FH5224/5234

用户参考手册

ZJFH 8 位单片机

功能特色：

- **4K OTP 程序存储器**
- **384Byte SRAM**
- **1 路 PWM，大驱动**
- **512 Byte E2PROM**

修正记录

版本号	日期	内容
VER 1.0	2018 年 5 月	初版

目录

1 产品简介 6
1.1 功能特性 6
1.2 系统结构框图 7
1.3 引脚配置 7
1.4 引脚说明 8
2 中央处理器（CPU） 9
2.1 程序存储器（ROM） 9
2.1.1 复位向量（0000H） 9
2.1.2 中断向量（0008H） 9
2.1.3 查表 10
2.1.4 跳转表 11
2.1.5 CHECKSUM 计算 12
2.2 数据存储器（RAM） 14
2.2.1 系统寄存器 14
2.2.2 累加器 16
2.2.3 程序状态寄存器 PFLAG 16
2.2.4 程序计数器 17
2.2.5 Y,Z 寄存器 17
2.2.6 R 寄存器 18
2.2.7 RAM 页面控制寄存器(RBANK) 18
2.2.8 OPTION 寄存器 18
2.2.9 OPTDRV 寄存器 19
2.2.10 PTCON 寄存器 20
2.3 堆栈 21
2.3.1 概述 21
2.3.2 堆栈寄存器 21
3 复位 22
3.1 概述 22
3.2 上电复位 22
3.3 看门狗复位 22
3.4 掉电复位 23
3.4.1 概述 23
3.4.2 系统工作电压 24
3.4.3 低电压检测 LVD 24
3.4.4 掉电复位性能改进 24
3.5 外部复位 25
4 系统时钟 26
4.1 概述 26
4.2 指令周期 FCPU 26
5 系统工作模式 27
5.1 概述 27
5.2 普通模式 28
5.3 低速模式 28
5.4 睡眠模式 28
5.5 绿色模式 29

5.6 工作模式控制宏 29
5.7 系统唤醒 30
5.7.1 概述 30
5.7.2 唤醒时间 30
6 中断 31
6.1 概述 31
6.2 中断请求使能寄存器 INTEN 31
6.3 中断请求寄存器 INTRQ 32
6.4 GIE 全局中断 32
6.5 PEDGE 33
6.6 INTN 中断 33
7 I/O 口 34
7.1 I/O 口模式 34
7.2 输入上拉寄存器 34
7.3 唤醒寄存器（端口变化唤醒） 34
7.4 端口寄存器 34
7.5 端口扫描寄存器 34
8 定时器 35
8.1 看门狗定时器 35
8.2T0 定时/计数器 36
8.2.1 T0M 36
8.2.2 T0C 36
8.2.3 T0R 36
8.2.4 T0D 37
8.2.5 PWM 37
8.3T1 定时/计数器 38
8.3.1 T1M 38
8.3.2 T1C 38
8.3.3 T1NEG (PWM00 占空比寄存器) 38
8.3.4 T1POS (PWM01 占空比寄存器) 39
9 E2PROM 控制器 40
9.1 E2PCON(E2PROM 控制寄存器) 40
9.2 E2PADR(E2PROM 地址寄存器) 40
9.3 E2PDB(E2PROM 数据寄存器) 40
10 内置运算放大器 42
10.1 OPA1 运放控制器 42
11 配置 CONFIG 43
CONFIG0： 43
CONFIG1： 44
12 指令集 45

1 产品简介

1.1 功能特性

储存器配置

◆OTP ROM：4K*16
◆SRAM：384*8
◆E2PROM：512*8
◆8 层堆栈

振荡器

◆高精度内置 RC 16MHz
◆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，带 1 路 PWM
◆1 路 16 位定时器/计数器，带捕捉功能
◆2 路外部中断，8/2 个 IO 口可选
◆6 个中断源

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 2 个 8/16 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

◆2.0v ∽ 3.5v Fcpu<2MHz
◆2.3v ∽ 3.5v Fcpu<4MHz
◆2.8v ∽ 3.5v Fcpu<8MHz

工作温度

◆-40∽85℃

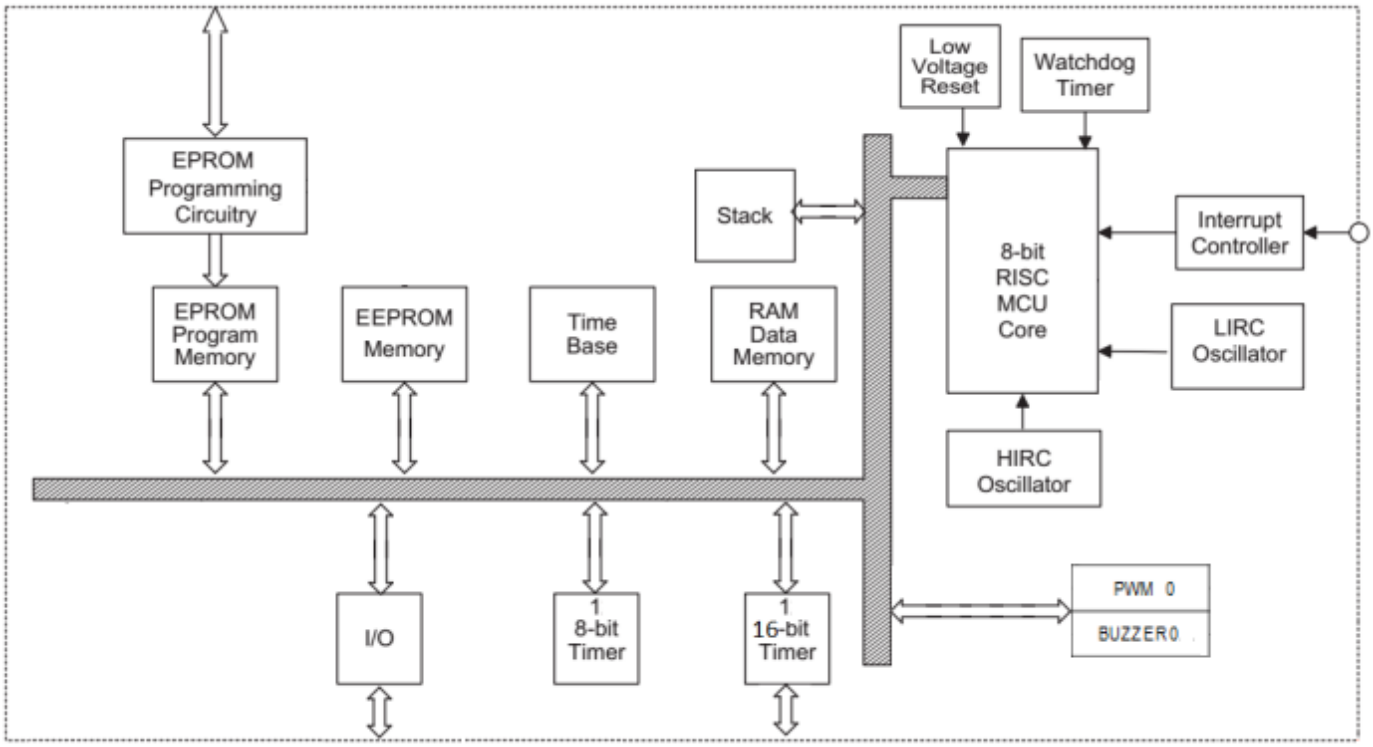
封装

◆SOP16/DIP16/TSSOP16

应用领域

◆小家电
◆玩具
◆电机控制

1.2 系统结构框图



1.3 引脚配置

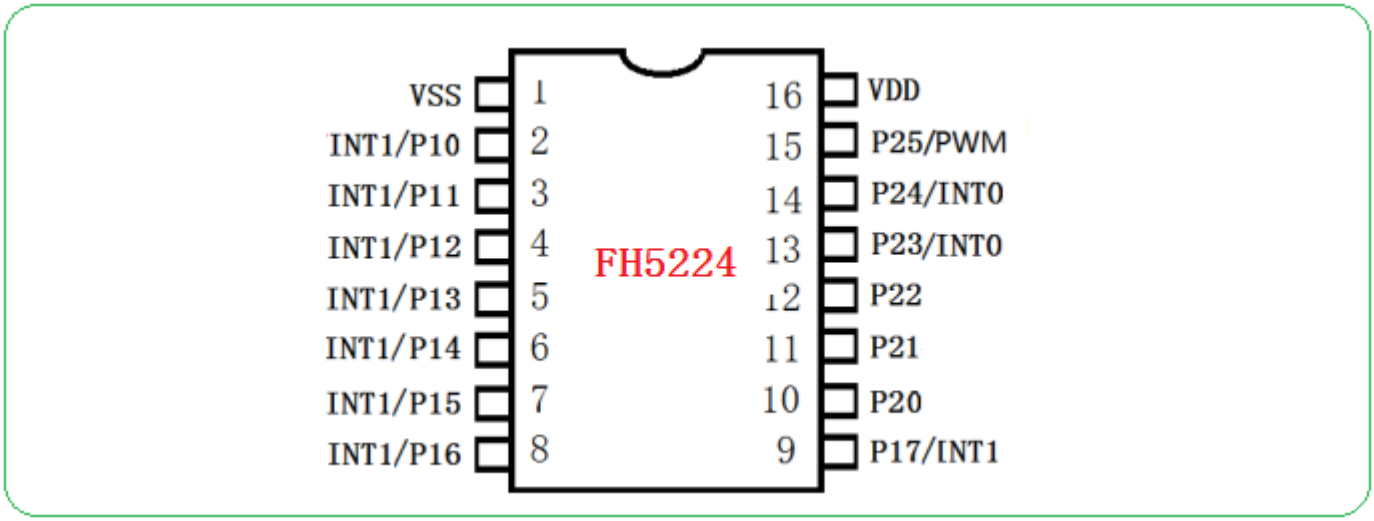


图 1 16 PIN 管脚分配

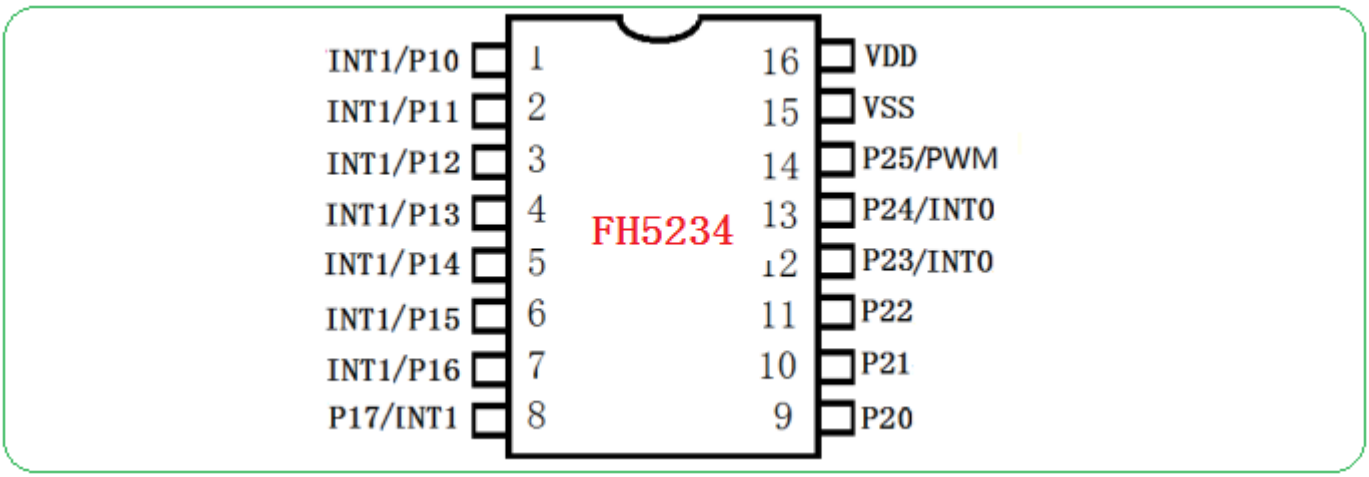


图 2 16 PIN 管脚分配

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；
P10~P17	I/O	双向输入输出脚，输入时施密特触发，内置上拉电阻；
P20~P25	I/O	双向输入输出脚，输入时施密特触发，内置上拉电阻；
VPP	I	OTP 烧写高压脚
PWM	O	1 路 PWM 输出脚
SCL,SDAIN,SDAOUT	I/O	OTP 烧写脚
INT0/INT1	I	外部中断 0/1 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）

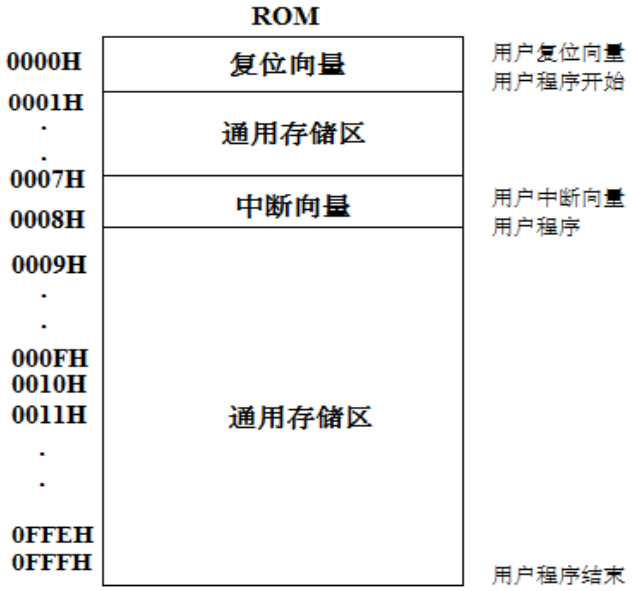
ROM：

1、有 12 位 PC，4K*16bit 的 OTP；

2、复位地址 0000h；

3、硬件中断地址 0008h；

4、JMP、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　ZJFH 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 FH5224/5234 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
```

FH5224 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
```

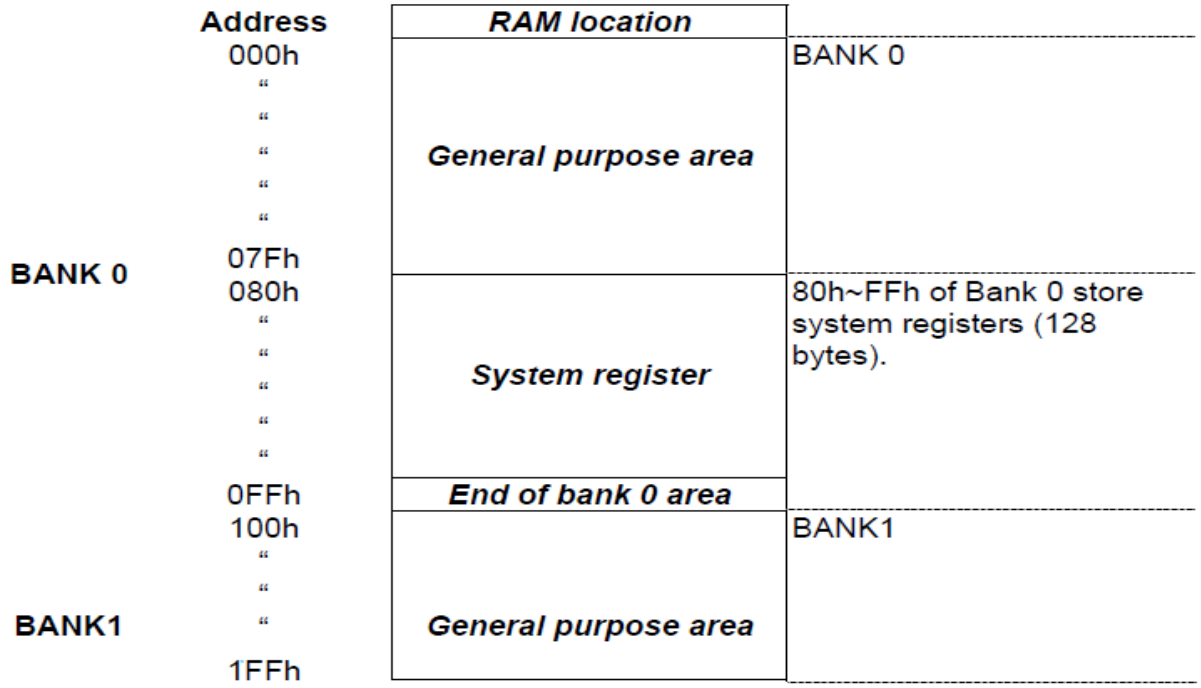
2.1.5 CHECKSUM 计算

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	RBANK								
9	T1NEGL	T1NEGH	T1P0SL	T1P0SH	T0D	T0R	OPTION	PTCON	E2PCON							
A				P1TM	P2TM	OPACON	P2W	E2PADH	E2PADL							
B					E2PDB											PEDGE
C	P1W	P1M	P2M						INTRQ	INTEN	OSCM	RCADJ	WDTR	OPTDRV	PCL	PCH
D		P1	P2						T0M	T0C	T1M	T1CL	T1CH			STKP
E		P1UR	P2UR					@YZ								

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
090H	下降沿捕捉 T1，低位								R	T1NEGL
091H	下降沿捕捉 T1，高位								R	T1NEGH
092H	上升沿捕捉 T1，低位								R	T1POSL
093H	上升沿捕捉 T1，高位								R	T1POSH
094H	T0 PWM 低电平占空比控制寄存器								R/W	T0D
095H	T0 定时器寄存器								R/W	T0R
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	PG8B	-	-	-	E2P_CER	E2P_SER	E2P_PG	E2P_RD	R/W	E2PCON
0A3H	P17TM	P16TM	P15TM	P14TM	P13TM	P12TM	P11TM	P10TM	R/W	P1TM
0A4H			P25TM	P24TM	P23TM	P22TM	P21TM	P20TM	R/W	P2TM
0A5H	OPAE	SYNEN	IRINO	IRINT 溢出时间控制			OPAI1	OPAI0	R/W	OPACON
0A6H	唤醒寄存器								R/W	P2W
0A7H			-	-	-	-	-	E2PADH0	R/W	E2PADH

0A8H	E2PADL7	E2PADL6	E2PADL5	E2PADL4	E2PADL3	E2PADL2	E2PADL1	E2PADL0	R/W	E2PADL
0B4H	E2PDB7	E2PDB6	E2PDB5	E2PDB4	E2PDB3	E2PDB2	E2PDB1	E2PDB0	R/W	E2PDB
0BFH	INT1S2	INT1S1	INT1S0	INT1G1	INT1G0	INT0G1	INT0G0	INT0S	R/W	PEDGE
0C0H	唤醒寄存器								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			E2PIF	RPIF	T1IF	T0IF	INT1IF	INT0IF	R/W	INTRQ
0C9H			E2PIE	RPIE	T1IE	T0IE	INT1IE	INT0IE	R/W	INTEN
0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CBH									R/W	RCADJ
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	R/W	WDTR
0CDH	RCADJEN	IRINT_SEL	PULLS_P2	PULLS_P1	DRVP24_1	DRVP24_0	DRVP11_0	DRVP11_0	R/W	OPTDRV
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH					PC11	PC10	PC9	PC8	R/W	PCH
0D1H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D2H			P25	P24	P23	P22	P21	P20	R/W	P2
0D8H	T0TR	T0PS2	T0PS1	T0PS0	T0PT1	T0PT0	T0CS	PWMOUT	R/W	T0M
0D9H	T0 定时器寄存器								R/W	T0C
0DAH	TR1	TP1S2	TP1S1	TP1S0	T1PT1	T1PT0	T1GN	T1TRSEL	R/W	T1M
0DBH	定时器低 8 位								R/W	T1CL
0DCH	定时器低 8 位								R/W	T1CH
0DFH	GIE					STKPB2	STKPB1	STKPB0	R/W	STKP
0E1H	P17UR	P16UR	P15UR	P14UR	P13UR	P12UR	P11UR	P10UR	R/W	P1UR
0E2H			P25UR	P24UR	P23UR	P22UR	P21UR	P20UR	R/W	P2UR
0E7H									R/W	@YZ

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 FLASH 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	LVDEN	LVDS2	LVDS1	LVDS0	T1LTS	WDTP2	WDTP1	WDTP0
RESET		0	0	0	0	1	1	1	1

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

111：1.75V;
110：1.91V;
101：2.05V;
100：2.15V;
011：2.24V;
010：2.43V;
001：2.57V;
000：2.73V;

注：如果电源电压低于选定的值，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**	1:2**(16ms)**
0	**0**	**1**	1:4
0	**1**	**0**	1:8
0	**1**	**1**	1:16
1	**0**	**0**	1:32
1	**0**	**1**	1:64
1	**1**	**0**	1:128
1	**1**	**1**	1:256

2.2.9 OPTDRV 寄存器

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
CDH(r/w)	OPTDRV	RCADJEN	IRINT_SEL	PULLS_P2	PULLS_P1	DRVP24_1	DRVP24_0	DRVP11_0	DRVP11_0
RESET		0	0	0	0	0	0	0	0

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(Y,Z 初始值为 2000h)。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。

IRINT_SEL:

0:IRINT 从 IRIN 第一个下降沿开始有效

1:IRINT 从 IRIN 第一个上升沿开始有效。

PULLSEL_P2:PORT2 端口上拉电阻大小选择

=0：80k

=1：20k

PULLSEL_P1:PORT1 端口上拉电阻大小选择

=0：80k

=1：20k

DRVP24<1:0>：PORT24 端口高低电平驱动的驱动能力选择

=11　低电平：20mA；高电平：10mA

=10　低电平：10mA；高电平：5mA

=01　低电平： 5mA；高电平：3mA

=00　低电平： 2mA；高电平：1mA

DRVP11<1:0>：PORT11 端口高低电平驱动的驱动能力选择

=11　低电平：20mA；高电平：10mA

=10　低电平：10mA；高电平：5mA

=01　低电平：5mA；高电平：3mA

=00　低电平：2mA；高电平：1mA

2.2.10 PTCON 寄存器

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
97H(r/w)	PTCON	SSEL0	DKWE	T1WDS	T1WDL	POWER	IRCON	DSEL1	DSEL0
RESET		0	0	0	1	1	1	1	1

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 ;

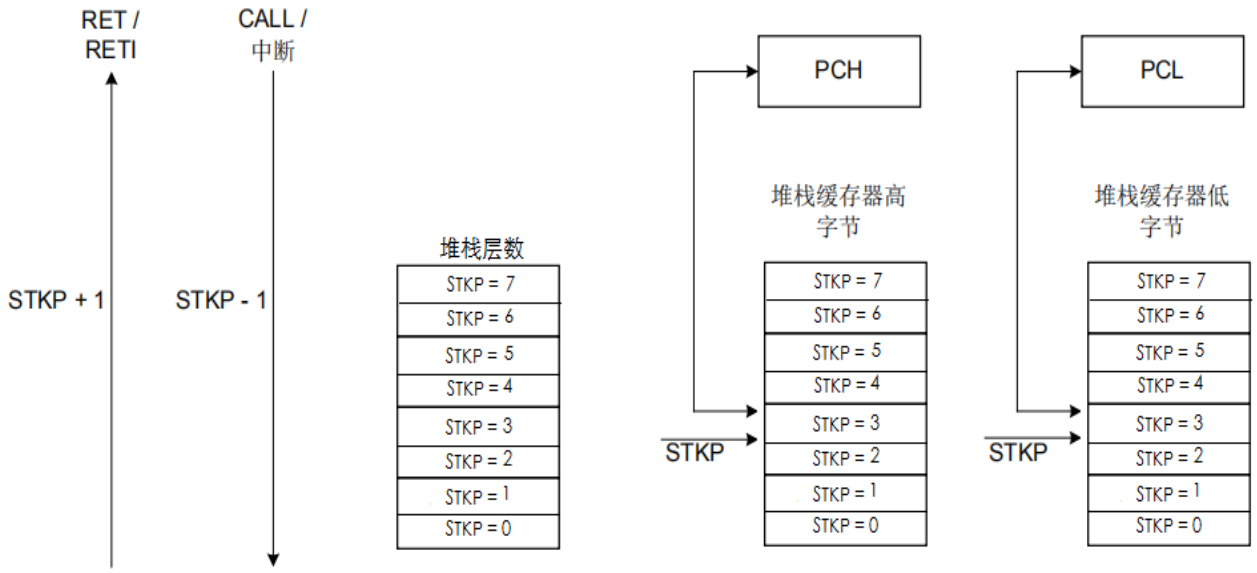
=01　300mA;

=00　500mA;

2.3 堆栈

2.3.1 概述

FH5224/5234 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

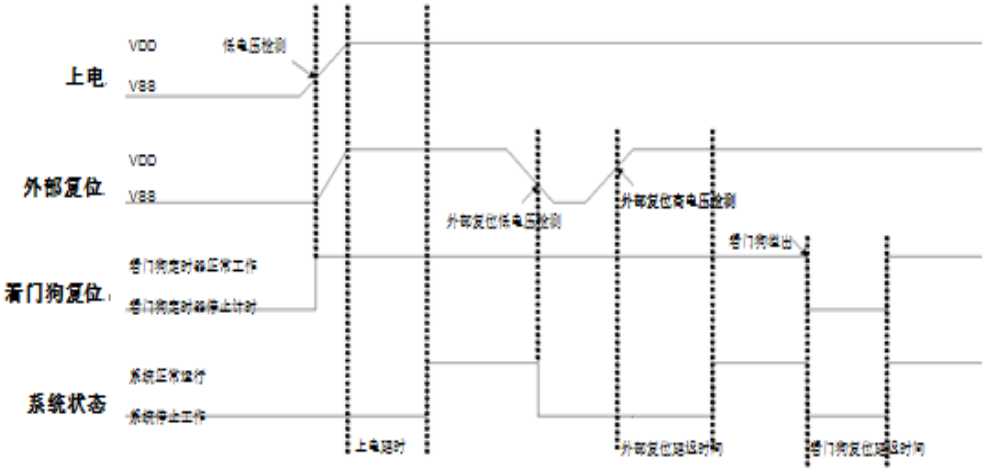
3.1 概述

FH5224/5234 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；

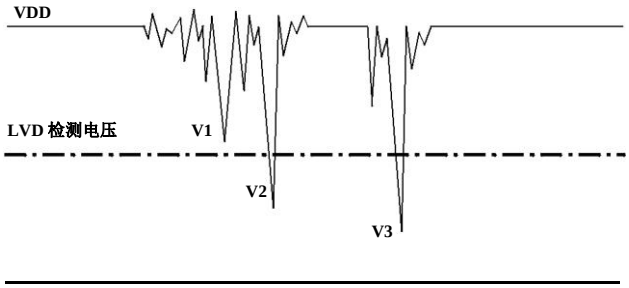
看门狗定时器应用注意事项：

- 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。

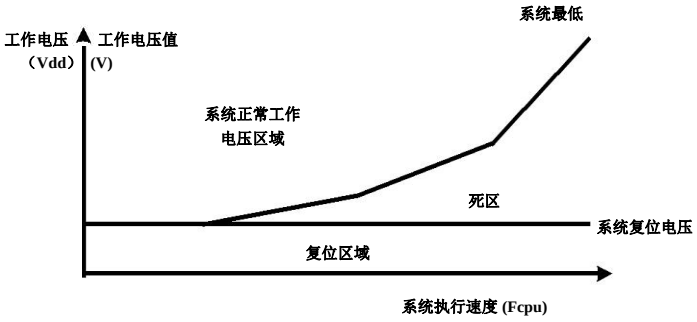
AC 运用中：

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

FH5224/5234 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 16MHz，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。

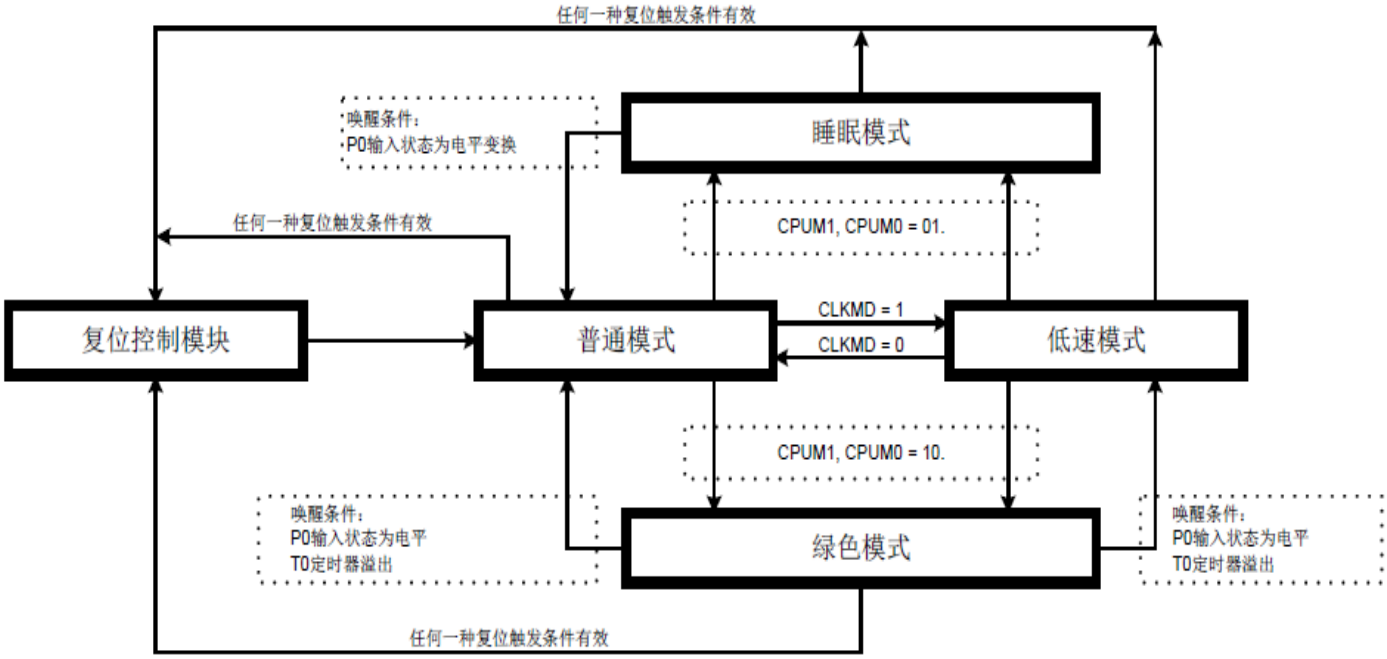
5 系统工作模式

5.1 概述

FH5224/5234 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	TR0	TR0	停止	停止
T1	TR1	TR1	可控	可控
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

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，P2 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，P2 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，P2 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

FH5224/5234 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
```

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，P2 的电平变换）和内部触发（T0 定时器溢出）。

- 从睡眠模式唤醒后只能进入普通模式，且将其唤醒的触发只能是外部触发信号（P0 电平变化）；
- 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，P2 电平变换）和内部触发信号（T1 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：

$$\text{唤醒时间} = 2048 * T_{OSC}$$

6 中断

6.1 概述

FH5224/5234 提供 6 个中断源：3 个内部中断和 3 个外部中断。IR，E2PROM 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			E2PIE	RPIE	T1IE	T0IE	INT1IE	INT0IE
R/W	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	0

E2PIE:E2P 写或者擦除时间到中断使能

RPIE: 端口变化中断使能

T1IE: TM1 溢出中断使能

T0IE: 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

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			E2PIF	RPIF	T1IF	T0IF	INT1IF	INT0IF
R/W	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	0

0C6H	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
INTRQ1	-	-	-	OPAIF	T0IF	SPIIF	TKTMRIF	E2PIF
R/W				R/W	R/W	R/W	R/W	R/W
复位				0	0	0	0	0

中断标志：

= 0：中断无效；

= 1：中断有效，软件清 0；

其中 INT0IF、INT1IF、T0IF（必须选用低速时钟）、RPIF 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 P24 口作为 INT0

1：选择 P23 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=P17;111=P10;

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
            …
```

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	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
C2H(r/w)	P2M	-	-	P25M	P24M	P23M	P22M	P21M	P20M
reset		0	0	0	0	0	0	0	0

PnM[7:0]：Pn 模式控制位（n=1~2）

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	P17UR	P16UR	P15UR	P14UR	P13UR	P12UR	P11UR	P10UR
E2H(r/w)	P2UR	-	-	P25UR	P24UR	P23UR	P22UR	P21UR	P20UR
reset		0	0	0	0	0	0	0	0

PnUR[7:0]：Pn 上拉控制位（n=1~2）

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							
A6H(r/w)	P2W	唤醒寄存器							
reset		00h							

PnW[7:0]：P1，2，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	P17	P16	P15	P14	P13	P12	P11	P10
D2H(r/w)	P2	-	-	P25	P24	P23	P22	P21	P20
reset		0	0	0	0	0	0	0	0

7.5 端口扫描寄存器

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
A3H(r/w)	P1TM	P17TM	P16TM	P15TM	P14TM	P13TM	P12TM	P11TM	P10TM
A4H(r/w)	P2TM			P25TM	P24TM	P23TM	P22TM	P21TM	P20TM
reset		0	0	0	0	0	0	0	0

PnTM[7:0]：Pn 上拉控制位（n=1~2）

0 = 端口不进行扫描；

1 = 端口扫描（P25TM 无法控制 PORT25 端口进行扫描，但可以使低速振荡器起振，用作 T0 的时钟源）；

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 = 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 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 8ms。

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
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
```

8.2T0 定时/计数器

8.2.1 T0M

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
0D8h(r/w)	T0M	T0TR	T0PS2	T0PS1	T0PS0	T0PT1	T0PT0	T0CS	PWMOU

PWMOUT:

0：PWM不能输出(T0变成16位的，此时T0D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	Fx（T0CS=0）	Fx（T0CS=1）
00	Fhosc	Fhosc
01	Flosc	Flosc
10	Fcpu	Fcpu
11	IRINS	INT0

注：选用 Flosc 做时钟源时，可在睡眠状态下唤醒。

T0PS[2:0]: TCn分频选择位

= 000，Fcpu/1

= 001，Fcpu/2

= 010，Fcpu/4

= 011，Fcpu/8

= 100，Fcpu16

= 101，Fcpu/32

= 110，Fcpu/64

= 111，Fcpu/128

T0TR: T1启动控制位

= 0，禁止T0定时器

= 1，开启T0定时器

8.2.2 T0C

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							

8.2.3 T0R

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
95h(r/w)	T0R	T0 自动装载寄存器							
reset	00	0000 0000							

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。

8.2.4 T0D

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
94h(r/w)	T0D	T0 PWM 低电平占空比控制寄存器							
reset	00	0000 0000							

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。

8.2.5 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。

8.3T1 定时/计数器

8.3.1 T1M

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
DAh (r/w)	T1M	TR1	TP1S2	TP1S1	TP1S0	T1PT1	T1PT0	T1GN	T1TRSEL
reset		0	0	0	0	0	0	0	0

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

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，如果T1时钟源是T0的溢出信号，则必须把T0IE关闭）

T1PT1,T1PT0：

T1PT1，T1PT0	T1 时钟源选择
00	Fhosc
01	T0 溢出
10	Fcpu
11	INT1

T1PS[2:0]: T1分频选择位

= 000，Ft1/1

= 001，Ft1/2

= 010，Ft1/4

= 011，Ft1/8

= 100，Ft1/16

= 101，Ft1/32

= 110，Ft1/64

= 111，Ft1/128

TR1: T1启动控制位

= 0，禁止T1定时器

= 1，开启T1定时器

8.3.2 T1C

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
DBh (r/w)	T1CL	定时器低 8 位							
DCh (r/w)	T1CH	定时器高 8 位							
reset	00h	00h							

16 位计数器为加计数 T1C 溢出时，T1IF 置 1，需软件清零，用来控制 T1 的中断间隔时间。

8.3.3 T1NEG (PWM00 占空比寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
90h (r)	T1NEGL	下降沿捕捉 T1，低位							
91h (r)	T1NEGH	下降沿捕捉 T1，高位							
reset	00h	00h							

8.3.4 T1POS (PWM01 占空比寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
92h (r)	T1POSL	上升沿捕捉 T1，低位							
93h (r)	T1POSH	上升沿捕捉 T1，高位							
reset	00h	00h							

T1 信号源捕捉有 T1LTS 确定。

T1LTS: T1捕捉器的源选择端

0：IRINS

1：IRINT

9 E2PROM 控制器

9.1 E2PCON(E2PROM 控制寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
98h (r/w)	E2PCON	PG8B	-	-	-	E2P_CER	E2P_SER	E2P_PG	E2P_RD
reset		0				0	0	0	0

PG8B: 编程时同时编程8bit数据

0：按bit编程，需1ms

1：编程按byte，需0.15ms。(VDD>3v)

注：选择按位编程后，则最低编程电压会达到2.0v。

E2P_CER:E2PROM的chip擦除使能

0：擦除完成

1：擦除启动，擦完后自动清0

E2P_SER: E2PROM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E2PADH,E2PADL的高5位地址确定。

E2P_PG: E2PROM的PG使能

0：编程完成

1：编程启动，擦完后自动清0

E2P_RD: E2PROM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E2PDB寄存器的值

9.2 E2PADR(E2PROM 地址寄存器)

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
A7h (r/w)	E2PADH				-	-	-	-	E2PADH0
A8h (r/w)	E2PADL	E2PADL7	E2PADL6	E2PADL5	E2PADL4	E2PADL3	E2PADL2	E2PADL1	E2PADL0

Bit8:BIT0: 内置E2PROM地址，地址000H~1FFH共512字节。

9.3 E2PDB(E2PROM 数据寄存器)

地址Bank1	名称	B7	B6	B5	B4	B3	B2	B1	B0
B4h (r/w)	E2PDB	E2PDB7	E2PDB6	E2PDB5	E2PDB4	E2PDB3	E2PDB2	E2PDB1	E2PDB0

E2PROM数据输入寄存器，在烧写内置E2PROM前将要写的数据写入E2PDB寄存器中。读E2PROM时，用MOV指令直接读入ACC中。

操作E2PDB时，最好使用MOV指令，不用使用运算指令，如INCMS，DECMS等。

E2PROM操作例程：

```
MOV   A,#40h
MOV  E2PROMCON,A        ; 延时100us，100ms，按bit编程
CLR  E2PADH
MOV  A,#02h
MOV MTTADL              ; 对E2PROM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   E2PROM_STE
CHECK_STE:
                BTS0   E2PROM_STE
                JMP    CHECK_STE        ; 需要100ms擦除，小心wdt复位
                MOV  A,#55H
                MOV  E2PDB,A            ; 赋值要写入的数据
LOOP_PG:
                BSET   E2PROM_PG
CHECK_PG:
                BTS0     E2PROM_PG
                JMP      CHECK_PG
PG_END:
                BSET   E2PROM_RD        ; 启动读
                MOV    A,E2PROM_DB      ; E2PROM的数据送入A
                MOV    TEMP1，A
```

10 内置运算放大器

10.1 OPA1 运放控制器

地址	名称	B7	B6	B5	B4	B3	B2	B1	B0
0A5h(r/w)	OPACON	OPAE	SYNEN	IRINO	IRINT 溢出时间控制			OPAI1	OPAI0

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。

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;

IRINO

IRINO=0:无效;

IRINO=1：有效，P12输出和IR接收到的波形一致;

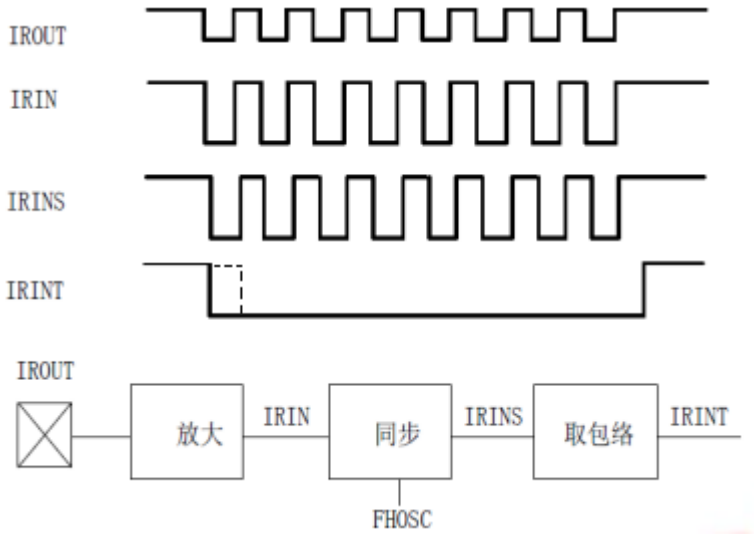
Bit4~bit2:IRINT溢出时间控制

000:16us 001:24us 010:32us 011：40us

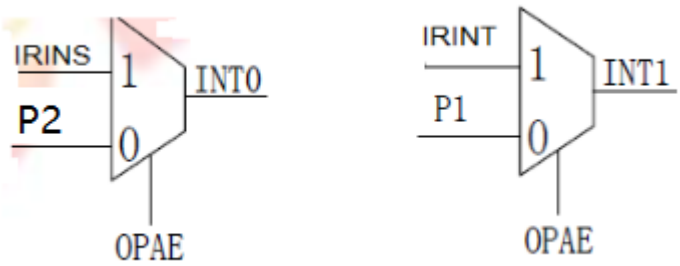
100:48us 101:56us 110:64us 111：96us

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;



注：当 OPAE 使能时，INT0 和 INT1 的输入分别连置 IRINS 和 IRINT。如下图：



11 配置 config

Config0：

位	名称	说明
1~0	IRAMP	=00：低 =01：高 =10：较高 =11：最高
2	LVD_SLEEP	0：LVR 选择复位,SLEEP 状态仍然有效。 1：LVR 选择复位，sleep 状态时无效。
5~3	LVDT<2:0>	低电压检测选择位 = 0 0 0→禁止低电压检测复位(默认) = x 0 1→enable， LVDT voltage = 2.1V = x 1 0→enable， LVDT voltage = 1.8V = x 1 1→enable， LVDT voltage = 2.6V = 1 0 0→enable， LVDT voltage = 2.4V
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	OTP_RDS （OTP 读取信号宽度）	0：1/2*Fcpu 1：1/4*Fcpu（读取宽度越宽，低压特性越好，但工作电路会大）

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 16M/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

12 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