```
// PIC16F877A Configuration Bit Settings

// 'C' source line config statements

// CONFIG
#pragma config FOSC = EXTRC     // Oscillator Selection bits (RC oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = OFF      // Brown-out Reset Enable bit (BOR disabled)
#pragma config LVP = OFF        // Low-Voltage (Single-Supply) In-Circuit Serial
#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit (Data E
#pragma config WRT = OFF        // Flash Program Memory Write Enable bits (Write
#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>

//defining clock frequency for the PIC controller
#define _XTAL_FREQ 6000000

//function declaration
void init(void);
void LCD_command(unsigned char);
void LCD_data(unsigned char);
void LCD_output(unsigned int);
```

```
//variable initialization
unsigned char L_value, H_value;
long value, voltage;
unsigned char num_arr[10]; // for storing LCD_output values

void main() {
    init();
    while (1) {
        ADCON0 = 0x81;
        ADCON0 |= 0x04; // starting ADC conversion
        while (ADCON0 & 0x04); // checking GO_done bit for 0 ,indication for ADC_over status
        L_value = ADRESL; //2 parts of the ADC conversion is stored in as right justified.all 8 bits are occupied
        H_value = ADRESH; //MSB 2 bits will be stored in ADRESH
        value = ((unsigned int) H_value << 8)+(unsigned int) L_value; // total ADC value is adjusted using shift operator and s
        voltage = (value * 337) / 1023; //calibration to user defined value.'337'- MAX value to be displayed.
        LCD_command(0X80);
        if (voltage == 0) LCD_data(0x30);
        else LCD_output(voltage);
        LCD_data(0x56); //ASCII value for V
        LCD_data(0x20);
        LCD_data(0x20);
        CCPR1L = (voltage) >> 2;
        CCP1CON = (CCP1CON & 0x0C)+((voltage & 0x03) << 4);
        __delay_ms(1000);
    }

}
```

```
void init(void) {
    //PIC_ADC initialization
    TRISA = 0x01; //0000 0001 setting RA0 as input - AN0 channel for ADC
    //bit 0(1) - ADC power ON
    //bit 3 to 5 (000)- selecting channel AN0
    //bit 6,7(10)- for choosing fosc/32
    ADCON0 = 0x81; //1000 0001
    //bit 3to0(1110)- AN0 alone as analog input other channel as digital
    //bit 6(0)- for choosing fosc/32
    //bit 7(1)-1 = Right justified. Six (6) Most Significant bits of ADRESH are read as ?0?.
    ADCON1 = 0x8E; //1000 1110
    //PWM initialization
    CCP1CON = 0x3C; //00111100 //3rd and 2nd bit as to be HIGH for enabling PWM mode
    //4th and 5th bit are dedicated for storing LSB data of CCPR1 register(duty cycle)
    T2CON = 0x06;
    //PIC controller timer module T2CON is used as timing reference for PWM
    //2nd bit has to be HIGH for enabling timer and bit 1,0 has to be set according to the required prescale value
    //bit 1 is set HIGH for 1:16 pre-scaler
    PR2 = 0x5E;
    //PR2 register is for setting the total duration of one period(PWM)
    //calculation: (FOSC/(4*pwm_freq*TIMER pre-scale value))-1
    //-1 is optional could be used or unused according to the practical results
    //PR2 = (6000000/(4*1000*16))= 94(0x5E)
    //PIC initialization for LCD
    TRISC = 0x00; // setting port C as output //RS,Enable pin
    TRISD = 0x00; // setting port D as output// data pins
    //LCD initialization
    LCD_command(0x30);
    __delay_ms(50);
```

```
    LCD_command(0x38); // function set (number of lines and 5*7 matrix)
    while (RD7 == 1); //checking busy flag connected D7 bit for ready condition
    LCD_command(0x08); //display OFF
    while (RD7 == 1); //checking busy flag connected D7 bit for ready condition
    LCD_command(0x01); //clear display
    while (RD7 == 1); //checking busy flag connected D7 bit for ready condition
    LCD_command(0x06); //entry mode
    while (RD7 == 1); //checking busy flag connected D7 bit for ready condition
    LCD_command(0x0E); //display ON,cursor display ON
    while (RD7 == 1);
}

void LCD_output(unsigned int n) {
    unsigned char result, i = 0;
    while (n != 0) {
        result = n - ((n / 10)*10);
        num_arr[i] = result;
        i++;
        n /= 10;
    }
    num_arr[i] = '\0';

    i -= 1;
    for (int j = i; j >= 0; j--) {
        LCD_data(0x30 + num_arr[j]);
    }
}
```

```
//RS pin - LOW (for sending command)-connected to RC3)
//Enable pin to go high to low(connected to RC0)

void LCD_command(unsigned char hex) {
    PORTC &= 0xF7; //1111 0111(mask bit for AND)clearing RC3
    PORTD = hex;
    PORTC |= 0x01; //0000 0001(mask bit for OR)setting RC0
    PORTC &= ~0x01; //1111 1110(mask bit for AND)clearing RC0
    __delay_ms(100);
}

//RS pin - HIGH (for sending data)-connected to RC3)
//Enable pin to go high to low(connected to RC0)

void LCD_data(unsigned char hex) {
    PORTC |= 0x08; //0000 1000(mask bit for OR)setting RC3
    PORTD = hex;
    PORTC |= 0x01; //0000 0001(mask bit for OR)setting RC0
    PORTC &= ~0x01; //1111 1110(mask bit for AND)clearing RC0
    __delay_ms(100);
}
```