



十速

TM57PA11

DATA SHEET

Rev 1.6

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
V1.0	Sep, 2015	<ol style="list-style-type: none"> 1. INT1(PA3) typing error (p19) 2. INT1(PA3), INT0(PA0) typing error (p42) 3. INT1IF, INT0IF description (p43)
V1.1	Dec, 2015	<ol style="list-style-type: none"> 1. Modify operating voltages (p6) 2. Add LVR selection table (p14) 3. Modify DC (p60) 4. Add LVR vs. temperature graph (p65)
V1.2	Aug, 2016	<ol style="list-style-type: none"> 1. Fix System Clock description 2. Remove PWMAPSC descriptions 3. Block Diagram modified 4. ADCx modified to ADx 5. Add PWMAPSC issue and PWMA block diagram modified 6. PWMA example code modified 7. ADC block diagram modified 8. Example code fix 9. ADCx modified to ADx 10. PWMAPSC description modified 11. ADCx modified to ADx
V1.3	Feb, 2017	<ol style="list-style-type: none"> 1. P5:remove "Dual System Clock" 2. P5:add Fast/Slow Mode switch method 3. p20:remove "Dual" from section name and related statement below 4. p21:change section name
V1.4	Sep, 2017	<ol style="list-style-type: none"> 1. p8,p9,p65~67:Add SOP8/DIP8 package information
V1.5	Dec, 2017	<ol style="list-style-type: none"> 1. p6 : modify/add Operating Voltage of FEATURES 2. p64: add Fsys vs. LVR selection graph
V1.6	Mar, 2018	<ol style="list-style-type: none"> 1. Add TSSOP-8 package type

CONTENTS

AMENDMENT HISTORY	2
CONTENTS.....	3
FEATURES	5
BLOCK DIAGRAM	7
PIN ASSIGNMENT	8
PIN DESCRIPTION	8
PIN SUMMARY.....	9
FUNCTIONAL DESCRIPTION	10
1. CPU Core	10
1.1 Clock Scheme and Instruction Cycle	10
1.2 Program ROM (PROM).....	11
1.3 System Configuration Register (SYSCFG)	12
1.4 Programming Counter (PC) and Stack.....	13
1.5 Reset (000H)	14
1.6 RAM Addressing Mode	15
1.7 ALU and Working (W) Register.....	16
1.8 STATUS Register (F-Plane 03H)	17
1.9 Interrupt.....	18
2 Chip Operation Mode	20
2.1 System Clock	20
2.2 Clock Modes Transition.....	21
3. I/O Port.....	23
3.1 PA0-2	23
3.2 PA3-4	24
3.3 PA7.....	26
4. Peripheral Functional Blocks.....	27
4.1 Watchdog (WDT) /Wakeup (WKT) Timer.....	27
4.2 Timer0.....	29
4.3 PWM0: 8-bit PWM.....	33
4.4 PWMA: (8+2) bits PWM.....	36
4.5 Analog-to-Digital Converter	39
4.6 System Clock Oscillator.....	41
MEMORY MAP.....	42
F-Plane	42
R-Plane.....	45

INSTRUCTION SET 47

ELECTRICAL CHARACTERISTICS 59

- 1. Absolute Maximum Ratings 59**
- 2. DC Characteristics 60**
- 3. Clock Timing 62**
- 4. Reset Timing Characteristics 62**
- 5. LVR Circuit and VBG (Bandgap Reference Voltage) Characteristics 62**
- 6. ADC Electrical Characteristics 62**
- 7. Characteristic Graphs 63**

PACKAGING INFORMATION 65

- SOP-8 (150mil) Package Dimension 66**
- DIP-8 (300mil) Package Dimension 67**
- TSSOP-8 (173mil) Package Dimension 68**

FEATURES

1. **ROM: 1K x 14 bits OTP or 1K x 14 bits TTP™ (Two Time Programmable ROM)**
2. **RAM: 48 x 8 bits**
3. **STACK: 4 Levels**
4. **System Oscillation Sources (Fsys)**
 - Fast-clock
 - FIRC (Fast Internal RC): can be selected to 1/2/4/8 MHz
 - Slow-clock
 - SIRC (Slow Internal RC):
VDD=5V, SIRC=167 KHz/86 KHz/43 KHz/21 KHz
VDD=3V, SIRC=136 KHz/68 KHz/34 KHz/17 KHz
5. **Power Saving Operation Mode**
 - FAST mode: Slow-clock can be disabled or enabled
 - SLOW mode: Fast-clock stops, CPU is running
 - Fast Mode and Slow Mode can be chosen by CPUCKS control bit.
 - STOP mode: All Clocks stop, Wake-up Timer is disabled or enabled
6. **1 Independent Timers**
 - Timer0
 - 8-bit timer divided by 1~256 pre-scaler option, Counter/Interrupt/Stop function
 - Capture – high duty or low duty (pulse width measurement)
 - Overflow and Toggle out
7. **Interrupt**
 - Three External Interrupt pins
 - 2 pins are falling edge wake-up triggered
 - 1 pin is rising or falling edge wake-up triggered
 - Timer0/WKT (wake-up) Interrupts
 - PWM0/PWMA / ADC Interrupts
8. **Wake-up (WKT) Timer**
 - Clocked by built-in RC oscillator with 4 adjustable Interrupt times
VDD=5V, WKT=0.8 ms/1.5 ms/24 ms/97 ms
VDD=3V, WKT=1.0 ms/2.0 ms/30 ms/120 ms

9. Watchdog Timer

- Clocked by built-in RC oscillator with 4 adjustable Reset Time
VDD=5V, WDT=100 ms/200 ms/800 ms/1600 ms
VDD=3V, WDT=123 ms/256 ms/1000 ms/2000 ms

10. 2 Independent PWMs

- PWM0:
 - 8-bit with 1~8 pre-scalers, period-adjustable/duty-adjustable/Clear&Hold
 - Clock source, PWMCLK, FIRC 8 MHz or 16 MHz
- PWMA:
 - 8+2 bits, period-adjustable / duty-adjustable / Clear&Hold
 - Clock source, PWMCLK, FIRC 8 MHz or 16 MHz

11. 12-bit ADC converter with 5 input channels**12. Reset Sources**

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

13. Low Voltage Reset Option: LVR2.1V, LVR2.1V disabled in STOP mode, LVR2.9V**14. Operating Voltage: Low Voltage Reset Level to 5.5V (tested under LVR OFF)**

F_{sys}=4 MHz, 2.2V~5.5V

F_{sys}=8 MHz, 2.4V~5.5V

Refer to the graph “F_{sys} Minimum Operating Volt vs. LVR optimized selection” for more details on Characteristic Graphs section.

15. Enhanced Power Noise Rejection.**16. Operating Temperature Range: -40°C to +85°C****17. Instruction set: 36 Instructions****18. Instruction Execution Time**

- 2 oscillation clocks per instruction except branch

19. I/O ports: Maximum 6 programmable I/O pins

- Pseudo-Open-Drain Output (PA0/PA1/PA2)
- Open-Drain Output (PA3/PA4, PA7 with internal fixed pull high resistor.)
- CMOS Push-Pull Output (PA0~PA4)
- Schmitt Trigger Input with pull-up resistor option

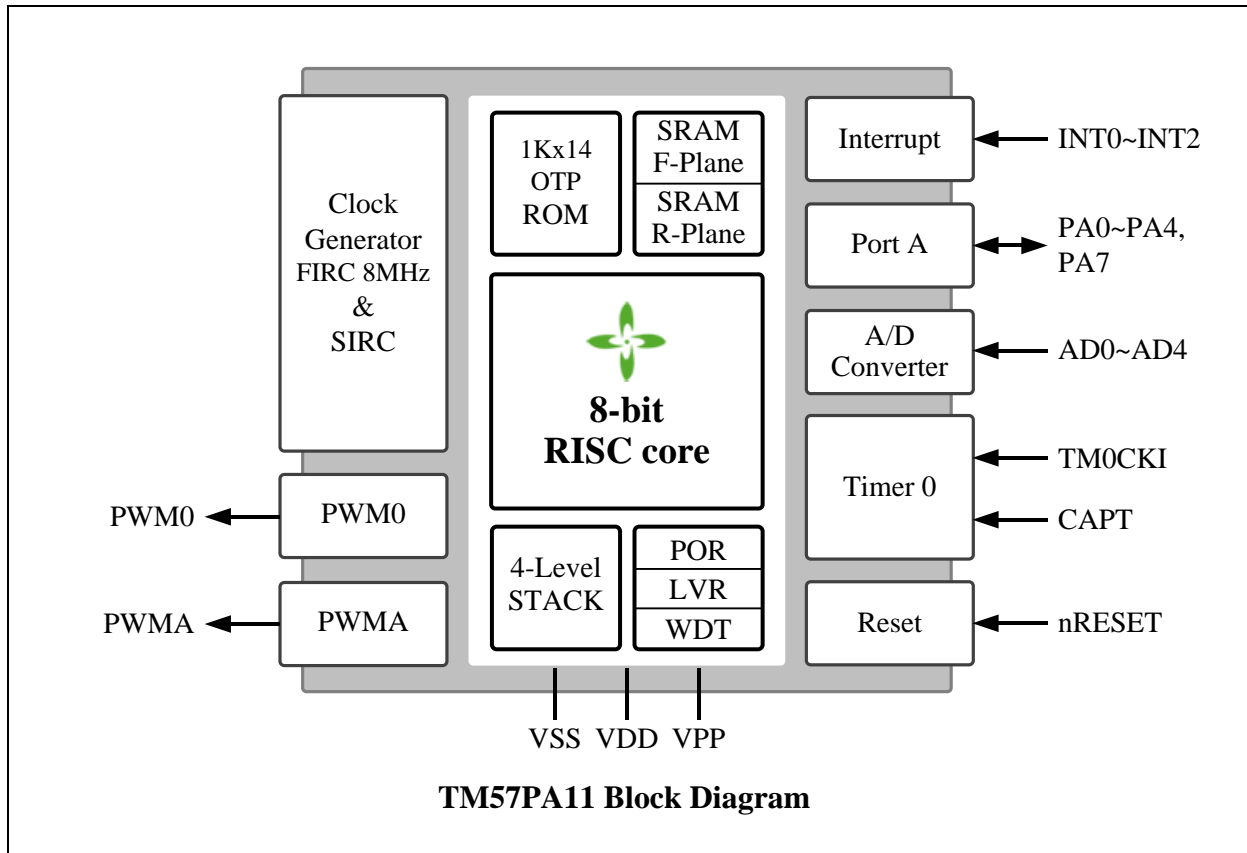
20. Package Types:

- 8-pin DIP (300 mil), SOP (300 mil), TSSOP (173 mil)

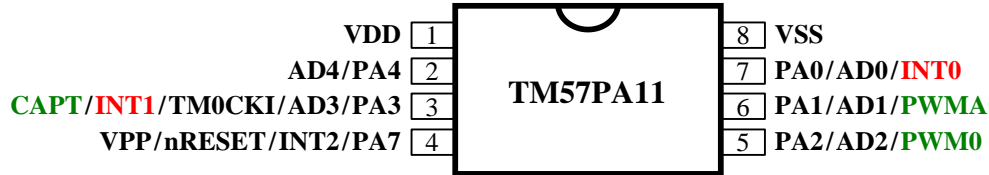
21. Supported EV board on ICE

EV board: EV2777

BLOCK DIAGRAM



PIN ASSIGNMENT



PIN DESCRIPTION

Name	In/Out	Pin Description
PA0–PA2	I/O	Bit-programmable I/O port for Schmitt-trigger input, CMOS push-pull output or “ pseudo-open-drain ” output. Pull-up resistors are assignable by software.
PA3–PA4	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.
VPP/nRESET/PA7	I	Schmitt-trigger input with pull-high, External active low reset, normal stay to “high”.
VDD, VSS	P	Power Voltage input pin and ground
VPP	I	PROM programming high voltage input
INT0–INT2	I	External interrupt input
PWM0 PWMA	O	PWM outputs
TM0CKI	I	Timer0’s input in counter mode
CAPT	I	Timer0 Capture input
AD0~AD4	I	A/D converter input

PROGRAMMING PINS:

VDD/VSS/PA0/PA1/PA3/PA4/PA7 (VPP)

PIN SUMMARY

Pin Number	Pin Name	Type	GPIO					Function After Reset	Alternate Function			
			Input		Output				PWM	Touch Key	ADC	MISC
			Weak Pull-up	Ext. Interrupt	O.D	P.O.D	P.P					
1	VDD	P										
2	AD4/PA4	I/O			○		○				○	
3	CAPT/INT1/TM0CKI/ AD3/PA3	I/O		○	○		○				○	CAPT/TM0CKI
4	VPP/nRESET/INT2/ PA7	I		○								nRESET
5	PA2/AD2/PWM0	I/O				○	○		○		○	
6	PA1/AD1/PWMA	I/O				○	○		○		○	
7	PA0/AD0/INT0	I/O				○	○				○	
8	VSS	P										

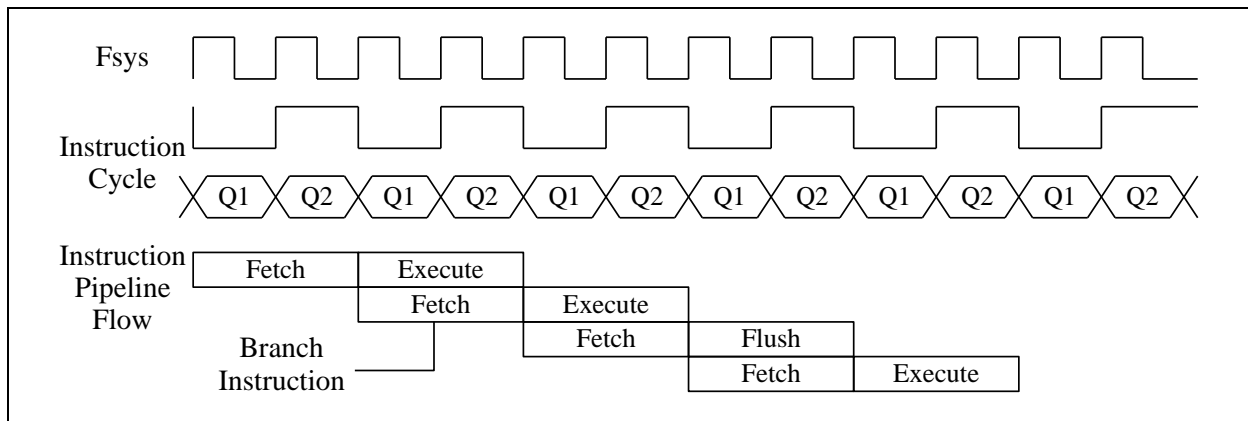
Symbol : P.P. = Push-Pull Output
P.O.D. = Pseudo Open Drain
O.D. = Open Drain

FUNCTIONAL DESCRIPTION

1. CPU Core

1.1 Clock Scheme and Instruction Cycle

The system clock is internally divided by two to generate Q1 state and Q2 state for each instruction cycle. The Programming Counter (PC) is updated at Q1 and the instruction is fetched from program ROM and latched into the instruction register in Q2. It is then decoded and executed during the following Q1-Q2 cycle. Branch instructions take two cycles since the fetch instruction is ‘flushed’ from the pipeline, while the new instruction is being fetched and then executed.



Terminology definitions:

Fsys: System clock. The main clock that drives the core logic and all peripherals. The clock source can be either Fast-clock or Slow-clock which can be set by registers.

Fast-clock: The clock source only from Fast Internal RC oscillator (FIRC).

Slow-clock: The clock source only from Slow Internal RC oscillator (SIRC).

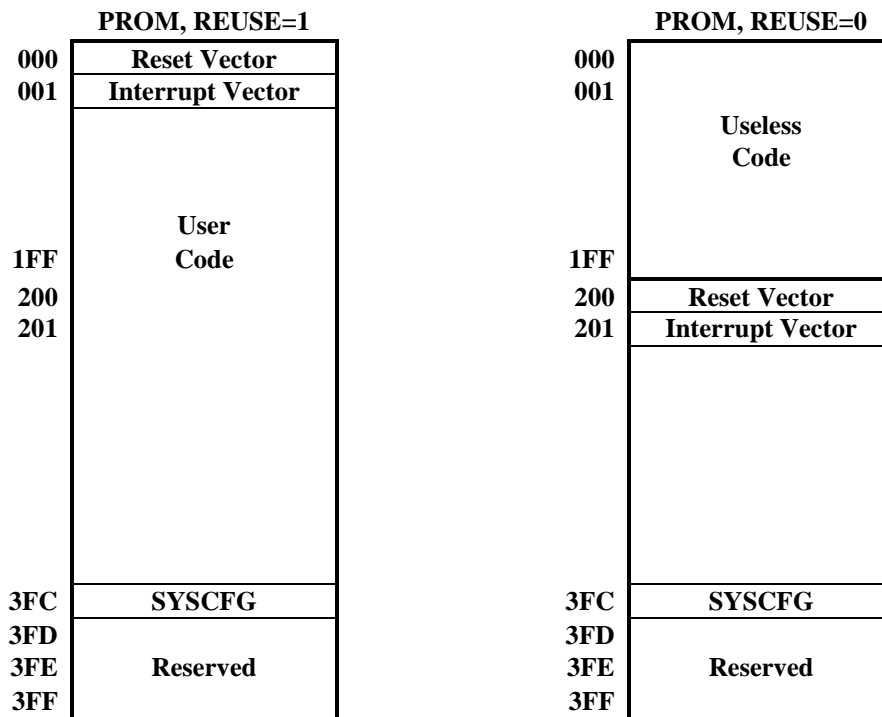
Instruction Cycle=Fsys/2

FIRC: Fast Internal RC oscillator

SIRC: Slow Internal RC oscillator

1.2 Program ROM (PROM)

The PROM of this device is 1K words. For some F/W program, the program size could be less than 1K words. To fully utilize the PROM, the device allows users to reuse the PROM. This feature is named as Two Time Programmable (TTP) ROM. While the first half of PROM is occupied by a useless program code and the second half of the PROM remains blank, users can re-write the PROM with the updated program code into the second half of the PROM. In the Re-use mode, the Reset Vector and Interrupt Vector are re-allocated at the beginning of the PROM's second half by the Assembly Compiler. Users simply choose the "REUSE" option in the ICE tool interface, and then the Compiler will move the object code to proper location. That is, the user's program still has reset vector at address 000h, but the compiled object code has reset vector at 200h. In the SYSCFG, if PROTECT=0 and REUSE=1, the Code protection area is first half of PROM. This allows the Writer tool to write then verify the Code during the Re-use Code programming. After the Re-use Code being written into the PROM's second half, user should write "REUSE" control bit to "0". In the mean while, the Code protection area becomes the whole PROM except the Reserved Area.



1.3 System Configuration Register (SYSCFG)

The System Configuration Register (SYSCFG) is located at ROM address 3FCh. The SYSCFG determines the option for initial condition of MCU. It is written by PROM Writer only. User can select LVR threshold voltage and chip operation mode by SYSCFG register. The default value of SYSCFG is 14'b11_1111_111x_xxxx. The 13th bit of SYSCFG is code protection selection bit. If this bit is 0, the data in PROM will be protected, when user read PROM.

Bit	13~0	
Default Value	11_1111_111x_xxxx	
Bit	Description	
13	PROTECT: Code protection selection	
	1	Disable
	0	Enable
12	REUSE: PROM Re-use control	
	1	Disable
	0	Enable
11-10	LVR: Low Voltage Reset Mode	
	11	2.1V, always enable
	10	2.1V, disable in STOP mode
	01	2.9V, always enable
00	Disable	
9-8	Reserved	
7	XRSTE: External Pin Reset Enable	
	1	Enable
	0	Disable
6	WDTE: WDT Reset Enable	
	1	Enable
	0	Disable
5-0	Reserved	

Bit13: PROTECT option

Protect code option is Program ROM (PROM) protection. When protect code option is enabled, the PROM code does not read PROM content.

Bit12: REUSE option

The REUSE function can be used if the code size less than 1FBh (508 words); however, REUSE cannot be used when the code size is larger than 508 words and REUSE bit must be set to '1'.

The REUSE can be enabled only if the user program size is less than 508 words. But they all use the same SYSCFG whose address is 3FCh.

Bit7: XRSTE option

The reset pin is shared with the general input pin (PA7) controlled by SYSCFG[7] option.

- Reset: The reset pin is external reset function. When falling edge trigger occurs, the system will be reset.
- PA7: Set reset pin to general input pin (PA7). The external reset function is disabled.

1.4 Programming Counter (PC) and Stack

The Programming Counter is 10-bit wide capable of addressing a 1K x 14 OTP ROM. 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 (001h) are provided for PC initialization and Interrupt. For CALL/GOTO instructions, PC loads 10 bits address from instruction word. For RET/RETI/RETLW instructions, PC retrieves its content from the top level STACK. For the other instructions updating PC [7:0], the PC [9:8] keeps unchanged. The STACK is 10-bit wide and 4-level in depth. The CALL instruction and hardware interrupt will push STACK level in order. While the RET/RETI/RETLW instruction pops the STACK level in order.

◇ External: To look up the PROM data located in “TABLE”.

```

ORG      000H          ; Reset Vector
GOTO     START

START:
MOVLW   00H
MOVWF   INDEX          ; Set lookup table's address .

LOOP:
MOVFW   INDEX          ; Move index value to W register.
CALL    TABLE         ; To look up data, W=55H.
.....
INCF    INDEX,1        ; Increment the index address for next address
.....
GOTO    LOOP           ; Go to LOOP label.

TABLE:
ADDWF   PC,1           ; Add the W register to PCL.
RETLW   55H            ; W=55h when returns
RETLW   56H            ; W=56h when returns
RETLW   58H            ; W=58h when returns
.....

```

1.5 Reset (000H)

This device can be RESET in four ways.

- Power-On-Reset
- Low Voltage Reset (LVR) (SYSCFG bit-11-10)
- External Pin Reset (PA7) (SYSCFG bit-7)
- Watchdog Reset (WDT) (SYSCFG bit-5)

After Power-On-Reset, all system and peripheral control registers are then set to their default hardware Reset values. The clock source, LVR level and chip operation mode are selected by the SYSCFG register value. The Low Voltage Reset features static reset when supply voltage is below a threshold level. There are three threshold levels can be selected. The LVR's operation mode is defined by the SYSCFG register. If operating frequency is faster than 4 MHz, selection LVR 2.1V is recommended. The External Pin Reset and Watchdog Reset can be disabled or enabled by the SYSCFG register. These two resets also set all the control registers to their default reset value. The TO/PD flags is not affected by these resets.

LVR Selection Table

LVR Threshold Level	Consider the operating voltage to choose LVR
LVR2.9	$5.5V > V_{DD} > 3.3V$ or $V_{DD}=5.0V$
LVR2.1	V_{DD} is wide voltage range

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

◇ Example: Defining Reset Vector

```

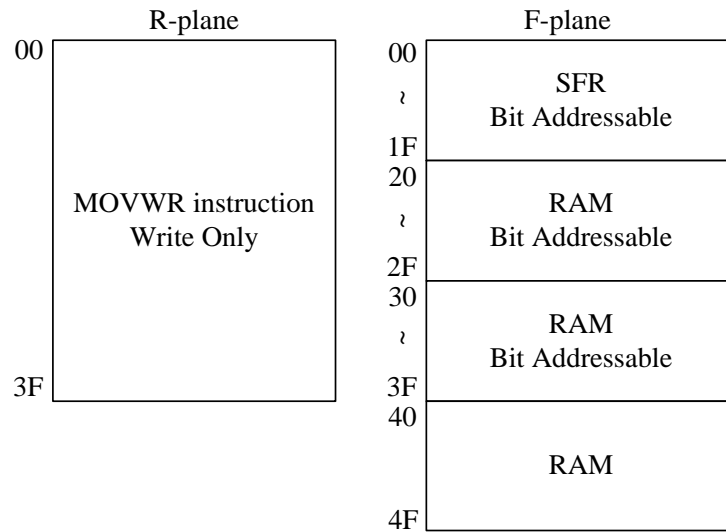
ORG      000H
GOTO     START      ; Jump to user program address.

ORG      010H

START:
...
...
GOTO     START
    
```

1.6 RAM Addressing Mode

There are two Data Memory Planes in CPU, R-Plane and F-Plane. The registers in R-Plane are write-only. The “MOVWR” instruction copies the W-register’s content to R-Plane registers by direct addressing mode. The lower locations of F-Plane are reserved for the SFR. Above the SFR is General Purpose Data Memory, implemented as static RAM. F-Plane can be addressed directly or indirectly. Indirect Addressing is made by INDF register. The INDF register is not a physical register. Addressing INDF actually addresses the register whose address is contained in the FSR register (FSR is a pointer). The first half of F-Plane is bit-addressable, while the second half of F-Plane is not bit-addressable.



◇Example: Write immediate data into R-Plane register.

```
MOVLW    AAH    ; Move immediate AAH into W register.
MOVWR    05H    ; Move W value into R-Plane location 05H data register.
```

◇Example: Move the immediate data 55H to W register and F-Plane location 20H.

```
MOVLW    55H    ; Move immediate 55H into W register.
MOVWF    20H    ; Get the content of W and save in F-Plane location 20H.
```

◇Example: Move F-Plane location 20H data into W register.

```
MOVFW    20H    ; Get the content of F-Plane location 20H and save in W.
```

◇Example: Indirectly addressing mode with FSR/INDF register (F-Plane 04H / 00H).

```
MOVLW    20H
MOVWF    FSR    ; Move immediate 20H into FSR register.
MOVLW    55H
MOVWF    INDF   ; Use data pointer FSR writes a data into F-Plane location 20H.
                    ; 55H into F-plane 20H.
INCF     FSR,1  ; Increment the index address for next address.
MOVFW    INDF   ; Use data pointer FSR reads a data from F-Plane 21H
```

1.7 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.

1.8 STATUS Register (F-Plane 03H)

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 BCF, BSF and MOVWF 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	R	R	R	R	R/W	R/W	R/W
Bit	Description							
7	Not implemented, read as '0'							
6	Not implemented, read as '0'							
5	Not implemented, read as '0'							
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			
	1: a carry from the low nibble bits of the result occurs 0: no carry				1: no borrow 0: a borrow from the low nibble bits of the result occurs			
0	C: Carry Flag or /Borrow Flag							
	ADD instruction				SUB instruction			
	1: a carry occurs from the MSB 0: no carry				1: no borrow 0: a borrow occurs from the MSB			

◇Example: Write immediate data into STATUS register.

```
MOVLW 00H
MOVWF STATUS ; Clear STATUS register.
```

◇Example: Bit addressing set and clear STATUS register.

```
BSF STATUS,0 ; Set C=1.
BSF 03H,5 ; Selection RAM Bank1
BCF STATUS,0 ; Clear C=0.
BCF 03H,5 ; Selection RAM Bank0
```

◇Example: Determine the C flag by BTFSS instruction.

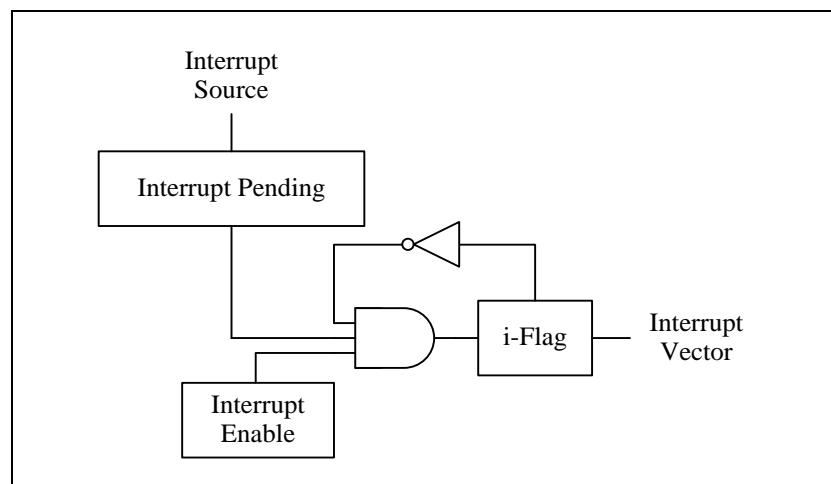
```
BTFSS STATUS,0 ; Check the carry flag
GOTO LABEL_1 ; If C=0, go to LABEL_1
GOTO LABEL_2 ; If C=1, go to LABEL_2
```

1.9 Interrupt

This device has 1 level, 1 vector and eight interrupt sources. Each interrupt source has its own enable control bit. An interrupt event will set its individual pending flag; no matter its interrupt enable control bit is 0 or 1. Because TM57PA11 has only 1 vector, there is not an interrupt priority register. The interrupt priority is determined by F/W.

If the corresponding interrupt enable bit has been set (INTE), 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 001” 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 (PA3) interrupt request and rising edge trigger.

```

    ORG      000H          ; Reset vector.
    GOTO     START        ; Go to user program address.
    ORG      01H          ; All interrupt vector.
    GOTO     INT_SUBROUTINE ; If INT1(PA3) input occurs, trigger rising edge.
    ORG      02H

START:
    MOVLW   11110111B
    MOVWR   PAPUN        ; Enable INT1 (PA3) input pull up resistor.
    MOVLW   01000111B
    MOVWR   R08          ; Set INT1 interrupt trigger as rising edge.
    MOVLW   1111101B
    MOVWF   INTIF        ; Clear INT1 interrupt request flag
    BSF     INT1IE       ; Enable INT1 interrupt.

MAIN:
    ...
    GOTO    MAIN

INT_SUBROUTINE:
    ... ; Push routine to Save W and STATUS data to buffers.
    BTFSS  INT1IF       ; Check INT1IF bit.
    GOTO   EXIT_INT     ; INT1IF=0, exit interrupt vector.
    ... ; INT1 interrupt service routine.
    MOVLW  1111101B
    MOVWF  INTIF        ; Clear INT1IF bit.
    GOTO   EXIT_INT

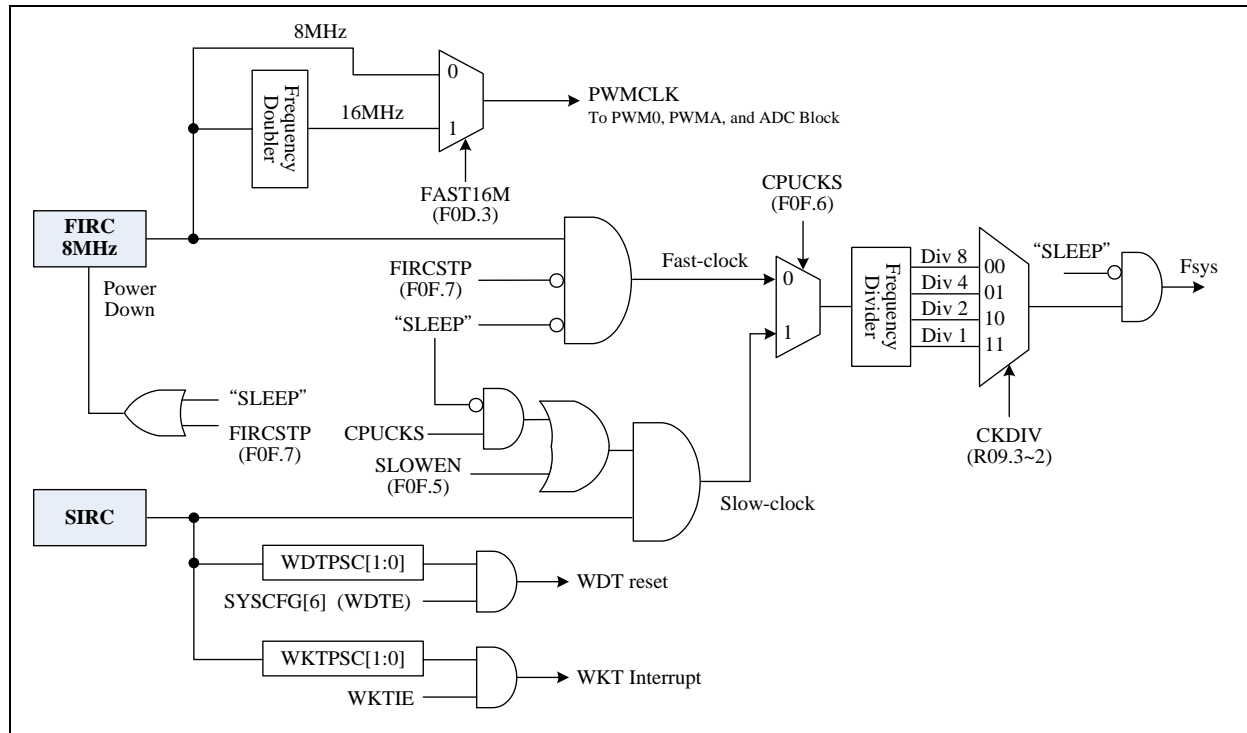
EXIT_INT:
    ... ; POP Routine W and STATUS data from buffers.
    RETI

```

2 Chip Operation Mode

2.1 System Clock

TM57PA11 has 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.



Clock Scheme Block Diagram

FAST mode

After power on or reset, TM57PA11 enters FAST mode. In FAST mode, TM57PA11 can only select FIRC as its CPU clock. Besides, firmware can also enable or disable the Slow-clock.

In this mode, the program is executed using Fast-clock as CPU clock (Fsys). The Timer0 is driven by Fast-clock. PWM0, PWMA, and ADC blocks are driven by PWMCLK which can be chosen either 8MHz or 16 MHz.

SLOW mode

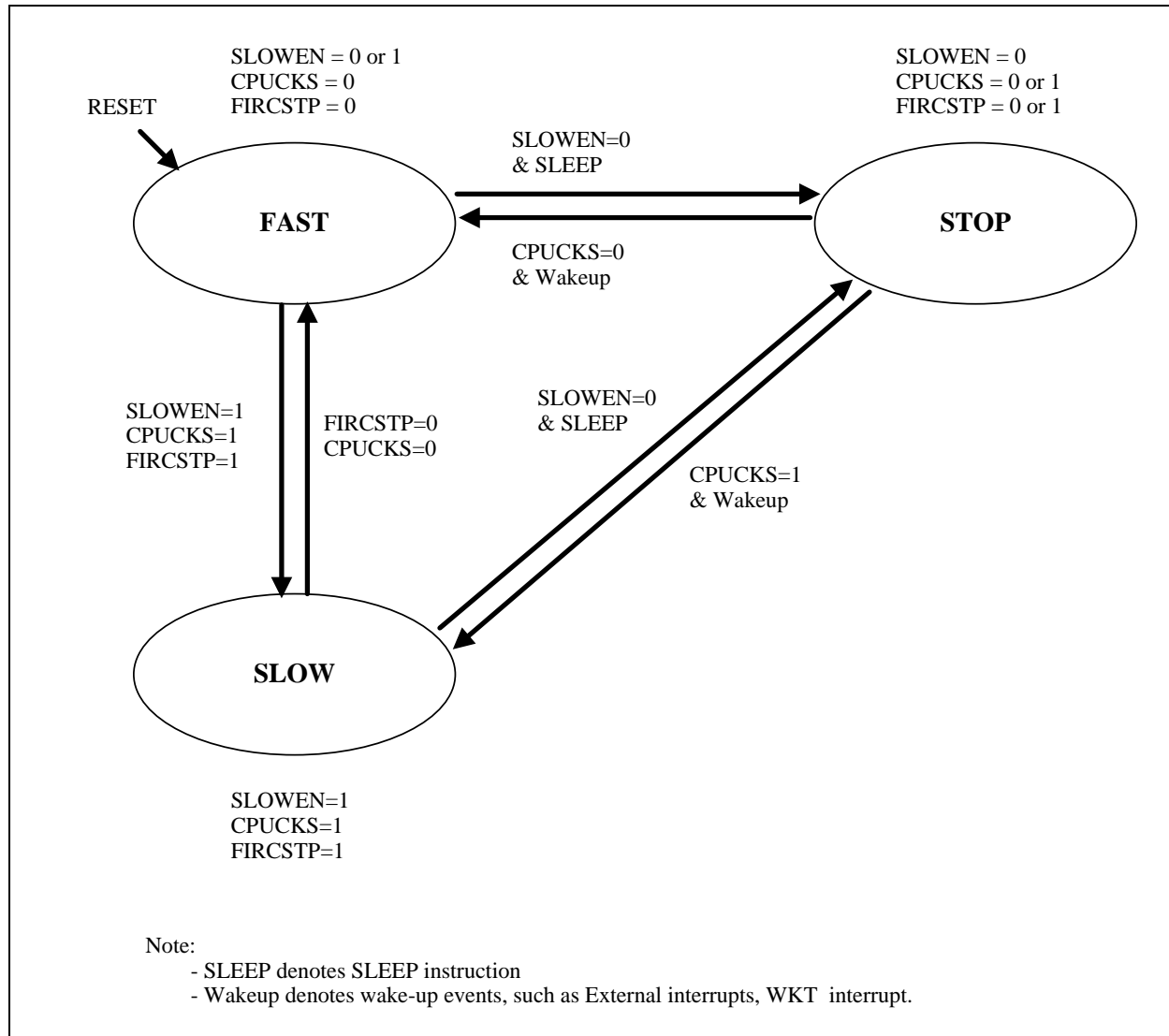
The SIRC is only one type for Slow-clock. In SLOW mode, the Fast-clock is stopped and Slow-clock is enabled for power saving. Timer0 clock source is Slow-clock in the SLOW mode.

STOP mode

If Slow-clock is disabled before executing the SLEEP instruction, every block is turned off and the TM57PA11 enters the STOP mode. All clock oscillators either Fast-clock or Slow-clock is power down and no clock is generated.

2.2 Clock Modes Transition

TM57PA11 is operated in one of three modes: FAST mode, SLOW mode, and STOP mode.



CPU Operation Block Diagram

Mode	Oscillator	Fsys	Fast-clock	Slow-clock	TM0	Wakeup event
FAST	FIRC	Fast-clock	Run	Set by SLOWEN bit	Run	X
SLOW	SIRC	Slow-clock	Set by FIRCSTP	Run	Run	X
STOP	SIRC ⁽¹⁾	Stop	Stop	Stop	Stop	WKT/INTx

(1) If WDT or WKT function is enabled

● **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 (SLOWEN =1)
- (2) Switch to Slow-clock (CPUCKS=1)
- (3) Stop Fast-clock (FIRCSTP=1)

◇Example: Switch FAST mode to SLOW mode.

```

MOVLW    00001100B
MOVWR    R09           ; CPU clock=FIRC/SIRC div 1.
BSF      SLOWEN        ; Enable Slow-clock.
NOP
BSF      CPUCKS        ; Fsys=SIRC Slow-clock.
BSF      FIRCSTP       ; Disable Fast-clock.
    
```

● **SLOW mode switches to FAST mode**

SLOW mode can be enabled by SLOWEN bit and CPUCKS bit in F0F register of F-Plane. The following steps are suggested to be executed by order when SLOW mode switches to FAST mode:

- (1) Enable Fast-clock (FIRCSTP=0)
- (2) Switch to Fast-clock (CPUCKS=0)
- (3) Stop Slow-clock (SLOWEN=0)

Note: Stop Slow-clock (SLOWEN=0) is optional. Slow-clock keep oscillating if WDTE or WKT Interrupt is enabled..

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

```

MOVLW    00001100B
MOVWR    R09           ; Fast-clock=FIRC 8 MHz div 1
BCF      FIRCSTP      ; Enable Fast-clock.
NOP
BCF      CPUCKS       ; Fsys=Fast-clock
BCF      SLOWEN       ; Disable Slow-clock
    
```

● **STOP mode setting**

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

- (1) Stop Slow-clock (SLOWEN=0)
- (2) Execute SLEEP instruction

STOP mode can be woken up by INT0, INT1, INT2, and WKT interrupt.

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

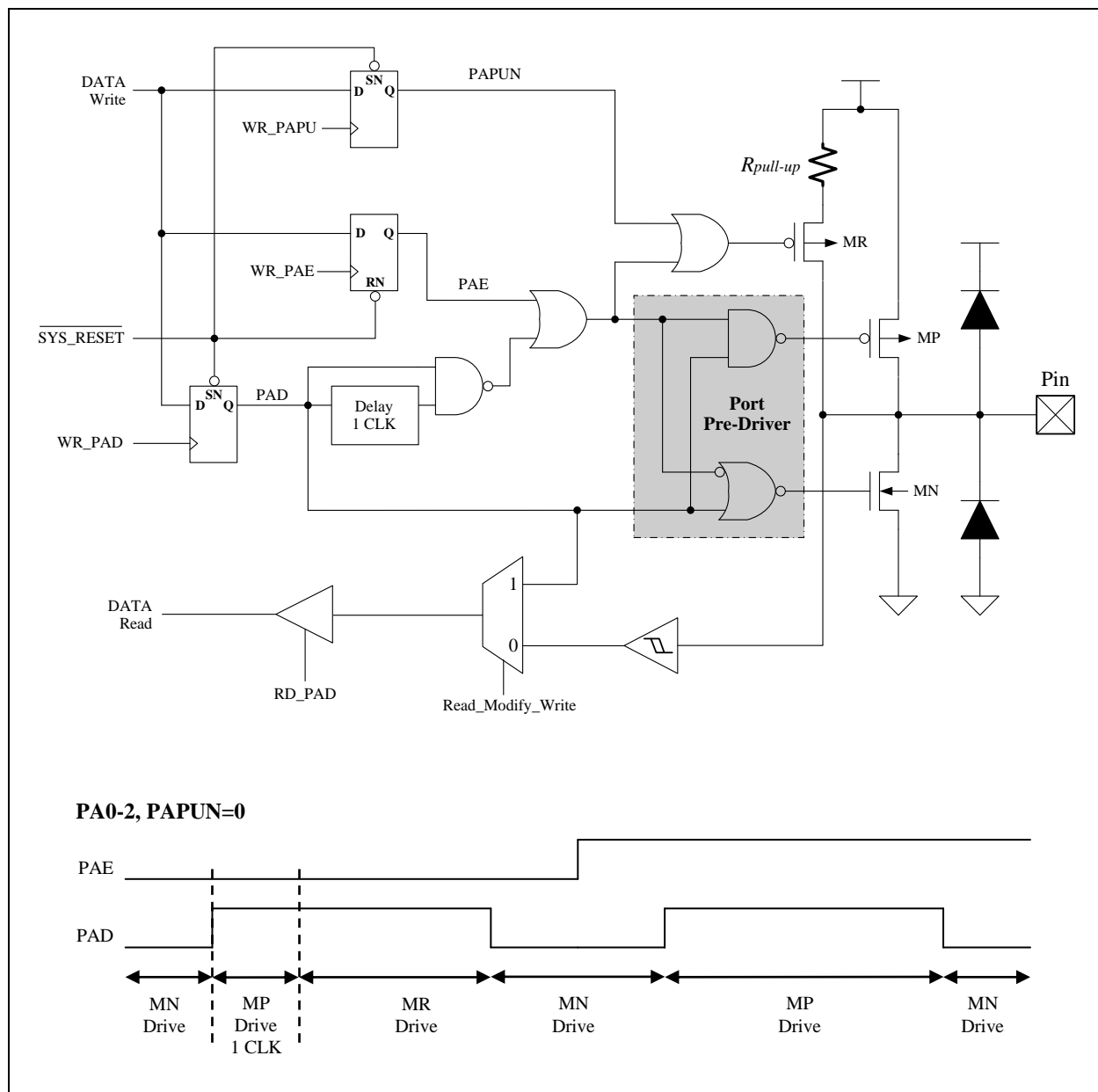
```

BCF      SLOWEN        ; Disable Slow-clock.
SLEEP                                ; Enter STOP mode.
    
```

3. I/O Port

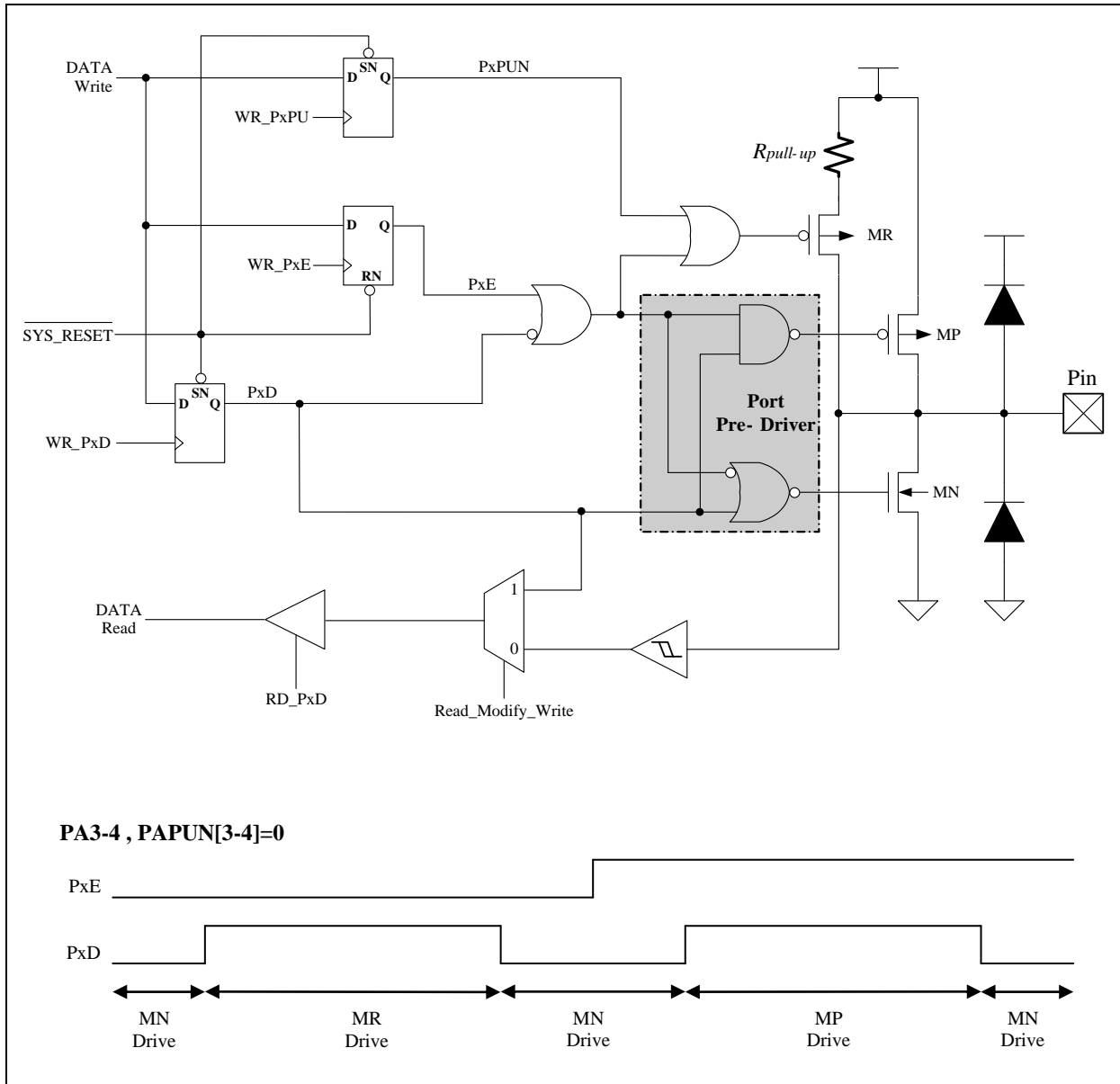
3.1 PA0-2

These pins can be used as Schmitt-trigger input, CMOS push-pull output or “pseudo-open-drain” 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 PAE=0 and PAD=1. To use the pin in pseudo-open-drain mode, S/W sets the PAE=0. The benefit of pseudo-open-drain structure is that the output rise time can be much faster than pure open-drain structure. S/W sets PAE=1 to use the pin in CMOS push-pull output mode. Reading the pin data (PAD) 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 BSF, BCF and all instructions using F-Plane as destination.



3.2 PA3-4

These pins are almost the same as PA0-2, except they do not support pseudo-open-drain mode. They can be used in pure open-drain mode, instead.



◇Example: I/O mode selecting

```
MOVLW    FFH
MOVWF    PAD
MOVLW    00H
MOVWR    PAE           ; Set all ports to be Schmitt-trigger input
```

◇Example: Set PA0-2 as pseudo-open-drain mode

```
MOVLW    0000000B
MOVWR    PAE           ; Set PA2-PA0 as pseudo-open-drain mode

MOVLW    0000000B
MOVWF    PAD           ; PA2~PA0 output low level
```

◇Example: Set PA0-2 is CMOS push-pull output mode.

```
MOVLW    0000111B
MOVWR    PAE           ; Set PA2-PA0 as CMOS push-pull output mode
```

◇Example: Read data from input port.

```
MOVFW    PAD           ; Read data from Port A
```

◇Example: Write data to output port.

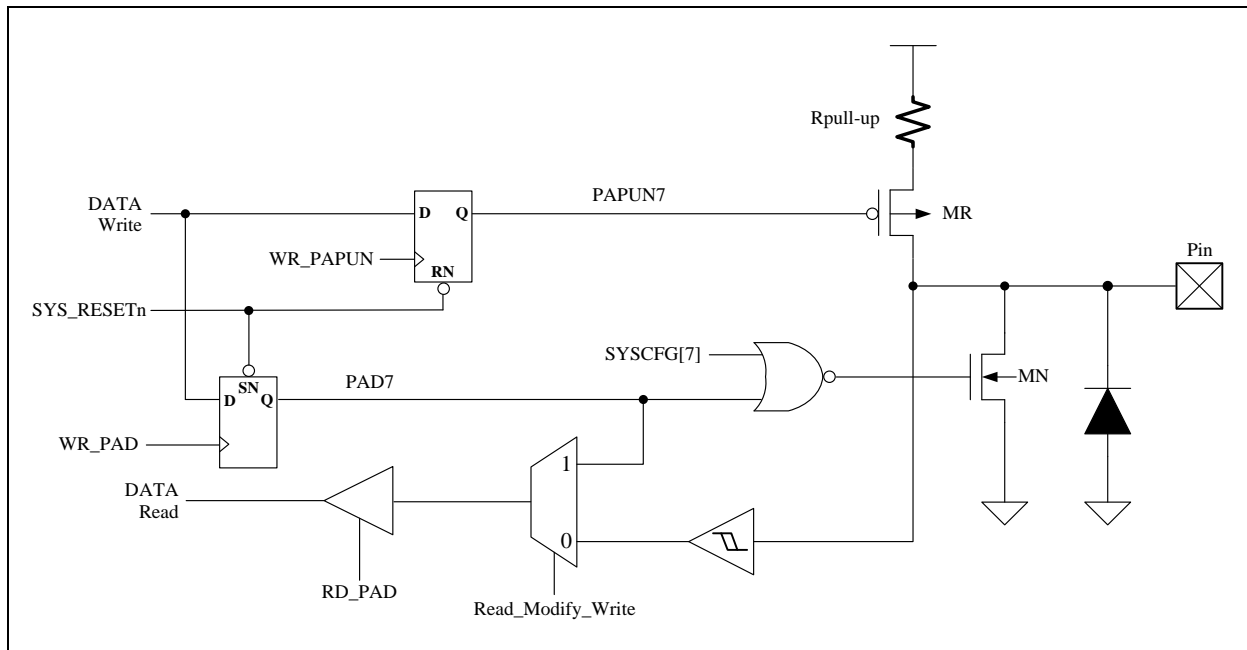
```
MOVLW    55H
MOVWF    PAD           ; Write data 55H to Port A
```

◇Example: Write one bit data to output port.

```
BCF      PAD,0         ; Set PA0 to be "0"
BSF      PAD,1         ; Set PA1 to be "1"
```

3.3 PA7

PA7 can be used in Schmitt-trigger input or open-drain output which is setting by the PAD[7] (F05.7) bit. When the PAD[7] bit is set, PA7 is assigned as Schmitt-trigger input mode, otherwise is assigned as open-drain output mode and output low. The pull-up resistor connected to this pin default, and can be disabled by S/W. In open-drain output mode, the pull-up resistor will not be disabled automatically. The pull-up resistor can be disabled by S/W in open-drain output mode for power saving.



How to control PA7 status can be concluded as following list.

SYSCFG[7]	PAD7	PAPAN7	PN STATE	Pull-up	MODE
0	0	0	Low	Yes	open-drain output with pull-high (not suggest to use this mode)
0	0	1	Low	No	open-drain output without pull-high
0	1	0	High	Yes	input with pull-high
0	1	1	Hi-Z	No	input without pull-high
1	X	0	High	Yes	reset input with pull-high
1	X	1	Hi-Z	No	reset input without pull-high

◇Example: Read state from PA7.

Condition: SYSCFG[7] is set to “0”. If SYSCFG[7] = “1”, then PA7 pin is external reset pin function.

```

BTFSS    PAD,7
GOTO     LOOP_A    ; If PA7=0.
GOTO     LOOP_B    ; If PA7=1.
    
```

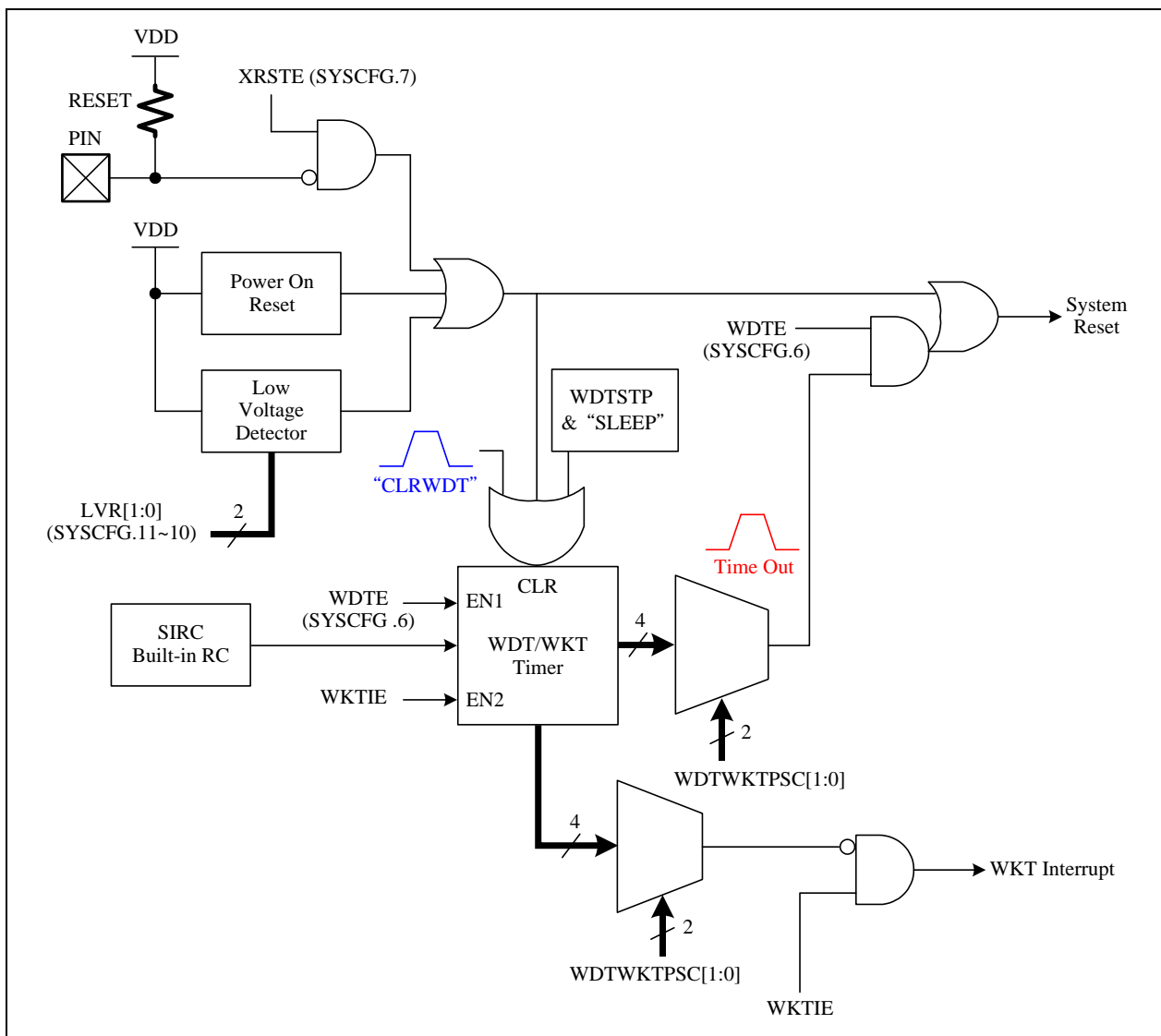
4. Peripheral Functional Blocks

4.1 Watchdog (WDT) /Wakeup (WKT) Timer

The WDT and WKT share the same internal RC Timer (SIRC). The overflow period of WDT, WKT can be selected by WDTWKTTPSC (R08.1~0). The WDT timer is cleared by the CLRWDT instruction. If the Watchdog is enabled (WDTE=1), the WDT generates the chip reset signal while WDT timer always keeps counting even if the SLEEP instruction is executed.

The WDTWKTTPSC also control the WKT interrupt interval, if WKT time is up, it will generate WKT Interrupt Flag (WKTIF). The WKT will not generate WKT interrupt if WKTIE=0. Set WKTIE=1, the WKT will generate interrupt when time is up.

The WDT and WKT functions are mutually exclusive. That means, if WDTE=1 the system only generate WDT timeout reset and WKT will not generate interrupt. On the other hand, if WDTE=0 the WKT interrupt can be activated.



WDT/WKT Block Diagram

Watchdog timer clear can be achieved by CLRWDT instruction or moving any value into WDTCLR (R04).

◇Example: Clear watchdog timer by CLRWDT instruction.

```
MAIN:
...                               ; Execute program
CLRWDT                            ; Execution of CLRWDT instruction
...
GOTO    MAIN
```

◇Example: Clear watchdog timer by writing WDTCLR register.

```
MAIN:
...                               ; Execute program
MOVWR    WDTCLR                  ; Write any value into WDTCLR register
...
GOTO    MAIN
```

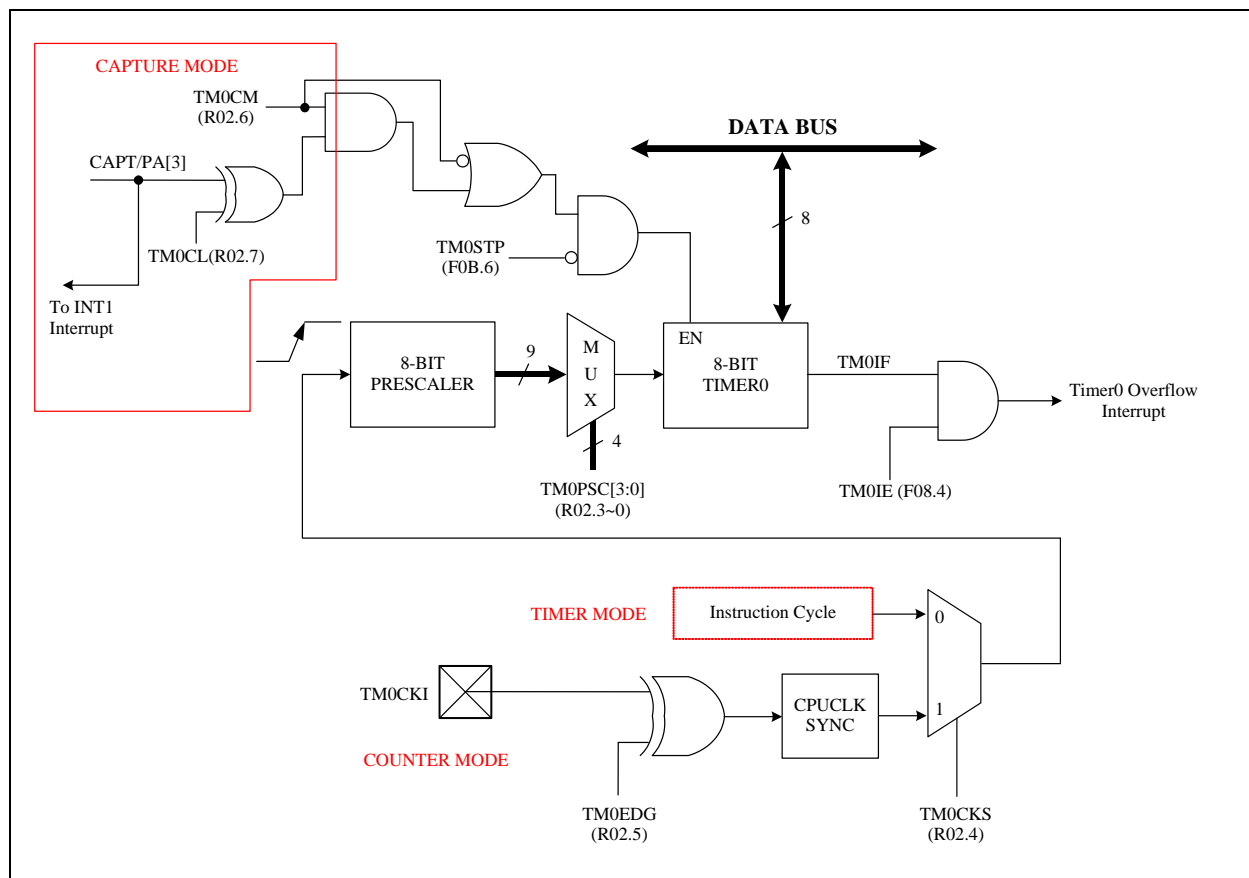
◇Example: Set WKT period and interrupt function.

```
MOVLW    00000010B
MOVWR    R08                    ; Select WKT period=24 ms @5V (default 97 ms)
MOVLW    11110111B            ; Clear WKT interrupt request flag
MOVWF    INTIF
BSF      WKTIE                  ; Enable WKT interrupt function
```

4.2 Timer0

The Timer0 is an 8-bit wide register of F-Plane 01h (TM0). It can be read or written as any other register of F-Plane. Besides, Timer0 increases itself periodically and automatically rolls over based on the pre-scaled clock source, which can be the instruction cycle or TM0CKI (PA3) rising/falling input. The Timer0 increase rate is determined by “Timer0 Pre-Scale” (TM0PSC) register in R-Plane. The Timer0 can generate interrupt flag (TM0IF) when it counts to rolls over if Timer0 Interrupt enable (TM0IE) is set. Timer0 can be stopped counting if the TM0STP bit is set.

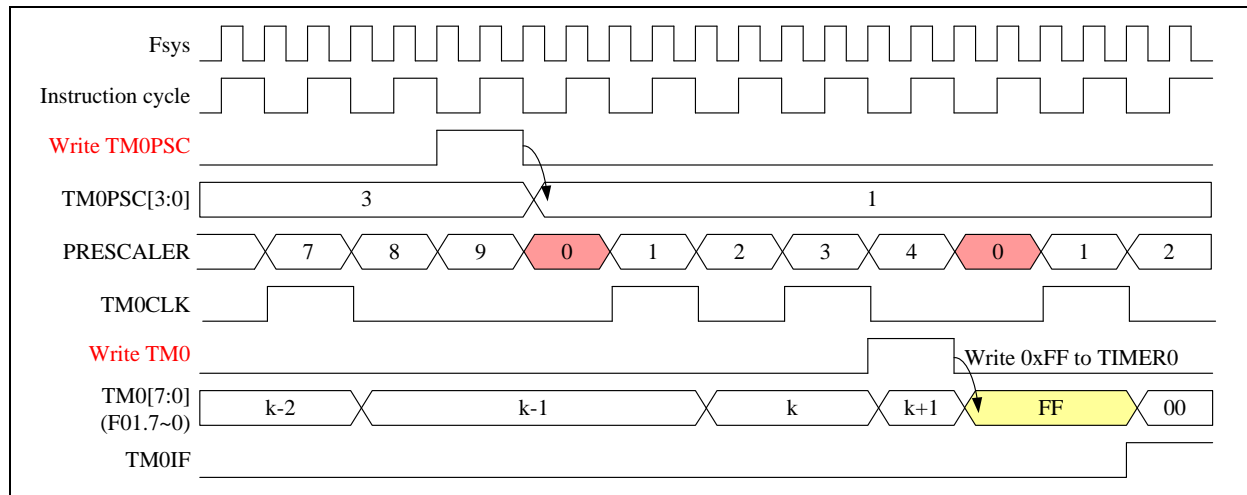
Timer0 can be configured as Capture mode. If TM0CM bit is set to “1”, Timer0 will not count until the CAPT pin (i.e. PA3) is high level (TM0CL=0) or low level (TM0CL=1).



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. Write TM0 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 00h, TM0IF (Timer0 Interrupt Flag) will be set to 1 and generate interrupt if TM0IE (Timer0 Interrupt Enable) is set.



Timer0 works in Timer mode

◇Example:

Setup TM0 Work in Timer mode.

; Setup TM0 clock source and divided.

```

MOVLW 0000101B
MOVWR R02 ; Setup TM0=Timer mode
; TM0 clock source=Instruction cycle
; Divided by 32

```

; Set TM0 timer.

```

BSF TM0STP ; Disable TM0 counting (Default "0")
MOVLW 156
MOVWF TM0 ; Write 156 into TM0 register of F-Plane

```

; Enable TM0 timer and interrupt function.

```

MOVLW 11101111B ; Clear TM0IF interrupt flag
MOVWF INTIF
BSF TMOIE ; Enable TM0 interrupt function
BCF TM0STP ; Enable TM0 counting (Default "0")

```

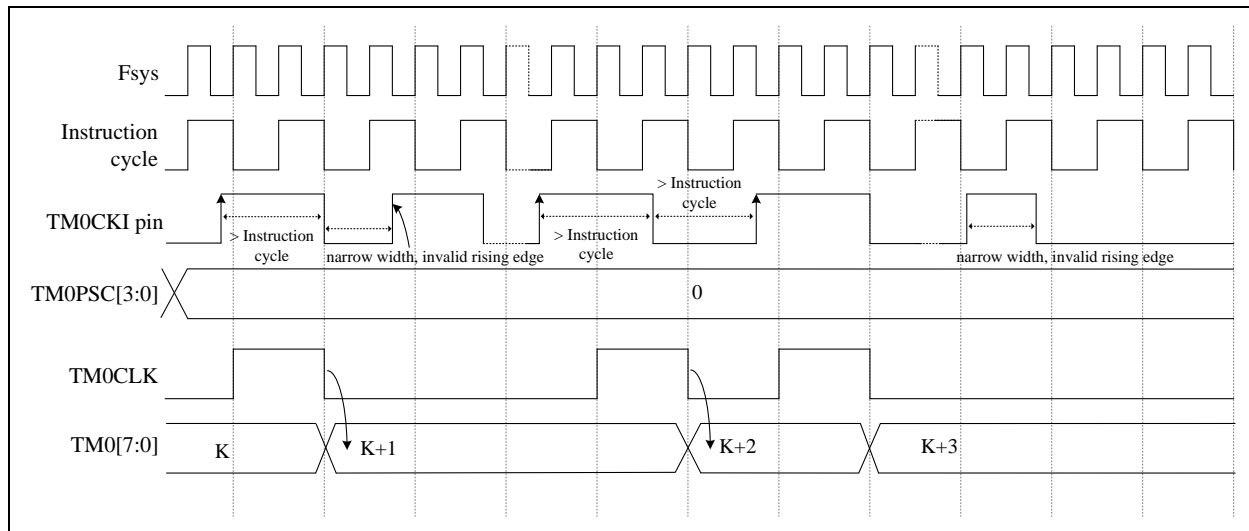
Example:

TM0 clock source is Fsys=4 MHz, Instruction cycle=2 MHz, TM0PSC=/32, TM0=156,

TM0 interrupt time= (1/2MHz) * 32*(256-156) =1.6 ms.

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

If TM0CKS=1 TM0 counter source clock is from TM0CKI pin. TM0CKI signal is synchronized by instruction cycle, which means the high/low time durations of TM0CKI must be longer than one instruction cycle time to guarantee each TM0CKI's change will be detected correctly by the synchronizer.



Timer0 works in Counter mode for TM0CKI (TM0EDG=0)

◇Example:

Setup TM0 Work in counter mode and clock source from TM0CKI pin (PA3) configuration.

; Setup TM0 clock source from TM0CKI pin (PA3) and divide.

```

MOVLW    00010000B
MOVWR    R02           ; Setup TM0=Counter mode.
                        ; Select TM0 prescaler counting edge=rising edge.
                        ; TM0 clock source=TM0CKI pin (PA3)
                        ; Divided by 1

```

; Set TM0 timer and stop TM0 counting.

```

BSF      TM0STP       ; Disable TM0 counting (Default "0").
MOVLW    00H
MOVWF    TM0          ; Write 0 into TM0 register of F-Plane.

```

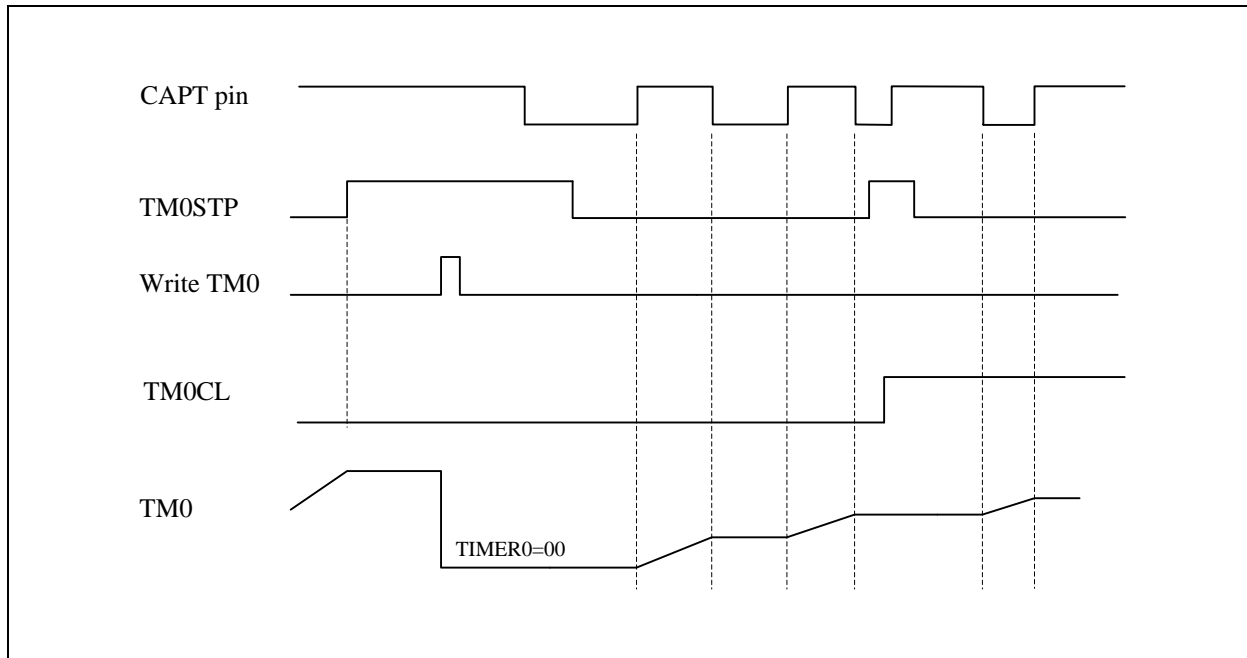
; Start TM0 count and read TM0 count.

```

BCF      TM0STP       ; Enable TM0 counting.
NOP
NOP
NOP
BSF      TM0STP       ; Disable TM0 counting (Default "0")
MOVWF    TM0

```

If only the duty cycle (CAPT high time) needs to be measured, TM0 can be used to measure the duty cycle of the pulse on CAPT pin. In such case, user can set the TM0CM=1. Timer0 is counting up only when CAPT pin is '1'. Note that the internal prescaler will be kept to next Timer0 count, so it will not lose the counting accuracy.



Timer0 is used to measure the high (or low) time on CAPT pin

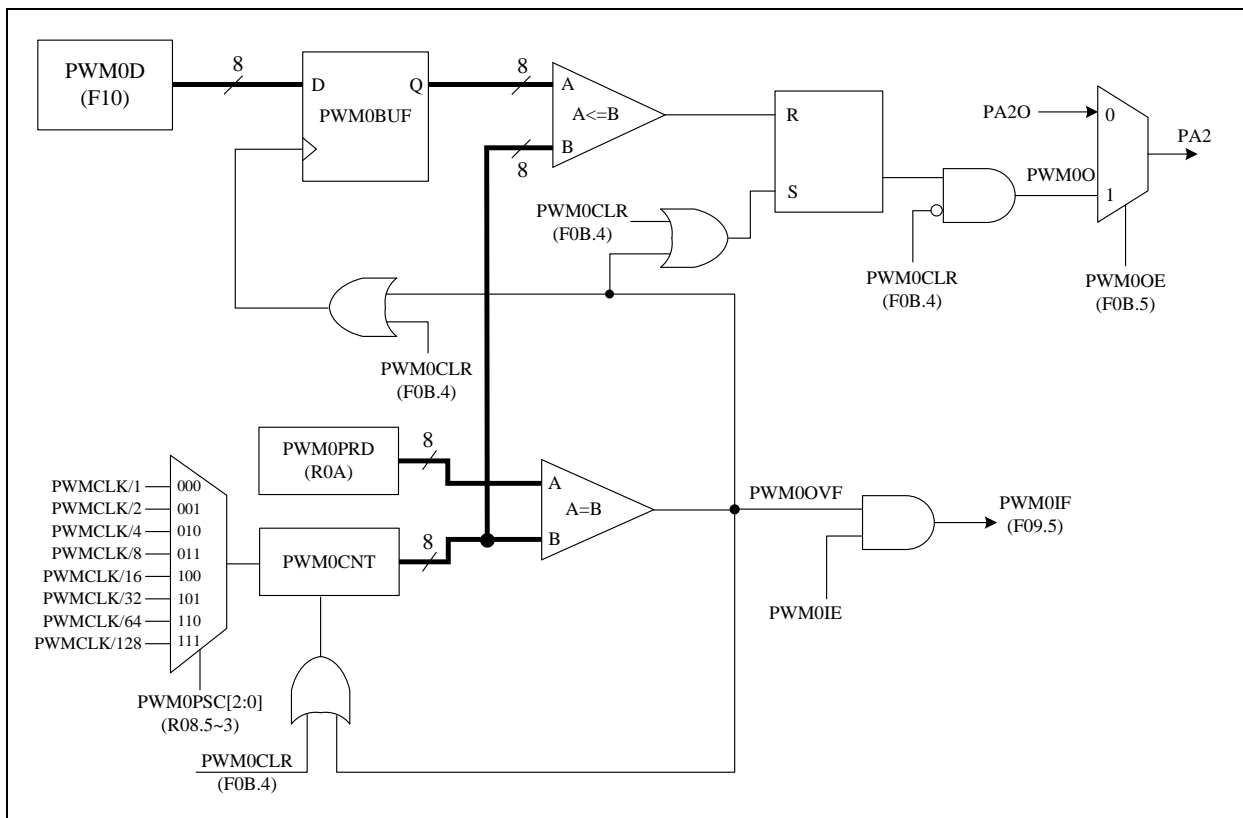
4.3 PWM0: 8-bit PWM

The chip has a built-in 8-bit PWM generator. The source clock comes from PWMCLK divided by 1, 2, 4, 8, 16, 32, 64, and 128. The PWM0 duty cycle can be changed with writing to PWM0D (F10), writing to PWM0D (F10) will not change the current PWM0 duty until the current PWM0 period completes. When current PWM0 period is finish, the new value of PWM0D (F10) will be updated to the PWM0BUF.

The PWM0 can be individually output to PA2 by PWM0OE (F0B.5) is set to 1. Setting the PWM0CLR (F0B.4) bit will clear the PWM0 counter and load the PWM0D (F10) to PWM0BUF, PWM0CLR (F0B.4) bit must be cleared so that the PWM0 counter can count. The following figure shows the block diagram of PWM0.

The clock source of PWMCLK is from either IRC 8 MHz or IRC 16 MHz which is generated from frequency doubler. PWMCKS (F0D.3) is used to select which clock will be PWMCLK.

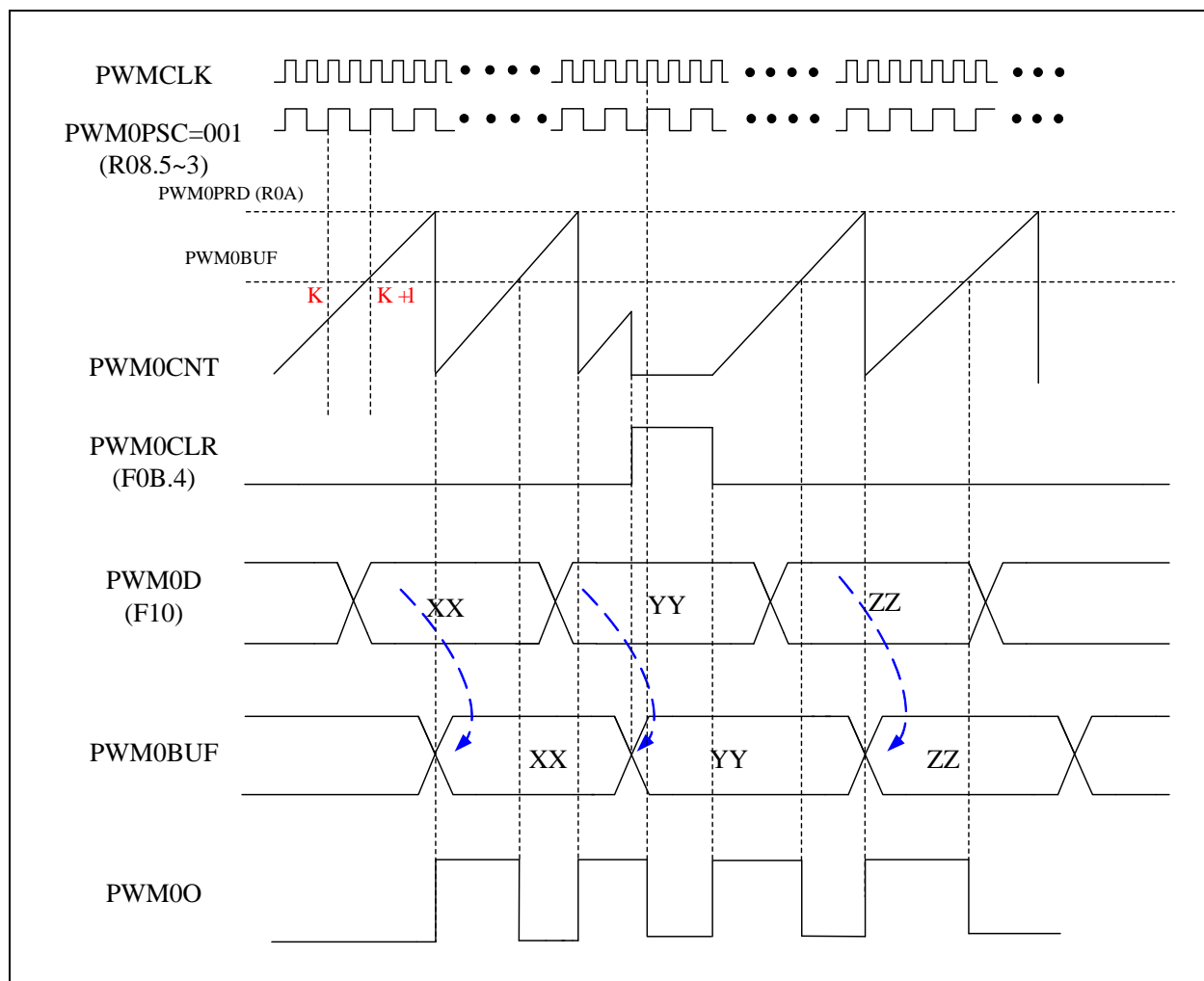
PWM0IF denotes PWM0 period finish interrupt. If PWM0IE=1, the interrupt will be generated when PWM0 finishes one cycle period. It acts as another timer for user specific applications. The programmers must take care interrupt time and Fsys clock relative timing issues to prevent dead lock. (i.e. faster PWMCLK and less PWM0PSC makes frequently PWM0 interrupt makes CPU lose main program executions.)



PWM0 Block Diagram

The following figure shows the PWM0 waveforms. When PWM0CLR (F0B.4) bit is set to '1', the PWM0 output is cleared to '0' no matter what its current status is. Once the PWM0CLR (F0B.4) bit is cleared to '0', the PWM0 output is set to '1' to begin a new PWM cycle. PWM0 output will be '0' when PWM0CNT is greater than or equals to PWM0BUF. PWM0CNT keeps counting up when equals to PWM0PRD (R0A), the PWM0 output is set to '1' again.

The PWM0 period can be set by writing period value to PWM0PRD (R0A) register. Note that changing the PWM0PRD (R0A) immediately changes the PWM0PRD (R0A) value in the Figure that is different from PWM0D (F10) which has PWM0BUF to update the duty at the end of current period. The Programmer must pay attention to the current time to change PWM0PRD (R0A) by observing the following figure. There is a digital comparator that compares the PWM0CNT and PWM0PRD (R0A), if PWM0CNT is larger than PWM0PRD (R0A) after setting the PWM0PRD (R0A), a fault long PWM cycle will be generated because PWM0CNT must count to overflow then keep counting to PWM0PRD (R0A) to finish the cycle.



PWM0 waveform

Example:

[CPU running at Fast mode , Fsys=4 MHz, PWMCLK=16 MHz]

◇Example:

```
; Setup PWM0 clock prescaler
BSF      PWMCKS      ; Set 16MHz as PWMCLK
MOVLW   00010000B   ; PWM0PSC=010 (divided by 4)
MOVWR   R08         ; PWM0 clock source PWMCLK/4=16 MHz/4=4 MHz

MOVLW   80H
MOVWF   PWM0D       ; Set PWM0 duty=80H/FFH=50%

MOVLW   0FFH
MOVWR   PWM0PRD     ; Set PWM0 period=FFH

BSF     PWM0OE      ; Enable PWM0 output to PA2
BCF     PWM0CLR     ; Enable PWM0 counting
```

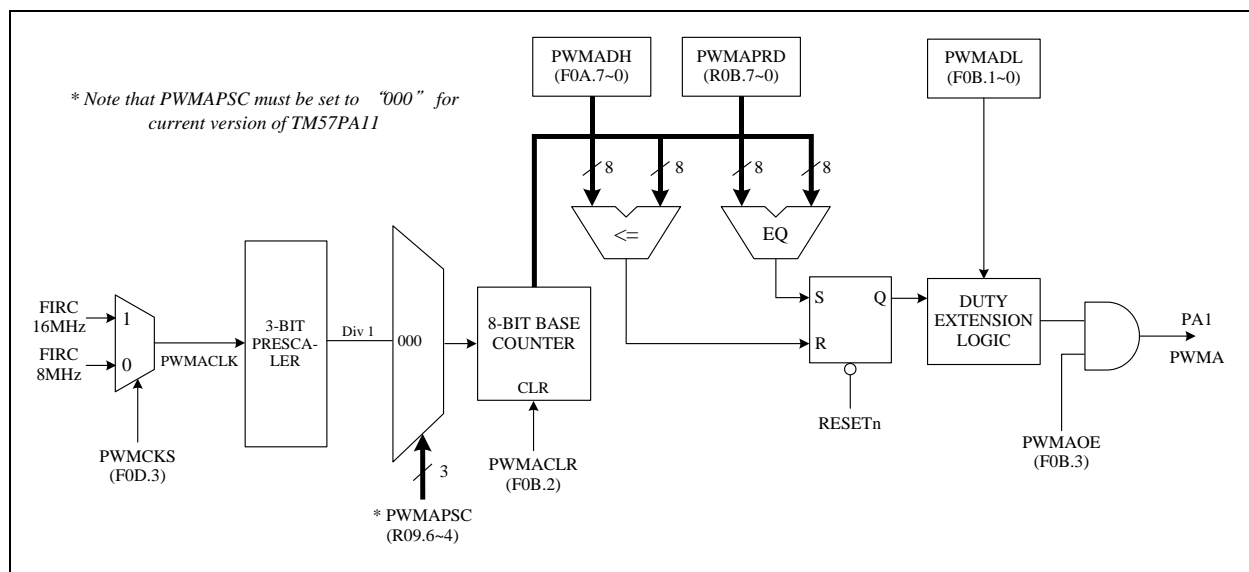
Example:

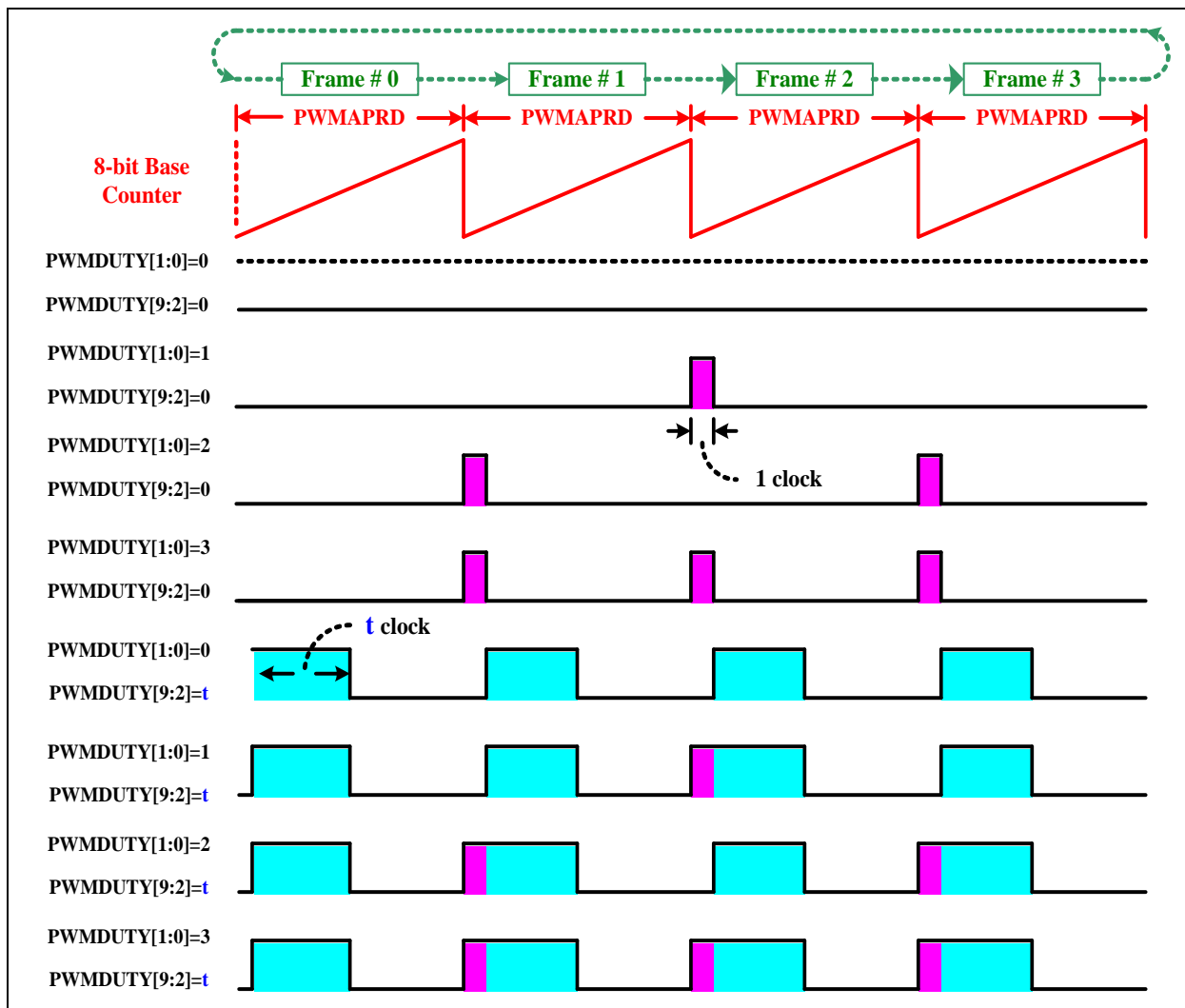
PWMCLK=16 MHz, PWM0PSC=/4, PWM0PRD=FFH, PWM0D=80H,
PWM0 output frequency=4 MHz/PWM0PRD=4 MHz/256=15.625 KHz.

4.4 PWMA: (8+2) bits PWM

The PWMA can generate various frequency waveforms with 1024 duty resolution based on PWMCLK. A spread LSB technique allows PWMA to run its frequency at “PWMCLK divided by 256” instead of “PWMCLK divided by 1024”, which means the frequency of PWMA is 4 times higher than normal. The advantage of higher PWMA frequency is that the post RC low-pass filter can transform the PWM signal to more stable DC voltage level. The PWMA output signal reset to low level whenever the 8-bit base counter matches the 8-bit MSB of PWMA duty register PWMADH (F0A). When the base counter rolls over, the 2-bit LSB of PWMA duty register PWMADL (F0B.1~0) decides whether to set the PWMA output signal high immediately or set it high after one clock cycle delay.

PWMA_PSC is not be implemented in this version, user must set PWMA_PSC to “000” to prevent malfunction.





Example:

[CPU running at Fast mode, PWMCLK=8 MHz]

◇Example:

```

BCF      PWMCKS      ; IRC 8 MHz as PWMCLK
MOVLW   00001000B   ; PWMAPSC=000, CKDIV=10 (divided by 2)
MOVWR   R09         ; PWMACLK=8 MHz

MOVLW   80H
MOVWR   PWMAPRD     ; Set PWMA period=80H

MOVLW   00010100B
MOVWF   F0B         ; Set PWMADL duty=00H

MOVLW   20H
MOVWF   PWMADH     ; Set PWMADH duty=20H

BSF     PWMAOE      ; Enable PWMA OUT (PA0)
BCF     PWMACLR     ; Enable PWMA counting
    
```

Example:

PWMACLK=8 MHz, PWMAPRD=80H,

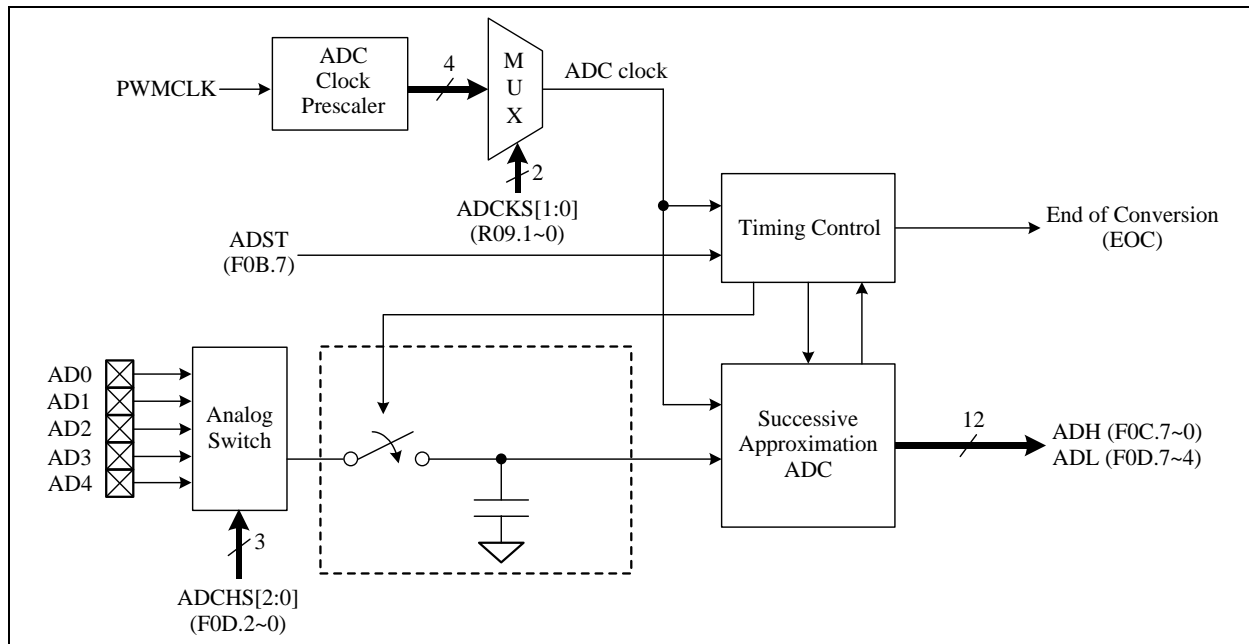
PWMADL=00H, PWMADH=20H

PWMA output frequency=8 MHz/ (PWMAPRD+1) =8 MHz/129=62 KHz.

PWMA output duty=32:129=24.8%.

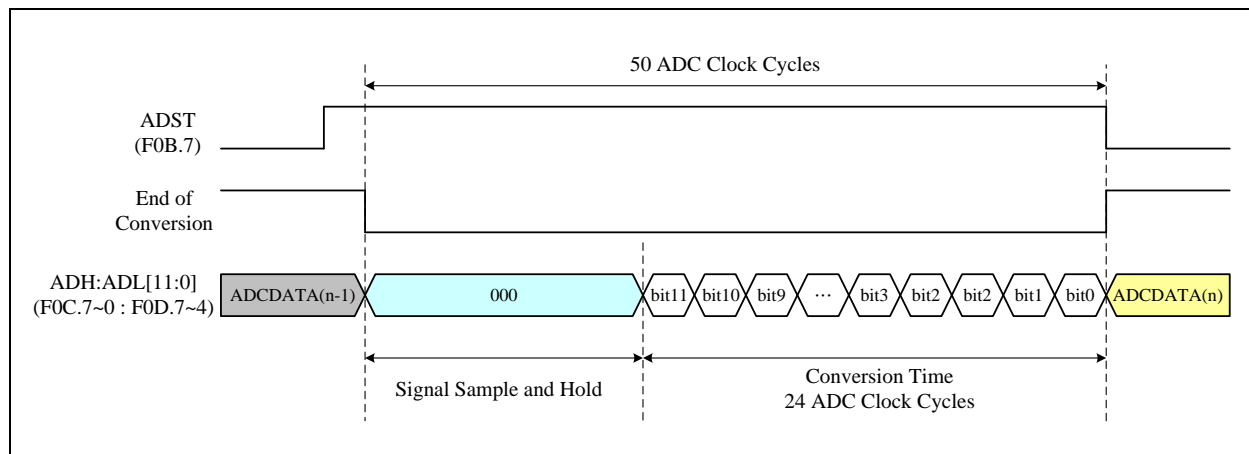
PWMAIF denotes PWMA period finish interrupt. If PWMAIE=1, the interrupt will be generated when PWMA reach the end of Frame #3 which is the end of one cycle period. It acts as another timer for user specific applications. The programmers must take care interrupt time and Fsys clock relative timing issues to prevent from dead lock. (i.e. faster PWMCLK makes frequently PWMA interrupt makes CPU lose main program executions.)

4.5 Analog-to-Digital Converter



The 12-bit ADC (Analog to Digital Converter) consists of a 5-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(R09.1~0) to choose a proper ADC clock frequency, which must be less than 1 MHz. User then launches the ADC conversion by setting the ADST (F0B.7) control bit. After end of conversion, H/W automatic clears the ADST (F0B.7) bit. User can poll this bit to know the conversion status. The PAM (R07.4~0) control register is used for ADC pin type setting, user can write the corresponding bit to “0” when the pin is used as an ADC input. The setting can disable the pin logical input path to save power consumption.

The ADC clock source is from PWMCLK passing through a clock divider and four clock rates can be select, which is divided by 16, 32, 64, and 128, respectively. The control register ADCKS (R09.1~0) select one of the four clocks to be ADC clock.



Example:

[CPU running at Fast mode , Fsys=FIRC 8 MHz]
 ADC clock frequency=500 KHz, ADC channel=AD2 (PA2).

◇Example:

```

BCF      PWMCKS      ; IRC 8 MHz as PWMCLK
MOVLW   00001100B  ; ADCKS=00 (PWMCLK/16), CKDIV=11 (divided by 1)
MOVWR   R09          ; Fsys=FIRC 8 MHz/1=8 MHz, ADC clock=8 MHz/16

MOVLW   11111011B
MOVWR   PAM          ; Enable PA2 pin (AD2) analog input

MOVLW   00000010B
MOVWF   F0D          ; ADC channel select AD2 (PA2 pin)

BSF     ADST         ; ADC start conversion

WAIT_ADC:
BTFSC   ADST         ; Wait ADC conversion
GOTO    WAIT_ADC

MOVFW   ADH          ; Read ADC value [11:4]
MOVWF   ADC_MSB
MOVFW   F0D          ; Read ADC value[3:0]
ANDLW   F0H          ; mask the bit-0 to bit-3
MOVWF   ADC_LSB     ; ADC_LSB.7~4 stores the least 4 bits of ADC result
...
  
```


4.6 System Clock Oscillator

System clock can be operated in two different oscillation modes, which is selected by setting the FIRCSTP, CPUCKS, CKDIV, and SLOWEN control bits. In the fast internal RC (FIRC) mode, the on-chip oscillator generates 8 MHz clock, which is divided to 1 MHz, 2 MHz, 4 MHz, and 8 MHz to be the system clock (Fsys) depends on CKDIV setting.

In the slow internal RC (SIRC) mode, the on-chip oscillator generates about 167 KHz at VDD=5V. The SIRC clock also can be divided by four clock rates to be system clock (Fsys) depends on CKDIV setting.

R-Plane

Name	Adr	R/W	Rst	Description
(R02) TM0CTL Function related to: TM0				
TM0CL	02.7	W	0	Timer0 Capture Mode Level 0 : High level capture 1 : Low level capture
TM0CM	02.6	W	0	Timer0 Mode 0: Timer/Counter Mode Clock source from TM0PSC (set R02.3~0) TM0CKI (set R02.4) 1: Capture Mode Clock source from CAPT pin
TM0EDG	02.5	W	0	Timer0 prescaler counting edge for TM0CKI pin 0: rising edge 1: falling edge
TM0CKS	02.4	W	0	Timer0 prescaler clock source 0: Instruction cycle 1: TM0CKI pin (PA3 pin)
TM0PSC	02.3~ 0	W	0	Timer0 prescaler. Timer0 prescaler clock source divided by 0000: /1 0001: /2 0010: /4 0011: /8 0100: /16 0101: /32 0110: /64 0111: /128 1xxx: /256
(R03) PWRDN Function related to: POWER DOWN				
PWRDN	03	W	-	write this register to enter Power-Down Mode (i.e. 'SLEEP' instruction)
(R04) WDTCLR Function related to: WDT				
WDTCLR	04	W	-	write this register to clear WDT timer
(R05) PAE Function related to: Port A				
PAE	05.4~ 3	W	0	Each bit controls its corresponding pin, if the bit is 0 : the pin is open-drain output or Schmitt-trigger input 1 : the pin is CMOS push-pull output
	05.2~ 0	W	0	Each bit controls its corresponding pin, if the bit is 0 : the pin is pseudo-open-drain output or Schmitt-trigger input 1 : the pin is CMOS push-pull output
(R06) PAPUN Function related to: Port A				
PAPUN7	06.7	W	0	PA7 pull up resistor control. 0: enable 1:disable
PAPUN	06.4~ 0	W	1F	Each bit controls its corresponding pin, if the bit is 0: the pin pull up resistor is enabled, except a. the pin's output data register (PAD) is 0 b. the pin's CMOS push-pull mode is chosen (PAE=1) c. the pin is working for PWM0/PWMA output 1: the pin pull up resistor is disabled

INSTRUCTION SET

Each instruction is a 14-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” or “r” 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	F-Plane Register File Address
r	R-Plane 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					
<u>ADDWF</u>	f,d	00 0111 dfff ffff	1	C, DC, Z	Add W and "f"
<u>ANDWF</u>	f,d	00 0101 dfff ffff	1	Z	AND W with "f"
<u>CLRF</u>	f	00 0001 1fff ffff	1	Z	Clear "f"
<u>CLRW</u>		00 0001 0100 0000	1	Z	Clear W
<u>COMF</u>	f,d	00 1001 dfff ffff	1	Z	Complement "f"
<u>DECF</u>	f,d	00 0011 dfff ffff	1	Z	Decrement "f"
<u>DECFSZ</u>	f,d	00 1011 dfff ffff	1 or 2	-	Decrement "f", skip if zero
<u>INCF</u>	f,d	00 1010 dfff ffff	1	Z	Increment "f"
<u>INCFSZ</u>	f,d	00 1111 dfff ffff	1 or 2	-	Increment "f", skip if zero
<u>IORWF</u>	f,d	00 0100 dfff ffff	1	Z	OR W with "f"
<u>MOVFW</u>	f	00 1000 0fff ffff	1	-	Move "f" to W
<u>MOVWF</u>	f	00 0000 1fff ffff	1	-	Move W to "f"
<u>MOVWR</u>	r	00 0000 00rr rrrr	1	-	Move W to "r"
<u>RLF</u>	f,d	00 1101 dfff ffff	1	C	Rotate left "f" through carry
<u>RRF</u>	f,d	00 1100 dfff ffff	1	C	Rotate right "f" through carry
<u>SUBWF</u>	f,d	00 0010 dfff ffff	1	C, DC, Z	Subtract W from "f"
<u>SWAPF</u>	f,d	00 1110 dfff ffff	1	-	Swap nibbles in "f"
<u>TESTZ</u>	f	00 1000 1fff ffff	1	Z	Test if "f" is zero
<u>XORWF</u>	f,d	00 0110 dfff ffff	1	Z	XOR W with "f"
Bit-Oriented File Register Instruction					
<u>BCF</u>	f,b	01 000b bbff ffff	1	-	Clear "b" bit of "f"
<u>BSF</u>	f,b	01 001b bbff ffff	1	-	Set "b" bit of "f"
<u>BTFSC</u>	f,b	01 010b bbff ffff	1 or 2	-	Test "b" bit of "f", skip if clear
<u>BTFSS</u>	f,b	01 011b bbff ffff	1 or 2	-	Test "b" bit of "f", skip if set
Literal and Control Instruction					
<u>ADDLW</u>	k	01 1100 kkkk kkkk	1	C,DC,Z	Add Literal "k" and W
<u>ANDLW</u>	k	01 1011 kkkk kkkk	1	Z	AND Literal "k" with W
<u>CALL</u>	k	10 kkkk kkkk kkkk	2	-	Call subroutine "k"
<u>CLRWD_T</u>		00 0000 0000 0100	1	TO,PD	Clear Watch Dog Timer
<u>GOTO</u>	k	11 kkkk kkkk kkkk	2	-	Jump to branch "k"
<u>IORLW</u>	k	01 1010 kkkk kkkk	1	Z	OR Literal "k" with W
<u>MOVLW</u>	k	01 1001 kkkk kkkk	1	-	Move Literal "k" to W
<u>NOP</u>		00 0000 0000 0000	1	-	No operation
<u>RET</u>		00 0000 0100 0000	2	-	Return from subroutine
<u>RETI</u>		00 0000 0110 0000	2	-	Return from interrupt
<u>RETLW</u>	k	01 1000 kkkk kkkk	2	-	Return with Literal in W
<u>SLEEP</u>		00 0000 0000 0011	1	TO,PD	Go into standby mode, Clock oscillation stops
<u>XORLW</u>	k	01 1111 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	01 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

ADDWF

Add W and "f"

Syntax	ADDWF f [,d]	
Operands	f : 00h ~ 7Fh, d : 0, 1	
Operation	$(\text{destination}) \leftarrow (W) + (f)$	
Status Affected	C, DC, Z	
OP-Code	00 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	ADDWF 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	01 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

ANDWF

AND W with "f"

Syntax	ANDWF f [,d]	
Operands	f : 00h ~ 7Fh, d : 0, 1	
Operation	$(\text{destination}) \leftarrow (W) \text{ AND } (f)$	
Status Affected	Z	
OP-Code	00 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	ANDWF FSR, 1	B : W = 0x17, FSR = 0xC2 A : W = 0x17, FSR = 0x02

BCF Clear "b" bit of "f"

Syntax	BCF f [,b]	
Operands	f : 00h ~ 3Fh, b : 0 ~ 7	
Operation	(f.b) ← 0	
Status Affected	-	
OP-Code	01 000b bbff ffff	
Description	Bit 'b' in register 'f' is cleared.	
Cycle	1	
Example	BCF FLAG_REG, 7	B : FLAG_REG = 0xC7 A : FLAG_REG = 0x47

BSF Set "b" bit of "f"

Syntax	BSF f [,b]	
Operands	f : 00h ~ 3Fh, b : 0 ~ 7	
Operation	(f.b) ← 1	
Status Affected	-	
OP-Code	01 001b bbff ffff	
Description	Bit 'b' in register 'f' is set.	
Cycle	1	
Example	BSF FLAG_REG, 7	B : FLAG_REG = 0x0A A : FLAG_REG = 0x8A

BTFSK Test "b" bit of "f", skip if clear(0)

Syntax	BTFSK f [,b]	
Operands	f : 00h ~ 3Fh, b : 0 ~ 7	
Operation	Skip next instruction if (f.b) = 0	
Status Affected	-	
OP-Code	01 010b bbff 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 BTFSK FLAG, 1 TRUE GOTO SUB1 FALSE ...	B : PC = LABEL1 A : if FLAG.1 = 0, PC = FALSE if FLAG.1 = 1, PC = TRUE

BTFSK Test "b" bit of "f", skip if set(1)

Syntax	BTFSK f [,b]	
Operands	f : 00h ~ 3Fh, b : 0 ~ 7	
Operation	Skip next instruction if (f.b) = 1	
Status Affected	-	
OP-Code	01 011b bbff 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 BTFSK FLAG, 1 TRUE GOTO SUB1 FALSE ...	B : PC = LABEL1 A : if FLAG.1 = 0, PC = TRUE if FLAG.1 = 1, PC = FALSE

COMF	Complement "f"
Syntax	COMF f [,d]
Operands	f : 00h ~ 7Fh, d : 0, 1
Operation	(destination) ← (\bar{f})
Status Affected	Z
OP-Code	00 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	COMF REG1, 0 B : REG1 = 0x13 A : REG1 = 0x13, W = 0xEC

DECF	Decrement "f"
Syntax	DECF f [,d]
Operands	f : 00h ~ 7Fh, d : 0, 1
Operation	(destination) ← (f) - 1
Status Affected	Z
OP-Code	00 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	DECF CNT, 1 B : CNT = 0x01, Z = 0 A : CNT = 0x00, Z = 1

DECFSZ	Decrement "f", Skip if 0
Syntax	DECFSZ f [,d]
Operands	f : 00h ~ 7Fh, d : 0, 1
Operation	(destination) ← (f) - 1, skip next instruction if result is 0
Status Affected	-
OP-Code	00 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 DECFSZ CNT, 1 GOTO LOOP CONTINUE B : PC = LABEL1 A : CNT = CNT - 1 if CNT = 0, PC = CONTINUE if CNT ≠ 0, PC = LABEL1 + 1

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

INCF	Increment "f"	
Syntax	INCF f [,d]	
Operands	f : 00h ~ 7Fh	
Operation	(destination) ← (f) + 1	
Status Affected	Z	
OP-Code	00 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	INCF CNT, 1	B : CNT = 0xFF, Z = 0 A : CNT = 0x00, Z = 1

INCFSZ	Increment "f", Skip if 0	
Syntax	INCFSZ f [,d]	
Operands	f : 00h ~ 7Fh, d : 0, 1	
Operation	(destination) ← (f) + 1, skip next instruction if result is 0	
Status Affected	-	
OP-Code	00 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 INCFSZ CNT, 1 GOTO LOOP CONTINUE	B : PC = LABEL1 A : CNT = CNT + 1 if CNT = 0, PC = CONTINUE if CNT ≠ 0, PC = LABEL1 + 1

IORLW	Inclusive OR Literal with W	
Syntax	IORLW k	
Operands	k : 00h ~ FFh	
Operation	(W) ← (W) OR k	
Status Affected	Z	
OP-Code	01 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

IORWF	Inclusive OR W with "f"	
Syntax	IORWF f [,d]	
Operands	f : 00h ~ 7Fh, d : 0, 1	
Operation	(destination) ← (W) OR k	
Status Affected	Z	
OP-Code	00 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	IORWF RESULT, 0	B : RESULT = 0x13, W = 0x91 A : RESULT = 0x13, W = 0x93, Z = 0


MOVFW	Move "f" to W	
Syntax	MOVFW f	
Operands	f : 00h ~ 7Fh	
Operation	(W) ← (f)	
Status Affected	-	
OP-Code	00 1000 0fff ffff	
Description	The contents of register 'f' are moved to W register.	
Cycle	1	
Example	MOVFW 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	01 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


MOVWF	Move W to "f"	
Syntax	MOVWF f	
Operands	f : 00h ~ 7Fh	
Operation	(f) ← (W)	
Status Affected	-	
OP-Code	00 0000 1fff ffff	
Description	Move data from W register to register 'f'.	
Cycle	1	
Example	MOVWF REG1	B : REG1 = 0xFF, W = 0x4F A : REG1 = 0x4F, W = 0x4F

MOVWR	Move W to "r"	
Syntax	MOVWR r	
Operands	r : 00h ~ 3Fh	
Operation	(r) ← (W)	
Status Affected	-	
OP-Code	00 0000 00rr rrrr	
Description	Move data from W register to register 'r'.	
Cycle	1	
Example	MOVWR REG1	B : REG1 = 0xFF, W = 0x4F A : REG1 = 0x4F, W = 0x4F

RLF Rotate Left "f" through Carry

Syntax	RLF f [,d]	
Operands	f : 00h ~ 7Fh, d : 0, 1	
Operation		
Status Affected	C	
OP-Code	00 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	RLF REG1, 0	B : REG1 = 1110 0110, C = 0 A : REG1 = 1110 0110 W = 1100 1100, C = 1

RRF Rotate Right "f" through Carry

Syntax	RRF f [,d]	
Operands	f : 00h ~ 7Fh, d : 0, 1	
Operation		
Status Affected	C	
OP-Code	00 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	RRF 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	00 0000 0000 0011	
Description	Go into Power-down mode with the oscillator stops.	
Cycle	1	
Example	SLEEP	-

SUBWF Subtract W from "f"

Syntax	SUBWF f [,d]	
Operands	f : 00h ~ 7Fh, d : 0, 1	
Operation	(destination) ← (f) – (W)	
Status Affected	C, DC, Z	
OP-Code	00 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	SUBWF REG1, 1	B : REG1 = 0x03, W = 0x02, C = ?, Z = ? A : REG1 = 0x01, W = 0x02, C = 1, Z = 0
	SUBWF REG1, 1	B : REG1 = 0x02, W = 0x02, C = ?, Z = ? A : REG1 = 0x00, W = 0x02, C = 1, Z = 1
	SUBWF REG1, 1	B : REG1 = 0x01, W = 0x02, C = ?, Z = ? A : REG1 = 0xFF, W = 0x02, C = 0, Z = 0

SWAPF Swap Nibbles in "f"

Syntax	SWAPF f [,d]	
Operands	f : 00h ~ 7Fh, d : 0, 1	
Operation	(destination,7~4) ← (f.3~0), (destination.3~0) ← (f.7~4)	
Status Affected	-	
OP-Code	00 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	SWAPF REG, 0	B : REG1 = 0xA5 A : REG1 = 0xA5, W = 0x5A

TESTZ Test if "f" is zero

Syntax	TESTZ f	
Operands	f : 00h ~ 7Fh	
Operation	Set Z flag if (f) is 0	
Status Affected	Z	
OP-Code	00 1000 1fff ffff	
Description	If the content of register 'f' is 0, Zero flag is set to 1.	
Cycle	1	
Example	TESTZ 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	01 1111 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

XORWF	Exclusive OR W with 'f'	
Syntax	XORWF f [,d]	
Operands	f : 00h ~ 7Fh, d : 0, 1	
Operation	(destination) ← (W) XOR (f)	
Status Affected	Z	
OP-Code	00 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	XORWF 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 $V_{SS} + 6.5$	V
Input voltage	$V_{SS} - 0.3$ to $V_{DD} + 0.3$	
Output voltage	$V_{SS} - 0.3$ to $V_{DD} + 0.3$	
Output current high per 1 PIN	-25	mA
Output current high per all PIN	-80	
Output current low per 1 PIN	+30	
Output current low per all PIN	+150	
Maximum Operating Voltage	5.5	V
Operating temperature	-40 to +85	°C
Storage temperature	-65 to +150	

2. DC Characteristics ($T_A=25^\circ\text{C}$, $V_{DD}=2.0\text{ V}$ to 5.5 V , unless otherwise specified)

Parameter	Sym	Conditions		Min	Typ	Max	Unit
Operating Voltage	V_{DD}	FAST mode, 25°C , $F_{OSC}=4\text{ MHz}$		2.2	–	5.5	V
Input High Voltage	V_{IH}	All Input, except PA7	$V_{DD}=5\text{ V}$	$0.6 V_{DD}$	–	V_{DD}	V
			$V_{DD}=3\text{ V}$	$0.6 V_{DD}$	–	V_{DD}	V
		PA7	$V_{DD}=5\text{ V}$	$0.7 V_{DD}$	–	V_{DD}	V
			$V_{DD}=3\text{ V}$	$0.7 V_{DD}$	–	V_{DD}	V
Input Low Voltage	V_{IL}	All Input, except PA7	$V_{DD}=5\text{ V}$	0	–	$0.2 V_{DD}$	V
			$V_{DD}=3\text{ V}$	0	–	$0.2 V_{DD}$	V
		PA7	$V_{DD}=5\text{ V}$	0	–	$0.3 V_{DD}$	V
			$V_{DD}=3\text{ V}$	0	–	$0.3 V_{DD}$	V
Output High Current	I_{OH}	All Output	$V_{DD}=5\text{ V}$, $V_{OH}=4.5\text{ V}$	4	8	–	mA
			$V_{DD}=3\text{ V}$, $V_{OH}=2.7\text{ V}$	1.5	3	–	
Output Low Current	I_{OL}	All Output	$V_{DD}=5\text{ V}$, $V_{OL}=0.5\text{ V}$	10	20	–	mA
			$V_{DD}=3\text{ V}$, $V_{OL}=0.3\text{ V}$	4.5	9	–	
Input Leakage Current (pin high)	I_{ILH}	All Input	$V_{IN}=V_{DD}$	–	–	1	μA
Input Leakage Current (pin low)	I_{ILL}	All Input	$V_{IN}=0\text{ V}$	–	–	-1	μA

Parameter	Sym	Conditions	Min	Typ	Max	Unit	
Supply Current	I _{DD}	FAST mode, FIRC No load All I/O pins with internal pull-up	V _{DD} =5 V, FIRC=1 MHz		0.9	1.2	mA
			V _{DD} =3 V, FIRC=1 MHz		0.5	0.7	
			V _{DD} =5 V, FIRC=2 MHz		1.1	1.4	
			V _{DD} =3 V, FIRC=2 MHz		0.6	0.8	
			V _{DD} =5 V, FIRC=4 MHz		1.5	2.0	
			V _{DD} =3 V, FIRC=4 MHz		0.8	1.0	
			V _{DD} =5 V, FIRC=8 MHz		2.2	2.9	
			V _{DD} =3 V, FIRC=8 MHz		1.2	1.6	
		SLOW mode, SIRC No loads, All I/O pins with internal pull-up.	V _{DD} =5 V, SXT=167 KHz		55	70	μA
			V _{DD} =3 V, SXT=136 KHz		20	26	
			V _{DD} =5 V, SIRC=86 KHz	–	40	52	
			V _{DD} =3 V, SIRC=68 KHz	–	15	20	
			V _{DD} =5 V, SIRC=43 KHz	–	35	45	
			V _{DD} =3 V, SIRC=34 KHz	–	10	13	
	V _{DD} =5 V, SIRC=21 KHz			30	40		
	V _{DD} =3 V, SIRC=17 KHz			9	12		
	STOP Mode LVR enable	V _{DD} =5 V			5	10	
		V _{DD} =3 V			1.5	3	
	STOP Mode LVR disable	V _{DD} =5 V				0.1	
		V _{DD} =3 V				0.1	
Pull-Up Resistor	R _P	V _{IN} =0 V Port A	V _{DD} =5 V V _{DD} =3 V	–	60 120	–	KΩ
		V _{IN} =0 V PA7	V _{DD} =5 V V _{DD} =3 V	–	60 125	–	KΩ

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

Parameter	Condition	Min	Typ	Max	Unit
FIRC Frequency (*)	25°C, $V_{DD}=3 \sim 5.5 \text{ V}$ ($\pm 3\%$)	-3%	8.00	+3%	MHz
	25°C, $V_{DD}=2.5 \sim 3 \text{ V}$ ($\pm 5\%$)	-5%	8.00	+5%	
	-40°C ~ 85°C, $V_{DD}=2.5 \sim 5.5 \text{ V}$	-7%	8.00	+7%	

(*) FIRC frequency can be selected to 1 MHz, 2 MHz, 4 MHz, and 8 MHz.

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

Parameter	Conditions	Min	Typ	Max	Unit
RESET Input Low width	Input $V_{DD}=5 \text{ V} \pm 10 \%$	3	–	–	μs
WDT time	$V_{DD}=5 \text{ V}$, $\text{WDTPSC}=00$	70	100	130	ms
	$V_{DD}=3 \text{ V}$, $\text{WDTPSC}=00$	85	123	160	ms
WKT time	$V_{DD}=5 \text{ V}$, $\text{WKTPSC}=00$	0.5	0.8	1.1	ms
	$V_{DD}=3 \text{ V}$, $\text{WKTPSC}=00$	0.7	1.0	1.3	ms
CPU start up time	$V_{DD}=5 \text{ V}$	–	14	–	ms

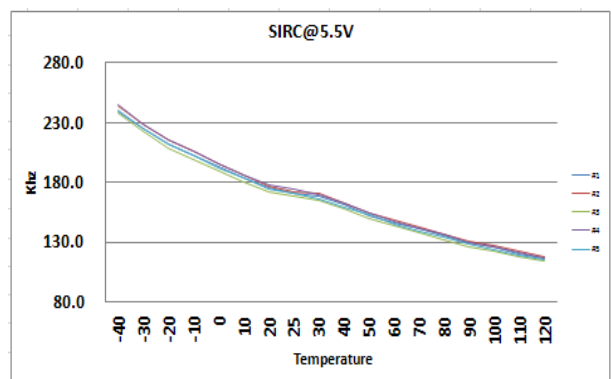
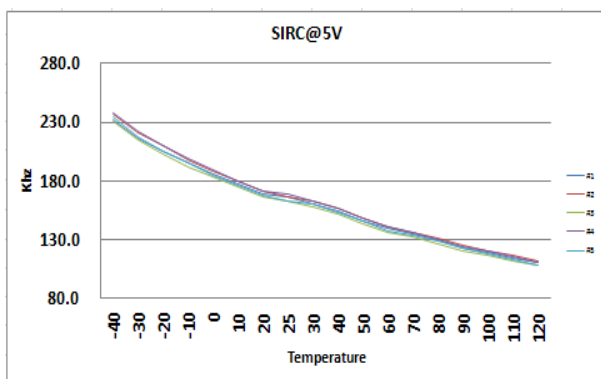
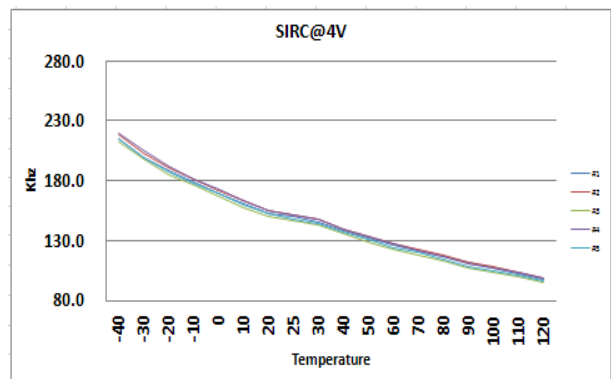
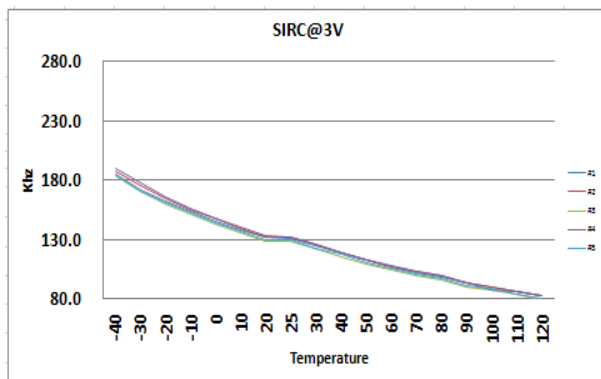
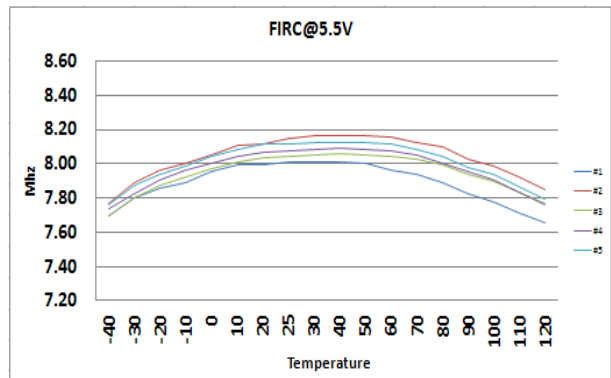
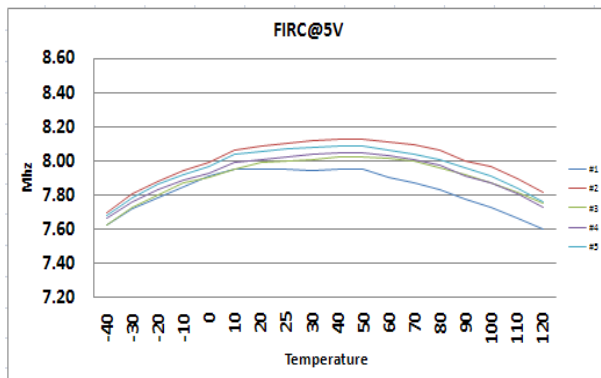
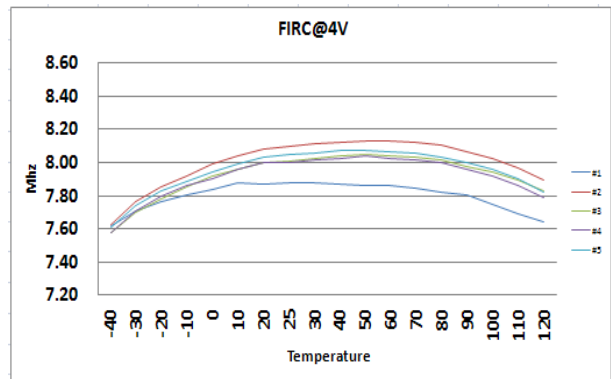
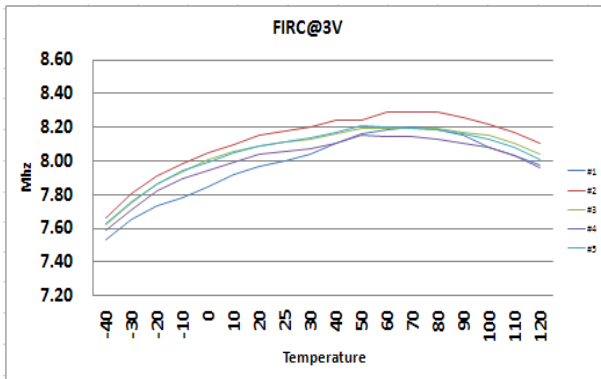
5. LVR Circuit and VBG (Bandgap Reference Voltage) Characteristics ($T_A = 25^\circ\text{C}$)

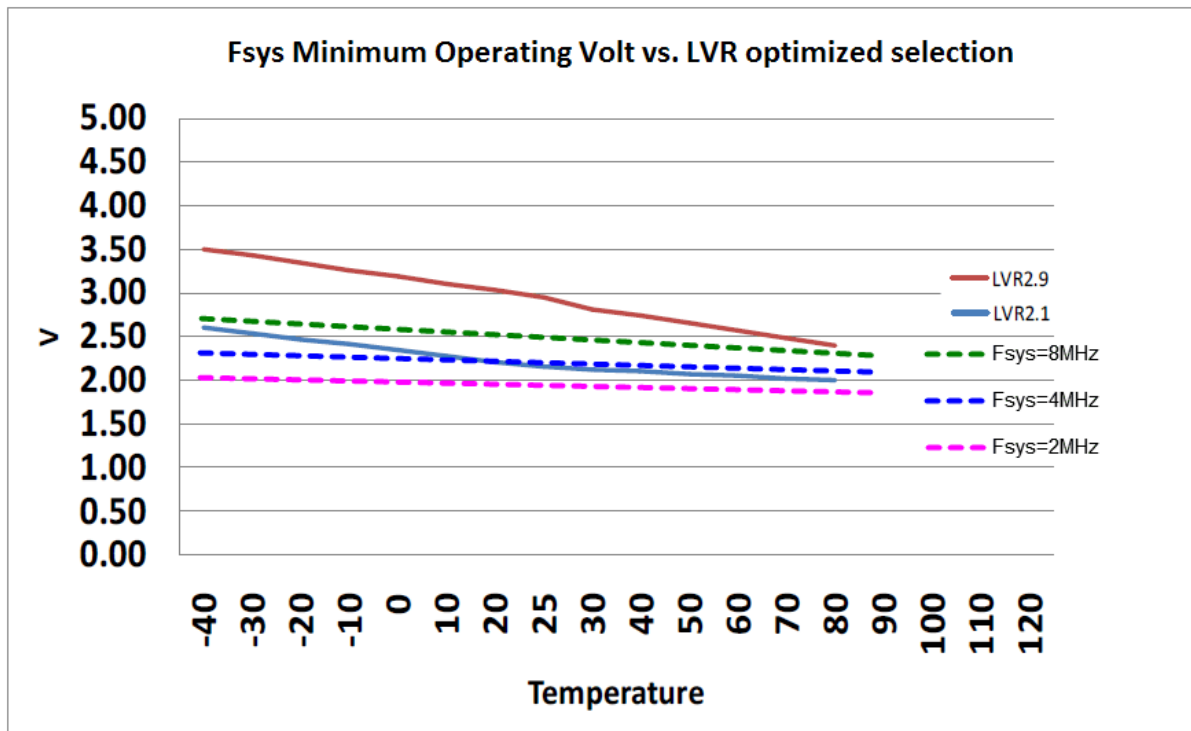
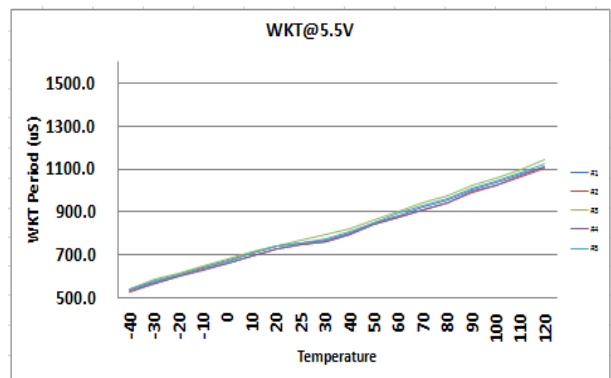
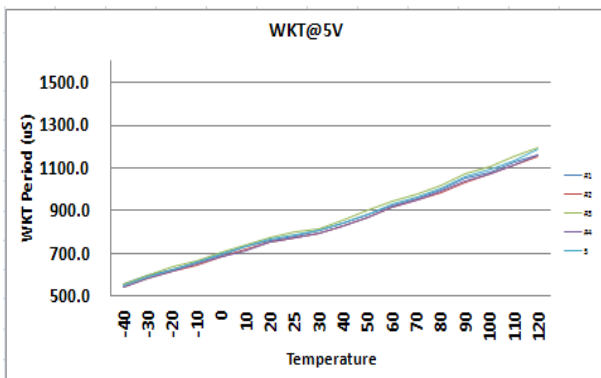
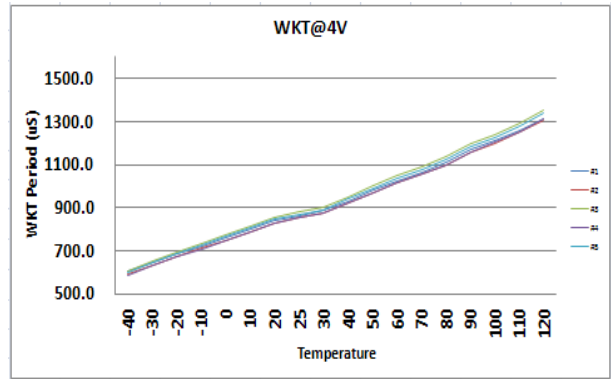
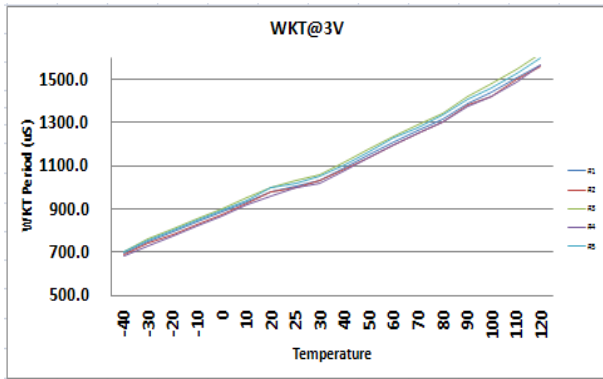
Parameter	Symbol	Min	Typ	Max	Unit
LVR Reference Voltage	V_{LVR}	–	2.15	–	V
		–	2.95	–	
LVR Hysteresis Voltage	V_{HYST}	–	± 0.1	–	V
Low Voltage Detection time	t_{LVR}	100	–	–	μs

6. ADC Electrical Characteristics ($T_A = 25^\circ\text{C}$, $V_{DD}=2.0\text{V}$ to 5.5V , $V_{SS}=0\text{V}$)

Parameter	Conditions	Min	Typ	Max	Units
Total Accuracy	$V_{DD}=5.12\text{V}$, $V_{SS}=0\text{V}$	–	± 2.5	± 5	LSB
Integral Non-Linearity		–	± 3.2	± 6	
Max Input Clock (f_{ADC})	–	–	–	2	MHz
Conversion Time (for AD0~AD4)	$f_{ADC(\text{max})} = 1 \text{ MHz}$	–	50	–	μs
Input Voltage	–	V_{SS}	–	V_{DD}	V

7. Characteristic Graphs

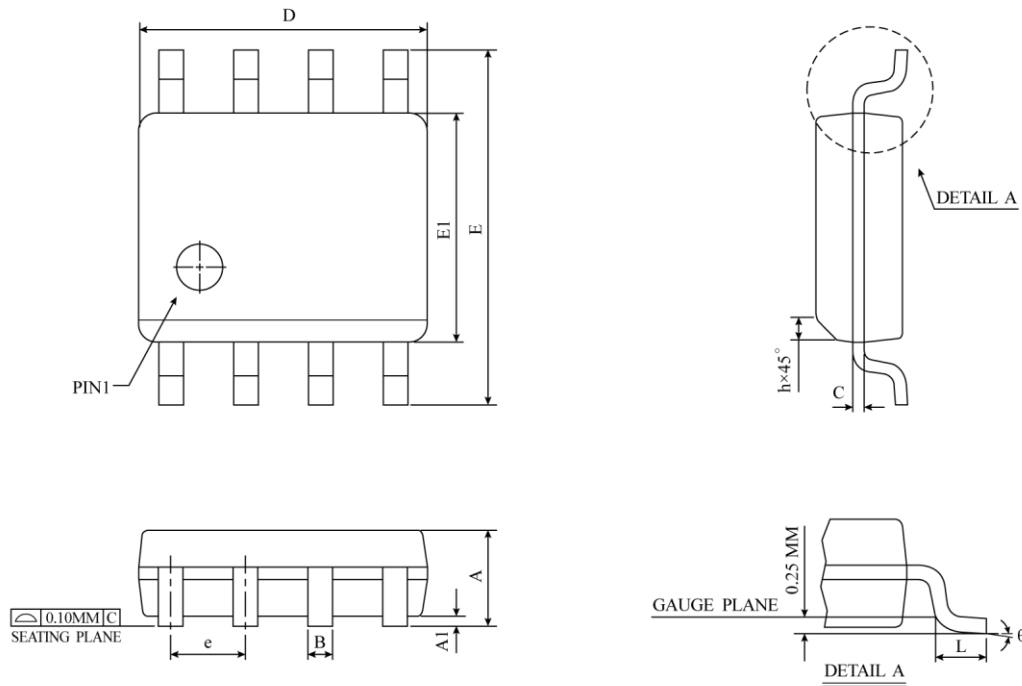




PACKAGING INFORMATION

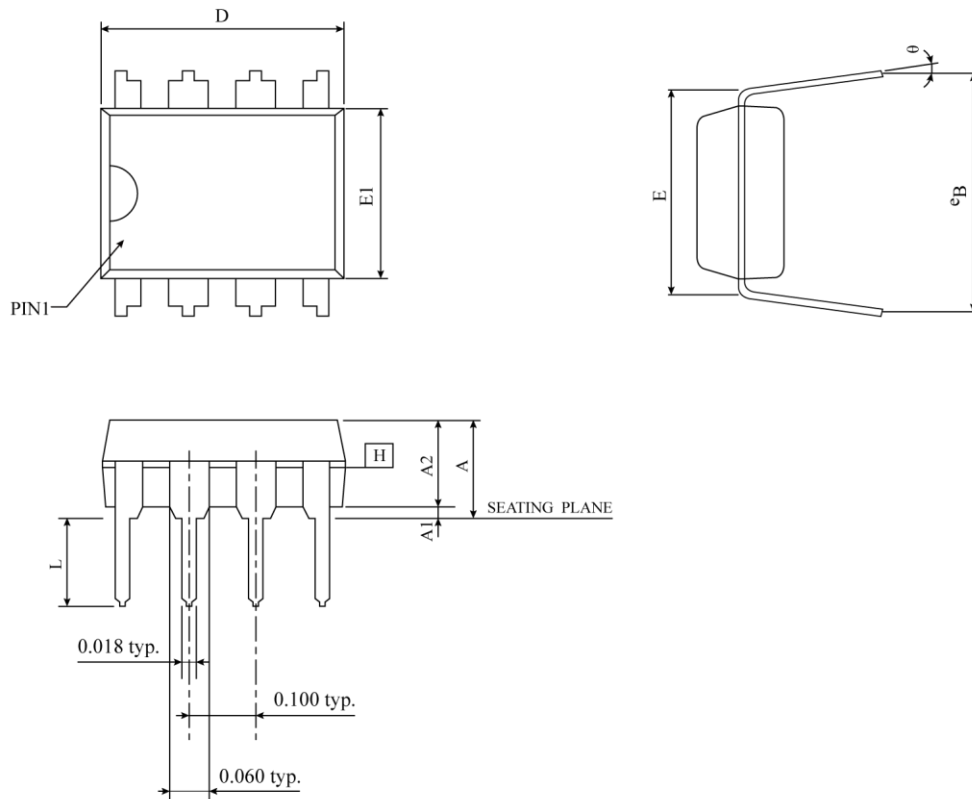
The ordering information:

Ordering number	Package
TM57PA11-OTP	Wafer/Dice blank chip
TM57PA11-COD	Wafer/Dice with code
TM57PA11-OTP-14	SOP 8-pin (150 mil)
TM57PA11-OTP-01	DIP 8-pin (300 mil)
TM57PA11-OTP-43	TSSOP 8-pin (173mil)

SOP-8 (150mil) Package Dimension


SYMBOL	DIMENSION IN MM			DIMENSION IN INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	1.35	1.55	1.75	0.0532	0.0610	0.0688
A1	0.10	0.18	0.25	0.0040	0.0069	0.0098
B	0.33	0.42	0.51	0.0130	0.0165	0.0200
C	0.19	0.22	0.25	0.0075	0.0087	0.0098
D	4.80	4.90	5.00	0.1890	0.1939	0.1988
E	5.80	6.00	6.20	0.2284	0.2362	0.2440
E1	3.80	3.90	4.00	0.1497	0.1536	0.1574
e	1.27 BSC			0.050 BSC		
h	0.25	0.38	0.50	0.0099	0.0148	0.0196
L	0.40	0.84	1.27	0.0160	0.0330	0.0500
θ	0°	4°	8°	0°	4°	8°
JEDEC	MS-012 (AA)					

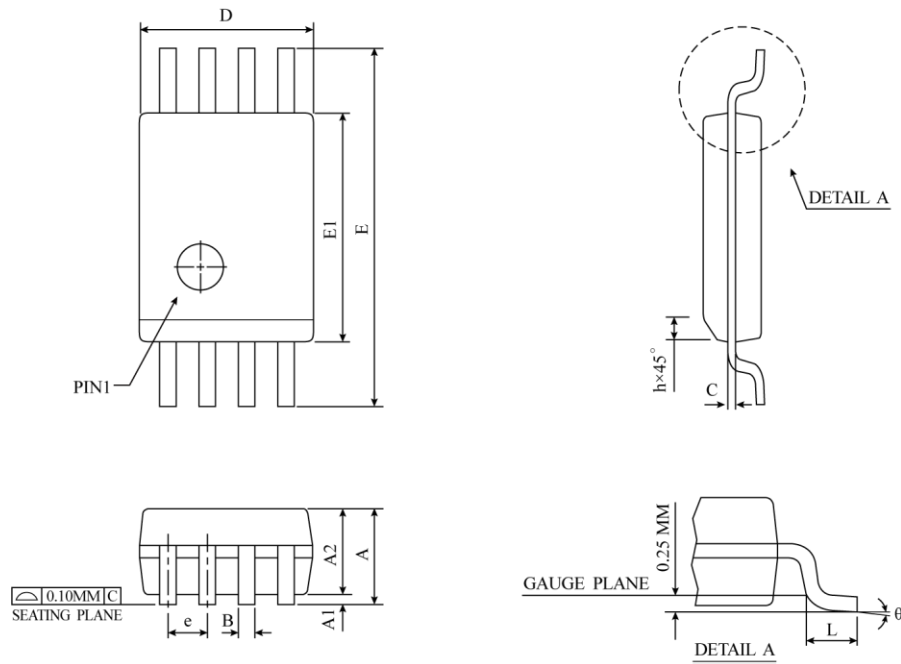
△ * NOTES : DIMENSION " D " DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS.
MOLD FLASH, PROTRUSIONS AND GATE BURRS SHALL
NOT EXCEED 0.15 MM (0.006 INCH) PER SIDE.

DIP-8 (300mil) Package Dimension


SYMBOL	DIMENSION IN MM			DIMENSION IN INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	-	-	5.334	-	-	0.210
A1	0.381	-	-	0.015	-	-
A2	3.175	3.302	3.429	0.125	0.130	0.135
D	9.017	9.586	10.160	0.355	0.378	0.400
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 (BA)					

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.

TSSOP-8 (173mil) Package Dimension


SYMBOL	DIMENSION IN MM			DIMENSION IN INCH		
	MIN	NOM	MAX	MIN	NOM	MAX
A	-	-	1.20	-	-	0.047
A1	0.05	0.10	0.15	0.002	0.004	0.006
A2	0.80	0.93	1.05	0.031	0.036	0.041
B	0.22 TYP			0.009 TYP		
D	2.90	3.00	3.10	0.114	0.118	0.122
E	6.40 BSC			0.252 BSC		
E1	4.30	4.40	4.50	0.169	0.173	0.177
e	0.65 TYP			0.026 TYP		
L	0.45	0.60	0.75	0.018	0.024	0.030
θ	0°	4°	8°	0°	4°	8°
JEDEC	MS-153 (AA)					

⚠ *NOTES : DIMENSION " D " DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS.
MOLD FLASH, PROTRUSIONS OR GATE BURRS SHALL NOT EXCEED 0.15 PER SIDE.
DIMENSION " E1 " DOES NOT INCLUDE INTERLEAD FLASH OR PROTRUSION.
INTERLEAD FLASH OR PROTRUSIONS SHALL NOT EXCEED 0.25 PER SIDE.
DIMENSION " B " DOES NOT INCLUDE DAMBAR PROTRUSION.
ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08 MM TOTAL IN EXCESS OF
THE " B " DIMENSION AT MAXIMUM LOWER RADIUS OF THE LOWER RADIUS OF THE FOOT.
MINIMUM SPACE BETWEEN PROTRUSION AND ADJACENT LEAD IS 0.07 MM.