AiP8P102G
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
2016-07-A1	2016-07	新制
2017-10-B1	2017-10	修订
2018-01-B2	2018-01	修订

1、产品概述

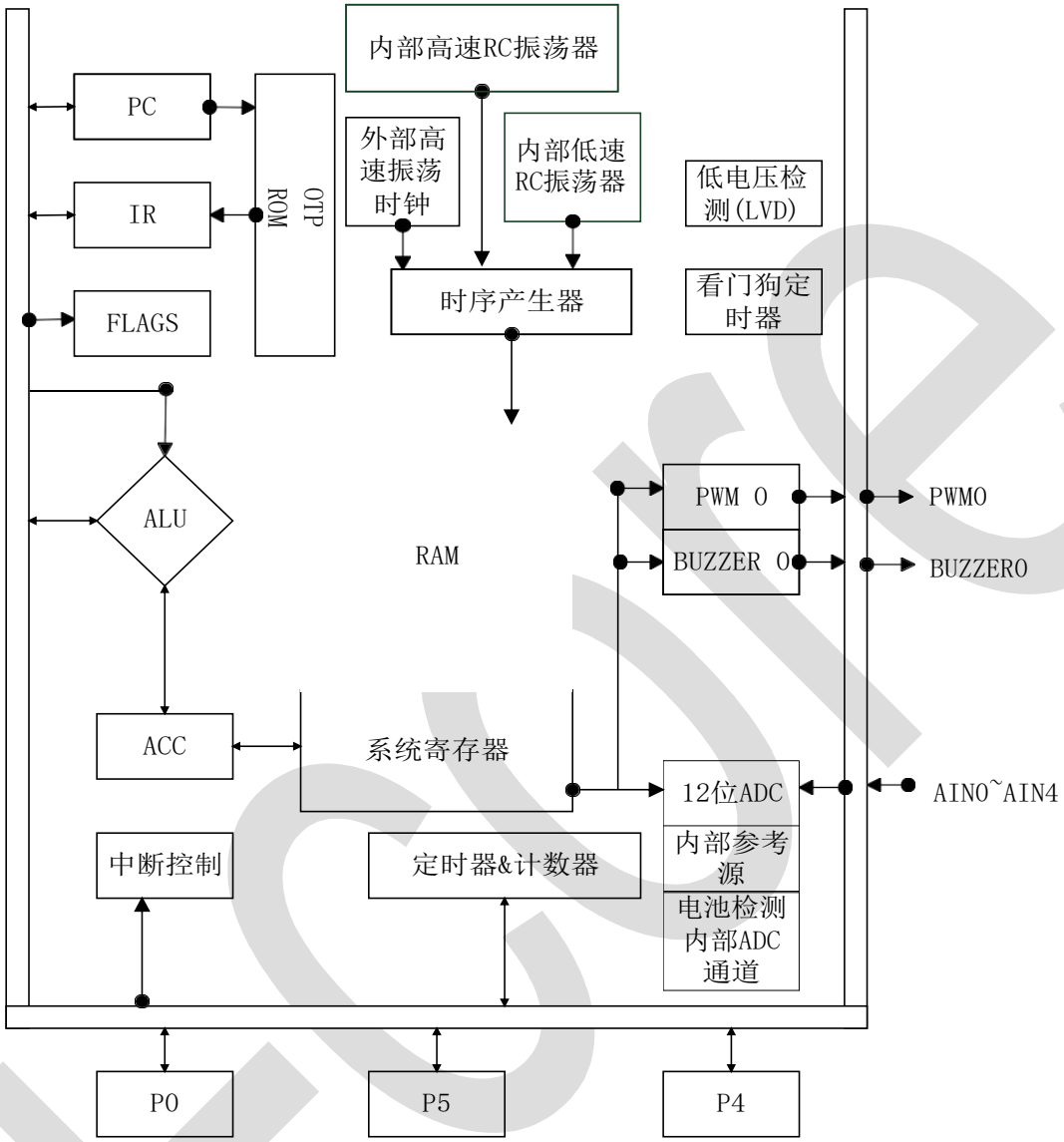
1.1、功能特性

- 存储器：
  - ROM，2K*16位
  - RAM，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端口配置
  - 双向输入输出口：P0、P4、P5
  - 具有唤醒功能端口：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
- 功能强大的指令系统：
  - 指令长度为1个字
  - 大部分指令执行只需1个周期
  - 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 位定时器
  - T0：基本定时器
  - TC0：自动装载定时/计数器/PWM/蜂鸣器输出
- 单通道8位PWM输出
- 单通道2KHz/4KHz蜂鸣器输出
- 内置开门狗定时器，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10MHz
  - 外部高速时钟：晶振模式，高达16 MHz
  - 内部高速时钟：RC模式，高达16MHz
  - 内部高速时钟：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、低时钟均不工作
  - 绿色模式：由定时器周期性唤醒
- 封装形式
  - DIP 20 pins
  - SOP 20 pins，SOP 16 pins
  - TSSOP20

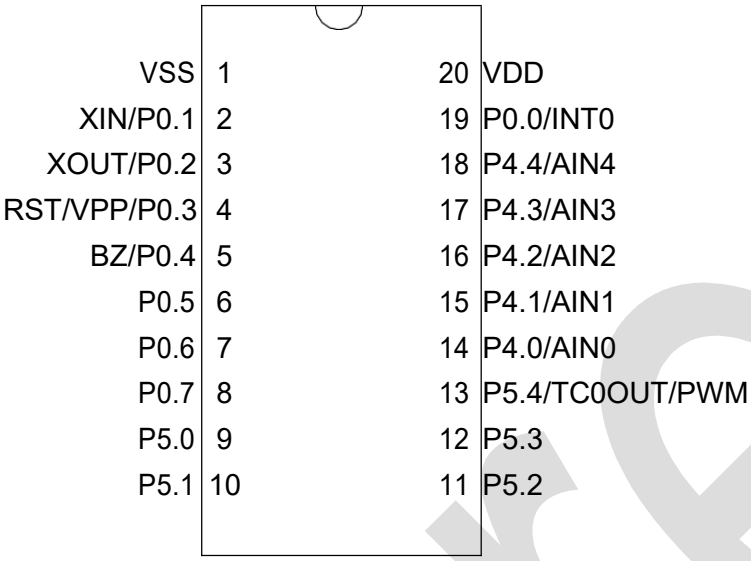
型号	全称	封装	备注
AiP8P102G	AiP8P102GGO	SOP20	
	AiP8P102GGP	DIP20	
	AiP8P102GGI	TSSOP20	
	AiP8P102GEO	SOP16	

2、功能框图及引脚说明

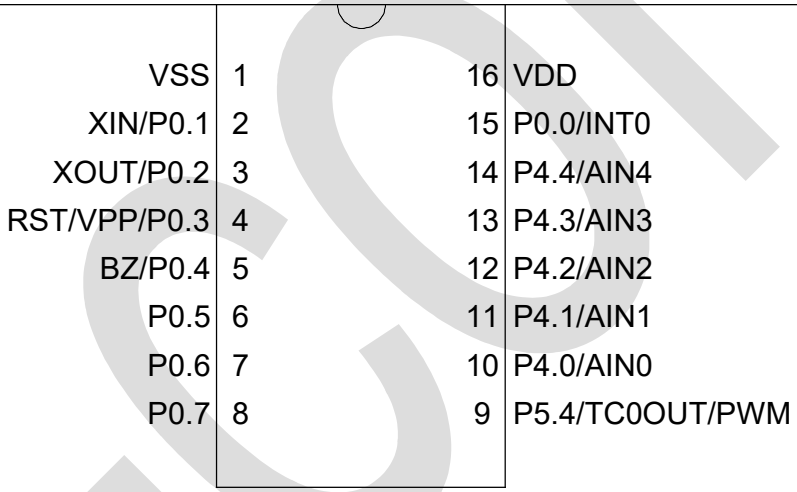
2.1、功能框图



2.2、引脚排列图



SOP20/DIP20/TSSOP20



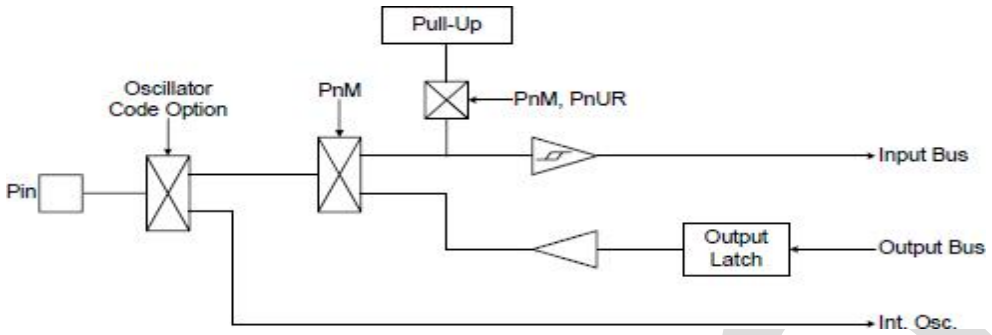
SOP16

2.3、引脚说明及结构原理图

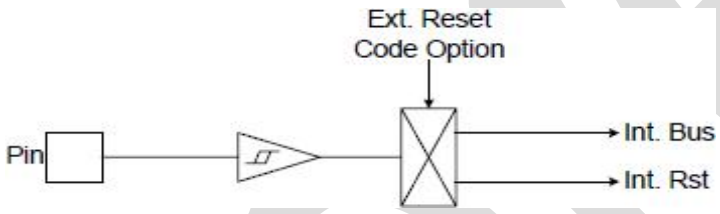
引脚（20 PIN）	引脚 （ 16 PIN）	符 号	功 能
1	1	VSS	电源输入端
2	2	XIN/P0.1	XIN：使能外部振荡器（晶振或 RC）时为振荡信号输入引脚。 P0.1：双向输入/输出引脚，输入模式时为施密特触发，内置上拉电阻，具有唤醒功能。
3	3	XOUT/P0.2	XOUT：使能外部晶振模式时为振荡信号输出引脚。 P0.2：双向输入/输出引脚，输入模式时为施密特触发，内置上拉电阻，具有唤醒功能。
4	4	P0.3/RST/VPP	P0.3：禁止外部复位时为单向输入引脚，施密特触发，

			无内置上拉电阻。 RST：系统复位输入引脚，施密特触发，低电平有效，通常保持高电平。 VPP：烧录时为 12.3V 电源输入引脚。
5	5	P0.4/BZ	P0.4：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。 BZ：2KHz/4KHz buzzer 输出引脚。
6	6	P0.5	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。
7	7	P0.6	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。
8	8	P0.7	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。
9		P5.0	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。
10		P5.1	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。
11		P5.2	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。
12		P5.3	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。
13	9	P5.4/PWM/TC0OUT	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。
14	10	P4.0/AIN0	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。 AIN0：ADC 的输入通道。
15	11	P4.1/AIN1	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。 AIN1：ADC 的输入通道。
16	12	P4.2/AIN2	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。 AIN2：ADC 的输入通道。
17	13	P4.3/AIN3	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。 AIN3：ADC 的输入通道。
18	14	P4.4/AIN4	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。 AIN4：ADC 的输入通道。
19	15	P0.0/INT0	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。
20	16	VDD	电源输入端。

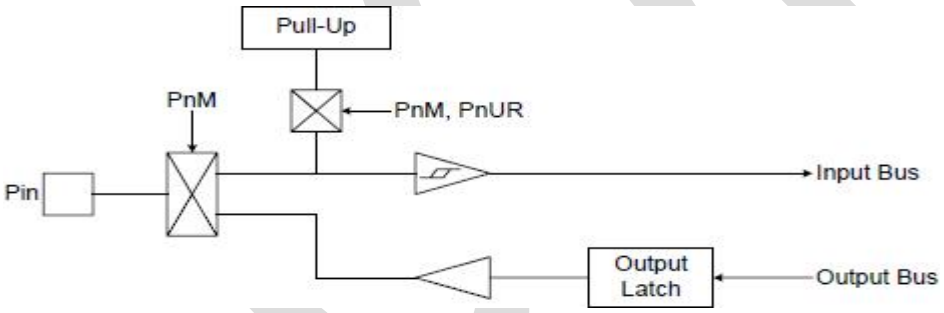
P0.1、P0.2 结构：



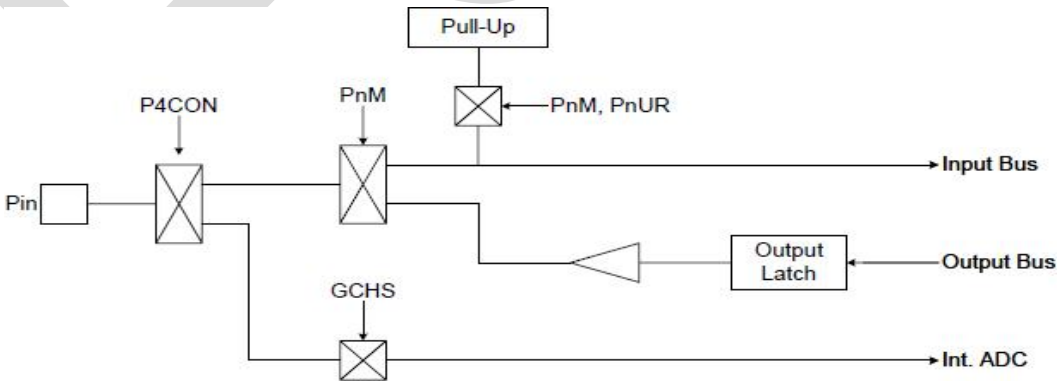
P0.3 结构：



P0、P5 结构：



P4 结构：



3、电特性

3.1、极限参数(除非另有规定，T_{amb}=25℃)

	符 号	额 定 值	单 位
工作电压	V_{CC}	6	V
输入电压	VIN	6.2	V
工作温度	Topr	70	°C
储存温度	Tstor	125	°C

3.2、电气特性（ 除非另有规定，T_{amb}=25℃,V_{DD}=5V， fosc = 4MHz,fcpu=1MHZ）

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	5.0	5.5	V
		普通模式，Vpp=Vdd，-40℃~85℃		2.5	5.0	5.5	V
RAM 数据保留电压	Vdr			1.5			V
Vdd 上升率	Vpor	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
	ViL3	P4 和 ADC 共享引脚		Vss		0.5Vdd	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
	ViH3	P4 和 ADC 共享引脚		0.5Vdd		Vdd	V
复位漏电流	Ilekg	Vin=Vdd				2	uA
I/O 上拉电阻	Rup	Vin=Vss, Vdd=3V		100	200	300	KΩ
		Vin=Vss, Vdd=5V		50	100	150	KΩ
I/O 口输入漏电流	Ilekg	上拉电阻失去作用，Vin=Vdd				2	uA
I/O 口输出源电流 反向电流	IoH	Vop = Vdd – 0.5V		8	12		mA
	IoL	Vop = Vdd +0.5V		8	15		mA
INTn 触发脉冲宽度	Tint0	INT0 中断请求的脉冲宽度		2/fcpu			cycle
电流 (不可见 ADC）	Idd1	运行模式(没有加载，Fcpu=Fosc/4)	Vdd=5V,4MHz		2.5	5	mA
			Vdd=3V,4MHz		1	2	mA
	Idd2	缓慢模式(内部低RC,在高电平时钟停止)	Vdd=5V,2KHz		20	40	uA
			Vdd=3V,16KHz		5	10	uA
	Idd3	睡眠模式	Vdd=5V,25℃		0.8	1.6	uA
			Vdd=3V,25℃		0.7	1.4	uA
			Vdd=5V, -40°C~ 85°C		10	21	uA
			Vdd=5V, -40°C~ 85°C		10	21	uA
	Idd4	绿色模式(没有加载,Fcpu=Fosc/4，看门狗关闭)	Vdd=5V,4Mhz		0.6	1.2	mA
			Vdd=3V,4Mhz		0.25	0.5	mA
			Vdd=5V,ILRC32Khz		15	30	uA
			Vdd=3V,ILRC16Khz		3	6	uA
内部高电平振	Fihrc	内部高电平	25°C, Vdd=5V,Fcpu =1MHz	15.68	16	16.32	MHz

荡频率			-40°C~85°C 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 = 1 MHz。低电压指示器电平, Fcpu = 1 MHz		2.0	2.3	3	V
	Vdet2	低电压指示器电平, Fcpu = 1 MHz		2.7	3.3	4.5	V
AIN0 ~ AIN5 输入电压	Vani	Vdd = 5.0V		0		Vrefh1 ~5	V
ADC 使能时间	Tast	ADENB=1，准备开始转换		100			us
ADC 消耗电流	IADC	Vdd=5.0V			0.6		mA
		Vdd=3.0V			0.4		mA
ADC 时钟频率	FAD CLK	Vdd=5.0V		32K		8M	Hz
		Vdd=3.0V		32K		8M	Hz
ADC 转换周期时间	FAD CYL	VDD=2.4V~5.5V		64			1/FAD CLK
ADC 取样速率 (设置 DS=1 Frequency)	FADS MP	VDD=5.0V				125	K/sec
		VDD=3.0V				80	K/sec
微分非线性	DNL	VDD=5.0V , AVREFH=3.2V, FADSMP =7.8K		±1	±2	±16	LSB
积分非线性	INL	VDD=5.0V , AVREFH=3.2V, FADSMP =7.8K		±2	±4	±16	LSB
无丢码丢失	NMC	VDD=5.0V , AVREFH=3.2V, FADSMP =7.8K		8	10	12	Bits

注：这些参数为参考参数，非实测值。

4、CPU 说明

4. 1、概述

AIP8P102G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）。

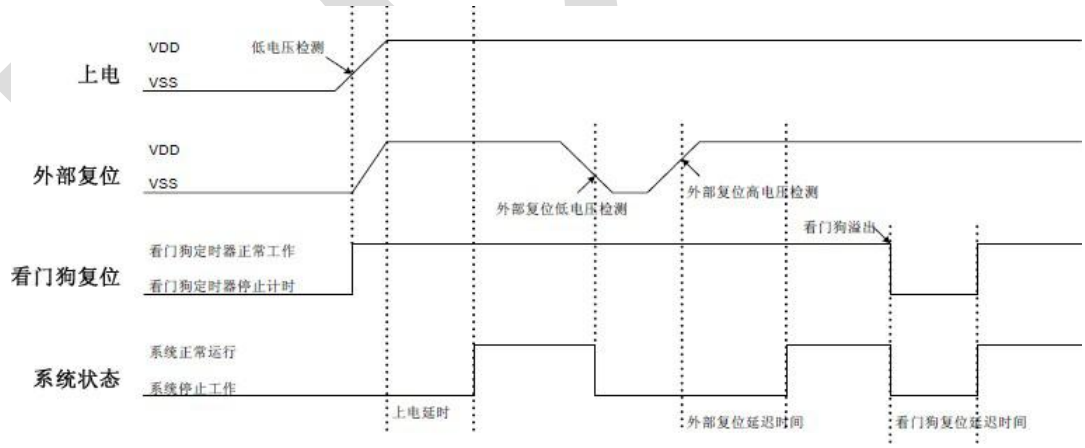
上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器PC清零。复位结束后，系统从向量0000H处重新开始运行。PFLAG寄存器的NT0和NPD两个标志位能够给出系统复位状态的信息。用户可以编程控制NT0和NPD，从而控制系统的运行路径。

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	X	X	0	0	-	0	0	0

Bit [7:6] NT0, NPD：复位状态标志。

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	电源电压处于 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。



4. 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。
-

4. 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：

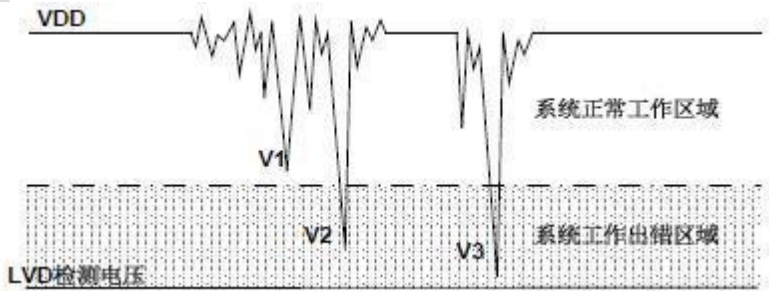
- 看门狗定时器状态：系统检测看门狗定时器是否溢出，若溢出，则系统复位；
- 系统初始化：所有的系统寄存器被置为默认状态；
- 振荡器开始工作：振荡器开始提供系统时钟；
- 执行程序：上电结束，程序开始运行。
- 看门狗定时器应用注意事项：
- 对看门狗清零之前，检查I/O口的状态和RAM的内容可增强程序的可靠性；
- 不能在中断中对看门狗清零，否则无法侦测到主程序跑飞的状况；
- 程序中应该只在主程序中有一次清看门狗的动作，这种架构能够最大限度的发挥看门狗的保护功能。

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。

4. 4、掉电复位

4.4.1 、概述

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



掉电复位示意图

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。

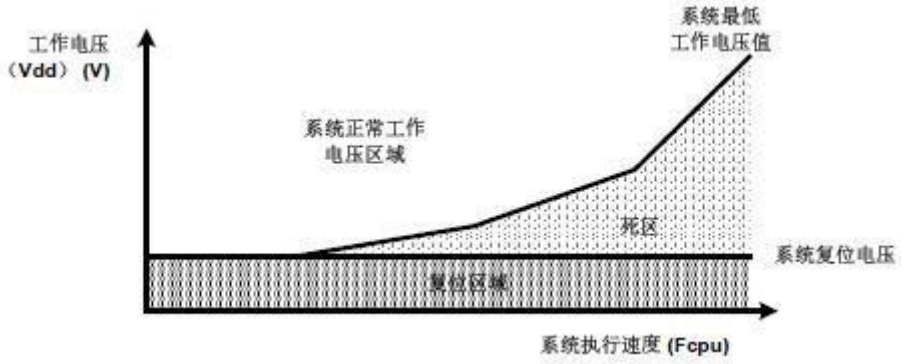
AC运用中：

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。

4.4.2 、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。



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。

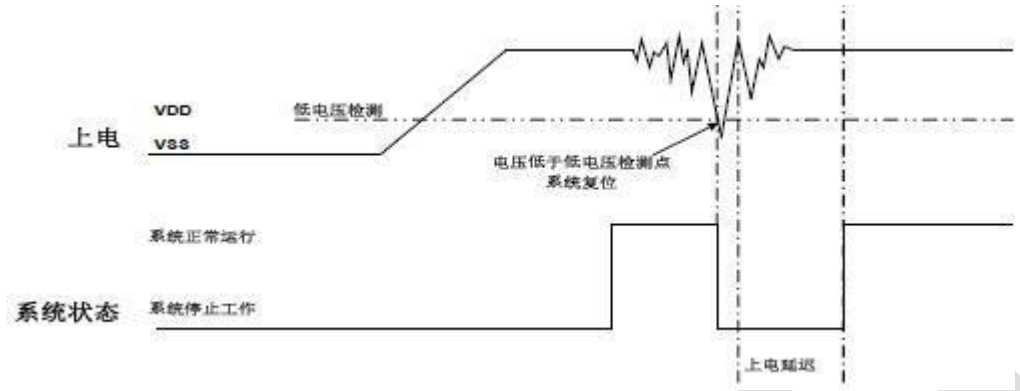
4.4.3 、掉电复位性能改进

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：



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	X	X	0	0	-	0	0	0

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

0＝系统工作电压VDD超过2.4V，低电压检测器没有工作；

1＝系统工作电压VDD低于2.4V，说明此时低电压检测器已处于监控状态。

LVD	LVD 编译选项		
	LVD_L	LVD_M	LVD_H
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；

LVD24标志位无意义。

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。

- 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。

4. 5、外部复位

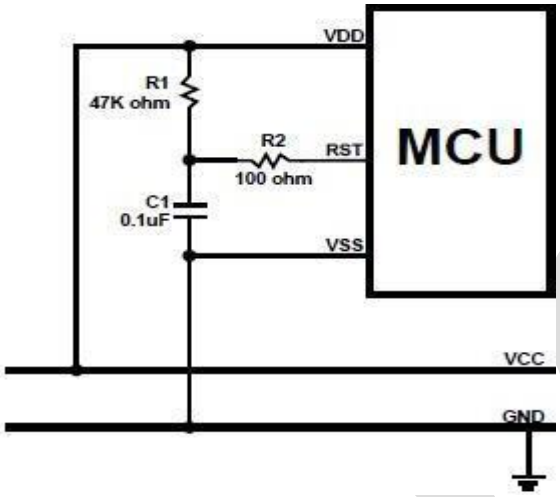
外部复位功能由编译选项“Reset_Pin”控制。将该编译选项置为“Reset”，可使能外部复位功能。外部复位引脚为施密特触发结构，低电平有效。复位引脚处于高电平时，系统正常运行。当复位引脚输入低电平信号时，系统复位。外部复位操作在上电和正常工作模式时有效。需要注意的是，在系统上电完成后，外部复位引脚必须输入高电平，否则系统将一直保持在复位状态。外部复位的时序如下：

- 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。

4. 6、外部复位电路

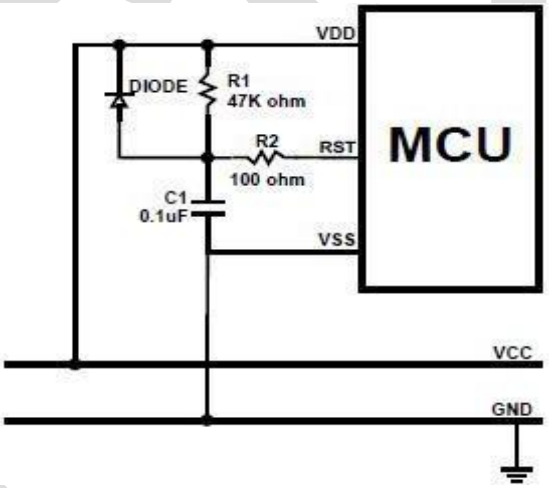
4.6.1、RC 复位电路



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

注：此 RC 复位电路不能解决非正常上电和掉电复位问题。

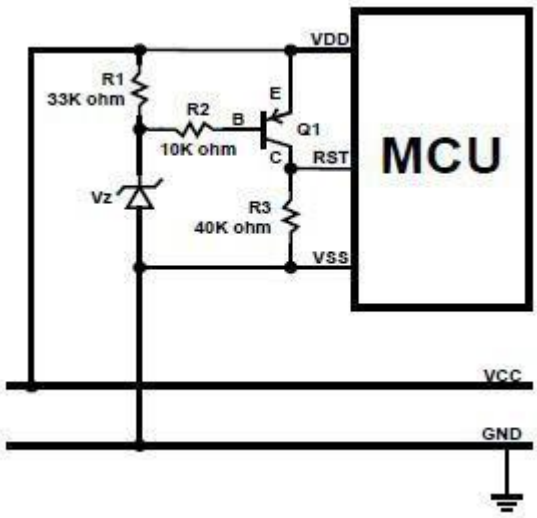
4.6.2 、二极管及 RC 复位电路



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

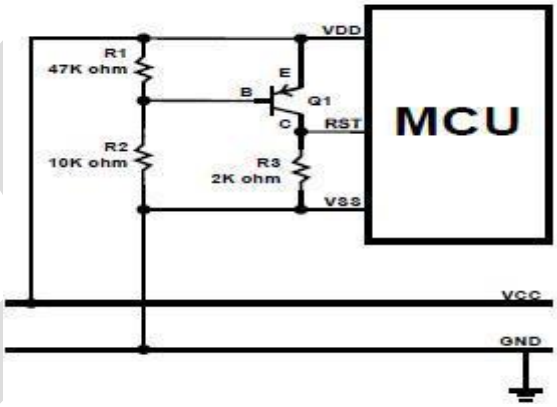
注：“基本RC复位电路”和“二极管及RC复位电路”中的电阻R2都是必不可少的限流电阻，以避免复位引脚ESD（Electrostatic Discharge）或EOS（Electrical Over-stress）击穿。

4.6.3 、稳压二极管复位电路



稳压二极管复位电路是一种简单的LVD电路，基本上可以完全解决掉电复位问题。如上图电路中，利用稳压管的击穿电压作为电路复位检测值，当VDD高于“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。

4.6.4 、电压偏置复位电路

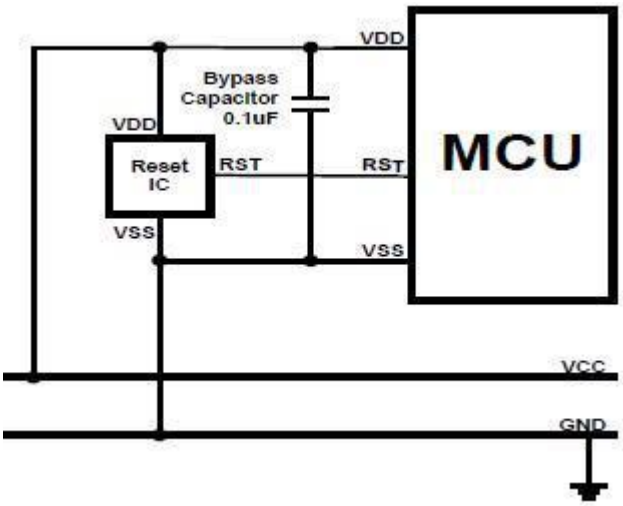


电压偏置复位电路是一种简单的LVD电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，这种复位电路的检测电压值的精确度有所降低。电路中，R1和R2构成分压电路，当VDD 高于和等于分压值“0.7V x (R1 + R2)/ R1”时，三极管集电极C输出高电平，单片机正常工作；VDD低于“0.7V x (R1 + R2) / R1”时，集电极C输出低电平，单片机复位。

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。

4.6.5 、外部IC复位



外部复位也可以选用IC进行外部复位，但是这样一来系统成本将会增加。针对不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、中央处理器（CPU）

5. 1、存储器

5.1.1 、程序存储器

ROM：2K

错误!未找到引用源。

5.1.1.1 、复位向量（0000H）

具有一个字长的系统复位向量（0000H）。

1. 上电复位（NT0=1，NPD=0）；
2. 看门狗复位（NT0=0，NPD=0）；
3. 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。
```

5.1.1.2 、中断向量（0008H）

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

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

例：定义中断向量，中断服务程序紧随ORG 8H 之后。

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

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

例：定义中断向量，中断程序在用户程序之后。

```
.CODE

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

5.1.1.3 、查表

在单片机中，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
                …
```

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
                …
```

5.1.1.4 、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。
```

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：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：如果跳转表跨越ROM边界，将引起程序错误。

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

注：“VAL”为跳转表列表中列表个数。

例：“@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。
```

5.1.2 、CHECKSUM 计算

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

例：示例程序演示了如何对00H到用户程序结束进行Checksum计算。

```
    MOV     A,#END_USER_CODE$L
    B0MOV   END_ADDR1, A    ;用户程序结束地址低位地址存入end_addr1
    MOV     A,#END_USER_CODE$M
    B0MOV   END_ADDR2, A    ;用户程序结束地址中间地址存入end_addr2。
    CLR Y                   ; 清Y。
    CLR Z                   ; 清Z。
@@:
    MOVC
    B0BSET  FC              ; 清标志位C。
    ADD     DATA1, A        ;
    MOV     A, R
    ADC     DATA2, A        ;
    JMP     END_CHECK       ; 检查YZ地址是否为代码的结束地址。
AAA:
    INCMS Z
    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。
```

5.1.3 、编译选项

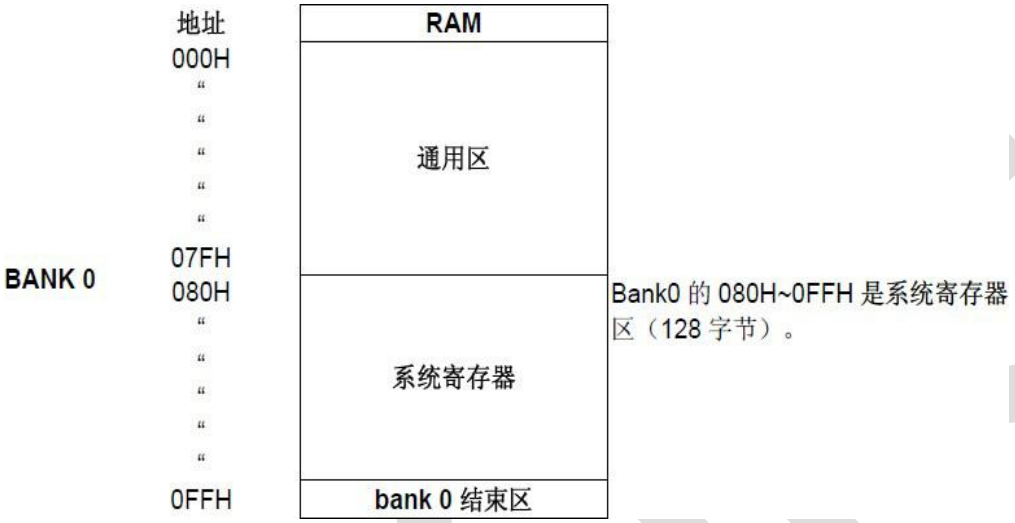
编译选项	内容	功能说明
High_Clk	IHRC_16M	内部高速振荡器采用 16MHz
	RC	外部高速时钟振荡器采用廉价的 RC
	32K	X’tal
	12M	X’tal
	4M	X’tal
Watch_Dog	Always_On	始终开启看门狗定时器，即使在睡眠模式和绿色模式下也处于开启状态。
	Enable	开启看门狗定时器，但在睡眠和绿色模式时关闭。
	Disable	关闭看门狗定时器。
Fcpu	Fhosc/4	指令周期=4
	Fhosc/8	指令周期=8 个时钟周期。
	Fhosc/16	指令周期=16
Reset_Pin	Reset	使能外部复位引脚.
	P03	P0.3
Security	Enable	ROM
	Disable	ROM
Noise_Filter	Enable	开启杂讯滤波功能。
	Disable	禁止杂讯滤波功能。
LVD	LVD_L	VDD
	LVD_M	VDD低于2.0V时，LVD复位系统； 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，LVD复位系统；

		PFLAG寄存器的LVD36位作为3.6V低电压监测器。

注: 1. 在干扰严重环境下，强烈建议开启杂迅滤波器，并将“Watch_Dog”选项设置为“Always_On”；
2. 编译选项Fcpu仅针对高速时钟，在低速模式下Fcpu = FILRC/4。

5.1.4 、数据存储器（RAM）

RAM：128 字节



5.1.5 、系统寄存器

5.1.5.1 、系统寄存器表

	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

5.1.5.2 、系统寄存器说明

R =工作寄存器及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页及特殊标志寄存器

P4CON = P4配置控制寄存器

ADB = ADC数据缓存器

ADM = ADC模式寄存器

PnM = Pn模式控制寄存器

ADR = ADC分辨率选择寄存器

INTRQ =中断请求寄存器

PEDGE = P0.0边沿触发寄存器

OSCM =振荡器模式寄存器

INTEN =中断使能寄存器

WDTR =看门狗定时器清零寄存器

PCH, PCL=程序计数器

Pn = Pn数据缓存器

PnUR = Pn上拉电阻控制寄存器

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

TC0R= TC0自动装载数据缓存器

BZM= 蜂鸣器控制寄存器

STK0~STK7=堆栈缓存器

@YZ= 间接寻址寄存器

STKP= 堆栈指针

5.1.5.3 、系统寄存器位定义

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
0AEH				P4CON4	P4CON3	P4CON2	P4CON1	P4CON0	R/W	P4CON
0AFH	EVHENB						VHS1	VHS2	R/W	VREFH
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
0B4H	ADTS1	ADTS0		ADT4	ADT3	ADT2	ADT1	ADT0	R/W	ADT
0B8H					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				R/W	P5M
0C8H	ADCIRQ	TC1IRQ	TC0IRQ				P01IRQ	P00IRQ	R/W	INTRQ

0C9H	ADCIEN	TC1IEN	TC0IEN				P01IEN	P00IEN	R/W	INTEN
0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	TC0R7	TC0R6	TC0R5	TC0R4	TC0R3	TC0R2	TC0R1	TC0R0	W	TC0R
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH							PC9	PC8	R/W	PCH
0D0H				P04	P03	P02	P01	P00	R/W	P0
0D4H				P44	P43	P42	P41	P40	R/W	P4
0D5H				P54	P53				R/W	P5
0D8H					TC1X8	TC0X8	TC0GN		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	TC1ENB	TC1rate2	TC1rate1	TC1rate0	TC1CKS	ALOAD1	TC1OUT	PWM1OUT	R/W	TC1M
0DDH	TC1C7	TC1C6	TC1C5	TC1C4	TC1C3	TC1C2	TC1C1	TC1C0	R/W	TC1C
0DEH	TC1R7	TC1R6	TC1R5	TC1R4	TC1R3	TC1R2	TC1R1	TC1R0	W	TC1R
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E0H					P03R	P02R	P01R	P00R	W	P0UR
0E4H				P44R	P43R	P42R	P41R	P40R	W	P4UR
0E5H				P54R	P53R				W	P5UR
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H							S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH							S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH							STK1L	STK1L	STK1L	STK1L
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH							S0PC9	S0PC8	R/W	STK0H

注：1. 所有寄存器名都已在SN8ASM编译器中做了宣告；

2. 在SN8ASM编译器中，对寄存器的位进行操作，必须以“F”开头（如：B0BCLR FT0IEN）；

3. 指令“b0bset”、“b0bclr”、“bset”、“bclr”只能用于可读写的（R/W）寄存器。

5.1.5.4 、累加器

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

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

例：读/写ACC。

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

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

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

例：ACC 和工作寄存器中断保护操作。

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

5.1.5.5 、程序状态寄存器PFLAG

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

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	X	X	0	0	-	0	0	0

Bit [7:6] NT0, NPD：复位状态标志。

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

Bit 5 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。Bit

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

0 =系统工作电压VDD超过2.4V，低电压检测器没有工作；

1 =系统工作电压VDD低于2.4V，说明此时低电压检测器已处于监控状态。

Bit 2 C：进位标志。

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

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

Bit 1 DC：辅助进位标志。

1 = 加法运算时低四位有进位，或减法运算后没有向高四位借位；

0 = 加法运算时低四位没有进位，或减法运算后有向高四位借位。

Bit 0 Z：零标志。

1 = 算术/逻辑/分支运算的结果为零；

0 = 算术/逻辑/分支运算的结果非零。

注：关于标志位C、DC和Z的更多信息请参阅指令集相关内容。

5.1.5.6 、程序计数器PC

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

若程序执行CALL和JMP指令时，PC指向特定的地址。

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC	-	-	-	-	-	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
复位后	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

单地址跳转

在单片机里面，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。如果位检测为真，则跳过下一条指令。

```
        B0BTS1    FC          ; 若Carry_flag = 1，跳过下一条指令。
        JMP C0STEP            ; 否则跳到C0STEP。
          …
C0STEP:  NOP
        B0MOV A, BUF0 ;
        B0BTS0 FZ             ; 若Zero flag = 0，跳过下一条指令。
        JMP C1STEP            ; 否则跳到C1STEP。
        …
        C1STEP: NOP
```

如果ACC 等于指定的立即数则PC值加2，跳过下一条指令。

```
        CMPRS    A, #12H      ; 若ACC = 12H，跳过下一条指令。
        JMP C0STEP            ; 否则跳到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。
        …
```

…

5.1.5.7 、Y，Z寄存器

寄存器Y和Z都是8位缓存器，主要用途如下：

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

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

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

例：用Y、Z作为数据指针，访问bank0中025H处的内容。

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

例：利用数据指针@YZ对RAM 数据清零。

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

5.1.5.8 、R寄存器

8位缓存器R主要有以下两个功能：

- 作为工作寄存器使用；
- 存储执行查表指令后的高字节数据。

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

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

复位后	X	X	X	X	X	X	X	X

5. 2、寻址模式

5.2.1 、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。

5.2.2 、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
```

5.2.3 、间接寻址

通过指针寄存器（Y/Z）访问RAM数据。

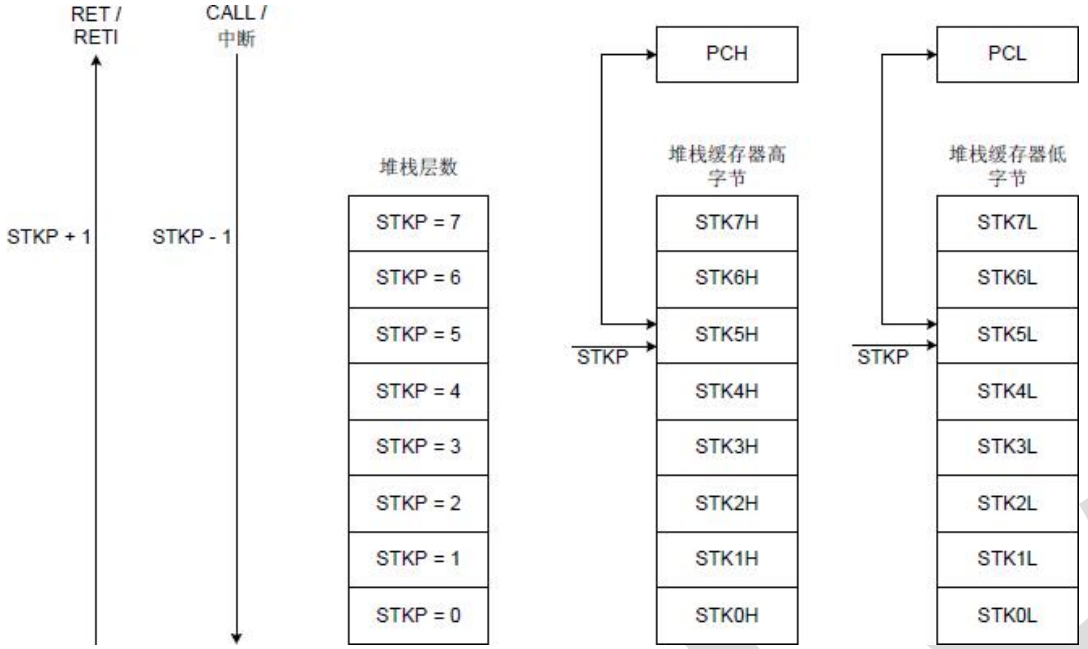
例：用@YZ实现间接寻址。

```
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
```

5. 3、堆栈(SP)

5.3.1 、概述

AIP8P102G的堆栈缓存器共有8层，程序进入中断或执行CALL指令时，用来存储程序计数器PC 的值。寄存器STKP为堆栈指针，指向堆栈缓存器顶层，STKnH和STKnL分别是各堆栈缓存器高、低字节。



5.3.2 、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，11位数据存储器STKnH和STKnL用于暂存堆栈数据。以上寄存器都位于bank 0。

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

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

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

Bit[2:0] STKPBn：堆栈指针（n = 0~2）。

Bit 7 GIE：全局中断控制位。

0 =禁止；

1 =使能。

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

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

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

5.3.3 堆栈操作举例

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

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

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

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

6、复位

6. 1、概述

AIP8P102G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）。

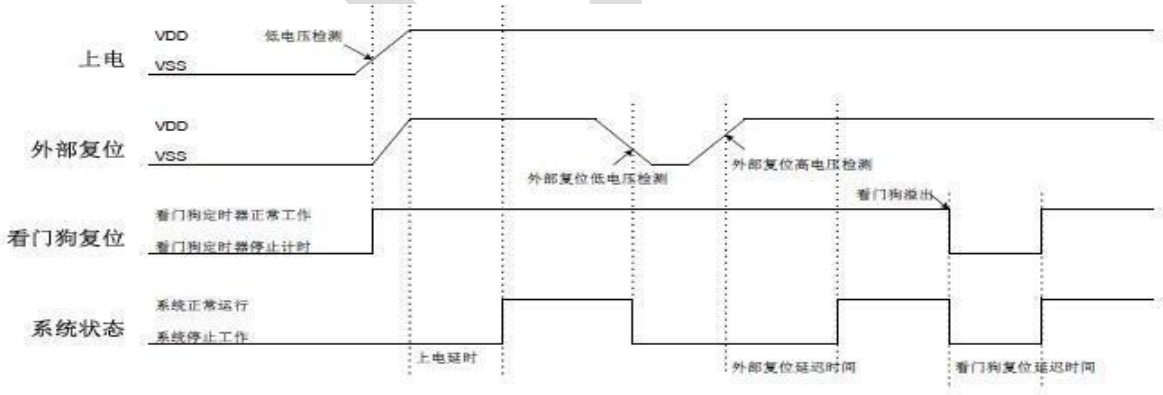
上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器PC清零。复位结束后，系统从向量0000H处重新开始运行。PFLAG寄存器的NT0和NPD两个标志位能够给出系统复位状态的信息。用户可以编程控制NT0和NPD，从而控制系统的运行路径。

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	LVD36	LCD24	-	C	DC	Z
读/写	R/W	R/W	R	R	-	R/W	R/W	R/W
复位后	X	X	0	0	-	0	0	0

Bit [7:6] NT0, NPD：复位状态标志。

NT0	NPD	复位情况	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。



6. 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。

6. 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。

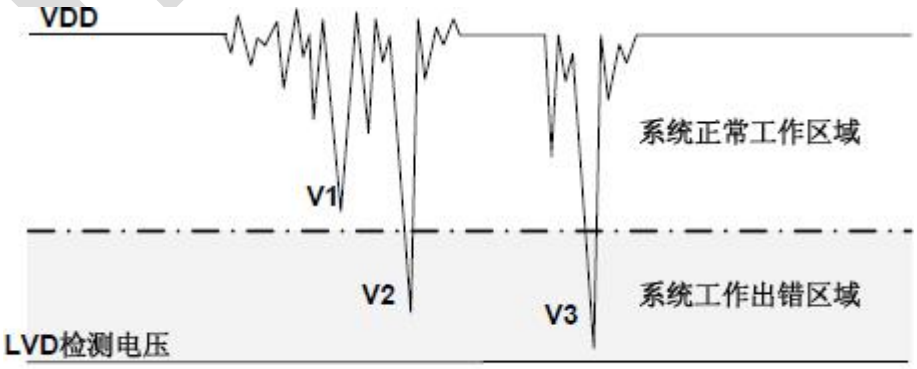
看门狗定时器应用注意事项：

- 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。

6. 4、掉电复位 LVR/LVD

6.4.1 、概述

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



掉电复位示意图

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：

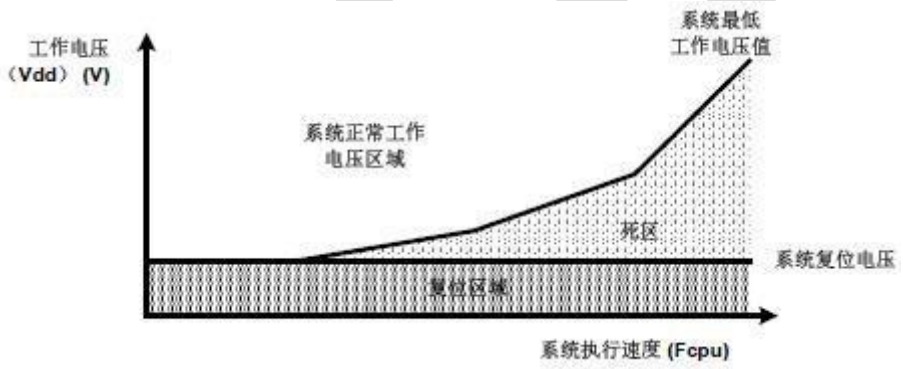
DC运用中一般都采用电池供电，当电池电压过低或单片机驱动负载时，系统电压可能跌落并进入死区。这时，电源不会进一步下降到LVD检测电压，因此系统维持在死区。 AC运用中：

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。

6.4.2 、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。



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。

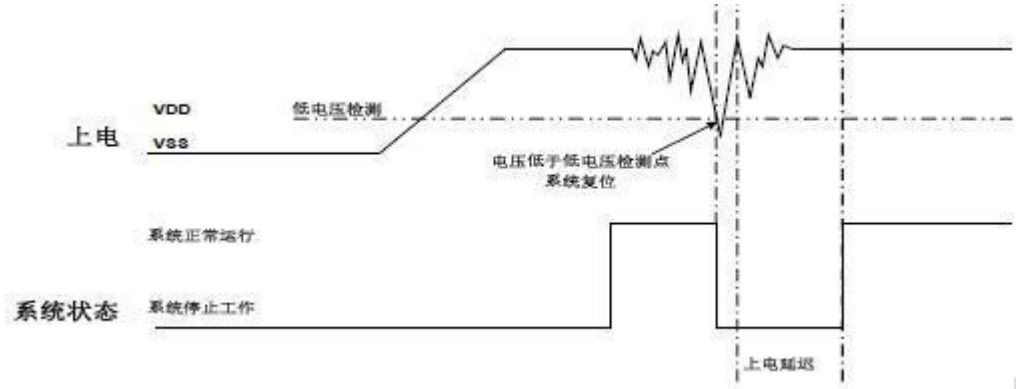
6.4.3 、掉电复位性能改进

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：



低电压检测（LVD）是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	LCD24	-	C	DC	Z
读/写	R/W	R/W	R	R	-	R/W	R/W	R/W
复位后	X	X	0	0	-	0	0	0

Bit 5 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。Bit 4 LVD24：2.4V LVD工作电压标志，LVD编译选项为LVD_M时有效。

0＝系统工作电压VDD超过2.4V，低电压检测器没有工作；

1＝系统工作电压VDD低于2.4V，说明此时低电压检测器已处于监控状态。

LVD	LVD编译选项		
	LVD_L	LVD_M	LVD_H
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；

LVD24标志位无意义。

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。

- 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。

6. 5、外部复位

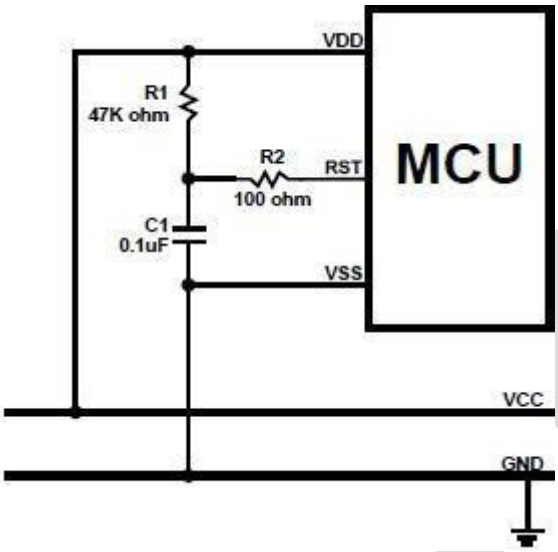
外部复位功能由编译选项“Reset_Pin”控制。将该编译选项置为“Reset”，可使能外部复位功能。外部复位引脚为施密特触发结构，低电平有效。复位引脚处于高电平时，系统正常运行。当复位引脚输入低电平信号时，系统复位。外部复位操作在上电和正常工作模式时有效。需要注意的是，在系统上电完成后，外部复位引脚必须输入高电平，否则系统将一直保持在复位状态。外部复位的时序如下：

- 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。

6. 6、外部复位电路

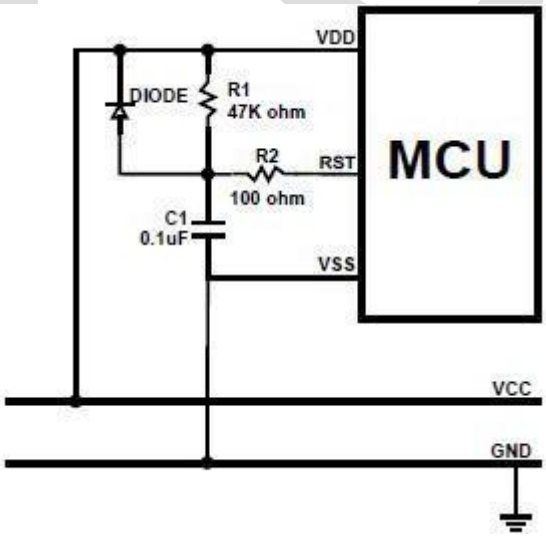
6.6.1 、RC复位电路



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

注：此RC复位电路不能解决非正常上电和掉电复位问题。

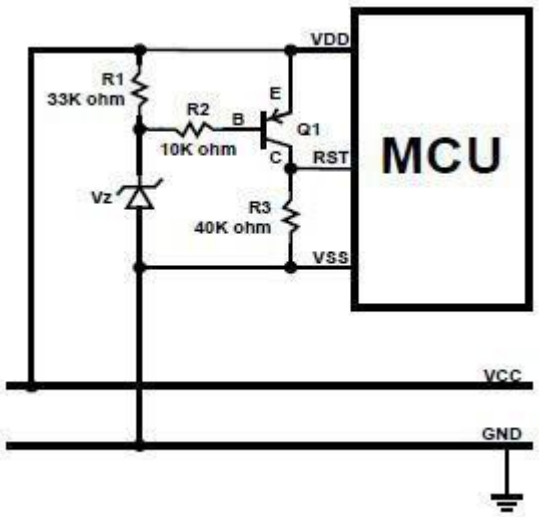
6.6.2 、二极管及RC复位电路



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

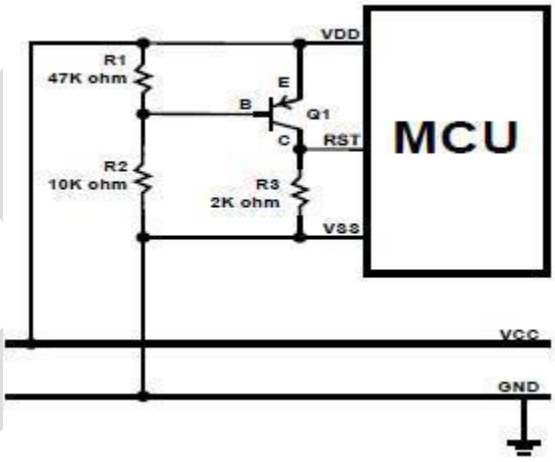
注：“基本RC复位电路”和“二极管及RC复位电路”中的电阻R2都是必不可少的限流电阻，以避免复位引脚ESD（Electrostatic Discharge）或EOS（Electrical Over-stress）击穿。

6.6.3 、稳压二极管复位电路



稳压二极管复位电路是一种简单的LVD电路，基本上可以完全解决掉电复位问题。如上图电路中，利用稳压管的击穿电压作为电路复位检测值，当VDD高于“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。

6.6.4 、电压偏置复位电路

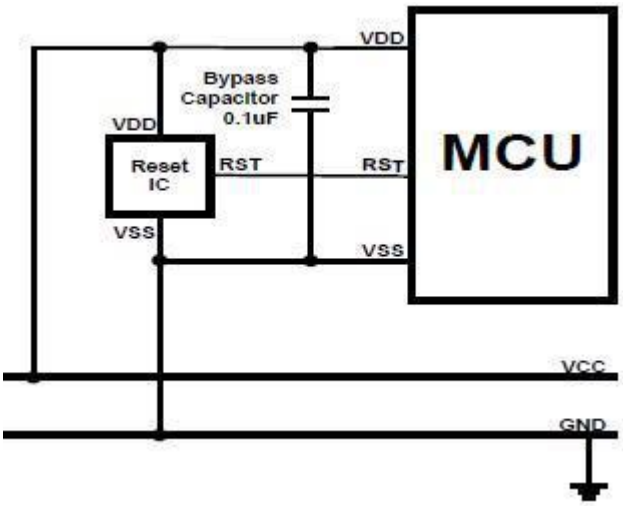


电压偏置复位电路是一种简单的LVD电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，这种复位电路的检测电压值的精确度有所降低。电路中，R1和R2构成分压电路，当VDD 高于和等于分压值“0.7V x (R1 + R2)/ R1”时，三极管集电极C输出高电平，单片机正常工作；VDD低于“0.7V x (R1 + R2) / R1”时，集电极C输出低电平，单片机复位。

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。

6.6.5 、外部IC复位



外部复位也可以选用IC进行外部复位，但是这样一来系统成本将会增加。针对不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。

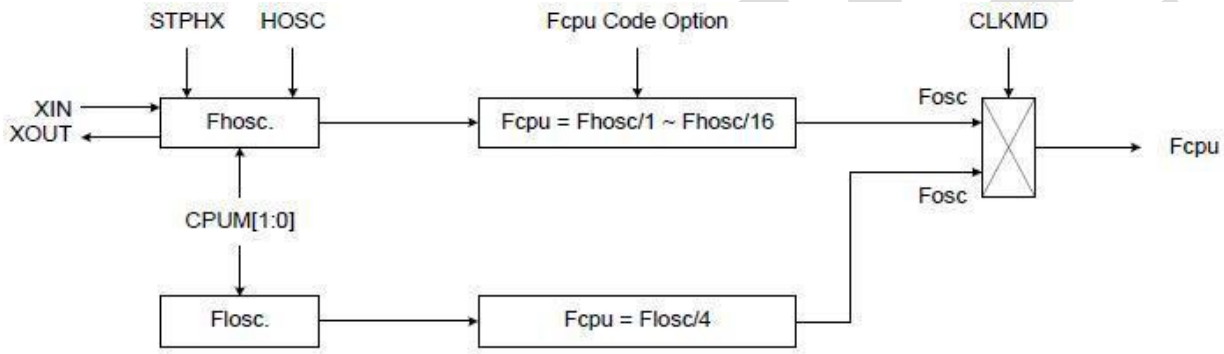
7、系统时钟

7. 1、概述

AIP8P102G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（ILRC 16KHz@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。

7. 2、时钟框图



- HOSC：High_Clk编译选项。
- Fhosc：外部高速时钟/内部高速RC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。

7. 3、OSCM 寄存器

寄存器OSCM控制振荡器的状态和系统模式。

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

Bit 1 STPHX：内部/外部高速振荡器控制位。

0 =运行；

1 =停止。内部低速RC振荡器仍然运行。

Bit 2 CLKMD：高/低速时钟模式控制位。

0 = 普通（双时钟）模式，系统时钟来自高速时钟；

1 = 低速模式，系统时钟来自低速时钟。

Bit[4:3] CPUM[1:0]：CPU工作模式控制位。

00 =普通模式；

01 =睡眠模式；

10 =绿色模式；

11 =系统保留。

例：停止高速振荡器。

```
B0BSET FSTPHX          ；停止内部/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
```

7. 4、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。

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

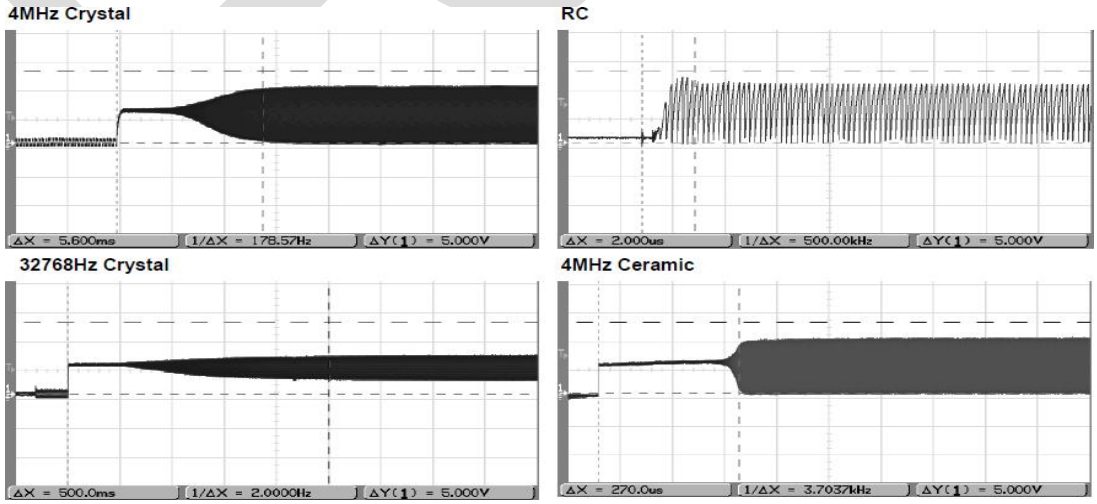
7.4.1 、内部高速 RC 振荡器

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：系统高速时钟来自内置16MHz RC振荡器，XIN/XOUT引脚作为普通的I/O引脚。

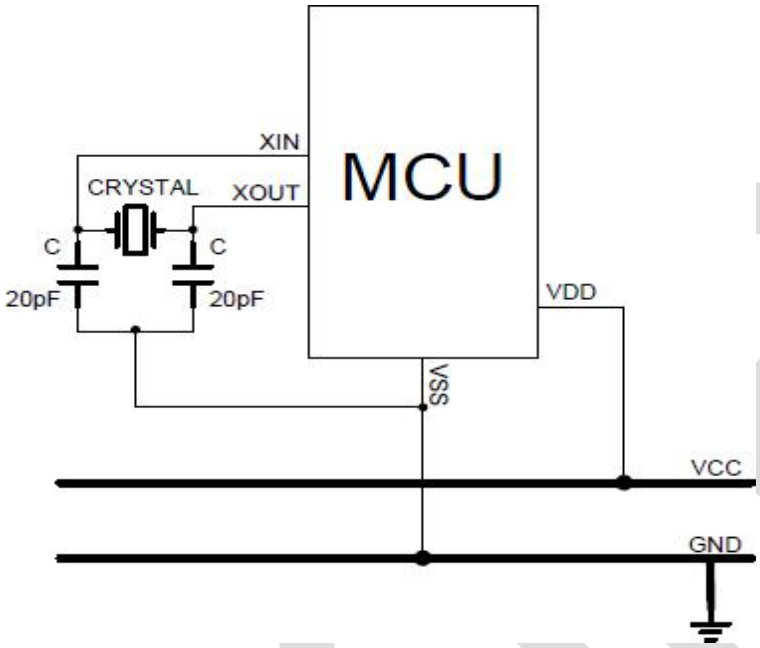
7.4.2 、外部高速时钟

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



7.4.2.1 、石英/陶瓷振荡器

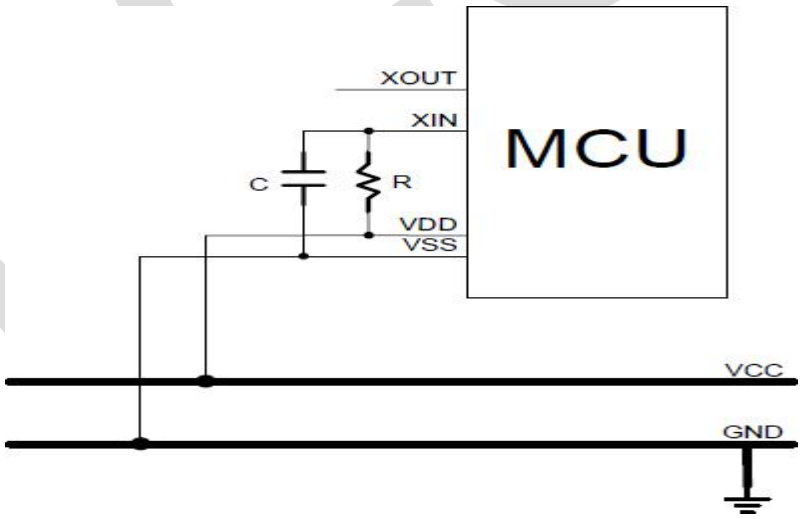
石英/陶瓷振荡器由XIN/XOUT口驱动，对于高速、普通和低速三种不同工作模式，振荡器的驱动电流也不同。不同的工作模式下，编译选项“High_Clk”支持不同的频率：12MHz，4MHz 及32KHz。



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。

7.4.2.2 RC振荡器

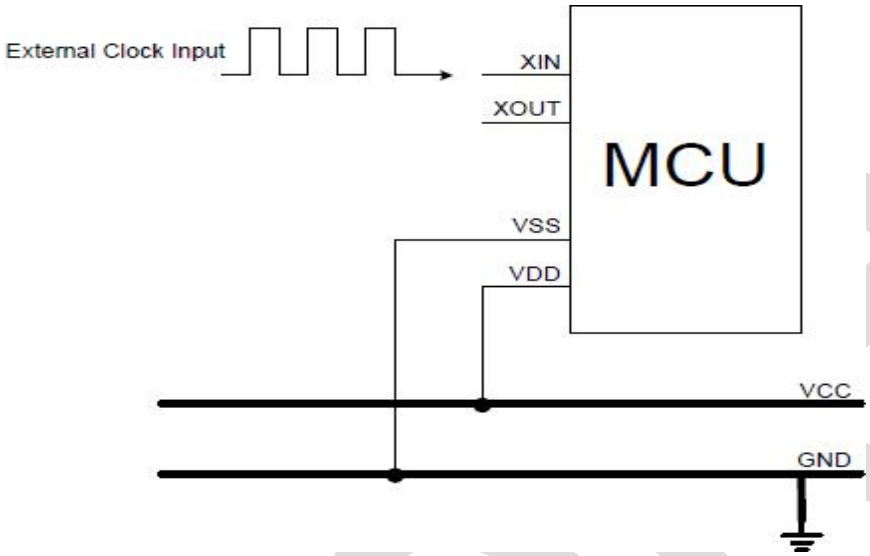
通过编译选项High_Clk的设置可控制RC振荡器的选择，RC振荡器输出频率最高可达10MHz。改变R可改变输出频率的大小，电容C的最佳容量为50P~100P，引脚XOUT为通用I/O口，如下图所示：



注：电容C和电阻R应尽可能的靠近单片机的VDD。

7.4.2.3 、外部时钟源

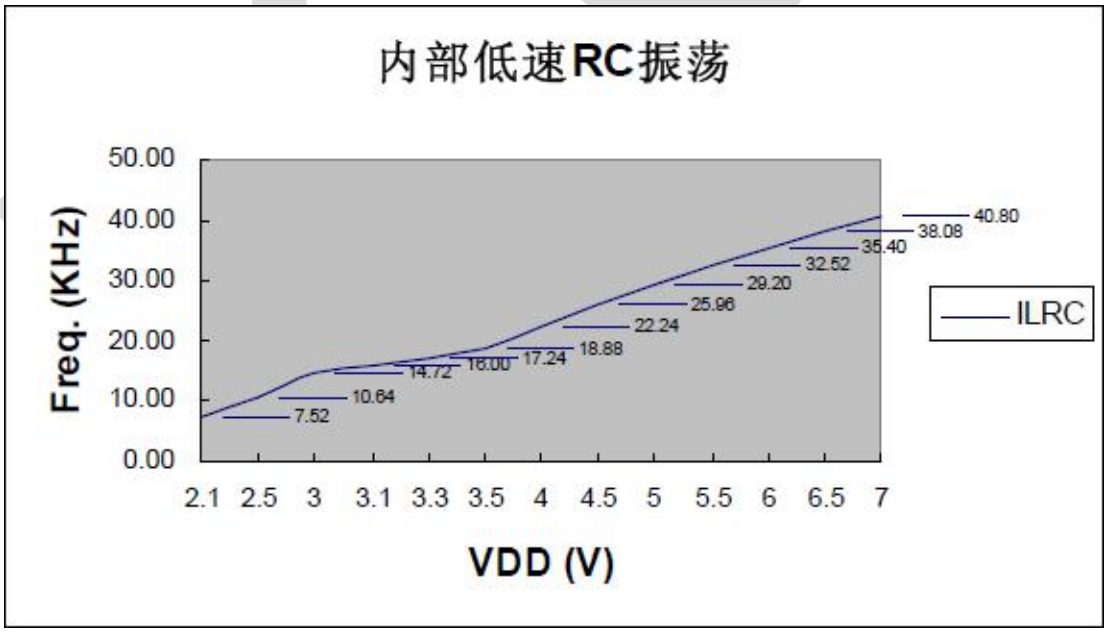
单片机可选择外部时钟信号作为系统时钟，由编译选项High_Clk控制，从XIN 脚送入。XOUT引脚作为普通的I/O口。



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。

7. 5、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。

- 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时，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(32K，禁止看门狗)设置决定内部低速时钟的状态。

7.5.1、系统时钟测试

在设计过程中，用户可通过软件指令周期Fcpu对系统时钟速度进行测试。

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。

8、系统工作模式

8. 1、概述

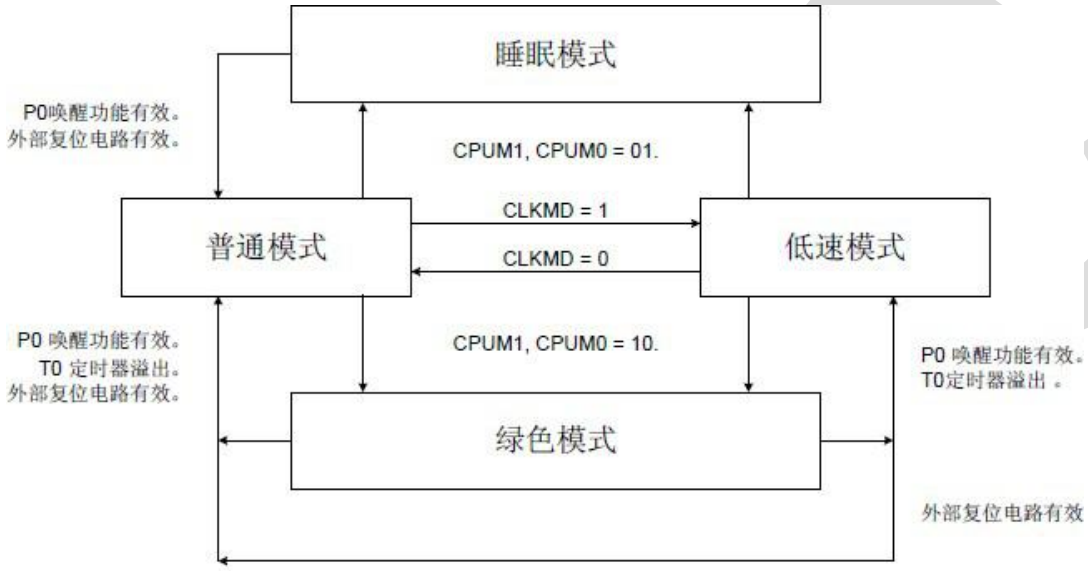
AIP8P102G可在以下4种模式下工作。

普通模式（高速模式）；

低速模式；

睡眠模式；

绿色模式。



系统模式切换框图

工作模式描述

系统模式	普通模式	低速模式	绿色模式	睡眠模式	注释
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	*有效	*有效	*有效	无效	*T0ENB=1 时有效
TC0	*有效	*有效	无效	无效	*TC0ENB=1 时有效
看门狗	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	参考编译选项描述

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时16K，5V时32K）。

8.2、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**)**。

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

例：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 X 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

8. 3、唤醒时间

8.3.1 、概述

系统在睡眠模式下并不执行程序。唤醒触发信号可以将系统唤醒进入普通模式或低速模式。唤醒触发信号来自外部触发信号（P0的电平变化）和内部触发信号（T0定时溢出）。

- 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。

8.3.2 、唤醒时间

系统进入睡眠模式后，高速时钟停止运行。把系统从睡眠模式下唤醒时，单片机需要等待2048个外部高速振荡器时钟周期以使振荡电路进入稳定工作状态，等待的这一段时间就称为唤醒时间。唤醒时间结束后，系统才进入到普通模式。

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

唤醒时间计算如下：

唤醒时间= 1/Fosc * 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：

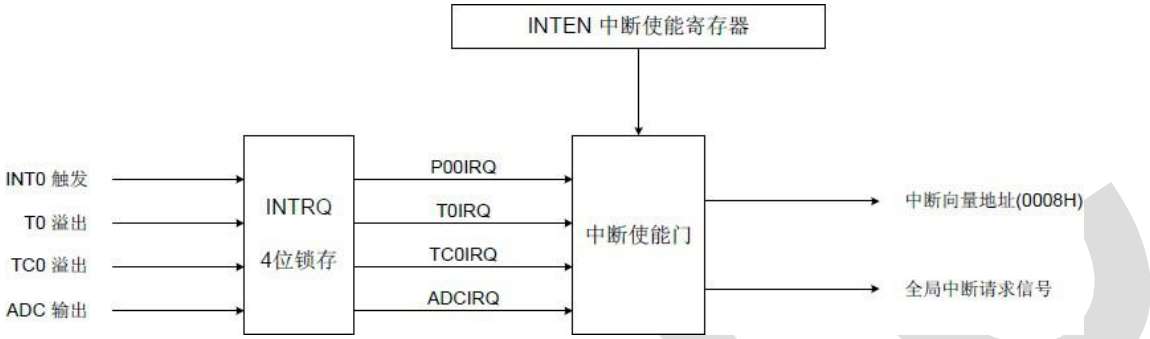
唤醒时间= 1/Fosc * 2048=0.512 ms (Fosc=4MHz)

总的唤醒时间=0.512 ms +振荡器启动时间

9、中断

9. 1、概述

AIP8P102G共有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。

9. 2、中断使能寄存器 INTEN

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

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

Bit 0 P00IEN ：P0.0外部中断（INT0）控制位。

0 = 禁止；

1 = 使能。

Bit 4 T0IEN ：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。

9.3、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，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	-	-	-	P00IRQ
读/写	R/W	-	R/W	R/W	-	-	-	R/W
复位后	0	-	0	0	-	-	-	0

Bit 0 P00IRQ： P0.0中断（INT0）请求标志位。

0 = INT0无中断请求；

1 = INT0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。

9.4、GIE 全局中断

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

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
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。

9.5、PUSH, POP 处理

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

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

例：对ACC和PAFLG进行入栈保护。

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

9.6、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：1、INT0的中断请求被P0.0的唤醒触发功能锁定。

2、P0.0的中断触发边沿由PEDGE控制。

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

Bit[4:3] P00G[1:0]： P0.0中断触发控制位。

00 = 保留；

01 = 上升沿触发；

10 = 下降沿触发；

11 = 上升/下降沿触发（电平触发）。

例：INT0中断请求设置，电平触发。

```
MOV     A, #18H
B0MOV   PEDGE, A        ; 设置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。
```

9.7、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。
```

9.8、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=10ms。
                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。
```

9.9、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。
        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。
```

9.10、多中断操作举例

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

中断	有效触发

P00IRQ	由PEDGE控制
T0IRQ	T0C溢出
TC0IRQ	TC0C溢出
ADCIRQ	AD 转换结束

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

例：多中断条件下检测中断请求。

```
ORG 8H
JMP INT_SERVICE
INT_SERVICE:
        …                   ; 保存ACC和PFLAG。
    INTP00CHK:              ; 检查是否有INT0中断请求。
    B0BTS1 FP00IEN          ; 检查是否使能INT0中断。
    JMP INTT0CHK            ; 跳到下一个中断。
    B0BTS0 FP00IRQ          ; 检查是否有INT0中断请求。
    JMP INTP00              ; 进入INT0中断。
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 INTADCHK            ; 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。
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
```

10、IO 口

10. 1、I/O 口模式

寄存器PnM控制I/O口的工作模式。

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	P07M	P06M	P05M	P04M	-	P02M	P01M	P00M
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	-	-	-	P44M	P43M	P42M	P41M	P40M
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	-	-	-	P54M	P53M	P52M	P51M	P50M
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

Bit[7:0] PnM[7:0]： Pn模式控制位（n = 0~5）。

0 =输入模式；

1 =输出模式。

注：

1. 用户可通过位操作指令（B0BSET，B0BCLR）对I/O口进行编程控制；
2. P0.3只能作为输入引脚，寄存器P0M.3的值保持为“1”。

例：I/O 模式选择。

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

10. 2、I/O 口上拉电阻

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	P07R	P06R	P05R	P04R	-	P02R	P01R	P00R
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	-	-	-	P44R	P43R	P42R	P41R	P40R
读/写	-	-	-	W	W	W	W	W
复位后	-	-	-	0	0	0	0	0

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

注：P0.3 只可用作输入引脚，无上拉电阻，寄存器P0UR.3保持为“1”。

例：I/O上拉寄存器。

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

10. 3、I/O 口数据寄存器

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

0D4H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4	-	-	-	P44	P43	P42	P41	P40
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	-	-	-	P54	P53	P52	P51	P50
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

注：当使能外部复位时， P03的值保持为1。

例：从输入口读取数据。

```
B0MOV A, P0         ; 从P0读数据。
B0MOV A, P4         ; 从P4读数据。
B0MOV A, P5         ; 从P5读数据。
```

例：写数据到输出端。

```
MOV A, #0FFH        ; 立即数0FFH写入所有输出口。
B0MOV P0, A
B0MOV P4, A
B0MOV P5, A
```

例：写1 位数据到输出口。

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

10. 4、P4 与 ADC 共用引脚

P4口和ADC的输入口共用。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。

0AFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P4CON	-	-	-	P4CON4	P4CON3	P4CON2	P4CON1	P4CON0
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	0

Bit[4:0] P4CON[4:0] ： 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	-	CHS2	CHS1	CHS0
读/写	R/W	R/W	R/W	R/W	0	R/W	R/W	R/W
复位后	0	0	0	0	-	0	0	0

Bit 4 GCHS：ADC输入通道选择位。

0 = 禁止AIN通道；

1 = 开启AIN通道。

Bit[2:0] CHS[2:0]：ADC输入通道选择位。

000 = AIN0；001 = AIN1；010 = AIN2；011 = AIN3；100 = AIN4；101 = AIN5。

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。

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（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。
; 开启P4.1的输入功能。
    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（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。
; 开启P4.1的输出模式。
    B0BSET P4M.1      ; 设置 P4.1 为输出模式
```

11、定时器

11.1、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）

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。

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

例：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。

```
MOV 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。
```

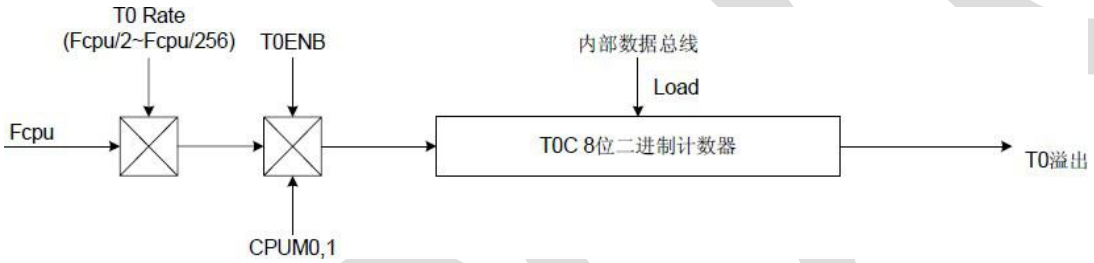
```
B0MOV WDTR, A
…
CALL SUB1
CALL SUB2
 …
 …
JMP MAIN
```

11. 2、定时器 T0

11.2.1 、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。



11.2.2 、T0M 模式寄存器

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0NB	T0rate2	T0rate1	T0rate0				
读/写	R/W	R/W	R/W	R/W				
复位后	0	0	0	0				

Bit [6:4] T0RATE[2:0]：T0分频选择位。

000 = fcpu/256；

001 = fcpu/128；

… ；

110 = fcpu/4；

111 = fcpu/2。

Bit 7 T0ENB：T0启动控制位。

0 = 关闭T0计数器；

1 = 开启T0计数器。

11.2.3 、T0C 计数寄存器

T0C用于控制T0的间隔时间。

0D9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TOC	T0C7	TOC6	TOC5	TOC4	TOC3	TOC2	TOC1	TOC0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

T0C初始值计算公式如下：

T0C初始值= 256-（T0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

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

11.2.4 、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
```

11. 3、定时/计数器 TC0

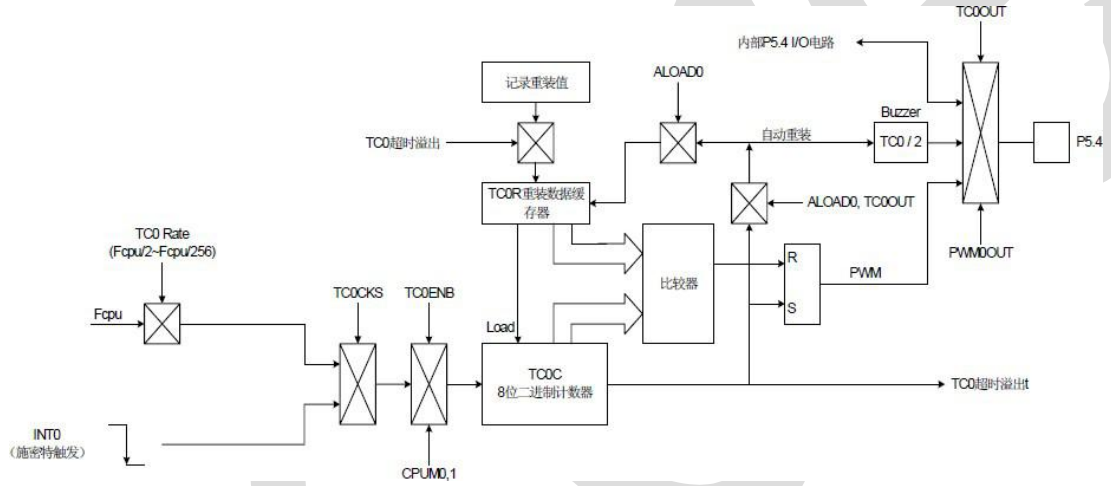
11.3.1 、概述

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的主要作用如下：

- **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。



11.3.2 、TC0M模式寄存器

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TC0M	TC0ENB	TCOrate2	TCOrate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 0 PWM0OUT： PWM输出控制。

0 = 禁止PWM输出；

1 = 使能PWM输出，PWM输出占空比由TC0OUT和ALOAD0控制。

Bit 1 TC0OUT： TC0 超时输出信号控制。仅当PWM0OUT= 0时有效。

0 = 禁止，P5.4作为输入/输出口；

1 = 使能，P5.4输出TC0OUT 信号。

Bit 2 ALOAD0： 自动装载控制。仅当PWM0OUT = 0时有效。

0 = 禁止TC0自动装载；

1 = 使能TC0自动装载。

Bit 3 TC0CKS： TC0时钟信号控制位。

0 = 内部时钟（Fcpu或Fosc）；

1 = 外部时钟，由P0.0/INT0输入。

Bit [6:4] TC0RATE[2:0]：TC0分频选择位。

000 = fcpu/256；

001 = fcpu/128；

…；

110 = fcpu/4；

111 = fcpu/2。

Bit 7 TC0ENB： TC0 启动控制位。

0 = 禁止TC0定时器；

1 = 开启TC0定时器。

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）。

11.3.3 、TC0C 计数寄存器

TC0C控制TC0的时间间隔。

0DBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TCOC	TC0C7	TCOC6	TCOC5	TCOC4	TCOC3	TCOC2	TCOC1	TCOC0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

TC0C初始值的计算公式如下：

TC0C初始值= 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：

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

例：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

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

11.3.4 、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值先被暂存在TC0R的第一个缓存器中，直到当前TC0溢出后，才被真正存入TC0R缓存器中，从而避免TC0中断时间出错以及PWM和蜂鸣器误动作。

注：在PWM模式下，系统自动开启自动装载功能。位寄存器ALOAD0用于控制溢出范围。

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

TC0R初始值计算公式如下：

TC0R初始值= N -（TC0 中断间隔时间 * 输入时钟）上式中，N为TC0 的最大计数范围。TC0的溢出时间有如下六种可能情况：

TC0CKS	PWM0	ALOAD0	TC0OUT	N	TC0C范围	TC0C二进制计数范围	注释
0	0	x	x	256	00H~0FFH	00000000b~11111111b	每计数256次溢出
	1	0	0	256	00H~0FFH	00000000b~11111111b	每计数256次溢出
	1	0	1	64	00H~3FH	xx000000b~xx111111b	每计数64次溢出
	1	1	0	32	00H~1FH	xxx00000b~xxx11111b	每计数32次溢出
	1	1	1	16	00H~0FH	xxxx0000b~xxxx1111b	每计数16次溢出
1	-	-	-	256	00H~0FFH	00000000b~11111111b	每计数256次溢出

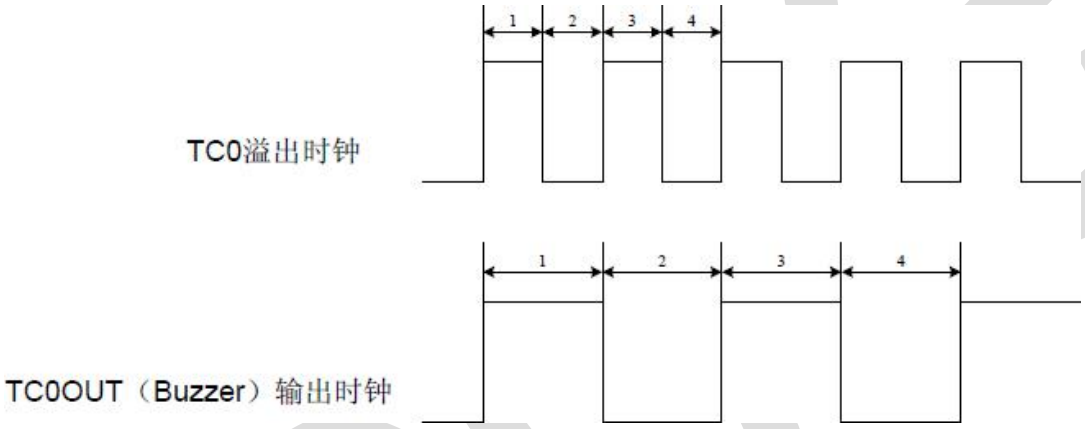
例：TC0中断间隔时间设置为10ms，时钟源选Fcpu（TC0KS = 0），无PWM输出（PWM0 = 0），高速时钟为

外部4MHz，Fcpu = Fosc/4，TC0RATE = 010（Fcpu/64）。

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

11.3.5 、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：



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

例：设置TC0OUT（P5.4）。

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

11.3.6 、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 A, #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
```

11.4 、PWM0

11.4.1、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

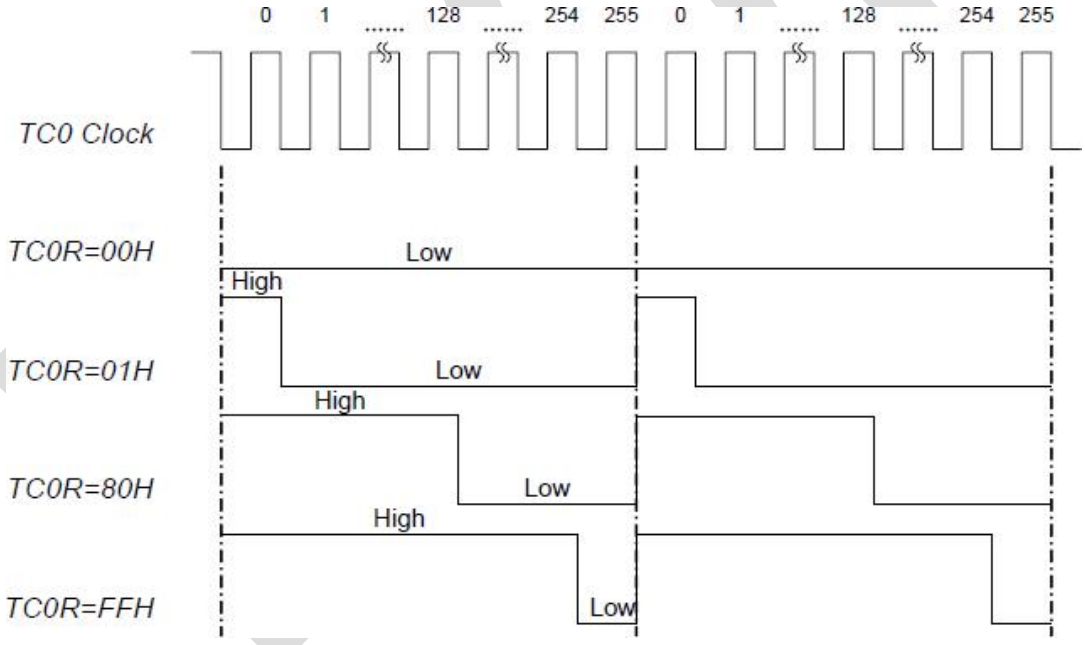
（256、64、32和16）。8位计数器TC0C计数过程中不断与TC0R相比较，当TC0C计数到两者相等时，PWM 输出低电平，当TC0C再次从零开始计数时，PWM被强制输出高电平。PWM0输出占空比= TC0R/计数量程（计数量程= 256、64、32 或16）。

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。

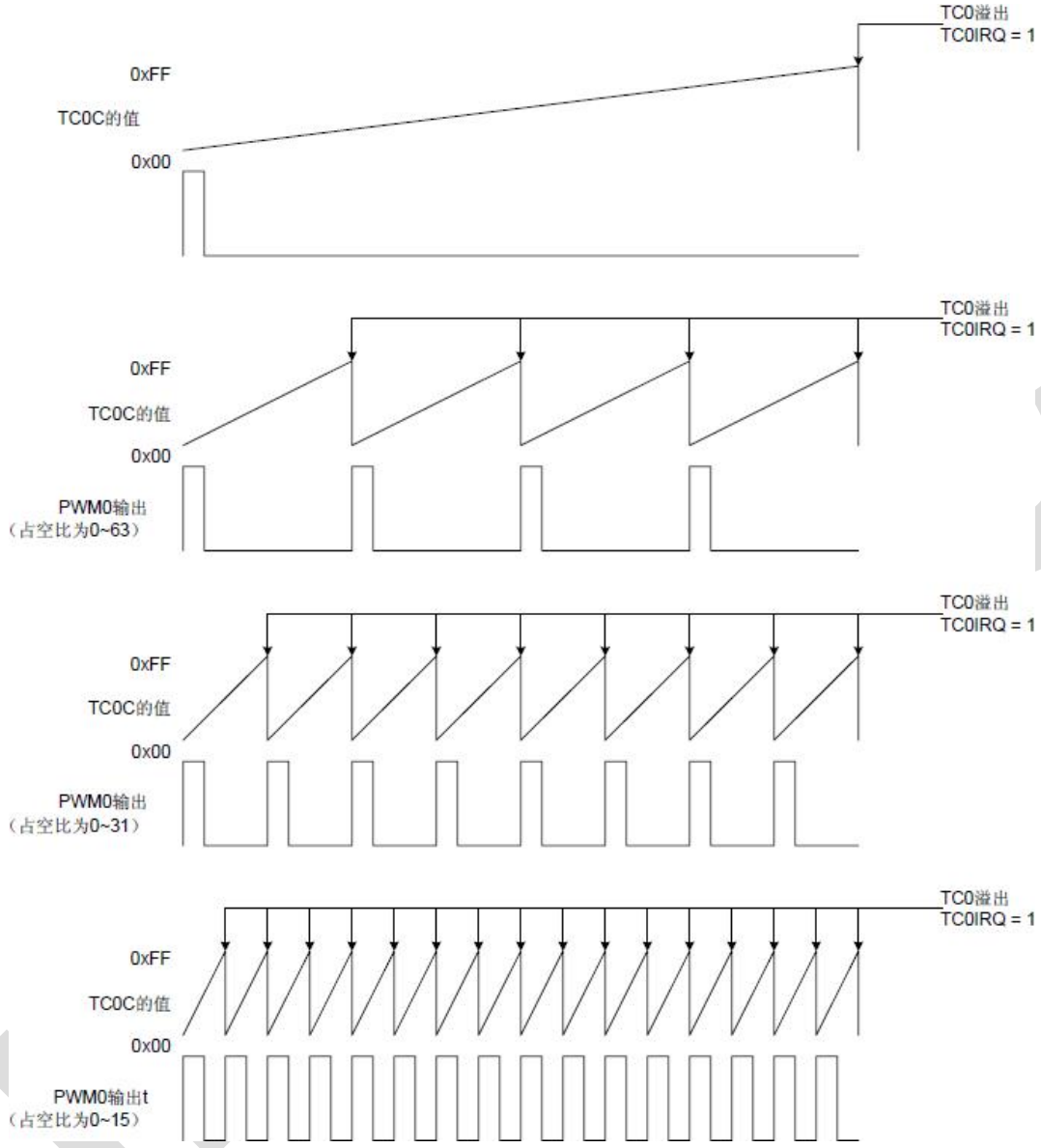
ALOD 0	TC0OUT	PWM占空比范围	TC0C有效值	TC0R有效值	MAX.PWM输出频率（Fcpu=4MHz）	注释
0	0	0/256~255/256	00H~0FFH	00H~0FFH	7.8125K	每计数256次溢出
0	1	0/64~63/64	00H~3FH	00H~3FH	31.25K	每计数256次溢出
1	0	0/32~31/32	00H~1FH	00H~1FH	32.5K	每计数256次溢出
0	1	0/16~15/16	00H~0FH	00H~0FH	125K	每计数256次溢出

PWM输出占空比随TC0R的变化而变化：0/256~255/256。



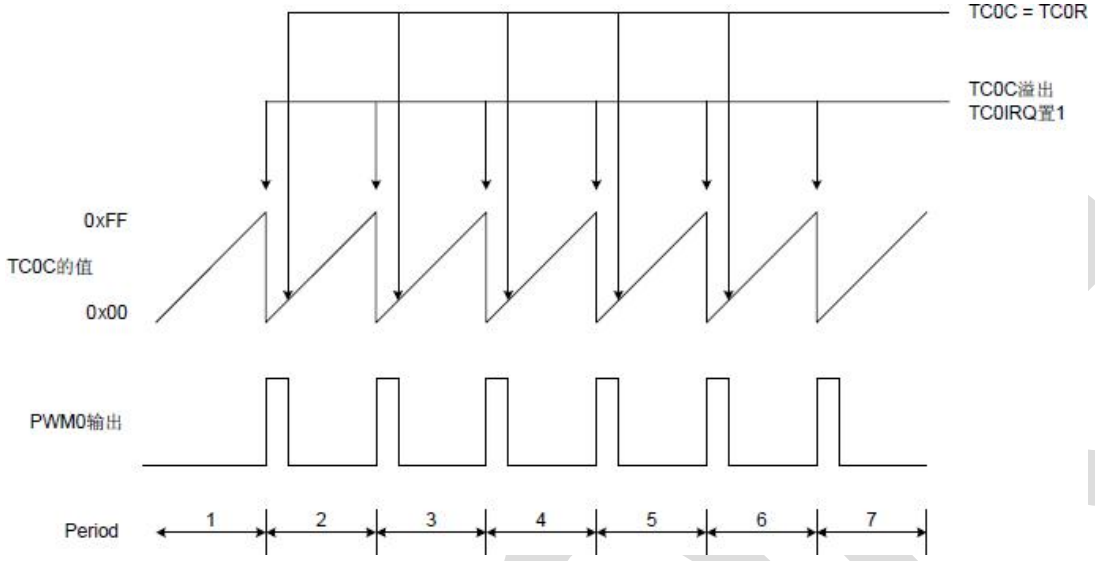
11.4.2 、TC0IRQ 和 PWM 输出占空比

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：

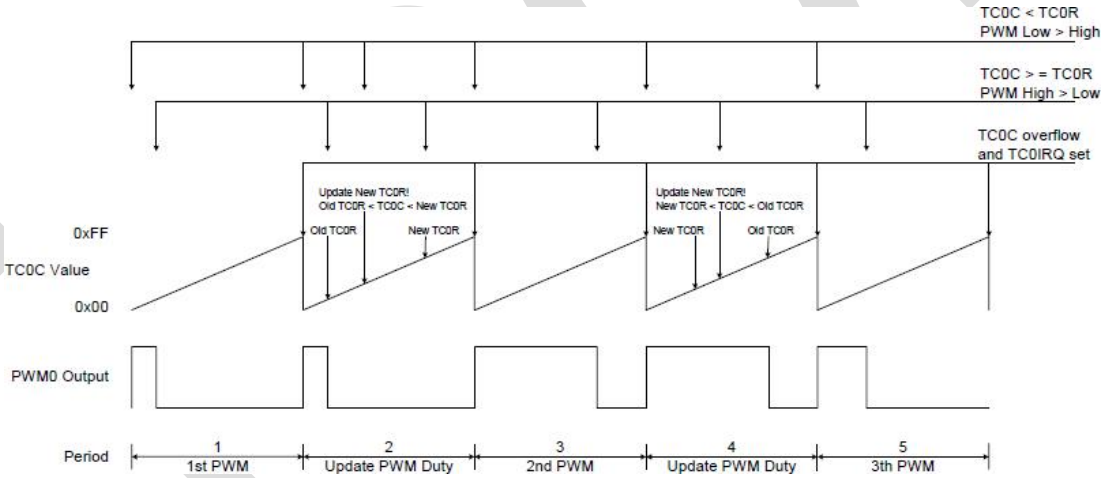


11.4.3 、PWM 输出占空比与 TC0R 的变化

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。



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：



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。

11.4.4 、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。

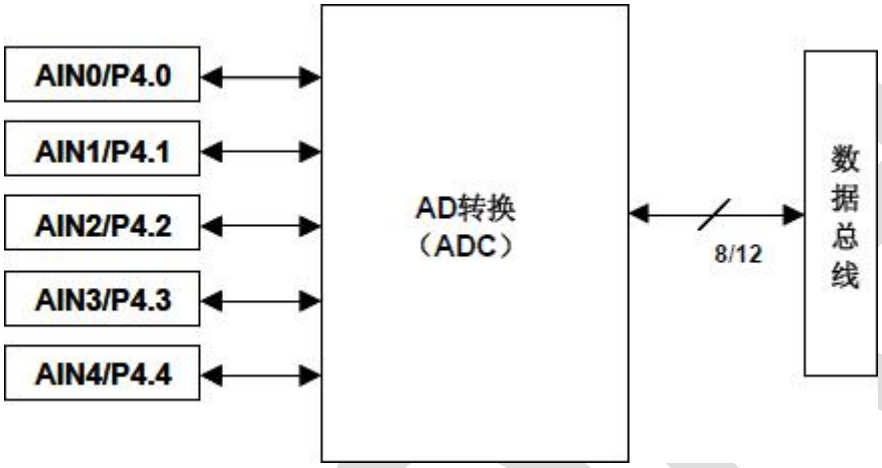
例：改变TC0R的内容。

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

12、通道 ADC

12. 1、概述

AIP8P102G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。



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。

12. 2、ADM 寄存器

0B1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADM	ADENB	ADS	EOC	GCHS	-	CHS2	CHS1	CHS0
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 7 ADENB：ADC控制位。

0 = 禁止；

1 = 允许。

Bit 6 ADS：ADC启动位 。

0 = 停止；

1 = 转换开始。

Bit 5 EOC：ADC状态控制位。

0 = 转换过程中；

1 = 转换结束，ADS复位。

Bit 4 GCHS：ADC输入通道控制位。

0 = 禁止AIN 通道；

1 = 使能AIN 通道。

Bit[2:0] CHS[2:0]：ADC输入通道选择位。

000 = AIN0；001 = AIN1；010 = AIN2；011 = AIN3；100 = AIN4。

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。

12.3 、ADR 寄存器

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

Bit 6,4 **ADCKS [1:0]:** ADC 时钟源选择位

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

Bit [3:0] ADB [3:0]: ADC 数据缓存器。

ADB11~ADB4（8位ADC）；

ADB11~ADB0（12位ADC）。

注：寄存器ADR的ADB[3:0]的初始值是未知的。

12.4 、ADB 寄存器

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	X	X	X	X	X	X	X	X

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

8 位数据缓存器ADB用来保存AD转换结果的高8位（bit4~bit11），转换结果的低4位则保存在ADR 寄存器中。ADB为只读寄存器，在8位ADC模式下，AD转换结果保存在寄存器ADB中；在12位模式下， 则分别保存在寄存器ADB和ADR中。

AIN的输入电压v.s. ADB的输出数据

AIN n	AD B11	AD B10	AD B9	AD B8	AD B7	AD B6	AD B5	AD B4	AD B3	AD B2	AD B1	AD B0
0/4096*VR EFH	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	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：寄存器 ADB 各位的初始值是未知的。

12.5 、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。

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

Bit[4:0] P4CON[4:0]：P4.n配置控制位。

0 = P4.n作为模拟输入（ADC输入）引脚或者数字I/O引脚；

1 = P4.n只能作为模拟输入引脚，不能作为数字I/O引脚。

注：当P4.n为基本I/O 而不是ADC通道时，P4CON.n必须置“0”，否则P4.n的数字I/O信号会被隔离。

12.6 、ADC 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）

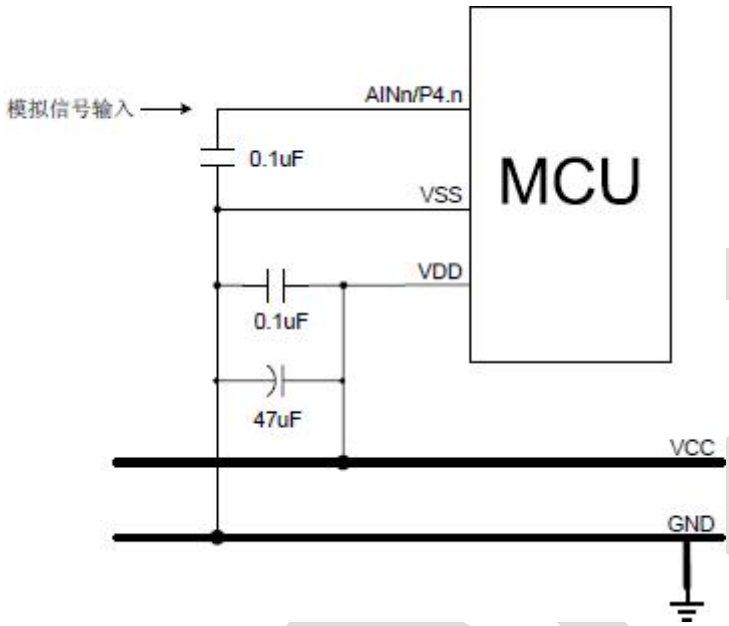
ADLEN	ADCKS1	ADCKS0	ADC Clock	ADC 转换时间
1(12-bit)	0	0	Fcpu/16	1/(4MHz/16/4)*16=256us
	0	1	Fcpu/8	1/(4MHz/8/4)*16=128us
	1	0	Fcpu/1	1/(4MHz/4)*16=16us
	1	1	Fcpu/2	1/(4MHz/2/4)*16=32us

12.7 、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。
```

12.8 、ADC 电路

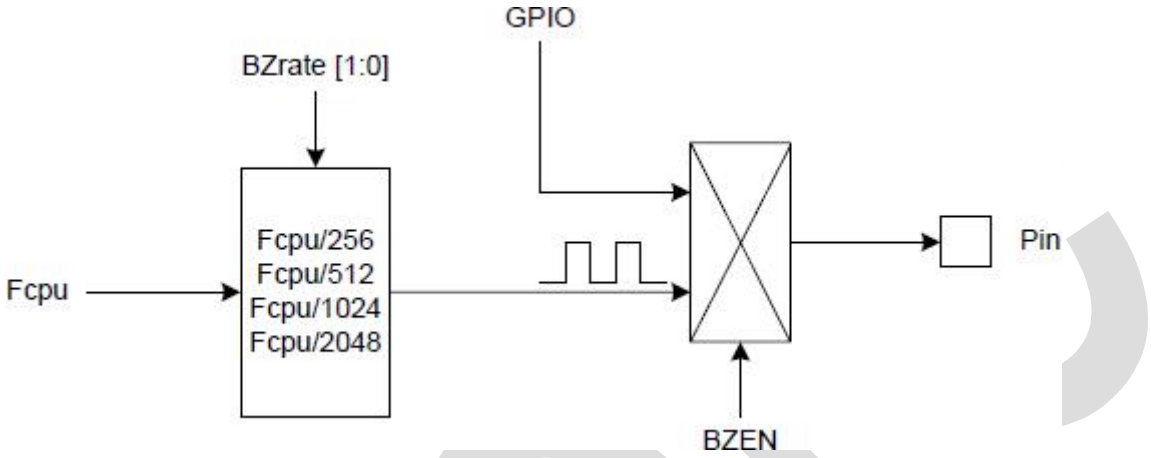


在ADC输入引脚接入0.1uF 的电容可以滤除电源端的杂汛。

13、2K/4K 蜂鸣器（Buzzer）输出

13. 1、概述

AIP8P102G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）。



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：

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。

13. 2、BZM 模式寄存器

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: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。

Bit[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。

14、指令表

指令	指令格式		描述	C	DC	Z	周期
MOVE	MOV	A,M	A ← M	-	-	√	1
	MOV	M,A	M ← A	-	-	-	1
	B0MOV	A,M	A ← M (bank 0)	-	-	√	1
	B0MOV	M,A	M (bank 0) ← A	-	-	-	1
	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I。M只支持80H~87H之间的寄存器（如PFLAG,R,Y,Z...）	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)。	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]。	-	-	-	2
ARITHMETIC	ADC	A,M	A ← A + M + C，如果产生进位则C=1，否则C=0。	√	√	√	1
	ADC	M,A	M ← A + M + C，如果产生进位则C=1，否则C=0。	√	√	√	1+N
	ADD	A,M	A ← A + M，如果产生进位则C=1，否则C=0。	√	√	√	1
	ADD	M,A	M ← A + M，如果产生进位则C=1，否则C=0。	√	√	√	1+N
	B0ADD	M,A	M (bank 0) ← M (bank 0) + A，如果产生进位则C=1，否则C=0。	√	√	√	1+N
	ADD	A,I	A ← A + I，如果产生进位则C=1，否则C=0。	√	√	√	1
	SBC	A,M	A ← A - M - /C，如果产生借位则C=0，否则C=1。	√	√	√	1
	SBC	M,A	M ← A - M - /C，如果产生借位则C=0，否则C=1。	√	√	√	1+N
	SUB	A,M	A ← A - M，如果产生借位则C=0，否则C=1。	√	√	√	1
	SUB	M,A	M ← A - M，如果产生借位则C=0，否则C=1。	√	√	√	1+N
	SUB	A,I	A ← A - I，如果产生借位则C=0，否则C=1。	√	√	√	1
	MUL A，M		R，A←A*M，乘积低字节存放到ACC，高字节存放在系统寄存器R中，ZF标志位受ACC内容影响	-	-	√	2

LOGIC	AND	A,M	A ← A与M。	-	-	√	1
	AND	M,A	M ← A与M。	-	-	√	1+N
	AND	A,I	A ← A与I。	-	-	√	1
	OR	A,M	A ← A或M。	-	-	√	1
	OR	M,A	M ← A或M。	-	-	√	1+N
	OR	A,I	A ← A或I。	-	-	√	1
	XOR	A,M	A ← A异或M。	-	-	√	1
	XOR	M,A	M ← A异或M。	-	-	√	1+N
	XOR	A,I	A ← A 异或I。	-	-	√	1
PROCESS	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)。	-	-	-	1
	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)。	-	-	-	1+N
	RRC	M	A ← M带进位右移。	√	-	-	1
	RRCM	M	M ← M带进位右移。	√	-	-	1+N
	RLC	M	A ← M带进位左移。	√	-	-	1
	RLCM	M	M ← M带进位左移。	√	-	-	1+N
	CLR	M	M ← 0。	-	-	-	1
	BCLR	M.b	M.b ← 0。	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0。	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1。	-	-	-	1+N
BRANCH	CMPRS	A,I	比较, 如果相等则跳过下一条指令C与ZF标志位会受影响。	√	-	√	1 + S
	CMPRS	A,M	比较, 如果相等则跳过下一条指令C与ZF标志位会受影响。	√	-	√	1 + S
	INCS	M	A ← M + 1，如果A = 0，则跳过下一条指令。	-	-	-	1+ S
	INCMS	M	M ← M + 1，如果M = 0，则跳过下一条指令。	-	-	-	1+N+S
	DECS	M	A ←M - 1，如果A = 0，则跳过下一条指令。	-	-	-	1+ S
	DECMS	M	M ← M - 1，如果M = 0，则跳过下一条指令。	-	-	-	1+N+S
	BTS0	M.b	如果M.b = 0，则跳过下一条指令。	-	-	-	1 + S
	BTS1	M.b	如果M.b = 1，则跳过下一条指令。	-	-	-	1 + S
	B0BTS0	M.b	如果M(bank 0).b = 0，则跳过下一条指令。	-	-	-	1 + S
	B0BTS1	M.b	如果M(bank 0).b = 1，则跳过下一条指	-	-	-	1 + S

			令。				
	JMP	d	跳转指令， ， 。	-	-	-	2
	CALL	d	子程序调用指令， ， PC15/14 ，PC13~PC0 。	-	-	-	2
MISC	RET		子程序跳出指令， 。	-	-	-	2
	RETI		中断处理程序跳出指令， ， 使能全局中断控制位。	-	-	-	2
	PUCH		进栈指令，保存ACC和PFLAG(除 NT0， NPD位)。	-	-	-	1
	POR		出栈指令，恢复ACC和PFLAG(除 NT0， NPD位)。	√	√	√	1
	NOP		空指令，无特别意义。	-	-	-	1

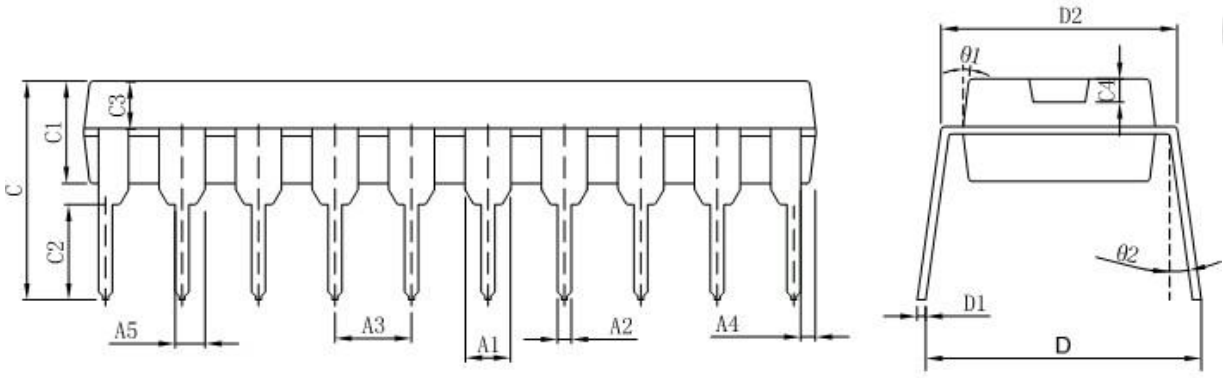
注：1.“M”是系统寄存器或RAM，若是系统寄存器，则N = 0，否则N = 1。

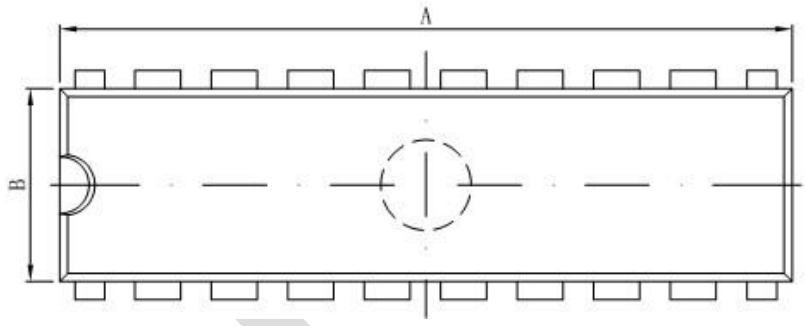
2. 若满足跳转条件，S = 1，否则S = 0

15、外形图

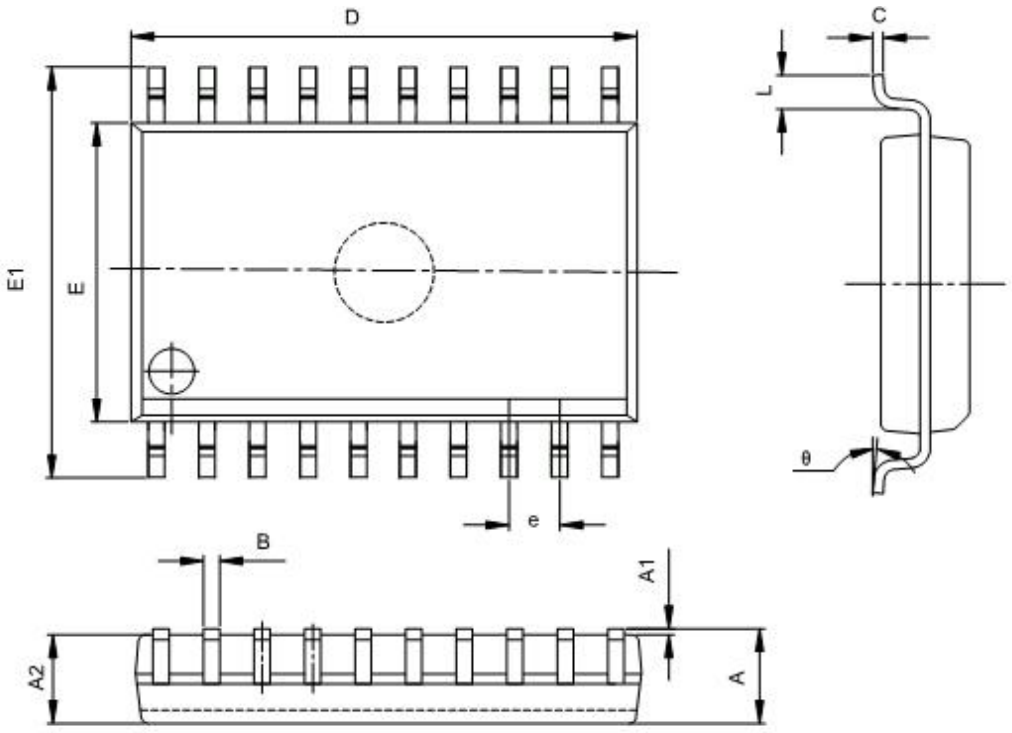
15.1 、DIP20 外形图与封装尺寸

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			



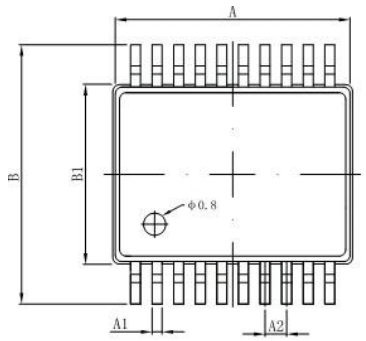


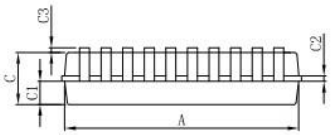
15.2 、SOP20 外形图与封装尺寸

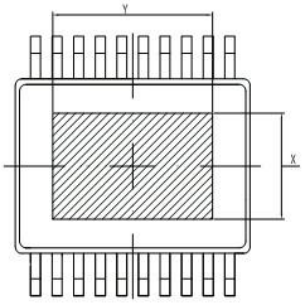


Symbol	Dimensions In Millimeters		Dimensions In Inches	
	Min	Max	Min	Max
A	2.280	2.630	0.090	0.104
A1	0.100	0.300	0.004	0.012
A2	2.180	2.330	0.086	0.092
B	0.350	0.510	0.014	0.020
C	0.204	0.360	0.008	0.014
D	12.600	13.000	0.496	0.512
E	7.400	7.600	0.291	0.299
E1	10.000	10.650	0.394	0.419
e	1.270(TYP)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

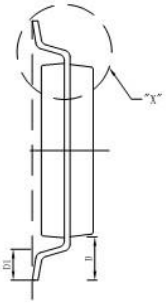
15.3 、TSSOP20 外形图与封装尺寸

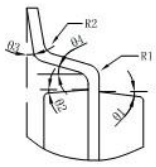






BOTTOM VIEW





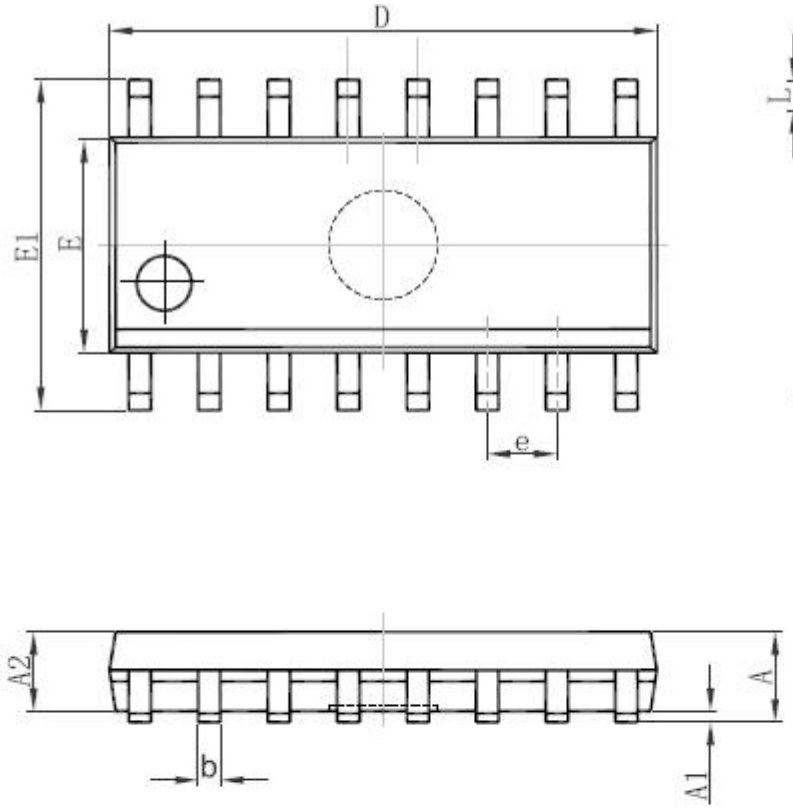
DETAIL "X"

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	

15.4 、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°

16、声明及注意事项：

16.1 、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）
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。					

16.2 、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