SN8P2949

用户参考手册

Version 1.1

SONiX 8 位单片机

修改历史

版本	时间	修改说明
V1.0	2012.04	初版。
V1.1	2012.12	1、在产品资源配置表中添加 1929 的相关信息。 2、调整 ADC Offset 选项为：(0, -1/4, -1/2, -3/4)*Vref。 3、增加注释：选择 ADC Offset 功能时，ENOB 会跌落。 4、调整 ADC 增益选项：1x，2x 和 4x。 5、在指令集中增加部分指令的指令周期。 6、升级 ADC ENOB/Noise 表：删除 ADC x4/x8 的相关数据。 7、在成品自立配置改良表中增加 1929 的相关信息。 8、修改 STKnH 寄存器：从 3 位更改为 5 位。 9、在系统高速时钟章节中，增加 16 分频和 32 分频的选项。 10、调整应用电路：VDD 和 AVDD 都有独立的电容。 11、在应用电路下方增加注释：电容要尽可能地靠近 IC 引脚。 12、将 ISP 模式的建议烧录时间调整为 30us，并更改示例程序中的相关设置。

目录

1 产品简介

1.1 产品性能表

单片机名称	ROM	RAM	堆栈	LCD	定时器			I/O	ADC	PWM Buzzer	RTC	唤醒功能引脚数目	封装形式
					T0	TC0	TC1						
SN8P1919	6K*16	256*8	8	4*32	V	V	V	22	16-bit	V	V	7	LQFP80
SN8P1929	4K*16	256*8	8	4*24	V	V	V	16	16-bit	V	V	6	LQFP80
SN8P2949	8K*16	256*8	8	4*32	V	V	V	20	20-bit	V	V	13	LQFP80

1.2 产品资源配置改良表

ITEM	SN8P1919	SN8P1929	SN8P2949
电源输入（VDD）	2.4V ~ 5.5V	2.4V ~ 5.5V	2.4V ~ 3.6V
功耗	多	多	少
PGIA 增益设置	1x, 12.5x, 50x, 100x, 200x	1x, 12.5x, 50x, 100x, 200x	1x, 12.5x, 50x, 100x, 200x
AVE+电压	3.0V, 2.4V or 1.5V	3.0V, 2.4V or 1.5V	2.0V or 1.5V
AVDDR 电压	3.8V	3.8V	2.4V
ACM 电压（仅灌电流）	1.2V	1.2V	1.0V
AVE / AVDDR 双倍电流	是	是	否
Charge pump 模拟电压（AVDDCP）	是	是	否
内部 ADC 参考电压 V(R+, R-)	0.8V, 0.64Vor 0.4V	0.8V, 0.64Vor 0.4V	0.3V ~ 0.8V
AO±和 X± Pads	是	仅 X±	否
ADC 增益	无	无	x1 or x2
ADC 补偿功能	无	无	有（0 或者-1/4*Vref）
ADC 分辨率	16-Bit	16-Bit	20-Bit
Channel Switching 后的 ADC 稳定时间	慢（0.5s）	慢（0.5s）	快（3 x ADC_WR）
ADC WR (Output Rate)	1Hz ~ 50Hz	1Hz ~ 50Hz	1Hz ~ 3.9kHz
ADC 快速测量	否	否	是
ADC 在低速/绿色模式下工作的唤醒功能	无	无	有
电池检测方法	通过 ADC 或者比较器	通过 ADC 或者比较器	通过 ADC 或者比较器（LBT 2.2V ~ 2.9V）
高速时钟	IHRC 或者 4MHz Crystal	IHRC 或者 4MHz Crystal	IHRC
LCD 类型	R	R	R/C
VLCD 和 VLCD1 电压	不同	VLCD	VLCD
电容需求	多	多	少
LED 背光电源	AVDDR 或者 AVE+	AVDDR 或者 AVE+	VLED (3.5V)
ISP 功能用于传感器矫正	无	有	有
ISP Vpp 源和电压	无	外部 12.5V	外部 6.5V 或内部产生器

1.3 功能特性

◆ **存储器配置**
OTP ROM：8K * 16 位
RAM：256 * 8 位（bank 0, bank 1）
8 层堆栈缓存器
LCD RAM：4*32 位

◆ **I/O 引脚配置**
双向输入输出端口：P0，P1，P2，P5
具有唤醒功能的端口：P0，P1
内置上拉电阻的端口：P0，P1，P2，P5
外部中断引脚：P00，P01

◆ **功能强大的指令系统**
一条指令需要 4 个时钟周期
所有的指令均为一个字长
绝大部分指令只需要一个周期
指令的最长周期为 2 个指令周期
JMP 指令可在整个 ROM 区执行
查表功能（MOVC）可寻址整个 ROM 区

◆ **3 个 8 位定时器**
T0：基本定时器，捏制 0.5S RTC 模式
TC0：自动重装定时器/计数器/PWM0/Buzzer
TC1：自动重装定时器/计数器/PWM1/Buzzer

◆ **可编程增益放大器**

◆ **PGIA 增益选项：1x/12.5x/50x/100x/200x**

◆ **20 位 Delta-Sigma ADC，带 18 位 Noise Free**
ADC 增益选项：1x，2x，4x
ADC 零点调整选项：(-1/4，-1/2，-3/4 or 0) x Vref
ADC 中断，具有绿色模式唤醒功能
3 种 ADC 通道配置
2 个全差分通道
1 个差分和 2 个单端输入通道
4 个单端输入通道

◆ **6 个中断源**
4 个内部中断：T0，TC0，TC1，ADC
2 个外部中断：INT0，INT1

◆ **单电源输入：2.4V ~ 3.6V（模拟部分）；2.2V ~ 3.6V（数字部分）**

◆ **内置看门狗定时器**

◆ **内置 2.4V 电压输出和 5mA 驱动电流的 Regulator（AVDDR）**

◆ **内置 3.5V 电压输出和 8mA LED 驱动电流的 Charge Pump Regulator（VLED）**

◆ **内置 2.0V/1.5V 电压输出的 Regulator（AVE+），负载电流功耗非双倍**

◆ **内置 1.2V Band Gap，用来监控电池电压**

◆ **内部 LBT 2.2V~2.9V；或者外部 P52 输入电池低电压检测**

◆ **内置 ADC 参考电压：内部 0.3V~ 0.8V**
外部参考电压输入 V(R+,R-)

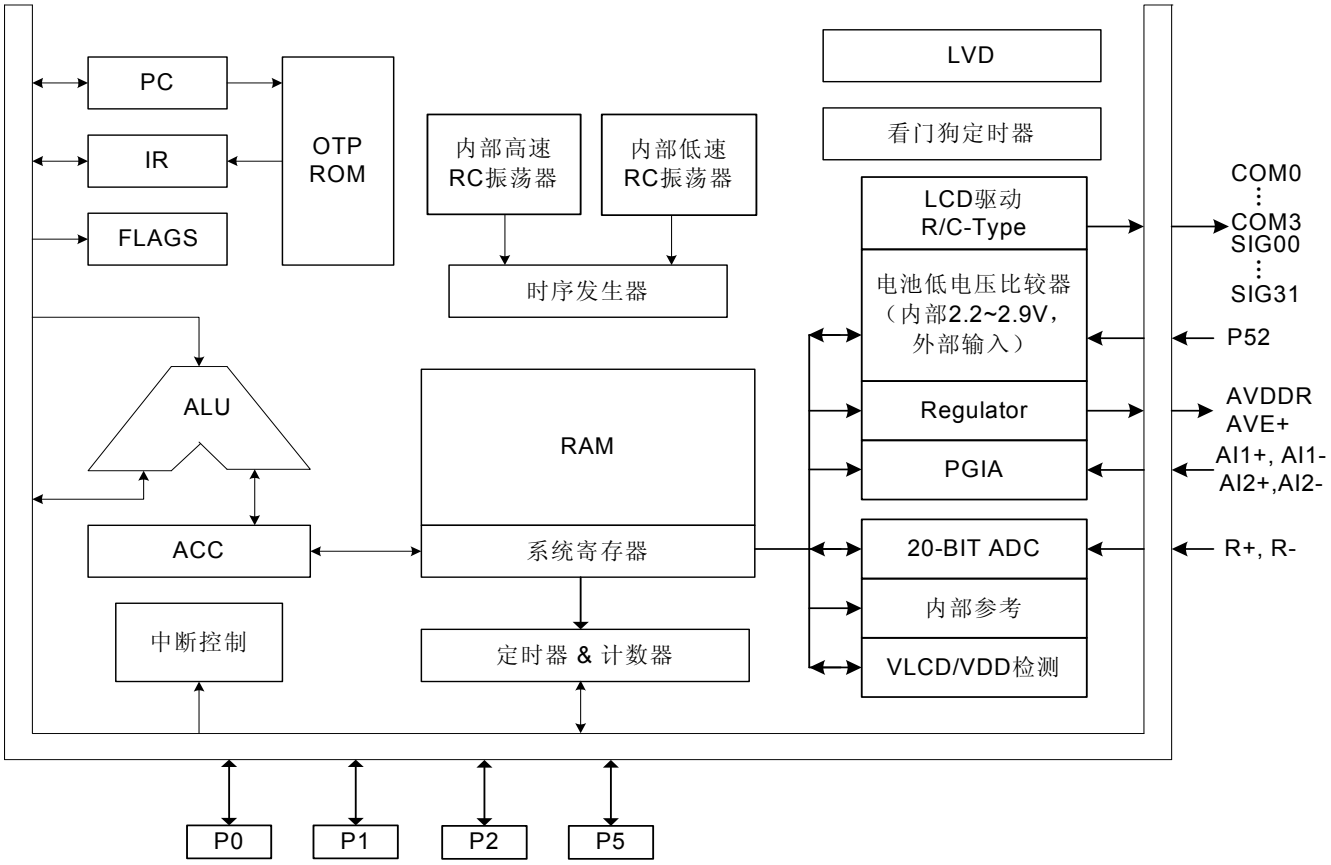
◆ **ROM 在线烧录，内建 6.5V VPP 电压产生器**

◆ **LCD 驱动**
1/3 或 1/2 偏压
4 common * 32 segment
包括 R 型和 C 型 LCD。
C 型 LCD 电压：2.7 ~3.4V。
R 型可选偏压电阻：400k，200k，100k，33k
C 型 LCD 占空比控制功能

◆ **双时钟系统提供 4 种工作模式**
高速时钟：IHRC 4 MHz
低速时钟：ILRC 32kHz 或者 32768 晶振@IHRC_RTC
普通模式：高低速时钟都正常工作
低速模式：低速时钟正常工作，高速时钟可选择性工作。
绿色模式：低速时钟正常工作，高速时钟可选择性工作
由 P0/T0/TC0/ADC 唤醒
睡眠模式：高低速时钟都停止工作

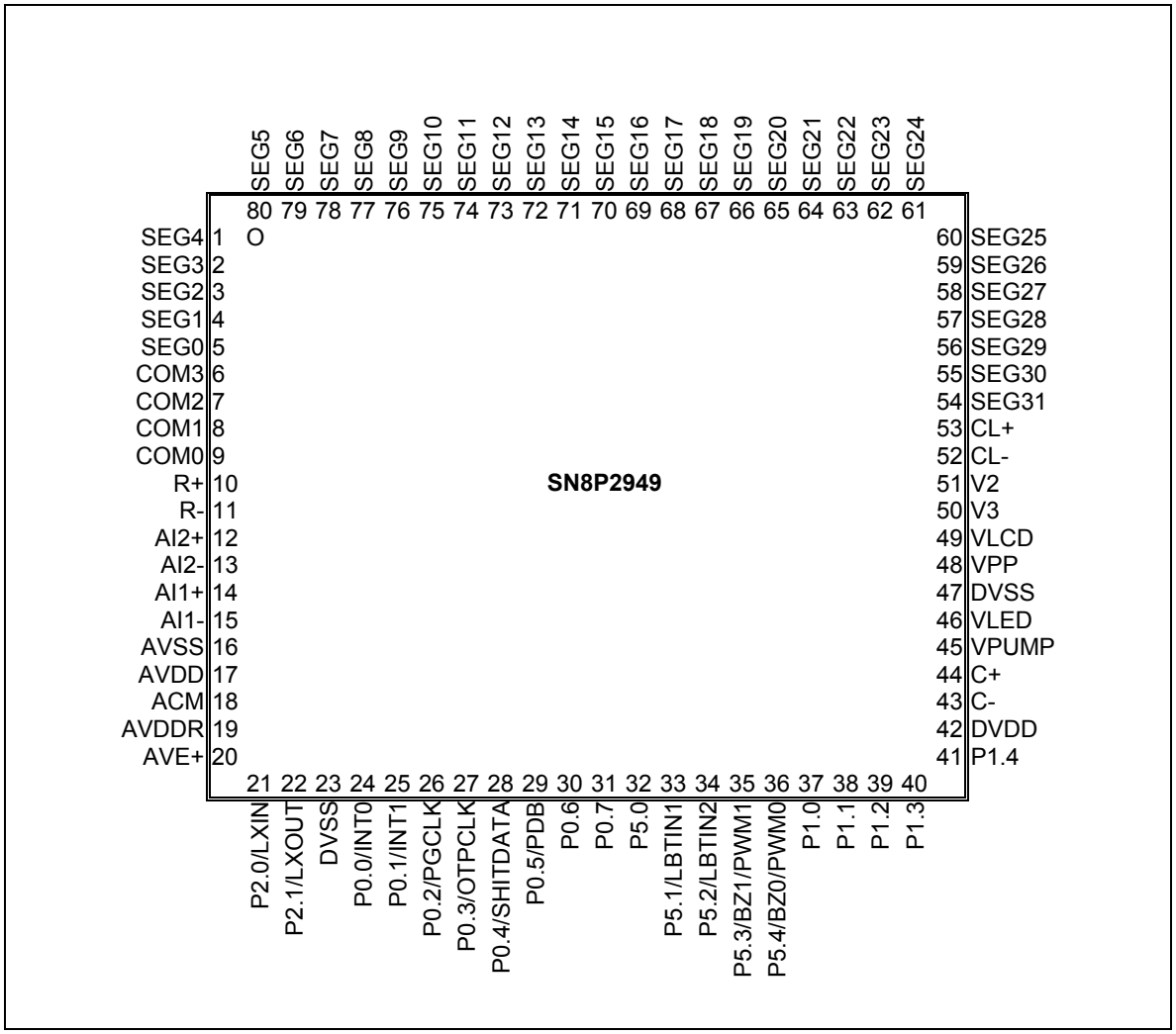
◆ **封装形式**
Dice /LQFP80

1.4 系统框图


PC
IR
FLAGS
OTP ROM
内部高速 RC振荡器
内部低速 RC振荡器
时序发生器
LVD
看门狗定时器
LCD驱动 R/C-Type
电池低电压比较器 （内部2.2~2.9V， 外部输入）
Regulator
PGIA
20-BIT ADC
内部参考
VLCD/VDD检测
COM0
COM3
SIG00
SIG31
P52
AVDDR
AVE+
AI1+, AI1-
AI2+,AI2-
R+, R-
ALU
RAM
ACC
系统寄存器
中断控制
定时器 & 计数器
P0
P1
P2
P5

1.5 引脚配置

SN8P2949 LQFP80

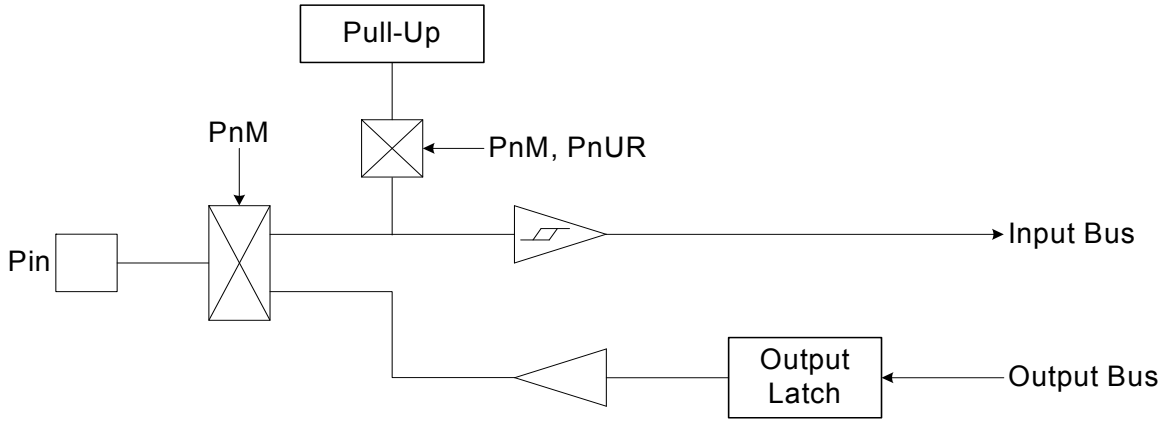


1.6 引脚说明

引脚名称	类型	功能说明
DVDD, DVSS	P	数字电源输入端。
AVDD, DVSS	P	模拟电源输入端。
CL+, CL-	P	Charge pump 引脚（VLCD），连接一个 0.1uF 的电容。
C+, C-	P	Charge pump 引脚（VLED），连接一个 1uF 的电容。
VPUMP	P	Charge pump 输出电压（2 x VDD）（LED regulator），与 GND 之间连接一个 1uF 的电容。
VLED	P	Charge pump regulator 输出电压（LED 电源），与 GND 之间连接一个 1uF 的电容。 VLED = 3.5V，最大输出电流为 8mA，在 VDD 范围类工作，其范围为 2.4V~3.6V。
AVDDR	P	Regulator 电源输出引脚，电压为 2.4V； AVDDR 为模拟电路电压源（PGIA, ADC...）。
AVE+	P	传感器 Regulator 输出 2.0/1.5V，最大输出电流为 5mA。与 GND 之间连接一个 1uF 的电容。
ACM	P	ACM 输出电压为 1V，与 AVDDR 之间连接一个 0.1uF 的电容。
R+	AI	ADC 参考源的正极输入端，R+输入的绝对电压范围为 AVDDR-1V 到 GND+0.4V，外部参考源可以在 R+和 R-之间应用。
R-	AI	ADC 参考源的负极输入端，R+输入的绝对电压范围为 AVDDR-1V 到 GND+0.4V。
AI1+, AI1-	AI	PGIA 通道的正负极输入端，每个输入端的绝对电压范围为 AVDDR-1V 到 GND+0.4V。
AI2+, AI2-	AI	PGIA 通道的正负极输入端，每个输入端的绝对电压范围为 AVDDR-1V 到 GND+0.4V。
VPP	P	OTP ROM 烧录引脚。从该引脚输入 6.5V 电压进行 OTP 烧录。 IC 正常工作时，保持电压等于小于 VDD。无复位功能。
P0.0 / INT0	I/O	P0.0 与 INT0 触发引脚（施密特触发）共用，内置上拉电阻。
P0.1 / INT1	I/O	P0.1 与 INT1 触发引脚（施密特触发）共用，内置上拉电阻。
P0 [7:0]	I/O	双向输入输出引脚，具有绿色模式/睡眠模式唤醒功能，内置上拉电阻。
P1 [4:0]	I/O	双向输入输出引脚，内置上拉电阻，由 P1UR 控制，其唤醒功能由 P1W 控制。
P2 [1:0]	I/O	双向输入输出引脚，内置上拉电阻，由 P2UR 控制，引脚与 LXIN/LXOUT 共用。
P5 [4:0]	I/O	双向输入输出引脚，内置上拉电阻，由 P5UR 控制。
COM [3:0]	O	COM0~COM3 LCD 驱动 common 端口。
LBTIN1/LBTIN2	I	LBTIN1（P51）为内部接地引脚，用来检测电池低电压。 LBTIN2（P52）为外部属于引脚，用来检测电池低电压（比较器输入引脚）。
SEG0 ~ SEG31	O	LCD 驱动 segment 引脚。
VLCD	P	LCD 驱动电源引脚；当选择 R 型 LCD 模式时，VLCD 引脚输入电源。
V3	P	选择 LCD 1/3 偏压模式时，2/3 VLCD 偏压输出引脚。
V2	P	选择 LCD 1/3 偏压模式时，1/3 VLCD 偏压输出引脚。

1.7 引脚电路结构图

P0，P1，P2 和 P5：



2 中央处理器（CPU）

2.1 存储器

2.1.1 程序存储器ROM

☞ **ROM：8K words**

2.1.2 复位向量（0000H）

具有一个字长的系统复位向量（0000H）。

- **上电复位；**
- **看门狗复位。**

发生上述任一种复位后，程序将从 0000H 处重新开始执行，系统寄存器也都将恢复为默认值。下面一段程序演示了如何定义 ROM 中的复位向量。

➢ **例：定义复位向量。**

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

        ORG     10H
START:                          ; 0010H，用户程序起始地址。
        …                       ; 用户程序。
        …

        ENDP                    ; 程序结束。
```

2.1.2.1 中断向量（0008H）

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

✷ **注：在中断发生时，用户必须用程序保存 ACC 和 PFLAG。**

➢ **例：定义中断向量，中断服务程序紧随 ORG 8 之后。**

```
.DATA       ACCBUF      DS   1          ;
            PFLAGBUF    DS   1          ;
.CODE
            ORG         0              ; 0000H。
            JMP         START          ; 跳到用户程序。
            …
            ORG         8H             ;
            B0XCH       A, ACCBUF      ; 保存 ACC。
            B0MOV       A, PFLAG
            B0MOV       PFLAGBUF, A    ; 保存 PFLAG。
            …
            B0MOV       A, PFLAGBUF
            B0MOV       PFLAG, A       ; 恢复 PFLAG。
            B0XCH       A, ACCBUF      ; 恢复 ACC。
            RETI                       ; 中断返回。
            …
START:                                 ; 用户程序开始。
            …                          ; 用户程序。
            JMP         START          ; 用户程序结束。
            …
            ENDP                       ;
```

➢ **例：定义中断向量，中断服务程序位于用户程序的后面。**

```
.DATA       ACCBUF      DS   1
            PFLAGBUF    DS   1
.CODE
            ORG         0              ; 0000H。
            JMP         START          ; 跳到用户程序。
            …
            ORG         8H             ;
            JMP         MY_IRQ         ; 跳到中断服务程序。

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

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

2.1.2.2 查表

在 SONiX 单片机中，对 ROM 区中的数据进行查找，寄存器 Y 指向所找数据地址的中间字节（bit8~bit15），寄存器 Z 指向所找数据地址的低字节（bit0~bit7）。执行完 MOVC 指令后，所查找数据低字节内容被存入 ACC 中，而数据高字节内容被存入 R 寄存器。

➢ **例：查找 ROM 地址为“TABLE1”的值。**

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

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

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

➢ **例：宏 INC_YZ。**

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

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

➢ **例：通过“INC_YZ”对上例进行优化。**

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

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

下面的程序通过累加器对 Y，Z 寄存器进行处理来实现查表功能，但需要特别注意进位时的处理。

➢ **例：由指令 B0ADD/ADD 对 Y 和 Z 寄存器加 1。**

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

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

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

2.1.2.3 跳转表

跳转表能够实现多地址跳转功能。PCL 和 ACC 的值相加即可得到新的 PCL。由此得到的新的 PC 值指向一系列跳至指令列表。

执行“ADD PCL, A”后，如果有进位发生，结果并不会影响 PCH 寄存器。用户必须检查跳转表是否跨越了 ROM 的页边界。如果跳转表跨越了 ROM 页边界（例如从 xxFFH 到 xx00H），将跳转表移动到下一页程序存储区的顶部（xx00H）。注：一页包含 **256words**。

✷ **注：在执行加法运算后，若 PCL 溢出，程序计数器 PC 并不会从 PCL 进位到 PCH。**

➢ **例：跳转表。**

```
        ORG     0100H       ; 跳转表从 ROM 前端开始。

        B0ADD   PCL, A      ; PCL = PCL + ACC，PCH 的值不会被改变。
        JMP     A0POINT     ; ACC = 0，跳至 A0POINT。
        JMP     A1POINT     ; ACC = 1，跳至 A1POINT。
        JMP     A2POINT     ; ACC = 2，跳至 A2POINT。
        JMP     A3POINT     ; ACC = 3，跳至 A3POINT。
```

在下面的例子中，跳转表从 00FDH 开始，执行 B0ADD PCL, A 后，如果 ACC = 0 或者 1，跳转表指向正确的地址，但如果 ACC 大于 1，因为 PCH 不能自动加一，程序就会出错。可以看到当 ACC = 2 时，PCL = 0，而 PCH 仍然保持为 0，则新的程序计数器 PC 将指向错误的地址 0000H，程序出错。因此，检查跳转表是否跨越边界（xxFFH 到 xx00H）非常重要。良好的编程风格是将跳转表放在 ROM 的开始边界（如 0100H）。

➢ **例：如果跳转表跨越 ROM 边界，将引起程序错误。**

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

SONiX 提供一个宏程序以保证可靠执行跳转表功能，它将检测 ROM 边界并自动将跳转表移至适当的位置。但采用该宏程序会占用部分 ROM 空间。

➢ **例：如果跳转表跨越 ROM 边界，将引起程序错误。**

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

✷ **注：“VAL”为跳转表列表中列表个数。**

➢ **例：宏“MACRO3.H”中，“@JMP_A”的应用。**

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

如果跳转表跨越了 ROM 的边界（0FFH~100H），宏指令“@JMP_A”会调整跳转表的位置使其从 ROM 的前端开始（0100H）。

➢ **例：宏指令@JMP_A 操作。**

; 编译前

```
ROM 地址
        B0MOV     A, BUF0       ; “BUF0”从 0 至 4。
        @JMP_A    5             ; 列表个数为 5。
0X00FD  JMP       A0POINT       ; ACC = 0，跳至 A0POINT。
0X00FE  JMP       A1POINT       ; ACC = 1，跳至 A1POINT。
0X00FF  JMP       A2POINT       ; ACC = 2，跳至 A2POINT。
0X0100  JMP       A3POINT       ; ACC = 3，跳至 A3POINT。
0X0101  JMP       A4POINT       ; ACC = 4，跳至 A4POINT。
```

; 编译后

```
ROM 地址
        B0MOV     A, BUF0       ; “BUF0”从 0 至 4。
        @JMP_A    5             ; 列表个数为 5。
0X0100  JMP       A0POINT       ; ACC = 0，跳至 A0POINT。
0X0101  JMP       A1POINT       ; ACC = 1，跳至 A1POINT。
0X0102  JMP       A2POINT       ; ACC = 2，跳至 A2POINT。
0X0103  JMP       A3POINT       ; ACC = 3，跳至 A3POINT。
0X0104  JMP       A4POINT       ; ACC = 4，跳至 A4POINT。
```

2.1.2.4 CHECKSUM计算

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

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

```
                MOV     A,#END_USER_CODE$L
                B0MOV   END_ADDR1, A        ; 用户程序结束地址低位地址存入end_addr1。
                MOV     A,#END_USER_CODE$M
                B0MOV   END_ADDR2, A        ; 用户程序结束地址中间地址存入end_addr2。
                CLR     Y                   ; 清 Y。
                CLR     Z                   ; 清 Z。
@@:
                MOVC
                B0BCLR  FC                  ; 清标志位 C。
                ADD     DATA1, A            ;
                MOV     A, R
                ADC     DATA2, A            ;
                JMP     END_CHECK           ; 检查 YZ 地址是否为代码的结束地址。
AAA:
                INCMS   Z
                JMP     @B                  ; 若 Z != 00H，进行下一个计算。
                JMP     Y_ADD_1             ; 若 Z = 00H，Y+1。
END_CHECK:
                MOV     A, END_ADDR1
                CMPRS   A, Z                ; 检查 Z 地址是否为用户程序结束地址低位地址。
                JMP     AAA                 ; 否，则进行 Checksum 计算。
                MOV     A, END_ADDR2
                CMPRS   A, Y                ; 是则检查 Y 的地址是否为用户程序结束地址中间地址。
                JMP     AAA                 ; 否，则进行 Checksum 计算。
                JMP     CHECKSUM_END        ; 是则 Checksum 计算结束。

Y_ADD_1:
                INCMS   Y                   ;
                NOP
                JMP     @B                  ; 跳转到 Checksum 计算。
CHECKSUM_END:
                …
                …
END_USER_CODE:                              ; 程序结束。
```

2.1.3 编译选项列表（CODE OPTION）

编译选项	内容	功能说明
Watch_Dog	Enable	在普通模式和绿色模式下开启看门狗定时器。
	Disable	关闭看门狗定时器。
	Always On	在所有工作模式下都开启看门狗定时器。
Security	Enable	ROM 代码加密。
	Disable	ROM 代码不加密。
High_Clk_Div	Fhosc/4	High clock Fcpu = IHRC/4 = 1MHz。
	Fhosc/8	High clock Fcpu = IHRC/8 = 500kHz。
	Fhosc/16	High clock Fcpu = IHRC/16 = 250kHz。
	Fhosc/32	High clock Fcpu = IHRC/32 = 125kHz。

✷ **注 1：在高干扰环境下，强烈建议设置 Watch_Dog 为 Always_On；**
✷ **注 2：Fcpu 编译选项仅对高速时钟有效，低速模式下 Fcpu = Flosc/4。**

2.2 数据存储器RAM

☞ **RAM：256 X 8-bit**

	地址	RAM	
Bank 0	000H … … 07FH	通用存储区域	RAM Bank 0
	080H … 0FFH	系统寄存器	Bank 0 结束
Bank 1	100H … … 17FH	通用存储区域	RAM Bank 1 Bank 1 结束
Bank 15	F00H … F1FH	**LCD RAM** 区域	RAM Bank 15 .. Bank 15 结束

2.2.1 系统寄存器

2.2.1.1 系统寄存器列表

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	L	H	R	Z	Y	X	PFLAG	RBANK	-	LCDM1	LCDM2	-	-	-	-	-
9	VREG	AMPM1	AMPM2	ADCM1	ADCM2	ADCM3	LBTM	ADCDH	ADCDM	ADCDL	LEDCPM	-	-	-	-	-
A	ROMADRH	ROMADRL	ROMDAH	ROMDAL	ROMCNT	-	-	-	-	-	-	-	-	-	-	
B	-	-	-	-	-	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	P2M		-	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	P1	P2	-	-	P5	-	-	T0M	T0C	TC0M	TC0C	TC1M	TC1C	TC1R	STKP
E	P0UR	P1UR	P2UR	-	-	P5UR	@HL	@YZ	-	-	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2 系统寄存器说明

L,H = 工作寄存器，@LH 间接寻址寄存器，ROM 寻址寄存器
Y, Z = 工作寄存器，@YZ 间接寻址寄存器，ROM 寻址寄存器
PFLAG = ROM 页和特殊标志寄存器
LCDM1 = LCD 模式寄存器
AMPM1 = PGIA 模式选择寄存器
ADCM1 = ADC 控制寄存器
ADCM3 = ADC 控制寄存器
ADCDH = ADC 高字节数据缓存器
ADCDL = ADC 低字节数据缓存器
ROMADRH = ISP ROM 地址
ROMDAH = ISP ROM 烧录数据缓存器
ROMCNT = ISP ROM 计数器
PNUR = Pn 上拉电阻寄存器
P1W = P1 唤醒功能控制寄存器
INTRQ = 中断请求寄存器
OSCM = 振荡器模式寄存器
PCH, PCL = 程序计数器
T0C = T0 计数寄存器
TC0C = TC0 计数寄存器
TC1C = TC1 计数寄存器
@HL = 间接寻址寄存器
@YZ = 间接寻址寄存器
STK0~STK7 = 堆栈缓存器

R = 工作寄存器，ROM 查表数据缓存器
X = 工作寄存器，ROM 查表数据缓存器
RBANK = Bank 选择寄存器
LCDM2 = LCD 模式寄存器
AMPM2 = PGIA 模式选择寄存器
ADCM2 = ADC 控制寄存器
VREG = 电压稳压控制寄存器
LBTM = 电池低电压检测寄存器
ADCDM = ADC 中间字节数据缓存器
LEDCPM = Charge Pump LED-regulator 控制寄存器
ROMADRL = ISP ROM 地址
ROMDAL = ISP ROM 数据烧录缓存器
PNM = Pn 输入输出模式寄存器
PN = Pn 数据缓存器
INTEN = 中断使能寄存器
WDTR = 看门狗定时器计数器
TC0R = TC0 自动重装数据缓存器
T0M = T0 模式寄存器
TC0M = TC0 模式寄存器
TC1M = TC1 模式寄存器
TC1R = TC1 自动重装数据缓存器
STKP = 堆栈指针

2.2.1.3 系统寄存器的位定义

地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	名称
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
085H	XBIT7	XBIT6	XBIT5	XBIT4	XBIT3	XBIT2	XBIT1	XBIT0	R/W	X
086H	-	-	-	-	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	RBNKS3	RBNKS2	RBNKS1	RBNKS0	R/W	RBANK
089H	LCDREF1	LCDREF0	LCDBNK	LCDTYPE	LCDENB	LCDBIAS	LCDRATE	LCDCLK	R/W	LCDM1
08AH	-	BGM	LCDPENB	VPPINTL	VCP3	VCP2	VCP1	VCP0	R/W	LCDM2
090H	BGRENB	ACMSEL	ACMENB	AVESEL	AVENB	AVDDRSEL	AVDDRENB	-	R/W	VREG
091H	CHS3	CHS2	CHS1	CHS0	GS2	GS1	GS0	AMPENB	R/W	AMPM1
092H	INRENB	GX	GR	AMPCKS1	AMPCKS0	PCHPENB	DTENB	DTSEL	R/W	AMPM2
093H	RVS	IRVS3	IRVS2	IRVS1	IRVS0	ADGN1	ADGN0	ADCENB	R/W	ADCM1
094H	-	OSR2	OSR1	OSR0	-	OFSEL1	OFSEL0	DRDY	R/W	ADCM2
095H	-	-	-	ACHPENB	ADCKINV	ADCKS2	ADCKS1	ADCKS0		ADCM3
096H	-	P51IO	LBTSEL3	LBTSEL2	LBTSEL1	LBTSEL0	LBTO	LBTENB	R/W	LBTM
097H	ADCB23	ADCB22	ADCB21	ADCB20	ADCB19	ADCB18	ADCB17	ADCB16	R	ADCDH
098H	ADCB15	ADCB14	ADCB13	ADCB12	ADCB11	ADCB10	ADCB9	ADCB8	R	ADCDM
099H	ADCB7	ADCB6	ADCB5	ADCB4	ADCB3	ADCB2	ADCB1	ADCB0	R	ADCDL
09AH	-	-	-	VLEDENB	CPCKS2	CPCKS1	CPCKS0	CPRENB	R	LEDCPM
0A0H	VPPCHK	-	-	ROMADR12	ROMADR11	ROMADR10	ROMADR9	ROMADR8	R/W	ROMADRH
0A1H	ROMADR7	ROMADR6	ROMADR5	ROMADR4	ROMADR3	ROMADR2	ROMADR1	ROMADR0	R/W	ROMADRL
0A2H	ROMDA15	ROMDA14	ROMDA13	ROMDA12	ROMDA11	ROMDA10	ROMDA9	ROMDA8	R/W	ROMDAH
0A3H	ROMDA7	ROMDA6	ROMDA5	ROMDA4	ROMDA3	ROMDA2	ROMDA1	ROMDA0	R/W	ROMDAL
0A4H	-	-	-	-	-	-	ROMCNT1	ROMCNT0	W	ROMCNT
0B8H	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M	R/W	P0M
0BFH	-	-	-	-	P01G1	P01G0	P00G1	P00G0	R/W	PEDGE
0C0H	-	-	-	P14W	P13W	P12W	P11W	P10W	W	P1W
0C1H	-	-	-	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C2H	-	-	-	-	-	-	P21M	P20M	R/W	P2M
0C5H	-	-	-	P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C8H	ADCIRQ	TC1IRQ	TC0IRQ	T0IRQ	-	-	P01IRQ	P00IRQ	R/W	INTRQ
0C9H	ADCIEN	TC1IEN	TC0IEN	T0IEN	-	-	P01IEN	P00IEN	R/W	INTEN
0CAH	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-	R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH	-	-	-	PC12	PC11	PC10	PC9	PC8	R/W	PCH
0D0H	P07	P06	P05	P04	P03	P02	P01	P00	R/W	P0
0D1H	-	-	-	P14	P13	P12	P11	P10	R/W	P1
0D2H	-	-	-	-	-	-	P21	P20	R/W	P2
0D5H	-	-	-	P54	P53	P52	P51	P50	R/W	P5
0D8H	T0ENB	T0RATE2	T0RATE1	T0RATE0	TC1x8	TC0x8	TC0GN	T0TB	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0RATE2	TC0RATE1	TC0RATE0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	TC1ENB	TC1RATE2	TC1RATE1	TC1RATE0	TC1CKS	ALOAD1	TC01OUT	PWM1OUT	R/W	TC1M
0DDH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
0DEH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	W	TC1R
0DFH	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H	P07R	P06R	P05R	P04R	P03R	P02R	P01R	P00R	W	P0UR
0E1H	-	-	-	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E2H	-	-	-	-	-	-	P21R	P20R	W	P2UR
0E5H	-	-	-	P54R	P53R	P52R	P51R	P50R	W	P5UR
0E6H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H	-	-	-	S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H	-	-	-	S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H	-	-	-	S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H	-	-	-	S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H	-	-	-	S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H

0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH	-	-	-	S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH	-	-	-	S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH	-	-	-	S0PC12	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

✷ **注：**
1、为了避免系统错误，在初始化时，请将上表所有寄存器的位都按照设计要求设置为确定的"1"或者"0"；
2、所有寄存器名都已在 SN8ASM 编译器中做了宣告；
3、用户使用 SN8ASM 编译器对寄存器的位进行操作时，须在该寄存器的位前加“F”；
4、指令“b0bset”，“b0bclr”，“bset”，“bclr”只能用于可读写的寄存器（“R/W”）。

2.2.2 累加器

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

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

➢ **例：读/写 ACC。**

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

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

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

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

➢ **例：ACC 和工作寄存器中断保护。**

```
.DATA           ACCBUF      DS   1          ; 定义 ACCBUF 为 ACC 数据存储单元。
                PFLAGBUF    DS   1          ; 定义 PFLAGBUF 为 PFLAG 数据存储单元。
.CODE
INT_SERVICE:
                B0XCH       A, ACCBUF      ;
                B0MOV       A, PFLAG
                B0MOV       PFLAGBUF, A    ; 存储 ACC 和 PFLAG。
                …
                B0MOV       A, PFLAGBUF
                B0MOV       PFLAG, A       ; 恢复 PFLAG。
                B0XCH       A, ACCBUF      ; 恢复 ACC。
                RETI                       ; 退出中断。
```

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

2.2.3 程序状态寄存器PFLAG

寄存器 PFLAG 中包含 ALU 运算状态信息，位 C、DC 和 Z 显示 ALU 的运算信息。

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

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

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

Bit 0 **Z**：零标志。
1 = 算术/逻辑/分支运算的结果为零；
0 = 算术/逻辑/分支运算的结果非零。

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

2.2.4 程序计数器

程序计数器 PC 是一个 11 位二进制程序地址寄存器，分高 3 位和低 8 位。专门用来存放下一条需要执行指令的内存地址。通常，程序计数器会随程序中指令的执行自动增加。

若程序执行 CALL 和 JMP 指令时，PC 指向特定的地址。

	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	-	-	-	-	-	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
复位后	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

☞ **单地址跳转**

在 SONiX 单片机里面，有 9 条指令（CMPRS、INCS、INCMS、DECS、DECMS、BTS0、BTS1、B0BTS0 和 B0BTS1）可完成单地址跳转功能。如果这些指令执行结果为真，那么 PC 值加 2 以跳过下一条指令。

如果位测试为真，PC 加 2。

```
                B0BTS1      FC          ; 若 Carry_flag = 1 则跳过下一条指令。
                JMP         C0STEP      ; 否则执行 C0STEP。
                …
C0STEP:         NOP

                B0MOV       A, BUF0     ; BUF0 送入 ACC。
                B0BTS0      FZ          ; Zero flag = 0 则跳过下一条指令。
                JMP         C1STEP      ; 否则执行 C1STEP。
                …
                …
C1STEP:         NOP
```

如果 ACC 等于指定的立即数则 PC 值加 2，跳过下一条指令。

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

执行加 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

INCS：

```
                INCS        BUF0
                JMP         C0STEP
                …
C0STEP:         NOP
```

INCMS：

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

执行减 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

DECS：

```
                DECS        BUF0
                JMP         C0STEP
                …
C0STEP:         NOP
```

DECMS：

```
                DECMS       BUF0
                JMP         C0STEP
                …
C0STEP:         NOP
```

☞ **多地址跳转**

执行 JMP 或 ADD M,A（M=PCL）指令可实现多地址跳转。执行 ADD M, A、ADC M, A 或 B0ADD M, A 后，若 PCL 溢出，PCH 会自动进位。对于跳转表及其它应用，用户可以通过上述 3 条指令计算 PC 的值而不需要担心 PCL 溢出的问题。

✷ 注：PCH 仅支持 PC 的递增运算而不支持递减运算。当 PCL+ACC 执行完 PCL 有进位时，PCH 会自动加 1；但执行 PCL-ACC 有借位发生，PCH 的值会保持不变。

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

```
; PC = 0323H
                MOV     A, #28H
                B0MOV   PCL, A          ; 跳到地址 0328H。
                …

; PC = 0328H
                MOV     A, #00H
                B0MOV   PCL, A          ; 跳到地址 0300H。
                …
```

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

```
; PC = 0323H
                B0ADD   PCL, A          ; PCL = PCL + ACC，PCH 的值不变。
                JMP     A0POINT         ; ACC = 0，跳到 A0POINT。
                JMP     A1POINT         ; ACC = 1，跳到 A1POINT。
                JMP     A2POINT         ; ACC = 2，跳到 A2POINT。
                JMP     A3POINT         ; ACC = 3，跳到 A3POINT。
                …
                …
```

2.2.5 Y，Z寄存器

寄存器 Y 和 Z 都是 8 位缓存器，主要用途如下：

- **普通工作寄存器；**
- **RAM 数据寻址指针@YZ；**
- **配合指令 MOVC 对 ROM 数据进行查表。**

084H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Y	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	X	X	X	X	X	X	X	X

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

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

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

➢ **例：利用数据指针@YZ 对 RAM 数据清零。**

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

CLR_YZ_BUF:
        CLR     @YZ         ; @YZ 清零。

        DECMS   Z           ;
        JMP     CLR_YZ_BUF  ; 不为零。

        CLR     @YZ
END_CLR:                    ;
        …
```

2.2.6 R寄存器

8 位缓存器 R 主要有以下两个功能：

- **作为工作寄存器使用；**
- **存储执行查表指令后的高字节数据。**（执行 MOVC 指令，指定 ROM 单元的高字节数据会被存入 R 寄存器而低字节数据则存入 ACC。）

082H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	X	X	X	X	X	X	X	X

✷ **注：关于 R 寄存器查表应用请参考查表章节。**

2.3 寻址模式

2.3.1 立即寻址

将立即数送入 ACC 或指定的 RAM 单元。

- **例：立即数 12H 送入 ACC。**

```
MOV     A, #12H
```

- **例：立即数 12H 送入寄存器 R。**

```
B0MOV   R, #12H
```

✷ **注：立即数寻址中，指定的 RAM 单元必须是 80H~87H 的工作寄存器。**

2.3.2 直接寻址

通过 ACC 对 RAM 单元数据进行操作。

- **例：地址 12H 处的内容送入 ACC。**

```
B0MOV   A, 12H
```

- **例：ACC 中数据写入 RAM 的 12H 单元。**

```
B0MOV   12H, A
```

2.3.3 间接寻址

通过指针寄存器（Y/Z）访问 RAM 数据。

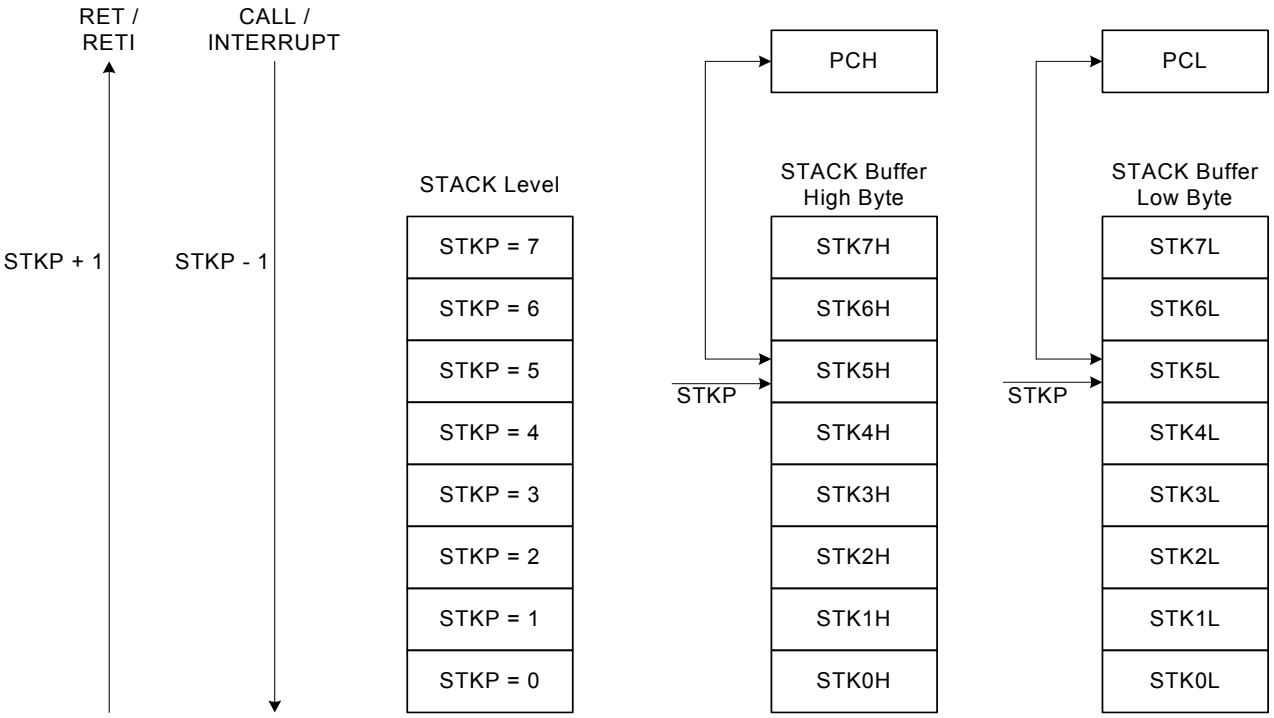
- **例：用 @YZ 实现间接寻址。**

```
B0MOV   Y, #0       ; Y 清零以寻址 RAM bank 0。
B0MOV   Z, #12H     ; 设定寄存器地址。
B0MOV   A, @YZ
```

2.4 堆栈

2.4.1 概述

SN8P2949 的堆栈缓存器共有 8 层，程序进入中断或执行 CALL 指令时，用来存储程序计数器 PC 的值。寄存器 STKP 为堆栈指针，指向堆栈缓存器顶层，STKnH 和 STKnL 分别是各堆栈缓存器高、低字节。



2.4.2 堆栈寄存器

堆栈指针 STKP 是一个 3 位寄存器，存放被访问的堆栈单元地址，13 位数据存储器 STKnH 和 STKnL 用于暂存堆栈数据。以上寄存器都位于 bank 0。

堆栈操作遵循后进先出（LIFO）的原则，入栈时堆栈指针 STKP 的值减 1，出栈时 STKP 的值加 1，这样，STKP 总是指向堆栈缓存器顶层单元。

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

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

Bit[2:0] **STKPBn**：堆栈指针（n = 0 ~ 2）。

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

➢ **例：系统复位时，堆栈指针寄存器内容为默认值，但强烈建议在程序初始部分重新设定，如下面所示：**

```
MOV     A, #01111111B
B0MOV   STKP, A
```

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnH	-	-	-	SnPC12	SnPC911	SnPC10	SnPC9	SnPC8
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnL	SnPC7	SnPC6	SnPC5	SnPC4	SnPC3	SnPC2	SnPC1	SnPC0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

STKn = STKnH , STKnL (n = 7 ~ 0)

2.4.3 堆栈操作举例

执行程序调用指令 CALL 和响应中断服务时，堆栈指针 STKP 的值减 1，指针指向下一个堆栈缓存器。同时，对程序计数器 PC 的内容进行入栈保存。

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

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

3 复位

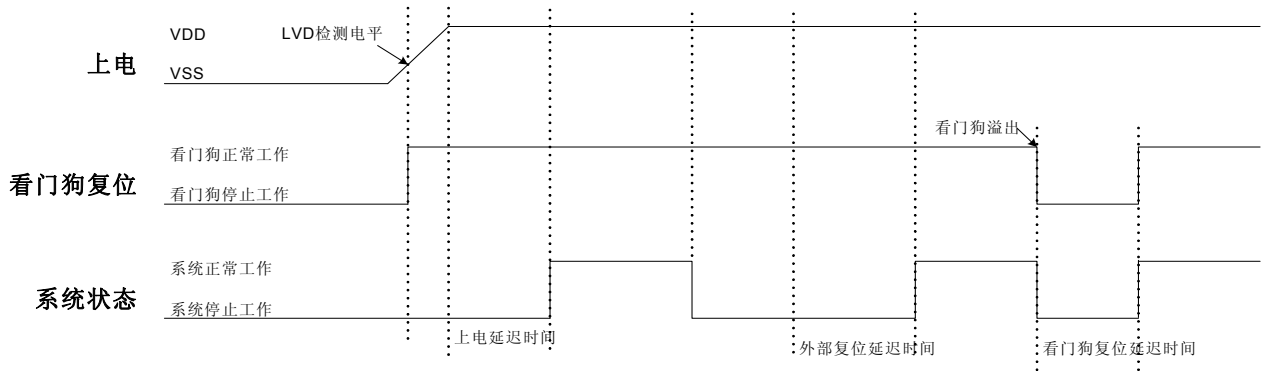
3.1 概述

SN8P2949 有以下几种复位方式：

- **上电复位；**
- **看门狗复位；**
- **掉电复位。**

上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器 PC 清零。复位结束后，系统从向量 0000H 处重新开始运行。

系统复位需要一定的时间，并提供完整的上电复位规程。对于不同类型的振荡器，完成复位所需要的时间也不同。因此，VDD 的上升速度和不同晶振的起振时间都不固定。RC 振荡器的启动时间非常短，晶体振荡器的启动时间则比较长。在用户终端使用的过程中，应注意考虑主机对上电复位时间的要求。下面给出了复位时序图。



3.2 上电复位

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

- **上电**：系统检测到电源电压上升并等待其稳定；
- **系统初始化**：所有的系统寄存器被置为默认状态；
- **振荡器开始工作**：振荡器开始提供系统时钟；
- **执行程序**：上电结束，程序开始运行。

3.3 看门狗复位

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

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

看门狗定时器运用注意事项：

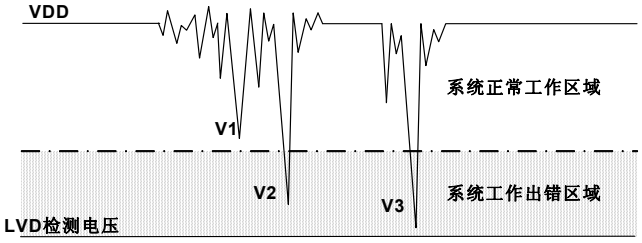
- 看门狗定时器清零之前，请检查 I/O 口的状态和 RAM 的内容可增强程序的可靠性；
- 不能在中断中对看门狗清零，否则无法侦测到主程序跑飞的状况；
- 程序中应该只在主程序中有一次清看门狗的动作，这种架构能够最大限度的发挥看门狗的保护功能。

✷ **注：关于看门狗定时器的详细内容，请参阅“看门狗定时器有关章节。**

3.4 掉电复位

3.4.1 概述

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



掉电复位电路图

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

DC 运用中：

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

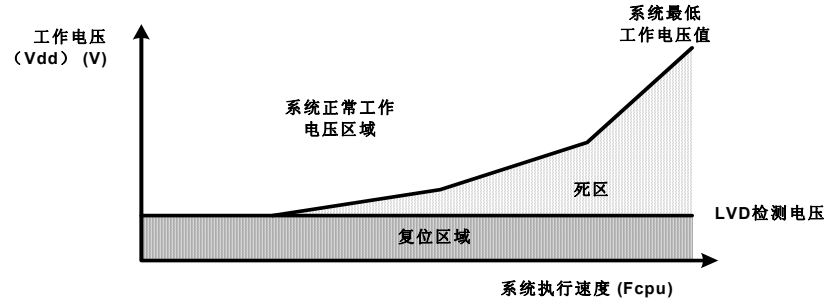
AC 运用中：

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

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

3.4.2 系统工作电压

为了改善系统掉电复位的性能，首先必须明确系统具有的最低工作电压值。系统最低工作电压与系统执行速度有关，不同的执行速度下最低工作电压亦不同。电气特性一章给出了系统工作电压与执行速度之间的关系。



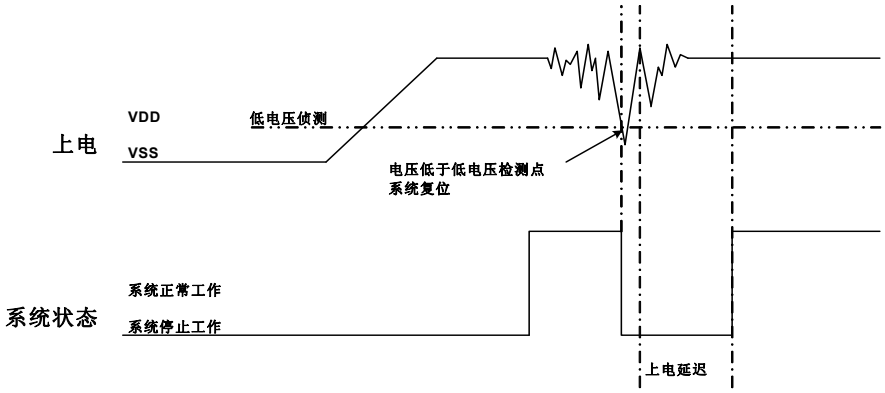
如上图所示，系统正常工作电压区域一般略高于系统复位电压，同时复位电压由 LVD 检测电平决定。当系统执行速度提高时，系统最低工作电压也相应提高。复位电压与最低工作电压之间的区域即是系统工作的死区。

3.4.3 掉电性能复位改进

如何改善系统掉电复位性能，有以下几点建议：

- **LVD 复位；**
- **看门狗复位；**
- **降低系统工作速度。**

LVD 复位：



低电压检测（LVD）是 SONiX 8 位单片机的内置掉电复位保护装置，当 VDD 跌落并低于 LVD 检测电压值时，LVD 被触发，系统复位。不同的单片机有不同的 LVD 检测电平，LVD 检测电平值仅为一个电压点，并不能覆盖所有死区范围。因此采用 LVD 依赖于系统要求和环境状况。电源变化较大时，LVD 能够起到保护作用，如果电源变化触发 LVD，系统工作仍出错，那么 LVD 就不能起到保护作用，就需要采用其它复位方法。电气特性一章中给出了更多关于 LVD 的详细内容。

看门狗复位：

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

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

降低系统工作速度：

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

4 系统时钟

4.1 概述

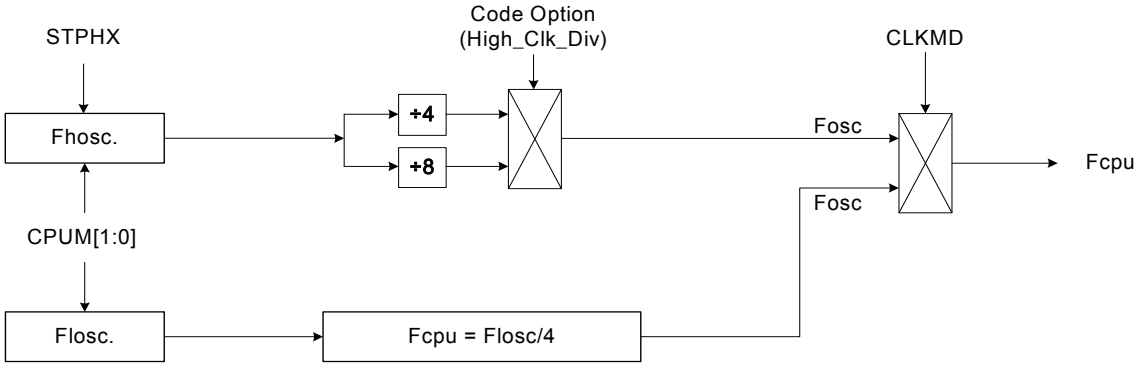
SN8P2949 内带双时钟系统：高速时钟和低速时钟。高速时钟由内置 4MHz 高速 RC 振荡电路（IHRC 4MHz）提供，低速时钟则由 RC 振荡电路或者外部 32768 晶振提供。

高低速时钟都可以作为系统时钟，当系统在低速模式下工作时，时钟信号 4 分频之后作为系统指令周期 Fcpu。

☞ **普通模式（高速时钟）：Fcpu = Fhosc / 4 = 1MHz，（Fhosc = 4M IHRC）；**
Fcpu = Fhosc / 8 = 500KHz，（Fhosc = 4M IHRC）；
Fcpu = Fhosc / 16 = 250KHz，（Fhosc = 4M IHRC）；
Fcpu = Fhosc / 32 = 125KHz，（Fhosc = 4M IHRC）.

☞ **低速模式（低速时钟）：Fcpu = Flosc/4。（Flosc = 32768Hz 或者 ILRC）。**

4.2 系统时钟框图



- Fhosc：系统内部高速 RC 时钟频率（IHRC）。
- Flosc：系统内部低速 RC 时钟频率（ILRC）或者外部 32K 时钟源。
- Fosc：系统时钟频率。
- Fcpu：指令执行频率。

4.3 OSCM寄存器

寄存器 OSCM 控制振荡器的状态和系统模式。

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

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

Bit 2 **CLKMD**：高/低速时钟模式控制位。
0 = 普通（双时钟）模式，系统时钟来自高速时钟；
1 = 低速模式，编译选项选择 IHRC 时，系统时钟为内部低速时钟；编译选项选择 IHRC_RTC 时，系统时钟为外部 32K 时钟。

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

- **例：停止高速振荡器。**

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

- **例：系统进入睡眠模式后，高低速振荡器都停止运行。**

```
B0BSET      FCPUM0
```

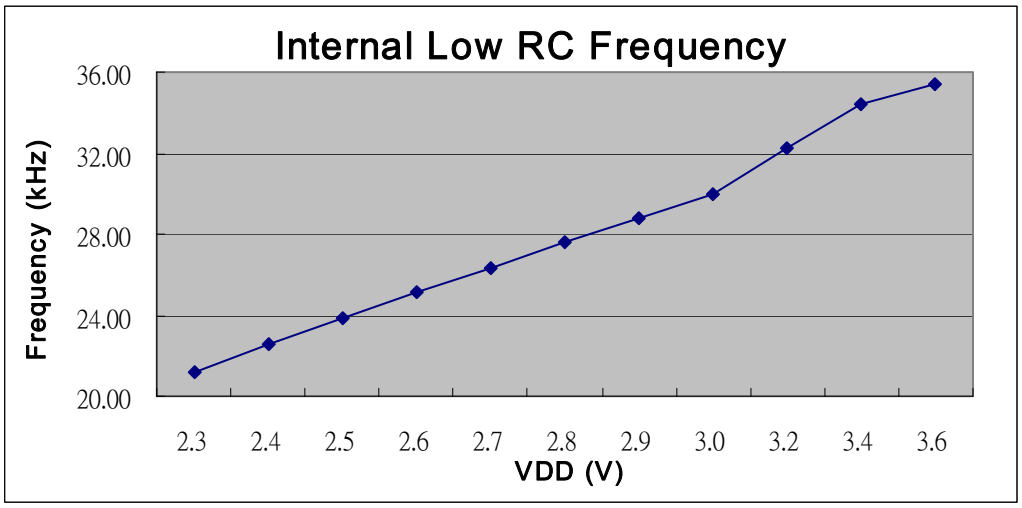
4.4 系统高速时钟

系统高速时钟仅由内部 4MHz RC 振荡电路提供（IHRC）。普通模式下，系统时钟 4 分频，8 分频，16 分频或者 32 分频（由编译选项 High_Clk_Div 控制）后作为指令周期（Fcpu）。

4.5 系统低速时钟

系统低速时钟仅由内部 RC 振荡电路提供（ILRC）。低速模式下，系统时钟 4 分频后作为指令周期（Fcpu）。

系统低速时钟源即内置的低速振荡器，采用 RC 振荡电路。低速时钟的输出频率受系统电压和环境温度的影响，通常为 3.2V 时输出 32KHz。输出频率与工作电压之间的关系如下图所示。



内部低速 RC 时钟可作为看门狗定时器的时钟源。由 CLKMD 控制系统低速工作模式。

- **Flosc = 内部低速 RC 振荡时钟（32KHz @3.2V）。**
- **低速模式 Fcpu = Flosc / 4。**

看门狗编译选项选择“Disable”或者“Enable”时，在睡眠模式下，ILRC 停止工作。看门狗编译选项选择“Always_On”，在睡眠模式下，ILRC 继续工作直至看门狗溢出复位。

➢ **例：睡眠模式下停止内部低速振荡器。**

```
B0BSET      FCPUM0
```

✷ **注：不可以单独停止内部低速时钟；由寄存器 OSCM 的位 CPUM0、CPUM1 的设置决定外部低速时钟的状态。**

4.5.1 系统时钟测试

在设计过程中，用户可通过软件指令周期 Fcpu 对系统时钟速度进行测试。这种测试方法只在 RC 模式下有效。

➢ **例：外部振荡器的 Fcpu 指令周期测试。**

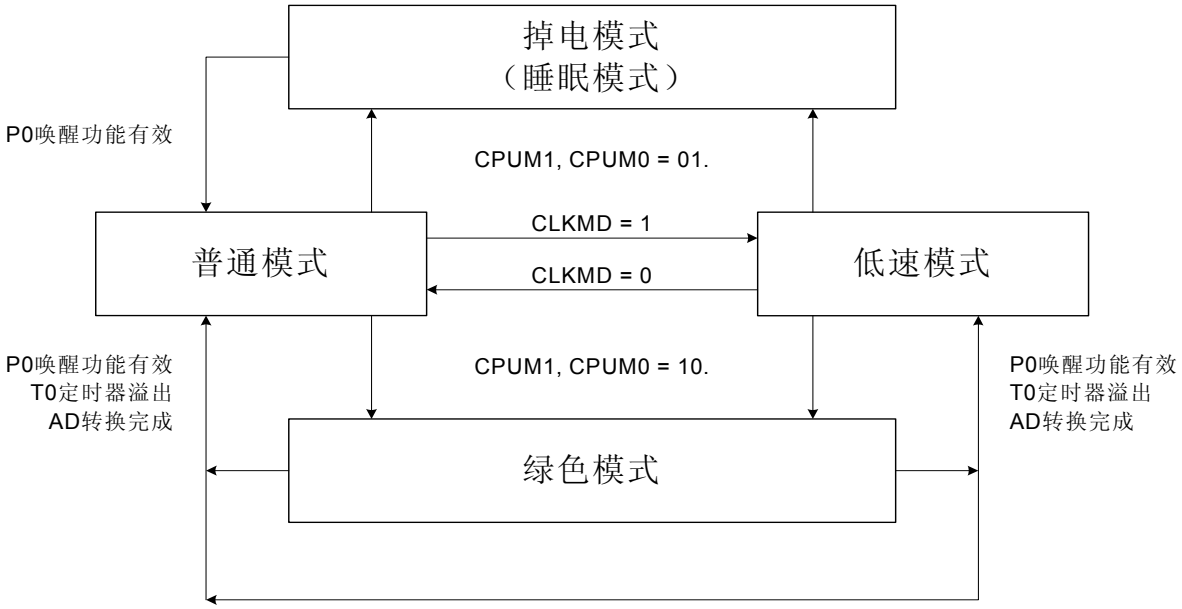
```
B0BSET      P1M.0           ; 设置 P1.0 位输出模式以输出 Fcpu 的触发信号。
@@:
B0BSET      P1.0
B0BCLR      P1.0
JMP         @B
```

5 系统工作模式

5.1 概述

SN8P2949 可以在如下 4 种工作模式之间切换：

- **高速模式；**
- **低速模式；**
- **睡眠模式；**
- **绿色模式。**



系统模式切换示意图

工作模式说明

工作模式	普通模式	低速模式	绿色模式	睡眠模式	备注
Fhosc	运行	STPHX	STPHX	停止	
Flosc	运行	运行	运行	停止	
CPU 指令	执行	执行	停止	停止	
T0 定时器	*有效	*有效	*有效	无效	* T0ENB=1 时有效
ADC	有效	*有效	*有效	停止	*高速时钟工作时有效（STPHX = 0）
看门狗定时器	Watch_Dog 编译选项	Watch_Dog 编译选项	Watch_Dog 编译选项	Watch_Dog 编译选项	参考编译选项说明
内部中断	T0, ADC	T0, *ADC	T0, *ADC	全部无效	*高速时钟工作时有效（STPHX = 0）
外部中断	P00	P00	P00	P00	
唤醒源	-	-	P0, T0, ADC	P0	*高速时钟工作时有效（STPHX = 0）

5.2 系统模式切换举例

➢ **例：系统从普通/低速模式切换进入睡眠模式。**

```
        B0BSET      FCPUM0          ;
```

✷ **注：睡眠模式下，只有具有唤醒功能的引脚和复位可以把系统唤醒进入普通模式。**

➢ **例：系统从普通模式切换进入低速模式。**

```
        B0BSET      FCLKMD          ; 设置 CLKMD = 1，系统进入低速模式。
        B0BSET      FSTPHX          ; 停止高速振荡器。
```

➢ **例：系统从低速模式切换进入普通模式（IHRC 正常运行）。**

```
        B0BCLR      FCLKMD          ;
```

➢ **例：系统从低速模式切换进入普通模式（IHRC 停止运行）。**

内部高速时钟停止后，系统返回普通模式需要等待 20ms 的延迟时间等待外部时钟电路稳定。

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

        MOV         A, #54          ; 若 VDD = 3.2V，内部 RC=32KHz（典型值）
@@:     DECMS       Z               ; 需延迟 0.125ms X 162 = 20.25ms 等待外部时钟稳定。
        JMP         @B
                                    ;
        B0BCLR      FCLKMD          ; 系统返回到普通模式。
```

➢ **例：从普通/低速模式切换进入绿色模式。**

```
        B0BSET      FCPUM1          ;
```

✷ **注：绿色模式下禁止 T0 的唤醒功能，只有具有唤醒功能的引脚和复位引脚可以将系统唤醒。**

➢ **例：从普通/低速模式切换进入绿色模式，并使能 T0 唤醒功能。**

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

; 进入绿色模式。
        B0BCLR      FCPUM0          ; 设置 CPUMx=10。
        B0BSET      FCPUM1
```

✷ **注：绿色模式下使能 T0 的唤醒功能，即具有唤醒功能的引脚、复位引脚和 T0 定时器都可以将系统从绿色模式唤醒。T0 的唤醒周期由程序控制，T0ENB 必须置 1。**

➢ **例：从普通/低速模式切换金融绿色模式，并使能 ADC 唤醒功能。**

```
; 设置 ADC 定时器的唤醒功能。
        MOV     A,#11111111b
        B0MOV   VREG,A          ; 开启所有模拟电压 regulators。
        MOV     A,#00000111b
        B0MOV   AMPM1,A         ; 设置 PGIA 功能。
        B0BSET  FADCENB         ; 使能 ADC 功能。

; 进入绿色模式，高速时钟正常工作。
        B0BSET  FCPUM1          ; 设置 CPUM0 = 1。
```

✷ **注 1：系统在绿色模式下，若使能 ADC 功能，高速时钟正常工作，则 AD 转换结束后将系统唤醒。**
✷ **注 2：若 ADCENB = 0，或者停止高速时钟（STPHX = 1），系统在进入绿色模式之前，须禁止 ADC 唤醒功能。**

5.3 系统唤醒

5.3.1 概述

系统在睡眠模式和绿色模式下时并不执行程序。唤醒触发信号可以将系统唤醒进入普通模式或低速模式。唤醒触发信号来自外部触发信号（P0 的电平变化）和内部触发信号（T0 溢出或者 AD 转换完成）。

- 从睡眠模式唤醒后只能进入普通模式，且将其唤醒的信号只能是外部触发信号（P0 电平变化）；
- 从绿色模式唤醒后返回上一个工作模式（普通模式或者低速模式），其唤醒信号为外部触发信号（P0 电平变化）和内部触发信号（T0 溢出或者 AD 转换完成）。

5.3.2 唤醒时间

系统进入睡眠模式后，高速时钟停止运行。把系统从睡眠模式下唤醒时，单片机需要等待 64 个内部高速振荡器（IHRC）时钟周期以使振荡电路进入稳定工作状态，等待的这一段时间就称为唤醒时间。唤醒时间结束后，系统才进入到普通模式。

✷ **注：从绿色模式下唤醒不需要唤醒时间，因为在绿色模式下，系统时钟仍然在工作。**

唤醒时间计算如下：

唤醒时间 = 1/Fhosc * 64 (sec) + 高速时钟启动时间

✷ **注：高速时钟的启动时间与 VDD 和振荡器类型有关。 通常情况下，VDD = 3V 时，高速时钟的启动时间为几个微秒。**

➢ **例：将系统从睡眠模式中唤醒，并设置系统进入普通模式。唤醒时间计算如下。**

唤醒时间= 1/Fhosc * 64 = 16 us(Fhosc = 4MHz)
总的唤醒时间 = 16 us + 振荡器启动时间（5us）

5.3.3 P1W唤醒控制寄存器

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

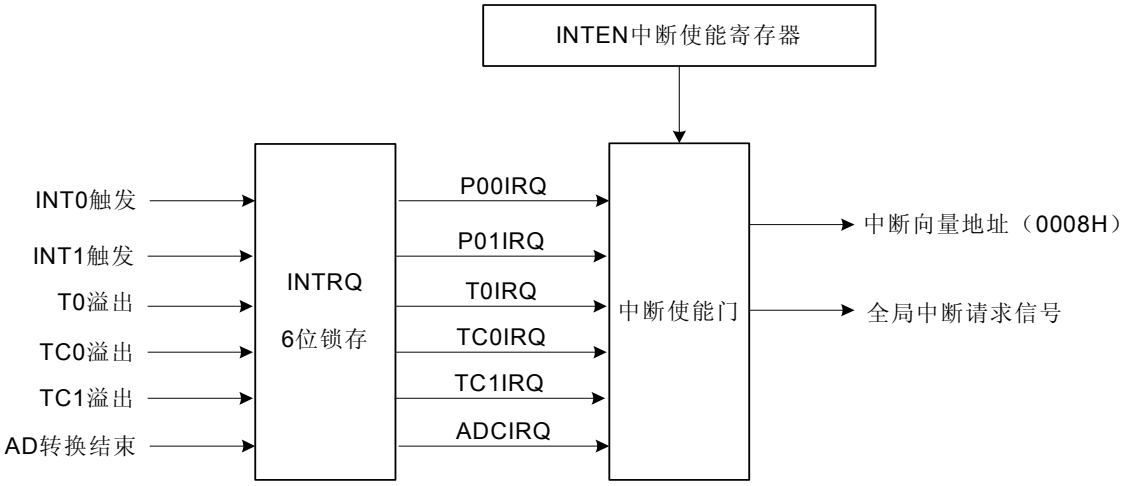
0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	-	-	-	P14W	P13W	P12W	P11W	P10W
读/写	-	-	-	W	W	W	W	W
复位后	-	-	-	0	0	0	0	0

Bit[4:0] **P10W~P14W：** P1 唤醒功能控制位。
0 =禁止 P1 的唤醒功能；
1 =开放 P1 的唤醒功能。

6 中断

6.1 概述

SN8P2949 共有 6 个中断源：4 个内部中断 T0/TC0/TC1/ADC 和 2 个外部中断 INT0/INT1。外部中断可以将系统从睡眠模式中唤醒进入高速普通模式。一旦程序进入中断，寄存器 STKP 的位 GIE 将被硬件自动清零以避免再次响应其它中断。系统退出中断后，硬件自动将 GIE 置“1”，以响应下一个中断。中断请求存放在寄存器 INTRQ 中。



✷ **注：程序响应中断时，位 GIE 必须处于有效状态。**

6.2 中断使能寄存器INTEN

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

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	ADCIEN	TC1IEN	TC0IEN	T0IEN	-	-	P01IEN	P00IEN
读/写	R/W	R/W	R/W	R/W	-	-	R/W	R/W
复位后	0	0	0	0	-	-	0	0

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

Bit1 **P01IEN**：P0.1 外部中断(INT1) 控制位。
0 = 无效；
1 = 有效。

Bit 4 **T0IEN**：T0 中断控制位。
0 = 无效；
1 = 有效。

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

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

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

6.3 中断请求寄存器INTRQ

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

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	ADCIRQ	TC1IRQ	TC0CIRQ	T0IRQ	-	-	P01IRQ	P00IRQ
读/写	R/W	R/W	R/W	R/W	-	-	R/W	R/W
复位后	0	0	0	0	-	-	0	0

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

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

Bit 4 **T0IRQ**：T0 中断请求标志。
0 = T0 无中断请求；
1 = T0 有中断请求。

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

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

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

6.4 全局中断控制位GIE

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

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

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

➢ **例：设置全局中断控制位（GIE）。**

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

✷ **注：在所有中断中，GIE 都必须处于使能状态。**

6.5 外部中断（INT0~INT1）

SN8F2949 提供 2 个外部中断 INT0/INT1。当发生外部触发，且使能外部中断控制位时，外部中断请求位置 1。如果禁止外部中断控制位，即使发生外部触发，外部中断请求位也不会请求中断。当使能外部中断控制位，且触发外部中断时，程序计数器会跳至中断向量 ORG 0008H 并执行外部中断。

外部中断内置唤醒锁存功能，即系统从睡眠模式触发唤醒时，唤醒源为外部中断源（P0.0 或者 P0.1），其唤醒触发方向与中断触发方向统一时，唤醒触发方向被锁定，即系统被唤醒后先执行外部中断。

✷ **注：INT0 中断请求由 P0.0 唤醒触发锁存。**
✷ **注：INT1 中断请求由 P0.1 唤醒触发锁存。**

✷ **注：P0.0/P0.1 中断触发放学由 PEDGE 寄存器控制。**

0BFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE	-	-	-	-	P01G1	P01G0	P00G1	P00G0
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	0	0	0	0

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

Bit[3:2] **P01G[1:0]**：P0.1 中断触发控制位。
00 = 保留；
01 = 上升沿触发；
10 = 下降沿触发；
11 = 上升/下降沿触发（电平变换触发）。

➢ **例：INT0 中断请求设置，电平触发。**

```
        MOV     A, #03H
        B0MOV   PEDGE, A        ; 设置 INT0 为电平触发。

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

➢ **例：INT0 中断。**

```
        ORG     8H              ;
        JMP     INT_SERVICE
INT_SERVICE:

        …                       ; 保存 ACC 和 PFLAG。

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

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

        RETI                    ; 退出中断。
```

6.6 多中断操作

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

中断	有效触发
P00IRQ	由 PEDGE 控制
P01IRQ	由 PEDGE 控制
T0IRQ	T0C 溢出
TC0IRQ	TC0C 溢出
TC1IRQ	TC1C 溢出
ADCIRQ	AD 转换结束

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

➢ **例：多中断条件下检测中断请求。**

```
        ORG     8H
        JMP     INT_SERVICE
INT_SERVICE:

        …                               ; 保存 ACC 和 PFLAG。

INTP00CHK:                              ; 检查是否有 INT0 中断请求。
        B0BTS1  FP00IEN                 ; 检查是否使能 INT0 中断。
        JMP     INTT0CHK                ; 跳到下一个中断。
        B0BTS0  FP00IRQ                 ; 检查是否有 INT0 中断请求。
        JMP     INTP00                  ; 进入 INT0 中断。
INTP01CHK:                              ; 检查是否有 INT1 中断请求。
        B0BTS1  FP01IEN                 ; 检查是否使能 INT1 中断。
        JMP     INTT0CHK                ; 跳到下一个中断。
        B0BTS0  FP01IRQ                 ; 检查是否有 INT1 中断请求。
        JMP     INTP01                  ; 进入 INT1 中断。
INTT0CHK:                               ; 检查是否有 T0 中断请求。
        B0BTS1  FT0IEN                  ; 检查是否使能 T0 中断。
        JMP     INTTC0CHK               ; 跳到下一个中断。
        B0BTS0  FT0IRQ                  ; 检查是否有 T0 中断请求。
        JMP     INTT0                   ; 进入 T0 中断。
INTTC0CHK:                              ; 检查是否有 TC0 中断请求。
        B0BTS1  FTC0IEN                 ; 检查是否使能 TC0 中断。
        JMP     INTTC1CHK               ; 跳到下一个中断。
        B0BTS0  FTC0IRQ                 ; 检查是否有 TC0 中断请求。
        JMP     INTTC0                  ; 进入 TC0 中断。
INTTC1CHK:                              ; 检查是否有 TC1 中断请求。
        B0BTS1  FTC1IEN                 ; 检查是否使能 TC1 中断。
        JMP     INTADCCHK               ; 跳到下一个中断。
        B0BTS0  FTC1IRQ                 ; 检查是否有 TC1 中断请求。
        JMP     INTTC1                  ; 进入 TC1 中断。
INTADCCHK:                              ; 检查是否有 ADC 中断请求。
        B0BTS1  FADCIEN                 ; 检查是否使能 ADC 中断。
        JMP     INT_EXIT                ; 退出中断。
        B0BTS0  FADCIRQ                 ; 检查是否有 ADC 中断请求。
        JMP     INTADC                  ; 进入 ADC 中断。
INT_EXIT:
        …                               ; 恢复 ACC 和 PFLAG。

        RETI                            ; 退出中断。
```

7 I/O口

7.1 I/O口模式

寄存器 PnM 控制 I/O 口的工作模式。

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

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

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2M	-	-	-	-	-	-	P21M	P20M
读/写	-	-	-	-	-	-	R/W	R/W
复位后	-	-	-	-	-	-	0	0

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	-	-	P54M	P53M	P52M	P51M	P50M
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

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

✷ 注：
1、用户可以通过位操作指令（B0BSET、B0BCLR）对 I/O 口进行编程控制；
2、P0 和 P1 均为双向 I/O 口。

➢ **例：I/O 模式选择。**

```
CLR     P0M          ; 设置所有端口为输入模式。
CLR     P1M

MOV     A, #0FFH     ; 设置所有端口为输出模式。
B0MOV   P0M,A
B0MOV   P0M, A

B0BCLR  P1M.0        ; 设置 P1.0 为输入模式。

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

7.2 I/O口上拉电阻寄存器

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	P07R	P06R	P05R	P04R	P03R	P02R	P01R	P00R
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

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

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2UR	-	-	-	-	-		P21R	P20R
读/写	-	-	-	-	-	-	W	W
复位后	-	-	-	-	-	-	0	0

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5UR	-	-	-	P54R	P53R	P52R	P51R	P50R
读/写	-	-	-	W	W	W	W	W
复位后	-	-	-	0	0	0	0	0

Bit [7:0] **Pn0R~Pn7R**：I/O 口上拉电阻控制位。

0 = 禁止；

1 = 使能。

✷ **注：PnUR 是只写寄存器。**

➢ **例：使能 I/O 口的上拉电阻。**

```
MOV     A, #0FFH        ; 使能 P1 口的上拉电阻。
B0MOV   P1UR,A
```

7.3 I/O口数据寄存器

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	P07	P06	P05	P04	P03	P02	P01	P00
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

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

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P2	-	-	-	-	-	-	P21	P20
读/写	-	-	-	-	-	-	R/W	R/W
复位后	-	-	-	-	-	-	0	0

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	-	-	-	P54	P53	P52	P51	P50
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

Bit [7:0] **Pn0~Pn7**：I/O 口数据缓存器。
0 = 输入低或者输出低；
1 = 输入高或者输出高。

- **例：从输入端口读取数据。**

```
B0MOV   A, P0     ; 读 P0 口的数据。
B0MOV   A, P1     ; 读 P1 口的数据。
```

- **例：写入数据到输出端口。**

```
MOV     A, #0FFH  ; 所有的端口写 FFH。
B0MOV   P0, A
B0MOV   P1, A
```

- **例：写入 1 位数据到输出端口。**

```
B0BSET  P1.0      ; 设置 P1.0 为 1。

B0BCLR  P1.0      ; 设置 P1.0 为 0。
```

8 定时器

8.1 看门狗定时器

看门狗定时器 WDT 是一个 4 位二进制计数器，用于监控程序的正常执行。如果由于干扰，程序进入了未知状态，看门狗定时器溢出，系统复位。看门狗的工作模式由编译选项控制，其时钟源由内部低速 RC 振荡器（32KHz @3V）提供。

看门狗溢出时间 = 16384 /内部低速振荡器周期（sec）

VDD	内部低速 RC 频率	看门狗溢出时间
3.3V	32KHz	512ms

✷ 注：
1、如果将看门狗定时器设为 Always_On，则即使在睡眠模式和绿色模式下，看门狗也保持工作状态。
2、用仿真器仿真时，必须通过宏指令@RST_WDT 清看门狗，否则仿真器的看门狗会出错。

看门狗清零的方法是对看门狗计数器清零寄存器 WDTR 写入清零控制字 5AH。

0CCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WDTR	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

➢ **例：下面是对看门狗定时器操作的示例程序，在主程序的开始清看门狗定时器。**

```
Main:
        MOV     A, #5AH         ; 清看门狗定时器。
        B0MOV   WDTR, A
        …
        CALL    SUB1
        CALL    SUB2
        …
        JMP     Main
```

➢ **例：用宏指令@RST_WDT 清看门狗定时器。**

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

看门狗定时器应用注意事项如下：

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

➢ **例：下面是对看门狗定时器操作的示例程序，在主程序的开始清看门狗定时器。**

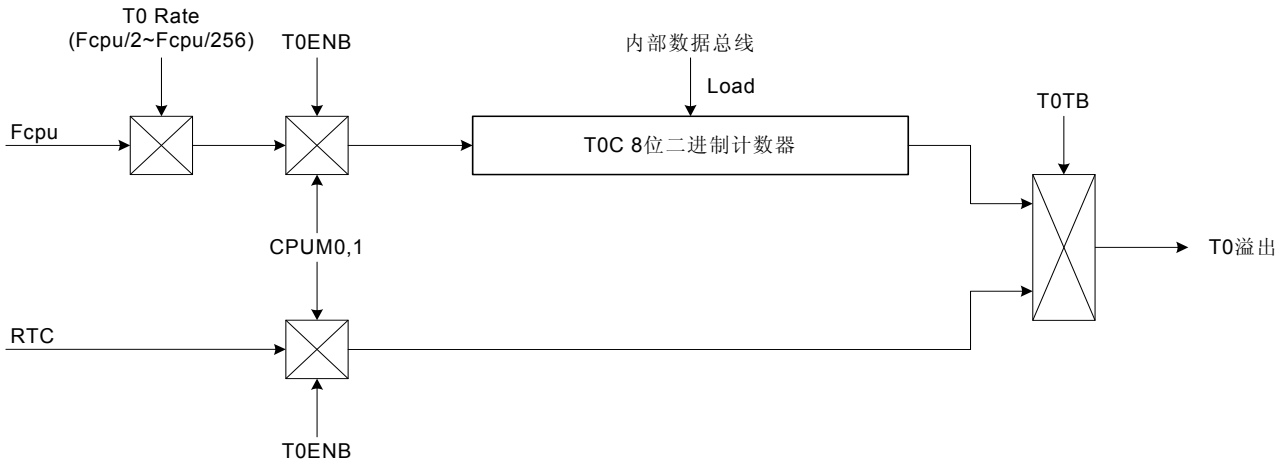
```
Main:
        …                       ; 检查 I/O 口的状态。
        …                       ; 检查 RAM 的内容。
Err:    JMP $                   ; I/O 口或 RAM 出错，不清看门狗等看门狗计时溢出。

Correct:                        ; I/O 口和 RAM 都正确，清看门狗定时器。
        B0BSET  FWDRST          ; 清看门狗定时器。
        …
        CALL    SUB1
        CALL    SUB2
        …
        JMP     Main
```

8.2 定时器T0

8.2.1 概述

二进制定时/计数器 T0 溢出（从 0FFH 到 00H）时，T0 继续计数并给出一个溢出信号触发 T0 中断。定时器 T0 的主要用途如下：

- **8 位可编程定时计数器：**根据选择的时钟频率周期性的产生中断请求；
- **RTC 定时器：**根据时钟信号在实时的产生中断请求，仅在编译选项选择 IHRC_RTC 时，RTC 功能才有效；
- **绿色模式唤醒功能：**T0ENB = 1 的条件下，T0 的溢出能够将系统从绿色模式下唤醒。



✷ 注 1：RTC 模式下，在 1/2 RTC 时钟源（32768Hz）后必须将 T0IRQ 清零，否则 RTC 间隔时间会出错。延迟时间约 16us，即使用 T0 中断程序的执行时间（约 16us）。
✷ 注 2：RTC 模式下，T0 间隔时间固定为 0.5S，T0C 计数 256 次。
✷ 注 3：在 High_Clk 编译选项选择 IHTC_RTC，且 T0TB=1 时，T0 定时器会丢失 T0IRQ，故用户必须在主程序中处理好这种情况。建议参考“T0 定时器操作流程（带有 RTC 功能）”。
✷ 注 4：“T0 定时器操作流程（带有 RTC 功能）”说明：主程序结构注意 T0C 溢出值以升级 RTC 时间。在一个周期内，主程序的间隔时间不多于 200ms。

8.2.2 T0M模式寄存器

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	TC1X8	TC0X8	TC0GN	T0TB
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

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

Bit 1 **TC0GN**：TC0 绿色模式下唤醒功能控制位。
0 = 禁止；
1 = 使能。

Bit 2 **TC0X8**：TC0 内部时钟源选择位。
0 = TC0 内部时钟源为 Fcpu，TC0RATE 可选范围 Fcpu/2~Fcpu/256；
1 = TC0 内部时钟源为 Fosc，TC0RATE 可选范围 Fosc/1~Fosc/128。

Bit 3 **TC1X8**：TC1 内部时钟源选择位。
0 = TC1 内部时钟源为 Fcpu，TC1RATE 可选范围 Fcpu/2~Fcpu/256；
1 = TC1 内部时钟源为 Fosc，TC1RATE 可选范围 Fosc/1~Fosc/128。

Bit [6:4] **T0RATE[2:0]**：T0 分频选择位。
000 = fcpu/256；
001 = fcpu/128；
… ；
110 = fcpu/4；
111 = fcpu/2。

Bit 7 **T0ENB**：T0 启动控制位。
0 =关闭；
1 =开启。

✷ **注：RTC 模式下，T0RATE 的功能被屏蔽，T0 的间隔时间固定为 0.5 sec.**

8.2.3 T0C计数寄存器

8 位计数寄存器 T0C 用于控制 T0 的中断间隔时间。

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

T0C 初始值计算公式如下：

T0C 初始值 = 256 -（T0 间隔时间 * 输入时钟）

- **例：T0 中断间隔时间置为 10ms，时钟信号 4MHz， Fcpu=Fosc/4， T0RATE=010 (Fcpu/64)。**

T0C 初始值 = 256 - (T0 间隔时间 * 时钟信号)
= 256 - (10ms * 4MHz / 4 / 64)
= 256 - (10-2 * 4 * 106 / 4 / 64)
= 100
= 64H

T0 间隔时间设置列表

T0RATE	T0CLOCK	高速模式（Fcpu = 4MHz / 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	65.536 ms	256 us	8000 ms	31250 us
001	Fcpu/128	32.768 ms	128 us	4000 ms	15625 us
010	Fcpu/64	16.384 ms	64 us	2000 ms	7812.5 us
011	Fcpu/32	8.192 ms	32 us	1000 ms	3906.25 us
100	Fcpu/16	4.096 ms	16 us	500 ms	1953.125 us
101	Fcpu/8	2.048 ms	8 us	250 ms	976.563 us
110	Fcpu/4	1.024 ms	4 us	125 ms	488.281 us
111	Fcpu/2	0.512 ms	2 us	62.5 ms	244.141 us

✷ **注：RTC 模式下，T0C 设置无效，T0 的间隔时间固定为 0.5S。**

8.2.4 T0 操作流程

T0 的操作流程举例如下：

- **T0 计时停止，禁止 T0 中断，清除中断请求标志。**

```
B0BCLR    FT0ENB      ; 关闭 T0 计数器。
B0BCLR    FT0IRQ      ; T0 中断请求标志位清零。
B0BCLR    FT0IEN      ; 禁止 T0 中断。
```

- **设置 T0 的计时速率。**

```
MOV       A, #0xxx0000b   ; T0M 的 bit4~bit6 将 T0 的速率控制在 x000xxxxb~x111xxxxb。
B0MOV     T0M,A           ; T0 定时器关闭。
```

- **设置 T0 的时钟信号（Fcpu 或 RTC）。**

```
B0BCLR    FT0TB       ; Fcpu 为时钟源。
```

或

```
B0BSET    FT0TB       ; 选择 RTC 时钟源。
```

- **设置 T0 的间隔时间。**

```
MOV       A,#7FH
B0MOV     T0C,A       ; 设置 T0C 的值。
```

- **设置 T0 定时器的模式。**

```
B0BSET    FT0IEN      ; 开启 T0 的中断功能。
```

- **开启 T0 定时器。**

```
B0BSET    FT0ENB      ; 开启 T0 定时器。
```

8.2.5 RTC操作流程（High_Clk =“IHRC_RTC” 且“T0TB = 1”）

带有 RTC 功能（High_Clk 编译选项选择 IHRC_RTC 且 T0TB=1）T0 定时器操作流程如下：

- **宣告缓存器。**

```
.DATA
        OLDT0C      DS          1
        NEWT0C      DS          1
        T0FLAG      DS          1
                    T0IRQFLAG   EQU         T0FLAG.0
```

- **T0 停止计数，禁止 T0 中断功能，清除 T0 中断请求标志。**

```
        B0BCLR      FT0ENB      ;
        B0BCLR      FT0IEN      ; 禁止 T0 中断。
        B0BCLR      FT0IRQ      ; 清除 T0 中断请求标志。
```

- **设置 T0M 寄存器。**

```
        MOV         A, #00000000b
        B0MOV       T0M,A
```

- **设置 T0 时钟源来自 RTC。**

```
        B0BSET      FT0TB       ; 选择 T0 RTC 时钟源。
```

- **设置 T0 中断间隔时间。**

```
        CLR         T0C         ; 清 T0C。
        CLR         OLDT0C      ; 清 OLDT0C。
        CLR         NEWT0C      ; 清 NEWT0C。
        CLR         T0FLAG      ; 执行主程序前清 T0FLAG。
```

- **禁止 T0 定时器功能。**

```
        B0BCLR      FT0IEN      ; 禁止 T0 中断功能。
```

- **使能带有 RTC 功能的 T0 定时器。**

```
        B0BSET      FT0ENB      ; 使能 T0 定时器。
```

- **执行主程序 Polling T0 定时器（执行主程序的间隔时间不多于 200ms）。**

```
        B0BCLR      FT0IRQ      ; 清 FT0IRQ。
MAIN:
        CALL        CKT_T0CVAL  ; 检测 T0C 是否溢出。
        CALL        CKT_T0FLAG  ; 检测 T0C 溢出标志和升级时间。
        .
        .
        JMP         MAIN
```

- **CKT_T0CVAL 子程序（检查 T0C 状态）。**

```
CKT_T0CVAL:
        MOV         A, T0C
        MOV         NEWT0C, A   ; 保存 NEWT0C。
        SUB         A, OLDT0C   ;
        B0BTS0      FC          ; 若 FC = 0，借位。
        JMP         EXIT_CKTT0CVAL: ; 若 FC = 1，跳至 EXIT_CKTT0CVAL。

        B0BSET      T0IRQFLAG   ; 设置 T0IRQFLAG（T0C 溢出）。
EXIT_CKTT0CVAL:
        MOV         A, NEWT0C
        MOV         OLDT0C, A   ; 升级 T0C。
        RET                     ;
```

● **CKT_T0FLAG 子程序（检查 T0 定时器溢出标志）。**

```
CKT_T0FLAG:
                B0BTS1      T0IRQFLAG           ; 检测 T0IRQ 的状态。
                JMP         EXIT_CKTT0FLAG      ; 跳至 EXIT_CKTT0FLAG。

                B0BCLR      T0IRQFLAG           ; 清 T0IRQFLAG。
                CALL        DELAY               ; 延迟时间大于 1/32.768ms（针对 RTC 限制）。
                B0BCLR      FT0IRQ              ; 清 FT0IRQ。
                .
                CALL        UPDATE_TIME         ; 更新时间。
                .
EXIT_CKTT0FLAG:
                RET                             ;
```

● **进入绿色模式之前。**

```
                CALL        CKT_T0CVAL          ; 检测 T0C 是否溢出。
                CALL        CKT_T0FLAG          ; 检测 T0C 溢出标志和设计时间。
```

● **唤醒后处理绿色模式。**

```
INTO_GREENMODE: .
                B0BCLR      FCPUM0
                B0BSET      FCPUM1              ; 进入绿色模式。

WAKEUP:
                B0BTS1      FT0IRQ              ; 检测 FT0IRQ。
                JMP         CKT_OTHER           ; 检测其他的触发唤醒源。

                CALL        DELAY               ; 延迟时间大于 1/32.768ms（针对 RTC 限制）。

                CLR         T0FLAG              ; 清 T0FLAG。
                B0BCLR      FT0IRQ              ; 清 FT0IRQ。

                MOV         A, T0C
                MOV         OLDT0C, A           ; 升级 T0C。

                CALL        UPDATE_TIME         ; 设计时间。
                .
CKT_OTHER:
                .
                .
```

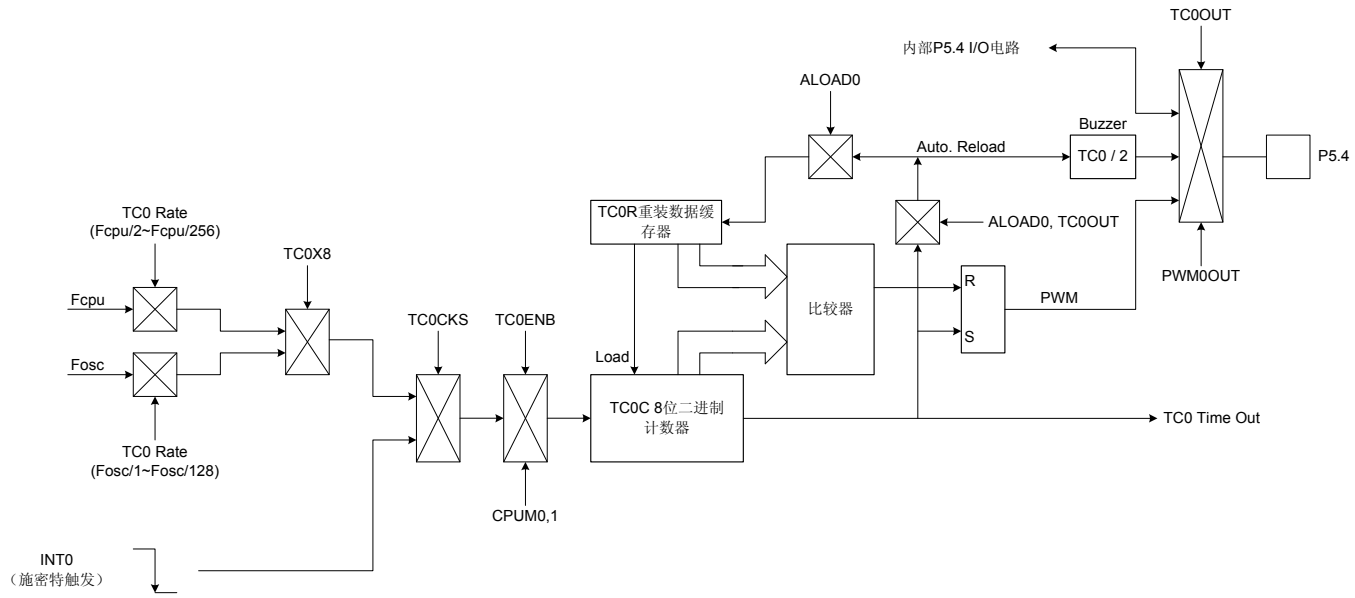
8.3 定时/计数器TC0

8.3.1 概述

定时/计数器 TC0 有 2 个时钟源，可根据实际需要选择内部时钟或外部时钟作为计时标准。其中，内部时钟源来自 Fcpu 或 Fosc（Fosc 由 TC0X8 标志控制）。外部时钟源 INT0 从 P0.0 端输入（下降沿触发）。寄存器 TC0M 控制 TC0 时钟源的选择。当 TC0 从 0FFH 溢出到 00H 时，TC0 在继续计数的同时产生一个溢出信号，触发 TC0 中断请求。在 PWM 模式，TC0 的溢出时间由 ALOAD0 和 TC0OUT 位控制。

TC0 的主要功能如下：

- **8 位可编程定时器：**根据选择的时钟频率信号，产生周期中断；
- **外部事件计数器：**对外部事件计数；
- **绿色模式唤醒功能：**TC0 可以将系统从绿色模式下唤醒；
- **Buzzer 输出；**
- **PWM 输出。**



8.3.2 TC0M模式寄存器

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 0 **PWM0OUT**：PWM 输出控制。
0 = 禁止 PWM 输出；
1 = 使能 PWM 输出，PWM 输出占空比由 TC0OUT 和 ALOAD0 控制。

Bit 1 **TC0OUT**：TC0 溢出信号输出控制位。仅当 PWM0OUT = 0 时有效。
0 = 禁止，P5.4 作为输入/输出口；
1 = 允许，P5.4 输出 TC0OUT 信号。

Bit 2 **ALOAD0**：自动装载控制位。仅当 PWM0OUT = 0 时有效。
0 = 禁止 TC0 自动重装；
1 = 允许 TC0 自动重装。

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

Bit [6:4] **TC0RATE[2:0]**：TC0 分频选择位。

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

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

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

8.3.3 TC1X8, TC0X8, TC0GN标志

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	TC1X8	TC0X8	TC0GN	T0TB
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

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

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

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

Bit 3 **TC1X8**：TC1 内部时钟选择控制位。
0 = TC1 内部时钟来自 Fcpu，TC0RATE = Fcpu/2~Fcpu/256；
1 = TC1 内部时钟来自 Fosc，TC0RATE = Fosc/1~Fosc/128。

Bit [6:4] **T0RATE[2:0]**：T0 分频选择位。
000 = fcpu/256；001 = fcpu/128； … ； 110 = fcpu/4；111 = fcpu/2。

Bit 7 **T0ENB**：T0 启动控制位。
0 =关闭；
1 =开启。

✷ **注：TC0CKS = 1 时，TC0X8 和 TC0RATE 可以忽略不计。**

8.3.4 TC0C计数寄存器

TC0C 控制 TC0 的时间间隔。

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

TC0C 初始值的计算公式如下：

TC0C 初始值 = 256 -（TC0 中断间隔时间 * 输入时钟）

TC0X8	TC0C 有效值	TC0C 有效值（二进制）	备注
0 (Fcpu/2~Fcpu/256)	00H~0FFH	00000000b~11111111b	每计数 256 次溢出
1 (Fosc/1~Fosc/128)	00H~0FFH	00000000b~11111111b	每计数 256 次溢出

➢ **例：TC0 的间隔时间为 10ms，时钟源来自 Fcpu（TC0CKS = 0，TC0X8 = 0），无 PWM 输出（PWM0 = 0），高速时钟 = 4MHz，Fcpu=Fosc/4，TC0RATE=010 (Fcpu/64)。**

TC0C 初始值 = N -（TC0 中断间隔时间 * 输入时钟）
= 256 -（10ms * 4MHz / 4 / 64）
= 256 -（10-2 * 4 * 106 / 4 / 64）
= 100
= 64H

TC0 中断间隔时间列表（TC0X8 = 0）

TC0RATE	TC0CLOCK	高速模式（Fcpu = 4MHz / 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	65.536 ms	256 us	8000 ms	31250 us
001	Fcpu/128	32.768 ms	128 us	4000 ms	15625 us
010	Fcpu/64	16.384 ms	64 us	2000 ms	7812.5 us
011	Fcpu/32	8.192 ms	32 us	1000 ms	3906.25 us
100	Fcpu/16	4.096 ms	16 us	500 ms	1953.125 us
101	Fcpu/8	2.048 ms	8 us	250 ms	976.563 us
110	Fcpu/4	1.024 ms	4 us	125 ms	488.281 us
111	Fcpu/2	0.512 ms	2 us	62.5 ms	244.141 us

TC0 中断间隔时间列表（TC0X8 = 1）

TC0RATE	TC0CLOCK	高速模式（Fcpu = 4MHz / 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	Fosc/128	8.192 ms	32 us	1000 ms	3906.25 us
001	Fosc/64	4.096 ms	16 us	500 ms	1953.125 us
010	Fosc/32	2.048 ms	8 us	250 ms	976.563 us
011	Fosc/16	1.024 ms	4 us	125 ms	488.281 us
100	Fosc/8	0.512 ms	2 us	62.5 ms	244.141 us
101	Fosc/4	0.256 ms	1 us	31.25 ms	122.07 us
110	Fosc/2	0.128 ms	0.5 us	15.625 ms	61.035 us
111	Fosc/1	0.064 ms	0.25 us	7.813 ms	30.517us

8.3.5 TC0R自动装载寄存器

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

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

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

TC0R 初始值计算公式如下：

TC0R 初始值 = N -（TC0 中断间隔时间 * 输入时钟）

N 是 TC0 最大溢出值。TC0 的溢出时间和有效值见下表：

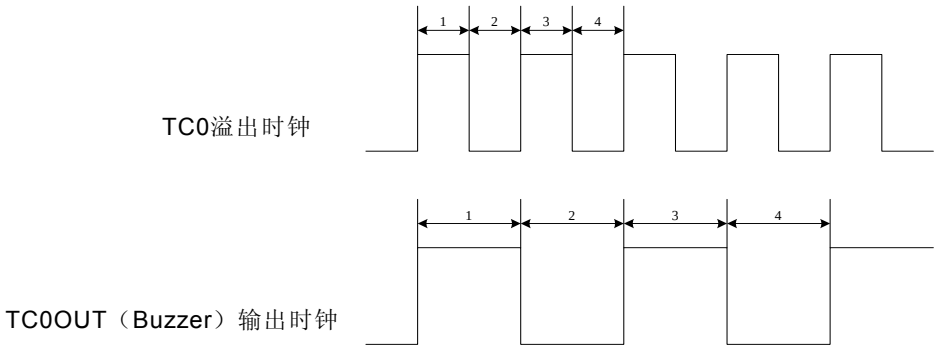
TC0X8	TC0C 有效值	TC0C 有效值（二进制）
0 (Fcpu/2~Fcpu/256)	00H~0FFH	00000000b~11111111b
1 (Fosc/1~Fosc/128)	00H~0FFH	00000000b~11111111b

➢ **例：TC0 中断间隔时间设置为 10ms，时钟源选 Fcpu（TC0KS=0，TC0X8 = 0），无 PWM 输出（PWM0=0），高速时钟为外部 4MHz，Fcpu=Fosc/4，TC0RATE=010（Fcpu/64）。**

TC0R = N -（TC0 中断间隔时间 * 输入时钟）
= 256 -（10ms * 4MHz / 4 / 64）
= 256 -（10-2 * 4 * 106 / 4 / 64）
= 100
= 64H

8.3.6 TC0 时钟频率输出（Buzzer）

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



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

➢ **例：设置 TC0OUT（P5.4）。**

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

MOV     A,#6            ; 自动装载参考值设置。
B0MOV   TC0C,A
B0MOV   TC0R,A

B0BSET  FTC0OUT         ; TC0 的输出信号由 P5.4 输出，禁止 P5.4 的普通 I/O 功能。
B0BSET  FALOAD0         ; 允许 TC0 自动重装功能。
B0BSET  FTC0ENB         ; 开启 TC0 定时器。
```

✷ **注：蜂鸣器的输出有效时，“PWM0OUT”必须被置为“0”。**

8.3.7 TC0 操作流程

TC0 定时器可用于定时器中断、事件计数、TC0OUT 和 PWM。下面分别举例说明。

- **停止 TC0 计数，禁止 TC0 中断，并清 TC0 中断请求标志。**

```
B0BCLR    FTC0ENB     ; 停止 TC0 计数、TC0OUT 和 PWM。
B0BCLR    FTC0IEN     ; 禁止 TC0 中断。
B0BCLR    FTC0IRQ     ; 清 TC0 中断请求标志。
```

- **设置 TC0 的速率 (不包含事件计数模式)。**

```
MOV       A, #0xxx0000b   ; TC0M 的 bit4~bit6 控制 TC0 的速率为 x000xxxxb~x111xxxxb。
B0MOV     TC0M,A          ; 禁止 TC0 中断。
```

- **设置 TC0 的时钟源。**

```
; 选择 TC0 内部/外部时钟源。
B0BCLR    FTC0CKS     ; 内部时钟。
或
B0BSET    FTC0CKS     ; 外部时钟。

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

✷ **注：在 TC0 外部时钟模式下，TC0X8 可以忽略不计。**

- **设置 TC0 的自动装载模式。**

```
B0BCLR    FALOAD0     ; 禁止 TC0 自动装载功能。
或
B0BSET    FALOAD0     ; 使能 TC0 自动装载功能。
```

- **设置 TC0 中断间隔时间，TC0OUT（Buzzer）频率或 PWM 占空比。**

```
; 设置 TC0 中断间隔时间，TC0OUT（Buzzer）频率或 PWM 占空比。
MOV       A,#7FH      ; TC0 的模式决定 TC0C 和 TC0R 的值。
B0MOV     TC0C,A      ; 设置 TC0C 的值。
B0MOV     TC0R,A      ; 在自动装载模式或 PWM 模式下设置 TC0R 的值。

; PWM 模式下设置 PWM 的周期。
B0BCLR    FALOAD0     ; ALOAD0，TC0OUT = 00，PWM 周期 = 0~255。
B0BCLR    FTC0OUT
或
B0BCLR    FALOAD0     ; ALOAD0，TC0OUT = 01，PWM 周期 = 0~63。
B0BSET    FTC0OUT
或
B0BSET    FALOAD0     ; ALOAD0，TC0OUT = 10，PWM 周期 = 0~31。
B0BCLR    FTC0OUT
或
B0BSET    FALOAD0     ; ALOAD0，TC0OUT = 11，PWM 周期 = 0~15。
B0BSET    FTC0OUT
```

- **设置 TC0 的模式。**

```
B0BSET    FTC0IEN     ; 使能 TC0 中断。
或
B0BSET    FTC0OUT     ; 使能 TC0OUT（Buzzer）功能。
或
B0BSET    FPWM0OUT    ; 使能 PWM。
或
B0BSET    FTC0GN      ; 使能 TC0 的绿色模式下的唤醒功能。
```

- **开启 TC0 定时器。**

```
B0BSET    FTC0ENB
```

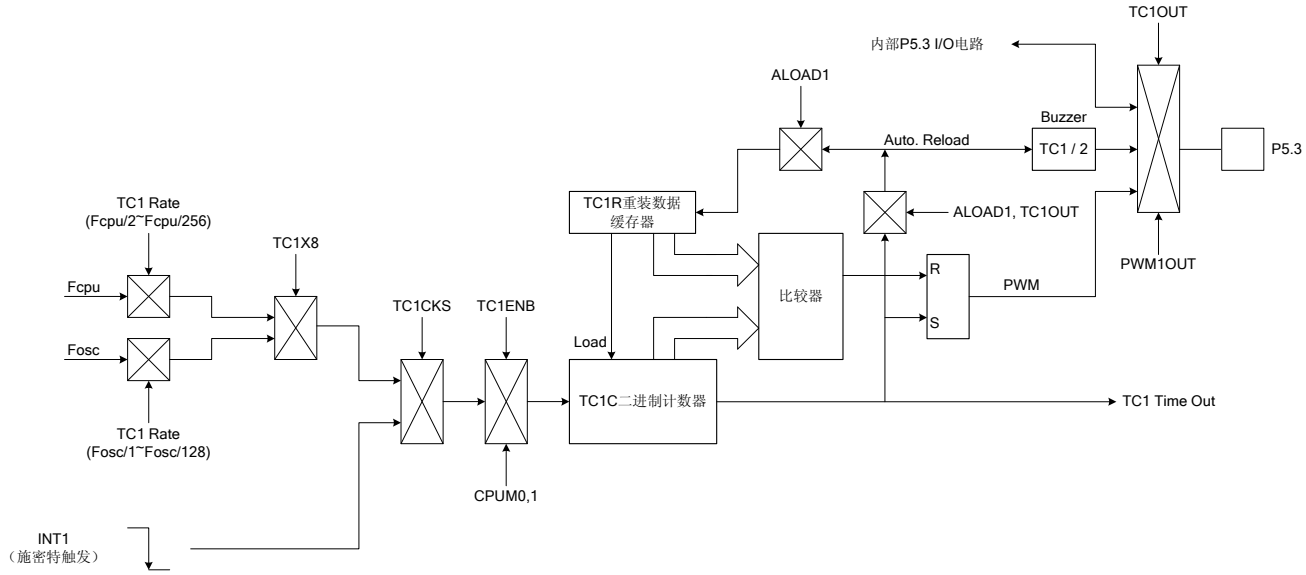
8.4 定时/计数器TC1

8.4.1 概述

定时/计数器 TC1 有 2 个时钟源，可根据实际需要选择内部时钟或外部时钟作为计时标准。其中，内部时钟源来自 Fcpu 或 Fosc（Fosc 由 TC1X8 标志控制）。外部时钟源 INT1 从 P0.1 端输入（下降沿触发）。寄存器 TC1M 控制 TC1 时钟源的选择。当 TC1 从 0FFH 溢出到 00H 时，TC1 在继续计数的同时产生一个溢出信号，触发 TC1 中断请求。在 PWM 模式，TC1 的溢出时间由 ALOAD1 和 TC1OUT 位控制。

TC1 的主要功能如下：

- **8 位可编程定时器：**根据选择的时钟信号，产生周期中断；
- **外部事件计数器：**对外部事件计数；
- **Buzzer 输出；**
- **PWM 输出。**



8.4.2 TC1M模式寄存器

0DCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC1M	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD1	TC1OUT	PWM1OUT
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 0 **PWM1OUT**：PWM 输出控制。
0 = 禁止 PWM 输出；
1 = 使能 PWM 输出，PWM 输出占空比由 TC1OUT 和 ALOAD1 控制。

Bit 1 **TC1OUT**：TC1 溢出信号输出控制位。仅当 PWM1OUT = 0 时有效。
0 = 禁止，P5.3 作为输入/输出口；
1 = 允许，P5.3 输出 TC1OUT 信号。

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

Bit 3 **TC1CKS**：TC1 时钟信号控制位。
0 = 内部时钟（Fcpu 或 Fosc）；
1 = 外部时钟，由 P0.1/INT1 输入。

Bit [6:4] **TC1RATE[2:0]**：TC1 分频选择位。

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

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

✷ **注：若 TC1CKS=1，则 TC1 用作外部事件计数器，此时不需要考虑 TC1RATE 的设置，P0.1 无中断请求（P01IRQ=0）。**

8.4.3 TC1X8, TC0X8, TC0GN标志

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	TC1X8	TC0X8	TC0GN	T0TB
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

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

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

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

Bit 3 **TC1X8**：TC1 内部时钟选择控制位。
0 = TC1 内部时钟来自 Fcpu，TC0RATE = Fcpu/2~Fcpu/256；
1 = TC1 内部时钟来自 Fosc，TC0RATE = Fosc/1~Fosc/128。

Bit [6:4] **T0RATE[2:0]**：T0 分频选择位。
000 = fcpu/256；001 = fcpu/128； … ； 110 = fcpu/4；111 = fcpu/2。

Bit 7 **T0ENB**：T0 启动控制位。
0 =关闭；
1 =开启。

✷ **注：TC1CKS = 1 时，TC1X8 和 TC1RATE 可以忽略不计。**

8.4.4 TC1C计数寄存器

TC1C 控制 TC1 的时间间隔。

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

TC1C 初始值的计算公式如下：

TC1C 初始值 = 256 -（TC1 中断间隔时间 * 输入时钟）

TC1X8	TC1C 有效值	TC1C 二进制计数范围	备注
0 (Fcpu/2~Fcpu/256)	00H~0FFH	00000000b~11111111b	每计数 256 次溢出
1 (Fosc/1~Fosc/128)	00H~0FFH	00000000b~11111111b	每计数 256 次溢出

➢ **例：TC1 的间隔时间为 10ms，时钟源来自 Fcpu（TC1CKS = 0，TC1X8 = 0），无 PWM 输出（PWM1 = 0），高速时钟 = 4MHz，Fcpu=Fosc/4，TC1RATE=010（Fcpu/64）。**

TC1C 初始值 = N -（TC1 中断间隔时间 * 输入时钟）
= 256 -（10ms * 4MHz / 4 / 64）
= 256 -（10-2 * 4 * 106 / 4 / 64）
= 100
= 64H

TC1 中断间隔时间列表（TC1X8=0）

TC1RATE	TC1CLOCK	高速模式（Fcpu = 4MHz / 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	65.536 ms	256 us	8000 ms	31250 us
001	Fcpu/128	32.768 ms	128 us	4000 ms	15625 us
010	Fcpu/64	16.384 ms	64 us	2000 ms	7812.5 us
011	Fcpu/32	8.192 ms	32 us	1000 ms	3906.25 us
100	Fcpu/16	4.096 ms	16 us	500 ms	1953.125 us
101	Fcpu/8	2.048 ms	8 us	250 ms	976.563 us
110	Fcpu/4	1.024 ms	4 us	125 ms	488.281 us
111	Fcpu/2	0.512 ms	2 us	62.5 ms	244.141 us

TC1 中断间隔时间列表（TC1X8=1）

TC1RATE	TC1CLOCK	高速模式（Fcpu = 4MHz / 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	Fosc/128	8.192 ms	32 us	1000 ms	3906.25 us
001	Fosc/64	4.096 ms	16 us	500 ms	1953.125 us
010	Fosc/32	2.048 ms	8 us	250 ms	976.563 us
011	Fosc/16	1.024 ms	4 us	125 ms	488.281 us
100	Fosc/8	0.512 ms	2 us	62.5 ms	244.141 us
101	Fosc/4	0.256 ms	1 us	31.25 ms	122.07 us
110	Fosc/2	0.128 ms	0.5 us	15.625 ms	61.035 us
111	Fosc/1	0.064 ms	0.25 us	7.813 ms	30.517us

8.4.5 TC1R自动装载寄存器

TC1 的自动装载功能由 TC1M 的 ALOAD1 位控制。当 TC1C 溢出时，TC1R 的值自动装入 TC1C 中。这样，用户在使用的过程中就不需要在中断中复位 TC1C。

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

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

TC1R 初始值计算公式如下：

TC1R 初始值 = N -（TC1 中断间隔时间 * 输入时钟）

N 是 TC1 最大溢出值。TC1 的溢出时间和有效值见下表：

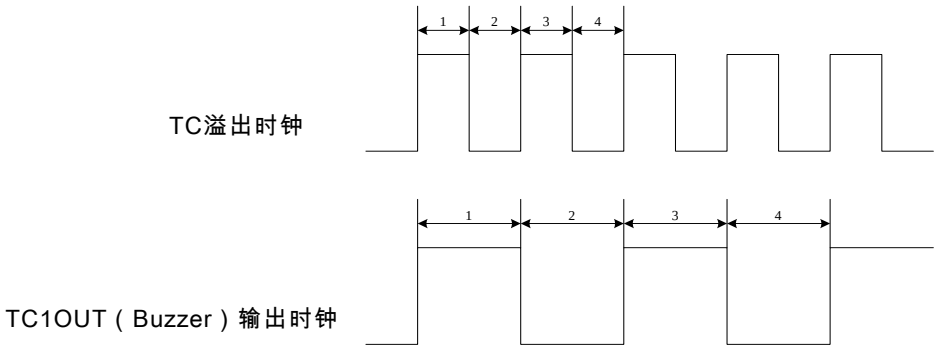
TC1X8	TC1C 有效值	TC1C 二进制计数范围
0 (Fcpu/2~Fcpu/256)	00H~0FFH	00000000b~11111111b
1 (Fosc/1~Fosc/128)	00H~0FFH	00000000b~11111111b

➢ **例：TC1 中断间隔时间设置为 10ms，时钟源选 Fcpu（TC1CKS=0，TC1X8 = 0），无 PWM 输出（PWM1=0），高速时钟为外部 4MHz，Fcpu=Fosc/4，TC1RATE=010（Fcpu/64）。**

TC1R 有效值 = N -（TC1 中断间隔时间 * 输入时钟）
= 256 -（10ms * 4MHz / 4 / 64）
= 256 -（10-2 * 4 * 106 / 4 / 64）
= 100
= 64H

8.4.6 TC1 时钟频率输出（Buzzer）

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



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

➢ **例：设置 TC1OUT（P5.3）。**

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

MOV       A,#6            ; 自动装载参考值设置。
B0MOV     TC1C,A
B0MOV     TC1R,A

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

✷ **注：蜂鸣器的输出有效时，“PWM1OUT”必须被置为“0”。**

8.4.7 TC1 操作流程

TC1 定时器可用于定时器中断、事件计数、TC1OUT 和 PWM。下面分别举例说明。

- **停止 TC1 计数，禁止 TC1 中断并清 TC1 中断请求标志。**

```
B0BCLR    FTC1ENB     ; 停止 TC1 计数、TC1OUT 和 PWM。
B0BCLR    FTC1IEN     ; 禁止 TC1 中断。
B0BCLR    FTC1IRQ     ; 清 TC1 中断请求标志。
```

- **设置 TC1 的速率 (不包含事件计数模式)。**

```
MOV       A, #0xxx0000b   ;TC1M 的 bit4~bit6 控制 TC1 的速率在 x000xxxxb~x111xxxxb。
B0MOV     TC1M,A          ; 禁止 TC1 中断。
```

- **设置 TC1 的时钟源。**

```
; 选择 TC1 内部/外部时钟源。
B0BCLR    FTC1CKS     ; 内部时钟。
或
B0BSET    FTC1CKS     ; 外部时钟。

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

✷ **注：在 TC1 外部时钟模式下，TC1X8 可以忽略不计。**

- **设置 TC1 的自动装载模式。**

```
B0BCLR    FALOAD1     ; 禁止 TC1 自动装载功能。
或
B0BSET    FALOAD1     ; 使能 TC1 自动装载功能。
```

- **设置 TC1 中断间隔时间，TC1OUT（Buzzer）频率或 PWM 占空比。**

```
; 设置 TC1 中断间隔时间，TC1OUT（Buzzer）频率或 PWM 占空比。
MOV       A,#7FH      ; TC1 的模式决定 TC1C 和 TC1R 的值。
B0MOV     TC1C,A      ; 设置 TC1C 的值。
B0MOV     TC1R,A      ; 在自动装载模式或 PWM 模式下设置 TC1R 的值。

; PWM 模式下设置 PWM 周期。
B0BCLR    FALOAD1     ; ALOAD1，TC1OUT = 00，PWM 周期 = 0~255。
B0BCLR    FTC1OUT
或
B0BCLR    FALOAD1     ; ALOAD1，TC1OUT = 01，PWM 周期 = 0~63。
B0BSET    FTC1OUT
或
B0BSET    FALOAD1     ; ALOAD1，TC1OUT = 10，PWM 周期 = 0~31。
B0BCLR    FTC1OUT
或
B0BSET    FALOAD1     ; ALOAD1，TC1OUT = 11，PWM 周期 = 0~15。
B0BSET    FTC1OUT
```

- **设置 TC1 的模式。**

```
B0BSET    FTC1IEN     ; 使能 TC1 中断。
或
B0BSET    FTC1OUT     ; 使能 TC1OUT（Buzzer）功能。
或
B0BSET    FPWM1OUT    ; 使能 PWM。
```

- **开启 TC1 定时器。**

```
B0BSET    FTC1ENB     ;
```

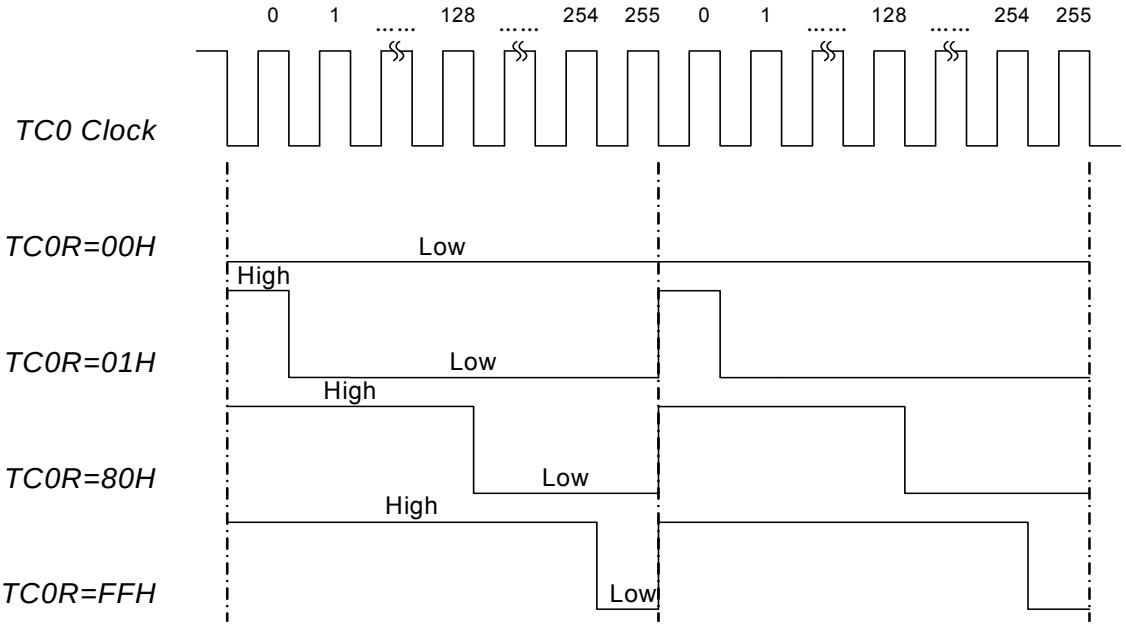
8.5 PWM0

8.5.1 概述

PWM 信号通过 PWM0OUT（P5.4 引脚）输出（256 阶）。8 位计数器 TC0C 计数过程中不断与 TC0R 相比较，当 TC0C 的值增加到与 TC0R 相等时，PWM 输出低电平，当 TC0C 的值溢出重新回到 0 时，PWM 被强制输出高电平。PWM0 输出占空比 = TC0R/ 256。

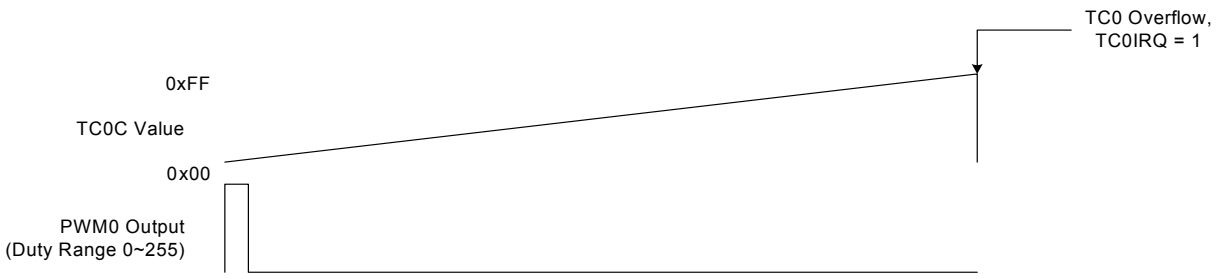
PWM 占空比范围	TC0C 有效值	TC0R 有效值	MAX. PWM 频率（Fcpu = 4MHz）	备注
0/256~255/256	00H~0FFH	00H~0FFH	7.8125K	每计数 256 次溢出

PWM 输出占空比随 TC0R 的变化而变化：0/256~255/256。



8.5.2 TC0IRQ和PWM0 输出占空比

在 PWM 模式下，TC0IRQ 的频率与 PWM 的占空比有关，具体情况如下图所示：



8.5.3 PWM0 编程举例

➢ **例：PWM0 输出设置。外部高速振荡器输出频率= 4MHZ，Fcpu = Fosc/4，PWM0 输出占空比= 30/256，输出频率 1KHZ，PWM 时钟源来自外部时钟，TC0 速率= Fcpu/4，TC0RATE2~TC0RATE0 = 110，TC0C = TC0R = 30。**

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

MOV     A,#30           ; PWM 输出占空比＝30/256。
B0MOV   TC0C,A
B0MOV   TC0R,A

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

✷ **注：TC0R 为只写寄存器，不能用 INCMS 和 DECMS 指令对其进行操作。**

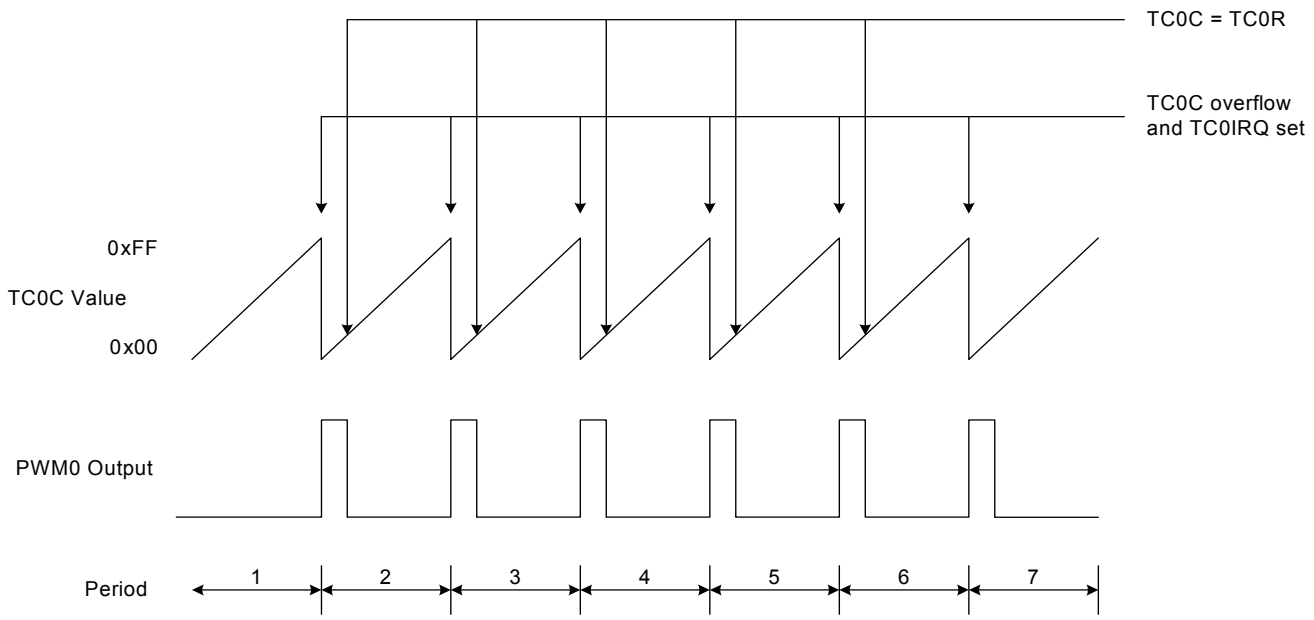
➢ **例：改变 TC0R 的内容。**

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

INCMS   BUF0
NOP
B0MOV   A, BUF0
B0MOV   TC0R, A
```

✷ **注：PWM0 可以在中断下工作。**

8.5.4 PWM0 占空比注意事项

在 PWM 模式下，系统会随时比较 TC0C 和 TC0R 的值。如果 TC0C<TC0R，PWM 输出高电平，而当 TC0C≧TC0R 时则输出低电平。当 TC0C 发生改变的时候，PWM 的占空比也随着改变，如果 TC0R 保持恒定，那么 PWM 输出波形也保持稳定。



上图所示是 TC0R 恒定时的波形。每当 TC0C 溢出时，PWM 都输出高电平，TC0C≧TC0R 时，PWM 输出低电平。

✷ **注：若要在程序处理过程中设置 PWM 的占空比，必须得在下一个周期开始时进行。**

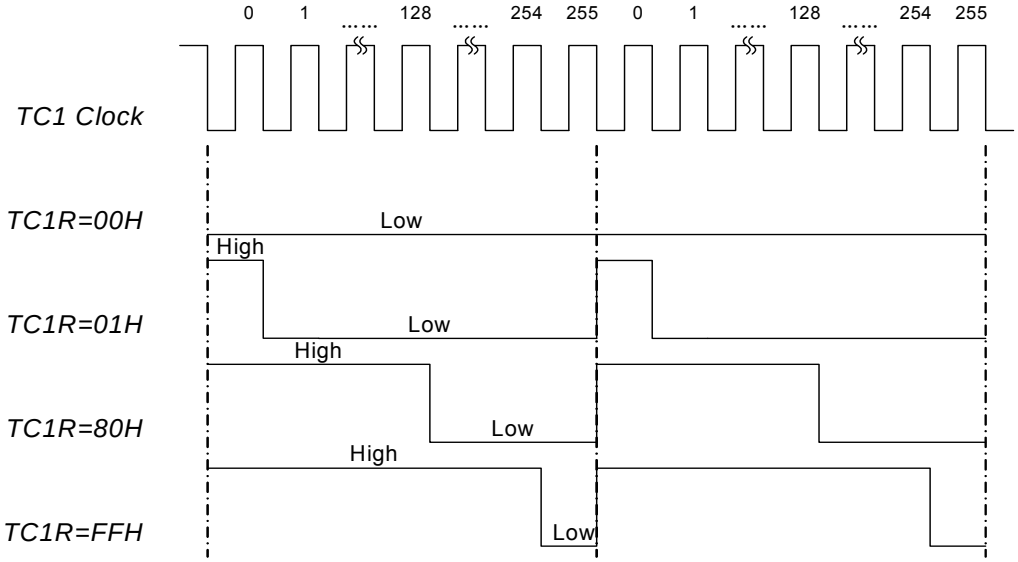
8.6 PWM1

8.6.1 概述

PWM 信号通过 PWM1OUT（P5.3 引脚）输出（256 阶）。8 位计数器 TC1C 计数过程中不断与 TC1R 相比较，当 TC1C 的值增加到与 TC1R 相等时，PWM 输出低电平，当 TC1C 的值溢出重新回到 0 时，PWM 被强制输出高电平。PWM1 输出占空比 = TC1R/ 256。

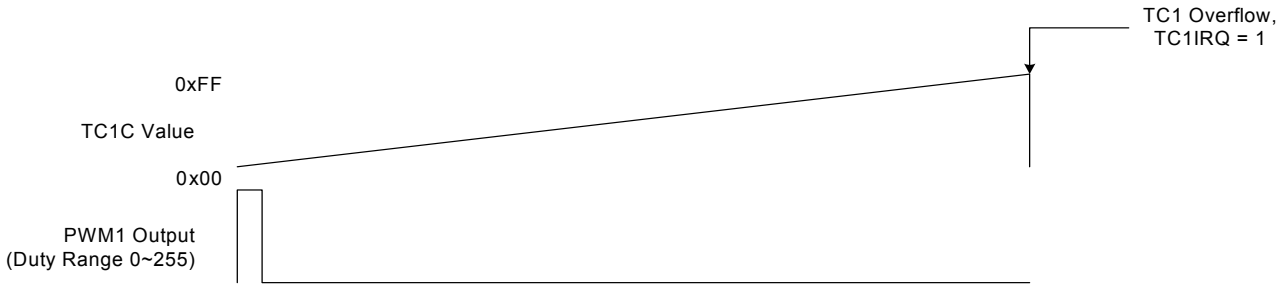
PWM 占空比范围	TC1C 有效值	TC1R 有效值	MAX. PWM 频率（Fcpu = 4MHz）	备注
0/256~255/256	00H~0FFH	00H~0FFH	7.8125K	每计数 256 次溢出

PWM 输出占空比随 TC1R 的变化而变化：0/256~255/256。



8.6.2 TC1IRQ 和PWM输出占空比

在 PWM 模式下，TC1IRQ 的频率与 PWM 的占空比有关，具体情况如下图所示：



8.6.3 PWM1 编程举例

➢ **例：PWM1 输出设置。外部高速振荡器输出频率= 4MHZ，Fcpu = Fosc/4，PWM1 输出占空比= 30/256，输出频率 1KHZ，PWM 时钟源来自外部时钟，TC1 速率= Fcpu/4，TC1RATE2~TC1RATE0 = 110，TC1C = TC1R = 30。**

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

MOV     A,#30           ; PWM 输出占空比＝30/256。
B0MOV   TC1C,A
B0MOV   TC1R,A

B0BSET  FPWM1OUT        ; PWM1 输出至 P5.3，禁止 P5.3 I/O 功能。
B0BSET  FTC1ENB         ; 使能 TC1 定时器。
```

✷ **注：TC1R 为只写寄存器，不能用 INCMS 和 DECMS 指令对其进行操作。**

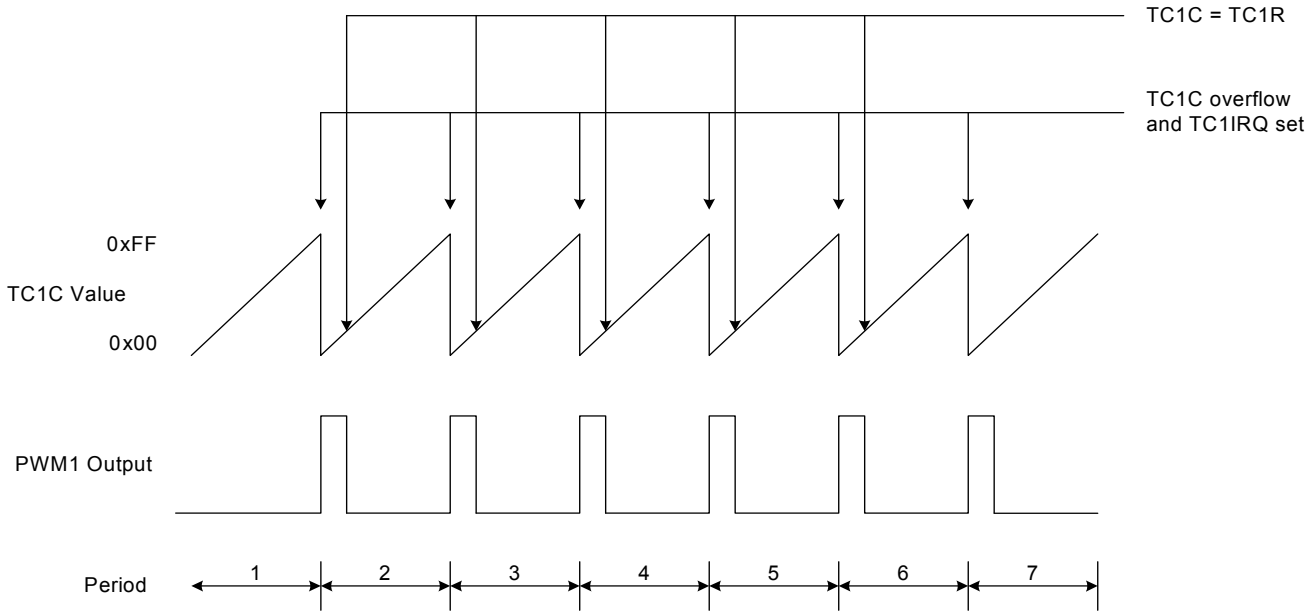
➢ **例：改变 TC1R 的内容。**

```
MOV     A, #30H
B0MOV   TC1R, A

INCMS   BUF0
NOP
B0MOV   A, BUF0
B0MOV   TC1R, A
```

✷ **注：PWM1 可以在中断下工作。**

8.6.4 PWM1 占空比注意事项

在 PWM 模式下，系统会随时比较 TC1C 和 TC1R 的值。如果 TC1C<TC1R，PWM 输出高电平，而当 TC1C≧TC1R 时则输出低电平。当 TC1C 发生改变的时候，PWM 的占空比也随着改变，如果 TC1R 保持恒定，那么 PWM 输出波形也保持稳定。



上图所示是 TC1R 恒定时的波形。每当 TC1C 溢出时，PWM 都输出高电平，TC1C≧TC1R 时，PWM 输出低电平。

✷ **注：若要在程序处理过程中设置 PWM 的占空比，必须得在下一个周期开始时进行。**

9 LCD驱动

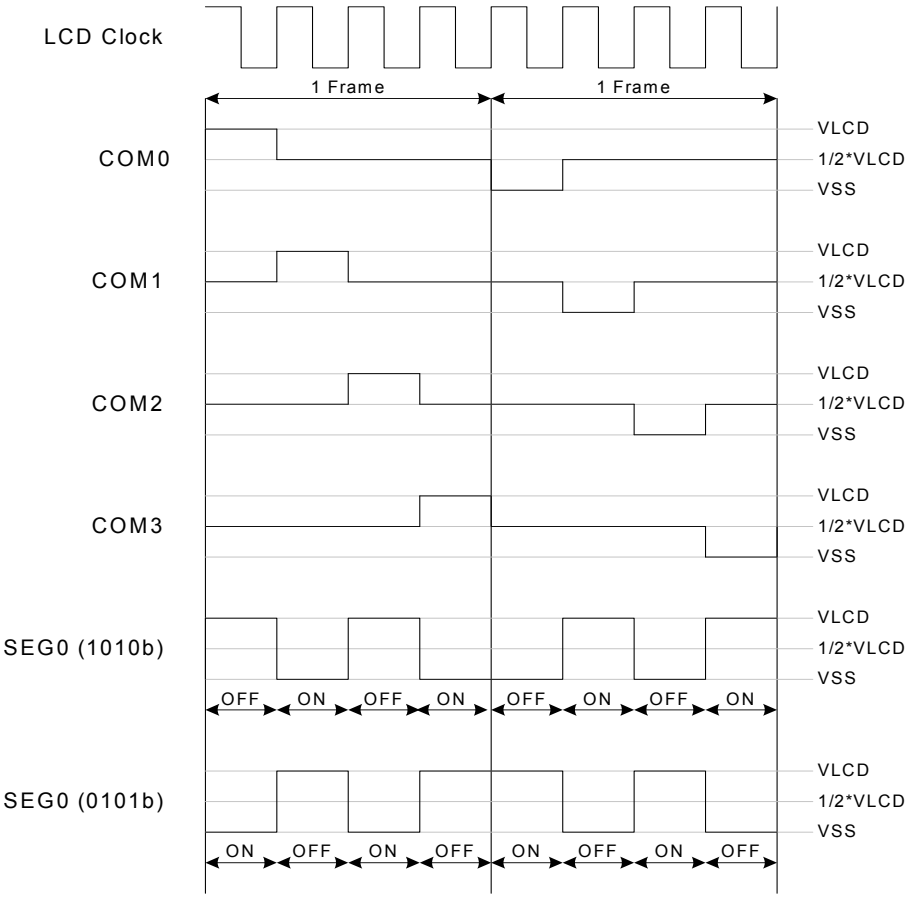
SN8P2949 的 LCD 驱动包括 R 型和 C 型，具有 4 个 common 引脚和 32 个 segment 引脚，LCD 扫描的时序占用 1/4 占空比，1/2 或者 1/3 偏压。R 型和 C 型 LCD 均支持所有功能，都有 128 点驱动。R 型 LCD 驱动下，LCD 电源和偏置电压都可以通过外部偏压电路进行调节；C 型 LCD 驱动下，通过内部 charge pump 进行调节。

9.1 LCD时序

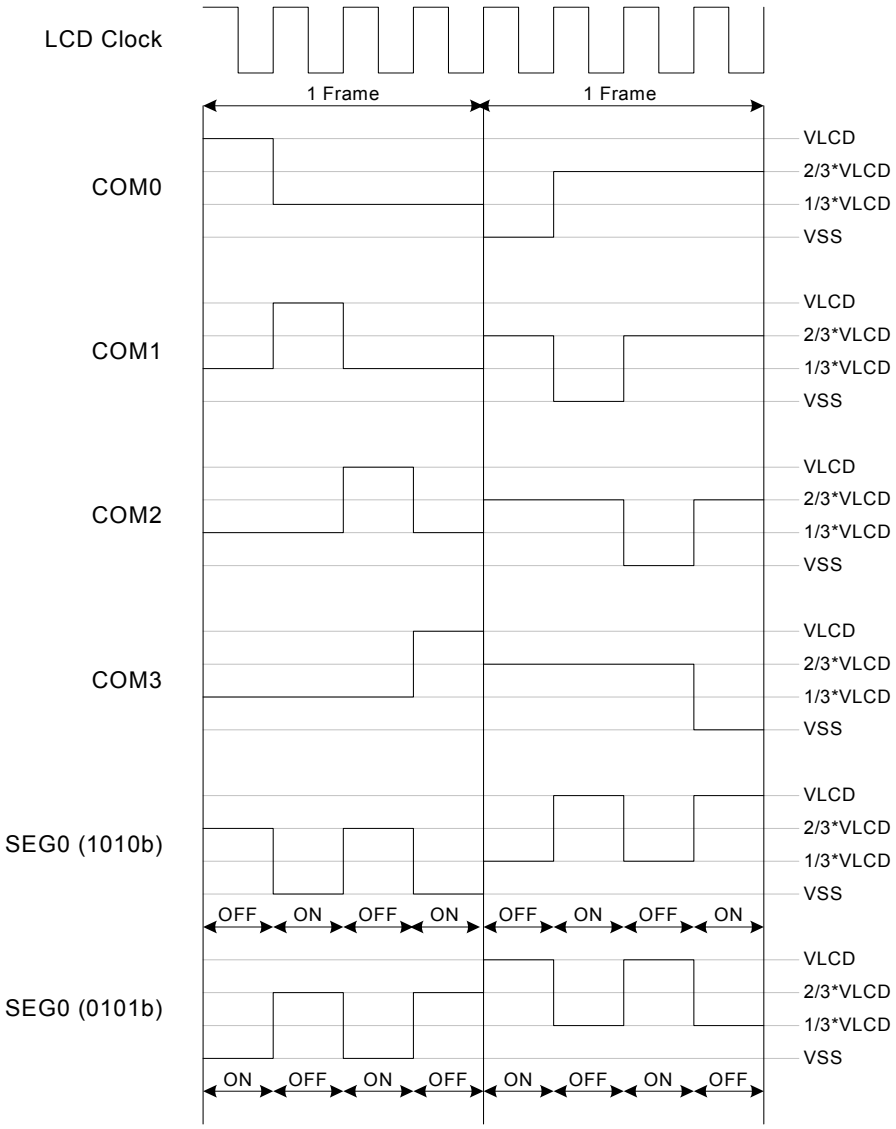
☞ **LCD 时序表**

LCDCLK	LCD 时钟源	LCDRATE	LCD 时钟	Frame = LCD clock/4	备注
0	Fhosc	X	4MHz / (2^14) = 244Hz	244Hz/4 = 61Hz	IHRC= 4MHz
1	Flosc	0	Flosc /128 = 250Hz	256Hz/4 = 64Hz	ILRC =32kHz@3.3V Or 32768Hz Crystal
1	Flosc	1	Flosc / 64 = 500Hz	512Hz/4 = 128Hz	

- 注 1：Flosc=ILRC, 32kHz@3.3V (Code Option=IHRC)
- 注 2：Flosc=32kHz Crystal (Code Option=IHRC_RTC)



LCD 驱动波形，1/4 duty，1/2 bias



LCD 驱动波形，1/4 duty，1/3 bias

9.2 LCDM1 寄存器

089H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM1	LCDREF1	LCDREF0	LCDBNK	LCDTYPE	LCDENB	LCDBIAS	LCDRATE	LCDCLK
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	1	1

Bit0 **LCDCLK**：LCD 时钟源选择控制位。
0：LCD 帧 Rate = IHRC/(2^16)= 61Hz，C 型 CP 时钟 = IHRC/64 = 62.5KHz；
1：LCD 帧 Rate = ILRC/512 = 64Hz（LCDRATE = 0），或者 ILRC/256 = 128Hz（LCDRATE = 1），C 型 CP 时钟 = ILRC = ~32KHz。

Bit1 **LCDRATE**：LCD 时钟 rate 控制位（LCDCLK=1）。LCD 时钟源由 ILRC 提供。
0 = LCD 帧 Rate = ILRC/512 = 64Hz；
1 = LCD 帧 Rate = ILRC/256 = 128Hz。

Bit2 **LCDBIAS**：LCD 偏压选择位。
0 = LCD 的偏压是 1/3；
1 = LCD 的偏压是 1/2。

Bit3 **LCDENB**：LCD 驱动使能控制位。
0 = 禁止，COM/SEG 不输出波形；
1 = 使能，COM/SEG 输出波形。

Bit4 **LCDTYPE**：R 型/ C 型 LCD 驱动控制位。
0 = R 型；
1 = C 型。

Bit5 **LCDBNK**：LCD 显示控制位。
0 = 正常显示；
1 = 关闭 LCD。

Bit[7:6] **LCDREF[1:0]**：R 型 LCD 偏压分压电阻的选择值。
00 = 400K；
01 = 200K；
10 = 100K；
11 = 33.3K。

R 型和 C 型 LCD 驱动控制：

	LCDPENB	LCDTYPE	LCDENB	BGM	LCDREF[1:0]
R 型设置	0	0	1	X	有效
C 型设置	1	1	1	1	无效
禁止 LCD 以省电（绿色模式和睡眠模式下）	0	X	0	X	X

✷ **注 1：绿色模式和睡眠模式下禁止 LCD，LCDENB 和 LCDPENB 均设为 0 以省电。**

9.3 LCDM2 寄存器

08AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LCDM2	-	BGM	LCDPENB	VPPINTL	VCP3	VCP2	VCP1	VCP0
读/写	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	1	0	0	0	0	1	1

Bit[0:3] **VCP[3:0]**：C 型 VLCD 输出电压选择控制位。（LCDEPNB = LCDTYPE = 1 时，LCD charge pump 开始 pumping)

	1/3 偏压条件下			1/2 偏压条件下		
VCP[3:0]	V2	V3	VLCD	V2	V3	VLCD
0000	0.9V	1.8V	2.7V	1.35V	1.35V	2.7V
0001	0.93V	1.86V	2.8V	1.40V	1.40V	2.8V
0010	0.96V	1.93V	2.9V	1.45V	1.45V	2.9V
0011	1.00V	2.00V	3.0V	1.50V	1.50V	3.0V
0100	1.03V	2.06V	3.1V	1.55V	1.55V	3.1V
0101	1.06V	2.13V	3.2V	1.60V	1.60V	3.2V
0110	1.10V	2.20V	3.3V	1.65V	1.65V	3.3V
0111	1.13V	2.26V	3.4V	1.70V	1.70V	3.4V
1000~1110	Reserve			Reserve		
1111	-	-	-	2.3V	4.6V	6.9V*

*** VLCD 输出 6.9V 用于 ISP VPP。**

Bit[4] **VPPINTL**：内部 VPP 控制位。
0 = VPP 无电压源；
1 = VPP 与 VLCD 内部短路，用于 ISP。（C 型 LCD CP 输出高电压，与 VPP 引脚短路用于 ISP）

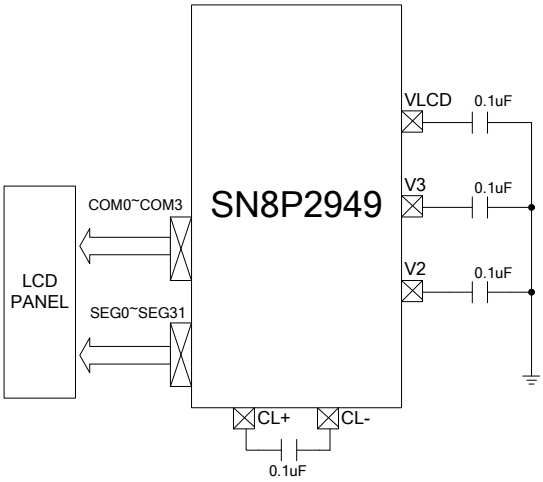
Bit5 **LCDPENB**：C 型 LCD Pump 使能位。
0 = 禁止；
1 = 使能。

Bit6 **BGM**：Band Gap 选择位，用于 LCD 参考电压。
0 = 保留；
1 = 系统 Band Gap 电压参考（始终置 1）。

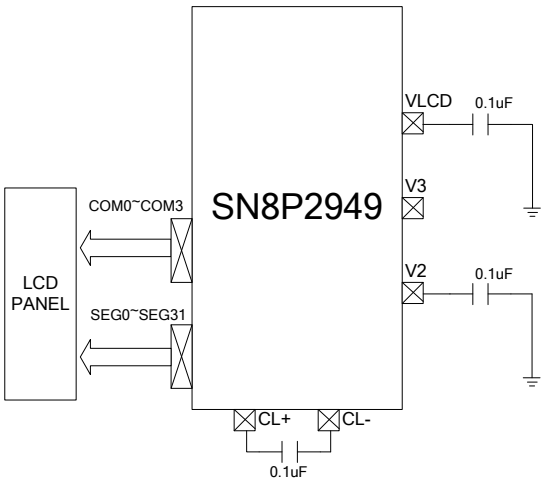
✷ 注 1：VCP[3:0]=1111，LCDBIAS=1，LCDTYPE=1 且 LCDPENB=1 时，LCD charge pump 用来产生 VPP 电压用于 ISP，无需额外的 6.5V 输入。不能使能 LCDENB，COM 和 SEG 脚不会出现高电压，以保护 LCD 面板。
✷ 注 2：VCP 和 VPPINTL 控制 ISP 功能，无需外部 VPP 6.5V 电压。请参考“ROM 在线烧录“章节。
✷ 注 3：如果设置 VLCD 设置 6.9V 高压用于 ISP，则 VLCD/V3/V2 必须单独连接一个 0.1uF 的电容到 DVSS，同样 CL+/CL-也需连接一个 0.1uF 的电容。
✷ 注 4：宏指令“ROMWRTVPP”包含了内部 VPP 产生的程序和用于 ISP 功能的 ROMWRT 指令，无需外部 6.5V 的要求。

9.4 C型LCD驱动模式

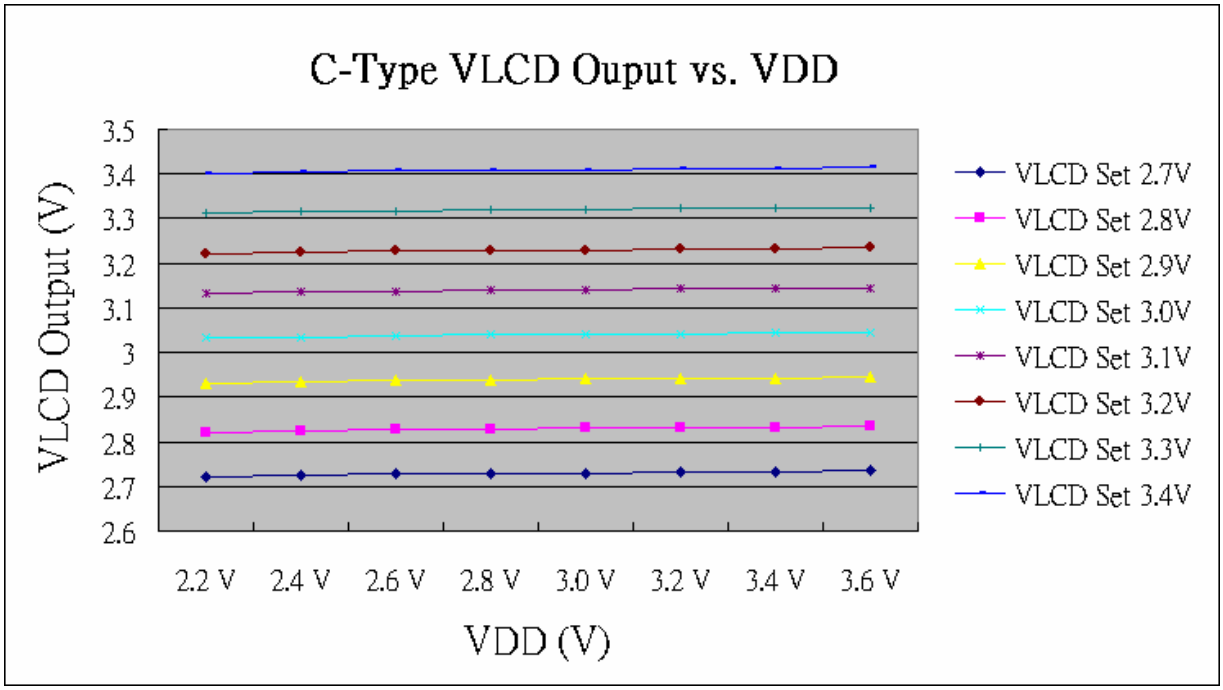
C 型 LCD 驱动模式支持 1/3 和 1/2 偏压 LCD 面板。LCD 电源（VLCD）由内部 LCD charge-pump 提供。C 型 LCD Charge-pump 的电压为 VLCD 的电压，V2 为 Charge-pump 的电压源，即 1/3 * VLCD；V3 是 2*V2，即 2/3 * VLCD。C 型 LCD 模式下，必须使能 LCDM1 寄存器的 LCDTYPE 和 LCEPENB 位。下图为 1/3 和 1/2 偏压 C 型 LCD 应用电路和 VLCD 输出电压曲线图。



1/3 偏压，C 型 LCD 应用电路



• 1 1/2 偏压，C 型 LCD 应用电路



✷ 注 1：C 型 LCD 模式下，VLCD 电源没有连接到 VDD。
✷ 注 2：C 型 LCD 模式下，在 CL+和 CL-引脚之间、VLCD/V3/V2 和 VSS 之间连接一个 0.1uF 的电容。
✷ 注 3：VLCD 的输出电压范围为 2.7V~3.4V，精确度为±0.2V。

➢ **例：使能 C 型 LCD 功能（设置 VLCD=3.0V，1/3 偏压模式）。**

C 型 LCD 设置：

```
B0BCLR      FLCDBIAS    ; 设置 1/3 偏压。
B0BCLR      FVCP3       ; 设置 VLCD = 3V。
B0BCLR      FVCP2       ; 设置 VLCD = 3V。
B0BSET      FVCP1       ; 设置 VLCD = 3V。
B0BSET      FVCP0       ; 设置 VLCD = 3V。
B0BSET      FBGM        ; 必须设置 BGM=1。
B0BSET      FLCDTYPE    ; C 型 LCD。
B0BSET      FLCDPENB    ; 使能 CP。
LcdPumpStart            ; C 型 pump 开始宏指令，必须执行此指令。
                        ; 使用宏指令“LcdPumpStart”。
B0BSET      FLCDENB     ; 使能 LCD 驱动，COM/SEG 输出波形。
Call        Set_LCD_RAM ; 设置 LCD RAM 显示 LCD 面板。
....
```

✸ 注 1：执行 C 型 LCD 驱动模式的流程是先设置 BGM=LCDTYPE=LCDPENB=1，偏压，和 VCP[3:0]，然后波形使用宏指令“LcdPumpStart”执行此功能并检测 pump 是否启动。

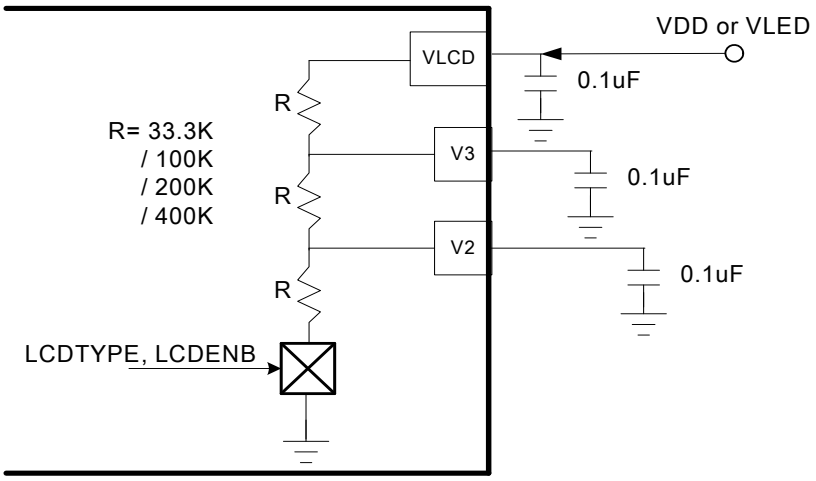
✸ 注 2：使用宏指令“LcdPumpStart”之前，先要包含 2949_Macro.h 文件。

SYM.	参数	说明	MIN.	TYP.	MAX.	UNIT
ILCDC	C 型 LCD 驱动 工作电流（无 LCD 面板）	温度：25°C，VDD = 3V，时钟：ILRC（LCDCLK=1） 电流：BGM(1) + Pump + LCD 驱动	-	120	150	uA
ILCDR	R 型 LCD 驱动 工作电流（无 LCD 面板）	温度：25°C，VDD = 3V，（内部 R=400K，1/3 偏压）	-	5	8	
VLCD	VCP[3:0] = 0011	温度：0~70°C，Vdd = 2.2 ~ 3.6V，C 型 LCD	2.8	3	3.2	V
VVPPL	内部 VPP 生成	温度：0~70°C，Vdd = 2.2 ~ 3.6V charge pump 输出与 VPP 短接	6.4	6.7	7.0	V

9.5 R型LCD驱动模式

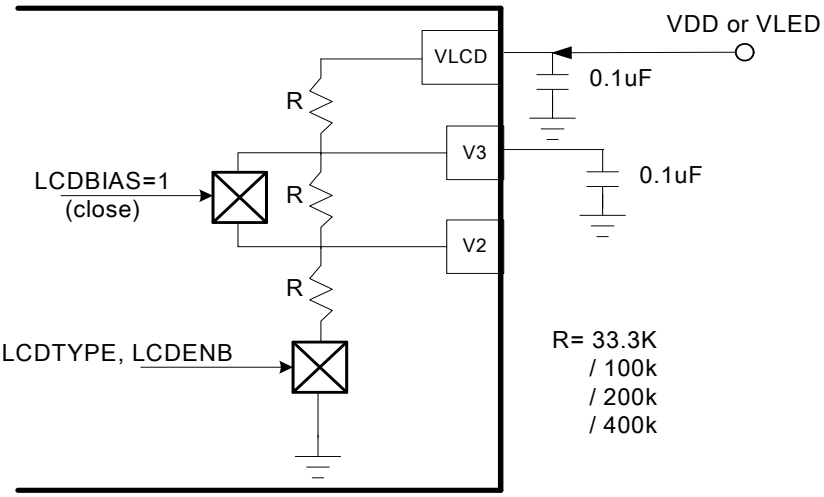
R 型 LCD 驱动模式下，LCD 电源（VLCD）通过 VLCD 引脚连接到外部电源，而不是连接到内部 VDD，V2 和 V3 偏压源为内部电压通过电阻分压后获得。LCD 驱动电路内置分压电阻，分别有 33.3K/100K/200K/400K 可供选择。用户可以在 VLCD/V2/V3 之间选择合适的外部电阻以获得更多的驱动电流。

1/4 占空比，1/3 偏压：



注：V3= 2/3*VLCD，V2 = 1/3*VLCD。LCD 电流功耗 $= \frac{VLCD}{3R}$

1/4 占空比，1/2 偏压：



注：V3 = V2 = 1/2*VLCD，LCD 电流功耗 $= \frac{VLCD}{2R}$

✷ **注 1：R 型 LCD 驱动模式下，VLCD 引脚必须输入电压以供 LCD 电路工作。**

9.6 LCD RAM位置

RAM bank 15 的地址与 Common/Segment 引脚位置的关系：

	Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7
	COM0	COM1	COM2	COM3	-	-	-	-
SEG 0	00H.0	00H.1	00H.2	00H.3	-	-	-	-
SEG 1	01H.0	01H.1	01H.2	01H.3	-	-	-	-
SEG 2	02H.0	02H.1	02H.2	02H.3	-	-	-	-
SEG 3	03H.0	03H.1	03H.2	03H.3	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	-	-	-
SEG 31	1FH.0	1FH.1	1FH.2	1FH.3	-	-	-	-

➢ **例：使能 LCD 功能。**

设置 LCD 控制位 LCDENB 和 LCD RAM，显示 LCD 面板。

R 型 LCD 设置：

```
B0BCLR      FLCDTYPE        ; R 型 LCD。
B0BCLR      FLCDPENB        ; 禁止 CP。
B0BSET      FLCDENB         ; 使能 LCD 驱动。
CALL        Set LCD RAM     ; 设置 LCD RAM 以显示 LCD 面板。
```

10 在线烧录（ISP）

10.1 概述

SN8P2949 具有在线烧录 ROM 功能（ISP ROM），为用户将数据存储在 ROM 中提供了一种简易的方式。选择 ROM 地址后，执行 ROM 烧录指令-ROMWDT，并向 VPP 输入 6.5V 的电压，由寄存器 ROMCNT 控制烧录时间。烧录完成后，ROMDAH/ROMDAL 中的数据被烧录到 ROMADRH/ROMADRL 地址处。

10.2 ROMADRH/ROMADRL寄存器

0A0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMADRH	VPPCHK	-	-	ROMADR12	ROMADR11	ROMADR10	ROMADR9	ROMADR8
读/写	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0A1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMADRL	ROMADR7	ROMADR6	ROMADR5	ROMADR4	ROMADR3	ROMADR2	ROMADR1	ROMADR0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **VPPCHK**：VPP 烧录电压检测位。
0 = VPP 电压不到 6.5V，不能进行 ISP ROM 烧录；
1 = VPP 的电压为 6.5V，可以进行 ISP ROM 烧录。

ROMADR[12:0]：ISP ROM 烧录地址。
即将被烧录数据的 ROM 地址。

10.3 ROMDAH/ROMDAL寄存器

0A2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMDAH	ROMDA15	ROMDA14	ROMDA13	ROMDA12	ROMDA11	ROMDA10	ROMDA9	ROMDA8
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0A3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMDAL	ROMDA7	ROMDA6	ROMDA5	ROMDA4	ROMDA3	ROMDA2	ROMDA1	ROMDA0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

ROMDA[15:0]：ISP ROM 烧录数据。
需要被烧录到 ROM 区域的 ROM 数据。

10.4 ROMCNT寄存器和ROMWRT指令

0A4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ROMCNT	-	-	-	-	-	-	ROMCNT1	ROMCNT0
读/写	-	-	-	-	-	-	W	W
复位后	-	-	-	-	-	-	0	0

Bit[1:0] **ROMCNT[1:0]**：ISP ROM 烧录时间计数器。
ISP ROM 烧录时间由 ROMCNT [1:0]控制。
请设置烧录时间 30us。

Fcpu	ROMCNT [1:0]	烧录时间
1 or 0.5MIP	00	240us
1 or 0.5MIP	01	120us
1 or 0.5MIP	10	60us
1 or 0.5MIP	11	30us

设置完成后，执行 ROMWRT 指令，将 ROMDA[15:0]的数据写入地址 ROMADR[12:0]中。

✷ 注 1：在访问 ISP ROM 时，请将 VDD 电压设置为 3V。
✷ 注 2：访问 ROMWRT 后，必须延时 3 个 NOP 指令的时间。
✷ 注 3：在室温环境下（25°C ）执行 ISP 功能。
✷ 注 4：在没有外部 VPP 6.5V 输入时，使用宏指令“ROMWRTVPP”执行 ISP 功能，由 IC 内部产生 VPP 高压。在此种条件下，VL+/-/VLCD/V3/V2 必须单独外接一个 0.1uF 的电容。
✷ 注 5：执行 ISP 之前必须禁止中断功能。

10.5 ISP ROM操作举例

➢ **例：ISP 示例程序（由外部 VPP 6.5V 输入）。**

```
;保留 ISP ROM 地址为 0FFFFH。
                ORG             0100H
@CALDATA:
                DW              0FFFFH
                ……
                ……
; 烧录数据 0AA55H 存入地址@CALDATA。
                MOV             A,#@CALDATA$L
                B0MOV           ROMADRL, A          ; 低字节地址存入 ROMADRL。
                MOV             A, #@CALDATA$H
                B0MOV           ROMADRH, A          ; 高字节地址存入 ROMADRH。
                MOV             A, #55H
                B0MOV           ROMDAL, A           ; 低字节数据存入 ROMDAL。
                MOV             A, #0AAH
                B0MOV           ROMDAH, A           ; 高字节数据存入 ROMDAH。
; VPP 电压检测。
                @B0BTS1_FVPPCHK                     ; 检测 VPP 电压是否为 6.5V。
                JMP             $-1                 ; 如果 VPP 不到 6.5V，继续等待。
; 设置烧录计数器，访问 ISP ROM。
@ROM_WRT:
                MOV             A, #1               ; 设置烧录计数器。
                B0MOV           ROMCNT, A
                B0BCLR          FGIE                ; 执行 ISP 之前禁止中断。
                ROMWRT                              ; 烧录 ISP ROM。
                NOP                                 ; NOP 延时。
                NOP                                 ; NOP 延时。
                NOP                                 ; NOP 延时。
                B0BSET          FGIE                ; 必须使能中断。
; VPP 电压检测。
                @B0BTS0_FVPPCHK                     ; 检测 VPP 电压是否为 VDD。
                JMP             $-1                 ; 如果 VPP 电压不为 VDD，继续等待。
; 检测烧录数据。
                B0MOV           Z, #@CALDATA$L
                B0MOV           Y, #@CALDATA$H
                MOVC                                ; 存储 ISP ROM 数据到 A 和 R。

                CMPRS           A, #55H
                JMP             @WRT_ERR
                B0MOV           A, R
                CMPRS           A, #0AAH
                JMP             @WRT_ERR            ; 检测 ISP ROM 数据是否正确。
```

➢ **例：ISP 示例程序（由内部 VPP 产生）。**

- RAM 的刻度数据“Cal_Data[8]”（8 字节）。
- ISP ROM 地址从 700H 到 703H（4 个字）。
- 使用宏指令“ROMWRTVPP”。

```
ISP_Internal:
                MOV             A, #07H
                B0MOV           ROMADRH, A          ; ISP ROM 初始地址 700H。
                CLR             ROMADRL
                MOV             A, #3               ; 设置 ISP 烧录时间最长为 30us。
                B0MOV           ROMCNT, A

; --------从 RAM“ISP_Data”下载 ISP 数据地址--------------------
                MOV             A, @Cal_Data$L      ; RAM 刻度数据“Cal_Data[8]”。
                B0MOV           Z, A
                CLR             Y
; ---------Cal_Data 数据到[ROMDAH, ROMDAL]------------------------
@@:
                B0MOV           A, @YZ
                B0MOV           ROMDAH, A
                INCMS           Z
                B0MOV           A, @YZ
                B0MOV           ROMDAL, A
; ---------存储寄存器 Y 和 Z-----------------------------------------------
                B0MOV           A, Z
                B0MOV           Z_buffer, A
                B0MOV           A, Y
                B0MOV           Y_buffer, A
; ---------[ISP ROM 写命令]-----------------------------------------------
                B0BCLR          FGIE                ; 禁止中断。
                ROMWRTVPP                           ; ISP ROM 写宏指令。
                B0BSET          FGIE                ; 使能中断。
; ---------恢复寄存器 Y 和 Z-----------------------------------------------
                B0MOV           A, Z_buffer
                B0MOV           Z, A
                B0MOV           A, Y_buffer
                B0MOV           Y, A

                INCMS           ROMADRL             ; 下一个 ISP ROM 地址。
                MOV             A, #4               ; ISP 地址[700H, 701H, 702H, 703H]。
                CMPRS           A, ROMADRL
                JMP             @B
; ------------[ISP 结束]-------------------------------------------------------
                CALL            ISP_ROM_CHECK       ; 检查 ISP ROM 数据是否正确。
```

11 Regulator, PGIA和ADC

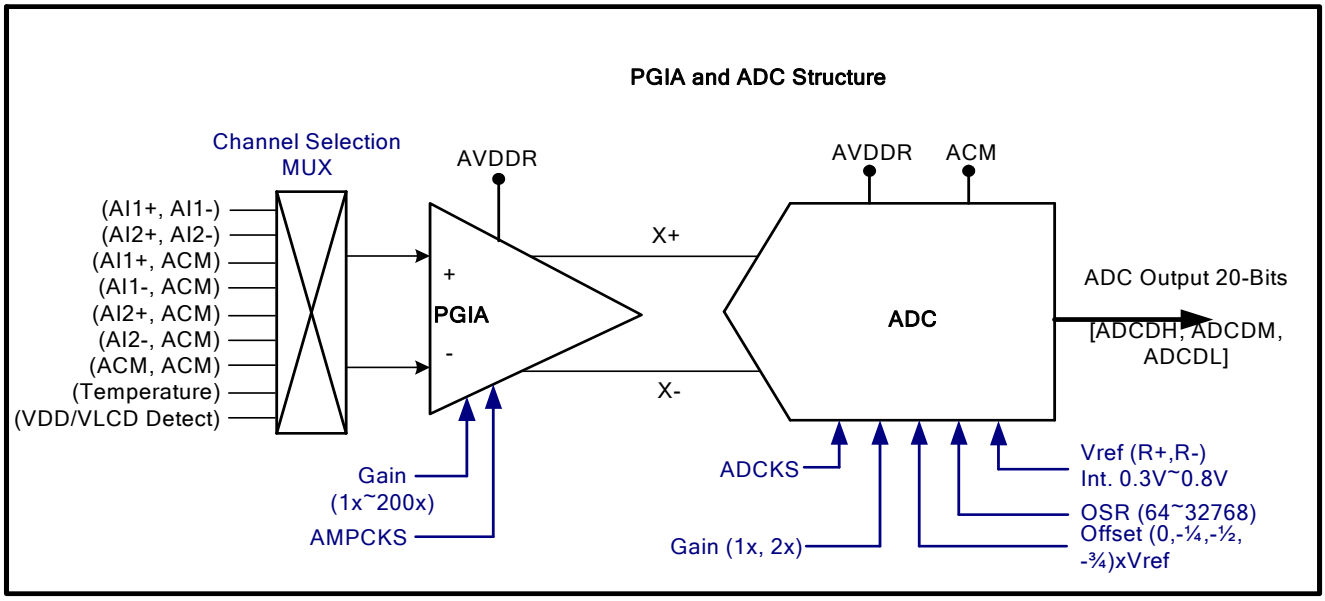
11.1 概述

SN8P2949 内置稳压器 Regulator，从 AVDDR 输出稳定的 2.4V，而从 AVE+输出稳定的 1.5V/2.0V，最大的驱动电流为 5mA。AVDDR 给内部电路（PGIA、ADC）和外部传感器（压力传感器或热敏电阻）提供稳定的电压。SN8P2949 具有完整的 ΔΣ 模拟数字转换器（ADC），具有 20 位性能，高达 18 位的精度。快速 ADC 模式下，ADC 转换 Rate 高达为 3.9KHz，12.4 位的分辨率（Gain=1，Vref=0.8V）。ADC 内置内部增益选项，可选择范围为 x1 和 x2。PGIA 有 2 个不同的输入信道模式：（1）2 个全差分输入[AI1+，AI1-]和[AI2+，AI2-]；（2）4 个单端输入。ADC 在压力测量和医用仪表方面可以进行单/双极性的测量。内置增益可调的低噪声可编程增益放大器（PGIA），在应用时，可以选择 1x、12.5x、50x、100x 和 200x 五种增益。

11.2 模拟输入

下图是 PGIA 和 ADC 模块结构简图，由一个多路选择器（用于输入通道的选择），一个可编程增益放大器（PGIA）和 ΔΣ ADC 模块组成。

为了使 ADC 输出的范围达到最大，ADC 的输入信号电压 V（X+、X-）应该接近于但不能超过参考电压 V（R+、R-），选择一个合适的参考电压和合适的 PGIA 可以使 ADC 的输出范围很大。相关的控制位是 ADCM1 寄存器的 RVS 和 IRVS（参考电压选择）位以及 AMPM 寄存器的 GS[2:0]（增益选择）位。



11.3 电压稳压器

SN8P2949 内置一个电压稳压器，提供 2.4V 的稳定电源（来自 AVDDR）和 1.5V/2.0V 的稳定电源（来自 AVE+），最大驱动电流 5mA。寄存器 VREG 可以控制 AVDDR、AVE+和 ACM 的电压输出状态。由于 PGIA 和 ADC 的电源来自 AVDDR，因此在使能 PGIA 和 ADC 之前要打开 AVDDR（AVDDRENB = 1），而 AVDDR 端的电压由 VDD 管控。

11.3.1 电压稳压器控制寄存器

090H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
VREG	BGRENB	ACMSEL	ACMENB	AVESEL	AVENB	AVDDRSEL	AVDDRENB	-
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	-
复位后	0	1	0	1	0	1	0	-

Bit1 **AVDDRENB**：Regulator（AVDDR）电压使能控制位。
0 = 禁止 AVDDR regulator 输出电压；
1 = 使能 AVDDR regulator 输出电压。

Bit2 **AVDDRSEL**： AVDDR 电压选择控制位。
0 = 保留；
1 = 输出 2.4V。

Bit3 **AVENB**：AVE+电压输出控制位。
0 = 禁止 AVE+输出电压；
1 = 使能 AVE+输出电压。

Bit4 **AVESEL**：AVE+电压选择控制位。
0 = AVE+输出 1.5V；
1 = AVE+输出 2.0V。

Bit5 **ACMENB**：模拟电路公共端（ACM）电压使能控制位。
0 = 禁止模拟电路公共端和 ACM 输出电压；
1 = 使能模拟电路公共端和 ACM 输出电压。

Bit6 **ACMSEL**：ACM 电压选择控制位。
0 = 保留
1 = 模拟电路公共端电压 ACM 输出 1V。

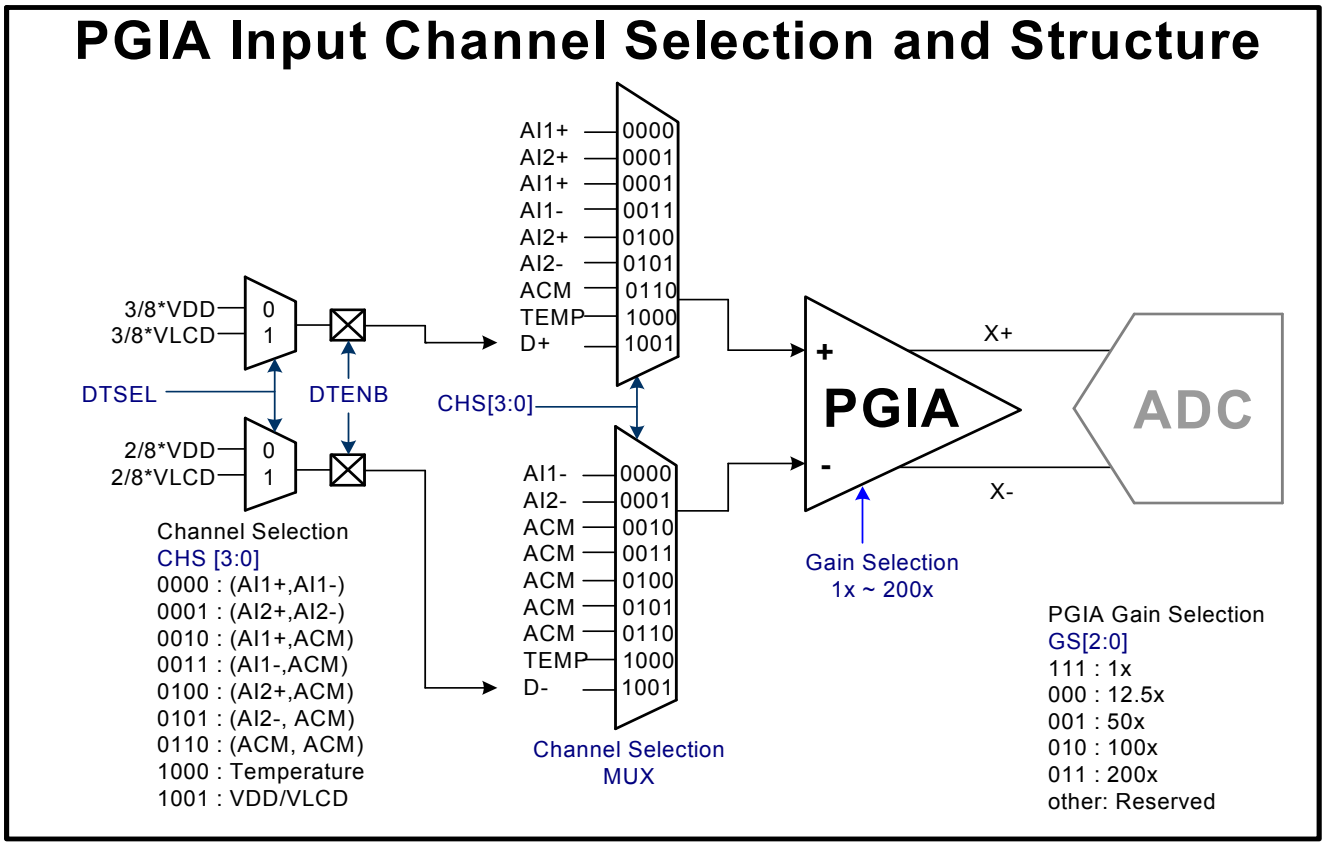
Bit7 **BGRENB**：Band Gap 参考电压使能控制位。
0 = 禁止 Band Gap 参考电压；
1 = 使能 Band Gap 参考电压。

✷ 注 1：在使能下列功能之前，必须先打开 Band Gap 的参考电压（详见 AMPM1 和 AMPM2 寄存器）。
（1）AVDDR、AVE+和 ACM 稳压器；
（2）PGIA 功能；
（3）低电压检测功能。
✷ 注 3：PGIA 可以在普通模式、低速模式和绿色模式下工作，此时高速时钟仍然工作（STPHX=0）。
✷ 注 4：使能每一个稳压器（AVDDR/AVE/ACM）后增加 10ms 的延迟时间，这样在 CR2032 电池应用场合时可以避免 VDD 跌落。

SYM.	说明	参数	MIN.	TYP.	MAX.	UNIT
VAVDDR	Regulator 输出电压 AVDDR	AVDDR=2.4V, @25°C, Vdd = 2.6~3.6V.	2.25	2.4	2.55	V
		AVDDR=2.4V, @25°C, Vdd < 2.45V	Vdd – 0.05V			
VAVE+	Regulator 输出电压 AVE+	VACM = 2.0V, @25°C, Vdd = 2.4 ~ 3.6V.	1.85	2.0	2.15	
		VACM = 1.5V, @25°C, Vdd = 2.4 ~ 3.6V.	1.35	1.5	1.65	
VACM	模拟公共电压	VACM = 0.6V, @25°C, Vdd = 2.4 ~ 3.7V.	0.9	1.0	1.1	

11.4 PGIA-可编程增益放大器

SN8P2947 内置一个增益可调的低噪声可编程增益放大器（PGIA），通过寄存器 AMPM1 可以选择 1x、12.5x、50x、100x 和 200x 增益。PGIA 还提供 2 种信道选择模式：（1）2 个全差分输入；（2）4 个单端输入，由寄存器 AMPM1 控制。



11.4.1 AMPM1- 放大器模式控制寄存器

091H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AMPM1	CHS3	CHS2	CHS1	CHS0	GS2	GS1	GS0	AMPENB
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	1	0	1	1	1	1	1	0

Bit0 **AMPENB**：PGIA 功能使能控制位。

0 = 禁止；

1 = 使能。

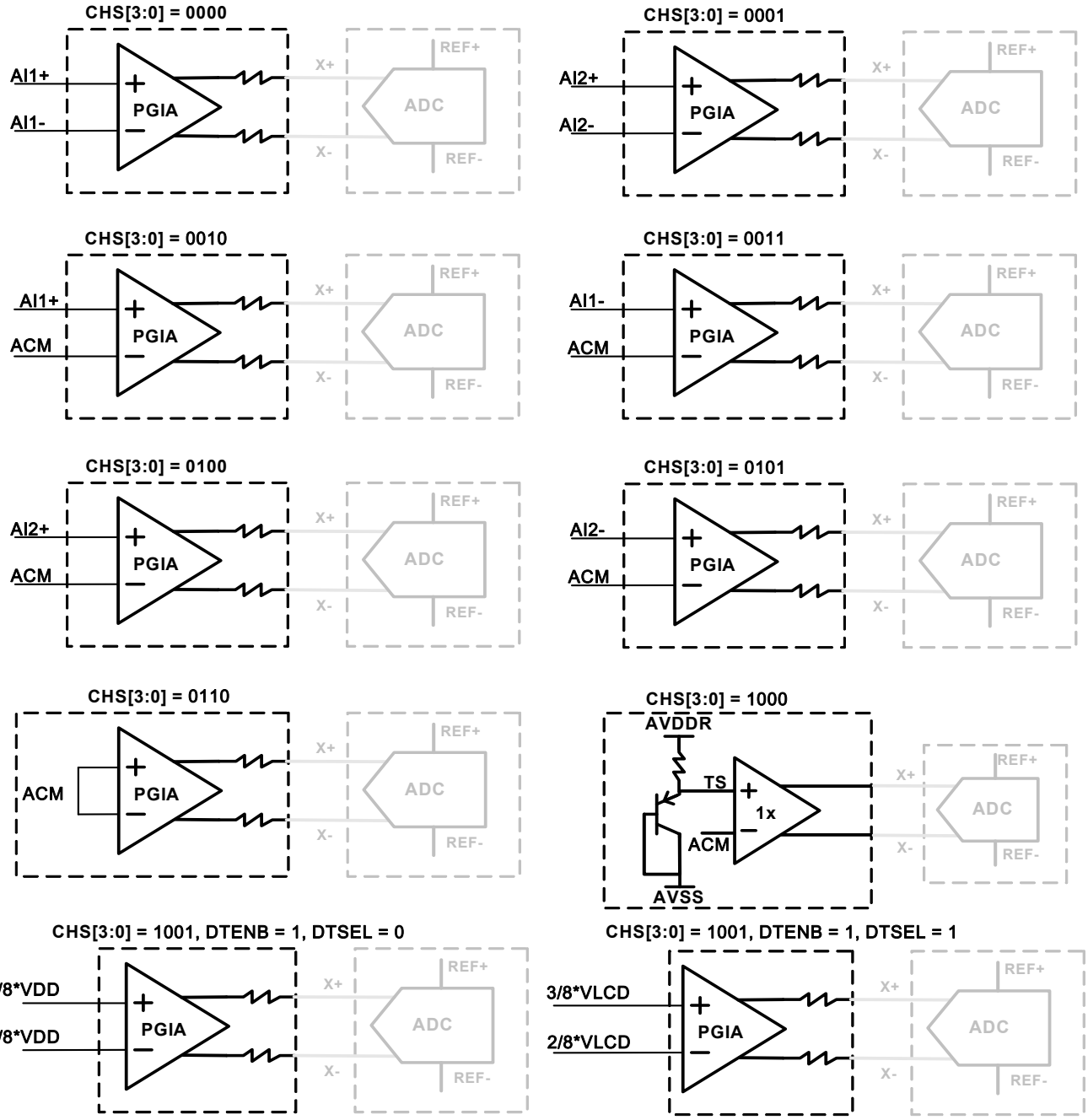
Bit[3:1] **GS [2:0]**：PGIA 增益选择控制位。

GS [2:0]	PGIA 增益
000	12.5
001	50
010	100
011	200
100,101,110	Reserved
111	1

Bit[7:4] **CHS [3:0]**：PGIA 通道选择。

PGIA 通道选择表：

CHS [3:0]	选择通道	ADC 输入	输入信号类型
0000	AI1+, AI1-	V (AI1+, AI1-) × PGIA Gain	差分
0001	AI2+, AI2-	V (AI2+, AI2-) × PGIA Gain	差分
0010	AI1+, ACM	V (AI1+, ACM) × PGIA Gain	单端
0011	AI1-, ACM	V (AI1-, ACM) × PGIA Gain	单端
0100	AI2+, ACM	V (AI2+, ACM) × PGIA Gain	单端
0101	AI2-, ACM	V (AI2-, ACM) × PGIA Gain	单端
0110	ACM, ACM	V (ACM, ACM) × PGIA Gain	输入短路
0111	保留	-	-
1000	温度传感器	V (VTS, ACM) × 1	N/A
1001	电压检测	VDD (2/8VDD, 3/8VDD) × PGIA Gain VLCD (2/8VLDD, 3/8VLDD) × PGIA Gain	差分



✷ 注 1：V（AI1+, AI1-）=（AI1+电压 – AI1-电压）
✷ 注 2：V（AI2+, ACM）=（AI2+电压 – ACM 电压）
✷ 注 3：输入短路模式仅用来测试 PGIA 的偏移量。
✷ 注 4：选择 1x 的增益（GS0[2:0]=111）时，AI+/AI-信号避开 PGIA 而直接输入 ADC。禁止 PGIA（AMPENB=0）以省电，必须使能 ADC 输入缓存器（GX=1），用于存储 ADC 的高阻抗性能。
✷ 注 5：使能输入缓存器（GX=1 或者 GR=1）时，输入信号的绝对电压范围必须在 0.4V-1.4V 之间。

11.4.2 AMPM2- 放大器模式控制寄存器

092H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AMPM2	INRENB	GX	GR	AMPCKS1	AMPCKS0	PCHPENB	DTENB	DTSEL
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	1	0	0	1	0	1	0	0

Bit0 **DTSEL**：VDD/VLCD 电压检测功能控制位。
0 = 选择 VDD 电压检测功能；
1 = 选择 VLCD 电压检测功能

Bit1 **DTENB**：VDD/VLCD 电压检测功能使能位。
0 = 禁止 VDD/VLCD 电压检测功能；
1 = 使能 VDD/VLCD 电压检测功能。

Bit2 **PCHPENB**：PGIA Chopper 使能位。
0 = 禁止 PGIA Chopper；
1 = 使能 PGIA Chopper。

Bit[4:3] **AMPCKS[1:0]**：PGIA Chopper 频率选择位。 （始终设置 AMPCKS[1:0]=10）

Bit5 **GR**：R+ R-单元增益缓存器使能位。
0 = 禁止 R+ R- UGB 功能；（ADC Vref 由内部设置）
1 = 使能 R+ R- UGB 功能。（ADC Vref 有外部引脚 R+和 R-设置）

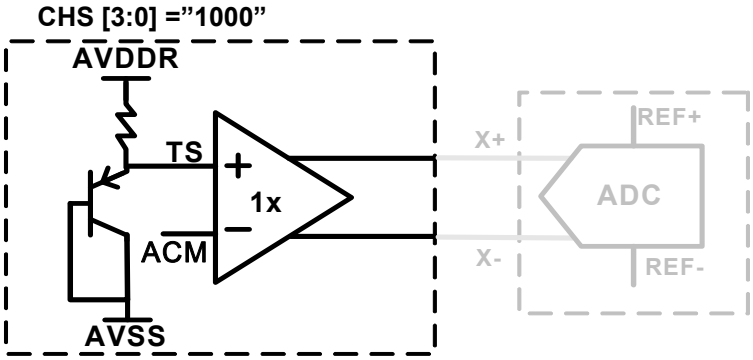
Bit6 **GX**：X+ X-单元增益缓存器使能位。
0 = 禁止 X+X- UGB 功能；（PGIA 增益设置为 x12.5，x50，x100 或 x200）
1 = 使能 X+X- UGB 功能；（PGIA 增益设置为 x1）

Bit7 **INRENB**：始终设置为 1。

- **注 1：普通应用中，请设置 AMPCKS[1;0] = 10。**
- **注 2：设置 INRENB=1 以增强 EMS 特性。**
- **注 3：使能输入缓存器（GX=1 或者 GR=1）时，输入信号的绝对电压范围必须在 0.4V-1.4V 之间。**

11.5 温度传感器（TS）

在应用中，不同的环境温度会使传感器的特性也有所不同，为了得到不同的温度信息，SN8P2949 内置了一个温度传感器（TS）来测量工作环境温度。通过相对应的 PGIA 通道达到测量环境温度的目的。



- **注 1：当选择温度传感器时，PGIA 的增益要选择为 1x，否则会出错。**
- **注 2：这样设置后，X+的电压就是 V（TS），X-的电压是 ACM。**
- **注 3：这里的温度传感器只是一个参考数据而不是真实的室温，在精确的应用中，请选用外部的温度传感器。**

在 25^{0}C 的环境下，V（TS）大约是 1V。如果温度上升 10C，V（TS）就会下降 32mV（VTS=1.032V）；相反，若温度下降 10C，V（TS）则会上升 32mV（VTS=0.968V）。

例：

温度	V(TS)	(X+) – (X-)	ADC Vref	ADC 输出（16-Bit）
15℃	0.968V	-0.032V	0.6V	-1747
25℃	1.000V	0V	0.6V	0
35℃	1.032V	0.032V	0.6V	1747

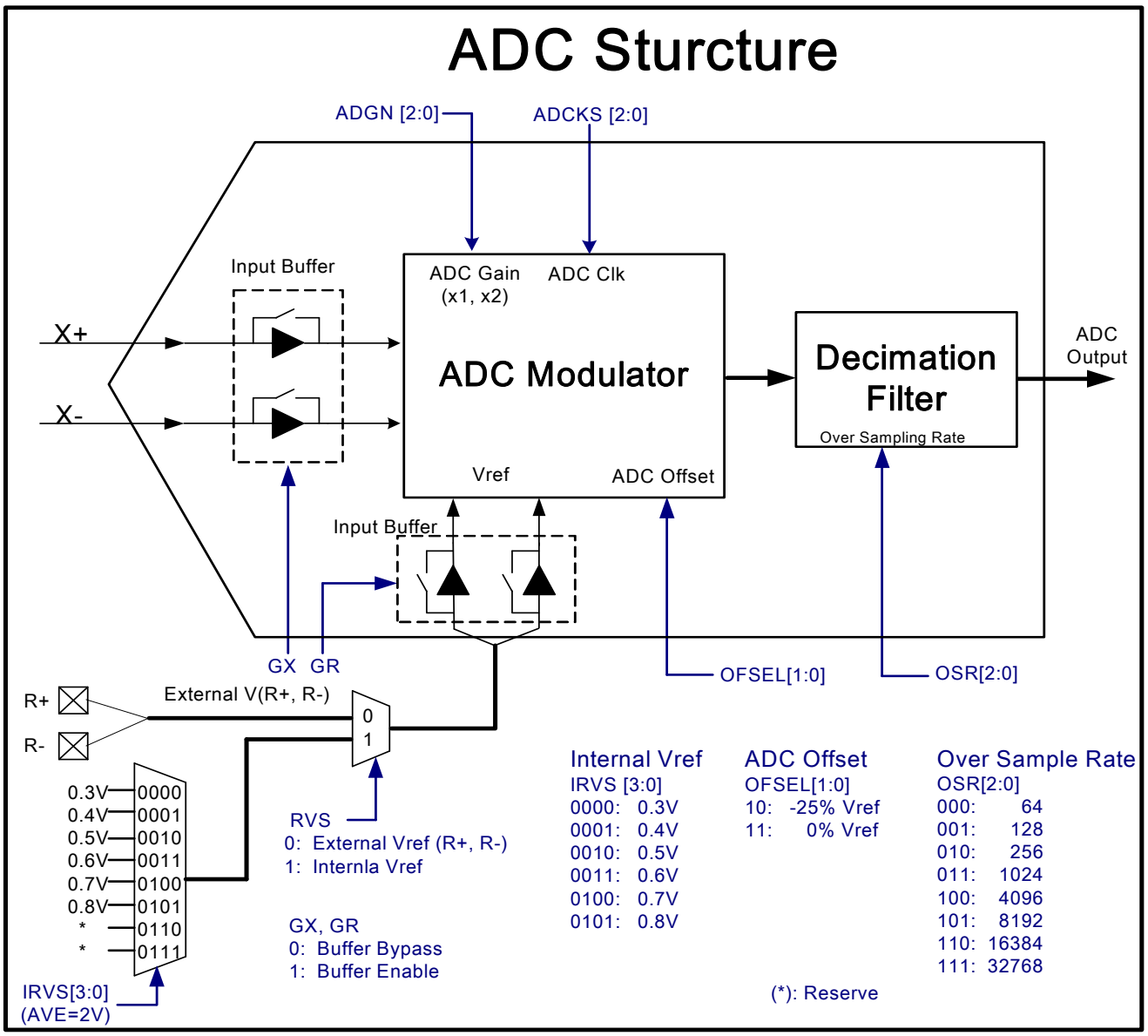
通过 V（TS）ADC 输出，可以得到温度信息和系统补偿。

- **注 1：每颗单片机的 V（TS）和温度曲线都是有差异的，建议在应用温度传感器时进行室内温度校准。**
- **注 2：温度传感器的典型温度参数是 3.23mV/C。每个芯片之间都是有差异的。**

SYM.	说明	参数	MIN.	TYP.	MAX.	UNIT
TR	温度传感器范围	AVDDR=2.4V, VDD = 3V.	-10	-	+70	℃
TS	温度传感器敏感度	AVDDR=2.4V, VDD = 3V.	3.52	3.2	2.88	mV/℃
ETS	温度传感器精度	1 个温度点 25℃	-10	-	+10	%
		2 个温度点	-1	-	+1	%

11.6 20 位ADC

SN8P2949 内置 20 位 ΔΣ ADC，同时带有抽取滤波器，输出速率的范围为 1Hz~3.9KHz，参考电压 Vref 由内部参考电压或者外部参考电压提供。AVE=2V 时，内部参考电压的范围为 0.3V~0.8V；外部参考电压由引脚 R+、R-提供，电压范围根据需要进行调整。内置的输入缓存器可在外部传感器输入信号直接与其连接时提供很高的输入阻抗。ADC 内置内部放大增益，将输入信号通过 PGIA 放大之后进一步放大，增益的选择选项为：1x 和 2x。



11.6.1 模拟输入和电压操作范围

ADC 和 PGIA 有 6 个模拟通道，包括 AI1+/AI1-、AI2+/AI2-、R+和 R-引脚。PGIA 的模拟输入 AI1+/AI1-、AI2+/AI2-连接到外部传感器输出信号，可组成差分模式（AI+到 AI-）或者单端模式（AI±到 ACM）。ADC 的外部 Vref 由输入引脚 R+和 R-的电压差决定。所有模拟输入的绝对电压范围被限制为 0.4V~1.4V，此外，PGIA 的输出 X+/X-必须保持在绝对电压范围内。

11.6.2 参考电压

ADC 有 2 个参考电压源（Vref），一个来自内部 Vref，另一个来自外部 Vref。通过寄存器 ADCM1 的 RVS 和 IRVS[3:0]选择 ADC 的参考电压源。RVS=1 时，选择内部 Vref 作为 ADC 参考电压源，内部 Vref 的值为 0.3V~0.8V，通过设置 IRVS[3:0]进行选择。RVS=0 时，选择外部 Vref 作为 ADC 参考电压源，外部 Vref 的值为引脚 R+和 R-的电压差。详见 ADCM1 寄存器说明。

11.6.3 输入缓存器

输入缓存器包括 ADC 信号输入缓存器和 ADC 外部参考输入缓存器 R+/R-，提供一个高阻抗的模拟输入信号，以减小 ADC 的输入电流，在进行灵敏测量时以避免负载的影响。PGIA 选择 1x 时，传感器的输出信号避开 PGIA，直接连接到 ADC 的输入端。在这种情况下，必须使能输入缓存器功能（GX=1）。若 ADC 选择外部 Vref，输入缓存器 R+/R-也必须使能（GR=1）。

11.6.4 ADC增益和偏移

ADC 内置内部增益选项，可选择范围为 x1，x2 和 x4，用于额外信号的放大。通过寄存器 ADCM1 的 ADGN[1:0]位选择 ADC 的增益。ADC 增益放大后的模拟信号可以通过减少或增加偏移量功能来调节偏移水平，以增加在称重应用下 ADC 的信号操作范围。ADC 的偏移功能由 ADCM2 寄存器的 OFSEL[1:0]位控制。下面显示了 ADC 输出码的计算。

ADC 输出码（差分模式）：

16bits:
$$\frac{[(AI+)-(AI-)]\times PGIA\times ADC\_Gain+V_{Offset}}{Vref}\times 2^{(16-1)} = +32767 \sim -32768$$

18bits:
$$\frac{[(AI+)-(AI-)]\times PGIA\times ADC\_Gain+V_{Offset}}{Vref}\times 2^{(18-1)} = +131071 \sim -131072$$

20bits:
$$\frac{[(AI+)-(AI-)]\times PGIA\times ADC\_Gain+V_{Offset}}{Vref}\times 2^{(20-1)} = +524287 \sim -524288$$

Voffset:	0, -1/4, -1/2, -3/4 x Vref
PGIA:	1x ~ 200x
ADC_Gain:	1x, 2x and 4x
Vref Source:	内部 Vref 或外部 Vref
Vref Range:	0.3V ~ 0.8V

✷ 注 1：增益选择 200*1 时，若 ADC Offset 功能选择-1/4*Vref，与没有 offset 功能进行比较，ADC ENOB 会掉落 0.1~0.3 位。
✷ 注 2：增益选择 200*1 时，若 ADC Offset 功能选择-1/2*Vref，与没有 offset 功能进行比较，ADC ENOB 会掉落 0.3~0.5 位。
✷ 注 3：增益选择 200*1 时，若 ADC Offset 功能选择-3/4*Vref，与没有 offset 功能进行比较，ADC ENOB 会掉落 0.6~0.8 位。

11.6.5 输出Word Rate

Delta-Sigma ADC 提供可变的输出 word rate，范围为 0.95Hz~3.9KHz，由 ADCKS[2:0]位和 OSR[2:0]位控制。低速输出 word rate 的 ADC 输出码要比快速的稳定。ADC 应用中，需要权衡 ADC 的输出 word rate 和稳定性（ENOB）。下表显示了 ADC 输出 word rate 的设置。

ADC 输出 word rate 列表：

ADCKS[2:0]	OSR [2:0]	ADC 时钟	WR	ADCKS[2:0]	OSR [2:0]	ADC Clock	WR
000	000	250KHz	3.9 kHz	001	000	125kHz	1.95KHz
000	001		1.95 kHz	001	001		976Hz
000	010		976 Hz	001	010		488Hz
000	011		244 Hz	001	011		122Hz
000	100		61 Hz	001	100		30.5Hz
000	101		30.5 Hz	001	101		15.2Hz
000	110		15.2 Hz	001	110		7.6Hz
000	111		7.6 Hz	001	111		3.8Hz

ADCKS[2:0]	OSR [2:0]	ADC 时钟	WR	ADCKS[2:0]	OSR [2:0]	ADC Clock	WR
010	000	62.5kHz	976Hz	011	000	31.25kHz	488Hz
010	001		488Hz	011	001		244Hz
010	010		244Hz	011	010		122Hz
010	011		61Hz	011	011		30.5Hz
010	100		15.2Hz	011	100		7.6Hz
010	101		7.6Hz	011	101		3.8Hz
010	110		3.8Hz	011	110		1.9Hz
010	111		1.9Hz	011	111		0.95Hz

11.6.6 ADCM1- ADC模式寄存器

093H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCM1	RVS	IRVS3	IRVS2	IRVS1	IRVS0	ADGN1	ADGN0	ADCENB
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	1	0	1	0	1	0	0	0

Bit0 **ADCENB**：ADC 功能控制位。
0 = 禁止 ADC；
1 = 使能 ADC。

Bit[2:1] **ADGN[1:0]**：ADC 增益选择位。

ADGN[1:0]	ADC Gain
00	X1
01	X2
10	X4
11	保留

Bit[6:3] **IRVS[3:0]**：ADC 内部参考电压选择位。

IRVS[3:0]	**Vref source**	
	AVE 1.5V	**AVE 2.0V**
0000	0.225V	0.3V
0001	0.300V	0.4V
0010	0.375V	0.5V
0011	0.450V	0.6V
0100	0.525V	0.7V
0101	0.600V	0.8V
0110~1111	Reserved	Reserved

Bit7 **RVS**：ADC 参考电压内部/外部选择位。
0 = 外部 R+，R-；
1 = 内部 AVE 或 AVDDR。

✷ **注 1：ADC 参考电压（Vref）的操作范围为 0.3V~0.8V。**
✷ **注 2：Vref（Int）代表 ADC 的参考电压由内部提供；Vref（Ext）代表 ADC 的参考电压由外部 R+/R-提供。**

11.6.7 ADCM2- ADC模式寄存器

094H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCM2	-	OSR2	OSR1	OSR0	-	OFSEL1	OFSEL0	DRDY
R/W	-	R/W	R/W	R/W	-	R/W	R/W	R/W
复位后	-	1	1	1	-	1	1	0

Bit0 **DRDY**：ADC 转换准备位。
1 = ADC 输出（升级）新的转换数据到 ADCDH、ADCDM 和 ADCDL 中；
0 = ADCDH，ADCDM 和 ADCDL 的转换数据还未准备好。

Bit[2:1] **OFSEL[1:0]**：ADC 偏移选择位。

OFSEL[1:0]	ADC Offset %
00	-75% Vref
01	-50% Vref
10	-25% Vref
11	0% Vref

Bit[6:4] **OSR [2:0]**：ADC OSR 选择位。

OSR [2:0]	OSR
000	64
001	128
010	256
011	1024
100	4096
101	8192
110	16384
111	32768

11.6.8 ADCM3- ADC模式寄存器

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCM3	-	-	-	ACHPENB	ADCKINV	ADCKS2	ADCKS1	ADCKS0
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	1	0	0	0	0

Bit[2:0]: **ADCKS[2:0]**：ADC时钟选择控制位。

ADCKS[2:0]	ADC 时钟
000	250kHz
001	125KHz
010	62.5KHz
011	31.25KHz
1xx	Reserved

Bit[3]: **ADCKINV**：ADC 时钟反向控制位（请一直置 0）。

Bit[4]: **ACHPENB**：ADC Chopper控制位（请一直置1）。

- **注 1：ADC 输出 Word Rate (WR) = ADC 时钟/ OSR；**
- **注 2：调节 ADC 时钟（ADCKS[2:0]）和 OSR 可以获得合适的 ADC 输出 word rate；**
- **注 3：在高分辨率的应用下，建议设置 OSR 为最大值 32768；**
- **注 4：在获得 ADC 数据后将 DRDY 位清零，否则改为会一直保持为高；**
- **注 5：在使能 ADC 后的第三个数据时，ADC 输出稳定。每个 1/WR 后的第一、第二、第三、第四、第五……ADC 输出数据稳定。**

11.6.9 ADC数据寄存器

097H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCDH	ADCB23	ADCB22	ADCB21	ADCB20	ADCB19	ADCB18	ADCB17	ADCB16
R/W	R	R	R	R	R	R	R	R
复位后	0	0	0	0	0	0	0	0

098H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCDM	ADCB15	ADCB14	ADCB13	ADCB12	ADCB11	ADCB10	ADCB09	ADCB08
R/W	R	R	R	R	R	R	R	R
复位后	0	0	0	0	0	0	0	0

099H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCDL	ADCB07	ADCB06	ADCB05	ADCB04	ADCB03	ADCB02	ADCB01	ADCB00
R/W	R	R	R	R	R	R	R	R
复位后	0	0	0	0	0	0	0	0

ADCDH [7:0]：输出 ADC 的高字节。
ADCDM [7:0]：输出 ADC 的中间字节。
ADCDL [7:0]：输出 ADC 的低字节。

ADC 转换数据 (2's compliment, Hexadecimal)	十进制数据
7FFFFH	524287
…	…
40000H	262144
…	…
10000H	65536
…	…
00002H	2
00001H	1
00000H	0
FFFFH	-1
FFFEH	-2
…	…
F0000H	-65536
…	…
C0000H	-262144
…	…
80000H	-524288

✷ 注 1：ADCDH [7:0]，ADCDM [7:0]和 ADCDL [7:0]是只读寄存器。
✷ 注 2：16 位 ADC：使用 ADCDH 和 ADCDM（ADCB23~ADCB08）寄存器；18 位 ADC：使用 ADCDH、ADCDM 和 ADCDL（ADCB23~ADCB06）；20 位 ADC：使用 ADCDH、ADCDM 和 ADCDL（ADCB23~ADCB04）；24 位 ADC：使用 ADCDH、ADCDM 和 ADCDL（ADCB23~ADCB00）。
✷ 注 3：ADC 的转换数据存储在 ADCDH，ADCDM 和 ADCDL 中，符号位为数字格式，ADCB23 是 ADC 数据的符号位，ADCB23=0 表示数据是正值，ADCB23=1 则表示数据是负值。
✷ 注 4：ADC 数据满量程的正值为 7FFFFH。
✷ 注 5：ADC 数据满量程的负值为 80000H。
✷ 注 6：由于 ADC 设计限制，ADC 的线性范围为+121071~-131072（18 位）（+0.9*Vref~-0.9*Vref），ADC 的最大输出值必须在此范围内。

下表显示了 SN8P2949 ADC 在不同输出速率和增益设置下的杂讯以及 ENOB（RMS 和 peak-to-peak）。这些数字是在差分输入短路，1024 笔数据的条件下测量得到的。

2949 ADC Performance ENOB (Noise Free Bit) vs. Output WR and Gain								
Gain	ADC Output Word Rate							
	3.9 kHz	2.0 kHz	977 Hz	244 Hz	61.0 Hz	30.5 Hz	15.3 Hz	7.6 Hz
1 x 1*	12.4	14.1	14.9	16.1	17.1	17.6	18.1	18.5
1 x 2*	12.3	13.4	14.3	15.4	16.5	17.0	17.6	18.2
12.5 x 1	12.3	13.4	14.2	15.3	16.2	16.8	17.3	17.6
50 x 1	12.1	13.2	13.6	14.8	15.7	16.1	16.7	17.1
100 x 1	11.6	12.5	13.0	14.1	15.0	15.4	16.0	16.4
100 x 2	10.9	11.3	12.0	13.1	14.1	14.6	15.2	15.5
200 x 1	11.0	11.6	12.2	13.2	14.3	14.8	15.2	15.6

*关闭缓存器（**GX=0，GR=0**）

测试条件：ADC 250KHz，输入短路，Vref=0.84V，Gain=PGIA×ADC，收集 1024 ADC date。

（1）Noise Free 分辨率=Log2（满量程范围/Peak-Peak Noise）

Where Full Scale Range=2×Vref/Gain（如：Vref=0.84V，Gain=200x）

（2）有效分辨率=Log2（满量程范围/RMS_Noise）

（3）RMS Noise=σ×LSB_Resolution

Where LSB_Resolution=满量程范围/2^Bit，Bit=20

σ =1024 ADC 输出数据的标准误差。

（4）Peak-Peak Noise=6.6×RMS Noise，或 code 变化范围×LSB_Resolution

Where code 变化范围=ADC counts max-min of 1024 data。

2949 ADC Peak to Peak Noise(uV) vs. Output WR and Gain								
Gain	ADC Output Word Rate							
	3.9 kHz	2.0 kHz	977 Hz	244 Hz	61.0 Hz	30.5 Hz	15.3 Hz	7.6 Hz
1 x 1	310.8	95.67	54.95	23.92	11.96	8.456	5.980	4.532
1 x 2	166.6	77.71	41.64	19.43	9.063	6.409	4.228	2.790
12.5 x 1	26.65	12.43	7.141	3.332	1.785	1.178	0.833	0.677
50 x 1	7.654	3.571	2.706	1.178	0.631	0.478	0.316	0.239
100 x 1	5.412	2.900	2.051	0.957	0.513	0.389	0.256	0.194
100 x 2	4.396	3.332	2.051	0.957	0.478	0.338	0.223	0.181
200 x 1	4.102	2.706	1.785	0.893	0.416	0.294	0.223	0.169

2949 ADC RMS Noise (uV) with Output WR and Gain								
Gain	ADC Output Word Rate							
	3.9 kHz	2.0 kHz	977 Hz	244 Hz	61.0 Hz	30.5 Hz	15.3 Hz	7.6 Hz
1 x 1	47.10	14.50	8.326	3.624	1.812	1.281	0.906	0.687
1 x 2	25.24	11.77	6.310	2.944	1.373	0.971	0.641	0.423
12.5 x 1	4.038	1.884	1.082	0.505	0.271	0.178	0.126	0.103
50 x 1	1.160	0.541	0.410	0.178	0.096	0.072	0.048	0.036
100 x 1	0.820	0.439	0.311	0.145	0.078	0.059	0.039	0.029
100 x 2	0.666	0.505	0.311	0.145	0.072	0.051	0.034	0.027
200 x 1	0.621	0.410	0.271	0.135	0.063	0.045	0.034	0.026

➢ **例：Regulator，PGIA 和 ADC 设置（Fhosc = IHRC 4MHz）。**

```
@CPREG_Init:      B0BSET      FBGRENB          ; 使能 Band Gap 参考电压。

@ACM_Enable:      B0BSET      FACMSEL          ; 设置 ACM 输出 1V。
                  B0BSET      FACMENB          ; 使能 ACM 电压。

@AVE_Enable:      B0BSET      FAVESEL          ; 设置 AVE+输出 2.0V。
                  B0BSET      FAVENB           ; 使能 AVE+电压。

@AVDDR_Enable:    B0BSET      FAVDDRSEL        ; 使能 AVDDR 电压为 2.4V。
                  B0BSET      FAVDDRENB        ;

@PGIA_Init:       MOV         A, #00000110B
                  B0MOV       AMPM1, A         ; PGIA 差分通道（AI1+, AI1-）和 PGIA Gain x 200。
                  MOV         A, #00101000B
                  B0MOV       AMPM2, A         ; 设置单元增益缓存器关闭，PGIA Chopper 31.25KHz。
                  B0BSET      FAMPENB          ; 使能 PGIA 功能。
                                               ; V (X+, X-) 输出= V (AI1+, AI1-) x 200。

@ADC_Init:        MOV         A, #10101000B
                  B0MOV       ADCM1, A         ; ADC 参考电压为内部 0.8V，ADC_Gain = 1x。
                  MOV         A, #01110110B
                  B0MOV       ADCM2, A         ; 设置 OSR=32768，offset=0V。
                  MOV         A, #00010000B
                  B0MOV       ADCM3, A         ; 设置 ADC 时钟为 250KHz。
                  B0BSET      FADCENB          ; 使能 ADC 功能。

@ADC_Wait:        B0BTS1      FDRDY            ; 检测 ADC 是否有输出新值。
                  JMP         @ADC_Wait        ; 等 DRDY 位置 1。
                                               ; 输出 ADC 转换 word。
@ADC_Read:        B0BCLR      FDRDY
                  B0MOV       A, ADCDH
                  B0MOV       Data_H_Buf, A    ; 存储 ADC 的高字节到数据缓存器中。
                  B0MOV       A, ADCDM
                  B0MOV       Data_M_Buf, A    ; 存储 ADC 的中间字节到数据缓存器中。
```

✷ **注 1：请先设置 ADC 的相关寄存器，然后再使能 ADC 功能。**
✷ **注 2：使能 ADC 之前，请先设置模拟功能（regulators，PGIA 和 ADC），再延时 300us 等待所有功能稳定。**

➢ **例：VDD/VLCD 电压检测。**

```
@CPREG_Init:        B0BSET      FBGRENB         ; 使能 Band Gap 参考电压。
@ACM_Enable:        B0BSET      FACMSEL         ; 设置 ACM 输出 1V。
                    B0BSET      FACMENB         ; 使能 ACM 电压。
@AVE_Enable:        B0BSET      FAVESEL         ; 设置 AVE+输出 2.0V。
                    B0BSET      FAVENB          ; 使能 AVE+电压。
@AVDDR_Enable:      B0BSET      FAVDDRSEL       ; 使能 AVDDR 电压为 2.4V。
                    B0BSET      FAVDDRENB

@VDD_Detection:
@PGIA_Init:         MOV         A, #10011110B
                    B0MOV       AMPM1, A        ; PGIA 电压检测和 PGIA Gain x 1。
                    MOV         A, #00010110B   ; 设置 VDD 检测功能。
                    B0MOV       AMPM2, A        ; 设置单元增益缓存器关闭，PGIA chopper 31.25KHz。
                    B0BSET      FAMPENB         ; 使能 PGIA 功能。
                                                ; V (X+, X-)输出= 1/8 VDD × 1。
@ADC_Init:          MOV         A, #10101000B
                    B0MOV       ADCM1, A        ; ADC 参考电压为内部 0.8V，ADC_Gain = 1x。
                    MOV         A, #01110110B
                    B0MOV       ADCM2, A        ; 设置 OSR=32768，offset=0V。
                    MOV         A, #00010000B   ;
                    B0MOV       ADCM3, A        ; 设置 ADC 时钟为 250KHz，
                    B0BSET      FADCENB         ; 使能 ADC 功能。

@ADC_Wait:          B0BTS1      FDRDY           ; 检查 ADC 是否有输出新值。
                    JMP         @ADC_Wait       ; 等待 DRDY 位置 1。
@ADC_Read:          B0BCLR      FDRDY           ; 输出 ADC 转换 word。
                    B0MOV       A, ADCDH
                    B0MOV       Data_H_Buf, A   ; 存储 ADC 的高字节到数据缓存器中。
                    B0MOV       A, ADCDM
                    B0MOV       Data_M_Buf, A   ; 存储 ADC 的中间字节到数据缓存器中。
                    …
                    …
                    …
@VLCD_Detection:
@PGIA_Init:         MOV         A, #10011110B
                    B0MOV       AMPM1, A        ; PGIA 电压检测和 PGIA Gain x 1。
                    MOV         A, #00010111B   ; 设置 VLCD 检测功能。
                    B0MOV       AMPM2, A        ; 设置单元增益缓存器关闭，PGIA chopper 31.25KHz。
                    B0BSET      FAMPENB         ; 使能 PGIA 功能。
                                                ; V (X+, X-)输出 = 1/8 VLCD x 1。
@ADC_Init:          MOV         A, #10101000B
                    B0MOV       ADCM1, A        ; ADC 参考电压为内部 0.8V，ADC_Gain = 1x。
                    MOV         A, #01110110B
                    B0MOV       ADCM2, A        ; 设置 OSR=32768，offset=0V。
                    MOV         A, #00010000B
                    B0MOV       ADCM3, A        ; 设置 ADC 时钟为 250KHz。
                    B0BSET      FADCENB         ; 使能 ADC 功能。

@ADC_Wait:          B0BTS1      FDRDY           ; 检查 ADC 是否有输出新值。
                    JMP         @ADC_Wait       ; 等待 DRDY 位置 1。
@ADC_Read:          B0BCLR      FDRDY           ; 输出 ADC 转换 word。
                    B0MOV       A, ADCDH
                    B0MOV       Data_H_Buf, A   ; 存储 ADC 的高字节到数据缓存器中。
                    B0MOV       A, ADCDM
                    B0MOV       Data_M_Buf, A   ; 存储 ADC 的中间字节到数据缓存器中。
```

➢ **例：快速 ADC 转换 Rate 设置。**

```
@CPREG_Init:        B0BSET      FBGRENB         ; 使能 Band Gap 参考电压。
@ACM_Enable:        B0BSET      FACMSEL         ; 设置 ACM 输出 1V。
                    B0BSET      FACMENB         ; 使能 ACM 电压。
@AVE_Enable:        B0BSET      FAVESEL         ; 设置 AVE+输出 2.0V。
                    B0BSET      FAVENB          ; 使能 AVE+电压。
@AVDDR_Enable:      B0BSET      FAVDDRSEL       ; 使能 AVDDR 电压为 2.4V。
                    B0BSET      FAVDDRENB

@VDD_Detection:
@PGIA_Init:         MOV         A, #00001110B
                    B0MOV       AMPM1, A        ; PGIA 差分通道（AI1+, AI1-）和 PGIA Gain x 1。
                    MOV         A, #00010100B   ; 设置 VDD 检测功能。
                    B0MOV       AMPM2, A        ; 设置单元增益缓存器关闭，PGIA chopper 31.25KHz。
                    B0BSET      FAMPENB         ; 使能 PGIA 功能。
                                                ; V (X+, X-)输出= V (AI1+, AI1-) x 1。
@ADC_Init:          MOV         A, #10101000B
                    B0MOV       ADCM1, A        ; ADC 参考电压为内部 0.8V，ADC_Gain = 1x。
                    MOV         A, #00100110B
                    B0MOV       ADCM2, A        ; 设置 ADC 时钟为 250KHz，OSR=32768，offset=0V，WR=~1KHz。
                    MOV         A, #00010000B
                    B0MOV       ADCM3, A
                    B0BSET      FADCENB         ; 使能 ADC 功能。
                    Call        Wait_500uS      ; 等待 500us 使 Regulators 和模拟功能稳定。

@ADC_Wait:          B0BTS1      FDRDY           ; 检查 ADC 是否有输出新值。
                    JMP         @ADC_Wait       ; 等待 DRDY 位置 1。
@ADC_Read:          B0BCLR      FDRDY           ; 输出 ADC 转换 word。
                    B0MOV       A, ADCDH
                    B0MOV       Data_H_Buf, A   ; 存储 ADC 的高字节到数据缓存器中。
                    B0MOV       A, ADCDM
                    B0MOV       Data_M_Buf, A   ; 存储 ADC 的中间字节到数据缓存器中。
```

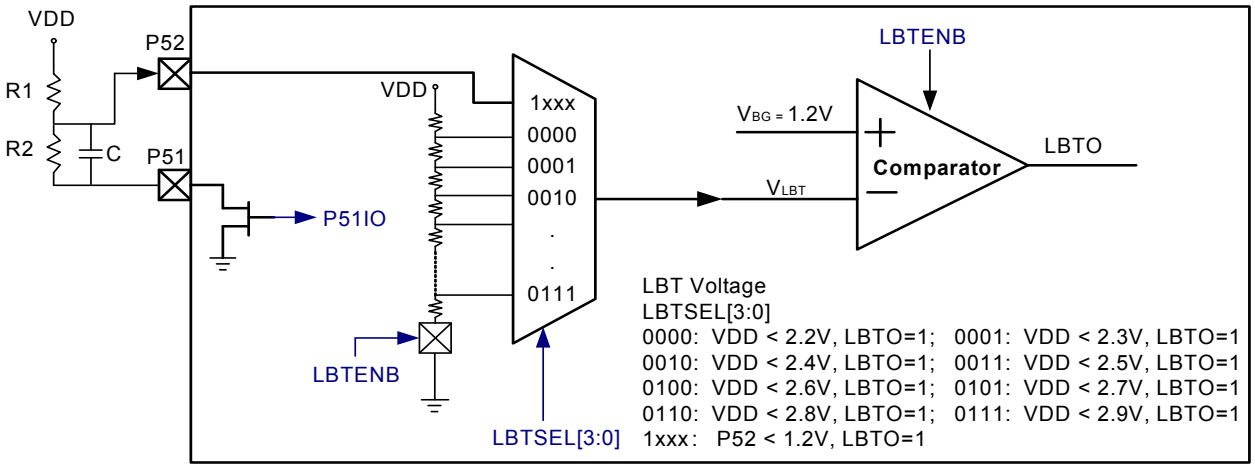
➢ 例：绿色模式和睡眠模式设置。

```
Green_Mode_1:   // 由 ADC 准备、T0 溢出和 P0 电平变换唤醒。
                MOV         A, #11110000B
                B0MOV       T0M, A          ; T0 溢出，每 512us 唤醒。（Fcpu=1MIP）
                MOV         A, #00100110B
                B0MOV       ADCM2, A        ; 设置 OSR=256， WR=976Hz。
                MOV         A, #00010000B
                B0MOV       ADCM3, A        ; 设置 ADC 时钟为 250KHz。
                B0BSET      FADCENB         ; 使能 ADC，每 1024us 唤醒系统。
                GreenMode                   ; 系统进入绿色模式（宏指令）。
Green_Mode_2:   // 由 T0 溢出和 P0 电平变换唤醒（电流的典型值为 3uA）。
                B0BCLR      FADCENB         ; 禁止 ADC。
                B0BCLR      FAMPENB         ; 禁止 PGIA。
                B0BCLR      FVLEDENB        ; 禁止 VLED。
                B0BCLR      FCPRENB         ; 禁止 LED charge pump。
                B0BCLR      FLBTENB         ; 禁止电池低电压检测功能。
                B0BCLR      FDTENB          ; 禁止 VDD/VLCD 检测功能。
                B0BCLR      FAVDDRENB       ; 禁止 AVDDR。
                B0BCLR      FAVENB          ; 禁止 AVE。
                B0BCLR      FACMENB         ; 禁止 ACM。
                B0BCLR      FLCDPENB        ; 禁止 C-Type LCD charge pump。.
                B0BCLR      FLCDENB         ; 禁止 LCD 显示。
                B0BCLR      FBGRENB         ; 禁止 Band Gap 电压。
                B0BSET      FLCKMD          ; 进入低速模式。
                B0BSET      FSTPHX          ; 停止高速时钟（IHRC）。
                MOV         A, #11000000
                B0MOV       T0M, A          ; T0 溢出，绿色模式唤醒。
                GreenMode                   ; 系统进入绿色模式。

Sleep_Mode:     //由 P0 电平变换唤醒。
                B0BCLR      FADCENB         ; 禁止 ADC。
                B0BCLR      FAMPENB         ; 禁止 PGIA。
                B0BCLR      FVLEDENB        ; 禁止 VLED。
                B0BCLR      FCPRENB         ; 禁止 LED charge pump。
                B0BCLR      FLBTENB         ; 禁止电池低电压检测功能。
                B0BCLR      FDTENB          ; 禁止 VDD/VLCD 检测功能。
                B0BCLR      FAVDDRENB       ; 禁止 AVDDR。
                B0BCLR      FAVENB          ; 禁止 AVE。
                B0BCLR      FACMENB         ; 禁止 ACM。
                B0BCLR      FLCDPENB        ; 禁止 C-Type LCD charge pump。.
                B0BCLR      FLCDENB         ; 禁止 LCD 显示。
                B0BCLR      FBGRENB         ; 禁止 Band Gap 电压。
                SleepMode                   ; 进入低速模式。
```

✷ **注 1：使能 ADC 功能之前，先设置 ADC 的相关寄存器。**
✷ **注 2：使能 ADC 功能之前，先设置模拟功能（Regulators，PGIA 和 ADC），然后延时 500us 等待所有功能稳定。**
✷ **注 3：在快速 ADC 转换应用中，第三个 ADC 数据对应用有效。**
✷ **注 4：在使能 ADC 的条件下，ADC 通道切换后的第二个 ADC 是有效的。**
✷ **注 5：为了增加 ADC 转换的精确度，建议求取几次 ADC row Data 的平均值。**

11.7 LBTM：电池低电压检测

SN8P2949 提供 2 种方式测量 VDD 电压：一种是通过选择 ADC 参考电压，这种方法比较精确，但是比较费时且比较复杂；另外一种是内置一个电压比较器，通过内部或者外部输入通道去检测 VDD 的电压电平，内部电压电平共有 8 个选择，2.2V/2.2V/2.4V/2.5V/2.6V/2.7V/2.8V/2.9V，均可用来进行电池低电压检测，或者是通过 VDD 分压，连接到 P5.2，从 LBT0 位显示出 LBT 的状态。



11.7.1 LBTM：电池低电压检测寄存器

095H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LBTM	-	P51IO	LBTSEL3	LBTSEL2	LBTSEL1	LBTSEL0	LBTO	LBTENB
R/W	-	R/W	R/W	R/W	R/W	R/W	R	R/W
复位后	-	0	0	0	0	0	0	0

Bit0 **LBTENB**：电池低电压检测模式控制位。
0 = 禁止电池低电压检测功能；
1 = 使能电池低电压检测功能。

Bit1 **LBTO**：电池低电压检测输出位。
0 = LBT 电压（VLBT）高于 Band Gap 参考电压 1.2V；
1 = LBT 电压（VLBT）低于 Band Gap 参考电压 1.2V。

Bit[5:2] **LBTSEL[3:0]**：电池低电压检测极限电压选择位。

LBTENB	LBTSEL [3:0]	LBTO = 1	Note
0	-	-	禁止 LBT 功能
1	0000	VDD < 2.2V	内部输入
1	0001	VDD < 2.3V	内部输入
1	0010	VDD < 2.4V	内部输入
1	0011	VDD < 2.5V	内部输入
1	0100	VDD < 2.6V	内部输入
1	0101	VDD < 2.7V	内部输入
1	0110	VDD < 2.8V	内部输入
1	0111	VDD < 2.9V	内部输入
1	1xxx	P52 < 1.2V, LBTO=1	外部输入

Bit6 **P51IO**：P5.1 输入/LBT 功能控制位。
0 = 设置 P51 为输入模式；
1 = 设置 P51 为 LBT 功能，P51 直接接地。

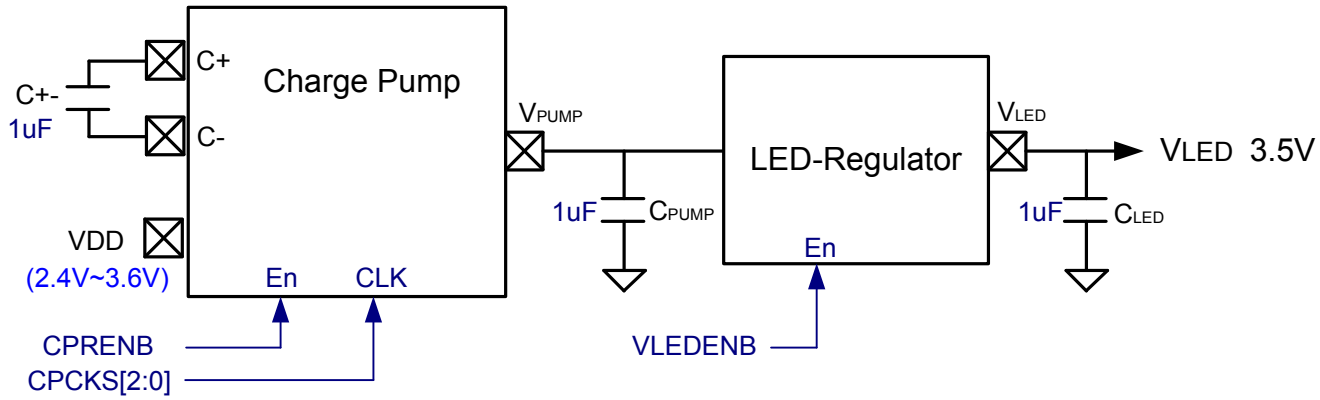
外部输入（P52）LBT 功能在 ICE 仿真模式下无效。

电池低电压	R1	R2	LBTO=1
2.3V	470kΩ	530kΩ	VDD<2.3V
2.8V	620kΩ	470kΩ	VDD<2.8V

- **注 1：在一段时间内至少连续检测到 10 次 LBTO = 1，如 20ms 或更长的时间以确认电池低电压信号稳定。**
- **注 2：LBT 外部输入 P52 和 P51IO 功能在 ICE 仿真模式下有效。**
- **注 3：IO 输入电压必须低于 VDD。**

11.8 Charge Pump和LED Regulator

SN8P2949 提供固定的 Regulator 输出电压，用于驱动 LED。LED-Regulator 的输出电压为 3.5V，来自 VDD（2.4V~3.6V）。当 DVDD 为 2.4V 时，最大驱动电流超过 8mA。下图显示了 Charge Pump 和 LED Regulator 系统。



Charge Pump 输出电压 Vpump = 2 x VDD，VDD 范围为 2.4V~3.6V。
电容要求：Cpump = C+- = CLED = 1uF。
LED-Regulator 输出电压为 3.5V。
在 VDD = 2.4V 时，LED-Regulator 最大驱动电流为 8mA。

CPM 寄存器初始值 = xxx0 0000

09AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LEDCPM	-	-	-	VLEDENB	CPCKS2	CPCKS1	CPCKS0	CPRENB
R/W	-	-	-	R/W	R/W	R/W	R	R/W
After Reset	-	-	-	0	0	0	0	0

Bit0 **CPRENB**：LED Charge Pump 使能位，用于 LVED 电源。
0 = 禁止；
1 = 使能。

Bit[3:1] **CPCKS[2:0]**：Charge Pump 时钟选择位。

CPCKS[2:0]	Charge pump 时钟
0xx	保留
100	15.625 KHz
101	31.25 KHz
110	62.5 KHz
111	125 KHz

Bit3 **LEDENB**：VLED 输出 3.5V 使能位。
0 = 禁止 VLCD 输出；
1 = 使能 Charge Pump，仅在普通模式下有效。

- **注 1：VLED 启动建议：使能 LEDENB 之前使能 CPRENB 5ms。**
- **注 2：Charge Pump LED-Regulatro 功能在普通模式和低速模式下有效（STPHX=0）。**
- **注 3：使能 Charge Pump 时，VLED 的电流功耗翻倍。**
- **注 4：CPRENB = 0 时，Vpump = VDD。**
- **注 5：若 VLED 输出电流超过 8mA，强烈建议选择 125KHz。**

SYM.	说明	参数	MIN.	TYP.	MAX.	UNIT
VLED	VLED 输出电压	VDD = 3V.	3.3	3.5	3.7	V
ILED	VLED 输出电流能力	VDD = 2.4V, Pump Clock = 125KHz.	8	-	-	mA

11.9 模拟电路的设置和应用

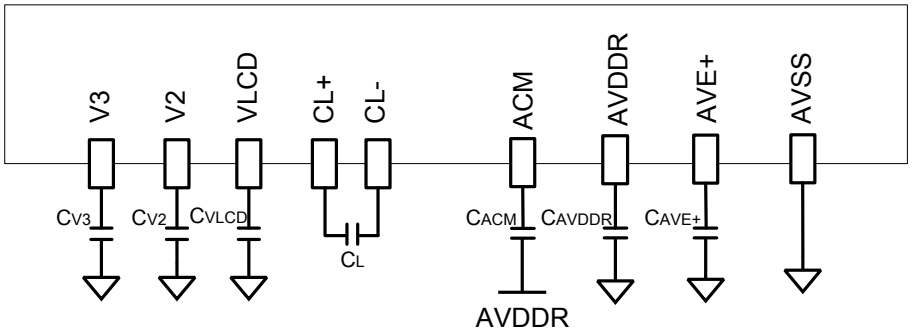
SN8P2949 主要应用于 DC 测量，如体重秤、压力测量等。下表列出了不同应用方案的建议，单片机的电源分别由 CR2032 电池、AA/AAA 干电池或者外部稳压电路供电。

电容列表：

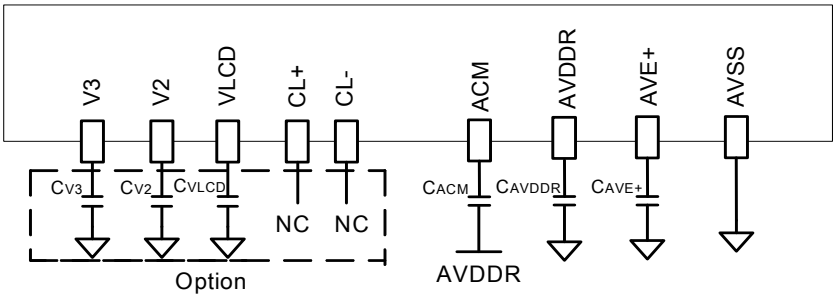
电源类型	AI+	AI-	R+/ R-	ACM	AVDDR	AVE+	CL+/CL-	C+/C-	VPUMP	VLED	AVDD	DVDD
	CAI+	CAI-	CR	CACM	CAVDDR	CAVE+	CL	CC	Cpump	CLED	CAVDD	CDVDD
CR2032 (2.4~3V)	0.01uF	0.01uF	0.1uF	0.1uF	0.47uF	0.47uF	0.1uF	1uF	1uF	1uF	10uF, 0.1uF	10uF, 0.1uF
AA/AAA Bat.(2.4~3V)	0.01uF	0.01uF	0.1uF	0.1uF	1uF	1uF	0.1uF	1uF	1uF	1uF	1uF, 0.1uF	1uF, 0.1uF

✷ 注 1：R 型 LCD 驱动模式下，C_L 不能连接到单片机上。
✷ 注 2：CR2032 电池应用时，在低电压电池应用条件下，AVE 和 AVDDR 需连接一个 0.47uf 的电容来应对 VDD 跌落。
✷ 注 3：AVE+和 AVEER 连接 0.47uF 的电容时，AVE+的最大输出电流限制在 3mA 以内。
✷ 注 4：当单片机的 VDD 电源直接由 AA/AAA 干电池提供时，在低电压电池应用条件下，VDD 需连接一个 1uf 的电容来应对 VDD 不跌落的情形。

- **VDD=2.4V ~ 3.6V 模拟电路电容连接示意图（C 型 LCD 驱动）：**

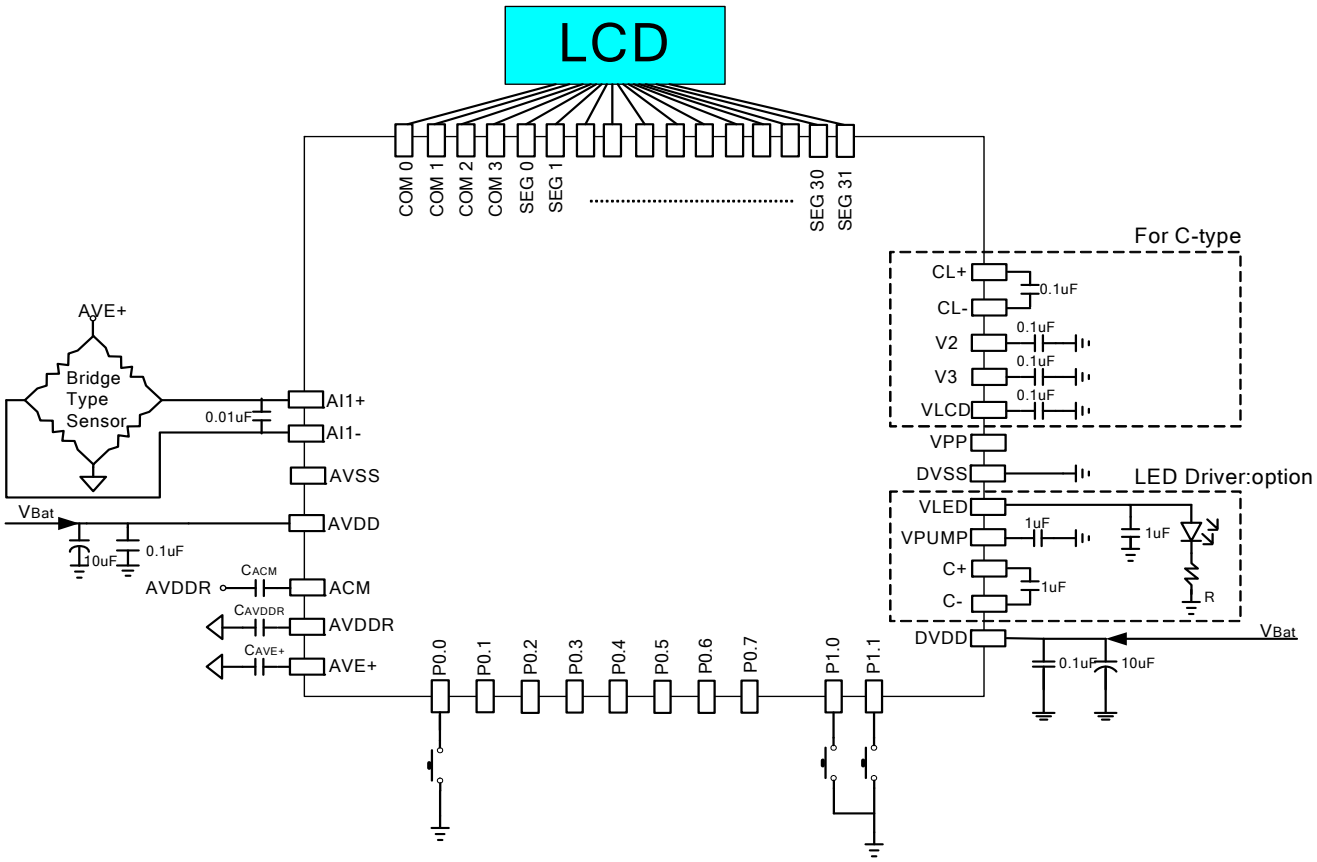


- **VDD=2.4V ~ 3.6V 模拟电路电容连接示意图（R 型 LCD 驱动）：**



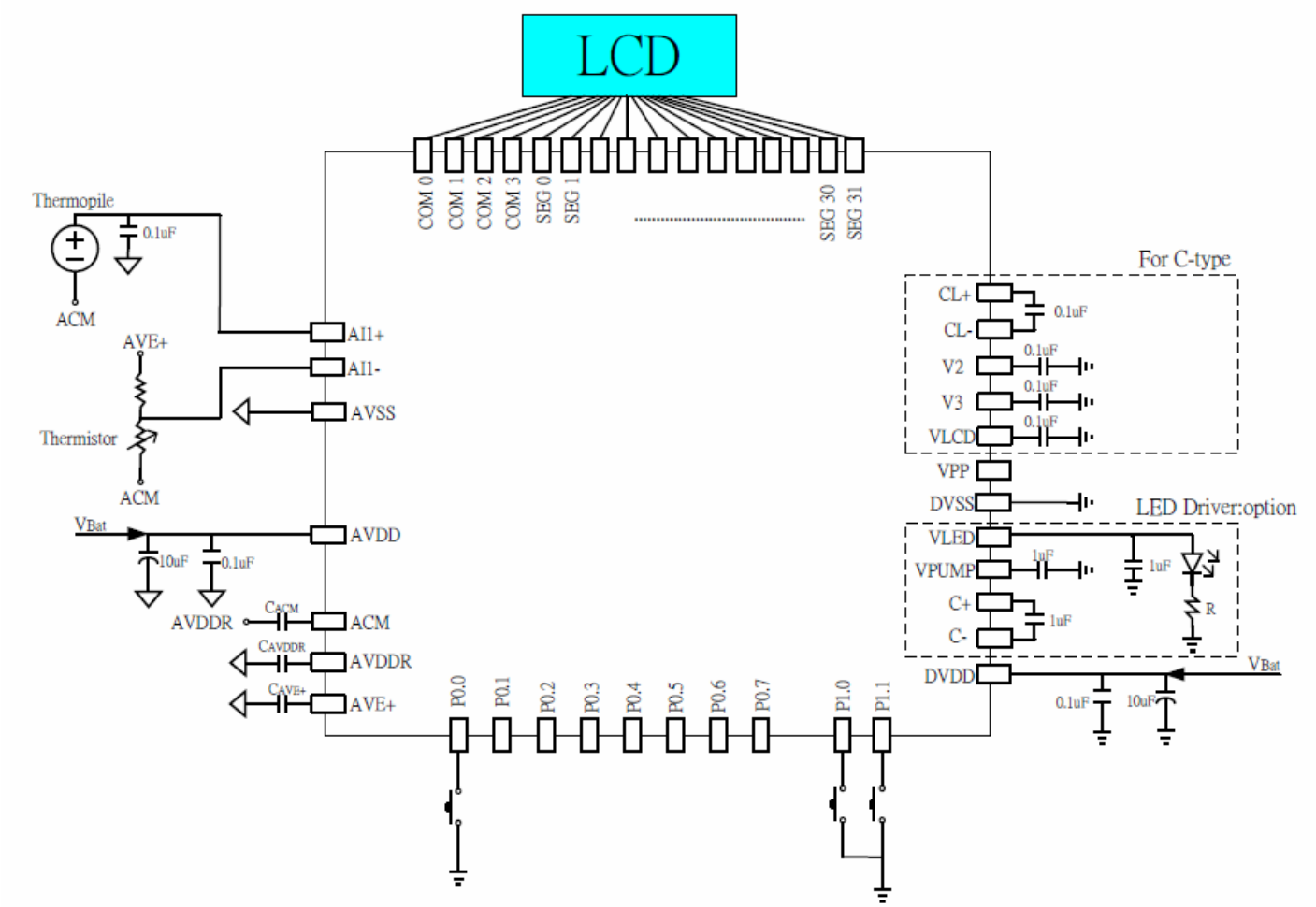
12 应用电路

12.1 电子秤（Load Cell）应用电路



✷ 注：DVDD/AVDD 电容要尽可能地靠近 ICE 的引脚。

12.2 温度计应用电路



✷ 注：DVDD/AVDD 电容要尽可能地靠近 ICE 的引脚。

13 指令集

指令	指令格式		指令说明	C	DC	Z	周期
MOVE	MOV	A,M	A ← M	-	-	√	1
	MOV	M,A	M ← A	-	-	-	1
	B0MOV	A,M	A ← M (bnak 0)	-	-	√	1
	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I，（M 指工作寄存器 R、Y、Z、RBANK 和 PFLAG 等。）	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]	-	-	-	2
ARITHMETIC	ADC	A,M	A ← A + M + C，如果产生进位，则 C=1，否则 C=0。	√	√	√	1
	ADC	M,A	M ← A + M + C，如果产生进位，则 C=1，否则 C=0。	√	√	√	1+N
	ADD	A,M	A ← A + M，如果产生进位，则 C=1，否则 C=0。	√	√	√	1
	ADD	M,A	M ← A + M，如果产生进位，则 C=1，否则 C=0。	√	√	√	1+N
	B0ADD	M,A	M (bank 0) ← M (bank 0) + A，如果产生进位，则 C=1，否则 C=0。	√	√	√	1+N
	ADD	A,I	A ← A + I，如果产生进位，则 C=1，否则 C=0。	√	√	√	1
	SBC	A,M	A ← A - M - /C，如果产生借位，则 C=0，否则 C=1。	√	√	√	1
	SBC	M,A	M ← A - M - /C, if occur borrow, then C=0, else C=1	√	√	√	1+N
	SUB	A,M	A ← A - M，如果产生借位，则 C=0，否则 C=1。	√	√	√	1
	SUB	M,A	M ← A - M，如果产生借位，则 C=0，否则 C=1。	√	√	√	1+N
	SUB	A,I	A ← A - I，如果产生借位，则 C=0，否则 C=1。	√	√	√	1
	DAA		将 ACC 中的数据由十六进制转换成十进制格式。	√	-	-	1
	MUL	A,M	R,A ← A * M，结果的 LB 存储在 ACC 中，HB 存入 R 寄存器中，ACC 控制 ZF。	-	-	√	2
LOGIC	AND	A,M	A ← A 与 M	-	-	√	1
	AND	M,A	M ← A 与 M	-	-	√	1+N
	AND	A,I	A ← A 与 I	-	-	√	1
	OR	A,M	A ← A 或 M	-	-	√	1
	OR	M,A	M ← A 或 M	-	-	√	1+N
	OR	A,I	A ← A 或 I	-	-	√	1
	XOR	A,M	A ← A 异或 M	-	-	√	1
	XOR	M,A	M ← A 异或 M	-	-	√	1+N
	XOR	A,I	A ← A 异或 I	-	-	√	1
PROCESS	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)	-	-	-	1
	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)	-	-	-	1+N
	RRC	M	A ← RRC M	√	-	-	1
	RRCM	M	M ← RRC M	√	-	-	1+N
	RLC	M	A ← RLC M	√	-	-	1
	RLCM	M	M ← RLC M	√	-	-	1+N
	CLR	M	M ← 0	-	-	-	1
	BCLR	M.b	M.b ← 0	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1	-	-	-	1+N
BRANCH	CMPRS	A,I	ZF,C ← A – I，如果 A = I，跳转到下一条指令。	√	-	√	1 + S
	CMPRS	A,M	ZF,C ← A – M，如果 A = M，跳转到下一条指令。	√	-	√	1 + S
	INCS	M	A ← M + 1，如果 A = 0，跳转到下一条指令。	-	-	-	1 + S
	INCMS	M	M ← M + 1，如果 M = 0，跳转到下一条指令。	-	-	-	1 + N + S
	DECS	M	A ← M – 1，如果 A = 0，跳转到下一条指令。	-	-	-	1 + S
	DECMS	M	M ← M – 1，如果 M = 0，跳转到下一条指令。	-	-	-	1 + N + S
	BTS0	M.b	如果 M.b = 0，跳转到下一条指令。	-	-	-	1 + S
	BTS1	M.b	如果 M.b = 1，跳转到下一条指令。	-	-	-	1 + S
	B0BTS0	M.b	如果 M(bank 0).b = 0，跳转到下一条指令。	-	-	-	1 + S
	B0BTS1	M.b	如果 M(bank 0).b = 1，跳转到下一条指令。	-	-	-	1 + S
	JMP	d	跳转指令，PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
	CALL	d	子程序调用指令，Stack ← PC15~PC0, PC15/14 ← RomPages1/0, PC13~PC0 ← d	-	-	-	2
MISC	RET		子程序跳出指令，PC ← Stack	-	-	-	2
	RETI		中断程序跳出指令，PC ← Stack，使能全局中断。	-	-	-	2
	NOP		空指令。	-	-	-	1

注： **条件跳转指令中，满足条件则 S=1，否则 S=0。**

14 开发工具

14.1 开发工具版本

14.1.1 在线仿真器ICE

- **SN8ICE2K Plus II**：可以仿真 **SN8P2949** 的所有功能。

- **SN8ICE2K ICE** 仿真时的注意事项：
  1、ICE 的工作电压：**3.3V**。
  2、建议 **3.3V** 工作电压下的最大仿真速度为：**1MIPS**（如 **Fcpu=Fhosc/4**）。
  3、利用 **SN8P2949 EV-KIT** 仿真模拟电路的功能。
  注：**SN8ICE1K** 不支持 **SN8P2949** 的仿真。

14.1.2 OTP Writer

- **MP Pro Writer**：联机/脱机支持 **SN8P2949** 大批量的烧录。

✷ **注：MPIII Writer 不支持 SN8P2949 OTP 的烧录。**

14.1.3 集成开发环境IDE

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

- **SN8ICE 2K Plus II**。
- **Easy Writer**，**MP-Easy** 和 **MPIII Writer** 均不支持 **SN8P2949**。

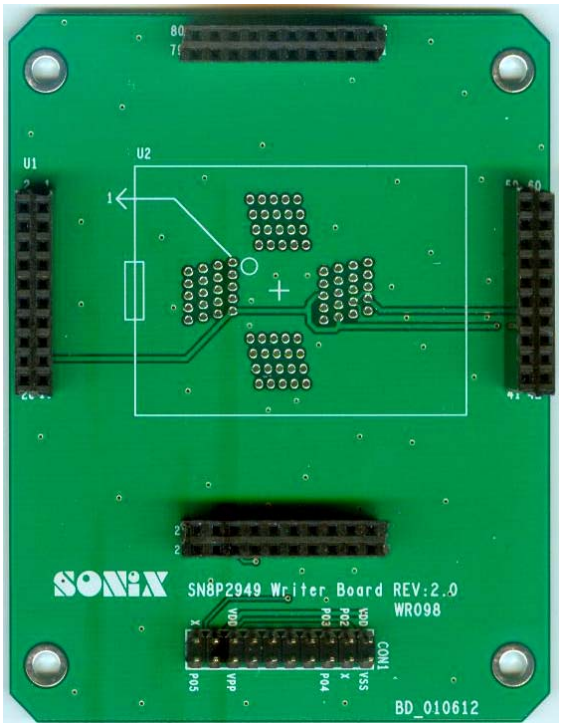
14.2 OTP烧录引脚和转接板配置

- **SN8P2949 COB 烧录引脚配置**

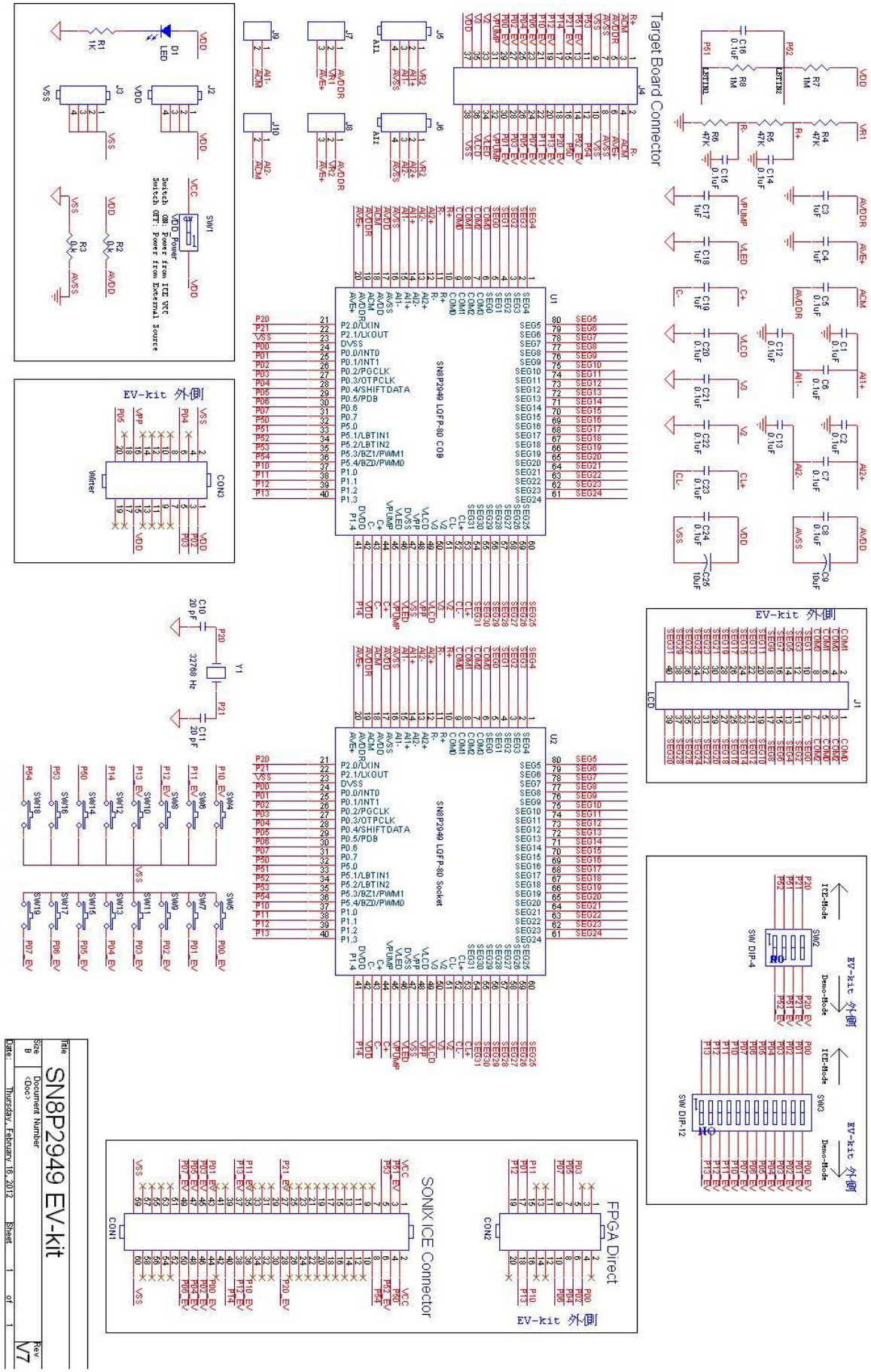
SN8P2949 烧录信息			
MP PRO Writer		SN8P2949	
JP3 of 40 PIN Adapter Board		引脚配置	
引脚编号	引脚名称	Pad 名称	LQFP80 引脚编号
1	VDD	DVDD/AVDD/ VLCD	42/17/49
2	GND	DVSS/AVSS	47/16
3	CLK / PGCLK	P0.2	26
4	CE	-	-
5	PGM / OTPCLK	P0.3	27
6	OE / ShiftData	P0.4	28
7	D1	-	-
8	D0	-	-
9	D3	-	-
10	D2	-	-
11	D5	-	-
12	D4	-	-
13	D7	-	-
14	D6	-	-
15	VDD	DVDD/AVDD / VLCD	42/17/49
16	VPP	VPP	48
17	HLS	-	-
18	RST	-	-
19	-	-	-
20	ALSB/PDB	P0.5	29

- **SN8P2949 封装片的烧录和转接板**

40 PIN 适配板连接到 MP PRO Writer 上。
连接 SN8P2949 Writer 转接板的 CON1 到适配板的 JP3。
将 LQFP-80 封装的 IC 放置到烧录工具 SOCKET 上。



14.3 附录A：EV-KIT电路图



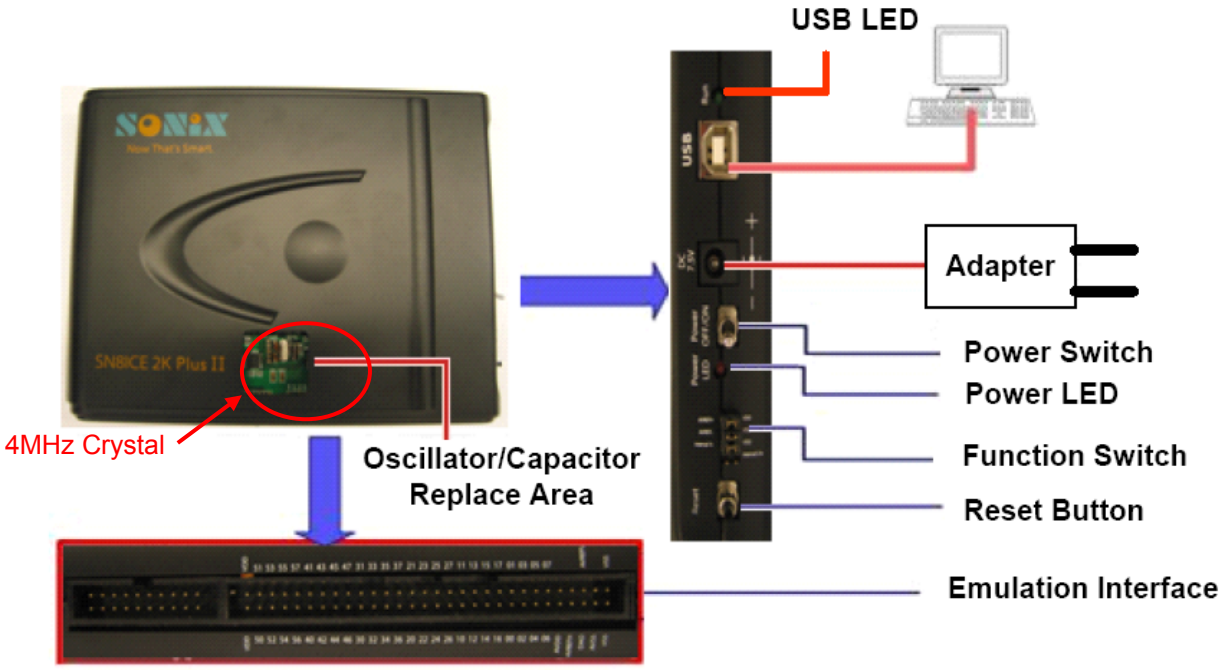
14.4 SN8P2949 仿真

14.4.1 概述

Sonix 提供一套完整的开发工具进行 SN8P2949 EV KIT 仿真，包括 SN8ICE2K_Plus_II，SN8P2949 EV KIT，Sonix 汇编编译器和 C 编译器。用户可以在电脑上进行程序的编写，通过编译软件或 ICE 进行仿真。用户还可以通过不同的功能如断点、单步执行等来执行程序或监控 RAM 的状态，这样程序员更容易进行调试。

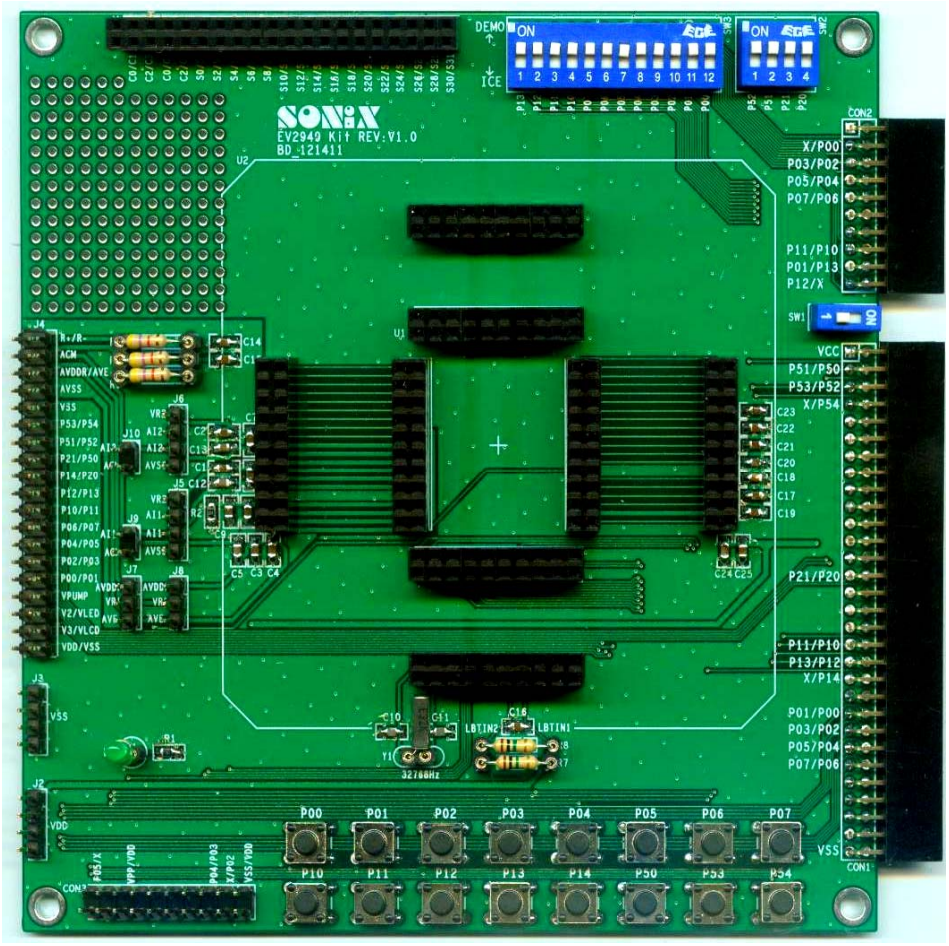
14.4.2 SN8ICE2K_Plus_II连接SN8P2949 EVKIT硬件设置注意事项

- 选择 4MHz 晶振连接到 ICE，用于产生系统高速时钟（Fhosc）。
- 检查 VDD 与内部 3.3V 是否短接，仿真 SN8P2949 时，VDD 只能选择内部 3.3V。
- 详见 SN8ICE2K Plus II 用户手册。



14.4.3 SN8P2949 EV KIT说明

Sonix 提供 SN8P2949 EV KIT 用于仿真 SN8P2949 的所有功能，如下图所示：



14.4.4 EV KIT设置

- U1/U2：SN8P2949 LQFP80 SOCKET。
- CON1/CON2：连接到 ICE 接口。
- SW1：EV Board 使用 ICE（3.3V）电源时，置于 ON 位置；EV Board 通过 J6/J7 输入外部电源时，置于 OFF 位置。
- SW2/SW3：EV Board 连接 ICE 时，置于 ICE 位置；EV Board 不连接 ICE 时，置于 DEMO 位置，否则相关 IO 口不工作。
- J1：LCD COM/SEG 连接引脚。
- J4：连接目标板。
- J5/J6：模拟差分（AI1+, AI1-）、（AI2+, AI2-）输入引脚。
- J7：选择外部参考电压源 V（R+,R-）来自 AVDDR 或 AVE+。
- J8：选择 Load Cell Power 来自 VDD 或 AVE+。
- J9/J10：模拟单端（AI1-/ACM）、（AI2-/ACM）输入引脚。
- R4/R5/R6：ADC 外部参考电压电阻。
- D1：电源指示灯。
- R7/R8：电池低电压检测电阻。R7 连接到 VDD 和 LBTIN2；R8 连接到 LBTIN2 和 LBTIN1。

14.4.5 仿真注意事项

- ICE VDD 必须连接 3.3V，5V 电源在 SN8P2949 EVKIT 上不工作。
- SW2/SW3 必须置于 ICE 位置。
- 电池低电压检测（LBT）功能支持内部 LBT 仿真和 P52 的输入。
- 在 EV KIT 上仿真 LBT 功能时，LBT 电压范围为 2.3V~2.9V。
- 进行内部 LBT 仿真之前，应该打开 SW1，参考电压由 J2 提供。
- 节点 LBTIN2 是 LBT 的外部输入脚；节点 LBTIN1 可以作为 I/O 引脚，或者通过设置标志 P51IO，直接内部接地。在仿真外部 LBT 前，必须打开 SW1。
- X 命令在 ICE 仿真时被取消（XB0MOV，XB0BSET...）。

15 电气特性

15.1 极限参数

Supply voltage (VDD)..... - 0.3V ~ 3.6V
Input in voltage (VIN)..... VSS - 0.2V ~ VDD + 0.2V
Operating ambient temperature (TOPR)..... 0°C ~ + 70°C
Storage ambient temperature (TSTOR)..... –40°C ~ + 125°C

15.2 电气特性

(All of voltages refer to VSS, V_{DD} = 3.0V,FOSC = IHRC(4MHz),Fcpu=1MHZ, ambient temperature is 25°C unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd		2.4	3.0	3.6	V
RAM Data Retention voltage	Vdr			-	1.5	-	V
VDD rise rate	VPOR	VDD rise rate to ensure power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	All input pins		Vss		0.3Vdd	V
Input High Voltage	ViH1	All input pins		0.7Vdd	-	Vdd	V
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
I/O port input leakage current	ILEKG	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O port source current	IoH	Vop = Vdd - 0.5V		4	8	-	mA
sink current	IoL	Vop = Vss + 0.5V		4	8	-	
INT0 trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
	Idd1	Normal Mode	Vdd=3V, Analog Parts OFF	-	0.5	1	mA
			Vdd=3V, Analog Parts ON	-	1.5	3	mA
			Vdd=3V, Analog Parts ON + LED driver	-	1.7	3.4	mA
		Slow Mode	Vdd=3V, IHRC On Analog Parts On, C-Type LCD On	-	1	2	mA
			Vdd=3V, IHRC OFF Analog Parts OFF, C-Type LCD On	-	150	300	uA
			Vdd=3V, HRC OFF Analog Parts OFF, R-LCD 400k On	-	10	20	uA
			Vdd=3V, IHRC OFF Analog Parts OFF, LCD OFF	-	5	10	uA
			Vdd=3V, IHRC OFF, IHRC_RTC Mode Analog Parts OFF, R-LCD 400k On	-	15	30	uA
			Vdd=3V, IHRC OFF, IHRC_RTC Mode Analog Parts OFF, LCD OFF	-	10	20	uA
		Green Mode	Vdd=3V, IHRC On Analog Parts On, C-Type LCD On	-	1	2	mA
			Vdd=3V, IHRC On Analog Parts OFF, C-Type LCD On	-	150	300	uA
			Vdd=3V, IHRC OFF, Analog Parts OFF, R-LCD 400k On	-	8	15	uA
			Vdd=3V, IHRC OFF, Analog Parts OFF, LCD OFF	-	3	6	uA
			Vdd=3V, IHRC OFF, IHRC_RTC Mode Analog Parts OFF, R-LCD 400k On	-	10	20	uA
			Vdd=3V, IHRC OFF, IHRC_RTC Mode Analog Parts OFF, LCD OFF	-	6	12	uA
		Sleep Mode	Vdd= 3V	-	1	2	uA
LVD detect level	VLVD	Internal POR detect level		1.7	1.9	2.2	uA
Internal High Clock Freq.	FIHRC	Internal High RC Oscillator Frequency (Vdd = 2.4V ~ 3.6V, Temperature: 0℃~ 70℃)		3.6	4	4.4	MHz

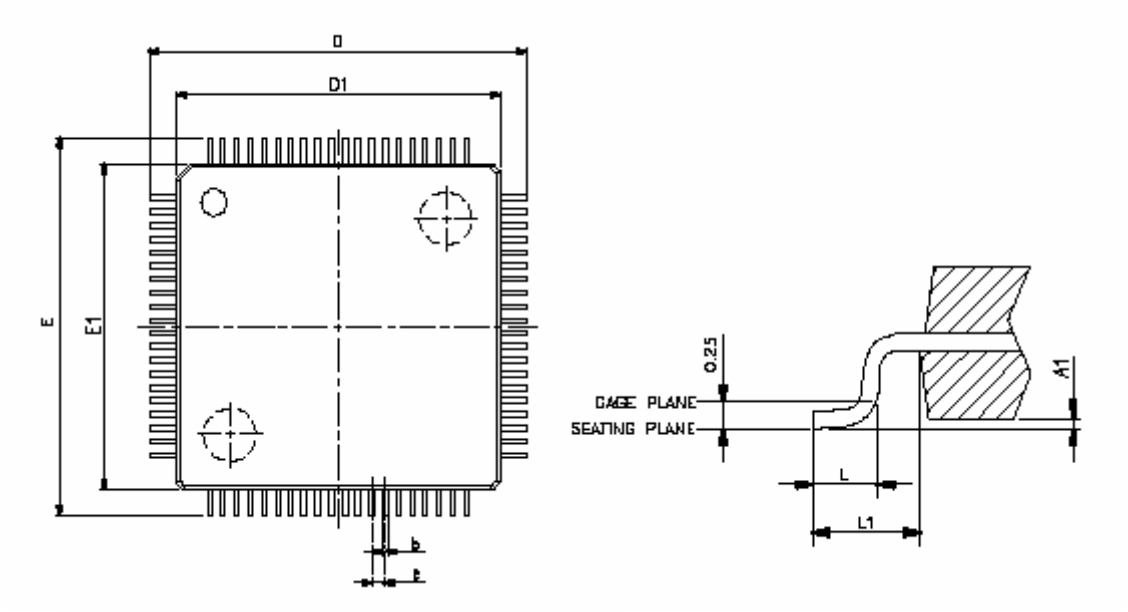
Note: Analog Parts including Regulator, PGIA and ADC.

(All of voltages refer to Vdd=3V FOSC = IHRC (4MHz), ambient temperature is 25°C unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
Analog to Digital Converter						
Operating current	IDD_ADC	Run mode @ 2.4V	-	200	300	uA
Power down current	IPDN	Stop mode @ 2.4V	-	0.1	-	μA
Conversion rate (Word Rare, WR)	FWR	ADC Clock=250KHz, OSR=32768	-	7.6	-	Sps
		ADC Clock=250KHz, OSR=64	-	3.9	-	kSps
Reference Voltage Input absolutely Voltage	VR+	GR=1, R+, R- Input absolutely Voltage	0.4	-	1.4	V
		GR=0, R+, R- Input absolutely Voltage	0.4	-	1.7	V
Reference Voltage Range	Vref	ADC Reference voltage range. (R+) - (R-)	0.3	-	0.8	V
ADC Input signal absolutely Voltage	VAIN	Input Buffer on (GX=1), AVDDR=2.4V	0.4	-	1.4	V
		Input Buffer off (GX=0) , AVDDR=2.4V	0.4	-	1.7	V
Integral non-linearity	INL	PGIA x 200, ADC Input Range ± 0.9xVref	-	-	0.01	%FSR
No missing code	NMC	ADC range ± 0.9 x Vref	20	-	-	bit
ADC Noise free bits	NFB	Gain=1, Vref:0.8V, OSR:32768, Input-short Buffer Off. (GR=GX=0)	-	18.4	-	bit
		Gain=200, Vref:0.8V, OSR:32768, Input-short	-	16	-	bit
Input Refer Noise (Peak to Peak)	VNp-p	Gain=1, WR=7.6Hz.	-	4.5	-	uV
		Gain=200, WR=7.6Hz.	-	0.17	-	uV
Unit Gain Buffer Current	IGX	ADC signal input buffer operation current.	-	80	100	uA
	IGR	ADC reference input buffer operation current.	-	80	100	uA
Unit Gain Buffer Input/Output Range	VGX	Absolutely voltage range. (AVDDR 2.4V)	0.4	-	1.4	V
	VGR	Min = 0 + 0.4V, Max = AVDDR-1V.	0.4	-	1.4	V
Temperature sensor Range	TR	Temperature Sensor Operation Range	-10	-	+70	℃
Temperature Sensitvity	TS	Temperature Sensor Sensitivity.	3.52	3.2	2.88	mV/℃
Temperature Sensor Accuracy	ETS	One Temperature point Calibration.	-10	-	+10	%
		Two Temperature points Calibration.	-1	-	+1	%
PGIA						
PGIA Current consumption	IDD_PGIA	Run mode @ 2.4V	-	250	400	uA
Power down current	IPDN	Stop mode @ 2.4V	-	-	0.1	uA
PGIA Gain Range	Gain	VDD = 2.4V, PGIA x 200	190	200	220	Gain
PGIA Input Range	Vopin	AI+, AI- signal input range. (AVDDR = 2.4V)	0.4	-	1.4	V
PGIA Output Range	Vopout	Signal output range. (AVDDR = 2.4V)	0.4	-	1.4	V
Band gap Reference (Refer to ACM)						
Band gap Reference Voltage	VBG	VDD: 2.4V ~ 3.6V	1.18	1.23	1.28	V
Reference Voltage Temperature Coefficient	TACM		-	50*	-	PPM/℃
Operating current	IBG	Run mode @ 2.4V	-	160	200	uA
Regulator						
Regulator output voltage AVDDR	VAVDDR		2.25	2.4	2.55	V
Regulator output voltage AVE+	VAVE+	AVE+ set as 2.0V	1.85	2.0	2.15	V
Analog common voltage	VACM	VACM = 1V	0.9	1	1.1	V
Regulator output current capacity	IVA+	AVDDR, AVE output current ability	-	-	5	mA
Quiescent current	IQI	ACM + AVDDR + AVE	-	130	150	uA
VACM driving capacity	ISRC		-	-	10	μA
VACM sinking capacity	ISNK		-	-	1	mA
LCD Driver						
R-Type LCD Operation Current	IRLCD	VDD=3V, 1/3 bias, 400k bias resistor, No panel		3	5	uA
		VDD=3V, 1/3 bias, 33k bias resistor, No panel		30	45	uA
C-Type LCD Operation Current	ICLCD	1/3 bias, LCD Charge pump + Bandgap current		160	200	uA
C-Type VLCD output Voltage	VLCD	VLCD set 3V,	2.85	3.05	3.25	V
VLCD Variation vs. VDD and Temp		VDD: 2.4~3.6V. Temp.: -10 ~ 50℃	-50	-	50	mV
VLED Driver (Pump + LDO)						
VLED Output Voltage	VLED	VDD=3V,	3.3	3.5	3.7	V
VLED Output Current Capacity	ILED	VDD=2.4V, Pump Clock = 125KHz	8			mA

16 封裝形式

16.1 LQFP 80 PIN



VARIATIONS (ALL DIMENSIONS SHOWN IN MM)

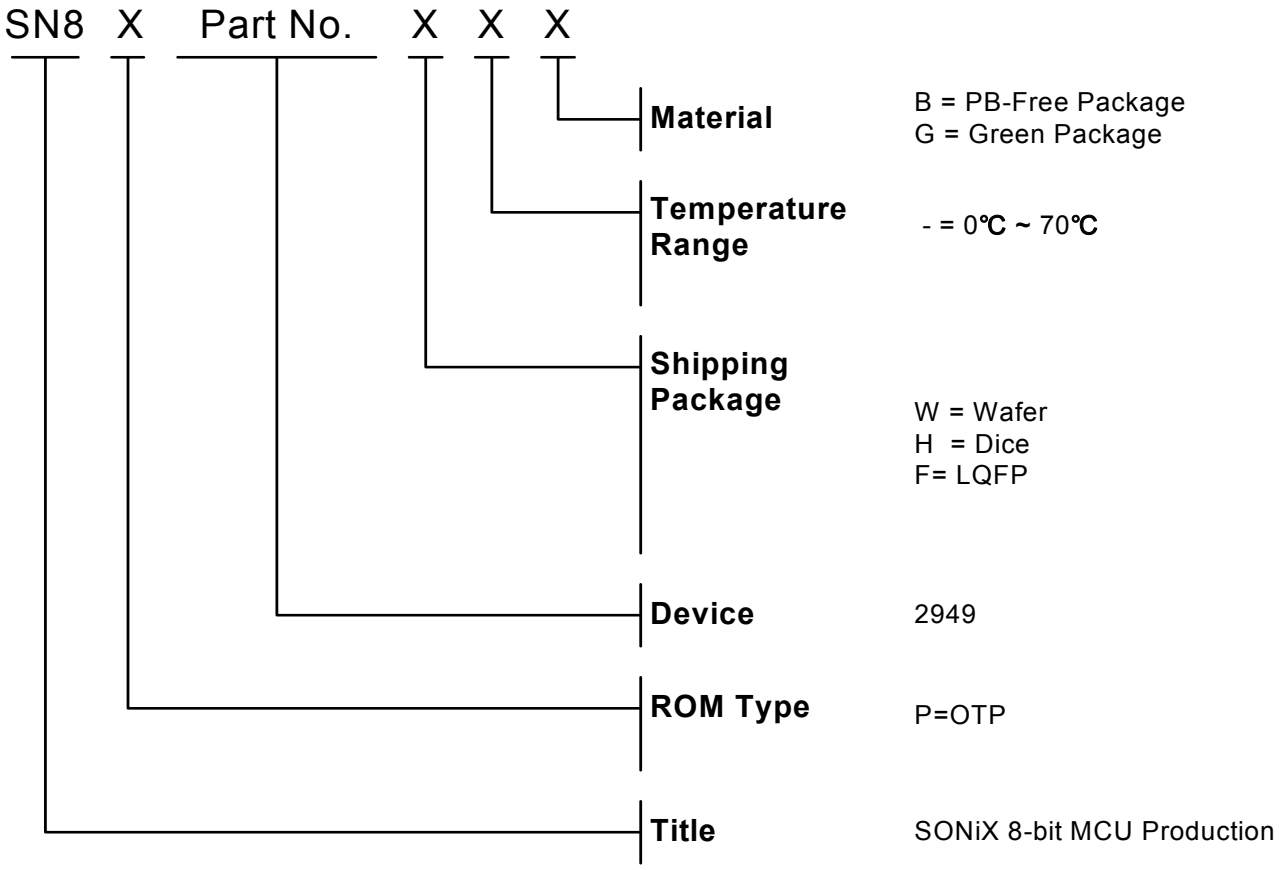
SYMBOLS	MIN.	MAX.
A	--	1.6
A1	0.05	0.15
A2	1.35	1.45
c1	0.09	0.16
D	12 BSC	
D1	10 BSC	
E	12 BSC	
E1	10 BSC	
e	0.4 BSC	
b	0.17	0.27
L	0.45	0.75
L1	1 REF	

17 单片机正印命名规则

17.1 概述

SONiX 8 位单片机产品具有多种型号，本章将给出所有 8 位单片机分类命名规则，适用于空片 OTP 型单片机。

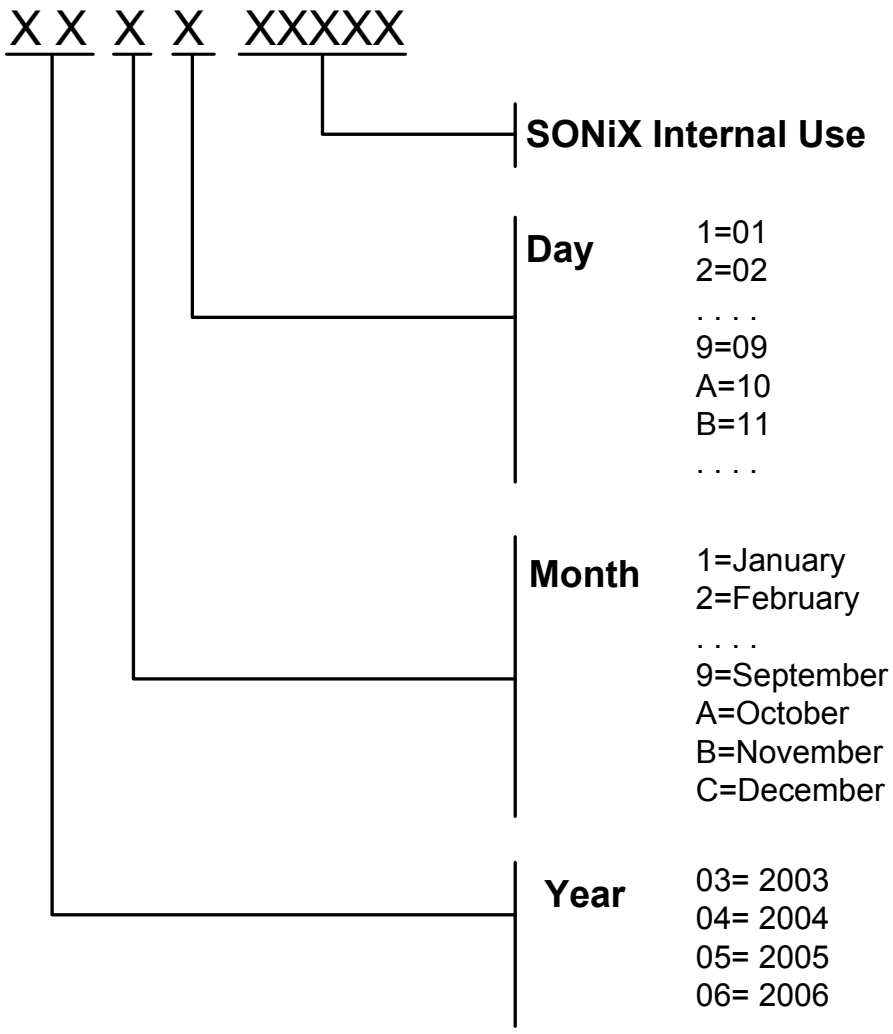
17.2 单片机型号说明



17.3 命名举例

单片机名称	ROM 类型	设备（Device）	封装形式	温度范围	封装材料
SN8P2949FG	OTP	2949	LQFP80	0℃~70℃	绿色封装

17.4 日期码规则



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

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