```
/* Die Schaltung soll eine time basis von 1 Sekunde mit dem
 * Tastverhältnis erzeugen.
 * Fosc=4,096MHz extern
 * TMROprescale=1/8
 * File:   sekunde.c
 * Author: lasaros Goumas
 * Created on 13. Januar 2023, 20:57
 */


/* Includes

************************************************************************
**/
#include <xc.h>
#include <p18cxxx.h>
#include <pic18f252.h>            //PIC 18F252 Controller


/*Configuration

************************************************************************
***/
#pragma config OSC = XT          //XT oscillator
#pragma config BORV = 20         //Brown-up reset voltage set to 2,0V
#pragma config WDT = OFF
#pragma config CCP2MUX = ON      //CCP2 input/output multiplexed with RC1
#pragma config LVP = OFF
#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 4096000       // Fosc frequency for _delay() library
#define LCD_DATA PORTC           //PORTC ist Datenport für das display
#define LCD_RS PORTCbits.RC4     //High: Data ; Low: Instruction code
#define LCD_E PORTCbits.RC5      //High: Chip enable
#define LCD_RESET PORTBbits.RB7 //LCD reset
#define LEDs PORTBbits.RB4        //Sekunden Ausgang
#define power_on PORTBbits.RB3

unsigned int count;              //Allgemeines Zählregister
unsigned int seconds;            //Halbe sekunde
unsigned int minutes;
unsigned int hours;
unsigned int lcd_info;           //DATA to be send to LCD
unsigned int temp;               //Temporäres LCD Register
unsigned int zehn_m;             //Minuten 10ner Wertigkeit
unsigned int zehn_h;             //Stunden 10ner Wertigkeit
unsigned int eins_m;             //Minuten einer Wertigkeit
unsigned int eins_h;            //Stunden einer Wertigkeit
const char *pnt;                //String pointer


/*Funktions
***********************************************************************
/
void init_PIC (void){
    TRISC = 0x00;               //RC PINs are outputs
    PORTC = 0x00;
    TRISB = 0b00000111;         //RB<7:3> outputs
    RCONbits.IPEN = 0;          //All unmasked interrupts enabled
    INTCON = 0b00110000;        //TMR0 overflow & INTO enabled
    INTCON2bits.INTEDG0 = 0;    //Interrupt on falling edge
    INTCON2bits.TMR0IP = 1;     //TMR0 overflow high priority
    INTCON2bits.INTEDG2 = 0;    //External interrupt2 on falling edge
    INTCON3 = 0b10010000;       //External interrupt2 enabled
    T0CON = 0b00000010;         //16 bit internal TMRO / 1:8 prescal
}


void __interrupt()_High_Prio (void){
    if (TMR0IE && TMR0IF){      //Timer 0 interrupt?
    INTCONbits.TMR0IF = 0;      //TMR0 overflow flag cleared
    TMR0 = 0x8330;              //TMR0 preset (33.536 ' 0x8300)
    count++;                    //TMR0 Overfow anzahl
    }
    if (INT0IE && INT0IF){      //External interrupt on RB0
        INTCONbits.INT0IF = 0;  //External interrupt flag cleared
        if(PORTBbits.RB1 == 1)  //Preset Minutes
            minutes++;
        else hours++;           //Preset hours
    }
    if (INT2IE && INT2IF){      //External interrupt on RB2
        INTCON3bits.INT2IF = 0; //External interrupt2 flag cleared
        if (PORTBbits.RB1 == 1) //Preset Minutes
            minutes--;
        else hours--;
    }
    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(1);              //Warte 1msec
    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(1);              //Warte 1msec
}


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(1);              //Warte 12msec
    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(1);              //Warte 1msec
}


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


void display_data (void){
    zehn_m = 0;
    lcd_info = minutes;
    while (1)
        if (lcd_info>=10){
            ++zehn_m;
            lcd_info = lcd_info-10;}
    else{
            zehn_m = zehn_m+0x30;     //Minuten zehner in ASCII
            break;
            }
    eins_m = 0;
    while (1)
            if (lcd_info>=1){
            ++eins_m;
            lcd_info = lcd_info-1;}
    else{
            eins_m = eins_m+0x30;     //Minuten einer in ASCII
            break;
            }

    zehn_h = 0;
    lcd_info = hours;
    while (1)
        if (lcd_info>=10){
            ++zehn_h;
            lcd_info = lcd_info-10;}
    else{
            zehn_h = zehn_h+0x30;     //Stunden zehner in ASCII
            break;
            }
    eins_h = 0;
    while (1)
            if (lcd_info>=1){
            ++eins_h;
            lcd_info = lcd_info-1;}
    else{
            eins_h = eins_h+0x30;     //Stunden einer in ASCII
            break;
            }

        lcd_info = (0x80);
        write_command();             //Position 1 in Zeile 1 (=0x80+0x00)
        lcd_info = (0x20);
        write_data();                //Leerzeichen
        writeString ("Real time clock");
        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 = (0x20);
        write_data();                //Leerzeichen
        lcd_info = zehn_h;
        write_data();
        lcd_info = eins_h;
        write_data();                //Stunden
        lcd_info = ('h');
        write_data();                //h
        lcd_info = 0x20;
        write_data();                //Leerzeichen
        lcd_info = 0xFD;
        write_data();                //:
        lcd_info = 0x20;
        write_data();                //Leerzeichen
        lcd_info = zehn_m;
        write_data();
        lcd_info = eins_m;
        write_data();                //Minuten
        lcd_info = ('m');
        write_data();                //m
        lcd_info = 0x02;
        write_command();             //Return cursor to home position
        __delay_ms(2);
        count = 0x00;
    }


/*Main Routine

************************************************************************
/
void main(void) {

    init_PIC ();
    init_LCD ();
    INTCONbits.GIE_GIEH = 1;         //global interrupt enabled
    count = 0x00;
    TMR0 = 0x8300;                   //Timer preset (33.536 ' 0x8300)
    T0CONbits.TMR0ON = 1;            //TMR0 timer enabled

        while(1){
        if(count <= 0x01);           //Wait until 0,5sec elapsed (128)
        else{
            count = 0x00;            //Clear count
            LEDs =~ LEDs;            //Flip the bit at 0,5 Hrz
            seconds++;

            if(seconds >= 0x78)      //Wait until 1 minute elapsed (120
´´:0x78)
            {minutes++;
            seconds = 0x00;
            }

            if(minutes >= 0x3C)      //Wait until 1 hour elapsed (60
´:0x3C)
            {hours++;
            minutes = 0x00;
            }

            if(hours >= 0x06)
            power_on = 0x01;         //Switch on power

            if(hours >= 0x18)        //One day has expired? (24h:0x18)
            {power_on = 0x00;
            hours = 0x00;
            }
            display_data();
            }
    }
}
```