SN8P2714X_2715

用户参考手册

Vesion1.0

SN8P2714
SN8P27142
SN8P27143
SN8P2715

SONiX 8 位单片机

修改记录

版本	日期	说明
VER1.0	2008 年 3 月	初版。
	2008 年 5 月	修改烧录信息章节内容。

目　录

1 产品简介

1.1 SN8P2710 系列的特性

- **存储器配置**
  OTP ROM 空间：2K * 16 位。
  RAM 空间：128 字节。
  8 层堆栈缓存器。

- **I/O 引脚配置**
  单向输入引脚：P0。
  双向输入/输出端口：P2、P4、P5
  具有唤醒功能的引脚：P0.0、P0.1、P0.2。
  外部中断引脚：P0.0、P0.1。
  内置上拉电阻的端口：P0、P2、P4、P5。
  P4 口和 ADC 输入引脚共用。

- **8 通道 12 位 ADC**

- **1 通道 7 位 DAC**

- **强大的指令系统**
  单周期指令系统（1T）。
  绝大部分指令只需要一个周期。
  查表指令 MOVC 可寻址整个 ROM 区。

- **4 个中断源**
  2 个内部中断：TC0，TC1。
  2 个外部中断：INT0，INT1。

- **2 个 8 位定时/计数器**
  TC0：自动装载定时/计数器/PWM0/Buzzer 输出。
  TC1：自动装载定时/计数器/PWM1/Buzzer 输出。

- **内置看门狗定时器**

- **系统时钟模式和操作模式**
  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 16MHz。
  内部低速时钟：RC 模式，16KHz（3V）\ 32KHz（5V）。
  普通模式：高速时钟和低速时钟同时工作。
  低速模式：仅低速时钟运行。
  睡眠模式：高低速时钟均停止运行。

- **封装形式**
  SN8P27142：P-DIP 18 PIN 、SOP 18 PIN。
  SN8P27143：P-DIP 20 PIN、SOP 20 PIN、SSOP 20 PIN。
  SN8P2714：SK-DIP 28 PIN、SOP 28 PIN。
  SN8P2715：P-DIP 32 PIN、SOP 32 PIN。

产品性能比较表

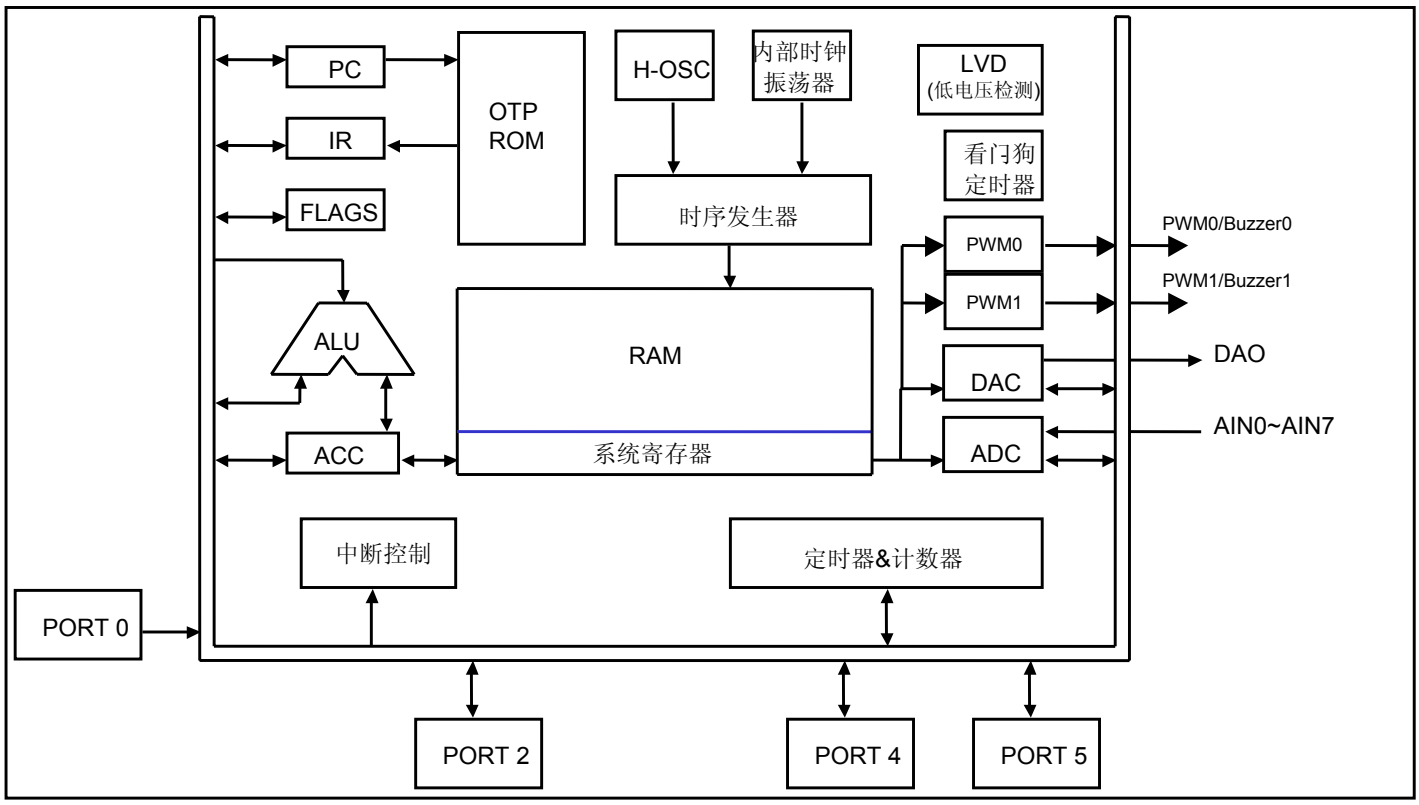
单片机型号	ROM	RAM	堆栈	定时器			I/O	ADC	DAC	PWM	SIO	唤醒功能引脚数目	封装形式
				T0	TC0	TC1				Buzzer			
SN8P27142	2K*16	128	8	-	V	V	15	5ch	-	2	-	2	DIP18/SOP18
SN8P27143	2K*16	128	8	-	V	V	16	6ch	-	2	-	2	DIP20/SOP20/SSOP20
SN8P2714	2K*16	128	8	-	V	V	23	8ch	1ch	2	-	3	SKDIP28/SOP28
SN8P2715	2K*16	128	8	-	V	V	27	8ch	1ch	2	-	3	DIP32/SOP32
SN8P2704A	4K*16	256	8	V	V	V	18	5ch	1ch	2	1	8	SKDIP28/SOP28
SN8P2705A	4K*16	256	8	V	V	V	23	8ch	1ch	2	1	9	DIP32/SOP32

✸ 注：**SN8P27142/SN8P27143 必须置 P02R（P0UR 的第 2 位）为 1 以避免不能进入睡眠模式。**

SN8P2714/15 和 SN8P2704A/05A 的比较表

条目	SN8P2714/15	SN8P2704A/05A
AC 抗干扰性	**很好**（在 VDD 和 GND 之间连接一个 47uF 的旁路电容）	**很好**（在 VDD 和 GND 之间连接一个 47uF 的旁路电容）
存储器	ROM：2K RAM：128	ROM：4K RAM：256
最多 I/O 引脚数	28PIN 的封装：23 个 I/O 32PIN 的封装：27 个 I/O	28PIN 的封装：18 个 I/O 32PIN 的封装：23 个 I/O
高速 PWM	PWM 分辨率：8/6t/5/4 位 8 位 PWM：在 16MHz 时高达 62.5K 4 位 PWM：在 16MHz 高达 1000K	PWM 分辨率：8/6/5/4 位 8 位 PWM：在 16MHz 时高达 7.8125K 4 位 PWM：在 16MHz 高达 125K
可编程开漏输出	N/A	P1.0 / P1.1 / P5.2 (SO)
B0MOV M, I	无限制	“I”不能是 0E6H 或者 0E7H
B0XCH A, M	无限制	“M”的地址不能是 80H~0FFH
ROM 中地址 8 处的限制指令	无限制	JMP 或者 NOP
ADC 中断	无	有
ADC 时钟频率	4 种设置（通过 ADCKS [1:0]设置）	7 种设置（通过 ADCKS[2:0]设置）
TC0C/TC1C/TC0R/TC1R 的有效范围	00H ~ 0FFH	00H ~ 0FFH
绿色模式	无	有
SIO 功能	无	有
LVD	2.0V 常开启	1.8V 常开启
P0	单向输入引脚	双向输入/输出引脚
中断向量的限制指令	无限制	NOP/JMP
PUSH/POP 指令	无	有

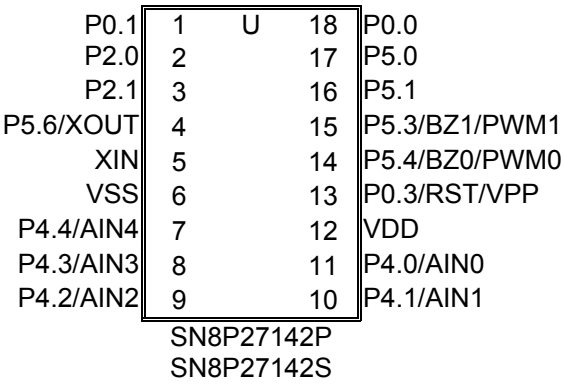
1.2 系统框图


PC
IR
FLAGS
OTP ROM
H-OSC
内部时钟振荡器
LVD (低电压检测)
看门狗定时器
时序发生器
PWM0
PWM1
DAC
ADC
PWM0/Buzzer0
PWM1/Buzzer1
DAO
AIN0~AIN7
ALU
RAM
ACC
系统寄存器
中断控制
定时器&计数器
PORT 0
PORT 2
PORT 4
PORT 5

1.3 引脚配置

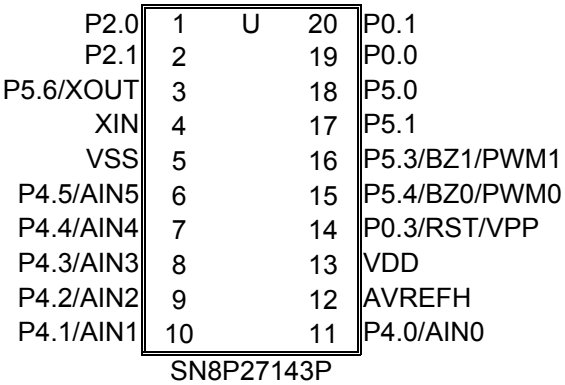
格式说明：SN8P271xY　　Y = K > SK-DIP，P > P-DIP，　S> SOP

1.3.1 SN8P27142 引脚配置



SN8P27142P
SN8P27142S

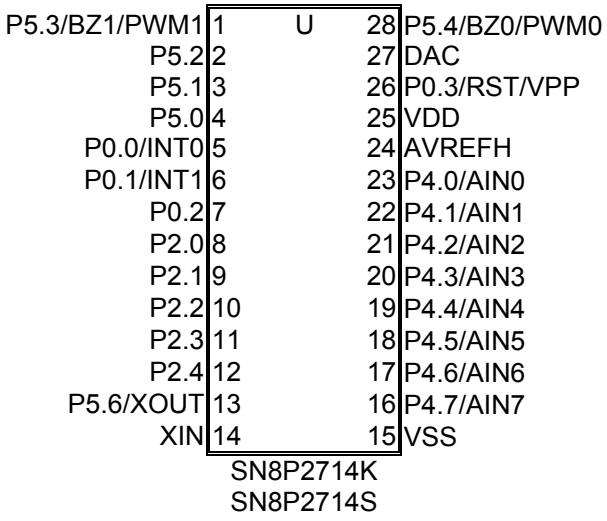
1.3.2 SN8P27143 引脚配置



SN8P27143P
SN8P27143S
SN8P27143X

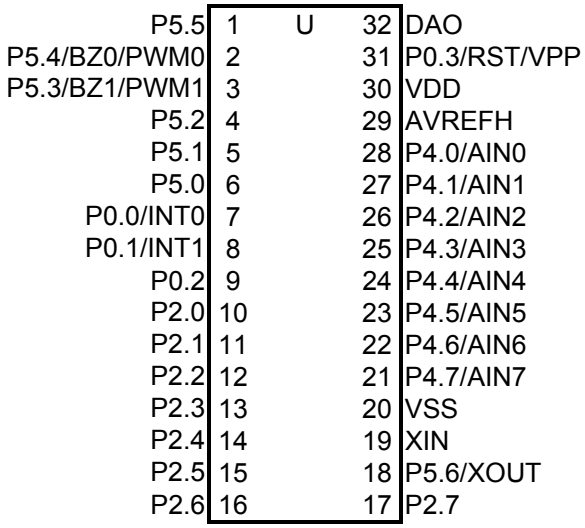
✸ 注：**SN8P27142/SN8P27143** 必须置 **P02R**（**P0UR** 的第 **2** 位）为 **1** 以避免不能进入睡眠模式。
✸ 注：**SN8P27142** 的 **ADC** 参考电压（**AVREFH**）是 **VDD**。

1.3.3 SN8P2714K 引脚配置



SN8P2714K
SN8P2714S

1.3.4 SN8P2715P 引脚配置



SN8P2715P
SN8P2715S

1.4 引脚说明

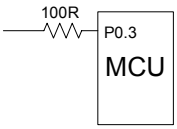
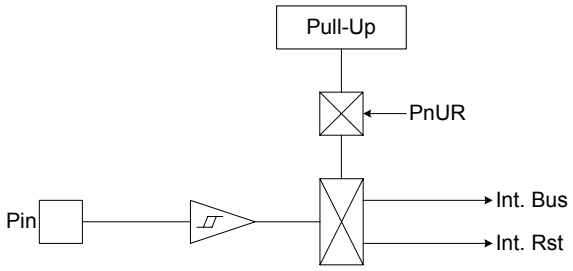
引脚名称	类型	说明
P0 [1:0] / INT [1:0]	I	单向输入引脚，施密特触发，内置上拉电阻，具有唤醒功能。 外部中断触发引脚（施密特触发）。
P0.2	I	单向输入引脚，施密特触发，内置上拉电阻，具有唤醒功能。
P2 [7:0]	I/O	双向输入/输出引脚，输入模式时为施密特触发，内置上拉电阻。
P4 [7:0] / AIN [7:0]	I/O	双向输入/输出引脚，非施密特触发，内置上拉电阻。 ADC 的输入通道。
P5 [5:0]	I/O	双向输入/输出引脚，输入模式时为施密特触发，内置上拉电阻。 P5[4:3]：Buzzer 输出引脚/PWM 输出引脚。
AVREFH	I	ADC 最大参考电压输入端。 （注：SN8P27142/SN8P27143 的 ADC 的参考电压是 VDD。）
DAO	O	DAC 信号输出引脚。
P0.3/RST/VPP	I/P	P0.3：单向输入引脚，施密特触发，**无上拉电阻**，**无唤醒功能**/RC 模式，作普通输入引脚使用时，用户需在单片机的 P0.3 外面串接一个 100 欧姆的电阻（如右图所示，电阻要尽可能的靠近单片机）。 RST：外部复位，低电平有效。 VPP：OTP 烧录引脚。 
XIN	I	振荡信号输入引脚。
XOUT/P5.6	I/O	XOUT：振荡信号输出引脚。 P5.6：双向输入输出引脚，内置上拉电阻，RC 模式时为施密特触发。
VDD, VSS	P	电源输入端。

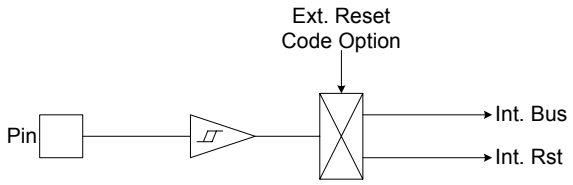
表 1-1. SN8P271x 引脚说明

1.5 引脚电路结构图

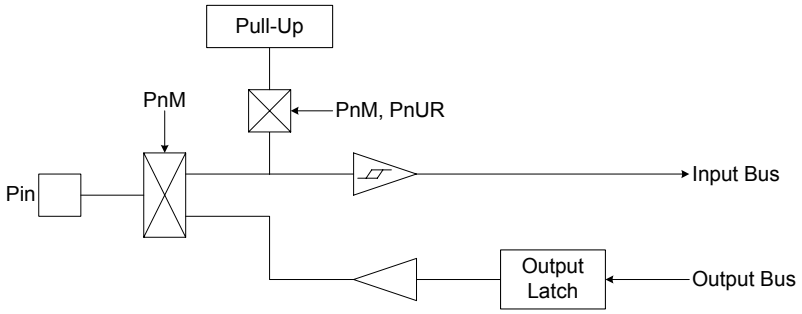
P0.1、P0.2 结构图：



P0.3 结构图：



P2.5 结构图：



P4 结构图：

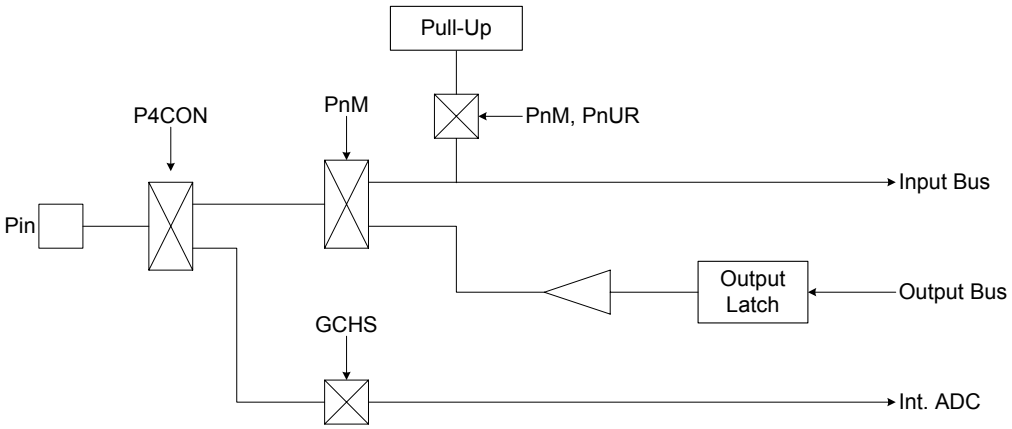


图 1-1. 引脚电路结构图

2 编译选项表（Code Option）

编译选项	内容	功能说明
High_Clk	Ext_RC	外部高速时钟振荡器采用廉价的 RC 振荡电路，XOUT 是 Fcpu 的输出引脚。
	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~16MHz）。
	4M_X'tal	外部高速时钟振荡器采用标准晶体/陶瓷振荡器（如 4M~10MHz）。
Noise_Filter	Enable	开启杂讯滤波功能以提高抗干扰性。
	Disable	关闭杂讯滤波功能。
Watch_Dog	Always_On	始终开启看门狗定时器，即使在睡眠模式下也处于开启状态。
	Enable	开启看门狗定时器，但在睡眠模式下关闭看门狗定时器。
	Disable	关闭看门狗定时器。
Fcpu	Fosc/1	指令周期 = 1 个时钟周期。
	Fosc/2	指令周期 = 2 个时钟周期。
	Fosc/4	指令周期 = 4 个时钟周期。
	Fosc/8	指令周期 = 8 个时钟周期。
Security	Enable	ROM 代码加密。
	Disable	ROM 代码不加密。
RST_P0.3	Reset	使能外部复位引脚。
	P0.3	P0.3 为单向输入引脚，无上拉电阻。
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。

表 2-1. SN8P271X 的编译选项表（Code Option）

✷ 注：

1. **在高干扰环境下，强烈建议开启杂讯滤波功能，并将“Watch_Dog”设置为“Always ON”；**
2. **开启杂讯滤波功能后会限制 Fcpu = Fosc/4 或者 Fosc/8；**
3. **Fcpu 编译选项仅对高速时钟有效；**
4. **普通模式下，Fosc = Fhosc（外部高速时钟）；**
5. **低速模式下，Fosc = Flosc（内部低速 RC 时钟）；**
6. **低速模式下，Fcpu = Fosc / 4。**

3 存储器配置

3.1 程序存储器（ROM）

3.1.1 概述

SN8P2710 的程序存储器 OTP ROM 的容量为 2K*16 位，可由 12 位程序计数器 PC 对 ROM 进行寻址访问或者由专用寄存器 R、X、Y、Z 对 ROM 进行查表访问。所有 2048*16 位的 ROM 通常分为 4 个区域。

- 复位向量
- 中断向量
- 保留区域
- 通用存储区

所有的 ROM 分为 3 个代码区：00H~07H 是复位向量区；0008H 用作中断向量区；0009H~07FBH 用作用户代码区。

图 3-1 ROM 区域划分

3.1.2 复位向量（0000H）

上电复位和看门狗溢出后，系统复位。所有的寄存器都恢复成默认值。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           ; 跳转到用户程序地址
        .                       ;
        ORG     10H
START:                          ; 用户程序开始
        .                       ; 用户程序
        .
        .
        ENDP                    ; 程序结束
```

3.1.3 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。

➢ **例：定义中断向量，主程序位于中断服务程序后面。**

```
                ORG     0               ;
                JMP     START           ; 跳转到用户程序。
                .
                ORG     8H              ; 中断服务程序。
                B0XCH   A, ACCBUF       ;
                B0MOV   A, PFLAG
                B0MOV   PFLAGBUF, A     ; 保存 PFLAG。
                .                       ; 用户程序。
                B0MOV   A, PFLAGBUF
                B0MOV   PFLAG, A        ; 恢复 PFLAG。
                B0XCH   A, ACCBUF
                RETI                    ; 中断返回。
START:                                  ; 用户程序开始。
                .                       ; 用户程序。
                JMP     START           ;

                ENDP                    ; 程序结束。
```

➢ **例：定义中断向量，中断服务程序位于主程序后面。**

```
                ORG     0               ;
                JMP     START           ; 跳转到用户程序。
                .
                ORG     08H
                JMP     MY_IRQ          ; 跳转到中断服务程序。

                ORG     10H
START:                                  ; 用户程序开始。
                .                       ; 用户程序。
                JMP     START           ;
MY_IRQ:                                 ; 中断服务程序开始。
                B0XCH   A, ACCBUF       ;
                B0MOV   A, PFLAG
                B0MOV   PFLAGBUF, A     ; 保存 PFLAG。
                .                       ; 用户程序。
                B0MOV   A, PFLAGBUF
                B0MOV   PFLAG, A        ; 恢复 PFLAG。
                RETI                    ; 中断返回。

                ENDP                    ; 程序结束。
```

✸ **注：从上面的程序中很容易得出 SONiX 的编程规则，有以下几点：**

1.地址 0000H 处的“JMP”指令使程序从头开始执行。

2. 中断服务程序从 0008H 开始，用户可以将中断程序的入口地址放在 0008H（如例 1），也可以在 0008H 处写入一个跳转指令（例 2），从而将整个中断程序放在通用 ROM 区，形成模块化编程风格。

3.1.4 通用存储区

ROM 中的 0001H~0007H 和 0009H~07FBH 用作通用存储区，主要用来存储指令操作代码和查表数据。SN8P2710 包括通过程序计数器 PC 实现的跳转表功能和通过寄存器 R、Y、Z 实现的查表功能。

在跳转表程序和查表程序中，程序计数器不会自动跳过边界，因此当 PCL 溢出（0FFFH 到 000H）时，用户必须自行将 PCH 调整为 PCH+1。

3.1.5 查表

对 ROM 数据进行查找，寄存器 Y 指向查找数据地址的高字节（bit8~bit15），寄存器 Z 指向地址的低字节（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：查表程序。**

```
                B0MOV     Y, #TABLE1$M    ; 设置 table1 的高位字节地址。
                B0MOV     Z, #TABLE1$L    ; 设置 table1 的低位字节地址。
                MOVC                      ; 查表，R = 00H，ACC = 35H。
                                          ; 查找下一地址
                INCMS     Z               ;
                JMP       @F              ; Z 没有溢出
                INCMS     Y               ; Z 溢出（0FFH → 00H），Y=Y+1。
                NOP                       ;
                                          ;
@@:             MOVC                      ; 查表，R = 51H，ACC = 05H。
                …                         ;
TABLE1:         DW        0035H           ; 定义数据表数据（16 位）。
                DW        5105H
                DW        2012H
                …
```

✸ 注：当寄存器 Z 溢出（从 0FFH 变成 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，下面的宏指令 INC_YZ 能够对 Y 和 Z 寄存器自动处理。

✸ 注：由于程序计数器 PC 只有 12 位，X 寄存器在查表的过程中时可以忽略的，用户可以省略“B0MOV X, #TABLE1$H”。SONiX ICE 支持更大的程序寻址能力，所以必须保证 X 寄存器为 0，以避免在查表中出现不可预知的错误。

➢ **例：宏指令 INC_YZ。**

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

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

下面的程序通过累加器对 Y、Z 寄存器进行处理。

➢ **例：B0ADD/ADD 对 Y 和 Z 寄存器加 1。**

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

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

                B0BTS1    FC              ; 检查进位标志。
                JMP       GETDATA         ; FC = 0。
                INCMS     Y               ; FC = 1，Y+1。
                NOP
GETDATA:                                  ;
                MOVC                      ; 查表，如果 BUF = 0。数据为 35H。
                                          ; 如果 BUF = 1，数据为 5105H。
                                          ; 如果 BUF = 2，数据为 2012H。
                …
TABLE1:         DW        0035H           ; 定义数据表数据（16 位）。
                DW        5105H
                DW        2012H
                …
```

3.1.6 跳转表

跳转表能够实现多地址跳转功能。PCL 和 ACC 的值相加即可得到新的 PCL。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，用户就可以轻松实现多地址的跳转。

执行“ADD PCL, A”后，如果有进位发生，结果并不会影响 PCH 寄存器。用户必须检查跳转表是否跨越了 ROM 的页边界。如果跳转表跨越了 ROM 页边界（例如从 xxFFH 到 xx00H），将跳转表移动到下一页程序存储区的顶部（xx00H）。

注：一页包含 256 字。

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

```
        ORG     100H            ; 跳转表从 ROM 前端开始。

        B0ADD   PCL, A          ; PCL = PCL + ACC，PCH 的值不会被改变。
        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。
```

在下面的例子中，跳转表从 0FDH 开始，执行 B0ADD PCL, A 后，如果 ACC = 0 或者 1，跳转表指向正确的地址，但如果 ACC 大于 1，因为 PCH 不能自动增量，程序就会出错。可以看到当 ACC = 2 时，PCL = 0，而 PCH 仍然保持为 0，则新的程序计数器 PC 将指向错误的地址 0000H，程序失效。因此，检查跳转表是否跨越边界（xxFFH 到 xx00H）非常重要。良好的编程风格是将跳转表放在 ROM 的开始边界（如 0100H）。

➢ **例：如果跳转表跨越 ROM 边界，将引起程序错误。**

```
ROM 地址
  …
  …
00FDH   B0ADD   PCL, A          ; PCL = PCL + ACC，PCH 的值不会被改变。
00FEH   JMP     A0POINT         ; ACC = 0。
00FFH   JMP     A1POINT         ; ACC = 1。
0100H   JMP     A2POINT         ; ACC = 2  ← 跳转表跨越边界。
0101H   JMP     A3POINT         ; ACC = 3。
  …
```

SONiX 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。但采用该宏会占用部分 ROM 空间。注意跳转表的最大数量低于 254。

```
@JMP_A  MACRO   VAL
        IF      (($+1) !& 0FF00H) !!= (($+(VAL)) !& 0FF00H)
        JMP     ($ | 0FFH)
        ORG     ($ | 0FFH)
        ENDIF
        ADD     PCL, A
        ENDM
```

注：“VAL”为跳转表列表中列表个数。

➢ **例：宏“MACRO3.H”中，“@JMP_A”的应用。**

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

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

3.2 数据存储器（RAM）

3.2.1 概述

SN8P2710 有 128 字节的通用数据存储区。

SN8P2710

- **128 字节的通用存储器（bank 0）。**
- **128 字节的系统专用寄存器。**

RAM 位于 Bank0，前 128 字节用作通用存储区，后 128 字节用作系统寄存器。

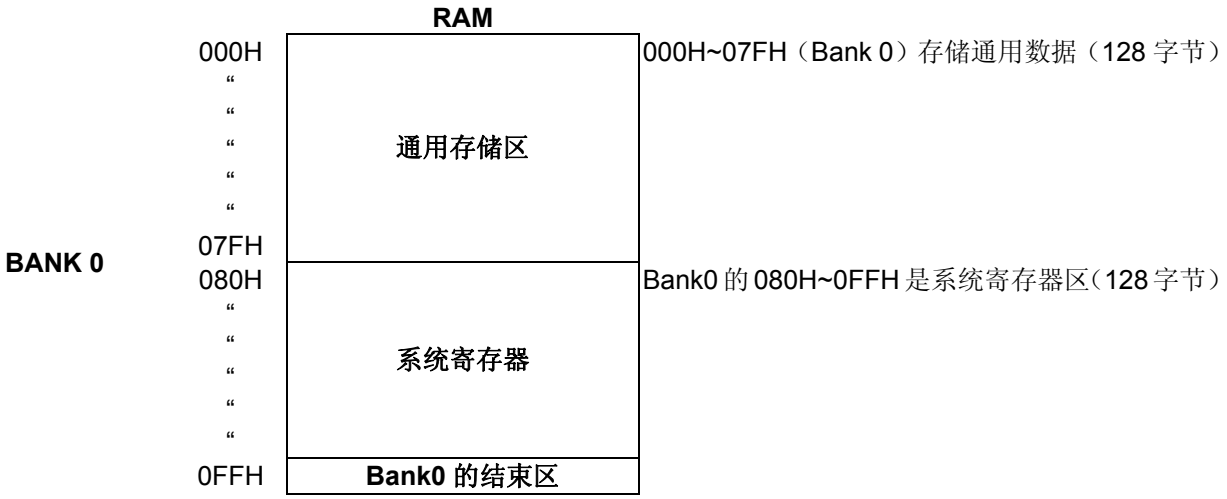


图 3-2 SN8P2710 的 RAM 区域划分

✸ **注：执行读指令“MOV A, M”后，系统寄存器中未定义的位置逻辑“高”。**

3.3 工作寄存器

RAM 中的 82H~84H 是系统专用寄存器，如 R、Y、Z 寄存器，如下表所示。这些寄存器可用作一般的工作寄存器或者用来访问 ROM 和 RAM 中的数据。例如，通过 R、Y、Z 寄存器可以在整个 ROM 区执行查表指令，也可以由 Y、Z 寄存器间接寻址 RAM。

	80H	81H	82H	83H	84H	85H	
RAM	-	-	R	Z	Y	-	
	-	-	R/W	R/W	R/W	-	

3.3.1 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。

Y = XXXX XXXX

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
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Z = XXXX XXXX

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
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

间接寻址寄存器@YZ 位于 RAM 中的 E7H，通过 Y、Z 寄存器访问 RAM 中的数据，ACC 对该数据进行读/写。Y 寄存器的低 4 位确定数据在 RAM 的位置，Z 寄存器则确定数据的具体地址，Y 寄存器的高 4 位在间接寻址中没有实际意义。

➢ **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。**

```
        MOV     A, #0
        B0MOV   Y, A        ; Y = 0，指向 bank 0。
        MOV     A, #07FH
        B0MOV   Z, A        ; Z = 7FH，RAM 区的最后单元。
CLR_YZ_BUF:
        CLR     @YZ         ; @YZ 清零。

        DECMS   Z           ; Z – 1，若 Z= 0，则程序结束。
        JMP     CLR_YZ_BUF  ;

        CLR     @YZ
END_CLR:                    ;
        .
```

✸ **注：关于 Y、Z 寄存器的查表功能请参阅“查表”章节。**

3.3.2 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，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。

R = XXXX XXXX

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
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

✸ **注：关于 R 寄存器的查表功能请参阅“查表”章节。**

3.4 程序状态寄存器（PFLAG）

寄存器 PFLAG 包括复位状态标志、低电压检测标志、进位标志、辅助进位标志（DC）和零标志（Z）。如果运算结果为 0 或者有进位、借位发生，将会影响到 PFLAG 寄存器。

PFLAG 初始值 = 00xx,x000

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
	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W

3.4.1 复位状态标志

NT0	NPD	复位状态
0	0	看门狗溢出
0	1	保留
1	0	LVD 复位
1	1	外部复位引脚复位

3.4.2 LVD 2.4V FLAG

LVD24	VDD 状态
1	VDD <= 2.4V
0	VDD > 2.4V

✸ **注：该位仅在 LVD = LVD_M 时有效。**

3.4.3 LVD 3.6V FLAG

LVD36	VDD 状态
1	VDD <= 3.6V
0	VDD > 3.6V

✸ **注：该位仅在 LVD = LVD_H 时有效。**

3.4.4 进位标志

C = 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。
C = 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。

3.4.5 辅助进位标志

DC = 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。
DC = 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。

3.4.6 零标志

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

3.5 累加器 ACC

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

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

➢ **例：读/写 ACC。**

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

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

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

系统执行中断操作时，ACC 的数据不会自动存储，用户需通过程序将中断入口处的 ACC 的数据送入存储器进行保存。

➢ **例：ACC 和工作寄存器中断保护。**

```
ACCBUF          EQU     00H             ;

INT_SERVICE:
                B0XCH   A, ACCBUF       ;
                B0MOV   A, PFLAG
                B0MOV   PFLAGBUF, A     ; 保存 PFLAG。
                .
                .
                B0MOV   A, PFLAGBUF
                B0MOV   PFLAG, A        ; 恢复 PFLAG。
                B0XCH   A, ACCBUF       ;

                RETI                    ; 中断返回。
```

✷ **注：必须使用“B0XCH”指令进行 ACC 数据的中断恢复，否则 PFLAG 会被更改而导致出错。**

3.6 堆栈

3.6.1 概述

SN8P2710 共 8 层堆栈，每层共 11 位。堆栈缓存器 STKnH 和 STKnL 在中断保护和恢复时用来来存放程序计数器 PC 的数据。堆栈指针 STKP 指向当前栈顶位置。

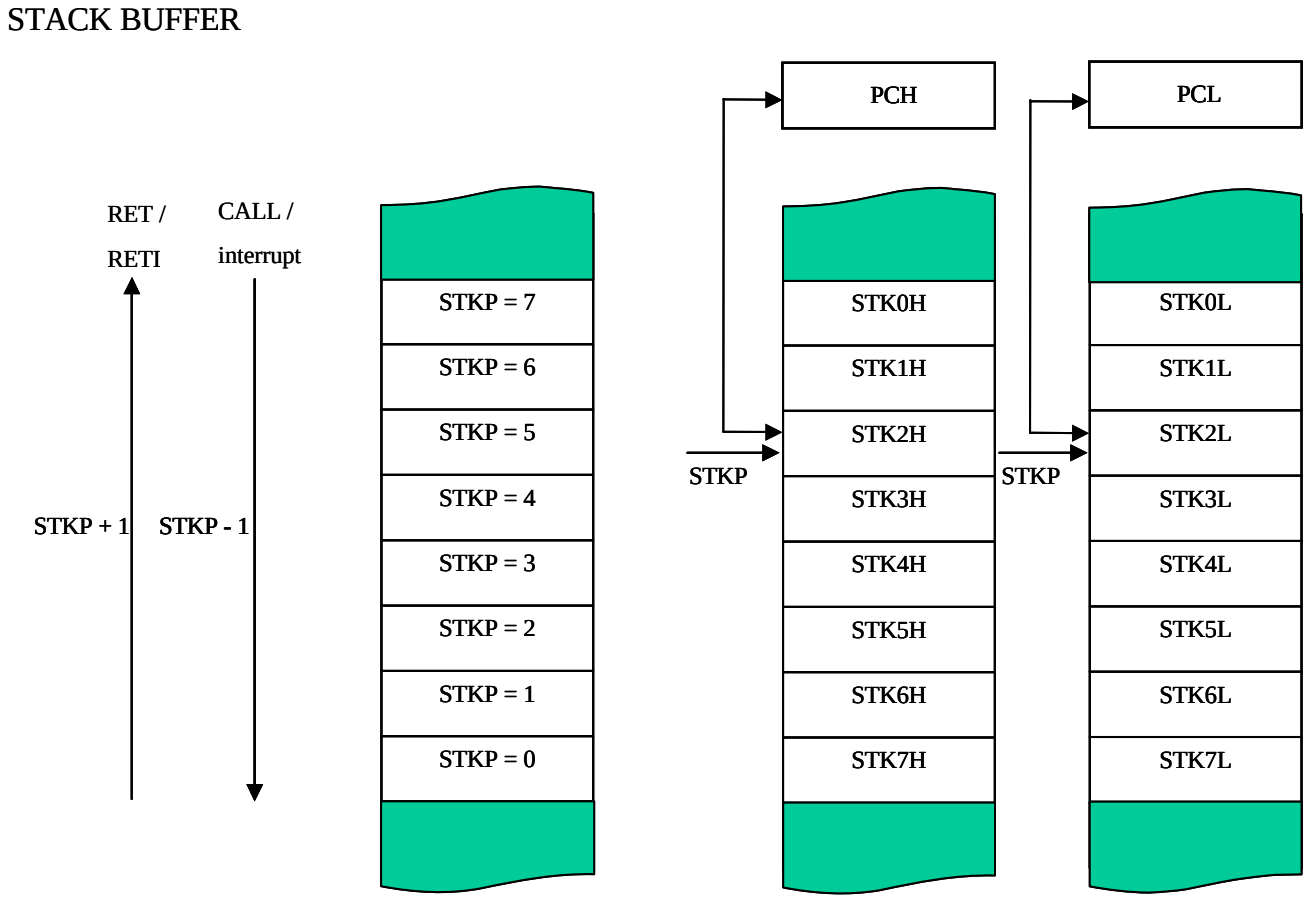


图 3-3 堆栈

3.6.2 堆栈寄存器

3 位寄存器堆栈指针（STKP）用来存放被访问的堆栈地址，11 位数据存储器（STKnH 和 STKnL）则用来暂存堆栈数据。

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

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

STKP 初始值 = 0xxx x111

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

STKPBn：堆栈指针。（n = 0 ~ 2）
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
```

STKn 初始值 = xxxx xxxx xxxx xxxx，STKn = STKnH + STKnL（n = 7 ~ 0）

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

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
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

STKnH：执行中断或 CALL 指令时存储 PCH 的值。n = 0 ~2。
STKnL：执行中断或 CALL 指令时存储 PCL 的值。n = 0 ~7。

3.6.3 堆栈操作举例

执行程序调用指令 CALL 和响应中断复位时，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。入栈操作如下表所示：

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

表 3-1. STKP, STKnH 和 STKnL 在堆栈保护中的关系

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

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

表 3-2. STKP, STKnH 和 STKnL 在堆栈恢复中的关系

3.7 程序计数器 PC

11 位程序计数器 PC 分为高 3 位和低 8 位，专门用来存放下一条需要执行指令的内存地址。通常，程序计数器会随程序中指令的执行自动增加。

若程序执行 CALL 和 JMP 指令时，PC 指向特定的地址。

PC 初始值 = xxxx x000 0000 0000

	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	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

PCH 初始值 = xxxx x000

0CFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PCH	-	-	-	-	-	PC10	PC9	PC8
	-	-	-	-	-	R/W	R/W	R/W

PCL 初始值 = 0000 0000

0CEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PCL	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

3.7.1 单地址跳转

SN8P2710 系列共有 7 条单地址跳转指令：CMPRS、INCS、INCMS、DECS、DECMS、B0BTS0 和 B0BTS1，如果这些指令执行的结果为真，PC 值加 2 跳过下一条指令。

如果位测试为真，PC 加 2，跳过下一条指令：

```
                B0BTS1      FC          ; C = 1，跳过下一条指令。
                JMP         C0STEP      ; 否则跳到 C0STEP。
                .
C0STEP:         NOP

                B0MOV       A, BUF0     ;
                B0BTS0      FZ          ; Z = 0，跳过下一条指令。
                JMP         C1STEP      ; 否则跳到 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     ; ACC = 12H，跳过下一条指令。
                JMP         C0STEP      ; 否则跳到 C0STEP。
                .
C0STEP:         NOP
```

执行加 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

INCS：

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

INCMS：

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

执行减 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

DECS：

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

DECMS：

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

3.7.2 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 PCL, A”若有进位，并不会影响 PCH。

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

```
; PC = 0323H
                    MOV         A, #28H
                    B0MOV       PCL, A          ; 跳转到 0328H。
                    .           .
                    .           .
; PC = 0328H        .           .
                    MOV         A, #00H
                    B0MOV       PCL, A          ; 跳转到 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，跳转到 A0POINT。
                    JMP         A1POINT         ; ACC = 1，跳转到 A1POINT。
                    JMP         A2POINT         ; ACC = 2，跳转到 A2POINT。
                    JMP         A3POINT         ; ACC = 3，跳转到 A3POINT。
                    .           .               ;
```

4 寻址模式

4.1 概述

SN8P2710 提供了 3 种 RAM 寻址方式：立即寻址、直接寻址和间接寻址。

4.1.1 立即寻址

立即寻址就是将立即数直接送入 ACC 或指定的 RAM 单元。如 MOV A, #I；B0MOV M, #I。

立即寻址模式

```
MOV     A, #12H     ; 立即数 12H 送入 ACC。
```

4.1.2 直接寻址

直接寻址就是通过 ACC 对 RAM 单元数据进行操作。如 MOV A,12H，MOV 12H,A。

直接寻址模式

```
B0MOV   A, 12H      ; 地址 12H 处的内容送入 ACC。
```

4.1.3 间接寻址

间接寻址就是通过数据指针（H/L、Y/Z）对数据存储单元进行读写。如 MOV A,@YZ，MOV @YZ,A。

- **例：间接寻址。**

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

4.1.4 寻址 RAM BANK 0

RAM bank0 的单元都可以通过下面两种寻址方式读/写。

- **例 1：直接寻址（如 B0XXX 指令）。**

```
B0MOV   A, 12H      ; 地址 12H 处的内容送入 ACC。
```

- **例 2：@YZ 间接寻址。**

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

5 系统寄存器

5.1 概述

RAM 的 80H~0FFH 用作系统专用寄存器，用来控制单片机的内部硬件资源，如 I/O 状态、ADC、PWM、定时器和计数器等。下面的系统专用寄存器地址分配表为编写应用程序提供了方便快捷的参考。

5.2 系统寄存器配置（BANK 0）

5.2.1 系统寄存器列表

SN8P2710

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	-	-	R	Z	Y	-	PFLAG	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P4CON	-
B	DAM	ADM	ADB	ADR	-	-	-	-	-	-	-	-	-	-	-	PEDGE
C	-	-	P2M	-	P4M	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	-	P2	-	P4	P5	-	-	T0M	-	TC0M	TC0C	TC1M	TC1C	TC1R	STKP
E	P0UR	-	P2UR	-	P4UR	P5UR	-	@YZ	-	-	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

表 5-1. SN8P2710 系统寄存器的地址分配表

说明

PFLAG = ROM 页，特殊标志寄存器
DAM = DAC 模式寄存器
ADB = ADC 数据寄存器
PnM = Pn I/O 模式缓存器
INTRQ = 中断请求寄存器
OSCM = 振荡器寄存器
T0M = TC0、TC1 的速度选择寄存器
TC1M = TC1 模式寄存器
TC1C = TC1 计数寄存器
STKP = 堆栈指针寄存器
@HL = 间接寻址寄存器
P4CON = P4 配置寄存器

R = 工作寄存器和 ROM 查表数据寄存器
Y, Z = 专用寄存器，@YZ 间接寻址寄存器，ROM 寻址寄存器
ADM = ADC 模式寄存器
ADR = ADC 精度选择寄存器
Pn = Pn 数据寄存器
INTEN = 中断使能寄存器
PCH, PCL = 程序计数器
TC0M = TC0 模式寄存器
TC0C = TC0 计数寄存器
TC0R = TC0 自动装载数据缓存器
TC1R = TC1 自动装载数据缓存器
STK0~STK7 = 堆栈缓存器
@YZ = 间接寻址寄存器

✸ 注：

a). 所有的寄存器名称已经在 SN8ASM 编译器中做了宣告；
b). 寄存器中各位的名称已经在 SN8ASM 编译器中以“F”为前缀定义过；
c). 用指令检查空单元时，返回逻辑“高”；
d). 寄存器 ADR 的低字节为只读寄存器；
e). 指令“b0bset”，“b0bclr”，“bset”，“bclr”只能用于可读写的寄存器（“R/W”）。

5.2.2 系统寄存器的位定义

地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	备注
080H										
081H										
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
085H										
086H	NT0	NPD	LVD36	LVD24	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	-	-	-	-	-	-
0AEH	P4CON7	P4CON6	P4CON 5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	W	P4CON
0B0H	DAENB	DAB6	DAB5	DAB4	DAB3	DAB2	DAB1	DAB0	R/W	DAM 数据寄存器
0B1H	ADENB	ADS	EOC	GCHS	-	CHS2	CHS1	CHS0	R/W	ADM 模式寄存器
0B2H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R	ADB 数据缓存器
0B3H	-	ADCKS1	-	ADCKS0	ADB3	ADB2	ADB1	ADB0	R/W	ADR 寄存器
0B4H	-	-	-	-	-	-	-	-	-	
0B5H	-	-	-	-	-	-	-	-	-	
0B6H	-	-	-	-	-	-	-	-	-	
0B8H	-	-	-	-	-	-	-	-	-	
0BFH	-	-	-	P00G1	P00G0	-	-	-	R/W	PEDGE
0C0H	-	-	-	-	-	-	-	-	-	
0C1H	-	-	-	-	-	-	-	-	-	
0C2H	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M I/O 模式控制寄存器
0C3H	-	-	-	-	-	-	-	-	-	
0C4H	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4M I/O 模式控制寄存器
0C5H	-	P56M	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5M I/O 模式控制寄存器
0C8H	-	TC1IRQ	TC0IRQ	-	-	-	P01IRQ	P00IRQ	R/W	INTRQ
0C9H	-	TC1IEN	TC0IEN	-	-	-	P01IEN	P00IEN	R/W	INTEN
0CAH	-	-	-	-	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	-	-	-	-	-	PC10	PC9	PC8	R/W	PCH
0D0H	-	-	-	-	P03	P02	P01	P00	R	P0 数据缓存器
0D1H	-	-	-	-	-	-	-	-	-	
0D2H	P27	P26	P25	P24	P23	P22	P21	P20	R/W	P2 数据缓存器
0D3H	-	-	-	-	-	-	-	-	-	
0D4H	P47	P46	P45	P44	P43	P42	P41	P40	R/W	P4 数据缓存器
0D5H	-	P56	P55	P54	P53	P52	P51	P50	R/W	P5 数据缓存器
0D8H	-	-	-	-	TC1X8	TC0X8	-	-	R/W	T0M
0D9H	-	-	-	-	-	-	-	-	-	
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 堆栈指针
0E0H	-	-	-	-	-	P02R	P01R	P00R	W	P0UR
0E1H	-	-	-	-	-	-	-	-	-	
0E2H	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R	W	P2UR
0E3H	-	-	-	-	-	-	-	-	-	
0E4H	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R	W	P4UR
0E5H	-	P56R	P54R	P54R	P53R	P52R	P51R	P50R	W	P5UR
0E6H	-	-	-	-	-	-	-	-	-	
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ 间接寻址寄存器
0E9H	-	-	-	-	-	-	-	-	-	
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H	-	-	-	-	-	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H	-	-	-	-	-	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H	-	-	-	-	-	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H	-	-	-	-	-	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H	-	-	-	-	-	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH	-	-	-	-	-	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH	-	-	-	-	-	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH	-	-	-	-	-	S0PC10	S0PC9	S0PC8	R/W	STK0H

✸ 注：a). 所有的寄存器名称已经在 **SN8ASM** 编译器中做了宣告；
b). 寄存器中各位的名称已经在 **SN8ASM** 编译器中以“**F**”为前缀定义过；
c). 指令“**b0bset**”，“**b0bclr**”，“**bset**”，“**bclr**”只能用于可读写的寄存器（“**R/W**”）；
d). 详见“系统寄存器的快速参照表”。

6 复位

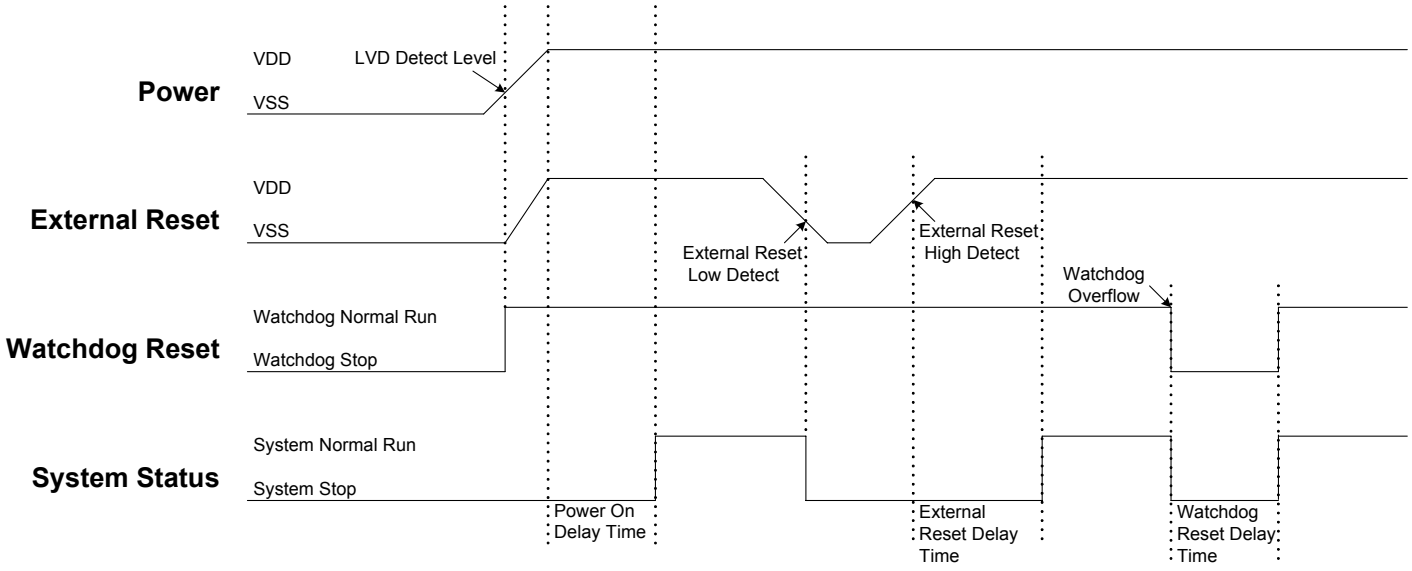
6.1 概述

SN8P271X 系列有以下 4 种复位方式：

- **上电复位；**
- **看门狗复位；**
- **掉电复位；**
- **外部复位（仅支持外部复位引脚使能时）**。

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。

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。



6.2 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到电压稳定；
- **外部复位：**系统检查外部复位引脚的状态（仅当外部复位有效时）。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；
- **振荡器起振：**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。

6.3 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：

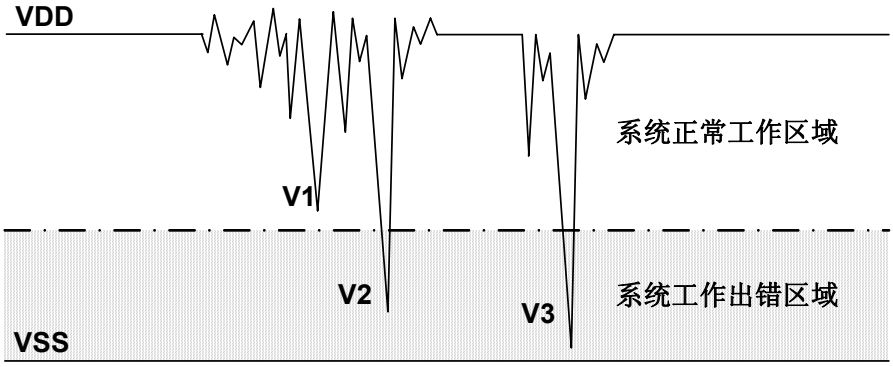
- 对看门狗清零之前，检查 I/O 口的状态和 RAM 的内容可增强程序的可靠性；
- 不能在中断中对看门狗清零，否则无法侦测到主程序跑飞的状况；
- 程序中应该只在主程序中有一次清看门狗的动作，这种架构能够最大限度的发挥看门狗的保护功能。

✷ **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。**

6.4 掉电复位

6.4.1 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 应用中：

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系统复位电压，因此系统维持在死区。

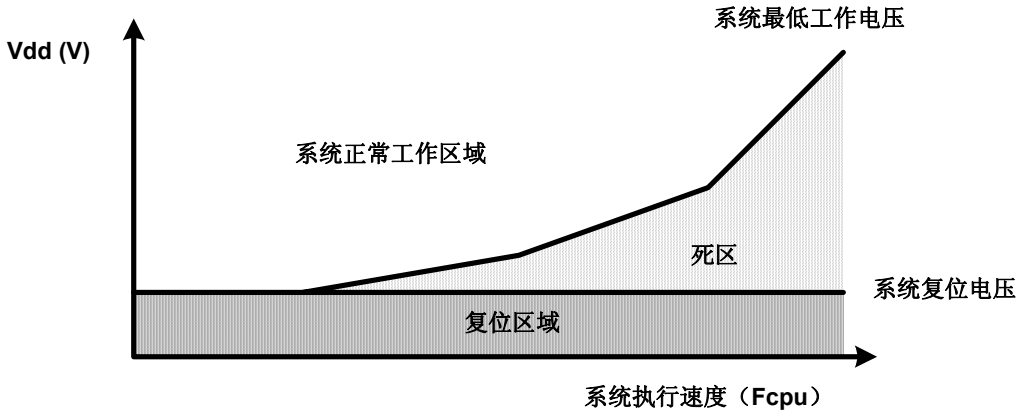
AC 应用中:

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。

6.4.2 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。电气特性一章给出了系统工作电压与执行速度之间的关系。



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。复位电压与最低工作电压之间的区域即是系统工作的死区。

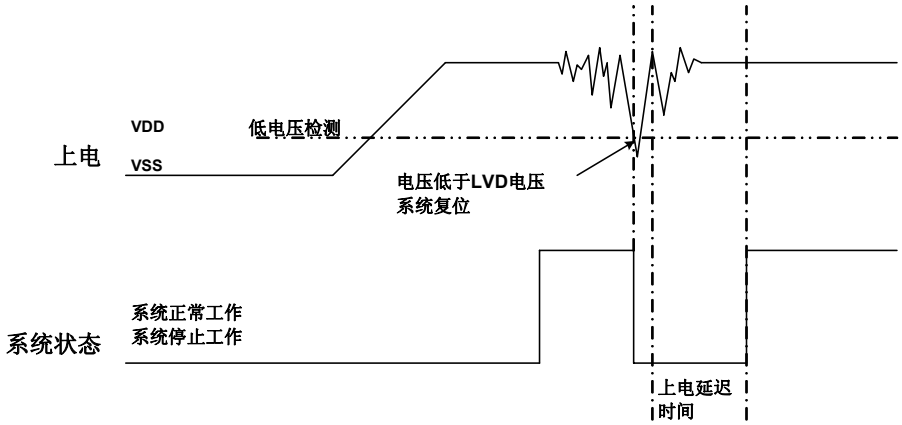
6.4.3 掉电复位性能改进

如何改善系统掉电复位性能，有以下几点建议：

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

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

LVD 复位：



LVD（低电压检测）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 有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，那么 LVD 就不能起到保护作用，就需要采用其它复位方法。电气特性一章中给出了更多关于 LVD 的详细内容。

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。

降低系统运行速度：

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。

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。

6.5 外部复位

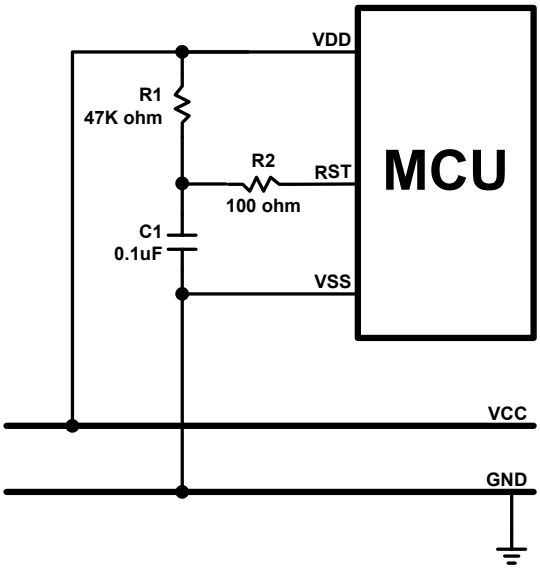
外部复位功能由编译选项“Reset_Pin”控制。将该编译选项置为“Reset”，可使能外部复位功能。外部复位引脚为施密特触发，低电平有效。复位引脚处于高电平时，系统正常运行。当复位引脚输入低电平信号时，系统复位。外部复位操作在上电和正常工作模式时有效。需要注意的是，在系统上电完成后，外部复位引脚必须输入高电平，否则系统将一直保持在复位状态。外部复位的时序如下：

- **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。

6.6 外部复位电路

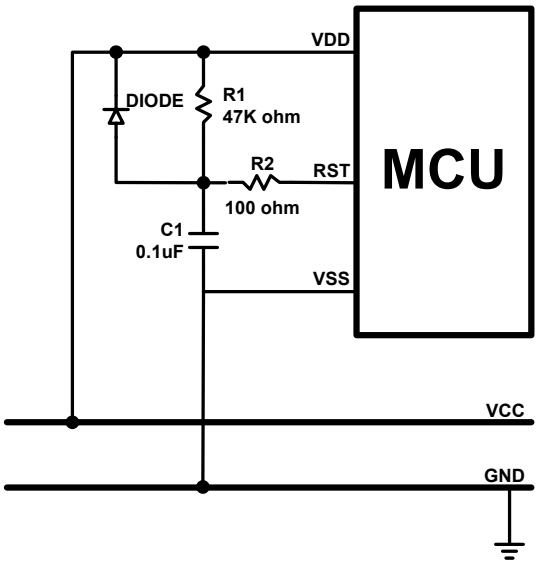
6.6.1 RC 复位电路



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，为系统提供适当的复位时间。

✷ **注：外部复位电路不能解决非正常上电和掉电复位问题。**

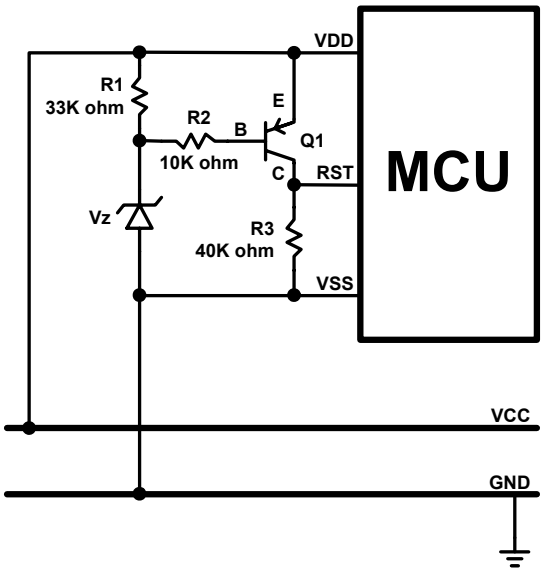
6.6.2 二极管及 RC 复位电路



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。

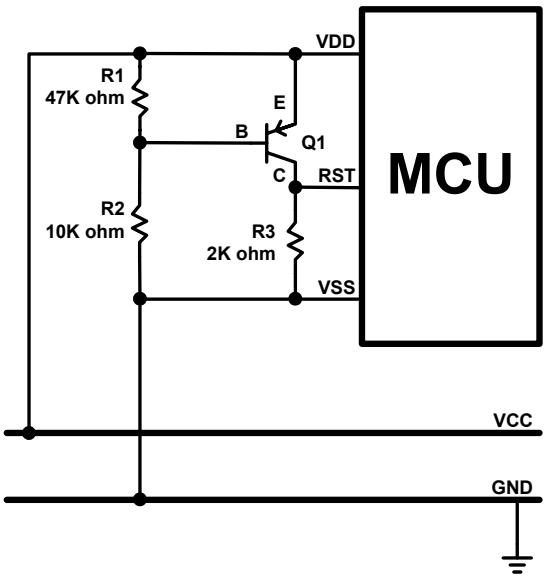
✷ **注:“基本 RC 复位电路”和“二极管及 RC 复位电路”中的电阻 R2 都是必不可少的限流电阻，以避免复位引脚 ESD（Electrostatic Discharge）或 EOS（Electrical Over-stress）击穿 。**

6.6.3 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。稳压二极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稳压二极管。

6.6.4 电压偏移复位电路

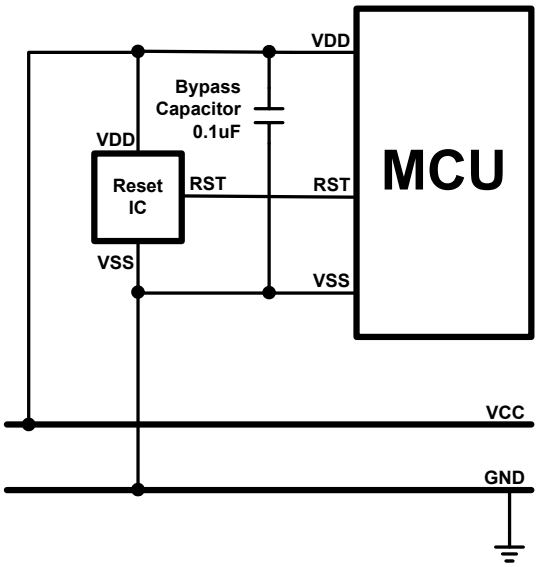


电压偏移复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，偏压复位电路的的检测电压值的精确度有所降低。电路中，R1 和 R2 构成分压电路，当 VDD 高于和等于分压值“0.7V x (R1 + R2) / R1”时，三极管集电极输出高电平，单片机正常工作；VDD 低于“0.7V x (R1 + R2) / R1”时，集电极 C 输出低电平，单片机复位。

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。**

6.6.5 外部 IC 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。

7 振荡器

7.1 概述

SN8P2710 系列内带双时钟系统：高速时钟和低速时钟。高速时钟由外部振荡电路提供，低速时钟则由内置 RC 振荡电路提供。

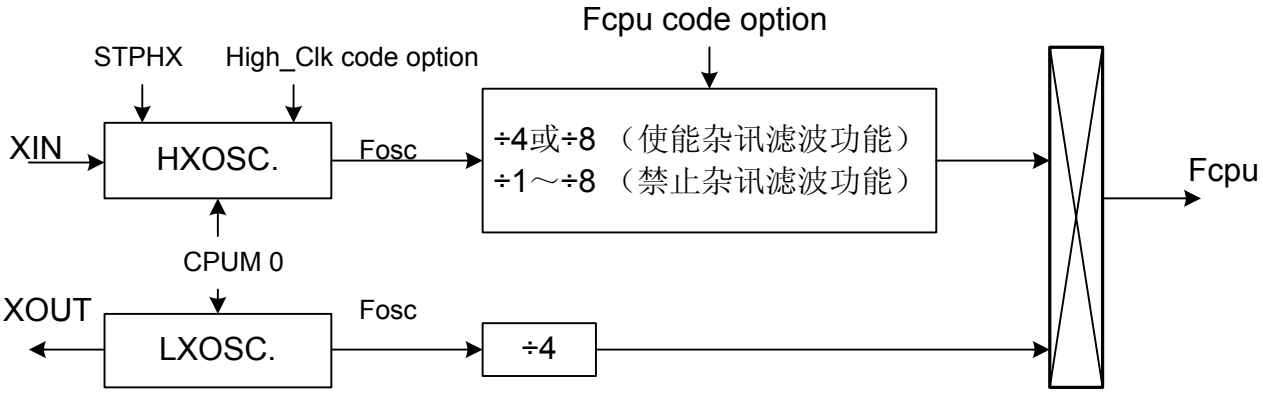


图 7-1 系统时钟框图

- 普通模式下，Fosc = Fhosc（外部高速时钟）。
- 低速模式下，Fosc = Flosc（内部低速 RC 时钟）。
- Fosc 是系统时钟，Fcpu 是指令周期时钟
- 普通模式下，Fcpu = Fosc/1 ~ Fosc/8。
- 低速模式下，Fcpu = Fosc/4。

下面是需要用到的系统时钟的外围设备：

- ✓ **定时器（ TC0 / TC1 / 看门狗定时器）；**
- ✓ **PWM 输出（PWM0、PWM1）；**
- ✓ **Buzzer 输出（TC0OUT、TC1OUT）；**
- ✓ **ADC。**

7.1.1 OSCM 寄存器

寄存器 OSCM 控制振荡器的状态和系统模式。

OSCM 初始值 = xxxx 000x

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

Bit 3 **CPUM0**：单片机工作模式控制位。
0 = 普通模式；
1 = 睡眠（省电）模式，唤醒后进入普通模式。

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。

Bit1 **STPHX**：外部高速时钟控制位。
0= 外部高速时钟正常运行；
1= 外部高速时钟停止运行。

7.1.2 外部高速振荡器

SN8P2710 可以在 3 种不同的振荡器模式下工作：外部 RC 振荡模式，高速晶体/陶瓷振荡模式（如 12MHz）和标准晶体/陶瓷振荡模式（如 4MHz）。在不同的应用环境下，可以通过“High_Clk”为系统选择合适的高速振荡器模式。

➢ **例：停止外部高速振荡器。**

```
B0BSET    FSTPHX      ; 停止外部高速振荡器。

B0BSET    FCPUM0      ; 停止外部高速振荡器和内低速振荡器并进入睡眠模式。
```

7.1.3 高速振荡器编译选项

SN8P2710 为不同的应用提供 3 种振荡模式：4M、12M 和 RC，以支持不同的振荡器类型和频率。单片机运行高速时钟系统比运行低速时钟系统的电流功耗较大。用户可以在编译前从编译选项表中选择合适的振荡器类型，编译选项表如下所示：

编译选项	振荡器类型	功能说明
High_Clk	Ext_RC	高速时钟为外部 RC 振荡器，从 Xout 引脚输出 Fpcu 时钟。
	12M_X'tal	高速时钟为外部高速振荡器，频率范围为 12M~16MHz。
	4M_X'tal	高速时钟为外部振荡器，频率范围为 4M~10MHz。

7.1.4 系统振荡器电路

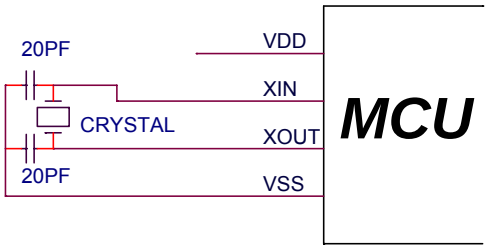


图 7-2. 石英振荡器

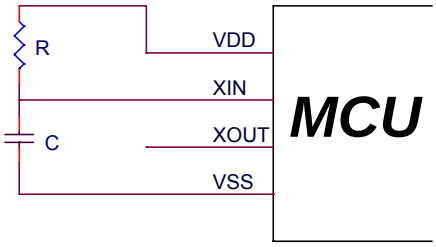


图 7-3. RC 振荡器

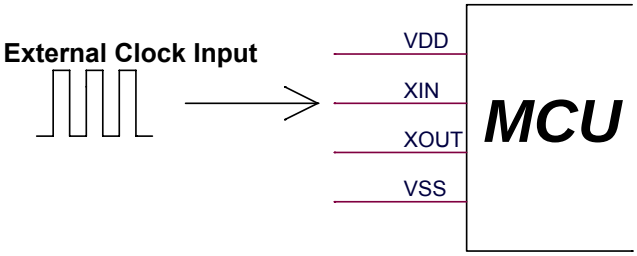


图 7-4. 外部时钟输入

- **注 1：外部振荡电路的 VDD 和 VSS 必须来自单片机，而不是相邻端。**
- **注 2：外部时钟输入可以选择 RC 振荡器或石英振荡器，产生的时钟由 XIN 引脚输入。**
- **注 3：外部振荡器的 VDD 和 VSS 必须和单片机的 VDD 和 VSS 相连，以提高整个系统的性能。**

7.1.5 外部 RC 振荡器频率测试

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：外部振荡器的指令周期测试。**

```
        B0BSET      P2M.0           ; 设置 P2.0 为输出模式，输出频率信号。
@@:
        B0BSET      P2.0            ; 在低速模式下输出频率信号。
        B0BCLR      P2.0            ; 利用示波器测量 Fcpu 频率。
        JMP         @B
```

7.2 内部低速振荡器

内部低速时钟源即内置的低速振荡器，采用 RC 振荡电路，可提供给系统时钟、定时计数器和看门狗定时器等作为时钟源。

➢ **例：停止内部低速振荡器。**

```
        B0BSET      FCPUM0          ; 停止高低速振荡器并进入睡眠模式。
```

✸ **注：内部低速时钟由寄存器 OSCM 的 CPUM0 位控制，不能单独停止运行。**

低速振荡器采用 RC 振荡电路，频率会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。RC 振荡器的频率与电压的关系如下图所示：

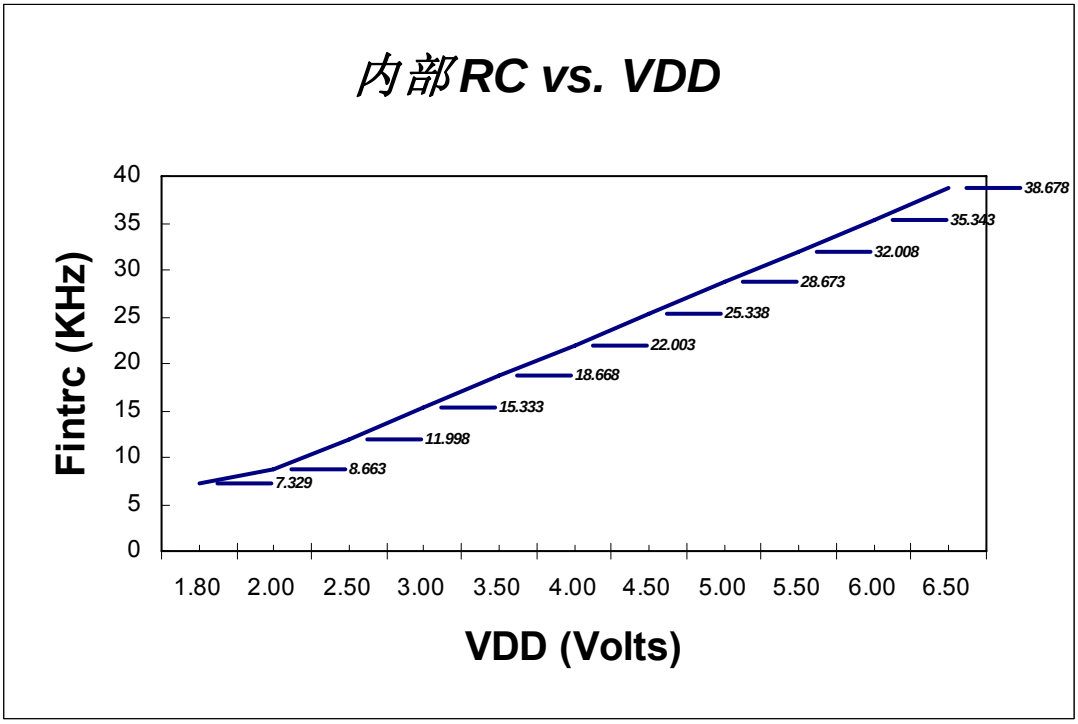


图 7-5. 内部 RC 和 VDD 的关系

➢ **例：由 Fcpu 测试内部 RC 的频率。**

```
        B0BSET      P2M.0           ; 设置 P2.0 为输出模式，输出频率信号。

        B0BSET      FCLKMD          ; 切换到内部低速。
@@:
        B0BSET      P2.0            ; 低速模式下输出频率信号。
        B0BCLR      P2.0            ;
        JMP         @B
```

7.3 系统模式

7.3.1 概述

SN8P2710 系列的单片机可以在 3 种不同的工作模式下进行切换。
实际应用中，可以通过 OSCM 寄存器调整系统的工作模式。

7.3.2 普通模式

普通模式下，系统采用外部高速时钟。系统上电后，系统默认为普通模式，所有的软硬件都在普通模式下正常运行。系统可以切换进入睡眠模式和低速模式。

7.3.3 低速模式

低速模式下，系统采用内部低速 RC 时钟。设置 CLKMD = 1，系统进入内部低速模式。低速模式和普通模式的工作状态相似，仅时钟频率有所降低。系统可以切换进入睡眠模式和普通模式。设置 STPHX = 1，可以停止外部高速振荡器，也可以降低系统功耗。

7.3.4 睡眠模式

睡眠模式又称为省电模式，在睡眠模式下，系统停止工作，功耗低至接近于零。睡眠模式通常用于电池供电等节电系统。设 CPUM0 = 1，系统进入睡眠模式，外部高速和内部低速时钟都停止运行，P0（P0.0、P0.1 和 P0.2）的触发信号可以将系统唤醒进入普通模式。

✷ 注：
Watch_Dog = Enable，在睡眠模式下关闭看门狗定时器，在普通模式下开启。
Watch_Dog = Always_ON，在普通模式和睡眠模式下都开启看门狗定时器。

7.4 系统工作模式

7.4.1 SN8P2710 系统模式切换框图

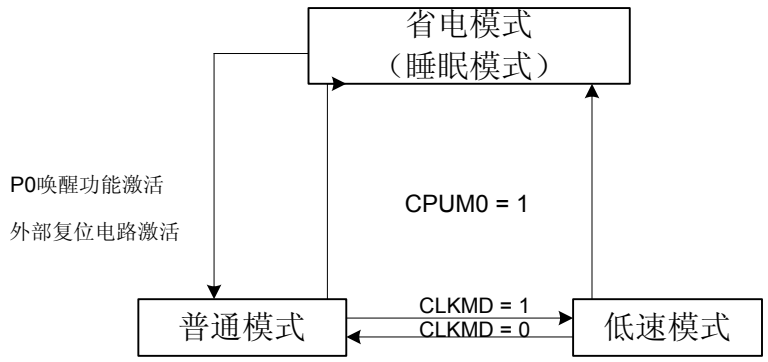


图 7-6. SN8P2710 系统模式切换示意图

模式	普通模式	低速模式	睡眠模式	备注
HX osc.	运行	由 STPHX 控制	停止	
LX osc.	运行	运行	停止	
CPU 指令	执行	执行	停止	
TC0/TC1	*有效	*有效	无效	* 由程序激活
看门狗定时器	有效	有效	由 Watch_Dog 编译选项控制	
内部中断	全部有效	全部有效	全部无效	
外部中断	全部有效	全部有效	全部无效	
唤醒功能	-	-	P0，RST，LVD，*看门狗定时器复位	* Watchdog = Always_ON

表 7-1. 工作模式说明

7.4.2 系统模式切换

普通/低速模式切换进入睡眠模式。
CPUM0 = 1

```
        B0BSET      FCPUM0          ; CPUM0 = 1。
```

系统在睡眠模式下，只有具有唤醒功能的引脚和复位引脚可以将系统唤醒。

普通模式切换进入低速模式。

```
        B0BSET      FCLKMD          ; 设置 CLKMD = 1，进入低速模式。
        B0BSET      FSTPHX          ; 高速时钟停止运行。
```

✸ **注：可以不用停止高速振荡器。**

低速模式切换进入普通模式（外部高速振荡器仍然运行）。

```
        B0BCLR      FCLKMD
```

低速模式切换进入普通模式（停止外部高速振荡器）。

若外部高速时钟停止时系统欲返回到普通模式，需延迟 10ms 等待外部时钟稳定。

```
        B0BCLR      FSTPHX          ; 启动外部高速振荡器。

        B0MOV       Z, #27          ; 若 VDD = 5V，内部 RC = 32KHz（典型值）
@@:     DECMS       Z               ; 外部高速振荡器稳定时间为 0.125ms×81 = 10.125ms。
        JMP         @B
                                    ;
        B0BCLR      FCLKMD          ; 进入普通模式。
```

7.5 喚醒時間

7.5.1 概述

外部高速振荡器从停止到重新正常运行需要一段时间的延迟，以等待振荡器稳定工作。外部高速振荡器重新启动需要的这段延迟时间称为喚醒时间。

有两种情况需要喚醒时间，一是有睡眠模式切换进入普通模式，这里 SN8P2710 提供了 4096 个振荡周期作为喚醒时间；另外一种是由低速模式切换进入普通模式，由用户自己提供喚醒时间。

7.5.2 喚醒时间

睡眠模式下，系统停止外部高速振荡器。从睡眠模式喚醒时，系统提供 4096 个外部高速振荡周期作为喚醒时间，等待外部振荡电路稳定工作。喚醒时间结束后，系统进入普通模式。喚醒时间的计算方法如下：

喚醒时间 = 1/Fosc * 3584（sec）+ X'tal 稳定时间

x'tal 的稳定时间由 x'tal 的类型决定，一般约为 2~4ms。

- **例：睡眠模式下，系统由 P0 触发信号喚醒进入普通模式。P0 的喚醒时间如下：**

喚醒时间 = 1/Fosc * 3584 = 1.001 ms（Fosc = 3.58MHz）

总的喚醒时间 = 1.001 ms + x'tal 稳定时间

睡眠模式下，只有具有喚醒功能的 P0.0 和 P0.1 可以将系统喚醒进入普通模式。

8 定时器

8.1 看门狗定时器（WDT）

内置的看门狗定时器用于监控系统的正常运行，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。用户必须在看门狗溢出前将看门狗定时器清零。看门狗定时器的时钟源来自内部低速 RC 振荡器。WDT 的溢出时间如下：

1 / (16K ÷ 512 ÷ 16) ~ 0.5s @ 3V
1 / (32K ÷ 512 ÷ 16) ~ 0.25s @ 5V

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	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

设置 WDTR = 0x5A，清看门狗定时器。

✸ 注：
在编译选项中，Watch_Dog 共有 3 种选项：Always_ON、Enable 和 Disabled。
Watch_Dog = Enable：在睡眠模式下关闭看门狗定时器，在普通模式下开启。
Watch_Dog = Always_ON：在睡眠模式和普通模式下都开启看门狗定时器。

➢ **例：下例是对看门狗定时器的操作，在主程序的开始对看门狗清零。**

```
Main:
        MOV     A,# 5AH
        MOV     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 TC0 可以用作通用定时器、Buzzer 输出和 PWM 输出，具有自动重新装载功能，主要由两部分组成：8 位自动装载寄存器 TC0R，用来存储计数参考值；8 位自动加 1 的计数寄存器 TC0C。

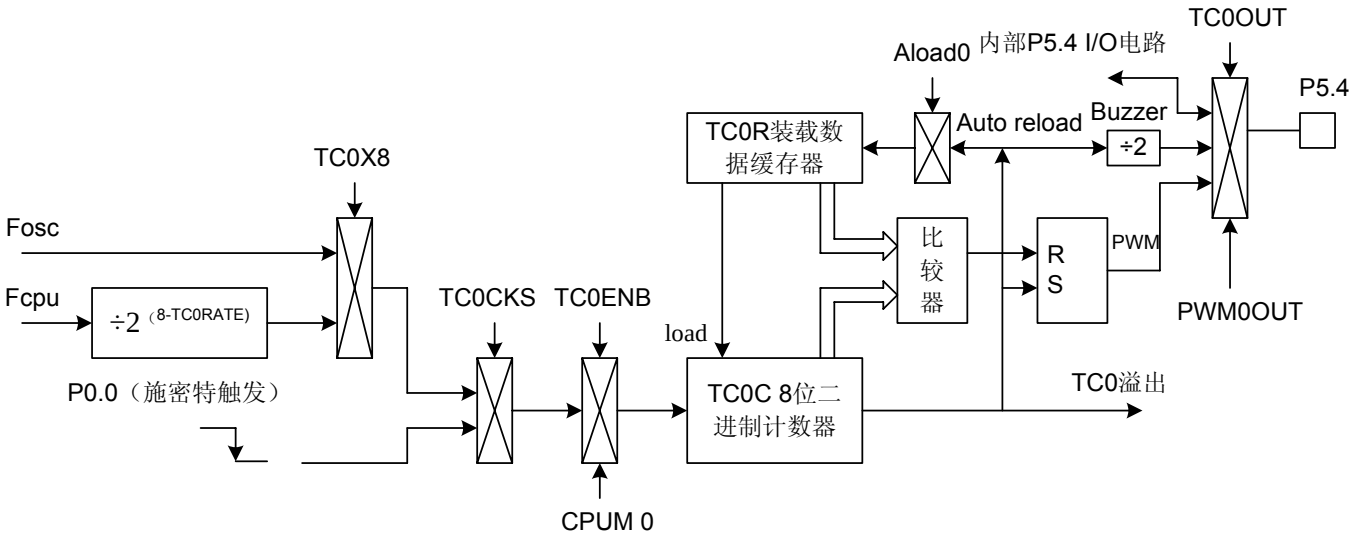


图 8-1. TC0 结构框图

TC0 的主要功能如下：

- **8 位可编程定时器：**根据所选的时钟频率，定时发出中断请求信号。
- **Buzzer 输出：**通过 BZ0 引脚（P5.4）输出一个可选择的时钟频率。
- **PWM：**PWM 输出由 PWM0OUT 位控制，PWM0OUT 引脚（P5.4）输出。

8.2.2 TC0M 模式寄存器

8 位读/写模式寄存器 TC0M 通过载入不同值，可以在执行程序的过程中动态的调整定时/计数器的频率。

通过设置 TC0RATE0~TC0RATE2 位和 TC0X8 位，TC0 可提供 8 种时钟源频率选择。若 TC0X8 = 1，TC0 的时钟源来自 Fosc，范围为 Fosc/1 ~ Fosc/128；若 TC0X8 = 0（初始化）时，TC0 时钟源的频率范围为 Fcpu/2~Fcpu/256。TC0M 的初始值为 0，时钟源频率为 Fcpu/256。TC0M 的 bit7（TC0ENB）是 TC0 定时器的启动控制位。

T0M 初始值 = xxxx 00xx

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	-	-	-	-	TC1X8	TC0X8	-	-
	-	-	-	-	R/W	R/W	-	-

Bit3 **TC1X8**：TC1 频率选择控制位，详见 TC1M 寄存器。
0 = TC1 时钟来自 Fcpu；
1 = TC1 时钟来自 Fosc。

Bit2 **TC0X8**：TC0 频率选择控制位，详见 TC0M 寄存器。
0 = TC0 时钟来自 Fcpu；
1 = TC0 时钟来自 Fosc。

✷ **注：在 TC0 处于时间计数器模式（TC0CKS=1）下，TC0X8 和 TC0RATE 可以忽略不计。**

TC0M 初始值 = 0000 0000

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
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

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

Bit [6:4] **TC0RATE [2:0]**：TC0 内部时钟速率选择位，仅对 TC0CKS = 0 有效。

TC0Rate	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 3 **TC0CKS**：TC0 时钟信号控制位。
0 = 内部时钟信号（Fcpu 或 Fosc）；
1 = 来自 P0.0（INT0）的外部时钟信号。

Bit 2 **ALOAD0**：自动装载控制位。
0 = 禁止；
1 = 使能。

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

Bit 0 **PWM0OUT**：PWM 输出控制位。
0 = 禁止 PWM 输出；
1 = 使能 PWM 输出，自动禁止 TC0OUT 功能。

PWM0OUT = 1，TC0X8=0

ALOAD0	TC0OUT	TC0 溢出边界	PWM 占空比	PWM 的最大频率 （Fosc = 16M）（Fcpu = 4M）	备注
0	0	0FFH ~ 00H	0/256 ~ 255/256	7.8125K	每计数 256 次溢出
0	1	3FH ~ 40H	0/64 ~ 63/64	31.25K	每计数 64 次溢出
1	0	1FH ~ 20H	0/32 ~ 31/32	62.5K	每计数 32 次溢出
1	1	0FH ~ 10H	0/16 ~ 15/16	125K	每计数 16 次溢出

PWM0OUT = 1，TC0X8=1

ALOAD0	TC0OUT	TC0 溢出边界	PWM 占空比	PWM 的最大频率 （Fosc = 16M）（Fcpu = 4M）	备注
0	0	0FFH ~ 00H	0/256 ~ 255/256	62.5K	每计数 256 次溢出
0	1	3FH ~ 40H	0/64 ~ 63/64	250K	每计数 64 次溢出
1	0	1FH ~ 20H	0/32 ~ 31/32	500K	每计数 32 次溢出
1	1	0FH ~ 10H	0/16 ~ 15/16	1000K	每计数 16 次溢出

✷ **注：TC0CKS = 1 时，TC0 作为外部事件计数器，不再响应 P0.0 的中断请求。（P0.0IRQ 始终置 0）。**

8.2.3 TC0C 计数寄存器

8 位计数寄存器 TC0C 是加 1 计数器，时钟源频率由 TC0RATE0~TC0RATE2 决定。只要 TC0ENB 置 1 就启动定时器。当 TC0C 计数到 0FFH，下一次计数 TC0C 将清零为“00H”，并且 TC0 中断请求标志置 1，若此时 TC0IEN = 1，系统将执行 TC0 中断服务程序。

TC0C 初始值 = xxxx xxxx

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
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TC0C 初始值的计算如下：

TC0C 初始值 = 256 - （TC0 中断间隔时间*输入时钟）

- **例：3.58MHz 高速模式，TC0 的中断间隔时间为 10ms。TC0C（74H） = 256 - （10ms * fcpu/256）**

  TC0C 初始值 = 256 - （TC0 中断间隔时间 * 输入时钟）
  = 256 - （10ms * 3.58 * 10^6 / 256）
  = 256 - （10^{-2} * 3.58 * 10^6 / 256）
  = 116= 74H

TC0_Counter=8-bit， TC0X8=0

TC0RATE	TC0CLOCK	高速模式（fcpu = 3.58MHz / 4）		低速模式（fcpu = 32768Hz / 4）	
		最大溢出间隔时间	单步间隔时间 = max/256	最大溢出间隔时间	单步间隔时间 = max/256
000	fcpu/256	73.2 ms	286us	8000 ms	31.25 ms
001	fcpu/128	36.6 ms	143us	4000 ms	15.63 ms
010	fcpu/64	18.3 ms	71.5us	2000 ms	7.8 ms
011	fcpu/32	9.15 ms	35.8us	1000 ms	3.9 ms
100	fcpu/16	4.57 ms	17.9us	500 ms	1.95 ms
101	fcpu/8	2.28 ms	8.94us	250 ms	0.98 ms
110	fcpu/4	1.14 ms	4.47us	125 ms	0.49 ms
111	fcpu/2	0.57 ms	2.23us	62.5 ms	0.24 ms

TC0_Counter=6-bit， TC0X8=0

TC0RATE	TC0CLOCK	高速模式（fcpu = 3.58MHz / 4）		低速模式（fcpu = 32768Hz / 4）	
		最大溢出间隔时间	单步间隔时间 = max/256	最大溢出间隔时间	单步间隔时间 = max/256
000	fcpu/256	18.3 ms	71.5us	2000 ms	7.8 ms
001	fcpu/128	9.15 ms	35.8us	1000 ms	3.9 ms
010	fcpu/64	4.57 ms	17.9us	500 ms	1.95 ms
011	fcpu/32	2.28 ms	8.94us	250 ms	0.98 ms
100	fcpu/16	1.14 ms	4.47us	125 ms	0.49 ms
101	fcpu/8	0.57 ms	2.23us	62.5 ms	0.24 ms
110	fcpu/4	0.285 ms	1.11us	31.25 ms	0.12 ms
111	fcpu/2	0.143 ms	0.56 us	15.63 ms	0.06 ms

TC0_Counter=5-bit， TC0X8=0

TC0RATE	TC0CLOCK	高速模式（fcpu = 3.58MHz / 4）		低速模式（fcpu = 32768Hz / 4）	
		最大溢出间隔时间	单步间隔时间 = max/256	最大溢出间隔时间	单步间隔时间 = max/256
000	fcpu/256	9.15 ms	35.8us	1000 ms	3.9 ms
001	fcpu/128	4.57 ms	17.9us	500 ms	1.95 ms
010	fcpu/64	2.28 ms	8.94us	250 ms	0.98 ms
011	fcpu/32	1.14 ms	4.47us	125 ms	0.49 ms
100	fcpu/16	0.57 ms	2.23us	62.5 ms	0.24 ms
101	fcpu/8	0.285 ms	1.11us	31.25 ms	0.12 ms
110	fcpu/4	0.143 ms	0.56 us	15.63 ms	0.06 ms
111	fcpu/2	71.25 us	0.278 us	7.81 ms	0.03 ms

TC0_Counter=4-bit， TC0X8=0

TC0RATE	TC0CLOCK	高速模式（fcpu = 3.58MHz / 4）		低速模式（fcpu = 32768Hz / 4）	
		最大溢出间隔时间	单步间隔时间 = max/256	最大溢出间隔时间	单步间隔时间 = max/256
000	fcpu/256	4.57 ms	17.9us	500 ms	1.95 ms
001	fcpu/128	2.28 ms	8.94us	250 ms	0.98 ms
010	fcpu/64	1.14 ms	4.47us	125 ms	0.49 ms
011	fcpu/32	0.57 ms	2.23us	62.5 ms	0.24 ms
100	fcpu/16	0.285 ms	1.11us	31.25 ms	0.12 ms
101	fcpu/8	0.143 ms	0.56 us	15.63 ms	0.06 ms
110	fcpu/4	71.25 us	0.278 us	7.81 ms	0.03 ms
111	fcpu/2	35.63 us	0.139 us	3.91 ms	0.015 ms

8.2.4 TC0R 自动装载寄存器

8 位自动装载寄存器 TC0R 主要用于 TC0OUT 和 PWM0OUT 功能，在 TC0OUT 功能下，用户必须开启 TC0R 功能，其主要功能如下：

- 存放自动装载的数据，当 TC0C 溢出时送入 TC0C（ALOAD0 = 1）；
- 存放 PWM0OUT 的占空比。

TC0R = xxxx xxxx

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
	W	W	W	W	W	W	W	W

TC0R 初始值的计算方法类似于 TC0C，如下所示：

TC0R 初始值 = 256 - （TC0 中断间隔时间 * 输入时钟）

注：TC0R 是只写寄存器，不能执行 INCMS、DECMS 指令。

8.2.5 TC0 操作流程

TC0 的操作流程如下：

- 设置初始值 TC0C 以设定定时器的中断间隔时间。
- TC0ENB 置 1，TC0 开始计数。
- 根据 T0M 选择的时钟源频率，TC0C 在每个时钟周期加 1。
- TC0C 从 0FFH 计数到 00H，TC0C 溢出。
- TC0C 溢出，TC0IRQ 置 1。
- 执行中断服务程序。
- 用户重置 TC0C，TC0 操作重新开始。

➢ **例：初始化 TC0M 和 TC0C，无自动装载功能。**

```
B0BCLR      FTC0IEN     ; 禁止 TC0 中断。
B0BCLR      FTC0ENB     ; 禁止 TC0 计数器。
MOV         A,#00H      ;
B0MOV       TC0M,A      ; TC0 时钟 = fcpu / 256。
MOV         A,#74H      ; TC0 初始值 = 74H。
B0MOV       TC0C,A      ; TC0 间隔时间 = 10 ms。

B0BSET      FTC0IEN     ; 使能 TC0 中断。
B0BCLR      FTC0IRQ     ; 清 TC0IRQ。
B0BSET      FTC0ENB     ; 开启 TC0 计数器。
```

➢ **例：初始化 TC0M 和 TC0C，有自动装载功能。**

```
B0BCLR      FTC0IEN     ; 禁止 TC0 中断。
B0BCLR      FTC0ENB     ; 禁止 TC0 计数器。
MOV         A,#00H      ;
B0MOV       TC0M,A      ; TC0 时钟 = fcpu / 256。
MOV         A,#74H      ; TC0 初始值 = 74H。
B0MOV       TC0C,A      ; TC0 间隔时间 = 10 ms。
B0MOV       TC0R,A      ; 设置 TC0R 自动装载寄存器。

B0BSET      FTC0IEN     ; 使能 TC0 中断。
B0BCLR      FTC0IRQ     ; 清 TC0IRQ。
B0BSET      FTC0ENB     ; 开启 TC0 计数器。
B0BSET      ALOAD0      ; 使能 TC0 的自动装载功能。
```

➢ **例：TC0 中断服务程序，无自动装载功能。**

```
        ORG     8H              ;
        JMP     INT_SERVICE
INT_SERVICE:

        B0XCH   A, ACCBUF       ; 保存 ACC。
        B0MOV   A, PFLAG        ;
        B0MOV   PFLAGBUF, A     ; 保存 PFLAG。

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

        B0BCLR  FTC0IRQ         ; 清 TC0IRQ。
        MOV     A,#74H          ; 重新装载 TC0C。
        B0MOV   TC0C,A
        .       .               ; TC0 中断程序。
        .       .
        JMP     EXIT_INT        ; 中断结束。
        .       .
        .       .
EXIT_INT:
                                ;
        B0MOV   A, PFLAGBUF
        B0MOV   PFLAG, A        ; 恢复 PFLAG。
        B0XCH   A, ACCBUF       ; 恢复 ACC。

        RETI                    ; 中断返回。
```

➢ **例：TC0 中断服务程序，有自动装载功能。**

```
        ORG     8H
        JMP     INT_SERVICE
INT_SERVICE:

        B0XCH   A, ACCBUF       ; 保存 ACC。
        B0MOV   A, PFLAG        ;
        B0MOV   PFLAGBUF, A     ; 保存 PFLAG。


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

        B0BCLR  FTC0IRQ         ; 清 TC0IRQ。
        .       .               ; TC0 中断服务程序。
        .       .
        JMP     EXIT_INT        ; 中断结束。
        .       .
        .       .
EXIT_INT:
        B0MOV   A, PFLAG        ;
        B0MOV   PFLAGBUF, A     ; 恢复 PFLAG。
        B0XCH   A, ACCBUF       ; 恢复 ACC。

        RETI                    ; 中断返回。
```

8.2.6 TC0 时钟频率输出（BUZZER）

对 TC0 时钟频率进行适当设置可得到特定频率的蜂鸣器输出（TC0OUT），并通过引脚 P5.4 输出。单片机内部设置 TC0 的溢出频率经过 2 分频后作为 TC0OUT 的频率，即 TC0 每溢出 2 次 TC0OUT 输出一个完整的脉冲，此时，P5.4 的 I/O 功能自动被禁止。TC0OUT 输出波形如下：

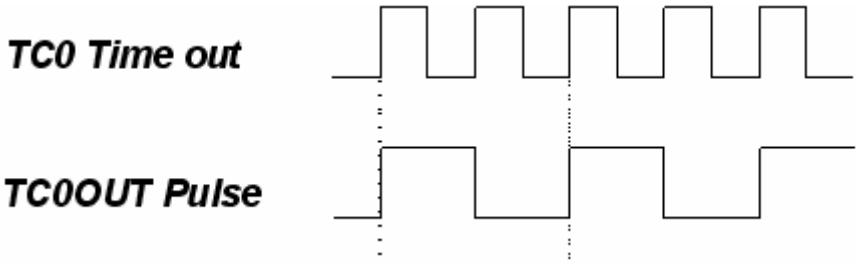


图 8-2 TC0OUT 频率图

➢ **例：设置 TC0OUT (P5.4)。其中，Fcpu = 4MHz，TC0OUT = 1KHz，TC0 = 2KHz，TC0 时钟源采用内部时钟 Fcpu/4，TC0RATE2~TC0RATE1 = 110， TC0C = TC0R = 131，TC0X8=1。**

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

MOV     A,#131          ; 自动装载参考值设置。
B0MOV   TC0C,A
B0MOV   TC0R,A
B0BCLR  FTC0X8

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。
```

8.3 定时/计数器 TC1

8.3.1 概述

8 位二进制定时/计数器 TC1 主要用来产生定时中断请求信号，具有自动重新装载功能，主要由两部分组成：8 位自动装载寄存器 TC1R，用来存储计数参考值；8 位自动加 1 的计数寄存器 TC1C。

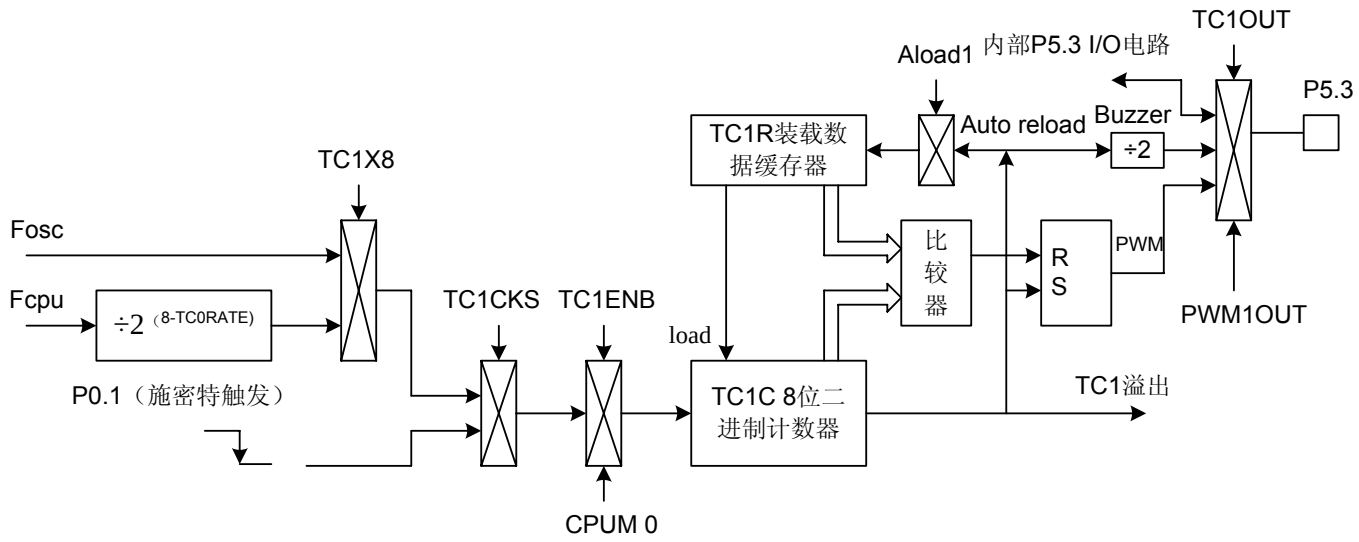


图 8-2. TC1 结构框图

TC1 的主要功能如下：

- **8 位可编程定时器：**根据所选的时钟频率，定时发出中断请求信号。
- **Buzzer 输出：**通过 BZ1 引脚（P5.3）输出一个可选择的时钟频率。
- **PWM：**PWM 输出由 PWM1OUT 位控制，PWM1OUT 引脚（P5.3）输出。

8.3.2 TC1M 模式寄存器

8 位读/写模式寄存器 TC1M 通过载入不同值，可以在执行程序的过程中动态的调整定时/计数器的频率。

通过设置 TC1RATE0~TC1RATE2 位和 TC1X8 位，TC1 可提供 8 种时钟源频率选择。若 TC1X8 = 1，TC1 的时钟源来自 Fosc，范围为 Fosc/1 ~ Fosc/128；若 TC1X8 = 0（初始化）时，TC1 时钟源的频率范围为 Fcpu/2~Fcpu/256。TC1M 的初始值为 0，时钟源频率为 Fcpu/256。TC1M 的 bit7（TC1ENB）是 TC1 定时器的启动控制位。

T0M = xxxx 00xx

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	-	-	-	-	TC1X8	TC0X8	-	-
	-	-	-	-	R/W	R/W	-	-

Bit3 **TC1X8**：TC1 频率选择控制位，详见 TC1M 寄存器。
0 = TC1 时钟来自 Fcpu；
1 = TC1 时钟来自 Fosc。

Bit2 **TC0X8**：TC0 频率选择控制位，详见 TC0M 寄存器。
0 = TC0 时钟来自 Fcpu；
1 = TC0 时钟来自 Fosc。

✷ **注：在 TC1 处于事件计数器模式（TC1CKS=1）下，TC1X8 和 TC1RATE 无效，可以忽略不计。**

TC1M = 0000 0000

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
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7 **TC1ENB**：TC1 计数器启动控制位。
0 = 禁止；
1 = 使能。

Bit [6:4] **TC1RATE [2:0]**：TC1 内部时钟速率选择位，仅对 TC1CKS = 0 有效。

TC1Rate	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 3 **TC1CKS**：TC1 时钟信号控制位。
0 = 内部时钟信号（Fcpu 或 Fosc）；
1 = 来自 P0.1（INT1）的外部时钟信号。

Bit 2 **ALOAD1**：自动装载控制位。
0 = 禁止；
1 = 使能。

Bit 1 **TC1OUT**：TC1 溢出信号输出控制位，**仅当 PWM1OUT = 0 时有效。**
0 = 禁止，P5.3 作为普通的 I/O 口；
1 = 使能，P5.3 输出 TC0OUT 信号。

Bit 0 **PWM1OUT**：PWM 输出控制位。
0 = 禁止 PWM 输出；
1 = 使能 PWM 输出，自动禁止 TC1OUT 功能。

PWM1OUT = 1，TC1X8=0

ALOAD1	TC1OUT	TC1 溢出边界	PWM 占空比	PWM 的最大频率 （Fosc = 16M）（Fcpu = 4M）	备注
0	0	0FFH ~ 00H	0/256 ~ 255/256	7.8125K	每计数 256 次溢出
0	1	3FH ~ 40H	0/64 ~ 63/64	31.25K	每计数 64 次溢出
1	0	1FH ~ 20H	0/32 ~ 31/32	62.5K	每计数 32 次溢出
1	1	0FH ~ 10H	0/16 ~ 15/16	125K	每计数 16 次溢出

PWM1OUT = 1，TC1X8=1

ALOAD1	TC1OUT	TC1 溢出边界	PWM 占空比	PWM 的最大频率 （Fosc = 16M）（Fcpu = 4M）	备注
0	0	0FFH ~ 00H	0/256 ~ 255/256	62.5K	每计数 256 次溢出
0	1	3FH ~ 40H	0/64 ~ 63/64	250K	每计数 64 次溢出
1	0	1FH ~ 20H	0/32 ~ 31/32	500K	每计数 32 次溢出
1	1	0FH ~ 10H	0/16 ~ 15/16	1000K	每计数 16 次溢出

✷ **注：TC1CKS = 1 时，TC1 是一个外部事件计数器，不再响应 P0.1 的中断请求。（P0.1IRQ 始终置 0）。**

8.3.3 TC1C 计数寄存器

8 位计数寄存器 TC1C 是加 1 计数器，时钟源频率由 TC1RATE0~TC1RATE2 决定。只要 TC1ENB 置 1 就启动定时器。当 TC1C 计数到 0FFH，下一次计数 TC1C 将清零为“00H”，并且 TC1 中断请求标志置 1，若此时 TC1IEN = 1，系统将执行 TC1 中断服务程序。

TC1C = xxxx xxxx

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
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

TC1C 初始值的计算如下：

TC1C 初始值 = 256 -（TC1 中断间隔时间*输入时钟）

- **例：3.58MHz 高速模式，TC1 的中断间隔时间为 10ms。TC1C（74H） = 256 - （10ms * fcpu/256）**

  TC1C 初始值 = 256 -（TC1 中断间隔时间 * 输入时钟）
  = 256 -（10ms * 3.58 * 10^6 / 256）
  = 256 -（10^{-2} * 3.58 * 10^6 / 256）
  = 116= 74H

TC1_Counter=8-bit, TC1X8=0

TC1RATE	TC1CLOCK	高速模式（fcpu = 3.58MHz / 4）		低速模式（fcpu = 32768Hz / 4）	
		最大溢出间隔时间	单步间隔时间 = max/256	最大溢出间隔时间	单步间隔时间 = max/256
000	fcpu/256	73.2 ms	286us	8000 ms	31.25 ms
001	fcpu/128	36.6 ms	143us	4000 ms	15.63 ms
010	fcpu/64	18.3 ms	71.5us	2000 ms	7.8 ms
011	fcpu/32	9.15 ms	35.8us	1000 ms	3.9 ms
100	fcpu/16	4.57 ms	17.9us	500 ms	1.95 ms
101	fcpu/8	2.28 ms	8.94us	250 ms	0.98 ms
110	fcpu/4	1.14 ms	4.47us	125 ms	0.49 ms
111	fcpu/2	0.57 ms	2.23us	62.5 ms	0.24 ms

TC1_Counter=6-bit, TC1X8=0

TC1RATE	TC1CLOCK	高速模式（fcpu = 3.58MHz / 4）		低速模式（fcpu = 32768Hz / 4）	
		最大溢出间隔时间	单步间隔时间 = max/256	最大溢出间隔时间	单步间隔时间 = max/256
000	fcpu/256	18.3 ms	71.5us	2000 ms	7.8 ms
001	fcpu/128	9.15 ms	35.8us	1000 ms	3.9 ms
010	fcpu/64	4.57 ms	17.9us	500 ms	1.95 ms
011	fcpu/32	2.28 ms	8.94us	250 ms	0.98 ms
100	fcpu/16	1.14 ms	4.47us	125 ms	0.49 ms
101	fcpu/8	0.57 ms	2.23us	62.5 ms	0.24 ms
110	fcpu/4	0.285 ms	1.11us	31.25 ms	0.12 ms
111	fcpu/2	0.143 ms	0.56 us	15.63 ms	0.06 ms

TC1_Counter=5-bit, TC1X8=0

TC1RATE	TC1CLOCK	高速模式（fcpu = 3.58MHz / 4）		低速模式（fcpu = 32768Hz / 4）	
		最大溢出间隔时间	单步间隔时间 = max/256	最大溢出间隔时间	单步间隔时间 = max/256
000	Fcpu/256	9.15 ms	35.8us	1000 ms	3.9 ms
001	Fcpu/128	4.57 ms	17.9us	500 ms	1.95 ms
010	fcpu/64	2.28 ms	8.94us	250 ms	0.98 ms
011	fcpu/32	1.14 ms	4.47us	125 ms	0.49 ms
100	fcpu/16	0.57 ms	2.23us	62.5 ms	0.24 ms
101	fcpu/8	0.285 ms	1.11us	31.25 ms	0.12 ms
110	fcpu/4	0.143 ms	0.56 us	15.63 ms	0.06 ms
111	fcpu/2	71.25 us	0.278 us	7.81 ms	0.03 ms

TC1_Counter=4-bit, TC1X8=0

TC1RATE	TC1CLOCK	高速模式（fcpu = 3.58MHz / 4）		低速模式（fcpu = 32768Hz / 4）	
		最大溢出间隔时间	单步间隔时间 = max/256	最大溢出间隔时间	单步间隔时间 = max/256
000	Fcpu/256	4.57 ms	17.9us	500 ms	1.95 ms
001	Fcpu/128	2.28 ms	8.94us	250 ms	0.98 ms
010	fcpu/64	1.14 ms	4.47us	125 ms	0.49 ms
011	fcpu/32	0.57 ms	2.23us	62.5 ms	0.24 ms
100	fcpu/16	0.285 ms	1.11us	31.25 ms	0.12 ms
101	fcpu/8	0.143 ms	0.56 us	15.63 ms	0.06 ms
110	fcpu/4	71.25 us	0.278 us	7.81 ms	0.03 ms
111	fcpu/2	35.63 us	0.139 us	3.91 ms	0.015 ms

8.3.4 TC1R 自动装载寄存器

8 位自动装载寄存器 TC1R 主要用于 TC1OUT 和 PWM1OUT 功能，在 TC1OUT 功能下，用户必须开启 TC1R 功能，其主要功能如下：

- 存放自动装载的数据，当 TC1C 溢出时送入 TC1C（ALOAD1 = 1）；
- 存放 PWM1OUT 的占空比。

TC1R = xxxx xxxx

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
	W	W	W	W	W	W	W	W

TC1R 初始值的计算方法类似于 TC1C，如下所示：

TC1R 初始值 = 256 -（TC1 中断间隔时间 * 输入时钟）

注：TC1R 是只写寄存器，不能执行 INCMS、DECMS 指令。

8.3.5 TC1 操作流程

TC1 的操作流程如下：

- 设置初始值 TC1C 以设定定时器的中断间隔时间。
- TC1ENB 置 1，TC1 开始计数。
- 根据 T0M 选择的时钟源频率，TC1C 在每个时钟周期加 1。
- TC1C 从 0FFH 计数到 00H，TC1C 溢出。
- TC1C 溢出，TC1IRQ 置 1。
- 执行中断服务程序。
- 用户重置 TC1C，TC1 操作重新开始。

➢ **例：初始化 TC1M 和 TC1C，无自动装载功能。**

```
B0BCLR      FTC1IEN     ; 禁止 TC1 中断。
B0BCLR      FTC1ENB     ; 禁止 TC1 计数器。
B0BCLR      FTC1X8      ;
MOV         A,#00H
B0MOV       TC1M,A      ; TC1 时钟 = fcpu / 256。
MOV         A,#74H      ; TC1 初始值 = 74H。
B0MOV       TC1C,A      ; TC1 间隔时间 = 10 ms。

B0BSET      FTC1IEN     ; 使能 TC1 中断。
B0BCLR      FTC1IRQ     ; 清 TC1IRQ。
B0BSET      FTC1ENB     ; 开启 TC1 计数器。
```

➢ **例：初始化 TC1M 和 TC1C，有自动装载功能。**

```
B0BCLR      FTC1IEN     ; 禁止 TC1 中断。
B0BCLR      FTC1ENB     ; 禁止 TC1 计数器。
B0BCLR      FTC1X8      ;
MOV         A,#00H
B0MOV       TC1M,A      ; TC1 时钟 = fcpu / 256。
MOV         A,#74H      ; TC1 初始值 = 74H。
B0MOV       TC1C,A      ; TC1 间隔时间 = 10 ms。
B0MOV       TC1R,A      ; 设置 TC1R 自动装载寄存器。

B0BSET      FTC1IEN     ; 使能 TC1 中断。
B0BCLR      FTC1IRQ     ; 清 TC1IRQ。
B0BSET      FTC1ENB     ; 开启 TC1 计数器。
B0BSET      ALOAD1      ; 使能 TC1 的自动装载功能。
```

➢ **例：TC1 中断服务程序，无自动装载功能。**

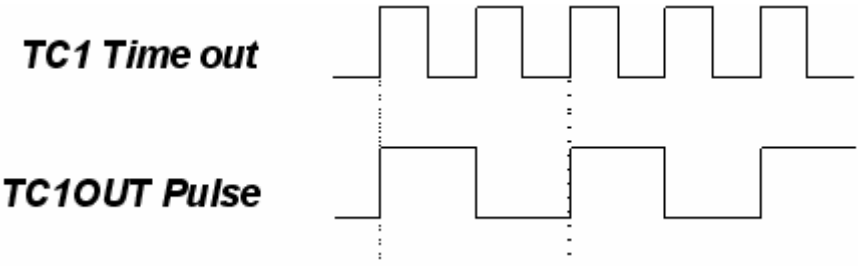
```
                ORG             8                   ;
                JMP             INT_SERVICE
INT_SERVICE:

                B0XCH           A, ACCBUF           ; 保存 ACC。
                B0MOV           A, PFLAG            ;
                B0MOV           PFLAGBUF, A         ; 保存 PFLAG。

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

                B0BCLR          FTC1IRQ             ; 清 TC1IRQ。
                MOV             A,#74H              ; 重新装载 TC1C。
                B0MOV           TC1C,A
                .               .                   ; TC1 中断程序。
                .               .
                JMP             EXIT_INT            ; 中断结束。
                .               .
                .               .
EXIT_INT:
                B0MOV           A, PFLAGBUF         ;
                B0MOV           PFLAG, A            ; 恢复 PFLAG。
                B0XCH           A, ACCBUF           ; 恢复 ACC。

                RETI                                ; 中断返回。
```

➢ **例：TC1 中断服务程序，有自动装载功能。**

```
                ORG             8                   ;
                JMP             INT_SERVICE
INT_SERVICE:

                B0XCH           A, ACCBUF           ; 保存 ACC。
                B0MOV           A, PFLAG
                B0MOV           PFLAGBUF, A         ; 保存 PFLAG。

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

                B0BCLR          FTC1IRQ             ; 清 TC1IRQ。
                .               .                   ; TC1 中断程序。
                .               .
                JMP             EXIT_INT            ; 中断结束。
                .               .
                .               .
EXIT_INT:
                B0MOV           A, PFLAGBUF
                B0MOV           PFLAG, A            ; 恢复 PFLAG。
                B0XCH           A, ACCBUF           ; 恢复 ACC。

                RETI                                ; 退出中断。
```

8.3.6 TC1 时钟频率输出 (BUZZER)

对 TC1 时钟频率进行适当设置可得到特定频率的蜂鸣器输出（TC1OUT），并通过引脚 P5.3 输出。单片机内部设置 TC1 的溢出频率经过 2 分频后作为 TC1OUT 的频率，即 TC1 每溢出 2 次 TC1OUT 输出一个完整的脉冲，此时，P5.3 的 I/O 功能自动被禁止。TC1OUT 输出波形如下：



➢ **例：设置 TC0OUT (P5.3)。其中，Fcpu = 4MHz，TC1OUT = 1KHz，TC1 = 2KHz，TC1 时钟源采用内部时钟 Fcpu/4，TC1RATE2~TC1RATE1 = 110， TC1C = TC1R = 131，TC1X8=1。**

```
B0BCLR      FTC1X8          ;
MOV         A,#01100000B
B0MOV       TC1M,A          ; TC1 速率 Fcpu/4。

MOV         A,#131          ; 自动装载参考值设置。
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 自动重装功能。
B0BSET      FTC1ENB         ; 开启 TC1 定时器。
```

注：TC1OUT 的频率表和 TC0OUT 的频率表类似，请参考 TC0OUT 的频率表。

8.4 PWM 功能说明

8.4.1 概述

PWM 信号通过 PWM0OUT（P5.4）/PWM1OUT（P5.3）输出。8 位计数器 TC0C/TC1C 计数过程中不断与 TC0R/TC1R 相比较，当 TC0C/TC1C 的值增加到与 TC0R/TC1R 相等时，PWM 输出低电平；当 TC0C/TC1C 的值溢出重新回到 0 时，PWM 被强制输出高电平。

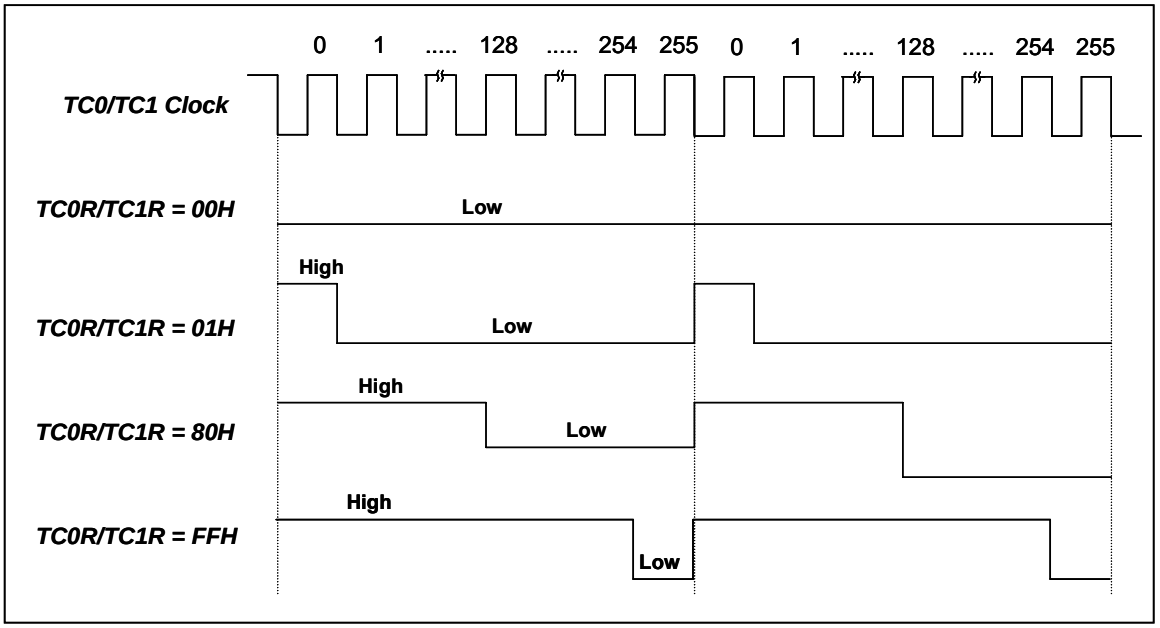
PWM 的初始输出值是低电平，若计数器溢出（如 TC0C 由 FFH 到 00H），PWM 则输出高电平。脉冲的宽度比例（占空比）由 TC0R/TC1R 寄存器控制，溢出边界则由 ALOAD0/ALOAD1 和 TC0OUT/TC1OUT 位控制。向参考寄存器 TC0R/TC1R 写入 00H 可以使 PWM 输出保持低电平；而连续写入 FFH 到 TC0R 则可以使 PWM 输出保持高电平，除了时钟源的最后一个脉冲是初始低电平。

PWM0OUT = 1, TC0X8=0

ALOAD0 ALOAD1	TC0OUT TC1OUT	TC0 溢出边界 TC1 溢出边界	PWM 占空比范围	PWM 的最大频率 （Fcpu = 4M）	备注
0	0	FFH ~ 00H	0/256 ~ 255/256	7.8125K	每计数 256 次溢出
0	1	3FH ~ 40H	0/64 ~ 63/64	31.25K	每计数 64 次溢出
1	0	1FH ~ 20H	0/32 ~ 31/32	62.5K	每计数 32 次溢出
1	1	0FH ~ 10H	0/16 ~ 15/16	125K	每计数 16 次溢出

ALOAD0 ALOAD1	TC0OUT TC1OUT	TC0R TC1R	PWM 占空比范围
0	0	00000000 ~ 11111111	0/256 ~ 255/256
0	1	XX000000 ~ XX111111	0/64 ~ 63/64
1	0	XXX00000 ~ XXX11111	0/32 ~ 31/32
1	1	XXXX0000 ~ XXXX1111	0/16 ~ 15/16

✷ 注：若使能 **PWM0OUT 或 TC0OUT** 功能，**P5.4** 自动被设为输出模式；
若禁止 **PWM0OUT 或 TC0OUT** 功能，**P5.4** 的模式则由 **P54M** 定义。



8.4.2 PWM 编程举例

➢ **例：PWM0 输出设置，由 PWM0OUT(P5.4)输出。Fcpu = 4MHZ，PWM 输出占空比＝30/256，输出频率 1KHZ，PWM 时钟源来自外部时钟，TC0 速率＝Fcpu/4，TC0RATE2~TC0RATE1 = 110，TC0C = TC0R = 30。**

```
B0BCLR      FTC0X8          ;
B0BCLR      FTC1X8          ;
MOV         A,#01100000B
B0MOV       TC0M,A          ; TC0 速率 = Fcpu/4。
B0MOV       TC0M,A          ;
MOV         A,#0x00         ; 初始化 TC0。

MOV         A,#30           ; PWM 输出占空比 = 30/256。
B0MOV       TC0R,A

B0BCLR      FTC0OUT         ; 禁止 TC0OUT 功能。
B0BSET      FPWM0OUT        ; PWM0 输出至 P5.4，禁止 P5.4 I/O 功能。
B0BSET      FTC0ENB         ; 开启 TC0 定时器。
```

✸ **注 1：TC0R/TC1R 为只写寄存器，不能用 INCMS 和 DECMS 指令对其进行操作。**
✸ **注 2：首先初始化 TC0C 保证占空比正确。**

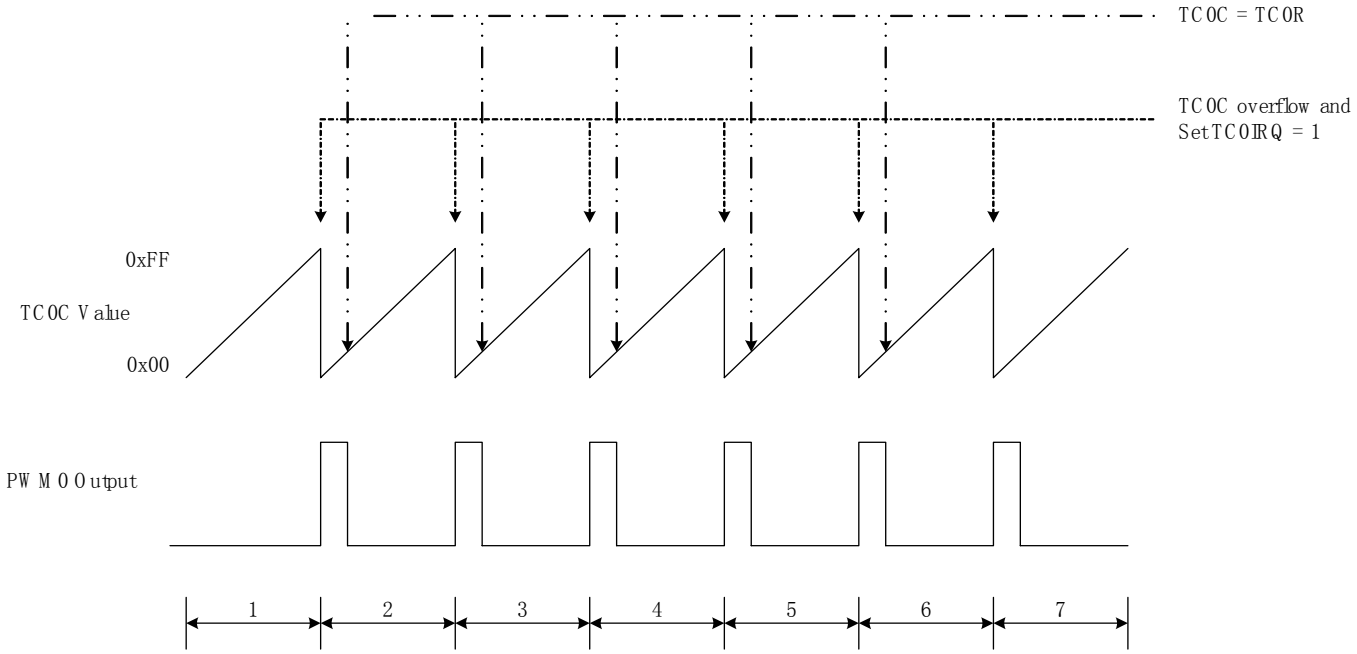
➢ **例：调整 TC0R/TC1R 的值。**

```
MOV         A, #30H         ;
B0MOV       TC0R, A

INCMS       BUF0            ;
B0MOV       A, BUF0         ;
B0MOV       TC0R, A
```

✸ **注 3：PWM0/PWM1 的占空比改变时，同时改变 TC0C/TC1C 和 TC0R/TC1R 的值，以避免 PWM0/PWM1 信号受到干扰。**
✸ **注 4：使能 PWM0/PWM1 的输出功能时，TC0OUT/TC1OUT 必须置“0”。**
✸ **注 5：PWM 可以在中断下工作。**

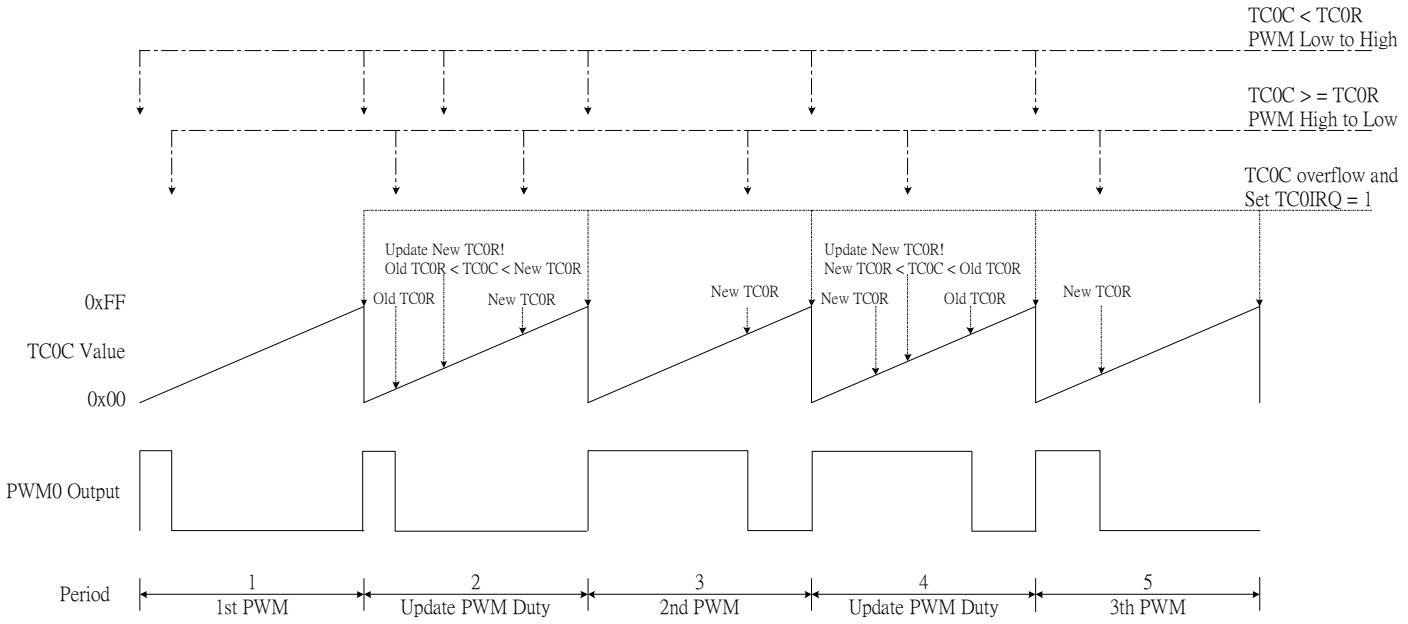
8.4.3 PWM 占空比与 TCxR 的值的改变

PWM 模式下，系统会随时比较 TCXC 和 TCXR 的值：当 TCxC<TCxR 时，PWM 输出高电平；当 TCxC≧TCxR 时，则输出低电平。当 TCXC 的值发生改变时，PWM 的占空比就会随着改变。若 TCXR 的值保持恒定，则 PWM 的波形也保持稳定。



上图所示是 TCXR 的值恒定时的波形。当 TCXC 溢出时，PWM 输出高电平；TCxC≧TCxR，PWM 输出低电平。

下图是 TCXR 发生变化时对应的波形图：



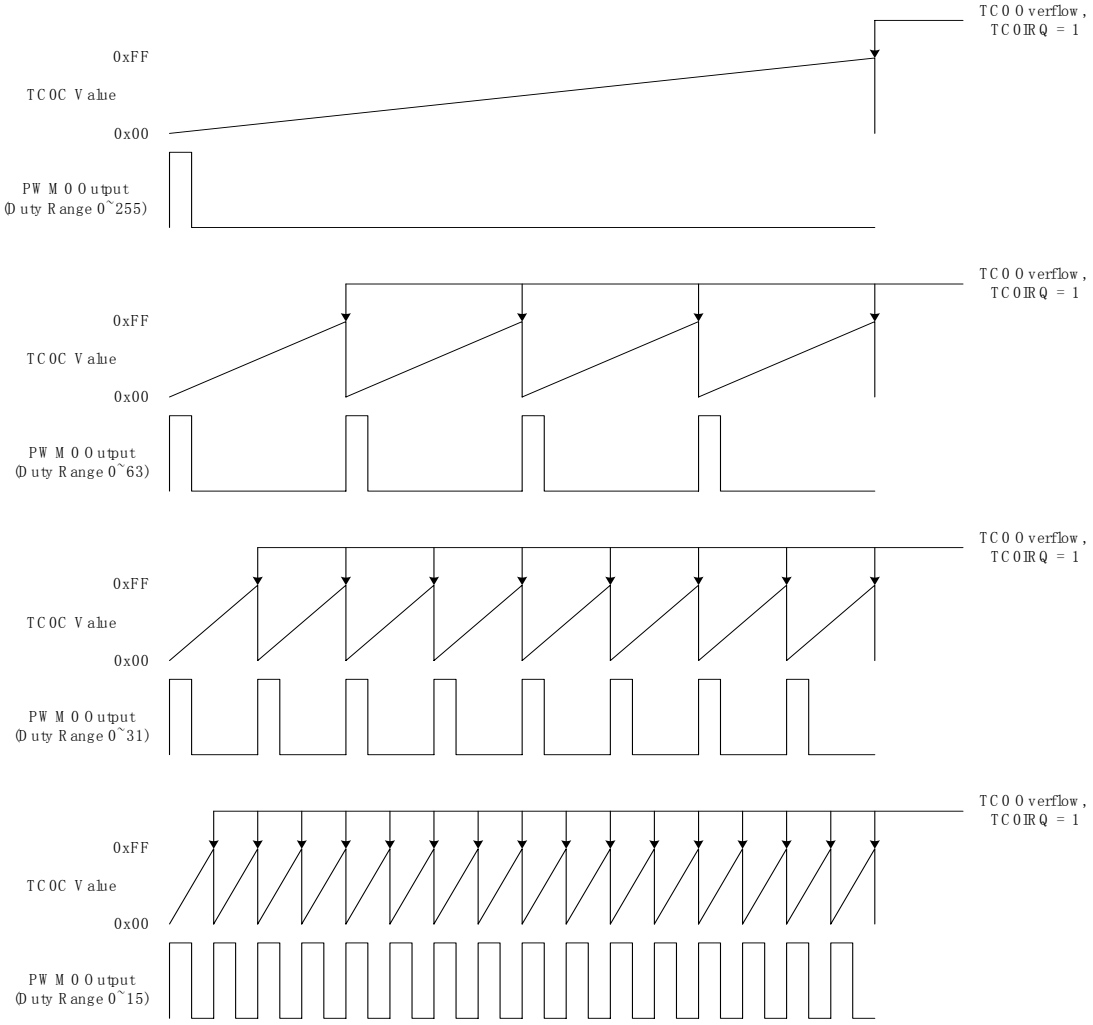
在 period 2 和 period 4 中，设置新的占空比（TC0R），但 PWM 在 period 2 和 period 4 的占空比要在下一个 period 才会改变。这样，可以避免 PWM 不随设定改变或在同一个周期内改变两次，从而避免系统发生不可预知的误动作。

8.4.4 TCxIRQ 和 PWM 占空比

PWM 模式下，PWM 占空比的范围决定 TCXIRQ 的频率。

ALOADx	TCxOUT	TCx 溢出边界	PWM 占空比范围	TCxIRQ 频率
0	0	FFh ~ 00h	0/256 ~ 255/256	TCx clock / 256
0	1	3Fh ~ 40h	0/64 ~ 63/64	TCx clock / 64
1	0	1Fh ~ 20h	0/32 ~ 31/32	TCx clock / 32
1	1	0Fh ~ 10h	0/16 ~ 15/16	TCx clock / 16

TCXIRQ 和 PWM 占空比的关系如下图所示：



9 中断

9.1 概述

SN8P2710 提供 4 个中断源：2 个内部中断（TC0/TC1）和 2 个外部中断（INT0/INT1）。外部中断的输入引脚 INT0/INT1 和 P0.0/P0.1 引脚共用，外部中断可以将系统从睡眠模式唤醒进入普通模式。一旦程序进入中断，寄存器 STKP 的位 GIE 将被硬件自动清零以避免再次响应其它中断。系统退出中断，即执行完 RETI 指令后，硬件自动将 GIE 置“1”，以响应下一个中断。中断请求存放在寄存器 INTRQ 中，用户可以自行设置中断的优先权。

✸ **注：程序响应中断时，GIE 必须处于有效状态。**

9.2 中断使能寄存器 INTEN

中断请求控制寄存器 INTEN 包括所有中断的使能控制位。INTEN 的有效位被置为“1”，则系统进入该中断服务程序，程序计数器入栈，程序转至 0008H 即中断程序。程序运行到指令 RETI 时，中断结束，系统退出中断服务。

INTEN = 0000 0000

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	-	TC1IEN	TC0IEN	-	-	-	P01IEN	P00IEN
	-	R/W	R/W	-	-	-	R/W	R/W

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

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

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

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

9.3 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。

INTRQ = x00x xx00

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	-	TC1IRQ	TC0IRQ	-	-	-	P01IRQ	P00IRQ
	-	R/W	R/W	-	-	-	R/W	R/W

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

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

Bit 1 **P01IRQ**：P0.1 中断（INT1）请求标志位。
0 = INT1 无中断请求；
1 = INT1 有中断请求。

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

9.4 P0.0 中断触发边沿控制寄存器

PEDGE = xxx1 0xxx

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

Bit [4:3] **P00G [1:0]**：P0.0 中断触发控制位。
00 = 保留；
01 = 上升沿；
10 = 下降沿（**复位后为默认值**）；
11 = 下降沿和上升沿都有效（电平变化触发）。

9.5 中断操作说明

SN8P2710 共有 4 个中断，各中断的详细操作如下：

9.5.1 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（0008H），堆栈层数加 1。

STKP = 0xxx 1111

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

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。**

9.5.2 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。

➢ **例：INT0 中断设置。**

```
B0BSET      FP00IEN     ; 使能 INT0 中断。
B0BCLR      FP00IRQ     ; 清 P00IRQ。
B0BSET      FGIE        ; 使能 GIE。
```

➢ **例：INT0 中断。**

```
        ORG         8H          ;
        JMP         INT_SERVICE
INT_SERVICE:

        B0XCH       A, ACCBUF   ; 保存 ACC。
        B0MOV       A, PFLAG
        B0MOV       PFLAGBUF, A ; 保存 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:
        B0MOV       A, PFLAGBUF
        B0MOV       PFLAG, A    ; 恢复 PFLAG。
        B0XCH       A, ACCBUF   ; 恢复 ACC。

        RETI                    ; 中断返回。
```

9.5.3 INT1（P0.1）中断

INT1 由下降沿触发。触发后，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。

➢ **例：INT1 中断请求设置。**

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

➢ **例：INT1 中断服务程序。**

```
        ORG         8H                  ;
        JMP         INT_SERVICE
INT_SERVICE:

        B0XCH       A, ACCBUF           ; 保存 ACC。
        B0MOV       A, PFLAG
        B0MOV       PFLAGBUF, A         ; 保存 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:
        B0MOV       A, PFLAGBUF
        B0MOV       PFLAG, A            ; 恢复 PFLAG。
        B0XCH       A, ACCBUF           ; 恢复 ACC。

        RETI                            ; 中断返回。
```

9.5.4 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。

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

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：TC0 中断服务程序。**

```
        ORG         8H              ;
        JMP         INT_SERVICE
INT_SERVICE:

        B0XCH       A, ACCBUF       ; 保存 ACC。
        B0MOV       A, PFLAG
        B0MOV       PFLAGBUF, A     ; 保存 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 中断服务程序。
        .               .
EXIT_INT:
        B0MOV       A, PFLAGBUF
        B0MOV       PFLAG, A        ; 恢复 PFLAG。
        B0XCH       A, ACCBUF       ; 恢复 ACC。

        RETI                        ; 中断返回。
```

9.5.5 TC1 中断

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

➢ **例：TC1 中断请求设置。**

```
        B0BCLR      FTC1IEN         ; 禁止 TC1 中断。
        B0BCLR      FT C1ENB        ;
        MOV         A, #20H         ;
        B0MOV       TC1M, A         ; TC1 时钟= Fcpu / 64。
        MOV         A, #64H         ; TC1C 初始值=64H 。
        B0MOV       TC1C, A         ; TC1 间隔= 10 ms。

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

        B0BSET      FGIE            ; 使能 GIE。
```

➢ **例：TC1 中断服务程序。**

```
        ORG         8H              ;
        JMP         INT_SERVICE
INT_SERVICE:

        B0XCH       A, ACCBUF       ; 保存 ACC。
        B0MOV       A, PFLAG
        B0MOV       PFLAGBUF, A     ; 保存 PFLAG。

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

        B0BCLR      FTC1IRQ         ; 清 TC1IRQ。
        MOV         A, #74H
        B0MOV       TC1C, A         ; 清 TC1C.。
        .           .               ; TC1 中断服务程序。
        .           .
EXIT_INT:
        B0MOV       A, PFLAGBUF
        B0MOV       PFLAG, A        ; 恢复 PFLAG。
        B0XCH       A, ACCBUF       ; 恢复 ACC。

        RETI                        ; 中断返回。
```

9.5.6 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	下降沿触发
P01IRQ	下降沿触发
TC0IRQ	TC0C 溢出
TC1IRQ	TC1C 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。**

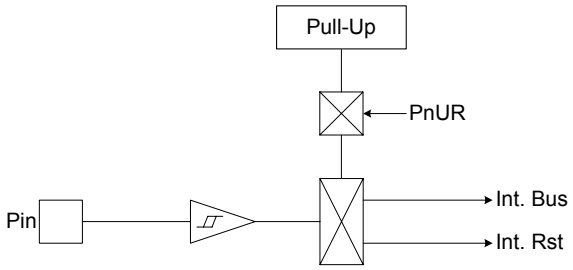
```
        ORG     8H                  ;
        NOP
        B0XCH   A, ACCBUF           ; 保存 ACC。
        B0MOV   A, PFLAG
        B0MOV   PFLAGBUF, A         ; 保存 PFLAG。
INTP00CHK:                          ; 检查是否有 INT0 中断请求。
        B0BTS1  FP00IEN             ; 检查是否使能 INT0 中断。
        JMP     INTP01CHK           ; 跳转到下一个中断。
        B0BTS0  FP00IRQ             ; 检查是否有 INT0 中断请求。
        JMP     INTP00              ; 跳转到 INT0 中断服务程序。
INTP01CHK:                          ; 检查是否有 INT1 中断请求。
        B0BTS1  FP01IEN             ; 检查是否使能 INT1 中断。
        JMP     INTTC0CHK           ; 跳转到下一个中断。
        B0BTS0  FP01IRQ             ; 检查是否有 INT1 中断请求。
        JMP     INTP01              ; 跳转到 INT1 中断服务程序。
INTTC0CHK:                          ; 检查是否有 TC0 中断请求。
        B0BTS1  FTC0IEN             ; 检查是否使能 TC0 中断。
        JMP     INTTC1CHK           ; 跳转到下一个中断。
        B0BTS0  FTC0IRQ             ; 检查是否有 TC0 中断请求。
        JMP     INTTC0              ; 跳转到 TC0 中断服务程序。
INTTC1HK:                           ; 检查是否有 TC1 中断请求。
        B0BTS1  FTC1IEN             ; 检查是否使能 TC1 中断。
        JMP     INT_EXIT            ;
        B0BTS0  FTC1IRQ             ; 检查是否有 TC1 中断请求。
        JMP     INTTC1              ; 跳转到 TC1 中断服务程序。
INT_EXIT:
        B0MOV   A, PFLAGBUF
        B0MOV   PFLAG, A            ; 恢复 PFLAG。
        B0XCH   A, ACCBUF           ; 恢复 ACC。

        RETI                        ; 中断返回。
```

10 I/O 口

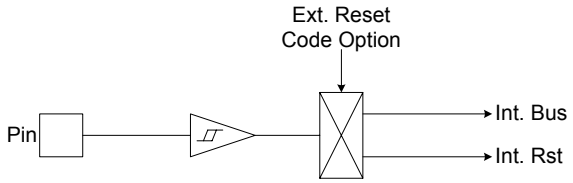
10.1 概述

SN8P2710 为用户提供 4 个 I/O 端口：一个输入端口 P0，3 个输入/输出端口（P2、P4 和 P5）。PnM 寄存器控制 I/O 的模式，寄存器 PnUR 则用来设置上拉电阻。系统复位后，所有的 I/O 都默认为无上拉电阻的输入模式。

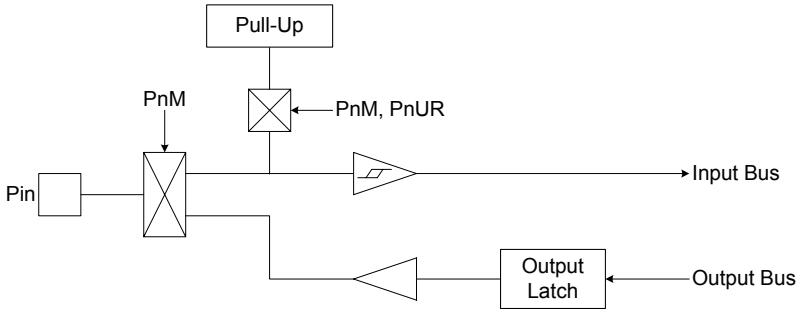
P0.1 和 P0.2：



P0.3：



P2, 5：



P4：

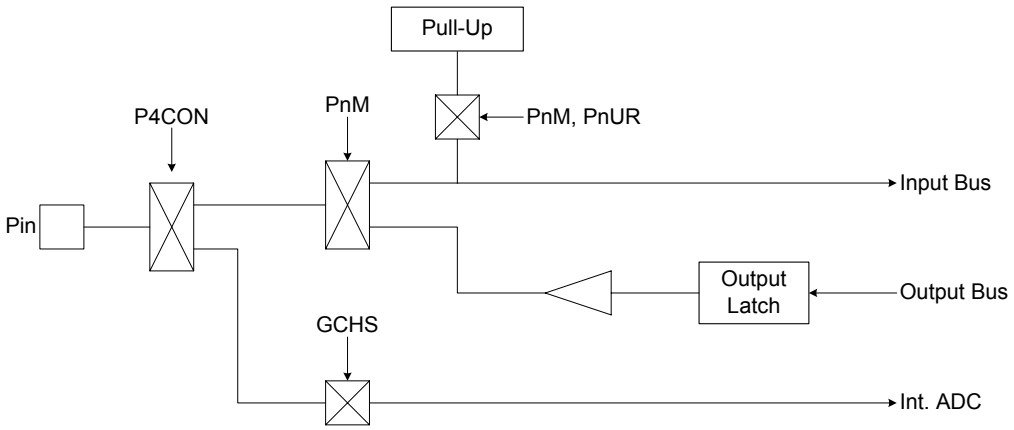


图 10-1. I/O 口电路简图

10.2 I/O 口功能表

端口/引脚	I/O	功能说明	备注
P0.0~P0.1	I	普通输入功能	P0.0：TC0 外部时钟输入引脚 P0.1：TC1 外部时钟输入引脚
		外部中断（INT0~INT1）	
		系统睡眠模式唤醒功能	
P0.2	I	普通输入功能	
		系统睡眠模式唤醒功能	
P0.3	I	普通输入功能	无上拉电阻，无唤醒功能
		和复位引脚共用	
P2.0~P2.7	I/O	普通输入/输出功能	
P4.0~P4.7	I/O	普通输入/输出功能	
		ADC 模拟信号输入引脚	
P5.0~P5.6	I/O	普通输入/输出功能	

10.3 上拉电阻寄存器

上拉电阻寄存器可以控制 I/O 口的上拉状态，上拉电阻的典型值是 200K@3V 和 100K@5V。

P0

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

P2

0E2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2UR	P27r	P26R	P25R	P24R	P23R	P22R	P21R	P20R
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

P4

0E4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4UR	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

P5

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

➢ **例：I/O 口的上拉电阻。**

```
CLR     P0UR        ; 禁止 P0 的上拉电阻。

MOV     A, #01H     ;
B0MOV   P0UR, A     ; 使能 P0 的上拉电阻。
```

10.4 I/O 口模式寄存器

寄存器 PnM 控制 I/O 口的工作模式。P0 是单向输入端口，P2、P4、P5 是双向输入输出端口。

P2M = 0000 0000

0C2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2M	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit [7:0] **P2 [7:0] M**：P2.0~P2.7 模式控制位。
0 = 输入模式；
1 = 输出模式。

P4M = 0000 0000

0C4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4M	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit [7:0] **P4 [7:0] M**：P4.0~P4.7 模式控制位。
0 = 输入模式；
1 = 输出模式。

P5M = x000 0000

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	P56M	P55M	P54M	P53M	P52M	P51M	P50M
	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit [6:0] **P5 [6:0] M**：P5.0~P5.6 模式控制位。
0 = 输入模式；
1 = 输出模式。

➢ **例：I/O 模式选择。**

```
CLR     P2M             ; 设置为输入模式。
CLR     P4M
CLR     P5M


MOV     A, #0FFH        ; 设置为输出模式。
B0MOV   P2M, A
B0MOV   P4M, A
B0MOV   P5M, A

B0BCLR  P2M.5           ; 设置 P2.5 为输入模式。

B0BSET  P2M.5           ; 设置 P2.5 为输出模式。
```

10.5 I/O 口数据寄存器

P0 = xxxx xxxx

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	-	-	-	-	P03	P02	P01	P00
	-	-	-	-	R	R	R	R

P2 = xxxx xxxx

0D2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2	P27	P26	P25	P24	P23	P22	P21	P20
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P4 = xxxx xxxx

0D4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4	P47	P46	P45	P44	P43	P42	P41	P40
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

P5 = xxxx xxxx

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	-	P56	P55	P54	P53	P52	P51	P50
	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W

➢ **例：从输入端口读取数据。**

```
B0MOV   A, P0     ; 读取 P0 的数据。
B0MOV   A, P2     ; 读取 P2 的数据。
B0MOV   A, P4     ; 读取 P4 的数据。
B0MOV   A, P5     ; 读取 P5 的数据。
```

➢ **例：写入数据到输出端口。**

```
MOV     A, #55H   ; 写入数据 55H 到 P2，P4，P5。
B0MOV   P2, A
B0MOV   P4, A
B0MOV   P5, A
```

➢ **例：写 1 位数据到输出端口。**

```
B0BSET  P2.3      ; 置 P2.3 和 P4.0 为 1。
B0BSET  P4.0

B0BCLR  P2.3      ; 置 P2.3 和 P4.0 为 0。
B0BCLR  P5.5
```

➢ **例：位检测。**

```
B0BTS1  P0.0      ; 检测 P0.0 是否为 1。
.
B0BTS0  P2.5      ; 检测 P2.5 是否为 0。
```

11 8 通道 ADC

11.1 概述

SN8P2710 系列最多可以提供 8 个通道、4096 阶分辨率的 A/D 转换器，ADC 可以将模拟信号转换成相应的 12 位数字信号。进行 AD 转换时，首先要选择输入通道（AIN0~AIN7），然后将 GCHS 和 ADS 置 1，并启动 ADC；转换结束后，系统自动将 EOC 置 1，并将转换结果存入寄存器 ADB 中。

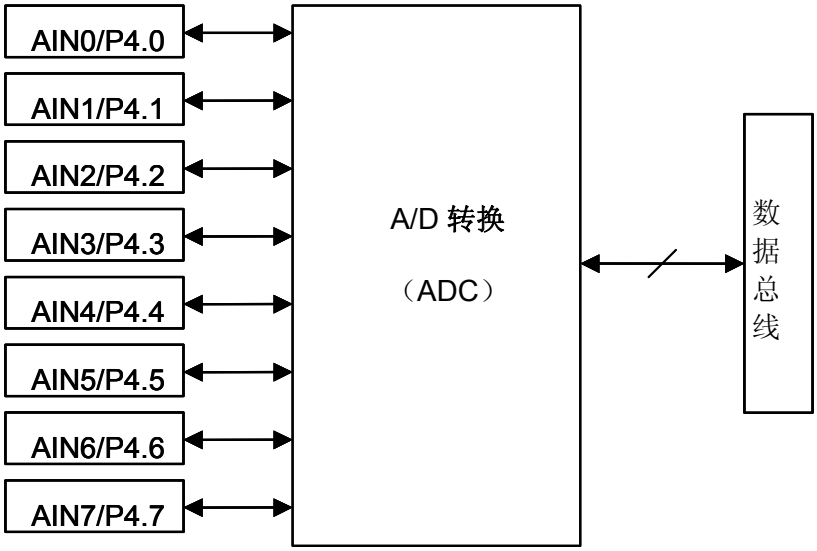


图 11-1. ADC 功能结构简图

- **注：模拟输入电压值必须处于 AVREFH 和 AVREFL 之间；**
- **注：AVREFH 的值必须处于 AVDD 和 VSS + 2.0V 之间；**
- **注：ADC 设计时应注意：**
  1. **ADC 的 I/O 引脚设置为输入模式；**
  2. **禁止 ADC 输入引脚的上拉电阻；**
  3. **进入睡眠模式前禁止 ADC 以省电；**
  4. **在睡眠模式下设置 P4CON 寄存器的有关位以避免额外的功耗；**
  5. **启动 ADC（ADENB = 1）后延迟 100us 等待 ADC 电路稳定。**

11.2 ADM 寄存器

ADM = 0000 x000

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
	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W

Bit 7 **ADENB**：ADC 控制位。
0 = 禁止；
1 = 使能。

Bit 6 **ADS**：ADC 启动位 。
0 = 停止；
1 = 转换开始。

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**：ADC 输入通道控制位。
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；110 = AIN6；111 = AIN7。

11.3 ADR 寄存器

ADR = x00x 0000

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR		ADCKS1		ADCKS0	ADB3	ADB2	ADB1	ADB0
		R/W		R/W	R	R	R	R

Bit 6,4 **ADCKS [1:0]**：ADC 分频选择位。

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

Bit [3:0] **ADB [3:0]**：ADC 数据缓存器。
12 位 ADC：ADB11~ADB0。

11.4 ADB 寄存器

ADB = xxxx xxxx

0B2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADB	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4
	R	R	R	R	R	R	R	R

8 位数据缓存器 ADB 用来保存 AD 转换结果的高 8 位（bit4~bit11），转换结果的低 4 位则保存在 ADR 寄存器中。ADB 为只读寄存器。

✸ **注：刚上电时，ADB [0:11]的值是未知的。**

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
0/4095*AVREFH	0	0	0	0	0	0	0	0	0	0	0	0
1/4095*AVREFH	0	0	0	0	0	0	0	0	0	0	0	1
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.
4094/4095*AVREFH	1	1	1	1	1	1	1	1	1	1	1	0
4095/4095*AVREFH	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 精度	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												

11.5 P4CON 寄存器

P4 口和 ADC 的输入口共享。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。

ADB = 0000 0000

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

Bit[7:0] **P4CON[7:0]**：P4.n 配置控制位。
0 = P4.n 作为模拟输入（ADC 输入）引脚或者普通 I/O 引脚；
1 = P4.n 只能作为模拟输入引脚，此时禁止 P4.n 的普通 I/O 功能。

✸ **注：当 P4.n 为基本 I/O 而不是 ADC 通道时，P4CON.n 必须置“0”，否则 P4.n 的数字 I/O 信号会被隔离。**

11.6 AD 转换时间

12 位 AD 转换时间 = 1/（ADC 时钟 / 4 ）*16 sec

Fosc = 4MHz

Fcpu		ADCKS1	ADCKS0	ADC 时钟	AD 转换时间
Fosc/	1	0	0	Fcpu/16	1/((4MHz / 1) / 16 /4) x16= 256 us
		0	1	Fcpu/8	1/((4MHz / 1) / 8 /4) x16= 128 us
		1	0	Fcpu	1/((4MHz / 1) / 1 /4) x16= 16 us
		1	1	Fcpu/2	1/((4MHz / 1) / 2 /4) x16= 32 us
Fosc/	2	0	0	Fcpu/16	1/((4MHz / 2) / 16 /4) x16= 512 us
		0	1	Fcpu/8	1/((4MHz / 2) / 8 /4) x16= 256 us
		1	0	Fcpu	1/((4MHz / 2) / 1 /4) x16= 32 us
		1	1	Fcpu/2	1/((4MHz / 2) / 2 /4) x16= 64 us
Fosc/	4	0	0	Fcpu/16	1/((4MHz / 4) / 16 /4) x16= 1024 us
		0	1	Fcpu/8	1/((4MHz / 4) / 8 /4) x16= 512 us
		1	0	Fcpu	1/((4MHz / 4) / 1 /4) x16= 64 us
		1	1	Fcpu/2	1/((4MHz / 4) / 2 /4) x16= 128 us
Fosc/	8	0	0	Fcpu/16	1/((4MHz / 8) / 16 /4) x16= 2048 us
		0	1	Fcpu/8	1/((4MHz / 8) / 8 /4) x16= 1024 us
		1	0	Fcpu	1/((4MHz / 8) / 1 /4) x16= 128 us
		1	1	Fcpu/2	1/((4MHz / 8) / 2 /4) x16= 256 us

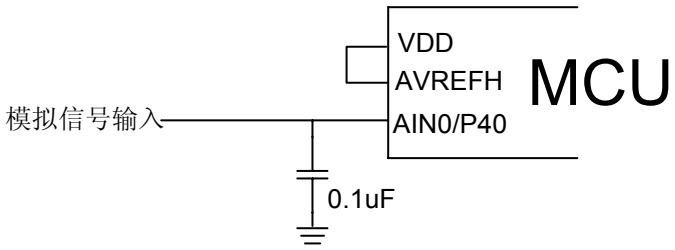
✸ 注：**ADC** 也可以在低速模式下运行。低速模式下，**Fcpu = LXOSC/4**（**LXOSC** 是内部低速 **RC** 振荡器）。

✸ 注：由于 **LXOSC** 的频率随着温度和 **VDD** 的变化而变化，因此 **AD** 转换时间也会因此而不同。

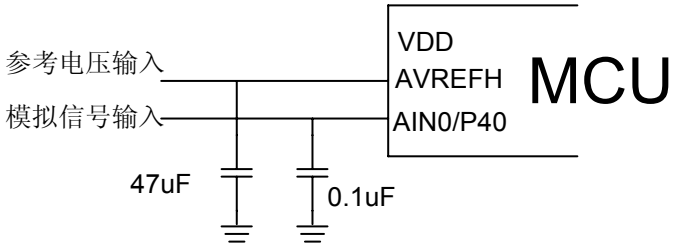
➢ **例：设置 AIN0 为 12 位 ADC 的输入通道，ADC 结束后进入睡眠模式。**

```
ADC0:
        B0BSET      FADENB          ; 使能 ADC 电路。
        CALL        Delay100uS      ; 延迟 100us 等待 ADC 电路稳定。
        MOV         A, #0FEh
        B0MOV       P4UR, A         ; 禁止 P4.0 的上拉电阻。
        B0BCLR      FP40M           ; 设置 P4.0 为输入模式。
        MOV         A, #01h
        B0MOV       P4CON, A        ; 设置 P4.0 为 ADC 输入模式。
        MOV         A, #40H
        B0MOV       ADR, A          ; 设置 12 位 ADC，ADC 时钟源 = Fosc。
        MOV         A,#90H
        B0MOV       ADM,A           ; 使能 ADC 并选择通道 AIN0。
        B0BSET      FADS            ; 开始转换。
WADC0:
        B0BTS1      FEOC            ; 判断转换是否结束。
        JMP         WADC0           ;
        B0MOV       A,ADB           ; 读取 AIN0 的高 8 位输入数据。
        B0MOV       Adc_Buf_Hi, A
        B0MOV       A,ADR           ; 读取 AIN0 的低 4 位输入数据。
        AND         A, 0Fh
        B0MOV       Adc_Buf_Low, A
Power_Down
        .           .
        B0BCLR      FADENB          ; 关闭 AD 转换。
        B0BSET      FCPUM0          ; 进入睡眠模式。
```

11.7 ADC 电路



AVREFH 连接到 VDD.



AVREFH 连接到外部 AD 参考电压

图 11-2. AD 转换的 AINx 和 AVREFH 电路

✸ 注：**AIN 和 GND 之间的旁路电容有助于模拟信号的稳定。**

12 7 位 DAC

12.1 概述

D/A 转换器是将 7 位数字信号转换成对应的 128 阶电流型模拟信号输出。当 DAENB 置“1”后，开启 DA 转换电路，DAM 寄存器中的 bit0~bit6 位通过梯形电阻网络被转换成相应的模拟信号并由 DAO 引脚输出。

图 12-1. DAC 结构简图

为了得到合适的线性输出信号，通常在 DAO 和 GND 之间接一个负载电阻，下面给出了 Vdd = 5V/R_L = 150ohm 和 Vdd = 3V/ R_L =150ohm 时的曲线图。

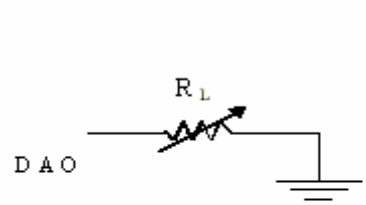


图 12-2 带 R_L 的 DAO 电路

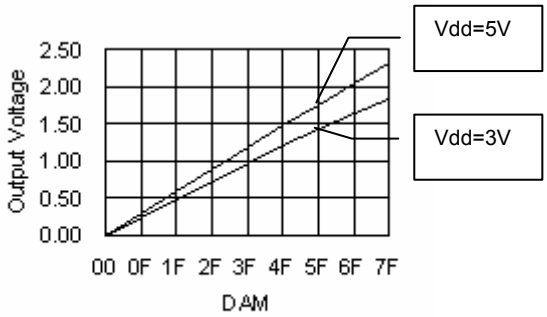


图 12-3. DAC 输出电压（Vdd=5V / 3V）

本 D/A 转换器的设计不适合用作精确的 DC 电压输出，只适合于简单的音频应用。

12.2 DAM 寄存器

DAM = 0000 0000

0B0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DAM	DAENB	DAB6	DAB5	DAB4	DAB3	DAB2	DAB1	DAB0
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Bit 7 **DAENB**：DAC 控制位。
0 = 禁止；
1 = 使能。

Bit [6:0] **DAB [6:0]**：数字输入数据。

12.3 D/A 转换说明

当 DAENB = 0 时，DAO 的输出不稳定状态，设置 DAENB = 1 后，DAO 的输出值就由 DAB 决定。

- **例：DAO 引脚输出 1/2 VDD。**

```
MOV         A, #00111111B
B0MOV       DAM, A          ;

B0BSET      FDAENB          ; 使能 DAC。
```

DAB 的数据 v.s.DAO 的输出电压：

DAB6	DAB5	DAB4	DAB3	DAB2	DAB1	DAB0	DAO
0	0	0	0	0	0	0	VSS
0	0	0	0	0	0	1	Idac
0	0	0	0	0	1	0	2 * Idac
0	0	0	0	0	1	1	3 * Idac
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
1	1	1	1	1	1	0	126 * Idac
1	1	1	1	1	1	1	127 * Idac

注：**Idac = I_{FSO} / (2^7-1)** **(I_{FSO}：全程输出量)**

13 编程

13.1 编程模板

```
;*************************************************************************
; FILENAME    : TEMPLATE.ASM
; AUTHOR      : SONiX
; PURPOSE     : Template Code for SN8P2710
; REVISION    : V1.0      First issue
;*************************************************************************
;*   (c) Copyright 2004, SONiX TECHNOLOGY CO., LTD.
;*************************************************************************
CHIP      SN8P2715                          ; Select the CHIP

;-------------------------------------------------------------------------
;                                  Include Files
;-------------------------------------------------------------------------
.nolist                                     ; do not list the macro file
      INCLUDESTD  MACRO1.H
      INCLUDESTD  MACRO2.H
      INCLUDESTD  MACRO3.H

.list                                       ; Enable the listing function

;-------------------------------------------------------------------------
;                                  Constants Definition
;-------------------------------------------------------------------------
;     ONE           EQU       1
;-------------------------------------------------------------------------
;                                  Variables Definition
;-------------------------------------------------------------------------
.DATA
                    org       0h            ;Bank 0 data section start from RAM address 0x000
      Wk00B0        DS        1             ;Temporary buffer for main loop
      Iwk00B0       DS        1             ;Temporary buffer for ISR
      AccBuf        DS        1             ;Accumulater buffer
      PflagBuf   DS        1                ;PFLAG buffer

                    org       100h          ;Bank 1 data section start from RAM address 0x100
      BufB1         DS        20            ;Temporary buffer in bank 1

;-------------------------------------------------------------------------
;                                  Bit Flag Definition
;-------------------------------------------------------------------------
      Wk00B0_0      EQU       Wk00B0.0      ;Bit 0 of Wk00B0
      Iwk00B0_1     EQU       Iwk00B0.1     ;Bit 1 of Iwk00

;-------------------------------------------------------------------------
;                                  Code section
;-------------------------------------------------------------------------
.CODE

      ORG           0                       ;Code section start
      jmp           Reset                   ;Reset vector
                                            ;Address 4 to 7 are reserved
      ORG           8
      jmp           Isr                     ;Interrupt vector

      ORG           10h
;-------------------------------------------------------------------------
;     Program reset section
;-------------------------------------------------------------------------
Reset:
      mov           A,#07Fh                 ;Initial stack pointer and
      b0mov         STKP,A                  ;disable global interrupt

      clr           PFLAG                   ;pflag = x,x,x,x,x,c,dc,z
      mov           A,#00h                  ;Initial system mode
```

```
        b0mov       OSCM,A

        mov         A, #0x5A
        b0mov       WDTR, A             ;Clear watchdog timer

        call        ClrRAM              ;Clear RAM
        call        SysInit             ;System initial
        b0bset      FGIE                ;Enable global interrupt

;-------------------------------------------------------------------------------
;       Main routine
;-------------------------------------------------------------------------------
Main:
        mov         A, #0x5A            ;Clear watchdog timer
        b0mov       WDTR, A

         call       MnApp

        jmp         Main
;-------------------------------------------------------------------------------
;       Main application
;-------------------------------------------------------------------------------
MnApp:

        ; Put your main program here
        ret

;-----------------------------------
;       Jump table routine
;-----------------------------------
        ORG         0x0100              ;The jump table should start from the head
                                        ;of boundary.
        b0mov       A,Wk00B0
        and         A,#3
         ADD        PCL,A
        jmp         JmpSub0
        jmp         JmpSub1
        jmp         JmpSub2
;-----------------------------------

JmpSub0:
        ; Subroutine 1
        jmp         JmpExit
JmpSub1:

        ; Subroutine 2
        jmp         JmpExit

JmpSub2:
        ; Subroutine 3
        jmp         JmpExit

JmpExit:
        ret                             ;Return Main

;-------------------------------------------------------------------------------
; Isr (Interrupt Service Routine)
; Arguments     :
; Returns  :
; Reg Change:
;-------------------------------------------------------------------------------
Isr:
;-----------------------------------
;       Save ACC and system registers
;-----------------------------------
        b0xch       A,AccBuf            ;B0xch instruction do not change C,Z flag
        B0MOV       A, PFLAG
        B0MOV       PFLAGBUF, A

;-----------------------------------
;     Check which interrupt happen
;-----------------------------------
```

```
IntP00Chk:
        b0bts1      FP00IEN
        jmp         IntTc0Chk       ;Modify this line for another interrupt
         b0bts0     FP00IRQ
        jmp         P00isr

        ;If necessary, insert another interrupt checking here

IntTc0Chk:
        b0bts1      FTC0IEN
        jmp         IsrExit         ;Suppose TC0 is the last interrupt which you
        b0bts0      FTC0IRQ         ;want to check
        jmp         TC0isr
;----------------------------------
; Exit interrupt service routine
;----------------------------------
IsrExit:
        B0MOV       A, PFLAGBUF                     ;
        B0MOV       PFLAG, A
        b0xch       A,AccBuf        ;B0xch instruction do not change C,Z flag

        reti                        ;Exit the interrupt routine

;--------------------------------------------------------------------------
;       INT0 interrupt service routine
;--------------------------------------------------------------------------
P00isr:
        b0bclr      FP00IRQ

        ;Process P0.0 external interrupt here

        jmp         IsrExit

;--------------------------------------------------------------------------
;       TC0 interrupt service routine
;--------------------------------------------------------------------------
TC0isr:
        b0bclr      FTC0IRQ

        ;Process TC0 timer interrupt here

        jmp         IsrExit

;--------------------------------------------------------------------------
;       SysInit
;       Initialize I/O, Timer, Interrupt, etc.
;--------------------------------------------------------------------------
SysInit:
        ret

;--------------------------------------------------------------------------
; ClrRAM
; Use index @YZ to clear RAM    (00h~7Fh)
;--------------------------------------------------------------------------

ClrRAM:

; RAM Bank 0
        clr         Y                   ;Select bank 0
        b0mov       Z,#0x7f             ;Set @YZ address from 7fh

ClrRAM10:
        clr         @YZ                 ;Clear @YZ content
        decms       Z                   ;z = z – 1 , skip next if z=0
        jmp         ClrRAM10
        clr         @YZ                 ;Clear address 0x00
        ret

;--------------------------------------------------------------------------
        ENDP
```

13.2 程序核对表

项目	说　　明
未定义的位	系统寄存器中所有标记为“0”的位（未定义的位）都必须置 0，以避免系统出错。
ADC	1. 设置 ADC 的输入引脚为输入模式； 2. 禁止 ADC 输入引脚的上拉电阻； 3. 进入睡眠模式前禁止 ADC； 4. 在睡眠模式下设置寄存器 P4CON 的有关位以避免额外的功耗； 5. 启动 ADC（ADENB = 1）后延迟 100us 等待 ADC 电路稳定。
中断	RAM 初始化之前禁止中断。
未使用的 I/O 口	未使用的 I/O 口应设为上拉/下拉的输入模式或低电平输出模式以减小电流损耗。
睡眠模式	使能 P0 的内置上拉电阻以避免未知的系统唤醒。
堆栈缓存器	注意子程序的调用和中断的使用，避免堆栈溢出。
系统初始化	1. 堆栈初始化：STKP = 0x7F，禁止全局中断； 2. RAM 初始化； 3. 初始化所有的系统寄存器； 4. 初始化所有的 I/O 引脚。
抗干扰性	1. 设置看门狗定时器为“Always On”以保护系统； 2. 使能“Noise Filter”； 3. 未使用的 I/O 口应设为低电平输出模式； 4. 不断刷新 RAM 中的重要的系统寄存器和变量以避免干扰。

14 指令集

Field	指令格式		指令描述	C	DC	Z	周期
MOVE	MOV	A,M	A ← M	-	-	√	1
	MOV	M,A	M ← A	-	-	-	1
	B0MOV	A,M	A ← M (bnak 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 和 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
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 ← M + A，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
	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，使能全局中断控制位	-	-	-	2
	NOP		空指令，无特别意义	-	-	-	1
	RETLW	I	PC← Stack, A← I	-	-	-	2

✸ 注：

1. **存储器 M 包括系统寄存器和用户自定义的寄存器；**
2. **若满足跳转条件，S = 1，否则 S = 0；**
3. **若 M 为系统寄存器（bank0 中的 80h~0FFh）则 N = 0，否则 N = 1。**

15 电气特性

15.1 极限参数

(All of the voltages referenced to Vss)

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin).....Vss - 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8P27142P, SN8P27142S, SN8P27143P, SN8P27143S, SN8P27143X, SN8P2714K, SN8P2714S, SN8P2715P, SN8P2715S..... 0°C ~ + 70°C
SN8P27142PD, SN8P27142SD, SN8P27143PD, SN8P27143SD, SN8P27143XD, SN8P2714KD, SN8P2714SD, SN8P2715PD, SN8P2715SD..... –40°C ~ + 85°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

15.2 电气特性

(All of voltages referenced to Vss, Vdd = 5.0V, fosc = 4 MHz,Fcpu=1MHZ, ambient temperature is 25ºC unless otherwise note.)

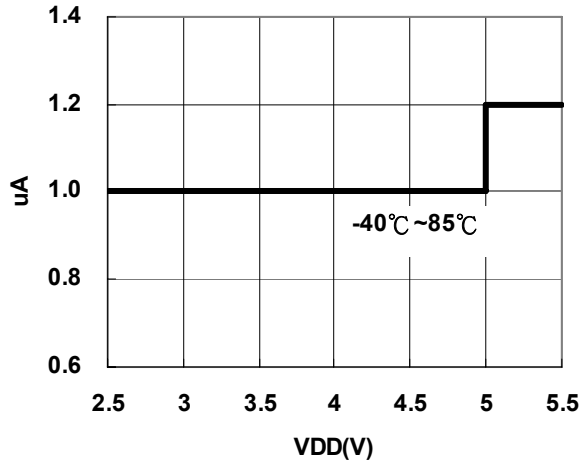
PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd, 25°C		2.4	5.0	5.5	V
		Normal mode, Vpp = Vdd, -40°C~85°C		2.5	5.0	5.5	
		Programming mode, Vpp = 12.5V			5.0		
RAM Data Retention voltage	Vdr			1.5	-	-	V
Internal POR	Vpor	Vdd rise rate to ensure internal power-on 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, 25°C		-	-	2	uA
		Vin = Vdd, -40°C~85°C		-	-	5	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
		Vin = Vss , Vdd = 5V		50	100	180	KΩ
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
Port2, Port4, Port 5 output current	IoH	Vop = Vdd - 0.5V(source)		8	12	-	mA
	IoL	Vop = Vss + 0.5V(sink)		8	15	-	mA
INTn trigger pulse width	Tint0	INT0 ~ INT1 interrupt request pulse width		2/fcpu	-	-	Cycle
AVREFH input voltage	Varfh	Vdd = 5.0V		2V	-	Vdd	V
AIN0 ~ AIN7 input voltage	Vani	Vdd = 5.0V		0	-	Varfh	V
ADC Clock Frequency	F_{ADCLK}	VDD=5.0V		-		8M	Hz
		VDD=3.0V		-		5M	Hz
ADC Conversion Cycle Time	F_{ADCYL}	VDD=2.4V~5.5V		64			$1/F_{ADCLK}$
ADC Sampling Rate (Set FADS=1 Frequency)	F_{ADSMP}	VDD=5.0V				125	K/sec
		VDD=3.0V				80	K/sec
Differential Nonlinearity	DNL	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K		±1	±2	±16	LSB
Integral Nonlinearity	INL	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K		±2	±4	±16	LSB
No Missing Code	NMC	VDD=5.0V , AVREFH=3.2V, F_{ADSMP} =7.8K		8	10	12	Bits
ADC enable time	Tast	Ready to start convert after set ADENB = "1"		100	-	-	uS
ADC current consumption	I_{ADC}	Vdd=5.0V		-	0.6*	-	mA
		Vdd=3.0V		-	0.4*	-	mA
DAC Full-scale Output Current	I_{FSO}	Vdd=5V, 25°C		8	14	21	mA
		Vdd=3V, 25°C		5	11	18	mA
		Vdd=5V, -40°C~85°C		8	18	27	mA
		Vdd=3V, -40°C~85°C		5	15	24	mA
DAC Loading Resistance	R_L	Vdd=5V		-	-	150	Ω
		Vdd=3V		-	-	100	Ω
DAC DNL	DAC_{DNL}	DAC Differential NonLinearity		-	±1*	-	LSB
DAC INL	DAC_{INL}	DAC Integral NonLinearity		-	±3*	-	LSB
Supply Current (Disable ADC)	Idd1	normal Mode Fcpu = Fosc/4	Vdd= 5V 4MHz	-	2.5	5	mA
			Vdd= 3V 4MHz	-	1.5	3	mA
	Idd2	Slow Mode (Internal RC mode, Stop high clock)	Vdd= 5V ~ ILRC 32Khz	-	25	50	uA
			Vdd= 3V ~ ILRC 16Khz	-	5	10	uA
	Idd3	Sleep mode (LVD = LVD_L)	Vdd= 5V, 25°C	-	1	2	uA
			Vdd= 3V, 25°C		0.6	1	uA
			Vdd= 5V, -40°C~85°C		10	21	uA
			Vdd= 3V, -40°C~85°C	-	10	21	uA
LVD Voltage	Vdet0	Low voltage reset level		1.7	2.0	2.3	V
	Vdet1	Low voltage reset/indicator level Fcpu=1MHz		2.0	2.4	3	V
	Vdet2	Low voltage indicator level Fcpu=1MHz		2.7	3.6	4.5	V

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

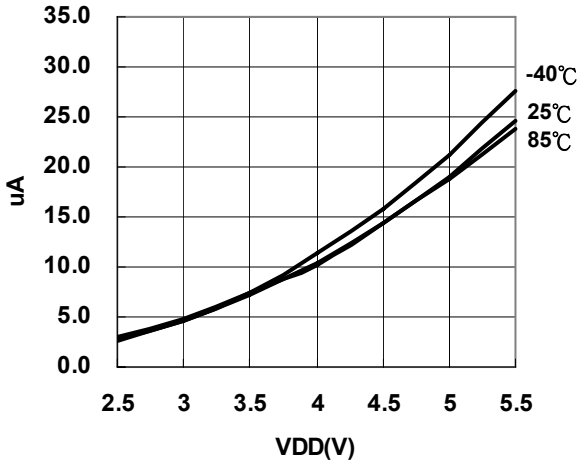
15.3 特性曲线图

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

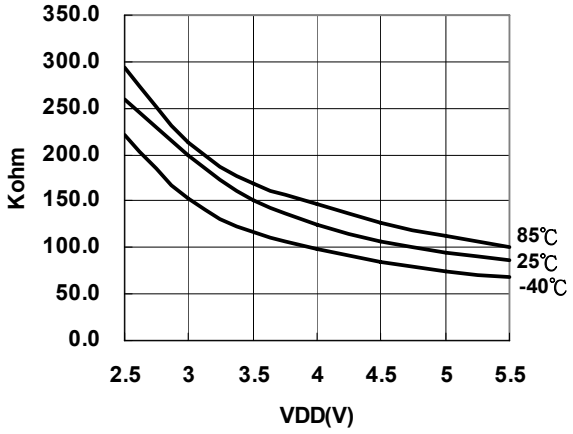
Standby current



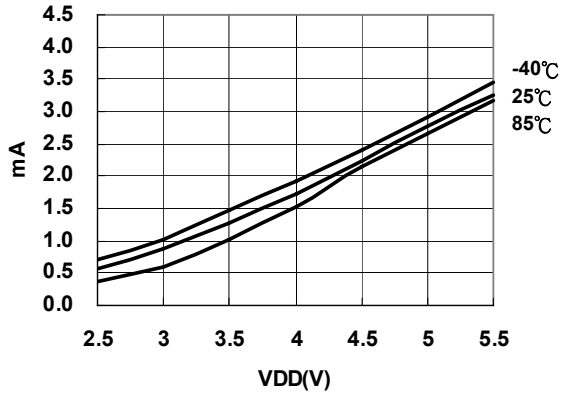
Slow mode current



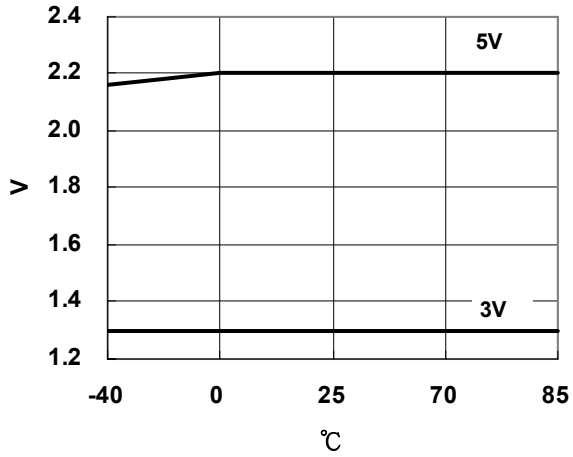
Pull up resistor (ALL PORT)



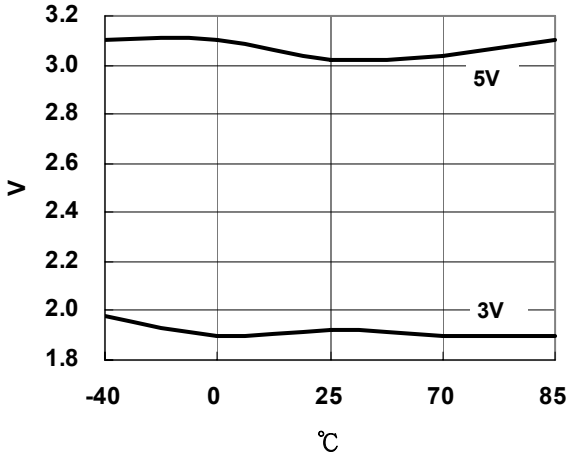
Operating current (Fosc=4MHZ、Fcpu=Fosc/4)



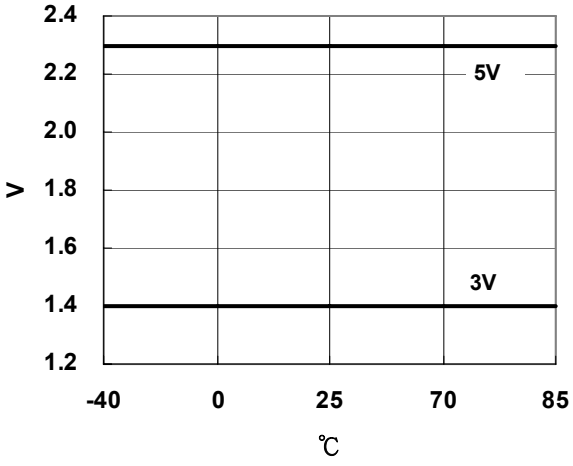
VIL(PORT2、PORT5)



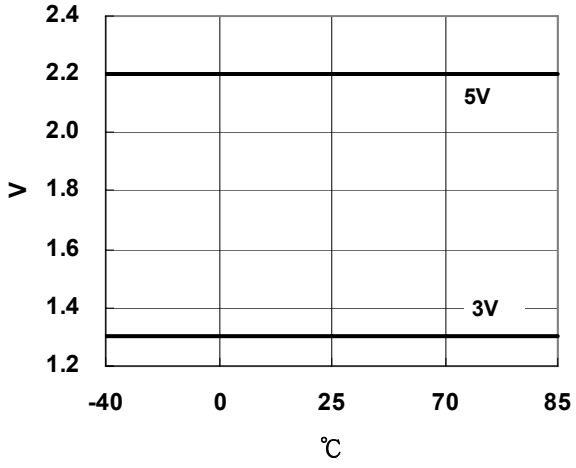
VIH(PORT2、PORT5)



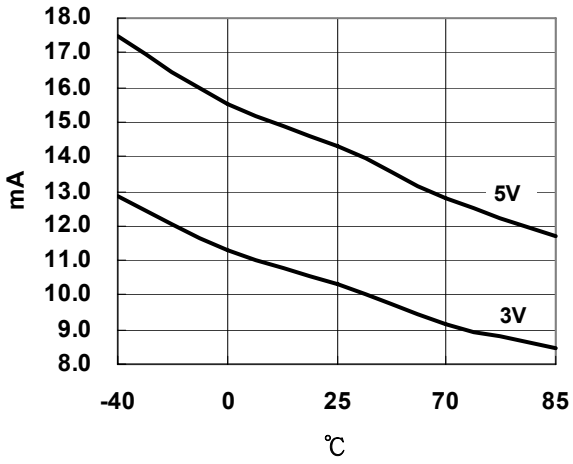
VIH(PORT4)



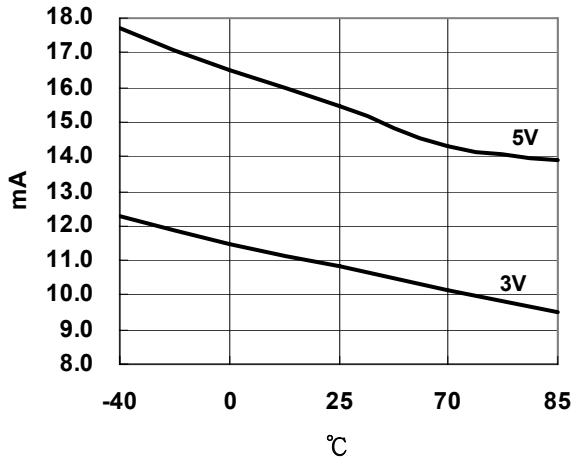
VIL(PORT4)



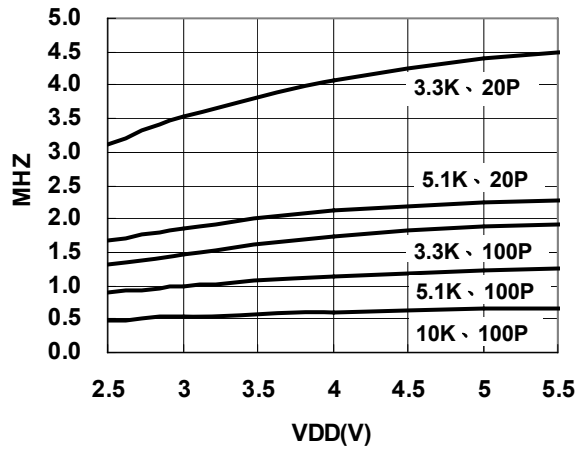
Sinking current (ALL PORT)



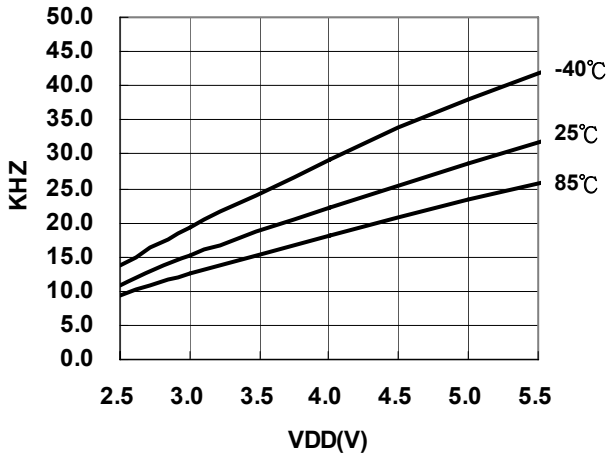
Driving current (ALL PORT)



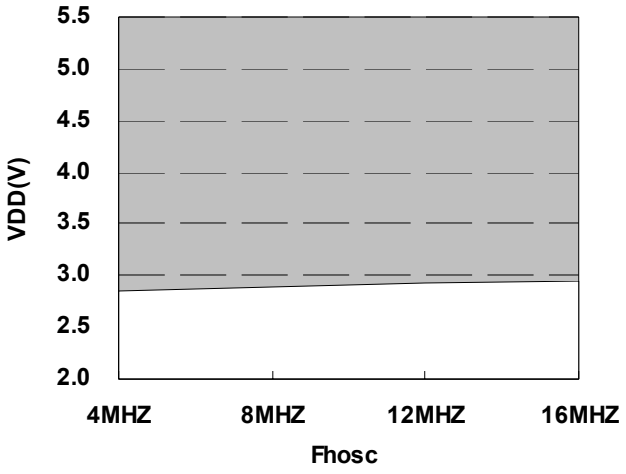
External RC Oscillator (25℃)



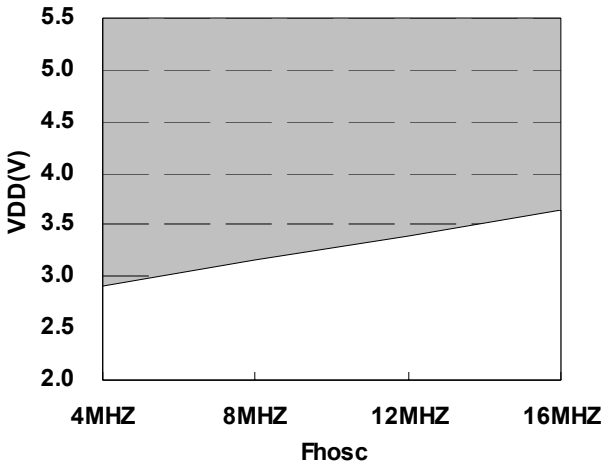
Internal low RC Oscillator



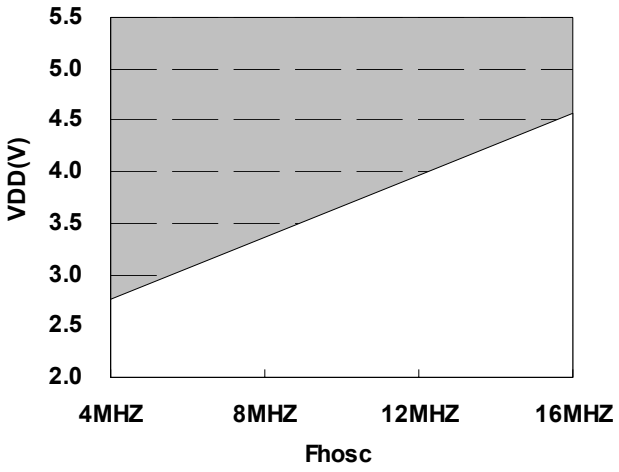
Fcpu=Fosc/4、Noise Filter Disable、-40~85℃



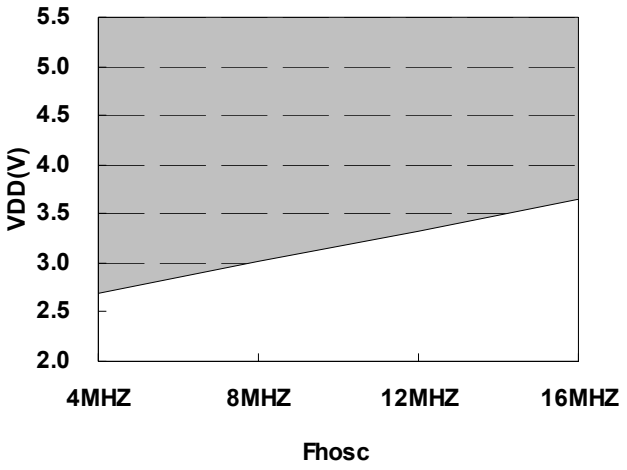
Fcpu=Fosc/1、Noise Filter Disable、-40~85℃



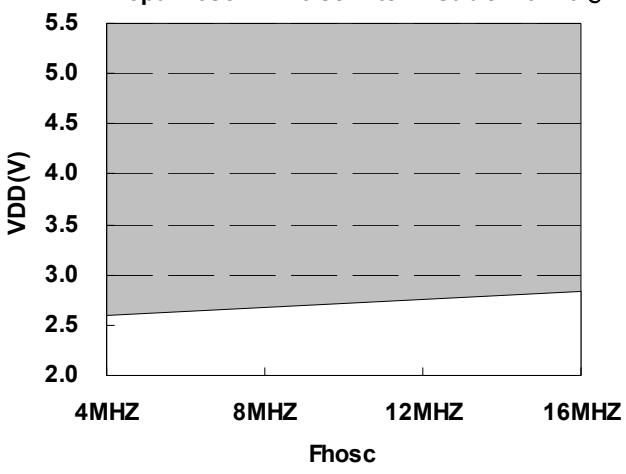
Fcpu=Fosc/4、Noise Filter Enable、-40~85℃



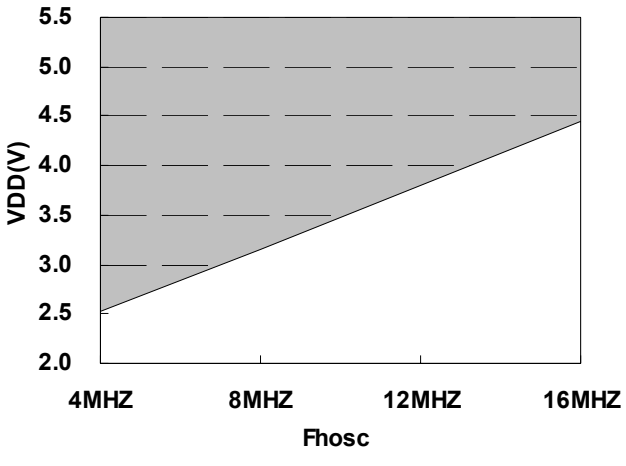
Fcpu=Fosc/1、Noise Filter Disable、0~70℃



Fcpu=Fosc/4、Noise Filter Disable、0~70℃



Fcpu=Fosc/4、Noise Filter Enable、0~70℃



16 开发工具

16.1 开发工具的版本

16.1.1 在线仿真器（ICE）

- **SN8ICE 2K**：仿真 SN8P2714/SN8P2715 系列的所有功能。

✷ **SN8ICE2K ICE 仿真时需注意：**
a. ICE 的工作电压：3.0V~5.0V.
b. 工作电压为 5V 时建议最大的工作速度为 8 MIPS（如 16MHZ 晶振，Fcpu = Fhosc/2）。
c. 使用 SN8P2714 / 2715 EV-KIT 仿真 LVD 功能。

✷ **注：S8KD-2 ICE 不支持 SN8P2714X 和 SN8P2715 系列的仿真。**

16.1.2 OTP Writer

- **MP WriterIII**：支持 SN8P2714/SN8P2715 系列的联机烧录，也支持大批量的脱机烧录。

16.1.3 集成开发环境（IDE）

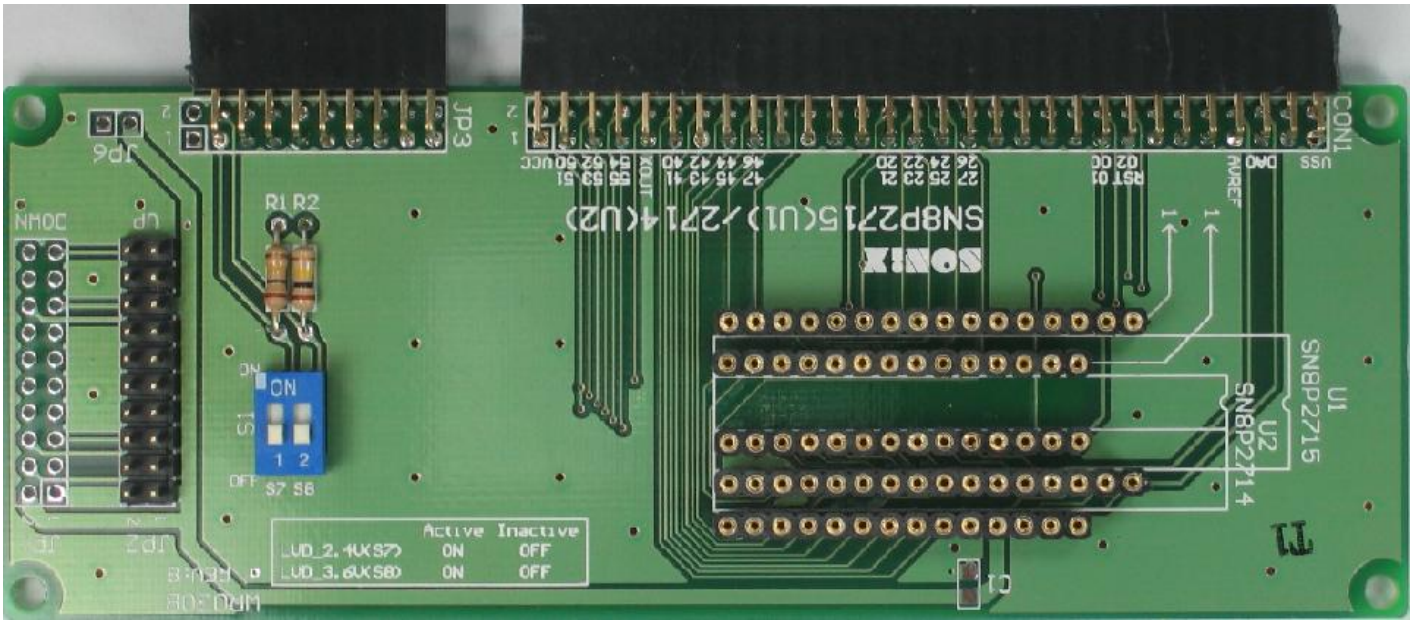
SONiX 8 位单片机的集成开发环境包括编译器、ICE 调试器和 OTP 的烧录软件。

- **SN8ICE 2K**：M2IDE_V115 或更新的版本。
- **MP WriterIII**：M2IDE_V115 或更新的版本。
- **SN8IDE V1.99X 不支持 SN8P2714/SN8P2715 的仿真和烧录。**

16.2 SN8P2715/SN8P2714 EV-KIT

16.2.1 PCB

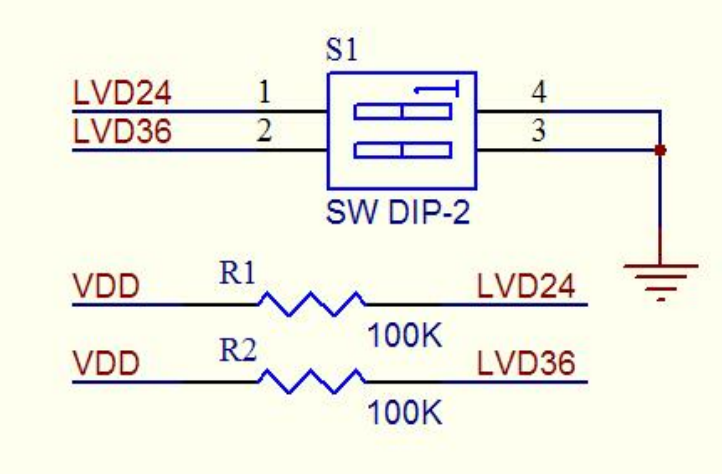
SONiX 提供 SN8P2715 EV-Kit 进行 ICE 仿真，另外 EV-Kit 还提供 LVD2.4V/3.6V 选择电路。



CON1 和 JP3：ICE I/O 接入端。

S1：LVD 2.4V 和 LVD 3.6V 控制开关，仿真 LVD 2.4V 标志/复位功能和 LVD3.6V 标志功能；

开关编号	On	Off	注
S7	LVD 2.4V 开	LVD 2.4V 关	VDD 小于 2.4V
S8	LVD 3.6V 开	LVD 3.6V 关	VDD 小于 3.6V



16.3 SN8P2715/14 EV-KIT 和 SN8ICE 2K 的连接

SN8P2714/SN8P2715 EV-KIT 和 SN8ICE 2K 的连接如下所示：

✷ **SN8ICE2K ICE 仿真时需注意：**

a. **ICE 的工作电压：3.0V~5.0V.**

b. **工作电压为 5V 时建议最大的工作速度为 8 MIPS（如 16MHZ 晶振，Fcpu = Fhosc/2）。**

c. **使用 SN8P2714 / 2715 EV-KIT 仿真 LVD 功能。**

d. **当芯片宣告 SN8P2715/SN8P27142/SN8P27143 时断开 SN8ICE 2K 上的 AVREFH/VDD 跳线。**

16.4 OTP 烧录信息

16.4.1 烧录转接板信息

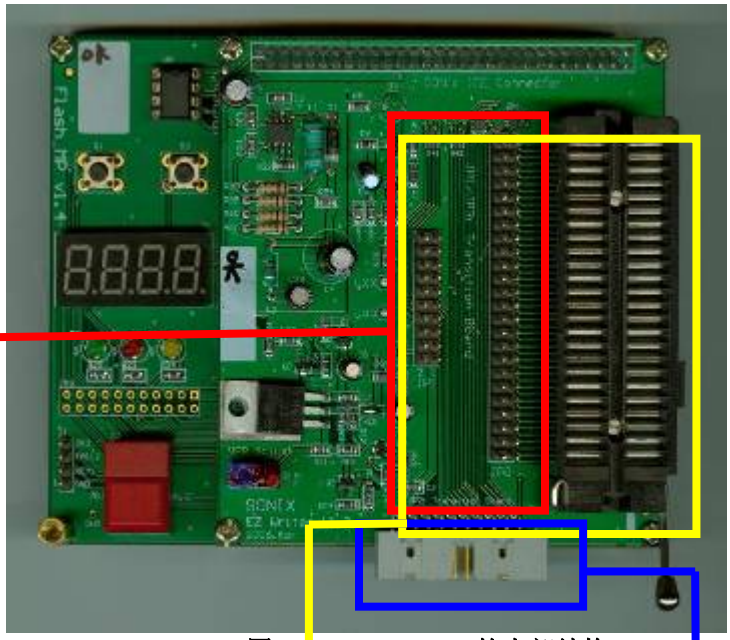
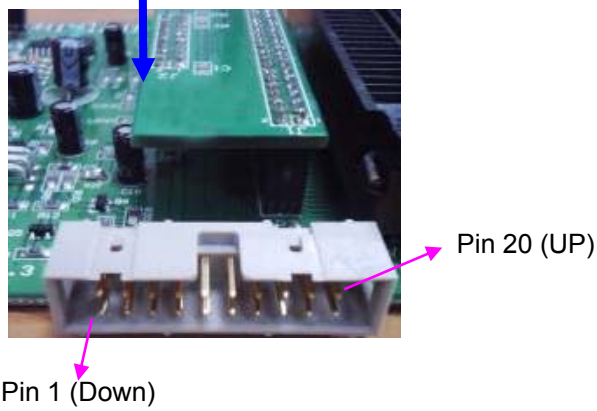


图 1 MPIII Writer 的内部结构

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。

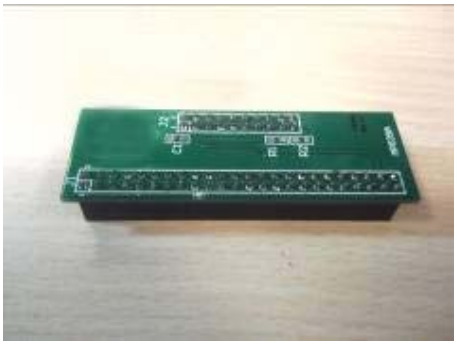


图 2

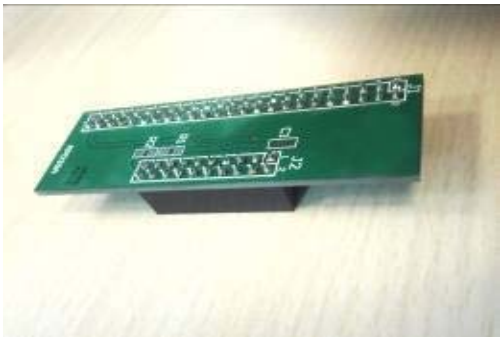


图 3

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。

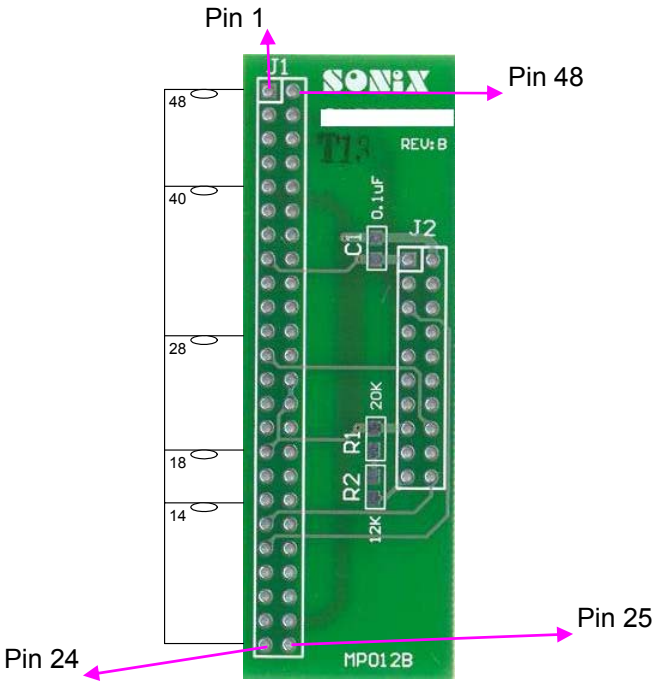


图 4　MP 转接板（连接到 JP1&JP3）

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

16.4.2 SN8P2710 系列烧录引脚信息

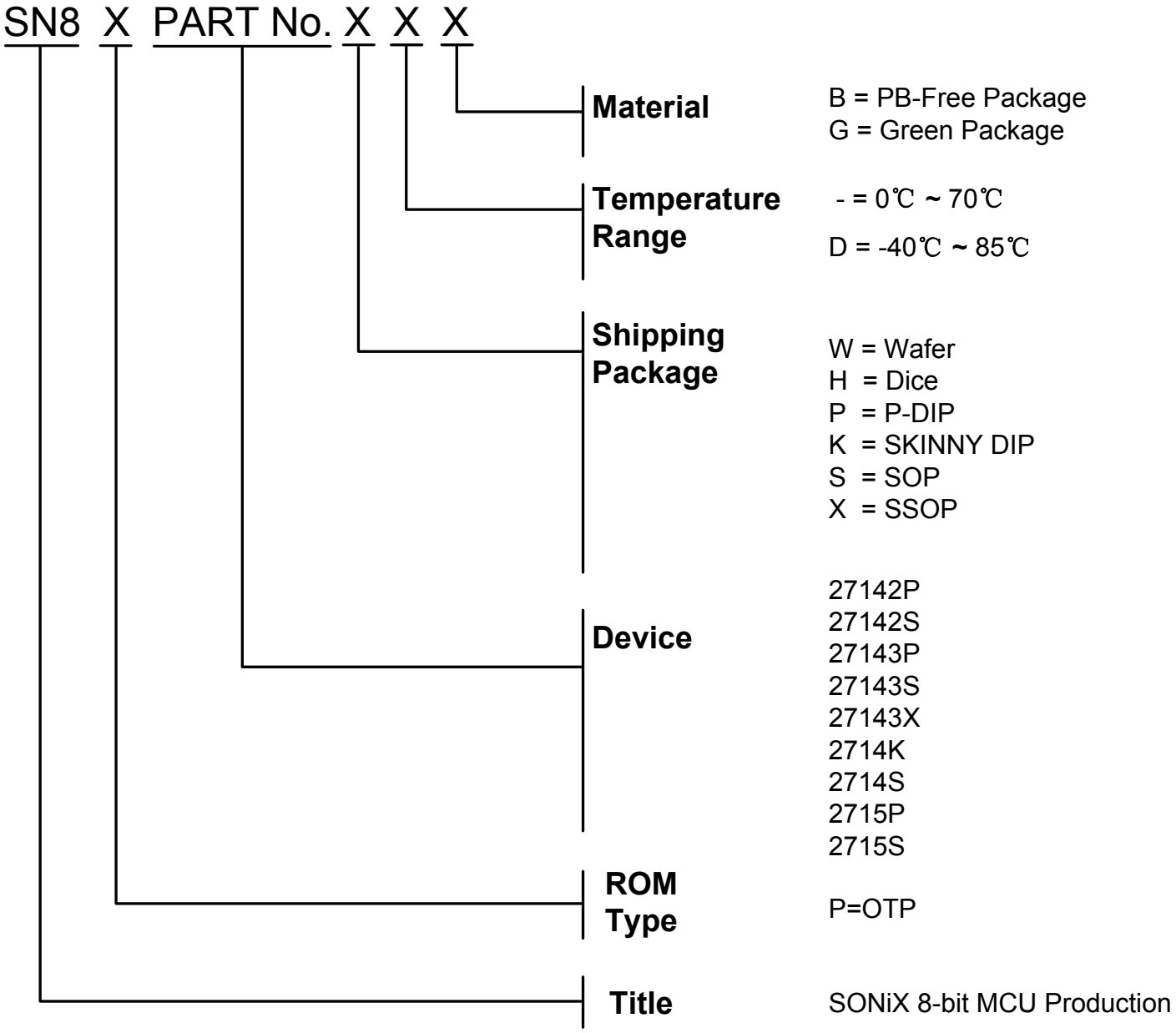
SN8P2710 系列烧录引脚配置													
单片机型号		SN8P2714			SN8P2715			SN8P27142			SN8P27143		
MPIII Writer		OTP IC / JP3 引脚配置											
JP1/JPJ2 Pin Number	JP1/JP2 Pin Name	IC Pin Number	IC Pin Name	JP3 Pin Number	IC Pin Number	IC Pin Name	JP3 Pin Number	IC Pin Number	IC Pin Name	JP3 Pin Number	IC Pin Number	IC Pin Name	JP3 Pin Number
1	VDD	25	VDD	35	30	VDD	38	12	VDD	27	13	VDD	27
2	GND	15	VSS	25	20	VSS	28	6	VSS	21	5	VSS	19
3	CLK	4	P5.0	14	6	P5.0	14	17	P5.0	32	18	P5.0	32
4	CE	-	-	-	-	-	-	-	-	-	-	-	-
5	PGM	8	P2.0	17	10	P2.0	18	2	P2.0	17	1	P2.0	15
6	OE	3	P5.1	31	5	P5.1	13	16	P5.1	31	17	P5.1	31
7	D1	-	-	-	-	-	-	-	-	-	-	-	-
8	D0	-	-	-	-	-	-	-	-	-	-	-	-
9	D3	-	-	-	-	-	-	-	-	-	-	-	-
10	D2	-	-	-	-	-	-	-	-	-	-	-	-
11	D5	-	-	-	-	-	-	-	-	-	-	-	-
12	D4	-	-	-	-	-	-	-	-	-	-	-	-
13	D7	-	-	-	-	-	-	-	-	-	-	-	-
14	D6	-	-	-	-	-	-	-	-	-	-	-	-
15	VDD	25	VDD	35	30	VDD	38	12	VDD	27	13	VDD	27
16	VPP	26	RST	36	31	RST	39	13	RST	28	14	RST	28
17	HLS	-	-	-	-	-	-	-	-	-	-	-	-
18	RST	-	-	-	-	-	-	-	-	-	-	-	-
19	-	-	-	-	-	-	-			-			-
20	ALSB/PDB	9	P2.1	19	11	P2.1	19	3	P2.1	18	2	P2.1	16

17 单片机正印命名规则

17.1 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。

17.2 单片机型号说明



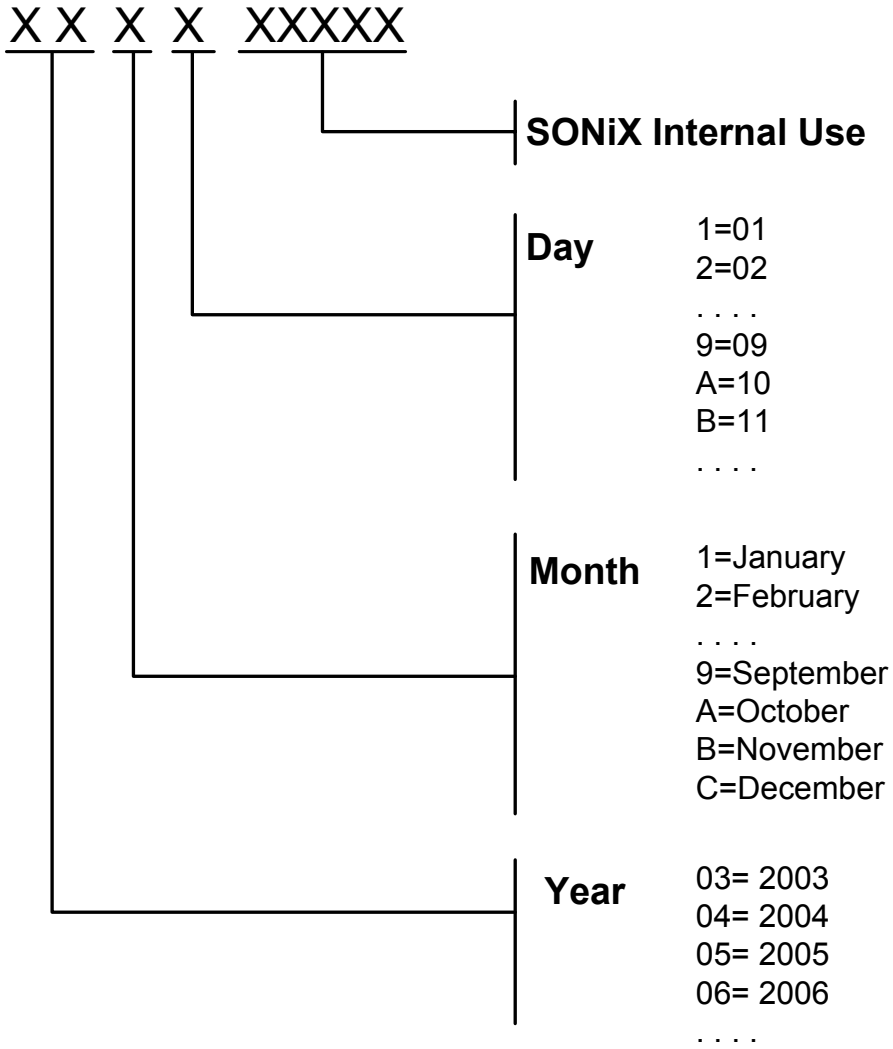
17.3 命名举例

名称	ROM 类型	器件（Device）	封装形式	温度范围	封装材料
SN8P27142PDG	OTP	27142	P-DIP	-40℃~85℃	绿色封装（Green Package）
SN8P2624SB	OTP	2624	SOP	0℃~70℃	无铅封装（PB-Free Package）
SN8P1907H	OTP	1907	Dice	0℃~70℃	N/A
SN8A1708AX	Mask	1708A	SSOP	0℃~70℃	N/A

✸ 注：工业级别的生产不支持 **Wafer** 或 **Dice** 的运送。

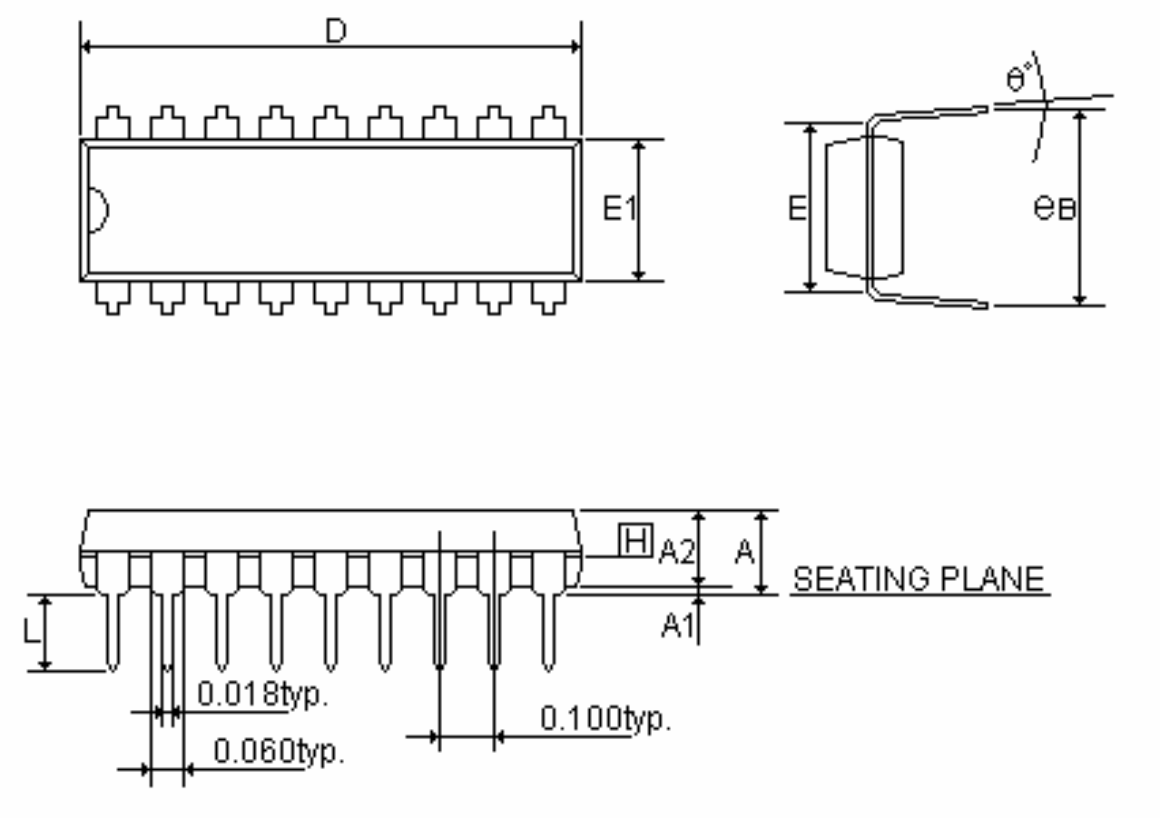
17.4 日期码规则

SONiX 的日期码规则共有 8~9 个字符，后 4（5）个字符是指 SONiX 的内部使用编号，前 4 个则是指封装的日期，包括年/月/日。如下图所示：



18 封装信息

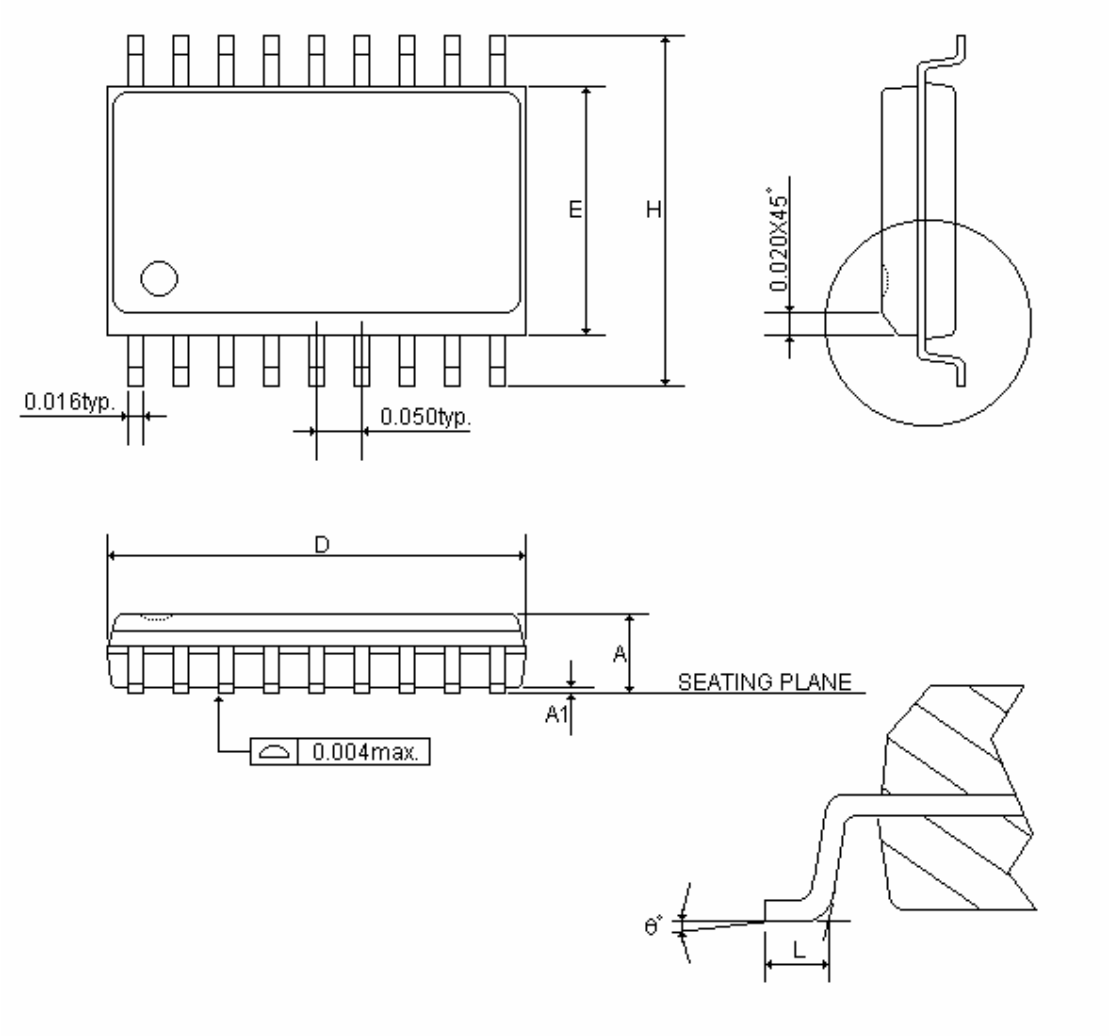
18.1 P-DIP18 PIN



Symbols	MIN.	NOR.	MAX.
A	-	-	0.210
A1	0.015	-	-
A2	0.125	0.130	0.135
D	0.880	0.900	0.920
E		0.300BSC.	
E1	0.245	0.250	0.255
L	0.115	0.130	0.150
e B	0.335	0.355	0.375
θ°	0	7	15

UNIT : INCH

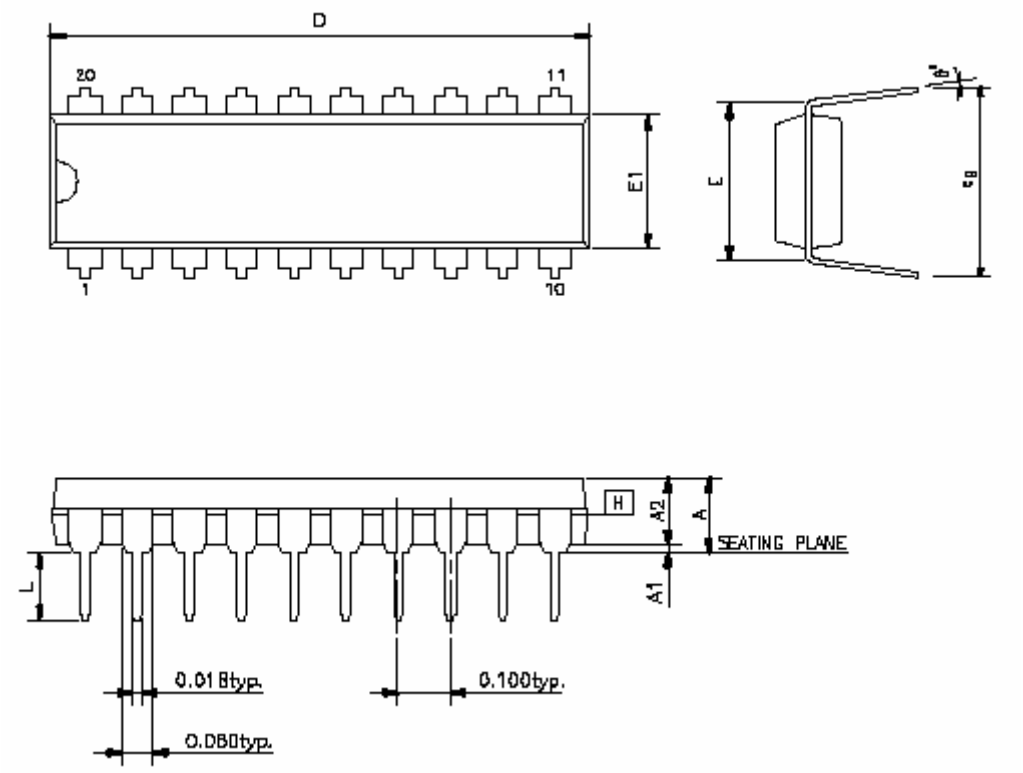
18.2 SOP18 PIN



Symbols	MIN.	MAX.
A	0.093	0.104
A1	0.004	0.012
D	0.447	0.463
E	0.291	0.299
H	0.394	0.419
L	0.016	0.050
θ°	0	8

UNIT : INCH

18.3 P-DIP 20 PIN

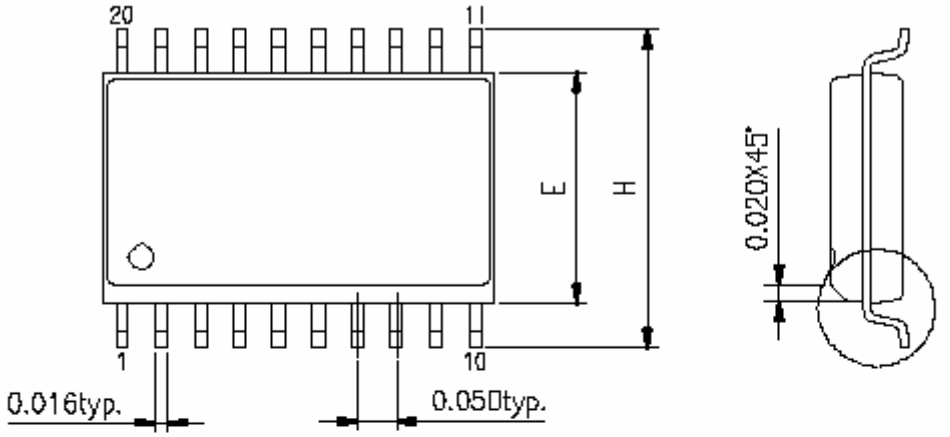


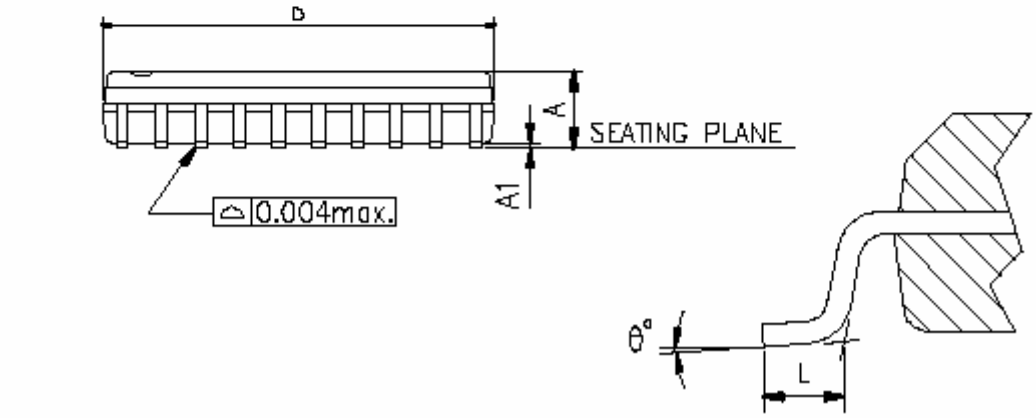
SYMBOLS	MIN.	NOR.	MAX.
A	–	–	0.210
A1	0.015	–	–
A2	0.125	0.130	0.135
D	0.98	1.030	1.060
E		0.300 BSC.	
E1	0.245	0.250	0.255
L	0.115	0.130	0.150
e_B	0.335	0.355	0.375
θ°	0	7	15

UNIT : INCH

NOTES:
1. JEDEC OUTLINE : MS-001 AD
2. "D","E1" DIMENSIONS DO NOT INCLUDE MOLD FLASH OR PROTRUSIONS.MOLD FLASH OR PROTRUSIONS SHALL NOT EXCEED .010 INCH.
3. eB IS MEASURED AT THE LEAD TIPS WITH THE LEADS UNCONSTRAINED.
4. POINTED OR ROUNDED LEAD TIPS ARE PREFERRED TO EASE INSERTION.
5. DISTANCE BETWEEN LEADS INCLUDING DAM BAR PROTRUSIONS TO BE .005 INCH MINIMUM.
6. DATUM PLANE [H] COINCIDENT WITH THE BOTTOM OF LEAD, WHERE LEAD EXITS BODY.

18.4 SOP 20 PIN



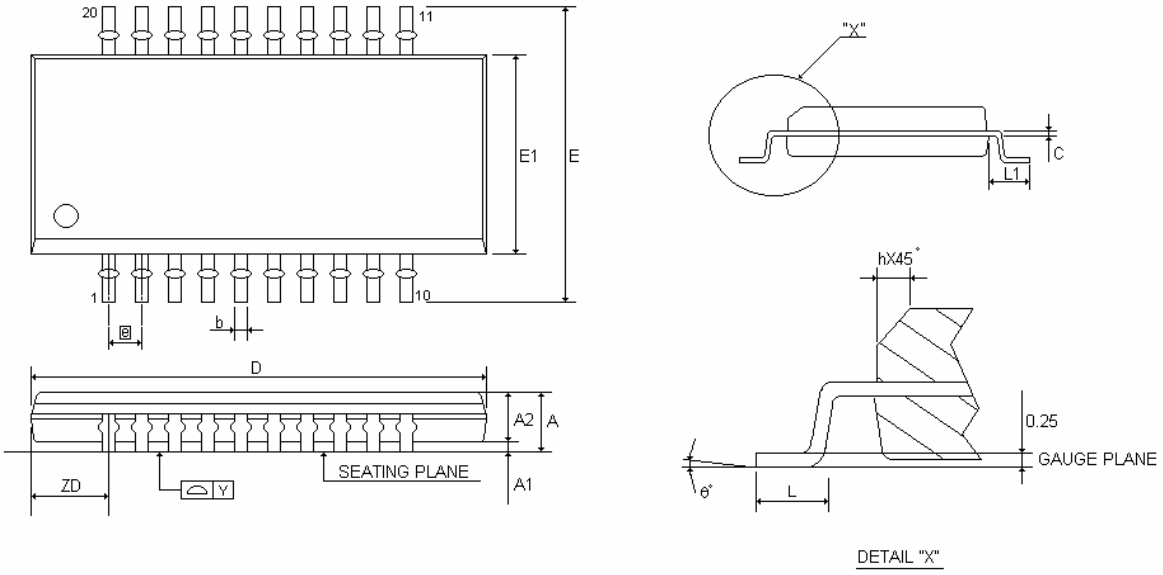


SYMBOLS	MIN.	MAX.
A	0.093	0.104
A1	0.004	0.012
D	0.496	0.508
E	0.291	0.299
H	0.394	0.419
L	0.016	0.050
θ°	0	8

UNIT : INCH

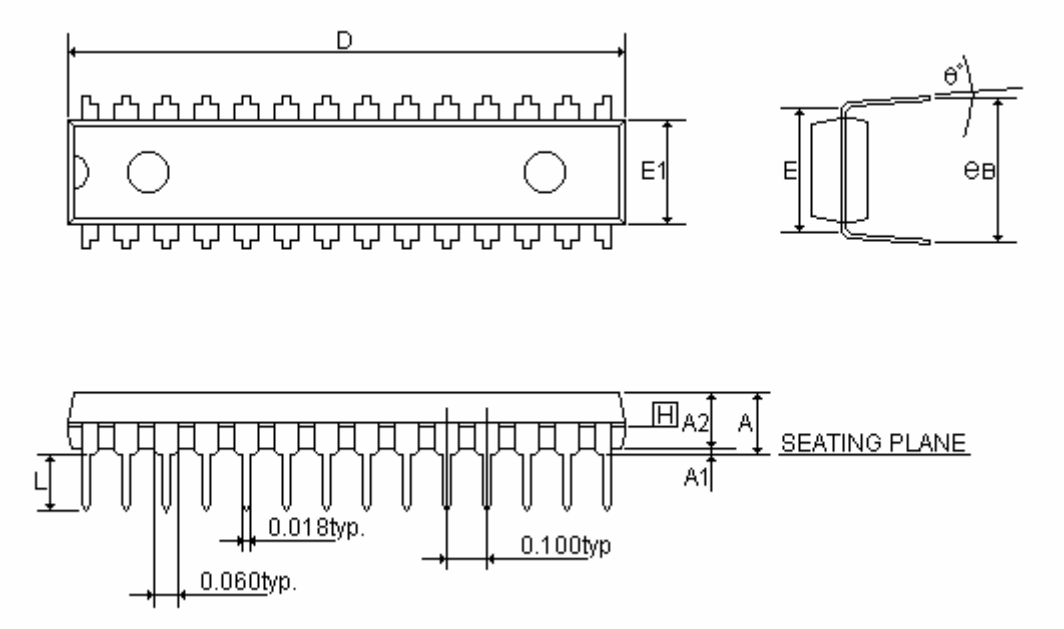
NOTES:
1.JEDEC OUTLINE : MS-013 AC
2.DIMENSIONS "D" DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS.MOLD FLASH, PROTRUSIONS AND GATE BURRS SHALL NOT EXCEED .15mm (.006in) PER SIDE.
3.DIMENSIONS "E" DOES NOT INCLUDE INTER-LEAD FLASH, OR PROTRUSIONS. INTER-LEAD FLASH AND PROTRUSIONS SHALL NOT EXCEED .25mm (.010in) PER SIDE.

18.5 SSOP20 PIN



Symbols	DIMENSION (MM)			DIMENSION (MIL)		
	MIN,	NOM.	MAX.	MIN.	NOM.	MAX.
A	1.35	1.60	1.75	53	63	69
A1	0.10	0.15	0.25	4	6	10
A2	-	-	1.50	-	-	59
b	0.20	0.254	0.30	8	10	12
b1	0.20	0.254	0.28	8	11	11
C	0.18	0.203	0.25	7	8	10
C1	0.18	0.203	0.23	7	8	9
D	8.56	8.66	8.74	337	341	344
E	5.80	6.00	6.20	228	236	244
E1	3.80	3.90	4.00	150	154	157
e	0.635 BSC			25 BSC		
h	0.25	0.42	0.50	10	17	20
L	0.40	0.635	1.27	16	25	50
L1	1.00	1.05	1.10	39	41	43
ZD	1.50 REF			58 REF		
Y	-	-	0.10	-	-	4
θ°	0°	-	8°	0°	-	8°

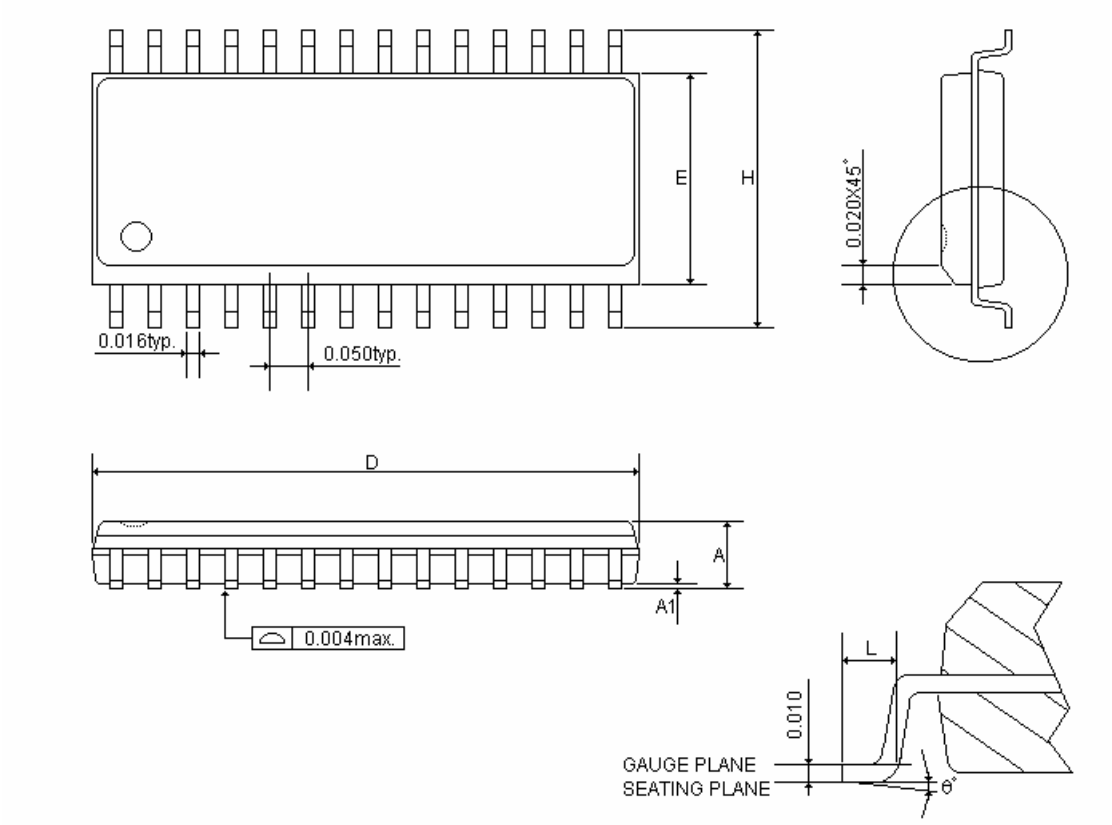
18.6 SK-DIP28 PIN



Symbols	MIN.	NOR.	MAX.
A	-	-	0.210
A1	0.015	-	-
A2	0.114	0.130	0.135
D	1.390	1.390	1.400
E		0.310BSC.	
E1	0.283	0.288	0.293
L	0.115	0.130	0.150
e B	0.330	0.350	0.370
θ°	0	7	15

UNIT : INCH

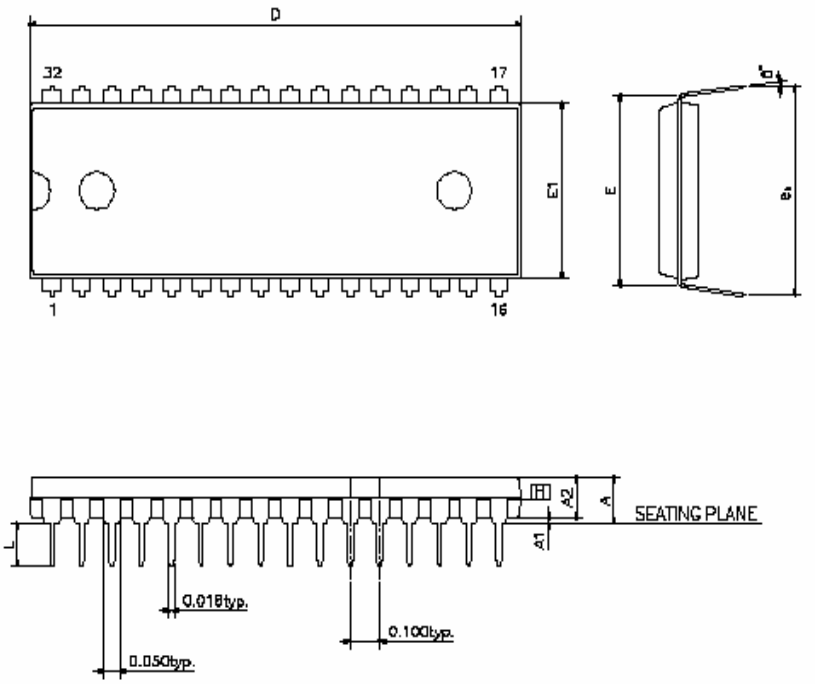
18.7 SOP28 PIN



Symbols	MIN.	MAX.
A	0.093	0.104
A1	0.004	0.012
D	0.697	0.713
E	0.291	0.299
H	0.394	0.419
L	0.016	0.050
θ°	0	8

UNIT : INCH

18.8 P-DIP 32 PIN

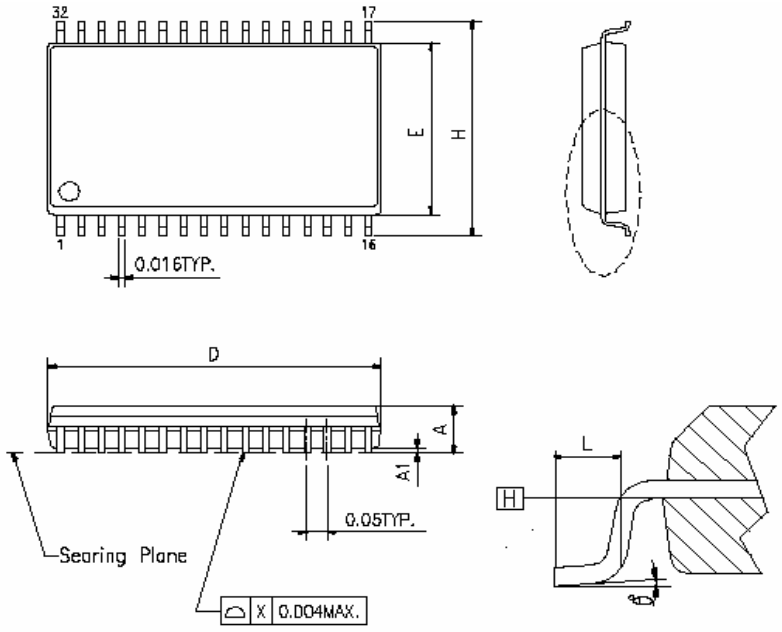


SYMBOLS	MIN.	NOR.	MAX.
A	–	–	0.220
A1	0.015	–	–
A2	0.150	0.155	0.160
D	1.645	1.650	1.660
E		0.600 BSC	
E1	0.540	0.545	0.550
L	0.115	0.130	0.150
e_B	0.630	0.650	0.670
θ°	0	7	15

UNIT : INCH

NOTE:
1.JEDEC OUTLINE : MS-011 AC

18.9 SOP 32 PIN



SYMBOLS	MIN.	MAX.
A	–	0.120
A1	0.004	0.014
D	0.799	0.815
E	0.437	0.450
H	0.530	0.580
L	0.016	0.050
θ°	0°	10°

UNIT : INCH

NOTE:
1.JEDEC OUTLINE:MO-099 AB
2.DATUM PLANE ⊞IS LOCATED AT THE BOTTOM OF THE MOLD PARTING LINE COINCIDENT WITH WHERE THE LEAD EXITS THE BODY.
3.DIMENSIONS E AND D DO NOT INCLUDE MOLD PROTRUSION. ALLOWABLE PROTRUSION IS 10 MIL PER SIDE. DIMENSIONS E AND D DO INCLUDE MOLD MISMATCH AND ARE DETERMINED AT DATUM PLANE ⊞
4.DIMENSION b DOES NOT INCLUDE DAMBAR PROTRUSION.

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

总公司：
地址：台湾新竹县竹北市台元街 36 号 10 楼之一
电话：886-3-5600-888
传真：886-3-5600-889
台北办事处：
地址：台北市松德路 171 号 15 楼之 2
电话：886-2-2759 1980
传真：886-2-2759 8180
香港办事处：
地址：香港新界沙田沙田乡宁会路 138＃新城市中央广场第一座 7 楼 705 室
电话：852-2723 8086
传真：852-2723 9179
松翰科技（深圳）有限公司
地址：深圳市南山区高新技术产业园南区 T2-B 栋 2 层
电话：86-755-2671 9666
传真：86-755-2671 9786
技术支持：
Sn8fae@sonix.com.tw