09 Write an Embedded C program for PIC Interfacing Temperature Sensor (LM35) and Display the current Temperature on LCD Display. (Answer sheet : draw interfacing diagram & write code) (Simulate using Proteous & Perform on PIC18 Kit)

#include <xc.h>

//#include <pic18f4520.h>

//Configuration bit setting//

//#pragma config OSC = HS //Oscillator Selection

#pragma config WDT = OFF //Disable Watchdog timer

#pragma config LVP = OFF //Disable Low Voltage Programming

#pragma config PBADEN = OFF //Disable PORTB Analog inputs

//Declarations for LCD Connection

#define LCD\_DATA PORTD //LCD data port

#define en PORTEbits.RE2 // enable signal

#define rw PORTEbits.RE1 // read/write signal

#define rs PORTEbits.RE0 // register select signal

//Function Prototypes

void ADC\_Init(void); //Function to initialize the ADC

unsigned int Get\_ADC\_Result(void); //Function to Get ADC result after conversion

void Start\_Conversion(void); //Function to Start of Conversion

void msdelay (unsigned int time); //Function to generate delay

void init\_LCD(void); //Function to initialise the LCD

void LCD\_command(unsigned char cmd); //Function to pass command to the LCD

void LCD\_data(unsigned char data); //Function to write character to the LCD

void LCD\_write\_string( char \*str);//Function to write string to the LCD

//Start of main program

void main()

{

char msg1[] = "LM35 Interface";

char msg2[] = "Temp.:";

char msg3[] = {0xDF, 0x43, 0x00};

unsigned char temp=0;

unsigned char i=0, Thousands,Hundreds,Tens,Ones;

unsigned int adc\_val;

unsigned char val, pot0[6];

ADCON1 = 0x0F; //Configuring the PORTE pins as digital I/O

TRISD = 0x00; //Configuring PORTD as output

TRISE = 0x00; //Configuring PORTE as output

ADC\_Init(); // Init ADC peripheral

init\_LCD(); // Init LCD Module

LCD\_write\_string(msg1); // Display Welcome Message

LCD\_command(0xC0); // Goto second line, 0th place of LCD

LCD\_write\_string(msg2); // Display Message "Temp:"

while(1)

{

Start\_Conversion(); //Trigger conversion

adc\_val= Get\_ADC\_Result();//Get the ADC output by polling GO bit

adc\_val = adc\_val/2; //Divide the value by 2 match with 10mv stepsize

LCD\_command (0xC7); //Goto 8th place on second line of LCD

val = (unsigned char) adc\_val;

i = (val/100); //Get the Hundreds place

Hundreds = i + 0x30; // Convert it to ASCII

LCD\_data (Hundreds); //Display Hundreds place

i = (val%100)/10; //Get the Tens place:remainder/10

Tens = i + 0x30; // Convert it to ASCII

LCD\_data (Tens); //Display Tens place

i = adc\_val%10 ; //Get the Ones place

Ones = i + 30; // Convert it to ASCII

LCD\_data (i + 0x30); //Display Ones place

LCD\_write\_string(msg3);

msdelay(300); //Delay between conversions.

}

}

//Function Definitions

void ADC\_Init()

{

ADCON0=0b00000100; //A/D Module is OFF and Channel 1 is selected

ADCON1=0b00001110; // Reference as VDD & VSS, AN0 set as analog pins

ADCON2=0b10001110; // Result is right Justified

//Acquisition Time 2TAD

//ADC Clk FOSC/64

ADCON0bits.ADON=1; //Turn ON ADC module

}

void Start\_Conversion()

{

ADCON0bits.GO=1;

}

//If you do not wish to use adc conversion interrupt you can use this

//to do conversion manually. It assumes conversion format is right adjusted

unsigned int Get\_ADC\_Result()

{

unsigned int ADC\_Result=0;

ADC\_Result = ADRESH ; /\* Copy the 2 bits in ADRESH to ADC\_Value\*/

ADC\_Result = ADC\_Result <<8; /\* Shift it 8 bits to left�������������� \*/

ADC\_Result = ADC\_Result +ADRESL; /\* Add left shifted value to ADRESL������\*/

return ADC\_Result;

}

void msdelay (unsigned int time) //Function to generate delay

{

unsigned int i, j;

for (i = 0; i < time; i++)

for (j = 0; j < 275; j++);//Calibrated for a 1 ms delay in MPLAB

}

void init\_LCD(void) // Function to initialise the LCD

{

LCD\_command(0x38); // initialization of 16X2 LCD in 8bit mode

msdelay(15);

LCD\_command(0x01); // clear LCD

msdelay(15);

LCD\_command(0x0C); // cursor off

msdelay(15);

LCD\_command(0x80); // go to first line and 0th position

msdelay(15);

}

void LCD\_command(unsigned char cmd) //Function to pass command to the LCD

{

LCD\_DATA = cmd; //Send data on LCD data bus

rs = 0; //RS = 0 since command to LCD

rw = 0; //RW = 0 since writing to LCD

en = 1; //Generate High to low pulse on EN

msdelay(15);

en = 0;

}

void LCD\_data(unsigned char data)//Function to write data to the LCD

{

LCD\_DATA = data; //Send data on LCD data bus

rs = 1; //RS = 1 since data to LCD

rw = 0; //RW = 0 since writing to LCD

en = 1; //Generate High to low pulse on EN

msdelay(15);

en = 0;

}

//Function to write string to LCD

void LCD\_write\_string(char \*str)

{

int i = 0;

while (str[i] != 0)

{

LCD\_data(str[i]); // sending data on LCD byte by byte

msdelay(15);

i++;

}

}

