```
/* Use a rotary incremental encoder providing 15° detents
 * ie 24 positions over 360°
 * File:   Rotary encoder.c
 * Author: lasaros Goumas
 * Created on 24. März 2023 11:30
 */


/* Includes
 ***************************************************************************/

#include <xc.h>
#include <math.h>
#include <pic18f25K22.h>            //PIC 18F25K22 Controller
#include "writeStringon lcd.h"


/*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 = ON
#pragma config CP1 = OFF
#pragma config CPB = OFF
#pragma config WRT0 = OFF
#pragma config WRTC = OFF
#pragma config EBTR0 = OFF
#pragma config EBTRB = OFF


/*Declarations
 ***************************************************************************/
#define _XTAL_FREQ 8000000      // Fosc frequency for _delay() library
#define LCD_RS PORTCbits.RC4    //High: Data ; Low: Instruction code
#define LCD_E PORTCbits.RC5     //High: Chip enable
#define LCD_DATA PORTC          //PORTC ist Datenport für das display
#define LCD_RESET PORTBbits.RB4
#define CLK PORTBbits.RB1       //Encoder A output
#define DT  PORTBbits.RB2       //Encoder B output
#define LED PORTBbits.RB3       //Encoder switch pressed
unsigned char CW;               //Encoder movement clockwise
unsigned  char counter1;        //Encoder detents clockwise
unsigned char CCW;              //Encoder movement counterclockwise
unsigned  char counter2;        //Encoder detents counterclockwise
unsigned char detents;          //Aktual number of detents
unsigned char lcd_info;         //DATA to be send to LCD
unsigned char temp;             //Temporäres LCD Register
const char *pnt;                //String pointer
unsigned char hund;             //Wertigkeit 100
unsigned char zehn;             //Wertigkeit 10
unsigned char eins;             //Wertigkeit einser
unsigned int count;             //Allgemeines zählregister
```

```
/*Funktionen
 ************************************************************************/
void init_PIC (void){
    TRISB = 0b00000111;         //RB<3:0> are inputs
    ANSELB = 0x00;              //PORTB as digital
    ANSELC = 0x00;              //PORTC as digital
    TRISC = 0x00;               //PORTC are outputs
    LATC = 0x00;                //Clear all RC output latches
    OSCCON = 0b01110110;        //16Mhz interner Oscillator stable
    counter1 = 0x00;
    counter2 = 0x00;
}

void init_interrupts (void){
    IPEN = 1;
    INTCON = 0b10010000;        //External INT0 interrupt enabled
    INTCON2 = 0b00000000;       //INT1/2 interrupts on falling egde
    WPUB = 0b00000111;          //PORTB pull ups enabled
    INTCON3 = 0b11011000;       //External ionterrupts INT1/2 enabled
    ei();
}


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 PORTC
    LCD_E = 1;                  //LCD enabled
    LCD_E = 0;                  //High Nibble an LCD übergeben
    __delay_ms(2);              //Warte 2msec
    temp=lcd_info;
    temp=temp & 0x0F;           //High nibbles of temp ausmaskiert
    LCD_DATA =temp;             //Low nibbles of lcd_info an PORTC
    LCD_E = 1;                  //LCD enabled
    LCD_E = 0;                  //Low Nibble an LCD übergeben
    __delay_ms(2);              //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 PORTC
    LCD_RS = 0x01;              //Write data
    LCD_E = 1;                  //LCD enabled
    LCD_E = 0;                  //High Nibble an LCD übergeben
    __delay_ms(2);              //Warte 2msec
    temp=lcd_info;
    temp=temp & 0x0F;           //High nibbles of temp ausmaskiert
    LCD_DATA =temp;             //Low nibbles of lcd_info an PORTC
    LCD_RS = 0x01;              //Write data
    LCD_E = 1;                  //LCD enabled
    LCD_E = 0;                  //Low Nibble an LCD übergeben
    __delay_ms(2);              //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);             //30?sec warten
    lcd_info = (0x1C);          //Bias set 1/4; 2 lines
    write_command();
    __delay_us(30);             //30?sec warten
    lcd_info = (0x52);          //Power control;ICON&Booster off;Contrast C5
    write_command();
    __delay_us(30);             //30?sec warten
    lcd_info = (0x69);          //Follower control on; Gain Rab0
    write_command();
    __delay_us(30);             //30?sec warten
    lcd_info = (0x74);          //Contrast c2 set;
    write_command();
    __delay_us(30);             //30?sec warten
    lcd_info = (0x28);          //Function set; Switch back to IS 0
    write_command();
    __delay_us(30);             //30?sec warten
    lcd_info = (0x0C);          //Display ON; Cursor & Cursor Blink off
    write_command();
    __delay_us(30);             //30?sec 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);             //30µ,sec warten
}
```

```
void __interrupt()_High_Prio (void){

    if ( INT0IE && INT0IF)                      //RB0 interrupt
    {LED = ~LED;                                //Switch pressed
    INTCONbits.INT0IF = 0;}

    if (INTCON3bits.INT1IF>INTCON3bits.INT2IF)  //RB1 flag raised first
    {CW++;
    if (DT == 0)
    {counter1 = CW-1;
    CW = counter1;}
    else counter1 = CW;
    }

    if (INTCON3bits.INT2IF>INTCON3bits.INT1IF)  //RB2 flag raised first
    {CCW++;
    if (CLK == 0)
    {counter2 = CCW-1;
    CCW = counter2;}
    else counter2 = CCW;
    }

    INTCON3bits.INT1IF = 0;
    INTCON3bits.INT2IF = 0;
    }


void data_wertigkeit (void){
    lcd_info = detents;

    hund = 0;                       //Hunderter Wertigkeit
    while (1)
        if (lcd_info>=100){
            ++hund;
            lcd_info = lcd_info-100;}
    else{
            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;                       //Einer Wertigkeit
    while (1)
            if (lcd_info>=1){
```

```
            ++eins;
            lcd_info = lcd_info-1;}
    else{
            eins = eins+0x30;       //Einer in ASCII
            break;
            }
}


void display_detents (void){
    lcd_info = (0x20);
    write_data();               //Leerzeichen
    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 = ('D');
    write_data();               //D
    lcd_info = ('e');
    write_data();               //e
    lcd_info = ('t');
    write_data();               //t
    lcd_info = ('e');
    write_data();               //e
    lcd_info = ('n');
    write_data();               //n
    lcd_info = ('t');
    write_data();               //t
    lcd_info = ('s');
    write_data();               //s
    lcd_info = 0x5D;
    write_data();               //]
    lcd_info = (0x02);
    write_command();            //Return cursor to home position
    __delay_ms(10);
}


/*Main Routine*
 **************************************************************************/

void main(void) {
    init_PIC();
    init_interrupts();
    init_LCD();                     //LCD iitialisierung

    for(;;){
    while (CLK == 0 | DT == 0);    //Wait with encoder on idle
    if (counter1>counter2){
    detents = counter1-counter2;
    data_wertigkeit();
    lcd_info = (0x80);
    write_command();            //Position 1 in Zeile 1 (=0x80+0x00)
    writeString ("***Clockwise****");
    lcd_info = (0xC0);
    write_command();            //Position 1 in Zeile 2 (=0x80+0x04)
    display_detents();
    }

else if (counter2>counter1){
    detents = counter2-counter1;
    data_wertigkeit();
    lcd_info = (0x80);
    write_command();            //Position 1 in Zeile 1 (=0x80+0x00)
    writeString ("Counterclockwise");
    lcd_info = (0xC0);
    write_command();            //Position 1 in Zeile 2 (=0x80+0x04)
    display_detents();
    }

    else {
    lcd_info = (0x80);
    write_command();            //Position 1 in Zeile 1 (=0x80+0x00)
    writeString ("****Encoder*****");
    lcd_info = (0xC0);
    write_command();            //Position 1 in Zeile 2 (=0x80+0x04)
    writeString (" Zero position");
    }
    }
}
```