```
/* Es soll die Frequenz eines externen signals gemessen und an
 * einem LCD Display angezeigt werden.
 * Das externe Signal wird an Timer0 clock input T0CKI (RA4)
 * angeschlossen.
 * Der Zähler wird innerhalb eines bekannten Fensters initiiert,
 * und sein Inhalt abgelesen.
 * Wird das Zeitfenster auf 1sec festgelegt, so wird der Zählerinhalt
 * am LCD in Hz angezeigt.
 * Der Controller PIC 18F25K22 wird mit 8Mhz internen clock
 * und einen 1/8 prescaler (Intruction cyrcle Ic=4µsec) betrieben.
 * Das Zeitfenster wird über einen 16bit Timer1 erreicht welcher
 * 4x250msec interrupts erzeugt.
 * Die Voreinstellung von TMR1 ergibt sich demnach zu:
 * 65.536-250.000/Ic = 3.036. (TMR1L = 0xDC / TMR1H = 0x0B)
 * File:   Fr_counter.c
 * Author: lasaros Goumas
 * Created on 9. Februar 2021, 11:21
 */


/* 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 BORV = 190       //VBOR set to 1.90 V nominal
#pragma config WDTEN = ON       //Power up timer enabled
#pragma config PBADEN = OFF
#pragma config LVP = ON
#pragma config CP0 = OFF
#pragma config CPD = OFF
#pragma config WRT0 = OFF
#pragma config WRTD = OFF
#pragma config EBTR0 = OFF
#pragma config EBTRB = 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 lcd_info;          //DATA to be send to LCD
unsigned int temp;             //Temporäres LCD Register
unsigned int overflow;          //Timer0 overflow count
```

```
unsigned int counter;           //Timer1 overflow count
unsigned int low;               //Timer0 low_byte
unsigned int high;              //Timer0 high_byte
unsigned long elapsed;          //Timer0 reading
unsigned int count;             //Allgemeines Zählregister
int ztaus;                      //Wertigkeit 10.000
int taus;                       //Wertigkeit 1.000
int hund;                       //Wertigkeit 100 = 1ste Komma Stelle
int zehn;                       //Wertigkeit 10
int eins;                       //Wertigkeit einer
const char *pnt;               //String pointer


/*Funktionen

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

void init_PIC (void){

    ANSELA = 0;                 //PORTA as digital
    TRISAbits.RA4 = 1;          //RA4 (TOCKI) is input
    ANSELB = 0;                 //PORTB as digital
    LATB = 0x00;                //Clear all RB output latches
    TRISB = 0x00;               //RB Pins sind Ausgänge
    OSCCON = 0b01100111;        //8Mhz interner Oscillator stable
    T0CON = 0b00101000;         //16bit counter/External clock/no
prescaler
    T1CON = 0b00110110;         //16bit instruct. source timer1;1/8
prescaler
}


void __interrupt()_High_Prio (void){
    if (TMR0IF == 1){               //Timer0 interrupt?
        overflow++;                 //Increment Timer0 overflow count
        INTCONbits.TMR0IF = 0;      //Clear Timer0 interrupt flag
    }
    if (PIR1bits.TMR1IF == 1){       //Timer1 interrupt?
        TMR1L = 0xDC;
        TMR1H = 0x0B;               //Reload Timer1 (DC/0B)
        counter++;                  //Increment Timer1 counter
        PIR1bits.TMR1IF = 0;        //Clear Timer1 interrupt flag
    }
}


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
    PIE1bits.TMR1IE = 0x00;         //Timer1 overflow interrupt disabled
    T1CONbits.TMR1ON = 0x01;        //Start TMR1 timer
    TMR1 = 0x00;
```

```
    while (TMR1!=0x1388);           //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
    TMR1 = 0x00;
    while (TMR1!=0x1388);           //2ms 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
    PIE1bits.TMR1IE = 0x00;         //Timer1 overflow interrupt disabled
    T1CONbits.TMR1ON = 0x01;        //Start TMR1 timer
    TMR1 = 0x00;
    while (TMR1!=0x1388);           //2ms 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
    TMR1 = 0x00;
    while (TMR1!=0x1388);           //2ms warten
}


void init_LCD (void){
    LCD_RS=0;
    LCD_E=0;
    PIE1bits.TMR1IE = 0x00;     //Timer1 overflow interrupt disabled
    T1CONbits.TMR1ON = 0x01; //Start TMR1 timer
    TMR1 = 0x00;
    while (TMR1!=0x60D4);    //50ms 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&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
}


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


void reset (void){
    lcd_info=0x80;
    write_command ();                   //Position in Zeile 1 (=0x80+
0x00)
    lcd_info=0x20;
    write_data ();                      //Leerzeichen 1 in Zeile 1
    writeString ("  Overflow");           //Text und Werte anzeigen
    lcd_info=0xC0;
    write_command ();                   //Position 1 in Zeile 2 (=0x80+
0x40)
    lcd_info=0x20;
    write_data ();                      //Leerzeichen 1 in Zeile 2
    writeString (" Press Reset   ");       //Text und Werte anzeigen
}


/*Main Routine

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

void main(void) {
    init_PIC ();
    init_LCD();

    PIR1bits.TMR1IF = 0;        //Clear TMR1 overflow interrupt flag
    PIE1bits.TMR1IE = 1;        //Timer1 overflow interrupt enabled
    INTCONbits.INT0IF = 0;      //Clear INTO external interrupt flag
    INTCONbits.INT0IE = 0;      //INTO external interrupt disabled
    INTCONbits.PEIE_GIEL = 1;    //Peripheral interrups enabled
    INTCONbits.GIE_GIEH = 1;    //Global interrupts enabled
    RCONbits.IPEN = 1;          //Interrupt priority enabled

    for(;;){
        T1CONbits.TMR1ON = 0;       //Stopp TMR1 Timer
        lcd_info = 0x00;
        TMR0L = 0;
        TMR0H = 0;              //Clear Timer0 registers
        TMR1L = 0xDC;
        TMR1H = 0x0B;           //Load Timer1 registers (DC/0B)
        overflow = 0;
        counter = 0;            //Clear both overflow counters
        T0CONbits.TMR0ON = 1;
```

```
        T1CONbits.TMR1ON = 1;   //Start both Timers
        while(counter!=0x04);   //Wait until 1sec elapsed
        T0CONbits.TMR0ON = 0;
        T1CONbits.TMR1ON = 0;   //Stopp both Timers
        low = TMR0L;
        high = TMR0H;           //Get Timer0 count
        elapsed = high*256+low;
        lcd_info = 65535*overflow+elapsed;      //Gemessene frequenz

        if (lcd_info >= 0XC350)  goto stopp;  //Gemessene
frequenz>50kHz?
    else{
        goto data;
        }

stopp:  while (1)
        reset ();                               //Überlaufsanzeige

data:   ztaus = 0;                             //Zehntausener Wertigkeit
        while (1)
        if (lcd_info>=10000){
            ++ztaus;
            lcd_info = lcd_info-10000;
        }
        else{
            ztaus = ztaus+0x30;             //Zehntauseren in ASCII
            break;
        }

        taus = 0;                              //Tauseren Wertigkeit
        while (1)
        if (lcd_info>=1000){
            ++taus;
            lcd_info = lcd_info-1000;
        }
        else{
            taus = taus+0x30;                   //Tauseren in ASCII
            break;
        }

        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;
    }

lcd_info = (0x80);
write_command();                //Position 1 in Zeile 1 (=0x80+0x00)
lcd_info = (0x20);
write_data();                   //Leerzeichen
writeString ("  Frequenz");
lcd_info = (0xC0);
write_command();                //Position 1 in Zeile 2 (=0x80+0x04)
lcd_info = (0x20);
write_data();                   //Leerzeichen
lcd_info = (0x20);
write_data();                   //Leerzeichen
lcd_info = ('F');
write_data();                   //F
lcd_info = ('=');
write_data();                   //=
lcd_info = ztaus;
write_data();                   //Zehntauserer
lcd_info = taus;
write_data();                   //Tausener
lcd_info = ('.');
write_data();                   //.
lcd_info = hund;
write_data();                   //Hunderter
lcd_info = zehn;
write_data();                   //Zehner
lcd_info = eins;
write_data();                   //Einser
lcd_info = 0x20;
write_data();                   //Leerzeichen
lcd_info = 0x5B;
write_data();                   //[
lcd_info = ('H');
write_data();                   //H
lcd_info = ('z');
write_data();                   //z
lcd_info = 0x5D;
write_data();                   //]
lcd_info = 0x02;
write_command();                //Return cursor to home position
__delay_ms(750);
}
```

```
}
```