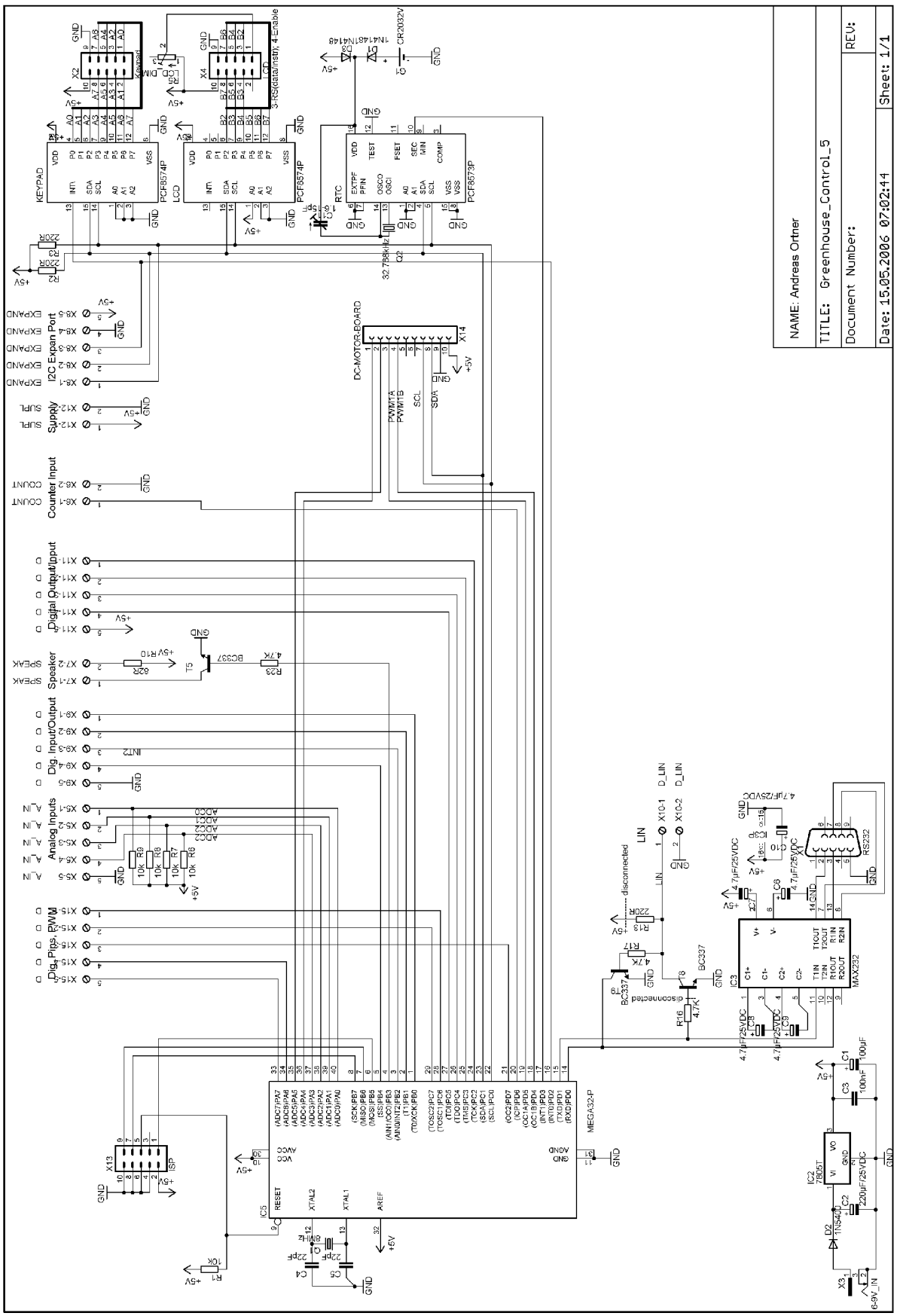
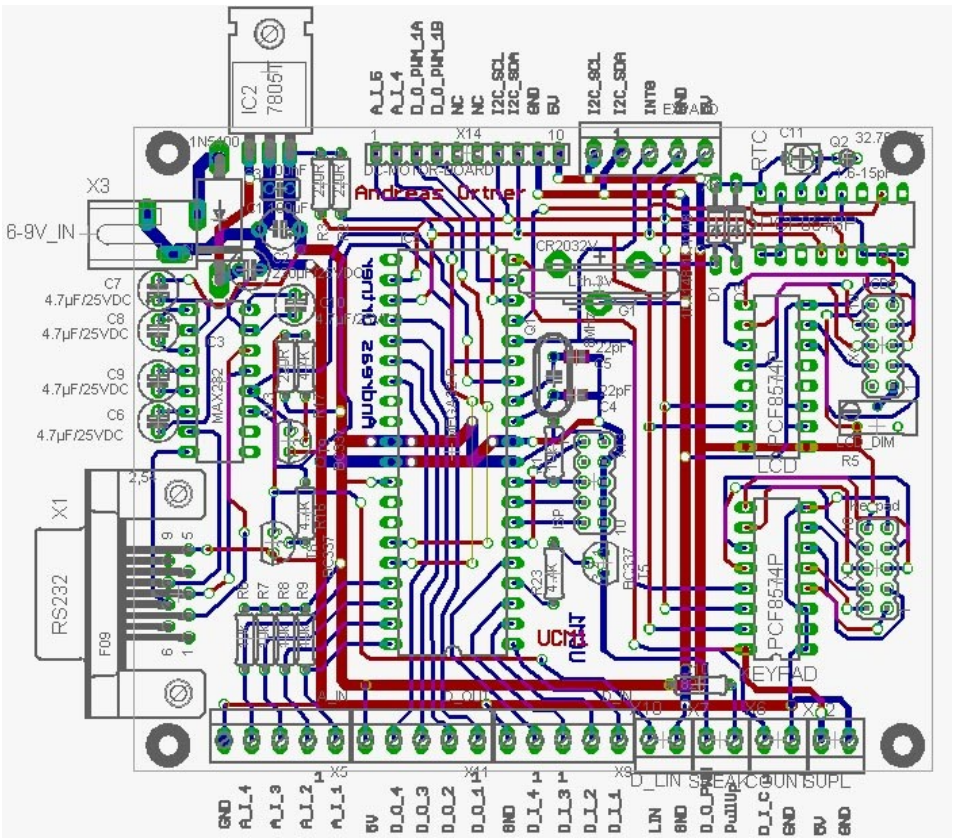
Hardware

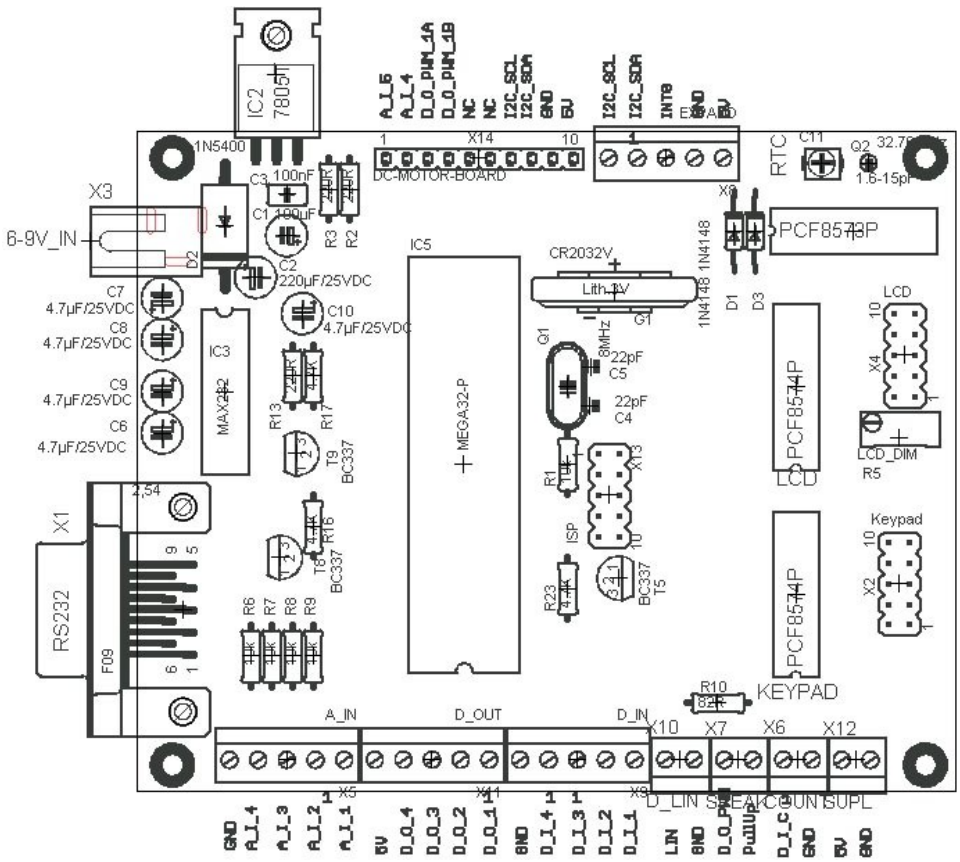
Steuergerät

Schaltplan BasicCard

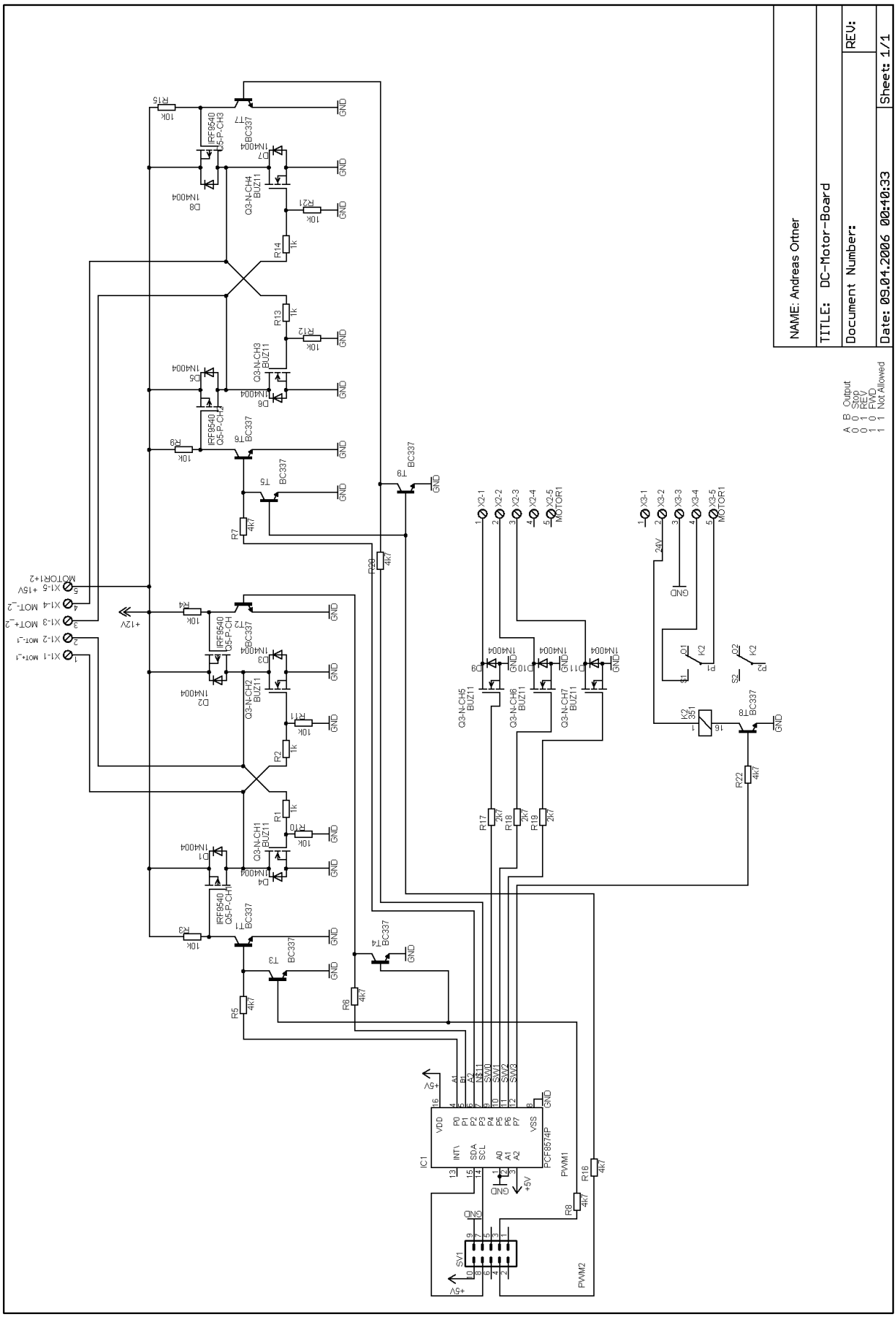


Layout BasicCard





Schaltplan PowerCard



Layout PowerCard

Komponenten BasicCard

Fenster-/Türantrieb

Motor, Typ Türantrieb ZA rechts, Nr. 0201.449

F 100 N
Länge 646 mm
Hersteller Motor Dunkermotoren UPM0,7A
, Getriebe Unkermotoren PLG30 M=180 Ncm, i=92,12
Spannung 15 VDC
Strom unbelastet 0,5 A
Strom belastet 1,1 A

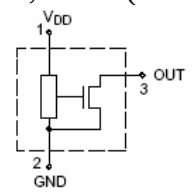
Sensor

90 Impulse für 8 cm Weg.
HAL502A von Micronas
Pullup am Ausgang ist notwendig.

Anschlussplan

1&6 Motor Minus, schwarz (br/bl)
2&7 Motor Plus, rot (br/rt)
3 GND, blau (br/gb)
8 Signal A, gelb (br/gn)
5 +5V…+24V, braun (br/ws)

Lead	MG
green	B
brown	+5V...+24V
blue	0V
yellow	A



Tauchpumpen (mit Rückschlagventil)

Hersteller Reich
Spannung 12 VDC
Strom 2A
Volumenleistung 12 l/min
Druck 0,5 bar

Magnetventil

Elektromagnet- Steuerventil (24 V AC)

Hersteller: Hunter

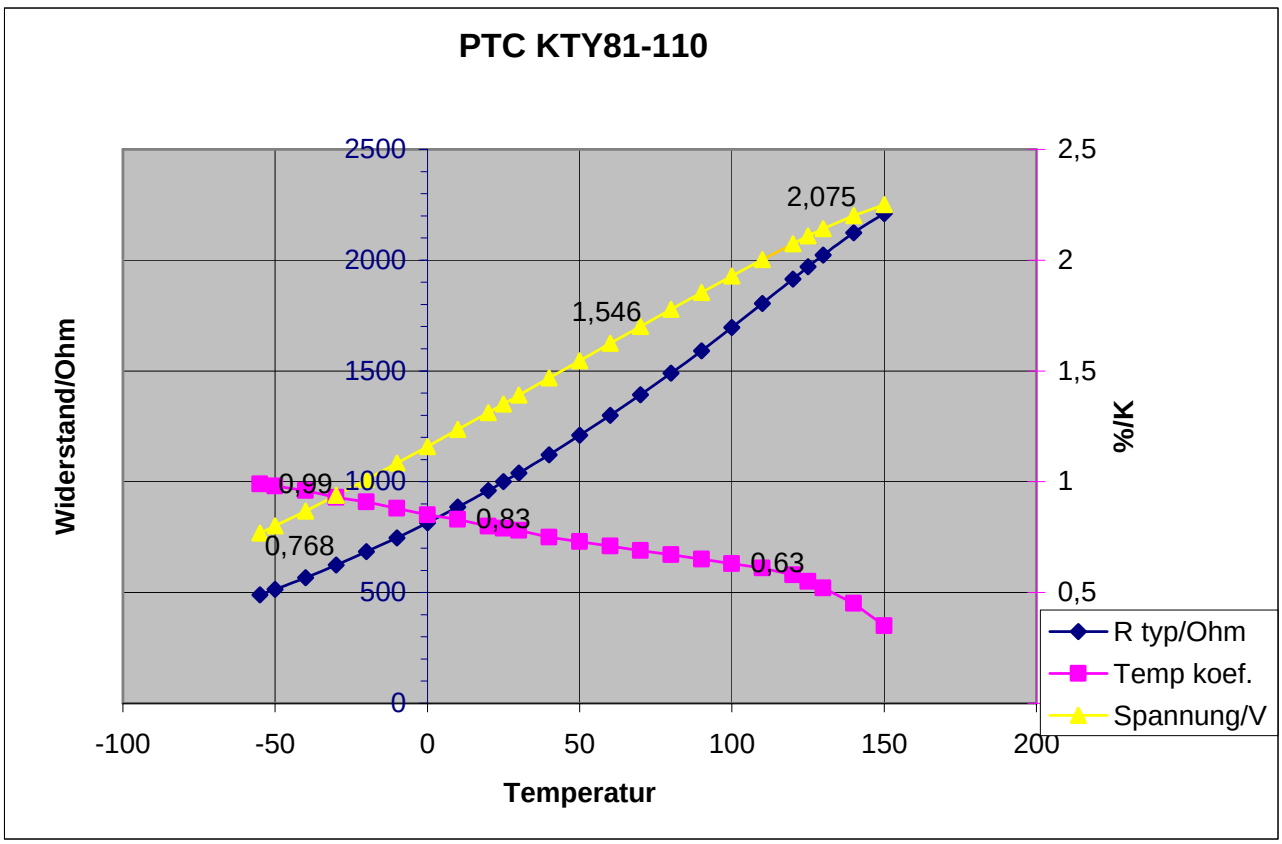
Artikel: SRV 101 G-B
Mit Durchflußregulierung, zur Reduzierung von Druck und Durchfluss, z.B. für Tropferleitungen und Microbewässerung.

Maximaler Betriebsdruck: 10 bar/ AS: 1" IG, gerade Anschlüsse, im drucklosen Zustand offen, unter Druck stromlos geschlossen.

Robuste Konstruktion aus hochwertigem PVC.
Für starke Beanspruchung geeignet!

Temperatursensoren

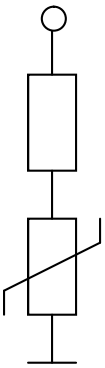
PTC-Element KTY 81-110
Manufacturer: Philips
Ambiet temperature: -55 to 150°C
Durchschnittliche Spannungsänderung7,22mV pro Kelvin



AMB. Temperature (C°)	TEMP. Coeff. (%/K)	KTY81-110 R/Ohm	Meas. Voltage /V **Rv=2700Ohm**	dU/dT(im mV/K)	Registerwert im ADC	Registerwert High im ADC	Registerwert Low im ADC
-55	0,99	490	0,768	6,582	310	1	54
-50	0,98	515	0,801	6,684	323	1	67
-40	0,96	567	0,868	7,086	350	1	94
-30	0,93	624	0,939	7,201	378	1	122
-20	0,91	684	1,011	7,291	407	1	151
-10	0,88	747	1,084	7,577	437	1	181
0	0,85	815	1,159	7,604	467	1	211
10	0,83	886	1,235	7,712	498	1	242
20	0,8	961	1,312	7,774	529	2	17
25	0,79	1000	1,351	7,805	545	2	33
30	0,78	1040	1,390	7,744	561	2	49
40	0,75	1122	1,468	7,861	592	2	80
50	0,73	1209	1,546	7,772	623	2	111
60	0,71	1299	1,624	7,672	655	2	143
70	0,69	1392	1,701	7,716	686	2	174
80	0,67	1490	1,778	7,584	717	2	205
90	0,65	1591	1,854	7,515	747	2	235
100	0,63	1696	1,929	7,430	778	3	10
110	0,61	1805	2,003	7,143	808	3	40
120	0,58	1915	2,075	6,890	836	3	68
125	0,55	1970	2,109	6,488	850	3	82
130	0,52	2023	2,142	5,985	863	3	95
140	0,45	2124	2,201	4,958	888	3	120
150	0,35	2211	2,251	15,007	908	3	140

Messung mit 10-bitADC

Auflösung:	5V/(2^10)Steps=5V/1024Steps=4,883 mV	
Delta	77mV/10°C=7,7mV/1°C	
Genauigkeit	ca. 1°C	
Temperatur	0°C	100°C
ADC-Wert	237	395



Rv=2700Ohm

Rptc

Auflösung:	5V/(2^10)Steps=5V/1024Steps=4,883 mV	
Delta	77mV/10°C=7,7mV/1°C	
Genauigkeit	ca. 1°C	
Temperatur	0°C	100°C
ADC-Wert	467	778

```
Steigung=(ADCValue50°C  623-  ADCValue0°C  467)/  (50°C-0°C)=  Value  156/50°C
Steigung = 3,12 (bzw. 0,32)
adc value at 0°C=237 T=(adc_value-467)*200/625 (0,32)
```

Pro 1°C hat man eine Änderung Wert von ADC-Register von 1,58

```
printf(" T1\r\n%d", (adc_value-237)*100/158);
//adc value at 0°C=237 T=(adc_value-237)*100/158 (0,63)
Steigung=(ADCValue50°C  317-  ADCValue0°C  237)/  (50°C-0°C)=  Value  80/50°C
Steigung = 1,6 (bzw. 0,625)
```

Reed-Kontakt

SAS 4AR/WS
Reedkontakt-Schalter
Umschaltkontakt, als Arbeits- und Ruhe-kontakt verwendbar.
Farbe: weiß
Maße: 65x 15,7x 13,6mm
Schaltkontakt:
30V / 0,3A
Preis: 3,25€

Angeschlossen: NO (Schließer)
Tür offen: geschlossen (Bit: Low)
Tür geschlossen: offen (Bit: High)

Schwimmschalter

Hersteller:	Condor
Typ:	PSN-X
~10A	250V
Niveau erreicht (voll):	Kontakt geschlossen (Low)
Niveau nicht erreicht (leer):	Kontakt geöffnet (High)

Lautsprecher

Using Timer 0
PWM frequency=8Mhz/2 *2^8=15.686 Hz

RC Tiefpass fg=7kHz
R=Xc
R=1/(2*Pi*fg*C)
C=1/(2*Pi*fg*R) = 1/(2*3,14*7kHz*100Ohm)=227nF

Gewächshaus.c

```
/*******************************************
Projekt : Gewaechshaussteuerung
Version : 1
Date    : 15.05.2006
Autor   : Andreas Ortner / a.ortner@freenet.de / www.a-ortner.de

Chip type           : ATMega32
Clock frequency     : 8 MHz
Board                       : GreenhouseControl AO6
*******************************************/
/*
Hardware Configuration

µC_PinD     µCPort      µCPinN      SigNam      Card  BoarPin     SW
Timer0      8-Bit Speak SysTim      BasicC
0C0         PB3   4     Speaker     BasicC      X7-1
T0          PB0   1     Windmet     BasicC      X9-1

Timer1      16-Bit                  BasicC
OC1A        PD5   19    PWM_M1      BasicC      X14-3
OC1B        PD4   18    PWM_M2      BasicC      X14-4
T1          PB1   2           BasicC      X9-2
ICP         PD6   20          BasicC      X6-1

Timer2      8-Bit             BasicC
0C2         PD7   21          BasicC      X15-3

INT0        PD2   16    IntKey      BasicC
INT1        PD3   17    INT_RTC     BasicC
INT2        PB2   3     ImpDoo      BasicC      X9-3

ADC0        PA0   40    PTCWat      BasicC      X5-1
ADC1        PA1   39    PTCAiri     BasicC      X5-2
ADC2        PA2   38    PTCAire     BasicC      X5-3
ADC3        PA3   37          BasicC
ADC4        PA4   36          BasicC      X14-1
ADC5        PA5   35          BasicC      X14-2
ADC6        PA6   34    SwNivea     BasicC      X15-4
ADC7        PA7   33          BasicC
AIN0        PB2   3           BasicC
AIN1        PB3   4           BasicC

SCK         PB7   8           BasicC
MISO        PB6   7           BasicC
SS          PB4   5     SwDoor      BasicC      X9-4
MOSI        PB5   6           BasicC
TXD         PD1   15          BasicC
RXD         PD0   14          BasicC
SDA         PC1   23          BasicC
SCL         PC0   22          BasicC

TDI         PC5   27          BasicC
TDO         PC4   26          BasicC
TMS         PC3   25          BasicC
TCK         PC2   24          BasicC
TOSC1       PC6   28    ReedWin     BasicC      X15-1
TOSC2       PC7   29    ReedDoo     BasicC      X15-2

*/
//#define DEBUG
```

```
typedef unsigned char  u08;

#define UCR UCSRB
//#define UCSRB UCR
//#define UCSRA USR
#define GIMSK GICR
#define uart_print() (&s)

#define MOT_1_RIGHT 1
#define MOT_1_LEFT 2
#define MOT_1_OFF 3
#define MOT_2_RIGHT 4
#define MOT_2_LEFT 5
#define MOT_2_OFF 6
#define SW_FET_1_ON 7
#define SW_FET_1_OFF 8
#define SW_FET_2_ON 9
#define SW_FET_2_OFF 10
#define SW_FET_3_ON 11
#define SW_FET_3_OFF 12
#define SW_REL_1_ON 13
#define SW_REL_1_OFF 14
#define ALL_OFF 15
#define SW_FET_1_TOGG 16
#define SW_FET_2_TOGG 17
#define SW_FET_3_TOGG 18
#define MOT_RIGHT 1
#define MOT_LEFT 2
#define MOT_OFF 0

#define MOT_DIST_IMPULSES_DOOR 1050
#define OPENED 1
#define CLOSED 2
#define OPENING 3
#define CLOSING 4
#define MIDDLE 5
#define STOP_REQUEST 6

#define ADC_AIR_INTERN 0
#define ADC_AIR_EXTERN 2
#define ADC_WATER 1

#define AUTOM 0
#define MANUALLY 1
#define ONLY_VENTILATION 2
#define ONLY_VENT_CHARGING 3

#define TRUE 1
#define FALSE 0

char s[20]=" ",s2[20]=" ";char s_empty[17]="             ";
char *p_s=s;
char Rx_buffer[30];char Rx_bit_number=0;char ascii_v[5]="34.3";
unsigned char KeyP_Buffer[7];u08 cur_menuPoint=0x00;u08 KeyP_in; //0-to
menu, 1-to function
struct sct_time {unsigned int year; u08 month; u08 day; u08 weekday; u08
hour; u08 minute; u08 second; u08 msecond500; u08 msecond250; u08
msecond;};
struct sct_time time ={2006,0,0,0,0,0,0,0,0,0};
struct sct_time time_
 ={0,0,0,0,0,0,0,0,0,0};
struct sct_weektime {unsigned int weekMinute_count; u08 minute; u08 hour;
```

```
u08 weekday;unsigned int EEPROMNumber; u08 channel; u08 on_off;};
struct sct_weektime weektime={0,0,0,0,0,0,0};
struct {
        unsigned int valve1  :1;
                 unsigned int valve2  :1;
                 unsigned int valve3  :1;
                 unsigned int valve4  :1;
                 unsigned int valve5  :1;
                 unsigned int valve6  :1;
                 unsigned int valve7  :1;
                 unsigned int valve8  :1;
                 }switchBits;

struct sct_temperature{unsigned char range; unsigned char
temperature_value;};
struct sct_temperature temperature={0,0};
unsigned int INT0_count, test=77,iX;

u08 test2=0,test3=0;

u08 sI8_second_event;
u08 sI8_1minute_event;
u08 sI8_5minute_event;
u08 sI8_1hour_event;
u08 sI8_1day_event;

char ch_x;
float fX;
unsigned int x;
u08 KP_Button=0;

u08 pCard_status=0;
u08 irrig_time_duration=1;
u08 begin_irrigation_time;
u08 water_temperature;
u08 req_water_temperature;
u08 today_irrigated=FALSE;

u08 max_wind_intensity;
u08 cur_wind_intensity;

u08 operation_on;   //bool
u08 begin_oper_time;
u08 end_oper_time;
u08 cur_int_air_temp=15;
u08 aver_temper_counter=0;
unsigned int accumulated_T_value=0;

u08 hyst_low_vent_temp=25;
u08 hyst_high_vent_temp=29;

u08 window_status;            //bool
u08 door_status;        //bool
u08 last_vent_time;
unsigned int iImpCount_Door;
unsigned int iImpCount_Window;


#include <macros.h>
#include <STDIO.h>
#include <iom32v.h>
#include <STDLIB.h>
#include <eeprom.h>
```

```
#include "STRING.H"
#include "uart.c"

#include "LCD_i2c.h"

#include "miscelaneous.h"



#include "miscelaneous.c"
#include "i2cmaster.c"
#include "LCD_i2c.c"
#include "keypad_i2c.c"

#pragma interrupt_handler INT0_ext_interrupt:2
#pragma interrupt_handler INT1_ext_interrupt:3
#pragma interrupt_handler INT2_ext_interrupt:4
//#pragma interrupt_handler timer0_ovf_isr:8 //at AT90S8515
#pragma interrupt_handler timer0_ovf_isr:12
#pragma interrupt_handler timer0_ovf_comp:11
//#pragma interrupt_handler timer1_ovf_isr:10
#pragma interrupt_handler timer2_ovf_isr:6
//#pragma interrupt_handler UART_RX_interrupt:10 //at AT90S8515
#pragma interrupt_handler UART_RX_interrupt:14
#pragma interrupt_handler twi_isr:20

void timer0_init(void);
void timer1_init(void);
void timer2_init(void);
void port_init(void);
void message_check(void);
void init_ext_interrupt(void);
void adc_init(unsigned char ADC_MUX);
unsigned int adc_conversation(unsigned char channel);
u08 get_temperature(u08 channel);
u08 get_weekday(void);
unsigned int change_Bytes_in_Int(unsigned int int_to_change);
unsigned int time_to_minuteValue(void);
void minuteValue_to_time(unsigned int minValue);
void EEPROMTime_to_weektime(unsigned int number);
unsigned int weektime_to_EEPROMTime(void);
void get_next_swValue(unsigned int curValue,unsigned int startValue);
void executeTimer(void);
unsigned int getCurMinuteWeekValue(void);
void powerCard(u08 ctrl_code);
void doorOpen(void);
void doorClose(void);
void windowOpen(void);
void windowClose(void);
void doorOperation(void);
void windowOperation(void);
void ventilation(void);
void warmWaterCharge(void);
void irrigation(void);


/******************** main************************/

void main( void )
 {
  //int i;
  //int zahl=0x1234;
```

```
  //char str1[20]= "Andreas***";

  InitUART( 51 ); // 51 set the baudrate to 9,6 kbps using a 8MHz crystal
(8 for 57,6kBaud)
  //InitUART( 25 ); // set the baudrate to 19,2 kbps using a 8MHz crystal
(8 for 57,6kBaud)
  UART_TRANSMIT_ON();
  CLI(); //disable all interrupts

  MCUCR = 0x00;
  GIMSK = 0x00;
  TIMSK = (1<<TOIE2); //|(1<<TOIE0)
  port_init();
  //timer2_init(); //Lautsprecher
  timer1_init();
  timer0_init();
  init_ext_interrupt();
  i2c_init();
  lcd_i2c_init();
  init_keypad();
  SEI(); //re-enable interrupts
  powerCard(ALL_OFF);

  //EEPROM_WRITE(0x01, time.month);

  i2c_PCF8573_TimeRD (i2cRTC);//update time from PCF8573 to time.x
  //time.weekday = get_weekday();//calculate weekday to time.weekday
  //get_next_swValue(getCurMinuteWeekValue() ,0);

  adc_init(0);
  //get_temperature(0);

 intro();
 printf("\r\nWILLKOMMEN");
 #ifndef DEBUG
 //Init door and window
  if(CHECKBIT_FOR_ZERO(PINC,6)) //if window is open then close
   {iImpCount_Window=MOT_DIST_IMPULSES_DOOR;
            windowClose();
            iImpCount_Window=0;
   }
  window_status=CLOSED;
  if(CHECKBIT_FOR_ZERO(PINC,7)) //if door is open then close
   {
    iImpCount_Door=MOT_DIST_IMPULSES_DOOR;
            doorClose();
            doorClose();
            iImpCount_Door=0;
   }
#endif
 door_status=CLOSED;
  //lcd_i2c_update();
  menuGuide();
 }
////////////////////////////////////////////////////////////////////////
///
void port_init(void)
{
 PORTB = 0xFF; DDRB  = 0xFF;
 PORTD = 0x00; DDRD  = 0xFF;

 CLEARBIT(DDRC,6); //as input for Reed_Window contact
 SETBIT(PORTC,6); //Pull up
```

```
 CLEARBIT(DDRC,7); //as input for Reed_Door contact
 SETBIT(PORTC,7); //Pull up
 CLEARBIT(DDRB,2); //as input for Switch_Door contact
 SETBIT(PORTB,2); //Pull up
 CLEARBIT(DDRB,4); //as input for Niveau_Switch contact
 SETBIT(PORTB,4); //Pull up
 CLEARBIT(DDRA,6); //as input for Impuls_Window contact
 SETBIT(PORTA,6); //Pull up
 CLEARBIT(DDRA,7); //as input for Impuls_Door contact
 SETBIT(PORTA,7); //Pull up
}
////////////////////////////////////////////////////////////////////////
///
void UART_RX_interrupt( void )
{
            //printf("rec");
            //TransmitByte( UDR ); //CLEARBIT(UCSRA,RXC);
            if (UDR == '<'){Rx_bit_number=0; TransmitByte('<');}
            else {if (UDR == '>') {TransmitByte('>');
message_check();Rx_bit_number=0;}
                  else {Rx_buffer[Rx_bit_number++]=UDR;}}
}
////////////////////////////////////////////////////////////////////////
///
void adc_init(unsigned char ADC_MUX)
{
  ADCSR = 0x00; //disable adc
  //ADMUX = 0x00; //select adc input 0
  ADMUX=(ADMUX>>3);
  ADMUX=(ADMUX<<3);
  ADMUX|=ADC_MUX;//clear three right bits and overwright them with ADC_MUX
  ACSR = (1<<ACD);//Analog comparator disable (for saving power
  //ADCSRA = ((1<<ADEN) | (1<<ADPS0));//enable ADC
  //ADCSRA = ((1<<ADEN) | (1<<ADPS0))| (1<<ADSC); //
  ADCSRA = ((1<<ADEN) | (1<<ADPS0))| (1<<ADATE) | (1<<ADSC);
}
////////////////////////////////////////////////////////////////////////
///
unsigned int adc_conversation(unsigned char channel)
{
  unsigned int adc_value=0;
    adc_init(channel);

  while (CHECKBIT_FOR_ZERO(ADCSRA,ADIF)){} //wait until conversated
(ADSC=0)
  CLEARBIT(ADCSRA,ADSC);//stop ADC

  adc_value =ADCL;
  adc_value |= ((int)ADCH<<8);
    return adc_value;
}
////////////////////////////////////////////////////////////////////////
///
void init_ext_interrupt( )
            {
            MCUCR=0;
            //*******************PORTD.2 as
INT0***************************
            MCUCR |= ((1<<ISC01)); /* The falling edge of INT0 */
            GICR |= (1<<INT0); //enable INT0 interrupt
            CLEARBIT(DDRD,2); //pin as input pull-up
            SETBIT(PORTD,2);
```

```
            //*******************PORTD.3 as
INT1***************************
            MCUCR |= ((1<<ISC11)); /* The falling edge of INT1 */
            GICR |= (1<<INT1); //enable INT1 interrupt
            CLEARBIT(DDRD,3); //pin as input pull-up
            SETBIT(PORTD,3);

            //*******************PORTB.2 as
INT2***************************
            MCUCSR &= ~(1<<ISC2); /* The falling edge of INT2 */
            GICR |= (1<<INT2); //enable INT1 interrupt
            CLEARBIT(DDRB,2); //pin as input pull-up
            SETBIT(PORTB,2);
            }
////////////////////////////////////////////////////////////////////////////
///
void INT0_ext_interrupt(void)
{
  KeyP_evaluate();
}
////////////////////////////////////////////////////////////////////////////
///
void INT1_ext_interrupt(void)//each second one interrupt from RTC
{

  time.second++;
  sI8_second_event=0xFF;
    cur_int_air_temp=get_temperature(ADC_AIR_INTERN);

////#########################################################################
  if(time.second == 60)
                {time.second = 0;
                 time.minute++;
                   adc_init(0);
                   get_temperature(0);
                   //executeTimer();
                   ////////////////////////////////////////////
                   lcd_i2c_print_time();

                 if(time.minute == 60)
                  {time.minute = 0;
                   time.hour++;
                     if(time.hour == 24)
                     {time.hour = 0;
                          time.weekday++;
                          ////////////////////////////////////////
                          today_irrigated=FALSE;

                          if(time.weekday==8){time.weekday=1;}
                          time.day++;
                           if((time.day==29)&(time.month==2))//if February
and 29, then...
                            {time.day=1;
                             time.month++;
                            }
                            //odd month has 31 days, even has 30 days

if(((time.day==31)&((time.month%2)==0))|(time.month==32)) //if more then 30
and even (April, Juny, August) or day==32
                            {time.day=1;
                            time.month++;
                            }
                            if (time.month==13)
```

```
                        {time.month=1;
                         time.year++;
                        } //month //day
                     } //hour
                    } //minute
          } //second
}
////////////////////////////////////////////////////////////////////////
///
void INT2_ext_interrupt(void)
{
  delms(30); //if problems, then increase to 20
  if(CHECKBIT_FOR_ZERO(PINB,PB2)) //only if button still pressed
   {doorOperation();
   }
}
////////////////////////////////////////////////////////////////////////
///
void timer0_init(void)
{
 TCCR0 = 0x00; //stop
 //TCNT0 = 0xC7; //set count
 OCR0  = 0x80;  //set compare

 //start timer  fast PWM, set OC0 on Compare mattch,  no prescale  //
 TCCR0 = (1<<WGM01)|(1<<WGM00)|(1<<COM01)|(1<<COM00)|(1<<CS00);
 //DDRB=DDRD|(1<<3);
 //DDRB=0xFF;
}
////////////////////////////////////////////////////////////////////////
///
void timer0_ovf_isr(void)
{
  //OCR0++;
}
////////////////////////////////////////////////////////////////////////
///
void timer0_ovf_comp(void)
{

}
////////////////////////////////////////////////////////////////////////
///
void timer1_init(void)
{
  TCCR1B = 0x00; //stop
  TCNT1H = 0xFB; //setup
  TCNT1L = 0xFC;
  OCR1AH = 0x01;
  OCR1AL = 0x5F; //OCR1AL = 0x8F;
  OCR1BH = 0x04;
  OCR1BL = 0x04;
  TCCR1A = 0x81; // 8= PWM normal, 1= 8bit PWM
  TCCR1B = 0x0B; //start Timer clock  divider=256
  //TCCR1B = 0x07; //start Timer from T0
}
////////////////////////////////////////////////////////////////////////
///
void timer2_init(void)
{
 TCCR2 = 0x00; //stop
 TCNT2 = 0xC7; //set count
 //OCR0  = 0x39;  //set compare
```

```
 TCCR2 = 0x05; //start timer      //M32
}
////////////////////////////////////////////////////////////////////////////
///
void timer2_ovf_isr(void)
{

}
////////////////////////////////////////////////////////////////////////////
///
void twi_isr(void)
{
 //twi event
}
////////////////////////////////////////////////////////////////////////////
///
u08 get_temperature(unsigned char channel)  //Linear + 10.0 mV/°C scale
factor
{
  //float T_float=24.89;
  unsigned int ADC_value_temp;
  u08 I8_temperature;
  ADC_value_temp=adc_conversation(channel);
  //printf("T:__%d", ADC_value_temp);
  printf("\r\nADCValue%d", ADC_value_temp);
  //printf(" T1\r\n%d", (adc_value-237)*100/158); //adc value at 0°C=237
T=(adc_value-237)*100/158
  //printf(" T1\r\n%d", (ADC_value_temp-237)*50/80); //adc value at 0°C=237
T=(adc_value-237)*100/158

  ADC_value_temp=((ADC_value_temp-237)*200/320)-4;
  printf(" T_berechn\r\n%d", ADC_value_temp);
  //ADC_value_temp=((ADC_value_temp-237)*50/80)-4;
  //printf(" T1\r\n%d", I8_temperature); //adc value at 0°C=237
T=(adc_value-237)*100/158
  I8_temperature=ADC_value_temp;
  printf(" T_berechn\r\n%d", I8_temperature);
  printf("\r\n");
  return I8_temperature;
}
////////////////////////////////////////////////////////////////////////////
///
u08 get_weekday()
{
/*
J sie die Zahl des Jahrhunderts,
K die Jahrszahl innerhalb desselben,
m die Zahl des Monats,
q die Zahl des Monatstags,
h die Zahl des Wochentags;
Bruchreste bleiben weg; Januar und Februar werden als 13. u. 14. Monat des
vorhergehenden Jahres betrachtet--:
dann ist h gleich dem Reste, welcher bleibt
B) im gregorianischen Kalender, wenn die Summe
q + ((m+1)26)/10 + K + K/4 + J/4 - 2J
durch 7 dividiert wird.


B) 1712, 24 Jan. Geburstag Friedr. II.
q = 24, m = 13, K = 11, J = 17
24 + ((13 + 1)26)/10 + 11 + 11/4 + 17/4 - 2*17 = 24 + 36 + 11 + 2 + 4 - 34
= 43 = 6*7 + 1
h = 1; also war Freidr. II geboren am 1. Wochentag oder Sonntag.
```

```
*/
unsigned int q, m, K, J, h, w;

i2c_PCF8573_TimeRD (i2cRTC); // update time from PCF8573
q=time.day;
if (time.month<3)m=time.month+12;
   else m=time.month;
K=time.year-2000;
J=time.year/100;
h=q+((m+1)*26)/10 + K + K/4 + J/4 - 2*J;
h=h%7;
if (h==1) {h=7;}
if (h==0) {h=6;}
else h-=1;
return h;
}
////////////////////////////////////////////////////////////////////////////
///
unsigned int change_Bytes_in_Int(unsigned int int_to_change)
{unsigned int int_to_change2;

 int_to_change2=int_to_change;
 int_to_change <<=8;  int_to_change2 >>=8;
 int_to_change2+= int_to_change;
 return int_to_change2;

}
////////////////////////////////////////////////////////////////////////////
///
void executeTimer(void)
{unsigned int curValue;
 if(weektime.weekMinute_count==getCurMinuteWeekValue())
 {switch(weektime.channel)
            {case 1: switchBits.valve1=weektime.on_off;

TOGGLEBIT(PORTC,2);lcd_WR_string(LCD_L1,"SW1");break;
             case 2: switchBits.valve1=weektime.on_off;

TOGGLEBIT(PORTC,3);lcd_WR_string(LCD_L3,"SW2");break;
             case 3: switchBits.valve1=weektime.on_off;
                         TOGGLEBIT(PORTC,4);

            }
            delms(2000);
 }

 get_next_swValue(getCurMinuteWeekValue() ,0);
}

void powerCard(u08 ctrl_code)
{//0&1:Motor 1, 2&3:Motor 2, 4:SW_FET_1, 5:SW_FET_2, 6:SW_FET_3, 7:SW_REL_1
switch(ctrl_code) //
             {case MOT_1_RIGHT:
                    pCard_status &= 0b11111100; //delete bits for motor 1
                    pCard_status |= MOT_RIGHT;  //set bits for motor 1
                    break;
              case MOT_1_LEFT:
                    pCard_status &= 0b11111100; //delete bits for motor 1
                    pCard_status |= MOT_LEFT;
                    break;
              case MOT_1_OFF:
                    pCard_status &= 0b11111100; //delete bits for motor 1
                    break;
```

```
                case MOT_2_RIGHT:
                      pCard_status &= 0b11110011; //delete bits for motor 1
                      pCard_status |= (MOT_RIGHT<<2);  //move 2 bit to left
and set bits for motor 1
                      break;
                case MOT_2_LEFT:
                      pCard_status &= 0b11110011; //delete bits for motor 1
                      pCard_status |= (MOT_LEFT<<2);
                      break;
                case MOT_2_OFF:
                      pCard_status &= 0b11110011; //delete bits for motor 1
                      break;
                case SW_FET_1_ON:
                      pCard_status |= 0b00010000; //set bit
                      break;
                case SW_FET_1_OFF:
                      pCard_status &= 0b11101111; //set bit
                      break;
                case SW_FET_2_ON:
                      pCard_status |= 0b00100000; //set bit
                      break;
                case SW_FET_2_OFF:
                      pCard_status &= 0b11011111; //set bit
                      break;
                case SW_FET_3_ON:
                      pCard_status |= 0b01000000; //set bit
                      break;
                case SW_FET_3_OFF:
                      pCard_status &= 0b10111111; //set bit
                      break;
                case SW_REL_1_ON:
                      pCard_status |= 0b10000000; //set bit
                      break;
                case SW_REL_1_OFF:
                      pCard_status &= 0b01111111; //set bit
                      break;
                case SW_FET_1_TOGG:
                      TOGGLEBIT(pCard_status,4); //toggle bit
                      break;
                case SW_FET_2_TOGG:
                      TOGGLEBIT(pCard_status,5); //toggle bit
                      break;
                case SW_FET_3_TOGG:
                      TOGGLEBIT(pCard_status,6); //toggle bit
                      break;
                default: pCard_status=0b00000000;

              }
i2c_WR(i2cPowerCard,pCard_status);
}
////////////////////////////////////////////////////////////////////////////
///
void doorOpen(void)
{
  u08 imp_state=0;
  powerCard(MOT_2_RIGHT);
  door_status=OPENING;
  //lcd_WR_string(LCD_Line_1+5,"Dop");
  while(iImpCount_Door<MOT_DIST_IMPULSES_DOOR)
  {switch(imp_state)
   {   case 0:
               {if(CHECKBIT(PINA,7))
                {imp_state=1;
```

```
                   }
                   break;
                }
                case 1:
                {if(CHECKBIT_FOR_ZERO(PINA,7))
                 {imp_state=0;iImpCount_Door++;
                   }
                   break;
                }
    }
    delus(5);
    if(door_status==STOP_REQUEST) {door_status=MIDDLE;break;}

   }
 powerCard(MOT_2_OFF);
 //lcd_WR_string(LCD_Line_1,"Dopd");
 if(iImpCount_Door>=MOT_DIST_IMPULSES_DOOR) {door_status=OPENED;}

}
////////////////////////////////////////////////////////////////////////
///
void doorClose()
{
  u08 imp_state=0;
  powerCard(MOT_2_LEFT);
  door_status=CLOSING;
  //lcd_WR_string(LCD_Line_1+5,"Dcls");
  while(CHECKBIT_FOR_ZERO(PINC,7))
  {switch(imp_state)
   {   case 0:
                {if(CHECKBIT(PINA,7))
                 {imp_state=1;
                   }
                   break;
                }
                case 1:
                {if(CHECKBIT_FOR_ZERO(PINA,7))
                 {imp_state=0;iImpCount_Door--;
                   }
                   break;
                }
    }
    delus(5);
    if(CHECKBIT(PINC,7)) {door_status=CLOSED;iImpCount_Door=0;break;}   //
if reed contact is shorted, then interrupt closing
    if(door_status==STOP_REQUEST) {door_status=MIDDLE;break;}
  }
  powerCard(MOT_2_OFF);
}
////////////////////////////////////////////////////////////////////////
///
void windowOpen(void)
{
  u08 imp_state=0;
  powerCard(MOT_1_RIGHT);
  window_status=OPENING;
  while(iImpCount_Window<MOT_DIST_IMPULSES_DOOR)
  {switch(imp_state)
   {   case 0:
                {if(CHECKBIT(PINA,6))
                 {imp_state=1;
                   }
                   break;
```

```
                }
                case 1:
                {if(CHECKBIT_FOR_ZERO(PINA,6))
                 {imp_state=0;iImpCount_Window++;
                   }
                   break;
                }
    }
    delus(5);
    if(window_status==STOP_REQUEST) {window_status=MIDDLE;break;}
   }

   powerCard(MOT_1_OFF);
   if(iImpCount_Window>=MOT_DIST_IMPULSES_DOOR) {window_status=OPENED;}

}
////////////////////////////////////////////////////////////////////////////
///
void windowClose()
{
   u08 imp_state=0;
   powerCard(MOT_1_LEFT);
   window_status=CLOSING;
   while(CHECKBIT_FOR_ZERO(PINC,6))
   {switch(imp_state)
    {   case 0:
                {if(CHECKBIT(PINA,6))
                 {imp_state=1;
                   }
                   break;
                }
                case 1:
                {if(CHECKBIT_FOR_ZERO(PINA,6))
                 {imp_state=0;iImpCount_Window--;
                   }
                   break;
                }
    }
    delus(5);
    if(CHECKBIT(PINC,6)) {window_status=CLOSED;iImpCount_Window=0;break;}
// if reed contact is shorted, then interrupt closing
    if(window_status==STOP_REQUEST) {window_status=MIDDLE;break;}
   }
   powerCard(MOT_1_OFF);
}
////////////////////////////////////////////////////////////////////////////
///
void doorOperation()
{
 #ifndef DEBUG
 test3++;
 if((window_status!=CLOSING)&&(window_status!=OPENING))//execute only if
window is not operating
 {
  switch(door_status)
   {   case CLOSED:
                {doorOpen();
                   break;
                }
                case OPENED:
                {doorClose();
                   break;
                }
```

```
                case CLOSING:
                {//door_status=STOP_REQUEST;
                   break;
                }
                case OPENING:
                {//door_status=STOP_REQUEST;
                   break;
                }
                case MIDDLE:
                {//doorClose();
                   break;
                }
                case STOP_REQUEST:
                {
                   break;
                }
    }
   }
 #endif
}
////////////////////////////////////////////////////////////////////////////
///
void windowOperation()
{
 #ifndef DEBUG
 switch(window_status)
    {   case CLOSED:
                {windowOpen();
                   break;
                }
                case OPENED:
                {windowClose();
                   break;
                }
                case CLOSING:
                {//window_status=STOP_REQUEST;
                   break;
                }
                case OPENING:
                {//window_status=STOP_REQUEST;
                   break;
                }
                case MIDDLE:
                {//windowClose();
                   break;
                }
                case STOP_REQUEST:
                {
                   break;
                }
    }
 #endif
}
////////////////////////////////////////////////////////////////////////////
///
void ventilation()
{
   //cooling down, open door
   //door_status=CLOSED;

   //printf("\r\nGoing_to_ventilation_");
   //sprintf(s, "\r\n cur_int_air_temp_%d", cur_int_air_temp);
uart_printstr( &s[0]);
```

```
   //sprintf(s, "\r\n hyst_high_vent_temp_%d", hyst_high_vent_temp);
uart_printstr( &s[0]);
   //if (door_status==CLOSED){printf("\r\n door_status==CLOSED");}
   //if ((cur_int_air_temp>hyst_high_vent_temp)){printf("\r\n
cur_int_air_temp>hyst_high_vent_temp=TRUE");}
   lcd_WR_string(LCD_Line_3+10,"in_v");
   if ((cur_int_air_temp>hyst_high_vent_temp)&&(door_status==CLOSED))
    {//printf("\r\nTemp__%d", hyst_high_vent_temp);
             //printf("_____cooling down____\r\n");
             //if
((cur_int_air_temp>hyst_high_vent_temp)){printf("\r\nAIR_TRU__");}
             //if ((door_status==CLOSED)){printf("\r\nDOOR_TRU__");}
             lcd_WR_string(LCD_Line_1+9,"kueh");
             doorOperation();//open door
             if(window_status==CLOSED){windowOperation();} //open window
            }

   //heating up, close door
   //door_status=OPENED;
   if ((cur_int_air_temp<hyst_low_vent_temp)&&(door_status==OPENED))
    {lcd_WR_string(LCD_Line_1+9,"heiz");
             //printf("_____heating up____\r\n");
             doorOperation();//close door
             if(window_status==OPENED){windowOperation();}//close window
            }

}
////////////////////////////////////////////////////////////////////////
///
void warmWaterCharge()
{
 //fill up the cisterne
 if(CHECKBIT(PINB,4)) //later && today_irrigated==FALSE (fill only if today
not irrigared
  {lcd_WR_string(LCD_Line_1+9,"fuel");
   powerCard(SW_FET_1_ON);
   delms(1000);
   powerCard(SW_FET_1_OFF);
  }
 //
 if(CHECKBIT_FOR_ZERO(PINB,4))//if Niveau_Sw closed (water full)
 {lcd_WR_string(LCD_Line_1+9,"zirk");
   powerCard(SW_FET_2_ON);
   delms(1000);
   powerCard(SW_FET_2_OFF);
  }
}
////////////////////////////////////////////////////////////////////////
///
void irrigation()
{
 powerCard(SW_FET_3_ON);
 lcd_WR_string(LCD_Line_1+9,"bews");
 delms(((int)irrig_time_duration*1000));
 powerCard(SW_FET_3_OFF);
 today_irrigated=TRUE;
}
////////////////////////////////////////////////////////////////////////
///
void automatic()
{
 u08 I8_Temperature2;
```

```
 powerCard(ALL_OFF);
 sprintf(s, "GRH AO06");
 lcd_WR_string(LCD_L1,&s[0]);
 sprintf(s, "Ti");
 lcd_WR_string(LCD_L3,&s[0]);
 sprintf(s, "Ta");
 lcd_WR_string(LCD_L3+7,&s[0]);
 sprintf(s, "Auto");
 lcd_WR_string(LCD_L4,&s[0]);
 lcd_i2c_print_time();

 while(!(ESC==KeyP_Buffer[0]))
  {//i2c_PCF8573_TimeRD (i2cRTC);
   //lcd_i2c_print_time();
   //adc_init(0);



   if(CHECKBIT(sI8_second_event,0))//execute only once per second, use only
one bite in the byte
    {sprintf(s, "Ti:%d", cur_int_air_temp);lcd_WR_string(LCD_L3,&s[0]);

     ventilation();
     warmWaterCharge();
     //irrigation();

             CLEARBIT(sI8_second_event,0);//event executed, therefore clear
bit
    }




   switch(KP_Button) //switch(scan_buff=(KEYP_PIN>>4))
             {case '1':
               break;
              case '2':
               break;
              case '3':
               break;
              case '4':
               break;
              case '5':
               break;
             }
             KP_Button=0;
             delus(1000);
  }
  KeyP_Buffer[0]=0;
}
////////////////////////////////////////////////////////////////////////////
///
void manuell()
{

 sprintf(s, "GRH AO06");
 lcd_WR_string(LCD_L1,&s[0]);
 sprintf(s, "V1:0 Z2:0 B3:0");
 lcd_WR_string(LCD_L2,&s[0]);
 sprintf(s, "F4:0 T4:0");
 lcd_WR_string(LCD_L3,&s[0]);
 sprintf(s, "Hand");
```

```
 lcd_WR_string(LCD_L4,&s[0]);
 lcd_i2c_print_time();

 while(!(ESC==KeyP_Buffer[0]))
  {//i2c_PCF8573_TimeRD (i2cRTC);
   //lcd_i2c_print_time();

   switch(KP_Button) //switch(scan_buff=(KEYP_PIN>>4))
             {case '1':
               powerCard(SW_FET_1_TOGG);

if(CHECKBIT(pCard_status,4)){lcd_WR_string(LCD_Line_2+3,"1");}

if(CHECKBIT_FOR_ZERO(pCard_status,4)){lcd_WR_string(LCD_Line_2+3,"0");}

               break;
              case '2':
               powerCard(SW_FET_2_TOGG);

if(CHECKBIT(pCard_status,5)){lcd_WR_string(LCD_Line_2+8,"1");}

if(CHECKBIT_FOR_ZERO(pCard_status,5)){lcd_WR_string(LCD_Line_2+8,"0");}

               break;
              case '3':
               powerCard(SW_FET_3_TOGG);

if(CHECKBIT(pCard_status,6)){lcd_WR_string(LCD_Line_2+13,"1");}

if(CHECKBIT_FOR_ZERO(pCard_status,6)){lcd_WR_string(LCD_Line_2+13,"0");}

               break;
              case '4':
               #ifndef DEBUG
               sprintf(s, "w%d",
window_status);lcd_WR_string(LCD_L1+13,&s[0]);
               windowOperation();
               #endif
               sprintf(s, "%d",
window_status);lcd_WR_string(LCD_L3+3,&s[0]);
               break;
              case '5':
               #ifndef DEBUG
               sprintf(s, "d%d",
door_status);lcd_WR_string(LCD_L3+13,&s[0]);
               doorOperation();
               #endif
               sprintf(s, "%d", door_status);lcd_WR_string(LCD_L3+8,&s[0]);
               break;
             }
             KP_Button=0;
             delus(1000);
  }
  KeyP_Buffer[0]=0;

}
////////////////////////////////////////////////////////////////////////////
///
void set_T_hyst()
{

}
```