FC706

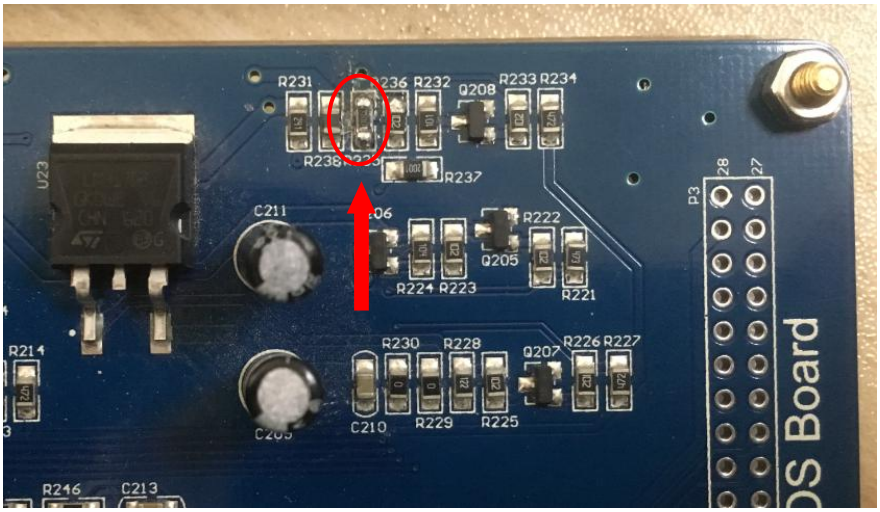
用户参考手册

SZFC 8 位单片机

注意事项：

1、 对 MTP 进行写数据时，建议使用单 bit 写，且连写 2 次，该操作使得 MTP 在低压下会有良好的表现。具体可见 23 页例程。

2、 在烧录量产 FC706 时，如下图所示，请确保烧写器右上方 R235 位置电阻为 47 Ω，并测量下 FC706 在烧录时 P10 脚的电压是为 7V 左右。客户可将烧写器寄回我司进行修改，也可自行修改。



3、 由于工艺的差异性，烧录软件上配置的 LVDT 复位电压，与芯片实际的复位电压，有至少±0.2V 以上的误差。

修正记录

版本号	日期	内容
VER 1.0	2017 年 2 月	初版
VER 2.0	2017 年 4 月	更新芯片部分寄存器的说明
VER 2.1	2017 年 4 月	更新 Y,Z,R 寄存器说明
VER 3.0	2017 年 6 月	更新 PEDGE 寄存器说明
VER 4.0	2017 年 10 月	增加封装信息
VER 5.0	2017 年 11 月	增加注意事项及 MTP 操作例程修改
VER 6.0	2017 年 12 月	增加烧录注意事项
VER 7.0	2018 年 1 月	增加极性参数
VER 8.0	2019 年 12 月	增加 8 脚脚位图
VER 8.1	2021 年 7 月	增加注意事项 3

目录

1 产品简介

1.1 功能特性

- 存储器配置
  OTP ROM 空间：**6K * 16** 位。
  RAM 空间：**512x8** 位。
  MTP 空间：**512x8** 位

- **8** 层堆栈缓存器

- **I/O** 引脚配置
  输入输出双向端口，T-KEY 扫描脚,具有唤醒功能的端口，内置上拉电阻端口；
  P10，P12，P13，P14，P15，P16，P17，P23，P22，P21，P20 脚
  外部中断引脚：
  P25，P24，P11

- **8** 级低电压检测系统（**LVD**）
  系统复位，监控系统电源。

- **6** 个中断源
  3 个内部中断：TC0、TC1、MTPIF。
  3 个外部中断：INT0、INT1、RBIF。

- 两个 **8** 位定时/计数器
  TC0：定时计数器/PWM0/ Buzzer 输出。

  TC1：定时计数器。

- 内置看门狗定时器，其时钟源由内部低速 **RC** 振荡器提供（**16KHz @2V**，**32KHz @3V**）

- **2** 个系统时钟

  内部高速时钟：RC 模式，高达 8MHz。
  内部低速时钟：RC 模式，16KHz(2V)，32KHz(3V)。

- 工作模式
  普通模式：高、低速时钟同时工作。
  低速模式：只有低速时钟工作。
  睡眠模式：高、低速时钟都停止工作。
  绿色模式：由 TC0 周期性的唤醒。

♦ 强大的指令系统

指令的长度为一个字。
大部分指令只需要一个时钟周期。
跳转指令 JMP 可在整个 ROM 区执行。
调用指令 CALL 可在整个 ROM 区执行。
查表指令 MOVC 可寻址整个 ROM 区执行。

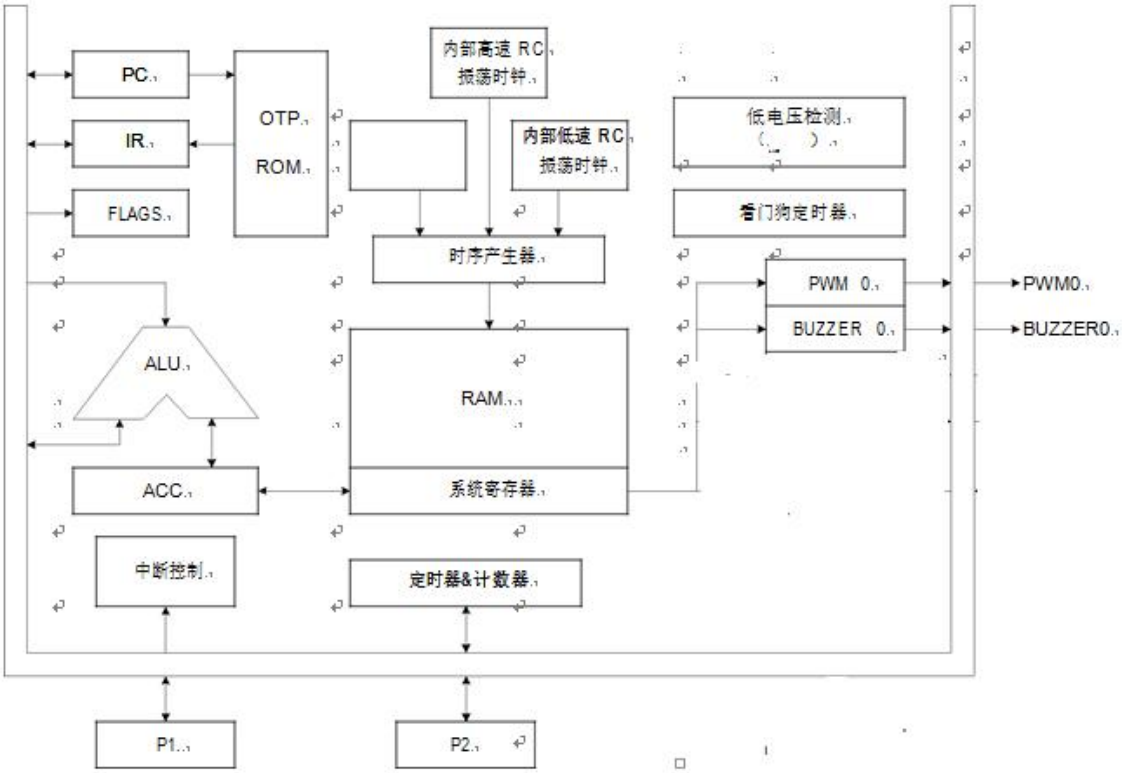
♦ 封装形式

♦ Fcpu（指令周期）
Fcpu = Fosc/2, Fosc/4, Fosc/8, Fosc/16,Fosc/32，Fosc/64

☞ 特性列表

单片机型号	ROM	RAM	堆栈	定时器		I/O	绿色模式	PWM	唤醒功能	封装形式
				TC0	TC1			Buzzer	引脚数目	
FC706	6K*16	512B	8	V	V	14	**V**	**1**	**13**	**P-DIP 16/SOP 16**

1.2 系统结构框图



1.3 引脚配置

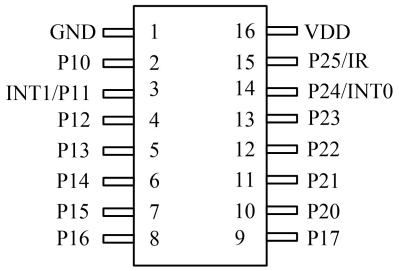


图 1 管脚分配

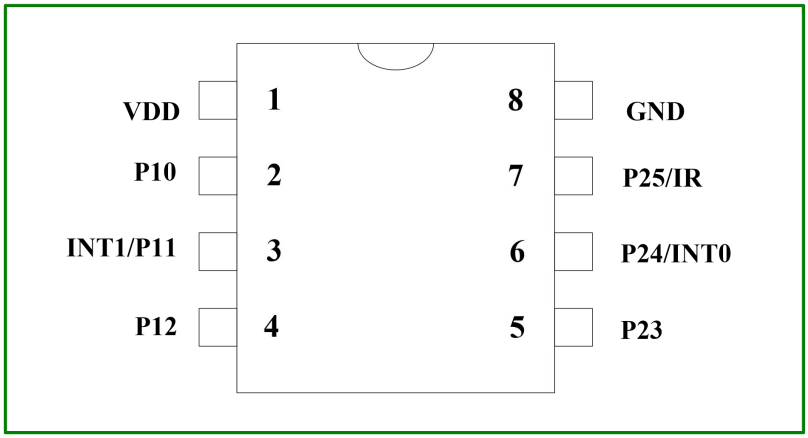


图 2 SOP8 脚位图

表 3-1 封装管脚说明

管脚名称	输入/输出	说　　明
VDD	I	电源正端（＋）输入端；
GND	I	电源负端（－）输入端；
P25/IR	I/O	双向输入输出脚，输入时施密特触发，内置上拉电阻； 编码输出端（可通过寄存器设置）； 红外小信号输入端（可通过寄存器设置）； T-KEY 扫描脚； 外部中断管脚
P24/INT0	I/O	双向输入输出脚，输入时施密特触发，内置上拉电阻； T-KEY 扫描脚； 外部中断管脚
P11/INT1	I/O	双向输入输出脚，输入时施密特触发，内置上拉电阻，可设置唤醒功能，可设置中断功能； T-KEY 扫描脚； 外部中断管脚

其他	I/O	双向输入输出脚，输入时施密特触发，内置上拉电阻，可设置唤醒功能，可设置中断功能； T-KEY 扫描脚；

注：I=input,O=output，I/O=input/output

2 中央处理器（CPU）

2.1 程序存储器（ROM）

☞ **ROM：**

1、有 13 位 PC，6K*16bit 的 OTP

2、复位地址 000h；

3、硬件中断地址 008h；

4、goto、call 指令可以全地址跳转。

程序存储器结构图

2.1.1 复位向量（0000H）

具有一个字长的系统复位向量（0000H）。

- 上电复位
- 看门狗复位

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。下面一段程序演示了如何定义 ROM 中的复位向量

➢ 例：定义复位向量。

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

```
                ORG     10H
START:                                  ; 用户程序起始地址。
                                        ; 用户程序。

                ENDP                    ; 程序结束。
```

2.1.2 中断向量（0008H）

中断向量地址为 0008H。一旦有中断响应，程序计数器 PC 的当前值就会存入堆栈缓存器并跳转到 0008H 开始执行中断服务程序。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 SZFC 的编程规则，有以下几点：
1、地址 0000H 的"JMP"指令使程序从头开始执行；2、地址 0008H 是中断向量；
3、用户的程序应该是一个循环。

2.1.3 查表

在 FC706 单片机中，对 ROM 区中的数据进行查找，寄存器 Y 指向所找数据地址的中间字节（bit8~bit15），寄存器 Z 指向所找数据地址的低字节（bit0~bit7）。执行完 MOVC 指令后，所查找数据低字节内容被存入 ACC 中，而数据高字节内容被存入 R 寄存器。

➢ **例：查找 ROM 地址为"TABLE1"的值。**

```
            B0MOV   Y, #TABLE1$M    ; 设置 TABLE1 地址高字节。
            B0MOV   Z, #TABLE1$L    ; 设置 TABLE1 地址低字节。
            MOVC                    ; 查表，R = 00H，ACC = 35H。

                                    ; 查找下一地址。
            INCMS   Z
            JMP     @F              ; Z 没有溢出。
            INCMS   Y               ; Z 溢出（FFH → 00），→ Y=Y+1
            NOP

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

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

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

➢ **例：宏 INC_YZ。**

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

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

➢ **例：通过"INC_YZ"对上例进行优化。**

```
            B0MOV   Y, #TABLE1$M    ; 设置 TABLE1 地址中间字节。
            B0MOV   Z, #TABLE1$L    ; 设置 TABLE1 地址低字节。
            MOVC                    ; 查表，R = 00H，ACC = 35H。

            INC_YZ                  ; 查找下一地址数据。

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

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

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

➢ **例：由指令 B0ADD/ADD 对 Y 和 Z 寄存器加 1。**

```
                B0MOV     Y, #TABLE1$M      ; 设置 TABLE1 地址中间字节。
                B0MOV     Z, #TABLE1$L      ; 设置 TABLE1 地址低字节。

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

                B0BTS1    FC                ; 检查进位标志。
                JMP       GETDATA           ; FC = 0。
                INCMS     Y                 ; FC = 1。
                NOP
GETDATA:
                MOVC                        ; 存储数据，如果 BUF = 0，数据为 0035H。
                                            ; 如果 BUF = 1，数据=5105H。
                                            ; 如果 BUF = 2，数据=2012H。

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

2.1.4 跳转表

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

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

➢ **例：跳转表。**

```
                ORG       0100H             ; 跳转表从 ROM 前端开始。

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

FC706 单片机提供一个宏以保证可靠执行跳转表功能，它会自动检测 ROM 边界并将跳转表移至适当的位置。但采用该宏程序会占用部分 ROM 空间。

➢ **例：如果跳转表跨越 ROM 边界，将引起程序错误。**

```
@JMP_A          MACRO     VAL
                IF        (($+1) !& 0XFF00) !!= (($+(VAL)) !& 0XFF00)
                JMP       ($ | 0XFF)
                ORG       ($ | 0XFF)
                ENDIF
                B0ADD     B0PCL, A
                ENDM
```

✷ 注：“VAL”为跳转表列表中列表个数。

➢ **例：宏“MACRO3.H”中，“@JMP_A”的应用。**

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

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

➢ **例：“@JMP_A”运用举例**

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

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

2.1.5 CHECKSUM 计算

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

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

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

Y_ADD_1:
                INCMS   Y
                NOP
                JMP     @B                  ; 跳转到 Checksum 计算。
CHECKSUM_END:
                …
                …
END_USER_CODE:                              ; 程序结束。
```

2.2 数据存储器（RAM）

☞ **RAM：512 字节**

	Address	RAM location	
BANK 0	000h " " " " " 07Fh	*General purpose area*	BANK 0
	080h " " " " " 0FFh	*System register*	80h~FFh of Bank 0 store system registers (128 bytes).
		End of bank 0 area	
BANK1	100h " " " " " 1FFh	*General purpose area*	BANK1
BANK2	200h " " " " " 27Fh	*General purpose area*	BANK2

2.2.1 系统寄存器

2.2.1.1 系统寄存器列表

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	/TO	/PD	LVDF			C	DC	Z	R/W	PFLAG
087H							RBANKS1	RBANKS0	R/W	RBANK
090H	由 T1LTS 选择的信号源 ↓下降沿读到 T1 低 8 位								R/W	TILTRL
091H	由 T1LTS 选择的信号源 ↓下降沿读到 T1 高 8 位								R/W	TILTRH
092H	由 T1LTS 选择的信号源 ↓上升沿读到 T1 低 8 位								R	TILTFL
093H	由 T1LTS 选择的信号源 ↓上升沿读到 T1 高 8 位								R/W	TILTFH
094H	T0D7	T0D6	T0D5	T0D4	T0D3	T0D2	T0D1	T0D0	R/W	TOD
095H	自动装载寄存器								R/W	TOR
096H	LVDEN	LVDS2	LVDS1	LVDS0	T1LTS	WDTP2	WDTP1	WDTP0	R/W	OPTION
097H	SSEL0	DKWE	T1WDS	T1WDL	POWER	IRCON	DSEL1	DSEL0	R/W	PTCON
098H									R/W	MTPCON1
0A3H									R/W	P1TM
0A4H									R/W	P2TM
0A5H	OPAE	SYNEN	IRINO	IRINT 溢出时间控制			OPAI1	OPAI0	R/W	OPA
0A6H									W	P2W
0A7H									W	MTPADH
0A8H									R/W	MTPADL
0B4H									R/W	MTPDB
0B6H									R/W	MTPCON
0BFH				P11G1	P11G0	P24G1	P24G0		R/W	PEDGE
0C0H									R/W	P1W
0C1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C2H			P25M	P24M	P23M	P22M	P21M	P20M	R/W	P2M
0C8H	MTPIF			RBIF	T1IF	T0IF	INT1IF	INT0IF	R/W	INTRQ
0C9H	MTPIE			RBIE	T1IE	T0IE	INT1IE	INT0IE	R/W	INTEN
0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CEH									R/W	PCL
0CFH									W	PCH
0D1H									W	P1
0D2H									W	P2
0D8H	TR0	T0S2	T0S1	T0S0	T0PT1	T0PT0	T0CS	PWMOUT	R/W	T0M
0D9H									R/W	T0C
0DAH	TR1	TP1S2	TP1S1	TP1S0	TP1T1	TP1T0	T1GN	T1TRSEL	R/W	T1M
0DBH	定时/计数器 T0 低八位								R/W	T1CL
0DCH	定时/计数器 T0 高八位								R/W	T1CH
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	R/W	P1UR
0E2H			P25R	P24R	P23R	P22R	P21R	P20R	R/W	P2UR
0E7H									R/W	@YZ

2.2.1.2 系统寄存器的位定义

1、Y/Z（间接寻址寄存器）

寄存器 Y 和 Z 都是 8 位缓存器，主要用途如下：

a. 普通工作寄存器；

b. RAM 数据寻址指针@YZ；

c. 配合指令 MOVC 对 ROM 数据进行查表。

地址	名称	D7	D6	D5	D4	D3	D2	D1	D0
83H(R/W)	Z	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0
84H(R/W)	Y	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0
复位状态	BOR	x	x	x	x	x	x	x	x
	WDT	u	u	u	u	u	u	u	u

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

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

➢ 例：利用数据指针@YZ 对 RAM 数据清零。

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

CLR_YZ_BUF:
        CLR     @YZ         ; @YZ 清零。

        DECMS   Z           ;
        JMP     CLR_YZ_BUF  ; 不为零。

        CLR     @YZ
END_CLR:                    。
```

2、R（间接寻址寄存器）

位缓存器 R 主要有以下两个功能：

a. 作为工作寄存器使用；

b. 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。）

地址	名称	D7	D6	D5	D4	D3	D2	D1	D0
82H(R/W)	R	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0
复位状态	BOR	x	x	x	x	x	x	x	x
	WDT	u	u	u	u	u	u	u	u

3、PC（程序计数器）与堆栈

	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	-	-	PC13	PC12	PC11	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
After reset	-	-	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

PCH，PCL 都是可读可写的。

- ◆ 产生 6K*16bits 片内 OTP ROM 地址以获取对应程序指令代码。
- ◆ 复位后 PCL 的所有位均清零
- ◆ “GOTO”指令直接装载 PC 的 13 位。因此，“GOTO”指令跳转范围 6K 程序区间。
- ◆ “CALL”指令加载 PC 的 13 位，然后 PC+1 进入堆栈。
- ◆ “RET”（“RETLW K”,”RETI”，“RETI”）指令将栈顶数据装入 PC
- ◆ “ADDWF PCL,1”允许“A”的值加到当前 PC，PC 的高 5 位将自然进位
- ◆ “MOVWF PCL”允许将寄存器“A”的值装入 PC 的低 8 位，同时 PC 的高 5 位保持不变。
- ◆ “SUBWF PCL,1”允许“A”的值被减到当前 PC，但 PC 的高 5 位保持不变！
- ◆ 改变 PCL 内容的指令需要 2 个指令周期

4、PFLAG（状态寄存器）

地址	名称	D7	D6	D5	D4	D3	D2	D1	D0
86H(R/W)	PFLAG	/TO	/PD	LVDF	–	–	C	DC	Z
	W/R	R	R	R	–	–	W/R	W/R	W/R
复位状态	WDT	–	–	–	–	–	0	0	0

寄存器包含 ALU 的算术状态，RESET 状态。

C：进位/借位（加减法指令）

=1，有进位/无借位

=0，无进位/有借位

DC：半进位/半借位（加减法指令）

=1，有低 4 位进位/无低 4 位借位

=0，无低 4 位进位/有低 4 位借位

Z：零标志位

=1，逻辑操作结果为 0

=0，逻辑操作结果不为 0

LVDF：LVD 电压检测标志（电压点由 OPTION<7:4>确定）

=1，电源电压小于设定电压点

=0，电源电压大于设定电压点

/PD：掉电标志位

=1，系统上电或执行 CLRWDT 指令后

=0，执行 SLEEP 指令后

/TO：时间溢出标志位

=1， 系统上电或执行 CLRWDT 或执行 SLEEP 指令后

=0，看门狗溢出后

5、RBANK（RAM 页面控制寄存器）

地址	名称	D7	D6	D5	D4	D3	D2	D1	D0
87H(R/W)	RBANK	-	-	-	-	-	-	RBANKS1	RBANKS0
复位状态	BOR	-	-	-	-	-	-	0	0
	WDT	-	-	-	-	-	-	0	0

RABNKS0：RAM 页面控制位

=1，选择 BANK1

=0，不选择 BANK1

RABNKS1：RAM 页面控制位

=1，选择 BANK2

=0，不选择 BANK2

6、OPTION

地址	名称	D7	D6	D5	D4	D3	D2	D1	D0
96H	OPTION	LVDEN	LVDS2	LVDS1	LVDS0	T1LTS	WDTP2	WDTP1	WDTP0

LVDEN: LVD 检测使能位。

0：无效

1：有效。

LVDS<2:0>：LVD 电压检测变换选择位

111：　1.8v-1.83v

110：　1.98v-2.05v

101：　2.05v-2.15v

100:　2.12v-2.25v

011:　2.20v-2.34v

010：　2.26v-2.43v

001：　2.40v-2.62v

000：　2.47v-2.72v

注：如果电源电压低于选定的只，status<5>(LVDF)会置 1，但不具备锁存功能。即如果下一刻电源电压又高于选定的值，则 LVDF 又会等于 0

T1LTS:　T1 捕捉器的源选择端。

0：IRINS

1：IRINT

WDTP<2:0>：WDT 溢出时间选择位

WDTP	WDTP1	WDTP0	WDT 的分频比
0	0	0	1:1(8ms)
0	0	1	1:2
0	1	0	1:4
0	1	1	1:8
1	0	0	1:16
1	0	1	1:32
1	1	0	1:64
1	1	1	1:128

7. PTCON

097H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PTCON	SSEL0	DKWE	T1WDS	T1WDL	POWER	IRCON	DSEL1	DSEL0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	-							

SSEL0: T 型阵列管脚扫描周期选择

0 8ms

1 16ms

DKWE:

0：矩形键盘唤醒

1：T 型键盘唤醒

T1WDS：T1 延时唤醒时间

0：32*Fosc

1：64*Fosc

T1WDL：T1 是否延时唤醒

0：不延迟

1：延迟,延时时间由 T1WDS 控制。

POWER:在使用大电流驱动时，电源波动较大，此项是优化电源电压波形。

=0：优化

=1：不优化（使用时需要加较大电容值的电容并接在电源地上）

IRCON:

0: I/O

1: 输出口，NMOS 开漏。驱动能力由 DSEL<1:0>选择。

DSEL<1:0>：IR 端低电平驱动的驱动能力选择

11 100mA
10 200mA
01 300mA
00 500mA

8.MTP控制器

8.1 MTPCON(MTP控制寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
B6h (r/w)	MTPCON			PG8B	BLPL2	BLPL1	BLPL0	DELAY1	DELAY0

DELAY<1:0>：擦除和编程延时选择。

DELAY<1:0>：	选择编程时	擦除时
00：	90us	90ms
01：	100us	100ms
10：	110us	110ms
11：	120us	120ms

注：标准选择是01值，即100us编程 和100ms擦除。

PG8B: 编程时同时编程8bit数据。

0：每次编程1个。哪一个由BLPL<2:0>确定。

1：编程每次一个byte。

注：选择按位编程后，则最低编程电压会达到1.8v。
但每个byte需要编程8次。

BLPL<2:0>：选择编程位

000： bit0

001： bit1

010： bit2

011： bit3

100： bit4

101： bit5

110： bit6

111： bit7

8.2 MTPCON1(MTP控制寄存器1)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
98h (r/w)	MTPCON1	-	-	-	-	MTP_CPE	MTP_STE	MTP_PG	MTP_RD

Bit7~bit4：读时数据不定。

MTP_CPE: MTP的chip擦除使能。

0：擦除完成。

1：擦除启动，擦完后自动清0

MTP_STE: MTP的sector擦除使能。

0：擦除完成。

1：擦除启动，擦完后自动清0

注：16个byte为一页，共32页，每页地址由
MTPADH,MTPADL的高5位地址确定。

MTP_PG: MTP的PG使能。

0：编程完成。

1：编程启动，擦完后自动清0

MTP_RD: MTP的read使能。

0：未读。

1：read启动，自动清0。此标志位不用查询，置1后，可以直接读MTPDB寄存器的值。

8.3 MTPADR(MTP地址寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
A7h (r/w)	MTPADH	-	-	-	-	-	-	-	MTPADH0
A8h (r/w)	MTPADL	MTPADL7	MTPADL6	MTPADL5	MTPADL4	MTPADL3	MTPADL2	MTPADL1	MTPADL0

Bit8:BIT0: 内置MTP地址，地址000H~1FFH共512字节

8.4 MTPDB(MTP数据寄存器)

地址Bank1	名称	B7	B6	B5	B4	B3	B2	B1	B0
B4h (r/w)	MTPDB	MTPDB7	MTPDB6	MTPDB5	MTPDB4	MTPDB3	MTPDB2	MTPDB1	MTPDB0

MTP数据输入寄存器，在烧写内置MTP前将要写的数据写入MTPDB寄存器中。读MTP时，用MOV指令直接读入ACC中.

MTP操作例程：

```
                MOV  A,#40h
                MOV  MTPCON,A  ; 延时100us，100ms，按bit编程。
                CLR  MTPADH
                MOV  A,#02h
                MOV MTTADL     ;对MTP的第二个地址编程。
;==========================================================================================
;页擦除，当前值下为第一页。擦除后，可以全部写完后再下次擦除。更新数据时也可擦除。
 LOOP_SECTCE:
                BSET  MTP_STE
 CHECK_STE:
                BTS0   MTP_STE
                JMP    CHECK_STE   ;需要100ms擦除，小心wdt复位
;==========================================================================================
;编程

                MOV  A,#08h
                MOV  TEMP,A    ;8次循环，用作8个bit的烧写。
                MOV  A,#55H
                MOV  MTPDB,A  ;赋值要写入的数据

LOOP_PG:
                BSET   MTP_PG    ;写第一遍
CHECK_PG:
                BTS0     MTP_PG
                JMP      CHECK_PG
                BSET   MTP_PG    ;写第二遍
CHECK_PG_2:
                BTS0     MTP_PG
                JMP      CHECK_PG_2
                DECMS  TEMP,1
                JMP      NEXT_PG1
                JMP      PG_END
NEXT_PG1：
                MOV      A,#0x04
                ADD      MTPCON,A  ;待编程位改变
                JMP      LOOP_PG
PG_END:
;==========================================================================================
;READ
               BSET   MTP_RD        ;启动读

               MOV    A,MTP_DB      ;MTP的数据送入A;
               MOV   TEMP1，A
```

3 复位

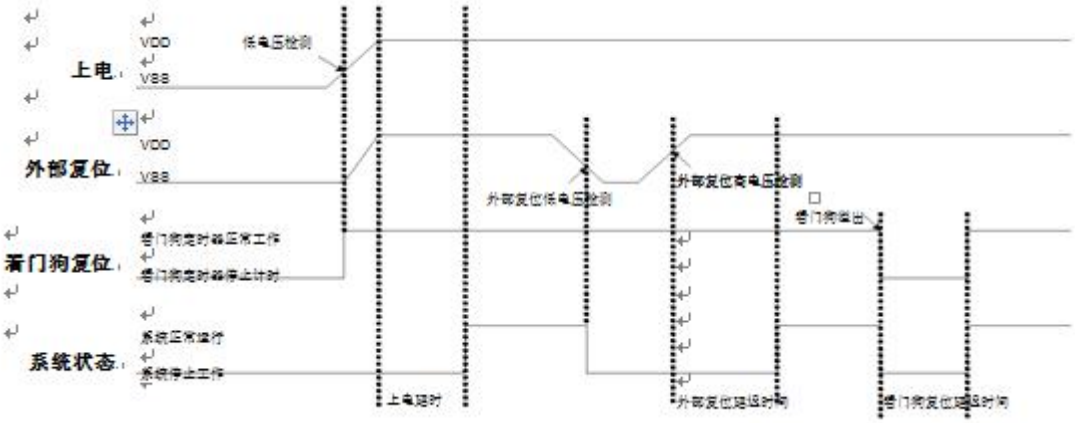
3.1 概述

FC706 有以下几种复位方式：

- 上电复位；
- 看门狗复位；
- 掉电复位；

上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器 PC 清零。复位结束后，系统从向量 0000H 处重新开始运行。

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。



3.2 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。

3.3 看门狗复位

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

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

看门狗定时器应用注意事项：

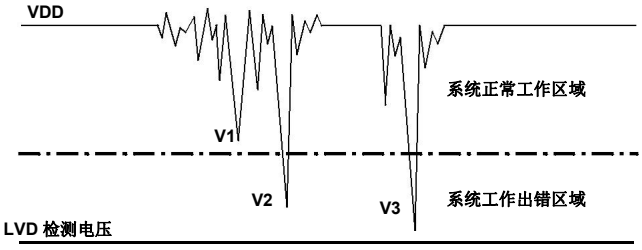
- 对看门狗清零之前，检查 I/O 口的状态和 RAM 的内容可增强程序的可靠性；
- 不能在中断中对看门狗清零，否则无法侦测到主程序跑飞的状况；
- 程序中应该只在主程序中有一次清看门狗的动作，这种架构能够最大限度的发挥看门狗的保护功能。

✷ **注：关于看门狗定时器的详细内容，请参阅“看门狗定时器”有关章节。**

3.4 掉电复位

3.4.1 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。

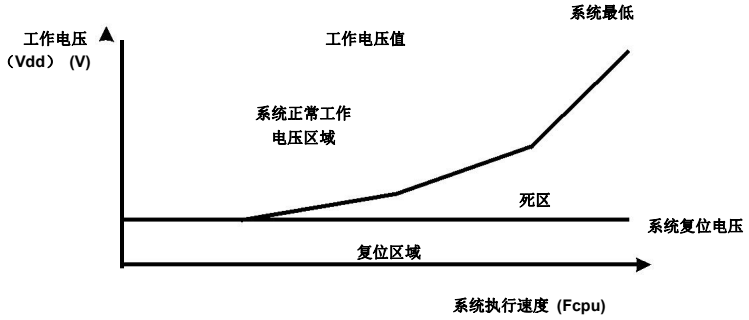
AC 运用中：

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。

3.4.2 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。

3.4.3 掉电复位性能改进

如何改善系统掉电复位性能，有以下几点建议：

- LVD 复位；
- 看门狗复位；
- 降低系统工作速度；

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。

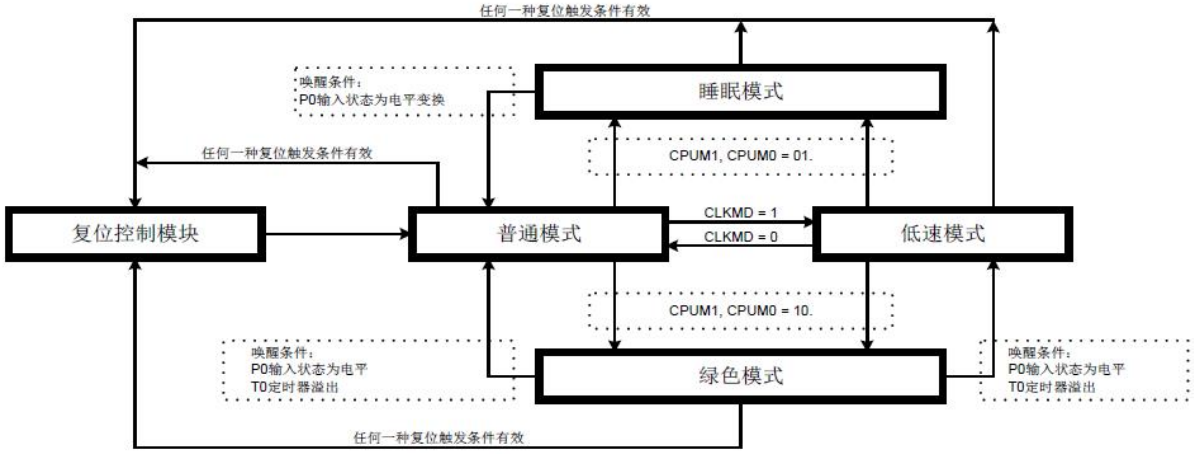
4 系统工作模式

4.1 概述

FC706 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。

工作模式控制框图



工作模式	普通模式	低速模式	绿色模式	睡眠模式
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
TC0	TC0EN控制	TC0EN控制	TC0EN控制 仅PWM/BUZZE有效	停止
WDT	WDTEN控制	WDTEN控制	WDTEN控制	WDTEN控制
内部中断	有效	有效	TC0	无效
外部中断	有效	有效	全部有效	无效
唤醒	—	—	端口变化，按键触发 WDT溢出，外部中断	端口变化，按键触发 WDT溢出，外部中断

注：绿色模式唤醒新增加 T1 定时器溢出唤醒

4.2 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。

4.3 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。

4.4 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 的电平变换触发唤醒。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 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 电平变换触发唤醒。

4.5 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 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 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：SZFC 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。

4.6 工作模式控制宏

FC706 提供工作模式控制宏以方便系统工作模式的切换。

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

➢ 例：从普通模式/低速模式切换进入睡眠模式。

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

➢ 例：从普通模式切换进入低速模式。

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

➢ 例：从低速模式切换进入普通模式（外部高速振荡器停止工作）。

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

➢ 例：从普通/低速模式切换进入绿色模式。

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

➢ 例：从普通/低速模式切换进入绿色模式，并使能 **TC0** 唤醒功能。

```
; 设置定时器 TC0 的唤醒功能。
        B0BCLR      FTC0IEN     ; 禁止 TC0 中断。
        B0BCLR      FTC0ENB     ; 禁止 TC0 定时器。
        MOV         A,#20H      ;
        B0MOV       TC0M,A      ; 设置 TC0 时钟= Fcpu / 64。
        MOV         A,#64H
        B0MOV       TC0C,A      ; 设置 TC0C 的初始值=64H（设置 T0 间隔值 = 10 ms）。
        B0BCLR      FTC0IEN     ; 禁止 TC0 中断。
        B0BCLR      FTC0IRQ     ; 清 TC0 中断请求。
        B0BSET      FTC0ENB     ; 使能 TC0 定时器。

; 进入绿色模式。
        GreenMode               ; 直接宣告“GreenMode”宏。
```

4.7 系统唤醒

4.7.1 概述

睡眠模式和绿色模式下，系统并不执行程序。唤醒触发信号可以将系统唤醒进入普通模式或低速模式。唤醒触发信号包括：外部触发信号（P0 的电平变换）和内部触发（TC0 定时器溢出）。

- 从睡眠模式唤醒后只能进入普通模式，且将其唤醒的触发只能是外部触发信号（P0 电平变化）；
- 如果是将系统由绿色模式唤醒返回到上一个工作模式（普通模式或低速模式），唤醒触发信号可以是外部触发信号（P0 电平变换）和内部触发信号（TC0 溢出）。

4.7.2 唤醒时间

系统进入睡眠模式后，高速时钟振荡器停止运行。把系统从睡眠模式唤醒时，单片机需要等待一段时间以等待振荡电路稳定工作，等待的这一段时间就称为唤醒时间。唤醒时间结束后，系统进入普通模式。

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

外部高速振荡器的唤醒时间的计算如下：

唤醒时间 = 0.128ms + 高速时钟启动时间

5 中断

5.1 概述

FC706 提供 6 个中断源：2 个内部中断（TC0/TC1）和 2 个外部中断（INT0/INT1），IR，mtp 中断外部中断可以将系统从睡眠模式中唤醒进入高速模式，在返回到高速模式前，中断请求被锁定。一旦程序进入中断，寄存器 STKP 的位 GIE 被硬件自动清零以避免响应其它中断。系统退出中断后，硬件自动将 GIE 置“1”，以响应下一个中断。中断请求存放在寄存器 INTRQ 中。

✷ **注：程序响应中断时，必须开启全局中断控制位 GIE。**

5.2 中断请求使能寄存器 INTEN

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

0C9H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTEN		-	-	RBIE	T1IE	T0IE	INT1IE	INT0IE
R/W		-	-	R/W	R/W	R/W	R/W	R/W
复位		-	-	0	0	0	0	0

RBIE: RB 中断使能

0：中断无效

1：中断使能

T1IE: T0 中断使能

0：中断无效

1：中断使能

T0IE: T0 中断使能

0：中断无效

1：中断使能

INT1IE: INT1 中断使能

0：中断无效

1：中断使能

INT0IE: INT0 中断使能

0：中断无效

1：中断使能

5.3 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	-	-	-	RBIF	T1IF	T0IF	INT1IF	INT0IF
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
复位	-	-	-	0	0	0	0	0

T0IF: T0 中断标志

0：中断无效

1：中断标志

T1IF: T0 中断标志

0：中断无效

1：中断标志

INT1IF: INT1 中断使能

0：中断无效

1：中断标志

INT0IF: INT0 中断使能

0：中断无效

1：中断标志

RBIF: RB 中断使能

0：中断无效

1：中断标志

5.4 GIE 全局中断

只有当全局中断控制位 GIE 置“1”的时候程序才能响应中断请求。 一旦有中断发生，程序计数器（PC）指向中断向量地址（ORG8），堆栈层数加 1。

0DFH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
STKP	GIE					STKPB2	STKPB1	STKPB0
R/W	R/W					R/W	R/W	R/W
复位	0					1	1	1

Bit7 GIE: 全局中断控制位

0：禁止全局中断

1：允许全局中断

Bit2-Bit0 STKPB2-STKPB0：堆栈指针

入栈时，将 PC 存入当前指针的堆栈寄存器，然后 STKP-1(初始为全 1)；出栈时 STKP+1，然后将堆栈寄存器的内容写入 PC.

✷ **注：在所有中断中，GIE 都必须处于使能状态。**

5.5 PEDGE

0BFH	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
PEDGE	-			P11G1	P11G0	P24G1	P24G0	
R/W	-			R/W	R/W	R/W	R/W	
复位	-			0	0	0	0	

Bit2:1 INT0 中断边沿控制位

00: 保留

01：上升沿

10：下降沿

11：上升/下降沿

Bit4:3 INT1 中断边沿控制位

00: 保留

01：上升沿

10：下降沿

11：上升/下降沿

5.6 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
```

6 I/O 口

6.1 I/O 口模式

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
C1H(r/w)	P1M	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
C2H(r/w)	P2M	-	-	P25M	P24M	P23M	P22M	P21M	P20M

PnM[7:0]：Pn 模式控制位（n=1~2）

0 = 输入模式；

1 = 输出模式。

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
E1H(r/w)	P1UR	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R
E2H(r/w)	P2UR	-	-	P25R	P24R	P23R	P22R	P21R	P20R

PnUR[7:0]：Pn 上拉控制位（n=1~2）

0 = 无上拉；

1 = 有上拉。

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
A3H(r/w)	P1TM								
A4H(r/w)	P2TM	-	-						

PnTM[7:0]：Pn 扫描控制位（n=1~2）

0 = IO；

1 =T 型键盘。

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
C0H(r/w)	P1W	唤醒寄存器							
A6H(r/w)	P2W	唤醒寄存器							

PnW[7:0]：P1，2 口唤醒控制位

0 = 不；

1 =能。

P1，P2 端口寄存器

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
D1H(r/w)	P1	P17	P16	P15	P14	P13	P12	P11	P10
D2H(r/w)	P2	-	-	P25	P24	P23	P22	P21	P20

✷ 注：

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

7 定时器

7.1 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 = 256 /内部低速振荡器周期（sec）*分频系数

看门狗定时器的 3 种工作模式由编译选项“WatchDog”控制：

- 不分频时 wdt 溢出时间为 8ms。
- **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
R/W	W	W	W	W	W	W	W	W
复位	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
```

7.2 定时/计数器 TC0

7.2.1 概述

8 位二进制定时/计数器具有基本定时器、事件计数器、PWM 和 Buzeer 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 T0C、T0M、T0R、TOD 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 还内置周期/占空比可编程控制的 PWM 功能，PWM 的周期和分辨率由 T0M 和 T0R 寄存器控制。TC0 还内置 Buzzer 功能，以输出 TC0/2 信号。TC0 支持自动重装功能。TC0 溢出时，T0R 的值自动装入 T0C。TC0 内置绿色模式唤醒功能，由 TC0GN 控制。

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，T0R 寄存器和 T0M 寄存器的 ALOAD0 和 TC0OUT 位控制占空比/周期；
- **Buzzer 输出：**Buzzer 输出信号为 TC0 间隔时间的 1/2 周期；
- **绿色模式功能：**TC0 溢出时，TC0 内置绿色模式唤醒功能，由 TC0GN 控制。

7.2.2 T0（定时/计数器寄存器）

1. T0（定时/计数器寄存器）

地址	名称	D7	D6	D5	D4	D3	D2	D1	D0
0D9H	T0C	T0 的 8 位寄存器数据							
0D8H	T0M	T0 模式控制位							
095H	T0R	T0 load							
094H	TOD	T0 的 PWM 高电平时间							

2. T0M 寄存器

地址	名称	D7	D6	D5	D4	D3	D2	D1	D0
0D8H	T0M	TR0	T0S2	T0S1	T0S0	T0PT1	T0PT0	T0CS	PWMOUT
复位状态	BOR	0	0	0	0	0	0	0	0
	WDT	0	0	0	0	0	0	0	0

PWMOUT:

0：PWM 不能输出(T0 变成 16 位的，此时 TOD 变为计数器的高 8 位)；

1：PWM 输出；相应端口必须设置为输出模式。

T0PT1，T0PT0：T0 时钟源选择位

T0PT1，T0PT0	T0CS=0	T0CS=1
00	Fhosc	Fhosc
01	Flosc	Flosc
10	Fcpu	Fcpu
11	IRINS	INT0

T0S[2:0]：T0 分频选择位。

T0S[2:0]	分频系数
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

TR0：T0 启动控制位。

0 = 禁止 T0 定时器；

1 = 开启 T0 定时器。

T0R 自动装载寄存器

地址	名称	D7	D6	D5	D4	D3	D2	D1	D0
095H	T0R	自动装载寄存器							

当T0溢出时，T0R的值自动装入T0中。这样，用户在使用的过程中就不需要在中断中重新赋值。T0为双重缓存器结构。若程序对T0R进行了修改，那么修改后的T0R值先被暂存在T0R的第一个缓存器中，直到当前T0溢出后，才被真正存入T0R缓存器中，从而避免T0中断时间出错以及PWM。

当启动T0时，T0R会自动装载。

T0D PWM 低电平占空比控制寄存器

名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
T0D	T0D7	T0D6	T0D5	T0D4	T0D3	T0D2	T0D1	T0D0

T0R控制PWM的周期及分辨率，T0D控制PWM低电平占空比，当T0C=T0D时，PWM翻转为高，完成低电平的占空比。

PWM

PWM信号输出到PWMOUT。PWM信号靠T0C,T0R,T0D的比较产生，当CREN为1或者T0溢出时，PWM输出低电平，为PWM的初始态，此时T0C重新载入T0R寄存器的值，T0R决定PWM的周期及分辨率。T0开始计数，当T0CD等于T0D时，PWM翻转为高电平，T0继续计数，当T0溢出时，一个PWM周期完成，PWM重新自动载入T0R的值并且输出低电平。

7.2.3 T1（定时/计数器寄存器）

地址	名称	D7	D6	D5	D4	D3	D2	D1	D0
0DAH	T1M	定时/计数器 T1 模式寄存器							
0DBH	T1C_L	定时/计数器 T1 低八位							
0DCH	T1C_H	定时/计数器 T1 高 8 位							

1. 概述

16位普通定时/计数器，内部时钟来自Fcpu，当T0C 从FFFFH 溢出到0000H 时，T0C 在继续计数的同时产生一个溢出信号T1IF，中断使能T1IE有效时，触发T0 中断请求。

2. T1C 寄存器

0DBH	T1C_L	定时/计数器 T0 低八位
0DCH	T1C_H	定时/计数器 T0 高 8 位

16位普通定时/计数器，内部时钟来自Fcpu、Fosc、INT1或低速时钟的2分频，当T1从FFFFH 溢出到0000H 时，T1C在继续计数的同时产生一个溢出信号T1IF，中断使能T1IE有效时，触发T0中断请求。

3.T1M 寄存器

地址	名称	D7	D6	D5	D4	D3	D2	D1	D0
0DAH	T1M	TR1	TP1S2	TP1S1	TP1S0	T1PT1	T1PT0	T1GN	T1TRSEL

TR1：T1 使能控制。

0 = 禁止；

1 = 使能。

TP1S[2:0]：T1 分频选择位。

TP1S[2:0]：T1分频选择位。TP1S[1:0]	分频系数
000	1
001	2
010	4
011	8
100	16
101	32
110	64
111	128

4. T1PT1，T1PT0：T1 时钟源选择位

T1PT1，T1PT0	
00	Fhosc
01	T0 溢出
10	Fcpu
11	INT1

T1GN：T1 绿色模式溢出唤醒使能控制。

0 = 禁止；

1 = 使能。（可设置延迟唤醒）

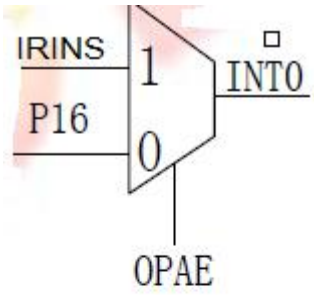
T1TRSEL：T1 开启使能控制选择。

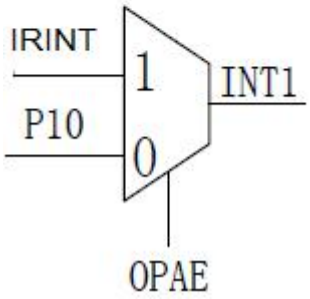
0 = TR1 控制；

1 = IRINT 下降沿开启 T1 定时，仅 OPAE 和 SYNEN 同时为 1 有效。

T1LATCH（T1 捕捉寄存器）

地址	名称	D7	D6	D5	D4	D3	D2	D1	D0
	T1LTFL	由 T1LTS 选择的信号源 ↓ 下降沿读到 T1 低 8 位							
	T1LTFH	由 T1LTS 选择的信号源 ↓ 下降沿读到 T1 高 8 位							
	T1LTRL	由 T1LTS 选择的信号源 ↑ 上升沿读到 T1 低 8 位							
	T1LTRH	由 T1LTS 选择的信号源 ↑ 上升沿读到 T1 高 8 位							


IRINS
P16
1
0
INT0
OPAE


IRINT
P10
1
0
INT1
OPAE

8 OPA IR 小信号放大电路

8.1 OPA 寄存器

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
0A5H	OPA	OPAE	SYNEN	IRINO	IRINT 溢出时间控制			OPAI1	OPAI0

1、OPAE 放大电路使能

OPAE=0:无效
OPAE=1：有效。IR 脚需置为输入模式，或者在 IRCON=1 时，输出高。
在 OPAE 有效时，读 P25 时，读的值是 IRIN 的值，读 P24 时，读的值是 IRINT 的值。

2、SYNEN:

同步使能，1 有效。为 1 时，IRINS 和 IRINT 才能有效。SYNEN 最好滞后 OPAE 几 us，以消除干扰。

3. IRINO

IRINO=0:无效
IRINO=1：有效(必须 IRXEN 同时有效)，P12 输出和 IR 接收到的波形一致。

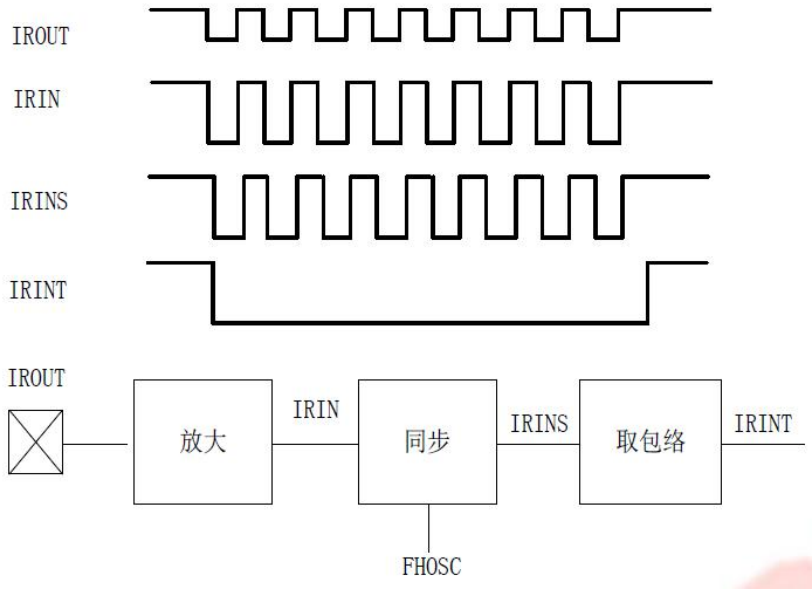
4、Bit4~bit2:IRINT 溢出时间控制。

000:16us　001:24us　010:32us　011：40us

100:48us　101:56us　110:64us　111：96us

5、OPAI<1:0> 运放偏置电流设置。

00 01 10 11 值越大，放大倍数 1，2，4，8，值越大，能力越强。一般选择 00 即可。



学习方法：
1、 开启 OPAE，并延时 1us 开 SYNEN，设置合适的放大倍数（一般默认即可）
2、设置 T0 和 T1。T0 计数接收到的脉冲个数，T1 定时，并利用 IRINT 的上升、下降沿捕捉该时刻的时间节点。保存 T1 的值和 T0 的值。
3、到定时周期后（如学习 100ms），利用第一帧的 T1 的捕捉值和 T0 的脉冲个数，计算接收到的频率周期。
4、分析后，得到一帧码的频率，码型和周期等特征，并存入 MTP。
5、发射学习到的码型。

9 配置 config

Word0<9;8>	security	其他	enable	加密
		0	disable	
Word0<7:6>	IRcomp	00	灵敏度最高	IR 小信号放大电路的灵敏度调整
		01	较高	
		10	较高	
		11	低	
Word0<5>.	WDT_SEL<1>	WDT_SEL<1;0>=00	Always_on	WDTE
Word0<4>.	WDT_SEL<0>	WDT_SEL<1;0>=01	enable	
		WDT_SEL<1;0>=11	disable	
Word0<3>	FCPU<2>	FCPU<2:0>=000	FCPU=FOSC/2	指令周期选择
Word0<2>	FCPU<1>	FCPU<2:0>=001	FCPU=FOSC/4	
Word0<1>	FCPU<0>	FCPU<2:0>=010	FCPU=FOSC/8	
		FCPU<2:0>=011	FCPU=FOSC/16	
		FCPU<2:0>=100	FCPU=FOSC/32	
		FCPU<2:0>=101	FCPU=FOSC/64	
Word0<0>	POWER_LOW	0	Disable	低功耗模式，主频小于 1MHZ 时，可以选用
		1	enable	

10 指令集

指令	指令格式	描述	C	DC	Z	周期
M O	MOV A,M	A ← M。	-	-	√	1
	MOV M,A	M ← A。	-	-	-	1
	B0MOV A,M	A ← M (bank 0)。	-	-	√	1

V	B0MOV	M,A	M (bank 0) ← A。	-	-	-	1
E	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I。（M 仅适用于系统寄存器 R、Y、Z、RBANK、PFLAG。）	-	-	-	1
	XCH	A,M	A ←→M。	-	-	-	1
	B0XCH	A,M	A ←→M (bank 0)。	-	-	-	1
	MOVC		R, A ← ROM [Y,Z]。	-	-	-	2
	ADC	A,M	A ← A + M + C，如果产生进位则 C=1，否则 C=0。	√	√	√	1
A	ADC	M,A	M ← A + M + C，如果产生进位则 C=1，否则 C=0。	√	√	√	1
R	ADD	A,M	A ← A + M，如果产生进位则 C=1，否则 C=0。	√	√	√	1
I	ADD	M,A	M ← A + M，如果产生进位则 C=1，否则 C=0。	√	√	√	1
T	B0ADD	M,A	M (bank 0) ← M (bank 0) + A，如果产生进位则 C=1，否则 C=0。	√	√	√	1
H	ADD	A,I	A ← A + I，如果产生进位则 C=1，否则 C=0。	√	√	√	1
M	SBC	A,M	A ← A - M - /C，如果产生借位则 C=0，否则 C=1。	√	√	√	1
E	SBC	M,A	M ← A - M - /C，如果产生借位则 C=0，否则 C=1。	√	√	√	1
T	SUB	A,M	A ← A - M，如果产生借位则 C=0，否则 C=1。	√	√	√	1
I	SUB	M,A	M ← A - M，如果产生借位则 C=0，否则 C=1。	√	√	√	
C	SUB	A,I	A ← A - I，如果产生借位则 C=0，否则 C=1。	√	√	√	1
	AND	A,M	A ← A 与 M。	-	-	√	1
L	AND	M,A	M ← A 与 M。	-	-	√	1
O	AND	A,I	A ← A 与 I。	-	-	√	1
G	OR	A,M	A ← A 或 M。	-	-	√	1
I	OR	M,A	M ← A 或 M。	-	-	√	1
C	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
	XOR	A,I	A ← A 异或 I。	-	-	√	1
	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)。	-	-	-	1
P	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)。	-	-	-	1
R	RRC	M	A ← M 带进位右移。	√	-	-	1
O	RRCM	M	M ← M 带进位右移。	√	-	-	1
C	RLC	M	A ← M 带进位左移。	√	-	-	1
E	RLCM	M	M ← M 带进位左移。	√	-	-	1
S	CLR	M	M ← 0。	-	-	-	1
S	BCLR	M.b	M.b ← 0。	-	-	-	1
	BSET	M.b	M.b ← 1	-	-	-	1
	B0BCLR	M.b	M(bank 0).b ← 0。	-	-	-	1
	B0BSET	M.b	M(bank 0).b ← 1。	-	-	-	1
	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
B	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
R	INCS	M	A ← M + 1，如果 A = 0，则跳过下一条指令。	-	-	-	1+ S
A	INCMS	M	M ← M + 1，如果 M = 0，则跳过下一条指令。	-	-	-	1+S
N	DECS	M	A ← M - 1，如果 A = 0，则跳过下一条指令。	-	-	-	1+ S
C	DECMS	M	M ← M - 1，如果 M = 0，则跳过下一条指令。	-	-	-	1+S
H	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
M	RET		子程序跳出指令，PC ← Stack。	-	-	-	2
I	RETI		中断处理程序跳出指令，PC ← Stack，使能全局中断控制位。	-	-	-	2
S	PUSH		进栈指令，保存 ACC 和工作寄存器。	-	-	-	1
C	POP		出栈指令，恢复 ACC 和工作寄存器。	√	√	√	1
	NOP		空指令，无特别意义。	-	-	-	1

注： **1.条件跳转指令的条件为真，则 S = 1，否则 S = 0**

11 电气特性

11.1 极限参数

Supply voltage (Vdd)..... - 0.3V ~ 3.6V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V Operating ambient temperature (Topr)
FC706..... .....0°C ~ + 70°C Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

11.2 电气特性

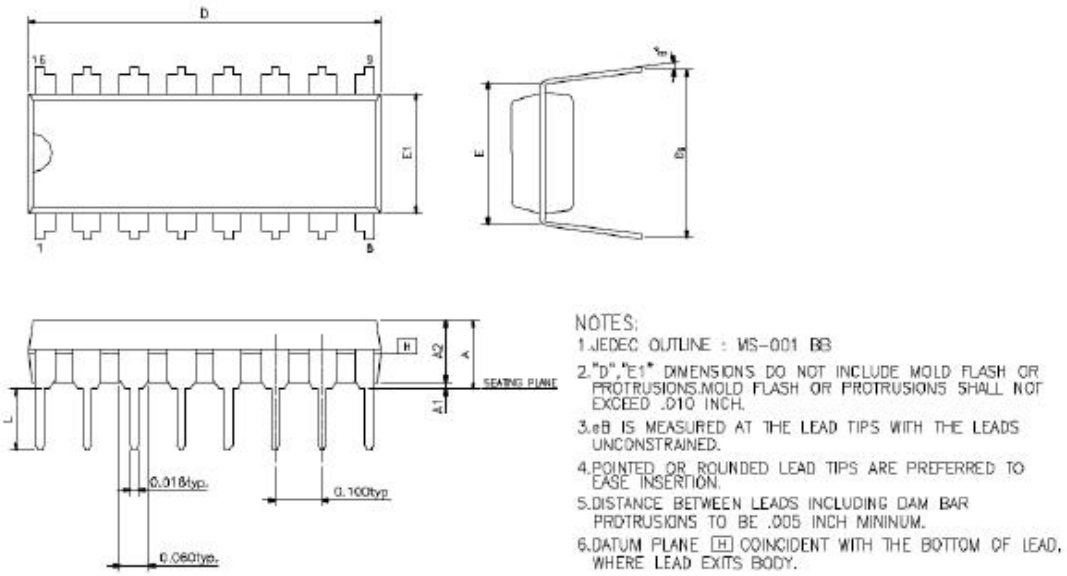
- **DC CHARACTERISTIC**

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

Characteristics	Symbol	Limit			Unit	Test Condition
		Min.	Typ.	Max.		
Operating Voltage1	VDD	1.6	-	3.6	V	FCPU=4MHz,For 2Batteries
Operating Voltage1	VDD	2.0	-	3.6	V	FCPU=8MHz,For 2Batteries
Operating Current	I_{OP}	-	1.5	3	mA	FCPU=8MHz@3.0V,no load
M-Type Key Standby Current	I_{MSTBY}	-	-	1.0	uA	VDD=3.6V
T-Type Key Standby Current	I_{TSTBY}	-	-	2.0	uA	VDD=3.6V,Key loading≤50pF
Input High Level	V_{IH}	0.7VDD	-	-	V	VDD=3.0V
Input Low Level	V_{IL}	-	-	0.3VDD	V	VDD=3.0V
Output High Level PB,PD	V_{OH}	0.8VDD	-	-	V	VDD=3.0V IOH=-6mA
Output Low Level PB,PD	V_{OL}	-	-	0.2VDD	V	VDD=3.0V IOL=16mA
Input Pull High Resistor PB,PD	R_H	100	125	170	Kohm	Pull High VDD=3.0V
Max PWM Driving Current	I_{PWM}	300	-	-	mA	VDD=3.0V,VRMT=3.0V PWMDRV0=1
LVR Active Voltage(by option)	V_{LVR}	1.8	-	1.83	V	LVDS=111
		1.98	-	2.05	V	LVDS=110
		2.05	-	2.15	V	LVDS=101
		2.12	-	2.25	V	LVDS=100
		2.20	-	2.34	V	LVDS=011
		2.26	-	2.43	V	LVDS=010
		2.40	-	2.62	V	LVDS=001
		2.47	-	2.72	V	LVDS=000

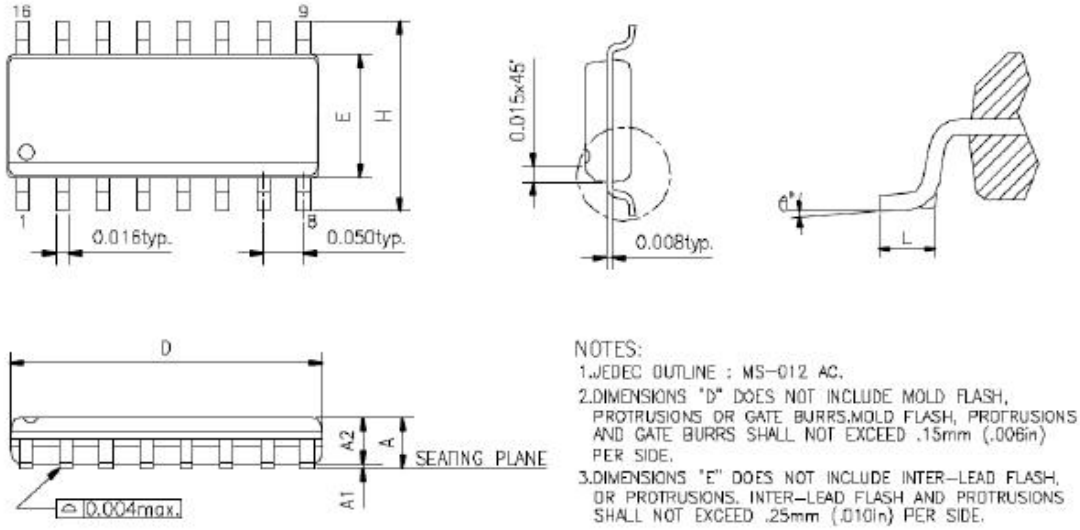
12.0 封装尺寸

12.1 PDIP 16 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-		0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.735	0.755	0.775	18.669	19.177	19.685
E	0.30 BSC			7.620 BSC		
E1	0.245	0.250	0.255	6.223	6.350	6.477
L	0.115	0.130	0.150	2.921	3.302	3.810
eB	0.335	0.355	0.375	8.509	9.017	9.525
θ°	0°	7°	15°	0°	7°	15°

12.2 SOP 16 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.053	-	0.069	1.346	-	1.753
A1	0.004	-	0.010	0.102	-	0.254
D	0.386	-	0.394	9.804	-	10.008
E	0.150	-	0.157	3.810	-	3.988
H	0.228	-	0.244	5.791	-	6.198
L	0.016	-	0.050	0.406	-	1.270
θ°	0°	-	8°	0°	-	8°