



TM57 系列

8-Bit MCU 的清中斷標誌說明

Application Note

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

tenx technology inc.

CONTENTS

PRODUCT NAME	2
TM57 系列IC	2
8-Bit MCU的清中斷標誌說明	2
APPLICATION NOTE	2
1. 以 TM57PA40 為例具體說明.....	2
2. BCF指令詳細執行過程.....	2
3. 附錄：測試線路圖	7

PRODUCT NAME

TM57 系列 IC

TITLE

8-Bit MCU 的清中斷標誌說明

APPLICATION NOTE

TM57 系列 MCU 中斷功能中，如果程式編寫不當，則有可能發生在同一個中斷申請寄存器裏的其他中斷請求被意外清除的問題。

1. 以 TM57PA40 為例具體說明

例如，TIMER0 和 TIMER1 的中斷都使能，TIMER0 首先發生中斷，當程式在清 TIMER0 中斷請求標誌寄存器時，恰好 TIMER1 也發生了中斷請求，而如果此時清 TIMER0 中斷標誌寄存器時又採用 BCF 指令，則會因為意外的清除掉 TIMER1 的中斷請求標誌而漏掉這個中斷請求。

2. BCF 指令詳細執行過程

BCF 指令執行的過程是 Read-Modify-Write，下面用具體指令來類比以上過程。

首先 TIMER0 發生中斷，中斷服務程式中使用 BCF INTF,4 指令清除 TIMER0 中斷請求標誌，BCF 指令詳細執行過程如下：

Movfw	INTF	;Read 中斷寄存器到 ACC 中，ACC=00010000b
Andlw	11101111B,W	;Modify ACC 寄存器的第四位，ACC=00000000b
		;如果這時發生了第二個中斷，假設為 TIMER1 中斷
		;則此時的 INTF=00110000b
Movwf	INTF	;Write ACC 的內容（0）到中斷請求寄存器，則意外
		;的把 TIMER1 的中斷請求標誌也清除掉了。

由於中斷請求寄存器寫入 1 無效，所以建議使用如下指令清中斷請求標誌：

Movlw	11101111B	;清 bit4
Movwf	INTF	;寫入中斷請求寄存器

下面的 DEMO 程式同時開啓了 TIMER0/ TIMER1/INT0 /INT1/INT2 中斷，並將在相應的中斷服務程式中演示如何正確的清除中斷請求標誌。

```

;*****
;*****
;*****
#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    1111011b
    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. 附錄：測試線路圖

