AiP8P102A

2K OTP ROM 的 AD 型 8 位微控制器

产品说明书

说明书发行履历：

版本	发行时间	新制/修订内容
V0.0	2016-07	新制
V0.1	2017-10	修订
V0.2	2018-01	修订
V0.3	2019-02	修订
V1.0	2019-07	修订

1、概述

1.1、说明

AiP8P102A 是一块 8 位的单片机电路，主要应用于为家电产品提供高抗干扰性能解决方案。其主要特点如下：

1.2、特性

- 存储器配置
  - OTP ROM 空间：2K * 16 位。
  - RAM 空间：128 字节。
- 8 层堆栈缓存器
  - 4 个中断源
  - 3 个内部中断源：T0、TC0、ADC
  - 1 个外部中断源：INT0
- I/O 引脚配置
  - 输入输出双向端口：P0、P4、P5。
  - 具有唤醒功能的端口：P0 电平触发
  - 内置上拉电阻端口：P0、P4、P5。
  - 外部中断引脚：P0.0
  - ADC 输入引脚：AIN0~AIN4
- FCPU(指令周期)
  - Fcpu=Fosc/4、Fosc/8、Fosc/16
- 强大的指令系统
  - 指令长度为 1 个字
  - 大部分指令只需要一个时钟周期
  - 跳转指令 JMP 可在整个 ROM 区执行
  - 查表指令 MOVC 可寻址整个 ROM 区
- 2 个 8 位定时/计数器
  - T0：基本定时器
  - TC0：自动装载定时器/计数器/PWM/ Buzzer 输出。
- 单通道 8 位 PWM 输出
- 单通道 2KHz/4KHz 蜂鸣器输出
- 内置看门狗定时器，其时钟源由内部低速 RC 振荡器提供（16KHz @3V，32KHz @5V）
- 5 通道 12 位 ADC
- 4 种时钟系统
  - 外部高速时钟：RC 模式，高达 10 MHz
  - 外部高速时钟：晶体模式，高达 16 MHz
  - 内部高速时钟：RC 模式，高达 16MHz
  - 内部低速时钟：RC 振荡器 16KHz（3V），32KHz（5V）
- 4 种工作模式
  - 普通模式：高、低速时钟同时工作。
  - 低速模式：只有低速时钟工作。
  - 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
  - DIP20/SOP20/TSSOP20
  - SOP16

表 1-1 AiP8P102A 型号

型号	全称	封装	备注
AiP8P102A	AiP8P102AGO	SOP20	
	AiP8P102AGP	DIP20	
	AiP8P102AGI	TSSOP20	
	AiP8P102AEO	SOP16	

2、功能框图及引脚说明

2.1、功能框图

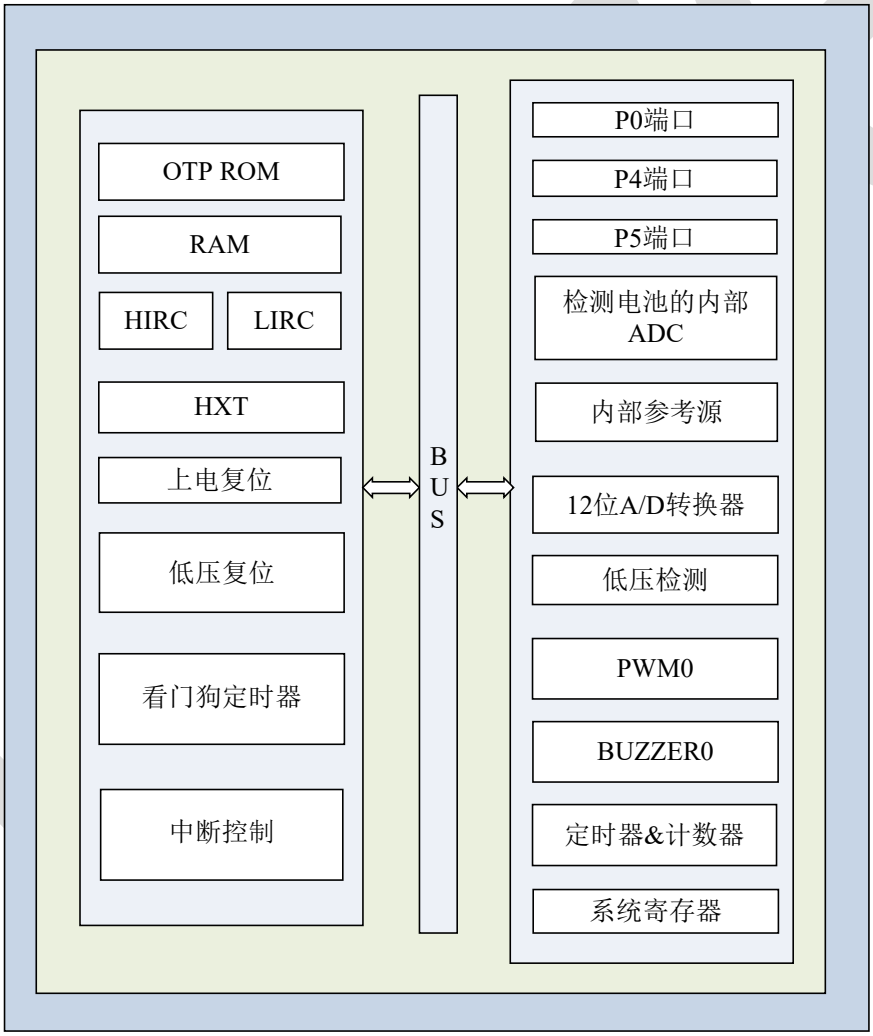


图 2-1 功能框图

2.2、引脚排列图

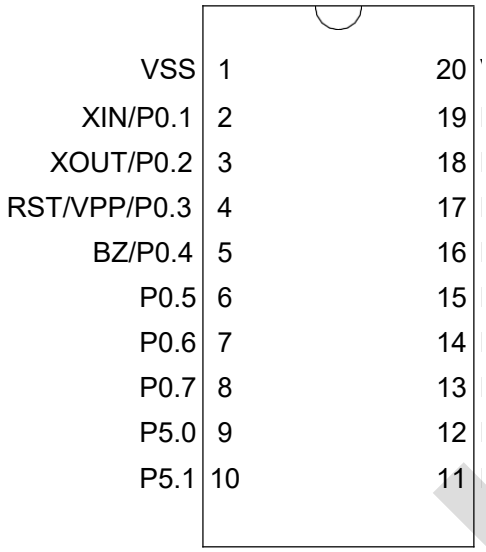


图 2-2 AiP8P102A- DIP20/SOP20/TSSOP20 引脚排列图

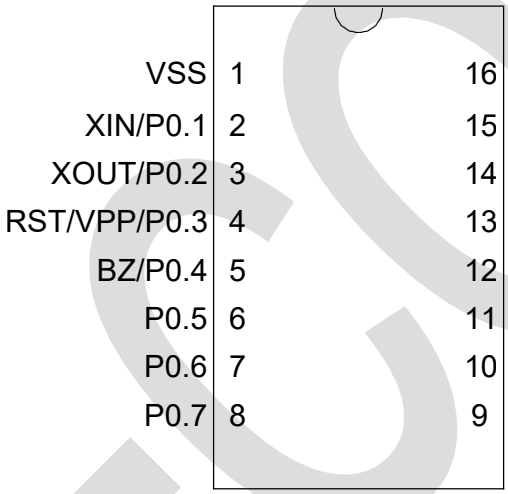


图 2-3 AiP8P102A- SOP16 引脚排列图

2.3、引脚说明

表 2-1 AiP8P102A 引脚说明

引脚名称	类型	说明
VDD,VSS	P	电源输入端
P0.3/RST/VPP	I, P	P0.3：禁止外部复位时为单向输入引脚，施密特触发，无内置上拉电阻， RST：系统复位输入引脚，施密特结构，低电平触发，通常保持高电平。 VPP：OTP 烧录引脚。

P0.1/XIN	I/O	P0.1：双向输入/输出引脚，输入模式时为施密特触发，内置上拉电阻，具有唤醒功能。 XIN：使能外部振荡电路（晶体/RC 振荡电路）时为振荡信号输入引脚。
P0.2/XOUT	I/O	P0.2：双向输入/输出引脚，输入模式时为施密特触发，内置上拉电阻，具有唤醒功能。 XOUT：使能外部晶体振荡器时为振荡器输出引脚。
P0.0/INT0	I/O	P0.0：双向输入/输出引脚，输入模式时为施密特触发，内置上拉电阻，具有唤醒功能。INT0：外部中断触发引脚（施密特触发）。 TC0 事件计数器的信号输入引脚。
P0.4/BZ	I/O	P0.4：双向输入/输出引脚，非施密特触发，内置上拉电阻，具有唤醒功能。 BZ：2KHz/4KHz buzzer 输出引脚。
P0[7:5]	I/O	双向输入/输出引脚，输入模式时为施密特触发，内置上拉电阻。
P4.[4:0]/AIN[4:0]	I/O	双向输入/输出引脚，非施密特触发，内置上拉电阻。 AIN[4:0]：ADC 输入通道。
P5[3:0]	I/O	双向输入/输出引脚，输入模式时为施密特触发，内置上拉电阻。
P5.4/PWM/TC0OUT	I/O	双向输入/输出引脚，输入模式时为施密特触发，内置上拉电阻。 PWM：PWM 输出引脚。 TC0OUT：TC0/2 信号输出引脚。

注：I=输入，O=输出，P=电源

2.4、引脚结构图

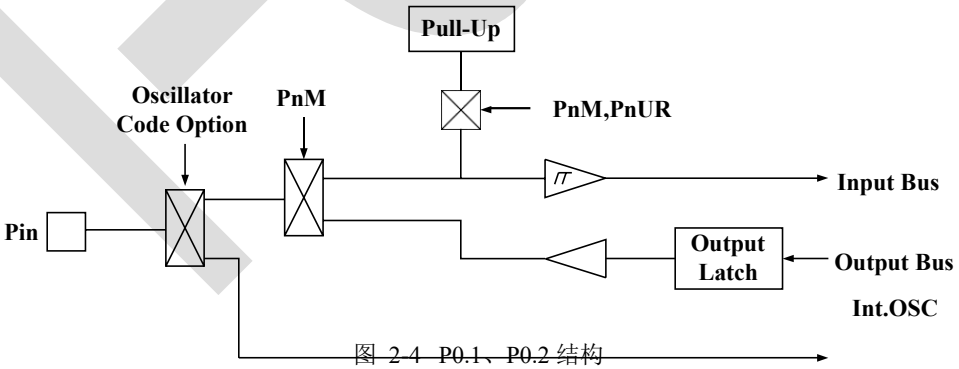


图 2-4 P0.1、P0.2 结构

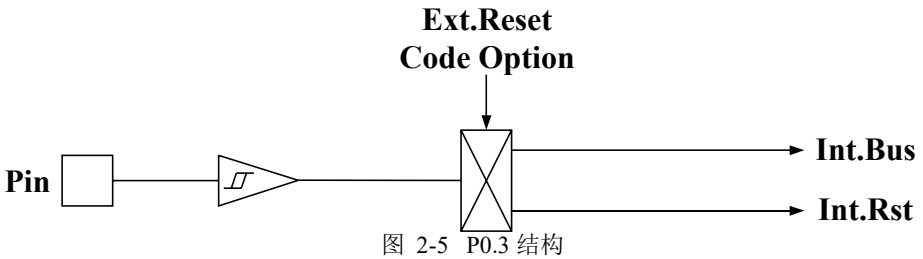


图 2-5 P0.3 结构

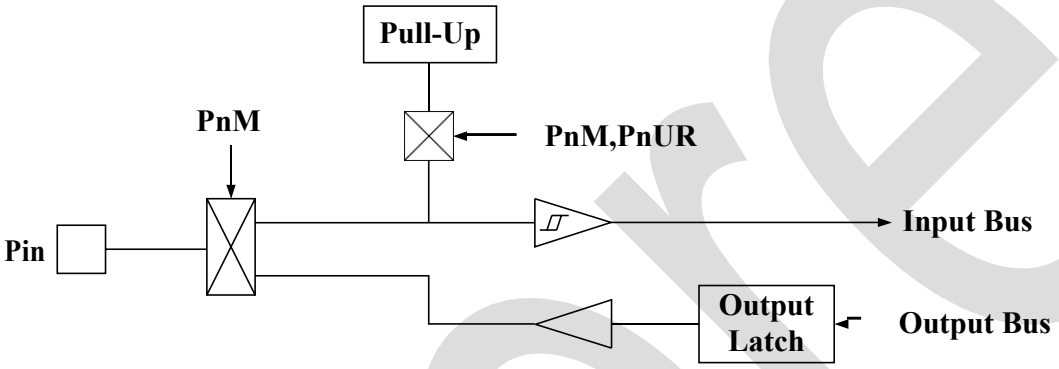


图 2-6 P0、P5 结构

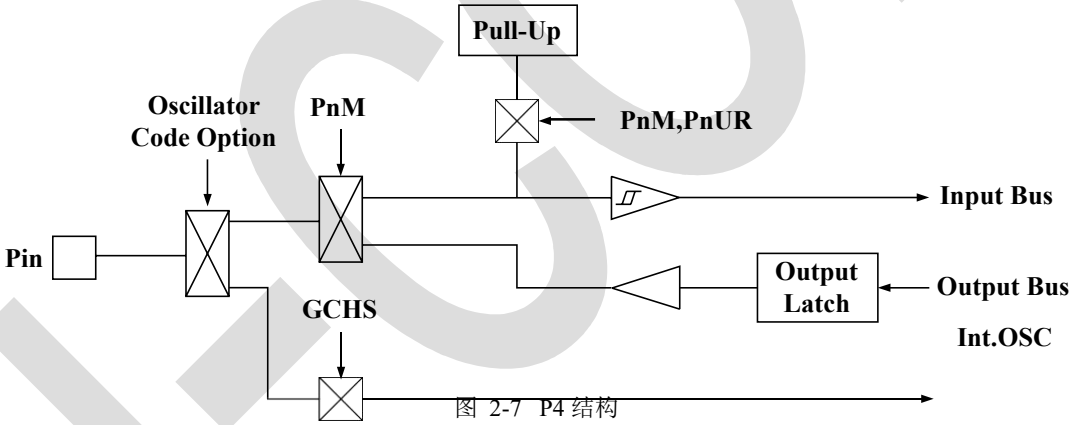


图 2-7 P4 结构

3、电特性

3.1、极限参数

表 3-1 极限参数

参数名称	符号	最小	最大	单位
工作电压	VDD	GND-0.3	+6.5	V
输入电压	VI	GND-0.3	VDD+0.3	V
输出电压	VO	GND-0.3	VDD+0.3	V
工作温度	TOPR	-40	+85	℃
储存温度	TSTG	-60	+150	℃
焊接温度	TL	-	+245	℃

注：1.除非另有规定，T_{amb}=25℃。

3.2、电气特性

表 3-2 电气特性

参数名称	符号	测试条件	最小	典型	最大	单位
工作电压	Vdd	普通模式，VPP=VDD,25℃	2.4	5	5.5	V
		普通模式，VPP=VDD,-40~85℃	2.5	5	5.5	V
RAM 数据保持电压	V_{DR}		0.35			V
VDD 上升率	V_{POR}	VDD 上升率确保内部上电复位	0.05			V/ms
输入低电平电压	VIL1	所有的输入口	VSS	-	0.3VDD	V
	VIL2	复位引脚	VSS	-	0.2VDD	V
输入高电平电压	VIH1	所有的输入口	0.7VDD	-	VDD	V
	VIH2	复位引脚	0.9VDD		VDD	V
复位漏电流	Ilekg	VIN=VDD			2	μA
I/O 上拉电阻	R_{UP}	VIN=VSS,VDD=3V	100	200	300	KΩ
		VIN=VSS,VDD=5V	50	100	150	
I/O 口输入漏电流	Ilekg	上拉电阻失去作用，Vin=Vdd			2	μA
I/O 口源电流	IoH	Vop = Vdd – 0.5V	8	12		mA
I/O 口灌电流	IoL	Vop = Vss +0.5V	8	15		mA
INTn 触发脉	Tint0	INT0 中断请求的脉冲宽度	2/fcpu	-	-	cycle

冲宽度							
电流	IDD1	运行模式（没有加载，fcpu=fosc/4）	VDD=5V,4MHz	-	2.36	-	mA
			VDD=3V,4MHz	-	0.81	-	mA
	IDD2	缓慢模式（内部低 RC,在高电平时钟停止）	VDD=5,32KHz		62.3	-	μA
			VDD=3V,16KHz		7.1	-	μA
	IDD3	睡眠模式	VDD=5V, 25℃		0.6	-	μA
			VDD=3V, 25℃		0.5	-	μA
			VDD=5V,-40~85℃		-	-	μA
			VDD=3V,-40~85℃		-	-	μA
	IDD4	绿色模式 没有加载，fcpu=fosc/4 看门狗关闭）	VDD=5V,4MHz		0.6	-	mA
			VDD=3V,4MHz		0.17	-	mA
			VDD=5V,ILRC 32KHz		15	30	μA
			VDD=3V,ILRC 16KHz		3	6	μA
内部高电平振荡频率	Fihrc	内部高电平 RC（IHRC）	25℃ VDD=5V, Fcpu=1MHz	15.68	16	16.32	MHZ
			-40~85℃ VDD=2.4V~5.5V Fcpu=1MHz~16Mhz	13	16	19	MHZ
LVD 电压	Vdet0	低电压复位电平		1.6	2.0	2.3	V
	Vdet1	低电压复位电平，Fcpu=1MHz		2.0	2.3	3	V
		低电压复位电平，Fcpu=1MHz		2.0	2.3	3	V
	Vdet2	低电压复位电平，Fcpu=1MHz		2.7	3.3	4.5	V
AIN0~AIN5 输入电压	Vani	VDD=5V		0		Vrefh1~5	V
ADC 使能时间	Tast	ADENB=1,准备开始转换		100			μs

ADC 消耗电流	I_{ADC}	VDD=5V		0.6		mA
		VDD=3V		0.4		mA
ADC 时钟频率	F_{ADCLK}	VDD=5V	32K		8M	Hz
		VDD=3V	32K		8M	Hz
ADC 转换周期时间	F_{ADCYL}	VDD=2.4~5.5V	64			1/ F_{ADCLK}
ADC 取样速率（FADS=1）	F_{ADSMP}	VDD=5V			125	K/sec
		VDD=3V			80	K/sec
微分非线性	DNL	VDD=5V,AVREFH=3.2V, F_{ADSMP}=7.8K	±1	±2	±16	LSB
积分非线性	INL	VDD=5V,AVREFH=3.2V, F_{ADSMP}=7.8K	±2	±4	±16	LSB
无代码丢失	NMC	VDD=5V,AVREFH=3.2V, F_{ADSMP}=7.8K	8	9	10	Bits

注：除非另有规定，T_{amb}=25℃,VDD=5V，fosc=4MHz，fcpu=1MHz。

4、CPU 特性

4.1、中央处理器

4.1.1、程序存储器（ROM）

- ROM：2K

图 4-1 程序存储器

4.1.1.1、复位向量（0000H）

具有一个字长的系统复位向量（0000H）。

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

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

例：定义复位向量。

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

...

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

4.1.1.2 、中断向量（0008H）

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

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

例：定义中断向量，中断服务程序紧随 ORG 8H 之后。

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

例：定义中断向量，中断程序在用户程序之后。

```
.CODE
            ORG     0               ;
            JMP     START           ;跳至用户程序
            •••
            ORG     8H              ;中断向量
            JMP     MY_IRQ          ;跳至中断程序
            ORG     10H
START:                              ;用户程序开始
            •••                     ;
            JMP     START           ;用户程序结束
            •••
MY_IRQ:                             ;中断程序开始
            PUSH                    ;保存 ACC 和 PFLAG
            •••
            POP                     ;恢复 ACC 和 PFLAG
            RETI                    ;中断程序结束
            •••
            ENDP                    ;程序结束
```

注：从上面的程序中容易得出 IC 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。

4.1.1.3 、查表

在 AiP8P102A 单片机中，对 ROM 区中的数据进行查找，寄存器 Y 指向所找数据地址的中间字节（bit8~bit15），寄存器 Z 指向所找数据地址的低字节（bit0~bit7）。执行完 MOVC 指令后，所查找数据低字节内容被存入 ACC 中，而数据高字节内容被存入 R 寄存器。

例：查找 ROM 地址为“TABLE1”的值。

```
                B0MOV       Y, #TABLE1$M    ;设置 TABLE1 地址高字节
                B0MOV       Z, #TABLE1$L    ;设置 TABLE1 地址低字节
                MOVC                        ;查表，R = 00H，ACC = 35H
                                            ;查找下一地址
                INCMS       Z
                JMP         @F              ;Z 没有溢出
                INCMS       Y               ;Z 溢出（FFH →00），→Y=Y+1
                NOP                         ;
                                            ;
@@:             MOVC                        ;查表，R = 51H，ACC= 05H
                …                           ;
TABLE1:         DW          0035H           ;定义数据表（16 位）数据
                DW          5105H
                DW          2012H
                …
```

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

例：宏 INC_YZ。

```
INC_YZ          MACRO                       ;
                INCMS       Z
                JMP         @F              ;没有溢出
                INCMS       Y
                NOP                         ;没有溢出
@@:
                ENDM
```

例：通过“INC_YZ”对上例进行优化。

```
                B0MOV       Y, #TABLE1$M    ;设置 TABLE1 地址中间字节
```

```
                B0MOV       Z, #TABLE1$L    ;设置 TABLE1 地址低字节
                MOVC                        ;查表，R = 00H，ACC = 35H
                INC_YZ                      ; 查找下一地址数据
                                            ;
@@:             MOVC                        ;查表，R = 51H，ACC = 05H
                …
TABLE1:         DW          0035H           ;定义数据表（16 位）数据
                DW          5105H
                DW          2012H
                …
```

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

例：由指令 B0ADD/ADD 对 Y 和 Z 寄存器加 1。

```
                B0MOV       Y, #TABLE1$M    ;设置 TABLE1 地址中间字节
                B0MOV       Z, #TABLE1$L    ;设置 TABLE1 地址低字节
                B0MOV       A, BUF          ; Z = Z + BUF
                B0ADD       Z, A
                B0BTS1      FC              ;检查进位标志
                JMP         GETDATA         ;FC = 0
                INCMS       Y               ;FC = 1
                NOP
GETDATA:                                    ;
                MOVC                        ;存储数据，如果 BUF = 0，数据为 0035H
                                            ;如果 BUF = 1，数据=5105H
                                            ;如果 BUF = 2，数据=2012H
                …
```

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

4.1.1.4 、跳转表

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

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

例：跳转表。

```
        ORG     0100H       ;跳转表从 ROM 前端开始
        B0ADD   PCL, A      ;PCL = PCL +ACC，PCL 溢出时 PCH 加 1
        JMP     A0POINT     ; ACC = 0，跳至 A0POINT
        JMP     A1POINT     ; ACC = 1，跳至 A1POINT
        JMP     A2POINT     ; ACC = 2，跳至 A2POINT
        JMP     A3POINT     ; ACC = 3，跳至 A3POINT
```

AiP8P102A 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：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
```

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：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       B0PCL, A
            ENDM
```

注：“VAL”为跳转表列表中列表个数。

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
```

4.1.1.5 、CHECKSUM 计算

ROM 的最后一个地址是系统保留区，用户应该在计算 Checksum 时跳过该区域。

例：下面的程序说明如何从 00H 至用户代码结束的区域内进行 Checksum 计算。

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

```
        MOV     A, END_ADDR2
        CMPRS   A, Y                ; 是则检查 Y 的地址是否为
        用户程序结束地址中间地址。
        JMP     AAA                 ; 否，则进行 Checksum 计算。
        JMP     CHECKSUM_END        ; 是则 Checksum 计算结束。
Y_ADD_1:
        INCMS Y ;
        NOP
        JMP     @B                  ; 跳转到 Checksum 计算。
CHECKSUM_END:
        …
        …
END_USER_CODE:                      ; 程序结束。
```

4.1.2 、数据存储器（RAM）

RAM：128 字节

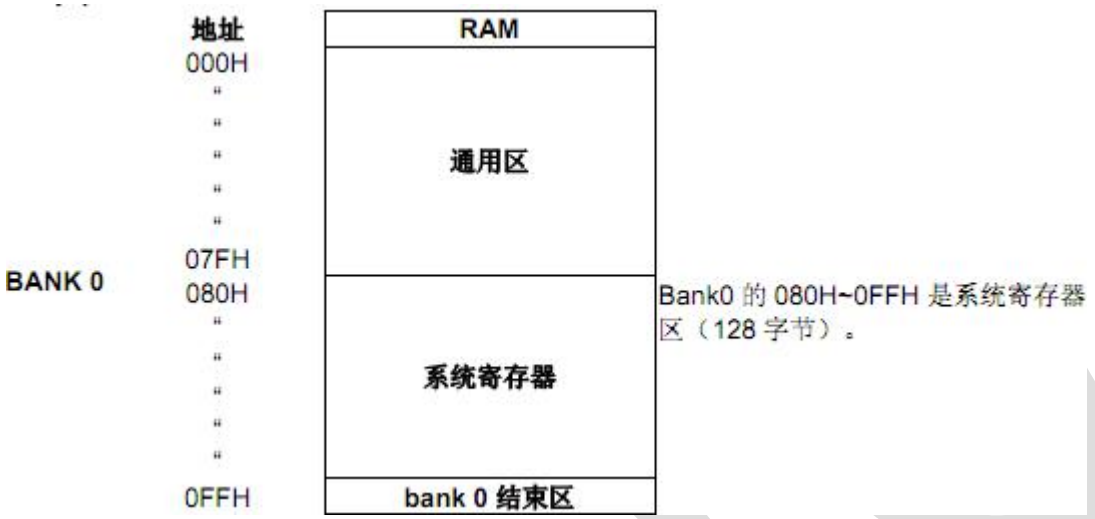


图 4-2 数据存储器

4.1.2.1 、系统寄存器

表 4-1 系统寄存器列表

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	-	-	R	Z	Y	-	PFLAG	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-		P4CON
B	-	ADM	ADB	ADR	-	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	-	-	-	-	P4M	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	TC0R	PCL	PCH
D	P0	-	-	-	P4	P5	-	-	T0M	T0C	TC0M	TC0C	BZM	-	-	STKP
E	P0UR	-	-	-	P4UR	P5UR	-	@YZ	-	-	-	-	-	-	-	-
F	STK7L	STK7H	STK6L	STK6H	STK5L	STK5H	STK4L	STK4H	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

系统寄存器说明

R =工作寄存器和 ROM 查表数据缓存器

Y, Z = 专用寄存器，@YZ 间接寻址寄存器，ROM 寻址寄存器

PFLAG = ROM 页和特殊标志寄存器

P4CON = P4 配置控制寄存器VERFH = ADC 参考电压寄存器ADM = ADC 模式寄存器

ADB = ADC 数据缓存器ADR = ADC 精度选择寄存器PnM = Pn 模式控制寄存器PEDGE= P0.0 模式控制寄存器

INTRQ= 中断请求寄存器

INTEN = 中断使能寄存器

OSCM= 振荡模式寄存器

WDTR= 看门狗清零寄存器

TC0R= TC0 自动装载数据缓存器

PCH, PCL = 程序计数器

Pn = Pn 数据缓存器

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

STK0~STK7= 堆栈缓存器

PnUR= Pn 上拉电阻控制寄存器

@YZ= 间接寻址寄存器

STKP = 堆栈指针

表 4-2 系统寄存器的位定义

地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	注释
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
0AFH				P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	R/W	P4CON
0B1H	ADENB	ADS	EOC	GCHS		CHS2	CHS1	CHS0	R/W	ADM
0B2H	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	R	ADB
0B3H		ADCKS1		ADCKS0	ADB3	ADB2	ADB1	ADB0	R/W	ADR
0B8H	P07M	P06M	P05M	P04M	P03M	P02M	P01M	P00M	R/W	P0M
0BFH				P00G1	P00G0				R/W	PEDGE
0C4H				P44M	P43M	P42M	P41M	P40M	R/W	P4M
0C5H				P54M	P53M	P52M	P51M	P50M	R/W	P5M
0C8H	ADCIRQ		TC0IRQ	T0IRQ				P00IRQ	R/W	INTRQ
0C9H	ADCIEN		TC0IEN	T0IEN				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						PC10	PC9	PC8	R/W	PCH
0D0H	P07	P06	P05	P04	P03	P02	P01	P00	R/W	P0
0D4H				P44	P43	P42	P41	P40	R/W	P4
0D5H				P54	P53	P52	P51	P50	R/W	P5
0D8H	T0ENB	T0rate2	T0rate1	T0rate0					R/W	T0M
0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DCH	BZEN	BZrate1	BZrate0						R/W	BZM
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H	P07R	P06R	P05R	P04R		P02R	P01R	P00R	W	P0UR
0E4H				P44R	P43R	P42R	P41R	P40R	W	P4UR
0E5H				P54R	P53R	P52R	P51R	P50R	W	P5UR

0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0F0H	S7PC7	S7PC6	S7PC5	S7PC4	S7PC3	S7PC2	S7PC1	S7PC0	R/W	STK7L
0F1H						S7PC10	S7PC9	S7PC8	R/W	STK7H
0F2H	S6PC7	S6PC6	S6PC5	S6PC4	S6PC3	S6PC2	S6PC1	S6PC0	R/W	STK6L
0F3H						S6PC10	S6PC9	S6PC8	R/W	STK6H
0F4H	S5PC7	S5PC6	S5PC5	S5PC4	S5PC3	S5PC2	S5PC1	S5PC0	R/W	STK5L
0F5H						S5PC10	S5PC9	S5PC8	R/W	STK5H
0F6H	S4PC7	S4PC6	S4PC5	S4PC4	S4PC3	S4PC2	S4PC1	S4PC0	R/W	STK4L
0F7H						S4PC10	S4PC9	S4PC8	R/W	STK4H
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H						S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH						S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH						S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH						S0PC10	S0PC9	S0PC8	R/W	STK0H

注: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. 所有寄存器的名称在SN8ASM编译器中做了宣告；

3. 寄存器中各位的名称已在SN8ASM编译器中以“F”为前缀定义过；

4. 指令“B0BSET”、“B0BCLR”、“BSET”、“BCLR”只能用于“R/W”寄存器。

4.1.2.2 、累加器

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

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

例：读/写ACC。

```
; 数据写入ACC。
        MOV         A, #0FH
; 读取ACC中的数据并存入BUF。
        MOV         BUF, A
        B0MOV       BUF, A
; BUF 中的数据写入ACC。
```

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

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

例：ACC 和工作寄存器中断保护操作。

```
INT_SERVICE:
        PUSH                    ;保存 PFLAG 和 ACC。
        …….
        …
        POP                     ; 恢复 ACC 和 PFLAG。
        RETI                    ; 退出中断。
```

4.1.2.3 、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 检测信息，其中，位 NT0 和 NPD 显示系统复位状态信息，包括上电复位、LVD 复位、外部复位和看门狗复位；位 C、DC 和 Z 显示 ALU 的运算信息。位 LVD24 和 LVD36 显示了单片机供电电压状况。

表 4-3 PFLAG 程序状态寄存器（086H）

Bit	7	6	5	4	3	2	1	0
Name	NT0	NPD	LVD36	LVD24	-	C	DC	Z
R/W	R/W	R/W	R	R	-	R/W	R/W	R/W
POR	X	X	0	0	-	0	0	0

位	字段	描述
7-6	NT0,NPD	NT0 NPD 复位状态 0 0 看门狗复位 0 1 保留 1 0 LVD 复位 1 1 外部复位
5	LVD36	3.6V LVD 工作电压标志，LVD 编译选项为 LVD_H 时有效。 0 =系统工作电压 VDD 超过 3.6V，低电压检测器没有工作； 1 =系统工作电压 VDD 低于 3.6V，说明此时低电压检测器已处于监控状态。
4	LVD24	2.4V LVD 工作电压标志，LVD 编译选项为 LVD_M 时有效。 0 =系统工作电压 VDD 超过 2.4V，低电压检测器没有工作；

		1 =系统工作电压 VDD 低于 2.4V，说明此时低电压检测器已处于监控状态。
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。
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。
0	Z	零标志。 1 = 算术/逻辑/分支运算的结果为零； 0 = 算术/逻辑/分支运算的结果非零。

注：关于标志位 C、DC 和 Z 的更多信息请参阅指令集相关内容。

4.1.2.4 、程序计数器

程序计数器PC是一个10位二进制程序地址寄存器，分高2位和低8位。专门用来存放下一条需要执行指令的内存地址。通常，程序计数器会随程序中指令的执行自动增加。

若程序执行 CALL 和 JMP 指令时，PC 指向特定的地址。

表 4-4 程序计数器

	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PC	-	-	-	-	-	-	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
POR	-	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

单地址跳转

在 AiP8P102A 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，则跳过下一条指令。

```
        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
        B0BTS0      FZ          ; Zero flag = 0则跳过下一条指令。
```

```
        JMP     C1STEP      ; 否则执行C1STEP。
        …
        …
C1STEP: NOP
```

如果 ACC 等于指定的立即数则 PC 值加 2，跳过下一条指令。

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

执行加 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

INCS：

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

INCMS：

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

执行减 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

DECS：

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

DECMS：

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

多地址跳转

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

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

例：PC = 0323H（PCH = 03H，PCL = 23H）。

```
; PC = 0323H
        MOV     A, #28H
        B0MOV   PCL, A      ;跳到地址0328H。
        …
; PC = 0328H
        MOV     A, #00H
        B0MOV   PCL, A      ;跳到地址0300H。
        …
```

例：PC = 0323H（PCH = 03H，PCL = 23H）。

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

4.1.2.5 、Y，Z 寄存器

寄存器 Y 和 Z 都是 8 位缓存器，主要用途如下：

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

表 4-5 Y 寄存器（084H）

Bit	7	6	5	4	3	2	1	0
Name	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

POR	X	X	X	X	X	X	X	X

表 4-6 Z 寄存器（083H）

Bit	7	6	5	4	3	2	1	0
Name	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	X	X	X	X	X	X	X	X

例：用 Y、Z 作为数据指针，访问 bank0 中 025H 处的内容。

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

例：利用数据指针@YZ 对 RAM 数据清零。

```
        B0MOV   Y, #0       ; Y = 0，指向bank 0。
        B0MOV   Z, #7FH     ; Z =7FH，RAM区的最后单元。
CLR_YZ_BUF:
        CLR     @YZ         ; @YZ清零。
        DECMS   Z           ;
        JMP     CLR_YZ_BUF  ;不为零。
        CLR     @YZ
END_CLR:                    ;
        …
```

4.1.2.6 、R 寄存器

8 位缓存器 R 主要有以下两个功能：

- 作为工作寄存器使用；
- 存储执行查表指令后的高字节数据。

（执行 MOVC 指令，指定 ROM 单元的高字节数据会被存入 R 寄存器而低字节数据则存入 ACC。）

表 4-7 R 寄存器（082H）

Bit	7	6	5	4	3	2	1	0
Name	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	X	X	X	X	X	X	X	X

4.1.3 、寻址模式

4.1.3.1 、立即寻址

将立即数送入 ACC 或指定的 RAM 单元。

例：立即数 12H 送入 ACC。

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

例：立即数 12H 送入寄存器 R。

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

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

4.1.3.2 、直接寻址

通过 ACC 对 RAM 单元数据进行操作。

例：地址 12H 处的内容送入 ACC。

```
B0MOV   A, 12H
```

例：ACC 中数据写入 RAM 的 12H 单元。

```
B0MOV   12H, A
```

4.1.3.3 、间接寻址

通过指针寄存器（Y/Z）访问 RAM 数据。

例：用@YZ 实现间接寻址。

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

4.1.4 、堆栈

4.1.4.1 、概述

堆栈缓存器共 4 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，STKnH 和 STKnL 分别是各堆栈缓存器的高、低字节。

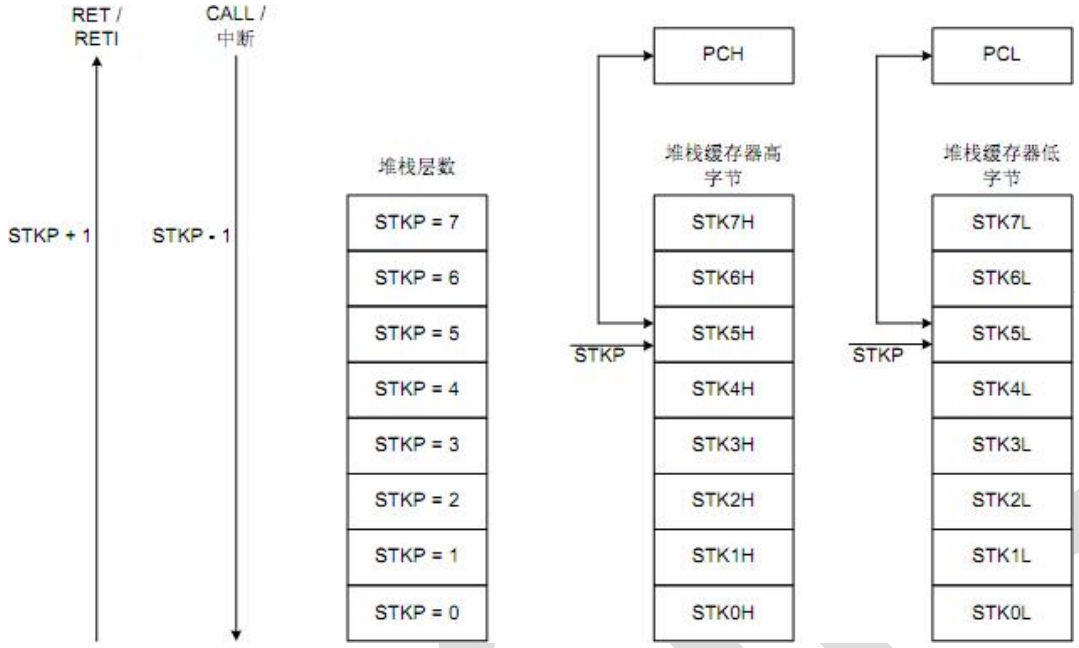


图 4-3 堆栈

4.1.4.2 、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，10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。

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

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

表 4-8 STKP 堆栈指针（0DFH）

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

位	字段	描述
7	GIE	全局中断控制位。 0 = 禁止； 1 = 使能。
2~0	STKPB2~ STKPB0	堆栈指针。

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

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

表 4-9 STKnH 堆栈数据寄存器 H（0F0H~0FFH）

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	-	-	SnPC10	SnPC9	SnPC8
R/W	-	-	-	-	-	R/W	R/W	R/W
POR	-	-	-	-	-	-	0	0

表 4-10 STKnL 堆栈数据寄存器 L（0F0H~0FFH）

Bit	7	6	5	4	3	2	1	0
Name	SnPC7	SnPC6	SnPC5	SnPC4	SnPC3	SnPC2	SnPC1	SnPC0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

注：STKn = STKnH，STKnL（n = 7 ~ 0）。

4.1.4.3 、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：

表 4-11 入栈操作

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

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

表 4-12 堆栈恢复操作

堆栈层数	STKP			缓存器		备注
	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	保留	保留	-

4.1.5、编译选项表（CODE OPTION）

表 4-13 编译选项表

编译选项	内容	功能说明
High_Clk	IHRC_16M	高速时钟采用内部 16MHz RC 振荡电路，XIN/XOUT（P0.3/P0.2）为普通的 I/O 引脚。
	RC	外部高速时钟振荡器采用廉价的 RC 振荡电路，XOUT（P0.2）为普通的 I/O 引脚。
	32K X'tal	外部高速时钟振荡器采用低频晶体/陶瓷振荡器（如 32.768kHz）
	12M X'tal	外部高速时钟振荡器采用高频晶体/陶瓷振荡器（如 12MHz）。
	4M X'tal	外部高速时钟振荡器采用标准晶体/陶瓷振荡器（如 4MHz）。
Watch_Dog	Always_On	始终开启看门狗定时器，即使在睡眠模式和绿色模式下也处于开启状态。
	Enable	开启看门狗定时器，但在睡眠模式和绿色模式下关闭。
	Disable	关闭看门狗定时器。
Fcpu	Fhosc/4	指令周期 =4 个时钟周期。
	Fhosc/8	指令周期 =8 个时钟周期。
	Fhosc/16	指令周期 =16 个时钟周期。
Reset_Pin	Reset	使能外部复位引脚。
	P03	P0.3 为单向输入引脚，无上拉电阻。
Security	Enable	ROM 程序加密。
	Disable	ROM 程序不加密。
Noise_Filter	Enable	使能杂讯滤除功能，Fcpu = Fosc/4~Fosc/16。
	Disable	禁止杂讯滤除功能，Fcpu = Fosc/1~Fosc/16。
LVD	LVD_L	VDD 低于 2.0V 时，系统复位。
	LVD_M	VDD 低于 2.0V 时，系统复位； PFLAG 寄存器的 LVD24 位作为 2.4V 低电压监测器。
	LVD_H	VDD 低于 2.4V 时，系统复位； PFLAG 寄存器的 LVD36 位作为 3.6V 低电压监测器。

5、通用功能

5.1、复位

5.1.1、概述

AiP8P102A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（仅在外部复位引脚处于使能状态）。

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

表 5-1 PFLAG 系统状态寄存器（086H）

Bit	7	6	5	4	3	2	1	0
Name	NT0	NPD	LVD36	LVD24	-	C	DC	Z
R/W	R/W	R/W	R	R	-	R/W	R/W	R/W
POR	X	X	0	0	-	0	0	0

NT0	NPD	复位情况	说明
0	0	看门狗复位	看门狗溢出
0	1	保留	-
1	0	上电及 LVD 复位	电源电压低于 LVD 检测值
1	1	外部复位	外部复位引脚检测到低电平

任何一种复位情况都需要一定的响应时间，系统提供完善的复位流程以保证复位动作的顺利进行。对于不同类型的振荡器，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。在用户终端使用的过程中，应注意考虑主机对上电复位时间的要求。

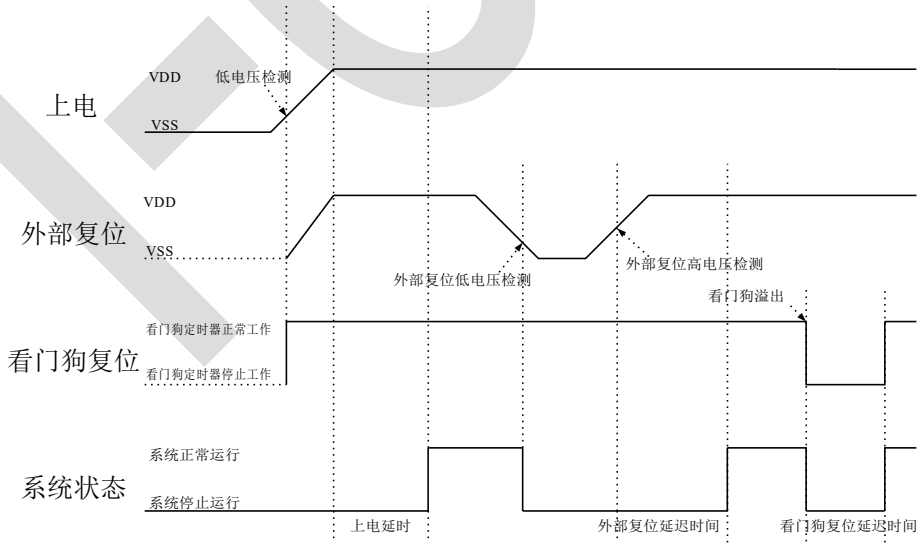


图 5-1 复位时间

5.1.2 、上电复位

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

- 上电：系统检测到电源电压上升并等待其稳定；
- 外部复位（仅限于外部复位引脚使能状态）：系统检测外部复位引脚状态。如果不为高电平，系统保持复位状态直到外部复位引脚释放；
- 系统初始化：所有的系统寄存器被置为初始值；
- 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。

5.1.3 、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：

- 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。

5.1.4 、掉电复位

5.1.4.1 、概述

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

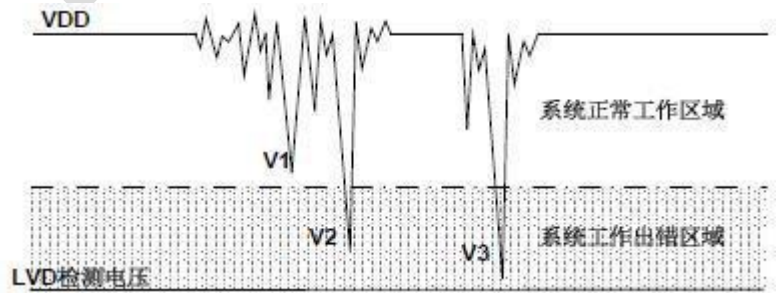


图 5-2 掉电复位示意图

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。

5.1.4.2 、系统工作电压

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

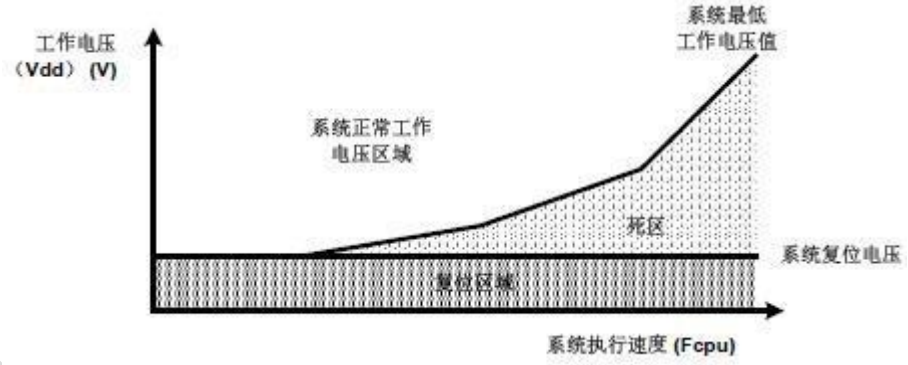


图 5-3 系统工作电压与执行速度关系图

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。

5.1.4.3 、掉电复位性能改进

如何改善系统掉电复位性能，有以下几点建议：

- LVD复位；
- 看门狗复位；

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

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

LVD 复位

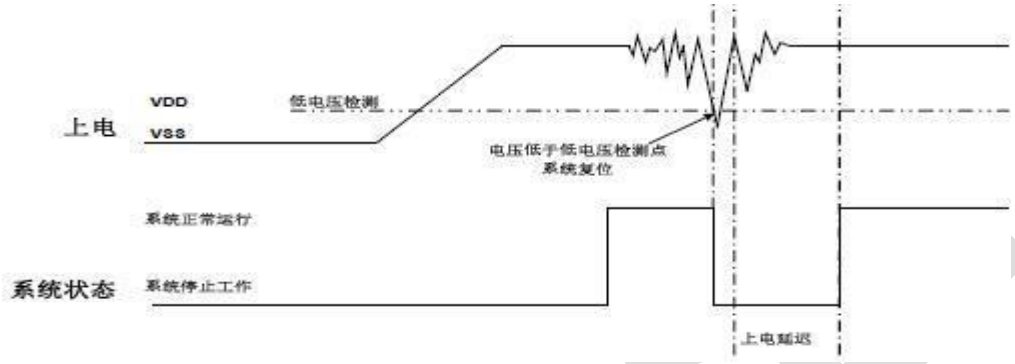


图 5-4 LVD 复位

表 5-2 PFLAG 寄存器（086H）

Bit	7	6	5	4	3	2	1	0
Name	NT0	NPD	LVD36	LVD24	-	C	DC	Z
R/W	R/W	R/W	R	R	-	R/W	R/W	R/W
POR	X	X	0	0	-	0	0	0

位	字段	描述
5	LVD36	3.6V LVD 工作电压标志，LVD 编译选项为 LVD_H 时有效。 0 =系统工作电压 VDD 超过 3.6V，低电压检测器没有工作； 1 =系统工作电压 VDD 低于 3.6V，说明此时低电压检测器已处于监控状态。
4	LVD24	2.4V LVD 工作电压标志，LVD 编译选项为 LVD_M 时有效。 0 =系统工作电压 VDD 超过 2.4V，低电压检测器没有工作； 1 =系统工作电压 VDD 低于 2.4V，说明此时低电压检测器已处于监控状态。

表 5-3 LVD 编译选项

LVD	LVD 编译选项		
	LVD_L	LVD_M	LVD_H
2.0V 复位	有效	有效	有效
2.4V 标志	-	有效	-
2.4V 复位	-	-	有效
3.6V 标志	-	-	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 <= 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 <= 3.6V，LVD36＝1；

注：1.LVD 复位结束后，LVD24 和 LVD36 都将被清零。

2.LVD2.4V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。

- 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。

5.1.5 、外部复位

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

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

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

5.1.6 、外部复位电路

5.1.6.1 、RC 复位电路

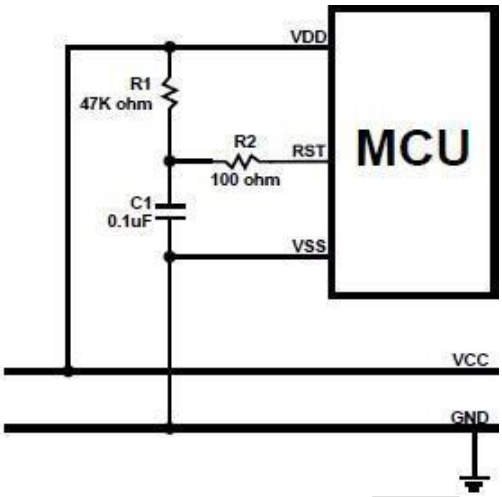


图 5-5 RC 复位电路

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。

5.1.6.2 、二极管及 RC 复位电路

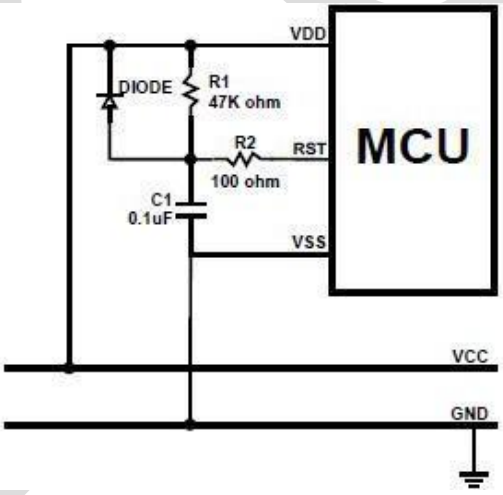


图 5-6 二极管及 RC 复位电路

上图中，R1 和 C1 同样是为复位引脚提供输入信号。对于电源异常情况，二极管正向导通使 C1 快速放电并与 VDD 保持一致，避免复位引脚持续高电平、系统无法正常复位。注：“基本RC复位电路”和“二极管及RC复位电路”中的电阻R2都是必不可少的限流电阻，以避免复位引脚ESD（Electrostatic Discharge）或EOS（Electrical Over-stress）击穿。

5.1.6.3 、稳压二极管复位电路

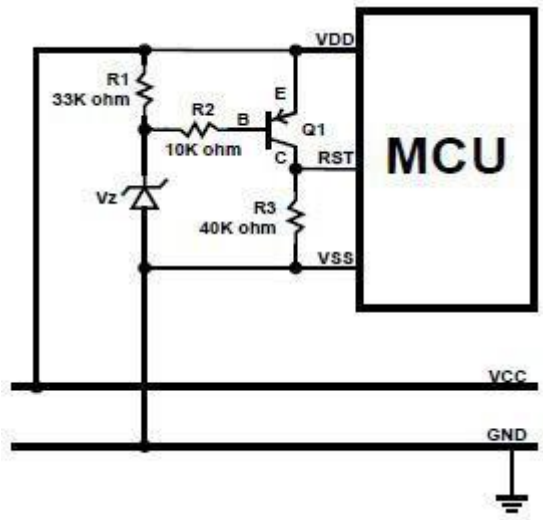


图 5-7 稳压二极管复位电路

稳压二极管复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。如上图电路中，利用稳压管的击穿电压作为电路复位检测值，当 VDD 高于“Vz + 0.7V”时， 三极管集电极输出高电平，单片机正常工作；当 VDD 低于“Vz + 0.7V”时，三极管集电极输出低电平，单片机复位。稳压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。

5.1.6.4 、电压偏置复位电路

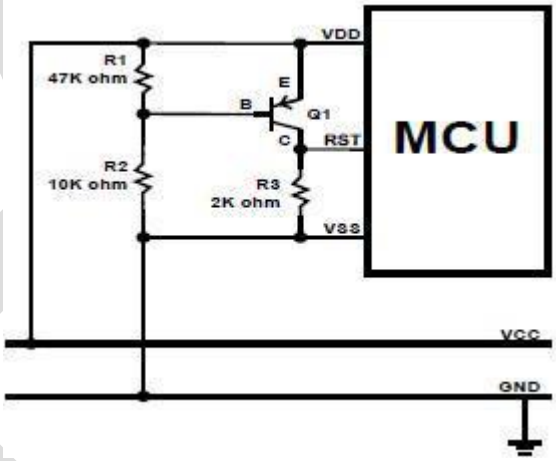


图 5-8 电压偏置复位电路

电压偏置复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，这种复位电路的检测电压值的精确度有所降低。电路中，R1 和R2 构成分压电路，当 VDD 高于和等于分压值“0.7V × (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 × (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。

5.1.6.5 、外部 IC 复位

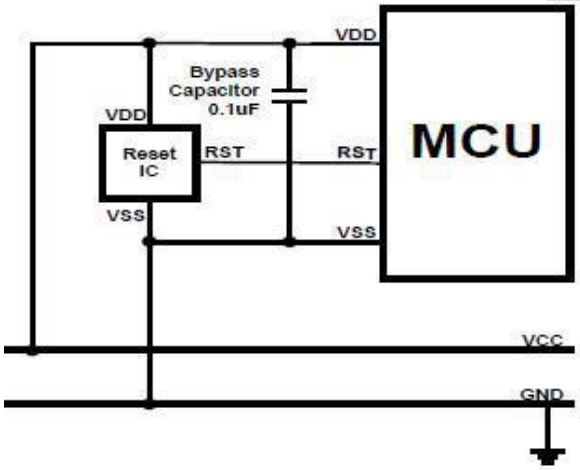


图 5-9 外部 IC 复位

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

5.2、系统时钟

5.2.1、概述

AIP8P102A 内带双时钟系统：高速时钟和低速时钟。高速时钟由外部振荡电路或内置 16MHZ 高速 RC 振荡电路（IHRC16MHz）提供，低速时钟则由内置低速 RC 振荡电路（ILRC16KHz@3V，32KHz@5V）提供。当系统在低速模式下工作时，时钟信号 4 分频之后作为系统指令周期 Fcpu。

- 普通模式（高速时钟）：Fcpu=Fhosc/N，N=4~16，Fcpu 编译选项决定 N 的值。
- 低速模式（低速时钟）：Fcpu=Flosc/4。

在干扰较严重的条件下，杂讯滤波功能能够对外部干扰进行隔离以保护系统的正常工作。

系统时钟框图

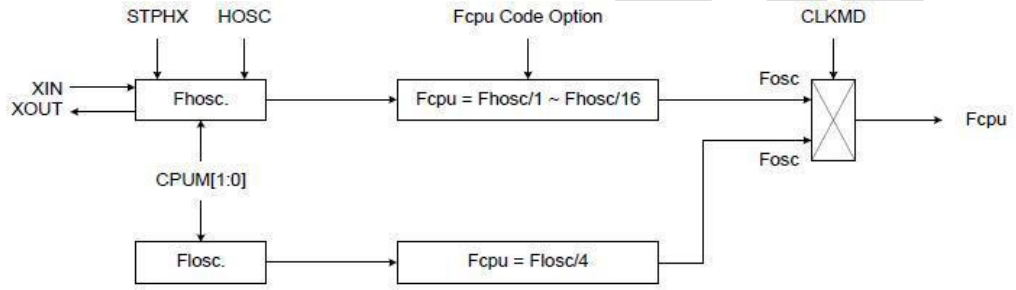


图 5-10 系统时钟框图

- HOSC：High_Clk 编译选项。
- Fhosc：外部高速/内部 RC 振荡器时钟频率。
- Flosc：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。

5.2.2、OSCM 寄存器

寄存器 OSCM 控制振荡器的状态和系统的工作模式。

表 5-4 OSCM 寄存器（0CAH）

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-
R/W	-	-	-	R/W	R/W	R/W	R/W	-
POR	-	-	-	0	0	0	0	-

位	字段	描述
4~3	CPUM1~0	CPU 工作模式控制位。 00 = 普通模式；01 = 睡眠模式； 10 = 绿色模式；11 = 系统保留。
2	CLKMD	系统时钟模式控制位。 0 = 普通（双时钟）模式，高速时钟作为系统时钟； 1 = 低速模式，低速时钟作为系统时钟。

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

例：停止高速振荡器。

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

例：系统进入睡眠模式时，高速振荡器和内部低速振荡器都被停止。

```
B0BSET      FCPUM0
```

5.2.3 、系统高速时钟

内部16MHZ RC振荡器或外部振荡器都可作为系统高速时钟源，由编译选项“High_Clk”控制。

表 5-5 时钟选择

HIGH_Clk	说明
IHRC_16M	内部 16MHz RC 振荡器作为系统时钟源，XIN 和 XOUT 引脚为通用 I/O 口。
RC	外部 RC 振荡器为系统高速时钟，XOUT 引脚为通用 I/O 口
32K	外部 327687Hz 低速振荡器为系统高速时钟
12M	外部高速振荡器作为系统高速时钟，典型频率为 12MHz
4M	外部振荡器作为系统高速时钟，典型频率为 4MHz

5.2.3.1 、内部高速 RC 振荡器

编译选项“IHRC_16M”控制单片机的内置 RC 高速时钟（16MHz），在“IHRC_16M”模式下，系统时钟来自内部 16MHz RC 振荡器，XIN/XOUT 引脚作为普通的 I/O 引脚。

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

5.2.3.2 、外部高速振荡器

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

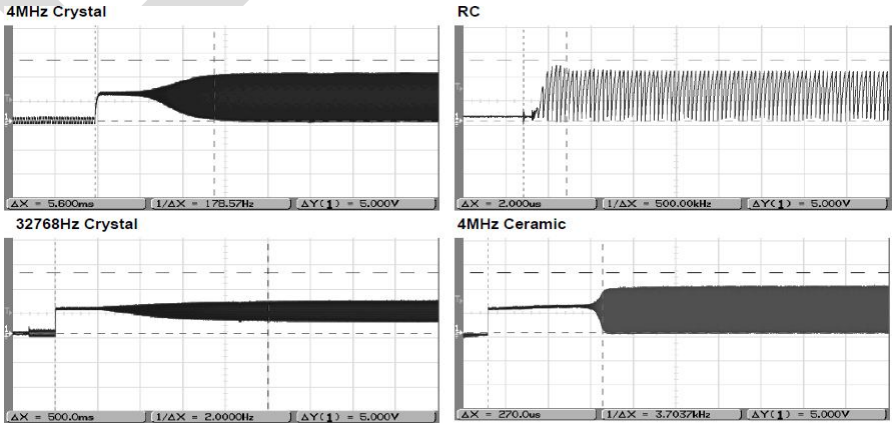


图 5-11 振荡起振时间

5.2.3.3 、石英/陶瓷振荡器

石英/陶瓷振荡器由XIN/XOUT口驱动，对于高速、普通和低速三种不同工作模式，振 荡器的驱动电流也不同。不同的工作模式下，编译选项“High_Clk”支持不同的频率：12MHz，4MHz及32KHz。

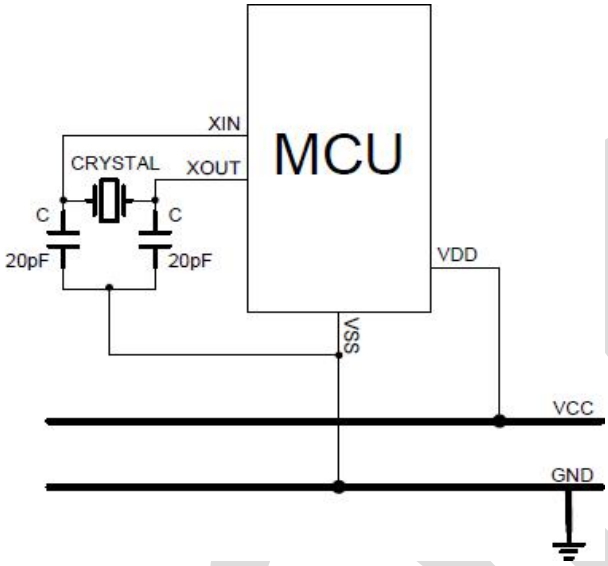


图 5-12 石英/陶瓷振荡器

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

5.2.3.4 、RC 振荡器

通过编译选项High_Clk的设置可控制RC振荡器的选择，RC振荡器输出频率最高可达10MHz。改变R可改变输出频率的大小，电容C的最佳容量为50P~100P，引脚XOUT为通用I/O口，如下图所示：

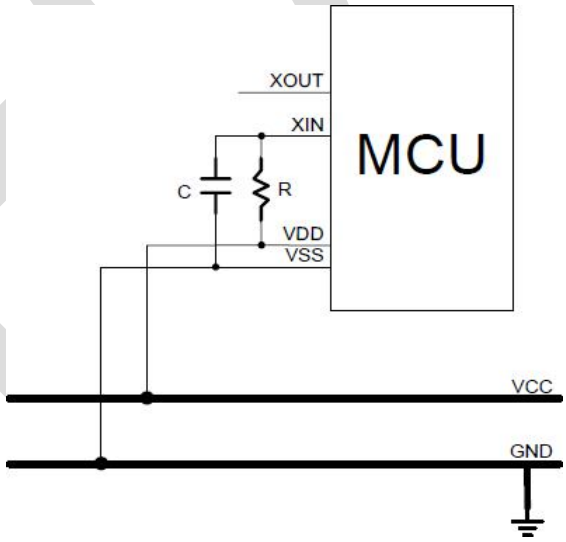


图 5-13 RC 振荡器

注：电容C和电阻R应尽可能的靠近单片机的VDD。

5.2.3.5 、外部时钟源

单片机可选择外部时钟信号作为系统时钟，由编译选项High_Clk控制，从XIN脚送入。XOUT引脚作为普通的I/O口。

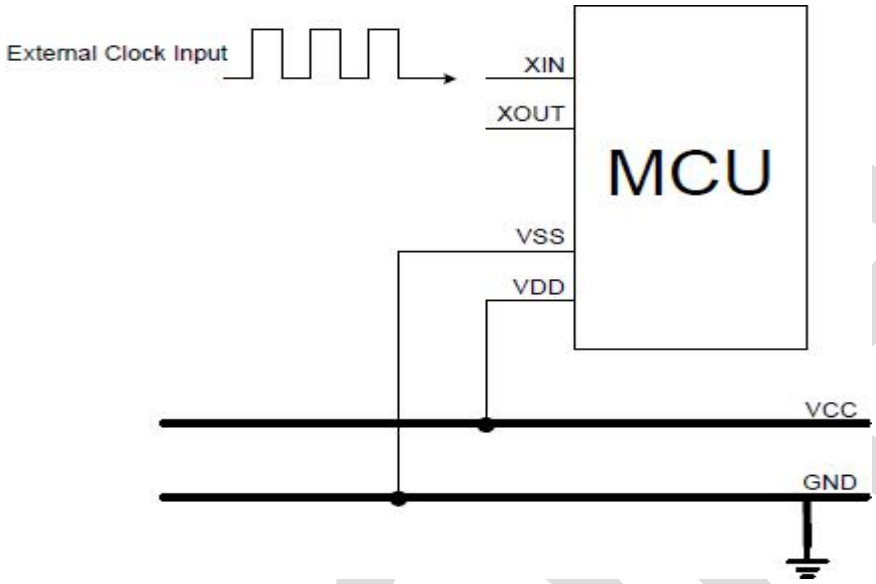


图 5-14 外部时钟源

注：外部振荡电路中的GND必须尽可能的接近单片机的VSS端口。

5.2.4 、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。

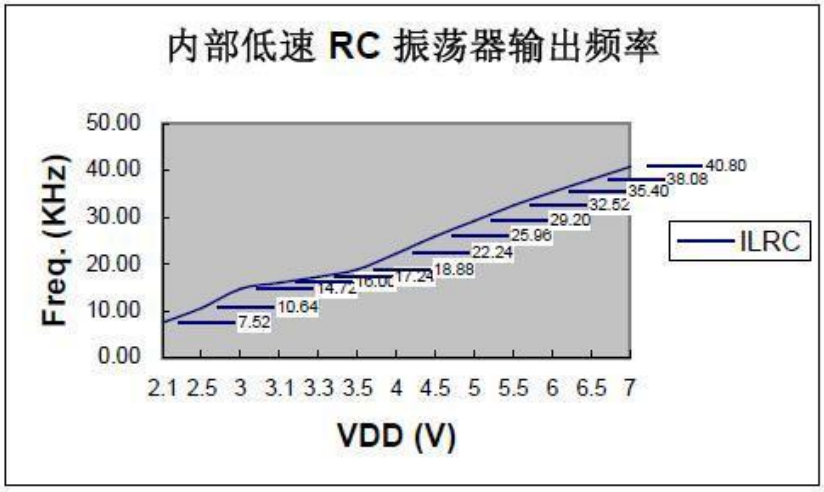


图 5-15 系统低速时钟频率与电压关系

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。

Flosc = 内部低速 RC 振荡器（16KHz @3V、32KHz @5V）。

低速模式 Fcpu = Flosc/4。

系统工作在睡眠模式下，可以停止低速 RC 振荡器。

例：停止内部低速振荡器。

```
B0BSET      FCPUM0
```

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

5.2.5 、系统时钟测试

在设计过程中，用户可通过软件指令周期 Fcpu 对系统时钟速度进行测试。

例：外部振荡器的 Fcpu 指令周期测试。

```
B0BSET      P0M.0      ; P0.0置为输出模式以输出Fcpu 的触发信号。
@@:
B0BSET      P0.0
B0BCLR      P0.0
JMP         @B
```

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

5.3、系统工作模式

5.3.1 、概述

AiP8P102A可以在以下4种工作模式下工作

- 普通模式（高速模式）；
- 低速模式；
- 睡眠模式；
- 绿色模式。

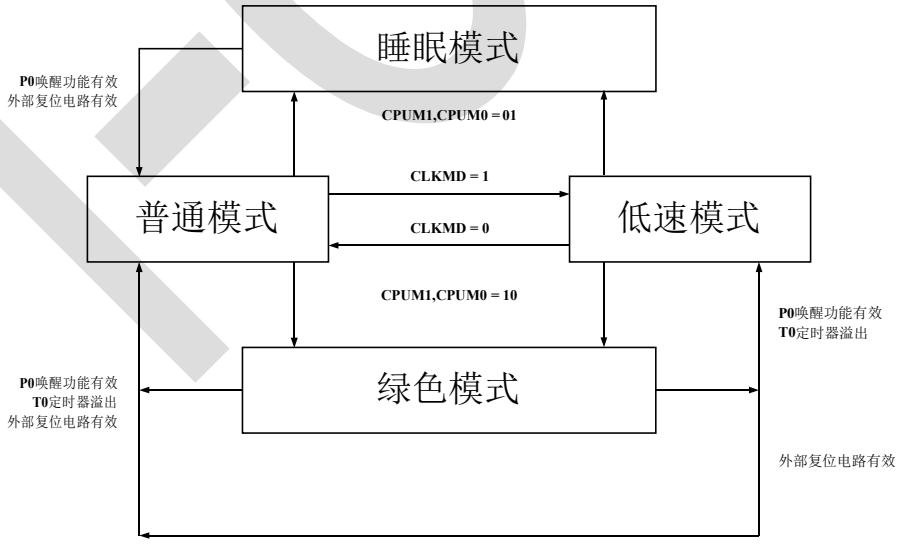


图 5-16 工作模式框图

表 5-6 工作模式说明

工作模式	普通模式	低速模式	绿色模式	睡眠模式
EHOSC	运行	STPHX 控制	STPHX 控制	停止
IHRC	运行	STPHX 控制	STPHX 控制	停止
ILRC	运行	运行	运行	停止
CPU 指令	执行	执行	停止	停止
T0	T0ENB 控制	T0ENB 控制	T0ENB 控制	无效
TC0	TC0ENB 控制	TC0ENB 控制	无效	无效
看门狗定时器	Watch_Dog 编译选项控制	Watch_Dog 编译选项控制	Watch_Dog 编译选项控制	Watch_Dog 编译选项控制
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，TC0，复位	P0，复位

- EHOSC：外部高速时钟。
- IHRC：内部高速时钟（16M RC 振荡器）。
- ILRC：内部低速时钟（RC 振荡器：3V 16K，5V 32K）

5.3.2 、系统模式切换举例

例：系统由普通/低速模式转换到睡眠模式。

```
B0BSET      FCPUM0
```

注：系统进入睡眠模式后，只有具有唤醒功能的引脚和复位信号能够将系统唤醒并回到普通模式中。

例：系统由普通模式转换为低速模式。

```
B0BSET      FCLKMD      ;设置 CLKMD 为 1，将其切换为低速模式。
B0BSET      FSTPHX      ;停止外部高速振荡器以降低功耗。
```

例：系统由低速模式转换到普通模式 **(**外部高速振荡器仍然工作**)**。例：

```
B0BCLR      FCLKMD      ;设置 CLKMD 为 0。
```

系统由低速模式转换到普通模式（外部高速振荡器停止工作）。

在外部高速时钟停振的情况下，系统回到普通模式时至少需要延迟10ms 以稳定振荡器。

```
        B0BCLR      FSTPHX      启动外部振荡器。
        MOV         A, #27      ;若 VDD = 5V，内部 RC=32KHz（典型值），
                                系统将延迟。
        B0MOV       Z, A
@@:     DECMS       Z           0.125ms × 81 = 10.125ms 以保证外部时钟稳
                                定。
        JMP         @B
        B0BCLR      FCLKMD      ;系统回到普通模式。
```

例：系统由普通模式/低速模式进入绿色模式。

```
B0BSET      FCPUM1      ;置 CPUM1=1。
```

注：绿色模式下如果禁止T0的唤醒功能，则只有具有唤醒功能的引脚和复位引脚可以将系统唤醒（具有唤醒功能的引脚将系统返回到上一个工作模式，复位引脚将系统返回到普通模式）。

例：系统由普通/低速模式进入绿色模式，并开启T0唤醒功能。

```
;设置T0定时器的唤醒功能。
B0BCLR      FT0IEN      ;禁止 T0 中断。
B0BCLR      FT0ENB      ;关闭 T0 定时器。
MOV         A,#20H
B0MOV       T0M,A       ; T0 时钟=Fcpu/64。
MOV         A,#64H
B0MOV       T0C,A       ; 设置 T0C 初始值=64H（T0 中断间隔=10ms）。
B0BCLR      FT0IEN      ; 禁止 T0 中断。
B0BCLR      FT0IRQ      ; T0 中断请求寄存器清零。
B0BSET      FT0ENB      ;开启 T0 定时器，进入绿色模式。
B0BCLR      FCPUM0      ;设置 CPUMx=10。
B0BSET      FCPUM1
```

注：绿色模式下如果使能T0的唤醒功能，则具有唤醒功能的引脚、复位引脚和T0都能够将系统唤醒回到上一工作模式。T0的唤醒周期可编程控制，请注意对T0ENB的设置

5.3.3 、系统唤醒

5.3.3.1 、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 的电平变换）和内部触发（T0 定时器溢出）。

- 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 电平变化）；
- 由绿色模式唤醒回到系统前一工作模式（普通模式或低速模式）可以用外部触发或者内部触发。

5.3.3.2 、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，单片机需要等待一段时间以等待振荡电路稳定工作，等待的这一段时间就称为唤醒时间。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。

外部高速振荡器的唤醒时间的计算如下：

唤醒时间=1/Fhosc*2048(sec)+高速时钟启动时间

注：高速时钟的启动时间与VDD和振荡器类型有关。

例：将系统从睡眠模式中唤醒，并设置系统进入普通模式。唤醒时间计算如下：

唤醒时间=1/Fosc * 2048=0.512 ms (Fosc=4MHz)

总的唤醒时间=0.512 ms +振荡器启动时间

5.4、中断

5.4.1、概述

AiP8P102A 提供 4 个中断源：3 个内部中断（T0/TC0/ADC）和 1 个外部中断（INT0）。外部中断可以将系统从睡眠模式中唤醒进入高速模式，在返回到高速模式前，中断请求被锁定。一旦程序进入中断，寄存器 STKP 的位 GIE 被硬件自动清零以避免响应其它中断。系统退出中断后，硬件自动将GIE 置“1”，以响应下一个中断。中断请求存放在寄存器INTRQ中。

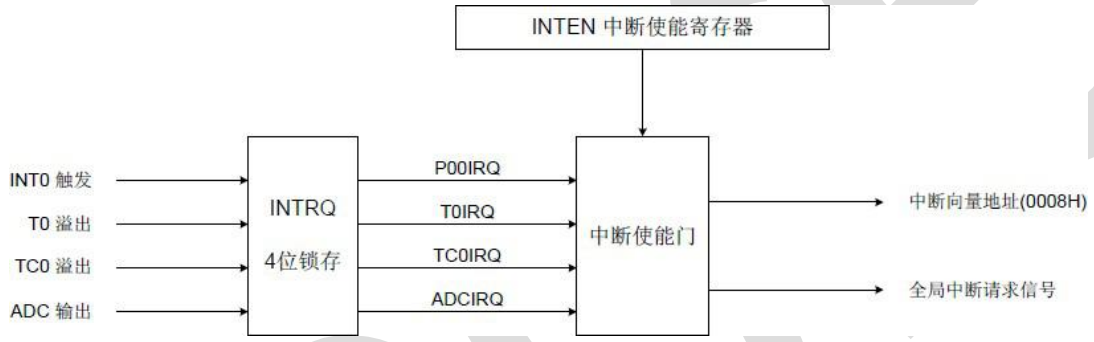


图 5-17 中断

注：程序响应中断时，必须开启全局中断控制位GIE。

5.4.2、中断请求使能寄存器 INTEN

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

表 5-7 INTEN 中断请求控制器（0C9H）

Bit	7	6	5	4	3	2	1	0
Name	ADCIEN		TC0IEN	T0IEN	-	-		P00IEN
R/W	R/W		R/W	R/W	-	-		R/W
POR	0		0	0	-	-		0

位	字段	描述
7	ADCIEN	ADC 中断控制位 0 = 无效； 1 = 有效。
5	TC0IEN	TC0 中断控制位 0 = 无效； 1 = 有效。

4	T0IEN	T0 中断控制位 0 = 无效； 1 = 有效。
0	P00IEN	P0.0 外部中断（INT0）控制位 0 = 无效； 1 = 有效。

5.4.3 、中断请求寄存器 INTRQ

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

表 5-8 INTRQ 中断请求寄存器（0C8H）

Bit	7	6	5	4	3	2	1	0
Name	ADCIRQ		TC0IRQ	T0IRQ	-	-		P00IRQ
R/W	R/W		R/W	R/W	-	-		R/W
POR	0		0	0	-	-		0

位	字段	描述
7	ADCIRQ	ADC 中断请求标志 0 = ADC 无中断请求； 1 = ADC 有中断请求。
5	TC0IRQ	TC0 中断请求标志 0 = TC0 无中断请求； 1 = TC0 有中断请求。
4	T0IRQ	T0 中断请求标志 0 = T0 无中断请求； 1 = T0 有中断请求。
0	P00IRQ	P0.0 中断（INT0）请求标志 0 = INT0 无中断请求； 1 = INT0 有中断请求。

5.4.4 、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（ORG8），堆栈层数加 1。

表 5-9 STKP 寄存器（0DFH）

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

位	字段	描述
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。

5.4.5 、PUSH, POP 处理

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

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

例：对ACC和PAFLG进行入栈保护。

```
            ORG         0
            JMP         START
            ORG         8H
            JMP         INT_SERVICE
            ORG         10H
START:
            …
INT_SERVICE:
            PUSH                    ;保存ACC 和PFLAG。
            …
            POP                     ;恢复ACC和PFLAG
            RETI                    ; 退出中断。

            …
            ENDP
```

5.4.6 、INT0（P0.0）中断

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

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

注：INT0的中断请求被P0.0的唤醒触发功能锁定。

注：P0.0的中断触发边沿由PEDGE控制。

表 5-10 PEDGE：边沿选择寄存器（0BFH）

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

POR	-	-	-	1	0	-	-	-

位	字段	描述
4:3	P00G[1:0]	P0.0 中断触发控制位 00 = 保留； 01 = 上升沿触发； 10 = 下降沿触发； 11 = 上升/下降沿触发（电平触发）。

例：INT0中断请求设置，电平触发。

```
        MOV     A, #18H
        B0MOV   PEDGE, A        ; 设置INT0为电平触发。
        B0BCLR  FP00IRQ         ; 清INT0中断请求标志。
        B0BSET  FP00IEN         ; 使能INT0中断。
        B0BSET  FGIE            ; 使能GIE。
```

例：INT0中断。

```
        ORG     8H              ;
        JMP     INT_SERVICE
INT_SERVICE:
        …                       ; 保存ACC和PFLAG。
        B0BTS1  FP00IRQ         ; 检查是否有P00中断请求标志。
        JMP     EXIT_INT        ; P00IRQ = 0，退出中断。
        B0BCLR  FP00IRQ         ; 清P00IRQ。
        …                       ; INT0中断服务程序。

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

5.4.7 、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          ;
        MOV     A, #20H         ;
        B0MOV   T0M, A          ; T0时钟＝Fcpu / 64。
        MOV     A, # 64H        ; T0C初始值＝64H。
```

```
        B0MOV   T0C, A          ; T0间隔= 10 ms。
        B0BCLR  FT0IRQ          ; 清T0中断请求标志。
        B0BSET  FT0IEN          ; 使能T0中断。
        B0BSET  FT0ENB          ;
        B0BSET  FGIE            ; 使能GIE。
```

例：TC0 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          ; 清T0C。
        …                       ; T0中断程序。
        …
EXIT_INT:
        …                       ; 恢复ACC和PFLAG。
        RETI                    ; 退出中断。
```

5.4.8 、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              ;
```

```
                JMP         INT_SERVICE
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。
```

5.4.9 、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 P4.0ADC 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          ;
                JMP         INT_SERVICE
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。
```

5.4.10 、多中断操作举例

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

表 5-11 中断触发事件表

中断	有效触发
P00IRQ	由 PEDGE 控制
T0IRQ	T0C 溢出
TC0IRQ	TC0C 溢出
ADCIRQ	AD 转换完成

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

例：多中断条件下检测中断请求。

```
                ORG     8H              ;
                JMP     INT_SERVICE
INT_SERVICE:
                …                       ; 保存ACC和PFLAG。
INTP00CHK:                              ; 检查是否有P00中断请求。
                B0BTS1  FP00IEN         ; 检查是否使能P00中断。
                JMP     INTP01CHK       ; 跳到下一个中断。
                B0BTS0  FP00IRQ         ; 检查是否有P00中断请求。
                JMP     INTP00          ; 进入INT0中断。
INTP01CHK:                              ; 检查是否有P01中断请求。
                B0BTS1  FP01IEN         ; 检查是否使能P01中断。
                JMP     INTTC0CHK       ; 跳到下一个中断。
                B0BTS0  FP01IRQ         ; 检查是否有P01中断请求。
                JMP     INTP01          ; 进入INT1中断。
INTTC0CHK:                              ; 检查是否有TC0中断请求。
                B0BTS1  FTC0IEN         ; 检查是否使能TC0中断。
```

```
                JMP     INTTC1CHK   ; 跳到下一个中断。
                B0BTS0  FTC0IRQ     ; 检查是否有TC0中断请求。
                JMP     INTTC0      ; 进入TC0中断。
INTTC1CHK:                          ; 检查是否有TC1中断请求。
                B0BTS1  FTC1IEN     ; 检查是否使能TC1中断。
                JMP     INTADCHK    ; 跳到下一个中断。
                B0BTS0  FTC1IRQ     ; 检查是否有TC1中断请求。
                JMP     INTTC1      ; 进入TC1中断。
INTADCHK:                           ; 检查是否有ADC中断请求。
                B0BTS1  FADCIEN     ; 检查是否使能ADC中断。
                JMP     INT_EXIT    ;
                B0BTS0  FADCIRQ     ; 检查是否有ADC中断请求。
                JMP     INTADC      ; 进入ADC中断。
INT_EXIT:
                …                   ; 恢复ACC和PFLAG。
                RETI                ; 退出中断。
```

5.5、I/O 口

5.5.1 、I/O 口模式

寄存器 PnM 控制 I/O 口的工作模式。

表 5-12 P0M 寄存器（0B8H）

Bit	7	6	5	4	3	2	1	0
Name	P07M	P06M	P05M	P04M	-	P02M	P01M	P00M
R/W	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W
POR	0	0	0	0	-	0	0	0

表 5-13 P4M 寄存器（0C4H）

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	P44M	P43M	P42M	P42M	P40M
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
POR	-	-	-	0	0	0	0	0

表 5-14 P5M 寄存器（0C5H）

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	P54M	P53M	P52M	P51M	P50M
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
POR	-	-	-	0	0	0	0	0

位	字段	描述
7：0	PnM~ PnM[7:0]	Pn 模式控制位（n = 0、4、5） 0 = 输入模式； 1 = 输出模式。

注：1、用户可以用位操作指令（B0BSET，B0BCLR）对I/O口进行操作；

2、P0.3 是单向输入引脚， P0.3M = 1。

例：I/O 模式选择。

```
CLR     P0M          ;所有端口置为输入模式。
CLR     P4M
CLR     P5M
MOV     A, #0FFH     ;所有端口置为输出模式。
B0MOV   P0M,A
B0MOV   P4M,A
B0MOV   P5M, A
B0BCLR  P4M.0        ; P4.0置为输入模式。
B0BSET  P4M.0        ; P4.0置为输出模式。
```

5.5.2 、I/O 上拉电阻寄存器

表 5-15 P0UR 寄存器（0E0H）

Bit	7	6	5	4	3	2	1	0
Name	P07R	P06R	P05R	P04R	-	P02R	P01R	P00R
R/W	W	W	W	W	-	W	W	W
POR	0	0	0	0	-	0	0	0

表 5-16 P4UR 寄存器（0E4H）

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	P44R	P43R	P42R	P41R	P40R
R/W	-	-	-	W	W	W	W	W
POR	-	-	-	0	0	0	0	0

表 5-17 P5UR 寄存器（0E5H）

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	P54R	P53R	P52R	P51R	P50R
R/W	-	-	-	W	W	W	W	W
POR	-	-	-	0	0	0	0	0

注：P0.4是单向输入引脚，无上拉电阻，因此P0UR.4始终为“1”。

例：I/O 上拉电阻。

```
MOV     A, #0FFH     ;使能P0、4、5上拉电阻。
B0MOV   P0UR, A      ;
B0MOV   P4UR,A
B0MOV   P5UR, A
```

5.5.3 、I/O 口数据寄存器

表 5-18 P0 数据寄存器（0D0H）

Bit	7	6	5	4	3	2	1	0
Name	P07	P06	P05	P04	P03	P02	P01	P00
R/W	W	W	W	W	W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

表 5-19 P4 数据寄存器（0D4H）

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	P44	P43	P42	P41	P40
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
POR	-	-	-	0	0	0	0	0

表 5-20 P5 数据寄存器（0D5H）

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	P54	P53	P52	P51	P50
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
POR	-	-	-	0	0	0	0	0

注：使能外部复位时，P04的值保持为“1”。

例：从输入端口读取数据。

```
B0MOV   A, P0      ; 读P0口的数据。
B0MOV   A, P4      ; 读P4口的数据。
B0MOV   A, P5      ; 读P5口的数据。
```

例：写数据到输出端口。

```
MOV     A, #0FFH   ; 写0FFH到所有的端口。
B0MOV   P0, A
B0MOV   P4, A
B0MOV   P5, A
```

例：写1位数据到输出端口。

```
B0BSET  P4.0       ; P4.0和P5.3设为“1”
B0BSET  P5.3
B0BCLR  P4.0       ; P4.0和P5.3为设“0”。
B0BCLR  P5.3
```

5.5.4 、P4 口 ADC 共用引脚

P4 口和 ADC 的输入口共用，非施密特触发。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 VDD 时，将可能产生额外的漏电流。同样，当 P4 口外接多个模拟信号时，也会产生额外的漏电流。在睡眠模式下，上述漏电流会严重影响到系统的整体功耗。P4CON 为 P4 口的配置寄存器。将 P4CON[7:0]置“1”，其对应的 P4 口将被设置为纯模拟信号输入口，从而避免上述漏电流的情况。

表 5-21 P4CON 寄存器（0AEH）

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
POR	-	-	-	0	0	0	0	0

位	字段	描述
4:0	P4CON[4:0]	P4.n 控制位

		0 = P4.n 作为模拟信号输入或普通 I/O 引脚； 1 = P4.n 作为仅作模拟信号输入引脚。

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。

表 5-22 ADM 寄存器（0B1H）

Bit	7	6	5	4	3	2	1	0
Name	ADENB	ADS	EOC	GCHS	-	CHS2	CHS1	CHS0
R/W	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W
POR	0	0	0	0	-	0	0	0

位	字段	描述
4	GCHS	ADC 输入通道控制位 0 = 禁止 AIN 通道； 1 = 开启 AIN 通道。
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。

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，CHS[2:0]所指向的P4.n会被自动设为ADC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     ; CHS[2:0]指向P4.1（CHS[2:0] = 001B），则GCHS=0。
                            ; CHS[2:0]没指向P4.1（CHS[2:0] ≠ 001B），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。

```
        B0BCLR    FGCHS     ; CHS[2:0]指向P4.1（CHS[2:0] = 001B），则GCHS=0。
                            ; CHS[2:0]没指向P4.1（CHS[2:0] ≠ 001B），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    ;P4.1设为输出模式。
```

6、外设模块

6.1、定时器

6.1.1、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（16KHz @3V，32KHz @5V）提供。

看门狗溢出时间= 8192/内部低速振荡器周期（sec）

表 6-1 看门狗工作模式

VDD	内部低速 RC Freq.	看门狗溢出时间
3V	16KHz	512ms
5V	32KHz	256ms

注：如果看门狗被置为“Always_On”模式，那么看门狗在睡眠模式和绿色模式下仍然运行。

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

表 6-2 WDTR 看门狗清零寄存器（0CCH）

Bit	7	6	5	4	3	2	1	0
Name	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0
R/W	W	W	W	W	W	W	W	W
POR	0	0	0	0	0	0	0	0

例：如下是对看门狗定时器的操作，在主程序开头对看门狗清零。

```
        A,#5AH                          ; 看门狗定时器清零。
        B0MOV       WDTR,A
        …
        CALL        SUB1
        CALL        SUB2
        …
        …
        JMP         MAIN
```

看门狗定时器应用注意事项如下：

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

例：如下是对看门狗定时器的操作，在主程序开头对看门狗清零。

```
main:
        …                               ; 检测I/O口的状态。
        …                               ; 检测RAM的内容。
Err:    JMP $                           ; I/O或RAM出错，不清看门狗等
```

```
                                        看门狗计时溢出。
Correct:                                ; I/O和RAM正常，看门狗清零。
                                        ;
        MOV         A, #5AH             ;在整个程序中只有一处地方清
                                        看门狗。
        B0MOV       WDTR, A
        …
        CALL        SUB1
        CALL        SUB2
        …
        …
        JMP         MAIN
```

6.1.2 、定时器 T0

6.1.2.1 、概述

8位二进制计数器T0溢出时（由0FFH计到00H），T0在继续计数的同时给出一个溢出信号触发T0 中断。计数器T0主要有以下功能：

- 8位可编程定时器：根据选定的时钟频率定时产生中断；
- 绿色模式唤醒功能：在T0ENB=1的条件下，T0的溢出可将系统从绿色模式中唤醒。

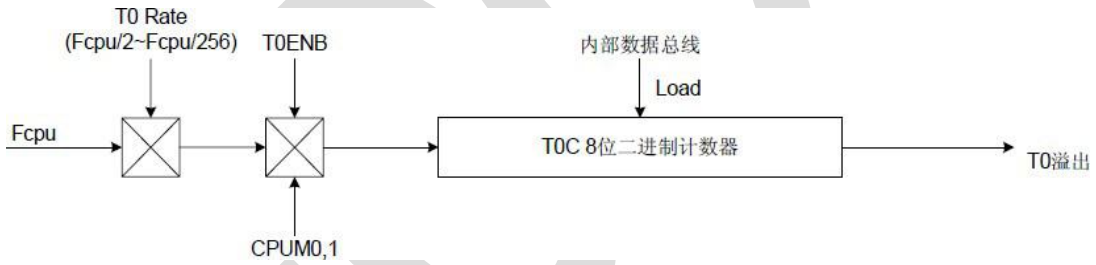


图 6-1 定时器 T0

6.1.2.2 、T0M 模式寄存器

表 6-3 T0M 寄存器（0D8H）

Bit	7	6	5	4	3	2	1	0
Name	T0ENB	T0rate2	T0rate1	T0rate0				
R/W	R/W	R/W	R/W	R/W				
POR	0	0	0	0				

位	字段	描述
7	T0ENB	T0 启动控制位 0 =关闭； 1 =开启。
6:4	T0RATE[2:0]	T0 分频选择位 000=Fcpu/256; 001= Fcpu/128;

		… 110= Fcpu/4; 111= Fcpu/2;

6.1.2.3 、T0C 计数寄存器

T0C 用于控制 T0 的间隔时间。

表 6-4 T0C 寄存器（0D9H）

Bit	7	6	5	4	3	2	1	0
Name	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

T0C初始值计算公式如下：

T0C初始值= 256-（T0中断间隔×时间输入时钟）

例：中断间隔时间设置为10ms，高速时钟选择外部4MHz，Fcpu=Fosc/4，T0RATE=010（Fcpu/64）。

T0C初始值= 256-（T0中断间隔时间×输入时钟）

= 256 -（10ms×4MHz / 4 / 64）

= 256 -（10-2×4×106 / 4 / 64）

= 100

= 64H

表 6-5 参数设定表

T0RATE	T0CLOCK	高速模式（Fcpu=4M/4）		低速模式（Fcpu=32768Hz/4）	
		最大溢出间隔	单步间隔 =max/256	最大溢出间隔	单步间隔 =max/256
000	Fcpu/256	65.536 ms	256 us	8000 ms	31250 us
001	Fcpu/128	32.768 ms	128 us	4000 ms	15625 us
010	Fcpu/64	16.384 ms	64 us	2000 ms	7812.5 us
011	Fcpu/32	8.192 ms	32 us	1000 ms	3906.25 us
100	Fcpu/16	4.096 ms	16 us	500 ms	1953.25 us
101	Fcpu/8	2.048 ms	8 us	250 ms	976.563 us
110	Fcpu/4	1.024 ms	4 us	125 ms	488.281 us
111	Fcpu/2	0.512 ms	2 us	62.5 ms	244.141 us

6.1.2.4 、T0 操作流程

- T0停止计数，禁止T0中断并将T0中断请求标志清零。

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

- 设置T0计时速率。

```
MOV       A,#0xxx00b          ; T0M的bit4~bit6 位可控制T0的计数速
                                率为x000xxxxb~x111xxxxb。
B0MOV     T0M,A
```

- 设置T0中断间隔时间。

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

- 设置T0工作模式。

```
B0BSET  FT0IEN
```

- 开启T0。

```
B0BSET  FT0ENB
```

6.1.3 、定时/计数器 TC0

6.1.3.1 、概述

定时/计数器TC0具有双时钟源，可根据实际需要选择内部时钟或外部时钟作为计时标准。其中，内部时钟来自Fcpu，外部时钟INT0由P0.0引脚（下降沿触发）输入。寄存器TC0M 控制时钟源的选择。当TC0从0FFH溢出到00H时， TC0在继续计数的同时产生一个溢出信号，触发TC0中断请求。

TC0的主要用途如下：

- 8位可编程定时器：根据选定的时钟频率在特定时间产生中断信号；
- 外部事件计数：对外部事件计数；
- 蜂鸣器输出；
- PWM输出。

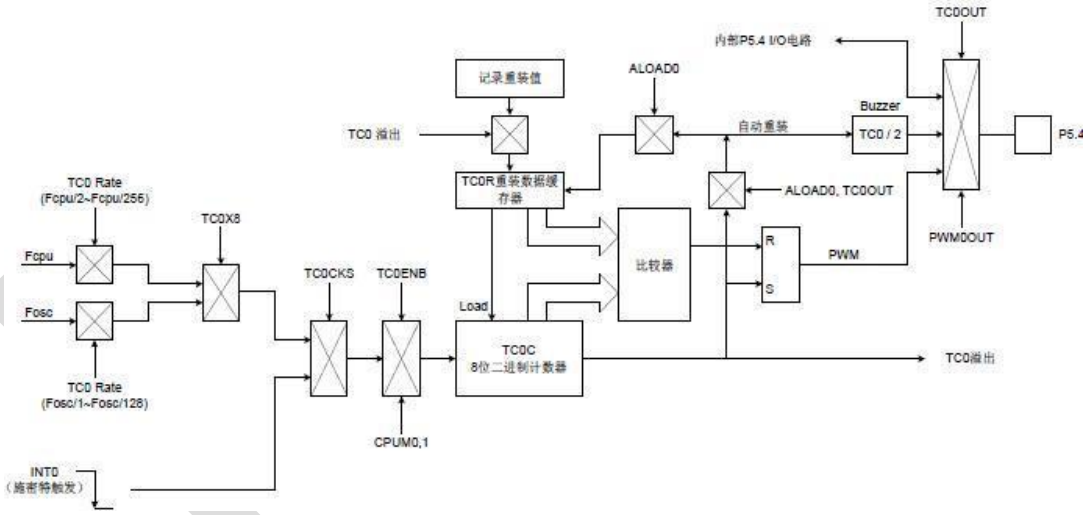


图 6-2 TC0 框图

6.1.3.2 、TC0M 模式寄存器

表 6-6 TC0M 寄存器（0DAH）

Bit	7	6	5	4	3	2	1	0
Name	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

位	字段	描述
7	TC0ENB	TC0 启动控制位 0 = 关闭； 1 = 开启。
6:4	TC0RATE[2:0]	TC0 分频选择位 TC0RATE [2:0] | 000 | Fcpu/256 001 | Fcpu/128 010 | Fcpu/64 011 | Fcpu/32 100 | Fcpu/16 101 | Fcpu/8 110 | Fcpu/4 111 | Fcpu/2
3	TC0CKS	TC0 时钟信号控制位 0 = 内部时钟（Fcpu 或 Fosc）； 1 = 外部时钟，由 P0.0/INT0 输入。
2	ALOAD0	自动装载控制位。仅当 PWM0OUT = 0 时有效。 0 = 禁止 TC0 自动重装； 1 = 允许 TC0 自动重装。
1	TC0OUT	TC0 溢出信号输出控制位。仅当 PWM0OUT = 0 时有效。 0 = 禁止，P5.4 作为输入/输出口； 1 = 允许，P5.4 输出 TC0OUT 信号。
0	PWM0OUT	PWM 输出控制 0 = 禁止 PWM 输出； 1 = 使能 PWM 输出，PWM 输出占空比由 T0OUT 和 ALOAD0 控制。

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

6.1.3.3 、TC0C 计数寄存器

TC0C 控制 TC0 的时间间隔。

表 6-7 TC0C 寄存器（0DBH）

Bit	7	6	5	4	3	2	1	0
Name	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
POR	0	0	0	0	0	0	0	0

TC0C 初始值计算公式如下：

TC0C 初始值= N -（TC0 中断间隔时间×输入时钟）

N 为 TC0 二进制计数范围。各模式下参数的设定如下表所示：

表 6-8 参数设定表

TC0CKS	PWM0	ALOAD0	TC0OUT	N	TC0C 有效值	TC0C 二进制计数范围	备注
0	0	x	x	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出
	1	0	0	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出

	1	0	1	64	00H~3FH	xx000000b~ xx111111b	每计数 64 次溢出
	1	1	0	32	00H~1FH	xxx00000b~ xxx11111b	每计数 32 次溢出
	1	1	1	16	00H~0FH	xxxx0000b~ xxxx1111b	每计数 16 次溢出
1	-	-	-	256	00H~0FFH	00000000b~ 11111111b	每计数 256 次溢出

例：TC0中断时间设为10ms，时钟源选择Fcpu（TC0KS = 0），无PWM输出（PWM0=0），高速时钟 = 4MHz.

Fcpu = Fosc/4，TC0RATE = 010（Fcpu/64）。

TC0C 初始值 = N -（TC0 中断时间×输入时钟）

= 256 -（10ms×4MHz / 4 / 64）

= 256 -（10-2×4×106 / 4 / 64）

= 100

= 64H

表 6-9 模式选择

TC0RATE	TC0CLOCK	高速模式（Fcpu=4M/4）		低速模式（Fcpu=32768Hz/4）	
		最大溢出间隔时间	单步间隔时间 =max/256	最大溢出间隔时间	单步间隔时间 =max/256
000	Fcpu/256	65.536 ms	256 us	8000 ms	31250 us
001	Fcpu/128	32.768 ms	128 us	4000 ms	15625 us
010	Fcpu/64	16.384 ms	64 us	2000 ms	7812.5 us
011	Fcpu/32	8.192 ms	32 us	1000 ms	3906.25 us
100	Fcpu/16	4.096 ms	16 us	500 ms	1953.25 us
101	Fcpu/8	2.048 ms	8 us	250 ms	976.563 us
110	Fcpu/4	1.024 ms	4 us	125 ms	488.281 us
111	Fcpu/2	0.512 ms	2 us	62.5 ms	244.141 us

6.1.3.4 、TC0R 自动装载寄存器

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

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 中断时间出错以及 PWM 误动作。

注：在PWM模式下，系统自动开启重装功能，ALOAD0用于控制溢出范围。

表 6-10 TC0R 自动装载寄存器（0CDH）

Bit	7	6	5	4	3	2	1	0
Name	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0
R/W	W	W	W	W	W	W	W	W
POR	0	0	0	0	0	0	0	0

TC0R 初始值计算公式如下：

TC0R 初始值=N-（TC0 中断间隔时间×输入时钟）

N 是 TC0 最大溢出值。TC0 的溢出时间和有效值见下表：

表 6-11 TC0 溢出时间及有效值表

TC0CKS	PWM0	ALOAD1	TC0OUT	N	TC0C 有效值	TC0C 二进制计数范围	备注
0	0	x	x	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出
	1	0	0	256	00H~0FFH	00000000b~11111111b	每计数 256 次溢出
	1	0	1	64	00H~3FH	xx000000b~xx111111b	每计数 64 次溢出
	1	1	0	32	00H~1FH	xxx00000b~xxx11111b	每计数 32 次溢出
	1	1	1	16	00H~0FH	xxxx0000b~xxxx1111b	每计数 16 次溢出
1	-	-	-	256	00H~0FFH	00000000b~11111111b	计数 256 次溢出

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

TC0R = N -（TC0 中断间隔时间×输入时钟）

= 256-（10ms× 4MHz/ 4/64）

= 256-（10-2× 4×106/4/ 64）

= 100

= 64H

6.1.3.5 、TC0 时钟频率输出（蜂鸣器输出）

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

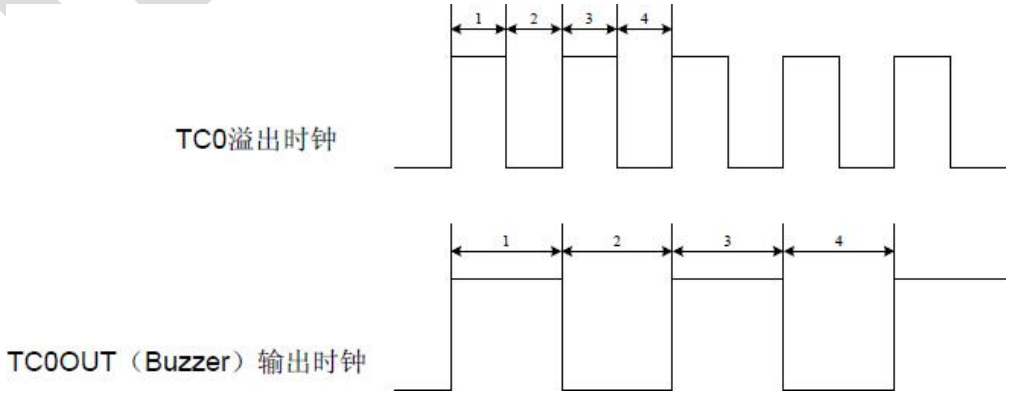


图 6-3 TC0 时钟频率输出

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

设置 TC0OUT（P5.4）。

```
MOV     A,#01100000B
B0MOV   TC0M,A      ; TC0速率=Fcpu/4。
MOV     A,#131      ; 自动加载参考值设置。
B0MOV   TC0C,A
B0MOV   TC0R,A
B0BSET  FTC0OUT     ; TC0的输出信号由P5.4输出，禁止P5.4的普通I/O功能。
B0BSET  FALOAD0     ; 使能TC0自动装载功能。
B0BSET  FTC0ENB     ; 开启TC0定时器。
```

注：蜂鸣器的输出有效时，“PWM0OUT”必须被置为0。

6.1.3.6 、TC0 操作举例

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

例：停止TC0计数器，禁止TC0中断并将TC0中断请求标志清零。

```
B0BCLR  FTC0ENB
B0BCLR  FTC0IEN
B0BCLR  FTC0IRQ
```

例：设置TC0的速率（不包含事件计数模式）。

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

例：设置TC0的时钟源。

```
B0BCLR  FTC0CKS         ; 选择内部时钟。
```

或

```
B0BSET  FTC0CKS         ; 选择外部时钟。
```

例：TC0自动装载模式设置。

```
B0BCLR  FALOAD0         ; 禁止自动装载功能。
```

或

```
B0BSET  FALOAD0         ; 使能TC0自动装载。
```

例：TC0中断间隔时间设置。

```
MOV     A,#7FH          ; TC0模式决定TC0C和TC0R的值。
B0MOV   TC0C,A          ; 设置TC0C。
B0MOV   TC0R,A          ; 设置TC0R。
B0BCLR  FALOAD0         ; ALOAD0，TC0OUT = 00，PWM周期= 0~255。
B0BCLR  FTC0OUT
```

或

```
B0BCLR   FALOAD0          ; ALOAD0，TC0OUT = 01，PWM周期= 0~63。
B0BSET   FTC0OUT
```

或

```
B0BSET   FALOAD0          ; ALOAD0，TC0OUT = 10，PWM周期= 0~31。
B0BCLR   FTC0OUT
```

或

```
B0BSET   FALOAD0          ; ALOAD0，TC0OUT = 11，PWM周期= 0~15。
B0BSET   FTC0OUT
```

例：设置TC0模式。

```
B0BSET   FTC0IEN          ; 使能TC0中断功能。
```

或

```
B0BSET   FTC0OUT          ; 使能TC0OUT（蜂鸣器）功能。
```

或

```
B0BSET   FPWM0OUT         ; 使能PWM功能。
```

例：开启TC0。

```
B0BSET   FTC0ENB
```

6.1.4 、PWM0

6.1.4.1 、概述

PWM信号输出到PWM0OUT（P5.4 引脚），TC0OUT和ALOAD0标志位控制PWM输出的阶数（256、64、32和16）。8位计数器TC0C计数过程中不断与TC0R相比较，当TC0C计数到两者相等时，PWM 输出低电平，当TC0C再次从零开始计数时，PWM被强制输出高电平。PWM0输出占空比= TC0R/计数量程（计数量程= 256、64、32或16）。

参考寄存器保持输入00H可使PWM的输出长时间维持在低电平，通过修改TC0R可改变PWM输出占空比。

注：TC0为双重缓存器结构，调整TC0R的值可以改变PWM的输出占空比。用户可随时改变TC0R的值，但是只有在TC0溢出后，这一修改值才真正被写入TC0R中。

表 6-12 PWM0 输出占空比

ALOD0	TC0OUT	PWM 占空比范围	TC0C 有效值	TC0R 有效值	MAX.PWM 输出频率（Fcpu=4MHz）	注释
0	0	0/256~255/256	00H~0FFH	00H~0FFH	7.8125K	每计数 256 次溢出
0	1	0/64~63/64	00H~3FH	00H~3FH	31.25K	每计数 256 次溢出
1	0	0/32~31/32	00H~1FH	00H~1FH	32.5K	每计数 256 次溢出

0	1	0/16~15/16	00H~0FH	00H~0FH	125K	每计数256 次溢出

PWM输出占空比随TC0R的变化而变化：0/256~255/256。

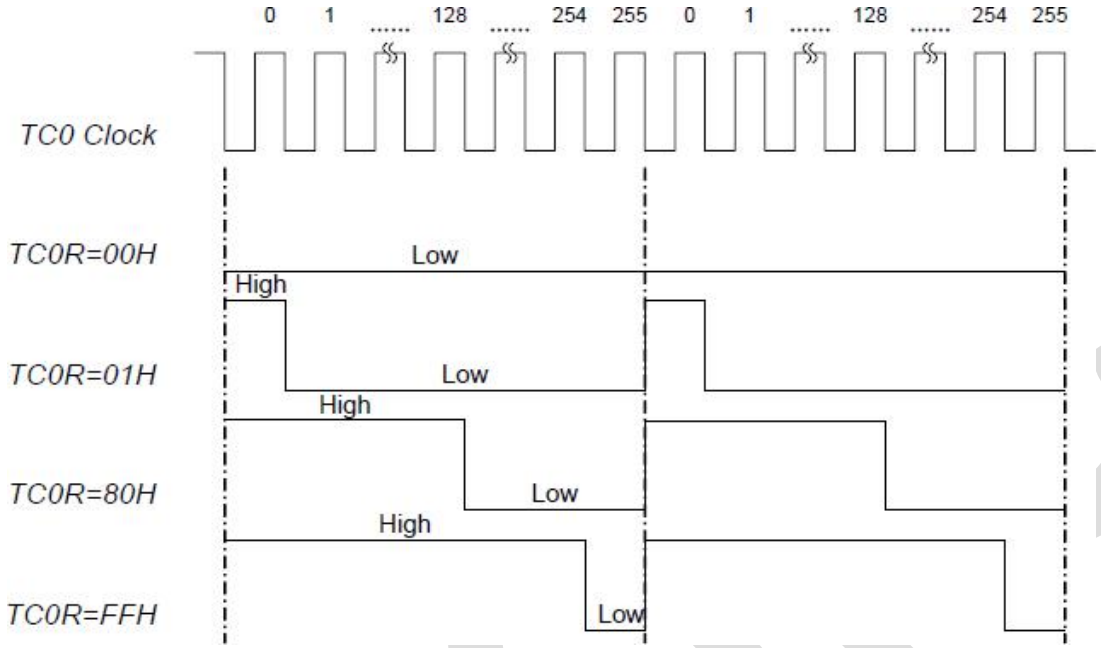


图 6-4 PWM 与 TC0R 关系

6.1.4.2 、TC0IRQ 和 PWM 输出占空比

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

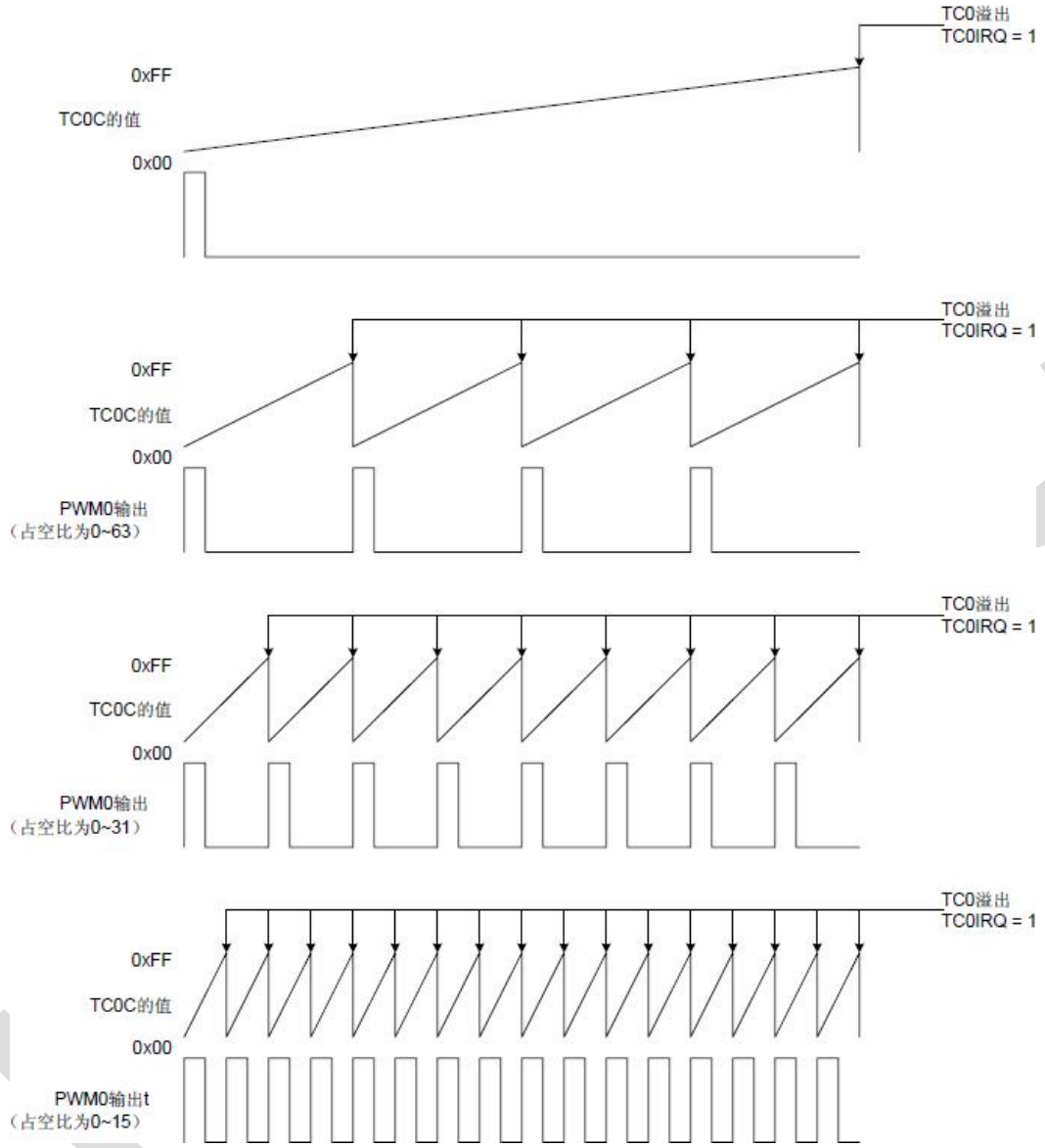


图 6-5 PWM 输出占空比

6.1.4.3 、PWM 输出占空比与 TC0R 的变化

在 PWM 模式下，系统随时比较 TC0C 和 TC0R 的异同。若 TC0C<TC0R，PWM 输出高电平，反之则输出低电平。当 TC0C 发生改变的时候，PWM 将在下一周期改变输出占空比。如果 TC0R 保持恒定，那么 PWM 输出波形也保持稳定。

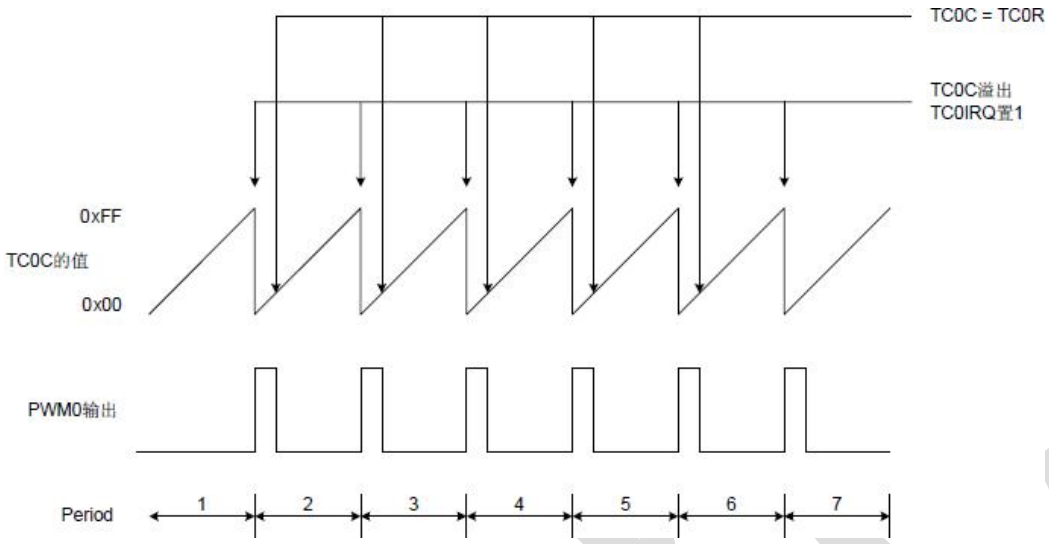


图 6-6 PWM 占空比与 TC0C 关系（TC0R 恒定）

上图所示是TC0R恒定时的波形。每当TC0C溢出时，PWM都输出高电平，TC0C≥TC0R 时，PWM 即输出低电平。下面所示是TC0R发生变化时对应的波形图：

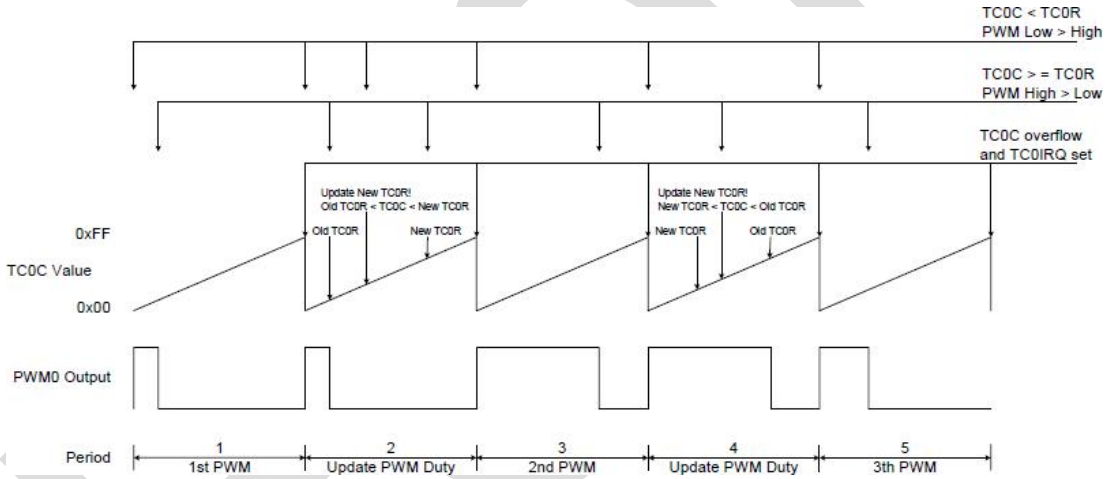


图 6-7 PWM 占空比与 TC0C 关系（TC0R 变化）

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

6.1.4.4 、PWM 编程举例

例：PWM输出设置。外部高速振荡器输出频率= 4MHZ，Fcpu = Fosc/4，PWM输出占空比 = 30/256，输出频率1KHZ，PWM时钟源来自外部时钟，TC0速率 = Fcpu/4，TC0RATE2~TC0RATE1 = 110，TC0C = TC0R = 30。

```
MOV         A,#01100000B
```

```
B0MOV     TC0M,A          ; TC0速率＝Fcpu/4。
MOV       A,#30           ; PWM输出占空比＝30/256。
B0MOV     TC0C,A
B0MOV     TC0R,A
B0BCLR    FTC0OUT         ;占空比变化范围为0/256~255/256。
B0BCLR    FALOAD0
B0BSET    FPWM0OUT        ; PWM0输出至P5.4，禁止P5.4 I/O功能。
B0BSET    FTC0ENB
```

注：TC0R为只写寄存器，不能用INCMS和DECMS指令对其进行操作。

例：改变TC0R的内容。

```
MOV       A, #30H
B0MOV     TC0R, A
INCMS     BUF0
NOP
B0MOV     A, BUF0
B0MOV     TC0R, A
```

6.2、5+1 通道 ADC

6.2.1 、概述

AIP8P102A的模数转换（A/D）模块可以提供5个模拟输入通道，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。进行AD转换时，首先要选择输入通道AIN0~AIN4），然后将GCHS和ADS置为“1”，并启动AD转换。当AD转换结束后，系统会自动的把EOC置为“1”，并将转换结果存入寄存器ADB中。

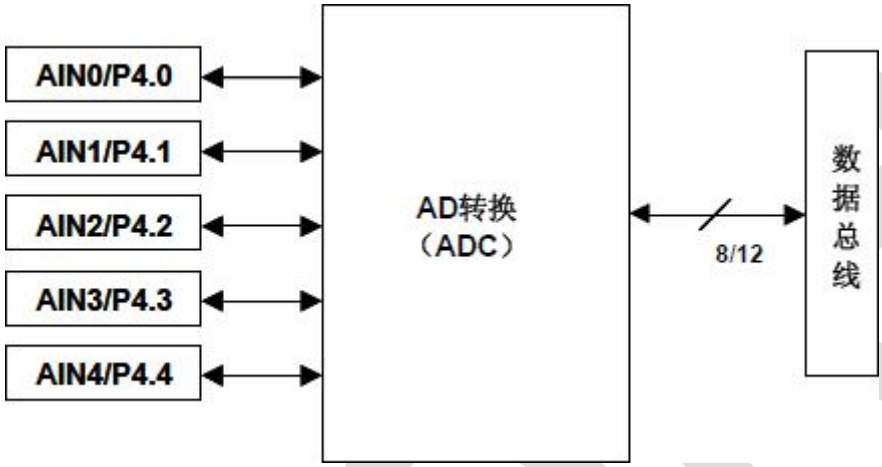


图 6-8 ADC

注：1、分辨率为12位时，转换时间需16个输入时钟。

2、模拟输入电压必须在VDD和VSS之间。

ADC设计时应注意：

1. ADC 的输入/输出引脚设置为输入模式。
2. 禁止 ADC 输入引脚的上拉电阻。
3. 在进入睡眠模式前禁止 ADC（ADENB = 0）以省电。
4. 在省电模式下设置 P4CON 的有关位以避免额外功耗。
5. 允许 ADC（ADENB = 1）后延迟 100us 等待 ADC 电路稳定。

6.2.2 、ADM 寄存器

表 6-13 ADM 寄存器（0B1H）

Bit	7	6	5	4	3	2	1	0
Name	ADENB	ADS	EOC	GCHS	-	CHS2	CHS1	CHS0
R/W	R/W	R/W	R/W	R/W	-	R/W	R/W	R/W
POR	0	0	0	0	-	0	0	0

位	字段	描述
7	ADENB	ADC 控制位。睡眠模式下，禁止 ADC 以省电。 0 = 禁止； 1 = 使能。
6	ADS	ADC 启动位。ADC 处理完成后，ADS 位自动清零。 0 = 停止； 1 = 开始。
5	EOC	ADC 状态控制位。

		0 = 转换过程中； 1 = 转换结束，ADS 复位。
4	GCHS	通道选择位。 0 = 禁止 AIN 通道； 1 = 使能 AIN 通道。
2:0	CHS[2:0]	CHS[2:0]：ADC 输入通道选择位。 000 = AIN0；001 = AIN1；010 = AIN2； 011 = AIN3；100 = AIN4。

注：若ADENB = 1，用户应设置P4.n/AINn为输入模式，且禁止上拉电阻，系统不会自动设置。若已经设置了P4CON.n，P4.n/AINn的数字功能（包括上拉电阻）被隔离。

6.2.3 、ADR 寄存器

表 6-14 ADR 寄存器（0B3H）

Bit	7	6	5	4	3	2	1	0
Name	-	ADCKS1	-	ADCKS0	ADB3	ADB2	ADB1	ADB0
R/W	-	R/W	-	R/W	R	R	R	R
POR	-	0	-	0	X	X	X	X

位	字段	描述
6,4	ADCKS1，ADCKS0	ADC 时钟源选择位。

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

6.2.4 、ADB 数据缓存器

ADC 数据缓存器共 12 位，用来存储 AD 转换结果，8 位只读寄存器 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[3:0]：12 位 ADC 模式下，ADC 数据存放于 ADB 和 ADR 寄存器中。

表 6-15 ADB 数据缓存器（0B2H）

Bit	7	6	5	4	3	2	1	0
ADB	ADB15	ADB14	ADB13	ADB12	ADB11	ADB10	ADB9	ADB8
读/写	R	R	R	R	R	R	R	R
复位后	X	X	X	X	X	X	X	X

位	字段	描述
7:0	ADB[7:0]	ADC 12 位分辨率的高字节数据缓存器。

表 6-16 ADR 寄存器（0B3H）

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

POR	-	0	-	0	X	X	X	X

位	字段	描述
3:0	ADR[3:0]	ADC 12 位分辨率的低字节数据缓存器。

表 6-17 AIN 的输入电压 v.s. ADB 的输出数据

AINn	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：

表 6-18 12 位分辨率模式选择表

ADC	ADB								ADR			
分辨率	ADB11	ADB10	ADB9	ADB8	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0
8-bit	O	O	O	O	O	O	O	O	x	x	x	x
9-bit	O	O	O	O	O	O	O	O	O	x	x	x
10-bit	O	O	O	O	O	O	O	O	O	O	x	x
11-bit	O	O	O	O	O	O	O	O	O	O	O	x
12-bit	O	O	O	O	O	O	O	O	O	O	O	O
O = 可选位，x = 未使用的位												

注：ADC缓存器ADB在复位后的初始值是未知的。

6.2.5 、P4CON 寄存器

P4口和ADC的输入口共享。同一时间只能设置P4口的一个引脚作为ADC的测量信号输入口（通过ADM寄存器来设置），其它引脚则作为普通I/O使用。具体应用中，当输入一个模拟信号到CMOS结构端口，尤其当模拟信号为1/2VDD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。

表 6-19 P4CON 寄存器（0AEH）

Bit	7	6	5	4	3	2	1	0
Name	-	-	-	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
R/W	-	-	-	W	W	W	W	W
POR	-	-	-	0	0	0	0	0

位	字段	描述
4:0	P4CON[4:0]	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。

6.2.6 、AD 转换时间

12位AD转换时间 = 1/（ADC clock /4）*16 sec

Fcpu = 4MHz（高速时钟，Fosc = 16MHz，Fcpu = Fosc/4）

表 6-20 AD 转换时间

ADLEN	ADCKS1	ADCKS0	ADC Clock	ADC 转换时间
1(12-bit)	0	0	Fcpu/16	1/(4MHz/16/4)*16=256us
	0	1	Fcpu/8	1/(4MHz/8/4)*16=128us
	1	0	Fcpu/1	1/(4MHz/4)*16=16us
	1	1	Fcpu/2	1/(4MHz/2/4)*16=32us

6.2.7 、ADC 操作实例

例：设置AIN0为12位ADC输入并在进入省电模式后关闭AD转换。

```
ADC0:
        B0BSET      FADENB          ;使能 ADC 电路。
        CALL        Delay100uS      ;延迟 100us 等待 ADC 电路开始转换。
        MOV         A, #0FEh
        B0MOV       P4UR, A         ;禁止 P4.0 上拉电阻。
        B0BCLR      FP40M           ;设置 P4.0 为输入模式。
        MOV         A, #01h
        B0MOV       P4CON, A        ;设置 P4.0 为模拟输入模式。
        MOV         A, #60H
        B0MOV       ADR, A          ;设置 12 位 ADC，ADC 时钟源= Fosc。
        MOV         A,#90H
        B0MOV       ADM,A           ;允许 ADC 并设置 AIN0 输入。
        B0BSET      FADS            ;开始转换。
WADC0:
        B0BTS1      FEOC
        JMP         WADC0
        B0MOV       A,ADB
        B0MOV       Adc_Buf_Hi, A
        B0MOV       A,ADR
        AND         A, 0Fh
        B0MOV       Adc_Buf_Low, A
Power_Down:
        B0BCLR      FADENB          ;关闭 AD 转换。
        B0BCLR      FCPUM1
        B0BSET      FCPUM0          ;进入睡眠模式。
```

6.2.8 、ADC 应用电路

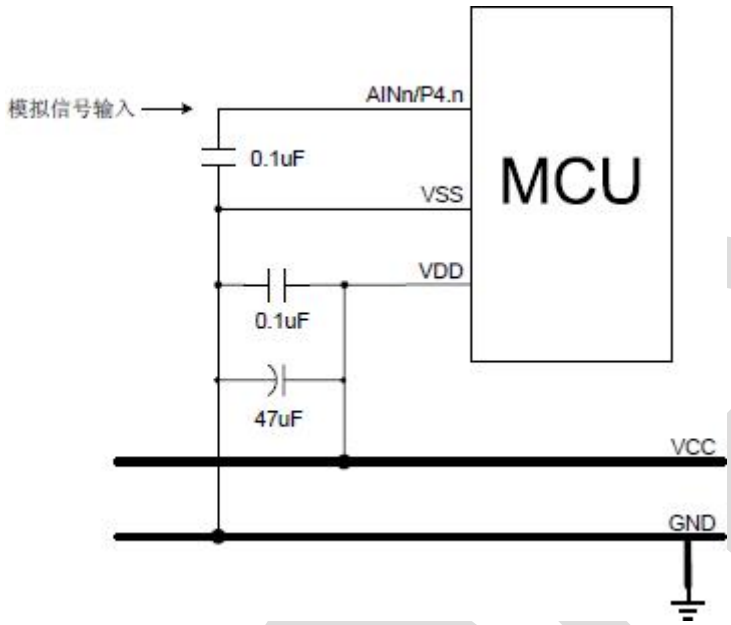


图 6-9 ADC 应用电路

在 ADC 输入引脚接入 0.1uF 的电容可以滤除电源端的杂汛。

6.3、2K/4K 蜂鸣器（Buzzer）输出

6.3.1 、概述

AIP8P102A 内置蜂鸣器产生模块，输出频率为 2KHz 或 4KHz。通过 BZM 寄存器可以调整蜂鸣器的输出频率。蜂鸣器输出引脚与普通 I/O 引脚共用。当 BZEN=1 时，引脚输出蜂鸣器信号。当 BZEN=0 时，引脚返回上一个 I/O 状态（输入模式，输出高或输出低）。

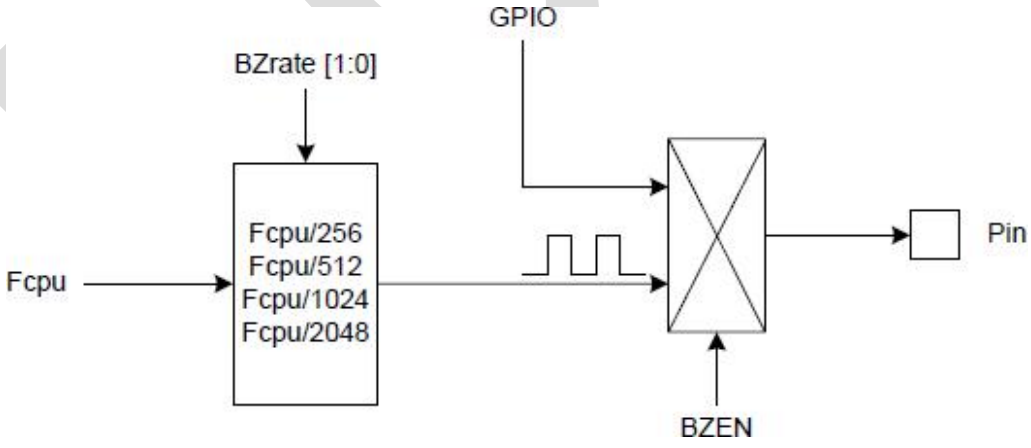


图 6-10 蜂鸣器

蜂鸣器输出频率可由Fcpu（指令周期）分频获得，用户可通过蜂鸣器分频选择位（BZrate）设定。Fcpu决定蜂鸣器的频率，频率选择列表如下：

表 6-21 频率选择列表

BZrate[1:0]	蜂鸣器除频数	蜂鸣器分频		
		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

为了获得蜂鸣器输出 2KHz 和 4KHz 的频率，需要选择合适的 Fcpu 分频数，可参考上表中所列 2KHz/4KHz 蜂鸣器输出。

6.3.2 、 BZM 模式寄存器

表 6-22 BZM 寄存器（0DCH）

Bit	7	6	5	4	3	2	1	0
Name	BZEN	BZrate1	BZrate0	-	-	-	-	-
R/W	R/W	R/W	R/W	-	-	-	-	-
POR	0	0	0	-	-	-	-	-

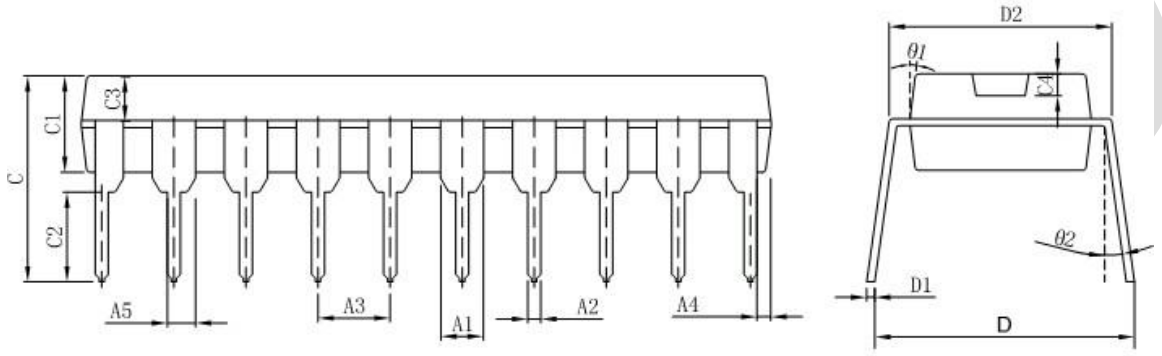
位	字段	描述
7	BZEN	蜂鸣器输出控制位。 0 =禁止蜂鸣器的输出功能，并将该引脚设置为普通的 I/O 引脚； 1 =使能蜂鸣器的输出功能，并禁止该引脚的普通 I/O 功能。
6::5	BZrate[1:0]	蜂鸣器分频选择位。 00 = Fcpu/256； 01 = Fcpu/512； 10 = Fcpu/1024； 11 = Fcpu/2048。

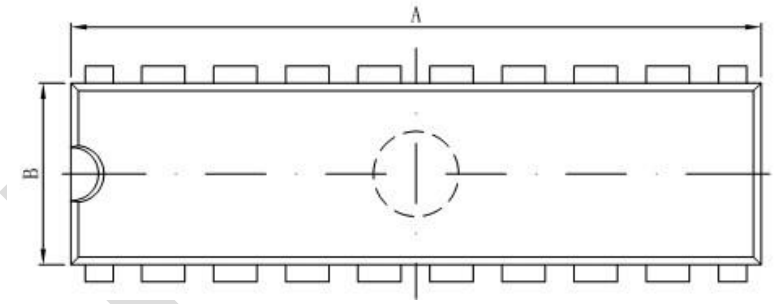
注： 1. 若 BZEN=1，P0.4 作为蜂鸣器输出引脚并关闭普通 I/O 功能。2. 若 BZEN=0，P0.4 为普通 I/O 引脚，并在禁止 Buzzer 输出功能后将该引脚转换回上一个 I/O 模式。

7、封装尺寸与外形图

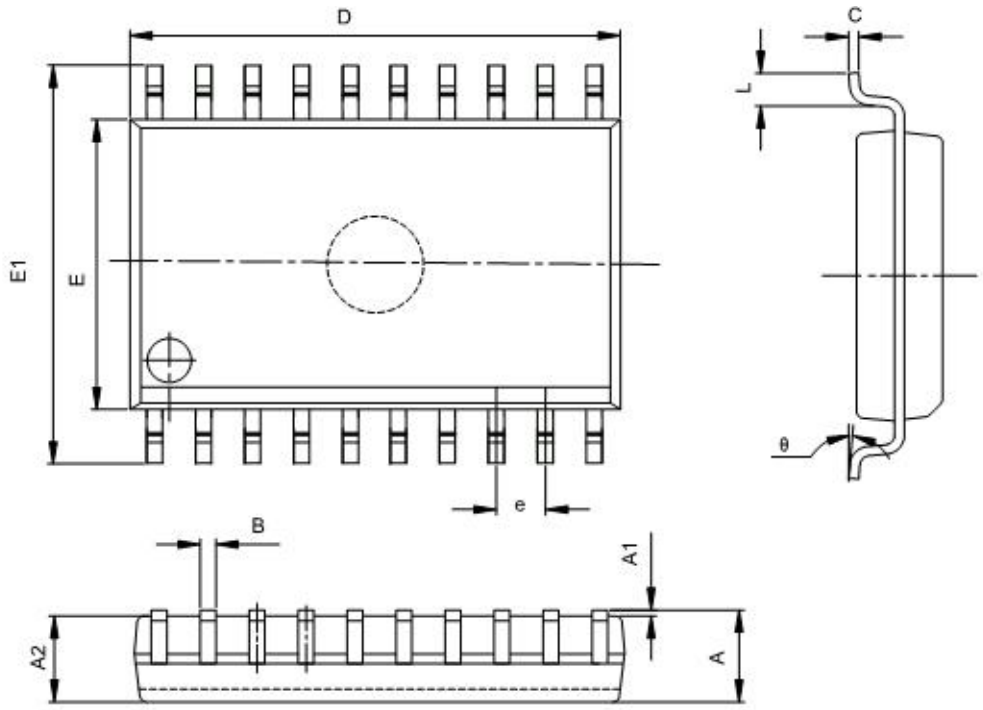
7.1、DIP20 外形图与封装尺寸

尺寸 / 标注	最小（mm）	最大（mm）	尺寸 / 标注	最小（mm）	最大（mm）
A	24.50	24.70	C2	2.9	
A1	1.40TYP		C3	1.56TYP	
A2	0.43	0.57	C4	0.80TYP	
A3	2.54TYP		D	8.20	9.70
A4	0.62TYP		D1	0.20	0.35
A5	0.95TYP		D2	7.62	7.87
B	6.3	6.5	θ1	8° TYP	
C	7.5TYP		θ2	5° TYP	
C1	3.30	3.50			



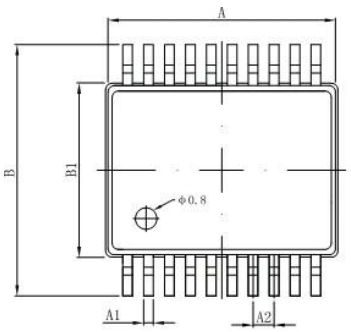


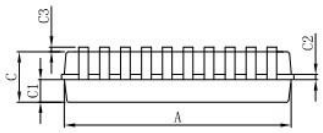
7.2、SOP20 外形图与封装尺寸

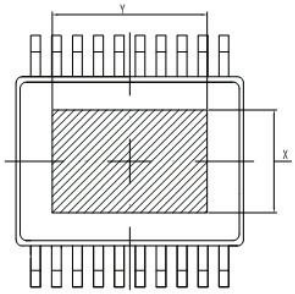


Symbol	Dimensions In Millimeters		Dimensions In Inches	
	Min	Max	Min	Max
A	2.280	2.630	0.090	0.104
A1	0.100	0.300	0.004	0.012
A2	2.180	2.330	0.086	0.092
B	0.350	0.510	0.014	0.020
C	0.204	0.360	0.008	0.014
D	12.600	13.000	0.496	0.512
E	7.400	7.600	0.291	0.299
E1	10.000	10.650	0.394	0.419
e	1.270(TYP)		0.050(TYP)	
L	0.400	1.270	0.016	0.050
θ	0°	8°	0°	8°

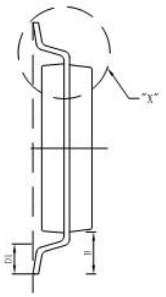
7.3、TSSOP20 外形图与封装尺寸

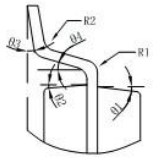






BOTTOM VIEW





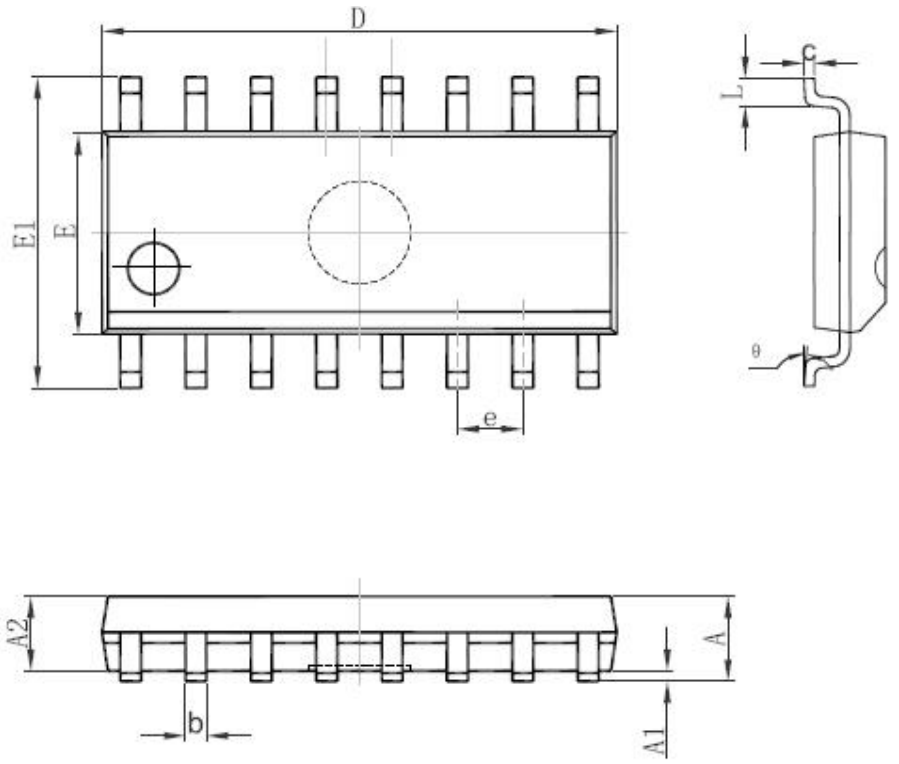
DETAIL "X"

Note:
1. Formed lead shall be planar with respect to one another within 0.004 inches,
2. Both package length and width do not include mold flash and burr.

尺寸 / 标注	最小（mm）	最大（mm）	尺寸 / 标注	最小（mm）	最大（mm）
A	6.40	6.60	C3	0.025	0.102
A1	0.20	0.30	D	1.0TYP	
A2	0.65TYP		D1	0.50	0.75
B	6.30	6.50	R1	0.15TYP	
B1	4.30	4.50	R2	0.15TYP	
C	0.90	1.05	θ1	12° TYP	
C1	0.4365TYP		θ2	12° TYP	
C2	0.09	0.2	θ3	0° TYP	8° TYP
			θ4	10° TYP	

OPTION	PAD SIZE	SYMBOL	DIMENSION	MARK
1	3(118)	X	MIN.2.60	NORMAL
			MAX.3.10	
	4.2(165)	Y	MIN.3.80	
			MAX.4.30	
2	2.11(83)	X	MIN.1.71	SPECIAL CUSTOMER
			MAX.2.21	
	3.15(124)	Y	MIN.2.75	
			MAX.3.25	

7.4、SOP16 外形图与封装尺寸



Symbol	Dimensions In Millimeters		Dimensions In Inches	
	Min	Max	Min	Max
A	1.350	1.750	0.053	0.069
A1	0.100	0.250	0.004	0.010
A2	1.350	1.550	0.053	0.061
b	0.330	0.510	0.013	0.020
c	0.170	0.250	0.007	0.010
D	9.800	10.200	0.386	0.402
E	3.800	4.000	0.150	0.157
E1	5.800	6.200	0.228	0.244
e	1.270(BSC)		0.050(BSC)	
L	0.400	1.270	0.016	0.050
θ	0°	8°	0°	8°

8、声明及注意事项

8.1、产品中有毒有害物质或元素的名称及含量

部件名称	有毒有害物质或元素									
	铅（Pb）	汞（Hg）	镉（Cd）	六阶铬（Cr（VI））	多溴联苯（PBBs）	多溴联苯醚（PBDEs）	邻苯二甲酸二丁酯（DBP）	邻苯二甲酸丁苄酯（BBP）	邻苯二甲酸二（2-乙基巳基）酯（DEHP）	邻苯二甲酸二异丁酯(DIBP)
引线框	○	○	○	○	○	○	○	○	○	○
塑封树脂	○	○	○	○	○	○	○	○	○	○
芯片	○	○	○	○	○	○	○	○	○	○
内引线	○	○	○	○	○	○	○	○	○	○
装片胶	○	○	○	○	○	○	○	○	○	○
说明	○：表示该有毒有害物质或元素的含量在 SJ/T11363-2006 标准的检出限以下。 ×：表示该有毒有害物质或元素的含量超出 SJ/T11363-2006 标准的限量要求。									

8.2、注意

在使用本产品之前建议仔细阅读本资料；

本资料中的信息如有变化，恕不另行通知；

本资料仅供参考，本公司不承担任何由此而引起的任何损失；

本公司也不承担任何在使用过程中引起的侵犯第三方专利或其它权利的责任。

无锡中微爱芯电子有限公司
国芯思辰（深圳）科技有限公司
深圳公司:深圳市福田区石厦街新天世纪商务中心A座1513室
公司网址:www.zhongke-ic.com
联系电话:0755-82565229