十速

# TM56F5412/16/12B/16B

## *DATA SHEET*

## *Rev 0.94*

**tenx** reserves the right to change or discontinue the manual and online documentation to this product herein to improve reliability, function or design without further notice. **tenx** does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others. **tenx** products are not designed, intended, or authorized for use in life support appliances, devices, or systems. If Buyer purchases or uses **tenx** products for any such unintended or unauthorized application, Buyer shall indemnify and hold **tenx** and its officers, employees, subsidiaries, affiliates and distributors harmless against all claims, cost, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use even if such claim alleges that **tenx** was negligent regarding the design or manufacture of the part.

# AMENDMENT HISTORY

| Version | Date | Description |
|---|---|---|
| 0.90 | Apr, 2020 | New release. |
| 0.91 | Aug, 2020 | Page10 Added SOP16/SOP20 pin assignment<br>Page12 Added SOP16/SOP20 pin summary<br>Page 23 Modify the lookup table example code<br>Page104 Added SOP16/SOP20 ordering information<br>page 105/106 add SOP16/SOP20 package dimension<br>Page 6/75/94 Modify the description error. |
| 0.92 | Sep, 2020 | Page 27/28 modify IRCF register description, and move CLKCTRL register description to page 28<br>Page 34 Added INT0EDG/INT1EDG register description<br>Page 35: modify TXD1F/TXD0F description<br>Page 54~58 modify the PWM0/1/2 CLR and PWM1A/B/C OE description<br>Page 72 modify Slave I2C Command description |
| 0.93 | Jan, 2021 | Add TM56F5412B/TM56F5416B Description<br>Page 64 Add Difference between TM56F5412/TM56F5416 and TM56F5412B/TM56F5416B<br>Page 16 modify Read EEPROM demo code<br>Page 96 modify SUBLW description |
| 0.94 | Apr, 2022 | Page10 Added MSOP10/QFN20 wire bonding<br>Page12 Added MSOP10/QFN20 pin summary<br>Page104 Added MSOP10/QFN20 ordering information<br>page 109/110 Added MSOP10/QFN20 package dimension |

# CONTENTS

# FEATURES

1. **ROM: 4K x 16 bits Flash Program Memory with Page Locker function**

2. **EEPROM: 128 x 8 bits**

3. **RAM: 336 x 8 bits**

4. **STACK: 8 Levels**

5. **System Oscillation Sources (Fsys) :**
   - Fast-clock
     - ✧ FIRC (Fast Internal RC) : 16 MHz
   - Slow-clock
     - ✧ SIRC (Slow Internal RC) : 70 KHz @VCC=5V

6. **System Clock Prescaler:**
   - System Oscillation Sources can be divided by 1/2/4/16 as System Clock (Fsys)

7. **Dual System Clock:**
   - FIRC+SIRC

8. **Power Saving Operation Mode**
   - FAST Mode: Slow-clock can be disabled or enabled, Fast-clock keeps CPU running
   - SLOW Mode: Fast-clock can be disabled or enabled, Slow-clock keeps CPU running
   - IDLE Mode: Fast-clock and CPU stop. Slow-clock, T2, or Wake-up Timer keep running
   - STOP Mode: All clocks stop, T2 and Wake-up Timer stop

9. **3 Independent Timers**
   - Timer0
     - ✧ 8-bit timer divided by 1~256 pre-scaler option, Reload/Interrupt/Stop function
   - Timer1
     - ✧ 8-bit timer divided by 1~256 pre-scaler option, Reload/Interrupt/Stop function
     - ✧ Overflow and Toggle out
   - T2
     - ✧ 15-bit timer with 4 interrupt interval time options
     - ✧ IDLE mode wake-up timer or used as one simple 15-bit time base

- ✧ Clock source: Slow-clock (SIRC), Fsys/128

## 10. Interrupt

- ● Three External Interrupt pins
  - ✧ 1 pin is falling edge wake-up triggered & interrupts
  - ✧ 2 pins are rising or falling edge wake-up triggered & interrupt
- ● Timer0/Timer1/T2/WKT (wake-up) Interrupts
- ● TK (Touch Key) /ADC Interrupt
- ● I2C Interrupt
- ● Individual Interrupt Vector

## 11. Wake-up (WKT) Timer

- ● Clocked by built-in RC oscillator with 4 adjustable interrupt times

  16 ms/33 ms/65 ms/130 ms @VCC=3V, 15 ms/29 ms/59 ms/118 ms @VCC=5V

## 12. Watchdog Timer

- ● Clocked by built-in RC oscillator with 4 adjustable reset times

  130 ms/260 ms/1040 ms/2080 ms @VCC=3V, 118 ms/236 ms/944 ms/1888 ms @VCC=5V
- ● Watchdog timer can be disabled/enabled in STOP mode

## 13. PWMx5

- ● PWM0:
  - ✧ 8 bits, duty-adjustable, period-adjustable controlled PWM
  - ✧ PWM0 clock source: Fast-clock or FIRC 16 MHz/32MHz, with 1~64 pre-scalers
- ● PWM1A/PWM1B/PWM1C:
  - ✧ 8 bits, duty-adjustable (Independent) , period-adjustable controlled (Shared) PWM x3
  - ✧ PWM1A/1B/1C clock source (Shared) : Fast-clock or FIRC 16 MHz/32MHz, with 1~64 pre-scalers
- ● PWM2:
  - ✧ 8 bits, duty-adjustable, period-adjustable controlled PWM
  - ✧ PWM2 clock source: Fast-clock or FIRC 16 MHz/32MHz, with 1~64 pre-scalers

**14. 12-bit ADC Converter with 12 input channels and 1 internal reference voltage**

- Internal Bandgap reference voltage1.25V ±2% @25°C, VCC=3V~5V
- ADC reference voltage=VCC/4V/3V/2.5V selection

**15. PB2~PB7 individual pin change wake up**

**16. Reset Sources**

- Power On Reset/Watchdog Reset/Low Voltage Reset/External Pin Reset

**17. Low Voltage Reset (LVR) /Low Voltage Detection Flag (LVD) Option:**

- 4-Level Low Voltage Reset: 2.4V/2.5V/3.2V/3.7V
- 3-Level Low Voltage Detection Flag: 2.5V/3.2V/3.7V ( when LVR = 2.4V)

**18. 16-Channel Touch Key with 3 TK-module (TKM0/TKM1/TKM2)**

- TKM0 and TKM1 with 4-channel touch key, TKM2 with 8-channel touch key
- Each module include:
  - ✧ 3-bit TK reference clock capacitor adjustment
  - ✧ 3-bit touch key clock frequency select(can be fixed frequency or auto change)
  - ✧ 12-bit TK scan length adjustment
- Interrupt/Wake-up CPU while key is pressed

**19. Operating Voltage** :

- Fsys= 1 MHz, LVR ~5.5V
- Fsys=16 MHz, 2.8V~5.5V

**20. Operating Temperature Range : -40°C to + 85°C**

**21. Table Read Instruction: 16-bit ROM data lookup table**

**22. Instruction set: 39 Instructions**

**23. Instruction Execution Time**

- 2 system clocks (Fsys) per instruction except branch

**24. I/O ports: Maximum 26 programmable I/O pins**

- Open-Drain Output
- CMOS Push-Pull Output
- Schmitt Trigger Input with pull-up resistor option

**25. Programming connectivity support 4-wire (ICP) or 5-wire program**

**26. Page Locker Size: 512W/640W/768W …./23040W by 128 words step**

**27. Package Types:**

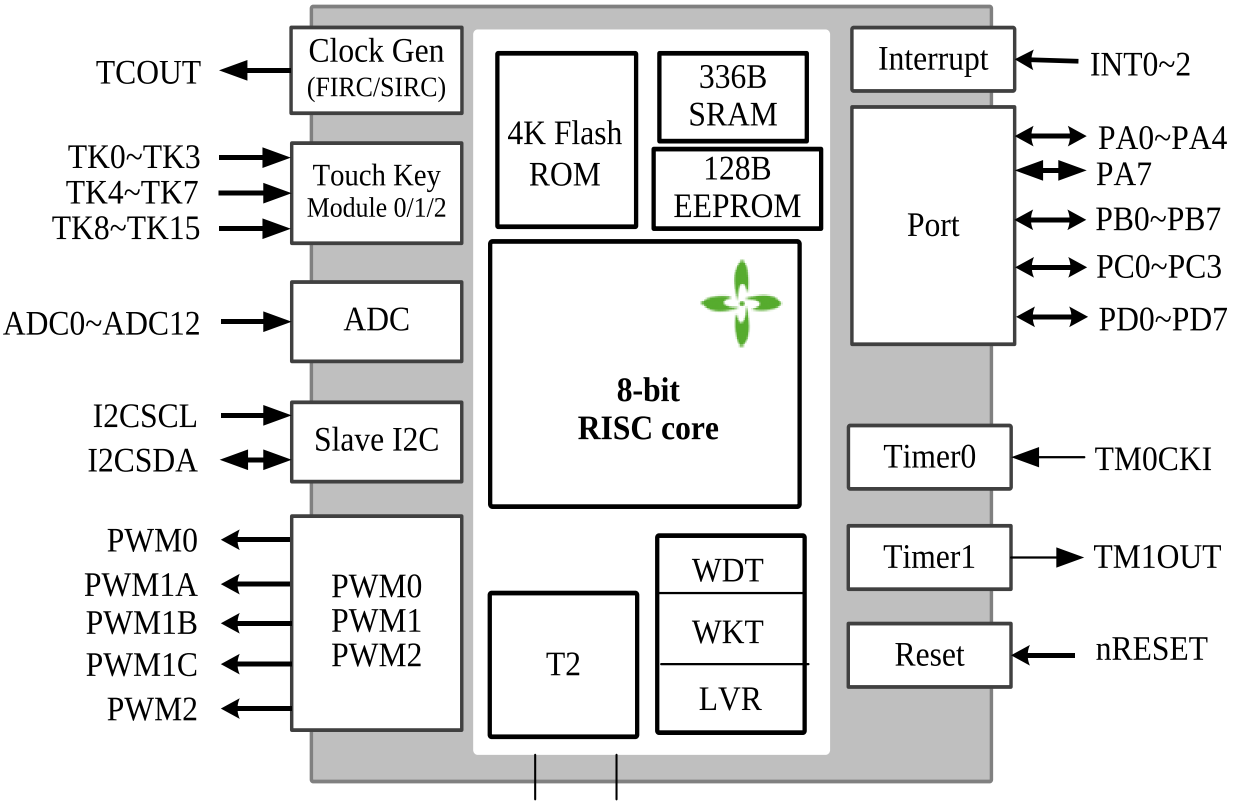
- SOP-28 (TM56F5416/TM56F5416B)

- SOP-24 (TM56F5412/TM56F5412B)
- SOP-20 (TM56F5416/TM56F5416B)
- SOP-16 (TM56F5416/TM56F5416B)
- QFN-20 (TM56F5416B)
- MSOP-10 (TM56F5416B)

**28. Supported EV board on ICE**

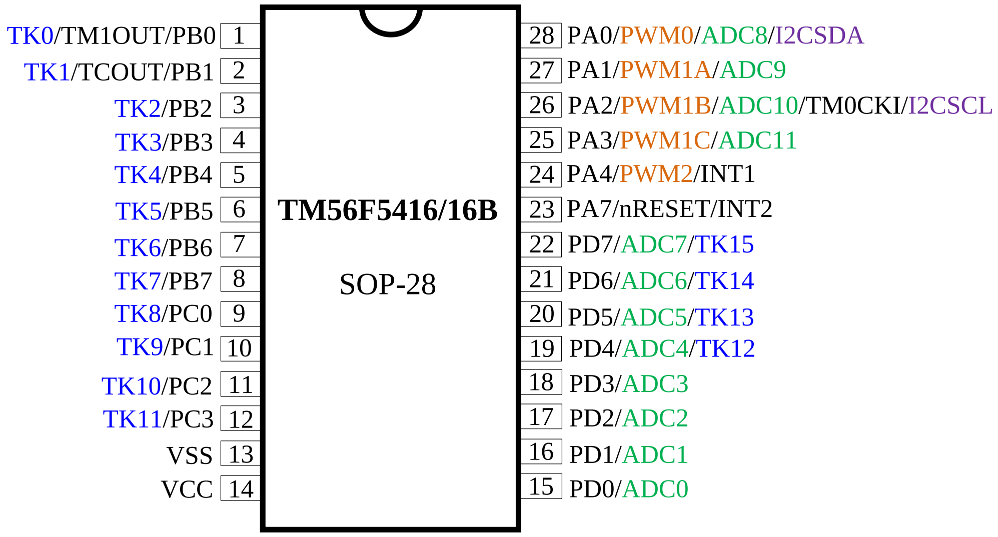
EV board: EV8226

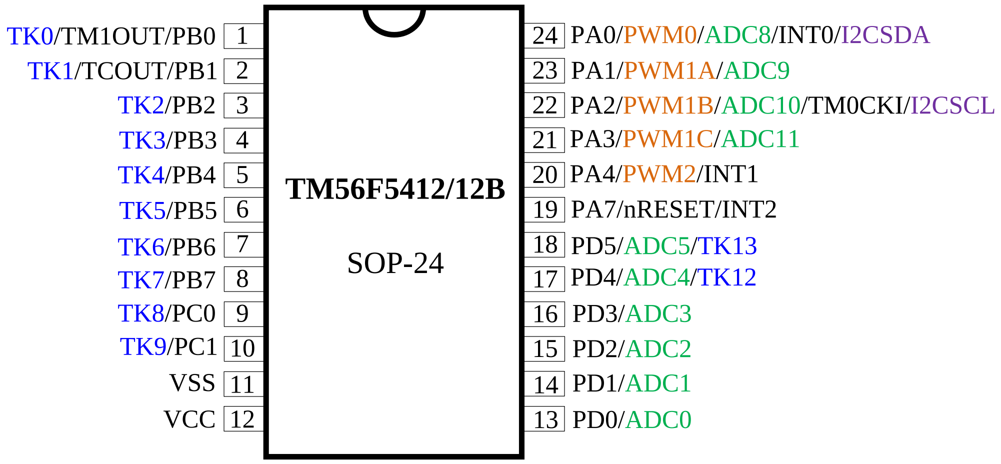
## SYSTEM BLOCK DIAGRAM

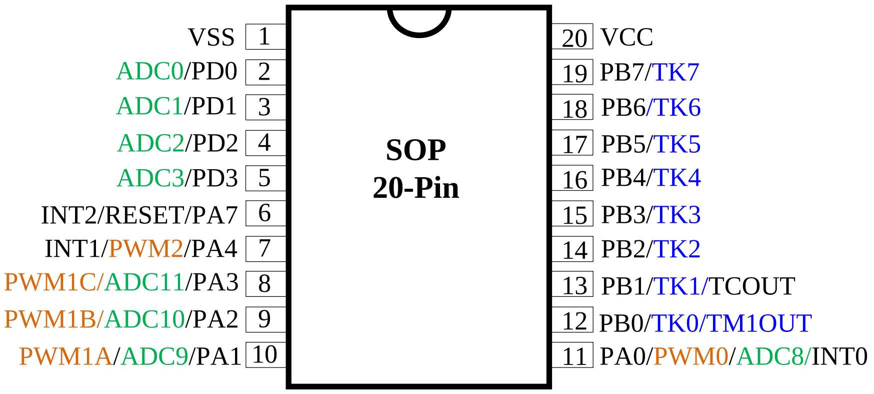


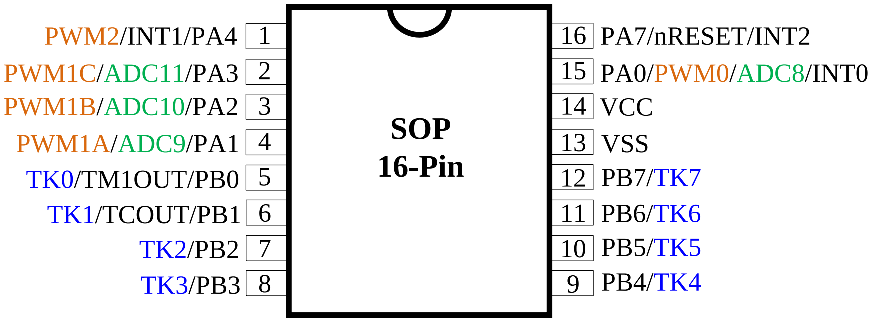
TM56F5412/16/12B/16B Block Diagram

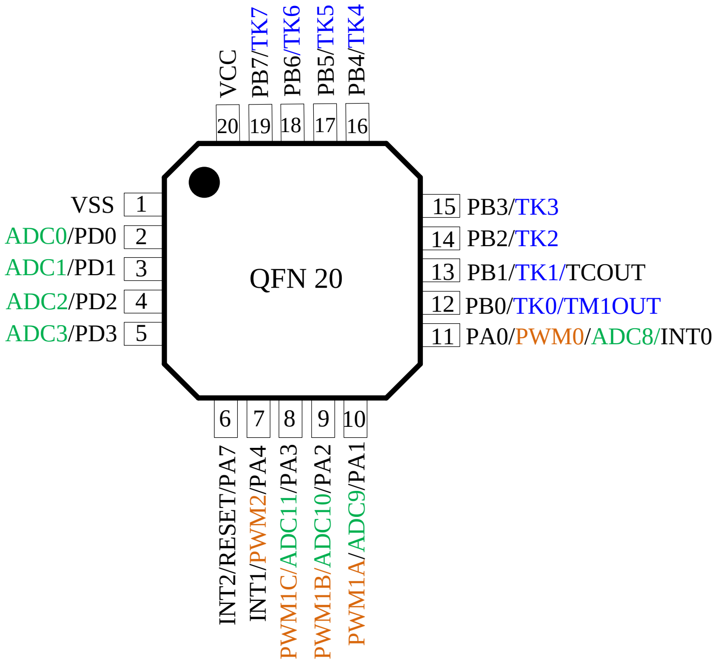
# PIN ASSIGNMENT DIAGRAM

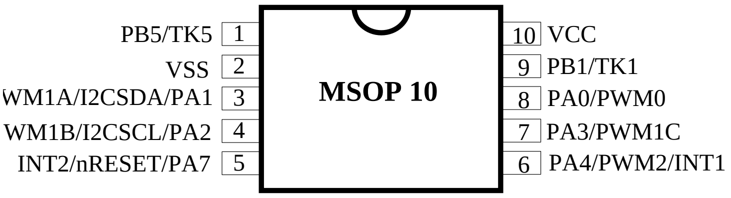





SOP
20-Pin
VSS 1
ADC0/PD0 2
ADC1/PD1 3
ADC2/PD2 4
ADC3/PD3 5
INT2/RESET/PA7 6
INT1/PWM2/PA4 7
PWM1C/ADC11/PA3 8
PWM1B/ADC10/PA2 9
PWM1A/ADC9/PA1 10
20 VCC
19 PB7/TK7
18 PB6/TK6
17 PB5/TK5
16 PB4/TK4
15 PB3/TK3
14 PB2/TK2
13 PB1/TK1/TCOUT
12 PB0/TK0/TM1OUT
11 PA0/PWM0/ADC8/INT0


SOP
16-Pin
PWM2/INT1/PA4 1
PWM1C/ADC11/PA3 2
PWM1B/ADC10/PA2 3
PWM1A/ADC9/PA1 4
TK0/TM1OUT/PB0 5
TK1/TCOUT/PB1 6
TK2/PB2 7
TK3/PB3 8
16 PA7/nRESET/INT2
15 PA0/PWM0/ADC8/INT0
14 VCC
13 VSS
12 PB7/TK7
11 PB6/TK6
10 PB5/TK5
9 PB4/TK4


QFN 20
VCC 20
PB7/TK7 19
PB6/TK6 18
PB5/TK5 17
PB4/TK4 16
VSS 1
ADC0/PD0 2
ADC1/PD1 3
ADC2/PD2 4
ADC3/PD3 5
15 PB3/TK3
14 PB2/TK2
13 PB1/TK1/TCOUT
12 PB0/TK0/TM1OUT
11 PA0/PWM0/ADC8/INT0
6 INT2/RESET/PA7
7 INT1/PWM2/PA4
8 PWM1C/ADC11/PA3
9 PWM1B/ADC10/PA2
10 PWM1A/ADC9/PA1


MSOP 10
PB5/TK5 1
VSS 2
WM1A/I2CSDA/PA1 3
WM1B/I2CSCL/PA2 4
INT2/nRESET/PA7 5
10 VCC
9 PB1/TK1
8 PA0/PWM0
7 PA3/PWM1C
6 PA4/PWM2/INT1

# PIN DESCRIPTIONS

| Name | In/Out | Pin Description |
|---|---|---|
| PA7, PA4–PA0 | I/O | Bit-programmable I/O port for Schmitt-trigger input, CMOS push-pull output or open-drain output. Pull-up resistor are assignable by software. |
| nRESET | I | External active low reset/ Schmitt-trigger input |
| VCC, VSS | P | Power input pin and ground |
| INT0~INT2 | I | External interrupt input |
| PB7-PB0 | I/O | Bit-programmable I/O port for Schmitt-trigger input, CMOS push-pull output or open-drain output. Pull-up resistor are assignable by software. |
| PC3-PC0 | I/O | Bit-programmable I/O port for Schmitt-trigger input, CMOS push-pull output or open-drain output. Pull-up resistor are assignable by software. |
| PD7-PD0 | I/O | Bit-programmable I/O port for Schmitt-trigger input, CMOS push-pull output or open-drain output. Pull-up resistor are assignable by software. |
| PWM0<br>PWM1A<br>PWM1B<br>PWM1C<br>PWM2 | O | PWM0/PWM1/PWM2 outputs |
| ADC12~ADC0 | I | Analog to Digital Convert input pin |
| TK3~TK0 | I | Touch Key Module 0 input |
| TK7~TK4 | I | Touch Key Module 1 input |
| TM15~TK8 | I | Touch Key Module 2 input |
| TM0CKI | I | Timer0's input pin in counter mode |
| TCOUT | O | Fsys/2 clock output |
| TM1OUT | O | Timer1 overflow toggle output |
| I2CSCL | I | I2C Serial clock input |
| I2CSDA | I/O | I2C serial data pin |

Programming pins:

Normal mode (5-wire): VCC / VSS / PA0 / PA1 / PA2

ICP mode (4-wire): VCC / VSS / PA0 / PA1 -When using ICP (In-circuit Program) mode, the PCB needs to remove all components of PA0, PA1.

## Pin Summary

| Pin number | | | | | | Pin Name | Type | GPIO | | | | Function after reset | Alternate Function | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | Input | | Output | | | | | | | |
| SOP-28 | SOP-24 | SOP-20 | QFN-20 | SOP16 | MSOP10 | | | Weak Pull-up | Ext. Interrupt | O.D. | P.P. | | PWM | ADC | Touch Key | I2C | MISC |
| 1 | 1 | 12 | 12 | 5 | | PB0/TM1OUT/TK0 | I/O | | | O | O | PB0 | | | O | | TM1OUT |
| 2 | 2 | 13 | 13 | 6 | 9 | PB1/TCOUT/TK1 | I/O | | | O | O | PB1 | | | O | | TCOUT |
| 3 | 3 | 14 | 14 | 7 | | PB2/TK2 | I/O | O | | O | O | PB2 | | | O | | |
| 4 | 4 | 15 | 15 | 8 | | PB3/TK3 | I/O | O | | O | O | PB3 | | | O | | |
| 5 | 5 | 16 | 16 | 9 | | PB4/TK4 | I/O | O | | O | O | PB4 | | | O | | |
| 6 | 6 | 17 | 17 | 10 | | PB5/TK5 | I/O | O | | O | O | PB5 | | | O | | |
| 7 | 7 | 18 | 18 | 11 | | PB6/TK6 | I/O | O | | O | O | PB6 | | | O | | |
| 8 | 8 | 19 | 19 | 12 | | PB7/TK7 | I/O | O | | O | O | PB7 | | | O | | |
| 9 | 9 | | | | | PC0/TK8 | I/O | | | O | O | PC0 | | | O | | |
| 10 | 10 | | | | | PC1/TK9 | I/O | | | O | O | PC1 | | | O | | |
| 11 | | | | | | PC2/TK10 | I/O | | | O | O | PC2 | | | O | | |
| 12 | | | | | | PC3/TK11 | I/O | | | O | O | PC3 | | | O | | |
| 13 | 11 | 1 | 1 | 13 | 2 | VSS | P | | | | | | | | | | |
| 14 | 12 | 20 | 20 | 14 | 10 | VCC | P | | | | | | | | | | |
| 15 | 13 | 2 | 2 | | | PD0/ADC0 | I/O | | | O | O | PD0 | | O | | | |
| 16 | 14 | 3 | 3 | | | PD1/ADC1 | I/O | | | O | O | PD1 | | O | | | |
| 17 | 15 | 4 | 4 | | | PD2/ADC2 | I/O | | | O | O | PD2 | | O | | | TM0OUT |
| 18 | 16 | 5 | 5 | | | PD3/ADC3 | I/O | | | O | O | PD3 | | O | | | |
| 19 | 17 | | | | | PD4/ADC4/TK12 | I/O | | | O | O | PD4 | | O | O | | |
| 20 | 18 | | | | | PD5/ADC5/TK13 | I/O | | | O | O | PD5 | | O | O | | |
| 21 | | | | | | PD6/ADC6/TK14 | I/O | | | | | PD6 | | O | O | | |
| 22 | | | | | | PD7/ADC7/TK15 | I/O | | | | | PD7 | | O | O | | |
| 23 | 19 | 6 | 6 | 16 | 5 | PA7/ INT2/ nRESET | I/O | | O | O | O | PA7 | | | | | nRESET |
| 24 | 20 | 7 | 7 | 1 | 6 | PA4/INT1/ PWM2 | I/O | | O | O | O | PA4 | O | | | | |
| 25 | 21 | 8 | 8 | 2 | 7 | PA3/ADC8/PWM1C | I/O | | | O | O | PA3 | O | O | | | |
| 26 | 22 | 9 | 9 | 3 | 4 | PA2/ADC9/TM0CKI/I2CSCL/PWM1B | I/O | | | O | O | PA2 | O | O | | O | TM0CKI |
| 27 | 23 | 10 | 10 | 4 | 3 | PA1/ADC10/PWM1A | I/O | | | O | O | PA1 | O | O | | | |
| 28 | 24 | 11 | 11 | 15 | 8 | PA0/ADC11/INT0/I2CSDA/PWM0 | I/O | | O | O | O | PA0 | O | O | | O | |

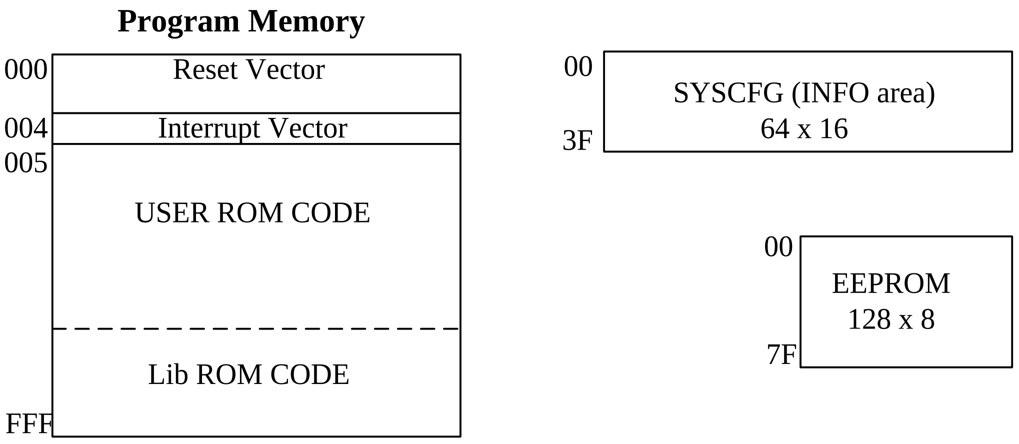
Symbol：O.D. = Open Drain
P.P. = Push-Pull Output

# FUNCTION DESCRIPTION

## 1. CPU Core

### 1.1 Program ROM (PROM)

The Flash Program ROM of this device is 4K words, with an extra 64-Word INFO area to store the SYSCFG and an extra 128-Byte EEPROM. The ROM can be written multi-times and can be read as long as the PROTECT and LPROT bit of SYSCFG are not set. The SYSCFG can be read no matter PROTECT or LPROT is set or cleared, but PROTECT bit can be cleared only when User ROM Code area is erased, and LPROT bit can be cleared only whe the Lib ROM Code area is erased. That is, unprotect the PROTECT or LPROT bit needsto erase the corresponding ROM area. If LPROT bit is set, the ROM can still be written multi-times in the User ROM Code area to update user ROM code again by writer, but the Lib ROM Code area will not be read or written again by writer untill the LPROT bit is cleared. On the other hand, if PROTECT bit is set, the user ROM code area will not be read by writer, and the user ROM code can’t be updated until the PROTECT bit is cleared.



### 1.1.1 Reset Vector (000H)

After reset, system will restart the program counter (PC) at the address 000h, all registers will revert to the default value.

### 1.1.2 Interrupt Vector (004H)

When an interrupt occurs, the program counter (PC) will be pushed onto the stack and jumps to address 004H.

## 1.2 System Configuration Register (SYSCFG)

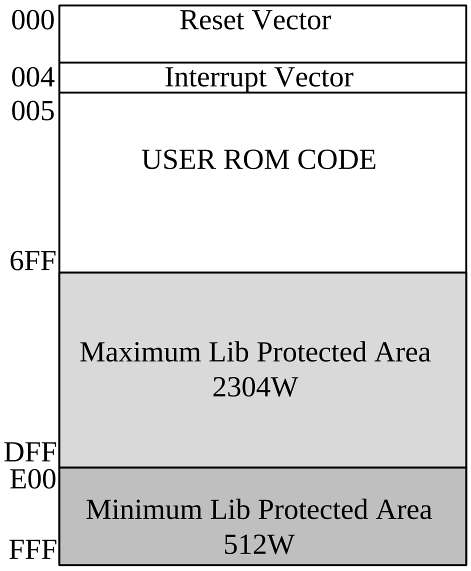
The System Configuration Register (SYSCFG) is located at Flash INFO area; it contains two 13bits registers (CFGWL/CFGWH). The SYSCFG determines the option for initial condition of CPU. It is written by PROM Writer only. User can select LVR operation Mode and chip operation mode by SYSCFG register. The 13th bit of CFGWH is code protect selection bit . If this bit is 1, the data in PROM will be protected, when user reads PROM.

| **Bit** | | **13~0** | |
|---|---|---|---|
| **Default Value** | | **00000000000000** | |
| **Bit** | | **Description** | |
| **CFGWL** | 13 | **LPROT**: Lib Code protection selection | |
| | | 1 | Enable |
| | | 0 | Disable |
| | 12~9 | **LSIZE**: Lib Size selection | |
| | | 1111 | 2304W |
| | | | |
| | | 0001 | 512W |
| | | 0000 | No use Page locker function |
| | 8~0 | Reserved | |
| **CFGWH** | 13 | **PROTECT**: Code protection selection | |
| | | 1 | Enable |
| | | 0 | Disable |
| | 12 | **XRSTE**: External Pin (PA7) Reset Enable | |
| | | 1 | Enable |
| | | 0 | Disable |
| | 11~10 | **LVR**: Low Voltage Reset Mode | |
| | | 11 | LVR 3.7V (without LVD function) |
| | | 10 | LVR 3.2V (without LVD function) |
| | | 01 | LVR 2.5V (without LVD function) |
| | | 00 | LVR 2.4V (with LVD function, LVDS 00 : 3.2V 01: 2.5V 1X: 3.7V) |
| | 9~8 | **WDTE**: WDT Reset Enable | |
| | | 11 | Always Enable |
| | | 10 | Enable in FAST/SLOW mode, Disable in IDLE/STOP mode |
| | | 0X | Disable |
| | 7~0 | Reserved | |

## 1.3 Page Lock Function

TM56F5412/16/12B/16B support Page locker function. By setting LPROT (CFGWL.13), user can choose whether to turn it on. If the user A (library code provider) turns this function on, the user A (library code provider) can select different size (512~2304W) of lib protected area by LSZIE (CFGWL 12~9). In lib protected area, the user B (firmware developer) can't read ROM code by TABRL/TABRH instruction or in any other way. By using the TICE99IDE tool, the user A can provide a protected lib for the user B to use, but the user B does not kno its details, and the user B still can continue to complete the main code in the unprotected area.

**4K Program ROM**



| LSIZE | Lib Protected Area |
|---|---|
| 2304 | (700H~FFFH) |
| 2176 | (780H~FFFH) |
| 2048 | (800H~FFFH) |
| 1920 | (880H~FFFH) |
| 1792 | (900H~FFFH) |
| 1664 | (980H~FFFH) |
| 1536 | (A00H~FFFH) |
| 1408 | (A80H~FFFH) |
| 1280 | (B00H~FFFH) |
| 1152 | (B80H~FFFH) |
| 1024 | (C00H~FFFH) |
| 896 | (C80H~FFFH) |
| 768 | (D00H~FFFH) |
| 640 | (D80H~FFFH) |
| 512 | (E00H~FFFH) |

## 1.4 EEPROM

The TM56F5412/16/12B/16B contain 128 bytes of data EEPROM memory. It is organized as a separate data space, in which single bytes can be read and written. According the physical characteristic the EEPROM need more long access time than Program ROM. The EEPROM has an endurance of at least 50K write/erase cycle.

**The EEPROM Read usage is** same as use Table Read instruction except EEPROM enable bit must be set to high. By writing 0xE2 to register EEPEN (18Eh) can set the EEPROM enable bit, writing other value to EEPEN (18Eh) will clear the EEPROM enable bit.

◇Example: read EEPROM data @address 23h

```
MOVLW       E2h         ;
MOVWX       EEPEN       ; set EEPROM enable bit
CLRX        DPH         ; set DPH=0 for EEPROM write/read
MOVLW       00h
MOVWX       DPH
MOVLW       23h
MOVWX       DPL         ; set DPTR=0023h
; read EEPROM @Address 23h data into W by using opcode TABRL
TABRL
….

; another way to read EEPROM data into W
MOVLW       01h         ;
MOVWX       TABR        ; TABR=01h=opcode TABRL
MOVXW       TABR        ; Read EEPROM data to W
…
```

**The EEPROM Write usage i**s similar to read EEPROM except the LVRPD must be set to 0x37 to disable LVR. When F/W writes data to the register EEPDT (18Fh), the data will also be written to EEPROM.

◇Example: write EEPROM data A5h to address 23h

```
MOVLW       E2h         ;
MOVWX       EEPEN       ; set EEPROM enable bit
CLRX        DPH         ; set DPH=0 for EEPROM write/read
MOVLW       23h
MOVWX       DPL         ; set DPTR=0023h
MOVLW       00000011b
MOVWX       EEPCTL      ; setEEPROM write with 7.2mS time out
MOVLW       037H        ; W=37H
MOVWX       LVRPD       ; LVRPD = 037H , force LVR disable
                        ; LVR must be disabled before EEP write operation
MOVLW       A5h
MOVWX       EEPDT       ; write data A5h EEPDT (18Fh)
                        ; the data also save to EEPROM @Address 23h
BTXSC       EEPTO       ; check EEPROM write time-out flag
GOTO        TIMEOUT
CLRX        EEPEN       ; protect EEPROM from abnormal write
CLRX        LVRPD       ; LVRPD = 00H , clear LVR disable
```

| 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 |
| Reset | 0 | – | – | – | – | – | 0 | 0 |

18Dh.7 **EEPTO:** EEPROM Write Time-Out flag
18Dh.1~0 **EEPTE:** Write Time-Out enable (Busy wait time)
00:Disable 01: 0.9ms 10:3.6ms 11:7.2ms

| 18Eh | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| EEPEN | EEPEN | | | | | | | |
| R/W | W | | | | | | | |
| Reset | 0 | | | | | | | |

18Eh.7~0 **EEPEN:** EEPROM Access Enable
write 0xE2 to this register will enable EEPROM access
write others value to this register will disable EEPROM access

| 18Fh | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| EEPDT | EEPDT | | | | | | | |
| R/W | W | | | | | | | |
| Reset | 0 | | | | | | | |

18Fh.7~0 **EEPDT:** EEPROM Data to write
write data to this register will let H/W write the data to EEPROM when EEPROM access is enable

| 109h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| LVRPD | LVRPD | | | | | | | |
| R/W | W | | | | | | | |
| Reset | 0 | | | | | | | |

109.7~0 **LVRPD:** LVR power down register
Write 0x37 to force LVR be disabled. (LVR must be disabled before EEPROM Write operation)
Write other value to clear LVR disable.

## 1.5 RAM Addressing Mode

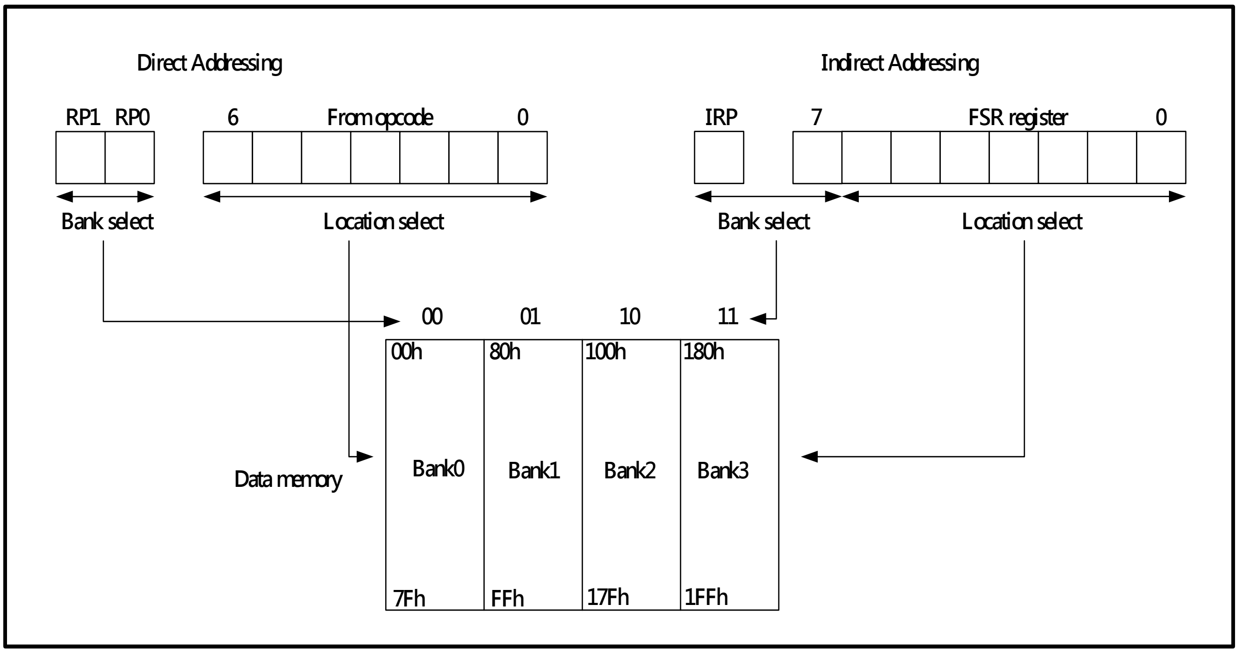
There is one Data Memory Plane in CPU. The Plane is partitioned into four banks. Each bank extends up to 7Fh (128 bytes). The lower locations of each bank are reserved for Special Function Register (SFR). Above the SFR are General Purpose Registers, implemented as static RAM. All implemented banks contain Special Function Registers. Some frequently used Special Function Registers from one bank may be mirrored in another bank for code reduction and quicker access.

Bit RP1 and RP0 (STATUS[6:5]) are the bank select bit

| RP1 : RP0 | BANK |
|---|---|
| 00 | 0 |
| 01 | 1 |
| 10 | 2 |
| 11 | 3 |

The plane can be addressed directly or indirectly. The INDF register is not a physical register. Addressing the INDF register will cause indirect addressing. Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses the register pointed to by the

File Select Register, FSR. Reading the INDF register itself, indirectly (FSR = ‘0’) will read 00h. Writing to the NDF register indirectly results in a no operation (althought status bit may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bir (STATUS<7>). Refer to the figure below.



**Direct / Indirect Addressing**

Keeping RP0=RP1=0 in the begining of the F/W code and using the new instruction set. The advantage of using new instruction is user can ignore the bank location of registers and the code size can be saved. The new instruction is almost same as the old instruction. By replacing the “F” to “X” in the instruction set can easily use the new instruction without switching the bank.

For example:

| | | | | |
|---|---|---|---|---|
| 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<br>00~7Fh | | BANK1<br>80h~FFh | | BANK2<br>100h~17Fh | | BANK3<br>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 | TESTREG | 185h | DPL |
| 06h | PBD | 86h | | 106h | | 186h | DPH |
| 07h | PCD | 87h | | 107h | | 187h | |
| 08h | PDD | 88h | | 108h | | 188h | |
| 09h | | 89h | | 109h | LVRPD | 189h | |
| 0Ah | PCLATH | 8Ah | PCLATH | 10Ah | PCLATH | 18Ah | PCLATH |
| 0Bh | INTIE | 8Bh | INTIE | 10Bh | INTIE | 18Bh | INTIE |
| 0Ch | INTIF | 8Ch | PAMODH | 10Ch | | 18Ch | TABR |
| 0Dh | | 8Dh | PAMODL | 10Dh | | 18Dh | EEPCTL |
| 0Eh | | 8Eh | PBMODH | 10Eh | BGTRIM | 18Eh | EEPEN |
| 0Fh | CLKCTL | 8Fh | PBMODL | 10Fh | | 18Fh | EEPDT |
| 10h | TM0RLD | 90h | PCMODL | 110h | I2CTXD0 | 190h | |
| 11h | TM0CTL | 91h | PDMODH | 111h | I2CTXD1 | 191h | TKMCTL0 |
| 12h | TM1 | 92h | PDMODL | 112h | I2CCTL | 192h | TKMCTL1 |
| 13h | TM1RLD | 93h | | 113h | I2CFLG | 193h | TKMCHS |
| 14h | TM1CTL | 94h | PWMCTL1 | 114h | I2CRCD0 | 194h | TKMFLG |
| 15h | T2CTL | 95h | PWM0PRD | 115h | I2CRCD1 | 195h | TKM0DL |
| 16h | MF016 | 96h | PWM1PRD | 116h | | 196h | TKM1DL |
| 17h | ADH | 97h | PWM2PRD | 117h | | 197h | TKM2DL |
| 18h | ADCTL | 98h | MF098 | 118h | | 198h | TKM10DH |
| 19h | MF019 | 99h | | 119h | | 199h | TKM2DH |
| 1Ah | PWM0D | 9Ah | TKM0CON | 11Ah | | 19Ah | TKM0TMRH |
| 1Bh | PWM1AD | 9Bh | TKM1CON | 11Bh | | 19Bh | TKM0TMRL |
| 1Ch | PWM1BD | 9Ch | TKM2CON | 11Ch | | 19Ch | TKM1TMRH |
| 1Dh | PWM1CD | 9Dh | | 11Dh | | 19Dh | TKM1TMRL |
| 1Eh | PWM2D | 9Eh | | 11Eh | | 19Eh | TKM2TMRH |
| 1Fh | PWMCTL0 | 9Fh | | 11Fh | | 19Fh | TKM2TMRL |
| 20h<br>~<br>~<br>6Fh | RAM Bank0<br>area<br>(80 Bytes) | A0h<br>~<br>~<br>EFh | RAM Bank1<br>area<br>(80 Bytes) | 120h<br>~<br>~<br>16Fh | RAM Bank2<br>area<br>(80 Bytes) | 1A0h<br>~<br>~<br>1EFh | RAM Bank3<br>area<br>(80 Bytes) |
| 70h<br>~<br>7Fh | common area<br>16 Bytes | F0h<br>~<br>FFh | accesses<br>70h~7Fh | 170h<br>~<br>17Fh | accesses<br>70h~7Fh | 1F0h<br>~<br>1FFh | accesses<br>70h~7Fh |

◇Example: read/write register by using direct addressing (RP0=RP1=0)

```
TM1             equ    12h    ;SFR in Bank0
PWM1PRD         equ    96h    ;SFR in Bank1
I2CTXD0         equ    110h   ;SFR in Bank2
TKMCTL0         equ    191h   ;SFR in Bank3
RAM20           equ    20h    ;RAM in Bank0
RAMA0           equ    A0h    ;RAM in Bank1
RAM120          equ    120h   ;RAM in Bank2
RAM1A0          equ    1A0h   ;RAM in Bank3

MOVXW   TM1            ; read TM1 (Bank0) to W
MOVXW   PWM1PRD        ; read PWM1PRD (Bank1) to W
MOVXW   I2CTXD0        ; read I2CTXD0 (Bank2) to W
MOVXW   TKMCTL0        ; read TKMCTL0 (Bank4) to W

MOVLW   16h
MOVWX   RAM20          ; W=16h write to RAM[0x20]
MOVWX   RAMA0          ; W=16h write to RAM[0xA0]
MOVWX   RAM120         ; W=16h write to RAM[0x120]
MOVWX   RAM1A0         ; W=16h write to RAM[0x1A0]
.
MOVLW   037H           ; W=37H
MOVWX    LVRPD         ; LVRPD  = W = 037H , force LVR disable
```

◇Example: read/write register by using indirect addressing  (RP0=RP1=0, IRP=1)

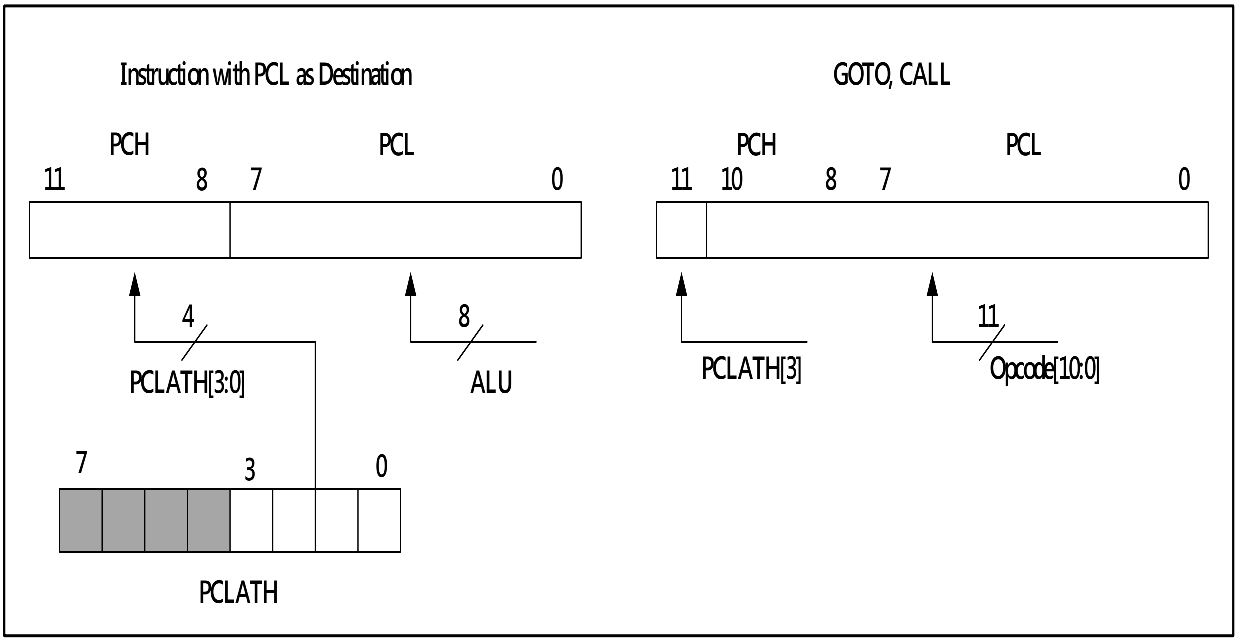
```
MOVLW   PWM2PRD        ; W=97H
MOVWX   FSR            ; FSR = W =97H
MOVXW   INDF           ; read SFR PWM2PRD (97h) to W
.
BSX     IRP            ; IRP=1 =>Bank2/3
MOVLW   09H            ; W=09H
MOVWX   FSR            ; FSR = W =09H
MOVLW   037H           ; W=37H
MOVWX    INDF          ; LVRPD (109H) = W = 037H
```

◇Example: read/write register by using indirect addressing (RP0=RP1=0, IRP=0)

```
MOVLW   PWM2PRD        ; W=97H
MOVWX   FSR            ; FSR = W =97H
MOVXW   INDF           ; read SFR PWM2PRD (97h) to W
.
BCX     IRP            ; IRP=0 =>Bank0/1
MOVLW   09H            ; W=09H
MOVWX   FSR            ; FSR = W =09H
MOVLW   037H           ; W=37H
MOVWX    INDF          ; LVRPD (109H) = W = 037H
```

## 1.6 Programming Counter (PC) and Stack

The Programming Counter is 12-bit wide capable of addressing a 4K x 16 Flash ROM. The low byte comes from PCL register, which is readable and writable register. The upper bits (PC[11:8]) are not readable, but are indirectly writable through the PCLATH register. On any RESET, the upper bits of the PC will be cleared. As a program instruction is executed, the PC will contain the address of the next program instruction to be executed. The PC value is normally increased by one except the followings. The Reset Vector (000h) and the Interrupt Vector (004h) are provided for PC initialization and Interrupt. For CALL/GOGO instruction, PC loads 12 bit address from instruction word. For RET/RETI/RETLW instruction, PC retrives its content from the top level STACK. Executeing any instruction with the PCL register as the destination simultaneously causes the Program Counter PC[11:8] bits (PCH) to be replaced by the contents of the PCLATH register. This allows the entire contents of the program counter to be changed by writing the desired upper 4 bits to the PCLATH register. When the lower 8 bits are written to the PCL register, all 12 bits of program counter wil change to the values contained in the PCLATH register and those being written to the PCL register.



The STACK is 12-bit wide and 8-level indepth. The CALL instruction and hardware interrupt will push STACK level in order. While the RET/RETI/RETLW instruction pops STACK level in order.

For table lookup, the device offer the powerful table read instructions TABRL, TABRH to return the 16-bit ROM data into W register by setting DPTR={DPH, DPL} registers.

◇ Example: To look up the PROM data located “TABLE1” and “TABLE2”.

```
        ORG       00h          ;  Reset Vector
        GOTO      START
        ORG       04h          ; Interrupt Entry Address
        MOVWX     W_TMP        ;  if HWAUTO =1, can omit this line
        MOVXW     STATUS       ;  if HWAUTO =1, can omit this line
        MOVWX     STATUS_TMP ;  if HWAUTO =1, can omit this line
        MOVXW     PCLATH
        MOVWX     PCLATH_TMP:
         PAGESEL    TM0INT        ; Macro instruction. Because TM0INT_TASK is located at
                    another 2K ROM page, CALL/GOTO only support 2K access.
          BTXSC      TM0IF
          CALL       TM0INT_TASK
          :
          MOVXW      PCLATH_TMP
          MOVWX      PCLATH
          MOVXW      STATUS_TMP ;  if HWAUTO =1, can omit this line
          MOVWX      STATUS      ;  if HWAUTO =1, can omit this line
          SWAPX      W_TMP, f    ;  if HWAUTO =1, can omit this line
          SWAPX      W_TMP       ;  if HWAUTO =1, can omit this line
          MOVXW      W_TMP       ;  if HWAUTO =1, can omit this line
          RETI
START:
          MOVLW      00h
          MOVWX      INDEX       ; Set lookup table's address
LOOP:
          MOVLW      (TABLE1>>8) & 0xff
          MOVWX      PCLATH      ; Instruction with PCL as Destination
          MOVXW      INDEX       ; Move index value to W register
          CALL       TABLE1      ; To lookup data, W=55h
          ……
          BCX    PCLATH,3
          GOTO  LOOP             ; Go to LOOP label
          ……
          MOVLW      (TABLE2>>8)&0xff
          MOVWX      DPH         ; DPH register (F186.3~0)
          MOVLW      (TABLE2)&0xff
          MOVWX      DPL         ; DPL register (F185.7~0)
          TABRL                  ; read PROM low byte data to W (W=86h)
          TABRH                  ; read PROM high byte data to W (W=19h)

          ORG        F68h
TABLE1:
          ADDWX      PCL, 1      ; Add the W with PCL, the result back in PCL
          RETLW      55h         ; W=55h when return
          ……

TABLE2:
          .DT 0x1986, 0x3719, 0x2983   ; 16-bit ROM data
TM0INT_TASK:
          MOVLW      11101111B
          MOVWX      INTIF
          :
          RET
```

```
PAGESEL .MAC   ROMADDR
   MOVLW   (ROMADDR>>8) & 0xFF
   MOVWX   PCLATH
.ENDM
```

***Note:*** The chip defines 256 ROM address as one page, so that ROM has 16 pages, 000H~0FFH, 100H~1FFH, **…** F00H~FFFH. One the other words, PC[11:8] can be defined as page. A lookup table must be located at the same page to avoide getting wrong data. Thus, the lookup table has maximum 255 data for above example with starting a lookup table at X00H (X=1,2,…F). If a lookup table has fewer data, it needs not setting the starting address at X00H, but only confirms all lookup table are located at the same page.

### 1.6.1 ALU and Working (W) Register

The ALU is 8-bit wide and capable of addition, subtraction, shift and logical operations. In two-operand instructions, typically one operand is the W register, which is an 8-bit non-addressable register used for ALU operations. The other operand is either a file register or an immediate constant. In single operand instructions, the operand is either W register or a file register. Depending on the instruction executed, the ALU may affect the values of Carry (C), Digit Carry (DC), and Zero (Z) Flags in the STATUS register. The C and DC flags operate as a /Borrow and /Digit Borrow, respectively, in subtraction.

Note: /Borrow represents inverted of Borrow register.
/Digit Borrow represents inverted of Digit Borrow register

### 1.6.2 STATUS Register (03H/83H/103H/183H)

This register contains the arithmetic status of ALU and the Reset status. The STATUS register can be the destination for any instruction, as with any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. It is recommended, therefore, that only BCX, BSX and MOVWX instructions are used to alter the STATUS Register because these instructions do not affect those bits.

<table>
<tr><th>STATUS</th><th>Bit 7</th><th>Bit 6</th><th>Bit 5</th><th>Bit 4</th><th>Bit 3</th><th>Bit 2</th><th>Bit 1</th><th>Bit 0</th></tr>
<tr><td><b>Reset Value</b></td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td><b>R/W</b></td><td>R/W</td><td>R/W</td><td>R/W</td><td>R</td><td>R</td><td>R/W</td><td>R/W</td><td>R/W</td></tr>
<tr><td>Bit</td><td colspan="8">Description</td></tr>
<tr><td>7</td><td colspan="8"><b>IRP</b>: Register Bank Select bit (used for indirect addressing)<br>0 = Bank 0,1 (00h - FFh)<br>1 = Bank 2,3 (100h - 1FFh)</td></tr>
<tr><td>6:5</td><td colspan="8"><b>RP1:RP0</b>: Register Bank Select bits (used for direct addressing)<br>00 = Bank 0 (00h - 7Fh)<br>01 = Bank 1 (80h - FFh)<br>10 = Bank 2 (100h - 17Fh)<br>11 = Bank 3 (180h - 1FFh)<br>Each bank is 128 bytes</td></tr>
<tr><td>4</td><td colspan="8"><b>TO</b>: Time Out Flag<br>0: after Power On Reset, LVR Reset or CLRWDT/SLEEP instruction<br>1: WDT time out occurs</td></tr>
<tr><td>3</td><td colspan="8"><b>PD</b>: Power Down Flag<br>0: after Power On Reset, LVR Reset or CLRWDT instruction<br>1: after SLEEP instruction</td></tr>
<tr><td>2</td><td colspan="8"><b>Z</b>: Zero Flag<br>0: the result of a logic operation is not zero<br>1: the result of a logic operation is zero</td></tr>
<tr><td rowspan="3">1</td><td colspan="8"><b>DC</b>: Decimal Carry Flag or Decimal / Borrow Flag</td></tr>
<tr><td colspan="4">ADD instruction</td><td colspan="4">SUB instruction</td></tr>
<tr><td colspan="4">0: no carry<br>1: a carry from the low nibble bits of the result occurs</td><td colspan="4">0: a borrow from the low nibble bits of the result occurs<br>1: no borrow</td></tr>
<tr><td rowspan="3">0</td><td colspan="8"><b>C</b>: Carry Flag or /Borrow Flag</td></tr>
<tr><td colspan="4">ADD instruction</td><td colspan="4">SUB instruction</td></tr>
<tr><td colspan="4">0: no carry<br>1: a carry occurs from the MSB</td><td colspan="4">0: a borrow occurs from the MSB<br>1: no borrow</td></tr>
</table>

◇ Example: Write immediate data into STATUS register.

```
MOVLW    00h
MOVWX     STATUS          ; Clear STATUS register
```

◇ Example: Bit addressing set and clear STATUS register.

```
BSX        STATUS, 0      ; Set C=1.
BCX        STATUS, 0      ; Clear C=0.
```

◇ Example: Determine the C flag by BTXSS instruction.

```
BTXSS      STATUS, 0      ; Check the carry flag
GOTO       LABEL_1        ; If C=0, goto label_1
GOTO       LABEL_2        ; If C=1, goto label_2
```

## 2. Reset

This device can be RESET in four ways.

- Power-On-Reset (POR)
- Low Voltage Reset (LVR)
- External Pin Reset (PA7)
- Watchdog Reset (WDT)

Resets can be caused by Power on Reset (POR) , External Pin Reset (XRST) , Watchdog Timer Reset (WDTR) , or Low Voltage Reset (LVR) . The CFGWH controls the Reset functionality. After Reset, the SFRs are returned to their default value, the program counter (PC) is cleared , and the system starts running from the reset vector 000H place. The TO and PD flags at status register (STATUS) are indicate system reset status.

### 2.1 Power on Reset

After Power-On-Reset, all system and peripheral control registers are then set to their default hardware Reset values. The clock source, LVR level and chip operation mode are selected by the SYSCFG register value.

### 2.2 Low Voltage Reset

The Low Voltage Reset features static reset when supply voltage is below a threshold level. There are three threshold levels can be selected. The LVR's operation mode is defined by the SYSCFG register. See the following LVR Selection Table; user must also consider the lowest operating voltage of operating frequency.

LVR Selection Table:

| LVR level | Operating voltage |
|---|---|
| LVR2.4 | 5.5V > VCC > 2.4V |
| LVR2.5 | 5.5V > VCC > 2.5V |
| LVR3.2 | 5.5V > VCC > 3.2V |
| LVR3.7 | 5.5V > VCC > 3.7V or $V_{CC}$ =5.0V |

Different Fsys have different system minimum operating voltage, reference to Operating Voltage of DC characteristics, if current system voltage is low than minimum operating voltage and lower LVR is selected, then the system maybe enters dead-band and error occurs.

### 2.3 External Pin Reset

The External Pin Reset can be disabled or enabled by the SYSCFG register. It needs to keep at least 2 SIRC clock cycle long to be seen by the chip. XRST also set all the control registers to their default reset value. The TO/PD flags are not affected by these resets.

## 2.4 Watchdog Timer Reset

WDT overflow Reset can be disabled or enabled by the SYSCFG register. It runs in Fast/Slow mode and runs or stops in IDLE/STOP mode. WDT overflow speed can be defined by WDTPSC SFR. WDT is cleared by device Reset or CLRWDT SFR bit WDT overflow Reset also set all the control registers to their default reset value. The TO/PD flags are not affected by these resets.

◇ Example: Defining Reset Vector

```
          ORG       000H
          GOTO      START          ; Jump to user program address.

          ORG       010H

  START:
          …                        ; 010H, The head of user program
          …
          GOTO      START
```

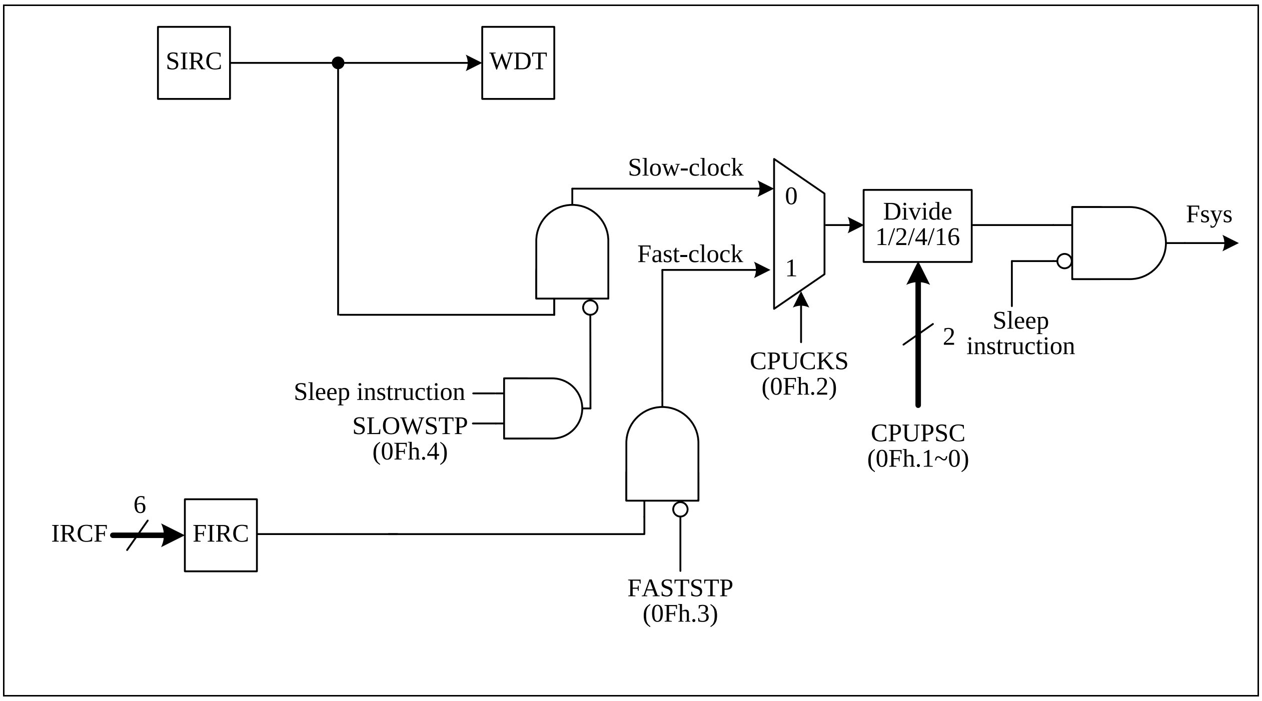
# 3. Clock Circuitry and Operation Mode

## 3.1 System Clock

The device is designed with dual-clock system. There are two kinds of clock source, i.e. SIRC (Slow Internal RC) , and FIRC (Fast Internal RC) . Each clock source can be applied to CPU kernel as system clock. When in IDLE mode, only Slow-clock can be configured to keep oscillating to provide clock source to T2 block. Refer to the figure below.

After Reset, the device is running at Slow mode with 80 KHz SIRC. S/W should select the proper clock rate for chip operation safety. The higher $V_{CC}$ allows the chip to run at a higher System clock frequency. In a typical condition, an 16 MHz System clock rate requires $V_{CC}$ > 2.8V.

The CLKCTL SFR controls the System clock operating. H/W automatically blocks the S/W abnormally setting for this register. S/W can only change the Slow-clock type in Fast mode and change the Fast-clock type in Slow mode. Never to write both FASTSTP=1 & CPUCKS=1. It is recommended to write this SFR bit by bit.



**Clock Scheme Block Diagram**

The frequency of FIRC (Fast Internal RC) can be adjusted by IRCF and IRCF is trimmed to FIRC=16 MHz in chip manufacturing

| 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 |
| Reset | – | – | – | 0 | 1 | 0 | 1 | 1 |

0Fh.4 **SLOWSTP:** Stop Slow-clock in Stop Mode
0: no Stop
1: Stop Slock-clock

0Fh.3 **FASTSTP:** Stop Fast Clock
0: Fast Clock Running
1: Fast Clock Stop

0Fh.2 **CPUCKS:** System Clock selection
0: Slow Clock as system clock
1: Fast Clock as system clock

0Fh.1~0 **CPUPSC:** System clock Prescaler
0: div 16
1: div 4
2: div 2
3: div 1

### FAST Mode:

In this mode, the program is executed using Fast-clock as CPU clock (Fsys) . The Timer0, Timer1 blocks are also driven by Fast-clock, The PWM0/PWM1 block can driven by FIRC16M, FIRC32M or Fsys. T2 can be driven by Slow-clock or Fsys/128 by setting T2CKS (15h.2).

### SLOW Mode:

After power-on or reset, device enters SLOW mode, the default Slow-clock is SIRC. In this mode, the Fast-clock can stopped (by FASTSTP=1, for power saving) or running (by FASTSTP=0) , and Slow-clock is enabled. All peripheral blocks (Timer0, Timer1etc...) clock sources are Slow-clock in the SLOW mode.

### IDLE Mode:

If Slow-clock is enabled (SLOWSTP=0) and T2CKS=0 before executing the SLEEP instruction, the CPU enters the IDLE mode. In this mode, the Slow-clock source keeps T2 block running. CPU stop fetching code and all blocks are stop except T2 related circuits. Idle mode is terminated by Reset or enabled Interrupts wake up.

Another way to keep clock oscillation in IDLE mode is setting WKTIE=1 before executing the SLEEP instruction. In such condition, the WKT keeps working and wake up CPU periodically.

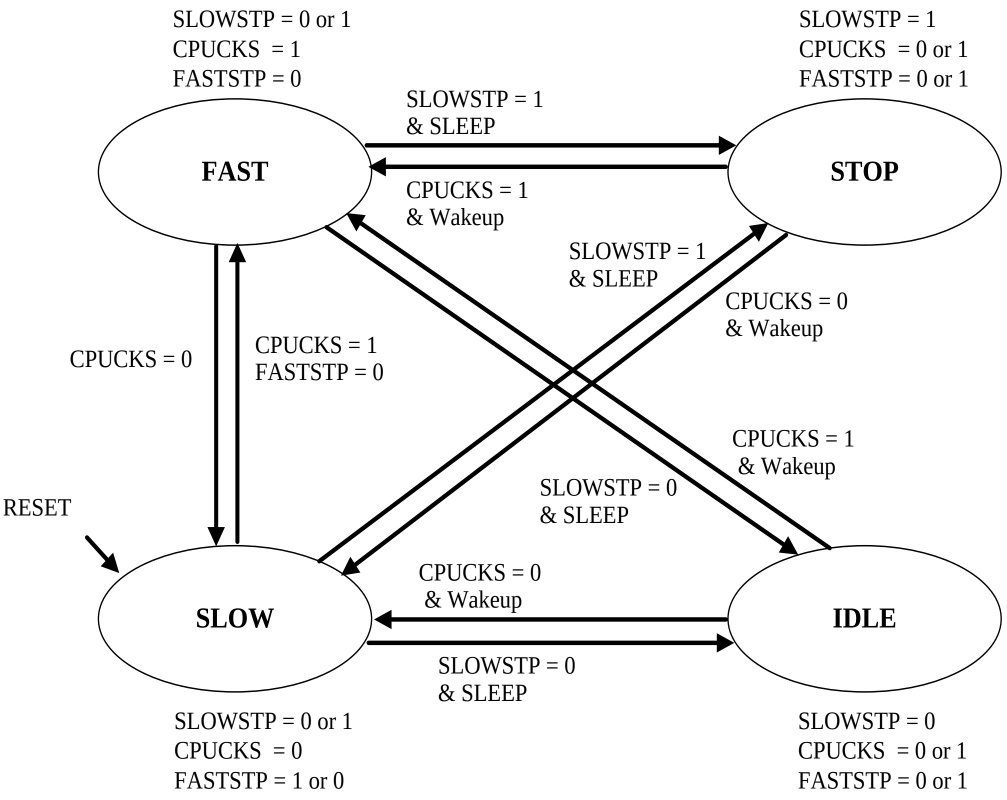
T2 and WKT/WDT are independent and have their own control registers. It is possible to keep both T2 and WKT working and wake-up in the IDLE mode.

### STOP Mode:

If Slow-clock and WKT/WDT are disabled before executing the SLEEP instruction, every block is turned off and the device enters the STOP mode. STOP mode is similar to IDLE mode. The difference is all clock oscillators either Fast-clock or Slow-clock is power down and no clock is generated.

## 3.2 Dual System Clock Modes Transition

The device is operated in one of four modes: FAST mode, SLOW mode, IDLE mode, and STOP mode.



Note:

- ✧ SLEEP denotes SLEEP instruction
- ✧ Wakeup denotes wake-up events, such as External pin interrupt, WKT interrupt or TM3 Interrupt.
- ✧

**CPU Operation Block Diagram**

CPU Mode & Clock Functions Table:

| Mode | Oscillator | Fsys | Fast-clock | Slow-clock | TM0/TM1 | T2 | Wakeup event |
|---|---|---|---|---|---|---|---|
| FAST | FIRC | Fast-clock | Run | Set by SLOWSTP | Run | Run | X |
| SLOW | SIRC | Slow-clock | Set by FASTSTP | Run | Run | Run | X |
| IDLE | SIRC | Stop | Stop | Run | Stop | Run | WKT/IO/T2 |
| STOP | Stop | Stop | Stop | Stop | Stop | Stop | IO |

- **FAST mode switches to SLOW mode**

The following steps are suggested to be executed by order when FAST mode switches to SLOW mode:

(1) Enable Slow-clock (SLOWSTP=0)

(2) Switch to Slow-clock (CPUCKS=0)

(3) Stop Fast-clock (FASTSTP=1)

◇ Example: Switch FAST mode to SLOW mode.

```
BCX      SLOWSTP     ; Enable Slow-clock.
NOP
BCX      CPUCKS      ; Fsys=Slow-clock.
BSX      FASTSTP     ; Disable Fast-clock.
```

- **SLOW mode switches to FAST mode**

SLOW mode can be enabled by CPUCKS=0 in CLKCTL register. The following steps are suggested to be executed by order when SLOW mode switches to FAST mode:

(1) Enable Fast-clock (FASTSTP=0)

(2) Switch to Fast-clock (CPUCKS=1)

◇ Example: Switch SLOW mode to FAST mode (The Fast-clock stop).

```
BCX      FASTSTP     ; Enable Fast-clock.
NOP
BSX      CPUCKS      ; Fsys=Fast-clock
```

- **IDLE mode Setting**

The IDLE mode can be configured by following setting in order:

(1) Enable Slow-clock (SLOWSTP=0) or WKT(WKTIE=1)

(2) Switch T2 clock source to Slow-clock (T2CKS=0)

(3) Execute SLEEP instruction

IDLE mode can be waken up by External interrupt, WKT interrupt and TM3 interrupt.

◇ Example: Switch FAST/SLOW mode to IDLE mode.

```
BCX      SLOWSTP      ; Enable Slow-clock.
MOVLW    00000000B
MOVWX    T2CTL        ; T2 Clock source=Slow-clock. T2PSC=div 32768
SLEEP                 ; Enter IDLE mode.
```

**STOP Mode Setting**

The STOP mode can be configured by following setting in order:

(1) Stop Slow-clock (SLOWSTP=1)

(2) Stop WKT/WDT (WKTIE=0, WDTE=10 or 0X)

(3) Execute SLEEP instruction

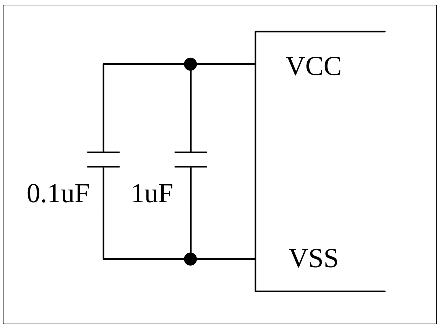
STOP mode can be waken up only by External pin interrupt.

◇ Example: Switch FAST/SLOW mode to STOP mode.

```
BSX      SLOWSTP      ; Disable Slow-clock.
MOVLW    00000000B    ; Disable WKT counting
MOVWX    INTIE
SLEEP                 ; Enter STOP mode.
```

### 3.3 System Clock Oscillator

In the Fast Internal RC (FIRC) mode, the on-chip oscillator generates 16 MHz system clock. Since power noise degrades the performance of Internal Clock Oscillator, placing power supply bypass capacitors 1 uF and 0.1 uF very close to VCC/VSS pins improves the stability of clock and the overall system.



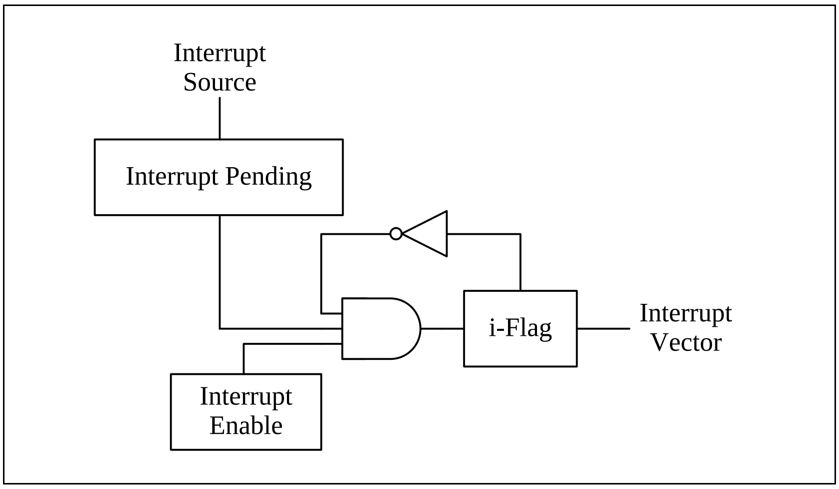
Internal RC Mode

## 4. Interrupt

TM56F5412/16/12B/16B has 1 level, 1 vector and 10 interrupt sources. Each interrupt source has its own enable control bit. An interrupt event will set its individual pending flag, no matter its enable control bit is 0 or 1.

If the corresponding interrupt enable bit (INTIE[7:0], I2CIE, TKIE) has been set, it would trigger CPU to service the interrupt. CPU accepts interrupt in the end of current executed instruction cycle. In the mean while, a "CALL 004" instruction is inserted to CPU, and i-flag is set to prevent recursive interrupt nesting.

The i-flag is cleared in the instruction after the "RETI" instruction. That is, at least one instruction in main program is executed before service the pending interrupt. The interrupt event is level triggered. F/W must clear the interrupt event register while serving the interrupt routine.



| 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

INTIE.7 **ADCIE:** ADC interrupt enable
0: disable
1: enable

INTIE.6 **T2IE:** T2 interrupt enable
0: disable
1: enable

INTIE.5 **TM1IE:** Timer1 interrupt enable
0: disable
1: enable

INTIE.4 **TM0IE:** Timer0 interrupt enable
0: disable
1: enable

INTIE.3 **WKTIE:** Wakeup Timer interrupt enable
0: disable
1: enable

INTIE.2 **INT2IE:** INT2 (PA7) interrupt enable
0: disable
1: enable

INTIE.1 **INT1IE:** INT1 (PA4) interrupt enable
0: disable
1: enable

INTIE.0 **INT0IE:** INT0 (PA0) interrupt enable
0: disable
1: enable

| 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0Ch.7 **ADCIF:** ADC interrupt event pending flag
This bit is set by H/W after end of ADC conversion , write 0 to this bit will clear this flag

0Ch.6 **T2IF:** T2 interrupt event pending flag
This bit is set by H/W while T2 overflows, write 0 to this bit will clear this flag

0Ch.5 **TM1IF:** Timer1 interrupt event pending flag
This bit is set by H/W while Timer1 overflows, write 0 to this bit will clear this flag

0Ch.4 **TM0IF:** Timer0 interrupt event pending flag
This bit is set by H/W while Timer0 overflows, write 0 to this bit will clear this flag

0Ch.3 **WKTIF:** Wakeup Timer interrupt event pending flag
This bit is set by H/W while Wakeup Timer is timeout, write 0 to this bit will clear this flag

0Ch.2 **INT2IF:** INT2 (PA7) pin falling interrupt pending flag
This bit is set by H/W at INT2 pin's falling edge, write 0 to this bit will clear this flag

0Ch.1 **INT1IF:** INT1 (PA4) pin falling/rising interrupt pending flag
This bit is set by H/W at INT1 pin's falling/rising edge, write 0 to this bit will clear this flag

0Ch.0 **INT0IF:** INT0 (PA0) pin falling/rising interrupt pending flag
This bit is set by H/W at INT0 pin's falling/rising edge, write 0 to this bit will clear this flag

| 81h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| OPTON | HWAUTO | INT0EDG | INT1EGE | - | 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.6 INT0EDG**:** INT0 interrupt trigger edge
0: falling edge trigger, 1: rising edge trigger

81h.5 INT1EDG**:** INT1 interrupt trigger edge
0: falling edge trigger, 1: rising edge trigger

| 192h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKMCTL1 | TKMHSEN | TKM2PD | TKM1PD | TKM0PD | TKIE | TKM2JMP | TKM1JMP | TKM0JMP |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

192h.3 **TKIE:** Touch Key interrupt enable
0: disable
1: enable

| 194h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKMFLG | - | TKM2IF | TKM1IF | TKM0IF | TKIF | - | - | - |
| R/W | - | R/W | R/W | R/W | R/W | - | - | - |
| Reset | - | 0 | 0 | 0 | 0 | - | - | - |

194h.6 **TKM2IF:** Touch Key module2 interrupt pending flag,
set by H/W after end of TK2 conversion, write 0 to clear this bit or write 1 to TKM2SOC will clear this flag

194h.5 **TKM1IF:** Touch Key module1 interrupt pending flag,
set by H/W after end of TK1 conversion, write 0 to clear this bit or write 1 to TKM1SOC will clear this flag

194h.4 **TKM0IF:** Touch Key module0 interrupt pending flag,
set by H/W after end of TK0 conversion, write 0 to clear this bit or write 1 to TKM0SOC will clear this flag

194h.3 **TKIF:** Touch Key interrupt pending flag,
set by H/W while TKM0, TKM1 or TKM2 are end of conversion, write 0 to this bit will clear all of Touch Key flag.

| 112h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| I2CCTL | - | - | - | I2CIE | I2CEN | - | I2CID | |
| R/W | - | - | - | R/W | R/W | - | R/W | |
| Reset | - | - | - | 0 | 0 | - | 0 | 0 |

112h.4 **I2CIE:** Slave I2C interrupt enable
0: disable
1: enable

| 113h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| I2CFLG | - | I2CIF | TXD1F | TXD0F | RCD1OVF | RCD1F | RCD0OVF | RCD0F |
| R/W | - | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

113h.6 **I2CIF:** Slave I2C interrupt pending flag
This bis is set by H/W while

✧ I2CRCD0 or I2CRCD1 receive data finished

✧ I2CRCD0 or I2CRCD1 data overflow occurred

✧ I2CTXD0 or I2CTXD1 data transmit finished

Write 0 to this bit will clear this flag and slave I2C related flags.

113h.5 **TXD1F:** Slave I2C transmitting data register 1 flag
This bit is set by H/W while I2CTXD1 data transmitting finished, write 0 to this bit will clear this flag

113h.4 **TXD0F:** Slave I2C transmitting data register 0 flag
This bit is set by H/W while I2CTXD0 data transmitting finished, write 0 to this bit will clear this flag

113h.3 **RCD1OVF:** Slave I2C receiving data register 1 overflow
This bit is set by H/W while receiving I2CRCD1 overflow, write 0 to this bit will clear this flag

113h.2 **RCD1F:** Slave I2C receiving data register 1 flag
This bit is set by H/W while data receiving to I2CRCD1 finished, write 0 to this bit will clear this flag

113h.1 **RCD0OVF:** Slave I2C receiving data register 0 overflow
This bit is set by H/W while receiving I2CRCD0 overflow, write 0 to this bit will clear this flag

113h.0 **RCD0F:** Slave I2C receiving data register 0 flag
This bit is set by H/W while data receiving to I2CRCD0 finished, write 0 to this bit will clear this flag

# 5. I/O Port

## 5.1 PA0-4,PA7, PB0-7, PC0-3 and PD0-7

These pins can be used as Schmitt-trigger input, CMOS push-pull output. The pull-up resistor is assignable to each pin by S/W setting. To use the pin in Schmitt-trigger input mode, S/W needs to set the I/O pin to Mode0 or Mode1 and PxD=1. Reading the pin data (PxD) has different meaning. In "Read-Modify-Write" instruction, CPU actually reads the output data register. In the others instructions, CPU reads the pin state. The so-called "Read-Modify-Write" instruction includes BSX, BCX and all instructions.

These pins can operate in four different modes as below.

| Mode | PA0~PA4, PB, PC, PD<br>pin function | PxD<br>SFR data | Pin State | Resistor<br>Pull-up | Digital Input |
|---|---|---|---|---|---|
| **Mode 0** | Open Drain | 0 | Drive Low | N | N |
| | Input | 1 | Pull-up | Y | Y |
| **Mode 1** | Open Drain | 0 | Drive Low | N | N |
| | | 1 | Hi-Z | N | Y |
| **Mode 2** | CMOS Output | 0 | Drive Low | N | N |
| | | 1 | Drive High | N | N |
| | Touch Key (when TKCHS) | 0 | TK | N | N |
| **Mode 3** | ADC | 0 | – | N | N |
| | Wakeup | 1 | – | Y | Y |
| | | 0 | – | N | Y |

**I/O Pin Function Table**

Beside I/O port function, each pin has one or more alternative functions, such as ADC and Touch Key.

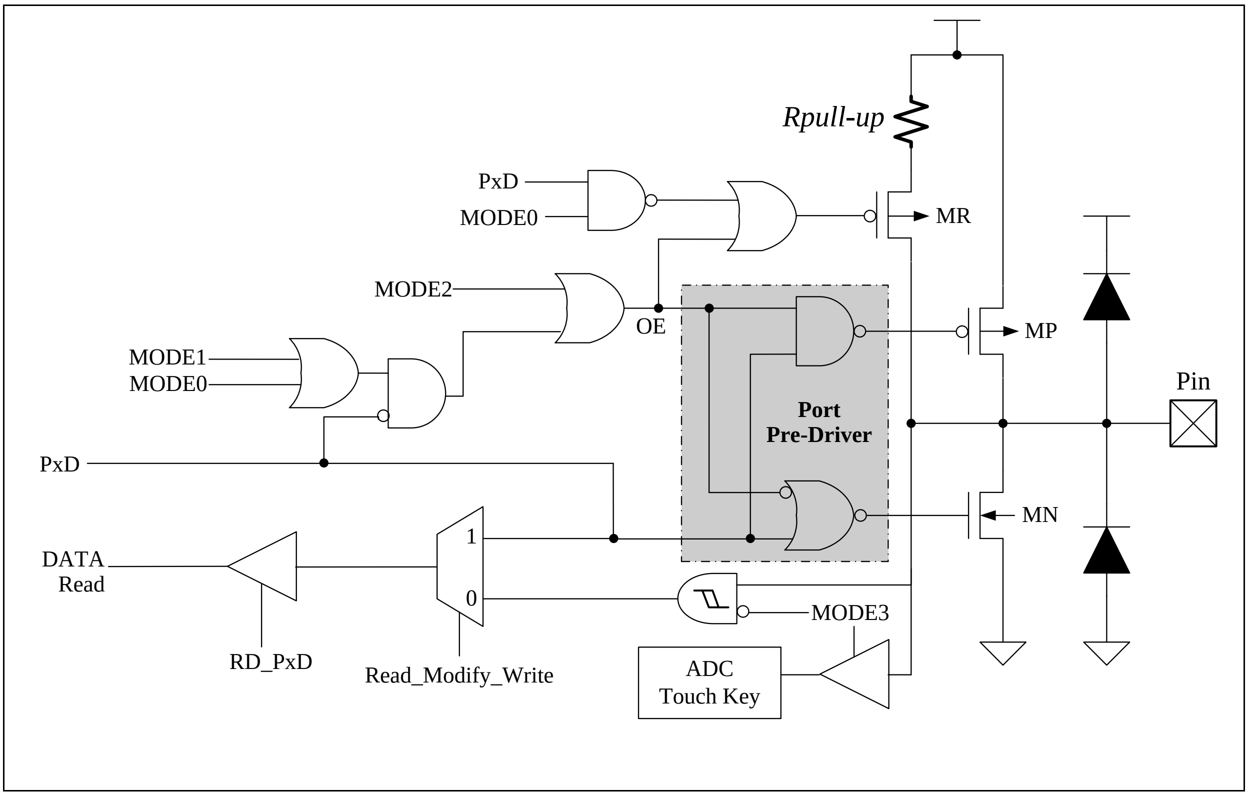
| Pin Name | Wake-up | CKO | ADC/TK | others | Mode3 |
|---|---|---|---|---|---|
| PA0 | INT0 |  | ADC8 | PWM0/I2CSDA | ADC8 |
| PA1 |  |  | ADC9 | PWM1A | ADC9 |
| PA2 |  |  | ADC10 | PWM1B/I2CSDC/ TM0CKI | ADC10 |
| PA3 |  |  | ADC11 | PWM1C | ADC11 |
| PA4 | INT1 |  |  | PWM2 |  |
| PA7 | INT2 |  |  |  |  |
| PB0 |  | TM1OUT | TK0 |  |  |
| PB1 |  | TCOUT | TK1 |  |  |
| PB2 | Wakeup |  | TK2 |  | Wakeup |
| PB3 | Wakeup |  | TK3 |  | Wakeup |
| PB4 | Wakeup |  | TK4 |  | Wakeup |
| PB5 | Wakeup |  | TK5 |  | Wakeup |
| PB6 | Wakeup |  | TK6 |  | Wakeup |
| PB7 | Wakeup |  | TK7 |  | Wakeup |
| PC0 |  |  | TK8 |  |  |
| PC1 |  |  | TK9 |  |  |
| PC2 |  |  | TK10 |  |  |
| PC3 |  |  | TK11 |  |  |
| PD0 |  |  | ADC0 |  | ADC0 |
| PD1 |  |  | ADC1 |  | ADC1 |
| PD2 |  |  | ADC2 |  | ADC2 |
| PD3 |  |  | ADC3 |  | ADC3 |
| PD4 |  |  | ADC4/TK12 |  | ADC4 |
| PD5 |  |  | ADC5/TK13 |  | ADC5 |
| PD6 |  |  | ADC6/TK14 |  | ADC6 |
| PD7 |  |  | ADC7/TK15 |  | ADC7 |

**PortA/B/C/D multi-function Table**

The necessary SFR setting for pin’s alternative function is list below.

| Alternative Function | Mode | PxD SFR data | Pin State | Other necessary SFR setting |
|---|---|---|---|---|
| INT0, INT1<br>TM0CKI | **0** | 1 | Input with Pull-up | INTxIE<br>TM0CTL |
| | **1** | 1 | Input | |
| TK0~TK15 | **2** | 0 | Touch Key Idling, CMOS output Low | TKMxCHS |
| | | | Touch Key Scanning | TKMxCHS |
| AD0~AD11 | **3** | X | ADC Channel | ADCHS |
| PWM0, PWM2,<br>PWM1A, PWM1B, PWM1C | **1** | X | PWM Output (Open Drain) | PWM0OE<br>PWM2OE<br>PWM1AOE<br>PWM1BOE<br>PWM1COE |
| | **2** | X | PWM Output (COMS Output) | |
| I2CSCL | **0** | 1 | Input with Pull-up | I2CCTL |
| | **1** | 1 | Input | |
| I2CSDA | **0** | X | Input with Pull-up/ Open Drain Output | |
| | **1** | X | Input / Open Drain Output | |
| Wake-up | **3** | 0 | Input | |
| | | 1 | Input with Pull-up | |

**Mode Setting for Port Alternative Function**

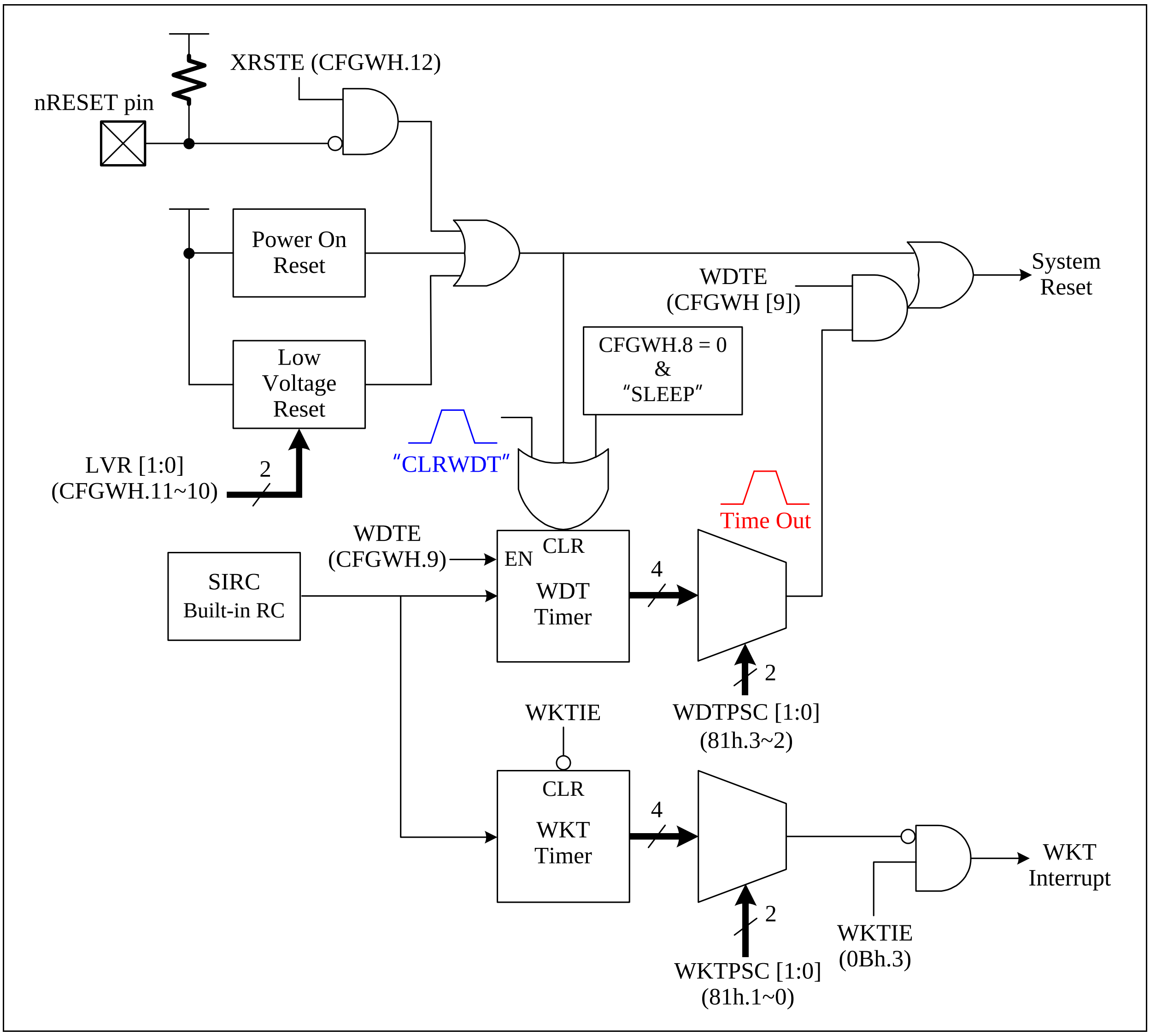


# 6. Peripheral Functional Block

## 6.1 Watchdog (WDT) /Wakeup (WKT) Timer

The WDT and WKT share the same built-in internal RC Oscillator and have individual own counters. The overflow period of WDT, WKT can be selected by individual prescaler (WDTPSC [1:0] , WKTPSC [1:0]) . The WDT timer is cleared by the CLRWDT instruction. If the Watchdog is enabled (CFGWH.9=WDTE=1) , the WDT generates the chip reset signal. Set CFGWH.8 to '0' can let WDT timer stop counting after executing SLEEP instruction, i.e. CFGWH.8=1 WDT timer is always keep counting even if the SLEEP instruction is executed.

The WKT timer is an interval timer, WKT time out will generate WKT Interrupt Flag (WKTIF) . The WKT timer is cleared/stopped by WKTIE=0. Set WKTIE=1, the WKT timer will always count regardless at any CPU operating mode.



**WDT/WKT Block Diagram**

Watchdog clear is controlled by CLRWDT instruction and moving any value into WDTCLR is to clear watchdog timer.

◇ Example: Clear watchdog timer by CLRWDT instruction.

```
 MAIN:
         …                        ; Execute program.
         CLRWDT                   ; Execute CLRWDT instruction.
         …
         GOTO      MAIN
```

◇ Example: Setup WDT time and disable after executing SLEEP instruction.

```
         MOVLW     00000111B
         MOVWX     OPTION         ; Select WDT Time out=256 ms @5V

         SLEEP
```

◇ Example: Set WKT period and interrupt function.

```
         MOVLW     00000110B
         MOVWX     OPTION         ; Select WKT period=64 ms @5V.

         MOVLW     11110111B      ; Clear WKT interrupt request flag by using byte operation
                                  ; Don’t use bit operation “BCX WKTIF” clear interrupt flag
         MOVWX     INTIF          ;

         MOVLW     00001000B      ; Enable WKT interrupt function
         MOVWX     INTIE
```

| 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0Ch.3 **WKTIF:** Wakeup Timer interrupt event pending flag
This bit is set by H/W while Wakeup Timer is timeout, write 0 to this bit will clear this flag

| 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0Bh.3 **WKTIE:** Wakeup Timer interrupt enable
0: disable
1: enable

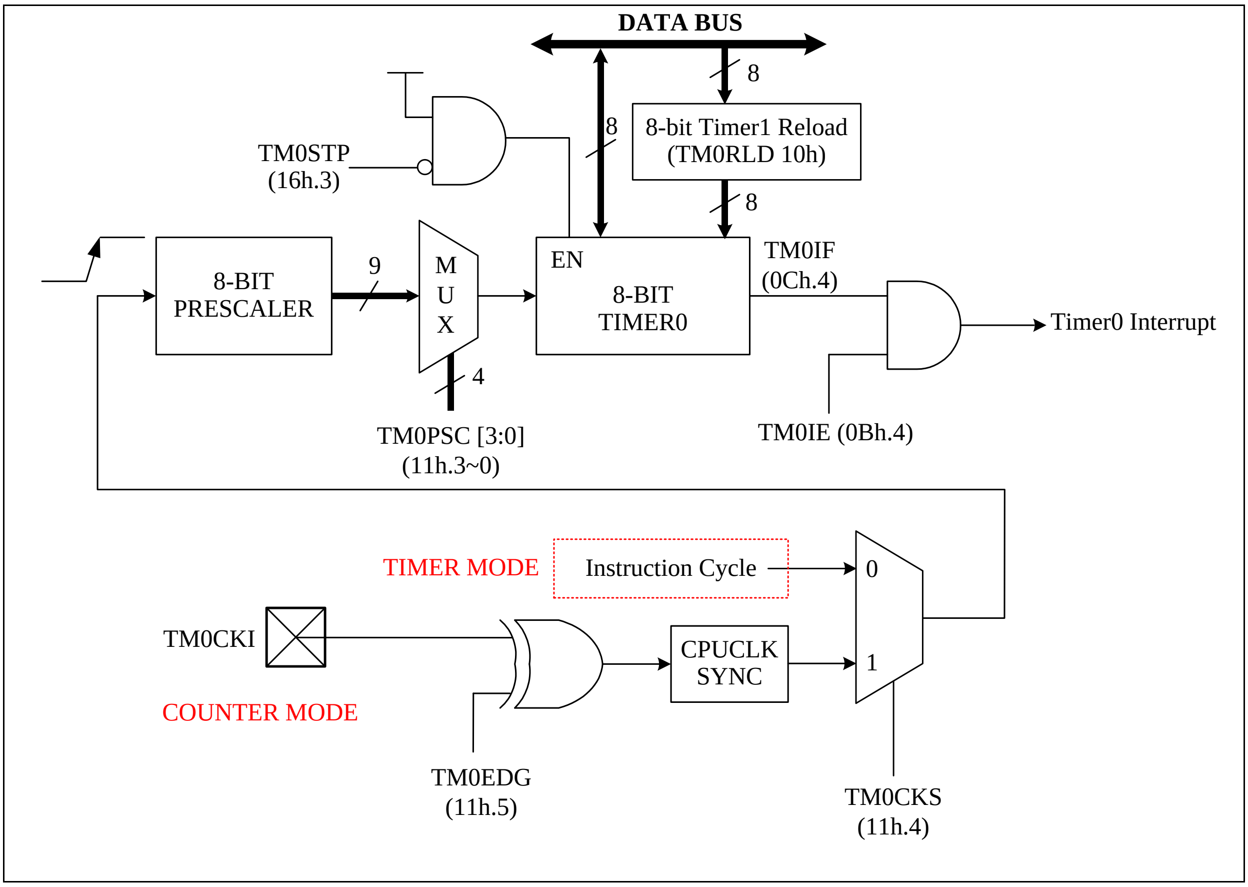
| 81h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| OPTON | HWAUTO | INT0EDG | INT1EGE | - | WDTPSC | | WKTPSC | |
| R/W | R/W | R/W | R/W | - | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | - | 1 | 1 | 1 | 1 |

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

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

## 6.2 Timer0

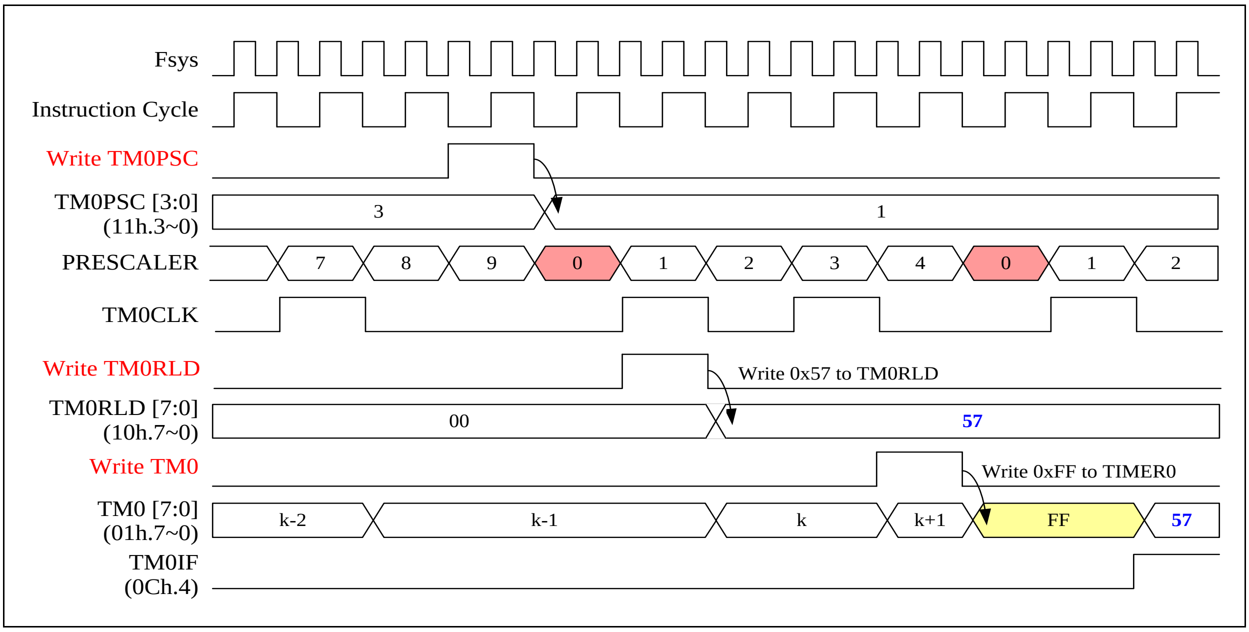
The Timer0 is an 8-bit wide register 01h (TM0). It can be read or written as any other register. Besides, Timer0 increases itself periodically and automatically rolls over a new "offset value" (TM0RLD) while it rolls over based on the pre-scaled clock source, which can be Fsys/2 or TM0CKI (PA2) rising/falling input. The Timer0 increase rate is determined by “Timer0 Pre-Scale” (TM0PSC) register. The Timer0 always generates TM0IF when its count rolls over. It generates Timer0 Interrupt if (TM0IE) is set. Timer0 can be stopped counting if the TM0STP bit is set.



**Timer0 Block Diagram**

The following timing diagram describes the Timer0 works in pure Timer mode.

When the Timer0 prescaler (TM0PSC) is written, the internal 8-bit prescaler will be cleared to 0 to make the counting period correct at the first Timer0 count. TM0CLK is the internal signal that causes the Timer0 to increase by 1 at the end of TM0CLK. TM0WR is also the internal signal that indicates the Timer0 is directly written by instruction; meanwhile, the internal 8-bit prescaler will be cleared. When Timer0 counts from FFh to TM0RLD, TM0IF (Timer0 Interrupt Flag) will be set to 1 and generate interrupt if TM0IE (Timer0 Interrupt Enable) is set.



**Timer0 works in Timer mode (TM0CKS=0)**

The equation of TM0 interrupt time value is as following:

TM0 interrupt interval cycle time=Fsys/2/TM0PSC/ (256-TM0)

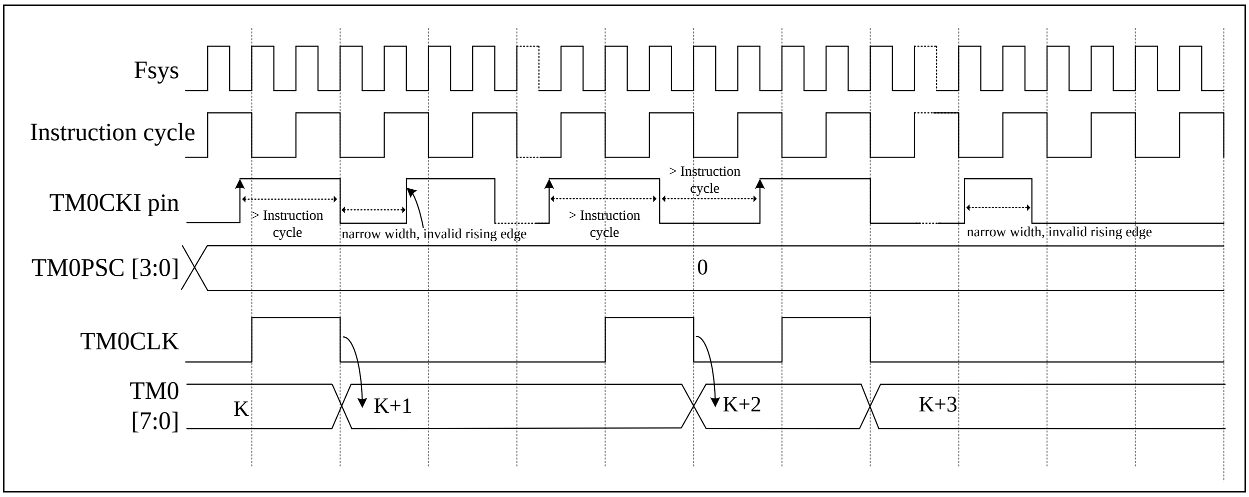
◇ Example: Setup TM0 work in Timer mode

```
; Setup TM0 clock source and divider
        MOVLW   00000101B       ; TM0CKS=0, Setup TM0 clock= Fsys/2
        MOVWX   TM0CTL          ; TM0PSC=5, TM0PSC= Fsys/64
; Set TM0 timer.
        BSX     TM0STP          ; Disable TM0 counting (Default "0").
        MOVLW   156
        MOVWX   TM0             ; Write 156 into TM0 register
        MOVLW   124
        MOVWX   TM0RLD          ; Write 156 into TM0RLD register

; Enable TM0 timer and interrupt function.
        MOVLW   11101111B       ; Clear TM0 request interrupt flag by byte operation
        MOVWX   INTIF           ; 0Ch
        MOVLW   00010000B       ; Enable TM0 interrupt function
        MOVWX   INTIE           ; 0Bh
        BCX     TM0STP          ; Enable TM0 counting (Default “0”).
```

The following timing diagram describes the Timer0 works in Counter mode.

If TM0CKS=1 then Timer0 counter source clock is from TM0CKI pin. TM0CKI signal is synchronized by instruction cycle (Fsys/2) that means the high/low time durations of TM0CKI must be longer than one instruction cycle time (Fsys/2) to guarantee each TM0CKI’s change will be detected correctly by the synchronizer.



**Timer0 works in Counter mode for TM0CKI (TM0EDG=0) , TM0CKS=1**

◇ Example: Setup TM0 work in Counter mode and clock source from TM0CKI pin (PA2)

```
; Setup TM0 clock source from TM0CKI pin (PA2) and divider.
          MOVLW    00110000B
          MOVWX    TM0CTL       ; TM0EDG=1
                                ; Select TM0 prescaler counting edge=falling edge.
                                ; TM0CKS=1, Setup TM0 clock=TM0CKI pin (PA2)
                                ; TM0PSC=0
                                ; TM0 clock prescaler= TM0CKI divided by 1
; Set TM0 timer and stop TM0 counting.
          BSX      TM0STP       ; Disable TM0 counting (Default "0").
          MOVLW    00H
          MOVWX    TM0          ; Write 0 into TM0 register 01H.
; Start TM0 count and read TM0 counter.
          BCX      TM0STP       ; Enable TM0 counting.
          NOP
          NOP
          NOP
          BSX      TM0STP       ; Disable TM0 counting (Default "0")

          MOVXW    TM0
```

| 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

01h **TM0:** Timer0 content

| 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0Bh.4 **TM0IE:** Timer0 interrupt enable
0: disable
1: enable

| 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0Ch.4 **TM0IF:** Timer0 interrupt event pending flag
This bit is set by H/W while Timer0 overflows, write 0 to this bit will clear this flag

| 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

10h **TM0RLD:** Timer0 Reload Data

| 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 |
| Reset | – | – | 0 | 0 | 0 | 0 | 0 | 0 |

11h.5 **TM0EDG:** Timer0 prescaler counting edge for TM0CKI pin
0: rising edge 1: falling edge

11h.4 **TM0CKS:** Timer0 prescaler clock source
0:Fsys/2 1: TM0CKI pin (PA2 pin)

11h.3~0 **TM0PSC:** Timer0 prescaler. Timer0 prescaler clock source divided by

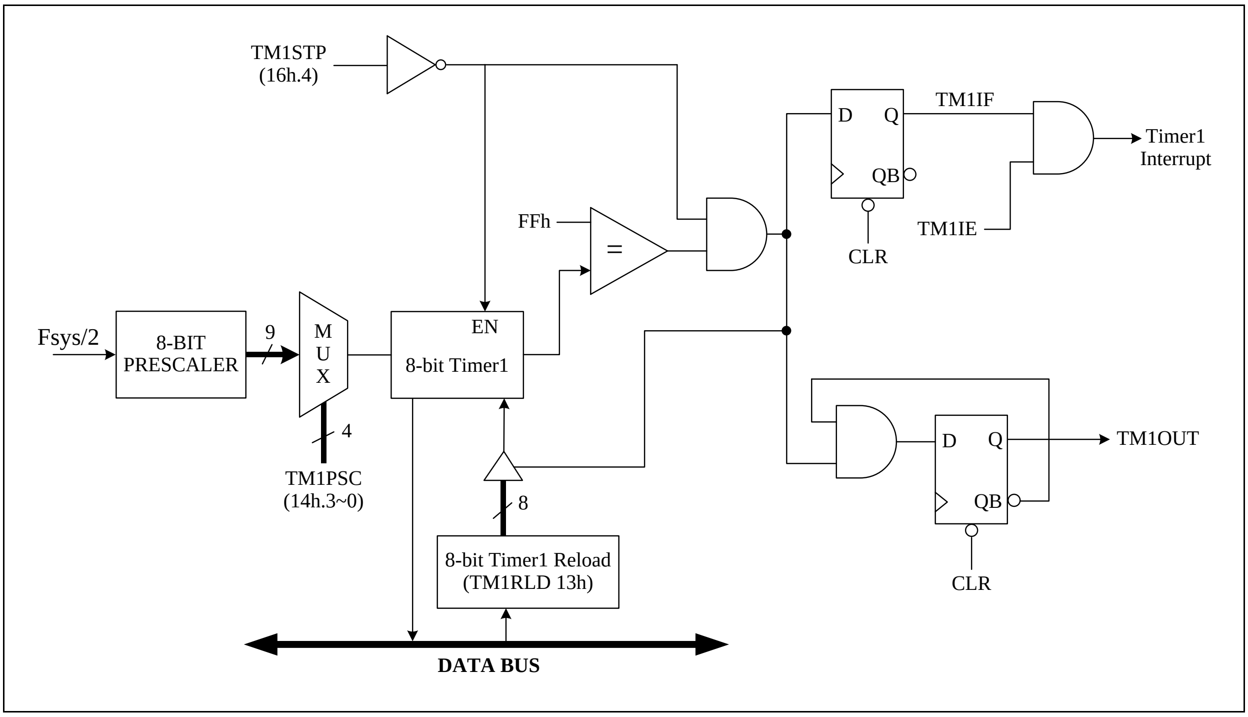
| | | | |
|---|---|---|---|
| 0000: /1 | 0001: /2 | 0010: /4 | 0011: /8 |
| 0100: /16 | 0101: /32 | 0110: /64 | 0111: /128 |
| 1000: /256 | 1001: /512 | 1010: /1024 | 1011: /2048 |
| 1100: /4096 | 1101: /8192 | 1110: /16384 | 1111: /32768 |

| 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 |
| Reset | – | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

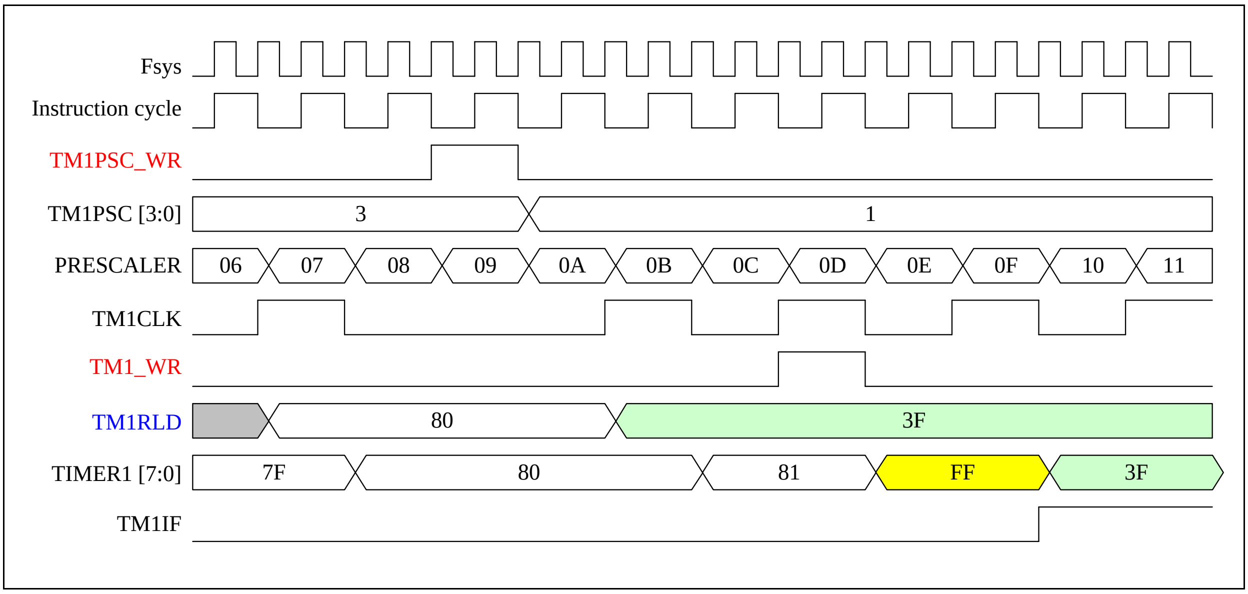
16h.3 **TM0STP:** Timer0 counter stop
0: Release 1: Stop counting

## 6.3 Timer1

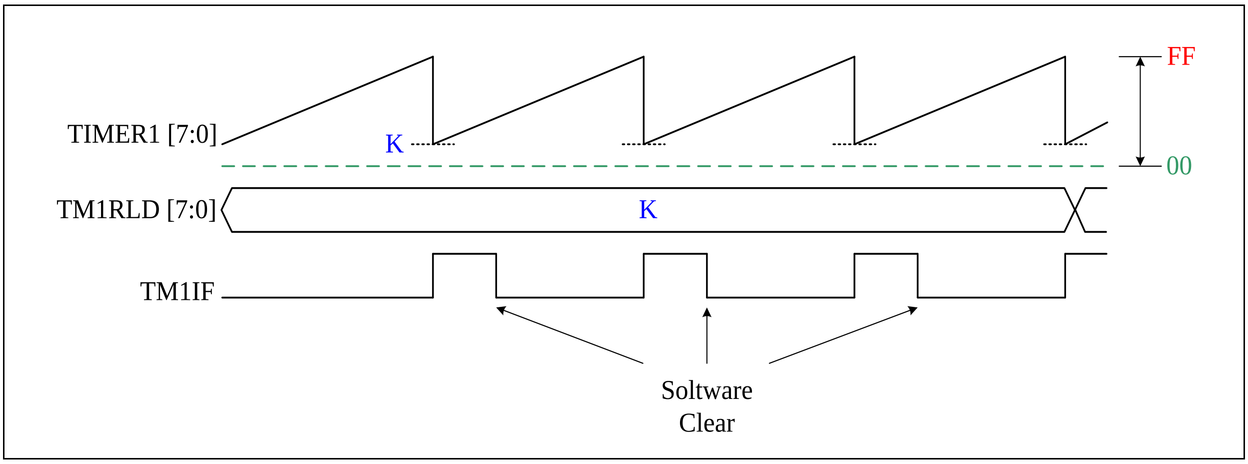
The Timer1 is an 8-bit wide register. It can be read or written as any other register. Besides, Timer1 increases itself periodically and automatically reloads a new "offset value" (TM1RLD) while it rolls over based on the pre-scaled instruction clock (Fsys/2). The Timer1 increase rate is determined by TM1PSC register. Set the TM1STP bit will stop Timer1 counting. TM1OUT is an output signal that toggles when Timer1 overflow.



**Timer1 Block Diagram**



**Timer1 Timing Diagram**



**Timer1 Reload Diagram**

◇ Example: Setup TM1 work in Timer mode and counting overflow toggle out to TM1OUT (PB0) configuration.

```
; Setup TM1 clock source, divider and enable TM1OUT
            MOVLW   00000101B
            MOVWX   TM1CTL          ; TM1PSC=5 , Select TM1 clock=Fsys/64.
            BSX     TM1OE           ; Enable TM1OUT function pin (PB0).

; Set TM1 timer offset and stops TM1 counting
            BSX     TM1STP          ; Stop TM1 counting (Default "0").
            MOVLW   F0H
            MOVWX   TM1             ; Write F0H into TM1 counter

; Enable TM1 timer and interrupt function.
            MOVLW   11011111B       ; Clear TM1 request interrupt flag by byte operation
            MOVWX   INTIF           ; 09H

            MOVLW   00100000B       ; Enable TM1 interrupt function.
            MOVWX   INTIE           ;

            BCX     TM1STP          ; Enable TM1 counting (Default “0”).
```

Example:

Fsys=4 MHz, TM1PSC=1, TM1 clock source=Fsys/4=1 MHz

TM1RLD=0xF0,

TM1 interrupt time= (1/1 MHz) * (0xFF – 0xF0) =1 us*16=16 us

TM1OUT output time period=16 us *2=32 us.

TM1OUT output frequency=1/32 us=31.250 KHz.

| 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0Bh.5 **TM1IE:** Timer1 interrupt enable
0: disable
1: enable

| 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0Ch.5 **TM1IF:** Timer1 interrupt event pending flag
This bit is set by H/W while Timer1 overflows, write 0 to this bit will clear this flag

| 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

12h **TM1:** Timer1 content

| 13h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TM1RLD | TM1RLD | | | | | | | |
| R/W | W | W | W | W | W | W | W | W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

13h.7~0 **TM1RLD:** Timer1 reload offset value while it rolls over

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

14h.3~0 **TM1PSC:** Timer1 prescaler. Timer1 clock source divided by

| | | | |
|---|---|---|---|
| 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 |
| Reset | – | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

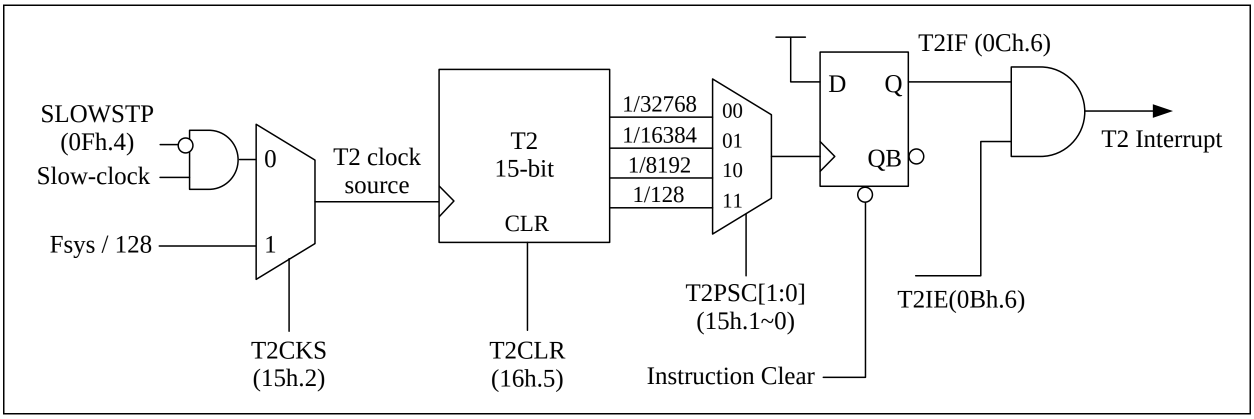
16h.4 **TM1STP:** Timer1 counter stop
0: Release
1: Stop counting

| 98h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| MF098 | TCOE | TM1OE | PWM0OE | PWM2OE | PWM1AOE | PWM1BOE | PWM1COE | - |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | - |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - |

F1B.5 **TM1OE:** Enable Timer1 overflow toggle output to PB0 pin (TM1OUT)

## 6.4 T2:15-bit Timer

The T2 is a 15-bit counter and the clock sources are from either Fsys/128 or Slow-clock. It is used to generate time base interrupt and T2 counter block clock. The T2 content cannot be read by instructions. It generates interrupt flag T2IF (0Ch.6) with the clock divided by 32768/16384/8192/128 depends on T2PSC[1:0] (15h.1~0) register bits. The following figure shows the block diagram of T2.



**T2 Block Diagram**

Example:

[CPU running at FAST mode, Fsys=Fast-clock= FIRC 4 MHz]

◇ Example:

```
; Setup T2 clock source and divider .
            MOVLW    00000101B     ;15h.2 (T2CKS) = 1, T2 clock source = Fsys/128
            MOVWX    T2CTL         ;15h.1~0 (T2PSC) =1, Divided by 16384
      ✧

            BSX      T2CLR         ;16h.5 (T2CLR)=1, Stop T2 counting.

; Enable T2 timer and interrupt function.
            MOVLW    10111111B     ; Clear T2 request interrupt flag by byte operation
            MOVWX    INTIF         ;

            MOVLW    01000000B     ; Enable T2 interrupt function.
            MOVWX    INTIE         ;

            BCX      T2CLR         ; Enable T2 counting (Default "0").
```

T2 clock source is Fsys/128 = 4 MHz/128 = 31250 Hz, T2PSC = /16384

T2 frequency = 31250 Hz / 16384 = 1.907 Hz

◇Example:

[CPU running at SLOW mode, Fsys = Slow-clock = SIRC 80Hz]

◇ Example:

```
; Setup T2 clock source and divider
            MOVLW   00000000B     ; 15h.2 (T2CKS) = 0, T2 clock source = Slow-clock
            MOVWX   T2CTL         ; 15.1~0 (T2PSC) =0, Divided by 32768

            BSX     T2CLR         ; Stop T2 counting.

; Enable T2 timer and interrupt function.
            MOVLW   10111111B     ; Clear T2 request interrupt flag
            MOVWX   INTIF

            MOVLW   01000000B     ; Enable T2 interrupt function.
            MOVWX   INTIE

            BCX     T2CLR         ; Enable T2 counting (Default “0”).
```

T2 clock source is Slow-clock = 80KHz, T2PSC = /32768,

T2 frequency = 80000Hz / 32768 ≒ 2.44Hz

| 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0Bh.6 **T2IE:** T2 interrupt enable
0: disable
1: enable

| 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 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

0Ch.6 **T2IF:** T2 interrupt event pending flag
This bit is set by H/W while T2 overflows, write 0 to this bit will clear this flag

| 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 |
| Reset | – | – | – | 0 | 1 | 0 | 1 | 1 |

0Fh.4 **SLOWSTP:** Stop Slow-clock in Stop Mode
0: no Stop 1: Stop

| 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 |
| Reset | – | – | – | – | – | 0 | 0 | 0 |

15h.2 **T2CKS:** "T2 clock source" selection.
0: Slow-clock ; 1: Fsys/128

15h.1~0 **T2PSC:** T2 prescaler. "T2 clock source" divided by -
00: 32768 01: 16384 10: 8192 11: 128

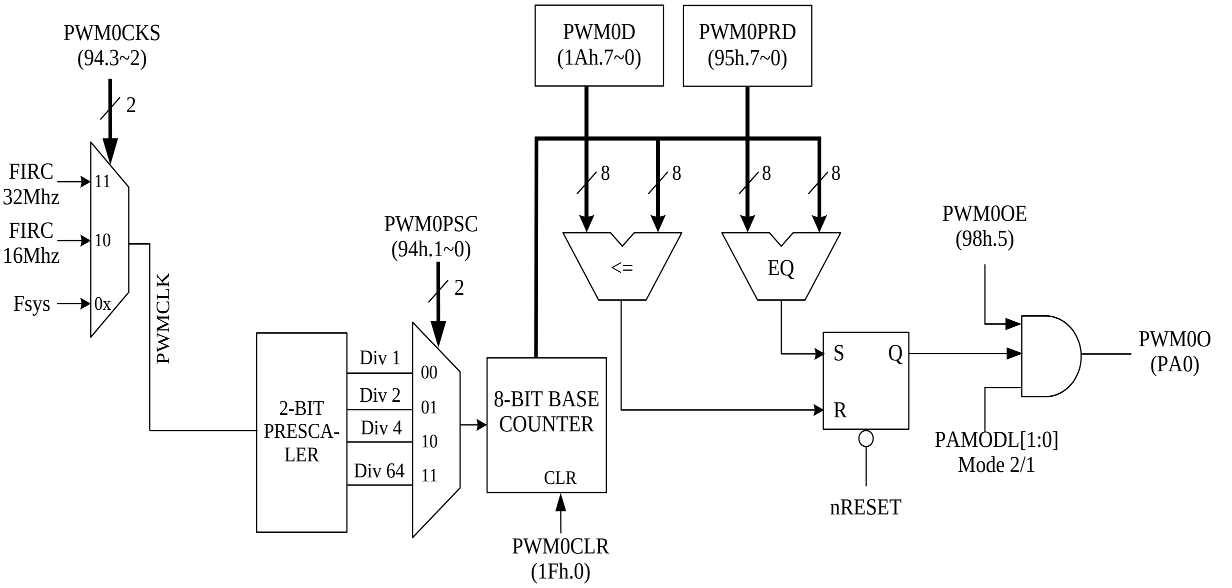
| 16h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| MF016 | LVDF | LVDEN | T2CLR | TM1STP | TM0STP | LVRSAV | LVDS | |
| R/W | R | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | – | 0 | 1 | 0 | 0 | 1 | 0 | 1 |

16h.5 **T2CLR:** T2 counter clear
0: Release 1: Stop counting

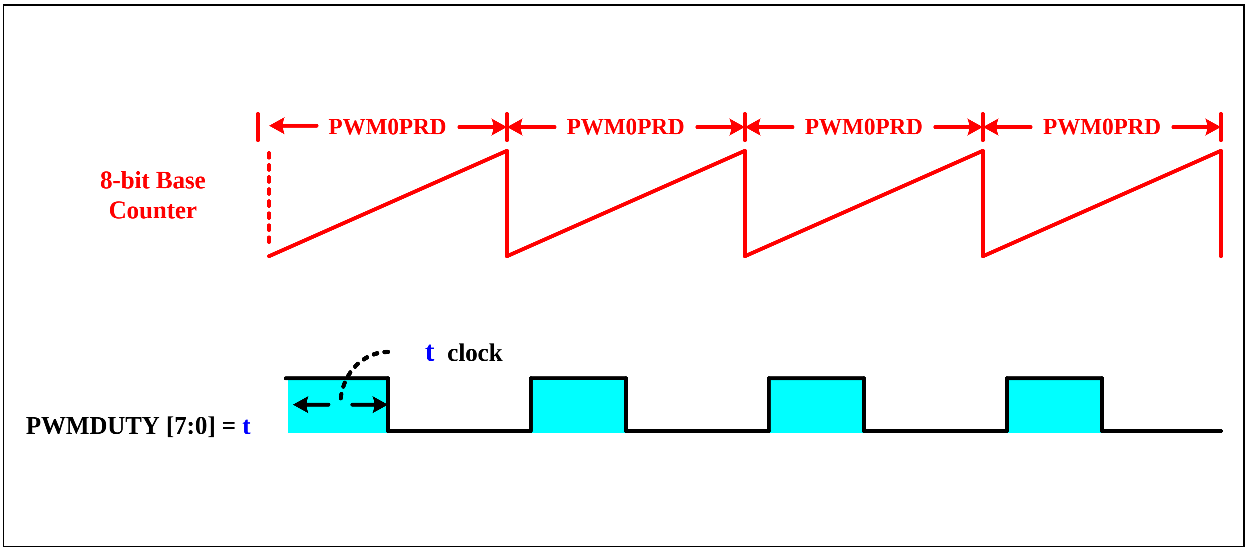
## 6.5 PWM0: 8 bits PWM

The PWM0 can generate various frequency waveforms with 1024 duty resolution based on PWM0CLK, which can select Fsys or FIRC 16MHz OR FIRC 32MHz, decided by PWM0CKS (94.3~2). The advantage of higher PWM frequency is that the post RC filter can transform the PWM signal to more stable DC voltage level. The PWM output signal reset to low level whenever the 8-bit base counter matches the PWM0 duty register PWM0D (1Ah.7~0).

The PWM0 period can be set by writing period value to PWM0PRD register (95h). Note that changing the PWM0PRD will immediately change the PWM0PRD values. Changing PWM0D (1Ah.7~0) also immediately change the PWM0D values. The Programmer must pay attention to the current time to change PWM0PRD by observing the following figure. There is a digital comparator that compares the PWM0 counter and PWM0RD, if PWM0 counter is larger than PWM0PRD after setting the PWM0PRD, a fault long PWM cycle will be generated because PWM0 counter must count to overflow then keep counting to PWM0PRD to finish the cycle.



**PWM0 Block Diagram**



**PWM0 Timing Diagram**

◇ Example: [CPU running at Fast mode, Fsys=FIRC 16Mhz]

; Setup PWM0 clock prescaler

```
        MOVLW       00001011B     ; PWM0 clock source = FIEC 16MHz
        MOVWX       PWMCTL1       ; PWM0 prescaler /64
        MOVLW       80h
        MOVWX       PWM0PRD       ; set PWM0 period =80h
        MOVLW       20h           ;
        MOVWX       PWM0D         ; set PWM0 duty    =20h
        BCX         PWM0CLR       ; release PWM0 clear flag
        MOVLW       xxxxxx10B     ;
        MOVWX       PAMODL        ; set PA0 as CMOS output
        BSX         PWM0OE        ; enable PWM0 output to PA0
```

| 95h | 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 |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

95h.7~0 **PWM0PRD:** PWM0 period data

| 1Ah | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PWM0D | PWM0D | | | | | | | |
| 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 |

1Ah.7~0 **PWM0D:** PWM0 duty 8-bit

| 1Fh | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PWMCTL0 | PWM2CKS | | PWM2PSC | | - | PWM2CLR | PWM1CLR | PWM0CLR |
| R/W | R/W | | R/W | R/W | - | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | - | 0 | 0 | 0 |

1Fh.0 **PWM0CLR:** PWM0 clear
0:PWM0 enable 1:PWM0 clear and keep low

| 94h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PWMCTL1 | PWM1CKS | | PWM1PSC | | PWM0CKS | | PWM0PSC | |
| R/W | R/W | | R/W | | R/W | | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

94h.3~2 **PWM0CKS:** PWM0 Clock Source
0x: Fsys
10:FIRC16M
11:FIRC32M

94h.1~0 **PWM0PSC:** PWM0 Clock Source Prescaler
00: DIV1 01: DIV2 10:DIV4 11:DIV64

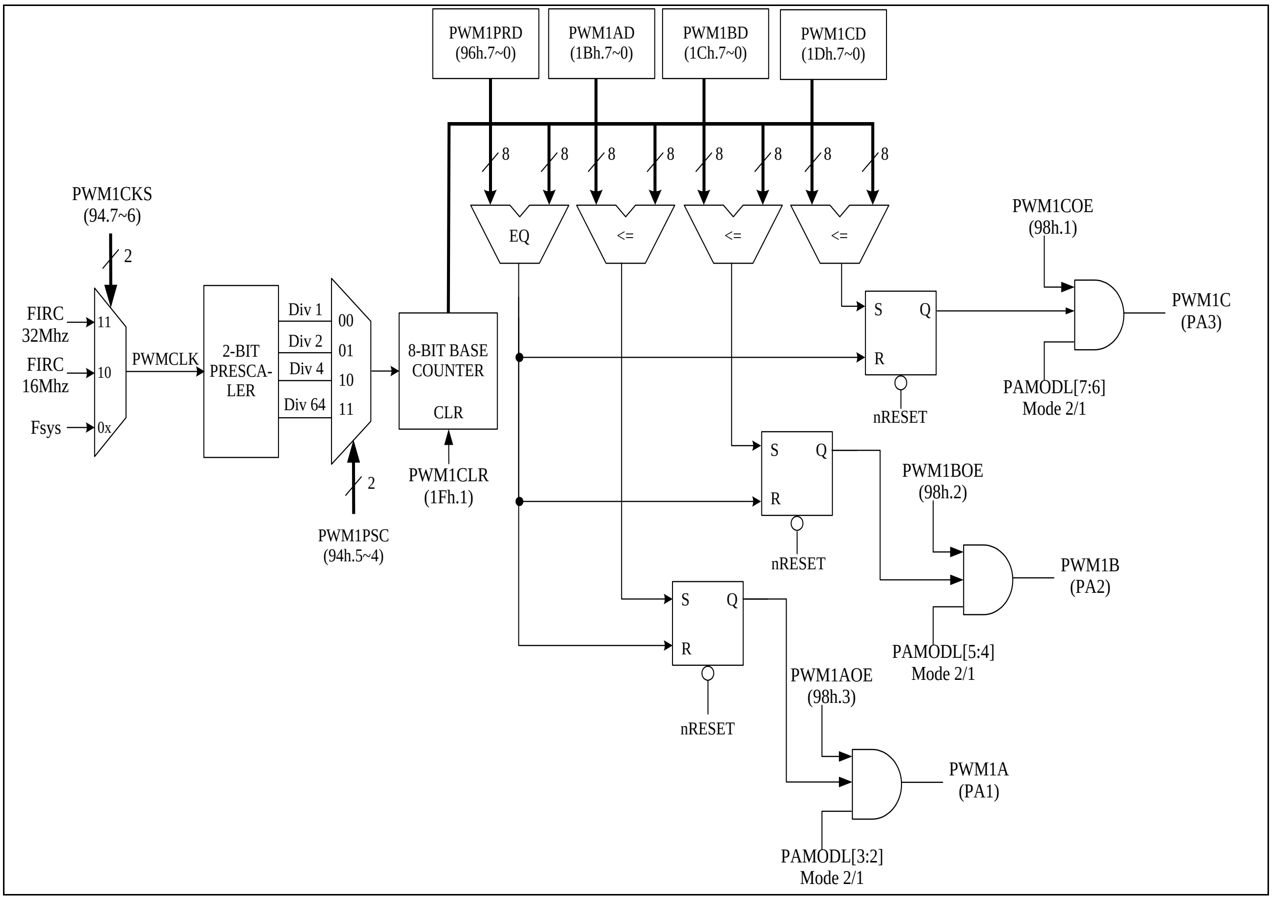
| 98h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| MF098 | TCOE | TM1OE | PWM0OE | PWM2OE | PWM1AOE | PWM1BOE | PWM1COE | - |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | - |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - |

98h.5 **PWM0OE:** PWM0 Output Enable
0: Disable 1:Enable, PWM0 output to PA0

## 6.6 PWM1A/PWM1B/PWM1C: 8 bits PWMs

PWM1A/PWM1B/PWM1C are 3 PWMs which have independent duty and common period. The PWMs can generate various frequency waveform with 256 duty resolution based on PWMCLK, which can select Fsys, FIRC 16 MHz or 32Mhz, decided by PWMCKS (96h.5~4). The advantage of higher PWM frequency is that the post RC filter can transform the PWM signal to more stable DC voltage level. The PWM output signal reset to low level whenever the 8-bit base counter matches the 8-bit of PWM duty register PWM1AD/PWM1BD/PWM1CD.

The PWM1A/1B/1C common period can be set by writing period value to PWM1PRD register (96h). Note that changing the PWM1PRD will immediately change the PWM1PRD values. The Programmer must pay attention to the current time to change PWM1PRD by observing the following figure. There is a digital comparator that compares the PWM1A/1B/1C counter and PWM1RD, if PWM1A/1B/C counter is larger than PWM1PRD after setting the PWM1PRD, a fault long PWM cycle will be generated because PWM1A/1B/1C counter must count to overflow then keep counting to PWM1PRD to finish the cycle.



**PWM1 Block Diagram**

| 96h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PWM1PRD | PWM1PRD | | | | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

96h.7~0 **PWM1PRD:** PWM1 period data

| 1Bh | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PWM1AD | PWM1AD | | | | | | | |
| 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 |

1Bh.7~0 **PWM1AD:** PWM1A duty 8-bit

| 1Ch | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PWM1BD | PWM1BD | | | | | | | |
| 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 |

1Ch.7~0 **PWM1BD:** PWM1B duty 8-bit

| 1Dh | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PWM1CD | PWM1CD | | | | | | | |
| 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 |

1Dh.7~0 **PWM1CD:** PWM1C duty 8-bit

| 1Fh | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PWMCTL0 | PWM2CKS | | PWM2PSC | | - | PWM2CLR | PWM1CLR | PWM0CLR |
| R/W | R/W | | R/W | R/W | - | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | - | 0 | 0 | 0 |

1Fh.1 **PWM1CLR:** PWM1 clear
0:PWM1 enable 1:PWM1 clear and keep low

| 94h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PWMCTL1 | PWM1CKS | | PWM1PSC | | PWM0CKS | | PWM0PSC | |
| R/W | R/W | | R/W | | R/W | | R/W | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

94h.7~6 **PWM1CKS:** PWM1 Clock Source
0x: Fsys
10:FIRC16M
11:FIRC32M

94h.5~4 **PWM1PSC:** PWM1 Clock Source Prescaler
00: DIV1 01: DIV2 10:DIV4 11:DIV64

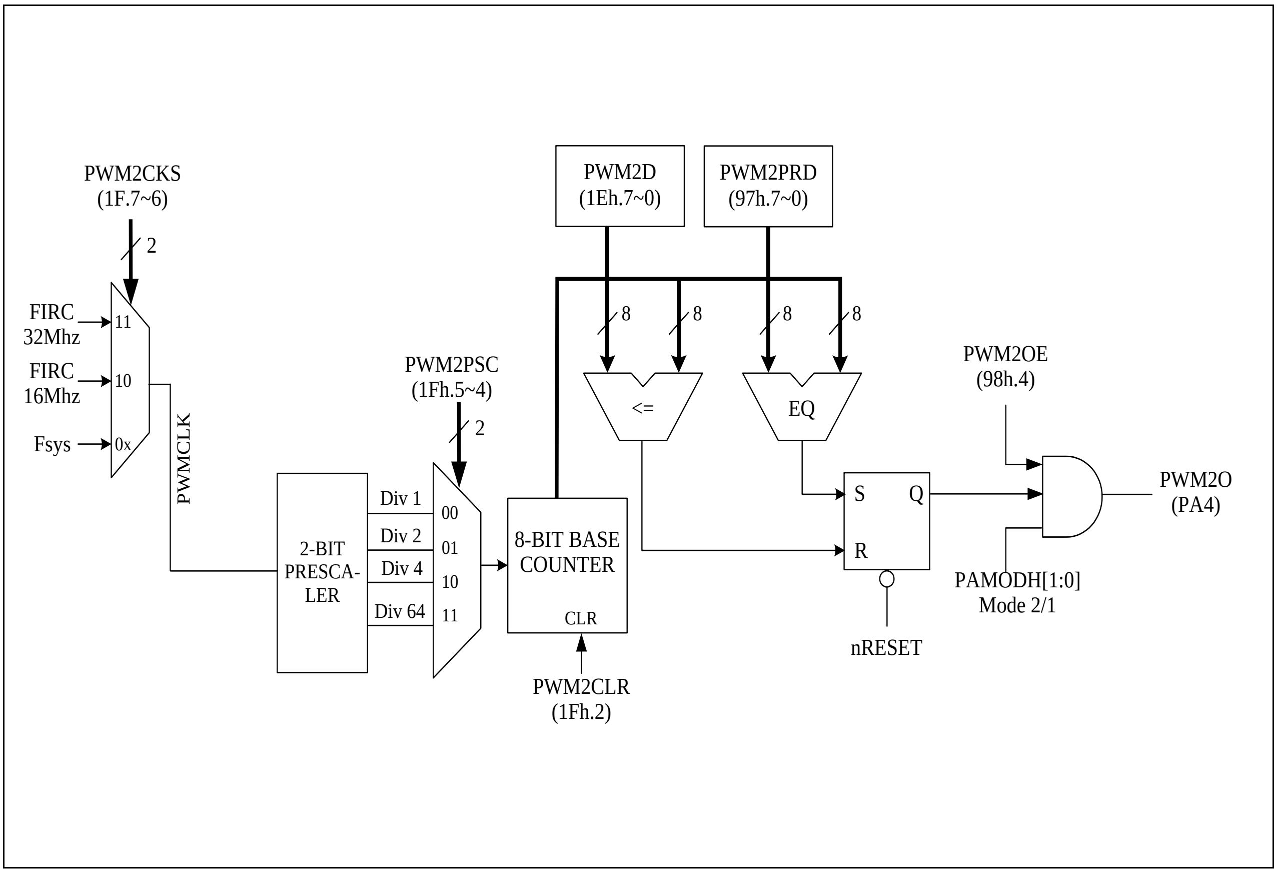
| 98h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| MF098 | TCOE | TM1OE | PWM0OE | PWM2OE | PWM1AOE | PWM1BOE | PWM1COE | - |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | - |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - |

98h.3 **PWM1AOE:** PWM1A Output Enable
0: Disable 1:Enable, PWM1A output to PA1
98h.2 **PWM1BOE:** PWM1B Output Enable
0: Disable 1:Enable, PWM1B output to PA2
98h.1 **PWM1COE: P**WM1C Output Enable
0: Disable 1:Enable, PWM1C output to PA3

## 6.7 PWM2: 8 bits PWM

The PWM2 has the same structure as PWM0. The PWM2 can generate various frequency waveforms with 1024 duty resolution based on PWM2CLK, which can select Fsys or FIRC 16MHz OR FIRC 32MHz, decided by PWM2CKS (1F.7~6). A spread LSB technique allows PWM2 to run its frequency at “PWM2CLK divided by 256” instead of “PWM2CLK divided by 1024”, which means the PWM2 is 4 times faster than normal. The advantage of higher PWM2 frequency is that the post RC filter can transform the PWM2 signal to more stable DC voltage level. The PWM output signal reset to low level whenever the 8-bit base counter matches the PWM0 duty register PWM2D (1Eh.7~0).

The PWM2 period can be set by writing period value to PWM2PRD register (97h). Note that changing the PWM2PRD will immediately change the PWM2PRD values. Changing PWM2D (1Eh.7~0) also immediately change the PWM2D values. The Programmer must pay attention to the current time to change PWM2PRD by observing the following figure. There is a digital comparator that compares the PWM0 counter and PWM2RD, if PWM2 counter is larger than PWM2PRD after setting the PWM2PRD, a fault long PWM cycle will be generated because PWM2 counter must count to overflow then keep counting to PWM2PRD to finish the cycle.



**PWM2 Block Diagram**

| 97h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PWM2PRD | PWM2PRD | | | | | | | |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

97h.7~0 **PWM2PRD:** PWM2 period data

| 1Eh | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PWM2D | PWM2D | | | | | | | |
| 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 |

1Eh.7~0 **PWM2D:** PWM2 duty 8-bit

| 1Fh | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| PWMCTL0 | PWM2CKS | | PWM2PSC | | - | PWM2CLR | PWM1CLR | PWM0CLR |
| R/W | R/W | | R/W | R/W | - | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | - | 0 | 0 | 0 |

1Fh.7~6 **PWM2CKS:** PWM2 Clock Source
0x: Fsys
10:FIRC16M
11:FIRC32M

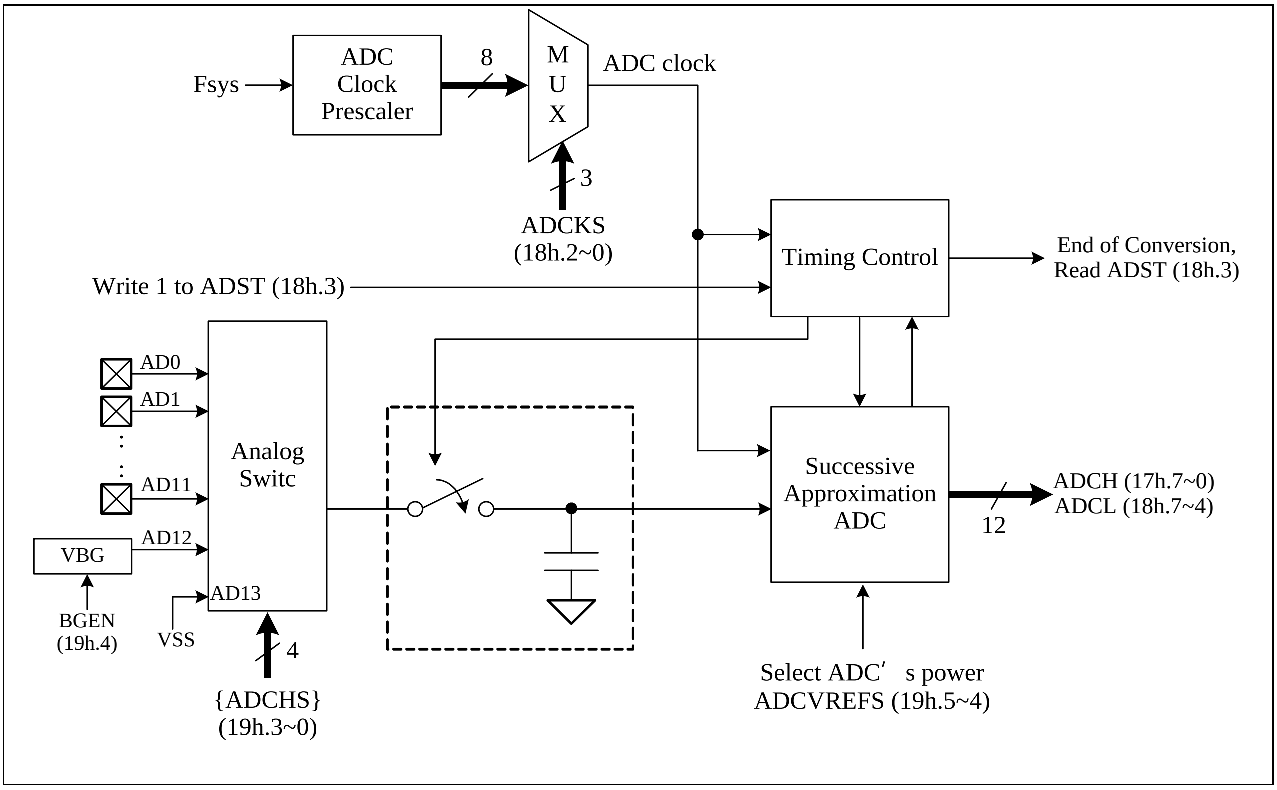
1Fh.5~4 **PWM2PSC:** PWM2 Clock Source Prescaler
00: DIV1 01: DIV2 10:DIV4 11:DIV64

1Fh.2 **PWM2CLR:** PWM2 clear
0:PWM2 enable 1:PWM2 clear and keep low

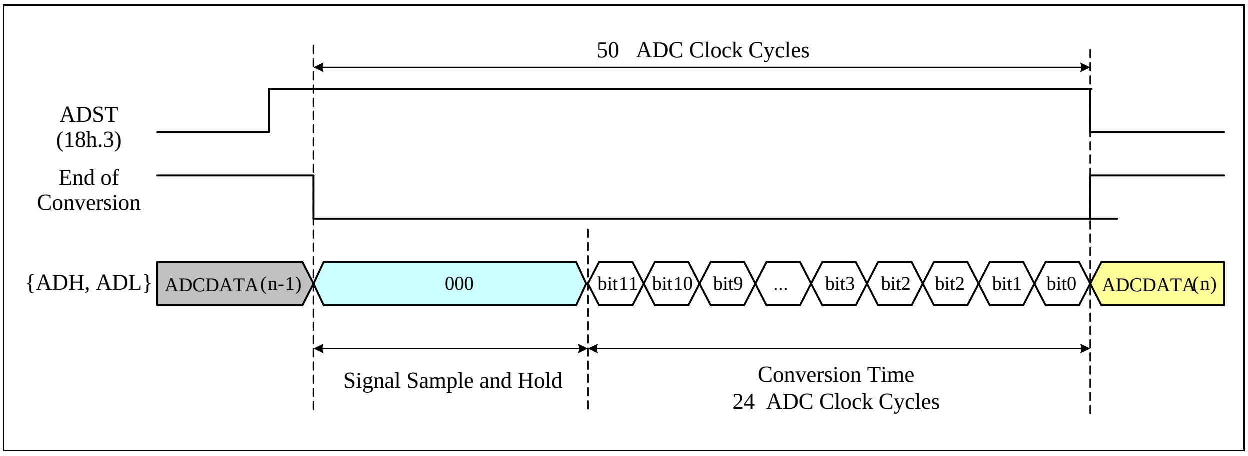
| 98h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| MF098 | TCOE | TM1OE | PWM0OE | PWM2OE | PWM1AOE | PWM1BOE | PWM1COE | - |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | - |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | - |

98h.4 **PWM2OE:** PWM2 Output Enable
0: Disable 1:Enable, PWM2 output to PA4

## 6.8 Analog-to-Digital Converter



The 12-bit ADC (Analog to Digital Converter) consists of a 12-channel analog input multiplexer, control register, clock generator, 12-bit successive approximation register, and output data register. To use the ADC, user needs to set ADCKS (18h.2~0) to choose a proper ADC clock frequency, which must be less than 1 MHz. User then launches the ADC conversion by setting the ADST (18h.3) control bit. After end of conversion, H/W automatic clears the ADST (18h.3) bit. User can poll this bit to know the conversion status. The PxMODE control registers are used for ADC pin configuration, user must set the Pin Mode=3 when the pin is used as an ADC input. The setting can disable the pin logical input path to save power consumption. User needs to set {ADCHS} (19h.3~0) to choose the input channel of ADC. One of them, AD12 is VBG1.25V input for ADC. Besides, VBG is controlled by BGEN (19h.6). ADC reference voltage is selected by ADCVREFS (19h.5~4).



Example:

[CPU running at FAST mode , Fsys=FIRC 16 MHz ]

ADC clock frequency=1 MHz, ADC channel=ADC10 (PA2).

◇ Example:

```
         MOVLW    00000111B     ; Fsys=16 MHz
         MOVWX    CLKCTL        ;

         MOVLW    01110101B     ; ADC2 (PA2) Pin Mode=3=ADC input
         MOVWX    PAMODL;

         MOVLW    00000011B     ; Fsys=16 MHz
         MOVWX    ADCTL         ; 18h.2~0 (ADCKS) =ADC clock=Fsys/32=500KHz

         MOVLW    01001010B     ; 19h.6=1, enable VBG1.25V; 19h.5~4=0, VREF=VCC
         MOVWX    MF019         ; 19h.3~0 (ADCHS [3:0]) =10, ADC select ADC10 (PA2 pin).

         BSX      ADST          ; 18h.3 (ADST) , ADC start conversion.

  WAIT_ADC:
         BTXSC    ADST          ; Wait ADC conversion finish.
         GOTO     WAIT_ADC

         MOVXW    ADH           ; 17h.7~0, Read ADC result [11:4] into W
         MOVXW    ADCTL         ; 18h.7~4, Read ADC result [3:0] into W

         :
         :
```

| 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 |
| Reset | – | – | – | – | – | – | – | – |

17h.7~0 **ADH:** ADC output data 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 |
| Reset | – | – | – | – | 0 | 0 | 0 | 0 |

18h.7~4 **ADCL:** ADC output data LSB, ADQ [3:0]

18h.3 **ADST:** ADC start bit.
0: H/W clear after end of conversion
1: ADC start conversion

18h.2~0 **ADCKS:** ADC clock frequency selection:

| | |
|---|---|
| 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 | - | BGEN | ADCVREFS | | ADCHS | | | |
| R/W | - | R/W | R/W | | R/W | | | |
| Reset | - | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

F19.6 **BGEN:** Band Gap BG1.25V enable
0: Disable
1: Enable and Auto disable in STOP/IDLE mode

F19.5~4 **ADCVREFS:** ADC VREF select,
00: VCC, 01: 2.5V, 10: 3V, 11: 4V

F19.3~0 **ADCHS:** ADC channel select
0000: ADC0 (PD0) 0001: ADC1 (PD1) 0010: ADC2 (PD2) 0011: ADC3 (PD3)
0100: ADC4 (PD4) 0101: ADC5 (PD5) 0110: ADC6 (PD6) 0111: ADC7 (PD7)
1000: ADC8(PA0) 1001: ADC9(PA1) 1010: ADC10(PA2) 1011: ADC11(PA3)
1100: VBG 1.25V 1101: ADC13(VSS) 1110: ADC14(VSS)

## 6.9 Touch Key

The Touch Key offers an easy, simple and reliable method to implement finger touch detection. In most applications, it doesn’t require any external component. The device support 3 modules, 16 channels touch key detection.

To use the Touch Key, user must setup the Pin Mode (***see Section 5***) correctly as below table. Setting Mode2 for an Idling Touch Key pin can CMOS output Low and reduce the mutual interference between the adjacent keys.

| PxMODx setting for Touch Key | TK0~TK15 |
|---|---|
| Pin is Touch Key, Idling | CMOS output Low<br>(Mode2) |
| Pin is Touch Key, Scanning | |

There are three Touch Key Modules (Module0, Module1 and Module2) in the TM56F5412/16/12B/16B. Each module can work independently. In the Touch Key Module, there are two oscillators: Reference Clock (RCKx) and Touch Clock (TCKx). They are connected to the Reference Counter and Data Counter respectively. The frequency of RCKx can be adjusted by setting TKMxREFC. Reference Counter is used to control conversion time. TKMxTCP is touch key clock frequency select (only available in TKMxJMP=0). When TKMxJMP =1, the touch key clock frequency is automatically change.

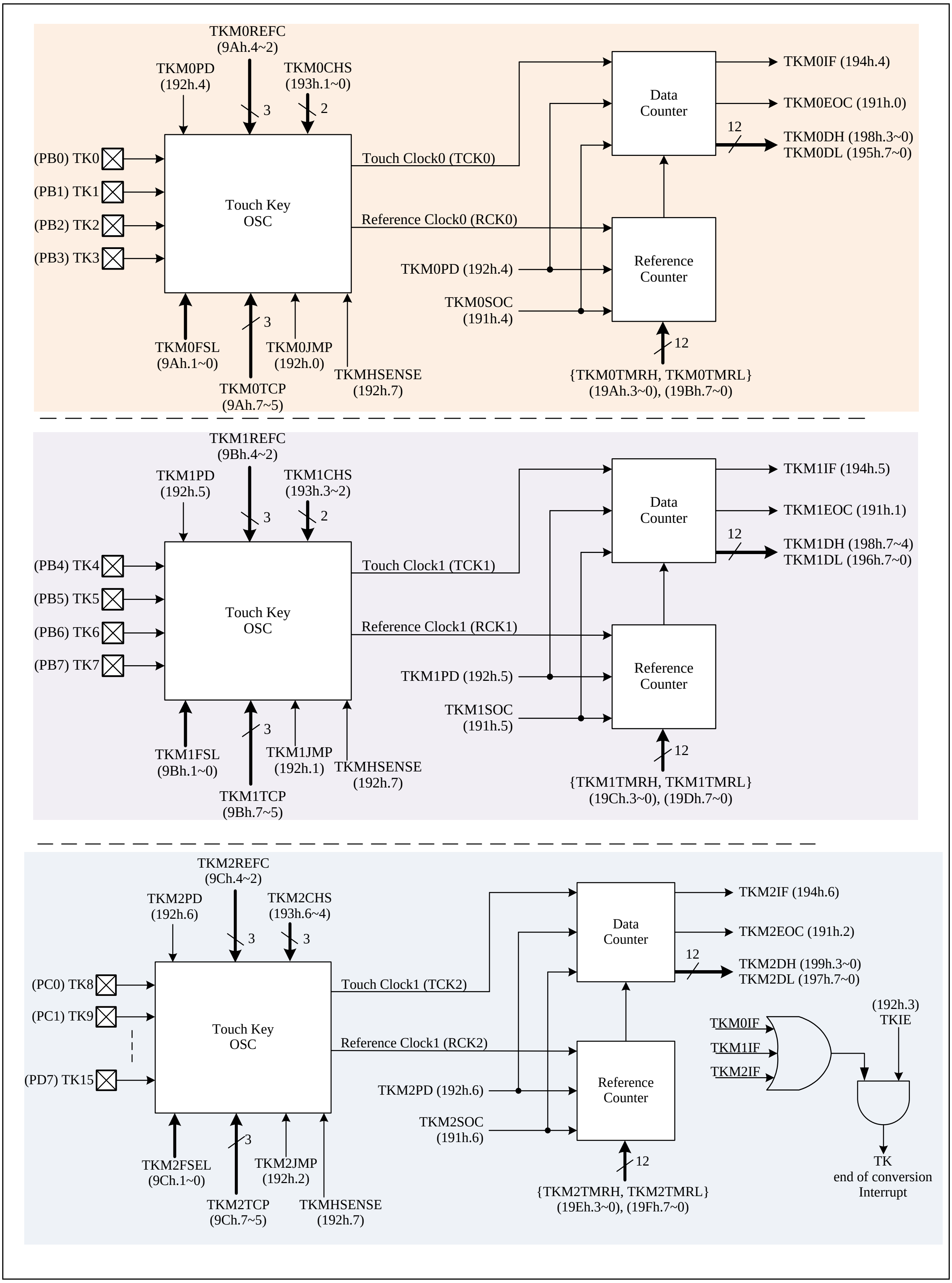
From starting touch key conversion to end, it will take 0 to 4096 RCK oscillation cycles by setting TKMxTMR. After end of conversion, user can get TK Data (TKMxDH, TKMxDL) from Data counter. TK Data is affected by finger touching. As finger touching TCK is getting slower, the value of TK Data is smaller than the no finger touching. According to the difference of TKMxDATA, user can check if it is touched of not. A suitable TKMxTMR and TKMxREFC setting can adjust TK Data to adapt the system board circumstances. To get the best TKMxREFC setting, user can try different TKMxREFC value, and then find the one which makes the TK Data and TKMxTMR as close as possible. In the other hand, user can adjust the overall operating frequency of the TK system (including TCKx & RCKx) by setting TKMxFSL (frequency select). For all tTouch Key Module, there is a control signal TKMHSENSE (192h.7) can be set to enhance the Touch key performance.

To start the Scanning, user assigns TKMxPD=0, then set the TKMxSOC bit to start touch key conversion, the TKMxSOC bit can be automatically cleared while end of conversion. However, if the SYSCLK is too slow, H/W might fail to clear TKMxSOC due to clock sampling rate. TKMxEOC=0 means conversion is in process. TKMxEOC=1 means the conversion is finish, and the touch key counting result is stored into the 12 bits TK Data Counter TKMxDH and TKMxDL.

TKIF (sum of TKM0IF, TKM1IF and TKM2IF) will active at the first time enable Touch Key function (TKMxPD=0), user should clear TKIF after TKMxPD cleared.

| TKM0IF | TKM1IF | TKM2IF | TKIF | STATE |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | IDLE |
| 1 | 0 | 0 | 1 | TK Module0 is end of conversion |
| 0 | 1 | 0 | 1 | TK Module1 is end of conversion |
| 1 | 1 | 0 | 1 | TK Module0 and Module1 are both end of conversion |
| 0 | 0 | 1 | 1 | TK Module2 is end of conversion |
| 1 | 0 | 1 | 1 | TK Module0 and Module2 are both end of conversion |
| 0 | 1 | 1 | 1 | TK Module1 and Module2 are both end of conversion |
| 1 | 1 | 1 | 1 | All TK module are end of conversion |

**Touch Key Interrupt Flag Description**



**Touch Key Structure**

The touch key modules of TM56F5412/TM56F5416 are different to the touch key modules of TM56F5412B/TM56F5416B. The difference is list below.

**The Touch Key module difference between TM56F5412/5416 and TM56F5412B/5416B**

| | 5412/5416 | 5412B/5416B |
|---|---|---|
| TKMxREFC | – | The capacitance is 2pF lower than 5412/5416 |
| TKMxTCP | – | When TCP=111,<br>RC oscillation become slow,<br>Voltage stabilization is better |
| Suggestion 1 | – | When FSEL=3, TCP=7, HSEN=1<br>Voltage stabilization effect is best |
| Suggestion 2 | – | For PCB application,<br>short wire path: set REFC=0,<br>long wire path: set REFC=1 or 2 |

x=**0,1,2**

◇ Example: Use TK module0, Module1 and Module2, Touch key channel = TK2 (PB2), TK4 (PB4), TK12 (PD4)

```
.ORG        004h
INT:
            BTXSC       TKIF        ; check TKIF
            CALL        INT_TK
            RETI
INT_TK:
            BTXSC       TKM0IF      ; check TKM0IF
            CALL        INT_TKM0
            BTXSC       TKM1IF      ; check TKM1IF
            CALL        INT_TKM1
            BTXSC       TKM2IF      ; check TKM2IF
            CALL        INT_TKM2
            RET
INT_TKM0:
            MOVLW       11101111B   ; clear TKM0IF
            MOVWX       TKMFLG
            MOVXW       TKM10DH     ; read TK0 DATA[11:8] into W register
            MOVXW       TKM0DL      ; read TK0 DATA[7:0] into W register
            RET
INT_TKM1:
            MOVLW       11011111B   ; clear TKM1IF
            MOVWX       TKMFLG
            MOVXW       TKM10DH     ; read TK1 DATA[11:8] into W register
            MOVXW       TKM1DL      ; read TK1 DATA[7:0] into W register
            RET
INT_TKM2:
            MOVLW       11101111B   ; clear TKM2IF
            MOVWX       TKMFLG
```

```
        MOVXW     TKM2DH      ; read TK2 DATA[11:8] into W register
        MOVXW     TKM2DL      ; read TK2 DATA[7:0] into W register
        RET


SET_MODE:
        MOVLW     xx10xxxxB   ; PBMODL[5:4] = 10b
        MOVWX     PBMODL      ; set PB2 as Mode 2 for touch key input
        BCX       PBD,2       ; clear PB2 as CMOS output low

        MOVLW     xxxxxx10B   ; PBMODH[1:0] = 10b
        MOVWX     PBMODH      ; set PB4 as Mode 2 for touch key input
        BCX       PBD,4       ; clear PB4 as CMOS output low

        MOVLW     xxxxxx10B   ; PDMODH[1:0] = 10b
        MOVWX     PDMODH      ; set PD4 as Mode 2 for touch key input
        BCX       PDD,4       ; clear PD4 as CMOS output low

TK_INIT:
        MOVLW     00100100B   ;
        MOVWX     TKM0CON     ; set TKM0TCP=1, TKM0REFC=1, TKM0FSL=0
        MOVLW     00h
        MOVWX     TKM0TMRH
        MOVLW     64H
        MOVWX     TKM0TMRL    ; TKM0TMR=64H=100

        MOVLW     01001000B   ;
        MOVWX     TKM1CON     ; set TKM0TCP=2, TKM0REFC=2, TKM0FSL=0
        MOVLW     00h
        MOVWX     TKM1TMRH
        MOVLW     C8H
        MOVWX     TKM1TMRL    ; TKM1TMR=C8H=200

        MOVLW     00000000B   ;
        MOVWX     TKM2CON     ; set TKM2TCP=0, TKM2REFC=0, TKM2FSL=0
        MOVLW     00h
        MOVWX     TKM2TMRH
        MOVLW     80H
        MOVWX     TKM2TMRL    ; TKM0TMR=80H=128

        MOVLW     x100 00 10B ; set TKM2CHS=4, TKM1CHS=0, TKM0CHS=2
        MOVWX     TKMCHS
        MOVLW     00000000B   ; enable TK Module0/1/2, TKMxJMP=0

        MOVLW     11110111B   ; clear TKIF
        MOVWX     TKMFLG
        ;;;BSX     TKIE
        MOVLW     10001100B
        MOVWX     TKMCTL1     ; TKMHSEN=1, TKM2 high sensitivity
                              ; TMK2PD=TKM1PD=TKM0PD=0,
                              ;    enable TMM2/TKM1/TKM0 activity
                              ; TKIE=1, enable TK interrupt
```

```
                                    ; TKM2JMP=1, TKM2 clock mode auto-change
                                    ; TKM1JMP= TKM0JMP=0, TKM2/TKM1
                                    ;   clock mode fixed(refer to TKM1TCP/ TKM0TCP)

TK_START:
          BSX         TKM0SOC       ; start TKM0 conversion
          BSX         TKM1SOC       ; start TKM1 conversion
          BSX         TKM2SOC       ; start TKM2 conversion
                      .
                      .
```

| 9Ah | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKM0CON | TKM0TCP | | | TKM0REFC | | | TKM0FSL | |
| 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 |

9Ah.7~5 **TKM0TCP:** TK module0 touch key clock frequency select; (only available in TKM0JMP=0)
TKM0TCP ={TKM0TCP[0], TKM0TCP[1], TKM0TCP[2] }, 000: slowest, …, 111: fastest

9Ah.4~2 **TKM0REFC:** TK module0 Reference clock capacitor select;
000:smallest, …, 111:biggest(conversion time longest)

9Ah.1~0 **TKM0FSL:** TK module0 clock(RCK0/TCK0) frequency selection;
00: slowest, …, 11: fastest

| 9Bh | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKM1CON | TKM1TCP | | | TKM1REFC | | | TKM1FSL | |
| 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 |

9Bh.7~5 **TKM1TCP:** TK module1 touch key clock frequency select; (only available in TKM1JMP=0)
TKM1TCP ={TKM0TCP[0], TKM1TCP[1], TKM1TCP[2] }, 000: slowest, …, 111: fastest

9Bh.4~2 **TKM1REFC:** TK module1 Reference clock capacitor select;
000:smallest, …, 111:biggest(conversion time longest)

9Bh.1~0 **TKM1FSL:** TK module1 clock(RCK1/TCK1) frequency selection;
00: slowest, …, 11: fastest

| 9Ch | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKM2CON | TKM2TCP | | | TKM2REFC | | | TKM2FSL | |
| 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 |

9Ch.7~5 **TKM2TCP:** TK module2 touch key clock frequency select; (only available in TKM2JMP=0)
TKM2TCP ={TKM2TCP[0], TKM2TCP[1], TKM2TCP[2] }, 000: slowest, …, 111: fastest

9Ch.4~2 **TKM2REFC:** TK module2 Reference clock capacitor select;
000:smallest, …, 111:biggest(conversion time longest)

9Ch.1~0 **TKM2FSL:** TK module2 clock(RCK2/TCK2) frequency selection;
00: slowest, …, 11: fastest

| 191h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKMCTL0 | - | TKM2SOC | TKM1SOC | TKM0SOC | - | TKM2EOC | TKM1EOC | TKM0EOC |
| R/W | - | R/W | R/W | R/W | - | R/W | R/W | R/W |
| Reset | - | 0 | 0 | 0 | - | - | - | - |

191h.6 **TKM2SOC:** Start Touch Key Module2 conversion
Set 1 to start Touch Key Module2 conversion. If SYSCLK is fast enough, this bit will be cleared by H/W at the end of conversion. S/W can also write 0 to clear this flag

191h.5 **TKM1SOC:** Start Touch Key Module1 conversion
Set 1 to start Touch Key Module1 conversion. If SYSCLK is fast enough, this bit will be cleared by H/W at the end of conversion. S/W can also write 0 to clear this flag

191h.4 **TKM0SOC:** Start Touch Key Module0 conversion
Set 1 to start Touch Key Module0 conversion. If SYSCLK is fast enough, this bit will be cleared by H/W at the end of conversion. S/W can also write 0 to clear this flag

191h.2 **TKM2EOC:** Touch Key Module2 end of conversion flag, TKM2EOC may have 3uS delay after TKM2SOC=1, so F/W must wait enough time before polling this Flag.
0: Indicates conversion is in progress
1: Indicates conversion is finished

191h.1 **TKM1EOC:** Touch Key Module1 end of conversion flag, TKM1EOC may have 3uS delay after TKM1SOC=1, so F/W must wait enough time before polling this Flag.
0: Indicates conversion is in progress
1: Indicates conversion is finished

191h.0 **TKM0EOC:** Touch Key Module0 end of conversion flag, TKM0EOC may have 3uS delay after TKM0SOC=1, so F/W must wait enough time before polling this Flag.
0: Indicates conversion is in progress
1: Indicates conversion is finished

| 192h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKMCTL1 | TKMHSEN | TKM2PD | TKM1PD | TKM0PD | TKIE | TKM2JMP | TKM1JMP | TKM0JMP |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

192h.7 **TKMHSEN:** TK Module Sensitivity, 1=higher sensitivity; 0=normal.

192h.6 **TKM2PD:** Touch Key Module2 power down
0: Touch Key Module2 running
1: Touch Key Module2 power down

192h.5 **TKM1PD:** Touch Key Module12 power down
0: Touch Key Module1 running
1: Touch Key Module1 power down

192h.4 **TKM0PD:** Touch Key Module0 power down
0: Touch Key Module0 running
1: Touch Key Module0 power down

192h.3 **TKIE:** Touch Key interrupt enable
0: Disable Touch Key interrupt
1: Enable Touch Key interrupt

192h.2 **TKM2JMP:** Touch Key Module2 clock mode
0: Fix frequency (refer to TKM2TCP**)**
1: Auto-change

192h.1 **TKM1JMP:** Touch Key Module1 clock mode
0: Fix frequency (refer to TKM1TCP)
1: Auto-change

192h.0 **TKM0JMP:** Touch Key Module0 clock mode
0: Fix frequency (refer to TKM0TCP)
1: Auto-change

| 193h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKMCHS | - | TKM2CHS | | | TKM1CHS | | TKM0CHS | |
| R/W | - | R/W | | | R/W | | R/W | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Reset | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

193h.6~4 **TKM2CHS**: TK module2 Channel Select
000: TK8 (PC0)
001: TK9 (PC1)
010: TK10 (PC2)
011: TK11 (PC3)
100: TK12 (PD4)
101: TK13 (PD5)
110: TK14 (PD6)
111: TK15 (PD7)

193h.3~2 **TKM1CHS**: TK module1 Channel Select
00: TK4 (PB4)
01: TK5 (PB5)
10: TK6 (PB6)
11: TK7 (PB7)

193h.1~0 **TKM0CHS**: TK module0 Channel Select
00: TK0 (PB0)
01: TK1 (PB1)
10: TK2 (PB2)
10: TK3 (PB3)

| 194h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKMFLG | - | TKM2IF | TKM1IF | TKM0IF | TKIF | - | - | - |
| R/W | - | R/W | R/W | R/W | R/W | - | - | - |
| Reset | - | 0 | 0 | 0 | 0 | - | - | - |

194h.6 **TKM2IF:** Touch Key Module2 interrupt event pending flag
This bit is set while Touch Key Module2 is end of conversion, write 0 to this bit will clear this flag or sets the TKM2SOC to clear this bit

194h.5 **TKM1IF:** Touch Key Module1 interrupt event pending flag
This bit is set while Touch Key Module1 is end of conversion, write 0 to this bit will clear this flag or sets the TKM1SOC to clear this bit

194h.4 **TKM0IF:** Touch Key Module0 interrupt event pending flag
This bit is set while Touch Key Module0 is end of conversion, write 0 to this bit will clear this flag or sets the TKM0SOC to clear this bit

194h.3 **TKIF:** Touch Key interrupt event pending flag
This bit is set by H/W while Touch Key Module0 or Module1 or Module2 are end of conversion, write 0 to this bit will clear **all** of the Touch Key flag

| 195h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKM0DL | TKM0DL | | | | | | | |
| R/W | R | R | R | R | R | R | R | R |
| Reset | - | - | - | - | - | - | - | - |

195h.7~0 **TKM0DL:** Touch Key Module0 data LSB[7:0]

| 196h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKM1DL | TKM1DL | | | | | | | |
| R/W | R | R | R | R | R | R | R | R |
| Reset | - | - | - | - | - | - | - | - |

196h.7~0 **TKM1DL:** Touch Key Module1 data LSB[7:0]

| 197h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKM2DL | TKM2DL | | | | | | | |
| R/W | R | R | R | R | R | R | R | R |
| Reset | - | - | - | - | - | - | - | - |

197h.7~0 **TKM2DL:** Touch Key Module2 data LSB[7:0]

| 198h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKM10DH | TKM1DH | | | | TKM0DH | | | |
| R/W | R | R | R | R | R | R | R | R |
| Reset | - | - | - | - | - | - | - | - |

198h.7~4 **TKM1DH:** Touch Key Module1 data MSB[11:8]

198h.3~0 **TKM0DH:** Touch Key Module0 data MSB[11:8]

| 199h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKM2DH | - | - | - | - | TKM2DH | | | |
| R/W | - | - | - | - | R | R | R | R |
| Reset | - | - | - | - | - | - | - | - |

198h.3~0 **TKM2DH:** Touch Key Module2 data MSB[11:8]

| 19Ah | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKM0TMRH | - | - | - | - | TKM0TMRH | | | |
| R/W | - | - | - | - | R/W | | | |
| Reset | - | - | - | - | 0 | 0 | 0 | 0 |

19Ah.3~0 **TKM0TMRH**: Touch Key Module0 reference counter MSB[11~8]

| 19Bh | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKM0TMRL | TKM0TMRL | | | | | | | |
| R/W | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

19Bh.7~0 **TKM0TMRL** Touch Key Module0 reference counter LSB[7~0]

| 19Ch | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKM1TMRH | - | - | - | - | TKM1TMRH | | | |
| R/W | - | - | - | - | R/W | | | |
| Reset | - | - | - | - | 0 | 0 | 0 | 0 |

19Ch.3~0 **TKM1TMRH**: Touch Key Module1 reference counter MSB[11~8]

| 19Dh | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKM1TMRL | TKM1TMRL | | | | | | | |
| R/W | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

19Dh.7~0 **TKM1TMRL** Touch Key Module1 reference counter LSB[7~0]

| 19Eh | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKM2TMRH | - | - | - | - | TKM2TMRH | | | |
| R/W | - | - | - | - | R/W | | | |
| Reset | - | - | - | - | 0 | 0 | 0 | 0 |

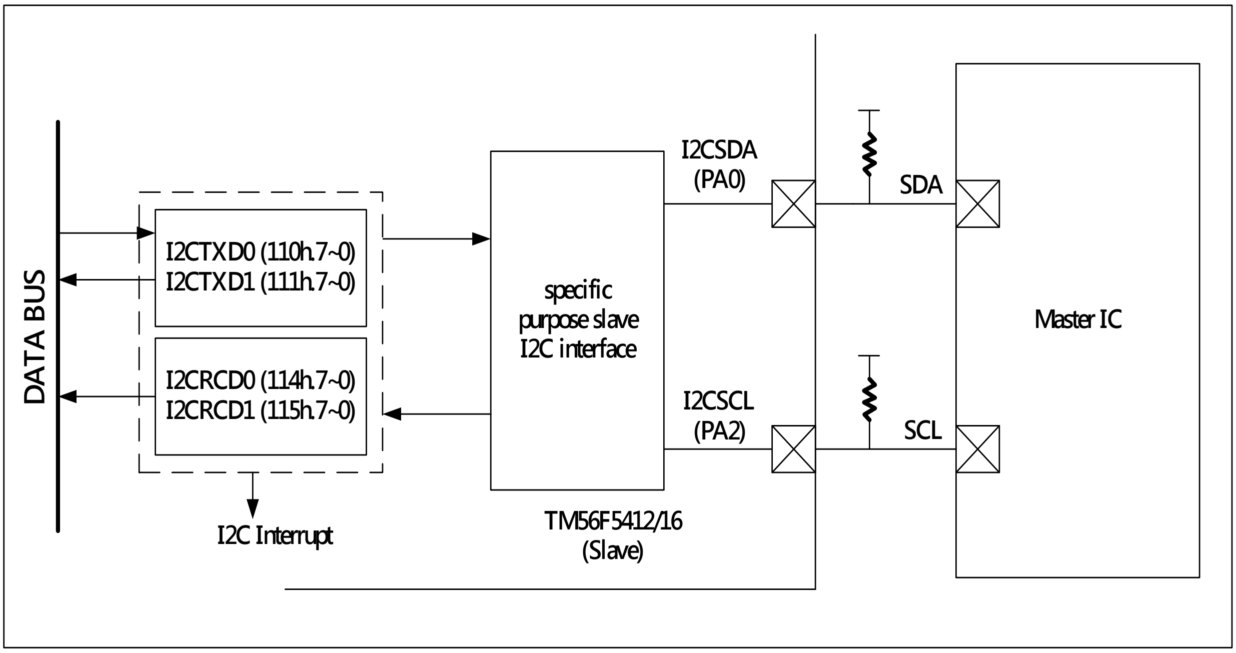
19Eh.3~0 **TKM2TMRH**: Touch Key Module2 reference counter MSB[11~8]

| 19Fh | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| TKM2TMRL | TKM2TMRL | | | | | | | |
| R/W | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

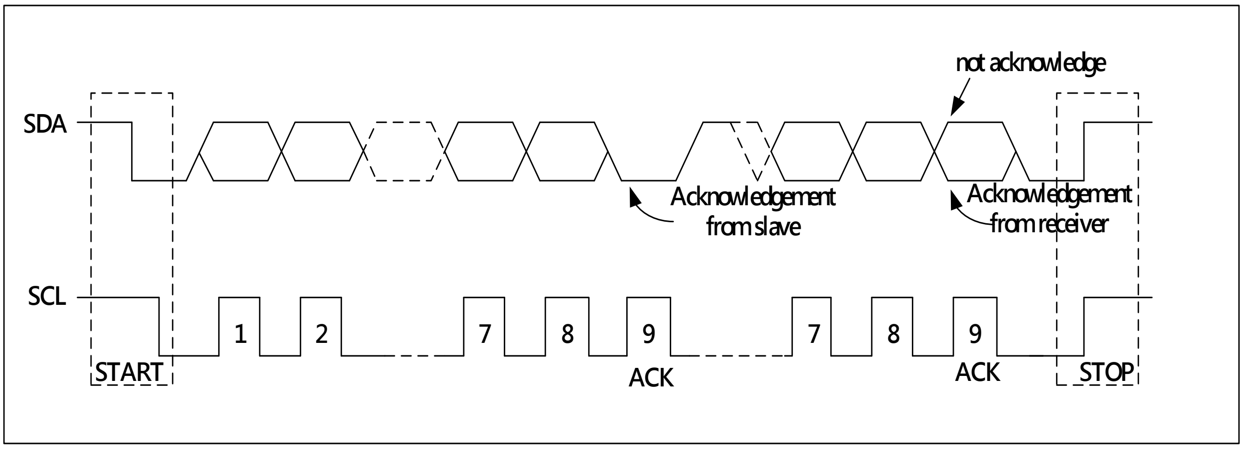
19Fh.7~0 **TKM2TMRL** Touch Key Module2 reference counter LSB[7~0]

## 6.10 Specific Purpose Slave I2C Interfac

Specific purpose slave I2C interface in TM56F5412/16/12B/16B could be used for data transmission. This interface is based on a standard I2C (Inter-Integrated Circuit), and TM56F5412/16/12B/16B is always as a slave mode. When the master node (another IC or device) sends the correct ID through I2C, it can read data from the register I2CTXD0 (110h.7~0) and I2CTXD1 (111h.7~0) of TM56F5412/16/12B/16B or write data to the register I2CRCD0 (114h.7~0) and I2CRCD1 (115h.7~0) of TM56F5412/16/12B/16B.

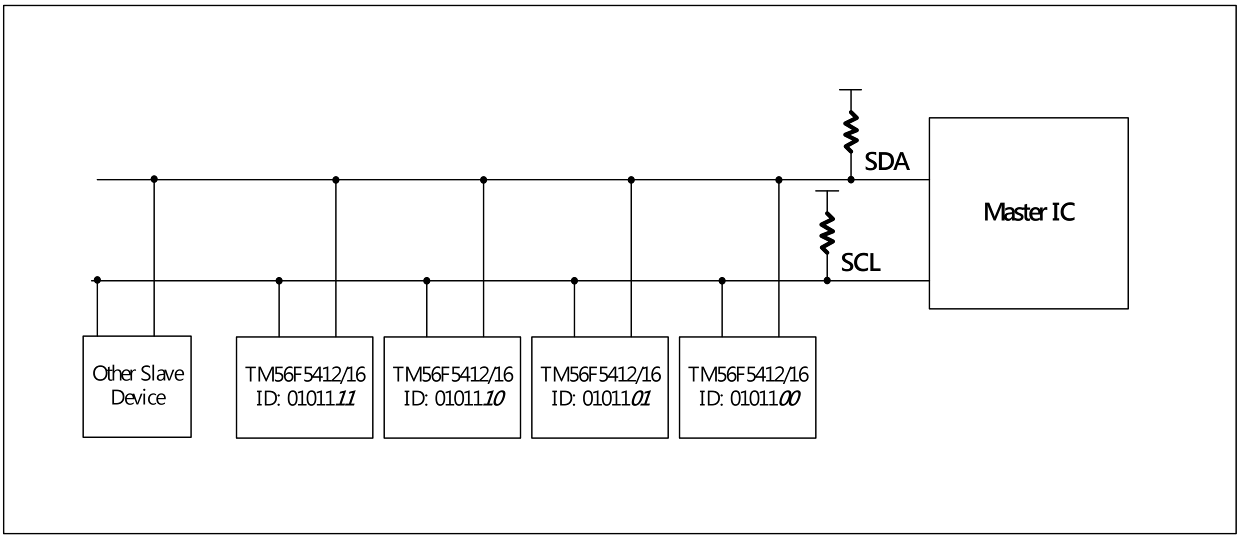


**Slave I2C Interface Block Diagram**

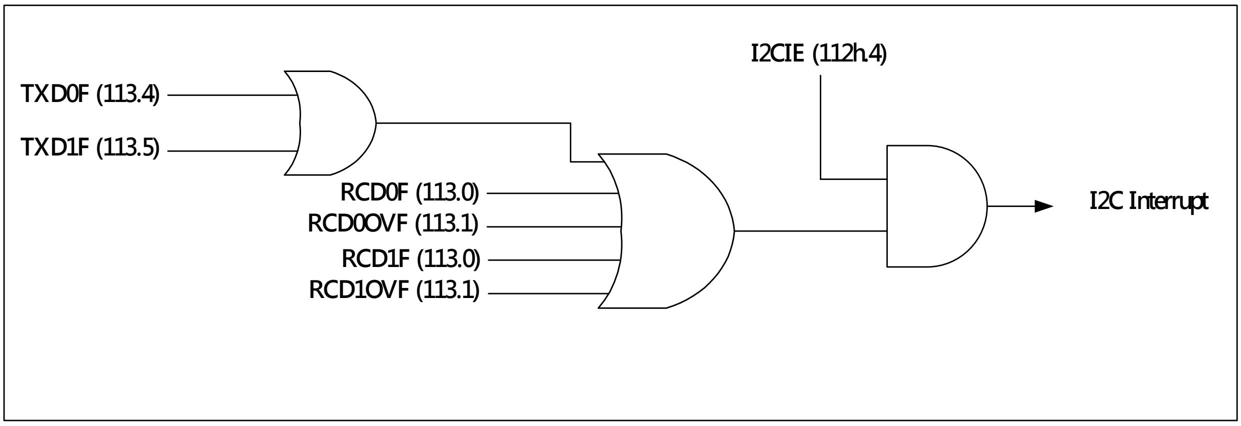


**I2C Protocol**

To use the slave I2C interface, the I2CEN (112h.3) bit has to be set. TM56F5412/16/12B/16B supports 4 slave device IDs by setting I2CID (112.1~0). TM56F5412/16/12B/16B can generate the transmitting flag TXD0F (113h.4) and TXD1F (113h.5) when data transmitting finished. It generates the receiving flag RCD0F (113h.0) and RCD1F (113h.2) when data receiving finished. It can also generates the receiving overflow flag RCD0OVF (113h.1) and RCD1OVF (113h.3) when data receiving finished but the receiving flag is not cleared. If one of those I2C flags is set, the I2C interrupt flag I2CIF (113h.6) will be generated. It generates I2C interrupt if the I2CIE (112h.4) bit is set. Refer to following table and figure.

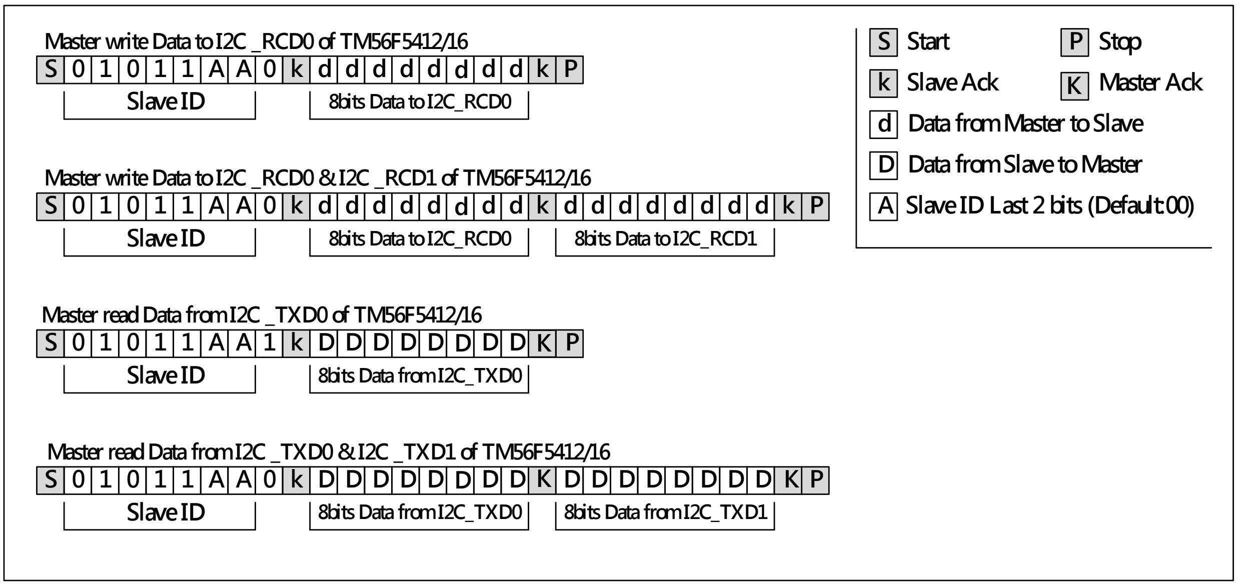

SDA
SCL
Master IC
Other Slave Device
TM56F5412/16 ID: 0101*11*
TM56F5412/16 ID: 0101*10*
TM56F5412/16 ID: 0101*01*
TM56F5412/16 ID: 0101*00*

I2C Parallel Connection Application Circuit



Slave I2C Interrupt Block Diagram

| RCDxOVF | RCDxF | I2CIF | STATE |
|---|---|---|---|
| 0 | 0 | 0 | IDLE |
| 0 | 1 | 1 | Date received to I2CRCDx register |
| 1 | 1 | 1 | Data overflow occurred at I2CRCDx register |


Master write Data to I2C _RCD0 of TM56F5412/16
S 0 1 0 1 1 A A 0 k d d d d d d d d k P
Slave ID
8bits Data to I2C_RCD0
Master write Data to I2C _RCD0 & I2C _RCD1 of TM56F5412/16
S 0 1 0 1 1 A A 0 k d d d d d d d d k d d d d d d d d k P
Slave ID
8bits Data to I2C_RCD0
8bits Data to I2C_RCD1
Master read Data from I2C _TXD0 of TM56F5412/16
S 0 1 0 1 1 A A 1 k D D D D D D D D K P
Slave ID
8bits Data from I2C_TXD0
Master read Data from I2C _TXD0 & I2C _TXD1 of TM56F5412/16
S 0 1 0 1 1 A A 0 k D D D D D D D D K D D D D D D D D K P
Slave ID
8bits Data from I2C_TXD0
8bits Data from I2C_TXD1
S Start
P Stop
k Slave Ack
K Master Ack
d Data from Master to Slave
D Data from Slave to Master
A Slave ID Last 2 bits (Default 00)

**TM56F5412/16/12B/16B I2C Commands**

| 110h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| I2CTXD0 | I2CTXD0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

110h.7~0 **I2CTXD0:** The transmitting register 0 of slave I2C

| 111h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| I2CTXD1 | I2CTXD1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

111h.7~0 **I2CTXD1:** The transmitting register 1 of slave I2C

| 112h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| I2CCTL | - | - | - | I2CIE | I2CEN | - | I2CID | |
| R/W | - | - | - | R/W | R/W | - | R/W | |
| Reset | - | - | - | 0 | 0 | - | 0 | 0 |

112h.4 **I2CIE:** Slave I2C interrupt enable
0: disable
1: enable

112h.3 **I2CEN:** Slave I2C interface enable
0: disable
1: enable

112h.1~0 **I2CIE:** Slave I2C ID last 2 bits

| 113h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| I2CFLG | - | I2CIF | TXD1F | TXD0F | RCD1OVF | RCD1F | RCD1OVF | RCD0F |
| R/W | - | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | - | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

113h.6 **I2CIF:** I2C interrupt event pending flag, This bit is set by H/W while
a: I2CRCD0 or I2CRCD1 receive data finished
b: I2CRCD0 or I2CRCD1 data overflow occured
c: I2CTXD0 or I2CTXD1 data transmit finished

113h.5 **TXD1F:** Slave I2C transmitting data register 1 flag
This bit is set by H/W while I2CTXD1 data transmitting finished, write 0 to this bit will clear this flag

113h.4 **TXD0F:** Slave I2C transmitting data register 0 flag
This bit is set by H/W while I2CTXD0 data transmitting finished, write 0 to this bit will clear this flag

113h.3 **RCD1OVF:** Slave I2C receiving data register 1 overflow
This bit is set by H/W while receiving data to I2CRCD1 overflow, write 0 to this bit will clear this flag

113h.2 **RCD1F:** Slave I2C receiving data register 1 flag
This bit is set by H/W while data receiving to I2CRCD1 finished, write 0 to this bit will clear this flag

113h.1 **RCD0OVF:** Slave I2C receiving data register 0 overflow
This bit is set by H/W while receiving data to I2CRCD0 overflow, write 0 to this bit will clear this flag

113h.0 **RCD0F:** Slave I2C receiving data register 0 flag
This bit is set by H/W while data receiving to I2CRCD0 finished, write 0 to this bit will clear this flag

| 114h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| I2CRCD0 | I2CRCD0 | | | | | | | |
| R/W | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

114h.7~0 **I2CRCD0:** The receiving register 0 of slave I2C

| 115h | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| I2CRCD1 | I2CRCD1 | | | | | | | |
| R/W | R/W | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

115h.7~0 **I2CRCD1:** The receiving register 1 of slave I2C

# MEMORY MAP

| Name | Address | R/W | Rst | Description |
|---|---|---|---|---|
| **INDF  (00h/80h/100h/180h)** | | | | **Function related to: RAM W/R** |
| INDF | 00.7~0 | R/W | - | Not a physical register, addressing INDF actually point to the register whose address is contained in the FSR register |
| **TM0  (01h/101h)** | | | | **Function related to: Timer0** |
| TM0 | 01.7~0 | R/W | 0 | Timer0 content |
| **PCL  (02h/82h/105h/182h)** | | | | **Function related to: PROGRAM COUNT** |
| PCL | 02.7~0 | R/W | 0 | Programming Counter LSB [7~0] |
| **STATUS  (03h/83h/103h/183h)** | | | | **Function related to: STATUS** |
| IRP | 03.7 | R/W | 0 | Register Bank Select bit (used for indirect addressing) |
| RP1 | 03.6 | R/W | 0 | Register Bank Select bit 1 (used for direct addressing) |
| RP0 | 03.5 | R/W | 0 | Register Bank Select bit 0 (used for direct addressing) |
| TO | 03.4 | R | 0 | WDT timeout flag, cleared by PWRST, ‘SLEEP’ or ‘CLRWDT’ instruction |
| PD | 03.3 | R | 0 | Power down flag, set by ‘SLEEP’, cleared by ‘CLRWDT’ instruction |
| Z | 03.2 | R/W | 0 | Zero flag |
| DC | 03.1 | R/W | 0 | Decimal Carry flag |
| C | 03.0 | R/W | 0 | Carry flag |
| **FSR  (04h/84h/104h/184h)** | | | | **Function related to: RAM W/R** |
| FSR | 04.7~0 | R/W | - | File Select Register, indirect address mode pointer |
| **PAD  (05h)** | | | | **Function related to: Port A** |
| PAD | 05.7~0 | R | - | Port A pin or “data register” state |
| | | W | FF | Port A output data register |
| **PBD  (06h)** | | | | **Function related to: Port B** |
| PBD | 06.7~0 | R | - | Port B pin or “data register” state |
| | | W | FF | Port B output data register |
| **PCD  (07h)** | | | | **Function related to: Port C** |
| PDD | 07.3~0 | R | - | Port C pin or “data register” state |
| | | W | FF | Port C output data register |
| **PDD  (08h)** | | | | **Function related to: Port D** |
| PDD | 08.7~0 | R | - | Port D pin or “data register” state |
| | | W | FF | Port D output data register |
| **PCLATH  (0Ah/8Ah/10Ah/18Ah)** | | | | **Function related to: PROGRAM COUNT** |
| PCLATH | 0A.3~0 | R/W | 0 | Write Buffer for the upper 4 bits of the Program Counter |
| **INTIE (0Bh/8Bh/10Bh/18Bh)** | | | | **Function related to: Interrupt Enable** |
| ADCIE | 0B.7 | R/W | 0 | ADC interrupt enable, 1=enable, 0=disable |
| T2IE | 0B.6 | R/W | 0 | T2 interrupt enable, 1=enable, 0=disable |
| TM1IE | 0B.5 | R/W | 0 | Timer1 interrupt enable, 1=enable, 0=disable |
| TM0IE | 0B.4 | R/W | 0 | Timer0 interrupt enable, 1=enable, 0=disable |
| WKTIE | 0B.3 | R/W | 0 | Wakeup Timer interrupt enable, 1=enable, 0=disable<br>Set 0 to clear & disable WKT timer |
| INT2IE | 0B.2 | R/W | 0 | INT2 pin (PA7) interrupt enable, 1=enable, 0=disable |
| INT1IE | 0B.1 | R/W | 0 | INT1 pin (PA4) interrupt enable, 1=enable, 0=disable |
| INT0IE | 0B.0 | R/W | 0 | INT0 pin (PA0) interrupt enable, 1=enable, 0=disable |

| Name | Address | R/W | Rst | Description |
|---|---|---|---|---|
| **INTIF (0Ch)** | | | | **Function related to: Interrupt Flag** |
| ADCIF | 0C.7 | R | - | ADC interrupt flag, set by H/W after end of ADC conversion |
| | | W | 0 | write 0: clear this flag; write 1: no action |
| T2IF | 0C.6 | R | - | T2 interrupt event pending flag, set by H/W while T2 overflows |
| | | W | 0 | write 0: clear this flag; write 1: no action |
| TM1IF | 0C.5 | R | - | Timer1 interrupt event pending flag, set by H/W while Timer1 overflows |
| | | W | 0 | write 0: clear this flag; write 1: no action |
| TM0IF | 0C.4 | R | - | Timer0 interrupt event pending flag, set by H/W while Timer0 overflows |
| | | W | 0 | write 0: clear this flag; write 1: no action |
| WKTIF | 0C.3 | R | - | WKT interrupt event pending flag, set by H/W while WKT time out |
| | | W | 0 | write 0: clear this flag; write 1: no action |
| INT2IF | 0C.2 | R | - | INT2 (PA7) interrupt event pending flag, set by H/W at INT2 pin’s falling edge |
| | | W | 0 | write 0: clear this flag; write 1: no action |
| INT1IF | 0C.1 | R | - | INT1 (PA4) interrupt event pending flag, set by H/W at INT1 pin’s falling/rising edge |
| | | W | 0 | write 0: clear this flag; write 1: no action |
| INT0IF | 0C.0 | R | - | INT0 (PA0) interrupt event pending flag, set by H/W at INT0 pin’s falling/rising edge |
| | | W | 0 | write 0: clear this flag; write 1: no action |
| **CLKCTL (0Fh)** | | | | **Function related to: Fsys** |
| - | 0F.7~5 | - | - | Reserved |
| SLOWSTP | 0F.4 | R/W | 0 | Stop Slow-clock in Stop Mode<br>0: no Stop 1: Stop |
| FASTSTP | 0F.3 | R/W | 1 | Stop Fast-clock<br>0:no Stop 1:Stop |
| CPUCKS | 0F.2 | R/W | 0 | Select Fast-clock<br>0: Fsys=Slow-clock 1: Fsys=Fast-clock |
| CPUPSC | 0F.1~0 | R/W | 11 | Fsys Prescaler,<br>0: div 16, 1: div 4, 2: div 2, 3: div 1 |
| **TM0RLD (10h)** | | | | **Function related to: TM0** |
| TM0RLD | 10.7~0 | R/W | 0 | Timer0 reload Data |
| **TM0CTL (11h)** | | | | **Function related to: TM0** |
| | 11.7~6 | | | |
| TM0EDG | 11.5 | R/W | 0 | Timer0 prescaler counting edge for TM0CKI pin<br>0: rising edge 1: falling edge |
| TM0CKS | 11.4 | R/W | 0 | Timer0 prescaler clock source<br>0: Fsys/2 1: TM0CKI pin (PA2 pin) |
| TM0PSC | 11.3~0 | R/W | 0 | Timer0 prescaler. Timer0 prescaler clock source divided by<br>0000: /1 0101: /32 1010: /1024 1111: /32768<br>0001: /2 0110: /64 1011: /2048<br>0010: /4 0111: /128 1100: /4096<br>0011: /8 1000: /256 1101: /8192<br>0100: /16 1001: /512 1110: /16384 |

| Name | Address | R/W | Rst | Description |
|---|---|---|---|---|
| **TM1  (12h)** | | | | **Function related to: Timer1** |
| TM1 | 12.7~0 | R/W | 0 | Timer1 content |
| **TM1RLD  (13h)** | | | | **Function related to: Timer1** |
| TM1RLD | 13.7~0 | R/W | 0 | Timer1 reload Data |
| **TM1CTL  (14h)** | | | | **Function related to: Timer1** |
| TM1PSC | 14.3~0 | R/W | 0 | Timer1 prescaler. Timer1 clock source<br>0000: Fsys/2<br>0001: Fsys/4<br>0010: Fsys/8<br>0011: Fsys/16<br>0100: Fsys/32<br>0101: Fsys/64<br>0110: Fsys/128<br>0111: Fsys/256<br>1xxx: Fsys/512 |
| **T2CTL  (15h)** | | | | **Function related to: T2** |
| T2CKS | 15.2 | R/W | 0 | T2 clock source selection.<br>1: Fsys/128   0: Slow-clock |
| T2PSC | 15.1~0 | R/W | 0 | T2 prescaler. T2 clock source divided by -<br>00: 32768     01: 16384     10: 8192     11: 128 |
| **MF016  (16h)** | | | | **Function related to: T2/TM1/TM0/LVR/LVD** |
| LVDF | 16.7 | R | - | Low voltage detection flag, set by H/W while Vcc ≦ LVD |
| LVDEN | 16.6 | R/W | 0 | Low voltage detection function enable, (When LVR=2.4V)<br>1=enable,<br>0=disable |
| T2CLR | 16.5 | R/W | 1 | T2 counter clear<br>0: Release     1: Stop counting |
| TM1STP | 16.4 | R/W | 0 | Timer1 counter stop<br>0: Release     1: Stop counting |
| TM0STP | 16.3 | R/W | 0 | Timer0 counter stop<br>0: Release     1: Stop counting |
| LVRSAV | 16.2 | R/W | 1 | LVR/LVD power save<br>1: LVR/LVD auto power off in STOP/IDLE mode<br>0: LVR/LVD enable in in STOP/IDLE mode |
| LVDS | 16.1~0 | R/W | 01 | LVD select  00: 3.2V   01: 2.5V  1x: 3.7V<br>(When LVR=2.4V) |
| **ADCH  (17h)** | | | | **Function related to: ADC** |
| ADCH | 17.7~0 | R | - | ADC output data MSB, ADQ [11:4] |
| **ADCTL  (18h)** | | | | **Function related to: ADC** |
| ADCL | 18.7~4 | R | - | ADC output data LSB, ADQ [3:0] |
| ADST | 18.3 | R/W | 0 | ADC start bit.<br>0: H/W clear after end of conversion    1: ADC start conversion |
| ADCKS | 18.2~0 | R/W | 0 | ADC clock frequency selection:<br>000: Fsys/256     100: Fsys/16<br>001: Fsys/128     101: Fsys/8<br>010: Fsys/64       110: Fsys/4<br>011: Fsys/32       111: Fsys/2 |

| Name | Address | R/W | Rst | Description |
|---|---|---|---|---|
| **MF019 (19h)** | | | | **Function related to: ADC** |
| - | 19.7 | - | - | Reserved |
| BGEN | 19.6 | R/W | 1 | Band Gap BG1.25V & Vtemp function enable<br>0: Disable<br>1: Enable and Auto disable in STOP/IDLE mode |
| ADCVREFS | 19.5~4 | R/W | 0 | ADC VREF select,<br>00: VCC, 01: 2.5V, 10: 3V, 11: 4V |
| ADCHS | 19.3~0 | R/W | 0 | ADC channel select<br>0000: ADC0 (PD0) 0110: ADC6 (PD6) 1100: VBG1.2V<br>0001: ADC1 (PD1) 0111: ADC7 (PD7) 1101: VSS<br>0010: ADC2 (PD2) 1000: ADC8(PA0) 1110: VSS<br>0011: ADC3 (PD1) 1001: ADC9(PA1)<br>0100: ADC4 (PD4) 1010: ADC10(PA2)<br>0101: ADC5 (PD5) 1011: ADC11(PA3) |
| **PWM0D (1Ah)** | | | | **Function related to: PWM0** |
| PWM0D | 1A.7~0 | R/W | 0 | PWM0 Duty |
| **PWM1AD (1Bh)** | | | | **Function related to: PWM1** |
| PWM1AD | 1B.7~0 | R/W | 0 | PWM1A Duty |
| **PWM1BD (1Ch)** | | | | **Function related to: PWM1** |
| PWM1BD | 1C.7~0 | R/W | 0 | PWM1B Duty |
| **PWM1CD (1Dh)** | | | | **Function related to: PWM1** |
| PWM1CD | 1D.7~0 | R/W | 0 | PWM1C Duty |
| **PWM2D (1Eh)** | | | | **Function related to: PWM2** |
| PWM2D | 1E.7~0 | R/W | 0 | PWM2 Duty |
| **PWMCTL0 (1Fh)** | | | | **Function related to: PWM0/1/2** |
| PWM2CKS | 1F.7~6 | R/W | 0 | PWM2 Clock Source<br>0x: Fsys<br>10:FIRC16M<br>11:FIRC32M |
| PWM2PSC | 1F.5~4 | R/W | 0 | PWM2 Clock Source Prescaler<br>00: DIV1 01: DIV2 10:DIV4 11:DIV64 |
| PWM2CLR | 1F.2 | R/W | 0 | PWM2 clear and hold<br>0:PWM2 enable 1:PWM2 clear and hold |
| PWM1CLR | 1F.1 | R/W | 0 | PWM1 clear and hold<br>0:PWM1 enable 1:PWM1 clear and hold |
| PWM0CLR | 1F. | R/W | 0 | PWM0 clear and hold<br>0:PWM0 enable 1:PWM0 clear and hold |
| **User Data Memory** | | | | |
| RAM | 20~6F | R/W | - | RAM Bank0 area (80 Bytes) |
| RAM | 70~7F | R/W | - | RAM common area (16 Bytes) |

| Name | Address | R/W | Rst | Description |
|---|---|---|---|---|
| **OPTION (81h/181h)** | | | | **Function related to: STATUS/INT0/INT1/WDT/WKT** |
| HWAUTO | 81.7 | R/W | 0 | Enter interrupt vector, HW auto save/restore WREG and STATUS w/o TO, PD<br>0:disable 1: Enable |
| INT0EDG | 81.6 | R/W | 0 | INT0 pin edge interrupt event<br>0: falling edge to trigger<br>1: rising edge to trigger |
| INT1EDG | 81.5 | R/W | 0 | INT1 pin edge interrupt event<br>0: falling edge to trigger<br>1: rising edge to trigger |
| WDTPSC | 81.3~2 | R/W | 11 | WDT pre-scale selections:<br>00: 128mS<br>01: 256mS<br>10: 1024mS<br>11: 2048mS |
| WKTPSC | 81.1~0 | R/W | 11 | WKT pre-scale selections:<br>00: 16mS<br>01: 32mS<br>10: 64mS<br>11: 128mS |
| **PAMODH (8Ch)** | | | | **Function related to: Port A** |
| PA7MOD | 8C.7~6 | R/W | 00 | PA7~PA4 I/O mode control<br>00: Mode0<br>01: Mode1<br>10: Mode2<br>11: Mode3 |
| PA6MOD | 8C.5~4 | R/W | 01 | |
| PA5MOD | 8C.3~2 | R/W | 01 | |
| PA4MOD | 8C.1~0 | R/W | 01 | |
| **PAMODL (8Dh)** | | | | **Function related to: Port A** |
| PA3MOD | 8D.7~6 | R/W | 01 | PA3~PA0 I/O mode control<br>00: Mode0<br>01: Mode1<br>10: Mode2<br>11: Mode3 |
| PA2MOD | 8D.5~4 | R/W | 01 | |
| PA1MOD | 8D.3~2 | R/W | 01 | |
| PA0MOD | 8D.1~0 | R/W | 01 | |
| **PBMODH (8Eh)** | | | | **Function related to: Port B** |
| PB7MOD | 8E.7~6 | R/W | 01 | PB7~PB4 I/O mode control<br>00: Mode0<br>01: Mode1<br>10: Mode2<br>11: Mode3 |
| PB6MOD | 8E.5~4 | R/W | 01 | |
| PB5MOD | 8E.3~2 | R/W | 01 | |
| PB4MOD | 8E.1~0 | R/W | 01 | |
| **PBMODL (8Fh)** | | | | **Function related to: Port B** |
| PB3MOD | 8F.7~6 | R/W | 01 | PB3~PB0 I/O mode control<br>00: Mode0<br>01: Mode1<br>10: Mode2<br>11: Mode3 |
| PB2MOD | 8F.5~4 | R/W | 01 | |
| PB1MOD | 8F.3~2 | R/W | 01 | |
| PB0MOD | 8F.1~0 | R/W | 01 | |
| **PCMODL (90h)** | | | | **Function related to: Port C** |
| PC3MOD | 90.7~6 | R/W | 01 | PC3~PC0 I/O mode control<br>00: Mode0<br>01: Mode1<br>10: Mode2<br>11: Mode3 |
| PC2MOD | 90.5~4 | R/W | 01 | |
| Pc1MOD | 90.3~2 | R/W | 01 | |
| PC0MOD | 90.1~0 | R/W | 01 | |

| Name | Address | R/W | Rst | Description |
|---|---|---|---|---|
| **PDMODH (91h)** | | | | **Function related to: Port D** |
| PD7MOD | 91.7~6 | R/W | 01 | PD7~PD4 I/O mode control<br>00: Mode0<br>01: Mode1<br>10: Mode2<br>11: Mode3 |
| PD6MOD | 91.5~4 | R/W | 01 | |
| PD5MOD | 91.3~2 | R/W | 01 | |
| PD4MOD | 91.1~0 | R/W | 01 | |
| **PDMODL (92h)** | | | | **Function related to: Port D** |
| PD3MOD | 91.7~6 | R/W | 01 | Pd3~PD0 I/O mode control<br>00: Mode0<br>01: Mode1<br>10: Mode2<br>11: Mode3 |
| PD2MOD | 91.5~4 | R/W | 01 | |
| PD1MOD | 91.3~2 | R/W | 01 | |
| PD0MOD | 91.1~0 | R/W | 01 | |
| **PWMCTL1 (94h)** | | | | **Function related to: PWM0/PWM1** |
| PWM1CKS | 94.7~6 | R/W | 0 | PWM1 Clock Source<br>0x: Fsys<br>10:FIRC16M<br>11:FIRC32M |
| PWM1PSC | 94.5~4 | R/W | 1 | PWM1 Clock Source Prescaler<br>00: DIV1 01: DIV2 10:DIV4 11:DIV64 |
| PWM0CKS | 94.3~2 | R/W | 0 | PWM0 Clock Source<br>0x: Fsys<br>10:FIRC16M<br>11:FIRC32M |
| PWM0PSC | 94.1~0 | R/W | 0 | PWM0 Clock Source Prescaler<br>00: DIV1 01: DIV2 10:DIV4 11:DIV64 |
| **PWM0PRD (95h)** | | | | **Function related to: PWM0** |
| PWM0PRD | 95.7~0 | R/W | 0 | PWM0 period data |
| **PWM1PRD (96h)** | | | | **Function related to: PWM1** |
| PWM1PRD | 96.1~0 | R/W | 0 | PWM1 period data |
| **PWM2PRD (97h)** | | | | **Function related to: PWM2** |
| PWM2PRD | 96.1~0 | R/W | 0 | PWM2 period data |
| **MF098 (98h)** | | | | **Function related to: PWM0/PWM1/PWM2/TM1/TC** |
| TCOE | 98.7 | R/W | 0 | TCOUT Output Enable<br>0: Disable 1: Enable, output to PB1 |
| TM1OE | 98.6 | R/W | 0 | Timer1 overflow toggle Output Enable<br>0: Disable 1:Enable, output to PB0 |
| PWM0OE | 98.5 | R/W | 0 | PWM0 Output Enable<br>0: Disable 1:Enable, PWM0 output to PA0 |
| PWM2OE | 98.4 | R/W | 0 | PWM2 Output Enable<br>0: Disable 1:Enable, PWM2 output to PA4 |
| PWM1AOE | 98.3 | R/W | 0 | PWM1A Output Enable<br>0: Disable 1:Enable, PWM1A output to PA1 |
| PWM1BOE | 98.2 | R/W | 0 | PWM1B Output Enable<br>0: Disable 1:Enable, PWM1B output to PA2 |
| PWM1COE | 98.1 | R/W | 0 | PWM1C Output Enable<br>0: Disable 1:Enable, PWM1C output to PA3 |

| Name | Address | R/W | Rst | Description |
|---|---|---|---|---|
| **TKM0CON (9Ah)** | | | | **Function related to: Touch Key Module0** |
| TKM0TCP | 9A.7~5 | R/W | 1 | TK module0 touch key clock frequency select;<br>000:slowest, …, 111:fastest |
| TKM0REFC | 9A.4~2 | R/W | 0 | TK module0 Reference clock capacitor select;<br>000:smallest, …, 111:biggest(conversion time longest) |
| TKM0FSL | 9A.1~0 | R/W | 0 | TK module0 clock(RCK0/TCK0) frequency selection;<br>00: slowest, …, 11: fastest |
| **TKM1CON (9Bh)** | | | | **Function related to: Touch Key Module1** |
| TKM1TCP | 9B.7~5 | R/W | 1 | TK module1 touch key clock frequency select;<br>000:slowest, …, 111:fastest |
| TKM1REFC | 9B.4~2 | R/W | 0 | TK module1 Reference clock capacitor select;<br>000:smallest, …, 111:biggest(conversion time longest) |
| TKM1FSL | 9B.1~0 | R/W | 0 | TK module1 clock(RCK1/TCK1) frequency selection;<br>00: slowest, …, 11: fastest |
| **TKM2CON (9Ch)** | | | | **Function related to: Touch Key Module2** |
| TKM2TCP | 9C.7~5 | R/W | 1 | TK module2 touch key clock frequency select;<br>000:slowest, …, 111:fastest |
| TKM2REFC | 9C.4~2 | R/W | 0 | TK module2 Reference clock capacitor select;<br>000:smallest, …, 111:biggest(conversion time longest) |
| TKM2FSL | 9C.1~0 | R/W | 0 | TK module2 clock(RCK2/TCK2) frequency selection;<br>00: slowest, …, 11: fastest |
| **User Data Memory** | | | | |
| RAM | A0~EF | R/W | - | RAM Bank1area (80 Bytes) |
| **TESTREG (105h)** | | | | **Function related to: TEST** |
| TESTREG | 105.1~0 | R/W | 11 | Test bits, Keep 11 |
| **LVRPD (109h)** | | | | **Function related to: LVR** |
| LVRPD | 109 | W | - | Write 0x37 to force LVR disable |
| **BGTRIM (10Eh)** | | | | **Function related to: VBG** |
| BGTRIM | 10E.3~0 | R/W | | VBG voltage adjustment<br>00h: Lowest voltage     Fh: Highest voltage |
| **I2CTXD0 (110h)** | | | | **Function related to: Slave I2C** |
| I2CTXD0 | 110.7~0 | R/W | 0 | The transmitting register0 of slave I2C |
| **I2CTXD1 (111h)** | | | | **Function related to: Slave I2C** |
| I2CTXD1 | 111.7~0 | R/W | 0 | The transmitting register1 of slave I2C |
| **I2CCTL (112h)** | | | | **Function related to: Slave I2C** |
| I2CIE | 112.4 | R/W | 0 | I2C Interrupt Enable<br>0: Disable<br>1: Enable |
| I2CEN | 112.3 | R/W | 0 | Slave I2C interface enable<br>0: Disable<br>1: Enable |
| I2CID | 112.1~0 | R/W | 0 | Slave I2C ID last 2 bits |

| Name | Address | R/W | Rst | Description |
|---|---|---|---|---|
| **I2CFLG (113h)** | | | | **Function related to: Touch Key Module0** |
| I2CIF | 113.6 | R/W | 0 | Slave I2C interrupt pending flag<br>This bis is set by H/W while<br>✧ I2CRCD0 or I2CRCD1 receive data finished<br>✧ I2CRCD0 or I2CRCD1 data overflow occurred<br>✧ I2CTXD0 or I2CTXD1 data transmit finished.<br>Write 0 to clear this flag |
| TXD1F | 113.5 | R/W | 0 | Slave I2C transmitting data register 1 flag, set by H/W while I2CTXD1 data transmitting finished<br>Write 0 to clear this flag |
| TXD0F | 113.4 | R/W | 0 | Slave I2C transmitting data register 0 flag, set by H/W while I2CTXD0 data transmitting finished<br>Write 0 to clear this flag |
| RCD1OVF | 113.3 | R/W | 0 | Slave I2C receiving data register 1 overflow, set by H/W while receiving data to I2CRCD1 overflow.<br>Write 0 to clear this flag |
| RCD1F | 113.2 | R/W | 0 | Slave I2C receiving data register 1 flag, set by H/W while I2CRCD1 data receiving finished<br>Write 0 to clear this flag |
| RCD0OVF | 113.1 | R/W | 0 | Slave I2C receiving data register 0 overflow, set by H/W while receiving data to I2CRCD0 overflow.<br>Write 0 to clear this flag |
| RCD0F | 113.0 | R/W | 0 | Slave I2C receiving data register 0 flag, set by H/W while I2CRCD0 data receiving finished<br>Write 0 to clear this flag |
| **I2CRCD0 (114h)** | | | | **Function related to: Slave I2C** |
| I2CRCD0 | 114.7~0 | R | 0 | The receiving register0 of slave I2C |
| **I2CRCD1 (115h)** | | | | **Function related to: Slave I2C** |
| I2CRCD1 | 115.7~0 | R | 0 | The receiving register1 of slave I2C |
| **User Data Memory** | | | | |
| RAM | 120~16F | R/W | - | RAM Bank2 area (80 Bytes) |
| **DPL (185h)** | | | | **Function related to: Table Read** |
| DPL | 185.7~0 | R/W | 0 | Table read low address, data ROM pointer (DPTR) low byte |
| **DPH (186h)** | | | | **Function related to: Table Read** |
| DPH | 186.3~0 | R/W | 0 | Table read high address, data ROM pointer (DPTR) high byte |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| **TABR (18Ch)** | | | | **Function related to: Table Read** |
| TABR | 18C.7~0 | R/W | 0 | 1: TABR write 1 = opcode TABRL<br>2: TABR write 2 = opcode TABRH<br>3: after step1 or step2, read TABR to get main ROM table read value<br>after step1, read TABR to get EEPROM value (when EEPEN=E2h) |

| Name | Address | R/W | Rst | Description |
|---|---|---|---|---|
| | | | | Table Read for ASM: TABRL/TABRH or TABR<br>Table Read for C: TABR |
| **EEPCTL (18Dh)** | | | | **Function related to: EEPROM** |
| EEPTO | 18D.7 | R | 0 | EEPROM Write Time-out flag |
| EEPTE | 18D.1~0 | R/W | 0 | EEPROM Write Time-out enable;<br>00: Disable; 01: 0.9ms; 10: 3.6ms; 11: 7.2ms |
| **EEPEN (18Eh)** | | | | **Function related to: EEPROM** |
| EEPEN | 18E.7~0 | W | 0 | EEPROM Enable<br>0xE2: Enable<br>Others: Disable |
| **EEPDT (18Fh)** | | | | **Function related to: EEPROM** |
| EEPDT | 18F.7~0 | W | 0 | EEPROM Data to write |
| **TKMCTL0 (191h)** | | | | **Function related to: Touch Key Module** |
| TKM2SOC | 191.6 | R/W | 0 | STK Module2 Start of Conversion,<br>HW clear it while end of conversion |
| TKM1SOC | 191.5 | R/W | 0 | STK Module1 Start of Conversion,<br>HW clear it while end of conversion |
| TKM0SOC | 191.4 | R/W | 0 | STK Module0 Start of Conversion,<br>HW clear it while end of conversion |
| TKM2EOC | 191.2 | R | - | STK Module2 End of Conversion |
| TKM1EOC | 191.1 | R | - | STK Module1 End of Conversion |
| TKM0EOC | 191.0 | R | - | STK Module0 End of Conversion |
| **TKMCTL1 (192h)** | | | | **Function related to: Touch Key Module** |
| TKMHSEN | 192.7 | R/W | 0 | All STK Module Sensitivity<br>1: higher sensitivity<br>0: lower sensitivuty |
| TKM2PD | 192.6 | R/W | 1 | STK Module2 Power Down<br>1: Touch Key enable<br>0: Touch Key disable |
| TKM1PD | 192.5 | R/W | 1 | STK Module1 Power Down<br>1: Touch Key enable<br>0: Touch Key disable |
| TKM0PD | 192.4 | R/W | 1 | STK Module0 Power Down<br>1: Touch Key enable<br>0: Touch Key disable |
| TKIE | 192.3 | R/W | 0 | Touch Key interrupt enable<br>0: Disable Touch Key interrupt<br>1: Enable Touch Key interrupt |
| TKM2JMP | 192.2 | R/W | 0 | STK Module2 touch key clock mode<br>0: fixed frequency<br>1: auto-change frequency |
| TKM1JMP | 192.1 | R/W | 0 | STK Module1 touch key clock mode<br>0: fixed frequency<br>1: auto-change frequency |
| TKM0JMP | 192.0 | R/W | 0 | STK Module1 touch key clock mode<br>0: fixed frequency<br>1: auto-change frequency |

| Name | Address | R/W | Rst | Description |
|---|---|---|---|---|
| **TKMCHS (193h)** | | | | **Function related to: Touch Key Module0/1/2** |
| TKM2CHS | 193.6~4 | R/W | 0 | STK Module2 Channel Select:<br>0:TK8 1:TK9 2:TK10 3:TK11 4:TK12 5:TK13 6:TK14 7:TK15 |
| TKM1CHS | 193.3~2 | R/W | 0 | STK Module1 Channel Select:<br>0:TK4 1:TK5 2:TK6 3:TK7 |
| TKM0CHS | 193.1~0 | R/W | 0 | STK Module0 Channel Select:<br>0:TK0 1:TK1 2:TK2 3:TK3 |
| **TKMFLG (194h)** | | | | **Function related to: Touch Key Module0/1/2** |
| TKM2IF | 194.6 | R/W | 0 | STK Module2 Interrupt event pending flag<br>set by H/W at the STK module2 end of conversion<br>S/W writes 0 to TKM2IF or sets the TKM2SOC bit to clear this flag. |
| TKM1IF | 194.5 | R/W | 0 | STK Module1 Interrupt event pending flag<br>set by H/W at the STK module1 end of conversion<br>S/W writes 0 to TKM1IF or sets the TKM1SOC bit to clear this flag. |
| TKM0IF | 194.4 | R/W | 0 | STK Module0 Interrupt event pending flag<br>set by H/W at the STK module0 end of conversion<br>S/W writes 0 to TKM0IF or sets the TKM0SOC bit to clear this flag. |
| TKIF | 194.3 | R/W | 0 | Touch Key interrupt event pending flag,<br>set by H/W at the end of all Touch Key conversion<br>S/W writes 0 to TKIF or sets the TKMxSOC bit to clear this flag. |
| **TKM0DL (195h)** | | | | **Function related to: Touch Key Module0** |
| TKM0DL | 195.7~0 | R | - | STK Module0 Data LSB [7~0] |
| **TKM1DL (196h)** | | | | **Function related to: Touch Key Module1** |
| TKM1DL | 196.7~0 | R | - | STK Module1 Data LSB [7~0] |
| **TKM2DL (197h)** | | | | **Function related to: Touch Key Module2** |
| TKM2DL | 197.7~0 | R | - | STK Module2 Data LSB [7~0] |
| **TKM10DH (198h)** | | | | **Function related to: Touch Key Module0/1** |
| TKM1DH | 198.7~4 | R | - | STK Module1 Data MSB [11~8] |
| TKM0DH | 198.3~0 | R | - | STK Module0 Data MSB [11~8] |
| **TKM2DH (199h)** | | | | **Function related to: Touch Key Module2** |
| TKM2DH | 199.3~0 | R | - | STK Module2 Data MSB [11~8] |
| **TKM0TMRH (19Ah)** | | | | **Function related to: Touch Key Module0** |
| TKM0TMRH | 19A.3~0 | R/W | 0 | STK Module0 reference counter MSB [11~8] |
| **TKM0TMRL (19Bh)** | | | | **Function related to: Touch Key Module0** |
| TKM0TMRL | 19B.7~0 | R/W | 0 | STK Module0 reference counter LSB [7~0] |
| **TKM1TMRH (19Ch)** | | | | **Function related to: Touch Key Module1** |
| TKM1TMRH | 19C.3~0 | R/W | 0 | STK Module1 reference counter MSB [11~8] |
| **TKM1TMRL (19Dh)** | | | | **Function related to: Touch Key Module1** |
| TKM1TMRL | 19D.7~0 | R/W | 0 | STK Module1 reference counter LSB [7~0] |
| **TKM2TMRH (19Eh)** | | | | **Function related to: Touch Key Module2** |
| TKM2TMRH | 19E.3~0 | R/W | 0 | STK Module2 reference counter MSB [11~8] |
| **TKM2TMRL (19Fh)** | | | | **Function related to: Touch Key Module2** |
| TKM2TMRL | 19F.7~0 | R/W | 0 | STK Module2 reference counter LSB [7~0] |
| **User Data Memory** | | | | |
| RAM | 1A0~1EF | R/W | - | RAM Bank3 area (80 Bytes) |

# INSTRUCTION SET

Each instruction is a 16-bit word divided into an Op Code, which specifies the instruction type, and one or more operands, which further specify the operation of the instruction. The instructions can be categorized as byte-oriented, bit-oriented and literal operations list in the following table.

For byte-oriented instructions, "f" represents the address designator and "d" represents the destination designator. The address designator is used to specify which address in Program memory is to be used by the instruction. The destination designator specifies where the result of the operation is to be placed. If "d" is "0", the result is placed in the W register. If "d" is "1", the result is placed in the address specified in the instruction.

For bit-oriented instructions, "b" represents a bit field designator, which selects the number of the bit affected by the operation, while "f" represents the address designator. For literal operations, "k" represents the literal or constant value.

| Field/Legend | Description |
|---|---|
| f | Register File Address |
| b | Bit address |
| k | Literal. Constant data or label |
| d | Destination selection field, 0: Working register, 1: Register file |
| W | Working Register |
| Z | Zero Flag |
| C | Carry Flag or/Borrow Flag |
| DC | Decimal Carry Flag or Decimal/Borrow Flag |
| PC | Program Counter |
| TOS | Top Of Stack |
| GIE | Global Interrupt Enable Flag (i-Flag) |
| [] | Option Field |
| ( ) | Contents |
| . | Bit Field |
| B | Before |
| A | After |
| ← | Assign direction |

| Mnemonic | | Op Code | Cycle | Flag Affect | Description |
|---|---|---|---|---|---|
| **Byte-Oriented File Register Instruction** | | | | | |
| ADDWX | f, d | ff00 0111 dfff ffff | 1 | C, DC, Z | Add W and "f" |
| ANDWX | f, d | ff00 0101 dfff ffff | 1 | Z | AND W with "f" |
| CLRX | f | ff00 0001 1fff ffff | 1 | Z | Clear "f" |
| CLRW | | 0000 0001 0100 0000 | 1 | Z | Clear W |
| COMX | f, d | ff00 1001 dfff ff ff | 1 | Z | Complement "f" |
| DECX | f, d | ff00 0011 dfff ffff | 1 | Z | Decrement "f" |
| DECXSZ | f, d | ff00 1011 dfff ffff | 1 or 2 | - | Decrement "f", skip if zero |
| INCX | f, d | ff00 1010 dfff ffff | 1 | Z | Increment "f" |
| INCXSZ | f, d | ff00 1111 dfff ffff | 1 or 2 | - | Increment "f", skip if zero |
| IORWX | f, d | ff00 0100 dfff ffff | 1 | Z | OR W with "f" |
| MOVX | f,d | ff00 1000 dfff ffff | 1 | Z | Move "f" |
| MOVXW | f | ff00 1000 0fff ffff | 1 | Z | Move "f" to W |
| MOVWX | f | ff00 0000 1fff ffff | 1 | - | Move W to "f" |
| RLX | f, d | ff00 1101 dfff ffff | 1 | C | Rotate left "f" through carry |
| RRX | f, d | ff00 1100 dfff ffff | 1 | C | Rotate right "f" through carry |
| SUBWX | f, d | ff00 0010 dfff ffff | 1 | C, DC, Z | Subtract W from "f" |
| SWAPX | f, d | ff00 1110 dfff ffff | 1 | - | Swap nibbles in "f" |
| TSTX | f | ff00 1000 1fff ffff | 1 | Z | Test if "f" is zero |
| XORWX | f, d | ff00 0110 dfff ffff | 1 | Z | XOR W with "f" |
| **Bit-Oriented File Register Instruction** | | | | | |
| BCX | f, b | ff11 00bb bfff ffff | 1 | - | Clear "b" bit of "f" |
| BSX | f, b | ff11 01bb bfff ffff | 1 | - | Set "b" bit of "f" |
| BTXSC | f, b | ff11 10bb bfff ffff | 1 or 2 | - | Test "b" bit of "f", skip if clear |
| BTXSS | f, b | ff11 11bb bfff ffff | 1 or 2 | - | Test "b" bit of "f", skip if set |
| **Literal and Control Instruction** | | | | | |
| ADDLW | k | 0001 1100 kkkk kkkk | 1 | C, DC, Z | Add Literal "k" and W |
| ANDLW | k | 0001 1011 kkkk kkkk | 1 | Z | AND Literal "k" with W |
| CALL | k | 0010 0kkk kkkk kkkk | 2 | - | Call subroutine "k" |
| CLRWDT | | 0001 1110 0000 0100 | 1 | TO, PD | Clear Watch Dog Timer |
| GOTO | k | 0010 1kkk kkkk kkkk | 2 | - | Jump to branch "k" |
| IORLW | k | 0001 1010 kkkk kkkk | 1 | Z | OR Literal "k" with W |
| MOVLW | k | 0001 1001 kkkk kkkK | 1 | - | Move Literal "k" to W |
| NOP | | 0000 0000 0000 0000 | 1 | - | No operation |
| RET | | 0000 0000 0100 0000 | 2 | - | Return from subroutine |
| RETI | | 0000 0000 0110 0000 | 2 | - | Return from interrupt |
| RETLW | k | 0001 1000 kkkk kkkk | 2 | - | Return with Literal in W |
| SLEEP | | 0001 1110 0000 0011 | 1 | TO, PD | Go into Power-down mode, Clock oscillation stops |
| SUBLW | k | 0001 1111 kkkk kkkk | 1 | C, DC, Z | Subtract W from literal |
| TABRH | | 0000 0000 0101 1000 | 2 | - | Lookup ROM high data to W |
| TABRL | | 0000 0000 0101 0000 | 2 | - | Lookup ROM low data to W |
| XORLW | k | 0001 1101 kkkk kkkk | 1 | Z | XOR Literal "k" with W |

| **ADDLW** | **Add Literal "k" and W** | |
|---|---|---|
| Syntax | ADDLW k | |
| Operands | k : 00h ~ FFh | |
| Operation | (W) ← (W) + k | |
| Status Affected | C, DC, Z | |
| OP-Code | 0001 1100 kkkk kkkk | |
| Description | The contents of the W register are added to the eight-bit literal 'k' and the result is placed in the W register. | |
| Cycle | 1 | |
| Example | ADDLW 0x15 | B : W =0x10<br>A : W =0x25 |

| **ADDWX** | **Add W and "f"** | |
|---|---|---|
| Syntax | ADDWX f [,d] | |
| Operands | f : 00h ~ 1FFh, d : 0, 1 | |
| Operation | (destination) ← (W) + (f) | |
| Status Affected | C, DC, Z | |
| OP-Code | ff00 0111 dfff ffff | |
| Description | Add the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'. | |
| Cycle | 1 | |
| Example | ADDWX FSR, 0 | B : W =0x17, FSR =0xC2<br>A : W =0xD9, FSR =0xC2 |

| **ANDLW** | **Logical AND Literal "k" with W** | |
|---|---|---|
| Syntax | ANDLW k | |
| Operands | k : 00h ~ FFh | |
| Operation | (W) ← (W) AND k | |
| Status Affected | Z | |
| OP-Code | 0001 1011 kkkk kkkk | |
| Description | The contents of W register are AND’ed with the eight-bit literal 'k'. The result is placed in the W register. | |
| Cycle | 1 | |
| Example | ANDLW 0x5F | B : W =0xA3<br>A : W =0x03 |

| **ANDWX** | **AND W with "f"** | |
|---|---|---|
| Syntax | ANDWX f [,d] | |
| Operands | f : 00h ~ 1FFh, d : 0, 1 | |
| Operation | (destination) ← (W) AND (f) | |
| Status Affected | Z | |
| OP-Code | ff00 0101 dfff ffff | |
| Description | AND the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'. | |
| Cycle | 1 | |
| Example | ANDWX FSR, 1 | B : W =0x17, FSR =0xC2<br>A : W =0x17, FSR =0x02 |

| **BCX** | **Clear "b" bit of "f"** | |
|---|---|---|
| Syntax | BCX f [,b] | |
| Operands | f : 00h ~ 1FFh, b : 0 ~ 7 | |
| Operation | (f.b) ← 0 | |
| Status Affected | - | |
| OP-Code | ff11 00bb bfff ffff | |
| Description | Bit 'b' in register 'f' is cleared. | |
| Cycle | 1 | |
| Example | BCX FLAG_REG, 7 | B : FLAG_REG =0xC7<br>A : FLAG_REG =0x47 |

| **BSX** | **Set "b" bit of "f"** | |
|---|---|---|
| Syntax | BSX f [,b] | |
| Operands | f : 00h ~ 1FFh, b : 0 ~ 7 | |
| Operation | (f.b) ← 1 | |
| Status Affected | - | |
| OP-Code | ff11 01bb bfff ffff | |
| Description | Bit 'b' in register 'f' is set. | |
| Cycle | 1 | |
| Example | BSX FLAG_REG, 7 | B : FLAG_REG =0x0A<br>A : FLAG_REG =0x8A |

| **BTXSC** | **Test "b" bit of "f", skip if clear(0)** | |
|---|---|---|
| Syntax | BTXSC f [,b] | |
| Operands | f : 00h ~ 1FFh, b : 0 ~ 7 | |
| Operation | Skip next instruction if (f.b) =0 | |
| Status Affected | - | |
| OP-Code | ff11 10bb bfff ffff | |
| Description | If bit 'b' in register 'f' is 1, then the next instruction is executed. If bit 'b' in register 'f' is 0, then the next instruction is discarded, and a NOP is executed instead, making this a 2nd cycle instruction. | |
| Cycle | 1 or 2 | |
| Example | LABEL1 BTXSC FLAG, 1<br>TRUE GOTO SUB1<br>FALSE ... | B : PC =LABEL1<br>A : if FLAG.1 =0, PC =FALSE<br>if FLAG.1 =1, PC =TRUE |

| **BTXSS** | **Test "b" bit of "f", skip if set(1)** | |
|---|---|---|
| Syntax | BTXSS f [,b] | |
| Operands | f : 00h ~ 1FFh, b : 0 ~ 7 | |
| Operation | Skip next instruction if (f.b) =1 | |
| Status Affected | - | |
| OP-Code | ff11 11bb bfff ffff | |
| Description | If bit 'b' in register 'f' is 0, then the next instruction is executed. If bit 'b' in register 'f' is 1, then the next instruction is discarded, and a NOP is executed instead, making this a 2nd cycle instruction. | |
| Cycle | 1 or 2 | |
| Example | LABEL1 BTXSS FLAG, 1<br>TRUE GOTO SUB1<br>FALSE ... | B : PC =LABEL1<br>A : if FLAG.1 =0, PC =TRUE<br>if FLAG.1 =1, PC =FALSE |

| **CALL** | **Call subroutine "k"** | |
|---|---|---|
| Syntax | CALL k | |
| Operands | k : 000h ~ FFFh | |
| Operation | Operation: TOS ← (PC) + 1, PC.10~0 ← k | |
| Status Affected | - | |
| OP-Code | kk10 0kkk kkkk kkkk | |
| Description | Call Subroutine. First, return address (PC+1) is pushed onto the stack. The 12-bit immediate address is loaded into PC bits <11:0>. The upper bits of PC are loaded from PCLATH. CALL is a two-cycle instruction. | |
| Cycle | 2 | |
| Example | LABEL1 CALL SUB1 | B : PC =LABEL1<br>A : PC =SUB1, TOS =LABEL1 + 1 |

| **CLRX** | **Clear "f"** | |
|---|---|---|
| Syntax | CLRX f | |
| Operands | f : 00h ~ 1FFh | |
| Operation | (f) ← 00h, Z ← 1 | |
| Status Affected | Z | |
| OP-Code | ff00 0001 1fff ffff | |
| Description | The contents of register 'f' are cleared and the Z bit is set. | |
| Cycle | 1 | |
| Example | CLRX FLAG_REG | B : FLAG_REG =0x5A<br>A : FLAG_REG =0x00, Z =1 |

| **CLRW** | **Clear W** | |
|---|---|---|
| Syntax | CLRW | |
| Operands | - | |
| Operation | (W) ← 00h, Z ← 1 | |
| Status Affected | Z | |
| OP-Code | 0000 0001 0100 0000 | |
| Description | W register is cleared and Z bit is set. | |
| Cycle | 1 | |
| Example | CLRW | B : W =0x5A<br>A : W =0x00, Z =1 |

| **CLRWDT** | **Clear Watchdog Timer** | |
|---|---|---|
| Syntax | CLRWDT | |
| Operands | - | |
| Operation | WDT/WKT Timer ← 00h | |
| Status Affected | TO, PD | |
| OP-Code | 0001 1110 0000 0100 | |
| Description | CLRWDT instruction clears the Watchdog/Wakeup Timer | |
| Cycle | 1 | |
| Example | CLRWDT | B : WDT counter =?<br>A : WDT counter =0x00 |

| **COMX** | **Complement "f"** | |
|---|---|---|
| Syntax | COMX f [,d] | |
| Operands | f : 00h ~ 1FFh, d : 0, 1 | |
| Operation | (destination) ← ($\overline{f}$) | |
| Status Affected | Z | |
| OP-Code | ff00 1001 dfff ffff | |
| Description | The contents of register 'f' are complemented. If 'd' is 0, the result is stored in W. If 'd' is 1, the result is stored back in register 'f'. | |
| Cycle | 1 | |
| Example | COMX REG1, 0 | B : REG1 =0x13<br>A : REG1 =0x13, W =0xEC |

| **DECX** | **Decrement "f"** | |
|---|---|---|
| Syntax | DECX f [,d] | |
| Operands | f : 00h ~ 1FFh, d : 0, 1 | |
| Operation | (destination) ← (f) - 1 | |
| Status Affected | Z | |
| OP-Code | ff00 0011 dfff ffff | |
| Description | Decrement register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'. | |
| Cycle | 1 | |
| Example | DECX CNT, 1 | B : CNT =0x01, Z =0<br>A : CNT =0x00, Z =1 |

| **DECXSZ** | **Decrement "f", Skip if 0** | |
|---|---|---|
| Syntax | DECXSZ f [,d] | |
| Operands | f : 00h ~ 1FFh, d : 0, 1 | |
| Operation | (destination) ← (f) - 1, skip next instruction if result is 0 | |
| Status Affected | - | |
| OP-Code | ff00 1011 dfff ffff | |
| Description | The contents of register 'f' are decremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 1, the next instruction is executed. If the result is 0, then a NOP is executed instead, making it a 2 cycle instruction. | |
| Cycle | 1 or 2 | |
| Example | LABEL1 DECXSZ CNT, 1<br>GOTO LOOP<br>CONTINUE | B : PC =LABEL1<br>A : CNT =CNT – 1<br>if CNT =0, PC =CONTINUE<br>if CNT ≠0, PC =LABEL1 + 1 |

| **GOTO** | **Unconditional Branch** | |
|---|---|---|
| Syntax | GOTO k | |
| Operands | k : 000h ~ FFFh | |
| Operation | PC.10~0 ← k | |
| Status Affected | - | |
| OP-Code | kk10 1kkk kkkk kkkk | |
| Description | GOTO is an unconditional branch. The 12-bit immediate value is loaded into PC bits <11:0>.The upper bits of PC are loaded from PCLATH. GOTO is a two-cycle instruction. | |
| Cycle | 2 | |
| Example | LABEL1 GOTO SUB1 | B : PC =LABEL1<br>A : PC =SUB1 |

## INCX Increment "f"

| | | |
|---|---|---|
| Syntax | INCX f [,d] | |
| Operands | f : 00h ~ 1FFh | |
| Operation | (destination) ← (f) + 1 | |
| Status Affected | Z | |
| OP-Code | ff00 1010 dfff ffff | |
| Description | The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. | |
| Cycle | 1 | |
| Example | INCX CNT, 1 | B : CNT =0xFF, Z =0 |
| | | A : CNT =0x00, Z =1 |

## INCXSZ Increment "f", Skip if 0

| | | |
|---|---|---|
| Syntax | INCXSZ f [,d] | |
| Operands | f : 00h ~ 1FFh, d : 0, 1 | |
| Operation | (destination) ← (f) + 1, skip next instruction if result is 0 | |
| Status Affected | - | |
| OP-Code | ff00 1111 dfff ffff | |
| Description | The contents of register 'f' are incremented. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. If the result is 1, the next instruction is executed. If the result is 0, a NOP is executed instead, making it a 2 cycle instruction. | |
| Cycle | 1 or 2 | |
| Example | LABEL1 INCXSZ CNT, 1 | B : PC =LABEL1 |
| | GOTO LOOP | A : CNT =CNT + 1 |
| | CONTINUE | if CNT =0, PC =CONTINUE |
| | | if CNT ≠0, PC =LABEL1 + 1 |

## IORLW Inclusive OR Literal with W

| | | |
|---|---|---|
| Syntax | IORLW k | |
| Operands | k : 00h ~ FFh | |
| Operation | (W) ← (W) OR k | |
| Status Affected | Z | |
| OP-Code | 0001 1010 kkkk kkkk | |
| Description | The contents of the W register are OR’ed with the eight-bit literal 'k'. The result is placed in the W register. | |
| Cycle | 1 | |
| Example | IORLW 0x35 | B : W =0x9A |
| | | A : W =0xBF, Z =0 |

## IORWX Inclusive OR W with "f"

| | | |
|---|---|---|
| Syntax | IORWF f [,d] | |
| Operands | f : 00h ~ 1FFh, d : 0, 1 | |
| Operation | (destination) ← (W) OR k | |
| Status Affected | Z | |
| OP-Code | ff00 0100 dfff ffff | |
| Description | Inclusive OR the W register with register 'f'. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. | |
| Cycle | 1 | |
| Example | IORWX RESULT, 0 | B : RESULT =0x13, W =0x91 |
| | | A : RESULT =0x13, W =0x93, Z =0 |

| MOVX | Move f | |
|---|---|---|
| Syntax | MOVX f,d | |
| Operands | f : 00h ~ 1FFh | |
| Operation | (destination) ← (f) | |
| Status Affected | Z | |
| OP-Code | ff00 1000 dfff ffff | |
| Description | The contents of register 'f' are moved to a destination dependent upon the status of d. If d=0, destination is W register. If d =1, the destination is file register f itself. d=1 is useful to test a file register, since status flag Z is affected. | |
| Cycle | 1 | |
| Example | MOVX FSR,0 | B : FSR =0xC2, W =?<br>A : FSR =0xC2, W 0xC2 |

| MOVXW | Move "f" to W | |
|---|---|---|
| Syntax | MOVXW f | |
| Operands | f : 00h ~ 1FFh | |
| Operation | (W) ← (f) | |
| Status Affected | Z | |
| OP-Code | ff00 1000 0fff ffff | |
| Description | The contents of register 'f' are moved to W register. | |
| Cycle | 1 | |
| Example | MOVXW FSR | B : FSR =0xC2, W =?<br>A : FSR =0xC2, W 0xC2 |

| MOVLW | Move Literal to W | |
|---|---|---|
| Syntax | MOVLW k | |
| Operands | k : 00h ~ FFh | |
| Operation | (W) ← k | |
| Status Affected | - | |
| OP-Code | 0001 1001 kkkk kkkk | |
| Description | The eight-bit literal 'k' is loaded into W register. The don’t cares will assemble as 0’s. | |
| Cycle | 1 | |
| Example | MOVLW 0x5A | B : W =?<br>A : W =0x5A |

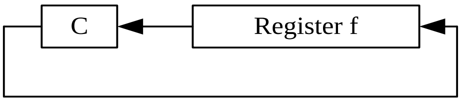
| MOVWX | Move W to "f" | |
|---|---|---|
| Syntax | MOVWX f | |
| Operands | f : 00h ~ 1FFh | |
| Operation | (f) ← (W) | |
| Status Affected | - | |
| OP-Code | ff00 0000 1fff ffff | |
| Description | Move data from W register to register 'f'. | |
| Cycle | 1 | |
| Example | MOVWX REG1 | B : REG1 =0xFF, W =0x4F<br>A : REG1 =0x4F, W =0x4F |

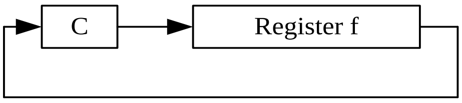
| **NOP** | **No Operation** | |
|---|---|---|
| Syntax | NOP | |
| Operands | - | |
| Operation | No Operation | |
| Status Affected | - | |
| OP-Code | 0000 0000 0000 0000 | |
| Description | No Operation | |
| Cycle | 1 | |
| Example | NOP | - |

| **RET** | **Return from Subroutine** | |
|---|---|---|
| Syntax | RET | |
| Operands | - | |
| Operation | PC ← TOS | |
| Status Affected | - | |
| OP-Code | 0000 0000 0100 0000 | |
| Description | Return from subroutine. The stack is POPed and the top of the stack (TOS) is loaded into the program counter. This is a two-cycle instruction. | |
| Cycle | 2 | |
| Example | RET | A : PC =TOS |

| **RETI** | **Return from Interrupt** | |
|---|---|---|
| Syntax | RETI | |
| Operands | - | |
| Operation | PC ← TOS, GIE ← 1 | |
| Status Affected | - | |
| OP-Code | 0000 0000 0110 0000 | |
| Description | Return from Interrupt. Stack is POPed and Top-of-Stack (TOS) is loaded in to the PC. Interrupts are enabled. This is a two-cycle instruction. | |
| Cycle | 2 | |
| Example | RETI | A : PC =TOS, GIE =1 |

| **RETLW** | **Return with Literal in W** | |
|---|---|---|
| Syntax | RETLW k | |
| Operands | k : 00h ~ FFh | |
| Operation | PC ← TOS, (W) ← k | |
| Status Affected | - | |
| OP-Code | 0001 1000 kkkk kkkk | |
| Description | The W register is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction. | |
| Cycle | 2 | |
| Example | CALL TABLE<br>:<br>TABLE ADDWX PCL, 1<br>RETLW k1<br>RETLW k2<br>:<br>RETLW kn | B : W =0x07<br>A : W =value of k8 |

| RLX | Rotate Left "f" through Carry | |
|---|---|---|
| Syntax | RLX f [,d] | |
| Operands | f : 00h ~ 1FFh, d : 0, 1 | |
| Operation | <br> | |
| Status Affected | C | |
| OP-Code | ff00 1101 dfff ffff | |
| Description | The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is stored back in register 'f'. | |
| Cycle | 1 | |
| Example | RLX REG1, 0 | B : REG1 =1110 0110, C =0<br>A : REG1 =1110 0110<br>W =1100 1100, C =1 |

| RRX | Rotate Right "f" through Carry | |
|---|---|---|
| Syntax | RRX f [,d] | |
| Operands | f : 00h ~ 1FFh, d : 0, 1 | |
| Operation | <br> | |
| Status Affected | C | |
| OP-Code | ff00 1100 dfff ffff | |
| Description | The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0, the result is placed in the W register. If 'd' is 1, the result is placed back in register 'f'. | |
| Cycle | 1 | |
| Example | RRX REG1, 0 | B : REG1 =1110 0110, C =0<br>A : REG1 =1110 0110<br>W =0111 0011, C =0 |

| SLEEP | Go into Power-down mode, Clock oscillation stops | |
|---|---|---|
| Syntax | SLEEP | |
| Operands | - | |
| Operation | - | |
| Status Affected | TO, PD | |
| OP-Code | 001 1110 0000 0011 | |
| Description | Go into Power-down mode with the oscillator stops. | |
| Cycle | 1 | |
| Example | SLEEP | - |

| **SUBLW** | **Subtract W from Literal** | |
|---|---|---|
| Syntax | SUBLW k | |
| Operands | k : 00h ~ FFh | |
| Operation | (W) ← k – (W) | |
| Status Affected | C, DC, Z | |
| OP-Code | 0001 1111 kkkk kkkk | |
| Description | The W register is subtracted (2’s complement method) from the eight-bit literal “k”. The result is placed in the W register. | |
| Cycle | 1 | |
| Example | SUBLW 0x25 | B : W =0x15<br>A : W =0x10 |

| **SUBWX** | **Subtract W from "f"** | |
|---|---|---|
| Syntax | SUBWX f [,d] | |
| Operands | f : 00h ~ 1FFh, d : 0, 1 | |
| Operation | (destination) ← (f) – (W) | |
| Status Affected | C, DC, Z | |
| OP-Code | ff00 0010 dfff ffff | |
| Description | Subtract (2’s complement method) W register from register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'. | |
| Cycle | 1 | |
| Example | SUBWX REG1, 1 | B : REG1 =0x03, W =0x02, C =?, Z =?<br>A : REG1 =0x01, W =0x02, C =1, Z =0 |
| | SUBWX REG1, 1 | B : REG1 =0x02, W =0x02, C =?, Z =?<br>A : REG1 =0x00, W =0x02, C =1, Z =1 |
| | SUBWX REG1, 1 | B : REG1 =0x01, W =0x02, C =?, Z =?<br>A : REG1 =0xFF, W =0x02, C =0, Z =0 |

| **SWAPX** | **Swap Nibbles in "f"** | |
|---|---|---|
| Syntax | SWAPX f [,d] | |
| Operands | f : 00h ~ 1FFh, d : 0, 1 | |
| Operation | (destination,7~4) ← (f.3~0), (destination.3~0) ← (f.7~4) | |
| Status Affected | - | |
| OP-Code | ff00 1110 dfff ffff | |
| Description | The upper and lower nibbles of register 'f' are exchanged. If 'd' is 0, the result is placed in W register. If 'd' is 1, the result is placed in register 'f'. | |
| Cycle | 1 | |
| Example | SWAPX REG, 0 | B : REG1 =0xA5<br>A : REG1 =0xA5, W =0x5A |

## TABRH Return DPTR high byte to W

| | |
|---|---|
| Syntax | TABRH |
| Operands | - |
| Operation | (W) ← ROM[DPTR] high byte content, Where DPTR = {DPH[max:8], FSR[7:0]} |
| Status Affected | - |
| OP-Code | 0000 0000 0101 1000 |
| Description | The W register is loaded with high byte of ROM[DPTR]. This is a two-cycle instruction. |
| Cycle | 2 |

Example

```
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:
DT       0x3789, 0x2277        ;ROM data 16 bits
```

## TABRL Return DPTR low byte to W

| | |
|---|---|
| Syntax | TABRL |
| Operands | - |
| Operation | (W) ← ROM[DPTR] low byte content, Where DPTR = {DPH[max:8], FSR[7:0]} |
| Status Affected | - |
| OP-Code | 0000 0000 0101 0000 |
| Description | The W register is loaded with low byte of ROM[DPTR]. This is a two-cycle instruction. |
| Cycle | 2 |

Example

```
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:
DT       0x3789, 0x2277        ;ROM data 16 bits
```

| **TSTX** | **Test if "f" is zero** | |
|---|---|---|
| Syntax | TSTX f | |
| Operands | f : 00h ~ 1FFh | |
| Operation | Set Z flag if (f) is 0 | |
| Status Affected | Z | |
| OP-Code | ff00 1000 1fff ffff | |
| Description | If the content of register 'f' is 0, Zero flag is set to 1. | |
| Cycle | 1 | |
| Example | TSTX REG1 | B : REG1 =0, Z =?<br>A : REG1 =0, Z =1 |

| **XORLW** | **Exclusive OR Literal with W** | |
|---|---|---|
| Syntax | XORLW k | |
| Operands | k : 00h ~ FFh | |
| Operation | (W) ← (W) XOR k | |
| Status Affected | Z | |
| OP-Code | 0001 1101 kkkk kkkk | |
| Description | The contents of the W register are XOR’ed with the eight-bit literal 'k'. The result is placed in the W register. | |
| Cycle | 1 | |
| Example | XORLW 0xAF | B : W =0xB5<br>A : W =0x1A |

| **XORWX** | **Exclusive OR W with "f"** | |
|---|---|---|
| Syntax | XORWX f [,d] | |
| Operands | f : 00h ~ 1FFh, d : 0, 1 | |
| Operation | (destination) ← (W) XOR (f) | |
| Status Affected | Z | |
| OP-Code | ff00 0110 dfff ffff | |
| Description | Exclusive OR the contents of the W register with register 'f'. If 'd' is 0, the result is stored in the W register. If 'd' is 1, the result is stored back in register 'f'. | |
| Cycle | 1 | |
| Example | XORWX REG, 1 | B : REG =0xAF, W =0xB5<br>A : REG =0x1A, W =0xB5 |

# ELECTRICAL CHARACTERISTICS

## 1. Absolute Maximum Ratings ($T_A$ =25℃)

| Parameter | Rating | Unit |
|---|---|---|
| Supply voltage | $V_{SS}$ -0.3 to $V_{SS}$ +5.5 | V |
| Input voltage | $V_{SS}$ -0.3 to $V_{CC}$ +0.3 | |
| Output voltage | $V_{SS}$ -0.3 to $V_{CC}$ +0.3 | |
| Output current high per 1 PIN | -25 | mA |
| Output current high per all PIN | -80 | |
| Output current low per 1 PIN | +30 | |
| Output current low per all PIN | +150 | |
| Maximum operating voltage | 5.5 | V |
| Operating temperature | -40 to +85 | °C |
| Storage temperature | -65 to +150 | |

## 2. DC Characteristics ($T_A$ =25℃, $V_{DD}$ =5.0V, unless otherwise specified)

| Parameter | Sym | Conditions | | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| Operating Voltage | $V_{cc}$ | Fsys=16Mhz | | 2.8 | | 5.5 | |
| | | Fsys= 8Mhz | | LVR | | 5.5 | |
| Input High Voltage | $V_{IH}$ | All Input | $V_{CC}$ = 3~5V | 0.6Vcc | – | Vcc | V |
| Input Low Voltage | $V_{IL}$ | All Input | $V_{CC}$ = 3~5V | Vss | – | 0.2Vcc | V |
| Output High Current | $I_{OH}$ | All Output | $V_{CC}$ = 5V,$V_{OH}$ = 4.5V | 5 | 12 | – | mA |
| | | | $V_{CC}$ = 3V,$V_{OH}$ = 2.7V | 2.5 | 4.5 | – | |
| Output Low Current | $I_{OL}$ | All Output | $V_{CC}$ = 5V,$V_{OL}$ = 0.5V | 20 | 39 | – | mA |
| | | | $V_{CC}$ = 3V,$V_{OL}$ = 0.3V | 8 | 17 | – | |
| Input Leakage Current (pin high) | $I_{ILH}$ | All Input | $V_{IN}$ = $V_{CC}$ | – | – | 1 | uA |
| Input Leakage Current (pin low) | $I_{ILL}$ | All Input | $V_{IN}$ = 0V | – | – | –1 | uA |
| Power Supply Current (No Load) | $I_{CC}$ | FAST mode<br>FIRC 16 MHz | $V_{CC}$ = 5V | – | 4.5 | – | mA |
| | | FAST mode<br>FIRC 8 MHz | $V_{CC}$ = 5V | | 3 | | |
| | | FAST mode<br>FIRC 4 MHz | $V_{CC}$ = 5V | – | 2.3 | – | |
| | | FAST mode<br>FIRC 1 MHz | $V_{CC}$ = 5V | – | 1.8 | – | |
| | | | $V_{CC}$ = 3V | – | 1.2 | – | |
| | | SLOW mode<br>SIRC 70KHz<br>FIRC STOP | $V_{CC}$ = 5.0V | – | 850 | – | uA |
| | | | $V_{CC}$ = 3.0V | – | 700 | – | |
| | | STOP mode<br>LVRSAV = 1 | $V_{CC}$ = 5.0V | – | 0.1 | – | uA |
| | | | $V_{CC}$ = 3.0V | – | 0.1 | – | |

| Parameter | Sym | Conditions | | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|---|
| Power Supply Current (No Load) | $I_{CC}$ | IDLE mode<br>SIRC 70 KHz<br>LVRSAV= 1 | $V_{CC}$ = 5.0V | – | 6 | – | uA |
| | | | $V_{CC}$ = 3.0V | – | 2 | – | |
| Pull-up Resistor | $R_{UP}$ | $V_{IN}$ = 0 V<br>Ports A/B/C/D | $V_{CC}$ = 5.0V | – | 39 | – | KΩ |
| | | | $V_{CC}$ = 3.0V | – | 70 | – | |

## 3. Clock Timing ($T_A$ = -40°C to +85°C)

| Parameter | Condition | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| FIRC Frequency (*) | -40℃ ~ 85℃, $V_{CC}$ = 3.0 ~ 5.0V | -6.5% | 16 | +1.5% | MHz |
| | -40℃ ~ 85℃, $V_{CC}$ = 4.0 V | -5% | 16 | +1.5% | |
| | 0℃ ~ 70℃, $V_{CC}$ = 4.0 V | -2% | 16 | +1.5% | |
| | 25℃, $V_{CC}$ = 3.0 ~ 5.0 V | -1.0% | 16 | +1.2% | |
| | 25℃, $V_{CC}$ =4.0 V | -0.5% | 16 | +0.5% | |

(*) FIRC frequency can be divided by 1/2/4/16.

## 4. Reset Timing Characteristics ($T_A$ = -40°C to +85°C)

| Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| RESET Input Low width | Input $V_{CC}$ = 5 V ±10 % | 30 | – | – | µs |
| WDT time | $V_{CC}$ = 3 V, WDTPSC = 11 | – | 2090 | – | ms |
| | $V_{CC}$ = 5 V, WDTPSC = 11 | | 1870 | | |
| WKT time | $V_{CC}$ = 3 V, WKTPSC = 11 | – | 133 | – | ms |
| | $V_{CC}$ = 5 V, WKTPSC = 11 | | 118 | | |
| CPU start up time | $V_{CC}$ = 5 V | – | 27 | – | ms |

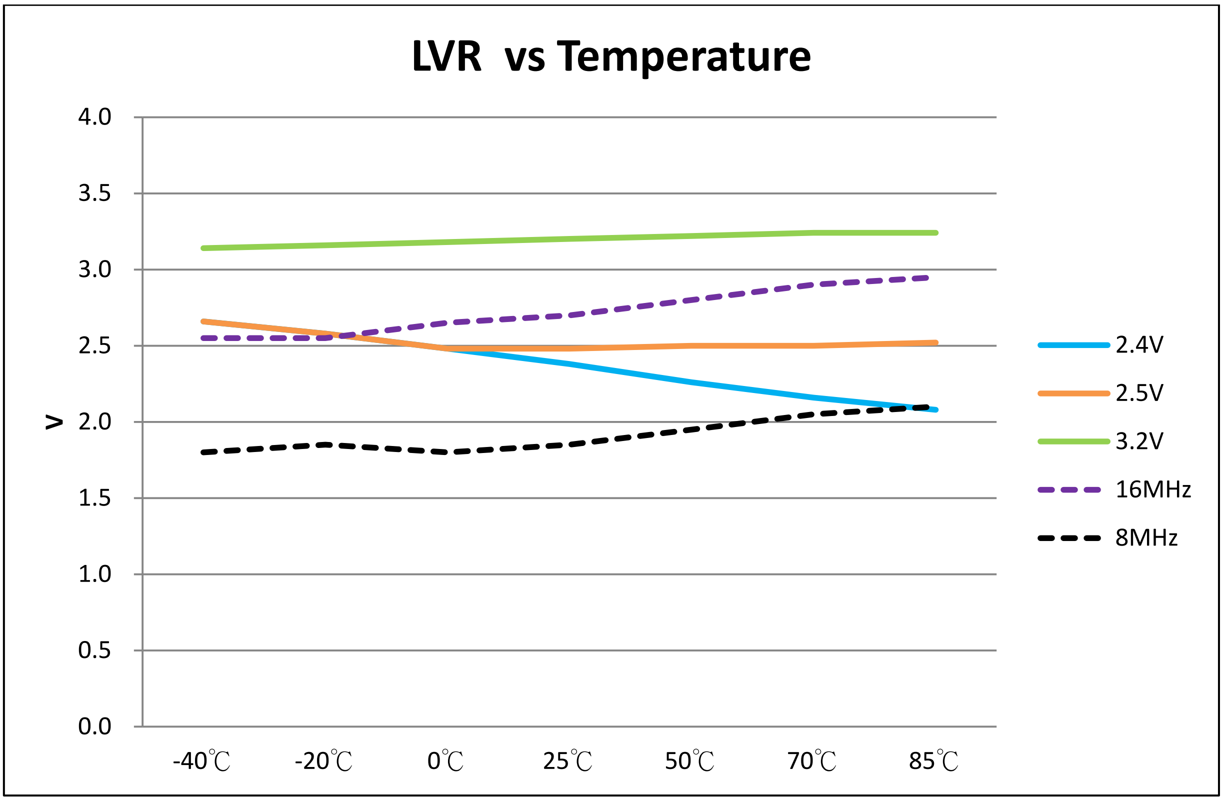
## 5. LVR Circuit Characteristics ($T_A$ = 25℃)

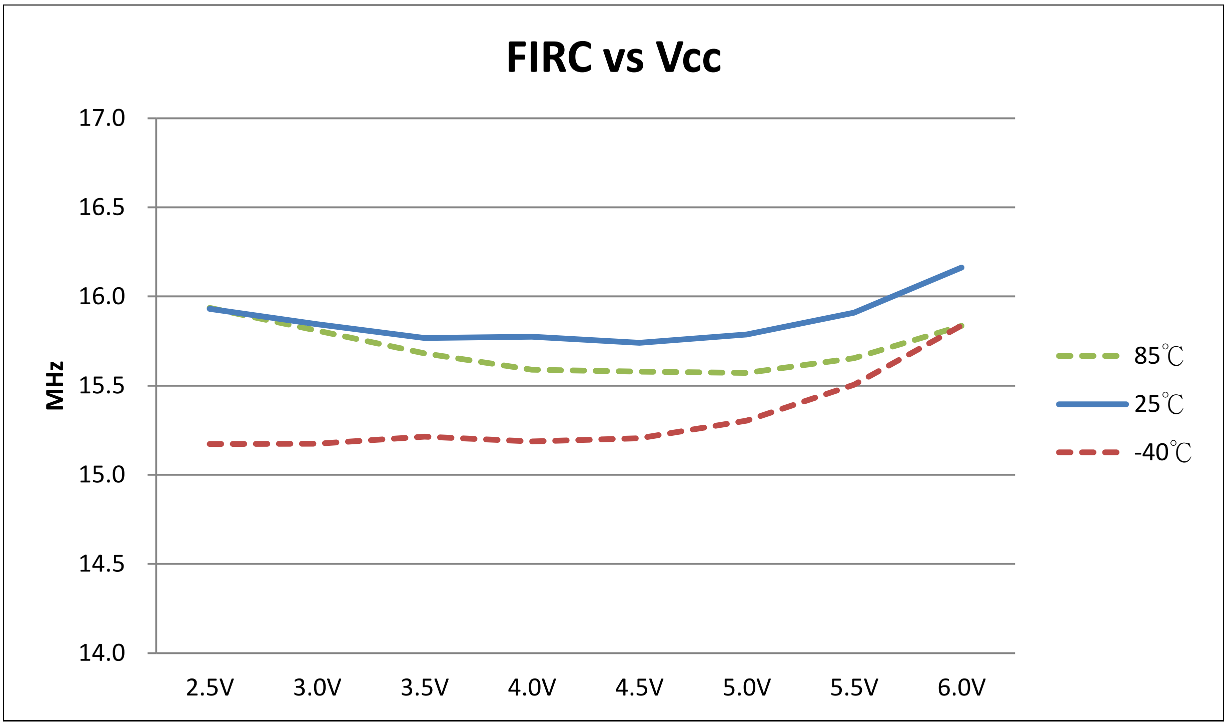
| Parameter | Symbol | Min | Typ | Max | Unit |
|---|---|---|---|---|---|
| LVR Reference Voltage | $LVR_{th}$ | – | 2.4 | – | V |
| | | – | 2.5 | – | |
| | | – | 3.2 | – | |
| | | – | 3.7 | – | |
| LVR Hysteresis Voltage | $V_{HYST}$ | – | ±0.1 | – | V |
| Low Voltage Detection time | $t_{LVR}$ | 100 | – | – | µs |

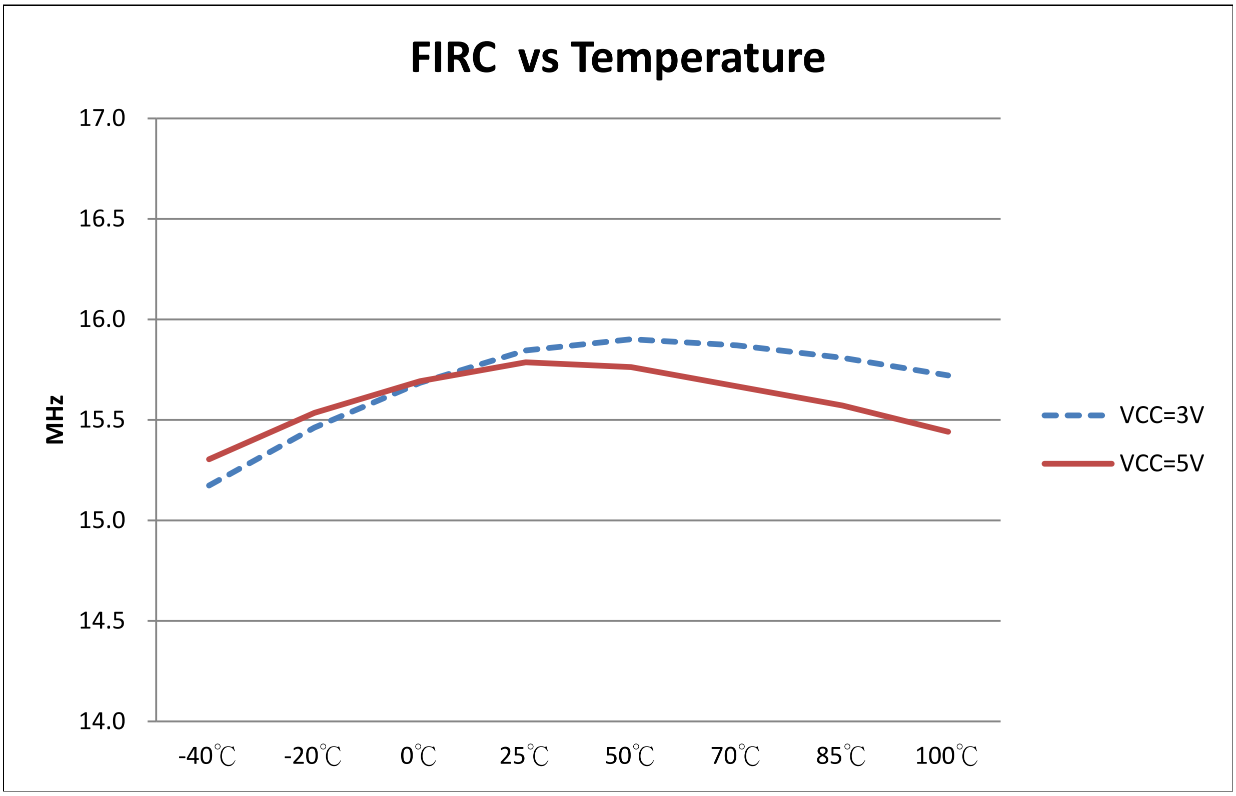
## 6. ADC Electrical Characteristics (T$_A$ = 25℃, VCC = 3.0V to 5.5V, VSS = 0V)

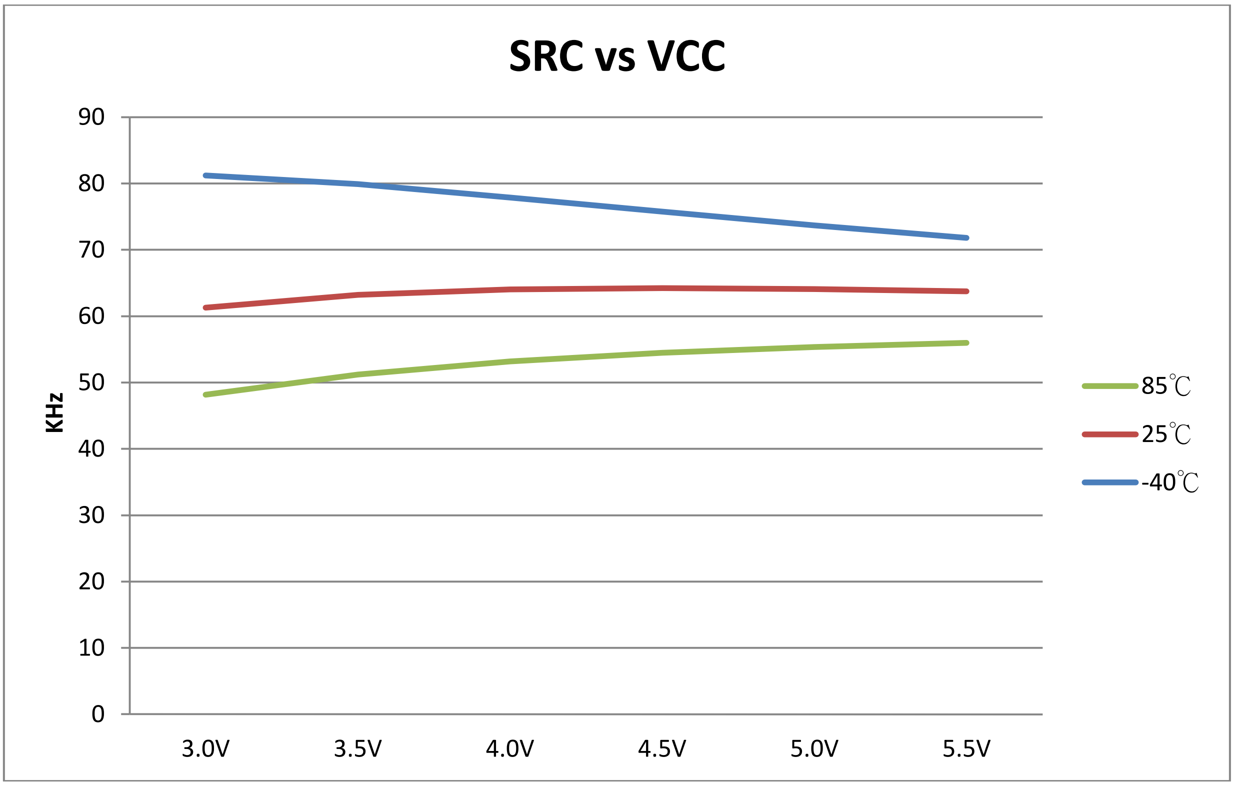
| Parameter | Conditions | Min | Typ | Max | Units |
|---|---|---|---|---|---|
| Total Accuracy | $V_{CC}$ = 5V, $V_{SS}$ = 0V, $f_{ADC}$ = 1 MHz | – | ±2.5 | ±13 | LSB |
| Integral Non-Linearity | | – | ±3.2 | ±15 | |
| Differential Non-linearity | | – | ±1 | ±4 | |
| Max Input Clock freq. ($f_{ADC}$) | Source impedance (Rs<10K omh) | – | – | 2 | MHz |
| | Source impedance (Rs<20K omh) | – | – | 1 | |
| | Source impedance (Rs<50K omh) | – | – | 0.5 | |
| | Source is VBG (ADCHS=1100) | – | – | 2 | |
| Conversion Time | $f_{ADC}$ = 1 MHz | – | 50 | – | μs |
| Input Voltage | – | $V_{SS}$ | – | **Vcc** | V |

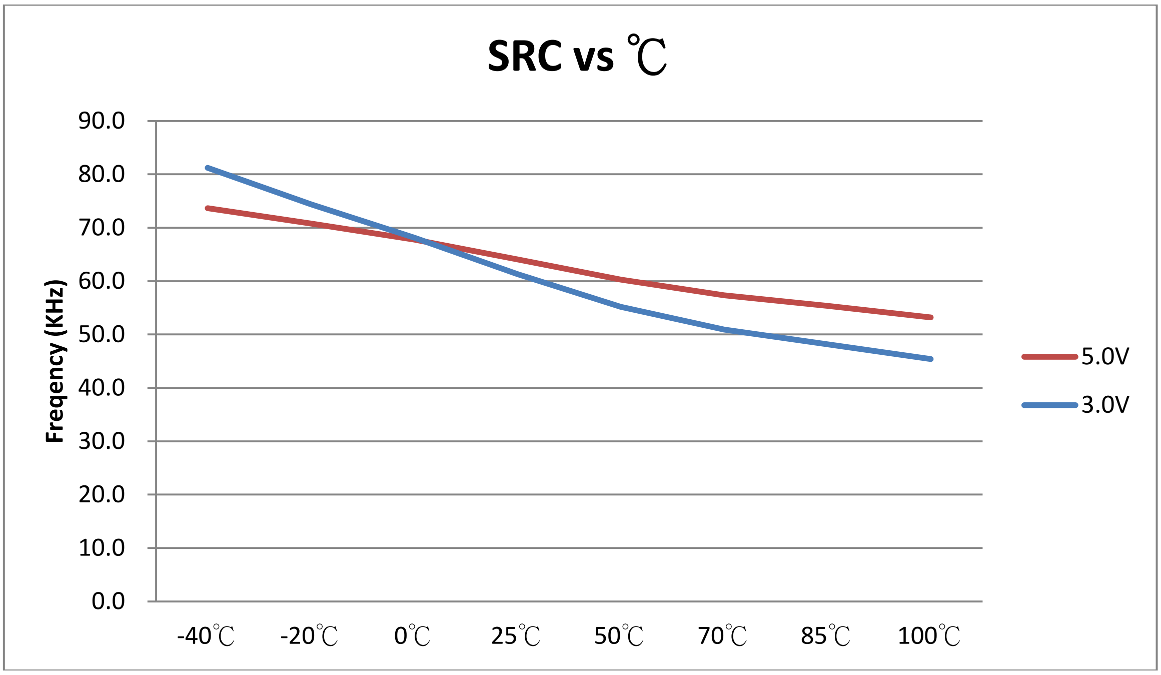
## 7. Characteristic Graphs






FIRC vs Temperature
MHz
17.0
16.5
16.0
15.5
15.0
14.5
14.0
-40℃
-20℃
0℃
25℃
50℃
70℃
85℃
100℃
VCC=3V
VCC=5V


SRC vs VCC
KHz
90
80
70
60
50
40
30
20
10
0
3.0V
3.5V
4.0V
4.5V
5.0V
5.5V
85℃
25℃
-40℃


SRC vs ℃
Freqency (KHz)
90.0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.0
-40℃
-20℃
0℃
25℃
50℃
70℃
85℃
100℃
5.0V
3.0V

# PACKAGING INFORMATION

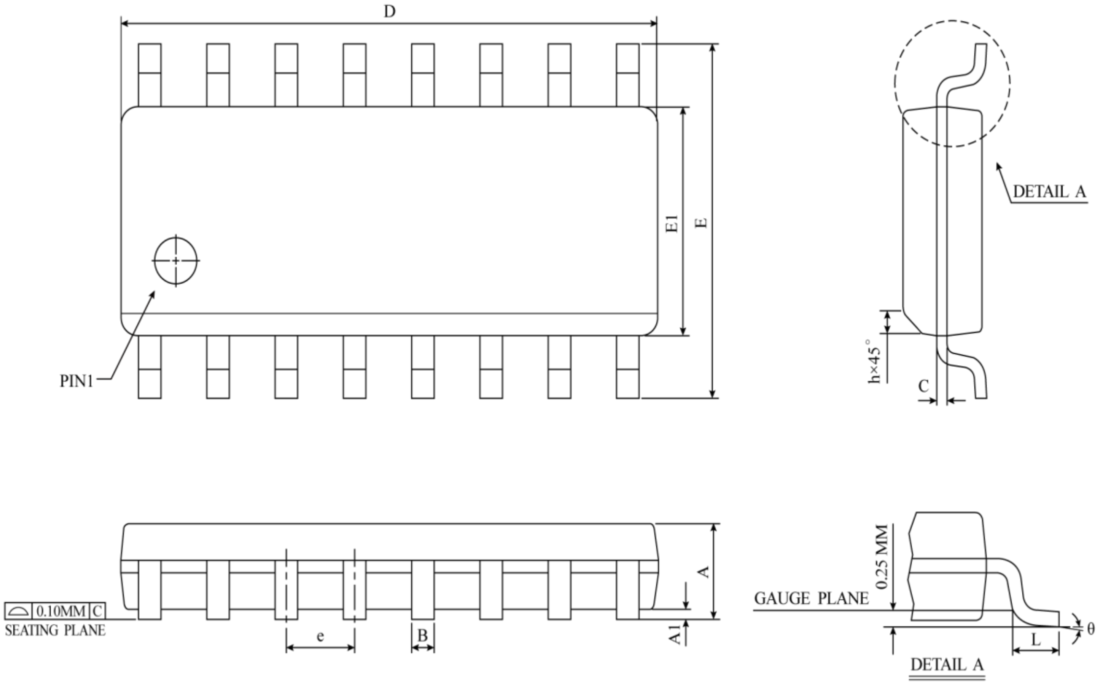
Please note that the package information provided is for reference only. Since this information is frequently updated, users can contact Sales to consult the latest package information and stocks.

The ordering information:

| Ordering number | Package |
|---|---|
| TM56F5412/12B-MTP | Wafer/Dice blank chip |
| TM56F5412/12B-COD | Wafer/Dice with code |
| TM56F5412/12B-MTP-22 | SOP 24-pin (300 mil) |

| Ordering number | Package |
|---|---|
| TM56F5416/16B-MTP | Wafer/Dice blank chip |
| TM56F5416/16B -COD | Wafer/Dice with code |
| TM56F5416/16B -MTP-23 | SOP 28-pin (300 mil) |
| TM56F5416/16B -MTP-16 | SOP 20-pin (300 mil) |
| TM56F5416/16B -MTP-21 | SOP 16-pin (150 mil) |
| TM56F5416/16B -MTP-D1 | QFN 20-pin (3*3*0.75-0.4mm)(L=0.25mm) |
| TM56F5416/16B -COD-D1 | |
| TM56F5416/16B -MTP-53 | MSOP 10-pin (118 mil) |
| TM56F5416/16B -COD-53 | |

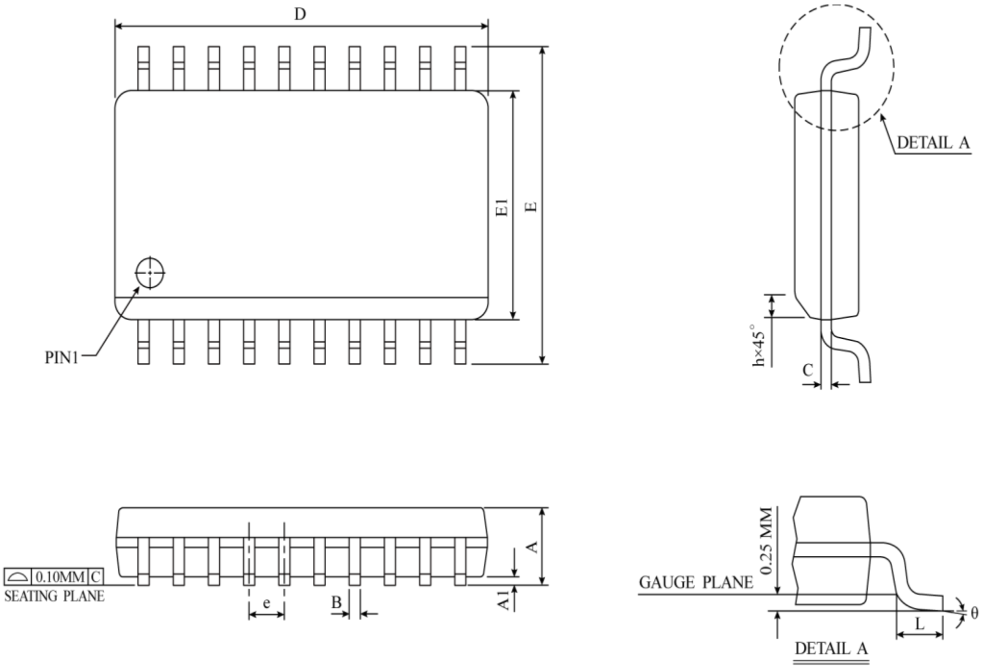
## ✧ SOP-16 ( 150mil ) Package Dimension



| 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.

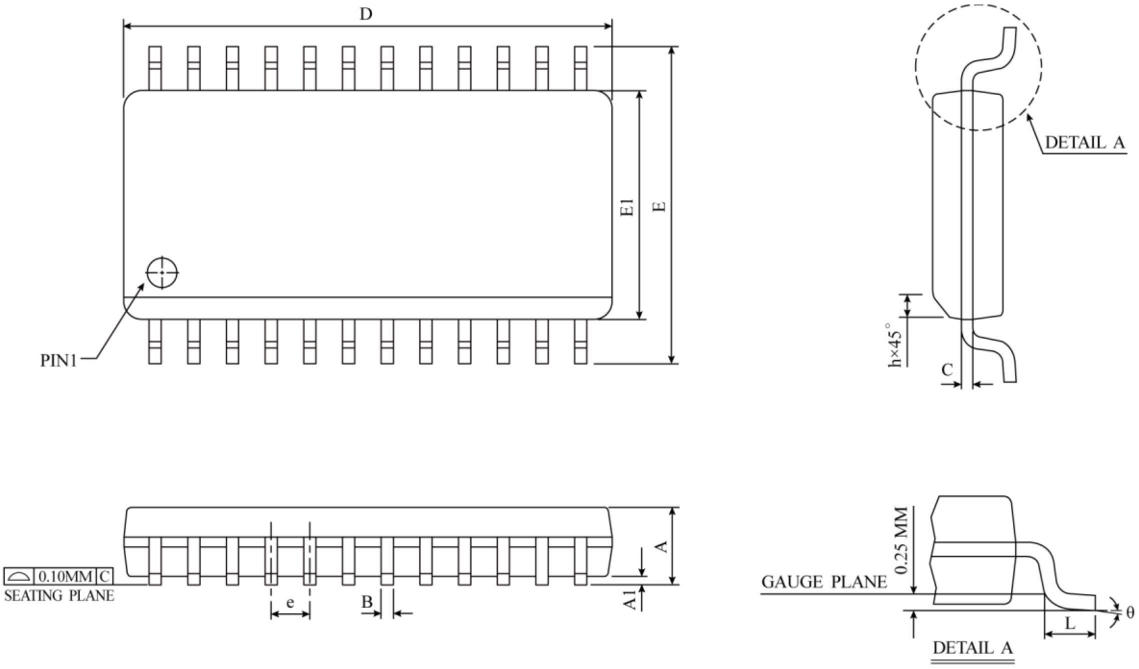
### ✧ SOP-20 ( 300mil ) Package Dimension



| SYMBOL | DIMENSION IN MM | | | DIMENSION IN INCH | | |
|---|---|---|---|---|---|---|
| | MIN | NOM | MAX | MIN | NOM | MAX |
| A | 2.35 | 2.50 | 2.65 | 0.0926 | 0.0985 | 0.1043 |
| A1 | 0.10 | 0.20 | 0.30 | 0.0040 | 0.0079 | 0.0118 |
| B | 0.33 | 0.42 | 0.51 | 0.0130 | 0.0165 | 0.0200 |
| C | 0.23 | 0.28 | 0.32 | 0.0091 | 0.0108 | 0.0125 |
| D | 12.60 | 12.80 | 13.00 | 0.4961 | 0.5040 | 0.5118 |
| E | 10.00 | 10.33 | 10.65 | 0.3940 | 0.4425 | 0.4910 |
| E1 | 7.40 | 7.50 | 7.60 | 0.2914 | 0.2953 | 0.2992 |
| e | 1.27 BSC | | | 0.050 BSC | | |
| h | 0.25 | 0.50 | 0.75 | 0.0100 | 0.0195 | 0.0290 |
| L | 0.40 | 0.84 | 1.27 | 0.0160 | 0.0330 | 0.0500 |
| θ | 0° | 4° | 8° | 0° | 4° | 8° |
| JEDEC | MS-013 (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.

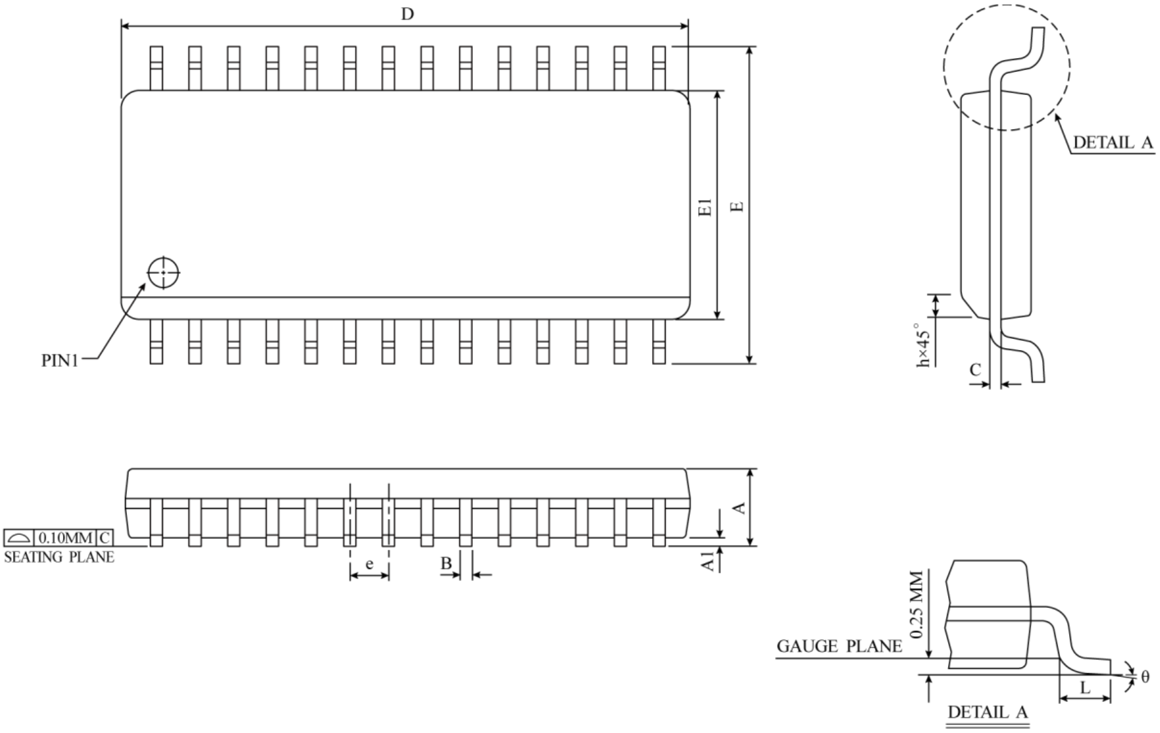
### ✧ SOP-24 ( 300mil ) Package Dimension



| SYMBOL | DIMENSION IN MM | | | DIMENSION IN INCH | | |
|---|---|---|---|---|---|---|
| | MIN | NOM | MAX | MIN | NOM | MAX |
| A | 2.35 | 2.50 | 2.65 | 0.0926 | 0.0985 | 0.1043 |
| A1 | 0.10 | 0.20 | 0.30 | 0.0040 | 0.0079 | 0.0118 |
| B | 0.33 | 0.42 | 0.51 | 0.0130 | 0.0165 | 0.0200 |
| C | 0.23 | 0.28 | 0.32 | 0.0091 | 0.0108 | 0.0125 |
| D | 15.20 | 15.40 | 15.60 | 0.5985 | 0.6063 | 0.6141 |
| E | 10.00 | 10.33 | 10.65 | 0.3940 | 0.4425 | 0.4910 |
| E1 | 7.40 | 7.50 | 7.60 | 0.2914 | 0.2953 | 0.2992 |
| e | 1.27 BSC | | | 0.050 BSC | | |
| h | 0.25 | 0.50 | 0.75 | 0.0100 | 0.0195 | 0.0290 |
| L | 0.40 | 0.84 | 1.27 | 0.0160 | 0.0330 | 0.0500 |
| θ | 0° | 4° | 8° | 0° | 4° | 8° |
| JEDEC | MS-013 (AD) | | | | | |

⚠ * 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.

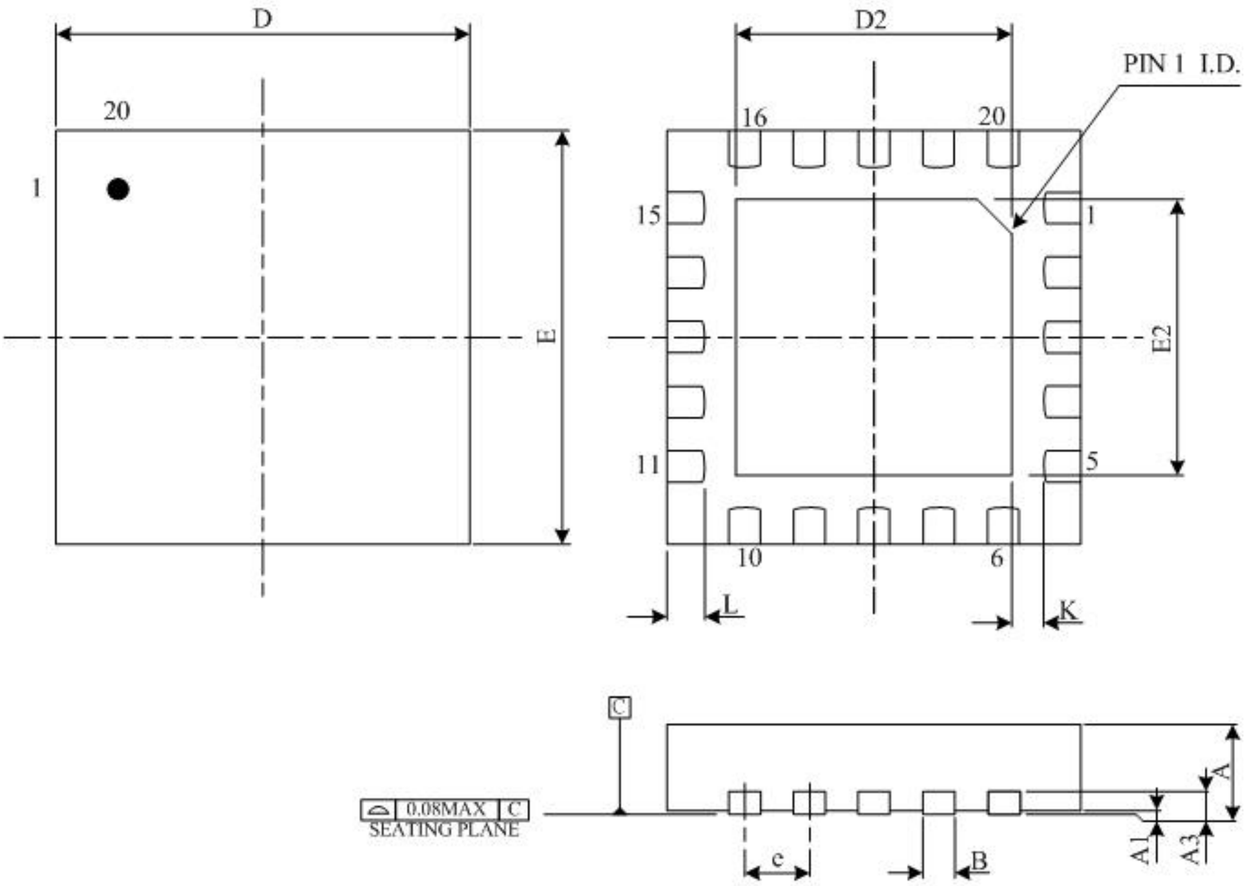
✧ SOP-28 ( 300mil ) Package Dimension



| SYMBOL | DIMENSION IN MM | | | DIMENSION IN INCH | | |
|---|---|---|---|---|---|---|
| | MIN | NOM | MAX | MIN | NOM | MAX |
| A | 2.35 | 2.50 | 2.65 | 0.0926 | 0.0985 | 0.1043 |
| A1 | 0.10 | 0.20 | 0.30 | 0.0040 | 0.0079 | 0.0118 |
| B | 0.33 | 0.42 | 0.51 | 0.0130 | 0.0165 | 0.0200 |
| C | 0.23 | 0.28 | 0.32 | 0.0091 | 0.0108 | 0.0125 |
| D | 17.70 | 17.90 | 18.10 | 0.6969 | 0.7047 | 0.7125 |
| E | 10.00 | 10.33 | 10.65 | 0.3940 | 0.4425 | 0.4910 |
| E1 | 7.40 | 7.50 | 7.60 | 0.2914 | 0.2953 | 0.2992 |
| e | 1.27 BSC | | | 0.050 BSC | | |
| h | 0.25 | 0.50 | 0.75 | 0.0100 | 0.0195 | 0.0290 |
| L | 0.40 | 0.84 | 1.27 | 0.0160 | 0.0330 | 0.0500 |
| θ | 0° | 4° | 8° | 0° | 4° | 8° |
| JEDEC | MS-013 (AE) | | | | | |

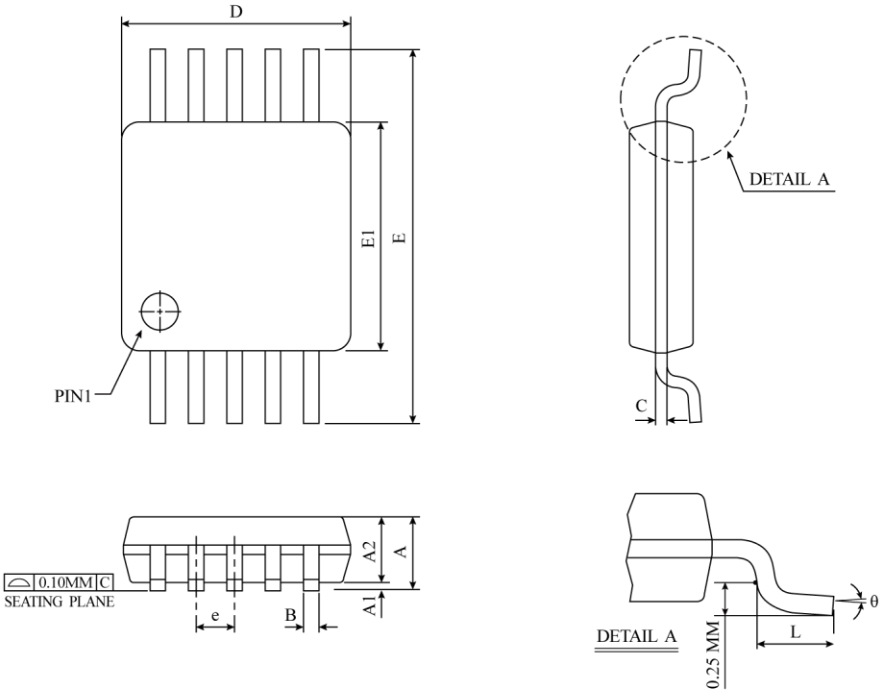
⚠ * NOTES : DIMENSION " D " DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS.
MOLD FLASH, PROTRUSIONS AND GATE BURRS SHALL
NOT EXCEED 0.15 MM ( 0.006 INCH ) PER SIDE.

✧ QFN-20 ( 3x3x0.75-0.4mm ) (L=0.25mm) Package Dimension



| SYMBOL | DIMENSION IN MM | | | DIMENSION IN INCH | | |
|---|---|---|---|---|---|---|
| | MIN | NOM | MAX | MIN | NOM | MAX |
| A | 0.70 | 0.75 | 0.80 | 0.028 | 0.030 | 0.031 |
| A1 | 0.00 | 0.02 | 0.05 | 0.00 | 0.001 | 0.002 |
| A3 | 0.203 REF | | | 0.008 REF | | |
| B | 0.15 | 0.20 | 0.25 | 0.006 | 0.008 | 0.010 |
| D | 3 BSC | | | 0.118 BSC | | |
| E | 3 BSC | | | 0.118 BSC | | |
| D2 | 1.80 | 1.90 | 2.00 | 0.071 | 0.075 | 0.079 |
| E2 | 1.80 | 1.90 | 2.00 | 0.071 | 0.075 | 0.079 |
| e | 0.40 BSC | | | 0.016 BSC | | |
| L | 0.15 | 0.25 | 0.35 | 0.006 | 0.010 | 0.014 |
| K | 0.30 REF | | | 0.012 REF | | |
| JEDEC | MO-220 | | | | | |

✧ MSOP-10 (118mil) Package Dimension



| SYMBOL | DIMENSION IN MM | | | DIMENSION IN INCH | | |
|---|---|---|---|---|---|---|
| | MIN | NOM | MAX | MIN | NOM | MAX |
| A | 0.81 | 0.96 | 1.10 | 0.032 | 0.038 | 0.043 |
| A1 | 0.05 | 0.10 | 0.15 | 0.002 | 0.004 | 0.006 |
| A2 | 0.75 | 0.85 | 0.95 | 0.030 | 0.034 | 0.037 |
| B | 0.17 | 0.22 | 0.27 | 0.007 | 0.009 | 0.011 |
| C | 0.13 | 0.18 | 0.23 | 0.005 | 0.007 | 0.009 |
| D | 2.90 | 3.00 | 3.10 | 0.114 | 0.118 | 0.122 |
| E | 4.75 | 4.90 | 5.05 | 0.187 | 0.193 | 0.199 |
| E1 | 2.90 | 3.00 | 3.10 | 0.114 | 0.118 | 0.122 |
| e | 0.50 BSC | | | 0.020 BSC | | |
| L | 0.40 | 0.55 | 0.70 | 0.016 | 0.022 | 0.028 |
| θ | 0° | 3° | 6° | 0° | 3° | 6° |
| JEDEC | | | | | | |

⚠ * NOTES：DIMENSION 〝D〞 DOES NOT INCLUDE MOLD PROTRUSIONS OR GATE BURRS.
MOLD PROTRUSIONS AND GATE BURRS SHALL NOT EXCEED 0.12 MM ( 0.005 INCH ) PER SIDE.
DIMENSION 〝E1〞 DOES NOT INCLUDE MOLD PROTRUSIONS
MOLD PROTRUSIONS SHALL NOT EXCEED 0.25 MM ( 0.010 INCH ) PER SIDE.