18
Projects

Project 1 Electronic dice

When using a Microcontroller in a control system the place to start is to decide what hardware you are controlling. In the Electronic Dice we will use 7 LEDs for the display and a push button to make the ‘‘throw’’. Just to make the dice a little more interesting we will use a buzzer to give an audible indication of the number thrown.

The circuit for the Dice is shown in Figure 18.1, using the 16F818 with its internal 31.25kHz clock. The push button is an input connected to PortA,2. The 7 LEDs are connected to PortB and the buzzer is on A1.

The truth table for the dice is shown in Table 18.1.

How does it work?

The dice has an input – the ‘‘throw’’ button. When it is pressed the internal count repeatedly runs through from 1 to 6 changing some 8000 times a second and stops on a number when the button is released.

This would be a complicated circuit to design with a timer, counter and decoder circuits. But now we can use one chip to do all the timing counting and decoding functions. Not only that I have also added a light flashing routine for the first few seconds when the dice is turned on. Try doing all that with one chip – other than a microcontroller.

The best way to describe the action of a program is with a flowchart. The flowchart for the dice is shown in Figure 18.2.

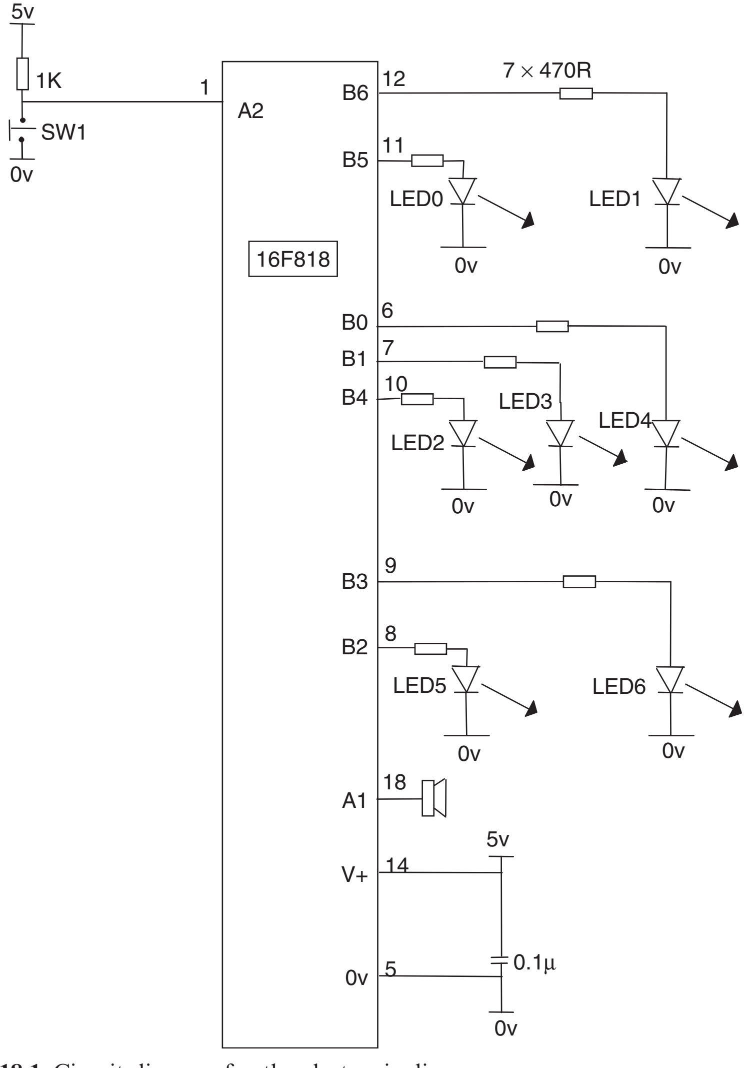

Figure 18.1 Circuit diagram for the electronic dice

Table 18.1 Truth table for the electronic dice

Throw	B7	B6	B5	B4	B3	B2	B1	B0
1	0	0	0	0	0	0	1	0
2	0	0	1	0	1	0	0	0
3	0	0	1	0	1	0	1	0
4	0	1	1	0	1	1	0	0
5	0	1	1	0	1	1	1	0
6	0	1	1	1	1	1	0	1

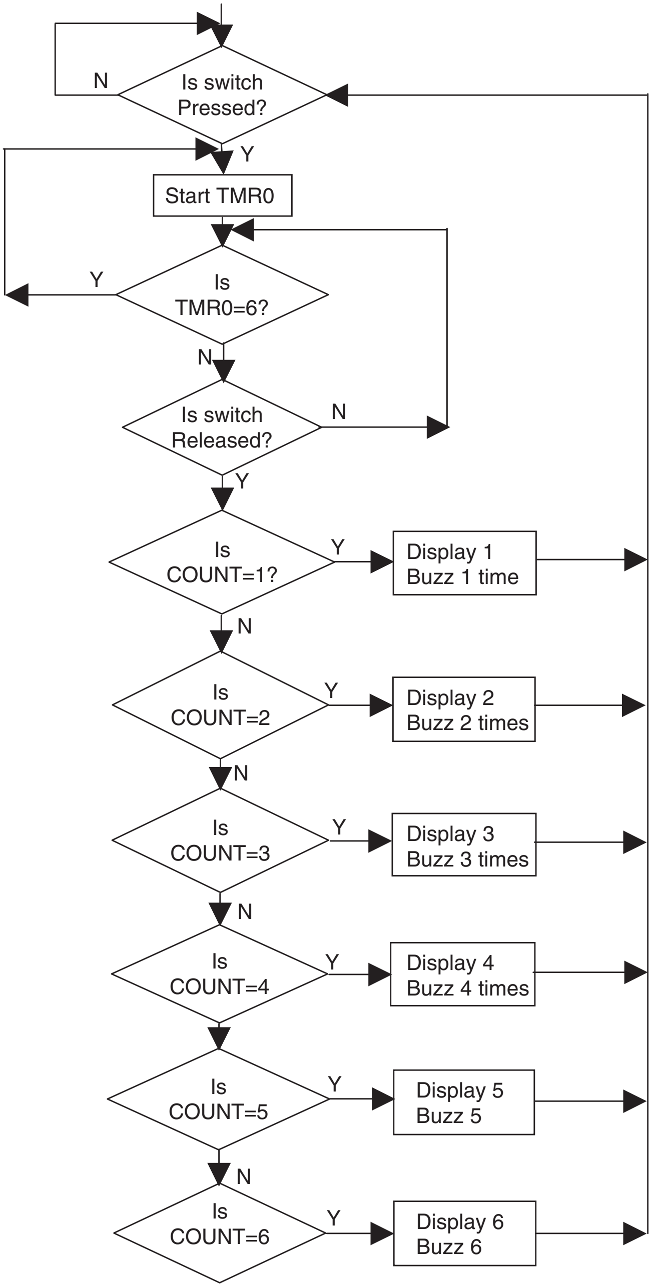

Figure 18.2 Flowchart for the dice

Program listing for the dice

The full program listing for the dice is given below in ;DICE.ASM.

```
;DICE.ASM

TMR0       EQU    1         ;means TMR0 is file 1.
PC         EQU    2
STATUS     EQU    3         ;means STATUS is file 3.
PORTA      EQU    5         ;means PORTA is file 5.
PORTB      EQU    6         ;means PORTB is file 6.
ZEROBIT    EQU    2         ;means ZEROBIT is bit 2.
ADCON0     EQU    1FH       ;A/D Configuration reg.0
ADCON1     EQU    9FH       ;A/D Configuration reg.1
ADRES      EQU    1EH       ;A/D Result register.
CARRY      EQU    0         ;CARRY IS BIT 0.
TRISA      EQU    85H       ;PORTA Configuration Register
TRISB      EQU    86H       ;PORTB Configuration Register
OPTION_R   EQU    81H       ;Option Register
OSCCON     EQU    8FH       ;Oscillator control register.
COUNT      EQU    20H       ;COUNT a register to count events.
COUNTA     EQU    21H

;**************************************************
           LIST   P=16F818  ;we are using the 16F818.
           ORG    0         ;the start address in memory is 0
           GOTO   START     ;goto start!

;**************************************************
;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.

;0.3 second delay.
DELAY         MOVLW      .3
              MOVWF      COUNT
LOOPC         CALL       DELAYP1
              DECFSZ     COUNT
              GOTO       LOOPC
              RETLW      0

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

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

START         BSF        STATUS,5      ;Turns to Bank1.

              MOVLW      B'11111101'   ;7 bits of PORTA are I/P
              MOVWF      TRISA

              MOVLW      B'00000110'   ;PORTA IS DIGITAL
              MOVWF      ADCON1

              MOVLW      B'00000000'
              MOVWF      TRISB         ;PORTB is OUTPUT

              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.

;*****************************************************
;Program starts now.
            CALL      DELAY1
            CALL      DELAY1
            CLRF      PORTB      ;Turn off LEDs and buzzer.
            MOVLW     .5
            MOVWF     COUNTA

SEC1        MOVLW     60H        ;Light flashing routine.
            MOVWF     PORTB
            CALL      DELAY
            MOVLW     13H
            MOVWF     PORTB
            CALL      DELAY
            MOVLW     0CH
            MOVWF     PORTB
            CALL      DELAY
            MOVLW     13H
            MOVWF     PORTB
            CALL      DELAY
            DECFSZ    COUNTA
            GOTO      SEC1

            CALL      DELAY1
            BSF       PORTA,1    ;Turn buzzer on
            CALL      DELAY1
            BCF       PORTA,1    ;Turn buzzer off

BEGIN       BTFSC     PORTA,2    ;Is switch pressed?
            GOTO      BEGIN      ;NO
            CALL      DELAYP1    ;YES
            CLRF      PORTB      ;Switch off LEDs
LOOP1       CLRF      TMR0       ;Start Timer
LOOP2       MOVF      TMR0,W     ;Put time into W.
            SUBLW      6         ;Is TMR0 = 6?
            BTFSC     STATUS,
                      ZEROBIT    ;Skip if TMR0 is not 6.
            GOTO      LOOP1      ;TMR0 is 6, so reset timer.
            BTFSS     PORTA,2    ;skip if button released?
            GOTO      LOOP2      ;No, Carry on timing
```

```
         MOVF     TMR0,W      ;yes, put the TMR0 into W.
         ADDWF    PC          ;Jump the value of W.
         GOTO     NUM1        ;TMR0=0
         GOTO     NUM2        ;TMR0=1
         GOTO     NUM3        ;TMR0=2
         GOTO     NUM4        ;TMR0=3
         GOTO     NUM5        ;TMR0=4
         GOTO     NUM6        ;TMR0=5

NUM1     MOVLW    B'00000010' ;Turn LED on
         MOVWF    PORTB
         BSF      PORTA,1     ;turn on buzzer for 1/4 sec.
         CALL     DELAY
         BCF      PORTA,1     ;Turn buzzer off.
         GOTO     BEGIN       ;BEGIN AGAIN.

NUM2     MOVLW    B'00101000' ;TURN ON 2 LEDS.
         MOVWF    PORTB
         BSF      PORTA,1     ;turn on buzzer for 1/4 sec.
         CALL     DELAY
         BCF      PORTA,1     ;turn off buzzer for 1/4 sec.
         CALL     DELAY
         BSF      PORTA,1     ;turn on buzzer for 1/4 sec.
         CALL     DELAY
         BCF      PORTA,1     ;Turn buzzer off.
         GOTO     BEGIN

NUM3     MOVLW    B'00101010'
         MOVWF    PORTB
         BSF      PORTA,1     ;turn on buzzer for 1/4 sec.
         CALL     DELAY
         BCF      PORTA,1     ;turn off buzzer for 1/4 sec.
         CALL     DELAY
         BSF      PORTA,1     ;turn on buzzer for 1/4 sec.
         CALL     DELAY
         BCF      PORTA,1     ;turn off buzzer for 1/4 sec.
         CALL     DELAY
         BSF      PORTA,1     ;turn on buzzer for 1/4 sec.
         CALL     DELAY
         BCF      PORTA,1     ;Turn off buzzer.
         GOTO     BEGIN

NUM4     MOVLW    B'01101100'
         MOVWF    PORTB
```

```
           BSF      PORTA,1  ;turn on buzzer for 1/4 sec.
           CALL     DELAY
           BCF      PORTA,1  ;turn off buzzer for 1/4 sec.
           CALL     DELAY
           BSF      PORTA,1  ;turn on buzzer for 1/4 sec.
           CALL     DELAY
           BCF      PORTA,1  ;turn off buzzer for 1/4 sec.
           CALL     DELAY
           BSF      PORTA,1  ;turn on buzzer for 1/4 sec.
           CALL     DELAY
           BCF      PORTA,1  ;turn off buzzer for 1/4 sec.
           CALL     DELAY
           BSF      PORTA,1  ;turn on buzzer for 1/4 sec.
           CALL     DELAY
           BCF      PORTA,1  ;Turn buzzer off.

           GOTO     BEGIN

NUM5       MOVLW    B'01101110'
           MOVWF    PORTB
           BSF      PORTA,1  ;turn on buzzer for 1/4 sec.
           CALL     DELAY
           BCF      PORTA,1  ;turn off buzzer for 1/4 sec.
           CALL     DELAY
           BSF      PORTA,1  ;turn on buzzer for 1/4 sec.
           CALL     DELAY
           BCF      PORTA,1  ;turn off buzzer for 1/4 sec.
           CALL     DELAY
           BSF      PORTA,1  ;turn on buzzer for 1/4 sec.
           CALL     DELAY
           BCF      PORTA,1  ;turn off buzzer for 1/4 sec.
           CALL     DELAY
           BSF      PORTA,1  ;turn on buzzer for 1/4 sec.
           CALL     DELAY
           BCF      PORTA,1  ;turn off buzzer for 1/4 sec.
           CALL     DELAY
           BSF      PORTA,1  ;turn on buzzer for 1/4 sec.
           CALL     DELAY
           BCF      PORTA,1  ;turn off buzzer.
           GOTO     BEGIN

NUM6       MOVLW    B'01111101'
           MOVWF    PORTB
           BSF      PORTA,1  ;turn on buzzer for 1/4 sec.
           CALL     DELAY
```

```
          BCF      PORTA,1   ;turn off buzzer for 1/4 sec.
          CALL     DELAY
          BSF      PORTA,1   ;turn on buzzer for 1/4 sec.
          CALL     DELAY
          BCF      PORTA,1   ;turn off buzzer for 1/4 sec.
          CALL     DELAY
          BSF      PORTA,1   ;turn on buzzer for 1/4 sec.
          CALL     DELAY
          BCF      PORTA,1   ;turn off buzzer for 1/4 sec.
          CALL     DELAY
          BSF      PORTA,1   ;turn on buzzer for 1/4 sec.
          CALL     DELAY
          BCF      PORTA,1   ;turn off buzzer for 1/4 sec.
          CALL     DELAY
          BSF      PORTA,1   ;turn on buzzer for 1/4 sec.
          CALL     DELAY
          BCF      PORTA,1   ;turn off buzzer for 1/4 sec.
          CALL     DELAY
          BSF      PORTA,1   ;turn on buzzer for 1/4 sec.
          CALL     DELAY
          BCF      PORTA,1   ;Turn buzzer off.
          GOTO     BEGIN
END
```

Modifications to the dice project

Can you think of any modifications you can make to this program? Perhaps you could add a roll routine so that a few numbers are shown before the dice finally comes to rest on the number.

The initial display routine could also be customized.

You could throw a 7.

Dice using 12C508

The dice circuit used 8 outputs and 1 input a total of 9 I/O.

But LEDs 0 and 6, 1 and 5, 2 and 4 work in pairs, i.e. they are on and off together. If these LEDs were paralleled up, then we only need 6 I/O, e.g.:

- Input from Switch
- Output to Buzzer

- Output to LEDs 0 and 6
- Output to LEDs 1 and 5
- Output to LEDs 2 and 4
- Output to LED 3

This project can then be undertaken using the 6 I/O of the 12C508.

Project 2 Reaction timer

There are many question and answer games on the market that would benefit from a reaction timer which indicates the first player of a team to press. This project has the facility for up to 6 players.

The circuit diagram for this project illustrated in Figure 18.3 uses 6 inputs and 7 outputs.

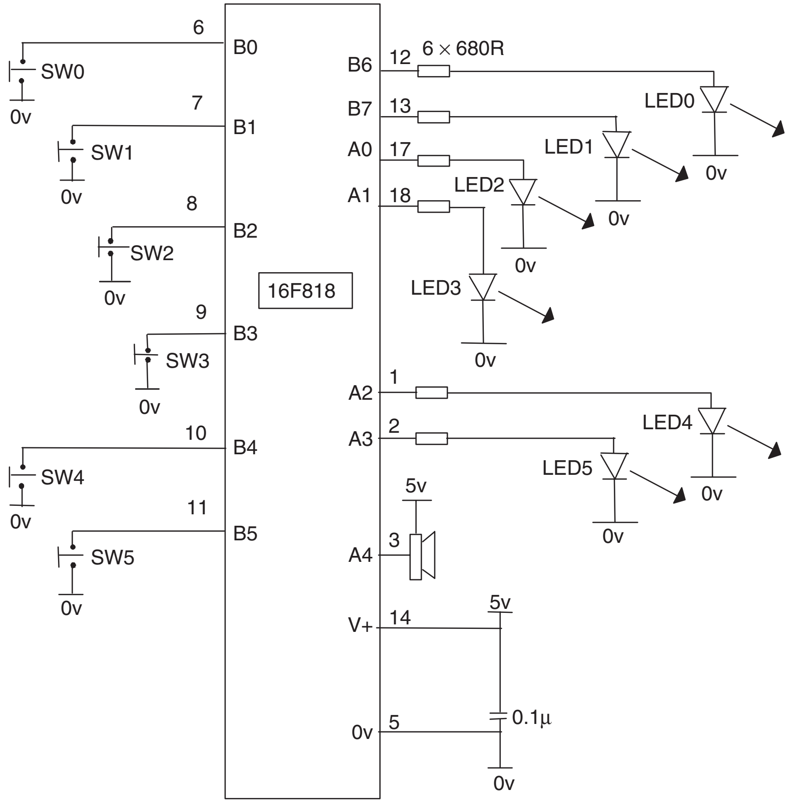

Figure 18.3 The reaction timer circuit

Reaction timer operation

If B0 is the first to press B6 output LED lights
If B1 is the first to press B7 output LED lights
If B2 is the first to press A0 output LED lights
If B3 is the first to press A1 output LED lights
If B4 is the first to press A2 output LED lights
If B5 is the first to press A3 output LED lights
The Buzzer is connected to A4.

The buzzer sounds for 4 seconds after a button is pressed. During this time no further presses are acknowledged. After the 4 seconds the buzzer stops and the LED is extinguished and the program resets.

The unit uses 13 I/O but not all 6 button/LED combinations need be used. The program will not need altering.

Just one point in case you were wondering: B0–B5 have been used as inputs instead of PORTA because PORTB has internal pull-up resistors on the inputs. The switches do not need their own – no point in using 5 resistors if you don't have to.

The reaction timer program

```
;REACTION.ASM

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.
ZEROBIT    EQU   2     ;means ZEROBIT is bit 2.
ADCON0     EQU   1FH   ;A/D Configuration reg.0
ADCON1     EQU   9FH   ;A/D Configuration reg.1
ADRES      EQU   1EH   ;A/D Result register.
CARRY      EQU   0     ;CARRY IS BIT 0.
TRISA      EQU   85H   ;PORTA Configuration Register
TRISB      EQU   86H   ;PORTB Configuration Register
OPTION_R   EQU   81H   ;Option Register
OSCCON     EQU   8FH   ;Oscillator control register.
COUNT      EQU   20H   ;COUNT a register to count events.

;*********************************************************
```

```
           LIST       P=16F818     ;we are using the 16F818.
           ORG        0            ;the start address in memory is 0
           GOTO       START        ;goto start!

;*******************************************************
;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.

;4 second delay.
DELAY4     MOVLW      .40
           MOVWF      COUNT
LOOPC      CALL       DELAYP1
           DECFSZ     COUNT
           GOTO       LOOPC
           RETLW      0

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

```
ON0         BSF         PORTB,6     ;Turn on LED0
            BSF         PORTA,4     ;Turn on buzzer
            CALL        DELAY4      ;Wait 4 seconds
            BCF         PORTB,6     ;Turn off LED0
            BCF         PORTA,4     ;Turn off buzzer
            GOTO        SCAN

ON1         BSF         PORTB,7     ;Turn on LED1
            BSF         PORTA,4     ;Turn on buzzer
            CALL        DELAY4      ;Wait 4 seconds
            BCF         PORTB,7     ;Turn off LED1
            BCF         PORTA,4     ;Turn off buzzer
            GOTO        SCAN

ON2         BSF         PORTA,0     ;Turn on LED2
            BSF         PORTA,4     ;Turn on buzzer
            CALL        DELAY4      ;Wait 4 seconds
            BCF         PORTA,0     ;Turn off LED2
            BCF         PORTA,4     ;Turn off buzzer
            GOTO        SCAN

ON3         BSF         PORTA,1     ;Turn on LED3
            BSF         PORTA,4     ;Turn on buzzer
            CALL        DELAY4      ;Wait 4 seconds
            BCF         PORTA,1     ;Turn off LED3
            BCF         PORTA,4     ;Turn off buzzer
            GOTO        SCAN

ON4         BSF         PORTA,2     ;Turn on LED4
            BSF         PORTA,4     ;Turn on buzzer
            CALL        DELAY4      ;Wait 4 seconds
            BCF         PORTA,2     ;Turn off LED4
            BCF         PORTA,4     ;Turn off buzzer
            GOTO        SCAN

ON5         BSF         PORTA,3     ;Turn on LED5
            BSF         PORTA,4     ;Turn on buzzer
            CALL        DELAY4      ;Wait 4 seconds
            BCF         PORTA,3     ;Turn off LED5
            BCF         PORTA,4     ;Turn off buzzer
            GOTO        SCAN

;**********************************************************
```

```
;CONFIGURATION SECTION.

START     BSF       STATUS,5    ;Turns to Bank1.

          MOVLW     B’0000000’  ;8 bits of PORTA are O/P
          MOVWF     TRISA

          MOVLW     B’00000110’ ;PORTA IS DIGITAL
          MOVWF     ADCON1

          MOVLW     B’00111111’
          MOVWF     TRISB       ;PORTB is mixed I/O

          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.
          CLRF      COUNT

;***************************************************

;Program starts now.
          MOVLW     0FFH
          MOVWF     PORTA       ;Turn on PORTA outputs
          BSF       PORTA,4     ;Turn on buzzer
          MOVWF     PORTB       ;Turn on PORTB outputs
          CALL      DELAY1      ;Wait 1 second
          CLRF      PORTA       ;Turn off PORTA outputs
          BCF       PORTA,4     ;Turn off buzzer
          CLRF      PORTB       ;Turn off PORTB outputs

SCAN      BTFSS     PORTB,0     ;Has B0 been pressed
          GOTO      ON0         ;Yes
          BTFSS     PORTB,1     ;Has B1 been pressed
```

```
            GOTO     ON1        ;Yes
            BTFSS    PORTB,2    ;Has B2 been pressed
            GOTO     ON2        ;Yes
            BTFSS    PORTB,3    ;Has B3 been pressed
            GOTO     ON3        ;Yes
            BTFSS    PORTB,4    ;Has B4 been pressed
            GOTO     ON4        ;Yes
            BTFSS    PORTB,5    ;Has B5 been pressed
            GOTO     ON5        ;Yes
            GOTO     SCAN
END
```

How does it work?

The program starts by turning all the LEDs and the buzzer on for 1 second to check they are all working.

The program then tests each input in turn starting with B0, if it is set i.e. not pressed the program skips and checks the next input. When the last input B5 is checked and it is not pressed then the program skips the next instruction and goes back to SCAN again.

If one of the inputs is pressed the program branches to the relevant subroutine to turn on the appropriate LED and buzzer for 4 seconds before returning to scan the switches again.

Reaction timer development

One way of making this program more interesting and to develop your programming skills – when a button is pressed have the outputs jump around B6, A0, A3, A1, A2 then B7 before landing on the correct output.

You could also have a flashing light routine at the start of the program to check they are working, you could also pulse the buzzer. The buzzer could be made to beep a number of times to give an audible indication of who was first to press. Another modification you could make is – think of one yourself, I'm not doing all the work.

Project 3 Burglar alarm

Operation

The circuit for the Burglar Alarm is shown in Figure 18.4 using the 16F818.

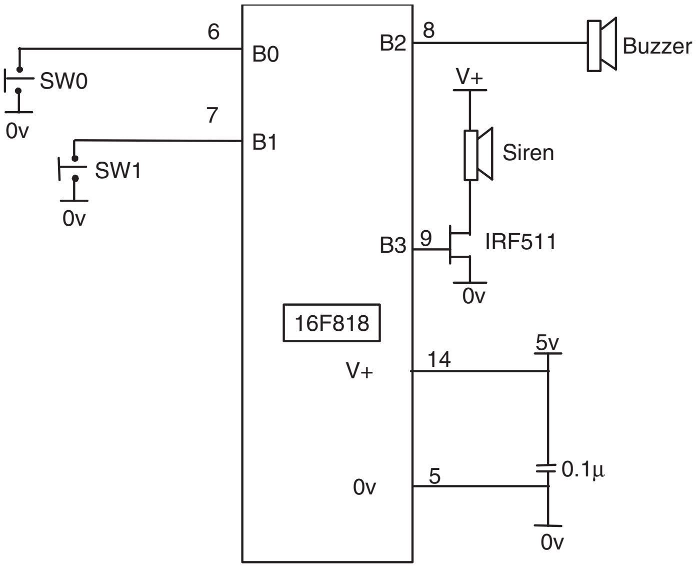

Figure 18.4 Burglar alarm circuit

It uses two inputs, SW0 and SW1 which are both normally closed. They can represent Door contacts, Passive Infra red sensor outputs, window contacts or tilt switches.

SW0 has a delay on it but SW1 is immediately active.

Both switches can have additional switches wired in series with them to provide extra security cover. If SW1 is a window contact in a caravan it could have a tilt switch wired in series with it, so if the caravan was moved the siren would sound immediately.

SW0 and SW1 are connected to PORTB so pull-ups are not required.

A buzzer is used to indicate entry and exit delays on the alarm and a siren is connected to the micro via an IRF511 (Power MOSFET).

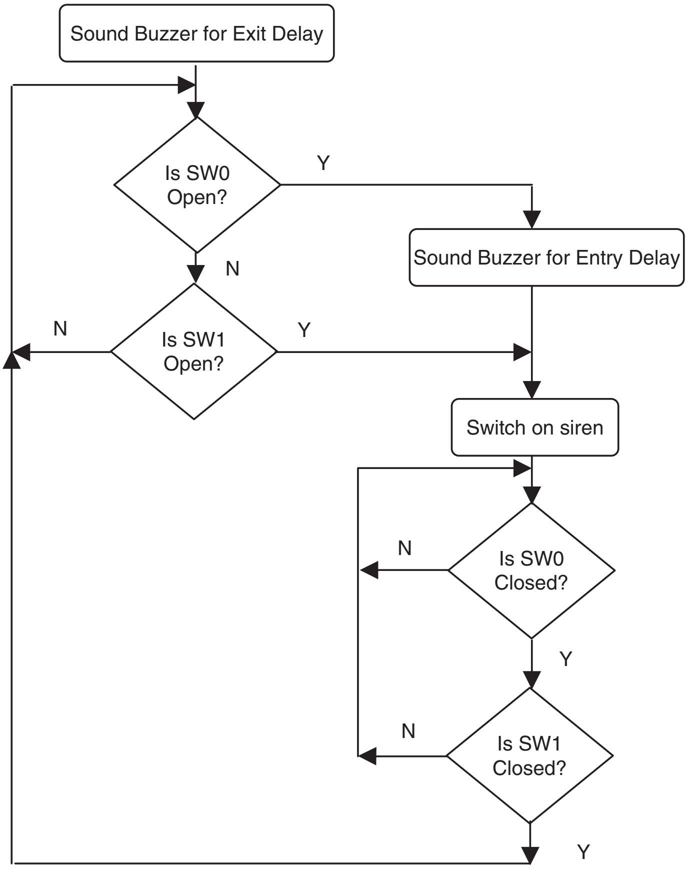

Figure 18.5 Burglar alarm flowchart

How does it work?

Consider the flow chart in Figure 18.5.

With reference to the flow chart:

When the alarm is switched on a 30 second exit delay is activated and the buzzer sounds for this time.

Switches 0 and 1 are continually checked until one of them is open.

If SW0 is opened a 30 second entry delay is activated and the buzzer sounds for this time, the siren will then sound for 5 minutes.

If SW1 is opened the siren will sound immediately for 5 minutes.

The switches are then checked until they are both closed when the alarm resets back to checking switches 0 and 1 until one of them opens again.

Switching off the power would disable the alarm.

Burglar alarm project code

The code for the Burglar Alarm is shown below in ALARM.ASM

```
;ALARM.ASM

;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.
ZEROBIT     EQU   2        ;means ZEROBIT is bit 2.
ADCON0      EQU   1FH      ;A/D Configuration reg.0
ADCON1      EQU   9FH      ;A/D Configuration reg.1
ADRES       EQU   1EH      ;A/D Result register.
CARRY       EQU   0        ;CARRY IS BIT 0.
TRISA       EQU   85H      ;PORTA Configuration Register
TRISB       EQU   86H      ;PORTB Configuration Register
OPTION_R    EQU   81H      ;Option Register
OSCCON      EQU   8FH      ;Oscillator control register.
COUNT       EQU   20H      ;COUNT a register to count events.
COUNTA      EQU   21H

;*****************************************************
            LIST       P=16F818    ;we are using the 16F818.
            ORG         0          ;the start address in memory is 0
            GOTO       START       ;goto start!

;*****************************************************
;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.

;0.5 second delay.
DELAYP5   MOVLW    .5
          MOVWF    COUNT
LOOPC     CALL     DELAYP1
          DECFSZ   COUNT
          GOTO     LOOPC
          RETLW    0

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

;0.25 second delay
DELAYP25  MOVLW    .3
          MOVWF    COUNT
LOOPD     CALL     DELAYP1
          DECFSZ   COUNT
          GOTO     LOOPD
          RETLW    0

;5 second delay
DELAY5    MOVLW    .50
          MOVWF    COUNT
LOOPE     CALL     DELAYP1
          DECFSZ   COUNT
          GOTO     LOOPE
          RETLW    0
```

```
BUZZER    MOVLW     .5
          MOVWF     COUNTA     ;5 × 2 SECONDS
BUZZ1     BSF       PORTB,2
          CALL      DELAY1
          BCF       PORTB,2
          CALL      DELAY1
          DECFSZ    COUNTA
          GOTO      BUZZ1
          MOVLW     .10
          MOVWF     COUNTA     ;10 × 1 SECOND
BUZZ2     BSF       PORTB,2
          CALL      DELAYP5
          BCF       PORTB,2
          CALL      DELAYP5
          DECFSZ    COUNTA
          GOTO      BUZZ2
          MOVLW     .20
          MOVWF     COUNTA
BUZZ3     BSF       PORTB,2    ;20 × 0.5 SECONDS
          CALL      DELAYP25
          BCF       PORTB,2
          CALL      DELAYP25
          DECFSZ    COUNTA
          GOTO      BUZZ3
          RETLW     0
;*********************************************************
;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'00000011'
          MOVWF     TRISB      ;PORTB is MIXED I/O

          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.
            CLRF      COUNT

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

            CALL      BUZZER     ;Exit delay
CHK_ON      BTFSC     PORTB,0    ;Check for alarm
            GOTO      ENTRY
            BTFSC     PORTB,1
            GOTO      SIREN
            GOTO      CHK_ON

ENTRY       CALL      BUZZER     ;Entry delay
SIREN       BSF       PORTB,3    ;5 minute siren
            MOVLW     .60
            MOVWF     COUNTA
WAIT5       CALL      DELAY5
            DECFSZ    COUNTA
            GOTO      WAIT5

            BCF       PORTB,3    ;Turn off Siren
CHK_OFF     BTFSC     PORTB,0    ;Check switches closed
            GOTO      CHK_OFF
            BTFSC     PORTB,1
            GOTO      CHK_OFF

            CALL      DELAYP25   ;antibounce
            GOTO      CHK_ON
END
```

The Burglar Alarm uses 2 inputs and 2 outputs a total of 4 I/O.

We can therefore program the Alarm with a 12C508 chip.

Burglar alarm using the 12C508

The circuit diagram for the Alarm with the 12C508 is shown in Figure 18.6.

Note in the circuit of Figure 18.6, showing the alarm using the 12C508, that no external oscillator circuit is required and that pull ups are not required on pins GPIO,0 or GPIO,1 (or GPIO,3). N.B. GPIO,3 is an input only pin.

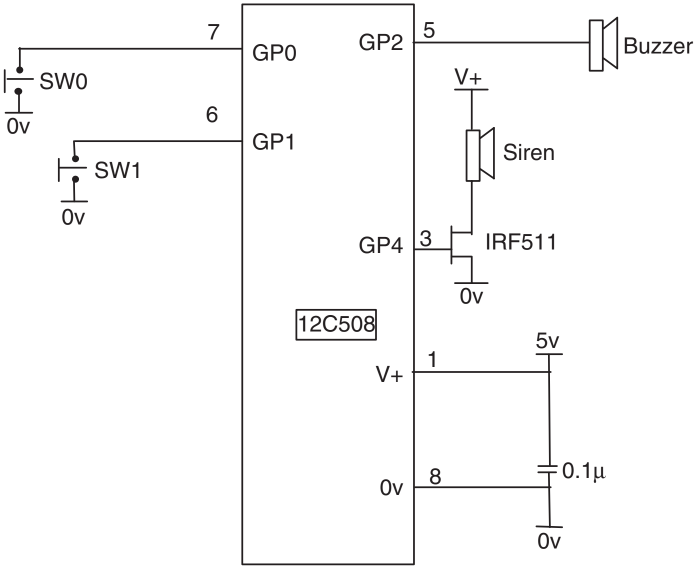

Figure 18.6 Burglar alarm using 12C508

The flowchart of course is the same. The code is shown below as ALARM_12.ASM using the header for the 12C508 from Chapter 15.

WARNING: The 12C508 only has a two level deep stack which means when you do a CALL you can only do one more CALL from that subroutine otherwise the program will get lost.

Program code for 12C508 burglar alarm

```
;ALARM_12.ASM FOR 12C508

TMR0      EQU     1       ;TMR0 is FILE 1.
GPIO      EQU     6       ;GPIO is FILE 6.
OSCCAL    EQU     5       ;Oscillator calibration.
STATUS    EQU     3       ;STATUS is FILE 3.
ZEROBIT   EQU     2       ;ZEROBIT is Bit 2.
COUNT     EQU     07H     ;USER RAM LOCATION.
TIME      EQU     08H     ;TIME IS 39
COUNTB    EQU     09H

;**********************************************************
```

```
            LIST        P=12C508    ;We are using the 12C508.
            ORG         0           ;0 is the start address.
            GOTO        START       ;goto start!

;*******************************************************
;Configuration Bits

__CONFIG H'0FEA'        ;selects Internal RC oscillator, WDT off,
                        ;Code Protection disabled.

;*******************************************************
;SUBROUTINE SECTION.

;1 second delay
DELAY1      MOVLW     .100                ;100 × 1/100 SEC.
            MOVWF     COUNT
TIMEA       CLRF      TMR0                ;Start TMR0
LOOPB       MOVF      TMR0,W              ;Read TMR0 into W
            SUBWF     TIME,W              ;TIME-W
            BTFSS     STATUS,ZEROBIT      ;Check TIME-W=0
            GOTO      LOOPB
            DECFSZ    COUNT
            GOTO      TIMEA
            RETLW     0

;1/2 second delay
DELAYP5     MOVLW     .50                 ;50 × 1/100 SEC.
            MOVWF     COUNT
TIMEB       CLRF      TMR0                ;Start TMR0
LOOPC       MOVF      TMR0,W              ;Read TMR0 into W
            SUBWF     TIME,W              ;TIME-W
            BTFSS     STATUS,ZEROBIT      ;CHECK TIME-W=0
            GOTO      LOOPC
            DECFSZ    COUNT
            GOTO      TIMEB
            RETLW     0

;1/4 second delay
DELAYP25MOVLW         .25                 ;25 × 1/100 SEC.
            MOVWF     COUNT
TIMEC       CLRF      TMR0                ;Start TMR0
LOOPD       MOVF      TMR0,W              ;Read TMR0 IN W
```

```
        SUBWF   TIME,W            ;TIME-W
        BTFSS   STATUS,ZEROBIT    ;Check TIME-W=0
        GOTO    LOOPD
        DECFSZ  COUNT
        GOTO    TIMEC
        RETLW   0


;2 second delay
DELAY2  MOVLW   .200              ;200 × 1/100 SEC.
        MOVWF   COUNT
TIMED   CLRF    TMR0              ;Start TMR0
LOOPE   MOVF    TMR0,W            ;Read TMR0 IN W
        SUBWF   TIME,W            ;TIME-W
        BTFSS   STATUS,ZEROBIT    ;Check TIME-W=0
        GOTO    LOOPE
        DECFSZ  COUNT
        GOTO    TIMED
        RETLW   0


BUZZER   MOVLW    .5
         MOVWF    COUNTB     ;5 × 2 Seconds
BUZZ1    BSF      GPIO,2
         CALL     DELAY1
         BCF      GPIO,2
         CALL     DELAY1
         DECFSZ   COUNTB
         GOTO     BUZZ1
         MOVLW    .10
         MOVWF    COUNTB     ;10 × 1 Second
BUZZ2    BSF      GPIO,2
         CALL     DELAYP5
         BCF      GPIO,2
         CALL     DELAYP5
         DECFSZ   COUNTB
         GOTO     BUZZ2
         MOVLW    .20
         MOVWF    COUNTB
BUZZ3    BSF      GPIO,2     ;20 × 0.5 Seconds
         CALL     DELAYP25
         BCF      GPIO,2
         CALL     DELAYP25
```

```
            DECFSZ    COUNTB
            GOTO      BUZZ3
            RETLW     0

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

START       MOVWF     OSCCAL
            MOVLW     B'00101011'  ;GPIO bits 2 and 4 are O/Ps.
            TRIS      GPIO
            MOVLW     B'00000111'
            OPTION                 ;PRESCALER is /256
            CLRF      GPIO         ;Clears GPIO
            MOVLW     .39
            MOVWF     TIME

;**********************************************************

;Program starts now.

            CALL      BUZZER       ;Exit delay
CHK_ON      BTFSC     GPIO,0       ;Check for alarm
            GOTO      ENTRY
            BTFSC     GPIO,1
            GOTO      SIREN
            GOTO      CHK_ON

ENTRY       CALL      BUZZER       ;Entry delay
SIREN       BSF       GPIO,4       ;5 minute siren
            MOVLW     .150
            MOVWF     COUNTB
WAIT5       CALL      DELAY2       ;150 × 2 seconds
            DECFSZ    COUNTB
            GOTO      WAIT5
            BCF       GPIO,4       ;Turn siren off

CHK_OFF     BTFSC     GPIO,0       ;Check switches closed
            GOTO      CHK_OFF
            BTFSC     GPIO,1
            GOTO      CHK_OFF
            CALL      DELAYP25     ;antibounce
            GOTO      CHK_ON
END
```

Fault finding

What if it all goes wrong!

The block diagram of the microcontroller in Figure 18.7 shows 3 sections:

Inputs, the microcontroller and outputs.

INPUTS
MICROCONTROLLER
OUTPUTS

Figure 18.7 Block diagram of the microcontroller circuit

The microcontroller makes the output respond to changes in the inputs under program control.

All microcontroller circuits will have outputs and most will have inputs.

Check the supply voltage

Check that the correct voltages are going to the pins. 5v on Vdd, pin 14 and MCLR, pin 4 and 0v on Vss, pin 5, on the 16F84.

Checking inputs

If the inputs are not providing the correct signals to the micro then the outputs will not respond correctly.

Before checking inputs or outputs it is best to remove the microcontroller from the circuit – with the power switched off. You have inserted the micro in an IC holder so that it can be removed easily! This is essential for development work.

In order to check the inputs and outputs to the microcontroller let us consider a circuit we have looked at before in Chapter 5, the Switch Scanning Circuit, shown below in Figure 18.8.

The four switches sw0, sw1, sw2 and sw3 turned on LED0, LED1, LED2 and LED3 respectively.

To test the inputs monitor the voltage on the input pins to the microcontroller, pins 1, 2, 17 and 18. They should go high and low as you throw the switches.

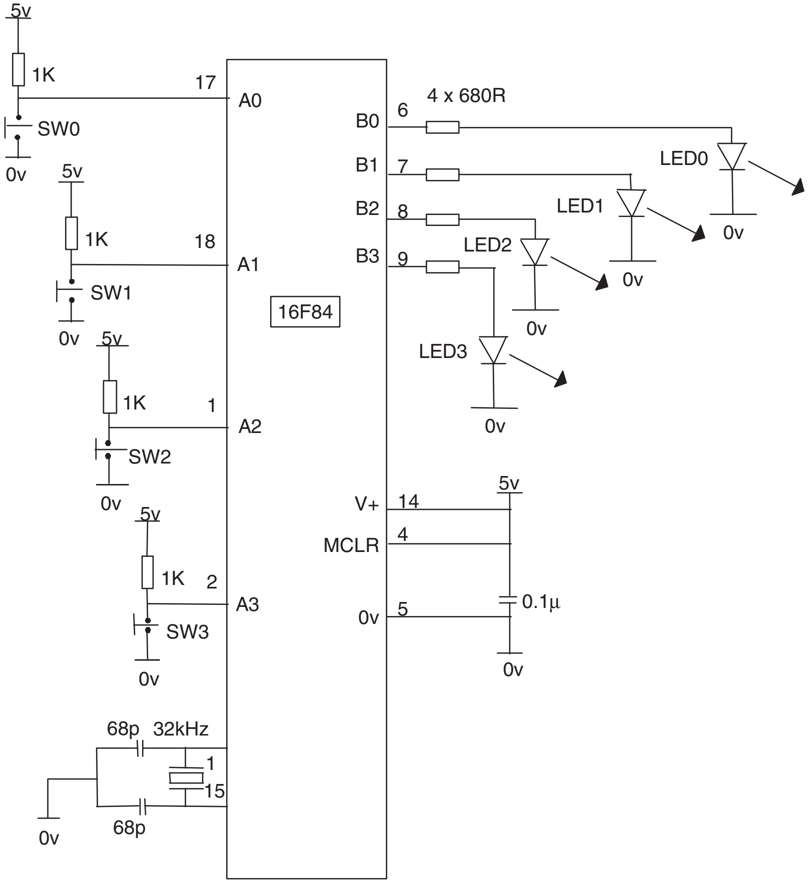

Figure 18.8 The switch scanning circuit

Checking outputs

The microcontroller will output 5v to turn on the outputs.

To make sure the outputs are connected correctly, apply 5v to each output pin in turn to make sure the corresponding LED lights.

When 5v is applied to pin 6, the B0 output then LED0 should light, etc. If it doesn't the resistor value could be incorrect or the LED faulty or in the wrong way round.

Check the oscillator

Check the oscillator is operating by monitoring the signal on CLKOUT, pin 15, with an oscilloscope or counter. Correct selection of the oscillator capacitor values are important – use 68pF with the 16C54 and 16F84 when using a 32kHz crystal.

Has the micro been programmed for the correct oscillator: R-C, LP, XT or HS. Most programs in this book use the LP configuration for the 32kHz Oscillator.

If everything is OK so far then the fault is with the microcontroller chip or the program.

Checking the microcontroller

If the program is not running it could be that you have a faulty microcontroller. You could of course try another, but how do you know if that is a good one or not. The best course of action is to load a program you know works, into the micro. Such as FLASHER.ASM from Chapter 2. This flashes an LED on and off for one second, it doesn't use any inputs and only 1 output B0.

Checking the code

If there are no hardware faults then the problem is in your code.

I find a useful aid is first of all turn an LED on for 1 second and then turn it off. When this works you know that the microcontroller is ok, and that your timing has been set correctly and the oscillator and power supply are functioning correctly. With the switch scanning circuit you could turn all 4 LEDs on for 1 second anyway to serve as an LED check.

To check your code, break it up into sections. Look at were the program stops running to identify the problem area.

If possible turn on LEDs on the outputs to indicate where you are in the program. If you are supposed to turn LED3 on when you go into a certain section of code and LED3 doesn't turn on, then of course you have not gone into that section you are stuck somewhere else.

These instructions can be removed later when the program is working.

Using a simulator

By using a simulator such as the one contained in MPLAB you can single step through the program and check it out a line at a time. To use the simulator from MPLAB select – Debugger, Select Tool, MPLAB SIM as shown in Figure 18.9.

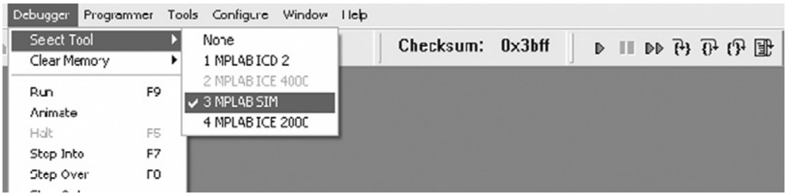

Figure 18.9 Selecting MPLAB SIM

Common faults

Here are just a few daft things my students (or I!) have done:

- Not switched the power on.
- Put the chip in upside down.
- Programmed the wrong program into the micro.
- Corrected faults in the code but forgot to assemble it again, thus blowing the previous incorrect HEX file again.
- Programmed incorrect fuses, i.e. Watchdog Timer and Oscillator.

Development kits

There are a number of development kits on the market (and you can make your own). They have a socket for your micro, inputs and outputs that you can connect to your micro. They are ideal for program development. Once verified using the kit if the system does not work then your circuit is at fault. I have developed such a kit shown in Figure 18.10. Details of it can be found on the SL Electrotech website at:
http://www.slelectrotech.com

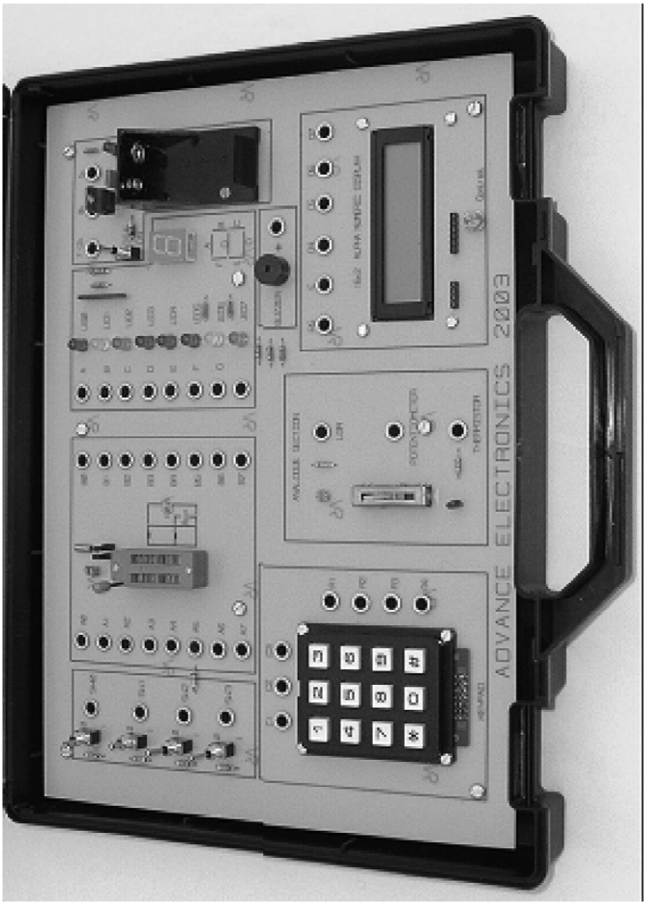

Figure 18.10 PIC microcontroller development kit