```
/* Es soll die Drehzal einer rotierenden Welle optisch gemessen
 * und an einem 4-Bit Display (EA DOG M162E-A)angezeigt werden.
 * Zur Frequenzerzeugung wird ein Präzition-timer NE555 verwendet.
 * Das Ausgangssignal des NE555 Timers wird an T0CKI
 * clock input des Microcontrolers angeschlossen.
 * Die NE555 Frequenz wird mit dem 16bit TMR0 timer der, innerhalb
 * eines bekannten Zeitfensters aktiviert wird, gemessen.
 * Das Zeitfenster wird mit einen 16bit TMR1 timer auf 1sec festgelegt,
 * so dass der TMR0 Zählerinhalt am LCD in Hz und RPM angezeigt wird.
 * Der Controller PIC 18F25K22 wird mit 8Mhz internen clock mit
 * einem 1/8 prescaler (Instruction cycle=4µsec) betrieben.
 * Das Zeitfenster wird über den 16bit Timer1 definiert welcher
 * 4x250msec interrupts erzeugt.
 * Die Voreinstellung von TMR1 ergibt sich demnach zu:
 * 65.536-250.000/Ic = 3.036. (TMR1L = 0xDC / TMR1H = 0x0B)
 * File: stroboscop.c
 * Author: lasaros Goumas
 * Created on 19. November 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 WDTEN = OFF      //WDT disabled
#pragma config PBADEN = OFF     //PORTB<5:0> digital I/O on reset
#pragma config LVP = OFF
#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 Dataport für das Display
#define LCD_RESET PORTCbits.RC7
unsigned int lcd_info;             //DATA to be send to LCD
unsigned int temp;                  //Temporäres LCD Register
unsigned int overflow;              //Timer0 overflow counter
unsigned int counter;               //Timer1 overflow counter
unsigned int low;                   //Timer0 low_byte
unsigned int high;                  //Timer0 high_byte
unsigned long elapsed;              //Timer0 reading
unsigned int count;                 //Allgemeines Zählregister
const char *pnt;                    //String pointer
int vztaus;                         //RPM Wertigkeit 10.000
int vtaus;                          //RPM Wertigkeit 1.000
int vhund;                          //RPM Wertigkeit 100 = 1ste Komma
Stelle
int vzehn;                          //RPM Wertigkeit 10
int veins;                          //RPM Wertigkeit einer
int fhund;                          //Frequenz Wertigkeit 100
int fzehn;                          //Frequenz Wertigkeit 10
int feins;                          //Frequenz Wertigkeit einer



/*Funktionen

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

void init_PIC (void){

    ANSELA = 0x00;              //PORTA as digital
    TRISA = 0b00010000;         //RA4 (TOCKI) is input
    ANSELB = 0x00;              //PORTB as digital
    TRISB = 0x00;               //RB Pins sind Ausgänge
    LATB = 0x00;                //Clear all PORTB output latches
    ANSELC = 0x00;              //PORTC as digital
    TRISC =0x00;                //RC Pins sind Ausgänge
    LCD_RESET = 0x01;
    OSCCON = 0b01100111;        //8Mhz interner Oscillator stable
    T0CON = 0b00101000;         //16bit counter/External clock/no
prescaler
    T1CON = 0b00110110;         //16bit instr. source timer1;1/8
prescaler
}


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!=0x01F4);      //Warte 2msec
    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!=0x01F4);      //Warte 2msec
}


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!=0x01F4);      //Warte 2msec
    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!=0x01F4);      //Warte 2msec
}


void init_LCD (void){
    LCD_RESET = 0x00;           //Clear LCD
    for (count=0; count<=4; count++)__delay_ms(25); //Warte 100ms
    LCD_RESET = 0x01;
    LCD_RS=0;
    LCD_E=0;
    lcd_info = (0x03);          //8bit
    write_command();
    __delay_us(30);
    lcd_info = (0x03);          //8bit
    write_command();
    __delay_us(30);
    lcd_info = (0x03);          //8bit
    write_command();
    __delay_us(30);
    lcd_info = (0x02);          //4bit
    write_command();
    __delay_us(30);
    lcd_info = (0x29);          //Function set; 4bit; 2 lines; IS 1
    write_command();
     __delay_us(30);            //30µsec warten
    lcd_info = (0x1C);          //Bias set 1/4; 2 lines
    write_command();
    __delay_us(30);             //30µsec warten
    lcd_info = (0x52);          //Power control;ICON&Booster
off;Contrast C5
    write_command();
    __delay_us(30);             //30µsec warten
    lcd_info = (0x69);          //Follower control on; Gain Rab0
    write_command();
    __delay_us(30);             //30µsec warten
    lcd_info = (0x74);          //Contrast c2 set;
    write_command();
    __delay_us(30);             //30µsec warten
    lcd_info = (0x28);          //Function set; Switch back to IS 0
    write_command();
    __delay_us(30);             //30µsec warten
    lcd_info = (0x0C);          //Display ON; Cursor & Cursor Blink off
    write_command();
     __delay_us(30);            //30µsec warten
    lcd_info = (0x01);
    write_command();            //Clear screen; Cursor to home position
    __delay_ms(2);              //2msec warten
    lcd_info = (0x06);
    write_command();            //Entry mode; Cursor auto-increment
     __delay_us(30);            //30µsec warten
}


void __interrupt()_High_Prio (void){
    if (INTCONbits.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
        counter++;                  //Increment Timer1 counter
        PIR1bits.TMR1IF = 0;        //Clear Timer1 interrupt flag
    }
}


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


void reset (void){                  //Überlauf Warnung
    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();                 //LCD iitialisierung

    for(;;){
        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
        RCONbits.IPEN = 1;          //Interrupt priority enabled
        INTCONbits.GIE_GIEH = 1;    //Global interrupts enabled
        INTCONbits.PEIE_GIEL = 1;   //Peripheral interrups enabled
        T1CONbits.TMR1ON = 0;       //Stopp TMR1 Timer
        TMR0 = 0;                   //Clear Timer0
        TMR1L = 0xDC;
        TMR1H = 0x0B;               //Load Timer1 registers
        overflow = 0;               //Clear TMR0 counter
        counter = 0;                //Clear TMR1 counter
        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;     //Timer0 reading
        lcd_info = 65535*overflow+elapsed; //Gemessene frequenz
        temp = lcd_info*60;         //Gemessenen Umdrehungen/Min

        if (lcd_info>= 0XFA)  goto stopp;  //Gemessene frequenz>250Hz?
        else{
        goto data;
        }

stopp:  while (1)
        reset ();

data:   fhund = 0;                     //Frequenz Hunderter Wertigkeit
        while (1)
        if (lcd_info>=100){
        ++fhund;
        lcd_info = lcd_info-100;
    }
    else{
        fhund = fhund+0x30;            //Frequenz Hunderter in ASCII
        break;
    }

    fzehn = 0;                         //Frequenz zehner Wertigkeit
```

```
while (1)
if (lcd_info>=10){
    ++fzehn;
    lcd_info = lcd_info-10;
}
else{
    fzehn = fzehn+0x30;            //Frequenz Zehner in ASCII
    break;
}

feins = 0;                          //Frequenz einer Wertigkeit
while (1)
if (lcd_info>=1){
    ++feins;
    lcd_info = lcd_info-1;
}
else{
    feins = feins+0x30;            //Frequenz einer in ASCII
    break;
}

lcd_info = temp;
vztaus = 0;                         //RPM Zehntausener Wertigkeit
while (1)
    if (lcd_info>=10000){
        ++vztaus;
        lcd_info = lcd_info-10000;
    }
        else{
            vztaus = vztaus+0x30;  //RPM Zehntausener in ASCII
            break;
        }

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

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

vzehn = 0;                        //RPM zehner Wertigkeit
while (1)
```

```
if (lcd_info>=10){
    ++vzehn;
    lcd_info = lcd_info-10;
}
else{
    vzehn = vzehn+0x30;            //RPM Zehner in ASCII
    break;
}

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


lcd_info=(0x80);
write_command ();          //Position in Zeile 1 (=0x80+0x00)
lcd_info=(0x20);
write_data();             //Leerzeichen
lcd_info=('F');
write_data();              //F
lcd_info=('o');
write_data();              //o
lcd_info=('=');
write_data();              //=
lcd_info=(fhund);
write_data();              //Frequenz hunderter
lcd_info=(fzehn);
write_data();              //Frequenz zehner
lcd_info=(feins);
write_data();              //Frequenz einzer
lcd_info=(0x20);
write_data();             //Leerzeichen
lcd_info=('[');
write_data();              //[
lcd_info=('H');
write_data();              //H
lcd_info=('z');
write_data();              //z
lcd_info=(']');
write_data();              //]
lcd_info=(0x20);
write_data();             //Leerzeichen
lcd_info=0xC0;
write_command ();           //Position 1 in Zeile 2 (=0x80+0x40)
lcd_info=0x20;
write_data ();              //Leerzeichen 1 in Zeile 2
lcd_info=('V');
write_data();              //V
lcd_info=('o');
write_data();              //o
```

```
    lcd_info=('=');
    write_data();               //=
    lcd_info=(vztaus);
    write_data();               //RPM zehntausender
    lcd_info=(vtaus);
    write_data();               //RPM tausender
    lcd_info=('.');
    write_data();               //.
    lcd_info=(vhund);
    write_data();               //RPM hunderter
    lcd_info=(vzehn);
    write_data();               //RPM zehner
    lcd_info=(veins);
    write_data();               //RPM einzer
    lcd_info=(0x20);
    write_data();              //Leerzeichen
    lcd_info=('[');
    write_data();               //[
    lcd_info=('R');
    write_data();               //R
    lcd_info=('P');
    write_data();               //P
    lcd_info=('M');
    write_data();               //M
    lcd_info=(']');
    write_data();               //]
    for (count=0; count<=8; count++)__delay_ms(125);    //Warte 1Sec
    }
}
```