SN8P2710 系列

用户手册

SN8P2714
SN8P2715

SONiX 8 位单片机

修改记录

版本	日期	说明
VER 0.1	2004 年 8 月	第一版

目　录

1 产品简介

1.1 SN8P2710 特性

- **存储器配置**
  OTP ROM：2K * 16 bits.
  RAM：128 * 8 bits
  8 层堆栈缓存器

- **I/O 引脚配置**
  单向输入：P0
  双向输入输出：P2, P4, P5
  唤醒功能：P0 电平变换触发
  外部中断：P0
  上拉电阻：P0, P2, P4, P5
  P4 引脚和 ADC 输入共享
- **8 通道 12 位 AD 转换**

- **1 通道 7 位 DA 转换**

- **功能强大的指令集**
  一个指令周期就是一个时钟周期(1T)
  大部分指令的执行时间均为一个周期
  查表指令（MOVC）可直接寻址整个 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：自动加载定时/计数器/PWM0/Buzzer 输出
  TC1：自动加载定时/计数器/PWM1/Buzzer 输出
- **内置看门狗定时器**

- **系统时钟和操作模式**
  外部高速时钟：RC　最大 10 MHz
  外部高速时钟：晶体　最大 16 MHz
  内部低速时钟：RC　16KHz(3V), 32KHz(5V)
  普通模式：高低速时钟均运行
  低速模式：仅低速时钟运行
  睡眠模式：高低速时钟均停止

- **封装**
  SN8P2714：SK-DIP 28 pins, SOP 28pins
  SN8P2715：PDIP 32 pins, SOP 32 pins

SN8P2700A/SN8P2710 系列产品性能表

芯片型号	ROM	RAM	堆栈层数	定时器			I/O	ADC	DAC	PWM	SIO	唤醒功能引脚数目	封装
				T0	TC0	TC1				Buzzer			
SN8P2714	2K*16	128	8	-	V	V	24	8ch	1ch	2	-	2	SKDIP28/SOP28
SN8P2715	2K*16	128	8	-	V	V	28	8ch	1ch	2	-	2	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

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 之间添加一个 47uF 的旁路电容)**	很好 **(在 VDD 和 GND 之间添加一个 47uF 的旁路电容)**
存储器	ROM：2K x 16 RAM：128 x 8	ROM：4K x 16 RAM：256 x 8
最多 I/O 引脚数	28 pins：24 I/O 32 pins：28 I/O	28 pins：18 I/O 32 pins：23 I/O
高速 PWM	PWM 分辨率：8/6/5/4 位 High Clock = 16MHz 8 位分辨率时高达 31.25K 4 位分辨率时高达 500K	PWM 分辨率：8/6/5/4 位 High Clock = 16MHz 8 位分辨率时高达 31.25K 4 位分辨率时高达 500K
可编程开漏输出	N/A	P1.0 / P1.1 / P5.2 (SO)
B0MOV M, I	无限制	I 不能为 0E6 或 0E7
B0XCH A, M	无限制	M 的地址不能为 80h~FFh
ROM 中地址 8 处的有效指令	无限制	JMP 或 NOP
ADC 中断	无	有
ADC 时钟频率	四种设置（通过 ADCKS [1:0]设置)	七种设置(通过 ADCKS [2:0]设置)
TC0C/TC1C/TC0R/TC1R 的有效范围	0x00 ~ 0xFF	0x00 ~ 0xFE
绿色模式	无	有
SIO 功能	无	有
LVD	电压：2.0V	1.8V 常开
P0	单向输入端口	双向输入/输出端口

1.2 系统框图

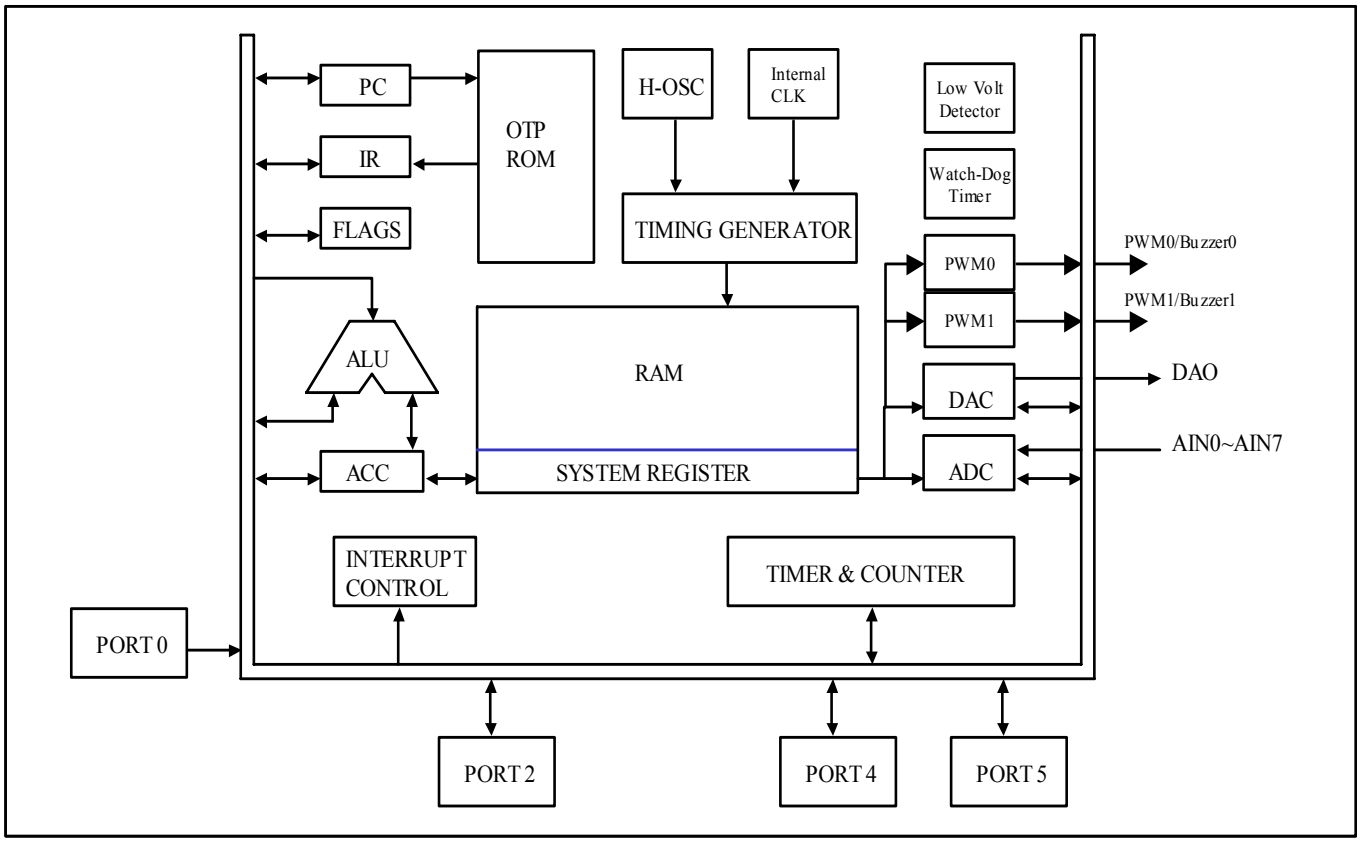
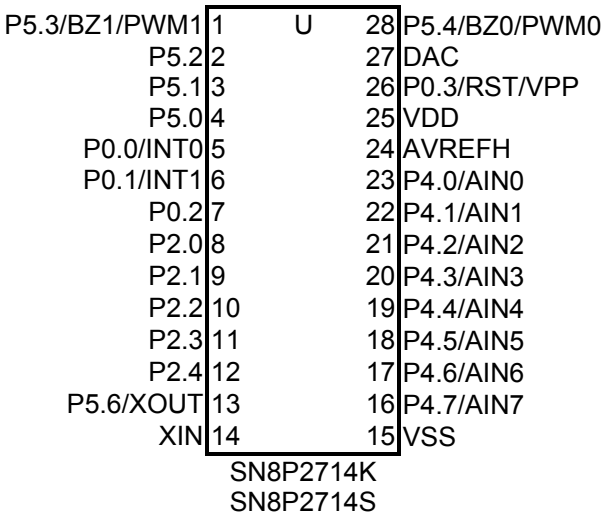


图 1-1.系统简图

1.3 引脚配置

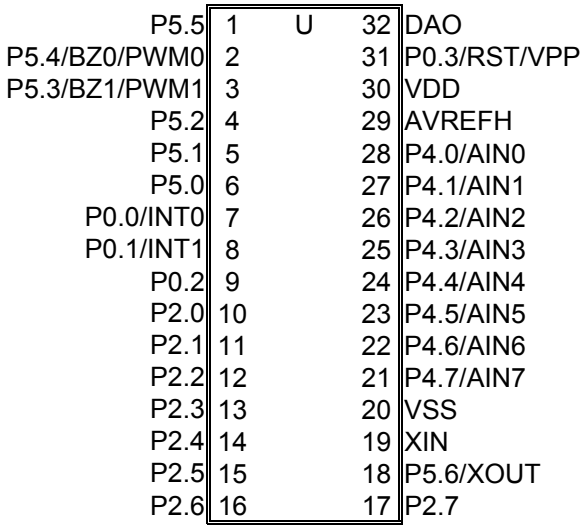
格式说明：SN8P271xY　　Y = K > SK-DIP，P > P-DIP，　S> SOP

SN8P2714K (SKDIP28)
SN8P2714S (SOP28)



SN8P2714K
SN8P2714S

SN8P2715P (P-DIP32)
SN8P2715S (SOP32)



SN8P2715P
SN8P2715S

1.4 引脚说明

引脚名称	类型	说明
P0 [1:0] / INT [1:0]	I	P0 [1:0]：单向输入引脚/唤醒功能引脚/上拉电阻/施密特触发 INT [1:0]：外部中断
P0 .2	I	P0.2 单向输入引脚/上拉电阻/施密特触发
P2 [7:0]	I/O	P2 [7:0]双向输入输出引脚/上拉电阻/施密特触发
P4 [7:0] / AIN [7:0]	I/O	双向输入输出引脚/上拉电阻/ADC 输入/施密特触发
P5 [5:0]	I/O	双向输入输出引脚/上拉电阻/施密特触发 P5.4：PWM0/BZ0，P5.3：PWM1/BZ1
AVREFH	I	ADC 参考电压输入端
DAO	O	DAC 信号输出端
P0.3/RST/VPP	I/P	P0.3：施密特触发/唤醒功能引脚/没有上拉电阻/RC 模式 RST：外部复位，低电平有效 VPP：OTP 编程引脚
XIN	I	外部振荡器输入引脚/外部 RC 振荡器输入
XOUT/P5.6	I/O	XOUT：外部振荡器输出引脚 P5.6：双向输入输出/上拉电阻/在 RC 模式中施密特触发
VDD, VSS	P	电源输入端

表 1-1. SN8P2714/15 引脚说明

1.5 引脚电路图

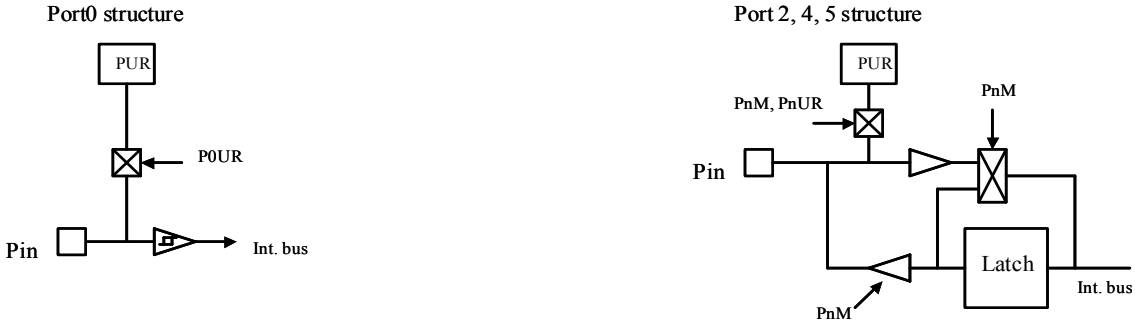


图 1-2. 引脚电路图

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 RC 振荡电路。 Fcpu 时钟由 Xout 引脚输出。
	32K_X’tal	外部高速时钟振荡器采用低频振荡器（如 32.768KHz)。
	12M_X’tal	外部高速时钟振荡器采用高频晶体振荡器/陶瓷谐振器（如 12MHz~16MHz）。
	4M_X’tal	外部高速时钟振荡器采用标准晶体振荡器/陶瓷谐振器（如 4M~10MHz）。
Noise_Filter	Enable	使能噪音滤除功能以提高抗干扰性能。
	Disable	禁止噪音滤除功能。
Watch_Dog	Always_On	看门狗定时器总数使能，即使在睡眠（省电）模式下。
	Enable	普通模式：使能看门狗定时器； 睡眠模式：停止看门狗定时器。
	Disable	禁止看门狗定时器功能。
Fcpu	Fosc/1	指令周期为 1 个振荡器时钟。
	Fosc/2	指令周期为 2 个振荡器时钟。
	Fosc/4	指令周期为 4 个振荡器时钟。
	Fosc/8	指令周期为 8 个振荡器时钟。
Security	Enable	允许 ROM 程序代码加密。
	Disable	禁止 ROM 程序代码加密。
RST_P0.3	Reset	使能外部复位
	P0.3	使能 P0.3 引脚的单向输入功能且无上拉电阻。
LVD	LVD0	LVD=2.0V 时复位
	LVD1	LVD=2.0V 时复位，由 LVD24 位作为 2.4V 的指示器。

表 2-1. SN8P271x 编译选项表

注：

- **在高干扰环境下，强烈建议使能 “Noise Filter” 选项并选择看门狗为 “Always_On” ；**
- **使能 “Noise Filter” 选项会限制 Fcpu，Fcpu = Fosc/4 ~ Fosc/8；**
- **Fcpu 编译选项只对高速时钟下有效；**
- **在普通模式下，Fosc=Fhosc（外部高速时钟）；**
- **在低速模式下，Fosc=Flosc（内部低速 RC 时钟）；**
- **低速模式下，Fcpu=Fpsc/4。**

3 存储器

3.1 程序存储器(ROM)

3.1.1概述

SN8P2710 程序存储器为 OTP ROM，容量为 2K*16-bit，可由 12 位程序计数器 PC 对程序存储器进行寻址，或由专用寄存器（R、X、Y 和 Z）对 ROM 进行查表访问。所有 2048 x 16 位程序存储器通常分为 4 个区域。

- 1-word 复位向量地址
- 1-word 中断向量地址
- 4-word 保留区域
- 2K words 通用区

所有的存储器分为 3 个代码区：0000H~0003H 用来执行复位向量；0004H~0007H 系统保留；0008H~07FBH 是中断向量和用户程序代码区。0008H 是中断服务程序的入口地址。

图 3-1 ROM 区域划分

3.1.2复位向量地址(0000H)

上电复位或看门狗计数器溢出复位后，系统从地址 0000H 开始重新执行程序，所有的系统寄存器恢复为默认值。下面的例子给出了如何在程序存储器里定义复位向量。

➲ **例： 定义复位向量**

```
            ORG     0           ; 0000H
            JMP     START       ; 跳转到用户程序区
            .                   ; 0004H ~ 0007H 保留
            ORG     10H
START:                          ; 0010H,用户程序的起始位置
            .                   ; 用户程序
            .
            ENDP                ; 程序结束
```

3.1.3中断向量地址 (0008H)

一旦有中断响应，程序计数器（PC）的值就会存入堆栈缓冲器中并跳转至 0008H 处执行中断服务程序。用户使用时必须自行定义中断向量，下面的例子给出了如何在程序中定义中断向量。

➲ **例 1：定义中断向量，主程序位于中断服务程序后面。**

```
        ORG     0               ; 0000H
        JMP     START           ; 跳转到用户程序
        .                       ; 0004H ~ 0007H 保留

        ORG     8               ; 中断服务程序

        B0XCH   A, ACCBUF       ; B0XCH 不会影响到 C, Z
        PUSH                    ; 保存工作寄存器
        .
        POP                     ; 恢复工作寄存器
        B0XCH   A, ACCBUF
        RETI                    ; 中断返回
START:                          ; 用户程序起始地址
        .                       ; 用户程序
        .
        JMP     START           ; 用户程序结束

        ENDP                    ; 程序结束
```

➲ **例 2：定义中断向量，中断服务程序位于主程序后面。**

```
        ORG     0               ; 0000H
        JMP     START           ; 跳转到用户程序.
        .                       ; 0004H ~ 0007H 保留

        ORG     08
        JMP     MY_IRQ          ; 0008H, 跳转到中断服务程序

        ORG     10H
START:                          ; 0010H, 用户程序起始地址
        .                       ; 用户程序
        .
        JMP     START           ; 用户程序结束

MY_IRQ:                         ;中断服务程序的起始地址
        B0XCH   A, ACCBUF       ; B0XCH 指令不会影响标志位 C，Z
        PUSH                    ; 保存工作寄存器
        .
        POP                     ; 恢复工作寄存器
        B0XCH   A, ACCBUF
        RETI                    ; 中断返回

        ENDP                    ; 程序结束
```

➢ **注：从上面的程序中很容易得出 SONIX 的主要编程规则，有以下几点：**

1. **地址 0000H 处的 “JMP” 指令使程序从头开始执行，0004H~0007H 区域由系统保留，用户必须跳过 0004H~0007H；**
2. **中断服务从 0008H 开始，用户可以将中断子程序的入口地址放在 0008H（例 1）或者在 0008H 写入一个跳转指令（例 2），而将整个中断子程序位于通用 ROM 区，从而形成模块化编程风格。**

3.1.4通用程序存储区

位于 ROM 中 0010H~08FBH 的 2028 个 word 单元作为通用程序存储区，这一区域主要用来存储指令操作代码和查表数据。SN8P2700A 中包括通过程序计数器实现的跳转表程序和通过寄存器（R、Y、Z）实现的查表程序。

在对程序计数器进行操作导致 PCL 溢出时，PCH 不会自动加 1，因此在跳转表程序和查表程序中，计数器不会自动跳过边界，当 PCL 发生溢出（从 0FFH 增至 000H）时，用户必须自行将 PCH 调整为 PCH+1。

3.1.5查表功能说明

在 ROM 的查表功能程序中， Y 寄存器指向地址的高 8 位，Z 寄存器指向低 8 位地址，执行 MOVC 指令后，数据的低字节存入累加器 ACC 中，而数据的高字节存入 R 寄存器中。

➲ 例：查找位于 “table1” 的 ROM 数据.

```
                B0MOV   Y, #TABLE1$M    ; 取得表格地址高字节
                B0MOV   Z, #TABLE1$L    ; 取得表格地址低字节
                MOVC                    ; 查表，R = 00H，ACC = 35H
                                        ;
                                        ; 索引地址加 1
                INCMS   Z               ; Z+1
                JMP     @F              ; 无进位
                INCMS   Y               ; Z 溢出(FFH → 00)，→ Y=Y+1
                NOP                     ;
                                        ;
@@:             MOVC                    ; 查表，R = 51H，ACC = 05H..
                .       .               ;
TABLE1:         DW      0035H           ; 定义表格数据（16 位）
                DW      5105H           ; “
                DW      2012H           ; “
```

- **注：当 Z 寄存器从 0XFFH 增至 0X00H，跨越页边界时，Y 寄存器不会自动增加。所以，用户必须注意处理这种情况，避免查表错误。如果 Z 寄存器发生溢出，Y 寄存器必须增 1。下面给出的宏指令 INC_YZ 提供了解决此问题的方法。**
- **注：因为程序计数器 PC 只有 12 位, X 寄存器在查表的时候实际是没有用处的，用户可以省略“B0MOV X, #TABLE1$H”。SONIX ICE 能够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              ; 无进位
                INCMS   Y               ; Y+1
                NOP                     ; 无进位
@@:
                ENDM
```

另一种方法是通过累加器来增加间接寄存器 Y 和 Z，但要注意是否有进位发生。下面的例子给出了详细操作步骤。

➲ 例：执行指令 B0ADD/ADD 增加 Y、Z

```
                B0MOV   Y, #TABLE1$M    ; 取得查表地址高字节
                B0MOV   Z, #TABLE1$L    ; 取得查表地址低字节
                B0MOV   A, BUF          ; Z = Z + BUF.
                B0ADD   Z, A
                B0BTS1  FC              ; 检查进位标志 C
                JMP     GETDATA         ; FC = 0
                INCMS   Y               ; FC = 1. Y+1.
                NOP
GETDATA:                                ;
                MOVC                    ; 查表，若 BUF = 0，结果是 0x0035
                                        ; 若 BUF = 1，结果是 0x5105
                                        ; 若 BUF = 2，结果是 0x2012
                .       .               ;
TABLE1:         DW      0035H           ; 定义一个 Word（16 位）的表格数据
                DW      5105H           ; “
                DW      2012H           ; “
```

3.1.6跳转表功能说明

跳转表操作可以完成多个地址跳转功能，将程序计数器的低字节 PCL 与累加器 ACC 相加从而得到一个指向新的跳转地址的程序计数器值，这种方法可以方便多个任务的处理。

执行 “ADD PCL, A”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 256words。**

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

```
        ORG     0X0100          ; 跳转表最好放在 ROM 的边界位置

        B0ADD   PCL, A          ; PCL = PCL + ACC,但 PCH 不会改变
        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
```

在下面的例子中，跳转表格从 0x00FD 开始，执行“B0ADD PCL，A”时，如果 ACC = 0 或 1，跳转表格指向正确的地址，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 0x0000，程序失效。 因此，检查跳转表格是否跨越边界(xxFFH 到 xx00H)非常重要。良好的编程风格是将跳转表格放在 ROM 的开始边界（如 0100H）。

➲ **例：如果跳转表格跨越 ROM 边界，将引起程序错误：**

```
ROM Address
    .           .
    .           .
0X00FD  B0ADD   PCL, A      ; PCL = PCL + ACC, PCH 的值不能改变.
0X00FE  JMP     A0POINT     ; ACC = 0
0X00FF  JMP     A1POINT     ; ACC = 1
0X0100  JMP     A2POINT     ; ACC = 2  ←跳转表跨越边界
0X0101  JMP     A3POINT     ; ACC = 3
    .           .
```

SONIX 提供了一条宏指令以保证安全的跳转表操作，这条宏指令会检查 ROM 的边界，并自动将跳转表移动到正确的位置。但宏指令会占用 ROM 的存储空间。

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

➢ **注：“VAL”为跳转表的个数。**

➲ **例：“@JMP_A” 在 SONIX 编译软件中的宏文件“MACRO3.H”中.**

```
        B0MOV   A, BUF0         ; “BUF0” 的值为 0-4
        @JMP_A  5               ; 要跳转的总的地址数是 5.
        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
        JMP     A4POINT         ; ACC = 4，跳转到 A4POINT
```

如果跳转表格的位置是从 00FDH 到 0101H，那么宏指令 “@JMP_A” 将使跳转表格从 0100h 开始。

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 * 8-bit 通用存储器(bank 0)
- 128 * 8-bit 系统专用寄存器

该存储器位于 bank0 单元，前 128-byte 作为通用存储区，后 128-byte 作为系统寄存器。

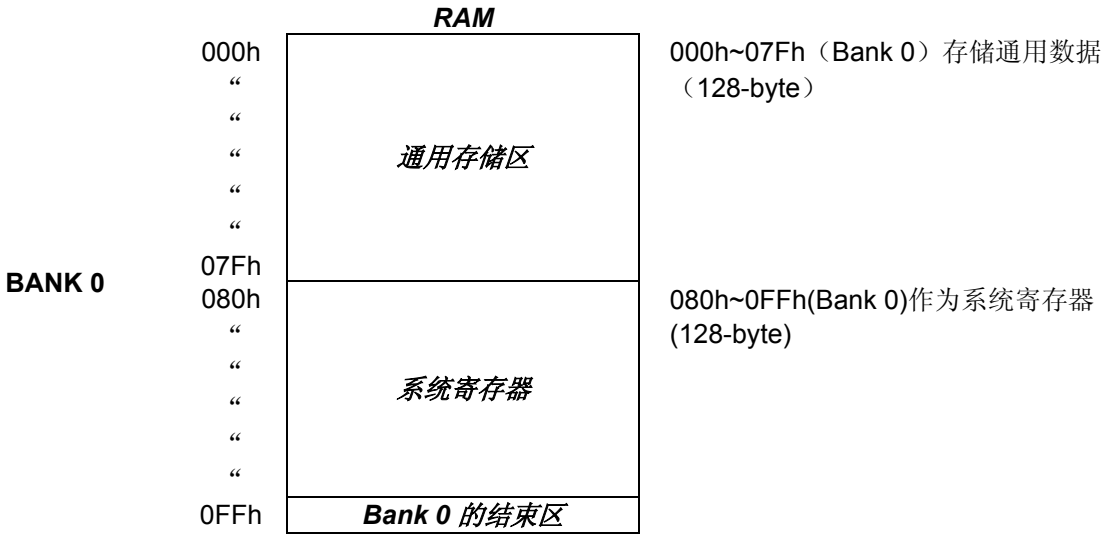


图 3-2 SN8P2710 的 RAM 区域划分

- **注：在执行读指令”MOV A, M”后，系统寄存器中未定义的未置为逻辑“高”。**

3.3 工作寄存器

RAM bank 0 中 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用来访问 ROM 和 RAM 中的数据。例如，所有的 ROM 列表可以通过 R、Y 和 Z 寄存器查找，RAM 则可由 Y 和 Z 寄存器间接访问。

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

3.3.1 Y, Z 寄存器

8 位寄存器 Y 和 Z 寄存器主要用作系统系统专用寄存器、@YZ 间接寻址和查表寻址寄存器。

Y 初始值 = 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 初始值 = 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 bank0 中 E7H 单元，通过 Y 和 Z 寄存器的内容可访问 RAM 数据，并可通过累加器 ACC 对该数据进行读/写。Y 寄存器的低 4 位决定了单元的 RAM 页，Z 寄存器给出单元在 RAM 某页中的具体地址。Y 的高 4 位在 RAM 间接寻址模式中无意义。

➲ **例：间接寻址访问 RAM bank 0 的 25H 单元**

```
B0MOV   Y, #00H     ; 把 BANK0 的 RAM 高 8 位地址送到 Y 寄存器
B0MOV   Z, #25H     ; 把 BANK0 的 RAM 低 8 位地址送到 Z 寄存器
B0MOV   A, @YZ      ; 读上面地址对应储存器的值到 ACC
```

➲ **例：用@YZ 对 RAM bank 0 清零**

```
        MOV     A, #0
        B0MOV   Y, A            ; Y = 0, bank 0
        MOV     A, #07FH
        B0MOV   Z, A            ; Z = 7FH，数据存储器的低地址
CLR_YZ_BUF:
        CLR     @YZ             ; 清零

        DECMS   Z               ; Z – 1，Z= 0 结束子程序
        JMP     CLR_YZ_BUF      ; 不为零继续计算

        CLR     @YZ
END_CLR:                        ; 结束通用区所有存储器的清零程序
```

➢ **注：关于 Y, Z 寄存器的查表应用请参考 “查表程序描述” 。**

3.3.2 R 寄存器

寄存器 R 是一个 8 位缓存器，可作为工作寄存器或在查找 ROM 数据时存放数据的高字节。执行 MOVC 指令后，ROM 数据的高字节存入 R 寄存器中，低字节存入累加器 ACC 中。

R 初始值 = 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 寄存器的查表应用请参考 “查表程序描述”**

3.4 程序状态寄存器(PFLAG)

程序状态寄存器 PFLAG 包括进位标志(C)，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 = xxxx x000

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	-	-	FLVD24	C	DC	Z
	R/W	R/W	-	-	R/W	R/W	R/W	R/W

3.4.1复位标志

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

3.4.2 LVD 2.4V FLAG

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

注：该位仅在 LVD = LVD1 时有效。

3.4.3进位标志

C = 1：执行算术加法操作后有进位发生，执行算术减法后没有借位或移位指令后移出逻辑 “1”。
C = 0：执行算术加法操作后没有进位发生，执行算术减法后有借位或移位指令后移出逻辑 “0”。

3.4.4辅助进位标志

DC = 1：执行算术加法操作产生由低字节向高字节的进位或执行算术减法操作没有从高字节借位。
DC = 0：执行算术加法操作没有产生由低字节向高字节的进位或执行算术减法操作从高字节借位。

3.4.5零标志

Z = 1：指令执行后，累加器 ACC 的结果为零。
Z = 0：指令执行后，累加器 ACC 的结果非零。

3.5 累加器(ACC)

累加器 ACC 是一个 8 位专用数据寄存器，用来进行算术逻辑运算或数据存储器之间数据的传送和处理。如果对 ACC 的操作结果为零（Z）或者有进位产生（C 或 DC），那么这些标志将会影响 PFLAG 寄存器。

由于 ACC 不在数据存储器（RAM）中，所以在立即寻址模式下，执行 “B0MOV” 指令不能够访问 ACC。

➲ **例：读/写 ACC 中数据**

```
; 把 ACC 中断数据送到 BUF 中

                MOV         BUF, A

; 给 ACC 送立即数

                MOV         A, #0FH

; 把 BUF 中的数据送到 ACC 中

                MOV         A, BUF
```

中断发生时，PUSH 和 POP 指令不会自动保存 ACC 的值，因此必须对它进行保护和恢复。一旦中断发生，必须把 ACC 存储在用户自定义的存储器中，如下所示：

➲ **例：保护 ACC 和工作寄存器。**

```
ACCBUF          EQU         00H             ;

INT_SERVICE:
                B0XCH       A, ACCBUF       ; B0XCH 不会影响 C、Z 标志
                B0MOV       A, PFLAG
                B0MOV       PFLAGBUF, A     ; 保存 PFLAG 的值
                .
                .
                B0MOV       A, PFLAGBUF
                B0MOV       PFLAG, A        ; 恢复 PFLAG 和 ACC 的值
                B0XCH       A, ACCBUF

                RETI                        ; 退出中断服务程序
```

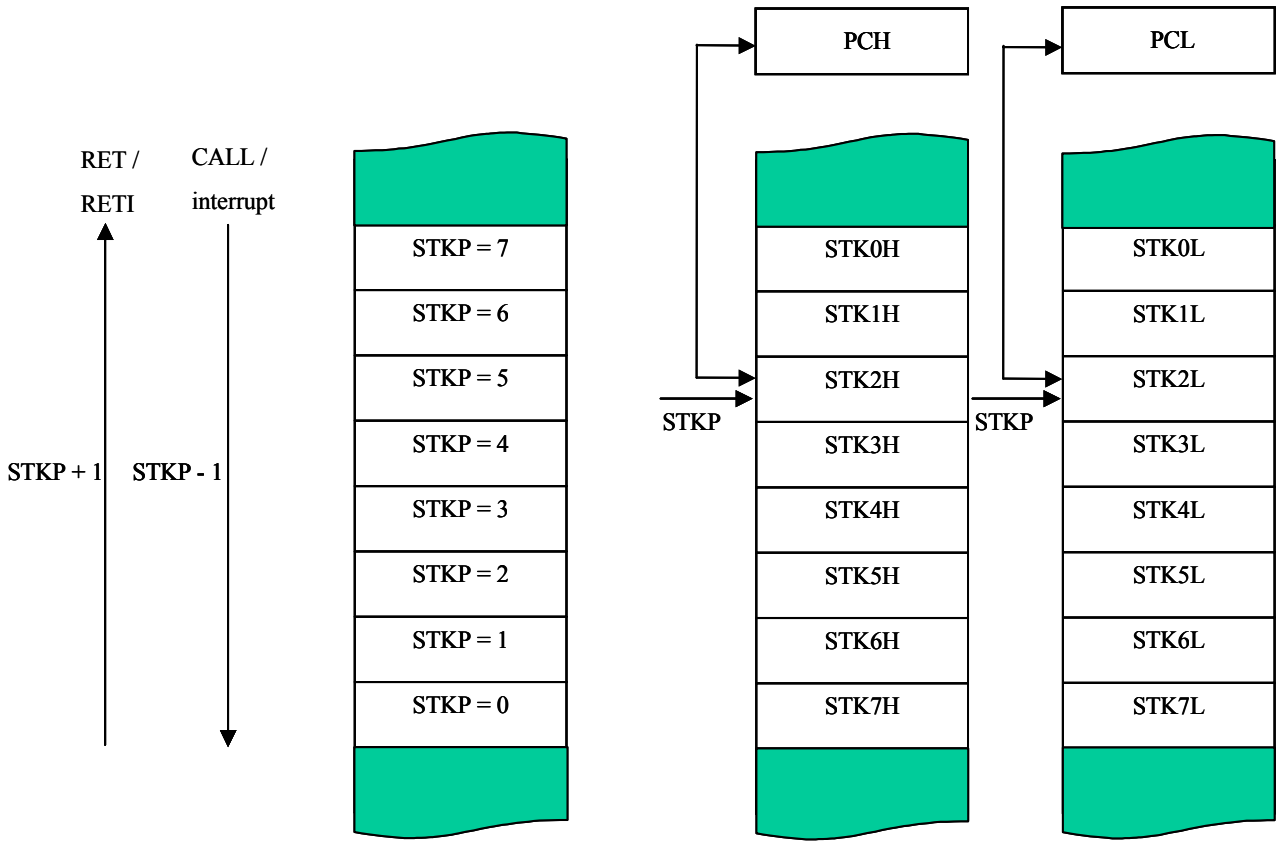
➢ **注：必须用指令”B0XCH”保存和恢复 ACC，否则会影响 PFLAG。**

3.6 堆栈操作

3.6.1概述

SN8P2710 共有 8 层堆栈。堆栈缓存器 STKnH 和 STKnL 在中断现场保护和恢复时用来存放程序计数器 PC 的数据。堆栈指针 STKP 指示当前栈顶位置以便保护和恢复数据。

STACK BUFFER



3.6.2 堆栈指针寄存器

堆栈指针 STKP 是一个 4 位寄存器，用来指示堆栈栈顶位置，12 位的数据存储器（STKnH 和 STKnL）用来存储程序计数器（PC）的值。

堆栈操作有两种：入栈（PUSH）和出栈（POP）。执行入栈（PUSH）操作，STKP 减一，执行出栈（POP）操作，STKP 加一。这样，STKP 总是指向栈顶位置。

执行 CALL 指令和响应中断时，程序计数器（PC）的值会被保存在堆栈中，堆栈操作遵循后进先出的原则。堆栈指针寄存器（STKP）和缓存器 STKnH、STKnL 位于 BANK0。

STKP (堆栈指针)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 ~ 3)
GIE：总中断控制位。0=禁止，1=使能。详见中断章节。

➲ 例：堆栈指针(STKP)复位

```
MOV     A, #00001111B
B0MOV   STKP, A
```

STKn (堆栈缓存器)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	-	-	-	-	SnPC11	SnPC10	SnPC9	SnPC8
	-	-	-	-	R/W	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：在执行中断或 CALL 指令时存储 PCH 的数据。n：8 ~11
STKnL：在执行中断或 CALL 指令时存储 PCL 的数据。n：0 ~7

3.6.3 堆栈操作举例

程序调用指令（CALL）和中断都涉及到对堆栈指针 STKP 的操作和将程序计数器 PC 保存在堆栈缓冲区。在两种操作中，堆栈指针 STKP 都会减 1 并指向下一个可用的堆栈区域，堆栈缓存器中则保存了程序指针。入栈保护操作如下表所示：

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)

程序计数器 PC 是一个 11 位专用二进制计数器，分为 3 位高字节和 8 位低字节，PC 总是指向下一条将要访问指令的地址，一般在程序执行过程中，PC 值会随着指令的执行自动增量。

执行程序调用（CALL）和跳转（JMP）指令时，下一条将要执行指令的地址会被装入 PC 的 0~10 位。

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

单地址跳转指令共有 7 条：CMPRS、INCS、INCMS、DECS、DECMS、B0BTS0、B0BTS1。如果运算的结果符合条件，则 PC 加 2，跳过当前指令的下一条指令。

➲ **如果位测试结果匹配，那么 PC 加 2，跳过当前指令的下一条指令：**

```
            B0BTS1    FC          ; 如果 C=1 则跳过下一条指令
            JMP       C0STEP      ; 否则调转到 C0STEP.
            .
C0STEP:     NOP

            B0MOV     A, BUF0     ; 把 BUF0 的值赋给 ACC.
            B0BTS0    FZ          ; 如果 Z=0，则跳过下一条指令
            JMP       C1STEP      ; 否则跳到 C1STEP.
            .
C1STEP:     NOP
```

➲ **如果 ACC 与立即数或存储器中的内容相等，那么 PC 加 2，跳过当前指令的下一条指令**

```
            CMPRS     A, #12H     ; 如果 ACC = 12H. 则跳过下一条指令
            JMP       C0STEP      ; 否则跳到 C0STEP.
            .
C0STEP:     NOP
```

➲ **如果加 1（INCS、INCMS）/减 1（DECS、DECMS）后的结果为 0x00H/0xffH,那么 PC 加 2 跳过当前指令的下一条指令。**

```
INCS:
            INCS      BUF0        ; 如果 ACC 为 0，则跳过下一条指令
            JMP       C0STEP      ; 否则跳到 C0STEP.
            .
C0STEP:     NOP
INCMS:
            INCMS     BUF0        ; 如果 BUF0 为 0，则跳过下一条指令
            JMP       C0STEP      ; 否则跳到 C0STEP.
            .
C0STEP:     NOP
DECS:
            DECS      BUF0        ; 如果 ACC 为 0，则跳过下一条指令
            JMP       C0STEP      ; 否则跳到 C0STEP.
            .
C0STEP:     NOP
DECMS:
            DECMS     BUF0        ; 如果 BUF0 为 0，则跳过下一条指令
            JMP       C0STEP      ; 否则跳到 C0STEP.
            .
C0STEP:     NOP
```

3.7.2 多地址跳转

用户可以通过 JMP 和“ADD PCL，A”指令实现多地址跳转。“ADD PCL，A”执行后若有进位发生，进位标志并不会影响 PCH 寄存器。

➲ **例： 如果 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： 如果 PC = 0323H (PCH = 03H、PCL = 23H)**

```
; PC = 0323H
                B0ADD   PCL, A          ; PCL = PCL + ACC，PCH 的值不会改变.
                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 提供了三种 RAM 寻址模式：立即寻址模式、直接寻址模式和间接寻址模式。这 3 种不同寻址模式的应用将在下面描述。

4.1.1 立即寻址

将一个立即数送入累加器或指定的 RAM 单元：MOV A, # I ； B0MOV M, # I

立即寻址模式

```
MOV     A, #12H         ; 立即数 12H 存入 ACC
```

4.1.2 直接寻址

通过单元地址访问存储器：MOV A, 12H ； MOV 12H, A

直接寻址模式

```
B0MOV   A, 12H          ; bank 0 中 12H 的数据送入 ACC
```

4.1.3 间接寻址

目的地址存放在指针寄存器（Y/Z）中，利用 MOV 指令在 ACC 和寄存器@YZ 之间读/写数据：

```
MOV     A, @YZ  ;  MOV  @YZ, A
```

➲ **例：对@YZ 间接寻址：**

```
CLR     Y       ; 清 Y，指向 RAM bank 0.
B0MOV   Z, #12H ; 送一个立即数 12H 到 Z
B0MOV   A, @YZ  ; 用数据指针 @YZ 取得相应存储器的地址 012H 的数据送给 ACC
```

4.1.4 寻址 RAM BANK 0

RAM bank 0 的单元都可以通过下面两种寻址方式进行读/写：

➲ **例 1：直接寻址（如 B0xxx 指令）**

```
B0MOV   A,  12H         ; 赋一个地址 12H（bank 0）对应的数到 ACC
```

➲ **例 2：@YZ 间接寻址**

```
CLR     Y       ; 清 Y，指向 RAM bank 0.
B0MOV   Z, #12H ; 送一个立即数 12H 到 Z
B0MOV   A, @YZ  ; 用数据指针 @YZ 取得相应存储器的地址 012H 的数据送给 ACC
```

5 系统寄存器

5.1 概述

RAM 中 bank0 的 80H~FFH 区域被留做系统专用寄存器，用来控制芯片的内部硬件资源，如输入/输出口状态、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基准源。

5.2 系统寄存器配置(BANK 0)

5.2.1系统寄存器的字节

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 的系统寄存器的地址分配配置表

说明

PFLAG = ROM 页，特殊标志寄存器
DAM = DAC 模式寄存器
ADB = ADC 数据缓存器
PnM = 输入输出模式寄存器
INTRQ = 中断请求寄存器
OSCM = 振荡器寄存器
T0M = 定时/计数器 0，定时/计数器 1 的速度选择
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 配置设置寄存器

R = 工作寄存器和 ROM 查表数据寄存器
Y, Z = 工作寄存器，@YZ 和 ROM 寻址寄存器
ADM = ADC 模式寄存器
ADR = ADC 精度设置寄存器
Pn = Pn 口数据缓存器
INTEN = 中断使能寄存器
PCH, PCL = 程序计数器
TC0M = TC0 模式寄存器
TC0C = TC0 数据寄存器
TC0R = TC0 自动加载数据寄存器
TC1R = TC1 自动加载数据寄存器
STK0~STK7 = 堆栈缓存器
@YZ = 间接寻址寄存器

➢ 注：：

a).所有的寄存器名称在 SONIX 8 位 MCU 编译器中是默认的；.
b).寄存器中各位的名称已经在 SONIX 8 位 MCU 编译器中以“F”为前缀定义过；
c).用空指令检验空单元时，返回逻辑“1”；
d).ADR 寄存器的低字节为只读寄存器；
e). “b0bset”，“b0bclr”，”bset”，”bclr”指令仅对“R/W”寄存器有效。

5.2.2 系统寄存器位地址配置表

SN8P2710 系统寄存器列表

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	-	-	LVD24	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	-	-	-	-	-	-
0AFH	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	P2 口方向控制寄存器
0C3H	-	-	-	-	-	-	-	-	-	
0C4H	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4 口方向控制寄存器
0C5H	-	P56M	P55M	P54M	P53M	P52M	P51M	P50M	R/W	P5 口方向控制寄存器
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
0E9H	-	-	-	-	-	-	-	-	-	
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H	-	-	-	-	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H	-	-	-	-	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H	-	-	-	-	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H	-	-	-	-	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H	-	-	-	-	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH	-	-	-	-	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH	-	-	-	-	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH	-	-	-	-	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

表 **SN8P2710 系统寄存器的位地址配置表**

➢ **注：**

a). **为避免系统出错，程序中使用到上表中所有标“0”的位时要设定为 “0”；**

b). **所有的寄存器名称在 SONIX 8 位 MCU 编译器中是默认的；**

c). **寄存器中各位的名称已在 SONIX 8 位 MCU 汇编语言中以“F”为前缀定义过；**

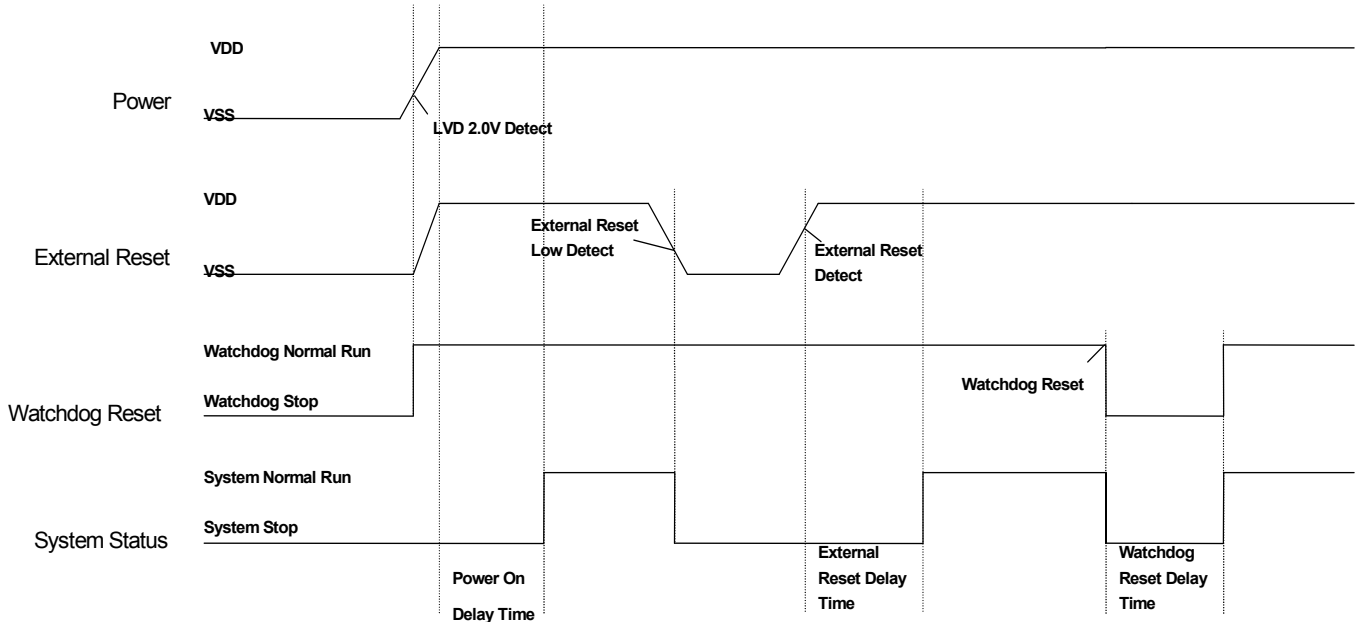
d). 指令“b0bset”、“b0bclr”、”bset”、”bclr”仅对“R/W”寄存器有效；

e). 详见“系统寄存器的简捷参照表”。

6 上电复位

6.1 概述

SN8P2710 有三种系统复位方式：上电复位、外部复位和看门狗复位。当其中任何一个复位信号产生时，系统复位并初始化系统寄存器，时序图如下：



6.2 上电复位

接通电源当电源从 VSS 上升到 VDD 时，系统就会复位，进入普通模式，从上电到系统开始运行的时间叫“上电延迟时间”。系统在上电延迟时间后开始正常运行。

VDD	晶振	上电延迟时间
3V	4MHz	大约 129ms
5V	4MHz	大约 65ms
3V	32768Hz	大约 237ms
5V	32768Hz	大约 173ms

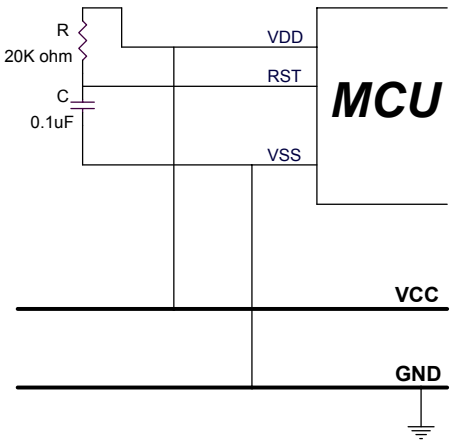
6.3 外部复位

外部复位引脚为施密特触发结构，当它设置为复位引脚时，就使能了外部复位功能。外部复位为低电平有效，当复位引脚侦测到低电压（低于 0.3VDD）时，系统复位；直到侦测到的电压达到高电平（高于 0.7VDD）。用户可以使用外部复位电路来控制系统。

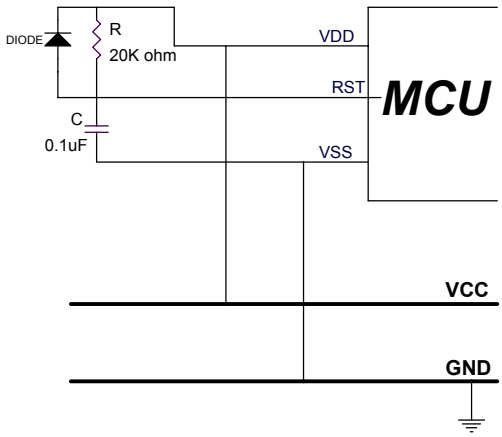
VDD	*晶振*	*外部复位延迟时间*
3V	4MHz	大约 129ms
5V	4MHz	大约 65ms
3V	32768Hz	大约 237ms
5V	32768Hz	大约 173ms

6.3.1外部复位电路

外部复位电路是一个简单的 RC 电路，如下所示：



简单的 RC 复位电路



加二极管后的节电复位电路

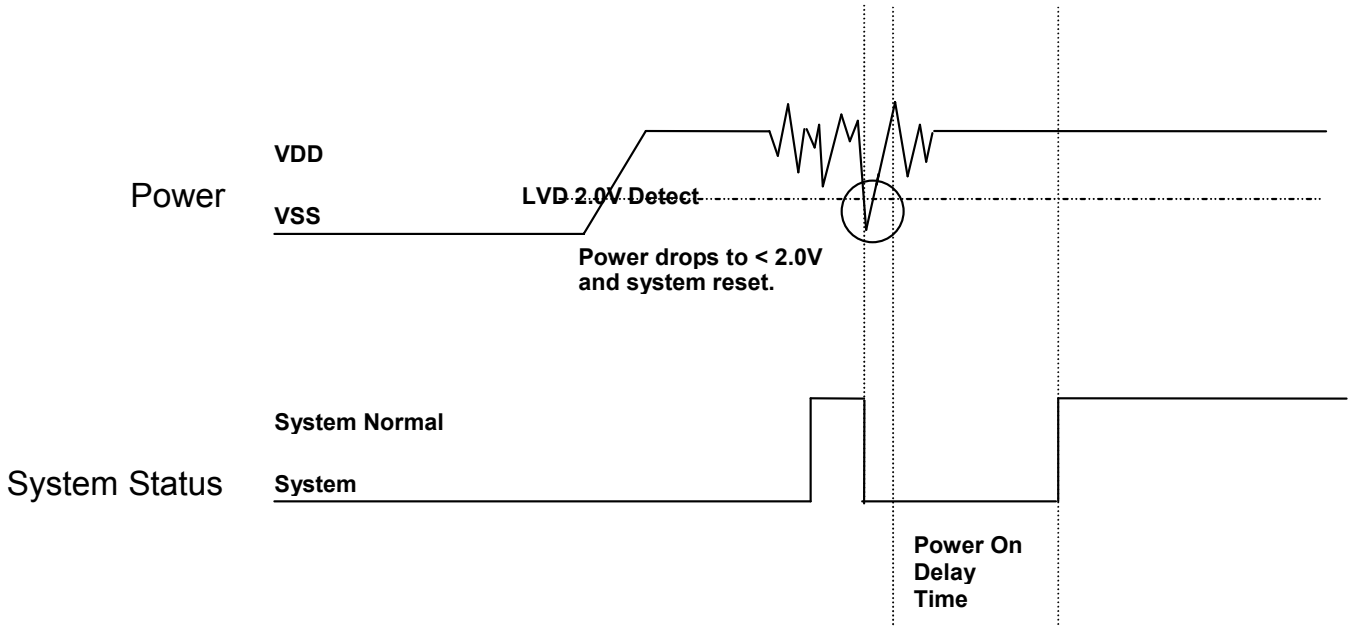
6.4 看门狗复位

看门狗复位是对系统的一种保护。在正常情况下，系统可以可以由程序对看门狗清零保证正常运行；但在错误状态下，系统位于未知状态，在看门狗定时器溢出前无法由程序对看门狗清零。看门狗定时器溢出后，系统复位，系统重启并进入普通模式。

VDD	晶振	看门狗复位延迟时间
3V	4MHz	大约 145ms
5V	4MHz	大约 73ms
3V	32768Hz	大约 253ms
5V	32768Hz	大约 181ms

6.5 低电压侦测(LVD)

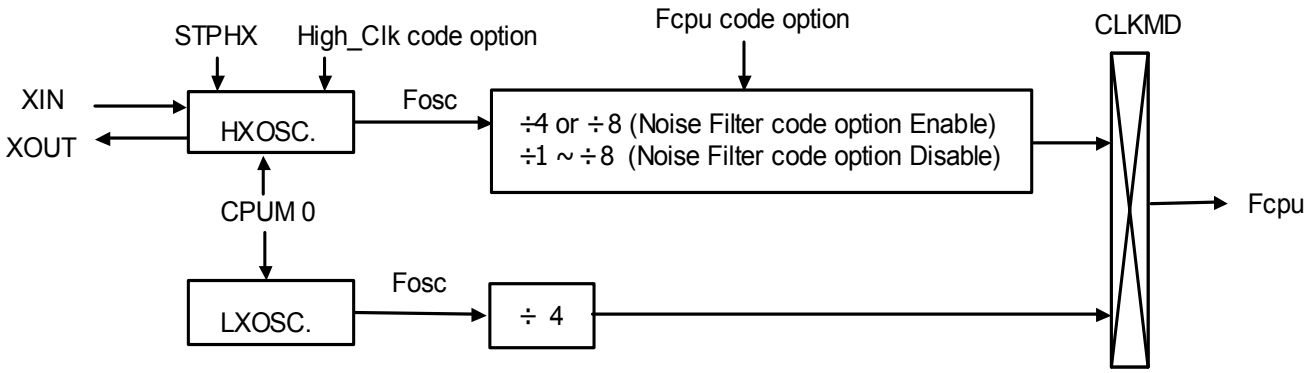
LVD 为低电压侦测，当侦测到 VDD 低于设定值时，系统就会复位。低电压侦测的电压值为 2.0V，如果 VDD 低于 2.0V，系统就会复位。如果 VDD 返回到 2.0V 以上，系统开始执行上电程序，在上电延迟时间结束后，系统开始正常运行，这叫做 BROWN OUT 复位。在高干扰环境下，上电会有些不稳定，LVD 可以防止噪音影响系统工作。当 VDD 保持 2.0V 以上的电压时，系统就会保持稳定的工作状态；当 VDD 低于 2.0V 时，LVD 使系统复位并等待 VDD 返回到正常电压。LVD 复位延迟就是上电复位延迟时间。



7 振荡器

7.1 概述

SN8P2710 系列是具有高速时钟和低速时钟的双时钟微控制器。高速时钟由外部振荡电路提供，低速时钟由芯片内部 RC 振荡电路提供。



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

下面是需要用到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)***
- ✓ ***AD 转换***

7.1.1 振荡寄存器

振荡器控制寄存器 OSCM 决定了系统振荡源、系统模式以及看门狗计数器的时钟源等。

OSCM 初始值 = xxx0 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 [4:3] **CPUM 0**：CPU 操作模式控制位
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/低速时钟模式选择位（Fcpu）
0 Fcpu = 外部高速时钟(Fhosc)，操作模式为普通模式。
1 Fcpu = 内部低速时钟(Flosc)，操作模式为普通模式。

Bit1 **STPHX**：高速时钟控制位
0=高速时钟正常运行
1=高速时钟停止

7.1.2 外部高速振荡器

SN8P2710 系列可以工作于四种振荡器模式：RC 振荡器模式、外部高速振荡模式（12M 编译选项）、标准振荡模式（4M 编译选项）和低速振荡模式（32K 编译选项）。在不同的应用场合，用户可通过编译选项，为工作系统选择适当的高速时钟源。

➲ **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/内部低速振荡器，并将系统设定进入到睡眠模式
```

7.1.3 高速时钟振荡器编译选项

SN8P2710 在不同的应用下提供 4 种振荡模式，分别为：4M, 12M, 32K 和 RC，以支持不同的振荡器类型和频率。MCU 运行高速时钟系统比运行低速时钟系统需要的电流要多。对于晶振，有 3 个选择项。在编译前，用户可以从编译选项表中选择振荡器的类型。编译选项如下表所示：

编译器	振荡器模式	功能说明
High_Clk	Ext_RC	廉价外部 RC 振荡器，从 XOUT 引脚输出频率为 Fcpu 的方波。
	32K_X’tal	低频、低功耗晶体振荡器（如 32.768KHz）
	12M_X’tal	高速晶体振荡器（如 12MHz ~ 16MHz）
	4M_X’tal	标准晶体振荡器（如 4MHz ~ 10Mhz）

7.1.4 系统振荡器电路

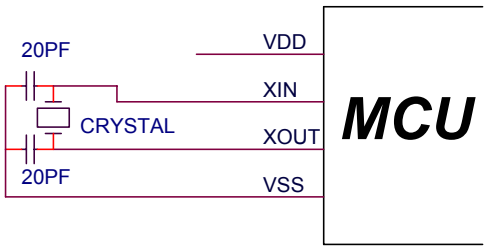


图 7-1. 晶体振荡器/陶瓷谐振器

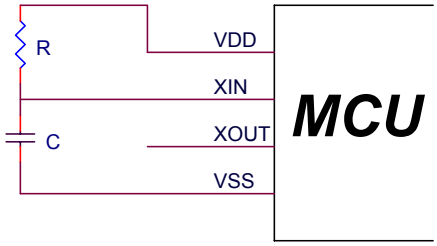


图 7-2. RC 振荡器

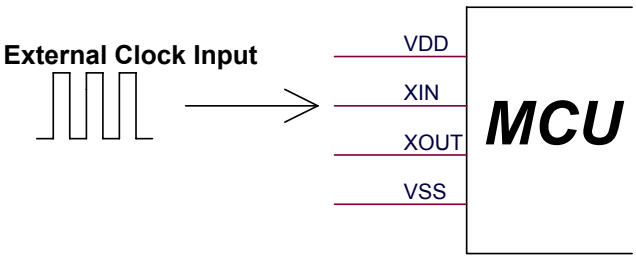


图 7-3. 外部时钟输入

- **注 1：外部振荡电路的 VDD 和 VSS 必须来自微控制器，而不是相邻电源端。**
- **注 2：外部时钟输入可以选择 RC 振荡器或晶体振荡器，产生的时钟由 XIN 引脚输入。**
- **注 3：外部振荡器的电源端和接地端必须和微控器的 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

可以由指令周期 Fcpu 得到外部 RC 振荡器的频率 Fosc，测试外部振荡器频率 Fcpu 的程序如下：

➲ **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 RC 模式下输出频率信号
        B0BCLR    P2.0      ;
        JMP       @B
```

7.2 内部低速振荡器

内部低速振荡器是一个芯片内置电路，它的时钟源由 RC 振荡电路提供，可提供系统时钟、定时计数器、看门狗定时器等的时钟源。

➲ **例：停止内部低速振荡器**

```
B0BSET    FCPUM0    ;停止外部高速/内部低速振荡器并将系统设定为睡眠（省电）模式
```

➢ **注：内部低速时钟不能单独停止，它由 OSCM 振荡器的 CPUM0 位控制。**

低速振荡器采用 RC 振荡电路，频率会受系统电压和温度的影响。通常情况下，RC 振荡器的频率约为 16KHz（3V）、32KHz（5V）。RC 频率和电压之间的关系如下图所示：

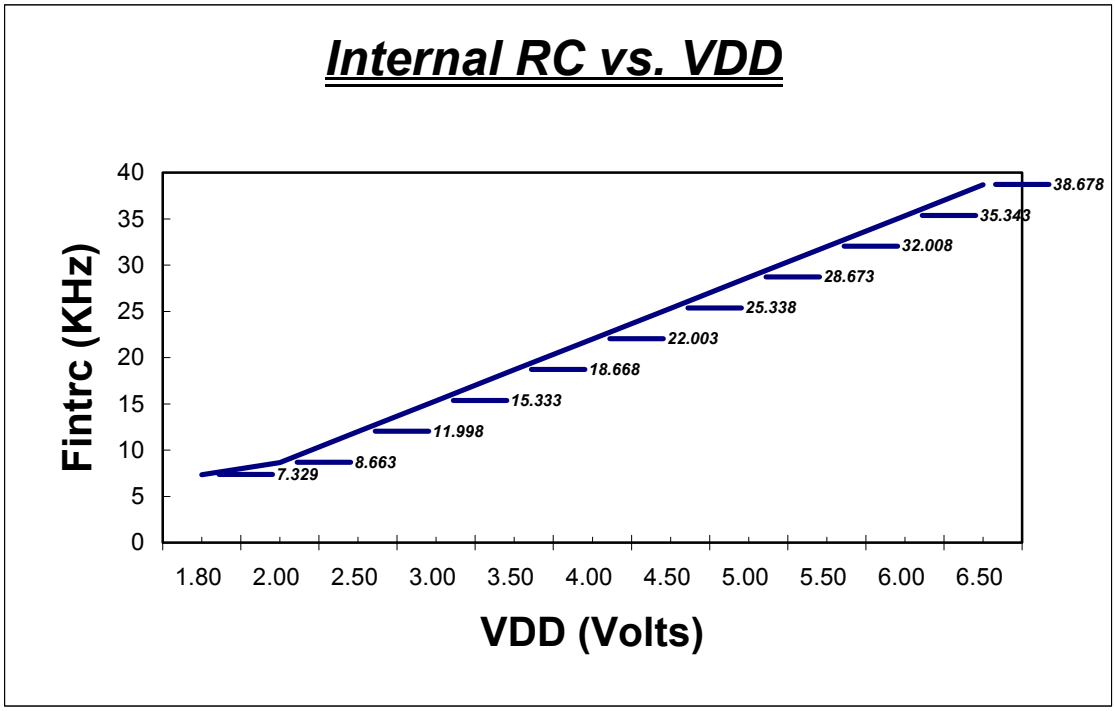


图 7-4. 内部 RC 和 VDD 的关系图

➲ **例：由 Fcpu 测试内部 RC 频率。内部 RC 频率等于 4 倍的 Fcpu。我们可以从 Fcpu 得到内部 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      ; 在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

SN8P2700 A 能够在如下 4 种不同的模式中进行切换：

- 普通（高速）模式
- 低速模式
- 省电（睡眠）模式

实际应用中，用户可以利用 OSCM 寄存器调整系统的工作模式。

7.3.2 普通模式

普通模式中，系统时钟源为外部高速时钟。系统上电时，系统默认为普通模式。所有的软件和硬件都能够在正常模式下运行，系统可转入睡眠模式和低速模式。

7.3.3 低速模式

低速模式时，系统时钟源为内部低速时钟。设置 CLKMD=1，系统就进入内部低速模式。低速模式下的运行与普通模式的工作状态相似，仅是时钟频率有所降低。系统可在低速模式下转入普通模式和睡眠模式。设置 STPHX=1，可以停止外部高速振荡器，系统功耗也会降低。

7.3.4 省电（睡眠）模式

省电模式也称睡眠模式，系统进入睡眠状态时，将停止工作，功耗低至近似于零。省电模式常用于电池供电等节电系统。设 CPUM0=1，系统进入省电模式，外部高速和内部低速振荡器均停止，P0、P1 的触发信号可将系统唤醒。

注：

- **Watch_Dog code option = Enable**
  - **在睡眠（省电）模式下停止。**
  - **在普通模式下使能。**
- **Watch_Dog code option = Always_ON**
  - **在普通模式和省电（睡眠）模式下使能。**

7.4 系统模式控制

7.4.1 SN8P2710 系统模式控制框图

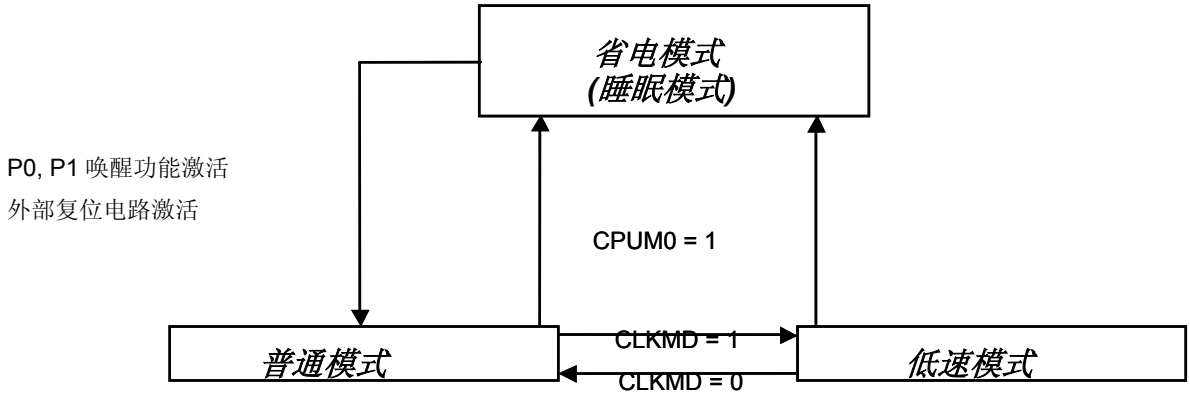


图 7-5. SN8P2710 系统模式框图

模式	普通模式	低速模式	省电（睡眠）模式	备注
HX osc.	运行	STPHX	停止	
LX osc.	运行	运行	停止	
CPU 指令	执行	执行	停止	
T0/TC0/TC1	*有效	*有效	无效	*由程序激活
看门狗定时器	有效	有效	由看门狗编译选项控制	
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	
看门狗时钟源	-	-	P0.0/P0.1，由 RST、LVD，*看门狗控制复位	* 看门狗编译选项必须是 “Always_ON”

表 7-1. 操作模式说明

7.4.2 系统模式转换

普通/低速模式转换为省电（睡眠）模式

CPUM0 = 1

```
        B0BSET      FCPUM0          ; 设置 CPUM0 = 1.
```

系统在睡眠模式中，只有具有唤醒功能的引脚和复位引脚可将系统唤醒。

普通模式转换为低速模式

```
        B0BSET      FCLKMD          ; 设置 CLKMD = 1，进入低速模式
        B0BSET      FSTPHX          ; 停止高速振荡器以省电
```

- **注：用户可以不用停止高速振荡器。**

低速模式转换为普通模式(外部高速振荡器仍然运行)

```
        B0BCLR      FCLKMD          ; 设置 CLKMD = 0
```

低速模式转换为普通模式（外部高速振荡器停止）

如果外部高速时钟停止时程序欲返回到普通模式，需延迟 10ms 以等待外部时钟稳定。

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

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

7.5 唤醒时间

7.5.1 概述

外部高速振荡器从停止到运行需要一段时间的延迟，这段延迟时间对振荡器的稳定工作是必需的。在有些应用中，外部高速振荡器可能需要经常的开停。外部高速振荡器重新启动需要的这一延迟时间称为唤醒时间。

有两种情况需要唤醒时间：一是从省电模式转换到普通模式，二是从低速模式转换到普通模式。对前一种情况，SN8P2710 提供了 4096 个振荡周期作为唤醒时间，后一种情况需要用户提供唤醒时间。

7.5.2 唤醒时间

当系统处于省电（睡眠）模式时，外部高速振荡器停止运行。从睡眠模式唤醒时，SN8P2710 提供 4096 个外部高速振荡周期作为唤醒时间，以使振荡电路达到稳定状态。唤醒时间结束后，系统进入普通模式，唤醒时间的计算方法如下：

唤醒时间 = 1/Fosc * 3584 (sec) + X'tal 稳定时间

x'tal 固定时间决定于 x'tal 的类型，一般约为 2~4ms。

➲ **例：在省电模式下，系统由 P0 或 P1 端的触发信号唤醒。唤醒时间结束后，系统进入普通模式，P0 和 P1 端唤醒时间如下：**

唤醒时间 = 1/Fosc * 3584 = 1.1001 ms (Fosc = 3.58MHz)

总的唤醒时间 = 1.1001ms + x'tal 稳定时间

省电（睡眠）模式下，具有唤醒功能的 P0 和 P1 都能将系统唤醒。

8 定时/计数器

8.1 看门狗定时器 (WDT)

看门狗计数器（WDT）是一个 4 位二进制计数器，用来监控程序的运行状态。若程序因干扰或程序失效进入未知状态，WDT 溢出促使微控制器复位。在系统不正常运行时，用户必须清看门狗以避免看门狗计数器溢出而导致系统复位。WDT 的时钟源由内部低速 RC 振荡器提供。WDT 的溢出时间见下：

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	-	-	-	-	-	-	-	-

设置 WDTR = 0x5A，清看门狗计数器

注：

- **在编译选项中可以设置看门狗计数器为“Always_ON”，“Enable”和“Disabled”三种状态。**
- **Watch_Dog = “Enable”**
  - **在睡眠（省电）模式下停止；**
  - **在普通模式下使能。**
- **Wztch_Dog = “Always_ON”**
  - **在普通模式和省电模式下，看门狗使能。**

➲ **例：看门狗计数器操作程序，在主程序的开始，清看门狗计数器。**

```
Main:
        MOV     A,#0x5A
        B0MOV   WDTR,A          ; 清看门狗计数器
        .       .
        CALL    SUB1
        CALL    SUB2
        .       .
        JMP     MAIN
```

- **注：在用 S8KD 的 ICE 做仿真时，请用宏指令 “@RST_WDT” 清看门狗。**

8.2 定时/计数器 (TC0)

8.2.1 概述

TC0 是一个 8 位二进制定时/计数器，用作通用定时器、buzzer 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，用于保存计数参考值；8 位自动加 1 的计数器 TC0C。

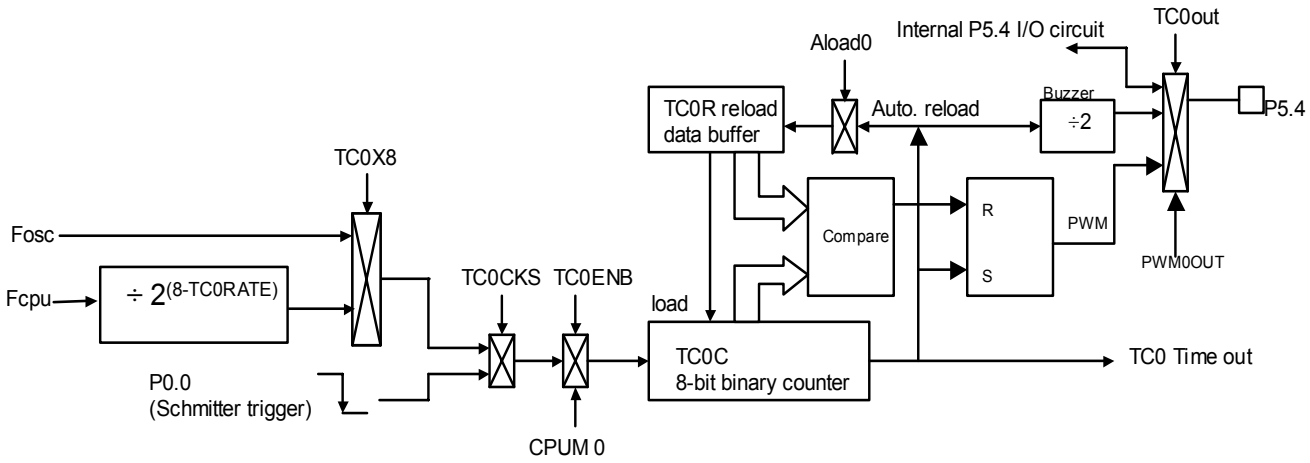


图 8-1. 定时/计数器 TC0 框图

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。
- **频率输出（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 PWM1OUT 位控制，通过 PWM0OUT 引脚（P5.4）输出。

8.2.2 TC0M 模式寄存器

TC0M 为 8 位可读/写模式控制寄存器。通过载入不同值，用户可以在执行程序过程中动态的调整定时器的时钟频率。

通过设置 TC0 的 TC0RATE0~TC0RATE2 及 TC0X8 位，定时器 TC0 提供了 8 种可选择的时钟源频率。若 TC0X8=1，TC0 的时钟频率初始值为 Fosc；当 TC0X8 = 0（初始化）时。时钟频率的范围为从 fcpu（Fosc）/2 到 fcpu（Fosc）/256。TC0M 的初始值为 0，对应的时钟源频率为 fcpu（Fosc）/256。TC0M 的第 7 位 TC0ENB 位是 TC0 的启动控制位。它们共同决定了 TC0 定时器的时钟源频率和定时间隔。

T0M 初始值 = xxxx 00xx

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	-	-	-	-	TC1X8	TC0X8		-
	-	-	-	-	R/W	R/W		-

Bit3 **TC1X8**：TC1 定时器速度的 8 倍。详见 TC1M 寄存器。
0 = 禁止
1 = 使能

Bit2 **TC0X8**：TC0 定时器速度的 8 倍。详见 TC0M 寄存器。
0 = 禁止
1 = 使能

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

TC0Rate	TC0X8=0	TC0X8=1
111	Fcpu/256	Fcpu/256x8
110	Fcpu/128	Fcpu/128 x8
101	Fcpu/64	Fcpu/64 x8
100	Fcpu/32	Fcpu/32 x8
011	Fcpu/16	Fcpu/16 x8
010	Fcpu/8	Fcpu/8 x8
001	Fcpu/4	Fcpu/4 x8
000	Fcpu/2	Fcpu/2 x8

Bit 3 **TC0CKS**：TC0 时钟源选择位
0 = 内部时钟源(fcpu)
1 = 来自 P0.0 (INT0)的外部时钟源

Bit 2 **ALOAD0**：自动装载控制位
0 = 禁止自动装载功能
1 = 使能自动装载功能

Bit 1 **TC0OUT**：TC0 超时信号输出控制位。**仅当 PWM0OU = 0 时有效。**
0 = 禁止 TC0OUT 信号的输出功能并使能 P5.4 的 I/O 功能。
1 = 使能 TC0OUT 信号的输出功能并禁止 P5.4 的 I/O 功能。

Bit 0 **PWM0OUT**：PWM 输出控制位。详见“PWM 功能说明”章节。
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(Fcpu = 4M)	备注
0	0	FFh to 00h	0/255 ~ 255/255	7.8125K	每计数 256 次溢出
0	1	3Fh to 40h	0/63 ~ 63/63	31.25K	每计数 64 次溢出
1	0	1Fh to 20h	0/31 ~ 31/31	62.5K	每计数 32 次溢出
1	1	0Fh to 10h	0/15 ~ 15/15	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)	备注
0	0	FFh to 00h	0/255 ~ 255/255	62.5K	每计数 256 次溢出
0	1	3Fh to 40h	0/63 ~ 63/63	250K	每计数 64 次溢出
1	0	1Fh to 20h	0/31 ~ 31/31	500K	每计数 32 次溢出
1	1	0Fh to 10h	0/15 ~ 15/15	1000K	每计数 16 次溢出

➢ **注 1：当 TC0CKS=1 时，TC0 是一个外部事件计数器，不再响应 P0.0 的中断请求。（P0.0IRQ 始终为 0）。)**

8.2.3 TC0C 计数寄存器

TC0C 是一个 8 位定时计数器，只要 TC0ENB 置“1”就开启定时器。TC0C 是个加 1 计数器，时钟源频率由 TC0RATE0~TC0RATE2 决定。当 TC0C 计到 “0FFH” 后，若再加 1 就会回到“00H”，产生溢出信号，TC0 中断请求标志被置为 “1”，如果 TC0 中断又同时被使能（TC0IEN =1），那么系统将执行 TC0 的中断服务程序。

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 高速模式下，设 TC0 时间间隔为 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

表 8-1. TC0 的时间表

8.2.4 TC0R 自动装载寄存器

TC0R 为 8 位自动装载寄存器，还用于 TC0OUT 和 PWM0OUT 功能。在 TC0OUT 功能下，用户必须使能 TC0R 并且要进行设置。其主要功能如下：

- 存放自动装载值，当 TC0C 溢出时将其写入 TC0C 中（ALOAD1 = 1）；
- 存放 PWM0OUT 的占空比。

TC0R 初始值 = 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

定时/计数器 TC0 的工作流程如下：

- 置 TC0C 初始值，设置定时器中断间隔时间；
- TC0ENB 置为“1”，TC0 计数开始；
- 根据 TC0M 的选择的时钟源频率，每个时钟 TC0C 加 1；
- 如果 TC0 从“FFH”增至“00H”，TC0 溢出；
- 当 TC0 发生溢出，TC0IRQ 通过硬件设为“1”；
- 执行中断服务程序；
- 用户复位 TC0C，重新开始 TC0 定时器操作。

➲ **例：初始化 TC0M，TC0C 无自动加载功能。**

```
B0BCLR      FTC0IEN     ; 禁止 TC0 中断
B0BCLR      FTC0ENB     ; 停止 TC0 计数
MOV         A,#00H      ;
B0MOV       TC0M,A      ; 设置 TC0 的分频数为 fcpu /256
MOV         A,#74H      ; 设置 TC0C 初始值 = 74H
B0MOV       TC0C,A      ; 定时时间 10 ms

B0BCLR      FTC0IRQ     ; 清 TC0 中断请求标志
B0BSET      FTC0IEN     ; 定时时间 10 ms
B0BSET      FTC0ENB     ; 开始 TC0 计数
```

➲ **例：初始化 TC0M，TC0C 有自动加载功能。**

```
B0BCLR      FTC0IEN     ; 禁止 TC0 中断
B0BCLR      FTC0ENB     ; 停止 TC0 计数
MOV         A,#00H      ;
B0MOV       TC0M,A      ; 设置 TC0 分频数为 fcpu /256
MOV         A,#74H      ; 设置 TC0C 初始值 = 74H
B0MOV       TC0C,A      ; 定时时间 10 ms
B0MOV       TC0R,A      ; 设置重新装载时时间常数

B0BSET      FTC0IEN     ; 使能 TC0 中断
B0BCLR      FTC0IRQ     ; 清 TC0 中断请求标志
B0BSET      FTC0ENB     ; 开始 TC0 计数
B0BSET      ALOAD0      ; 使能 TC0R 自动重新装载功能
```

➲ 例：无自动加载功能的 **TC0** 中断服务程序

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

                B0XCH       A, ACCBUF       ; 使用 B0XCH 不会影响到 C,Z 等标志位
                B0MOV       A, PFLAG
                B0MOV       PFLAGBUF, A     ; 不错 PFLAG。

                B0BTS1      FTC0IRQ         ; 检查是否是 TC0IRQ 中断请求
                JMP         EXIT_INT        ; TC0IRQ = 0, 退出中断

                B0BCLR      FTC0IRQ         ; 清 TC0IRQ
                MOV         A,#74H          ; 重新设置 TC0C
                B0MOV       TC0C,A
                .           .               ; TC0 中断服务程序
                .           .
                JMP         EXIT_INT        ; 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
```

➲ 例：有自动加载功能的 **TC0** 中断服务程序

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

                B0XCH       A, ACCBUF       ; 使用 B0XCH 不会影响 C, Z 等标志
                B0MOV       A, PFLAG
                B0MOV       PFLAGBUF, A     ; 不错 PFLAG

                B0BTS1      FTC0IRQ         ; 检查是否是 TC0IRQ 中断请求
                JMP         EXIT_INT        ; TC0IRQ = 0, 退出中断

                B0BCLR      FTC0IRQ         ; 清 TC0IRQ
                .           .               ; TC0 中断服务程序
                .           .
                JMP         EXIT_INT        ; TC0 中断服务程序结束
                .           .
                .           .
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)

TC0 定时器/计数器提供了一个频率输出功能。通过设置 TC0 的时钟频率，可以从 P5.4 输出时钟信号，同时 P5.4 的基本输入/输出功能会自动屏蔽。TC0 的输出信号是 2 分频的。由于 TC0 时钟源可以有多种选择，相应就可产生多种频率输出，这个功能常用作 buzzer 输出。

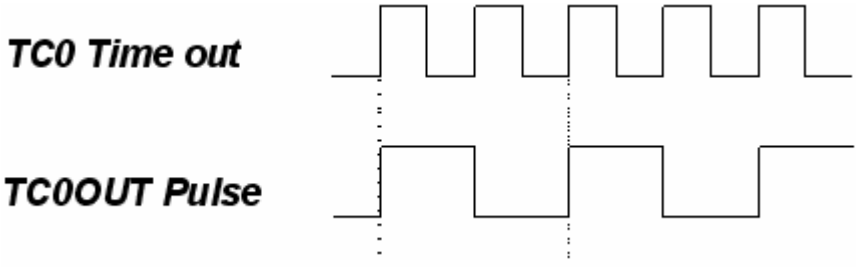


图 8-2. TC0OUT 脉冲频率

➲ **例：设置 TC0 的 TC0OUT (P5.4)输出，要求外部高速时钟 4MHz，TC0OUT 的频率 1KHz。因为 TC0OUT 经过了 2 分频 ，所以 TC0 的时钟设为 2KHz。TC0 时钟源来自外部振荡器，TC0 的速率为 Fcpu/4。TC0RATE2~TC0RATE1 = 110，TC0C = TC0R = 131。**

```
MOV     A,#01100000B
B0MOV   TC0M,A          ; 设置 TC0 的分频数为 Fcpu/4

MOV     A,#131          ; 设置自动重新装载的时间常数
B0MOV   TC0C,A
B0MOV   TC0R,A

B0BSET  FTC0OUT         ; 允许 TC0 频率输出（P5.4），同时禁止 P5.4 的 I/O 功能
B0BSET  FALOAD0         ; 允许自动装载功能
B0BSET  FTC0ENB         ; TC0 开始计数
```

8.3 定时/计数器(TC1)

8.3.1 概述

定时/计数器 TC1 用来产生定时中断请求。它是具有自动重新装载功能的定时计数器，主要由两部分组成：8 位自动装载寄存器 TC1R，用于保存计数参考值；8 位自动加 1 的计数器 TC1C。

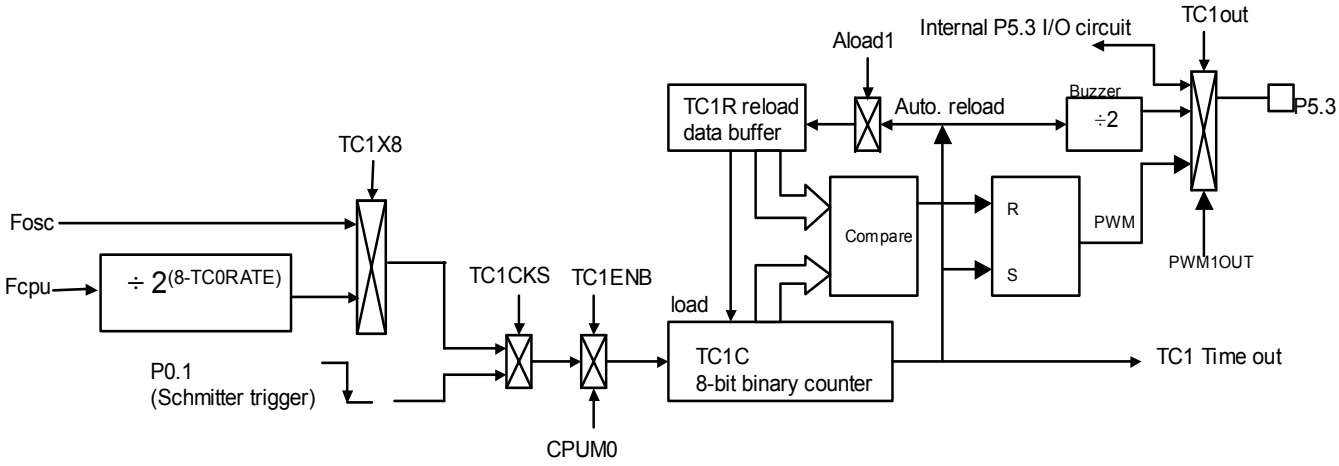


图 8-3. 定时/计数器 TC1 框图

TC1 的主要功能如下：

- **8 位可编程定时器：**根据设定的时钟频率，产生定时中断。
- **频率输出（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，通过 PWM1OUT 引脚 (P5.3)输出。

8.3.2 TC1M 模式寄存器

TC1M 为 8 位可读/写模式控制寄存器。通过载入不同值，用户可以在执行程序过程中动态的调整定时器的时钟频率。

通过设置 TC1 的 TC1RATE0~TC1RATE2，定时器 TC1 提供了 8 种可选择的时钟源频率。若 TC1X8 = 1，TC1 的时钟频率的初始值为 Fosc，TC1X8 = 0 时，TC1 的时钟频率范围为 Fcpu(Fosc)/2~Fcpu(Fosc)/256。TC1M 的第 7 位 TC1ENB 位是 TC1 的启动控制位。它们共同决定了 TC1 定时器的时钟源频率和定时间隔。

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 定时器速度的 8 倍。
0 = 禁止
1 = 使能

Bit2 **TC0X8**：TC0 定时器速度的 8 倍。
0 = 禁止
1 = 使能

TC1M 初始值 = 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

TC0Rate	TC1X8=0	TC0X8=1x8
111	Fcpu/256	Fcpu/256x8
110	Fcpu/128	Fcpu/128x8
101	Fcpu/64	Fcpu/64x8
100	Fcpu/32	Fcpu/32x8
011	Fcpu/16	Fcpu/16x8
010	Fcpu/8	Fcpu/8x8
001	Fcpu/4	Fcpu/4x8
000	Fcpu/2	Fcpu/2x8

Bit 3 **TC1CKS**：TC1 时钟源选择位
0 = 内部时钟源(fcpu)
1 = 来自 P0.1(INT1)的外部时钟源

Bit 2 **ALOAD1**：自动装载控制位
0 = 禁止自动装载功能
1 = 使能自动装载功能

Bit 1 **TC1OUT**：TC1 超时信号输出控制位，**仅当 PWM1OUT = 0 时有效。**
0 = 禁止 TC1OUT 的信号输出功能并使能 P5.3 的 I/O 功能
1 = 使能 TC1OUT 的信号输出功能并禁止 P5.3 的 I/O 功能

Bit 0 **PWM1OUT**：PWM 输出控制位。详见“PWM 功能说明”章节。
0 = 禁止 PWM 输出功能
1 = 使能 PWM 输出功能（自动禁止 TC1OUT 功能）

PWM1OUT = 1, TC1X8=0

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

PWM0OUT = 1, TC1X8=1

ALOAD0	TC0OUT	TC1 溢出边界	PWM 占空比范围	PWM 的最大频率(Fosc = 16M)	备注
0	0	FFh to 00h	0/255 ~ 255/255	62.5K	每计数 256 次溢出
0	1	3Fh to 40h	0/63 ~ 63/63	250K	每计数 64 次溢出
1	0	1Fh to 20h	0/31 ~ 31/31	500K	每计数 32 次溢出
1	1	0Fh to 10h	0/15 ~ 15/15	1000K	每计数 16 次溢出

➢ **注 1：当 TC1CKS=1 时，TC0 是一个外部事件计数器，不再响应 P0.1 的中断请求。（P0.1IRQ 时钟为 0）**

8.3.3 TC1C 计数寄存器

TC1C 是一个 8 位定时计数器，只要 TC1ENB 置“1”就开启定时器。TC1C 是个加 1 计数器，时钟源频率由 TC1RATE0~TC1RATE2 决定。当 TC1C 计到“0FFH”后，若再加 1 就会回到“00H”，产生溢出信号，TC1 中断请求标志被置为“1”，如果 TC1 中断又同时被使能（TC1IEN =1），那么系统将执行 TC1 的中断服务程序。

TC1C 初始值 = 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 初始值的计算方法如下：

TC1C 初始值 = 256 - (TC1 中断间隔时间 * 输入时钟)

➲ 例： **3.58MHZ 高速模式下，设 TC1 的时间间隔为 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)	
		最大溢出时间间隔	单步间隔时间 = max/256	最大溢出时间间隔	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)	
		最大溢出时间间隔	单步间隔时间 = max/256	最大溢出时间间隔	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)	
		最大溢出时间间隔	单步间隔时间 = max/256	最大溢出时间间隔	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)	
		最大溢出时间间隔	单步间隔时间 = max/256	最大溢出时间间隔	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

表 8-2. TC1 时间表

8.3.4 TC1R 自动装载寄存器

TC1R 为 8 位自动装载寄存器，还用于频率输出和 PWM 功能。在 TC1OUT 功能下，用户必须使能 TC1R 并且要进行设置。其主要功能如下：

- 存放自动装载值，当 TC1C 溢出时将其写入 TC1C 中（ALOAD1 = 1）；
- 存放 PWM1OUT 的占空比

TC1R 初始值 = 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 的计算，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 定时/计数器操作流程

定时/计数器 TC1 的工作流程如下：

- 置 TC1C 初始值，设置定时器中断间隔时间；
- TC1ENB 置为“1”，TC1 计数开始；
- 根据 TC1M 的中选择的时钟源频率，每个时钟 TC1C 加 1；
- 如果 TC1 从“FFH”增至“00H”，TC1 溢出；
- 当 TC1 发生溢出，TC1IRQ 通过硬件设为“1”；
- 执行中断服务程序；
- 用户复位 TC1C，重新开始 TC1 定时器操作。

➲ **例：初始化 TC1M 和 TC1C，无自动加载功能**

```
B0BCLR    FTC1IEN     ; 禁止 TC1 中断
B0BCLR    FTC1ENB     ; 停止 TC1 计数
MOV       A,#00H      ;
B0MOV     TC1M,A      ; 设置 TC0 的分频数为 fcpu / 256
MOV       A,#74H      ; 设置 TC1C 初始值 = 74H
B0MOV     TC1C,A      ; 定时时间 10 ms

B0BCLR    FTC1IRQ     ; 清 TC1 中断请求标志
B0BSET    FTC1IEN     ; 使能 TC1 中断
B0BSET    FTC1ENB     ; 开始 TC1 计数
```

➲ **例：初始化 TC1M 和 TC1C，有自动装载功能**

```
B0BCLR    FTC1IEN     ; 禁止 TC1 中断
B0BCLR    FTC1ENB     ; 停止 TC1 计数
MOV       A,#00H      ;
B0MOV     TC1M,A      ; 设置 TC0 的分频数为 fcpu / 256
MOV       A,#74H      ; 设置 TC1C 初始值 = 74H
B0MOV     TC1C,A      ; 定时时间 10 ms
B0MOV     TC1R,A      ; 设置重新装载时间常数

B0BSET    FTC1IEN     ; 使能 TC1 中断
B0BCLR    FTC1IRQ     ; 清 TC1 中断请求标志
B0BSET    FTC1ENB     ; 开始 TC1 计数
B0BSET    ALOAD1      ; 使能 TC1R 自动重新装载功能
```

➲ 例：无自动装载功能的 **TC1** 中断服务程序

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

                B0XCH       A, ACCBUF       ; 使用 B0XCH 不会影响到 C, Z 等标志位
                B0MOV       A, PFLAG
                B0MOV       PFLAGBUF, A     ; 保存 PFLAG 的值

                B0BTS1      FTC1IRQ         ; 检查是否是 TC1IRQ 中断请求
                JMP         EXIT_INT        ; TC1IRQ = 0, 退出中断

                B0BCLR      FTC1IRQ         ; 清 TC1IRQ
                MOV         A,#74H          ; 重新设置 TC1C
                B0MOV       TC1C,A
                .           .               ; TC1 中断服务程序
                .           .
                JMP         EXIT_INT        ; TC1 中断服务程序结束
                .           .
EXIT_INT:
                B0MOV       A, PFLAGBUF     ; 恢复 PFLAG 的值
                B0MOV       PFLAG, A
                B0XCH       A, ACCBUF       ; 恢复 ACC

                RETI                        ; 中断返回
```

➲ 例：有自动装载功能的 **TC1** 中断服务程序。

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

                B0XCH       A, ACCBUF       ; 使用 B0XCH 不会影响到 C, Z 等标志位
                B0MOV       A, PFLAG
                B0MOV       PFLAGBUF, A     ; 保存 PFLAG 的值

                B0BTS1      FTC1IRQ         ; 检查是否是 TC1IRQ 中断请求
                JMP         EXIT_INT        ; TC1IRQ = 0, 退出中断

                B0BCLR      FTC1IRQ         ; 清 TC1IRQ
                .           .               ; TC1 中断服务程序
                .           .
                JMP         EXIT_INT        ; TC1 中断服务程序结束
                .           .
                .           .
EXIT_INT:
                B0MOV       A, PFLAGBUF
                B0MOV       PFLAG, A        ; 恢复 PFLAG 的值
                B0XCH       A, ACCBUF       ; 恢复 ACC

                RETI                        ; 中断返回
```

8.3.6 TC1 时钟频率输出(BUZZER 输出)

TC1 定时器/计数器提供了一个频率输出功能。通过设置 TC1 的时钟频率，可以从 P5.3 输出时钟信号，同时 P5.3 的基本输入/输出功会自动屏蔽。TC1 的输出信号是 2 分频的。由于 TC1 时钟源可以有多种选择，相应就可产生多种频率输出，这个功能常用作蜂鸣器输出。

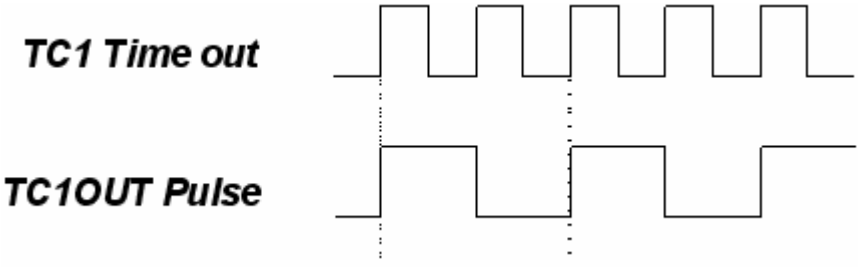


图 8-4. TC1OUT 脉冲频率

➲ **例：设置 TC1 的 TC1OUT (P5.3)输出，要求外部高速时钟 4MHz，TC1OUT 的频率 1KHz。因为 TC1OUT 经过 2 分频 ，因此 TC1 的时钟设为 2KHz。TC1 时钟源来自外部振荡器，T1C 的速率为 Fcpu/4。所以 TC1RATE2~TC1RATE1 = 110。 TC1C = TC1R = 131**

```
MOV     A,#01100000B
B0MOV   TC1M,A          ; 设置 TC1 的分频数为 Fcpu/4

MOV     A,#131          ; 设置自动重新装载的时间常数
B0MOV   TC1C,A
B0MOV   TC1R,A

B0BSET  FTC1OUT         ; 允许 TC1 频率输出（P5.3），同时禁止 P5.3 的 I/O 功能
B0BSET  FALOAD1         ; 允许自动装载功能
B0BSET  FTC1ENB         ; TC1 开始计数
```

➢ **注：TC1OUT 的频率表类似于 TC0OUT 的频率表，请参考 TC0OUT 频率表（表 7-2～7-5）。**

8.4 PWM 功能说明

8.4.1 概述

PWM 功能使用的时基为 TC0 或 TC1，产生的 PWM 信号通过 PWM0OUT 引脚（P5.4）或 PWM1OUT（P5.3）输出。8 位定时计数器的计数范围为 0~255。8 位计数器的值与 TC0R/TC1R 的值相比较。当 TC0R 和 TC0C 的值相等时，PWM 输出低电平；当 TC0C 的值重新回到 0 时，PWM 输出高电平。

PWM 的初始输出值是低电平直到计数器的值溢出（如 TC0C 由 FFH 到 00H），PWM 的输出变为高电平。脉冲的宽度比例（占空比）是由 TC0R/TC1R 寄存器控制的，溢出边界则是由 ALOAD0/ALOAD1 和 TC0OUT/TC1OUT 位控制的。向参考寄存器 TC0R/TC1R 中写入 00H 可以使 PWM 输出保持在低电平；而连续写入 FFH 到 TC0R 中则可以使 PWM 输出保持在高电平，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	PWM 的最大频率 (Fcpu = 4M)	备注
0	0	FFh to 00h	0/255 ~ 255/255	7.8125K	每计数 256 次溢出
0	1	3Fh to 40h	0/63 ~ 63/63	31.25K	每计数 64 次溢出
1	0	1Fh to 20h	0/31 ~ 31/31	62.5K	每计数 32 次溢出
1	1	0Fh to 10h	0/15 ~ 15/15	125K	每计数 16 次溢出

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

表 8-2. PWM 占空比表

注：若使能 PWM0OUT 或 TC0OUT 功能，P5.4 自动地变为输出模式；
若禁止 PWM0OUT 或 TC0OUT 功能，则 P5.4 的模式由 P54M 为定义。

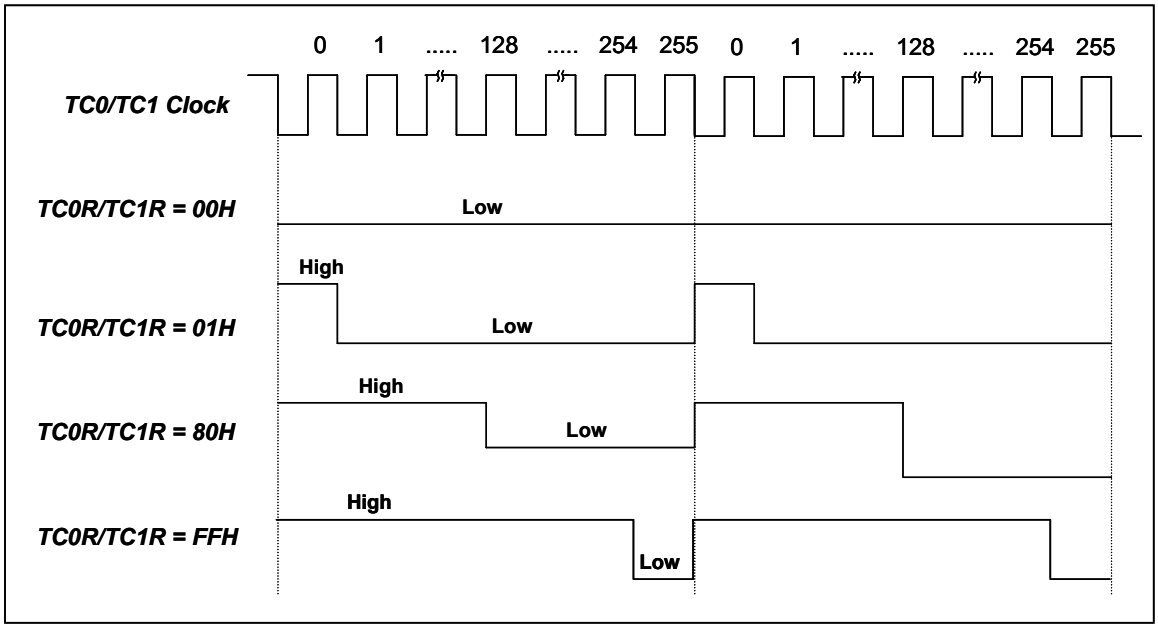


图 8-5 不同 TC0R/TC1R 值的 PWM 输出占空比

8.4.2 PWM 操作举例

➲ **例：设置 PWM0 的输出 PWM0OUT (P5.4)，其外部高速振荡器时钟 4MHz，PWM 输出占空比为 30/256。PWM 的输出频率是 1KHz。PWM 的时钟源来自外部振荡器，TC0 的速率是 Fcpu/4。TC0RATE2~TC0RATE1 =110。TC0C = TC0R = 30**

```
MOV     A,#01100000B
B0MOV   TC0M,A      ; 设置 TC0 的分频数为 Fcpu/4
MOV     A,#0x00     ; 初始化 TC0

MOV     A,#30       ; 设置 PWM 的占空比为 30/256
B0MOV   TC0R,A

B0BCLR  FTC0OUT     ; 禁止 TC0OUT 功能
B0BSET  FPWM0OUT    ; 使能 PWM0 输出到 P5.4 并禁止 P5.4 的输入输出功能
B0BSET  FTC0ENB     ; TC0 开始计数
```

➢ **注 1：TC0R 和 TC1R 是只写寄存器，不能执行 INCMS、DECMS 指令。**
➢ **注 2：初始化 TC0C，确保第一个占空比正确。使能 TC0 后，不能调整 TC0R 的值以避免 PWM 输出的占空比出错。**

➲ **例：调整 TC0R/TC1R 的值**

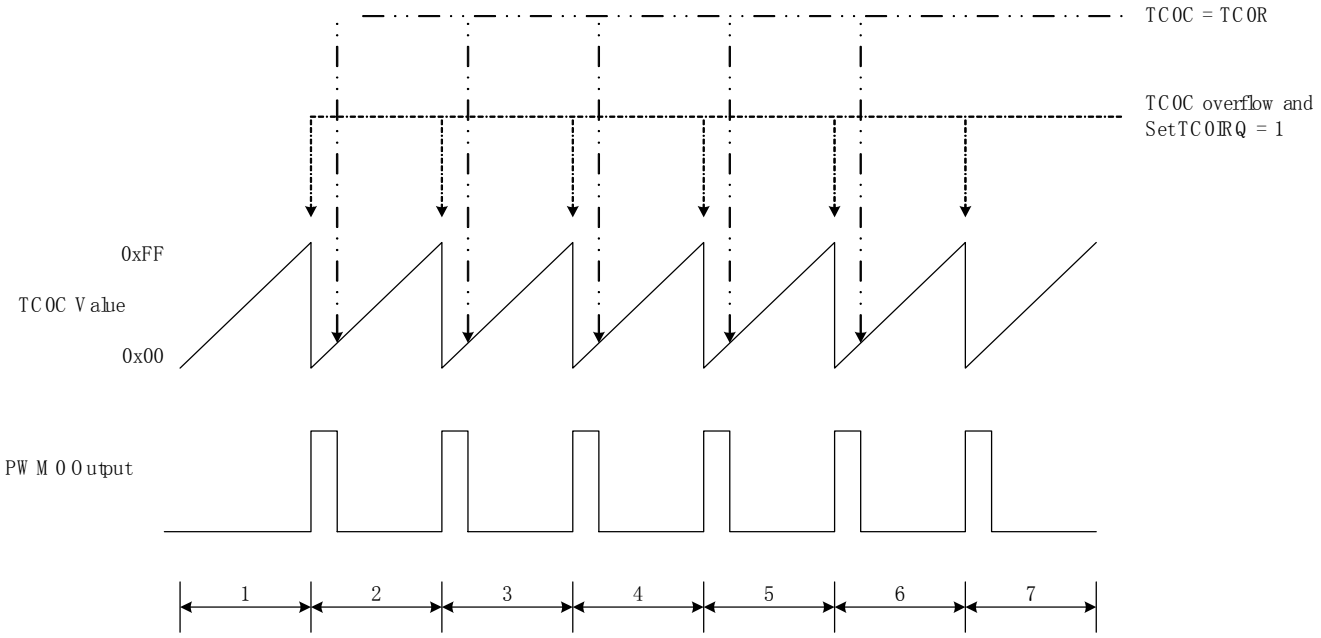
```
MOV     A, #30H     ; 赋予一个立即数
B0MOV   TC0R, A

INCMS   BUF0        ; 从 BUF0 得到一个新的 TC0R 值
B0MOV   A, BUF0
B0MOV   TC0R, A
```

➢ **注 3：PWM 占空比改变时，同时改变 TC0C/TC1C 和 TC0R/TC1R 的值，以避免 PWM 信号受到干扰；**
➢ **注 4： 使能 PWM0/ PWM1 输出功能时，TC0OUT/TC1OUT 必须置为“0”；**
➢ **注 5： PWM 功能和中断可同时使用。**

8.4.3 TCxR 改变时 PWM0 的占空比

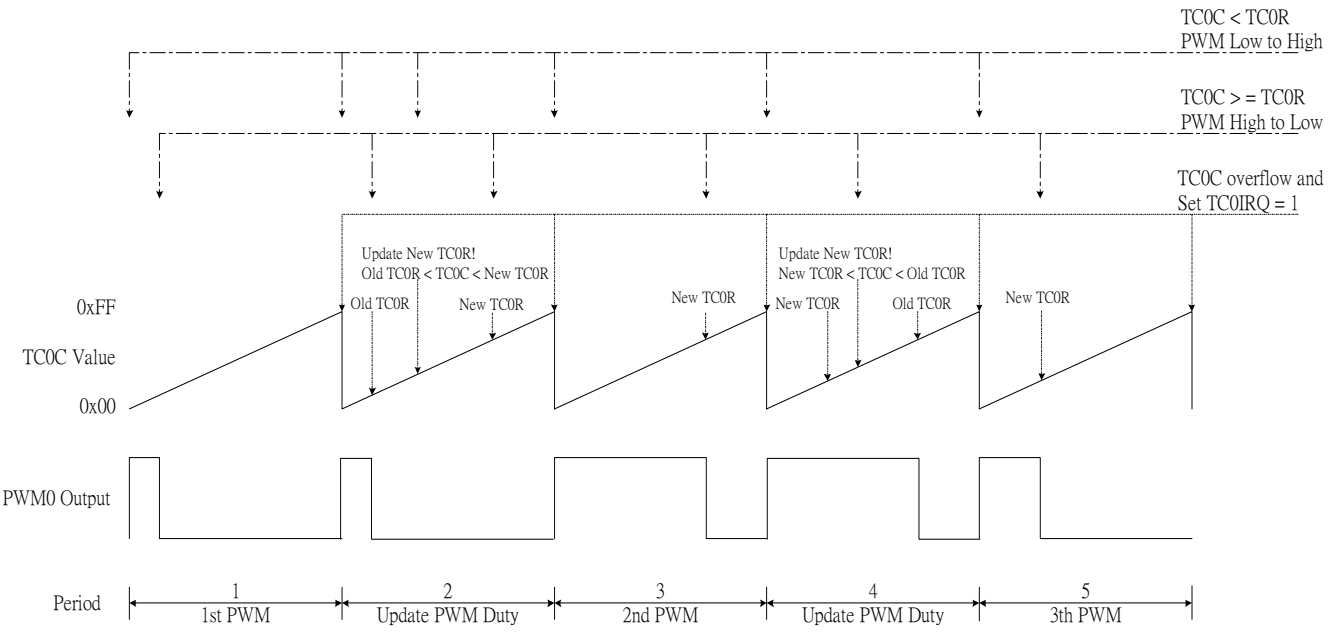
在 PWM 模式下，系统一直在比较 TCxC 和 TCxR 的值：当 TCxC<TCxR 时，PWM 输出逻辑“高”电平；当 TCxC≧TCxR 时，PWM 输出逻辑“低”电平。若 TCxC 在某个特定的时间改变，PWM 的占空比则就在下个 PWM 段内改变。

若 TCxR 一直固定为某个值，PWM 的波形也就始终是一样的。



上图所示为 TCxR 值固定时，PWM 的波形图。在每个 TCxC 溢出时，PWM 输出高电平；当 TCxC≧TCxR 时，PWM 输出低电平。

当 TCxR 在执行程序的过程中改变时，PWM 的波形图如下所示：



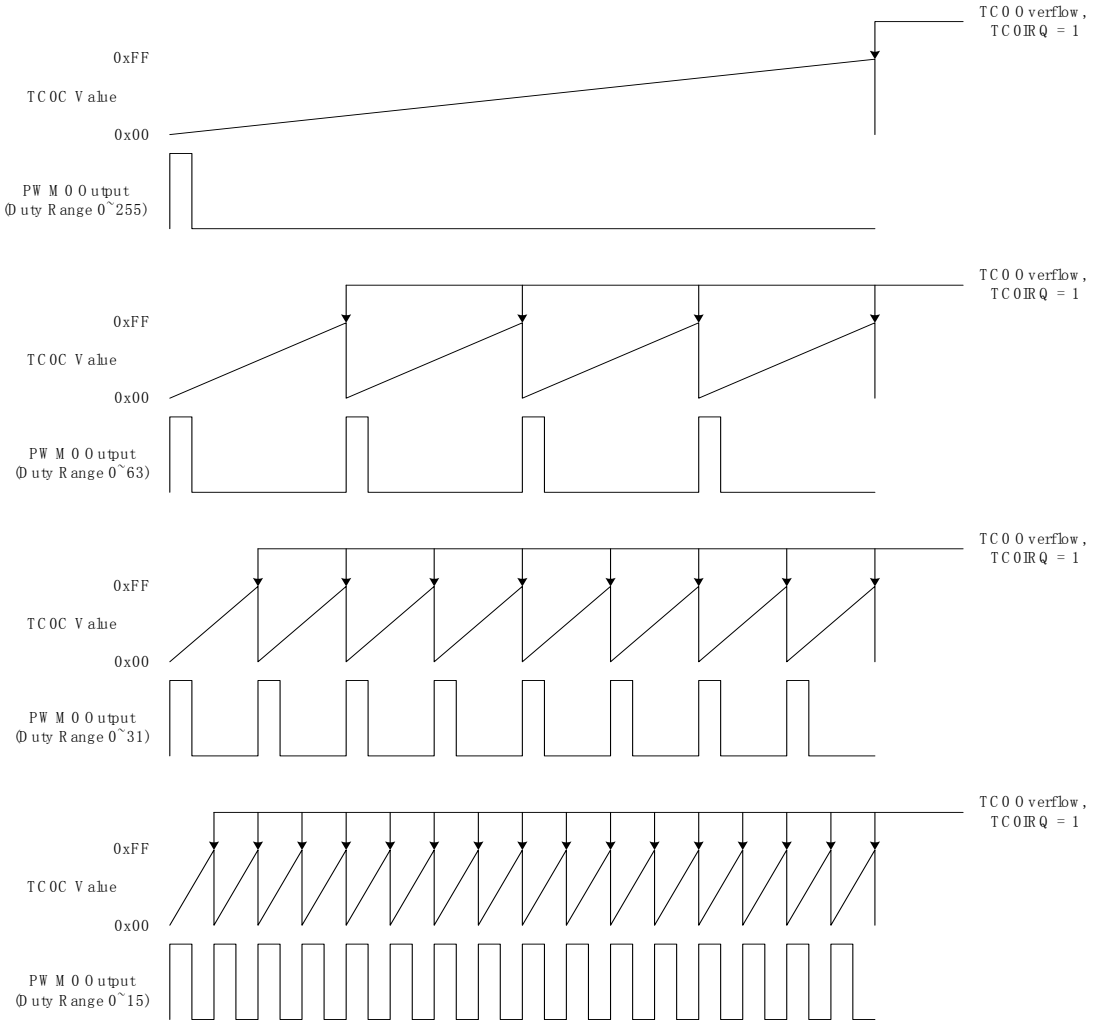
在第二段和第四段中，显示新的占空比（TCxR），但是，PWM 在第二段和第四段中的占空比仍是一样的，它们将会在下一段中改变。这时，系统可以 PWM 作出改变或在同一个周期内 H/L 改变两次，这样系统可以预防未知的系统错误。

8.4.4 TCxIRQ 和 PWM 占空比

在 PWM 模式下，TCxIRQ 的频率决定于 PWM 的占空比范围。

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

下图显示了 TCxIRQ 的频率和 PWM 占空比的关系：



9 中断

9.1 概述

SN8P2710 有 4 个中断源：2 个内部中断源（TC0、TC1），2 外部中断源（INT0、INT1）。外部中断能够唤醒睡眠模式进入高速模式，它们的信号输入引脚 INT0/INT1 与 P0.0/P0.1 引脚共享。当系统进入到中断服务程序时，全局中断控制位 GIE 清零，系统退出中断服务后，GIE 被置为 “1” 以准备响应下一个中断请求。所有的中断请求存放于寄存器 INTRQ 中，用户可通过程序设置中断优先级。

- **注：GIE 使能后才能响应各中断。**

9.2 INTEN 中断使能寄存器

INTEN 为中断使能控制寄存器，包括 2 个内部中断和 2 个外部中断。INTEN 的某位被置为“1”，则相对应的中断请求便能够被响应。一旦有中断发生，程序将跳至 ORG 8 处执行中断服务程序。当执行到中断服务返回指令（RETI）时，将退出中断程序。

INTEN 初始值 = 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**：定时/计数器 TC1 中断控制位
0 = 禁止
1 = 使能

Bit 5 **TC0IEN**：定时/计数器 TC0 中断控制位
0 = 禁止
1 = 使能

Bit 1 **P01IEN**：外部中断 P0.1 控制位
0 = 禁止
1 = 使能

Bit 0 **P00IEN**：外部中断 P0.0 控制位
0 = 禁止
1 = 使能

9.3 INTRQ 中断请求寄存器

INTRQ 为中断请求寄存器，包含了所有的中断请求标志，当有中断发生时，INTRQ 寄存器中的相应位会置为“1”。中断请求标志需要用软件清零。用户通过检查中断请求寄存器可以知道中断的种类，从而执行相应的中断服务程序。

INTRQ 初始值 = 0000 0000

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 定时/计数器中断请求控制位
0 = 无中断请求
1 = 请求中断服务

Bit 5 **TC0IRQ**：TC0 定时/计数器中断请求控制位
0 = 无中断请求
1 = 请求中断服务

Bit 1 **P01IRQ**：外部中断 P0.1 中断请求控制位
0 = 无中断请求
1 = 请求中断服务

Bit 0 **P00IRQ**：外部中断 P0.0 中断请求控制位
0 = non-request
1 = request

9.4 P0.0 中断触发边沿控制寄存器

PEDGE 初始值 = 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 中断触发边沿控制寄存器
00 = 保留
01 = 上升沿
10 = 下降沿**(复位后为默认设置)**
11 = 下降沿和上升沿均有效（变换电平触发）

9.5 中断操作举例

SN8P2710 共有 2 个中断源，各中断的详细操作如下：

9.5.1 GIE 总中断操作

GIE 是总中断控制位。所有的中断在 GIE 使能的前提下才能够得到响应。一旦有中断请求发生，程序计数器 PC 指向中断向量地址（ORG 8），堆栈层数加 1。

STKP 初始值 = 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**：总中断控制位
0 = 禁止
1 = 使能

➲ **例： 设置总中断控制位(GIE)**

```
B0BSET      FGIE        ; 总中断使能
```

➢ **注：GIE 必须使能，所有中断才能被响应。**

9.5.2 INT0 (P0.0)中断操作

当 INT0 中断发生时，不管 P00IEN 是否使能，P00IRQ 都会置“1”。若 P00IEN=1，且 P00IRQ=1，那么系统就进入中断向量地址（ORG 8）执行中断服务程序。但若 P00IEN=0，不管 P00IRQ 是否等于 1，系统都不进入中断。用户应注意多种中断下的处理。

➲ **例：初始化 INT0**

```
B0BSET      FP00IEN     ; INT0 中断使能
B0BCLR      FP00IRQ     ; 清 INT0 中断请求标明志
B0BSET      FGIE        ; 总中断使能
```

➲ **例：INT0 中断服务程序**

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

            B0XCH       A, ACCBUF   ; B0XCH 不影响标志位 C，Z
            B0MOV       A, PFLAG
            B0MOV       PFLAGBUF, A ; 保存 PFLAG 的值

            B0BTS1      FP00IRQ     ; 判断是否有外部中断请求
            JMP         EXIT_INT    ; P00IRQ = 0，退出中断服务程序

            B0BCLR      FP00IRQ     ; 清中断标志
            .           .           ; INT0 中断服务程序
            .           .
EXIT_INT:
            B0MOV       A, PFLAGBUF
            B0MOV       PFLAG, A    ; 恢复 PFLAG 的值
```

9.5.3 INT1 (P0.1)中断操作

当 INT1 中断发生时，不管 P01IEN 是否使能，P01IRQ 都会置“1”。若 P01IEN=1，且 P01IRQ=1，那么系统就进入中断向量地址（ORG 8）执行中断服务程序。但若 P01IEN=0，不管 P01IRQ 是否等于 1，系统都不进入中断。用户应注意多种中断下的处理。

➲ **例：初始化 INT1**

```
        B0BSET      FP01IEN             ; INT1 中断使能
        B0BCLR      FP01IRQ             ; 清 INT1 中断请求
        B0BSET      FGIE                ; 总中断使能
```

➲ **例：INT1 中断服务。**

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

        B0XCH       A, ACCBUF           ; B0XCH 不影响标志位 DC、C、Z
        B0MOV       A, PFLAG
        B0MOV       PFLAGBUF, A         ; 保存 PFLAG 的值

        B0BTS1      FP01IRQ             ; 判断是否有外部中断请求
        JMP         EXIT_INT            ;

        B0BCLR      FP01IRQ             ; 清中断标志
        .           .                   ; INT1 中断服务程序
        .           .
EXIT_INT:
        B0MOV       A, PFLAGBUF
        B0MOV       PFLAG, A            ; 恢复 PFLAG 的值
        B0XCH       A, ACCBUF           ; 恢复 ACC

        RETI                            ; 中断返回
```

9.5.4 TC0 中断操作

当计数器 TC0C 溢出时，无论 TC0IEN 是否使能，TC0IRQ 都会置“1”。若 TC0IEN =1，且 TC0IRQ =1，那么系统就进入中断向量地址（ORG 8）执行中断服务程序。若 TC0IEN =0，不管 TC0IRQ 是否等于 1，系统都不进入中断。用户应注意多种中断下的处理。

➲ **例：初始化 TC0**

```
        B0BCLR      FTC0IEN         ; 禁止 TC0 中断
        B0BCLR      FTC0ENB         ; 停止 TC0 计数
        MOV         A, #20H         ;
        B0MOV       TC0M, A         ; 设置 TC0 定时模式 Fcpu / 64
        MOV         A, #74H         ; 设置 TC0 的初始值 = 74H
        B0MOV       TC0C, A         ; 定时中断为 10 ms

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

        B0BSET      FGIE            ; 使能总中断
```

➲ **例：TC0 中断服务**

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

        B0XCH       A, ACCBUF       ; 用 B0XCH 保存现场不会影响状态寄存器的值
        B0MOV       A, PFLAG
        B0MOV       PFLAGBUF, A     ; 保存 PFLAG 的值

        B0BTS1      FTC0IRQ         ; 检查是否是 TC0 中断请求
        JMP         EXIT_INT

        B0BCLR      FTC0IRQ         ; 清 TC0 中断标志
        MOV         A, #74H
        B0MOV       TC0C, A         ; 重新装载时间常数
        .           .               ; TC0 中断服务程序
EXIT_INT:
        B0MOV       A, PFLAGBUF
        B0MOV       PFLAG, A        ; 恢复 PFLAG 的值
        B0XCH       A, ACCBUF       ; 恢复 ACC 的值

        RETI                        ; 中断返回
```

9.5.5 TC1 中断操作

当计数器 TC1C 溢出时，无论 TC1IEN 是否使能，TC1IRQ 都会置“1”。若 TC1IEN =1，且 TC1IRQ =1，那么系统就进入中断向量地址（ORG 8）执行中断服务程序。若 TC1IEN =0，不管 TC1IRQ 是否等于 1，系统都不进入中断。用户应注意多种中断下的处理。

➲ **例：初始化 TC1**

```
        B0BCLR      FTC1IEN         ; 禁止 TC1 中断
        B0BCLR      FTC1ENB         ; 停止 TC1 计数
        MOV         A, #20H         ;
        B0MOV       TC1M, A         ; 设置 TC1 定时模式 Fcpu / 64
        MOV         A, #74H         ; 设置 TC1 的初始值 = 74H
        B0MOV       TC1C, A         ; 定时中断为 10 ms

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

        B0BSET      FGIE            ; 使能总中断
```

➲ **例：TC1 中断服务**

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

        B0XCH       A, ACCBUF       ; 用 B0XCH 保存现场不会影响状态寄存器的值
        B0MOV       A, PFLAG
        B0MOV       PFLAGBUF, A     ; 保存 PFLAG 的值

        B0BTS1      FTC1IRQ         ; 检查是否是 TC1 中断请求
        JMP         EXIT_INT        ;

        B0BCLR      FTC1IRQ         ; 清 TC1 中断标志
        MOV         A, #74H
        B0MOV       TC1C, A         ; 重新装载时间常数
        .           .               ; TC1 中断服务程序
EXIT_INT:
        B0MOV       A, PFLAGBUF
        B0MOV       PFLAG, A        ; 恢复 PFLAG 的值
        B0XCH       A, ACCBUF       ; 恢复 ACC 的值

        RETI                        ; 中断返回
```

9.5.6 多个中断操作

大部分情况下，用户需要同时处理多个中断。处理多个中断就需要设置中断的优先权。中断请求由不同的事件控制，但是，有中断请求并不意味着系统就会去执行中断服务程序。不管中断是否使能，都可触发中断请求，一旦有中断发生，相应的中断请求标志就会被置为“1”。各中断与对应的触发事件关系如下表所示：

中断	触发信号
P00IRQ	P0.0：下降沿触发
P01IRQ	P0.1：下降沿触发
TC0IRQ	TC0C 溢出
TC1IRQ	TC1C 溢出

在处理多中断请求下，用户必须对各中断进行优先权的设置，并根据 IEN 和 IRQ 的状态决定系统是否响应中断请求。用户必须在中断向量里检查中断控制位和中断请求标志位。

➲ **例：在多中断情况下，检查是否响应各中断请求**

```
                ORG         8                   ; 中断向量地址
                NOP
                B0XCH       A, ACCBUF           ; 用 B0XCH 保存现场不会影响状态寄存器的值
                B0MOV       A, PFLAG
                B0MOV       PFLAGBUF, A         ; 保存 PFLAG 的值
INTP00CHK:                                      ; 检查是否有 INT0 中断
                B0BTS1      FP00IEN             ; 检查是否允许外部中断 0
                JMP         INTP01CHK           ; 跳转到下一个中断
                B0BTS0      FP00IRQ             ; 检查是否有外部中断 0 的请求
                JMP         INTP00              ; 跳转到 INT0 的中断服务程序
INTP01CHK:                                      ; 检查是否有 INT1 中断
                B0BTS1      FP01IEN             ; 检查是否允许 INT1 中断
                JMP         INTP02CHK           ; 跳转到下一个中断
                B0BTS0      FP01IRQ             ; 检测是否有 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
                B0BTS1      FTC0IEN             ; 检查是否允许 TC0 中断
                JMP         INTTC1CHK           ; 跳转到下一个中断
                B0BTS0      FTC0IRQ             ; 检测是否有 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
                B0BTS1      FTC1IEN             ; 检查是否允许 TC1 中断
                JMP         INTSIOCHK           ; 跳转到下一个中断
                B0BTS0      FTC1IRQ             ; 检测是否有 TC1 中断请求
                JMP         INTTC1              ; 跳转到 TC1 中断服务程序
INT_EXIT:
                POP                             ; 恢复工作寄存器的内容
                B0XCH       A, ACCBUF           ; 恢复 ACC 的值
                RETI                            ; 中断返回
```

10 I/O 端口

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、P5)。输入/输出的方向由 PnM 寄存器控制，用户可以用 PnUR 寄存器设置上拉电阻。系统复位后，所有端口都默认为无上拉电阻的输入模式。

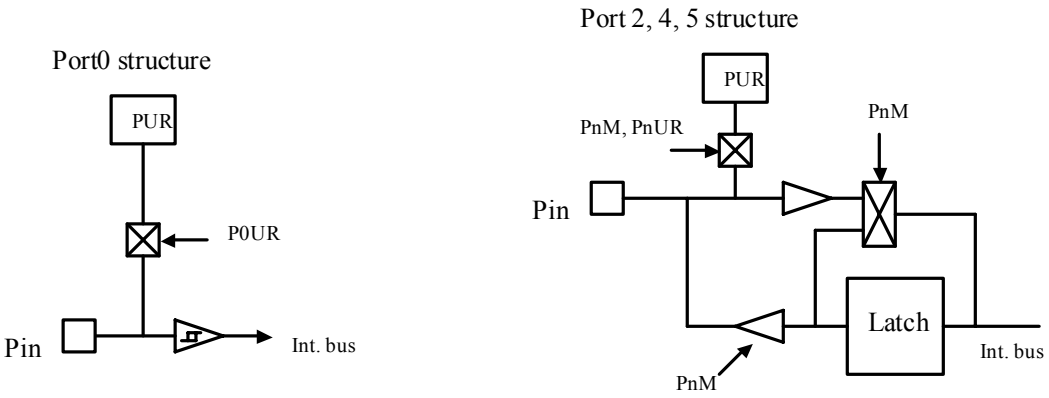


图 10-1. I/O 端口框图

10.2 I/O 端口功能说明

Port/Pin	I/O	功能说明	备注
P0.0~P0.1	I	基本输入功能	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	
P0.2	I	基本输入功能	
P0.3	I	基本输入功能	
		和复位引脚共享	
P2.0~P2.7	I/O	基本输入/输出功能	
P4.0~P4.7	I/O	基本输入/输出功能	
		ADC 模拟信号输入	
P5.0~P5.6	I/O	基本输入/输出引脚	

表 10-1 I/O 功能表

10.3 上拉电阻寄存器

上拉电阻的典型值是：200K@3V；100K@5V.

➢ ***Port0***

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

➢ ***Port2***

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

➢ ***Port4***

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

➢ ***Port5***

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 口上拉电阻**

```
CLR     P0UR        ; 禁止 P0 的上拉电阻

MOV     A, #01H     ;
B0MOV   P0UR, A     ; 使能 P0.0 的上拉电阻
```

10.4 I/O 端口模式

寄存器 PnM 控制端口的输入输出方向， P0 是单向输入模式，P2，P4，P5 可以选择作为输入或输出。

P2M 初始值 = 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] **P[27:20]M**：P2.0~P2.7 I/O 方向控制位
0 = 输入模式
1 = 输出模式

P4M 初始值 = 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] **P[47:40]M**：P4.0~P4.7 I/O 方向控制位
0 = 输入模式
1 = 输出模式

P5M 初始值 = 0000 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 [7:0] **P[56:50]M**：P5.0~P5.6 I/O 方向控制位
0 = 输入模式
1 = 输出模式

➲ **例：I/O 模式选择**

```
CLR     P2M
CLR     P4M
CLR     P5M

MOV     A, #0FFH        ; 设置所有的端口都为输出模式
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 去输出模式
```

10.5 I/O 端口数据寄存器

P0 初始值 = 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 初始值 = 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 初始值 = 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 初始值 = 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 P0 口的数据
B0MOV   A, P2     ; 读 P2 口的数据
B0MOV   A, P4     ; 读 P4 口的数据
B0MOV   A, P5     ; 读 P5 口的数据
```

➲ **例：写入数据到输出端口**

```
MOV     A, #55H   ; 写 55H 到 P2, P4, P5
B0MOV   P2, A
B0MOV   P4, A
B0MOV   P5, A
```

➲ **例：写入 1 位数据到输出端口。**

```
B0BSET  P2.3      ; 设置 P2.3 和 P4.0 为“1”
B0BSET  P4.0

B0BCLR  P2.3      ; 设置 P2.3 和 P5.5 为“0”
B0BCLR  P5.5
```

➲ **例：位检测**

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

11 8 通道 AD 转换

11.1 概述

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

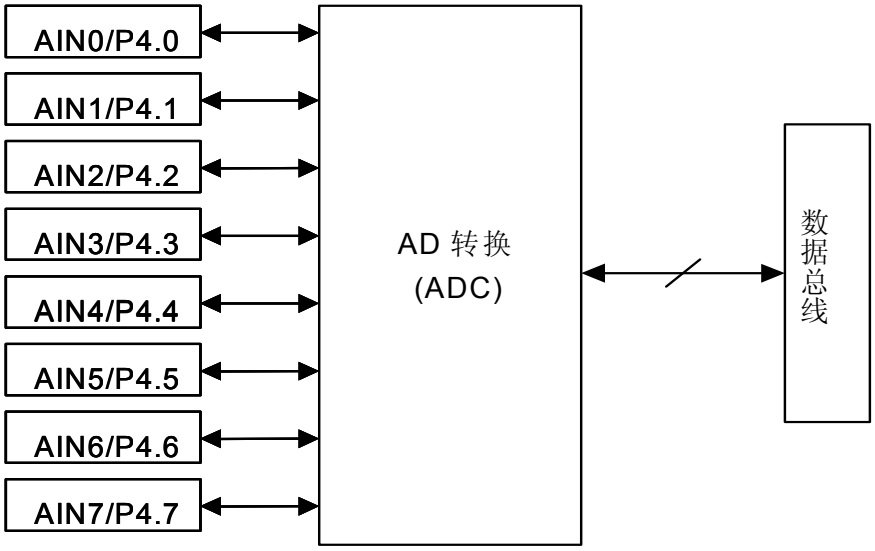


图 11-1. AD 转换功能图

- **注：分辨率为 12 位时，转换时间为 16 个 ADC 输入时钟。**
- **注：模拟输入电压必须在 AVREFH 和 AVREFL 之间。**
- **注：AVREFH 的值必须在 AVDD 和 AVREFL+2.0V 之间。**
- **注：ADC 设计时应注意：**
  - **设 ADC 的输入引脚为输入模式。**
  - **禁止 ADC 输入引脚的上拉电阻。**
  - **在进入睡眠模式省电前禁止 ADC。**
  - **在省电模式下设置 P4CON 的有关位以避免额外功耗。**
  - **使能 ADC(ADENB = 1)后延迟 100us 等待 ADC 电路准备好转换。**
  - **在进入睡眠模式前禁止 ADC(设置 ADENB = 0)以省电。**

11.2 ADM 寄存器

ADM 初始值 = 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 = 启动

Bit 5 **EOC**：ADC 状态位
0 = 转换过程中
1 = 转换结束，ADS 位清零

Bit 4 **GCHS**：输入通道控制位
0 = 禁止所有的 AIN 通道
1 = 使能所有的 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 初始值 = 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 时钟源选择位

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

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

11.4 ADB 寄存器

ADB 初始值 = 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

ADB 为 8 位数据缓冲器，用来保存 ADC 转换结果。ADB 只包含 ADC 转换结果中的高 8 位，把 ADB 寄存器和 ADR 的低半字节结合在一起，可得到一个 12 位的转换结果。ADB 为只读寄存器，在 8 位 ADC 模式下，ADC 转换结果保存在寄存器 ADB 中；在 12 位模式下，则分别保存在寄存器 ADB 和 ADR 中。

➢ **注：在上电时，ADB [0:11]的值是未知的。**

11.5 P4CON 寄存器

ADB 初始值 = xxxx 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

P4CON 是 P4 的配置寄存器，通过设置该寄存器可减小在 AD 转换或睡眠模式下的漏电流。P4CON[7:0]置 1 时可断开外部信号。

例如：若 AIN0 (P4.0)和 AIN1(P4.1)设置为 ADC 信道，当选择的通道是 AIN0 时，若 P4CON1 置“1”，可以避免通过 AIN1 通道的漏电流。同理：当转换通道是 AIN1 时，P4CON0 也要置“1”。在进入睡眠模式后，P4CON0 和 P4CON1 都必须置“1”，以确保无来自这两个通道的漏电流。

Bit[7:0] **P4CON**：P4 配置寄存器

P4CON7	0	允许 AIN7 (P4.7)的信号通过
	1	将 AIN7(P4.7)的信号隔离
P4CON6	0	允许 AIN7 (P4.7)的信号通过
	1	将 AIN7(P4.7)的信号隔离
P4CON5	0	允许 AIN7 (P4.7)的信号通过
	1	将 AIN6(P4.6)的信号隔离
P4CON4	0	允许 AIN6 (P4.6)的信号通过
	1	将 AIN5(P4.5)的信号隔离
P4CON3	0	允许 AIN4 (P4.4)的信号通过
	1	将 AIN3(P4.3)的信号隔离
P4CON2	0	允许 AIN2 (P4.2)的信号通过
	1	将 AIN2(P4.2)的信号隔离
P4CON1	0	允许 AIN1 (P4.1)的信号通过
	1	将 AIN1(P4.1)的信号隔离
P4CON0	0	允许 AIN0 (P4.0)的信号通过
	1	将 AIN0(P4.0)的信号隔离

➢ **注：当 P4[7:0]为基本 I/O 端口而不是 ADC 信道时，P4CON [7:0]必须置“0”，否则 P4 的信号会被隔离。**

AIN 的输入电压与 ADB 的输出数据的关系

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

针对不同的应用，用户可能需要一个 8 位到 12 位之间的分辨率。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 ADC 转换，然后在转换结果中去掉最低的几位得到需要的结果。如下表所示：

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

11.6 ADC 转换时间

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

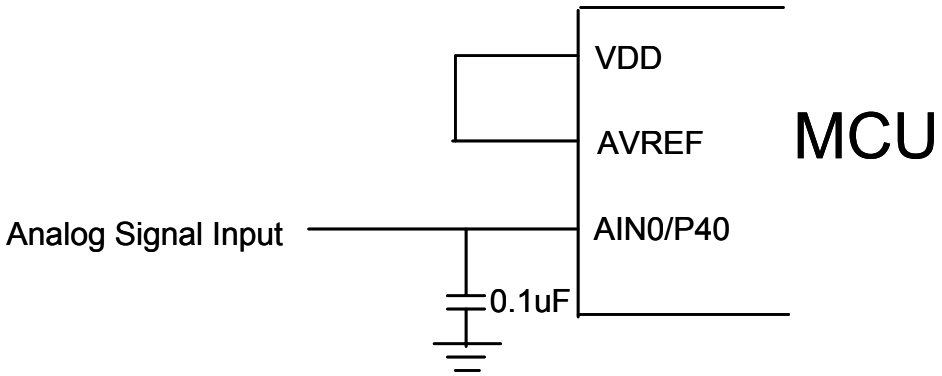
高速时钟(fosc)为@4MHz

ADCKS1	ADCKS0	ADC 时钟	ADC 转换时间
0	0	Fcpu/16	1/(4MHz/16)*16 = 64 us
0	1	Fcpu/8	1/(4MHz/8)*16 = 32 us
1	0	Fcpu	1/(4MHz)*16 = 4 us
1	1	Fcpu/2	1/(4MHz/2)*16 = 8 us

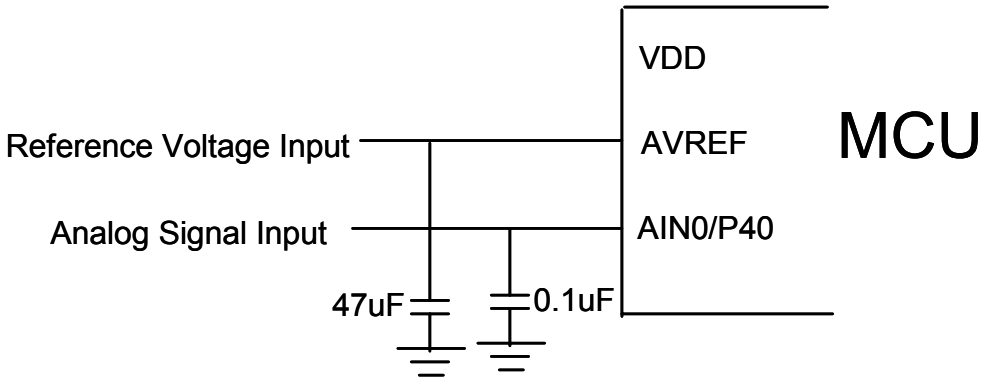
➲ 例：选取 AIN0 最为 12 位 ADC 的输入通道，AD 转换后进入省电模式。

```
ADC0:
        B0BSET      FADENB          ; 使能 ADC 电路
        CALL        Delay100uS      ; 延迟 100uS 等待 ADC 电路准备好以进行 ADC 转换
        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 为纯模拟输入引脚
        MOV         A, #40H
        B0MOV       ADR, A          ; 设置 ADC 为 12 位，ADC 时钟源为 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            ; 判断是否转换结束
        JMP         WADC0           ; 未结束则跳转到 WADC0
        B0MOV       A,ADB           ; 得到 AIN0 的输入数据（bit11 ~ bit4）
        B0MOV       Adc_Buf_Hi, A
        B0MOV       A,ADR           ; 得到 AIN0 的输入数据(bit3 ~ bit0)
        AND         A, 0Fh
        B0MOV       Adc_Buf_Low, A
Power_Down  .       .
        B0BCLR      FADENB          ; 释放 AD 通道
        B0BSET      FCPUM0          ; 进入睡眠模式
```

11.7 ADC 电路



AVREFH 和 VDD 连接.



AVREFH 和外部的 AD 参考电压连接.

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 DA 转换

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 位被置为 “1”时，数模转换电路使能，DAM 寄存器中的第 0 位到第 6 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。

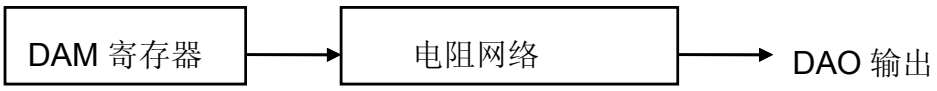


图 12-1. DA 转换框图

为了得到适当的线性输出信号，通常在 DAO 端接一负载电阻。下面给出了 Vdd = 5V/R_L =150Ω 和 Vdd = 3V/R_L =150Ω 时的结果。

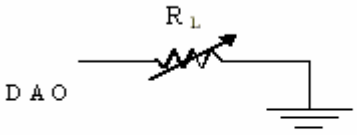


图 12-2 带 R_L 的 DAO 电路

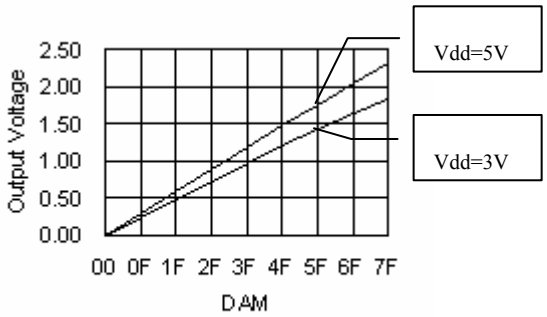


图 13-3 DAC 输出电压（Vdd=5V/3V）

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 初始值 = 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**：DA 转换控制位
0 = 禁止
1 = 使能

Bit [6:0] **DAB [6:0]**：DA 转换数字输入

12.3 D/A 转换说明

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

➲ **例：从 DAO 引脚输出 1/2 VDD**

```
MOV     A, #00111111B
B0MOV   DAM, A          ; 设置 DAB 为 1/2 刻度

B0BSET  FDAENB          ; 使能 DA 功能
```

DAB 的数据和 DAO 的输出电压之间的关系如下：

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

表 12-1 DAB 和 DAO 的关系表

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

➢

13 编程

13.1 编程模板

```
;*********************************************************************************
; 文件名 : TEMPLATE.ASM
; 作者    : SONiX
; 用途    : Template Code for SN8P2710
; 版本    : 05/12/2004 V1.0  First issue
;*********************************************************************************
;*   (c) Copyright 2002, SONiX TECHNOLOGY CO., LTD.
;*********************************************************************************
CHIP     SN8P2715                              ; 选择 IC 型号

;--------------------------------------------------------------------------
;                                包含文档
;--------------------------------------------------------------------------
.nolist                                        ; 在列表文件中不列出
    INCLUDESTD    MACRO1.H
    INCLUDESTD    MACRO2.H
    INCLUDESTD    MACRO3.H

.list                                  ; 允许列表

;--------------------------------------------------------------------------
;                                常量定义
;--------------------------------------------------------------------------
;    ONE              EQU          1
;--------------------------------------------------------------------------
;                                变量定义
;--------------------------------------------------------------------------
.DATA
                org      0h              ;数据放在 bank0 中从地址 0x00 开始的地址
    Wk00B0      DS       1               ;主循环用到的临时变量
    Iwk00B0     DS       1               ;中断用到的临时变量
    AccBuf      DS       1               ;用来保存 ACC 数据的寄存器
    PflagBuf DS      1              ;用来保存 PFLAG 数据的寄存器

                org      100h            ;Bank1 数据区
    BufB1       DS       20              ;bank1 中的临时变量

;--------------------------------------------------------------------------
;                                标志位定义
;--------------------------------------------------------------------------
    Wk00B0_0        EQU          Wk00B0.0          ;Wk00B0 的第 0 位
    Iwk00B0_1    EQU          Iwk00B0.1     ;IWK00 的第 1 位

;--------------------------------------------------------------------------
;                                代码区
;--------------------------------------------------------------------------
.CODE

    ORG             0                       ;代码开始位置
    jmp             Reset                   ;复位向量地址
                                            ;地址 4 到 7 系统保留
    ORG             8
```

```
    jmp         Isr                 ;中断向量地址

    ORG             10h
;-------------------------------------------------------------------------
;   Program reset section
;-------------------------------------------------------------------------
Reset:
    mov             A,#07Fh             ;初始化堆栈指针
    b0mov       STKP,A              ;禁止中断

    clr         PFLAG               ;pflag = x,x,x,x,x,c,dc,z
    mov             A,#00h              ;初始化系统模式
    b0mov           OSCM,A

    mov         A, #0x5A
    b0mov       WDTR, A             ;清看门狗

    call        ClrRAM              ;清 RAM
    call        SysInit             ;系统初始化程序
    b0bset      FGIE                ;使能总中断

;-------------------------------------------------------------------------
;   主程序循环区
;-------------------------------------------------------------------------
Main:
    mov         A, #0x5A            ;清看门狗
    b0mov       WDTR, A

     call       MnApp

    jmp         Main
;-------------------------------------------------------------------------
;   主程序
;-------------------------------------------------------------------------
MnApp:

    ; 在这里仿真主程序
    ret

;-----------------------------------
;   Jump table routine
;-----------------------------------
    ORG             0x0100              ;跳转表的位置最好放在页头
    b0mov       A,Wk00B0
    and         A,#3
     ADD            PCL,A
    jmp         JmpSub0
    jmp         JmpSub1
    jmp         JmpSub2
;-----------------------------------

JmpSub0:
    ; 子程序 1
    jmp         JmpExit
JmpSub1:

    ; 子程序 2
    jmp         JmpExit

JmpSub2:
    ; 子程序 3
```

```
    jmp         JmpExit

JmpExit:
    ret                                 ;返回主程序

;-------------------------------------------------------------------------
; Isr (中断服务程序)
; Arguments   :
; Returns :
; Reg Change:
;-------------------------------------------------------------------------
Isr:
;----------------------------------
;    保存 ACC 和工作寄存器的值
;----------------------------------
    b0xch       A,AccBuf        ;使用 B0XCH 不会影响到 C，Z 标志
    B0MOV       A, PFLAG
    B0MOV       PFLAGBUF, A

;----------------------------------
;   检查是否有中断发生
;----------------------------------

IntP00Chk:
    b0bts1      FP00IEN
    jmp         IntTc0Chk           ;在其他的中断里调整这里
     b0bts0         FP00IRQ
    jmp         P00isr

    ;如果需要，可以在这里插入其他的中断

IntTc0Chk:
    b0bts1      FTC0IEN
    jmp         IsrExit             ;
    b0bts0      FTC0IRQ                 ;
    jmp         TC0isr
;----------------------------------
; 退出中断
;----------------------------------
IsrExit:
    B0MOV       A, PFLAGBUF                         ;
    B0MOV       PFLAG, A
    b0xch       A,AccBuf        ;使用 B0XCH 不会影响到 C，Z 标志

    reti                        ;中断返回

;-------------------------------------------------------------------------
;    INT0 中断服务程序
;-------------------------------------------------------------------------
P00isr:
    b0bclr      FP00IRQ

    ;在这里处理外部中断

    jmp         IsrExit

;-------------------------------------------------------------------------
;    TC0 中断服务程序
;-------------------------------------------------------------------------
TC0isr:
    b0bclr      FTC0IRQ
```

```
    ;在这里处理 TC0 中断

    jmp         IsrExit

;----------------------------------------------------------------------
;   系统初始化程序
;   初始化 I/O，定时器，中断等
;----------------------------------------------------------------------
SysInit:
    ret

;----------------------------------------------------------------------
; 清 RAM
; 使用@YZ 寄存器清 RAM   (00h~7Fh)
;----------------------------------------------------------------------

ClrRAM:

; RAM Bank 0
    clr         Y                   ;选择 bank0
    b0mov       Z,#0x7f             ;设置@YZ 地址为 7fh

ClrRAM10:
    clr         @YZ                     ;清@YZ
    decms       Z                       ;z = z – 1，若 Z = 0 则跳过下一条指令
    jmp             ClrRAM10
    clr         @YZ                     ;清地址 0x00
    ret

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

13.2 程序检查对照表

项目	说明
未定义位	系统寄存器中所有标记为“0”（未定义）的位都必须置“0”，以避免系统出错。
ADC	1. ADC 的输入 I/O 口设为输入模式； 2. 禁止 ADC 输入引脚的上拉电阻； 3. 进入省电模式省电前禁止 ADC； 4. 在省电模式下设置 P4CON 的有关位避免额外的功耗； 5. 使能 ADC（ADENB = 1）后延迟 100uS 等待 ADC 电路准备好转换； 6. 在进入睡眠模式前禁止 ADC（ADENB = 0)以省电
中断	RAM 初始化前禁止中断
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	小心子程序的调用和中断操作，避免堆栈溢出
系统初始化	1. 堆栈初始化：STKP = 0x7F，禁止总中断； 2. 清 RAM； 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. 使能看门狗选项中的“Always On”选项，保护系统； 2. 使能编译选项中的“Noise Filter”选项； 3. 未使用的 I/O 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	Mnemonic		Description	C	DC	Z	Cycle
	MOV	A,M	A (M	-	-	√	1
M	MOV	M,A	M (A	-	-	-	1
O	B0MOV	A,M	A (M (bnak 0)	-	-	√	1
V	B0MOV	M,A	M (bank 0) (A	-	-	-	1
E	MOV	A,I	A (I	-	-	-	1
	B0MOV	M,I	M (I, (M = Working registers & 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
	ADC	A,M	A (A + M + C, if occur carry, then C=1, else C=0	√	√	√	1
A	ADC	M,A	M (A + M + C, if occur carry, then C=1, else C=0	√	√	√	1+N
R	ADD	A,M	A (A + M, if occur carry, then C=1, else C=0	√	√	√	1
I	ADD	M,A	M (M + A, if occur carry, then C=1, else C=0	√	√	√	1+N
T	B0ADD	M,A	M (bank 0) (M (bank 0) + A, if occur carry, then C=1, else C=0	√	√	√	1+N
H	ADD	A,I	A (A + I, if occur carry, then C=1, else C=0	√	√	√	1
M	SBC	A,M	A (A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1
E	SBC	M,A	M (A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1+N
T	SUB	A,M	A (A - M, if occur borrow, then C=0, else C=1	√	√	√	1
I	SUB	M,A	M (A - M, if occur borrow, then C=0, else C=1	√	√	√	**1+N**
C	SUB	A,I	A ← A - I, if occur borrow, then C=0, else C=1	√	√	√	1
	AND	A,M	A ← A and M	-	-	√	1
L	AND	M,A	M ← A and M	-	-	√	**1+N**
O	AND	A,I	A ← A and I	-	-	√	1
G	OR	A,M	A ← A or M	-	-	√	1
I	OR	M,A	M ← A or M	-	-	√	**1+N**
C	OR	A,I	A ← A or I	-	-	√	1
	XOR	A,M	A ← A xor M	-	-	√	1
	XOR	M,A	M ← A xor M	-	-	√	**1+N**
	XOR	A,I	A ← A xor I	-	-	√	1
	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
P	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	**1+N**
R	RRC	M	A ← RRC M	√	-	-	1
O	RRCM	M	M ← RRC M	√	-	-	1+N
C	RLC	M	A ← RLC M	√	-	-	1
E	RLCM	M	M ← RLC M	√	-	-	1+N
S	CLR	M	M ← 0	-	-	-	1
S	BCLR	M.b	M.b ← 0	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1+N
	CMPRS	A,I	ZF,C ← A - I, If A = I, then skip next instruction	√	-	√	1 + S
B	CMPRS	A,M	ZF,C ← A - M, If A = M, then skip next instruction	√	-	√	1 + S
R	INCS	M	A ← M + 1, If A = 0, then skip next instruction	-	-	-	1 + S
A	INCMS	M	M ← M + 1, If M = 0, then skip next instruction	-	-	-	1+N+S
N	DECS	M	A ← M - 1, If A = 0, then skip next instruction	-	-	-	1 + S
C	DECMS	M	M ← M - 1, If M = 0, then skip next instruction	-	-	-	1+N+S
H	B0BTS0	M.b	If M(bank 0).b = 0, then skip next instruction	-	-	-	1 + S
	B0BTS1	M.b	If M(bank 0).b = 1, then skip next instruction	-	-	-	1 + S
	JMP	D	PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	D	Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
M	RET		PC ← Stack	-	-	-	2
I	RETI		PC ← Stack, and to enable global interrupt	-	-	-	2
S	NOP		No operation	-	-	-	1
C	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存储器。
2. 若跳转的条件为真，则“S”=0，否则“S”=1。
3. 若“M”是系统寄存器（bank0 中的 80h~FFh），则“N”=0，否则“N”=1。

15 电气特性

15.1 极限参数

(所有电压以 Vss 为参考基准)

电源电压(Vdd)..... - 0.3V ~ 6.0V
输入电压(Vin).....Vss - 0.2V ~ Vdd + 0.2V
工作环境温度(Topr)
SN8P2714K、SN8P2714S、SN8P2715P、SN8P2715S.....–20℃ ~ + 70℃
SN8P2714AKD、SN8P2714ASD、SN8P2715APD、SN8P2715ASD..... –40℃ ~ + 85℃
存储环境温度(Tstor) ..... –30℃ ~ + 125℃
功耗(Pc).....500 mW

15.2 电气特性

(所有电压以 Vss，Vdd = 5.0V 为参考值，fosc = 4 MHz，环境温度为 25℃)

参数	符号	说明		最小值	标准值	最大值	单位
工作电压	Vdd	普通模式，Vpp = Vdd		2.2	5.0	5.5	V
		编程模式，Vpp = 12.5V		4.5	5.0	5.5	
RAM 数据保持电压	Vdr			-	1.5	-	V
内部 POR	Vpor	Vdd 的上升速率以确保内部的上电复位		-	0.05	-	V/ms
输入低电压	ViL	施密特触发输入		Vss	-	0.3Vdd	V
输入高电压	ViH	施密特触发输入		0.7Vdd	-	Vdd	V
复位引脚漏电流	Ilekg	Vin = Vdd		-	-	2	uA
上拉电阻	Rup	Vin = Vss , Vdd = 3V		-	200	-	KΩ
		Vin = Vss , Vdd = 5V		-	100	-	KΩ
I/O 口输入漏电流	Ilekg	禁止上拉电阻，Vin = Vdd		-	-	2	uA
P2、P4、P5 的输出源电流	IoH	Vop = Vdd - 0.5V		-	12	-	mA
灌电流	IoL	Vop = Vss + 0.5V		-	15	-	mA
INTn 触发脉冲宽度	Tint0	INT0 ~ INT1 中断请求脉冲宽度		2/fcpu	-	-	Cycle
AVREFH 输入电压	Varfh	Vdd = 5.0V		2V	-	Vdd	V
AIN0 ~ AIN7 输入电压	Vani	Vdd = 5.0V		0	-	Varfh	V
ADC 使能时间	Tast	置 ADENB = 1 后准备开始转换		-	100	-	uS
电源电流	Idd1	运行模式 Fcpu = Fosc/4	Vdd= 5V 4MHz	-	3	5	mA
			Vdd= 3V 4MHz	-	1.5	3	mA
			Vdd= 3V 32768Hz	-	10	20	uA
	Idd2	内部 RC 模式	Vdd= 5V ~ 32KHz	-	25	50	uA
			Vdd= 3V ~ 16KHz	-	5	10	uA
	Idd3	睡眠模式 （LVD = LVD0）	Vdd= 5V	-	1	2	uA
			Vdd= 3V	-	0.5	1	uA
	Idd4	睡眠模式 （LVD = LVD1）	Vdd= 5V	-	2	4	uA
			Vdd= 3V	-	1	2	uA
LVD 侦测电压	Vdet0	低电压侦测电平		-	2.0	-	V
LVD 指示电压	Vdet1	低电压指示电平		-	2.4	-	V
DAC 全程输出电流	I_{FSO}	Vdd=5V, RL =150 ohm		-	12	-	mA

16 开发工具

16.1 开发工具版本

16.1.1 ICE (在线仿真器)

- **SN8ICE 2K**：全功能仿真 SN8P271X 系列。

16.1.2 OTP 烧录器

- **Writer 3.0**：支持 SN8P2715/SN802714，但是没有脱机模式。
- **Easy Writer V1.0**：由 ICE 控制 OTP 编程，且没有升级硬件的烦恼。详见 easy writer 的用户手册。
- **MP-Easy Writer V1.0**：可脱机烧录，支持 SN8P2715/SN8P2714 的大规模烧录。

16.1.3 IDE (综合的开发工具)

SONiX 8 位 MCU 综合的开发工具包括由编译器、ICE 调试器和 OTP 烧录软件。

- **For SN8ICE 2K**：M2IDE V1.00 或以后的版本。
- **For Writer 3.0 and Easy Writer**：SN8IDE V1.99N 或以后的版本。

16.2 OTP 烧录引脚

16.2.1 EZ-MP Writer 转接板的引脚配置

Easy Writer JP1/JP2

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

MP 转接板 JP1
Writer3.0 转接板 JP2

Easy Writer JP3

DIP1	1	48	DIP48
DIP2	2	47	DIP47
DIP3	3	46	DIP46
DIP4	4	45	DIP45
DIP5	5	44	DIP44
DIP6	6	43	DIP43
DIP7	7	42	DIP42
DIP8	8	41	DIP41
DIP9	9	40	DIP40
DIP10	10	39	DIP39
DIP11	11	38	DIP38
DIP12	12	37	DIP38
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

MP 转接板 JP3

16.2.2 Writer3.0 和 Writer2.5 转接板的引脚配置

GND	**1**	**2**	**VDD**
CE	**3**	**4**	**CLK**
OE	**5**	**6**	**PGM**
D0	**7**	**8**	**D1**
D2	**9**	**10**	**D3**
D4	**11**	**12**	**D5**
D6	**13**	**14**	**D7**
VPP	**15**	**16**	**VDD**
RST	**17**	**18**	**HLS**

Writer V2.5 JP1 引脚配置

GND	**2**	**1**	**VDD**
CE	**4**	**3**	**CLK**
OE	**6**	**5**	**PGM**
D0	**8**	**7**	**D1**
D2	**10**	**9**	**D3**
D4	**12**	**11**	**D5**
D6	**14**	**13**	**D7**
VPP	**16**	**15**	**VDD**
RST	**18**	**17**	**HLS**
	20	**19**	

Writer V3.0 JP1 引脚配置

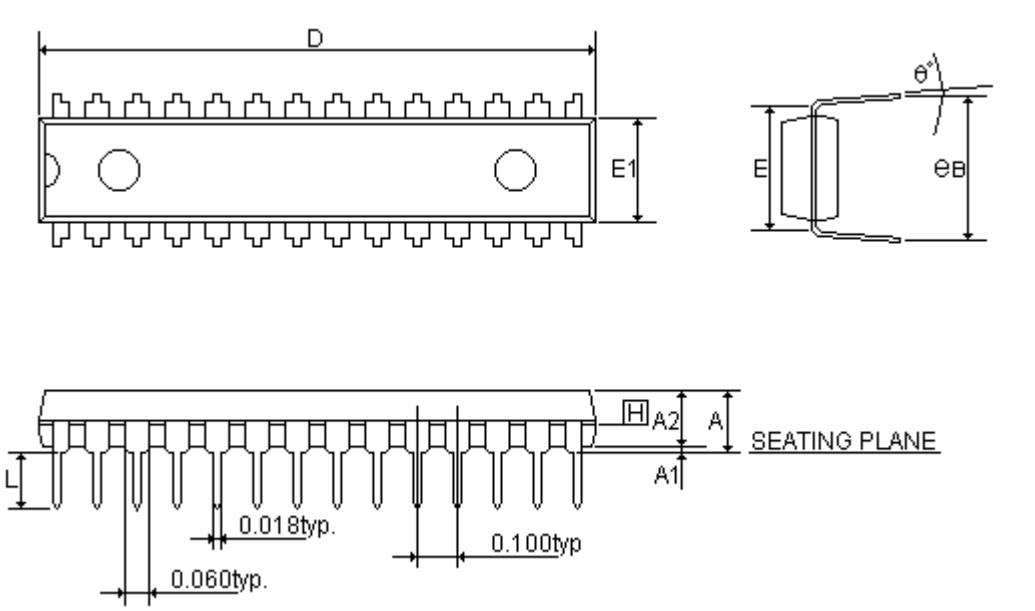
✸ **注：烧录 SN8P2715/SN8P2714 的产品时，SONIX Writer2.5 必须升级为 Writer3.0。有关 SONIX Writer2.5 的升级问题请联系 SONIX 的代理商。.**

16.2.3 SN8P271x 系列烧录引脚对应表

OTP Programming Pin of SN8P2710 Series											
Chip Name		SN8P2714		SN8P2715							
EZ Writer / Writer V3.0		OTP IC / JP3 Pin Assignment									
Number	Pin	Number	Pin	Number	Pin						
1	VDD	25	VDD	30	VDD						
2	GND	15	VSS	20	VSS						
3	CLK	4	P5.0	6	P5.0						
4	CE	-	-	-	-						
5	PGM	8	P2.0	10	P2.0						
6	OE	3	P5.1	5	P5.1						
7	D1	-	-	-	-						
8	D0	-	-	-	-						
9	D3	-	-	-	-						
10	D2	-	-	-	-						
11	D5	-	-	-	-						
12	D4	-	-	-	-						
13	D7	-	-	-	-						
14	D6	-	-	-	-						
15	VDD	25	VDD	30	VDD						
16	VPP	26	RST	31	RST						
17	HLS	-	-	-	-						
18	RST	-	-	-	-						
19	-	-	-	-	-						
20	ALSB/PDB	9	P2.1	11	P2.1						

17 封装信息

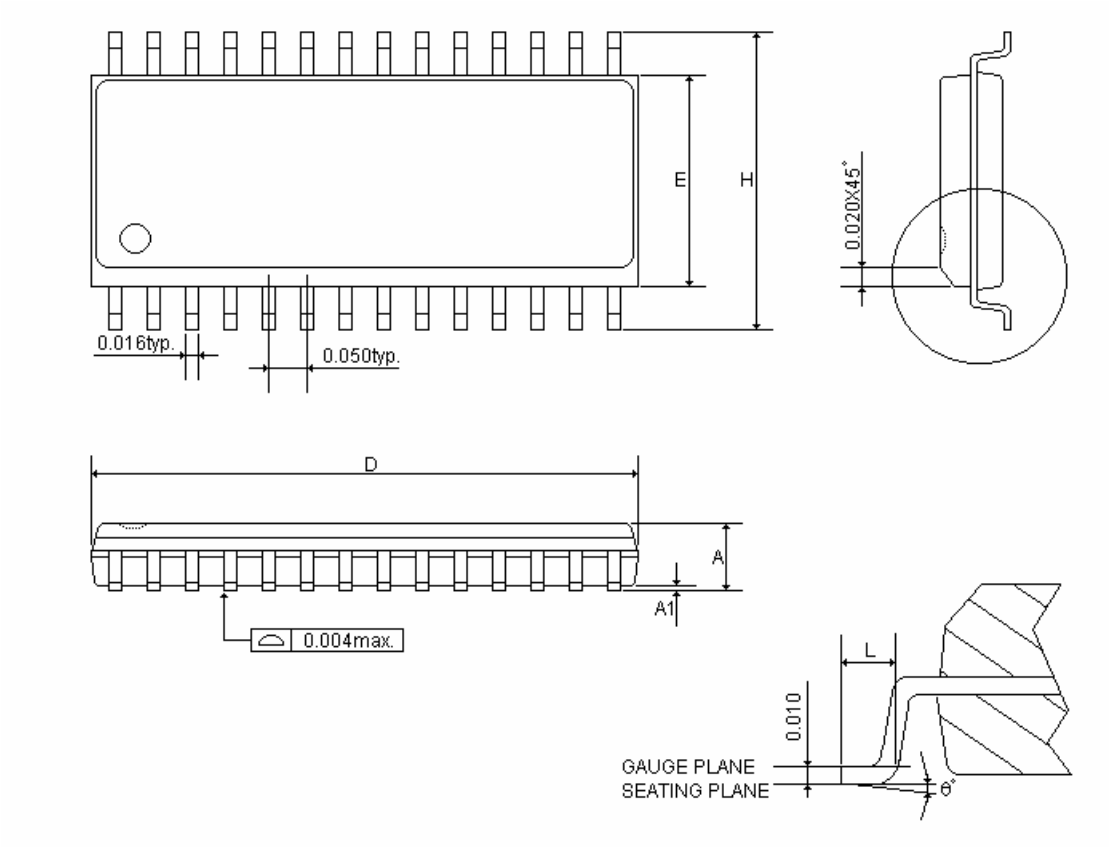
17.1 SK-DIP28 PIN



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

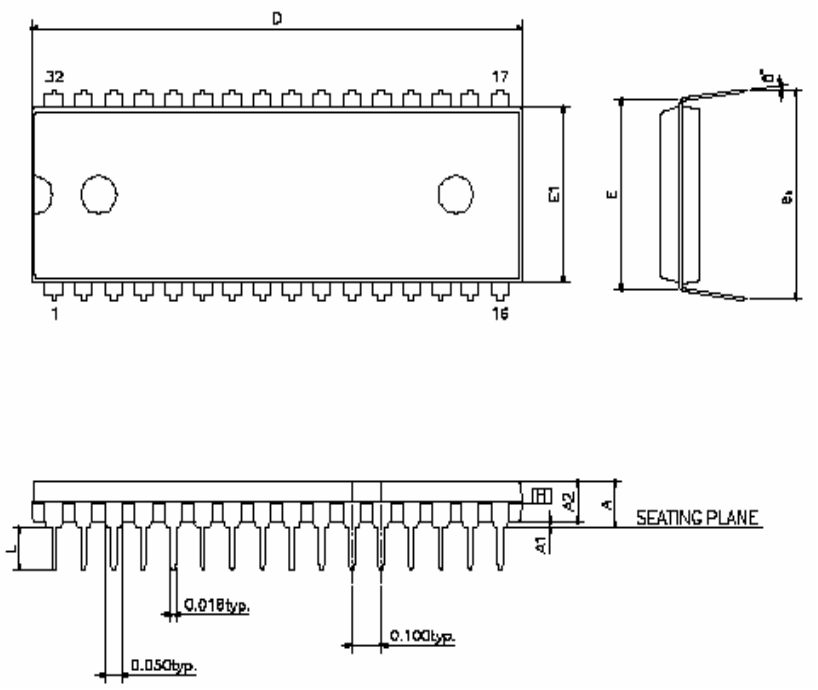
17.2 SOP28 PIN



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

17.3 P-DIP 32 PIN

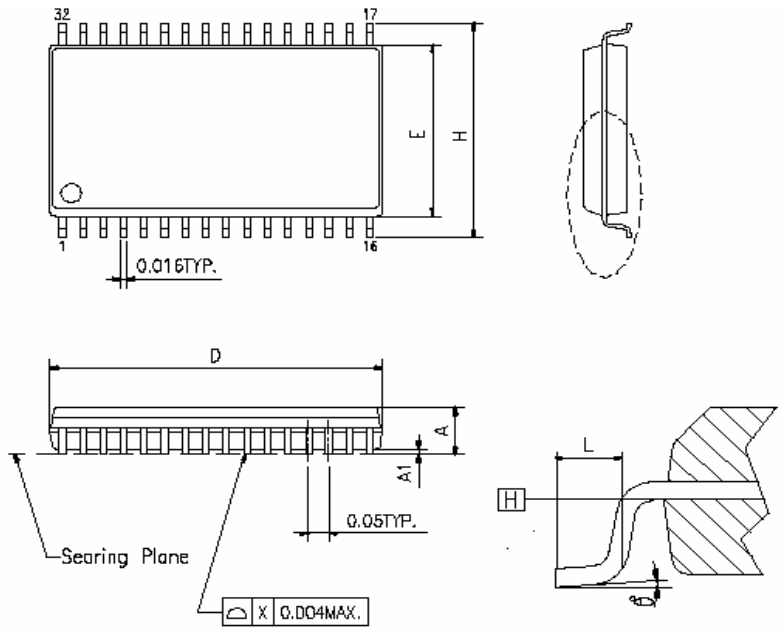


SYMBOLS	MIN.	NOR.	MAX.
A	–	–	0.220
A1	0.015	–	–
A2	0.150	0.155	0.160
D	1.645	1.650	1.660
E		0.600 BSC	
E1	0.540	0.545	0.550
L	0.115	0.130	0.150
e_B	0.630	0.650	0.670
θ°	0	7	15

UNIT : INCH

NOTE:
1.JEDEC OUTLINE : MS-011 AC

17.4 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 H 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 H
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做进一步说明的权利。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。

Main Office:
Address: 9F, NO. 8, Hsien Cheng 5th St, Chupei City, Hsinchu, Taiwan R.O.C.
Tel: 886-3-551 0520
Fax: 886-3-551 0523

Taipei Office:
Address: 15F-2, NO. 171, Song Ted Road, Taipei, Taiwan R.O.C.
Tel: 886-2-2759 1980
Fax: 886-2-2759 8180

Hong Kong Office:
Address: Flat 3 9/F Energy Plaza 92 Granville Road, Tsimshatsui East Kowloon.
Tel: 852-2723 8086
Fax: 852-2723 9179

Technical Support by Email:
Sn8fae@sonix.com.tw

深圳技术支持中心：
深圳市科技园南区 T2-B 栋 2 楼
电话：0755-26719666
传真：0755-26719786