# EPROM -Based 8-Bit Microcontroller

## Devices Included in this Data Sheet:

- FM8PA76AE : 20-pin EPROM device
- FM8PA76BE : 14-pin EPROM device
- FM8PA76DE: 16-pin EPROM device
- FM8PA76EE : 24-pin EPROM device with VR pin
- FM8PA76FE : 16-pin EPROM device with VR pin

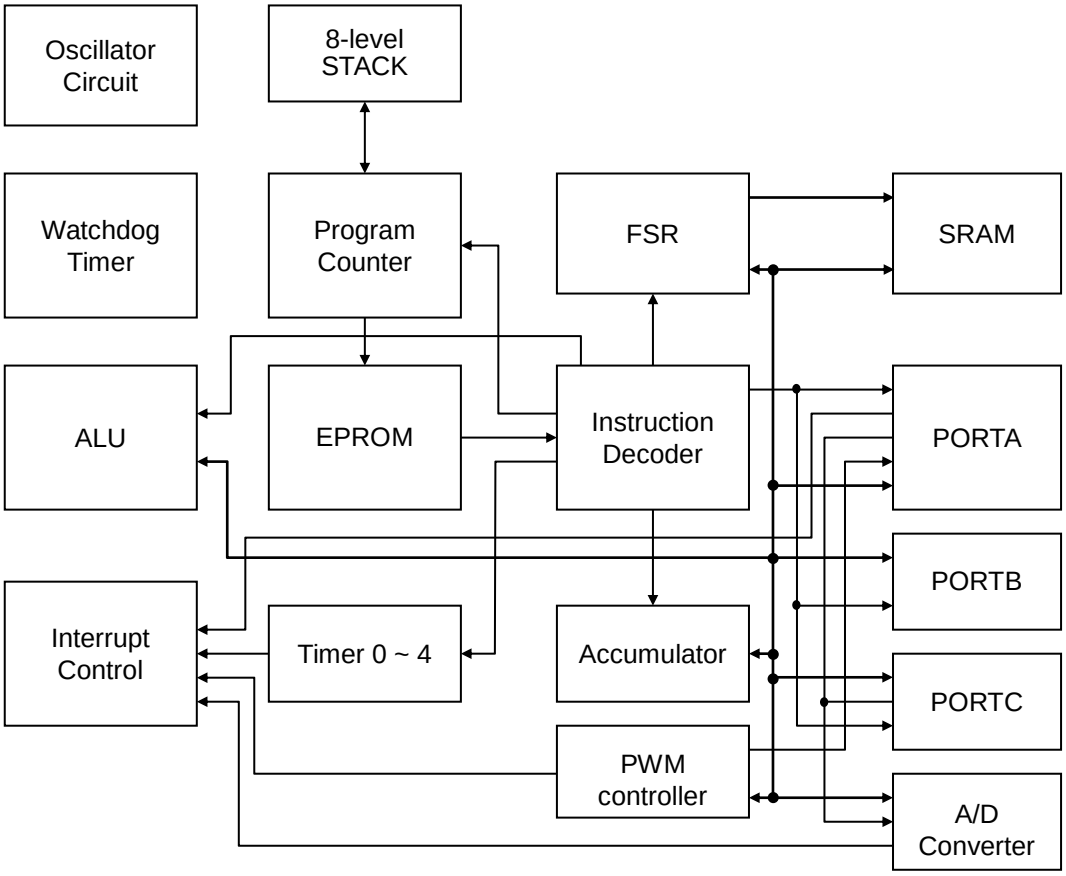
## FEATURES

- Total 9 channel 12bit AD converter with ±2LSB resolution
- All instructions are single cycle except for program branches which are two-cycles
- 16-bit wide instructions
- Configurable CPU clock per instruction cycle: Focs/4 and Fosc/2
- All EPROM area LGOTO instruction
- All EPROM area subroutine LCALL instruction
- 8-bit wide data path
- 8-level deep hardware stack
- 2K x 16 bits on chip EPROM
- 45x8 bits on chip special purpose registers and 128 x 8 bits on chip general purpose registers (SRAM)
- Operating speed: DC-20 MHz clock input, or DC-100 ns instruction cycle
- Direct, indirect addressing modes for data accessing
- Five real time up count Timer/counter with 3-bit programmable prescaler
  - TMR0: 16-bit Timer(up-counter)
  - TMR1: 8-bit, PWM1(Period) & Timer
  - TMR2: 8-bit, PWM1(Duty) & Timer
  - TMR3: 8-bit, PWM2(Period) & Timer
  - TMR4: 8-bit, PWM2(Duty) & Timer
- Built-in 3 levels Low Voltage Detector (LVDT) (2.2V/2.6V/3.7V) for Brown-out Reset (BOR)
- Power-up Reset Timer (PWRT)
- On chip Watchdog Timer (WDT) with internal oscillator for reliable operation and soft-ware watch-dog enable/disable control
- Three I/O ports IOA, IOB and IOC with independent direction control
  - 17 Bi-direction I/O port (Programmable Pull-up enable in Input mode)
  - One Input only port (IOB2/RSTB)
- Four kinds of interrupt source: 5 Timers/Counters, 8 external interrupt sources: IOA0~IOA7, Internal watchdog timer (i_WDT) wakeup, and A/D end of conversion
- Wake-up from SLEEP:
  - Port A (IOA0~IOA7) pin change wakeup
  - WDT overflow
  - i_WDT overflow
- Power saving SLEEP mode
- Programmable Code Protection
- Selectable oscillator options:
  - ERC: External Resistor/ Voltage Controlled Oscillator
  - XT: Crystal/Resonator Oscillator
  - HF: High Frequency Crystal/Resonator Oscillator
  - LF: Low Frequency Crystal Oscillator
  - IRC: Internal Resistor/Capacitor Oscillator
- Wide-operating voltage range:
  - EPROM : 2.2V to 5.5V

## GENERAL DESCRIPTION

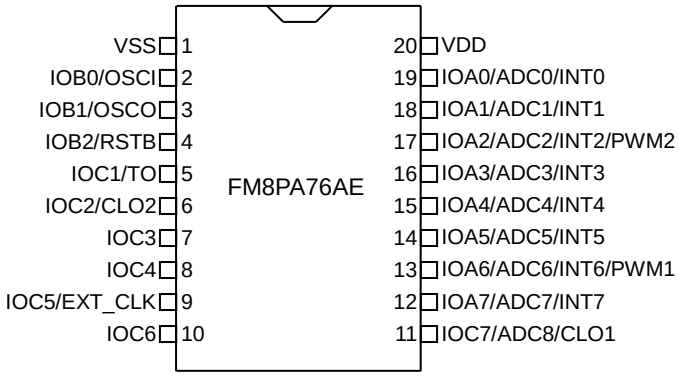
The FM8PA76 is a family of low-cost, high speed, high noise immunity, EPROM-based 8-bit CMOS microcontrollers. It employs a RISC architecture with 54 instructions. All instructions are single cycle except for program branches which take two cycles. The easy to use and easy to remember instruction set reduces development time significantly.
The FM8PA76 consists of Power-on Reset (POR), Brown-out Reset (BOR), Power-up Reset Timer (PWRT), Watchdog Timer, EPROM, SRAM, tri-state I/O port, I/O pull-high control, Power saving SLEEP mode, 5 real time programmable clock/counter, Interrupt, Wake-up from SLEEP mode, and Code Protection for EPROM products. There are eight oscillator configurations to be chosen from, including the power-saving LP (Low Power) oscillator and cost saving internal RC oscillator.
The FM8PA76 address 2K×16 of program memory.
The FM8PA76 can directly or indirectly address its register files and data memory. All special function registers including the program counter are mapped in the data memory.
The FM8PA76 provides total 9 channel 12bit AD converter with ±2LSB resolution.

## BLOCK DIAGRAM

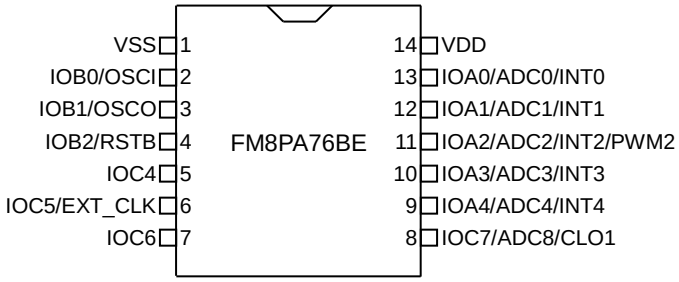


## PIN CONNECTION

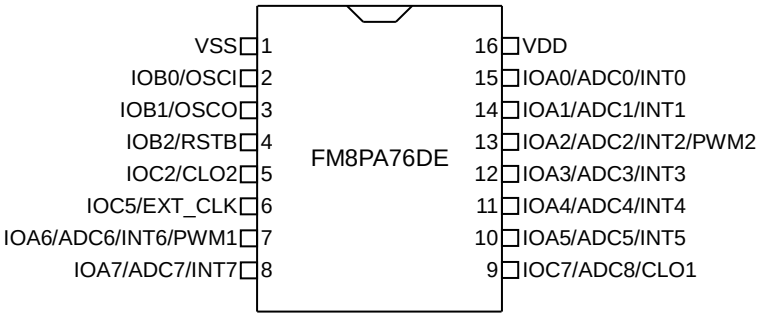
### PDIP20, SOP20



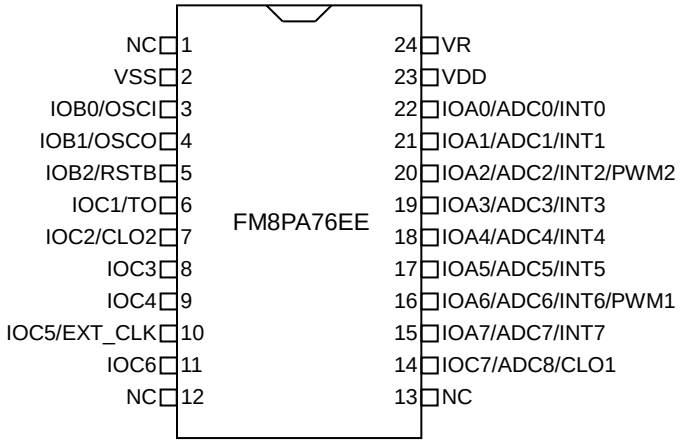

### PDIP14, SOP14



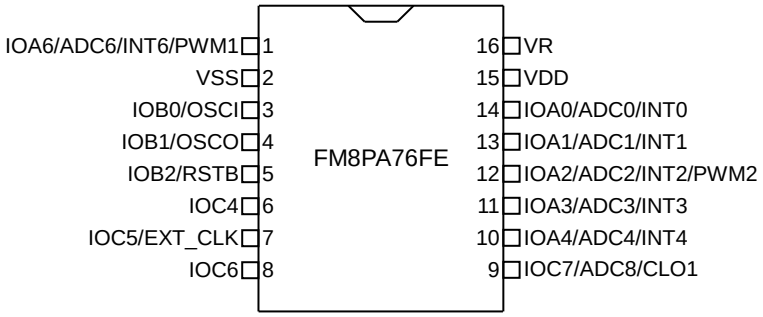

### PDIP16, SOP16



**PDIP24, SOP24 (With VR PIN)**




**PDIP16, SOP16 (With VR PIN)**



## PIN DESCRIPTIONS

| Name | I/O | Description |
|---|---|---|
| IOA0/AD0/INT0 ~ IOA7/AD7/INT7 | I/O | 1. Bi-direction I/O port (programmable Pull-high in Input mode)<br>2. Wake-up on pin change<br>3. External interrupt input<br>4. A/D converter input<br>5. IOA2 is PWM2 output<br>6. IOA6 is PWM1 output |
| IOB0/OSCI | I/O | 1. Bi-direction I/O port (programmable Pull-high in Input mode)<br>2. Oscillator input (HF, XT, LF, ERC mode) |
| IOB1/OSCO | I/O | 1. Bi-direction I/O port (programmable Pull-high in Input mode)<br>2. Oscillator output (HF, XT, LF, ERC mode) |
| IOB2/RSTB | I | 1. Input port, **voltage on this pin must not exceed VDD.**<br>2. System clear (RESET) input. This pin is an active low RESET to the device |
| IOC1/TO~ IOC2/CLO2 | I/O | 1. IOC1~IOC2 is Bi-direction I/O port (programmable Pull-high in Input mode)<br>2. TO (PWM2 interrupt/2) shared with IOC1<br>3. Clock output 2 with prescaler shared with IOC2 |
| IOC3 | I/O | Bi-direction I/O port (programmable Pull-high in Input mode) |
| IOC4 | I/O | Bi-direction I/O port (programmable Pull-high in Input mode) |
| IOC5/EXT_CLK ~ IOC6 | I/O | 1. IOC5~IOC6 Bi-direction I/O port (programmable Pull-high in Input mode)<br>2. EXT_CLK (External clock input) shared with IOC5 |
| IOC7/AD8/CLO1 | I/O | 1. Bi-direction I/O port (programmable Pull-high in Input mode)<br>2. A/D converter input<br>3. CLO1 (system clock out) shared with IOC7 |
| VR | - | ADC module reference input, **voltage on this pin must not exceed VDD.** |
| VDD | - | Positive supply |
| VSS | - | Ground |

Legend: I=input, O=output, I/O=input/output

Note: Please refer to 2.2 for detail IO type description

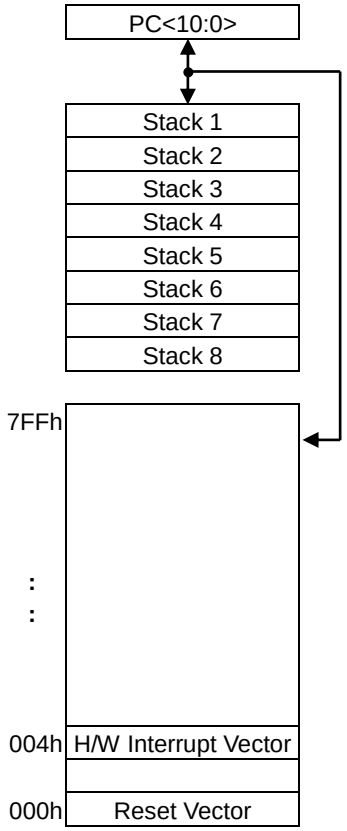
## 1.0 MEMORY ORGANIZATION

FM8PA76 memory is organized into program memory and data memory.

### 1.1 Program Memory Organization

The FM8PA76 have an 11-bit Program Counter capable of addressing a 2K×16 program memory space.
The RESET vector for the FM8PA76 is at 000h.
The H/W interrupt vector is at 004h.
User can use “LCALL(far call)/LGOTO(far goto)” instructions to program user's code within entire program area.

**FIGURE 1.1: Program Memory Map and STACK**




FM8PA76

## 1.2 Data Memory Organization

Data memory is composed of 45 bytes Special Function Registers and 128 bytes General Purpose Registers. The data memory can be accessed either directly or indirectly through the FSR register.

**TABLE 1.1: Registers File Map for FM8PA76**

| Address | Description |
|---|---|
| 00h<br>:<br>:<br>3EH | Special Purpose Register |
| 40H<br><br><br>:<br>:<br><br><br>BFH | General Purpose Register |

**TABLE 1.2: Special Purpose Registers Map**

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| **System** | | | | | | | | | |
| 00h (r/w) | INDF | Uses contents of FSR to address data memory (not a physical register) | | | | | | | |
| 01h (r/w) | PCL | Low order 8 bits of PC | | | | | | | |
| 02h (r/w) | PCH | - | - | - | - | - | High order 3 bits of PC | | |
| 03h (r/w) | STATUS | - | - | - | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | Z | DC | C |
| 04h (r/w) | FSR | Indirect data memory address pointer | | | | | | | |
| **IO PAD & CONTROL** | | | | | | | | | |
| 05h (r/w) | IOSTA | IOSTA7 | IOSTA6 | IOSTA5 | IOSTA4 | IOSTA3 | IOSTA2 | IOSTA1 | IOSTA0 |
| 06h (r/w) | PORTA | IOA7 | IOA6 | IOA5 | IOA4 | IOA3 | IOA2 | IOA1 | IOA0 |
| 07h (r/w) | IOSTB | - | - | - | - | - | - | IOSTB1 | IOSTB0 |
| 08h (r/w) | PORTB | - | - | - | - | - | IOB2 | IOB1 | IOB0 |
| 09h (r/w) | IOSTC | IOSTC7 | IOSTC6 | IOSTC5 | IOSTC4 | IOSTC3 | IOSTC2 | IOSTC1 | - |
| 0Ah (r/w) | PORTC | IOC7 | IOC6 | IOC5 | IOC4 | IOC3 | IOC2 | IOC1 | - |
| **Timer0:16-bit timer** | | | | | | | | | |
| 10h (r/w) | **TMR0_CTL** | T0EN | T0LOAD | T0SO1 | T0SO0 | T0EDGE | T0PS2 | T0PS1 | T0PS0 |
| 11h (r/w) | **TMR0L_LA** | 16-bit real-time timer/counter latch Low | | | | | | | |
| 12h (r/w) | **TMR0H_LA** | 16-bit real-time timer/counter latch High | | | | | | | |
| 13h (r) | **TMR0L_CNT** | 16-bit real-time timer/counter count Low | | | | | | | |
| 14h (r) | **TMR0H_CNT** | 16-bit real-time timer/counter count High | | | | | | | |
| **Timer1: 8-bit, PWM1 (Period) & Timer** | | | | | | | | | |
| 15h (r/w) | **TMR1_CTL1** | T1EN | T1LOAD | T1SO1 | T1SO0 | T1EDGE | T1PS2 | T1PS1 | T1PS0 |
| 16h (r/w) | **TMR1_CTL2** | T12MOD | **PWM1_INI** | - | - | PWM1R3 | PWM1R2 | PWM1R1 | PWM1R0 |
| 17h (r/w) | TMR1_LA | 8-bit real-time timer/counter Latch | | | | | | | |
| 18h (r) | TMR2_CNT | 8-bit real-time timer/counter Count | | | | | | | |
| **Timer2: 8-bit, PWM1 (Duty) & Timer** | | | | | | | | | |
| 19h (r/w) | **TMR2_CTL1** | T2EN | T2LOAD | T2SO1 | T2SO0 | T2EDGE | T2PS2 | T2PS1 | T2PS0 |
| 1Ah (r/w) | TMR2_LA | 8-bit real-time timer/counter Latch | | | | | | | |
| 1Bh (r) | TMR2_CNT | 8-bit real-time timer/counter Count | | | | | | | |

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| **Timer3: 8-bit, PWM2 (Period) & Timer** | | | | | | | | | |
| 1Ch (r/w) | **TMR3_CTL1** | T3EN | T3LOAD | T3SO1 | T3SO0 | T3EDGE | T3PS2 | T3PS1 | T3PS0 |
| 1Dh (r/w) | **TMR3_CTL2** | T34MOD | **PWM2_INI** | - | - | PWM2R3 | PWM2R2 | PWM2R1 | PWM2R0 |
| 1Eh (r/w) | TMR3_LA | 8-bit real-time timer/counter Latch | | | | | | | |
| 1Fh (r) | TMR3_CNT | 8-bit real-time timer/counter Count | | | | | | | |
| **Timer4: 8-bit, PWM2 (Duty) & Timer** | | | | | | | | | |
| 20h (r/w) | **TMR4_CTL1** | T4EN | T4LOAD | T4SO1 | T4SO0 | T4EDGE | T4PS2 | T4PS1 | T4PS0 |
| 21h (r/w) | TMR4_LA | 8-bit real-time timer/counter Latch | | | | | | | |
| 22h (r/w) | TMR4_CNT | 8-bit real-time timer/counter Count | | | | | | | |
| **IRQ** | | | | | | | | | |
| 25h (r/w) | INTEN | GIE | ADCIE | PAIE | T4IE | **T3_PWM2IE** | T2IE | **T1_PWM1IE** | T0IE |
| 26h (r/w) | INTFLAG | - | ADCIF | PAIF | T4IF | **T3_PWM2IF** | T2IF | **T1_PWM1IF** | T0IF |
| **ADC Control** | | | | | | | | | |
| 29h (r/w) | AD_CTL1 | ADCEN | - | MODE | - | CHSL3 | CHSL2 | CHSL1 | CHSL0 |
| 2Ah (r/w) | AD_CTL2 | CMP_D | - | - | - | - | CLKSL2 | CLKSL1 | CLKSL0 |
| 2Bh (r/w) | AD_CTL3 | - | - | - | - | ANISL3 | ANISL2 | ANISL1 | ANISL0 |
| 2Ch (r/w) | AD_DATL | D3 | D2 | D1 | D0 | - | - | - | - |
| 2Dh (r/w) | AD_DATH | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 |
| **Others** | | | | | | | | | |
| 2Fh (r/w) | SYS_CLK | CLKS | - | - | - | - | - | IRCPD | ECLKPD |
| 30h(r/w) | CLO_CTL | CLO2SO | **CLO2PS1** | **CLO2PS0** | - | EXT_CLK | CLO2_E | CLO1_E | TO_E |
| 31h(r/w) | APHCON | PHA7 | PHA6 | PHA5 | PHA4 | PHA3 | PHA2 | PHA1 | PHA0 |
| 32h(r/w) | BPHCON | - | - | - | - | - | - | PHB1 | PHB0 |
| 33h(r/w) | CPHCON | PHC7 | PHC6 | PHC5 | PHC4 | PHC3 | PHC2 | PHC1 | - |
| 3Ah(r/w) | INT_PA | PA7IEN | PA6IEN | PA5IEN | PA4IEN | PA3IEN | PA2IEN | PA1IEN | PA0IEN |
| 3Dh (r/w) | WDT_CTL | WDTEN | I_WDT | I_TWDT | - | - | WDTPS2 | WDTPS1 | WDTPS0 |
| 3Eh (r/w) | TB_BNK | - | - | - | - | - | BNK2 | BNK1 | BNK0 |

Legend: - = unimplemented, read as '0'.

## 2.0 FUNCTIONAL DESCRIPTIONS

### 2.1 Operational Registers

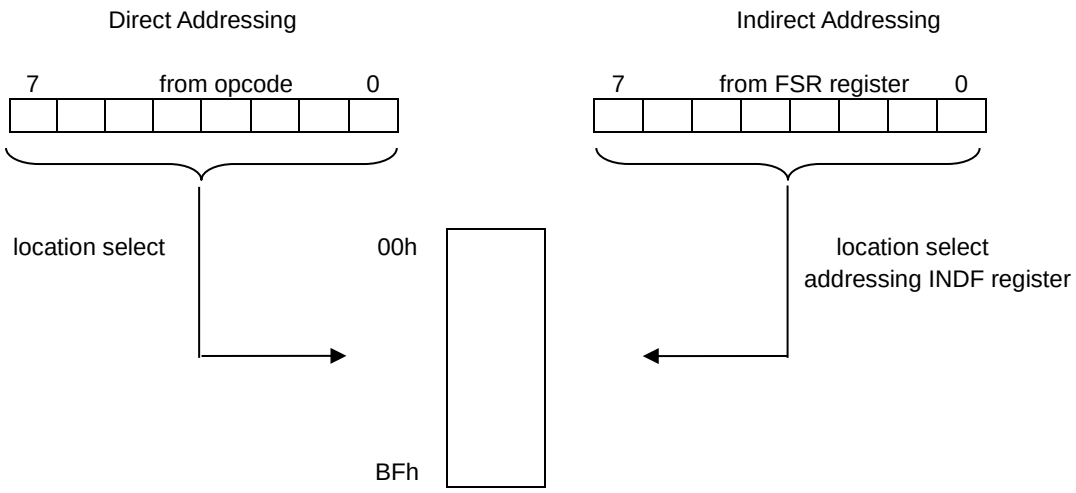
#### 2.1.1 INDF (Indirect Addressing Register)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 00h (r/w) | INDF | Uses contents of FSR to address data memory (not a physical register) | | | | | | | |

The INDF Register is not a physical register. Any instruction accessing the INDF register can actually access the register pointed by FSR Register. Reading the INDF register itself indirectly (FSR="0") will read 00h. Writing to the INDF register indirectly results in a no-operation (although status bits may be affected).

**EXAMPLE 2.1: INDIRECT ADDRESSING**
Register file 48 contains the value 10h
Register file 49 contains the value 0Ah
Load the value 48 into the FSR Register
A read of the INDF Register will return the value of 10h
Increment the value of the FSR Register by one (@FSR=49h)
A read of the INDF register now will return the value of 0Ah.

**FIGURE 2.1: Direct/Indirect Addressing for FM8PA76**



### 2.1.2 PCL/PCH (Low/High Bytes of Program Counter) & Stack

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 01h (r/w) | PCL | Low order 8 bits of PC | | | | | | | |
| 02h (r/w) | PCH | - | - | - | - | - | High order 3 bits of PC | | |

FM8PA76 devices have an 11-bits wide Program Counter (PC) and eight-level deep 11-bit hardware push/pop stack. This 11-bits Program Counter can be accessed and controlled by two registers, PCH and PCL. The low byte of PC control register is called the PCL. This register is readable and writable. The high byte of PC control register is called the PCH. This register contains the PC<10:8> bits also readable or writable.
The PCL and PCH registers normally indicate the value of Program Counter. But when interrupt occurrence and execution of RETF and RETFIE, the PCH data would not be update.
Any address within the program memory can be written into PCL and PCH registers. If the PCH register been changed and different from Program Counter., the value of PCH register updated only when execute LGOTO, LCALL, RETURN, or PCL value changed or increases from 0xff to 0x00. Once the value of PCL register changed, the program and Program Counter will jump to the location indicated by PCL and PCH register.

**FIGURE 2.2: Loading of PC in Different Situations**

Situation 1: **PCH** and **PCL** increase from FFH to 00H

| Program Counter | PCH | PCL |
|---|---|---|
| 1FEH | 01h | FEh |
| 1FFH | 01h | FFh |
| 200H | 02h | 00h |

```
Instruction     Operands
#include <8PA76.ash>
…
MOVIA           55h
NOP
NOP
…
```

Situation 2: **PCH/PCL** direct load address

| Program Counter | PCH | PCL |
|---|---|---|
| 1ACH | 01H | ACH |
| 1ADH | 01H | ADH |
| 1AEH | 02H | AEH |
| 1AFH | 02H | AFH |
| | | |
| 281H | 02H | 81H |
| 282H | 02h | 82h |

```
Instruction     Operands
#include <8PA76.ash>
…
MOVIA           02h
MOVAR           PCH
MOVIA           81h
MOVAR           PCL     PC={(PCH):(PCL)}
                        PC={(02H):(81H)}
                        PC=281H
NOP
,,,
…
ORG             0281h
MOVIA           55h
NOP
…
```

Situation 3: **PCH/PCL** mathematic operation

| Program Counter | PCH | PCL |
|---|---|---|
| 1ACH | 01H | ACH |
| 1ADH | 01H | ADH |
| 1AEH | 02H | AEH |
| 1AFH | 02H | AFH |
| | | |
| 331H | 03H | 31H |
| 332H | 03h | 32h |

```
                    Instruction    Operands
                    #include <8PA76.ash>
                    …
                    MOVIA          02h
                    MOVAR          PCH
                    MOVIA          81h
                    ADDAR          PCL
                                                  PC={(PCH+Carry):(PCL+1+81H)}
                                                  PC={(02H+1):(AFH+1+81H)}
                                                  PC=331H
                    NOP
                    ,,,
                    …
                    ORG            331h
                    MOVIA          55h
                    NOP
                    …
```

### 2.1.3 STATUS (Status Register)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 03h (r/w) | STATUS | - | - | - | $\overline{TO}$ | $\overline{PD}$ | Z | DC | C |

This register contains the arithmetic status of the ALU, and the RESET status. And the reset status bits $\overline{TO}$ and $\overline{PD}$, are not writable.

**C** : Carry/borrow bit.
ADDAR, ADDIA
= 1, Carry occurred.
= 0, No Carry occurred.
SUBAR, SUBIA
= 1, No borrow occurred.
= 0, Borrow occurred.
Note : A subtraction is executed by adding the two's complement of the second operand. For rotate (RRR, RLR) instructions, this bit is loaded with either the high or low order bit of the source register.

**DC** : Half carry/half borrow bit
ADDAR, ADDIA
= 1, Carry from the 4th low order bit of the result occurred.
= 0, No Carry from the 4th low order bit of the result occurred.
SUBAR, SUBIA
= 1, No Borrow from the 4th low order bit of the result occurred.
= 0, Borrow from the 4th low order bit of the result occurred.

**Z** : Zero bit.
= 1, The result of a logic operation is zero.
= 0, The result of a logic operation is not zero.

$\overline{PD}$ : Power down flag bit.
= 1, after power-up or by the CLRWDT instruction.
= 0, by the SLEEP instruction.

$\overline{TO}$ : Time overflow flag bit.
= 1, after power-up or by the CLRWDT or SLEEP instruction
= 0, a watch-dog time overflow occurred

| $\overline{TO}$ | $\overline{PD}$ | Description |
|---|---|---|
| 0 | 0 | WDT timer overflow from sleep mode |
| 0 | 1 | WDT timer overflow from normal mode |
| 1 | 0 | Set ‘low” at RESETB from sleep mode |
| 1 | 1 | Power on reset |
| Unchanged | Unchanged | Set “low” at RESETB from normal mode |

### 2.1.4 FSR (Indirect Data Memory Address Pointer)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 04h (r/w) | FSR | Indirect data memory address pointer | | | | | | | |

**Bit7:Bit0** : Select registers address in the indirect addressing mode. See 2.1.1 for detail description.

### 2.1.5 PORTA, PORTB & PORTC (Port Data Registers)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 05h (r/w) | IOSTA | IOSTA7 | IOSTA6 | IOSTA5 | IOSTA4 | IOSTA3 | IOSTA2 | IOSTA1 | IOSTA0 |
| 06h (r/w) | PORTA | IOA7 | IOA6 | IOA5 | IOA4 | IOA3 | IOA2 | IOA1 | IOA0 |
| 07h (r/w) | IOSTB | - | - | - | - | - | - | IOSTB1 | IOSTB0 |
| 08h (r/w) | PORTB | - | - | - | - | - | IOB2* | IOB1 | IOB0 |
| 09h (r/w) | IOSTC | IOSTC7 | IOSTC6 | IOSTC5 | IOSTC4 | IOSTC3 | IOSTC2 | IOSTC1 | - |
| 0Ah (r/w) | PORTC | IOC7 | IOC6 | IOC5 | IOC4 | IOC3 | IOC2 | IOC1 | - |

The registers (IOSTA, IOSTB, and IOSTC) are used to define the input or output of each port.
**1** : Input.
**0** : Output.
Reading the port (PORTA, PORTB, and PORTC register) reads the status of the pins independent of the pin’s input/output modes. Writing to these ports will write to the port data latch. Please refer to 2.2 for detail I/O Port description.
Note: IOB2 is read only.

### 2.1.6 TMR0 (16 bits Time Clock/Counter register)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 10h (r/w) | **TMR0_CTL** | T0EN | T0LOAD | T0SO1 | T0SO0 | T0EDGE | T0PS2 | T0PS1 | T0PS0 |
| 11h (r/w) | **TMR0L_LA** | 16-bit real-time timer/counter latch Low | | | | | | | |
| 12h (r/w) | **TMR0H_LA** | 16-bit real-time timer/counter latch High | | | | | | | |
| 13h (r) | **TMR0L_CNT** | 16-bit real-time timer/counter count Low | | | | | | | |
| 14h (r) | **TMR0H_CNT** | 16-bit real-time timer/counter count High | | | | | | | |

The Timer0 is a 16-bit up count timer/counter which includes high byte (TMR0H_CNT), low byte (TMR0L_CNT) counter register, high byte (TMR0H_LA), and low byte (TMR0L_LA) latch register. Please refer to 2.3 for detail Timer description.

**T0EN** : TMR0 Enable/Disable
= 0, TMR0 Disable.
= 1, TMR0 Enable.

**T0LOAD** : Enable/Disable Latch Buffer automatically load to counter register while writing to latch register
= 1, Enable TMR0 latch buffer automatically load to counter register while writing to latch register.
= 0, Disable TMR0 latch buffer automatically load to counter register while writing to latch register.
Note: This bit is only affected after latch register written. When the timer underflows, the latch register data will automatically load into counter register.

**T0SO1:T0SO0** : TMR0 clock source selection

| Bit5<br>T0SO1 | Bit4<br>T0SO0 | TMR0 clock source |
|---|---|---|
| 0 | 0 | EXT_CLK (IOC5) |
| 0 | 1 | Crystal mode OSCI or EXT_RC(In dual RC clock mode) |
| 1 | 0 | Internal 4MHz RC |
| 1 | 1 | Don't use |

**T0EDGE** : TMR0 clock edge selection. This bit works only when external clock source EXT_CLK (IOC5) selected.
= 0, TMR0 increased while external clock L→H.
= 1, TMR0 increased while external clock H→L.

**T0PS2:T0PS0** : TMR0 Prescaler selection

| Bit2<br>T0PS2 | Bit1<br>T0PS1 | Bit0<br>T0PS0 | TMR0 Prescaler rate |
|---|---|---|---|
| 0 | 0 | 0 | 1:1 |
| 0 | 0 | 1 | 1:2 |
| 0 | 1 | 0 | 1:4 |
| 0 | 1 | 1 | 1:8 |
| 1 | 0 | 0 | 1:16 |
| 1 | 0 | 1 | 1:32 |
| 1 | 1 | 0 | 1:64 |
| 1 | 1 | 1 | 1:128 |

### 2.1.7 Timer1: 8-bit, PWM1 (Period) & Timer

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 15h (r/w) | **TMR1_CTL1** | T1EN | T1LOAD | T1SO1 | T1SO0 | T1EDGE | T1PS2 | T1PS1 | T1PS0 |
| 16h (r/w) | **TMR1_CTL2** | T12MOD | **PWM1_INI** | - | - | PWM1R3 | PWM1R2 | PWM1R1 | PWM1R0 |
| 17h (r/w) | TMR1_LA | 8-bit real-time timer/counter Latch | | | | | | | |
| 18h (r) | TMR1_CNT | 8-bit real-time timer/counter Count | | | | | | | |

The Timer1 is an 8-bit up count timer/counter which includes counter register (TMR1_CNT), and latch register (TMR1_LA). Please refer to 2.3 for detail Timer description.
The Timer1 can also be combined with Timer2 as PWM1 period and duty and controlled by the register TMR1_CTL2. Please refer to 2.4 for detail PWM description.

#### 2.1.7.1 Timer1 control (TMR1_CTL1)

**T1EN** : TMR1 (PWM1) Enable/Disable
= 0, TMR1 (PWM1) Disable.
= 1, TMR1 (PWM1) Enable.

**T1LOAD** : Enable/Disable Latch Buffer automatically load to counter register while writing to latch register
= 1, Enable TMR1 latch buffer automatically load to counter register while writing to latch register.
= 0, Disable TMR1 latch buffer automatically load to counter register while writing to latch register.
Note: This bit is only affected after latch register written. When the timer underflows, the latch register data will automatically load into counter register.

**T1SO1:T1SO0** : TMR1 clock source selection

| Bit5 | Bit4 | TMR1 clock source |
|---|---|---|
| T1SO1 | T1SO0 | |
| 0 | 0 | EXT_CLK (IOC5) |
| 0 | 1 | Crystal mode OSCI or EXT_RC(In dual RC clock mode) |
| 1 | 0 | Internal 4MHz RC |
| 1 | 1 | Don't use |

**T1EDGE** : TMR1 clock edge selection. This bit works only when external clock source EXT_CLK (IOC5) selected.
= 0, TMR1 increased while external clock L→H.
= 1, TMR1 increased while external clock H→L.

**T1PS2:T1PS0** : TMR1 Prescaler selection

| Bit2 | Bit1 | Bit0 | TMR1 Prescaler rate |
|---|---|---|---|
| T1PS2 | T1PS1 | T1PS0 | |
| 0 | 0 | 0 | 1:1 |
| 0 | 0 | 1 | 1:2 |
| 0 | 1 | 0 | 1:4 |
| 0 | 1 | 1 | 1:8 |
| 1 | 0 | 0 | 1:16 |
| 1 | 0 | 1 | 1:32 |
| 1 | 1 | 0 | 1:64 |
| 1 | 1 | 1 | 1:128 |

#### 2.1.7.2 PWM1 control (TMR1_CTL2)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 16h (r/w) | **TMR1_CTL2** | T12MOD | **PWM1_INI** | - | - | PWM1R3 | PWM1R2 | PWM1R1 | PWM1R0 |

**T12MOD** : TMR1 and TMR2 working mode (TMR / PWM1)
= 0, TMR1 and TMR2 is Timer.
= 1, TMR1 and TMR2 is PWM1.

**PWM1_INI** : Initial State of PWM1 output duty.
= 0, Set the initial state to H, change to L when TMR2 duty overflow.
= 1, Set the initial state to L, change to H when TMR2 duty overflow.

**PWM1R3:PWM1R0** : Interrupt Event Rate of PWM1.
"1:N" means interrupt occurred after "N" PWM1 pulses.

| Bit3 | Bit2 | Bit1 | Bit0 | PWM1 Interrupt rate |
|---|---|---|---|---|
| PWM1R3 | PWM1R2 | PWM1R1 | PWM1R0 | |
| 0 | 0 | 0 | 0 | 1:1 |
| 0 | 0 | 0 | 1 | 1:2 |
| 0 | 0 | 1 | 0 | 1:3 |
| . . | . . | . . | . . | . . |
| 1 | 1 | 1 | 0 | 1:15 |
| 1 | 1 | 1 | 1 | 1:16 |

### 2.1.8 TMR2: 8-bit, PWM1 (Duty) & Timer

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 19h (r/w) | **TMR2_CTL1** | T2EN | T2LOAD | T2SO1 | T2SO0 | T2EDGE | T2PS2 | T2PS1 | T2PS0 |
| 1Ah (r/w) | TMR2_LA | 8-bit real-time timer/counter Latch | | | | | | | |
| 1Bh (r) | TMR2_CNT | 8-bit real-time timer/counter Count | | | | | | | |

The Timer2 is an 8-bit up count timer/counter which includes counter register (TMR2_CNT), and latch register (TMR2_LA). Please refer to 2.3 for detail Timer description.

**T2EN** : TMR2 Enable/Disable
= 0, TMR2 (PWM1) Disable.
= 1, TMR2 (PWM1) Enable.

**T2LOAD** : Enable/Disable Latch Buffer automatically load to counter register while writing to latch register
= 1, Enable TMR2 latch buffer automatically load to counter register while writing to latch register.
= 0, Disable TMR2 latch buffer automatically load to counter register while writing to latch register.
Note: This bit is only affected after latch register written. When the timer underflows, the latch register data will automatically load into counter register.

**T2SO1:T2SO0** : TMR2 clock source selection

| Bit5 T2SO1 | Bit4 T2SO0 | TMR2 clock source |
|---|---|---|
| 0 | 0 | EXT_CLK (IOC5) |
| 0 | 1 | Crystal mode OSCI or EXT_RC(In dual RC clock mode) |
| 1 | 0 | Internal 4MHz RC |
| 1 | 1 | Don't use |

**T2EDGE** : TMR2 clock edge selection. This bit works only when external clock source EXT_CLK (IOC5) selected.
= 0, TMR2 increased while external clock L→H.
= 1, TMR2 increased while external clock H→L.

**T2PS2:T2PS0** : TMR2 Prescaler selection

| Bit2 T2PS2 | Bit1 T2PS1 | Bit0 T2PS0 | TMR2 Prescaler rate |
|---|---|---|---|
| 0 | 0 | 0 | 1:1 |
| 0 | 0 | 1 | 1:2 |
| 0 | 1 | 0 | 1:4 |
| 0 | 1 | 1 | 1:8 |
| 1 | 0 | 0 | 1:16 |
| 1 | 0 | 1 | 1:32 |
| 1 | 1 | 0 | 1:64 |
| 1 | 1 | 1 | 1:128 |

### 2.1.9 Timer3: 8-bit, PWM2 (Period) & Timer

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 1Ch (r/w) | **TMR3_CTL1** | T3EN | T3LOAD | T3SO1 | T3SO0 | T3EDGE | T3PS2 | T3PS1 | T3PS0 |
| 1Dh (r/w) | **TMR3_CTL2** | T34MOD | **PWM2_INI** | - | - | PWM2R3 | PWM2R2 | PWM2R1 | PWM2R0 |
| 1Eh (r/w) | TMR3_LA | 8-bit real-time timer/counter Latch | | | | | | | |
| 1Fh (r) | TMR3_CNT | 8-bit real-time timer/counter Count | | | | | | | |

The Timer3 is an 8-bit up count timer/counter which includes counter register (TMR3_CNT), and latch register (TMR3_LA). Please refer to 2.3 for detail Timer description.
The Timer3 can also be combined with Timer4 as PWM2 period and duty and controlled by the register TMR2_CTL2. Please refer to 2.4 for detail PWM description.

#### 2.1.9.1 Timer3 control (TMR3_CTL1)

**T3EN** : TMR3 (PWM2) Enable/Disable
= 0, TMR3 (PWM2) Disable.
= 1, TMR3 (PWM2) Enable.

**T3LOAD** : Enable/Disable Latch Buffer automatically load to counter register while writing to latch register
= 1, Enable TMR3 latch buffer automatically load to counter register while writing to latch register.
= 0, Disable TMR3 latch buffer automatically load to counter register while writing to latch register.
Note: This bit is only affected after latch register written. When the timer underflows, the latch register data will automatically load into counter register.

**T3SO1:T3SO0** : TMR3 clock source selection

| Bit5 T3SO1 | Bit4 T3SO0 | TMR3 clock source |
|---|---|---|
| 0 | 0 | EXT_CLK (IOC5) |
| 0 | 1 | Crystal mode OSCI or EXT_RC(In dual RC clock mode) |
| 1 | 0 | Internal 4MHz RC |
| 1 | 1 | Don't use |

**T3EDGE** : TMR3 clock edge selection. This bit works only when external clock source EXT_CLK (IOC5) selected.
= 0, TMR3 increased while external clock L→H.
= 1, TMR3 increased while external clock H→L.

**T3PS2:T3PS0** : TMR3 Prescaler selection

| Bit2 T3PS2 | Bit1 T3PS1 | Bit0 T3PS0 | TMR3 Prescaler rate |
|---|---|---|---|
| 0 | 0 | 0 | 1:1 |
| 0 | 0 | 1 | 1:2 |
| 0 | 1 | 0 | 1:4 |
| 0 | 1 | 1 | 1:8 |
| 1 | 0 | 0 | 1:16 |
| 1 | 0 | 1 | 1:32 |
| 1 | 1 | 0 | 1:64 |
| 1 | 1 | 1 | 1:128 |

### 2.1.9.2 PWM2 control (TMR3_CTL2)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 1Dh (r/w) | **TMR3_CTL2** | T34MOD | **PWM2_INI** | - | - | PWM2R3 | PWM2R2 | PWM2R1 | PWM2R0 |

**T34MOD** : TMR3 and TMR4 working mode (TMR/ PWM2)
= 0, TMR3 and TMR4 is Timer.
= 1, TMR3 and TMR4 is PWM2.

**PWM2_INI** : Initial State of PWM2 output duty
= 0, Set the initial state to H, change to L when TMR4 duty overflow.
= 1, Set the initial state to L, change to H when TMR4 duty overflow.

**PWM2R3:PWM2R0** : Interrupt Event Rate of PWM2.
"1:N" means interrupt occurred after "N" PWM2 pulses.

| Bit3 | Bit2 | Bit1 | Bit0 | PWM2 Interrupt rate |
|---|---|---|---|---|
| PWM2R3 | PWM2R2 | PWM2R1 | PWM2R0 | |
| 0 | 0 | 0 | 0 | 1:1 |
| 0 | 0 | 0 | 1 | 1:2 |
| 0 | 0 | 1 | 0 | 1:3 |
| . . | . . | . . | . . | . . |
| 1 | 1 | 1 | 0 | 1:15 |
| 1 | 1 | 1 | 1 | 1:16 |

### 2.1.10 TMR4: 8-bit, PWM2 (Duty) & Timer

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 20h (r/w) | **TMR4_CTL1** | T4EN | T4LOAD | T4SO1 | T4SO0 | T4EDGE | T4PS2 | T4PS1 | T4PS0 |
| 21h (r/w) | TMR4_LA | 8-bit real-time timer/counter Latch | | | | | | | |
| 22h (r/w) | TMR4_CNT | 8-bit real-time timer/counter Count | | | | | | | |

The Timer4 is an 8-bit up count timer/counter which includes counter register (TMR4_CNT), and latch register (TMR4_LA). Please refer to 2.3 for detail Timer description.

**T4EN** : TMR4 Enable/Disable
= 0, TMR4 Disable.
= 1, TMR4 Enable.

**T4LOAD** : Enable/Disable Latch Buffer automatically load to counter register while writing to latch register
= 1, Enable TMR4 latch buffer automatically load to counter register while writing to latch register.
= 0, Disable TMR4 latch buffer automatically load to counter register while writing to latch register.
Note: This bit is only affected after latch register written. When the timer underflows, the latch register data will automatically load into counter register.

**T4SO1: T4SO0** : TMR4 clock source selection

| Bit5 T4SO1 | Bit4 T4SO0 | TMR4 clock source |
|---|---|---|
| 0 | 0 | EXT_CLK (IOC5) |
| 0 | 1 | Crystal mode OSCI or EXT_RC(In dual RC clock mode) |
| 1 | 0 | Internal 4MHz RC |
| 1 | 1 | Don't use |

**T4EDGE** : TMR4 clock edge selection. This bit works only when external clock source EXT_CLK (IOC5) selected.
= 0, TMR4 increased while external clock L→H.
= 1, TMR4 increased while external clock H→L.

**T4PS2:T4PS0** : TMR4 Prescaler selection

| Bit2 T4PS2 | Bit1 T4PS1 | Bit0 T4PS0 | TMR4 Prescaler rate |
|---|---|---|---|
| 0 | 0 | 0 | 1:1 |
| 0 | 0 | 1 | 1:2 |
| 0 | 1 | 0 | 1:4 |
| 0 | 1 | 1 | 1:8 |
| 1 | 0 | 0 | 1:16 |
| 1 | 0 | 1 | 1:32 |
| 1 | 1 | 0 | 1:64 |
| 1 | 1 | 1 | 1:128 |

### 2.1.11 INTEN (Interrupt Mask Register)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 25h (r/w) | INTEN | GIE | ADCIE | PAIE | T4IE | **T3_PWM2IE** | T2IE | **T1_PWM1IE** | T0IE |

**GIE** : Global interrupt enable bit.
= 0, Disable all interrupts.
= 1, Enable all un-masked interrupts.
Note : When an interrupt event occurred with the GIE bit and its corresponding interrupt enable bits are set, the GIE bit will be cleared by hardware to disable any further interrupts. The RETFIE instruction will exit the interrupt routine and set the GIE bit to re-enable interrupt.

**ADCIE** : ADC end of conversion (EOC) interrupt enable
= 0, Disable interrupt.
= 1, Enable interrupt.

**PAIE** : PORTA interrupt enable
= 0, Disable interrupt.
= 1, Enable interrupt.

**T4IE** : Timer4 overflow interrupt enable bit.
= 0, Disable interrupt.
= 1, Enable interrupt.

**T3_PWM2IE** : Timer3/PWM2 overflow interrupt enable bit.
= 0, Disable interrupt.
= 1, Enable interrupt.

**T2IE** : Timer2 overflow interrupt enable bit.
= 0, Disable interrupt.
= 1, Enable interrupt.

**T1_PWM1IE** : Timer1/PWM1 overflow interrupt enable bit.
= 0, Disable interrupt.
= 1, Enable interrupt.

**T0IE** : Timer0 overflow interrupt enable bit.
= 0, Disable interrupt.
= 1, Enable interrupt.

### 2.1.12 INTFLAG (Interrupt Status Register)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 26h (r/w) | INTFLAG | - | ADCIF | PAIF | T4IF | **T3_PWM2IF** | T2IF | **T1_PWM1IF** | T0IF |

**BIT7** : Not used. Read as “0”.

**ADCIF** : ADC end of conversion interrupt flag. Set after ADC end of conversion, and reset by software.

**PAIF** : PortA IOA0~7 Interrupt flag. Set when pin changed on selected IOA by register INT_PA (3Ah), and reset by software.

**T4IF** : TMR4 interrupt flag. Set when TMR4 overflows, and reset by software.

**T3_PWM2IF** : TMR3 interrupt or PWM2 interrupt flag. Set when TMR3 overflows or PWM2 pulse counts to selected interrupt rate, and reset by software.

**T2IF** : TMR2 interrupt flag. Set when TMR2 overflows, and reset by software.

**T1_PWM1IF** : TMR1 interrupt or PWM1 interrupt flag. Set when TMR1 overflows or PWM1 pulse counts to selected interrupt rate, and reset by software.

**T0IF** : TMR0 interrupt flag. Set when TMR0 overflows, and reset by software.

### 2.1.13 AD_CTL1 (AD converter Control Register1)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 29h (r/w) | AD_CTL1 | ADCEN | - | MODE | - | CHSL3 | CHSL2 | CHSL1 | CHSL0 |

**ADCEN** : ADC enable/disable setting
= 0, Disable.
= 1, Enable.
Note : This bit should be set by software and would be reset by hardware after the ADC end of conversion.

**MODE** : ADC operation mode selection
= 0, The ADC is operated in Analog to Digital Conversion mode.
= 1, The ADC is operated in Comparator mode.
Note : When the ADC in comparator mode, the converted data of input voltage would be compared to AD_DAT. The compared result would be stored in the bit7 of AD_CTL2 register.

**CHSL3:CHSL0** : ADC input channel select

| Bit3 | Bit2 | Bit1 | Bit0 | Input channel |
|---|---|---|---|---|
| CHSL3 | CHSL2 | CHSL1 | CHSL0 | |
| 0 | 0 | 0 | 0 | Channel 0, IOA0 pin |
| 0 | 0 | 0 | 1 | Channel 1, IOA1 pin |
| 0 | 0 | 1 | 0 | Channel 2, IOA2 pin |
| 0 | 0 | 1 | 1 | Channel 3, IOA3 pin |
| 0 | 1 | 0 | 0 | Channel 4, IOA4 pin |
| 0 | 1 | 0 | 1 | Channel 5, IOA5 pin |
| 0 | 1 | 1 | 0 | Channel 6, IOA6 pin |
| 0 | 1 | 1 | 1 | Channel 7, IOA7 pin |
| 1 | 0 | 0 | 0 | Channel 8, IOC7 pin |

### 2.1.14 AD_CTL2 (AD converter Control Register2)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 2Ah (r/w) | AD_CTL2 | CMP_D | - | - | - | - | CLKSL2 | CLKSL1 | CLKSL0 |

**CMP_D** : Comparison result of ADC in Comparator Mode
= 0, Input Voltage < AD_DAT.
= 1, Input Voltage ≧ AD_DAT.

**CLKSL2:CLKSL0** : Conversion Clock Source Selection

| Bit2 | Bit1 | Bit0 | Conversion clock |
|---|---|---|---|
| CKSL2 | CKSL1 | CKSL0 | |
| 0 | 0 | 0 | System clock /2 |
| 0 | 0 | 1 | System clock /8 |
| 0 | 1 | 0 | System clock /32 |
| 0 | 1 | 1 | System clock /128 |
| 1 | 0 | 0 | System clock /64 |
| 1 | 0 | 1 | System clock /16 |
| 1 | 1 | 0 | System clock /4 |
| 1 | 1 | 1 | Don't use |

Note : This clock is used to control the conversion precision and speed. The precision will be dropped off if faster conversion rate been used. The lowest conversion rate would be recommended in order to acquire most accurate data.

### 2.1.15 AD_CTL3 (AD converter Control Register3)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 2Bh (r/w) | AD_CTL3 | - | - | - | - | ANISL3 | ANISL2 | ANISL1 | ANISL0 |

**ANISL3:ANISL0** : Analog input selection

| Bit3 | Bit2 | Bit1 | Bit0 | Analog Input Selection |
|---|---|---|---|---|
| ANISL3 | ANISL2 | ANISL1 | ANISL0 | |
| 0 | 0 | 0 | 0 | All the ports are digital input |
| 0 | 0 | 0 | 1 | AN0 |
| 0 | 0 | 1 | 0 | AN1 |
| 0 | 0 | 1 | 1 | AN2 |
| 0 | 1 | 0 | 0 | AN3 |
| 0 | 1 | 0 | 1 | AN4 |
| 0 | 1 | 1 | 0 | AN5 |
| 0 | 1 | 1 | 1 | AN6 |
| 1 | 0 | 0 | 0 | AN7 |
| 1 | 0 | 0 | 1 | AN8 |

Note : To minimize power consumption, all the analog I/O pins should be setup properly before entering sleep mode.

### 2.1.16 AD_DAT (AD conversion data high and low)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 2Ch (r/w) | AD_DATL | D3 | D2 | D1 | D0 | - | - | - | - |
| 2Dh (r/w) | AD_DATH | D11 | D10 | D9 | D8 | D7 | D6 | D5 | D4 |

The AD_DAT registers contain the Analog to Digital converted data in the AD conversion mode. When operated in comparator mode, the data written to those registers would be used to compare to the converted data of input voltage.

### 2.1.17 SYS_CLK (System Clock Control)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 2Fh (r/w) | SYS_CLK | CLKS | - | - | - | - | - | IRCPD | ECLKPD |

The FM8PA76 could be operated either dual or single clock system selected by configuration words. Please refer to 2.12 for detail configuration selection description. This register is used to control the switch between different system clocks and power-down function of those clocks.

**CLKS** : System Clock Selection (only valid in dual clock mode)
= 0, System Clock is Internal 4MHz RC.
= 1, System Clock is External OSC/RC.

**IRCPD** : Internal 4MHz RC Power down Control (only valid in dual clock mode)
= 0, Internal 4MHz RC Power ON.
= 1, Internal 4MHz RC Power Down.
Note : Make sure the system clock been switch to external OSC/RC before power down internal RC.

**ECLKPD** : External clock (OSC/RC) Power down Control (only valid in dual clock mode)
= 0, External OSC/RC Power ON.
= 1, External OSC/RC Power Down.
Note : Make sure the system clock been switch to internal 4MHz RC before power down external OSC/RC.

### 2.1.18 CLO_CTL (Clock output Control)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 30h(r/w) | CLO_CTL | CLO2SO | **CLO2PS1** | **CLO2PS0** | - | EXT_CLK | CLO2_E | CLO1_E | TO_E |

The FM8PA76 provides three kinds of clock output. The first one (CLO1) is the system clock output. The 2nd one (CLO2) is the selected internal or external clock output with prescaler function. The 3rd one (TO) is the TMR3 output with the frequency of TMR3 divided by 2.

**EXAMPLE 2.2: When setting the following register**

```
CLO2SO = 0                      ; Internal 4MHz RC for CLO2
CLO2PS1:CLO2PS1 = <10>          ; CLO2 Prescaler 1:8
CLO2_E = 1                      ; IOC2 set to CLO2 output
IOSTC2 = 0                      ; Set IOC2 to output
```

There will be a clock output at IOC2 with frequency 500 KHz.

**CLO2SO** : System Clock output 2 source select
= 0, Clock Output 2 source is internal 4MHz RC (default).
= 1, Clock Output 2 source is external OSC/RC.

**CLO2PS1:CLO2PS0** : Clock Output 2 prescaler setting

| Bit6 CLO2PS1 | Bit5 CLO2PS0 | Clock Output 2 prescaler |
|---|---|---|
| 0 | 0 | 1:2 |
| 0 | 1 | 1:4 |
| 1 | 0 | 1:8 |
| 1 | 1 | 1:16 |

**EXT_CLK** : EXT_CLK (IOC5) function selection
= 0, IOC5 is normal I/O.
= 1, IOC5 is external clock input of timer.

**CLO2_E** : Clock Output 2 (IOC2) function selection
= 0, IOC2 is normal I/O.
= 1, IOC2 is Clock Output 2.

**CLO1_E** : Clock Output (IOC7) function selection
= 0, IOC7 is normal I/O.
= 1, IOC7 is System Clock Output.

**TO_E** : TMR3 output (IOC1) Enable/Disable
= 0, IOC1 is normal I/O.
= 1, IOC1 is the frequency of TMR3 (PWM2) divided by 2.

### 2.1.19 APHCON/ BPHCON/ CPHCON (Port* Pull-high Control)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 31h(r/w) | APHCON | PHA7 | PHA6 | PHA5 | PHA4 | PHA3 | PHA2 | PHA1 | PHA0 |
| 32h(r/w) | BPHCON | - | - | - | - | - | - | PHB1 | PHB0 |
| 33h(r/w) | CPHCON | PHC7 | PHC6 | PHC5 | PHC4 | PHC3 | PHC2 | PHC1 | - |

Those registers are used to setup pull-high resistor enable/disable of each IO pins.
= 0, Pull-high resistor disable.
= 1, Pull-high resistor enable.

### 2.1.20 INT_PA (IOA Interrupt/ wakeup control)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 3Ah(r/w) | INT_PA | PA7IEN | PA6IEN | PA5IEN | PA4IEN | PA3IEN | PA2IEN | PA1IEN | PA0IEN |

This register is used to enable/disable the interrupt/wakeup function of Port A. Please refer to 2.7.1 for detail description of External Interrupt and Wake up function.
= 0, selected IO interrupt/wakeup disable.
= 1, selected IO interrupt/wakeup enable.

### 2.1.21 WDT_CTL (Watchdog Timer Control)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 3Dh (r/w) | WDT_CTL | WDTEN | I_WDT | I_TWDT | - | - | WDTPS2 | WDTPS1 | WDTPS0 |

The FM8PA76 builds in a watchdog timer with two different modes, normal watchdog reset and internal watchdog wakeup. The watchdog timer is controlled by this register (WDT_CTL). Please refer to 2.5 for detail Watchdog Timer description.

**WDTEN** : Watchdog Timer Enable/ Disable.
= 0, WDT disable.
= 1, WDT Enable.

**I_WDT** : Internal Watchdog Wakeup mode selection.
= 0, Internal Watchdog Wakeup Disable.
= 1, Internal Watchdog Wakeup Enable.

**I_TWDT** : Watchdog Timer Stable time required when operating in I_WDT mode.
= 0, 5ms (default).
= 1, 1.25ms.

**WDTPS2:WDTPS0** : Watchdog timer prescaler setting

| Bit2 | Bit1 | Bit0 | WDT Prescaler rate |
|---|---|---|---|
| WDTPS2 | WDTPS1 | WDTPS0 | |
| 0 | 0 | 0 | 20mS |
| 0 | 0 | 1 | 40mS |
| 0 | 1 | 0 | 80mS |
| 0 | 1 | 1 | 160mS |
| 1 | 0 | 0 | 320mS |
| 1 | 0 | 1 | 640mS |
| 1 | 1 | 0 | 1.28S |
| 1 | 1 | 1 | 2.56S |

### 2.1.22 TAB_BNK (Table Look-up function)

| Address | Name | B7 | B6 | B5 | B4 | B3 | B2 | B1 | B0 |
|---|---|---|---|---|---|---|---|---|---|
| 3Eh (r/w) | TB_BNK | - | - | - | - | - | BNK2 | BNK1 | BNK0 |

The FM8PA76 provides a table look-up function and the bank selection of ROM data is controlled by this register. Please refer to 2.9 for detail operation of look-up table function.

**BNK2:BNK0** : Page selection of Look-up table

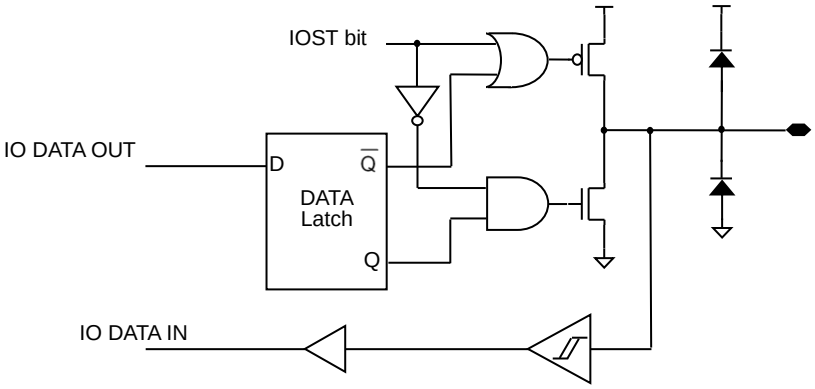
| Bit2 | Bit1 | Bit0 | BANK select |
|---|---|---|---|
| BNK2 | BNK1 | BNK0 | |
| 0 | 0 | 0 | 000 XXXX XXXX Table location |
| 0 | 0 | 1 | 001 XXXX XXXX Table location |
| 0 | 1 | 0 | 010 XXXX XXXX Table location |
| . | . | . | . |
| 1 | 1 | 1 | 111 XXXX XXXX Table location |

## 2.2 I/O Ports

There are totally 16 bi-directional tri-state I/O ports and one (IOB2) input only. All I/O pins (IOA<7:0>, IOB<1:0> and IOC<7:1>) have specified data direction control registers (IOSTA, IOSTB, IOSTC) which can configure these pins as output or input.
All the IO pins can also enable or disable a weak internal pull-high by setting APHCON, BPHCON, and CPHCON. This weak pull-high will be automatically turned off when the pin is configured as an output pin.
VR pin is reference voltage input pin of the ADC module, this pin does not have I/O function. The voltage on this pin must not exceed VDD, otherwise it will cause the pin burned down.

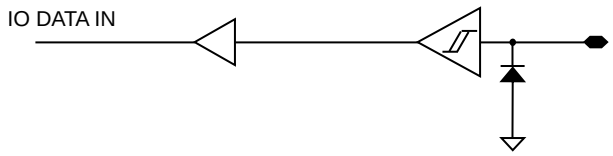
**FIGURE 2.3: Block Diagram of I/O Pins**

GENERIC IO BLOCK DIAGRAM:



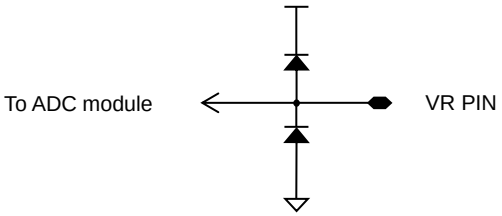

Pull-high/ADC/OSC control is not shown in this figure

IOB2/RSTB BLOCK DIAGRAM:




**Voltage on this pin must not exceed VDD.**

VR BLOCK DIAGRAM:

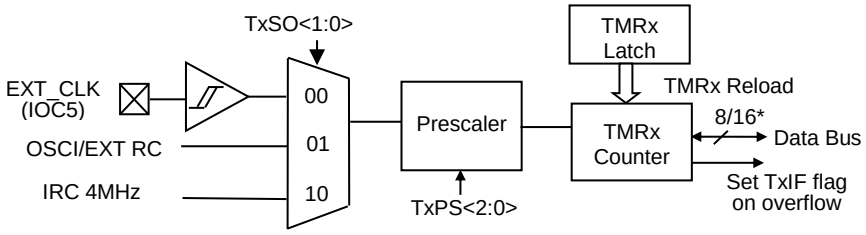



**Voltage on this pin must not exceed VDD.**

## 2.3 Timer/Event Counter (TMR0, TMR1, TMR2, TMR3, TMR4)

The FM8PA76 contains one 16-bit and four 8-bit up count Timers/counters. All these timers have auto reload function, TMR1/TMR2 and TMR3/TMR4 can be combined to perform PWM function

**FIGURE 2.4**: Block Diagram of the Timer




*TMR0 is 16 bits counter, the others is 8 bits counter.

### 2.3.1 Clock Source

There are 3 clock sources could be selected by each timer separately.

#### 2.3.1.1 EXT_CLK (IOC5)

The event counter mode would be activated when the source of EXT_CLK (IOC5) used. At this mode, the rising/ falling edge of the event could also be selected separately.

#### 2.3.1.2 External OSC or RC

Each timer can select external crystal or external RC as clock source when the external clock is available.

#### 2.3.1.3 Internal 4MHz RC

Each timer can select internal 4MHz RC as clock source.

### 2.3.2 Prescaler

Each timer contains a 3-bits prescaler which can scale the timer or counter from 1:1 to 1:128.

| Bit2 | Bit1 | Bit0 | Prescaler rate |
|---|---|---|---|
| 0 | 0 | 0 | 1:1 |
| 0 | 0 | 1 | 1:2 |
| 0 | 1 | 0 | 1:4 |
| 0 | 1 | 1 | 1:8 |
| 1 | 0 | 0 | 1:16 |
| 1 | 0 | 1 | 1:32 |
| 1 | 1 | 0 | 1:64 |
| 1 | 1 | 1 | 1:128 |

## 2.4 Pulse Width Modulation (PWM)

FM8PA76 provides two PWM output shared with TMR1/2 and TMR3/4. When PWM1 or PWM2 selected, TMR1/TMR3 becomes the period of PWM1/PWM2 and TMR2/TMR4 will be the duty of PWM1/PWM2.
If the system frequency is 4MHz, the range of PWM period could be from 0.5us to 8,192ms.
Please refer to the sample program and timing diagram below.

Note : When PWM duty or period needed to be changed, the auto-load control bit of the timer (TxLOAD) must be cleared before new data writes to latch register. If this bit still set, the data written to latch register would be load into counter register immediately and cause PWM output anomaly.

**EXAMPLE 2.3: PWM1 Setting**

```
//Set PWM1 Period
   MOVIA     01100001b      // Set Source: 4MHz IRC Prescaler 1:2
   MOVAR     TMR1_CTL1
   MOVIA     10000011b
   MOVAR     TMR1_CTL2      // Set PWM Interrupt Rate 1:4
   MOVIA     0xC1
   MOVAR     TMR1_LA        // Set Period (0xC1 >> 0xFF)
 //Set PWM1 Duty
   MOVIA     01010010b      // Set Source: Ext OSC(1) Prescaler 1:4
   MOVAR     TMR2_CTL1
   MOVIA     0xC8
   MOVAR     TMR2_LA        // Set Duty (0xC8 >> 0xFF)
   BSR       TMR1_CTL1,7    // Start PWM1
   MOVIA     0x82
   MOVAR     INTEN          // enable INTEN & PWM1 interrupt
   CLRR      INTFLAG        // clear interrupt flag
```

Note 1: This example demonstrates the PWM applied in dual clock system. The PWM duty (Timer2) must be smaller then PWM period (Timer1). In this example, the frequency of external OSC is approximately 16MHz.

**FIGURE 2.5: PWM Output Waveform**

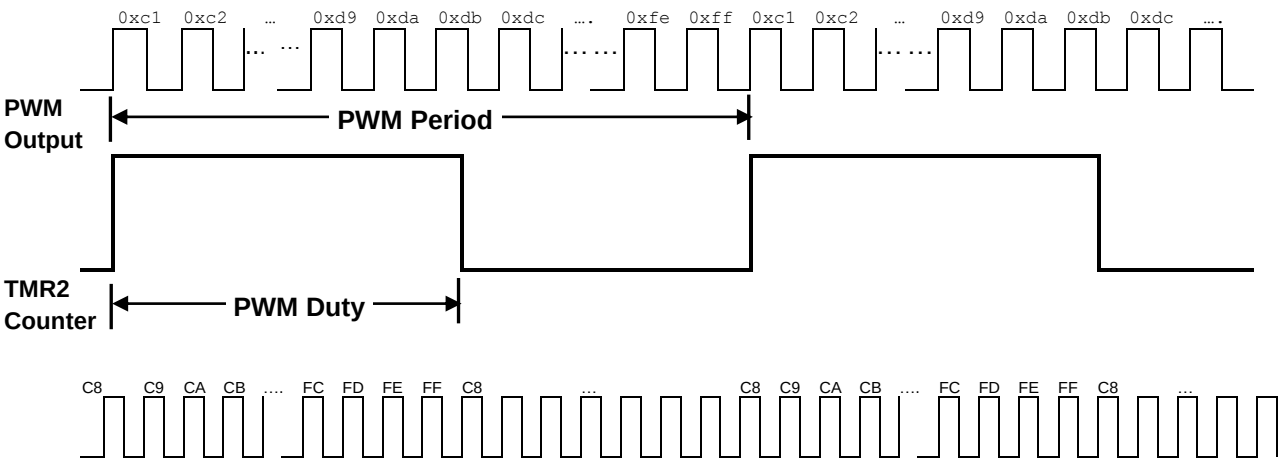
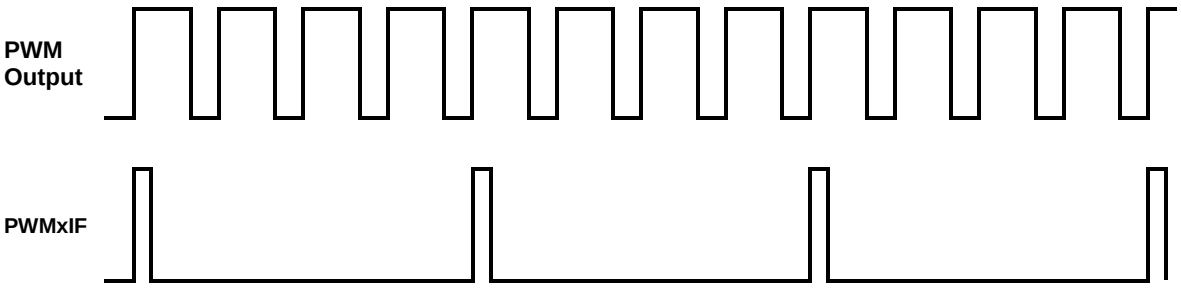


**FIGURE 2.6: PWM Interrupt Waveform**

**PWM Interrupt Rate PWMxR<3:0>= 0011 (1:4)**




## 2.5 Watch Dog Timer (WDT)

The Watchdog Timer (WDT) is a free running on-chip RC oscillator which does not require any external components. So the WDT will still run even if the clock on the OSCI and OSCO pins is turned off, such as in SLEEP mode. The WDT can be disabled by clearing the control bit WDTEN (WDT_CTL <7>) to “0”.

The WDT has a typical time-out period of 20 ms (without prescaler). This period of this timer may be variant slightly because of temperature, voltage, and process variation. If a longer time-out period is desired, a prescaler with a division ratio of up to 1:128 can be assigned to the WDT controlled by the WDT_CTL register <2:0>. Thus, the longest time-out period is approximately 2.56 seconds.
The CLRWDT instruction clears the WDT and prevents it from timing out and generating a device reset.
The SLEEP instruction also resets the WDT. This gives the maximum SLEEP time before a WDT Wake-up Reset.

There are two type of watchdog timer mode could be selected by I_WDT (WDT_CTL <6>). When I_WDT bit disable, normal watchdog timer reset is selected. During normal operation or in SLEEP mode, a WDT time-out will cause the device reset and the $\overline{TO}$ bit (STATUS<4>) will be cleared.
If I_WDT bit enabled, the internal watchdog timer wakeup will be used. The system wakeups from sleep, then jumps into interrupt vector with external interrupt request PAIF (INTFLAG<5>) and continues from next instruction instead of triggering a reset event. There is a stabilization time required for internal watchdog wakeup could be selected by I_ TWDT (WDT_CTL<5>). The default value of this stabilization timer is 5ms.

**EXAMPLE 2.4: Typical Watchdog Reset**

```
START                                   // Program Start
        CLRWDT
        MOVIA       0x87
        MOVAR       WDT_CTL             // Sleep: 2.56s + Wakeup: 20ms
        …
        …
        SLEEP
        NOP
Watchdog Reset  …
        …
```

**EXAMPLE 2.5: Internal Watchdog Wakeup**

```
START                                   // Program Start
        MOVIA   0xA0
        MOVAR   INTEN       // Enable global & PORTA Interrupt
        CLRWDT
        MOVIA   0xE7
        MOVAR   WDT_CTL     // Sleep: 2.56s + Wakeup: 5ms
        …
        …
        SLEEP
        NOP                                          <-- WDT Wakeup
        CLRWDT              // WDT wakeup event will keep //occurring if WDT
        …                   not cleared.
ORG     0x004               <-- WDT Wakeup
        …
        BCR     INTFLAG,5
        …
        RETFIE              IRQ Return
```

## 2.6 Reset

FM8PA76 devices may be RESET in one of the following ways:

1. Power-on Reset (POR)
2. Brown-out Reset (BOR)
3. RSTB Pin Reset
4. WDT time-out Reset

Some registers are not affected in any RESET condition. Their status is unknown on Power-on Reset and unchanged in any other RESET. Most other registers are reset to a “reset state” on Power-on Reset, RSTB or WDT Reset.

A Power-on RESET pulse is generated on-chip when Vdd rise is detected. To use this feature, the user merely ties the RSTB pin to Vdd.

On-chip Low Voltage Detector (LVDT) places the device into reset when Vdd is below a fixed voltage. This ensures that the device does not continue program execution outside the valid operation Vdd range. Brown-out RESET is typically used in AC line or heavy loads switched applications.

A RSTB or WDT Wake-up from SLEEP also results in a device RESET, and not a continuation of operation before SLEEP.

The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits (STATUS<4:3>) are set or cleared depending on the different reset conditions.

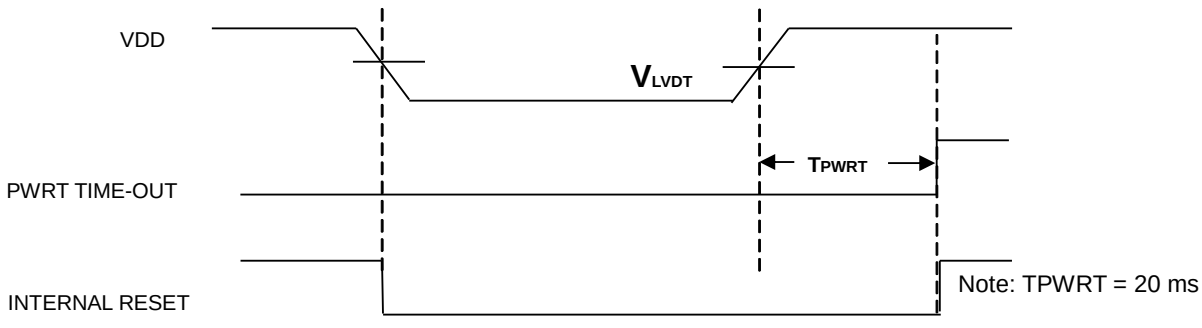
### 2.6.1 Power-up Reset Timer(PWRT)

The Power-up Reset Timer provides a nominal 20ms delay after Power-on Reset (POR), Brown-out Reset (BOR), RSTB Reset or WDT time-out Reset. The device is kept in reset state as long as the PWRT is active.

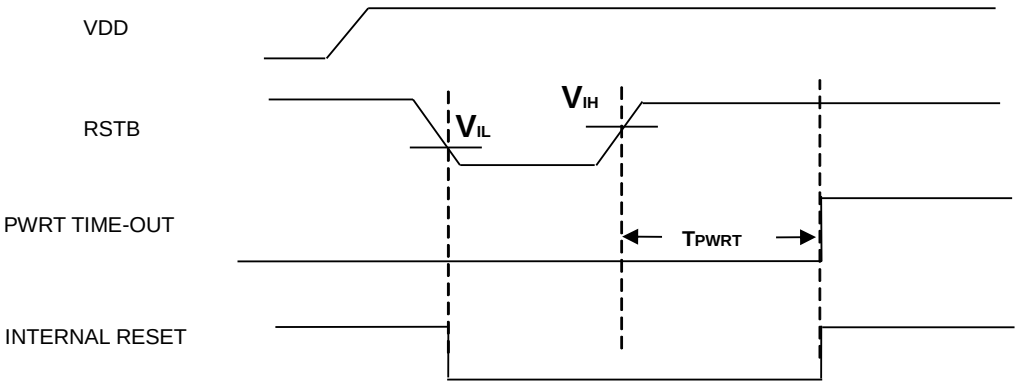
The PWDT delay will vary from device to device due to Vdd, temperature, and process variation.

**FIGURE 2.7: Reset Timing**

**LVDT on/RSTB Disable**



**LVDT off/RSTB Enable**




Note: TPWRT = 20 ms

**LVDT off/RSTB Disable**

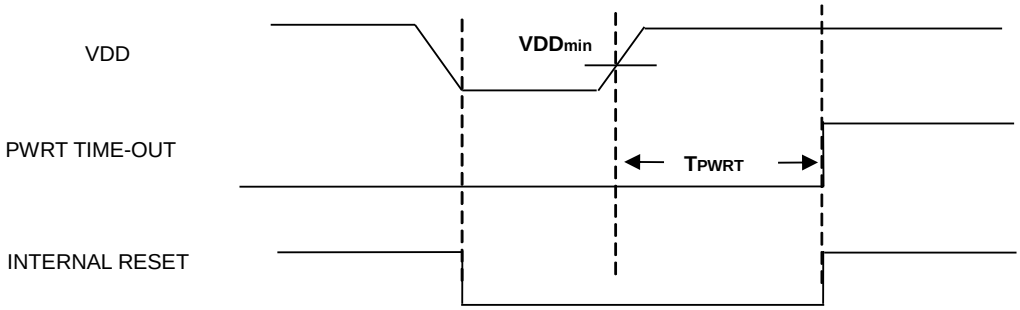



**FIGURE 2.8: Simplified Block Diagram of on-chip Reset Circuit**

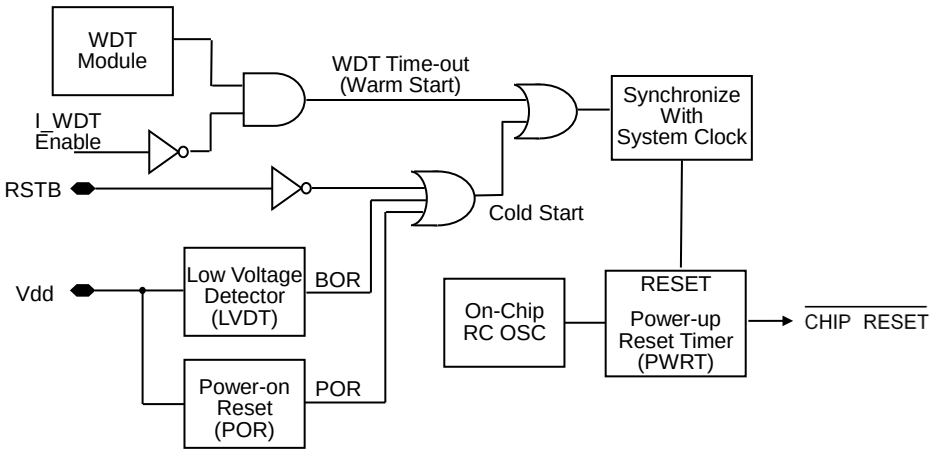


**TABLE 2.1: Reset Conditions for Operational Registers**

| Address | Name | Power-on Reset<br>Brown-out Reset<br>RSTB Reset | WDT Reset |
|---|---|---|---|
| N/A | Accumulator | xxxx xxxx | uuuu uuuu |
| 00H | INDF | xxxx xxxx | uuuu uuuu |
| 01H | PCL | 0000 0000 | 0000 0000 |
| 02H | PCH | ---- -000 | ---- -000 |
| 03H | STATUS | 0001 1xxx | 0001 1xxx |
| 04H | FSR | xxxx xxxx | uuuu uuuu |
| 05H | IOSTA | 1111 1111 | 1111 1111 |
| 06H | PORTA | xxxx xxxx | uuuu uuuu |
| 07H | IOSTB | xxxx xx11 | xxxx xx11 |
| 08H | PORTB | xxxx xxxx | xxxx xxuu |
| 09H | IOSTC | 1111 111x | 1111 111x |
| 0AH | PORTC | xxxx xxxx | uuuu uuux |
| 10H | TMR0_CTL | 0000 0000 | 0000 0000 |
| 11H | TMR0L_LA (Don't write FF) | 0000 0000 | 0000 0000 |
| 12H | TMR0H_LA (Don't write FF) | 0000 0000 | 0000 0000 |
| 13H | TMR0L_CNT (Read only) | 0000 0000 | 0000 0000 |
| 14H | TMR0H_CNT (Read only) | 0000 0000 | 0000 0000 |
| 15H | TMR1_CTL1 | 0000 0000 | 0000 0000 |
| 16H | TMR1_CTL2 | 0000 0000 | 0000 0000 |
| 17H | TMR1_LA (Don't write FF) | 0000 0000 | 0000 0000 |
| 18H | TMR1_CNT (Read only) | 0000 0000 | 0000 0000 |
| 19H | TMR2_CTL1 | 0000 0000 | 0000 0000 |
| 1AH | TMR2_LA (Don't write FF) | 0000 0000 | 0000 0000 |
| 1BH | TMR2_CNT (Read only) | 0000 0000 | 0000 0000 |
| 1CH | TMR3_CTL1 | 0000 0000 | 0000 0000 |
| 1DH | TMR3_CTL2 | 0000 0000 | 0000 0000 |
| 1EH | TMR3_LA (Don't write FF) | 0000 0000 | 0000 0000 |
| 1FH | TMR3_CNT (Read only) | 0000 0000 | 0000 0000 |
| 20H | TMR4_CTL1 | 0000 0000 | 0000 0000 |
| 21H | TMR4_LA (Don't write FF) | 0000 0000 | 0000 0000 |
| 22H | TMR4_CNT (Read only) | 0000 0000 | 0000 0000 |
| 25H | INTEN | 0000 0000 | 0000 0000 |
| 26H | INTFLAG | x000 0000 | x000 0000 |

| Address | Name | Power-on Reset<br>Brown-out Reset<br>RSTB Reset | WDT Reset |
|---|---|---|---|
| 29H | AD_CTL1 | 0x0x 0000 | 0x0x 0000 |
| 2AH | AD_CTL2 | 0xxx x000 | 0xxx x000 |
| 2BH | AD_CTL3 | xxxx 0000 | xxxx 0000 |
| 2CH | AD_DATL | 0000 xxxx | 0000 xxxx |
| 2DH | AD_DATH | 0000 0000 | 0000 0000 |
| 2FH | SYS_CLK | 0xxx xx00 | 0xxx xx00 |
| 30H | CLO_CTL | 000x 0000 | 000x 0000 |
| 31H | APHCON | 0000 0000 | 0000 0000 |
| 32H | BPHCON | xxxx xx00 | xxxx xx00 |
| 33H | CPHCON | 0000 000x | 0000 000x |
| 3AH | INT_PA | 0000 0000 | 0000 0000 |
| 3DH | WDT_CTL | 100x x111 | 100x x111 |
| 3EH | TB_BNK | xxxx x000 | xxxx x000 |

Legend: x = unknown, - = unimplemented and read “0”, u = unchanged

## 2.7 Interrupt

The FM8PA76 has three kinds of interrupt sources:
1. 8 External IOA<0:7> pin changed interrupt
2. 5 Timers / Counters overflow interrupt
3. an end of AD conversion interrupt

INTFLAG is the interrupt flag register that recodes the interrupt requests to the relative flags.
A global interrupt enable bit, GIE (INTEN<7>), enables (if set) all un-masked interrupts or disables (if cleared) all interrupts. Individual interrupts can be enabled/ disabled through their corresponding enable bits in INTEN register regardless of the status of the GIE bit.
When an interrupt event occur with the GIE bit and its corresponding interrupt enable bit are all set, the GIE bit will be cleared by hardware to disable any further interrupts, and the next instruction will be fetched from address 004h.
The interrupt flag bits must be cleared by software before re-enabling GIE bit to avoid recursive interrupts.
The RETFIE instruction exits the interrupt routine and set the GIE bit to re-enable interrupt.
The RETF instruction exits the interrupt routine and does NOT set the GIE bit.
The flag bit in INTFLAG register is set by interrupt event regardless of the status of its mask bit.

### 2.7.1 PORTA<0:7> External Interrupt and Wakeup Function

The external interrupt on PORTA<0:7> are selected by INT_PA<0:7> and PAIE (INTEN<5>). When the device is in normal mode and the specified IO status changed, the interrupt event will be triggered and the program will jump to 004h.
When the device is in sleep mode, those interrupts can also be used as an external wakeup signal. The device will restart system clock and the program will jump to 004h after startup timer timeout.

**EXAMPLE 2.6: External IOA<0> pin change interrupt**

```
START                                   // Program Start
                  …
                  MOVIA     0xFF
                  MOVAR     IOSTA       ; Set PORTA as input
                  MOVIA     0xA0
                  MOVAR     INTEN       ; Enable global & PORTA Interrupt
                  BCR       INTFLAG,5   ; Clear PORTA Flag
                  MOVR      PORTA,0     ; Read Port A status
                  MOVIA     0x01
                  MOVAR     INT_PA      ;Set IOA0 Pin Change wakeup
                  …
IOA0              …
Interrupt         …
Occurred          …

ORG               0x004
                  …
                  BCR       INTFLAG,5
                  …
                  RETFIE                IRQ Return
```

**EXAMPLE 2.7: External IOA<0> pin change Wakeup/ Interrupt**

```
START                                   // Program Start
                  …
                  MOVIA     0xFF
                  MOVAR     IOSTA       ; Set PORTA as input
                  MOVIA     0xA0
                  MOVAR     INTEN       ; Enable global & PORTA Interrupt
                  BCR       INTFLAG,5   ; Clear PORTA Flag
                  MOVR      PORTA,0     ; Read Port A status
                  MOVIA     0x01
                  MOVAR     INT_PA      ;Set IOA0 Pin Change wakeup
                  BTRSS     INTFLAG,5
                  SLEEP
IOA0              NOP                   ; System Wakeup
Interrupt         …                     ; Wakeup timer timeout
Occurred          …                     ; Jump to 0x004

ORG               0x004
                  …
                  BCR       INTFLAG,5
                  …
                  RETFIE                IRQ Return
```

## 2.8 Analog to Digital Converter (ADC)

This analog to digital converter has 9 channels 12bits (10+2) resolution. The ADC is controlled by three control register, ADC_CTL1, ADC_CTL2, and ADC_CTL2. The FM8PA76 provides two operation modes, AD conversion mode and comparator mode. The operation mode can be selected by MODE (AD_CTL1<5>).
In AD conversion mode, the AD_DATL and AD_DATH register shows the AD conversion result. If the comparator selected, the data written to those two registers will be compared to the converted data of input voltage. The result will be shown in CMP_D (AD_CTL2<7>).

**EXAMPLE 2.8: Analog to Digital Conversion (Channel0 AD conversion)**

```
…
BTRSC   AD_CTL1,7
LGOTO   $-1             ; Make Sure no ADC is processing
BCR     INTFLAG,6
MOVIA   0x00
MOVAR   AD_CTL1         ; Select ADC Channel 0 conversion
MOVIA   0x03
MOVAR   AD_CTL2         ; Set AD conversion rate:  System clock /128
MOVIA   0x01
MOVAR   AD_CTL3         ; Set AN0 analog input
BSR     AD_CTL1,7       ; ADC start
BTRSS   INTFLAG ,6
LGOTO   $-1             ; Wait AD end of conversion
MOVR    AD_DATH,0       ; Read ADC high byte data
…
MOVR    AD_DATL,0       ; Read ADC Low byte data
…
```

## 2.9 Look-Up Table Function

The Look-up Table function is built-in to access the data table within entire ROM area. The TB_BNK register is used to address the high byte of the location of required ROM. The instructions TABL and TABH are used to read low byte and high byte of the addressed ROM. The result of instructions will be stored at ACC register. Please refer to the following example for detail.

```
            …
            MOVIA   03H
            MOVAR   0x5B        ; Save 03H to register 0x5B (low bit address)
            MOVIA   0x07
            MOVAR   TB_BNK      ; Save 0x07 value to TAB_BNK (high bit address)
            TABL    0x5B        ; Read Low Byte 0x0703 ROM Data, and saved it to A register.
                                ; ACC = 0xAA
            TABH    0x5B        ; Read High Byte 0x0703 ROM Data, and saved it to A register.
                                ; ACC = 0x55

(address)   ORG     0700H       ; Program start from 0700H of ROM
  0700H     DW      1122H       ; Store ROM Table Start from 0700H
  0701H     DW      3344H
  0702H     DW      6677H
  0703H     DW      55AAH
```

## 2.10 Hexadecimal Convert to Decimal (HCD)

Decimal format is another number format for FM8PA76. When the content of the data memory has been assigned as decimal format, it is necessary to convert the results to decimal format after the execution of ALU instructions. When the decimal converting operation is processing, all of the operand data (including the contents of the data memory (RAM), accumulator (ACC), immediate data, and look-up table) should be in the decimal format, or the results of conversion will be incorrect.

Instruction DAA can convert the ACC data from hexadecimal to decimal format after any addition operation and restored to ACC. The conversion operation is illustrated in example 2.9.

**EXAMPLE 2.9: DAA CONVERSION**

```
MOVIA   90h     ;Set immediate data = decimal format number “90” (ACC ← 90h)
MOVAR   40h     ;Load immediate data “90” to data memory address 40H
MOVIA   10h     ;Set immediate data = decimal format number “10” (ACC ← 10h)
ADDAR   40h, 0  ;Contents of the data memory address 40H and ACC are binary-added
                ;the result loads to the ACC (ACC ← A0h, C ← 0)
DAA     40h, 0  ;Convert the content of ACC to decimal format, and restored to ACC
                ;The result in the ACC is “00” and the carry bit C is “1”. This represents the
                ;decimal number “100”
```

Instruction DAS can convert the ACC data from hexadecimal to decimal format after any subtraction operation and restored to ACC. The conversion operation is illustrated in example 2.10.

**EXAMPLE 2.10: DAS CONVERSION**

```
MOVIA   10h     ;Set immediate data = decimal format number “10” (ACC ← 10h)
MOVAR   40h     ;Load immediate data “10” to data memory address 40H
MOVIA   20h     ;Set immediate data = decimal format number “20” (ACC ← 20h)
SUBAR   40h, 0  ;Contents of the data memory address 40H and ACC are binary-subtracted
                ;the result loads to the ACC (ACC ← F0h, C ← 0)
DAS     40h, 0  ;Convert the content of ACC to decimal format, and restored to ACC
                ;The result in the ACC is “90” and the carry bit C is “0”. This represents the
                ;decimal number “ -10”
```

## 2.11 Oscillator Configurations

FM8PA76 can be operated in eight different combinations of oscillator modes. Users can program configuration words (Fosc) to select the appropriate modes. The eight different system clock modes are combination of the following oscillators:

- LF: Low Frequency Crystal Oscillator
- XT: Crystal/Resonator Oscillator
- HF: High Frequency Crystal/Resonator Oscillator
- ERC: External Resistor/ Voltage Controlled Oscillator
- IRC: Internal Resistor/Capacitor Oscillator

In LF, XT, or HF modes, a crystal or ceramic resonator in connected to the OSCI and OSCO pins to establish oscillation. When in LF, XT, or HF modes, the devices can have an external clock source drive the OSCI pin.

The ERC device option offers additional cost savings for timing insensitive applications. The RC oscillator frequency is a function of the supply voltage, the resistor (Rext) and capacitor (Cext), the operating temperature, and the process parameter.

The IRC option offers largest cost savings for timing insensitive applications.

**FIGURE 2.9: HF, XT or LF Oscillator Modes (Crystal Operation or Ceramic Resonator)**

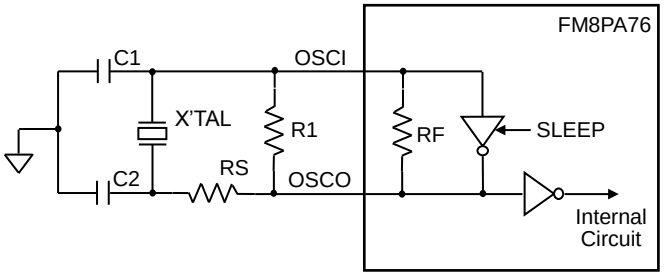



**FIGURE 2.10: HF, XT or LF Oscillator Modes (External Clock Input Operation)**

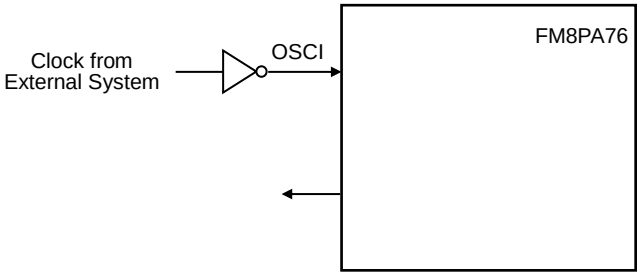
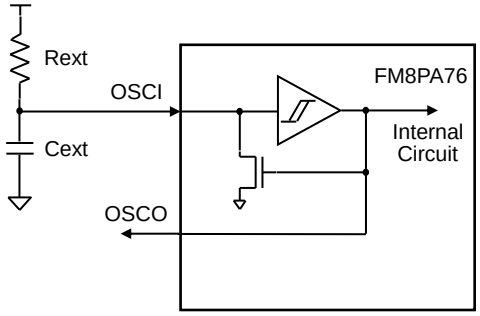



**FIGURE 2.11: ERC (External Resistor Controlled) Oscillator Mode**

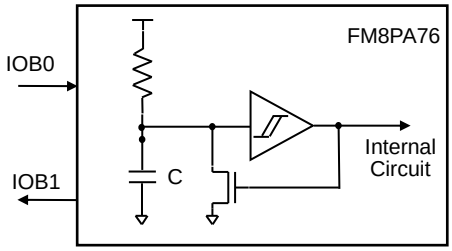



The typical oscillator frequency vs. external resistor is as following table
When Cext = 0.01uf (103)

| 5V | | 3V | |
|---|---|---|---|
| Rext | Frequency | Rext | Frequency |
| 4.3M | 32 KHZ | 3.6M | 32 KHZ |
| 220K | 500 KHZ | 189K | 500 KHZ |
| 107K | 1.0 MHZ | 100K | 1.0 MHZ |
| 56K | 2.0 MHZ | 53K | 2.0 MHZ |
| 30K | 4.0 MHZ | 28K | 4.0 MHZ |
| 16K | 8.0 MHZ | 15K | 8.0 MHZ |
| 11K | 12.0 MHZ | 11K | 12.0 MHZ |

Note: Values are provided for design reference only.

**FIGURE 2.12: IRC Oscillator Mode (Internal R, Internal C Oscillator)**




## 2.12 Configuration Words

**TABLE 2.2: Configuration Words**

| Name | Description |
|---|---|
| Fosc | Oscillator Selection Bits<br>→ IRC(4MHz) mode (default)<br>→ HF crystal & IRC(4MHz) mode<br>→ XT crystal & IRC(4MHz) mode<br>→ LF crystal & IRC(4MHz) mode<br>→ ERC & IRC(4MHz) mode<br>→ HF crystal mode<br>→ XT crystal mode<br>→ LF crystal mode |
| WDTEN | Watchdog Timer Enable Bit<br>→ WDT enabled (default)<br>→ WDT disabled |
| LVDT | Low Voltage Detector Selection Bit<br>→ LVDT = 2.6V (default)<br>→ LVDT = 2.2V<br>→ OFF<br>→ LVDT = 3.7V |
| RSTBIN | IOB2/RSTB Pin Selection Bit<br>→ RSTB pin is selected (default)<br>→ IOB2 pin is selected |
| CPU_S | Instruction Period Selection Bits<br>→ four oscillator periods (4T) (default)<br>→ two oscillator periods (2T) |
| CAL | Calibration Selection Bits for IRC Mode |
| PROTECT | Code Protection Bit<br>→ NO, EPROM code protection off (default)<br>→ YES, EPROM code protection on |

**TABLE 2.3: Selection of IOB1/OSCO Pin**

| Mode of oscillation | IOB1/OSCO Pin Selection |
|---|---|
| IRC | IOB1 |
| ERC | OSCO |
| HF, XT, LF | OSCO |

**TABLE 2.4: Selection of IOB0/OSCI Pin**

| Mode of oscillation | IOB0/OSCI Pin Selection |
|---|---|
| IRC | IOB0 |
| ERC | OSCI |
| HF, XT, LF | OSCI |

## 3.0 INSTRUCTION SET

| Mnemonic, Operands | | Description | Operation | Cycles | Status Affected |
|---|---|---|---|---|---|
| **BCR** | **R, bit** | Clear bit in R | 0 → R<b> | 1 | - |
| **BSR** | **R, bit** | Set bit in R | 1 → R<b> | 1 | - |
| **BTRSC** | **R, bit** | Test bit in R, Skip if Clear | Skip if R<b> = 0 | 1/2(1) | - |
| **BTRSS** | **R, bit** | Test bit in R, Skip if Set | Skip if R<b> = 1 | 1/2(1) | - |
| **NOP** | | No Operation | No operation | 1 | - |
| **CLRWDT** | | Clear Watchdog Timer | 00h → WDT,<br>00h → WDT prescaler | 1 | $\overline{\text{TO}}$, $\overline{\text{PD}}$ |
| **SLEEP** | | Go into power-down mode | 00h → WDT,<br>00h → WDT prescaler | 1 | $\overline{\text{TO}}$, $\overline{\text{PD}}$ |
| **TABL** | **R** | Read low byte ROM table to (acc)<br>ROM table address={TB_BNK,index of R } | ACC=ROM{BANK index:<br>R }[7:0] | 2 | - |
| **TABH** | **R** | Read high byte ROM table to (acc)<br>ROM table address={TB_BNK,index of R } | ACC=ROM{BANK index :<br>R }[15:8] | 2 | - |
| **DAA** | **R, d** | Adjust data format of register from HEX to DEC after any addition operation | R(hex) → dest (dec) | 1 | C |
| **DAS** | **R, d** | Adjust data format of register from HEX to DEC after any subtraction operation | R(hex) → dest (dec) | 1 | C |
| **RETURN** | | Return from subroutine | Top of Stack → PC | 2 | - |
| **RETFIE** | | Return from interrupt, set GIE bit | Top of Stack → PC,<br>1 → GIE | 2 | - |
| **RETF** | | Return from interrupt | Top of Stack → PC, | 2 | - |
| **CLRA** | | Clear ACC | 00h → ACC | 1 | Z |
| **CLRR** | **R** | Clear R | 00h → R | 1 | Z |
| **MOVAR** | **R** | Move ACC to R | ACC → R | 1 | - |
| **MOVR** | **R, d** | Move R | R → dest | 1 | Z |
| **MOV2** | **R, d** | Move R | R → dest | 1 | - |
| **DECR** | **R, d** | Decrement R | R - 1 → dest | 1 | Z |
| **DECRSZ** | **R, d** | Decrement R, Skip if 0 | R - 1 → dest,<br>Skip if result = 0 | 1/2(1) | - |
| **INCR** | **R, d** | Increment R | R + 1 → dest | 1 | Z |
| **INCRSZ** | **R, d** | Increment R, Skip if 0 | R + 1 → dest,<br>Skip if result = 0 | 1/2(1) | - |
| **ADDAR** | **R, d** | Add ACC and R | R + ACC → dest | 1 | C, DC, Z |
| **SUBAR** | **R, d** | Subtract ACC from R | R - ACC → dest | 1 | C, DC, Z |
| **ADCAR** | **R, d** | Add ACC and R with Carry | R + ACC + C → dest | 1 | C, DC, Z |
| **SBCAR** | **R, d** | Subtract ACC from R with Carry | R + $\overline{\text{ACC}}$ + C → dest | 1 | C, DC, Z |
| **ANDAR** | **R, d** | AND ACC with R | ACC and R → dest | 1 | Z |
| **IORAR** | **R, d** | Inclusive OR ACC with R | ACC or R → dest | 1 | Z |
| **XORAR** | **R, d** | Exclusive OR ACC with R | R xor ACC → dest | 1 | Z |
| **COMR** | **R, d** | Complement R | $\overline{\text{R}}$ → dest | 1 | Z |
| **RL** | **R, d** | Rotate left R | R<6:0> → dest<7:1>,<br>R<7>→ dest<0> | 1 | - |
| **RLR** | **R, d** | Rotate left R through Carry | R<7> → C,<br>R<6:0> → dest<7:1>,<br>C → dest<0> | 1 | C |

| Mnemonic, Operands | | Description | Operation | Cycles | Status Affected |
|---|---|---|---|---|---|
| **RL0** | **R, d** | Rotate left R through 0 | R<6:0> → dest<7:1>,<br>0 → dest<0> | 1 | - |
| **RL1** | **R, d** | Rotate left R through 1 | R<6:0> → dest<7:1>,<br>1 → dest<0> | 1 | - |
| **RR** | **R, d** | Rotate right R | R<7:1> → dest<6:0>,<br>R<0> → dest<7> | 1 | - |
| **RRR** | **R, d** | Rotate right R through Carry | C → dest<7>,<br>R<7:1> → dest<6:0>,<br>R<0> → C | 1 | C |
| **RR0** | **R, d** | Rotate right R with 0 | 0 → dest<7>,<br>R<7:1> → dest<6:0>, | 1 | - |
| **RR1** | **R, d** | Rotate right R with 1 | 1 → dest<7>,<br>R<7:1> → dest<6:0>, | 1 | - |
| **SWAPR** | **R, d** | Swap R | R<3:0> → dest<7:4>,<br>R<7:4> → dest<3:0> | 1 | - |
| **MOVIA** | **I** | Move Immediate to ACC | I → ACC | 1 | - |
| **ADDIA** | **I** | Add ACC and Immediate | I + ACC → ACC | 1 | C, DC, Z |
| **SUBIA** | **I** | Subtract ACC from Immediate | I - ACC → ACC | 1 | C, DC, Z |
| **ANDIA** | **I** | AND Immediate with ACC | ACC and I → ACC | 1 | Z |
| **IORIA** | **I** | OR Immediate with ACC | ACC or I → ACC | 1 | Z |
| **XORIA** | **I** | Exclusive OR Immediate to ACC | ACC xor I → ACC | 1 | Z |
| **RETIA** | **I** | Return, place Immediate in ACC | I → ACC,<br>Top of Stack → PC | 2 | - |
| **LCALL** | **I** | Call subroutine | PC + 1 → Top of Stack,<br>I → PC<10:0><br>PC<10:8> → PCH<2:0> | 2 | - |
| **LGOTO** | **I** | Unconditional branch | I → PC<10:0><br>PC<10:8> → PCH<2:0> | 2 | - |
| **TMSZA** | | If (ACC) =0, skip next instruction | Skip if ACC = 0 | 1/2(1) | - |
| **TMSZR** | **R** | If (R) =0, skip next instruction | Skip if R = 0 | 1/2(1) | - |
| **TMSNZR** | **R** | If (R) ≠ 0, skip next instruction | Skip if R ≠ 0 | 1/2(1) | - |
| **TMCOMP** | **R** | If (acc) =(R), skip next instruction | Skip if (acc) =(R) | 1/2(1) | - |
| **TMCOMPB** | **R** | If (acc) ≠(R), skip next instruction | Skip if (acc) ≠ (R) | 1/2(1) | - |

Note: 1. 2 cycles for skip, else 1 cycle.
2. bit : Bit address within an 8-bit register R
R : Register address (00h to BFh)
I : Immediate data
ACC : Accumulator
d : Destination select;
=0 (store result in ACC)
=1 (store result in file register R)
dest : Destination
PC : Program Counter
PCH : High Byte register of Program Counter
WDT : Watchdog Timer Counter
GIE : Global interrupt enable bit
$\overline{\text{TO}}$ : Time-out bit
$\overline{\text{PD}}$ : Power-down bit
C : Carry bit
DC : Digital carry bit
Z : Zero bit
3. CLRR STATUS will set Z bit and leave the remaining bits unchanged.

| **ADCAR** | **Add ACC and R with Carry** |
|---|---|
| Syntax: | ADCAR R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R + ACC + C → dest |
| Status Affected: | C, DC, Z |
| Description: | Add the contents of the ACC register and register 'R' with Carry. If 'd' is 0 the result is stored in the ACC register. If 'd' is '1' the result is stored back in register 'R'. |
| Cycles: | 1 |

| **ADDAR** | **Add ACC and R** |
|---|---|
| Syntax: | ADDAR R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | ACC + R → dest |
| Status Affected: | C, DC, Z |
| Description: | Add the contents of the ACC register and register 'R'. If 'd' is 0 the result is stored in the ACC register. If 'd' is '1' the result is stored back in register 'R'. |
| Cycles: | 1 |

| **ADDIA** | **Add ACC and Immediate** |
|---|---|
| Syntax: | ADDIA I |
| Operands: | 0 ≤ I ≤ 0xFF |
| Operation: | ACC + I → ACC |
| Status Affected: | C, DC, Z |
| Description: | Add the contents of the ACC register with the 8-bit immediate 'I'. The result is placed in the ACC register. |
| Cycles: | 1 |

| **ANDAR** | **AND ACC and R** |
|---|---|
| Syntax: | ANDAR R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | ACC and R → dest |
| Status Affected: | Z |
| Description: | The contents of the ACC register are AND'ed with register 'R'. If 'd' is 0 the result is stored in the ACC register. If 'd' is '1' the result is stored back in register 'R'. |
| Cycles: | 1 |

| **ANDIA** | **AND Immediate with ACC** |
|---|---|
| Syntax: | ANDIA I |
| Operands: | 0 ≤ I ≤ 0xFF |
| Operation: | ACC AND I → ACC |
| Status Affected: | Z |
| Description: | The contents of the ACC register are AND'ed with the 8-bit immediate 'I'. The result is placed in the ACC register. |
| Cycles: | 1 |

| **BCR** | **Clear Bit in R** |
|---|---|
| Syntax: | BCF R, b |
| Operands: | 0 ≤ R ≤ 0xBF<br>0 ≤ b ≤ 7 |
| Operation: | 0 → R<b> |
| Status Affected: | None |
| Description: | Clear bit 'b' in register 'R'. |
| Cycles: | 1 |

| **BSR** | **Set Bit in R** |
|---|---|
| Syntax: | BSR R, b |
| Operands: | 0 ≤ R ≤ 0xBF<br>0 ≤ b ≤ 7 |
| Operation: | 1 → R<b> |
| Status Affected: | None |
| Description: | Set bit 'b' in register 'R'. |
| Cycles: | 1 |

| **BTRSC** | **Test Bit in R, Skip if Clear** |
|---|---|
| Syntax: | BTRSC R, b |
| Operands: | 0 ≤ R ≤ 0xBF<br>0 ≤ b ≤ 7 |
| Operation: | Skip if R<b> = 0 |
| Status Affected: | None |
| Description: | If bit 'b' in register 'R' is 0 then the next instruction is skipped.<br>If bit 'b' is 0 then next instruction fetched during the current instruction execution is discarded, and a NOP is executed instead making this a 2-cycle instruction. |
| Cycles: | 1/2 |

| **BTRSS** | **Test Bit in R, Skip if Set** |
|---|---|
| Syntax: | BTRSS R, b |
| Operands: | 0 ≤ R ≤ 0xBF<br>0 ≤ b ≤ 7 |
| Operation: | Skip if R<b> = 1 |
| Status Affected: | None |
| Description: | If bit 'b' in register 'R' is '1' then the next instruction is skipped.<br>If bit 'b' is '1', then the next instruction fetched during the current instruction execution, is discarded and a NOP is executed instead, making this a 2-cycle instruction. |
| Cycles: | 1/2 |

| **CLRA** | **Clear ACC** |
|---|---|
| Syntax: | CLRA |
| Operands: | None |
| Operation: | 00h → ACC;<br>1 → Z |
| Status Affected: | Z |
| Description: | The ACC register is cleared. Zero bit (Z) is set. |
| Cycles: | 1 |

| **CLRR** | **Clear R** |
|---|---|
| Syntax: | CLRR R |
| Operands: | $0 \leq R \leq 0xBF$ |
| Operation: | 00h → R;<br>1 → Z |
| Status Affected: | Z |
| Description: | The contents of register 'R' are cleared and the Z bit is set. |
| Cycles: | 1 |

| **CLRWDT** | **Clear Watchdog Timer** |
|---|---|
| Syntax: | CLRWDT |
| Operands: | None |
| Operation: | 00h → WDT;<br>1 → $\overline{TO}$;<br>1 → $\overline{PD}$ |
| Status Affected: | $\overline{TO}$, $\overline{PD}$ |
| Description: | The CLRWDT instruction resets the WDT. The status bits $\overline{TO}$ and $\overline{PD}$ will be set. |
| Cycles: | 1 |

| **COMR** | **Complement R** |
|---|---|
| Syntax: | COMR R, d |
| Operands: | $0 \leq R \leq 0xBF$<br>$d \in [0,1]$ |
| Operation: | $\overline{R}$ → dest |
| Status Affected: | Z |
| Description: | The contents of register 'R' are complemented. If 'd' is 0 the result is stored in the ACC register. If 'd' is 1 the result is stored back in register 'R'. |
| Cycles: | 1 |

| **DAA** | **Adjust ACC's data format from HEX to DEC** |
|---|---|
| Syntax: | DAA R, d |
| Operands: | $0 \leq R \leq 0xBF$<br>$d \in [0,1]$ |
| Operation: | R(hex) → dest(dec) |
| Status Affected: | C |
| Description: | Convert the register data from hexadecimal to decimal format after any addition operation. If 'd' is 0 the result is stored in the ACC register. If 'd' is 1 the result is stored back in register 'R'. |
| Cycles: | 1 |

| **DAS** | **Adjust ACC's data format from HEX to DEC** |
|---|---|
| Syntax: | DAS R, d |
| Operands: | $0 \leq R \leq 0xBF$<br>$d \in [0,1]$ |
| Operation: | R(hex) → dest(dec) |
| Status Affected: | C |
| Description: | Convert the register data from hexadecimal to decimal format after any subtraction operation. If 'd' is 0 the result is stored in the ACC register. If 'd' is 1 the result is stored back in register 'R'. |
| Cycles: | 1 |

| **DECR** | **Decrement R** |
|---|---|
| Syntax: | DECR R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R - 1 → dest |
| Status Affected: | Z |
| Description: | Decrement of register 'R'. If 'd' is 0 the result is stored in the ACC register. If 'd' is 1 the result is stored back in register 'R'. |
| Cycles: | 1 |

| **DECRSZ** | **Decrement R, Skip if 0** |
|---|---|
| Syntax: | DECRSZ R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R - 1 → dest; skip if result =0 |
| Status Affected: | None |
| Description: | The contents of register 'R' are decrement. If 'd' is 0 the result is placed in the ACC register. If 'd' is 1 the result is stored back in register 'R'.<br>If the result is 0, the next instruction, which is already fetched, is discarded and a NOP is executed instead and making it a two-cycle instruction. |
| Cycles: | 1/2 |

| **LCALL** | **Subroutine Call** |
|---|---|
| Syntax: | LCALL I |
| Operands: | 0 ≤ I ≤ 0x7FF |
| Operation: | PC + 1 → Top of Stack,<br>I → PC<10:0><br>PC<10:8> → PCH<2:0> |
| Status Affected: | None |
| Description: | Subroutine call. First, return address (PC+1) is pushed onto the stack. The 11-bit immediate address is loaded into PC bits <10:0>. |
| Cycles: | 2 |

| **LGOTO** | **Unconditional Branch** |
|---|---|
| Syntax: | LGOTO I |
| Operands: | 0 ≤ I ≤ 0x7FF |
| Operation: | I → PC<10:0><br>PC<10:8> → PCH<2:0> |
| Status Affected: | None |
| Description: | LGOTO is an unconditional branch. The 11-bit immediate value is loaded into PC bits <10:0>. |
| Cycles: | 2 |

| **INCR** | **Increment R** |
|---|---|
| Syntax: | INCR R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R + 1 → dest |
| Status Affected: | Z |
| Description: | The contents of register 'R' are increment. If 'd' is 0 the result is placed in the ACC register. If 'd' is 1 the result is stored back in register 'R'. |
| Cycles: | 1 |

| **INCRSZ** | **Increment R, Skip if 0** |
|---|---|
| Syntax: | INCRSZ R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R + 1 → dest, skip if result = 0 |
| Status Affected: | None |
| Description: | The contents of register 'R' are increment. If 'd' is 0 the result is placed in the ACC register. If 'd' is the result is stored back in register 'R'.<br>If the result is 0, then the next instruction, which is already fetched, is discarded and a NOP is executed instead and making it a two-cycle instruction. |
| Cycles: | 1/2 |

| **IORAR** | **OR ACC with R** |
|---|---|
| Syntax: | IORAR R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | ACC or R → dest |
| Status Affected: | Z |
| Description: | Inclusive OR the ACC register with register 'R'. If 'd' is 0 the result is placed in the ACC register. If 'd' is 1 the result is placed back in register 'R'. |
| Cycles: | 1 |

| **IORIA** | **OR Immediate with ACC** |
|---|---|
| Syntax: | IORIA I |
| Operands: | 0 ≤ I ≤ 0xFF |
| Operation: | ACC or I → ACC |
| Status Affected: | Z |
| Description: | The contents of the ACC register are OR'ed with the 8-bit immediate 'I'. The result is placed in the ACC register. |
| Cycles: | 1 |

| **MOVAR** | **Move ACC to R** |
|---|---|
| Syntax: | MOVAR R |
| Operands: | 0 ≤ R ≤ 0xBF |
| Operation: | ACC → R |
| Status Affected: | None |
| Description: | Move data from the ACC register to register 'R'. |
| Cycles: | 1 |

| **MOVIA** | **Move Immediate to ACC** |
|---|---|
| Syntax: | MOVIA I |
| Operands: | 0 ≤ I ≤ 0xFF |
| Operation: | I → ACC |
| Status Affected: | None |
| Description: | The 8-bit immediate 'I' is loaded into the ACC register. The don't cares will assemble as 0s. |
| Cycles: | 1 |

| **MOVR** | **Move R** |
|---|---|
| Syntax: | MOVR R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R → dest |
| Status Affected: | Z |
| Description: | The contents of register 'R' is moved to destination 'd'. If 'd' is 0, destination is the ACC register. If 'd' is 1, the destination is file register 'R'. 'd' is 1 is useful to test a file register since status flag Z is affected. |
| Cycles: | 1 |

| **MOV2** | **Move R** |
|---|---|
| Syntax: | MOV2 R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R → dest |
| Status Affected: | None |
| Description: | The contents of register 'R' is moved to destination 'd'. If 'd' is 0, destination is the ACC register. If 'd' is 1, the destination is file register 'R'. The zero status flag <Z> is not affected. |
| Cycles: | 1 |

| **NOP** | **No Operation** |
|---|---|
| Syntax: | NOP |
| Operands: | None |
| Operation: | No operation |
| Status Affected: | None |
| Description: | No operation. |
| Cycles: | 1 |

| **RETF** | **Return from Interrupt** |
|---|---|
| Syntax: | RETF |
| Operands: | None |
| Operation: | Top of Stack → PC |
| Status Affected: | None |
| Description: | The program counter is loaded from the top of the stack (the return address). The 'GIE' bit would NOT be set to 1. This is a two-cycle instruction. |
| Cycles: | 2 |

| **RETFIE** | **Return from Interrupt, Set 'GIE' Bit** |
|---|---|
| Syntax: | RETFIE |
| Operands: | None |
| Operation: | Top of Stack → PC<br>1 → GIE |
| Status Affected: | None |
| Description: | The program counter is loaded from the top of the stack (the return address). The 'GIE' bit is set to 1. This is a two-cycle instruction. |
| Cycles: | 2 |

| **RETIA** | **Return with Immediate in ACC** |
|---|---|
| Syntax: | RETIA I |
| Operands: | 0 ≤ I ≤ 0xFF |
| Operation: | I → ACC;<br>Top of Stack → PC |
| Status Affected: | None |
| Description: | The ACC register is loaded with the 8-bit immediate 'I'. The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction. |
| Cycles: | 2 |

| **RETURN** | **Return from Subroutine** |
|---|---|
| Syntax: | RETURN |
| Operands: | None |
| Operation: | Top of Stack → PC |
| Status Affected: | None |
| Description: | The program counter is loaded from the top of the stack (the return address). This is a two-cycle instruction. |
| Cycles: | 2 |

| **RL** | **Rotate Left R** |
|---|---|
| Syntax: | RL R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R<6:0> → dest<7:1>,<br>R<7>→ dest<0> |
| Status Affected: | None |
| Description: | The contents of register 'R' are rotated left one bit. If 'd' is 0 the result is placed in the ACC register. If 'd' is 1 the result is stored back in register 'R'. |
| Cycles: | 1 |

| **RL0** | **Rotate Left R with 0** |
|---|---|
| Syntax: | RL0 R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R<6:0> → dest<7:1>,<br>0 → dest<0> |
| Status Affected: | None |
| Description: | The contents of register 'R' are rotated left one bit to the left and bit0 fills with "0". If 'd' is 0 the result is placed in the ACC register. If 'd' is 1 the result is stored back in register 'R'. |
| Cycles: | 1 |

| **RL1** | **Rotate Left R with 1** |
|---|---|
| Syntax: | RL1 R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R<6:0> → dest<7:1>,<br>1 → dest<0> |
| Status Affected: | None |
| Description: | The contents of register 'R' are rotated left one bit to the left and bit0 fills with "1". If 'd' is 0 the result is placed in the ACC register. If 'd' is 1 the result is stored back in register 'R'. |
| Cycles: | 1 |

| **RLR** | **Rotate Left R through Carry** |
|---|---|
| Syntax: | RLR R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R<7> → C;<br>R<6:0> → dest<7:1>;<br>C → dest<0> |
| Status Affected: | C |
| Description: | The contents of register 'R' are rotated left one bit to the left through the Carry Flag. If 'd' is 0 the result is placed in the ACC register. If 'd' is 1 the result is stored back in register 'R'. |
| Cycles: | 1 |

| **RR** | **Rotate Right R** |
|---|---|
| Syntax: | RR R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R<7:1> → dest<6:0>,<br>R<0> → dest<7> |
| Status Affected: | None |
| Description: | The contents of register 'R' are rotated right one bit. If 'd' is 0 the result is placed in the ACC register. If 'd' is 1 the result is placed back in register 'R'. |
| Cycles: | 1 |

| **RR0** | **Rotate Right R with 0** |
|---|---|
| Syntax: | RR0 R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | 0 → dest<7>,<br>R<7:1> → dest<6:0>, |
| Status Affected: | None |
| Description: | The contents of register 'R' are rotated right one bit and bit7 fills with "0". If 'd' is 0 the result is placed in the ACC register. If 'd' is 1 the result is placed back in register 'R'. |
| Cycles: | 1 |

| **RR1** | **Rotate Right R with 1** |
|---|---|
| Syntax: | RR1 R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | 1 → dest<7>,<br>R<7:1> → dest<6:0>, |
| Status Affected: | None |
| Description: | The contents of register 'R' are rotated right one bit and bit7 fills with "1". If 'd' is 0 the result is placed in the ACC register. If 'd' is 1 the result is placed back in register 'R'. |
| Cycles: | 1 |

| **RRR** | **Rotate Right R through Carry** |
|---|---|
| Syntax: | RRR R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | C → dest<7>;<br>R<7:1> → dest<6:0>;<br>R<0> → C |
| Status Affected: | C |
| Description: | The contents of register 'R' are rotated one bit to the right through the Carry Flag. If 'd' is 0 the result is placed in the ACC register. If 'd' is 1 the result is placed back in register 'R'. |
| Cycles: | 1 |

| **SLEEP** | **Enter SLEEP Mode** |
|---|---|
| Syntax: | SLEEP |
| Operands: | None |
| Operation: | 00h → WDT;<br>1 → $\overline{\text{TO}}$;<br>0 → $\overline{\text{PD}}$ |
| Status Affected: | $\overline{\text{TO}}$, $\overline{\text{PD}}$ |
| Description: | Time-out status bit ($\overline{\text{TO}}$) is set. The power-down status bit ($\overline{\text{PD}}$) is cleared. The WDT is cleared.<br>The processor is put into SLEEP mode. |
| Cycles: | 1 |

| **SBCAR** | **Subtract ACC from R with Carry** |
|---|---|
| Syntax: | SBCAR R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R + $\overline{\text{ACC}}$ + C → dest |
| Status Affected: | C, DC, Z |
| Description: | Add the 2's complement data of the ACC register from register 'R' with Carry. If 'd' is 0 the result is stored in the ACC register. If 'd' is 1 the result is stored back in register 'R'. |
| Cycles: | 1 |

| **SUBAR** | **Subtract ACC from R** |
|---|---|
| Syntax: | SUBAR R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R - ACC → dest |
| Status Affected: | C, DC, Z |
| Description: | Subtract (2's complement method) the ACC register from register 'R'. If 'd' is 0 the result is stored in the ACC register. If 'd' is 1 the result is stored back in register 'R'. |
| Cycles: | 1 |

| **SUBIA** | **Subtract ACC from Immediate** |
|---|---|
| Syntax: | SUBIA I |
| Operands: | 0 ≤ I ≤ 0xFF |
| Operation: | I - ACC → ACC |
| Status Affected: | C, DC, Z |
| Description: | Subtract (2's complement method) the ACC register from the 8-bit immediate 'I'. The result is placed in the ACC register. |
| Cycles: | 1 |

| **SWAPR** | **Swap nibbles in R** |
|---|---|
| Syntax: | SWAPR R, d |
| Operands: | 0 ≤ R ≤ 0xBF<br>d ∈ [0,1] |
| Operation: | R<3:0> → dest<7:4>;<br>R<7:4> → dest<3:0> |
| Status Affected: | None |
| Description: | The upper and lower nibbles of register 'R' are exchanged. If 'd' is 0 the result is placed in ACC register. If 'd' is 1 the result in placed in register 'R'. |
| Cycles: | 1 |

| **TABL** | **Table Look-up Low Byte** |
|---|---|
| Syntax: | TABL R |
| Operands: | 0 ≤ R ≤ 0xBF |
| Operation: | ACC=ROM{TB_BNK index : R }[7:0] |
| Status Affected: | None |
| Description: | Read low byte ROM table to (ACC)<br>ROM table address={TB_BNK index : R } |
| Cycles: | 2 |

| **TABH** | **Table Look-up High Byte** |
|---|---|
| Syntax: | TABH R |
| Operands: | 0 ≤ R ≤ 0xBF |
| Operation: | ACC=ROM{TB_BNK index : R }[15:8] |
| Status Affected: | None |
| Description: | Read High byte ROM table to (ACC)<br>ROM table address={TB_BNK index : R } |
| Cycles: | 2 |

| **TMCOMP** | **Test ACC and R, Skip if equal** |
|---|---|
| Syntax: | TMCOMP R |
| Operands: | 0 ≤ R ≤ 0xBF |
| Operation: | Skip if ACC = R |
| Status Affected: | None |
| Description: | If ACC is equal to R then the next instruction is skipped.<br>If ACC is equal to R then next instruction fetched during the current instruction execution is discarded, a NOP is executed instead and making this a 2-cycle instruction. |
| Cycles: | 1/2 |

| **TMCOMPB** | **Test ACC and R, Skip if not equal** |
|---|---|
| Syntax: | TMCOMPB R |
| Operands: | 0 ≤ R ≤ 0xBF |
| Operation: | Skip if ACC ≠ R |
| Status Affected: | None |
| Description: | If ACC is not equal to R then the next instruction is skipped.<br>If ACC is not equal to R then next instruction fetched during the current instruction execution is discarded, a NOP is executed instead and making this a 2-cycle instruction. |
| Cycles: | 1/2 |

| **TMSZA** | **Test ACC, Skip if equal to 0** |
|---|---|
| Syntax: | TMSZA |
| Operands: | None |
| Operation: | Skip if ACC = 0 |
| Status Affected: | None |
| Description: | If ACC is equal to 0 then the next instruction is skipped.<br>If ACC is equal to 0 then next instruction fetched during the current instruction execution is discarded, a NOP is executed instead and making this a 2-cycle instruction. |
| Cycles: | 1/2 |

| **TMSNZR** | **Test R, Skip if not equal to 0** |
|---|---|
| Syntax: | TMSNZR R |
| Operands: | $0 \leq R \leq 0xBF$ |
| Operation: | Skip if $R \neq 0$ |
| Status Affected: | None |
| Description: | If R is not equal to 0 then the next instruction is skipped.<br>If R is not equal to 0 then next instruction fetched during the current instruction execution is discarded, a NOP is executed instead and making this a 2-cycle instruction. |
| Cycles: | 1/2 |

| **TMSZR** | **Test R, Skip if equal to 0** |
|---|---|
| Syntax: | TMSZR R |
| Operands: | $0 \leq R \leq 0xBF$ |
| Operation: | Skip if R = 0 |
| Status Affected: | None |
| Description: | If R is equal to 0 then the next instruction is skipped.<br>If R is equal to 0 then next instruction fetched during the current instruction execution is discarded, a NOP is executed instead and making this a 2-cycle instruction. |
| Cycles: | 1/2 |

| **XORAR** | **Exclusive OR ACC with R** |
|---|---|
| Syntax: | XORAR R, d |
| Operands: | $0 \leq R \leq 0xBF$<br>$d \in [0,1]$ |
| Operation: | ACC xor R → dest |
| Status Affected: | Z |
| Description: | Exclusive OR the contents of the ACC register with register 'R'. If 'd' is 0 the result is stored in the ACC register. If 'd' is 1 the result is stored back in register 'R'. |
| Cycles: | 1 |

| **XORIA** | **Exclusive OR Immediate with ACC** |
|---|---|
| Syntax: | XORIA I |
| Operands: | $0 \leq I \leq 0xFF$ |
| Operation: | ACC xor I → ACC |
| Status Affected: | Z |
| Description: | The contents of the ACC register are XOR'ed with the 8-bit immediate 'I'. The result is placed in the ACC register. |
| Cycles: | 1 |

## 4.0 ABSOLUTE MAXIMUM RATINGS

| | |
|---|---|
| Ambient Operating Temperature | -40℃ to +85℃ |
| Store Temperature | -65℃ to +150℃ |
| DC Supply Voltage (Vdd) | 0V to +6.0V |
| Input Voltage with respect to Ground (Vss) | -0.3V to (Vdd + 0.3)V |

## 5.0 OPERATING CONDITIONS

| | |
|---|---|
| DC Supply Voltage | +2.2V to +5.5V |
| Operating Temperature | -40℃ to +85℃ |

## 6.0 ELECTRICAL CHARACTERISTICS

### 6.1 ELECTRICAL CHARACTERISTICS of FM8PA76AE/76BE/DE/EE/FE

Ta=25℃

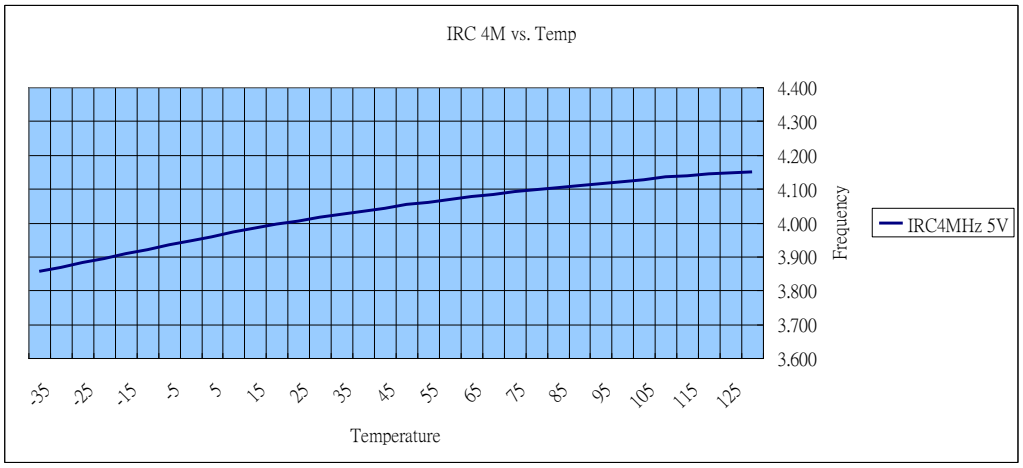
Under Operating Conditions, at four clock instruction cycles and WDT & LVDT are disabled

| Sym | Description | Conditions | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| $F_{HF}$ | X'tal oscillation range | HF mode, Vdd=5V, Fcpu=Fosc/2 | | | 20 | MHz |
| | | HF mode, Vdd=3V, Fcpu=Fosc/2 | | | 15 | |
| $F_{XT}$ | X'tal oscillation range | XT mode, Vdd=5V, Fcpu=Fosc/2 | | | 10 | MHz |
| | | XT mode, Vdd=3V, Fcpu=Fosc/2 | | | 10 | |
| $F_{LF}$ | X'tal oscillation range | LF mode, Vdd=5V, Fcpu=Fosc/2 | | | 4000 | KHZ |
| | | LF mode, Vdd=3V, Fcpu=Fosc/2 | | | 1000 | |
| $F_{ERC}$ | RC oscillation range | ERC mode, Vdd=5V, Fcpu=Fosc/2 | | | 15 | MHz |
| | | ERC mode, Vdd=3V, Fcpu=Fosc/2 | | | 7 | |
| $V_{IH}$ | Input high voltage | With schmitter | | | | V |
| | | I/O ports | 0.7vdd | | vdd | |
| | | RSTB pin | 0.8vdd | | vdd | |
| $V_{IL}$ | Input low voltage | With schmitter | | | | V |
| | | I/O ports | vss | | 0.2vdd | |
| | | RSTB pin | vss | | 0.2vdd | |
| $I_{IL}$ | Input Leakage Current | Vin = 5V, Vdd=5V | | | 1 | uA |
| | | Vin = 0V, Vdd=5V | | | 1 | |
| $I_{OH}$ | IO Drive Current | VOH =4.5V, Vdd = 5V | | 9 | | mA |
| | | VOH =4V, Vdd = 5V | | 17 | | |
| $I_{OL}$ | IO Sink Current | VOL =0.5V, Vdd = 5V | | 19 | | mA |
| | | VOL =0.75V, Vdd = 5V | | 26 | | |
| $R_{PH}$ | Pull-high resister | Input pin at Vss, vdd=5V | 65 | 130 | 195 | KΩ |
| | | Input pin at Vss, vdd=3V | 125 | 250 | 375 | |
| $I_{WDT}$ | WDT current | Vdd=5V | | 8 | | uA |
| | | Vdd=3V | | 2 | | |
| $T_{WDT}$ | WDT period | Vdd=3V | | 24 | | mS |
| | | Vdd=5V | | 20 | | |
| $I_{LVDT}$ | LVDT current | LVDT = 3.7V, vdd=5V | | 2 | | uA |
| | | LVDT = 2.6V, vdd=5V | | 3 | | |
| | | LVDT = 2.6V, vdd=3V | | 0.5 | | |
| | | LVDT = 2.2V, vdd=5V | | 3 | | |
| | | LVDT = 2.2V, vdd=3V | | 0.5 | | |
| $V_{LVDT}$ | LVDT voltage | LVDT = 3.7V | 3.5 | 3.7 | 3.9 | V |
| | | LVDT = 2.6V | 2.4 | 2.6 | 2.8 | |
| | | LVDT = 2.2V | 2.0 | 2.2 | 2.4 | |
| $V_{AD}$ | A/D input Voltage | | 0 | | VDD | V |
| $R_{AD}$ | Resolution | | | | 12 | Bits |
| DNL | A/D Differential Non-Linear | | | | 2 | LSB |
| INL | A/D Integral Non-Linear | | | | 3 | LSB |
| $I_{ADC}$ | A/D Operation Current | Vdd = 5V, Fcpu=Fosc/4 | | 600 | | uA |
| | | Vdd = 3V, Fcpu=Fosc/4 | | 100 | | |

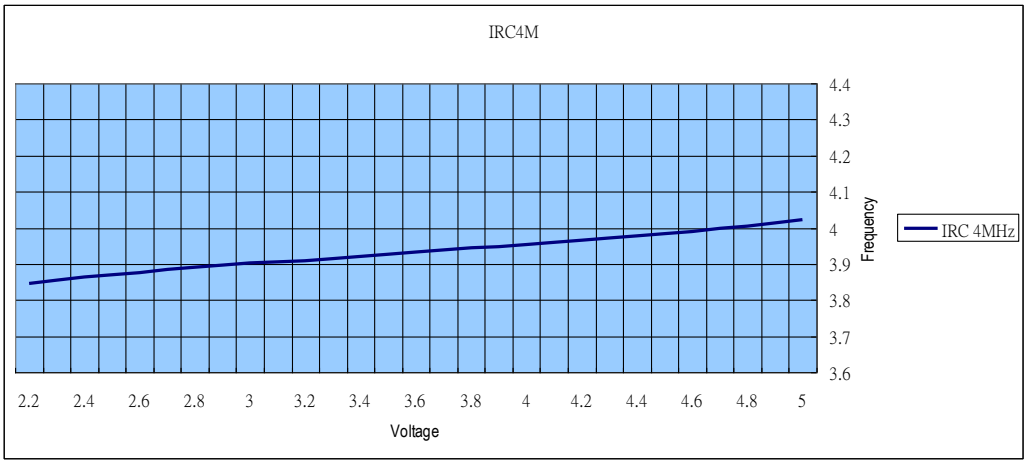
| Sym | Description | Conditions | Min. | Typ. | Max. | Unit |
|---|---|---|---|---|---|---|
| $T_{AD}$ | A/D clock period | | 8 | | | us |
| $T_{ADC}$ | A/D Conversion Time | | | 25 | | $T_{AD}$ |
| $T_{ADCS}$ | A/D Sampling Time | | | 8 | | $T_{AD}$ |
| $I_{SB}$ | Power down current | Sleep mode, Vdd=5V, WDT enable, LVDT off | | 9 | | uA |
| | | Sleep mode, Vdd=5V, WDT disable, LVDT off | | | 1 | |
| | | Sleep mode, Vdd=3V, WDT enable, LVDT off | | 3 | | |
| | | Sleep mode, Vdd=3V, WDT disable, LVDT off | | | 1 | |
| $I_{DD1}$ | Operating current | IRC mode, vdd=5V, 4 clock instruction | | 0.9 | | mA |
| $I_{DD2}$ | Operating current | IRC mode, vdd=5V, 2 clock instruction | | 1.4 | | mA |
| $I_{DD3}$ | Operating current | IRC mode, vdd=3V, 4 clock instruction | | 0.5 | | mA |
| $I_{DD4}$ | Operating current | IRC mode, vdd=3V, 2 clock instruction | | 0.7 | | mA |
| $I_{DD5}$ | Operating current | HF mode, vdd=5V, 4 clock instruction | | | | mA |
| | | 20MHz | | 4 | | |
| $I_{DD6}$ | Operating current | HF mode, vdd=5V, 2 clock instruction | | | | mA |
| | | 20MHz | | 6 | | |
| $I_{DD7}$ | Operating current | HF mode, vdd=3V, 4 clock instruction | | | | mA |
| | | 20MHZ | | 2 | | |
| $I_{DD8}$ | Operating current | XT mode, Vdd=5V, 4 clock instruction | | | | |
| | | 10MHz | | 3 | | mA |
| | | 4MHz | | 1.5 | | |
| $I_{DD9}$ | Operating current | XT mode, Vdd=5V, 2 clock instruction | | | | mA |
| | | 10MHz | | 3.5 | | |
| | | 4MHz | | 1.8 | | |
| $I_{DD10}$ | Operating current | XT mode, Vdd=3V, 4 clock instruction | | | | mA |
| | | 10MHz | | 1 | | |
| | | 4MHz | | 0.5 | | |
| $I_{DD11}$ | Operating current | XT mode, Vdd=3V, 2 clock instruction | | | | mA |
| | | 10MHz | | 1.5 | | |
| | | 4MHz | | 0.8 | | |
| $I_{DD}$ | Operating current | LF mode, Vdd=5V, 4 clock instruction | | | | uA |
| | | 32KHz | | 30 | | |
| $I_{DD12}$ | Operating current | LF mode, Vdd=5V, 2 clock instruction | | | | uA |
| | | 32KHz | | 35 | | |
| $I_{DD13}$ | Operating current | LF mode, Vdd=3V, 4 clock instruction | | | | uA |
| | | 32KHz | | 8 | | |
| $I_{DD14}$ | Operating current | LF mode, Vdd=3V, 2 clock instruction | | | | uA |
| | | 32KHz | | 10 | | |

## 6.2 ELECTRICAL CHARACTERISTICS Charts of FM8PA76AE/76BE/DE/EE/FE

### 6.2.1 Internal 4MHz RC vs. Temperature (VDD=5V)

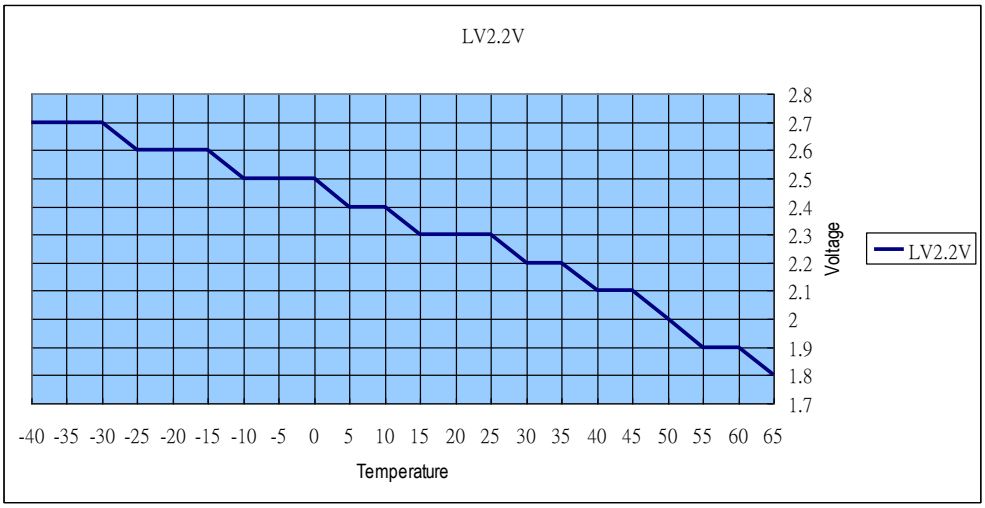



### 6.2.2 Internal 4MHz RC vs. Supply Voltage (Ta=25℃)

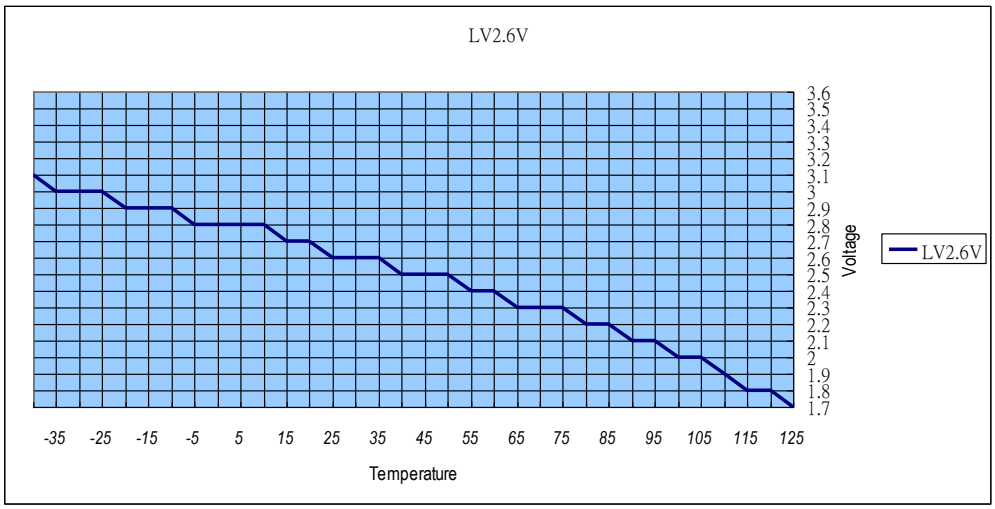



### 6.2.3 Low Voltage Detect (LVDT=2.2V) vs. Temperature

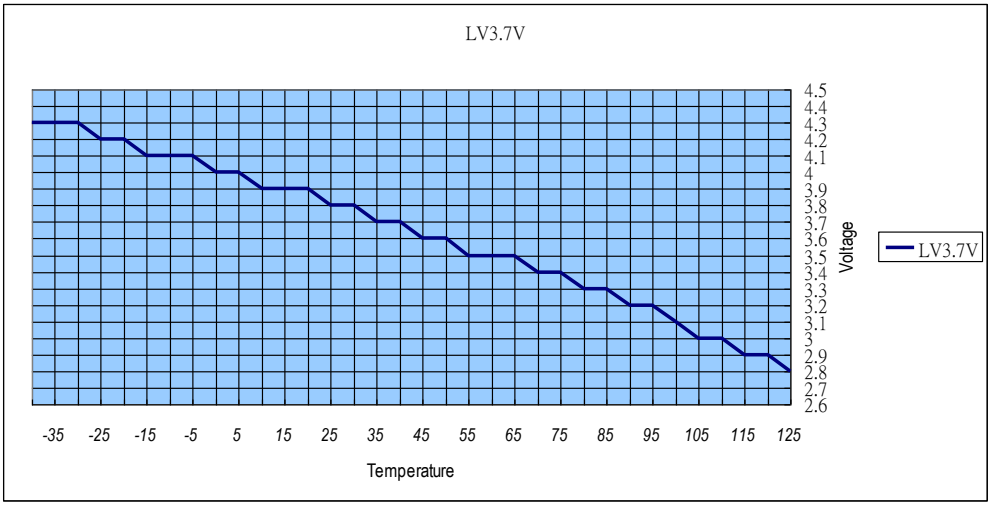
**CAUTION: The LVDT 2.2V option can only support temperature range between -40~65℃**



### 6.2.4 Low Voltage Detect (LVDT=2.6V) vs. Temperature

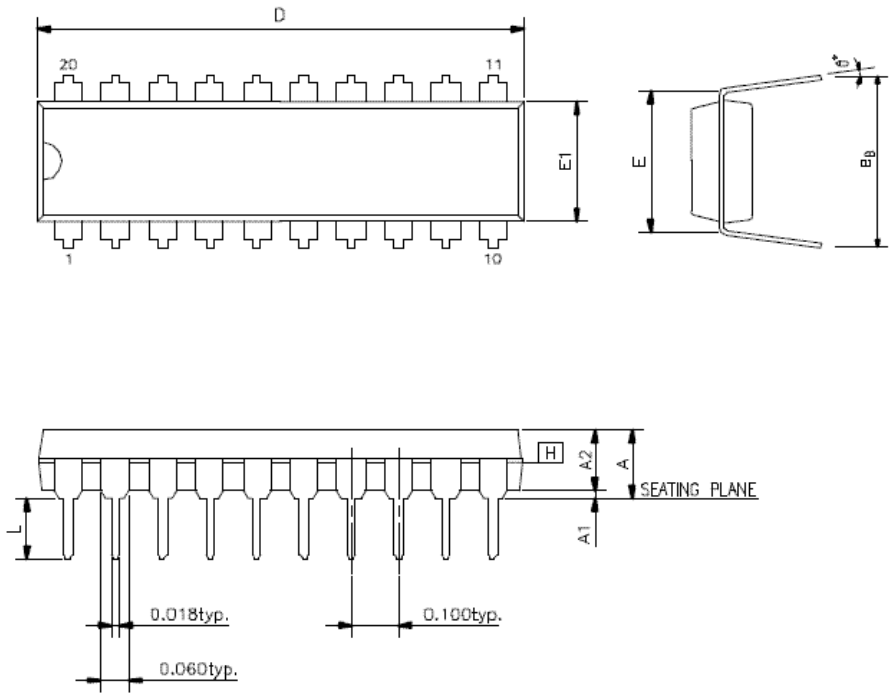



### 6.2.5 Low Voltage Detect (LVDT=3.7V) vs. Temperature



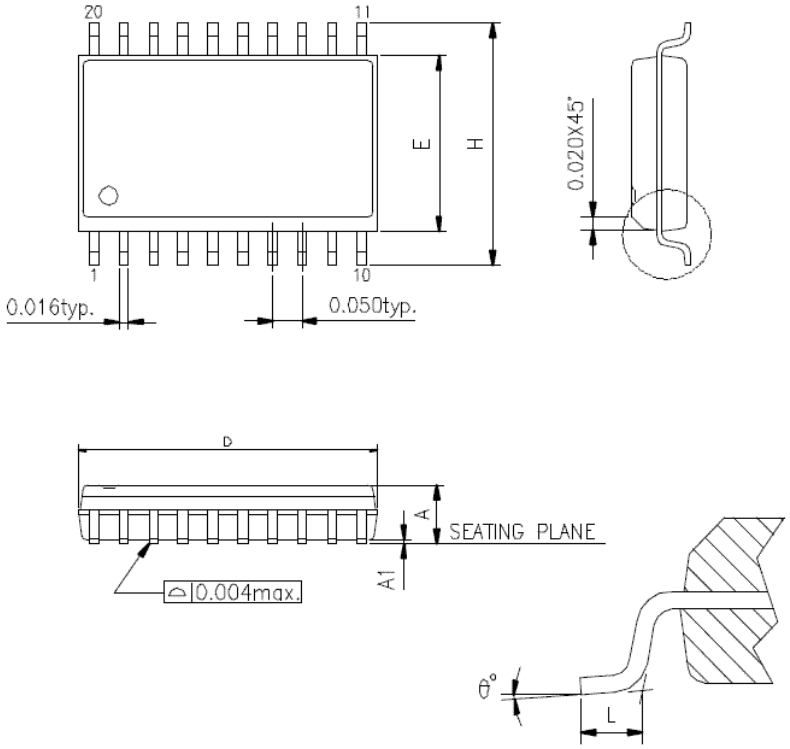
## 7.0 PACKAGE DIMENSION

### 7.1 20-PIN PDIP



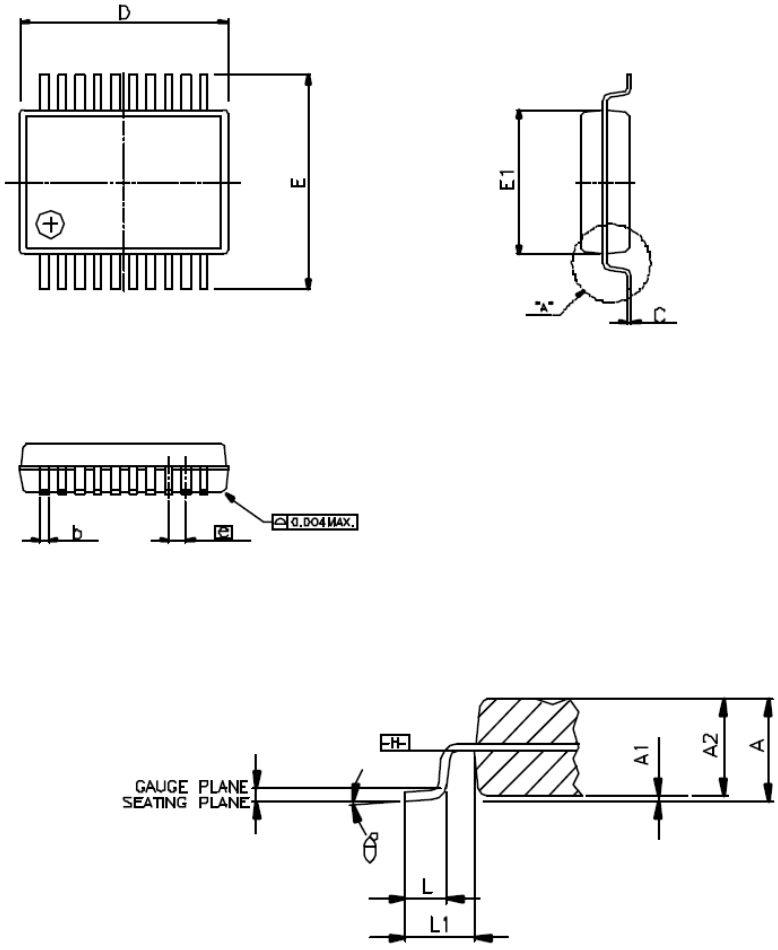

| Symbols | Dimension In Inches | | |
|---|---|---|---|
| | Min | Nom | Max |
| A | - | - | 0.210 |
| A1 | 0.015 | - | - |
| A2 | 0.125 | 0.130 | 0.135 |
| D | 0.98 | 1.030 | 1.060 |
| E | | 0.300 BSC | |
| E1 | 0.245 | 0.250 | 0.255 |
| L | 0.115 | 0.130 | 0.150 |
| eB | 0.335 | 0.355 | 0.375 |
| θ° | 0° | 7° | 15° |

### 7.2 20-PIN SOP



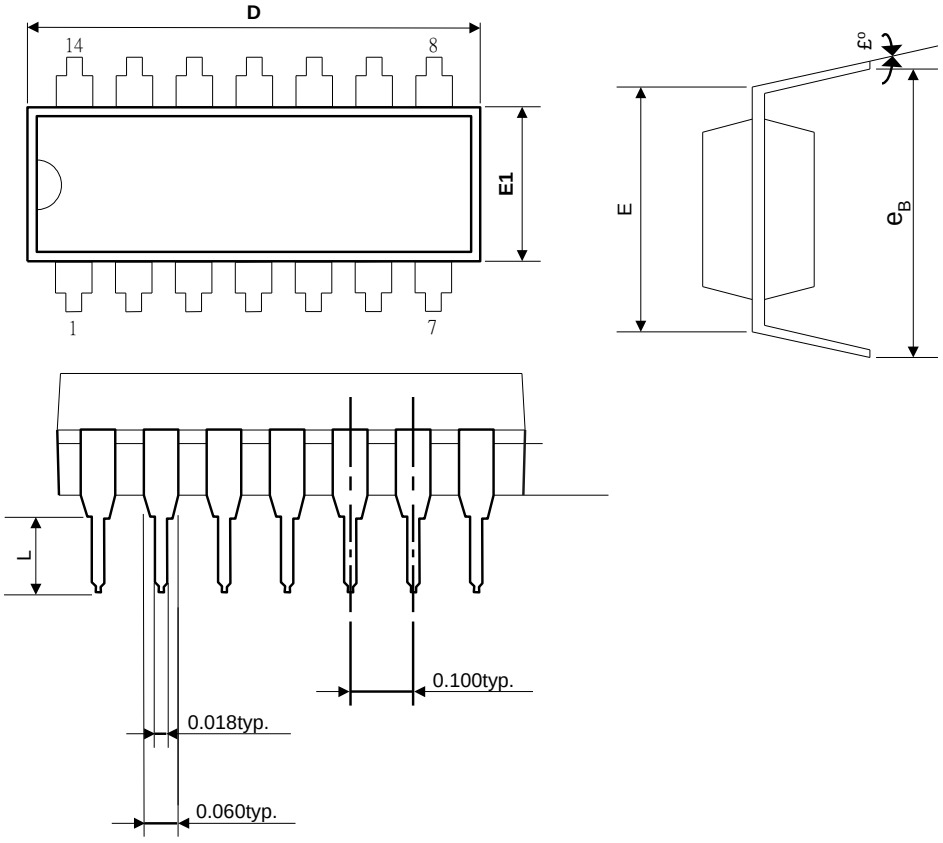

| Symbols | Dimension In Inches | | |
|---|---|---|---|
| | Min | Nom | Max |
| A | 0.093 | - | 0.104 |
| A1 | 0.004 | - | 0.012 |
| D | 0.496 | - | 0.508 |
| E | 0.291 | - | 0.299 |
| H | 0.394 | - | 0.419 |
| L | 0.016 | - | 0.050 |
| θ° | 0° | - | 8° |

### 7.3 20-PIN SSOP 209 mil



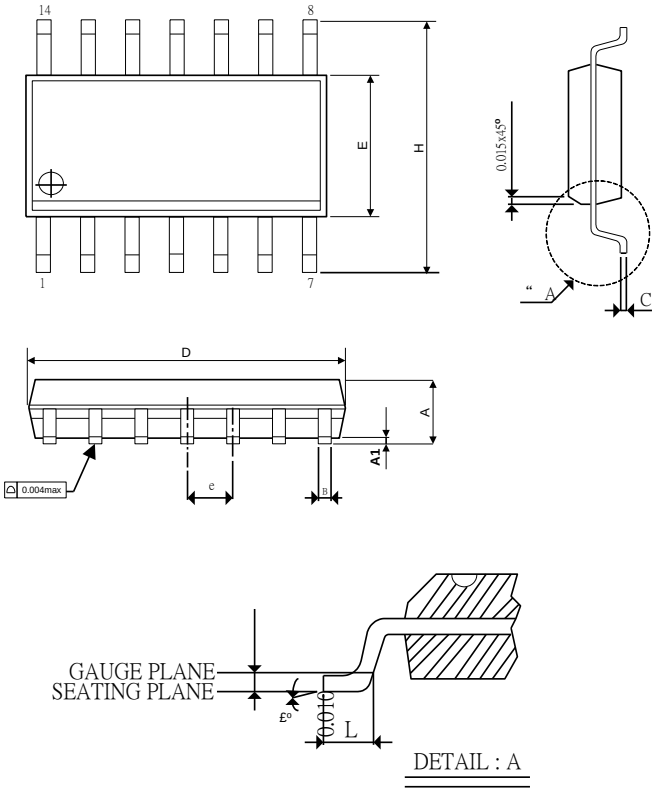

| Symbols | Dimension In Millimeters | | |
|---|---|---|---|
| | Min | Nom | Max |
| A | - | - | 2.00 |
| A1 | 0.05 | - | - |
| A2 | 1.65 | 1.75 | 1.85 |
| b | 0.22 | - | 0.38 |
| c | 0.09 | - | 0.21 |
| D | 6.90 | 7.20 | 7.50 |
| E | 7.40 | 7.80 | 8.20 |
| E1 | 5.00 | 5.30 | 5.60 |
| e | - | 0.65 | - |
| L | 0.55 | 0.75 | 0.95 |
| L1 | - | 1.25 | - |
| θ° | 0° | 4° | 8° |

### 7.4 14-PIN PDIP 300mil



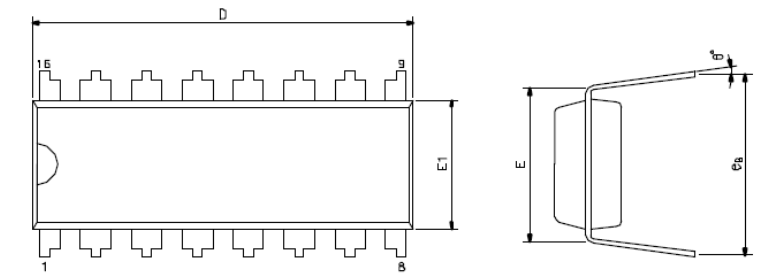

| Symbols | Dimension In Inches | | |
|---|---|---|---|
| | Min | Nom | Max |
| A | - | - | 0.210 |
| A1 | 0.015 | - | - |
| A2 | 0.125 | 0.130 | 0.135 |
| D | 0.735 | 0.750 | 0.775 |
| E | | 0.300 BSC. | |
| E1 | 0.245 | 0.250 | 0.255 |
| L | 0.115 | 0.130 | 0.150 |
| eB | 0.335 | 0.355 | 0.375 |
| θ° | 0° | 7° | 15° |

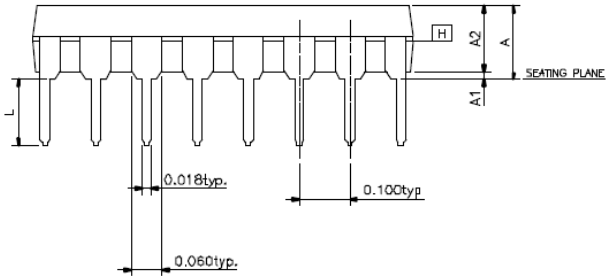
### 7.5 14-PIN SOP 150mil




| Symbols | Dimension In Inches | | |
|---|---|---|---|
| | Min | Nom | Max |
| A | 0.058 | 0.064 | 0.068 |
| A1 | 0.004 | - | 0.010 |
| B | 0.013 | 0.016 | 0.020 |
| C | 0.0075 | 0.008 | 0.0098 |
| D | 0.336 | 0.341 | 0.344 |
| E | 0.150 | 0.154 | 0.157 |
| e | - | 0.050 | - |
| H | 0.228 | 0.236 | 0.244 |
| L | 0.015 | 0.025 | 0.050 |
| θ° | 0° | - | 8° |

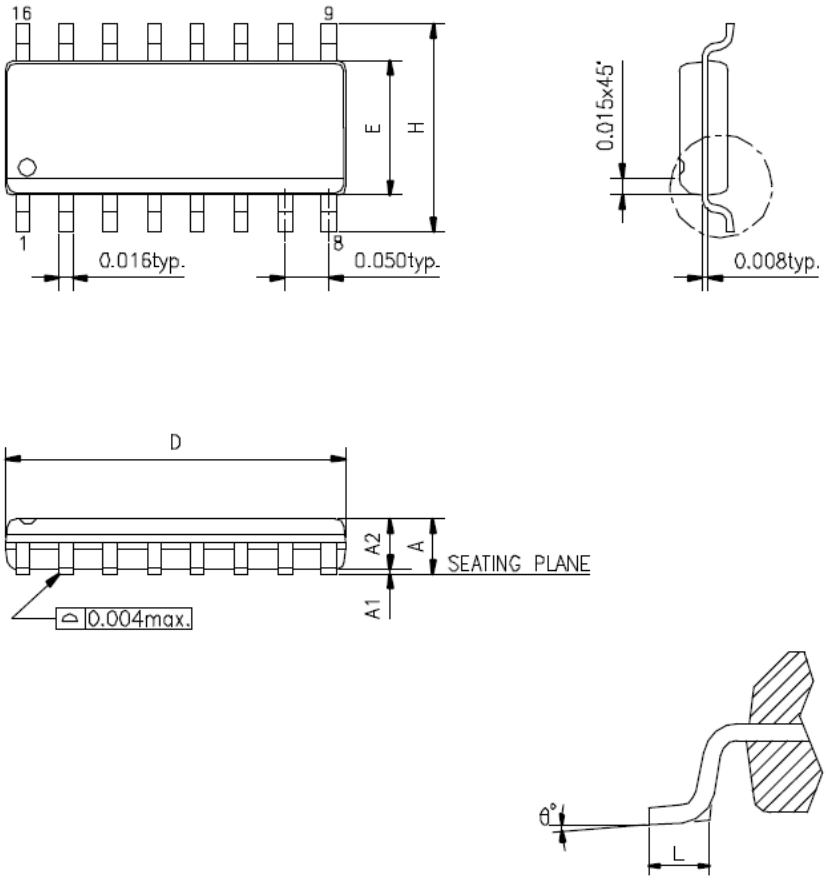
### 7.6 16-PIN PDIP 300mil



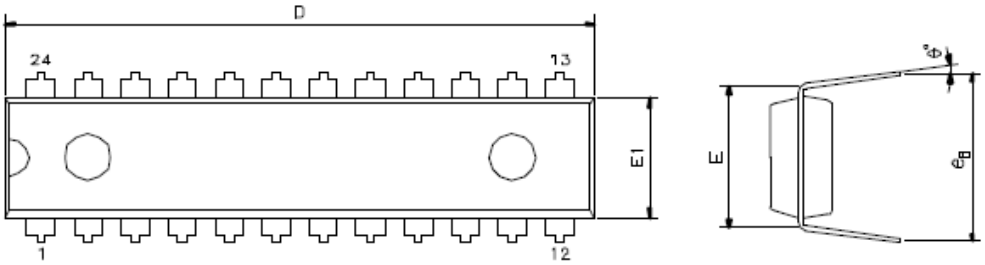

| Symbols | Dimension In Inches | | |
|---|---|---|---|
| | Min | Nom | Max |
| A | - | - | 0.210 |
| A1 | 0.015 | - | - |
| A2 | 0.125 | 0.130 | 0.135 |
| D | 0.735 | 0.755 | 0.775 |
| E | | 0.300 BSC | |
| E1 | 0.245 | 0.250 | 0.255 |
| L | 0.115 | 0.130 | 0.150 |
| eB | 0.335 | 0.355 | 0.375 |
| θ° | 0° | 7° | 15° |

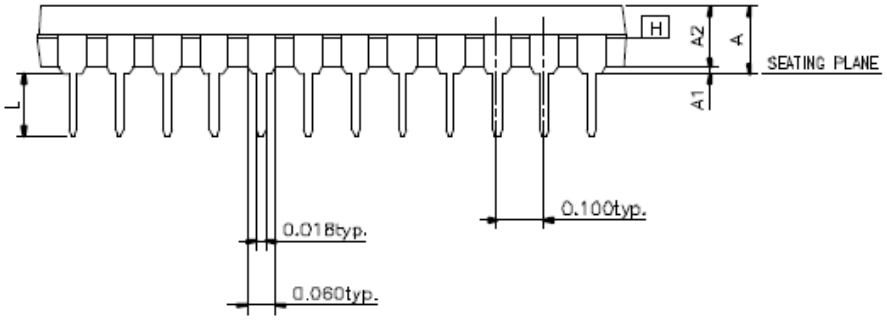
### 7.7 16-PIN SOP 150mil




| Symbols | Dimension In Inches | |
|---|---|---|
| | Min | Max |
| A | 0.053 | 0.069 |
| A1 | 0.004 | 0.010 |
| A2 | 0.049 | 0.065 |
| D | 0.386 | 0.394 |
| E | 0.150 | 0.157 |
| H | 0.228 | 0.244 |
| L | 0.016 | 0.050 |
| θ° | 0° | 8° |

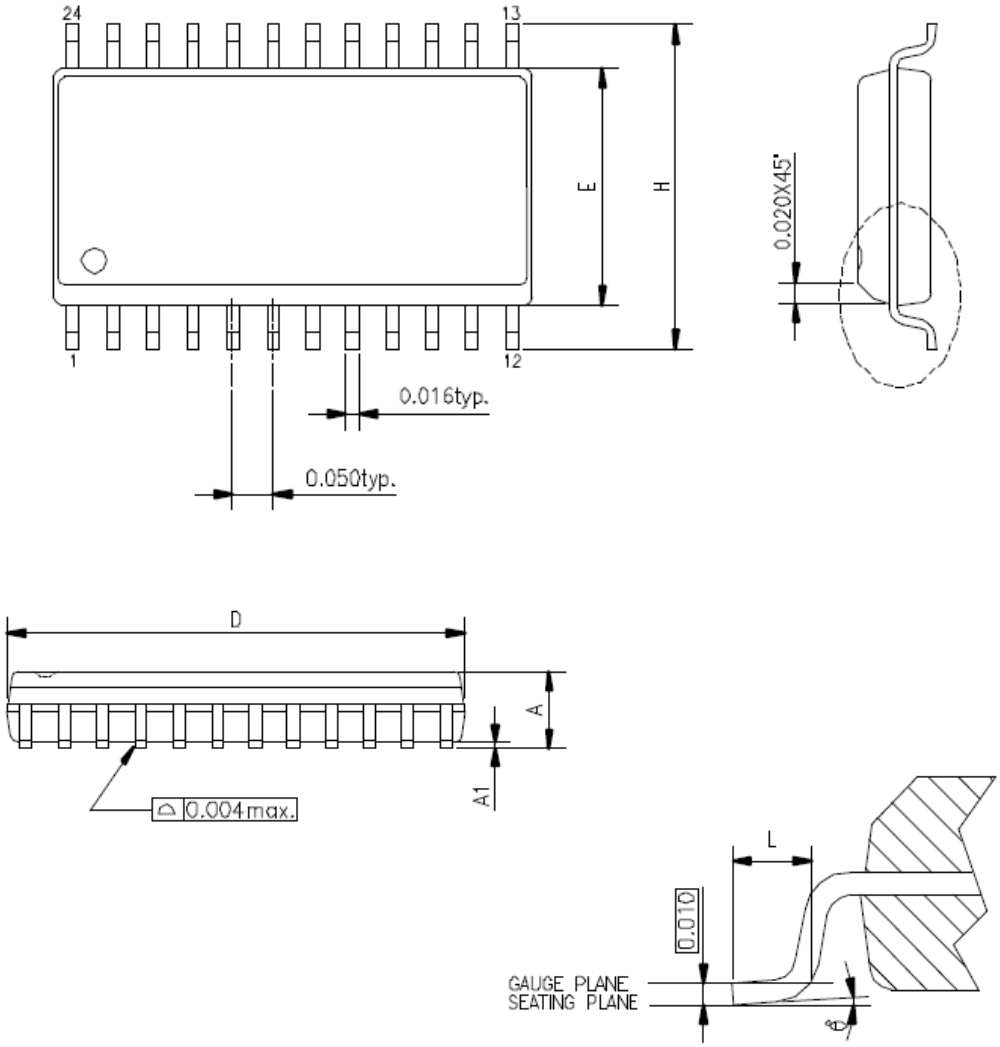
### 7.8 24-PIN PDIP 300mil (SKINNY)



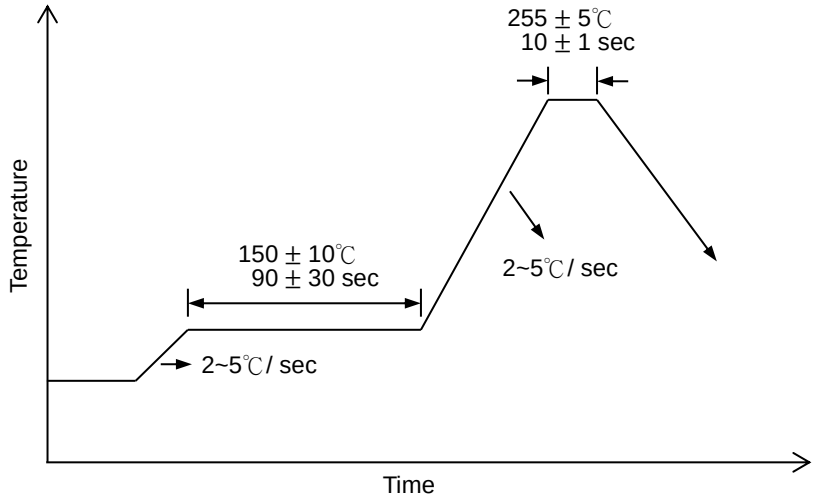

| Symbols | Dimension In Inches | | |
|---|---|---|---|
| | Min | Nom | Max |
| A | - | - | 0.210 |
| A1 | 0.015 | - | - |
| A2 | 0.125 | 0.130 | 0.135 |
| D | 1.230 | 1.250 | 1.280 |
| E | | 0.300 BSC. | |
| E1 | 0.253 | 0.258 | 0.263 |
| L | 0.115 | 0.130 | 0.150 |
| eB | 0.335 | 0.355 | 0.375 |
| θ° | 0° | 7° | 15° |

### 7.9 24-PIN SOP 300mil




| Symbols | Dimension In Inches | | |
|---|---|---|---|
| | Min | Nom | Max |
| A | - | - | 0.104 |
| A1 | 0.004 | - | - |
| D | 0.599 | 0.600 | 0.624 |
| E | 0.291 | 0.295 | 0.299 |
| H | 0.394 | 0.406 | 0.419 |
| L | 0.016 | 0.035 | 0.050 |
| θ° | 0° | 4° | 8° |

## 8.0 PACKAGE IR Re-flow Soldering Curve




## 9.0 ORDERING INFORMATION

| OTP Type MCU | Package Type | Pin Count | Package Size |
|---|---|---|---|
| FM8PA76AEP | PDIP | 20 | 300 mil |
| FM8PA76AED | SOP | 20 | 300 mil |
| FM8PA76AER | SSOP | 20 | 209 mil |
| FM8PA76BEP | PDIP | 14 | 300 mil |
| FM8PA76BED | SOP | 14 | 150 mil |
| FM8PA76DEP | PDIP | 16 | 300 mil |
| FM8PA76DED | SOP | 16 | 150 mil |
| FM8PA76EEP | PDIP | 24 | 300 mil |
| FM8PA76EED | SOP | 24 | 300 mil |
| FM8PA76FEP | PDIP | 16 | 300 mil |
| FM8PA76FED | SOP | 16 | 150 mil |