FX0004

用户参考手册

SZFC 8 位单片机

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

功能特色：

- **4K OTP 存储器**
- **384 Byte SRAM**
- **1*2 路 PWM，大驱动，带死区控制**
- **2*4 路 PWM，带死区控制**
- **I2C 协议**

修正记录

版本号	日期	内容
VER 1.0	2021 年 6 月	初版
VER 1.1	2021 年 10 月	与 FX3004 规格书区分开，此芯片没有触摸功能

目录

1 产品简介 6
1.1 功能特性 6
1.2 系统结构框图 8
1.3 引脚配置 9
1.4 引脚说明 10
2 中央处理器（CPU） 11
2.1 程序存储器（ROM） 11
2.1.1 复位向量（0000H） 11
2.1.2 中断向量（0008H） 11
2.1.3 查表 12
2.1.4 跳转表 13
2.1.5 CHECKSUM 计算 15
2.2 数据存储器（RAM） 16
2.2.1 系统寄存器 16
2.2.1.2 系统寄存器的位定义 16
2.2.2 累加器 18
2.2.3 程序状态寄存器 PFLAG 18
2.2.4 程序计数器 19
2.2.5 Y,Z 寄存器 19
2.2.6 R 寄存器 20
2.2.7 RAM 页面控制寄存器(RBANK) 20
2.2.8 OPTION 寄存器 20
2.3 堆栈 22
2.3.1 概述 22
2.3.2 堆栈寄存器 22
3 复位 23
3.1 概述 23
3.2 上电复位 23
3.3 看门狗复位 23
3.4 掉电复位 24
3.4.1 概述 24
3.4.2 系统工作电压 25
3.4.3 低电压检测 LVD 25
3.4.4 掉电复位性能改进 25
3.5 外部复位 26
4 系统时钟 27
4.1 概述 27
4.2 指令周期 FCPU 27
5 系统工作模式 28
5.1 概述 28
5.2 普通模式 29
5.3 低速模式 29
5.4 睡眠模式 29
5.5 绿色模式 30
5.6 工作模式控制宏 30

5.7 系统唤醒 31
5.7.1 概述 31
5.7.2 唤醒时间 31
5.8 OSCM 寄存器 31

6 中断 32

6.1 概述 32
6.2 中断请求使能寄存器 INTEN 32
6.3 中断请求寄存器 INTRQ 33
6.4 GIE 全局中断 33
6.5 PEDGE 34
6.6 INTN 中断 34

7 I/O 口 36

7.1 I/O 口模式 36
7.2 输入上拉寄存器 36
7.3 唤醒寄存器（端口变化唤醒） 36
7.4 端口寄存器 36
7.5 开漏寄存器 36
7.6 输入下拉寄存器 36

8 定时器 37

8.1 看门狗定时器 37
8.2 定时/计数器 38
8.2.1 T0 定时器 38
8.2.2 TM0 定时器/计数器 38
8.2.3 TM1 8bit 定时器/计数器 40
8.2.4 TM2 8bit 定时器/计数器 43

9 I2C 接口 46

9.1 I2CCON 寄存器 46
9.2 I2CADD/I2CBUF 寄存器 47
9.3 I2C 起始和终止信号 47
9.4 从机接收 48
9.5 从机发送 48

11 配置 CONFIG 49

CONFIG1 50
CONFIG2 50

12 指令集 51

1 产品简介

1.1 功能特性

储存器配置

◆OTP ROM：4K*16
◆SRAM：384*8
◆8 层堆栈

振荡器

◆高精度内置 RC 24MHz
◆外接晶体振荡器，20M，8M，4M，1M，32K 模式可选
◆外接 RC 振荡器
◆内置低速振荡器

外设特性

◆14 路 IO 口，可选上拉，驱动可选，最大可达 50mA
◆1 路 8 位定时/计数器
◆1 路 16 位定时/计数器，带 2 路 PWM，PWM 带死区控制，驱动可增加至 100mA
◆2 路 8 位定时/计数器，各带 4 路 PWM，PWM 带死区控制（0,1 路 PWM）
◆2 路外部中断，8/2 个 IO 口可选
◆11 个中断源
◆I2C 从机接口

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 3 个 16 位定时器可周期性的唤醒

指令特性

◆所有跳转指令如 JMP,CALL 等可在整个 ROM 区执行
◆可在整个 ROM 区查表
◆除了跳转指令，其他指令只需要一个指令周期
◆Fcpu=Fosc/2,Fosc/4,Fosc/8,Fosc/16,Fosc/32,Fosc/64

工作电压范围

◆2.3v ∽ 5.5v Fcpu<4MHz
◆3.5v ∽ 5.5v Fcpu<8MHz
◆4.0v ∽ 5.5v Fcpu<12MHz

工作温度

◆-40∽85℃

封装

◆SOP16/DIP16/TSSOP16
◆SOP14/DIP14

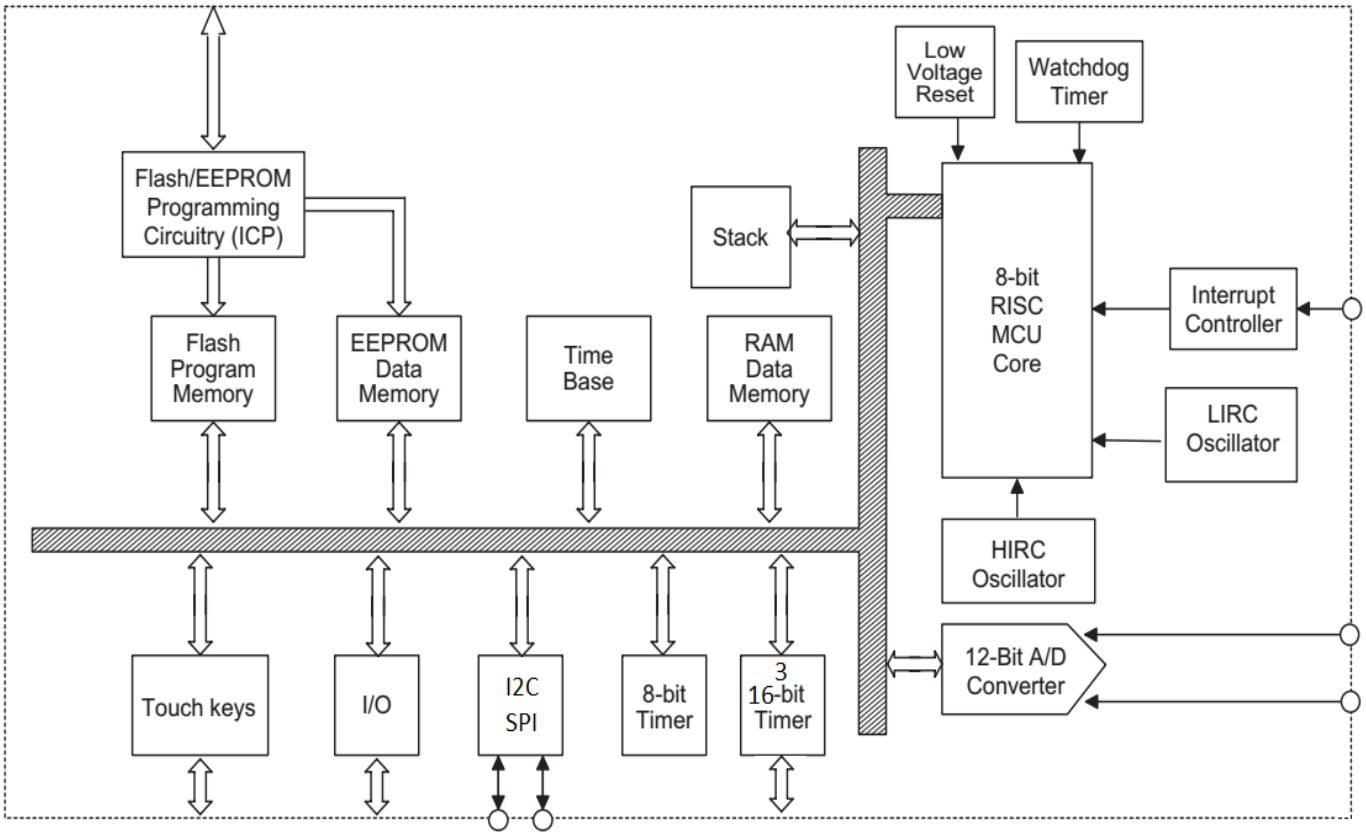
应用领域

◆小家电、玩具、电机控制

特性列表

单片机型号	ROM	RAM	ADC	4 路定时器 10 路 PWM	I/O	E2PROM	封装形式
FX0004	4K*16	384B	*	V	14	*	SOP16/DIP16/TSSOP16 SOP14/DIP14

1.2 系统结构框图


Low Voltage Reset
Watchdog Timer
Flash/EEPROM Programming Circuitry (ICP)
Stack
8-bit RISC MCU Core
Interrupt Controller
Flash Program Memory
EEPROM Data Memory
Time Base
RAM Data Memory
LIRC Oscillator
HIRC Oscillator
12-Bit A/D Converter
Touch keys
I/O
I2C SPI
8-bit Timer
3 16-bit Timer

1.3 引脚配置

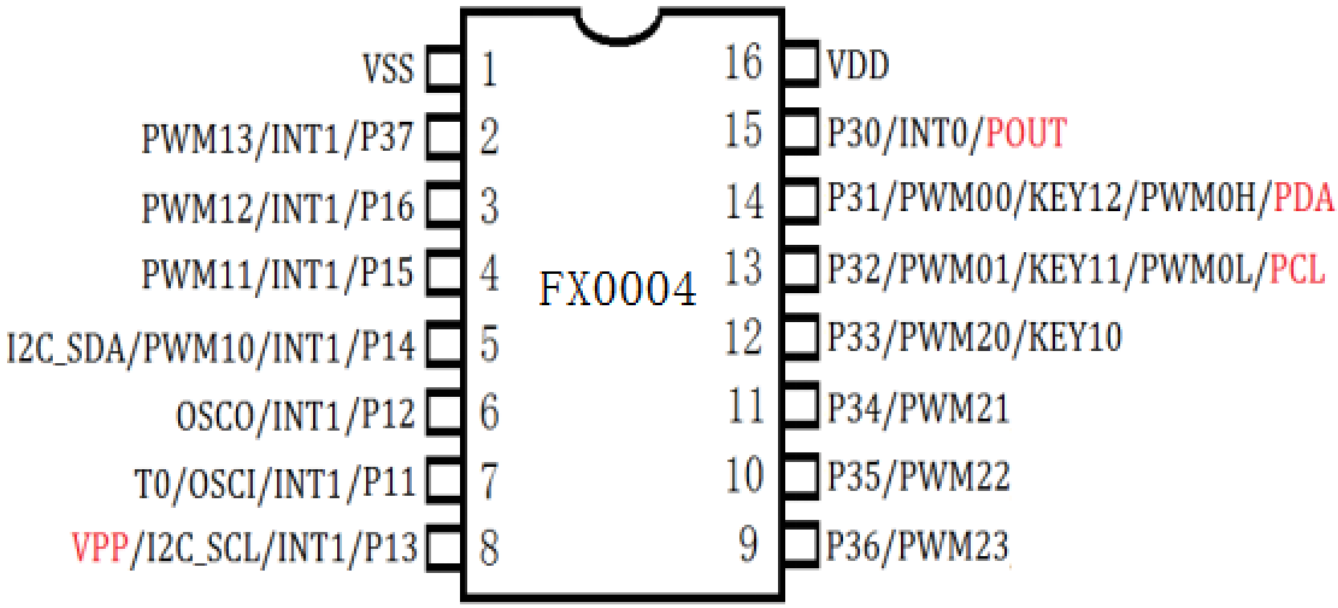


图 1 16 PIN 管脚分配

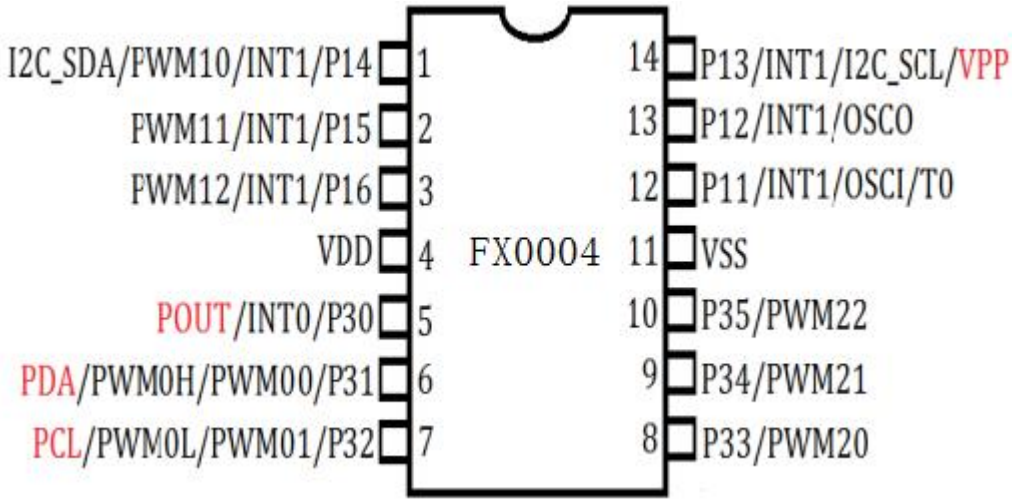


图 2 14 PIN 管脚分配

1.4 引脚说明

管脚名称	输入/输出	说　　明
VDD	I	电源正端（＋）输入端；
VSS	I	电源负端（－）输入端；
P11~P16	I/O	双向输入输出脚，输入时施密特触发，内置上拉\下拉电阻；
P30~P37	I/O	双向输入输出脚，输入时施密特触发，内置上拉\下拉电阻；
PWM0,1,2	O	3 路 PWM 输出脚
PCL,PDA,POUT	P	FLASH 编程脚
INT0/INT1	I	外部中断 0/1 脚
I2C_SCL/I2C_SDA	I/O	I2C 协议管脚

注：I=input,O=output，I/O=input/output

2 中央处理器（CPU）

2.1 程序存储器（ROM）

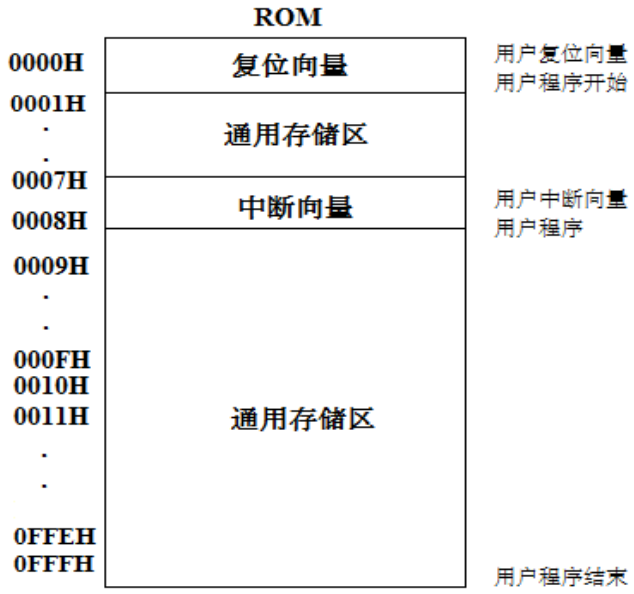
ROM：

1、有 12 位 PC，4K*16bit 的 flash；

2、复位地址 0000h；

3、硬件中断地址 0008h；

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；
- 掉电复位；

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     0000H
        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
```

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 FX0004 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高字节（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
INCMS  Z               ;查找下一地址
JMP    @F              ;Z 没有溢出
INCMS  Y               ;Z 溢出（FFH　00），　Y=Y+1
```

```
            NOP
@F:         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
    @F:
            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
GETDATA:    NOP             ;
            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
```

FX0004 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_A5:                                ; 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_A5:                                ; 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
```

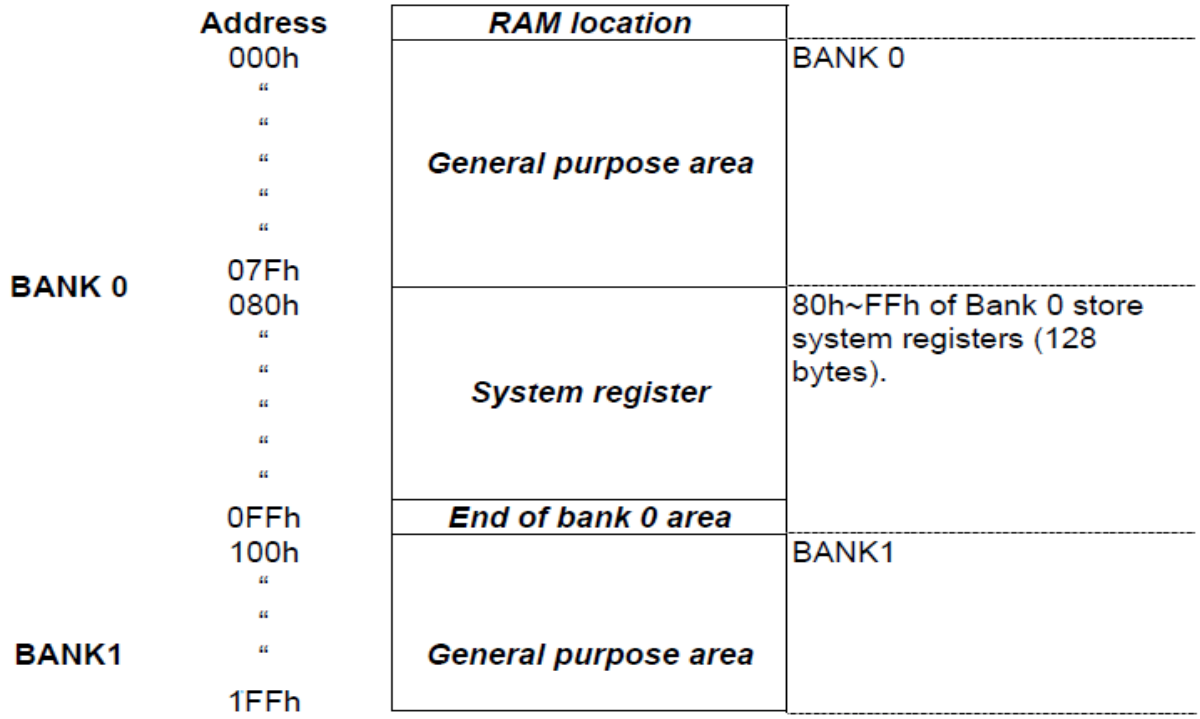
2.1.5 CHECKSUM 计算

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
            CLRY        ; 清 Y
            CLRZ        ; 清 Z
@B:
            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）



2.2.1 系统寄存器

2.2.1.1 系统寄存器列表

地址	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8			R	Z	Y		PFLAG	BANK								
9	TM0M	TM0CL	TM0CH	TM0D0L	TM0D0H	TM0D1L	TM0D1H	PWM0E								
A	TM1M	TM1C	TM1D0	TM1D1	TM1D2	TM1D3	PWM1E		I2CCON	I2CADD	I2CBUF					
B	TM2M	TM2C	TM2D0	TM2D1	TM2D2	TM2D3	PWM2E									PEDGE
C	P1W	P1M		P3M			INTRQ1	INTEN1	INTRQ	INTEN	OSCM	RCADJ	WDTR	OPTION	PCL	PCH
D		P1		P3					T0M	T0C			P1OD	P3OD	P3W	STKP
E		P1UR		P3UR				@YZ					P1PD		P3PD	

2.2.1.2 系统寄存器的位定义

地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	注释
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	/TO	/PD	LVDF	-	-	C	DC	Z	R/W	PFLAG
087H	-	-	-	-	-	-	RBANKS1	RBANKS0	R/W	RBANK
088H										
089H										
08AH										
08BH										
08CH										
08DH										
08EH										
08FH										
090H	TM0TR	TM0PS2	TM0PS1	TM0PS0	TM0CKS1	TM0CKS0	ALOAD0	PWM00E	R/W	TM0M
091H	定时器低 8 位								R/W	TM0CL
092H	定时器高 8 位								R/W	TM0CH
093H	PWM00 占空比低 8 位								R/W	TM0D0L
094H	PWM00 占空比高 8 位								R/W	TM0D0H

095H	PWM01 占空比低 8 位								R/W	TM0D1L
096H	PWM01 占空比高 8 位								R/W	TM0D1H
097H		TM0CW	PWM01OE	PWM00OE	PWM0M	PWM01HL	PWM00HL	PWM01E	R/W	PWM0E
0A0H	TM1TR	TM1PS2	TM1PS1	TM1PS0	TM1CKS1	TM1CKS0	ALOAD1	PWM10E	R/W	TM1M
0A1H	定时器 8 位								R/W	TM1C
0A2H	PWM10 占空比								R/W	TM1D0
0A3H	PWM11 占空比								R/W	TM1D1
0A4H	PWM12 占空比								R/W	TM1D2
0A5H	PWM13 占空比								R/W	TM1D3
0A6H	PWM13OE	PWM12OE	PWM11OE	PWM10OE	PWM1M	PWM13E	PWM12E	PWM11E	R/W	PWM1E
0A8H	HCF	HAAS	HBB	SRW	TXAK	I2CCKP	I2CPU	I2CEN	R/W	I2CCON
0A9H	AD7	AD6	AD5	AD4	AD3	AD2	AD1	-	R/W	I2CADD
0AAH	读写数据缓冲器								R/W	I2CBUF
0B0H	TM2TR	TM2PS2	TM2PS1	TM2PS0	TM2CKS1	TM2CKS0	ALOAD	PWM20E	R/W	TM2M
0B1H	定时器 8 位								R/W	TM2C
0B2H	PWM20 占空比								R/W	TM2D0
0B3H	PWM21 占空比								R/W	TM2D1
0B4H	PWM22 占空比								R/W	TM2D2
0B5H	PWM23 占空比								R/W	TM2D3
0B6H	PWM23OE	PWM22OE	PWM21OE	PWM20OE	PWM2M	PWM23E	PWM22E	PWM21E	R/W	PWM2E
0BFH	INT1S2	INT1S1	INT1S0	INT1G1	INT1G0	INT0G1	INT0G0	INT0S	R/W	PEDGE
0C0H	唤醒寄存器								R/W	P1W
0C1H	-	P16M	P15M	P14M	P13M	P12M	P11M	-	R/W	P1M
0C3H	P37M	P36M	P35M	P34M	P33M	P32M	P31M	P30M	R/W	P3M
0C6H	-	-	-	-	T0IF	-		-	R/W	INTRQ1
0C7H	-	-	-	-	T0IE	-		-	R/W	INTEN1
0C8H	-	I2CIF	RPIF	TM2IF	TM1IF	TM0IF	INT1IF	INT0IF	R/W	INTRQ
0C9H	-	I2CIE	RPIE	TM2IE	TM1IE	TM0IE	INT1IE	INT0IE	R/W	INTEN
0CAH	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-	R/W	OSCM
0CBH	RCADJ 寄存器									RCADJ
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	R/W	WDTR
0CDH	RCADJEN	LIRC_EN	-	LVDTEN	RTCCON	WDTP2	WDTP1	WDTP0	R/W	OPTION
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH	-	-	-	-	PC11	PC10	PC9	PC8	R/W	PCH
0D1H	-	P16	P15	P14	P13	P12	P11	-	R/W	P1
0D3H	P37	P36	P35	P34	P33	P32	P31	P30	R/W	P3
0D8H	T0TR	T0PS2	T0PS1	T0PS0	-	-	-	-	R/W	T0M
0D9H	T0 定时器寄存器								R/W	T0C
0DCH	-	P16OD	P15OD	P14OD	P13OD	P12OD	P11OD	P10OD	R/W	P1OD
0DEH	唤醒寄存器								R/W	P3W
0DFH	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0	R/W	STKP
0E1H	-	P16UR	P15UR	P14UR	P13UR	P12UR	P11UR	-	R/W	P1UR
0E3H	P37UR	P36UR	P35UR	P34UR	P33UR	P32UR	P31UR	P30UR	R/W	P3UR
0E7H	-	-	-	-	-	-	-	-	R/W	@YZ
0ECH	-	P16PD	P15PD	P14PD	P13PD	P12PD	P11PD	-	R/W	P1PD
0EEH	P37PD	P36PD	P35PD	P34PD	P33PD	P32PD	P31PD	P30PD	R/W	P3PD

2.2.2 累加器

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

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

例：读/写 ACC

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

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

例：ACC 和工作寄存器中断保护操作

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

2.2.3 程序状态寄存器 PFLAG

地址	名称	D7	D6	D5	D4	D3	D2	D1	D0
86H(R/W)	PFLAG	/TO	/PD	LVDF	-	-	C	DC	Z

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

=1，电源电压大于设定电压点；

=0，电源电压小于设定电压点；

/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；

2.2.4 程序计数器

地址	名称	D7	D6	D5	D4	D3	D2	D1	D0
0CE(R/W)	PCL	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
0CF(R/W)	PCH	-	-	-	-	PC11	PC10	PC9	PC8
复位状态	BOR	0	0	0	0	0	0	0	0
	WDT	0	0	0	0	0	0	0	0

PCH，PCL 都是可读可写的。

- 产生 4K*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 12 位。因此，“GOTO”指令跳转范围 4K 程序区间；
- “CALL”指令加载 PC 的 12 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 4 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 4 位保持不变；
- “SUBWF PCL,1”允许“A”的值被减到当前 PC，但 PC 的高 4 位保持不变；
- 改变 PCL 内容的指令需要 2 个指令周期；

2.2.5 Y,Z 寄存器

寄存器 Y 和 Z 都是 8 位缓存器，主要用途如下：

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

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
```

2.2.6 R 寄存器

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。）

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

2.2.7 RAM 页面控制寄存器(RBANK)

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

RBANKS0：RAM 页面控制位

=1，选择 BANK1

=0，不选择 BANK1

RBANKS1：RAM 页面控制位

=1，选择 BANK2

=0，不选择 BANK2

2.2.8 OPTION 寄存器

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
CDH(r/w)	OPTION	RCADJEN	LIRC_EN	-	LVDTEN	RTCCON	WDTP2	WDTP1	WDTP0
RESET		0	0	0	1	1	1	1	1

RCADJEN:内置高频 RC 频率调整位

0：默认 config _word1<7:0>控制 RC 的频率；

1：选择 RCADJ 寄存器的值控制 RC 的频率；

注：本机支持 movc 查表指令查找 config 配置位的值。查到对应 config 的值后，可以先把 config 的值写 RCADJ 寄存器，然后再根据需要调整 RC 的频率。

LIRC_EN:

0：不单独控制（在 WDT 使能时也会有效）；

1：低速 RC 振荡；

LVDTEN: LVD 检测使能位

0：无效；

1：有效；

RTCCON: RTC 模式时的睡眠模式

0：睡眠时 RTC 的外置振荡器也关闭；

1：一直打开外置振荡器；

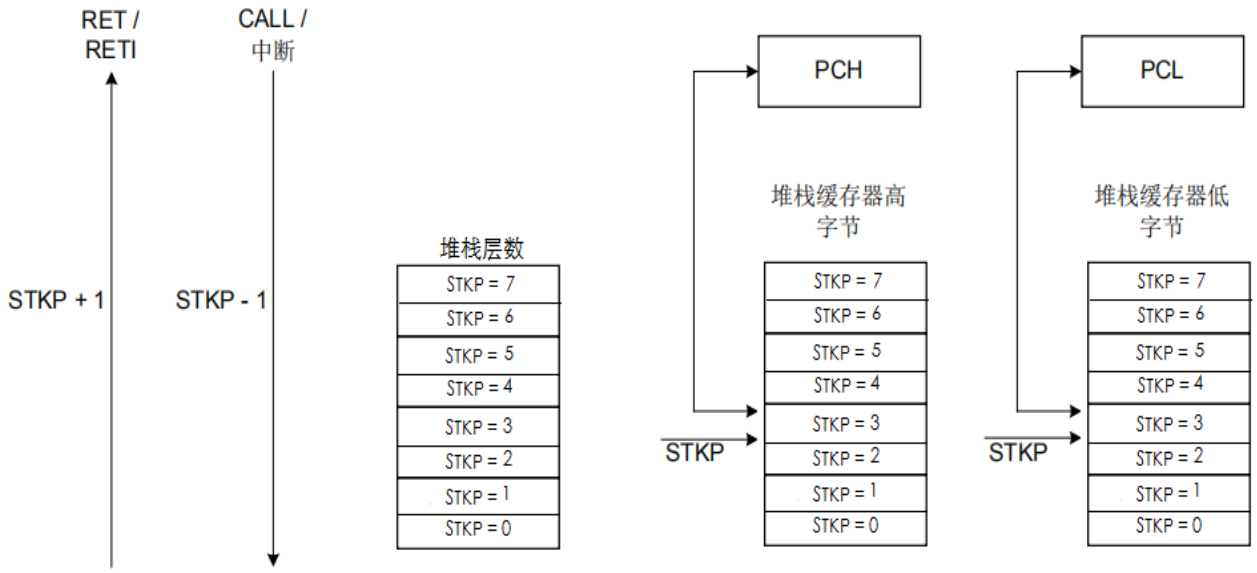
WDTP<2:0>：WDT 溢出时间选择位

WDTP	WDTP1	WDTP0	WDT 的分频比
0	**0**	**0**	1:1**(16ms)**
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

2.3 堆栈

2.3.1 概述

FX0004 的堆栈缓存器共 8 层，程序进入中断或执行 CALL 指令时，用来存储程序计数器 PC 的值。寄存器 STKP 为堆栈指针，STKnH 和 STKnL 分别是各堆栈缓存器的高、低字节。



2.3.2 堆栈寄存器

堆栈指针 STKP 是一个 3 位寄存器，存放被访问的堆栈单元地址，10 位数据存储器 STKnH 和 STKnL 用于暂存堆栈数据。以上寄存器都位于 bank 0。

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

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

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	-	-	-	-	STKPB2	STKPB1	STKPB0
复位后	0	-	-	-	-	1	1	1

Bit[2:0]　　**STKPBn：**堆栈指针（n = 0 ~ 2）。

Bit 7 GIE：全局中断控制位。 0 = 禁止； 1 = 使能。

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

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

3 复位

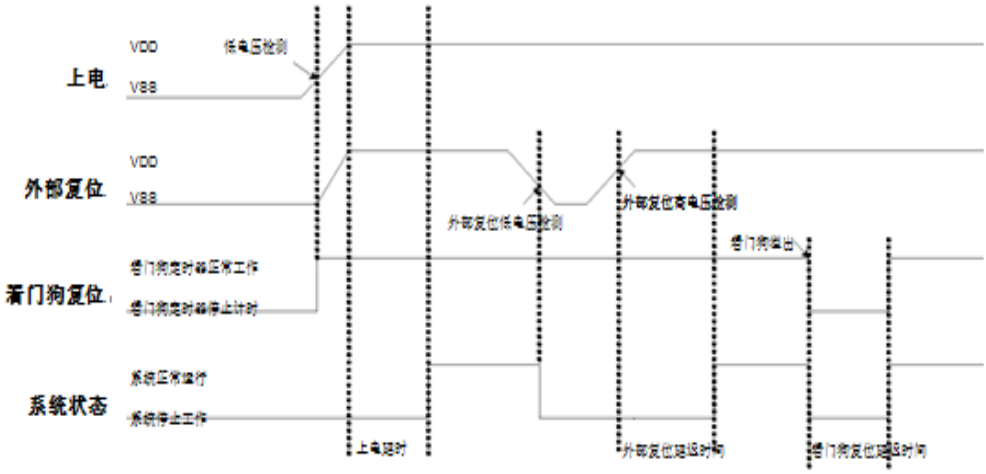
3.1 概述

FX0004 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；

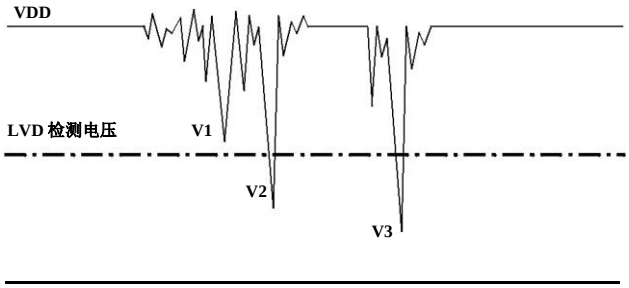
看门狗定时器应用注意事项：

- 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。

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。

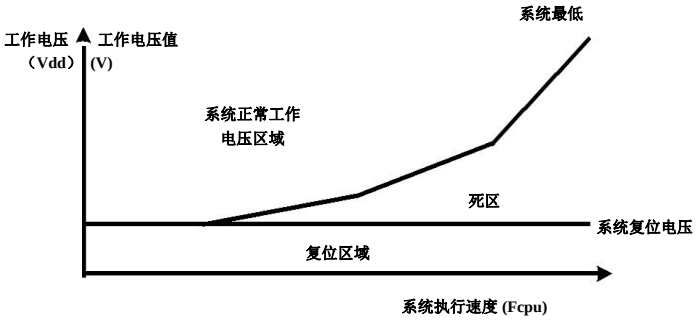
AC 运用中：

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 低电压检测 LVD

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。

3.4.4 掉电复位性能改进

如何改善系统掉电复位性能，有以下几点建议：

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

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

看门狗复位：看门狗定时器用于保证系统正常工作。通常，会在主程序中将看门狗定时器清零，但不要在多个分支程序中清看门狗。若程序正常运行，看门狗不会复位。当系统进入死区或程序运行出错的时候，看门狗定时器继续计数

直至溢出，系统复位。如果看门狗复位后电源仍处于死区，则系统复位失败，保持复位状态，直到系统工作状态恢复到正常值。

降低系统工作速度：系统工作速度越快最低工作电压值越高，从而加大工作死区的范围，因此降低系统工作速度不失为降低系统进入死区几率的有效措施。所以，可选择合适的工作速度以避免系统进入死区，这个方法需要调整整个程序使其满足系统要求。

附加外部复位电路：外部复位也能够完全改善掉电复位性能。有三种外部复位方式可改善掉电复位性能：稳压二极管复位电路，电压偏移复位电路和外部 IC 复位。它们都采用外部复位信号控制单片机可靠复位。

3.5 外部复位

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

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

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

4 系统时钟

4.1 概述

FX0004 内置双时钟系统：高速时钟和低速时钟。高速时钟包括内部高速时钟和外部高速时钟，由 OPTION 选项 RCADJEN 选择。低速时钟由内部低速振荡器提供，高、低速时钟都可以作为系统时钟源。

- **高速振荡器**

  内部高速振荡器：高达 24MHz，称为 IHRC；外部高速振荡器：包括晶体（20MHz、8MHz、4MHz、1MHz、32KHz）振荡器和 RC 振荡器。

- **低速振荡器**

  内部低速振荡器：32KHz@5V，称为 ILRC。

4.2 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 Config0 配置选项表<10:8>决定，正常模式下，Fcpu=Fosc/2~Fosc/64。若高速时钟源为外部 4MHz 振荡器，则 Config0 配置选项表<10:8>选择 Fhosc/4，则 Fcpu 频率为 4MHz/4=1MHz。低速模式下，Fcpu=Flosc/2，即 32KHz/2=16KHz@5V。

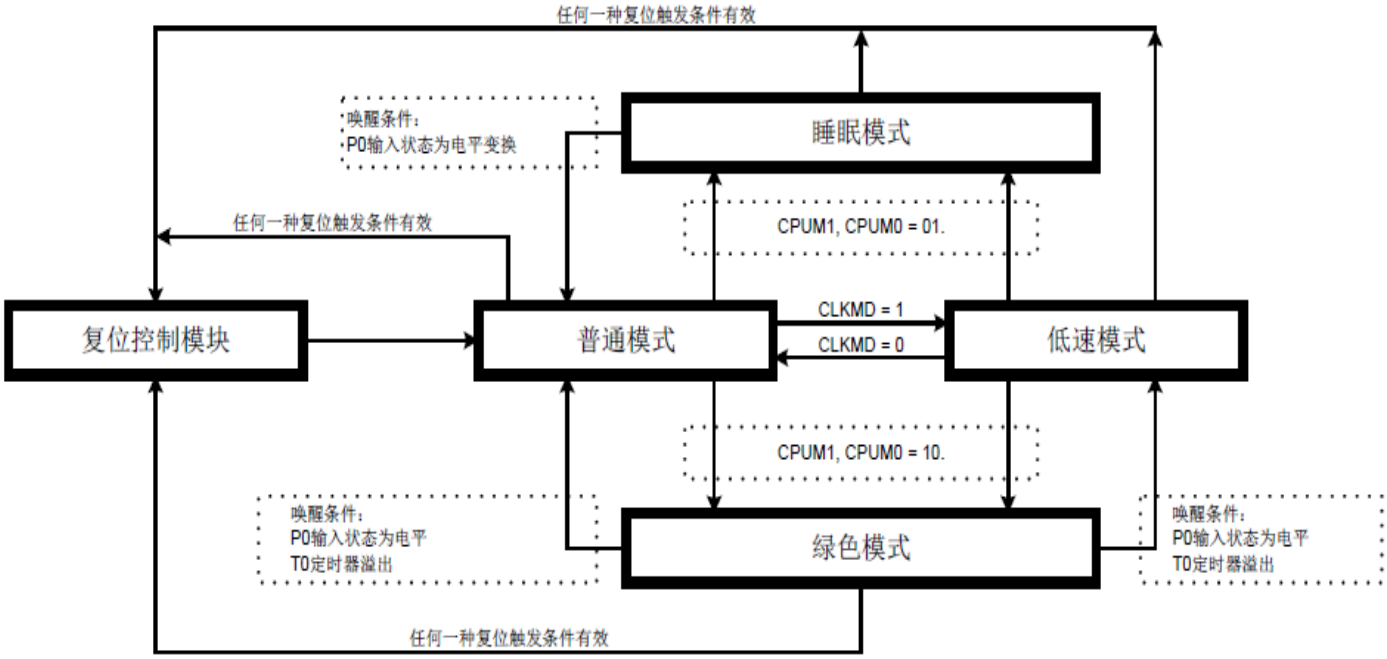
5 系统工作模式

5.1 概述

FX0004 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



模块功能	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	可控	可控
CPU 指令	执行	执行	停止	停止
T0	T0TR	T0TR 控制	停止	停止
TM1/TM2/TM3	TMTR1/2/3 控制	TMTR1/2/3 控制	TMTR1/2/3 控制	TMTR1/2/3 控制
WDT	WDTEN	WDTEN	WDTEN	WDTEN
外部中断	可控	可控	可控	可控
内部中断	可控	可控	可控	可控
唤醒	—	—	有效	有效
I2C	可控	可控	可控	可控

注：绿色模式唤醒新增加 T1 定时器溢出唤醒

5.2 普通模式

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

- 程序被执行，所有的功能都可控制；
- 系统速率为高速；
- 高速振荡器和内部低速 RC 振荡器都正常工作；
- 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；

5.3 低速模式

低速模式为系统低速时钟正常工作模式。系统时钟源由内部低速 RC 振荡器提供。低速模式由 OSCM 寄存器的 CLKMD 位控制。当 CLKMD=0 时，系统为普通模式；当 CLKMD=1 时，系统进入低速模式。切换进入低速模式后，不能自动禁止高速振荡器，必须通过 SPTHX 位来禁止以减少功耗。低速模式下，系统速率被固定为 Flosc/2（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/2）；
- 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；

5.4 睡眠模式

睡眠模式是系统的理想状态，不执行程序，振荡器也停止工作。整个芯片的功耗低于 1uA。睡眠模式可以由 P1，P3 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 P1，P3 电平变换触发和 TM0/TM1/TM2 选择外部或低速 RC 为时钟源时；

5.5 绿色模式

绿色模式是另外的一种理想状态。在睡眠模式下，所有的功能和硬件设备都被禁止，但在绿色模式下，系统时钟保持工作，绿色模式下的功耗大于睡眠模式下的功耗。绿色模式下，不执行程序，但具有唤醒功能的定时器仍正常工作，定时器的时钟源为仍在工作的系统时钟。绿色模式下，有 2 种方式可以将系统唤醒：1、P1，P3 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 功能仍然有效，但是定时器溢出时不能唤醒系统；

5.6 工作模式控制宏

FC252X/352X 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
```

5.7 系统唤醒

5.7.1 概述

睡眠模式和绿色模式下，系统并不执行程序。唤醒触发信号可以将系统唤醒进入普通模式或低速模式。唤醒触发信号包括：外部触发信号（P1，P3 的电平变换）和内部触发（TM0/TM1/TM2 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（P1，P3 电平变换）和内部触发信号（TM0/TM1/TM2 溢出）。

5.7.2 唤醒时间

系统进入睡眠模式后，高速时钟振荡器停止运行。把系统从睡眠模式唤醒时，单片机需要等待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：

$$\text{唤醒时间} = 2048 * T_{OSC}$$

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

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。

➢ **例：停止高速振荡器。**

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

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

6 中断

6.1 概述

FX0004 提供 9 个中断源：4 个内部中断和 5 个外部中断。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。

6.2 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	-	I2CIE	RPIE	TM2IE	TM1IE	TM0IE	INT1IE	INT0IE
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	-	0	0	0	0	0	0	0

I2CIE： I2C 接口接收或发送完成，中断使能；

RPIE: 端口变化中断使能（P1，P3）

TM2IE: TM2 溢出中断使能

TM1IE: TM1 溢出中断使能

TM0IE: TM0 溢出中断使能

INT1IE: INT1 中断使能

INT0IE: INT0 中断使能

0：中断无效

1：中断使能

0C7H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTEN1	-	-	-	-	T0IE	-	-	-
R/W	-	-	-	-	R/W	-	-	-
复位	-	-	-	-	0	-	-	-

T0IE: T0 溢出中断使能；

6.3 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	-	I2CIF	RPIF	TM2IF	TM1IF	TM0IF	INT1IF	INT0IF
R/W	-	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	-	0	0	0	0	0	0	0

0C6H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTRQ1	-	-	-	-	T0IF	-	-	-
R/W	-	-	-	-	R/W	-	-	-
复位	-	-	-	-	0	-	-	-

中断标志：

= 0：中断无效；

= 1：中断有效，软件清 0；

其中 INT0IF、INT1IF、TM0IF、TM1IF、TM2IF、RPIF、I2CIF 可唤醒。

6.4 GIE 全局中断

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

6.5 PEDGE

地址	名称	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
BFh(r/w)	PEDGE	INT1S2	INT1S1	INT1S0	INT1G1	INT1G0	INT0G1	INT0G0	INT0S
reset		0	0	0	0	0	0	0	0

INT0S:

0：选择 P30 口作为 INT0

1：选择 P31 口作为 INT0

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

INT1S[2:0] INT1 输入脚选择

000 =P11;001=P12;010=P13;011=P14;100=P15;101=P16;110=P36;111=P37;

6.6 INTn 中断

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
```

6.7 TC0 中断

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

例：TC0 中断请求设置

```
        B0BCLR      FTC0IEN     ; 禁止 TC0 中断
        B0BCLR      FTC0ENB
        MOV         A, #20H
        B0MOV       TC0M, A     ; TC0 时钟＝Fcpu / 64
        MOV         A, # 64H    ; TC0C 初始值＝64H
        B0MOV       TC0C, A     ; TC0 间隔= 10 ms
        B0BCLR      FTC0IRQ     ; 清 TC0 中断请求标志
        B0BSET      FTC0IEN     ; 使能 TC0 中断
        B0BSET      FTC0ENB
        B0BSET      FGIE        ; 使能 GIE
```

例：TC0 中断服务程序

```
        ORG         8H
        JMP INT_SERVICE
INT_SERVICE：
        …                       ; 保存 ACC 和 PFLAG
        B0BTS1      FTC0IRQ     ; 检查是否有 TC0 中断请求标志
        JMP EXIT_INT            ; TC0IRQ = 0，退出中断
        B0BCLR      FTC0IRQ     ; 清 TC0IRQ
        MOV         A, #64H
        B0MOV       TC0C, A     ; 清 TC0C
        …                       ; TC0 中断程序
        …
EXIT_INT：
        …                       ; 恢复 ACC 和 PFLAG
        RETI                    ; 退出中断
```

7 I/O 口

7.1 I/O 口模式

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
C1H(r/w)	P1M	-	P16M	P15M	P14M	P13M	P12M	P11M	-
C3H(r/w)	P3M	P37M	P36M	P35M	P34M	P33M	P32M	P31M	P30M
reset		0	0	0	0	0	0	0	0

PnM[7:0]：P1,P3 模式控制位　0 = 输入模式；1 = 输出模式；

7.2 输入上拉寄存器

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
E1H(r/w)	P1UR	-	P16UR	P15UR	P14UR	P13UR	P12UR	P11UR	-
E3H(r/w)	P3UR	P37UR	P36UR	P35UR	P34UR	P33UR	P32UR	P31UR	P30UR
reset		0	0	0	0	0	0	0	0

PnUR[7:0]：P1,P3 上拉控制位　0 = 无上拉；1 = 有上拉（必须输入模式下）；

7.3 唤醒寄存器（端口变化唤醒）

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
C0H(r/w)	P1W	唤醒寄存器							
DEH(r/w)	P3W	唤醒寄存器							
reset		00h							

PnW[7:0]：P1,P3 口唤醒控制位　0 = 屏蔽；1 = 使能；

7.4 端口寄存器

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
D1H(r/w)	P1	-	P16	P15	P14	P13	P12	P11	-
D3H(r/w)	P3	P37	P36	P35	P34	P33	P32	P31	P30
reset		0	0	0	0	0	0	0	0

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

7.5 开漏寄存器

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
DCH(r/w)	P1OD	-	P16OD	P15OD	P14OD	P13OD	P12OD	P11OD	-
DDH(r/w)	P3OD	P37OD	P36OD	P35OD	P34OD	P33OD	P32OD	P31OD	P30OD
reset		0	0	0	0	0	0	0	0

PnOD[7:0]：P1,P3 口开漏控制位　0 = 屏蔽；1 = 使能；

7.6 输入下拉寄存器

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
ECH(r/w)	P1PD	-	P16PD	P15PD	P14PD	P13PD	P12PD	P11PD	-
EEH(r/w)	P3PD	P37PD	P36PD	P35PD	P34PD	P33PD	P32PD	P31PD	P30PD
reset		0	0	0	0	0	0	0	0

PnPD[7:0]：P1,P3 口下拉控制位　0 = 屏蔽；1 = 使能；

8 定时器

8.1 看门狗定时器

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

看门狗溢出时间 = 1024/内部低速振荡器频率*分频系数*0.5

看门狗定时器的 3 种工作模式由 OPTION 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不工作；
- 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；

注意： 1.不分频时 wdt 溢出时间为 16ms。

2.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
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
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:                        ;I/O 或 RAM 出错，不清看门狗等看门狗计时溢出
        JMP$
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
```

8.2 定时/计数器

8.2.1 T0 定时器

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
0D8h(r/w)	T0M	T0TR	T0PS2	T0PS1	T0PS0	-	-	-	-

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
0D9h(r/w)	T0C	T0 定时器寄存器							
reset	00	0000 0000							

T0PS[2:0]: TCn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

T0TR: T1启动控制位

= 0，禁止T0定时器

= 1，开启T0定时器

8.2.2 TM0 定时器/计数器

8.2.2.1 TM0M

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
90H(r/w)	TM0M	TM0TR	TM0PS2	TM0PS1	TM0PS0	TM0CKS1	TM0CKS0	ALOAD0	PWM00E

PWM00E: PWM0输出控制

= 0，禁止PWM00

= 1，使能PWM00

ALOAD: 自动装载控制

= 0，禁止TM0自动装载

= 1，使能TM0自动装载

T0CKS1:T0CKS0: Tn时钟选择

= 00，TM0选择Fcpu作为时钟输入

= 01，TM0选择Fhosc作为时钟输入

= 10，T M0选择外部管脚作为时钟输入

= 11，T M0选择RTC振荡器作为时钟输入

TM0PS[2:0]: T M0分频选择位

= 000，Ft1/128

= 001，Ft1/64

= 010，Ft1/32

= 011，Ft1/16

= 100，Ft1/8

= 101，Ft1/4

= 110，Ft1/2

= 111，Ft1/1

TM0TR: T M0启动控制位

= 0，禁止T M0定时器

= 1，开启T M0定时器

8.2.2.2 TM0C(TM0 计数寄存器)

地址 Bank1	名称	B7	B6	B5	B4	B3	B2	B1	B0
91H (r/w)	TM0CL	定时器低 8 位							
92H (r/w)	TM0CH	定时器高 8 位							

16 位计数器 TM0C 溢出时，TM0IF 置 1，需软件清零，用来控制 TM0 的中断间隔时间。TM0C 具有单向缓冲结构，写 TM0C 的同时，会把 TM0C 的值写入缓冲器。

TM0C 具有溢出自动重装载功能，前提必须 aload=1 或者 pwm 开启。重载入的值就是软件写入的值。

8.2.2.3 TM0D0 (PWM00 占空比寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
93H (r/w)	TM0D0L	PWM00 占空比低 8 位							
94H (r/w)	TM0D0H	PWM00 占空比高 8 位							

用于设置 PWM00 高电平时间，PWM00 高电平时间= TM0D0- TM0C*ALOAD

8.2.2.4 TM0D1 (PWM01 占空比寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
95H (r/w)	TM0D1L	PWM01 占空比低 8 位							
96H (r/w)	TM0D1H	PWM01 占空比高 8 位							

用于设置 PWM01 高电平时间，PWM01 高电平时间= TM0D1- TM0C*ALOAD

8.2.2.5 PWM0E(PWM 控制寄存器)

地址 Bank1	名称	B7	B6	B5	B4	B3	B2	B1	B0
97H(r/w)	PWM0E		TM0CW	PWM01OE	PWM00OE	PWM0M	PWM01HL	PWM00HL	PWM01E

PWM01E: PWM01输出控制

= 0，禁止PWM01；

= 1，使能PWM01;

PWM00HL: PWM00逻辑电平选择（最后选择）

= 0，0电平。即输出从0开始，TM0D0到后输出1；

= 1，1电平。即输出从1开始，TM0D0到后输出0;

PWM01HL: PWM01逻辑电平选择

= 0，0电平。即输出从0开始，TM0D1到后输出1；

= 1，1电平。即输出从1开始，TM0D1到后输出0;

PWM0M: PWM0M是否互补输出

= 0，PWM00,PWM01是各自独立输出；

= 1，PWM00,PWM01是互补输出（PWM00E(TM0M<0>)必须=1）；

注：当 PWM 互补输出时，为了防止外部驱动管同时导通产生大电流，设计了死区控制模块。

互补输出时的死区时间：当 PWM 工作在互补模式时，TM0D1 被当做死区时间控制器，死区时间为 TM0D1*F_{pwm}。F_{pwm}为 PWM 的分频后的输入频率，即 TM0PS[2:0]选择后的值。死区时间必须小于 PWM 的占空比时间（小于高、低电平宽度的最小值）。死区定时器只能在主定时器启动后才能启动。如果 TM0D1=0，则死区时间为一个的输入时钟周期（TM0CKS 选择的）。

PWM00OE：PWM0OE输出使能

= 0，IO；

= 1，PWM00输出，IO口变成输出口；

PWM01OE：PWM1OE输出使能

= 0，IO；

= 1，PWM01输出，IO口变成输出口；

TM0CW：

= 0，不能从sleep状态唤醒；

= 1，使用外部时钟或低速时钟或rtc时钟可唤醒；

8.2.3 TM1 8bit 定时器/计数器

8.2.3.1 TM1M

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
A0h (r/w)	TM1M	TM1TR	TM1PS2	TM1PS1	TM1PS0	TM1CKS1	TM1CKS0	ALOAD1	PWM10E
reset		0	0	0	0	0	0	0	0

PWM10E: PWM1输出控制

= 0，禁止PWM10

= 1，使能PWM10

ALOAD: 自动装载控制

= 0，禁止TM1自动装载

= 1，使能TM1自动装载

TM1CKS1:TM1CKS0: Tn时钟选择

= 00，TM1选择FCPU作为时钟输入

= 01，TM1选择FHOSC作为时钟输入

= 10，TM1选择外部管脚作为时钟输入（TM1不选择外部管脚，选择时钟源为16K低速时钟）

= 11，TM1选择RTC振荡器作为时钟输入

TM1PS[2:0]: TM1分频选择位

= 000，Ft1/128

= 001，Ft1/64

= 010，Ft1/32

= 011，Ft1/16

= 100，Ft1/8

= 101，Ft1/4

= 110，Ft1/2

= 111，Ft1/1

TM1TR: T1启动控制位

= 0，禁止TM1定时器

= 1，开启TM1定时器

8.2.3.2 TM1C(TM1 计数寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
A1h (r/w)	TM1C	定时器 8 位							
reset	00h	00h							

8 位计数器为加计数 TM1C 溢出时，TM1IF 置 1，需软件清零，用来控制 TM1 的中断间隔时间。TM1C 具有单向缓冲结构，写 TM1C 的同时，会把 TM1C 的值写入缓冲器。

TM1C 具有溢出自动重装载功能，前提必须 ALOAD=1 或者 PWM 开启。重载入的值就是上一次写入 TM1C 的值。

8.2.3.3 TM1D0 (PWM10 占空比寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
A2h (r/w)	TM1D0	PWM10 占空比							

用于设置 PWM10 高电平时间，PWM1 高电平时间= TM1D0 - TM1C 的初值

8.2.3.4 TM1D1 (PWM11 占空比寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
A3h (r/w)	TM1D1	PWM11 占空比							

用于设置 PWM11 高电平时间，PWM11 高电平时间= TM1D1 - TM1C 的初值

8.2.3.5 TM1D2 (PWM12 占空比寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
A4h (r/w)	TM1D2	PWM12 占空比							

用于设置 PWM12 高电平时间，PWM12 高电平时间= TM1D2 - TM1C 的初值

8.2.3.6 TM1D3 (PWM13 占空比寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
A5h (r/w)	TM1D3	PWM13 占空比							

用于设置 PWM13 高电平时间，PWM13 高电平时间= TM1D3 - TM1C 的初值

8.2.3.7 PWM1E(PWM 控制寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
A6h (r/w)	PWM1E	PWM13OE	PWM12OE	PWM11OE	PWM10OE	PWM1M	PWM13E	PWM12E	PWM11E
reset		0	0	0	0	0	0	0	0

PWM11E: PWM11输出控制

= 0，禁止PWM11

= 1，使能PWM11

PWM12E: PWM12输出控制

= 0，禁止PWM12

= 1，使能PWM12

PWM13E: PWM13输出控制

= 0，禁止PWM13

= 1，使能PWM13

PWM1M: PWM0M是否互补输出

= 0，PWM10,PWM11是各自独立输出

= 1，PWM10,PWM11是互补输出（PWM00E(TM0M<0>)必须=1）

注：当 PWM 互补输出时，为了防止外部驱动管同时导通产生大电流，设计了死区控制模块。

互补输出时的死区时间：当 PWM 工作在互补模式时，TM1D1 被当做死区时间控制器，死区时间为 TM0D1*FPWM。FPWM 为 PWM 的分频后的输入频率，即 TM1PS[2:0]选择后的值。死区时间必须小于 PWM 的占空比时间（小于高、低电平宽度的最小值）。死区定时器只能在主定时器启动后才能启动。如果 TM0D1=0，则死区时间为一个的输入时钟周期（TM0CKS 选择的）。

PWM10OE: PWM10输出选择

= 0，禁止PWM10输出，P14作为I/O

= 1，使能PWM10输出，P14输出PWM10信号

PWM11OE: PWM11输出选择

= 0，禁止PWM11输出，P15作为I/O

= 1，使能PWM11输出，P15输出PWM11信号

PWM12OE: PWM12输出选择

= 0，禁止PWM12输出，P16作为I/O

= 1，使能PWM12输出，P16输出PWM12信号

PWM13OE: PWM13输出选择

= 0，禁止PWM13输出，P37作为I/O

= 1，使能PWM13输出，P37输出PWM13信号

8.2.4 TM2 8bit 定时器/计数器

8.2.4.1 TM2M

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
B0h (r/w)	TM2M	TM2TR	TM2PS2	TM2PS1	TM2PS0	TM2CKS1	TM2CKS0	ALOAD2	PWM20E
reset		0	0	0	0	0	0	0	0

PWM20E: PWM2输出控制

= 0，禁止PWM20

= 1，使能PWM20

ALOAD: 自动装载控制

= 0，禁止TM2自动装载

= 1，使能TM2自动装载

TM2CKS1:TM2CKS0: Tn时钟选择

= 00，TM2选择FCPU作为时钟输入

= 01，TM2选择FHOSC作为时钟输入

= 10，选择时钟源为16K低速时钟

= 11，TM2选择RTC振荡器作为时钟输入

TM2PS[2:0]: TM2分频选择位

= 000，Ft1/128

= 001，Ft1/64

= 010，Ft1/32

= 011，Ft1/16

= 100，Ft1/8

= 101，Ft1/4

= 110，Ft1/2

= 111，Ft1/1

TM2TR: T1启动控制位

= 0，禁止TM2定时器

= 1，开启TM2定时器

8.2.4.2 TM2C(TM2 计数寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
B1h (r/w)	TM2C	定时器 8 位							
reset	00h	00h							

8 位计数器为加计数 TM2C 溢出时，TM2IF 置 1，需软件清零，用来控制 TM2 的中断间隔时间。TM2C 具有单向缓冲结构，写 TM2C 的同时，会把 TM2C 的值写入缓冲器。

TM2C 具有溢出自动重装载功能，前提必须 ALOAD=1 或者 PWM 开启。重载入的值就是上一次写入 TM2C 的值。

8.2.4.3 TM2D0 (PWM20 占空比寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
B2h (r/w)	TM2D0	PWM20 占空比							

用于设置 PWM20 高电平时间，PWM20 高电平时间= TM2D0 - TM2C 的初值

8.2.4.4 TM2D1 (PWM21 占空比寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
B3h (r/w)	TM2D1	PWM21 占空比							

用于设置 PWM21 高电平时间，PWM21 高电平时间= TM2D1 - TM2C 的初值

8.2.4.5 TM2D2 (PWM22 占空比寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
B4h (r/w)	TM2D2	PWM22 占空比							

用于设置 PWM22 高电平时间，PWM22 高电平时间= TM2D2 - TM2C 的初值

8.2.4.6 TM2D3 (PWM23 占空比寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
B5h (r/w)	TM2D3	PWM23 占空比							

用于设置 PWM23 高电平时间，PWM23 高电平时间= TM2D3 - TM2C 的初值

8.2.4.7 PWM2E(PWM 控制寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
B6h (r/w)	PWM2E	PWM23OE	PWM22OE	PWM21OE	PWM20OE	PWM2M	PWM23E	PWM22E	PWM21E
reset		0	0	0	0	0	0	0	0

PWM21E: PWM21输出控制

= 0，禁止PWM21

= 1，使能PWM21

PWM22E: PWM22输出控制

= 0，禁止PWM22

= 1，使能PWM22

PWM23E: PWM23输出控制

= 0，禁止PWM23

= 1，使能PWM23

PWM2M: PWM0M是否互补输出

= 0，PWM20,PWM21是各自独立输出

= 1，PWM20,PWM21是互补输出（PWM00E(TM0M<0>)必须=1）

注：当 PWM 互补输出时，为了防止外部驱动管同时导通产生大电流，设计了死区控制模块。

互补输出时的死区时间：当 PWM 工作在互补模式时，TM2D1 被当做死区时间控制器，死区时间为 TM0D1*FPWM。FPWM 为 PWM 的分频后的输入频率，即 TM2PS[2:0]选择后的值。死区时间必须小于 PWM 的占空比时间（小于高、低电平宽度的最小值）。死区定时器只能在主定时器启动后才能启动。如果 TM0D1=0，则死区时间为一个的输入时钟周期（TM0CKS 选择的）。

PWM20OE: PWM20输出选择

= 0，禁止PWM20输出，P33作为I/O

= 1，使能PWM20输出，P33输出PWM20信号

PWM21OE: PWM21输出选择

= 0，禁止PWM21输出，P34作为I/O

= 1，使能PWM21输出，P34输出PWM21信号

PWM22OE: PWM22输出选择

= 0，禁止PWM22输出，P35作为I/O

= 1，使能PWM22输出，P35输出PWM22信号

PWM23OE: PWM23输出选择

= 0，禁止PWM23输出，P36作为I/O

= 1，使能PWM23输出，P36输出PWM23信号

9 I2C 接口

I2C 串行接口是一个双线的接口，有一条串行数据线 SDA 和一条串行时钟线 SCL。由于可能有多个设备在同一条总线上相互连接，所以这些设备的输出都是开漏型输出。因此应在这些输出口上都应加上拉电阻。应注意的是：I2C 总线上的每个设备都没有选择线，但分别与唯一的地址一一对应，用于 I2C 通信。如果有两个设备通过双向的 I2C 总线进行通信，那么就存在一个主机和一个从机。主机和从机都可以用于传输和接收数据，但只有主机才可以控制总线动作。那些处于从机模式的设备，要在 I2C 总线上传输数据只有两种方式，一是从机发送模式，二是从机接收模式。

本机只有从机模式。

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
A8H(r/w)	I2CCON	HCF	HAAS	HBB	SRW	TXAK	I2CCKP	I2CPU	I2CEN
A9H(r/w)	I2CADD	AD7	AD6	AD5	AD4	AD3	AD2	AD1	-
AAH(r/w)	I2CBUF	读写数据缓冲器							

9.1 I2CCON 寄存器

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
A8H(r/w)	I2CCON	HCF	HAAS	HBB	SRW	TXAK	I2CCKP	I2CPU	I2CEN
R/W	R/W	R	R	R	R	R/W	R/W	R/W	R/W
reset		0	0	0	0	1	1	0	0

Bit 7 HCF：

0：数据正在被传输；

1：8位数据传输完成；

HCF 是数据传输标志位。数据正在传输时该位为低。当 8 位数据传输完成时，此位为高。

Bit 6 HAAS：I2C地址匹配标志位

0：地址不匹配；

1：地址匹配；

此标志位用于决定从机地址是否与主机发送地址相同。若地址匹配此位为高，否则此位为低。

Bit 5 HBB：I2C 总线忙标志位

0：I2C 总线闲；

1：I2C 总线忙；

当检测到 START 信号时 I2C 忙，此位变为高电平。当检测到 STOP 信号时 I2C 总线停止，该位变为低电平。

Bit 4 SRW：I2C 从机读/ 写位

0：从机应处于接收模式；

1：从机应处于发送模式；

SRW 位是从机读写位。决定主机是否希望传输或接收来自 I2C 总线的数据。当传输地址和从机的地址相同时，主机将检测 SRW 位来决定进入发送模式还是接收模式。如果 SRW 位为高时，主机会请求从总线上读数据，此时设备处于传输模式。当 SRW 位为“0”时，主机往总线上写数据，设备处于接收模式以读取该数据。

I2C 传送重新启动时，SRW 会自动变成 0。

Bit 3 TXAK：I2C 总线在接收模式时的ACK位

0：从机发送确认标志

1：从机没有发送确认标志

这是作为从机接收模式的响应信号。在从机接收 8 位数据之后，将该位在第九个时钟传到总线上。如果单片机想要接收更多的数据，则应在接收数据之前将此位设置为“0”，如果不想继续接收数据，则把该位置 1。但在第一帧接收地址时，系统会自动根据地址匹配结果做出响应，和该位的值无关。

I2C 传送重新启动时，TXAK 会自动变成 1。

Bit 2 I2CCKP：SCL时钟使能控制（从机接收模式专用，初值默认1）

0：将时钟SCL拉低，以延长时钟周期，来确保数据建立时间；

1：时钟SCL正常工作；

Bit 1 I2CPU I2C 总线SDA脚上拉使能

0：无上拉；

1：有上拉（电阻大概在5~20K之间）；

Bit 0 I2CEN I2C 总线使能

0：无效；

1：有效；

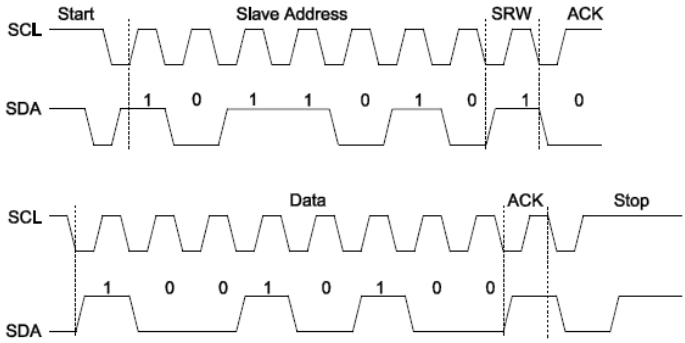
9.2 I2CADD/I2CBUF 寄存器

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
A9H(r/w)	I2CADD	AD7	AD6	AD5	AD4	AD3	AD2	AD1	-
AAH(r/w)	I2CBUF	读写数据缓冲器							

A7~A1 是从机地址对应的 6~0 位。寄存器 I2CADD 中的第 7~1 位是单片机的从机地址，位 0 未定义。如果接至 I2C 的主机发送处的地址和寄存器 I2CADD 中存储的地址相符，那么就选中了这个从机。

9.3 I2C 起始和终止信号

I2C 总线上的通信需要四步完成，一个起始信号，一个从机地址发送，一个数据传输，还有一个停止信号。当起始信号被写入 I2C 总线时，总线上的所有从机都会接收到这个起始信号并且被通知总线上会即将有数据到达。数据总在 SCL 的高电平读入，高位在前，低位在后。



一旦 I2C 模块被使能，处于从机模式 I2C 模块会一直等待启动信号的出现。检测到启动信号后，会把 SDA 脚上的信息在 SCL 的上升沿采样逐步移入 I2CSR 寄存器，在 SCL 的第 8 个下降沿，把 I2CSR<7:1>的地址和 I2CADD 的内容进行比较，如果匹配，并且 HCF=0 时，将自动发生以下操作：

1、第8个下降沿，把I2CSR装入I2CBUF；

2、第8个下降沿，把HCF=1；

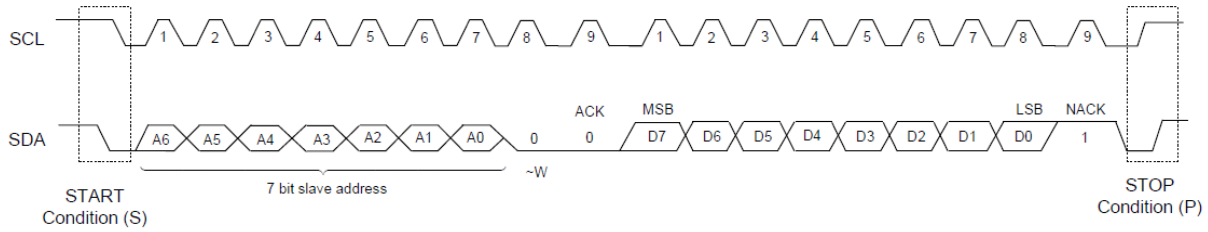
3、产生一个应答信号ACK；

4、第9个下降沿，置中断标志位I2CIF=1；

9.4 从机接收

当接收到的地址字节中的读写位 R/W 为 0，且 7 位地址码和本机匹配时，I2CCON 中的读写位 R/W 会自动清 0，同时读到的地址会自动装入到 I2CBUF 中，且自动回复一个 ACK。

之后数据传输中，如果 TXAK=1，将导致接收器在第 9 个时钟周期期间不会发送有效应答位（即 SDA=1）。I2C 模块在每一个字节传输之后，都会将 I2CIF 置 1（I2CIF 须软件清 0）。如果主机收到响应应答位不等于 0，则可能会终止数据传输过程。是否发送有效应答位不影响 I2CBUF 数据的接收。即如果从机不再接收数据，则 TXAK 不清 0 即可。

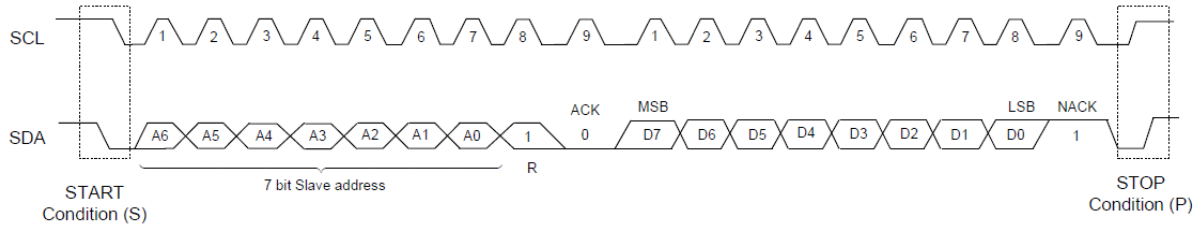


上图中的 ACK 为自动产生的，NACK 为 TXAK 的值。如果地址不匹配，则第一个响应为 NACK，但如果主机继续发送数据，从机也会继续接收数据。

9.5 从机发送

当接收到的地址字节中的读写位 R/W 为 1，且地址匹配时，读写位 R/W 会自动置 1，同时读到的地址会自动装入到 I2CBUF 中。在第 9 个时钟周期期间从机回复一个有效应答位/ACK，从机会自动把 SCL 锁定在低电平上（同时时钟使能位 CKP 被自动清 0），以插入等待时间。（插入等待时间的目的是从机有时间处理数据，数据处理完后才能允许数据传送进行）。

用户程序须把发送的字节存入 I2CBUF 中，同时 HCF 自动置 0，然后用软件把 CKP 置 1，来释放对 SCL 的锁定，使其恢复工作。从机通过这种强行拉长 SCL 低电平时间的手段来迫使主机与之同步。主机产生下一个脉冲之前，必须检测 SCL 的逻辑电平，只有 SCL 为高时才可继续发送。如果发送之前没有写 I2CBUF 寄存器，则发送的数据是上次已写入的。



从机发送模式时，I2CIF 标志也在第九个时钟的下降沿被置 1。

作为从机发送器，在 SCL 输入时钟的第九个时钟锁存主机发送来的应答信号。根据应答信号的电平，从机有两种反应方式：

如果 ACK=1，即不应答，表明主机需要的全部数据已经传输完毕，从机会放开数据总线，主机可以发送停止信号。如果主机不发送停止信号，继续想要接收数据的话，从机也不会继续发送数据，只会发送 0xffh。

如果 ACK=0，即有效应答，表明主机需要的全部数据尚未传输完毕，从机会锁定数据总线，继续发送数据，主机须继续接收。

10 配置 config

Config0：

位	名称	说明
2~0	Fosc<2:0>	振荡源选择位 = 0 0 0→ERC mode (external R & C) (默认) = 1 1 0→ HF mode = 1 0 1→XT mode = 1 0 0→LF mode = 0 1 1→IRC mode (internal R & C) = 0 0 1→ERIC mode (external R & internal C) = 1 1 1→IRC_RTC mode (RTC 模式可以作为 TM1\TM2\TM3 的时钟源)
5~3	LVDT<2:0>	低电压检测选择位 = 0 0 0→禁止低电压检测(默认) = 0 0 1→enable， LVDT voltage = 2.0V， 睡眠模式关闭 = 0 1 0→enable， LVDT voltage = 2.0V = 0 1 1→enable， LVDT voltage = 3.6V = 1 0 0→enable， LVDT voltage = 2.9V = 1 0 1→enable， LVDT voltage = 2.2V = 1 1 0→enable， LVDT voltage = 2.4V = 1 1 1→enable， LVDT voltage = 2.6V
9~6	WDT_SEL<3：0>	= 0101 Enable,OFF WHEN SLEEPING = 1010 disable = others Always_on
12~10	FCPU<2:0>	= 000 $F_{CPU}=F_{OSC}/2$ = 001 $F_{CPU}=F_{OSC}/4$ = 010 $F_{CPU}=F_{OSC}/8$ = 011 $F_{CPU}=F_{OSC}/16$ = 100 $F_{CPU}=F_{OSC}/32$ = 101 $F_{CPU}=F_{OSC}/64$
14~13	PWRT	PWRT 上电延时选择 =00：40ms =01：72ms =10：168ms =11：392ms
15	RSTBIN	P13/RSTB 选择位置 = 0， P13 (默认) = 1， RSTB，内部上拉

Config1：

位	名称	说明
7~0	IHRC	内置振荡器修调
8	IHRC2X	IHRC 的频率是否加倍 = 0，不加倍 = 1，加倍。最快达到 24M/2 分频
9	POWER_LOW	= 0， Disable = 1， enable 低功耗模式，主频小于 1MHZ 时，可以选用
10	security	= 0， Disable = 1， enable flash 加密
11	FLUSH_RDS	0：1/2*Fcpu 1：1/4*Fcpu
15-14	x0：IHRC，有修调 8 位 01：ILRC，没有修调	IHRC 为内置高精度高频振荡器

Config2：

位	名称	说明
0	P1_DRIVE_BT	P1 端口驱动能力选择 = 0，DRV0=30mA，DRV1=15mA； = 1，DRV0=80mA，DRV1=40mA；
1	P3_DRIVE_BT	P3 端口驱动能力选择（不含 P30 端口） = 0，DRV0=30mA，DRV1=15mA； = 1，DRV0=80mA，DRV1=40mA；
11:2	未用	
12	UVCK	=0，disable UVCK =1，enable UVCK

11 指令集

指令	指令格式	描述	C	DC	Z	周期
MOVE	MOV A,M	A ← M。	-	-	√	1
	MOV M,A	M ← A。	-	-	-	1
	B0MOV A,M	A ← M (bank 0)。	-	-	√	1
	B0MOV M,A	M (bank 0) ← A。	-	-	-	1
	MOV A,I	A ← I	-	-	-	1
	B0MOV M,I	M ← I。（M 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
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
	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
	B0ADD M,A	M (bank 0) ← M (bank 0) + A，如果产生进位则 C=1，否则 C=0。	√	√	√	1
	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
	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。	√	√	√	
	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
	AND A,I	A ← A 与 I。	-	-	√	1
	OR A,M	A ← A 或 M。	-	-	√	1
	OR M,A	M ← A 或 M。	-	-	√	1
	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
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
	RRC M	A ← M 带进位右移。	√	-	-	1
	RRCM M	M ← M 带进位右移。	√	-	-	1
	RLC M	A ← M 带进位左移。	√	-	-	1
	RLCM M	M ← M 带进位左移。	√	-	-	1
	CLR M	M ← 0。	-	-	-	1
	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
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+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+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，使能全局中断控制位。	-	-	-	2
	PUSH	进栈指令，保存 ACC 和工作寄存器。	-	-	-	1
	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