



# **TM57 Series**

## **Introduction to clear 8-Bit MCU interrupt flags**

### **Application Note**

**Tenx reserves the right to change or  
discontinue this product without notice.**

**tenx technology inc.**

**CONTENTS**

**PRODUCT NAME .....2**

**TM57series IC.....2**

**Introduction to clear 8-Bit MCU interrupt flags .....2**

**APPLICATION NOTE .....2**

**1. Use TM57PA40 as an example to demonstrate the details .....2**

**2. The detail execution process of the BCF instruction .....2**

**3. Appendix : Test circuit diagram .....7**

**PRODUCT NAME**

TM57series IC

**TITLE**

Introduction to clear 8-Bit MCU interrupt flags

**APPLICATION NOTE**

In TM57 series MCU interrupt function, if the program is not written properly, the problem that other interrupt requests in the same interrupt request register are cleared accidentally might arise.

**1. Use TM57PA40 as an example to demonstrate the details**

For example, both TIMER0 and TIMER1 interrupt are enabled; TIMER0 interrupt occur first. When the program is in the process of clearing TIMER0 interrupt request register, TIMER1 interrupt request occurs. And if it happens that BCF instruction is also used to clear TIMER0 interrupt request register at the same time, this interrupt request will be missed because TIMER1 interrupt request flag will be cleared accidentally.

**2. The detail execution process of the BCF instruction**

The execution process of BCF instruction is Read-Modify-Write. The above process will be simulated using instructions as follows:

TIMER0 interrupt occurs first, BCF INTF, 4 instruction is used in the interrupt service routine to clear TIMER0 interrupt request flag. The detail execution process of the BCF instruction is demonstrated as follows :

Movfw	INTF	;Read interrupt request register to ACC, ;ACC=00010000b
Andlw	11101111B,W	;Modify the 4 <sup>th</sup> bit of ACC register, ACC=00000000b ;At this moment, if the second interrupt occurs, ;assuming it is an TIMER1 interrupt, ;INTF=00110000b at this point of time.
Movwf	INTF	;Write the content of ACC (0) to the interrupt request ;register. ;and TIMER1 interrupt request flag is cleared ;accidentally.

**Because it is invalid to write 1 to an interrupt request register, it is recommended to use the following instruction to clear the interrupt request flag :**

Movlw	11101111B	;Clear bit4
Movwf	INTF	;Write to interrupt request register

The following DEMO program enables TIMER0/ TIMER1/INT0 /INT1/INT2 interrupt at the same time and will demonstrate how to clear interrupt request flags properly in the

```
;_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_
;_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_
#include<TM57PA40.INC>
;_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_
;_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_*_
;***** Program Start *****
;Config:
;LV=Enable
;OSC Mode=Internal RC(4MH)
;External pin reset=Disable
;WDTE=Disable


    org     000h
    goto    main_reset
    org     001h
    goto    interrupt
    org     002h
;-----
interrupt:
    movwf   ACCSave      ;save ACC in a backup register
    movfw   STAT         ;save Status register in a backup one
    movwf   StatusSave

Tm0:
    btfss   INTF,TM0I
    goto    Tm1
    movlw   11101111b
    movwf   INTF          ;Clear Timer0 interrupt pending Flag
    movlw   .130
    movwf   TIMERO        ;Reload Timer0 manually
    btfss   PDD,0         ;whenever the timer0 interrupt occurs,
    goto    $+3           ;Program will toggle the voltage on PortD.0
    bcf     PDD,0         ;Timer0 interrupt time interval will be 8ms @4MHz
    goto    Tm1
    bsf     PDD,0
;-----

Tm1:                                ;Timer1 will reload itself automatically when overflow
    btfss   INTF,TM1I
    goto    Wkt
    movlw   11011111b
```

```

movwf    INTF        ;Clear Timer1 interrupt pending Flag
btfss    PDD,1        ;whenever timer1 interrupt occurs,
goto     $+3          ;Program will toggle the voltage on PortD.1
bcf      PDD,1        ;Timer1 interrupt time interval will be 8ms @4MHz
goto     Wkt
bsf      PDD,1
;-----
Wkt:
btfss    INTF,WKTI
goto     Int0
movlw    11110111b
movwf    INTF        ;Clear WKT interrupt pending Flag
;-----
Int0:
btfss    INTF,XINT0
goto     Int1
movlw    11111110b
movwf    INTF        ;Clear INTO interrupt pending Flag
btfss    PDD,2        ;whenever INTO interrupt occurs,
goto     $+3          ;Program will toggle the voltage on PortD.2
bcf      PDD,2
goto     Int1
bsf      PDD,2
;-----
Int1:
btfss    INTF,XINT1
goto     Int2
movlw    11111101b
movwf    INTF        ;Clear INT1 interrupt pending Flag
btfss    PDD,3        ;whenever INT1 interrupt occurs,
goto     $+3          ;Program will toggle the voltage on PortD.3
bcf      PDD,3
goto     Int2
bsf      PDD,3
;-----
Int2:
btfss    INTF,XINT2
goto     exit_interrupt
movlw    11111011b
movwf    INTF        ;Clear INT2 interrupt pending Flag
btfss    PDD,4        ;whenever INT2 interrupt occurs,
goto     $+3          ;Program will toggle the voltage on PortD.4
bcf      PDD,4

```

```

goto    exit_interrupt
bsf     PDD,4
;-----
exit_interrupt:
movfw   StatusSave    ;Restore Status register
movwf   STAT
movfw   ACCSave       ;restore ACC
reti

;;=====
main_reset:
movlw   0x07
movwr   R_OPTION
        ;bit0~bit3  timer0 pre-scale (0000->1:1 / 0001->1:2
        ;0111->1:128 / 1111->1:256)
        ;bit4 timer0 psc clock  (0: instruction  1: t0i pin)
        ;bit5 timer0 t0i(pa2)edge(0:rising  1:failling)
        ;bit6~bit7  tcout pre scale
        ;(00->1:1 , 01->1:2 ,10->1:4 , 11->1:8)
;;=====
movlw   0xff          ;(0:input  1:output)
movwr   R_PAE         ;set pa as output port
movwf   PAD          ;write pa output data register as 0xff
movlw   0x00          ;(0:input  1:output)
movwr   R_PAE         ;set pa as input
;;=====
movlw   0x00          ;(0:input  1:output)
movwr   R_PBE         ;set pb as input
;;=====
Movlw   0xff          ;(0:input  1:output)
movwr   R_PDE         ;set pd as output
;;=====
movlw   0x00          ;(0:enable  1:disable)
movwr   R_PAPU        ;set pa pull up enable
;;=====
movlw   0xff          ;(0:enable  1:disable)
movwr   R_PBPU        ;set pb pull up disable
;;=====
movlw   0xff          ;(0:enable  1:disable)
movwr   R_PDPU        ;set pd pull up disable
;;=====
movlw   0x00
movwr   R_WKTPSC

```

```

;bit0~bit1 wdt/wkt period(00:13ms 01:25ms 10:50ms
;11:100ms)
;bit2 timer1 match out to pd0 (0:disable 1:enable)
;bit3 instruction out to pd6 (0:disable 1:enable)
;bit4 int1 edge (0:falling 1:rising)
;bit5 pwm1 output to pb0 (0:disable 1:enable)
;bit6 pwm0 output to pa0 (0:disable 1:enable)
;bit7 buzzer output pd1 (0:disable 1:enable)
;;=====
movlw 0x17
movwr R_BUZEN
;bit0~bit3 tmlpsc
;(0000->1:1 / 0001->1:2 0111->1:128 / 1111->1:256)
;bit4~bit6 adc clock(000->1:128 /001->1:64 111->1:1)
;bit7 buzzer(0:disable 1:enable)
;;=====
movlw 0x37
movwr R_IE
;interrupt (0:disable 1:enable)
;bit0 int0 interrupt
;bit1 int1 interrupt
;bit2 int2 interrupt
;bit3 wakeup timer interrupt
;bit4 timer0 interrupt
;bit5 timer1 interrupt
;bit6~7 not define
;;=====
movlw 0xff
movwr R_ADCIEn ;adc(0:adc input 1:digital input)
;;=====
; /*****F-REG INT*****/
;;=====
movlw .130
movwf TIMERO
movlw 0x00
movwf INTF ;Clear All the interrupt pending Flag
;;=====
movlw 0x1f
movwf PDD
;;=====
movlw .130
movwf TIMER1 ;set Timer1 and Timer1 reload register
movwr R_TM1RELOAD

```

```
nop
goto    $-1
end

;=====
; ;
```

3. Appendix : Test circuit diagram

