YF2501
I/O 型 8 位单片机

产品说明书

说明书发行履历：

版本	发行时间	新制/修订内容
2016-03-A1	2016-03	新制

1、概 述

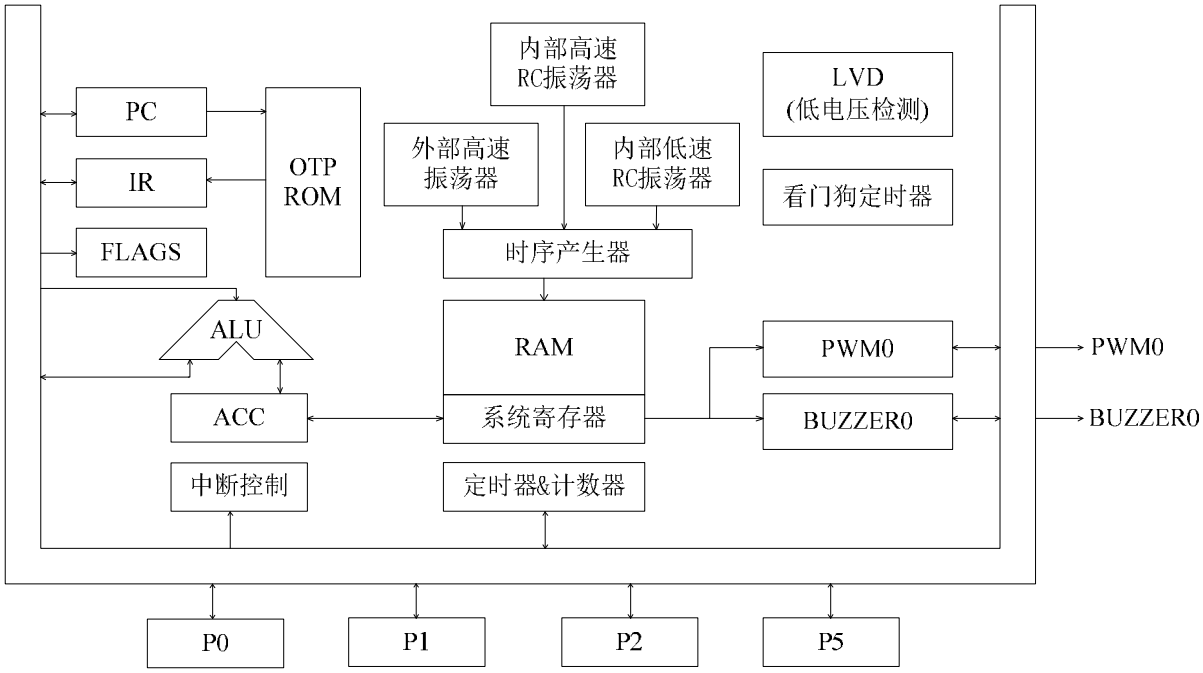
YF2501是一款OTP ROM的I/O型8位微控制器，其具有良好的内部高速RC振荡，高抗干扰性能可更好的应用于小家电领域。

功能特性：

- 存储器：
  - ROM：OTP、1K*16位
  - RAM：48字节
- I/O引脚配置
  - 输入输出双向端口：P0、P1、P2、P5
  - 单向输入引脚：P1.1
  - 可编程的漏极开路引脚：P1.0
  - 具有唤醒功能端口：P0、P1电平变化触发
  - 内置上拉电阻端口：P0、P1、P2、P5
  - 外部中断引脚：P0.0 由寄存器PEDGE控制，其触发方式为上升沿或下降沿
- 3级低电压检测系统(LVD)
  - 可监控系统电源(若低于设定LVD则复位)
- 强大的指令系统
  - 单周期指令系统（1T）
  - 大部分指令仅需一个周期
  - 跳转指令JMP可在整个ROM区执行
  - 子程序条用指令CALL可在整个ROM区执行
  - 查表指令MOVC可寻址整个ROM区
- 工作电压：
  - VDD: 2.4V~5.5V
- 双时钟系统：
  - 外部高速时钟：RC模式高达10MHz
  - 外部高速时钟：晶振模式高达16MHz
  - 内部高速模式：16MHz RC（Fcpu=Fosc/4~Fosc/16）
  - 内部低速模式：RC振荡器，16KHz(3V),32KHz(5V)
- 3个中断源：
  - 2个内部中断源：T0、TC0
  - 1个外部中断源：INT0
- 两个8位定时/计数器：
  - TC0：自动装载定时器/计数器/蜂鸣器(BUZZER)输出
  - T0：基本定时/计数器，具有0.5sec实时时钟功能（RTC）
- 内置看门狗定时器，其时钟源由内部低速RC振荡提供（16KHz@3V，32KHz@5V）
- 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：有T0周期性的唤醒
- 封装形式
  - SOP8
  - SOP14/DIP14

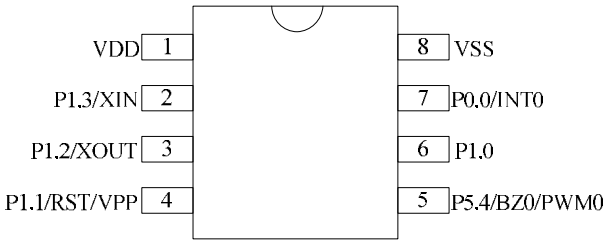
2、功能框图及引脚说明

2.1、功能框图

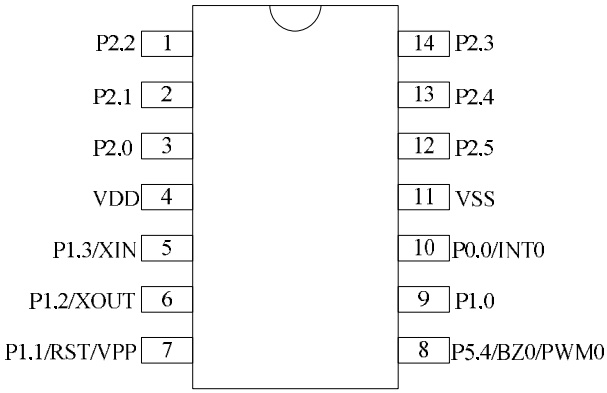


2.2、引脚排列图

SOP8/DIP8



SOP14/DIP14



2.3、引脚说明

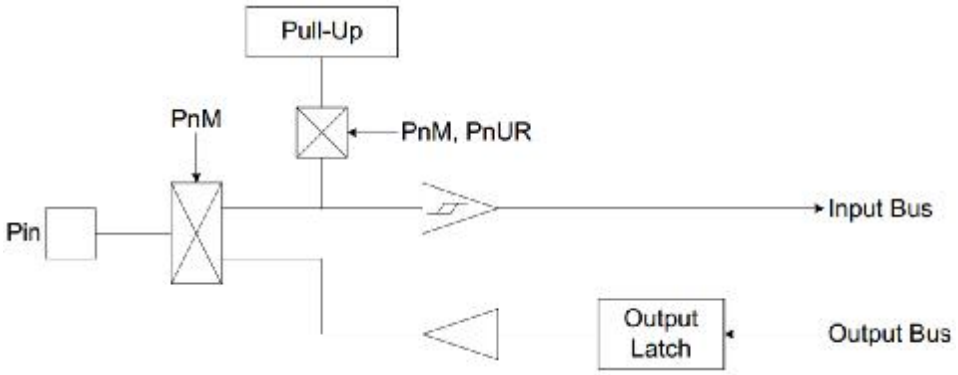
符　号	属性	功　能
VDD,VSS	P	电源输入端。

P1.1/RST/VPP	I,P	P1.1: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。作普通 I/O 口使用时,用户需在 P1.1 外面串接一个 100 欧姆的电阻，具有唤醒功能。 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: OTP 烧录引脚。
P1.3/XIN	I/0	P1.3: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,具有唤醒功能。 XIN:使能外部振荡器(晶振或 RC)时为振荡信号输入引脚。
P1.2/XOUT	I/O	P1.2: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:使能外部晶振模式时为振荡信号输出引脚。
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。
P1.0	I/0	双向输入/输出引脚,漏极开路引脚,输入模式时为施密特触发,内置上拉电阻。
P2[5:0]	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/BZ0/PWM0	I/O	P5.4: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。Buzzer 输出引脚 / PWM 输出引脚。

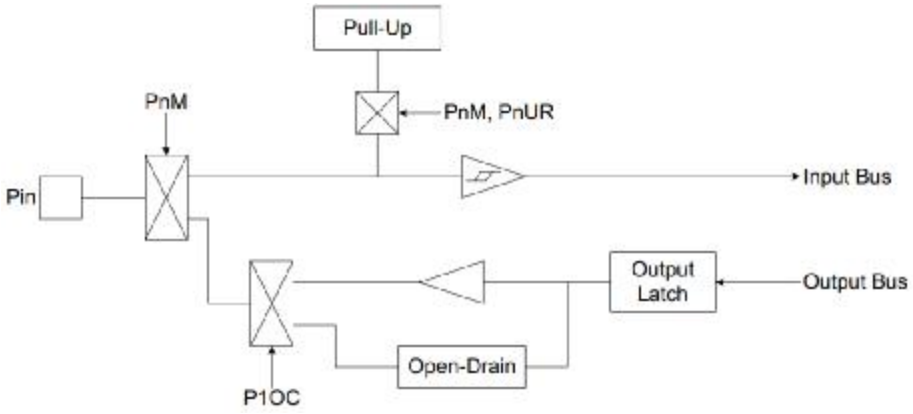
2.4、引脚电路结构图

Ø P0、2、5 端口结构

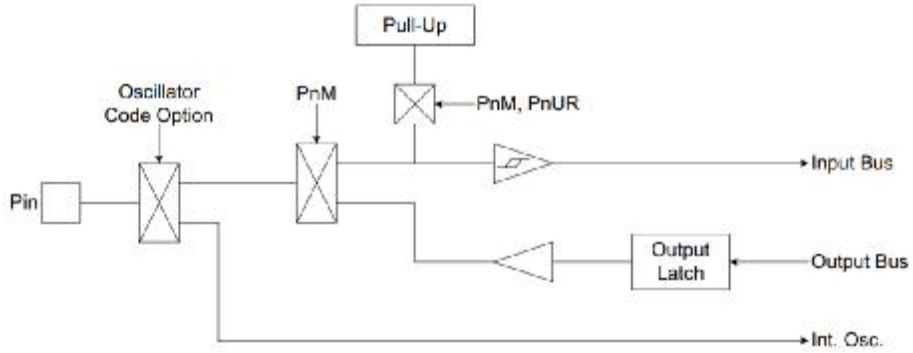
Ø



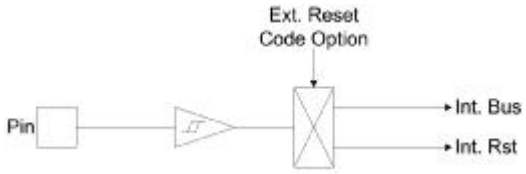
Ø P1.0 端口结构



Ø P1.2、P1.3 端口结构



Ø P1.1 端口结构



3、电特性

3.1、 极限参数

除非另有规定，T_{amb}=25℃

参 数 名 称	符 号	条 件		额 定 值	单 位
电源电压	V_{DD}			-0.3~6.0	V
输入电压	V_{IN}			VSS-0.2V~VDD+0.2V	mA
工作环境温度	T_{amb}			-40~+85	℃
贮存温度	T_{stg}			-40~+125	℃
焊接温度	T_L	10 秒	DIP	245	℃
			SOP	250	

3.2、电气特性

测试条件：V_{DD}=5V，V_{SS}=0V ,T_{amb}=25℃,fosc=4MHz，Fcpu=1MHz，除非另有规定。

参 数 名 称	符 号	测 试 条 件	最小	典型	最大	单 位
工作电压	V_{DD}	普通模式：V_{PP}=V_{DD}	2.4	5.0	5.5	V
RAM 数据保留电压	V_{DR}		1.5			V
VDD 上升率	V_{POR}	V_{DD} 上升至上电复位	0.05			V/ms
输入低电平电压	V_{IL1}	所有输入端口	V_{SS}		0.3V_{DD}	V
	V_{IL2}	复位端口	V_{SS}		0.2V_{DD}	V
输入高电平电压	V_{IH1}	所有输入端口	0.7V_{DD}		V_{DD}	V
	V_{IH2}	复位端口	0.9V_{DD}		V_{DD}	V
RST 口漏电流	I_{LEKG}	V_{IN}=V_{DD}, 25℃			2	uA
		V_{IN}=V_{DD}, -40℃~+85℃			5	uA
I/O 口上拉电阻	R_{UP}	V_{IN}=V_{SS},V_{DD}=3V	100	200	300	KΩ
		V_{IN}=V_{SS},V_{DD}=5V	50	100	180	KΩ
I/O 口输入漏电流	I_{LEKG}	禁用上拉电阻，V_{IN}=V_{DD}			2	uA
I/O 输出拉电流	I_{OH}	V_{OP}=V_{DD}-0.5V	8	12		mA
I/O 输出灌电流	I_{OL}	V_{OP}=V_{SS}+0.5V	8	15		mA
INT0 触发脉冲宽度	T_{INT0}	INT0 中断触发脉冲宽度	2/fcpu			
工作电流	I_{DD1}	普通模式：V_{DD}=5V 4MHz ， Fcpu=Fosc/4		2.5	5	mA
		普通模式：V_{DD}=3V 4MHz ， Fcpu=Fosc/4		1	2	mA
	I_{DD2}	低速模式(内部低速 RC)：V_{DD}=5V，ILRC 32KHz		20	40	uA
		低速模式(内部低速 RC)：V_{DD}=3V，ILRC 32KHz		5	10	uA
	I_{DD3}	睡眠模式：V_{DD}=5V, 25℃		0.8	1.6	uA
		睡眠模式：V_{DD}=3V, 25℃		0.7	1.4	uA
		睡眠模式：V_{DD}=5V, -40℃~+85℃		10	21	uA
		睡眠模式：V_{DD}=3V, -40℃~+85℃		10	21	uA
	I_{DD4}	绿色模式(无负载，Fcpu=Fosc/4，关闭看门狗)：V_{DD}=5V,4MHz		0.6	1.2	mA

		绿色模式(无负载，Fcpu=Fosc/4，关闭看门狗)：V_{DD}=3V,4MHz		0.25	0.5	mA
		绿色模式(无负载，Fcpu=Fosc/4，关闭看门狗)：V_{DD}=5V,ILRC 32KHz		15	30	uA
		绿色模式(无负载，Fcpu=Fosc/4，关闭看门狗)：V_{DD}=3V, ILRC 32KHz		3	6	uA
内部高速振荡频率	F_{IHRC}	VDD=5V,25℃,Fcpu=1MHz	15.68	16	16.32	MHz
		VDD=2.4V~5.5V, -40℃~+85℃, Fcpu=1MHz~4MHz	14	16	18	
LVD 电压	V_{DET0}	低压复位	1.6	2.0	2.3	V
	V_{DET1}	低压复位，Fcpu=1MHz，低压检测，Fcpu=1MHz。	1.8	2.4	3	V
	V_{DET2}	低压检测，Fcpu=1MHz。	2.5	3.6	4.5	V

注：上表数据用于设计参考，并非实测。

4、CPU 说明

4. 1、程序存储器（ROM）

- YF2501 的 ROM 空间为 1K

YF2501 的程序存储器(OTP ROM)包括复位向量、中断向量、通用存储器区域和系统保留区域。复位向量为程序的开始地址，中断向量为中断服务程序的开始地址。通用存储器区域存储主程序，包括主循环、子程序和数据表。

4. 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。下面例程演示了如何定义 ROM 中的复位向量。

Ø 例：定义复位向量

```
ORG     0       ;
JMP     SRART   ;跳转用户程序
…
ORG     10H     ;
```

```
START:                          ;用户程序起始地址
        …                       ;用户程序
        …                       ;
        ENDP                    ;程序结束
```

4. 1.2、中断向量(0008H)

中断向量地址为 0008H。一但有中断相应，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。下面的例程说明了如何编写中断服务程序。

Ø 例：定义中断向量，中断服务程序金穗 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
```

Ø 例：定义中断向量，中断程序在用户程序之后。

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

START:                      ;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                ;中断结束
                …
                ENDP                ;程序结束
```

注：从上面的程序中容易得到 YF2501 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；

4.1.3、查表

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。

Ø 例：查找 ROM 地址为“TABLE1”的值。

```
                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
                                            ;查找下一个地址
                INCMS       Z               ;
                JMP         @F              ;Z 没有溢出
                INCMS       Y               ;Z 溢出(FFH à 00), à 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。

Ø 例：宏 INC_YZ

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

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

Ø 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。

Ø 例：由指令 B0ADD/ADD 对 Y 和 Z 寄存器加 1。

```
                B0MOV       Y,#TABLE$M      ;设置 TABLE1 地址中间字节
                B0MOV       Z,#TABLE$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                ;存数数据，如果 BUF=0，数据未 0035H
                                    ；如果 BUF=1，数据=5105H
                                    ；如果 BUF=2，数据=5105H
                …
TABLE1:         DW      0035H       ；定义数据表(16 位)数据
                DW      5105H
                DW      2012H
                …
```

4.1.4、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，可以通对 PCL 加上不同的的 ACC 值来实现地址跳转。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，可参考一下范例。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。

Ø 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+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
```

注：如果跳转表跨越 ROM 边界，将引起程序错误。

4.1.5、CHECKSUM 计算

ROM 的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。

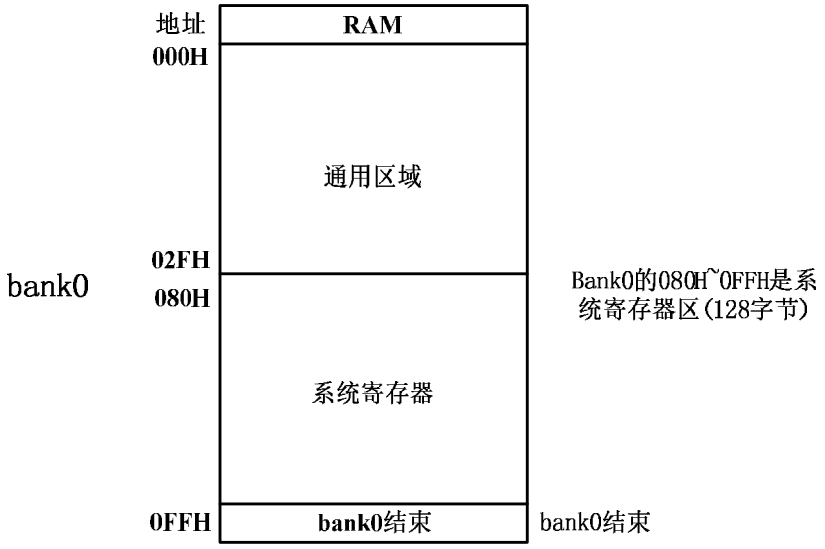
Ø 例：示例程序演示了如何对 **00H** 到用户程序结束进行 **Checksum** 计算。

```
                MOV     A,#END_USER_CODE$L
                B0MOV   END_ADDR1, A    ;用户程序终端地址低位字节存入 end_addr1
                MOV     A,#END_USER_CODE$M
                B0MOV   END_ADDR2, A    ;用户程序终端地址中间字节存入 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              ;如果 Z != 00H，进行下一个计算。
                JMP       Y_ADD_1         ;如果 Z = 00H， Y 加 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              ;转至 checksum 计算。
CHECKSUM_END:
                …
                …
END_USER_CODE:
```

4.2、数据存储器(RAM)

- RAM:48 字节



48 字节的通用存储区域位于 Bank0，控制器提供 bank0 型指令(如 B0MOV， B0ADD，B0BTS1 ， B0BSET……)直接控制 Bank0 RAM。

4.2.1、系统寄存器

4.2.1.1、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	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	P2M	-	-	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	T0R	PCL	PCH
D	P0	P1	P2	-	-	P5	-	-	T0M	T0C	TC0M	TC0C	-	-	-	STKP
E	P0UR	P1UR	-	-	-	P5UR	-	@YZ	-	P1OC	-	-	-	-	-	-
F	-	-	-	-	-	-	-	-	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

4.2.1.2、系统寄存器说明

PFLAG	=特殊标志寄存器	R	= 工作寄存器和 ROM 查表数据缓存器
P1W	= P1 唤醒功能寄存器	Y, Z	= 专用寄存器， @YZ 间接寻址寄存器， ROM 寻址寄存器
PEDGE	= P0.0 触发方向寄存器	@YZ	= 间接寻址寄存器
PnM	= Pn 模式控制寄存器	Pn	= Pn 数据缓存器
P1OC	= P1 漏极开路控制寄存器	PnUR	= Pn 上拉电阻控制寄存器
INTRQ	= 中断请求寄存器	INTEN	= 中断使能寄存器
OSCM	= 振荡器模式寄存器	PCH, PCL	= 程序计数器
T0M	= T0 模式寄存器	T0C	= T0 计数寄存器
TC0M	= TC0 模式寄存器	TC0C	= TC0 计数寄存器
TC0R	= TC0 自动装载数据缓存器	WDTR	= 看门狗定时器清零寄存器
STKP	= 堆栈指针	STK0~STK3	= 堆栈缓存器

4.2.1.3、系统寄存器说明

地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	备注
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
0B8H	-	-	-	-	-	-	-	P00M	R/W	P0M
0BFH	-	-	-	P00G1	P00G0	-	-	-	R/W	PEDGE
0C0H	-	-	-	-	P13W	P12W	P11W	P10W	W	P1W 唤醒寄存器
0C1H	-	-	-	P14M	P13M	P12M	-	P10M	R/W	P1M I/O
0C2H	-	-	P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M I/O
0C5H	-	-	-	P54M	-	-	-	-	R/W	P5M I/O
0C8H	-	-	TC0IRQ	T0IRQ	-	-	-	P00IRQ	R/W	INTRQ
0C9H	-	-	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	-	-	-	-	-	-	PC9	PC8	R/W	PCH
0D0H	-	-	-	-	-	-	-	P00	R/W	P0 数据缓存器
0D1H	-	-	-	-	P13	P12	P11	P10	R/W	P1 数据缓存器
0D2H	-	-	P25	P24	P23	P22	P21	P20	R/W	P2 数据缓存器
0D5H	-	-	-	P54	-	-	-	-	R/W	P5 数据缓存器
0D8H	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	T0TB	R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT	R/W	TC0M
0DBH	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0	R/W	TC0C
0DFH	GIE	-	-	-	-	-	STKPB1	STKPB0	R/W	STKP 堆栈指针
0E0H	-	-	-	-	-	-	-	P00R	W	P0 上拉电阻
0E1H	-	-	-	-	P13R	P12R	-	P10R	W	P1 上拉电阻
0E2H	-	-	P25R	P24R	P23R	P22R	P21R	P20R	W	P2 上拉电阻
0E5H	-	-	-	P54R	-	-	-	-	W	P5 上拉电阻
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E9H	-	-	-	-	-	-	-	P10OC	W	P1OC 漏极开路
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	-	-	-	-	-	-	S1PC9	S1PC8	R/W	STK1H
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编译器中做了宣告；
2. 在编译器中，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。

4.2.1.4、系统寄存器说明

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。

Ø 例：读/写 ACC。

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

Ø 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。
```

4.2.1.5、程序状态寄存器 PFLAG

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

086H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
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

Bit5 **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。

4.2.1.6、程序状态寄存器 PFLAG

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。

	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
复位后	-	-	-	-	-	-	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。如果位测试为真，PC加2。

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

如果ACC等于指定的立即数则PC值加2，跳过下一条指令。

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

执行加1指令后,结果为零时,PC的值加2,跳过下一条指令。

INCS：

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

INCMS：

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

执行减1指令后,结果为零时,PC的值加2,跳过下一条指令。

DECS：

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

DECMS：

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

- 多地址跳转

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

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

Ø 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。
                …
```

Ø 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。
                …
                …
```

4.2.1.7、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	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
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	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
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

Ø 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。
```

Ø 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:
```

4.2.1.8、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	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
Z	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

4. 3、寻址模式

4. 3.1、立即寻址

将立即数送入ACC或指定的RAM单元

Ø 例：立即数12H 送入ACC。

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

Ø 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. 3.2、直接寻址

通过 ACC 对 RAM 单元数据进行操作。

Ø 例：地址 12H 处的内容送入 ACC。

```
B0MOV       A, 12H
```

Ø 例：ACC 中数据写入 RAM 的 12H 单元。

```
B0MOV       12H, A
```

4. 3.3、间接寻址

通过指针寄存器（ Y/Z）访问 RAM 数据。

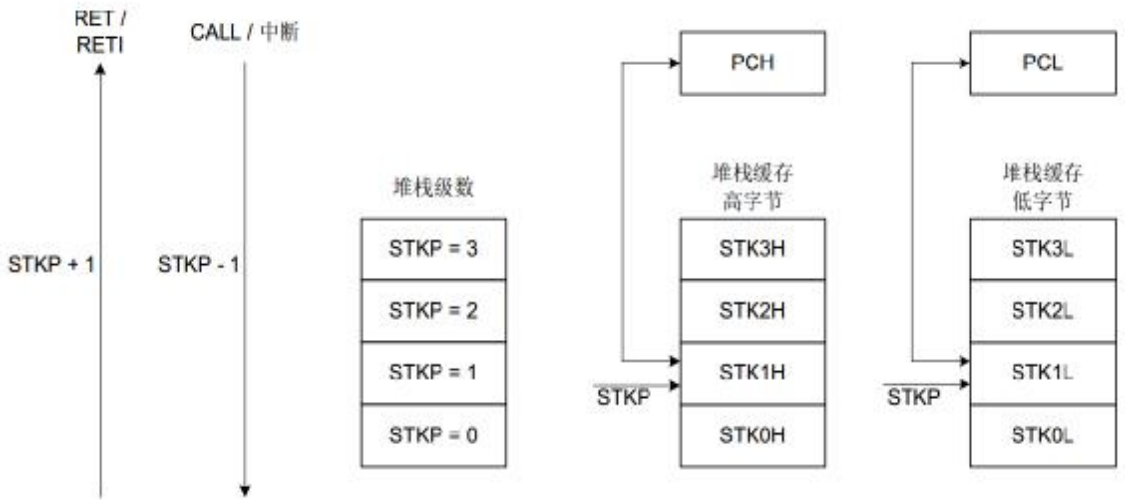
Ø 例：用 @YZ 实现间接寻址。

```
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. 4、堆栈

4. 4.1、概述

YF2501的堆栈缓存器共有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，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。



4. 4.2、堆栈寄存器

堆栈指针 STKP 是一个 2 位寄存器，存放被访问的堆栈单元地址，9 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	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STKP	GIE	-	-	-	-	-	STKPB1	STKPB0
读/写	R/W	-	-	-	-	-	R/W	R/W
复位后	0	-	-	-	-	-	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。

Ø 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，如下图所示：

```
MOV     A, #00000011B
B0MOV   STKP, A
```

0F0~0F8H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

STKnH	-	-	-	-	-	-	SnPC9	SnPC8
读/写	-	-	-	-	-	-	R/W	R/W
复位后	-	-	-	-	-	-	0	0

0F0~0F8H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
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=3~0)

4. 4.3、堆栈操作举例

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

堆栈层数	STKP		堆栈缓存器		备注
	STKPB1	STKPB0	高字节	低字节	
0	1	1	Free	Free	-
1	1	0	STK0H	STK0L	-
2	0	1	STK1H	STK1L	-
3	0	0	STK2H	STK2L	-
4	1	1	STK3H	STK3L	-
>4	1	0	-	-	溢出出错

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

堆栈层数	STKP		堆栈缓存器		备注
	STKPB1	STKPB0	高字节	低字节	
4	1	1	STK3H	STK3L	-
3	0	0	STK2H	STK2L	-
2	0	1	STK1H	STK1L	-
1	1	0	STK0H	STK0L	-
0	1	1	-	-	-

4. 5、编译选项表(CODE OPTION)

编译选项（CODE OPTION）是一种系统的硬件配置，包括系统时钟 rate，看门狗定时器的操作，LVD选项，复位引脚选项以及OTP ROM的安全控制。如下表所示：

编译选项	配置项目	功能说明
Noise_Filter	Enable	开启杂讯滤波功能，Fcpu = Fosc/4~Fosc/16。
	Disable	关闭杂讯滤波功能，Fcpu = Fosc/1~Fosc/16。
High_Clk	IHRC_16M	高速时钟采用内部 16MHz RC 振荡电路，XIN/XOUT（P1.3/P1.2）作为普通的 I/O 引脚； IHRC_16M 和 IHRC_RTC 模式下，不支持 Fosc/1，Fosc/2。
	IHRC_RTC	高速时钟采用内部 16MHz RC 振荡电路，具有 RTC 功能（0.5sec），XIN/XOUT（P1.3/P1.2）作为外部 32K 晶体输入引脚； IHRC_16M 和 IHRC_RTC 模式下，不支持 Fosc/1，Fosc/2。
	RC	外部高速时钟振荡器采用廉价的 RC 振荡电路，XOUT（P1.2）为普通的 I/O 引脚。
	32K X'tal	外部高速时钟振荡器采用低频、省电晶体/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 10MHz~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 1M~10MHz）。
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/1	指令周期 = 1 个时钟周期，必须关闭杂讯滤波功能。
	Fhosc/2	指令周期 = 2 个时钟周期，必须关闭杂讯滤波功能。
	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。
	P11	P1.1 为单向输入引脚，无上拉电阻。
Security	Enable	ROM 代码加密。
	Disable	ROM 代码不加密。
LVD	LVD_L	VDD 低于 2.0V 时，LVD 复位系统。
	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。

4. 5. 1、Fcpu 编译选项

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

4. 5. 2、Reset_Pin 编译选项

复位引脚与单向输入引脚共用，由编译选项控制。

- Reset：使能外部复位引脚功能。当下降沿触发时，系统复位。
- P11： 使能 P1.1 为单向输入引脚。此时禁止外部复位引脚功能。

4. 5. 3、Security 编译选项

Security 编译选项是对OTP ROM的一种保护，当使能Secuity编译选项，ROM代码加密，可以保护 ROM 的内容。

4. 5. 4、Noise Filter 编译选项

Noise Filter编译选项是强杂讯滤除功能以减少杂讯对系统时钟的影响。如使能杂讯滤波器，在高干扰环境下，使能看门狗定时器且选择一个合适的LVD选项可以使整个系统更好的工作。

5、复位

5. 1、概述

YF2501 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)；

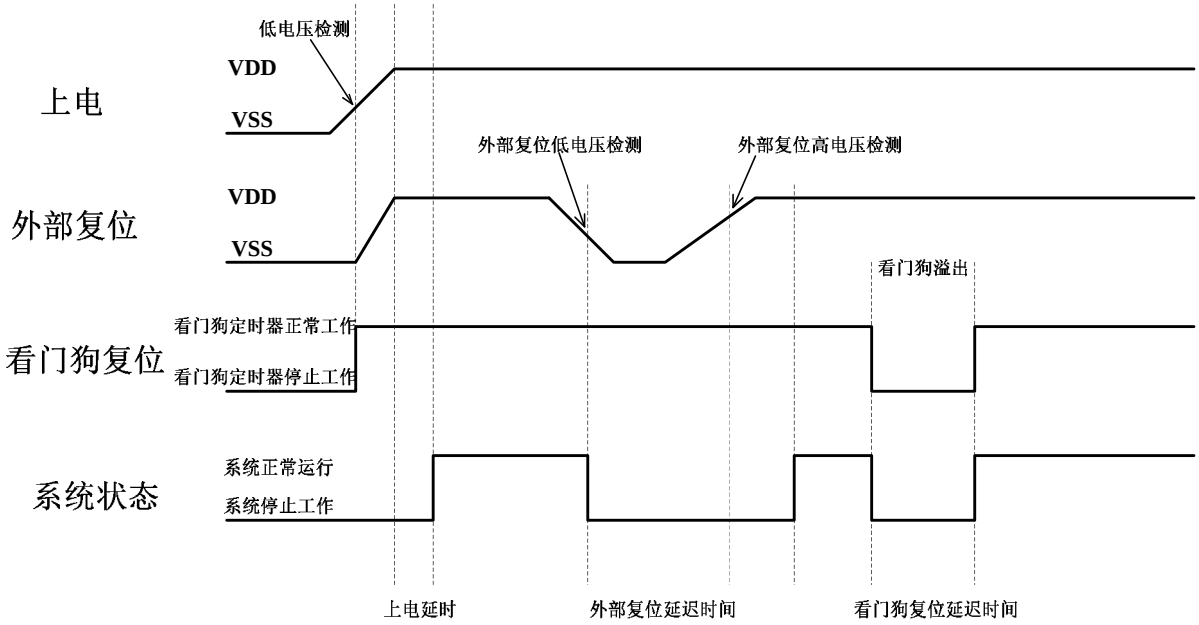
上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器PC清零。复位结束后，系统从向量0000H处重新开始运行。PFLAG寄存器的NT0和NPD两个标志位能够给出系统复位状态的信息。用户可以编程控制NT0和NPD，从而控制系统的运行路径。

086H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
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。



5. 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. 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：

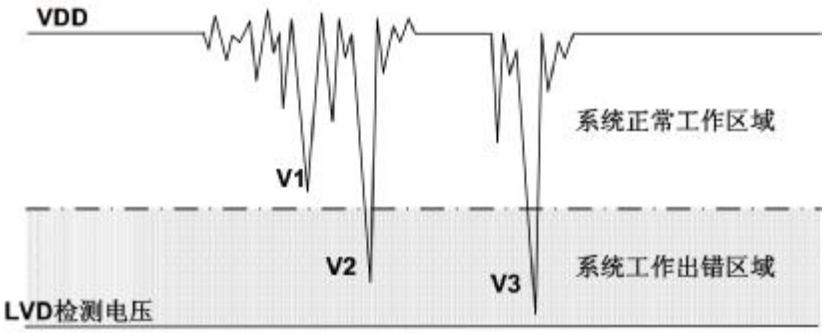
- 对看门狗清零之前，检查 I/O 口的状态和 RAM 的内容可增强程序的可靠性；

- 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. 4、掉电复位

5. 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：

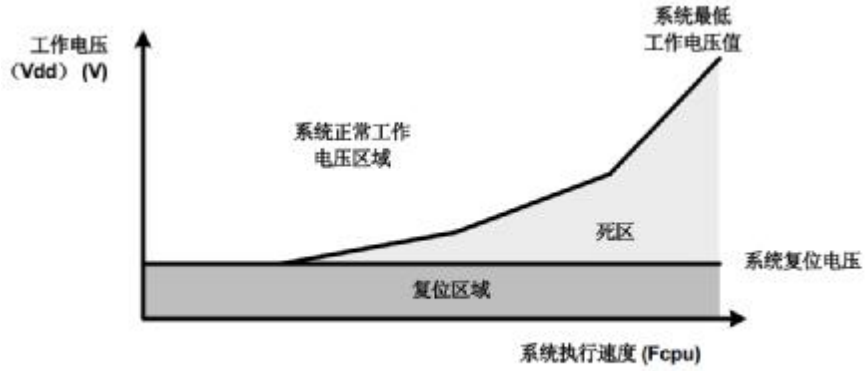
DC 运用中：DC 运用中一般都采用电池供电，当电池电压过低或单片机驱动负载时，系统电压可能跌落并进入死区。这时，电源不会进一步下降到 LVD 检测电压，因此系统维持在死区。

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。

5. 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

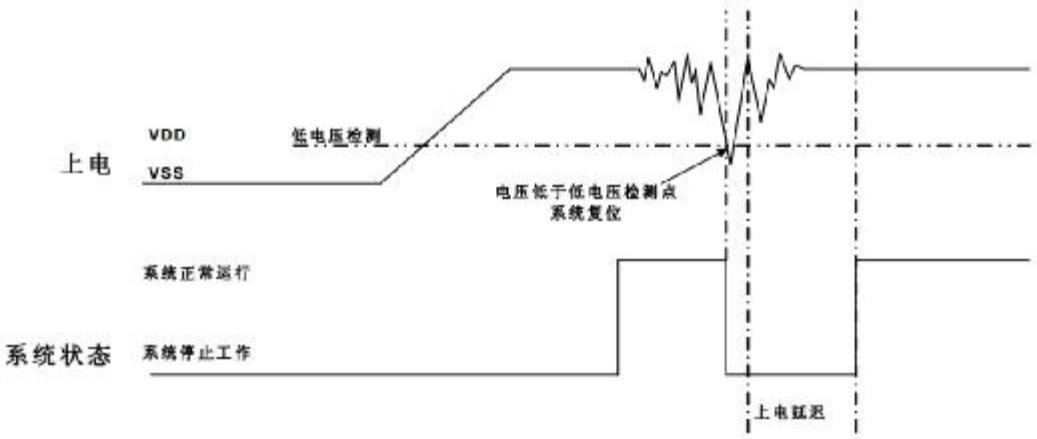
如上图所示，系统正常工作电压区域一般高于系统复位电压，同时复位电压由低电压检测（ LVD）电平决定。当系统执行速度提高时，系统最低工作电压也相应提高，但由于系统复位电压是固定的，因此在系统最低工作电压与系统复位电压之间就会出现一个电压区域，系统不能正常工作，也不会复位，这个区域即为死区。

5. 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）。

LVD 复位：



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

LVD 设计为三层结构（2.0V/2.4V/3.6V），由LVD编译选项控制。对于上电复位和掉电复位，2.0V

LVD始终处于使能状态；2.4V LVD具有LVD复位功能，并能通过标志位显示VDD状态；3.6V LVD 具有标记功能，可显示VDD的工作状态。LVD标志功能只是一个低电压检测装置，标志位LVD24和LVD36给出VDD的电压情况。对于低电压检测应用，只需查看LVD24和LVD36的状态即可检测电池状况。

086H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
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		
	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：(1)、 LVD 复位结束后，LVD24 和 LVD36 都将被清零；

(2)、 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。

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。

5. 5、外部复位

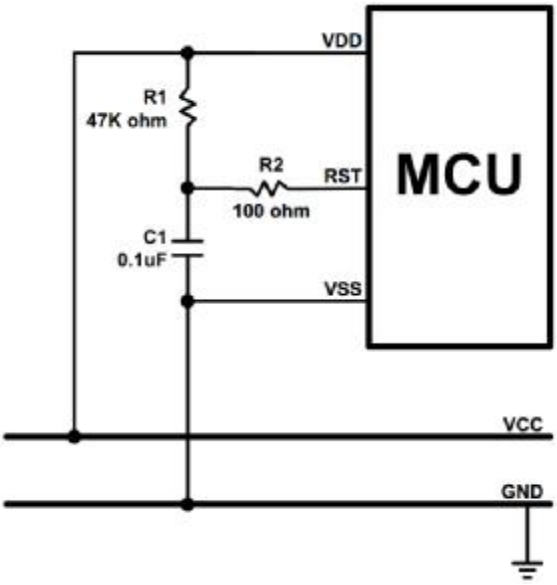
外部复位功能由编译选项“Reset_Pin”控制。将该编译选项置为“Reset”，可使能外部复位功能。外部复位引脚为施密特触发结构，低电平有效。复位引脚处于高电平时，系统正常运行。当复位引脚输入低电平信号时，系统复位。外部复位操作在上电和正常工作模式时有效。需要注意的是，在系统上电完成后，外部复位引脚必须输入高电平，否则系统将一直保持在复位状态。外部复位的时序如下：

- **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。

5.6、外部复位电路

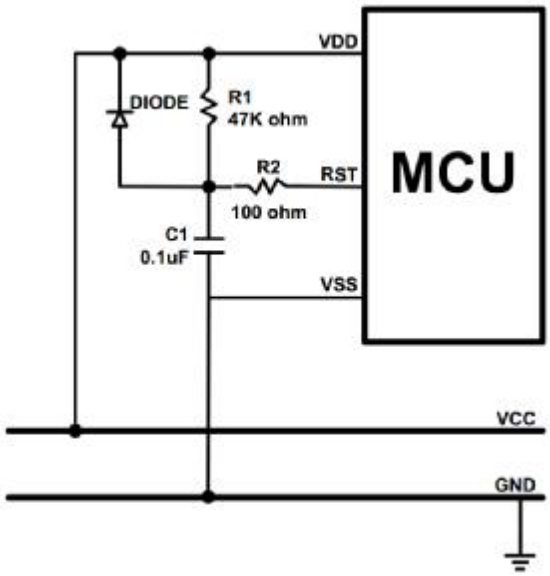
5.6.1、RC 复位电路



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。

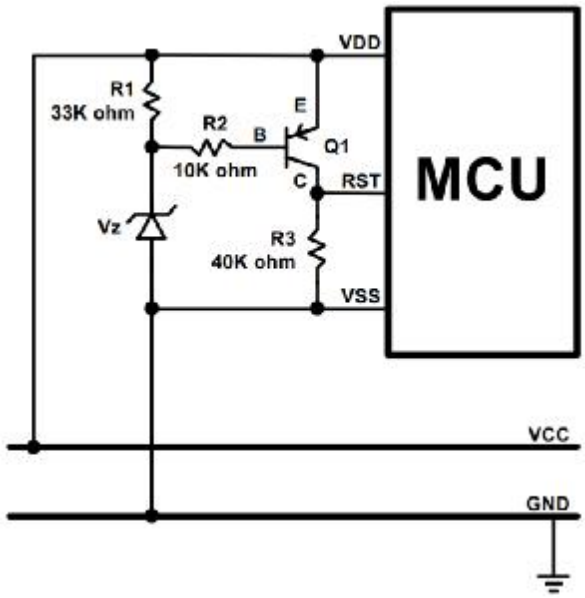
5.6.2、二极管及 RC 复位电路



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

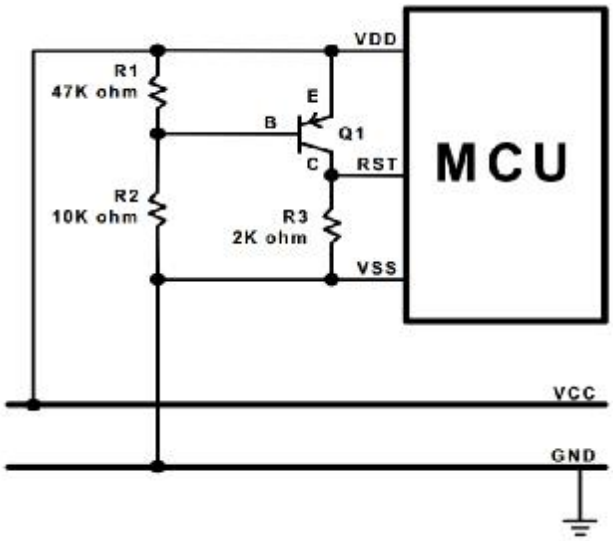
注：“基本 RC 复位电路”和“二极管及 RC 复位电路”中的电阻 R2 都是必不可少的限流电阻，以避免复位引脚 ESD（Electrostatic Discharge）或 EOS（Electrical Over-stress）击穿。

5.6.3、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。

5.6.4、电压偏置复位电路

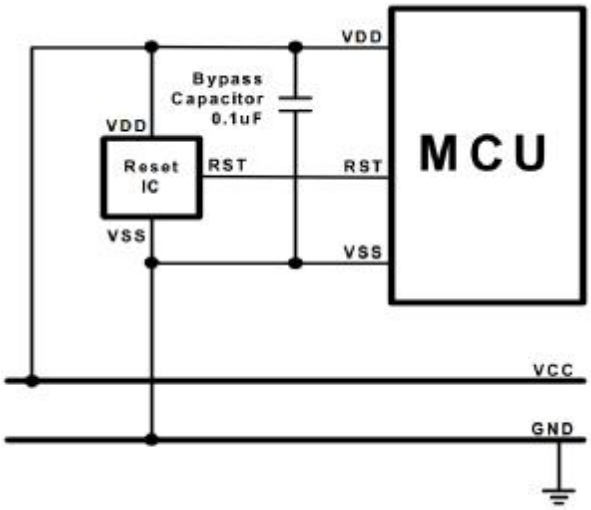


电压偏置复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，这种复位电路的检测电压值的精确度有所降低。 电路中， R1 和 R2 构成分压电路，当 VDD 高于和等于分压值“0.7V x (R1 + R2)/ R1 ”时，三极管集电极 C 输出高电平，单片机正常工作； VDD 低于“0.7V x (R1 + R2) / R1 ”时，集电极 C 输出低电平，单片机复位。

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。

5.6.5、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。

6、系统时钟

6. 1、概述

YF2501内带双时钟系统：高速时钟和低速时钟。高速时钟的时钟源由外部晶振或内部 16MHz RC 振荡电路（ IHRC 16MHz）提供， 由编译选项 High_CLK选择控制。低速时钟的时钟源则由内部低速 RC 振荡电路（ ILRC 16KHz@3V）提供，由 OSCM 寄存器的 CLKMD 位控制。两种时钟都可作为系统时钟源 Fosc，系统在低速模式下工作时， Fosc 4 分频后作为一个指令周期。

- 高速振荡器

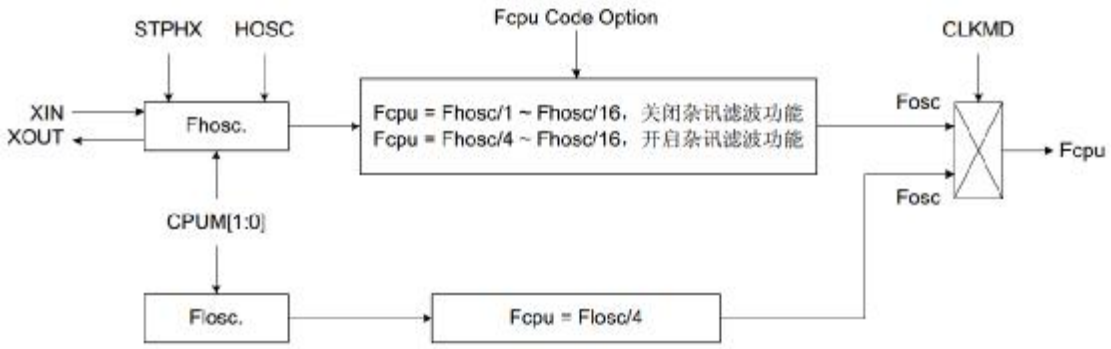
  内部高速振荡器： RC，高达 16MHz，称为 IHRC；

  外部高速振荡器：晶体/陶瓷（ 4MHz， 12MHz 和 32KHz）和 RC。

- 低速振荡器

  内部低速振荡器： RC， 16KHz@3V， 32KHz@5V，称为 ILRC。

- 系统时钟框图



- HOSC： High_Clk 编译选项。
- Fhosc：外部高速时钟频率。
- 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。

在干扰较严重的运用条件下， SONiX 提供的杂讯滤波器能够对外部干扰进行隔离以保护系统的正常工作。但使能杂讯滤波器时，高速时钟的 Fcpu 被限制为 Fcpu＝Fhosc/4。

6. 2、指令周期 Fcpu

系统时钟速率，即指令周期（ Fcpu），从系统时钟源分离出来，决定系统的工作速率。Fcpu 的速率由 Fcpu 编译选项决定，正常模式下，Fcpu=Fhosc/1~Fhosc/16。当系统高速时钟源为外部 4MHz 晶体时， Fcpu 编译选项选择Fhosc/4，则Fcpu 频率为4MHz/4=1MHz。低速模式下，Fcpu=Flosc/4，即 16KHz/4=4KHz@3V， 32KHz/4=8KHz@5V。

当 High_CLK 选择 IHRC_16M 和 IHRC_RTC 时， Fcpu 被限制为 Fhosc/4~Fhosc/16。

高干扰环境下，强烈建议设置编译选项 Fcpu 频率低于 Fhosc/4，以减少干扰的影响。

6. 3、Noise Filter

杂讯滤波器（由编译选项“Noise_Filter”控制）是一个低通滤波器，支持外部振荡器，包括 RC 和晶体模式。杂讯滤波器可以滤除来自外部振荡器的高干扰信号。

在高干扰环境下，强烈建议开启杂讯滤波器以减少干扰的影响。

6. 4、HIGH_CLK 编译选项

系统高速时钟包括外部高速时钟和内部高速时钟。外部高速时钟又包括 4MHz、12MHz、32KHz 晶体/陶瓷和 RC 振荡器，高速时钟振荡器由编译选项 High_CLK 选择。内部高速时钟支持实时时钟（ RTC）功能，在 IHRC_RTC 模式下，内部高速时钟和外部 32KHz 振荡器有效，内部高速时钟作为系统时钟源，而外部 32KHz 振荡器为 RTC 时钟源，提供一个精确的实时时钟频率。

6. 5、HIGH_CLK 编译选项

对应不同的时钟功能，SONIX 提供多种高速时钟选项，由 High_CLK 选项控制。High_CLK

选项可以选择 IHRC_16M、IHRC_RTC、 RC、 32K X’tal、 12M X’tal 和 4M X’tal，以支持不同带宽的振荡器。

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

关于功耗，选择 IHRC_RTC 选项时，绿色模式下内部高速振荡器和内部低速振荡器都停止工作，仅外部 32768Hz 晶振正常工作，此时，看门狗定时器不能选择 Always_On 选项，否则内部低速振荡器会正常工作。

6. 5.1、内部高速 RC 振荡器(IHRC)

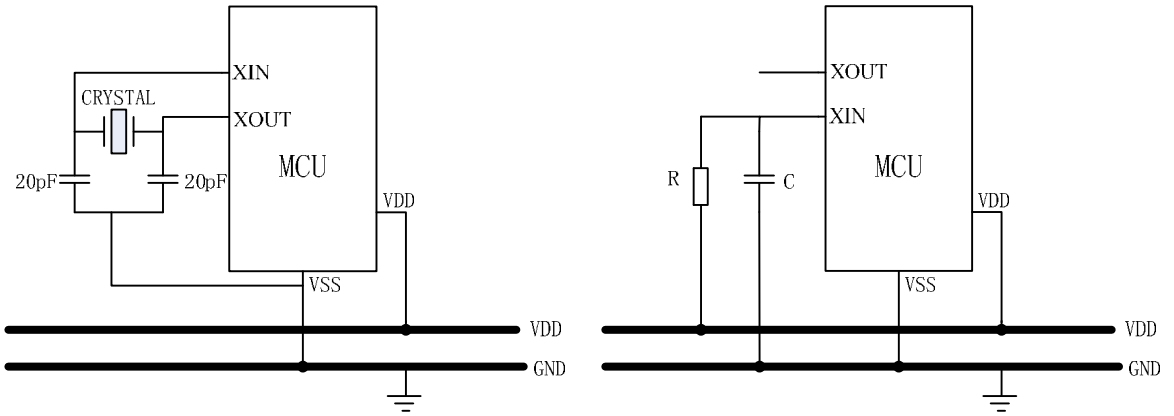
内部高速 16MHz RC 振荡器，普通环境下精确度为± 2%，当选择 IHRC_16M 或者 IHRC_RTC 时，使能内部高速振荡器。

- **IHRC_16M**： 系统高速时钟为内部 16MHz RC 振荡器， XIN/XOUT 为普通 I/O 引脚。
- **IHRC_RTC**：系统高速时钟为内部 16MHz RC 振荡器，外部 32768Hz 晶振作为实时时钟的时钟源， XIN/XOUT 连接外部 32768Hz 晶振。
- **IHRC_16M** 和 **IHRC_RTC** 模式下的 **Fcpu**：Fhosc/4~Fhosc/16。

6. 5.2、外部高速振荡器

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

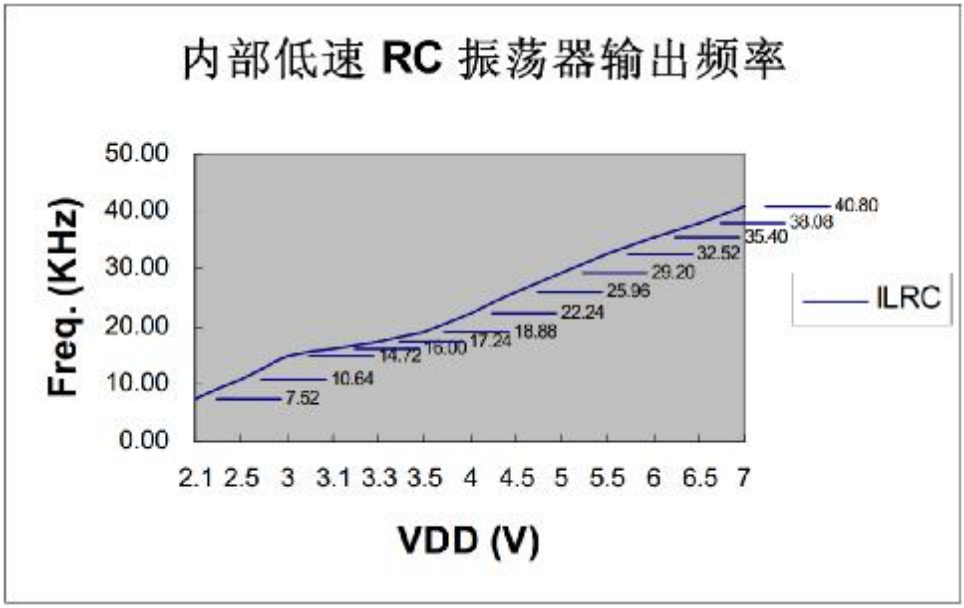
6. 5.3、外部振荡应用电路



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

6. 6、系统低速时钟

系统低速时钟源即内置的低速振荡器，采用 RC 振荡电路。低速时钟的输出频率受系统电压和环境温度的影响，通常为 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。**

有两种方式可以停止内部低速震荡，一种是进入睡眠模式，另外一种是系统使用外部 32k 作为震荡源， 进入绿色模式，且关闭看门狗；此时，系统仅仅只有外部 32K 震荡在工作，系统处于低功耗状态。

Ø 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。

6. 7、OSCM 寄存器

寄存器 OSCM 控制振荡器的状态和系统的工作模式。

0CAH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OSCM	0	0	0	CPUM1	CPUM0	CLKMD	STPHX	0
读/写	-	-	-	R/W	R/W	R/W	R/W	R/W
复位后	-	-	-	0	0	0	0	-

Bit 1 STPHX： 高速振荡器控制位。

0 = 运行；

1 = 停止，内部低速 RC 振荡器仍然运行。

Bit 2 CLKMD： 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。

Bit[4:3] CPUM[1:0]： CPU 工作模式控制位。

00 = 普通模式；

01 = 睡眠模式；

10 = 绿色模式；

11 = 系统保留。

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

- **IHRC_16M： STPHX=1**，禁止内部高速 **RC** 振荡器；
- **IHRC_RTC： STPHX=1**，禁止内部高速 **RC** 振荡器和外部 **32768Hz** 振荡器；
- **RC, 4M, 12M, 32K： STPHX=1**，禁止外部振荡器。

6. 8、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。

Ø 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。

6.9、系统时钟时序

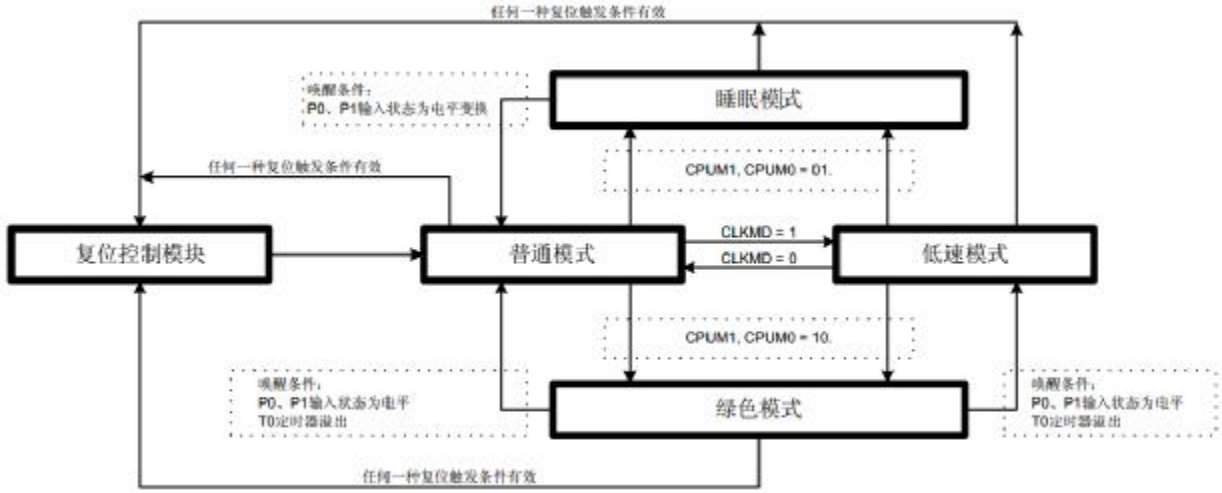
参数	符号	说明	典型值
硬件配置时间	Tcfg	2048*FILRC	64ms @ FILRC = 32KHz 128ms @ FILRC =16KHz
振荡器启动时间	Tost	启动时间取决于振荡器的材料、工艺等。通常情况下，低速振荡器的启动时间要比高速振荡器的启动时间慢，RC 振荡器的启动时间要比晶体/陶瓷振荡器的启动时间快。	-
振荡器起振时间	Tosp	复位情况下的振荡器起振时间为:2048*Fhosc (使能上电复位，LVD 复位，看门狗复位，外部复位引脚)	64ms @ Fhosc = 32KHz 512us @ Fhosc = 4MHz 128us @ Fhosc = 16MHz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

7、系统工作模式

7.1、概述

YF2501 可以在 4 种工作模式下以不同的时钟频率工作，这些模式可以控制振荡器的工作、程序的执行以及模拟电路的功能损耗。

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



系统工作模式切换框图

工作模式说明：

模式	普通	低速	绿色	睡眠	注释
EHOSC	运行	STPHX	STPHX	停止	
IHRC	运行	STPHX	STPHX	停止	
ILRC	运行	运行	运行	停止	
EHOSC，带有 RTC 功能	运行	By STPHX	运行	停止	
IHRC，带有 RTC 功能	运行	By STPHX	停止	停止	
ILRC，带有 RTC 功能	运行	运行	停止	停止	
CPU 指令	执行	执行	停止	停止	
T0 定时器	*有效	*有效	*有效	无效	* T0ENB=1 有效
TC0 定时器	*有效	*有效	无效	无效	* TC0ENB=1 有效
看门狗定时器	由 Watch_Dog 编译选项控制	由 Watch_Dog 编译选项控制	由 Watch_Dog 编译选项控制	由 Watch_Dog 编译选项控制	参考编译选项说明
内部中断	都有效	都有效	T0	都有效	
外部中断	都有效	都有效	都有效	都有效	
唤醒源	-	-	P0，P1，T0，复位	P0，P1，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（ 3V 时 16K， 5V 时 32K）。

7. 2、普通模式

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

- 程序被执行，所有的功能都可控制。
- 系统速率为高速。
- 高速振荡器和内部低速 RC 振荡器都正常工作。
- 通过 OSCM 寄存器，系统可以从普通模式切换到其它任何一种工作模式。

- 系统从睡眠模式唤醒后进入普通模式。
- 低速模式可以切换到普通模式。
- 从普通模式切换到绿色模式，唤醒后返回到普通模式。

7. 3、低速模式

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

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

7. 4、睡眠模式

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

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

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

7.5、绿色模式

绿色模式是另外的一种理想状态。在睡眠模式下，所有的功能和硬件设备都被禁止，但在绿色模式下，系统时钟保持工作，绿色模式下的功耗大于睡眠模式下的功耗。绿色模式下，不执行程序，但具有唤醒功能的定时器仍正常工作，定时器的时钟源为仍在工作的系统时钟。绿色模式下，有 2

种方式可以将系统唤醒： 1 、 P0 和 P1 电平变换触发； 2、具有唤醒功能的定时器溢出，这样，用户可以给定时器设定固定的周期，系统就在溢出时被唤醒。由 OSCM 寄存器 CPUM1 位决定是否进入绿色模式，当 CPUM1=1 ，系统进入绿色模式。当系统从绿色模式下被唤醒后，自动禁止 CPUM1（ 0 状态）。

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

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

7.6、工作模式控制宏

编译器提供工作模式控制宏以方便系统工作模式的切换。

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

Ø 例：从普通模式/低速模式切换进入睡眠模式

```
SleepMode                    ; 直接宣告“SleepMode”宏
```

Ø 例：从普通模式切换进入低速模式。

```
SlowMode                     ; 直接宣告“SlowMode”宏
```

Ø 例：从低速模式切换进入普通模式（外部高速振荡器停止工作）

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

Ø 例：从普通/低速模式切换进入绿色模式

```
GreenMode                    ; 直接宣告“GreenMode”宏
```

Ø 例：从普通/低速模式切换进入绿色模式，并使能 T0 唤醒功能。

```
; 设置定时器 T0 的唤醒功能。
    B0BCLR F        T0IEN        ; 禁止 T0 中断。
```

```
        B0BCLR      FT0ENB      ; 禁止 T0 定时器。
        MOV         A,#20H      ;
        B0MOV       T0M,A       ; 设置 T0 时钟= Fcpu / 64。
        MOV         A,#74H
        B0MOV       T0C,A       ;设置 T0C 的初始值=74H(设置 T0 间隔值=10 ms)。
        B0BCLR      FT0IEN      ; 禁止 T0 中断。
        B0BCLR      FT0IRQ      ; 清 T0 中断请求。
        B0BSET      FT0ENB      ; 使能 T0 定时器。
; 进入绿色模式。
        GreenMode               ; 直接宣告“GreenMode”宏
```

Ø 例：从普通/低速模式切换进入绿色模式，并使能 T0 的唤醒功能，带有 RTC 功能

```
        CLR         T0C         ; 清 T0 计数器。
        B0BSET      FT0TB       ; 使能 T0 RTC 功能。
        B0BSET      FT0ENB      ; 使能 T0 定时器。
; 进入绿色模式。
        GreenMode               ; 直接宣告“GreenMode”宏
```

7.7、唤醒时间

7.7.1、概述

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

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

7.7.2、唤醒时间

系统进入睡眠模式后，高速时钟停止运行。把系统从睡眠模式下唤醒时，单片机需要等待 2048 个外部高速振荡器时钟周期以使振荡电路进入稳定工作状态，等待的这一段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。

唤醒时间的计算如下：

唤醒时间=1/Fhosc*2048(sec)+高速时钟启动时间

Ø 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

内部高速 RC 振荡器的唤醒时间的计算如下：

唤醒时间=1/Fhosc*32(sec)+高速时钟启动时间

Ø 例：将系统从睡眠模式中唤醒，并设置系统进入普通模式。唤醒时间计算如下：

唤醒时间=1/Fhosc*32=2us

注：高速时钟的启动时间与 VDD 和振荡器类型有关。

7.7.3、P1W 唤醒控制寄存器

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

0C0H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P1W	-	-	-	-	P13W	P12W	P11W	P10W
读/写	-	-	-	-	W	W	W	W
复位后	-	-	-	-	0	0	0	0

Bit[3:0] P10W~P13W： Port 1 唤醒功能控制位。

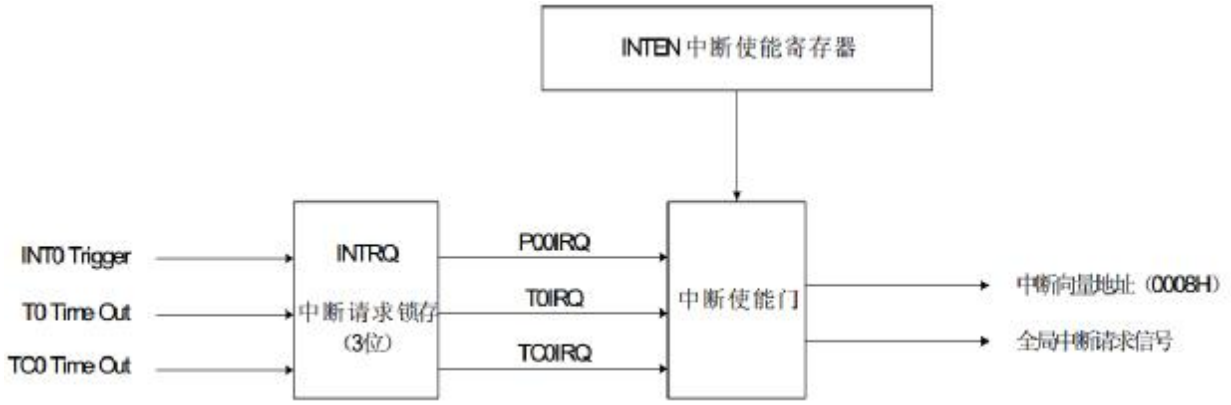
0 = 禁止 P1n 唤醒功能；

1 = 开放 P1n 唤醒功能。

8、中断

8. 1、概述

YF2501 提供 3 个中断源： 2 个内部中断（ T0/TC0）和 1 个外部中断（ INT0）。系统从睡眠模式进入高速普通模式时，外部中断能够将单片机唤醒。一旦程序进入中断，寄存器 STKP 的位 GIE 将被硬件自动清零以避免再次响应其它中断。系统退出中断后，硬件自动将 GIE 置“1 ”，以响应下一个中断。中断请求存放在寄存器 INTRQ 中。



注：程序响应中断时，位 GIE 必须处于有效状态。

8. 2、中断请求使能寄存器 INTEN

中断请求控制寄存器 INTEN 包括 2 个内部中断和 1 个外部中断，当有效位被置为 1 后，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	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTEN	-	-	TC0IEN	T0IEN	-	-	-	P00IEN
读/写	-	-	R/W	R/W	-	-	-	R/W
复位后	-	-	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。

8. 3、中断请求寄存器 INTRQ

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

0C8H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

INTRQ	-	-	TC0IRQ	T0IRQ	-	-	-	P00IRQ
读/写	-	-	R/W	R/W	-	-	-	R/W
复位后	-	-	0	0	-	-	-	0

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

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

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

8. 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。

0DFH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STKP	GIE	-	-	-	-	-	STKPB1	STKPB0
读/写	R/W	-	-	-	-	-	R/W	R/W
复位后	0	-	-	-	-	-	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。

Ø 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。

8. 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。

Ø 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
```

8. 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。

注： P0.0 的中断触发方式由 PEDGE 控制。

0BFH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
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）。

Ø 例： INT0 中断请求设置，电平触发。

```
                MOV         A, #18H
                B0MOV       PEDGE, A        ; INT0 置为电平触发。
                B0BCLR      FP00IRQ         ; INT0 中断请求标志清零。
                B0BSET      FP00IEN         ; 使能 INT0 中断。
                B0BSET      FGIE            ; 使能 GIE。
```

Ø 例： INT0 中断。

```
                ORG         8H
                JMP         INT_SERVICE
INT_SERVICE:
```

```
        …                       ; ACC 和 PFLAG 入栈保护。
        B0BTS1  FP00IRQ         ; 检测 P00IRQ。
        JMP     EXIT_INT        ; P00IRQ = 0，退出中断。
        B0BCLR  FP00IRQ         ; P00IRQ 清零。
        …                       ; INT1 中断服务程序。
        …
EXIT_INT:
        …                       ; ACC 和 PFLAG 出栈恢复。
        RETI                    ; 退出中断。
```

8. 7、T0 中断

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

Ø 例： T0 中断请求设置。

```
        B0BCLR  FT0IEN          ; 禁止 T0 中断。
        B0BCLR  FT0ENB
        MOV     A, #20H
        B0MOV   T0M, A          ; T0 时钟= Fcpu / 64。
        MOV     A, # 64H        ; T0C 初始值置为 64H。
        B0MOV   T0C, A          ; T0 间隔为 10 ms。
        B0BCLR  FT0IRQ          ; T0 中断请求标志清零。
        B0BSET  FT0IEN          ; 允许响应 T0 中断。
        B0BSET  FT0ENB
        B0BSET  FGIE            ; 使能 GIE。
```

Ø 例： T0 设置为无 RTC。

```
        ORG     8H
        JMP     INT_SERVICE
INT_SERVICE:
        …                       ; ACC 和 PFLAG 入栈保存。
        B0BTS1  FT0IRQ          ; 检查是否有 T0 中断请求标志。
        JMP     EXIT_INT
        B0BCLR  FT0IRQ          ; 清 T0IRQ。
        MOV     A, #64H
        B0MOV   T0C, A ;
        …                       ; T0 中断程序。
EXIT_INT:
        …                       ; ACC 和 PFLAG 出栈恢复。
        RETI                    ; 退出中断。
```

注：

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

Ø 例： RTC 下执行 T0 中断。

```
            ORG         8H
            JMP         INT_SERVICE
INT_SERVICE:
            …                       ; ACC 和 PFLAG 中断保护。
            B0BTS1      FT0IRQ      ; 检测 T0IRQ。
> 16us      JMP         EXIT_INT    ; T0IRQ = 0，退出中断。
            …                       ; T0 中断程序。
            …                       ;
            B0BCLR      FT0IRQ      ; T0IRQ 清零。
EXIT_INT:
            …                       ; 恢复 ACC 和 PFLAG。
            RETI                    ; 退出中断
```

8. 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。

Ø 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。
```

Ø 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。
```

8. 9、TC0 中断

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

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。

Ø 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     INTP01CHK       ; 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 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。
INT_EXIT:
        …                            ; 恢复 ACC 和 PFLAG。
        RETI                         ; 退出中断。
```

9、I/0 口

9. 1、概述

YF2501 内置 12 个 I/O 引脚，大多数 I/O 引脚与模拟引脚及特殊功能的引脚共用。详见下表：

I/O 引脚		公用引脚		公用引脚控制条件
引脚名称	引脚类型	引脚名称	引脚类型	
P0.0	I/O	INT0	DC	P00IEN=1
P1.1	I	RST	DC	Reset_Pin code option = Reset
		VPP	HV	OTP Programming
P1.2	I/O	XOUT	AC	High_CLK code option = 32K, 4M, 12M, IHRC_RTC
P1.3	I/O	XIN	AC	High_CLK code option = RC, 32K, 4M, 12M, IHRC_RTC
P1.4	I/O	PWM0	DC	TC0ENB = 1, PWM0OUT =1
		BZ0	DC	TC0ENB = 1, TC0OUT = 1, PWM0OUT = 0

注：DC：数字特性； AC：模拟特性； HV：高压特性。

9. 2、I/O 模式

寄存器 PnM 控制 I/O 口的工作模式。

0B8H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P0M	-	-	-	-	-	-	-	P00M
读/写	-	-	-	-	-	-	-	R/W
复位后	-	-	-	-	-	-	-	0

0C1H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P1M	-	-	-	-	P13M	P12M	-	P10M
读/写	-	-	-	-	R/W	R/W	-	R/W
复位后	-	-	-	-	0	0	-	0

0C2H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P2M	-	-	P25M	P24M	P23M	P22M	P21M	P20M

读/写	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	0	0	0	0	0	0

0C5H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P5M	-	-	-	P54M	-	-	-	-
读/写	-	-	-	R/W	-	-	-	-
复位后	-	-	-	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. P1.1 只能作为输入引脚，寄存器 P1M.1 未定义。

Ø 例： I/O 模式选择。

```
CLR     P0M         ; 所有端口设为输入模式。
CLR     P1M
CLR     P5M
MOV     A, #0FFH    ; 所有端口设为输出模式。
B0MOV   P0M, A
B0MOV   P1M, A
B0MOV   P5M, A
B0BCLR  P1M.2       ; P1.2 设为输入模式。
B0BSET  P1M.2       ; P1.2 设为输出模式。
```

9. 3、I/O 口上拉电阻

0E0H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P0UR	-	-	-	-	-	-	-	P00R
读/写	-	-	-	-	-	-	-	W
复位后	-	-	-	-	-	-	-	0

0E1H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P1UR	-	-	-	-	P13R	P12R		P10R
读/写	-	-	-	-	W	W		W
复位后	-	-	-	-	0	0		0

0E2H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P2UR	-	-	P25R	P24R	P23R	P22R	P21R	P20R
读/写	-	-	W	W	W	W	W	W

复位后	-	-	0	0	0	0	0	0

0E5H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P5UR	-	-	-	P54R	-	-	-	-
读/写	-	-	-	W	-	-	-	-
复位后	-	-	-	0	-	-	-	-

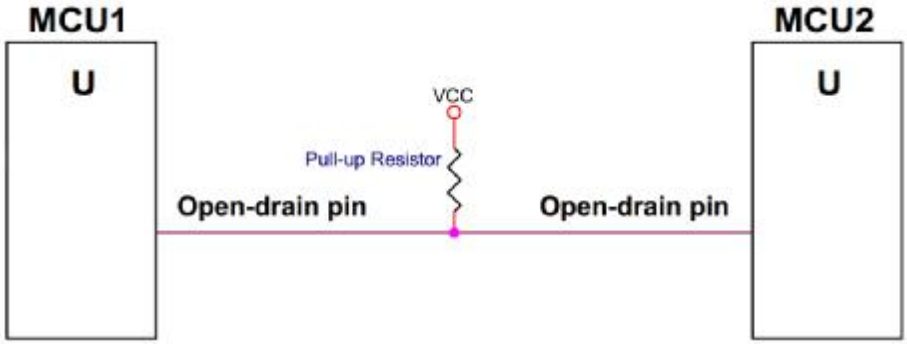
注： P1.1 是单向输入引脚，无上拉电阻， P1UR.1 未定义。

Ø 例： I/O 口的上拉电阻。

```
MOV     A, #0FFH        ; 使能 P0、 1 、 5 的上拉电阻。
B0MOV   P0UR, A         ;
B0MOV   P1UR, A
B0MOV   P5UR, A
```

9. 4、I/O 漏极开路寄存器

P1.0 内置漏极开路功能，当使能该功能时， P1.0 必须被置为输出模式。漏极开路的外部电路如下，图中的上拉电阻必不可少，漏极开路的输出高电平由上拉电阻驱动。



0E9H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P10C	-	-	-	-	-	-	-	P10OC
读/写	-	-	-	-	-	-	-	W
复位后	-	-	-	-	-	-	-	0

Bit 0 P10OC： P1.0 漏极开路控制位。

0 = 禁止漏极开路功能；

1 = 使能漏极开路功能。

Ø 例：开启 P1.0 的漏极开路功能并输出高电平。

```
B0BSET          P1.0                ; P1.0 置高。
```

```
B0BSET      P1M.0       ; P1.0 设为输出模式。
MOV         A, #01H     ; 开启 P1.0 的漏极开路功能。
B0MOV       P1OC, A
```

注： P10OC 是只写寄存器，只能通过指令“MOV”设置 P10OC。

Ø 例：禁止 P1.0 的漏极开路功能并输出低电平。

```
MOV         A, #0       ; 禁止 P1.0 的漏极开路功能。
B0MOV       P1OC, A
```

注：禁止 P1.0 的漏极开路功能后， P1.0 恢复到之前的 I/O 模式。

9. 5、I/O 数据寄存器

0D0H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P0	-	-	-	-	-	-	-	P00
读/写	-	-	-	-	-	-	-	R/W
复位后	-	-	-	-	-	-	-	0

0D1H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P1	-	-	-	-	P13	P12	P11	P10
读/写	-	-	-	-	R/W	R/W	R	R/W
复位后	-	-	-	-	0	0	0	0

0D2H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P2	-	-	P25	P24	P23	P22	P21	P20
读/写	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位后	-	-	0	0	0	0	0	0

0D5H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
P5	-	-	-	P54	-	-	-	-
读/写	-	-	-	R/W	-	-	-	-
复位后	-	-	-	0	-	-	-	-

注：当使能外部复位时， P11 的值保持为“1”。

Ø 例：从输入口读取数据。

```
B0MOV       A, P0       ; 从 P0 读数据。
B0MOV       A, P1       ; 从 P1 读数据。
B0MOV       A, P5       ; 从 P5 读数据。
```

Ø 例：从输入口读取数据。

```
B0MOV    A, P0      ; 从 P0 读数据。
B0MOV    A, P1      ; 从 P1 读数据。
B0MOV    A, P5      ; 从 P5 读数据。
```

Ø 例：写 1 位数据到输出口。

```
B0BSET   P1.3       ; P1.3 和 P5.4 置“1 ”。
B0BSET   P5.4
B0BCLR   P1.3       ; P1.3 和 P5.4 置“0”。
B0BCLR   P5.4
```

10、定时器

10. 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

看门狗定时器的 3 中工作模式有编译选项“WatchDog”控制：

- Disable： 禁止看门狗定时器功能。
- Enable： 使能看门狗定时器功能，在普通模式和低速模式下有效，在睡眠模式和绿色模式下看门狗停止工作。
- Always_On： 使能看门狗定时器功能，在睡眠模式和绿色模式下，看门狗仍会正常工作。

在高干扰环境下，强烈建议将看门狗设置为“Always_On”以确保系统在出错状态和重启时正常复位。

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

0CCH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
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

Ø 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
```

Ø 例：用宏指令@RST_WDT 清看门狗定时器。

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

看门狗定时器应用注意事项如下：

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

Ø 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。

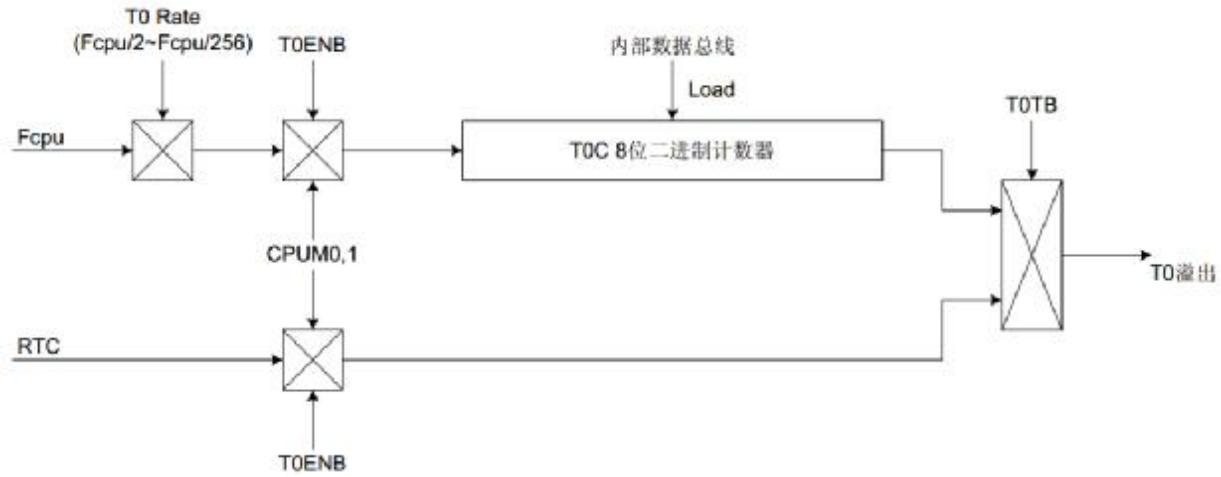
```
main:
        …                       ; 检测 I/O 口的状态。
        …                       ; 检测 RAM 的内容。
Err:    JMP     $               ; I/O 或 RAM 出错，不清看门狗等看门狗计时溢出。
Correct:                        ; I/O 和 RAM 正常，看门狗清零。
        MOV     A, #5AH ; 在整个程序中只有一处地方清看门狗。
        B0MOV   WDTR, A
        …
        CALL    SUB1
        CALL    SUB2
        …
        JMP     MAIN
```

10. 2、定时器 T0

10. 2.1、概述

8 位二进制基本定时器 T0 具有定时器功能置：支持标志指示（ T0IRQ）和中断操作（中断向量）。可以通过 T0M 和 T0C 寄存器控制间隔时间，支持 RTC 功能，具有在绿色模式下唤醒功能。在绿色模式下， T0 溢出，则将系统唤醒返回到上一个工作模式。

- 8 位可编程计数定时器： 根据选择的时钟频率周期性的产生中断请求。
- 中断功能： T0 定时器支持中断功能，当 T0 溢出， T0IRQ 有效，程序计数器跳到中断向量地址执行中断。
- RTC 功能： T0 支持 RTC 功能。 T0TB=1 时， RTC 时钟源由外部低速 32K 振荡器提供。 RTC 功能仅在 High_Clk 选择 IHRC_RTC 时有效。
- 绿色模式唤醒功能： T0 定时器在绿色模式下正常工作，溢出时将系统从绿色模式下唤醒。

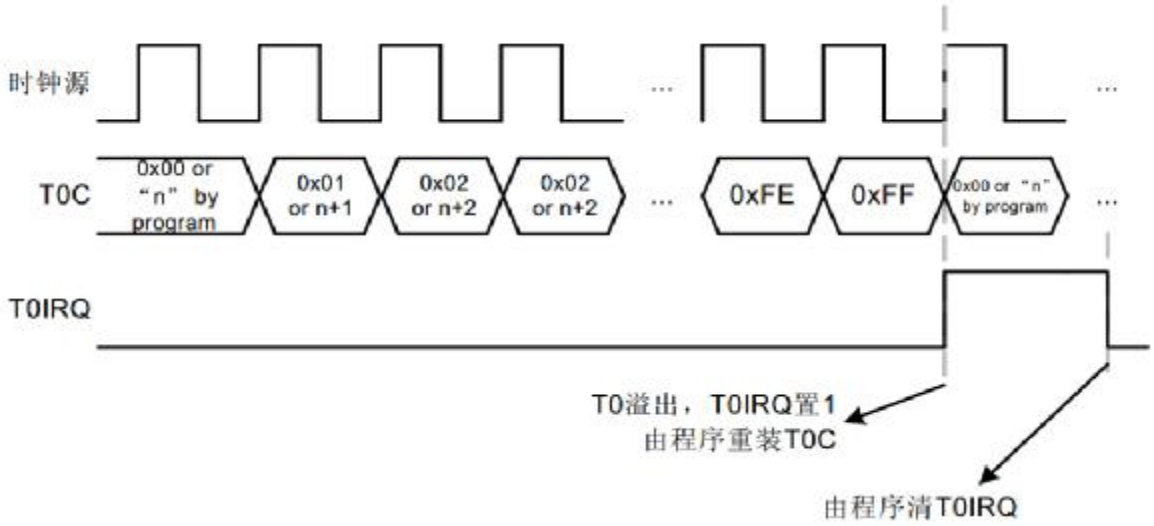


注：

1. 在 RTC 模式下，必须在 1/2 RTC 时钟源（ 32768HZ）后清 T0IRQ，否则 RTC 的间隔时间会出错。利用 T0 中断服务程序的执行时间来延时 16us；
2. 在 RTC 模式下，不能在中断中复位 T0C。

10. 2.2、概述

T0 定时器由 T0ENB 控制。当 T0ENB=0 时，T0 停止工作；当 T0ENB=1 时， T0 开始计数。 T0C 溢出（从 0FFH 到 00H）时， T0IRQ 置 1 显示溢出状态并由程序清零。 T0 无内置双重缓存器，故 T0 溢出时由程序加载新值给 T0C，以选定合适的间隔时间。如果使能 T0 中断（T0IEN=1 ），T0 溢出后系统执行中断服务程序，在中断下必须由程序清 T0IRQ。T0 可以在普通模式、低速模式和绿色模式下工作，绿色模式下，T0 溢出时 T0IRQ 置 1，系统被唤醒。



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

T0rate[2:0]	T0 Clock	T0 间隔时间					
		Fhosc=16MHz, Fcpu=Fhosc/4		Fhosc=4MHz, Fcpu=Fhosc/4		IHRC_RTC 模式	
		max. (ms)	Unit (us)	max. (ms)	Unit (us)	max. (sec)	Unit (ms)
000b	Fcpu/256	16.384	64	65.536	256	-	-
001b	Fcpu/128	8.192	32	32.768	128	-	-
010b	Fcpu/64	4.096	16	16.384	64	-	-
011b	Fcpu/32	2.048	8	8.192	32	-	-
100b	Fcpu/16	1.024	4	4.096	16	-	-
101b	Fcpu/8	0.512	2	2.048	8	-	-
110b	Fcpu/4	0.256	1	1.024	4	-	-
111b	Fcpu/2	0.128	0.5	0.512	2	-	-
-	32768Hz/64	-	-	-	-	0.5	1.953

10. 2.3、T0M 模式寄存器

模式寄存器 T0M 设置 T0 的工作模式，包括 T0 前置分频器、时钟源等，这些设置必须在使能 T0 定时器之前完成。

0D8H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	T0TB
读/写	R/W	R/W	R/W	R/W	-	-	-	R/W
复位后	0	0	0	0	-	-	-	0

Bit0 T0TB： RTC 时钟源选择控制位。

0 = 禁止 RTC(T0 时钟源为 Fcpu)；

1 = 使能 RTC。

Bit [6:4] T0RATE[2:0]： T0 分频选择位。

000 = Fcpu/256；

001 = Fcpu/128；

010 = Fcpu/64；

011 = Fcpu/32；

100 = Fcpu/16；

101 = Fcpu/8；

110 = Fcpu/4；

111 = Fcpu/2。

Bit 7 T0ENB： T0 启动控制位。

0 = 禁止；

1 = 使能。

注： RTC 模式下， T0RATE 处于无效状态。 T0 的间隔时间固定为 0.5S。

10. 2.4、T0M 模式寄存器

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

0D9H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T0C	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

T0C 初始值的计算公式如下：

T0C 初始值=256-(T0 间隔时间*T0 时钟 Rate)

Ø 例：计算 T0C。 T0 间隔时间为 10ms。 T0 时钟源 Fcpu = 4MHz/4 = 1MHz， T0RATE = 001（ Fcpu/128）。 T0 间隔时间=10ms， T0 时钟 rate=4MHz/4/128

T0C 初始值 = 256 -（ T0 间隔时间 * 输入时钟信号）

= 256 -（ 10ms * 4MHz / 4 / 128）

= 256 -（ 10-2 * 4 * 106 / 4 / 128）

= B2H

注： RTC 模式下， T0C 为 256， T0 的间隔时间为 0.5S。不能在 RTC 模式下修改 T0C 的值。

10. 2.5、T0 操作举例

l T0 定时器：

```
; 复位 T0 定时器。
        MOV     A, #00H         ; 清 T0M。
        B0MOV   T0M, A
; 设置 T0 时钟源和 T0Rate。
        MOV     A, #0nnn0000b
```

```
        B0MOV   T0M, A
; 设置 T0C 寄存器获取 T0 间隔时间。
        MOV     A, #value
        B0MOV   T0C, A
; 清 T0IRQ。
        B0BCLR  FT0IRQ
; 使能 T0 定时器和中断功能。
        B0BSET  FT0IEN          ; 使能 T0 中断。
        B0BSET  FT0ENB          ; 使能 T0 定时器。
```

- T0 在 RTC 模式下工作：

```
; 复位 T0 定时器。
        MOV     A, #00H         ; 清 T0M。
        B0MOV   T0M, A
; 设置 T0 RTC 功能。
        B0BSET  FT0TB
; 清 T0C。
        CLR     T0C
; 清 T0IRQ。
        B0BCLR  FT0IRQ
; 使能 T0 定时器和中断功能。
        B0BSET  FT0IEN          ; 使能 T0 中断。
        B0BSET  FT0ENB          ; 使能 T0 定时器。
```

10. 3、定时/计数器 TC0

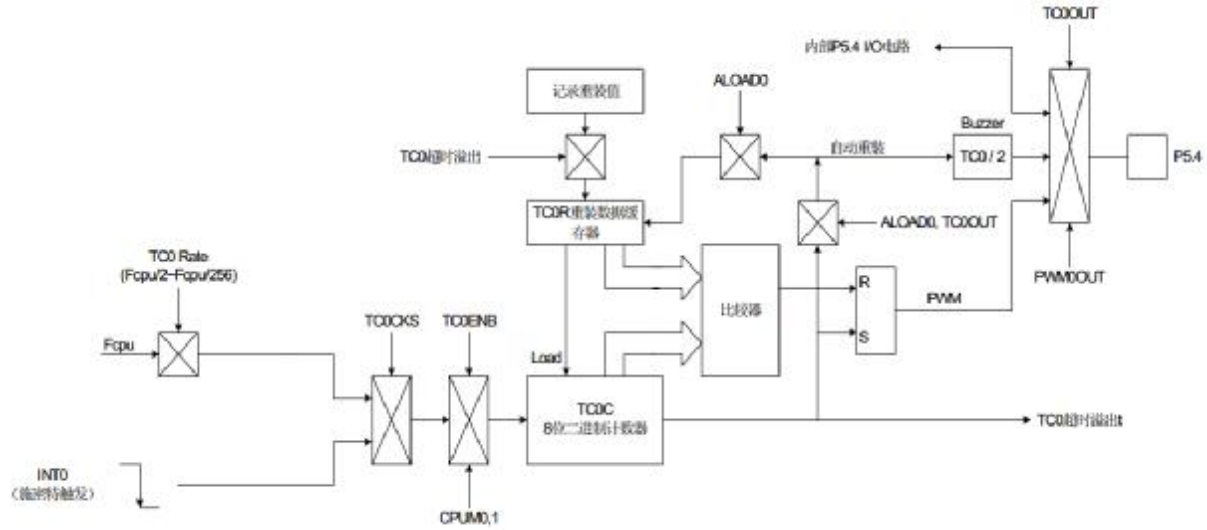
10. 3.1、概述

8 位二进制定时/计数器具有基本定时器、事件计数器、 Buzzer 和 PWM 功能。基本定时器功能可以支持标志显示（ TC0IRQ）和中断操作（中断向量）。由 TC0M、 TC0C、 TC0R 寄存器控制 TC0 的中断间隔时间。事件计数器可以将 TC0 时钟源由系统时钟更改为外部时钟信号（如连续的脉冲、 R/C 振荡信号等）。 TC0 作为计数器时记录外部时钟数目以进行测量应用。 TC0 还内置 Buzzer 和 PWM 功能， Buzzer 和 PWM 的周期和分辨率由 TC0 时钟 Rate、 TC0R 寄存器控制，故具有良好性能的 Buzzer 和 PWM 可以处理 IR 载波信号，马达控制和光度调节等。

TC0 的主要用途如下：

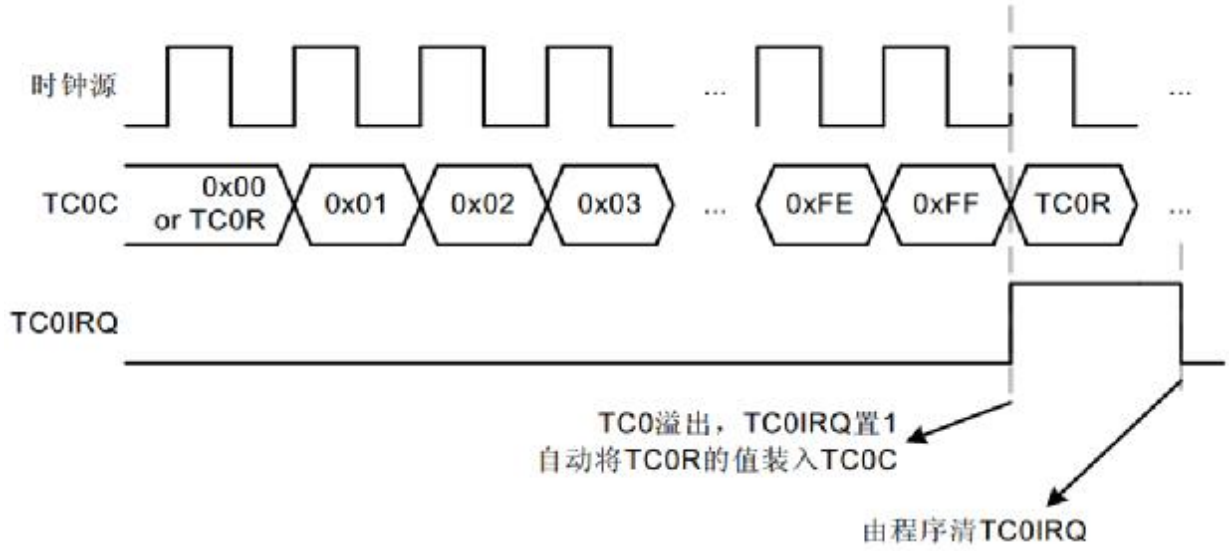
- 8 位可编程定时器：根据选择的时钟信号，产生周期中断；
- 中断功能：TC0 定时器支持中断，当 TC0 溢出时，TC0IRQ 置 1，系统执行中断；
- 外部事件计数器：对外部事件计数；

- PWM 输出：由 TC0rate 和 TC0R 寄存器控制占空比/周期；
- Buzzer 输出： Buzzer 的输出信号是 TC0 间隔时间的 1/2 个周期；
- 绿色模式功能：绿色模式下，TC0 正常工作，但无唤醒功能。



10. 3.2、TC0 操作

TC0 定时器由 TC0ENB 控制。当 TC0ENB=0 时， TC0 停止工作；当 TC0ENB=1 时， TC0 开始计数。使能 TC0 之前，先要设定好 TC0 的功能模式，如基本定时器、TC0 中断等。 TC0C 溢出（从 0FFH 到 00H）时，TC0IRQ 置 1 以显示溢出状态并由程序清零。在不同的功能模式下，TC0C 不同的值对应不同的操作，若改变 TC0C 的值影响到操作，会导致功能出错。TC0 内置双重缓存器以避免此种状况的发生。在 TC0C 计数的过程中不断的刷新 TC0C，保证将最新的值存入 TC0R（重装缓存器）中，当 TC0 溢出后，TC0R 的值由自动存入 TC0C。进入下一个周期后，TC0 进入新的工作状态。定时器/计数器模式下，由 ALOAD0 控制自动重装功能；PWM 模式下，使能 TC0 时，自动使能 TC0 的自动重装功能。如果使能 TC0 中断功能（TC0IEN=1），在 TC0 溢出时系统执行中断服务程序，在中断时必须由程序清 TC0IRQ。TC0 可以在普通模式、低速模式和绿色模式下工作。但在绿色模式下， TC0 虽继续工作，但不能唤醒系统。



TC0 根据不同的时钟源选择不同的应用模式， TC0 的时钟源由 Fcpu（指令周期）和外部引脚输入（ P0.0）提供，由 TC0CKS 控制。 TC0CKS 选择时钟源来自 Fcpu 或者外部引脚输入。当 TC0CKS=0 时， TC0 时钟源来自 Fcpu，可以由 TC0Rate[2:0]选择不同的分频。 当 TC0CKS=1 时， TC0 时钟源由外部引脚提供， 此时使能外部事件计数功能。 TC0CKS=1 时， TC0Rate[2:0]处于无效状态。

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

10. 3.3、TC0M 模式寄存器

模式寄存器 TC0M 控制 TC0 的工作模式：包括 TC0 前置分频，时钟源， PWM 功能等，这些功能必须在使能 TC0 定时器之前设置完成。

0DAH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TC0M	TC0ENB	TC0rate2	TC0rate1	TC0rate0	TC0CKS	ALOAD0	TC0OUT	PWM0OUT

读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

Bit 0 PWM0OUT： PWM 输出控制。

0 = 禁止 PWM 输出；

1 = 使能 PWM 输出， PWM 输出占空比由 TC0OUT 和 ALOAD0 控制。

Bit 1 TC0OUT： TC0 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；

1 = 使能。

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）。

10. 3.4、TC0C 计数寄存器

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

0DBH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TC0C	TC0C7	TC0C6	TC0C5	TC0C4	TC0C3	TC0C2	TC0C1	TC0C0

读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

TC0C 初始值的计算公式如下：

TC0C 初始值=N-(TC0 中断间隔时间*TC0 时钟 Rate)

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	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

10. 3.5、TC0C 计数寄存器

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

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

0CDH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
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 的溢出时间有如下五种可能情况：

TC0CKS	PWM0	ALOAD0	TC0OUT	N	TCOR 计数范围	TC0R 二进制计数范围
0	0	X	X	256	00H~0FFH	000000000B~11111111B
	1	0	0	256	00H~0FFH	000000000B~11111111B
	1	0	1	64	00H~3FH	000000000B~xx111111B
	1	1	0	32	00H~1FH	000000000B~xxx11111B
	1	1	1	16	00H~0FH	000000000B~xxxx1111B
1	-	-	-	256	00H~0FFH	000000000B~11111111B

Ø 例：计算 TC0C 和 TC0R 的值， TC0 间隔时间为 10ms， 时钟源 Fcpu=4MHz/4=1MHz，TC0RATE = 001（ Fcpu/128）。TC0 间隔时间为 10ms， TC0 时钟 rate 为 4Hz/428

TC0C/TC0R 初始值 = 256 -（ TC0 中断间隔时间 * 输入时钟源）

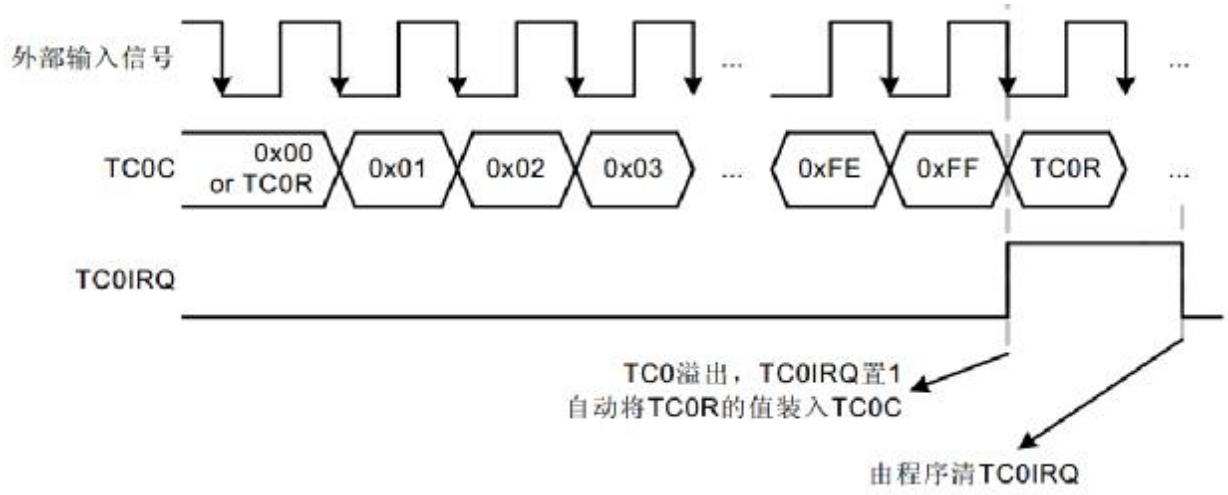
= 256 -（ 10ms * 4Hz / 4/ 128）

= 256 -（ 10-2 * 4* 106 / 4/ 128）

= 0B2H

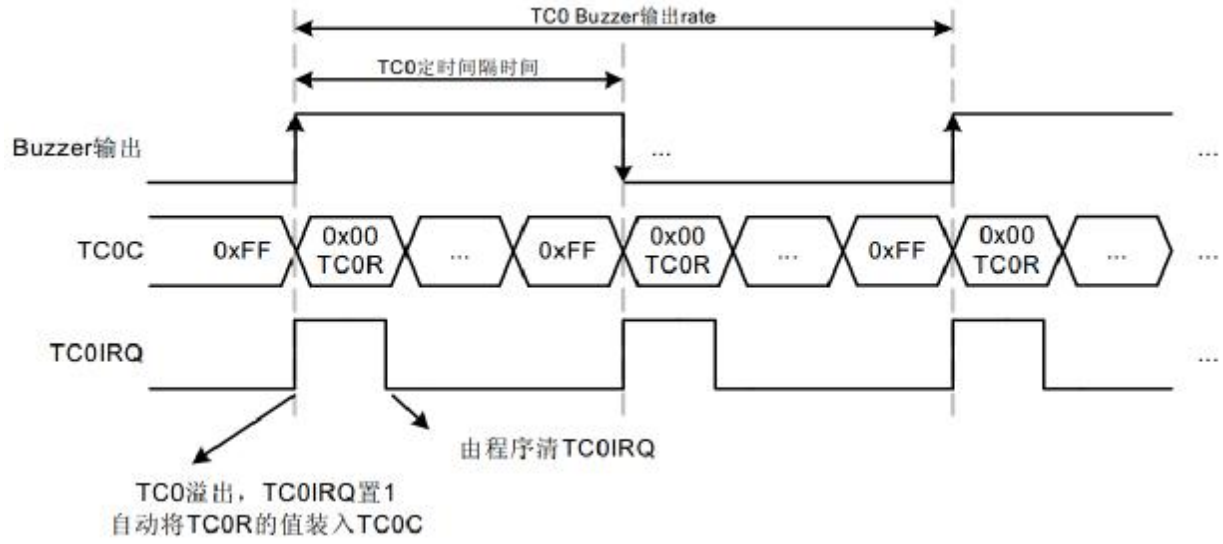
10. 3.6、TC0C 事件计数器

TC0 作为外部事件计数器时，其时钟源由外部输入引脚（ P0.0）提供。当 TC0CKS1=1 时，TC0 的时钟源由外部输入引脚（ P0.0）提供，下降沿触发。 TC0C 溢出（从 FFH 到 00H）时，TC0触发事件计数器溢出。使能外部事件计数功能，同时禁止外部输入引脚的唤醒功能以避免外部事件的触发信号将系统唤醒而耗电。此时， P0.0 的外部中断功能也被禁止，即 P00IRQ=0。外部事件计数器通常用来测量外部连续信号的比率，如连续的脉冲信号， R/C 振荡信号等，外部信号的相位与 MCU 时钟的相位并不同步。



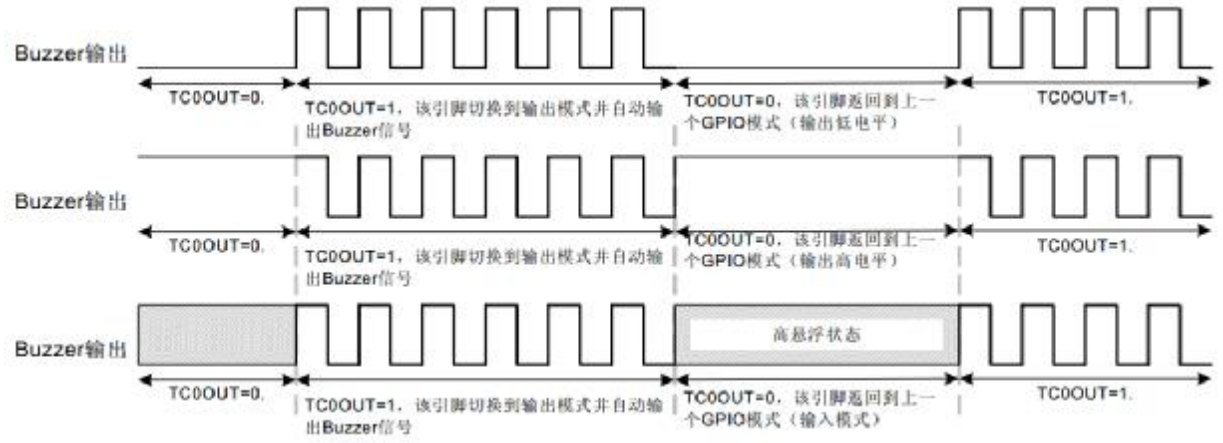
10. 3.7、TC0C BUZZER 输出

当每次TC0溢出时，Buzzer的输出电平发生改变，从而产生一个方波。Buzzer 输出方波的频率为TC0溢出频率的1/2。Buzzer输出的波形图如下所示：



TC0 溢出后，Buzzer输出时，TC0IRQ有效，且当TC0IEN=1 时，使能TC0中断功能。但强烈建议小心同时使用Buzzer和TC0定时器，以确保两种功能都能正常工作。

Buzzer 输出引脚与GPIO引脚共用，TC0OUT=1时，该引脚自动设为Buzzer输出引脚。如清TC0OUT 位以禁止Buzzer输出后，该引脚自动返回到最后一个GPIO 模式。

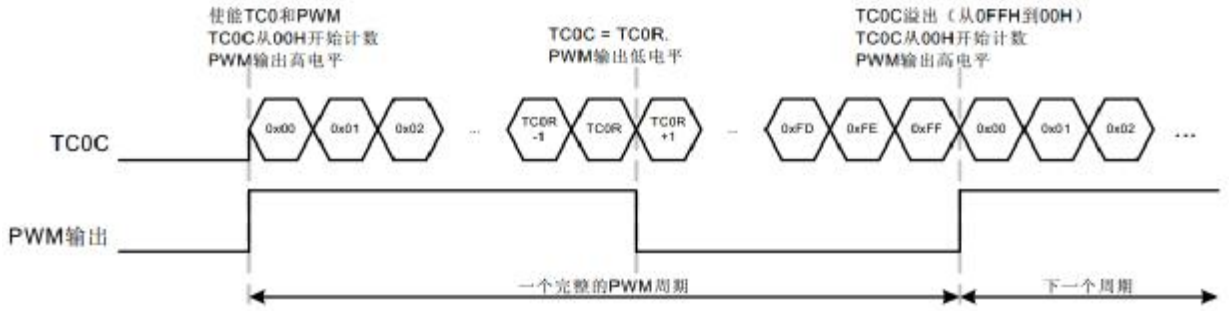


注： PWM 模式下，由于是 TC0OUT 决定 PWM 的周期，故 Buzzer 输出时， PWM0OUT 必须置 0。

10. 3.8、脉冲宽度调制（PWM）

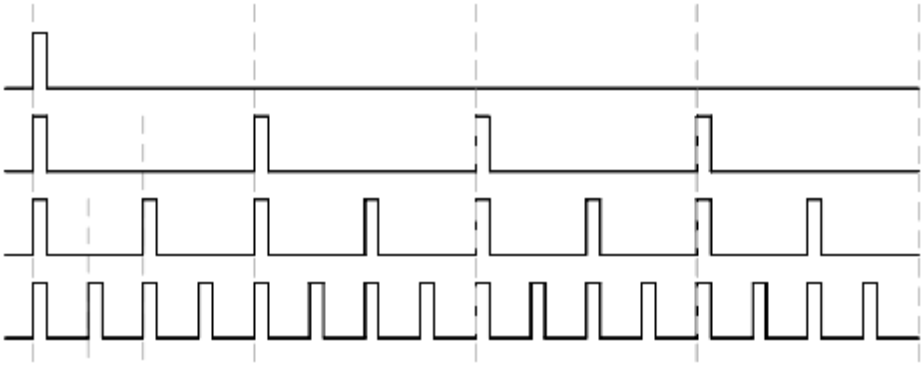
可编程控制占空比/周期的 PWM 可以提供不同的 PWM 信号。使能 TC0 定时器且 PWM0OUT=1 时，由 PWM 输出引脚（ P5.4）输出 PWM 信号。 1 个 PWM 周期首先输出高电平，然后输出低电平。 TC0Rate 控制 PWM 的周期， ALOAD0和 TC0OUT 决定 PWM 的分辨率，TC0R 寄存器决定 PWM 的占空比（脉冲高电平的长度）。开启 TC0 定时器后， TC0C从 0 开始计数，一直到 TC0C 溢出。当 TC0C=TC0R 时， PWM 输出从高电平变为低电平； TC0 溢出时，PWM 输出从低电平变为高电平，整个 PWM 周期完成，并进入下一个周期。只有在每次 TC0 溢出时（即一个 PWM 周期完成时），定时器才会载入 TC0R 的值，以保持 PWM 的连贯性，换一句话理解，既是 PWM 输出的过程中由程序更改 PWM 的占空比，则在下一个周期才开始输出新的占空

比的 PWM.。



PWM 的分辨率由 ALOAD0和TC0OUT 决定，包括 1/256、1/64、1/32、1/16，以实现高速 PWM 信号。当 ALOAD0、C0OUT = 00 时， PWM 的分辨率为 1/256； ALOAD0、 TC0OUT = 01 时，PWM 的分辨率为 1/64； ALOAD0、 TC0OUT= 10 时， PWM 的分辨率为 1/32； ALOAD0、TC0OUT = 11 时， PWM 的分辨率为 1/16。若需调制 PWM 的分辨率， TC0RPWM 的占空比控制范围必须调制到一个合适的分辨率。 PWM 输出过程中， TC0 溢出时， TC0IRQ 有效， TC0IEN=1 时，则使能 TC0 中断。但强烈建议小心同时使用 PWM 和 TC0 定时器功能，保证两种功能都能正常工作。

ALOAD0	TC0OUT	PWM 分辨率	TC0R 有效值	TC0R 有效值（二进制）
0	0	256	00H~FFH	00000000b~11111111b
0	1	64	00H~3FH	xx000000b~xx111111b
1	0	32	00H~1FH	xxx00000b~xxx11111b
1	1	16	00H~0FH	xxxx0000b~xxxx1111b



PWM 输出引脚和 GPIO 引脚共用， PWM0OUT=1 时， 该引脚自动输出 PWM 信号。如果清 PWM0OUT 位以禁止 PWM时，该引脚返回到最后一个 GPIO 模式。

10. 3.9、TC0 操作举例

I TC0 定时器

```
; 复位 TC0。
        MOV     A,#00H          ; 清 TC0M。
```

```
                B0MOV       TC0M,A
; 设置 TC0Rate 和自动重装功能。
                MOV         A, #0nnn0000b       ; TC0rate[2:0]。
                B0MOV       TC0M, A
                B0BSET      FALOAD0
; 设置 TC0C 和 TC0R 获得 TC0 的间隔时间。
                MOV         A, #value           ; TC0C 必须和 TC0R 相等。
                B0MOV       TC0C, A
                B0MOV       TC0R, A
; 清 TC0IRQ。
                B0BCLR      FTC0IRQ
; 使能 TC0 定时器和中断功能。
                B0BSET      FTC0IEN             ; 使能 TC0 中断。
                B0BSET      FTC0ENB             ; 使能 TC0 定时器。
```

■ TC0 事件计数器

```
; 复位 TC0。
                MOV         A,#00H              ; 清 TC0M。
                B0MOV       TC0M,A
; 设置 TC0 自动重装功能。
                B0BSET      FALOAD0
; 使能 TC0 事件计数器。
                B0BSET      FTC0CKS             ; 设置 TC0 的时钟源由外部输入引脚（P0.0）提供。
; 设置 TC0C 和 TC0R 寄存器获得 TC0 的间隔时间。
                MOV         A, #value           ; TC0C 必须和 TC0R 相等。
                B0MOV       TC0C, A
                B0MOV       TC0R, A
; 清 TC0IRQ。
                B0BCLR      FTC0IRQ
; 使能 TC0 定时器和中断功能。
                B0BSET      FTC0IEN             ; 使能 TC0 中断。
                B0BSET FTC0ENB                  ; 使能 TC0 定时器。
```

■ TC0 BUZZER 输出

```
; 复位 TC0。
                MOV         A, #00H             ; 清 TC0M。
                B0MOV       TC0M, A
; 设置 TC0rate 和自动重装功能。
```

```
        MOV     A, #0nnn0000b       ; TC0rate[2:0]。
        B0MOV   TC0M, A
        B0BSET  FALOAD0
; 设置 TC0C 和 TC0R 寄存器获得 TC0 的间隔时间。
        MOV     A, #value           ; TC0C 必须和 TC0R 相等。
        B0MOV   TC0C, A
        B0MOV   TC0R, A
; 使能 TC0 定时器和 Buzzer 输出功能。
        B0BSET  FTC0ENB             ; 使能 TC0 定时器。
        B0BSET  FTC0OUT             ; 使能 TC0 buzzer 输出功能。
```

- TC0 PWM

```
; 复位 TC0。
        MOV     A, #00H             ; 清 TC0M。
        B0MOV   TC0M, A
; 设置 TC0Rate 和 PWM 周期。
        MOV     A, #0nnn0000b       ; TC0rate[2:0]。
        B0MOV   TC0M, A
; 设置 PWM 分辨率。
        MOV     A, #00000nn0b       ; ALOAD0 和 TC0OUT。
        OR      TC0M,A
; 设置 TC0R 寄存器，获取 PWM 占空比。
        MOV     A, #value
        B0MOV   TC0R,A
; 清 TC0C。
        CLR     TC0C
; 使能 PWM 和 TC0 定时器。
        B0BSET  FTC0ENB             ; 使能 TC0 定时器。
        B0BSET  FPWM0OUT            ; 使能 PWM。
```

11、指令表

Field	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 仅适用地址是 0x80~0x87 的系统寄存器，如 R、Y、Z...，I 不能是 E6h、E7h。)	-	-	-	1
	XCH	A,M	A ←→M	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)。(M 不使用地址是 0x80~0x87 的系统寄存器)	-	-	-	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
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 标志位可能受影响。	√	-	√	1 + S
	CMPRS	A,M	比较，如果相等则跳过下一条指令，C 与 ZF 标志位可能受影响。	√	-	√	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，则跳过下一条指令。	-	-	-	1 + S
	JMP	d	跳转指令，PC15/14 ← RomPages1/0，PC13~PC0 ← d。	-	-	-	2
	CALL	d	子程序调用指令，Stack ← PC15~PC0，PC15/14 ← RomPages1/0，PC13~PC0 ← d。	-	-	-	2
MISC	RET		子程序跳出指令，PC ← Stack。	-	-	-	2
	RETI		中断处理程序跳出指令，PC ← Stack，并使能全局中断控制位。	-	-	-	2
	NOP		空指令，无特别意义。	-	-	-	1

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

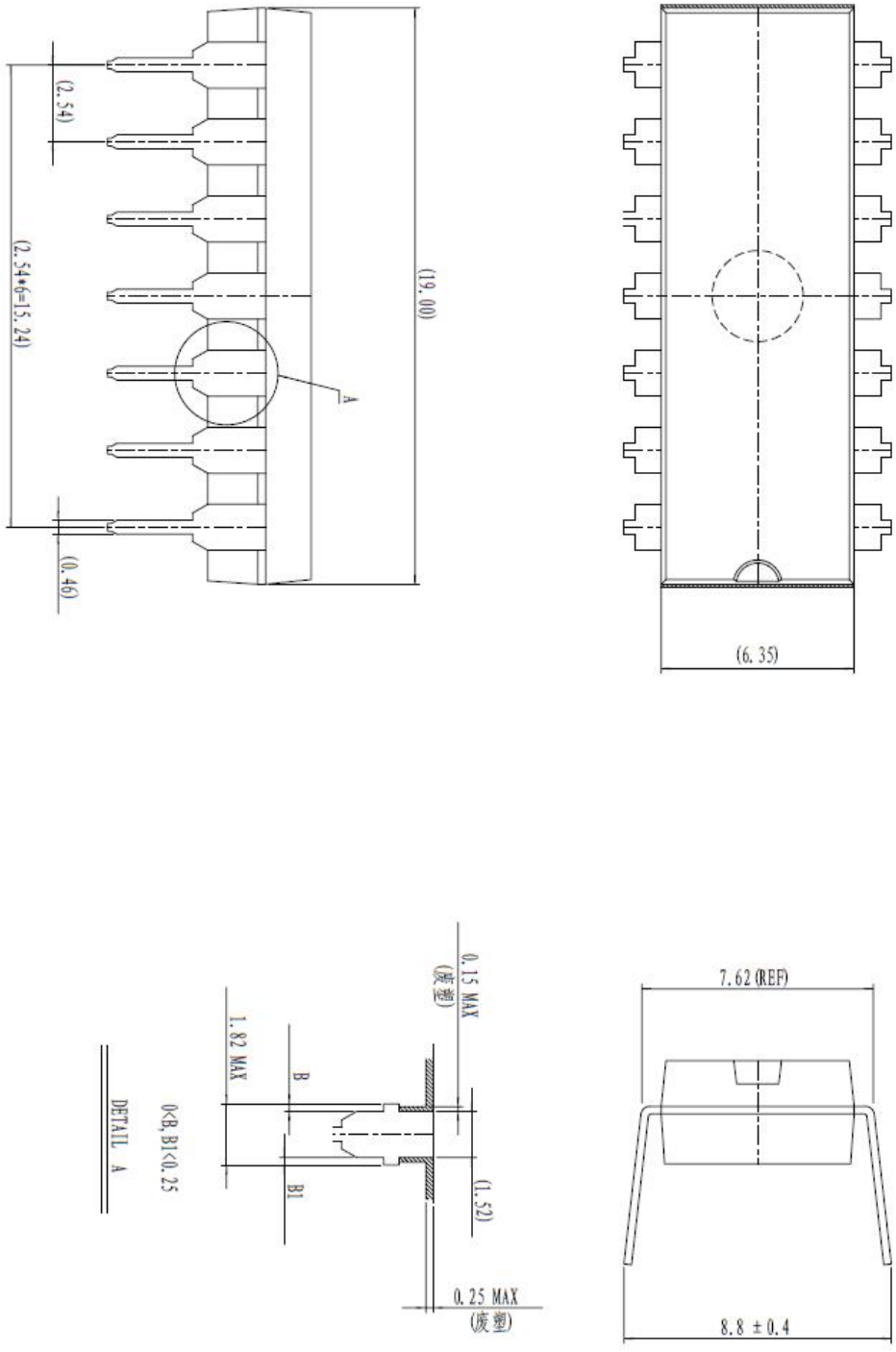
2. 条件跳转指令的条件为真，则 S = 1 ，否则 S = 0。

3. 指令“B0MOV M, I”中的“I”不能是“E6h”或“E7h”。

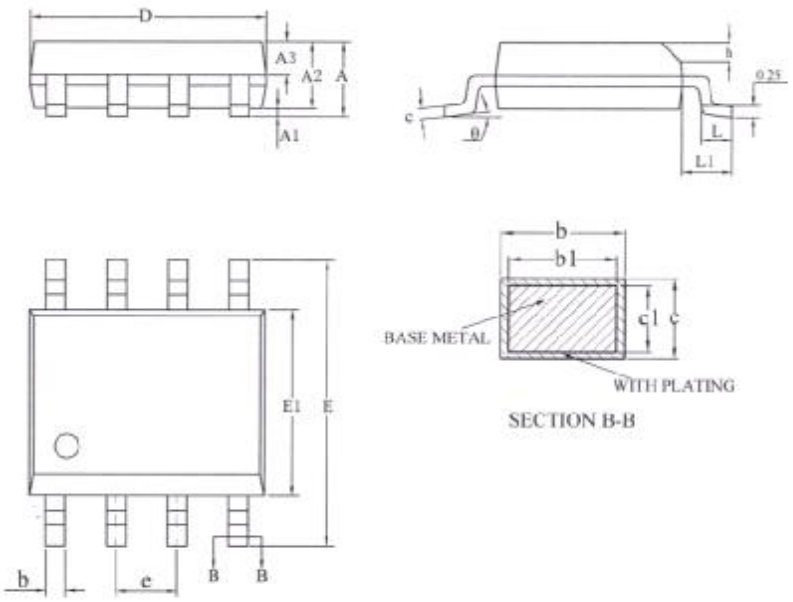
4. 指令“B0XCH”中的 M 不能是系统寄存器的 0x80~0xFF 单元。

12、封装尺寸与外形图

12. 1、DIP14 外形图与封装尺寸(单位：MM)

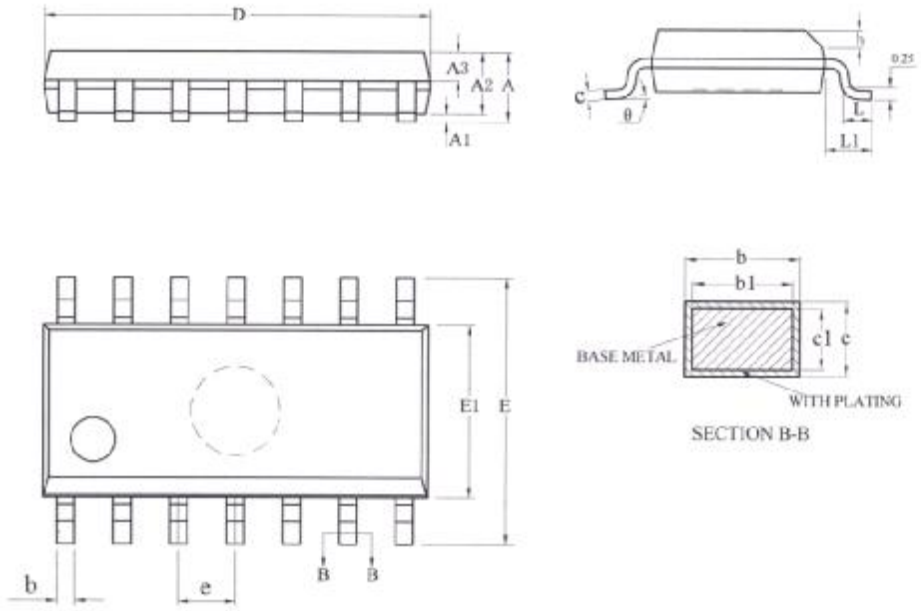


12. 2、SOP8 外形图与封装尺寸



SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	1.75
A1	0.10	—	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	—	0.48
b1	0.38	0.41	0.43
c	0.21	—	0.26
c1	0.19	0.20	0.21
D	4.70	4.90	5.10
E	5.80	6.00	6.20
E1	3.70	3.90	4.10
e	1.27BSC		
h	0.25	—	0.50
L	0.50	—	0.80
L1	1.05BSC		
θ	0	—	8°

12. 3、SOP14 外形图与封装尺寸



SYMBOL	MILLIMETER		
	MIN	NOM	MAX
A	—	—	1.75
A1	0.05	—	0.225
A2	1.30	1.40	1.50
A3	0.60	0.65	0.70
b	0.39	—	0.48
b1	0.38	0.41	0.43
c	0.21	—	0.26
c1	0.19	0.20	0.21
D	8.45	8.65	8.85
E	5.80	6.00	6.20
E1	3.70	3.90	4.10
e	1.27BSC		
h	0.25	—	0.50
L	0.50	—	0.80
L1	1.05BSC		
θ	0	—	8°

13、声明及注意事项：

13.1、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。					

13.2 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。