十速

TM56M1511

TM56M1531

规格书

版本0.90

修改记录

版次	日期	描述
0.90	Mar, 2023	新颁

目录

比较表

	EV8239B	TM56P8336	TM56M1531
EV board	–	EV8239B	EV8239B
Touch Key	TK转换结束，CTKCKO继续运行	TK转换结束，CTKCKO停止运行并保持1	TK转换结束，CTKCKO停止运行并保持1

特性

1. **ROM:**
   - 2K x 16 位 MTP (可多次编程的 ROM)
2. **RAM: 256 x 8 位**
3. **堆栈: 8 级**
4. **系统时钟类型选择:**
   - 内部 RC 快时钟: 16 MHz (FIRC)
   - 内部 RC 慢时钟: (SIRC, 80 KHz @4V)
5. **系统时钟预分频器:**
   - 系统时钟可以 1/2/4/8 分频选项
6. **省电工作模式**
   - 快速模式： 慢时钟使能，CPU 在快速时钟状态下运行
   - 慢时钟模式： 快时钟可以关闭或使能，CPU 在慢时钟状态下运行
   - 空闲模式：快速时钟和 CPU 停止，慢时钟继续运行
   - 停止模式：所有的时钟停止
7. **2 个独立定时器**
   - Timer0
     - 8 位计时器除以 1~256 预分频选项，具有自动重新加载/计数器/中断/停止功能
   - Timer1
     - 8 位计时器除以 1~256 预分频选项，具有自动重新加载/中断/停止功能
8. **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 / WKT 中断
   - PWM 中断
   - 触摸键中断
   - LVD 中断

9. 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

  - 12 ms / 25 ms / 50 ms / 100 ms @ 4V

10. 看门狗定时器 (WDT)

- 由内置 RC 振荡器计时，具有 4 个可调的复位时间

  - 100 ms / 200 ms / 800 ms / 1600 ms @ 4V

- 在停止模式下可以关闭 / 使能看门狗定时器

11. 5 个 16-位 PWM（共享期间）

- PWM0, PWM1, PWM2, PWM4, PWM5
- 5 个单独的工作占空比可调，共享的周期可调
- PWM 时钟源：系统时钟（Fsys）、FIRC（16 MHz）或 FIRC*2（32MHz）

12. 触摸键(仅限 M1531)

- 两种不同的架构： CTK 和 STK
- 4 个带有数据计数器的通道触摸键
- 1 个针对 CTK 的外部 CLD
- 1 个内部参考电容器

13. 复位源

- 上电复位
- 看门狗复位
- 低电压复位

  外部引脚复位

14. 低电压复位 (LVR) / 低电压检测 (LVD)

- 16-级低电压复位: 2.05V ~ 4.15V, 可以选择被关闭
- 15 级低电压检测: 2.20V ~ 4.15V, 可以选择被关闭

15.工作电压

Fsys= 16 MHz，1.9V~5.5V @LVR 禁用@25°C。建议 LVR 2.05V @-40°C~105°C

注意：通电的 VCC 必须超过 POR 和所选的 LVR 电平以上，请参考“电气特性图”，以避免进入 ROM 死区。

16. 工作温度范围： -40°C ~ 105°C

17. 表读取指令：16 位 ROM 数据查找表

18. 集成的 16 位循环冗余检查（CRC）功能

19. 指令集：39 条指令

20. I/O 端口:

- 最多 6 个可编程 I/输出引脚
  - - 开漏输出
  - - CMOS 推挽输出
  - - 施密特触发器输入，带上拉/下拉电阻选项
  - - 所有 I/O 具备高灌电流
  - - 1/2 V CC (1/2 偏压) 输出

21. 编程连接支持 5 线（ICP）或 8 线程序

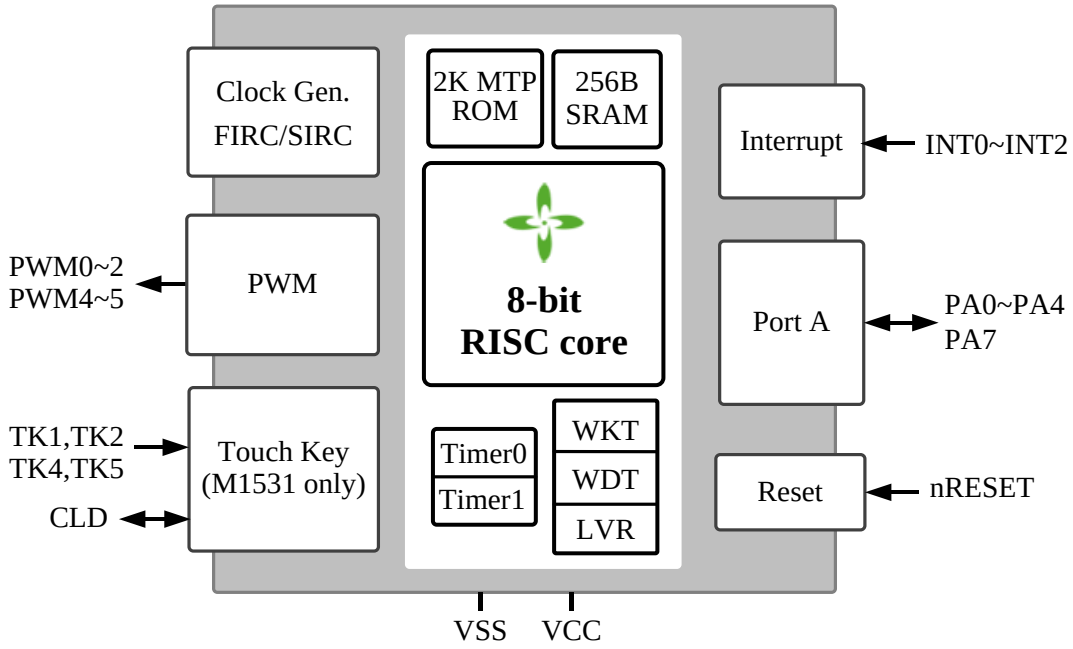
22. 封装类型:

- 8-pin SOP (150 mil)
- 6-pin SOT23-6

23. 片上调试/ICE 接口

EV board: EV8239B

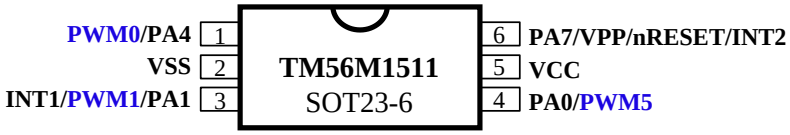
系统框图

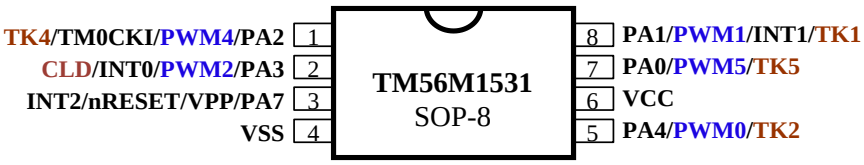


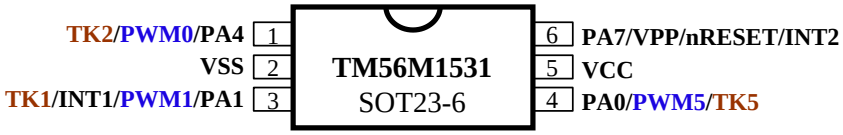
TM56M1511/31 Block Diagram

引脚分配图


TM0CKI/PWM4/PA2 1
INT0/PWM2/PA3 2
INT2/nRESET/VPP/PA7 3
VSS 4
TM56M1511
SOP-8
8 PA1/PWM1/INT1
7 PA0/PWM5
6 VCC
5 PA4/PWM0


PWM0/PA4 1
VSS 2
INT1/PWM1/PA1 3
TM56M1511
SOT23-6
6 PA7/VPP/nRESET/INT2
5 VCC
4 PA0/PWM5


TK4/TM0CKI/PWM4/PA2 1
CLD/INT0/PWM2/PA3 2
INT2/nRESET/VPP/PA7 3
VSS 4
TM56M1531
SOP-8
8 PA1/PWM1/INT1/TK1
7 PA0/PWM5/TK5
6 VCC
5 PA4/PWM0/TK2


TK2/PWM0/PA4 1
VSS 2
TK1/INT1/PWM1/PA1 3
TM56M1531
SOT23-6
6 PA7/VPP/nRESET/INT2
5 VCC
4 PA0/PWM5/TK5

引脚描述

名称	输入/输出	引脚描述
PA0~PA4, PA7	I/O	可位编程 I/O 端口，为施密特触发器输入，CMOS 推拉输出，开漏极输出或 1/2VCC 输出。上拉/下拉电阻可通过软件分配
nRESET	I	外部低电平有效复位
VCC, VSS	P	电源电压输入引脚和接地
VPP	P	程序高压输入
INT0–INT2	I	外部中断输入
TM0CKI	I	计数器模式下的计数器 0 的输入
PWM0~2,PWM4~5	O	16 位 PWM 输出
TK1,TK2,TK4,TK5	I	触摸键输入
CLD	I/O	外部触摸键充电电容连接引脚

编程引脚：
正常模式：VCC / VSS / VPP / PA0 / PA1 / PA2 / PA3 / PA4
ICP 模式：VCC / VSS / VPP / PA0 / PA1
-当使用 ICP（在线编程）模式时，PCB 需要除去 PA0、PA1 的所有部件

引脚摘要

SOP8-A	引脚名称	类型	GPIO			替代功能		
			输入	输出				
TM56M1511/31			Ext. Interrupt	O.D	P.P	PWM	TK (M1531 Only)	MISC
1	PA2/PWM4/TM0CKI/TK4	I/O		O	O	O	O	TM0CKI
2	PA3/PWM2/INT0/CLD	I/O	O	O	O	O	O	
3	PA7/VPP/nRESET/INT2	I/O	O	O	O			nRESET
4	VSS	P						
5	PA4/PWM0/TK2	I/O		O	O	O	O	
6	VCC	P						
7	PA0/PWM5/TK5	I/O		O	O	O	O	
8	PA1/PWM1/INT1/TK1	I/O	O	O	O	O	O	

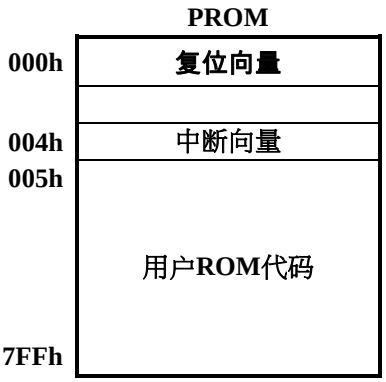
Symbol：P.P. = COM Push-Pull 输出
O.D. = Open Drain 输出

功能描述

1. CPU 核心

1.1 程序 ROM (PROM)

该器件的 MTP（多次可编程）ROM 为 2K 字，带有一个额外的信息区域来存储 SYSCFG 。只要 SYSCFG 的 PROT（CFGWH[15]）位不设置 1，就可以写入 ROM。



1.1.1 复位向量 (000H)

重置后，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到默认值。

1.1.2 中断向量(004H)

当中断发生时，程序计数器（PC）将被推到堆栈上，并跳转到地址 004h。

1.2 系统配置寄存器(SYSCFG)

系统配置寄存器 (SYSCFG) 位于 MTP ROM 信息区域；它包含一个 16 位寄存器 (CFGWH)。SYSCFG 决定了 CPU 初始状态的选项。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 15 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。

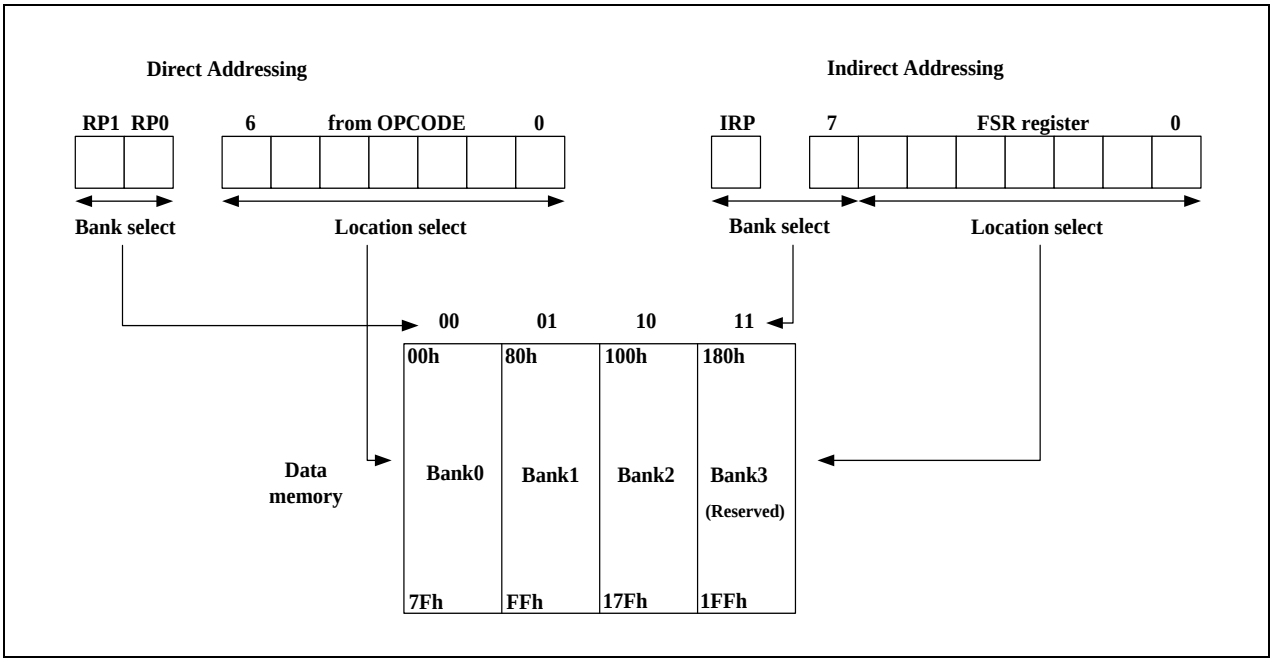
位		**15~0**	
默认值		**1111_1111_1111_1111**	
位		描述	
CFGWH	15	**PROT**: 代码保护选择	
		0	关闭
		1	使能
	14	Tenx 保留	
		0	
		1	
	13-12	**WDTE**: WDT 复位使能	
		0X	使能关闭
		10	在快速/慢速模式下使能，在空闲/停止模式下关闭
		11	始终启用
	11-8	**LVR**: 低电压复位模式	
		0000	Low Voltage Reset 2.05V
		0001	Low Voltage Reset 2.20V
		0010	Low Voltage Reset 2.30V
		0011	Low Voltage Reset 2.45V
		0100	Low Voltage Reset 2.60V
		0101	Low Voltage Reset 2.75V
		0110	Low Voltage Reset 2.90V
		0111	Low Voltage Reset 3.00V
		1000	Low Voltage Reset 3.15V
		1001	Low Voltage Reset 3.30V
		1010	Low Voltage Reset 3.45V
		1011	Low Voltage Reset 3.60V
		1100	Low Voltage Reset 3.70V
		1101	Low Voltage Reset 3.85V
		1110	Low Voltage Reset 4.00V
		1111	Low Voltage Reset 4.15V
	7	**XRSTE**: 外部引脚 (PA7) 复位使能	
		0	关闭 (PA7 作为 I/O 引脚)
		1	使能
	6-0	Tenx 保留	

1.3 RAM 寻址模式

CPU 中有一个数据存储器分为四个部分 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。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。
位 RP1 和 RP0（状态[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 寄存器进行写值为间接寻址，INDF 寄存器不是实质寄存器，任何使用 INDF 寄存器的指令实际上都为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。



直接 / 间接寻址

在 F/W 代码的开头保持 RP0=RP1=0 并使用新的指令集。

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:

BC**F**	TM0IE	➔	BC**X**	TM0IE
DEC**F**	CNT, 1	➔	DEC**X**	CNT, 1
INC**F**SZ	RAM25, 0	➔	INC**X**SZ	RAM25, 0
MOVW**F**	PAMODL	➔	MOVW**X**	PAMODL
RL**F**	RAMA0, 0	➔	RL**X**	RAMA0, 0
SWAP**F**	ADCTL, 0	➔	SWAP**X**	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	PAMOD10	105h		185h	DPL
06h		86h	PAMOD32	106h		186h	DPH
07h		87h	PAMOD54	107h		187h	CRCDL
08h		88h	PAMOD76	108h		188h	CRCDH
09h		89h	PWMCTL	109h	LVRPD	189h	CRCIN
0Ah	**PCLATH**	8Ah	**PCLATH**	10Ah	**PCLATH**	18Ah	**PCLATH**
0Bh	**INTIE**	8Bh	**INTIE**	10Bh	**INTIE**	18Bh	**INTIE**
0Ch	INTIF	8Ch		10Ch	PCH	18Ch	TABR
0Dh	INTIE1	8Dh		10Dh		18Dh	
0Eh	INTIF1	8Eh		10Eh	BGTRIM	18Eh	
0Fh	CLKCTL	8Fh		10Fh	IRCF	18Fh	
10h	TM0RLD	90h		110h		190h	
11h	TM0CTL	91h	OPTION2	111h		191h	
12h	TM1	92h	PWMPRDH	112h		192h	
13h	TM1RLD	93h	PWMPRDL	113h		193h	
14h	TM1CTL	94h	PWM0DH	114h		194h	
15h		95h	PWM0DL	115h		195h	
16h	LVCTL	96h	PWM1DH	116h		196h	
17h		97h	PWM1DL	117h		197h	
18h		98h	PWM2DH	118h		198h	
19h		99h	PWM2DL	119h		199h	
1Ah	TKDL	9Ah		11Ah		19Ah	
1Bh	TKDH	9Bh		11Bh		19Bh	
1Ch	TKTMRL	9Ch	PWM4DH	11Ch		19Ch	
1Dh	TKMFS	9Dh	PWM4DL	11Dh		19Dh	
1Eh	TKCTL	9Eh	PWM5DH	11Eh		19Eh	
1Fh	TKCTL2	9Fh	PWM5DL	11Fh		19Fh	
20h RAM Bank0 area (80 Bytes) 6Fh		A0h RAM Bank1 area (80 Bytes) EFh		120h RAM Bank2 area (80 Bytes) 16Fh		1A0h RAM Bank3 area (80 Bytes) 1EFh	
70h **common area** **16 Bytes** 7Fh		F0h **accesses** **70h~7Fh** FFh		170h **accesses** **70h~7Fh** 17Fh		1F0h **accesses** **70h~7Fh** 1FFh	

◇范例: 通过使用直接寻址来读 / 写寄存器 (RP0=RP1=0)

```
TM1       equ    12H      ;SFR 在Bank0
OPTION2   equ    91H      ;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             ; 读取TM1 (Bank0) 至W
MOVXW     OPTION2         ; 读取OPTION2 (Bank1) 至W
MOVXW     IRCF            ; 读取IRCF (Bank2) 至W
MOVXW     DPL             ; 读取DPL (Bank3) 至W

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

MOVLW     37H
MOVWX     LVRPD           ; LVRPD = W = 37h,强制关闭 LVR/POR

MOVXW     CLKCTL          ; 读取SFR CLKCTL (0Fh) 至W
MOVXW     IRCF            ; 读取SFR IRCF (10Fh) 至W

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

◇范例: 通过使用间接寻址来读 / 写寄存器 (RP0=RP1=0)

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

1.4 编程计数器（PC）和堆栈

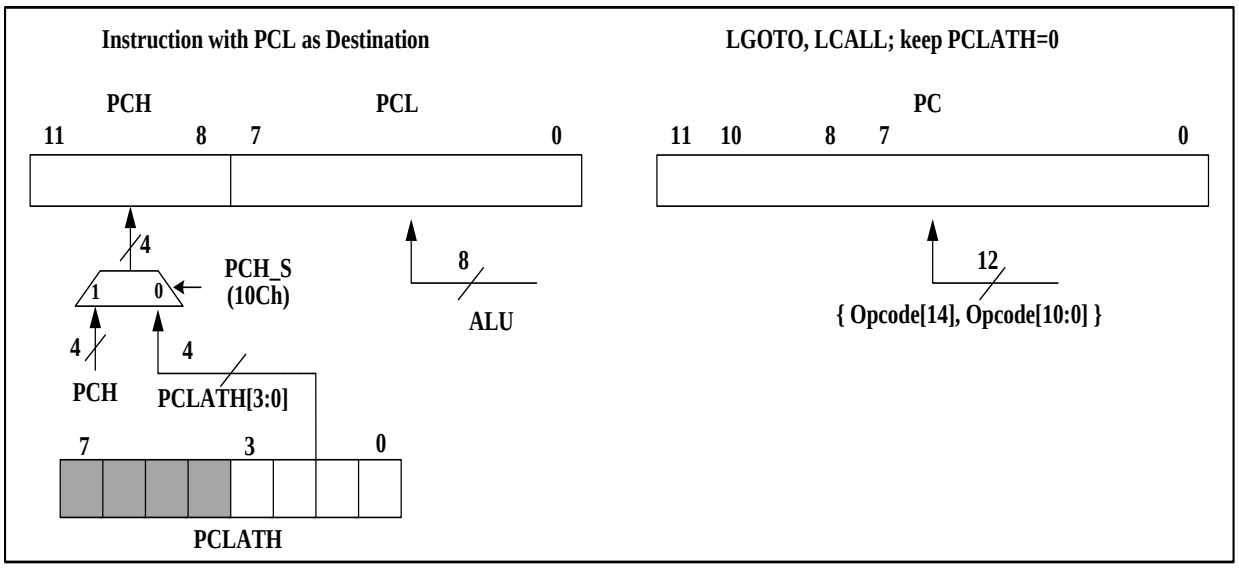
该程序计数器为 11 位的编程计数器，能够寻址一个 2K x 16 的 MTP ROM。当执行一个程序指令时，PC 将包含下一个要执行的程序指令的地址。除以下情况外，PC 值通常会增加 1。初始设置复位向量（000h）和中断向量（004h）用于 PC 初始化和中断。对于 CALL/GOTO 指令，PC 从指令字加载下 11 位地址。对于 RET/RETI/RETLW 指令，PC 从顶级堆栈中检索其内容。

与 RAM 寻址模式（请参考 1.3)类似，芯片提供了新的指令集 LCALL/LGOTO 来替换 CALL/GOTO 指令集。

编程计数器（PC[7：0]）的低字节数据可以由 PCL 寄存器（002h/082h/102h/182h）进行读写。编程计数器（PC[11：8]）的高字节数据只能通过 PCH 寄存器（10Ch）读取。当执行以 PCL 寄存器为目标的任何指令时，内部标志 PCH_S 用于选择 PCH 的源。将 0x1C 写入 PCH 寄存器可以设置 PCH_S，将其他值写入 PCH 寄存器将清除 PCH_S。重置后，PCH_S 将被清除。

当 PCH_S 被清除为“0”时，同时执行任何以 PCL 寄存器为目标的指令，都会导致 PCH 被 PCLATH（00 Ah/08Ah/10Ah/18Ah）寄存器的内容所取代。这允许通过将所需的高字节写入 PCLATH 寄存器来更改程序计数器的整个内容。当低字节被写入 PCL 寄存器时，程序计数器的所有内容将变为 PCLATH 寄存器中包含的值和那些被写入 PCL 寄存器的值。

当 PCH_S 设置为“1”时，执行以 PCL 寄存器为目标的任何指令，低字节将被写入 PCL 寄存器，并且不会改变 PCH。当使用任何以 PCL 寄存器为目标的指令时，建议将 PCH_S 设置为“1”，但 C 语言不支持此函数。



002h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PCL	PCL							
R/W	R/W							
Reset	0	0	0	0	0	0	0	0

002h.7~0 **PCL:** 程序计数器数据位 7~0

00Ah	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PCLATH	GPR				-	PCLATH		

R/W	R/W				-	R/W		
Reset	0	0	0	0	-	0	0	0

00Ah.2~0 **PCLATH:** 针对程序计数器的高字节的写缓冲区

10Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PCH	PCH							
R/W	W				R/W			
Reset	0	0	0	0	0	0	0	0

10Ch.7~0 **PCH (W):** 当执行以 PCL 作为目标的指令时，编程计数器高字节源选择 写 0x1C 来设置 PCH_S = 1： PCH 保持原始值 写其他的东西来清除 PCH_S = 0： PCH 是来自 PCLATH 的

10Ch.3~0 **PCH (R):** 编程计数器数据位 11~8

堆栈为 11 位宽度，8 阶深度。LCALL 指令和硬件中断将按顺序推送堆栈，当执行 RET/RETI/ RETLW 指令时按顺序弹堆栈。对于表查该器件提供了功能强大的表读取指令 TABRL 及 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。

例如：查找位于“ TABLE1”和“ TABLE2”的 PROM 数据。

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

START:
            MOVLW       00h
            MOVWX       INDEX           ; 设置查找表的地址
            MOVLW       1Ch             ; 写入1Ch到PCH以设置PCH_S标志
            MOVWX       PCH
LOOP:
            MOVXW       INDEX           ; 将索引值移到W寄存器
            LCALL       TABLE1          ; 查找数据
            …
            INCX        INDEX, 1        ; 增加下一个地址的索引地址
            …
            LGOTO       LOOP            ; 转到LOOP标签
            …
            MOVLW       (TABLE2 >>8) & 0xff
            MOVWX       DPH             ; DPH寄存器（F186h.3~0）
            MOVLW       (TABLE2) & 0xff
            MOVWX       DPL             ; DPL寄存器（F185h.7~0）
; Table Read by instructions TABRL / TABRH
            TABRL                       ; 将PROM低字节数据读取到W（W = 86h）
            TABRH                       ; 将PROM高字节数据读取到W（W = 19h）
            …
; Table Read by SFR TABR
            MOVLW       01h             ; TABR = 01h =指令TABRL
            MOVWX       TABR            ; 将PROM低字节数据读取到TABR（TABR = 86h）
            MOVXW       TABR            ; 读取TABR到W（W = 86）
            MOVLW       02h             ; TABR = 02h =指令TABRH
            MOVWX       TABR            ；将PROM高字节数据读取到TABR（TABR =
```

```
                                                    19h）
            MOVXW       TABR                    ;读取TABR到W（W = 19）
            …
ORG         X00h
TABLE1:
            ADDWX       PCL, 1                  ; 将W与PCL相加，然后将结果返回到PCL中。
            RETLW       55h                     ; W=55h when return
            RETLW       56h                     ; W=56h when return
            RETLW       58h                     ; W=58h when return
            …
TABLE2:
            .DT         0x1986                  ; 16-bit ROM data
            .DT         0x3719
            …
```

注： 芯片将 256 个 ROM 地址定义为一页，因此 ROM 有 16 页，000h~0FFh，100h~1FFh，...，F00h~FFFh。换句话说，PC 可以被定义为页面。查找表必须位于同一个页面上，以避免获取错误的数据。因此，例如，在 X00h (X = 1, 2, 3, ... , E, F) 启动一个查找表时，查找表最多有 255 个数据。如果查找表中的数据较少，则不需要将起始地址设置为 X00h，而只需要确认所有查找表数据都位于同一个页面上 。

18Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TABR	TABR							
R/W	R/W							
Reset	0	0	0	0	0	0	0	0

18Ch.7~0 TABR 写 01h =指令 TABRL

1. TABR 写 02h =指令 TABRH
2. 步骤 1 后或步骤 2 读取 TABR 以获得主 ROM 表的读取值
   步骤 1 后读取 TABR 得到 EEPROM 值（当 EEPEN=E2h 时）

表读取的 ASM：指令 TABRL / TABRH 或寄存器 TABR

表读取为 C：使用寄存器 TABR

1.4.1 ALU 和工作(W)寄存器

该 ALU 是 8 位宽的，能够进行加法、减法、移位和逻辑操作。在两个操作数指令中，通常有一个操作数是 W 寄存器，它是一个用于 ALU 操作的 8 位不可寻址寄存器。另一个操作数或者是文件寄存器，或者是直接常数。在单个操作数指令中，操作数是 W 寄存器或文件寄存器。根据所执行的指令，ALU 可能会影响状态寄存器中的携带(C)、数字携带（DC）和零(Z)标志的值。C 和 DC 标志分别作为减法借位和数字借位。

Note: 借位状态和借位值相反
半借位状态和半借位值相反

1.4.2 状态寄存器 (03H/83H/103H/183H)

此寄存器包含 ALU 的算术状态和重置状态。状态寄存器可以是任何指令的目的地，就像任何其他寄存器一样。如果状态寄存器是影响 Z、DC 或 C 位的指令的目的地，那么对这三个位的写入将被禁用。根据设备逻辑设置或清除这些位。因此，建议只使用 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
Reset Value	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R	R	R/W	R/W	R/W
Bit	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

该器件可以通过四种方式进行复位。

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

复位可能由上电复位（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。CFGWH 控制复位功能。复置后，SFR 恢复到其默认值，程序计数器（PC）被清除，系统从复置向量 000H 开始运行。状态寄存器（状态）的 TO 标志指示系统看门狗计时器重位状态。

2.1 上电复位 (POR)

在上电重置后，所有系统和外围控制寄存器将设置为默认硬件重置值。

2.2 低电压复位(LVR)

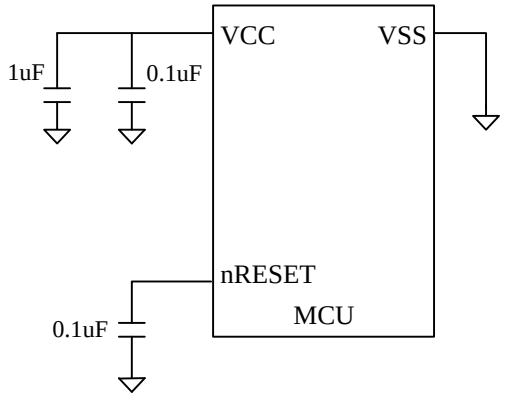
低电压复位的功能是在电源电压低于阈值水平时进行静态复位。可以选择 16 个阈值级别。LVR 的操作模式由 CFGWH 寄存器定义。请参见下面的 LVR 表。用户还必须考虑工作频率的最低工作电压。

CFGWH.11~8	LVR 阈值
0000	低压复位 2.05V
0001	低压复位 2.20V
0010	低压复位 2.30V
0011	低压复位 2.45V
0100	低压复位 2.60V
0101	低压复位 2.75V
0110	低压复位 2.90V
0111	低压复位 3.00V
1000	低压复位 3.15V
1001	低压复位 3.30V
1010	低压复位 3.45V
1011	低压复位 3.60V
1100	低压复位 3.70V
1101	低压复位 3.85V
1110	低压复位 4.00V
1111	低压复位 4.15V

不同的 FSYS 有不同的最低工作电压，参考直流特性的工作电压，如果电流系统电压低于最小工作电压，选择低于 LVR，则系统可能进入死区并发生错误。

2.3 外部引脚复位 (XRST)

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 (WDTR)

可以通过 CFGWH 寄存器禁用或启用 WDT 重置。设置 WDTPSC（81h.3~2），以定义 WDT 复位发生的时间范围。WDT 复位计数器可以通过设备复位或 CLRWDT 指令进行清除。WDT 重置还可以将所有控制寄存器设置为其默认值。 WDT 在不同模式下的行为如下表所示。

模式	CFGWH[13:12]		看门狗 定时器
	WDTE[1]	WDTE[0]	
Normal Mode	0	0	停止
	0	1	停止
	1	0	运行
	1	1	运行
空闲/停止模式 (睡眠)	0	0	停止
	0	1	停止
	1	0	停止
	1	1	运行

看门狗清除由 CLRWDT 指令控制。

◇ 例子：通过 CLRWDT 指令清除看门狗计时器

```
MAIN：  …                       ; 执行程序。
        CLRWDT                  ; 执行CLRWDT指令。
        …
        LGOTO       MAIN
```

◇ 例子： Setup WDT 时间

```
        MOVLW       0000011b
```

```
MOVWX       OPTION                  设置WDT 时间输出=200 ms @4V
…
```

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
Reset	0	0	0	0	0	0	0	0

03h.4 **TO:** WDT 超时标志
0: 重新开机后，或CLRWDT /睡眠 指令后
1: 发生WDT超时

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

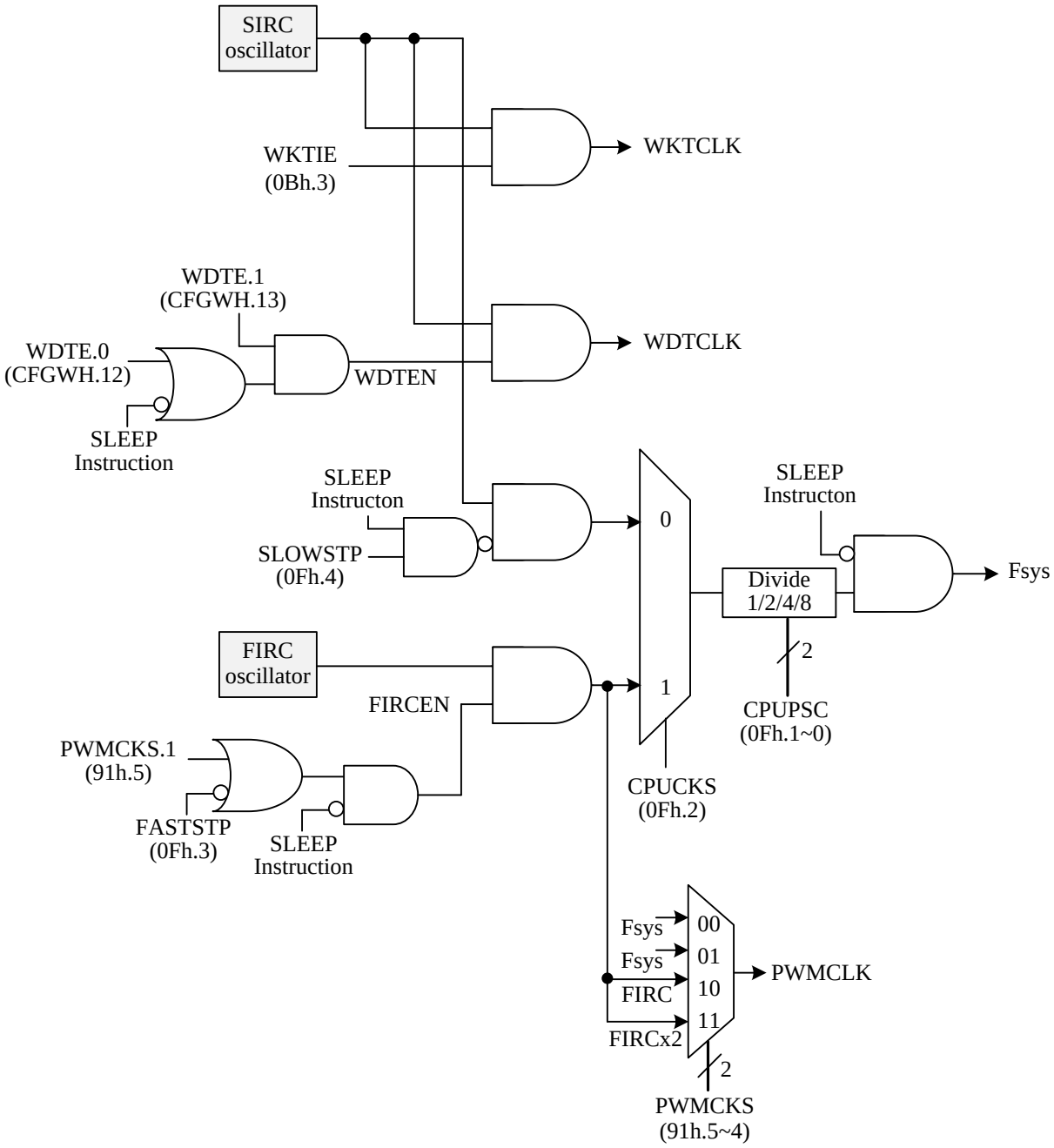
81h.3~2 **WDTPSC:** WDT 周期
00: 100 ms 01: 200 ms 10: 800 ms 11: 1600 ms @VCC=4V

3. 时钟电路和工作模式

3.1 系统时钟

该设备采用双时钟系统进行设计。时钟源有两种，SIRC（慢速内部 RC)时钟和 FIRC（快速内部 RC)时钟。每个时钟源都可以作为一个系统时钟（Fsys）应用于 CPU 核心。

当用户发出睡眠命令时，系统将进入空闲模式或停止模式。当系统根本不使用 SIRC 振荡器时，这是目前最省电的方式，我们称之为停止模式。请参见下图。



Clock Scheme Block Diagram

CPU 重置后，设备以 90 KHz SIRC 的慢速模式运行。用户应选择合适的时钟频率。较高的 VCC 值允许芯片在较高的系统时钟频率下运行。

CLKCTL（0Fh）SFR 控制系统时钟的运行。硬件将自动防止此寄存器的不规范设置。建议用户逐位地编写这个 SFR，并且不要同时设置 FASTSTP=1 和 CPUCKS=1。

FIRC（内部 RC 快时钟）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，所以我们可以在上电后以确保上电复位后 FIRC=16MHz 的频率。

快速模式:

在这种模式下，将使用 FIRC 或 FXT 作为 CPU 时钟 (Fsys) 来执行程序。Timer0，Timer1 块由快时钟驱动。PWM0 块可以通过设置 PWMCKS (91h.5~4)选择 Fsys、FIRC (16 MHz) 或 FIRC*2 (32 MHz) 来驱动。

慢速模式:

开机或复位后，系统时钟（Fsys）默认情况下使用慢时钟，用户可以停止快速时钟（通过 FASTSTP=1，节省电）或运行（通过 FASTSTP=0）。除了 PWM 模块可以选择其他时钟源外，所有外围模块的定时器源（定时器 0、定时器 1 等）的时钟源。此时将使用慢速时钟。

空闲/停止模式:

在用户执行休眠指令后，设备将关闭系统时钟。如果 SIRC 振荡器仍在运行，则称为空闲模式，如果 SIRC 振荡器停止，则称为停止模式。

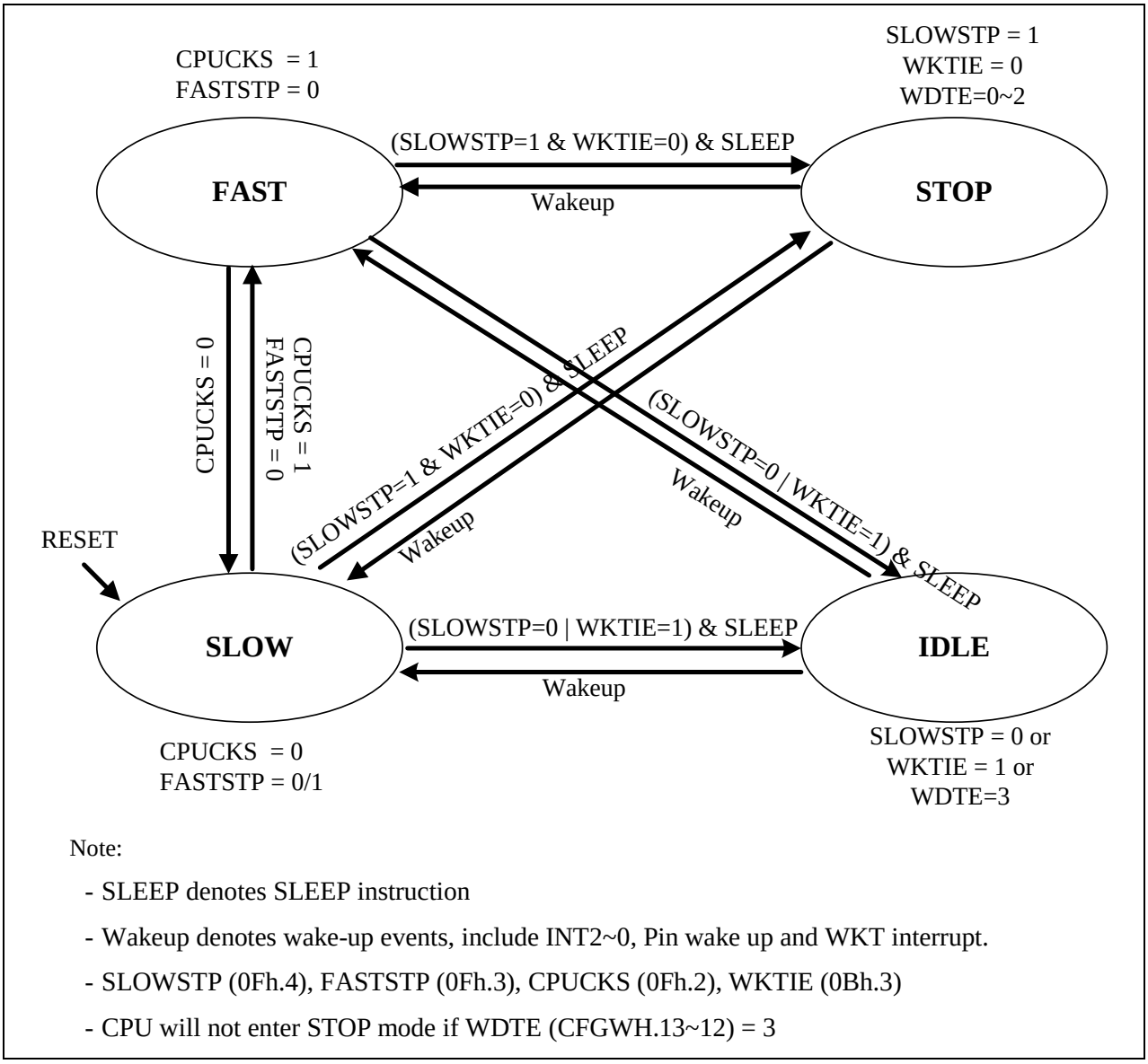
SIRC 振荡器用于以下项目：

1. WKT (controlled by WKTIE)
2. WDT (controlled by WDTE)
3. 系统时钟 (Fsys)

当上述项目全部停止，而慢速=1 时，SIRC 振荡器将停止，此时该设备是最节能的，称为停止模式。

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。



Note:

- SLEEP denotes SLEEP instruction
- Wakeup denotes wake-up events, include INT2~0, Pin wake up and WKT interrupt.
- SLOWSTP (0Fh.4), FASTSTP (0Fh.3), CPUCKS (0Fh.2), WKTIE (0Bh.3)
- CPU will not enter STOP mode if WDTE (CFGWH.13~12) = 3

CPU 工作框图

CPU 模式和 时钟菜单：

模式	FIRC 震荡	SIRC 震荡	Fsys/ TM0/TM1	WKT	WDT	唤醒事件
FAST	运行	运行	运行	运行	运行	-
SLOW	Set by FASTSTP	运行	运行	运行	运行	-
IDLE	停止	运行	停止	Set by WKTIE	Set by WDTE	WKT/IO
STOP	停止	停止	停止	停止	停止	IO

- **快速模式切换至慢速模式**

当 FAST 模式切换到慢速模式时，建议按顺序执行以下步骤：

(1) 切换到慢时钟 (CPUCKS=0)

(2) 停止快速时钟(FASTSTP=1)

◇ 示例：将快速模式切换到慢速模式

```
BCX     CPUCKS      ; Fsys=慢时钟
BSX     FASTSTP     ; 停止快时钟
```

- **慢速模式切换到快速模式**

SLOW 模式可启用 CPUCKS=0。当 SLOW 模式切换到 FAST 模式时，建议按顺序执行以下步骤：

(1) 使能 Fast-clock (FASTSTP=0)

(2) 切换到快时钟 (CPUCKS=1)

◇ 示例：将慢速模式切换到 FAST 模式（快速时钟停止）

```
BCX     FASTSTP     ; 启用快速时钟
NOP
BSX     CPUCKS      ; Fsys=快速时钟
```

- **IDLE/STOP 模式设置**

IDLE 模式可以按顺序设置进行配置：

(1) 启用 WKT (WKTIE=1)

(2) 执行睡眠指令

空闲模式可以被外部中断、引脚唤醒或 WKT 中断唤醒。

◇ 示例：将 FAST/SLOW 模式切换到空闲模式

```
BSX     WKTIE       ; 启用WKT计数
SLEEP               ;进入IDLE模式
```

- **停止模式设置**

停止模式可以通过以下设置按顺序进行配置：

(1) 停止慢时钟 (SLOWSTP=1)

(2) 停止 WKT (WKTIE=0)

(3) 执行睡眠指令

停止模式只能通过外部引脚中断来唤醒.

备注：当 WDTE=11b 时，CPU 无法进入停止模式

◇ 示例：将快速/慢速模式切换到停止模式

```
BSX     SLOWSTP     ; 在执行睡眠指令后禁用慢时钟
BCX     WKTIE       ;禁用WKT计数
SLEEP               ; 进入停止模式.
```

0Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE	-	-	TM1IE	TM0IE	WKTIE	INT2IE	INT1IE	INT0IE
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	0	0	0	0	0	0

0Bh.3 **WKTIE:** 唤醒计时器中断启用和唤醒计时器启用
0: 关闭
1: 使能够

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	
Reset	–	–	–	0	1	0	1	1

0Fh.4 **SLOWSTP:** 在 SLEEP 指令后停止慢时钟
0: 慢时钟在 SLEEP 指令后持续运行
1: 慢时钟在 SLEEP 指令后停止运行

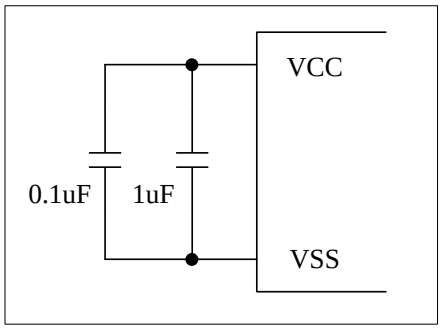
0Fh.3 **FASTSTP:** 停止快时钟
0: 快时钟运行
1: 快时钟停止

0Fh.2 **CPUCKS:** 系统时钟源选择
0: 慢时钟
1: 快时钟

0Fh.1~0 **CPUPSC**: 系统时钟源预分频器。系统时钟源
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 16 MHz 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。



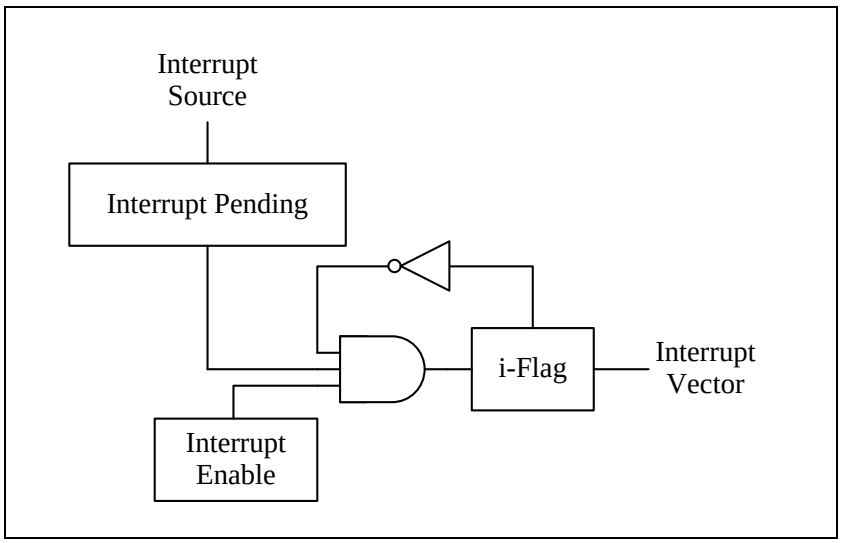
Internal RC Mode

4. 中断

TM56M1511 有 1 级、1 个向量和 9 个中断源。每个中断源都有它自己的使能控制位。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。

如果相应的中断启用位(INTIE。5~0，INTIE1.2~0)已经设置好，它将触发 CPU 来服务中断。CPU 在当前执行的指令周期结束时接受中断。同时，向 CPU 插入“LCALL 004“指令，并设置 i-flag 以防止递归中断嵌套。

i-flag 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

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

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

                ORG     005h
START:
                MOVLW   0000xxxxb
                MOVWX   PAMOD10         ; 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   xx1xxxxxb
                MOVWX   OPTION          ; 将INT1中断触发设置为上升沿

                MOVLW   111111 0 1b
                MOVWX   INTIF           ; 清除INT1中断请求标志

                MOVLW   000000 1 0b
                MOVWX   INTIE           ; 使能INT1中断
MAIN:
                …
                LGOTO   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

                BTXSC   INT1IF          ; 检测INT1IF位
                LCALL   INT1_SUBROUTINE ; 如果 INT1IF = 1，跳转到 INT1 子程序

                …                       ; INT1中断服务程序
                MOVXW   21h
                MOVWX   STATUS          ; 清除INT1中断请求标志

                MOVXW   20h
                RETI                    ; 从中断返回

INT1_SUBROUTINE:


                …
```

```
MOVLW     111111 0 1b
MOVWX     INTIF              ; 清除 INT1 中断请求标志
RET
```

0Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE	-	-	TM1IE	TM0IE	WKTIE	INT2IE	INT1IE	INT0IE
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	0	0	0	0	0	0

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

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

0Bh.3 **WKTIE:** WKT中断使能
0: 关闭
1: 使能

0Bh.2 **INT2IE:** INT2 中断使能
0: 关闭
1: 使能

0Bh.1 **INT1IE:** INT1 中断使能
0: 关闭
1: 使能

0Bh.0 **INT0IE:** INT0 中断使能
0: 关闭
1: 使能

0Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIF	-	-	TM1IF	TM0IF	WKTIF	INT2IF	INT1IF	INT0IF
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	0	0	0	0	0	0

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	–	–	–	–	–	TKIE	PWMIE	LVDIE
R/W	–	–	–	–	–	R/W	R/W	R/W
Reset	–	–	–	–	–	0	0	0

0Dh.2 **TKIE**:允许按键中断
0:关闭
1:启用

0Dh.1 **PWMIE**: PWM0~5中断启用
0:关闭
1:启用

0Dh.0 **LVDIE**:使能LVD中断
0:关闭
1:启用

0Eh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIF1	–	–	–	–	–	TKIF	PWMIF	LVDIF
R/W	–	–	–	–	–	R/W	R/W	R/W
Reset	–	–	–	–	–	0	0	0

0Eh.2 **TKIF**:触键中断事件挂起标志
此位由H/W设置，在触摸键转换结束后，将0写入此位将清除此标志

0Eh.1 **PWMIF**: PWM中断事件等待标志
该位在PWM周期计数器翻转后由H/W设置，将0写入该位将清除该标志

0Eh.0 **LVDIF**: LVD中断事件等待标志
该位在LVDO=1后由H/W设置，将0写入该位将清除该标志

5. I/O 端口

5.1 PA0-PA4, PA7

每个 IO 引脚都有 4 位作为模式设置。模式设置包括以下功能：开漏输出、CMOS 输出、上拉电阻、下拉电阻、引脚改变唤醒、PWMO 等。 这些引脚可以在不同的模式下操作。

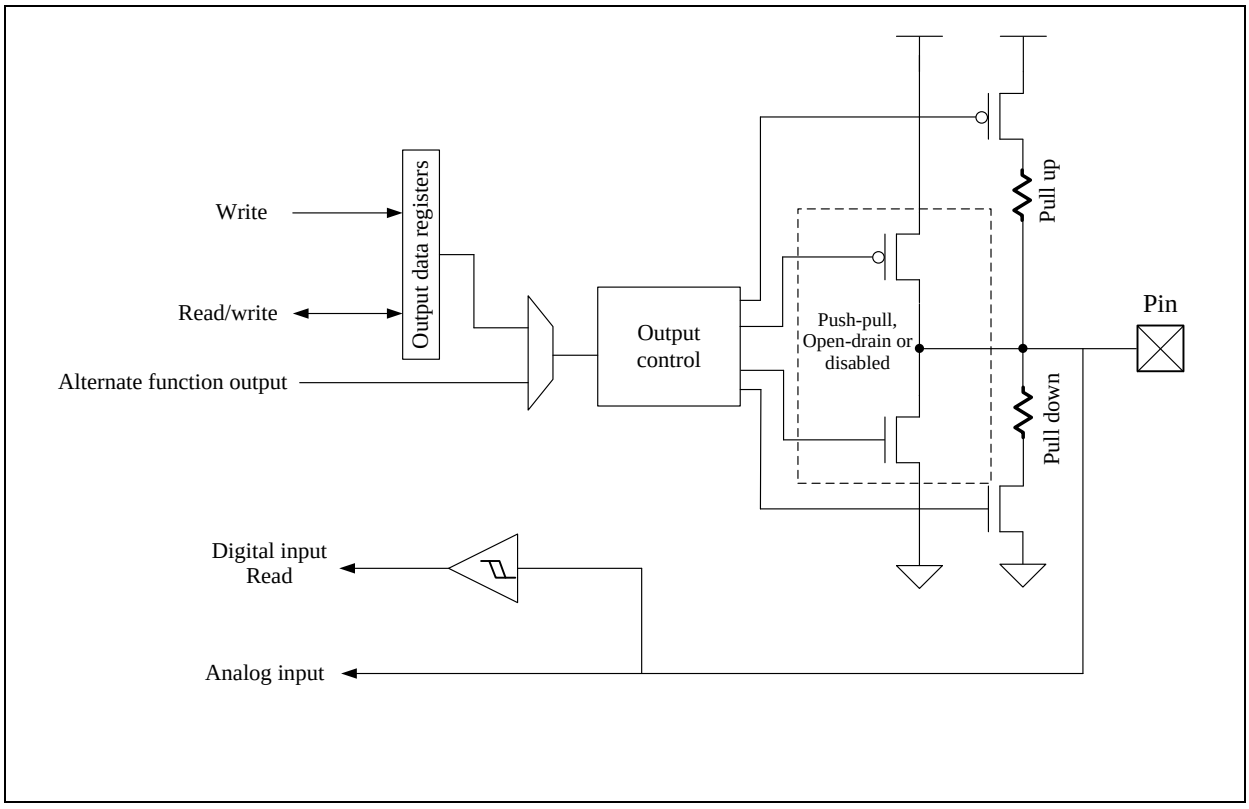
PA7 不提供下拉电阻、1/2 偏置、PWMO 和 CTKCKO 等功能，所以 PA7 的 PIN 模式设置请不要选择 xx11b 或 x100b。

PAMOD	PAD	Pin Function	引脚状态	上拉电阻	下拉电阻	数字输入	引脚改变唤醒
0000b	0	开漏	低驱动	-	-	-	-
	1	输入	上拉	Y	-	Y	-
0001b	0	开漏	低驱动	-	-	-	-
	1		高组	-	-	Y	-
0010b	0	CMOS 输出	低驱动	-	-	-	-
	1		高驱动	-	-	-	-
0011b	X	Tenx 保留	高组	-	-	-	-
0100b	0	开漏	低驱动	-	-	-	-
	1	Input	下拉	-	Y	Y	-
0101b	0	开漏	低驱动	-	-	-	-
	1		高组	-	-	Y	-
0110b	0	CMOS 输出	低驱动	-	-	-	-
	1		高驱动	-	-	-	-
0111b	X	PWMO的CMOS输出	-	-	-	-	-
1000b	0	Open Drain	低驱动	-	-	-	-
	1	Input	上拉	Y	-	Y	Y
1001b	0	开漏	低驱动	-	-	-	-
	1		高组	-	-	Y	Y
1010b	0	CMOS 输出	低驱动	-	-	-	-
	1		高驱动	-	-	-	-
1011b	X	PWMO的CMOS输出	-	-	-	-	-
1100b	0	开漏	低驱动	-	-	-	-
	1	Input	下拉	-	Y	Y	Y
1101b	0	开漏	低驱动	-	-	-	-
	1		高组	-	-	Y	Y
1110b	0	CMOS 输出	低驱动	-	-	-	-
	1		高驱动	-	-	-	-
1111b	X	模拟输出 1/2偏压（1/2 VCC）	-	-	-	-	-

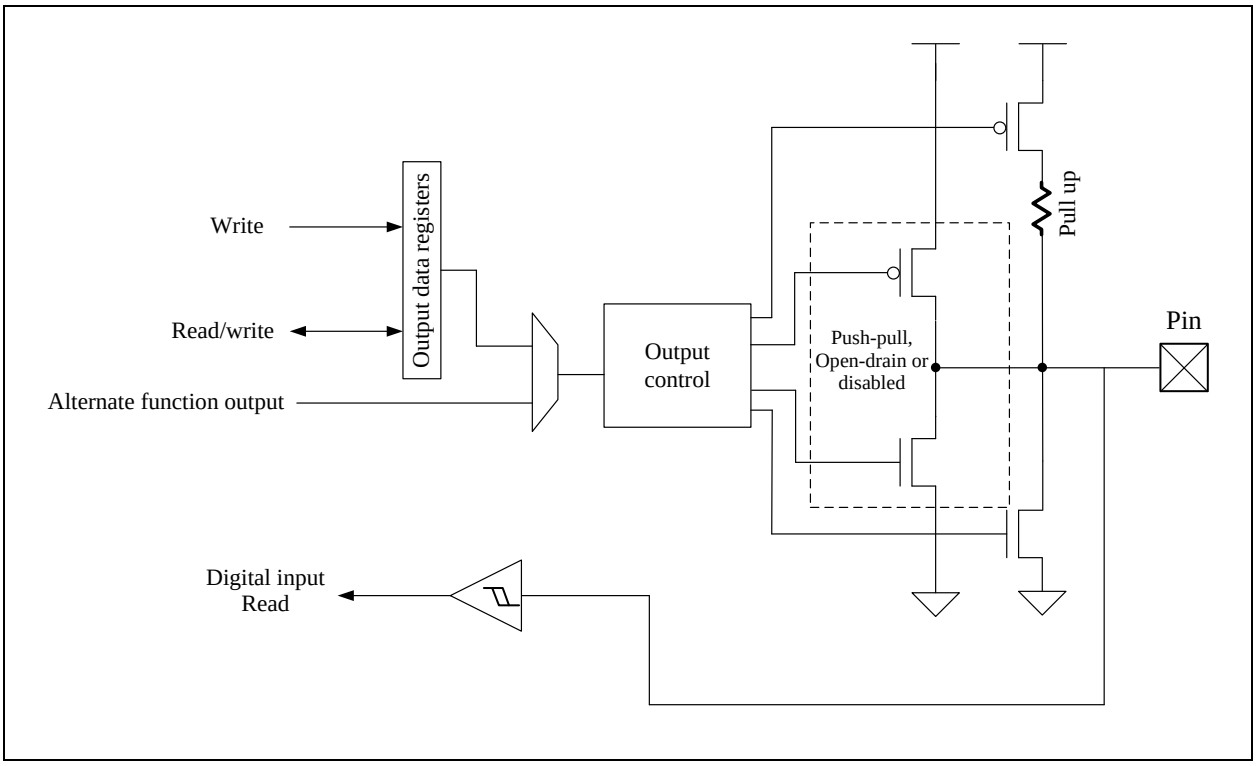
通用 I/O Pin 功能表

引脚名称	PAMOD					
	0111b (数字输出)	1011b (数字输出)	1111b (模拟输出)	xx10b & PADx=1	xx10b & PADx=0	xx10b & PADx=0
PA0	PWM5O	CTKCKO	1/2 bias	TK5 (CTK)	TK5 (STK)	
PA1	PWM1O	CTKCKO	1/2 bias	TK1 (CTK)	TK1 (STK)	
PA2	PWM4O	CTKCKO	1/2 bias	TK4 (CTK)	TK4 (STK)	
PA3	PWM2O		1/2 bias			CLD
PA4	PWM0O	CTKCKO	1/2 bias	TK2 (CTK)	TK2 (STK)	
PA7						

引脚表的特殊功能



一般引脚结构(不包括 PA7)


Write
Read/write
Alternate function output
Output data registers
Output control
Push-pull, Open-drain or disabled
Pull up
Pin
Digital input Read

PA7 引脚结构

85h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PAMOD10	PA1MOD				PA0MOD			
R/W	R/W				R/W			
Reset	0	0	0	1	0	0	0	1

86h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PAMOD32	PA3MOD				PA2MOD			
R/W	R/W				R/W			
Reset	0	0	0	1	0	0	0	1

87h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PAMOD54	PA5MOD				PA4MOD			
R/W	R/W				R/W			
Reset	0	0	0	1	0	0	0	1

88h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PAMOD76	PA7MOD				PA6MOD			
R/W	R/W				R/W			
Reset	0	0	0	0	0	0	0	1

88h.7~4 **PA7MOD ~ PA0MOD**: PA7, PA4~PA0 引脚模式控制
87h.3~0
86h.7~4
86h.3~0
85h.7~4
85h.3~0

0000: 开漏或数字输入带上拉
0001:开漏
0010: 推挽输出
0011: 模拟输入
0100: 开漏下拉
0101: 开漏
0110: 推挽
0111: 外设功能输出
1000: 开漏或数字输入带上拉和引脚改变唤醒
1001: 开漏或数字输入和引脚改变唤醒
1010: CMOS 推挽输出
1011: CTKCKO 输出
1100: 开漏或数字输入带下拉和引脚改变唤醒
1101: 开漏或数字输入和引脚改变唤醒
1110: 推挽
1111:1/2 分压

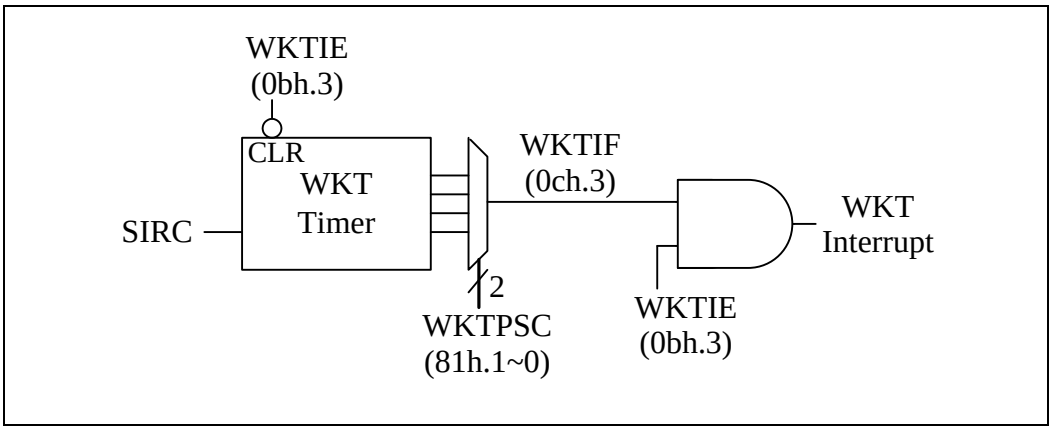
05h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PAD	PAD							
R/W	R/W							
Reset	1	1	1	1	1	1	1	1

05h.7~0 **PAD**: PA7, PA4~PA0 数据

6. 外围功能模块

6.1 唤醒定时器(WKT)

WKT 是一个间隔计时器，间隔长度可以通过 WKTPSC（81h.1~0）进行调整，WKT 中断标志(WKTIF)。WKT 定时器由 WKTIE=0 清除/停止。由于 WKT 具有功耗，所以 WKTIE=0 是进入停止模式需要配置。



WKT 框图

◇ 示例：设置 WKT 周期和中断功能

```
MOVLW   000001<u>**10**</u>b
MOVWX   OPTION          ; 选择WKT周期=50 ms @4V
MOVLW   1111<u>**0**</u>111b    ; 通过使用字节操作来清除WKT中断标志
MOVWX   INTIF           ; 不要使用位操作“BCX WKTIF”来清除
BSX     WKTIE           ; 启用WKT中断功能
```

0Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIF	-	-	TM1IF	TM0IF	WKTIF	INT2IF	INT1IF	INT0IF
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	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	-	-	TM1IE	TM0IE	WKTIE	INT2IE	INT1IE	INT0IE
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	0	0	0	0	0	0

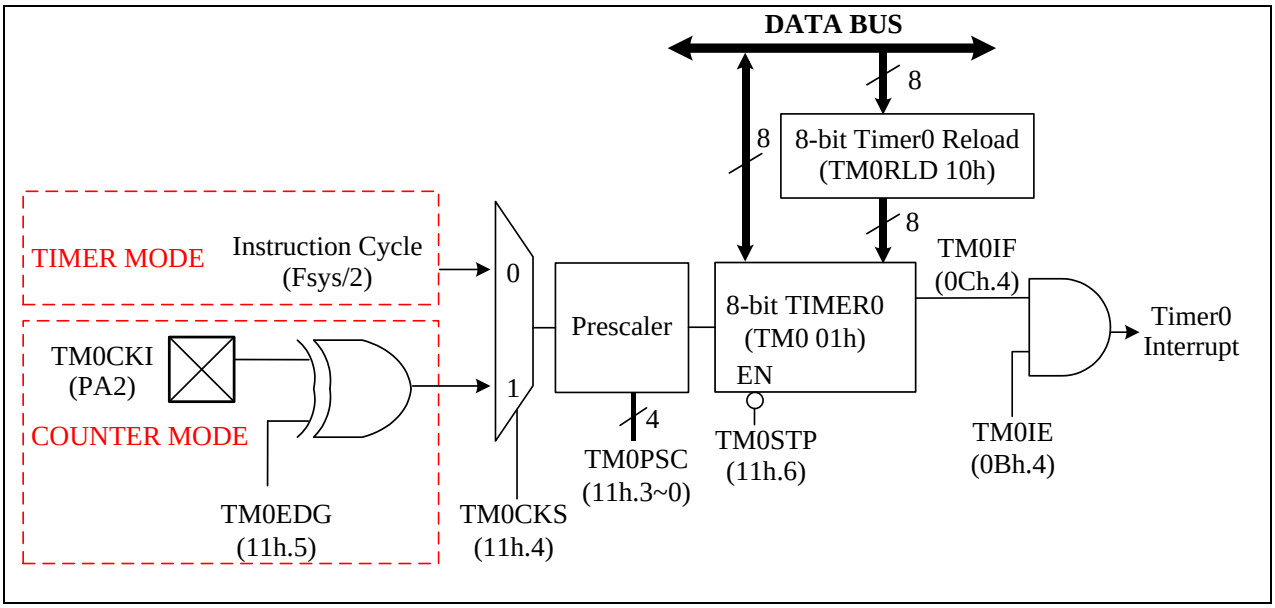
0Bh.3 **WKTIE:** WKT中断使能
0: 关闭
1: 使能

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

81h.1~0 **WKTPSC:** WKT 周期
00: 12 ms 01: 25 ms 10: 50 ms 11: 100 ms @ VCC=4V

6.2 Timer0

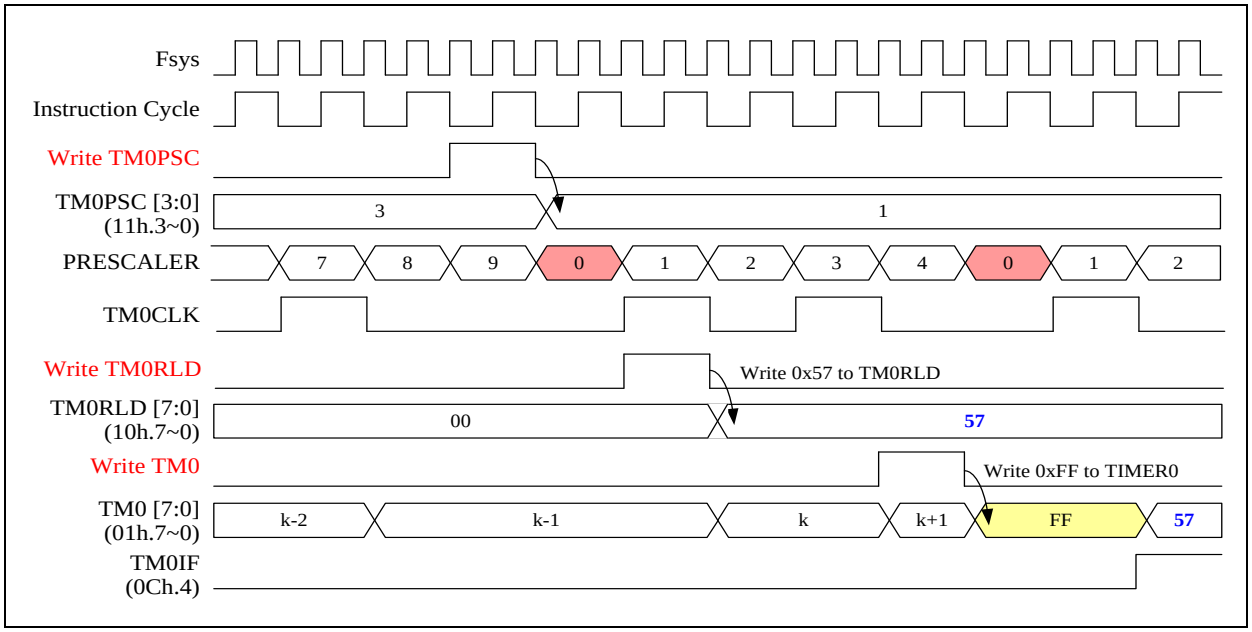
TM0 (01h.7~0) 是一个 8 位宽的寄存器。和其他任何寄存器一样可以读取或写入。此外，Timer0 会周期性地增加自身并自动翻转一个新的“偏移值”(TM0RLD)，同时它会根据预先调整的时钟源进行翻转，该时钟源可以是 Fsys/2 或 TM0CKI (PA2) 上升/下降输入。Timer0 的增加速率由“Timer0 预分频” (TM0PSC) 寄存器决定。当 Timer0 的计数翻转时，它总是产生 TM0IF (0Ch.4)。如果 TM0IE (0Bh.4) 置位，它会产生 Timer0 中断。如果 TM0STP (11h.6) 位置位，Timer0 可以停止计数。



Timer0 框图

下面的时序图描述了计时器 0 在纯计时器模式下工作。

当写入计时器 0 预调节器（TM0PSC）时，内部 8 位预调节器将被清除为 0，以便在第一次计时器 0 计数时计数周期正确。TM0CLK 是在 TM0CLK 结束时导致计时器 0 增加 1 的内部信号。TM0WR 也是内部信号，表示计时器 0 直接由指令写入，同时清除内部 8 位预压器。当计时器 0 从 FFh 计算到 TM0RLD 时，TM0IF(计时器 0 中断标志）将设置为 1，如果设置了 TM0IE（计时器 0 中断启用），则生成中断。



计时器 0 以计时器模式工作 (TM0CKS=0)

TM0 中断时间值方程如下：

◇ 示例：设置计时器 0 在计时器模式下工作，如果 Fsys = 8 MHz

```
; 设置计时器0时钟源和分频器
        MOVLW   00000101b       ; TM0CKS = 0，计时器0时钟源为指令周期（Fsys/2）
        MOVWX   TM0CTL          ; TM0PSC = 0101b，除以32

; 设置定时器0重新加载数据
        MOVLW   80h
        MOVWX   TM0RLD          ; 设置计时器0，重新加载数据= 128

; 设置定时器0
        BSX     TM0STP          ; 定时器0停止计数
        CLRX    TM0             ; 清除定时器0内容

; 启用计时器0和中断功能
        MOVLW   1110 1111b
        MOVWX   INTIF           ; 清除计时器0请求中断标志
        BSX     TM0IE           ; 启用Timer0中断功能
        BCX     TM0STP          ; 启用计时器0计数
```

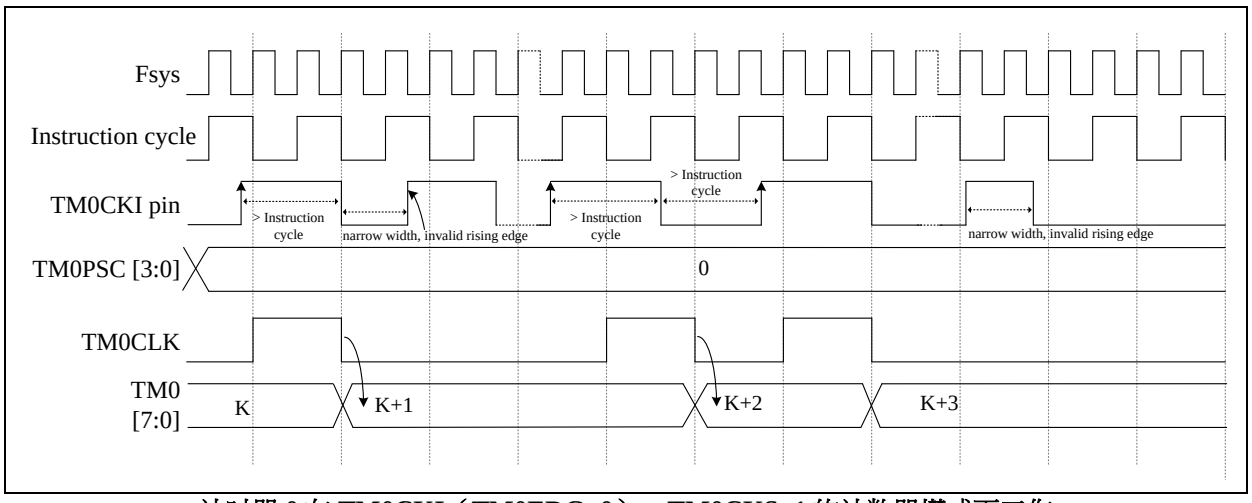
定时器 0 中断频率 = 定时器 0 时钟频率 / (256-TM1RLD)

= (Fsys / 2 / TM0PSC) / (256-TM0RLD)

= (128 KHz) / (256-128) = 1 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。



计时器 0 在 TM0CKI（TM0EDG=0）、TM0CKS=1 的计数器模式下工作

◇示例：设置 TM0 在计数器模式下工作，从 TM0CKI 针（PA2）

```
; Setup Timer0 clock source and divider
        MOVLW   00110000B       ; TM0EDG = 1，计数边为下降沿
        MOVWX   TM0CTL          ; TM0CKS = 1, 计时器0时钟为TM0CKI
                                ; TM0PSC = 0000b,除以1
; Setup Timer0
        BSX     TM0STP          ; 定时器0停止计数
        CLRX    TM0             ; 清除定时器0内容

; Enable Timer0 and read Timer0 counter
        BCX     TM0STP          ; 启用计时器0计数
        …
        BSX     TM0STP          ; 定时器0停止计数
        MOVXW   TM0             ; 定时器0停止计数
```

01h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TM0	TM0							
R/W	R/W							
Reset	0	0	0	0	0	0	0	0

01h.7~0 **TM0:** Timer0内容

0Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE	-	-	TM1IE	TM0IE	WKTIE	INT2IE	INT1IE	INT0IE
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	0	0	0	0	0	0

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

0Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIF	-	-	TM1IF	TM0IF	WKTIF	INT2IF	INT1IF	INT0IF
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	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							
Reset	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	–	TM0STP	TM0EDG	TM0CKS	TM0PSC			
R/W	–	R/W	R/W	R/W	R/W			
Reset	–	0	0	0	0	0	0	0

11h.6 **TM0STP:** Timer0计数器停止
0:释放1:停止计数

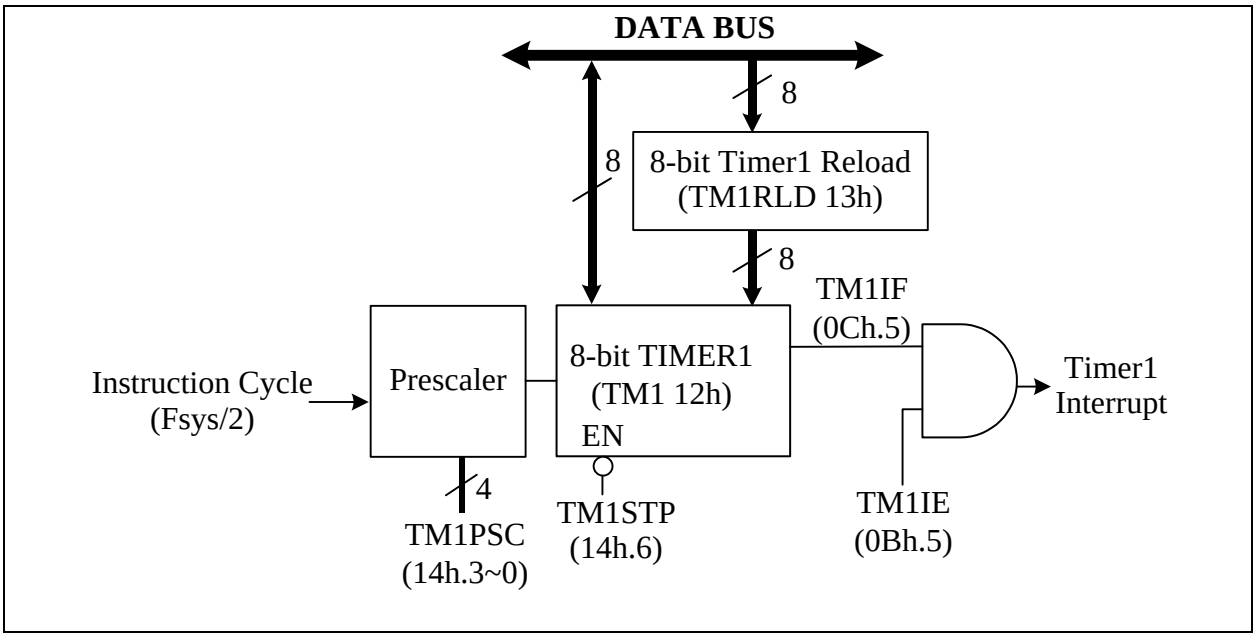
11h.5 **TM0EDG:** TM0CKI 引脚的 Timer0 预分频器计数沿
0: 上升沿 1: 下降沿

11h.4 **TM0CKS:** Timer0 预分频器时钟源
0:Fsys/2 1: TM0CKI引脚(PA2 引脚)

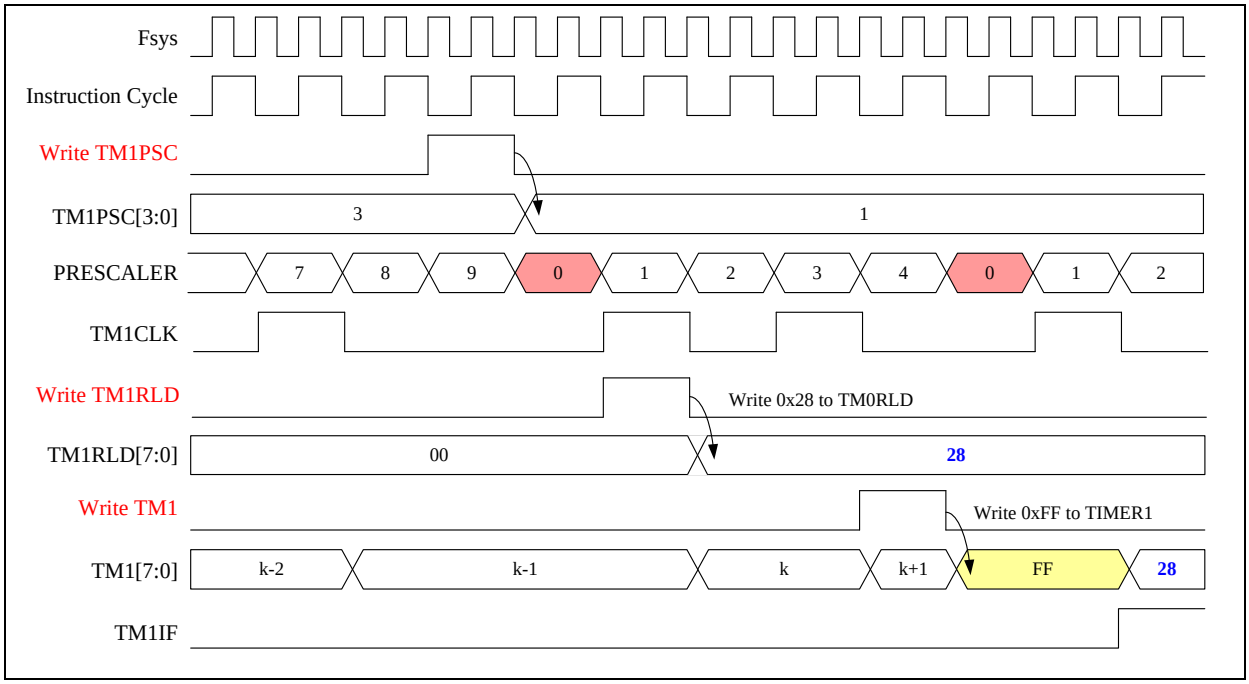
11h.3~0 **TM0PSC:**Timer0 预分频器。 Timer0 时钟源（Fsys/2 或 TM0CKI）除以
0000: 1 0011: 8 0110: 64
0001: 2 0100: 16 0111: 128
0010: 4 0101: 32 1xxx: 256

6.3 Timer1

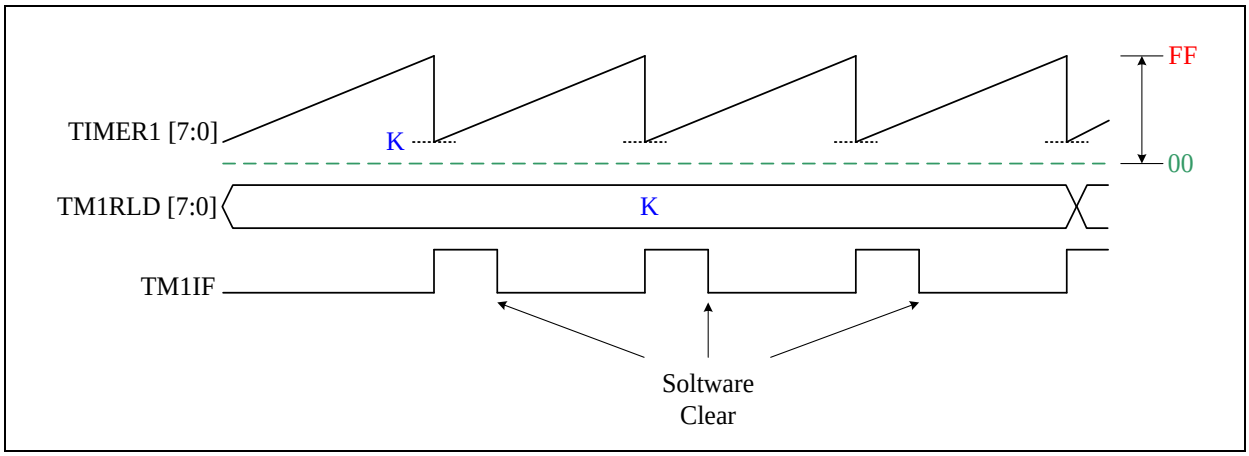
TM1 (12h.7~0) 是一个 8 位宽的寄存器。和任何其他寄存器一样可以读取或写入。此外，Timer1 会定期自行递增，并根据预分频的指令时钟 (Fsys/2) 在翻转时自动重载新的“偏移值” (TM1RLD）。Timer1 的增加速率由 TM1PSC 寄存器选择。如果 TM1IE 位置 1，它将产生 Timer1 中断。如果 TM1STP 位置 1，可以停止 Timer1 的计数。



Timer1 框图



Timer1 定时器图



Timer1 重新加载图

TM1 中断时间值方程如下：

◇ 示例：设置计时器 1 在计时器模式下工作，如果 Fsys = 8 MHz

```
; 设置计时器：1个时钟源和分频器
        MOVLW   00000101b       ; 计时器1时钟源为指令周期（Fsys/2）
        MOVWX   TM1CTL          ; TM1PSC = 0101b, 除以32

; 设置定时器1重新加载数据
        MOVLW   FFh
        MOVWX   TM0RLD          ; 设置计时器1，重新加载数据= 255

; 设置时间1
        BSX     TM1STP          ; 定时器1停止计数
        CLRX    TM1             ; 清除定时器1内容

; 启用定时器1和中断功能
        MOVLW   11011111b
        MOVWX   INTIF           ; 清除计时器1请求中断标志
        BSX     TM1IE           ; 启用Timer1中断功能
        BCX     TM1STP          ; 启用计时器1计数
```

定时器 1 中断频率 = 定时器 1 时钟频率 / (256-TM1RLD)

= (Fsys / 2 / TM0PSC) / (256-TM0RLD)

= (128 KHz) / (256-128) = 1 KHz

0Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE	-	-	TM1IE	TM0IE	WKTIE	INT2IE	INT1IE	INT0IE
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	0	0	0	0	0	0

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

0Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIF	-	-	TM1IF	TM0IF	WKTIF	INT2IF	INT1IF	INT0IF
R/W	-	-	R/W	R/W	R/W	R/W	R/W	R/W
Reset	-	-	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							
Reset	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							
Reset	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	–	TM1STP	–	–	TM1PSC			
R/W	–	R/W	–	–	R/W			
Reset	–	0	–	–	0	0	0	0

14h.3~6 **TM1STP:** Timer1 计数器停止
0: 计数器运行
1: 计数器停止

14h.3~0 **TM1PSC:** Timer1 预分频器。 Timer1 时钟源 (Fsys/2) 除以

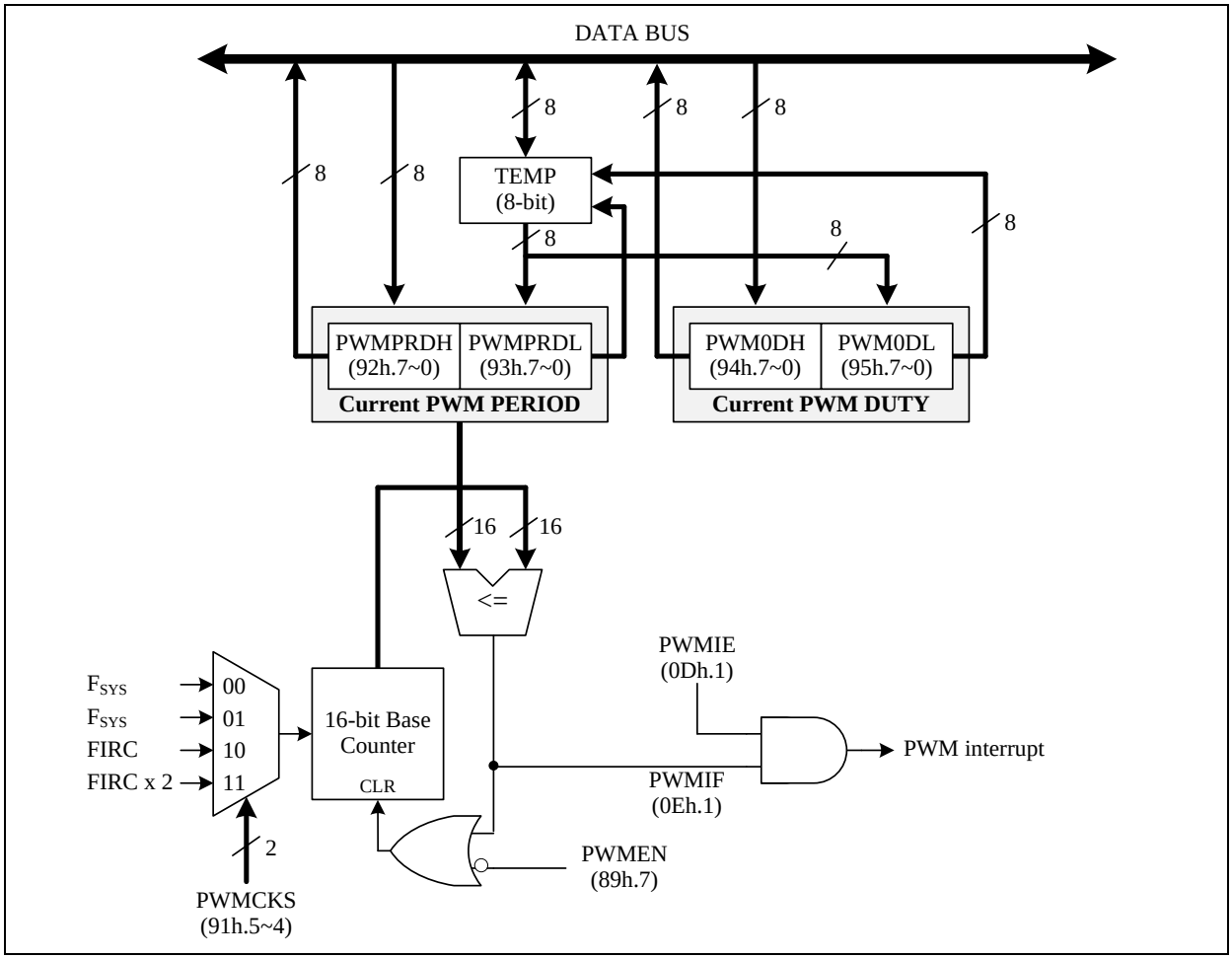
0000: 1	0011: 8	0110: 64
0001: 2	0100: 16	0111: 128
0010: 4	0101: 32	1xxx: 256

6.4 PWM: 16 bits PWM

在这个芯片中有五个 pwm。所有的 PWM 都有独立的 16 位占空比控制寄存器，并共享一组 16 位周期寄存器。PWM 可以在 PWM 时钟的基础上产生具有 65536 占空比分辨率的变化频率波形。PWM 时钟可以选择 FIRC*2、FIRC 或 FSYS 作为其时钟源。

16 位 PWMPRD、PWM0D 寄存器都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的 8 位缓冲区访问，必须以特定的方式对这些寄存器对进行读取或写。需要注意的一点是，8 位缓冲区及其相关低字节的数据传输只有在执行相应高字节的写或读操作时才会发生。**简单地说，先写低字节，然后写高字节；先读高字节，然后读低字节。**

如果清除了 PWMEN，则将清除并停止 PWM0~5，否则 PWM0~5 将继续运行。PWM0 的结构如下所示。PWM0 的占空比可以通过写入 PWM0DH 和 PWM0DL 来更改。当 16 位基计数器与 16 位 PWM0 占空比寄存器 {PWM0DH，PWM0DL} 匹配时，PWM0 输出信号将重置到低电平。PWM0 周期可以通过将周期值写入 PWMPRDH 和 PWMPRDL 寄存器来设置。在写入 PWM0DH 或 PWMPRDH 寄存器后，H/W 将立即更新 PWM 周期。PWM0~5 共享一个中断标志，并在该期间结束时生成一个中断标志。



PWM0 框图

0Dh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE1	–	–	–	–	–	TKIE	PWMIE	LVDIE
R/W	–	–	–	–	–	R/W	R/W	R/W
Reset	–	–	–	–	–	0	0	0

0Dh.1 **PWMIE:** PWM中断允许
0: 关闭
1: 使能

0Eh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIF1	–	–	–	–	–	TKIF	PWMIF	LVDIF
R/W	–	–	–	–	–	R/W	R/W	R/W
Reset	–	–	–	–	–	0	0	0

0Eh.1 **PWMIF**: PWM中断事件等待标志
该位在PWM周期计数器翻转后由H/W设置，将0写入该位将清除该标志

89h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWMCTL	PWMEN	–	–	–	–	–	–	–
R/W	R/W	–	–	–	–	–	–	–
Reset	0	–	–	–	–	–	–	–

89h.7 **PWMEN:** PWM 使能
0: 关闭
1: 使能

91h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPTION2	–	–	PWMCKS		–	–	–	–
R/W	–	–	R/W		–	–	–	–
Reset	–	–	0	0	–	–	–	–

91h.5~4 **PWMCKS:** PWM clock 输入源选择
00: F_{SYS}
01: F_{SYS}
10: FIRC
11: FIRC x 2 (VCC>2.3V@25℃)

92h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWMPRDH	PWMPRDH							
R/W	R/W							
Reset	1	1	1	1	1	1	1	1

92h.7~0 **PWMPRDH:** PWM 周期高字节
写顺序:PWMPRDL 然后 PWMPRDH
读顺序:PWMPRDH 然后 PWMPRDL

93h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWMPRDL	PWMPRDL							
R/W	R/W							
Reset	1	1	1	1	1	1	1	1

93h.7~0 **PWMPRDL:** PWM 周期低字节
写顺序:PWMPRDL 然后 PWMPRDH
读顺序:PWMPRDH 然后 PWMPRDL

94h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM0DH	PWM0DH							
R/W	R/W							
Reset	1	0	0	0	0	0	0	0

94h.7~0 **PWM0DH:** PWM0 占空高字节
写顺序:PWMxDL 先 PWMxDH
读顺序:PWMxDH 后 PWMxDL

95h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM0DL	PWM0DL							
R/W	R/W							
Reset	0	0	0	0	0	0	0	0

95h.7~0 **PWM0DL:**PWM0 占空低字节
写顺序:PWMxDL 然后 PWMxDH
读顺序:PWMxDH 然后 PWMxDL

96h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM1DH	PWM1DH							
R/W	R/W							
Reset	1	0	0	0	0	0	0	0

96h.7~0 **PWM1DH**: PWM1 占空高字节
写顺序:PWMxDL 然后 PWMxDH
读顺序:PWMxDH 然后 PWMxDL

97h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM1DL	PWM1DL							
R/W	R/W							
Reset	0	0	0	0	0	0	0	0

97h.7~0 **PWM1DL**: PWM1 占空低字节
写顺序:PWMxDL 然后 PWMxDH
读顺序:PWMxDH 然后 PWMxDL

98h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM2DH	PWM2DH							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	0	0	0	0	0

98h.7~0 **PWM2DH**: PWM2 占空高字节
写顺序:PWMxDL 然后 PWMxDH
读顺序:PWMxDH 然后 PWMxDL

99h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM2DL	PWM2DL							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

99h.7~0 **PWM2DL**: PWM2 占空低字节
写顺序:PWMxDL 然后 PWMxDH
读顺序:PWMxDH 然后 PWMxDL

9Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM4DH	PWM4DH							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	0	0	0	0	0

9Ch.7~0 **PWM4DH:** PWM4 占空高字节
写顺序:PWMxDL 然后 PWMxDH
读顺序:PWMxDH 然后 PWMxDL

9Dh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM4DL	PWM4DL							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

9Dh.7~0 **PWM4DL:** PWM4 占空低字节
写顺序:PWMxDL 然后 PWMxDH
读顺序:PWMxDH 然后 PWMxDL

9Eh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM5DH	PWM5DH							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	0	0	0	0	0	0	0

9Eh.7~0 **PWM5DH:** PWM5 占空高字节
写顺序:PWMxDL 然后 PWMxDH
读顺序:PWMxDH 然后 PWMxDL

9Fh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM5DL	PWM5DL							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

9Fh.7~0 **PWM5DL:** PWM5 占空低字节
写顺序:PWMxDL 然后 PWMxDH
读顺序:PWMxDH 然后 PWMxDL

6.5 触摸键 (仅限 M1531)

该设备支持 4 个通道的触摸按键检测。触摸键提供了两种简单、简单可靠的方法来实现手指触摸检测。一个结构是 STK，另一个结构是 CTK。在大多数应用程序中，它不需要任何在 STK 模式下的外部组件。即使在 CTK 模式下，它也只需要一个外部电容组件（CLD）。

TK 扫描期间的针脚状态由硬件自动切换，而 TK 空闲期间的针脚状态需要由用户设置。强烈建议按下表所示的方式设置引脚。详情请参见第 5 节。

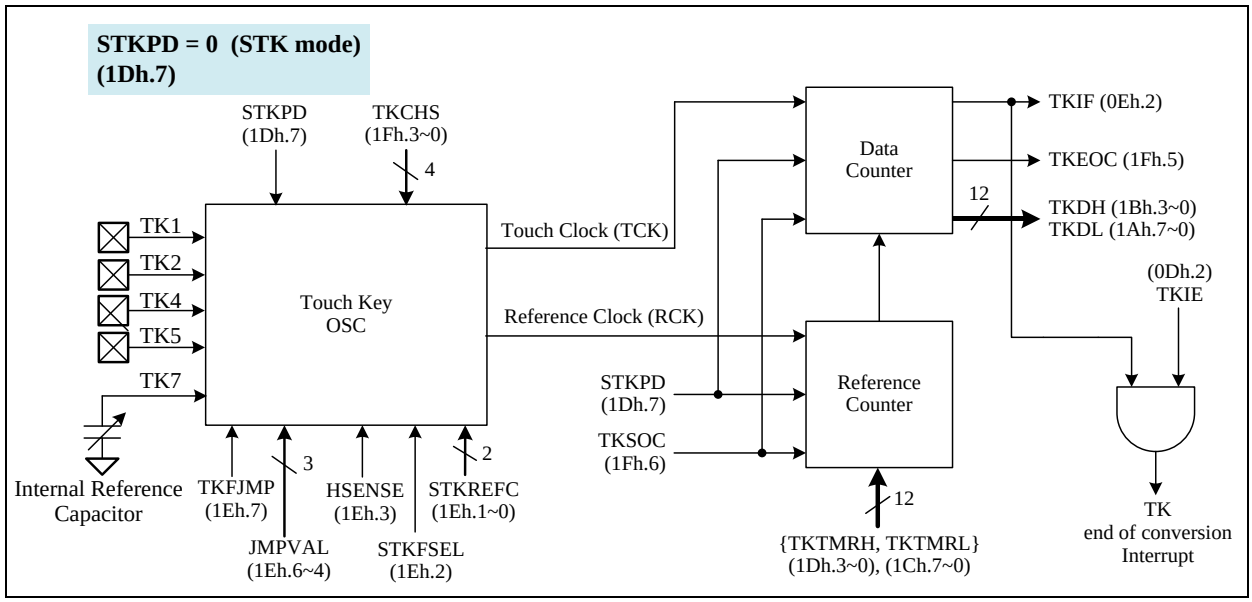
通过这些引脚设置，可以减少 TK 键的相互影响，并可以加速外部电容 CLD 的放电速度。

Pin Name	引脚模式设置			
	PAMOD=1011b (数字输出)	PAMOD=xx10b & PAD=1	PAMOD=xx10b & PAD=0	PAMOD=xx10b & PAD=0
PA0	CTKCKO	TK5 (CTK)	TK5 (STK)	
PA1	CTKCKO	TK1 (CTK)	TK1 (STK)	
PA2	CTKCKO	TK4 (CTK)	TK4 (STK)	
PA3				CLD
PA4	CTKCKO	TK2 (CTK)	TK2 (STK)	
PA7				

6.5.1 STK

设置 STKPD=0，您可以选择放松振动结构，触摸按钮。在 STK 模式下，有两个振荡器：参考时钟（RCK）和触摸时钟（TCK）。它们分别连接到参考计数器和数据计数器。RCK 的频率可以通过设置 STKREFC 来调整，参考计数器用于控制转换时间。所需的 RCK 振荡周期数（0 到 4096）可以通过设置从触摸键转换开始到结束的 TKTMR 来确定。转换结束后，用户可以从数据计数器获得 TKDATA（TKDH，TKDL）。TKDATA 会受到手指触摸的影响。

当手指触摸减慢 TCK 时，TKDATA 的值小于没有手指触摸的值。根据 TKDATA 的不同，用户可以检查它是否被触摸过。另一方面，设置 STKFSEL 可以调整 TK 系统的整体频率（包括 TCK 和 RCK）。STKREFC 只控制 RCK 的频率。设置 TKFJMP=1 时，系统将自动随机改变 TCK 频率。在触摸按键单元内有一个内置的参考电容器来模拟按键的行为。设置 TKCHS= 7 或 15 时，系统将切换到内置的参考电容器，并开始触摸按钮进行转换，得到这个参考电容器的 TKDATA。因为内部电容从来不受水或手机的影响，所以它对比较环境背景噪声很有用。触摸键时钟频率可以通过内部硬件通过设置 TKFJMP=1 来自动调整，这可以有效地提高触摸键的高干扰特性。



开始扫描时，用户先设置 STKPD=0，再设置 TKSOC 为 1，开始触摸按键转换。 TKEOC = 0 表示转换正在进行中。 TKEOC = 1 表示转换完成，触摸按键计数结果存入 12 位 TK 数据计数器 TKDH 和 TKDL。 TKSOC 位可以在转换结束时 TKEOC=1 后自动清零。

首次使能触摸按键功能时（STKPD = 0），TKIF 将被激活，用户应在 STKPD = 0 后清除 TKIF。

例：

```
BCX     STKPD       ; STKPD=0 (设置STK)
MOVLW   09H
MOVWX   TKCTL       ; HSENSE=1, STKREFC=1
                    ;
MOVLW   04H
MOVWX   TKMFS       ; TKTMR=400h
MOVLW   00H
MOVWX   TKTMRL      ; TKTMR=400h
MOVLW   A1H
MOVWX   TKCTL2      ; 设置TK1
MOVLW   11111011B
MOVWX   INTIF1      ; 清除TKIF
MOVLW   00000100B
IORWX   INTIE1      ; 设置TKIE
BSX     TKSOC
```

01Ah	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TKDL	TKDL							
R/W	R							
Reset	1	1	1	1	1	1	1	1

1Ah.7 **TKDL:** 触控键计数器数据 7~0

01Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TKDH	–	–	–	–	TKDH			
R/W	–	–	–	–	R			
Reset	–	–	–	–	1	1	1	1

1Bh.7 **TKDH:** 触控键计数器数据11~8

01Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TKTMRL	TKTMRL							
R/W	R/W							
Reset	1	1	1	1	1	1	1	1

1Ch.7 **TKTMR:** STK扫描长度调整位7~0。 12位TKTMR：
000：最短
FFF：最长

01Dh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TKMFS	STKPD	–	–	–	TKTMRH			
R/W	R/W	–	–	–	R/W			
Reset	1	–	–	–	0	0	0	0

1Dh.7 **STKPD：** STK断电
0: 已启用STK和CTK已禁用
1: STK禁用

1Dh.3~0 **TKTMRH:** STK扫描长度调整位11~8。 12位TKTMR：
000：最短
FFF：最长

01Eh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TKCTL	TKFJMP	JMPVAL			HSENSE	STKFSEL	STKREFC	
R/W	R/W	R/W			R/W	R/W	R/W	
Reset	0	0	0	0	0	0	0	0

1Eh.7 **TKFJMP:** 触摸按键时钟频率自动更改选择
0: 禁用，由SFR JMPVAL决定
1: 启用，触摸键时钟频率自动更改

1Eh.6~4 **JMPVAL:** 触摸键时钟频率选择（仅在TKFJMP=0中可用）

1Eh.3 **HSENSE:** STK通道灵敏度选择
0: 正常振荡型
1: 高灵敏度

1Eh.2 **STKFSEL:** STK时钟（RCK/TCK）频率选择。
0: STK时钟频率最慢
1: STK时钟频率最快

1Eh.1~0 **STKREFC:** STK参考时钟（RCK）电容器选择。
0: 最小值（转换时间最短）
3: 最大值（转换时间最长）

01Fh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TKCTL2	CTKPD	TKSOC	TKEOC	–	TKCHS			
R/W	R/W	R/W	R	–	R/W			
Reset	1	0	1	–	1	1	1	1

1Fh.6 **TKSOC:** 触摸键开始转换，转换结束时硬件清除
0: 关闭
1: 启用，触摸键开始转换

1Fh.5 **TKEOC:** 触摸按键结束时的转换

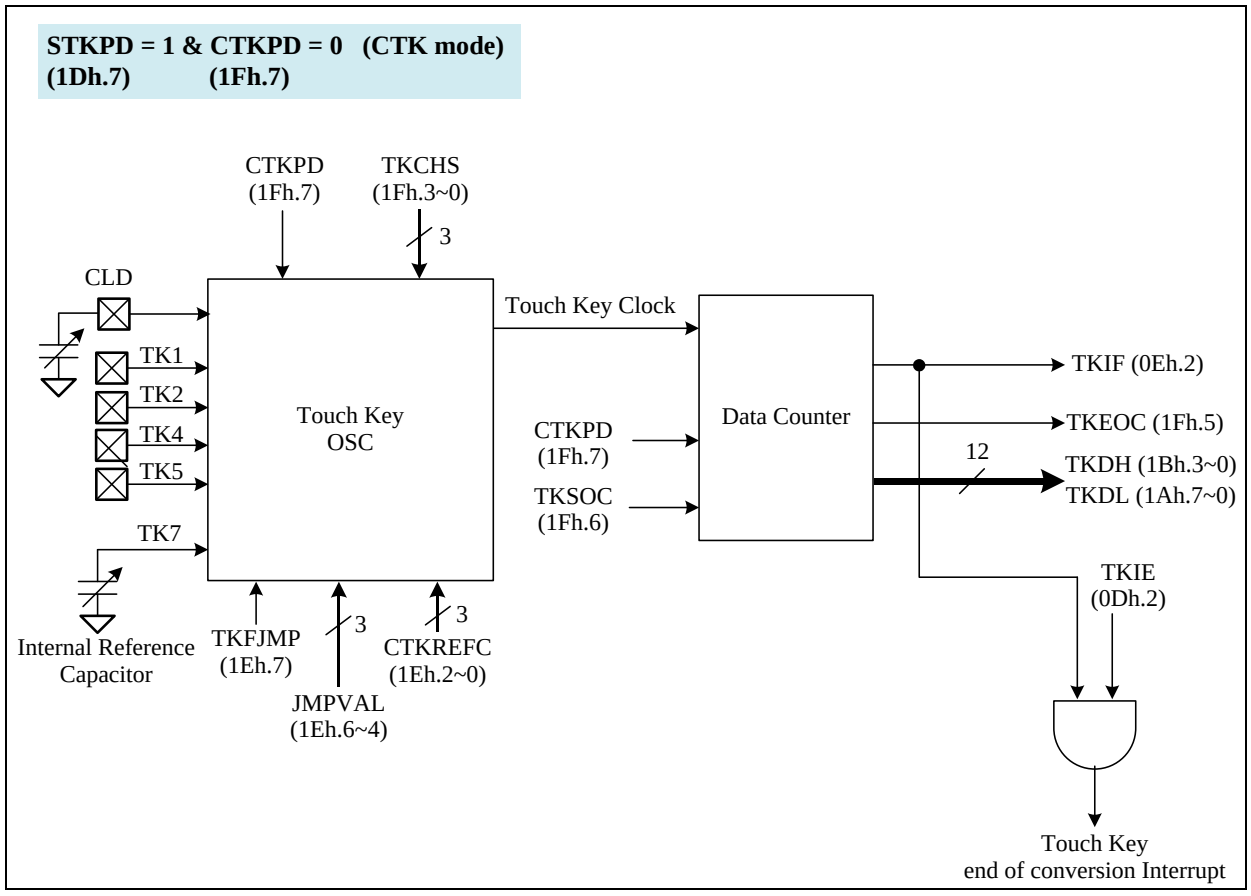
1Fh.3~0 **TKCHS:** 触摸键通道选择

x000: 预留	x100: TK4
x001: TK1	x101: TK5
x010: TK2	x110: 预留
x011: 预留	x111: 内部参考电容

6.5.2 CTK

设置 STKPD=1 和 CTKPD=0 时，用户可选择外部电容式结构触摸按钮。在 CTK 模式下，一个外部电容器需要连接到 CLD 引脚。要开始扫描，用户需要分配 CTKPD=0，然后设置 TKSOC 位以开始触摸键转换。当转换结束时，可以自动清除 TKSOC 位。TKEOC=为 0，表示转换正在进行中。TKEOC=1 表示转换结束，触摸键的计数值存储在 12 位 TK 数据计数（TKDH 和 TKDL）中。在 TKEOC=1 之后，用户必须等待 50μ s 进行下一次转换。如果转换后的值超出了周期范围，则减少/增加 CTKREFC 可以减少/增加 TK 数据计数，以适应系统电路板的条件。

在触摸按键单元内有一个内置的参考电容器来模拟按键的行为。设置 TKCHS= 7 或 15 时，系统将切换到内置的参考电容器，并开始触摸按钮进行转换，得到这个参考电容器的 TKDATA。因为内部电容从来不受水或手机的影响，所以它对比较环境背景噪声很有用。触摸键时钟频率可以通过内部硬件通过设置 TKFJMP=1 来自动调整，这可以有效地提高触摸键的高干扰特性。



例：

触摸键通道选择参考 TK，并选择加速 CLD 放电模式.

```
        MOVLW   00101011b       ; PA3 (CLD) = 输出低, PA2 (TK4) = CTKCKO
        MOVWX   PAMOD32         ; PA3 加速CLD放电的设置
        BCX     PAD,3           ; 设置PA3 输出低

        MOVLW   10000101b       ; 为CTK时钟自动更改
        MOVWX   TKCTL           ; CTK conversion time select 5

        MOVLW   00001111b       ; CTK 触摸键操作
        MOVWX   TKCTL2          ; 触摸键通道，选择参考TK
        BSX     TKSOC           ; 触摸键启动转换
                                ; (需要延迟5us)
        LCALL   WAIT_TK

WAIT_TK:
        BTXSS   TKEOC           ; 等待TK转换完成
        LGOTO   WAIT_TK
        RET
```

01Ah	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TKDL	TKDL							
R/W	R							
Reset	1	1	1	1	1	1	1	1

1Ah.7 **TKDL:** 触摸键计数器数据7~0

01Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TKDH	–	–	–	–	TKDH			
R/W	–	–	–	–	R			
Reset	–	–	–	–	1	1	1	1

1Bh.7 **TKDH:** 触摸键计数器数据 11~8

01Eh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TKCTL	TKFJMP	JMPVAL			HSENSE	CTKREFC		
R/W	R/W	R/W			R/W	R/W		
Reset	0	0	0	0	0	0	0	0

1Eh.7 **TKFJMP:** 触摸按键时钟频率自动更改选择
0: 禁用，由SFR JMPVAL决定
1: 启用，触摸键时钟频率自动更改

1Eh.6~4 **JMPVAL:** 触摸键时钟频率选择（仅在TKFJMP=0中可用）

1Eh.2~0 **CTKREFC:** CTK转换时间。
0: 最短
7: 最长

01Fh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TKCTL2	CTKPD	TKSOC	TKEOC	–	TKCHS			
R/W	R/W	R/W	R	–	R/W			
Reset	1	0	1	–	1	1	1	1

1Fh.7 **CTKPD:** CTK电源下降
0: 如果启用STKPD=1
1: CTK禁用。

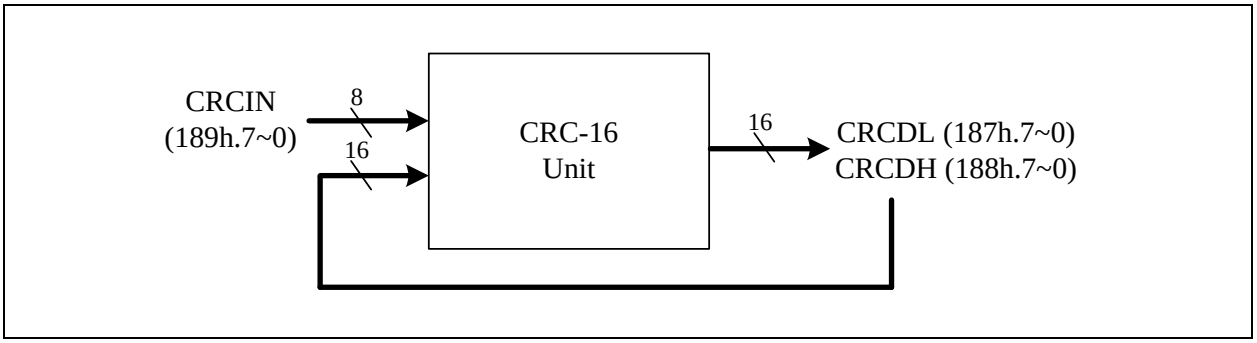
1Fh.6 **TKSOC:** 触摸键开始转换，转换结束时硬件清除
0: 关闭
1: 启用，触摸键开始转换

1Fh.5 **TKEOC:** 触摸按键结束时的转换

1Fh.3~0 **TKCHS:** 触摸键通道选择
x000: 预留 x100: TK4
x001: TK1 x101: TK5
x010: TK2 x110: 预留
x011: 预留 x111:内部参考电容

6.6 循环冗余检查 (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 多项式表示：$\mathbf{X^{16} + X^{15} + X^2 + 1}$

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

187h.7~0 **CRCDL:** 16 位 CRC 校验数据位 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							
Reset	1	1	1	1	1	1	1	1

188h.7~0 **CRCDH:** 16 位 CRC 校验数据位 15~8

189h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CRCIN	CRCIN							
W	W							
Reset	–	–	–	–	–	–	–	–

189h.7~0 **CRCIN:** CRC 数据输入，写入此寄存器以开始 CRC 计算

存储器功能图

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	-	不是实质寄存器，寻址 INDF 实际上指向其地址为 FSR 寄存器中的寄存值
TM0 (01h/101h)				**相关功能: Timer0**
TM0	01.7~0	R/W	0	Timer0 计数数据
PCL (02h/82h/102h/182h)				**相关功能: PROGRAM COUNT**
PCL	02.7~0	R/W	0	程序计数器数据位 7~0
STATUS (03h/83h/103h/183h)				**相关功能: STATUS**
IRP	03.7	R/W	0	寄存器 Bank 选择位 (用于间接寻址)
RP1	03.6	R/W	0	寄存器 Bank 选择位 1 用于直接寻址
RP0	03.5	R/W	0	寄存器 Bank 选择位 0 用于直接寻址
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: 端口 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**
GPR	0A.7~5	R/W	0	通用寄存器
PCLATH	0A.2~0	R/W	0	针对程序计数器的高字节的写缓冲区
INTIE (0Bh/8Bh/10Bh/18Bh)				**相关功能: 中断使能**
TM1IE	0B.5	R/W	0	Timer1 中断使能 0: 关闭 1: 使能
TM0IE	0B.4	R/W	0	Timer0 中断使能 0: 关闭 1: 使能
WKTIE	0B.3	R/W	0	唤醒定时器中断使能和唤醒定时器使能 0: 关闭 1: 使能
INT2IE	0B.2	R/W	0	INT2 引脚 (PA7 或 PB5) 中断使能 0: 关闭 1: 使能
INT1IE	0B.1	R/W	0	INT1 引脚 (PA1 或 PB1) 中断使能 0: 关闭 1: 使能
INT0IE	0B.0	R/W	0	INT0 引脚 (PA3 或 PB2) 中断使能 0: 关闭 1: 使能
INTIF (0Ch)				**相关功能: 中断标志**
TM1IF	0C.5	R	-	Timer1 中断事件挂起标志，当 Timer1 溢出时由 H/W 置位
		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
		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	-	INT2 (PA7 或 PB5) 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	-	INT1 (PA1 或 PB1) 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	-	INT0 (PA3 或 PB2) 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:中断使能组 1**
TKIE	0D.2	R/W	0	TK 中断启用, 0 =禁用 1 =启用
PWMIE	0D.1	R/W	0	PWM 中断使能 0：禁用 1：启用
LVDIE	0D.0	R/W	0	LVD 中断启用 0: 禁用 1：启用
INTIF1 (0Eh)				**相关功能: 中断标志**
TKIF	0E.2	R	-	触摸键中断标志
		W	0	写入 0：清除此标志；写入 1：无操作
PWMIF	0E.1	R	-	PWM 中断标志
		W	0	写入 0：清除此标志；写入 1：无操作
LVDIF	0E.0	R	-	LVD 中断标志
		W	0	写入 0：清除此标志；写入 1：无操作
CLKCTL (0Fh)				**相关功能：系统时钟（Fsys）**
SLOWSTP	0F.4	R/W	0	在停止模式下的慢时钟停止控制。 0: 慢时运行 1：慢时停止
FASTSTP	0F.3	R/W	1	快时钟停止控制 0: 快时运行 1：快时停止
CPUCKS	0F.2	R/W	0	选择快速时钟 0: Fsys=慢时钟 1：Fsys=快时钟
CPUPSC	0F.1~0	R/W	11	Fsys 预分频器, 00: div 8 01: div 4 10: div 2 11: div 1
TM0RLD (10h)				**相关功能：TM0**
TM0RLD	10.7~0	R/W	0	定时器 0 重新加载数据
TM0CTL (11h)				**相关功能：TM0**
TM0STP	11.6	R/W	0	停止计时器 0 0: 定时器 0 运行 1: 定时器 0 停止
TM0EDG	11.5	R/W	0	TM0CKI（PA2） 0: 上升沿 1：下降沿
TM0CKS	11.4	R/W	0	定时器 0 预分频器时钟源 0: Fsys/2 1: TM0CKI (PA2)
TM0PSC	11.3~0	R/W	0	Timer0 预分频器。 Timer0 时钟源（Fsys/2 或 TMCKI）除以 0000: 1 0011: 8 0110: 64 0001: 2 0100: 16 0111: 128 0010: 4 0101: 32 1xxx: 256
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 重新加载数据
TM1CTL (14h)				**相关功能: Timer1**
TM1STP	14.6	R/W	0	停止计时器 1 0: 定时器 1 运行 1: 定时器 1 停止
TM1PSC	14.3~0	R/W	0	Timer1 预分频器. Timer1 时钟源（Fsys/2）除以

				0000: 1 0011: 8 0110: 64 0001: 2 0100: 16 0111: 128 0010: 4 0101: 32 1xxx: 256
LVCTL (16h)				**相关功能: LVD/LVR**
LVDF	16.7	R	0	低压检测标志，由 H/W 和 VCC≦LVD 设置
LVDHYS	16.6	R/W	0	LVD 滞后。 0: 禁用 1：启用
LVRSAV	16.5	R/W	1	POR/LVR 自动关机
LVDSAV	16.4	R/W	1	LVD 在关机/空闲模式下自动关机
LVDS	16.3~0	R/W	00	LVD 设置 0000: 关闭 0100 : 2.60V 1000: 3.15V 1100: 3.70V 0001: 2.20V 0101: 2.75V 1001: 3.30V 1101: 3.85V 0010: 2.30V 0110: 2.90V 1010: 3.45V 1110: 4.00V 0011: 2.45V 0111: 3.00V 1011: 3.60V 1111: 4.15V
TKDL (1Ah)				**相关功能: STK/CTK**
TKDL	1A.7~0	R	-	触摸键计数器数据 7~0
TKDH (1Bh)				**相关功能: STK/CTK**
TKDH	1B.3~0	R	-	触摸键计数器数据 11~8
TKTMRL(1Ch)				**相关功能: STK**
TKTMRL	1C.7~0	R/W	FF	STK 扫描长度调整位 7~0。 12 位 TKTMR： 000：最短，FFF：最长
TKMFS (1Dh)				**相关功能: STK**
STKPD	1D.7	R/W	1	STK 断电 0: STK 启用和 CTK 禁用。 1: STK 禁用
TKTMRH	1D.3~0	R/W	0	STK 扫描长度调整位 11~8。 12 位 TKTMR： 000：最短，FFF：最长
TKCTL (1Eh)				**相关功能: STK/CTK**
TKFJMP	1E.7	R/W	0	触摸按键时钟频率自动更改选择 0: 禁用，时钟频率由 SFR JMPVAL 决定 1: 使能够
JMPVAL	1E.6~4	R/W	0	触摸按键时钟频率的选择。（仅可在 TKFJMP=0 中使用）
HSENSE	1E.3	R/W	0	STK 通道灵敏度选择 0: 正常振荡型 1: 高灵敏度
STKFSEL	1E.2	R/W	0	STK 时钟（RCK/TCK）频率选择 0: STK 时钟频率最慢 1: STK 时钟频率最快
STKREFC	1E,1~0	R/W	0	STK 参考时钟（RCK）电容器选择 0:最小（转换时间最短） 3:最大（转换时间最长）
CTKREFC	1E,2~0	R/W	0	CTK 转换时间。 0: 最小 7：最长
TKCTL2 (1Fh)				**相关功能: STK/CTK**
CTKPD	1F.7	R/W	1	CTK 断电。 0：CTK 使能 (if STKPD=1) 1：CTK 停用
TKSOC	1F.6	R/W	0	触摸键开始转换，转换结束时清除
TKEOC	1F.5	R	1	触摸按键的转换结束
TKCHS	1F.3~0	R/W	F	触摸键通道选择。 x000: reserved x100: TK4 x001: TK1 x101: TK5 x010: TK2 x110: 保留 x011: 保留 x111: 内部参考电容
User Data Memory 用户数据存储器				
RAM	20~6F	R/W	-	RAM Bank0 区域(80 Bytes)
RAM	70~7F	R/W	-	RAM common 区域(16 Bytes)

OPTION (81h/181h)				相关功能: STATUS/INT0/INT1/WDT/WKT
HWAUTO	81.7	R/W	0	输入中断向量， 硬件保存/恢复 WREG 和状态，PD 0:禁用 1：启用
INT0EDG	81.6	R/W	0	INT0 引脚边缘中断事件 0: 下降边缘触发 1: 上升边缘触发
INT1EDG	81.5	R/W	0	INT1 引脚边缘中断事件 0: 下降边缘触发 1: 上升边缘触发
WDTPSC	81.3~2	R/W	11	WDT 周期选择: 00: 100mS 01: 200mS 10: 800mS 11: 1600mS @4V
WKTPSC	81.1~0	R/W	11	WKT 周期选择: 00: 12mS 01: 25mS 10: 50mS 11: 100mS @4V
PAMOD10 (85h)				**相关功能: Port A**
PA1MOD	85.7~4	R/W	0001	PA1~PA0 I/O 模式控制
PA0MOD	85.3~0	R/W	0001	
PAMOD32 (86h)				**相关功能: Port A**
PA3MOD	86.7~4	R/W	0001	PA3~PA2 I/O 模式控制
PA2MOD	86.3~0	R/W	0001	
PAMOD54 (87h)				**相关功能: Port A**
-	87.7~4	R/W	0001	PA4 I/O 模式控制
PA4MOD	87.3~0	R/W	0001	
PAMOD76 (88h)				**相关功能:Port A**
PA7MOD	88.7~4	R/W	0000	PA7 I/O 模式控制
-	88.3~0	R/W	0001	
PWMCTL (89h)				**相关功能:PWM0**
PWMEN	89.7	R/W	0	PWM 时钟使能 0:禁用 1:启用
OPTION2 (91h)				**相关功能:PWM0**
PWMCKS	91.5~4	R/W	00	PWM 时钟源选择 0x: Fsys 10: FIRC (16MHz) 11: FIRC*2 (32MHz) (VCC>2.3V@25℃)
PWMPRDH (92h)				**相关功能：PWM**
PWMPRDH	92.7~0	R/W	FF	PWM 周期 MSB 数据
PWMPRDL (93h)				**相关功能：PWM**
PWMPRDL	93.7~0	R/W	FF	PWM 周期 LSB 数据
PWM0DH (94h)				**相关功能：PWM0**
PWM0DH	94.7~0	R/W	80	PWM0 周期 MSB 周期
PWM0DL (95h)				**相关功能：PWM0**
PWM0DL	95.7~0	R/W	00	PWM0 周期 LSB 周期
PWM1DH (96h)				**相关功能：PWM1**
PWM1DH	96.7~0	R/W	80	PWM1 周期 MSB 周期
PWM1DL (97h)				**相关功能：PWM1**
PWM1DL	97.7~0	R/W	00	PWM1 周期 LSB 周期
PWM2DH (98h)				**相关功能：PWM2**
PWM2DH	98.7~0	R/W	80	PWM2 周期 MSB 周期
PWM2DL (99h)				**相关功能：PWM2**
PWM2DL	99.7~0	R/W	00	PWM2 Duty LSB 周期

PWM4DH (9Ch)				**相关功能：PWM4**
PWM4DH	9C.7~0	R/W	80	PWM4 周期 MSB 周期
PWM4DL (9Dh)				**相关功能：PWM4**
PWM4DL	9D.7~0	R/W	00	PWM4 周期 LSB 数据
PWM5DH (9Eh)				**相关功能：PWM5**
PWM5DH	9E.7~0	R/W	80	PWM5 周期 MSB 数据
PWM5DL (9Fh)				**相关功能：PWM5**
PWM5DL	9F.7~0	R/W	00	PWM5 周期 LSB 数据
User Data Memory				
RAM	A0~EF	R/W	-	RAM bank1 区域(80 字节)
LVRPD (109h)				**相关功能： LVR/POR**
LVRPD	109.7~0	W	0	寫入 37h: 強制關閉 LVR, 強制關閉 POR 寫入 38h: 強制關閉 LVR，使能 POR 寫入 39h: 強制關閉 POR ，使能 LVR 寫入其他任意值: 使能 LVR，使能 POR
PORPDF	109.1	R	0	POR 强制断电标志 0: POR 启用 1: POR 强制断电
LVRPDF	109.0	R	0	LVR 强制断电标志 0: LVR 启用 1: LVR 强制断电
PCH (10Ch)				**相关功能: PCH**
PCH	10C.7~0	W	0	当执行以 PCL 作为目标的指令时，编程计数器高字节源选择 写 0x1C 来设置 PCH_S = 1： PCH 保持原始值 写其他的东西来清除 PCH_S = 0： PCH 是来自 PCLATH 的
PCH	10C.2~0	R	-	程序计数器数据位 10~8
BGTRIM (10Eh)				**相关功能： Bandgap**
BGTRIM	10E.4~0	R/W	CFG	VBG 电压调整
IRCF (10Fh)				**相关功能： Internal RC**
IRCF	10F.6~0	R/W	CFG	FIRC 频率调整
User Data Memory 用户数据存储器				
RAM	120~16F	R/W	-	RAM Bank2 区域(80 Bytes)

DPL (185h)				**相关功能:表读取**
DPL	185.7~0	R/W	00	表读低地址，数据 ROM 指针(DPTR)低字节
DPH (186h)				**相关功能:表读取**
DPH	186.3~0	R/W	00	表读高地址，数据 ROM 指针(DPTR)高字节
CRCDL (187h)				**相关功能: CRC16**
CRCDL	187.7~0	R/W	FF	CRC16 数据 7~0
CRCDH (188h)				**相关功能:CRC16**
CRCDH	188.7~0	R/W	FF	CRC16 数据 15~8
CRCIN (189h)				**相关功能: CRC16**
CRCIN	189.7~0	W	0	CRC 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. 读取 TABR 以获得主 ROM 表的读取值 *表读取的 ASM：使用 TABRL / TABRH 指令或寄存器 TABR* *表读取为 C：使用寄存器 TABR*

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 ff ff	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
LCALL	k	kk10 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
LGOTO	k	kk10 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	立即数"k"减去 W
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 : 000h ~ 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 : 000h ~ 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 : 000h ~ 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 : 000h ~ 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 : 000h ~ 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 : 000h ~ 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

CLRX	清除 "f"	
语法	CLRX f	
操作数	f : 000h ~ 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 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 : 000h ~ 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 : 000h ~ 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 : 000h ~ 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

INCX	"f" 递增
语法	INCX f [,d]
操作数	f : 000h ~ 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 : 000h ~ 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 : 000h ~ 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

LCALL	**调用子程序 “k”**
语法	LCALL k
操作数	k : 0000h ~ 1FFFh
运作方式	Operation: TOS ← (PC) + 1, PC.12~0 ← k
影响的状态位	-
操作码	kk10 0kkk kkkk kkkk
描述	LCALL 子例程。首先，返回地址 (PC+1) 被推送到堆栈上。13 位直接地址加载到 PC 位 <12:0>。LCALL 是一个两周期指令。
周期	2
范例	LABEL1 LCALL SUB1 B : PC =LABEL1 A : PC =SUB1, TOS =LABEL1 + 1

LGOTO	**无条件转移**
语法	LGOTO k
操作数	k : 0000h ~ 1FFFh
运作方式	PC.12~0 ← k
影响的状态位	-
操作码	kk10 1kkk kkkk kkkk
描述	LGOTO 是一个无条件的分支。13 位立即值加载到 PC 位 <12:0>。LGOTO 是一个两周期指令。
周期	2
范例	LABEL1 LGOTO SUB1 B : PC =LABEL1 A : PC =SUB1

MOVX	**移 "f"**
语法	MOVX f [,d]
操作数	f : 000h ~ 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 : 000h ~ 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

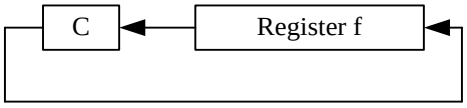
MOVWX	**将 W 移至 "f"**	
语法	MOVWX f	
操作数	f : 000h ~ 1FFh	
运作方式	(f) ← (W)	
影响的状态位	-	
操作码	ff00 0000 1fff ffff	
描述	将数据从 W 寄存器移至寄存器 'f'。	
周期	1	
范例	MOVWX REG1	B : REG1 =0xFF, W =0x4F A : REG1 =0x4F, W =0x4F

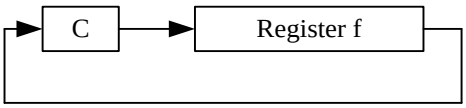
NOP	**空操作**	
语法	NOP	
操作数	-	
运作方式	空操作	
影响的状态位	-	
操作码	0000 0000 0000 0000	
描述	空操作	
周期	1	
范例	NOP	-

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	A : PC =TOS

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	LCALL 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 : 000h ~ 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 : 000h ~ 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	**立即数减去 W**
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	立即数 'k'减 W 寄存器。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 : 000h ~ 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 : 000h ~ 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 REG1, 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], DPL[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 ;DPL 为寄存器 MOVLW (TBA1>>8)&0xFF MOVWX DPH ;DPH 为寄存器 TABRL ;W =0x89 TABRH ;W =0x37 ORG 0234H TAB1: DT 0x3789, 0x2277 ;16 位 ROM 数据

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], DPL[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 ;DPL 为寄存器 MOVLW (TBA1>>8)&0xFF MOVWX DPH ;DPH 为寄存器 TABRL ;W =0x89 TABRH ;W =0x37 ORG 0234H TAB1: DT 0x3789, 0x2277 ;16 位 ROM 数据

TSTX	检测 "f" 是否为 0
语法	TSTX f
操作数	f : 000h ~ 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 : 000h ~ 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 REG1, 1　　B : REG1 =0xAF, W =0xB5 A : REG1 =0x1A, W =0xB5

电气特性

1. 最大绝对额定值 (T_A =25℃)

参数	范围	单位
电源电压	V_{SS} -0.3 to V_{SS} +5.5	V
输入电压	V_{SS} -0.3 to V_{CC} +0.3	
输出电压	V_{SS} -0.3 to V_{CC} +0.3	
每个引脚的高电位输出电流	-25	mA
所有引脚的高电位输出电流	-80	
每个引脚的低电位输出电流	+30	
所有引脚的低电位输出电流	+150	
最大工作电压	5.5	V
工作温度	-40 to +105	℃
储存温度	-65 to +150	

2. 直流特性 (TA =25℃, VCC =5.0V, unless otherwise specified)

参数	符号	条件		最小	典型	最大	单位
工作电压	V_{cc}	Fsys = 16Mhz		1.9	–	5.5	V
		Fsys = 8Mhz		1.4	–	5.5	V
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
I/O 端口 拉电流	I_{OH}	所有 I/O 引脚 (除去 PA7)	V_{CC} = 5V,V_{OH} = 4.5V	–	TBD	–	mA
			V_{CC} = 3V,V_{OH} = 2.7V	–	TBD	–	
		PA7	V_{CC} = 5V,V_{OH} = 4.5V	–	TBD	–	
			V_{CC} = 3V,V_{OH} = 2.7V	–	TBD	–	
I/O 端口 灌电流	I_{OL}	所有 I/O 引脚 (除去 PA7)	V_{CC} = 5V,V_{OL} = 0.5V	–	TBD	–	mA
			V_{CC} = 3V,V_{OL} = 0.3V	–	TBD	–	
		PA7	V_{CC} = 5V,V_{OL} = 0.5V	–	TBD	–	
			V_{CC} = 3V,V_{OL} = 0.3V	–	TBD	–	
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

参数	符号	条件		最小	典型	最大	单位
工作电流 (空载)	I_{CC}	FAST mode FIRC 16 MHz	V_{CC} = 5V	–	6.4	–	mA
			V_{CC} = 3V		3.7		
		FAST mode FIRC 8 MHz	V_{CC} = 5V	–	3.8	–	
			V_{CC} = 3V		2.3		
		FAST mode FIRC 4 MHz	V_{CC} = 5V	–	3.2	–	
			V_{CC} = 3V	–	1.9	–	
		SLOW mode SIRC/1 FIRC STOP	V_{CC} = 5V	–	1.7	–	
			V_{CC} = 3V	–	1.3	–	
		IDLE mode LVRSAV = 0 LVDSAV=0	V_{CC} = 5V	–	96	–	uA
			V_{CC} = 3V	–	60	–	
		IDLE mode LVRSAV = 1 LVDSAV=1	V_{CC} = 5V	–	20		uA
			V_{CC} = 3V	–	6		
		STOP mode LVRSAV = 1 LVDSAV=1	V_{CC} = 5V	–	0.1		uA
			V_{CC} = 3V	–	0.1		
上拉电阻	R_{UP}	V_{IN} = 0 V Ports A	V_{CC} = 5V	–	TBD	–	KΩ
			V_{CC} = 3V	–	TBD	–	
		V_{IN} = 0 V PA7	V_{CC} = 5V	–	TBD	–	
			V_{CC} = 3V	–	TBD	–	

3. 时钟时序

参数	条件	最小	典型	最大	单位
FIRC 频率(*)	-20℃ ~ 85℃, V_{CC} = 3.0 ~ 5.0V	-5%	16	+1.5%	MHz
	-20℃ ~ 85℃, V_{CC} = 4.0 V	-3%	16	+1.5%	
	0℃ ~ 70℃, V_{CC} = 4.0 V	-2%	16	+1.5%	
	25℃, V_{CC} = 3.0 ~ 5.0 V	-1%	16	+1%	
	25℃, V_{CC} =4.0 V	-0.5%	16	+0.5%	

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

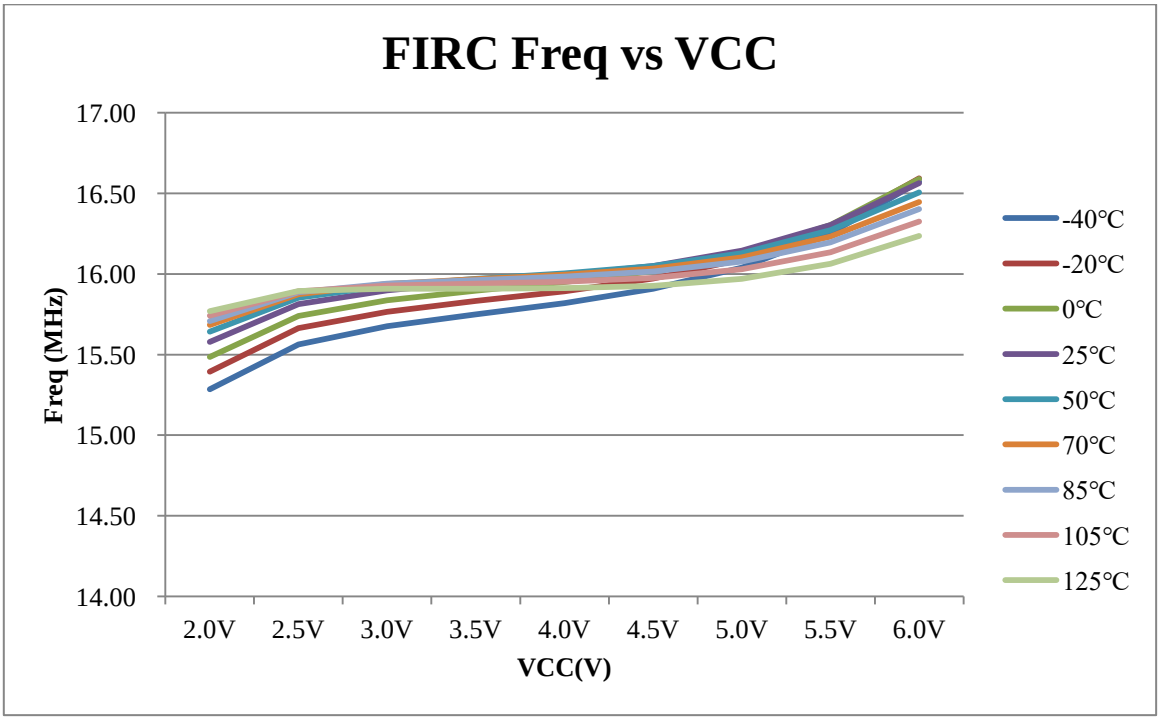
4. 复位时间特性 (T_A = 25°C)

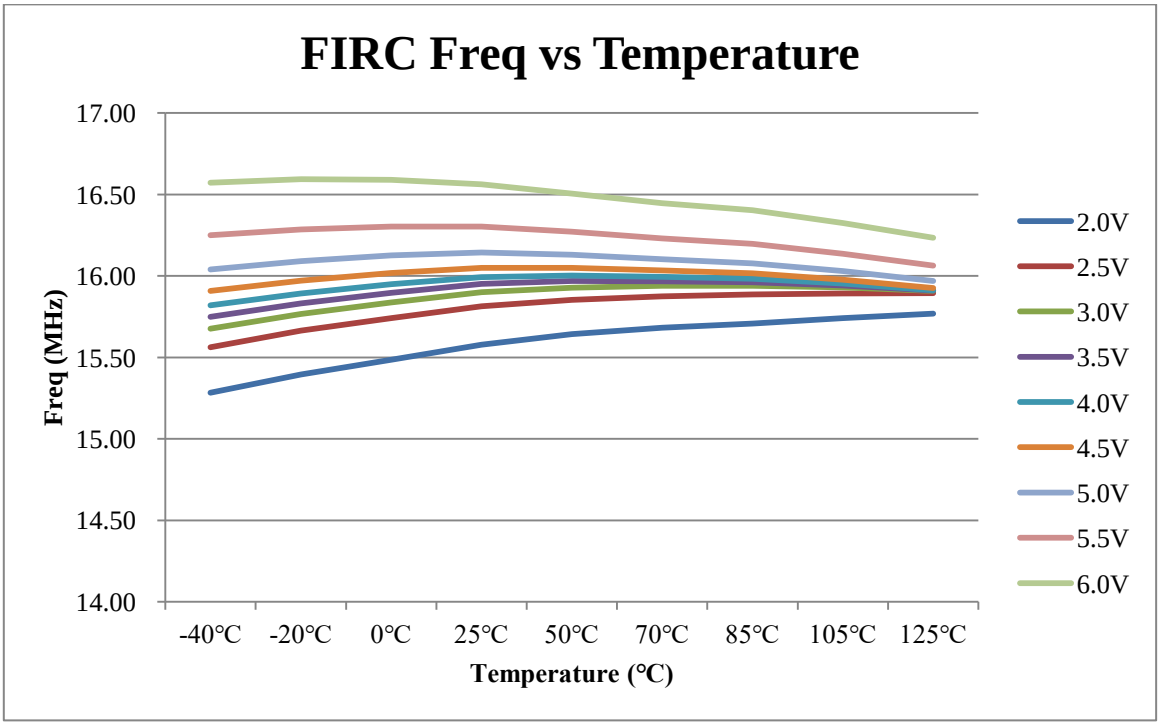
参数	条件	最小	典型	最大	单位
复位输入低脉宽	Input V_{CC} = 5 V ±10 %	–	30	–	μs
WDT 时间	V_{CC} = 4 V, WDTPSC = 11	–	1600	–	ms
WKT 时间	V_{CC} = 4 V, WKTPSC = 11	–	100	–	ms
CPU 启动时间	V_{CC} = 4 V	–	25	–	ms

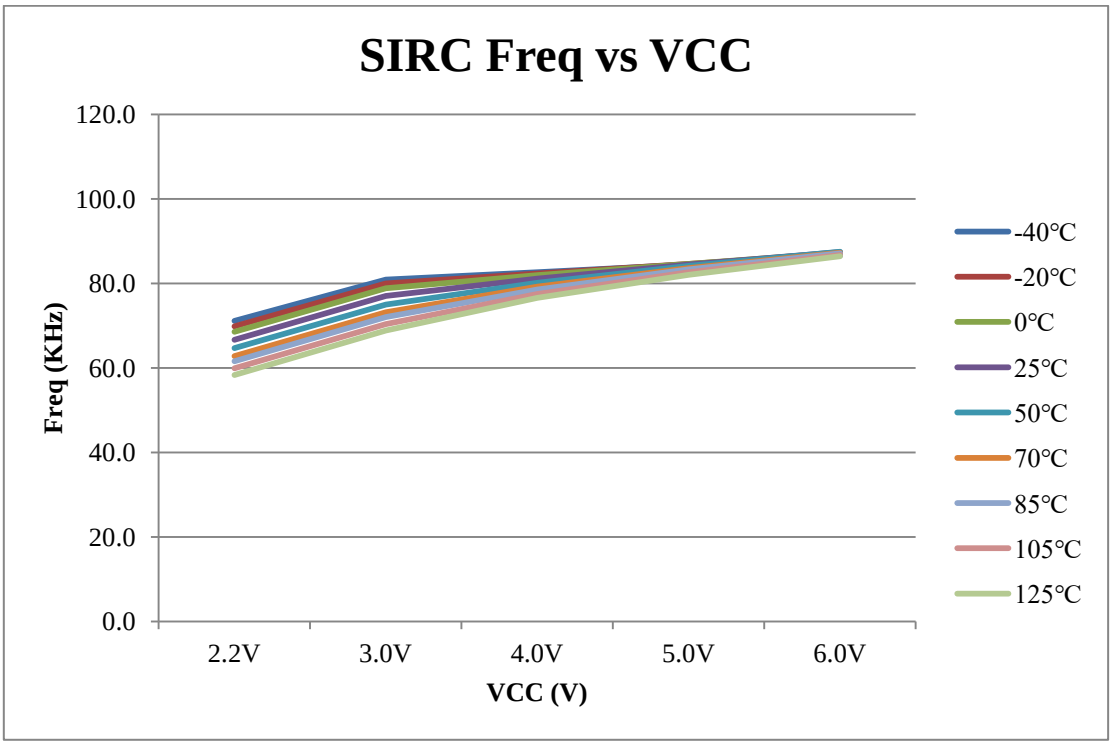
5. LVR 和 LVD 电路特性 (TA = 25℃)

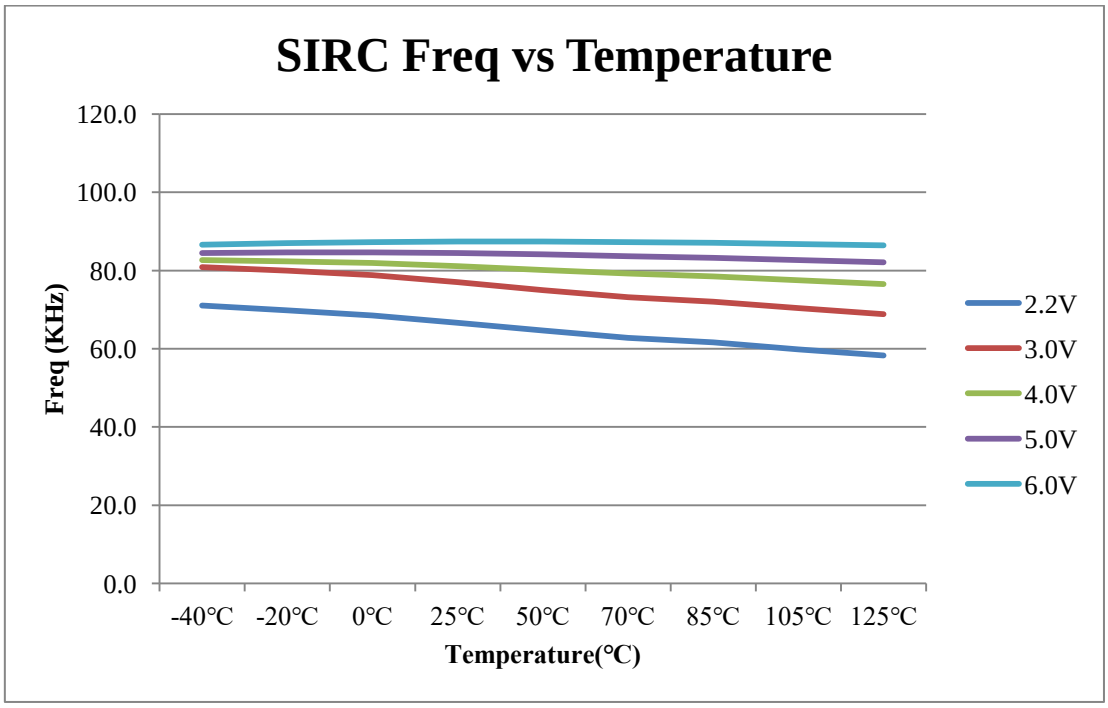
参数	符号	条件	最小	典型	最大	单位
LVR 参考电压	V_{LVR}	T_A=25°C	–	2.05	–	V
			–	2.20	–	
			–	2.30	–	
			–	2.45	–	
			–	2.60	–	
			–	2.75	–	
			–	2.90	–	
			–	3.00	–	
			–	3.15	–	
			–	3.30	–	
			–	3.45	–	
			–	3.60	–	
			–	3.70	–	
			–	3.85	–	
			–	4.00	–	
			–	4.15	–	
LVR 迟滞电压	V_{HYS_LVR}	T_A=25°C	–	0	–	mV
低电压检测时间	T_{LVR}	T_A=25°C	100	–	–	μs
LVD 参考电压	V_{LVD}	T_A=25°C	–	2.20	–	V
			–	2.30	–	
			–	2.45	–	
			–	2.60	–	
			–	2.75	–	
			–	2.90	–	
			–	3.00	–	
			–	3.15	–	
			–	3.30	–	
			–	3.45	–	
			–	3.60	–	
			–	3.70	–	
			–	3.85	–	
			–	4.00	–	
			–	4.15	–	
LVD 迟滞电压	V_{HYS_LVD}	LVDHYS=0	–	0	–	mV
		LVDHYS=1, LVD_{th}=2.20V		30		
		LVDHYS=1, LVD_{th}=3.15V		50		
		LVDHYS=1, LVD_{th}=4.15V	–	70	–	
低电压检测时间	T_{LVD}	T_A=25°C	100	–	–	μs

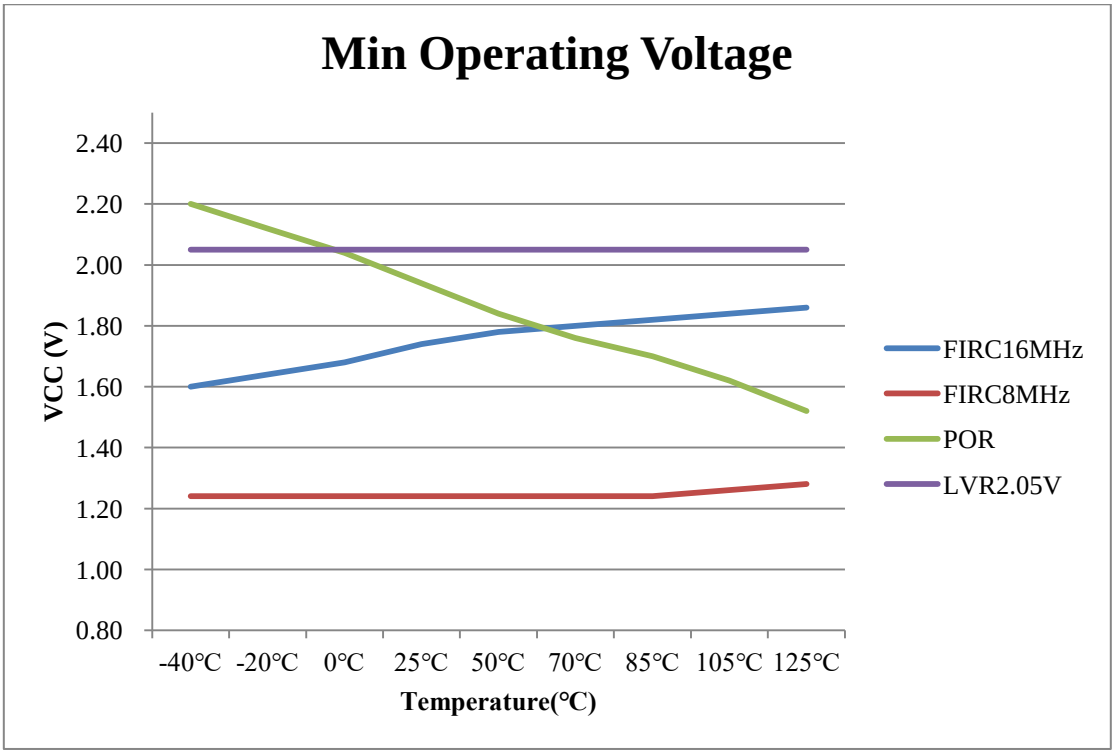
6. 电气特性曲线图


FIRC Freq vs VCC
17.00
16.50
16.00
15.50
15.00
14.50
14.00
Freq (MHz)
2.0V
2.5V
3.0V
3.5V
4.0V
4.5V
5.0V
5.5V
6.0V
VCC(V)
-40℃
-20℃
0℃
25℃
50℃
70℃
85℃
105℃
125℃


FIRC Freq vs Temperature
17.00
16.50
16.00
15.50
15.00
14.50
14.00
Freq (MHz)
-40℃
-20℃
0℃
25℃
50℃
70℃
85℃
105℃
125℃
Temperature (℃)
2.0V
2.5V
3.0V
3.5V
4.0V
4.5V
5.0V
5.5V
6.0V


SIRC Freq vs VCC
120.0
100.0
80.0
60.0
40.0
20.0
0.0
Freq (KHz)
2.2V
3.0V
4.0V
5.0V
6.0V
VCC (V)
-40℃
-20℃
0℃
25℃
50℃
70℃
85℃
105℃
125℃


SIRC Freq vs Temperature
120.0
100.0
80.0
60.0
40.0
20.0
0.0
Freq (KHz)
-40℃
-20℃
0℃
25℃
50℃
70℃
85℃
105℃
125℃
Temperature(℃)
2.2V
3.0V
4.0V
5.0V
6.0V


Min Operating Voltage
2.40
2.20
2.00
1.80
1.60
1.40
1.20
1.00
0.80
VCC (V)
-40℃ -20℃ 0℃ 25℃ 50℃ 70℃ 85℃ 105℃ 125℃
Temperature(℃)
FIRC16MHz
FIRC8MHz
POR
LVR2.05V

封装信息

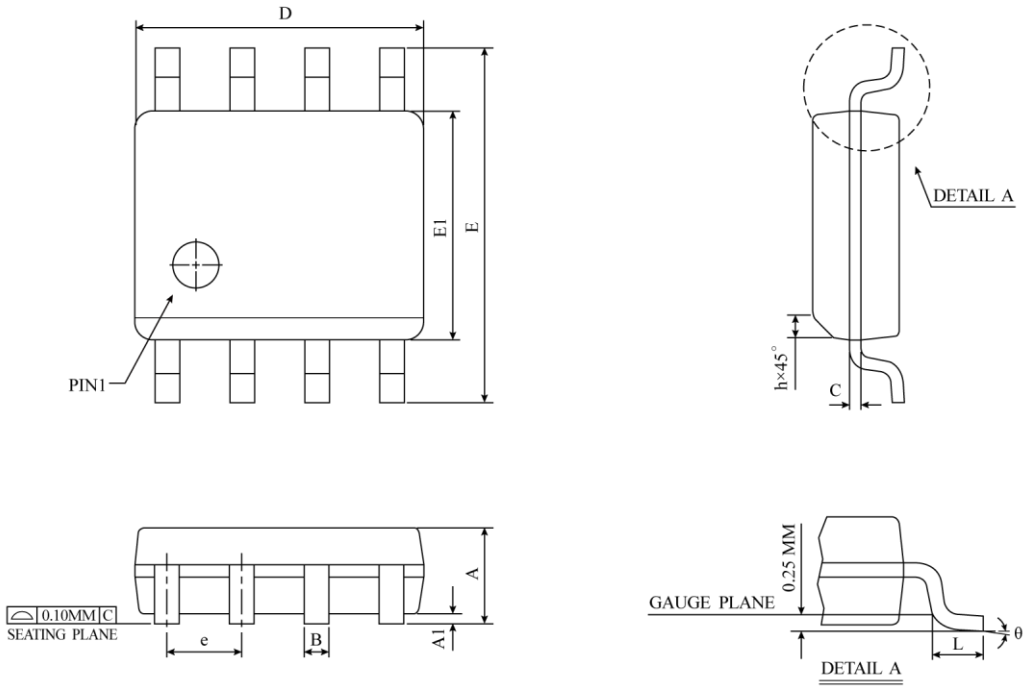
请注意，此处提供的包装信息仅供参考。 由于此信息经常更新，因此用户可以联系销售人员以咨询最新的包装信息和库存。

订购信息:

Ordering number	Package
TM56M1511-MTP	Wafer / Dice blank chip
TM56M1511-COD	Wafer / Dice with code
TM56M1511-MTP-14	SOP 8 pin (150mil)
TM56M1511-MTP-A8	SOT23-6

Ordering number	Package
TM56M1531-MTP	Wafer / Dice blank chip
TM56M1531-COD	Wafer / Dice with code
TM56M1531-MTP-14	SOP 8 pin (150mil)
TM56M1531-MTP-A8	SOT23-6

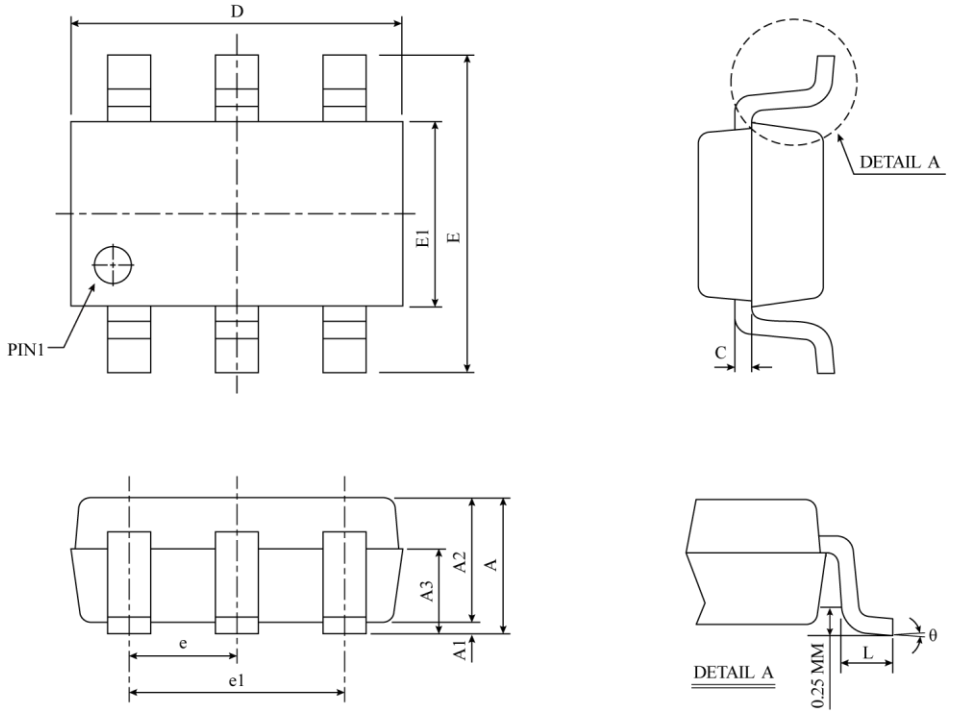
SOP-8 (150mil) 封装尺寸



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	4.80	4.90	5.00	0.1890	0.1939	0.1988
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 (AA)					

⚠ * 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.

SOT23-6 封装尺寸



SYMBOL	DIMENSION IN MM			DIMENSION IN INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	-	-	1.45	-	-	0.057
A1	0	0.08	0.15	0	0.003	0.006
A2	0.90	1.10	1.30	0.035	0.043	0.051
A3	0.60	0.65	0.70	0.024	0.026	0.028
c	0.12	0.16	0.19	0.005	0.006	0.007
D	2.82	2.92	3.02	0.111	0.115	0.119
E	2.70	2.90	3.10	0.106	0.114	0.122
E1	1.52	1.62	1.72	0.060	0.064	0.068
e	0.85	0.95	1.05	0.033	0.037	0.041
e1	1.80	1.90	2.00	0.071	0.075	0.079
L	0.35	0.48	0.60	0.014	0.019	0.024
θ	0°	4°	8°	0°	4°	8°
JEDEC	M0-178 (AB)					

⚠ * NOTES：ALL DIMENSIONS REFER TO JEDEC STANDARD MO-178 AB
DO NOT INCLUDE MOLD FLASH OR PROTRUSIONS.