```
/* Hierbei soll mit Hilfe des Temperatursensors LM35 DZ die
 * Umgebungstemperatur gemessen und an einem LCD Display (GE_1602 B_T)
 * angezeigt werden.
 * Maximale temperatur: Tmax=100°C=1000mV*1.024/5.000=204,8bits.
 * Anzuzeigende Spannung: Vmax=204,8*(4.000/1024+1.000/1024) bzw.
 * Vmax=204,8*125*(1/32+1/128)
 * RA0 analog Input; PORTB gigital outputs
 * Fosc=8MHz (internal OSC.)/Instruction circe=0,5µS
 * ADC selected conversion clock: Fosc/16=0,5MHz ie. T=2µsec
 * Laut Data sheet: TADmin = 7,45µsec
 * Requiered TAD=TADmin/2=3,725.
 * Selected TAD: ADCON2bits.ACQT = 010 = 4TAD
 * Author: lasaros Goumas
 * Created on 27. Januar 2021
 * */


/* Includes

**********************************************************************
**/
#include <xc.h>
#include <p18cxxx.h>                //PIC 18F25K22 Controller

/*Configuration

**********************************************************************
***/
#pragma config FOSC = INTIO67   //Internal oscillator block
#pragma config BORV = 190       //VBOR set to 1.90 V nominal
#pragma config WDTEN = OFF
#pragma config LVP = OFF
#pragma config PBADEN = OFF     //PORTB<5:0> digital I/O on reset
#pragma config CP0 = OFF        // Code Protection Block 0
#pragma config EBTRB = OFF      // Boot Block Table Read Protection
#pragma config WRT0 = OFF       // Write Protection Block 0
#pragma config WRTD = OFF       // Data EEPROM Write Protection bit
#pragma config EBTR0 = OFF      // Table Read Protection Block 0
#pragma config CPD = OFF


/*Declarations

**********************************************************************
**/
#define _XTAL_FREQ 8000000      // Fosc  frequency for _delay() library
#define LCD_RS PORTBbits.RB4    //High: Data ; Low: Instruction code
#define LCD_E PORTBbits.RB5     //High: Chip enable
#define LCD_DATA PORTB          //PORTB ist Datenport für das display
unsigned int result;            //ADC Ergebni
unsigned int result125;         //ADC ergebnis*125
unsigned int result_32;         //ADC ergebnis*125/32
unsigned int lcd_info;          //DATA to be send to LCD
unsigned char temp;             //Temporäres LCD Register
int hund;                       //Wertigkeit 100
int zehn;                       //Wertigkeit 10
int eins;                       //Wertigkeit einer = 1ste Komma Stelle
```

```
int count;                          //Allgemeines zähler register
const char *pnt;                    //String pointer


/*Funktions

*************************************************************************
/

void init_PIC (void){
    LATB = 0x00;                    //Clear all RB output latches
    TRISB = 0x00;                   //RB Pins sind Ausgänge
    LATA = 0x00;                    //Clear all RA output latches
    TRISA = 0x01;                   //RA0 ist Eingang
    ANSELAbits.ANSA0 = 1;           //AN0 Analogeingang
    OSCCON = 0b01100100;            //8Mhz interner Oscillator
    T0CON = 0b11000111;             //8 Bit TMR0 enabled; prescaler 1:256
    ADCON0bits.ADON = 0;             //ADC depowered
}


void ADC (void){
    ADCON2=0b10010101;              //Right justified; 4 TAD; Fosc/16
    ADCON0bits.ADON=1;              //ADC powered
    __delay_us(20);                 //20µsec warten
    ADCON0bits.GO = 1;              //ADC in progress
    while(ADCON0bits.NOT_DONE);     //Warte auf Conversion ende
    result = ADRES;                 //ADC Ergebnis
    ADCON0bits.ADON = 0;            //ADC depowered
}


void write_command (void){
    temp=lcd_info;
    temp=(temp<<4 | temp>>4);        //Swab the nibbles around
    temp=temp & 0x0F;                 //High nibbles of temp ausmaskiert
    LCD_DATA=temp;                   //High nibbles of lcd_info an PORTB
    LCD_E = 1;                       //LCD enabled
    LCD_E = 0;                       //High Nibble an LCD übergeben
    __delay_ms(2);                  //2msec warten
    temp=lcd_info;
    temp=temp & 0x0F;                 //High nibbles of temp ausmaskiert
    LCD_DATA =temp;                  //Low nibbles of lcd_info an PORTB
    LCD_E = 1;                       //LCD enabled
    LCD_E = 0;                       //Low Nibble an LCD übergeben
    __delay_ms(2);                   //2msec warten
}


void write_data (void){
    temp=lcd_info;
    temp=(temp<<4 | temp>>4);        //Swab the nibbles around
    temp=temp & 0x0F;                 //High nibbles of temp ausmaskiert
    LCD_DATA=temp;                   //High nibbles of lcd_info an PORTB
    LCD_RS=0x01;                     //Write data
    LCD_E = 1;                       //LCD enabled;
    LCD_E = 0;                       //High Nibble an LCD übergeben
```

```
    __delay_ms(2);                  //2msec warten
    temp=lcd_info;
    temp=temp & 0x0F;                //High nibbles of temp ausmaskiert
    LCD_DATA =temp;                 //Low nibbles of lcd_info an PORTB
    LCD_RS=0x01;                    //Write data
    LCD_E = 1;                      //LCD enabled
    LCD_E = 0;                      //Low Nibble an LCD übergeben
    __delay_ms(2);                  //2msec warten
}


void writeString (const char *pnt){
    while (*pnt)
    {
        lcd_info = *pnt;
        write_data();
        *pnt++;
    }
}


/*Main Routine

**************************************************************************
/
void main(void){
    init_PIC ();

    LCD_RS=0;               //Init LCD
    LCD_E=0;
    __delay_ms(50);         //50msec warten
    lcd_info = (0x30);       //LCD in 8bit Betrieb
    write_command();
    lcd_info = (0x30);       //LCD in 8bit Betrieb
    write_command();
    lcd_info = (0x32);
    write_command();
    lcd_info = (0x2C);
    write_command();        //Funktion set (4bit,2 zeilen, 5x8)
    lcd_info = (0x06);
    write_command();        //Display ON/OFF (Diplay;Cursor und Blink
=aus)
    lcd_info = (0x0C);
    write_command();        //Entry mode (DD-RAM Increment;Cursor
schieben)
    lcd_info = (0x01);
    write_command();        //Clear screen
    lcd_info = (0x02);
    write_command();        //Return cursor to home position


    for(;;)                                     //Endless loop
    {
    ADC ();                                     //ADC starten
    result125 = ADRES*125;
    result_32 = result125>>5;
    lcd_info = (result_32)+(result_32>>2);      //Anzuzeigender wert
```

```
hund = 0;                                       //hunderter Wertigkeit
while (1)
if (lcd_info>=100){
    ++hund;
    lcd_info = lcd_info-100;
}
else{
    hund = hund+0x30;                           //Hunderter in ASCII
    break;
}

zehn = 0;                                       //zehner Wertigkeit
while (1)
if (lcd_info>=10){
    ++zehn;
    lcd_info = lcd_info-10;
}
else{
    zehn = zehn+0x30;                           //Zehner in ASCII
    break;
}

eins = 0;
while (1)
if (lcd_info>=1){
    ++eins;
    lcd_info = lcd_info-1;
}
else{
    eins = eins+0x30;                           //Einer in ASCII
    break;
}

writeString ("Raum Temperatur");    //Text und Werte anzeigen
lcd_info = (0xC0);
write_command();             //Zweite Zeile
lcd_info = (0x20);
write_data();               //Leerzeichen
lcd_info = (0x20);
write_data();               //Leerzeichen
lcd_info = (0x74);
write_data();               //t
lcd_info = ('=');
write_data();               //=
lcd_info = (hund);
write_data();               //Hunderter
lcd_info = (zehn);
write_data();               //Zehner
lcd_info = (',');
write_data();               //,
lcd_info = (eins);
write_data();               //Einser
lcd_info = (0x20);
write_data();               //Leerzeichen
lcd_info = (0x5B);
write_data();               //[
```

```
    lcd_info = (0x27);
    write_data();               //'
    lcd_info = (0x43);
    write_data();               //C
    lcd_info = (0x5D);
    write_data();               //]
    lcd_info = (0x02);
    write_command();           //Return cursor to home position

    for (count=0; count<=4; count++)__delay_ms(250);  //Warte 1Sec
    }
}
```