Chapter 5

Keypad Scanning

There are numerous applications using a keypad in an electronic circuit to enter data or make a choice from a number of selections. In such circuits as:

- Door entry systems
- Scoreboards, darts, snooker, cricket
- Burglar alarms
- Program selection as in a washing machine
- Remote controls
- Calculators

Recapping the if statement.
Just before we look at the keypad let us recap the if statement again.
The if statement looks like this:

```
if (condition is true)
{
    •
    •  Execute this code
    •
}
```

As an example

```
if (Temperature>60)
{
    Turn on fan
    Turn on alarm
    Open window
}
```

The if statement can be modified to include what to do if the condition is not true, i.e.,

```
if (Temperature>60)
```

```
{
        Turn on fan
        Turn on alarm
        Open window
        Turn red light on and green light off
}
else
{
        Turn off fan
        Turn off alarm
        Close window
        Turn red light off and green light on
}
```

The if statement can be written on one line if only one statement is needed, i.e.,

if (Temperature > 60) Turn on fan;   // note the ; here at the end of the statement.

The 12 key keypad.

Now back to the keypad.

Keypads are normally arranged in a matrix formation which reduces the number of connections.

A 12 key keypad arranged in a 3×4 matrix requires seven connections.

A 16 key keypad arranged in a 4×4 matrix requires eight connections.

A 100 key keypad arranged in a 10×10 matrix would require 20 connections.

We will consider the 12 key keypad for ease of use but the discussion also applies to the other keypads.

Consider the 12 key keypad. This is arranged in three columns and four rows as shown in Figure 5.1. There are seven connections to the keypad—C1, C2, C3, R1, R2, R3, and R4.

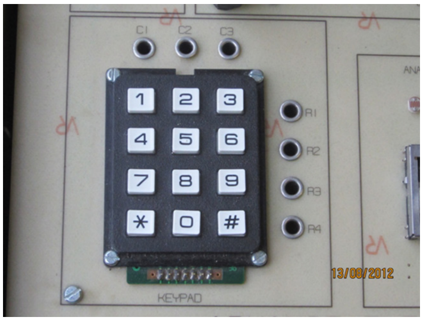

FIGURE 5.1 12 Key keypad. (The photograph is taken from the author's development kit.)

	Column1, C1	Column2, C2	Column3, C3
Row1, R1	1	2	3
Row2, R2	4	5	6
Row3, R3	7	8	9
Row4, R4	*	0	#

This connection to the micro is shown in Figure 5.2.

The keypad works in the following way:

If for example key 6 is pressed then RB6 will be joined to RB1. For key 1 RB4 would be joined to RB0 and so on as shown in Figure 5.2.

The micro would set RB4 low and scan RB0, RB1, RB2, and RB3 for a low to see if keys 1, 4, 7, or * had been pressed.

The micro would then set RB5 low and scan RB0, RB1, RB2, and RB3 for a low to see if keys 2, 5, 8, or 0 had been pressed. RB0, RB1, RB2, and RB3 would normally all be pulled up high.

Finally RB6 would be set low, and RB0, RB1, RB2, and RB3 scanned for a low to see if keys 3, 6, 9, or # had been pressed.

PORTB has been connected to the keypad instead of PORTA because there is a facility on the 18F1220 (and most other PICs) to connect internal pull up resistors so that the four resistors on RB0–RB3 are not required. Line 138 in the keypad code.

Suppose we wish to use the keypad so that pressing 5 would turn the LED on RA0 ON and pressing a 9 would turn it OFF.

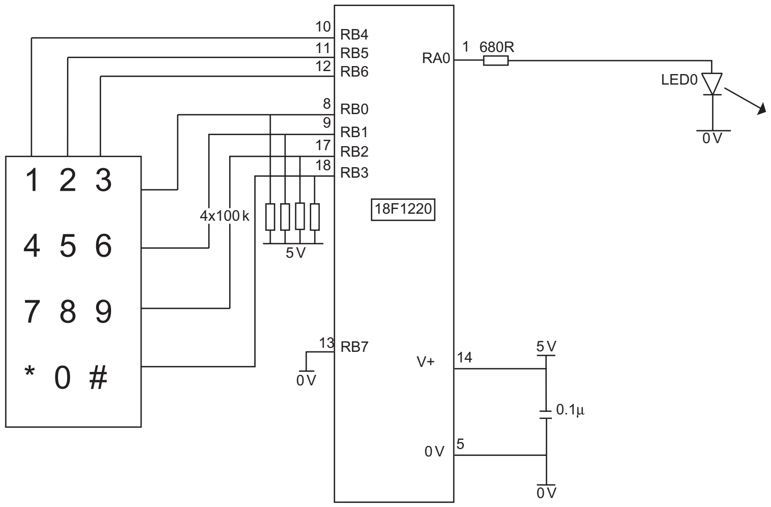

FIGURE 5.2 Keypad connection to the microcontroller.

The program for the keypad called **keypad.c** would be:

KEYPAD PROGRAM

```
// keypad.c by DW Smith, 11 October 2011
#include <p18f1220.h>
#pragma config WDT=OFF, OSC=INTIO2, PWRT=ON, LVP=OFF, MCLRE=OFF
#include <delays.h>

// put VARIABLES here.
int numberPressed;

int Scan()
{
numberPressed = 12;
while (numberPressed==12)//there is no number 12 on the keypad, wait until 12 is overwritten by number pressed
  {
PORTB = 0b11101111;                    //B4 is 0 scanning 1,4,7,*

if (PORTBbits.RB0==0)
  {
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        while (PORTBbits.RB0==0);      // wait until switch has been released
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        numberPressed = 1;
        return numberPressed;
  }

if (PORTBbits.RB1==0)
  {
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        while (PORTBbits.RB1==0);      // wait until switch has been released
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        numberPressed = 4;
        return numberPressed;
  }

if (PORTBbits.RB2==0)
  {
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        while (PORTBbits.RB2==0);      // wait until switch has been released
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        numberPressed = 7;
        return numberPressed;
  }
if (PORTBbits.RB3==0)
  {
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        while (PORTBbits.RB3==0);      // wait until switch has been released
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        numberPressed = 10;            // 10 is for * pressed
        return numberPressed;
  }


  PORTB = 0b11011111;                  //B5 is 0 scanning 2,5,8,0

 if (PORTBbits.RB0==0)
  {
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        while (PORTBbits.RB0==0);      // wait until switch has been released
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        numberPressed = 2;
        return numberPressed;
   }

 if (PORTBbits.RB1==0)
  {
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        while (PORTBbits.RB1==0);      // wait until switch has been released
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        numberPressed = 5;
        return numberPressed;
  }

 if (PORTBbits.RB2==0)
  {
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        while (PORTBbits.RB2==0);      // wait until switch has been released
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        numberPressed = 8;
        return numberPressed;
  }
 if (PORTBbits.RB3==0)
  {
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        while (PORTBbits.RB3==0);      // wait until switch has been released
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        numberPressed = 0;
        return numberPressed;
  }

  PORTB = 0b10111111;                  //B6 is 0 scanning 3,6,9,#

 if (PORTBbits.RB0==0)
   {
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        while (PORTBbits.RB0==0);      // wait until switch has been released
        Delay100TCYx(8);               // wait 0.1 s for switch bounce to stop
        numberPressed = 3;
        return numberPressed;
  }

```

```
if (PORTBbits.RB1==0)
  {
        Delay100TCYx(8);                  // wait 0.1 s for switch bounce to stop
        while (PORTBbits.RB1==0);         // wait until switch has been released
        Delay100TCYx(8);                  // wait 0.1 s for switch bounce to stop
        numberPressed = 6;
        return numberPressed;
  }

if (PORTBbits.RB2==0)
  {
        Delay100TCYx(8);                  // wait 0.1 s for switch bounce to stop
        while (PORTBbits.RB2==0);         // wait until switch has been released
        Delay100TCYx(8);                  // wait 0.1 s for switch bounce to stop
        numberPressed = 9;
        return numberPressed;
  }
if (PORTBbits.RB3==0)
  {
        Delay100TCYx(8);                  // wait 0.1 s for switch bounce to stop
        while (PORTBbits.RB3==0);         // wait until switch has been released
        Delay100TCYx(8);                  // wait 0.1 s for switch bounce to stop
        numberPressed = 11;
        return numberPressed;
  }
}
}


void main (void)
{
        //SET UP
        // OSCCON defaults to 31 kHz. So no need to alter it.
        ADCON1 = 0x7F;          //all IO are digital or 0b01111111 in binary
        TRISA = 0b11110000;     //sets PORTA bits 0–3 as outputs, 4–7 as inputs
        PORTA = 0b00000000;     //turns off PORTA outputs
        TRISB = 0b10001111;  //sets PORTB bits 0–3 and 7 as inputs, 4–6 as outputs
        PORTB = 0b00000000;     //turns off PORTB outputs, good start position
        INTCON2bits.RBPU = 0;  // turns on the internal pull ups on PORTB inputs

    while (1)
    {
    numberPressed = Scan();
    if (numberPressed==5)         PORTA = 0b00000001;

    if (numberPressed==9)         PORTA = 0b00000000;
    }
}
```

Again open a new project, give it a name say, keypad and add the keypad.c code.

This seems a rather lengthy and complicated program but let us split it into components to analyze it.

Note the line numbers would not be typed in the program, I have put them there to identify the lines.

The main part of the program is written from lines 140–147, not so lengthy after all.

- Line 140 is our continuous (while) loop that we have seen previously. The code for the loop is written between lines {141 and 146}.
- Line 142 calls the Scan routine which waits for a number to be pressed on the keypad and then returns and puts the number in the file numberPressed. Note Scan() is called a method, or if you prefer a function or subroutine. The Scan() method, lines 9–126, is called by stating its name Scan() as in line 142.
- Line 143 asks if the number pressed is equal to 5 then execute PORTA = 0b00000001 turning on bit 0 of PORTA. Note the two== means is equal to NOT equals which of course is one =.
- Line 145 asks if the number pressed is equal to 9 then execute PORTA = 0b00000000 turning off bit 0 of PORTA.
- Line 6 is a statement identifying where variables are being kept. It is good practice to declare the variables together. If you declare the variables inside a method then they can only be used in that method.
- Line 7 declares that numberPressed is a variable and that its type is an integer.
- Line 9 starts the Scan () method.
- Line 11 puts the number 12 into the variable numberPressed.
- Line 12 executes the while loop, while numberPressed is equal to 12. There is no number 12 on the keypad so that the loop will continue until 12 is overwritten by a number being pressed.
- Line 14 sets RB4 low so that we can test for RB0, RB1, RB2 or RB3 going low when a 1, 4, 7, or * (11) is pressed.
- Line 16 asks if PORTB bit RB0 is equal to 0 then execute the code between lines 18–22. If it isn't jump to line 25.
- Line 18 calls a delay of 0.1 s to allow the switch which has just been pressed to stop bouncing.
- Line 19. while (PORTBbits.RB0= =0); is a while loop which is only one line long, because of the ; at the end. So the loop keeps executing line 19 asking the question is PORTBbits.RB0 equal to 0, until it becomes 1 when it is released and the program will move onto line 20.
- Line 20 is another anti-bounce delay waiting for the switch which has just been released to stop bouncing.
- Line 21 sets numberPressed to 1 because key 1 has been pressed.

- Line 22. numberPressed is no longer 12 (it is 1) so the program returns from the Scan method to line 142 where it was called and the 1 is placed in the file numberPressed.
- Lines 143 and 145 are checked to see if numberPressed = 5 or 9 to execute the code. Then the while (1) loop will scan again line 142.
- Line 25. If a 1 wasn't pressed the code will check to see if key 4 was pressed in a similar fashion and execute lines 27–31 if 4 was pressed. If key 4 wasn't pressed then the code goes onto check for a 7 or * (10) being pressed lines 34–48.
- Line 52. if a 1, 4, 7, or * wasn't pressed then RB5 is set low and the keypad is scanned for a 2, 5, 8, or 0 being pressed as before.
- Line 89. RB6 is set low to scan for 3, 6, 9, or # (11) being pressed. If no keys are pressed then numberPressed is still 12 and the loop at line 12 executes again.

When you use the keypad for another application all you need to do is copy and paste the above code and write your new program starting at line 142.

For example consider the code to turn the LED on with 27 and off with 83.

Code to Turn LED On with 27 and Off with 83

```
142             numberPressed = Scan();
143             number1 = numberPressed;
144             numberPressed = Scan();
145             number2 = numberPressed;
146             if (number1==2 && number2==7) PORTB = 0b00000001;
147             numberPressed = Scan();
148             number3 = numberPressed;
149             numberPressed = Scan();
150             number4 = numberPressed;
151             if (number3==8 && number4==3) PORTB = 0b00000000;
```

FOUR DIGIT CODE

Suppose we wish to turn an output on when the correct four digits are entered. We could do the same as in the two digit example above, but suppose when we enter the four digits we hit two keys at once and enter five digits. The output will not turn on and we will always be out of step with our code. What we need is a way of resetting so that we know we are scanning for number1. We can of course use a reset key but the following code is a useful solution here.

```
140     // code is 2468
141
142             while (1)
143             {
144             if (num1==2 && num2==4 && num3==6 && num4==8) break;
145
146                     while (1)
147                     {
148                     num1 = Scan();
149                     if (num1!=2) break;         //start again wrong number entered
150
151                     num2 = Scan();
152                     if (num2!=4) break;         //start again wrong number entered
153
154                     num3 = Scan();
155                     if (num3!=6) break;         //start again wrong number entered
156
157                     num4 = Scan();
158                     break;
159                     }
160             }
161     PORTA = 0b00000001;//turn LED on
```

Explanation of Code

There are two continuous loops starting at line 142 and 146; the exit from these loops is via the break keyword. They are arranged with one inside the other.

- Line 140 indicates the required data is 2468.
- Lines 146–159 is a continuous while loop that collects the four digits storing them in the variables num1, num2, num3, and num4. These variables have to be declared in the "put variables here" section. The while loop continues indefinitely until a forced break out is executed.
- Line 148 scans and enters the numberPressed in num1.
- Line 149 asks if num1 is not equal to 2, then break out of the loop and re-enter in line 147. If num1 is 2 then scan for the second digit line 151. Repeat the scanning until line 157 is reached.
- Line 157 scans for num4 and breaks out whatever number is pressed!
- Line 144 checks the correct numbers have been pressed, in which case the program breaks out of the outer while(1) loop and goes to line 161 to turn on the output. Or if num4!=8 then it rejoins the scanning at the reset position at line 148.

Try modifying the code to turn the output off with four different digits. You need not use 4 to turn the output off you could use any number.

Now that we have the facility to enter data into the microcontroller a number of applications are feasible.

PUTTING THE SCAN ROUTINE IN A HEADER FILE, dwsScan.h

In the keypad program **keypad.c** the scan routine starts on line 7 and finishes on line 126, using 120 lines of the 147 line program! In order to make the program easier to read and fault find we can place the scan routine in a text file called dwsScan.h written in MPLAB. Just copy and paste lines 7–126.

Save the file dwsScan.h in the same folder as delays.h.
(Right clicking on line 4

#include <delays.h>
and opening the file delays.h will identify the path for the header file.)

The keypad program **keypad.c** can now be amended to look like **keypadh.c** below:

```
1    // keypadh.c by DW Smith, 2 April 2012
2    #include <p18f1220.h>
3    #pragma config WDT=OFF, OSC=INTIO2, PWRT=ON, LVP=OFF, MCLRE=OFF
4    #include <delays.h>
5    #include <dwsScan.h>
6    void main (void)
7    {
8    //SET UP
9    // OSCCON defaults to 31 kHz. So no need to alter it.
10   ADCON1 = 0x7F;            //all IO are digital or 0b01111111 in binary
11   TRISA = 0b11110000;       //sets PORTA bits 0–3 as outputs, 4–7 as inputs
12   PORTA = 0b00000000;       //turns off PORTA outputs
13   TRISB = 0b10001111;       //sets PORTB bits 0–3 and 7 as inputs, 4–6 as outputs
14   PORTB = 0b00000000;       //turns off PORTB outputs, good start position
15   INTCON2bits.RBPU = 0;
16   while (1)
17      {
18   numberPressed = Scan();
19   if (numberPressed == 5)    PORTA = 0b11111111;     //RA5 is Input only.
20   if (numberPressed == 9)    PORTA = 0b00000000;
21      }
22   }
```

Instead of 147 lines the program is a much more manageable 22 lines! Note the scan routine has been included in line 5:

#include <dwsScan.h>

Both programs **keypad.c** and **keypadh.c** are the same to the compiler.

You can always view the scan routine by right clicking the #include dwsScan.h line and Open File dwsScan.h.

In the dwsScan.h file I have put in seven define statements:

```
#define c1       PORTBbits.RB4//means c1 = PORTBbits.RB4
#define c2       PORTBbits.RB5
#define c3       PORTBbits.RB6
#define r1       PORTBbits.RB0
#define r2       PORTBbits.RB1
#define r3       PORTBbits.RB2
#define r4       PORTBbits.RB3
```

Where c1, c2, and c3 correspond to the columns on the keypad and r1, r2, r3, and r4 to the rows. If you are not configuring your hardware the same as mine you only need to alter the define lines because the scan routine has been written for the rows and columns not the PORT bits.