd – 0.5V		8	15	-	mA
	IoL1	Vop = Vss + 0.5V		8	15	-	
	IoL2	Vop = Vss + 0.5V, P5.0~P5.3, P5.5		20	40	-	
1/2 * Bias Voltage	Vbias	P5.0~P5.3 pull-up / pull-down resistors enable.		2	2.5	3	V
*INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current	Idd1	Run Mode (Low power disable)	Vdd= 3V, Fcpu = 16MHz	-	2.8	-	mA
			Vdd= 5V, Fcpu = 16MHz	-	5.8	-	mA
			Vdd= 3V, Fcpu = 4MHz	-	1.5	-	mA
			Vdd= 5V, Fcpu = 4MHz	-	3	-	mA
			Vdd= 3V, Fcpu = 1MHz	-	1.1	-	mA
			Vdd= 5V, Fcpu = 1MHz	-	2.3	-	mA
			Vdd= 3V, Fcpu = 32KHz	-	20	-	uA
			Vdd= 5V, Fcpu = 32KHz	-	45	-	uA
	Idd2	Run Mode (Low power enable)	Vdd= 3V, Fcpu = 4MHz	-	1.3	-	mA
			Vdd= 5V, Fcpu = 4MHz	-	2.2	-	mA
			Vdd= 3V, Fcpu = 1MHz	-	0.7	-	mA
			Vdd= 5V, Fcpu = 1MHz	-	1	-	mA
	Idd3	Slow Mode (Internal low RC, Stop high clock)	Vdd= 3V, ILRC=16KHz	-	2.5	-	uA
			Vdd= 5V, ILRC=32KHz	-	7.8	-	uA
	Idd4	Sleep Mode	Vdd= 5V/3V	-	1	2	uA
	Idd5	Green Mode (No loading, Watchdog Disable)	Vdd= 3V, IHRC=16MHz	-	0.45	-	mA
			Vdd= 5V, IHRC=16MHz	-	0.5	-	mA
			Vdd= 3V, Ext. 32KHz X'tal	-	6	-	uA
			Vdd= 5V, Ext. 32KHz X'tal	-	18	-	uA
			Vdd= 3V, ILRC=16KHz	-	1.5	-	uA
			Vdd= 5V, ILRC=32KHz	-	4.5	-	uA
Internal High Oscillator Freq.	Fihrc	Internal Hihg RC (IHRC)	25°C, Vdd=2.2V~ 5.5V Fcpu=Fhosc/1~Fhosc/128	15.68	16	16.32	MHz
			-40°C~85°C,Vdd=2.4V~ 5.5V Fcpu=Fhosc/1~Fhosc/128	15.2	16	16.8	MHz
LVD Voltage	Vdet0	Low voltage reset level. -40°C~85°C		1.6	2.0	2.3	V
	Vdet1	Low voltage reset/indicator level. -40°C~85°C		1.8	2.4	3	V
	Vdet2	Low voltage reset/indicator level. -40°C~85°C		2.5	3.6	4.5	V

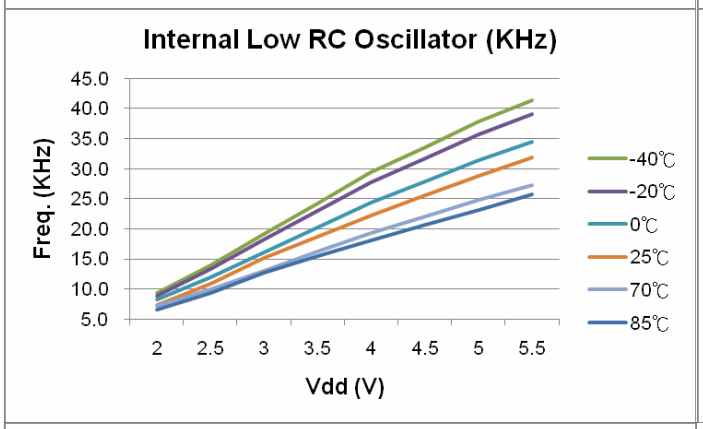
" *" These parameters are for design reference, not tested.

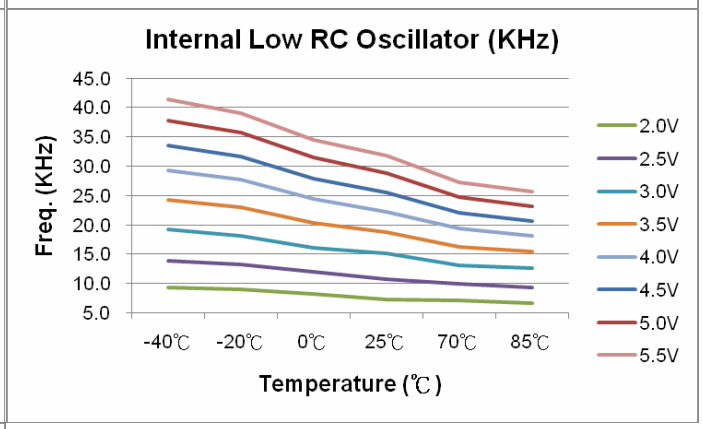
11.3 特性曲线图

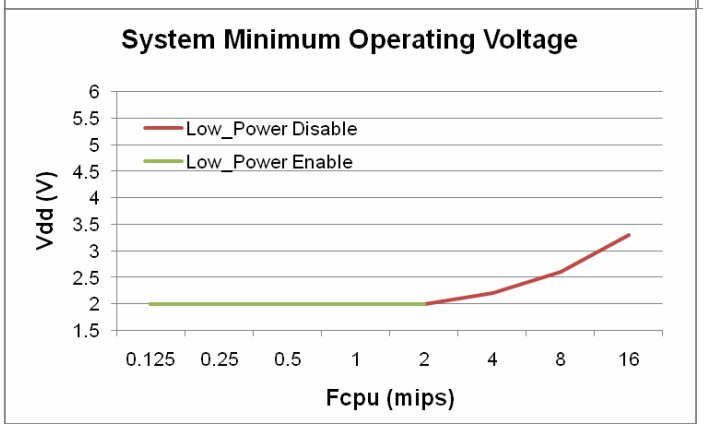
本章所列的各曲线图仅作设计参考，其中给出的部分数据可能超出了芯片指定的工作范围，为保证芯片的正常工作，请严格参照电气特性说明。











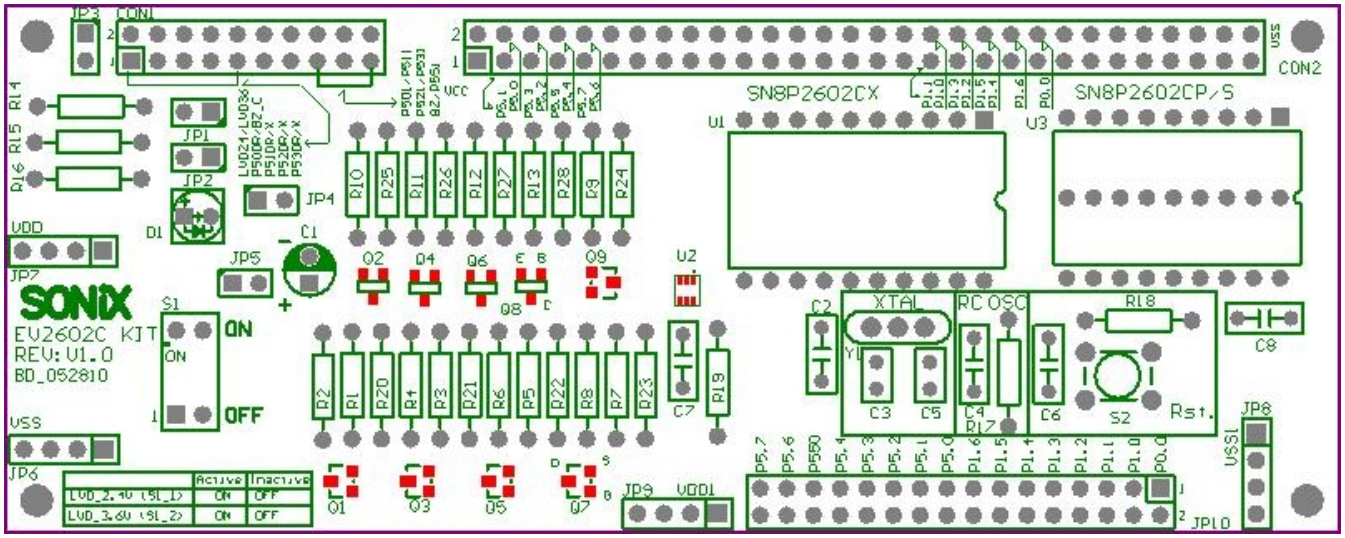
12 开发工具

在进行 SN8P2602C 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。（仿真 IHRC 功能时，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：SN8P2602C_EV-kit Rev: V1.0。**
- **IDE：SONiX IDE M2IDE_V129 或更晚的版本。**
- **Writer：MPIII writer。**
- **Writer 转接板：SN8P2602C。**

12.1 SN8P2602C EV-kit

SN8P2602C EV-kit PCB 外形图如下：

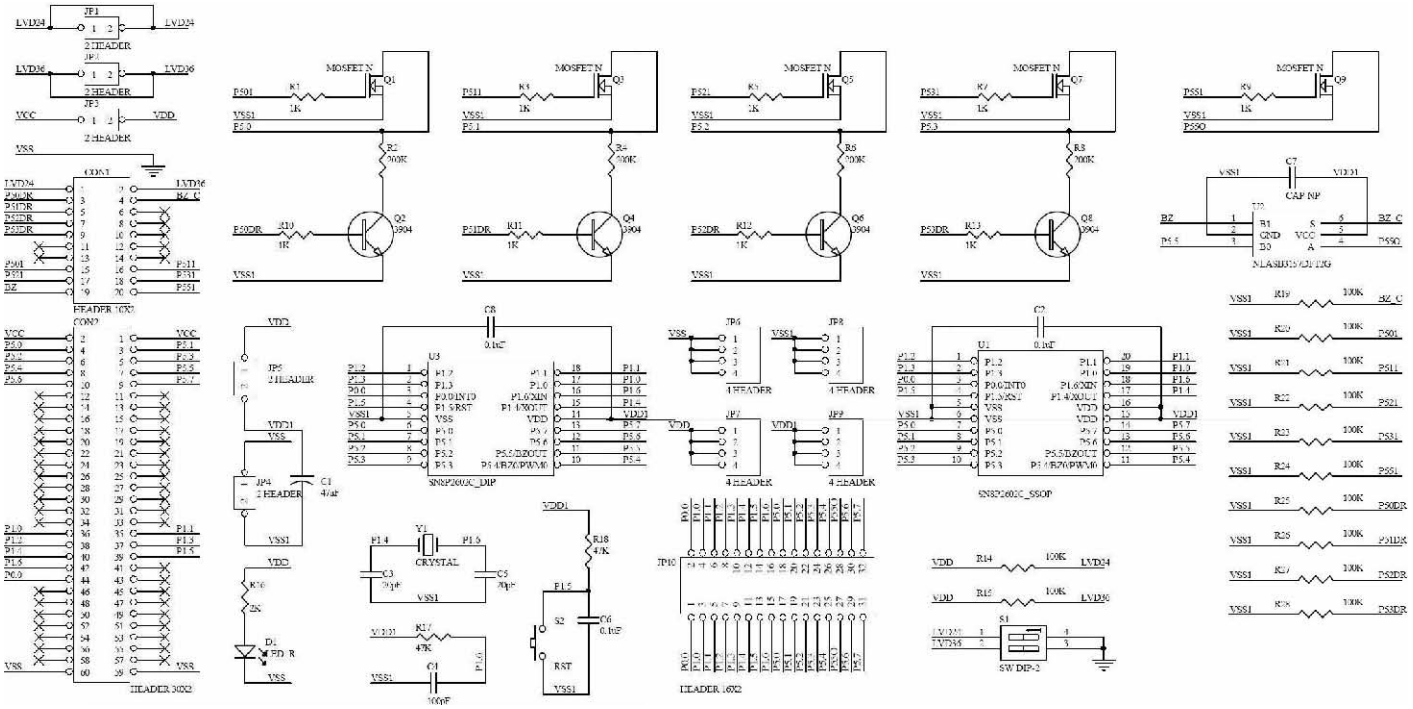


- CON2：连接到 SN8ICE2K Plus 2 CON1（包括 GPIO、EV-Kit 控制信号等）。
- CON1：连接到 SN8ICE2K Plus 2 JP3（EV-Kit 与 ICE 的通讯总线，控制信号等）。
- S1：LVD24V / LVD36V 控制开关，仿真 LVD2.4V 标志/复位功能和 LVD3.6V/标志功能。

开关编号	ON	OFF
LVD24	LVD 2.4V 有效	LVD 2.4V 无效
LVD36	LVD 3.6V 有效	LVD 3.6V 无效

- JP3：单片机和 ICE 电源接口。
- JP6/JP8：EV-kit GND 接口。
- JP7/JP9：EV-kit 电源接口。
- JP10：单片机 I/O 引脚接口。
- U1：SN8P2602C 20-pin SSOP 封装形式单片机接口，连接用户目标板。
- U3：SN8P2602C 18-pin 封装形式单片机接口，连接用户目标板。

SN8P2602C EV-Kit 原理图如下：

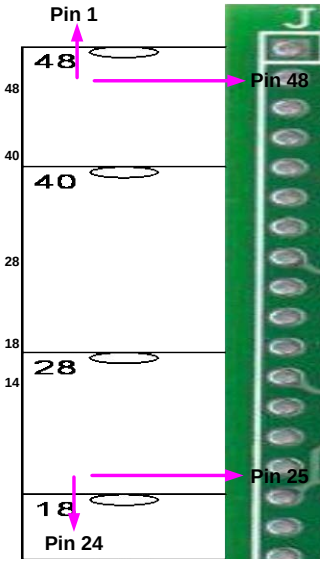


12.2 ICE与EV-KIT应用注意事项

1、SN8P2602C EV-KIT 连接到 SN8ICE2K Plus 2 之前必须关闭 SN8ICE2K Plus 2 的电源。
2、EV-KIT 的 CON1/CON2 连接到 ICE 的 JP3/CON1。
3、开始仿真之前开启 SN8ICE2K Plus 2 的电源。
4、如果电源指示灯（LED D1）不亮，则 EV-Kit 出现不良现象，请联系 SONIX 的代理商处理。
5、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。
6、当 P5_SINK 编译选项选择 40mA 时，EV-Kit 不支持 P5.0~P5.3/P5.5 slew rate 超过 1MIPS 的指令周期，但实际芯片可以。

13 OTP烧录

13.1 烧录转接板引脚配置



JP3 (Mapping to 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

Writer JP1/JP2

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

JP1 连接烧录转接板
JP2 连接 dice 和>48 pin 封装的 IC

13.2 烧录引脚配置

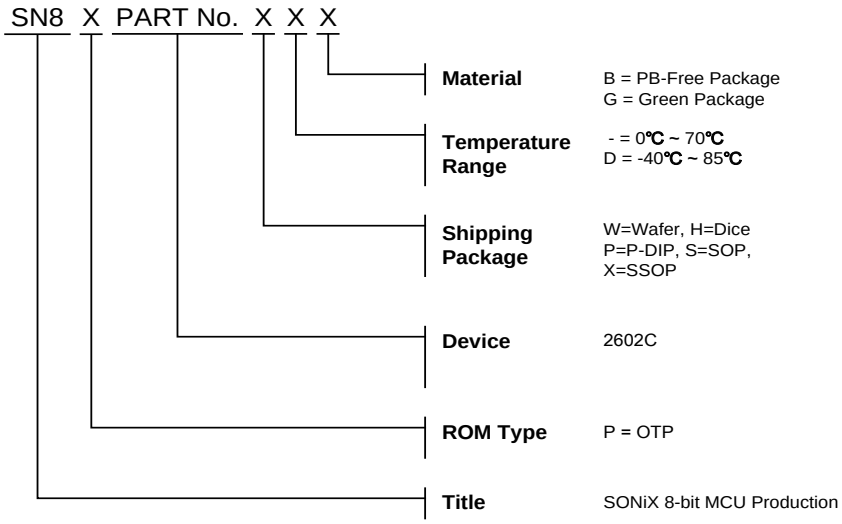
SN8P2602C 烧录引脚信息							
单片机名称		SN8P2602CP/S(DIP/SOP)			SN8P2602CX(SSOP)		
烧录器接口		IC 和 JP3 48-pin text tool 引脚配置					
JP1/JP2 引脚编号	JP1/JP2 引脚名称	IC 引脚编号	IC 引脚说明	JP3 引脚编号	IC 引脚编号	IC 引脚说明	JP3 引脚编号
1	VDD	14	VDD	29	15, 16	VDD	29, 30
2	GND	5	VSS	20	5, 6	VSS	19, 20
3	CLK	6	P5.0	21	7	P5.0	21
4	CE	-	-	-	-	-	-
5	PGM	17	P1.0	32	19	P1.0	33
6	OE	7	P5.1	22	8	P5.1	22
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP	4	RST	19	4	RST	18
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	18	P1.1	33	20	P1.1	34

14 单片机正印命名规则

14.1 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。

14.2 单片机正印说明



14.3 命名举例

- **Wafer, Dice：**

单片机名称	ROM 类型	器件（Device）	封装形式	温度范围	封装材料
S8P2602CW	OTP	2602C	Wafer	-10℃~70℃	-
SN8P2602CH	OTP	2602C	Dice	-10℃~70℃	-

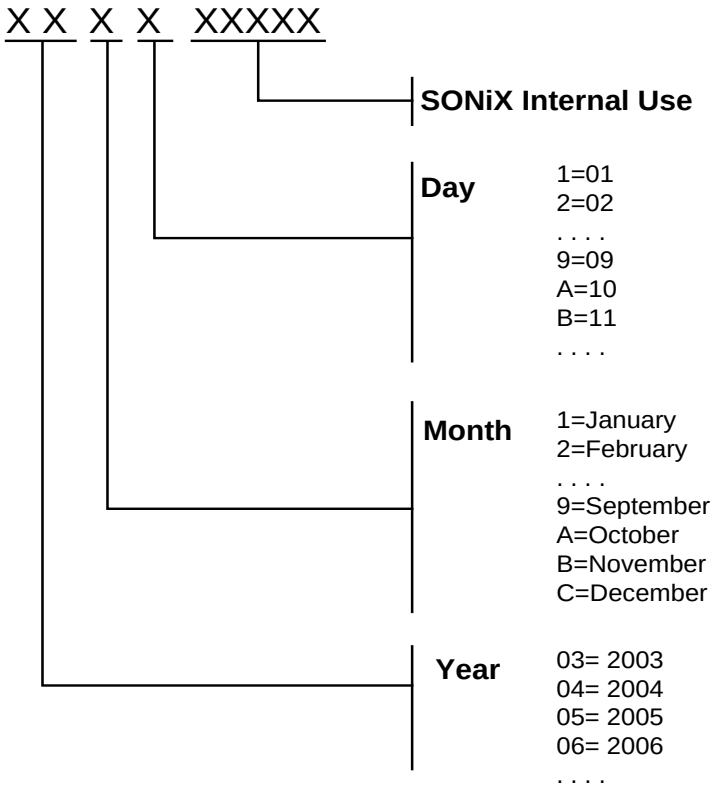
- **绿色封装：**

单片机名称	ROM 类型	器件（Device）	封装形式	温度范围	封装材料
SN8P2602CPG	OTP	2602C	P-DIP	-10℃~70℃	绿色封装
SN8P2602CSG	OTP	2602C	SOP	-10℃~70℃	绿色封装
SN8P2602CXG	OTP	2602C	SSOP	-10℃~70℃	绿色封装
SN8P2602CPDG	OTP	2602C	P-DIP	-40℃~85℃	绿色封装
SN8P2602CSDG	OTP	2602C	SOP	-40℃~85℃	绿色封装
SN8P2602CXDG	OTP	2602C	SSOP	-40℃~85℃	绿色封装

- **无铅封装：**

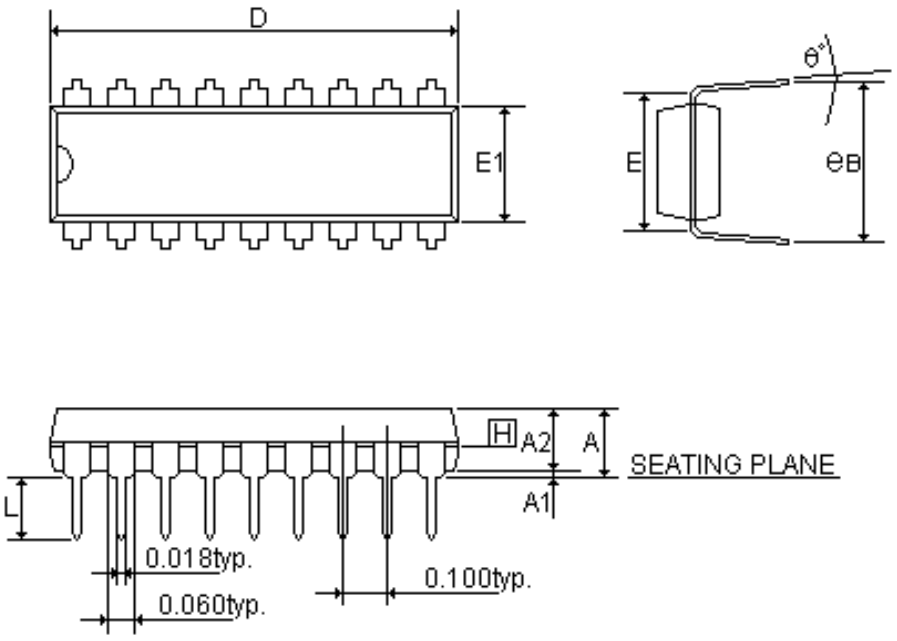
单片机名称	ROM 类型	器件（Device）	封装形式	温度范围	封装材料
SN8P2602CPB	OTP	2602C	P-DIP	-10℃~70℃	无铅封装
SN8P2602CSB	OTP	2602C	SOP	-10℃~70℃	无铅封装
SN8P2602CXB	OTP	2602C	SSOP	-10℃~70℃	无铅封装
SN8P2602CPDB	OTP	2602C	P-DIP	-40℃~85℃	无铅封装
SN8P2602CSDB	OTP	2602C	SOP	-40℃~85℃	无铅封装
SN8P2602CXDB	OTP	2602C	SOP	-40℃~85℃	无铅封装

14.4 日期码规则


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

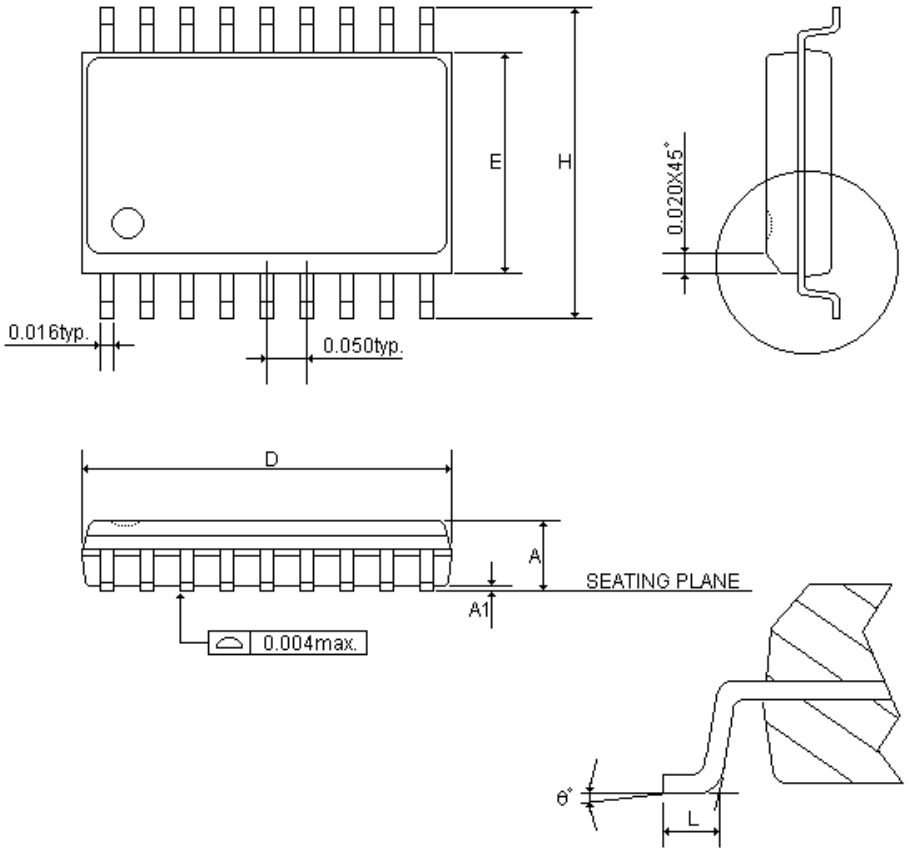
15 封装信息

15.1 P-DIP 18 PIN



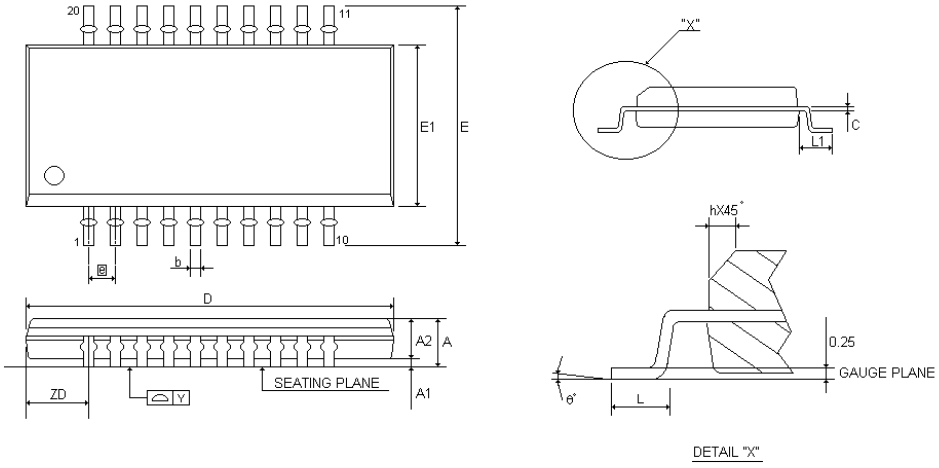
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.880	0.900	0.920	22.352	22.860	23.368
E		0.300			7.620	
E1	0.245	0.250	0.255	6.223	6.350	6.477
L	0.115	0.130	0.150	2.921	3.302	3.810
e B	0.335	0.355	0.375	8.509	9.017	9.525
θ°	0°	7°	15°	0°	7°	15°

15.2 SOP 18 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.093	0.099	0.104	2.362	2.502	2.642
A1	0.004	0.008	0.012	0.102	0.203	0.305
D	0.447	0.455	0.463	11.354	11.557	11.760
E	0.291	0.295	0.299	7.391	7.493	7.595
H	0.394	0.407	0.419	10.008	10.325	10.643
L	0.016	0.033	0.050	0.406	0.838	1.270
θ°	0°	4°	8°	0°	4°	8°

15.3 SSOP 20 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.053	0.063	0.069	1.350	1.600	1.750
A1	0.004	0.006	0.010	0.100	0.150	0.250
A2	-	-	0.059	-	-	1.500
b	0.008	0.010	0.012	0.200	0.254	0.300
c	0.007	0.008	0.010	0.180	0.203	0.250
D	0.337	0.341	0.344	8.560	8.660	8.740
E	0.228	0.236	0.244	5.800	6.000	6.200
E1	0.150	0.154	0.157	3.800	3.900	4.000
[e]		0.025			0.635	
h	0.010	0.017	0.020	0.250	0.420	0.500
L	0.016	0.025	0.050	0.400	0.635	1.270
L1	0.039	0.041	0.043	1.000	1.050	1.100
ZD		0.059			1.500	
Y	-	-	0.004	-	-	0.100
θ°	0°	-	8°	0°	-	8°

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

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