Objectives

A digital thermometer with a multiplexed 7-segment display using the EasyPIC 7 demo board and the LM35 temperature sensor which measures temperature range between 0 and 100 °C and outputs voltage from 0 V to 1 V. A potentiometer is used to calibrate the temperature in range between 0 and 100 °C. The thermometer has the option of displaying the temperature in Celsius or Fahrenheit, toggled using the a push button. Temperature range are represented with LEDs:

$0\,°C \le T_C < 20°C$, *Turn on LO LED*

$20°C \le T_C \le 30°C$, *Turn on MED LED*

$30°C < T_C \le 100°C$, *Turn on HI LED*

Program

```
#include <p18cxxx.h>
#include <BCDlib.h>
#include <delays.h>

#define Hi PORTBbits.RB7 // High temperature indicator
#define Med PORTBbits.RB6 // Medium temp. indicator
#define Lo PORTBbits.RB5 // Low temperature indicator
#define DegC FLAGS.B0 // DegC = 1, C else F
#define HiTemp 76 // A/D reading for T = 30 C
#define LoTemp 51 // A/D reading for T = 20 C

void Setup(void);
void Control(void);
void C_or_F(void);
void TestPB(void);
void InitMuxedDisplay(void);
void ISR(void);

char Digits[5];
unsigned char Qstates, val, i, j;
char SSCodes[] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07,
                  0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71, 0x00};
```

Figure 1: Set up parameters

```
24  void main(void) {
25      Setup();
26      while (1) {
27          Control(); // indicate hi, lo or medium temperature
28          C_or_F(); // use current A/D result
29          TestPB(); // start A/D & test PB
30      }
31  }
32
33  void Setup(void) {
34      ADCON0bits.ADON = 1; // turn on A/D, AN0
35      ADCON0bits.CHS = 7; // select AN7: take input from LM35 sensor
36      ADCON1bits.PVCFG = 0b10; // connect Vref+ to FVR
37      ADCON2 = 0b00001001; // LEFT justify (0b0), Tconv = 22us, Tacq = 4us
38      VREFCON0 = 0b10010000; // set Vref to 1.024V
39
40      ANSELB = 0x00; TRISB &= 0x1F; // RB7..RB5 outputs, RB0 input
41      INTCON2bits.INTEDG0 = 0; // RB0 reacts to falling edge
42      INTCON2bits.RBPU = 0; // enable PORTB internal pull-ups
43
44      DegC = 1; // start with Celsius
45      InitMuxedDisplay();
46  }
```

Figure 2: Main and Setup functions defined

```
48  void InitMuxedDisplay(void) {
49      TRISD = 0x00;                   // PORTD output
50      TRISA &= 0b11110000;            // RA3..RA0: output, rest: input
51      T0CON = 0b11010101;             // divide clock by 64, 8-bit mode
52      INTCONbits.T0IE = 1;            // enable interrupt
53      INTCONbits.GIE = 1;
54
55      i = 0;                          // start with digits[0]
56      Qstates = 0b00001000;           // start with MSD
57  }
58
59  void Control(void) {
60      PORTB &= 0b00011111; // reset all LEDs
61      if (ADRESH > HiTemp)
62          Hi = 1; // T > 30, turn on high LED
63      else if (ADRESH >= LoTemp)
64          Med = 1; // 20 <= T <= 30, turn on med. LED
65      else
66          Lo = 1; // else turn on low LED
67  }
```

```
69  void TestPB(void) { // C or F
70      ADCON0bits.GO = 1; // start AD conversion
71      if (INTCONbits.INT0IF) { // if RB0 pressed
72          INTCONbits.INT0IF = 0;
73          DegC = ~DegC; // toggle unit
74      }
75  }
76
77  void C_or_F(void) {
78      float Celsius,Fahrenheit; // use float +0.5 for rounding instead of +128
79      Celsius = 102.4 * ADRESH / 256.0; // convert AD output to Celsius
80      Fahrenheit = 18.0 * Celsius + 320.0;
81      if (DegC) {
82          Bin2Bcd((unsigned char)(Celsius + 0.5), Digits);
83      } else {
84          INTCONbits.GIE = 0; // disable interrupt while shifting digits
85          Bin2BcdE((unsigned int)(Fahrenheit + 0.5), Digits); // result = digits[1,2,3,4]
86          for (j=0; j<4; j++)
87              Digits[j] = Digits[j+1]; // shift result to digits[0,1,2,3]
88          INTCONbits.GIE = 1; // enable interrupt, resolved flickering
89      }
90  }
```

Figure 3: Digital Thermometer functionality and unit conversion defined

```
92   #pragma code ISR = 0x0008
93   #pragma interrupt ISR
94
95   void ISR(void) { // display value every 8ms
96       unsigned char val;
97       INTCONbits.TMR0IF = 0;          // acknowledge interrupt
98       TMR0L = 256 - 125;              // 64us * 125 = 8ms
99       PORTD = 0x00;                   // remove shadowing
100      PORTA = Qstates;                // display shows 1 digit at a time
101
102      val = SSCodes[Digits[i]];
103      if (DegC && i==3)
104          val = 0x63;                 // C degree symbol
105      else if (~DegC && i==2)
106          val |= 0x80;                // DP for F
107      PORTD = val;
108
109      Qstates >>= 1;                  // move to next transistor
110      i++;                            // display next digit of Digits
111      if (i == 4){
112          i = 0;
113          Qstates = 0b00001000;       // back to MSD
114      }
115  }
```

Figure 4: Interrupt setup to display value on multiplex 7-segment display

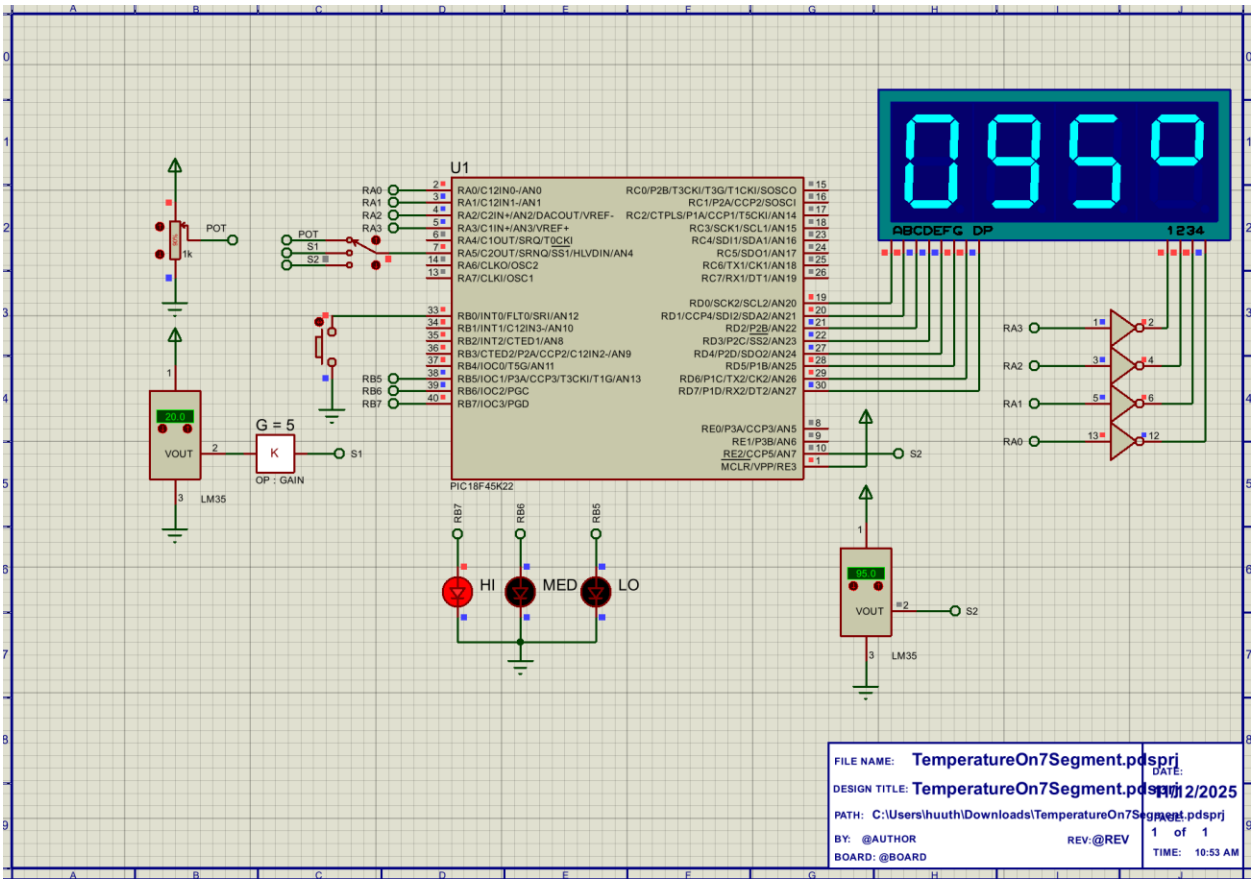

Figure 5: Proteus simulation