Chapter 14

Projects

PROJECT 1: COUNTDOWN TIMER

The circuit for the countdown timer is shown in Figure 14.1.

The countdown time mm:ss is entered via the keypad and the display shows the time remaining up to 99 min 59 s.

The program for the countdown timer is shown below in **Countdown.C**

```
1.  //Countdown.C by DW Smith 20-6-12
2.  #include <p18f1220.h>
3.  #pragma config WDT=OFF, OSC=INTIO2, PWRT=ON, LVP=OFF, MCLRE=OFF
4.  #include <delays.h>
5.  #include <dwsLCD.h>
6.  #include <dwsScan.h>

7.  // put variables here
8.  char secUnit, secTen, minUnit, minTen, result;

9.  void main (void)
10. {
11. //SET UP
12. // OSCCON defaults to 31 kHz. So no need to alter it.
13. ADCON1=0x7F;              //all IO are digital or 0b01111111 in binary
14. TRISA=0b00100000;         //sets PORTA
15. PORTA=0b00000000;         //turns off PORTA outputs, not required, no outputs
16. TRISB=0b00001111;         //sets PORTB as all outputs
17. PORTB=0b00000000;         //turns off PORTB outputs, good start position
18. INTCON2bits.RBPU=0;       //turns on PORTB pullups
19. T0CON=0b11010100;   //TMR0 on, 8bit timer, clocking on internal clock, prescaler/32.

20. // this code configures the display
21. WriteCmd (0x02);                          // sets 4bit operation
22. WriteCmd (FOUR_BIT & LINES_5X7);          // sets 5x7 and multiline operation.
23. WriteCmd (CURSOR_BLINK);                  // blinks cursor
24. WriteCmd (CURSOR_RIGHT);                  // moves cursor right

25. Redo:
26. WriteCmd (CLEAR_SCREEN);
27. WriteString("Enter mm:ss");
28. SetAddr (0xC0);
29. WriteString("*cancel #accept");
```

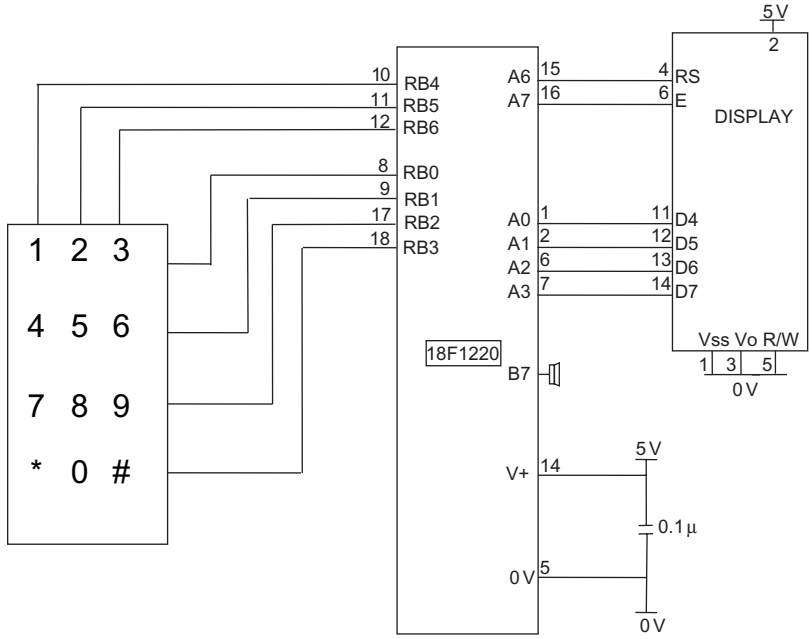

FIGURE 14.1 The countdown timer.

```
30.     Rescan:
31.     minTen=Scan();              //for mx:xx
32.     PORTBbits.RB7=0;            //buzzer off
33.     switch(minTen)
34.       {
35.     case 10:                    // for *
36.     goto Redo;
37.     break;
38.     case 11:                    // for #
39.     goto Rescan;
40.     break;
41.     default:
42.     SetAddr (0x86);
43.     WriteChar(0x30 + minTen);
44.       }

45.     minUnit=Scan();             //for xm:xx
46.     switch(minUnit)
47.       {
48.     case 10:
49.     goto Redo;
50.     break;
51.     case 11:
52.     goto Rescan;
53.     break;
```

```
54.     default:
55.     SetAddr (0x87);
56.     WriteChar(0x30 + minUnit);
57.       }

58.     secTen=Scan();              //for xx:sx
59.     switch(secTen)
60.       {
61.     case 10:
62.     goto Redo;
63.     break;
64.     case 11:
65.     goto Rescan;
66.     break;
67.     default:
68.     SetAddr (0x89);
69.     WriteChar(0x30 + secTen);
70.       }

71.     secUnit=Scan();             //for xx:xs
72.     switch(secUnit)
73.        {
74.     case 10:
75.     goto Redo;
76.     break;
77.     case 11:
78.     goto Rescan;
79.     break;
80.     default:
81.     SetAddr (0x8A);
82.     WriteChar(0x30 + secUnit);
83.       }

84.     Rescan2:
85.     result=Scan();
86.     switch(result)
87.       {
88.     case 10:
89.     goto Redo;
90.     break;
91.     case 11:                    //# accept time and start,
92.     goto Time;
93.     break;
94.     default:
95.     goto Rescan2;
96.     break;
97.       }

98.     Time:
99.     WriteCmd (CLEAR_SCREEN);
100.    WriteChar(0x30 + minTen);
101.    WriteChar(0x30 + minUnit);
102.    WriteChar(':');
103.    WriteChar(0x30 + secTen);
104.    WriteChar(0x30 + secUnit);
```

```
105.    if (secUnit!=0)
106.      {
107.    (secUnit=secUnit−1);
108.      }
109.    else
110.       {
111.                if (secTen!=0)
112.                {
113.                secTen=secTen−1;
114.                secUnit=9;
115.                }
116.                else
117.                {
118.                    if (minUnit!=0)
119.                    {
120.                    minUnit=minUnit−1;
121.                    secTen=5;
122.                    secUnit=9;
123.                    }
124.                    else
125.                        {
126.                            if(minTen!=0)
127.                            {
128.                            minUnit=9;
129.                            minTen=minTen−1;
130.                            secTen=5;
131.                            secUnit=9;
132.                            }
133.                            else
134.                                {
135.                                PORTBbits.RB7=1;   //buzzer on
136.                                goto Redo;
137.                                }
138.                        }
139.                }
140.       }
141.    while(INTCONbits.TMR0IF==0);        //wait until TMR0L=0, TMR0 interrupt flag=1
142.    TMR0L=12;                           //TMR0 overflows in 244 pulses
143.    INTCONbits.TMR0IF=0;                // reset TMR0IF
144.    goto Time;
145.    }
```

Explanation of the Code

- Lines 5 and 6 include the scan and LCD routines in the program.
- Line 8 defines the variables as characters. Characters need 8 bits of memory and store variables up to ±128. Unsigned characters store variables from 0 to 255.
- Line 19 sets up TMR0. Prescaler is / 32 so TMR0 has a pulse rate of 128μs × 32 = 4.096ms.
- Line 25 defines the label Redo. The program can branch to a label.

- Line 30 defines the label Rescan. The code following scans for the data, minutes tens, minutes units, seconds tens, and seconds units.
- Lines 31–41 scan the keypad and use the **case statement** instead of an if statement to determine which key is pressed and stores the number 0–9 in the variable minTens and displays it on the screen. For example line 33 calls the function switch (minTens) which then looks at various cases for the value of minTens (lines 34–44). Line 35 case 10 is a shorthand way of saying, if minTens is 10 execute the code and then break out of the switch section.
- Lines 45–83 collect the remainder of the time.
- Lines 84–97 Rescan2, scans for # or * to accept the data or recollect the data.
- Lines 98–104 Time, writes the remaining time to the screen.
- Lines 105–140 decrements the time every second and updates the screen.
- Line 141 waits until TMR0 overflows by monitoring the TMR0 interrupt flag.
- Line 142 preloads TMR0 with 12 so it overflows at 256 in 244 pulses, 0.999s. 244 pulses sets the TMR0 interrupt flag.
- Line 143 resets the interrupt flag.
- Line 144 redoes the Time section again.

PROJECT 2: CHESS CLOCK

Since we have just looked at the countdown timer I thought a chess clock using the countdown code would be nice application of it. I have a confession to make, it wasn't as straightforward as I thought it was going to be. So if you have nothing better to do!

The circuit diagram for the chess clock is shown in Figure 14.2.

The design of the chess clock was done using as few components as possible.

The push button connected to RB0 is used to set the time for the game in multiples of 5 min and also to switch over the times from WHITE to BLACK. This multifunction use is possible because the program is deciding what pushing the button means at any specific time.

The display gives instructions and shows the remaining times. The buzzer on RB4 sounds when one of the players is out of time.

Operation of the Unit

- When the unit is switched on the display shows:

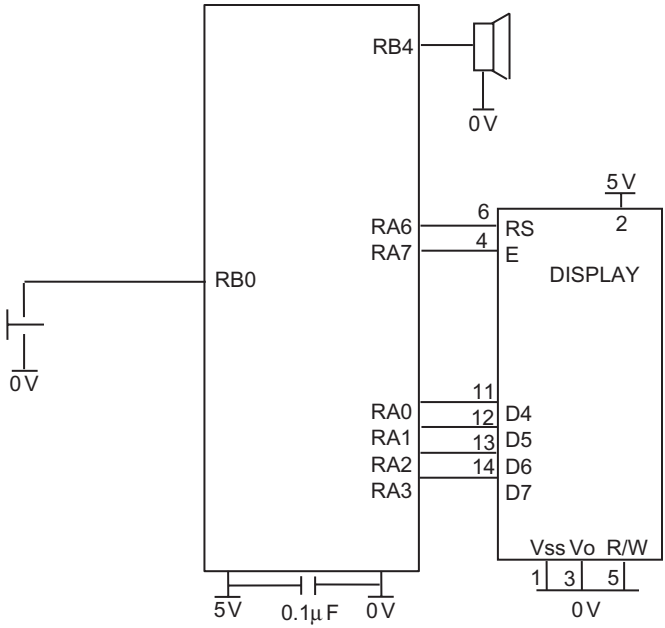

FIGURE 14.2 The chess clock.

- When the button is pressed, the time for the game is indicated on the display in multiples of 5 min. If the button is not pressed for 2 s then the time set is the one used for the game.

- The display shows the time and instructs the operator to press the button to start WHITE timing down.

- When WHITE presses the button that time stops and BLACK starts to time down.

- When one of the players is out of time the display shows:

The code for the chess clock, **chessClock.C** is shown below.

```
1.  //chessClock.C by DW Smith 20-6-12
2.  #include <p18f1220.h>
3.  #pragma config WDT=OFF, OSC=INTIO2, PWRT=ON, LVP=OFF, MCLRE=OFF
4.  #include <delays.h>
5.  #include <dwsLCD.h>
6.  #pragma interrupt isr                    // declares isr as my interrupt (service) routine.

7.  // put variables here
8.  char secUnit=0, secTen=0, minUnit=0, minTen=0, WHITE=1, minutes=0;
9.  char WsecUnit=0, WsecTen=0, WminUnit=0, WminTen=0, BsecUnit=0, BsecTen=0,
    BminUnit=0, BminTen=0;

10. #pragma code isr=0x08
11. void isr(void)
12. {
13. while (PORTBbits.RB0 ==1);              //press
14. Delay100TCYx(8);                        //0.1 s
15. while (PORTBbits.RB0 ==0);              //release
16. Delay100TCYx(8);                        //0.1 s

17. if (WHITE==0) WHITE=1;
18. else WHITE=0;                           //toggle colour

19. INTCONbits.INT0IF=0;                    // clear PORTB,0 IF
20. }
21. #pragma code

22. // subroutines here
23. void Time(void)
24. {
25. if (WHITE==1)
26.    {
27. secUnit=WsecUnit;
28. secTen=WsecTen;
29. minUnit=WminUnit;
30. minTen=WminTen;
```

```
31.          }
32.       else
33.          {
34.       secUnit=BsecUnit;
35.       secTen=BsecTen;
36.       minUnit=BminUnit;
37.       minTen=BminTen;
38.          }
39.       if (secUnit!=0)
40.         {
41.          (secUnit=secUnit -1);
42.         }
43.       else
44.         {
45.              if (secTen!=0)
46.              {
47.              secTen=secTen−1;
48.              secUnit=9;
49.              }
50.              else
51.              {
52.                 if (minUnit!=0)
53.                 {
54.                 minUnit=minUnit−1;
55.                 secTen=5;
56.                 secUnit=9;
57.                 }
58.                 else
59.                 {
60.                    if(minTen!=0)
61.                    {
62.                    minUnit=9;
63.                    minTen=minTen−1;
64.                    secTen=5;
65.                    secUnit=9;
66.                    }
67.                    else
68.                       {
69.                       PORTBbits.RB4=1;              //buzzer on
70.                       }
71.              }
72.            }
73.         }
74.       if (WHITE==1)
75.         {
76.       WsecUnit=secUnit;
77.       WsecTen=secTen;
78.       WminUnit=minUnit;
79.       WminTen=minTen;
80.         }
81.       else
82.         {
```

```
83.     BsecUnit=secUnit;
84.     BsecTen=secTen;
85.     BminUnit=minUnit;
86.     BminTen=minTen;
87.       }
88.     }

89.     void main (void)
90.     {
91.     //SET UP
92.     // OSCCON defaults to 31 kHz. So no need to alter it.
93.     ADCON1=0x7F;              //all IO are digital or 0b01111111 in binary
94.     TRISA=0b00100000;         //sets PORTA
95.     PORTA=0b00000000;         //turns off PORTA outputs, not required, no outputs
96.     TRISB=0b00001111;         //sets PORTB
97.     PORTB=0b00000000;         //turns off PORTB outputs, good start position
98.     INTCON2bits.RBPU=0;       //pull ups on.
99.     T0CON=0b11010101;   //TMR0 on, 8bit timer, clocking on internal clock, prescaler/64.
100.    INTCON=0b10000000;        //sets GIE

101.    // this code configures the display
102.    WriteCmd (0x02);              // sets 4bit operation
103.    WriteCmd (FOUR_BIT & LINES_5X7);          // sets 5x7 and multiline operation.
104.    WriteCmd (CURSOR_BLINK);          // blinks cursor
105.    WriteCmd (CURSOR_RIGHT);          // moves cursor right

106.    Start:
107.    minutes=0;
108.    WsecUnit=0;
109.    WsecTen=0;
110.    WminUnit=0;
111.    WminTen=0;
112.    BsecUnit=0;
113.    BsecTen=0;
114.    BminUnit=0;
115.    BminTen=0;

116.    WriteCmd (CLEAR_SCREEN);
117.    WriteString("Press button");
118.    SetAddr (0xC0);
119.    WriteString("for time");

120.    // TMR0 is continually reset to 12 if it times out then 2 s has elapsed without sw.press.
121.    while (PORTBbits.RB0 ==1);          //press
122.    TMR0L=12;           //TMR0 overflows in 244 pulses ie 2 s.
123.    INTCONbits.TMR0IF=0;          // reset TMR0IF

124.    while(INTCONbits.TMR0IF==0) //wait until TMR0L=0, TMR0 interrupt flag=1
125.      {
126.    if (PORTBbits.RB0 ==0)//press
127.       {
128.    Delay100TCYx(8);          //0.1 s bounce
129.    TMR0L=12;
130.    while (PORTBbits.RB0 ==0); // wait for release
131.    Delay100TCYx(8);          //0.1 s
```

```
132.    minutes=minutes+5;              // add 5 minutes when switch pressed
133.    minTen=(int) minutes/10;
134.    minUnit=(int) minutes%10;
135.    WriteCmd (CLEAR_SCREEN);
136.    WriteChar(0x30 + minTen);
137.    WriteChar(0x30 + minUnit);
138.    WriteChar(':');
139.    WriteChar(0x30 + secTen);
140.    WriteChar(0x30 + secUnit);
141.      }
142.        }
143.    WriteString(" press to");
144.    SetAddr (0xC0);
145.    WriteString("start WHITE");
146.    while (PORTBbits.RB0 ==1);              //press
147.    Delay100TCYx(8);                        //0.1 s
148.    while (PORTBbits.RB0 ==0);              //release
149.    Delay100TCYx(8);                        //0.1 s

150.    WminTen=minTen;
151.    BminTen=minTen;
152.    WminUnit=minUnit;
153.    BminUnit=minUnit;

154.    WriteCmd (CLEAR_SCREEN);
155.    WriteString("WHITE");
156.    WriteChar(0x30 + WminTen);
157.    WriteChar(0x30 + WminUnit);
158.    WriteChar(':');
159.    WriteChar(0x30 + WsecTen);
160.    WriteChar(0x30 + WsecUnit);

161.    SetAddr         (0xC0);
162.    WriteString("BLACK");
163.    WriteChar(0x30 + BminTen);
164.    WriteChar(0x30 + BminUnit);
165.    WriteChar(':');
166.    WriteChar(0x30 + BsecTen);
167.    WriteChar(0x30 + BsecUnit);

168.    // timing, waiting for PORTB,0 interrupt.
169.    INTCONbits.INT0IF=0;                    // clear PORTB,0 IF
170.    INTCONbits.INT0IE=1;          //enable INT0IE interrupt on PortB,0.

171.    whiteTiming:                  //white=1
172.    TMR0L=134;                    //TMR0 overflows in 122 pulses ie 1 s.
173.    INTCONbits.TMR0IF=0;          // reset TMR0IF
174.    while(INTCONbits.TMR0IF==0);    //wait until TMR0L=0, TMR0 interrupt flag=1, had
                                          1sec
175.    Time();
176.    if (PORTBbits.RB4==1)               //set in Time() if time=0.
177.    {
178.    SetAddr (0x80);
179.    WriteString("White time up");
180.    SetAddr (0xC0);
```

```
181.  WriteString("Press to restart");
182.  INTCONbits.INT0IE=0;          //disable INT0IE interrupt on PortB,0.
183.  while (PORTBbits.RB0 ==1);          //press
184.  Delay100TCYx(8);          //0.1 s
185.  while (PORTBbits.RB0 ==0);          //release
186.  Delay100TCYx(8);          //0.1 s
187.  PORTBbits.RB4=0;
188.  goto          Start;
189.  }
190.  SetAddr (0x86);
191.  WriteChar(0x30 + WminTen);
192.  WriteChar(0x30 + WminUnit);
193.  WriteChar(':');
194.  WriteChar(0x30 + WsecTen);
195.  WriteChar(0x30 + WsecUnit);
196.  if (WHITE ==1) goto whiteTiming;
197.  else goto blackTiming;

198.  blackTiming:          //white=0
199.  TMR0L=134;          //TMR0 overflows in 122 pulses ie 1 s.
200.  INTCONbits.TMR0IF=0;          // reset TMR0IF
201.  while(INTCONbits.TMR0IF==0);          //wait until TMR0L=0, TMR0 interrupt flag=1,
                                           had 1sec
202.  Time();
203.  if (PORTBbits.RB4==1)          //set in Time() if time=0.
204.
205.  {
206.  SetAddr (0x80);
207.  WriteString("Black time up");
208.  SetAddr (0xC0);
209.  WriteString("Press to restart");
210.  INTCONbits.INT0IE=0;          //disable INT0IE interrupt on PortB,0.
211.  while (PORTBbits.RB0 ==1);          //press
212.  Delay100TCYx(8);          //0.1 s
213.  while (PORTBbits.RB0 ==0);          //release
214.  Delay100TCYx(8);          //0.1 s
215.  PORTBbits.RB4=0;
216.  goto          Start;
217.  }
218.  SetAddr (0xC6);
219.  WriteChar(0x30 + BminTen);
220.  WriteChar(0x30 + BminUnit);
221.  WriteChar(':');
222.  WriteChar(0x30 + BsecTen);
223.  WriteChar(0x30 + BsecUnit);
224.  if (WHITE ==0) goto blackTiming;
225.  else goto whiteTiming;
226.  }
```

Explanation of the Code

- An interrupt is used on PORTBbits.RB0 to interrupt the countdown timer when the button has been pressed to change color countdown. This routine

is declared on line 6 and is written on lines 10–21. Please refer to the chapter on interrupts, if required.
- Lines 8 and 9 declare the variables for the time and color.
- Lines 23–73 are the countdown code (see Project 1.) At the start the "color" time is substituted into the time counting down and at the end the times are changed back, lines 74–88.
- The setup includes setting up TMR0 control, line 99, and the interrupt control, line 100.
- Line 106 is a label, Start, so that the program can go back to there at the end to repeat.
- Lines 107–115, the variables are set to zero.
- Line 121 waits for the switch to be pressed.
- Lines 122–131 starts TMR0L off from a value of 12. It will overflow and set the interrupt flag, TMR0IF, in 244 pulses of 132 µs×64(prescaler line 99)=2 s. Every time the button is pressed 5 min and is added to the timer. If the button is not pressed within 2 s the time is set and displayed, lines 132–140.
- Lines 143–167 initialise and displays the times.
- Lines 168–197 start decrementing and displaying WHITE's timer and testing to see if it is zero (line 176). The PORTB change interrupt is set to interrupt if WHITE has moved and pressed the button to start BLACK's timer.
- Lines 198–226 do the same counting down and displaying BLACK's time.

Happy chess playing

Just for information these 226 lines of C code take up 1011 lines of code in the program memory. The 18F1220 has 2048 words of memory.

PROJECT 3: RESISTANCE METER

In this project a potential divider is used to measure the value of an unknown resistor, $R2$. The circuit diagram is shown in Figure 14.3.

The value of the voltage, V, across $R2$ is given by:

$$V = \frac{5R2}{R1 + R2}$$

$$V(R1 + R2) = 5R2$$

$$VR1 = R2(5 - V)$$

So

$$R2 = \frac{VR1}{5 - V}$$

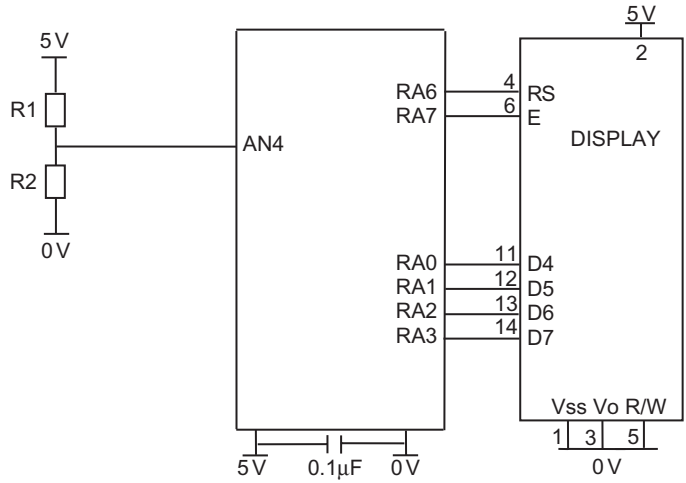

FIGURE 14.3 The resistance meter.

If we assume the value of $R1$ is 1 k and we measure the corresponding value of V on AN4 then we can calculate the value of $R2$. The code to achieve this, **resistanceMeter.C** is shown below.

```
1.  //resistanceMeter.C by DW Smith 4-7-12
2.  #include <p18f1320.h>
3.  #pragma config WDT=OFF, OSC=INTIO2, PWRT=ON, LVP=OFF, MCLRE=OFF
4.  #include <delays.h>
5.  #include <dwsLCD.h>

6.  // put variables here
7.  float v, R1=1000, R2, a;
8.  int READING;
9.  int tenThou, Thou, Hund, Tens, Units;

10. void main (void)
11. {
12. //SET UP
13. // OSCCON defaults to 31 kHz. So no need to alter it. 128 ms timing
14. ADCON1=0x6F;              //AN4 set, rest are digital or 0b01101111 in binary
15. TRISA=0b00100000;         //sets PORTA bit5 as input.
16. PORTA=0b00000000;         //turns off PORTA outputs, not required, no outputs
17. TRISB=0b00000000;         //sets PORTB as all outputs
18. PORTB=0b00000000;         //turns off PORTB outputs, good start position
19. ADCON0bits.ADON=1;          //turn on A/D
20. ADCON2=0b10000000;          //right justified, acquisition times are ok at 0 with 31 kHz

21. // this code configures the display
22. WriteCmd (0x02);              // sets 4bit operation
23. WriteCmd (FOUR_BIT & LINES_5X7);     // sets 5x7 and multiline operation.
24. WriteCmd (CURSOR_OFF);                //cursor off
25. WriteCmd (CURSOR_RIGHT);              // moves cursor right
```

```
26.   //selects CH.4 (100)
27.   ADCON0bits.CHS0=0;
28.   ADCON0bits.CHS1=0;
29.   ADCON0bits.CHS2=1;

30.   ADCON0bits.GO_DONE=1;                  // do A/D measurement
31.   while (ADCON0bits.GO_DONE == 1);       //wait until bit=0 for measurement completed
32.   READING=ADRESL+(ADRESH * 256);
33.   v=READING / 204.6;                     //5V=1023, 1V=204.6

34.   R2=v*R1/(5-v);              //if R2=12345.67 then

35.   tenThou=(int)R2/10000;      //(int)R2=12345          /10000=1.2345 then int only
36.   a=R2/1000;                  //a=12.34567
37.   Thou=(int)a%10;             //(int)a=12              %10=2
38.   a=R2/100;                   //a=123.4567
39.   Hund=(int)a%10;             /(int)a=123            %10=3
40.   a=R2/10;                    //a=1234.567
41.   Tens=(int)a%10;             //(int)a=1234            %10=4
42.   Units=(int)R2%10;           //(int)=12345            %10=5

43.   WriteCmd (CLEAR_SCREEN);
44.   WriteString("The Resistance ");
45.   SetAddr (0xC0);
46.   WriteString("is ");
47.   WriteChar(0x30 + tenThou);
48.   WriteChar(0x30 + Thou);
49.   WriteChar(0x30 + Hund);
50.   WriteChar(0x30 + Tens);
51.   WriteChar(0x30 + Units);
52.   WriteChar(0xDE);            // this is the omega sign.

53.   while(1);                   //stop
54.   }
```

Explanation of the Code

- Lines 26–33 select AN4 and make the voltage measurement, V.
- Line 34 obtains *R*2 from the potential divider equation.
- Lines 35–42 compute the values of the digits in *R*2.
- Lines 43–52 write the integer value of *R*2 to the display.
- Line 53 halts the program.

The accuracy of this reading depends on the tolerance of *R*1 and the accuracy of the 5 V supply. The 5 V supply would probably be derived from a 5 V regulator, such as the 7805, which has a tolerance of 5%. Using a voltage reference to supply the voltage to the micro and the potential divider circuit would increase the accuracy of the reading. For example an LM4040AIZ-5.0 a 5 V reference from National Semiconductor has a voltage tolerance of 5 mV and can deliver a current of 15 mA.

The resistance meter display is shown in Figure 14.4.

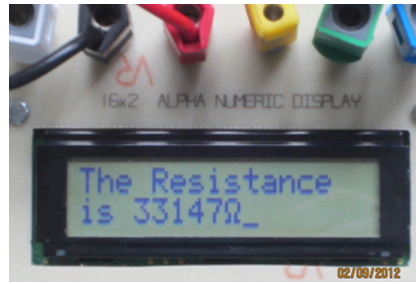

FIGURE 14.4 Resistance meter display.

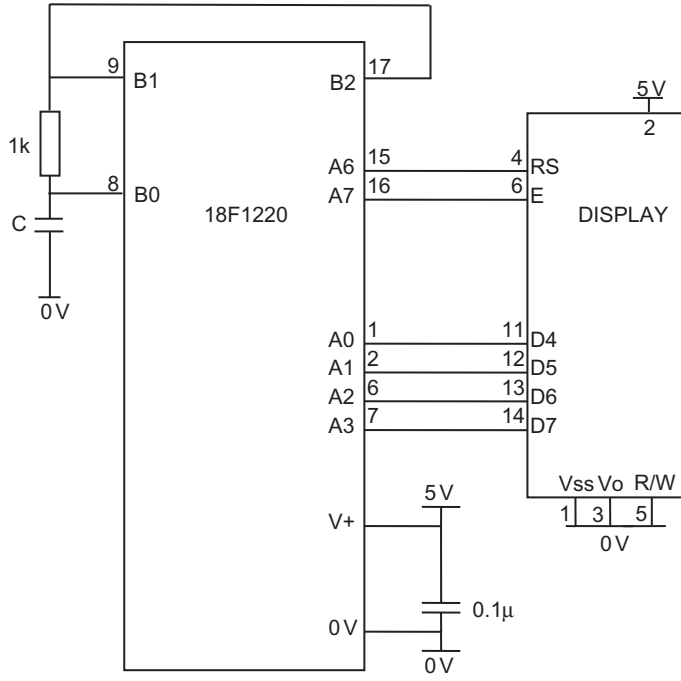

FIGURE 14.5 Capacitance meter.

PROJECT 4: CAPACITANCE METER

This project determines the value of a capacitor by charging it through a known resister. It also demonstrates how the C language can be used to do mathematics.

The circuit for this capacitance meter is shown in Figure 14.5.

When B2 is set, it puts out a voltage, a high but not necessarily 5V. This voltage, V_{max}, say is measured by AN5 on B1. The capacitor *C* then charges

through the 1 k resistor toward V_{max}. The voltage, V, on the capacitor after a time t is given by the equation:

$$V = V_{max}(1 - e^{-t/CR})$$

So to obtain the capacitor value, C.

$$\frac{V}{V_{max}} = (1 - e^{-t/CR})$$

$$e^{-t/CR} = 1 - \frac{V}{V_{max}}$$

$$-\frac{t}{CR} = \log\left(1 - \frac{V}{V_{max}}\right)$$

$$C = -\frac{t}{R(\log(1 - (V/V_{max})))}$$

The code to calculate the capacitor value and display it on the LCD is shown in **Capacitor.C**

```
1.  //Capacitor.C by DW Smith 18-6-12
2.  #include <p18f1320.h>
3.  #pragma config WDT=OFF, OSC=INTIO2, PWRT=ON, LVP=OFF, MCLRE=OFF
4.  #include <delays.h>
5.  #include math.h
6.  #include <dwsLCD.h>

7.  // put variables here
8.  float voltage, Vmax;
9.  float Capacitance,a,b,C;
10. int capTens, capUnits, capTenths;

11. void main (void)
12. {
13. //SET UP
14. OSCCON=0b01100100;      //4 MHz, 1 µs time
15. ADCON1=0x4F;            // 0b01001111 in binary ch 4/5
16. TRISA=0b00100000;       //sets PORTA bit5 as input
17. PORTA=0b00000000;       //turns off PORTA outputs, not required, no outputs
18. TRISB=0b00000011;       //sets PORTB bits 1 and 2 as inputs
19. PORTB=0b00000000;       //turns off PORTB outputs, good start position
20. T0CON=0b11000010;       //prescaler/8 8 µs time

21. ADCON0bits.ADON=1;      //turn on A/D
22. ADCON2=0b10100001;      //right justified, acquisition times=8TAD, osc/8 for Tad 1.6 µs
23. ADCON0bits.CHS2=1;      //selects AD CH4

24. ADCON0bits.CHS0=1;      //selects AD CH5
```

```
25.  TMR0L=131;                    //overflows after 125×8 µs=1 ms
26.  PORTBbits.RB2=1;

27.  ADCON0bits.GO_DONE=1; // do A/D measurement
28.  while (ADCON0bits.GO_DONE==1);      //wait until bit=0 for measurement completed
29.  Vmax=ADRESL+(ADRESH * 256);

30.  while (TMR0L!=0);                   //wait until TMR0L=0
31.  ADCON0bits.CHS0=0;                  //selects AD CH4
32.  ADCON0bits.GO_DONE=1;               // do A/D measurement
33.  while (ADCON0bits.GO_DONE == 1);    //wait until bit=0 for measurement completed
34.  voltage=ADRESL+(ADRESH * 256);

35.  PORTBbits.RB2=0;

36.        //Capacitance=−t/R(log(1−voltage/Vmax))
37.        //t/R=0.000001;   t=1 ms R=1 k

38.  a=(1−voltage/Vmax);
39.  b =log(a);
40.  //Capacitance=−0.000001/b

41.  //capacitance in microF=Capacitance * 1000000
42.  Capacitance=−1/b;          //in microF

43.  capTens=(int)Capacitance/10;
44.  capUnits=(int)Capacitance%10;
45.  Capacitance=Capacitance *10.0;
46.  capTenths=(int) Capacitance%10;

47.  OSCCON=0b00000000;          //31.25 kHz, 128 µs time

48.  // this code configures the display
49.  WriteCmd (0x02);                          // sets 4bit operation
50.  WriteCmd (FOUR_BIT & LINES_5X7);          // sets 5x7 and multiline operation.
51.  WriteCmd (CURSOR_OFF);                    //cursor off
52.  WriteCmd (CURSOR_RIGHT);                  // moves cursor right

53.  WriteCmd (CLEAR_SCREEN);
54.  WriteString("The capacitance");
55.  SetAddr (0xC0);
56.  WriteString("is ");
57.  WriteChar(0x30 + capTens);
58.  WriteChar(0x30 + capUnits);
59.  WriteChar(0x2E);              //decimal point
60.  WriteChar(0x30 + capTenths);
61.  WriteChar(0xEA);              //micro
62.  WriteChar('F');

63.  while(1);            //stop
64.  }
```

Explanation of Capacitor.C Code

- Line 5 includes the maths routine to calculate, log().
- Line 22 sets the acquisition time and the Tad times for the A/D convertor.

FIGURE 14.6 Capacitance meter display.

- Line 25 sets up TMR0 to 131 so that it overflows in 1 ms.
- Lines 27–29 measure the voltage, V_{max} on channel 5.
- Line 30 waits until TMR0 has overflowed in 1 ms.
- Lines 31–34 measures the capacitor voltage on channel 4.
- Lines 36–42 calculate the capacitor value in micro Farads.
- Lines 43–46 calculate the vales of tens, units, and tenths for the capacitor digits, i.e.,
  - If the capacitor value was 12.3 µF, (int)12.3/10=1, (int)12.3%10=2, 10xCapacitance=123.0 then (int) 123.0%10=3
- Lines 53–62 write the capacitor value to the LCD.

To achieve an accurate reading use a high tolerance 1 k resistor and supply the micro from a voltage reference chip, LM4040AIZ-5.0, rather than a voltage regulator.

The capacitance meter display is shown in Figure 14.6.

This project has demonstrated the "power" of the C language used in programming the micro. I would not have liked to calculate the value of C using assembly language!!

PROJECT 5: VOLTMETER

Figure 14.7 shows the microcontroller measuring a voltage on a potentiometer and displaying the result on an LCD. The voltage is displayed to one decimal place.

The voltmeter measures the input voltage on AN4 and displays the result on the LCD.

The code for the circuit is shown below in **voltageDisplay.C**

```
1.  //voltageDisplay.C DW Smith, 13th August 2012

2.  #include <p18f1220.h>
3.  #pragma config WDT=OFF, OSC=INTIO2, PWRT=ON, LVP=OFF, MCLRE=OFF
4.  #include <delays.h>
5.  #include <dwsLCD.h>
```

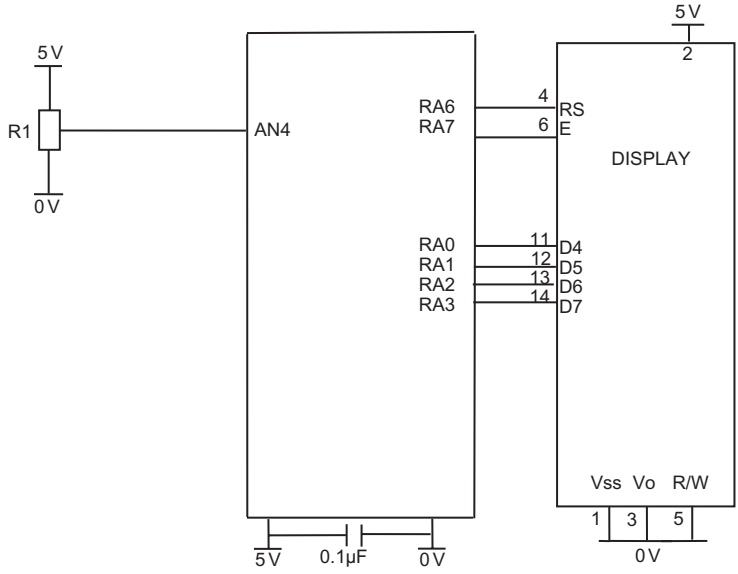

FIGURE 14.7 The voltmeter circuit.

```
6.  int READING;
7.  float VOLTAGE;          //Voltage is a decimal number not an integer.
8.  char voltUnit, voltTenths;

9.  void main (void)
10. {
11. //SET UP
12. // OSCCON defaults to 31 kHz. So no need to alter it.
13. ADCON1=0x6F;                //AN4 is analogue or 0b01101111 in binary
14. TRISA=0b00110000;           //sets PORTA
15. PORTA=0b00000000;           //turns off PORTA outputs
16. TRISB=0b00000001;           //sets PORTB
17. PORTB=0b00000000;           //turns off PORTB outputs, good start position
18. ADCON0bits.ADON=1;          //turn on A/D
19. ADCON2=0b10000000;          //right justified, acquisition times are ok at 0 with 31 kHz
20. ADCON0bits.CHS0=0;          //selects CH.4 (100)
21. ADCON0bits.CHS1=0;
22. ADCON0bits.CHS2=1;

23. // this code configures the display
24. WriteCmd (0x02);            // sets 4bit operation
25. WriteCmd (FOUR_BIT & LINES_5X7);    // sets 5x7 and multiline operation.
26. WriteCmd (CURSOR_RIGHT);            // moves cursor right
27. WriteCmd (CURSOR_OFF);              //cursor off);

28. WriteCmd (CLEAR_SCREEN);
29. WriteString("The voltage");
```

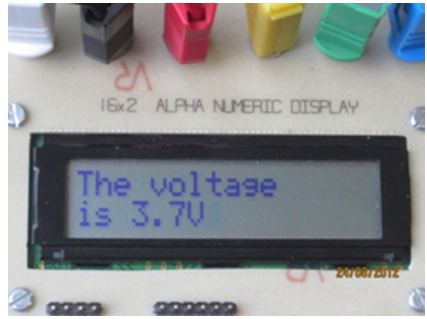

FIGURE 14.8 The voltmeter display.

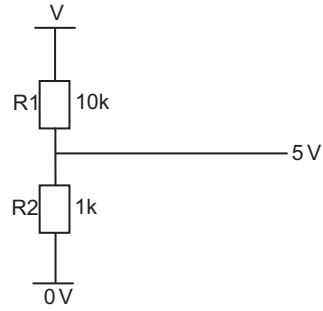

FIGURE 14.9 Voltage divider.

```
30.   SetAddr (0xC0);
31.   WriteString("is");

32.   while (1)
33.     {
34.   ADCON0bits.GO_DONE=1;                    // do A/D measurement
35.   while (ADCON0bits.GO_DONE == 1);         //wait until bit=0 for measurement completed
36.   READING=ADRESL+(ADRESH * 256);
37.   VOLTAGE=READING / 204.6;
38.   voltUnit=(int) VOLTAGE;
39.   voltTenths =VOLTAGE*10;
40.   voltTenths=(int) voltTenths %10;

41.   SetAddr(0xC3);
42.   WriteChar(0x30 + voltUnit);
43.   WriteChar('.');
44.   WriteChar(0x30 + voltTenths);
45.   WriteChar('V');
46.     }
47.   }
```

We have seen most of the code previously in other programs.

In line 37, 5V=Reading of 1023, the 10 bit A/D. So 1V=204.6.

Lines 38–40 convert the voltage, say 2.3 to units 2 and tenths 3.

The screen shot of the display is shown in Figure 14.8.

A potential divider is required to enable a voltage higher than 5V to be measured, to ensure that the voltage into the A/D input is a maximum of 5V.

For example with the circuit shown in Figure 14.9. The maximum applied voltage to give 5V input is $V = 5\dfrac{R1 + R2}{R2} = 55V$.

So this circuit would measure voltages up to 55V.