LEO_DE_21506

LEO PROGRAMMIERBOARD

BETRIEBSANLEITUNG

ORBIT CONTROLS AG
Zürcherstrasse 137
CH-8952 Schlieren/ZH

Tel: + 41 44 730 2753
Fax: + 41 44 730 2783

info@orbitcontrols.ch
www.orbitcontrols.ch

Vor dem Einschalten

Überzeugen Sie sich, ob Ihre Sendung das richtige Gerät Orbit Controls Modell LEO Board beinhaltet, einschliesslich einer Betriebsanleitung Leo Board.
Vor dem Einschalten des Gerätes überprüfen Sie die Anschlüsse und die Versorgungsspannung. Ein falsch angeschlossenes Gerät kann beschädigt werden und damit auch die mitverbundene Folgeelektronik. Für falsche Handhabung wird jede Haftung abgelehnt.

ZU BEACHTEN

Dieses Gerät wurde sorgfältig verpackt. Falls es bei Ihnen in beschädigtem Zustand eintrifft, benachrichtigen Sie unverzüglich den Orbit Controls Kundendienst (Tel: +41 44 730 2753 oder Fax: +41 44 730 2783) und nehmen Sie einen Schadenrapport auf, welchen Sie auch von der Transportgesellschaft unterschreiben lassen. Bewahren Sie bitte das Verpackungsmaterial für eventuelle Reklamationen auf.

Unpacking Instructions

Remove the Packing List and verify that you have received all equipment, including the following:
Orbit Controls Model LEO Board.
Operator's Manual LEO Board.
If you have any questions about the shipment, please call the Orbit Controls Customer Service Department.

NOTE

When you receive the shipment, inspect the container and equipment for signs of damage. Note any evidence of rough handling in transit. Immediately report any damage to the Orbit Controls customer service, Phone +4144 730 2753 or Fax +4144 730 2783 and to the shipping agent.
The carrier will not honour damage claims unless all shipping material is saved for inspection. After examining and removing contents, save packing material and carton in event the reshipment is necessary.

INHALTSVERZEICHNIS

1 PROGRAMMIERBOARD LEO

- ✓ 24 Bit Analog/Digital Wandler
- ✓ 12 Bit Digital/Analog Wandler
- ✓ Frequenzzähler 0.2Hz-20KHz
- ✓ 64x128 Pixel LCD
- ✓ Atmel 8-Bit Mikrokontroller AT89C51ED2
- ✓ RS232 und USB Windows 8 Programmierung
- ✓ Relais, Piepser, 5 Tasten, 3 LED
- ✓ DC - Versorgung

Das LEO Programmierboard ist für Schüler und Studenten vorgesehen, welche ihre Programmierkenntnisse erweitern möchten. Das Board verfügt über einen Analogeingang, einen Frequenzeingang und einen Analogausgang.

Das LEO-Board besitzt ein LC-Grafikdisplay (PG12864J) mit 128x64 Pixel. Darunter sind fünf Taster positioniert, drei SMD LEDs, ein Piepser und ein Relais welche zu je einem Port geführt sind. Das Relais ist außerdem auf eine Schraubklemme hinausgeführt.

Gleichzeitig können alle bereits verdrahtete Ports über Jumper Pins für externe Anwendungen verwendet werden.
Jegliche Versorgungsspannungen sowie A/D und D/A Spannungen sind an Schraubklemmen hinausgeführt und können auf diese Weise weiter verwendet werden.

Programmiert wird der Mikrokontroller über die RS232 Schnittstelle und dem mitgelieferten, oder von der Website herunterladbare, Software FLIP von Atmel. Die Programmiersprache ist “ C/C++ “.

1.1 Programmieren

Um das Leo Board zu programmieren werden die Standard serielle COM-Schnittstelle und die FLIP Software von Atmel benötigt.
Um in den Programmiermodus zu gelangen müssen die Taster RESET (SW1) UND BOOT (SW2) gedrückt gehalten werden, nachher RESET und BOOT nacheinander loslassen. Erst danach kann der Mikrokontroller mit FLIP verbunden werden.

2 TECHNISCHE DATEN

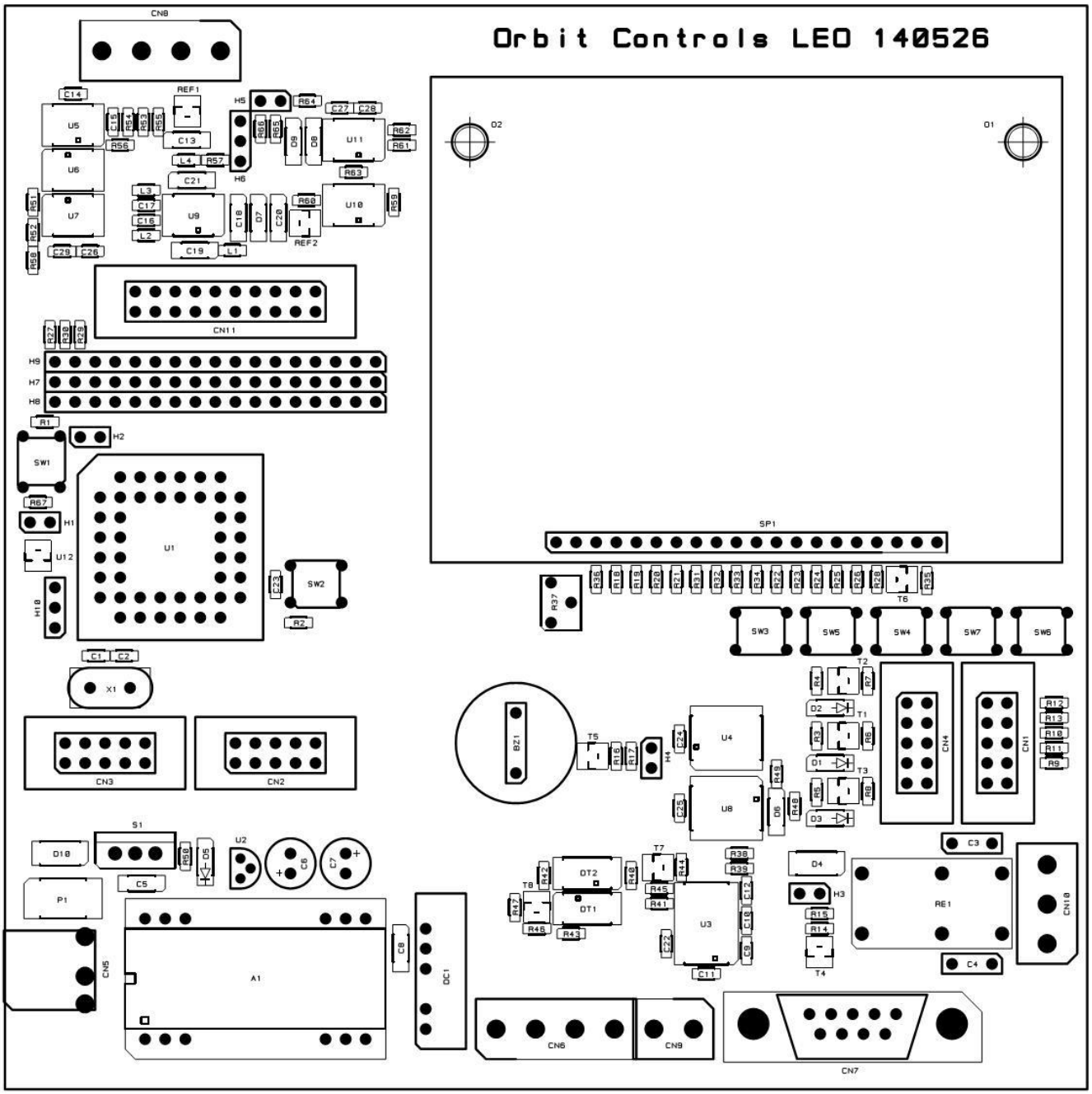
Eingang:	Spannung:	bis ±10V DC.
	Strom:	bis 20mA.
LTC2400:	24 Bit, bipolar, Messzyklusdauer 63ms.	
	Integral Nonlinearity:	± 0.006% v. Bereich
	Zero Error:	± 0.0168% v. Bereich
	Rollover Error:	± 0.032% v. Bereich
	Tempco:	Temperaturkoeffizient ±10ppm°C
	Linearität:	± (1 LSB + 1 Digit).
Ausgang:	Spannung:	0...14V mit 16 Bit Auflösung.
	Genauigkeit:	± (0.05% vom Wert + 0.1% vom Bereich)
LTC1595 :	16Bit D/A-Wandler, DNL und INL: 1LSB Max, SRI	
AT89C51ED2:	44-Pin, 8Bit-Flash, 80C52/8051 kompatibel	
Relais:	5A/250VAC	
PG12864J:	128x64 Pixel, LCD grau, 8Bit parallel Dateneingang, LED Hintergrundbeleuchtung	
Hauptversorgung :	Stecker CN5 :	Netzadapter 100-240V, 50/60Hz / 24V-330mA DC
Spannungen:	Stecker CN6 :	- Pin1: VCC (5V) (Max. 300mA) - Pin2: GND - Pin3: -15V - Pin4: +15V
A/D & D/A:	Stecker CN8 :	- Pin1: U-Input - Pin2: GND - Pin3: GND - Pin4: U-Output
Frequenz:	Stecker CN9 :	- Pin1: Frequenz Input - - Pin2: Frequenz Input +
Platine:	DIN 1.5x137x137 mm (HxBxT).	
	Anschlüsse:	Schraubklemme.

.

3 WAHL VON MESSBEREICH

Jumper	20mA	1V	10V	wählbar 10mV-1V
H5	1+2	offen	offen	offen
H6	2+3	2+3	1+2	2+3
R63 (G = Verstärkung)	offen	offen	offen	R=50k/G-1

4 BESTÜCKUNGSPLAN

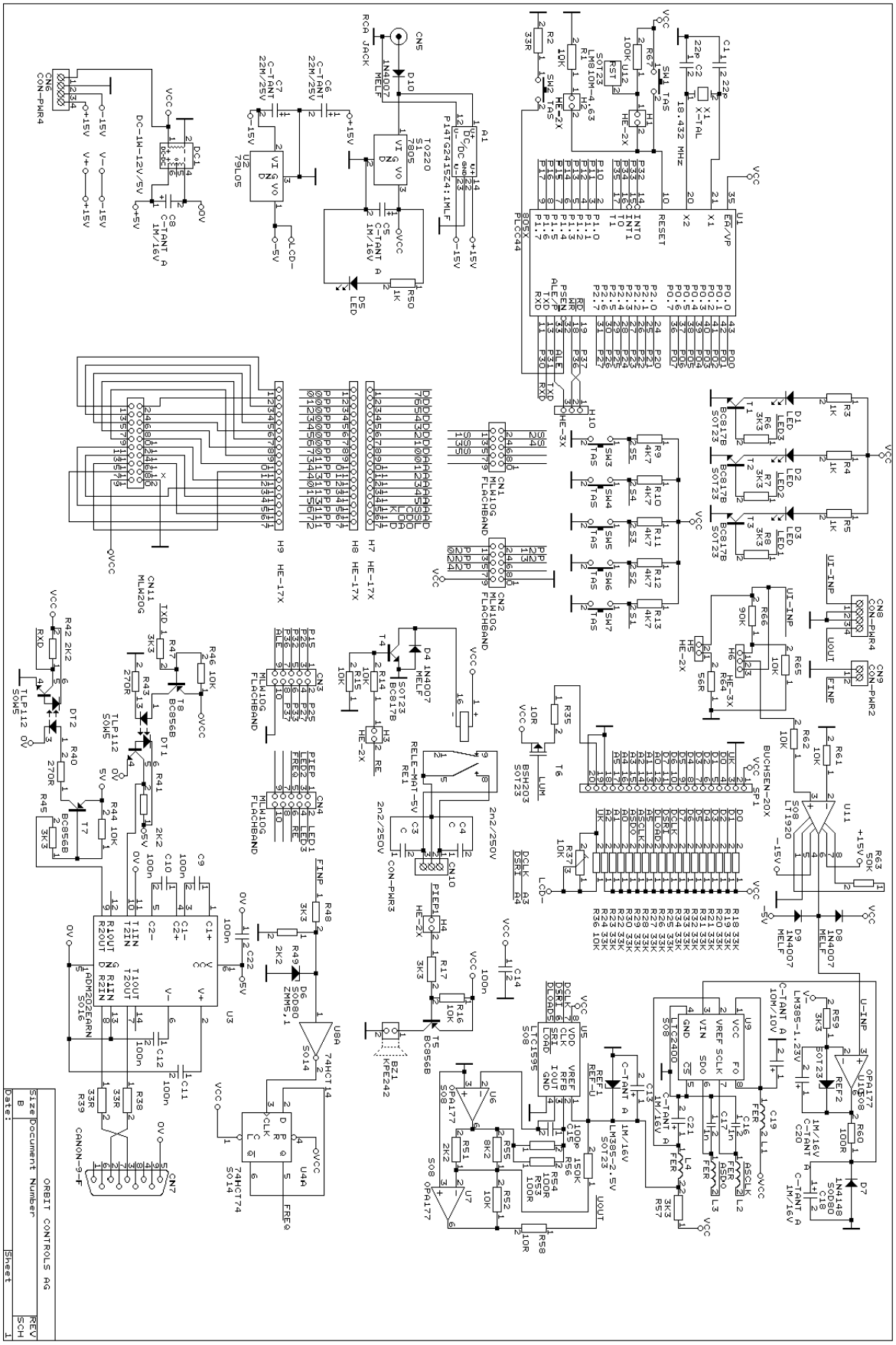


5 STANDARD PORT BELEGUNG (AT89C51ED2)

AT89C51ED2 (Mikrokontroller)		PG12864J (Display)		LTC1595 (D/A-Konverter)	
Port	**Pin**	**Bezeichnung**	**Pin**	**Bez.**	**Pin**
P0.0-P0.7	Pin 43-36	**DB7-DB0**	Pin 4-11		
P1.0	Pin 2	R/W	Pin 15	DCLK	Pin 7
P1.1	Pin 3	D/I	Pin 16	DSRI	Pin 6
P1.2	Pin 4			DLOAD	Pin 5
P1.3	Pin 5	CS1	Pin 12		
P1.4	Pin 6	CS2	Pin 13		
P3.4	Pin 16	RST	Pin 14		
P3.5	Pin 17	Enable	Pin 17		
P3.6	Pin 18	BKL (LUM)	Pin 19		

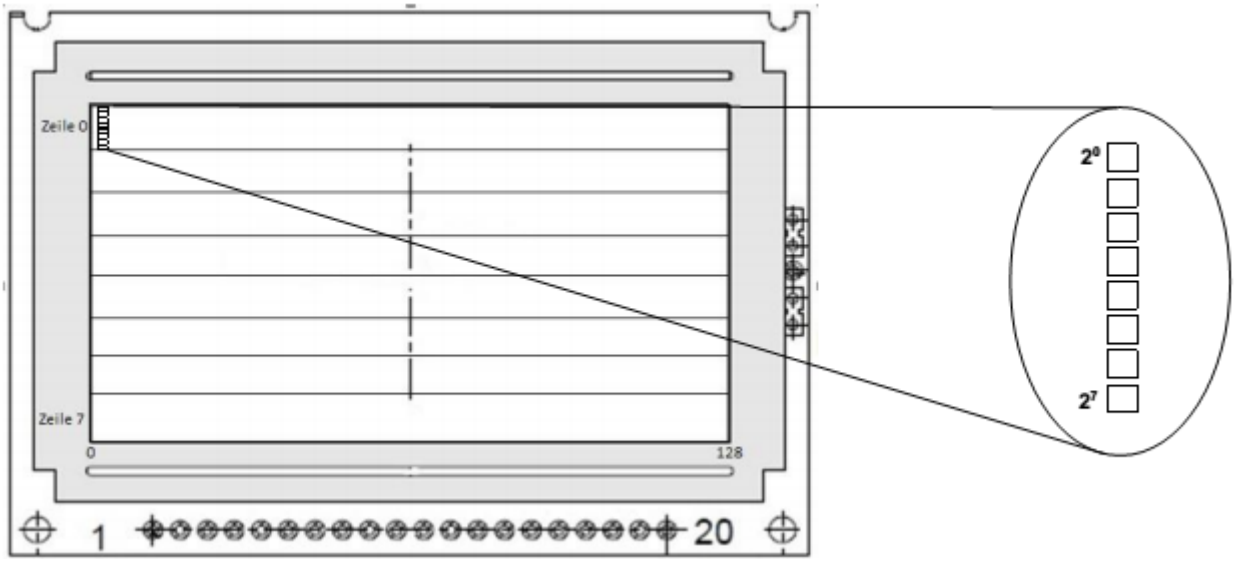
		LTC2400 (A/D-Konverter)	
		Bezeichnung	**Pin**
P1.6	Pin 8	ASCLK	Pin 7
P1.7	Pin 9	ASDO	Pin 6
P1.5	Pin 7	Piepser BZ1 (aktiv LOW & H4 geschlossen)	
P3.0/RxD	Pin 11		
P3.1/TxD	Pin 13		
P3.2	Pin 14	Frequenzeingang	
P3.3	Pin 15	RELAIS (aktiv High & H3 geschlossen)	
P2.0	Pin 24	S1 (SW7)	
P2.1	Pin 25	S2 (SW6)	
P2.2	Pin 26	S3 (SW5)	
P2.3	Pin 27	S4 (SW4)	
P2.4	Pin 28	S5 (SW3)	
P2.5	Pin 29	LED1	
P2.6	Pin 30	LED2	
P2.7	Pin 31	LED3	

6 SCHEMA

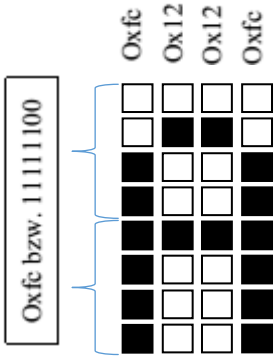


7 ERLÄUTERUNGEN ZUM LC-DISPLAY (PG12864J)

Als erstes muss man verstehen wie ein LCD-Display aufgebaut ist. Es besteht aus acht so genannten Zeilen mit je 8 Pixel Höhe und je 128 Pixel Breite. Das Display ist in der Mitte aufgeteilt und wird je mit einem Chipselect ein bzw. ausgeschaltet. Das heisst, die erste Hälfte ist von 0-63 definiert und die zweite Hälfte fängt bei 64 an und geht bis 128 weiter. Wenn man die Höhen und Breiten betrachtet entsteht dadurch der Begriff 64x128 Pixeldisplay.



Weiter ist wichtig zu verstehen, dass die acht Pixel Höhe jeder Zeile als Speicherinhalt von oben nach unten gelesen werden muss (siehe Abbildung 2). Das ist für die spätere Programmierung wichtig. Sprich die oberste Pixelzeile oder –reihe entspricht dem Wert von 20 und die unterste 27.
Möchte man nun etwas auf das LCD-Display ausgeben, muss man diese von Pixelbyte zu Pixelbyte selbst definieren. Will man eine ganze 8bit-Reihe bzw. Pixelbyte ausfüllen entspräche das der Zahl 11111111 in binärer Form oder 0xff in hexadezimaler Form.



Beispiel um ein ‚A' auf das LCD zu zeichnen.
Falls nur auf eine Zeile geschrieben wird, kann ich die Höhe nicht über 8 Pixels sein, jedoch kann die Breite Selbst gewählt werden.

8 BEISPIEL PROGRAMME

8.1 Löschen und Beschreiben des LC-Displays

```
//////////////////////////////////////////
//////////////////////////////////////////
//                                      //
//  Grafik Display Ansteuerung          //
//  Projektarbeit                       //
//  13.04.2015                          //
//                                      //
//////////////////////////////////////////

#include <at89c51xd2.h>


//Pin Definitionen
//////////////////////////////////////////
#define LCD_DATA     P0          // Data-Bus P0.0 bis P0.7
#define LCD_RW       P1_0        // Data Read/Write
#define LCD_DI       P1_1        // Register Selection input
#define LCD_CS1      P1_3        // Chipselect (Links)
#define LCD_CS2      P1_4        // Chipselect (Rechts)
#define LCD_EN       P3_5        // LCD Enable (Einschalten)
#define LCD_RST      P3_4        // Reset (Zurücksetzung)
// #define LCD_BKL      P3_6         // LCD Hintergrundbeleuchtung
//ADC (SPI)
#define ASCLK    P1_6            // Digital Clock
#define ASDO     P1_7            // Digital Output

void lcd_init()                  //LCD Initialisierung
{
    LCD_RW  = 0;                 // schreiben
    LCD_DI  = 0;                 // Instruktionen
    LCD_EN  = 0;                 // LCD-Enable
    LCD_CS1 = 0;                 // Chipselect1 ein
    LCD_CS2 = 0;                 // Chipselect2 ein
    LCD_RST = 1;                 // LCD ein

}

int p=0,z=0;                     //Zaehlvariable

// Verzoegerungs-Funktion
void delay_ms(long x)            //Verzögerunsfunktion
{
    long i=0;
    for (i; i<x; i++){;}
}
```

```
void lcd_send_data()
{
    delay ms(33);               //Warten
    delay ms(33);               //Warten
    LCD EN = 1;                 //LCD enable
    delay ms(33);               //Warten
    delay ms(33);               //Warten
    LCD_EN = 0;                 //LCD disable
}

void lcd_page(page)
{
    LCD_CS1=0;                  //Chipselect1 ein
    LCD_CS2=0;                  //Chipselect2 ein
    LCD_DI = 0;                 //Display-RAM im "Einstellungs-Modus"
    LCD DATA=(0xB8|page);       //Hier wird die "page" gesetzt bzw. es wird mit page 7 begonnen.
                                //Siehe auch Seite 9 des Display-Datenblatts
    lcd send data();            //Datenbus P0 wird in den RAM des Displays geschrieben
    LCD_DI = 1;                 //Display-RAM im "Empfang-Modus" (siehe S.10)
}

void lcd_zeile(zeile)
{
    LCD CS1=0;                  //Chipselect1 ein
    LCD CS2=0;                  //Chipselect2 ein
    LCD DI = 0;                 //Display-RAM im "Einstellungs-Modus"
    LCD_DATA=(0x40|zeile);      //Hier wird die "Zeile" gesetzt bzw. es wird mit Zeile 0 begonnen.
                                //Siehe auch Seite 9 des Display-Datenblatts
    lcd_send_data();            //Datenbus P0 wird in den RAM des Displays geschrieben
    LCD_DI = 1;                 //Display-RAM im "Empfang-Modus" (siehe S.10)
}



void lcd_clear()                //Subroutine um das Display zu loeschen
{
    LCD CS1=0;                  //Chipselect1 ein
    LCD_CS2=0;                  //Chipselect2 ein

    for(p=7;p>=0;p--)           //Acht Durchgaenge für acht "pages" des Displays
    {
        lcd_page(p);            //Page setzen
        lcd_zeile(z);           //Zeilenposition setzen
```

```
        while(z<64)                 //Endlosschleife
        {
            LCD DATA=0x00;          //0x00 in den Datenbus P0 schreiben
            LCD DI = 1;             //Display-RAM im "Empfang-Modus" (siehe S.10)
            lcd send data();        //Datenbus P0 wird in den RAM des Displays geschrieben
            LCD DI = 0;             //Display-RAM im "Einstellungs-Modus"
            z++;                    //Zaehlvariable inkrementieren
        }
        z=0;                        //Zaehlvariable zuruecksetzen
    }
}

void main()                         //Hauptprogramm
{

    lcd_init();                     //LCD initalisieren
    LCD DATA=0x3f;                  //LCD on-Befehl
    lcd_send_data();                //Daten an Datenbus P0 senden


    lcd_clear();                    //LCD loeschen

        for(p=0;p<8;p++)            //Acht Durchgaenge für acht "pages" des Displays
        {
            lcd page(p);            //Page setzen
            lcd_zeile(z);           //Zeilenposition setzen
            while(z<64)             //64 Durchgaenge
            {
                LCD_DATA=0xff;      //Ganze Zeile ausmalen
                lcd_send_data();//Datenbus P0 wird in den RAM des Displays geschrieben
                z++;                //Zaehlvariable inkrementieren
            }
            z=0;                    //Zaehlvariable zuruecksetzen
        }
}
```

8.2 A/D- Konvertierung mit LCD-Anzeige

```
#include <at89c51xd2.h>

// === MCU INTERRUPT SOURCES ===========================================================
============

#define SPI     9   // Serial Port Interface (004Bh)

//Pin Definitionen
//////////////////////////
#define LCD DATA    P0          // Data-Bus P0.0 bis P0.7
#define LCD RW      P1 0        // Data Read/Write
#define LCD DI      P1 1        // Register Selection input
#define LCD CS1     P1 3        // Chipselect (Links)
#define LCD CS2     P1 4        // Chipselect (Rechts)
#define LCD EN      P3 5        // LCD Enable (Einschalten)
#define LCD RST     P3 4        // Reset (Zurücksetzung)
#define LED1        P2_5
#define LED2        P2_7
#define S1          P2_0
#define BKL         P3_6

//ADC (SPI)
#define ASCLK       P1 6        // Digital Clock
#define ASDO        P1_7        // Digital Output



// Byte Zuordnungen

#define BYTE3(v)    (*((unsigned char *) (&v)+0))    //uebergeben wird der Wert in 'resu
lt',
#define BYTE2(v)    (*((unsigned char *) (&v)+1))
#define BYTE1(v)    (*((unsigned char *) (&v)+2))
#define BYTE0(v)    (*((unsigned char *) (&v)+3))


// ADC related variables and constants
#define ADC_MIN            0                    // initialisieren von Null Wert (0V
)
#define ADC_MAX     16777216                    // initialisieren von Max Wert (2.5
V)
#define ADC_STEP    13421773                    // initialisieren von Bereich  (2.0
V)


unsigned long int adc value=0;
unsigned long int adc value 3=0;
unsigned char adc_outofrange=0;

void init()
{
    // ADC
    // configure SPI
    SPCON=0x6C;     // SPI on, slave mode, no SS, idle H, sampling auf pos.Flanke    SP
EN=1 (SPI ein) SSDIS=1 (slave mode,no SS) CPOL=1 (Clock polarity,idle H) CPHA=1(Clock p
hase,sampling on rising edge)
    IPL1|=0x04;     // SPI interrupt hoechste Priorität
    IPH1|=0x04;     //  ^^
```

```
    IEN1=0x04;      // SPI interrupt ein

    EA=1;           // global interrupt ein
    BKL=1;          // Backlight off
    S1=1;           // Schalter off
}

// Verzoegerungs-Funktion
//////////////////////////
void delay_ms(long x)           //Verzoegerungsfunktion
{

    long u=0,O=0;

    for (u=0; u<x; u++)         //Verzoerung = 1ms * uebergebener Wert
    {
        for(O=0;O<33;O++);      //1ms Verzoegerungszeit
    };
}

void lcd_send_data()
{
    delay ms(1);                //Warten
    delay ms(1);                //Warten
    LCD EN = 1;                 //LCD enable
    delay_ms(1);                //Warten
    delay_ms(1);                //Warten
    LCD_EN = 0;                 //LCD disable
}

void lcd_init()                 //LCD Initialisierung
{
    LCD RW  = 0;                // schreiben
    LCD DI  = 0;                // Instruktionen
    LCD EN  = 0;                // LCD-Enable
    LCD CS1 = 0;                // Chipselect1 ein
    LCD CS2 = 0;                // Chipselect2 ein
    LCD_RST = 1;                // LCD ein

    LCD_DATA=0x3f;              //LCD on-Befehl
    lcd_send_data();            //Daten an Datenbus P0 senden

}

void ISR SPI (void) interrupt SPI {
    unsigned long int result;               //Variable deklariert
    unsigned char tmp;                      //Variable deklariert

    if (SPSTA&0x80) {
        BYTE3(result) = SPDAT;              // 4. Byte in 'result' speichern
        while (!(SPSTA&0x80));              // auf nächstes Byte warten...
        BYTE2(result) = SPDAT;              // 3. Byte in 'result' speichern
        while (!(SPSTA&0x80));              // auf nächstes Byte warten...
        BYTE1(result) = SPDAT;              // 2. Byte in 'result' speichern
        while (!(SPSTA&0x80));              // auf nächstes Byte warten...
        BYTE0(result) = SPDAT;              // 1. Bytein 'result' speichern

        // Bereich check
        tmp = result>>28;                   //32Bit Variabel 28 mal schieben
                                            //um die Status Bytes zu erhalten
        if (tmp == 1)                       // Falls Ueberbereich oder Unterbereich
            adc_outofrange = 1;             // Variable auf 1 setzen
        else
            adc_outofrange = 0;             // Sonst auf 0 setzen

        adc_value = (result>>4)&0x00FFFFFF; //von der 32 Bit Variable nur die relevante
n
                                            //24 Bit in adc_value abspeichern
```

```
        LED2=~LED2;                         //LED toggeln lassen

       }

}

int p=0,z=0;                        //Zaehlvariablen

//Y-Achsen-Positionierung
void lcd_page(page)
{
    LCD_CS1=0;                      //Chipselect1 ein
    LCD_CS2=0;                      //Chipselect2 ein
    LCD_DI = 0;                     //Display-RAM im "Einstellungs-Modus"
    LCD_DATA=(0xB8|page);           //Hier wird die "page" gesetzt bzw. es wird mit page 7
begonnen.
                                    //Siehe auch Seite 9 des Display-Datenblatts
    lcd_send_data();                //Datenbus P0 wird in den RAM des Displays geschrieben
    LCD_DI = 1;                     //Display-RAM im "Empfang-Modus" (siehe S.10)
}
//X-Achsen-Positionierung
void lcd_zeile(zeile)
{
    LCD_CS1=0;                      //Chipselect1 ein
    LCD_CS2=0;                      //Chipselect2 ein
    LCD_DI = 0;                     //Display-RAM im "Einstellungs-Modus"
    LCD_DATA=(0x40|zeile);          //Hier wird die "Zeile" gesetzt bzw. es wird mit Zeile
0 begonnen.
                                    //Siehe auch Seite 9 des Display-Datenblatts
    lcd_send_data();                //Datenbus P0 wird in den RAM des Displays geschrieben
    LCD_DI = 1;                     //Display-RAM im "Empfang-Modus" (siehe S.10)
}

//Array fuer ganzes Alphabet
static code int letter[112]={
    0xfc, 0x12, 0x12, 0xfc, // A
    0xfe, 0x92, 0x92, 0x6c, // B
    0x7c, 0x82, 0x82, 0x82, // C
    0xfe, 0x82, 0x44, 0x38, // D
    0xfe, 0x92, 0x92, 0x92, // E
    0xfe, 0x12, 0x12, 0x12, // F
    0x7c, 0x82, 0x92, 0x72, // G
    0xfe, 0x10, 0x10, 0xfe, // H
    0x82, 0xfe, 0x82, 0x00, // I
    0x40, 0x80, 0x80, 0x7e, // J
    0xfe, 0x28, 0x44, 0x82, // K
    0xfe, 0x80, 0x80, 0x80, // L
    0xfe, 0x04, 0x04, 0xfe, // M
    0xfe, 0x08, 0x10, 0xfe, // N
    0x7c, 0x82, 0x82, 0x7c, // O
    0xfe, 0x12, 0x12, 0x0c, // P
    0x7c, 0x82, 0xa2, 0x5c, // Q
    0xfe, 0x32, 0x52, 0x8c, // R
    0x8c, 0x92, 0x92, 0x62, // S
    0x02, 0xfe, 0x02, 0x00, // T
    0x7e, 0x80, 0x80, 0x7e, // U
    0x3e, 0xc0, 0xc0, 0x3e, // V
    0xfe, 0x40, 0x40, 0xfe, // W
    0xee, 0x10, 0x10, 0xee, // X
    0x06, 0xf8, 0x06, 0x00, // Y
    0xe2, 0x92, 0x92, 0x8e, // Z

};
//Array fuer Zahlen von 1-9
static code int number[40]={
    0x7c, 0x82, 0x82, 0x7c, // 0
    0x08, 0x04, 0xfe, 0x00, // 1
    0xe2, 0x92, 0x92, 0x8e, // Z
```

```
    0x82, 0x92, 0x92, 0x7c, // 3
    0x18, 0x14, 0xfe, 0x10, // 4
    0x9e, 0x92, 0x92, 0x72, // 5
    0xfc, 0x92, 0x92, 0xf2, // 6
    0x02, 0x02, 0x02, 0xfe, // 7
    0x6c, 0x92, 0x92, 0x6c, // 8
    0x9e, 0x92, 0x92, 0xfe, // 9

};

//Fuer Textausgabe
void LCD_text (char *string)          //Speicherplatz von String wird übernommen
{
    int addr;                         //Varable deklariert
    while(*string)                    //Solange nicht Stringeende erreicht
    {
        addr = (*string++ - 0x41)*4;  //Berechnung: ascii-code des Buchstaben minus 0x4
1 mal 4
                                      //ergibt Anfangsstelle im 'letter'-array.
        LCD_DATA=(letter[addr+0]);    //in P0 wird erstes 'Byte' gespeichert
        lcd_send_data();              //P0 in LCD-RAM uebergeben
        LCD_DATA=(letter[addr+1]);    //in P0 wird zweites 'Byte' gespeichert
        lcd_send_data();              //P0 in LCD-RAM uebergeben
        LCD_DATA=(letter[addr+2]);    //in P0 wird drittes 'Byte' gespeichert
        lcd_send_data();              //P0 in LCD-RAM uebergeben
        LCD_DATA=(letter[addr+3]);    //in P0 wird viertes 'Byte' gespeichert
        lcd_send_data();              //P0 in LCD-RAM uebergeben
        LCD_DATA=0x00;                //in P0 wird ein Abstand gespeichert
        lcd_send_data();              //P0 in LCD-RAM uebergeben
    }
}

//Fuer Nummerausgabe
void LCD_number (char *string)        //Speicherplatz von String wird übernommen
{
    int addr;                         //Varable deklariert
    while(*string)                    //Solange nicht Stringeende erreicht
    {
        addr = (*string++ - 0x30)*4; //Berechnung: ascii-code des Buchstaben minus 0x30
 mal 4
                                      //ergibt Anfangsstelle im 'number'-array.
        LCD_DATA=(number[addr+0]);    //in P0 wird erstes 'Byte' gespeichert
        lcd_send_data();              //P0 in LCD-RAM uebergeben
        LCD_DATA=(number[addr+1]);    //in P0 wird zweites 'Byte' gespeichert
        lcd_send_data();              //P0 in LCD-RAM uebergeben
        LCD_DATA=(number[addr+2]);    //in P0 wird drittes 'Byte' gespeichert
        lcd_send_data();              //P0 in LCD-RAM uebergeben
        LCD_DATA=(number[addr+3]);    //in P0 wird viertes 'Byte' gespeichert
        lcd_send_data();              //P0 in LCD-RAM uebergeben
        LCD_DATA=0x00;                //in P0 wird ein Abstand gespeichert
        lcd_send_data();              //P0 in LCD-RAM uebergeben
    }
}

void number_check(int x)              //Digit von ADC-Wert wird uebernommen
{
    int y=1;                          //Variable deklariert

        if(x>=10)                     //ist Digit groesser gleich 10
        {
            y=2;                      //y=2
            x=x/10;                   //Digit Wert durch 10 teilen
        }

        for(y=1*y;y>0;y--)            //Falls Digit groesser gleich 10
        {                             //wird die Schleife 2 mal durchgefuehrt
            switch(x)
            {
```

```
                case 0: LCD number("0"); break; //je nach Digit-Wert wird
                case 1: LCD number("1"); break; //gleicher Digit-String
                case 2: LCD number("2"); break; //uebergeben.
                case 3: LCD number("3"); break;
                case 4: LCD number("4"); break;
                case 5: LCD number("5"); break;
                case 6: LCD number("6"); break;
                case 7: LCD number("7"); break;
                case 8: LCD number("8"); break;
                case 9: LCD_number("9"); break;
                default: break;
            }
        x--;                                    //Digit-Wert dekrementieren
        }
}

//Abstand zeichnen
void space()
{
    int c;                      //Variable deklariert

    for (c=0; c<4 ; c++)        //Abstand zeichnen
    {
        LCD DATA=0x00;          //in P0 wird ein Abstand gespeichert
        lcd_send_data();        //P0 in LCD-RAM uebergeben
    }
}

//Punkt zeichnen
void dot()
{
    int c;                      //Variable deklariert

    for (c=0; c<2 ; c++)        //Abstand zeichnen
    {
        LCD DATA=0xc0;          //in P0 wird ein Punkt gespeichert
        lcd_send_data();        //P0 in LCD-RAM uebergeben
    }
    LCD_DATA=0x00;              //in P0 wird ein Abstand gespeichert
    lcd_send_data();            //P0 in LCD-RAM uebergeben
}

void LCD_print_adc (unsigned long int number, int out_of_range) {  // number ist ADC-We
rt

    int i=0,r=0;                //Zaehlvariable deklariert
    unsigned long int tmp;

    lcd page(3);                //Page setzen
    lcd zeile(64);              //Zeilenposition setzen
    LCD_CS1=1;                  //linke Displayhaelfte abschalten

        tmp=number;             //tmp = ADC-Wert

        if (out_of_range==1||(tmp<=1))  //Ist der Ueber-/Unterbereich vorhanden?
        {
            lcd page(3);            //Page setzen
            lcd zeile(64);          //Zeilenposition setzen
            LCD CS1=1;              //linke Displayhaelfte abschalten
            number check(0);        //eine Null wird gezeichnet
            dot();                  //ein Dezimalpunkt wird gezeichnet
           for(i=0;i<3;i++)         //3-Schleifendurchgaenge
           {
            number_check(0);        //eine Null wird gezeichnet
           }
           delay_ms(1000);          //1s wird gewartet
        }
```

```
        else if(!out_of_range)      //Ansonsten
        {

            tmp*=10;                //tmp verzehnfachen
            while(tmp>=ADC_STEP)    //Solange tmp grösser als  Wert von ca 2.0V ist

            {
                i++;                //Zaehlvariable inkrementieren
        // Dieser Schritt hier zaehlt
                tmp-=ADC_STEP;      //tmp -  2.0V
        // die "Einer" also 1-10V
            }
            number_check(i);        //Digit-Wert uebergeben und zeichnen
            i=0;                    //Zaehlvariable nullen

            dot();                  //Dezimalpunkt zeichnen

            tmp*=10;                //tmp verzehnfachen
            while(tmp>=ADC_STEP)    //Solange tmp grösser als  Wert von ca 2.0V ist
            {
                i++;                //Zaehlvariable inkrementieren
        // Dieser Schritt hier zaehlt
                tmp-=ADC_STEP;      //tmp -  2.0V
        // die "Zehntel" also 0.1-0.9V
            }
            number_check(i);        //Digit-Wert uebergeben und zeichnen
            i=0;                    //Zaehlvariable nullen

            tmp*=10;                //tmp verzehnfachen
            while(tmp>=ADC_STEP)    //Solange tmp grösser als  Wert von ca 2.0V ist
            {
                i++;                //Zaehlvariable inkrementieren
       // Dieser Schritt hier zaehlt
                tmp-=ADC STEP;      //tmp -  2.0V
       // die "Hundertstel" also 0.01-0.09V
            }
            number check(i);        //Digit-Wert uebergeben und zeichnen
            i=0;                    //Zaehlvariable nullen

            tmp*=10;                //tmp verzehnfachen
            while(tmp>=ADC_STEP)    //Solange tmp grösser als  Wert von ca 2.0V ist
            {
                i++;                //Zaehlvariable inkrementieren
       // Dieser Schritt hier zaehlt
                tmp-=ADC STEP;      //tmp -  2.0V
       // die "Tausendstel" also 0.001-0.009V
            }
            number check(i);         //Digit-Wert uebergeben und zeichnen
            i=0;                     //Zaehlvariable nullen
            }
}

void lcd_start()
{
    int c=0;                    //Zaehlvariable

    lcd page(0);                //Page setzen
    lcd zeile(0);               //Zeilenposition setzen
    LCD CS2=1;                  //rechte Displayhaelfte abschalten
    LCD text("ORBIT");          //Textausgabe
    lcd page(1);                //Page setzen
    lcd zeile(0);               //Zeilenposition setzen
    LCD CS2=1;                  //rechte Displayhaelfte abschalten
    LCD text("CONTROLS");       //Textausgabe
    space();
    LCD_text("AG");             //Textausgabe
    lcd_page(3);                //Page setzen
    lcd_zeile(0);               //Zeilenposition setzen
```

```
    LCD_CS2=1;                  //rechte Displayhaelfte abschalten
    LCD_text("IPA");            //Textausgabe
    lcd_page(4);                //Page setzen
    lcd_zeile(0);               //Zeilenposition setzen
    LCD_CS2=1;                  //rechte Displayhaelfte abschalten
    LCD_text("LEO");            //Textausgabe
    space();
    LCD_text("MUELLER");        //Textausgabe
    lcd_page(7);                //Page setzen
    lcd_zeile(0);               //Zeilenposition setzen
    LCD_CS2=1;                  //rechte Displayhaelfte abschalten
    LCD_number("15");           //Nummerausgabe
    dot();                      //Punktausgabe

    LCD_number("04");           //Nummerausgabe
    dot();                      //Punktausgabe
    LCD_number("2015");         //Nummerausgabe


}


//Subroutine um das Display zu loeschen
void lcd_clear()
{
    LCD_CS1=0;                  //Chipselect1 ein
    LCD_CS2=0;                  //Chipselect2 ein

    for(p=7;p>=0;p--)           //Acht Durchgaenge für acht "pages" des Displays
    {
        lcd_page(p);            //Page setzen
        lcd_zeile(z);           //Zeilenposition setzen

        while(z<64)             //Endlosschleife
        {
            LCD_DATA=0x00;      //0x00 in den Datenbus P0 schreiben
            LCD_DI = 1;         //Display-RAM im "Empfang-Modus" (siehe S.10)
            lcd_send_data();    //Datenbus P0 wird in den RAM des Displays geschrieben
            LCD_DI = 0;         //Display-RAM im "Einstellungs-Modus"
            z++;                //Zaehlvariable inkrementieren
        }
        z=0;                    //Zaehlvariable zuruecksetzen
    }
}

//Flankenerkennung
int flankenerkennung(int x)
{
  static char old_level,fl;       //Hilfsvariablen deklariert

  if(old_level&&!fl)old_level=0;  //Damit Subroutine nicht immer
                                  //1 zurueck gibt
  if((x==0)&&(fl==0))             //Ist Taste gedrueckt?
  {                               //und flankenbit auf 0?
    old_level=fl;                 //alte Flankenstellung speichern
    fl=1;                         //pos. Flanke erkannt
  }

  if((x==1)&&(fl==1))             //war Flanke 1? und Taste nicht mehr gedrückt
  {
    old_level=fl;                 //alte Flankenstellung speichern
    fl=0;                         //neg. Flanke erkannt
  }
  return(old_level);              //old_level zurückgeben
}                                 //Wenn Taste gedrueckt und wieder losgelassen wurde
                                  //wird old_level auf 1
```

```
//Taster für BKL auslesen
void taster_check()
{
    static char y;                  //Hilfsvariable deklariert

    if(flankenerkennung(S1))        //Wurde Taste gedrueckt und wieder losgelassen?
    {
        if(y==1)                    //Ist Hilfvariable auf 1?
        {
            BKL=1;                  //Backlight off
            y=0;                    //Hilfsvariable nullen
        }
        else if(y==0)               //Ist Hilfsvariable 0?
        {
            BKL=0;                  //Backlight on
            y=1;                    //Hilfvariable auf 1
        }
    }
}

void main()                         //Hauptprogramm
{
    int i=0;                        //Zaehlvariable
    init();                         //Initialisierung
    lcd init();                     //LCD initalisieren
    lcd_clear();                    //LCD loeschen

    lcd_start();                    //Start Bildschirm zeichnen

    delay_ms(1000);                 //1s warten

    lcd_clear();                    //LCD loeschen

    lcd page(3);                    //Page setzen
    lcd zeile(0);                   //Zeilenposition setzen
    LCD CS2=1;                      //rechte Displayhaelfte abschalten
    LCD text("SPANNUNG");           //Textausgabe
    space();
    LCD_text("U");                  //Textausgabe
    space();
    delay_ms(1000);                 //1s warten
    delay ms(1000);                 //1s warten

    while(1)
    {

        do{
           adc value 3+=adc value;                  //Filter der Anzeige
           delay ms(10);                            //5ms Verzoegerung
           taster check();
           i++;                                     //Zaehlvariable inkrementieren
          }while(i<159);                            //128 Messungen
        LED1=~LED1;
        adc_value_3=(adc_value_3/160);              //Durchschnitt der 128 Messungen be
rechnen
        LCD print adc(adc value 3, adc outofrange); //ADC-Wert anzeigen
        adc value 3=0;                              //gespeicherter Wert loeschen
        i=0;                                        //Zaehlvariable loeschen
    }
}
```