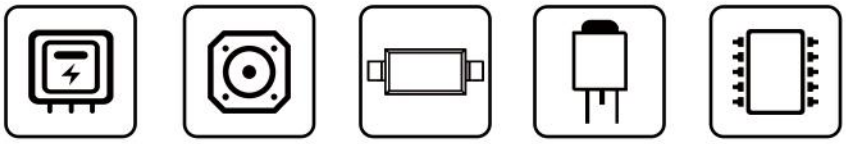
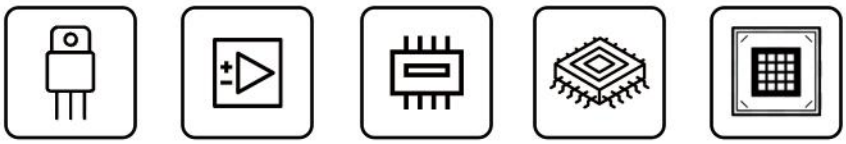
TDSEMIC
拓電半導體

自主封測 品質把控 售後保障

WEB | WWW.TDSEMIC.COM





TM56F8225-TD

產品規格說明書

TM56F8225

电瓶车,电动工具
充电控制专用芯片

规格书

版本 0.93

修改记录

版次	生效日	修订内容概要
0.90	Sep,2020	新颁
0.91	Dec,2020	1.修正 ADVREFS 默认值 (P.74) 2.增加 VR 对温度特性曲线 (P.96) 3.增加建议 EEPTE 设置数值
0.92	Oct,2021	1.修正了间接寻址描述中的拼写错误（第 18 页） 2.修正了示例代码中的拼写错误（第 21 页） 3.修订示例代码中的说明和拼写错误（第 23 页） 4.修订 INT1IF 说明（第 36 页） 5.添加特殊功能寄存器 OPTION 表来说明中断（37 页） 6.修订了输出模式示例的说明（第 39 页） 7.修订了清除看门狗定时器的说明（第 42 页） 8. 修正了特殊功能寄存器 OPTION 表中的打字错误（第 43 页） 9.修正了 timer0 在计数器模式下工作的时序图（第 47 页） 10.修正了 PWM1AOE1 描述中的拼写错误（第 76 页） 11.修正了 CALL 指令描述中的错误（第 85 页） 12.修正了 GOTO 指令描述中的错误（第 86 页） 13.修正了 MOVX 和 MOVXW 指令描述中的拼写错误（第 88 页） 14.修正了 SUBLW 指令的固定描述（第 91 页） 15.修正了示例代码解释中的拼写错误（第 56 页）
0.93	Feb, 2022	1. 增加描述：复位后 PWM0 时钟是使能的而且 PWM0 不会保持固定（第 54 页） 2. 增加描述：复位后 PWM1 时钟是使能的而且 PWM1 不会保持固定（第 60 页）

目录

特性

1. **ROM: 2K x 16 位 Flash 程序存储器**
   - 至少 10K 次擦除
   - 至少 10 年的数据保存
2. **EEPROM: 128 x 8 位**
   - 至少 50K 次擦除
   - 至少 10 年的数据保存
3. **RAM: 176 x 8 位**
4. **STACK: 8 级**
5. **系统振荡源 (Fsys):**
   - 快速时钟：FIRC（内部快速 RC）：8 MHz
   - 慢速时：SIRC（内部慢速 RC）：70 KHz @VCC=5V
6. **系统时钟预分频器:**
   - 系统振荡源可除以 1 / 2 / 4 / 8 作为系统时钟 (Fsys)
7. **双系统时钟:**
   - FIRC + SIRC
8. **省电工作模式**
   - 快速模式:慢速时钟可以禁止或使能，CPU 在快速时钟下保持运行
   - 慢速模式: 快速时钟可以禁止或使能，CPU 在慢速时钟下保持运行
   - 空闲模式: 快速时钟和 CPU 停止。慢速时钟，T2 和 WKT 保持运行
   - 停止模式: 所有时钟停止，T2 和 WKT 停止运行
9. **3 个独立定时器**
   - Timer0
     - 8 位定时器，除以 1~32768 预分频选项 ，具有自动重载 / 计数器 /中断/ 停止功能
   - Timer1
     - 8 位定时器，除以 1~32768 预分频选项 ，具有自动重载 /中断/ 停止功能
   - T2
     - 15 位定时器，带有 4 个中断间隔时间选项
     - 空闲模式唤醒定时器或用作一个简单的 15 位时基
     - 时钟源: 慢速时钟 (SIRC), Fsys/128

10. 中断

- 三个外部中断引脚
  - 1 个引脚为下降沿唤醒触发并中断
  - 2 个引脚为上升或下降沿唤醒触发并中断
- Timer0 / Timer1 / T2 / WKT 中断
- ADC 中断
- PWM1 周期和 PWM1A / PWM1B / PWM1C 占空比中断

11. 唤醒定时器（WKT）

- 由内置 RC 振荡器提供时钟，具有 4 个可调节的中断时间
  - 16 ms / 33 ms / 65 ms / 130 ms @VCC=3V
  - 15 ms / 29 ms / 59 ms / 118 ms @VCC=5V

12. 看门狗定时器（WDT）

- 由内置 RC 振荡器提供时钟，具有 4 个可调的复位时间
  - 130 ms / 260 ms / 1040 ms / 2080 ms @VCC=3V
  - 118 ms / 236 ms / 944 ms / 1888 ms @VCC=5V
- 在停止模式下可以禁止/使能看门狗定时器

13. PWM x 4

- PWM0
  - 8 + 2 位，占空比可调，周期可调的 PWM
  - PWM0 时钟源：快速时钟或 FIRC 8 MHz / 16MHz，具有 1～8 个预分频器
  - 互补 PWM 输出（PWM0P，PWM0N）
  - 非重叠时间可调：（0～8）*（PWMCLK）
- PWM1A / PWM1B / PWM1C
  - 具有三组独立的占空比可调和共享周期可调的 16 位 PWM1
  - PWM1 共享时钟源：系统时钟（Fsys）或 FIRC 8 MHz / 16 MHz
  - 具有占空比和周期中断功能

14. 具有 8 个输入通道和 1 个内部参考电压的 12 位 ADC

- 内部参考电压 VR（3V ±1.2% @25°C, VCC=3V~5V）
- ADC 参考电压 = VCC 或 VR

15. 电池充电模块

- OPA x 2

- 10 位 DAC x 2
  - DAC 参考电压 = VR（3V ±1.2% @25°C, VCC=3V~5V）

16. 复位源

- 上电复位
- 看门狗复位
- 低电压复位
- 外部引脚复位

17. 低电压复位（LVR）/ 低电压检测标志（LVD）

- 4 级低电压复位：2.2V / 2.8V / 3.6V / 4.2V
- 3 级低电压检测：2.8V / 3.6V / 4.2V（当 LVR = 2.2V）

18.工作电压

- Fsys = 4 MHz，1.6V～5.5V @LVR 禁止。建议 LVR 在-40℃至+ 85℃时为 2.2V 或更高
- Fsys = 8 MHz，2.1V～5.5V @LVR 禁用。建议 LVR 在-40℃至+ 85℃时为 2.8V 或更高

注意：上电 VCC 必须超过 LVR 2.2V 和所选的 LVR 电平，请参阅“电气特性图”以避免进入 ROM 死区。

19. 工作温度范围：-40°C to + 85°C

20. 读表取指令：16 位 ROM 数据查找表

21. 集成的 16 位循环冗余校验功能

22. 指令集：39 条指令

23. I/O 端口：

- 最多 8 个可编程 I / O 引脚
  - 开漏输出
  - CMOS 推挽输出
  - 施密特触发器输入，带上拉电阻选项
  - 所有 I/O 具备 High-Sink 和 High-Drive
- 5 个 OPA 模拟 I / O 引脚
- 1 个参考电压输出引脚（VR = 3V ±1.2% @25°C, VCC=3V~5V）

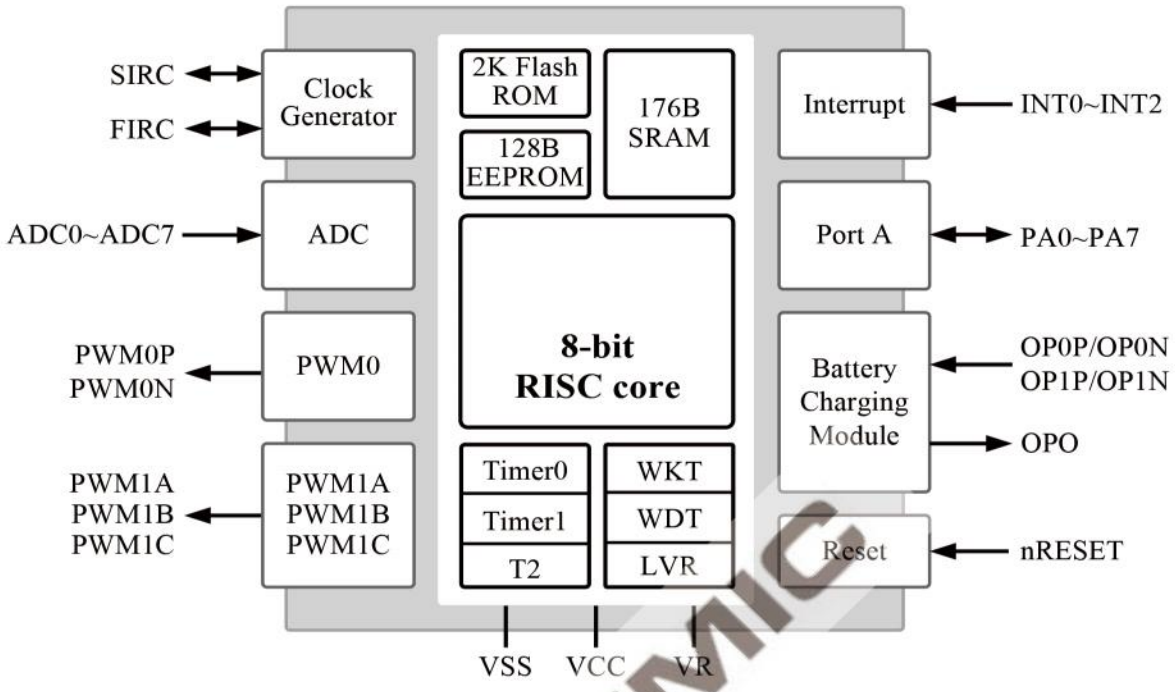
24. 编程连接支持 4 线（ICP）或 7 线（正常模式）

25. 封装类型：

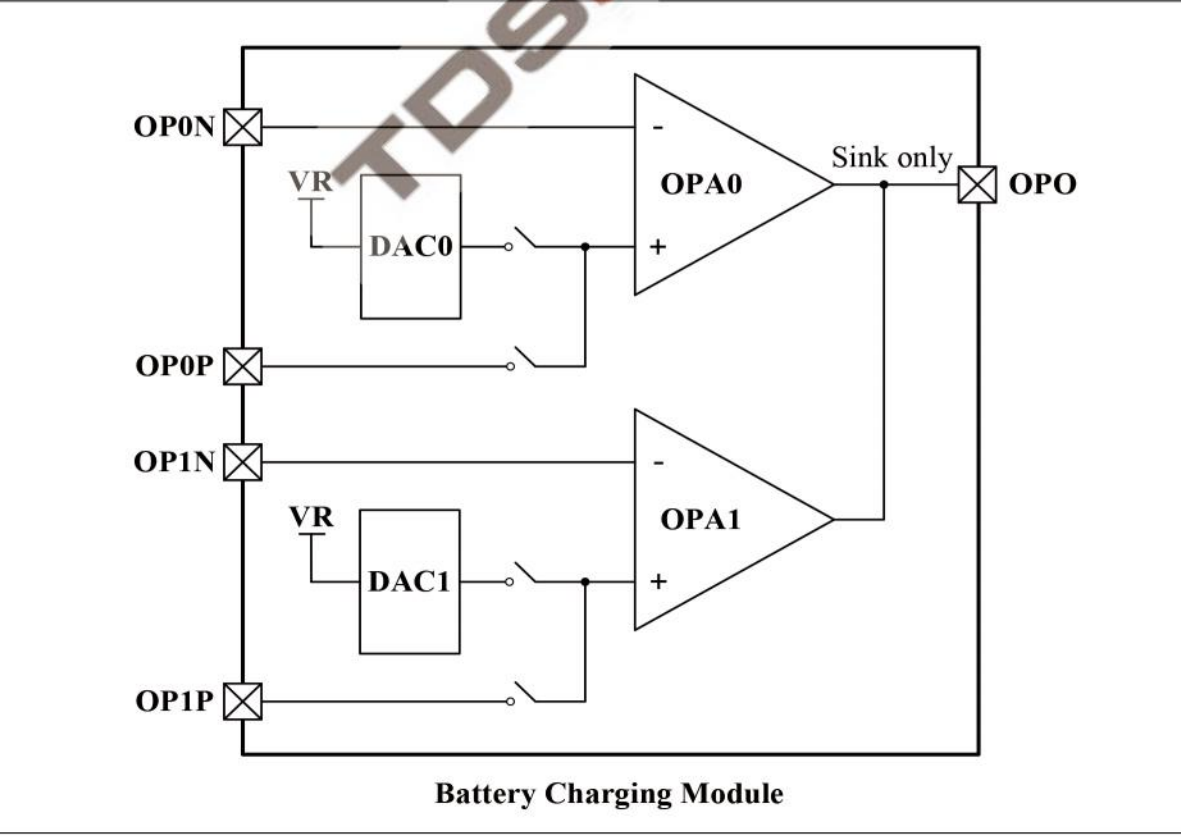
- SOP-16（150 mil）
- DIP-16（300 mil）

26. 在 ICE 上支持的 EV 板

系统框图

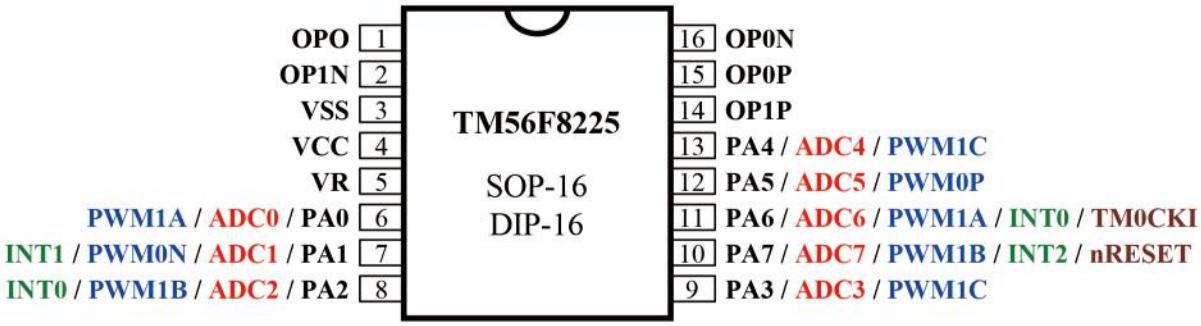


TM56F8225 Block Diagram



Battery Charging Module

引脚分配图



引脚描述

名称	输入/输出	引脚描述
PA0 - PA7	I/O	位可编程 I / O 端口，用于施密特触发器输入，CMOS 推挽输出或漏极开路输出。 上拉电阻可由软件分配。
nRESET	I	外部低电平有效复位
VCC, VSS	P	电源输入引脚和地
VR	O	内部参考电压 3V 输出（至少 1uF 接地）
INT0 - INT2	I	外部中断输入
TM0CKI	I	Timer0 在计数器模式下的输入
PWM0P	O	(8+2) bit PWM0 正输出
PWM0N	O	(8+2) bit PWM0 负输出
PWM1A	O	16 bit PWM1 输出
PWM1B	O	16 bit PWM1 输出
PWM1C	O	16 bit PWM1 输出
ADC0 - ADC7	I	ADC 通道输入
OP0P	I	OPA0 正输入
OP0N	I	OPA0 负输入
OP1P	I	OPA1 正输入
OP1N	I	OPA1 负输入
OP0	O	OPA0, OPA1 输出

编程引脚:

正常模式: VCC / VSS / PA0 / PA1 / PA2 / PA3 / PA4

ICP 模式: VCC / VSS / PA0 / PA1-使用 ICP（在线编程）模式时，PCB 需要除去 PA0，PA1 的所有组件。

引脚摘要

引脚序号	引脚名称	类型	GPIO			复位后的功能	替代功能			
			输入	输出						
16-SOP/DIP			外部中断	O.D	P.P		PWM	ADC	OPA	MISC
1	OP0	O							○	
2	OP1N	I							○	
3	VSS	P								
4	VCC	P								
5	VR	O								
6	PA0/ADC0/PWM1A	I/O		○	○	PA0	○	○		
7	PA1/ADC1/PWM0N/INT1	I/O	○	○	○	PA1	○	○		
8	PA2/ADC2/PWM1B/INT0	I/O	○	○	○	PA2	○	○		
9	OP0N	I							○	
10	OP0P	I							○	
11	OP1P	I							○	
12	PA4/ADC4/PWM1C	I/O		○	○	PA4	○	○		
13	PA5/ADC5/PWM0P	I/O		○	○	PA5	○	○		
14	PA6/ADC6/PWM1A/INT0/TM0CKI	I/O	○	○	○	PA6	○	○		TM0CKI
15	PA7/ADC7/PWM1B/INT2/n 复位	I/O	○	○	○	PA7	○	○		nRESET
16	PA3/ADC3/PWM1C	I/O		○	○	PA3	○	○		

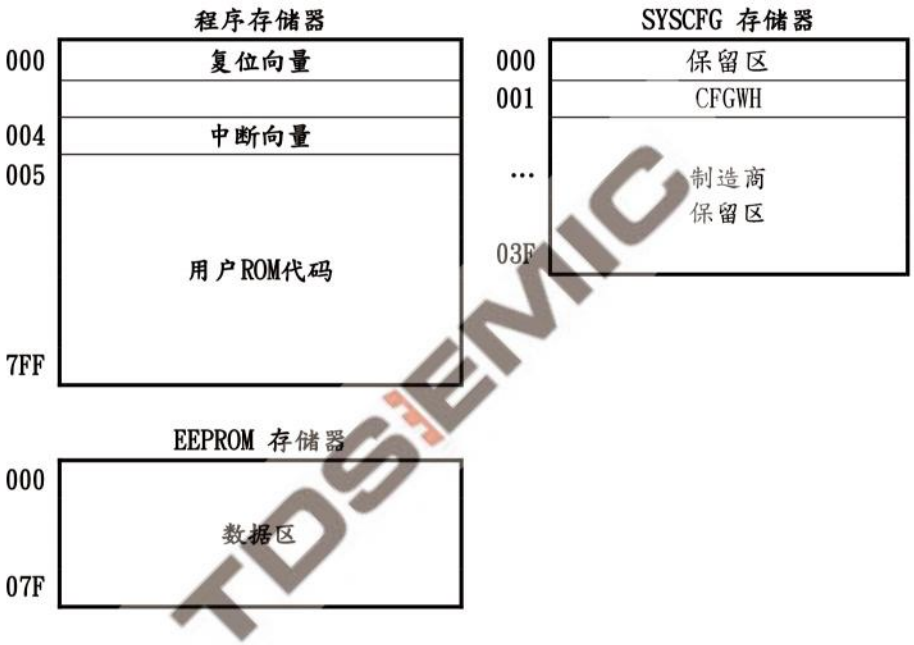
符号：P.P. = COMS 推挽输出
O.D. = 开漏输出

功能描述

1 CPU 核心

1.1 程序 ROM (PROM)

该器件的 falsh 程序 ROM 为 2K 字，具有一个额外的 64 字 INFO 区域来存储 SYSCFG 和一个额外的 128 字节 EEPROM。只要未设置 SYSCFG 的 PROTECT 位，就可以多次写入和读取 ROM。无论 PROTECT 位被置位还是清零，都可以读取 SYSCFG，但是只有当擦除用户 ROM 代码区域时，才能清除 PROTECT 位。也就是说，取消保护 PROTECT 位需要擦除相应的 ROM 区域。如果将 PROTECT 位置位，则写入器将不会读取用户 ROM 代码区域，并且直到 PROTECT 位被清零后，才能更新用户 ROM 代码。



1.1.1 复位向量 (000H)

复位后，系统将在地址 000h 处重新启动程序计数器（PC），所有寄存器都将恢复为默认值。

1.1.2 中断向量(004H)

发生中断时，程序计数器（PC）将被压入堆栈并跳转至地址 004H。

1.2 系统配置寄存器(SYSCFG)

系统配置寄存器（SYSCFG）位于 Flash INFO 区域。它包含一个 13 位寄存器（CFGWH）。 SYSCFG 确定 CPU 初始状态的选项。它仅由 PROM 编程者编写。用户可以通过 SYSCFG 寄存器选择 LVR 工作模式和芯片工作模式。CFGWH 的第 13 位是代码保护选择位。如果该位为 1，则当用户读取 PROM 时，PROM 中的数据将受到保护。

位		13~0	
默认值		00_0000_0000_0000	
位		描述	
CFGWH	13	**PROTECT**: 代码保护选择	
		1	使能
		0	禁止
	12	**XRSTE**:外部引脚（PA7）复位使能	
		1	使能
		0	禁止(PA7 作为 I/O 引脚)
	11-10	**LVR**:低电压复位模式	
		11	4.2V
		10	3.6V
		01	2.8V
		00	2.2V + LVD 功能(LVDS 00: 3.6V 01: 2.8V 1X: 4.2V)
	9-8	**WDTE**: WDT 复位使能	
		11	始终使能
		10	在 FAST / SLOW 模式下使能，在 IDLE / STOP 模式下禁止
		0X	禁止
	7-0	十速保留	

1.3 数据 ROM (EEPROM)

TM56F8225 包含 128 字节的数据 EEPROM 存储器。它组织为一个单独的数据空间，可以在其中读写单个字节。根据物理特性，EEPROM 比程序 ROM 需要更长的访问时间。EEPROM 具有至少 50K 写入/擦除周期的持久性。

EEPROM 读取 用法与读表取指令相似，只是必须将 EEPROM 使能位置高。通过向寄存器 EEPEN（18Eh）写入 0xE2 可以设置 EEPROM 使能位，向 EEPEN（18Eh）写入其他值将清除 EEPROM 使能位。

◇ 范例:读取 EEPROM 数据@地址 23h

```
        MOVLW   E2H             ;
        MOVWX   EEPEN           ;设置EEPROM使能位
        CLRX    DPH             ;设置DPH = 0用于EEPROM读/写
        MOVLW   00H
        MOVWX   DPH
        MOVLW   23H             ; 设置DPTR=0023h
        MOVWX   DPL

;使用操作码TABRL将EEPROM@地址23h数据读入W
        TABRL
        …

; 另一种方法使用TABR将EEPROM@地址 23h数据读入W
        MOVLW   01H
        MOVWX   TABR            ; TABR = 01h = 操作码TABRL
        …
```

EEPROM 写入 用法与读取 EEPROM 相似，但必须将 LVRPD 设置为 0x37 才能禁用 LVR。 当 F / W 将数据写入寄存器 EEPDT（18Fh）时，数据也将被写入 EEPROM。

◇ 范例:将 EEPROM 数据 A5h 写入地址 23h

```
        MOVLW   E2H             ;
        MOVWX   EEPEN           ; 设置EEPROM使能位
        CLRX    DPH             ; 设置DPH=0用于EEPROM写/读
        MOVLW   23H
        MOVWX   DPL             ; 设置DPTR=0023h
        MOVLW   00000011B
        MOVWX   EEPCTL          ; 将EEPROM写入设置为7.2mS超时
        MOVLW   37H             ; 设置W=LVRPD=37h,强制禁止LVR
        MOVWX   LVRPD           ;在EEP写操作之前必须禁用LVR
        MOVLW   A5H
        MOVWX   EEPDT           ; 写入数据A5h EEPDT（18Fh）
                                ; 数据也保存到EEPROM @地址23h
        BTXSC   EEPTO           ; 检测EEPROM写入超时标志位
        GOTO    TIMEOUT
        CLRX    EEPEN           ; 保护EEPROM免受异常写入
```

0Fh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CLKCTL	-	-	-	SLOWSTP	FASTSTP	CPUCKS	CPUPSC	
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
复位	-	-	-	0	1	0	1	1

0Fh.4 **SLOWSTP:** 在停止模式下停止慢时钟
0：慢速时钟运行
1：慢速时钟停止

0Fh.3 **FASTSTP:** 停止快速时钟
0：快速时钟运行
1：快速时钟停止

0Fh.2 **CPUCKS:** 系统时钟选择
0：慢速时钟作为系统时钟
1：快速时钟作为系统时钟

0Fh.1~0 **CPUPSC:** 系统时钟预分频器
0：除以 8　　1：除以 4　　2:除以 2　　3：除以 1

109h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LVRPD	LVRPD							
R/W	W	W	W	W	W	W	W	W
复位	0	0	0	0	0	0	0	0

109h.7~0 **LVRPD:** LVR 掉电寄存器
写 37h 强制 LVR 禁止。 （必须在 EEPROM 写操作之前禁止 LVR）

18Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TABR	TABR							
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

18Ch.7~0 1. TABR 写 01h =操作码 TABRL
2. TABR 写 02h =操作码 TABRH
3.在步骤 1 或步骤 2 之后，读取 TABR 以获取 Main ROM 读表取值
在步骤 1 之后，读取 TABR 以获取 EEPROM 值（当 EEPEN = E2h 时）
ASM 的读表：TABRL / TABRH 或 TABR
C 的读表：TABR

18Dh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EEPCTL	-	-	-	-	-	EEPTO	EEPTE	
R/W	-	-	-	-	-	R	R/W	R/W
复位	-	-	-	-	-	0	0	0

18Dh.2 **EEPTO:** EEPROM 写入超时标志
当 EEPROM 写入发生超时时由 H/W 置位
当 EEPTE=0 时由 H/W 清零

18Dh.1~0 **EEPTE:** EEPROM 写入看门狗定时使能 **（建议设置 11）**
00：禁止　　01：1.8ms　　10：7.3ms　　11：14.6ms

18Eh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EEPEN	EEPEN							
R/W	W	W	W	W	W	W	W	W
复位	0	0	0	0	0	0	0	0

18Eh.7~0 **EEPEN**: EEPROM 访问使能
向该寄存器写入 0xE2 将使能 EEPROM 访问
向该寄存器写入其他值将禁止 EEPROM 访问

18Fh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EEPDT	EEPDT							
R/W	W	W	W	W	W	W	W	W
复位	0	0	0	0	0	0	0	0

18Fh.7~0 **EEPDT**: EEPROM 数据写入
使能 EEPROM 访问后，将数据写入该寄存器将使 H / W 将数据写入 EEPROM

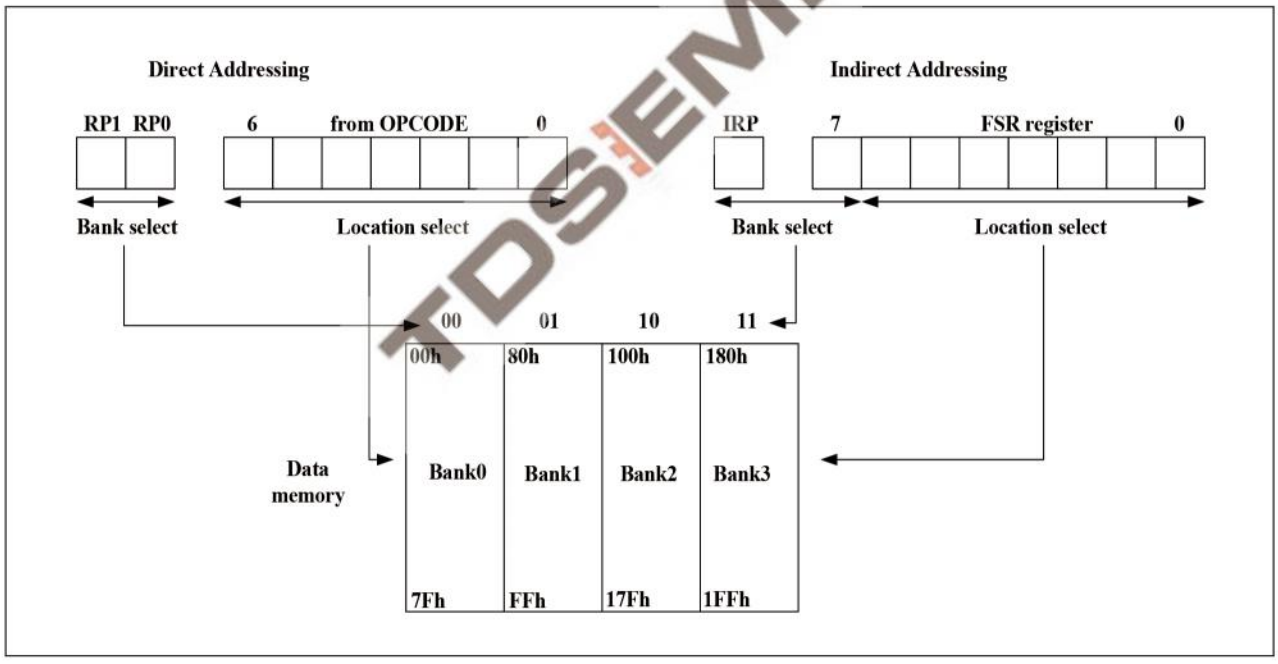
1.4 RAM 寻址模式

CPU 中有一个数据存储器平面。 分为四个部分 BANK。 每个 BANK 最多可扩展到 7Fh（128 字节）。 每个 BANK 的下部位置保留用于特殊功能寄存器（SFR）。 SFR 上方是通用寄存器，实现为静态 RAM。 所有已实现的 BANK 都包含特殊功能寄存器。 来自一个 BANK 的一些常用特殊功能寄存器可能会在另一个 BANK 中进行镜像，以减少代码并加快访问速度。

RP1 和 RP0 位（STATUS [6：5]）是 BANK 选择位

[RP1, RP0]	BANK
00	0
01	1
10	2
11	3

该页面可以直接或间接寻址。 对 INDF 寄存器进行寻址将导致间接寻址， INDF 寄存器不是物理寄存器， 任何使用 INDF 寄存器的指令实际上都访问文件选择寄存器 FSR 指向的寄存器。 间接读取 INDF 寄存器本身（FSR = “ 0”）将读为 00h。 间接写入 INDF 寄存器将导致空操作（可能会影响状态位）。 通过将 8 位 FSR 寄存器和 IRP 位（STATUS [7]）连接起来，可以获得有效的 9 位地址。 请参考下图。



Direct / Indirect Addressing

在 F / W 代码的开头使用新的指令集，保持 RP0 = RP1 = 0。 使用新指令的好处是用户可以忽略寄存器的 bank 区位置，并且可以节省代码大小。 新指令与旧指令几乎相同。 通过将指令集中的 “ F” 替换为 “ X” ，可以轻松使用新指令，而无需切换 BANK。

例如：

BCF	TM0IE	→	BCX	TM0IE
DECF	CNT, 1	→	DECX	CNT, 1
INCFSZ	RAM25, 0	→	INCXSZ	RAM25, 0
MOVWF	PAMODL	→	MOVWX	PAMODL
RLF	RAMA0, 0	→	RLX	RAMA0, 0
SWAPF	ADCTL, 0	→	SWAPX	ADCTL, 0

【BANK0】 00~7Fh		【BANK1】 80h~FFh		【BANK2】 100h~17Fh		【BANK3】 180h~1FFh	
00h	**INDF**	80h	**INDF**	100h	**INDF**	180h	**INDF**
01h	TM0	81h	OPTION	101h	TM0	181h	OPTION
02h	**PCL**	82h	**PCL**	102h	**PCL**	182h	**PCL**
03h	**STATUS**	83h	**STATUS**	103h	**STATUS**	183h	**STATUS**
04h	**FSR**	84h	**FSR**	104h	**FSR**	184h	**FSR**
05h	PAD	85h		105h		185h	DPL
06h		86h		106h		186h	DPH
07h		87h		107h		187h	CRCDL
08h		88h		108h		188h	CRCDH
09h		89h		109h	LVRPD	189h	CRCIN
0Ah	**PCLATH**	8Ah	**PCLATH**	10Ah	PCLATH	18Ah	**PCLATH**
0Bh	**INTIE**	8Bh	**INTIE**	10Bh	INTIE	18Bh	**INTIE**
0Ch	INTIF	8Ch	PAMODH	10Ch		18Ch	TABR
0Dh	INTIE1	8Dh	PAMODL	10Dh		18Dh	EEPCTL
0Eh	INTIF1	8Eh		10Eh	BGTRIM	18Eh	EEPEN
0Fh	CLKCTL	8Fh		10Fh	IRCF	18Fh	EEPDT
10h	TM0RLD	90h		110h		190h	
11h	TM0CTL	91h	PWM0E	111h		191h	
12h	TM1	92h	PWM0PRD	112h		192h	
13h	TM1RLD	93h	PWM0DH	113h		193h	
14h	TM1CTL	94h	PWM0DL	114h		194h	
15h	T2CTL	95h	PWM0CTL	115h		195h	
16h	MF016	96h	PWM0CTL1	116h		196h	
17h	ADCH	97h	PWM1CTL	117h		197h	
18h	ADCTL	98h	PWM1PRDH	118h		198h	
19h	MF019	99h	PWM1PRDL	119h		199h	
1Ah	OPA0CTL	9Ah	PWM1ADH	11Ah		19Ah	
1Bh	OPA1CTL	9Bh	PWM1ADL	11Bh		19Bh	
1Ch	DAC0DH	9Ch	PWM1BDH	11Ch		19Ch	
1Dh	DAC0DL	9Dh	PWM1BDL	11Dh		19Dh	
1Eh	DAC1DH	9Eh	PWM1CDH	11Eh		19Eh	
1Fh	DAC1DL	9Fh	PWM1CDL	11Fh		19Fh	
20h	RAM Bank0 area (80 Bytes)	A0h	RAM Bank1 area (80 Bytes)	120h		1A0h	

6Fh		EFh		16Fh		1EFh	
70h	公共区域	F0h	访问	170h	访问	1F0h	访问
~	16 字节	~	70h~7Fh	~	70h~7Fh	~	70h~7Fh
7Fh		FFh		17Fh		1FFh	

◇范例:通过直接寻址来读取/写入寄存器(强制 RP0 = RP1 = 0)

```
TM1         equ     12H         ; SFR在Bank0
PWM0PRD     equ     92H         ; SFR在Bank1
IRCF        equ     10FH        ; SFR在Bank2
DPL         equ     185H        ; SFR在Bank3
RAM20       equ     20H         ; RAM在Bank0
RAMA0       equ     A0H         ; RAM在Bank1

MOVXW       TM1                 ; 将 TM1 (Bank0) 读到 W
MOVXW       PWM0PRD             ; 将 PWM0PRD (Bank1) 读到 W
MOVXW       IRCF                ; 将 IRCF (Bank2) 读到 W
MOVXW       DPL                 ; 将 DPL (Bank3) 读到 W

MOVLW       16H
MOVWX       RAM20               ; W = 16h 写入 RAM[0x20]
MOVWX       RAMA0               ; W = 16h 写入 RAM[0xA0]

MOVLW       37H
MOVWX       LVRPD               ; LVRPD = W = 37h, 强制LVR禁止

MOVXW       CLKCTL              ;将SFR CLKCTL (0Fh) 读取到W
MOVXW       IRCF                ; 将SFR IRCF (10Fh) 读到W

MOVLW       0BH
MOVWX       CLKCTL              ; CLKCTL (0Fh) = W = 0Bh
MOVWX       IRCF                ; IRCF (10Fh) = W = 0Bh
```

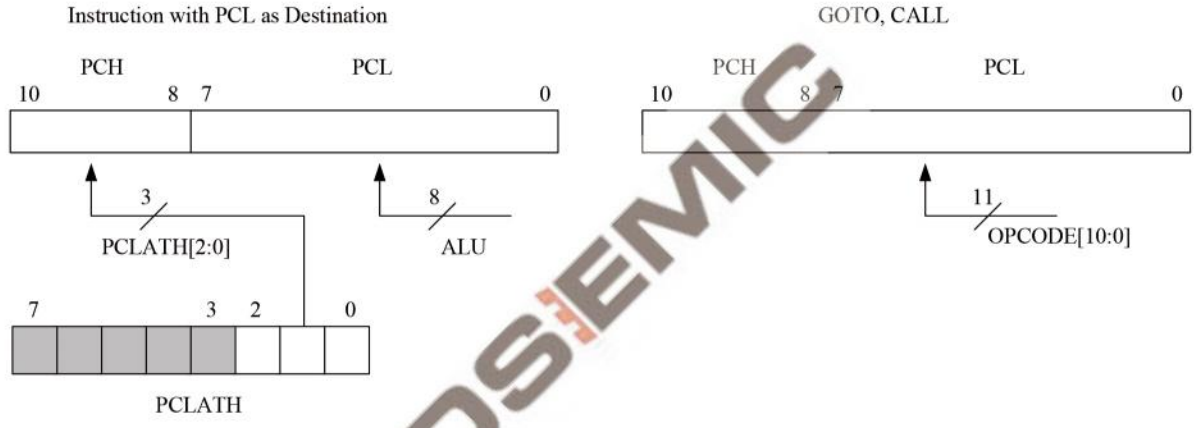
◇范例:通过间接寻址来读取/写入寄存器(强制 RP0 = RP1 = 0)

```
BSX         IRP                 ; IRP = 1 => Bank2/3
MOVLW       0FH                 ; W = 0Fh
MOVWX       FSR                 ; FSR = W = 0Fh
MOVXW       INDF                ; 将SFR IRCF (10Fh) 读到 W

BSX         IRP                 ; IRP = 1 => Bank2/3
MOVLW       0FH                 ; W = 0Fh
MOVWX       FSR                 ; FSR = W =0Fh
MOVLW       0BH                 ; W = 0Bh
MOVWX       INDF                ; IRCF (10Fh) = W = 0Bh

BCX         IRP                 ; IRP=0 =>Bank0/1
MOVLW       0FH                 ; W = 0Fh
MOVWX       FSR                 ; FSR = W = 0Fh
MOVXW       INDF                ; read SFR CLKCTL (0Fh) to W

BCX         IRP                 ; IRP = 0 => Bank0/1
MOVLW       0FH                 ; W = 0Fh
```

```
MOVWX     FSR              ; FSR = W =0Fh
MOVLW     0BH              ; W = 0Bh
MOVWX     INDF             ; CLKCTL (0Fh) = W = 0Bh
```

1.5 程序计数器 (PC)和堆栈

程序计数器为 11 位宽，能够寻址 2K x 16 Flash ROM。低字节 PCL 寄存器是可读写的寄存器。高字节（PC [10：8]）不可读，但可通过 PCLATH 寄存器间接写入。在任何复位状态下，PC 的高字节将被清除。当执行一条程序指令时，PC 将包含下一条要执行的程序指令的地址。除以下情况外，PC 值通常增加一。复位向量（000h）和中断向量（004h）用于 PC 初始化和中断。对于 CALL / GOGO 指令，PC 从指令字中加载 11 位地址。对于 RET / RETI / RETLW 指令，PC 从顶层堆栈中检索其内容。同时,执行以 PCL 寄存器为目的地的任何指令都会导致程序计数器 PC [10：8]位（PCH）被 PCLATH 寄存器的内容替换。这样可以通过将所需的高 3 位写入 PCLATH 寄存器来更改程序计数器的全部内容。当低 8 位写入 PCL 寄存器时，程序计数器的所有 11 位将更改为 PCLATH 寄存器中包含的值以及写入 PCL 寄存器中的值。



STACK 为 11 位宽度，8 级深度。 CALL 指令和硬件中断将按顺序推入 STACK。 当执行 RET / RETI / RETLW 指令时按顺序弹出 STACK。 对于表查，该器件提供了功能强大的表读取指令 TABRL，TABRH，通过设置 DPTR = {DPH，DPL}寄存器将 16 位 ROM 数据返回到 W 寄存器。 通过将 C 语言设置为 TABR（18Ch），它还提供了另一种将 16 位 ROM 数据读入 W 寄存器的方法。

◇ 范例：要查找位于“ TABLE1”和“ TABLE2”的 PROM 数据。

```
ORG       000h                    ;  复位向量
          GOTO      START

START:
          MOVLW     00H
          MOVWX     INDEX           ; 设置查表地址
LOOP:
          MOVLW     (TABLE1>>8) & 0xff
          MOVWX     PCLATH          ;以PCL为目标的指令
          MOVXW     INDEX           ; 将索引值移至W寄存器
          CALL      TABLE1          ; 要查找数据，W = 55h
```

```
            …
            INCX        INDEX, 1            ;递增下一个地址的索引地址
            …
            GOTO        LOOP                ; 跳转到LOOP标签
            …
            MOVLW       (TABLE2 >>8) & 0xff
            MOVWX       DPH                 ; DPH寄存器 (F186.2~0)
            MOVLW       (TABLE2) & 0xff
            MOVWX       DPL                 ; DPL寄存器 (F185.7~0)

; 通过操作码TABRL / TABRH读表
            TABRL                           ; 将PROM低字节数据读到 W (W=86h)
            TABRH                           ; 将PROM高字节数据读到W (W=19h)
            …

; 通过特殊功能寄存器TABR查表
            MOVLW       01H                 ; TABR = 01H = 操作码 TABRL
            MOVWX       TABR                ; 将 PROM低字节数据读到W (W=86h)
            MOVLW       02H                 ; TABR = 02H =操作码TABRH
            MOVWX       TABR                ; 将PROM高字节数据读到W (W=19h)
            …

TABLE1:
            ADDWX       PCL, 1              ; 将W与PCL相加，结果返回PCL。
            RETLW       55H                 ; 返回时W=55h
            RETLW       56H                 ; 返回时W=56h
            RETLW       57H                 ; 返回时W=57h
            RETLW       58H                 ; 返回时W=58h
            …

            ORG         368h
TABLE2:
            .DT         0x1986              ; 16位ROM数据
            .DT         0x3719
            .DT         0x2983
            …
```

18Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TABR	TABR							
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

18Ch.7~0 1. TABR 写 01h =操作码 TABRL
2. TABR 写入 02h =操作码 TABRH
3.在步骤 1 或步骤 2 之后，读取 TABR 以获取主 ROM 读表取值
在步骤 1.之后，读取 TABR 以获取 EEPROM 值（当 EEPEN = E2h 时）
ASM 的读表：TABRL / TABRH 或 TABR

C 的读表：TABR

1.5.1ALU 和工作寄存器（W）

ALU 为 8 位宽，能够进行加，减，移位和逻辑运算。 在双操作数指令中，通常一个操作数是 W 寄存器，它是用于 ALU 操作的 8 位不可寻址寄存器。 另一个操作数是文件寄存器或立即数。 在单操作数指令中，操作数可以是 W 寄存器或文件寄存器。 根据执行的指令，ALU 可能会影响 STATUS 寄存器中的进位（C），十进制进位（DC）和零（Z）标志的值。 C 和 DC 标志在减法中分别用作/借位和/ 十进制借位。

注： /借位表示借位寄存器的取反。
/十进制借位代表十进制借位寄存器的取反

1.5.2STATUS 寄存器(03H/83H/103H/183H)

该寄存器包含 ALU 的算术状态和复位状态。 与任何其他寄存器一样，STATUS 寄存器可以是任何指令的目的地。如果 STATUS 寄存器是影响 Z，DC 或 C 位的指令的目标，则禁止对这三个位的写操作。 这些位根据器件逻辑置位或清除。 因此，建议仅使用 BCX，BSX 和 MOVWX 指令来更改状态寄存器，因为这些指令不会影响这些位。.

STATUS	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
复位值	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W
位	描述							
7	IRP: 寄存器位于 Bank 的 选择位（用于间接寻址） 0 = Bank 0,1 (00h - FFh) 1 = Bank 2,3 (100h - 1FFh)							
6:5	RP1:RP0: 寄存器位于 Bank 的选择位（用于直接寻址） 00 = Bank 0 (00h - 7Fh) 01 = Bank 1 (80h - FFh) 10 = Bank 2 (100h - 17Fh) 11 = Bank 3 (180h - 1FFh) 每个 bank 为 128 字节							
4	TO: 超时标志 0：上电复位，LVR 复位或 CLRWDT / SLEEP 指令后 1： WDT 超时							
3	PD: 掉电标志 0：上电复位，LVR 复位或 CLRWDT 指令后 1：执行 SLEEP 指令后							
2	Z:：零标志 0：逻辑运算的结果不为零 1：逻辑运算的结果为零							
1	DC：十进制进位标志或/借位标志							
	ADD 指令				SUB 指令			
	0：没进位 1：低字节有进位				0：低字节有借位 1： 没借位			
0	C：进位标志或/借位标志							
	ADD 指令				SUB 指令			
	0：没进位 1：MSB 有进位				0： MSB 有借位 1: 没借位			

◇ 范例：将立即数据写入状态寄存器 STATUS 寄存器.

```
MOVLW    00H
MOVWX    STATUS          ; 清除STATUS寄存器
```

◇ 范例：位寻址置位和清除 STATUS 寄存器.

```
BSX      STATUS, 0       ; 设置C=1
BCX      STATUS, 0       ; 清除C=0
```

◇ 范例： 通过 BTXSS 指令确定 C 标志.

```
BTXSS    STATUS, 0       ; 检查进位标志
GOTO     LABEL_1         ; 如果C=0，跳转到label_1
GOTO     LABEL_2         ; 如果C=1，跳转到label_2
```

2 复位

该器件有四种复位方式.

- 上电复位(POR)
- 低电压复位(LVR)
- 外部引脚复位(PA7)
- 看门狗复位(WDT)

复位可以是由上电复位（POR），外部引脚复位（XRST），看门狗定时器复位（WDTR）或低电压复位（LVR）引起的。CFGWH 控制复位功能。复位后，SFR 返回其默认值，程序计数器（PC）被清除，并且系统从复位向量 000H 处开始运行。状态寄存器（STATUS）上的 TO 和 PD 标志指示系统复位状态。

2.1 上电复位

上电复位后，所有系统和外围控制寄存器都将恢复为其默认硬件复位值。 时钟源，LVR 电平和芯片工作模式由 CFGWH 寄存器值选择。

2.2 低电压复位

当电源电压低于阈值电压时，低压复位具有静态复位功能。可以选择四个阈值级别。LVR 的工作模式由 CFGWH 寄存器定义。参见以下 LVR 选择表；用户还必须考虑工作频率的最低工作电压。
LVR 选择表：

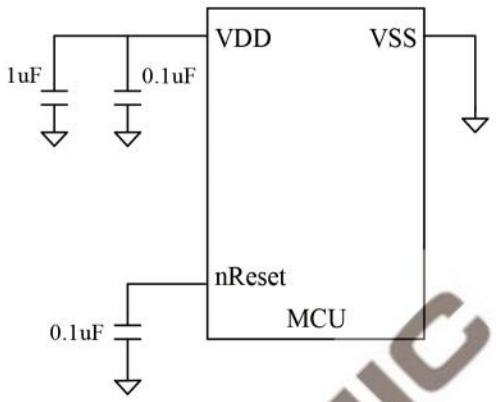
LVR 阈值	工作电压
LVR2.2	5.5V > VCC > 2.2V
LVR2.8	5.5V > VCC > 2.8V
LVR3.6	5.5V > VCC > 3.6V
LVR4.2	5.5V > VCC > 4.2V or Vcc=5.0V

不同的 Fsys 具有不同的系统最小工作电压，参考 DC 特性的工作电压，如果当前系统电压低于最小工作电压，选择较低的 LVR，则系统可能进入死区并发生错误。

2.3 外部引脚复位

外部引脚复位可以通过 CFGWH 寄存器禁止或使能。 它至少需要保持 2 个 SIRC 时钟周期才能被芯片看到。 XRST 还将所有控制寄存器恢复为其默认复位值。 TO / PD 标志不受这些复位的影响。.

外部复位引脚为低电平有效。 复位引脚为高电平时，系统正在运行。 复位引脚接收低电压，系统复位。 外部复位可以在上电期间使系统复位，良好的外部复位电路可以保护系统以避免在异常电源条件下工作。



2.4 看门狗定时器复位

WDT 溢出复位可通过 CFGWH 寄存器禁止或使能。它在快速/慢速模式运行，在空闲/停止模式运行或停止。 WDT 溢出速度可以由 WDTPSC SFR 定义。 WDT 通过器件复位或 CLRWDT SFR 位清除，WDT 溢出复位也将所有控制寄存器恢复为其默认复位值。 TO / PD 标志不受这些复位的影响。

◇ 范例: 定义复位向量

```
        ORG     000H            ; 复位向量
        GOTO    START           ; 跳转到用户程序地址

        ORG     010H            ; 010H, 用户程序开头
START:
        …
        …
        GOTO    START
```

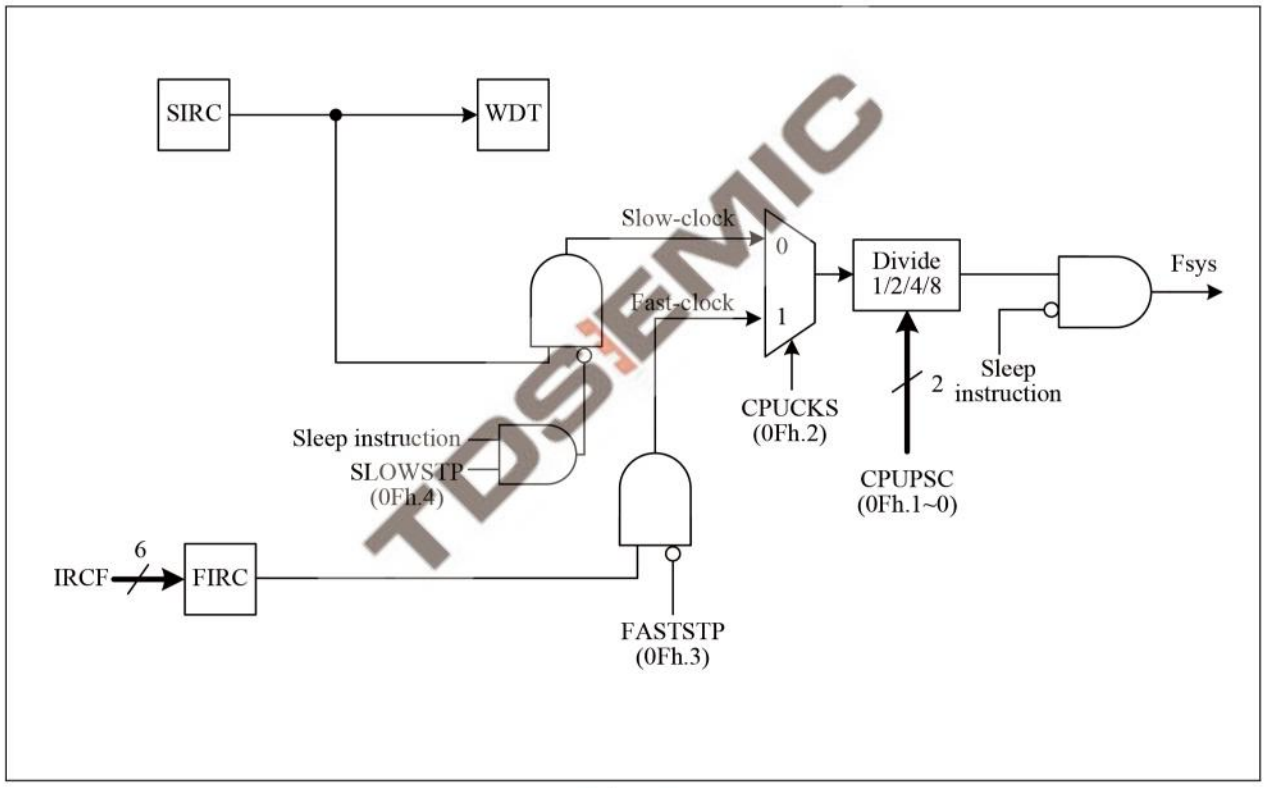
3 时钟电路和工作模式

3.1 系统时钟

该器件采用双系统时钟设计。有两种时钟源，即 SIRC（内部慢速 RC）和 FIRC（内部快速 RC）。每个时钟源都可以作为系统时钟应用于 CPU 内核。 在空闲模式下，只有慢速时钟保持振荡以向 T2 模块提供时钟源。 请参考下图。

复位后，器件以 70 KHz SIRC 在慢速模式下运行。S/W 应该选择适当的时钟频率以确保芯片工作稳定。更高的 VCC 允许芯片以更高的系统时钟频率运行。在典型情况下，8 MHz 系统时钟频率要求 VCC> 2.1V。

CLKCTL（0Fh）SFR 控制系统时钟的操作。H/W 自动阻止该寄存器的 S/W 异常设置。S/W 只能在快速模式下更改慢时钟类型，在慢模式下更改快速时钟类型。永远不要同时写入 FASTSTP = 1 和 CPUCKS = 1。建议逐位写入该 SFR。



时钟电路框图

FIRC（内部快速 RC）的频率可以通过 IRCF（10Fh）进行调整。 当 IRCF = 00h 时，频率最低。当 IRCF = 7Fh 时，频率最高。 因为每个 IC 可能具有不同的 IRCF 默认值，所以使用此功能，我们可以在上电后调整 FIRC 的频率，以确保上电复位后 FIRC = 8MHz 的频率。

快速模式：

在这种模式下，程序使用快速时钟作为 CPU 时钟（Fsys）执行。Timer0，Timer1 模块也由快速时钟驱动。PWM0 / PWM1 模块可以由 FIRC 8M，FIRC 16M 或 Fsys 驱动。通过设置 T2CKS（15h.2），可以由慢时钟或 Fsys/128 驱动 T2。

慢速模式：

上电或复位后，器件进入慢速模式，慢速时钟默认为 SIRC。在此模式下，始终使能慢速时钟，但快时钟可以停止（通过 FASTSTP = 1，以节省功耗）或运行（通过 FASTSTP = 0）。在慢速模式下，所有外围模块（Timer0，Timer1 等）的时钟源均为慢时钟。

空闲模式：

如果在执行 SLEEP 指令之前使能了慢速时钟（SLOWSTP = 0）且 T2CKS = 0，则 CPU 进入空闲模式。在此模式下，慢速时钟源使 T2 模块保持运行状态。CPU 停止获取代码，除 T2 相关电路外，所有模块均停止。空闲模式通过复位或使能的中断唤醒而终止。

在空闲模式下，保持慢速时钟振荡的另一种方法是在执行 SLEEP 指令之前将 WKTIE = 1（0Bh.3）以保持 WKT 运行，或者将 WDTE = 11（CFGWH.9～8）保持 WDT 运行。在这种情况下，无论是否设置或清除了 SLOWSTP，慢速时钟都将继续工作并定期唤醒 CPU。

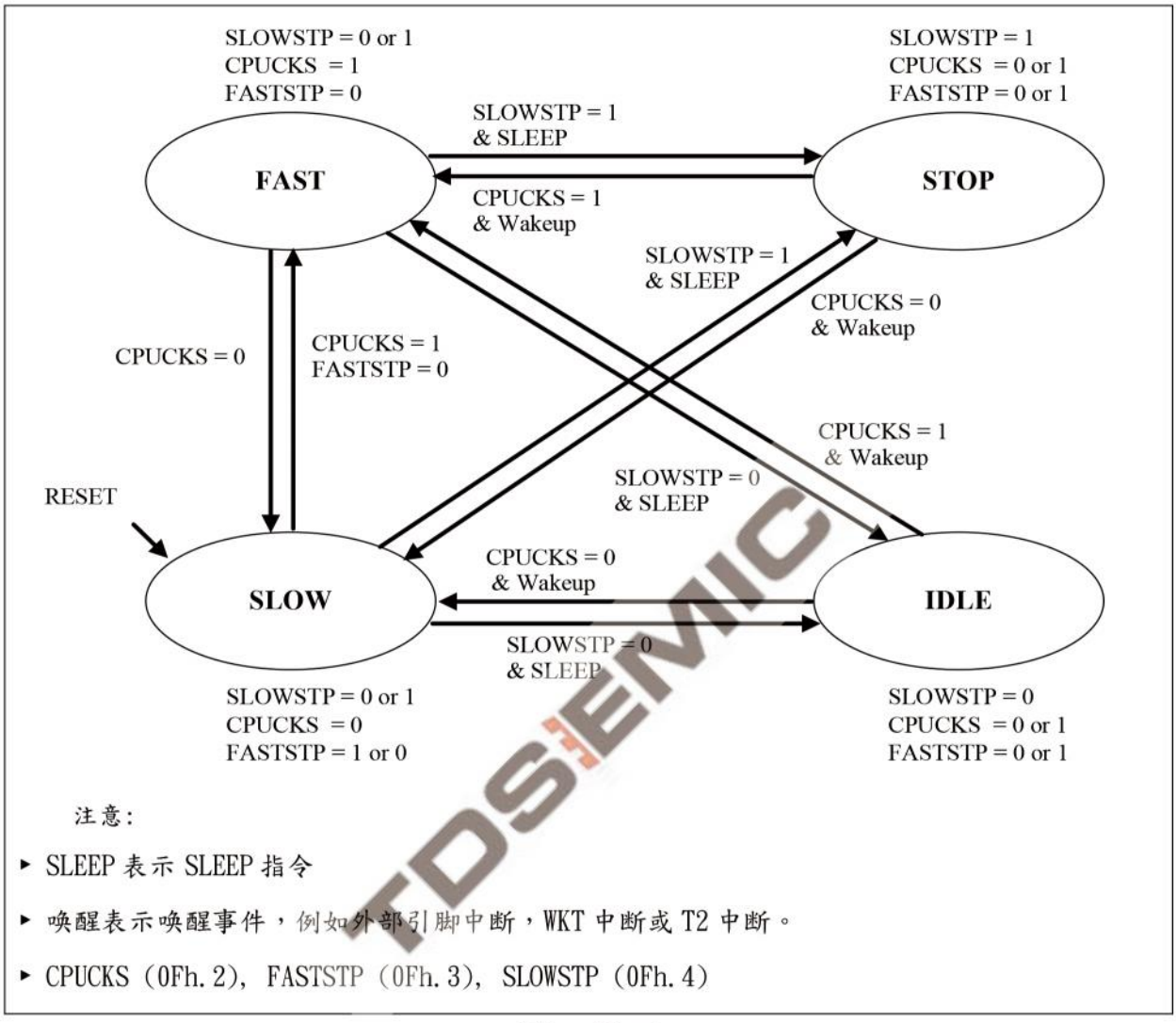
T2 和 WKT / WDT 是独立的，并且具有自己的控制寄存器。可以将 T2 和 WKT 都保持工作并在 IDLE 模式下唤醒。

停止模式：

在执行 SLEEP 指令之前将 SLOWSTP（0Fh.4）置位，WKTIE（0Bh.3）清零和 WDTE = 10 或 0X，则芯片将进入停止模式，所有模块都将停止。停止模式类似于空闲模式。 不同之处在于，所有时钟振荡器（快时钟或慢时钟）均已停止，并且不产生任何时钟。

3.2 双系统时钟模式转换

该器件运行在以下四种模式之一：快速模式，慢速模式，空闲模式和停止模式。



注意：

- SLEEP 表示 SLEEP 指令
- 唤醒表示唤醒事件，例如外部引脚中断，WKT 中断或 T2 中断。
- CPUCKS (0Fh.2), FASTSTP (0Fh.3), SLOWSTP (0Fh.4)

CPU 工作框图

CPU 模式和时钟功能表：

模式	振荡器	Fsys	快时钟	慢时钟	TM0/TM1	T2	唤醒事件
FAST	FIRC	快速时钟	运行	通过设置 SLOWSTP	运行	运行	X
SLOW	SIRC	慢速时钟	Set by FASTSTP	运行	运行	运行	X
IDLE	SIRC	停止	停止	Run	停止	运行	WKT/IO/T2
STOP	停止	停止	停止	停止	停止	停止	IO

- **快速模式切换至慢速模式**

当快速模式切换至慢速模式时，建议按顺序执行以下步骤：

(1)使能慢速模式（SLOWSTP=0）

(2)切换至慢速模式(CPUCKS=0)

(3)停止快速模式（FASTSTP=1）

◇ 范例：快速模式切换至慢速模式.

```
BCX     SLOWSTP         ; 使能慢速时钟
NOP
BCX     CPUCKS          ; Fsys=慢速时钟
BSX     FASTSTP         ; 停止快速时钟
```

- **慢速模式切换至快速模式**

可以通过 CLKCTL 寄存器中的 CPUCKS = 0 来使能慢速模式。当慢速模式切换到快速模式时，建议按顺序执行以下步骤：

(1)使能快速时钟（FASTSTP=0）

(2)切换至快速时钟（CPUCKS=1）

◇ 范例:慢速模式切换至快速模式

```
BCX     FASTSTP         ; 使能快速时钟
NOP
BSX     CPUCKS          ; Fsys=快速时钟
```

- **空闲模式设置**

可以通过以下顺序设置空闲模式:

(1)使能慢速时钟（SLOWSTP=0)或 WKT(WKTIE=1)

(2)将 T2 时钟源切换为慢时钟（T2CKS=0）

(3)执行 SLEEP 指令

空闲模式可以通过外部中断，WKT 中断和 T2 中断来唤醒.

◇ 范例：快速/慢速模式切换至空闲模式.

```
BCX     SLOWSTP         ; 使能慢速时钟
MOVLW   00000000B
MOVWX   T2CTL
SLEEP                   ; 进入空闲模式
```

- **停止模式设置**

可以通过以下顺序设置停止模式:

(1)停止慢速时钟(SLOWSTP=1)

(2)停止 WKT/WDT (WKTIE=0, WDTE=10 或 0X)

(3)执行 SLEEP 指令

停止模式只能通过外部引脚中断来唤醒.

◇ 范例:快速/慢速模式切换至停止模式.

```
BSX     SLOWSTP          ; 停止慢速时钟.
MOVLW   00000000B        ; 禁止 WKT计数
MOVWX   INTIE
SLEEP                    ; 进入停止模式.
```

0Fh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CLKCTL	-	-	-	SLOWSTP	FASTSTP	CPUCKS	CPUPSC	
R/W	-	-	-	R/W	R/W	R/W	R/W	
复位	-	-	-	0	0	0	1	1

0Fh.4 **SLOWSTP**: 停止慢速时钟
0:慢速时钟运行
1:慢速时钟在断电模式下停止运行

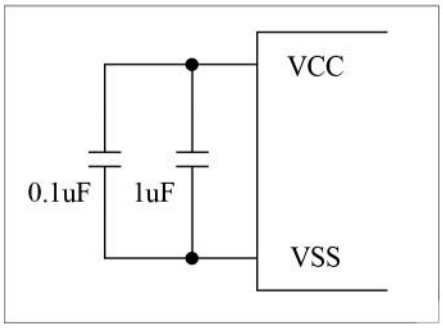
0Fh.3 **FASTSTP**: 停止快速时钟
0:快速时钟运行
1:快速时钟停止

0Fh.2 **CPUCKS**:系统时钟源选择
0:慢速时钟
1:快速时钟

0Fh.1~0 **CPUPSC**: 系统时钟源预分频器，系统时钟源除以
00:除以8
01:除以4
10:除以2
11:除以1

3.3 系统时钟振荡器

在内部快速 RC（FIRC）模式下，片上振荡器产生 8MHz 的系统时钟。由于电源噪声会降低内部时钟振荡器的性能，因此将电源旁路电容器 1 uF 和 0.1 uF 放置在非常靠近 VCC/VSS 引脚，可以提高时钟和整个系统的稳定性。



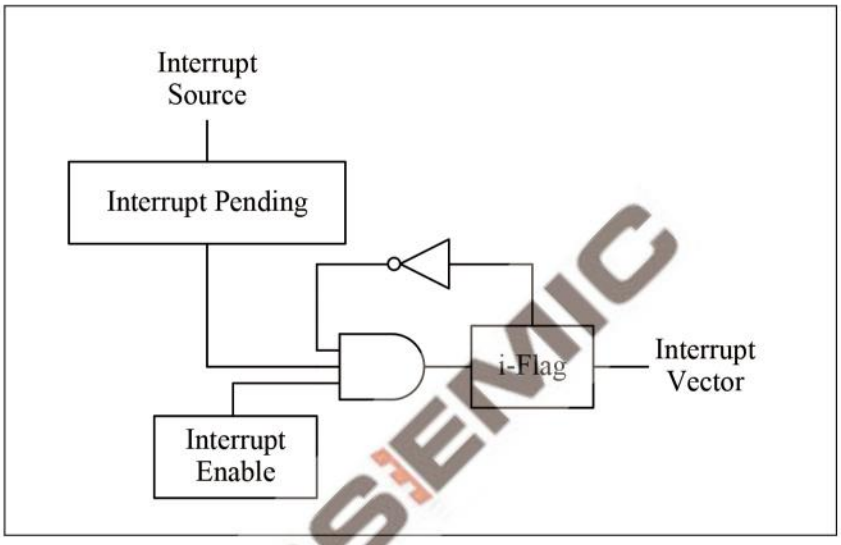
Internal RC Mode

4 中断

TM56F8225 有 1 级、1 个向量和 12 个中断源。每个中断源都有自己的使能控制位。无论其使能控制位是 0 还是 1，中断事件都将设置其各自的挂起标志，。

如果相应的中断使能位（INTIE[7:0]，INTIE1[3:0]）已置位，它将触发 CPU 来服务该中断。CPU 在当前执行的指令周期结束时接受中断。同时，向 CPU 插入“CALL 004”指令，并设置 i 标志以防止递归中断嵌套。

i 标志在执行“RETI”指令之后被清除。也就是说，在中断服务未完成之前，主程序中至少有一条指令被执行。中断事件是电平触发的。F/W 在服务中断程序时必须清除中断事件寄存器。



◇ 范例：使用上升沿触发来设置 INT1（PA1）中断请求 r

```
        ORG     000H            ; 复位向量
        GOTO    START           ; 跳转到用户程序地址

        ORG     004H            ; 所有中断的向量
        GOTO    INT             ; 如果INT1（PA1）输入出现上升沿

        ORG     005H
START:
        MOVLW   xxxx00xxB
        MOVWX   PAMODL          ; 选择INT引脚模式作为模式0
                                ; 开漏输出低电平或上拉输入
        MOVLW   xxxxxx1xB
        MOVWX   PAD             ; 释放INT1，变为施密特触发器
                                ; 输入带上拉电阻
        MOVLW   001xxxxxB
        MOVWX   OPTION          ; 将INT1中断触发设置为上升沿
        MOVLW   11111101B
        MOVWX   INTIF           ; 清除INT1中断请求标志
        MOVLW   00000010B
        MOVWX   INTIE           ; 使能INT1中断
MAIN:
        …
        GOTO    MAIN

INT:
        MOVWX   20H             ; 将W数据存储到FRAM 20H
        MOVXW   STATUS          ; 获取STATUS数据
        MOVWX   21H             ; 将STATUS数据存储到FRAM 21H

        BTXSS   INT1IF          ; 检测INT1IF位
        GOTO    EXIT_INT        ; INT1IF = 0,退出中断子程序
        …                       ; INT1中断服务程序
        MOVLW   111111 01B
        MOVWX   INTIF           ; 清除INT1中断请求标志
EXIT_INT:
        MOVXW   21H             ; 获取FRAM 21H数据
        MOVWX   STATUS          ; 恢复 STATUS数据
        SWAPX   20H, f
        SWAPX   20H, w          ; 恢复W数据
        RETI                    ; 从中断返回
```

0Bh/8Bh/10Bh/18Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE	ADCIE	T2IE	TM1IE	TM0IE	WKTIE	INT2IE	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

INTIE.7 **ADCIE:** ADC 中断使能
0: 禁止
1: 使能

INTIE.6 **T2IE:** T2 中断使能
0: 禁止
1: 使能

INTIE.5 **TM1IE:** Timer1 中断使能
0: 禁止
1: 使能

INTIE.4 **TM0IE:** Timer0 中断使能
0: 禁止
1: 使能

INTIE.3 **WKTIE:** WKT断使能
0: 禁止
1: 使能

INTIE.2 **INT2IE:** INT2 (PA7) 中断使能
0: 禁止
1: 使能

INTIE.1 **INT1IE:** INT1 (PA1) 中断使能
0: 禁止
1: 使能

INTIE.0 **INT0IE:** INT0 (PA2或PA6) 中断使能
0: 禁止
1: 使能

0Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIF	ADCIF	T2IF	TM1IF	TM0IF	WKTIF	INT2IF	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

0Ch.7 **ADCIF:** ADC中断事件挂起标志
当ADC转换结束后由H/W 置位，对此位写0将清除该标志

0Ch.6 **T2IF:** T2中断事件挂起标志
当T2溢出时由H/W置位，对此位写0将清除该标志

0Ch.5 **TM1IF:** Timer1中断事件挂起标志
Timer1溢出时由H/W置位，对此位写0将清除该标志

0Ch.4 **TM0IF:** Timer0中断事件挂起标志
当Timer0溢出时由H/W置位，对此位写0将清除该标志

0Ch.3 **WKTIF:** WKT中断事件挂起标志
当WKT超时时由H/W置位，对此位写0将清除该标志

0Ch.2 **INT2IF:** INT2 (PA7)引脚下降沿中断挂起标志
INT2引脚发生下降沿时由H/W置位，对此位写0将清除该标志

0Ch.1 **INT1IF:** INT1 (PA1)引脚下降沿/上升沿中断挂起标志
当INT1引脚发生下降/上升沿时由H/W置位，对此位写0将清除该标志

0Ch.0 **INT0IF:** INT0 (PA2或PA6)引脚下降沿/上升沿中断事件挂起标志
当INT0引脚发生下降/上升沿时由H/W置位，对此位写0将清除该标志

0Dh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE1	-	-	-	-	PWM1PIE	PWM1CIE	PWM1BIE	PWM1AIE
R/W	-	-	-	-	R/W	R/W	R/W	R/W
复位值	-	-	-	-	0	0	0	0

0Dh.3 **PWM1PIE:** PWM1周期中断使能
0：禁止
1：使能

0Dh.2 **PWM1CIE:** PWM1C占空比中断使能
0：禁止
1：使能

0Dh.1 **PWM1BIE:** PWM1B占空比中断使能
0：禁止
1：使能

0Dh.0 **PWM1AIE:** PWM1A占空比中断使能
0：禁止
1：使能

0Eh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIF1	-	-	-	-	PWM1PIF	PWM1CIF	PWM1BIF	PWM1AIF
R/W	-	-	-	-	R/W	R/W	R/W	R/W
复位	-	-	-	-	0	0	0	0

0Eh.3 **PWM1PIF:** PWM1周期中断事件挂起标志
在PWM1计数器计数到设置的周期后由H/W置位，对此位写0将清除该标志

0Eh.2 **PWM1CIF:** PWM1C占空比中断事件挂起标志
在PWM1计数器计数到设置的PWM1C占空比后由H/W置位，对此位写0将清除该标志

0Eh.1 **PWM1BIF:** PWM1B占空比中断事件挂起标志
在PWM1计数器计数到设置的PWM1B占空比后由H/W置位，对此位写0将清除该标志

0Eh.0 **PWM1AIF:** PWM1A占空比中断事件挂起标志
在PWM1计数器计数到设置的PWM1A占空比后由H/W置位，对此位写0将清除该标志

81h/181h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPTION	HWAUTO	INT0EDG	INT1EDG	INT0SEL	WDTPSC		WKTPSC	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	1	1	1	1

81h.6 INT0EDG: INT0 引脚中断边沿触发
0：INT0引脚下降沿触发
1：INT0引脚上升沿触发

81h.5 INT1EDG: INT1 引脚中断边沿触发
0：INT1引脚下降沿触发
1：INT1引脚上升沿触发

81h.4 INT0SEL: INT0 管脚选择
0：PA6 引脚
1：PA2 引脚

5 I/O 端口

5.1 PA0-PA7

这些引脚可用作施密特触发器输入，CMOS 推挽输出。上拉电阻可通过 S/W 设置分配给每个引脚。要在施密特触发器输入模式下使用该引脚，S / W 需要将 I/O 引脚设置为模式 0 或模式 1 且 PxD = 1。读取引脚数据（PxD）具有不同的含义。在“读-修改-写”指令中，CPU 实际上是读取输出数据寄存器。在其他指令中，CPU 读取引脚状态。 所谓的“读-修改-写”指令包括 BSX，BCX 和所有指令。

这些引脚可以在以下四种不同模式下运行。.

模式	PA0~PA7 引脚功能	PxD SFR 数据	平角状态	上拉电阻	数字输入
模式 0	开漏	0	低驱动	N	N
	输入	1	上拉	Y	Y
模式 1	开漏	0	低驱动	N	N
		1	Hi-Z	N	Y
模式 2	CMOS 输出	0	低驱动	N	N
		1	高驱动	N	N
Mode 3	ADC的模拟输入	X	-	N	N

模式 3

I/O 引脚功能表

除了 I / O端口功能外，每个引脚还具有一个或多个其他功能，例如 PWM 和 ADC。.

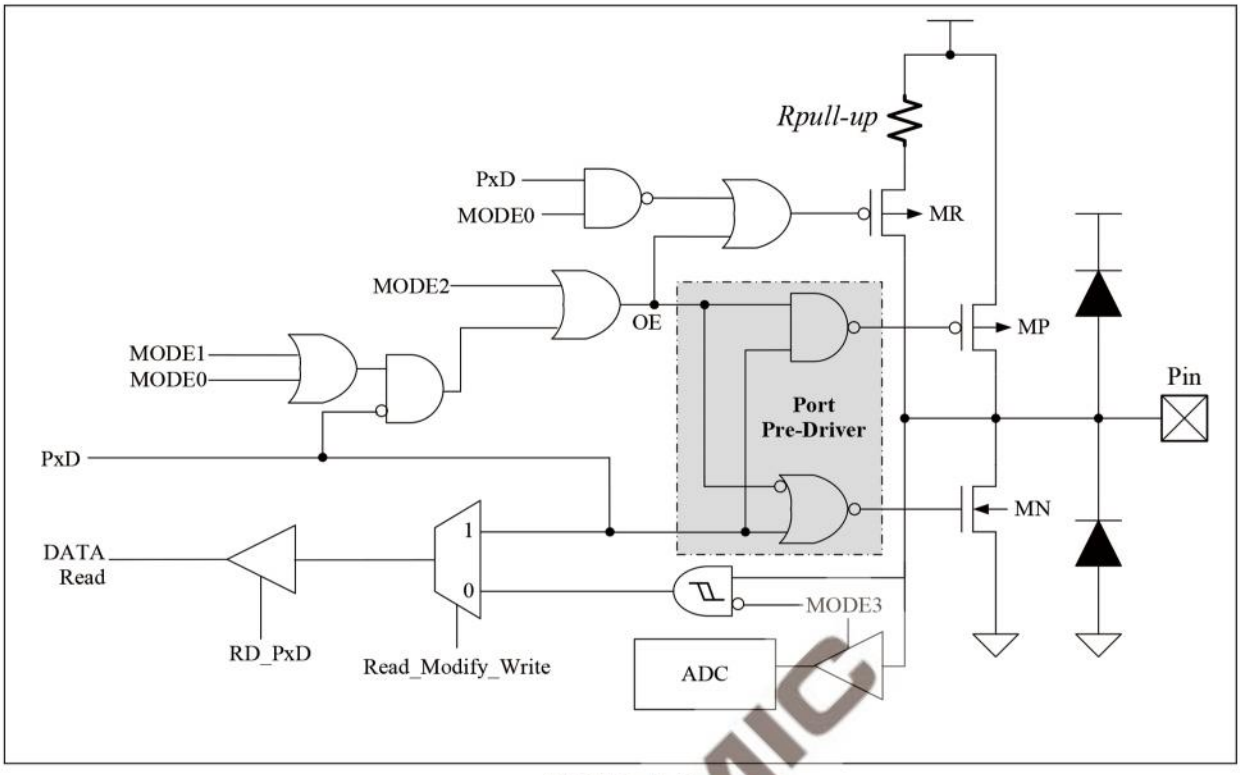
引脚名称	唤醒	ADC	其它	模式 3
PA0		ADC0	PWM1A	ADC0
PA1	INT1	ADC1	PWM0N	ADC1
PA2	INT0	ADC2	PWM1B	ADC2
PA3		ADC3	PWM1C	ADC3
PA4		ADC4	PWM1C	ADC4
PA5		ADC5	PWM0P	ADC5
PA6	INT0	ADC6	TM0CKI/PWM1A	ADC6
PA7	INT2	ADC7	PWM1B	ADC7

PortA 多功能表

下表列出了引脚替代功能所需的 SFR 设置.

替代功能	模式	PxD SFR 数据	引脚状态	其它必要的 SFR 设置
INT0, INT1, INT2 TM0CKI	0	1	输入带上拉	INTxIE TM0CTL
	1	1	输入	
ADC0~ADC7	3	X	ADC 通道	ADCHS
PWM0N, PWM0P, PWM1A, PWM1B, PWM1C	1	X	PWM 输出(开漏)	PWMOE
	2	X	PWM 输出(COMS 输出)	

端口替代功能的模式设置



通用引脚结构

◇ 范例：将 PA0 设置为施密特触发输入带上拉(模式 0)

```
MOVLW     xxxxxxx1B
MOVWX     PAD
MOVLW     xxxxxx00B
MOVWX     PAMODL          ; 将PA0设置为施密特触发输入带上拉
```

◇ 范例：将 PA0 设置为施密特触发输入不带上拉(模式 1)

```
MOVLW     xxxxxxx1B
MOVWX     PAD
MOVLW     xxxxxx01B
MOVWX     PAMODL          ; 将PA0设置为施密特触发输入不带上拉
```

◇ 范例：将 PA0 设置为 CMOS 推挽输出模式 (模式 2)

```
MOVLW     xxxxxx10B
MOVWX     PAMODL
```

◇范例：将 PA0 设置为 ADC0 模拟输入模式(模式 3)

```
MOVLW     xxxxxx11B
MOVWX     PAMODL          ;将PA0设为模式3
```

8Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PAMODH	PA7MOD		PA6MOD		PA5MOD		PA4MOD	
R/W	R/W		R/W		R/W		R/W	

复位	0	0	0	1	0	1	0	1

8Ch.7~6 **PA7MOD**: PA7引脚模式控制
00: 模式0
01: 模式1
10: 模式2
11: 模式3, PA7作为ADC7通道输入

8Ch.5~4 **PA6MOD**: PA6引脚模式控制
00: 模式0
01: 模式1
10: 模式2
11: 模式3, PA6作为ADC6通道输入

8Ch.3~2 **PA5MOD**: PA5引脚模式控制
00: 模式0
01: 模式1
10: 模式2
11: 模式3, PA5作为ADC5通道输入

8Ch.1~0 **PA4MOD**: PA4引脚模式控制
00: 模式0
01: 模式1
10: 模式2
11: 模式3, PA4作为ADC4通道输入

8Dh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PAMODL	PA3MOD		PA2MOD		PA1MOD		PA0MOD	
R/W	R/W		R/W		R/W		R/W	
复位	0	1	0	1	0	1	0	1

8Dh.7~6 **PA3MOD**: PA3引脚模式控制
00: 模式0
01: 模式1
10: 模式2
11: 模式3, PA3作为ADC3通道输入

8Dh.5~4 **PA2MOD**: PA2引脚模式控制
00: 模式0
01: 模式1
10: 模式2
11: 模式3, PA2作为ADC2通道输入

8Dh.3~2 **PA1MOD**: PA1引脚模式控制
00: 模式0
01: 模式1
10: 模式2
11: 模式3, PA1作为ADC1通道输入

8Dh.1~0 **PA0MOD**: PA0引脚模式控制
00: 模式0
01: 模式1
10: 模式2
11: 模式3, PA0作为ADC0通道输入

05h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PAD	PAD							
R/W	R/W							
复位	1	1	1	1	1	1	1	1

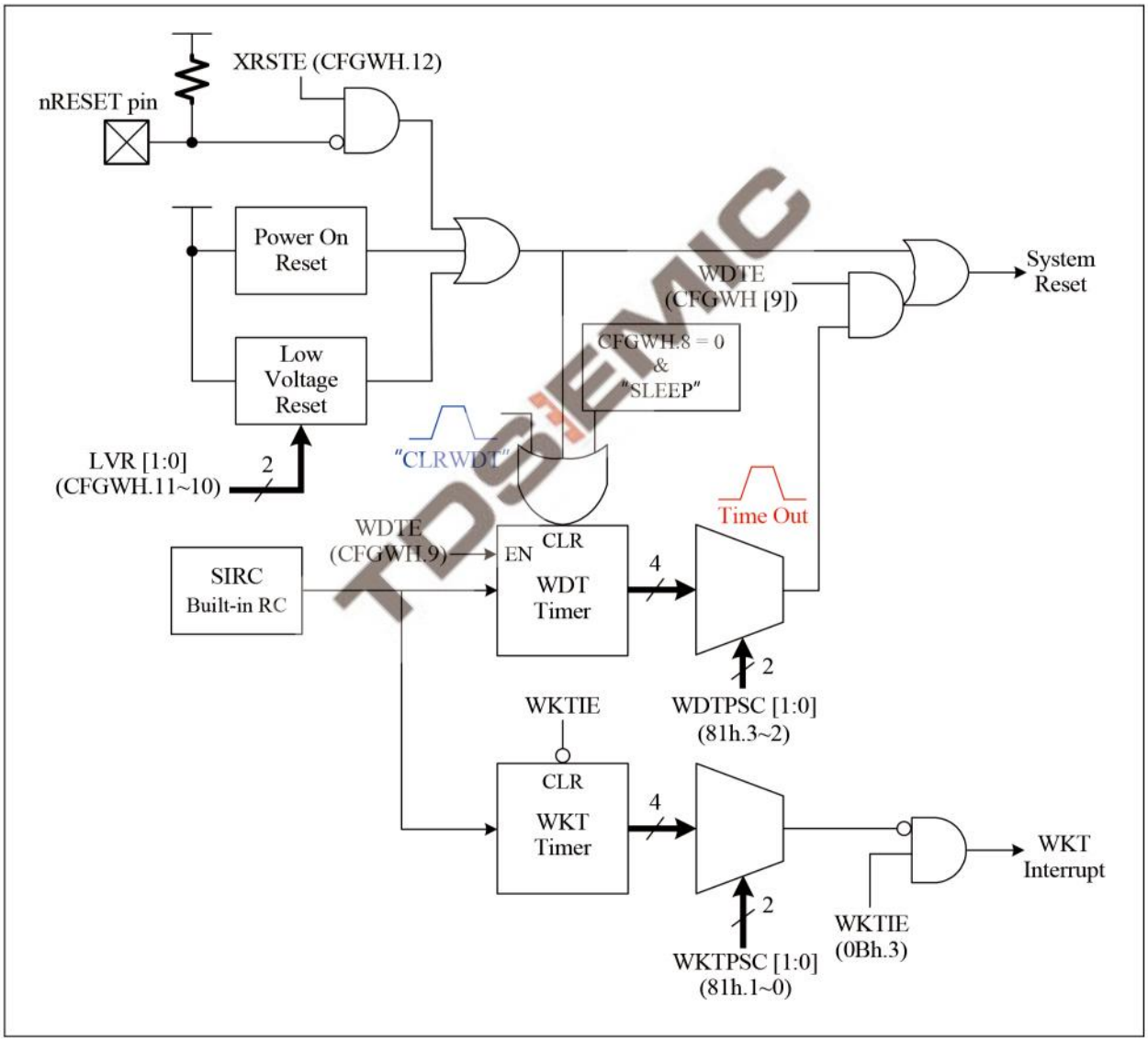
05h.7~0 **PAD**: PA7~PA0数据

6 外围功能模块

6.1 看门狗(WDT)/唤醒定时器 (WKT)

WDT 和 WKT 共享相同的内部 RC 振荡器，并且具有各自的计数器。WDT，WKT 的溢出周期可以通过单独的预分频（WDTPSC [1：0]，WKTPSC [1：0]）选择。WDT 定时器由 CLRWDT 指令清零。如果使能了看门狗（CFGWH.9 = WDTE = 1），则 WDT 会产生芯片复位信号。 将 CFGWH.8 设置为 0 可以使 WDT 定时器在执行 SLEEP 指令后停止计数，CFGWH.8 = 1 即使执行 SLEEP 指令，WDT 定时器也始终保持计数。

WKT 是一个间隔定时器，WKT 超时将产生 WKT 中断标志（WKTIF）。WKTIE = 0 清除/停止 WKT 定时器。 设置 WKTIE = 1，无论 CPU 在何种工作模式下，WKT 定时器都会一直计数



WDT/WKT 框图

WDT 在不同模式下的行为如下表所示

模式	WDTE[1]	WDTE[0]	WDT
正常模式	0	0	停止
	0	1	停止
	1	0	运行
	1	1	运行
掉电模式（SLEEP）	0	0	停止
	0	1	停止
	1	0	停止
	1	1	运行

通过 CLRWDT 指令可以清除看门狗定时器

◇ 范例：通过 CLRWDT 指令清除看门狗定时器.

```
MAIN:   …                       ; 执行程序
        CLRWDT                  ; 执行CLRWDT指令.
        …
        GOTO        MAIN
```

◇ 范例:设置 WDT 时间并在执行 SLEEP 指令后禁止.

```
        MOVLW       0000011 1B
        MOVWX       OPTION      ; 选择WDT超时t=256ms @5V
        …
        SLEEP
```

◇ 范例：设置 WKT 周期和中断功能.

```
        MOVLW       00000110B
        MOVWX       OPTION      ; 选择WKT周期=64 ms @5V.
        MOVLW       11110111B   ; 通过字节操作清除WKT中断请求标志
        MOVWX       INTIF       ;不要使用位操作“BCX WKTIF”清除中断标志
        MOVLW       00001000B
        MOVWX       INTIE       ; 使能 WKT中断功
```

03h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STATUS	IRP	RP1	RP0	TO	PD	Z	DC	C
R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

03h.4 **TO:**WDT超时标志，只读
0:上电复位，LVR复位或CLRWDT / SLEEP指令后
1: WDT发生超时

0Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIF	ADCIF	T2IF	TM1IF	TM0IF	WKTIF	INT2IF	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

0Ch.3 **WKTIF:** WKT中断事件挂起标志
WKT 超时时由H/W置位，对此位写0将清除该标志

0Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE	ADCIE	T2IE	TM1IE	TM0IE	WKTIE	INT2IE	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

0Bh.3 **WKTIE:** WKT中断使能
0: 禁止
1: 使能

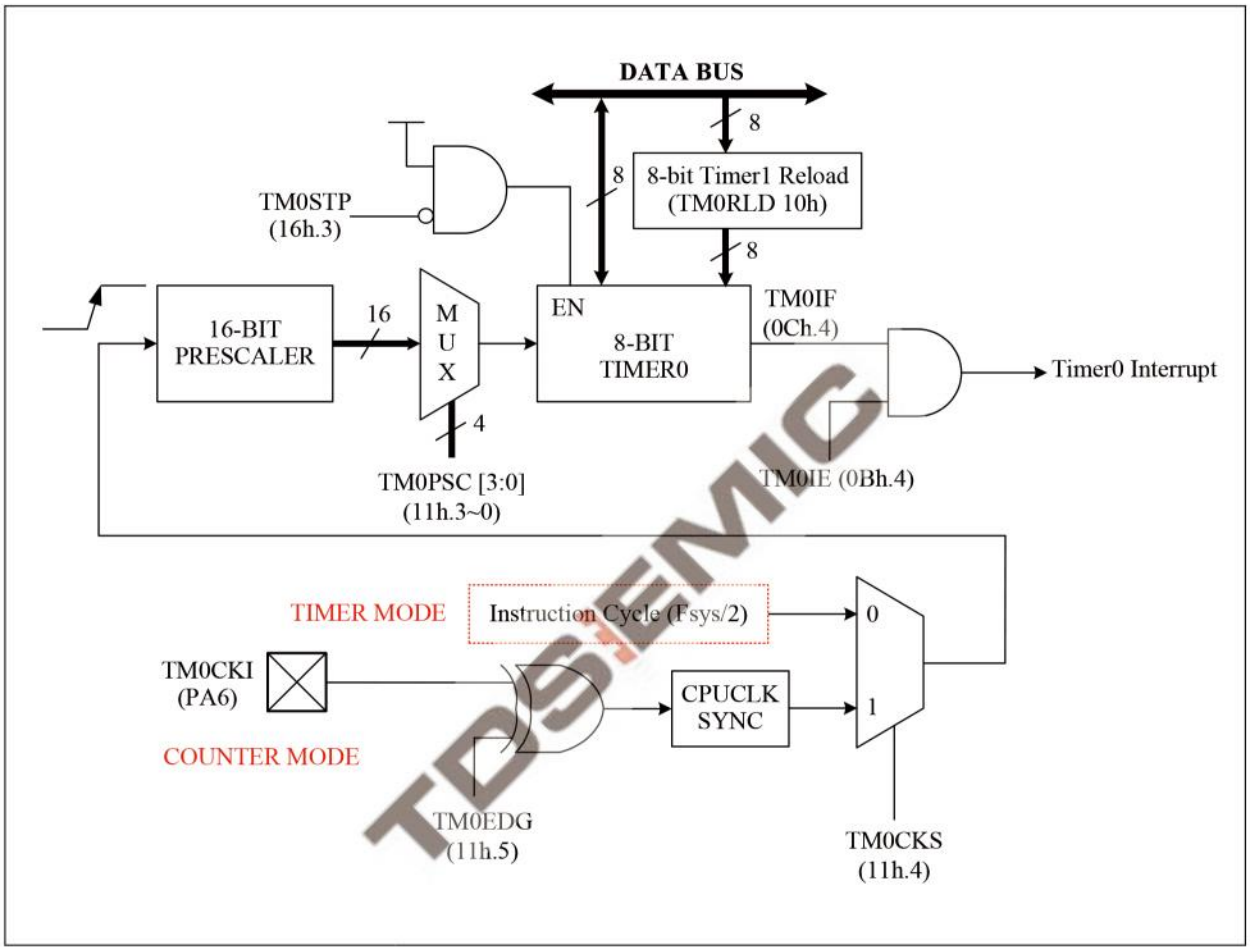
81h/181h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPTION	HWAUTO	INT0EDG	INT1EDG	INT0SEL	WDTPSC		WKTPSC	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	1	1	1	1

81h.3~2 **WDTPSC:** WDT 周期(@VCC=5V)
00: 128 ms
01: 256 ms
10: 1024 ms
11: 2048 ms

81h.1~0 **WKTPSC:** WKT 周期(@VCC=5V)
00: 16 ms
01: 32 ms
10: 64 ms
11: 128 ms

6.2 Timer0

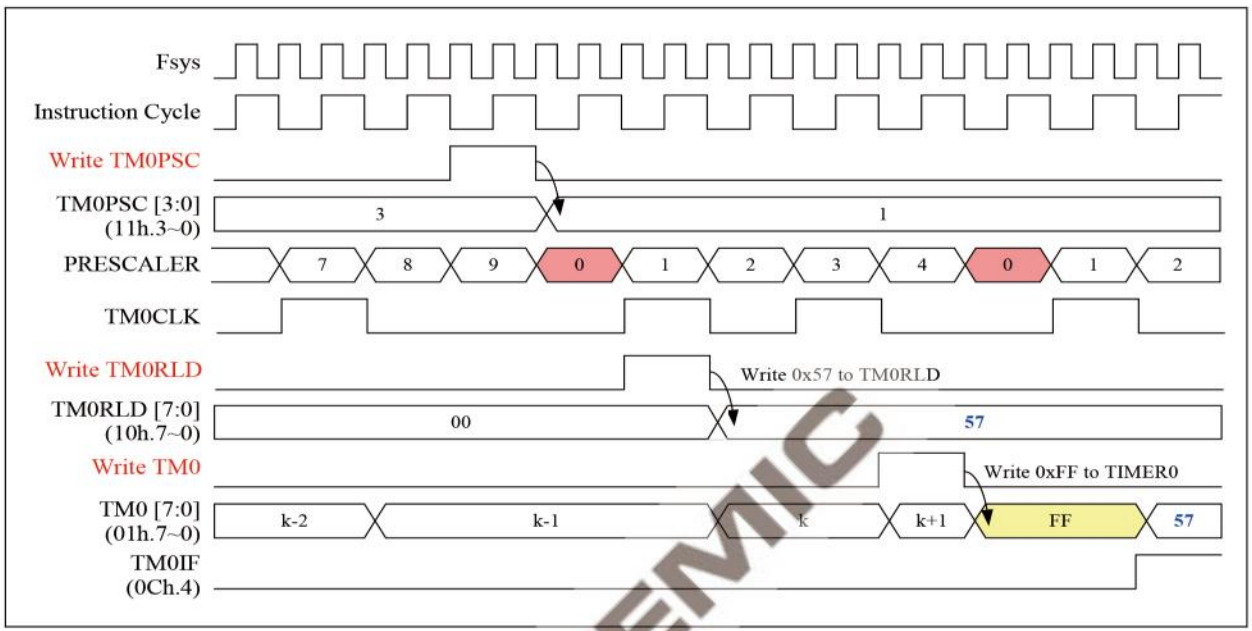
Timer0 是一个 8 位宽的寄存器 01h（TM0）。和其他任何寄存器一样可以读取或写入。此外，Timer0 会定期自行递增，并根据预分频的时钟源（可以是 Fsys/2 或 TM0CKI（PA6）上升/下降输入）跳转时自动重载新的“偏移值”（TM0RLD）。 Timer0 的增加速率由“Timer0 预分频”（TM0PSC）寄存器决定。当 Timer0 的计数翻转时，它总是产生 TM0IF。如果（TM0IE）被置位，它将产生 Timer0 中断。如果 TM0STP 位置 1，可以停止 Timer0 的计数。



Timer0 框图

下图描述了 Timer0 在纯定时器模式下工作.

写 Timer0 预分频（TM0PSC）时，内部 8 位预分频将清零，以使 Timer0 第一次计数周期正确。TM0CLK 是导致 Timer0 在 TM0CLK 结束时增加 1 的内部信号。TM0WR 也是内部信号，表示 Timer0 是通过指令直接写入的。同时，内部 8 位预分频将被清除。当 Timer0 从 FFh 计数到 TM0RLD 时，如果将 TM0IE（Timer0 中断使能）置位，TM0IF（Timer0 中断标志）将被置 1 并产生中断.



Timer0 在定时器模式下工作(TM0CKS=0)

TM0 中断时间的计算公式如下:

TM0 中断间隔周期时间 = Fsys / 2 / TM0PSC / (256-TM0RLD)

◇ 范例: 如果 Fsys=8 MHz，设置定时器 0 在定时器模式下工作

```
; 设置TM0时钟源并分频
        BSX     CPUCKS          ; 将快速时钟设置为系统时钟
        MOVLW   00x00101B       ; TM0CKS=0，Timer0时钟为指令周期
        MOVWX   TM0CTL          ; TM0PSC = 0101b,除以32

; Setup Timer0 reload data
        MOVLW   80H
        MOVWX   TM0RLD          ; 设置Timer0周期数据 = 128

; Setup Timer0
        BSX     TM0STP          ; Timer0 停止计数
        CLRX    TM0             ; 清除计时器0内容

; 使能 Timer0 and interrupt function
        MOVLW   11101111B
        MOVWX   INTIF           ; 清除Timer0中断请求标志
        BSX     TM0IE           ; 使能Timer0中断功能
        BCX     TM0STP          ; 使能Timer0计数
```

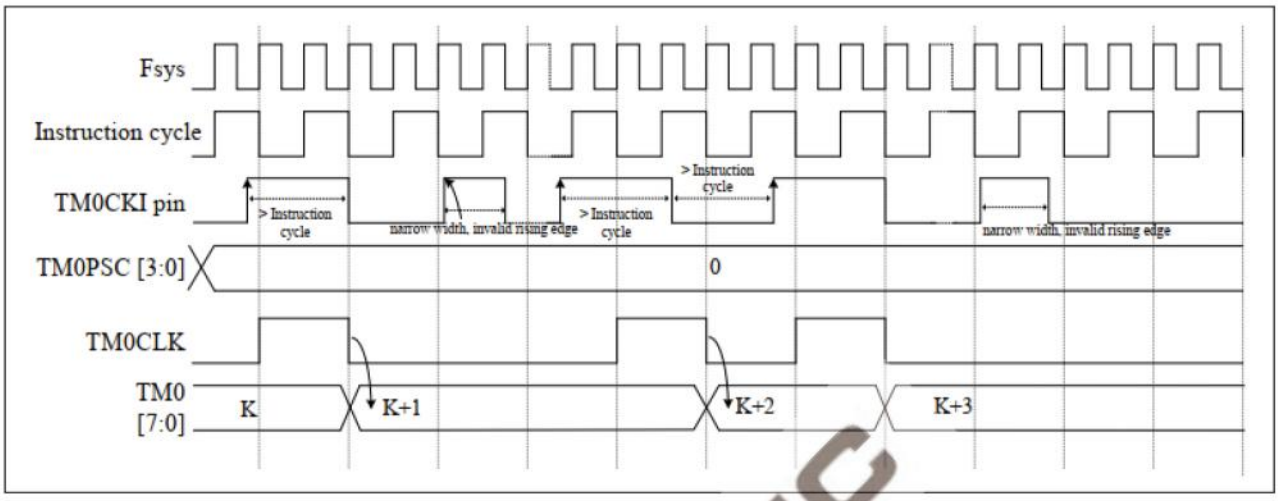
Timer0 中断频率 = Fsys / 2 / TM0PSC / (256-TM0RLD),

Fsys = 8MHz, TM0PSC =除以 32

Timer0 中断频率 = 8 MHz /2 / 32 / (256-128) = 0.976 KHz

下图描述了 Timer0 在计数器模式下的工作。.

如果 TM0CKS = 1，则 Timer0 计数器的源时钟来自 TM0CKI 引脚。TM0CKI 信号通过指令周期（Fsys / 2）进行同步，这意味着 TM0CKI 的高/低持续时间必须长于一个指令周期时间（Fsys / 2），以确保同步器正确检测到每个 TM0CKI 的变化。



Timer0 在 TM0CKI（TM0EDG = 0），TM0CKS = 1 的计数器模式下工作

◇ 范例： 设置 TM0 在计数器模式下工作，并且时钟源来自 TM0CKI 引脚（PA6）

```
; 设置Timer0时钟源和分频器
        MOVLW   00110000B       ; TM0EDG = 1，计数沿为下降沿
        MOVWX   TM0CTL          ; TM0CKS = 1，Timer0时钟为TM0CKI
                                ; TM0PSC = 0000b, 除以1
; Setup Timer0
        BSX     TM0STP          ; Timer0停止计数
        CLRX    TM0             ; 清除Timer0内容

; 使能 Timer0 and read Timer0 counter
        BCX     TM0STP          ; 使能Timer0计数
        ...
        BSX     TM0STP          ; Timer0停止计数
        MOVXW   TM0             ; 读取Timer0内容
```

01h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TM0	TM0							
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

01h **TM0:** Timer0内容

0Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE	ADCIE	T2IE	TM1IE	TM0IE	WKTIE	INT2IE	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

0Bh.4 **TM0IE:** Timer0中断使能
0: 禁止 1: 使能

0Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIF	ADCIF	T2IF	TM1IF	TM0IF	WKTIF	INT2IF	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

0Ch.4 **TM0IF:** Timer0中断事件挂起标志
当Timer0溢出时由H/W置位，对此位写0将清除该标志

10h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TM0RLD	TM0RLD							
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

10h **TM0RLD:** Timer0重载数据

11h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TM0CTL	-	-	TM0EDG	TM0CKS	TM0PSC			
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位	-	-	0	0	0	0	0	0

11h.5 **TM0EDG:** TM0CKI 引脚的Timer0 预分频器计数沿
0: 上升沿 1: 下降沿

11h.4 **TM0CKS:** Timer0 预分频器时钟源
0:Fsys/2 1: TM0CKI引脚(PA6 引脚)

11h.3~0 **TM0PSC:** Timer0 预分频器。 Timer0 预分频器时钟源除以

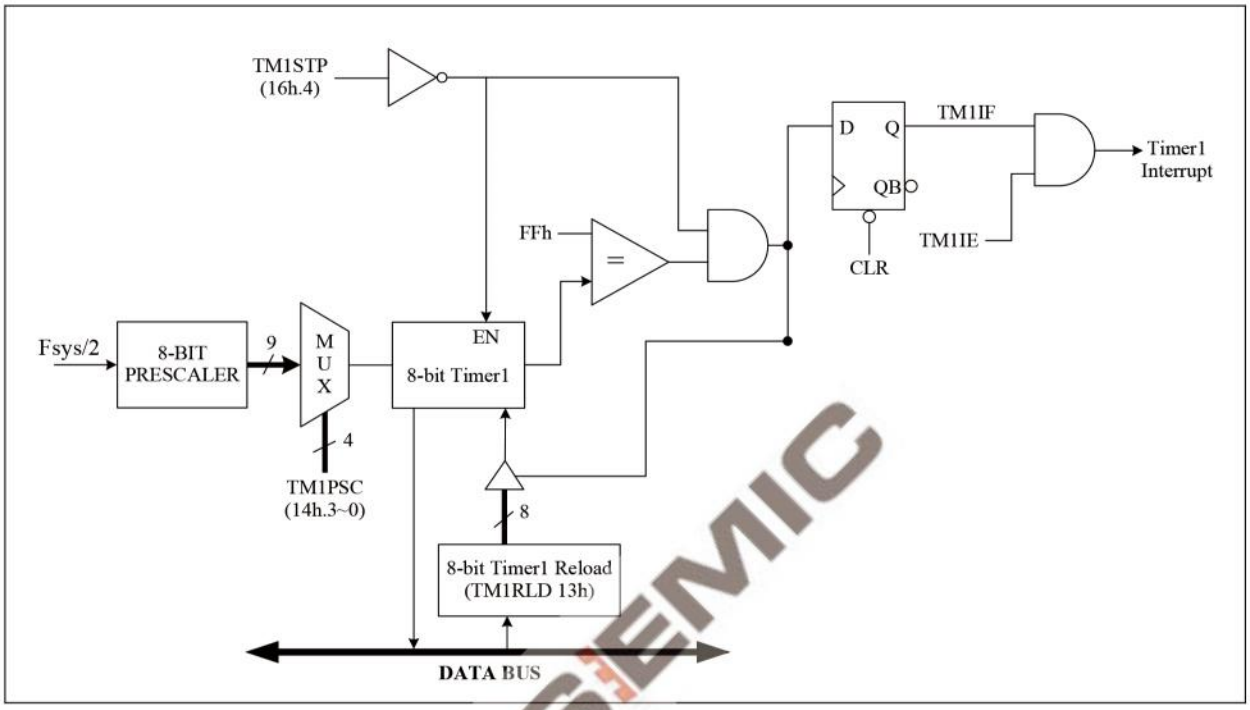
0000: /1	0001: /2	0010: /4	0011: /8
0100: /16	0101: /32	0110: /64	0111: /128
1000: /256	1001: /512	1010: /1024	1011: /2048
1100: /4096	1101: /8192	1110: /16384	1111: /32768

16h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MF016	LVDF	LVDEN	T2CLR	TM1STP	TM0STP	LVRSAV	LVDS	
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	-	0	1	0	0	1	0	1

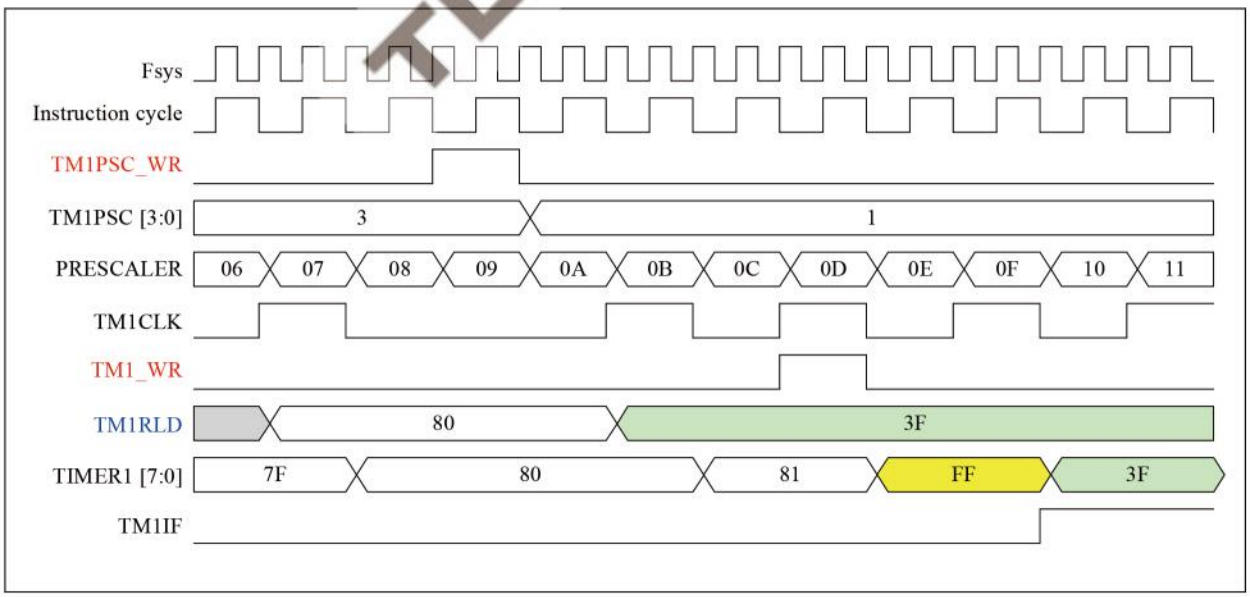
16h.3 **TM0STP:** Timer0计数器停止
0: Timer0计数器运行 1: Timer0计数器停止

6.3 Timer1

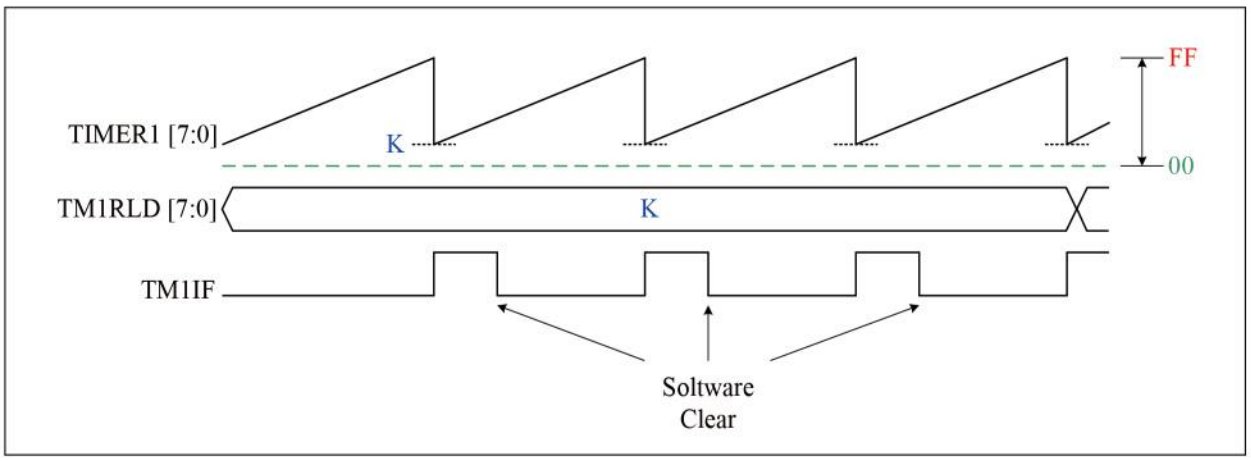
Timer1 是一个 8 位宽的寄存器。 和任何其他寄存器一样可以读取或写入。 此外，Timer1 会定期自行递增，并根据预分频的指令时钟（Fsys / 2）在翻转时自动重载新的“偏移值”（TM1RLD）。 Timer1 的增加速率由 TM1PSC 寄存器确定。 如果 TM1IE 位置 1，它将产生 Timer1 中断。 如果 TM1STP 位置 1，可以停止 Timer1 的计数。



Timer1 框图



Timer1 时序图



Timer1 重新加载图

◇ 范例: CPU 在慢速模式下运行，Fsys =慢时钟/ CPUPSC = 70 KHz / 2 = 35 KHz

```
; 设置Timer1时钟源和分频器
        MOVLW   00000010B       ; 将慢时钟设置为系统时钟
        MOVWX   CLKCTL          ; CPUPSC = 10b,除以2
        MOVLW   00000010B
        MOVWX   TM1CTL          ; TM1PSC = 0010b,除以8

;设置Timer1重新加载数据
        MOVLW   FFH
        MOVWX   TM1RLD          ; 设置Timer1重载数据= 255

; 设置Timer1
        BSX     TM1STP          ; Timer1停止计数
        CLRX    TM1             ; 清除Timer1内容

; 使能Timer1计数和中断功能
        MOVLW   11011111B
        MOVWX   INTIF           ; 清除Timer1请求中断标志
        BSX     TM1IE           ; 使能Timer1中断功能
        BCX     TM1STP          ; 使能Timer1计数
```

Timer1 时钟源为 Fsys / 2 = 35 KHz / 2 = 17.5 KHz，Timer1 除以 8
Timer1 中断频率= 17.5 KHz / 2/8 /（256-255）= 1.09 Hz

0Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE	ADCIE	T2IE	TM1IE	TM0IE	WKTIE	INT2IE	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

0Bh.5 **TM1IE:** Timer1中断使能
0: 禁止
1: 使能

0Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIF	ADCIF	T2IF	TM1IF	TM0IF	WKTIF	INT2IF	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

0Ch.5 **TM1IF:** Timer1中断事件挂起标志
当Timer1 溢出时由H/W置位，对此位写0将清除该标志

12h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TM1	TM1							
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

12h **TM1:** Timer1内容

13h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TM1RLD	TM1RLD							
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

13h.7~0 **TM1RLD:** Timer1翻转时重新加载偏移值

14h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TM1CTL	-	-	-	-	TM1PSC			
R/W	-	-	-	-	R/W	R/W	R/W	R/W
复位	-	-	-	-	0	0	0	0

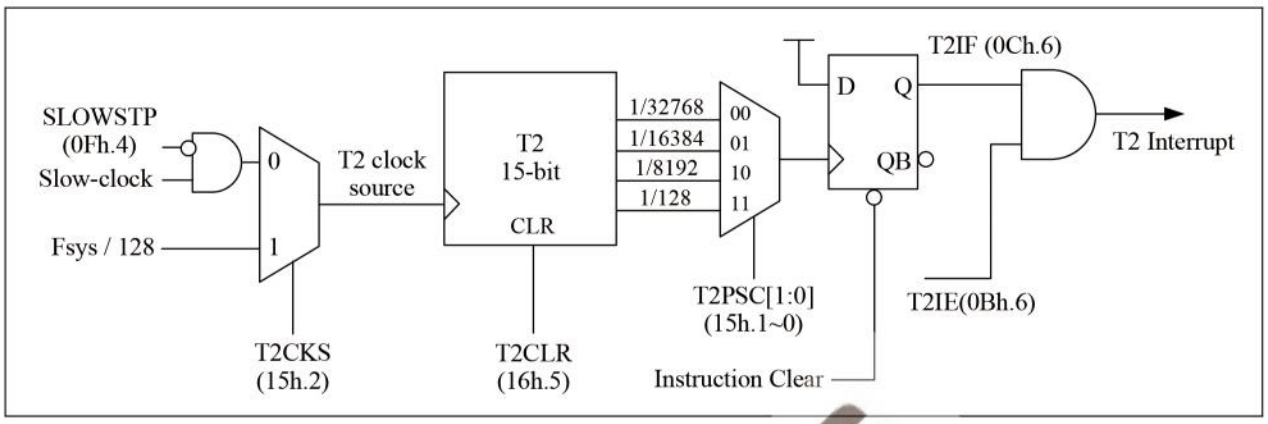
14h.3~0 **TM1PSC:** Timer1 预分频器。 Timer1 时钟源除以
0000: Fsys/2 0001: Fsys/4 0010: Fsys/8
0011: Fsys/16
0100: Fsys/32 0101: Fsys/64 0110: Fsys/128 0111: Fsys/256
1xxx: Fsys/512

16h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MF016	LVDF	LVDEN	T2CLR	TM1STP	TM0STP	LVRSAV	LVDS	
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	-	0	1	0	0	1	0	1

16h.4 **TM1STP:** Timer1 计数器停止
0: 计数器运行
1: 计数器停止

6.4 T2:15 位 Timer

T2 是一个 15 位计数器，时钟源来自 Fsys/128 或慢时钟。它用于生成时基中断和 T2 计数器模块时钟。指令无法读取 T2 内容。T2 时钟频率取决于寄存器 T2PSC[1：0]（15h.1～0）位，可以是除以 32768/16384/8192/128 的其一，T2 计时溢出将产生中断标志 T2IF（0Ch.6）。T2 功能如下框图所描述。



T2 框图

◇范例：CPU 以 FAST 模式运行，Fsys =快速时钟/ CPUPSC = FIRC 8MHz，T2 时钟源为 Fsys/128

```
; 设置FIRC频率
        MOVLW   00000111B
        MOVWX   CLKCTL          ; Fsys=8 MHz

; Setup T2 clock source and 除以ider
        MOVLW   00000101B       ; T2CKS(15h.2) = 1,T2时钟源=Fsys/128
        MOVWX   T2CTL           ; T2PSC(15h.1~0) = 1,除以16384
        BSX     T2CLR           ; T2CLR = 1,清除T2内容

; 使能T2中断功能
        MOVLW   10111111B
        MOVWX   INTIF           ; 清除T2中断请求标志
        BSX     T2IE            ; 使能T2中断功能
        BCX     T2CLR           ; T2CLR = 0,使能T2计数
```

T2 时钟源是 Fsys/128 = 8 MHz/128 = 62500 Hz, T2PSC = /16384

T2 频率= 62500 Hz / 16384 = 3.815 Hz

0Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE	ADCIE	T2IE	TM1IE	TM0IE	WKTIE	INT2IE	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

0Bh.6 **T2IE:** T2中断使能
0: 禁止
1: 使能

0Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIF	ADCIF	T2IF	TM1IF	TM0IF	WKTIF	INT2IF	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

0Ch.6 **T2IF:** T2中断事件挂起标志
当T2溢出时由H/W置位，对此位写0将清除该标志

0Fh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CLKCTL	-	-	-	SLOWSTP	FASTSTP	CPUCKS	CPUPSC	
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
复位	-	-	-	0	1	0	1	1

0Fh.4 **SLOWSTP:** 在停止模式下停止慢速时钟
0: 不停止 p 1: 停止

15h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T2CTL	-	-	-	-	-	T2CKS	T2PSC	
R/W	-	-	-	-	-	R/W	R/W	R/W
复位	-	-	-	-	-	0	0	0

15h.2 **T2CKS:** “T2 时钟源选择.
1: Fsys/128 0:慢速时钟
15h.1~0 **T2PSC:** T2 预分频器。“T2 时钟源”除以-
00: 32768 01: 16384 10: 8192 11: 128

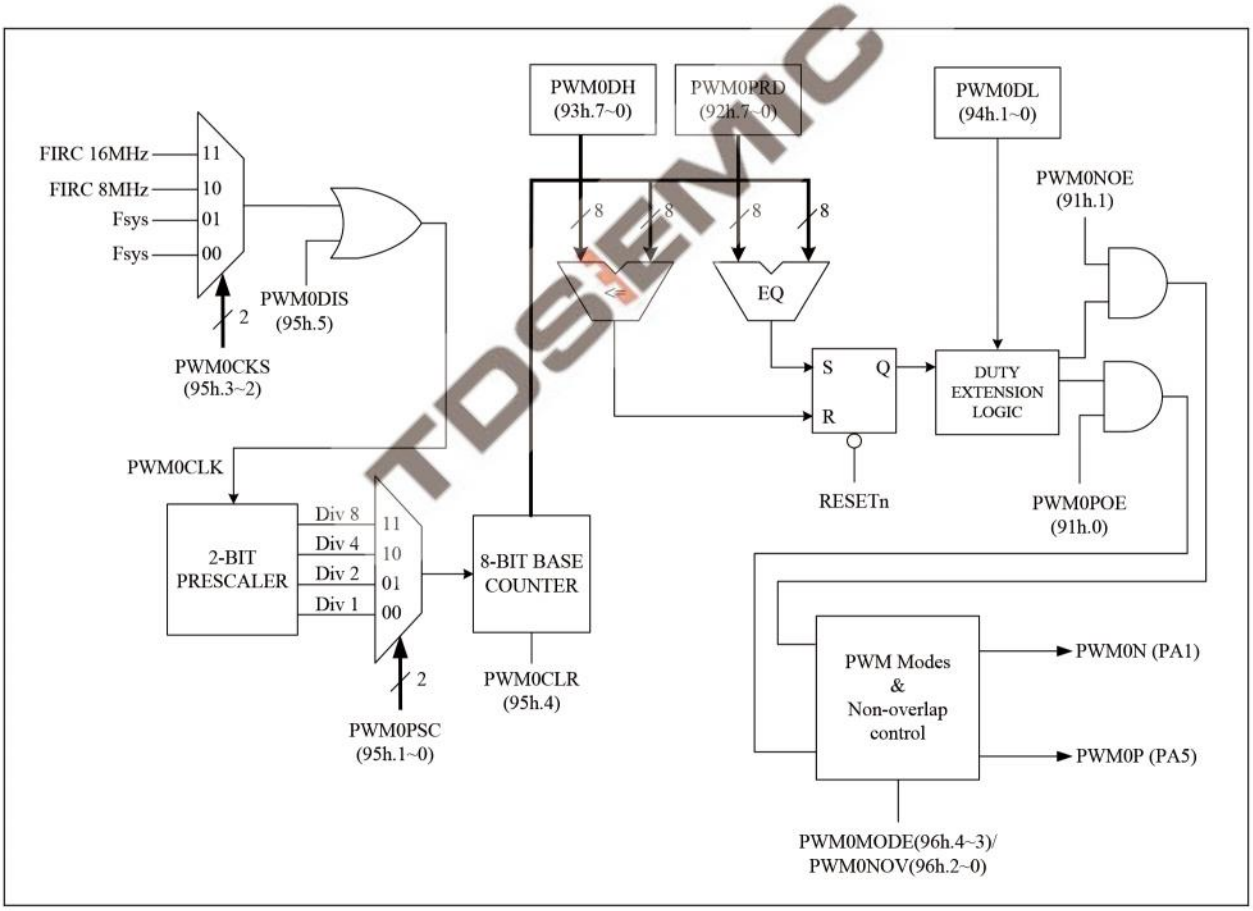
16h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MF016	LVDF	LVDEN	T2CLR	TM1STP	TM0STP	LVRSAV	LVDS	
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	-	0	1	0	0	1	0	1

16h.5 **T2CLR:** T2计数器停止
0: 计数器运行 1: 计数器停止

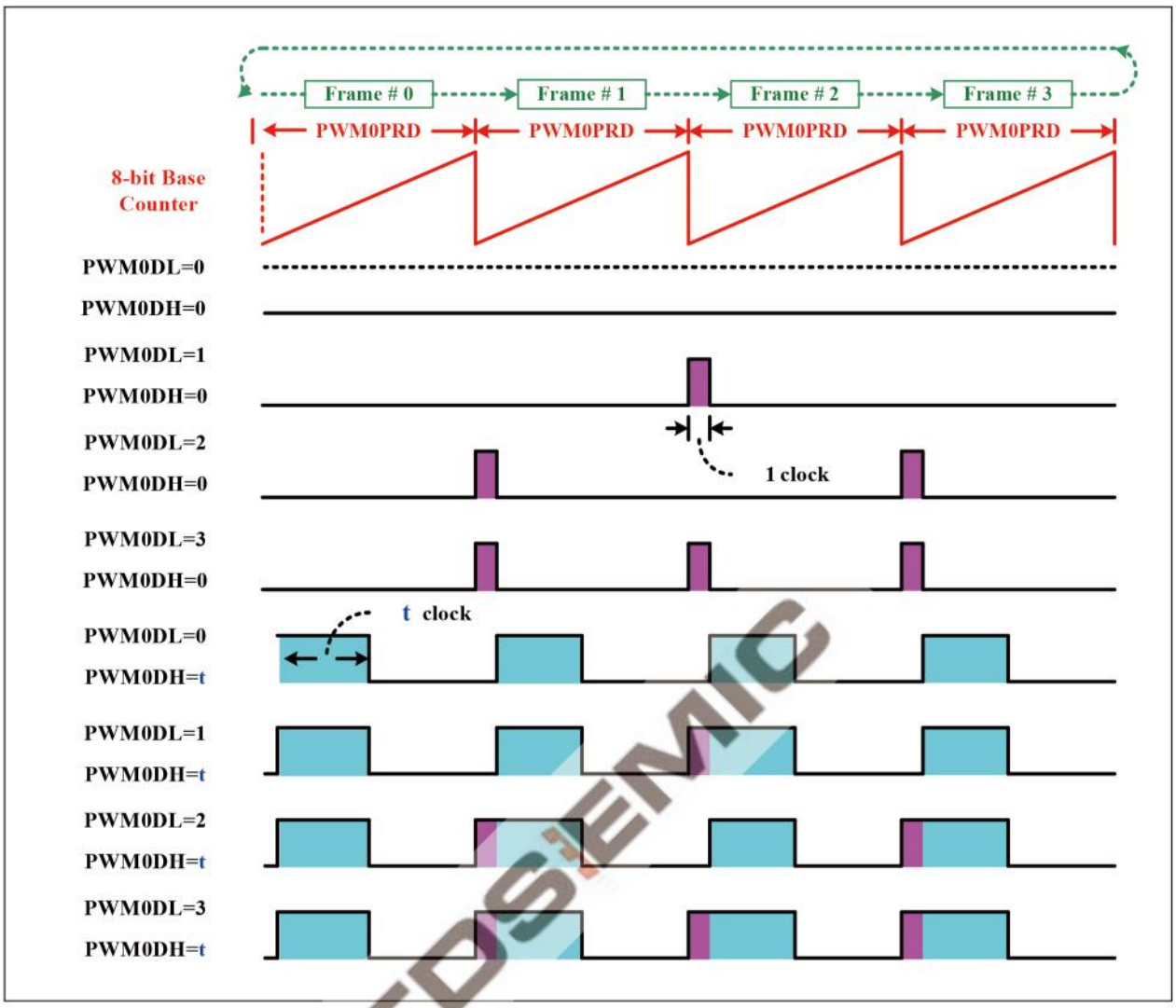
6.5 PWM0：(8+2) 位 PWM

PWM0 可以基于 PWM0CLK 生成具有 1024 占空比分辨率的各种频率波形，PWM0 时钟可以通过 PWM0CKS（95h.3～2）位决定选择 Fsys 或 FIRC 8MHz 或 FIRC 16MHz。扩展 LSB 技术允许 PWM0 以“由 256 除以 PWM0CLK”运行其频率，而不是“由 1024 除以 PWM0CLK”运行，这意味着 PWM 比正常速度快 4 倍。 较高的 PWM 频率的优点在于，后置 RC 滤波器可以将 PWM 信号转换为更稳定的直流电压。 每当 8 位基数计数器与 PWM 占空比寄存器 PWM0DH（93h.7～0）的 8 位 MSB 匹配时，PWM 输出信号就会复位为低电平。 当基数计数器翻转时，PWM 占空比寄存器 PWM0DL（94h.1～0）的 2 位 LSB 决定是立即将 PWM 输出信号设置为高电平，还是在一个时钟周期延迟后将其设置为高电平。 **复位后 PWM0 时钟是使能的而且 PWM0 不会保持固定。（参阅 PWM0DIS 和 PWM0CLR 的描述）**

可以通过将周期值写入 PWM0PRD 寄存器（92h.7～0）来设置 PWM0 周期。 请注意，更改 PWM0PRD 将立即更改 PWM0PRD 值，这与 PWM0DH / PWM0DL 不同，后者具有在当前周期结束时更新占空比的缓冲区。 程序员必须注意观察下图的当前时间来更改 PWM0PRD。 有一个数字比较器比较 PWM0 计数器和 PWM0PRD，如果在设置 PWM0PRD 之后 PWM0 计数器大于 PWM0PRD，将产生错误的长 PWM 周期，因为 PWM0 计数器必须计数到溢出然后继续计数到 PWM0PRD 才能完成周期。



PWM0 框图



PWM0 8+2 时序图

◇范例: CPU 在快速模式下运行，Fsys = FIRC 8 MHz

```
; 设置引脚模式
        MOVLW     xxxx10xxB          ; PA1引脚模式 = 模式2
        MOVWX     PAMODL             ; 模式2: CMOS输出

        MOVLW     xxxx10xxB          ; PA5 引脚模式 = 模式2
        MOVWX     PAMODH             ; 模式2: CMOS输出

; Setup PWM0 clock prescaler
        MOVLW     xx01 10 11B        ; 95h.4 = 1, PWM0清除并保持
        MOVWX     PWM0CTL            ; 95h.3~2 = 2, PWM0时钟源 = FIRC 8MHz
                                     ; 95h.1~0 = 3, PWM0预分频除以8
; Setup PWM0 mode & Non-overlap control
        MOVLW     xxx00 000B         ; 96h.4~3 = 0, PWM0 模式 = 模式0
        MOVWX     PWM0CTL1           ; 96h.2~0 = 0, PWM0非重叠时间= 0

        MOVLW     7FH
        MOVWX     PWM0PRD            ;设置 PWM0周期 = 7FH

        MOVLW     xxxxxx00B
        MOVWX     PWM0DL             ; 设置PWM0DL占空比 = 00H

        MOVLW     20H
        MOVWX     PWM0DH             ; 设置PWM0DH占空比 = 20H

        MOVLW     xxxxxx11B          ; 91h.1 = 1,使能PWM0N 输出至PA1
        MOVWX     PWMOE              ; 91h.0 = 1,使能PWM0P 输出至PA5

        BCX       PWM0CLR            ; 95h.4 = 0,释放PWM0 清除并保持
```

说明:

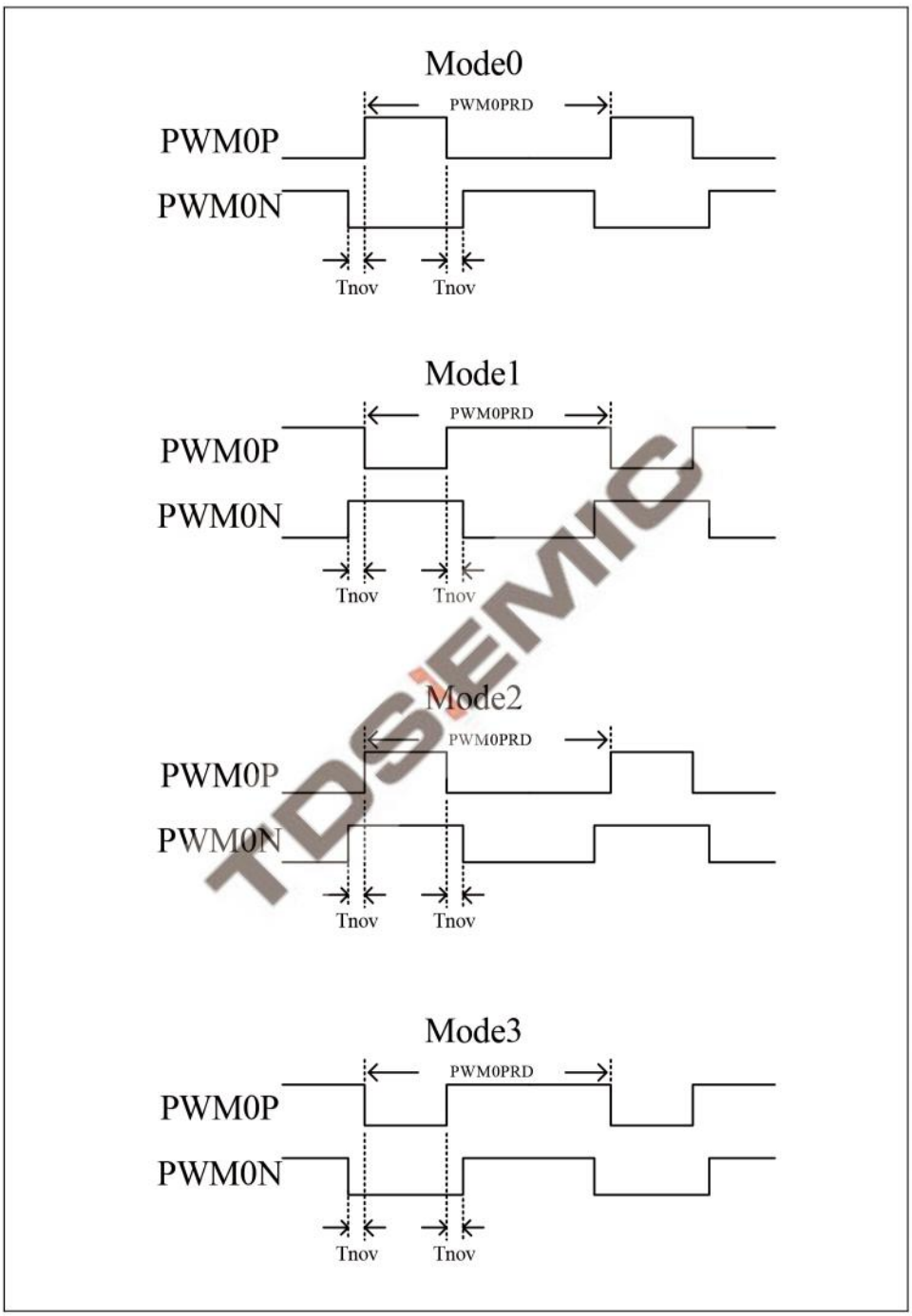
PWM0 时钟源 = FIRC 8M, PWM0PSC = 除以 8, PWM0PRD = 7FH,

PWM0DL = 00H, PWM0DH = 20H

PWM0 输出频率= 8 MHz / 8 / (PWM0PRD+1) = 8 MHz / 8 / 128 = 7.8125 KHz.

PWM0P 输出占空比 = 32:128 = 25 %.

可以通过 PWM0P 和 PWM0N 以四种不同模式输出 PWM0。PWM 脉冲的边沿可以通过 6 个不同的非重叠时间时钟间隔（Tnov），0s，4 个 PWM0CLK，5 个 PWM0CLK，6 个 PWM0CLK，7 个 PWM0CLK 和 8 个 PWM0CLK 进行分隔，这些间隔由 PWM0NOV（96h.2～0 ）选择。默认输出形式为模式 0。四种输出模式的波形如下图所示。



PWM0 波形模式

91h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM0E	PWM1COE1	PWM1COE0	PWM1BOE1	PWM1BOE0	PWM1AOE1	PWM1AOE0	PWM0NOE	PWM0POE
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

91h.1 **PWM0NOE:**使能 PWM0N 输出至 PA1
0：禁止
1：使能，PWM0N 输出至PA1

91h.0 **PWM0POE:**使能 PWM0P 输出至 PA5
0：禁止
1：使能，PWM0P 输出至PA5

92h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM0PRD	PWM0PRD							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	1	1	1	1	1	1	1	1

92h.7~0 **PWM0PRD:** PWM0 周期数据

93h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM0DH	PWM0DH							
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

93h.7~0 **PWM0DH:** PWM0占空比8位MSB

94h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM0DL	-	-	-	-	-	-	PWM0DL	
R/W	-	-	-	-	-	-	R/W	R/W
复位	-	-	-	-	-	-	0	0

94h.1~0 **PWM0DL:** PWM0占空比2位LSB

95h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM0CTL	-	-	PWM0DIS	PWM0CLR	PWM0CKS		PWM0PSC	
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
复位	-	-	0	0	0	0	0	0

95h.5 **PWM0DIS:** PWM0 时钟禁止
0：时钟使能
1：时钟禁止

95h.4 **PWM0CLR:** PWM0 清除并保持
0：PWM0 使能
1：PWM0 清除并保持

95h.3~2 **PWM0CKS:** PWM0 时钟源选择
0x：Fsys
10:FIRC 8MHz
11:FIRC 16MHz

95h.1~0 **PWM0PSC:** PWM0 时钟源预分频器
00：除以1
01：除以2
10：除以4
11：除以 8

96h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM0CTL1	-	-	-	PWM0MODE		PWM0NOV		
R/W	-	-	-	R/W	R/W	R/W	R/W	R/W
复位	-	-	-	0	0	0	0	0

96h.4~3 **PWM0MODE:** PWM0 不同输出模式
00: 模式 0
01: 模式 1
10: 模式 2
11: 模式 3

96h.2~0 **PWM0NOV:** PWM0 非重叠控制
000:原始 PWM0
001: 非重叠 4 个 PWM0CLK
010: 非重叠 5 个 PWM0CLK
011: 非重叠 6 个 PWM0CLK
100: 非重叠 7 个 PWM0CLK
101: 非重叠 8 个 PWM0CLK

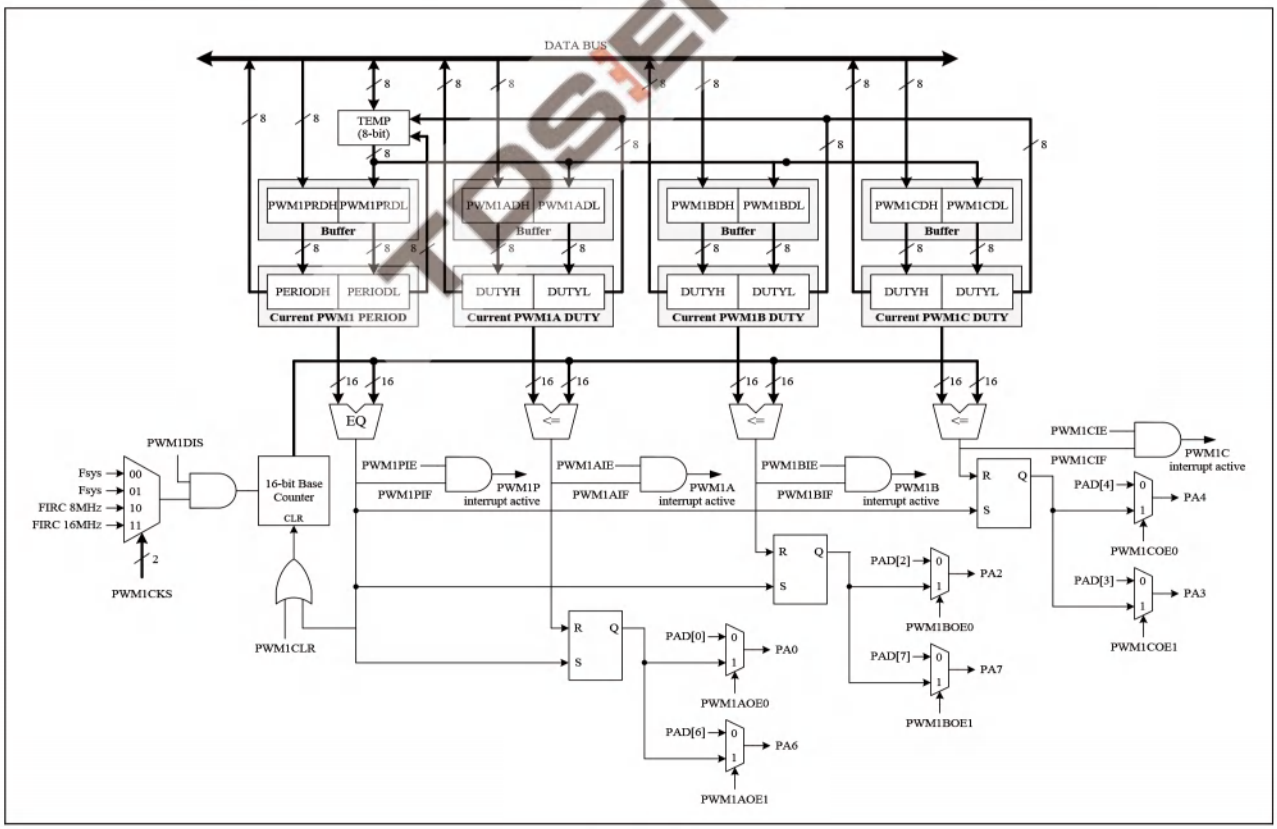
6.6 PWM1A / PWM1B / PWM1C: 16 位 PWMs

PWM1A，PWM1B 和 PWM1C 是 3 个 PWM，它们具有独立的占空比和共享周期。PWM1 可以基于 PWM1 时钟生成占空比分辨率为 65536 的可变频率波形。PWM 时钟可以通过 PWMCKS（97h.3～2）位决定选择 Fsys，FIRC 8 MHz 或 16 MHz。通过将 PWM1DIS（97h.5）位置 1，可以停止 PWM1 时钟。**复位后 PWM1 时钟是使能的而且 PWM1 不会保持固定。（参阅 PWM1DIS 和 PWM1CLR 的描述）**

引脚模式 SFR 控制 PWM 输出波形格式。 模式 1 使 PWM 开漏输出，而模式 2 使 PWM CMOS 推挽输出。（请参阅第 5 节）

16 位 PWM1PRD，PWM1AD，PWM1BD，PWM1CD 寄存器均具有低字节和高字节结构。 高字节可以直接访问，但低字节只能通过内部 8 位缓冲区访问，必须以特定方式对这些寄存器对进行读写。 需要注意的重要一点是，只有在执行对其相应的高字节的写或读操作时，才与 8 位缓冲区及其相关的低字节进行数据传输。 简而言之，**先写低字节，再写高字节。 首先读取高字节，然后读取低字节**

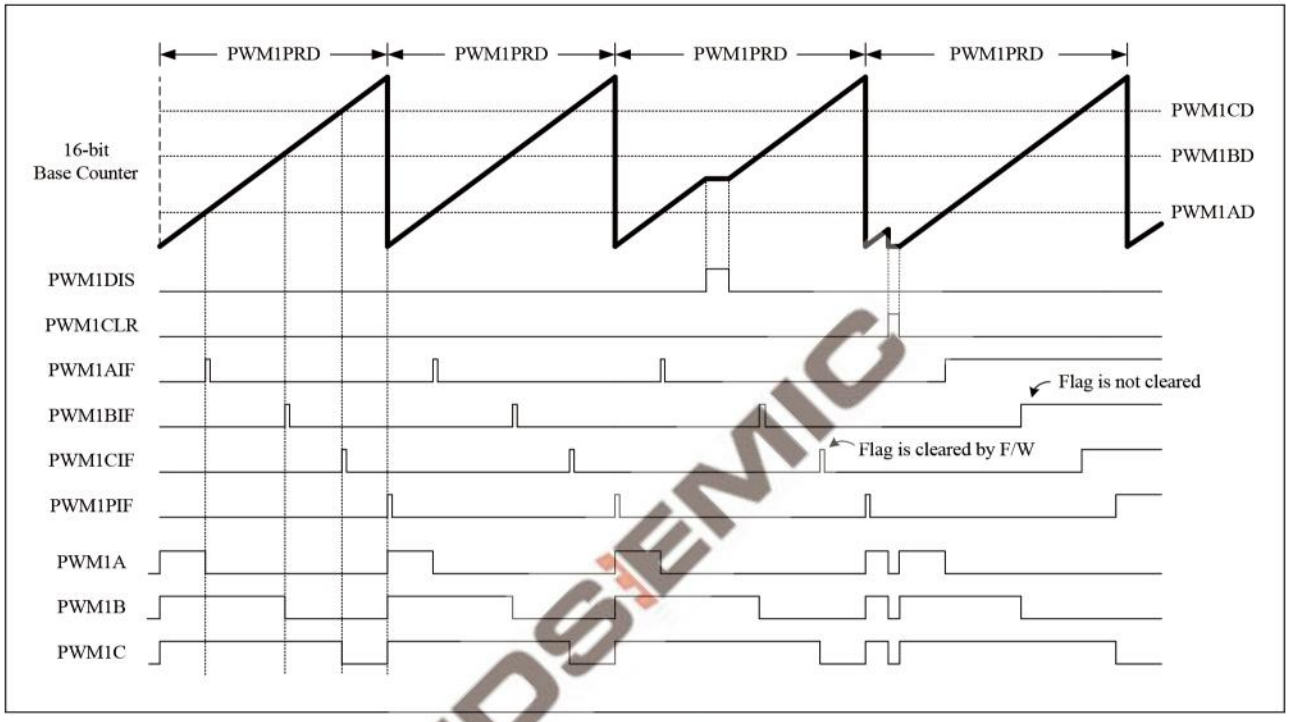
PWM1 的结构如下图所示。当 PWM1CLR（97h.4）位置位时，PWM1 将被清除并保持，否则 PWM1 正在运行。 可以通过写 PWM1DH 和 PWM1DL 来更改 PWM1 占空比。 每当 16 位基数计数器与 16 位 PWM1 占空比寄存器{PWM1DH，PWM1DL}匹配时，PWM1 输出信号就会复位为低电平。 可以通过将周期值写入 PWM1PRDH 和 PWM1PRDL 寄存器来设置 PWM1 周期。写入 PWM1D 或 PWM1PRD 寄存器后，新值将立即保存到其自己的缓冲区中。在当前周期结束时或清除 PWM1 时由 H/W 更新这些值。



PWM1 框图

PWM1A，PWM1B 和 PWM1C 具有相应的中断标志 PWM1AIF（0Eh.0），PWM1BIF（0Eh.1）和 PWM1CIF（0Eh.2），这些中断标志是在 PWM1 16 位基础计数器计数到设置的占空比时产生的。PWM1 还具有相应的周期中断标志 PWM1PIF（0Eh.3），并且在周期结束时产生一个中断标志。 将它们的相应中断允许位 PWM1AIE（0Dh.0），PWM1BIE（0Dh.1），PWM1CIE（0Dh.2）和 PWM1PIE（0Dh.3）置 1 可以产生相应的中断。

PWM1xOEn（91h.7～2）位用于选择相关的 PWM 输出至 I / O。无论是否设置了 PWMOE，PWM1 都可以在后台作为定时器运行。



PWM1 时序图

0Dh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE1	-	-	-	-	PWM1PIE	PWM1CIE	PWM1BIE	PWM1AIE
R/W	-	-	-	-	R/W	R/W	R/W	R/W
复位	-	-	-	-	0	0	0	0

0Dh.3 **PWM1PIE:** PWM1周期中断使能
0：禁止
1：使能

0Dh.2 **PWM1CIE:** PWM1C占空比中断使能
0：禁止
1：使能

0Dh.1 **PWM1BIE:** PWM1B占空比中断使能
0：禁止
1：使能

0Dh.0 **PWM1AIE:** PWM1A占空比中断使能
0：禁止
1：使能

0Eh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIF1	-	-	-	-	PWM1PIF	PWM1CIF	PWM1BIF	PWM1AIF
R/W	-	-	-	-	R/W	R/W	R/W	R/W
复位	-	-	-	-	0	0	0	0

0Eh.3 **PWM1PIF:** PWM1 period中断事件挂起标志
在PWM1计数器计数到设置的周期后由H/W置位，对此位写0将清除该标志
0Eh.2 **PWM1CIF:** PWM1C duty中断事件挂起标志
在PWM1计数器计数到设置的PWM1C占空比后由H/W置位，对此位写0将清除该标志
0Eh.1 **PWM1BIF:** PWM1B duty中断事件挂起标志
在PWM1计数器计数到设置的PWM1B占空比后由H/W置位，对此位写0将清除该标志
0Eh.0 **PWM1AIF:** PWM1A duty中断事件挂起标志
在PWM1计数器计数到设置的PWM1A占空比后由H/W置位，对此位写0将清除该标志

91h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWMOE	PWM1COE1	PWM1COE0	PWM1BOE1	PWM1BOE0	PWM1AOE1	PWM1AOE0	PWM0NOE	PWM0POE
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

91h.7 **PWM1COE1:** PWM1C 输出至 PA3 使能
0: 禁止 1: 使能, PWM1C 输出至PA3
91h.6 **PWM1COE0:** PWM1C 输出至 PA4 使能
0: 禁止 1: 使能, PWM1C 输出至PA4
91h.5 **PWM1BOE1:** PWM1B 输出至 PA7 使能
0: 禁止 1: 使能, PWM1B 输出至PA7
91h.4 **PWM1BOE0:** PWM1B 输出至 PA2 使能
0: 禁止 1: 使能, PWM1B 输出至PA2
91h.3 **PWM1AOE1:** PWM1A 输出至 PA6 使能
0: 禁止 1: 使能, PWM1A 输出至PA6
91h.2 **PWM1AOE0:** PWM1A 输出至 PA0 使能
0: 禁止 1: 使能, PWM1A 输出至PA0

97h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM1CTL	-	-	PWM1DIS	PWM1CLR	PWM1CKS		-	-
R/W	-	-	R/W	R/W	R/W	R/W	-	-
复位	-	-	0	0	0	0	-	-

97h.5 **PWM1DIS:** PWM1 时钟禁止
0: 时钟使能
1: 时钟禁止
97h.4 **PWM1CLR:** PWM1 清除并保持
0: PWM1 使能
1: PWM1 清除并保持
97h.3~2 **PWM1CKS:** PWM1 时钟源选择 t
0x: Fsys
10:FIRC 8MHz
11:FIRC 16MHz

98h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM1PRDH	PWM1PRDH							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	1	1	1	1	1	1	1	1

98h.7~0 **PWM1PRDH:** PWM1（PWM1A / PWM1B / PWM1C）周期数据8位MSB

99h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM1PRDL	PWM1PRDL							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	1	1	1	1	1	1	1	1

99h.7~0 **PWM1PRDL:** PWM1（PWM1A / PWM1B / PWM1C）周期数据8位LSB
大约16位数据写入：首先写入PWM1PRDL，然后写入PWM1PRDH
大约读取16位数据：首先读取PWM1PRDH，然后读取PWM1PRDL

9Ah	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM1ADH	PWM1ADH							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	1	0	0	0	0	0	0	0

9Ah.7~0 **PWM1ADH:** PWM1A占空比8位MSB

9Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM1ADL	PWM1ADL							
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

9Bh.7~0 **PWM1ADL:** PWM1A占空比8位LSB
大约16位数据写入：首先写入PWM1ADL，然后写入PWM1ADH
大约读取16位数据：首先读取PWM1ADH，然后读取PWM1ADL

9Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM1BDH	PWM1BDH							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	1	0	0	0	0	0	0	0

9Ch.7~0 **PWM1BDH:** PWM1B占空比8位MSB

9Dh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM1BDL	PWM1BDL							
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

9Dh.7~0 **PWM1BDL:** PWM1B占空比8位LSB
大约16位数据写入：首先写入PWM1BDL，然后写入PWM1BDH
大约读取16位数据：首先读取PWM1BDH，然后读取PWM1BDL

9Eh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM1CDH	PWM1CDH							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

复位	1	0	0	0	0	0	0	0

9Eh.7~0 **PWM1CDH:** PWM1C占空比8位MSB

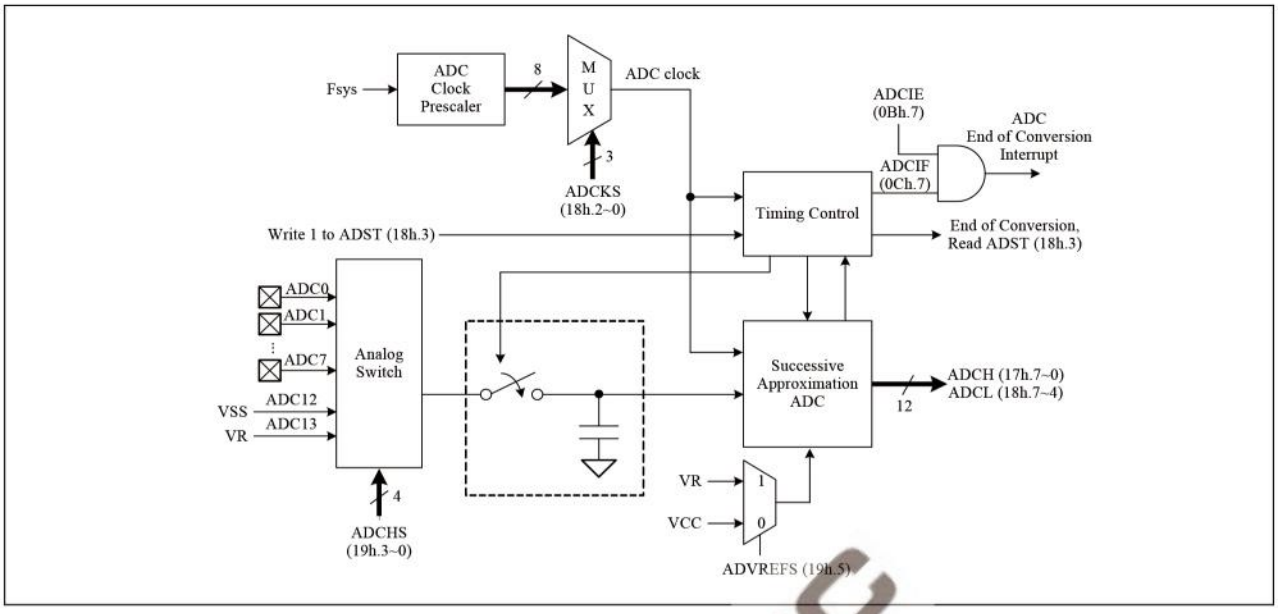
9Fh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM1CDL	PWM1CDL							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	0	0	0	0	0	0

9Fh.7~0 **PWM1CDL:** PWM1C占空比8位LSBt

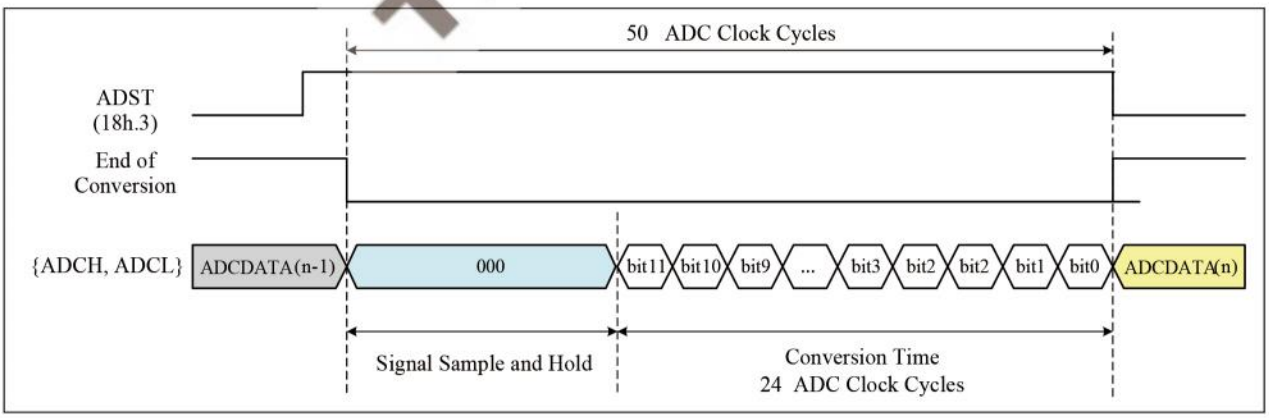
大约16位数据写入：首先写入PWM1CDL，然后写入PWM1CDH

大约读取16位数据：首先读取PWM1CDH，然后读取PWM1CDL

6.7 模数转换器



12 位 ADC（模数转换器）由一个 16 通道模拟输入多路复用器，控制寄存器，时钟发生器，12 位逐次逼近寄存器和输出数据寄存器组成。要使用 ADC，用户需要设置 ADCKS（18h.2～0）来选择合适的 ADC 时钟频率，该频率必须小于 1 MHz。然后，用户通过将 ADST（18h.3）控制位置 1 来启动 ADC 转换。转换结束后，H/W 将自动清除 ADST（18h.3）位。用户可以查询该位以了解转换状态。 PAMODH 和 PAMODL 控制寄存器用于 ADC 引脚配置，当该引脚用作 ADC 输入时，用户必须将引脚设置为模式 3。该设置可以禁止引脚逻辑输入路径以节省功耗。用户需要设置 ADCHS（19h.3～0）来选择 ADC 的输入通道。此外，可以选择一些参考输入通道，ADC12 为 VSS，ADC13 为 VR = 3V。ADC 参考电压可以通过 ADCVREFS（19h.5～4）选择 VCC 或 VR。



说明:

[CPU 运行在快速模式 , Fsys = FIRC 8MHz]
ADC 时钟频率 = 1 MHz, ADC 通道 = ADC2 (PA2).

◇ 范例:

```
MOVLW     00000111B          ; Fsys = 8 MHz
MOVWX     CLKCTL             ;

MOVLW     01110101B          ; ADC2(PA2)引脚模式 = 3 = ADC输入
MOVWX     PAMODL;

MOVLW     00000101B          ; 18h.2~0 (ADCKS) = Fsys/8, ADC clock =
                             1MHz
MOVWX     ADCTL

MOVLW     00110010B          ; 19h.5 = 0, ADC参考电压选择VCC
MOVWX     MF019              ; 19h.3~0 = 2,ADC输入通道选择 ADC2

BSX       ADST               ; 18h.3 (ADST), ADC开始转换

WAIT_ADC:
BTXSC     ADST               ; 等待ADC转换完成.
GOTO      WAIT_ADC

MOVXW     ADCH               ; 17h.7～0，将ADC结果[11：4]读入W
MOVXW     ADCTL              ; 18h.7～4，将ADC结果[3：0]读入W
...
```

0Bh/8Bh/10Bh/18Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE	ADCIE	T2IE	TM1IE	TM0IE	WKTIE	INT2IE	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

INTIE.7 **ADCIE:** ADC中断使能
0: 禁止
1: 使能

0Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIF	ADCIF	T2IF	TM1IF	TM0IF	WKTIF	INT2IF	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

0Ch.7 **ADCIF:** ADC中断事件挂起标志
ADC转换结束后由H/W置位，对该位置写0将清除该标志

17h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCH	ADCH							
R/W	R	R	R	R	R	R	R	R

复位	-	-	-	-	-	-	-	-

17h.7~0 **ADCH:** ADC输出数据MSB，ADQ［11：4］

18h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
ADCTL	ADCL				ADST	ADCKS		
R/W	R	R	R	R	R/W	R/W	R/W	R/W
复位	-	-	-	-	0	0	0	0

18h.7~4 **ADCL:** ADC输出数据LSB, ADQ［3:0］

18h.3 **ADST:** ADC 始起位
0：转换结束后由 H/W 清除
1：ADC 开始转换

18h.2~0 **ADCKS:** ADC 时钟频率选择：
000：Fsys/256　　100：Fsys/16
001：Fsys/128　　101：Fsys/8
010：Fsys/64　　110：Fsys/4
011：Fsys/32　　111：Fsys/2

19h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MF019	-	-	ADVREFS	VREN	ADCHS			
R/W	-	-	R/W	R/W	R/W			
复位	-	-	1	1	0	0	0	0

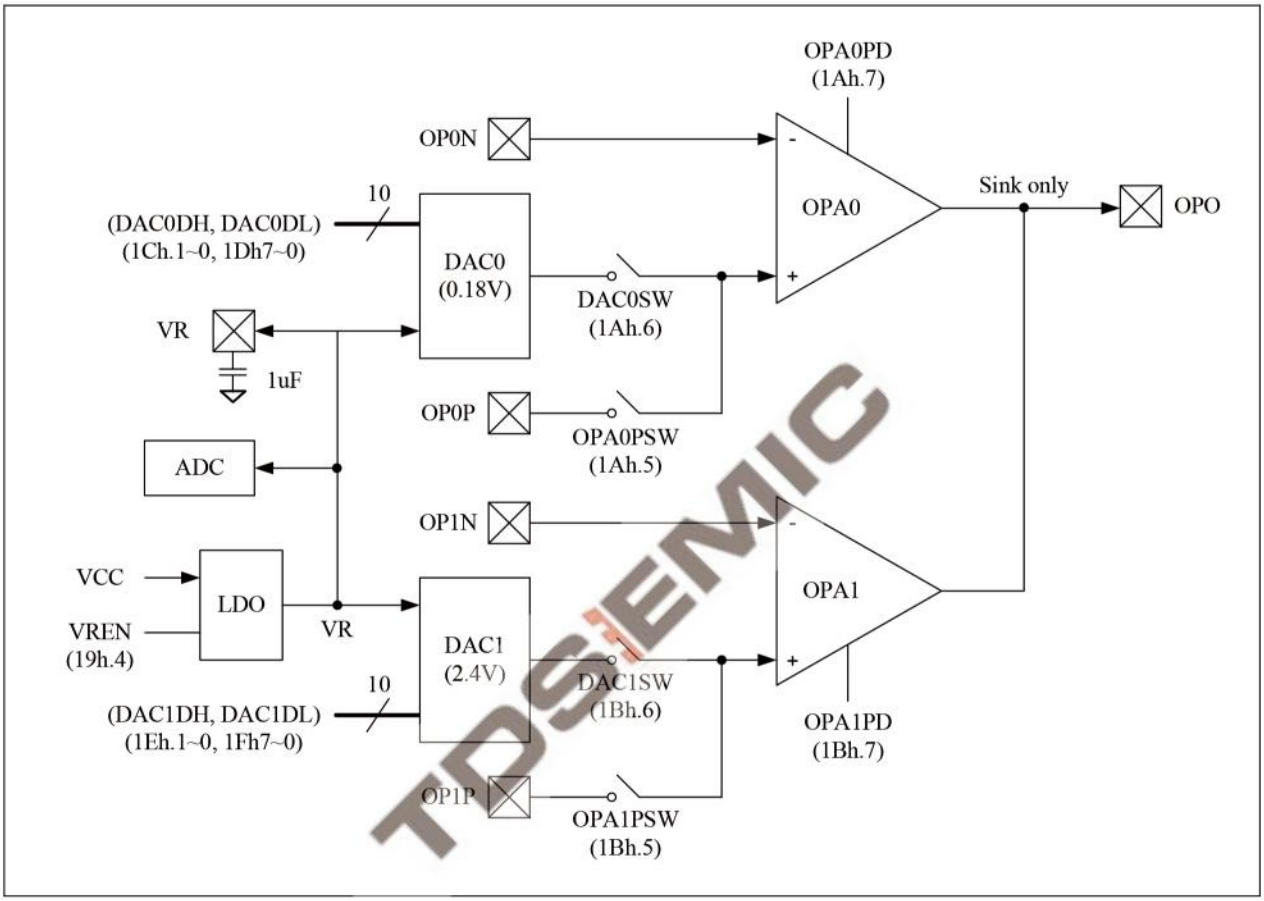
19h.5 **ADVREFS:** ADC 参考电压选择
0：VCC
1：VR

19h.4 **VREN:** 内部参考电压 VR 使能（3V ±1.2% @25℃, VCC=3V~5V）
0：禁止
1：在 STOP / IDLE 模式下使能和自动禁止

19h.3~0 **ADCHS:** ADC 通道选择
0000：ADC0（PA0）　　1000：预留
0001：ADC1（PA1）　　1001：预留
0010：ADC2（PA2）　　1010：预留
0011：ADC3（PA3）　　1011：预留
0100：ADC4（PA4）　　1100：VSS
0101：ADC5（PA5）　　1101：VR
0110：ADC6（PA6）　　1110：预留
0111：ADC7（PA7）　　1111：预留

6.8 电池充电模块(BCM)

TM56F8225 包含一个电池充电模块（BCM）。该模块由两个 10 位数模转换器（DAC）和两个运算放大器（OPA）组成。DAC 参考电压来自内部参考电压 VR（VR = 3V±1.2%@ 25°C，VCC = 3V～5V）。可以通过设置 VREN = 0（19h.4）来关闭 VR 的电压。上电后，DAC0 的默认电压输出约为 0.18V，而 DAC1 约为 2.4V。它们分别连接到 OPA0 和 OPA1 的正输入。此外，上电后 OPA0 和 OPA1 始终处于使能状态。电池充电模块如下图所示。



BCM 框图

10 位 DAC0D 和 DAC1D 寄存器均具有低字节和高字节结构。 在读取操作中，高字节和低字节可以直接访问。 在写操作中，高字节可以直接访问，但低字节只能通过内部 8 位缓冲区访问。 写入这些寄存器对必须以特定方式进行。 需要注意的重要一点是，只有在执行对其相应的高字节的写操作时，才与 8 位缓冲区及其相关的低字节进行数据传输。简而言之，**先写低字节，再写高字节。 直接读取高字节和低字节。**

1Ah	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPA0CTL	OPA0PD	DAC0SW	OPA0PSW	OPA0ADJ				
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	1	1	by CFG				

1Ah.7 **OPA0PD:** OPA0 掉电
0: OPA0开
1: OPA0 关

1Ah.6 **DAC0SW:** DAC0 至 OPA0 同相输入打开
0: 关闭
1: 开启

1Ah.5 **OPA0PSW:** OPA0P 至 OPA0 同相输入打开
0: 关闭
1: 开启

1Ah.4~0 **OPA0ADJ:** OPA0 输入失调电压校准控制

1Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPA1CTL	OPA1PD	DAC1SW	OPA1PSW	OPA1ADJ				
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	1	1	by CFG				

1Bh.7 **OPA1PD:** OPA1 掉电
0: OPA1开
1: OPA1关

1Bh.6 **DAC1SW:** DAC1 至 OPA1 同相输入开关打开
0: 关闭
1: 开启

1Bh.5 **OPA1PSW:** OPA1P 至 OPA1 同相输入打开
0: 关闭
1: 开启

1Bh.4~0 **OPA1ADJ:** OPA1 输入失调电压校准控制

1Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DAC0DH	-	-	-	-	-	-	DAC0DH	
R/W	-	-	-	-	-	-	R/W	R/W
复位	-	-	-	-	-	-	0	0

1Ch.1~0 **DAC0DH:** DAC0输入数据MSB[9~8]

1Dh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DAC0DL	DAC0DL							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	1	0	0	0	0	0	0

1Dh.7~0 **DAC0DL:** DAC0输入数据LSB[7~0]
先写DAC0DL，然后写DAC0DH
直接读取DAC0DH和DAC0DL

1Eh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DAC1DH	-	-	-	-	-	-	DAC1DH	
R/W	-	-	-	-	-	-	R/W	R/W
复位	-	-	-	-	-	-	1	1

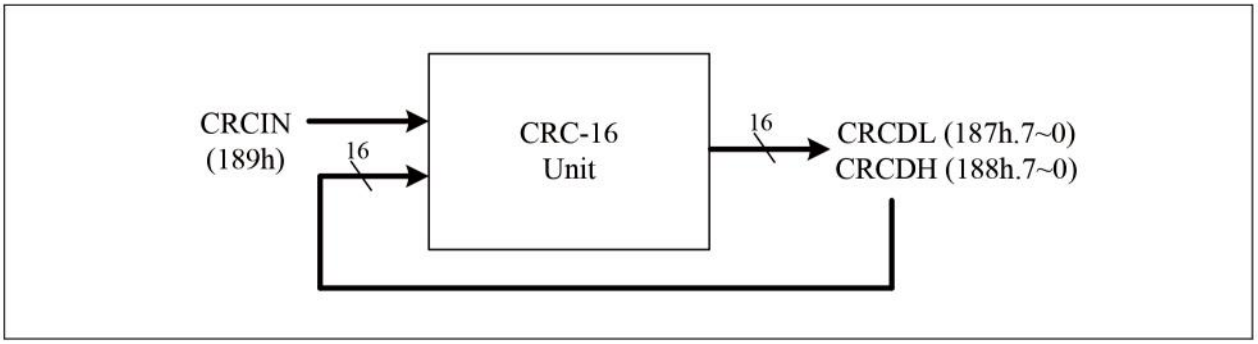
1Eh.1~0 **DAC1DH:** DAC1输入数据MSB[9~8]

1Fh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DAC1DL	DAC1DL							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位	0	0	1	1	1	0	0	1

1Fh.7~0 **DAC1DL:** DAC1输入数据LSB[7~0]
先写DAC0DL，然后写DAC0DH
直接读取DAC0DH和DAC0DL

6.9 循环冗余校验(CRC)

该芯片支持集成的 16 位循环冗余校验功能。 循环冗余校验（CRC）计算单元是一种错误检测技术测试算法，用于验证数据传输或存储数据的正确性。 CRC 计算将 8 位数据流或数据块作为输入，并生成 16 位输出余数。数据流由相同的生成多项式计算。



CRC16 框图

CRC 生成器基于 CRC-16-IBM 多项式提供 16 位 CRC 结果计算。 在此 CRC 生成器中，只有一个多项式可用于数值计算。 它不支持基于任何其他多项式的 16 位 CRC 计算。 对 CRCIN 寄存器的每次写操作都会创建存储在 CRCDH 和 CRCDL 寄存器中的先前 CRC 值的组合。 计算将需要一个 MCU 指令周期。

CRC-16-IBM（Modbus）多项式表示： $X^{16}+X^{15}+X^{2}+1$

187h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CRCDL	CRCDL							
R/W	R/W							
复位	1	1	1	1	1	1	1	1

187h.7~0 **CRCDL:** 16 位 CRC checksum 数据位 7~0

188h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CRCDH	CRCDH							
R/W	R/W							
复位	1	1	1	1	1	1	1	1

188h.7~0 **CRCDL:** 16-位 CRC checksum 数据位 15~8

189h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CRCIN	CRCIN							
W	W							
复位	-	-	-	-	-	-	-	-

189h.7~0 **CRCIN:** 写入该寄存器以开始 CRC 计算

存储器映射

名称	地址	读/写	复位	描述
INDF (00h/80h/100h/180h)				**相关功能：RAM W/R**
INDF	00.7~0	R/W	-	不是物理寄存器，寻址 INDF 实际上指向其地址包含在 FSR 寄存器中的寄存器
TM0 (01h/101h)				**相关功能：Timer0**
TM0	01.7~0	R/W	0	Timer0 内容
PCL (02h/82h/105h/182h)				**相关功能：PROGRAM COUNT**
PCL	02.7~0	R/W	0	程序计数器 LSB [7~0]
STATUS (03h/83h/103h/183h)				**相关功能：STATUS**
IRP	03.7	R/W	0	Bank 寄存器选择位（用于间接寻址）
RP1	03.6	R/W	0	Bank 1 寄存器选择位（汇编保持该位为 0）
RP0	03.5	R/W	0	Bank0 寄存器选择位(汇编保持该位为 0)
TO	03.4	R	0	WDT 超时标志，通过 PWRST，“ SLEEP”或“ CLRWDT”指令清除
PD	03.3	R	0	掉电标志，通过指令 ‘SLEEP’设置，通过指令 ‘CLRWDT’ 清除
Z	03.2	R/W	0	零标志
DC	03.1	R/W	0	十进制进位标志
C	03.0	R/W	0	进位标志
FSR (04h/84h/104h/184h)				**相关功能：RAM W/R**
FSR	04.7~0	R/W	-	文件选择寄存器，间接地址模式指针
PAD (05h)				**相关功能：Port A**
PAD	05.7~0	R	-	端口 A 引脚或“数据寄存器”状态
		W	FF	端口 A 输出数据寄存器
PCLATH (0Ah/8Ah/10Ah/18Ah)				**相关功能：PROGRAM COUNT**
PCLATH	0A.2~0	R/W	0	程序计数器高 3 位的写缓冲区
INTIE (0Bh/8Bh/10Bh/18Bh)				**相关功能：Interrupt 使能**
ADCIE	0B.7	R/W	0	ADC 中断使能 0：禁止 1：使能
T2IE	0B.6	R/W	0	T2 中断使能 0：禁止 1：使能
TM1IE	0B.5	R/W	0	Timer1 中断使能 0：禁止 1：使能
TM0IE	0B.4	R/W	0	Timer0 中断使能 0：禁止 1：使能
WKTIE	0B.3	R/W	0	WKT 中断使能,设置为 0 清除和禁止 WKT 计时器 0：禁止 1：使能
INT2IE	0B.2	R/W	0	INT2 引脚(PA7)中断使能 0：禁止 1：使能
INT1IE	0B.1	R/W	0	INT1 引脚(PA1)中断使能 0：禁止 1：使能
INT0IE	0B.0	R/W	0	INT0 引脚(PA6 或 PA2)中断使能 0：禁止 1：使能

INTIF (0Ch)				相关功能: Interrupt Flag
ADCIF	0C.7	R	-	DC 中断标志，在 ADC 转换完成后由 H/W 置位
		W	0	写 0：清除该标志； 写 1：无动作
T2IF	0C.6	R	-	T2 中断事件挂起标志，当 T2 溢出时由 H/W 置位
		W	0	写 0：清除该标志； 写 1：无动作
TM1IF	0C.5	R	-	Timer1 中断事件挂起标志，当 Timer1 溢出时由 H/W 置位 s
		W	0	写 0：清除该标志； 写 1：无动作
TM0IF	0C.4	R	-	Timer0 中断事件挂起标志，当 Timer0 溢出时由 H/W 置位 s
		W	0	写 0：清除该标志； 写 1：无动作
WKTIF	0C.3	R	-	WKT 中断事件挂起标志，当 WKT 超时时由 H/W 置位
		W	0	写 0：清除该标志； 写 1：无动作
INT2IF	0C.2	R	-	IINT2 (PA7) 中断事件挂起标志，当 INT2 引脚发生下降沿时由 H/W 置位
		W	0	写 0：清除该标志； 写 1：无动作
INT1IF	0C.1	R	-	NT1 (PA4)中断事件挂起标志，当 INT1 引脚发生下降沿/上升沿时由 H/W 置位
		W	0	写 0：清除该标志； 写 1：无动作
INT0IF	0C.0	R	-	NT0 (PA6)中断事件挂起标志，当 INT0 引脚发生下降沿/上升沿时由 H/W 置位
		W	0	写 0：清除该标志； 写 1：无动作
INTIE1 (0Dh)				相关功能:中断使能
PWM1PIE	0D.3	R/W	0	PWM1 周期中断使能 0: 禁止 1: 使能
PWM1CIE	0D.2	R/W	0	PWM1C 占空比中断使能 0: 禁止 1: 使能
PWM1BIE	0D.1	R/W	0	PWM1B 占空比中断使能 0: 禁止 1: 使能
PWM1AIE	0D.0	R/W	0	PWM1A 占空比中断使能 0: 禁止 1: 使能
INTIF1 (0Eh)				相关功能: Interrupt Flag
PWM1PIF	0E.3	R	-	PWM1 周期中断世界挂起标志,在 PWM1 计数器计数到设置的周期后由 H/W 置位
		W	0	写 0：清除该标志； 写 1：无动作
PWM1CIF	0E.2	R	-	PWM1C 占空比中断事件挂起标志，在 PWM1 计数器计数到设置的 PWM1C 占空比后由 H/W 置位
		W	0	写 0：清除该标志； 写 1：无动作
PWM1BIF	0E.1	R	-	PWM1B 占空比中断事件挂起标志，在 PWM1 计数器计数到设置的 PWM1B 占空比后由 H/W 置位
		W	0	写 0：清除该标志； 写 1：无动作
PWM1AIF	0E.0	R	-	PWM1A 占空比中断事件挂起标志，在 PWM1 计数器计数到设置的 PWM1A 占空比后由 H/W 置位
		W	0	写 0：清除该标志； 写 1：无动作
CLKCTL (0Fh)				相关功能: Fsys
SLOWSTP	0F.4	R/W	0	在停止模式下停止慢速时钟 0: 不停止 1: 停止
FASTSTP	0F.3	R/W	1	停止快速时钟 0: 不停止 1: 停止
CPUCKS	0F.2	R/W	0	选择系统时钟 0: Fsys=慢速时钟 1: Fsys=快速时钟
CPUPSC	0F.1~0	R/W	11	Fsys 预分频器， 00: 除以 8 01: 除以 4 10: 除以 2 11: 除以 1

TM0RLD (10h)				相关功能：TM0
TM0RLD	10.7~0	R/W	0	Timer0 重载数据
TM0CTL (11h)				**相关功能：TM0**
TM0EDG	11.5	R/W	0	TM0CKI 引脚的 Timer0 预分频器计数沿 0：上升沿 1：下降沿
TM0CKS	11.4	R/W	0	Timer0 预分频器时钟源 0：Fsys/2 1：TM0CKI 引脚(PA6 引脚)
TM0PSC	11.3~0	R/W	0	Timer0 预分频器。 Timer0 预分频器时钟源除以 0000：/1 0100：/16 1000：/256 1100：/4096 0001：/2 0101：/32 1001：/512 1101：/8192 0010：/4 0110：/64 1010：/1024 1110：/16384 0011：/8 0111：/128 1011：/2048 1111：/32768
TM1 (12h)				**相关功能：Timer1**
TM1	12.7~0	R/W	0	Timer1 内容
TM1RLD (13h)				**相关功能：Timer1**
TM1RLD	13.7~0	R/W	0	Timer1 重载数据
TM1CTL (14h)				**相关功能：Timer1**
TM1PSC	14.3~0	R/W	0	Timer1 预分频器。 Timer1 时钟源 0000：Fsys/2 0100：Fsys/32 1xxx：Fsys/512 0001：Fsys/4 0101：Fsys/64 0010：Fsys/8 0110：Fsys/128 0011：Fsys/16 0111：Fsys/256
T2CTL (15h)				**相关功能：T2**
T2CKS	15.2	R/W	0	T2 时钟源 0：慢速时钟 1：Fsys/128
T2PSC	15.1~0	R/W	0	T2 预分频器。 T2 时钟源除以- 00：32768 01：16384 10：8192 11：128
MF016 (16h)				**相关功能：T2/TM1/TM0/LVR/LVD**
LVDF	16.7	R	-	低压检测标志，当 VCC≤LVD 时由 H/W 置位
LVDEN	16.6	R/W	0	使能低压检测功能（仅当 LVR = 2.2V 时） 0：禁止 1：使能
T2CLR	16.5	R/W	1	T2 计数器清除 0：计数 1:计数停止
TM1STP	16.4	R/W	0	Timer1 计数器停止 0：计数 1：计数停止
TM0STP	16.3	R/W	0	Timer0 计数器停止 0：计数 1：计数停止
LVRSAV	16.2	R/W	1	LVR / LVD 省电 0：在 STOP / IDLE 模式下使能 LVR / LVD 1：在 STOP / IDLE 模式下 LVR / LVD 自动关闭电源
LVDS	16.1~0	R/W	01	LVD 选择(当 LVR=2.2V) 00：3.6V 01：2.8V 1x：4.2V
ADCH (17h)				**相关功能：ADC**
ADCH	17.7~0	R	-	ADC 输出数据 MSB，ADQ [11：4]

ADCTL (18h)				相关功能：ADC
ADCL	18.7~4	R	-	ADC 输出数据 LSB，ADQ [3:0]
ADST	18.3	R/W	0	ADC 始起位 0：转换完成后由 H/W 清除 1：ADC 开始转换
ADCKS	18.2~0	R/W	0	ADC 时钟频率选择： 000：Fsys/256　100：Fsys/16 001：Fsys/128　101：Fsys/8 010：Fsys/64　110：Fsys/4 011：Fsys/32　111：Fsys/2
MF019 (19h)				相关功能：ADC
ADVREFS	19.5	R/W	1	ADC 参考电压选择 0：VCC　1：VR
VREN	19.4	R/W	1	内部参考电压 VR 使能（3V ±1.2% @25°C，VCC=3V~5V） 0：禁止 1：在 STOP / IDLE 模式下使能和自动禁止
ADCHS	19.3~0	R/W	0	ADC 通道选择 0000：ADC0 (PA0)　0110：ADC6 (PA6)　1100：VSS 0001：ADC1 (PA1)　0111：ADC7 (PA7)　1101：VR 0010：ADC2 (PA2)　1000：预留　1110：预留 0011：ADC3 (PB3)　1001：预留　1111：预留 0100：ADC4 (PA4)　1010：预留 0101：ADC5 (PA5)　1011：预留
OPA0CTL (1Ah)				相关功能：OPA
OPA0PD	1A.7	R/W	0	OPA0 掉电 0：OPA0 开启　1：OPA0 关闭
DAC0SW	1A.6	R/W	1	DAC0 至 OPA0 同相输入打开 0：关闭 1：开启
OPA0PSW	1A.5	R/W	1	OPA0P 至 OPA0 同相输入 打开 0：关闭 1：开启
OPA0ADJ	1A.4~0	R/W	CFG	OPA0 输入失调电压校准控制
OPA1CTL (1Bh)				相关功能：OPA
OPA1PD	1B.7	R/W	0	OPA1 掉电 0：OPA1 开启　1：OPA1 关闭
DAC1SW	1B.6	R/W	1	DAC1 至 OPA1 同相输入开关打开 0：关闭 1：开启
OPA1PSW	1B.5	R/W	1	OPA1P 至 OPA1 同相输入打开 0：关闭 1：开启
OPA1ADJ	1B.4~0	R/W	CFG	OPA1 输入失调电压校准控制
DAC0DH (1Ch)				相关功能：DAC
DAC0DH	1C.1~0	R/W	00	DAC0 输入数据 MSB[9~8]
DAC0DL (1Dh)				相关功能：DAC
DAC0DL	1D.7~0	R/W	40	DAC0 输入数据 LSB [7～0]，先写 DAC0DL，然后写 DAC0DH
DAC1DH (1Eh)				相关功能：DAC
DAC1DH	1E.1~0	R/W	03	DAC1 输入数据 MSB[9~8]
DAC1DL (1Fh)				相关功能：DAC
DAC1DL	1F.7~0	R/W	39	DAC1 输入数据 LSB [7～0]，先写 DAC1DL，再写 DAC1DH
用户数据存储器				
RAM	20~6F	R/W	-	RAM Bank0 区（80 字节）
RAM	70~7F	R/W	-	RAM 公共区域（16 字节）
OPTION (81h/181h)				相关功能：STATUS/INT0/INT1/WDT/WKT

HWAUTO	81.7	R/W	0	进入中断向量，硬件自动保存/恢复 WREG 和 STATUS 寄存器，除 TO 和 PD 标志外 0:禁止 1: 使能
INT0EDG	81.6	R/W	0	INT0 引脚边沿中断事件 0: 下降沿触发 1: 上升沿触发
INT1EDG	81.5	R/W	0	INT1 引脚边沿中断事件 0: 下降沿触发 1: 上升沿触发
INT0SEL	81.4	R/W	0	INT0 脚选择 0: PA6 1: PA2
WDTPSC	81.3~2	R/W	11	WDT 预分频选择 00: 128mS 01: 256mS 10: 1024mS 11: 2048mS
WKTPSC	81.1~0	R/W	11	WKT 预分频选择 00: 16mS 01: 32mS 10: 64mS 11: 128mS
PAMODH (8Ch)				**相关功能: Port A**
PA7MOD	8C.7~6	R/W	00	PA7~PA4 I/O 模式控制 00: 模式 0 01: 模式 1 10: 模式 2 11: 模式 3
PA6MOD	8C.5~4	R/W	01	
PA5MOD	8C.3~2	R/W	01	
PA4MOD	8C.1~0	R/W	01	
PAMODL (8Dh)				**相关功能: Port A**
PA3MOD	8D.7~6	R/W	01	PA3~PA0 I/O 模式控制 00: 模式 0 01: 模式 1 10: 模式 2 11: 模式 3
PA2MOD	8D.5~4	R/W	01	
PA1MOD	8D.3~2	R/W	01	
PA0MOD	8D.1~0	R/W	01	
PWMOE (91h)				**相关功能: PWM0 / PWM1A / PWM1B / PWM1C**
PWM1COE1	91.7	R/W	0	PWM1C 输出至 PA3 使能 0: 禁止 1:使能
PWM1COE0	91.6	R/W	0	PWM1C 输出至 PA4 使能 0: 禁止 1:使能
PWM1BOE1	91.5	R/W	0	PWM1B 输出至 PA7 使能 0: 禁止 1:使能
PWM1BOE0	91.4	R/W	0	PWM1B 输出至 PA2 使能 0: 禁止 1:使能
PWM1AOE1	91.3	R/W	0	PWM1A 输出至 PA6 使能 0: 禁止 1:使能
PWM1AOE0	91.2	R/W	0	PWM1A 输出至 PA0 使能 0: 禁止 1:使能
PWM0NOE	91.1	R/W	0	PWM0N 输出至 PA1 使能 0: 禁止 1:使能
PWM0POE	91.0	R/W	0	PWM0P 输出至 PA5 使能 0: 禁止 1:使能
PWM0PRD (92h)				**相关功能: PWM0**
PWM0PRD	92.7~0	R/W	FF	PWM0 周期数据
PWM0DH (93h)				**相关功能: PWM0**
PWM0DH	93.7~0	R/W	00	PWM0 的 8 位占空比 MSB
PWM0DL (94h)				**相关功能: PWM0**

PWM0DL	94.1~0	R/W	0	PWM0 的 2 位占空比 LSB
PWM0CTL (95h)				**相关功能：PWM0**
PWM0DIS	95.5	R/W	0	PWM0 时钟禁止 0：时钟使能 1：时钟禁止
PWM0CLR	95.4	R/W	0	PWM0 清除并保持 0：PWM0 使能 1：PWM0 清除并保持
PWM0CKS	95.3~2	R/W	0	PWM0 时钟源选择 0x：Fsys 10:FIRC 8MHz 11:FIRC 16MHz
PWM0PSC	95.1~0	R/W	0	PWM0 时钟源预分频器 00：除以 1　01：除以 2　10：除以 4　11：除以 8
PWM0CTL1 (96h)				**相关功能：PWM0**
PWM0MODE	96.4~3	R/W	0	PWM0 不同输出模式 00：模式 0 01：模式 1 10：模式 2 11：模式 3
PWM0NOV	96.2~0	R/W	0	PWM0 非重叠控制 000:原始 PWM0 001：非重叠 4 个 PWM0CLK 010：非重叠 5 个 PWM0CLK 011：非重叠 6 个 PWM0CLK 100：非重叠 7 个 PWM0CLK 101：非重叠 8 个 PWM0CLK
PWM1CTL (97h)				**相关功能：PWM1A / PWM1B / PWM1C**
PWM1DIS	97.5	R/W	0	PWM1（PWM1A/PWM1B/PWM1C)时钟禁止 0:时钟使能 1:时钟禁止
PWM1CLR	97.4	R/W	0	PWM1（PWM1A/PWM1B/PWM1C)清除并保持 0：PWM1 使能 1：PWM1 清除并保持
PWM1CKS	97.3~2	R/W	0	PWM1（PWM1A/PWM1B/PWM1C)时钟源选择 0x：Fsys 10:FIRC 8MHz 11:FIRC 16MHz
-	97.1~0	R/W	0	预留，保持两位为 00
PWM1PRDH (98h)				**相关功能：PWM1A / PWM1B / PWM1C**
PWM1PRDH	98.7~0	R/W	FF	PWM1（PWM1A / PWM1B / PWM1C）周期 8 位数据 MSB
PWM1PRDL (99h)				**相关功能：PWM1A / PWM1B / PWM1C**
PWM1PRDL	99.7~0	R/W	FF	PWM1（PWM1A / PWM1B / PWM1C）周期 8 位数据 LSB 大约 16 位数据写入：首先写入 PWM1PRDL，然后写 PWM1PRDH 大约读取 16 位数据：首先读取 PWM1PRDH，然后读 PWM1PRDL
PWM1ADH (9Ah)				**相关功能：PWM1A**
PWM1ADH	9A.7~0	R/W	80	PWM1A 的 8 位占空比 MSB
PWM1ADL (9Bh)				**相关功能：PWM1A**
PWM1ADL	9B.7~0	R/W	00	PWM1A 的 8 位占空比 LSB 大约 16 位数据写入：首先写入 PWM1ADL，然后写入 PWM1ADH 大约读取 16 位数据：首先读取 PWM1ADH，然后读取 PWM1ADL

PWM1BDH (9Ch)				**相关功能：PWM1B**
PWM1BDH	9C.7~0	R/W	80	PWM1B 占空比 8 位 MSB
PWM1BDL (9Dh)				**相关功能：PWM1B**
PWM1BDL	9D.7~0	R/W	00	PWM1B 占空比 8 位 LSB 大约 16 位数据写入：首先写入 PWM1BDL，然后写入 PWM1BDH 大约读取 16 位数据：首先读取 PWM1BDH，然后读取 PWM1BDL
PWM1CDH (9Eh)				**相关功能：PWM1C**
PWM1CDH	9E.7~0	R/W	80	PWM1C 占空比 8 位 MSB
PWM1CDL (9Fh)				**相关功能：PWM1C**
PWM1CDL	9F.7~0	R/W	00	PWM1C 占空比 8 位 LSB 大约 16 位数据写入：首先写入 PWM1CDL，然后写入 PWM1CDH 大约读取 16 位数据：首先读取 PWM1CDH，然后读取 PWM1CDL
用户数据存储器				
RAM	A0~EF	R/W	-	RAM Bank1 区域（80 字节）
LVRPD (109h)				**相关功能：LVR**
LVRPD	109	W	-	写入 37h 以强制禁用 LVR
BGTRIM (10Eh)				**相关功能：Bandgap**
BGTRIM	10E.3~0	R/W	CFG	带隙电压调整： 0000：最低电压 … 1111：最高电压
IRCF (10Fh)				**相关功能：Internal RC**
IRCF	10F.6~0	R/W	CFG	FIRC 频率调整： 00h：最低频率 … 7Fh：最高频率
DPL (185h)				**相关功能：Table Read**
DPL	185.7~0	R/W	0	读表取低位地址，数据 ROM 指针（DPTR）低字节
DPH (186h)				**相关功能：Table Read**
DPH	186.2~0	R/W	0	读表取高位地址，数据 ROM 指针（DPTR）高字节
CRCDL (187h)				**相关功能：CRC16**
CRCDL	187.7~0	R/W	FF	16 位 CRC 的 8 位 LSB 数据
CRCDH (188h)				**相关功能：CRC16**
CRCDH	188.7~0	R/W	FF	16 位 CRC 的 8 位 MSB 数据
CRCIN (189h)				**相关功能：CRC16**
CRCIN	189.7~0	W	-	CRC 数据输入
TABR (18Ch)				**相关功能：Table Read**
TABR	18C.7~0	R/W	0	1. TABR 写 01h =操作码 TABRL 2. TABR 写 02h =操作码 TABRH 3.在步骤 1 或步骤 2 之后，读取 TABR 以获取 Main ROM 读表取值 在步骤 1 之后，读取 TABR 以获取 EEPROM 值（当 EEPEN = E2h 时） ASM 的读表：TABRL / TABRH 或 TABR C 的读表：TABR

EEPCTL （18Dh）				相关功能：EEPROM
EEPTO	18D.2	R/W	0	EEPROM 写超时标志 发生 EEPROM 写超时时由 H/W 置位 当 EEPTE = 0 时由 H/W 清除
EEPTE	18D.1~0	R/W	0	EEPROM 写看门狗定时器使能 （**建议设置 11**） 00：禁止 01：等待 1.6mS 触发看门狗超时标志 10：等待 6.4mS 触发看门狗超时标志 11：等待 12.8mS 触发看门狗超时标志
EEPEN （18Eh）				**相关功能：EEPROM**
EEPEN	18E.7~0	W	-	EEPROM 读/写使能 E2h：使能 EEPROM 读/写 其他：禁止 EEPROM 读/写
EEPDT （18Fh）				**相关功能：EEPROM**
EEPDT	18F.7~0	W	-	EEPROM 写入数据

指令集

每条指令是一个 16 位字，分为一个操作码（用于指定指令类型）和一个或多个操作数（用于进一步指定该指令的操作）。下表中将指令分为面向字节，面向位和面向立即数的操作列表。

对于面向字节的指令，“ f”代表地址指示符，“ d”代表目标指示符。地址指示符用于指定指令将使用程序存储器中的哪个地址。目标指示符指定将操作结果放置在何处。如果“ d”为“ 0”，则结果存入 W 寄存器。如果“ d”为“ 1”，则结果放置在指令指定的地址中。

对于面向位的指令，“ b”代表位字段指示符，它选择受操作影响的位数，而“ f”代表地址指示符。对于立即数运算，“ k”代表立即数或常量值。

简记符号	描述
f	文件寄存器地址
b	位地址
k	立即数.常量数字或标签
d	目标选择字段，0：工作寄存器，1：文件寄存器
W	工作寄存器
Z	零标志
C	进位标志或/借位标志
DC	十进制进位标志或十进制/借位标志
PC	程序计数器
TOS	顶层堆栈
GIE	总中断使能标志（i-Flag）
[]	选择字段
( )	内容
.	位域
B	之前
A	之后
←	分配方向

助记符		操作码	周期	影响标志	描述
面向字节的文件寄存器指令					
ADDWX	f, d	ff00 0111 dfff ffff	1	C, DC, Z	W 和 "f" 相加
ANDWX	f, d	ff00 0101 dfff ffff	1	Z	W 和 "f" 相与
CLRX	f	ff00 0001 1fff ffff	1	Z	清除"f"
CLRW		0000 0001 0100 0000	1	Z	清除 W
COMX	f, d	ff00 1001 dfff ffff	1	Z	"f"取反
DECX	f, d	ff00 0011 dfff ffff	1	Z	“ f” 减 1
DECXSZ	f, d	ff00 1011 dfff ffff	1 or 2	-	"f"减 1，如果为零则跳过
INCX	f, d	ff00 1010 dfff ffff	1	Z	"f"加 1
INCXSZ	f, d	ff00 1111 dfff ffff	1 or 2	-	"f"加 1, 如果为零则跳过
IORWX	f, d	ff00 0100 dfff ffff	1	Z	W 和 “f” 相或
MOVX	f, d	ff00 1000 dfff ffff	1	Z	移 "f"
MOVXW	f	ff00 1000 0fff ffff	1	Z	将 "f"移至 W
MOVWX	f	ff00 0000 1fff ffff	1	-	将 W 移至 "f"
RLX	f, d	ff00 1101 dfff ffff	1	C	“f” 带进位位左移
RRX	f, d	ff00 1100 dfff ffff	1	C	“f” 带进位位右移
SUBWX	f, d	ff00 0010 dfff ffff	1	C, DC, Z	"f"减 W
SWAPX	f, d	ff00 1110 dfff ffff	1	-	"f"的高低半字节互换
TSTX	f	ff00 1000 1fff ffff	1	Z	检测"f" 是否为 0
XORWX	f, d	ff00 0110 dfff ffff	1	Z	W 和 "f"相异或
面向位的文件寄存器指令					
BCX	f, b	ff11 00bb bfff ffff	1	-	"f"的"b"位清零
BSX	f, b	ff11 01bb bfff ffff	1	-	"f"的"b"位置位
BTXSC	f, b	ff11 10bb bfff ffff	1 or 2	-	"f"的"b"位为 0 则跳过
BTXSS	f, b	ff11 11bb bfff ffff	1 or 2	-	"f"的"b"位为 1 则跳过
立即数和控制指令					
ADDLW	k	0001 1100 kkkk kkkk	1	C, DC, Z	立即数 "k" 和 W 相加
ANDLW	k	0001 1011 kkkk kkkk	1	Z	立即数 "k" 和 W 相与
CALL	k	0010 0kkk kkkk kkkk	2	-	调用子程序 "k"
CLRWDT		0001 1110 0000 0100	1	TO, PD	清除看门狗定时器
GOTO	k	0010 1kkk kkkk kkkk	2	-	跳转至分支"k"

IORLW	k	0001 1010 kkkk kkkk	1	Z	立即数"k" 和 W 相或
MOVLW	k	0001 1001 kkkk kkkK	1	-	将立即数 "k" 移至 W
NOP		0000 0000 0000 0000	1	-	空操作指令
RET		0000 0000 0100 0000	2	-	从子程序返回
RETI		0000 0000 0110 0000	2	-	从中断返回
RETLW	k	0001 1000 kkkk kkkk	2	-	带立即数返回，返回值在 W 中
SLEEP		0001 1110 0000 0011	1	TO, PD	进入睡眠模式，时钟振荡停止
SUBLW	k	0001 1111 kkkk kkkk	1	C, DC, Z	W 减去即数"k"
TABRH		0000 0000 0101 1000	2	-	查找 ROM 高数据到 W
TABRL		0000 0000 0101 0000	2	-	查找 ROM 低数据到 W
XORLW	k	0001 1101 kkkk kkkk	1	Z	立即数"k" 和 W 相异或

ADDLW	W和立即数 "k" 相加
语法	ADDLW k
操作数	k : 00h ~ FFh
运作方式	(W) ← (W) + k
影响的状态位	C, DC, Z
操作码	0001 1100 kkkk kkkk
描述	将W寄存器的内容和8位立即数“ k”相加，将结果放入W寄存器中。
周期	1
范例	ADDLW 0x15 B : W =0x10 A : W =0x25

ADDWX	W和 "f"相加
语法	ADDWX f [,d]
操作数	f : 00h ~ 1FFh, d : 0, 1
运作方式	(目标) ← (W) + (f)
影响的状态位	C, DC, Z
操作码	ff00 0111 dfff ffff
描述	将W寄存器的内容和寄存器“ f”相加。 如果d为0，则结果放在W寄存器中。 如果“ d”为1，结果放回寄存器“ f”。
周期	1
范例	ADDWX FSR, 0 B : W =0x17, FSR =0xC2 A : W =0xD9, FSR =0xC2

ANDLW	W和 立即数 "k" 逻辑与
语法	ANDLW k
操作数	k : 00h ~ FFh
运作方式	(W) ← (W) AND k
影响的状态位	Z
操作码	0001 1011 kkkk kkkk
描述	W寄存器的内容与8位立即数“ k”相与。 结果放在W寄存器中。
周期	1
范例	ANDLW 0x5F B : W =0xA3 A : W =0x03

ANDWX	W 和 "f"逻辑与
语法	ANDWX f [,d]
操作数	f : 00h ~ 1FFh, d : 0, 1
运作方式	(目标) ← (W) AND (f)
影响的状态位	Z
操作码	ff00 0101 dfff ffff
描述	W 寄存器的内容和寄存器“ f”相与。 如果 d 为 0，则结果放在 W 寄存器中。 如果“ d”为 1，结果放回寄存器“ f”。
周期	1
范例	ANDWX FSR, 1 B : W =0x17, FSR =0xC2 A : W =0x17, FSR =0x02

BCX	"f"的"b"位清零	
语法	BCX f [,b]	
操作数	f : 00h ~ 1FFh, b : 0 ~ 7	
运作方式	(f.b) ← 0	
影响的状态位	-	
操作码	ff11 00bb bfff ffff	
描述	寄存器“f”中的“ b”位被清零。	
周期	1	
范例	BCX FLAG_REG, 7	B : FLAG_REG =0xC7 A : FLAG_REG =0x47

BSX	"f"的"b"位置位	
语法	BSX f [,b]	
操作数	f : 00h ~ 1FFh, b : 0 ~ 7	
运作方式	(f.b) ← 1	
影响的状态位	-	
操作码	ff11 01bb bfff ffff	
描述	寄存器“f”中的“ b”位被置位。	
周期	1	
范例	BSX FLAG_REG, 7	B : FLAG_REG =0x0A A : FLAG_REG =0x8A

BTXSC	检测"f"的 "b" 位，为 0 则跳过	
语法	BTXSC f [,b]	
操作数	f : 00h ~ 1FFh, b : 0 ~ 7	
运作方式	如果 (f.b) = 0 则跳过下一条指令	
影响的状态位	-	
操作码	ff11 10bb bfff ffff	
描述	如果寄存器“ f”中的位“ b”为 1，则执行下一条指令。 如果寄存器“ f”中的位“ b”为 0，则下一条指令放弃执行，而是执行一条 NOP，使其成为 2 周期指令。	
周期	1 or 2	
范例	LABEL1 BTXSC FLAG, 1 TRUE GOTO SUB1 FALSE ...	B : PC =LABEL1 A : 如果 FLAG.1 =0, PC =FALSE 如果 FLAG.1 =1, PC =TRUE

BTXSS	检测"f"的 "b" 位，为 1 则跳过	
语法	BTXSS f [,b]	
操作数	f : 00h ~ 1FFh, b : 0 ~ 7	
运作方式	如果 (f.b) = 1 则跳过下一条指令	
影响的状态位	-	
操作码	ff11 11bb bfff ffff	
描述	如果寄存器“ f”中的位“ b”为 0，则执行下一条指令。 如果寄存器“ f”中的位“ b”为 1，则下一条指令放弃执行，而是执行一条 NOP 指令，使其成为 2 周期指令。	
周期	1 or 2	
范例	LABEL1 BTXSS FLAG, 1 TRUE GOTO SUB1 FALSE ...	B : PC =LABEL1 A : 如果 FLAG.1 =0, PC =TRUE 如果 FLAG.1 =1, PC =FALSE

CALL	调用子程序"k"
语法	CALL k
操作数	k : 000h ~ FFFh
运作方式	运作: TOS ← (PC) + 1, PC.10~0 ← k
影响的状态位	-
操作码	0010 0kkk kkkk kkkk
描述	调用子程序。首先，将返回地址（PC + 1）压入堆栈。11 位立即数地址被装入 PC 位<10：0>。PC 的高位从 PCLATH 加载。CALL 是两个周期的指令。
周期	2
范例	LABEL1 CALL SUB1 B : PC =LABEL1 A : PC =SUB1, TOS =LABEL1 + 1

CLRX	清除"f"
语法	CLRX f
操作数	f : 00h ~ 1FFh
运作方式	(f) ← 00h, Z ← 1
影响的状态位	Z
操作码	ff00 0001 1fff ffff
描述	清除寄存器“ f”的内容，并将 Z 位置 1。
周期	1
范例	CLRX FLAG_REG B : FLAG_REG =0x5A A : FLAG_REG =0x00, Z =1

CLRW	清除 W
语法	CLRW
操作数	-
运作方式	(W) ← 00h, Z ← 1
影响的状态位	Z
操作码	0000 0001 0100 0000
描述	清除 W 寄存器，并将 Z 位置 1
周期	1
范例	CLRW B : W =0x5A A : W =0x00, Z =1

CLRWDT	清除看门狗定时器
语法	CLRWDT
操作数	-
运作方式	WDT/WKT Timer ← 00h
影响的状态位	TO, PD
操作码	0001 1110 0000 0100
描述	CLRWDT 指令清除看门狗/唤醒定时器
周期	1
范例	CLRWDT B : WDT 计数器 =? A : WDT 计数器 =0x00

COMX	"f"取反
语法	COMX f [,d]
操作数	f : 00h ~ 1FFh, d : 0, 1
运作方式	(目标) ← ($\bar{f}$)
影响的状态位	Z
操作码	ff00 1001 dfff ffff
描述	寄存器“ f”的内容被取反。 如果“ d”为 0，结果放在 W 中。如果“ d”为 1，结果放回寄存器“ f”中。
周期	1
范例	COMX REG1, 0 B : REG1 =0x13 A : REG1 =0x13, W =0xEC

DECX	"f"递减
语法	DECX f [,d]
操作数	f : 00h ~ 1FFh, d : 0, 1
运作方式	(目标) ← (f) - 1
影响的状态位	Z
操作码	ff00 0011 dfff ffff
描述	寄存器“ f”的内容递减。 如果 d 为 0，则结果放在 W 寄存器中。 如果“ d”为 1，结果放回寄存器“ f”。
周期	1
范例	DECX CNT, 1 B : CNT =0x01, Z =0 A : CNT =0x00, Z =1

DECXSZ	"f"递减，如果为 0 则跳过
语法	DECXSZ f [,d]
操作数	f : 00h ~ 1FFh, d : 0, 1
运作方式	(目标) ← (f) - 1, 如果结果为 0，则跳过下一条指令
影响的状态位	-
操作码	ff00 1011 dfff ffff
描述	寄存器“ f”的内容递减。 如果“ d”为 0，结果放入 W 寄存器。 如果“ d”为 1，结果放回寄存器“ f”。 如果结果为 1，则执行下一条指令。 如果结果为 0，则改为执行 NOP，使其成为 2 周期指令。
周期	1 或 2
范例	LABEL1 DECXSZ CNT, 1 GOTO LOOP CONTINUE B : PC =LABEL1 A : CNT =CNT - 1 如果 CNT =0, PC =CONTINUE 如果 CNT ≠0, PC =LABEL1 + 1

GOTO	无条件跳转
语法	GOTO k
操作数	k : 000h ~ FFFh
运作方式	PC.10~0 ← k
影响的状态位	-
操作码	0010 1kkk kkkk kkkk
描述	GOTO 是无条件跳转。 11 位立即数装入 PC 位<10：0>中。PC 高位从 PCLATH 中装入。 GOTO 是两个周期的指令。
周期	2
范例	LABEL1 GOTO SUB1 B : PC =LABEL1 A : PC =SUB1

INCX	"f"递增
语法	INCX f [,d]
操作数	f : 00h ~ 1FFh
运作方式	(目标) ← (f) + 1
影响的状态位	Z
操作码	ff00 1010 dfff ffff
描述	寄存器“ f”的内容递增。 如果“ d”为 0，结果放入 W 寄存器。 如果“ d”为 1，结果放回寄存器“ f”。
周期	1
范例	INCX CNT, 1 B : CNT =0xFF, Z =0 A : CNT =0x00, Z =1

INCXSZ	“f”递增，如果为 0 则跳过
语法	INCXSZ f [,d]
操作数	f : 00h ~ 1FFh, d : 0, 1
运作方式	(目标) ← (f) + 1,如果结果为 0，则跳过下一条指令
影响的状态位	-
操作码	ff00 1111 dfff ffff
描述	寄存器“ f”的内容递增。如果“ d”为 0，结果放入 W 寄存器。如果“ d”为 1，结果放回寄存器“ f”。如果结果为 1，则执行下一条指令。如果结果为 0，则改为执行 NOP，使其成为 2 周期指令。
周期	1 or 2
范例	LABEL1 INCXSZ CNT, 1 GOTO LOOP CONTINUE B : PC =LABEL1 A : CNT =CNT + 1 如果 CNT =0, PC =CONTINUE 如果 CNT ≠0, PC =LABEL1 + 1

IORLW	W 和立即数“ k”逻辑或
语法	IORLW k
操作数	k : 00h ~ FFh
运作方式	(W) ← (W) OR k
影响的状态位	Z
操作码	0001 1010 kkkk kkkk
描述	W 寄存器的内容与 8 位立即数“ k”进行或运算。 结果放在 W 寄存器中。
周期	1
范例	IORLW 0x35 B : W =0x9A A : W =0xBF, Z =0

IORWX	W 和立即数“ f”逻辑或
语法	IORWF f [,d]
操作数	f : 00h ~ 1FFh, d : 0, 1
运作方式	(目标) ← (W) OR k
影响的状态位	Z
操作码	ff00 0100 dfff ffff
描述	W 寄存器与寄存器“ f”进行或运算。 如果“ d”为 0，结果放入 W 寄存器。 如果“ d”为 1，结果放回寄存器“ f”。
周期	1
范例	IORWX RESULT, 0 B : RESULT =0x13, W =0x91 A : RESULT =0x13, W =0x93, Z =0

MOVX	移 "f"
语法	MOVX f,d
操作数	f : 00h ~ 1FFh
运作方式	(目标) ← (f)
影响的状态位	Z
操作码	ff00 1000 dfff ffff
描述	寄存器“ f”的内容根据 d 的状态移至目标。 如果 d = 0，则目标为 W 寄存器。 如果 d = 1，则目标是文件寄存器 f 本身。 d = 1 对测试文件寄存器很有用，因为状态标志 Z 受到影响。
周期	1
范例	MOVX FSR,0 B : FSR =0xC2, W =? A : FSR =0xC2, W =0xC2

MOVXW	将"f"移至 W
语法	MOVXW f
操作数	f : 00h ~ 1FFh
运作方式	(W) ← (f)
影响的状态位	Z
操作码	ff00 1000 0fff ffff
描述	寄存器'f'的内容移至 W 寄存器。
周期	1
范例	MOVXW FSR B : FSR =0xC2, W =? A : FSR =0xC2, W =0xC2

MOVLW	将立即数移至 W
语法	MOVLW k
操作数	k : 00h ~ FFh
运作方式	(W) ← k
影响的状态位	-
操作码	0001 1001 kkkk kkkk
描述	八位立即数“ k”被加载到 W 寄存器中。 无关位将为 0。
周期	1
范例	MOVLW 0x5A B : W =? A : W =0x5A

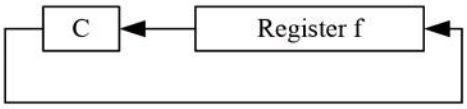
MOVWX	将 W 移至 "f"
语法	MOVWX f
操作数	f : 00h ~ 1FFh
运作方式	(f) ← (W)
影响的状态位	-
操作码	ff00 0000 1fff ffff
描述	将数据从 W 寄存器移至寄存器'f'。
周期	1
范例	MOVWX REG1 B : REG1 =0xFF, W =0x4F A : REG1 =0x4F, W =0x4F

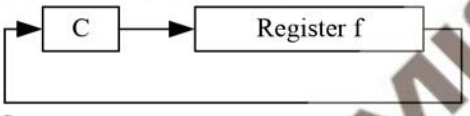
NOP	空操作
语法	NOP
操作数	-
运作方式	空操作
影响的状态位	-
操作码	0000 0000 0000 0000
描述	空操作
周期	1
范例	NOP -

RET	从子程序返回
语法	RET
操作数	-
运作方式	PC ← TOS
影响的状态位	-
操作码	0000 0000 0100 0000
描述	从子程序返回。 堆栈被弹出，并且堆栈的顶部（TOS）被装入程序计数器。 这是两个周期的指令。
周期	2
范例	RET

RETI	从中断返回
语法	RETI
操作数	-
运作方式	PC ← TOS, GIE ← 1
影响的状态位	-
操作码	0000 0000 0110 0000
描述	从中断返回。 弹出堆栈，并将堆栈顶层（TOS）加载到 PC 中。 中断使能。 这是两个周期的指令。
周期	2
范例	RETI A : PC =TOS, GIE =1

RETLW	W 带立即数返回
语法	RETLW k
操作数	k : 00h ~ FFh
运作方式	PC ← TOS, (W) ← k
影响的状态位	-
操作码	0001 1000 kkkk kkkk
描述	W 寄存器加载了八位立即数“ k”。 程序计数器从堆栈的顶层（返回地址）加载。 这是两个周期的指令。
周期	2
范例	CALL TABLE B : W =0x07 : A : W =k8 的值 TABLE ADDWX PCL, 1 RETLW k1 RETLW k2 : RETLW kn

RLX	"f" 通过进位向左移
语法	RLX f [,d]
操作数	f : 00h ~ 1FFh, d : 0, 1
运作方式	
影响的状态位	C
操作码	ff00 1101 dfff ffff
描述	寄存器“ f”的内容通过进位标志向左移动一位。如果“ d”为 0，结果放入 W 寄存器。 如果“ d”为 1，结果放回寄存器“ f”。
周期	1
范例	RLX REG1, 0 B : REG1 =1110 0110, C =0 A : REG1 =1110 0110 W =1100 1100, C =1

RRX	"f"通过进位向右移
语法	RRX f [,d]
操作数	f : 00h ~ 1FFh, d : 0, 1
运作方式	
影响的状态位	C
操作码	ff00 1100 dfff ffff
描述	寄存器“ f”的内容通过进位标志向右移动一位。如果“ d”为 0，结果放入 W 寄存器。如果“ d”为 1，结果放回寄存器“ f”。
周期	1
范例	RRX REG1, 0 B : REG1 =1110 0110, C =0 A : REG1 =1110 0110 W =0111 0011, C =0

SLEEP	进入睡眠模式，时钟振荡停止
语法	SLEEP
操作数	-
运作方式	-
影响的状态位	TO, PD
操作码	001 1110 0000 0011
描述	进入睡眠模式，时钟振荡停止
周期	1
范例	SLEEP -

SUBLW	W减去立即数
语法	SUBLW k
操作数	k : 00h ~ FFh
运作方式	(W) ←k- (W)
影响的状态位	C, DC, Z
操作码	0001 1111 kkkk kkkk
描述	W寄存器（2的补码方法）减去八位立即数“ k”。 结果放在W寄存器中。
周期	1
范例	SUBLW 0x15 B : W =0x25 A : W =0xF0

SUBWX	"f"减去 W
语法	SUBWX f [,d]
操作数	f : 00h ~ 1FFh, d : 0, 1
运作方式	(目标) ← (f) - (W)
影响的状态位	C, DC, Z
操作码	ff00 0010 dfff ffff
描述	从寄存器“ f”中减去 W 寄存器（2 的补码方法）。 如果 d 为 0，则结果放在 W 寄存器中。 如果“ d”为 1，结果放回寄存器“ f”。
周期	1
范例	SUBWX REG1, 1 B : REG1 =0x03, W =0x02, C =?, Z =? A : REG1 =0x01, W =0x02, C =1, Z =0 SUBWX REG1, 1 B : REG1 =0x02, W =0x02, C =?, Z =? A : REG1 =0x00, W =0x02, C =1, Z =1 SUBWX REG1, 1 B : REG1 =0x01, W =0x02, C =?, Z =? A : REG1 =0xFF, W =0x02, C =0, Z =0

SWAPX	"f"互换半字节
语法	SWAPX f [,d]
操作数	f : 00h ~ 1FFh, d : 0, 1
运作方式	(目标,7~4) ← (f.3~0), (目标.3~0) ← (f.7~4)
影响的状态位	-
操作码	ff00 1110 dfff ffff
描述	寄存器“ f”的高低半字节互换。 如果“ d”为 0，结果放入 W 寄存器。 如果“ d”为 1，结果放入寄存器“ f”。
周期	1
范例	SWAPX REG, 0 B : REG1 =0xA5 A : REG1 =0xA5, W =0x5A

TABRH **将 DPTR 高字节返回给 W**

语法	TABRH
操作数	-
运作方式	(W) ← ROM [DPTR]高字节内容,其中 DPTR = {DPH [max：8]，FSR [7：0]}
影响的状态位	-
操作码	0000 0000 0101 1000
描述	W寄存器加载ROM [DPTR]的高字节。 这是两个周期的指令。
周期	2
范例	

```
MOVLW       (TAB1&0xFF)
MOVWX       DPL                     ;Where DPL is register
MOVLW       (TBA1>>8)&0xFF
MOVWX       DPH                     ;Where DPH is register

TABRL                               ;W =0x89
TABRH                               ;W =0x37

                        ORG
0234H
TAB1:                               ;16 位 ROM 数据
DT                  0x3789,
0x2277
```

TABRL **将 DPTR 低字节返回给 W**

语法	TABRL
操作数	-
运作方式	(W) ← ROM [DPTR]低字节内容,其中 DPTR = {DPH [max：8]，FSR [7：0]}
影响的状态位	-
操作码	0000 0000 0101 0000
描述	W寄存器加载ROM [DPTR]的低字节。 这是两个周期的指令。
周期	2
范例	

```
MOVLW       (TAB1&0xFF)
MOVWX       DPL                     ;Where DPL register
MOVLW       (TBA1>>8)&0xFF
MOVWX       DPH                     ;Where DPH register

TABRL                               ;W =0x89
TABRH                               ;W =0x37

                        ORG
0234H
TAB1:                               ;16 位 ROM  数据
DT                  0x3789,
0x2277
```

TSTX	检测"f" 是否为 0
语法	TSTX f
操作数	f : 00h ~ 1FFh
运作方式	如果（f）为 0，则设置 Z 标志
影响的状态位	Z
操作码	ff00 1000 1fff ffff
描述	如果寄存器“ f”的内容为 0，则零标志设置为 1。
周期	1
范例	TSTX REG1 B : REG1 =0, Z =? A : REG1 =0, Z =1

XORLW	W 和立即数异或
语法	XORLW k
操作数	k : 00h ~ FFh
运作方式	(W) ← (W) XOR k
影响的状态位	Z
操作码	0001 1101 kkkk kkkk
描述	W 寄存器的内容与 8 位立即数“ k”进行异或。 结果放在 W 寄存器中。
周期	1
范例	XORLW 0xAF B : W =0xB5 A : W =0x1A

XORWX	W 和 "f"异或
语法	XORWX f [,d]
操作数	f : 00h ~ 1FFh, d : 0, 1
运作方式	(目标) ← (W) XOR (f)
影响的状态位	Z
操作码	ff00 0110 dfff ffff
描述	W 寄存器与寄存器“ f”的内容异或。 如果 d 为 0，则结果放在 W 寄存器中。 如果“ d”为 1，结果放回寄存器“ f”。
周期	1
范例	XORWX REG, 1 B : REG =0xAF, W =0xB5 A : REG =0x1A, W =0xB5

电气特性

1. 绝对最大额定值(T_A =25°C)

参数	范围	单位
电源电压	V_{SS} -0.3 至 V_{SS} +5.5	V
输入电压	V_{SS} -0.3 至 V_{CC} +0.3	V
输出电压	V_{SS} -0.3 至 V_{CC} +0.3	V
每个引脚的输出大电流	-25	mA
所有引脚的输出大电流	-80	mA
每个引脚的输出小电流	+30	mA
所有引脚的输出小电流	+150	mA
最大工作电压	5.5	V
工作温度	-40 至 +85	°C
储存温度	-65 至 +150	°C

2. DC 特性(TA =25°C, V_{CC} =5.0V,除非另有规定)

参数	符号	条件		最小	典型	最大	单位
工作电压	V_{CC}	Fsys = 8Mhz		2.1	-	5.5	
		Fsys = 4Mhz		1.6	-	5.5	
输入高电压	V_{IH}	所有输入	V_{CC} = 3~5V	0.6Vcc	-	Vcc	V
输入低电压	V_{IL}	所有输入	V_{CC} = 3~5V	Vss	-	0.2Vcc	V
输出大电流	I_{OH}	所有输出	V_{CC} = 5V, V_{OH} = 4.5V	6	12	-	mA
			V_{CC} = 3V, V_{OH} = 2.7V	2.5	5	-	
输出小电流	I_{OL}	所有输出	V_{CC} = 5V, V_{OL} = 0.5V	20	40	-	mA
			V_{CC} = 3V, V_{OL} = 0.3V	8	16	-	
输入漏电流(引脚为高)	I_{ILH}	所有输入	V_{IN} = V_{CC}	-	-	1	uA
输入漏电流(引脚为低)	I_{ILL}	所有输入	V_{IN} = 0V	-	-	-1	uA
电源电流(空载)	I_{CC}	快速模式 FIRC 8 MHz	V_{CC} = 5V	-	3.5	-	mA
		快速模式 FIRC 4 MHz	V_{CC} = 5V	-	2.7	-	
		快速模式 FIRC 2 MHz	V_{CC} = 5V	-	2.3	-	
		快速模式 FIRC 1 MHz	V_{CC} = 5V	-	2.1	-	
			V_{CC} = 3V	-	1.5	-	
		慢速模式 SIRC 70KHz OPA ON VR ON	V_{CC} = 5.0V	-	1300	-	uA
			V_{CC} = 3.0V	-	1100	-	
		慢速模式 SIRC 70KHz OPA STOP VR STOP	V_{CC} = 5.0V		750		
			V_{CC} = 3.0V		620		

		停止模式 LVRSAV = 1	V_{CC} = 5.0V	-	0.1	-	uA
			V_{CC} = 3.0V	-	0.1	-	
参数	符号	条件		最小	典型	最大	单位
电压电流 （空载）	I_{CC}	空闲模式 SIRC 70 KHz LVRSAV= 1	V_{CC} = 5.0V	-	4.2	-	uA
			V_{CC} = 3.0V	-	1.2	-	
上拉电阻	R_{UP}	V_{IN} = 0 V 端口 A	V_{CC} = 5.0V	-	41	-	KΩ
			V_{CC} = 3.0V	-	76	-	
内部参考电压	VR	VCC = 5.0V，至少 1uF 接地		-1.2%	3	1.2%	V

3. 时钟时序（T_A = -40℃ 至 +85℃）

参数	条件	最小	典型值	最大	单位
FIRC 频率（*）	-40°C ~ 85°C, V_{CC} = 3.0 ~ 5.0V	-2%	8	+2%	MHz
	-40°C ~ 85°C, V_{CC} = 4.0 V	-2%	8	+1.5%	
	0°C ~ 70°C, V_{CC} = 4.0 V	-2%	8	+1.5%	
	25°C, V_{CC} = 3.0 ~ 5.0 V	-1.0%	8	+2%	
	25°C, V_{CC} =4.0 V	-0.5%	8	+0.5%	

(*) FIRC frequency can be 除以 ided by 1/2/4/8.

4. 复位时间特性（T_A = 25℃）

参数	条件	最小	典型值	最大	单位
复位输入低脉宽	输入 V_{CC} = 5 V ±10 %	30	-	-	μs
WDT 时间	V_{CC} = 3 V, WDTPSC = 11	-	1920	-	ms
	V_{CC} = 5 V, WDTPSC = 11		1760		
WKT 时间	V_{CC} = 3 V, WKTPSC = 11	-	120	-	ms
	V_{CC} = 5 V, WKTPSC = 11		108		
CPU 启动时间	V_{CC} = 5 V	-	24	-	ms

5. LVR 电路特性（TA = 25°C）

参数	条件	最小	典型值	最大	单位
LVR 参考电压	LVR_{th}	-	2.2	-	V
		-	2.8	-	
		-	3.6	-	
		-	4.2	-	
LVR 迟滞电压	V_{HYST}	-	±0.1	-	V
低电压检测时间	t_{LVR}	100	-	-	μs

6. ADC 电气特性(T_A = 25°C, V_{CC} = 3.0V 至 5.5V, V_{SS} = 0V)

参数	条件	最小	典型值	最大	单位
总精度	V_{CC} = 5V, V_{SS} = 0V, f_{ADC} = 1 MHz	-	±2.5	±13	LSB
积分非线性		-	±3.2	±15	
微分非线性		-	±1	±4	
最大输入时钟频率. (f_{ADC})	源阻抗 (Rs<10K omh)	-	-	2	MHz
	源阻抗 (Rs<20K omh)	-	-	1	
	源阻抗 (Rs<50K omh)	-	-	0.5	
	来源于 VR (ADCHS=1101)	-	-	2	
转换时间	f_{ADC} = 1 MHz	-	50	-	μs
输入电压	-	V_{SS}	-	Vcc	V

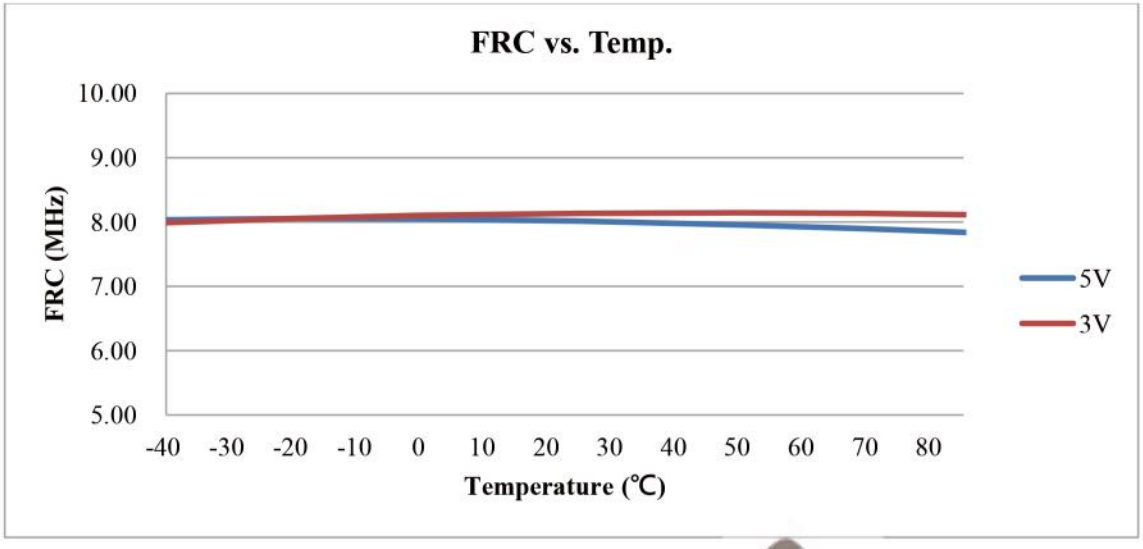
7. BCM 电气特性(T_A = 25°C, V_{CC} = 5V, V_{SS} = 0V)

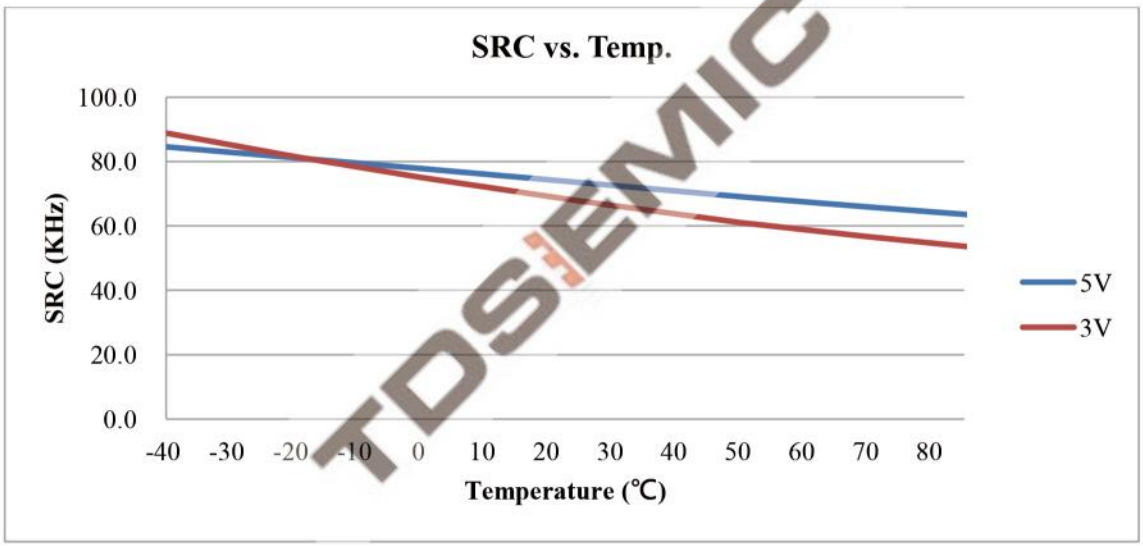
参数	条件	最小	典型值	最大	单位
电源 V_{CC}	-	2.2	-	5.5	V
OPA 偏移	修改后, OPA_输入=2.5 V	-	±1	-	mV
OPA Vcm	-	0.1	-	V_{CC}-1.2	V
OPA IOL	V_i=V_{CC}, V_{i+}=V_{SS}, T_{in}=100 ohm	-	20	-	mA
OPAAOL	RL=1M ohm, CL=100 pF	60	80	-	dB
OPA GBW	RL=1M ohm, CL=100 pF	-	1	-	MHz
OPA SR	No load	-	1	2	V/usec
OPA 电流功耗	V_{CC}=5 V	-	-	150	uA
DAC 参考 VR	V_{CC}=5 V, 内部 LDO	-	3	-	V
DAC 输出范围	VR=3 V	0	-	3	V
DAC 电流功耗	VCC=5 V	-	100	200	uA
DAC 积分非线性度	VCC=5 V, VSS=0 V, VR=3 V (DAC0DL >3Ch & DAC1DL >3Ch)	-	±2	-	LSB
DAC 差分非线性度		-	±2	-	
DAC 输出阻抗	R-2R+R 结构	-	10	-	Kohm

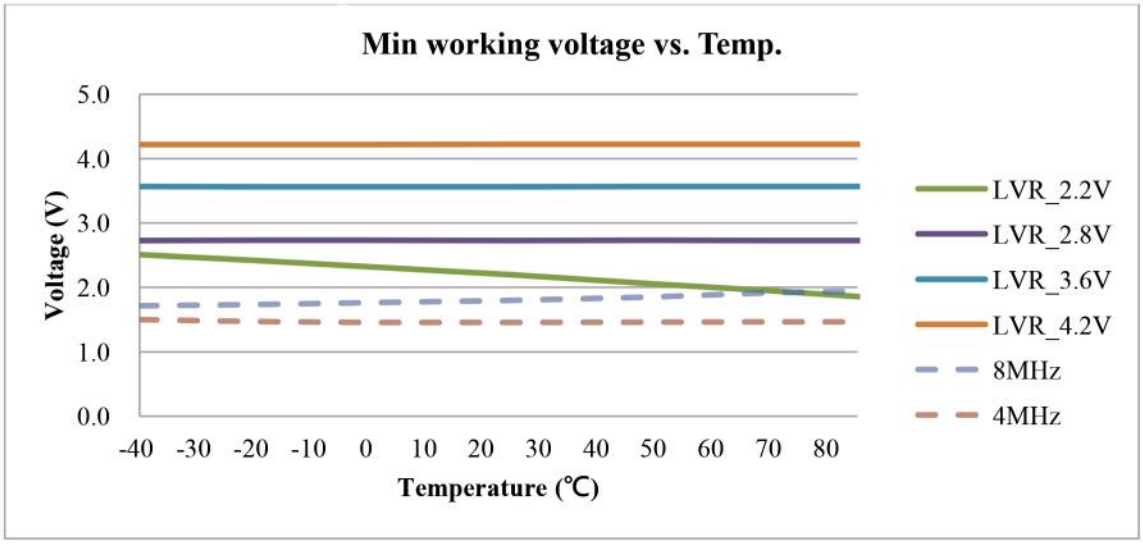
8. EEPROM 特性 (T_A = 25°C, V_{CC} = 5V, V_{SS} = 0V)

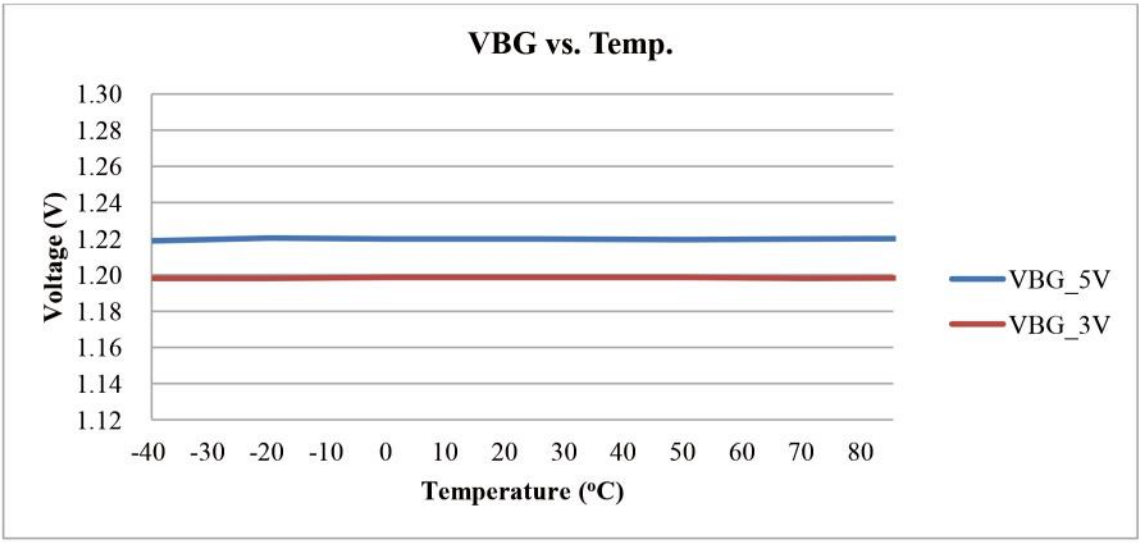
Parameter	Conditions	Min	Typ	Max	Units
读取电压	VCC	1.8	-	5.5	V
写入电压	VCC	2.7	-	5.5	V
写入电流	-	-	5	30	mA
写入时间	Byte Write Time	-	0.7	-	ms
擦除次数 (Byte Write)	-	-	-	50,000	cycles

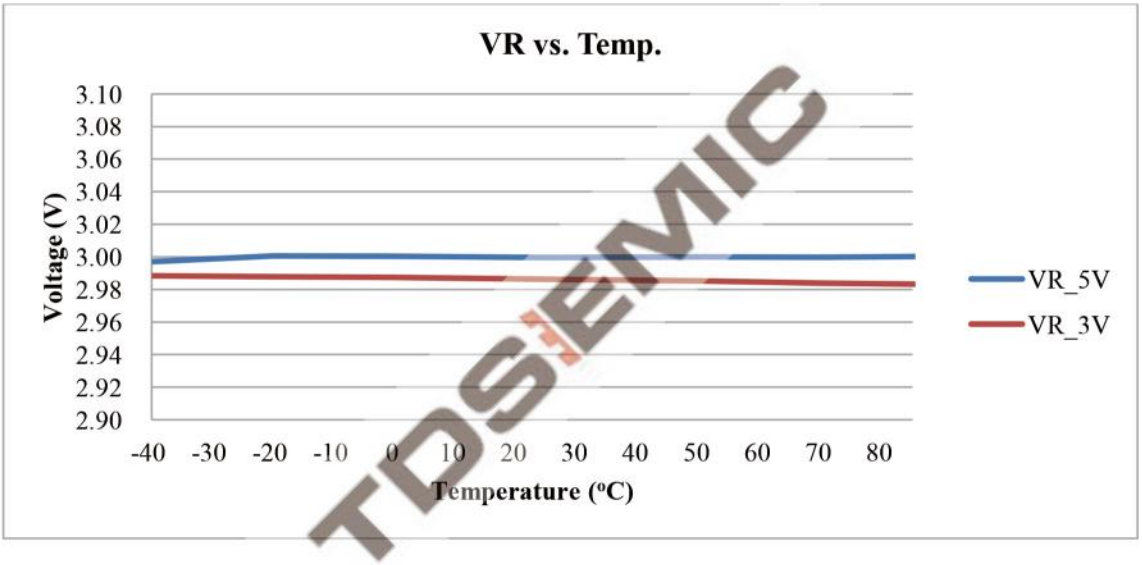
9. 电气特性图

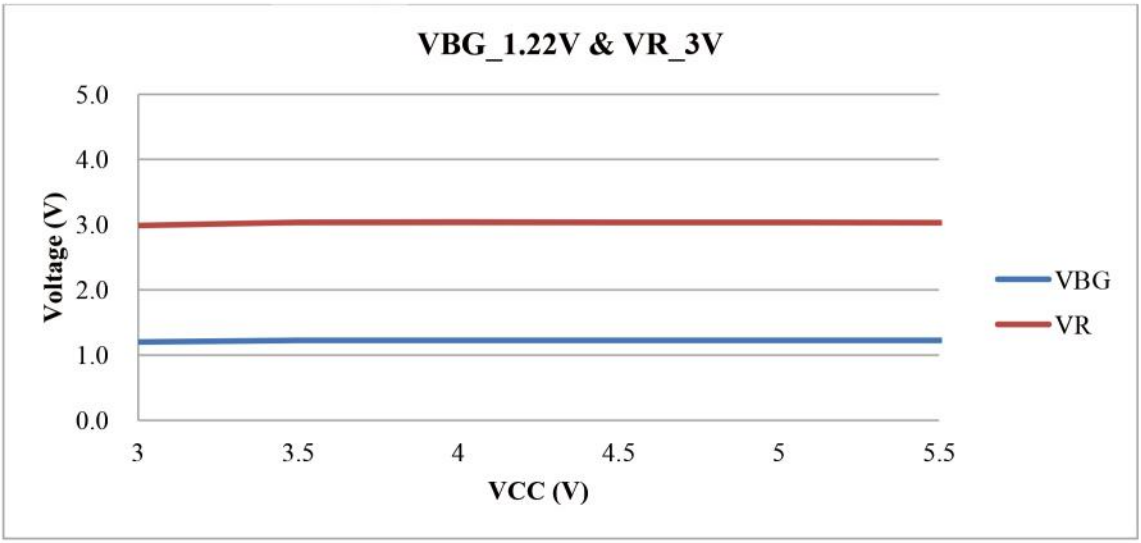

FRC vs. Temp.
FRC (MHz)
10.00
9.00
8.00
7.00
6.00
5.00
-40 -30 -20 -10 0 10 20 30 40 50 60 70 80
Temperature (℃)
5V
3V


SRC vs. Temp.
SRC (KHz)
100.0
80.0
60.0
40.0
20.0
0.0
-40 -30 -20 -10 0 10 20 30 40 50 60 70 80
Temperature (℃)
5V
3V

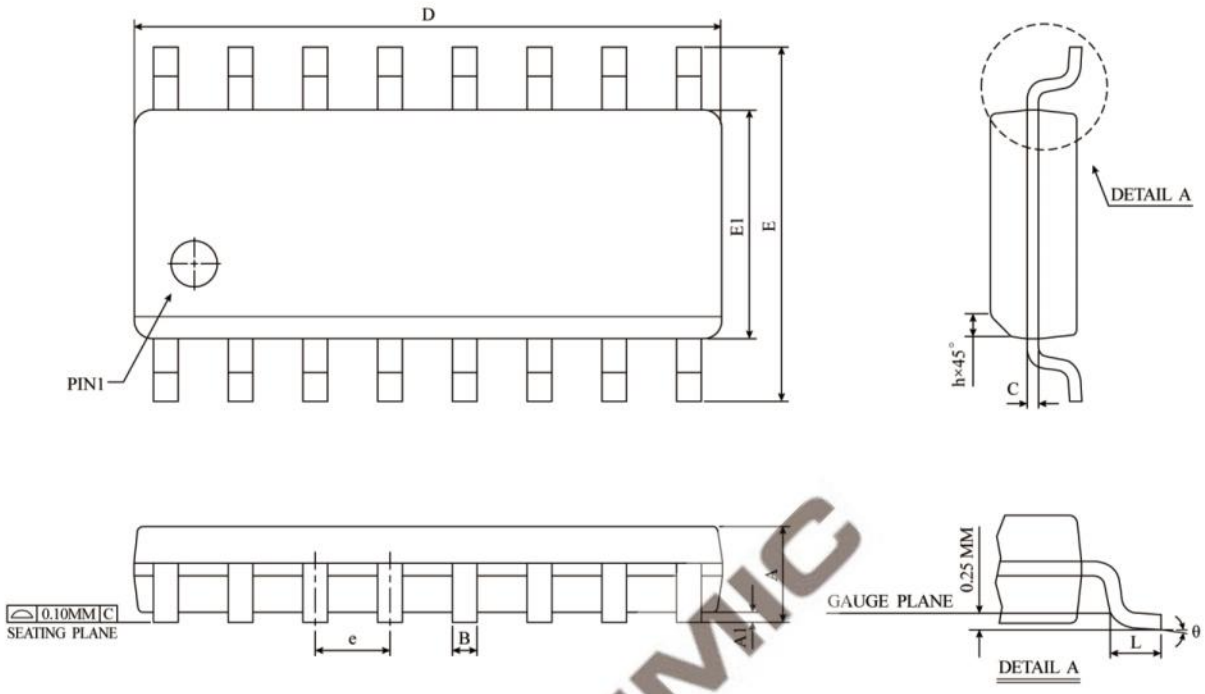

Min working voltage vs. Temp.
Voltage (V)
5.0
4.0
3.0
2.0
1.0
0.0
-40 -30 -20 -10 0 10 20 30 40 50 60 70 80
Temperature (℃)
LVR_2.2V
LVR_2.8V
LVR_3.6V
LVR_4.2V
8MHz
4MHz


VBG vs. Temp.
Voltage (V)
1.30
1.28
1.26
1.24
1.22
1.20
1.18
1.16
1.14
1.12
-40 -30 -20 -10 0 10 20 30 40 50 60 70 80
Temperature (°C)
VBG_5V
VBG_3V


VR vs. Temp.
Voltage (V)
3.10
3.08
3.06
3.04
3.02
3.00
2.98
2.96
2.94
2.92
2.90
-40 -30 -20 -10 0 10 20 30 40 50 60 70 80
Temperature (°C)
VR_5V
VR_3V


VBG_1.22V & VR_3V
Voltage (V)
5.0
4.0
3.0
2.0
1.0
0.0
3 3.5 4 4.5 5 5.5
VCC (V)
VBG
VR

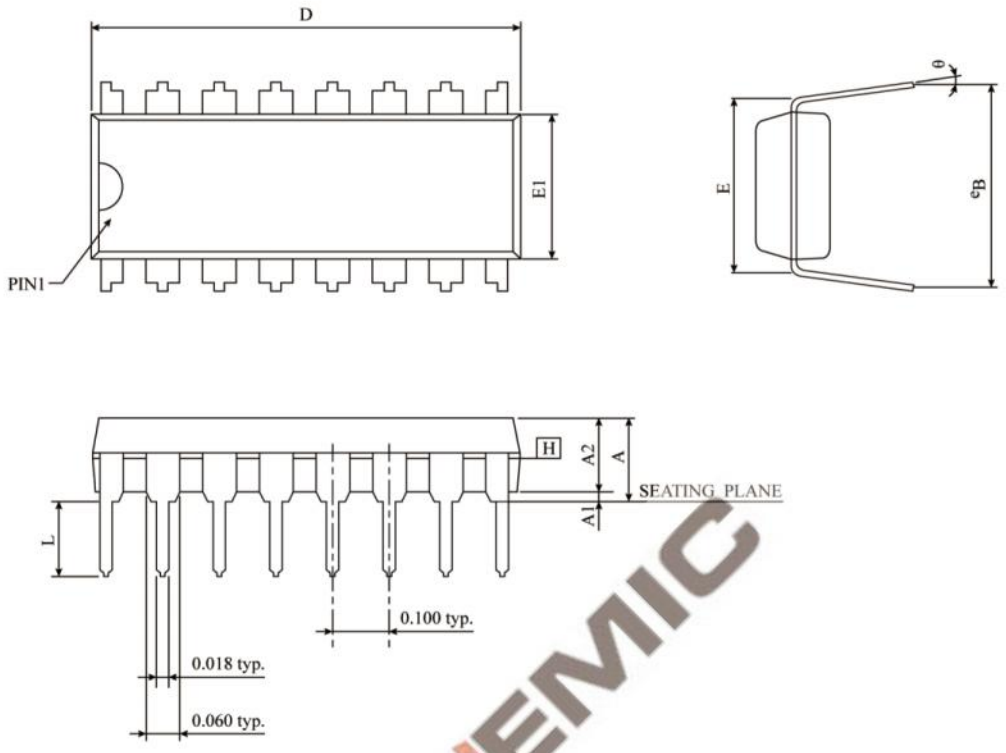
16-SOP 封装尺寸



SYMBOL	DIMENSION IN MM			DIMENSION IN INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	1.35	1.55	1.75	0.0532	0.0610	0.0688
A1	0.10	0.18	0.25	0.0040	0.0069	0.0098
B	0.33	0.42	0.51	0.0130	0.0165	0.0200
C	0.19	0.22	0.25	0.0075	0.0087	0.0098
D	9.80	9.90	10.00	0.3859	0.3898	0.3937
E	5.80	6.00	6.20	0.2284	0.2362	0.2440
E1	3.80	3.90	4.00	0.1497	0.1536	0.1574
e	1.27 BSC			0.050 BSC		
h	0.25	0.38	0.50	0.0099	0.0148	0.0196
L	0.40	0.84	1.27	0.0160	0.0330	0.0500
θ	0°	4°	8°	0°	4°	8°
JEDEC	MS-012 (AC)					

⚠ * NOTES : DIMENSION " D " DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS. MOLD FLASH, PROTRUSIONS AND GATE BURRS SHALL NOT EXCEED 0.15 MM (0.006 INCH) PER SIDE.

16-DIP 封装尺寸



SYMBOL	DIMENSION IN MM			DIMENSION IN INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	-	-	4.369	-	-	0.172
A1	0.381	0.673	0.965	0.015	0.027	0.038
A2	3.175	3.302	3.429	0.125	0.130	0.135
D	18.669	19.177	19.685	0.735	0.755	0.775
E		7.620 BSC			0.300 BSC	
E1	6.223	6.350	6.477	0.245	0.250	0.255
L	2.921	3.366	3.810	0.115	0.133	0.150
^{e}B	8.509	9.017	9.525	0.335	0.355	0.375
θ	0°	7.5°	15°	0°	7.5°	15°
JEDEC	MS-001 (BB)					

NOTES：

1. "D" , "E1" DIMENSIONS DO NOT INCLUDE MOLD FLASH OR PROTRUSIONS. MOLD FLASH OR PROTRUSIONS SHALL NOTEXCEED .010 INCH.
2. eB IS MEASURED AT THE LEAD TIPS WITH THE LEADS UNCONSTRAINED.
3. POINTED OR ROUNDED LEAD TIPS ARE PREFERRED TO EASE INSERTION.
4. DISTANCE BETWEEN LEADS INCLUDING DAM BAR PROTRUSIONS TO BE .005 INCH MININUM.
5. DATUM PLANE H COINCIDENT WITH THE BOTTOM OF LEAD, WHERE LEAD EXITS BODY.