

SN8P2740 系列

用户参考手册

Version 1.6

SN8P2743
SN8P2742
SN8P27411

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	时间	修改内容
V1.0	2010/4	初版。
V1.1	2011/6	1、修改开发工具章节内容。 2、修改电气特性曲线图。 3、修改 SN8P2742P/S 的烧录引脚配置。
V1.2	2011/10	1、增加 SN8P2741 的相关内容。
V1.3	2012/02	1、删除 SN8P2741 的相关内容。 2、增加 SN8P27411 的相关内容。
V1.4	2012/11	1、调整 SN8P27411 的引脚配置。
V1.5	2012/12	1、调整电气特性章节中 OP AMP CHARACTERISTIC 的部分内容。
V1.6	2012/7	1、调整电气特性章节中 OP AMP CHARACTERSITIC 的 Offset 范围。

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 引脚结构图	10
2 中央处理器 (CPU)	13
2.1 程序存储器 (ROM)	13
2.1.1 复位向量 (0000H)	13
2.1.2 中断向量 (0008H)	14
2.1.3 查表	15
2.1.4 跳转表	17
2.1.5 CHECKSUM计算	19
2.2 数据存储器 (RAM)	20
2.2.1 系统寄存器	21
2.2.2 累加器ACC	23
2.2.3 程序状态寄存器PFLAG	24
2.2.4 程序计数器	25
2.2.5 H, L寄存器	27
2.2.6 Y, Z寄存器	28
2.2.7 R寄存器	28
2.3 寻址模式	29
2.3.1 立即寻址模式	29
2.3.2 直接寻址模式	29
2.3.3 间接寻址模式	29
2.4 堆栈操作	30
2.4.1 概述	30
2.4.2 堆栈寄存器	31
2.4.3 堆栈操作举例	32
2.5 编译选项列表 (CODE OPTION)	33
2.5.1 Fcpu编译选项	33
2.5.2 Reset_Pin编译选项	33
2.5.3 Security编译选项	33
3 复位	34
3.1 概述	34
3.2 上电复位	35
3.3 看门狗复位	35
3.4 掉电复位	36
3.4.1 最低工作电压	36
3.4.2 低电压检测 (LVD)	37
3.4.3 掉电复位性能改进	38
3.5 外部复位	39
3.6 外部复位电路	40
3.6.1 基本RC复位电路	40
3.6.2 二极管& RC复位电路	40
3.6.3 齐纳二极管复位电路	41
3.6.4 电压偏移复位电路	41
3.6.5 外部IC复位电路	42
4 系统时钟	43
4.1 概述	43
4.2 Fcpu (指令周期)	43
4.3 系统高速时钟	44
4.3.1 HIGH_CLK编译选项	44
4.3.2 内部高速RC振荡器 (IHRC)	44
4.3.3 外部高速振荡器	44

4.3.4 外部振荡器应用电路	44
4.4 系统低速时钟	45
4.5 OSCM寄存器	46
4.6 系统时钟测试	46
4.7 系统时钟时序	47
5 系统工作模式	49
5.1 概述	49
5.2 普通模式	50
5.3 低速模式	50
5.4 睡眠模式	50
5.5 绿色模式	51
5.6 工作模式控制宏	51
5.7 系统唤醒	52
5.7.1 概述	52
5.7.2 唤醒时间	52
5.7.3 P1W唤醒控制寄存器	52
6 中断	53
6.1 概述	53
6.2 INTEN中断使能寄存器	54
6.3 INTRQ中断请求寄存器	55
6.4 GIE全局中断	56
6.5 PUSH, POP	56
6.6 外部中断INT0	57
6.7 T0 中断	58
6.8 TC0 中断	59
6.9 ADC中断	60
6.10 比较器中断 (CMP0~CMP2)	61
6.11 多中断操作	62
7 I/O口	63
7.1 概述	63
7.2 I/O口模式	64
7.3 I/O口上拉电阻	65
7.4 I/O口数据寄存器	66
7.5 P4 与ADC共用引脚	67
8 定时器	69
8.1 看门狗定时器	69
8.2 8 位基本定时器T0	71
8.2.1 概述	71
8.2.2 T0 定时器操作	71
8.2.3 TOM模式寄存器	72
8.2.4 TOC计数寄存器	72
8.2.5 T0 操作举例	72
8.3 8 位定时/计数器TC0	73
8.3.1 概述	73
8.3.2 TC0 定时器操作	74
8.3.3 脉冲宽度调制 (PWM)	75
8.3.4 TC0 脉冲产生器功能	76
8.3.5 TC0M模式寄存器	78
8.3.6 TC0C计数寄存器	78
8.3.7 TC0R自动重装寄存器	79
8.3.8 TC0D PWM占空比寄存器	79
8.3.9 TC0 定时器操作举例	80
9 模拟比较器CM0	82
9.1 概述	82
9.2 普通比较器模式	83
9.3 比较器CM0 特殊功能	85
9.4 比较器CM0 模式寄存器	86

9.5	比较器应用注意事项	86
9.6	比较器CM0 操作举例	87
10	模拟比较器CM1	88
10.1	概述	88
10.2	普通比较器模式	89
10.3	比较器CM1 特殊功能	91
10.4	比较器CM1 模式寄存器	92
10.5	比较器应用注意事项	93
10.6	比较器CM1 操作举例	93
11	模拟比较器CM2	94
11.1	概述	94
11.2	普通比较器模式	95
11.3	比较器CM2 特殊功能	97
11.4	比较器CM2 模式寄存器	98
11.5	比较器应用注意事项	99
11.6	比较器CM2 操作举例	99
12	2K/4K BUZZER 产生器	100
12.1	概述	100
12.2	BZM寄存器	100
13	8 通道模拟数字转换 (ADC)	101
13.1	概述	101
13.2	ADC模式寄存器	102
13.3	ADC数据缓存寄存器	103
13.4	ADC操作说明和注意事项	104
13.4.1	ADC信号格式	104
13.4.2	AD转换时间	104
13.4.3	ADC引脚配置	105
13.5	ADC操作举例	106
13.6	ADC应用电路	108
14	RAIL to RAIL OP放大器	109
14.1	概述	109
14.2	OP AMP寄存器	109
15	指令集	110
16	电气特性	111
16.1	极限参数	111
16.2	电气特性	111
16.3	特性曲线	113
17	开发工具	114
17.1	SN8P2743 EV-KIT	114
17.2	ICE和EV-KIT应用注意事项	116
18	OTP烧录引脚	118
18.1	WRITER转接板引脚分配	118
18.2	烧录引脚配置	119
19	单片机正印命名规则	120
19.1	概述	120
19.2	单片机型号说明	120
19.3	命名举例	121
19.4	日期码规则	122
20	封装信息	123
20.1	SK-DIP 24 PIN	123
20.2	SOP 24 PIN	124
20.3	P-DIP 20 PIN	125
20.4	SOP 20 PIN	126
20.5	P-DIP 16 PIN	127
20.6	SOP 16 PIN	128

1 产品简介

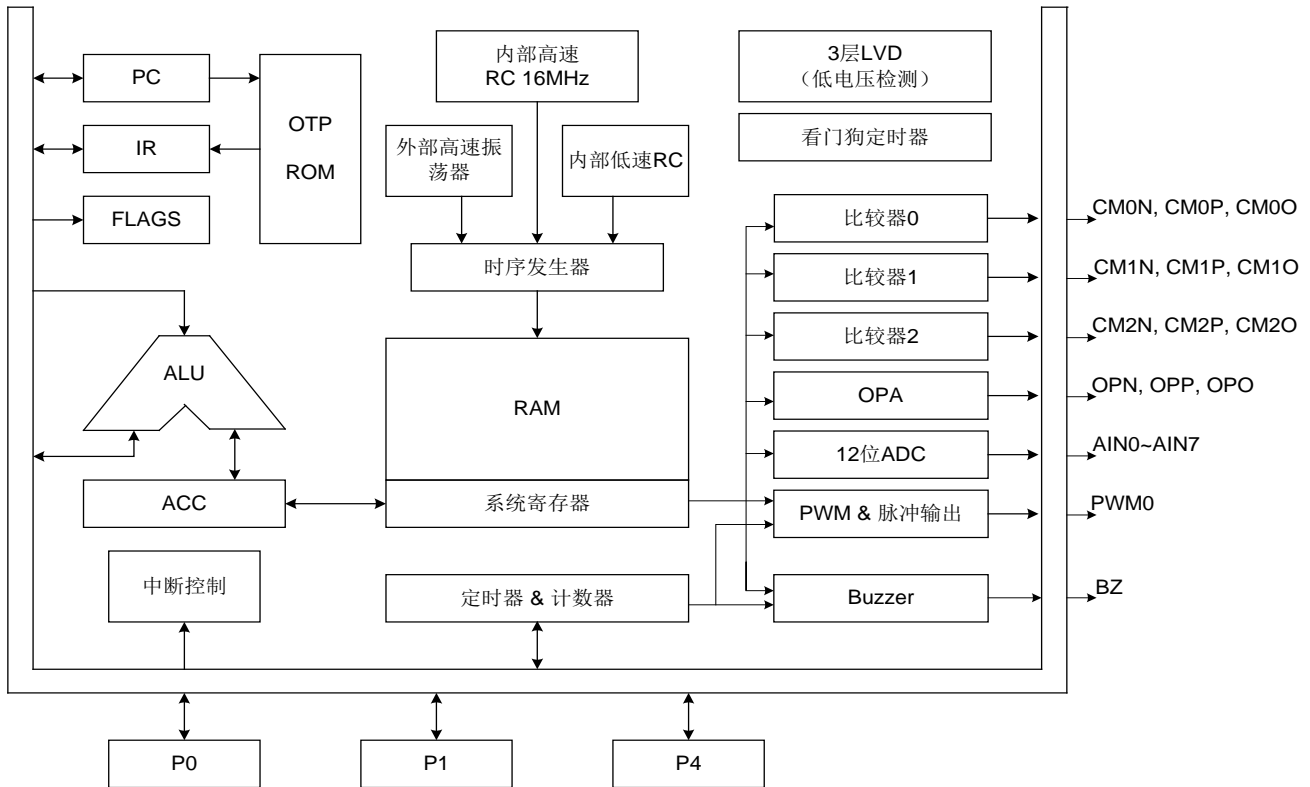
1.1 功能特性

- ◆ **存储器配置**
ROM: 4K * 16 位
RAM: 128 * 8 位
- ◆ **8 层堆栈缓存器**
- ◆ **7 个中断源**
6 个内部中断: T0、TC0、ADC、CM0、CM1、CM2
1 个外部中断: INT0
- ◆ **IO 引脚配置**
双向输入输出 IO 口: P0、P1、P4
具有唤醒功能的 IO 口: P0、P1 电平变换
具有上拉电阻的 IO 口: P0、P1、P4
OP-amp/比较器引脚: P1、P4
ADC 输入引脚: P4.0~P4.7
- ◆ **Fcpu (指令周期)**
Fcpu = Fosc/4、Fosc/8、Fosc/16
- ◆ **3 级 LVD**
2.0V/2.4V/3.6V
- ◆ **功能强大的指令集**
指令的长度为一个字
大多数指令周期为一个时钟周期
JMP 指令可在整个 ROM 区运行。
查表指令 (MOVC) 可寻址整个 ROM 区。
- ◆ **一个 8 位基本定时器 (T0)**
- ◆ **一个 8 位定时器, 带有 PWM 和脉冲产生器 (TC0)**
- ◆ **一个 2K/4K 可编程 Buzzer 输出**
- ◆ **8 通道 12 位 SAR ADC**
- ◆ **1-set rail-to-rail OP-amp**
- ◆ **3-set 比较器**
- ◆ **内置看门狗定时器, 时钟源由内部低速 RC 时钟提供 (16KHz@3V, 32KHz@5V)**
- ◆ **4 种系统时钟**
外部高速时钟: RC, 高达 10MHz
外部高速时钟: 晶体, 高达 16MHz
内部高速时钟: RC, 16MHz
内部低速时钟: RC, 16KHz (3V), 32KHz (5V)
- ◆ **4 种工作模式**
普通模式: 高低速时钟都正常工作
低速模式: 只有低速时钟正常工作
睡眠模式: 高低速时钟都停止工作
绿色模式: 由定时器周期性的唤醒
- ◆ **封装形式**
SKDIP 24 pin
PDIP 20 pin
SOP 24 pin
SOP 20 pin
PDIP 16 pin
SOP 16 pin

☞ 产品选型表

单片机名称	ROM	RAM	堆栈	定时器		PWM	脉冲产生器	Buzzer	外部中断	I/O	ADC	OP-amp	比较器	封装形式
				T0	TC0									
SN8P2743	4K*16	128*8	8	V	V	1	1	1	1	22	8-ch	1	3	SKDIP24 SOP24
SN8P2742	4K*16	128*8	8	V	V	1	1	1	-	18	6-ch	1	3	PDIP20 SOP20
SN8P27411	4K*16	128*8	8	V	V	1	1	1	-	14	6-ch	1	3	PDIP16 SOP16

1.2 系统框图



1.3 引脚配置

SN8P2743K (SKDIP 24 pin)

SN8P2743S (SOP 24 pin)

VSS	1	U	24	VDD
XIN/P0.6	2		23	P4.7/AIN7
XOUT/P0.5/BZ	3		22	P4.6/AIN6
RST/VPP/P0.4	4		21	P4.5/AIN5
P0.0/INT0	5		20	P4.4/AIN4
P0.1/PWM0	6		19	P4.3/AIN3/CM0O
P0.2/CM0P	7		18	P4.2/AIN2/CM1O
P0.3/CM0N	8		17	P4.1/AIN1/CM2O
P1.6/CM1P	9		16	P4.0/AIN0/AVREFH
P1.5/CM1N	10		15	P1.0/OPN
P1.4/CM2P	11		14	P1.1/OPP
P1.3/CM2N	12		13	P1.2/OPO

SN8P2742P (DIP 20 pin)

SN8P2742S (SOP 20 pin)

VSS	1	U	20	VDD
XIN/P0.6	2		19	P4.5/AIN5
XOUT/P0.5/BZ	3		18	P4.4/AIN4
RST/VPP/P0.4/P0.1/PWM0	4		17	P4.3/AIN3/CM0O
P0.2/CM0P	5		16	P4.2/AIN2/CM1O
P0.3/CM0N	6		15	P4.1/AIN1/CM2O
P1.6/CM1P	7		14	P4.0/AIN0/AVREFH
P1.5/CM1N	8		13	P1.0/OPN
P1.4/CM2P	9		12	P1.1/OPP
P1.3/CM2N	10		11	P1.2/OPO

SN8P27411P (DIP 16 pin)

VDD	1	U	16	P4.7/AIN7
XOUT/P0.5/BZ	2		15	P4.6/AIN6
RST/VPP/P0.4/P0.1/PWM0	3		14	P4.4/AIN4
P0.2/CM0P	4		13	VSS
P0.3/CM0N	5		12	P4.2/AIN2/CM1O
P1.5/CM1N	6		11	P4.1/AIN1/CM2O
P1.3/CM2N	7		10	P4.0/AIN0/AVREFH
P4.3/OPO	8		9	P1.0/OPN

* OPP (OPA 的正极引脚) 连接到 GND

SN8P27411S (SOP 16 pin)

VDD	1	U	16	VSS
XOUT/P0.5/BZ	2		15	P4.7/AIN7
RST/VPP/P0.4/P0.1/PWM0	3		14	P4.6/AIN6
P0.2/CM0P	4		13	P4.4/AIN4
P0.3/CM0N	5		12	P4.2/AIN2/CM1O
P1.5/CM1N	6		11	P4.1/AIN1/CM2O
P1.3/CM2N	7		10	P4.0/AIN0/AVREFH
P4.3/OPO	8		9	P1.0/OPN

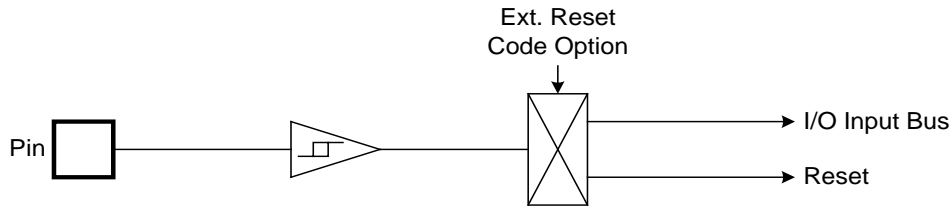
* OPP (OPA 的正极引脚) 连接到 GND

1.4 引脚说明

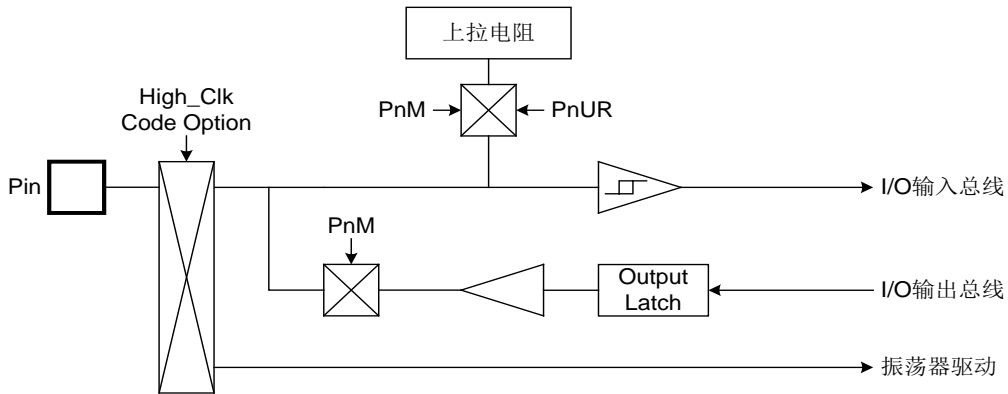
引脚名称	引脚类型	功能说明
VDD,VSS	P	电源输入端。
P0.4/RST/VPP	I,P	RST: 系统外部复位输入引脚, 施密特触发, 低电平有效, 通常保持高电平。 VPP: OTP 烧录引脚。 P0.4: 单向输入引脚, 施密特触发, 无上拉电阻。电平变换时可唤醒系统。
XIN/P0.6	I/O	XIN: 外部振荡信号输入引脚。 P0.6: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻, 具有唤醒功能。
XOUT/P0.5/BZ	I/O	XOUT: 外部振荡信号输出引脚。 P0.5: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻, 具有唤醒功能。 BZ: 2K/4K 可编程 Buzzer 输出引脚。
P0.0/INT0	I/O	P0.0: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻。 INT0: 外部中断 INT0 输入引脚。
P0.1/PWM0	I/O	P0.1: 漏极开路的输出引脚。 PWM0: PWM 输出引脚和脉冲输出引脚。
P0.2/CM0P	I/O	P0.2: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻, 具有唤醒功能。 CM0P: 比较器的正极输入引脚。
P0.3/CM0N	I/O	P0.3: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻, 具有唤醒功能。 CM0N: 比较器的负极输入引脚。
P1.0/OPN	I/O	P1.0: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻, 具有唤醒功能。 OPN: OP amp 的负极输入引脚。
P1.1/OPP	I/O	P1.1: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻, 具有唤醒功能。 OPP: OP amp 的正极输入引脚。
P1.2/OPO	I/O	P1.2: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻, 具有唤醒功能。 OPO: OP amp 的输出引脚。
P1.3/CM2N	I/O	P1.3: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻, 具有唤醒功能。 CM2N: 比较器的负极输入引脚。
P1.4/CM2P	I/O	P1.4: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻, 具有唤醒功能。 CM2P: 比较器的正极输入引脚。
P1.5/CM1N	I/O	P1.5: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻, 具有唤醒功能。 CM1N: 比较器的负极输入引脚。
P1.6/CM1P	I/O	P1.6: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻, 具有唤醒功能。 CM1P: 比较器的正极输入引脚。
P4.0/AIN0/AVREFH	I/O	P4.0: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻。 AIN0: ADC 模拟输入引脚。 AVREFH: ADC 参考电压高电平输入引脚。
P4.1/AIN1/CM2O	I/O	P4.1: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻。 AIN1: ADC 模拟输入引脚。 CM2O: 比较器的输出引脚。
P4.2/AIN2/CM1O	I/O	P4.2: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻。 AIN2: ADC 模拟输入引脚。 CM1O: 比较器的输出引脚。
P4.3/AIN3/CM0O	I/O	P4.3: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻。 AIN3: ADC 模拟输入引脚。 CM0O: 比较器的输出引脚。
P4.4/AIN4	I/O	P4.4: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻。 AIN4: ADC 模拟输入引脚。
P4.5/AIN5	I/O	P4.5: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻。 AIN5: ADC 模拟输入引脚。
P4.6/AIN6	I/O	P4.6: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻。 AIN6: ADC 模拟输入引脚。
P4.7/AIN7	I/O	P4.7: 双向输入输出引脚, 输入模式时施密特触发, 内置上拉电阻。 AIN7: ADC 模拟输入引脚。

1.5 引脚结构图

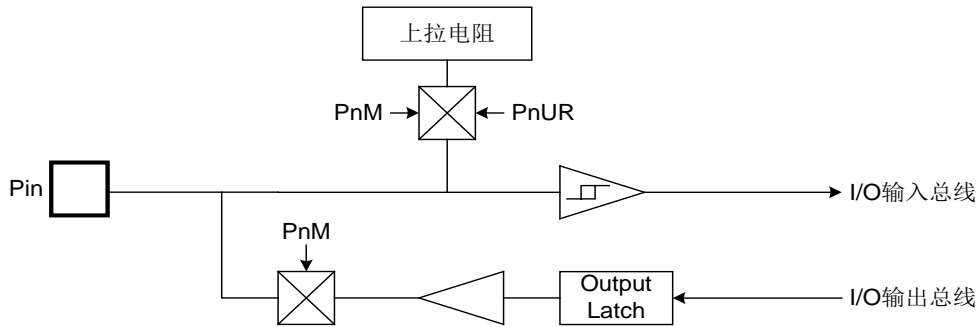
- 复位引脚



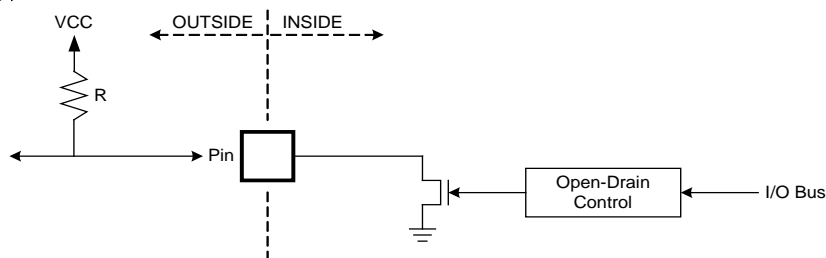
- 振荡器引脚



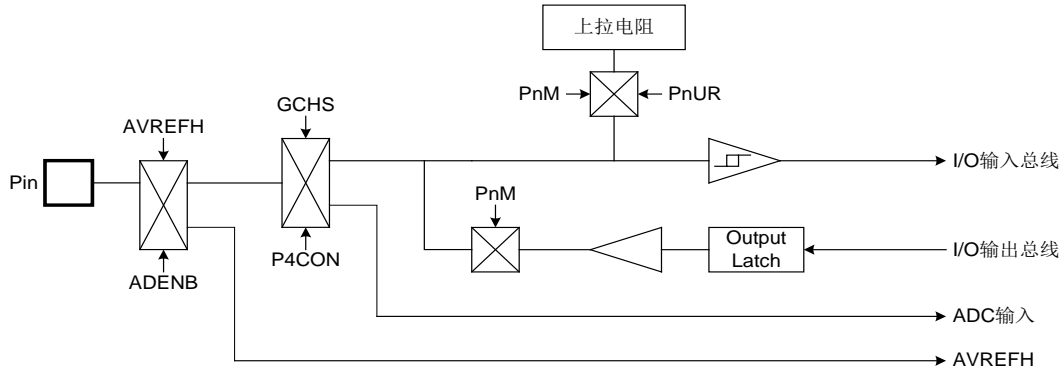
- GPIO 引脚



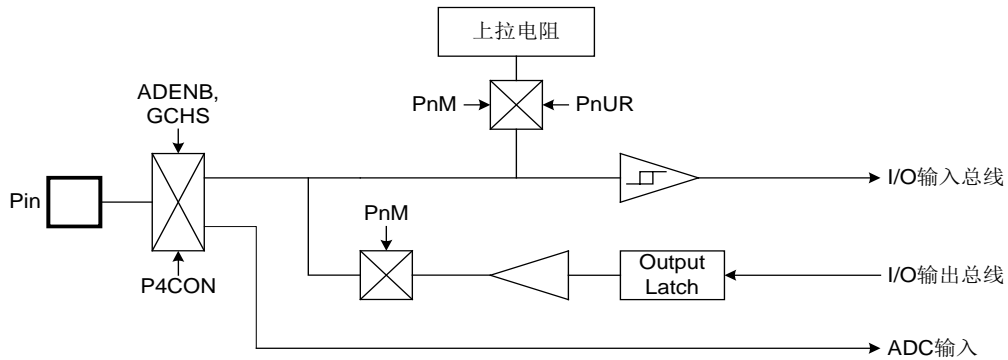
- P0.1: 漏极开路引脚



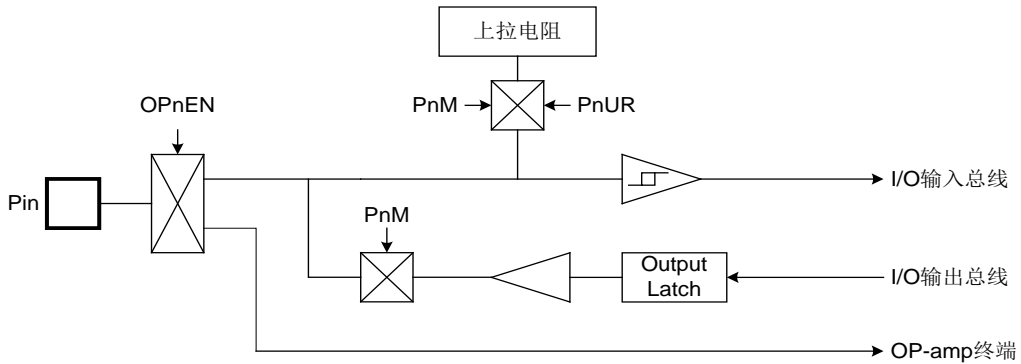
● **ADC 输入引脚，与参考电压高电平共用**



● **ADC 输入引脚**

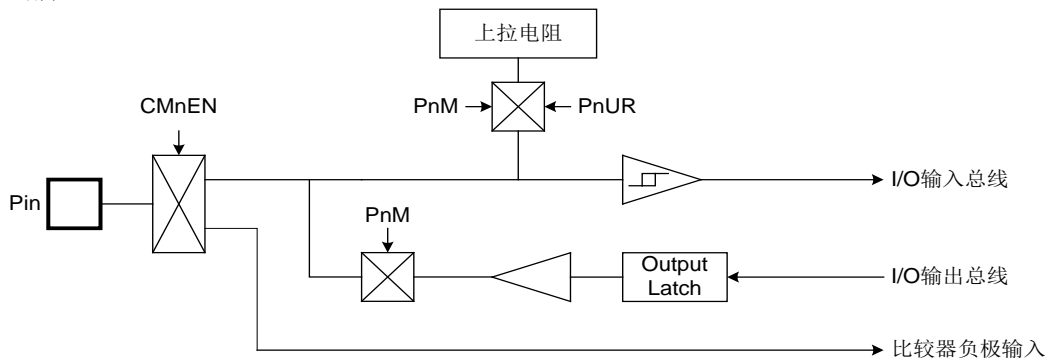


● **OP-amp 引脚**

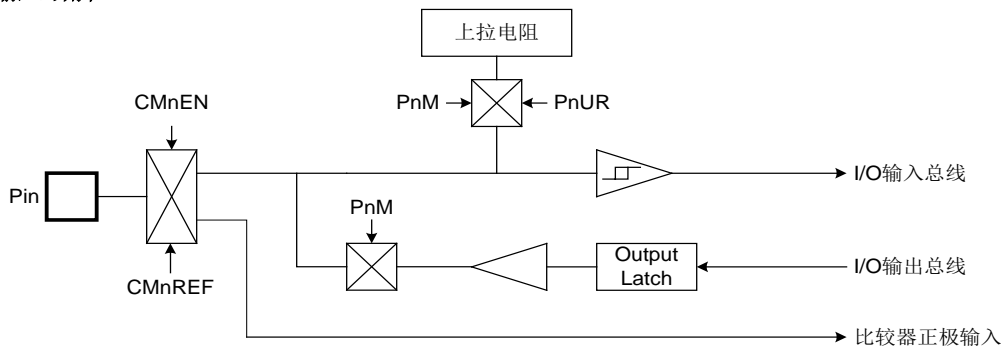


● 比较器引脚

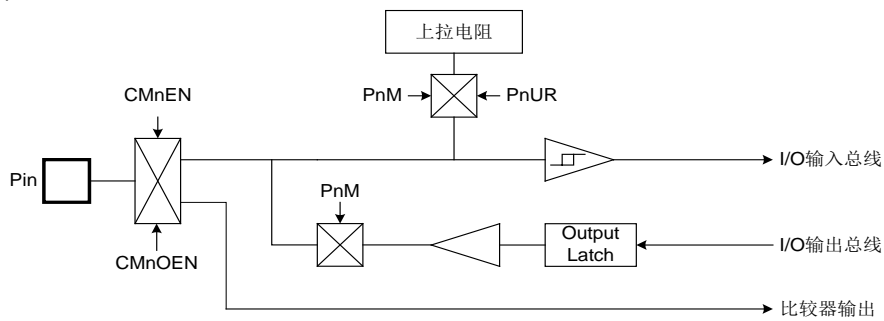
比较器负极输入引脚:



● 比较器正极输入引脚:



● 比较器输出引脚:



2 中央处理器 (CPU)

2.1 程序存储器 (ROM)

☞ ROM: 4K



ROM 包含复位向量、中断向量、通用存储区域和系统保留部分。复位向量是程序的开始位置，中断向量则是中断服务程序的开始位置，而通用存储区域则保存主程序、子程序及数据表格。

2.1.1 复位向量 (0000H)

具有一个字长的系统复位向量 (0000H)。

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

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

➤ 例：定义复位向量。

```

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

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

```

2.1.2 中断向量（0008H）

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

* 注：“PUSH”、“POP”指令只保存 ACC/PFLAG，无 NT0 和 NPD。PUSH/POP 缓存器只有一层。

➤ 例：定义中断向量，中断服务程序紧随 ORG 8 之后。

```
.CODE
    ORG      0          ; 0000H。
    JMP      START     ; 跳转到用户程序。
    ...

    ORG      8H        ; 中断向量。
    PUSH                     ; 保存 ACC 和 PFLAG。
    ...
    ...
    POP                     ; 恢复 ACC 和 PFLAG。
    RETI                    ; 中断返回。
    ...

START:
    ...                    ; 用户程序开始。
    ...                    ; 用户程序。
    ...
    JMP      START     ;
    ...

    ENDP                    ; 程序结束。
```

➤ 例：定义中断向量。中断服务程序在用户程序之后。

```
.CODE
    ORG      0          ; 0000H。
    JMP      START     ; 跳转到用户程序。
    ...

    ORG      8H        ; 中断向量。
    JMP      MY_IRQ    ; 0008H，跳转到中断服务程序。

START:
    ORG      10H       ; 0010H，用户程序开始。
    ...                ; 用户程序。
    ...
    ...
    JMP      START     ;

MY_IRQ:
    ...                ; 中断服务程序开始。
    PUSH                     ; 保存 ACC 和 PFLAG。
    ...
    ...
    POP                     ; 恢复 ACC 和 PFLAG。
    RETI                    ; 中断返回。
    ...
    ENDP                    ; 程序结束。
```

* 注：从上面的程序中容易得出 SONiX 的编程规则，有以下几点：

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

2.1.3 查表

在 SONiX 单片机中，对 ROM 区中的数据进行查找，查表指针存放在寄存器 Y，Z 中。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$M      ; 设置 table1 的中间字节地址。
B0MOV      Z, #TABLE1$L      ; 设置 table1 的低位字节地址。
MOVC                               ; 查表，R = 00H，ACC = 35H。

                               ; 查找下一地址。
INCMS      Z                  ; Z+1。
JMP        @F                 ; Z 没有溢出。
INCMS      Y                  ; Z 溢出，Y=Y+1。
NOP

@@:        MOVC                               ; 查表，R = 51H，ACC = 05H。
...

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

```

* 注：当寄存器 Z 溢出（从 FFH 变成 00H）时，Y 寄存器并不会自动加 1。用户必须注意这种情况以免查表错误。若 Z 溢出，Y 必须由程序加 1，下面的宏指令 INC_YZ 可以对 Y 和 Z 寄存器自动处理。

➤ 例：宏指令 INC_YZ。

```

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

           INCMS      Y                  ; Y+1
           NOP                               ; 没有溢出。

@@:

           ENDM

```

➤ 例：通过宏指令 INC_YZ 对上例优化。

```

B0MOV      Y, #TABLE1$M      ; 设置 table1 的中间字节地址。
B0MOV      Z, #TABLE1$L      ; 设置 table1 的低位字节地址。
MOVC                               ; 查表，R = 00H，ACC = 35H。

           INC_YZ                       ; 查找下一地址。
                               ;
@@:        MOVC                               ; 查表，R = 51H，ACC = 05H。
...

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

```

下面的程序通过累加器对 Y 和 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：通过指令 **B0ADD/ADD** 实现查表功能。

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

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

```
B0BTS1   FC              ; 检查进位标志。
JMP      GETDATA        ; FC = 0。
INCMS    Y               ; FC = 1, Y+1。
NOP
```

```
GETDATA:                                     ;
MOV      MOVC             ; 存储数据，如果 BUF = 0，数据为 35H。
                                     ; 如果 BUF = 1，数据=5105H。
                                     ; 如果 BUF = 2，数据=2012H
```

...

```
TABLE1:  DW      0035H    ; 定义数据表数据（16-bit）。
          DW      5105H
          DW      2012H
          ...
```


2.1.4 跳转表

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

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

➤ 例：跳转表。

```

ORG      0100H      ; 跳转表从 ROM 前端开始。

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

```

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

➤ 例：如果跳转表跨越 ROM 边界，将引起程序错误。

```

@JMP_A   MACRO      VAL
          IF        (($+1) !& 0XFF00) != (($+(VAL)) !& 0XFF00)
          JMP      ($ | 0XFF)
          ORG      ($ | 0XFF)
          ENDIF
          B0ADD    PCL, A
          ENDM

```

* 注：“VAL”为跳转表列表中列表个数。

➤ 例：宏“MACRO3.H”中，“@JMP_A”的应用。

```

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

```

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

➤ 例：“@JMP_A”运用举例

; 编译前

ROM 地址

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

; 编译后

ROM 地址

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

2.1.5 CHECKSUM计算

ROM 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      C
B0BCLR  FC                ; 清标志位 C。
        ADD      DATA1, A
        MOV      A, R
        ADC      DATA2, A
        JMP      END_CHECK ; 检查 YZ 地址是否为代码的结束地址。

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

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

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

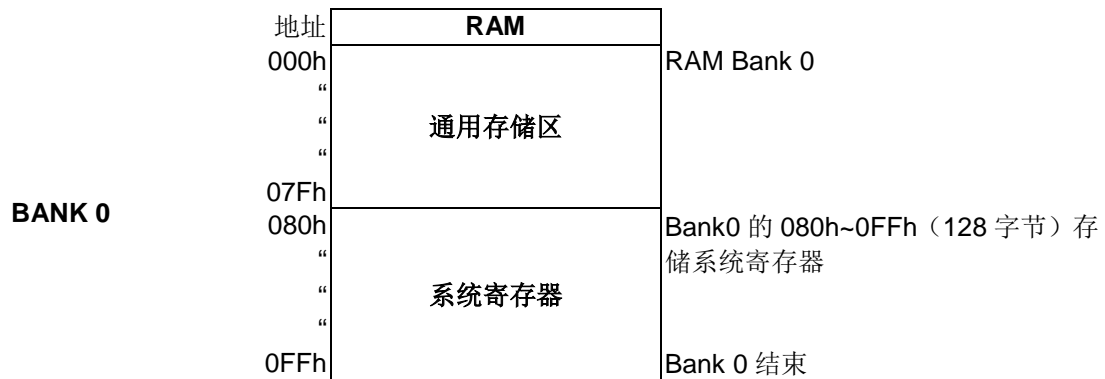
CHECKSUM_END:
        ...
        ...

END_USER_CODE:
        ; 程序结束。

```

2.2 数据存储器（RAM）

RAM: 128*8-bit



128 字节的通用存储区分别位于 Bank0。SONiX 提供了“Bank0”指令（如 B0MOV, B0ADD, B0BTS1, B0BSET 等），可以在非 Bank0 区直接访问 Bank0。

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	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	CMDB0	CMDB1	CM0M	CM1M	CM2M	OPM
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	P4CON	-
B	-	ADM	ADB	ADR	ADT	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	-	-	P4M	-	-	-	INTRQ	INTEN	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	P1	-	-	P4	-	-	-	T0M	T0C	TC0M	TC0C	BZM	-	-	STKP
E	P0UR	P1UR	-	-	P4UR	-	@HL	@YZ	TC0D	-	-	-	-	-	-	-
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

H, L = 工作寄存器, @HL 间接寻址寄存器	Y, Z = 工作寄存器, @YZ 间接寻址寄存器, ROM 寻址寄存器
R = 工作寄存器, ROM 查表数据寄存器	PFLAG = 特殊标志寄存器
CMDB0= 比较器输出 de-bounce 控制寄存器 0	CMDB1= 比较器输出 de-bounce 控制寄存器 1
CM0M= 比较器 0 模式寄存器	CM1M = 比较器 1 模式寄存器
CM2M = 比较器 2 模式寄存器	OPM = OP 放大器 0~2 模式寄存器
P4CON = P4 配置寄存器	ADM = ADC 模式寄存器
ADB = ADC 数据缓存器	ADR = ADC 精度选择寄存器
ADT = ADC 偏移校准寄存器	PEDGE = P0.0、P0.1、P0.2 边沿方向寄存器
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	WDTR = 看门狗清零寄存器
PnM = Pn 输入输出模式寄存器	Pn = Pn 数据寄存器
PnUR = Pn 上拉电阻控制寄存器	PCH, PCL = 程序计数器
T0M = T0 模式寄存器	T0C = T0 计数寄存器
TC0M = TC0 模式寄存器	TC0C = TC0 计数寄存器
TC0R = TC0 自动重装缓存器	TC0D= TC0 占空比控制寄存器
BZM= 2K/4K buzzer 模式寄存器	@HL = 间接寻址寄存器
@YZ = 间接寻址寄存器	STKP = 堆栈指针缓存器
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	R/W	说明
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	LVD36	LVD24		C	DC	Z	R/W	PFLAG
09AH	CM1D3	CM1D2	CM1D1	CM1D0	CM0D3	CM0D2	CM0D1	CM0D0	R/W	CMDB0
09BH					CM2D3	CM2D2	CM2D1	CM2D0	R/W	CMDB1
09CH	CM0EN	CM0OEN	CM0OUT	CM0SF	CM0G				R/W	CM0M
09DH	CM1EN	CM1OEN	CM1OUT	CM1SF	CM1G	CM2RS2	CM2RS1	CM2RS0	R/W	CM1M
09EH	CM2EN	CM2OEN	CM2OUT	CM2SF	CM2G	CM2RS2	CM2RS1	CM2RS0	R/W	CM2M
09FH								OPEN	R/W	OPM
0AEH	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	W	P4CON
0B1H	ADENB	ADS	EOC	GCHS	AVREFH	CHS2	CHS1	CHS0	R/W	ADM
0B2H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R	ADB
0B3H		ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0	R/W	ADR
0B4H	ADTS1	ADTS0		ADT4	ADT3	ADT2	ADT1	ADT0	R/W	ADT
0B8H		P06M	P05M	-	P03M	P02M	-	P00M	R/W	P0M
0BFH							P00G1	P00G0	R/W	PEDGE
0C0H		P16W	P15W	P14W	P13W	P12W	P11W	P10W	W	P1W
0C1H		P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C4H	P47M	P46M	P45M	P44M	P43M	P42M	P41M	P40M	R/W	P4M
0C8H	ADCIRQ		TC0IRQ	T0IRQ	CM2IRQ	CM1IRQ	CM0IRQ	P00IRQ	R/W	INTRQ
0C9H	ADCIEN		TC0IEN	T0IEN	CM2IEN	CM1IEN	CM0IEN	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					PC11	PC10	PC9	PC8	R/W	PCH
0D0H		P06	P05	P04	P03	P02	P01	P00	R/W	P0
0D1H		P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D4H	P47	P46	P45	P44	P43	P42	P41	P40	R/W	P4
0D8H	T0ENB	T0rate2	T0rate1	T0rate0					R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	TC0DIR	TC0PO	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	BZEN	BZRate1	BZrate0						R/W	BZM
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H		P06R	P05R	-	P03R	P02R	-	P00R	W	P0UR
0E1H		P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E4H	P47R	P46R	P45R	P44R	P43R	P42R	P41R	P40R	W	P4UR
0E6H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E8H	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0	R/W	TC0D
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H					S7PC11	S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H					S6PC11	S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H					S5PC11	S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H					S4PC11	S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H					S3PC11	S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH					S2PC11	S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH					S1PC11	S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH					S0PC11	S0PC10	S0PC9	S0PC8	R/W	STK0H

* 注:

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

2.2.2 累加器ACC

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

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

➤ **例：读/写 ACC。**

; 将立即数写入 ACC。

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

;把 ACC 中的数据存入 BUF 中。

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

; 把 BUF 中的数据送到 ACC 中。

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

系统执行中断操作时，ACC 和 PFLAG 的值不会自动存储，用户必须使用 PUSH 和 POP 指令来保存 ACC 和 PFLAG。

➤ **例：ACC 和工作寄存器中断保护操作。**

INT_SERVICE:

```
PUSH                                ; 保存 ACC 和 PFALG。
```

```
...
```

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

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

2.2.3 程序状态寄存器PFLAG

寄存器PFLAG中包含ALU运算状态信息、系统复位状态信息和LVD低电压检测状态信息。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。LVD24、LVD36显示LVD检测电压状态。

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

Bit [7:6] **NT0, NPD**: 复位状态标志位。

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

Bit 5 **LVD36**: LVD 3.6V 工作电压标志，LVD 编译选项为 LVD_H 时有效。

- 0 = 无效 (VDD > 3.6V);
- 1 = 有效 (VDD ≤ 3.6V)。

Bit 4 **LVD24**: LVD 2.4V 工作电压标志，LVD 编译选项为 LVD_M 时有效。

- 0 = 无效 (VDD > 2.4V);
- 1 = 有效 (VDD ≤ 2.4V)。

Bit 2 **C**: 进位标志。

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

Bit 1 **DC**: 辅助进位标志。

- 1 = 加法运算时低四位有进位，或减法运算后没有向高四位借位;
- 0 = 加法运算时低四位没有进位，或减法运算后有向高四位借位。

Bit 0 **Z**: 零标志。

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

* 注：关于标志位 C、DC 和 Z 的更多信息请参阅指令集相关内容。

2.2.4 程序计数器

程序计数器 PC 是一个 12 位二进制程序地址寄存器，分高 4 位和低 8 位。专门用来存放下一条需要执行指令的地址。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	-	-	-	-	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
	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
JMP       C0STEP

C0STEP:   ...
          NOP

```

执行加 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

```

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

C0STEP:   ...
          NOP

```

INCMS:

```

INCMS     INCS       BUF0
          JMP       C0STEP     ;如果 BUF0 不为“0”，则跳至 C0STEP。

```

C0STEP:

```

...
NOP

```

执行减 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

DECS:

```

DECS     DECS       BUF0
          JMP       C0STEP     ;如果 ACC 不为“0”，则跳至 C0STEP。

```

C0STEP:

```

...
NOP

```

DECMS:

```

DECMS     DECS       BUF0
          JMP       C0STEP     ;如果 BUF0 不为“0”，则跳至 C0STEP。

```

C0STEP:

```

...
NOP

```

☞ 多地址跳转

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

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

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

; PC = 0323H

```
MOV      A, #28H
B0MOV    PCL, A      ; 跳到地址 0328H。
...
```

; PC = 0328H

```
MOV      A, #00H
B0MOV    PCL, A      ; 跳到地址 0300H。
...
```

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

; PC = 0323H

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

2.2.5 H, 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
H	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	-	-	-	-	-	-	-	-

080H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
L	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	-	-	-	-	-	-	-	-

- 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
...
...
```

2.2.6 Y, Z寄存器

寄存器 Y 和 Z 都是 8 位缓存器，主要用途如下：

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

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

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

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

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

➤ 例：利用数据指针 @YZ 对 RAM 数据清零。

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

CLR_YZ_BUF:

```
CLR      @YZ          ; @YZ 清零。

DECMS   Z             ;
JMP     CLR_YZ_BUF    ; 不为零。
```

END_CLR:
...

2.2.7 R寄存器

8 位缓存器 R 主要有以下两个功能：

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

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

* 注：关于 R 寄存器查表应用，请查看“查表”章节的内容。

2.3 寻址模式

2.3.1 立即寻址模式

将立即数直接送入 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：立即寻址模式中，指定的 RAM 单元必须是 80H~87H 的工作寄存器。

2.3.2 直接寻址模式

通过 ACC 对 RAM 单元数据进行操作。

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

2.3.3 间接寻址模式

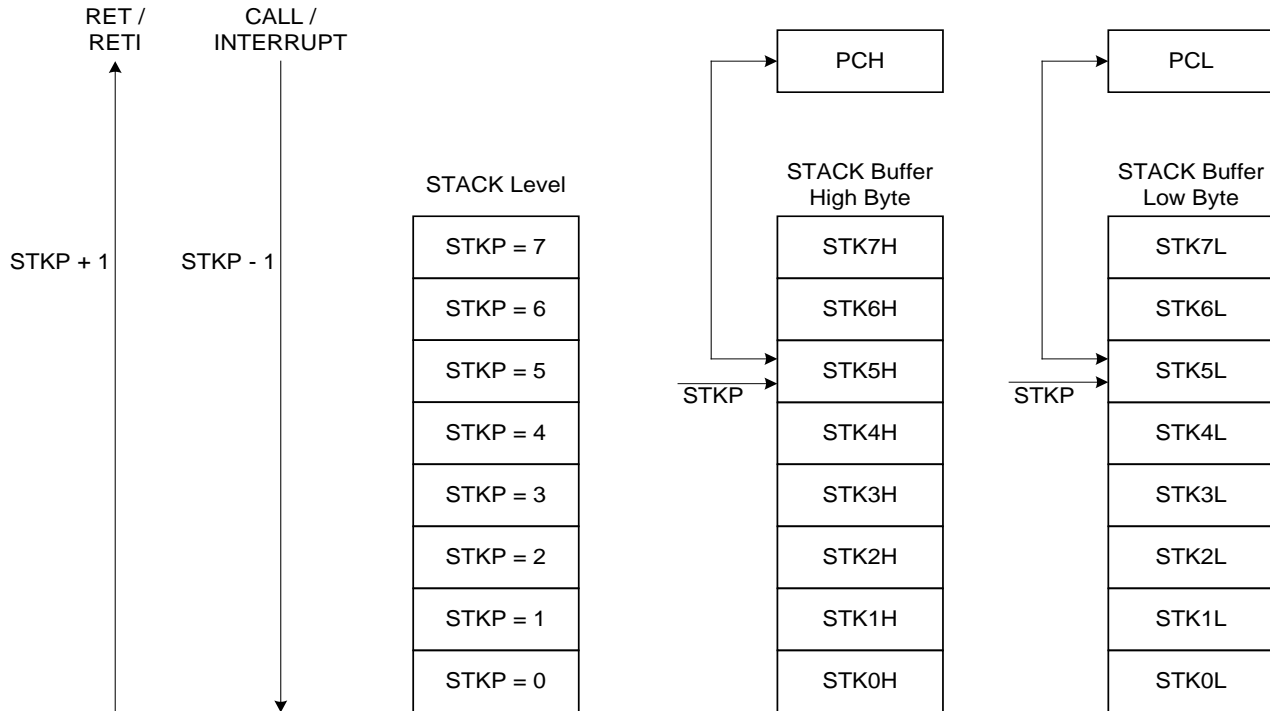
通过数据指针（H/L、Y/Z）对数据存储单元进行读写。

- 例：通过指针@HL 间接寻址。
B0MOV H, #0 ; 清“H”以寻址 RAM bank 0。
B0MOV L, #12H ; 设定寄存器地址。
B0MOV A, @HL
- 例：通过指针@YZ 间接寻址。
B0MOV Y, #0 ; 清“Y”以寻址 RAM bank 0。
B0MOV Z, #12H ; 设定寄存器地址。
B0MOV A, @YZ

2.4 堆栈操作

2.4.1 概述

SN8P2743/42 的堆栈缓存器共有 8 层，程序进入中断或执行 CALL 指令时，用来存储程序计数器 PC 的值。寄存器 STKP 为堆栈指针，指向堆栈缓存器顶层，STK_nH 和 STK_nL 分别是各堆栈缓存器高、低字节。



2.4.2 堆栈寄存器

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

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

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

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

Bit[2:0] **STKPBn**: 堆栈指针 (n = 0 ~ 2)。

Bit 7 **GIE**: 全局中断控制位。

0 = 禁止;

1 = 允许。

➤ 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, #0000111B
B0MOV   STKP, A
```

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

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

2.4.3 堆栈操作举例

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

堆栈层数	STKP 寄存器			堆栈缓存器		说明
	STKPB2	STKPB1	STKPB0	高字节	低字节	
0	1	1	1	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，出错

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

堆栈层数	STKP 寄存器			堆栈缓存器		说明
	STKPB2	STKPB1	STKPB0	高字节	低字节	
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 编译选项列表 (CODE OPTION)

编译选项 (CODE OPTION) 是一种系统的硬件配置，包括振荡器的类型，看门狗定时器的工作，LVD 选项，复位引脚选项以及 OTP ROM 的安全控制。如下表所示：

编译选项	配置项目	功能说明
High_Clk	IHRC_16M	高速内部 16MHz RC 振荡器，XIN/XOUT 作为普通的 I/O 引脚。
	RC	外部高速振荡器采用廉价的 RC 振荡电路，XIN 连接 RC 电路，XOUT 作为普通的 I/O 引脚。
	32K X'tal	外部高速振荡器采用低频振荡器（如 32768Hz）。
	12M X'tal	外部高速振荡器采用高频振荡器（如 12MHz）。
	4M X'tal	外部高速振荡器采用标准振荡器（如 4MHz）。
Fcpu	Fhosc/4	指令周期为 4 个振荡时钟。
	Fhosc/8	指令周期为 8 个振荡时钟。
	Fhosc/16	指令周期为 16 个振荡时钟。
Watch_Dog	Always_On	始终开启看门狗定时器，在睡眠模式和绿色模式下也处于开启状态。
	Enable	使能看门狗定时器，但在睡眠模式和绿色模式下处于关闭状态。
	Disable	关闭看门狗定时器。
Reset_Pin	Reset	使能外部复位引脚。
	P04	使能 P0.4 为单向输入引脚，无上拉电阻。
Security	Enable	ROM 代码加密。
	Disable	ROM 代码不加密。
LVD	LVD_L	如果 VDD 低于 2.0V，LVD 复位系统。
	LVD_M	如果 VDD 低于 2.0V，LVD 复位系统。 寄存器 PFLAG 的 LVD24 位作为 2.4V 低电压的监测器。
	LVD_H	如果 VDD 低于 2.4V，LVD 复位系统。 寄存器 PFLAG 的 LVD36 位作为 3.6V 低电压的监测器。
	LVD_MAX	如果 VDD 低于 3.6V，LVD 复位系统。

2.5.1 Fcpu编译选项

Fcpu 指在普通模式下的指令周期。低速模式下，系统时钟源由内部低速 RC 振荡电路提供，Fcpu 不受 Fcpu 编译选项的影响，固定为 Fhosc/4（16KHz/4@3V，32KHz/4@5V）。

2.5.2 Reset_Pin编译选项

复位引脚与单向输入引脚共用，由编译选项控制。

- **Reset:** 使能外部复位引脚功能。当下降沿触发时，系统复位。
- **P04:** 使能 P0.4 为单向输入引脚。此时禁止外部复位引脚功能。

2.5.3 Security编译选项

Security 编译选项是对 OTP ROM 的一种保护，当使能 Security 编译选项，ROM 代码加密。

3 复位

3.1 概述

SN8P2743/42 有以下几种复位方式：

- 上电复位；
- 看门狗复位；
- 掉电复位；
- 外部复位。

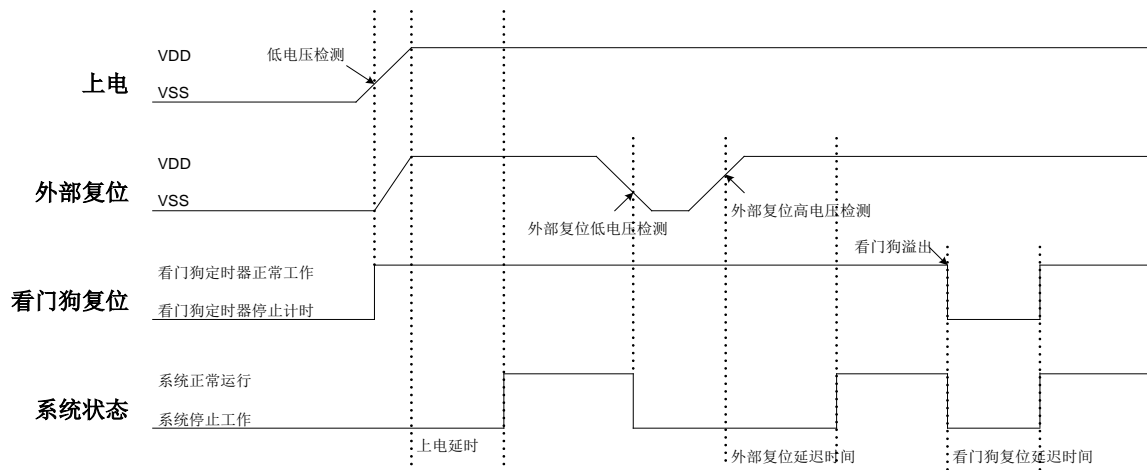
上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器 PC 清零。复位结束后，系统从向量 0000H 处重新开始运行。PFLAG 寄存器的 NT0 和 NPD 两个标志位能够给出系统复位状态的信息。用户可以编程控制 NT0 和 NPD，从而控制系统的运行路径。

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

Bit [7:6] **NT0, NPD**: 复位状态标志位。

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	保留	-
1	0	上电复位和 LVD 复位	电源电压低于 LVD 检测电压
1	1	外部复位	外部复位引脚检测到低电平

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



3.2 上电复位

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

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

3.3 看门狗复位

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

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

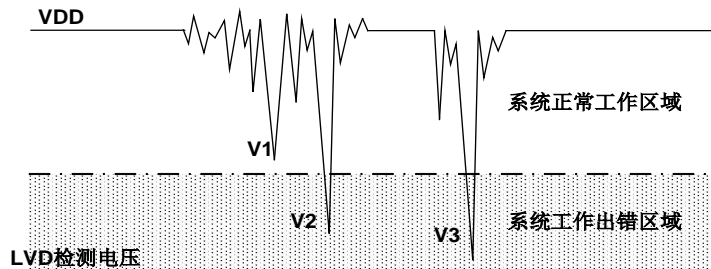
看门狗定时器应用注意事项如下：

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

* 注：关于看门狗定时器的详细内容，请参阅“看门狗定时器”有关章节。

3.4 掉电复位

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



掉电复位示意图

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

DC 运用中：

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

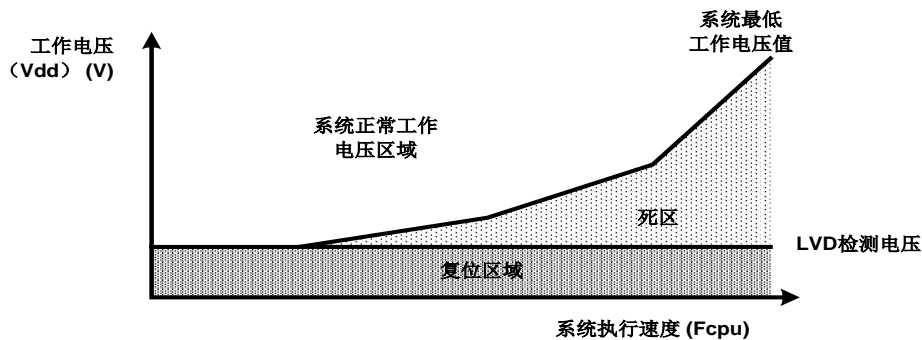
AC 运用中：

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

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

3.4.1 最低工作电压

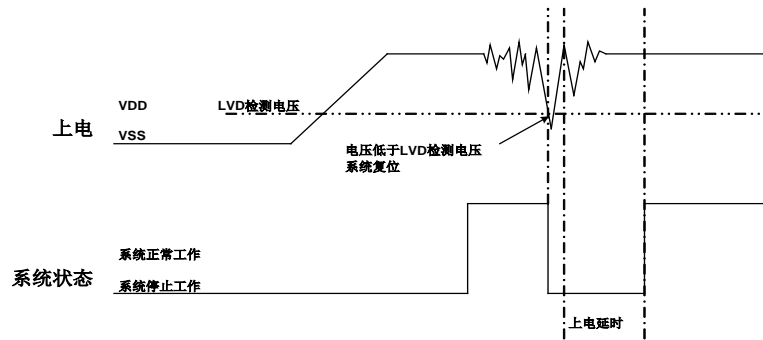
为了改善系统掉电复位的性能，首先必须明确系统具有的最低工作电压值。系统最低工作电压与系统执行速度有关，不同的执行速度下最低工作电压值也不同。



系统工作电压与执行速度关系图

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

3.4.2 低电压检测 (LVD)



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

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

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

Bit 5 **LVD36**: LVD 3.6V 工作电压标志, LVD 编译选项为 LVD_H 时有效。

- 0 = 无效 ($VDD > 3.6V$);
- 1 = 有效 ($VDD \leq 3.6V$)。

Bit 4 **LVD24**: LVD 2.4V 工作电压标志, LVD 编译选项为 LVD_M 时有效。

- 0 = 无效 ($VDD > 2.4V$);
- 1 = 有效 ($VDD \leq 2.4V$)。

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

LVD_L

如果 $VDD < 2.0V$, 系统复位;

LVD24 和 LVD36 标志位无意义。

LVD_M

如果 $VDD < 2.0V$, 系统复位;

LVD24: 如果 $VDD > 2.4V$, LVD24 = 0; 如果 $VDD \leq 2.4V$, LVD24 = 1;

LVD36 标志位无意义。

LVD_H

如果 $VDD < 2.4V$, 系统复位;

LVD36: 如果 $VDD > 3.6V$, LVD36 = 0; 如果 $VDD \leq 3.6V$, LVD36 = 1;

LVD24 标志位无意义。

LVD_MAX

如果 $VDD < 3.6V$ 系统复位。

*** 注:**

a) LVD 复位结束后, LVD24 和 LVD36 都将被清零;

b) LVD 2.4V 和 LVD3.6V 检测电平值仅作为设计参考, 不能用作芯片工作电压值的精确检测。

3.4.3 掉电复位性能改进

如何改善系统掉电复位性能，有以下几点建议：

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

* 注：

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

2、系统时钟为 4MHZ/4（1Mips）并使用外部复位电路（“稳压二极管复位电路”、“电压偏移复位电路”、“外部 IC 复位”）可提高 AC 供电系统的 EFT 特性。这种设计可提高系统的抗干扰性能并得到良好的 EFT 特性。

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

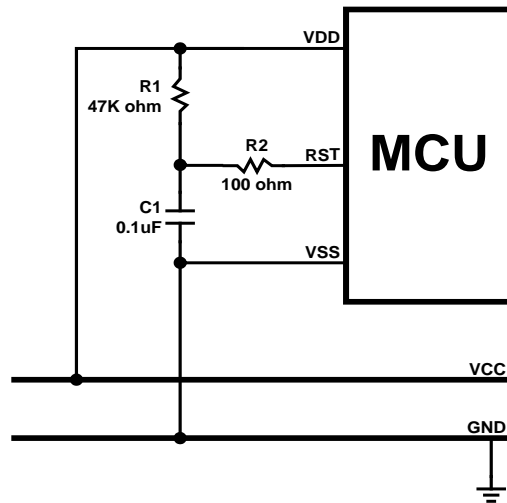
外部复位功能由编译选项“Reset_Pin”控制。将该编译选项置为“Reset”，可使能外部复位功能。外部复位引脚为施密特触发结构，低电平有效。复位引脚处于高电平时，系统正常运行。当复位引脚输入低电平信号时，系统复位。外部复位操作在上电和正常工作模式时有效。需要注意的是，在系统上电完成后，外部复位引脚必须输入高电平，否则系统将一直保持在复位状态。外部复位的时序如下：

- **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：**所有的系统寄存器被置为初始状态；
- **振荡器开始工作：**振荡器开始提供系统时钟；
- **执行程序：**上电结束，程序开始运行。

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

3.6 外部复位电路

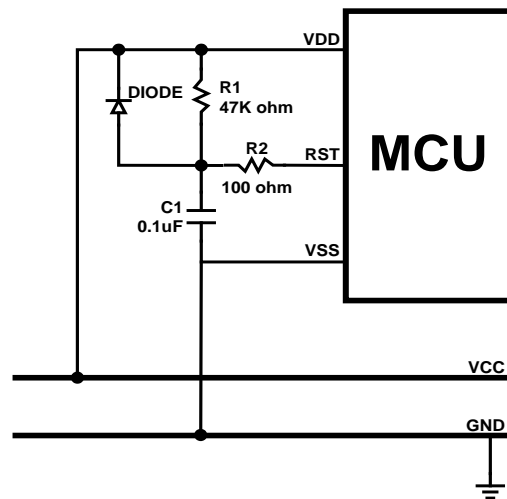
3.6.1 基本RC复位电路



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

* 注：此 RC 复位电路不能解决非正常上电和掉电复位问题。

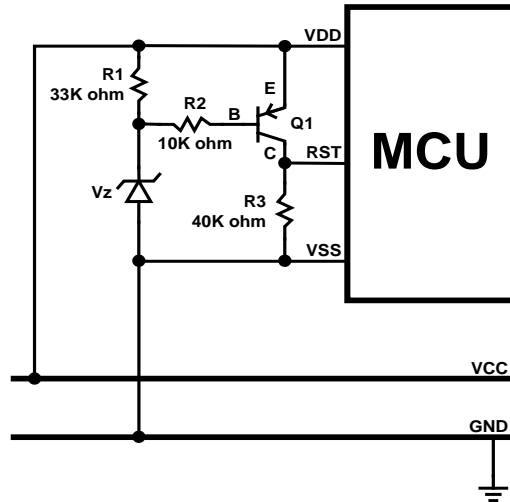
3.6.2 二极管 & RC 复位电路



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

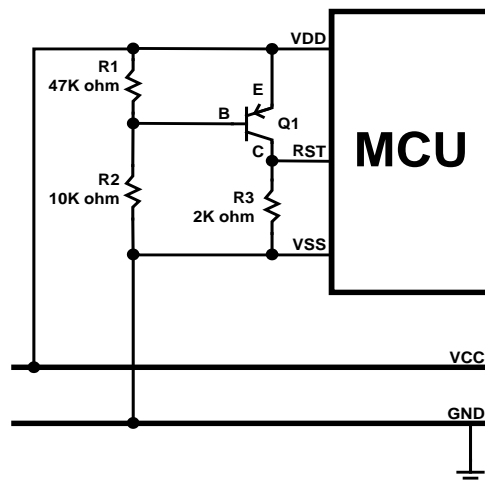
* 注：“基本 RC 复位电路”和“二极管及 RC 复位电路”中的电阻 R2 都是必不可少的限流电阻，以避免复位引脚 ESD (Electrostatic Discharge) 或 EOS (Electrical Over-stress) 击穿。

3.6.3 齐纳二极管复位电路



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 电压偏移复位电路

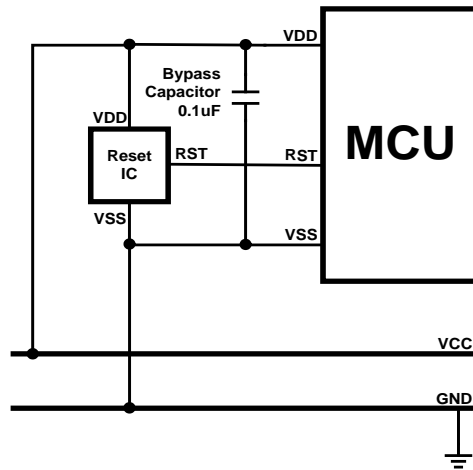


电压偏置复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，这种复位电路的检测电压值的精确度有所降低。电路中，R1 和 R2 构成分压电路，当 VDD 高于和等于分压值“ $0.7V \times (R1 + R2) / R1$ ”时，三极管集电极 C 输出高电平，单片机正常工作；VDD 低于“ $0.7V \times (R1 + R2) / R1$ ”时，集电极 C 输出低电平，单片机复位。

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复位电路



外部复位也可以选用 IC 进行外部复位，但是这样一来系统成本将会增加。针对不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

SN8P2743/42 内置双时钟系统：高速时钟和低速时钟。高速时钟包括内部高速振荡器和外部高速振荡器，由编译选项 High_CLK 选择。低速时钟由内部低速振荡器提供，由 OSCM 寄存器的 CLKMD 位控制，高、低速时钟都可以作为系统时钟源。

- **高速振荡器**

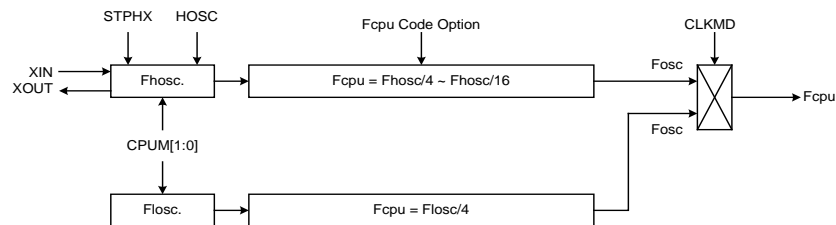
内部高速振荡器：RC，高达 16MHz，称为 IHRC；

外部高速振荡器：晶体/陶瓷（4MHz/12MHz/32KHz）和 RC。

- **低速振荡器**

内部低速振荡器：RC，16KHz@3V，32KHz@5V，称为 ILRC。

- **系统时钟框图**



- HOSC: High_Clk 编译选项。
- Fosc: 外部高速时钟频率/内部高速时钟频率。
- Fosc: 内部低速 RC 时钟频率（16KHz@3V，32KHz@5V）。
- Fosc: 系统时钟频率。
- Fcpu: 指令执行频率。

4.2 Fcpu（指令周期）

系统时钟速率，即指令周期（Fcpu），从系统时钟源分离出来，决定系统的工作速率。Fcpu 的速率由 Fcpu 编译选项决定，正常模式下， $F_{cpu} = F_{osc}/4 \sim F_{osc}/16$ 。当系统高速时钟源由外部 4MHz 晶振提供时，Fcpu 编译选项选择 $F_{osc}/4$ ，则 Fcpu 频率为 $4\text{MHz}/4 = 1\text{MHz}$ 。低速模式下， $F_{cpu} = F_{osc}/4$ ，即 $16\text{KHz}/4 = 4\text{KHz}@3\text{V}$ ， $32\text{KHz}/4 = 8\text{KHz}@5\text{V}$ 。

4.3 系统高速时钟

系统高速时钟包括外部高速时钟和内部高速时钟。外部高速时钟又包括 4MHz、12MHz、32KHz 晶体/陶瓷和 RC 振荡器，高速时钟振荡器由编译选项 High_CLK 选择。

4.3.1 HIGH_CLK编译选项

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

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

4.3.2 内部高速RC振荡器（IHRC）

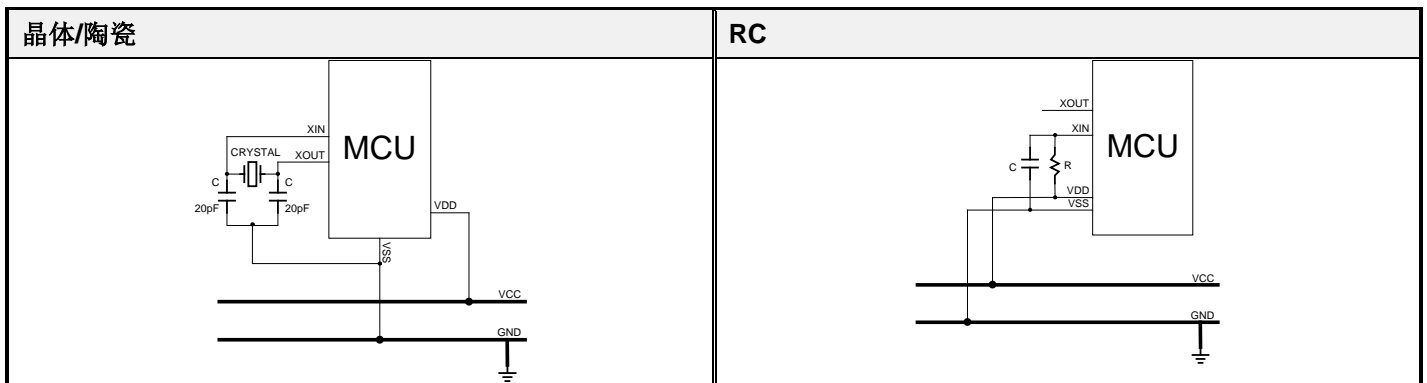
内部高速 16MHz RC 振荡器，普通环境下精确度为 $\pm 2\%$ ，当选择 IHRC_16M 时，使能内部高速振荡器。

- **IHRC_16M**: 系统高速时钟为内部 16MHz RC 振荡器，XIN/XOUT 为普通 I/O 引脚。

4.3.3 外部高速振荡器

外部高速振荡器包括 4MHz，12MHz，32KHz 和 RC。4M、12M 和 32K 可以使用晶体和陶瓷振荡器，XIN/XOUT 和 GND 之间需连接一个 20pF 的电容。廉价的 RC 振荡电路只需要和 XIN 引脚连接，电容的容值不能低于 100pF，电阻的阻值由频率决定。

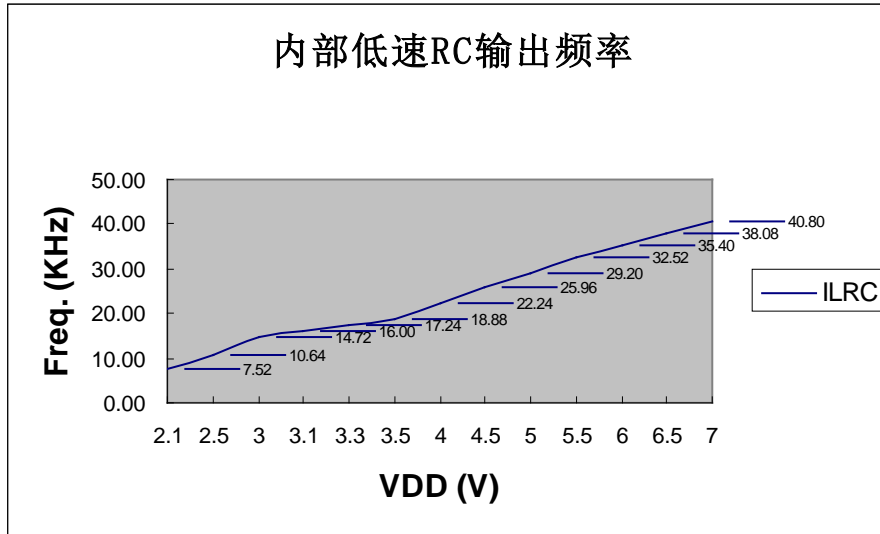
4.3.4 外部振荡器应用电路



* 注：晶体/陶瓷和电容 C 要尽可能的靠近单片机的 XIN/XOUT/VSS；电阻 R 和电容 C 要尽可能的靠近单片机的 VDD。

4.4 系统低速时钟

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



低速时钟可作为看门狗定时器的时钟源。由 CLKMD 控制系统低速工作模式。

- $F_{osc} =$ 内部低速 RC 振荡器 (16KHz @3V, 32KHz @5V)。
- 低速模式 $F_{cpu} = F_{osc} / 4$ 。

有两种情况下内部低速振荡器停止工作，一种是进入睡眠模式，另一种为选外部 32K X'tal 进入绿色模式并关掉看门狗。如果系统选外部 32K X'tal 进入绿色模式并关掉看门狗，此时只有外部 32K X'tal 有效并且系统处于省电状态。

- 例：在睡眠模式下，停止内部低速振荡器。
B0BSET FCPUM0

* 注：不可以单独停止内部低速时钟；内部低速的状态由寄存器 OSCM 的 CPUM0 和 CPUM1 (32K X'tal、关闭看门狗) 位决定。

4.5 OSCM寄存器

寄存器 OSCM 控制振荡器的状态和系统的工作模式。

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

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

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

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

STPHX 位为内部高速 RC 振荡器和外部振荡器的模式控制位。当 STPHX=0，外部振荡器和内部高速 RC 振荡器正常运行；当 STPHX=1，外部振荡器或内部高速 RC 振荡器停止运行。不同的高速时钟选项决定不同的 STPHX 功能。

- **IHRC_16M:** STPHX=1，禁止内部高速 RC 振荡器；
- **RC, 4M, 12M, 32K:** STPHX=1，禁止外部振荡器。

4.6 系统时钟测试

在设计过程中，用户可通过软件指令周期对系统时钟速度进行测试。

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

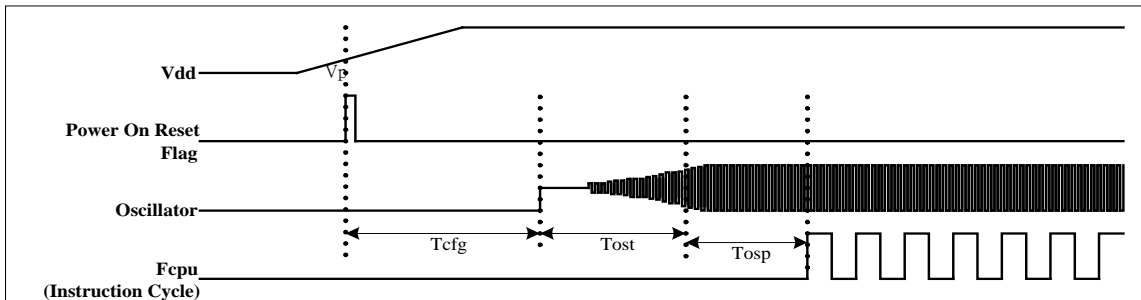
@@:
B0BSET P0.0
B0BCLR P0.0
JMP @B

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

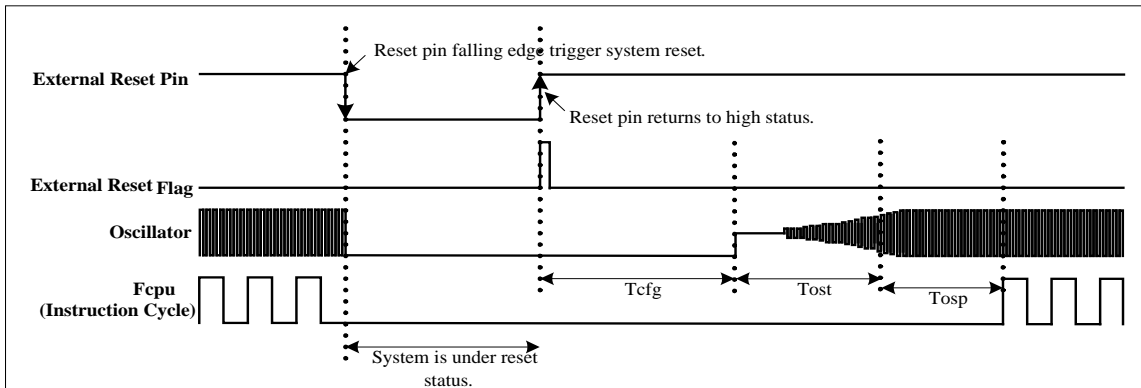
4.7 系统时钟时序

参数	符号	说明	典型值
硬件配置时间	Tcfg	$2048 * F_{ILRC}$	64ms @ $F_{ILRC} = 32\text{KHz}$ 128ms @ $F_{ILRC} = 16\text{KHz}$
振荡器启动时间	Tost	启动时间取决于振荡器的材料、工艺等。通常情况下，低速振荡器的启动时间要比高速振荡器的启动时间长，RC 振荡器的启动时间要比晶体/陶瓷振荡器的启动时间短。	-
振荡器起振时间	Tosp	复位情况下的振荡器起振时间为 $2048 * F_{hosc}$ （使能上电复位，LVD 复位，看门狗复位，外部复位引脚）	64ms @ $F_{hosc} = 32\text{KHz}$ 512us @ $F_{hosc} = 4\text{MHz}$ 128us @ $F_{hosc} = 16\text{MHz}$
		睡眠模式唤醒情况的振荡器起振时间为： $2048 * F_{hosc}$晶体/陶瓷振荡器，如 32768Hz晶振，4MHz晶振，16MHz晶振等； $32 * F_{hosc}$RC振荡器，如外部RC振荡电路，内部高速RC振荡器。	X'tal: 64ms @ $F_{hosc} = 32\text{KHz}$ 512us @ $F_{hosc} = 4\text{MHz}$ 128us @ $F_{hosc} = 16\text{MHz}$ RC: 8us @ $F_{hosc} = 4\text{MHz}$ 2us @ $F_{hosc} = 16\text{MHz}$

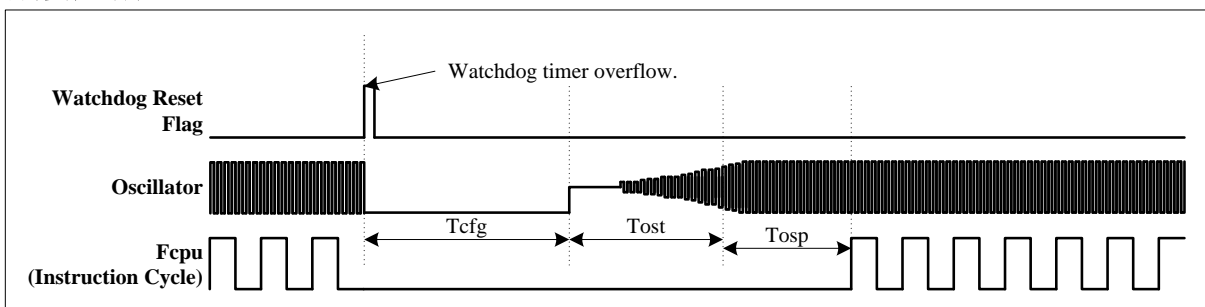
- 上电复位时序



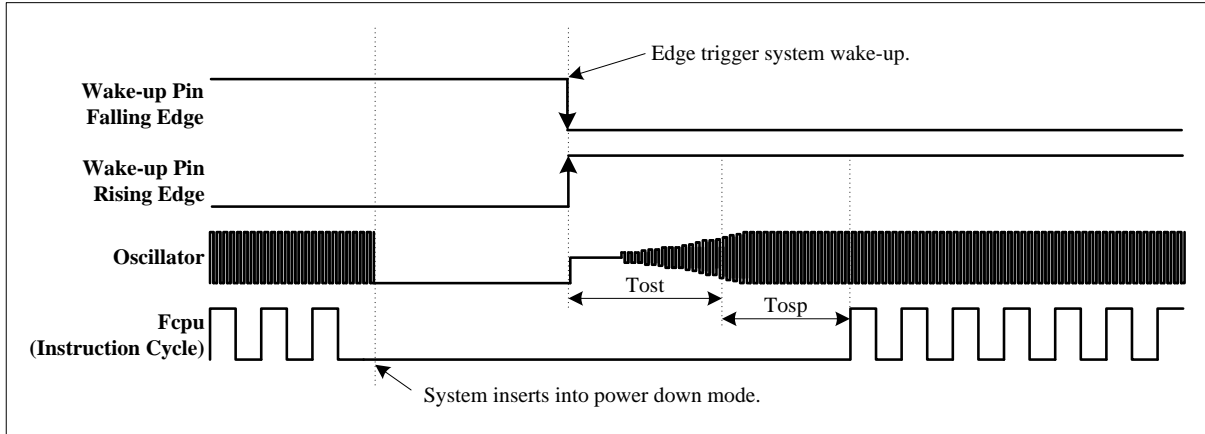
- 外部复位引脚复位时序



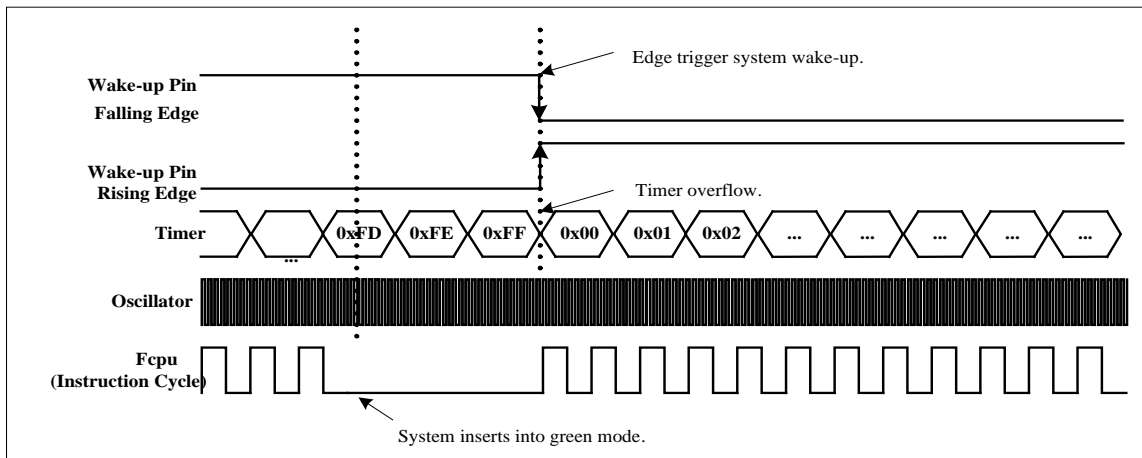
- 看门狗复位时序



● 睡眠模式唤醒时序

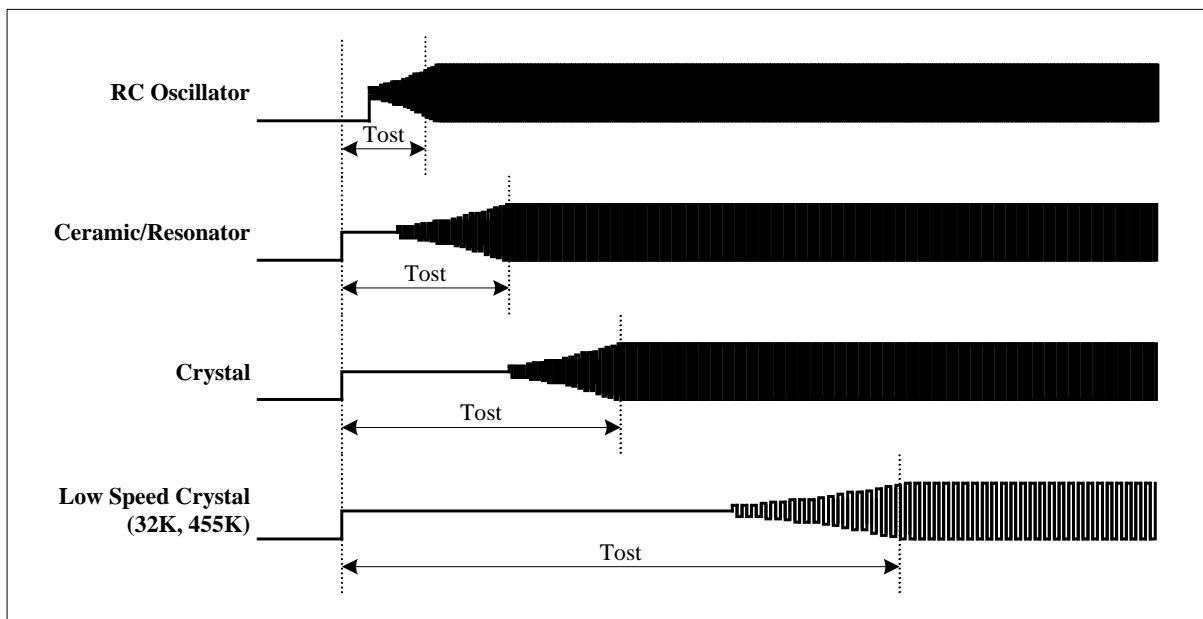


● 绿色模式唤醒时序



● 振荡器启动时间

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



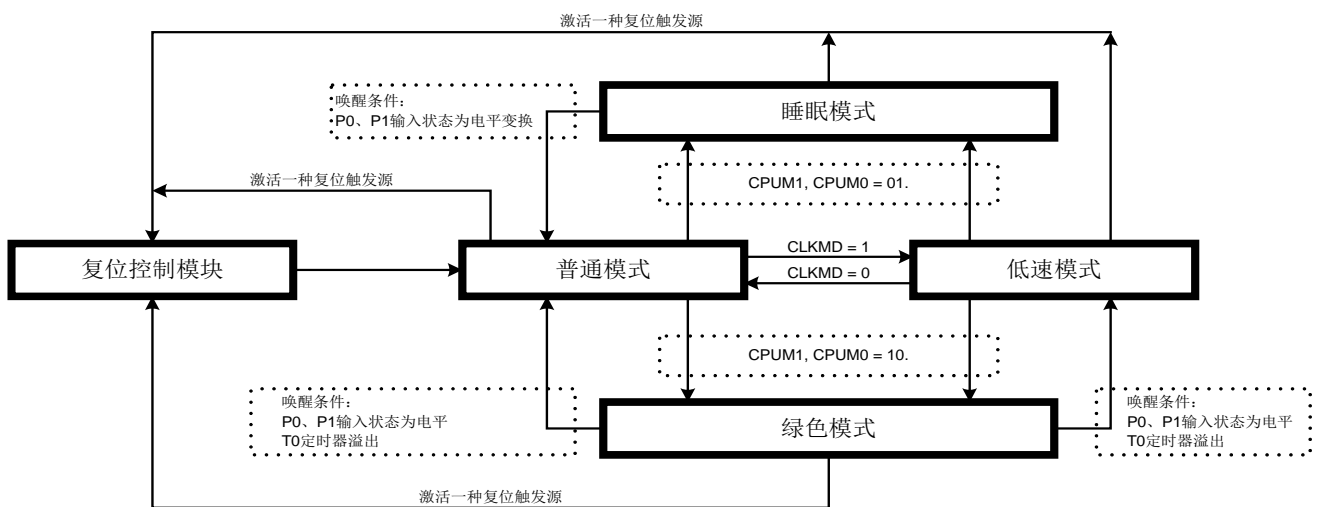
5 系统工作模式

5.1 概述

SN8P2743/42 可以在 4 种工作模式下以不同的时钟频率工作，这些模式可以控制振荡器的工作、程序的执行以及外围模拟电路的操作。

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

工作模式控制框图



工作模式时钟控制表

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

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

5.2 普通模式

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

- 程序被执行，所有的功能都可控制。
- 系统速率为高速。
- 高速振荡器和内部低速 RC 振荡器都正常工作。
- 可以通过 OSCM 寄存器从普通模式切换进入任何模式。
- 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。
- 系统速率为低速 ($F_{osc}/4$)。
- 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 1 μ A。睡眠模式可以由 P0、P1 的电平变换触发唤醒。P1 的唤醒功能由 P1W 寄存器控制。从任何工作模式进入睡眠模式，被唤醒后都返回到普通模式。由 OSCM 寄存器的 CPUM0 位控制是否进入睡眠模式，当 CPUM0=1，系统进入睡眠模式。当系统从睡眠模式被唤醒后，CPUM0 被自动禁止 (0 状态)。

- 程序停止执行，所有的功能被禁止。
- 所有的振荡器，包括外部高速振荡器、内部高速振荡器和内部低速振荡器都停止工作。
- 功耗低于 1 μ A。
- 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，可以确保系统进入睡眠模式，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 为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。
- 绿色模式下 PWM 功能仍然有效，但是定时器溢出时不能唤醒系统。

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

5.6 工作模式控制宏

Sonix 提供工作模式控制宏以方便系统工作模式的切换。

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

- 例：从普通模式/低速模式切换进入睡眠模式。

```
SleepMode ; 直接宣告“SleepMode”宏。
```

- 例：从普通模式切换进入低速模式。

```
SlowMode ; 直接宣告“SlowMode”宏。
```

- 例：从低速模式切换进入普通模式（外部高速振荡器停止工作）。

```
Slow2Normal ; 直接宣告“Slow2Normal”宏。
```

- 例：从普通/低速模式切换进入绿色模式。

```
GreenMode ; 直接宣告“GreenMode”宏。
```

- 例：从普通/低速模式切换进入绿色模式，并使能 T0 唤醒功能。

; 设置定时器 T0 的唤醒功能。

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

; 进入绿色模式。

```
GreenMode ; 直接宣告“GreenMode”宏。
```

5.7 系统唤醒

5.7.1 概述

睡眠模式和绿色模式下，系统并不执行程序。唤醒触发信号可以将系统唤醒进入普通模式或低速模式。唤醒触发信号包括：外部触发信号（P0、P1 的电平变换）和内部触发（T0 定时器溢出）。唤醒功能内置中断请求标志位，触发后执行中断服务程序。

- 从睡眠模式唤醒后只能进入普通模式，且将其唤醒的触发只能是外部触发信号（P0、P1 电平变化）；
- 如果是将系统由绿色模式唤醒返回到上一个工作模式（普通模式或低速模式），则可以是外部触发信号（P0、P1 电平变换）和内部触发信号（T0 溢出）。
- 系统从睡眠模式或绿色模式下被唤醒时，Wakeup 中断功能给出 WAKEIRQ 中断请求。若 WAKEIEN 为 1，触发中断，执行中断服务程序。

* 注：如果唤醒源为外部中断，WAKE 位不能置 1，外部中断请求标志位被置 1。系统给出外部中断请求并执行中断服务程序。

5.7.2 唤醒时间

系统进入睡眠模式后，高速时钟振荡器停止运行。把系统从睡眠模式唤醒时，单片机需要等待 2048 个外部高速时钟周期和 32 个内部高速时钟周期的时间，以等待振荡电路稳定工作，等待的这一段就称为唤醒时间。唤醒时间结束后，系统进入普通模式。

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

唤醒时间的计算如下：

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

➤ 例：睡眠模式下，系统被唤醒进入普通模式。唤醒时间的计算如下：

$$\begin{aligned} \text{唤醒时间} &= 1/F_{\text{osc}} * 2048 = 0.512 \text{ ms} \quad (F_{\text{osc}} = 4\text{MHz}) \\ \text{总的唤醒时间} &= 0.512\text{ms} + \text{振荡器启动时间} \end{aligned}$$

内部高速 RC 时钟的唤醒时间的计算如下：

$$\text{唤醒时间} = 1/F_{\text{osc}} * 32 \text{ (sec)} + \text{高速时钟启动时间}$$

➤ 例：睡眠模式下，系统被唤醒进入普通模式。唤醒时间的计算如下：

$$\text{唤醒时间} = 1/F_{\text{osc}} * 32 = 2 \text{ ms} \quad (F_{\text{osc}} = 16\text{MHz})$$

* 注：高速时钟的启动时间决定于 VDD 和振荡器的类型。

5.7.3 P1W唤醒控制寄存器

在绿色模式和睡眠模式下，有唤醒功能的 I/O 口能够将系统唤醒到普通模式。P0 和 P1 都具有唤醒功能，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
P1W	-	P16W	P15W	P14W	P13W	P12W	P11W	P10W
读/写	-	W	W	W	W	W	W	W
复位后	-	0	0	0	0	0	0	0

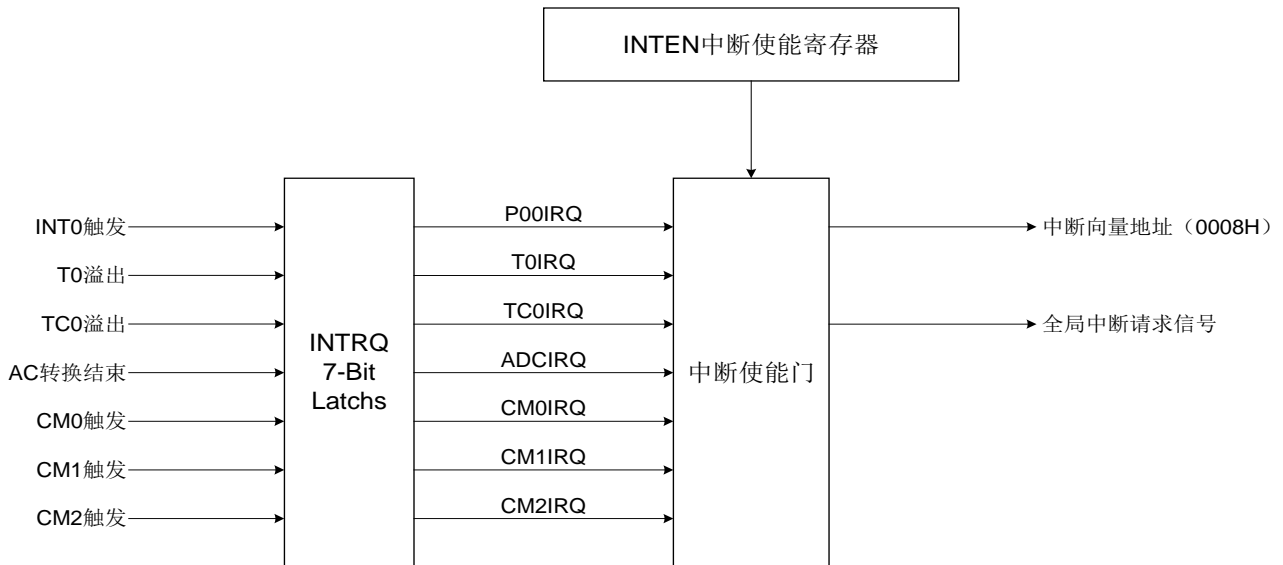
Bit[6:0] P10W~P16W: P1 唤醒功能控制位。

- 0 = 禁止；
- 1 = 使能。

6 中断

6.1 概述

SN8P2743/42 提供 7 种中断源：6 个内部中断（T0/TC0/CM0/CM1/CM2/ADC）和 1 个外部中断（INT01）。外部中断可以将系统从睡眠模式唤醒进入高速模式，在返回高速模式前，外部中断请求被锁定。一旦程序进入中断，寄存器 STKP 的位 GIE 将被硬件自动清零以避免再次响应其它中断。系统退出中断，即执行完 RETI 指令后，硬件自动将 GIE 置“1”，以响应下一个中断。中断请求存放在寄存器 INTRQ 中。



* 注：程序响应中断时，位 GIE 必须处于有效状态。

6.2 INTEN中断使能寄存器

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

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	ADCIEN	-	TC0IEN	TOIEN	CM2IEN	CM1IEN	CM0IEN	P00IEN
读/写	R/W	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	-	0	0	0	0	0	0

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

0 = 禁止;
1 = 使能。

Bit 1 **CM0IEN**: CM0 中断控制位。

0 = 禁止;
1 = 使能。

Bit 2 **CM1IEN**: CM1 中断控制位。

0 = 禁止;
1 = 使能。

Bit 3 **CM2IEN**: CM2 中断控制位。

0 = 禁止;
1 = 使能。

Bit 4 **TOIEN**: T0 中断控制位。

0 = 禁止;
1 = 使能。

Bit 5 **TC0IEN**: TC0 中断控制位。

0 = 禁止;
1 = 使能。

Bit 7 **ADCIEN**: ADC 中断控制位。

0 = 禁止;
1 = 使能。

6.3 INTRQ中断请求寄存器

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

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	ADCIRQ	-	TC0IRQ	T0IRQ	CM2IRQ	CM1IRQ	CM0IRQ	P00IRQ
读/写	R/W	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	-	0	0	0	0	0	0

Bit 0 **P00IRQ**: P0.0 中断 (INT0) 请求标志位。

0 = INT0 无中断请求;
1 = INT0 有中断请求。

Bit 1 **CM0IRQ**: CM0 请求标志位。

0 = CM0 无中断请求;
1 = CM0 有中断请求。

Bit 2 **CM1IRQ**: CM1 请求标志位。

0 = CM1 无中断请求;
1 = CM1 有中断请求。

Bit 3 **CM2IRQ**: CM2 请求标志位。

0 = CM2 无中断请求;
1 = CM2 有中断请求。

Bit 4 **T0IRQ**: T0 中断请求标志位。

0 = T0 无中断请求;
1 = T0 有中断请求。

Bit 5 **TC0IRQ**: TC0 中断请求标志位。

0 = TC0 无中断请求;
1 = TC0 有中断请求。

Bit 7 **ADCIRQ**: ADC 中断请求标志位。

0 = ADC 无中断请求;
1 = ADC 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。

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

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

➤ 例：设置全局中断控制位（GIE）。
B0BSET FGIE ; 使能 GIE。

* 注：在所有中断中，GIE 都必须处于使能状态。

6.5 PUSH, POP

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

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

➤ 例：用 PUSH、POP 指令来保护和恢复 ACC 和 PFLAG。

```
.CODE
    ORG     0
    JMP     START

    ORG     8H
    JMP     INT_SERVICE

START:
    ...

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

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


6.6 外部中断INT0

SN8P2743/42 只有 1 个外部中断 INT0。当外部中断被触发时，若使能外部中断使能控制位，则外部中断请求标志位被置 1。若使能外部中断使能控制位且外部中断请求位置 1 时，程序计数器跳转到中断向量地址 0008H 开始执行中断服务。若禁止外部中断使能控制位，则外部中断请求标志位无效，则不会执行中断服务程序。

外部中断内置唤醒锁存功能，就是指系统从睡眠模式唤醒，唤醒源位中断源（P0.0）。触发沿的方向与中断沿的配置相互配合。当触发沿被锁存后，系统被唤醒后先执行中断服务程序。

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

Bit[1:0] **P00G[1:0]**: INT0 中断触发控制位。

- 00 = 保留；
- 01 = 上升沿；
- 10 = 下降沿；
- 11 = 上升/下降沿（电平变换触发）。

➤ 例：INT0 中断请求设置，电平触发。

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

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

➤ 例：INT0 中断。

```
ORG      8H           ;
JMP      INT_SERVICE

INT_SERVICE:

...           ; ACC 和 PFLAG 入栈保护。

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

B0BCLR   FP00IRQ      ; P00IRQ 清零。
...      ; INT1 中断服务程序。
...

EXIT_INT:

...           ; ACC 和 PFLAG 出栈恢复。

RETI     ; 退出中断。
```

6.7 T0 中断

T0C 溢出时，无论 T0IEN 处于何种状态，T0IRQ 都会置“1”。若 T0IEN 和 T0IRQ 都置“1”，系统就会响应 T0 的中断；若 T0IEN = 0，则无论 T0IRQ 是否置“1”，系统都不会响应 T0 中断。尤其需要注意多种中断下的情形。

例：设置 T0 中断。

```

B0BCLR    FT0IEN    ; 禁止 T0 中断。
B0BCLR    FT0ENB    ; 关闭 T0。
MOV       A, #20H   ;
B0MOV     T0M, A    ; 设置 T0 时钟= Fcpu / 64。
MOV       A, #64H   ; 初始化 T0C = 64H。
B0MOV     T0C, A    ; 设置 T0 间隔时间= 10 ms。

B0BCLR    FT0IRQ    ; T0IRQ 清零。
B0BSET    FT0IEN    ; 使能 T0 中断。
B0BSET    FT0ENB    ; 开启定时器 T0。

B0BSET    FGIE      ; 使能 GIE。

```

例：T0 中断服务程序。

```

ORG       8H
JMP       INT_SERVICE
INT_SERVICE:

...      ; 保存 ACC 和 PFLAG。

B0BTS1   FT0IRQ    ; 检查是否有 T0 中断请求标志。
JMP      EXIT_INT  ; T0IRQ = 0，退出中断。

B0BCLR   FT0IRQ    ; 清 T0IRQ。
MOV      A, #64H
B0MOV    T0C, A    ;
...
EXIT_INT:
...      ; 恢复 ACC 和 PFLAG。

RETI     ; 退出中断。

```

6.8 TC0 中断

TC0C 溢出时，无论 TC0IEN 处于何种状态，TC0IRQ 都会置“1”。若 TC0IEN 和 TC0IRQ 都置“1”，系统就会响应 TC0 的中断；若 TC0IEN = 0，则无论 TC0IRQ 是否置“1”，系统都不会响应 TC0 中断。尤其需要注意多种中断下的情形。

➤ 例：TC0 中断请求设置。

```

B0BCLR    FTC0IEN    ; 禁止 TC0 中断。
B0BCLR    FTC0ENB    ;
MOV       A, #20H    ;
B0MOV     TC0M, A    ; TC0 时钟 = Fcpu / 64。
MOV       A, # 64H   ; TC0C 初始值 = 64H。
B0MOV     TC0C, A    ; TC0 间隔 = 10 ms。

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

B0BSET    FGIE       ; 使能 GIE。

```

➤ 例：TC0 中断服务程序。

```

ORG       8H          ;
INT_SERVICE:
JMP      INT_SERVICE

...          ; 保存 ACC 和 PFLAG。

B0BTS1   FTC0IRQ    ; 检查是否有 TC0 中断请求标志。
JMP      EXIT_INT    ; TC0IRQ = 0，退出中断。

B0BCLR   FTC0IRQ    ; 清 TC0IRQ。
MOV      A, #64H     ;
B0MOV    TC0C, A    ; 清 TC0C。
...      ; TC0 中断程序。
...

EXIT_INT:
...      ; 恢复 ACC 和 PFLAG。

RETI     ; 退出中断。

```

6.9 ADC中断

当ADC转换完成后，无论ADCIEN是否使能，ADCIQR都会置“1”。若ADCIEN和ADCIQR都置“1”，那么系统就会响应ADC中断。若ADCIEN = 0，不管ADCIRQ是否置“1”，系统都不会进入ADC中断。用户应注意多种中断下的处理。

➤ 例：ADC中断设置。

```

B0BCLR      FADCIEN      ; 禁止 ADC 中断。

MOV         A, #10110000B ;
B0MOV      ADM, A        ; 允许 P4.0 ADC 输入，使能 ADC 功能。
MOV         A, #00000000B ; 设置 AD 转换速率 = Fcpu/16。
B0MOV      ADR, A

B0BCLR      FADCIRQ     ; 清除 ADC 中断请求标志。
B0BSET      FADCIEN     ; 使能 ADC 中断。
B0BSET      FGIE        ; 使能 GIE。

B0BSET      FADS        ; 开始 AD 转换。

```

➤ 例：ADC中断服务程序。

```

ORG         8H          ; 中断向量地址。
INT_SERVICE:
JMP         INT_SERVICE

...          ; 保存 ACC 和 PFLAG。

B0BTS1     FADCIRQ     ; 检查是否有 ADC 中断。
JMP        EXIT_INT    ; ADCIRQ = 0，退出中断。

B0BCLR      FADCIRQ     ; 清 ADCIRQ。
...          ; ADC 中断服务程序。
...

EXIT_INT:
...          ; 恢复 ACC 和 PFLAG。

RETI       ; 退出中断。

```

6.10 比较器中断 (CMP0~CMP2)

SN8P2743/42 共有 3 个比较器中断 (CMP0~CMP2)。寄存器 CM0M 的 CM0G[1:0] 控制中断 CMP0 的中断触发方向，寄存器 CM1M 的 CM1G[1:0] 控制中断 CMP1 的中断触发方向，寄存器 CM2M 的 CM2G[1:0] 控制中断 CMP2 的中断触发方向。当比较器输出状态开始转变时，不管比较器中断使能控制位的状态，比较器振荡器的请求控制位都会置 1。当使能比较器中断使能控制位且比较器中断请求控制位置 1 时，程序会跳转到中断向量 0008 处开始执行中断服务程序。当禁止比较器中断使能控制位时，比较器中断请求控制位会处于无效状态，此时就不执行中断服务。

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM0M	CM0EN	CM0OEN	CM0OUT	CM0SF	CM0G	-	-	-
读/写	R/W	R/W	R/W	R/W	R/W	-	-	-
复位后	0	0	0	0	0	-	-	-

Bit 3 **CM0G**: 比较器 CM0 中断触发方向控制位。

- 0 = 下降沿触发，比较器输出状态由高变低 (CM0P < CM0N)；
- 1 = 上升沿触发，比较器输出状态由低变高 (CM0P > CM0N)。

09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM1M	CM1EN	CM1OEN	CM1OUT	CM1SF	CM1G	CM1RS2	CM1RS1	CM1RS0
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 3 **CM1G**: 比较器 CM1 中断触发方向控制位。

- 0 = 下降沿触发，比较器输出状态由高变低 (CM1P < CM1N)；
- 1 = 上升沿触发，比较器输出状态由低变高 (CM1P > CM1N)。

09EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM2M	CM2EN	CM2OEN	CM2OUT	CM2SF	CM2G	CM2RS2	CM2RS1	CM2RS0
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 3 **CM2G**: 比较器 CM2 中断触发方向控制位。

- 0 = 下降沿触发，比较器输出状态由高变低 (CM2P < CM2N)；
- 1 = 上升沿触发，比较器输出状态由低变高 (CM2P > CM2N)。

➤ 例：设置 CMP0 中断。

```
MOV      A, #00H
BOBMOV  CM0M, A      ; 设置 CMP0 中断为电平变换触发。

BOBSET  FCM0IEN      ; 使能 CMP0 中断。
BOBCLR  FCM0IRQ      ; 清 CMP0 中断请求。
BOBSET  FCM0EN
BOBSET  FGIE         ; 使能 GIE。
```

➤ 例：CMP0 中断服务程序。

```
ORG      8           ; 中断向量。
JMP      INT_SERVICE

INT_SERVICE:
...      ; 保存 ACC 和 PFLAG。

BOBTS1  FCM0IRQ      ; 检查 CM0IRQ 状态。
JMP      EXIT_INT    ; 若 CMP0IRQ = 0，退出中断。

BOBCLR  FCM0IRQ      ; 复位 CM0IRQ。
...      ; CMP0 中断服务程序。

EXIT_INT:
...      ; 恢复 ACC 和 PFLAG。
RETI     ; 退出中断。
```

6.11 多中断操作

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

中断	有效触发
P00IRQ	由 PEDGE 控制
T0IRQ	T0C 溢出
TC0IRQ	TC0C 溢出
ADCIRQ	AD 转换结束
CM0IRQ	CM0 输出电平转变
CM1IRQ	CM1 输出电平转变
CM2IRQ	CM2 输出电平转变

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

➤ 例：多中断条件下检测中断请求。

```

ORG          8H          ;
JMP          INT_SERVICE

INT_SERVICE:
...           ; 保存 ACC 和 PFLAG。
INTP00CHK:   ...           ; 检查是否有 INTO 中断请求。
             B0BTS1      FP00IEN      ; 检查是否使能 INTO 中断。
             JMP         INTT0CHK      ; 跳到下一个中断。
             B0BTS0      FP00IRQ      ; 检查是否有 INTO 中断请求。
             JMP         INTP00        ; 进入 INTO 中断。
INTT0CHK:    ...           ; 检查是否有 T0 中断请求。
             B0BTS1      FT0IEN       ; 检查是否使能 T0 中断。
             JMP         INTTC0CHK     ; 跳到下一个中断。
             B0BTS0      FT0IRQ       ; 检查是否有 T0 中断请求。
             JMP         INTT0        ; 进入 T0 中断。
INTTC0CHK:   ...           ; 检查是否有 TC0 中断请求。
             B0BTS1      FTC0IEN      ; 检查是否使能 TC0 中断。
             JMP         INTADCCHK    ; 跳到下一个中断。
             B0BTS0      FTC0IRQ      ; 检查是否有 TC0 中断请求。
             JMP         INTTC0       ; 进入 TC0 中断。
INTADCCHK:   ...           ; 检查是否有 ADC 中断请求。
             B0BTS1      FADC IEN     ; 检查是否使能 ADC 中断。
             JMP         INT_EXIT      ;
             B0BTS0      FADC IRQ     ; 检查是否有 ADC 中断请求。
             JMP         INTADC       ; 进入 ADC 中断。
INT_EXIT:    ...           ; 恢复 ACC 和 PFLAG。
             RETI                    ; 退出中断。

```

7 I/O口

7.1 概述

SN8P2743/42 共有 22 个 I/O 引脚，大多数 I/O 引脚与模拟引脚和特殊功能的引脚共用，详见下表：

I/O 引脚		共用引脚		共用引脚控制条件
名称	类型	名称	类型	
P0.0	I/O	INT0	DC	P00IEN=1
P0.1	O	PWM0	DC	PWM0OUT=1.
P0.2	I/O	CM0P	AC	CM0EN=1
P0.3	I/O	CM0N	AC	CM0EN=1
P0.4	I	RST	DC	Reset_Pin code option = Reset
		VPP	HV	OTP Programming
P0.5	I/O	XOUT	AC	High_CLK code option = 32K, 4M, 12M
		BZ	DC	BZEN=1
P0.6	I/O	XIN	AC	High_CLK code option = RC, 32K, 4M, 12M
P1.0	I/O	OPN	AC	OPEN=1
P1.1	I/O	OPP	AC	OPEN=1
P1.2	I/O	OPO	AC	OPEN=1
P1.3	I/O	CM2N	AC	CM2EN=1
P1.4	I/O	CM2P	AC	CM2EN=1, CM2RS[2:0]=000b
P1.5	I/O	CM1N	AC	CM1EN=1
P1.6	I/O	CM1P	AC	CM1EN=1, CM1RS[2:0]=000b
P4.0	I/O	AIN0	AC	ADENB=1, GCHS=1, CHS[2:0] = 000b
		AVREFH	AC	ADENB=1, AVREFH=1
P4[7:1]	I/O	AIN[7:1]	AC	ADENB=1, GCHS=1, CHS[2:0] = 001b~111b

* DC：数字特性；AC：模拟特性；HV：高压特性。

7.2 I/O口模式

寄存器 PnM 控制 I/O 口的工作模式。

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

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1M	-	P16M	P15M	P14M	P13M	P12M	P11M	P10M
读/写	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	0	0	0	0	0	0	0

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

Bit[7:0] **PnM[7:0]**: Pn 模式控制位 (n = 0~4)。

0 = 输入模式;

1 = 输出模式。

- * 注: 用户可通过位操作指令 (B0BSET、B0BCLR) 对 I/O 口进行编程控制;
- * 注: P0.4 是单向输入引脚, P0M.4 未定义。

➤ 例: I/O 模式选择。

```

CLR          P0M          ; 设置为输入模式。
CLR          P4M

MOV          A, #0FFH     ; 设置为输出模式。
B0MOV       P0M, A
B0MOV       P4M, A

B0BCLR      P4M.0        ; P4.0 设为输入模式。

B0BSET      P4M.0        ; P4.0 设为输出模式。

```


7.3 I/O口上拉电阻

输入模式下内置上拉电阻。PnUR 寄存器可以控制上拉电阻，当 PnUR 为 0 时，禁止上拉电阻，为 1 时使能上拉电阻。

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	-	P06R	P05R	-	P03R	P02R	-	P00R
读/写	-	W	W	-	W	W	-	W
复位后	-	0	0	-	0	0	-	0

0E1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	-	P16R	P15R	P14R	P14R	P12R	P11R	P10R
读/写	-	W	W	W	W	W	W	W
复位后	-	0	0	0	0	0	0	0

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

* 注：P0.4 为单向输入引脚，无上拉电阻，故 P0UR.4 未定义。

➤ 例：I/O 口的上拉电阻。

```
MOV          A, #0FFH          ; 使能 P0、P4 的上拉电阻。
B0MOV       P0UR, A
B0MOV       P4UR, A
```

7.4 I/O口数据寄存器

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

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

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

1、当通过编译选项使能外部复位时，P04 保持为 1。

2、若需要设置 P0 寄存器的某位 (P0.n) 时，建议使用“MOV”或者“B0MOV”指令，而不要使用“read & modify write”型的指令（如，BSET、BCLR、B0BSET、B0BCLR...），否则执行这些指令后，P0 寄存器的只写位 P0.1 会被更改。

➤ 例：读取输入口的数据。

```
B0MOV    A, P0           ; 读取 P0、P4 口的数据。
B0MOV    A, P4
```

➤ 例：写入数据到输出端口。

```
MOV      A, #0FFH       ; 写入数据 FFH 到 P0、P4。
B0MOV    P0, A
B0MOV    P4, A
```

➤ 例：写入 1 位数据到输出端口。

```
B0BSET   P4.0           ; P4.0 置 1。
```

```
B0BCLR   P4.0           ; P4.0 清 0。
```

7.5 P4 与ADC共用引脚

P4 口和 ADC 的输入口共用，无施密特触发。同一时间只能设置 P4 口的一个引脚作为 ADC 的测量信号输入口（通过 ADM 寄存器来设置），其它引脚则作为普通 I/O 使用。具体应用中，当输入一个模拟信号到 CMOS 结构端口，尤其当模拟信号为 $1/2 V_{DD}$ 时，将可能产生额外的漏电流。同样，当 P4 口外接多个模拟信号时，也会产生额外的漏电流。在睡眠模式下，上述漏电流会严重影响到系统的整体功耗。P4CON 为 P4 口的配置寄存器。将 P4CON[7:0]置“1”，其对应的 P4 口将被设置为纯模拟信号输入口，从而避免上述漏电流的情况。

0AEH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON	P4CON7	P4CON6	P4CON5	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **P4CON[7:0]**: P4.n 控制位。

0 = P4.n 作为模拟信号输入引脚（ADC 输入引脚）或普通 I/O 引脚；

1 = P4.n 只能作为模拟信号输入引脚，不能作为普通 I/O 引脚。

* 注：当 P4.n 作为普通 I/O 口而不是 ADC 输入引脚时，P4CON.n 必须置为 0，否则 P4.n 的普通 I/O 信号会被隔离开来。

P4 的 ADC 模拟输入由寄存器 ADM 的 GCHS 和 CHSn 位控制，若 GCHS = 0，P4.n 为普通的 I/O 引脚，若 GCHS = 1，CHSn 所对应的 P4.n 用作 ADC 模拟信号输入引脚。

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	AVREFH	CHS2	CHS1	CHS0
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 4 **GCHS**: ADC 输入通道控制位。

0 = 禁止 AIN 通道；

1 = 开启 AIN 通道。

Bit 3 **AVREFH**: ADC 外部参考电压高电平输入引脚控制位。

0 = ADC 参考电压由内部 VDD 提供，P4.0 作为 GPIO 或 AIN0 引脚；

1 = 使能 ADC 外部参考电压高电平由 P4.0 输入。

Bit[2:0] **CHS[2:0]**: ADC 输入通道选择位。

000 = AIN0; 001 = AIN1; ...; 110 = AIN6; 111 = AIN7。

* 注：在设置 P4.n 为普通的 I/O 引脚时，必须保证 P4.n 的 ADC 功能已经被禁止。否则当 GCHS=1 时，会将 CHS[2:0]对应的 P4.n 自动设为 ADC 模拟输入引脚。

➤ 例：设置 P4.1 为普通的输入引脚，P4CON.1 必须置为 0。

; 检查 GCHS 和 CHS[2:0]的状态。

B0BCLR

FGCHS

; 若 CHS[2:0]指向 P4.1 (CHS[2:0] = 001B), GCHS=0。

; 若 CHS[2:0]没有指向 P4.1, 则忽略 GCHS 的状态。

; 清 P4CON。

B0BCLR

P4CON.1

; 使能 P4.1 的普通 I/O 功能。

; P4.1 设为输入模式。

B0BCLR

P4M.1

; 设置 P4.1 为输入模式。

➤ 例：设置 P4.1 为普通的输出模式，P4CON.1 必须置为 0。

; 检查 GCHS 和 CHS[2:0]的状态。

B0BCLR	FGCHS	; 若 CHS[2:0]指向 P4.1 (CHS[2:0] = 001B), GCHS=0。
		; 若 CHS[2:0]没有指向 P4.1, 则忽略 GCHS 的状态。

; 清 P4CON。

B0BCLR	P4CON.1	; 使能 P4.1 的普通 I/O 功能。
--------	---------	-----------------------

; 设置 P4.1 为输出模式以避免误操作。

B0BSET	P4.1	; 设置 P4.1 为 1。
--------	------	----------------

; 或

B0BCLR	P4.1	; 设置 P4.1 为 0。
--------	------	----------------

; P4.1 设为输出模式。

B0BSET	P4M.1	; 设置 P4.1 为输出模式。
--------	-------	------------------

P4.0 与 ADC 输入引脚 (AIN0) 和 ADC 外部参考电压高电平输入引脚共用。ADM 寄存器的 AVREFH 位为 ADC 参考电压高电平输入的控制位。若使能 AVREFH, 禁止 P4.0 的普通 I/O 功能和 ADC 模拟输入功能 (AIN0), P4.0 直接连接 ADC 参考电压高电平输入。

* 注：若需使能 P4.0 的普通 I/O 功能和 AIN0 功能，则 AVREFH 必须置 0。

➤ 例：设置 P4.0 为普通输入引脚，AVREFH 和 P4CON.0 必须置 0。

; 检查 AVREFH 的状态。

B0BTS0	FAVREFH	;
B0BCLR	FAVREFH	; AVREFH = 1, 清 AVREFH, 禁止外部 ADC 参考电压高电平输入。
		; AVREFH = 0, 跳到下一条程序。

; 检查 GCHS 和 CHS[2:0]的状态。

B0BCLR	FGCHS	; 若 CHS[2:0]指向 P4.0 (CHS[2:0]=000B), GCHS=0.
		; 若 CHS[2:0]没有指向 P4.0, 则忽略 GCHS 的状态。

; 清 P4CON。

B0BCLR	P4CON.0	; 使能 P4.0 的普通 I/O 功能。
--------	---------	-----------------------

; 设置 P4.0 为输入模式。

B0BCLR	P4M.0	
--------	-------	--

➤ 例：设置 P4.0 为普通输出引脚，EVREFH 和 P4CON.0 必须置 0。

; 检查 AVREFH 的状态。

B0BTS0	FAVREFH	;
B0BCLR	FAVREFH	; AVREFH = 1, 清 AVREFH, 禁止外部 ADC 参考电压高电平输入。
		; AVREFH = 0, 跳到下一条程序。

; 检查 GCHS 和 CHS[2:0]的状态。

B0BCLR	FGCHS	; 若 CHS[2:0]指向 P4.0 (CHS[2:0]=000B), GCHS=0.
		; 若 CHS[2:0]没有指向 P4.0, 则忽略 GCHS 的状态。

; 清 P4CON。

B0BCLR	P4CON.0	; 使能 P4.0 的普通 I/O 功能。
--------	---------	-----------------------

; 设置 P4.0 为输出模式以避免误操作。

B0BSET	P4.0	; 设置 P4.0 为 1。
--------	------	----------------

; 或

B0BCLR	P4.0	; 设置 P4.0 为 0。
--------	------	----------------

; P4.0 设为输出模式。

B0BSET	P4M.0	
--------	-------	--

8 定时器

8.1 看门狗定时器

看门狗定时器 WDT 是一个 4 位二进制计数器，用于监控程序的正常执行。如果由于干扰，程序进入了未知状态，看门狗定时器溢出，系统复位。看门狗的工作模式由编译选项控制，其时钟源由内部低速 RC 振荡器提供，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	16KHz	512ms
5V	32KHz	256ms

看门狗定时器的 3 种工作模式由编译选项“WatchDog”控制：

- **Disable:** 禁止看门狗定时器功能。
- **Enable:** 使能看门狗定时器功能，在普通模式和低速模式下有效，在睡眠模式和绿色模式下看门狗停止工作。
- **Always_On:** 使能看门狗定时器功能，在睡眠模式和绿色模式下，看门狗仍会正常工作。
在高干扰环境下，强烈建议将看门狗设置为“Always_On”以确保系统在出错状态和重启时正常复位。

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

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，否则无法检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。

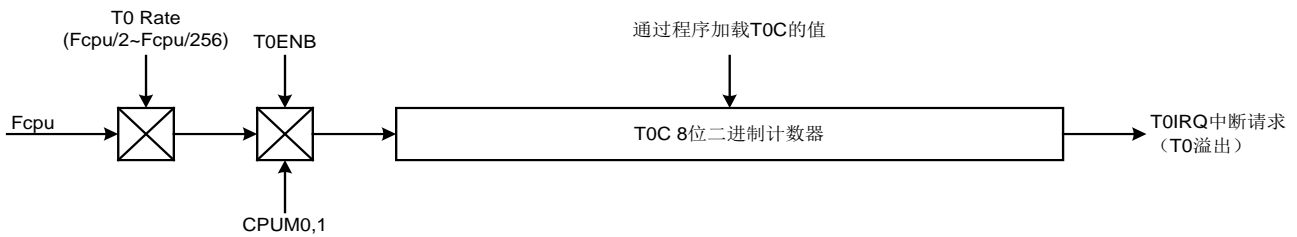
Correct:
    ...                ; I/O 口和 RAM 都正确，清看门狗定时器。
    ...                ;
    MOV                A, #5AH
    B0MOV             WDTR, A
    ...
    CALL              SUB1
    CALL              SUB2
    ...
    JMP               Main
  
```

8.2 8 位基本定时器T0

8.2.1 概述

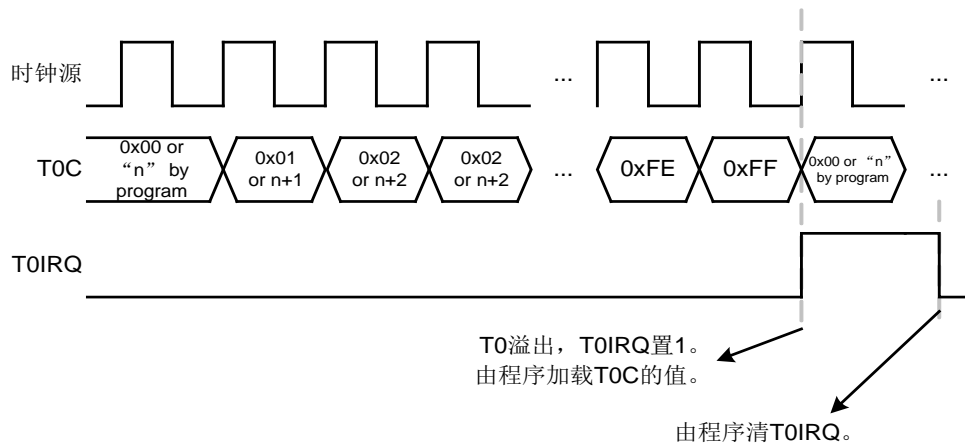
8 位二进制基本定时器 T0 具有定时器功能置：支持中断请求标志的显示（TOIRQ）和中断操作（中断向量）。可以通过 TOM 和 TOC 寄存器控制间隔时间，具有在绿色模式下唤醒功能。在绿色模式下，T0 溢出，则将系统唤醒返回到上一个工作模式。

- ☞ **8 位可编程计数定时器：**根据选择的时钟频率周期性的产生中断请求。
- ☞ **中断功能：**T0 定时器支持中断功能，当 T0 溢出，TOIRQ 有效，程序计数器跳到中断向量地址执行中断。
- ☞ **绿色模式唤醒功能：**T0 定时器在绿色模式下正常工作，溢出时将系统从绿色模式下唤醒。



8.2.2 T0 定时器操作

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



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

T0rate[2:0]	T0 时钟	T0 间隔时间	
		max. (ms)	Unit (us)
000b	Fcpu/256	8.192	32
001b	Fcpu/128	4.096	16
010b	Fcpu/64	2.048	8
011b	Fcpu/32	1.024	4
100b	Fcpu/16	0.576	2.25
101b	Fcpu/8	0.256	1
110b	Fcpu/4	0.128	0.5
111b	Fcpu/2	0.064	0.25

8.2.3 TOM模式寄存器

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

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

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;

1 = 使能。

8.2.4 T0C计数寄存器

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

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

T0C 初始值的计算公式如下：

$$\text{T0C 初始值} = 256 - (\text{T0 中断间隔时间} * \text{T0 时钟 Rate})$$

➤ 例：计算 T0C，T0 的间隔时间为 10ms，T0 时钟源 Fcpu = 4MHz/4=1MHz，TORATE = 001 (Fcpu/128)。

T0 间隔时间=10ms，T0 时钟 Rate=4MHz/4/128

$$\begin{aligned} \text{T0C 初始值} &= 256 - (\text{T0 中断间隔时间} * \text{输入时钟}) \\ &= 256 - (10\text{ms} * 4\text{MHz} / 4 / 128) \\ &= 256 - (10^{-2} * 4 * 10^6 / 4 / 128) \\ &= \text{B2H} \end{aligned}$$

8.2.5 T0 操作举例

● T0 定时器设置

; 复位 T0 定时器。

```
CLR          TOM          ; 清 TOM。
```

; 设置 T0 时钟源和 T0Rate。

```
MOV          A, #0nnn0000b
BO MOV      TOM, A
```

; 设置 T0 中断间隔时间。

```
MOV          A, #value
BO MOV      T0C, A
```

; 清 T0IRQ。

```
BO CLR      FT0IRQ
```

; 使能 T0 定时器和中断功能。

```
BO SET      FT0IEN      ; 使能 T0 中断功能。
BO SET      FT0ENB      ; 使能 T0 定时器。
```

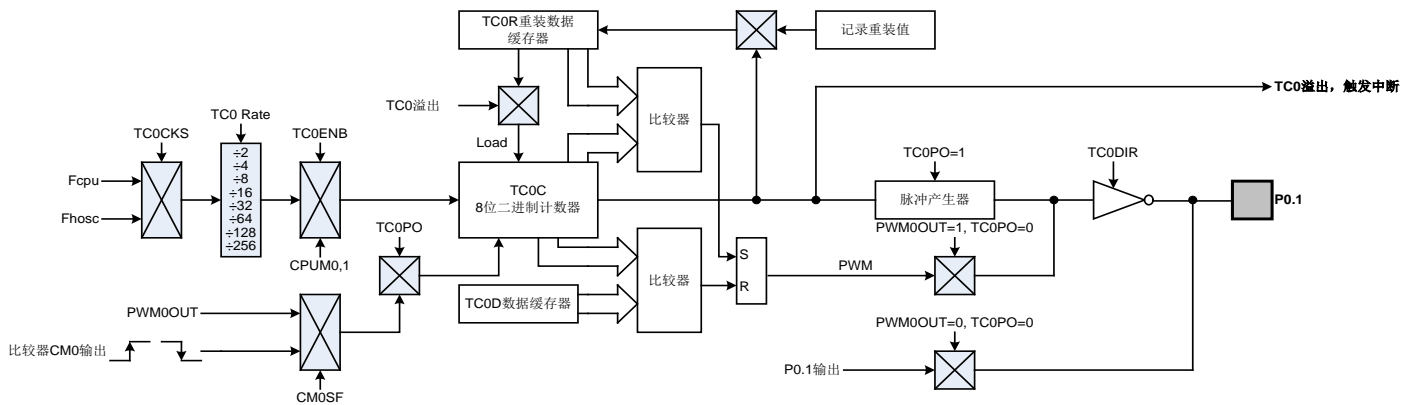

8.3 8 位定时/计数器TC0

8.3.1 概述

8 位二进制定时/计数器具有基本定时器、PWM 和脉冲产生器功能。基本定时器功能可以支持中断请求标志的显示 (TC0IRQ) 和中断操作 (中断向量)。由 TC0M、TC0C、TC0R 寄存器控制 TC0 的中断间隔时间。TC0 内置 PWM 功能，其周期/分辨率由 TC0 定时器时钟频率和 TC0R、TC0D 寄存器控制，故具有良好性能的 PWM 可以处理 IR 载波信号、马达控制和光度调节等。TC0 还内置一个脉冲产生器，脉冲发生器是以一个 PWM 周期开启脉冲触发功能。脉冲触发源由 TC0PO 控制位和比较器触发沿寄存器控制。TC0 计数器支持自动重装功能 (始终使能)，TC0 溢出时，TC0R 的值自动装入 TC0C。TC0 在绿色模式下不能将系统唤醒。

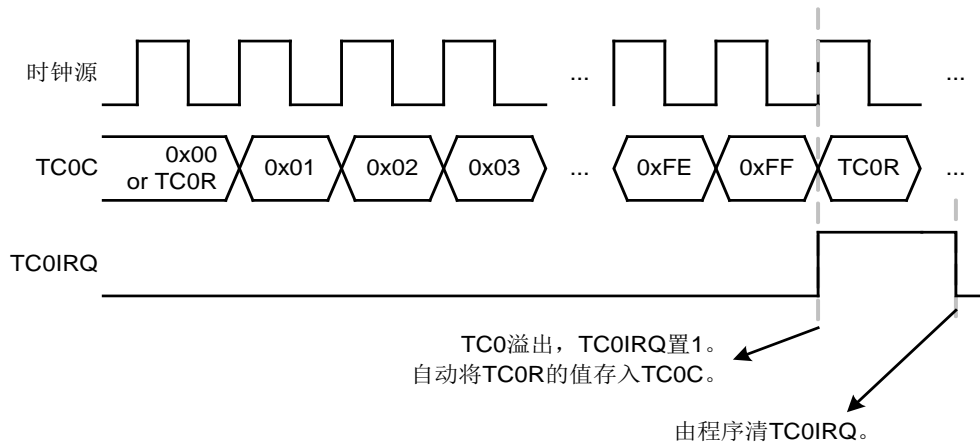
TC0 的主要用途如下：

- ☞ **8 位可编程定时器：**根据选择的时钟信号，产生周期中断。
- ☞ **中断功能：**TC0 定时器支持中断，当 TC0 溢出时，TC0IRQ 置 1，系统执行中断。
- ☞ **PWM (占空比/周期可编程控制)：**由 TC0R 和 TC0D 寄存器控制 PWM 的占空比/周期。
- ☞ **脉冲产生器：**脉冲发生器是以一个 PWM 周期开启脉冲触发功能。脉冲触发源由 TC0PO 控制位和比较器触发沿寄存器控制 TC0PO=1, CM0SF=0 时，PWM0OUT 位触发脉冲产生器；TC0PO=1, CM0SF=1 时，比较器 CM0 输出沿触发脉冲产生器。
- ☞ **绿色模式功能：**绿色模式下，TC0 正常工作，但无唤醒功能。当请求中断 (如定时器溢出...) 时，中断标志位置 1。



8.3.2 TC0 定时器操作

TC0 定时器由 TC0ENB 控制。当 TC0ENB=0 时，TC0 停止工作；当 TC0ENB=1 时，TC0 开始计数。使能 TC0 之前，先要设定好 TC0 的功能模式，如基本定时器、TC0 中断等。TC0C 溢出（从 0FFH 到 00H）时，TC0IRQ 置 1 以显示溢出状态并由程序清零。在不同的功能模式下，TC0C 不同的值对应不同的操作，若改变 TC0C 的值影响到操作，会导致功能出错。TC0 内置双重缓存器以避免此种状况的发生。在 TC0C 计数的过程中不断的刷新 TC0C，保证将最新的值存入 TC0R（重装缓存器）中，当 TC0 溢出后，TC0R 的值自动存入 TC0C。进入下一个周期后，TC0 进行新的工作状态。定时/计数器模式时，自动重装功能没有控制位，使能 TC0 时有效。如果使能 TC0 中断功能（TC0IEN=1），在 TC0 溢出时系统执行中断服务程序，在中断时必须由程序清 TC0IRQ。TC0 可以在普通模式、低速模式和绿色模式下工作。但在绿色模式下，TC0 虽继续工作，但不能唤醒系统。

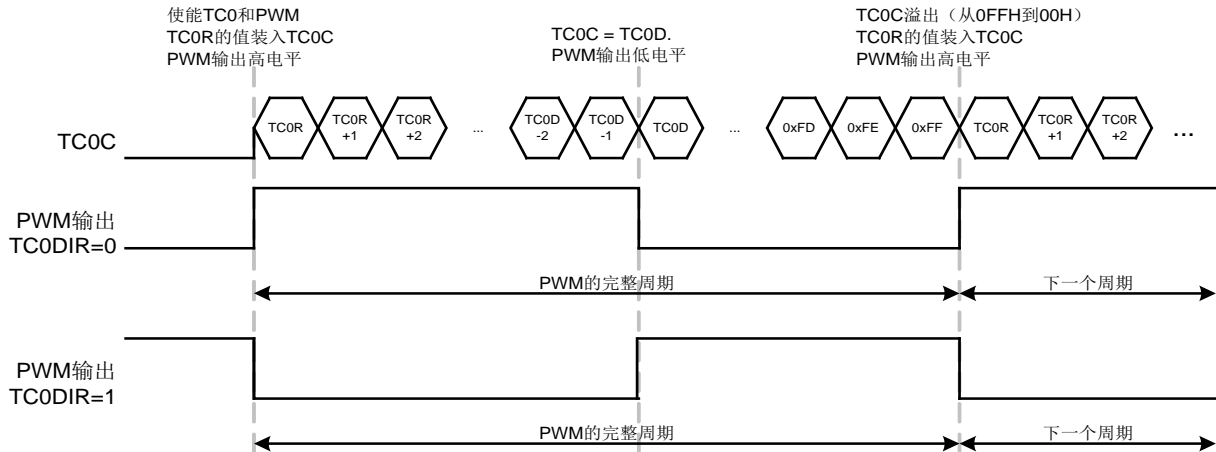


TC0 根据不同的时钟源选择不同的应用模式，TC0 的时钟源为 F_{cpu} （指令周期）和 F_{hosc} （高速振荡器），由 TC0CKS 控制。当 TC0CKS=0 时，TC0 时钟源来自 F_{cpu} ，可以由 TC0Rate[2:0] 选择不同的分频。当 TC0CKS=1 时，TC0 时钟源来自 F_{hosc} ，可以由 TC0Rate[2:0] 选择不同的分频。

TC0CKS	TC0rate[2:0]	TC0 时钟	TC0 间隔时间			
			Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4	
			max. (ms)	Unit (us)	max. (ms)	Unit (us)
0	000b	Fcpu/256	16.384	64	65.536	256
0	001b	Fcpu/128	8.192	32	32.768	128
0	010b	Fcpu/64	4.096	16	16.384	64
0	011b	Fcpu/32	2.048	8	8.192	32
0	100b	Fcpu/16	1.024	4	4.096	16
0	101b	Fcpu/8	0.512	2	2.048	8
0	110b	Fcpu/4	0.256	1	1.024	4
0	111b	Fcpu/2	0.128	0.5	0.512	2
1	000b	Fhosc/256	4.096	16	16.384	64
1	001b	Fhosc/128	2.048	8	8.192	32
1	010b	Fhosc/64	1.024	4	4.096	16
1	011b	Fhosc/32	0.512	2	2.048	8
1	100b	Fhosc/16	0.256	1	1.024	4
1	101b	Fhosc/8	0.128	0.5	0.512	2
1	110b	Fhosc/4	0.064	0.25	0.256	1
1	111b	Fhosc/2	0.032	0.125	0.128	0.5

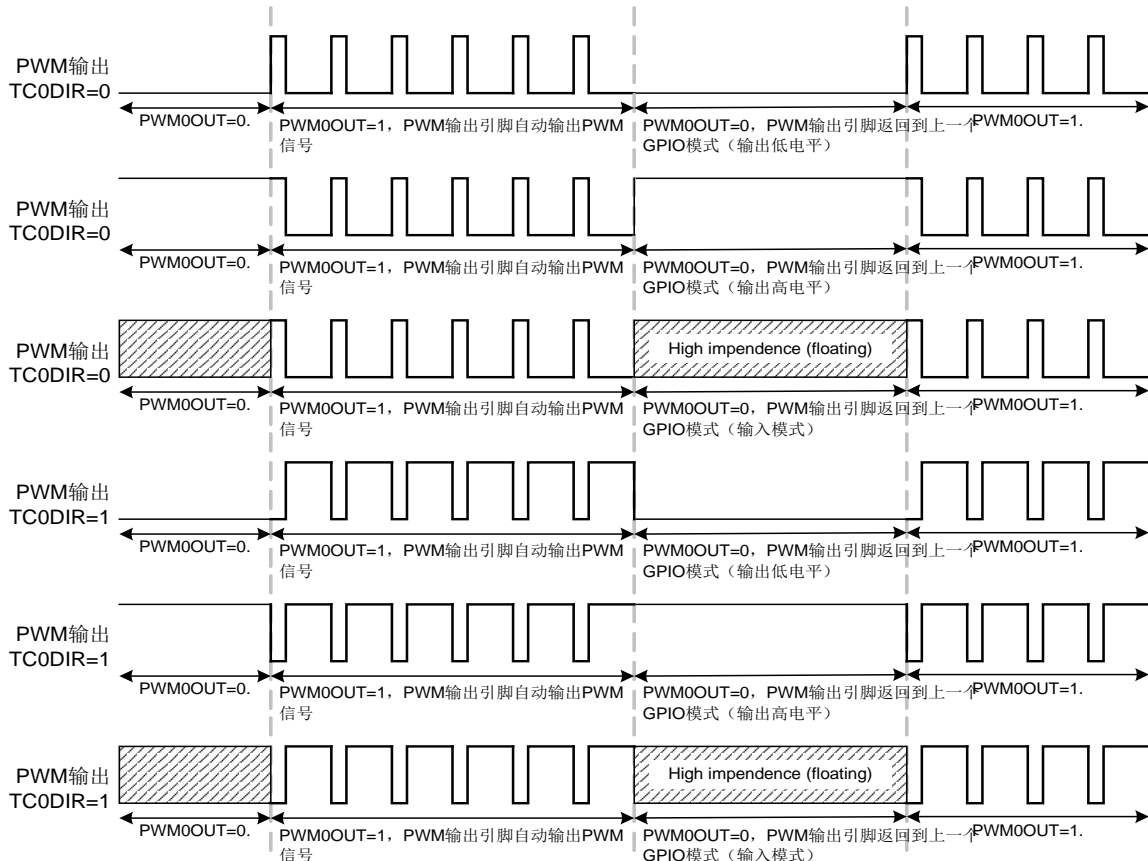
8.3.3 脉冲宽度调制 (PWM)

TC0 内置 PWM 功能，由 PWM0OUT 位控制，PWM 输出引脚与 GPIO 引脚共用。PWM0OUT=1 时，使能 PWM 功能，PWM 输出引脚切换到 PWM 输出状态；PWM0OUT=0 时，PWM 输出引脚返回到上一个 GPIO 状态。PWM 信号由 TC0C、TC0R 和 TC0D 产生。PWM0OUT=1，或者 TC0C 溢出（从 0FFH 到 00H）时，PWM 输出高电平（PWM 初始状态），加载最新的 TC0R 的值到 TC0C，以决定 PWM 周期和分辨率。TC0C 继续计数，系统比较 TC0C 和 TC0D 的值，TC0C=TC0D 时，PWM 输出低电平，TC0C 继续计数直到 TC0 溢出，完成一个 PWM 信号周期。TC0R 的值自动装入 TC0C，在下一个周期开始时，PWM 输出高电平。TC0D 决定高电平的持续时间，TC0R 决定 PWM 的分辨率和周期，TC0R 不能大于 TC0D，否则 PWM 信号出错。PWM 的相位可以通过 TC0DIR 位控制，TC0DIR=0 时，PWM 的相位高电平脉冲，低电平脉冲空闲；TC0DIR=1 时，PWM 的相位是低电平脉冲，高电平脉冲空闲。



PWM 的分辨率由 TC0R 决定，TC0R 的范围值为 00H~0FFH。TC0R=00H 时，PWM 的分辨率为 1/256，TC0R=80H 时，PWM 的分辨率为 1/128。TC0D 控制 PWM 脉冲高电平的宽度。TC0C=TC0D 时，PWM 输出低电平，TC0D 必须大于 TC0R，否则 PWM 一直输出低电平。PWM 模式下，TC0 溢出时，TC0IRQ 有效，且 TC0IEN=1 时，使能 TC0 中断。但强烈建议小心同时使用 PWM 和 TC0 定时器，以保证两种功能都能正常工作。

PWM 输出引脚与 GPIO 引脚共用，由 PWM0OUT 位控制。PWM0OUT=1 时自动输入 PWM 信号；若 PWM0OUT 清零禁止 PWM 时，PWM 输出引脚自动返回到上一个 GPIO 模式。这样易于执行载波信号 on/off 操作，而不控制 TC0ENB 位。



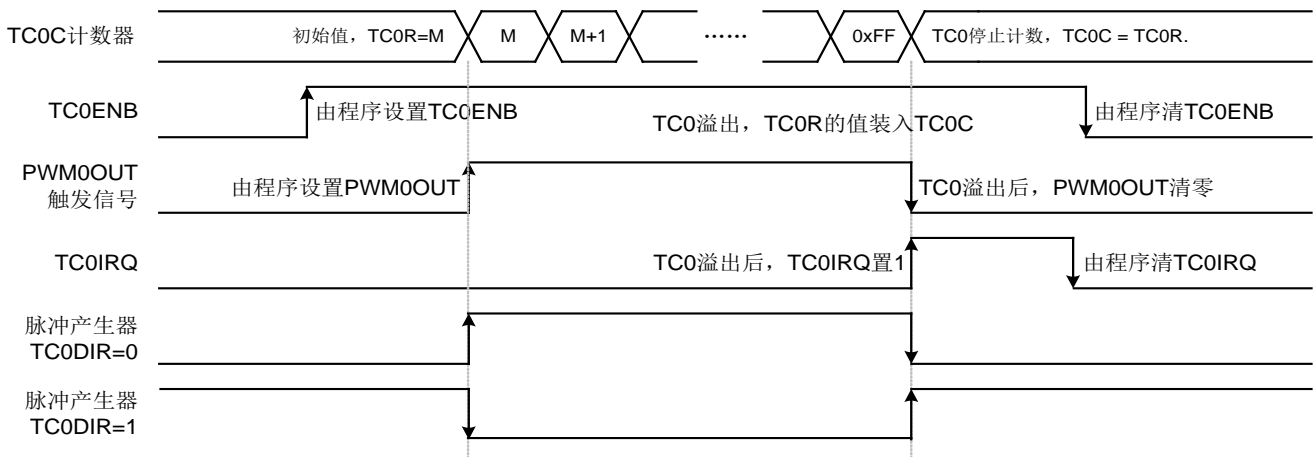
8.3.4 TC0 脉冲产生器功能

TC0 定时器内置脉冲产生器功能。脉冲产生器输出一个脉冲，由 TC0 定时器的间隔时间决定该脉冲的宽度。脉冲产生器由 TC0PO 位控制，TC0PO=0 时，TC0 为普通定时器模式或 PWM 功能模式；TC0PO=1 时，TC0 为脉冲产生器模式。脉冲产生器需要一个起始触发信号以控制 TC0 计数器和脉冲信号输出。TC0PO=1 时，TC0 计数器停止计数，TC0C/TC0R 寄存器的值由程序设置，TC0C 的值决定脉冲的宽度。触发事件发生，TC0 开始计数，TC0DIR 控制脉冲输出引脚的输出状态。TC0 溢出后，脉冲信号完成输出并切换到空闲状态，TC0C 停止计数，并将 TC0R 最新的值装入 TC0C。脉冲产生器模式下，TC0 溢出后，TC0IRQ 有效。在脉冲产生器工作期间，通过更改 TC0R（而不是 TC0C）的值来改变脉冲的宽度，否则脉冲宽度出错。

脉冲输出控制信号包括两种触发源，由 CMOSF 控制：一种是 PWM0OUT (CMOSF=0)，另一种是比较器 CM0 输出沿 (CMOSF=1)。若触发源为 PWM0OUT，由程序设置 PWM0OUT 位输出脉冲信号，TC0 溢出后，PWM0OUT 清零。可以通过程序再次设置 PWM0OUT 输出下一个脉冲信号。

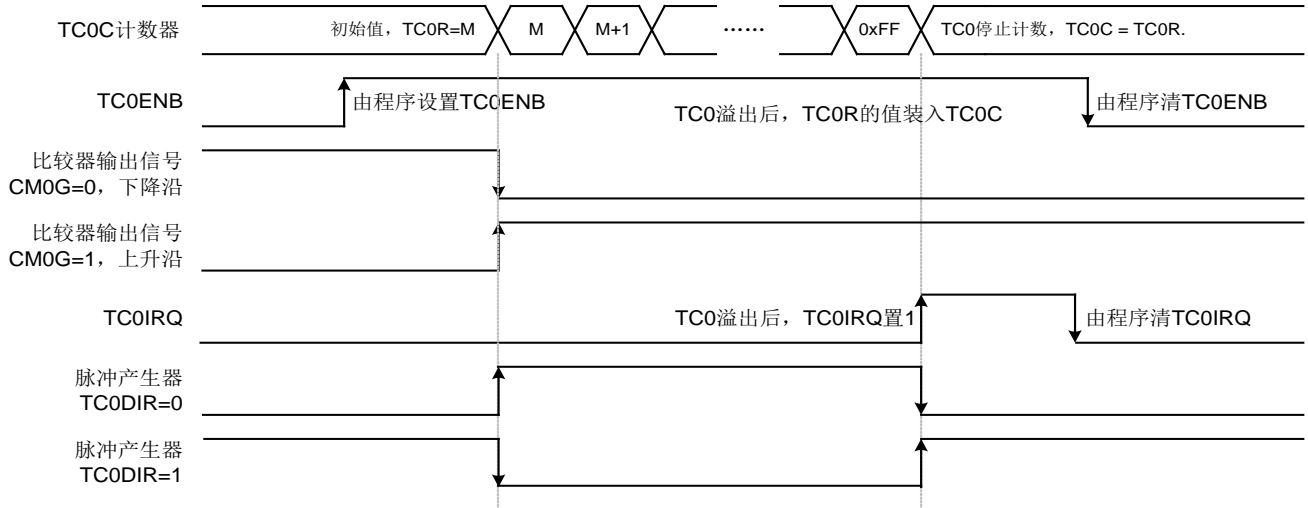
TC0 Rate=Fhosc/2=8MHz @Fhosc=16MHz					
TC0C/TC0R	0x00	0x01	...	0xFE	0xFF
脉宽 (ns)	31875	31750	...	125	0
TC0 Rate=Fhosc/4=4KHz @Fhosc=16MHz					
TC0C/TC0R	0x00	0x01	...	0xFE	0xFF
脉宽 (us)	63750	63500	...	250	0
TC0 Rate=Fhosc/256=62.5KHz @Fhosc=16MHz					
TC0C/TC0R	0x00	0x01	...	0xFE	0xFF
脉宽 (us)	4080	4064	...	16	0
TC0 Rate=Fcpu/2=0.5MHz @Fcpu=Fhosc/16=16MHz/16=1MHz					
TC0C/TC0R	0x00	0x01	...	0xFE	0xFF
脉宽 (us)	510	508	...	2	0
TC0 Rate=Fcpu/256=3.90625KHz @Fcpu=Fhosc/16=16MHz/16=1MHz					
TC0C/TC0R	0x00	0x01	...	0xFE	0xFF
脉宽 (us)	65280	65024	...	256	0

TC0PO=1, CMOSF=0: TC0ENB 必须置 1, TC0 8 位二进制计数器由 PWM0OUT 位控制。若由程序设置 PWM0OUT 为 1, TC0 开始计数, TC0 溢出时, TC0 停止计数, PWM0OUT 自动清零, TC0IRQ 有效, TC0R 最新的置装入 TC0C。必须通过程序设置 PWM0OUT 为 1, 让 TC0 再次开始计数。



若触发源为比较器 CM0 输出沿（上升沿和下降沿，由比较器的控制寄存器的 CM0G 位控制），触发时开始输出脉冲信号，TC0 溢出后，脉冲输出引脚返回到空闲状态。

TC0PO=1, CM0SF=1: TC0ENB 必须置 1, TC0 8 位二进制计数器由比较器 CM0 输出沿条件控制，通过 CM0G 位选择触发沿。比较器输出沿触发时，TC0 开始计数，TC0 溢出后，TC0 停止计数，TC0IRQ 有效，TC0R 最新的值装入 TC0C 中。



8.3.5 TC0M模式寄存器

模式寄存器 TC0M 控制 TC0 的工作模式，包括 TC0 分频、时钟源和 PWM 功能等。这些设置必须在使能 TC0 定时器之前完成。

0B4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	TC0DIR	TC0PO	PWM0OUT
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 0 **PWM0OUT**: PWM0 输出控制位。

TC0PO=0:

- 0 = 禁止 PWM0 输出，P0.1 为普通 I/O 引脚；
- 1 = 允许 PWM0 输出，P0.1 输出 PWM 信号。

TC0PO=1:

- 0 = 停止脉冲输出，否则脉冲输出的结尾会自动清零；
- 1 = 使能脉冲输出。

Bit 1 **TC0PO**: TC0 脉冲输出功能控制位。

- 0 = 禁止；
- 1 = 使能 TC0 脉冲输出功能，由 P0.1 输出。

Bit 2 **TC0DIR**: PWM0 和脉冲产生器输出相位选择位。

- 0 = 普通相位，高电平有效，低电平空闲；
- 1 = 反向相位，低电平有效，高电平空闲。

Bit 3 **TC0CKS**: TC0 时钟信号控制位。

- 0 = 内部时钟 Fcpu；
- 1 = 高速时钟 Fhsoc。

Bit [6:4] **TC0RATE[2:0]**: TC0 分频选择位。

TC0CKS=0:

- 000 = Fcpu/256; 001 = Fcpu/128; 010 = Fcpu/64; 011 = Fcpu/32; 100 = Fcpu/16; 101 = Fcpu/8;
- 110 = Fcpu/4; 111 = Fcpu/2。

TC0CK2=1:

- 000 = Fhosc/256; 001 = Fhosc/128; 010 = Fhosc/64; 011 = Fhosc/32; 100 = Fhosc/16; 101 = Fhosc/8;
- 110 = Fhosc/4; 111 = Fhosc/2。

Bit 7 **TC0ENB**: TC0 启动控制位。

- 0 = 关闭 TC0 定时器；
- 1 = 开启 TC0 定时器。

8.3.6 TC0C计数寄存器

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

0B5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0C	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

TC0C 初始值的计算公式如下：

$$\text{TC0C 初始值} = 256 - (\text{TC0 中断间隔时间} * \text{输入时钟 rate})$$

8.3.7 TC0R 自动重装寄存器

TC0 内置自动重装功能，TC0R 寄存器存储重装值。TC0C 溢出时，TC0R 的值自动装入 TC0C 中。TC0 定时器工作在计时模式时，要通过修改 TC0R 寄存器来修改 TC0 的间隔时间，而不是通过修改 TC0C 寄存器。在 TC0 定时器溢出后，新的 TC0C 值会被更新，TC0R 会将新的值装载到 TC0C 寄存器中。但在初次设置 TC0M 时，必须要在开启 TC0 定时器前把 TC0C 以及 TC0R 设置成相同的值。

TC0 为双重缓存器结构。若程序对 TC0R 进行了修改，那么修改后的 TC0R 值首先被暂存在 TC0R 的第一个缓存器中，TC0 溢出后，TC0R 的新值就会被存入 TC0R 缓存器中，从而避免 TC0 间隔时间出错以及 PWM 误动作。

0B6H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0R	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

TC0R 初始值计算公式如下：

$$\text{TC0R 初始值} = 256 - (\text{TC0 中断间隔时间} * \text{TC0 时钟 Rate})$$

➤ 例：计算 TC0C 和 TC0R 的值，TC0 的间隔时间为 10ms，时钟源来自 $F_{cpu} = 16\text{MHz}/16 = 1\text{MHz}$ ， $\text{TC0RATE} = 000$ ($F_{cpu}/128$)。

TC0 间隔时间为 10ms，TC0 时钟 Rate=16MHz/16/128

$$\begin{aligned} \text{TC0C/TC0R 初始值} &= 256 - (\text{TC0 中断间隔时间} * \text{输入时钟}) \\ &= 256 - (10\text{ms} * 16\text{MHz} / 16 / 128) \\ &= 256 - (10 \cdot 2 * 16 * 106 / 16 / 128) \\ &= \text{B2H} \end{aligned}$$

8.3.8 TC0D PWM 占空比寄存器

TC0D 寄存器用来控制 PWM 的占空比。PWM 模式下，TC0R 控制 PWM 的周期，TC0D 控制 PWM 的占空比。TC0C=TC0D 时，PWM 切换为低电平。这样在应用中易于设置 TC0D 以选择合适的 PWM 占空比。

0B7H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0D	TC0D7	TC0D6	TC0D5	TC0D4	TC0D3	TC0D2	TC0D1	TC0D0
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

TC0D 初始值的计算方法如下：

$$\text{TC0D 初始值} = \text{TC0R} + (\text{PWM 脉冲高电平宽度周期} / \text{TC0 时钟 rate})$$

➤ 例：计算 TC0D 的值。1/3 占空比 PWM，TC0 时钟源 $F_{cpu} = 16\text{MHz}/16 = 1\text{MHz}$ ， $\text{TC0RATE} = 000$ ($F_{cpu}/128$)。TC0R = B2H，TC0 间隔时间=10ms，PWM 周期频率为 100Hz，1/3 占空比条件下，PWM 高电平的宽度值约为 3.33ms。

$$\begin{aligned} \text{TC0D 初始值} &= \text{B2H} + (\text{PWM 脉冲高电平宽度值} / \text{TC0 时钟 Rate}) \\ &= \text{B2H} + (3.33\text{ms} * 16\text{MHz} / 16 / 128) \\ &= \text{B2H} + 1\text{AH} \\ &= \text{CCH} \end{aligned}$$

8.3.9 TC0 定时器操作举例

● TC0 定时器

; 复位 TC0。

```
CLR          TC0M          ; 清 TC0M。
```

; 设置 TC0Rate。

```
MOV          A, #0nnn0n00b
B0MOV        TC0M, A
```

; 设置 TC0 时钟源。

```
B0BCLR       FTC0CKS      ; Fcpu。
```

; or

```
B0BSET       FTC0CKS      ; Fhosc。
```

; 设置 TC0C 和 TC0R 获得 TC0 的间隔时间。

```
MOV          A, #value
B0MOV        TC0C, A      ; TC0C 必须和 TC0R 相等。
B0MOV        TC0R, A
```

; 清 TC0IRQ。

```
B0BCLR       FTC0IRQ
```

; 使能 TC0 定时器和中断功能。

```
B0BSET       FTC0IEN      ; 使能 TC0 中断。
B0BSET       FTC0ENB      ; 使能 TC0 定时器。
```

● TC0 PWM

; 复位 TC0。

```
CLR          TC0M          ; 清 TC0M。
```

; 设置 TC0Rate。

```
MOV          A, #0nnn0n00b
B0MOV        TC0M, A
```

; 设置 TC0 时钟源。

```
B0BCLR       FTC0CKS      ; Fcpu。
```

; or

```
B0BSET       FTC0CKS      ; Fhosc。
```

; 设置 TC0C 和 TC0R 寄存器获得 PWM 周期。

```
MOV          A, #value1
B0MOV        TC0C, A      ; TC0C 必须和 TC0R 相等。
B0MOV        TC0R, A
```

; 设置 TC0D 寄存器获得 PWM 占空比。

```
MOV          A, #value2
B0MOV        TC0D, A      ; TC0D 的值必须大于 TC0R 的值。
```

; 设置 PWM 输出相位。

```
B0BCLR       FTC0DIR      ; 高电平。
```

; or

```
B0BSET       FTC0DIR      ; 低电平。
```

; 使能 PWM 和 TC0 定时器。

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


- TC0 脉冲产生器:

```

; 复位 TC0。
CLR TC0M ; 清 TC0M。

; 设置 TC0Rate。
MOV A, #0nnn0n00b
B0MOV TC0M, A

; 设置 TC0 时钟源。
B0BCLR FTC0CKS ; Fcpu。
; or
B0BSET FTC0CKS ; Fhosc。

; 设置 TC0C 和 TC0R 寄存器获得脉冲宽度。
MOV A, #value1 ; TC0C 必须和 TC0R 相等。
B0MOV TC0C, A
B0MOV TC0R, A

; 设置脉冲输出相位。
B0BCLR FTC0DIR ; 高电平。
; or
B0BSET FTC0DIR ; 低电平。

; 设置脉冲输出触发源。
B0BCLR FCM0SF ; PWM0OUT。
; or
B0BSET FCM0SF ; 比较器 CM0 输出沿。

; 使能脉冲输出和 TC0 定时器。
B0BSET FTC0PO ; 使能脉冲输出功能。
B0BSET FTC0ENB ; 使能 TC0 定时器。

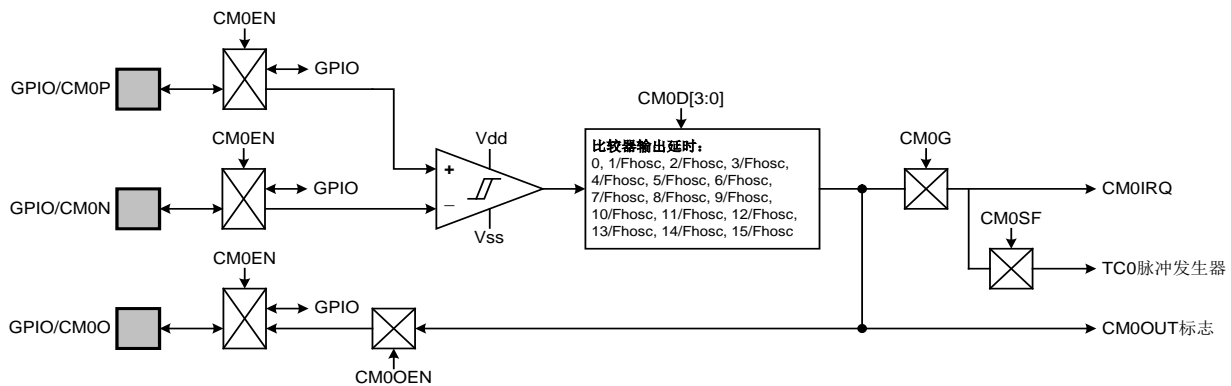
```

9 模拟比较器CM0

9.1 概述

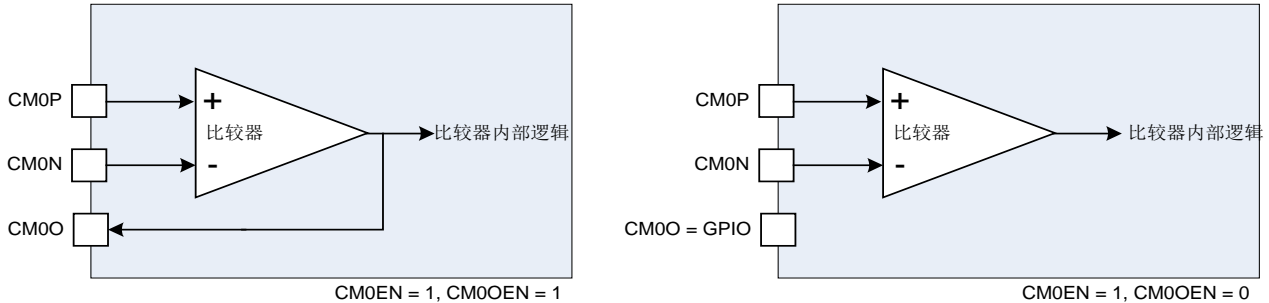
SN8P2740 系列单片机内置比较器 CM0，具有 TC0 脉冲产生器触发功能。比较器 CM0 包括普通比较器模式和 TC0 脉冲输出触发源模式。比较器 CM0 为 rail-to-rail 结构，即输入/输出电压的实际值在 VDD~VSS 之间。当输入电压的正极大于输入电压的负极时，比较器输出高电平；当输入电压的正极小于输入电压的负极时，比较器输出低电平。比较器 CM0 的主要用途如下：

- ☞ **普通比较器功能：**比较输入端的正极电压和负极电压。
- ☞ **中断功能：**比较器 CM0 支持中断功能。当 CM0 的输出方向等同于所选择的方向时，CM0IRQ 有效，程序计数器跳至中断向量执行中断。
- ☞ **TC0 脉冲产生器触发源：**比较器 CM0 可以为 TC0 脉冲产生器的触发源，由 CM0SF 位控制。TC0PO=1，且 CM0SF=1 时，CM0 输出状态触发 TC0 脉冲产生器，输出脉冲信号。
- ☞ **绿色模式功能：**CM0 在绿色模式下继续工作，但无唤醒功能。触发时 CM0IRQ 被锁存，直到系统被唤醒并返回到普通模式。系统被唤醒后，执行 CM0 中断。



9.2 普通比较器模式

比较器引脚与 GPIO 引脚共用，由 CM0EN 位控制，CM0EN=1 时，使能 CM0N 引脚，连接到比较器的负极输入引脚，使能 CM0P 引脚，连接到比较器的正极输入引脚。CM0OEN 控制比较器的输出引脚是否使能 GPIO 功能，CM0OEN=1 时，比较器的输出端连接至 CM0O 引脚，并隔离 GPIO 功能；CM0OEN=0 时，比较器的输出状态可以通过 CM0OUT 读取，CM0O 引脚为 GPIO 引脚。

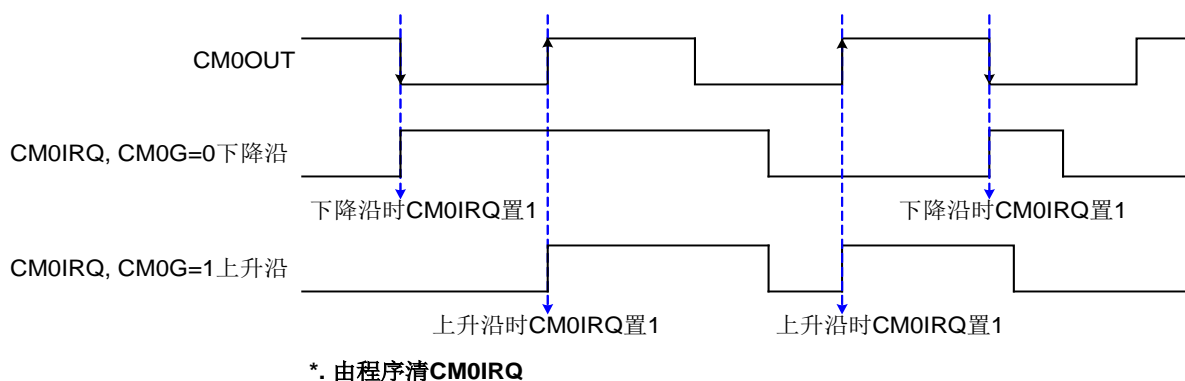


* 注：比较器的使能条件为：CM0EN=1，否则比较器引脚为 GPIO 引脚，比较器功能被禁止。

CM0OUT 和 CM0IRQ 位都显示比较结果：不同的是 CM0OUT 立即显示出实时值，CM0IRQ 显示比较结果，由寄存器的 CM0G 位（触发沿控制位）控制，触发沿包括上升沿（CM0OUT 由低到高），下降沿（CM0OUT 由高到低），CM0G=0，下降沿触发比较器 CM0 中断；CM0G=1，上升沿触发比较器 CM0 中断。

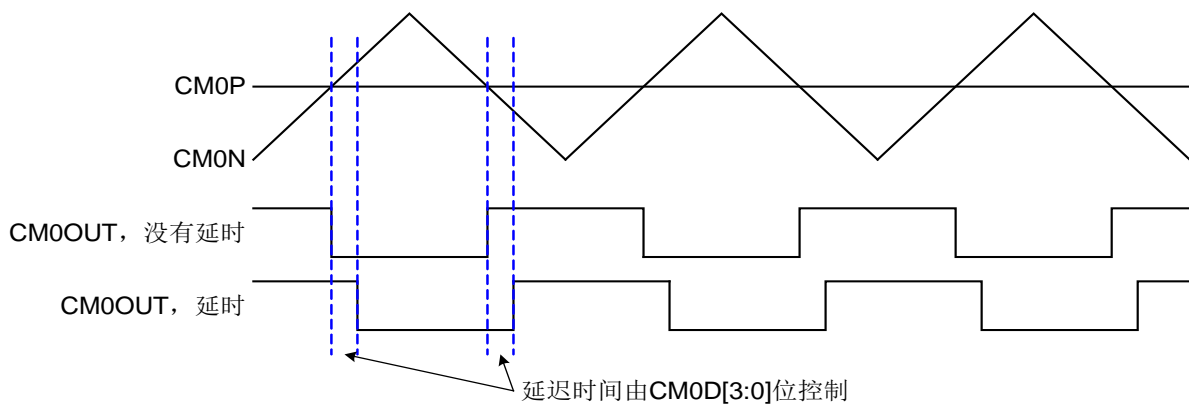
* 注：CM0OUT 显示比较器的原始状态，不锁存，根据比较器的处理结果而更改。但 CM0IRQ 锁存比较器的输出结果，由程序清零。

比较器 CM0 支持中断，中断触发源由 CM0G 位选择，包括上升沿和下降沿。CM0G=0，下降沿触发；CM0G=1，上升沿触发。通过延时处理结果显示比较器的输出信号的方向，当比较器的输出沿方向与 CM0G 的条件等同时，CM0IRQ 有效，CM0IEN=1 时，程序计数器跳至中断向量执行中断。



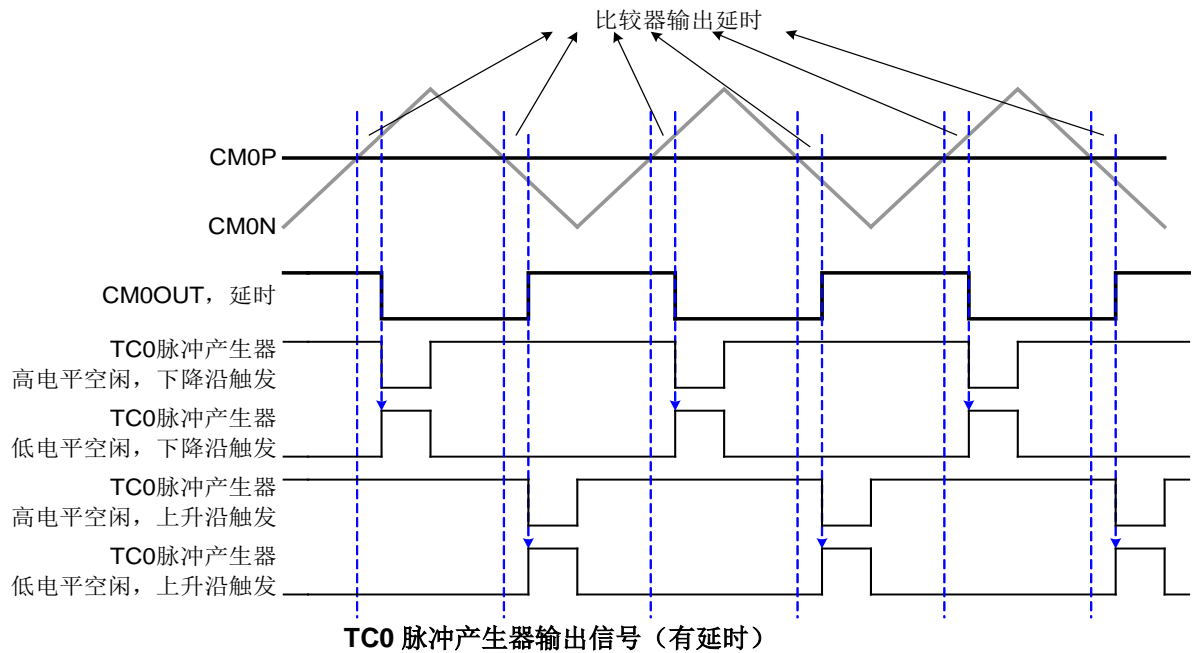
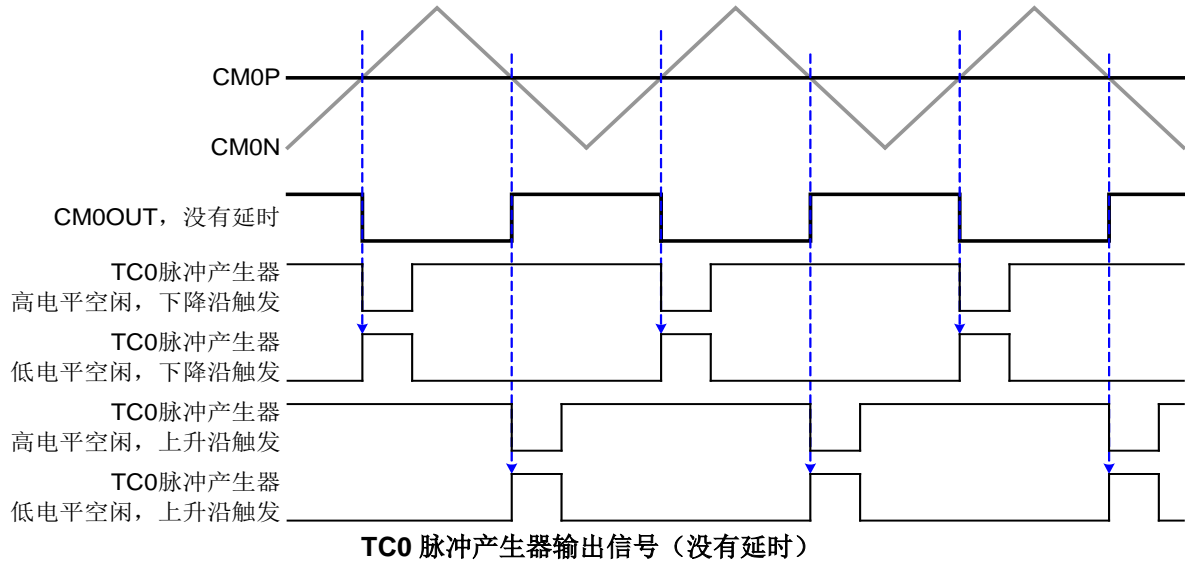
比较器 CM0 比较正极输入端和负极输入端的电压，并将结果由输出引脚输出。当 $V+ > V-$ ，比较器输出高电平； $V+ < V-$ ，比较器输出低电平。比较器输出端内置一个延时控制模块，延迟输出结果并滤除输出转换条件。延时有 16-step，包括无延时、 $1/F_{osc}$ 、 $2/F_{osc}$ 、 $3/F_{osc}$ 、 $4/F_{osc}$ 、 $5/F_{osc}$ 、 $6/F_{osc}$ 、 $7/F_{osc}$ 、 $8/F_{osc}$ 、 $9/F_{osc}$ 、 $10/F_{osc}$ 、 $11/F_{osc}$ 、 $12/F_{osc}$ 、 $13/F_{osc}$ 、 $14/F_{osc}$ 和 $15/F_{osc}$ ，由 CMOD[3:0]控制。

CMOD[3:0]	0000b	0001b	0010b	0011b	0100b	0101b	0110b	0111b
	No	$1/F_{osc}$	$2/F_{osc}$	$3/F_{osc}$	$4/F_{osc}$	$5/F_{osc}$	$6/F_{osc}$	$7/F_{osc}$
延迟时间(us) $F_{osc}=16\text{MHz}$	0	0.0625	0.125	0.1875	0.25	0.3125	0.375	0.4375
延迟时间(us) $F_{osc}=4\text{MHz}$	0	0.25	0.5	0.75	1	1.25	1.5	1.75
CMOD[3:0]	1000b	1001b	1010b	1011b	1100b	1101b	1110b	1111b
	$8/F_{osc}$	$9/F_{osc}$	$10/F_{osc}$	$11/F_{osc}$	$12/F_{osc}$	$13/F_{osc}$	$14/F_{osc}$	$15/F_{osc}$
延迟时间(us) $F_{osc}=16\text{MHz}$	0.5	0.5625	0.625	0.6875	0.75	0.8125	0.875	0.9375
延迟时间(us) $F_{osc}=4\text{MHz}$	2	2.25	2.5	2.75	3	3.25	3.5	3.75



9.3 比较器CM0 特殊功能

除了普通比较器功能之外，CM0 还内置一个特殊功能，通过比较器的输出沿来触发 TC0 脉冲产生器，由 CM0SF 位控制。CM0SF=1 时，使能 CM0 特殊功能。若 CM0 的输出满足 TC0 脉冲产生器的触发条件，则 TC0 脉冲产生器被触发并输出脉冲信号，CM0 的中断功能有效。详情请参考 TC0 脉冲产生器章节。



9.4 比较器CM0 模式寄存器

09CH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM0M	CM0EN	CM0OEN	CM0OUT	CM0SF	CM0G	-	-	-
读/写	R/W	R/W	R/W	R/W	R/W	-	-	-
复位后	0	0	0	0	0	-	-	-

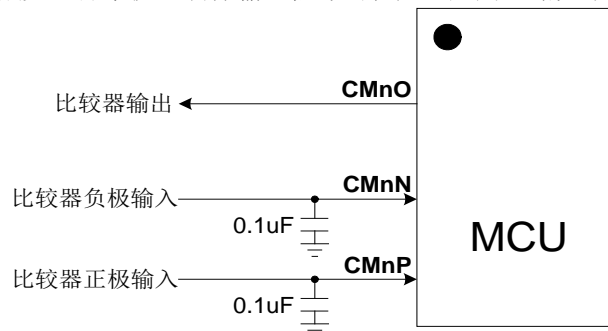
- Bit 3 **CM0G**: 比较器 CM0 输出触发方向选择控制位。
0 = 下降沿触发, $CM0P < CM0N$, 比较器输出状态由高变低;
1 = 上升沿触发, $CM0P > CM0N$, 比较器输出状态由低变高。
- Bit 4 **CM0SF**: 比较器 CM0 特殊模式控制位。
0 = 禁止, CM0 为普通比较器模式;
1 = 使能, CM0 输出沿触发 TC0 脉冲产生器。
- Bit 5 **CM0OUT**: 比较器 CM0 输出标志位。
0 = CM0P 电压小于 CM0N 电压;
1 = CM0P 电压大于 CM0N 电压。
- Bit 6 **CM0OEN**: 比较器 CM0 输出引脚控制位。
0 = 禁止, CM0O 为 GPIO 引脚;
1 = 使能, CM0O 为比较器输出引脚, 隔离 GPIO 功能。
- Bit 7 **CM0EN**: 比较器 CM0 控制位。
0 = 禁止, 比较器引脚为 GPIO 引脚;
1 = 使能, CM0N 和 CM0P 引脚为比较器引脚, CM0O 由 CM0OEN 为控制。

09AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMDB0	CM1D3	CM1D2	CM1D1	CM1D0	CM0D3	CM0D2	CM0D1	CM0D0
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 [3:0] **CM0D[3:0]**: 比较器 CM0de-bounce 时间控制位。
0000= 没有延时; 0001=1/Fhos; 0010=2/Fhosc; 0011=3/Fhosc; 0100=4/Fhosc; 0101=5/Fhosc;
0110=6/Fhosc; 0111=7/Fhosc; 1000=8/Fhosc; 1001=9/Fhosc; 1010=10/Fhosc; 1011=11/Fhosc;
1100=12/Fhosc; 1101=13/Fhosc; 1110=14/Fhosc; 1111=15/Fhosc。

9.5 比较器应用注意事项

比较器是指比较正极电压和负极电压并将结果输出, 正负极的信号都是模拟信号。在硬件应用电路中, 比较器的输入引脚必须连接一个 0.1uF 的电容以减少电源干扰, 确保输入信号的稳定。应用电路如下所示:



9.6 比较器CM0 操作举例

- 比较器 CM0:

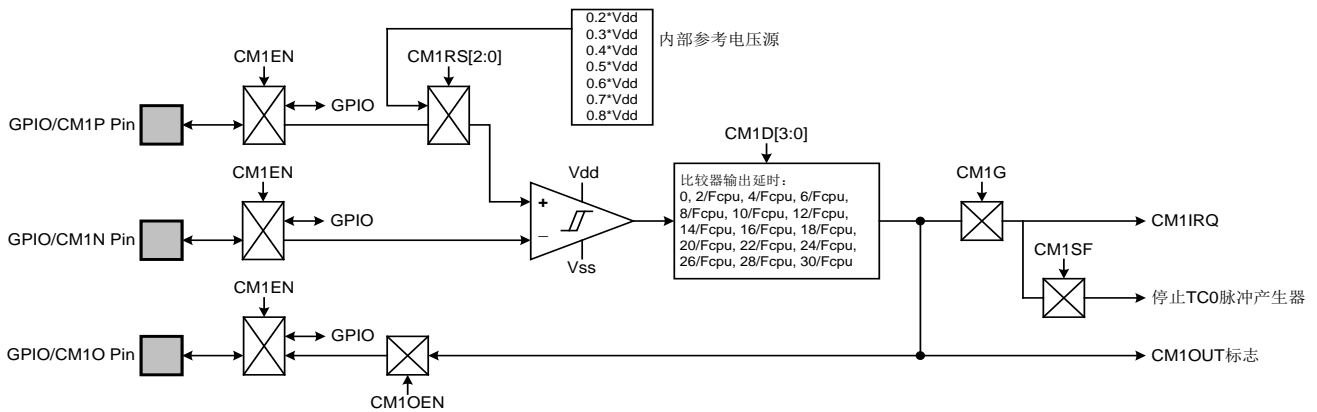
; 复位比较器 CM0。	CLR	CM0M	; 清 CM0M 寄存器。
; 设置比较器 CM0 的功能模式。	B0BCLR	FCM0SF	; 普通比较器模式。
; or	B0BSET	FCM0SF	; 特殊功能模式。
; 设置比较器 CM0 的输出引脚。	B0BCLR	FCM0OEN	; 禁止比较器 CM0 的输出引脚。
; or	B0BSET	FCM0OEN	; 使能比较器 CM0 的输出引脚。
; 设置比较器 CM0 中断触发沿。	B0BCLR	FCM0G	; 下降沿。
; or	B0BSET	FCM0G	; 上升沿。
; 设置比较器 CM0 输出 de-bounce。	B0MOV AND OR B0MOV	A, CMDB0 A, #11110000b A, #0000nnnnb CMDB0, A	; 通过 CM0D[3:0]设置比较器 CM0 输出 de-bounce。
; 清 CM0IRQ。	B0BCLR	FCM0IRQ	
; 使能比较器 CM0 和中断功能。	B0BSET B0BSET	FCM0IEN FCM0EN	; 使能比较器 CM0 中断功能。 ; 使能比较器 CM0。

10 模拟比较器CM1

10.1 概述

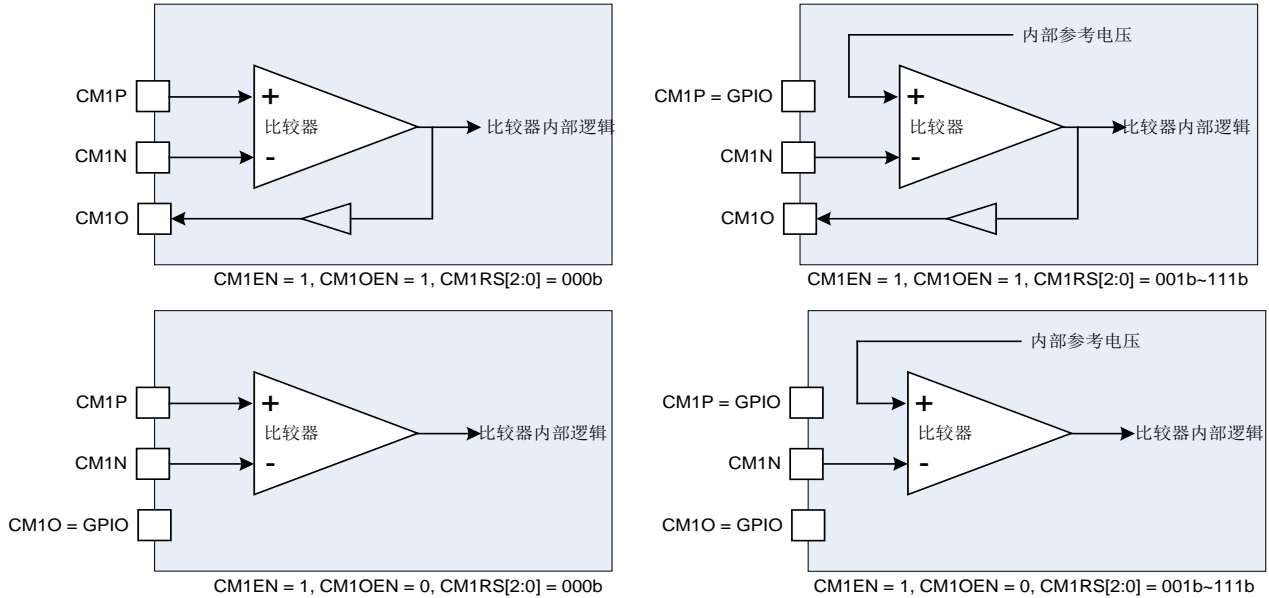
SN8P2740 系列单片机内置比较器 CM1，可以停止 TC0 脉冲产生器触发功能。比较器 CM1 包括普通比较器模式和停止 TC0 脉冲输出触发源模式。比较器 CM1 为 rail-to-rail 结构，即输入/输出电压的实际值在 VDD~VSS 之间。当输入电压的正极大于输入电压的负极时，比较器输出高电平；当输入电压的正极小于输入电压的负极时，比较器输出低电平。比较器 CM1 还内置一个内部参考电压源，连接到比较器的正极输入端，使能内部参考电压源时，比较器 CM1 的正极输入引脚为 GPIO 模式。比较器 CM1 的主要用途如下：

- ☞ **普通比较器功能：**比较输入端的正极电压和负极电压。
- ☞ **中断功能：**比较器 CM1 支持中断功能。当 CM1 的输出方向等同于所选择的方向时，CM1IRQ 有效，程序计数器跳至中断向量执行中断。
- ☞ **停止 TC0 脉冲产生器触发源：**比较器 CM1 可以停止 TC0 脉冲产生器的触发源，由 CM1SF 位控制。TC0PO=1，且 CM1SF=1 时，CM1 输出状态触发 TC0 脉冲产生器，停止输出脉冲信号。
- ☞ **绿色模式功能：**CM1 在绿色模式下继续工作，但无唤醒功能。触发时 CM1IRQ 被锁存，直到系统被唤醒并返回到普通模式。系统被唤醒后，执行 CM1 中断。



10.2 普通比较器模式

比较器引脚与 GPIO 引脚共用，由 CM1EN 位控制，CM1EN=1 时，使能 CM1N 引脚，连接到比较器的负极输入引脚。比较器 CM1 的正极输入端由 CM1RS[2:0]控制，CM1RS[2:0]=000 时，比较器正极输入端连接到 CM1P，隔离该引脚的 GPIO 功能；CM1RS[2:0]=001~111 时，比较器的正极输入端连接到内部参考电压源上，包括 7 种参考源：0.2*Vdd、0.3*Vdd、0.4*Vdd、0.5*Vdd、0.6*Vdd、0.7*Vdd 和 0.8*Vdd，CM1P 引脚为 GPIO 引脚。CM1OEN 控制比较器的输出引脚是否使能 GPIO 功能，CM1OEN=1 时，比较器的输出端连接至 CM1O 引脚，并隔离 GPIO 功能；CM1OEN=0 时，比较器的输出状态可以通过 CM1OUT 读取，CM1O 引脚为 GPIO 引脚。

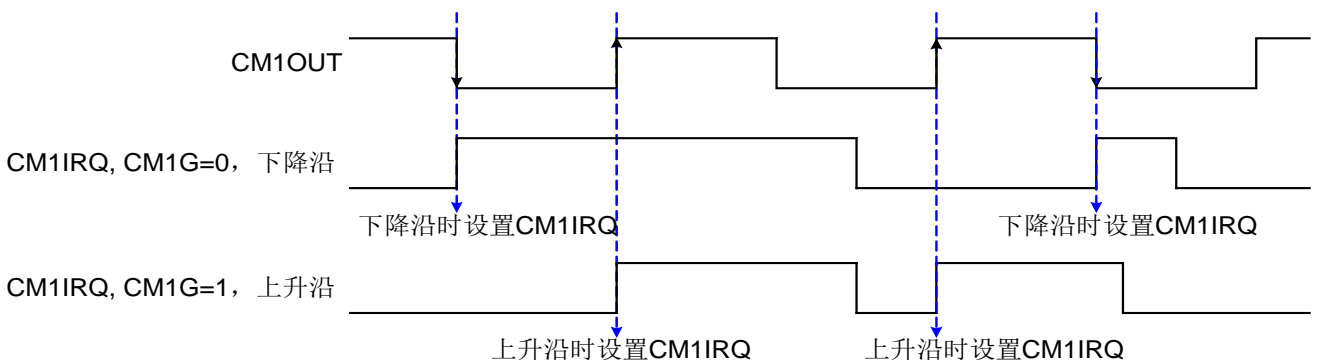


* 注：比较器的使能条件为：CM1EN=1，否则比较器引脚为 GPIO 引脚，比较器功能被禁止。

CM1OUT 和 CM1IRQ 位都显示比较结果：不同的是 CM1OUT 立即显示出实时值，CM1IRQ 显示比较结果，由寄存器的 CM1G 位（触发沿控制位）控制，触发沿包括上升沿（CM1OUT 由低到高），下降沿（CM1OUT 由高到低），CM1G=0，下降沿触发比较器 CM1 中断；CM1G=1，上升沿触发比较器 CM1 中断。

* 注：CM1OUT 显示比较器的原始状态，不锁存，根据比较器的处理结果而更改。但 CM1IRQ 锁存比较器的输出结果，由程序清零。

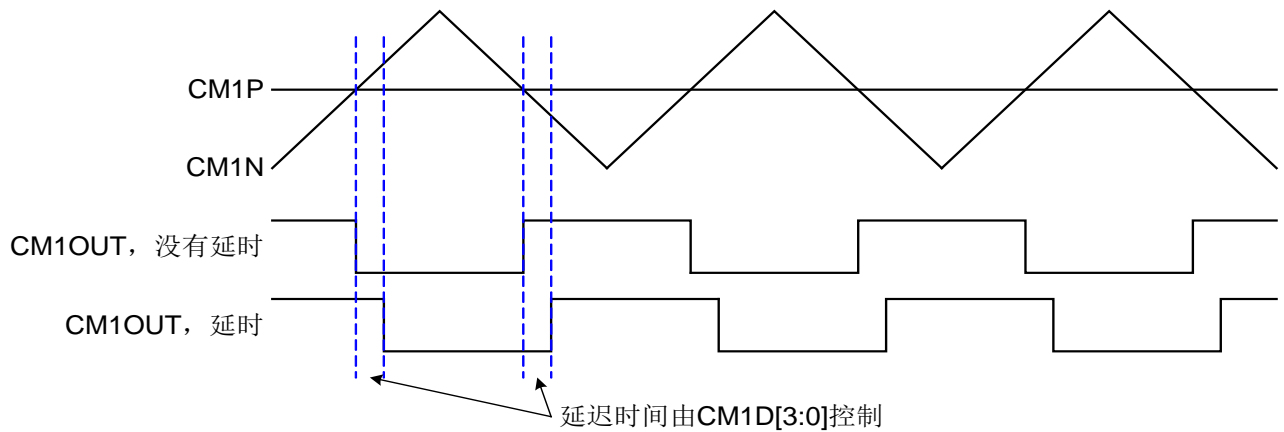
比较器 CM1 支持中断，中断触发源由 CM1G 位选择，包括上升沿和下降沿。CM1G=0，下降沿触发；CM1G=1，上升沿触发。通过延时处理结果显示比较器的输出信号的方向，当比较器的输出沿方向与 CM1G 的条件等同时，CM1IRQ 有效，CM1IEN=1 时，程序计数器跳至中断向量执行中断。



*. 由程序清CM1IRQ

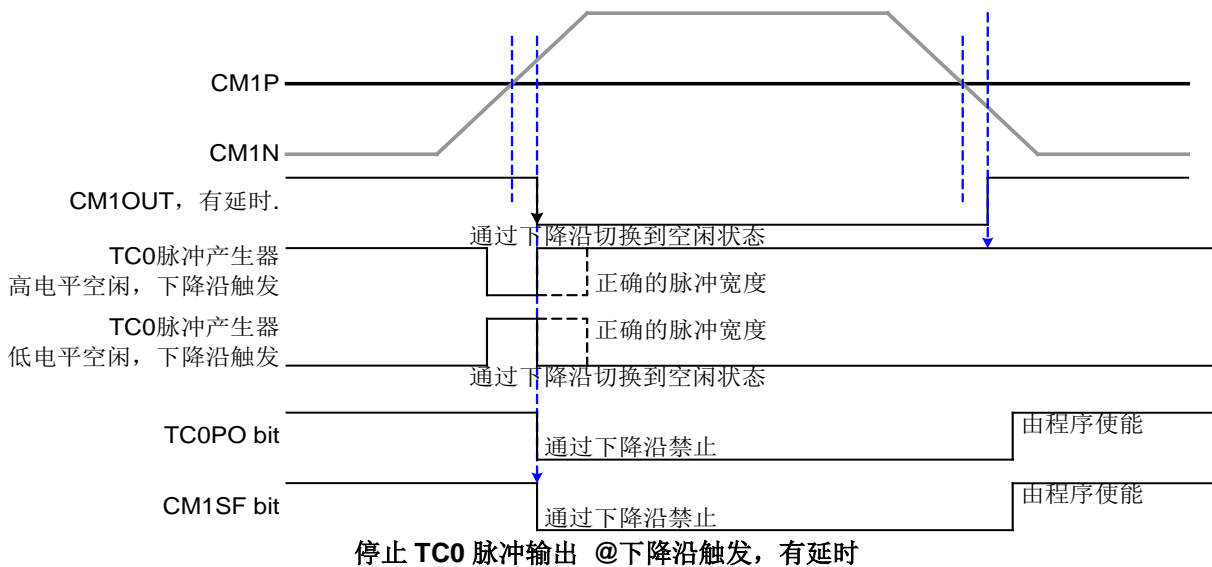
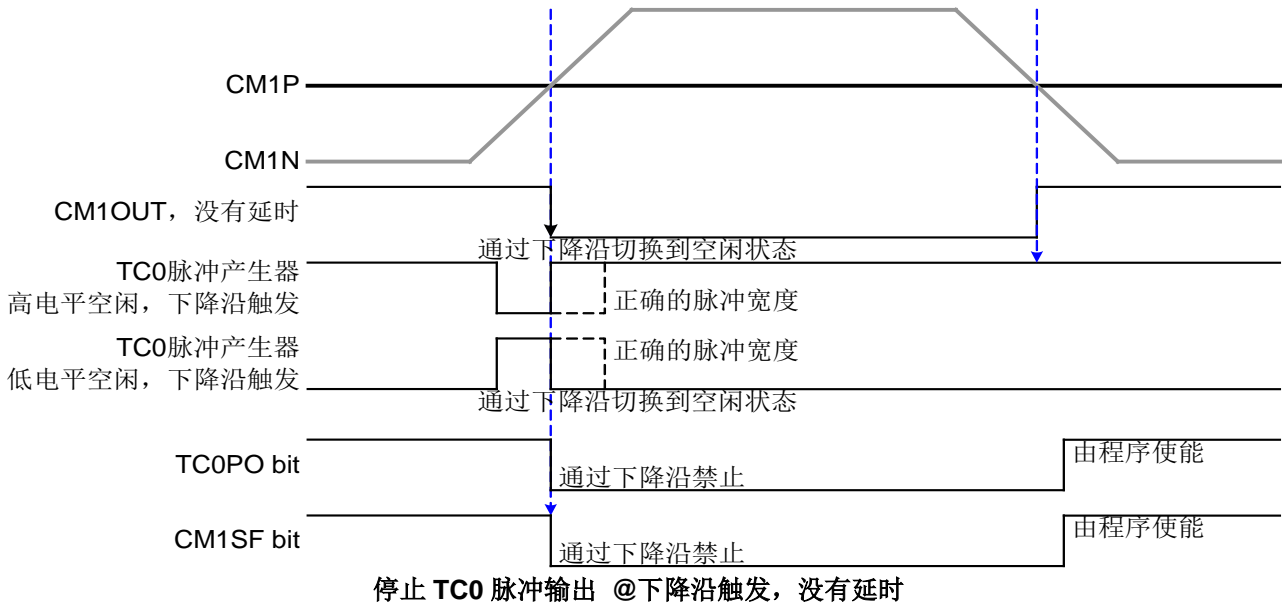
比较器 CM1 比较正极输入端和负极输入端的电压，并将结果由输出引脚输出。当 $V+ > V-$ ，比较器输出高电平； $V+ < V-$ ，比较器输出低电平。比较器输出端内置一个延时控制模块，延迟输出结果并滤除输出转换条件。延时有 16-step，包括无延时、 $2/F_{cpu}$ 、 $4/F_{cpu}$ 、 $6/F_{cpu}$ 、 $8/F_{cpu}$ 、 $10/F_{cpu}$ 、 $14/F_{cpu}$ 、 $16/F_{cpu}$ 、 $18/F_{cpu}$ 、 $20/F_{cpu}$ 、 $22/F_{cpu}$ 、 $24/F_{cpu}$ 、 $26/F_{cpu}$ 、 $28/F_{cpu}$ 和 $30/F_{cpu}$ ，由 CM1D[3:0]控制。

CM1D[2:0]	0000b	0001b	0010b	0011b	0100b	0101b	0110b	0111b
	No	$2/F_{cpu}$	$4/F_{cpu}$	$6/F_{cpu}$	$8/F_{cpu}$	$10/F_{cpu}$	$12/F_{cpu}$	$14/F_{cpu}$
延迟时间(us) $F_{cpu}=F_{osc}/4=16MHz/4=4MHz$	0	0.5	1	1.5	2	2.5	3	3.5
延迟时间(us) $F_{cpu}=F_{osc}/16=16MHz/16=1MHz$	0	2	4	6	8	10	12	14
CM1D[2:0]	1000b	1001b	1010b	1011b	1100b	1101b	1110b	1111b
	$16/F_{cpu}$	$18/F_{cpu}$	$20/F_{cpu}$	$22/F_{cpu}$	$24/F_{cpu}$	$26/F_{cpu}$	$28/F_{cpu}$	$30/F_{cpu}$
延迟时间(us) $F_{cpu}=F_{osc}/4=16MHz/4=4MHz$	4	4.5	5	5.5	6	6.5	7	7.5
延迟时间(us) $F_{cpu}=F_{osc}/16=16MHz/16=1MHz$	16	18	20	22	24	26	28	30



10.3 比较器CM1 特殊功能

除了普通比较器功能之外，CM1 还内置一个特殊功能，通过比较器的输出触发沿触发 TC0 脉冲产生器停止名称输出，由 CM1SF 位控制。CM1SF=1 时，使能 CM1 特殊功能。若 CM1 的输出满足停止 TC0 脉冲产生器的触发条件，则禁止 TC0 脉冲产生器功能并关闭外部设备，此时 TC0PO、TC0ENB 和 CM0SF 位都被清零，自动禁止脉冲输出功能，脉冲输出引脚返回到上一个 GPIO 模式，CM1IRQ 有效。必须通过程序使能脉冲产生器。



* 注：通过比较器 CM1 特殊模式触发停止 TC0 脉冲输出，CM1SF 和 CM0PO 位自动清零。必须通过程序设置 CM1SF、TC0PO 和 TC0ENB 位，以恢复 TC0 脉冲产生器功能。

10.4 比较器CM1 模式寄存器

09DH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM1M	CM1EN	CM1OEN	CM1OUT	CM1SF	CM1G	CM1RS2	CM1RS1	CM1RS0
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 [2:0] **CM1RS[2:0]**: 比较器 CM1 正极电压选择控制位。

- 000 = CM1P 引脚为比较器正极输入引脚, GPIO 功能被屏蔽;
- 001 = 内部 0.2*Vdd, CM1P 为 GPIO 引脚;
- 010 = 内部 0.3*Vdd, CM1P 为 GPIO 引脚;
- 011 = 内部 0.4*Vdd, CM1P 为 GPIO 引脚;
- 100 = 内部 0.5*Vdd, CM1P 为 GPIO 引脚;
- 101 = 内部 0.6*Vdd, CM1P 为 GPIO 引脚;
- 110 = 内部 0.7*Vdd, CM1P 为 GPIO 引脚;
- 111 = 内部 0.8*Vdd, CM1P 为 GPIO 引脚。

Bit 3 **CM1G**: 比较器 CM1 输出触发方向选择控制位。

- 0 = 下降沿触发, $CM1P < CM1N$, 比较器输出状态由高变低;
- 1 = 上升沿触发, $CM1P > CM1N$, 比较器输出状态由低变高。

Bit 4 **CM1SF**: 比较器 CM1 特殊模式控制位。

- 0 = 禁止, CM1 为普通比较器模式;
- 1 = 使能, CM1 输出沿触发停止 TC0 脉冲产生器。

Bit 5 **CM1OUT**: 比较器 CM1 输出标志位。

- 0 = CM1P 电压小于 CM1N 电压;
- 1 = CM1P 电压大于 CM1N 电压。

Bit 6 **CM1OEN**: 比较器 CM1 输出引脚控制位。

- 0 = 禁止, CM1O 为 GPIO 引脚;
- 1 = 使能, CM1O 为比较器输出引脚, 隔离 GPIO 功能。

Bit 7 **CM1EN**: 比较器 CM1 控制位。

- 0 = 禁止, 比较器引脚为 GPIO 引脚;
- 1 = 使能, CM1N 引脚为比较器引脚, CM1O 由 CM1OEN 为控制, CM1P 引脚由 CM1RS[2:0]控制。

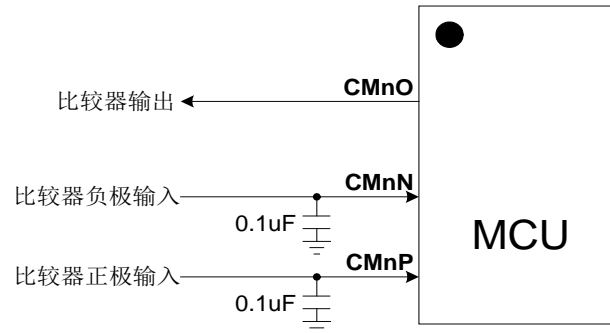
09AH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMDB0	CM1D3	CM1D2	CM1D1	CM1D0	CM0D3	CM0D2	CM0D1	CM0D0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit [7:4] **CM1D[3:0]**: 比较器 CM1 de-bounce 时间控制位。

- 0000=没有延时; 0001=2/Fcpu; 0010=4/Fcpu; 0011=6/Fcpu; 0100=8/Fcpu; 0101=10/Fcpu; 0110=12/Fcpu;
- 0111=14/Fcpu; 1000=16/Fcpu; 1001=18/Fcpu; 1010=20/Fcpu; 1011=22/Fcpu; 1100=24/Fcpu;
- 1101=26/Fcpu; 1110=28/Fcpu; 1111=30/Fcpu。

10.5 比较器应用注意事项

比较器是指比较正极电压和负极电压并将结果输出，正负极的信号都是模拟信号。在硬件应用电路中，比较器的输入引脚必须连接一个 0.1uF 的电容以减少电源干扰，确保输入信号的稳定。应用电路如下所示：



10.6 比较器CM1 操作举例

● 比较器 CM1:

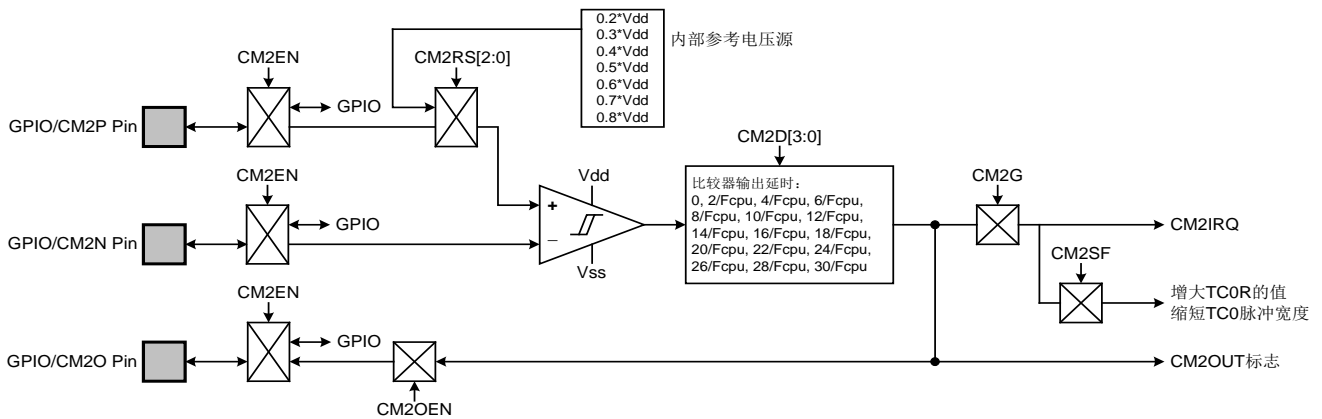
; 复位比较器 CM1。	CLR	CM1M	; 清 CM1M 寄存器。
; 设置比较器 CM1 正极输入端。	MOV B0MOV	A, #00000nnnb CM1M, A	; 通过 CM1RS[2:0]设置比较器 CM1 正极输入端。
; 设置比较器 CM1 功能模式。	B0BCLR	FCM1SF	; 普通比较器。
; or	B0BSET	FCM1SF	; 特殊功能。
; 设置比较器 CM1 输出引脚。	B0BCLR	FCM1OEN	; 禁止。
; or	B0BSET	FCM1OEN	; 使能。
; 设置比较器 CM1 中断触发沿。	B0BCLR	FCM1G	; 下降沿。
; or	B0BSET	FCM1G	; 上升沿。
; 设置比较器 CM1 输出 de-bounce。	B0MOV AND OR B0MOV	A, CMDB0 A, #11110000b A, #nnnn0000b CMDB0, A	; 通过 CM1D[3:0]设置比较器 CM1 输出 de-bounce。
; 清 CM1IRQ。	B0BCLR	FCM1IRQ	
; 使能比较器 CM1 和中断功能。	B0BSET B0BSET	FCM1IEN FCM1EN	; 使能比较器 CM1 中断功能。 ; 使能比较器 CM1。

11 模拟比较器CM2

11.1 概述

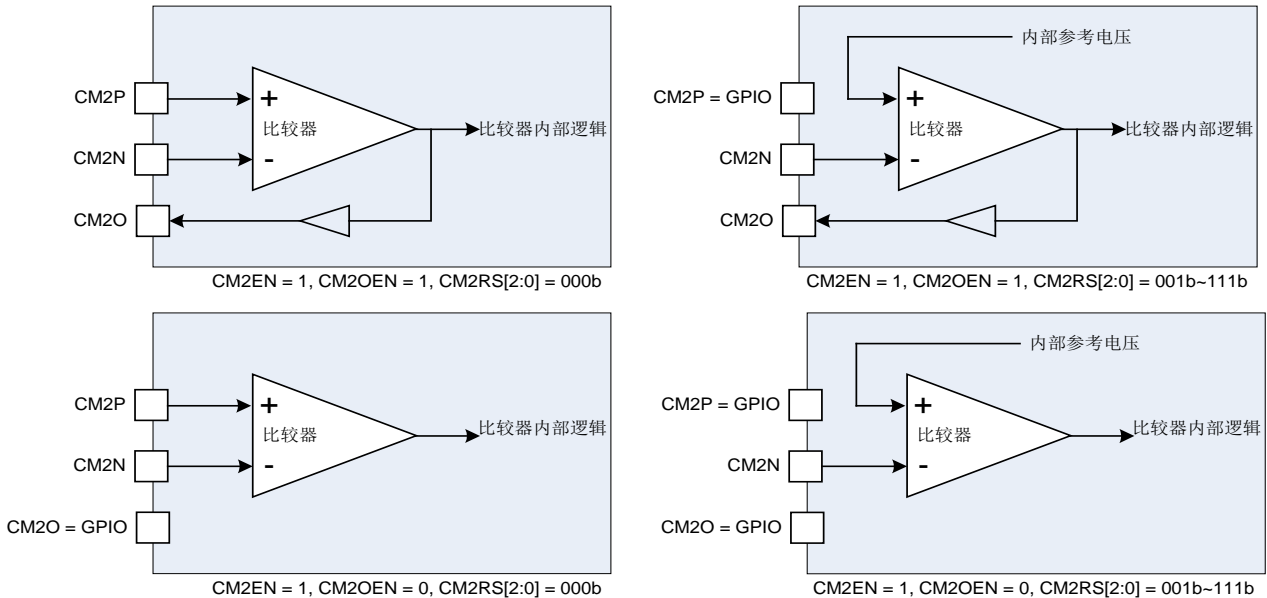
SN8P2740 系列单片机内置比较器 CM2，可以缩短 TC0 脉冲宽度。比较器 CM2 包括普通比较器模式和缩短 TC0 脉冲宽度触发源模式。比较器 CM2 为 rail-to-rail 结构，即输入/输出电压的实际值在 VDD~VSS 之间。当输入电压的正极大 于输入电压的负极时，比较器输出高电平；当输入电压的正极小于输入电压的负极时，比较器输出低电平。比较器 CM2 还内置一个内部参考电压源，连接到比较器的正极输入端，使能内部参考电压源时，比较器 CM2 的正极输入引脚为 GPIO 模式。比较器 CM2 的主要用途如下：

- ☞ **普通比较器功能：**比较输入端的正极电压和负极电压。
- ☞ **中断功能：**比较器 CM2 支持中断功能。当 CM2 的输出方向等同于所选择的方向时，CM2IRQ 有效，程序计数器跳至中断向量执行中断。
- ☞ **缩短 TC0 脉冲宽度触发源：**比较器 CM2 可以将 TC0 的脉冲宽度缩短，由 CM2SF 位控制。TC0PO=1，且 CM2SF=1 时，CM2 输出状态触发 TC0 脉冲产生器，通过增大 TC0R 的值缩短 TC0 脉冲宽度。
- ☞ **绿色模式功能：**CM2 在绿色模式下继续工作，但无唤醒功能。触发时 CM2IRQ 被锁存，直到系统被唤醒并返回到普通模式。系统被唤醒后，执行 CM2 中断。



11.2 普通比较器模式

比较器引脚与 GPIO 引脚共用，由 CM2EN 位控制，CM2EN=1 时，使能 CM2N 引脚，连接到比较器的负极输入引脚。比较器 CM2 的正极输入端由 CM2RS[2:0]控制，CM2RS[2:0]=000 时，比较器正极输入端连接到 CM2P，隔离该引脚的 GPIO 功能；CM2RS[2:0]=001~111 时，比较器的正极输入端连接到内部参考电压源上，包括 7 种参考源：0.2*Vdd、0.3*Vdd、0.4*Vdd、0.5*Vdd、0.6*Vdd、0.7*Vdd 和 0.8*Vdd，CM2P 引脚为 GPIO 引脚。CM2OEN 控制比较器的输出引脚是否使能 GPIO 功能，CM2OEN=1 时，比较器的输出端连接至 CM2O 引脚，并隔离 GPIO 功能；CM2OEN=0 时，比较器的输出状态可以通过 CM2OUT 读取，CM2O 引脚为 GPIO 引脚。

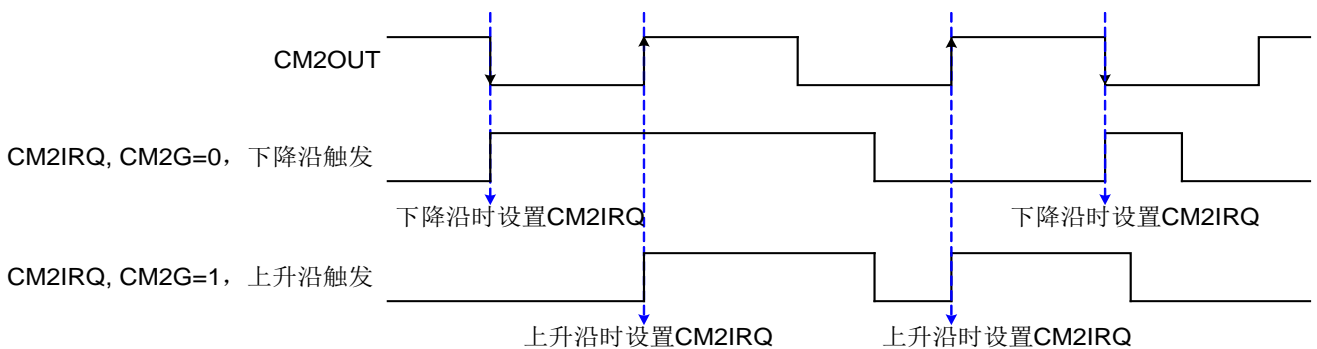


* 注：比较器的使能条件为：CM2EN=1，否则比较器引脚为 GPIO 引脚，比较器功能被禁止。

CM2OUT 和 CM2IRQ 位都显示比较结果：不同的是 CM2OUT 立即显示出实时值，CM2IRQ 显示比较结果，由寄存器的 CM2G 位（触发沿控制位）控制，触发沿包括上升沿（CM2OUT 由低到高），下降沿（CM2OUT 由高到低），CM2G=0，下降沿触发比较器 CM2 中断；CM2G=1，上升沿触发比较器 CM2 中断。

* 注：CM2OUT 显示比较器的原始状态，不锁存，根据比较器的处理结果而更改。但 CM2IRQ 锁存比较器的输出结果，由程序清零。

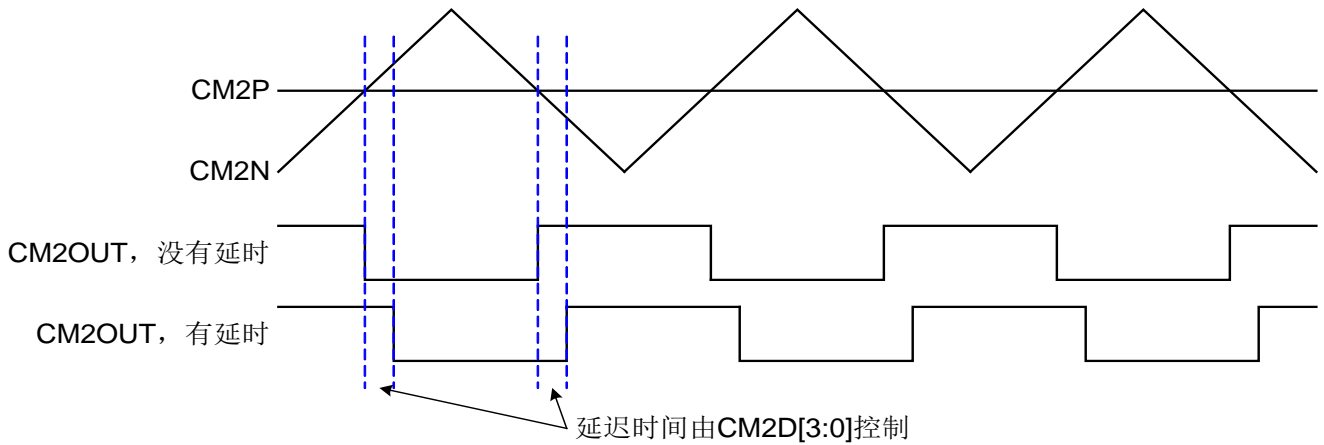
比较器 CM2 支持中断，中断触发源由 CM2G 位选择，包括上升沿和下降沿。CM2G=0，下降沿触发；CM2G=1，上升沿触发。通过延时处理结果显示比较器的输出信号的方向，当比较器的输出沿方向与 CM2G 的条件等同时，CM2IRQ 有效，CM2IEN=1 时，程序计数器跳至中断向量执行中断。



*.由程序清 CM2IRQ

比较器 CM2 比较正极输入端和负极输入端的电压，并将结果由输出引脚输出。当 $V+ > V-$ ，比较器输出高电平； $V+ < V-$ ，比较器输出低电平。比较器输出端内置一个延时控制模块，延迟输出结果并滤除输出转换条件。延时有 16-step，包括无延时、 $2/F_{cpu}$ 、 $4/F_{cpu}$ 、 $6/F_{cpu}$ 、 $8/F_{cpu}$ 、 $10/F_{cpu}$ 、 $14/F_{cpu}$ 、 $16/F_{cpu}$ 、 $18/F_{cpu}$ 、 $20/F_{cpu}$ 、 $22/F_{cpu}$ 、 $24/F_{cpu}$ 、 $26/F_{cpu}$ 、 $28/F_{cpu}$ 和 $30/F_{cpu}$ ，由 CM2D[3:0]控制。

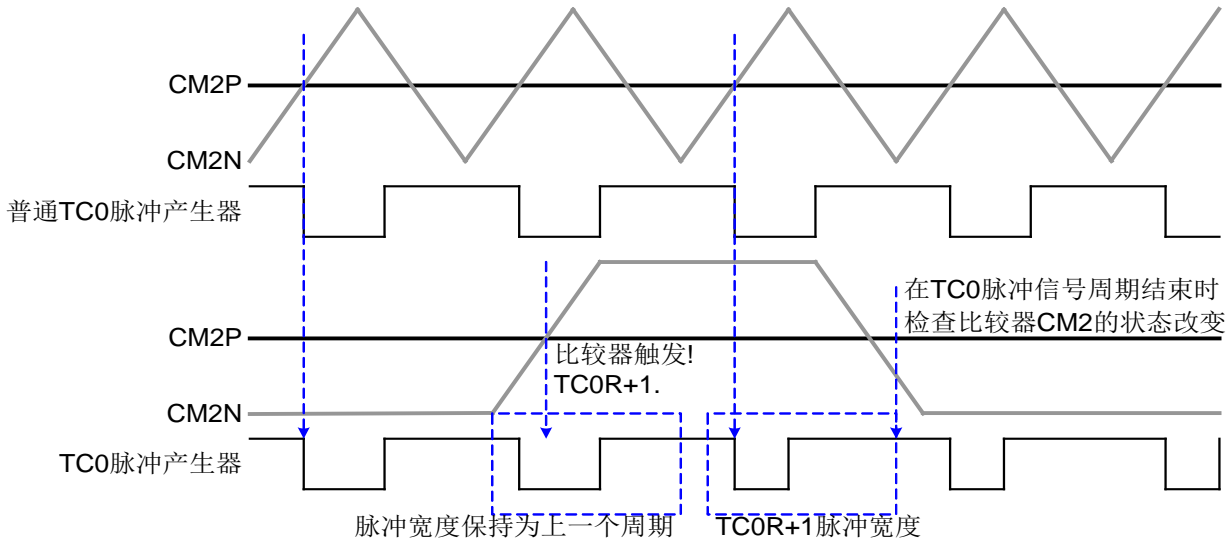
CM2D[2:0]	0000b	0001b	0010b	0011b	0100b	0101b	0110b	0111b
	No	$2/F_{cpu}$	$4/F_{cpu}$	$6/F_{cpu}$	$8/F_{cpu}$	$10/F_{cpu}$	$12/F_{cpu}$	$14/F_{cpu}$
延迟时间(us) $F_{cpu}=F_{osc}/4$ $=16MHz/4=4MHz$	0	0.5	1	1.5	2	2.5	3	3.5
延迟时间(us) $F_{cpu}=F_{osc}/16$ $=16MHz/16=1MHz$	0	2	4	6	8	10	12	14
CM2D[2:0]	1000b	1001b	1010b	1011b	1100b	1101b	1110b	1111b
	$16/F_{cpu}$	$18/F_{cpu}$	$20/F_{cpu}$	$22/F_{cpu}$	$24/F_{cpu}$	$26/F_{cpu}$	$28/F_{cpu}$	$30/F_{cpu}$
延迟时间(us) $F_{cpu}=F_{osc}/4$ $=16MHz/4=4MHz$	4	4.5	5	5.5	6	6.5	7	7.5
延迟时间(us) $F_{cpu}=F_{osc}/16$ $=16MHz/16=1MHz$	16	18	20	22	24	26	28	30



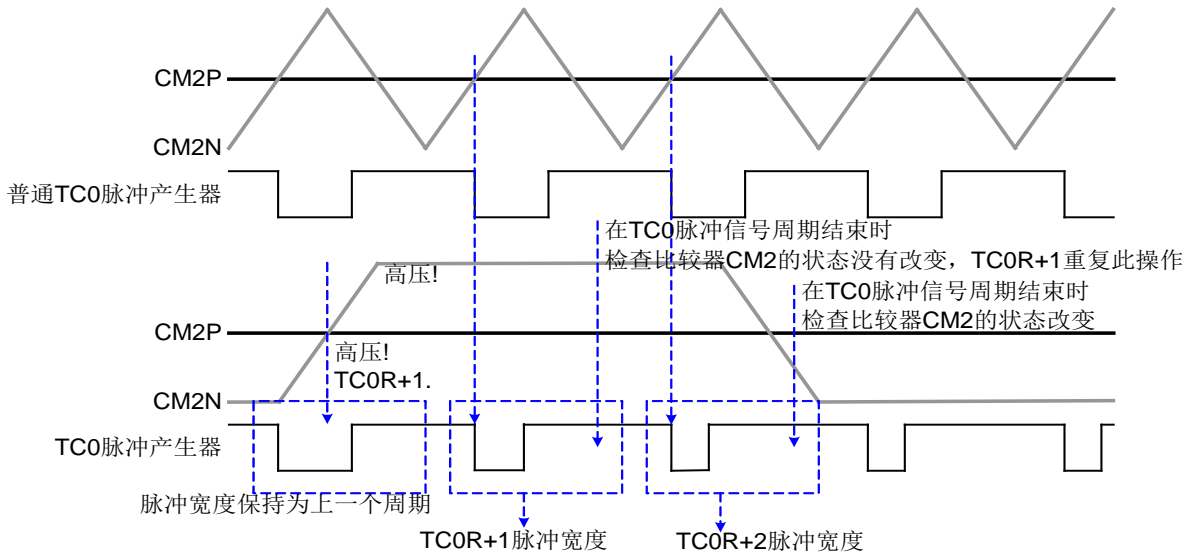
11.3 比较器CM2 特殊功能

除了普通比较器功能之外，CM2 还内置一个特殊功能，通过增加 TC0R 寄存器的值，缩短 TC0 脉冲宽度。此功能通过比较器 CM2 输出沿触发，增加寄存器 TC0R 的值，TC0 脉冲宽度被缩短，由 CM2SF 位控制。CM2SF=1 时，使能 CM2 特殊功能。若 CM2 的输出满足停止特殊功能的触发条件，则寄存器 TC0R 的值加 1，缩短 TC0 脉冲宽度。

- 步骤 1：当触发比较器 CM2 时，TC0R+1 立即缩短 TC0 的脉冲宽度，TC0 的脉冲宽度每次自动缩减一个单位。在 TC0 脉冲信号周期结束时，硬件检查比较器 CM2 的高压状态，若比较器 CM2 的输出状态改变，则通过程序修改 TC0R 的值改变 TC0 的脉冲宽度。



- 步骤 2：如果比较器 CM2 的输出状态没有改变，TC0R+1 位于 TC0 脉冲信号周期的结束处，输出一个新的脉冲直到比较器 CM2 输出状态改变。



* 注：如果 TC0R 增加至 0FFH，TC0R 的值一直保存为 0FFH，不再增加，即使比较器 CM2 的输出状态没有任何改变。

11.4 比较器CM2 模式寄存器

09EH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CM2M	CM2EN	CM2OEN	CM2OUT	CM2SF	CM2G	CM2RS2	CM2RS1	CM2RS0
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 [2:0] **CM2RS[2:0]**: 比较器 CM2 正极电压选择控制位。

- 000 = CM2P 引脚为比较器正极输入引脚，GPIO 功能被屏蔽；
- 001 = 内部 0.2*Vdd，CM2P 为 GPIO 引脚；
- 010 = 内部 0.3*Vdd，CM2P 为 GPIO 引脚；
- 011 = 内部 0.4*Vdd，CM2P 为 GPIO 引脚；
- 100 = 内部 0.5*Vdd，CM2P 为 GPIO 引脚；
- 101 = 内部 0.6*Vdd，CM2P 为 GPIO 引脚；
- 110 = 内部 0.7*Vdd，CM2P 为 GPIO 引脚；
- 111 = 内部 0.8*Vdd，CM2P 为 GPIO 引脚。

Bit 3 **CM2G**: 比较器 CM2 输出触发方向选择控制位。

- 0 = 下降沿触发，CM2P < CM2N，比较器输出状态由高变低；
- 1 = 上升沿触发，CM2P > CM2N，比较器输出状态由低变高。

Bit 4 **CM2SF**: 比较器 CM2 特殊模式控制位。

- 0 = 禁止，CM2 为普通比较器模式；
- 1 = 使能，CM2 输出沿触发缩短 TC0 脉冲宽度。

Bit 5 **CM2OUT**: 比较器 CM2 输出标志位。

- 0 = CM2P 电压小于 CM2N 电压；
- 1 = CM2P 电压大于 CM2N 电压。

Bit 6 **CM2OEN**: 比较器 CM2 输出引脚控制位。

- 0 = 禁止，CM2O 为 GPIO 引脚；
- 1 = 使能，CM2O 为比较器输出引脚，隔离 GPIO 功能。

Bit 7 **CM1EN**: 比较器 CM2 控制位。

- 0 = 禁止，比较器引脚为 GPIO 引脚；
- 1 = 使能，CM2N 引脚为比较器引脚，CM2O 由 CM2OEN 为控制，CM2P 引脚由 CM2RS[2:0]控制。

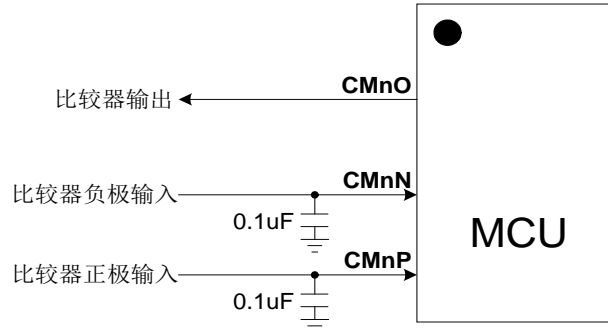
09BH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CMDB1	-	-	-	-	CM2D3	CM2D2	CM2D1	CM2D0
读/写	-	-	-	-	R/W	R/W	R/W	R/W
复位后	-	-	-	-	0	0	0	0

Bit [7:4] **CM2D[3:0]**: 比较器 CM2 de-bounce 时间控制位。

- 0000=没有延时；0001=2/Fcpu；0010=4/Fcpu；0011=6/Fcpu；0100=8/Fcpu；0101=10/Fcpu；0110=12/Fcpu；
- 0111=14/Fcpu；1000=16/Fcpu；1001=18/Fcpu；1010=20/Fcpu；1011=22/Fcpu；1100=24/Fcpu；
- 1101=26/Fcpu；1110=28/Fcpu；1111=30/Fcpu。

11.5 比较器应用注意事项

比较器是指比较正极电压和负极电压并将结果输出，正负极的信号都是模拟信号。在硬件应用电路中，比较器的输入引脚必须连接一个 0.1uF 的电容以减少电源干扰，确保输入信号的稳定。应用电路如下所示：



11.6 比较器CM2 操作举例

● 比较器 CM2:

; 复位比较器 CM2。

```
CLR          CM2M          ; 清 CM2M。
```

; 设置比较器 CM2 正极输入端。

```
MOV          A, #0000nnnb    ; 通过 CM2RS[2:0]设置比较器 CM2 正极输入端。
B0MOV        CM2M, A
```

; 设置比较器 CM2 功能模式。

```
B0BCLR        FCM2SF          ; 普通比较器模式。
```

; or

```
B0BSET        FCM2SF          ; 特殊功能模式。
```

; 设置比较器 CMW 输出引脚。

```
B0BCLR        FCM2OEN        ; 禁止。
```

; or

```
B0BSET        FCM2OEN        ; 使能。
```

; 设置比较器 CM2 中断触发沿。

```
B0BCLR        FCM2G          ; 下降沿。
```

; or

```
B0BSET        FCM2G          ; 上升沿。
```

; 设置比较器 CM2 输出 de-bounce。

```
MOV          A, #0000nnnnb    ; 通过 CM2D[3:0]设置比较器 CM2 输出 de-bounce。
B0MOV        CMDB1, A
```

; 清 CM2IRQ。

```
B0BCLR        FCM2IRQ
```

; 使能比较器 CM2 和中断功能。

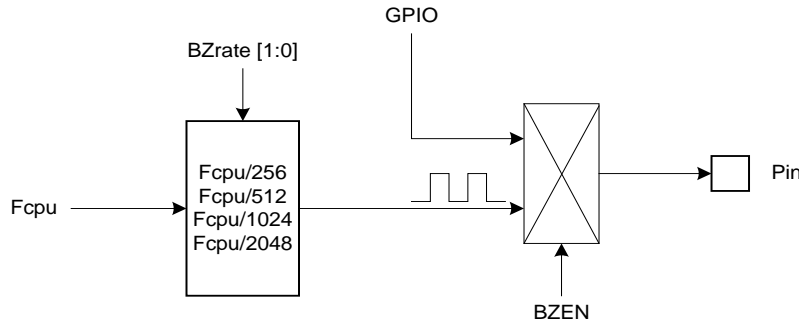
```
B0BSET        FCM2IEN        ; 使能比较器 CM2 中断功能。
```

```
B0BSET        FCM2EN          ; 使能比较器 CM2。
```

12 2K/4K BUZZER产生器

12.1 概述

SN8P2743/42 内置 Buzzer 产生器，用来驱动外部 buzzer 装置，可以驱动 2KHz 或者 4KHz buzzer。通过 BZM 寄存器调整 buzzer 输出频率。Buzzer 输出引脚与 GPIO 引脚共用，BZEN=1 时，该引脚输出 buzzer 载波信号；BZEN=0 时，该引脚返回到上一个 GPIO 模式（输入模式、输出高或输出低模式）。



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

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

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

12.2 BZM寄存器

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

Bit 7 **BZEN**: Buzzer 输出控制位。

- 0 = 禁止 BZ 输出，BZ 输出引脚返回到上一个 GPIO 模式；
- 1 = 使能 BZ 输出，禁止 BZ 输出引脚的 GPIO 功能。

Bit[6:5] **BZrate[1:0]**: Buzzer rate 控制位。

- 00 = Fcpu/256
- 01 = Fcpu/512
- 10 = Fcpu/1024
- 11 = Fcpu/2048

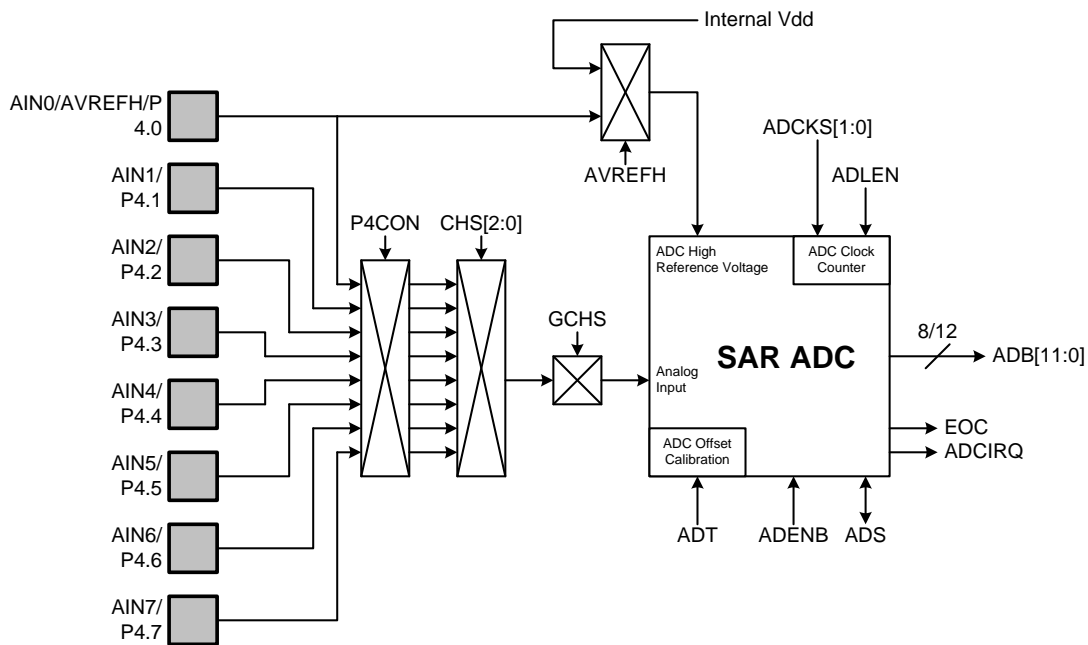
* 注：

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

13 8 通道模拟数字转换 (ADC)

13.1 概述

模拟数字转换 (ADC) 是一个 SAR 结构, 内置 8 个模拟通道 (AIN0~AIN7), 高达 4096 阶的分辨率, 能将一个模拟信号转换成相应的 12 位数字信号。ADC 内置的 8 个模拟通道可以测量 8 种不同的模拟信号源, 由 CHS[2:0] 和 GCHS 位控制。通过 ADLEN 位可以选择 ADC 的分辨率为 8 位或者 12 位; 可以通过 ADCKS[1:0] 位选择 ADC 的转换速率以决定 ADC 的转换时间。ADC 参考电压的高电平有两种提供方式, 由 AVREFH 位决定: 其中一种是内部 VDD (AVREFH=0), 另外一种是由 P4.0 输入的外部参考源 (AVREFH=1)。ADC 内置 P4CON 寄存器来设置模拟输入引脚, 必须由程序将 P4 设置为不带上拉电阻的输入引脚。设置好 ADENB 和 ADS 位后, ADC 开始转换, 转换结束时, ADC 电路将 EOC 和 ADCIRQ 置 1, 并将转换结果存入 ADB 和 ADR 寄存器中。若 ADCIEN=1, ADC 请求中断, AD 转换完成后, ADCIRQ=1 时, 程序计数器跳至中断向量地址执行中断服务程序。中断时必须由程序将 ADCIRQ 清零。



13.2 ADC模式寄存器

ADC 模式寄存器 ADM 设置 ADC 的相关配置：包括 ADC 启动，ADC 通道选择，ADC 参考电压高电平输入源和 ADC 处理状态显示等。必须在 AD 开始转换前将这些配置设置完毕。

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	AVREFH	CHS2	CHS1	CHS0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 7 **ADENB**: ADC 控制位。省电模式下，禁止 ADC 以省电。

- 0 = 禁止 ADC;
- 1 = 使能 ADC。

Bit 6 **ADS**: ADC 启动控制位。AD 转换完成后自动将 ADS 位清零。

- 0 = 停止 AD 转换;
- 1 = 开始 AD 转换。

Bit 5 **EOC**: ADC 状态位。

- 0 = AD 转换中;
- 1 = AD 转换结束，ADS 位复位。

Bit 4 **GCHS**: ADC 全局通道控制位。

- 0 = 禁止 AIN 通道;
- 1 = 使能 AIN 通道。

Bit 3 **AVREFH**: ADC 参考电压高电平输入控制位。

- 0 = 内部 VDD，P4.0 为 GPIO 或 AIN0 引脚;
- 1 = 由 P4.0 输入的外部参考电压。

Bit [2:0] **CHS[2:0]**: ADC 输入通道选择位。

- 000 = AIN0; 001 = AIN1; 010 = AIN2; 011 = AIN3; 100 = AIN4; 101 = AIN5; 110 = AIN6; 111 = AIN7。

ADR 寄存器包括 ADC 模式控制和 ADC 低字节数据缓存器，ADC 配置包括 ADC 时钟速率和 ADC 分辨率。必须在启动 ADC 之前设置好这些配置。

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	-	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0
读/写	-	R/W	R/W	R/W	R	R	R	R
复位后	-	0	0	0	-	-	-	-

Bit 6,4 **ADCKS [1:0]**: ADC 时钟 rate 选择位。

- 00 = Fcpu/16; 01 = Fcpu/8; 10 = Fcpu/1; 11 = Fcpu/2。

Bit 5 **ADLEN**: ADC 分辨率选择位。

- 0 = 8 位;
- 1 = 12 位。

13.3 ADC数据缓存寄存器

ADC 数据缓存器共 12 位，用来存储 AD 转换结果，ADB 存放结果的高字节（bit4~bit11），ADR（ADR[3:0]）存放低字节（bit0~bit3）。ADC 数据缓存器是只读寄存器，系统复位后处于未知状态。

- **ADB[11:4]: 8 位 ADC 模式下，ADC 数据存放于 ADB 寄存器中。**
- **ADB[11:0]: 12 位 ADC 模式下，ADC 数据存放于 ADB 和 ADR 寄存器中。**

0B2H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADB	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4
读/写	R	R	R	R	R	R	R	R
复位后	-	-	-	-	-	-	-	-

Bit[7:0] **ADB[7:0]: 8 位 ADC 数据缓存器，存放 12 位 ADC 的高字节数据。**

0B3H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADR	-	ADCKS1	ADLEN	ADCKS0	ADB3	ADB2	ADB1	ADB0
读/写	-	R/W	R/W	R/W	R	R	R	R
复位后	-	0	0	0	-	-	-	-

Bit [3:0] **ADB [3:0]: 12 位 ADC 的低字节数据缓存器。**

AIN 输入电压 v.s. ADB 输出数据

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

针对不同的应用，用户可能需要精度介于 8 位到 12 位之间的 AD 转换器。对于这种情况，可以通过对保存在 ADR 和 ADB 中的转换结果进行处理得到。首先，用户必须选择 12 位分辨率的模式，进行 AD 转换，然后在转换结果中去掉最低的几位得到需要的结果。如下表所示：

ADC 分辨率	ADB								ADR			
	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
8-bit	0	0	0	0	0	0	0	0	x	x	x	x
9-bit	0	0	0	0	0	0	0	0	0	x	x	x
10-bit	0	0	0	0	0	0	0	0	0	0	x	x
11-bit	0	0	0	0	0	0	0	0	0	0	0	x
12-bit	0	0	0	0	0	0	0	0	0	0	0	0

0 = 可选位, x = 无效位

* 注：ADC 数据缓存器包括 ADB 和 ADR 低字节，系统复位后，ADC 数据缓存器的值是未知的。

13.4 ADC操作说明和注意事项

13.4.1 ADC信号格式

ADC 采样电压范围为参考电压高/低电平之间，ADC 参考低电压为 VSS，高电压有两种，由 AVREFH 为决定：内部 VDD 和 P4.0/AVREFH 输入的外部参考电压。AVREFH=0 时，ADC 参考电压由内部 VDD（单片机电源电压）提供；AVREFH=1 时，ADC 参考电压由外部参考源提供。ADC 参考电压的范围为：(ADC 参考高电压-ADC 参考低电压) \geq 2V，ADC 参考低电压为 VSS=0V，故 ADC 参考高电压范围为 2V~VDD，外部参考电压需在此范围之内。

- ADC 内部参考低电压=0V。
- ADC 内部参考高电压=VDD (AVREFH=0)。
- ADC 外部参考电压=2V~VDD (AVREFH=1)。

ADC 采样输入信号电压必须在 ADC 参考低电压和 ADC 参考高电压之间，若 ADC 输入信号的电压不在此范围内，则 ADC 的转换结果会出错（满量程或者为 0）。

$$\text{ADC 参考低电压} \leq \text{ADC 采用输入信号电压} \leq \text{ADC 参考高电压}$$

13.4.2 AD转换时间

ADC 转换时间是指从 ADS=1（开始 ADC）到 EOC=1（ADC 结束）所用的时间，由 ADC 分辨率和 ADC 时钟 Rate 控制，12 位 ADC 的转换时间为 $1/(\text{ADC 时钟}/4) * 16 \text{ S}$ ；8 位 ADC 的转换时间为 $1/(\text{ADC 时钟}/4) * 12 \text{ S}$ 。ADC 的时钟源为 Fcpu，包括 Fcpu/1，Fcpu/2，Fcpu/8，Fcpu/16，由 ADCKS[1:0]位控制。

ADC 的转换时间会影响 ADC 的性能，如果输入高 Rate 的模拟信号，必须要选择一个高 Rate 的 ADC 转换 Rate。如果 ADC 的转换时间比模拟信号的转换 Rate 慢，则 ADC 的结果出错。故选择合适的 ADC 时钟 Rat 和 ADC 分辨率才能得到合适的 ADC 转换 Rate。

$$\text{12 位 ADC 转换时间} = 1/(\text{ADC 时钟 Rate}/4) * 16 \text{ sec}$$

ADLEN	ADCKS1, ADCKS0	ADC 时钟 Rate	Fcpu=4MHz		Fcpu=16MHz	
			AD 转换时间	AD 转换 Rate	AD 转换时间	AD 转换 Rate
1 (12-bit)	00	Fcpu/16	$1/(4\text{MHz}/16/4) * 16 = 256 \text{ us}$	3.906KHz	$1/(16\text{MHz}/16/4) * 16 = 64 \text{ us}$	15.625KHz
	01	Fcpu/8	$1/(4\text{MHz}/8/4) * 16 = 128 \text{ us}$	7.813KHz	$1/(16\text{MHz}/8/4) * 16 = 32 \text{ us}$	31.25KHz
	10	Fcpu	$1/(4\text{MHz}/4) * 16 = 16 \text{ us}$	62.5KHz	$1/(16\text{MHz}/4) * 16 = 4 \text{ us}$	250KHz
	11	Fcpu/2	$1/(4\text{MHz}/2/4) * 16 = 32 \text{ us}$	31.25KHz	$1/(16\text{MHz}/2/4) * 16 = 8 \text{ us}$	125KHz

$$\text{8 位 ADC 转换时间} = 1/(\text{ADC 时钟 Rate}/4) * 12 \text{ sec}$$

ADLEN	ADCKS1, ADCKS0	ADC 时钟 Rate	Fcpu=4MHz		Fcpu=16MHz	
			AD 转换时间	AD 转换 Rate	AD 转换时间	AD 转换 Rate
0 (8-bit)	00	Fcpu/16	$1/(4\text{MHz}/16/4) * 12 = 192 \text{ us}$	5.208KHz	$1/(16\text{MHz}/16/4) * 12 = 48 \text{ us}$	20.833KHz
	01	Fcpu/8	$1/(4\text{MHz}/8/4) * 12 = 96 \text{ us}$	10.416KHz	$1/(16\text{MHz}/8/4) * 12 = 24 \text{ us}$	41.667KHz
	10	Fcpu	$1/(4\text{MHz}/4) * 12 = 12 \text{ us}$	83.333KHz	$1/(16\text{MHz}/4) * 12 = 3 \text{ us}$	333.333KHz
	11	Fcpu/2	$1/(4\text{MHz}/2/4) * 12 = 24 \text{ us}$	41.667KHz	$1/(16\text{MHz}/2/4) * 12 = 6 \text{ us}$	166.667KHz

13.4.3 ADC引脚配置

ADC 输入引脚与 P4 口共用，ADC 输入通道的选择由 ADCCHS[2:0]控制，ADCCHS[2:0]=000 时选择 AIN0，ADCCHS[2:0]=001 时选择 AIN1.....同一时间设置 P4 口的一个引脚作为 ADC 的输入引脚，该引脚必须设置为输入引脚，禁止内部上拉，并首先由程序使能 P4CON 寄存器。通过 ADCCHS[2:0]选择好 ADC 输入通道后，GCHS 置 1 以使能 ADC 功能。

- ADC 输入引脚为 GPIO 引脚时必须设为输入模式。
- 必须禁止 ADC 输入引脚的内部上拉电阻。
- ADC 输入通道的 P4CON 位必须置 1。

AVREFH=1 时，P4.0/AIN0 可以作为 ADC 外部参考源的输入引脚，此时，该引脚必须设为输入模式，且禁止内部上拉。

- ADC 外部参考电压的输入引脚必须设为输入模式。
- ADC 外部参考电压的输入引脚必须禁止内部上拉电阻。

ADC 输入引脚与普通 I/O 引脚共用。当输入一个模拟信号到 CMOS 结构端口时，尤其当模拟信号为 $1/2 V_{DD}$ 时，可能产生额外的漏电流。当 P4 输入多个模拟信号时，也会产生额外的漏电流。睡眠模式下，上述漏电流会严重影响到系统的整体功耗。P4CON 为 P4 口的配置寄存器，将 P4CON[7:0]置 1，其对应的 P4 引脚将被设为纯模拟信号输入引脚，从而避免上述漏电流的产生。

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

Bit[7:0] **P4CON[7:0]**: P4.n 配置控制位。

- 0 = P4.n 可以作为模拟输入（ADC 输入）引脚或者 GPIO 引脚；
- 1 = P4.n 只能作为模拟输入引脚，不能作为 GPIO 引脚。

* 注：P4.n 作为 GPIO 引脚而不是 ADC 输入引脚时，P4CON.n 必须置 0，否则 P4.n 的普通 I/O 信号会被隔离。

13.5 ADC操作举例

● ADC:

; 复位 ADC。

CLR ADM ; 清 ADM 寄存器。

; 设置 ADC 时钟 Rate 和 ADC 分辨率。

MOV A, #0nmn0000b ; nn:ADCKS[1:0]代表 ADC 时钟 Rate。
B0MOV ADR, A ; m 代表 ADC 分辨率。

; 设置 ADC 参考高电压。

B0BCLR FAVREFH ; 内部 VDD。

or

B0BSET FAVREFH ; 外部参考源。

; 设置 ADC 输入通道。

MOV A, #value1 ; 设置 P4CON 选择 ADC 输入通道。

B0MOV P4CON, A

MOV A, #value2 ; 设置 ADC 输入通道为输入模式。

B0MOV P4M, A

MOV A, #value3 ; 禁止 ADC 输入通道的内部上拉电阻。

B0MOV P4UR, A

; 使能 ADC。

B0BSET FADCENB

; 执行 ADC 100us 启动时间延迟循环。

CALL 100usDLY ; 100us 延迟循环。

; 选择 ADC 输入通道。

MOV A, #value ; 设置 ADCHS[2:0]选择 ADC 输入通道。

OR ADM, A

; 使能 ADC 输入通道。

B0BSET FGCHS

; 使能 ADC 中断功能。

B0BCLR FADCIRQ ; 清 ADC 中断请求。

B0BSET FADCIE ; 使能 ADC 中断功能。

; 开始 AD 转换。

B0BSET FADS

* 注:

1、使能 ADENB 后（不是使能 ADS），系统必须延迟 100us 等待启动 ADC，然后设置 ADS 开始 AD 转换，否则 ADC 的结果出错。系统正常运行时，设置 ADENB 一次，延时一次。

2、睡眠模式和绿色模式下禁止 ADC 以省电。

● ADC 转换:

; 禁止 ADC 中断。

@@:

```

B0BTS1      FE0C      ; 检查 ADC 转换标志。
JMP          @B      ; EOC=0: ADC 转换过程中。
B0MOV       A, ADB   ; EOC=1: ADC 转换结束, 处理 ADC 结果。
B0MOV       BUF1,A
MOV         A, #00001111b
AND        A, ADR
B0MOV       BUF2,A
...
CLR         FE0C     ; ADC 结果处理完成。
                ; 清 ADC 转换标志开始下个 ADC 转换。

```

; 使能 ADC 中断。

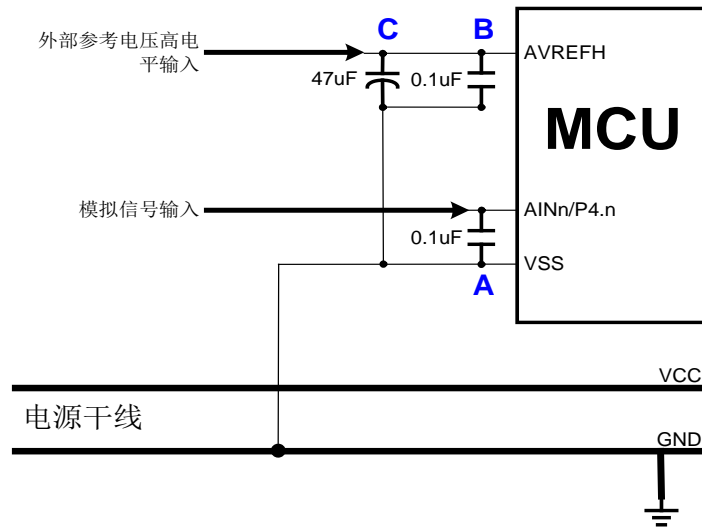
```

INT_SR:      ORG 8      ; 中断向量。
                ; 中断服务程序。
                PUSH
B0BTS1      FADCIRQ    ; 检查 ADC 中断请求标志。
JMP         EXIT_INT  ; ADCIRQ=0: 无 ADC 中断请求。
B0MOV       A, ADB   ; ADCIRQ=1: ADC 转换结束, 处理 ADC 结果。
B0MOV       BUF1,A
MOV         A, #00001111b
AND        A, ADR
B0MOV       BUF2,A
...
CLR         FE0C     ; ADC 结果处理完成。
                ; 清 ADC 转换标志开始下个 ADC 转换。
JMP         INT_EXIT
INT_EXIT:    POP
                RETI   ; 退出中断。

```

* 注: ADC 转换结束后, 自动清 ADS。EOC 实时显示 ADC 的转换过程, ADS=1 时被自动清零。用户不需要在程序中清 EOC。

13.6 ADC应用电路

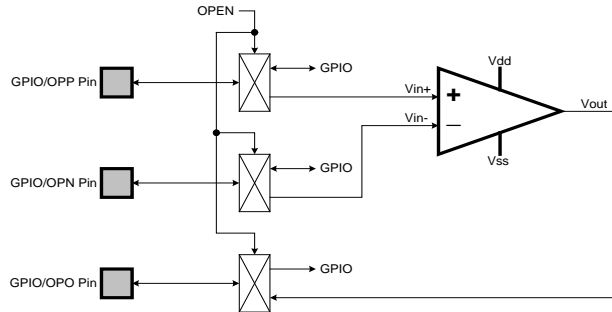


模拟信号从 ADC 输入引脚 AINn/P4.n 输入。在 ADC 输入引脚和 VSS 之间 (A) 必须连接一个 0.1uF 的电容，且要尽可能的靠近 ADC 输入引脚。不能将电容的 GND 直接连接到电源干线上的 GND，必须通过 VSS 引脚。该电容可以减少电源干扰对模拟信号的影响。

如果 ADC 参考高电压由外部参考源提供，则必须由 AVREFH/P4.0 引脚输入。且在 AVREFH 引脚和 VSS 之间连接电容，首先在图中 C 处连接一个 47uF 的电解电容，再在 B 处连接一个 0.1uF 的电容，且要尽可能的靠近 AVREFH 引脚。不能将电容的 GND 直接连接到电源干线上的 GND，必须通过 VSS 引脚。

14 RAIL to RAIL OP放大器

14.1 概述



SN8P2743/42 内置 1 个 Rail-to-Rail 结构的 OP 放大器，即输入/输出电压的实际值在 VDD~VSS 之间。

OP 放大器的引脚与 GPIO 引脚共用，由 OPEN 位控制。OPEN=0 时，OP AMP 引脚为 GPIO 模式，OPEN=1 时，GPIO 引脚切换到 OP 放大器引脚并隔离 GPIO 功能。OP 引脚功能的选择列表如下所示：

OP No.	OPEN	OP 正极输入引脚	OP 负极输入引脚	OP 输出引脚
OP	OPEN=0	所有引脚都为 GPIO 模式，OP AMP 引脚功能被禁止。		
	OPEN=1	OPP (Vin+)	OPN (Vin-)	OPO (Vout)

* 注：如果禁止 OP-AMP，OP-AMP 引脚都返回到上一个 GPIO 模式。

14.2 OP AMP寄存器

09FH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPM	-	-	-	-	-	-	-	OPEN
读/写	-	-	-	-	-	-	-	R/W
复位后	-	-	-	-	-	-	-	0

Bit 0 **OPEN**: OP Amp 控制位。

0 = 禁止，P1.0、P1.1、P1.2 为 GPIO 模式；

1 = 使能，P1.0、P1.1、P1.2 为 OP-AMP 引脚。

15 指令集

指令	指令格式	指令说明	C	DC	Z	周期
MOV O V E	MOV A,M	$A \leftarrow M$	-	-	√	1
	MOV 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 指工作寄存器 R、Y、Z、RBANK 和 PFLAG 等。)	-	-	-	1
	XCH A,M	$A \leftrightarrow M$	-	-	-	1+N
	B0XCH A,M	$A \leftrightarrow M$ (bank 0)	-	-	-	1+N
	MOV C	$R, A \leftarrow ROM [Y,Z]$	-	-	-	2
	A R I T H M E T I C	ADC A,M	$A \leftarrow A + M + C$, 如果产生进位, 则 C=1, 否则 C=0。	√	√	√
ADC 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
ADD 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
SBC M,A		$M \leftarrow A - M - /C$, if occur borrow, then C=0, else 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
SUB 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
DAA		将 ACC 中的数据由十六进制转换成十进制格式。	√	-	-	1
MUL A,M	$R, A \leftarrow A * M$, 结果的 LB 存入 ACC, HB 存入 R 寄存器, ZF 受 ACC 的影响。	-	-	√	2	
L O G I C	AND A,M	$A \leftarrow A$ 与 M	-	-	√	1
	AND M,A	$M \leftarrow A$ 与 M	-	-	√	1+N
	AND A,I	$A \leftarrow A$ 与 I	-	-	√	1
	OR A,M	$A \leftarrow A$ 或 M	-	-	√	1
	OR 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
	XOR M,A	$M \leftarrow A$ 异或 M	-	-	√	1+N
	XOR A,I	$A \leftarrow A$ 异或 I	-	-	√	1
P R O C E S S	SWAP M	A (b3~b0, b7~b4) $\leftarrow M$ (b7~b4, b3~b0)	-	-	-	1
	SWAPM M	M (b3~b0, b7~b4) $\leftarrow M$ (b7~b4, b3~b0)	-	-	-	1+N
	RRC M	$A \leftarrow RRC M$	√	-	-	1
	RRCM M	$M \leftarrow RRC M$	√	-	-	1+N
	RLC M	$A \leftarrow RLC M$	√	-	-	1
	RLCM M	$M \leftarrow RLC M$	√	-	-	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 (bank 0).b $\leftarrow 0$	-	-	-	1+N
B0BSET M.b	M (bank 0).b $\leftarrow 1$	-	-	-	1+N	
B R A N C H	CMPRS A,I	ZF,C $\leftarrow A - I$, 如果 A=I, 跳转到下一条指令。	√	-	√	1+S
	CMPRS A,M	ZF,C $\leftarrow A - M$, 如果 A=M, 跳转到下一条指令。	√	-	√	1+S
	INCS M	$A \leftarrow M + 1$, 如果 A=0, 跳转到下一条指令。	-	-	-	1+S
	INCMS M	$M \leftarrow M + 1$, 如果 M=0, 跳转到下一条指令。	-	-	-	1+N+S
	DECS M	$A \leftarrow M - 1$, 如果 A=0, 跳转到下一条指令。	-	-	-	1+S
	DECMS M	$M \leftarrow M - 1$, 如果 M=0, 跳转到下一条指令。	-	-	-	1+N+S
	BTS0 M.b	如果 M.b=0, 跳转到下一条指令。	-	-	-	1+S
	BTS1 M.b	如果 M.b=1, 跳转到下一条指令。	-	-	-	1+S
	B0BTS0 M.b	如果 M(bank 0).b=0, 跳转到下一条指令。	-	-	-	1+S
	B0BTS1 M.b	如果 M(bank 0).b=1, 跳转到下一条指令。	-	-	-	1+S
	JMP d	跳转指令, $PC15/14 \leftarrow RomPages1/0$, $PC13-PC0 \leftarrow d$	-	-	-	2
	CALL d	子程序调用指令, $Stack \leftarrow PC15-PC0$, $PC15/14 \leftarrow RomPages1/0$, $PC13-PC0 \leftarrow d$	-	-	-	2
M I S C	RET	子程序跳出指令, $PC \leftarrow Stack$	-	-	-	2
	RETI	中断程序跳出指令, $PC \leftarrow Stack$, 使能全局中断。	-	-	-	2
	PUSH	保存工作寄存器。	-	-	-	1
	POP	恢复工作寄存器。	√	√	√	1
	NOP	空指令。	-	-	-	1

注: 1. M 指系统寄存器或者 RAM, 如果 M 指系统寄存器, 则 N=0, 否则 N=1。
2. 条件跳转指令中, 满足条件则 S=1, 否则 S=0。

16 电气特性

16.1 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)	
SN8P2743K, SN8P2743S, SN8P2742P, SN8P2742S, SN8P27411P, SN8P27411S.....	0°C ~ + 70°C
SN8P2743DK, SN8P2743DS, SN8P2742DP, SN8P2742DS, SN8P27411PD, SN8P27411SD.....	-40°C ~ + 85°C
Storage ambient temperature (Tstor)	-40°C ~ + 125°C

16.2 电气特性

● DC CHARACTERISTIC

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

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT	
Operating voltage	Vdd	Normal mode. Fcpu = 1MHz	2.2	-	5.5	V	
		Normal mode. Fcpu = 4MHz	2.4	-	5.5	V	
RAM Data Retention voltage	Vdr		1.5	-	-	V	
*Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset	0.05	-	-	V/ms	
Input Low Voltage	ViL1	All input ports	Vss	-	0.3Vdd	V	
	ViL2	Reset pin	Vss	-	0.2Vdd	V	
Input High Voltage	ViH1	All input ports	0.7Vdd	-	Vdd	V	
	ViH2	Reset pin	0.8Vdd	-	Vdd	V	
Reset pin leakage current	Ilekg	Vin = Vdd	-	-	2	uA	
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd	-	-	2	uA	
I/O port pull-up resistor	Rup	Vin = Vss, Vdd = 3V	100	200	300	KΩ	
		Vin = Vss, Vdd = 5V	50	100	150		
I/O output source current	IoH	Vop = Vdd - 0.5V	8	-	-	mA	
	IoL	Vop = Vss + 0.5V	8	-	-		
*INTn trigger pulse width	Tint0	INT0 interrupt request pulse width	2/fcpu	-	-	cycle	
Supply Current (Disable ADC, OP-amp, Comparator)	Idd1	Run Mode (No loading)	Vdd= 3V, Fcpu = 4MHz	-	2	-	mA
			Vdd= 5V, Fcpu = 4MHz	-	4	-	mA
			Vdd= 3V, Fcpu = 1MHz	-	1.5	-	mA
			Vdd= 5V, Fcpu = 1MHz	-	3	-	mA
			Vdd= 3V, Fcpu = 32KHz	-	20	-	uA
			Vdd= 5V, Fcpu = 32KHz	-	45	-	uA
	Idd2	Slow Mode	Vdd= 3V, ILRC=16KHz	-	3.5	-	uA
			Vdd= 5V, ILRC=32KHz	-	10	-	uA
	Idd3	Sleep Mode	Vdd= 5V/3V	-	1	2	uA
	Idd4	Green Mode (No loading, Watchdog Disable)	Vdd= 3V, IHRC=16MHz	-	0.35	-	mA
			Vdd= 5V, IHRC=16MHz	-	0.55	-	mA
			Vdd= 3V, Ext. 32KHz X'tal	-	6	-	uA
			Vdd= 5V, Ext. 32KHz X'tal	-	18	-	uA
			Vdd= 3V, ILRC=16KHz	-	3	-	uA
Vdd= 5V, ILRC=32KHz			-	5.5	-	uA	
Internal High Oscillator Freq.	Fihrc	Internal High RC (IHRC)	25°C, Vdd=2.2V~ 5.5V Fcpu=Fhosc/4~Fhosc/16	15.68	16	16.32	MHz
			-40°C~85°C, Vdd=2.4V~ 5.5V Fcpu=Fhosc/4~Fhosc/16	15.2	16	16.8	MHz
LVD Voltage	Vdet0	Low voltage reset level. 25°C	1.9	2.0	2.1	V	
		Low voltage reset level. -40°C~85°C	1.8	2.0	2.3	V	
	Vdet1	Low voltage reset/indicator level. 25°C	2.3	2.4	2.5	V	
		Low voltage reset/indicator level. -40°C~85°C	2.2	2.4	2.7	V	
	Vdet2	Low voltage reset/indicator level. 25°C	3.5	3.6	3.7	V	
		Low voltage reset/indicator level. -40°C~85°C	3.3	3.6	3.9	V	

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

● **ADC CHARACTERISTIC**

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

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
AIN0 ~ AIN7 input voltage	Vani	Vdd = 5.0V	0	-	Avrefh	V
ADC reference Voltage	Vref		2	-	-	V
*ADC enable time	Tast	Ready to start convert after set ADENB = "1"	100	-	-	us
*ADC current consumption	I _{ADC}	Vdd=5.0V	-	0.6	-	mA
		Vdd=3.0V	-	0.4	-	mA
ADC Clock Frequency	F _{ADCLK}	VDD=5.0V	-	-	8M	Hz
		VDD=3.0V	-	-	5M	Hz
ADC Conversion Cycle Time	F _{ADCYL}	VDD=2.4V~5.5V	64	-	-	1/F _{ADCLK} K
ADC Sampling Rate (Set FADS=1 Frequency)	F _{ADSMP}	VDD=5.0V	-	-	125	K/sec
		VDD=3.0V	-	-	80	K/sec
Differential Nonlinearity	DNL	VDD=5.0V , AVREFH=3.2V, F _{ADSMP} =7.8K	±1	-	-	LSB
Integral Nonlinearity	INL	VDD=5.0V , AVREFH=3.2V, F _{ADSMP} =7.8K	±2	-	-	LSB
No Missing Code	NMC	VDD=5.0V , AVREFH=3.2V, F _{ADSMP} =7.8K	10	11	12	Bits
ADC offset Voltage	V _{ADCoffset}	Non-trimmed	-10	0	+10	mV
		Trimmed	-2	0	+2	mV

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

● **OP AMP CHARACTERISTIC**

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

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
*Supply Current	I _{op}	Unit Gain Buffer, Vdd= 3V, OPP = Vss.	-	130	-	uA
		Unit Gain Buffer, Vdd= 5V, OPP = Vss.	-	150	-	uA
Common Mode Input Voltage Range	V _{cmr}	Vdd=5.0V	Vss-0.3	-	Vdd+0.3	V
Input Offset Voltage	V _{os}	V _{cm} =V _{ss}	-3	-	+3	mV
Power Supply Rejection Ratio	PSRR	V _{cm} =V _{ss}	50	-	70	dB
Common Mode Rejection Ratio	CMRR	V _{cm} =-0.3V~2.5V. Vdd=5V.	50	-	80	dB
Open-Loop Gain (Large Signal)	A _{ol}	V _{out} =0.2V~Vdd-0.2V. V _{cm} =V _{ss} .	90	110	-	dB
Maximum Output Voltage Swing	V _{ol} , V _{oh}	0.5V output overdrive.	Vss+15	-	Vdd-15	V
Output Short Current	I _{sc}	Unit Gain Buffer. V _o = V _{ss} , Vdd=VPP=3V	-17	-	+17	mA
		Unit Gain Buffer. V _o = V _{ss} , Vdd=VPP=5V	-45	-	+45	mA
Output Slew Rate	T _{osr}	V _o = V _{ss} to Vdd, Vdd to V _{ss} .	3	-	5	us

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

● **COMPARATOR CHARACTERISTIC**

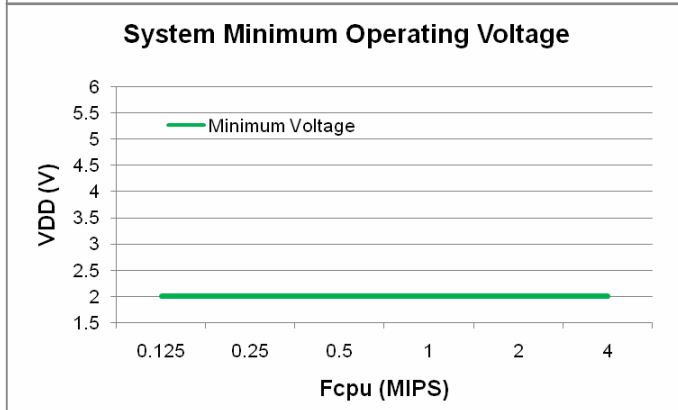
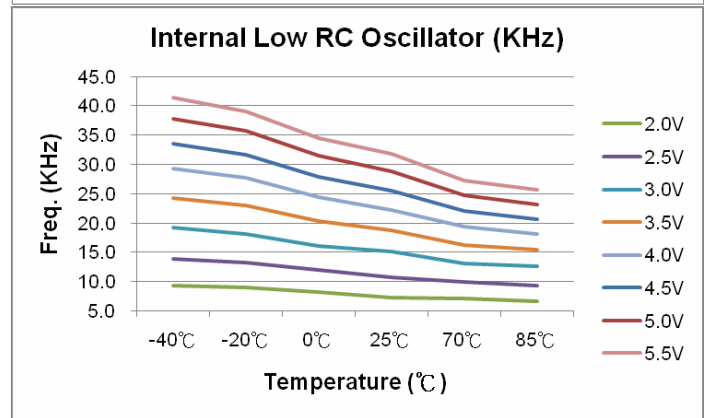
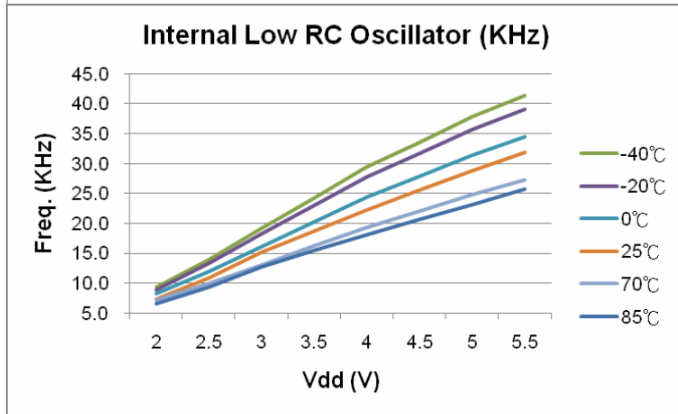
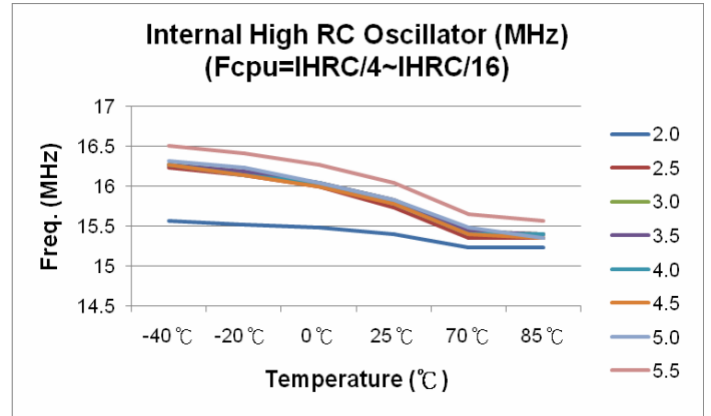
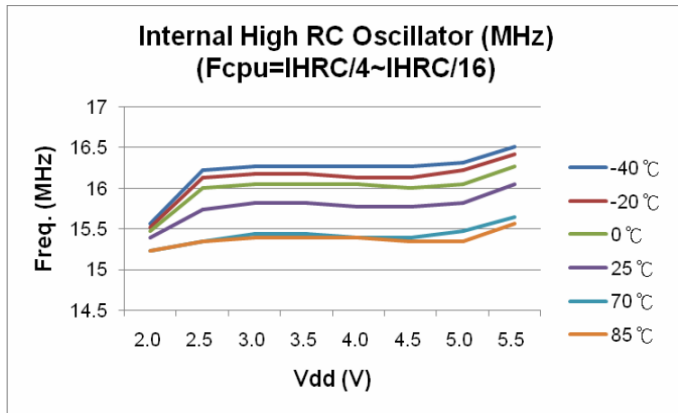
(All of voltages refer to Vss, Vdd = 5.0V, Fosc = 4MHz, Fcpu=1MHz, ambient temperature is 25°C unless otherwise note.)

PARAMETER	SYM.	DESCRIPTION	MIN.	TYP.	MAX.	UNIT
Power Supply Range	V _{cmp}		2.0		5.5	V
		I _{cm}			70	uA
Supply Current	I _{cm}	Vdd=3V. Internal reference disables.			70	uA
		Vdd=5V. Internal reference disables.			70	uA
Quiescent Current	I _q	I _{out} =0	0.3	0.6	1	uA
Input Offset Voltage	V _{os}	V _{cm} =V _{ss}	-5		+5	mV
Response Time	T _{rs}	Positive input voltage = 1/2*Vdd. Negative input voltage transitions from Vss to Vdd.			100	ns
Output Slew Rate	T _{osr}	Comparator output voltage Vdd=3V			100	ns
		transitions from Vss to Vdd. Vdd=5V			100	ns
		Comparator output voltage Vdd=3V			100	ns
		transitions from Vdd to Vss. Vdd=5V			100	ns
Common Mode Input Voltage Range	V _{cmr}	Vdd=5.0V	Vss+0.5		Vdd-0.5	V

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

16.3 特性曲线

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



17 开发工具

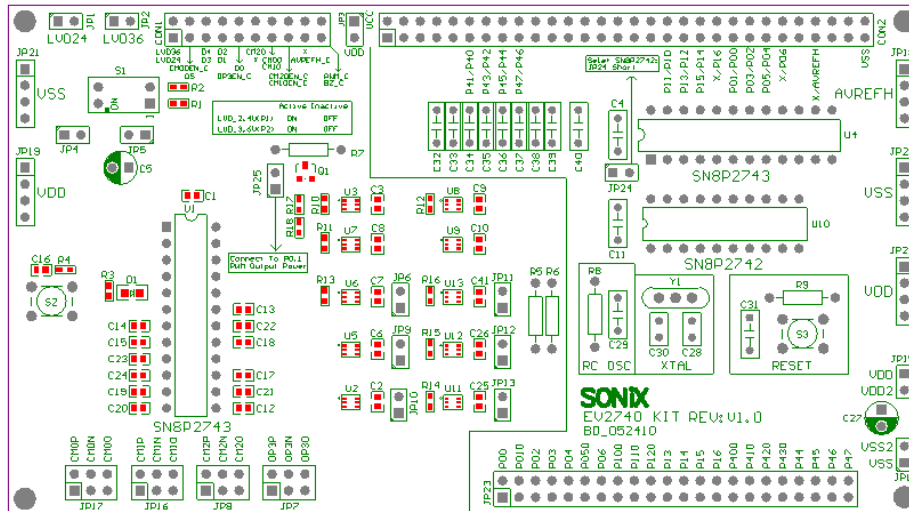
在进行 SN8P2743 的开发时，SONiX 提供 ICE（在线仿真器），IDE（集成开发环境）和 EV-Kit 开发工具。ICE 和 EV-Kit 为外部硬件装置，IDE 有一个友好的用户界面进行软件开发与仿真。各工具的版本如下所示：

- ICE: SN8ICE2K Plus 2。（在进行 IHRC 仿真时，请选择 16MHz 的晶振。）
- ICE 的最高仿真速度为：8MIPS @5V（如 16MHz 晶振， $F_{cpu} = F_{osc}/2$ ）
- EV-kit: EV2740 KIT REV: V1.0。
- IDE: SONiX IDE M2IDE_V129 或更晚的版本。
- Writer: MPiII writer。
- Writer 转接板: SN8P2742/SN8P2743。

17.1 SN8P2743 EV-KIT

SONiX 提供 SN8P2740 系列单片机的 EV-Kit 来仿真单片机的所有功能，包括 PWM，ADC，比较器和 OP 模拟功能，这些功能不能直接通过 SN8ICE2K Plus2 仿真，必须通过 SN8P2743 实际芯片进行仿真。实际芯片通过 EV-Kit 来进行多路 PWM 和模拟功能的仿真。EV Kit 还提供了 OP/比较器/PWM/ADC/LVD2.4V/3.6V 的电路切换。

SN8P2743 EV-Kit PCB 如下所示：



- CON1: 连接到 SN8ICE2K Plus2 JP3（EV-KIT 与 ICE 的通讯总线，控制信号及其他的信号）。
- CON2: 连接到 SN8ICE2K Plus2 CON1（包括 GPIO、EV-KIT 控制信号及其他的信号）。
- S1: LVD24V/LVD36V 控制开关。仿真 LVD2.4V 标志/复位功能和 LVD3.6V/标志功能。

开关 No.	ON	OFF
LVD24	LVD 2.4V 有效	LVD 2.4V 无效
LVD36	LVD 3.6V 有效	LVD 3.6V 无效

- S2: SN8P2743 EV-chip 复位按键。若 EV-KIT 激活无效，按下 S2 复位 EV-KIT 实际芯片（U1）。
- JP23: GPIO 接口。
- JP18: 使用 ADC 功能之前，必须断开 SN8ICE2K Plus2 AVREFH/VDD 之间的跳线。如果使能 ADC 的外部参考电压功能，JP18（AVREFH）或 P400 为外部参考电压的输入引脚。
- JP17: 监测 CMP0 输入/输出电压。
- JP16: 监测 CMP1 输入/输出电压。
- JP8: 监测 CMP2 输入/输出电压。
- JP7: 监测 OP-Amp 输入/输出电压（OP3P = OPP，OP3N = OPN，OP3O = OPO）。
- U1: SN8P2743 EV-chip 仿真模拟功能。

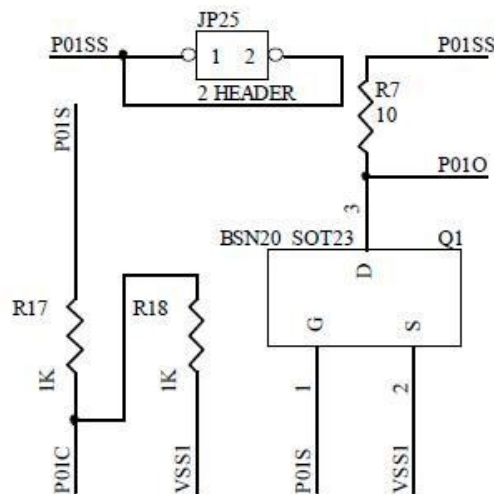
- U4: SN8P2743 DIP 封装形式的 IC 接口，连接到用户目标板。

VSS	1	U	24	VDD
XIN/P0.6	2		23	P4.7/AIN7
XOUT/P0.5/BZ	3		22	P4.6/AIN6
RST/VPP/P0.4	4		21	P4.5/AIN5
P0.0/INT0	5		20	P4.4/AIN4
P0.1/PWM0	6		19	P4.3/AIN3/CM0O
P0.2/CM0P	7		18	P4.2/AIN2/CM1O
P0.3/CM0N	8		17	P4.1/AIN1/CM2O
P1.6/CM1P	9		16	P4.0/AIN0/AVREFH
P1.5/CM1N	10		15	P1.0/OPN
P1.4/CM2P	11		14	P1.1/OPP
P1.3/CM2N	12		13	P1.2/OPO

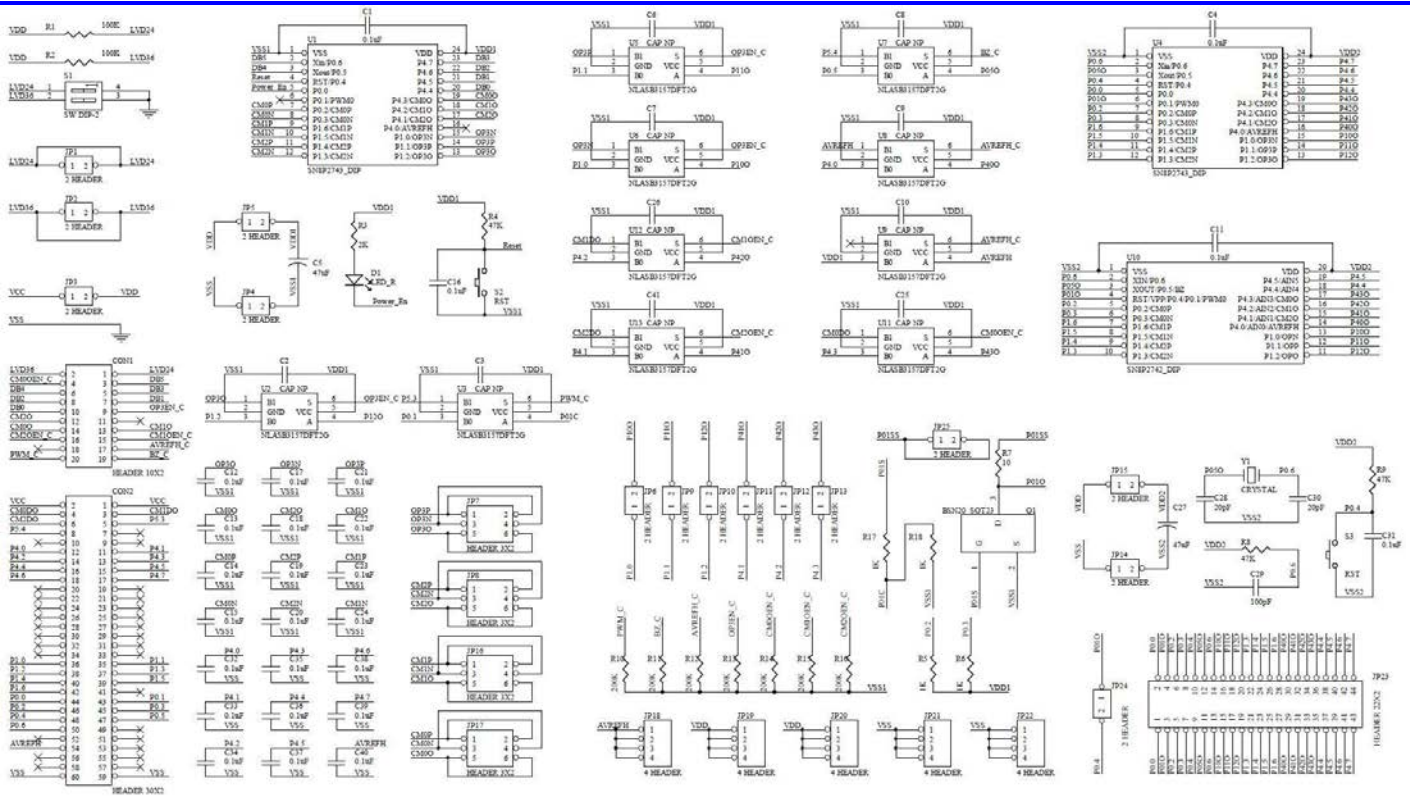
- U10: SN8P2742 DIP 封装形式的 IC 接口，连接到用户目标板。

VSS	1	U	20	VDD
XIN/P0.6	2		19	P4.5/AIN5
XOUT/P0.5/BZ	3		18	P4.4/AIN4
RST/VPP/P0.4/P0.1/PWM0	4		17	P4.3/AIN3/CM0O
P0.2/CM0P	5		16	P4.2/AIN2/CM1O
P0.3/CM0N	6		15	P4.1/AIN1/CM2O
P1.6/CM1P	7		14	P4.0/AIN0/AVREFH
P1.5/CM1N	8		13	P1.0/OPN
P1.4/CM2P	9		12	P1.1/OPP
P1.3/CM2N	10		11	P1.2/OPO

- C32~C39: 0.1uF 的旁路电容，连接到 AIN0~AIN7 引脚处。
- C40: 0.1uF 的旁路电容，连接到 AVREFH 引脚处。
- C13: 0.1uF 的旁路电容，连接到 CMP0 的输出引脚处。
- C22: 0.1uF 的旁路电容，连接到 CMP1 的输出引脚处。
- C18: 0.1uF 的旁路电容，连接到 CMP2 的输出引脚处。
- C17: 0.1uF 的旁路电容，连接到 OP-Amp 的负极输入引脚处。
- C21: 0.1uF 的旁路电容，连接到 OP-Amp 的正极输入引脚处。
- C12: 0.1uF 的旁路电容，连接到 OP-Amp 的输出引脚处。
- C14: 0.1uF 的旁路电容，连接到 CMP0 的正极输入引脚处。
- C15: 0.1uF 的旁路电容，连接到 CMP0 的负极输入引脚处。
- C23: 0.1uF 的旁路电容，连接到 CMP1 的正极输入引脚处。
- C24: 0.1uF 的旁路电容，连接到 CMP1 的负极输入引脚处。
- C19: 0.1uF 的旁路电容，连接到 CMP2 的正极输入引脚处。
- C20: 0.1uF 的旁路电容，连接到 CMP2 的负极输入引脚处。
- JP24: 单片机选择跳线（跳线短接时选择 SN8P2742，断开时选择 SN8P2743）。
- JP25: P0.1 输出 MOS 电路电源接口。

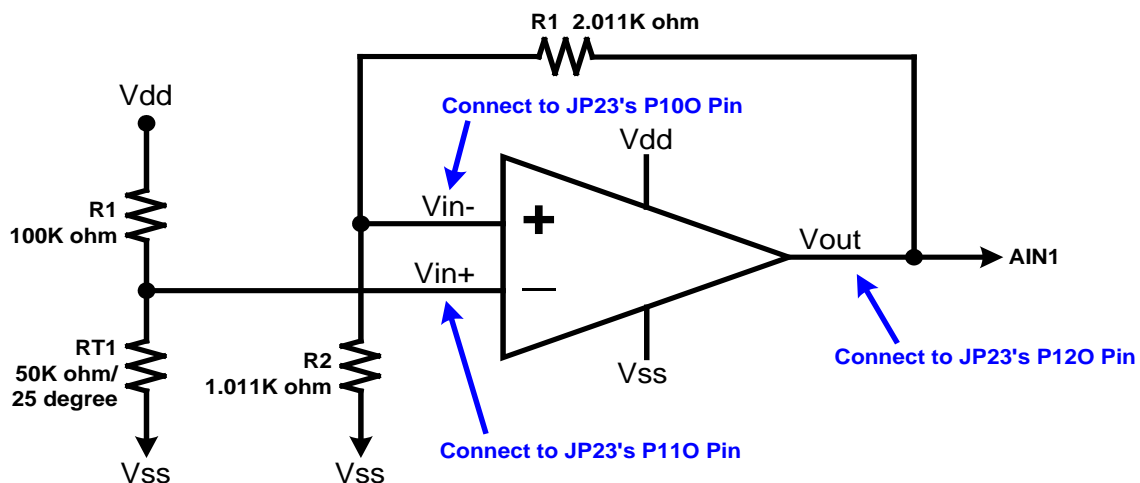


SN8P2743 EV-kit 电路图如下:



17.2 ICE和EV-KIT应用注意事项

1. 连接 SN8P2740 EV-Kit 到 SN8ICE2K Plus2 之前，必须将 SN8ICE2K Plus2 上的电源开关关闭。
2. 将 EV-KIT 上的 CON1/CON2 连接到 ICE 上的 JP3/CON1。
3. 必须将 SN8ICE2K Plus2 上的 AVREFH/VDD 跳线断开。
4. 用户完成 1、2、3 之后，将 SN8ICE2K Plus2 上的电源开关打开。
5. 打开 SN8ICE2K Plus2 的电源开关后，EV-Kit 上的电源指示灯 D1 亮，若 D1 不亮，用户可马上联系 SONIX 的代理商。
6. 如果用户选择 SN8P2743，则 JP24 处于断开状态；若选择 SN8P2742，则 JP24 就处于短接状态。
7. 在 ICE 上仿真 IHRC_16M 功能时，必须连接 16MHz 的外部晶振。
8. 使能 ADC 功能时，ADM 的 bit3 (FAVREFH) 置高时，P400 或 JP18 作为外部参考电压输入引脚。
9. 使能 ADC 功能时，ADM 的 bit3 (FAVREFH) 置低时，P400 作为模拟信号输入引脚，JP18 (AVREFH) 不能连接任何电源设备。
10. 在 JP18 (AVREFH) 观察 ADC 内部或外部参考电压。
11. SN8P2743 的 OP-Amp 应用电路如下图所示 (OP-AMP 连接)：



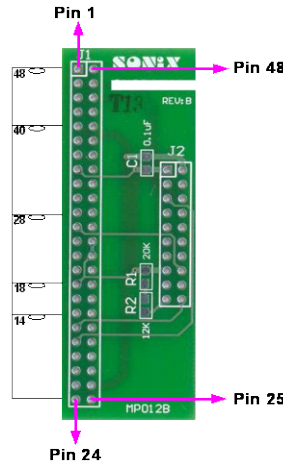
12. 上图中，使能 SN8P2743 的 OP-AMP 功能时，P120 (JP23) 为 OP-AMP 的输出引脚，P110 (JP23) 为 OP-AMP

的正极输入引脚，P100 (JP23) 为 OP-AMP 的负极输入引脚。

13. 上图中,用户想要测量 OP-AMP $V_+/V_-/V_o$ 的电压,OP-AMP 的实际负极电压是 OP3N(JP7), 正极电压是 OP3P(JP7), 输出电压是 OP3O (JP7)。
14. 由于 OP-AMP 系列与模拟开关内部电阻 (R_{on}) 连接, 故测量时 OP-AMP 有不同的连接方式。
15. 使能 CMP0 功能时, CM0P/CM0N (JP17) 为外部模拟信号输入引脚, P430 (JP23) 输出 CMP0 的输出结果。
16. 使能 CMP1 功能时, CM1P/CM1N (JP16) 为外部模拟信号输入引脚, P420 (JP23) 输出 CMP1 的输出结果。
17. 使能 CMP2 功能时, CM2P/CM2N (JP8) 为外部模拟信号输入引脚, P410 (JP23) 输出 CMP2 的输出结果。
18. 用户需要使用 CMP0~CMP2 的 CM0P/CM0N/CM1P/CM1N/CM2P/CM2N 模拟功能时, 必须连接到 JP17/JP16/JP8; 若用户需要使用 CMP0~CMP2 的 CM0P/CM0N/CM0O/CM1P/CM1N/CM1O/CM2P/CM2N/CM2O 的逻辑功能时, 必须连接到 JP23 的 P02/P03/P43O/P16/P15/P42O/P14/P13/P41O。
19. 用户在 ICE 仿真时需要使用 TC0 特殊功能 (脉冲产生器功能, TC0 时钟源为 Fcpu) 时, 若用户设置了 IDE 断点, PWM 脉冲产生器的输出状态不确定 (Fcpu 停止)。
20. 用户在 ICE 仿真时需要使用 TC0 特殊功能 (脉冲产生器功能, TC0 时钟源为 Fhosc) 时, 若用户设置了 IDE 断点, PWM 脉冲产生器的输出完成 (Fhosc 仍然工作), PWM 脉冲产生器的输出状态返回到空闲状态。
21. 用户在 ICE 仿真时需要使用 CMP1 和 CMP2 的 de-bounce 时间控制 (CM1D3~CM1D0, CM2D3~CM2D0, 时钟源为 Fcpu) 时, 若用户设置了 IDE 断点, CM0O 的输出会受到影响 (Fcpu 停止)。
22. 用户在 ICE 仿真时需要使用 CMP0 的 de-bounce 时间控制 (CM0D3~CM0D0, 时钟源为 Fhosc) 时, 若用户设置了断点, CM0O 的输出不会受到影响 (Fhosc 仍然工作)。

18 OTP烧录引脚

18.1 WRITER转接板引脚分配



JP3 (48-Pin IC 座引脚分配)

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

Writer JP1/JP2

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

JP1 连接烧录转接板

JP2 连接 DICE 和 >48 pin 封装形式

18.2 烧录引脚配置

SN8P2740 系列单片机的烧录引脚信息							
单片机名称		SN8P2743K(SKDIP)/S(SOP)			SN8P2742P(DIP)/S(SOP)		
Writer 接口		IC 和 JP3 的引脚配置					
JP1/JP2 引脚编号	JP1/JP2 引脚名称	IC 引脚编号	IC 引脚名称	JP3 引脚编号	IC 引脚编号	IC 引脚名称	JP3 引脚编号
1	VDD	24	VDD	36	20	VDD	34
2	GND	1	VSS	13	1	VSS	15
3	CLK	16	P4.0	28	14	P4.0	28
4	CE	-	-	-	-	-	-
5	PGM	20	P4.4	32	18	P4.4	32
6	OE	17	P4.1	29	15	P4.1	29
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP	4	RST	16	4	RST	18
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	3	XOUT/P0.5	15	3	XOUT/P0.5	17

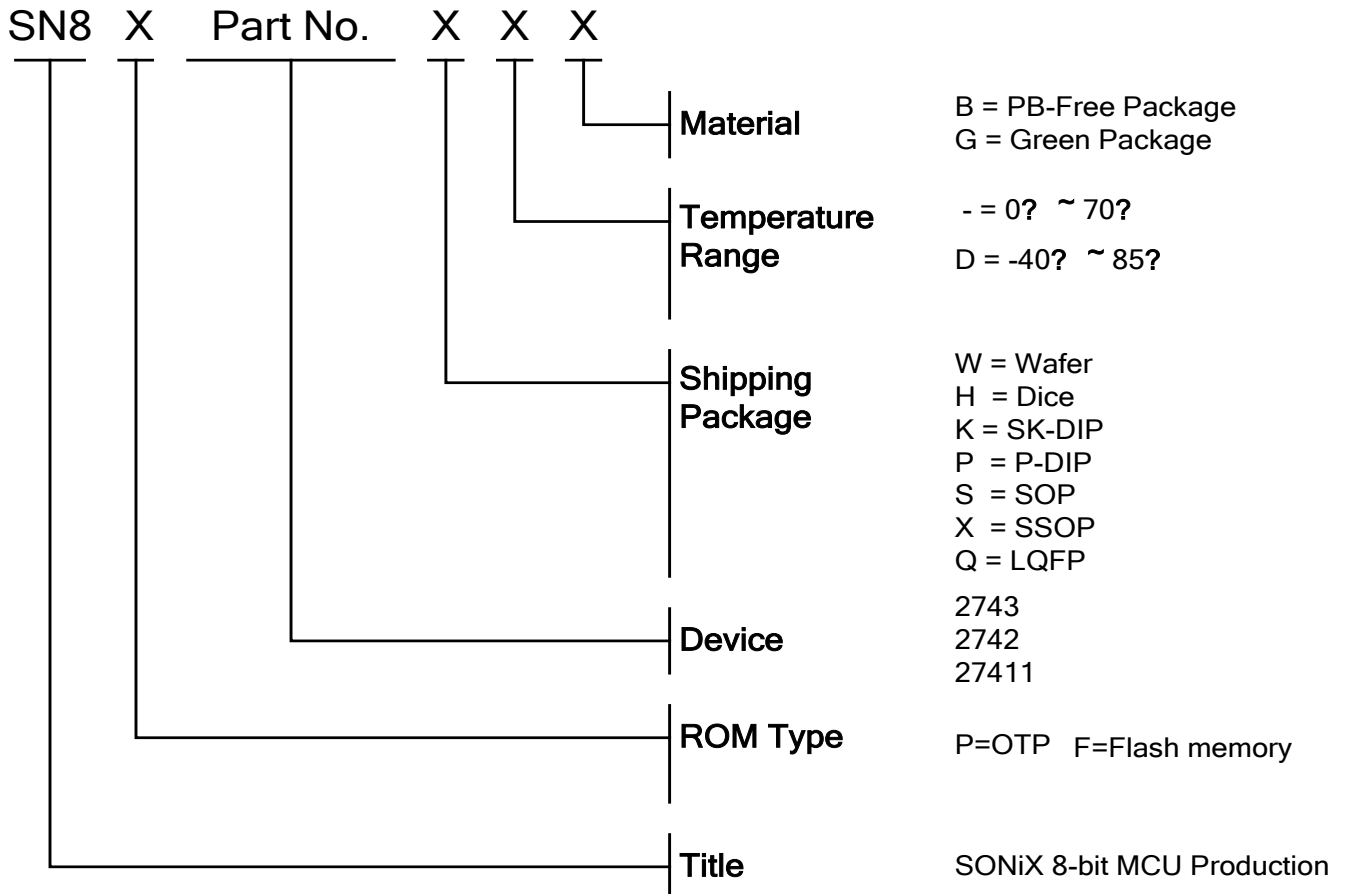
SN8P2740 系列单片机的烧录引脚信息							
单片机名称		SN8P27411P(P-DIP)			SN8P27411S(SOP)		
Writer 接口		IC 和 JP3 的引脚配置					
JP1/JP2 引脚编号	JP1/JP2 引脚名称	IC 引脚编号	IC 引脚名称	JP3 引脚编号	IC 引脚编号	IC 引脚名称	JP3 引脚编号
1	VDD	1	VDD	17	1	VDD	17
2	GND	13	VSS	29	16	VSS	29
3	CLK	10	P4.0	26	10	P4.0	26
4	CE	-	-	-	-	-	-
5	PGM	14	P4.4	30	13	P4.4	30
6	OE	11	P4.1	27	11	P4.1	27
7	D1	-	-	-	-	-	-
8	D0	-	-	-	-	-	-
9	D3	-	-	-	-	-	-
10	D2	-	-	-	-	-	-
11	D5	-	-	-	-	-	-
12	D4	-	-	-	-	-	-
13	D7	-	-	-	-	-	-
14	D6	-	-	-	-	-	-
15	VDD	-	-	-	-	-	-
16	VPP	3	RST	19	3	RST	19
17	HLS	-	-	-	-	-	-
18	RST	-	-	-	-	-	-
19	-	-	-	-	-	-	-
20	ALSB/PDB	2	XOUT/P0.5	18	2	XOUT/P0.5	18

19 单片机正印命名规则

19.1 概述

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

19.2 单片机型号说明



19.3 命名举例

- **Wafer, Dice:**

单片机名称	ROM 类型	器件 (Device)	封装形式	温度范围	封装材料
S8P2743W	OTP	2743	Wafer	0°C~70°C	-
SN8P2743CH	OTP	2743	Dice	0°C~70°C	-

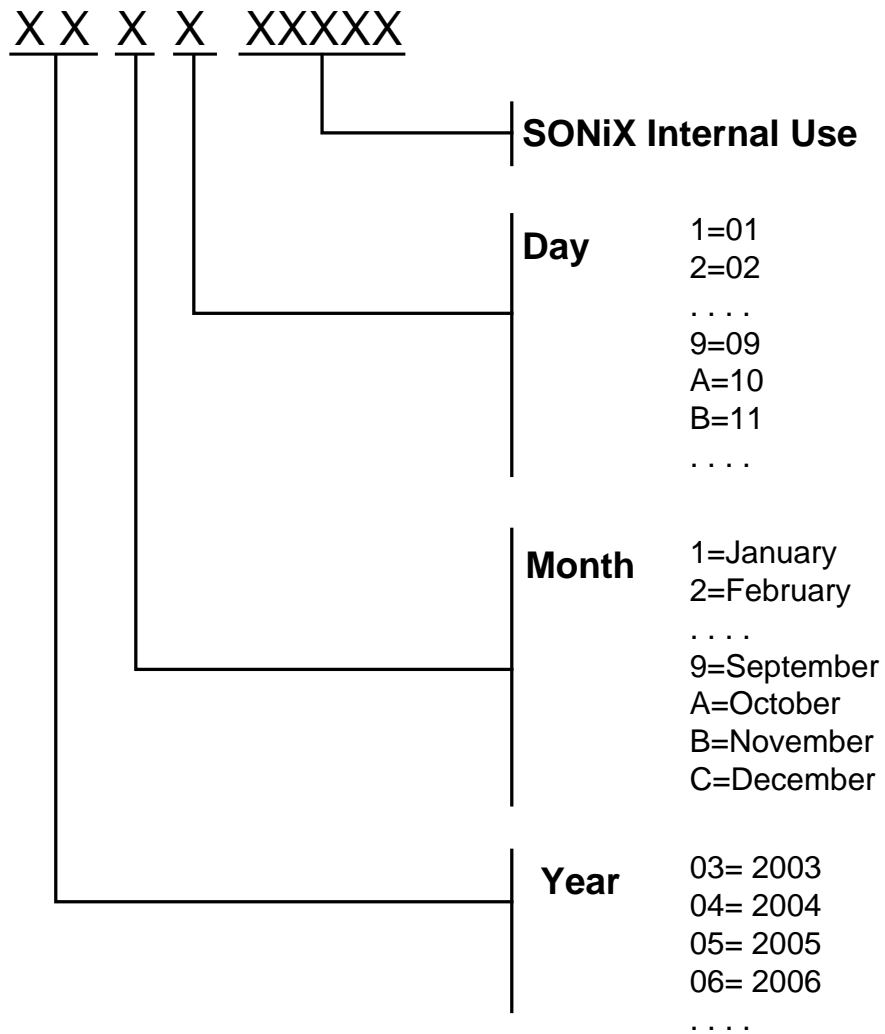
- **绿色封装:**

单片机名称	ROM 类型	器件 (Device)	封装形式	温度范围	封装材料
SN8P2743KG	OTP	2743	SK-DIP	0°C~70°C	绿色封装
SN8P2743SG	OTP	2743	SOP	0°C~70°C	绿色封装
SN8P2743KDG	OTP	2743	SK-DIP	-40°C~85°C	绿色封装
SN8P2743SDG	OTP	2743	SOP	-40°C~85°C	绿色封装
SN8P2742PG	OTP	2743	DIP	0°C~70°C	绿色封装
SN8P2742SG	OTP	2743	SOP	0°C~70°C	绿色封装
SN8P2742PDG	OTP	2743	DIP	-40°C~85°C	绿色封装
SN8P2742SDG	OTP	2743	SOP	-40°C~85°C	绿色封装
SN8P27411PG	OTP	2743	DIP	0°C~70°C	绿色封装
SN8P27411SG	OTP	2743	SOP	0°C~70°C	绿色封装
SN8P27411PDG	OTP	2743	DIP	-40°C~85°C	绿色封装
SN8P27411SDG	OTP	2743	SOP	-40°C~85°C	绿色封装

- **无铅封装:**

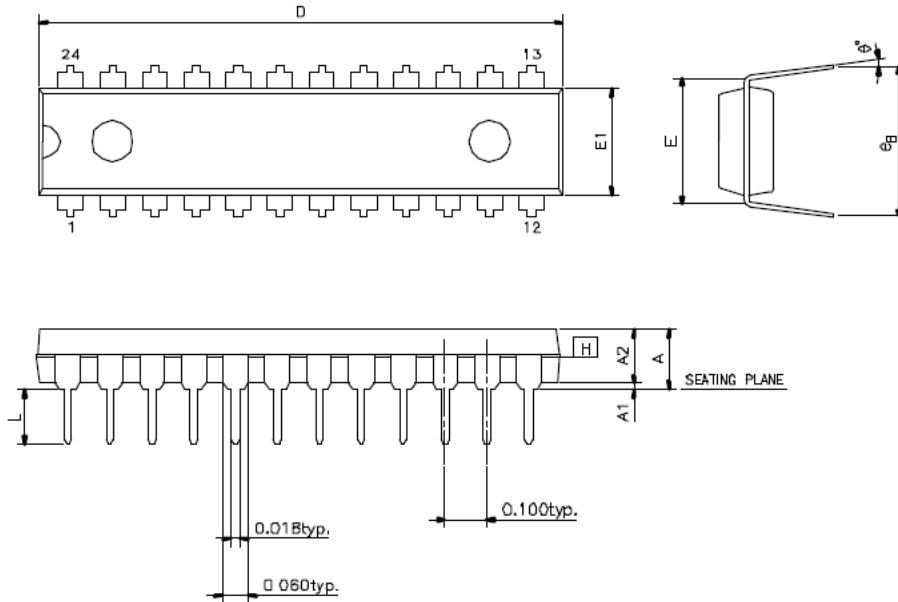
单片机名称	ROM 类型	器件 (Device)	封装形式	温度范围	封装材料
SN8P2743KB	OTP	2743	SK-DIP	0°C~70°C	无铅封装
SN8P2743SB	OTP	2743	SOP	0°C~70°C	无铅封装
SN8P2743KDB	OTP	2743	SK-DIP	-40°C~85°C	无铅封装
SN8P2743SDB	OTP	2743	SOP	-40°C~85°C	无铅封装
SN8P2742PB	OTP	2743	DIP	0°C~70°C	无铅封装
SN8P2742SB	OTP	2743	SOP	0°C~70°C	无铅封装
SN8P2742PDB	OTP	2743	DIP	-40°C~85°C	无铅封装
SN8P2742SDB	OTP	2743	SOP	-40°C~85°C	无铅封装
SN8P27411PB	OTP	2743	DIP	0°C~70°C	无铅封装
SN8P27411SB	OTP	2743	SOP	0°C~70°C	无铅封装
SN8P27411PDB	OTP	2743	DIP	-40°C~85°C	无铅封装
SN8P27411SDB	OTP	2743	SOP	-40°C~85°C	无铅封装

19.4 日期码规则



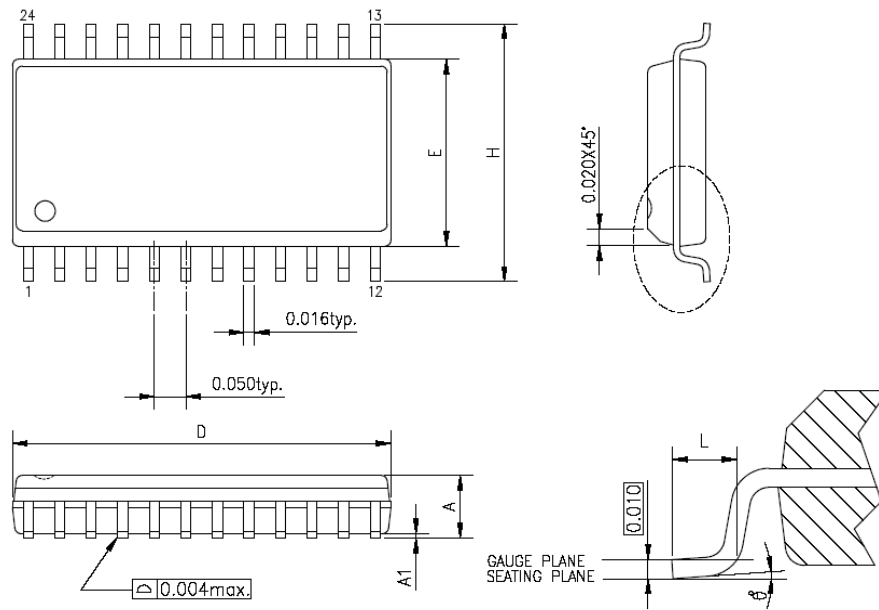
20 封装信息

20.1 SK-DIP 24 PIN



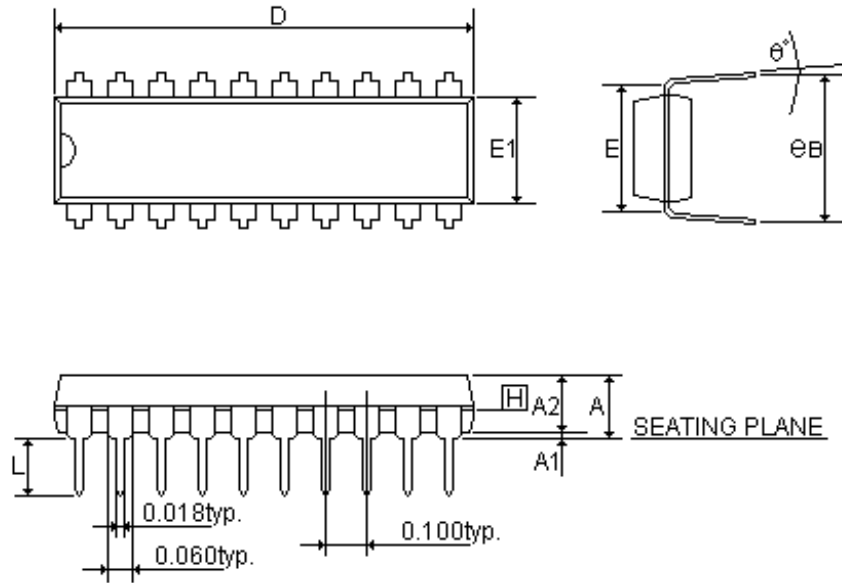
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-		0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	1.230	1.250	1.280	31.242	31.75	32.512
E	0.30 BSC			7.620 BSC		
E1	0.253	0.258	0.263	6.426	6.553	6.680
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.335	0.355	0.375	8.509	9.017	9.525
θ°	0°	7°	15°	0°	7°	15°

20.2 SOP 24 PIN



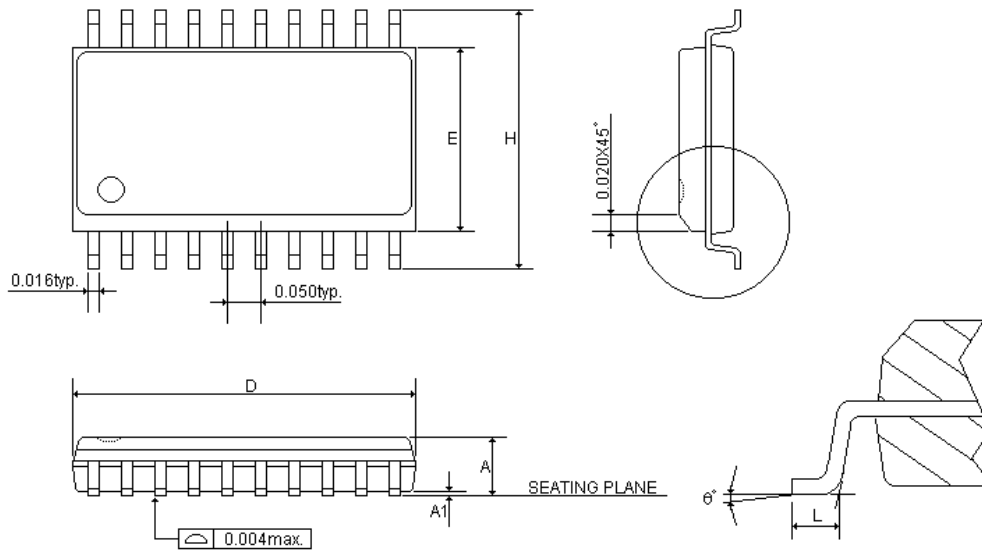
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.069	-	-	1.753
A1	0.004	-	0.010	0.102	-	0.254
D	0.612	0.618	0.624	15.545	15.697	15.850
E	0.292	0.296	0.299	7.417	7.518	7.595
H	0.405	0.412	0.419	10.287	10.465	10.643
L	0.021	0.031	0.041	0.533	0.787	1.041
θ°	0°	4°	8°	0°	4°	8°

20.3 P-DIP 20 PIN



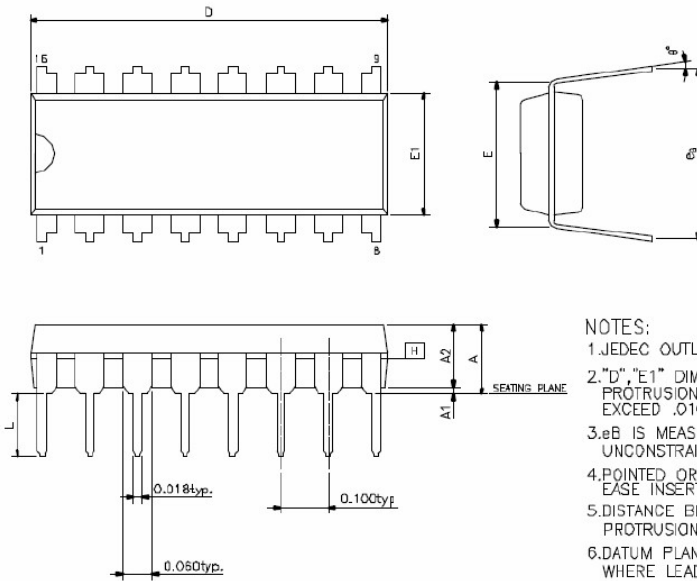
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.980	1.030	1.060	24.892	26.162	26.924
E	0.300			7.620		
E1	0.245	0.250	0.255	6.223	6.350	6.477
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.335	0.355	0.375	8.509	9.017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

20.4 SOP 20 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	0.496	0.502	0.508	12.598	12.751	12.903
E	0.291	0.295	0.299	7.391	7.493	7.595
H	0.394	0.407	0.419	10.008	10.325	10.643
L	0.016	0.033	0.050	0.406	0.838	1.270
θ°	0°	4°	8°	0°	4°	8°

20.5 P-DIP 16 PIN

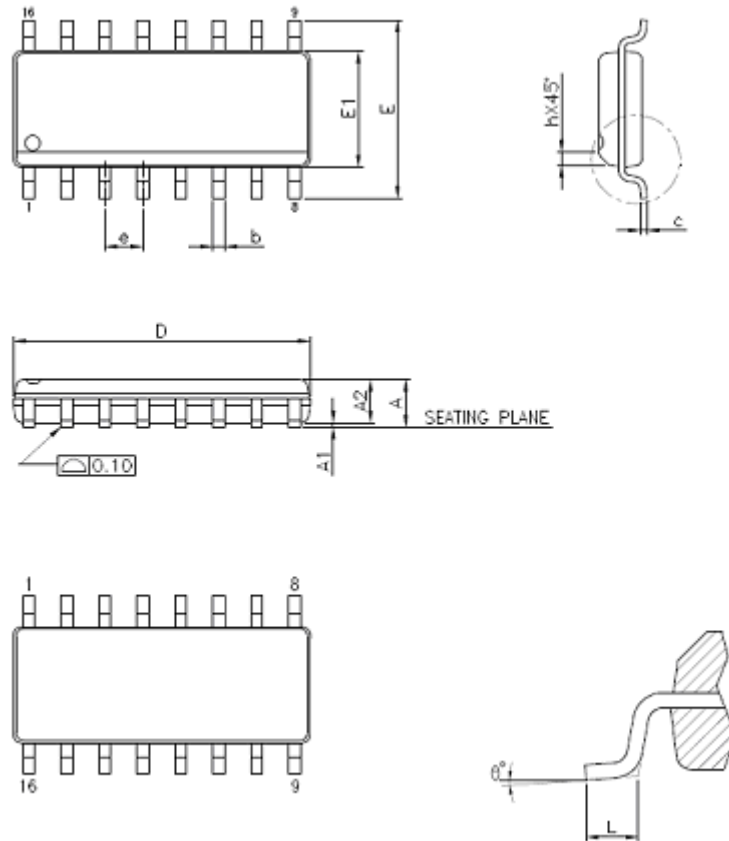


NOTES:

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

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	0.735	0.775	0.775	18.669	19.177	19.685
E	0.300BSC			7.620BSC		
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°

20.6 SOP 16 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.069	-	-	1.75
A1	0.004	-	0.010	0.10	-	0.25
A2	0.049	-		1.25	-	-
b	0.012	-	0.020	0.31	-	0.51
c	0.004	-	0.010	0.10	-	0.25
D	9.90BSC			9.90BSC		
E	6.00BSC			6.00BSC		
E1	3.90BSC			3.90BSC		
e	1.27BSC			1.27BSC		
h	0.016	-	0.050	0.40	-	1.27
L	0.010	-	0.020	0.25	-	0.50
θ°	0°	-	8°	0°	-	8°

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

总公司：

地址：台湾新竹县竹北市台元街 36 号 10 楼之一

电话：886-3-5600-888

传真：886-3-5600-889

台北办事处：

地址：台北市松德路 171 号 15 楼之 2

电话：886-2-2759 1980

传真：886-2-2759 8180

香港办事处：

地址：香港新界沙田火炭禾盛街 11 号，中建电讯大厦 26 楼 03 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