



十速

TM56F8225

Battery Car and Electrical

Tools Charger ASSP IC

DATA SHEET

Rev 0.95

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	June, 2020	New Release
0.91	Aug, 2020	Revised Describe and Update Parameter 1. Fig. BCM Block Diagram (P.67) 2. Memory map HWAUTO describe (P. 75) 3. VR parameter (P.93) 4. BCM Electrical Characteristics (P. 94)
0.92	Sep, 2020	1. Update Electrical Characteristics (P.92-P.94) 2. Revised Describe - SFR description (P.58)
0.93	Dec, 2020	1. Revised ADVREFS default value (P.74) 2. Add VR v.s. Temp. curve (P.96) 3. Suggest EEPTE setting
0.94	Sep, 2021	1. Fixed typo in the description of indirect addressing(P.17) 2. Fixed typo in example code (P.19) 3. Revised description and fixed typos in example code (P.21) 4. Revised description for INT1IF(P.34) 5. Add SFR OPTION table for the description of interrupt(P.35) 6. Revised description of output mode example(P.37) 7. Revised description to clear watchdog timer(P.41) 8. Fixed typos in the table of SFR OPTION(P.42) 9. Revised figure of timer0 worked in counter mode(P.46) 10. Fixed typo in the description of PWM1AOE1(P.75) 11. Fixed typos in the description of CALL instruction(P.83) 12. Fixed typos in the description of GOTO instruction(P.84) 13. Fixed typos in the description of MOVX and MOVXW instructions(P.86) 14. Fixed description of SUBLW instruction(P.89) 15. Fixed typos in the explanation of example code(P.55)
0.95	Feb, 2022	1. Add the description of “PWM0 clock is enabled and PWM0 is not hold after reset”.(P.53) 2. Add the description of “PWM1 clock is enabled and PWM1 is not hold after reset”. (P.59)

CONTENTS

AMENDMENT HISTORY	2
CONTENTS.....	3
FEATURES	5
SYSTEM BLOCK DIAGRAM.....	8
PIN ASSIGNMENT DIAGRAM.....	9
PIN DESCRIPTIONS	10
PIN SUMMARY.....	11
FUNCTION DESCRIPTION.....	12
1 CPU Core.....	12
1.1 Program ROM (PROM)	12
1.1.1 Reset Vector (000H).....	12
1.1.2 Interrupt Vector (004H)	12
1.2 System Configuration Register (SYSCFG).....	13
1.3 Data ROM (EEPROM)	14
1.4 RAM Addressing Mode	17
1.5 Programming Counter (PC) and Stack.....	20
1.5.1 ALU and Working (W) Register	22
1.5.2 STATUS Register (03H/83H/103H/183H)	22
2 Reset	24
2.1 Power on Reset.....	24
2.2 Low Voltage Reset	24
2.3 External Pin Reset	25
2.4 Watchdog Timer Reset.....	25
3 Clock Circuitry and Operation Mode	26
3.1 System Clock.....	26
3.2 Dual System Clock Modes Transition	28
3.3 System Clock Oscillator.....	31
4 Interrupt	32
5 I/O Port	36
5.1 PA0-PA7	36
6 Peripheral Functional Block	40
6.1 Watchdog (WDT) /Wakeup (WKT) Timer	40
6.2 Timer0	43
6.3 Timer1	48
6.4 T2:15-bit Timer	51
6.5 PWM0: (8+2) bits PWM.....	53
6.6 PWM1A / PWM1B / PWM1C: 16 bits PWMs.....	59
6.7 Analog-to-Digital Converter	64
6.8 Battery Charge Module (BCM).....	67
6.9 Cyclic Redundancy Check (CRC).....	70

MEMORY MAP.....	71
INSTRUCTION SET.....	79
ELECTRICAL CHARACTERISTICS	92
1. Absolute Maximum Ratings	92
2. DC Characteristics (TA =25°C, V _{CC} =5.0V, unless otherwise specified).....	92
3. Clock Timing	93
4. Reset Timing Characteristics	93
5. LVR Circuit Characteristics (TA = 25°C)	93
6. ADC Electrical Characteristics	94
7. BCM Electrical Characteristics	94
8. EEPROM Block Characteristics	94
9. Electrical Characteristics Graphs.....	95
PACKAGING INFORMATION	97
16-SOP Package Dimension	98
16-DIP Package Dimension	99

FEATURES

1. ROM: 2K x 16 bits Flash Program Memory

- 10K erase times at least
- 10 years data retention at least

2. EEPROM: 128 x 8 bits

- 50K erase times at least
- 10 years data retention at least

3. RAM: 176 x 8 bits

4. STACK: 8 Levels

5. System Oscillation Sources (Fsys) :

- Fast-clock
 - FIRC (Fast Internal RC) : 8 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 / 8 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 1~32768 pre-scale option / auto-reload / counter / interrupt / stop function
- Timer1
 - 8-bit timer divided by 1~256 pre-scale option / auto-reload / interrupt / stop function
- 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) or 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 / Wake-up Timer Interrupt
- ADC Interrupt
- PWM1 period and PWM1A / PWM1B / PWM1C duty Interrupt

11. Wake-up Timer (WKT)

- 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 (WDT)

- 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. PWM x 4

- PWM0
 - 8+2 bits, duty-adjustable, period-adjustable controlled PWM
 - PWM0 clock source: Fast-clock or FIRC 8 MHz / 16MHz, with 1~8 pre-scalers
 - Complementary PWM output (PWM0P, PWM0N)
 - Non-overlap time durations adjustable: (0~8)*(PWMCLK)
- PWM1A / PWM1B / PWM1C
 - 16-bit PWM1 with three groups independent duty-adjustable function and shared period-adjustable controlled
 - PWM1 shared clock source: System clock (Fsys) or FIRC 8 MHz / 16 MHz
 - With duty and period interrupt function

14. 12-bit ADC with 8 input channels and 1 internal reference voltage

- Internal reference voltage VR ($3V \pm 1.2\%$ @25°C, VCC=3V~5V)
- ADC reference voltage = VCC or VR

15. Battery Charging Module

- OPA x 2
- 10-bit DAC x 2
 - DAC reference voltage = VR ($3V \pm 1.2\%$ @25°C, VCC=3V~5V)

16. Reset Sources

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

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

- 4-Level Low Voltage Reset: 2.2V / 2.8V / 3.6V / 4.2V
- 3-Level Low Voltage Detection Flag: 2.8V / 3.6V / 4.2V (when LVR = 2.2V)

18. Operating Voltage

- Fsys= 4 MHz, 1.6V~5.5V @LVR disable. Suggest LVR 2.2V or above at -40°C to +85°C
- Fsys=8 MHz, 2.1V~5.5V @LVR disable. Suggest LVR 2.8V or above at -40°C to +85°C

Note: Power-up VCC must exceed LVR 2.2V and selected LVR level, refer to the “Electrical Characteristics Graphs” to avoid entering ROM deadzone.

19. Operating Temperature Range : -40°C to + 85°C**20. Table Read Instruction: 16-bit ROM data lookup table****21. Integrated 16-bit Cyclic Redundancy Check function****22. Instruction set: 39 Instructions****23. I/O ports:**

- Maximum 8 programmable I/O pins
 - Open-Drain Output
 - CMOS Push-Pull Output
 - Schmitt Trigger Input with pull-up resistor option
 - All I/O with High-Sink and High-Drive
- 5 OPA analog I/O pins
- 1 reference voltage output pin (VR = 3V ±1.2% @25°C, VCC=3V~5V)

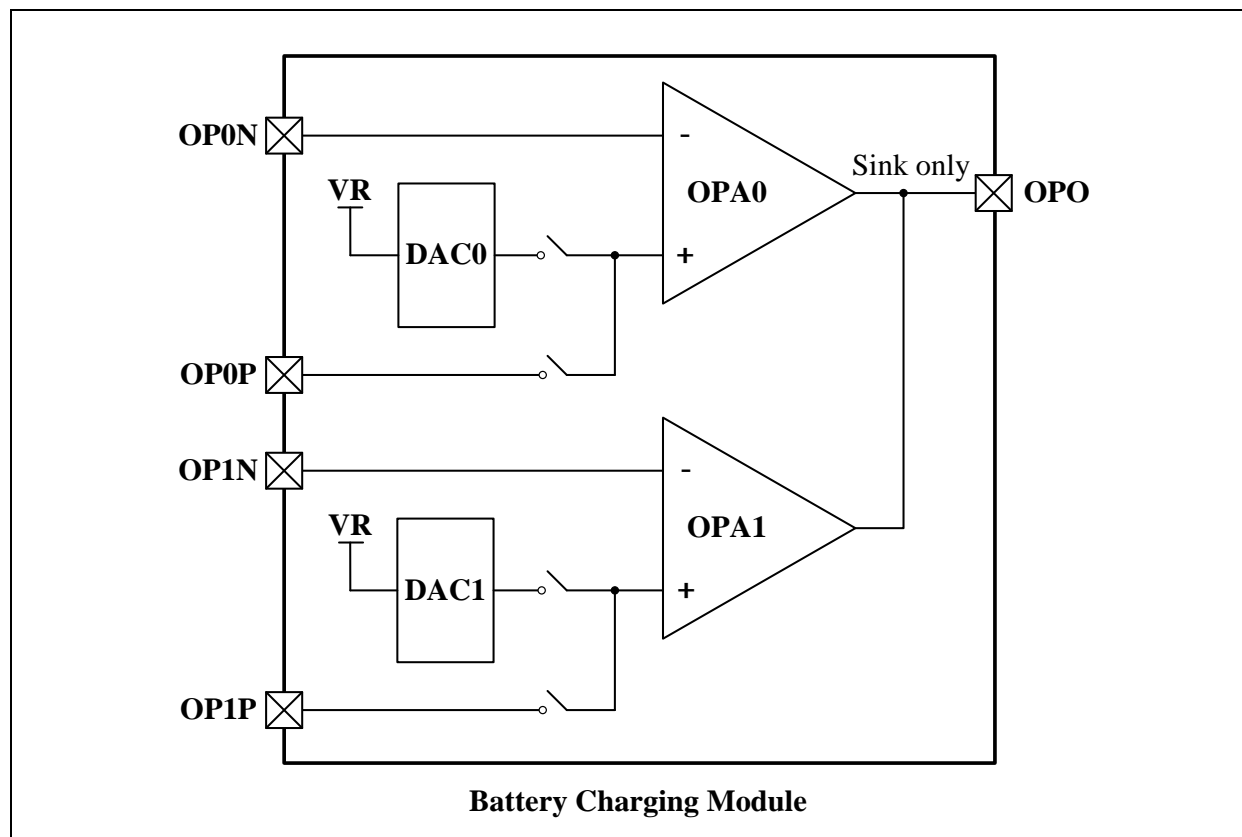
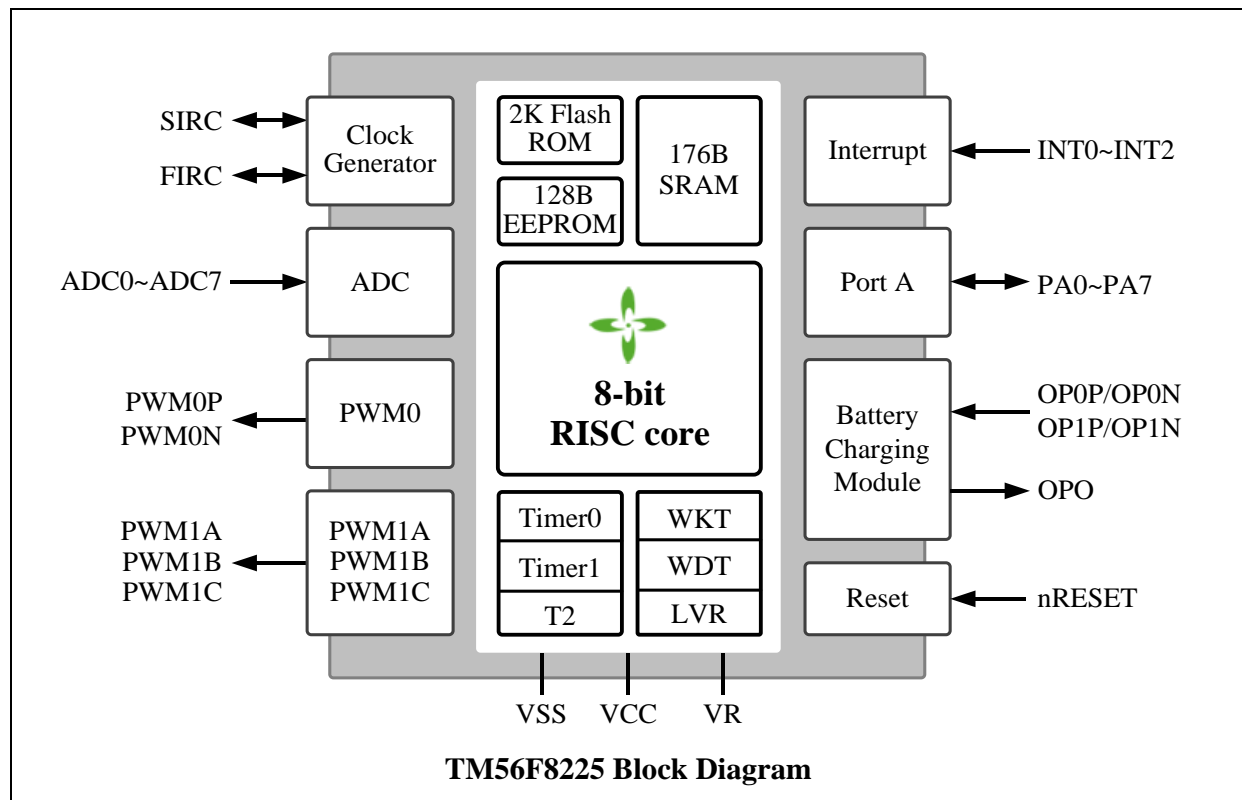
24. Programming connectivity support 4-wire (ICP) or 7-wire program**25. Package Types:**

- 16-pin SOP (150 mil)
- 16-pin DIP (300 mil)

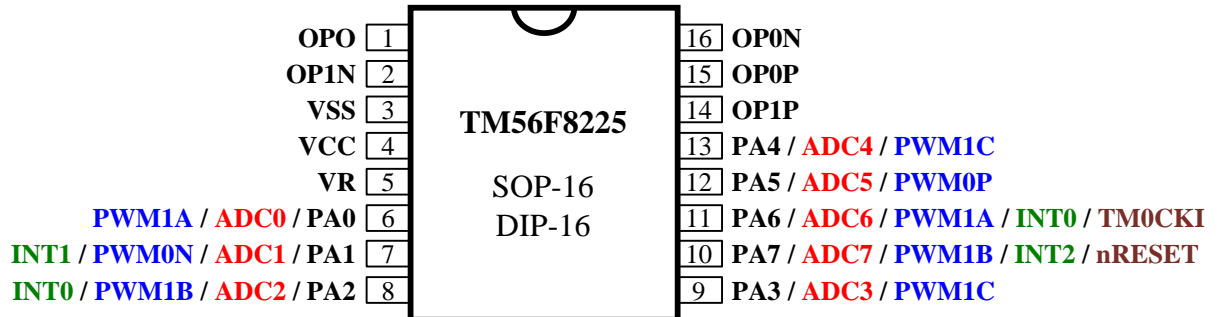
26. Supported EV board on ICE

EV board: EV8235

SYSTEM BLOCK DIAGRAM



PIN ASSIGNMENT DIAGRAM



PIN DESCRIPTIONS

Name	In/Out	Pin Description
PA0–PA7	I/O	Bit-programmable I/O port for Schmitt-trigger input, CMOS push-pull output or open-drain output. Pull-up resistors are assignable by software.
nRESET	I	External active low reset
VCC, VSS	P	Power Voltage input pin and ground
VR	O	Internal reference voltage 3V output (at least 1uF to ground)
INT0–INT2	I	External interrupt input
TM0CKI	I	Timer0's input in counter mode
PWM0P	O	(8+2) bit PWM0 positive output
PWM0N	O	(8+2) bit PWM0 negative output
PWM1A	O	16 bit PWM1 output
PWM1B	O	16 bit PWM1 output
PWM1C	O	16 bit PWM1 output
ADC0–ADC7	I	ADC channels input
OP0P	I	OPA0 positive input
OP0N	I	OPA0 negative input
OP1P	I	OPA1 positive input
OP1N	I	OPA1 negative input
OPO	O	OPA0, OPA1 output

Programming pins:

Normal mode: VCC / VSS / PA0 / PA1 / PA2 / PA3 / PA4

ICP mode: 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 AfterReset	Alternate Function			
16-SOP/DIP			Input	Output			PWM	ADC	OPA	MISC
			Ext. Interrupt	O.D	P.P					
1	OPO	O							○	
2	OP1N	I							○	
3	VSS	P								
4	VCC	P								
5	VR	O								
6	PA0/ADC0/PWM1A	I/O		○	○	PA0	○	○		
7	PA1/ADC1/PWM0N/INT1	I/O	○	○	○	PA1	○	○		
8	PA2/ADC2/PWM1B/INT0	I/O	○	○	○	PA2	○	○		
9	OP0N	I							○	
10	OP0P	I							○	
11	OP1P	I							○	
12	PA4/ADC4/PWM1C	I/O		○	○	PA4	○	○		
13	PA5/ADC5/PWM0P	I/O		○	○	PA5	○	○		
14	PA6/ADC6/PWM1A/INT0/TM0CKI	I/O	○	○	○	PA6	○	○		TM0CKI
15	PA7/ADC7/PWM1B/INT2/nRESET	I/O	○	○	○	PA7	○	○		nRESET
16	PA3/ADC3/PWM1C	I/O		○	○	PA3	○	○		

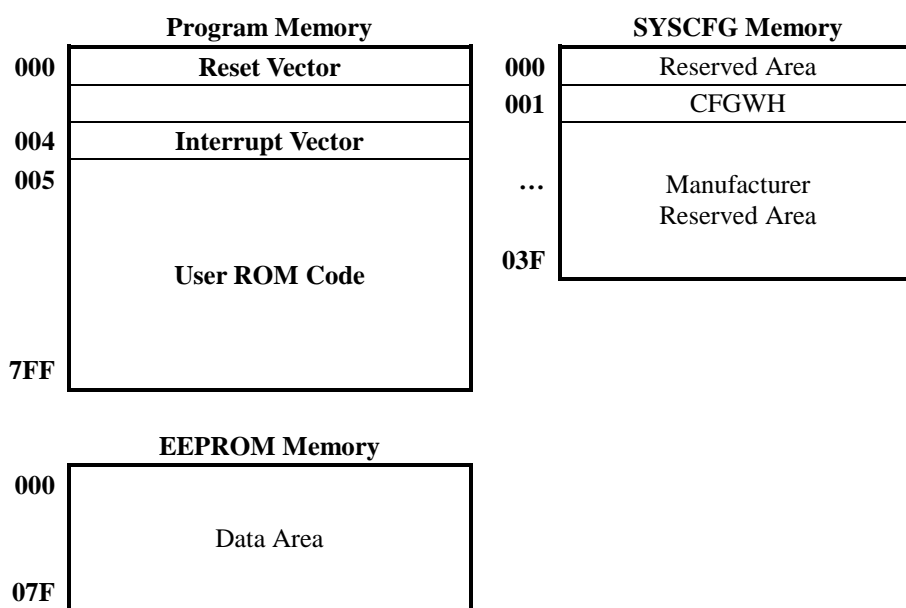
Symbol : P.P. = COM Push-Pull Output
O.D. = Open Drain Output

FUNCTION DESCRIPTION

1 CPU Core

1.1 Program ROM (PROM)

The Flash Program ROM of this device is 2K 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 bit of SYSCFG is not set. The SYSCFG can be read no matter PROTECT bit is set or cleared, but PROTECT bit can be cleared only when User ROM Code area is erased. That is, unprotect the PROTECT bit needs to erase the corresponding ROM area. 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 a 13 bits register (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		00_0000_0000_0000	
Bit		Description	
CFGWH	13	PROTECT: Code protection selection	
		1	Enable
		0	Disable
	12	XRSTE: External Pin (PA7) Reset Enable	
		1	Enable
		0	Disable (PA7 as input I/O pin)
	11-10	LVR: Low Voltage Reset Mode	
		11	4.2V
		10	3.6V
		01	2.8V
		00	2.2V + LVD function (LVDS 00: 3.6V 01: 2.8V 1X: 4.2V)
	9-8	WDTE: WDT Reset Enable	
		11	Always Enable
		10	Enable in FAST/SLOW mode, Disable in IDLE/STOP mode
		0X	Disable
	7-0	tenx Reserved	

1.3 Data ROM (EEPROM)

The TM56F8225 contains 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
CLRXL    DPH                ; set DPH=0 for EEPROM write/read
MOVLW    00H
MOVWX    DPH
MOVLW    23H                ; set DPTR=0023h
MOVWX    DPL

```

```

; Read EEPROM @Address 23h data into W by using opcode TABRL
TABRL
...

```

```

; Another way to read EEPROM @Address 23h data into W by using TABR
MOVLW    01H
MOVWX    TABR                ; TABR = 01h = opcode TABRL
...

```

The EEPROM Write usage is similar to read EEPROM expect 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
CLRXL    DPH                ; set DPH=0 for EEPROM write/read
MOVLW    23H
MOVWX    DPL                ; set DPTR=0023h
MOVLW    00000011B
MOVWX    EEPCTL              ; set EEPROM write with 12.8mS time out
MOVLW    37H                ; set W=LVRPD=37h, force LVR disable
MOVWX    LVRPD              ; 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
CLRXL    EEPEN              ; protect EEPROM from abnormal write

```

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 Slow-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 8 1: div 4 2: div 2 3: div 1

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

109h.7~0 **LVRPD**: LVR power down register

Write 37h to force LVR disable. (LVR must be disabled before EEPROM Write operation)

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

18Ch.7~0 1. TABR write 01h = opcode TABRL

2. TABR write 02h = opcode TABRH

3. After step.1 or step.2, read TABR to get main ROM table read value

After step.1, read TABR to get EEPROM value (when EEPEN = E2h)

Table Read for ASM: TABRL/ TABRH or TABR

Table Read for C: TABR

18Dh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
EEPCCTL	—	—	—	—	—	EEPTO	EEPTE	
R/W	—	—	—	—	—	R	R/W	R/W
Reset	—	—	—	—	—	0	0	0

18Dh.2 **EEPTO**: EEPROM write time-Out flag

Set by H/W when EEPROM write time-out occurs

Cleared by H/W when EEPTE=0

18Dh.1~0 **EEPTE**: EEPROM write watchdog timer enable (**suggest set 11**)

00: disable 01: 1.6ms 10: 6.4ms 11: 12.8ms

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

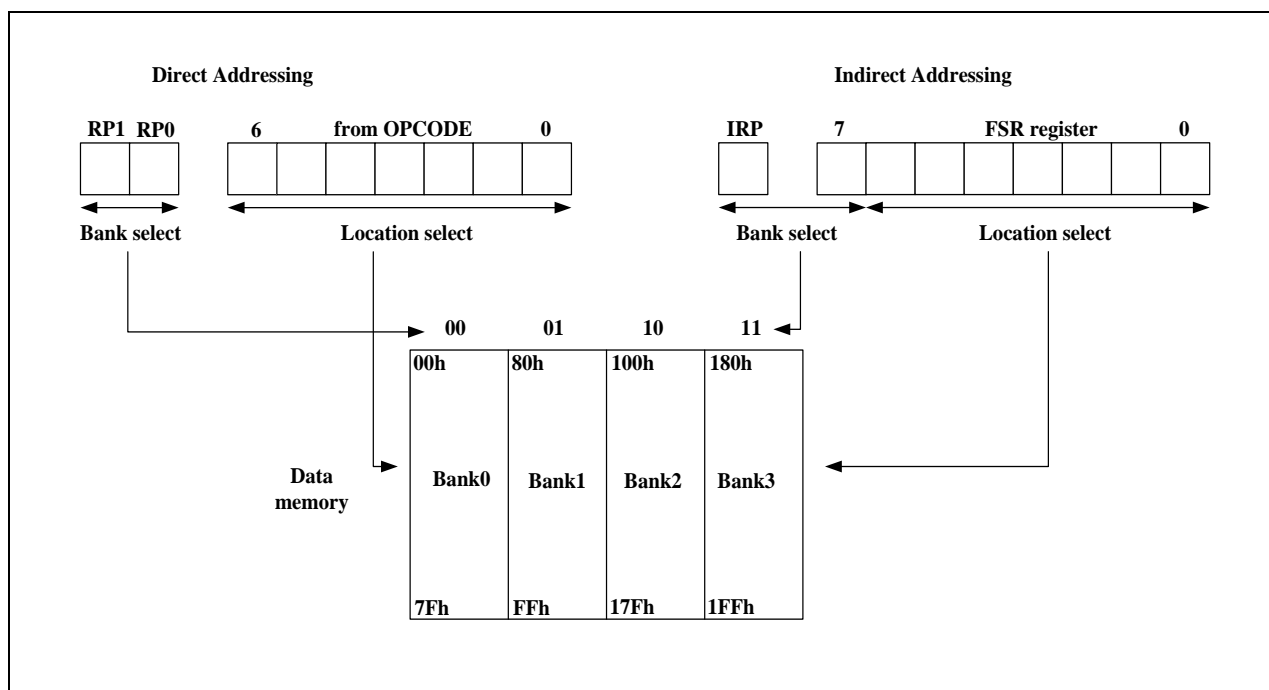
1.4 RAM Addressing Mode

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 INDF register indirectly results in a no operation (although status bit may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS[7]). Refer to the figure below.



Direct / Indirect Addressing

Keeping RP0=RP1=0 in the beginning 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:

BCF	TM0IE	→	BCX	TM0IE
DEC	CNT, 1	→	DECX	CNT, 1
INCF	RAM25, 0	→	INCX	RAM25, 0
MOVWF	PAMODL	→	MOVWX	PAMODL
RLF	RAMA0, 0	→	RLX	RAMA0, 0
SWAPF	ADCTL, 0	→	SWAPX	ADCTL, 0

【BANK0】 00h~7Fh		【BANK1】 80h~FFh		【BANK2】 100h~17Fh		【BANK3】 180h~1FFh	
00h	INDF	80h	INDF	100h	INDF	180h	INDF
01h	TM0	81h	OPTION	101h	TM0	181h	OPTION
02h	PCL	82h	PCL	102h	PCL	182h	PCL
03h	STATUS	83h	STATUS	103h	STATUS	183h	STATUS
04h	FSR	84h	FSR	104h	FSR	184h	FSR
05h	PAD	85h		105h		185h	DPL
06h		86h		106h		186h	DPH
07h		87h		107h		187h	CRCDL
08h		88h		108h		188h	CRCDH
09h		89h		109h	LVRPD	189h	CRCIN
0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah	PCLATH
0Bh	INTIE	8Bh	INTIE	10Bh	INTIE	18Bh	INTIE
0Ch	INTIF	8Ch	PAMODH	10Ch		18Ch	TABR
0Dh	INTIE1	8Dh	PAMODL	10Dh		18Dh	EEPCTL
0Eh	INTIF1	8Eh		10Eh	BGTRIM	18Eh	EEPEN
0Fh	CLKCTL	8Fh		10Fh	IRCF	18Fh	EEPDT
10h	TM0RLD	90h		110h		190h	
11h	TM0CTL	91h	PWMOE	111h		191h	
12h	TM1	92h	PWM0PRD	112h		192h	
13h	TM1RLD	93h	PWM0DH	113h		193h	
14h	TM1CTL	94h	PWM0DL	114h		194h	
15h	T2CTL	95h	PWM0CTL	115h		195h	
16h	MF016	96h	PWM0CTL1	116h		196h	
17h	ADCH	97h	PWM1CTL	117h		197h	
18h	ADCTL	98h	PWM1PRDH	118h		198h	
19h	MF019	99h	PWM1PRDL	119h		199h	
1Ah	OPA0CTL	9Ah	PWM1ADH	11Ah		19Ah	
1Bh	OPA1CTL	9Bh	PWM1ADL	11Bh		19Bh	
1Ch	DAC0DH	9Ch	PWM1BDH	11Ch		19Ch	
1Dh	DAC0DL	9Dh	PWM1BDL	11Dh		19Dh	
1Eh	DAC1DH	9Eh	PWM1CDH	11Eh		19Eh	
1Fh	DAC1DL	9Fh	PWM1CDL	11Fh		19Fh	
20h	RAM Bank0 area (80 Bytes)	A0h	RAM Bank1 area (80 Bytes)	120h		1A0h	
~		~		~		~	
6Fh		EFh		16Fh		1EFh	
70h		F0h		170h		1F0h	
~		~		~		~	
7Fh	common area 16 Bytes	FFh	accesses 70h~7Fh	17Fh	accesses 70h~7Fh	1FFh	accesses 70h~7Fh

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

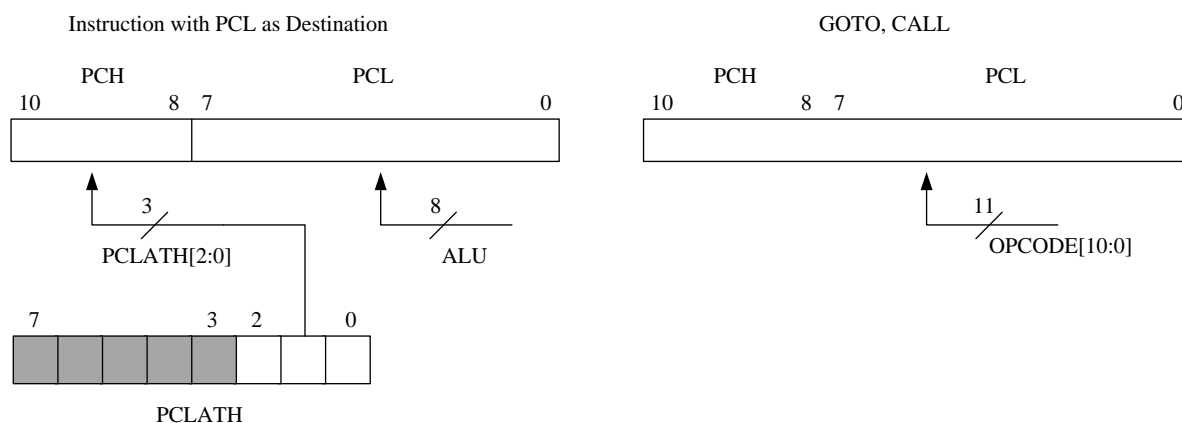
TM1	equ	12H	;SFR in Bank0
PWM0PRD	equ	92H	;SFR in Bank1
IRCF	equ	10FH	;SFR in Bank2
DPL	equ	185H	;SFR in Bank3
RAM20	equ	20H	;RAM in Bank0
RAMA0	equ	A0H	;RAM in Bank1
MOVXW	TM1		; read TM1 (Bank0) to W
MOVXW	PWM0PRD		; read PWM0PRD (Bank1) to W
MOVXW	IRCF		; read IRCF (Bank2) to W
MOVXW	DPL		; read DPL (Bank3) to W
MOVLW	16H		
MOVWX	RAM20		; W = 16h write to RAM[0x20]
MOVWX	RAMA0		; W = 16h write to RAM[0xA0]
MOVLW	37H		
MOVWX	LVRPD		; LVRPD = W = 37h, force LVR disable
MOVXW	CLKCTL		; read SFR CLKCTL (0Fh) to W
MOVXW	IRCF		; read SFR IRCF (10Fh) to W
MOVLW	0BH		
MOVWX	CLKCTL		; CLKCTL (0Fh) = W = 0Bh
MOVWX	IRCF		; IRCF (10Fh) = W = 0Bh

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

BSX	IRP		; IRP = 1 => Bank2/3
MOVLW	0FH		; W = 0Fh
MOVWX	FSR		; FSR = W = 0Fh
MOVXW	INDF		; read SFR IRCF (10Fh) to W
BSX	IRP		; IRP = 1 => Bank2/3
MOVLW	0FH		; W = 0Fh
MOVWX	FSR		; FSR = W = 0Fh
MOVLW	0BH		; W = 0Bh
MOVWX	INDF		; IRCF (10Fh) = W = 0Bh
BCX	IRP		; IRP=0 =>Bank0/1
MOVLW	0FH		; W = 0Fh
MOVWX	FSR		; FSR = W = 0Fh
MOVXW	INDF		; read SFR CLKCTL (0Fh) to W
BCX	IRP		; IRP = 0 => Bank0/1
MOVLW	0FH		; W = 0Fh
MOVWX	FSR		; FSR = W = 0Fh
MOVLW	0BH		; W = 0Bh
MOVWX	INDF		; CLKCTL (0Fh) = W = 0Bh

1.5 Programming Counter (PC) and Stack

The Programming Counter is 11-bit wide capable of addressing a 2K x 16 Flash ROM. The low byte comes from PCL register, which is readable and writable register. The upper bits (PC[10:8]) 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 11 bit address from instruction word. For RET/RETI/RETLW instruction, PC retrieves its content from the top level STACK. Executing any instruction with the PCL register as the destination simultaneously causes the Program Counter PC[10:8] 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 3 bits to the PCLATH register. When the lower 8 bits are written to the PCL register, all 11 bits of program counter will change to the values contained in the PCLATH register and those being written to the PCL register.



The STACK is 11-bit wide and 8-level in depth. 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. It also offers another way to read the 16-bit ROM data into W register by setting TABR (18Ch) for C language.

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

```

ORG      000h                                ; Reset Vector
GOTO     START

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
    ...
    INCX  INDEX, 1                                ; Increment the index address for next address
    ...
    GOTO  LOOP                                ; Go to LOOP label
    ...
    
```

```

MOVLW    (TABLE2 >>8) & 0xff
MOVWX    DPH                                ; DPH register (F186.2~0)
MOVLW    (TABLE2) & 0xff
MOVWX    DPL                                ; DPL register (F185.7~0)

```

; Table Read by opcode TABRL / TABRH

```

TABRL                                ; read PROM low byte data to W (W=86h)
TABRH                                ; read PROM high byte data to W (W=19h)
...

```

; Another way of Table Read by sfr TABR

```

MOVLW    01H                            ; TABR = 01H = opcode TABRL
MOVWX    TABR                          ; read PROM low byte data to W (W=86h)
MOVLW    02H                            ; TABR = 02H = opcode TABRH
MOVWX    TABR                          ; read PROM high byte data to W (W=19h)
...

```

TABLE1:

```

ADDWX    PCL, 1                        ; Add the W with PCL, the result back in PCL.
RETLW    55H                          ; W=55h when return
RETLW    56H                          ; W=56h when return
RETLW    57H                          ; W=57h when return
RETLW    58H                          ; W=58h when return
...

```

ORG 368h

TABLE2:

```

.DT      0x1986                        ; 16-bit ROM data
.DT      0x3719
.DT      0x2983
...

```

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

- 18Ch.7~0
1. TABR write 01h = opcode TABRL
 2. TABR write 02h = opcode TABRH
 3. After step.1 or step.2, read TABR to get main ROM table read value
After step.1, read TABR to get EEPROM value (when EEPEN = E2h)
- Table Read for ASM: TABRL/ TABRH or TABR
Table Read for C: TABR

1.5.1 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.

1.5.2 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.

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

◇ Example: Write immediate data into STATUS register.

```
MOVLW    00H
MOVW     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 (WDT), 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 CFGWH register value.

2.2 Low Voltage Reset

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

LVR Selection Table:

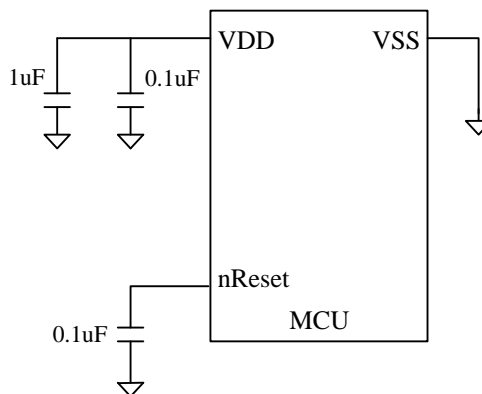
LVR level	Operating voltage
LVR2.2	$5.5V > VCC > 2.2V$
LVR2.8	$5.5V > VCC > 2.8V$
LVR3.6	$5.5V > VCC > 3.6V$
LVR4.2	$5.5V > VCC > 4.2V$ or $V_{CC}=5.0V$

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 CFGWH 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.

External reset pin is low level active. The system is running when reset pin is high level voltage input. The reset pin receives the low voltage and the system is reset. The external reset can reset the system during power on duration, and good external reset circuit can protect the system to avoid working at unusual power condition.



2.4 Watchdog Timer Reset

WDT overflow Reset can be disabled or enabled by the CFGWH register. It runs in Fast/Slow mode and runs or stops in IDLE/STOP mode. WDT overflow speed can be defined by WDT_PSC 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                ; Reset Vector
GOTO     START              ; Jump to user program address.

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

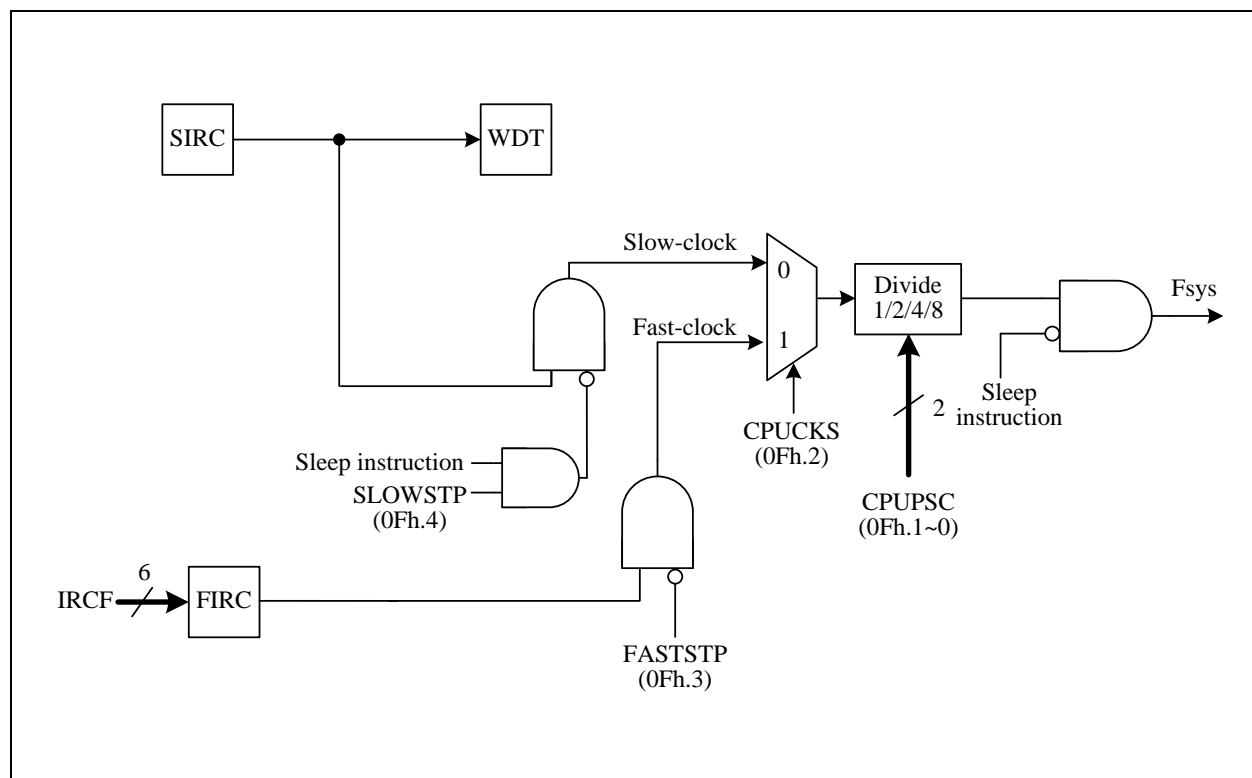
3 Clock Circuitry and Operation Mode

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 70 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, a 8MHz System clock rate requires $V_{CC} > 2.1V$.

The CLKCTL (0Fh) SFR controls the System clock operating. H/W automatically blocks the S/W abnormally setting for this register. 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 (10Fh) . When IRCF=00h, frequency is the lowest. When IRCF=7Fh, frequency is the highest. With this function, we can adjust the frequency of FIRC after power on. Each IC may have different default value of IRCF, to make sure the frequency of FIRC=8 MHz after Power on Reset.

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 FIRC 8M, FIRC 16M 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, Timer1 etc...) 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 Slow-clock oscillation in IDLE mode is setting WKTIE=1 (0Bh.3) to keeping WKT running before executing the SLEEP instruction or WDTE=11 (CFGWH.9~8) to keeping WDT running. In such condition, the Slow-clock keeps working and wakes up CPU periodically no matter SLOWSTP is set or cleared.

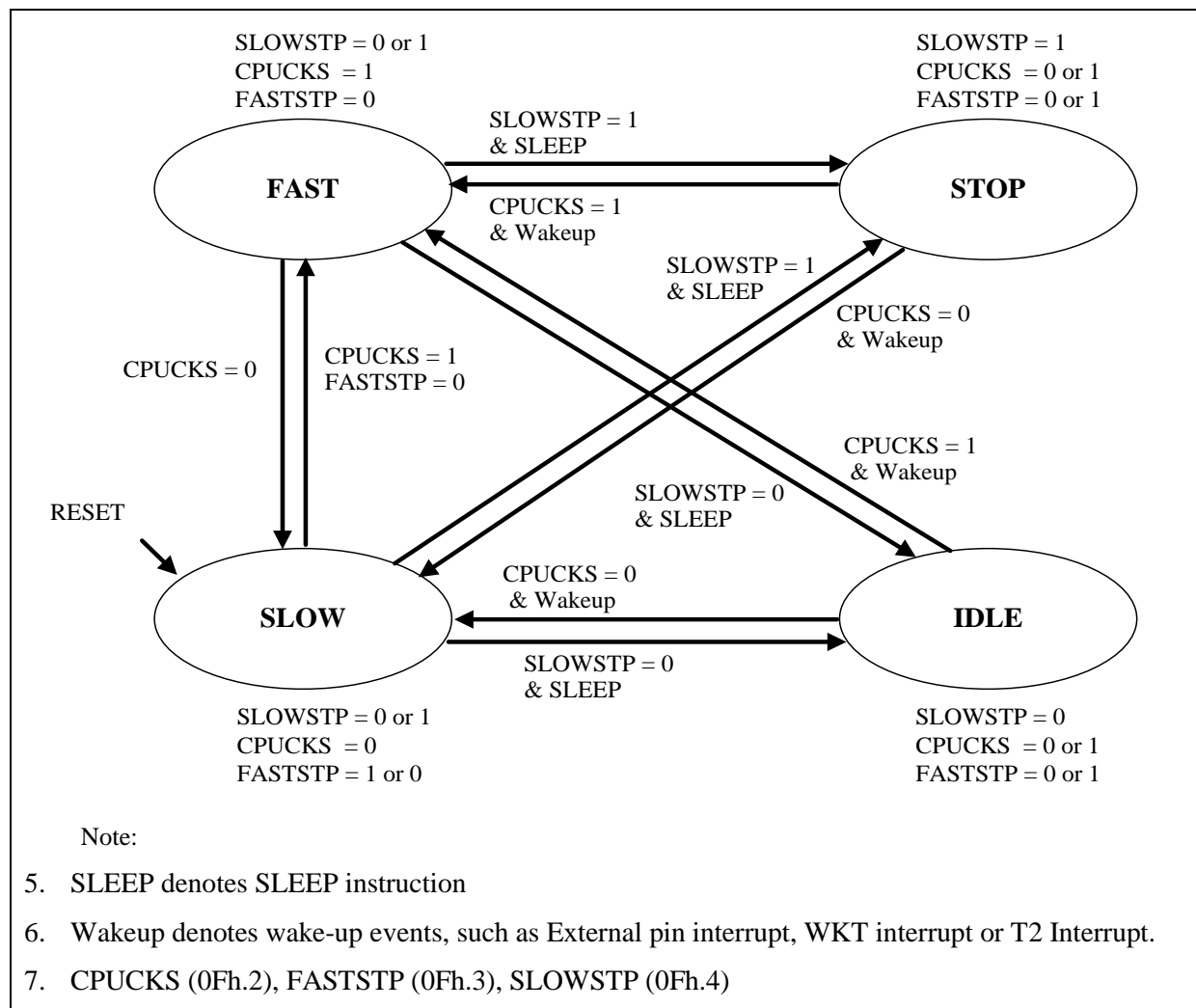
T2 and WKT/WDT are independent and have their own control registers. It is possible to keep both T2 and WKT working and wake-up in the IDLE mode.

STOP Mode:

When SLOWSTP (0Fh.4) is set, WKTIE (0Bh.3) is cleared and WDTE=10 or 0X, all blocks will be turned off and the Chip will enter the “STOP Mode” after executing the SLEEP instruction. STOP mode is similar to IDLE mode. The difference is all clock oscillators either Fast-clock or Slow-clock are stopped and no clocks are 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.



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 wake up by External interrupt, WKT interrupt and T2 interrupt.

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

BCX	SLOWSTP	; Enable Slow-clock
MOVLW	00000 <u>000</u> B	
MOVWX	T2CTL	
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 woken up only by External pin interrupt.

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

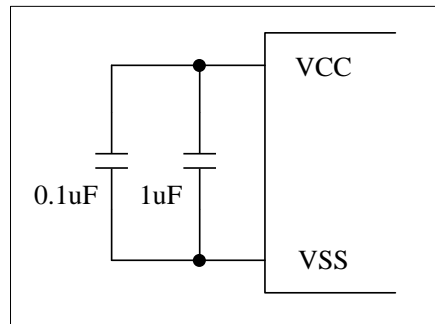
```
BSX      SLOWSTP      ; Disable Slow-clock.
MOVLW    00000000B    ; Disable WKT counting
MOVWX    INTIE
SLEEP                                ; Enter STOP mode.
```

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

- 0Fh.4 **SLOWSTP**: Slow-clock stop
0: Slow-clock is running
1: Slow-clock stops running in Power-down mode
- 0Fh.3 **FASTSTP**: Fast-clock stop
0: Fast-clock is running
1: Fast-clock stops running
- 0Fh.2 **CPUCKS**: System clock source select
0: Slow-clock
1: Fast-clock
- 0Fh.1~0 **CPUPSC**: System clock source prescaler. System clock source
00: divided by 8
01: divided by 4
10: divided by 2
11: divided by 1

3.3 System Clock Oscillator

In the Fast Internal RC (FIRC) mode, the on-chip oscillator generates 8 MHz system clock. Since power noise degrades the performance of Internal Clock Oscillator, placing power supply bypass capacitors 1 μF and 0.1 μF very close to VCC/VSS pins improves the stability of clock and the overall system.



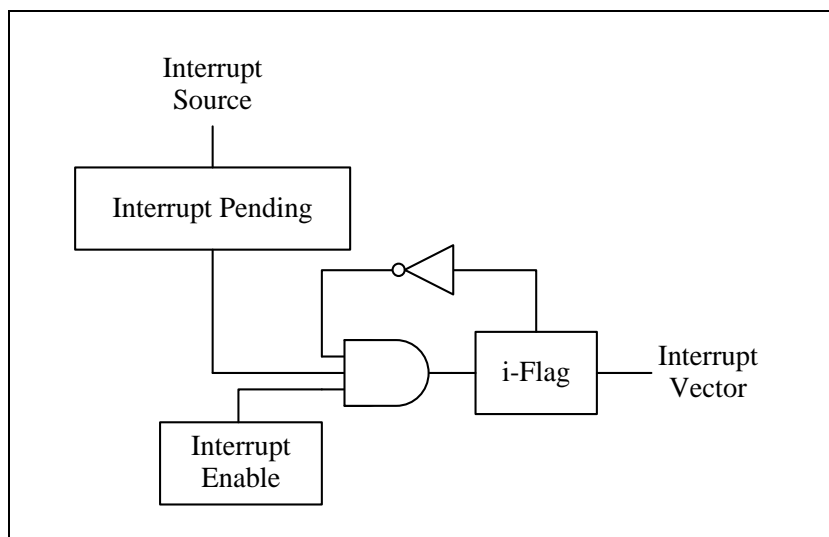
Internal RC Mode

4 Interrupt

TM56F8225 has 1 level, 1 vector and 12 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], INTIE1[3:0]) 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.



◇ Example: Setup INT1 (PA1) interrupt request with rising edge trigger

```

ORG      000H                ; Reset Vector
GOTO     START                ; Goto user program address

ORG      004H                ; All interrupt vector
GOTO     INT                  ; If INT1 (PA1) input occurred rising edge

START:
ORG      005H

MOVLW    xxxx00xxB
MOVWX    PAMODL              ; Select INT1 Pin Mode as Mode0
                                ; Open drain output low or input with Pull-up

MOVLW    xxxxxx1xB
MOVWX    PAD                 ; Release INT1, it becomes Schmitt-trigger
                                ; input with input pull-up resistor

MOVLW    001xxxxxB
MOVWX    OPTION              ; Set INT1 interrupt trigger as rising edge
MOVLW    11111101B
MOVWX    INTIF               ; Clear INT1 interrupt request flag
MOVLW    00000010B
MOVWX    INTIE               ; Enable INT1 interrupt

MAIN:
...
GOTO     MAIN

INT:
MOVWX    20H                 ; Store W data to FRAM 20H
MOVXW    STATUS              ; Get STATUS data
MOVWX    21H                 ; Store STATUS data to FRAM 21H

BTXSS    INT1IF              ; Check INT1IF bit
GOTO     EXIT_INT            ; INT1IF = 0, exit interrupt subroutine
...
MOVLW    11111101B
MOVWX    INTIF               ; Clear INT1 interrupt request flag

EXIT_INT:
MOVXW    21H                 ; Get FRAM 21H data
MOVWX    STATUS              ; Restore STATUS data
SWAPX    20H,f
SWAPX    20H,w               ; Restore W data
RETI                                ; Return from interrupt

```

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 (PA1) interrupt enable
0: disable
1: enable
- INTIE.0 **INT0IE**: INT0 (PA2 or PA6) 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 (PA1) 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 (PA2 or PA6) 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

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

0Dh.3 **PWM1PIE:** PWM1 period interrupt enable

0: disable

1: enable

0Dh.2 **PWM1CIE:** PWM1C duty interrupt enable

0: disable

1: enable

0Dh.1 **PWM1BIE:** PWM1B duty interrupt enable

0: disable

1: enable

0Dh.0 **PWM1AIE:** PWM1A duty interrupt enable

0: disable

1: enable

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

0Eh.3 **PWM1PIF:** PWM1 period interrupt event pending flag

This bit is set by H/W after PWM1 counter count to the set period, write 0 to this bit will clear this flag

0Eh.2 **PWM1CIF:** PWM1C duty interrupt event pending flag

This bit is set by H/W after PWM1 counter count to the set PWM1C duty, write 0 to this bit will clear this flag

0Eh.1 **PWM1BIF:** PWM1B duty interrupt event pending flag

This bit is set by H/W after PWM1 counter count to the set PWM1B duty, write 0 to this bit will clear this flag

0Eh.0 **PWM1AIF:** PWM1A duty interrupt event pending flag

This bit is set by H/W after PWM1 counter count to the set PWM1A duty, write 0 to this bit will clear this flag

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

81h.6 **INT0EDG:** INT0 pin interrupt trigger edge

0: falling edge to trigger

1: rising edge to trigger

81h.5 **INT1EDG:** INT1 pin interrupt trigger edge

0: falling edge to trigger

1: rising edge to trigger

81h.4 **INT0SEL:** INT0 pin select

0: PA6 1: PA2

5 I/O Port

5.1 PA0-PA7

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~PA7 pin function	PxD SFR data	Pin State	Resistor 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
Mode 3	Analog input for ADC	X	—	N	N

I/O Pin Function Table

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

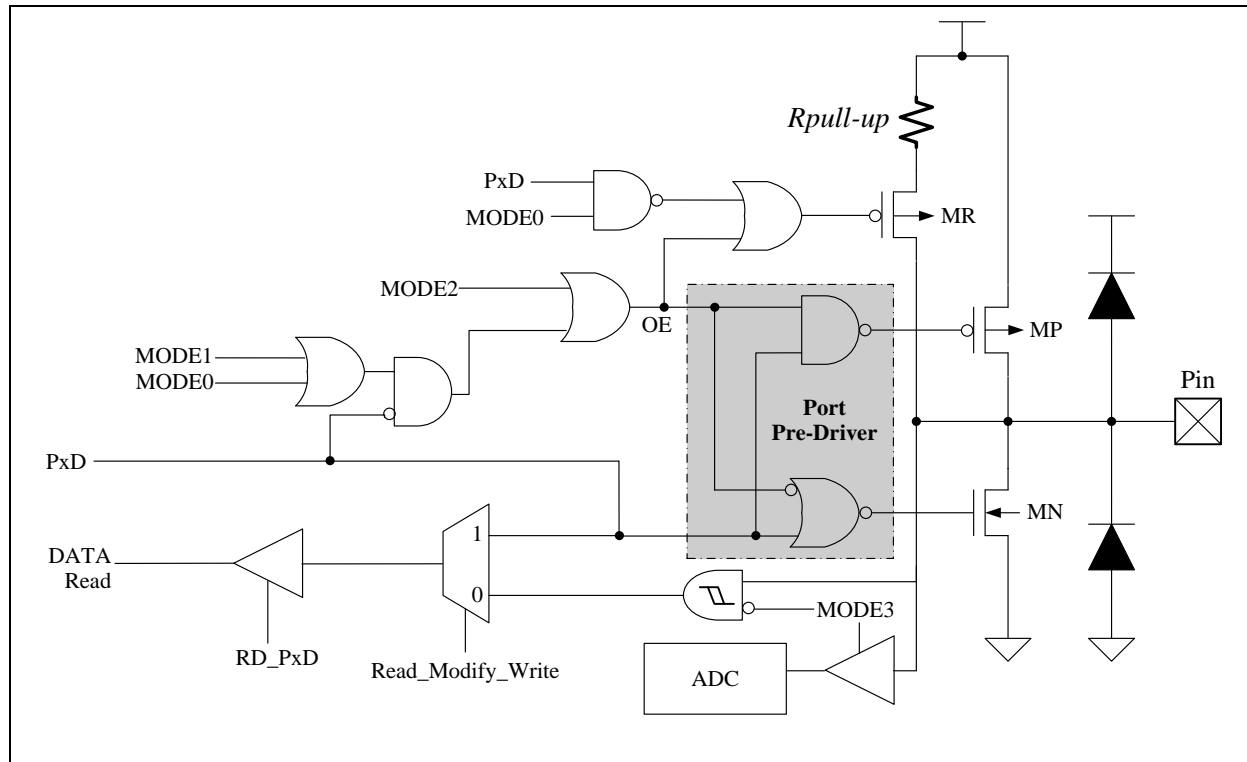
Pin Name	Wake-up	ADC	others	Mode3
PA0		ADC0	PWM1A	ADC0
PA1	INT1	ADC1	PWM0N	ADC1
PA2	INT0	ADC2	PWM1B	ADC2
PA3		ADC3	PWM1C	ADC3
PA4		ADC4	PWM1C	ADC4
PA5		ADC5	PWM0P	ADC5
PA6	INT0	ADC6	TM0CKI/PWM1A	ADC6
PA7	INT2	ADC7	PWM1B	ADC7

PortA 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, INT2 TM0CKI	0	1	Input with Pull-up	INTxIE TM0CTL
	1	1	Input	
ADC0~ADC7	3	X	ADC Channel	ADCHS
PWM0N, PWM0P, PWM1A, PWM1B, PWM1C	1	X	PWM Output (Open Drain)	PWMOE
	2	X	PWM Output (COMS Output)	

Mode Setting for Port Alternative Function



General Pin Structure

◇ Example: Set PA0 as Schmitt-trigger input with pull-up (Mode0)

```

MOVLW    xxxxxx1B
MOVWX    PAD
MOVLW    xxxxxx00B
MOVWX    PAMODL           ; Set PA0 as Schmitt-trigger input with pull-up
    
```

◇ Example: Set PA0 as Schmitt-trigger input without pull-up (Mode1)

```

MOVLW    xxxxxx1B
MOVWX    PAD
MOVLW    xxxxxx01B
MOVWX    PAMODL           ; Set PA0 as Schmitt-trigger input without pull-up
    
```

◇ Example: Set PA0 as CMOS push-pull output mode (Mode2)

```

MOVLW    xxxxxx10B
MOVWX    PAMODL
    
```

◇ Example: Set PA0 as ADC0 analog input mode (Mode3)

```

MOVLW    xxxxxx11B
MOVWX    PAMODL           ; Set PA0 as mode3
    
```

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

- 8Ch.7~6 **PA7MOD**: PA7 Pin Mode Control
 00: Mode0
 01: Mode1
 10: Mode2
 11: Mode3, PA7 as ADC7 channel input
- 8Ch.5~4 **PA6MOD**: PA6 Pin Mode Control
 00: Mode0
 01: Mode1
 10: Mode2
 11: Mode3, PA6 as ADC6 channel input
- 8Ch.3~2 **PA5MOD**: PA5 Pin Mode Control
 00: Mode0
 01: Mode1
 10: Mode2
 11: Mode3, PA5 as ADC5 channel input
- 8Ch.1~0 **PA4MOD**: PA4 Pin Mode Control
 00: Mode0
 01: Mode1
 10: Mode2
 11: Mode3, PA4 as ADC4 channel input

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

- 8Dh.7~6 **PA3MOD**: PA3 Pin Mode Control
 00: Mode0
 01: Mode1
 10: Mode2
 11: Mode3, PA3 as ADC3 channel input
- 8Dh.5~4 **PA2MOD**: PA2 Pin Mode Control
 00: Mode0
 01: Mode1
 10: Mode2
 11: Mode3, PA2 as ADC2 channel input
- 8Dh.3~2 **PA1MOD**: PA1 Pin Mode Control
 00: Mode0
 01: Mode1
 10: Mode2
 11: Mode3, PA1 as ADC1 channel input
- 8Dh.1~0 **PA0MOD**: PA0 Pin Mode Control
 00: Mode0
 01: Mode1
 10: Mode2
 11: Mode3, PA0 as ADC0 channel input

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

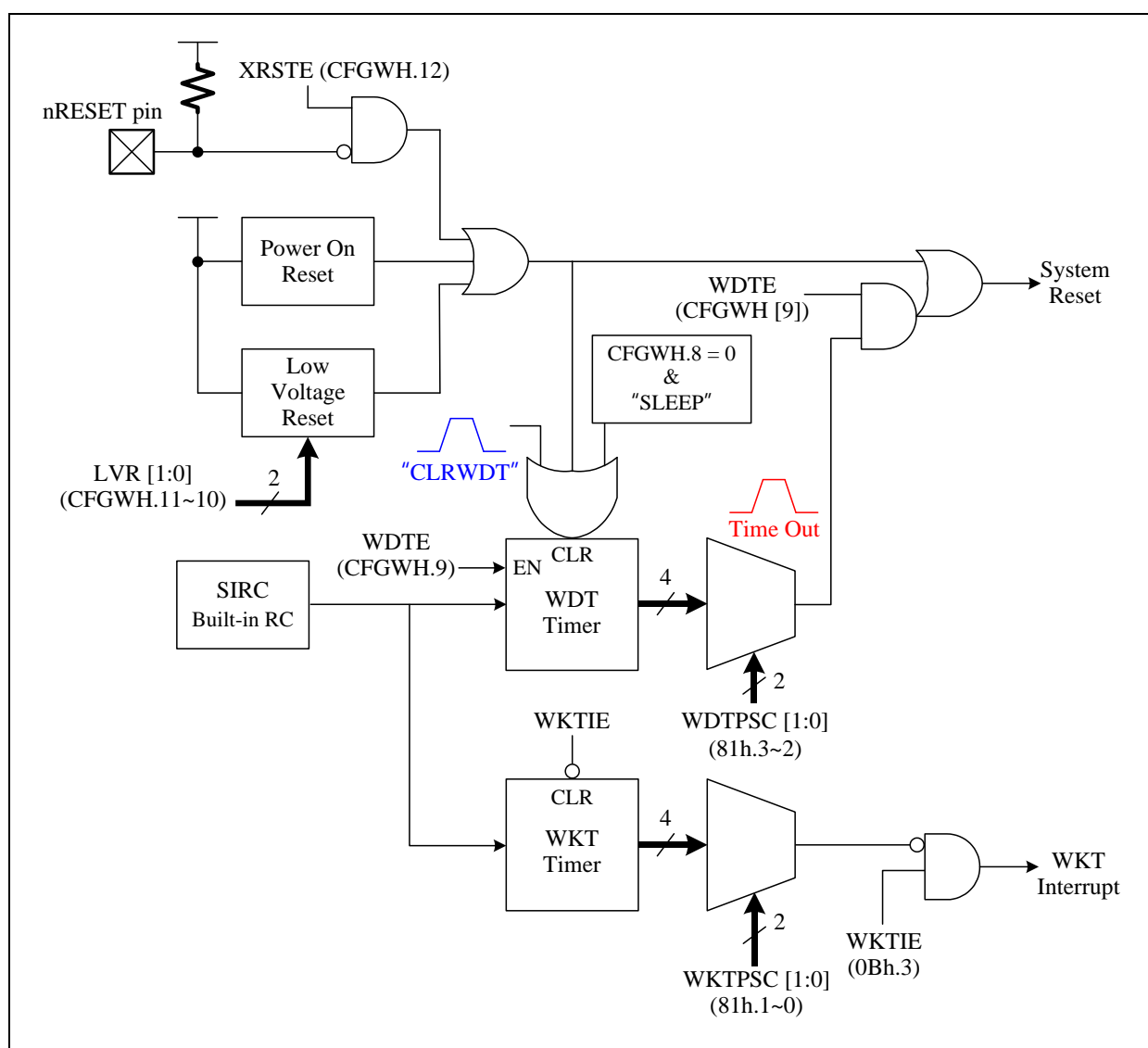
05h.7~0 **PAD:** PA7~PA0 data

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

The WDT's behavior in different Mode is shown as below table.

Mode	WDTE[1]	WDTE[0]	WDT
Normal Mode	0	0	Stop
	0	1	Stop
	1	0	Run
	1	1	Run
Power-down Mode (SLEEP)	0	0	Stop
	0	1	Stop
	1	0	Stop
	1	1	Run

CLRWDWT instruction could clear watchdog timer.

◇ Example: Clear watchdog timer by CLRWDWT instruction.

```

MAIN:  ...                ; Execute program.
        CLRWDWT           ; Execute CLRWDWT 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 flag by using byte operation
        MOVWX    INTIF           ; Don't use bit operation "BCX WKTIF" to clear
        MOVLW    00001000B
        MOVWX    INTIE           ; Enable WKT interrupt function

```

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

03h.4 **TO:** WDT time out flag, read-only
0: after Power On Reset, LVR Reset, or CLRWDT / SLEEP instructions
1: WDT time out occurs

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

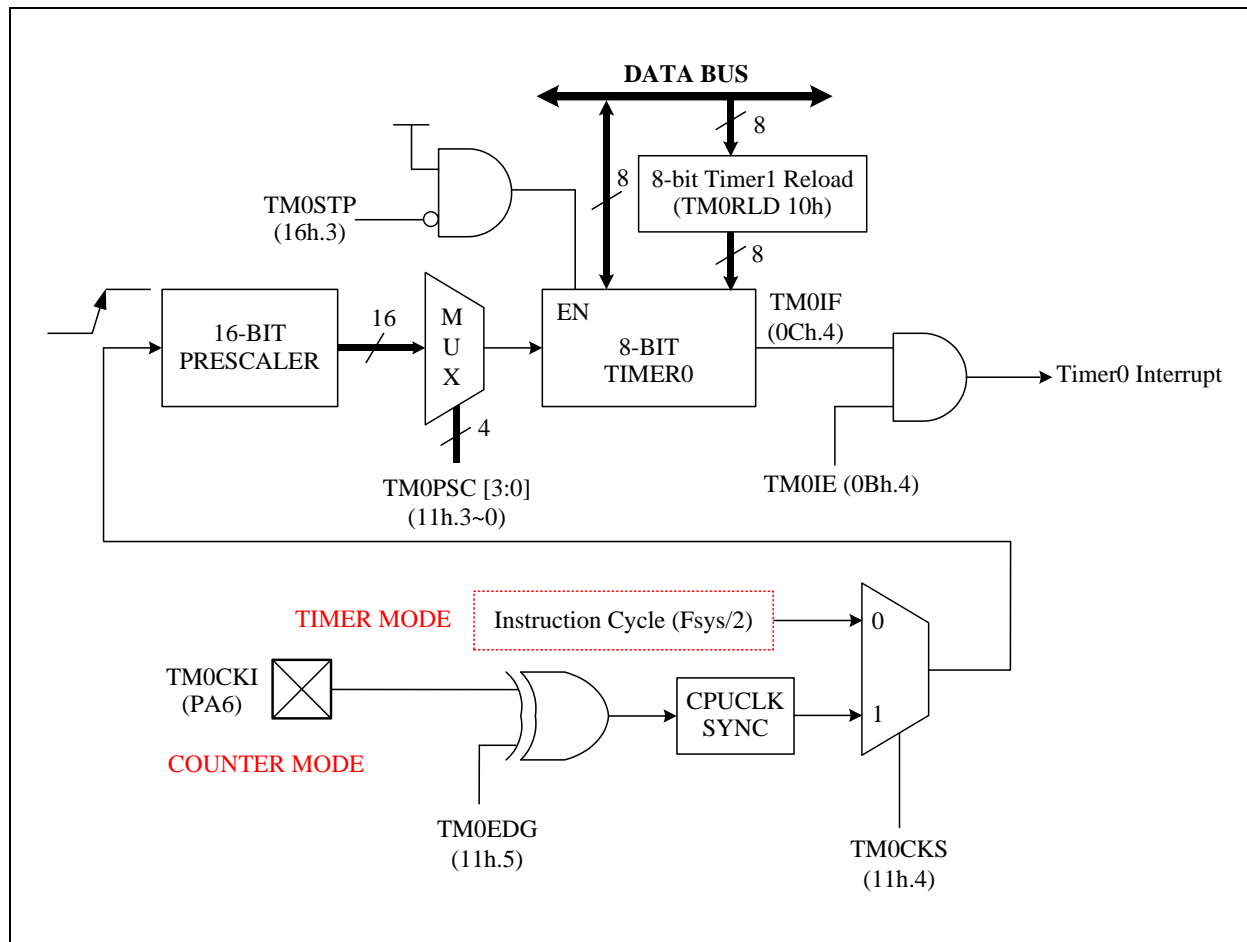
81h/181h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OPTION	HWAUTO	INT0EDG	INT1EDG	INT0SEL	WDTTPSC		WKTTPSC	
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	1	1	1	1

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

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

6.2 Timer0

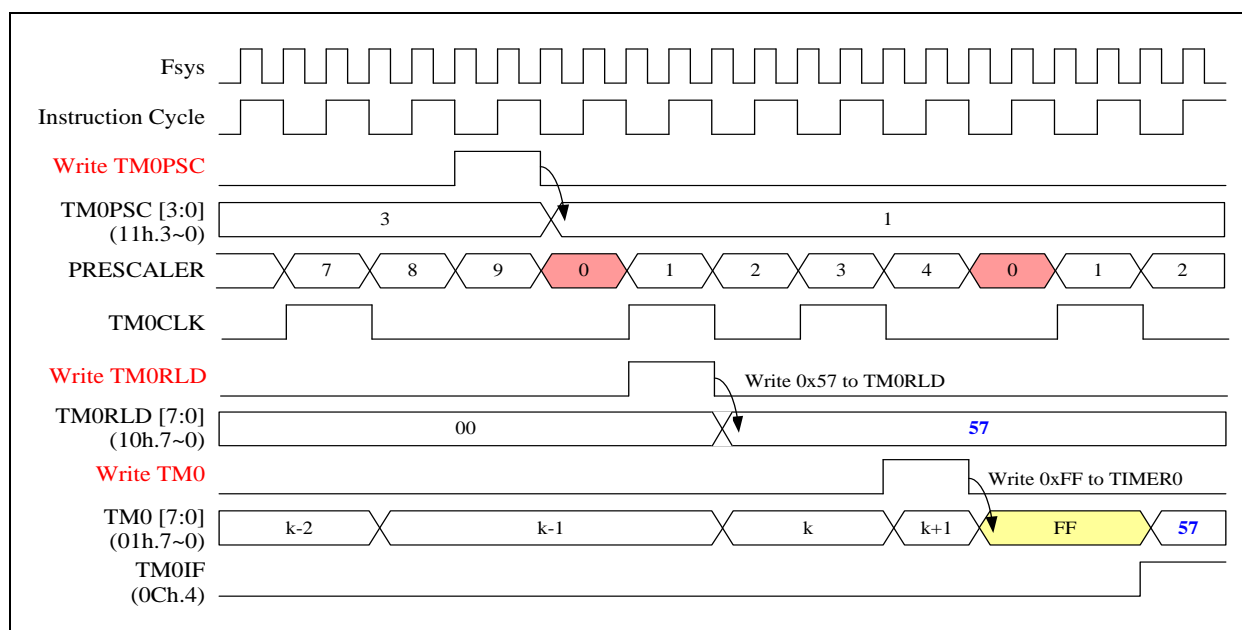
The Timer0 is an 8-bit wide register 01h (TM0). It can be read or written as any other register. Besides, Timer0 increases itself periodically and automatically rolls over a new "offset value" (TM0RLD) while it rolls over based on the pre-scaled clock source, which can be $F_{sys}/2$ or TM0CKI (PA6) 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:

$$\text{TM0 interrupt interval cycle time} = F_{\text{sys}} / 2 / \text{TM0PSC} / (256 - \text{TM0RLD})$$

◇ Example: Setup Timer0 work in Timer mode, if $F_{\text{sys}} = 8 \text{ MHz}$

; Setup Timer0 clock source and divider

BSX	CPUCKS	; Set Fast-clock as system clock
MOVLW	00x <u>00101</u> B	; TM0CKS = 0, Timer0 clock is instruction cycle
MOVWX	TM0CTL	; TM0PSC = 0101b, divided by 32

; Setup Timer0 reload data

MOVLW	80H	
MOVWX	TM0RLD	; Set Timer0 reload data = 128

; Setup Timer0

BSX	TM0STP	; Timer0 stops counting
CLRXL	TM0	; Clear Timer0 content

; Enable Timer0 and interrupt function

MOVLW	111 <u>0</u> 1111B	
MOVWX	INTIF	; Clear Timer0 request interrupt flag
BSX	TM0IE	; Enable Timer0 interrupt function
BCX	TM0STP	; Enable Timer0 counting

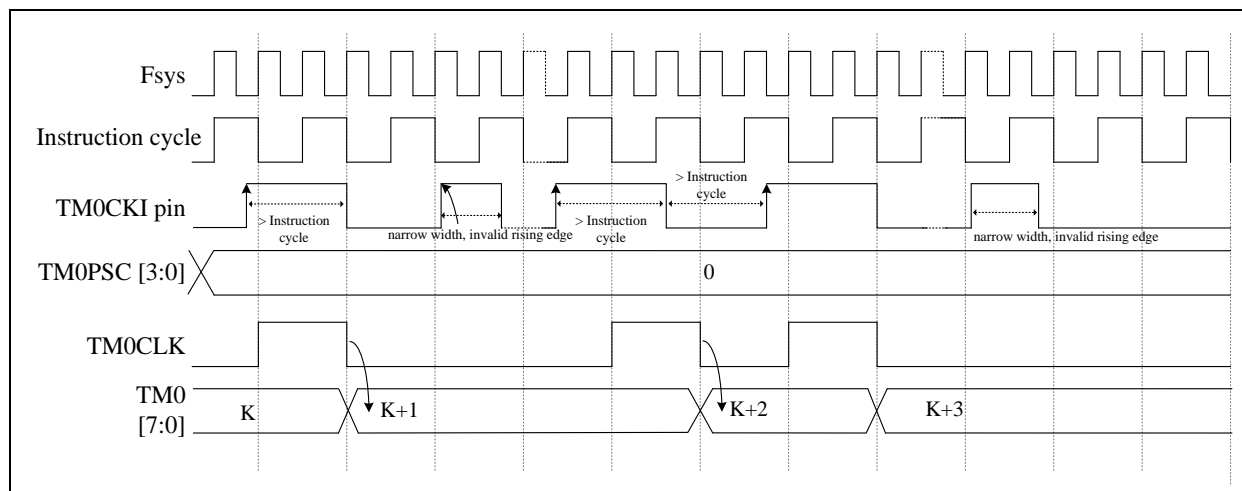
Timer0 interrupt frequency = $F_{\text{sys}} / 2 / \text{TM0PSC} / (256 - \text{TM0RLD})$,

$F_{\text{sys}} = 8\text{MHz}$, $\text{TM0PSC} = \text{div } 32$

Timer0 interrupt frequency = $8 \text{ MHz} / 2 / 32 / (256 - 128) = 0.976 \text{ KHz}$

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 ($F_{sys}/2$) that means the high/low time durations of TM0CKI must be longer than one instruction cycle time ($F_{sys}/2$) 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 (PA6)

; Setup Timer0 clock source and divider

```
MOVLW    00110000B
MOVWX    TM0CTL
```

```
; TM0EDG = 1, counting edge is falling edge
; TM0CKS = 1, Timer0 clock is TM0CKI
; TM0PSC = 0000b, divided by 1
```

; Setup Timer0

```
BSX      TM0STP
CLR      TM0
```

```
; Timer0 stops counting
; Clear Timer0 content
```

; Enable Timer0 and read Timer0 counter

```
BCX      TM0STP
...
BSX      TM0STP
MOVXW    TM0
```

```
; Enable Timer0 counting
; Timer0 stops counting
; Read Timer0 content
```

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 (PA6 pin)

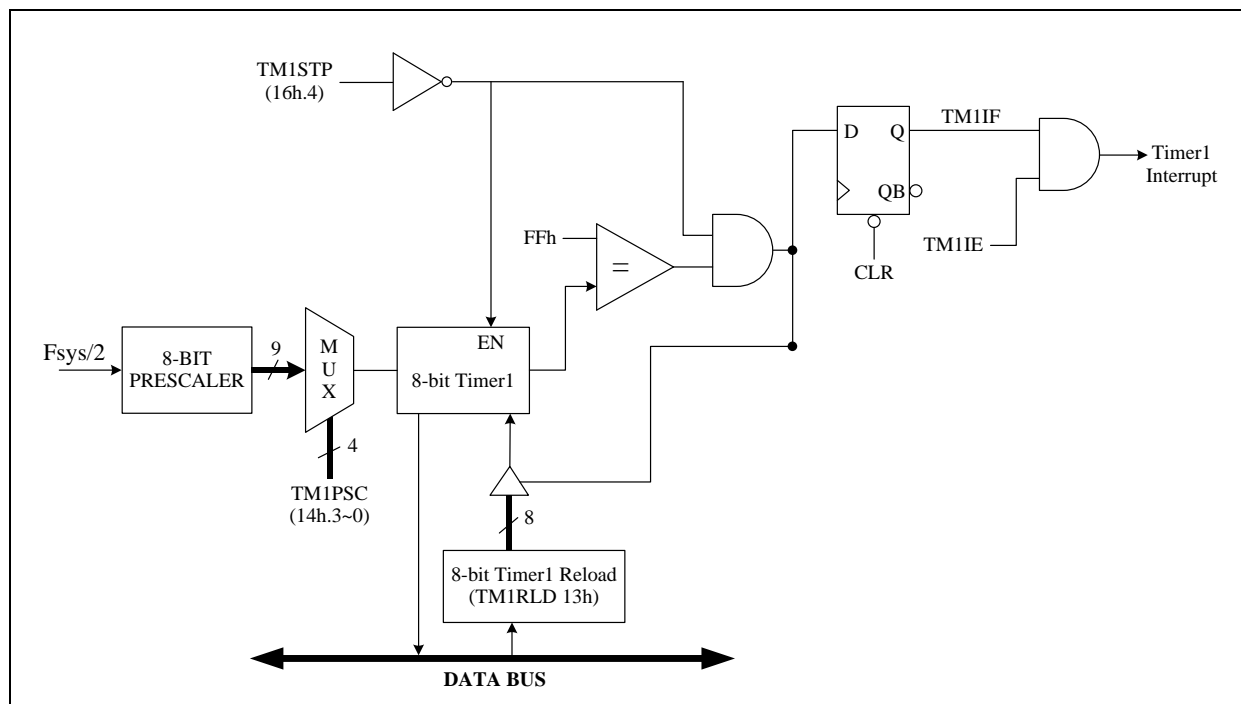
11h.3~0 **TM0PSC:** Timer0 prescaler. Timer0 prescaler clock source divided by
0000: /1 0001: /2 0010: /4 0011: /8
0100: /16 0101: /32 0110: /64 0111: /128
1000: /256 1001: /512 1010: /1024 1011: /2048
1100: /4096 1101: /8192 1110: /16384 1111: /32768

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

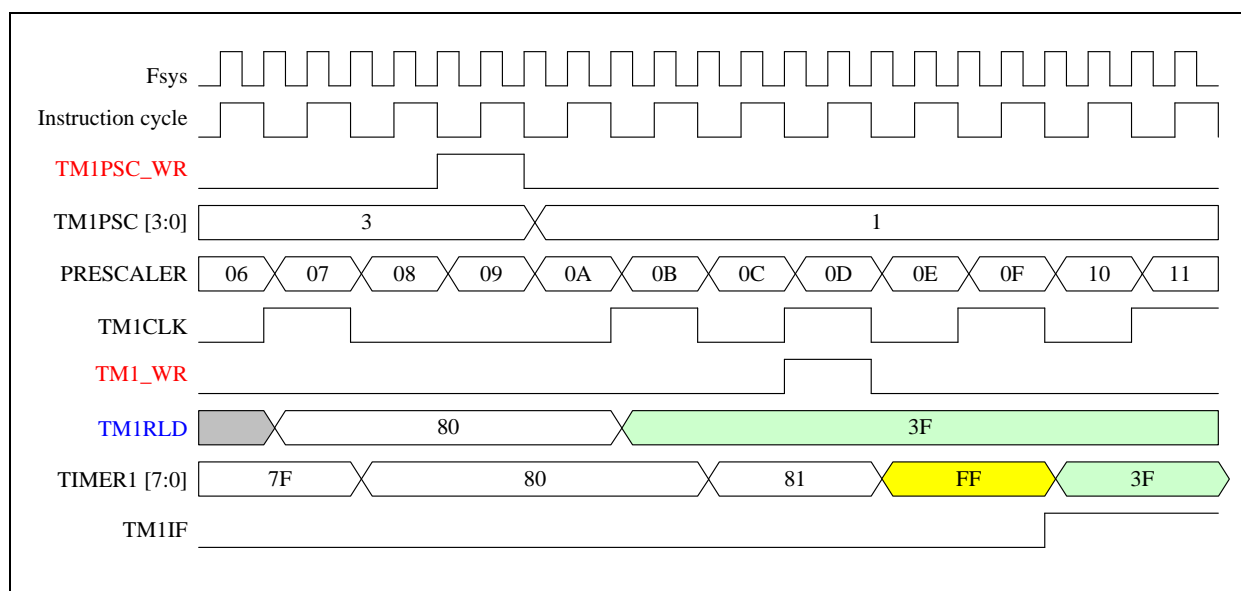
16h.3 **TM0STP:** Timer0 counter stop
0: Release 1: Stop counting

6.3 Timer1

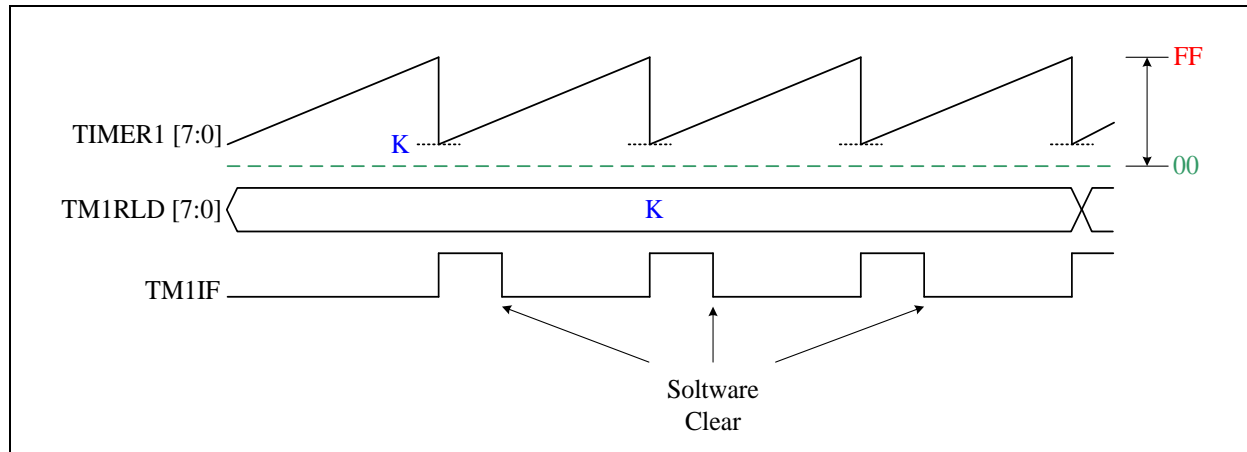
The Timer1 is an 8-bit wide register. It can be read or written as any other register. Besides, Timer1 increases itself periodically and automatically reloads a new "offset value" (TM1RLD) while it rolls over based on the pre-scaled instruction clock ($F_{sys}/2$). The Timer1 increase rate is determined by TM1PSC register. It generates Timer1 interrupt if the TM1IE bit is set. Timer1 can be stopped counting if the TM1STP bit is set.



Timer1 Block Diagram



Timer1 Timing Diagram



Timer1 Reload Diagram

◇ Example: CPU is running in SLOW mode, $F_{sys} = \text{Slow-clock} / \text{CPUPSC} = 70 \text{ KHz} / 2 = 35 \text{ KHz}$

; Setup Timer1 clock source and divider

MOVLW	00000 <u>010</u> B	; Set Slow-clock as system clock
MOVWX	CLKCTL	; CPUPSC = 10b, divided by 2
MOVLW	00000 <u>010</u> B	
MOVWX	TM1CTL	; TM1PSC = 0010b, divided by 8

; Setup Timer1 reload data

MOVLW	FFH	
MOVWX	TM1RLD	; Set Timer1 reload data = 255

; Setup Timer1

BSX	TM1STP	; Timer1 stops counting
CLR X	TM1	; Clear Timer1 content

; Enable Timer1 and interrupt function

MOVLW	11 <u>0</u> 11111B	
MOVWX	INTIF	; Clear Timer1 request interrupt flag
BSX	TM1IE	; Enable Timer1 interrupt function
BCX	TM1STP	; Enable Timer1 counting

Timer1 clock source is $F_{sys}/2 = 35 \text{ KHz} / 2 = 17.5 \text{ KHz}$, Timer1 divided by 8

Timer1 interrupt frequency = $17.5 \text{ KHz} / 2 / 8 / (256-255) = 1.09 \text{ Hz}$

0Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE	ADCIE	T2IE	TM1IE	TM0IE	WKTIE	INT2IE	INT1IE	INT0IE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
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	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

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	—	—	—	—	R/W	R/W	R/W	R/W
Reset	—	—	—	—	0	0	0	0

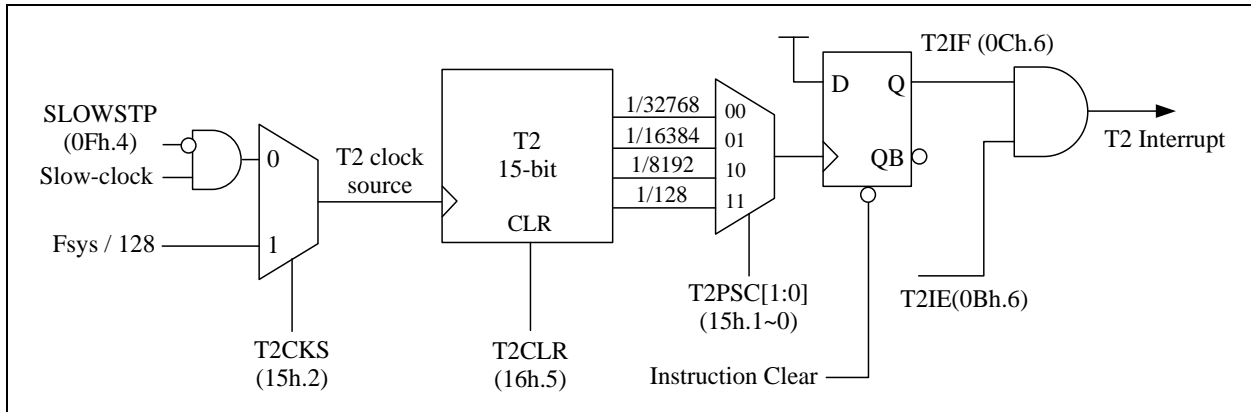
14h.3~0 **TM1PSC**: Timer1 prescaler. Timer1 clock source divided by
0000: Fsys/2 0001: Fsys/4 0010: Fsys/8 0011: Fsys/16
0100: Fsys/32 0101: Fsys/64 0110: Fsys/128 0111: Fsys/256
1xxx: Fsys/512

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

6.4 T2:15-bit Timer

The T2 is a 15-bit counter and the clock sources are from either $F_{sys}/128$ 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 is running at FAST mode, $F_{sys} = \text{Fast-clock} / \text{CPUPSC} = \text{FIRC } 8 \text{ MHz}$,

T2 clock source is $F_{sys}/128$

; Setup FIRC frequency

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

; Setup T2 clock source and divider

```
MOVLW    00000101B       ; T2CKS(15h.2) = 1, T2 clock source is Fsys/128
MOVWX    T2CTL            ; T2PSC(15h.1~0) = 1, divided by 16384
BSX      T2CLR            ; T2CLR = 1, clear T2 counter
```

; Enable T2 interrupt function

```
MOVLW    10111111B
MOVWX    INTIF            ; Clear T2 request interrupt flag
BSX      T2IE             ; Enable T2 interrupt function
BCX      T2CLR            ; T2CLR = 0, Enable T2 counting
```

T2 clock source is $F_{sys}/128 = 8 \text{ MHz}/128 = 62500 \text{ Hz}$, T2PSC = /16384

T2 frequency = $62500 \text{ Hz} / 16384 = 3.815 \text{ Hz}$

0Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE	ADCIE	T2IE	TM1IE	TM0IE	WKTIE	INT2IE	INT1IE	INT0IE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
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.
 1: Fsys/128 0: Slow-clock
 15h.1~0 **T2PSC:** T2 prescaler. “T2 clock source” divided by -
 00: 32768 01: 16384 10: 8192 11: 128

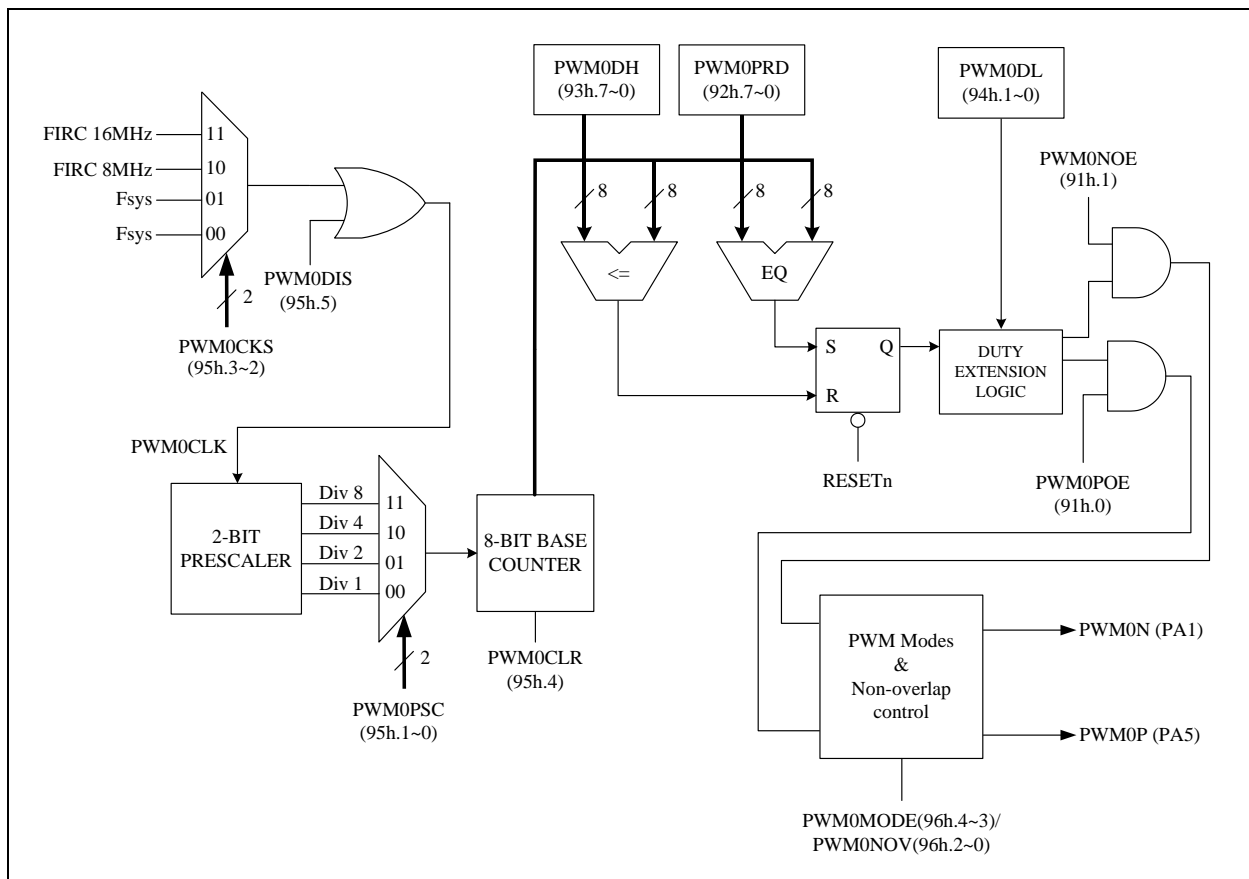
16h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MF016	LVDF	LVDEN	T2CLR	TM1STP	TM0STP	LVRSAV	LVDS	
R/W	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	—	0	1	0	0	1	0	1

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

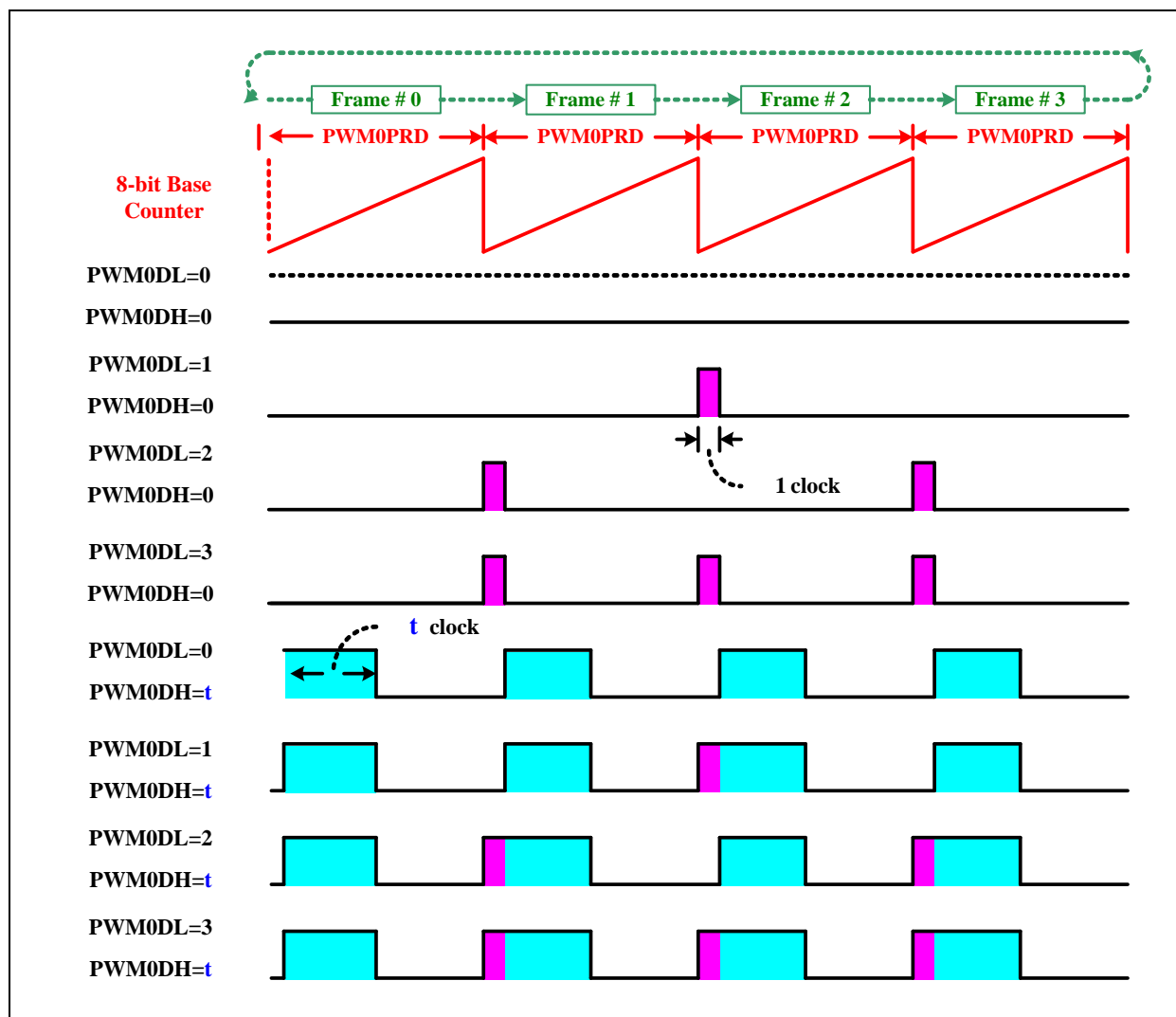
6.5 PWM0: (8+2) bits PWM

The PWM0 can generate various frequency waveforms with 1024 duty resolution based on PWM0CLK, which can select Fsys or FIRC 8MHz or FIRC 16MHz, decided by PWM0CKS (95h.3~2). A spread LSB technique allows PWM0 to run its frequency at “PWM0CLK divided by 256” instead of “PWM0CLK divided by 1024”, which means the PWM is 4 times faster than normal. 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 MSB of PWM duty register PWM0DH (93h.7~0). When the base counter rolls over, the 2-bit LSB of PWM duty register PWM0DL (94h.1~0) decides whether to set the PWM output signal high immediately or set it high after one clock cycle delay. **PWM0 clock is enabled and PWM0 is not hold after reset. (cf. the description of PWM0DIS and PWM0CLR)**

The PWM0 period can be set by writing period value to PWM0PRD register (92h.7~0). Note that changing the PWM0PRD will immediately change the PWM0PRD values, which are different from PWM0DH / PWM0DL which has buffer to update the duty at the end of current period. 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 PWM0PRD, 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 8+2 Timing Diagram

◇Example: CPU running at Fast mode, Fsys = FIRC 8 MHz

; Setup Pin mode

```

MOV LW    xxxx10xxB    ; PA1 Pin mode = Mode2
MOV WX    PAMODL        ; Mode2: CMOS output

MOV LW    xxxx10xxB    ; PA5 Pin mode = Mode2
MOV WX    PAMODH        ; Mode2: CMOS output

```

; Setup PWM0 clock prescaler

```

MOV LW    xx01 10 11B    ; 95h.4 = 1, PWM0 clear and hold
MOV WX    PWM0CTL        ; 95h.3~2 = 2, PWM0 clock source = FIRC 8MHz
                        ; 95h.1~0 = 3, PWM0 prescaler div 8

```

; Setup PWM0 mode & Non-overlap control

```

MOV LW    xxx00 000B    ; 96h.4~3 = 0, PWM0 mode = Mode0
MOV WX    PWM0CTL1        ; 96h.2~0 = 0, PWM0 non-overlap time = 0

MOV LW    7FH
MOV WX    PWM0PRD        ; Set PWM0 period = 7FH

MOV LW    xxxxxx00B
MOV WX    PWM0DL        ; Set PWM0DL duty = 00H

MOV LW    20H
MOV WX    PWM0DH        ; Set PWM0DH duty = 20H

MOV LW    xxxxxx11B    ; 91h.1 = 1, Enable PWM0N output to PA1
MOV WX    PWMOE        ; 91h.0 = 1, Enable PWM0P output to PA5

BCX      PWM0CLR        ; 95h.4 = 0, release PWM0 clear and hold

```

Example:

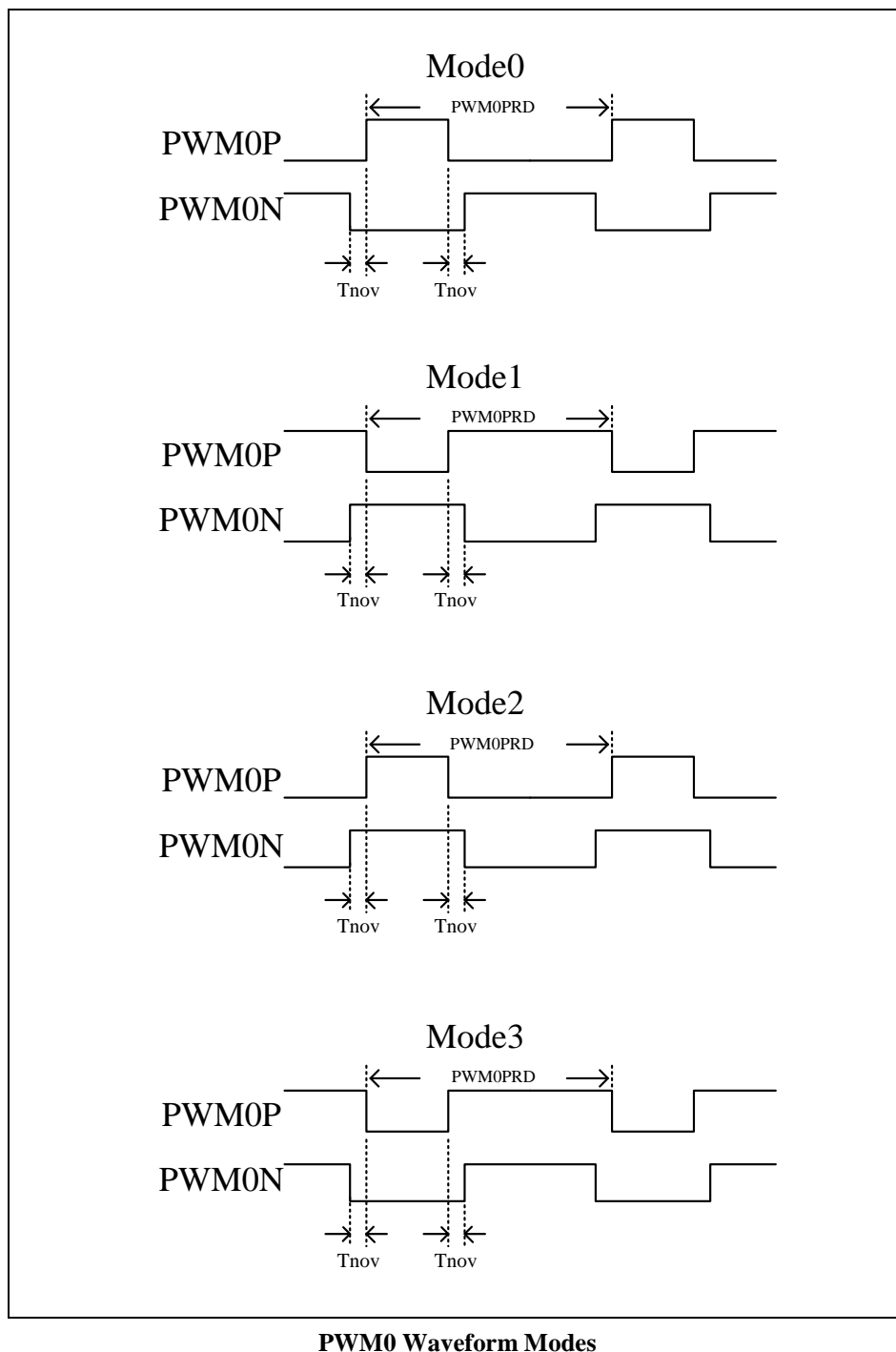
PWM0 clock source = FIRC 8M, PWM0PSC = div 8, PWM0PRD = 7FH,

PWM0DL = 00H, PWM0DH = 20H

PWM0 output frequency = 8 MHz / 8 / (PWM0PRD+1) = 8 MHz / 8 / 128 = 7.8125 KHz.

PWM0P output duty = 32:128 = 25 %.

PWM0 can be output via PWM0P and PWM0N with four different modes. The edges of the PWM pulse can be separated with 6 different time non-overlap clocks intervals (T_{nov}), 0s, 4 PWM0CLKs, 5 PWM0CLKs, 6 PWM0CLKs, 7 PWM0CLKs, and 8 PWM0CLKs which are selected by PWM0NOV (96h.2~0). The default output form is Mode0. The waveforms of the four output modes are shown below.



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

91h.1 **PWM0NOE:** PWM0N output to PA1 enable

0: disable

1: enable, PWM0N output to PA1

91h.0 **PWM0POE:** PWM0P output to PA5 enable

0: disable

1: enable, PWM0P output to PA5

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

92h.7~0 **PWM0PRD:** PWM0 period data

93h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM0DH	PWM0DH							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

93h.7~0 **PWM0DH:** PWM0 duty MSB 8bit

94h	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM0DL	—	—	—	—	—	—	PWM0DL	
R/W	—	—	—	—	—	—	R/W	R/W
Reset	—	—	—	—	—	—	0	0

94h.1~0 **PWM0DL:** PWM0 duty LSB 2bit

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

95h.5 **PWM0DIS:** PWM0 clock disable

0: clock enable

1: clock disable

95h.4 **PWM0CLR:** PWM0 clear and hold

0: PWM0 enable

1: PWM0 clear and hold

95h.3~2 **PWM0CKS:** PWM0 clock source select

0x: Fsys

10: FIRC 8MHz

11: FIRC 16MHz

95h.1~0 **PWM0PSC:** PWM0 clock source prescaler

00: divided by 1

01: divided by 2

10: divided by 4

11: divided by 8

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

96h.4~3 **PWM0MODE:** PWM0 differential output mode

00: Mode 0

01: Mode 1

10: Mode 2

11: Mode 3

96h.2~0 **PWM0NOV:** PWM0 non-overlap control

000: original PWM0

001: non-overlap 4 PWM0CLKs

010: non-overlap 5 PWM0CLKs

011: non-overlap 6 PWM0CLKs

100: non-overlap 7 PWM0CLKs

101: non-overlap 8 PWM0CLKs

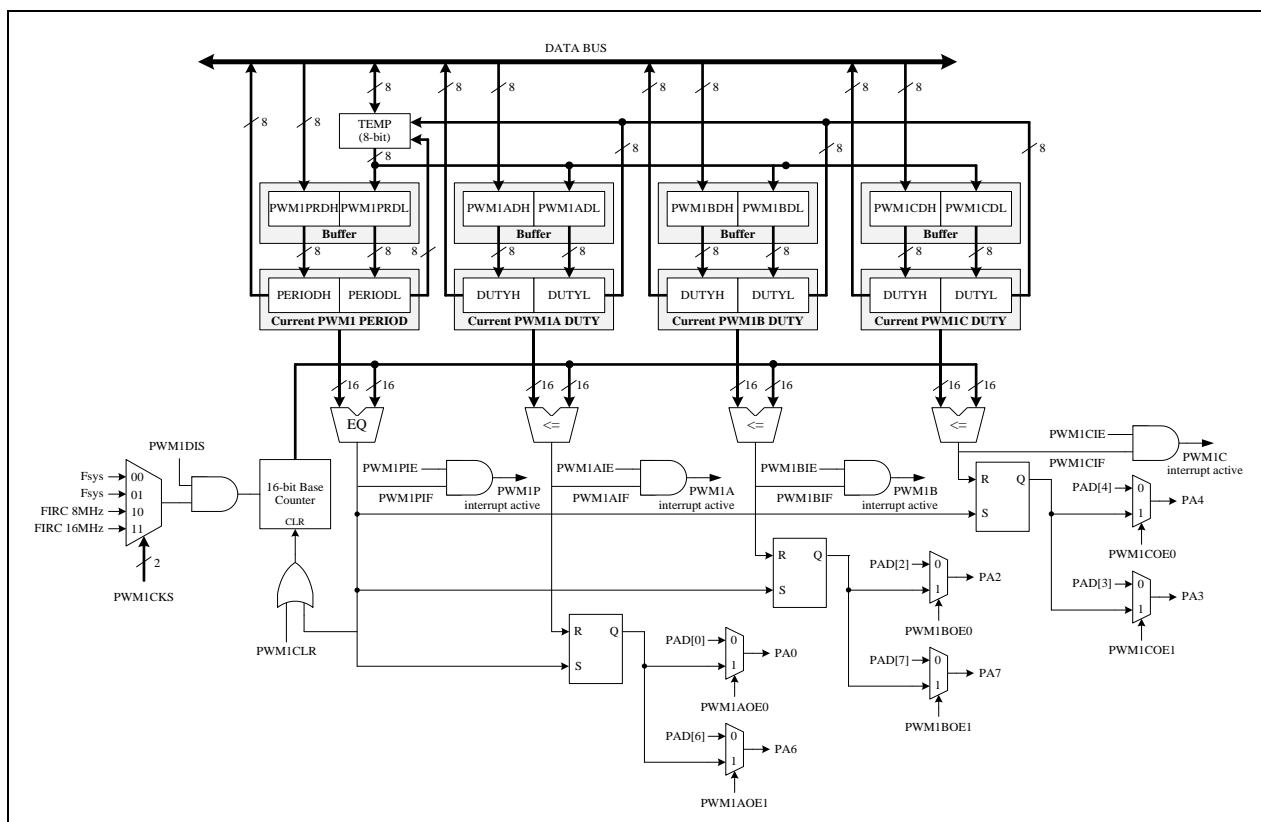
6.6 PWM1A / PWM1B / PWM1C: 16 bits PWMs

PWM1A, PWM1B and PWM1C are 3 PWMs which have independent duty and common period. The PWM1 can generate varies frequency waveform with 65536 duty resolution on the basis of the PWM1 clock. The PWM clock can select Fsys, FIRC 8 MHz or 16 MHz, decided by PWMCKS (97h.3~2). The PWM1 clock also can be stop by setting PWM1DIS (97h.5) bit. **PWM1 clock is enabled and PWM1 is not hold after reset. (cf. the description of PWM1DIS and PWM1CLR)**

The pin mode SFR controls the PWM output waveform format. Mode1 makes the PWM open drain output and Mode2 makes the PWM CMOS push-pull output. (see section 5)

The 16-bit PWM1PRD, PWM1AD, PWM1BD, PWM1CD registers all have a low byte and high byte structure. The high bytes can be directly accessed, but the low bytes can only be accessed via an internal 8-bit buffer, reading or writing to these register pairs must be carried out in a specific way. The important point to notes is that data transfer to and from the 8-bit buffer and its related low byte only takes place when write or read operation to its corresponding high bytes is executed. Briefly speaking, **write low byte first and then high byte; read high byte first and then low byte.**

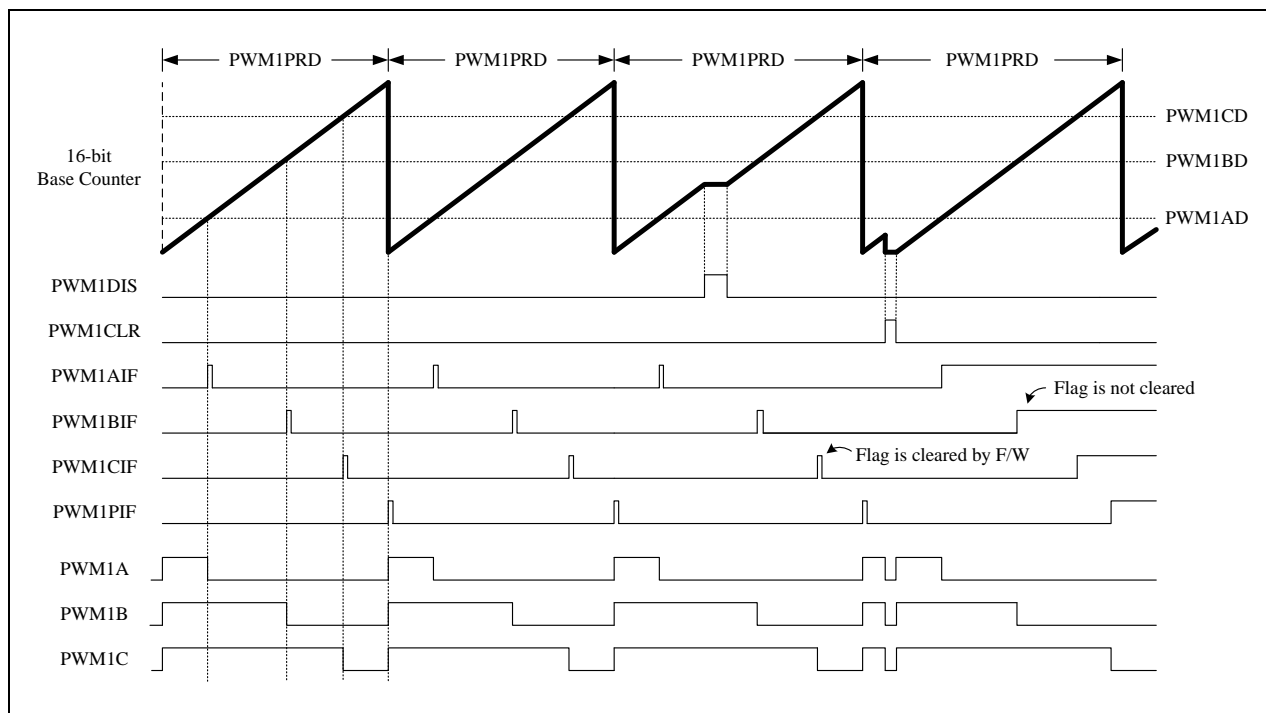
The PWM1 structure is shown as follow. When PWM1CLR (97h.4) bit is set, the PWM1 will be cleared and held, otherwise the PWM1 is running. The PWM1 duty cycle can be changed by writing to PWM1DH and PWM1DL. The PWM1 output signal resets to a low level whenever the 16-bit base counter matches the 16-bit PWM1 duty register {PWM1DH, PWM1DL}. The PWM1 period can be set by writing the period value to the PWM1PRDH and PWM1PRDL registers. After writing the PWM1D or PWM1PRD register, the new values will immediately save to their own buffer. H/W will update these values at the end of current period or while PWM1 is cleared.



PWM1 Block Diagram

PWM1A, PWM1B and PWM1C have a corresponding interrupt flag PWM1AIF (0Eh.0), PWM1BIF (0Eh.1) and PWM1CIF (0Eh.2), and those interrupt flags are generated while PWM1 16-bit base counter count to the setting duties. The PWM1 also has a corresponding period interrupt flag PWM1PIF (0Eh.3), and an interrupt flag is generated at the end of the period. Setting their corresponding interrupt enable bit PWM1AIE (0Dh.0), PWM1BIE (0Dh.1), PWM1CIE (0Dh.2) and PWM1PIE (0Dh.3) can generate the corresponding interrupt.

The PWM1xOEn (91h.7~2) bits are used to select the related PWM output to I/O. No matter PWMOE is set or not, PWM1 can keep running in the background as a timer.



PWM1 Timing Diagram

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

0Dh.3 **PWM1PIE:** PWM1 period interrupt enable

0: disable

1: enable

0Dh.2 **PWM1CIE:** PWM1C duty interrupt enable

0: disable

1: enable

0Dh.1 **PWM1BIE:** PWM1B duty interrupt enable

0: disable

1: enable

0Dh.0 **PWM1AIE:** PWM1A duty interrupt enable

0: disable

1: enable

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

- 0Eh.3 **PWM1PIF:** PWM1 period interrupt event pending flag
This bit is set by H/W after PWM1 counter count to the set period, write 0 to this bit will clear this flag
- 0Eh.2 **PWM1CIF:** PWM1C duty interrupt event pending flag
This bit is set by H/W after PWM1 counter count to the set PWM1C duty, write 0 to this bit will clear this flag
- 0Eh.1 **PWM1BIF:** PWM1B duty interrupt event pending flag
This bit is set by H/W after PWM1 counter count to the set PWM1B duty, write 0 to this bit will clear this flag
- 0Eh.0 **PWM1AIF:** PWM1A duty interrupt event pending flag
This bit is set by H/W after PWM1 counter count to the set PWM1A duty, write 0 to this bit will clear this flag

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

- 91h.7 **PWM1COE1:** PWM1C output to PA3 enable
0: disable 1: enable, PWM1C output to PA3
- 91h.6 **PWM1COE0:** PWM1C output to PA4 enable
0: disable 1: enable, PWM1C output to PA4
- 91h.5 **PWM1BOE1:** PWM1B output to PA7 enable
0: disable 1: enable, PWM1B output to PA7
- 91h.4 **PWM1BOE0:** PWM1B output to PA2 enable
0: disable 1: enable, PWM1B output to PA2
- 91h.3 **PWM1AOE1:** PWM1A output to PA6 enable
0: disable 1: enable, PWM1A output to PA6
- 91h.2 **PWM1AOE0:** PWM1A output to PA0 enable
0: disable 1: enable, PWM1A output to PA0

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

- 97h.5 **PWM1DIS:** PWM1 clock disable
0: clock enable
1: clock disable
- 97h.4 **PWM1CLR:** PWM1 clear and hold
0: PWM1 enable
1: PWM1 clear and hold
- 97h.3~2 **PWM1CKS:** PWM1 clock source select
0x: Fsys
10: FIRC 8MHz
11: FIRC 16MHz

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

98h.7~0 **PWM1PRDH:** PWM1 (PWM1A / PWM1B / PWM1C) period data MSB 8bit

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

99h.7~0 **PWM1PRDL:** PWM1 (PWM1A / PWM1B / PWM1C) period data LSB 8bit

About 16-bit data write: Write PWM1PRDL first, then PWM1PRDH

About 16-bit data read: Read PWM1PRDH first, then PWM1PRDL

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

9Ah.7~0 **PWM1ADH:** PWM1A duty MSB 8bit

9Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM1ADL	PWM1ADL							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

9Bh.7~0 **PWM1ADL:** PWM1A duty LSB 8bit

About 16-bit data write: Write PWM1ADL first, then PWM1ADH

About 16-bit data read: Read PWM1ADH first, then PWM1ADL

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

9Ch.7~0 **PWM1BDH:** PWM1B duty MSB 8bit

9Dh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM1BDL	PWM1BDL							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

9Dh.7~0 **PWM1BDL:** PWM1B duty LSB 8bit

About 16-bit data write: Write PWM1BDL first, then PWM1BDH

About 16-bit data read: Read PWM1BDH first, then PWM1BDL

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

9Eh.7~0 **PWM1CDH:** PWM1C duty MSB 8bit

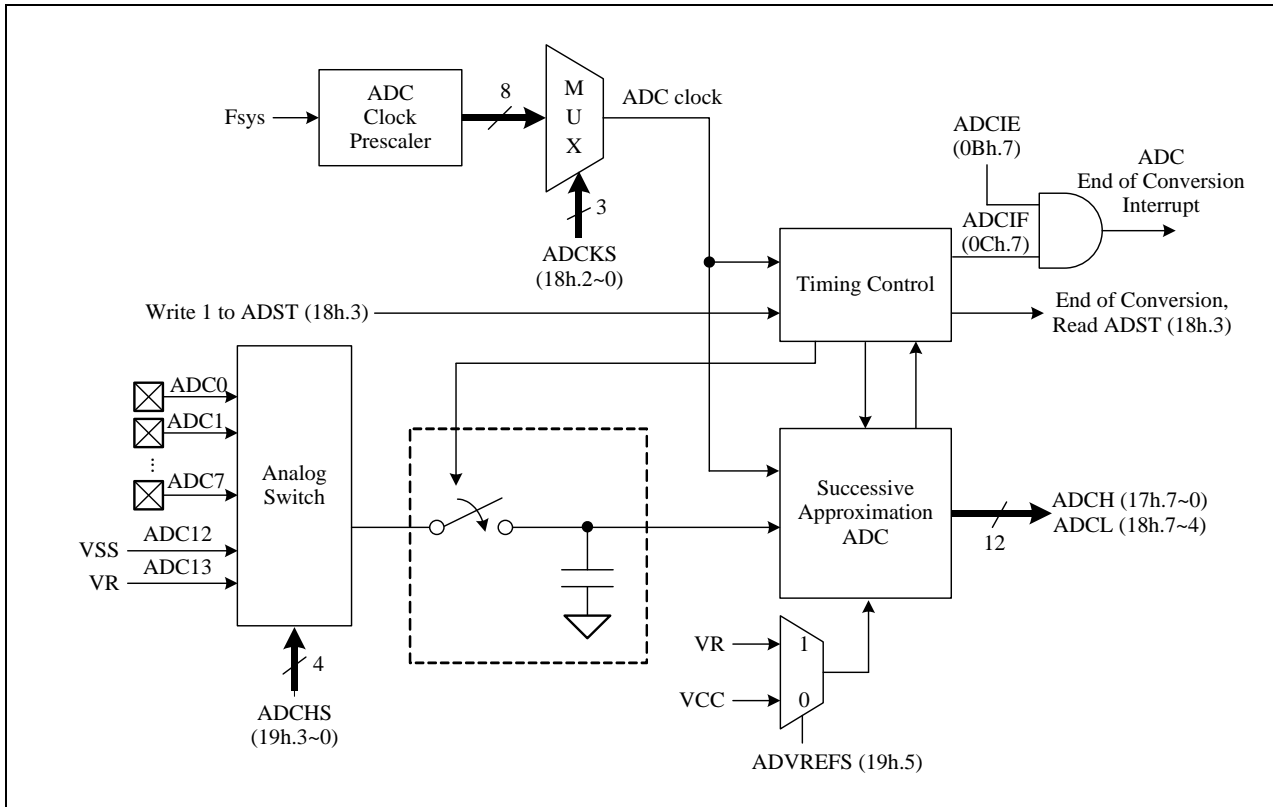
9Fh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PWM1CDL	PWM1CDL							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

9Fh.7~0 **PWM1CDL:** PWM1C duty LSB 8bit

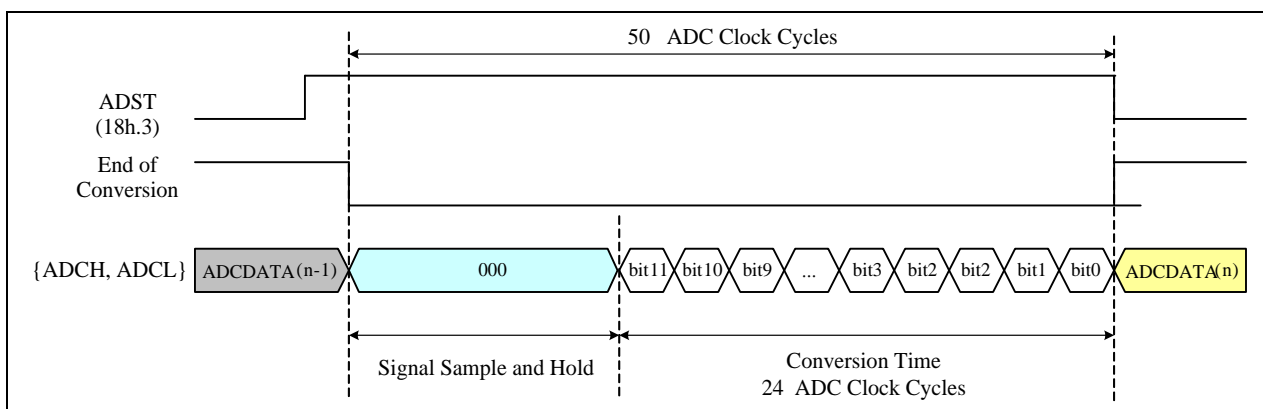
About 16-bit data write: Write PWM1CDL first, then PWM1CDH

About 16-bit data read: Read PWM1CDH first, then PWM1CDL

6.7 Analog-to-Digital Converter



The 12-bit ADC (Analog to Digital Converter) consists of a 16-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 **PAMODH** and **PAMODL** 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. Besides, there are some reference input channel can be selected, **ADC12** is VSS and **ADC13** is VR=3V. ADC reference voltage can select VCC or VR by **ADCVREFS** (19h.5~4).



Example:

[CPU running at FAST mode , Fsys = FIRC 8MHz]
ADC clock frequency = 1 MHz, ADC channel = ADC2 (PA2).

◇ Example:

```

MOVLW    00000111B      ; Fsys = 8 MHz
MOVWX    CLKCTL           ;

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

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

MOVLW    00110010B      ; 19h.5 = 0, ADC reference voltage select VCC
MOVWX    MF019           ; 19h.3~0 = 2, ADC input channel select ADC2

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

```

WAIT_ADC:

```

BTXSC    ADST             ; Wait ADC conversion finish.
GOTO     WAIT_ADC

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

```

0Bh/8Bh/10Bh/18Bh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTIE	ADCIE	T2IE	TM1IE	TM0IE	WKTIE	INT2IE	INT1IE	INT0IE
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

INTIE.7 ADCIE: ADC 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

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 ADCH: 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	–	–	ADVREFS	VREN	ADCHS			
R/W	–	–	R/W	R/W	R/W			
Reset	–	–	1	1	0	0	0	0

19h.5 **ADVREFS:** ADC reference voltage select

0: VCC

1: VR

19h.4 **VREN:** Internal reference voltage VR enable (3V ±1.2% @25°C, VCC=3V~5V)

0: Disable

1: Enable and Auto disable in STOP/IDLE mode

19h.3~0 **ADCHS:** ADC channel select

0000: ADC0 (PA0) 1000: Reserved

0001: ADC1 (PA1) 1001: Reserved

0010: ADC2 (PA2) 1010: Reserved

0011: ADC3 (PA3) 1011: Reserved

0100: ADC4 (PA4) 1100: VSS

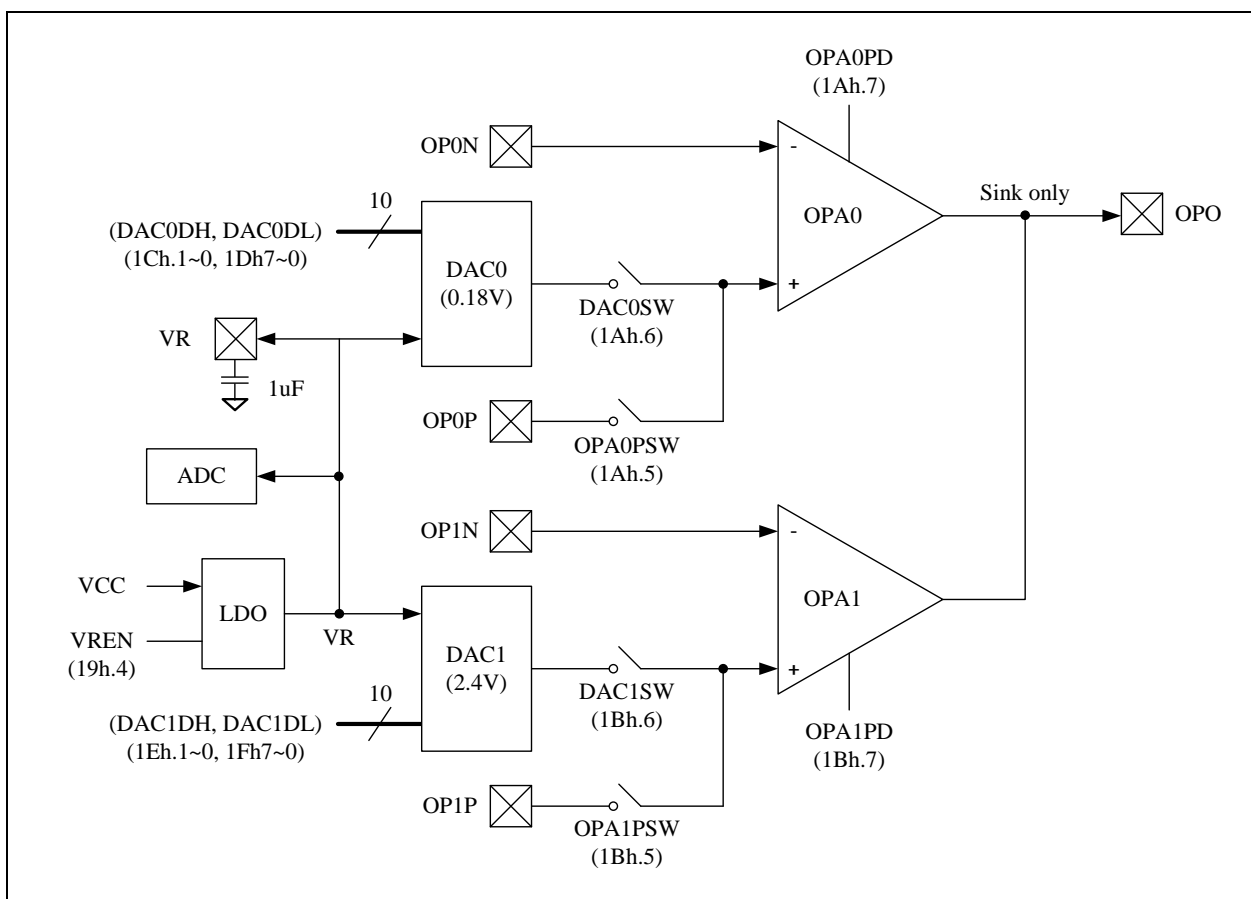
0101: ADC5 (PA5) 1101: VR

0110: ADC6 (PA6) 1110: Reserved

0111: ADC7 (PA7) 1111: Reserved

6.8 Battery Charge Module (BCM)

The TM56F8225 contains a Battery Charging Module (BCM). The module consists of two 10-bit Digital-to-Analog converters (DAC) and two operational amplifiers (OPA). The DAC reference voltage comes from the internal reference voltage VR ($VR = 3V \pm 1.2\% @ 25^{\circ}C$, $VCC = 3V \sim 5V$). The voltage of VR can be turned off by setting VREN = 0 (19h.4). After power-on, the default voltage output of DAC0 is about 0.18V, and DAC1 is about 2.4V. They are connected to the positive input of OPA0 and OPA1, respectively. In addition, the OPA0 and OPA1 are always enabled after power-on. The Battery Charging Module block diagram is shown below.



BCM Block Diagram

The 10-bit DAC0D and DAC1D registers all have a low byte and high byte structure. When in read operation, the high bytes and the low bytes can be accessed directly. When in write operation, the high bytes can be accessed directly, but the low bytes can only be accessed via an internal 8-bit buffer. Writing to these register pairs must be carried out in a specific way. The important point to notes is that data transfer to and from the 8-bit buffer and its related low byte only takes place when write operation to its corresponding high bytes is executed. Briefly speaking, **write low byte first and then high byte; read high byte and low byte directly.**

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

- 1Ah.7 **OPA0PD:** OPA0 Power Down
 0: OPA0 on
 1: OPA0 off
- 1Ah.6 **DAC0SW:** DAC0 to OPA0 non-inverting input switch on
 0: switch off
 1: switch on
- 1Ah.5 **OPA0PSW:** OPA0P to OPA0 non-inverting input switch on
 0: switch off
 1: switch on
- 1Ah.4~0 **OPA0ADJ:** OPA0 input offset voltage calibration control

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

- 1Bh.7 **OPA1PD:** OPA1 Power Down
 0: OPA1 on
 1: OPA1 off
- 1Bh.6 **DAC1SW:** DAC1 to OPA1 non-inverting input switch on
 0: switch off
 1: switch on
- 1Bh.5 **OPA1PSW:** OPA1P to OPA1 non-inverting input switch on
 0: switch off
 1: switch on
- 1Bh.4~0 **OPA1ADJ:** OPA1 input offset voltage calibration control

1Ch	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DAC0DH	—	—	—	—	—	—	DAC0DH	
R/W	—	—	—	—	—	—	R/W	R/W
Reset	—	—	—	—	—	—	0	0

- 1Ch.1~0 **DAC0DH:** DAC0 input data MSB[9~8]

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

- 1Dh.7~0 **DAC0DL:** DAC0 input data LSB[7~0]
 write DAC0DL first, then DAC0DH
 read DAC0DH and DAC0DL directly

1Eh	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DAC1DH	—	—	—	—	—	—	DAC1DH	
R/W	—	—	—	—	—	—	R/W	R/W
Reset	—	—	—	—	—	—	1	1

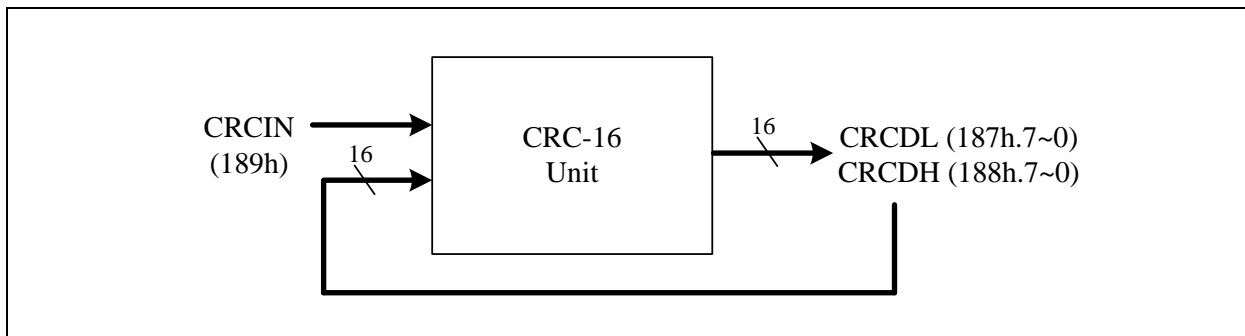
- 1Eh.1~0 **DAC1DH:** DAC1 input data MSB[9~8]

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

1Fh.7~0 **DAC1DL:** DAC1 input data LSB[7~0]
 write DAC1DL first, then DAC1DH
 read DAC1DH and DAC1DL directly

6.9 Cyclic Redundancy Check (CRC)

The chip supports an integrated 16-bit Cyclic Redundancy Check function. The Cyclic Redundancy Check (CRC) calculation unit is an error detection technique test algorithm and uses to verify data transmission or storage data correctness. The CRC calculation takes a 8-bit data stream or a block of data as input and generates a 16-bit output remainder. The data stream is calculated by the same generator polynomial.



CRC16 Block Diagram

The CRC generator provides the 16-bit CRC result calculation based on the CRC-16-IBM polynomial. In this CRC generator, there is only one polynomial available for the numeric values calculation. It can't support the 16-bit CRC calculations based on any other polynomials. Each write operation to the CRCIN register creates a combination of the previous CRC value stored in the CRCDH and CRCDL registers. It will take one MCU instruction cycle to calculate.

CRC-16-IBM (Modbus) Polynomial representation: $X^{16} + X^{15} + X^2 + 1$

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

187h.7~0 **CRCDL:** 16-bit CRC checksum data bit 7~0

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

188h.7~0 **CRCDL:** 16-bit CRC checksum data bit 15~8

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

189h.7~0 **CRCIN:** write this register to start CRC calculation

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 (assembly keep this bit is 0)
RP0	03.5	R/W	0	Register Bank Select bit 0 (assembly keep this bit is 0)
TO	03.4	R	0	WDT timeout flag, cleared by PWRST, ‘SLEEP’ or ‘CLRWDI’ instruction
PD	03.3	R	0	Power down flag, set by ‘SLEEP’, cleared by ‘CLRWDI’ 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
PCLATH (0Ah/8Ah/10Ah/18Ah)		Function related to: PROGRAM COUNT		
PCLATH	0A.2~0	R/W	0	Write Buffer for the upper 3 bits of the Program Counter
INTIE (0Bh/8Bh/10Bh/18Bh)		Function related to: Interrupt Enable		
ADCIE	0B.7	R/W	0	ADC interrupt enable 0: disable 1: enable
T2IE	0B.6	R/W	0	T2 interrupt enable 0: disable 1: enable
TM1IE	0B.5	R/W	0	Timer1 interrupt enable 0: disable 1: enable
TM0IE	0B.4	R/W	0	Timer0 interrupt enable 0: disable 1: enable
WKTIE	0B.3	R/W	0	Wakeup Timer interrupt enable, set 0 to clear & disable WKT timer 0: disable 1: enable
INT2IE	0B.2	R/W	0	INT2 pin (PA7) interrupt enable 0: disable 1: enable
INT1IE	0B.1	R/W	0	INT1 pin (PA1) interrupt enable 0: disable 1: enable
INT0IE	0B.0	R/W	0	INT0 pin (PA6 or PA2) interrupt enable 0: disable 1: enable

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 (PA1) 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 (PA6) 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
INTIE1 (0Dh)		Function related to: Interrupt Enable		
PWM1PIE	0D.3	R/W	0	PWM1 period interrupt enable 0: disable 1: enable
PWM1CIE	0D.2	R/W	0	PWM1C duty interrupt enable 0: disable 1: enable
PWM1BIE	0D.1	R/W	0	PWM1B duty interrupt enable 0: disable 1: enable
PWM1AIE	0D.0	R/W	0	PWM1A duty interrupt enable 0: disable 1: enable
INTIF1 (0Eh)		Function related to: Interrupt Flag		
PWM1PIF	0E.3	R	-	PWM1 period interrupt event pending flag, set by H/W while PWM1 counter count to the set period
		W	0	write 0: clear this flag; write 1: no action
PWM1CIF	0E.2	R	-	PWM1C duty interrupt event pending flag, set by H/W while PWM1 counter count to the set PWM1C duty
		W	0	write 0: clear this flag; write 1: no action
PWM1BIF	0E.1	R	-	PWM1B duty interrupt event pending flag, set by H/W while PWM1 counter count to the set PWM1B duty
		W	0	write 0: clear this flag; write 1: no action
PWM1AIF	0E.0	R	-	PWM1A duty interrupt event pending flag, set by H/W while PWM1 counter count to the set PWM1A duty
		W	0	write 0: clear this flag; write 1: no action
CLKCTL (0Fh)		Function related to: Fsys		
SLOWSTP	0F.4	R/W	0	Stop Slow-clock in Stop Mode 0: no Stop 1: Stop
FASTSTP	0F.3	R/W	1	Stop Fast-clock 0:no Stop 1:Stop
CPUCKS	0F.2	R/W	0	Select Fast-clock 0: Fsys=Slow-clock 1: Fsys=Fast-clock
CPUPSC	0F.1~0	R/W	11	Fsys Prescaler, 00: div 8 01: div 4 10: div 2 11: div 1

TM0RLD (10h)				Function related to: TM0
TM0RLD	10.7~0	R/W	0	Timer0 reload Data
TM0CTL (11h)				Function related to: TM0
TM0EDG	11.5	R/W	0	Timer0 prescaler counting edge for TM0CKI pin 0: rising edge 1: falling edge
TM0CKS	11.4	R/W	0	Timer0 prescaler clock source 0: Fsys/2 1: TM0CKI pin (PA6 pin)
TM0PSC	11.3~0	R/W	0	Timer0 prescaler. Timer0 prescaler clock source divided by 0000: /1 0100: /16 1000: /256 1100: /4096 0001: /2 0101: /32 1001: /512 1101: /8192 0010: /4 0110: /64 1010: /1024 1110: /16384 0011: /8 0111: /128 1011: /2048 1111: /32768
TM1 (12h)				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 0000: Fsys/2 0100: Fsys/32 1xxx: Fsys/512 0001: Fsys/4 0101: Fsys/64 0010: Fsys/8 0110: Fsys/128 0011: Fsys/16 0111: Fsys/256
T2CTL (15h)				Function related to: T2
T2CKS	15.2	R/W	0	T2 clock source 0: Slow-clock 1: Fsys/128
T2PSC	15.1~0	R/W	0	T2 prescaler. T2 clock source divided by - 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=2.2V only) 0: disable 1: enable
T2CLR	16.5	R/W	1	T2 counter clear 0: Release 1: Stop counting
TM1STP	16.4	R/W	0	Timer1 counter stop 0: Release 1: Stop counting
TM0STP	16.3	R/W	0	Timer0 counter stop 0: Release 1: Stop counting
LVRSAV	16.2	R/W	1	LVR/LVD power save 0: LVR/LVD enable in in STOP/IDLE mode 1: LVR/LVD auto power off in STOP/IDLE mode
LVDS	16.1~0	R/W	01	LVD select (when LVR=2.2V) 00: 3.6V 01: 2.8V 1x: 4.2V

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. 0: H/W clear after end of conversion 1: ADC start conversion
ADCKS	18.2~0	R/W	0	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
MF019 (19h)				Function related to: ADC
ADVREFS	19.5	R/W	1	ADC reference voltage select 0: VCC 1: VR
VREN	19.4	R/W	1	Internal reference voltage VR enable (3V ±1.2% @25°C, VCC=3V~5V) 0: Disable 1: Enable and Auto disable in STOP/IDLE mode
ADCHS	19.3~0	R/W	0	ADC channel select 0000: ADC0 (PA0) 0110: ADC6 (PA6) 1100: VSS 0001: ADC1 (PA1) 0111: ADC7 (PA7) 1101: VR 0010: ADC2 (PA2) 1000: Reserved 1110: Reserved 0011: ADC3 (PA3) 1001: Reserved 1111: Reserved 0100: ADC4 (PA4) 1010: Reserved 0101: ADC5 (PA5) 1011: Reserved
OPA0CTL (1Ah)				Function related to: OPA
OPA0PD	1A.7	R/W	0	OPA0 Power Down 0: OPA0 on 1: OPA0 off
DAC0SW	1A.6	R/W	1	DAC0 to OPA0 non-inverting input switch on 0: switch off 1: switch on
OPA0PSW	1A.5	R/W	1	OPA0P to OPA0 non-inverting input switch on 0: switch off 1: switch on
OPA0ADJ	1A.4~0	R/W	CFG	OPA0 input offset voltage calibration control
OPA1CTL (1Bh)				Function related to: OPA
OPA1PD	1B.7	R/W	0	OPA1 Power Down 0: OPA1 on 1: OPA1 off
DAC1SW	1B.6	R/W	1	DAC1 to OPA1 non-inverting input switch on 0: switch off 1: switch on
OPA1PSW	1B.5	R/W	1	OPA1P to OPA1 non-inverting input switch on 0: switch off 1: switch on
OPA1ADJ	1B.4~0	R/W	CFG	OPA1 input offset voltage calibration control
DAC0DH (1Ch)				Function related to: DAC
DAC0DH	1C.1~0	R/W	00	DAC0 input data MSB[9~8]
DAC0DL (1Dh)				Function related to: DAC
DAC0DL	1D.7~0	R/W	40	DAC0 input data LSB[7~0], write DAC0DL first, then DAC0DH
DAC1DH (1Eh)				Function related to: DAC
DAC1DH	1E.1~0	R/W	03	DAC1 input data MSB[9~8]
DAC1DL (1Fh)				Function related to: DAC
DAC1DL	1F.7~0	R/W	39	DAC1 input data LSB[7~0], write DAC1DL first, then DAC1DH
User Data Memory				
RAM	20~6F	R/W	-	RAM Bank0 area (80 Bytes)
RAM	70~7F	R/W	-	RAM common area (16 Bytes)

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 0:disable 1: enable
INT0EDG	81.6	R/W	0	INT0 pin edge interrupt event 0: falling edge to trigger 1: rising edge to trigger
INT1EDG	81.5	R/W	0	INT1 pin edge interrupt event 0: falling edge to trigger 1: rising edge to trigger
INT0SEL	81.4	R/W	0	INT0 pin select, 0: PA6 1: PA2
WDTOSC	81.3~2	R/W	11	WDT pre-scale selections: 00: 128mS 01: 256mS 10: 1024mS 11: 2048mS
WKTOSC	81.1~0	R/W	11	WKT pre-scale selections: 00: 16mS 01: 32mS 10: 64mS 11: 128mS
PAMODH (8Ch)				Function related to: Port A
PA7MOD	8C.7~6	R/W	00	PA7~PA4 I/O mode control 00: Mode0 01: Mode1 10: Mode2 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 00: Mode0 01: Mode1 10: Mode2 11: Mode3
PA2MOD	8D.5~4	R/W	01	
PA1MOD	8D.3~2	R/W	01	
PA0MOD	8D.1~0	R/W	01	
PWMOE (91h)				Function related to: PWM0 / PWM1A / PWM1B / PWM1C
PWM1COE1	91.7	R/W	0	PWM1C output to PA3 enable 0: disable 1:enable
PWM1COE0	91.6	R/W	0	PWM1C output to PA4 enable 0: disable 1:enable
PWM1BOE1	91.5	R/W	0	PWM1B output to PA7 enable 0: disable 1:enable
PWM1BOE0	91.4	R/W	0	PWM1B output to PA2 enable 0: disable 1:enable
PWM1AOE1	91.3	R/W	0	PWM1A output to PA6 enable 0: disable 1:enable
PWM1AOE0	91.2	R/W	0	PWM1A output to PA0 enable 0: disable 1:enable
PWM0NOE	91.1	R/W	0	PWM0N output to PA1 enable 0: disable 1:enable
PWM0POE	91.0	R/W	0	PWM0P output to PA5 enable 0: disable 1:enable
PWM0PRD (92h)				Function related to: PWM0
PWM0PRD	92.7~0	R/W	FF	PWM0 period data
PWM0DH (93h)				Function related to: PWM0
PWM0DH	93.7~0	R/W	00	PWM0 Duty MSB 8bit

PWM0DL (94h)				Function related to: PWM0
PWM0DL	94.1~0	R/W	0	PWM0 Duty LSB 2bit
PWM0CTL (95h)				Function related to: PWM0
PWM0DIS	95.5	R/W	0	PWM0 clock disable 0: clock enable 1: clock disable
PWM0CLR	95.4	R/W	0	PWM0 clear and hold 0: PWM0 enable 1: PWM0 clear and hold
PWM0CKS	95.3~2	R/W	0	PWM0 clock source select 0x: Fsys 10: FIRC 8MHz 11: FIRC 16MHz
PWM0PSC	95.1~0	R/W	0	PWM0 clock source prescaler 00: div1 01: div2 10: div4 11: div8
PWM0CTL1 (96h)				Function related to: PWM0
PWM0MODE	96.4~3	R/W	0	PWM0 differential output mode 00: Mode 0 01: Mode 1 10: Mode 2 11: Mode 3
PWM0NOV	96.2~0	R/W	0	PWM0 non-overlap control 000: original PWM0 001: non-overlap 4 PWM0CLKs 010: non-overlap 5 PWM0CLKs 011: non-overlap 6 PWM0CLKs 100: non-overlap 7 PWM0CLKs 101: non-overlap 8 PWM0CLKs
PWM1CTL (97h)				Function related to: PWM1A / PWM1B / PWM1C
PWM1DIS	97.5	R/W	0	PWM1 (PWM1A/PWM1B/PWM1C) clock disable 0: clock enable 1: clock disable
PWM1CLR	97.4	R/W	0	PWM1 (PWM1A/PWM1B/PWM1C) clear and hold 0: PWM1 enable 1: PWM1 clear and hold
PWM1CKS	97.3~2	R/W	0	PWM1 (PWM1A/PWM1B/PWM1C) clock source select 0x: Fsys 10: FIRC 8MHz 11: FIRC 16MHz
-	97.1~0	R/W	0	Reserved, keep the two bits are 00
PWM1PRDH (98h)				Function related to: PWM1A / PWM1B / PWM1C
PWM1PRDH	98.7~0	R/W	FF	PWM1 (PWM1A/PWM1B/PWM1C) period data MSB 8bit
PWM1PRDL (99h)				Function related to: PWM1A / PWM1B / PWM1C
PWM1PRDL	99.7~0	R/W	FF	PWM1 (PWM1A/PWM1B/PWM1C) period data LSB 8bit About 16-bit data write: Write PWM1PRDL first, then PWM1PRDH About 16-bit data read: Read PWM1PRDH first, then PWM1PRDL
PWM1ADH (9Ah)				Function related to: PWM1A
PWM1ADH	9A.7~0	R/W	80	PWM1A Duty MSB 8bit
PWM1ADL (9Bh)				Function related to: PWM1A
PWM1ADL	9B.7~0	R/W	00	PWM1A Duty LSB 8bit About 16-bit data write: Write PWM1ADL first, then PWM1ADH About 16-bit data read: Read PWM1ADH first, then PWM1ADL

PWM1BDH (9Ch)				Function related to: PWM1B
PWM1BDH	9C.7~0	R/W	80	PWM1B Duty MSB 8bit
PWM1BDL (9Dh)				Function related to: PWM1B
PWM1BDL	9D.7~0	R/W	00	PWM1B Duty LSB 8bit About 16-bit data write: Write PWM1BDL first, then PWM1BDH About 16-bit data read: Read PWM1BDH first, then PWM1BDL
PWM1CDH (9Eh)				Function related to: PWM1C
PWM1CDH	9E.7~0	R/W	80	PWM1C Duty MSB 8bit
PWM1CDL (9Fh)				Function related to: PWM1C
PWM1CDL	9F.7~0	R/W	00	PWM1C Duty LSB 8bit About 16-bit data write: Write PWM1CDL first, then PWM1CDH About 16-bit data read: Read PWM1CDH first, then PWM1CDL
User Data Memory				
RAM	A0~EF	R/W	-	RAM Bank1 area (80 Bytes)
LVRPD (109h)				Function related to: LVR
LVRPD	109	W	-	Write 37h to force LVR disable
BGTRIM (10Eh)				Function related to: Bandgap
BGTRIM	10E.3~0	R/W	CFG	Bandgap voltage adjustment: 0000: Lowest voltage ... 1111: Highest voltage
IRCF (10Fh)				Function related to: Internal RC
IRCF	10F.6~0	R/W	CFG	FIRC frequency adjustment: 00h: Lowest frequency ... 7Fh: Highest frequency
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	186.2~0	R/W	0	Table read high address, data ROM pointer (DPTR) high byte
CRCDL (187h)				Function related to: CRC16
CRCDL	187.7~0	R/W	FF	16-bit CRC data LSB 8bit
CRCDH (188h)				Function related to: CRC16
CRCDH	188.7~0	R/W	FF	16-bit CRC data MSB 8bit
CRCIN (189h)				Function related to: CRC16
CRCIN	189.7~0	W	-	CRC data input
TABR (18Ch)				Function related to: Table Read
TABR	18C.7~0	R/W	0	1. TABR write 01h = opcode TABRL 2. TABR write 02h = opcode TABRH 3. After step.1 or step.2, read TABR to get main ROM table read value After step.1, read TABR to get EEPROM value (when EEPEN = E2h) <i>Table Read for ASM: TABRL / TABRH or TABR</i> <i>Table Read for C: TABR</i>

EEPCTL (18Dh)				Function related to: EEPROM
EEPTO	18D.2	R/W	0	EEPROM write time-out flag Set by H/W when EEPROM write time-out occurs Cleared by H/W when EEPTE=0
EEPTE	18D.1~0	R/W	0	EEPROM write watchdog timer enable (suggest set 11) 00: disable 01: wait 1.6mS trigger watchdog time-out flag 10: wait 6.4mS trigger watchdog time-out flag 11: wait 12.8mS trigger watchdog time-out flag
EEPEN (18Eh)				Function related to: EEPROM
EEPEN	18E.7~0	W	-	EEPROM read/write enable E2h: enable EEPROM read/write others: disable EEPROM read/write
EEPDT (18Fh)				Function related to: EEPROM
EEPDT	18F.7~0	W	-	EEPROM date to write

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					
ADDW X	f, d	ff00 0111 dfff ffff	1	C, DC, Z	Add W and "f"
ANDW X	f, d	ff00 0101 dfff ffff	1	Z	AND W with "f"
CLR X	f	ff00 0001 1fff ffff	1	Z	Clear "f"
CLR W		0000 0001 0100 0000	1	Z	Clear W
COM X	f, d	ff00 1001 dfff ff ff	1	Z	Complement "f"
DEC X	f, d	ff00 0011 dfff ffff	1	Z	Decrement "f"
DEC X SZ	f, d	ff00 1011 dfff ffff	1 or 2	-	Decrement "f", skip if zero
INC X	f, d	ff00 1010 dfff ffff	1	Z	Increment "f"
INC X SZ	f, d	ff00 1111 dfff ffff	1 or 2	-	Increment "f", skip if zero
IORW X	f, d	ff00 0100 dfff ffff	1	Z	OR W with "f"
MOV X	f, d	ff00 1000 dfff ffff	1	Z	Move "f"
MOV X W	f	ff00 1000 0fff ffff	1	Z	Move "f" to W
MOVW X	f	ff00 0000 1fff ffff	1	-	Move W to "f"
RL X	f, d	ff00 1101 dfff ffff	1	C	Rotate left "f" through carry
RR X	f, d	ff00 1100 dfff ffff	1	C	Rotate right "f" through carry
SUBW X	f, d	ff00 0010 dfff ffff	1	C, DC, Z	Subtract W from "f"
SWAP X	f, d	ff00 1110 dfff ffff	1	-	Swap nibbles in "f"
TST X	f	ff00 1000 1fff ffff	1	Z	Test if "f" is zero
XORW X	f, d	ff00 0110 dfff ffff	1	Z	XOR W with "f"
Bit-Oriented File Register Instruction					
BC X	f, b	ff11 00bb bfff ffff	1	-	Clear "b" bit of "f"
BS X	f, b	ff11 01bb bfff ffff	1	-	Set "b" bit of "f"
BT X SC	f, b	ff11 10bb bfff ffff	1 or 2	-	Test "b" bit of "f", skip if clear
BT X SS	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"
CLRWD T		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	$(W) \leftarrow (W) + k$	
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 A : W =0x25

ADDWX	Add W and "f"	
Syntax	ADDWX f [,d]	
Operands	f : 00h ~ 1FFh, d : 0, 1	
Operation	$(\text{destination}) \leftarrow (W) + (f)$	
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 A : W =0xD9, FSR =0xC2

ANDLW	Logical AND Literal "k" with W	
Syntax	ANDLW k	
Operands	k : 00h ~ FFh	
Operation	$(W) \leftarrow (W) \text{ AND } k$	
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 A : W =0x03

ANDWX	AND W with "f"	
Syntax	ANDWX f [,d]	
Operands	f : 00h ~ 1FFh, d : 0, 1	
Operation	$(\text{destination}) \leftarrow (W) \text{ AND } (f)$	
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 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 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 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	B : PC =LABEL1
	TRUE GOTO SUB1	A : if FLAG.1 =0, PC =FALSE
	FALSE ...	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	B : PC =LABEL1
	TRUE GOTO SUB1	A : if FLAG.1 =0, PC =TRUE
	FALSE ...	if FLAG.1 =1, PC =FALSE

CALL Call subroutine "k"

Syntax	CALL k
Operands	k : 000h ~ FFFh
Operation	Operation: TOS \leftarrow (PC) + 1, PC.10~0 \leftarrow k
Status Affected	-
OP-Code	0010 0kkk kkkk kkkk
Description	Call Subroutine. First, return address (PC+1) is pushed onto the stack. The 11-bit immediate address is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH. CALL is a two-cycle instruction.
Cycle	2
Example	LABEL1 CALL SUB1 B : PC =LABEL1 A : PC =SUB1, TOS =LABEL1 + 1

CLR X Clear "f"

Syntax	CLR X f
Operands	f : 00h ~ 1FFh
Operation	(f) \leftarrow 00h, Z \leftarrow 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	CLR X FLAG_REG B : FLAG_REG =0x5A A : FLAG_REG =0x00, Z =1

CLR W Clear W

Syntax	CLR W
Operands	-
Operation	(W) \leftarrow 00h, Z \leftarrow 1
Status Affected	Z
OP-Code	0000 0001 0100 0000
Description	W register is cleared and Z bit is set.
Cycle	1
Example	CLR W B : W =0x5A A : W =0x00, Z =1

CLR WDT Clear Watchdog Timer

Syntax	CLR WDT
Operands	-
Operation	WDT/WKT Timer \leftarrow 00h
Status Affected	TO, PD
OP-Code	0001 1110 0000 0100
Description	CLR WDT instruction clears the Watchdog/Wakeup Timer
Cycle	1
Example	CLR WDT B : WDT counter =? A : WDT counter =0x00

COMX	Complement "f"
Syntax	COMX f [,d]
Operands	f : 00h ~ 1FFh, d : 0, 1
Operation	(destination) \leftarrow (\bar{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 A : REG1 =0x13, W =0xEC

DECX	Decrement "f"
Syntax	DECX f [,d]
Operands	f : 00h ~ 1FFh, d : 0, 1
Operation	(destination) \leftarrow (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 A : CNT =0x00, Z =1

DECXSZ	Decrement "f", Skip if 0
Syntax	DECXSZ f [,d]
Operands	f : 00h ~ 1FFh, d : 0, 1
Operation	(destination) \leftarrow (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 GOTO LOOP CONTINUE B : PC =LABEL1 A : CNT =CNT - 1 if CNT =0, PC =CONTINUE if CNT \neq 0, PC =LABEL1 + 1

GOTO	Unconditional Branch
Syntax	GOTO k
Operands	k : 000h ~ FFFh
Operation	PC.10~0 \leftarrow k
Status Affected	-
OP-Code	0010 1kkk kkkk kkkk
Description	GOTO is an unconditional branch. The 11-bit immediate value is loaded into PC bits <10:0>. The upper bits of PC are loaded from PCLATH. GOTO is a two-cycle instruction.
Cycle	2
Example	LABEL1 GOTO SUB1 B : PC =LABEL1 A : PC =SUB1

INCX	Increment "f"
Syntax	INCX f[,d]
Operands	f : 00h ~ 1FFh
Operation	(destination) \leftarrow (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) \leftarrow (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 \neq 0, PC =LABEL1 + 1

IORLW	Inclusive OR Literal with W
Syntax	IORLW k
Operands	k : 00h ~ FFh
Operation	(W) \leftarrow (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) \leftarrow (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 =? 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 =? 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 =? 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 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

[illegible]

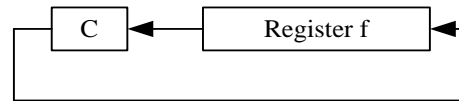
RETLW	Return with Literal in W
Syntax	RETLW k
Operands	k : 00h ~ FFh
Operation	PC \leftarrow TOS, (W) \leftarrow k
Status Affected	-
OP-Code	0001 1000 kkkk kkkk
Description	The W register is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address).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

RLX Rotate Left "f" through Carry

Syntax RLX f [,d]

Operands f : 00h ~ 1FFh, d : 0, 1

Operation



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

A : REG1 =1110 0110

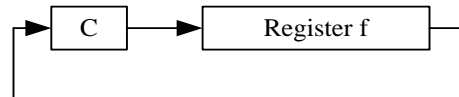
W =1100 1100, C =1

RRX Rotate Right "f" through Carry

Syntax RRX f [,d]

Operands f : 00h ~ 1FFh, d : 0, 1

Operation



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

A : REG1 =1110 0110

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	$(W) \leftarrow k - (W)$
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 0x15 B : W =0x25 A : W =0xF0

SUBWX
Subtract W from "f"

Syntax	SUBWX f [,d]
Operands	f : 00h ~ 1FFh, d : 0, 1
Operation	(destination) $\leftarrow (f) - (W)$
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=? A : REG1 =0x01, W =0x02, C=1, Z=0 SUBWX REG1, 1 B : REG1 =0x02, W =0x02, C=?, Z=? A : REG1 =0x00, W =0x02, C=1, Z=1 SUBWX REG1, 1 B : REG1 =0x01, W =0x02, C=?, Z=? A : REG1 =0xFF, W =0x02, C=0, Z=0

SWAPX
Swap Nibbles in "f"

Syntax	SWAPX f [,d]
Operands	f : 00h ~ 1FFh, d : 0, 1
Operation	(destination, 7~4) $\leftarrow (f.3\sim0)$, (destination, 3~0) $\leftarrow (f.7\sim4)$
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 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], DPL[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	(TAB1>>8)&0xFF	
	MOVWX	DPH	;Where DPH is register
			;W =0x89
			;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], DPL[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	(TAB1>>8)&0xFF	
	MOVWX	DPH	;Where DPH register
			;W =0x89
			;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 =? 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 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 A : REG =0x1A, W =0xB5

ELECTRICAL CHARACTERISTICS

1. Absolute Maximum Ratings ($T_A = 25^\circ\text{C}$)

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	$^\circ\text{C}$
Storage temperature	-65 to +150	

2. DC Characteristics ($T_A = 25^\circ\text{C}$, $V_{CC} = 5.0\text{V}$, unless otherwise specified)

Parameter	Sym	Conditions	Min	Typ	Max	Unit
Operating Voltage	V_{CC}	$F_{sys} = 8\text{MHz}$	2.1	—	5.5	
		$F_{sys} = 4\text{MHz}$	1.6	—	5.5	
Input High Voltage	V_{IH}	All Input $V_{CC} = 3\sim 5\text{V}$	$0.6V_{CC}$	—	V_{CC}	V
Input Low Voltage	V_{IL}	All Input $V_{CC} = 3\sim 5\text{V}$	V_{SS}	—	$0.2V_{CC}$	V
Output High Current	I_{OH}	All Output $V_{CC} = 5\text{V}, V_{OH} = 4.5\text{V}$ $V_{CC} = 3\text{V}, V_{OH} = 2.7\text{V}$	6	12	—	mA
			2.5	5	—	
Output Low Current	I_{OL}	All Output $V_{CC} = 5\text{V}, V_{OL} = 0.5\text{V}$ $V_{CC} = 3\text{V}, V_{OL} = 0.3\text{V}$	20	40	—	mA
			8	16	—	
Input Leakage Current (pin high)	I_{ILH}	All Input $V_{IN} = V_{CC}$	—	—	1	μA
Input Leakage Current (pin low)	I_{ILL}	All Input $V_{IN} = 0\text{V}$	—	—	-1	μA
Power Supply Current (No Load)	I_{CC}	FAST mode FIRC 8 MHz $V_{CC} = 5\text{V}$	—	3.5	—	mA
		FAST mode FIRC 4 MHz $V_{CC} = 5\text{V}$	—	2.7	—	
		FAST mode FIRC 2 MHz $V_{CC} = 5\text{V}$	—	2.3	—	
		FAST mode FIRC 1 MHz $V_{CC} = 5\text{V}$ $V_{CC} = 3\text{V}$	—	2.1	—	
			—	1.5	—	
		SLOW mode SIRC 70KHz OPA ON VR ON $V_{CC} = 5.0\text{V}$ $V_{CC} = 3.0\text{V}$	—	1300	—	μA
			—	1100	—	
		SLOW mode SIRC 70KHz OPA STOP VR STOP $V_{CC} = 5.0\text{V}$ $V_{CC} = 3.0\text{V}$	—	750	—	μA
			—	620	—	
		STOP mode LVRSAV = 1 $V_{CC} = 5.0\text{V}$ $V_{CC} = 3.0\text{V}$	—	0.1	—	μA
			—	0.1	—	

Parameter	Sym	Conditions	Min	Typ	Max	Unit
-----------	-----	------------	-----	-----	-----	------

Power Supply Current (No Load)	I_{CC}	IDLE mode SIRC 70 KHz LVRSAV= 1	$V_{CC} = 5.0V$	—	4.2	—	μA
			$V_{CC} = 3.0V$	—	1.2	—	
Pull-up Resistor	R_{UP}	$V_{IN} = 0 V$ Ports A	$V_{CC} = 5.0V$	—	41	—	$K \Omega$
			$V_{CC} = 3.0V$	—	76	—	
Internal Reference Voltage	VR	$V_{CC}=5.0V$, at least 1uF to ground		-1.2%	3	1.2%	V

3. Clock Timing ($T_A = -40^{\circ}C$ to $+85^{\circ}C$)

Parameter	Condition	Min	Typ	Max	Unit
FIRC Frequency (*)	$-40^{\circ}C \sim 85^{\circ}C, V_{CC} = 3.0 \sim 5.0V$	-2%	8	+2%	MHz
	$-40^{\circ}C \sim 85^{\circ}C, V_{CC} = 4.0 V$	-2%	8	+1.5%	
	$0^{\circ}C \sim 70^{\circ}C, V_{CC} = 4.0 V$	-2%	8	+1.5%	
	$25^{\circ}C, V_{CC} = 3.0 \sim 5.0 V$	-1.0%	8	+2%	
	$25^{\circ}C, V_{CC} = 4.0 V$	-0.5%	8	+0.5%	

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

4. Reset Timing Characteristics ($T_A = 25^{\circ}C$)

Parameter	Conditions	Min	Typ	Max	Unit
RESET Input Low width	Input $V_{CC} = 5 V \pm 10 \%$	30	—	—	μs
WDT time	$V_{CC} = 3 V, WDT PSC = 11$	—	1920	—	ms
	$V_{CC} = 5 V, WDT PSC = 11$		1760		
WKT time	$V_{CC} = 3 V, WKT PSC = 11$	—	120	—	ms
	$V_{CC} = 5 V, WKT PSC = 11$		108		
CPU start up time	$V_{CC} = 5 V$	—	24	—	ms

5. LVR Circuit Characteristics ($T_A = 25^{\circ}C$)

Parameter	Symbol	Min	Typ	Max	Unit
LVR Reference Voltage	LVR_{th}	—	2.2	—	V
		—	2.8	—	
		—	3.6	—	
		—	4.2	—	
LVR Hysteresis Voltage	V_{HYST}	—	± 0.1	—	V
Low Voltage Detection time	t_{LVR}	100	—	—	μs

6. ADC Electrical Characteristics ($T_A = 25^\circ\text{C}$, $V_{CC} = 3.0\text{V}$ to 5.5V , $V_{SS} = 0\text{V}$)

Parameter	Conditions	Min	Typ	Max	Units
Total Accuracy	$V_{CC} = 5\text{V}$, $V_{SS} = 0\text{V}$, $f_{\text{ADC}} = 1\text{MHz}$	–	± 2.5	± 13	LSB
Integral Non-Linearity		–	± 3.2	± 15	
Differential Non-linearity		–	± 1	± 4	
Max Input Clock freq. (f_{ADC})	Source impedance ($R_s < 10\text{K ohm}$)	–	–	2	MHz
	Source impedance ($R_s < 20\text{K ohm}$)	–	–	1	
	Source impedance ($R_s < 50\text{K ohm}$)	–	–	0.5	
	Source is VR (ADCHS=1101)	–	–	2	
Conversion Time	$f_{\text{ADC}} = 1\text{MHz}$	–	50	–	μs
Input Voltage	–	V_{SS}	–	V_{CC}	V

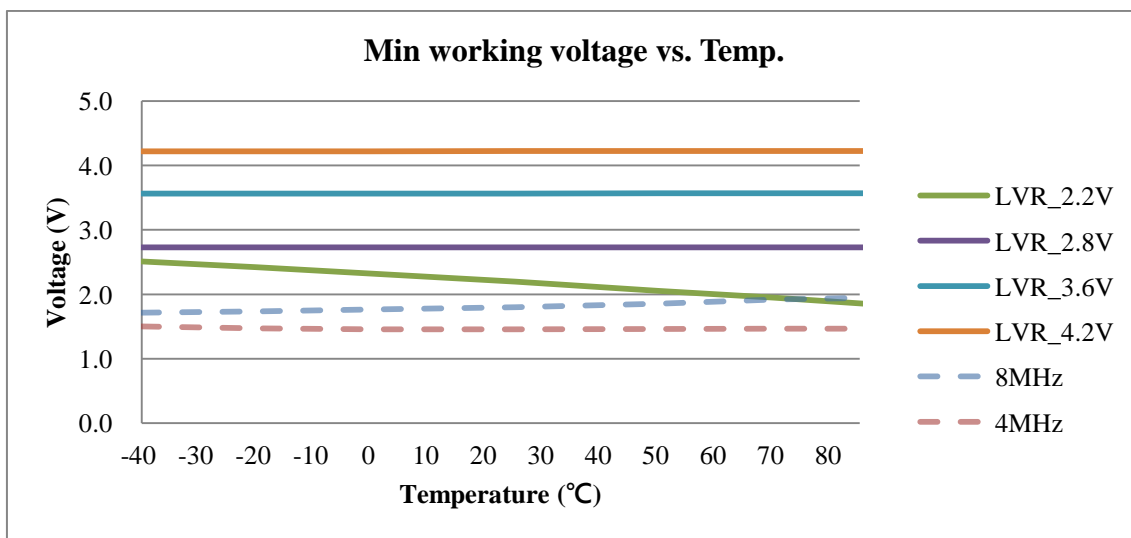
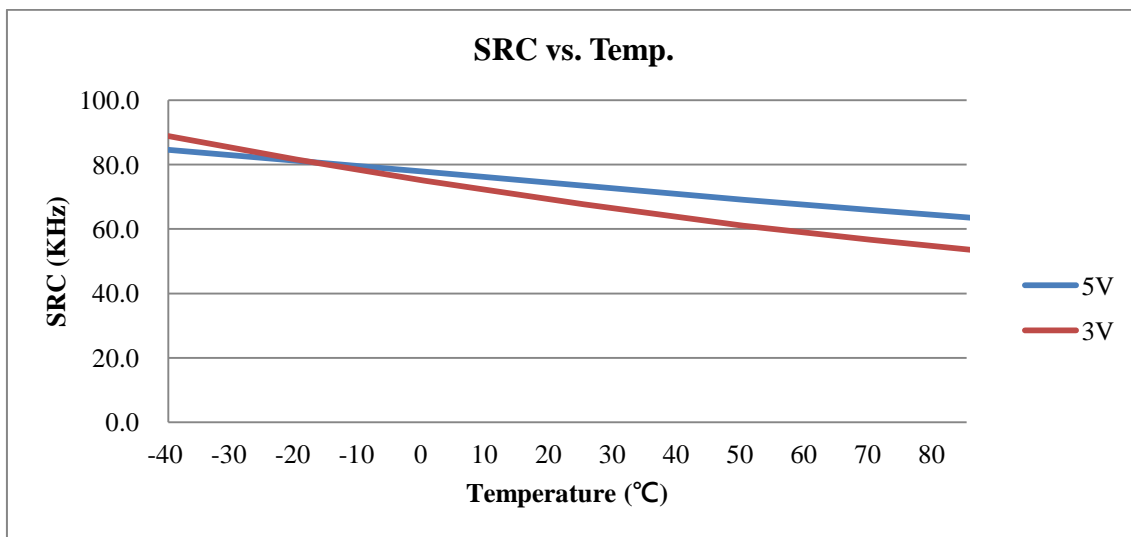
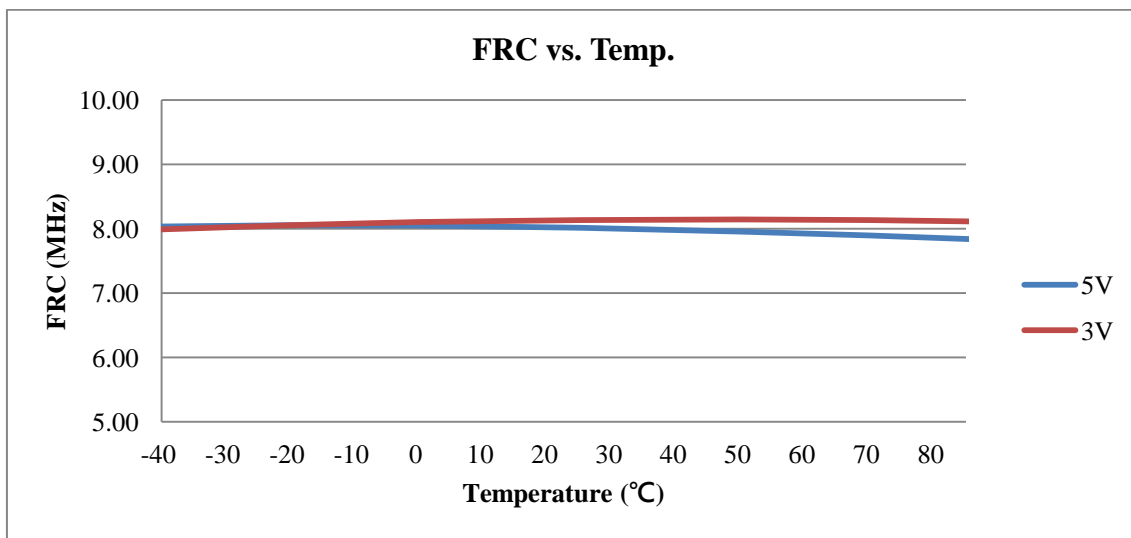
7. BCM Electrical Characteristics ($T_A = 25^\circ\text{C}$, $V_{CC} = 5\text{V}$, $V_{SS} = 0\text{V}$)

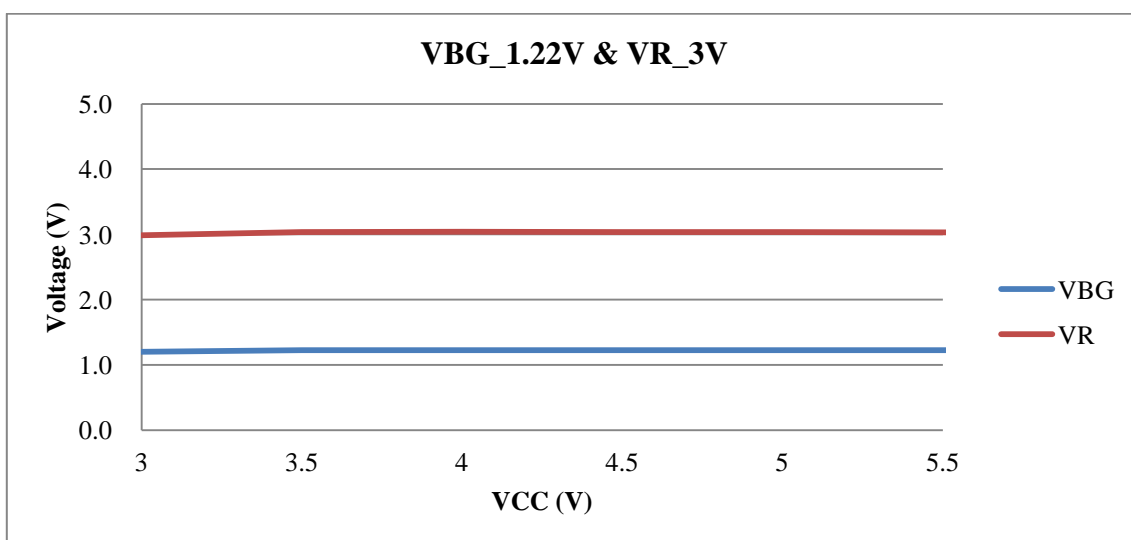
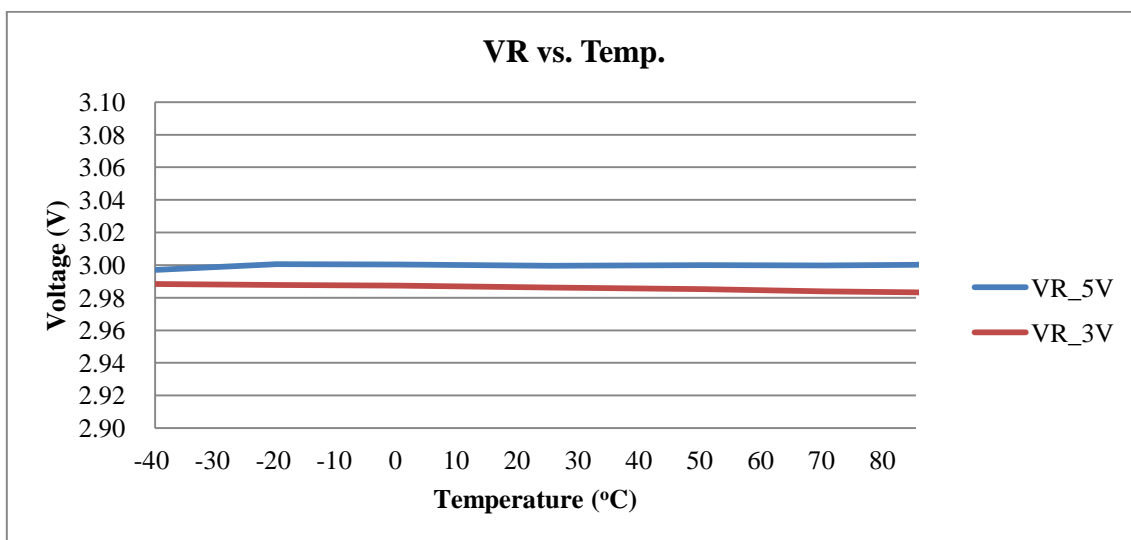
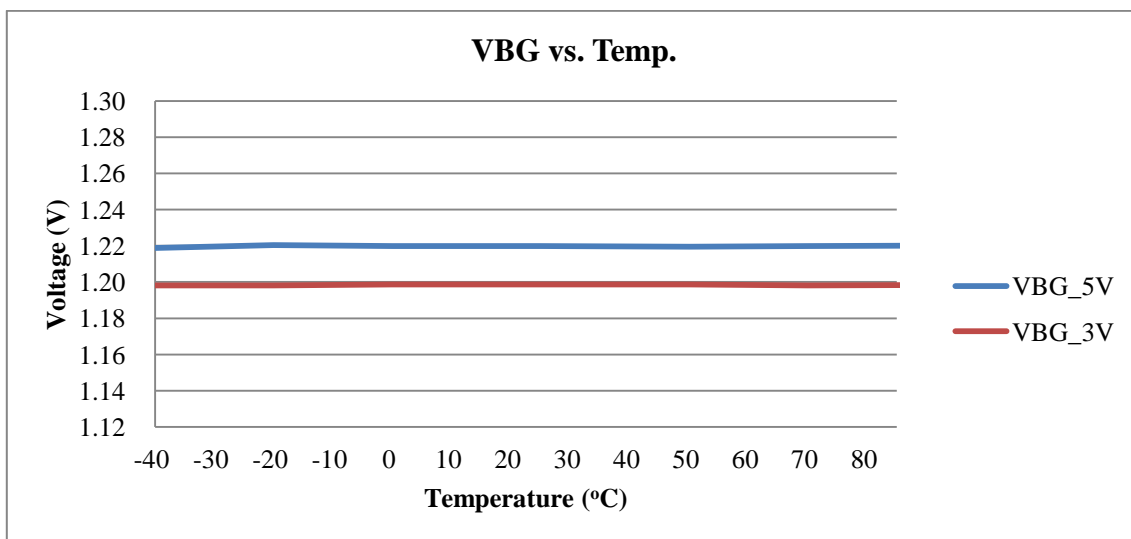
Parameter	Conditions	Min	Typ	Max	Units
Power supply V_{CC}	–	2.2	–	5.5	V
OPA offset	After trim, OPA_input=2.5 V	–	± 1	–	mV
OPA Vcm	–	0.1	–	$V_{CC}-1.2$	V
OPA IOL	$V_i = V_{CC}$, $V_{i+} = V_{SS}$, $T_{in} = 100\text{ ohm}$	–	20	–	mA
OPAAOL	$R_L = 1\text{M ohm}$, $C_L = 100\text{ pF}$	60	80	–	dB
OPA GBW	$R_L = 1\text{M ohm}$, $C_L = 100\text{ pF}$	–	1	–	MHz
OPA SR	No load	–	1	2	V/usec
OPA current consumption	$V_{CC} = 5\text{V}$	–	–	150	μA
DAC reference VR	$V_{CC} = 5\text{V}$, internal LDO	–	3	–	V
DAC output range	$VR = 3\text{V}$	0	–	3	V
DAC current consumption	$V_{CC} = 5\text{V}$	–	100	200	μA
DAC Integral Non-Linearity	$V_{CC} = 5\text{V}$, $V_{SS} = 0\text{V}$, $VR = 3\text{V}$ (DAC0DL > 3Ch & DAC1DL > 3Ch)	–	± 2	–	LSB
DAC Differential Non-Linearity		–	± 2	–	
DAC output impedance	R-2R+R structure	–	10	–	Kohm

8. EEPROM Block Characteristics ($T_A = 25^\circ\text{C}$, $V_{CC} = 5\text{V}$, $V_{SS} = 0\text{V}$)

Parameter	Conditions	Min	Typ	Max	Units
Read Voltage	V_{CC}	1.8	–	5.5	V
Write Voltage	V_{CC}	2.7	–	5.5	V
Write Current	–	–	5	30	mA
Write Time	Byte Write Time	–	0.7	–	ms
Endurance (Byte Write)	–	–	–	50,000	cycles

9. Electrical Characteristics Graphs





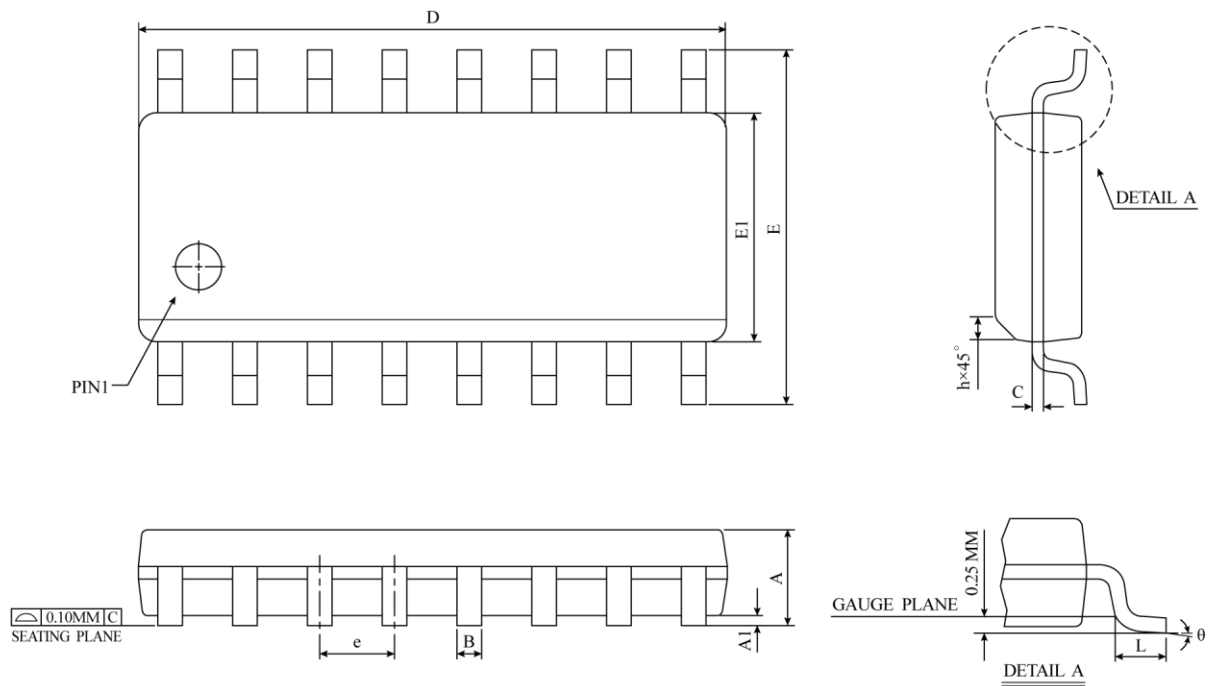
PACKAGING INFORMATION

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
TM56F8225-MTP	Wafer / Dice blank chip
TM56F8225-COD	Wafer / Dice with code
TM56F8225-MTP-16	SOP 16-pin (150 mil)
TM56F8225-MTP-03	DIP 16-pin (300 mil)

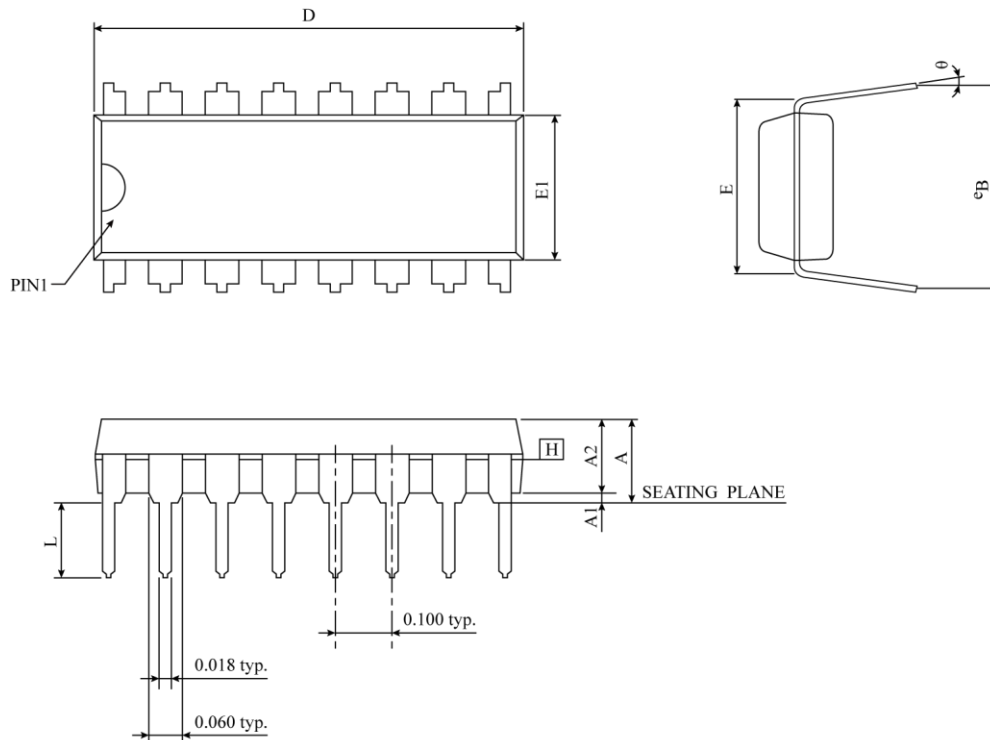
16-SOP Package Dimension



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.

16-DIP Package Dimension



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

NOTES :

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