5
Using inputs

A control program usually requires more than turning outputs on and off. They switch on and off because an event has happened. This event is then connected to the input of the microcontroller to 'tell' it what to do next. The input could de derived from a switch or it could come from a sensor measuring temperature, light levels, soil moisture, air quality, fluid pressure, engine speed etc.

Analogue inputs are dealt with later, in this chapter we will concern ourselves with digital on/off inputs.

New instructions used in this chapter:

- BTFSC
- BTFSS
- CLRF
- MOVF
- SUBLW
- SUBWF
- RETLW

As an example let us design a circuit so that switch, SW1 will turn an LED on and off.

The circuit diagram is shown in Figure 5.1.

This circuit is using the 16F84 microcontroller with a 32kHz crystal.

It can of course also be performed with any of the microcontrollers discussed previously. Including the 16F818 using its internal oscillator, in which case the crystal and 2×68pF capacitors are not required.

The program to control the hardware would use the following steps:

1. Wait for SW1 to close.
2. Turn on LED1.
3. Wait for SW1 to open.

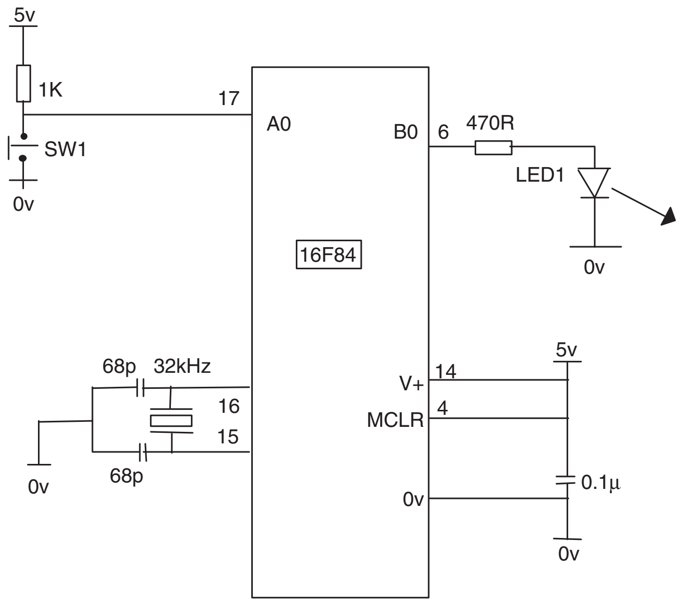

Figure 5.1 Circuit diagram of the microcontroller switch

4. Turn off LED1.
5. Repeat.

In the circuit diagram SW1 is connected to A0 and LED1 to B0.

When the switch is closed A0 goes low or clear. So we wait until A0 is clear. The code for this is:

```
BEGIN    BTFSC    PORTA,0 (test bit 0 in file PORTA skip if clear)
         GOTO     BEGIN
         BSF      PORTB,0
```

- The command BTFSC is Bit Test in File Skip if Clear, and the instruction BTFSC PORTA,0 means Test the Bit in the File PORTA, i.e. Bit0, Skip the next instruction if Clear. If A0 is Clear Skip the next instruction (GOTO BEGIN) if it isn't Clear then do not Skip and GOTO BEGIN to check the switch again.

The program will check the switch thousands maybe millions of times a second, depending on your clock.

- When the switch is pressed the program moves on and executes the instruction BSF PORTB,0 to turn on the LED.

We then wait for the switch to open.

When the switch is open A0 goes Hi or Set, we then wait until A0 is Set i.e.

```
SWOFF   BTFSS   PORTA,0
        GOTO    SWOFF
        BCF     PORTB,0
        GOTO    BEGIN
```

- The command BTFSS is Bit Test in File Skip if Set, and the instruction BTFSS PORTA,0 means Test the Bit in the File PORTA, i.e. Bit0, Skip the next instruction if Set. If A0 is Set Skip the next instruction (GOTO SWOFF) if it isn't Set then do not Skip and GOTO SWOFF to check the switch again.
- When the switch is set the program moves on and executes the instruction BCF PORTB,0 to switch off the LED.
- The program then goes back to the label BEGIN, to repeat.

The program is now added to the header. (NB. Use the TAB to make your listing easy to read.) It is then saved as SWITCH.ASM.

```
;SWITCH.ASM
;**************************************************
;Program starts now.
BEGIN   BTFSC   PORTA,0   ;Wait for SW1 to be pressed
        GOTO    BEGIN
        BSF     PORTB,0   ;Turn on LED1.
SWOFF   BTFSS   PORTA,0   ;Wait for SW1 to be released.
        GOTO    SWOFF
        BCF     PORTB,0   ;Switch off LED1.
        GOTO    BEGIN     ;Repeat sequence.

END
```

Switch flowchart

It will be obvious from the program listing of the solution to the switch problem that listings are difficult to follow. A picture is worth a thousand words has never been more apt than it is with a program listing. The picture of the program is shown below in the flowchart for the solution to our initial switch problem, Figure 5.2. Before a programming listing is attempted it is very worthwhile drawing a flowchart to depict the program steps. Diamonds are used to show a decision (i.e. a branch) and rectangles are used to show

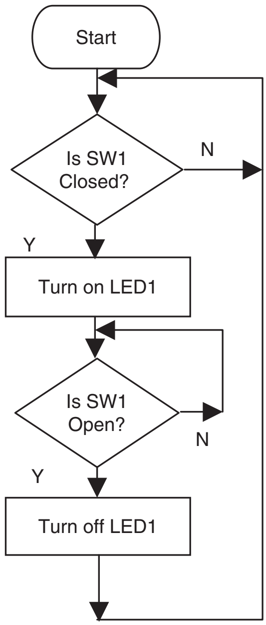

Figure 5.2 Flowchart for the switch

a command. Each shape may take several lines of program to implement. But the idea of the flowchart should be evident. Note that the flowchart describes the problem – you can draw it without any knowledge of the instruction set.

Program development

From our basic switch circuit an obvious addition would be to include a delay so that the LED would go off automatically after a set time.

Suppose we wish to switch the light on for 5 seconds, using A0 as the switch input. Figure 5.3 shows this Delay Flowchart.

The complete listing for this program for the 16F84 is shown below. I have shown the complete code including the header because I have added a 5 second delay in the subroutine section.

```
;DELAY.ASM

;EQUATES SECTION

TMR0        EQU    1        ;TMR0 is FILE 1.
PORTA       EQU    5        ;PORTA is FILE 5.
```

```
PORTB       EQU   6         ;PORTB is FILE 6.
STATUS      EQU   3         ;STATUS is FILE3.
TRISA       EQU   85H       ;TRISA (the PORTA I/O selection)
TRISB       EQU   86H       ;TRISB (the PORTB I/O selection)
OPTION_R    EQU   81H       ;the OPTION register is file 81H
ZEROBIT     EQU   2         ;ZEROBIT is Bit 2.
COUNT       EQU   0CH       ;USER RAM LOCATION.
;*****************************************************
LIST        P=16F84         ;We are using the 16F84.
ORG         0               ;0 is the start address.
GOTO        START           ;goto start!
;*****************************************************
;Configuration Bits

__CONFIG H'3FF0'          ;selects LP oscillator, WDT off, PUT on,
                          ;Code Protection disabled.
;************************************************
;SUBROUTINE SECTION.

;5 second delay.
DELAY5    CLRF      TMR0              ;Start TMR0.
LOOPA     MOVF      TMR0,W            ;Read TMR0 into W.
          SUBLW     .160              ;TIME - 160
          BTFSS     STATUS,ZEROBIT    ;Check TIME-W=0
          GOTO      LOOPA             ;Time is not=160.
          RETLW     0                 ;Time is 160, return.

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

START       BSF       STATUS,5      ;Turn to BANK1
            MOVLW     B'00011111'   ;5 bits of PORTA are I/Ps.
            MOVWF     TRISA
            MOVLW     B'00000000'
            MOVWF     TRISB         ;PORTB IS OUTPUT
            MOVLW     B'00000111'
            MOVWF     OPTION_R      ;PRESCALER is /256
            BCF       STATUS,5      ;Return to BANK0
            CLRF      PORTA         ;Clears PORTA
            CLRF      PORTB         ;Clears PORTB
            CLRF      COUNT

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

```
;Program starts now.
ON          BTFSC       PORTA,0     ;Check button pressed.
            GOTO        ON
            BSF         PORTB,0     ;Turn on LED.
            CALL        DELAY5      ;CALL 5 second delay
            BCF         PORTB,0     ;Turn off LED.
            GOTO        ON          ;Repeat

END
```

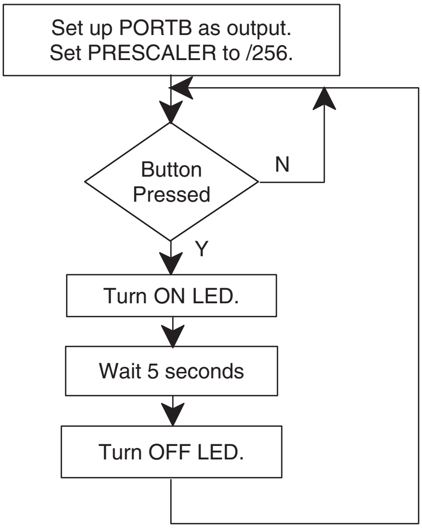

Figure 5.3 Delay flowchart

How does it work?

- We check to see if the switch has been pressed (clear). If not GOTO ON and check again. If it has skip that line and Turn on the LED on B0. The code is:

```
ON          BTFSC       PORTA,0     ;Check button pressed.
            GOTO        ON
            BSF         PORTB,0     ;Turn on LED.
```

- Wait 5 seconds. The 5 second delay has been included for you in the subroutine section. Code:

```
CALL        DELAY5
```

- Turn the LED off and go back to the beginning. Code:

```
BCF       PORTB,0     ;Turn off LED.
GOTO      ON
```

Try this next problem for yourselves, before looking at the solution.

Problem 1: Using Port A bit 0 as a start button and outputs on PortB bits 0-3. Switch on Port B bits 0 and 2 for ¼ second, switch off bits 0 and 2.
Switch on Port B bits 1 and 3 for ¼ second, switch off bits 1 and 3.
Repeat continuously.
The ¼ second delay is provided for you.

The flowchart for the solution to problem1 is shown in Figure 5.4

Program solution to problem1 for the 16F84

```
;PROBLEM1.ASM

;EQUATES SECTION

TMR0        EQU    1       ;TMR0 is FILE 1.
PORTA       EQU    5       ;PORTA is FILE 5.
PORTB       EQU    6       ;PORTB is FILE 6.
STATUS      EQU    3       ;STATUS is FILE 3.
TRISA       EQU    85H     ;TRISA (the PORTA I/O selection)
TRISB       EQU    86H     ;TRISB (the PORTB I/O selection)
OPTION_R    EQU    81H     ;the OPTION register is file 81H
ZEROBIT     EQU    2       ;ZEROBIT is Bit 2.
COUNT       EQU    0CH     ;USER RAM LOCATION.
;****************************************************
LIST        P=16F84        ;we are using the 16F84.
ORG         0              ;the start address in memory is 0
GOTO        START          ;goto start!
;****************************************************
```

```
;Configuration Bits

__CONFIG H'3FF0'              ;selects LP oscillator, WDT off, PUT on
                              ;Code Protection disabled.
;*******************************************************
;SUBROUTINE SECTION.

;0.25 second delay.
DELAY   CLRF    TMR0              ;START TMR0.
LOOPA   MOVF    TMR0,W            ;READ TMR0 INTO W.
        SUBLW   .8                ;TIME - 8
        BTFSS   STATUS,ZEROBIT    ;Check TIME-W = 0
        GOTO    LOOPA             ;Time is not = 8.
        RETLW   0                 ;Time is 8, return.

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

;CONFIGURATION SECTION

START   BSF     STATUS,5      ;Turn to BANK1
        MOVLW   B'00011111'   ;5 bits of PORTA are I/Ps.
        MOVWF   TRISA
        MOVLW   B'00000000'
        MOVWF   TRISB         ;PORTB IS OUTPUT
        MOVLW   B'00000111'
        MOVWF   OPTION_R      ;PRESCALER is /256
        BCF     STATUS,5      ;Return to BANK0
        CLRF    PORTA         ;Clears PORTA
        CLRF    PORTB         ;Clears PORTB

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

ON      BTFSC   PORTA,0       ;Check button pressed.
        GOTO    ON
REPEAT  MOVLW   B'00000101'
        MOVWF   PORTB         ;Turn on bits 0 and 2
        CALL    DELAY         ;¼ second delay
        MOVLW   B'00001010'
        MOVWF   PORTB         ;Turn on bits 1 and 3
        CALL    DELAY         ;¼ second delay
        GOTO    REPEAT        ;Repeat
END
```

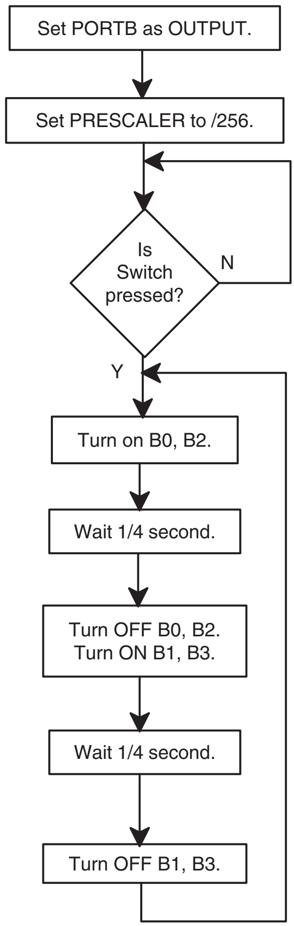

Figure 5.4 Flowchart for problem

How does it work?

- Wait for the switch on PORTA,0 to clear, with BTFSC PORTA,0 then skip to
- MOVLW B'00000101' this sets up the data in the W register.
- MOVWF PORTB transfers the W register to PORTB and puts 5v on B0 and B2 only.
- CALL DELAY waits for ¼ second.
- MOVLW B'00001010' this sets up the data in the W register.
- MOVWF PORTB transfers the W register to PORTB and puts 5v on B1 and B3 only.
- CALL DELAY waits for ¼ second.

- GOTO REPEAT sends the program back to (my) label, REPEAT. This will keep the lights flashing all the time without checking the switch again.

Question. How do we make the program look at the switch, so that we can control whether or not the program repeats?

Answer: Instead of GOTO REPEAT use GOTO BEGIN. The program will then goto the label BEGIN instead of REPEAT and will wait for the switch to be Clear before repeating.

Extra Work. Try and make the flashing routine more interesting by adding more combinations.

Scanning (using multiple inputs)

Scanning (also called polling) is when the microcontroller looks at the condition of a number of inputs in turn and executes a section of program depending on the state of those inputs.

Applications include:

- Burglar Alarms – when sensors are monitored and a siren sounds either immediately or after a delay depending on which input is active.
- Keypad scanning – a key press could cause an LED to light, a buzzer to sound or a missile to be launched. Just do not press the wrong key!

Let's consider a simple example:

Switch scanning

Design a circuit so that if a switch is pressed a corresponding LED will light. i.e.

If SW0 is Hi, (logic1 or Set) then LED0 is on.
If SW0 is Low, (logic 0 or Clear) then LED0 is off.
If SW1 is Hi, (logic1 or Set) then LED1 is on.
If SW1 is Low, (logic 0 or Clear) then LED1 is off.
etc.

The circuit diagram for this is shown if Figure 5.5 and the corresponding flowchart in Figure 5.6.

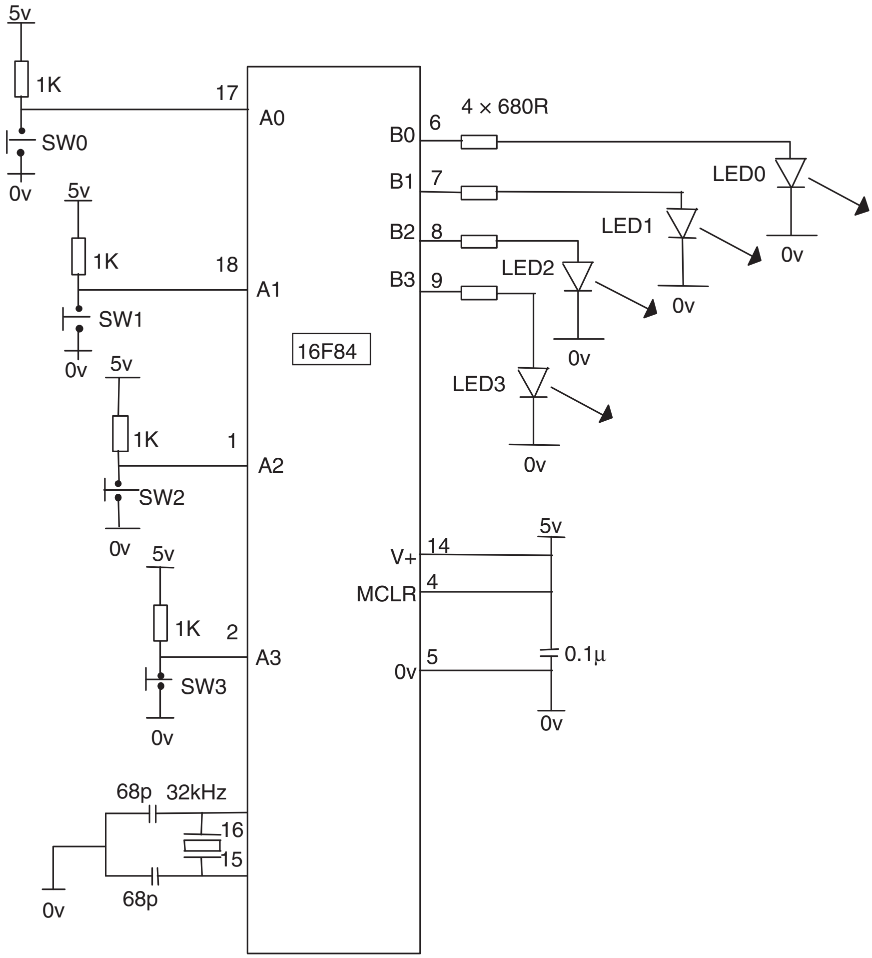

Figure 5.5 Switch scanning circuit

The program for this switch scan is:

;SWSCAN.ASM using 16F84 and 32kHz crystal.

;EQUATES SECTION

TMR0	EQU	1	;TMR0 is FILE 1.
PORTA	EQU	5	;PORTA is FILE 5.
PORTB	EQU	6	;PORTB is FILE 6.

```
STATUS      EQU    3          ;STATUS is FILE3.
TRISA       EQU    85H        ;TRISA (the PORTA I/O selection)
TRISB       EQU    86H        ;TRISB (the PORTB I/O selection)
OPTION_R    EQU    81H        ;the OPTION register is file 81H
ZEROBIT     EQU    2          ;ZEROBIT is Bit 2.
COUNT       EQU    0CH        ;USER RAM LOCATION.
;*****************************************************
LIST        P=16F84           ;We are using the 16F84.
ORG         0                 ;0 is the start address.
GOTO        START             ;goto start!
;*****************************************************
;Configuration Bits

__CONFIG H'3FF0'              ;selects LP oscillator, WDT off, PUT on,
                              ;Code Protection disabled.

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

START    BSF       STATUS,5      ;Turn to BANK1
         MOVLW     B'00011111'   ;5 bits of PORTA are I/Ps.
         MOVWF     TRISA
         MOVLW     B'00000000'
         MOVWF     TRISB         ;PORTB IS OUTPUT
         MOVLW     B'00000111'
         MOVWF     OPTION_R      ;PRESCALER is /256
         BCF       STATUS,5      ;Return to BANK0
         CLRF      PORTA         ;Clears PORTA
         CLRF      PORTB         ;Clears PORTB
         CLRF      COUNT

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

SW0         BTFSC     PORTA,0       ;Switch0 pressed?
            GOTO      TURNON0       ;Yes
            BCF       PORTB,0       :No, Switch off LED0.

SW1         BTFSC     PORTA,1       ;Switch1 pressed?
            GOTO      TURNON1       ;Yes
            BCF       PORTB,1       :NO Switch off LED1.
```

```
SW2         BTFSC   PORTA,2     ;Switch2 pressed?
            GOTO    TURNON2     ;Yes
            BCF     PORTB,2     :NO Switch off LED2.

SW3         BTFSC   PORTA,3     ;Switch3 pressed?
            GOTO    TURNON3     ;Yes
            BCF     PORTB,3     :NO Switch off LED3.
            GOTO    SW0         ;Rescan.

TURNON0     BSF     PORTB,0     ;Turn on LED0
            GOTO    SW1

TURNON1     BSF     PORTB,1     ;Turn on LED1
            GOTO    SW2

TURNON2     BSF     PORTB,2     ;Turn on LED2
            GOTO    SW3

TURNON3     BSF     PORTB,3     ;Turn on LED3
            GOTO    SW0

END
```

How does it work?

- SW0 is checked first with the instruction BTFSC PORTA,0. If the switch is closed when the program is executing this line then we GOTO TURNON0. That is the program jumps to the label TURNON0 which turns on LED0 and then jumps the program back to check SW1 at, of course, the label, SW1.
- SW1 is then checked in the same manner and then SW2 and SW3.

Suppose we press the switch when the program is not looking at it. The program lines are being executed at ¼ of the clock frequency i.e. 32,768Hz that is 8192 lines a second. The program will always catch you!

Try modifying the program so that the switches can flash 4 different routines e.g. SW0 flashes all lights on and off 5 times for 1 second.

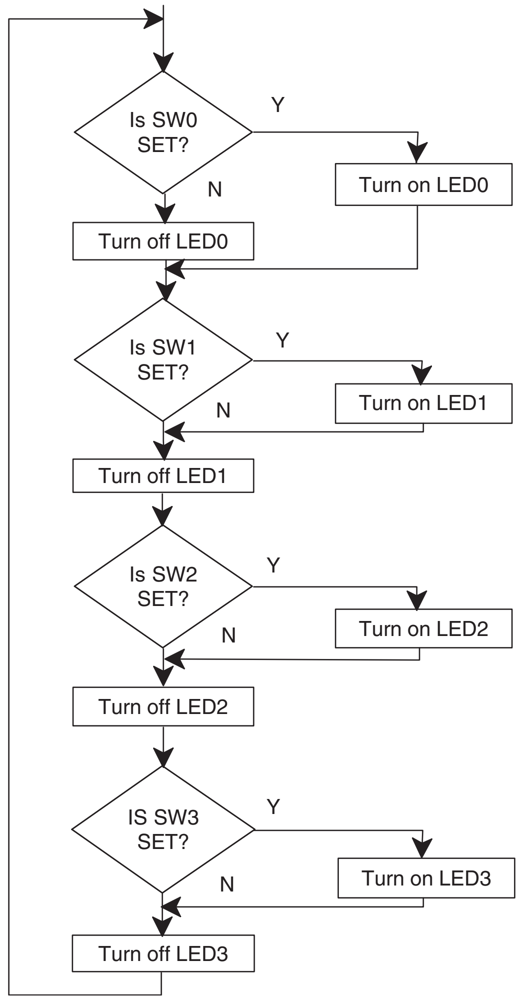

Figure 5.6 Flowchart for switch scan

Control application – a hot air blower

The preceding section outlined how to monitor inputs by looking at them in turn. This application will 'read' all the bits on the port at once, because we will be concerned with particular combinations of inputs rather than individual ones.

The bits on the Input Port will be 0s or 1s and we can treat this binary pattern like any other number in a file.

Consider a controller for a hot air radiator. When the water is warm the fan will blow the warm air into the room. The heater and fan are controlled by 3 temperature sensors: (a) a room temperature sensor, (b) a boiler water temperature sensor and (c) a safety overheating sensor. The truth table for the system is shown in Table 5.1, where a 1 means hot and a 0 means cold for the sensors.

The block diagram for the system is shown in Figure 5.7.

Note A3, A4, A5, A6 and A7 are inputs and need to be connected to 0v. Do not leave them floating – you would not know if they were 0 or 1! Even though

Table 5.1 Truth table for the hot air system

INPUTS								OUTPUTS	
A 7	A 6	A 5	A 4	A3	Room A2	Water A1	OverH A0	Heater B1	Fan B0
0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1
0	0	0	0	0	0	1	1	0	1
0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	1	0	1	0	1
0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	1	1	1	0	1

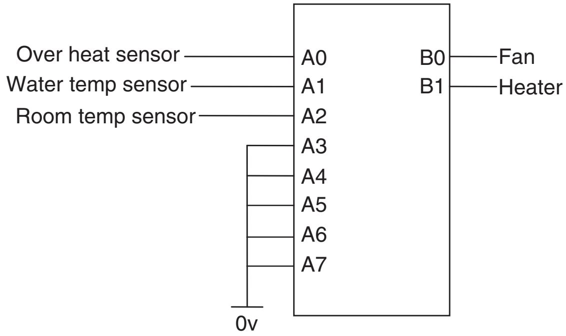

Figure 5.7 Block diagram for the hot air system

they are not being used they are still being read. NB. The inputs A5, A6 and A7 do not exist on the 16F84.

There are 8 input conditions from our 3 sensors. So all 8 must be checked to determine which condition is true.

Consider the first condition A2 = A1 = A0 = 0, i.e. PORTA reads 0000 0000. How do we know that PORTA is 0000 0000? We do not have an instruction which says "is PORTA equal to 0000 0000" or any other value for that matter. So we need to look at our 35 instructions and come up with a way of finding out what is the binary value of PORTA.

We check for this condition by subtracting 00000000 from it, if the answer is zero then PORTA reads 00000000. I.e. 0000 0000 − 0000 0000 = 0 (obviously). But how do we subtract the two numbers and how do we know if the answer is zero?

This is a very important piece of programming so read the next few lines *carefully*.

- We first of all read PORTA into the W register with the instruction MOVF PORTA,W that moves the data, (setting of the switches, 1s or 0s), into W.
- We then subtract the number we looking for in this case 00000000 from W.
- We then need to know if the answer to this subtraction is zero. If it is then the value on PORTA was 00000000. If the answer is not zero then the value of the data on PORTA was not zero.
- So is the answer zero? Yes or No? The answer is held in a register called the Status Register, in bit 2 of this register, called the zero bit. If the zero bit, called a flag is 1, it is indicating that the statement is true the calculation was zero. If the zero bit is 0 that indicates the statement is false the answer was not zero.
- We test the zero bit in the status register just like we tested the bit on the switch connected to PORTA at the start of this chapter. We use the command BTFSC and the instruction BTFSC STATUS,ZEROBIT. If the zero bit is clear we skip the next instruction if it is set we have a match and do not skip.

The code for this is:

```
MOVLW    B'00000000'        ;put 000000 in W
SUBWF    PORTA              ;subtract W from PORTA
BTFSC    STATUS,ZEROBIT     ;PORTA = 00000000?
CALL     CONDA              ;yes
```

CONDA is short for condition A where we require the heater on and the fan off.

- To check for A2 = A1 = 0 and A0 = 1 we subtract 00000001. To check for the next condition A2 = 0, A1 = 1, A0 = 0 we subtract 00000010, and so on for the other 5 conditions.

```
MOVLW      B'00000001'          ;put 00000001 in W
SUBWF      PORTA                ;subtract W from PORTA
BTFSS      STATUS,ZEROBIT       ;PORTA = 00000001?
CALL       CONDB                ;yes
etc.
```

The opcode for this program CONTROL.ASM is:

```
;CONTROL.ASM

;SUBROUTINE SECTION.

CONDA      BCF        PORTB,0        ;turns fan off
           BSF        PORTB,1        ;turns heater on
           RETLW      0

CONDB      BSF        PORTB,0        ;turns fan on
           BCF        PORTB,1        ;turns heater off
           RETLW      0

CONDC      BSF        PORTB,0        ;turns fan on
           BSF        PORTB,1        ;turns heater on
           RETLW      0

CONDD      BCF        PORTB,0        ;turns fan off
           BCF        PORTB,1        ;turns heater off
           RETLW      0
;*****************************************************
;Program starts now.

BEGIN     MOVLW     B'00000000'        ;put 00000000 in W
          SUBWF     PORTA              ;PORTA - W
          BTFSC     STATUS,ZEROBIT     ;PORTA = 00000000?
          CALL      CONDA              ;yes

          MOVLW     B'00000001'        ;put 00000001 in W
          SUBWF     PORTA              ;PORTA - W
          BTFSC     STATUS,ZEROBIT     ;PORTA = 00000001?
          CALL      CONDB              ;yes
```

```
        MOVLW   B'00000010'        ;put 00000010 in W
        SUBWF   PORTA              ;PORTA - W
        BTFSC   STATUS,ZEROBIT     ;PORTA = 00000010?
        CALL    CONDC              ;yes

        MOVLW   B'00000011'        ;put 00000011 in W
        SUBWF   PORTA              ;PORTA - W
        BTFSC   STATUS,ZEROBIT     ;PORTA = 00000011?
        CALL    CONDB              ;yes

        MOVLW   B'00000100'        ;put 00000100 in W
        SUBWF   PORTA              ;PORTA - W
        BTFSC   STATUS,ZEROBIT     ;PORTA = 00000100?
        CALL    CONDD              ;yes

        MOVLW   B'00000101'        ;put 00000101 in W
        SUBWF   PORTA              ;PORTA - W
        BTFSC   STATUS,ZEROBIT     ;PORTA = 00000101?
        CALL    CONDB              ;yes

        MOVLW   B'00000110'        ;put 00000110 in W
        SUBWF   PORTA              ;PORTA - W
        BTFSC   STATUS,ZEROBIT     ;PORTA = 00000110?
        CALL    CONDD              ;yes

        MOVLW   B'00000111'        ;put 00000111 in W
        SUBWF   PORTA              ;PORTA - W
        BTFSC   STATUS,ZEROBIT     ;PORTA = 00000111?
        CALL    CONDB              ;yes

        GOTO    BEGIN

END
```

Notice that the SUBROUTINE SECTION needs to be changed to include the conditions, CONDA, CONDB, CONDC and CONDD. The DELAY subroutines are not required in this example.

The program can be checked by using switches for the input sensors and LEDs for the outputs.

There is more than one way of skinning a cat, another way of writing this program is shown in Chapter 8, in the section on look up tables.