```
/*Hierbei soll die Kapazität eines Kondesators von 100nf bis 100µf
 * ermittelt werden.
 * Das Ergebnis soll an einem SPI LCD display angezeigt werden.
 * RC<1:0> Eingänge; Rest Ausgänge
 * PORTA Pins sind Ausgänge
 * Fosc=8MHz (internal OSC.)/Instruction circe=0,5µS
 * TMR0 overflow auf 100µSec eingestellt.
 * Timer startet nach 2 Endladungszeitkonstanten des Prüflings.
 * File:   Captest.c
 * Author: lasaros Goumas
 * Created on 05. Februar 2021, 20:4845
 */


/* Includes

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


/*Configuration

************************************************************************
***/
#pragma config FOSC = INTIO67   //Internal oscillator block
#pragma config PWRTEN = ON      //Power up timer enabled
#pragma config WDTEN = OFF      //Watch dog timer OFF
#pragma config PBADEN = OFF
#pragma config LVP = OFF
#pragma config CP0 = OFF        // No Code Protection Block 0
#pragma config CPB = OFF      // Boot Block not protected
#pragma config WRT0 = OFF       //No Write Protection Block 0
#pragma config WRTD = OFF       // Data EEPROM Write Protection bit
#pragma config EBTR0 = OFF      // Table Read Protection Block 0
#pragma config EBTRB = OFF


/*Declarations

************************************************************************
**/
#define _XTAL_FREQ 8000000      // Fosc  frequency for _delay() library
#define RS PORTAbits.RA3        //Read/write command
#define CSB PORTAbits.RA4       //Display activation
#define startendl PORTBbits.RB0
#define stopptim PORTCbits.RC1
#define LCD_RESET PORTCbits.RC2
#define SI PORTCbits.RC3        //Display data
#define CLK PORTCbits.RC4       //Display Clock
#define SB PORTCbits.RC5
#define SA PORTCbits.RC6
unsigned int display_store;     //LCD Eingangsspeicherregister
unsigned int counter;           //Allgemeiner Zähler
unsigned int capwert;           //Kapazitätswert
unsigned int timeover;         //TMRO überlauf
unsigned char hund;             //Hunderter wertigkeit
unsigned char zehner;           //Zehner wertigkeit
unsigned char einer;            //Einer wertigkeit
const char *pnt;                //String pointer


/*Funktionen

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

void init_PIC (void){
    TRISA = 0x00;               //PORTA pins sind Ausgänge
    LATA = 0x00;
    TRISB = 0b00000001;         //RB0 ist eingang
    ANSELB = 0x00;              //PORTB digital inputs
    TRISC = 0b00000011;         //RC<1:0> Pin Eingänge; Rest Ausgänge
    ANSELC = 0x00;              //RC<7:2> digital outputs
    LCD_RESET = 0x01;           //Initiate LCD
    display_store = 0x00;
    counter = 0x00;
    capwert = 0x00;
    timeover = 0x00;
    OSCCON = 0b01100111;        //8 MHz internal oscillator stable
    T0CON = 0b11001000;         //8bit TMRO enabled; Prescaler bypassed
    INTCON2bits.TMR0IP = 1;     //High priority TMR0 overflow
    RCONbits.IPEN = 1;          //Interrupt priority enabled
    INTCONbits.GIEH = 0;        //GIE disabled
    SB = 0x00;                  //SB "OFF"
    NOP();                      //Propagation delay 500nsec
    SA = 0x01;                  //SA "ON"
    for (counter=0; counter<=8; counter++)__delay_ms(1);  //Warte 1Sec
(125)!!!)
}


    void __interrupt()_High_Prio (void){
    if (TMR0IE && TMR0IF)           //Timer 0 interrupt?
        INTCONbits.TMR0IF = 0;      //TMR0 overflow flag löschen
    capwert++;
    timeover++;
    TMR0L = 0x38;
}


void write_command (void){
    PORTAbits.RA4=0;                    //Display aktivieren (CBS)
    PORTAbits.RA3=0;                    //Register selector (RS)in
command.
    PORTCbits.RC4=0;                    //Takt deaktivieren (CLK)
    counter=0x08;                       //Es werden 8 bits übertragen
    display_store=display_store<<1;     //Zwischenspeicher nach links
    while (1)
    if (counter>0)
    {
    if (display_store>=0x0100)
    {
```

```
    PORTCbits.RC3=1;                    //Dateneingang (SI) aktivieren
    NOP();
    PORTCbits.RC4=1;                    //Daten mit CLK übernehmen
    NOP();
    PORTCbits.RC4=0;                    //Takt deaktivieren (CLK)
    display_store=display_store-0x0100; //Highbyte display_store löschen
    display_store=display_store<<1;
    --counter;
    }
    else
    {
    PORTCbits.RC3=0;                    //Dateneingang (SI) deaktivieren
    NOP();
    PORTCbits.RC4=1;                    //Daten mit CLK übernehmen
    NOP();
    PORTCbits.RC4=0;                    //Takt deaktivieren (CLK)
    display_store=display_store<<1;
    --counter;
    }
    }
    else
    {
    PORTCbits.RC4=0;                    //Dateneingang (SI) deaktivieren
    PORTAbits.RA3=0;                    //Register selector (RS)in
command.
    break;
    }
    __delay_us(30);
    }


void write_data (void){
    PORTAbits.RA4=0;                    //Display aktivieren (CBS)
    PORTAbits.RA3=1;                    //Register selector (RS)in data.
    PORTCbits.RC4=0;                    //Takt deaktivieren (CLK)
    counter=0x08;                       //Es werden 8 bits übertragen.
    display_store=display_store<<1;     //Zwischenspeicher nach links
    while (1)
    if (counter>0)
    {
    if (display_store>=0x0100)
    {
    PORTCbits.RC3=1;                    //Dateneingand (SI) aktivieren
    NOP();

    PORTCbits.RC4=1;                    //Daten mit CLK übernehmen
    NOP();
    PORTCbits.RC4=0;                    //Takt deaktivieren (CLK)
    display_store=display_store-0x0100;  //Highbyte display_store
löschen
    display_store=display_store<<1;
    --counter;
    }
    else
    {
     PORTCbits.RC3=0;                   //Dateneingang (SI)
     NOP();
     PORTCbits.RC4=1;                  //CLK aktiv
     NOP();
     PORTCbits.RC4=0;                 //Takt deaktivieren
     display_store=display_store<<1;
     --counter;
    }
    }
    else
    {
    PORTCbits.RC3=0;                  //Dateneingang (SI) deaktivieren
    PORTAbits.RA3=0;                  //Register selektor (RS) in
command
    break;
    }
    __delay_us(30);
}


void init_LCD (void){
    __delay_ms(750);                  //Warte 750ms
    display_store=0x38;               //Display initialisieren
    write_command ();                 //Function set
    display_store=0x39;
    write_command ();                 //Function set
    display_store=0x14;
    write_command ();                 //Bias set
    display_store=0x78;
    write_command ();                 //Contrast set
    display_store=0x52;
    write_command ();                 //Power/ICON/Contrast control
    display_store=0x69;
    write_command ();                 //Follower control
    __delay_ms(20);                  //Warte 200msec!!!!!!
    display_store=0x0C;
    write_command ();                 //Display ON/OFF control
    display_store=0x01;
    write_command ();                 //Clear display
     __delay_ms(2);                   //Warte 2msec
    display_store=0x06;
    write_command ();                 //Entry mode set
    __delay_ms(20);
}


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


void restart (void){
    display_store=0x80;
    write_command ();                 //Position in Zeile 1 (=0x80+
0x00)
    display_store=0x20;
    write_data ();                      //Leerzeichen 1 in Zeile 1
    writeString (" OVERFLOW!!!");            //Text und Werte anzeigen
    display_store=0xC0;
    write_command ();                   //Position 1 in Zeile 2 (=0x80+
0x40)
    display_store=0x20;
    write_data ();                      //Leerzeichen 1 in Zeile 2
    writeString (" Press RESTART");           //Text und Werte anzeigen
}


/*Main Routine

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

restart:    while (1)
            if(startendl==0x00){
            init_LCD();                     //LCD iitialisierung
            display_store=0x80;
            write_command ();               //Position in Zeile 1 (=0x80
+0x00)
            display_store=0x20;
            write_data ();                  //Leerzeichen 1 in Zeile 1
            writeString ("    Press      ");//Text und Werte anzeigen
            display_store=0xC0;
            write_command ();               //Position 1 in Zeile 2 (=
0x80+0x40)
            display_store=0x20;
            write_data ();                  //Leerzeichen 1 in Zeile 2
            writeString ("    START     "); //Text und Werte anzeigen
            display_store = 0x02;
            write_command();                //Return cursor to home
position
            }
            else{
                break;
            }

    __delay_ms(20);                 //Starttasten endprellung
    SA = 0x00;                      //SA "OFF"
    NOP();                          //Propagation delay 500nsec
    SB = 0x01;                      //Enladung in progress
    timeover = 0;
    TMR0L = 0x38;                   //TMR0 overflow: [4*200]/8=100µsec
    INTCONbits.TMR0IF = 0;          //TMR0 overflow interrupt cleared
    INTCONbits.TMR0IE = 1;          //TMR0 overflow interrupt enabled
    INTCONbits.GIEH = 1;            //Global interrupt enabled
    ei ();

    while (1){
    while (timeover ==0);           //Warte auf TMR0 überlauf
    if (stopptim ==1) break;         //Endladung abgeschlossen
    else{
        timeover = 0;
    }
    }
    INTCONbits.TMR0IE = 0;          //TMR0 overflow interrupt disabled
    INTCONbits.TMR0IF = 0;          //TMR0 overflow interrupt cleared
    INTCONbits.GIEH = 0;            //All interrupts disabled
    SB = 0x00;                      //Stopp Endladung
    NOP();                          //Propagation delay 500nsec
    SA = 0x01;                      //Aufladungsbeginn

    if (capwert>=0x07D0) goto stopp;
    else goto data;


stopp:  for(;;){                            //Überlaufsanzeig
        if(PORTCbits.RC0 == 0){
            restart ();
        }
        else{
        LCD_RESET = 0x00;           //Clear LCD
        for (counter=0; counter<=4; counter++)__delay_ms(25); //Warte
100msec
        LCD_RESET = 0x01;           //Initiate LCD
            break;
        }
    }

data:   display_store = capwert;    //Kapazitätswert in ASCII anzeigen
        hund = 0x00;
        while (1)
        if (display_store>=0xC8)   //Capwert>=200
            {
            ++hund;
            display_store = display_store - 0xC8;
            }
        else{
            hund = hund+0x30;       //Hunderte Wertigkeit in ASCII
            break;
        }

    zehner = 0;
    while (1)
    if (display_store>=0x14)        //Capwert >= 20
        {
        ++zehner;
        display_store=display_store-0x14;
        }
       else{
            zehner = zehner+0x30;   //Zehner wertigkeit in ASCII
            break;
        }

    einer = 0;
    while (1)
    if (display_store>=0x02)        //Capwert >= 2
        {
```

```
        ++einer;
        display_store=display_store-0x02;
        }
        else{
            einer = einer+0x30;     //Einer wertigkeit in ASCII
            break;
            }

    for(;;){
    if (PORTCbits.RC0 == 0x00){
        display_store=0x80;
        write_command ();           //Position in Zeile 1 (=0x80+0x00)
        writeString ("Capacitor value");    //Text und Werte anzeigen
        display_store=0xC0;
        write_command ();            //Position 1 in Zeile 2 (=0x80+
0x40)
        display_store=0x20;
        write_data ();               //Leerzeichen 1 in Zeile 2
        display_store=0x20;
        write_data ();              //Leerzeichen
        display_store=('C');
        write_data ();              //C
        display_store=('=');
        write_data ();              //=
        display_store=(hund);
        write_data ();              //Hunderter
        display_store=(zehner);
        write_data ();              //Zehner
        display_store=(',');
        write_data ();              //,
        display_store=(einer);
        write_data ();              //Einer
        display_store=(0x5B);
        write_data ();              //[
        display_store=('u');
        write_data ();              //u
        display_store=('F');
        write_data ();              //F
        display_store=(0x5D);
        write_data ();              //]
        display_store = 0x02;
        write_command();            //Return cursor to home position
    }
    else{
        LCD_RESET = 0x00;           //Clear LCD
        for (counter=0; counter<=4; counter++)__delay_ms(25); //Warte
100msec
        LCD_RESET = 0x01;           //Initiate LCD
        break;
    }
}
    goto restart;
}
```