```
/* MOSFET gesteuert Batterieladeregler
 * Ubulk=14,25V; Ufloat=13,7V
 * AN0 Analog Eingang; AN0[V] = 0,25*Ub[V]
 * AN1 Analog Eingang; AN1[V] = 0,3*Ib[A]
 * RB<7:0> Digitale Ausgänge für Display & PICKIT 3
 * RC<2:0> Digitale Ausgänge für Analogschalter(high "ON"/low "OFF")
 * RC<5:4> Digitale Ausgänge für "Bulg" & "Float" LEDs
 * Fosc=8MHz intern/stable
 * File:   MOSFET_Batterieladeregler.c
 * Author: Lasaros Goumas
 * Created on 05. April 2021, 20:34
 */

/* 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 BOREN = ON       //Brown-out Reset enabled
#pragma config WDTEN = OFF      //Watchdoc timer disabled
#pragma config PBADEN = OFF     //PORTB<5:0> configured digital 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 Datenport für das display
unsigned int volt;
unsigned int amper;
unsigned int lcd_info;          //DATA to be send to LCD
unsigned char temp;             //Temporäres LCD Register
int count;
int vzehntaus;                 //Spannungswertigkeit 10.000
int vtaus;                      //Spannungswertigkeit 1000
int vhund;                      //Spannungswertigkeit 100
int vzehn;                      //Spannungswertigkeit 10
```

```
int veins;                          //Spannungswertigkeit einer
int ataus;                          //Sromwertigkeit 1000
int ahund;                          //Stromwertigkeit 100
int azehn;                          //Stromwertigkeit 10
int aeins;                          //Stromwertigkeit einer


/*Funktionen

*****************************************************************************
/
void init_PIC (void){
    TRISA = 0b00000011;          //AN<1:0> inputs
    ANSELA = 0x03;               //AN<1:0> sind analoge eingänge
    TRISB = 0x00;                //PORTB Pins sind Ausgänge
    LATB = 0x00;
    TRISC = 0x00;                //PORTC pins sind Ausgänger
    PORTC = 0b00010000;          //S1=OFF;S3=OFF;Bulg=ON;Float=OFF
    OSCCON = 0b01100111;          //8MHz; Internal Oscilator; stable

}


void init_ADC (void){
     ADCON2 = 0b10010101;             //Right justified; Fosc/16; 4 TAD
    ADCON0bits.ADON = 0;        //ADC depowered
}


void volt_ascii (void){
    volt=volt*50;                    //volt*(3.200/1024+1.600/1024+
200/1024)
    volt=(volt>>4)+(volt>>5)+(volt>>8);
    lcd_info=volt<<2;
    vzehntaus = 0;                  //Zehntausener Spannungswertigkeit
    while (1)
    if (lcd_info>=10000){
        ++vzehntaus;
        lcd_info = lcd_info-10000;
    }
    else{
        vzehntaus = vzehntaus+0x30; //Spannungszehntausener in ASCII
        break;
    }

    vtaus = 0;                          //Tausener Spannungswertigkeit
    while (1)
    if (lcd_info>=1000){
        ++vtaus;
        lcd_info = lcd_info-1000;
    }
    else{
        vtaus = vtaus+0x30;            //Spannungstausener in ASCII
        break;
    }

    vhund = 0;                         //Hunderter Spannungswertigkeit
```

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

    vzehn = 0;                      //Zehner Spannungswertigkeit
    while (1)
    if (lcd_info>=10){
        ++vzehn;
        lcd_info = lcd_info-10;
    }
    else{
        vzehn = vzehn+0x30;         //Spannungszehner in ASCII
        break;
    }

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


void amper_ascii (void){
    amper=amper*50;                   //volt*(3.200/1024+1.600/1024+
200/1024)
    amper=(amper>>4)+(amper>>5)+(amper>>8);
    lcd_info=amper;
    ataus = 0;                         //Tausener Stromwertigkeit
    while (1)
    if (lcd_info>=3000){
        ++ataus;
        lcd_info = lcd_info-3000;
    }
    else{
        ataus = ataus+0x30;            //Stromtausener in ASCII
        break;
    }

    ahund = 0;                        //Hunderter Stromwertigkeit
    while (1)
    if (lcd_info>=300){
        ++ahund;
        lcd_info = lcd_info-300;
    }
    else{
```

```
        ahund = ahund+0x30;          //Stromhunderter in ASCII
        break;
    }

    azehn = 0;                       //Zehner Stromwertigkeit
    while (1)
    if (lcd_info>=30){
        ++azehn;
        lcd_info = lcd_info-30;
    }
    else{
        azehn = azehn+0x30;          //Stromzehner in ASCII
        break;
    }

    aeins = 0;                       //Einer Stromwertigkeit
    while (1)
    if (lcd_info>=3){
        ++aeins;
        lcd_info = lcd_info-3;
    }
    else{
        aeins = aeins+0x30;          //Stromeiner in ASCII
        break;
    }
}

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
    __delay_ms(2);                  //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
    __delay_ms(2);                   //2msec 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
    __delay_ms(2);                  //2msec 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
    __delay_ms(2);                 //2msec warten
}

void init_LCD (void){
    __delay_ms(50);          //Init LCD
    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 ausgabe (void){
    lcd_info=(0x80);
    write_command();            //Position in Zeile 1 (=0x80+0x00)
    lcd_info=(0x20);
    write_data();               //Leerzeichen
    lcd_info=('U');
```

```
    write_data();               //U
    lcd_info=('b');
    write_data();               //b
    lcd_info=('=');
    write_data();               //=
    lcd_info=(vzehntaus);
    write_data();               //Spannung zehntausener
    lcd_info=(vtaus);
    write_data();               //Spannung tausener
    lcd_info=('.');
    write_data();               //.
    lcd_info=(vhund);
    write_data();               //Spannung hunderte
    lcd_info=(vzehn);
    write_data();               //Spannung zehner
    lcd_info=(veins);
    write_data();               //Spannung einer
    lcd_info=(0x20);
    write_data();              //Leerzeichen
    lcd_info=('[');
    write_data();               //[
    lcd_info=('V');
    write_data();               //V
    lcd_info=(']');
    write_data();               //]
    lcd_info=0xC0;
    write_command ();            //Position 1 in Zeile 2 (=0x80+0x40)
    lcd_info=0x20;
    write_data ();               //Leerzeichen 1 in Zeile 2
    lcd_info=(ataus);
    write_data();               //Strom tausener
    lcd_info=(ahund);
    write_data();               //Strom hunderte
    lcd_info=('.');
    write_data();               //.
    lcd_info=(azehn);
    write_data();               //Strom zehner
    lcd_info=(aeins);
    write_data();               //Strom einer
    lcd_info=(0x20);
    write_data();              //Leerzeichen
    lcd_info=('[');
    write_data();               //[
    lcd_info=('A');
    write_data();               //A
    lcd_info=(']');
    write_data();               //]
    lcd_info = (0x02);
    write_command();           //Return cursor to home position
    }


/*Main Routine

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

```
init_PIC();
init_LCD();
init_ADC();

while(1){                          //"Bulk" Aufladung
ADCON0 = 0b00000001;                //AN0 selected; ADC Powered
__delay_us(20);                    //20µsec warten
ADCON0bits.GO = 0x01;              //ADC in progress
while(ADCON0bits.NOT_DONE);
volt = ADRES;                      //ADC Spannungsergebnis
ADCON0bits.ADON=0;                 //ADC depowered
if (volt>=0x02DA)                 //730*5/(0,25*1024)>=14,26V
break;

else{
volt_ascii();                      //Spannugswerte in ASCII
ADCON0 = 0b00000101;               //AN1 selected; ADC Powered
__delay_us(20);                    //20µsec warten
ADCON0bits.GO = 0x01;              //ADC in progress
while(ADCON0bits.NOT_DONE);
amper = ADRES;                     //ADC Stromergebnis
ADCON0bits.ADON=0;                 //ADC depowered
amper_ascii();                     //Stromwerte in ASCII
ausgabe();
for (count=0; count<=4; count++)__delay_ms(250);  //Warte 1Sec
}
}

PORTC = 0b00100001;                //S1=ON;S3=OFF;Bulk=OFF;Float=ON
                                   //"float" Aufladung
while(1){
ADCON0 = 0b00000001;                //AN0 selected; ADC Powered
__delay_us(20);                    //20µsec warten
ADCON0bits.GO = 0x01;              //ADC in progress
while(ADCON0bits.NOT_DONE);
volt = ADRES;                      //ADC Spannungsergebnis
ADCON0bits.ADON=0;                 //ADC depowered
if (volt>=0x02BE){                 //702*5/(0,25*1024)=13,71V
PORTC = 0b00100101;                //S1=ON;S3=ON;Bulk=OFF;Float=ON
                                   //Aufladung "Blocked"
volt_ascii();                      //Spannugswerte in ASCII
ADCON0 = 0b00000101;               //AN1 selected; ADC Powered
__delay_us(20);                    //20µsec warten
ADCON0bits.GO = 0x01;              //ADC in progress
while(ADCON0bits.NOT_DONE);
amper = ADRES;                     //ADC Stromergebnis
ADCON0bits.ADON=0;                 //ADC depowered
amper_ascii();                     //Stromwerte in ASCII
ausgabe();
for (count=0; count<=20; count++)__delay_ms(250);  //Warte 5Sec
}

else{
PORTC = 0b00100001;                 //S1=ON;S3=OFF;Bulg=OFF;Float=ON
                                   //"float" Aufladung
volt_ascii();                      //Spannugswerte in ASCII
ADCON0 = 0b00000101;               //AN1 selected; ADC Powered
```

```
__delay_us(20);                 //20µsec warten
ADCON0bits.GO = 0x01;           //ADC in progress
while(ADCON0bits.NOT_DONE);
amper = ADRES;                  //ADC Stromergebnis
ADCON0bits.ADON=0;              //ADC depowered
amper_ascii();                  //Stromwerte in ASCII
ausgabe();
for (count=0; count<=2; count++)__delay_ms(250);  //Warte 0,5Sec
}
}
}
```