

# SN8P26L00 系列

## 用户参考手册

Version 1.0

**SN8P26L34**  
**SN8P26L32**  
**SN8P26L321**

## SONiX 8 位单片机

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。

修改记录

版本	日期	说明
VER 1.0	2008 年 6 月	初版。

## 目 录

1 产品简介 .....	6
1.1 功能特性 .....	6
1.2 系统框图 .....	7
1.3 引脚配置 .....	8
1.4 引脚说明 .....	9
1.5 引脚电路结构图 .....	10
2 中央处理器（CPU） .....	11
2.1 程序存储器（ROM） .....	11
2.1.1.1 复位向量（0000H） .....	11
2.1.1.2 中断向量（0008H） .....	12
2.1.1.3 查表 .....	13
2.1.1.4 跳转表 .....	15
2.1.2 CHECKSUM计算 .....	16
2.2 数据存储器（RAM） .....	17
2.2.1 系统寄存器 .....	18
2.2.1.1 系统寄存器列表 .....	18
2.2.1.2 系统寄存器说明 .....	18
2.2.1.3 系统寄存器的位定义 .....	19
2.2.1.4 累加器 .....	20
2.2.1.5 程序状态寄存器PFLAG .....	21
2.2.1.6 程序计数器 .....	22
2.2.1.7 H, L寄存器 .....	24
2.2.1.8 Y, Z寄存器 .....	25
2.2.1.9 R寄存器 .....	25
2.3 寻址模式 .....	26
2.3.1 立即寻址 .....	26
2.3.2 直接寻址 .....	26
2.3.3 间接寻址 .....	26
2.4 堆栈 .....	27
2.4.1 概述 .....	27
2.4.2 堆栈寄存器 .....	28
2.4.3 堆栈操作 .....	29
2.5 编译选项表（CODE OPTION） .....	30
2.5.1.1 RESET_PIN编译选项 .....	30
2.5.1.2 SECURITY编译选项 .....	30
3 复位 .....	31
3.1 概述 .....	31
3.2 上电复位 .....	32
3.3 看门狗复位 .....	32
3.4 掉电复位 .....	33
3.4.1 系统工作电压 .....	33
3.4.2 低电压检测（LVD） .....	34
3.4.3 掉电复位性能改进 .....	35
3.5 外部复位 .....	36
3.6 外部复位电路 .....	37
3.6.1 RC复位电路 .....	37
3.6.2 二极管及RC复位电路 .....	37
3.6.3 稳压二极管复位电路 .....	38
3.6.4 电压偏置复位电路 .....	38
3.6.5 外部IC复位 .....	39
4 系统时钟 .....	40
4.1 概述 .....	40
4.2 时钟框图 .....	40

4.3 FcPU (指令周期)	40
4.4 OSCM寄存器	41
4.5 系统高速时钟	42
4.5.1 内部高速RC振荡器	42
4.5.2 外部高速时钟	43
4.5.2.1 石英/陶瓷振荡器	43
4.5.2.2 RC振荡器	44
4.5.2.3 外部时钟源	44
4.6 系统低速时钟	45
4.6.1 系统时钟测试	45
5 系统工作模式	46
5.1 概述	46
5.2 普通模式	47
5.3 低速模式	47
5.4 睡眠模式	47
5.5 绿色模式	48
5.6 工作模式控制宏	49
5.7 唤醒时间	50
5.7.1 概述	50
5.7.2 唤醒时间	50
5.7.3 P1W唤醒功能控制寄存器	50
6 中断	51
6.1 概述	51
6.2 中断请求使能寄存器INTEN	51
6.3 中断请求寄存器INTRQ	52
6.4 GIE全局中断	52
6.5 PUSH, POP处理	53
6.6 INT0 (P0.0) 中断	54
6.7 INT1 (P0.1) 中断	55
6.8 T0 中断	56
6.9 TC1 中断	57
6.10 多中断操作举例	58
7 I/O口	59
7.1 概述	59
7.2 I/O口模式	60
7.3 I/O上拉电阻	61
7.4 I/O漏极开路寄存器	62
7.5 I/O口数据寄存器	63
8 定时器	64
8.1 看门狗定时器	64
8.2 定时器T0	66
8.2.1 概述	66
8.2.2 模式寄存器T0M	66
8.2.3 计数寄存器T0C	67
8.2.4 T0 操作时序	67
8.3 定时/计数器TC1	68
8.3.1 概述	68
8.3.2 TC1M模式寄存器	69
8.3.3 TC1C计数寄存器	70
8.3.4 TC1R自动装载寄存器	71
8.3.5 TC1 时钟频率输出 (蜂鸣器输出)	72
8.3.6 TC1 操作流程	73
8.4 PWM1	74
8.4.1 概述	74
8.4.2 TC1IRQ和PWM占空比	75
8.4.3 PWM编程举例	75
8.4.4 PWM1 占空比注意事项	76

9	模拟比较器 .....	78
9.1	概述 .....	78
9.2	CP0M寄存器 .....	80
9.3	CP1M寄存器 .....	81
9.4	模拟比较器的应用 .....	82
10	红外信号输出 .....	83
10.1	概述 .....	83
10.2	红外信号控制寄存器 .....	84
10.2.1	TC0M模式寄存器 .....	84
10.2.2	TC0C计数寄存器 .....	84
10.2.3	TC0R自动装载寄存器 .....	84
10.2.4	TC0D红外信号占空比控制寄存器 .....	85
10.2.5	红外信号输出操作流程 .....	85
11	指令集 .....	86
12	电气特性 .....	87
12.1	极限参数 .....	87
12.2	电气特性 .....	87
13	开发工具 .....	88
13.1	SN8P26L34/32/321 EV-KIT .....	88
13.2	ICE和EV-KIT的应用注意事项 .....	89
14	OTP烧录信息 .....	90
14.1	烧录转接板信息 .....	90
14.2	烧录引脚信息 .....	92
15	单片机正印命名规则 .....	93
15.1	概述 .....	93
15.2	单片机型号说明 .....	93
15.3	命名举例 .....	94
15.4	日期码规则 .....	94
16	封装信息 .....	95
16.1	SK-DIP 28 PIN .....	95
16.2	SOP 28 PIN .....	96
16.3	SSOP 28 PIN .....	97
16.4	P-DIP 18 PIN .....	98
16.5	SOP 18 PIN .....	99
16.6	P-DIP 20 PIN .....	100
16.7	SOP 20 PIN .....	101
16.8	SSOP 20 PIN .....	102

# 1 产品简介

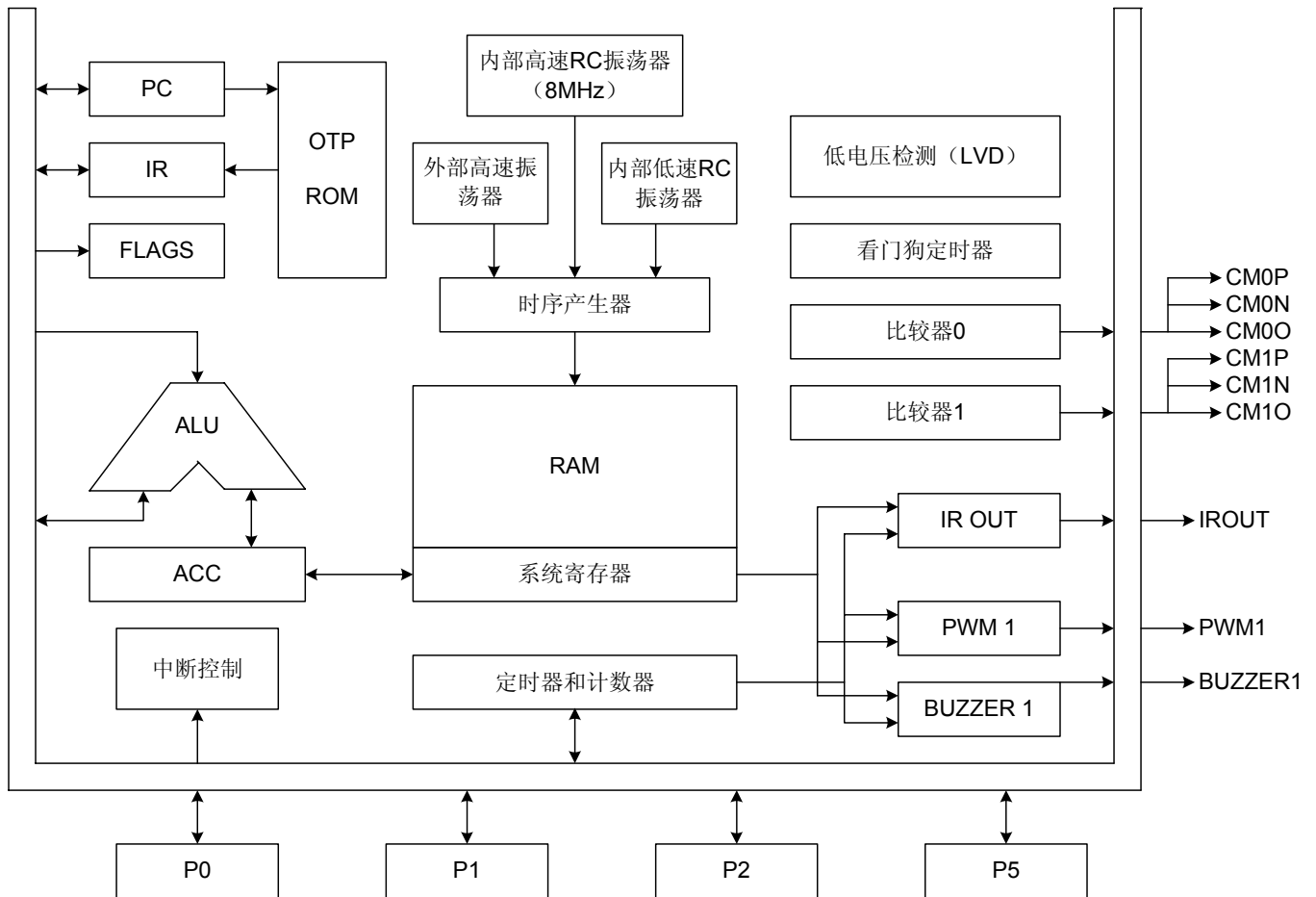
## 1.1 功能特性

- ◆ **存储器配置**  
OTP ROM 空间：8K \* 16 位。  
RAM 空间：240 字节。
- ◆ **8 层堆栈缓存器**
- ◆ **I/O 引脚配置**  
输入输出双向端口：P0、P1、P2、P5。  
可编程的漏极开路引脚：P1.0、P1.1。  
具有唤醒功能的端口：P0、P1 的电平变化触发。  
P1 的唤醒功能有 P1W 控制。  
内置上拉电阻的端口：P0、P1、P2、P5。  
外部中断引脚：P0.0、P0.1。  
外部中断触发边沿：  
P0.0 的中断由 PEDGE 控制。
- ◆ **3 层 LVD**  
复位系统，监控电源。
- ◆ **2 通道的模拟比较器，内置 0.9V/1.0V/1.1V/1.2V 可选择的参考电压和外部参考输入。**
- ◆ **6 个中断源**  
4 个内部中断：T0、TC1、CM0、CM1。  
2 个外部中断：INT0、INT1。
- ◆ **强大的指令系统**  
单周期指令系统（1T）。  
JMP 指令可在整个 ROM 区执行。  
CALL 指令可寻址整个 ROM 区。  
查表指令（MOVC）可寻址整个 ROM 区。
- ◆ **2 个 8 位定时/计数器**  
T0：基本定时器。  
TC1：自动装载定时/计数器。
- ◆ **一个 RTC 定时器（T0）**
- ◆ **一通道的 PWM 输出**
- ◆ **一通道的 Buzzer 输出**
- ◆ **一通道的 IR 输出（占空比可编程的 PWM 和 TC0）**
- ◆ **内置看门狗定时器，时钟源由内部低速 RC 振荡电路提供（10KHz @3V）**
- ◆ **4 种系统时钟**  
外部高速时钟：RC 模式，高达 8 MHz。  
外部高速时钟：晶体模式，高达 8 MHz。  
内部高速时钟：RC 模式，高达 8MHz。  
内部低速时钟：RC 模式，10KHz（3V）。
- ◆ **4 种工作模式**  
普通模式：高、低速时钟同时工作。  
低速模式：只有低速时钟在工作。  
睡眠模式：高、低速时钟同时停止工作。  
绿色模式：由定时器周期性的唤醒。
- ◆ **封装形式**  
SK-DIP 28 pins。  
SOP 28 pins。  
SSOP 28 pins。  
P-DIP 18 pins。  
SOP 18 pins。  
SSOP 20 pins。

### 特性选择列表

单片机型号	ROM	RAM	堆栈	定时器		RTC	IR 输出	内部高速振荡器	I/O	比较器	PWM	具有唤醒功能的引脚输出	封装形式
				T0	TC1						Buzzer		
SN8P26L32	8K	240	8	V	V	V	V	8M	16	1-ch	V	8	DIP18/SOP18
SN8P26L321	8K	240	8	V	V	V	V	8M	18	2-ch	V	8	DIP20/SOP20/SSOP20
SN8P26L34	8K	240	8	V	V	V	V	8M	26	2-ch	V	13	SKDIP28/SOP28/SSOP28

## 1.2 系统框图



## 1.3 引脚配置

**SN8P26L34K (SK-DIP 28 pins)**

**SN8P26L34S (SOP 28 pins)**

**SN8P26L34X (SSOP 28 pins)**

P0.1/INT1	1	U	28	P2.7/CM1O
VDD	2		27	P2.6/CM1P
P5.4/IROUT	3		26	P2.5/CM1N
VSS	4		25	P2.4/CM0O
P0.0/INT0	5		24	P2.3/CM0P
P5.0	6		23	P2.2/CM0N
P5.1	7		22	XIN/P0.3
P5.2	8		21	XOUT/P0.4
P5.3/BZ1/PWM1	9		20	P2.1
P0.2/RST/VPP	10		19	P2.0
P1.0	11		18	P1.7
P1.1	12		17	P1.6
P1.2	13		16	P1.5
P1.3	14		15	P1.4

SN8P26L34K

SN8P26L34S

SN8P26L34X

**SN8P26L32P (P-DIP 18 pins)**

**SN8P26L32S (SOP 18 pins)**

P0.1/INT1	1	U	18	P2.4/CM0O
VDD	2		17	P2.3/CM0P
P5.4/IROUT	3		16	P2.2/CM0N
VSS	4		15	XIN/P0.3
P5.0	5		14	XOUT/P0.4
P5.1	6		13	P1.3
P5.2	7		12	P1.2
P5.3/BZ1/PWM1	8		11	P1.1
P0.2/RST/VPP	9		10	P1.0

SN8P26L32P

SN8P26L32S

**SN8P26L321P (P-DIP 20 pins)**

**SN8P26L321S (SOP 20 pins)**

**SN8P26L321X (SSOP 20 pins)**

P2.7/CM1O	1	U	20	P2.6/CM1P
P0.1/INT1	2		19	P2.5/CM1N
VDD	3		18	P2.3/CM0P
P5.4/IROUT	4		17	P2.2/CM0N
VSS	5		16	XIN/P0.3
P5.0	6		15	XOUT/P0.4
P5.1	7		14	P1.3
P5.2	8		13	P1.2
P5.3/BZ1/PWM1	9		12	P1.1
P0.2/RST/VPP	10		11	P1.0

SN8P26L321P

SN8P26L321S

SN8P26L321X

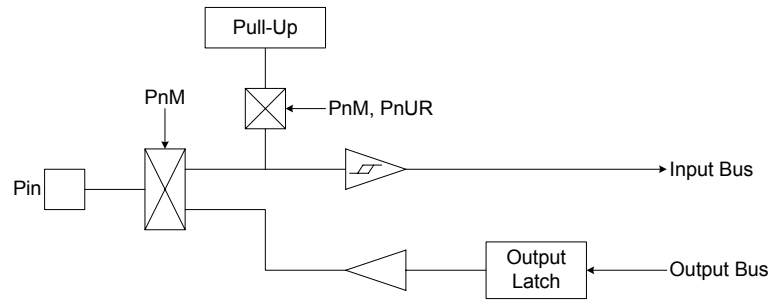


## 1.4 引脚说明

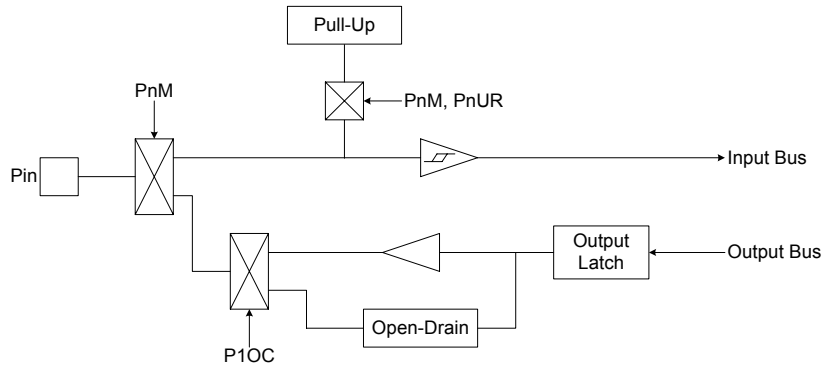
引脚名称	类型	功能说明
VDD, VSS	P	电源输入端。
P0[1:0]/INT[1:0]	I/O	双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻，具有唤醒功能。外部中断触发引脚。
P0.2/RST/VPP	I, P	RST: 在 Ext_RST 模式下为系统复位输入引脚。 P0.2: 单向输入引脚，无上拉电阻，施密特触发，低电平有效，通常保持高电平。具有唤醒功能。 烧录模式下的电源输入端。
P0.3/XIN	I/O	XIN: 使能外部振荡器（晶体和 RC）时为振荡信号的输入引脚。 P0.3: 在 8M RC 模式下为双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻，具有唤醒功能。
P0.4/XOUT	I/O	XOUT: 使能外部振荡器时为振荡信号的输出引脚。 P0.4: 在内部 8M RC 和外部 RC 模式下为双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻，具有唤醒功能。
P1[1:0]	I/O	双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻，由寄存器“P10C”控制漏极开路功能，具有唤醒功能。
P1[7:2]	I/O	双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻，具有唤醒功能。
P2[1:0]	I/O	双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻。
P2.2/CM0N	I/O	P2.2: 双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻。 CM0N: 模拟比较器 0 的负极输入引脚。
P2.3/CM0P	I/O	P2.3: 双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻。 CM0P: 模拟比较器 0 的正极输入引脚。
P2.4/CM0O	I/O	P2.4: 双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻。 CM0O: 模拟比较器 0 的输出引脚。当 CM0N 输入电平低于 CM0P 的电平时，CM0O 输出 Vdd; 当 CM0N 输入电平高于 CM0P 的电平时，CM0O 输出 Vss。
P2.5/CM1N	I/O	P2.5: 双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻。 CM1N: 模拟比较器 1 的负极输入引脚。
P2.6/CM1P	I/O	P2.6: 双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻。 CM1P: 模拟比较器 1 的正极输入引脚。
P2.7/CM1O	I/O	P2.7: 双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻。 CM1O: 模拟比较器 1 的输出引脚。当 CM1N 输入电平低于 CM1P 的电平时，CM1O 输出 Vdd; 当 CM1N 输入电平高于 CM1P 的电平时，CM0O 输出 Vss。
P5[2:0]	I/O	双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻。
P5.3/BZ1/PWM1	I/O	双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻。 Buzzer 输出引脚/PWM 输出引脚。
P5.4/IROUT	I/O	P5.4 双向输入/输出引脚，施密特触发，输入模式时内置上拉电阻 IROUT: IR 输出引脚。

## 1.5 引脚电路结构图

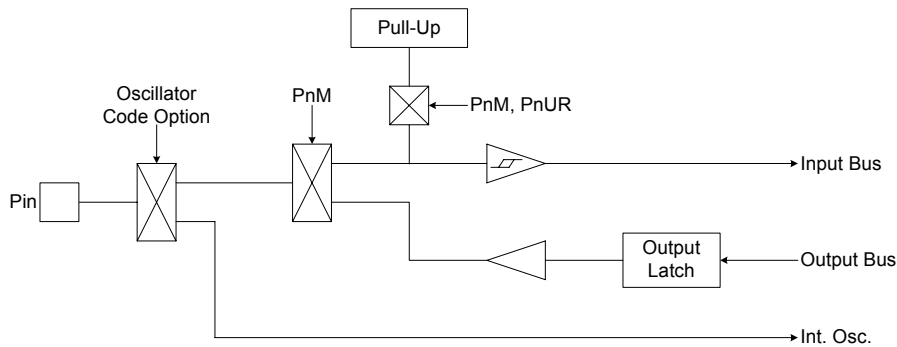
Port 0, 1, 2, 5 结构:



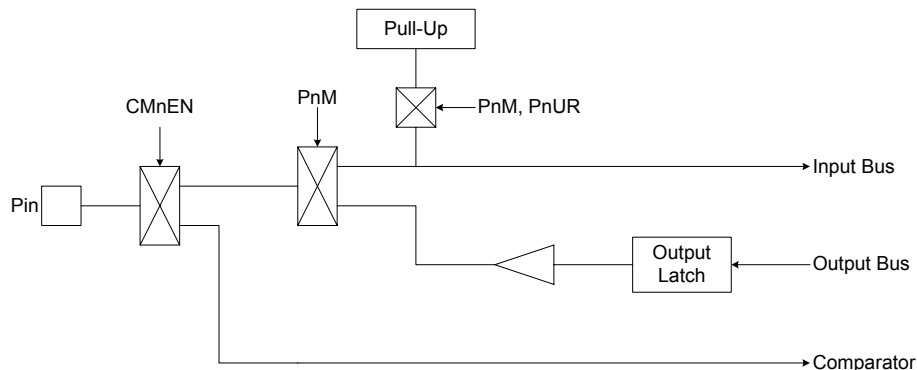
Port 1.0, P1.1 结构:



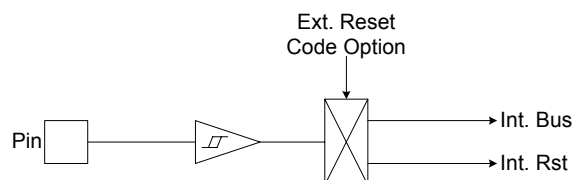
Port 0.3, 0.4 结构:



Port 2 结构:



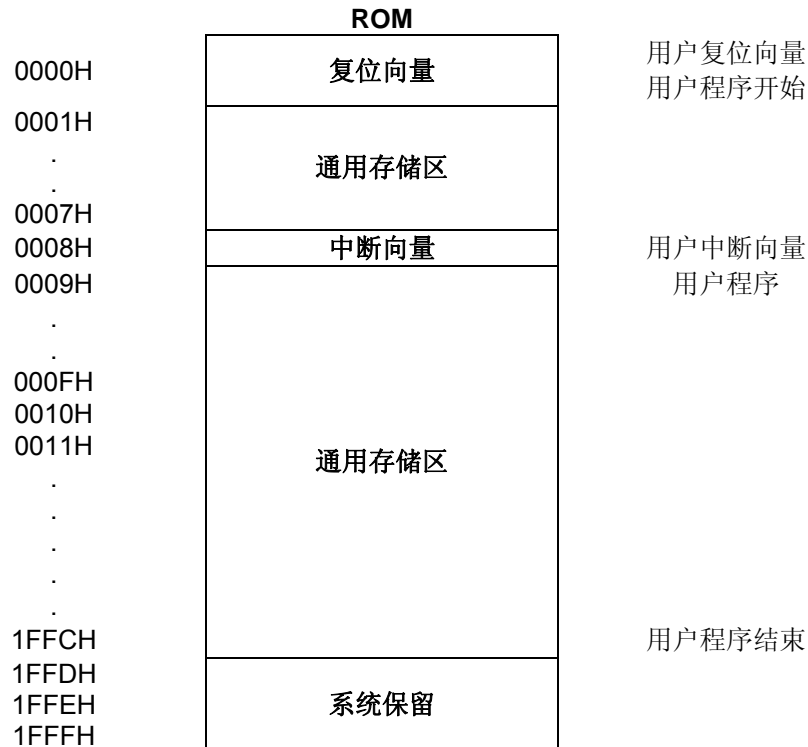
Port 0.2 结构:



# 2 中央处理器 (CPU)

## 2.1 程序存储器 (ROM)

☞ ROM: 8K



ROM 包括复位向量、中断向量、通用存储区域和系统保留部分。复位向量位于程序的开始处，中断向量则位于中断程序的开始处，通用存储区域则存放用户程序。

### 2.1.1.1 复位向量 (0000H)

具有一个字长的系统复位向量 (0000H)。

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

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

➤ 例：定义复位向量。

```

ORG      0          ;
JMP      START     ; 跳至用户程序。
...

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

```

## 2.1.1.2 中断向量 (0008H)

中断向量地址为 0008H。一旦有中断响应，程序计数器 PC 的当前值就会存入堆栈缓存器并跳转到 0008H 开始执行中断服务程序。0008H 处的第一条指令必须是“JMP”或“NOP”。下面的示例程序说明了如何编写中断服务程序。

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

➤ 例：定义中断向量，中断服务程序紧随 ORG 8H 之后。

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

START:
    ...           ; 用户程序开始。
    ...           ;
    JMP     START    ; 用户程序结束。
    ...           ;
    ENDP        ; 程序结束。
```

➤ 例：定义中断向量，中断程序在用户程序之后。

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

START:
    ORG     10H        ; 用户程序开始。
    ...
    JMP     START     ; 用户程序结束。
    ...

MY_IRQ:
    ...           ; 中断程序开始。
    PUSH   ; 保存 ACC 和 PFLAG。
    ...
    POP    ; 恢复 ACC 和 PFLAG。
    RETI   ; 中断程序结束。
    ...
    ENDP        ; 程序结束。
```

\* 注：从上面的程序中容易得出 SONiX 的编程规则，有以下几点：

1. 地址 0000H 的“JMP”指令使程序从头开始执行；
2. 地址 0008H 是中断向量；
3. 用户的程序应该是一个循环。

## 2.1.1.3查表

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

➤ 例：查找 ROM 地址为“TABLE1”的值。

```

B0MOV  Y, #TABLE1$M    ; 设置 TABLE1 地址高字节。
B0MOV  Z, #TABLE1$L    ; 设置 TABLE1 地址低字节。
MOVC   R, @Y           ; 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   R, @Y           ; 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:
        MOV    ;
        MOV    ; 存储数据, 如果 BUF = 0, 数据为 0035H。
        MOV    ; 如果 BUF = 1, 数据=5105H。
        MOV    ; 如果 BUF = 2, 数据=2012H。

TABLE1:
        ...
        DW     0035H        ; 定义数据表 (16 位) 数据。
        DW     5105H
        DW     2012H
        ...

```

## 2.1.1.4 跳转表

跳转表能够实现多地址跳转功能。由于 PCL 和 ACC 的值相加即可得到新的 PCL，因此，可以通过对 PCL 加上不同的 ACC 值来实现多地址跳转。ACC 值若为 n，PCL+ACC 即表示当前地址加 n，执行完当前指令后 PCL 值还会自加 1，可参考以下范例。如果 PCL+ACC 后发生溢出，PCH 则自动加 1。由此得到的新的 PC 值再指向跳转指令列表中新的地址。这样，用户就可以通过修改 ACC 的值轻松实现多地址的跳转。

\* 注：PCH 只支持 PC 增量运算，而不支持 PC 减量运算。当 PCL+ACC 后如有进位，PCH 的值会自动加 1。PCL-ACC 后若有借位，PCH 的值将保持不变，用户在设计应用时要加以注意。

## ➤ 例：跳转表。

```

ORG      0100H      ; 跳转表从 ROM 前端开始。

B0ADD    PCL, A      ; PCL = PCL + ACC, PCL 溢出时 PCH 加 1。
JMP      A0POINT    ; ACC = 0, 跳至 A0POINT。
JMP      A1POINT    ; ACC = 1, 跳至 A1POINT。
JMP      A2POINT    ; ACC = 2, 跳至 A2POINT。
JMP      A3POINT    ; ACC = 3, 跳至 A3POINT。

```

SONiX 单片机提供一个宏以保证可靠执行跳转表功能，它会自动检测 ROM 边界并将跳转表移至适当的位置。但采用该宏程序会占用部分 ROM 空间。

## ➤ 例：如果跳转表跨越 ROM 边界，将引起程序错误。

```

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

```

\* 注：“VAL”为跳转表列表中列表个数。

## ➤ 例：宏“MACRO3.H”中，“@JMP\_A”的应用。

```

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

```

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

## ➤ 例：“@JMP\_A”运用举例

; 编译前  
ROM 地址

```

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

```

; 编译后  
ROM 地址

```

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

```

## 2.1.2 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格的访问。

➤ 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。

@@:
MOV      FC
B0BSET  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                ; Z=Z+1
JMP     @B                ; 若 Z != 00H, 进行下一个计算。
JMP     Y_ADD_1          ; 若 Z = 00H, Y+1。

END_CHECK:
MOV     A, END_ADDR1
CMPRS   A, Z              ; 检查 Z 地址是否为用户程序结束地址低位地址。
JMP     AAA              ; 否, 则进行 Checksum 计算。
MOV     A, END_ADDR2
CMPRS   A, Y              ; 是则检查 Y 的地址是否为用户程序结束地址中间地址。
JMP     AAA              ; 否, 则进行 Checksum 计算。
JMP     CHECKSUM_END     ; 是则 Checksum 计算结束。

Y_ADD_1:
INCMS   Y                ;
NOP
JMP     @B                ; 跳转到 Checksum 计算。

CHECKSUM_END:
...
...

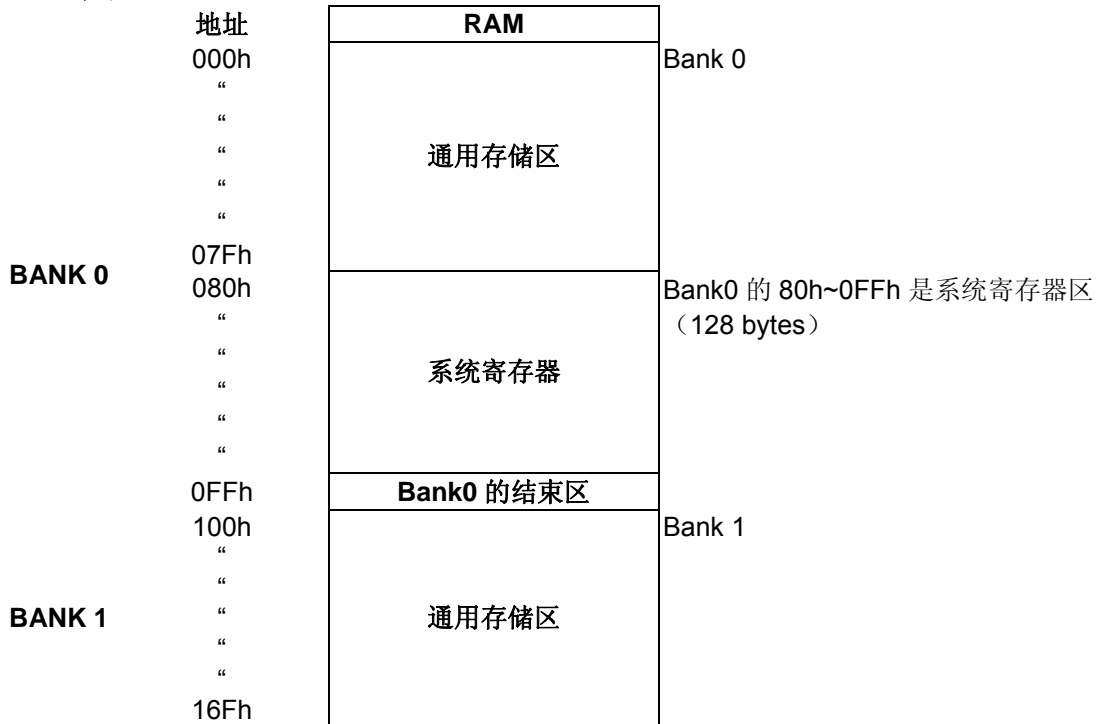
END_USER_CODE:
; 程序结束。

```



## 2.2 数据存储 (RAM)

☞ 2 RAM: 240 字节



240 字节的 RAM 分为 Bank0 和 Bank1 两个区域。通过寄存器“RBANK”决定访问哪个 Bank。如 RBANK=0，直接访问 Bank0，而当 RBANK=1，则直接访问 Bank1。如果在一个 Bank 区域但需要访问另一个 Bank 区域，就必须设置寄存器“RBANK”。当 RAM 不在 Bank0 区域时，Sonix 提供了“Bank0”类型的指令（如：B0MOV、B0ADD、B0BTS1、B0BSET 等）来直接访问 Bank0。

➤ 例：在 Bank1 下访问 Bank0。移动 Bank0 RAM (WK00) 的值到 Bank1 RAM (WK01)。

; Bank 1 (RBANK = 1)

```
B0MOV    A, WK00          ; 使用 Bank 0 类型的指令访问 Bank0 RAM。
MOV      WK01,A
```

\* 注：对于多 Bank RAM 的程序，不是很容易的选择 RAM Bank。用户必须小心注意 RBANK 的状况，在中断下尤其要注意。系统不会保存 RBANK，也不会把 RAM Bank 转换到 Bank0，用户必须通过程序自行解决以上问题。

## 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	-	PFLAG	RBANK	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	CMP0M	CMP1M	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
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	TC0D	P10C	-	-	-	-	-	-
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

PFLAG	= ROM 页及特殊标志寄存器	R	= 工作寄存器和 ROM 查表数据缓存器
H, L	= 专用寄存器, @HL 间接寻址寄存器, ROM 寻址寄存器	Y, Z	= 专用寄存器, @YZ 间接寻址寄存器, ROM 寻址寄存器
P1W	= P1 口的唤醒功能寄存器	RBANK	= RAM bank 选择寄存器
CMPnM	= 比较器控制寄存器	@HL	= 间接寻址寄存器
PEDGE	= P0.0 触发模式控制寄存器	@YZ	= 间接寻址寄存器
PnM	= Pn 输入/输出模式控制寄存器	Pn	= Pn 数据缓存器
P10C	= P1 漏极开路控制寄存器	PnUR	= Pn 上拉电阻控制寄存器
INTRQ	= 中断请求寄存器	INTEN	= 中断使能寄存器
OSCM	= 振荡模式控制寄存器	PCH, PCL	= 程序计数器
T0M	= T0 模式寄存器	T0C	= T0 计数寄存器
TC0M	= IR 输出控制寄存器	TC0C	= IR 周期控制寄存器
TC0D	= IR 输出占空比寄存器	TC0R	= IR 自动装载缓存器
TC1C	= TC1 计数寄存器	TC1M	= TC1 模式控制寄存器
WDTR	= 看门狗定时器清零寄存器	TC1R	= TC1 自动装载缓存器
STKP	= 堆栈指针寄存器	STK0~STK7	= 堆栈缓存器

## 2.2.1.3 系统寄存器的位定义

地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	读/写	备注
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
086H	NT0	NPD	LVD30	LVD24	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	-	-	-	RBANKS0	R/W	RBANK
09CH	CM0EN	CM0IEN	CM0IRQ	CM0OEN	CM0REF	CM0OUT	CMS1	CMS0	R/W	CMP0M
09DH	CM1EN	CM1IEN	CM1IRQ	CM1OEN	CM1REF	CM1OUT	-	-	R/W	CMP1M
0B8H	-	-	-	P04M	P03M	-	P01M	P00M	R/W	P0M
0BFH	-	-	-	P00G1	P00G0	-	-	-	R/W	PEDGE
0C0H	P17W	P16W	P15W	P14W	13W	P12W	P11W	P10W	W	P1W 唤醒功能寄存器
0C1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M I/O 模式
0C2H	P27M	P26M	P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M I/O 模式
0C5H	-	-	-	P54M	P53M	P52M	P51M	P50M	R/W	P5M I/O 模式
0C8H	-	TC1IRQ	-	T0IRQ	-	-	P01IRQ	P00IRQ	R/W	INTRQ
0C9H	-	TC1IEN	-	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	-	-	-	P04	P03	P02	P01	P00	R/W	P0 数据缓存器
0D1H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1 数据缓存器
0D2H	P27	P26	P25	P24	P23	P22	P21	P20	R/W	P2 数据缓存器
0D5H	-	-	-	P54	P53	P52	P51	P50	R/W	P5 数据缓存器
0D8H	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	T0TB	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	-	-	-	-	-	-	IREN	CREN	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD1	TC1OUT	PWM1OUT	R/W	TC1M
0DDH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
0DEH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	W	TC1R
0DFH	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0	R/W	STKP 堆栈指针
0E0H	-	-	-	P04R	P03R	-	P01R	P00R	W	P0 上拉电阻
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1 上拉电阻
0E2H	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R	W	P2 上拉电阻
0E5H	-	-	-	P54R	P53R	P52R	P51R	P50R	W	P5 上拉电阻
0E6H	@HL7	@ HL 6	@ HL5	@ HL4	@ HL3	@ HL2	@ HL1	@ HL0	R/W	@HL 间接寻址寄存器
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ 间接寻址寄存器
0E8H	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0	W	TC0D
0E9H	-	-	-	-	-	-	P110C	P100C	W	P10C
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H	1	1	1	S7PC12	S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H	1	1	1	S6PC12	S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H	1	1	1	S5PC12	S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H	1	1	1	S4PC12	S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H	1	1	1	S3PC12	S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH	1	1	1	S2PC12	S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH	1	1	1	S1PC12	S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH	1	1	1	S0PC12	S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

\* 注:

1. 所有系统寄存器的名称都已经在 SN8ASM 编译器中作了宣告;
2. 用户使用 SN8ASM 编译器对寄存器的位进行操作时, 需在该寄存器的位前加“F”;
3. 指令“B0BSET”、“B0BCLR”、“BSET”和“BCLR”只能用于可读写的寄存器;

### 2.2.1.4 累加器

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

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

➤ **例：读/写 ACC。**

; 数据写入 ACC。

```
MOV      A, #0FH
```

; 读取 ACC 中的数据并存入 BUF。

```
MOV      BUF, A
B0MOV    BUF, A
```

; BUF 中的数据写入 ACC。

```
MOV      A, BUF
B0MOV    A, BUF
```

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

➤ **例：ACC 和工作寄存器中断保护操作。**

INT\_SERVICE:

```
PUSH                                ; 保存 PFLAG 和 ACC。
```

```
...
```

```
...
```

```
POP                                  ; 恢复 ACC 和 PFLAG。
```

```
RETI                                ; 退出中断。
```

### 2.2.1.5 程序状态寄存器 PFLAG

寄存器 PFLAG 中包含 ALU 运算状态信息、系统复位状态信息和 LVD 检测信息，其中，位 NT0 和 NPD 显示系统复位状态信息，包括上电复位、LVD 复位、外部复位和看门狗复位；位 C、DC 和 Z 显示 ALU 的运算信息。位 LVD24 和 LVD28 显示了芯片供电电压状况。

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>PFLAG</b>	NT0	NPD	LVD28	LVD24	-	C	DC	Z
读/写	R/W	R/W	R	R	-	R/W	R/W	R/W
复位后	X	X	0	0	-	0	0	0

Bit [7:6] NT0, NPD: 复位状态标志。

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

Bit 5 LVD28: 2.8V LVD 工作电压标志，LVD 编译选项为 LVD\_H 时有效。  
0 = 系统工作电压 VDD 超过 2.8V，低电压检测器没有工作；  
1 = 系统工作电压 VDD 低于 2.8V，说明此时低电压检测器已处于监控状态。

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

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

### 2.2.1.6 程序计数器

程序计数器 PC 是一个 13 位二进制程序地址寄存器，分高 5 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	-	-	-	PC12	PC11	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	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          ; 若 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 的值而无需担心 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。
...
```

## 2.2.1.7H, L 寄存器

寄存器 H 和 L 都是 8 位寄存器，主要有以下两个功能：

- 通用工作寄存器；
- RAM 数据寻址指针@HL。

081H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>H</b>	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0
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

080H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>L</b>	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0
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 H、L 作为数据指针，访问 bank0 中 020H 处的内容。

```
B0MOV    H, #00H
B0MOV    L, #20H
B0MOV    A, @HL
```

➤ 例：对 bank 0 中的数据进行清零处理。

```
CLR      H                ; H = 0, 指向 bank 0。
B0MOV    L, #7FH         ; L = 7FH。

CLR_HL_BUF:
CLR      @HL              ; @HL 清零。
DECMS    L                ; L - 1, 如果 L = 0, 程序结束。
JMP      CLR_HL_BUF

END_CLR:
CLR      @HL
...
...
```



### 2.2.1.8Y, Z 寄存器

寄存器 Y 和 Z 都是 8 位寄存器，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
<b>Y</b>	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
<b>Z</b>	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:

```
...
```

### 2.2.1.9R 寄存器

8 位寄存器 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
<b>R</b>	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	X	X	X	X	X	X	X	X

## 2.3 寻址模式

### 2.3.1 立即寻址

将立即数送入 ACC 或指定的 RAM 单元。

- 例：立即数 12H 送入 ACC。  
MOV A, #12H
- 例：立即数 12H 送入寄存器 R。  
B0MOV R, #12H

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

### 2.3.2 直接寻址

通过 ACC 对 RAM 单元数据进行操作。

- 例：地址 12H 处的内容送入 ACC。  
B0MOV A, 12H
- 例：ACC 中数据写入 RAM 的 12H 单元。  
B0MOV 12H, A

### 2.3.3 间接寻址

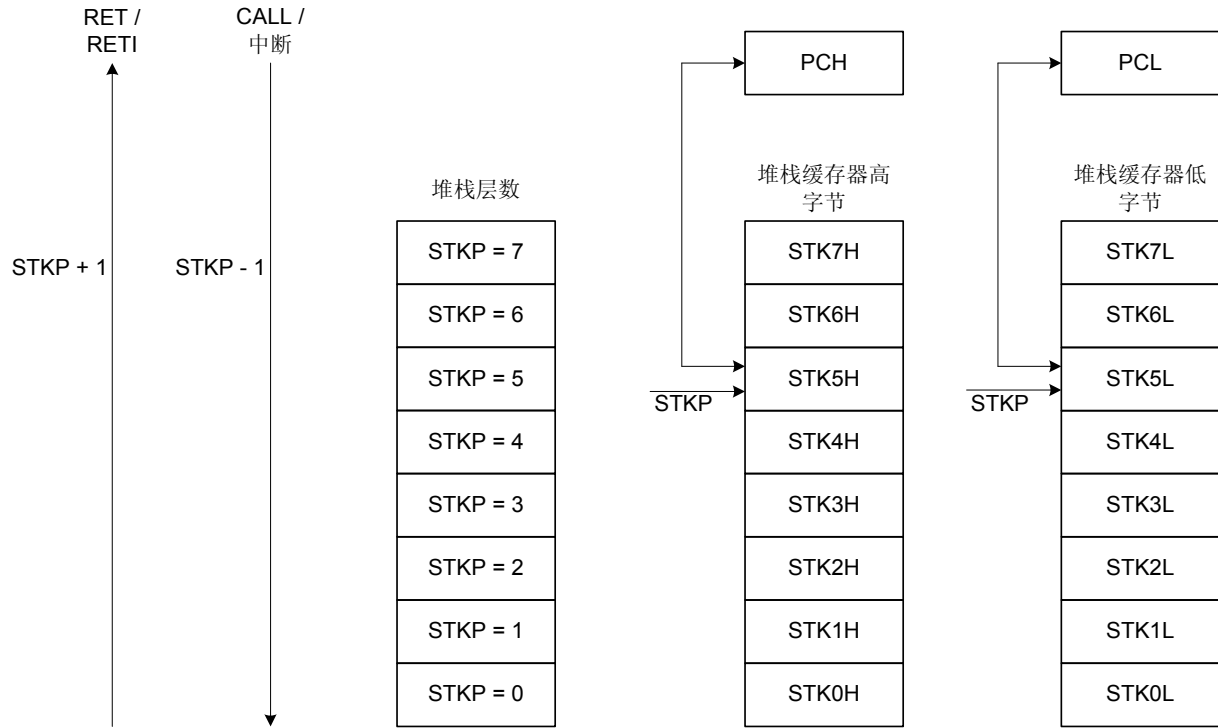
通过指针寄存器 (Y/Z) 访问 RAM 数据。

- 例：用 @YZ 实现间接寻址。  
B0MOV Y, #0 ; Y 清零以寻址 RAM bank 0。  
B0MOV Z, #12H ; 设定寄存器地址。  
B0MOV A, @YZ

## 2.4 堆栈

### 2.4.1 概述

SN8P26L00 系列的堆栈缓存器共 8 层，程序进入中断或执行 CALL 指令时，用于存储程序计数器 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。

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

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

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

Bit[2:0] **STKPBn**: 堆栈指针(n = 0 ~ 2)。

Bit 7 **GIE**: 全局中断控制位。

0 = 禁止;

1 = 使能。

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

```
MOV      A, #00000111B
B0MOV   STKP, A
```

0F0H~0FFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>STKnH</b>	-	-	-	SnPC12	SnPC11	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
<b>STKnL</b>	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

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，错误

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

堆栈层数	STKP			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	-

## 2.5 编译选项表（CODE OPTION）

编译选项为系统的硬件配置，包括振荡器的类型，看门狗定时器的工作模式，LVD 选项，复位引脚选项和 OTP ROM 的安全控制。其项目如下表所示：

编译选项	配置项目	功能说明
Fcpu	Fhosc/1	指令周期 = 1 个时钟周期。
	Fhosc/2	指令周期 = 2 个时钟周期。
	Fhosc/4	指令周期 = 4 个时钟周期。
	Fhosc/8	指令周期 = 8 个时钟周期。
High_Clk	IHRC_8M	高速时钟振荡器采用内部 8MHz RC 振荡器，XIN/XOUT (P0.3/P0.4) 作为普通的 I/O 引脚。
	IHRC_RTC	高速时钟振荡器采用内部 8MHz RC 振荡器，带有 0.5s RTC 功能。XIN/XOUT (P0.3/P0.4) 作为普通的 I/O 引脚。
	RC	外部高速时钟振荡器采用廉价的 RC 振荡电路，XOUT (P0.4) 作为普通的 I/O 引脚。
	32K X'tal	外部高速振荡器采用低频的晶体振荡器（如 32.768KHz）。
	8M X'tal	外部高速振荡器采用高频的晶体振荡器（如 8MHz）。
	4M X'tal	外部高速振荡器采用标准的晶体振荡器（如 4MHz）。
Watch_Dog	Always_On	始终开启看门狗定时器，即使在睡眠模式和绿色模式下也处于开启状态。
	Enable	开启看门狗定时器，但在睡眠模式和绿色模式下关闭。
	Disable	关闭看门狗定时器。
Reset_Pin	Reset	使能外部复位引脚。
	P02	P0.2 作为普通的输入引脚，但无上拉电阻。
LVD	LVD_L	若 VDD 低于 1.8V，LVD 复位。
	LVD_M	若 VDD 低于 1.8V，LVD 复位。 使能寄存器 PFLAG 的 LVD24 位以检测 2.4V 的低电压点。
	LVD_H	若 VDD 低于 2.4V，LVD 复位。 使能寄存器 PFLAG 的 LVD28 位以检测 2.8V 的低电压点。
Security	Enable	ROM 代码加密。
	Disable	ROM 代码不加密。

### 2.5.1.1 RESET\_PIN 编译选项

复位引脚和普通的输入引脚共用，通过编译选项控制工作模式。

- **Reset:** 外部复位功能。下降沿触发，系统复位。
- **P02:** 单向输入引脚。此时禁止外部复位功能，P0.2 作为普通的输入引脚。

### 2.5.1.2 SECURITY 编译选项

Security 编译选项保护 OTP ROM。当使能 Security 编译选项时，ROM 的代码处于安全状态，这样 ROM 的内容就不会丢失。

# 3 复位

## 3.1 概述

SN8P26L00 系列的单片机有以下几种复位方式：

- 上电复位；
- 看门狗复位；
- LVD 检测复位；
- 外部复位（仅在外外部复位引脚处于使能状态）。

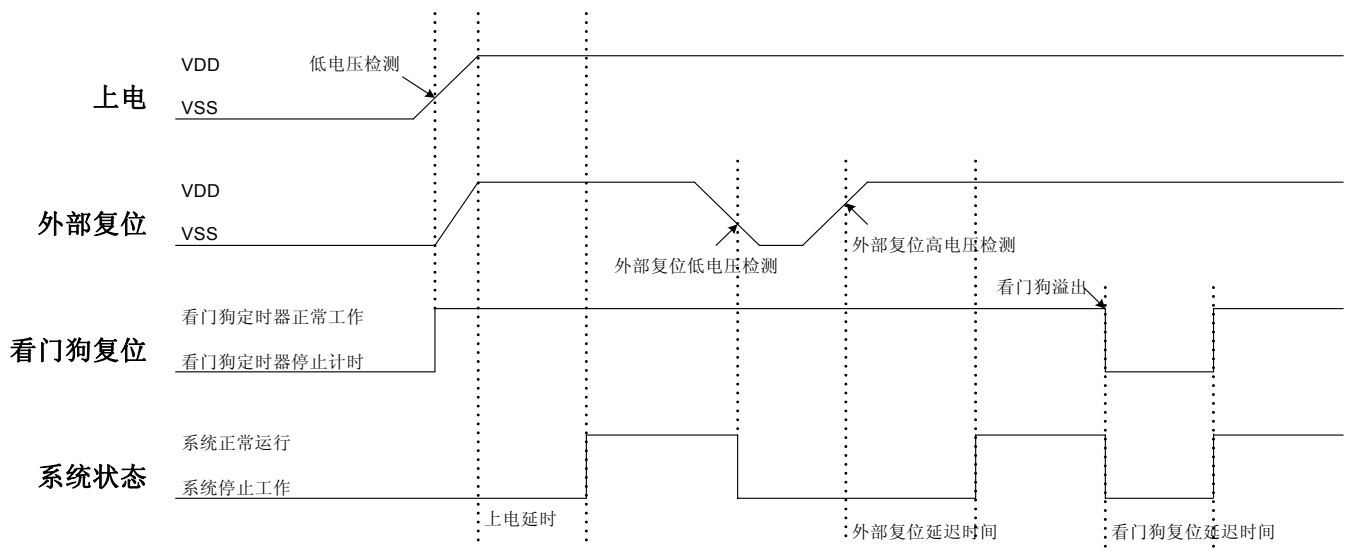
上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器 PC 清零。复位结束后，系统从向量 0000H 处重新开始运行。PFLAG 寄存器的 NT0 和 NPD 两个标志位能够给出系统复位状态的信息。用户可以编程控制 NT0 和 NPD，从而控制系统的运行路径。

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>PFLAG</b>	NT0	NPD	LVD28	LVD24	-	C	DC	Z
读/写	R/W	R/W	R	R	-	R/W	R/W	R/W
复位后	X	X	0	0	-	0	0	0

Bit [7:6] **NT0, NPD**: 复位状态标志位。

NT0	NPD	复位情况	说明
0	0	看门狗复位	看门狗定时器溢出
0	1	系统保留	-
1	0	上电及 LVD 复位	电源电压低于 LVD 检测电压
1	1	外部复位	外部复位引脚检测到低电平

任何一种复位情况都需要一定的响应时间，系统提供完善的复位流程以保证复位动作的顺利进行。对于不同类型的振荡器，复位完成所需要的时间也不同。因此，VDD 的上升速度和不同晶振的起振时间都不固定。RC 振荡器的起振时间最短，晶体振荡器的起振时间则较长。在用户使用的过程中，应注意考虑主机对上电复位时间的要求。



## 3.2 上电复位

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

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

看门狗定时器运用注意事项：

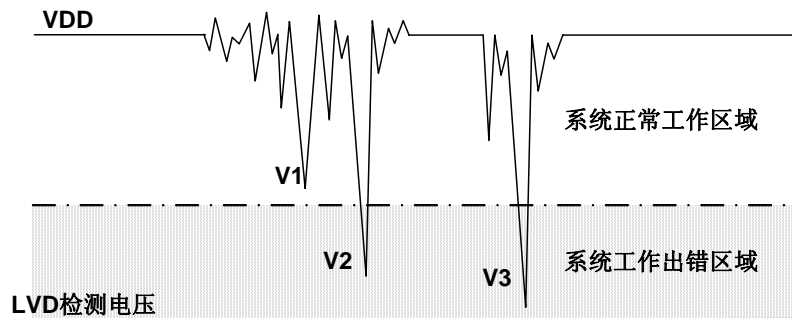
- 看门狗定时器清零之前，请检查 I/O 端口状态及 RAM 数据；
- 不能在中断向量和中断程序中将看门狗定时器清零，否则无法起到侦测程序跑飞的目的；
- 程序中应该只在主程序中有一个清看门狗的动作，这种架构能够最大限度的发挥看门狗的保护功能。

\* 注：关于看门狗定时器的详细内容，请参阅“看门狗定时器有关章节。”



### 3.4 掉电复位

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



掉电复位示意图

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

#### DC 运用中：

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

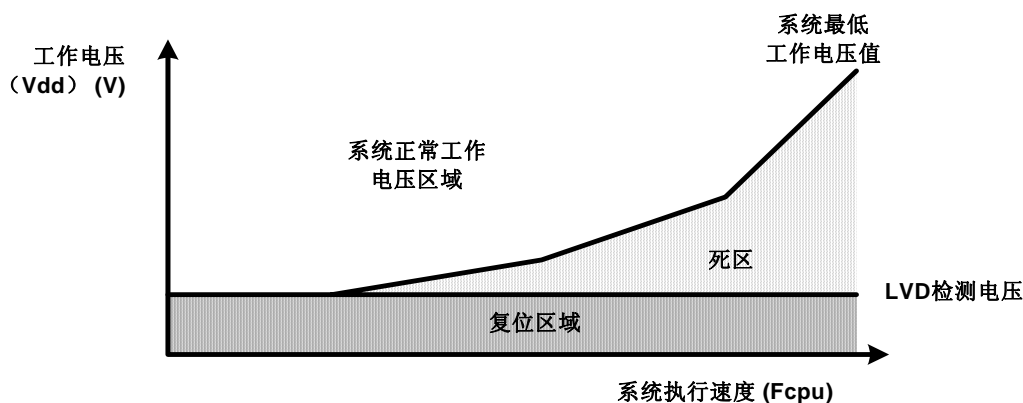
#### AC 运用中：

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

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

#### 3.4.1 系统工作电压

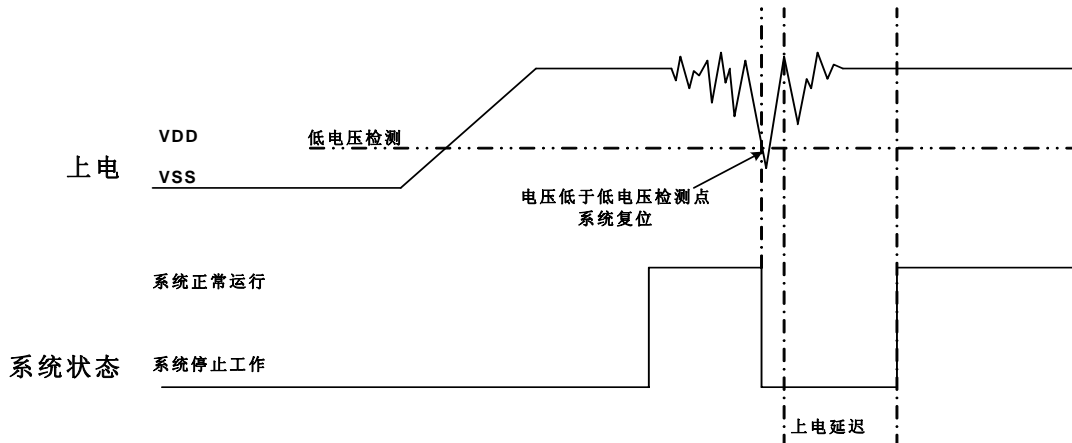
为了改善系统掉电复位的性能，首先必须明确系统具有的最低工作电压值。系统最低工作电压与系统执行速度有关，不同的执行速度下最低工作电压值也不同。



系统工作电压与执行速度关系图

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

## 3.4.2 低电压检测 (LVD)



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

LVD 设计为三层结构 (1.8V/2.4V/2.8V), 由 LVD 编译选项控制。对于上电复位和掉电复位, 1.8V LVD 始终处于使能状态; 2.4V LVD 具有 LVD 复位功能, 并能通过标志位显示 VDD 状态; 2.8V LVD 具有标记功能, 可显示 VDD 的工作状态。LVD 标志功能只是一个低电压检测装置, 标志位 LVD24 和 LVD28 给出 VDD 的电压情况。对于低电压检测应用, 只需查看 LVD24 和 LVD28 的状态即可检测电池状况。

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>PFLAG</b>	NT0	NPD	LVD28	LVD24	-	C	DC	Z
读/写	R/W	R/W	R	R	-	R/W	R/W	R/W
复位后	X	X	0	0	-	0	0	0

Bit 5 **LVD28**: 2.8V LVD 工作电压标志, LVD 编译选项为 LVD\_H 时有效  
0 = 系统工作电压 VDD 超过 2.8V, 低电压检测器没有工作;  
1 = 系统工作电压 VDD 低于 2.8V, 说明此时低电压检测器已处于监控状态。

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

LVD	LVD 编译选项		
	LVD_L	LVD_M	LVD_H
1.8V 复位	有效	有效	有效
2.4V 标志	-	有效	-
2.4V 复位	-	-	有效
2.8V 标志	-	-	有效

**LVD\_L**

如果 VDD < 1.8V, 系统复位。

LVD24 和 LVD28 标志位无意义。

**LVD\_M**

如果 VDD < 1.8V, 系统复位。

LVD24: 如果 VDD > 2.4V, LVD24 = 0; 如果 VDD ≤ 2.4V, LVD24 = 1;

LVD28 标志位无意义。

**LVD\_H**

如果 VDD < 2.4V, 系统复位;

LVD24: 如果 VDD > 2.4V, LVD24 = 0; 如果 VDD ≤ 2.4V, LVD24 = 1;

LVD28: 如果 VDD > 2.8V, LVD28 = 0; 如果 VDD ≤ 2.8V, LVD28 = 1。

## \* 注:

- 1、 LVD 复位, LVD24, LVD28 都将被清零;
- 2、 LVD 2.4V 和 2.8V 检测电平值仅作为设计参考, 不能用作芯片工作电压值的精确检测。

### 3.4.3 掉电复位性能改进

如何改善系统掉电复位性能，有以下几点建议：

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

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

#### 看门狗复位：

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

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

#### 降低系统工作速度：

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

#### 附加外部复位电路：

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

### 3.5 外部复位

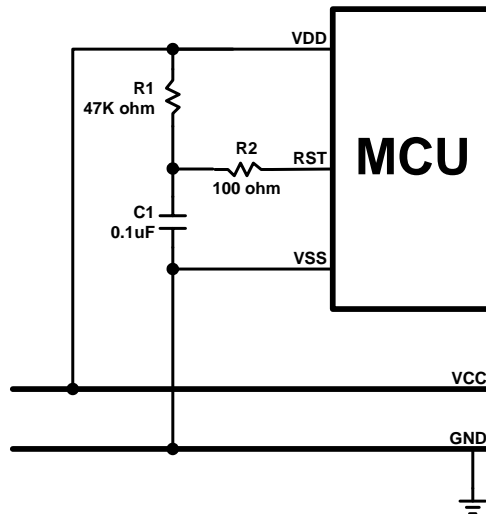
外部复位功能由编译选项“Reset\_Pin”控制。将该编译选项置为“Reset”，可使能外部复位功能。外部复位引脚为施密特触发结构，低电平有效。复位引脚处于高电平时，系统正常运行。当复位引脚输入低电平信号时，系统复位。外部复位操作在上电和正常工作模式时有效。需要注意的是，在系统上电完成后，外部复位引脚必须输入高电平，否则系统将一直保持在复位状态。外部复位的时序如下：

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

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

## 3.6 外部复位电路

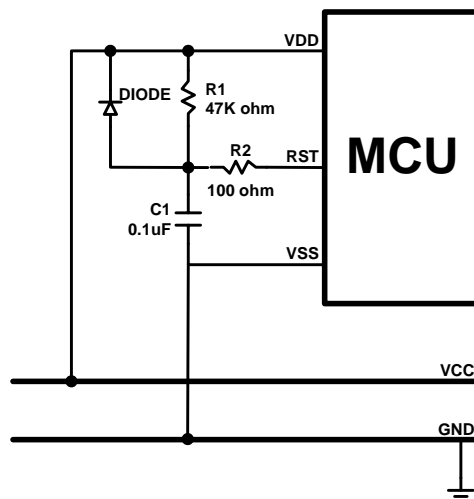
### 3.6.1 RC 复位电路



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

\* 注：此 RC 复位电路不能解决非正常上电和掉电复位问题。

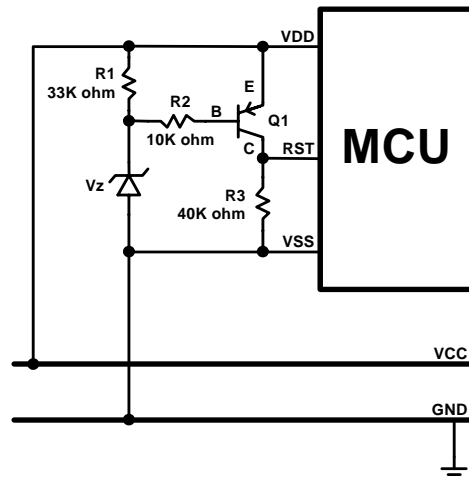
### 3.6.2 二极管及 RC 复位电路



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

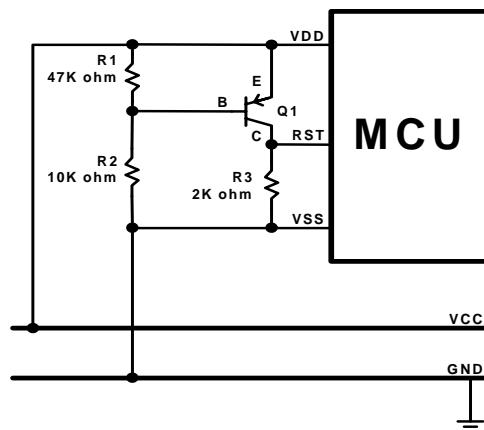
\* 注：“基本 RC 复位电路”和“二极管及 RC 复位电路”中的电阻 R2 都是必不可少的限流电阻，以避免复位引脚 ESD (Electrostatic Discharge) 或 EOS (Electrical Over-stress) 击穿。

### 3.6.3 稳压二极管复位电路



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

### 3.6.4 电压偏置复位电路

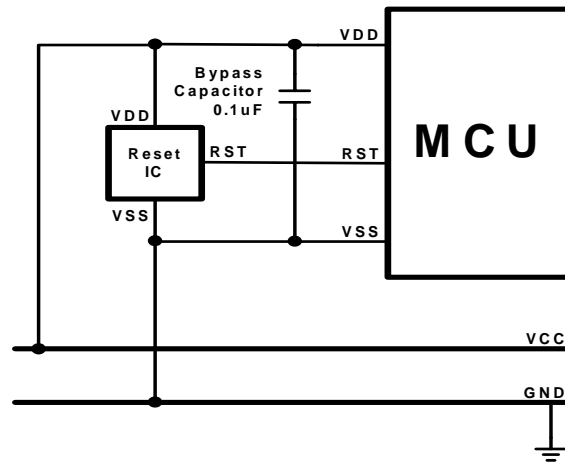


电压偏置复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，这种复位电路的检测电压值的精确度有所降低。电路中，R1 和 R2 构成分压电路，当 VDD 高于和等于分压值“ $0.7V \times (R1 + R2) / R1$ ”时，三极管集电极 C 输出高电平，单片机正常工作；VDD 低于“ $0.7V \times (R1 + R2) / R1$ ”时，集电极 C 输出低电平，单片机复位。

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

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

### 3.6.5 外部 IC 复位



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

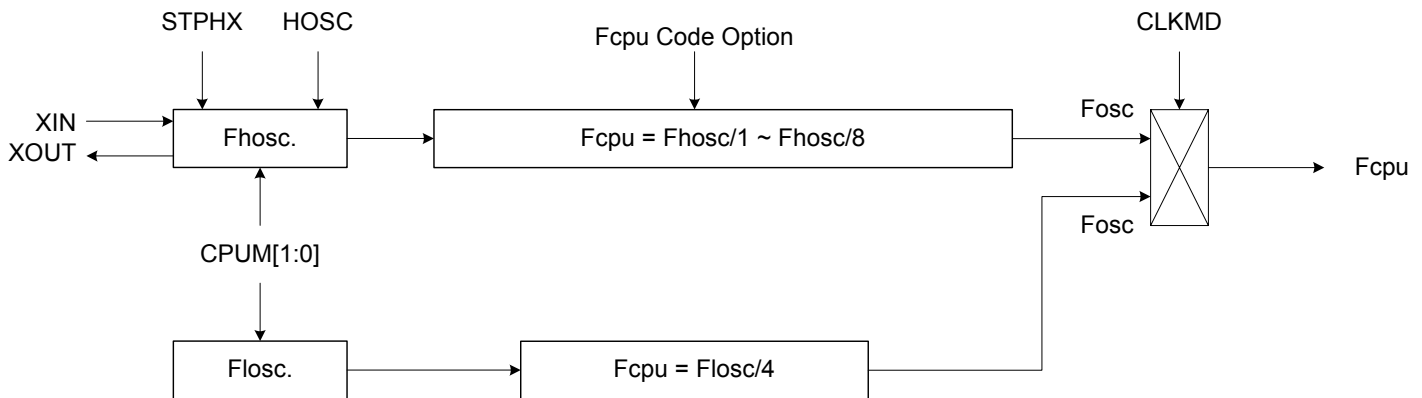
# 4 系统时钟

## 4.1 概述

26L00系列单片机内置双时钟系统：高速时钟和低速时钟。高速时钟由外部晶振或内置的8MHz RC 振荡电路（IHRC 8MHz）提供。低速时钟由内置的低速RC振荡电路（ILRC 10KHz@3V）提供。两种时钟都可作为系统时钟源Fosc，系统工作在低速模式时，Fosc 4分频后作为一个指令周期。

- ☞ 普通模式 (高速时钟):  $F_{cpu} = F_{osc} / N$ ,  $N = 1 \sim 8$ ,  $F_{cpu}$  的编译选项决定 N 的值。
- ☞ 低速模式 (低速时钟):  $F_{cpu} = F_{osc}/4$ 。

## 4.2 时钟框图



- **HOSC:** High\_Clk 编译选项。
- **Fhosc:** 外部高速时钟频率/ 内部高速 RC 时钟频率。
- **Fosc:** 内部低速 RC 时钟频率 (10KHz@3V)。
- **Fosc:** 系统时钟频率。
- **Fcpu:** 指令执行频率。

## 4.3 Fcpu（指令周期）

系统时钟周期（Fosc）分频后可作为指令周期（Fcpu），它控制系统的执行速率。对于同一时钟周期，分频数不同，指令周期也不同，可通过编译选项在  $F_{osc}/1 \sim F_{osc}/8$  范围之间选择。如果系统高速时钟源为外部 16MHz 晶振，编译选项中选择 Fcpu 为  $F_{osc}/4$ ，那么 Fcpu 的频率即为  $16\text{MHz}/4 = 4\text{MHz}$ 。在低速模式下，由于 Fcpu 被限制为  $F_{osc}/4$ ，因而不能通过编译选项来选择频率。

在高干扰环境下，强烈建议 Fcpu 编译选项选择  $F_{osc}/4$  以下的选项。



## 4.4 OSCM 寄存器

寄存器 OSCM 控制振荡器的状态和系统的工作模式。

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

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

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

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

STPHX 标志位控制内部高速 RC 振荡器和外部振荡器工作。STPHX=0 时, 外部振荡器或内部 RC 振荡器工作; STPHX=1 时, 外部振荡器和内部高速 RC 振荡器停止工作。所选择的振荡器不同, STPHX 所代表的意义也不同。

High\_CLK = IHRC\_8M: “STPHX=1” 停止内部高速 RC 振荡器。

High\_CLK = IHRC\_RTC: “STPHX=1” 停止内部高速 RC 振荡器及外部 32768Hz 晶振。

High\_CLK = RC, 4M, 8M, 32K: “STPHX=1” 停止外部振荡器。

## 4.5 系统高速时钟

系统高速时钟为内部 8MHz RC 振荡器或外部振荡器，高速时钟可通过“High\_Clk”编译选项选择。

- **IHRC\_8M:** 系统时钟源为内部高速 RC 振荡器。在该模式下，XIN 和 XOUT 引脚均可作为普通 I/O 引脚（双向），且不与其它外部振荡器连接。
- **IHRC\_RTC:** 系统时钟源为内部高速 8MHz RC 振荡器，外接外部低速 32768Hz 晶振时，单片机具有实时时钟(RTC)功能。此时，XIN 和 XOUT 引脚与 32768Hz 晶振连接，引脚不具有普通 I/O 功能。
- **RC:** 系统时钟源为外部低速 RC 振荡器。RC 振荡电路与 XIN 引脚相连，此时 XOUT 引脚为普通 I/O 模式。
- **32K X'tal:** 系统时钟源为外部低速 32768Hz 晶振，该选项仅支持 32768Hz 晶振。
- **8M X'tal:** 系统时钟源为外部高速晶振/陶振。支持振荡器频率范围为：8MHz~16MHz。
- **4M X'tal:** 系统时钟源为外部高速晶振/陶振。支持振荡器频率范围为：1MHz~10MHz。

### 4.5.1 内部高速 RC 振荡器

编译选项“IHRC\_8M”和“IHRC\_RTC”控制单片机的内置 RC 高速时钟（8MHz）。若选择“IHRC\_8M”，则内置 8MHz RC 振荡器作为系统时钟源，XIN 和 XOUT 引脚作为通用 I/O 口。若选择“IHRC\_RTC”，则系统时钟来自内置 8MHz RC 振荡器，同时 XIN 和 XOUT 引脚与实时时钟源（RTC）32768Hz 石英晶体相连。

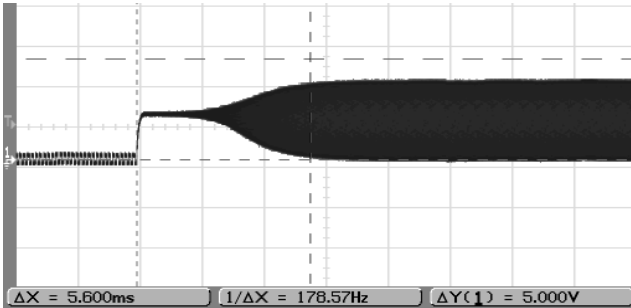
- **IHRC:** 系统高速时钟来自内置 8MHz RC 振荡器，XIN/XOUT 引脚作为普通的 I/O 引脚。
- **IHRC\_RTC:** 内置 8MHz RC 振荡器作为系统高速时钟，同时 XIN 和 XOUT 引脚与实时时钟源（RTC）32768Hz 石英晶体相连。

RTC 的周期为 0.5 s，RTC 的定时器为 T0。

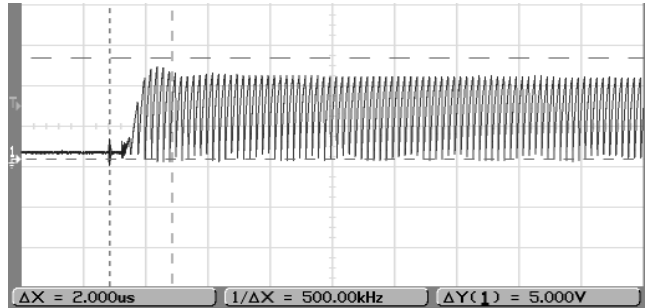
## 4.5.2 外部高速时钟

外部高速时钟共三种模式：石英/陶瓷振荡器，RC 及外部时钟源，由编译选项 High\_Clk 控制具体模式的选择。石英/陶瓷振荡器和 RC 振荡器的上升时间各不相同。RC 振荡器的上升时间相对较短。振荡器上升时间与复位时间的长短密切相关。

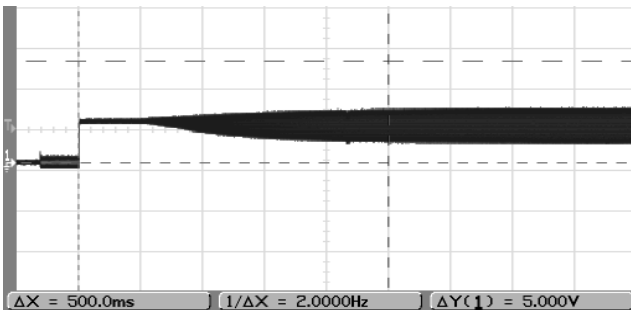
4MHz Crystal



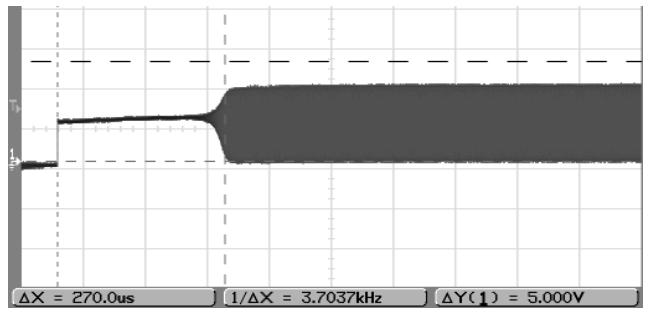
RC



32768Hz Crystal

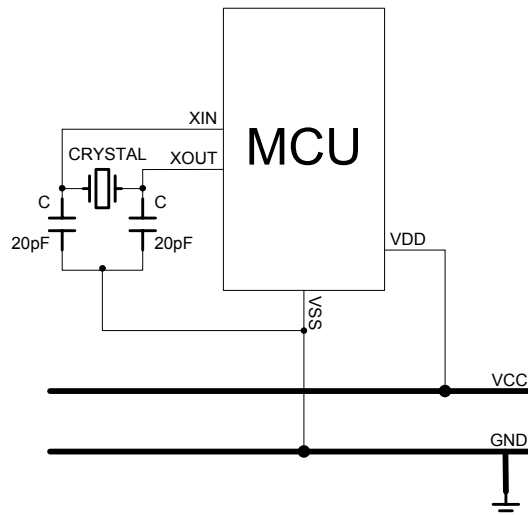


4MHz Ceramic



### 4.5.2.1 石英/陶瓷振荡器

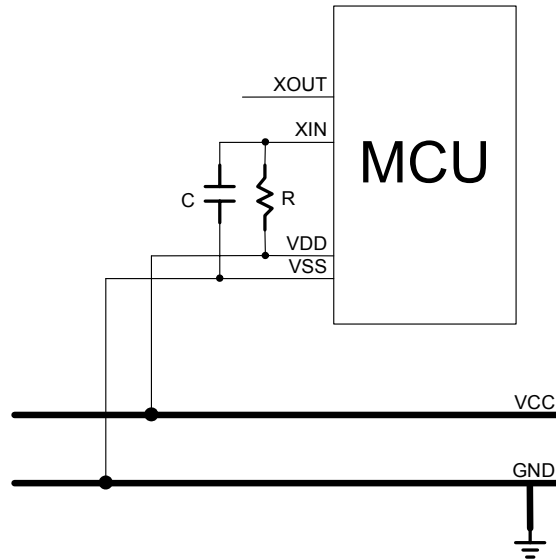
石英/陶瓷振荡器由 XIN/XOUT 口驱动，对于高速、普通和低速三种不同工作模式，振荡器的驱动电流也不同。不同的工作模式下，编译选项 High\_Clk 支持不同的频率条件：8MHz、4MHz 以及 32KHz 工作频率。



\* 注：上图中，XIN/XOUT/VSS 引脚与石英/陶瓷振荡器以及电容 C 之间的距离越近越好。

### 4.5.2.2 RC 振荡器

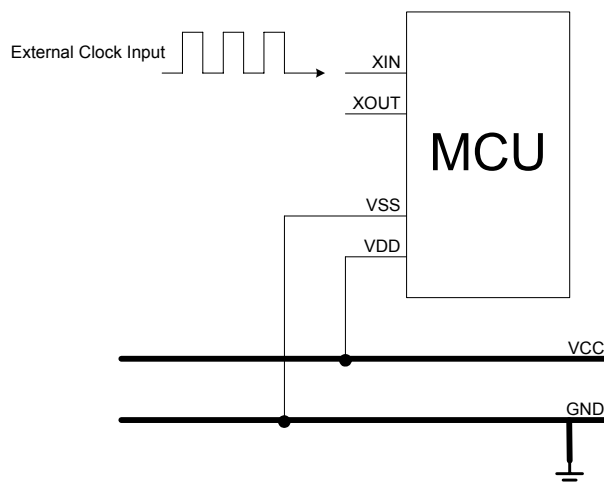
通过编译选项 High\_Clk 的设置可控制 RC 振荡器的选择，RC 振荡器输出频率最高可达 8MHz。改变 R 可改变输出频率的大小，电容 C 的最佳容量为 50pF~100pF，引脚 XOUT 为通用 I/O 口，如下图所示：



\* 注：电容 C 和电阻 R 应尽可能的接近单片机的 VDD。

### 4.5.2.3 外部时钟源

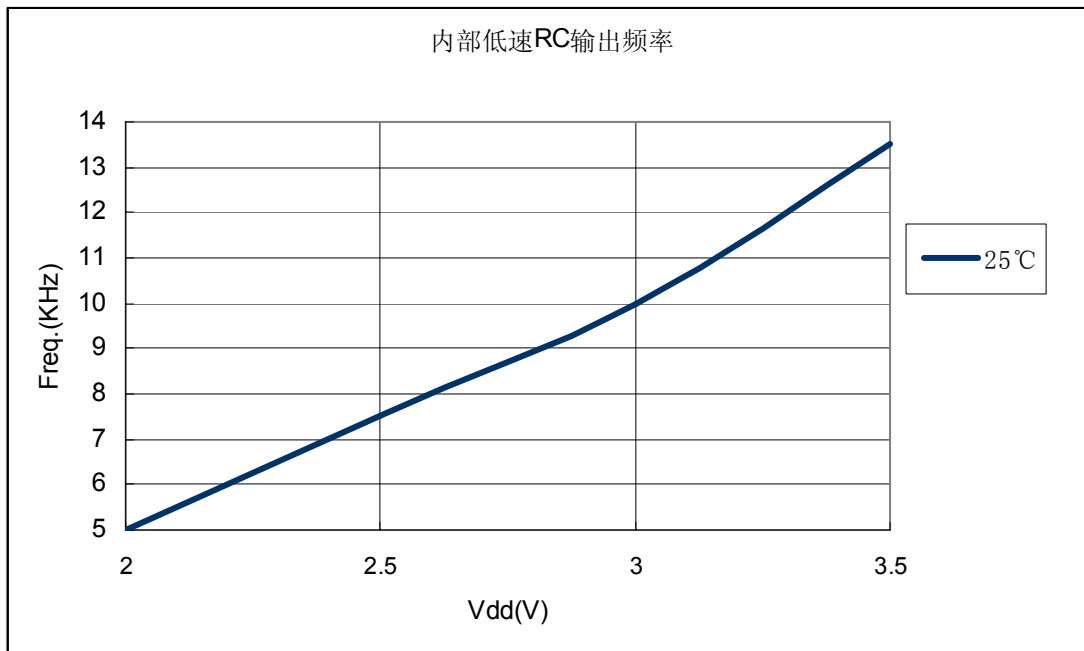
单片机可选择外部时钟信号作为系统时钟，由编译选项 High\_Clk 控制，从 XIN 脚送入。



\* 注：外部振荡电路中的 GND 必须尽可能的接近单片机的 VSS 端口。

## 4.6 系统低速时钟

系统低速时钟源即内置的低速振荡器，采用 RC 振荡电路。低速时钟的输出频率受系统电压和环境温度的影响，通常为 3V 时输出 10KHZ。输出频率与工作电压之间的关系如下图所示。



低速时钟可作为看门狗定时器的时钟源和低速模式的时钟源，由 CLKMD 控制系统低速工作模式。

- ☞ **Flosc = 内部低速 RC 振荡器 (10KHz @3V)。**
- ☞ **低速模式 Fcpu = Flosc / 4。**

睡眠模式可以停止低速 RC 振荡器；同样，在绿色模式下，使用 455K 振荡模式且禁止看门狗时，也可以停止低速 RC 振荡器，此时只有 455K 振荡器在进行工作。

- **例：停止内部低速振荡器。**  
B0BSET      FCPUM0

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

### 4.6.1 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。

- **例：外部振荡器的 Fcpu 指令周期测试。**  
B0BSET      P0M.0                      ; P0.0 置为输出模式以输出 Fcpu 的触发信号。
- @@:  
B0BSET      P0.0  
B0BCLR      P0.0  
JMP            @B

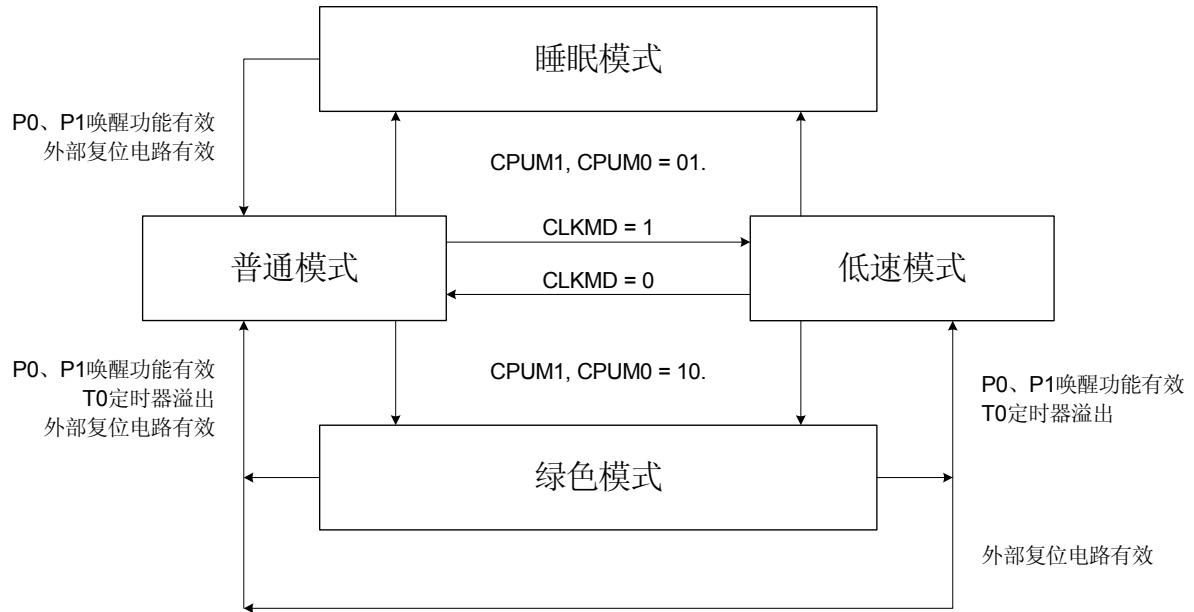
\* 注：不能直接从 XIN 引脚测试 RC 振荡频率，因为探针的连接会影响测试的准确性。

# 5 系统工作模式

## 5.1 概述

SN8P26L00 系列的单片机可在如下四种工作模式之间进行切换：

- 普通模式（高速模式）；
- 低速模式；
- 省电模式（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	注释
EHOSC	运行	由 STPHX 控制	由 STPHX 控制	停止	
IHRC	运行	由 STPHX 控制	由 STPHX 控制	停止	
ILRC	运行	运行	运行	停止	
EHOSC (带 RTC 功能)	运行	由 STPHX 控制	运行	停止	
IHRC (带 RTC 功能)	运行	由 STPHX 控制	停止	停止	
ILRC (带 RTC 功能)	运行	运行	停止	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
TC1 定时器	*有效	*有效	无效	无效	* TC1ENB=1 时有效
看门狗定时器	由 Watch_Dog 编译选项控制	由 Watch_Dog 编译选项控制	由 Watch_Dog 编译选项控制	由 Watch_Dog 编译选项控制	请参阅编译选项章节
内部中断	全部有效	全部有效	T0	全部无效	
外部中断	全部有效	全部有效	全部有效	全部无效	
唤醒功能	-	-	P0, P1, T0, 复位	P0, P1, 复位	

- **EHOSC**: 外部高速时钟；
- **IHRC**: 内部高速时钟(8M RC 振荡器)；
- **ILRC**: 内部低速时钟 (RC 振荡器: 3V 时 10K)。

## 5.2 普通模式

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

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

## 5.3 低速模式

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

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

## 5.4 睡眠模式

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

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

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

## 5.5 绿色模式

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

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

\* 注：sonix 提供宏“GreenMode”来控制绿色模式的工作状态，必要时使用宏“GreeMode”进绿色模式。该宏共有 3 条指令。但在使用 BRANCH 指令（如 BTS0、BTS1、B0BTS0、B0BTS1、INCS、INCMS、DECS、DECMS、CMPRS、JMP）时必须注意宏的长度，否则程序会出错。



## 5.6 工作模式控制宏

Sonix 提供工作模式控制宏以方便系统工作模式的切换。

宏名称	长度	说明
<b>SleepMode</b>	1-word	系统进入睡眠模式。
<b>GreenMode</b>	3-word	系统进入绿色模式。
<b>SlowMode</b>	2-word	系统进入低速模式并停止高速振荡器。
<b>Slow2Normal</b>	5-word	系统从低速模式返回到普通模式。该宏包括工作模式的切换,使能高速振荡器,高速振荡器唤醒延迟时间。

- 例: 从普通模式/低速模式切换进入睡眠模式。

```
SleepMode ; 直接宣告 "SleepMode" 宏。
```

- 例: 从普通模式切换进入低速模式。

```
SlowMode ; 直接宣告 "SlowMode" 宏。
```

- 例: 从低速模式切换进入普通模式 (外部高速振荡器停止工作)。

```
Slow2Normal ; 直接宣告 "Slow2Normal" 宏。
```

- 例: 从普通/低速模式切换进入绿色模式。

```
GreenMode ; 直接宣告 "GreenMode" 宏。
```

- 例: 从普通/低速模式切换进入绿色模式, 并使能 T0 唤醒功能。

; 设置定时器 T0 的唤醒功能。

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

; 进入绿色模式。

```
GreenMode ; 直接宣告 "GreenMode" 宏。
```

- 例: 从普通/低速模式切换进入绿色模式, 并使能 T0 的 RTC 功能和唤醒功能。

```
CLR       T0C        ; 清定时器 T0。
B0BSET    FT0ENB    ; 使能 T0 定时器。
```

; 进入绿色模式。

```
GreenMode ; 直接宣告 "GreenMode" 宏。
```

## 5.7 唤醒时间

### 5.7.1 概述

在绿色模式和睡眠模式下，系统并不执行程序指令，唤醒触发信号能够将系统唤醒到普通模式或低速模式。这里的唤醒触发信号包括外部触发信号（P0、P1 引脚的电平变化）和内部触发信号（T0 溢出信号），具体为：

- 从睡眠模式唤醒后只能进入普通模式，且将其唤醒的触发只能是外部触发信号（P0、P1 电平变化）；
- 由绿色模式唤醒回到系统前一工作模式（普通模式或低速模式）可以用外部触发或者内部触发。

### 5.7.2 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 2048 个外部高速振荡器时钟周期以使振荡电路进入稳定工作状态，等待的这一段就称为唤醒时间。唤醒时间结束后，系统才进入到普通模式。

\* 注：将系统从绿色模式中唤醒是不需要唤醒时间的，因为在绿色模式下高速时钟仍然正常工作。

唤醒时间的计算如下：

$$\text{唤醒时间} = 1/\text{Fosc} * 2048 \text{ (sec)} + \text{高速时钟启动时间}$$

\* 注：高速时钟的启动时间与 VDD 和振荡器类型有关。

➤ 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。

$$\text{唤醒时间} = 1/\text{Fosc} * 2048 = 0.512 \text{ ms (Fosc = 4MHz)}$$

$$\text{总的唤醒时间} = 0.512 \text{ ms} + \text{振荡器启动时间}$$

### 5.7.3 P1W 唤醒功能控制寄存器

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 都具有唤醒功能，二者区别在于：P0 的唤醒功能始终有效，而 P1 的唤醒功能则由寄存器 P1W 控制。

0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P1W</b>	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

Bit[7:0]

**P10W~P17W:** P1 唤醒功能控制位。

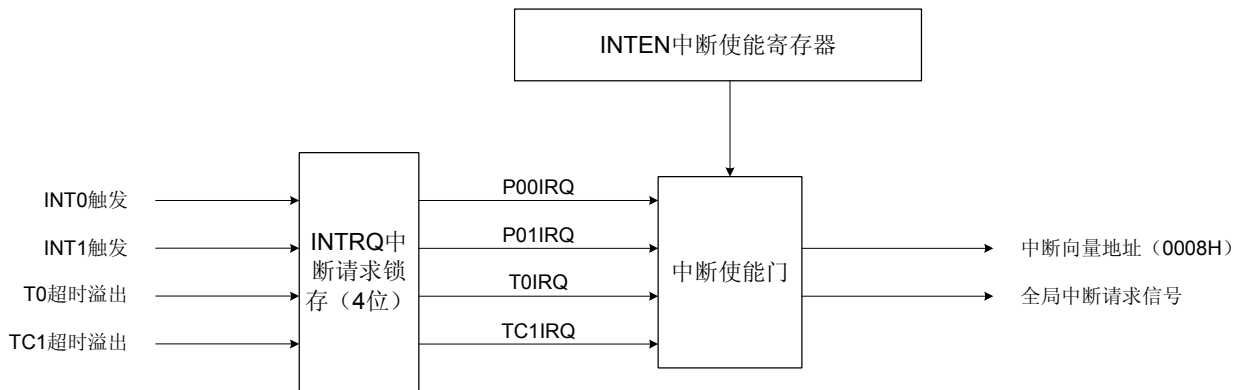
0 = 禁止；

1 = 使能。

# 6 中断

## 6.1 概述

SN8P26L00 提供 4 个中断源，2 个内部中断（T0/TC1）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 中断请求使能寄存器 INTEN

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

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>INTEN</b>	-	TC1IEN	-	TOIEN	-	-	P01IEN	P00IEN
读/写	-	R/W	-	R/W	-	-	R/W	R/W
复位后	-	0	-	0	-	-	0	0

Bit 0 **P00IEN**: P0.0 外部中断 (INT0) 控制位。

0 = 无效;  
1 = 有效。

Bit 1 **P01IEN**: P0.1 外部中断 (INT1) 控制位。

0 = 无效;  
1 = 有效。

Bit 4 **TOIEN**: T0 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。

## 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
<b>INTRQ</b>	-	TC1IRQ	-	T0IRQ	-	-	P01IRQ	P00IRQ
读/写	-	R/W	-	R/W	-	-	R/W	R/W
复位后	-	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 = INT0 无中断请求;  
1 = INT0 有中断请求。

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

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

## 6.4 GIE 全局中断

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

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>STKP</b>	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)。  
BOBSET FGIE ; 使能 GIE。

\* 注: 在所有中断中, GIE 都必须处于使能状态。

## 6.5 PUSH, POP 处理

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

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

➤ 例：对 ACC 和 PAFLG 进行入栈保护。

```
                ORG      0
                JMP      START

                ORG      8H
                JMP      INT_SERVICE

START:          ORG      10H
                ...

INT_SERVICE:   PUSH                    ; 保存 ACC 和 PFLAG。
                ...
                POP                    ; 恢复 ACC 和 PFLAG。

                RETI                   ; 退出中断。
                ...
                ENDP
```

## 6.6 INTO (P0.0) 中断

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

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

\* 注：INT0 的中断请求被 P0.0 的唤醒触发功能锁定。

\* 注：P0.0 的中断触发边沿由 PEDGE 控制。

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

Bit[4:3] **P00G[1:0]**: P0.0 中断触发控制位。

00 = 保留；

01 = 上升沿触发；

10 = 下降沿触发；

11 = 上升/下降沿触发（电平触发）。

➤ 例：INT0 中断请求设置，电平触发。

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

B0BCLR  FP00IRQ      ; 清 INTO 中断请求标志。
B0BSET  FP00IEN      ; 使能 INTO 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。
...
; INTO 中断服务程序。
...

EXIT_INT:
...
; 恢复 ACC 和 PFLAG。

RETI
; 退出中断。
```

## 6.7 INT1 (P0.1) 中断

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

\* 注：P0.1 中断由下降沿触发。

### 例：INT1 中断请求设置。

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

### 例：INT1 中断。

```
ORG      8H
JMP     INT_SERVICE
INT_SERVICE:
...
; ACC 和 PFLAG 入栈保护。

B0BTS1  FP01IRQ    ; 检查是否有 P01 中断请求标志。
JMP     EXIT_INT   ; P01IRQ = 0，退出中断。

B0BCLR  FP01IRQ    ; 清 P01IRQ。
...
; INT1 中断服务程序。

EXIT_INT:
...
; ACC 和 PFLAG 出栈恢复。

RETI
; 退出中断。
```

## 6.8 T0 中断

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

### 例：T0 中断请求设置。

```

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

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

B0BSET    FGIE      ; 使能 GIE。

```

### 例：T0 设置为无 RTC。

```

ORG       8H        ;
INT_SERVICE:
JMP      INT_SERVICE

...          ; ACC 和 PFLAG 入栈保存。
B0BTS1   FT0IRQ    ; 检查是否有 T0 中断请求标志。
JMP      EXIT_INT  ;

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

```

#### \* 注：

1. 在 RTC 模式下，必须延迟 1/2 RTC 时钟源（32768Hz）之后再对 T0IRQ 作清零动作，否则 RTC 间隔时间可能出错。即从程序响应 T0 中断开始到 T0IRQ 再次被清零大约需要 16us。
2. RTC 模式下，中断服务程序中不能对 T0C 进行清零。

### 例：RTC 下执行 T0 中断。

```

ORG       8H        ;
INT_SERVICE:
JMP      INT_SERVICE

...          ; ACC 和 PFLAG 中断保护。
> 16us { B0BTS1   FT0IRQ    ; 检测 T0IRQ。
        JMP      EXIT_INT  ; T0IRQ = 0, 退出中断。
        ...          ; T0 中断程序。
        ...          ;
EXIT_INT: B0BCLR   FT0IRQ    ; T0IRQ 清零。
...          ; 恢复 ACC 和 PFLAG。
RETI      ; 退出中断。

```



## 6.9 TC1 中断

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

### ➤ 例：设置 TC1 中断请求。

```

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

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

B0BSET    FGIE       ; 使能 GIE。

```

### ➤ 例：TC1 中断服务程序。

```

ORG      8H          ;
INT_SERVICE:
JMP      INT_SERVICE

...                ; 保存 ACC 和 PFLAG。

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

B0BCLR   FTC1IRQ    ; 清 TC1IRQ。
MOV      A, #64H    ;
B0MOV    TC1C, A     ; 清 TC1C。
...      ; TC1 中断服务程序。
...

EXIT_INT:
...      ; 恢复 ACC 和 PFLAG。

RETI     ; 退出中断。

```

## 6.10 多中断操作举例

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

中断	有效触发
P00IRQ	PEDGE 控制
P01IRQ	下降沿触发
T0IRQ	T0C 溢出
TC1IRQ	TC1C 溢出

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

➤ 例：多中断条件下检测中断请求。

```

                ORG          8H          ;
                JMP          INT_SERVICE
INT_SERVICE:
                ...                    ; 保存 ACC 和 PFLAG。

INTP00CHK:
                B0BTS1      FP00IEN    ; 检查是否有 P00 中断请求。
                JMP          INTP01CHK  ; 检查是否使能 P00 中断。
                B0BTS0      FP00IRQ    ; 跳到下一个中断。
                JMP          INTP00     ; 检查是否有 P00 中断请求。
                ...                    ; 进入 INT0 中断。

INTP01CHK:
                B0BTS1      FP01IEN    ; 检查是否有 P01 中断请求。
                JMP          INTT0CHK   ; 检查是否使能 P01 中断。
                B0BTS0      FP01IRQ    ; 跳到下一个中断。
                JMP          INTP01     ; 检查是否有 P01 中断请求。
                ...                    ; 进入 INT1 中断。

INTT0CHK:
                B0BTS1      FT0IEN     ; 检查是否有 T0 中断请求。
                JMP          INTTC1CHK  ; 检查是否使能 T0 中断。
                B0BTS0      FT0IRQ     ; 跳到下一个中断。
                JMP          INTT0      ; 检查是否有 T0 中断请求。
                ...                    ; 进入 T0 中断。

INTTC1CHK:
                B0BTS1      FTC1IEN    ; 检查是否有 TC1 中断请求。
                JMP          INT_EXIT   ; 检查是否使能 TC1 中断。
                B0BTS0      FTC1IRQ    ; 检查是否有 TC1 中断请求。
                JMP          INTTC1     ; 进入 TC1 中断。

INT_EXIT:
                ...                    ; 恢复 ACC 和 PFLAG。

                RETI                    ; 退出中断。

```

# 7 I/O 口

## 7.1 概述

SN8P26L34 内建 26 个 I/O 引脚。大多数 I/O 引脚都和模拟引脚及特殊功能的引脚共用，I/O 引脚的共用情况如下表所示：

I/O 引脚		共用引脚		共用引脚的控制条件
名称	类型	名称	类型	
P0.0	I/O	INT0	DC	P00IEN=1
P0.1	IO	INT1	DC	P01IEN=1
P0.2	I	RST	DC	Reset_Pin = Reset
		VPP	HV	OTP 可烧录
P0.3	I/O	XIN	AC	High_CLK = IHRC_RTC、RC、32K、4M、8M
P0.4	I/O	XOUT	AC	High_CLK = IHRC_RTC、32K、4M、8M
P2.2	I/O	CM0N	AC	CM0EN=1
P2.3	I/O	CM0P	AC	CM0EN=1, CM0REF=0
P2.4	I/O	CM0O	AC	CM0EN=1, CM0OEN=1
P2.5	I/O	CM1N	AC	CM1EN=1
P2.6	I/O	CM1P	AC	CM1EN=1, CM1REF=0
P2.7	I/O	CM1O	AC	CM1EN=1, CM1OEN=1
P5.3	I/O	BZ1/PWM1	DC	TC1ENB=1, TC1OUT=1 或 PWM1OUT=1
P5.4	I/O	IROUT	DC	IREN=1

\* DC: 数字特性, AC: 模拟特性, HV: 高压特性。

## 7.2 I/O 口模式

寄存器 PnM 控制 I/O 的工作模式。

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P0M</b>	-	-	-	P04M	P03M	-	P01M	P00M
读/写	-	-	-	R/W	R/W	-	R/W	R/W
复位后	-	-	-	0	0	-	0	0

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P1M</b>	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
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

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

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P5M</b>	-	-	-	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 模式选择控制位 (n = 0~5)。

0 = 输入模式;

1 = 输出模式。

**\* 注:**

1. 用户可通过位操作指令 (B0BSET、B0BCLR) 对 I/O 口进行编程控制;
2. P0.2 是单向输入引脚, P0M.2 保持为“1”。

➤ **例: I/O 模式选择。**

```
CLR      P0M          ; 设置为输入模式。
CLR      P1M
CLR      P5M
```

```
MOV      A, #0FFH    ; 设置为输出模式。
B0MOV    P0M, A
B0MOV    P1M, A
B0MOV    P5M, A
```

```
B0BCLR   P1M.2       ; 设置 P1.2 为输入模式。
```

```
B0BSET   P1M.2       ; 设置 P1.2 为输出模式。
```

## 7.3 I/O 上拉电阻

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P0UR</b>	-	-	-	P04R	P03R	-	P01R	P00R
读/写	-	-	-	W	W	-	W	W
复位后	-	-	-	0	0	-	0	0

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P1UR</b>	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0E2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P2UR</b>	P27R	P26R	P25R	P24R	P23R	P22R	P21R	P20R
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0E5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P5UR</b>	-	-	-	P54R	P53R	P52R	P51R	P50R
读/写	-	-	-	W	W	W	W	W
复位后	-	-	-	0	0	0	0	0

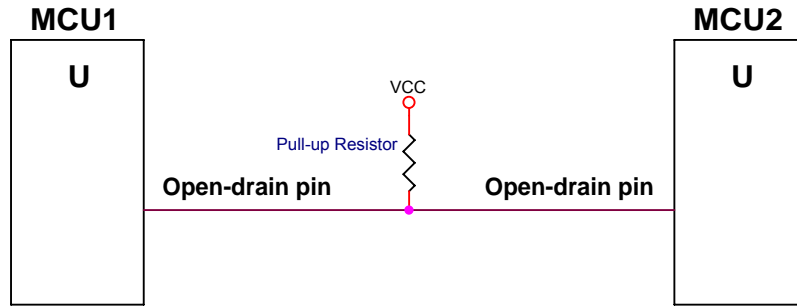
\* 注：P0.2 为单向输入引脚，且无上拉电阻，P0UR.2 保持为“1”。

➤ 例：I/O 口的上拉电阻。

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

## 7.4 I/O 漏极开路寄存器

P1.0/P1.1 内置漏极开路功能，当使能该功能时，P1.0/P1.1 必须设置为输出模式。漏极开路的外部电路如下图所示：



上图中的上拉电阻必不可少，漏极开路的输出高电平由上拉电阻驱动。

0E9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P10C</b>	-	-	-	-	-	-	P110C	P100C
读/写	-	-	-	-	-	-	W	W
复位后	-	-	-	-	-	-	0	0

Bit [1:0] **P110C~P100C**: P1.1 和 P1.0 漏极开路控制位。

0 = 禁止漏极开路功能；

1 = 使能漏极开路功能。

➤ 例：使能 P1.0 的漏极开路功能并输出高电平。

```

B0BSET      P1.0          ; P1.0 置高。

B0BSET      P10M          ; P1.0 设为输出模式。
MOV         A, #01H       ; 使能 P1.0 的漏极开路功能。
B0MOV       P10C, A

```

\* 注：P100C 是只写寄存器，只能通过指令“MOV”设置 P100C。

➤ 例：禁止 P1.0 的漏极开路功能并输出低电平。

```

MOV         A, #0          ; 禁止 P1.0 的漏极开路功能。
B0MOV       P10C, A

```

\* 注：禁止 P1 的漏极开路功能后，P1 返回到之前的 I/O 模式。

## 7.5 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
<b>P0</b>	-	-	-	P04	P03	P02	P01	P00
读/写	-	-	-	R/W	R/W	R	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
<b>P1</b>	P17	P16	P15	P14	P13	P12	P11	P10
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

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

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>P5</b>	-	-	-	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

\* 注：当使能外部复位功能后，P02 保持为“1”。

➤ 例：读取输入口的数据。

```
B0MOV      A, P0           ; 读取 P0、P1 和 P5 口的数据。
B0MOV      A, P1
B0MOV      A, P5
```

➤ 例：写入数据到输出端口。

```
MOV        A, #0FFH       ; 写入数据 FFH 到 P0、P1 和 P5。
B0MOV      P0, A
B0MOV      P1, A
B0MOV      P5, A
```

➤ 例：写入 1 位数据到输出端口。

```
B0BSET     P1.3           ; P1.3 和 P5.4 置“1”。
B0BSET     P5.4

B0BCLR     P1.3           ; P1.3 和 P5.4 置“0”。
B0BCLR     P5.4
```

# 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(10KHz @3V) 提供，它是将内部 RC 振荡器经 512 分频后送往看门狗定时器进行计数。

看门狗溢出时间 = 8192 / 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
3V	10KHz	819.2ms

看门狗定时器由编译选项控制，在编译选项 (code option) 中，用户可以通过 WatchDog 的三个配置选项，选择开启或关闭 WatchDog。

- **Disable:** 关闭看门狗定时器。
- **Enable:** 开启看门狗定时器。看门狗在普通模式和低速模式时开启，在睡眠模式和绿色模式下关闭。
- **Always\_On:** 开启看门狗定时器。看门狗始终处于开启状态，睡眠模式和绿色模式下也开启。

在干扰严重的环境下，强烈建议用户选择看门狗定时器“Always\_On”选项，系统出错后会自动复位并重新运行。

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。

OCCH	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。
    ...                ;
    MOV                A, #5AH           ; 清看门狗定时器。
    B0MOV              WDTR, A
    ...
    CALL               SUB1
    CALL               SUB2
    ...
    ...
    JMP                MAIN

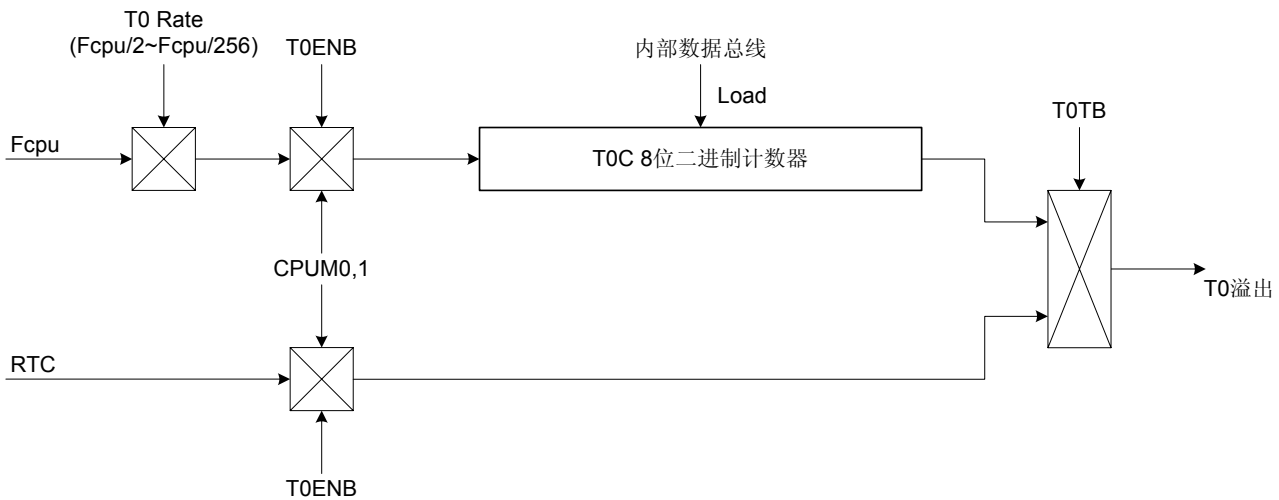
```

## 8.2 定时器 T0

### 8.2.1 概述

二进制定时器 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 中断请求。定时器 T0 的主要用途如下：

- ☞ **8 位可编程定时器**：根据选定的时钟频率定时产生中断请求；
- ☞ **RTC 定时器**：根据选定的时钟信号产生中断请求，RTC 功能仅限于编译选项为 **High\_Clk = IHRC\_RTC**；
- ☞ **绿色模式唤醒功能**：T0ENB = 1 时，T0 溢出将系统从绿色模式中唤醒。



\* 注：

- 1、在 RTC 模式下，进入中断之后必须经过 1/2 RTC（32768HZ），即 16us 的延时后才能对 T0IRQ 清零；
- 2、在 RTC 模式下，T0 间隔时间固定为 0.5s，T0C 计数范围也固定为 256。

### 8.2.2 模式寄存器 TOM

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

Bit 0 **T0TB**：RTC 时钟控制位。

- 0 = 禁止 RTC；
- 1 = RTC 模式。

Bit [6:4] **TORATE[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 = 禁止 T0；
- 1 = 开启 T0。

\* 注：在 RTC 模式下，TORATE 的设置是无效的。

## 8.2.3 计数寄存器 T0C

8 位计数寄存器 T0C 用于控制 T0 的间隔时间。

0D9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>T0C</b>	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：

$$\text{T0C 初始值} = 256 - (\text{T0 间隔时间} * \text{输入时钟})$$

➤ 例：T0 间隔时间为 10ms。其中，高速时钟 = 4MHz，Fcpu = Fosc/4，TORATE = 010 (Fcpu/64)。

$$\begin{aligned} \text{T0C 初始值} &= 256 - (\text{T0 间隔时间} * \text{输入时钟信号}) \\ &= 256 - (10\text{ms} * 4\text{MHz} / 4 / 64) \\ &= 256 - (10^{-2} * 4 * 10^6 / 4 / 64) \\ &= 100 \\ &= 64\text{H} \end{aligned}$$

T0 间隔时间列表

TORATE	T0 时钟	高速模式 (Fcpu = 4MHz / 4)		低速模式 (Fcpu = 32768Hz / 4)	
		最大溢出间隔 max	单步计数时间 =max/256	最大溢出间隔 max	单步计数时间 =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：在 RTC 模式下，T0C 的计数范围为 256，请不要随意改动。

## 8.2.4 T0 操作时序

T0 操作时序如下：

☞ T0 停止计数，关闭 T0 中断功能，并将 T0 中断请求标志清零。

```
B0BCLR    FT0ENB    ;
B0BCLR    FT0IEN
B0BCLR    FT0IRQ
```

☞ 设置 T0 速率。

```
MOV      A, #0xxx0000b
;
B0MOV    T0M,A
;
```

☞ T0 时钟选择 Fcpu 或 RTC。

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

或

```
B0BSET    FT0TB    ; 选择 RTC 模式。
```

☞ 设置 T0 中断间隔时间。

```
MOV      A,#7FH
B0MOV    T0C,A
```

☞ 使能 T0 定时中断功能。

```
B0BSET    FT0IEN
```

☞ 使能 T0 定时功能。

```
B0BSET    FT0ENB
```

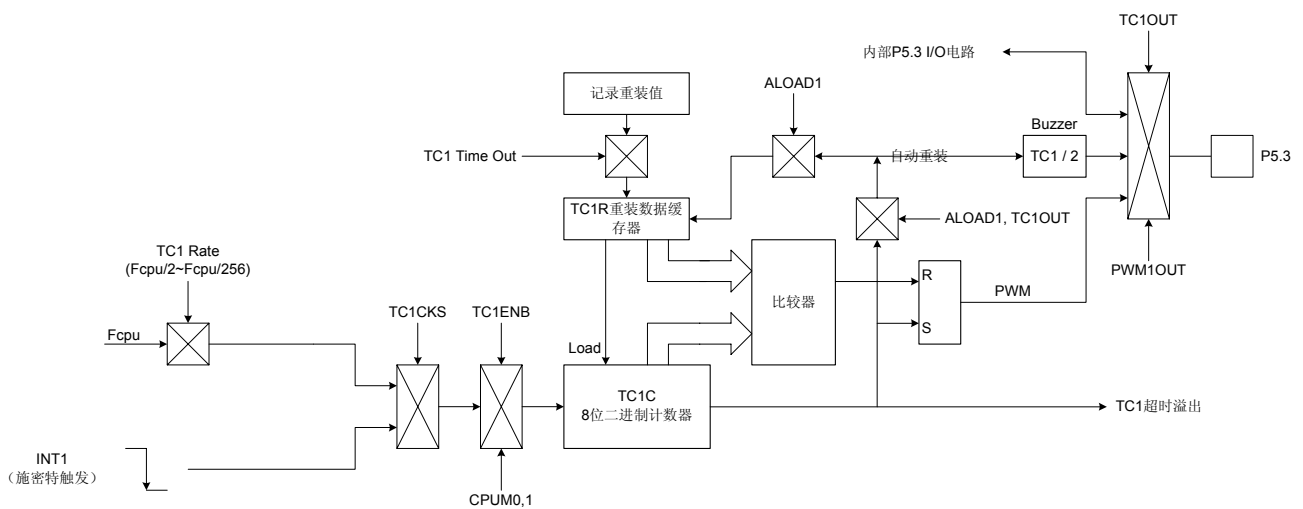
## 8.3 定时/计数器 TC1

### 8.3.1 概述

8 位定时/计数器 TC1 具有双时钟源，可根据实际的需要选择内部时钟或外部时钟作为计时标准。其中，内部时钟来自 Fcpu，外部时钟源 INT1 则由 P0.1 下降沿触发产生。通过寄存器 TC1M 选择 TC1 的时钟源。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。PWM 模式下，TC1 的溢出由 ALOAD1 和 TC1OUT 控制。

TC1 的主要功能如下：

- ☞ **8 位可编程计数定时器：**根据选择的时钟信号，周期性的产生中断；
- ☞ **外部事件计数器：**对外部事件计数；
- ☞ **Buzzer 输出；**
- ☞ **PWM 输出。**



## 8.3.2 TC1M 模式寄存器

ODCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC1M</b>	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 超时输出信号控制。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 TC1 自动重装;  
1 = 允许 TC1 自动重装。
- Bit 3 **TC1CKS**: TC1 时钟源控制位。  
0 = 内部时钟 Fcpu;  
1 = 外部时钟, 由 P0.1/INT1 输入。
- Bit [6:4] **TC1RATE[2:0]**: TC1 分频选择位。  
000 = fcpu/256;  
001 = fcpu/128;  
... ;  
110 = fcpu/4;  
111 = fcpu/2。
- Bit 7 **TC1ENB**: TC1 启动控制位。  
0 = 禁止 TC1 定时器;  
1 = 开启 TC1 定时器。

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

### 8.3.3 TC1C 计数寄存器

TC1C 控制 TC1 的时间间隔。

ODDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC1C</b>	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：

$$\text{TC1C 初始值} = N - (\text{TC1 中断间隔时间} * \text{输入时钟})$$

N 为 TC1 二进制计数范围。各模式下参数的设定如下表所示：

TC1CKS	PWM1	ALOAD1	TC1OUT	N	TC1C 有效值	TC1C 二进制计数范围	注释
0	0	x	x	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出
	1	0	0	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出
	1	0	1	64	00H~3FH	xx000000b~xx111111b	每计数 64 次溢出
	1	1	0	32	00H~1FH	xxx00000b~xxx11111b	每计数 32 次溢出
	1	1	1	16	00H~0FH	xxxx0000b~xxxx1111b	每计数 16 次溢出
1	-	-	-	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出

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

$$\begin{aligned} \text{TC1C 初始值} &= N - (\text{TC1 中断间隔时间} * \text{输入时钟}) \\ &= 256 - (10\text{ms} * 4\text{MHz} / 4 / 64) \\ &= 256 - (10^{-2} * 4 * 10^6 / 4 / 64) \\ &= 100 \\ &= 64\text{H} \end{aligned}$$

TC1 的中断间隔时间列表

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

### 8.3.4 TC1R 自动装载寄存器

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

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

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

ODEH	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：

$$\text{TC1R 初始值} = N - (\text{TC1 中断间隔时间} * \text{输入时钟})$$

N 是 TC1 最大溢出值。TC1 的溢出时间和有效值见下表：

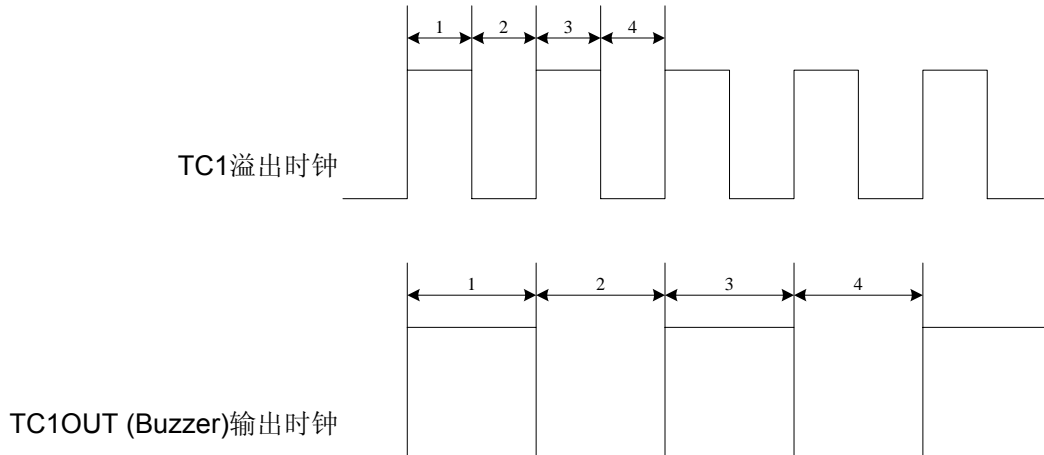
TC1CKS	PWM1	ALOAD1	TC1OUT	N	TC1R 有效值	TC1R 二进制有效范围
0	0	x	x	256	00H~0FFH	00000000b~11111111b
	1	0	0	256	00H~0FFH	00000000b~11111111b
	1	0	1	64	00H~3FH	xx000000b~xx111111b
	1	1	0	32	00H~1FH	xxx00000b~xxx11111b
1	1	1	16	00H~0FH	xxxx0000b~xxxx1111b	
1	-	-	-	256	00H~0FFH	00000000b~11111111b

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

$$\begin{aligned}
 \text{TC1R} &= N - (\text{TC1 中断间隔时间} * \text{输入时钟}) \\
 &= 256 - (10\text{ms} * 4\text{MHz} / 4 / 64) \\
 &= 256 - (10^{-2} * 4 * 10^6 / 4 / 64) \\
 &= 100 \\
 &= 64\text{H}
 \end{aligned}$$

### 8.3.5 TC1 时钟频率输出（蜂鸣器输出）

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  $F_{osc}/4$ ，程序中设置  $TC1RATE2 \sim TC1RATE1 = 110$ ， $TC1C = TC1R = 131$ ，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      ; TC1 速率 = Fcpu/4。
B0MOV    TC1M,A

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

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

```

\* 注：蜂鸣器的输出有效时，“PWM1OUT”必须被置为“0”。



### 8.3.6 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 的速率为 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。
```

or

```
B0BSET      FTC1CKS      ; 外部时钟。
```

☞ 设置 TC1 的自动装载模式。

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

or

```
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
```

or

```
B0BCLR      FALOAD1      ; ALOAD1, TC1OUT = 01, PWM 周期 = 0~63。
B0BSET      FTC1OUT
```

or

```
B0BSET      FALOAD1      ; ALOAD1, TC1OUT = 10, PWM 周期 = 0~31。
B0BCLR      FTC1OUT
```

or

```
B0BSET      FALOAD1      ; ALOAD1, TC1OUT = 11, PWM 周期 = 0~15。
B0BSET      FTC1OUT
```

☞ 设置 TC1 的模式。

```
B0BSET      FTC1IEN      ; 允许 TC1 中断。
```

or

```
B0BSET      FTC1OUT      ; 开启 TC1OUT (Buzzer) 功能。
```

or

```
B0BSET      FPWM1OUT     ; 开启 PWM。
```

☞ 开启 TC1 定时器。

```
B0BSET      FTC1ENB      ;
```

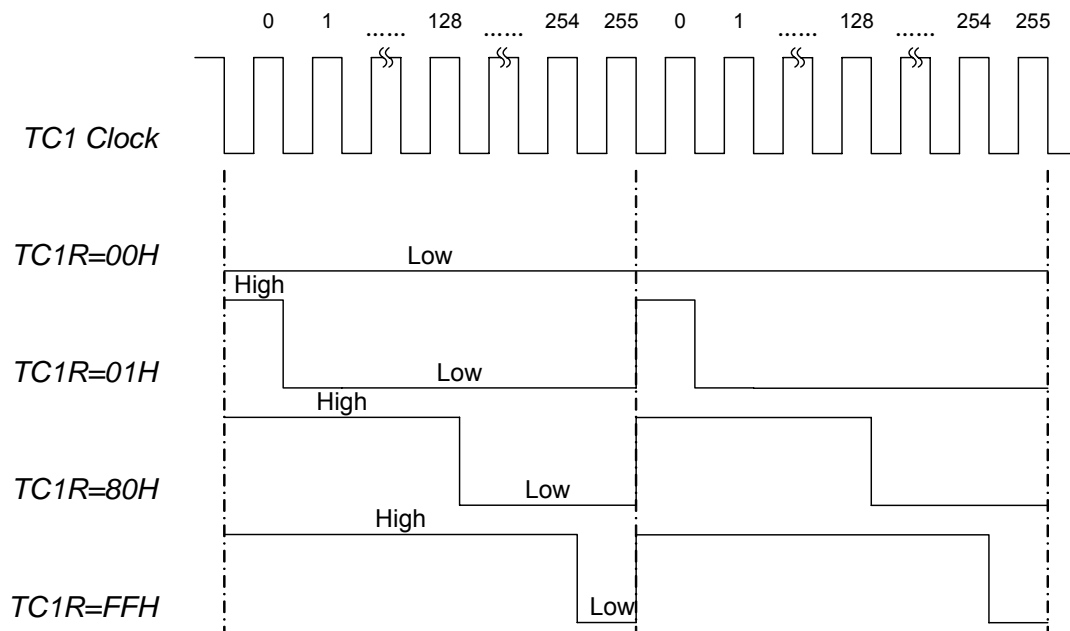
## 8.4 PWM1

### 8.4.1 概述

PWM 信号输出到 PWM1OUT (P5.3 引脚)，位 TC1OUT 和 ALOAD1 控制 PWM 输出的阶数 (256、64、32 和 16)。8 位计数器 TC1C 计数过程中不断与 TC1R 相比较，当 TC1C 计数到两者相等时，PWM 输出低电平，当 TC1C 再次从零开始计数时，PWM 被强制输出高电平。PWM1 输出占空比 = TC1R/阶数 (阶数 = 256、64、32 或 16)。

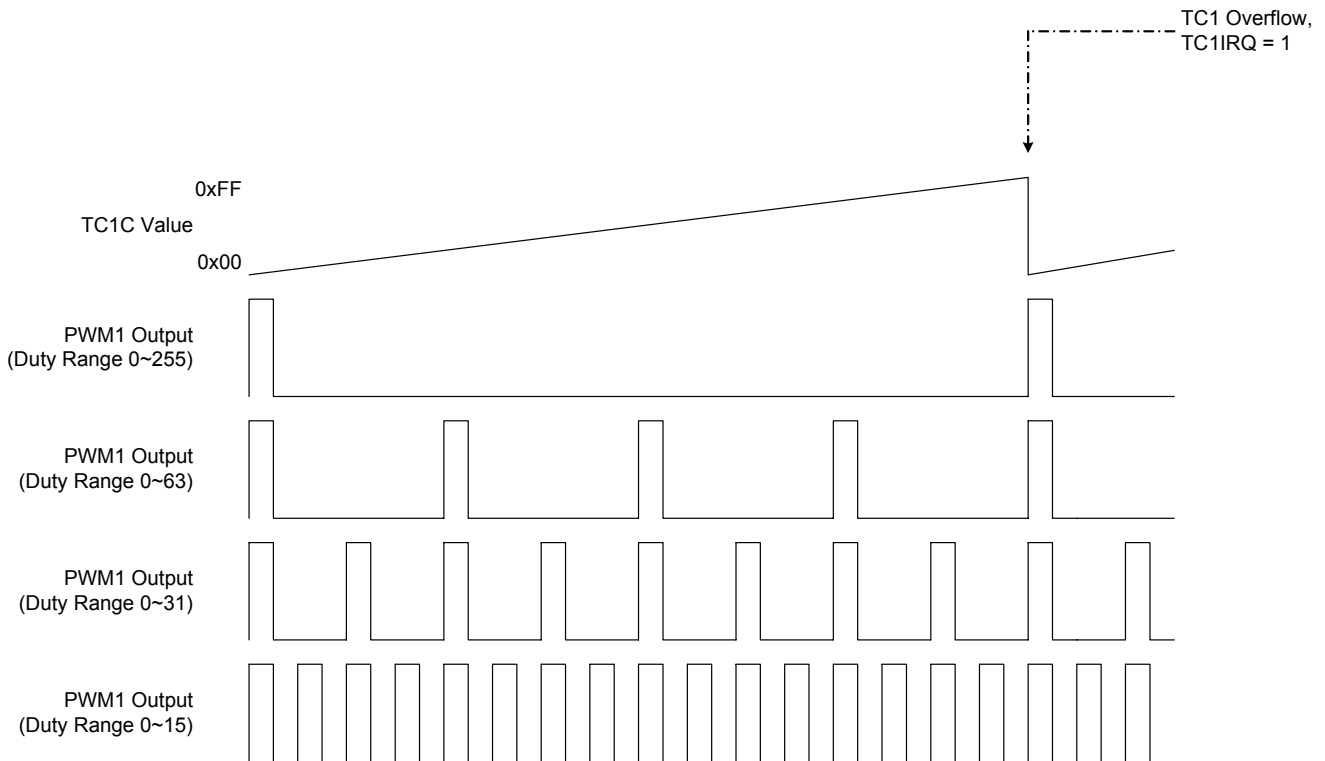
ALOAD1	TC1OUT	PWM 占空比范围	TC1C 有效值	TC1R 有效值	MAX. PWM 频率 (Fcpu = 4MHz)	备注
0	0	0/256~255/256	00H~0FFH	00H~0FFH	7.8125K	每计数 256 次溢出
0	1	0/64~63/64	00H~3FH	00H~3FH	31.25K	每计数 64 次溢出
1	0	0/32~31/32	00H~1FH	00H~1FH	62.5K	每计数 32 次溢出
1	1	0/16~15/16	00H~0FH	00H~0FH	125K	每计数 16 次溢出

PWM 输出占空比随 TC1R 的变化而变化：0/256~255/256。



## 8.4.2 TC1IRQ 和 PWM 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：



## 8.4.3 PWM 编程举例

- 例：PWM1 输出设置。外部高速振荡器输出频率 = 4MHZ,  $F_{cpu} = F_{osc}/4$ , PWM1 输出占空比 = 30/256, 输出频率 1KHZ, PWM1 时钟源来自外部时钟, TC1 速率 =  $F_{cpu}/4$ ,  $TC1RATE2 \sim TC1RATE1 = 110$ ,  $TC1C = TC1R = 30$ 。

```

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

MOV          A,#30
B0MOV       TC1C,A
B0MOV       TC1R,A           ; PWM1 输出占空比=30/256。

B0BCLR      FTC1OUT          ; 占空比变化范围： 0/256~255/256。
B0BCLR      FALOAD1
B0BSET      FPWM1OUT         ; PWM0 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

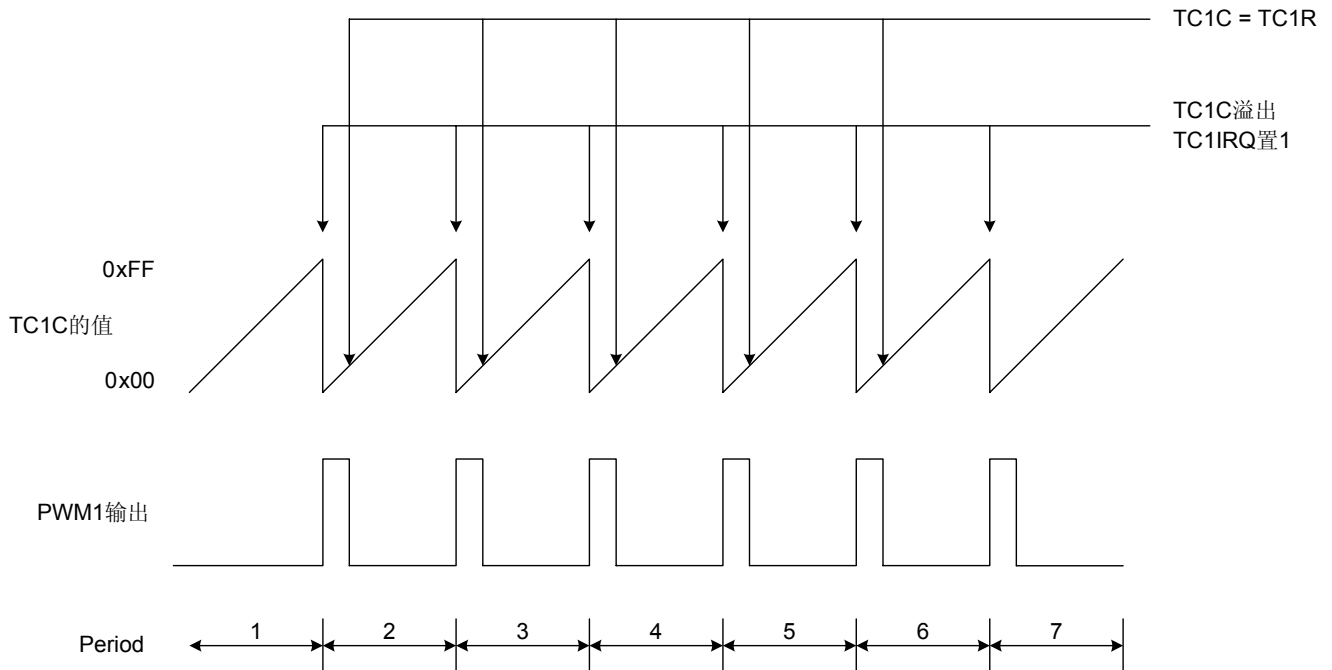
INCMS       BUF0
NOP
B0MOV       A, BUF0
B0MOV       TC1R, A

```

\* 注：PWM 可以在中断下工作。

### 8.4.4 PWM1 占空比注意事项

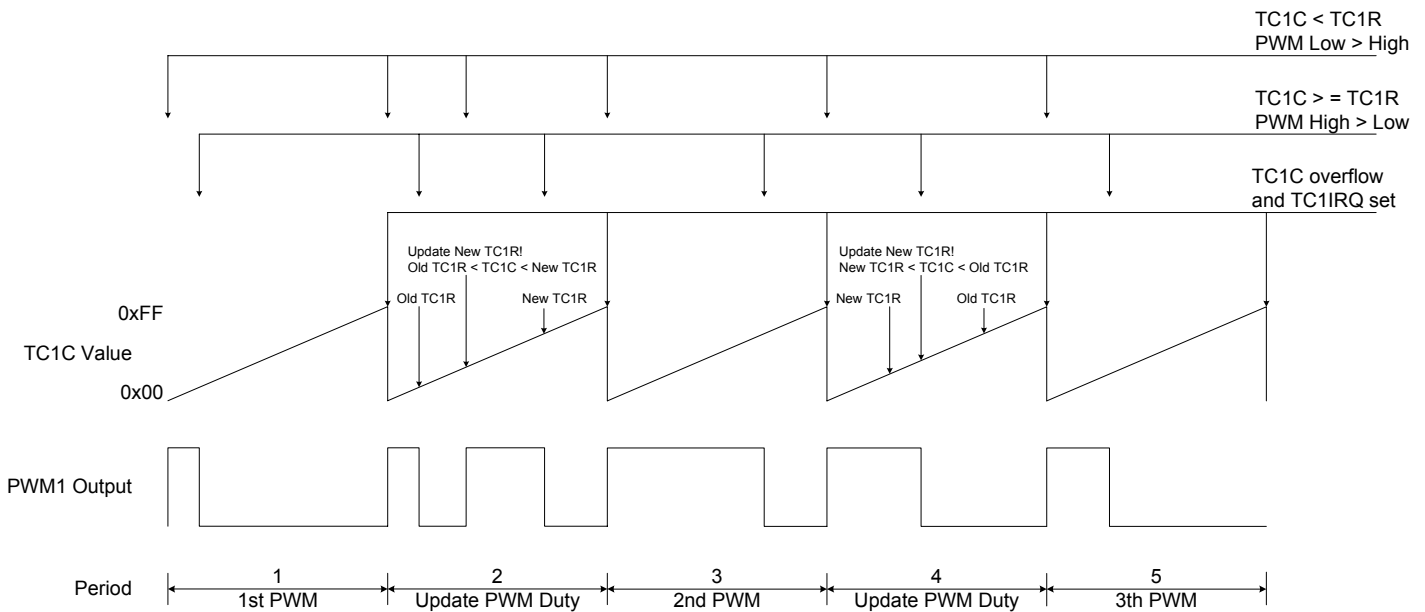
在 PWM 模式下，系统会随时比较 TC1C 和 TC1R 的值。如果  $TC1C < TC1R$ ，PWM 输出高电平，而当  $TC1C \geq TC1R$  时则输出低电平。当 TC1R 发生改变的时候，PWM 的占空比也随着改变，如果 TC1R 保持恒定，那么 PWM 输出波形也保持稳定。



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， $TC1C \geq TC1R$  时，PWM 即输出低电平。

\* 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。

下面所示是 TC1R 发生变化时对应的波形图：



在 period 2 和 period 4 中，设置新的占空比（TC1R），但 period 2 和 period 4 输出波形却是不正确的。在 period 2 中，TC1R 的新值会比旧值略大，如果在 PWM 输出低电平后设置 TC1R 的新值，系统就会获得  $TC1C < TC1R$  的结果，PWM 输出高电平。在一个周期内两次输出高电平，波形图也不是预期的。直到下一个周期，PWM 才会正确输出。在 period 4 中，TC1R 的新值会比旧值略小，在 PWM 输出低电平之前设置 TC1R 的新值，系统就会获得  $TC1C \geq TC1R$  的结果，PWM 输出低电平。在同一个周期内，高电平的占空比小于上一个周期，但却大于高电平占空比的正确值。

虽然不正确的波形图只会在一个周期内存在，但仍然会影响 PWM 工作的精确度并导致外部负载操作出错。在 TC1 定时器溢出后加载 TC1R 的新值时，利用 TC1IRQ 的状态来确定 TC1 定时器是否溢出。若  $TC1IRQ = 1$ ，则开始加载 TC1R 的新值，这样可以避免出现未知的 PWM 输出。

➤ 例：利用 TC1 中断请求标志来决定是否加载 TC1R 的新值以改变 PWM 的输出占空比。

MAIN:

```
...
B0MOV      TC1RBUF, A      ; 加载 PWM 占空比的新值到 TC1RBUF。
...
...
```

INT\_SER:

```
...
... ; 保存 ACC 和 PFLAG。
...
B0BTS1    FTC1IRQ
JMP       INT_SER90
B0MOV     A, TC1RBUF      ; TC1 中断时，更新 TC1R。
B0MOV     TC1R, A
...
...
```

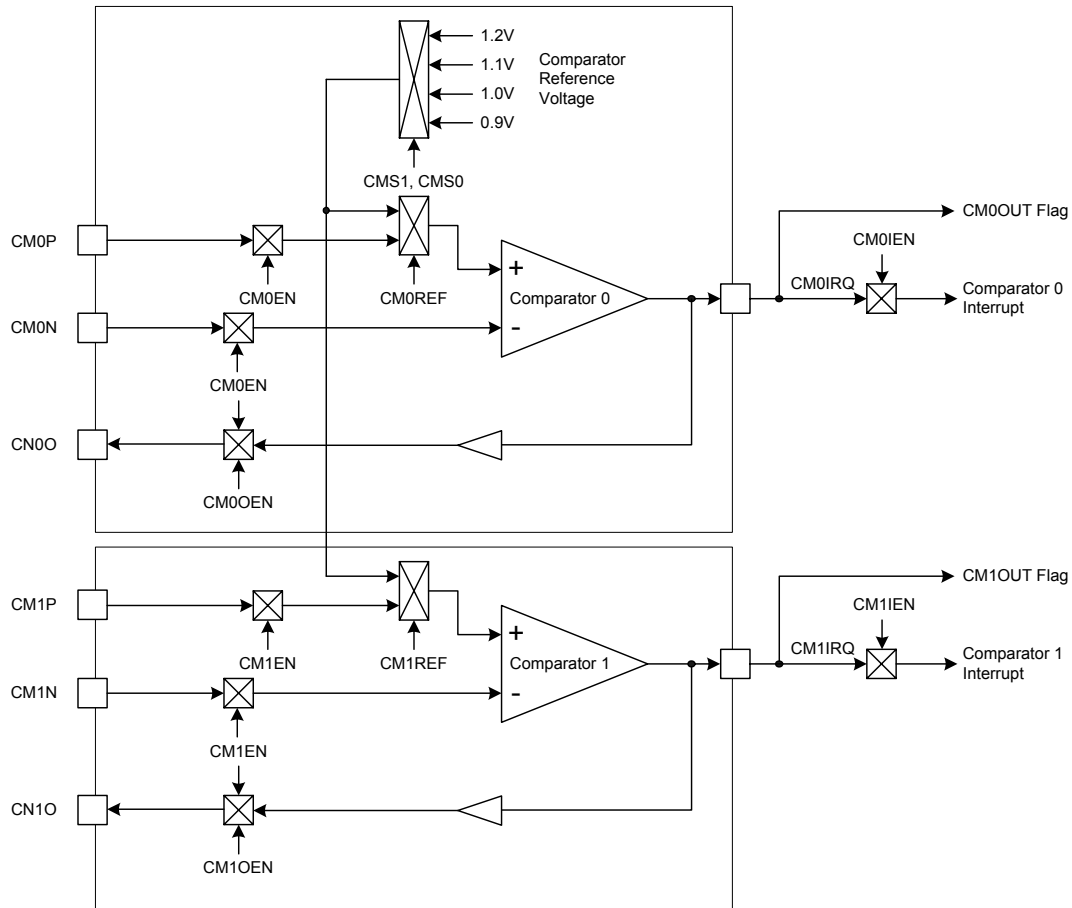
INT\_SER90:

```
...
... ; 恢复 ACC 和 PFLAG。
RETI
```

# 9 模拟比较器

## 9.1 概述

模拟比较模块包括 2 通道模拟比较器和内部参考电压，比较器主要用于 DC 电压和低电压的比较。模拟比较器的同相端、反向端及输出引脚都与普通 I/O 引脚共用，且由寄存器控制。2 通道的模拟比较器的结构图如下所示：



### 比较器 0 的引脚配置：

CM0P：比较器 0 的同相端与 P2.3 共用，当 CM0EN = 1 且 CM0REF = 0 时使能 CM0P。

CM0N：比较器 0 的反相端与 P2.2 共用，当 CM0EN = 1 时使能 CM0N。

CM0O：比较器 0 的输出引脚与 P2.4 共用，当 CM0EN = 1 且 CM0OEN = 1 时使能 CM0O。

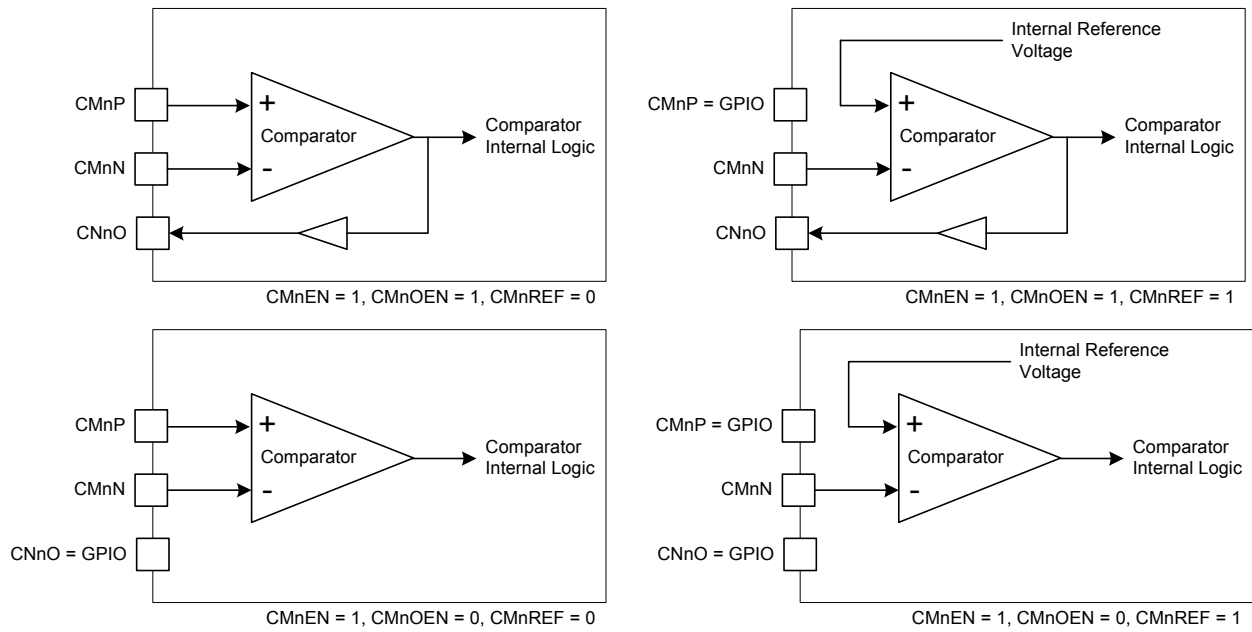
### 比较器 1 的引脚配置：

CM1P：比较器 1 的同相端与 P2.6 共用，当 CM1EN = 1 且 CM1REF = 0 时使能 CM1P。

CM1N：比较器 1 的反相端与 P2.5 共用，当 CM1EN = 1 时使能 CM1N。

CM1O：比较器 1 的输出引脚与 P2.7 共用，当 CM1EN = 1 且 CM1OEN = 1 时使能 CM1O。

除了上述条件外，比较器的引脚即为普通 I/O 模式。



- \* 注:
- 1、 比较器的输出信号是通过内部缓存器输出到输出引脚，而不是单纯的模拟比较输出。
  - 2、 比较器的正极和负极输入引脚都必须和 GND 之间连接一个 0.1uF 的电容，并尽可能的靠近单片机的引脚。

## 9.2 CP0M 寄存器

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>CMP0M</b>	CM0EN	CM0IEN	CM0IRQ	CM0OEN	CM0REF	CM0OUT	CMS1	CMS0
读/写	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
复位后	0	0	0	0	0	0	0	0

- Bit 7     **CM0EN**: 比较器 0 使能控制位。  
0 = 禁止, 所有的比较器引脚为普通 I/O 引脚;  
1 = 使能, CMON 引脚为模拟输入引脚。
- Bit 6     **CM0IEN**: 比较器 0 中断允许控制位。  
0 = 禁止;  
1 = 使能。
- Bit 5     **CM0IRQ**: 比较器 0 中断请求标志位。  
0 = CM0P 电压或比较器 0 的参考电压低于 CMON 的电压;  
1 = CM0P 电压或比较器 0 的参考电压高于 CMON 的电压。
- Bit 4     **CM0OEN**: 比较器 0 输出引脚控制位。  
0 = 禁止, CM0O 引脚为普通 I/O 引脚;  
1 = 使能, CMON 引脚为比较器的输出引脚。
- Bit 3     **CM0REF**: 比较器 0 内部参考电压控制位。  
0 = 禁止, CM0P 引脚为模拟输入引脚;  
1 = 使能, CM0P 引脚为普通 I/O 引脚。
- Bit 2     **CM0OUT**: 比较器 0 RAW 输出标志位。  
0 = CM0P 电压或比较器 0 的参考电压低于 CMON 电压;  
1 = CM0P 电压或比较器 0 的参考电压高于 CMON 电压。
- Bit[1:0]   **CMS[1:0]**: 比较器内部参考电压选择位。  
00 = 0.9V, 01 = 1.0V, 10 = 1.1V, 11 = 1.2V。



### 9.3 CP1M 寄存器

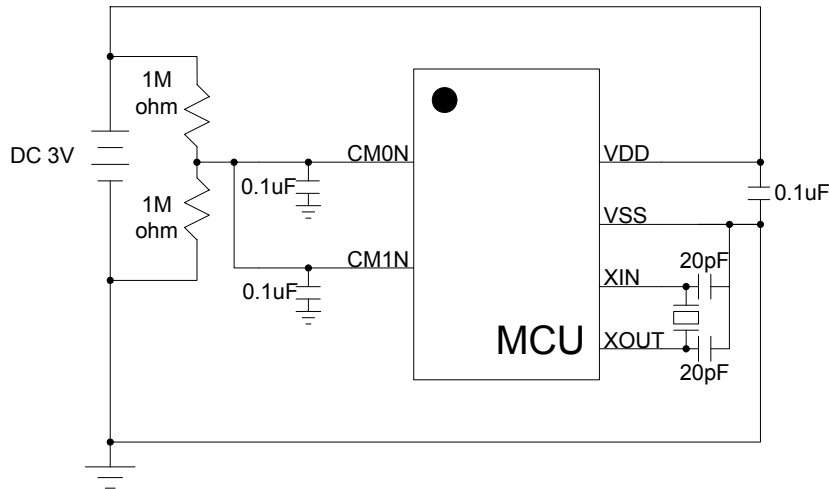
09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>CMP1M</b>	CM1EN	CM1IEN	CM1IRQ	CM1OEN	CM1REF	CM1OUT	-	-
读/写	R/W	R/W	R/W	R/W	R/W	R	-	-
复位后	0	0	0	0	0	0	-	-

- Bit 7 **CM1EN**: 比较器 1 使能控制位。  
0 = 禁止, 所有的比较器引脚为普通 I/O 引脚;  
1 = 使能, CM1N 引脚为模拟输入引脚。
- Bit 6 **CM1IEN**: 比较器 1 中断允许控制位。  
0 = 禁止;  
1 = 使能。
- Bit 5 **CM1IRQ**: 比较器 1 中断请求标志位。  
0 = CM1P 电压或比较器 1 的参考电压低于 CM1N 电压;  
1 = CM1P 电压或比较器 1 的参考电压高于 CM1N 电压。
- Bit 4 **CM1OEN**: 比较器 1 输出引脚控制位。  
0 = 禁止, CM1O 引脚为普通 I/O 引脚;  
1 = 使能, CM1O 引脚为比较器的输出引脚。
- Bit 3 **CM1REF**: 比较器 1 内部参考电压控制位。  
0 = 禁止, CM1P 引脚为模拟输入引脚;  
1 = 使能, CM1P 引脚为普通 I/O 引脚。
- Bit 2 **CM1OUT**: 比较器 1 RAW 输出标志位。  
0 = CM1P 电压或比较器 1 的参考电压低于 CM1N 电压;  
1 = CM1P 电压或比较器 1 的参考电压高于 CM1N 电压。

\* 注: CMnOUT 为比较器的 RAW 输出引脚, 没有锁存结构, 根据比较器的处理结果而变化。但 CMnIRQ 为比较器的锁存结果, 必须由程序来清零。

## 9.4 模拟比较器的应用

利用模拟比较器可以实现 2 个电池低电压的检测，这 2 个低电压分别是 2.2V 和 2.0V。当电池的电压低于 2.2V 时，系统开始低电压处理；当电池的电压低于 2.0V 时，系统就进行无电处理。电池的检测电压为电池的原电池电压的 1/2。比较器的同相端电压（参考电压）为比较器的内部参考电压。应用电路如下所示：



该应用电路采用内部参考源，比较器通过内部标志进行输出处理，故该电路仅使用 CMnN 引脚输入 1/2 电池电压和内部参考电压相比较，使用比较器 0 去检测 2.2V 的电池电压，比较器 1 检测 2.0V 的电池电压。

➤ 例：利用 2 通道的比较器检测电池的状态。电池电压低于 2.2V 时为电池低电压状态；电池电压低于 2.0V 时则判定电池无电。下面的示例程序通过轮流检测 CMnOUT 标志位以检测电池的电压状态以便进行不同的处理。用户也可以使用比较器的中断功能进行实时处理。

；比较器初始化。

```
MOV      A, #00001010b      ; 使能内部参考源-1.1V。
BOMOV    CMP0M, A          ; 禁止比较器的输出引脚。
BOMOV    CMP1M, A          ; 禁止比较器的中断功能。
```

```
BOBSET   FCM0EN           ; 使能比较器 0。
BOBSET   FCM1EN           ; 使能比较器 1。
```

；检测电池的 2.2V 电压。

CMP0\_CHK:

```
BOBSET   FCMS1            ; 设置内部参考电压为 1.1V。
BOBCLR   FCMS0
NOP      ; 延时 2 个指令周期作为内部 band-gap 设置时间。
NOP
```

```
BOBTS1   FCM0OUT          ; 通过 CM0OUT 标志检测比较器 0 的状态。
JMP      Main             ; 电池没电，跳回主程序。
JMP      CMP1_CHK         ; 电池处于低电压状态，检测比较器 1。
```

；检测电池的 2.0V 电压。

CMP1\_CHK:

```
BOBCLR   FCMS1            ; 设置内部参考电压为 1.0V。
BOBSET   FCMS0
NOP      ; 延时 2 条指令周期作为内部 band-gap 设置时间。
NOP
```

```
BOBTS1   FCM1OUT          ; 通过 CM1OUT 标志检测比较器 1 的状态。
JMP      LowBat           ; 电池处于低电压状态，进行低电压处理。
JMP      NoBat            ; 电池没电，进行无电处理。
```

；低电压处理。

LowBat:

```
...
JMP      Main             ; 返回主程序。
```

；无电处理。

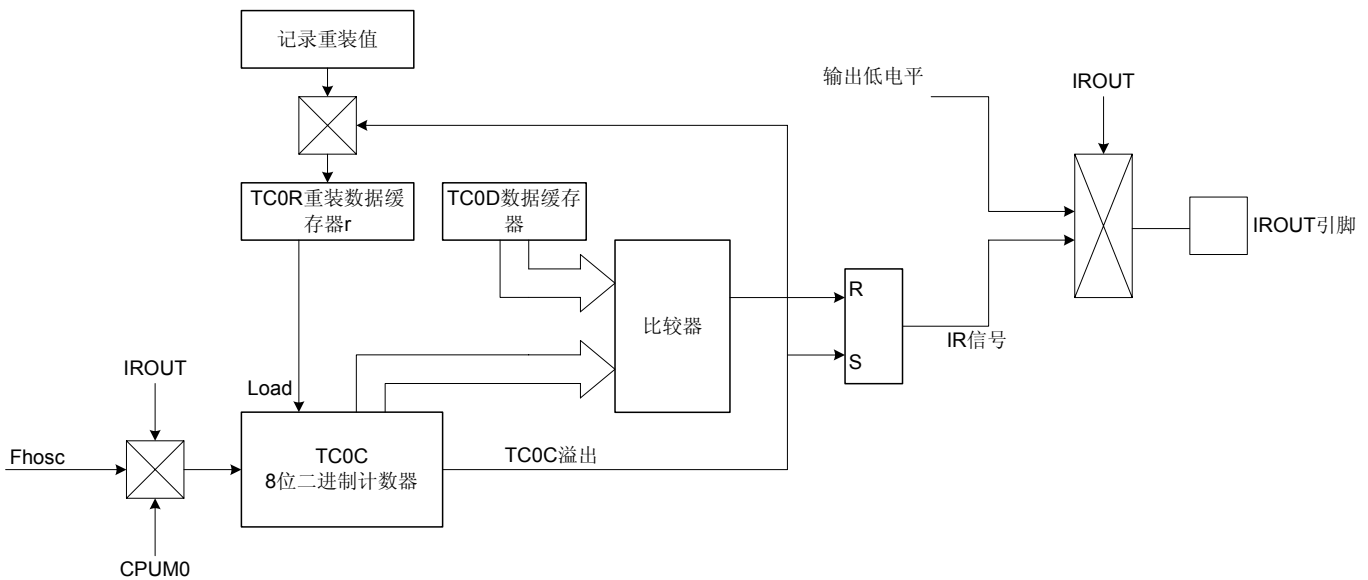
NoBat:

```
...
JMP      Main             ; 返回主程序。
```

# 10 红外信号输出

## 10.1 概述

二进制计数定时器 TC0 产生红外信号，IROUT 是红外信号的输出引脚。当 TCOM 的 IREN 位为“1”时，IROUT 引脚由普通 IO 转变为红外输出模式。若 CREN = 0 或者系统处于睡眠模式时，IROUT 输出低电平。TC0R 和 TC0D 控制红外信号的频率以及占空比，TC0R 控制红外信号的频率，TC0D 控制红外信号的占空比。外部高速时钟源 (Fhosc) 作为 TC0 的单一的时钟源，如 4MHz 晶振。如果外部振荡器是 4MHz，则 TC0 的时钟速率就是 4MHz。TC0 只能产生红外信号的输出信号，不支持中断功能。当使能红外信号输出功能时，红外信号输出高电平。TC0C 的初始值是 TC0R 并开始计数。当 TC0C = TC0D，红外信号输出低电平并停止输出高电平，当 TC0C 溢出 (0FFH 到 00H) 时，红外信号停止输出低电平。系统会自动将 TC0R 中的值装载到 TC0C 寄存器中并开始进入下一个循环。



## 10.2 红外信号控制寄存器

### 10.2.1 TC0M 模式寄存器

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC0M</b>	-	-	-	-	-	-	IREN	CREN
读/写	-	-	-	-	-	-	R/W	R/W
复位后	-	-	-	-	-	-	0	0

Bit 1 **IREN**: IROUT 引脚输出控制位。  
0 = 禁止, IROUT 和 P5.4 共享引脚为普通 I/O 模式。  
1 = 使能, IROUT 输出引脚, 且输出低电平。

Bit 0 **CREN**: IR 载波输出控制位。  
0 = 禁止, IROUT 输出低电平。  
1 = 使能, IROUT 输出 IR 载波信号。

\* 注: IR 载波输出条件为 IREN=1 且 CREN=1, 如果 CREN=1 而 IREN=0, IROUT 和 P5.4 引脚为普通 I/O 模式。

### 10.2.2 TC0C 计数寄存器

8 位二进制计数寄存器 TC0C 控制 TC0 的间隔时间。

0DBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC0C</b>	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

\* 注: 在允许红外信号输出功能之前设置 TC0C = TC0D, 以保证第一次循环是正确的。

### 10.2.3 TC0R 自动装载寄存器

寄存器 TC0R 控制红外信号的频率。TC0 具有自动装载功能, 当 TC0C 溢出, TC0R 的值会自动装载到 TC0C, 为 IR 信号的周期提供了精确的计时标准。用户不需要在中断中复位 TC0C。

TC0 为双重缓存器设计, 若 TC0R 的值由程序设置, 其新值就保存在第一个缓存器中, 直到 TC0 溢出, 其新值就移动到真正的 TC0R 缓存器中。

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

TC0R 初始值的计算如下:

$$\text{TC0R 初始值} = 256 - (\text{TC0 间隔时间} * \text{输入时钟})$$

\* 注: 输入时钟来自外部 4MHz 晶振。

➤ 例: 设置红外信号循环的频率是 38KHz, 输入时钟 = 4MHz。

$$\text{TC0R 初始值} = 256 - (\text{TC0 间隔时间} * \text{输入时钟})$$

TC0 间隔时间 = 1/38KHz = 26.3us

输入时钟 = 外部晶振 4MHz

$$\begin{aligned} \text{TC0R} &= 256 - (26.3\text{us} * 4\text{MHz}) \\ &= 150.8 = 151 = 97\text{H} \end{aligned}$$

## 10.2.4 TC0D 红外信号占空比控制寄存器

寄存器 TC0D 可以控制红外信号的占空比和高电平脉冲的宽度。当 TC0C = TC0D 时，红外信号由高电平脉冲变成低电平脉冲。当 TC0C 溢出，低电平脉冲停止，高电平脉冲的宽度是 TC0D ~ TC0R，低电平脉冲的宽度是 256 ~ TC0D。这样通过 TC0R 和 TC0D 寄存器就可以调整红外信号的频率和占空比（duty / cycle）。

0E8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
<b>TC0D</b>	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

TC0D 初始值的计算如下：

$$\text{TC0D 初始值} = \text{TC0R} + (256 - \text{TC0R}) / (1/\text{红外信号占空比})$$

➤ 例：红外信号的频率是 38KHz，占空比是 1/3，输入时钟 = 4MHz。

$$\text{TC0D 初始值} = \text{TC0R} + (256 - \text{TC0R}) / (1/\text{IR duty})$$

**TC0R（红外信号频率是 38KHz） = 151**

$$\begin{aligned} \text{TC0D} &= 151 + (256 - 151) / (1 / (1/3)) \\ &= 186 \\ &= \mathbf{BAh} \end{aligned}$$

红外信号的基本列表。系统时钟 = 4MHz.

IR Freq. (KHz)	TC0C TC0R		TC0D						Freq. Error Rate
			1/2 duty		1/3 duty		1/4duty		
	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	
32	131	83	193.50	C1	172.67	AC	162.25	A2	0.00%
36	145	91	200.50	C8	182.00	B6	172.75	AC	0.10%
38	151	97	203.50	CB	186.00	BA	177.25	B1	0.25%
39.2	154	9A	205.00	CD	188.00	BC	179.50	B3	0.04%
40	156	9C	206.00	CE	189.33	BD	181.00	B5	0.00%
56	185	B9	220.50	DC	208.67	D0	202.75	CA	0.60%

## 10.2.5 红外信号输出操作流程

☞ 设置 TC0C 和 TC0R，得到红外信号的周期。

```
MOV      A, #IRCYCVAL      ;
MOV      TC0C, A
MOV      TC0R, A
```

☞ 设置 TC0D，得到红外信号的占空比。

```
MOV      A, #IRDUTYVAL     ;
MOV      TC0D, A
```

☞ 使能红外信号输出。

```
BSET     FIREN              ;设置 IROUT 和 P5.4 共享引脚为 IR 功能输出引脚。
BSET     FCREN              ;设置 IR 输出载波信号。
```

# 11 指令集

指令	指令格式	描述	C	DC	Z	周期	
MOV	A,M	$A \leftarrow M$ 。	-	-	√	1	
	M,A	$M \leftarrow A$ 。	-	-	-	1	
	B0MOV	A,M	$A \leftarrow M$ (bank 0)。	-	-	√	1
	B0MOV	M,A	$M$ (bank 0) $\leftarrow A$ 。	-	-	-	1
	MOV	A,I	$A \leftarrow I$	-	-	-	1
	B0MOV	M,I	$M \leftarrow I$ 。(M 仅适用于系统寄存器 R、Y、Z、RBANK、PFLAG。)	-	-	-	1
	XCH	A,M	$A \leftrightarrow M$ 。	-	-	-	1+N
	B0XCH	A,M	$A \leftrightarrow M$ (bank 0)。	-	-	-	1+N
MOVC		R, $A \leftarrow ROM[Y,Z]$ 。	-	-	-	2	
ADC	A,M	$A \leftarrow A + M + C$ ，如果产生进位则 C=1，否则 C=0。	√	√	√	1	
	M,A	$M \leftarrow A + M + C$ ，如果产生进位则 C=1，否则 C=0。	√	√	√	1+N	
ADD	A,M	$A \leftarrow A + M$ ，如果产生进位则 C=1，否则 C=0。	√	√	√	1	
	M,A	$M \leftarrow A + M$ ，如果产生进位则 C=1，否则 C=0。	√	√	√	1+N	
B0ADD	M,A	$M$ (bank 0) $\leftarrow M$ (bank 0) + A，如果产生进位则 C=1，否则 C=0。	√	√	√	1+N	
ADD	A,I	$A \leftarrow A + I$ ，如果产生进位则 C=1，否则 C=0。	√	√	√	1	
SBC	A,M	$A \leftarrow A - M - /C$ ，如果产生借位则 C=0，否则 C=1。	√	√	√	1	
	M,A	$M \leftarrow A - M - /C$ ，如果产生借位则 C=0，否则 C=1。	√	√	√	1+N	
SUB	A,M	$A \leftarrow A - M$ ，如果产生借位则 C=0，否则 C=1。	√	√	√	1	
	M,A	$M \leftarrow A - M$ ，如果产生借位则 C=0，否则 C=1。	√	√	√	1+N	
SUB	A,I	$A \leftarrow A - I$ ，如果产生借位则 C=0，否则 C=1。	√	√	√	1	
AND	A,M	$A \leftarrow A$ 与 M。	-	-	√	1	
	M,A	$M \leftarrow A$ 与 M。	-	-	√	1+N	
	A,I	$A \leftarrow A$ 与 I。	-	-	√	1	
	OR	A,M	$A \leftarrow A$ 或 M。	-	-	√	1
		M,A	$M \leftarrow A$ 或 M。	-	-	√	1+N
	OR	A,I	$A \leftarrow A$ 或 I。	-	-	√	1
	XOR	A,M	$A \leftarrow A$ 异或 M。	-	-	√	1
		M,A	$M \leftarrow A$ 异或 M。	-	-	√	1+N
XOR	A,I	$A \leftarrow A$ 异或 I。	-	-	√	1	
SWAP	M	$A(b3\sim b0, b7\sim b4) \leftarrow M(b7\sim b4, b3\sim b0)$ 。	-	-	-	1	
	SWAPM	M	$M(b3\sim b0, b7\sim b4) \leftarrow M(b7\sim b4, b3\sim b0)$ 。	-	-	-	1+N
	RRC	M	$A \leftarrow M$ 带进位右移。	√	-	-	1
	RRCM	M	$M \leftarrow M$ 带进位右移。	√	-	-	1+N
	RLC	M	$A \leftarrow M$ 带进位左移。	√	-	-	1
	RLCM	M	$M \leftarrow M$ 带进位左移。	√	-	-	1+N
	CLR	M	$M \leftarrow 0$ 。	-	-	-	1
	BCLR	M.b	$M.b \leftarrow 0$ 。	-	-	-	1+N
	BSET	M.b	$M.b \leftarrow 1$	-	-	-	1+N
	B0BCLR	M.b	$M(\text{bank } 0).b \leftarrow 0$ 。	-	-	-	1+N
B0BSET	M.b	$M(\text{bank } 0).b \leftarrow 1$ 。	-	-	-	1+N	
CMPRS	A,I	比较，如果相等则跳过下一条指令 C 与 ZF 标志位可能受影响。	√	-	√	1+S	
	A,M	比较，如果相等则跳过下一条指令 C 与 ZF 标志位可能受影响。	√	-	√	1+S	
INCS	M	$A \leftarrow M + 1$ ，如果 A = 0，则跳过下一条指令。	-	-	-	1+S	
INCMS	M	$M \leftarrow M + 1$ ，如果 M = 0，则跳过下一条指令。	-	-	-	1+N+S	
DECS	M	$A \leftarrow M - 1$ ，如果 A = 0，则跳过下一条指令。	-	-	-	1+S	
DECMS	M	$M \leftarrow M - 1$ ，如果 M = 0，则跳过下一条指令。	-	-	-	1+N+S	
BTS	M.b	如果 M.b = 0，则跳过下一条指令。	-	-	-	1+S	
	M.b	如果 M.b = 1，则跳过下一条指令。	-	-	-	1+S	
	M.b	如果 M(bank 0).b = 0，则跳过下一条指令。	-	-	-	1+S	
	M.b	如果 M(bank 0).b = 1，则跳过下一条指令。	-	-	-	1+S	
JMP	d	跳转指令， $PC_{15/14} \leftarrow RomPages1/0$ ， $PC_{13\sim PC0} \leftarrow d$ 。	-	-	-	2	
CALL	d	子程序调用指令， $Stack \leftarrow PC_{15\sim PC0}$ ， $PC_{15/14} \leftarrow RomPages1/0$ ， $PC_{13\sim PC0} \leftarrow d$ 。	-	-	-	2	
RET		子程序跳出指令， $PC \leftarrow Stack$ 。	-	-	-	2	
RETI		中断处理程序跳出指令， $PC \leftarrow Stack$ ，使能全局中断控制位。	-	-	-	2	
PUSH		进栈指令，保存 ACC 和工作寄存器。	-	-	-	1	
POP		出栈指令，恢复 ACC 和工作寄存器。	√	√	√	1	
NOP		空指令，无特别意义。	-	-	-	1	

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

# 12 电气特性

## 12.1 极限参数

Supply voltage (Vdd).....	- 0.3V ~ 6.0V
Input in voltage (Vin).....	Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)	
SN8P26L34K, SN8P2L634S, SN8P26L34X .....	0°C ~ + 70°C
SN8P26L32P, SN8P26L32S, .....	0°C ~ + 70°C
SN8P26L321P, SN8P2L6321S, SN8P26L321X.....	0°C ~ + 70°C
Storage ambient temperature (Tstor) .....	-40°C ~ + 125°C

## 12.2 电气特性

(All of voltages refer to Vss, Vdd = 3.0V, fosc = 4MHz, Fcpu=1MHz, ambient temperature is 25°C unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT	
Operating voltage	Vdd	Normal mode, Vpp = Vdd, 25°C, Fcpu = 2mips.	1.8	3.0	3.6	V	
RAM Data Retention voltage	Vdr		1.5		-	V	
Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset	0.05	-	-	V/ms	
Input Low Voltage	ViL1	All input ports	Vss	-	0.3Vdd	V	
	ViL2	Reset pin	Vss	-	0.2Vdd	V	
Input High Voltage	ViH1	All input ports	0.7Vdd	-	Vdd	V	
	ViH2	Reset pin	0.9Vdd	-	Vdd	V	
Reset pin leakage current	Ilekg	Vin = Vdd, 25°C	-	-	2	uA	
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 output source current sink current	IoH	Vop = Vdd – 0.5V	8	10	-	mA	
	IoL	Vop = Vss + 0.5V	8	12	-		
INTn trigger pulse width	Tint0	INT0 interrupt request pulse width	2/fcpu	-	-	cycle	
Supply Current	Idd1	Run Mode (No loading, Fcpu = Fosc/4)	Vdd= 3V, 4Mhz	-	1	2	mA
	Idd2	Slow Mode (Internal low RC, Stop high clock)	Vdd=3V, ILRC 10Khz	-	5	10	uA
	Idd3	Sleep Mode	Vdd= 3V, 25°C	-	1	2	uA
	Idd4	Green Mode (No loading, Fcpu = Fosc/4, Watchdog Disable)	Vdd= 3V, 4Mhz	-	0.25	0.5	mA
Vdd=3V, ILRC 10Khz			-	3	6	uA	
Internal High Oscillator Freq.	Fihrc	Internal High RC (IHRC) 25°C, Vdd= 3V, Fcpu = 1MHz	7.84	8	8.16	Mhz	
Band-gap Output Voltage	Vbd1	CMS0, CMS1 = 00	-	0.9	-	V	
	Vbd2	CMS0, CMS1 = 01	-	1.0	-		
	Vbd3	CMS0, CMS1 = 10	-	1.1	-		
	Vbd4	CMS0, CMS1 = 11	-	1.2	-		
Comparator Current	Icm1	Fcpu = 1MHz, Vdd=3V. Disable internal reference.	-	50	-	uA	
	Icm2	Fcpu = 1MHz, Vdd=3V. Enable internal reference.	-	150	-		
Comparator Input Offset Voltage	Vcmoff	Fcpu = 1MHz, Vdd=3V,	-	±5	-	mV	
LVD Voltage	Vdet0	Low voltage reset level.	-	1.7	-	V	
	Vdet1	Low voltage reset level. Fcpu = 1 MHz.	-	2.4	-	V	
		Low voltage indicator level. Fcpu = 1 MHz.	-	2.4	-		
Vdet2	Low voltage indicator level. Fcpu = 1 MHz	-	2.8	-	V		

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



# 13 开发工具

SONiX 为 SN8P26L34/32/321 的开发提供了在线仿真器 (ICE)、集成开发环境 (IDE) 和 EV-Kit。ICE 和 EV-Kit 是外部硬件装置，IDE 有一个友好的界面，为用户调试提供方便。各开发工具的版本如下：

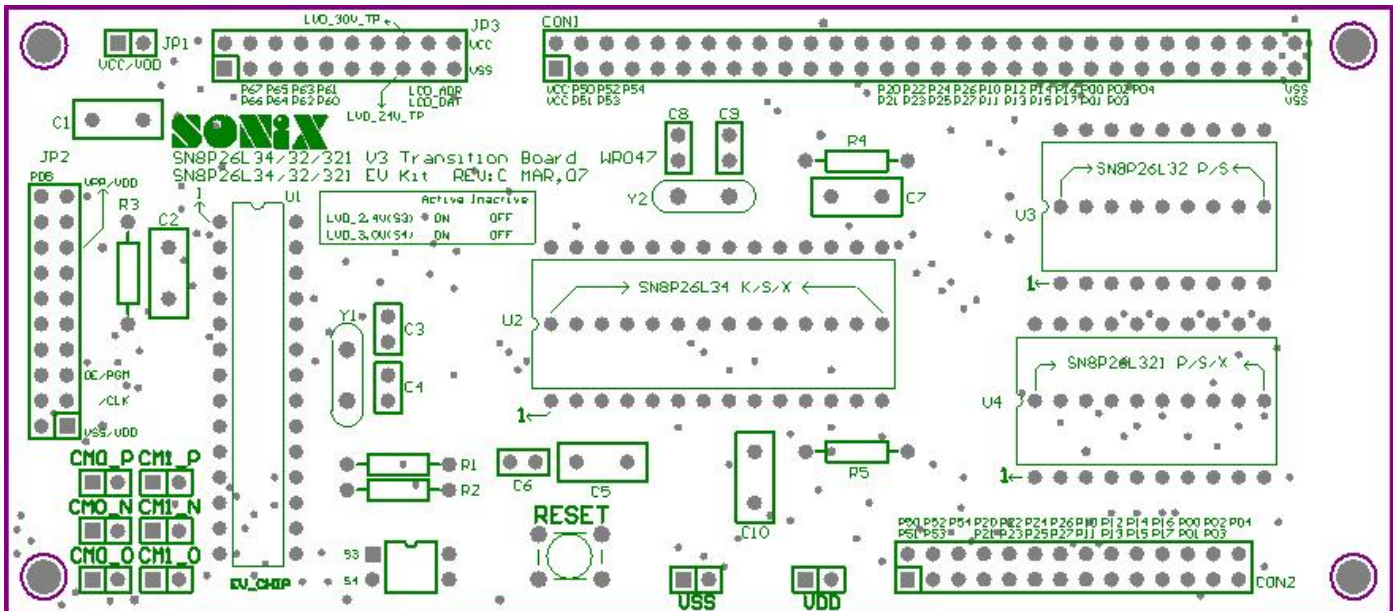
- ICE: SN8ICE2K;
- EV-kit: SN8P26L34/32 EV-kit Rev. A. (只支持 SN8P26L34) ;  
SN8P26L34/32/321 EV-kit Rev. B. (只支持 SN8P26L34/321) ;  
SN8P26L34/32/321 EV-kit Rev. C. (支持 SN8P26L34/32/321) ;
- IDE: SONiX IDE M2IDE\_V115 或更晚的版本;
- Writer: MPIII WRITE-LV.

## 13.1 SN8P26L34/32/321 EV-kit

SN8P26L34/32/321 EV-kit 包括 ICE 界面、GPIO 界面和 EV-Chip 模块。

- EV-chip 模块：仿真比较器功能。

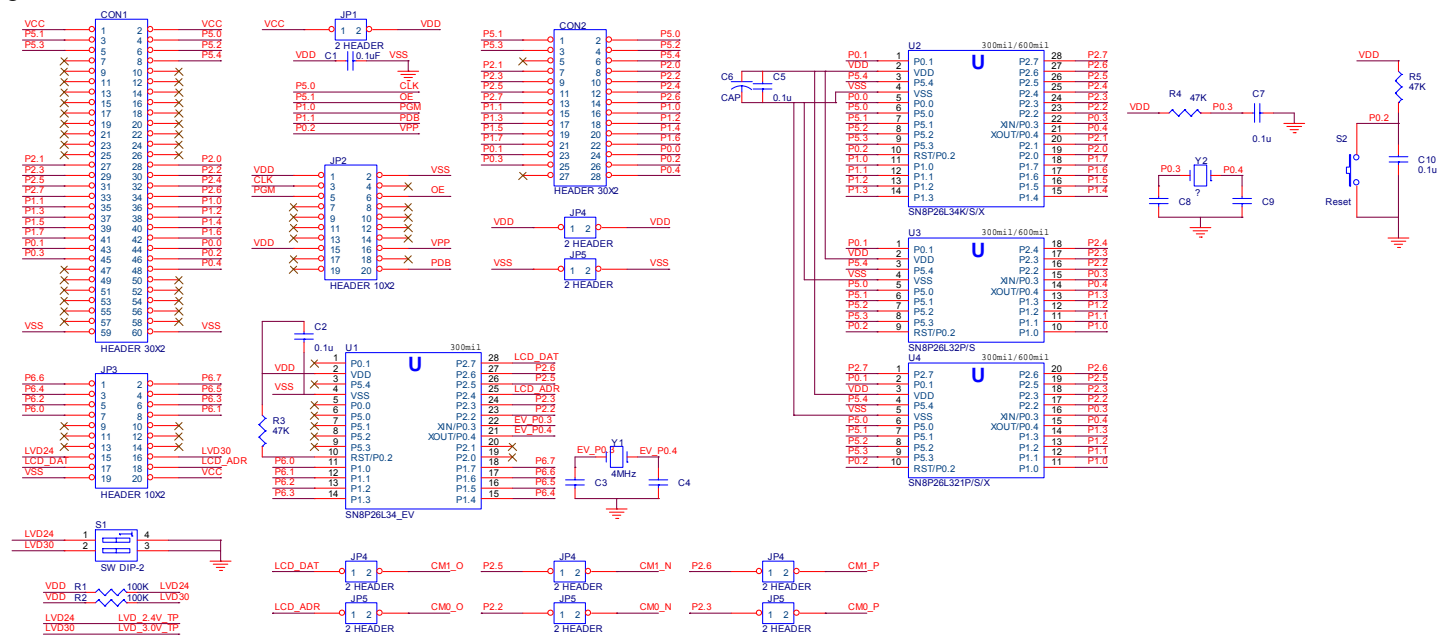
SN8P26L34/32/321 EV-kit 的 PCB 图如下所示：



- CON1、JP3: 连接到 SN8ICE2K。
- CON2: GPIO 接口。
- JP2: 连接到 MPIII writer。
- U1: SN8P26L34 EV-chip, 比较器的仿真模块。
- U2: 放置 SN8P26L34 DIP 封装并连接到用户的目标板。
- U3: 放置 SN8P26L32 DIP 封装并连接到用户的目标板。
- U4: 放置 SN8P26L321 DIP 封装并连接到用户的目标板。
- CM0\_P: 比较器 0 正极输入引脚。
- CM0\_N: 比较器 0 负极输入引脚。
- CM0\_O: 比较器 0 输出引脚。
- CM1\_P: 比较器 1 正极输入引脚。
- CM1\_N: 比较器 1 负极输入引脚。
- CM1\_O: 比较器 1 输出引脚。
- S1: LVD24 和 LVD28 转换开关。



● SN8P26L34/32/321 EV-kit 电路图:



**SONIX**

SN8P26L34/32/321 V3 Transition Board WRxxxx  
SN8P26L34/32/321 EV Kit REV:C

## 13.2 ICE 和 EV-KIT 的应用注意事项

SN8P26L34/32/321 EV-kit 包括比较器的仿真模块。使用 SN8P26L34P 的仿真代码来仿真比较器功能，SN8P26L34/32/321 的比较器的引脚与 P2 普通 I/O 口共用。用仿真器仿真时，不能把比较器的输入引脚连接 GPIO 引脚。

- 通过 SN8P26L34/32/321 EV-Kit 的 SN8P26L34 EV-Chip 来仿真比较器的功能。为了仿真比较器，通过上述引脚输入输出比较器的信号。
- 通过 SN8P26L34/32/321 EV-Kit 的 P2 口来仿真比较器的 P2（与 GPIO 共用）的功能。
- SN8P26L34/32/321 EV-Kit 的电源必须是外部 3V，而不能使用 ICE 的内部 5V 电源。在 SN8ICE2K 上断开内部 5V 电源的引脚，而采用外部 3V 电源。

# 14 OTP 烧录信息

## 14.1 烧录转接板信息

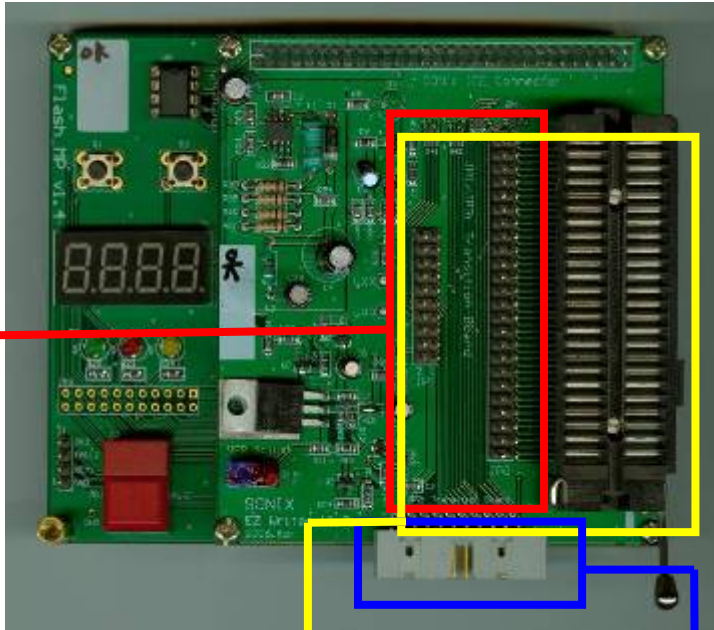


图 1 MP1111 Writer 的内部结构



Writer 上板 JP1/JP3



Writer 上板 JP1/JP3



Pin1 (down)

Pin20 (up)

Writer 上板 JP2

注 1: JP1 连接 MP 烧录转接板, JP3 连接 OTP MCU。

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

下面两个图演示了如何焊接烧录转接板。



图 2

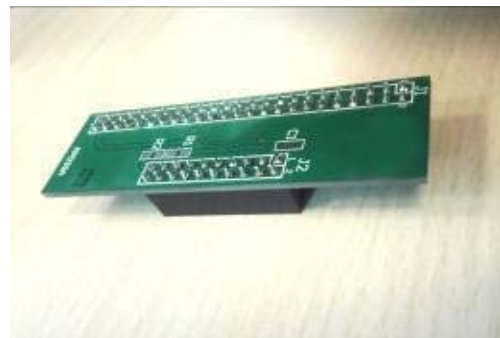


图 3

注: 印有 IC 型号的这一面为转接板的正面。180 度的母座必须焊接在 MP 转接板的背面。请参考图 2 和图 3。

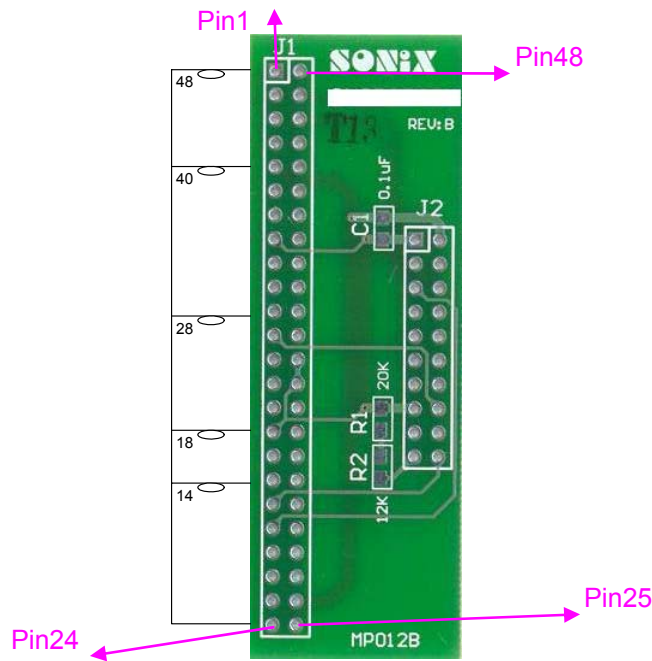


图 4 MP 转接板 (连接到 JP1&JP3)

**JP3 (连接 48-pin text tool)**

DIP 1	1	48	DIP48
DIP 2	2	47	DIP47
DIP 3	3	46	DIP46
DIP 4	4	45	DIP45
DIP 5	5	44	DIP44
DIP 6	6	43	DIP43
DIP 7	7	42	DIP42
DIP 8	8	41	DIP41
DIP 9	9	40	DIP40
DIP10	10	39	DIP39
DIP11	11	38	DIP38
DIP12	12	37	DIP37
DIP13	13	36	DIP36
DIP14	14	35	DIP35
DIP15	15	34	DIP34
DIP16	16	33	DIP33
DIP17	17	32	DIP32
DIP18	18	31	DIP31
DIP19	19	30	DIP30
DIP20	20	29	DIP29
DIP21	21	28	DIP28
DIP22	22	27	DIP27
DIP23	23	26	DIP26
DIP24	24	25	DIP25

**JP1/JP2**

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

JP1 连接 MP 烧录转接板

JP2 连接外部转接板

## 14.2 烧录引脚信息

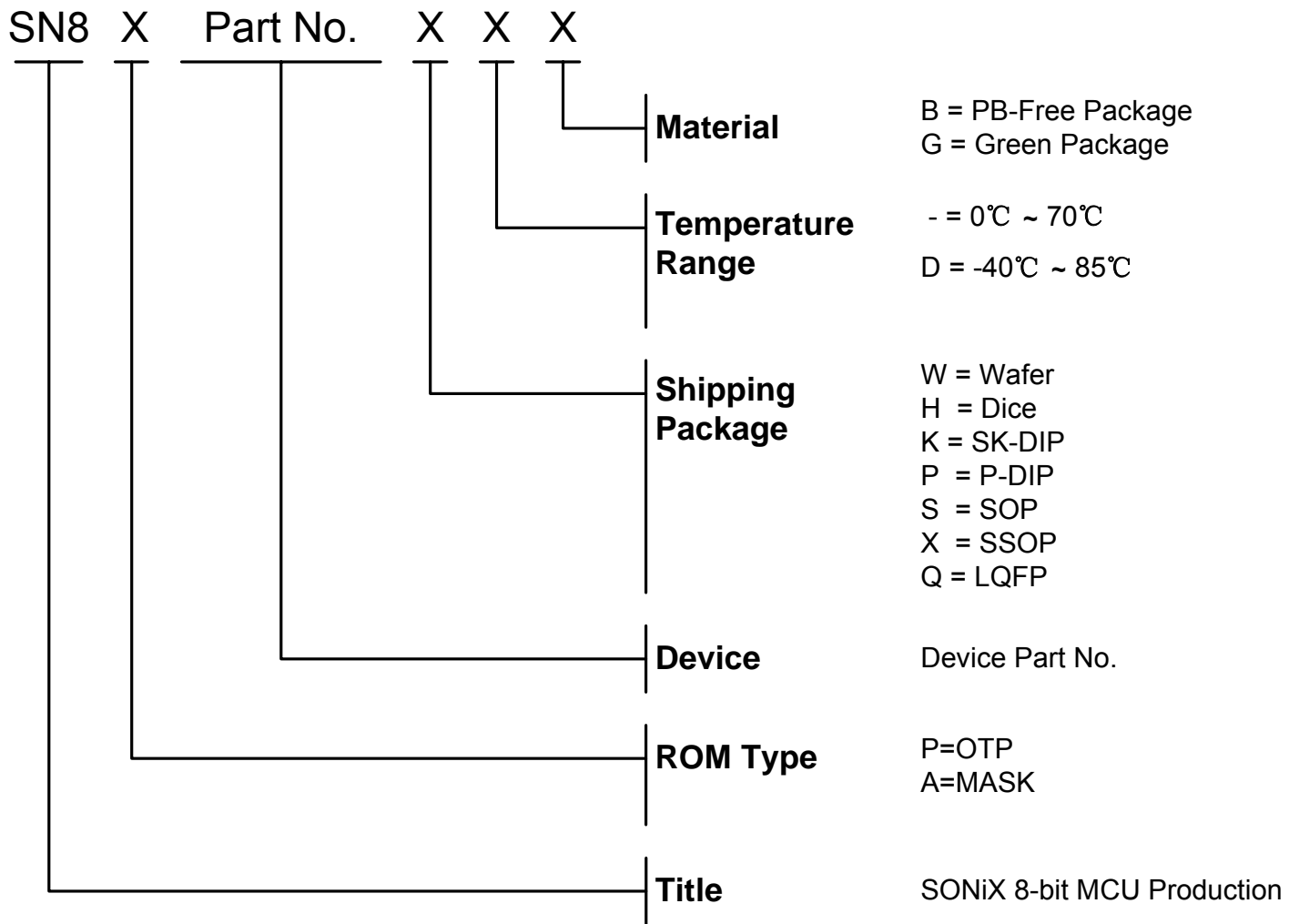
SN8P26L34/26L32/26L321 的烧录信息										
单片机名称		SN8P26L34KSX			SN8P26L32PS			SN8P26L321P/S/X		
MPSIII Writer		OTP IC / JP3 引脚配置								
JP1/JP2 Pin Number	JP1/JP2 Pin Name	IC Pin Number	IC Pin Name	JP3 Pin Number	IC Pin Number	IC Pin Name	JP3 Pin Number	IC Pin Number	IC Pin Name	JP3 Pin Number
1	VDD	2	VDD	12	2	VDD	17	3	VDD	17
2	GND	4	VSS	14	4	VSS	19	5	VSS	19
3	CLK	6	P5.0	16	5	P5.0	20	6	P5.0	20
4	CE	-	-	-	-	-	-	-	-	-
5	PGM	11	P1.0	21	10	P1.0	25	11	P1.0	25
6	OE	7	P5.1	17	6	P5.1	21	7	P5.1	21
7	D1	-	-	-	-	-	-	-	-	-
8	D0	-	-	-	-	-	-	-	-	-
9	D3	-	-	-	-	-	-	-	-	-
10	D2	-	-	-	-	-	-	-	-	-
11	D5	-	-	-	-	-	-	-	-	-
12	D4	-	-	-	-	-	-	-	-	-
13	D7	-	-	-	-	-	-	-	-	-
14	D6	-	-	-	-	-	-	-	-	-
15	VDD	2	VDD	12	2	VDD	17	3	VDD	17
16	VPP	10	RST	20	9	RST	24	10	RST	24
17	HLS	-	-	-	-	-	-	-	-	-
18	RST	-	-	-	-	-	-	-	-	-
19	-	-	-	-	-	-	-	-	-	-
20	ALSB/PDB	12, 8	P1.1, P5.2	22, 18	11, 7	P1.1, P5.2	26, 22	12, 8	P1.1, P5.2	26, 12

# 15 单片机正印命名规则

## 15.1 概述

SONiX 8 位单片产品具有多种型号，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。

## 15.2 单片机型号说明

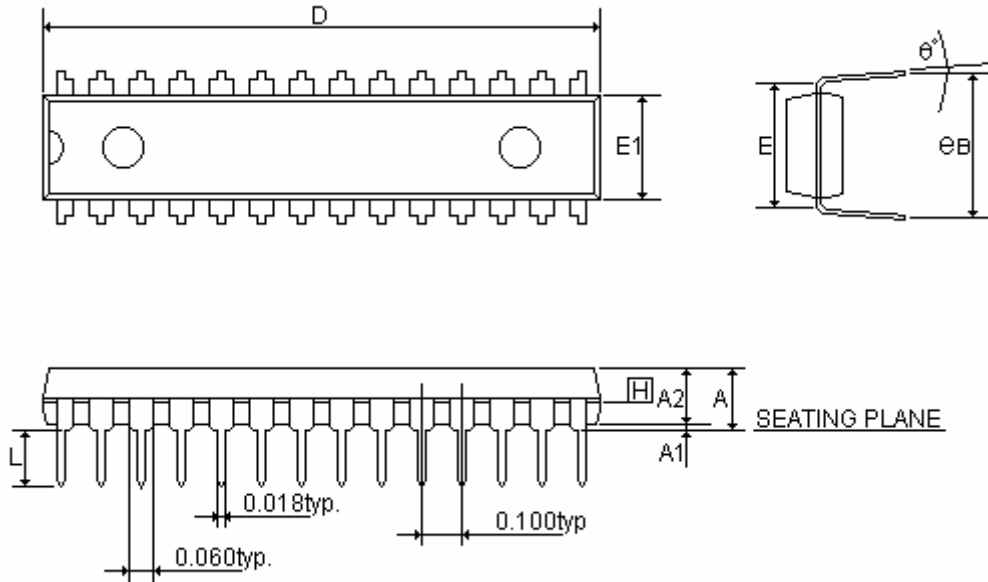


\* 注：SN8P26L34/26L32/26L321 不支持-40°C~85°C 的温度范围，也没有 LQFP 的封装和 MASK ROM 型号。



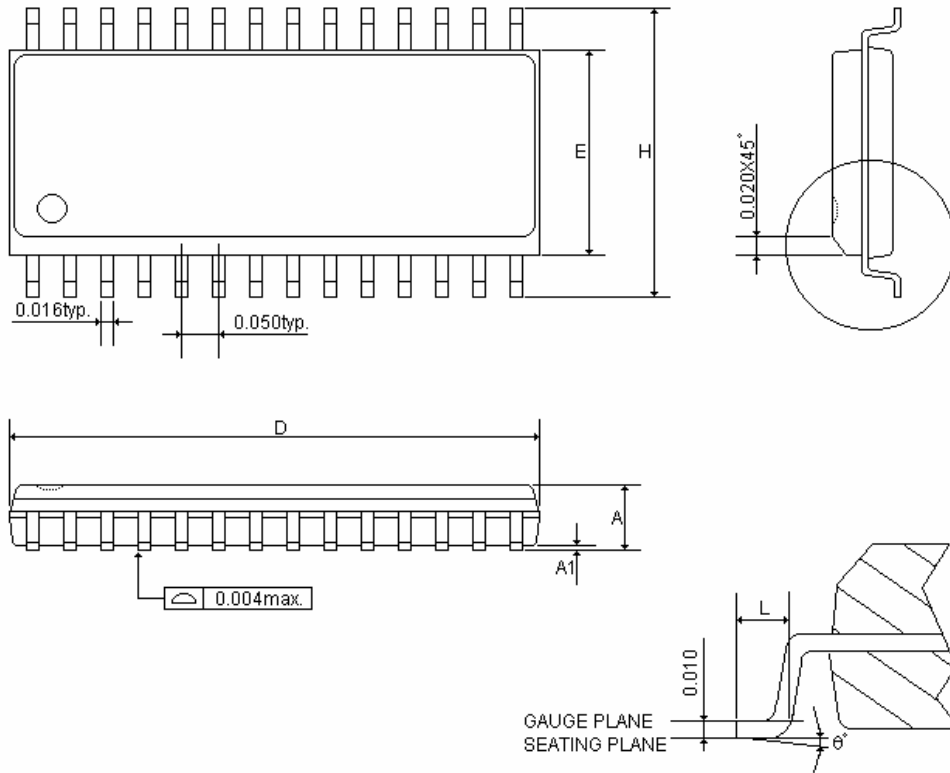
# 16 封装信息

## 16.1 SK-DIP 28 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.114	0.130	0.135	2.896	3.302	3.429
D	1.390	1.390	1.400	35.306	35.306	35.560
E	0.310			7.874		
E1	0.283	0.288	0.293	7.188	7.315	7.442
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.330	0.350	0.370	8.382	8.890	9.398
$\theta^\circ$	0°	7°	15°	0°	7°	15°

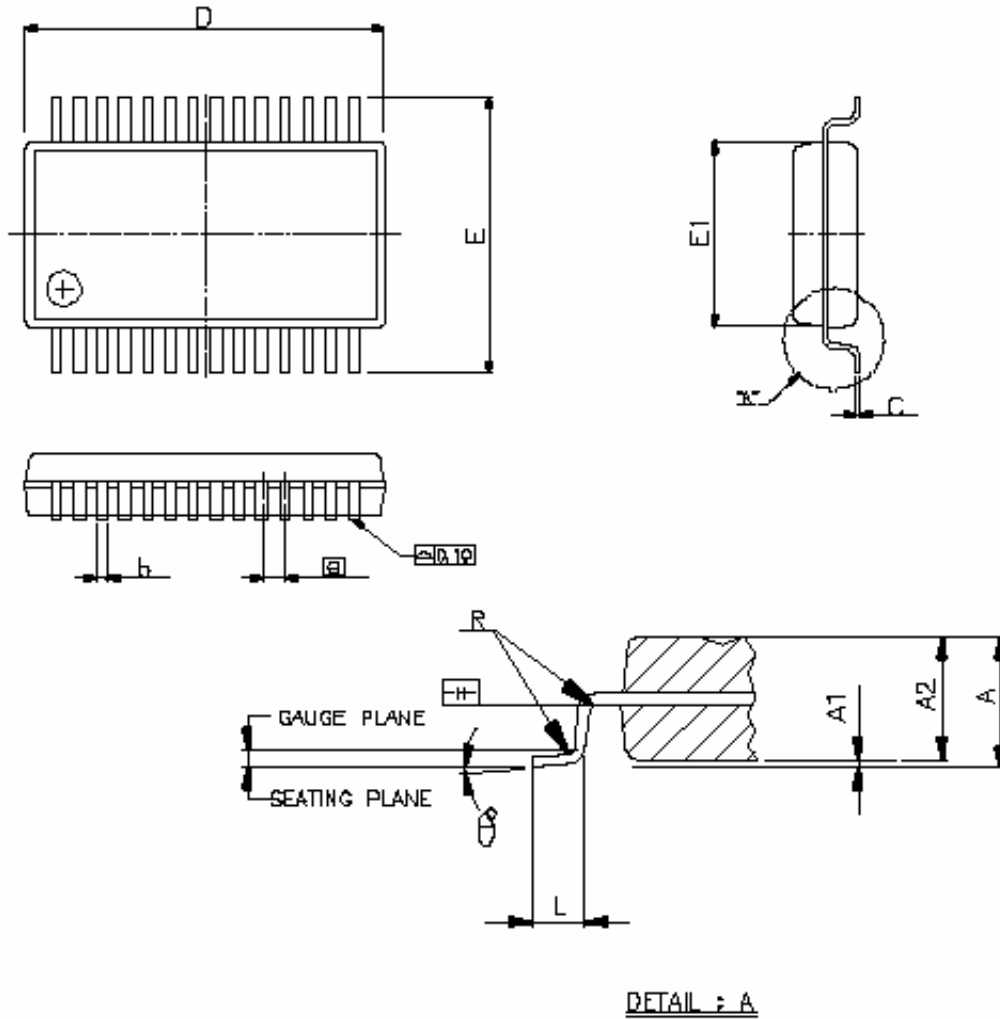
## 16.2 SOP 28 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.093	0.099	0.104	2.362	2.502	2.642
A1	0.004	0.008	0.012	0.102	0.203	0.305
D	0.697	0.705	0.713	17.704	17.907	18.110
E	0.291	0.295	0.299	7.391	7.493	7.595
H	0.394	0.407	0.419	10.008	10.325	10.643
L	0.016	0.033	0.050	0.406	0.838	1.270
$\theta^\circ$	0°	4°	8°	0°	4°	8°

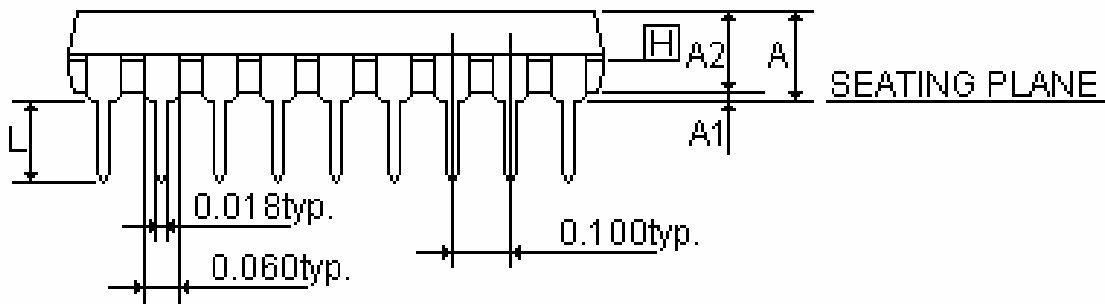
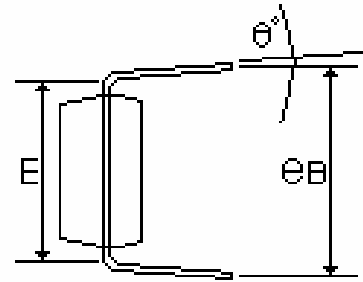
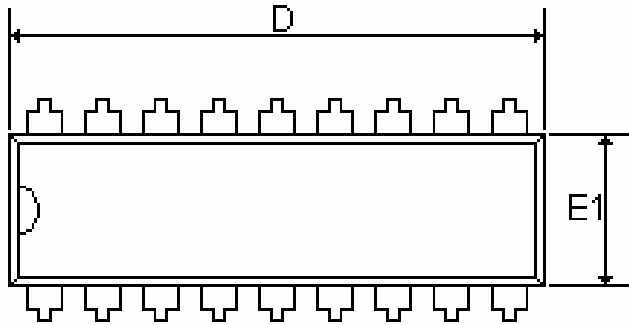


### 16.3 SSOP 28 PIN



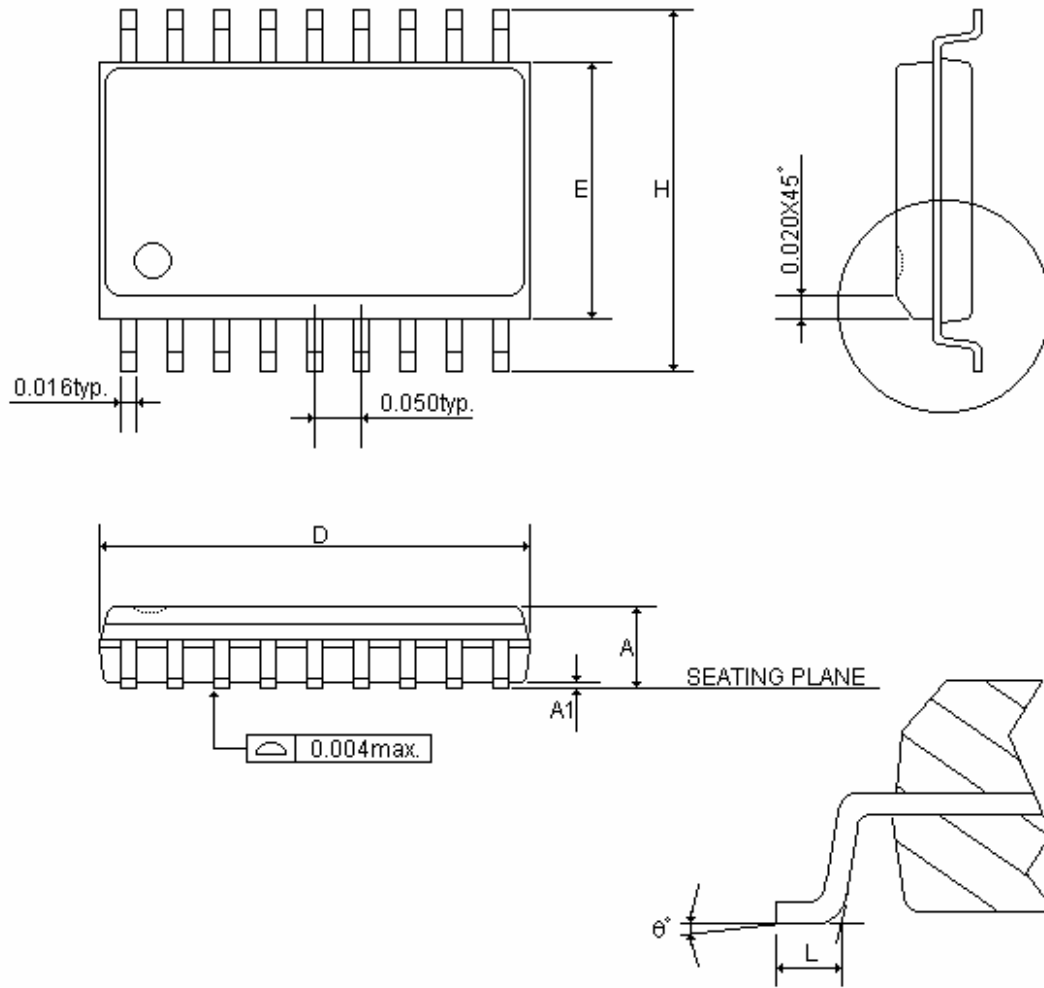
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.08	-	-	2.13
A1	0.00	-	0.01	0.05	-	0.25
A2	0.06	0.07	0.07	1.63	1.75	1.88
b	0.01	-	0.01	0.22	-	0.38
C	0.00	-	0.01	0.09	-	0.20
D	0.39	0.40	0.41	9.90	10.20	10.50
E	0.29	0.31	0.32	7.40	7.80	8.20
E1	0.20	0.21	0.22	5.00	5.30	5.60
[e]	0.0259BSC			0.65BSC		
L	0.02	0.04	0.04	0.63	0.90	1.03
R	0.00	-	-	0.09	-	-
$\theta^\circ$	0°	4°	8°	0°	4°	8°

## 16.4 P-DIP 18 PIN



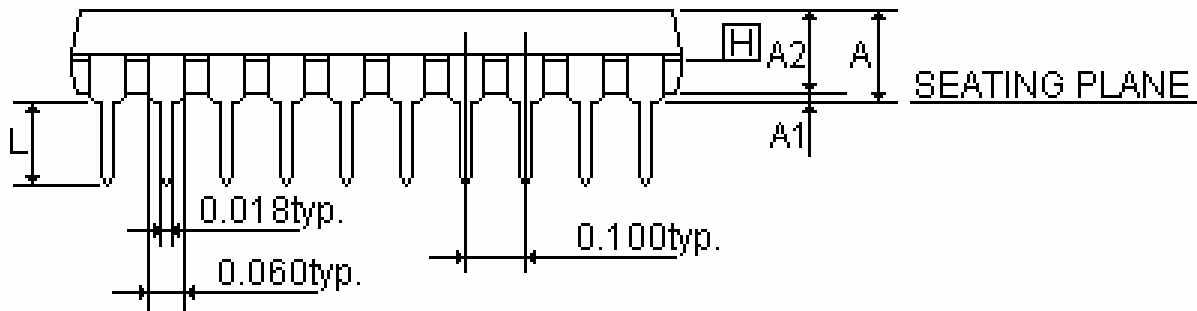
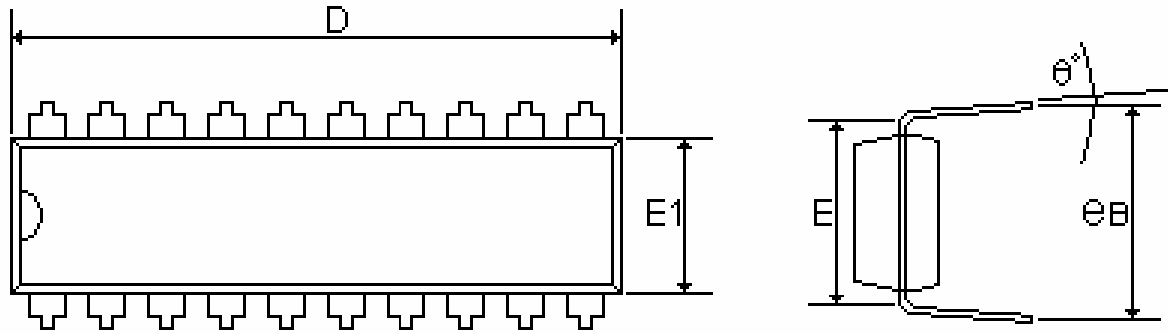
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.880	0.900	0.920	22.352	22.860	23.368
E	0.300			7.620		
E1	0.245	0.250	0.255	6.223	6.350	6.477
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.335	0.355	0.375	8.509	9.017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

## 16.5 SOP 18 PIN



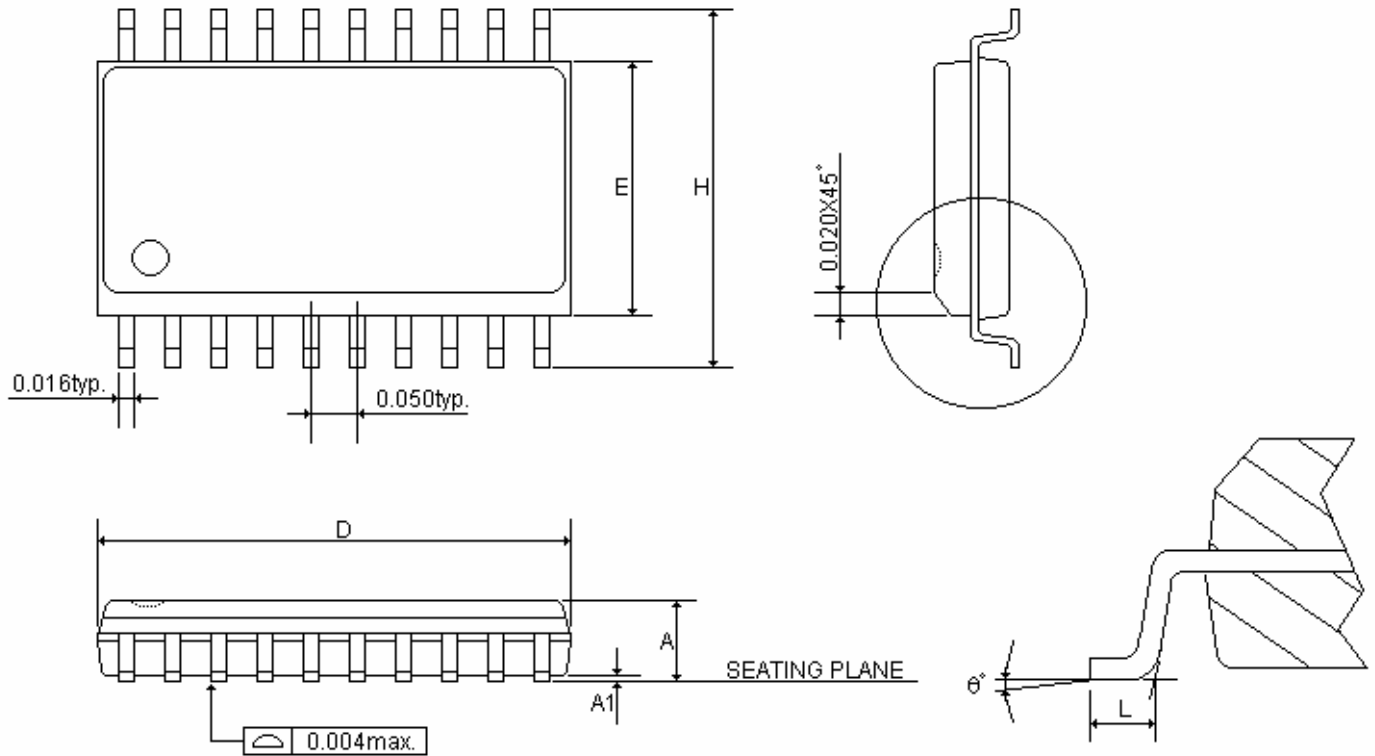
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.093	0.099	0.104	2.362	2.502	2.642
A1	0.004	0.008	0.012	0.102	0.203	0.305
D	0.447	0.455	0.463	11.354	11.557	11.760
E	0.291	0.295	0.299	7.391	7.493	7.595
H	0.394	0.407	0.419	10.008	10.325	10.643
L	0.016	0.033	0.050	0.406	0.838	1.270
$\theta^\circ$	0°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

## 16.6 P-DIP 20 PIN



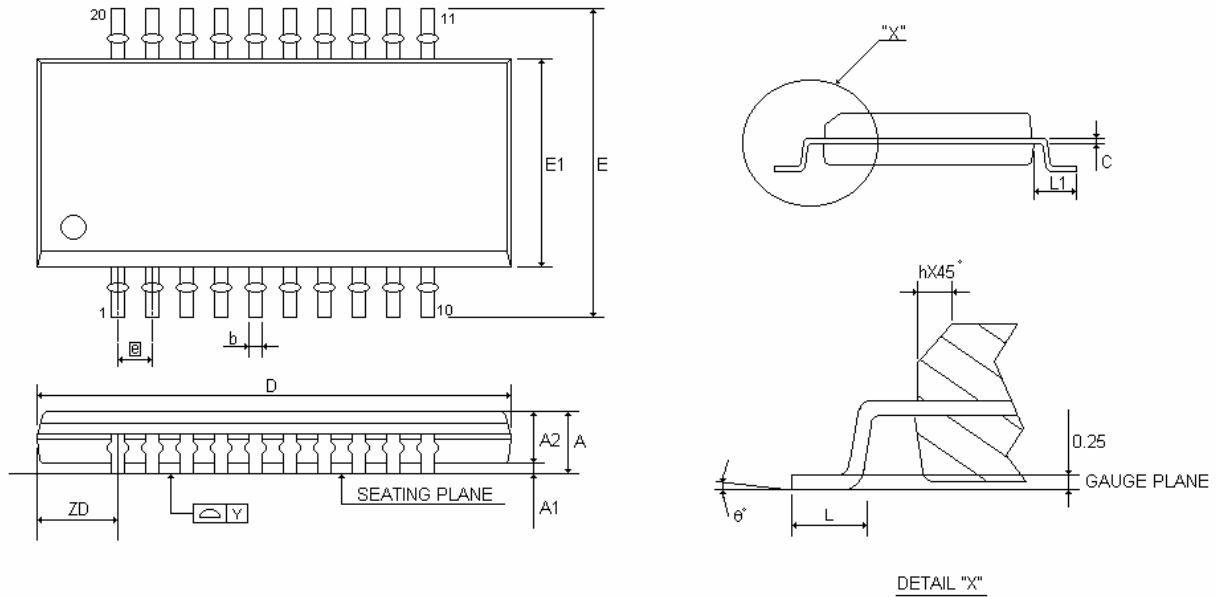
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.980	1.030	1.060	24.892	26.162	26.924
E	0.300			7.620		
E1	0.245	0.250	0.255	6.223	6.350	6.477
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.335	0.355	0.375	8.509	9.017	9.525
θ°	0°	7°	15°	0°	7°	15°

## 16.7 SOP 20 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
<b>A</b>	<b>0.093</b>	<b>0.099</b>	<b>0.104</b>	<b>2.362</b>	<b>2.502</b>	<b>2.642</b>
<b>A1</b>	<b>0.004</b>	<b>0.008</b>	<b>0.012</b>	<b>0.102</b>	<b>0.203</b>	<b>0.305</b>
<b>D</b>	<b>0.496</b>	<b>0.502</b>	<b>0.508</b>	<b>12.598</b>	<b>12.751</b>	<b>12.903</b>
<b>E</b>	<b>0.291</b>	<b>0.295</b>	<b>0.299</b>	<b>7.391</b>	<b>7.493</b>	<b>7.595</b>
<b>H</b>	<b>0.394</b>	<b>0.407</b>	<b>0.419</b>	<b>10.008</b>	<b>10.325</b>	<b>10.643</b>
<b>L</b>	<b>0.016</b>	<b>0.033</b>	<b>0.050</b>	<b>0.406</b>	<b>0.838</b>	<b>1.270</b>
<b>θ°</b>	<b>0°</b>	<b>4°</b>	<b>8°</b>	<b>0°</b>	<b>4°</b>	<b>8°</b>

## 16.8 SSOP 20 PIN



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

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

**总公司：**

地址：台湾新竹县竹北市台元街 36 号 10 楼之一

电话：886-3-5600-888

传真：886-3-5600-889

**台北办事处：**

地址：台北市松德路 171 号 15 楼之 2

电话：886-2-2759 1980

传真：886-2-2759 8180

**香港办事处：**

地址：香港新界沙田沙田乡宁会路 138# 新城市中央广场第一座 7 楼 705 室

电话：852-2723 8086

传真：852-2723 9179

**松翰科技（深圳）有限公司**

地址：深圳市南山区高新技术产业园南区 T2-B 栋 2 层

电话：86-755-2671 9666

传真：86-755-2671 9786

**技术支持：**

Sn8fae@SONiX.com.tw