```
/*
 Kopp's Solarthermie Elektronik Steuerung (30.10.2018)

    Dieses Programm ist Freie Software: Sie können es unter den Bedingungen
    der GNU General Public License, wie von der Free Software Foundation,
    Version 3 der Lizenz oder (nach Ihrer Wahl) jeder neueren
    veröffentlichten Version, weiter verteilen und/oder modifizieren.

    Dieses Programm wird in der Hoffnung bereitgestellt, dass es nützlich sein wird,
    jedoch
    OHNE JEDE GEWÄHR,; sogar ohne die implizite
    Gewähr der MARKTFÄHIGKEIT oder EIGNUNG FÜR EINEN BESTIMMTEN ZWECK.
    Siehe die GNU General Public License für weitere Einzelheiten.

    Sie sollten eine Kopie der GNU General Public License zusammen mit diesem
    Programm erhalten haben. Wenn nicht, siehe <https://www.gnu.org/licenses/>.
*/

#ifndef F_CPU
#define F_CPU 12000000UL
#endif

#define EEPROM_ADRESSE_TEMP_DIFF 1
#define PUMPE_AUS               0
#define PUMPE_START             1
#define PUMPE_AUSSCHALTEN       2
#define PUMPE_LAEUFT            3

#include "T6963c.h"
#include "CustomChars.h"
#include <util/delay.h>
#include <avr/eeprom.h>
#include <avr/interrupt.h>
#include <avr/wdt.h>

uint8_t pos_offset;
volatile int16_t temp_dach, temp_speicher_drain_back, temp_speicher_oben,
    temp_speicher_unten, temp_heizung;
char buffer1[6];
uint16_t eeprom_adresse EEMEM = 10;
uint8_t pumpe_drehzahl_soll=0,pumpe_drehzahl=0, modus_pumpe=0, druck=0, wasser_hoehe=2;
volatile uint32_t sec=0, sec_pumpe_start=0, sec_pumpe_aus=0, sec_pumpe_ein=0,
    min_pumpe_ein_ges ;
volatile uint16_t min_pumpe_ein=0, hour_pumpe_ein_ges;

ISR(TIMER1_COMPA_vect) //1 Sekunden Timer
{
++sec;
if(modus_pumpe==PUMPE_LAEUFT) ++sec_pumpe_ein;  //Anzeige der Laufzeit der Pumpe um 1
    Sekunde erhoehen
}

void ADC_Init(void)                     // ADC initialisieren
{
    ADMUX = (1<<REFS0);                     // interne Referenzspannung
    ADCSRA = (1<<ADPS1) | (1<<ADPS0);       // Frequenzvorteiler
    ADCSRA |= (1<<ADEN);                    // ADC aktivieren
    ADCSRA |= (1<<ADSC);                    // eine Dummy-ADC-Wandlung
    while (ADCSRA & (1<<ADSC) ) {}          // auf Abschluss der Konvertierung warten
    (void) ADCW;
}

uint16_t ADC_Read( uint8_t channel )    // ADC Einzelmessung
{
    ADMUX = (ADMUX & ~(0x1F)) | (channel & 0x1F);
    ADCSRA |= (1<<ADSC);                            // eine Wandlung "single conversion"
    while (ADCSRA & (1<<ADSC) ) {}                  // auf Abschluss der Konvertierung
    warten

    return ADCW;                                    // ADC auslesen und zurückgeben
}

int16_t ADC_Read_Avg( uint8_t channel, uint8_t nsamples )   // ADC Mehrfachmessung mit
    Mittelwertbildung
```

```
{
    uint32_t sum = 0;
    int16_t wert;

    for (uint8_t i = 0; i < nsamples; ++i )
    {
        sum += ADC_Read( channel );
    }
    wert = (int16_t)( sum / nsamples );
    wert=wert-519;                          //Null Grad Offset
    wert=wert*73;                           //Kennliniensteigung 73°/100
    wert=(int16_t)wert/100;
    itoa( wert, buffer1, 10 );
    if(wert>99) pos_offset=0;
    if((wert<100) & (wert>9)) pos_offset=1;
    if((wert<10) & (wert>-1)) pos_offset=2;
    if((wert<0) & (wert>-10)) pos_offset=1;
    if(wert<-10) pos_offset=0;
    return wert;
}

int32_t avg(int16_t neu)
{
   static int32_t speicher=0;
   int16_t temp;
   temp=speicher/32;        //shiften nur einmal durchführen  (?? Warum /32 ??
   speicher=speicher - temp + neu;
   return temp;
}

uint8_t position_offset (uint16_t zahl)   //Routine zur rechtsbuendigen Dartsellung der
    Zahlen
{
    if(zahl>999) pos_offset=0;
    if((zahl<1000) & (zahl>99)) pos_offset=1;
    if((zahl<100) & (zahl>9)) pos_offset=2;
    if((zahl<10) & (zahl>=0)) pos_offset=3;
    return pos_offset;
}

void LCD_update(void)                   //Hinweis: Die Variable buffer1 wird in der
    ADC_Read-Routine gesetzt
{
    temp_dach=ADC_Read_Avg(0, 100);  // Kanal 0, Mittelwert aus 100 Messungen
    T6963cPutStringXY_P(0, 2, PSTR("Kollektor          ")); T6963cPutStringXY(12+
    pos_offset, 2, buffer1); T6963cPutString("\x80");
    temp_speicher_drain_back=ADC_Read_Avg(1, 100);
    T6963cPutStringXY_P(0, 3, PSTR("Drain Back         ")); T6963cPutStringXY(12+
    pos_offset, 3, buffer1); T6963cPutString("\x80");
    temp_speicher_oben=ADC_Read_Avg(2, 100);
    T6963cPutStringXY_P(0, 4, PSTR("Speich. ob.        ")); T6963cPutStringXY(12+
    pos_offset, 4, buffer1); T6963cPutString("\x80");
    temp_speicher_unten=ADC_Read_Avg(3, 100);
    T6963cPutStringXY_P(0, 5, PSTR("Speich. un.        ")); T6963cPutStringXY(12+
    pos_offset, 5, buffer1); T6963cPutString("\x80");
    temp_heizung=ADC_Read_Avg(4, 100);
    T6963cPutStringXY_P(0, 6, PSTR("Heizung            ")); T6963cPutStringXY(12+
    pos_offset, 6, buffer1); T6963cPutString("\x80");

    itoa( (pumpe_drehzahl_soll*100)/256, buffer1, 10 );
    T6963cPutStringXY_P(18, 2, PSTR("Pumpe       "));   T6963cPutStringXY(25, 2, buffer1);
     T6963cPutString("%");  //Pumpen Leistung


    if(druck==0) itoa( druck, buffer1, 10 ); else itoa( ((druck+1)*2), buffer1, 10 );
    if(((druck+1)*2)<10) {T6963cPutStringXY_P(18, 3, PSTR("Druck  0.   "));
    T6963cPutStringXY(27, 3, buffer1);} else// T6963cPutString("bar");   //Wasser-Druck
             {T6963cPutStringXY_P(18, 3, PSTR("Druck  1.   "));itoa( (((druck+1)*2)-10),
    buffer1, 10 );T6963cPutStringXY(27, 3, buffer1);}

    itoa( wasser_hoehe, buffer1, 10 );  T6963cPutStringXY_P(18, 4, PSTR("Wasser      "));
        T6963cPutStringXY(25, 4, buffer1);
    if((wasser_hoehe==0) | (wasser_hoehe==1)) T6963cPutString(" NOK");  //Wasser-Höhe im
    Drain Back zu niedrig
```

```
    if(wasser_hoehe==2) T6963cPutString(" !!!");  //Wasser-Höhe im Drain Back niedrig
    if(wasser_hoehe==3) T6963cPutString(" ok ");  //Wasser-Höhe im Drain Back ok

    min_pumpe_ein=sec_pumpe_ein/60; position_offset(min_pumpe_ein); utoa( min_pumpe_ein,
    buffer1, 10 );
    T6963cPutStringXY_P(18, 5, PSTR("On_d        "));   T6963cPutStringXY(22+pos_offset, 5
    , buffer1); T6963cPutString(" min"); //On_Time letzten 24h
    hour_pumpe_ein_ges=min_pumpe_ein_ges/60;position_offset(hour_pumpe_ein_ges); utoa(
    hour_pumpe_ein_ges, buffer1, 10 );
    T6963cPutStringXY_P(18, 6, PSTR("On          "));   T6963cPutStringXY(22+pos_offset, 6
    , buffer1); T6963cPutString(" h");  // On_Time gesamt
}

void pumpe_start(void)
{
    for (uint8_t k = 0; k < 250; ++k ) //Hochlauf Höchstdrehzahl
    {
        _delay_ms(50);
        wdt_reset();
        pumpe_drehzahl_soll=k;
        pumpe_drehzahl=avg(pumpe_drehzahl_soll);
        OCR0 = pumpe_drehzahl;
        LCD_update();
    }
    for (uint16_t k = 0; k < 40; ++k )  //40s Full Power-Lauf
    {
        wdt_reset();
        _delay_ms(1000);
        LCD_update();
    }
    for (uint8_t k = 230; k>154; --k ) //Reduzierung auf 60%
    {
        _delay_ms(30);
        wdt_reset();
        pumpe_drehzahl_soll=k;
        pumpe_drehzahl=avg(pumpe_drehzahl_soll);
        OCR0 = pumpe_drehzahl;
        LCD_update();
    }
    modus_pumpe=PUMPE_LAEUFT;
    return(0);
}

void pumpe_ausschalten(void)
{
    for (uint8_t k = 155; k>1; --k )
    {
        _delay_ms(30);
        wdt_reset();
        pumpe_drehzahl_soll=k;
        pumpe_drehzahl=avg(pumpe_drehzahl_soll);
        OCR0 = pumpe_drehzahl;
        LCD_update();
    }
    modus_pumpe=PUMPE_AUS;
    return(0);
}

void druck_pruefung(void)
{
    PORTB |= (1<<PB7);                                  //Port PB7 high = Druck-
    Elektronik wird eingeschaltet
    PORTD |= (1<<PD6);                                  //Clock-Leitung high
    _delay_ms(20);                                      //wegen RC-Reset-
    Zeitkonstante
    for (uint8_t k = 0; k<10; ++k )
    {
        if(PIND & (1 << PD5)) druck=k;                  //Ist die Leitung Druck_in
     high ?
        PORTD &= ~(1<<PD6);                           //Clock-Leitung low
        _delay_ms(1);
        PORTD |= (1<<PD6);                              //Clock-Leitung high
        _delay_ms(1);
    }
```

```
    PORTD &= ~(1<<PD6);                                    //Clock-Leitung low
    PORTB &= ~(1<<PB7);                                    //Port PB7 low = Druck-
    Elektronik wird ausgeschaltet
    return(0);
}


int main()
{
    //Pumpe PWM Output (Timer0)
    DDRB  |= (1<<PB3);                                          // Setup PB3
    as output
    TCCR0 |= (1<<WGM00)|(1<<WGM01)|(1<<COM01)|(1<<CS01)|(1<<CS00);      // Start timer0
    with prescaler 64 im PWM Modus
    min_pumpe_ein_ges = eeprom_read_dword(&eeprom_adresse);              //Hole den
    alten Wert der Pump-Stunden

    //1-Sekunden-Timer (Timer1)
    OCR1A =  0x2DC6;                                            //Set OCR1A
    für 1 Sekunde
    TCCR1A = 0x00;                                              //Timer1
    counter control register
    TCCR1B = (0 << WGM13)|(1 << WGM12)|(1 << CS12)|(0 << CS11)|(1 << CS10); // WGM1=4,
    prescale at 1024
    TIMSK |= (1 << OCIE1A);                                     //Set bit 6 in
    TIMSK to enable Timer 1 compare interrupt.

    //Festes PWM-Verhältnis für Pumpen-Test (Timer2)
    DDRD  |= (1<<PD7);                                        // Setup PD7 as
     output
    TCCR2 |= (1<<WGM20)|(1<<WGM21)|(1<<COM21)|(1<<CS22);          // Start
    timer0 with prescaler 64 im PWM Modus
    OCR2 = 250; //PWM =  97%

    //Init Wasser-Stand-Schalter
    DDRD  &= ~((1<<PD4)|(1<<PD1)|(1<<PD0));                     // Setup PD0, PD1,
     PD4 as input

    //Init Druck-Messung
    DDRB  |= (1<<PB7);                                          // Setup PB7
    as output  (Power_on)
    DDRD  |= (1<<PD6);                                          // Setup PD6
    as output   (Clock)
    DDRD  &= ~(1<<PD5);                                         // Setup PD5 as
    input   (Druck_in)

    //Init Stoerungs LED rot
    DDRB  |= (1<<PB2);                                          // Setup PB2
    as output  (rote LED)

    //Init Heizung Pumpe Steuerung (Heizungsunterstuetzung)
    DDRD  &= ~(1<<PD2);                                         // Setup PD2 as
    input   (Heizungsunterstuetzung ?)
    PORTD |= (1<<PD2);                                          // Pullup ein
    DDRD  |= (1<<PD3);                                        // Setup PD3 as
     output (Heizungspumpe ein)

    ADC_Init();

    //LCD-Init
    T6963cInit( T6963C_MODE_TEXTATTR | T6963C_CG_INTERNALROM, T6963C_TEXT_GRAPHIC ); //|
    T6963C_CURSOR_BLINK );
    T6963cWriteChunkAt_P(T6963C_ADDR_CGRAMH, g_CustomChars, sizeof(g_CustomChars)); //
    Custom Char for use with Character '\x80'
    T6963cPutStringXY_P(0, 0, PSTR("Solarthermie-Steuerung"));

    //Watchdog auf 2s
    wdt_enable(WDTO_2S);

    sei();

    while(1)
    {
        LCD_update();
```

```
        wdt_reset();
        _delay_ms(300);

        if((sec>85500) & (modus_pumpe==PUMPE_AUS))                    // 23.75 Stunden
    sind vergangen (Speichern nach letztem Ausschalten
        {                                                             // der Pumpe
    abends = grobe Tag/Nacht-Syncronisation)
            min_pumpe_ein_ges = min_pumpe_ein_ges + min_pumpe_ein;
            if(min_pumpe_ein_ges > 600000) min_pumpe_ein_ges = 0;
            eeprom_write_dword(&eeprom_adresse, min_pumpe_ein_ges);   //Stunden ins EEprom
     schreiben
            sec_pumpe_ein=0;
            sec=0;
        }

        druck_pruefung();

        wasser_hoehe=0;

        if(!(PIND & (1<<PD0))) {wasser_hoehe=1; PORTB &= ~(1<<PB2);}               //zu
    niediger Wasserstand, rote LED on
        if(!(PIND & (1<<PD1))) {wasser_hoehe=2; PORTB |=  (1<<PB2);}               //
    niedriger Wasserstand, rote LED aus
        if(!(PIND & (1<<PD4))) {wasser_hoehe=3; PORTB |=  (1<<PB2);}               //
    normaler Wasserstand, rote LED aus
        if((!(modus_pumpe))&((wasser_hoehe==0)|(druck==0))) PORTB ^= (1<<PB2);  //toggle
    PB2 = Blinken rote LED

        switch(modus_pumpe)             //Pumpen-Steuerung
        {
            case 0:                         //Pumpe ist aus
                break;
            case 1:                         //Pumpe soll starten
                if((wasser_hoehe==3)|(wasser_hoehe==2)) pumpe_start();
                break;
            case 2:                         //Pumpe soll ausgeschaltet werden
                pumpe_ausschalten();
                break;
            case 3:                         //Pumpe läuft
                break;
        }
        pumpe_drehzahl=avg(pumpe_drehzahl_soll);        //Glättung der Pumpendrehzahl
        OCR0 = pumpe_drehzahl;                          //Übergabe an PWM-Einheit

        //Heizungsunterstützung (Bedingung PIN_PD2 auf low)
        if((!(PIND & (1<<PD2))) & (temp_speicher_oben>55) & (temp_heizung<55)) {PORTD |=
    (1<<PD3);} else {PORTD &= ~(1<<PD3);}

        //Wann soll die Solar-Pumpe laufen ?
        if(druck==0) {modus_pumpe=PUMPE_AUS;pumpe_drehzahl_soll=0;continue;}
                        //Druck-Prüfung
        if((!(modus_pumpe))&(wasser_hoehe==0)) {modus_pumpe=PUMPE_AUS;pumpe_drehzahl_soll=
    0;continue;}    //Wasserstand-Prüfung
        if((!(modus_pumpe))&(wasser_hoehe==1)) {modus_pumpe=PUMPE_AUS;pumpe_drehzahl_soll=
    0;continue;}    //Wasserstand-Prüfung

        if(temp_dach<10) {modus_pumpe=PUMPE_AUS;pumpe_drehzahl_soll=0;continue;}
                        //alles aus bei zu niedrigen
        if(temp_dach>95) {modus_pumpe=PUMPE_AUS;pumpe_drehzahl_soll=0;continue;}
                        //oder zu hohen Temperaturen
        //if(temp_speicher_drain_back<3) {modus_pumpe=PUMPE_AUS;pumpe_drehzahl_soll=0;
    continue;} else {}
        if(temp_dach<temp_speicher_unten) {modus_pumpe=PUMPE_AUS;pumpe_drehzahl_soll=0;
    continue;}
        if((temp_dach-temp_speicher_unten>10)&(modus_pumpe==PUMPE_AUS)){modus_pumpe=
    PUMPE_START;}       //Diffrenz größer 10° ?
        if((temp_dach-temp_speicher_unten<4)&(modus_pumpe==PUMPE_LAEUFT)){modus_pumpe=
    PUMPE_AUSSCHALTEN;} //Diffrenz kleiner 4° ?
    }
    return 0;
}
```