14
Interrupts

New instructions used in this chapter:

- RETFIE

We all know what interrupts are and we don't like being interrupted. We are busy doing something and the phone rings or someone arrives at the door.

If we are expecting someone, we could look out of the window every now and again to see if they had arrived or we could carry on with what we are doing until the doorbell rings. These are two ways of receiving an interrupt. The first when we keep checking in software terms is called polling, the second when the bell rings is equivalent to the hardware interrupt.

We have looked at polling when we used the keypad to see if any keys had been pressed. We will now look at the interrupt generated by the hardware.

Before moving onto an example of an interrupt consider the action of the door in a washing machine. The washing cycle does not start until the door is closed, but after that the door does not take any part in the program. But what if a child opens the door, water could spill out or worse!! We need to switch off the outputs if the door is opened. To keep looking at the door at frequent intervals in the program (software polling) would be very tedious indeed, so we use a hardware interrupt. We carry on with the program and ignore the door. But if the door is opened the interrupt switches off the outputs – spin motor etc. If the door had been opened accidentally then closing the door would return back to the program for the cycle to continue.

This suggests that when an interrupt occurs we need to remember what the contents of the files were. i.e. the STATUS register, W register, TMR0 and PORT settings so that when we return from the interrupt the settings are restored. If we did not remember the settings, we could not continue where we left off, because the interrupt switches off all the outputs and the W register would also be altered, at the very least.

Interrupt sources

The 16F84 has 4 interrupt sources.

- Change of rising or falling edge of PORTB,0.
- TMR0 overflowing from FFh to 00h.
- PORTB bits 4–7 changing.
- DATA EEPROM write complete.

The 16F818/9 has 9 interrupt sources, and of course need extra bits in the interrupt registers to handle them. The additional interrups used in the 16F818/9 are

- A/D conversion complete
- Synchronous Serial Port Interrupt
- TMR1 overflowing
- TMR2 overflowing
- Capture Compare Pulse Width Modulator Interrupt.

These interrupts can be enabled or disabled as required by their own interrupt enable/disable bits. These bits can be found in the interrupt control register INTCON for the 16F84 and also on the Peripheral Interrupt Enable Register1, PIE1 on the 16F818/9.

In this section we will be looking at the interrupt caused by a rising or falling edge on PORTB,0.

Interrupt control register

The Interrupt Control Register INTCON, file 0Bh is shown in Figure 14.1.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit0
GIE	EEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF

Figure 14.1 The interrupt control register, INTCON of 16F84

Bit 6 in this register is designated as the Peripheral Interrupt Enable Bit, PEIE for the 16F818/9.
Before any of the individual enable bits can be switched ON, the Global Interrupt Enable (GIE) bit 7 must be set, i.e. a 1 enables all unmasked interrupts and a 0 disables all interrupts.

Bit 6 EEIE (16F84) is an EEPROM data write complete interrupt enable bit, a 1 enables this interrupt and a 0 disables it.

Bit 6 PEIE (16F818/9) is the bit that permits enabling of the extra, peripheral bits.

Bit 5 T0IE is the TMR0 overflow interrupt enable bit, a 1 enables this interrupt and a 0 disables it.

Bit 4 INTE is the RB0/INT Interrupt Enable bit, a 1 enables this interrupt and a 0 disables it.

Bit 3 RBIE is the RB Port change (B4-B7) Interrupt enable bit, a 1 enables it and a 0 disables it.

Bit 2 T0IF is the flag, which indicates TMR0 has overflowed to generate the interrupt. 1 indicates TMR0 has overflowed, 0 indicates it hasn't. This bit must be cleared in software.

Bit 1 INTF is the RB0/INT Interrupt flag bit which indicates a change on PORTB,0. A 1indicates a change has occurred, a 0 indicates it hasn't.

Bit 0 RBIF is the RB PORT Change Interrupt flag bit. A 1 indicates that one of the inputs PORTB,4–7 has changed state. This bit must be cleared in software. A 0 indicates that none of the PORTB,4-7 bits have changed.

Program using an interrupt

As an example of how an interrupt works consider the following example:

Suppose we have 4 lights flashing consecutively for 5 seconds each. A switch connected to B0 acts as an interrupt so that when B0 is at a logic 0 an interrupt routine is called. This interrupt routine flashes all 4 lights ON and OFF twice at 1 second intervals and then returns back to the program providing the switch on B0 is at a logic1.

I have used the 16F818 for this application.

The circuit diagram for this application is shown in Figure 14.2.

One thing to note from the circuit the 16F818 chip has internal pull-up resistors on PORTB so B0 does not need a pull up resistor on the switch.

The interrupt we are using is a change on B0, we are therefore concerned with the following bits in the INTCON register, i.e. INTE bit4 the enable bit and INTF bit1 the flag showing B0 has changed, and of course GIE bit7 the Global Interrupt Enable Bit.

Program operation

When B0 generates an interrupt the program branches to the interrupt service routine. Where? Program memory location 4 tells the Microcontroller where to go to find the interrupt service routine.

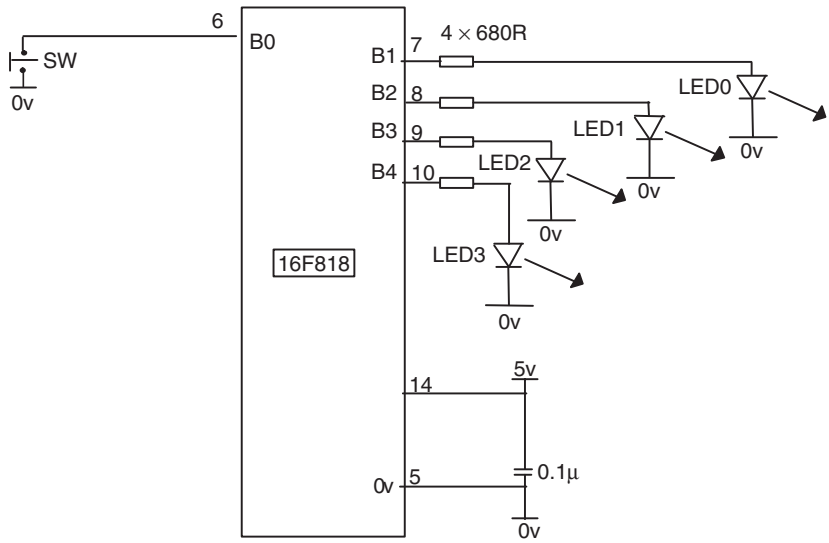

Figure 14.2 Interrupt demonstration circuit

Program memory location 4 is then programmed using the org statement as:

```
ORG     4       ;write next instruction in program memory location 4
GOTO    ISR     ;jump to the Interrupt Service Routine.
```

The interrupt service routine

The Interrupt Service Routine, ISR, is written like a subroutine and is shown below:

```
;Interrupt Service Routine

        MOVWF   W_TEMP      ;Save W
        SWAPF   STATUS,W
        MOVWF   STATUS_T    ;Save STATUS
        MOVF    TMR0,W
        MOVWF   TMR0_T      ;Save TMR0
        MOVF    PORTB,W
        MOVWF   PORTB_T     ;Save PORTB

        MOVLW   0FFH
        MOVWF   PORTB       ;turn on all outputs.
        CALL    DELAPY1     ;1 second delay
```

```
        MOVLW   0
        MOVWF   PORTB       ;turn off all outputs
        CALL    DELAPY1     ;1 second delay
        MOVLW   0FFH
        MOVWF   PORTB       ;turn on all outputs.
        CALL    DELAPY1     ;1 second delay
        MOVLW   0
        MOVWF   PORTB       ;turn off all outputs
        CALL    DELAPY1     ;1 second delay
SW_HI   BTFSS   PORTB,0
        GOTO    SW_HI       ;wait for switch to be HI.

        SWAPF   STATUS_T,W
        MOVWF   STATUS      ;Restore STATUS
        MOVF    TMR0_T,W
        MOVWF   TMR0        ;Restore TMR0
        MOVF    PORTB_T,W
        MOVWF   PORTB       ;Restore PORTB
        MOVF    W_TEMP,W    ;Restore W

        BCF     INTCON, INTF ;Reset Interrupt Flag
        RETFIE              ;Return from the interrupt
```

Operation of the interrupt service routine

The interrupt service routine operates in the following way.

- When an interrupt is made the Global Interrupt Enable is cleared automatically (disabled) to switch off all further interrupts. We would not wish to be interrupted while we are being interrupted.
- The registers W, STATUS, TMR0 and PORTB are saved in temporary locations W_TEMP, STATUS_T, TMR0_T and PORTB_T.
- The interrupt routine is executed, the lights flash on and off twice. This is a separate sequence than before to show the interrupt has interrupted the normal flow of the program. NB. The program has not been looking at the switch that generated the interrupt.
- We then wait until the switch returns HI.
- The temporary files W_TEMP, STATUS_T, TMR0_T and PORTB_T are restored back into W, STATUS, TMR0 and PORTB.
- The PORTB,0 interrupt flag INTCON,INTF is cleared ready to indicate further interrupts.
- We return from the interrupt, and the Global Interrupt Enable bit is automatically set to enable further interrupts.

Program of the interrupt demonstration

The complete code for this program is shown below as INTFLASH.ASM.

```
;INTFLASH.ASM Flashing lights being interrupted by a switch on B0.
;Using 16F818
;EQUATES SECTION

TMR0        EQU   1       ;means TMR0 is file 1.
STATUS      EQU   3       ;means STATUS is file 3.
PORTA       EQU   5       ;means PORTA is file 5.
PORTB       EQU   6       ;means PORTB is file 6.
TRISA       EQU   85H     ;TRISA (the PORTA I/O selection) is file 85H
TRISB       EQU   86H     ;TRISB (the PORTB I/O selection) is file 86H
INTCON      EQU   0BH     ;Interrupt Control Register
ZEROBIT     EQU   2       ;means ZEROBIT is bit 2.
CARRY       EQU   0       ;CARRY IS BIT 0.
GIE         EQU   7       ;Global Interrupt bit
INTE        EQU   4       ;B0 interrupt enable bit.
INTF        EQU   1       ;B0 interrupt flag
OPTION_R    EQU   81H
ADCON0      EQU   1FH     ;A/D Configuration reg.0
ADCON1      EQU   9FH     ;A/D Configuration reg.1
ADRES       EQU   1EH     ;A/D Result register.
OSCCON      EQU   8FH     ;Oscillator control register.
COUNT       EQU   20H     ;COUNT a register to count events.
                          ;a register to count events
TMR0_T      EQU   21H     ;TMR0 temporary file
W_TEMP      EQU   22H     ;W temporary file
STATUS_T    EQU   23H     ;STATUS temporary file
PORTB_T     EQU   24H     ;PORTB temporary file
COUNTA      EQU   25H
;*****************************************************

LIST        P=16F818              ;we are using the 16F818.
            ORG       0           ;the start address in memory is 0
            GOTO      START       ;goto start!
            ORG       4           ;write to memory location 4
            GOTO      ISR         ;location4 jumps to ISR
;*****************************************************

;Configuration Bits
__CONFIG H'3F10'          ;sets INTRC-A6 is port I/O, WDT off, PUT
```

```
                        ;on, MCLR tied to VDD A5 is I/O
                        ;BOD off, LVP disabled, EE protect disabled,
                        ;Flash Program Write disabled,
                        ;Background Debugger Mode disabled, CCP
                        ;function on B2,
                        ;Code Protection disabled.
;*********************************************************

;SUBROUTINE SECTION

;0.1 second delay, actually 0.099968s
DELAYP1 CLRF    TMR0            ;START TMR0.
LOOPB   MOVF    TMR0,W          ;READ TMR0 INTO W.
        SUBLW   .3              ;TIME-3
        BTFSS   STATUS,ZEROBIT  ;Check TIME-W = 0
        GOTO    LOOPB           ;Time is not = 3.
        NOP                     ;add extra delay
        NOP
        RETLW   0               ;Time is 3, return.

;5 second delay.
DELAY5  MOVLW   .50
        MOVWF   COUNTA
LOOPC   CALL    DELAYP1
        DECFSZ  COUNTA
        GOTO    LOOPC
        RETLW   0

;1 second delay.
DELAY1  MOVLW   .10
        MOVWF   COUNT
LOOPA   CALL    DELAYP1
        DECFSZ  COUNT
        GOTO    LOOPA
        RETLW   0

;Interrupt Service Routine.

ISR     MOVWF   W_TEMP          ;Save W
        SWAPF   STATUS,W
        MOVWF   STATUS_T        ;Save STATUS
        MOVF    TMR0,W
        MOVWF   TMR0_T          ;Save TMR0
        MOVF    PORTB,W
        MOVWF   PORTB_T         ;Save PORTB
```

```
                MOVLW       0FFH
                MOVWF       PORTB           ;turn on all outputs.
                CALL        DELAY1          ;1 second delay
                MOVLW       0
                MOVWF       PORTB           ;turn off all outputs
                CALL        DELAY1          ;1 second delay
                MOVLW       0FFH
                MOVWF       PORTB           ;turn on all outputs.
                CALL        DELAY1          ;1 second delay
                MOVLW       0
                MOVWF       PORTB           ;turn off all outputs
                CALL        DELAY1          ;1 second delay
SW_HI           BTFSS       PORTB,0
                GOTO        SW_HI           ;wait for switch to be HI.

                SWAPF       STATUS_T,W
                MOVWF       STATUS          ;Restore STATUS
                MOVF        TMR0_T,W
                MOVWF       TMR0            ;Restore TMR0
                MOVF        PORTB_T,W
                MOVWF       PORTB           ;Restore PORTB
                MOVF        W_TEMP,W        ;Restore W

                BCF         INTCON,INTF     ;Reset Interrupt Flag
                RETFIE                      ;Return from the interrupt

;*****************************************************
;CONFIGURATION SECTION

START           BSF         STATUS,5        ;Turns to Bank1.
                MOVLW       B'11111111'     ;8 bits of PORTA are I/P
                MOVWF       TRISA

                MOVLW       B'00000110'     ;PORTA IS DIGITAL
                MOVWF       ADCON1

                MOVLW       B'00000001'
                MOVWF       TRISB           ;PORTB,0 is I/P

                MOVLW       B'00000000'
                MOVWF       OSCCON          ;oscillator 31.25kHz

                MOVLW       B'00000111'     ;Prescaler is /256
                MOVWF       OPTION_R        ;TIMER is 1/32 secs.
```

```
        BCF     STATUS,5        ;Return to Bank0.
        CLRF    PORTA           ;Clears PortA.
        CLRF    PORTB           ;Clears PortB.
        BSF     INTCON,GIE      ;Enable Global Interrupt
        BSF     INTCON,INTE     ;Enable B0 interrupt

;*****************************************************
;Program starts now.

BEGIN   MOVLW   B'00000010'     ;Turn on B1
        MOVWF   PORTB
        CALL    DELAY5          ;wait 5 seconds
        MOVLW   B'00000100'     ;Turn on B2
        MOVWF   PORTB
        CALL    DELAY5          ;wait 5 seconds
        MOVLW   B'00001000'     ;Turn on B3
        MOVWF   PORTB
        CALL    DELAY5          ;wait 5 seconds
        MOVLW   B'00010000'     ;Turn on B4
        MOVWF   PORTB
        CALL    DELAY5          ;wait 5 seconds
        GOTO    BEGIN
END
```

The 4 lights are flashing on and off slowly enough (5 second intervals) so that you can interrupt part way through taking B0 low via the switch, (make sure B0 is hi when starting). The interrupt service routine then flashes all the lights on and off twice at 1 second intervals.

When returning from the interrupt with B0 hi again, the program resumes from where it left off, i.e. if the 2nd LED had been on for 3 seconds it would come back on for the remaining 2 seconds and the sequence would continue.