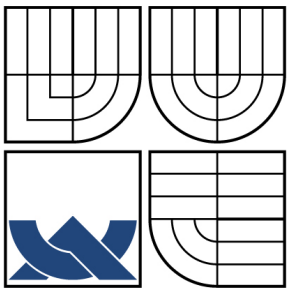
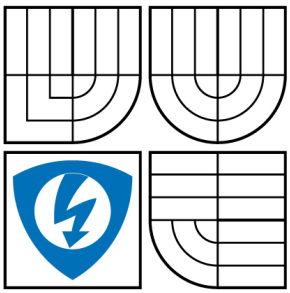


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV MIKROELEKTRONIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF MICROELECTRONICS

NÁVRH HARDWARU ŘÍDÍCÍHO MODULU PRO AUTOBUSOVÉ KLIMATIZACE

DESIGN OF CONTROL MODULE FOR BUS AIR-CONDITION

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE Bc. LUKÁŠ MELICHÁREK

AUTHOR

VEDOUCÍ PRÁCE ING. MATÚŠ RÁKOCI

SUPERVISOR

BRNO 2008

PŘÍLOHA

OBSAH

1 Program

1.1 Main

```
#include <REG164CR.H>
#include <MATH.H>
#include <can_ext.h>
#include "ds1307.h"
#include "cas.h"
#include "display.h"
#include "main.h"
#include "menu.h"
#include "eeprom.h"
#include "button.h"
#include "Can_167.h"

#define XTD_MES 1
#define STD_MES 0

#define TEST_IDENTIFIER 0x151
#define RX      0
#define TX      1

#define RUN  0
#define STOP 1

#define COOL  "COOL"
#define HEAT  "HEAT"
#define AUTO  "AUTO"
#define VENT  "VENT"

#define OFF  "OFF"
#define P50  "50%"
#define P100  "100%"

//konec hodiny
sTIME cas;
sDATE datum;

sfrbit  driver_led  _atbit(P8, 3);
sfrbit  passangers_led  _atbit(P8, 2);

unsigned int  i;

bit on;
unsigned char on_send;
bit set_value;
unsigned char  out_counter;
unsigned char data[8];
```

```
unsigned char *posli;

unsigned char ad_input1[8];
unsigned char ad_input2[8];
unsigned char input[8];

unsigned int reguest;
unsigned int wait_b;
bit reguest_bit;
bit read;
unsigned char reguest_reg[8];

_bitword tlac;

_bit tlac0 _atbit( tlac, 0 );
_bit tlac1 _atbit( tlac, 1 );
_bit tlac2 _atbit( tlac, 2 );
_bit tlac3 _atbit( tlac, 3 );

_bitword  tlac_norm;
_bit PO_P1H4 _atbit( tlac_norm, 0 );
_bit PO_P1H5 _atbit( tlac_norm, 1 );
_bit PO_P1H6 _atbit( tlac_norm, 2 );
_bit PO_P50 _atbit( tlac_norm, 3 );
_bit PO_P51 _atbit( tlac_norm, 4 );
_bit PO_P52 _atbit( tlac_norm, 5 );
_bit PO_P53 _atbit( tlac_norm, 6 );
_bit PO_P54 _atbit( tlac_norm, 7 );
_bit PO_P55 _atbit( tlac_norm, 8 );
_bit PO_P56 _atbit( tlac_norm, 9 );
_bit PO_P57 _atbit( tlac_norm, 10 );

unsigned int measure_driver_temperature;
unsigned int measure_passangers_temperature;
unsigned int measure_out_temperature;

unsigned char output_bts6208_61;
unsigned char output_bts6208_62;
unsigned char output_bts723;
signed char driver_temperature;
signed char passangers_temperature;
signed char out_temperature;

bit set_driver_zone;
bit set_passangers_zone;
unsigned char driver_zone;
unsigned char passangers_zone;
unsigned char fresh_air_passangers_zone;
unsigned char fresh_air_driver_zone;
```

```
unsigned char servise_menu;
bit   setup_menu;
unsigned char  setup_menu_help;

unsigned char mode;
unsigned char mode_state;
unsigned int i;

char *text = "Text ulozeny v RAM 1307";
char rxtext[24];

unsigned char P_ambient_switch_point; // 0°C to 15°C
unsigned char P_ambient_floor_heat;   // 0°C to 30°C
unsigned char P_REHEAT_mode;                    // 00 01 02 03
unsigned char P_DELTA_T;                        // 1°C to 15°C
bit  P_Floor_heating_mode;                  // 0 1
unsigned char P_Floor_heating_offset; // 10°C to 15°C
bit P_Floor_heating_in_reheat_mode;   //0 1
unsigned char P_floor_heating_fixed_set_point;          // 10°C to 30°C
bit  P_Winter_start;    //0 1

unsigned char P_low_set_point;            // 15°C to 30°C
unsigned char P_high_set_point;           // 15°C to 30°C

signed char P_evaporator_freeze; // -5°C to 5°C
unsigned char P_heating_water;            // 15°C to 75°C
unsigned char P_low_speed_evaporator;  //0-100
unsigned char P_med_speed_evaporator;  //0-100
unsigned char P_high_speed_evaporator;  //0-100

// A/D Prevodnik
unsigned int adconv (unsigned char channel)
{
 ADCON = 0x0000 | (channel & 0x0F);
 ADST = 1;
 while (ADBSY);
 ADST = 0;
 return (ADDAT & 0x03FF);
}
////////// HODINY
// ...Nastavi datum do hodin
void nastav_datum(unsigned char den1,unsigned char datum1,unsigned char mesic1,unsigned
char rok1)
{
 datum.day[10] = den1;    //nastaveni Pondeli, dne,  mesice, roku
 datum.date = datum1;
 datum.month = mesic1;
 datum.year = rok1;      //je od 00 do 99
```

```
 set_date(&datum);     // nastav datum do hodin
}

// ... Nastavi cas do hodin
void nastav_cas(unsigned char hodina,unsigned char minuta,unsigned char sekunda,unsigned
char A_P)
{
 cas.sec = sekunda;    //nastaveni sekund, minut, hodin, 24hod. cyklus
 cas.min = minuta;
 cas.hour = hodina;
 cas.am_pm = A_P;
 set_time(&cas);       // nastav hodiny
}
//////// KONEC HODIN

interrupt 0x23
void GT1_viIsrTmr3(void)
{
 button_timer();
  tlac0=0;
  tlac1=0;
  tlac2=0;
  tlac3=0;

}  //end casovac

interrupt 0x22
void GT1_viIsrTmr2(void)
{

    if(data[0]==0){ reguest_bit = 1;}
   if ((reguest_bit ==1)&&(data[0]!=7)&&(data[0]!=80))
     {
      reguest++;
     switch(reguest)
             {
                    case 1:  {reguest_reg[0] =1;
                                  reguest_reg[1] = on_send; break;}
                    case 2:  {reguest_reg[0] =2; reguest_reg[1]= on_send;  break;}
                    case 3:  {reguest_reg[0] =3;
                                  reguest_reg[1] = on_send;
                           reguest_reg[4] = output_bts6208_62;
                           reguest_reg[5] = driver_zone;
                           reguest_reg[6] = fresh_air_driver_zone; break;}
               default: {reguest_reg[0]=1;reguest=1;}
             }

      posli=(unsigned char*)&reguest_reg;
      ld_modata_16x(4,posli);
```

```
       send_mo_16x(4);
      }

}
//....Preruseni P1H.0.......TL 1
interrupt 0x18
void INT_viIsrEx0(void)
  {
  tlac0=1;
  i=0;
  }
//....Preruseni P1H.1.......TL 2
interrupt 0x19
void INT_viIsrEx1(void)
{
 tlac1=1;
  i=0;
}
//....Preruseni P1H.2.......TL 3
interrupt 0x1A
void INT_viIsrEx2(void)
{
 tlac2=1;
}
//....Preruseni P1H.3.......TL 4
interrupt 0x1B
void INT_viIsrEx3(void) {
 tlac3=1;
}

interrupt(0x40)
void CAN_interrupt (void)
{
    unsigned char status, intid;
    unsigned char download_data_buf[8] = { 0, 0, 0, 0, 0, 0, 0, 0 };

reguest_bit = 0;
    while((intid=IR) != 0)
    {
            WDTCON = 0x0001;
       _srvwdt();
       status=SR;                                   /* copy status REG to status variable
    */
            SR=0;                                       /* clear status register        */
            switch (intid)
            {
      case 1:                                  // Status Change Interrupt

      if (CR & 0x04)              // if SIE is set (status interrupts)
```

```
    {
        if (status & 0x08) { data[0]=8;}      // transmit interrupt
        if (status & 0x10) { data[0]=10;}     // receive interrupt
        if (status & 0x07) { data[0]=7;}      // CAN bus error interrupt
    }
    if (CR & 0x08)              // if EIE is set (error interrupts)
    {
        if (status & 0x40) { data[0]=40;}     // error counter warning
        if (status & 0x80) { data[0]=80;}     // bus off situation
          CR = (CR & 0xfe); // recover from BOFF (clear INIT)
    }
    break;

 case 3:                        // Message 1 Interrupt
        {

        rd_modata_16x(1, download_data_buf);
         for (i=0;i<8;i++)
                {input[i] = download_data_buf[i]; }
        }   break;


   case 4:                      // Message 2 Interrupt
        {
        rd_modata_16x(2, download_data_buf);

        for (i=0;i<8;i++)
                {ad_input1[i] = download_data_buf[i]; }
     }break;

   case 5:                      // Message 3 Interrupt
        {
     rd_modata_16x(3, download_data_buf);
                        for (i=0;i<8;i++)
                {ad_input2[i] = download_data_buf[i];}
          }break;

   case 6:                      // Message 2 Interrupt
   {
     *msgctrl_ptr_16x[3] = 0x7FFD; // reset INTPND
     if (status & 0x08)         // if transmit interrupt... (TXOK=1)
     {
                data[0]=13;
     }
    } break;

  } // end  switch (intid)
}   // end while ((intid=IR) != 0)
reguest_bit = 1;
```

```
}    // void CAN_interrupt (void)



void main(void)
{
  WDTCON = 0x0001;
  _srvwdt();       // Rychlost programu 323Hz

  i=0;

  T2IC = 0x006B;
  T2CON = 0x0044;
  T2R = 1;

  T3IC = 0x006F;
  T3CON = 0x0044;
  T3R = 1;

  DP1H = 0x0000;

  EXICON = 0x00AA; //    10101010      2A
  CC8IC = 0x0065;  //      nastaveni priority preruseni  P1H.0
  CC9IC = 0x0066;  //      nastaveni priority preruseni  P1H.1
  CC10IC = 0x0067;  //    nastaveni priority preruseni  P1H.2
  CC11IC = 0x0068;  //    nastaveni priority preruseni  P1H.2

    XP0IC = 0x77;   //povolit preruseni od CAN, priorita 13, group 3

    init_can_16x(250,0,1,1); //250kBd
    C1BTR = 0x2344;

    def_mo_16x (1,XTD_MES,0x0000011,RX,8,0,1);
    def_mo_16x (2,XTD_MES,0x0000012,RX,8,0,1);
    def_mo_16x (3,XTD_MES,0x0000013,RX,8,0,1);
    def_mo_16x (4,XTD_MES,0x0000001,TX,8,0,0);


     language = CESKY;              //nastaveni cestiny
    init_1307(24,RUN,0x00);         //inicializace hodin

   //  nastav_datum(11,19,05,04);        // nastaveni datumu
   //  nastav_cas(17,47,00,'P');       // nastaveni hodin
    //write_1307_RAM(0x00,text,24);  //zapsani textu do hodin
    //read_1307_RAM(0x00,rxtext,24); //cteni textu z hodin

 tlac=0;
 tlac_norm=0;
```

```
 passangers_zone=0;
 driver_zone =0;
 set_driver_zone=1;
 set_passangers_zone=0;
 driver_led = 1;
 passangers_led =0;

 driver_temperature = 22;
 passangers_temperature = 22;
// out_temperature=22;
 setup_menu=0;
 setup_menu_help=0;
 set_value=0;
 mode = 0;

 ODP8 = 0x0000;      // set port open drain control register
 DP8  = 0x000F;      // set port direction register
 P8= 0x0;

 display_init();       //inicializace disp.
 clear_lcd();          //vymazani disp.
 set_special_char();
//nastav_hodnoty();

 IEN=1;
  on=0;
  while(1)
{
 WDTCON = 0x0001;
 _srvwdt();       // Rychlost programu 323Hz
   read_time(&cas);     // cte cas
 read_date(&datum);   // cte datum
// measure_driver_temperature     = ((ad_input1[0])|(ad_input1[1]<<2));
// measure_passangers_temperature = ((ad_input1[2])|(ad_input1[3]<<2));
 measure_out_temperature        = ((ad_input1[0])|(ad_input1[1]<<2));


 //driver_temperature = (0.015*measure_driver_temperature)+15;
// passangers_temperature = (0.015*measure_passangers_temperature)+15;

 out_temperature =(0.107*measure_out_temperature-35);
  //          out_temperature   = (ad_input1[0]);

if((P500==0)&&(PO_P50==0))
            {PO_P50=1;
             on=~on;
             write_EE_intK(0x1120,driver_temperature,1); // adesa INT, co se ma ulozit
SIGNED LONG, pocet bytu
             write_EE_intK(0x1130,passangers_temperature,1);
```

```
                write_EE_intK(0x1140,out_temperature,1);
                                }

     if((P500==1)&&(PO_P50==1))
                {PO_P50=0;}

 on_send = on;

if (on==0) {send_hodinys(0,0,cas.hour,cas.min,cas.sec);

           }

      else
   { // zacatek OFF

    if (setup_menu==0)
  {
    DP8  = 0x000F;      // set port direction register

 send_hodiny(0,7,cas.hour,cas.min);


  send_cislo(out_temperature,0,15,3,0,5);
  send_znak(0xDF,0,18);
  send_znak(0x43,0,19);

  send_cislo(driver_temperature,1,1,2,0,5);
 // send_znak(0xDF,1,3);
 // send_znak(0x43,1,4);

  send_cislo(passangers_temperature,1,6,2,0,5);
  send_znak(0xDF,1,8);
  send_znak(0x43,1,9);

  button();

  output_bts6208_61 = driver_zone;
if(set_passangers_zone==0)
     {
    driver_led = 1;
    passangers_led =0;
     }else
     {
     driver_led = 0;
    passangers_led =1;
     }


     switch(mode)
```

```
        {
        case 0:         {
                                send_text(COOL,0,0);
                                if( out_temperature > driver_temperature ) output_bts6208_62     =
0x1;
                                  else output_bts6208_62 = 0x0;
                                } break;

        case 1:    {
                                send_text(HEAT,0,0);
                                if( out_temperature < driver_temperature ) output_bts6208_62     =
0x2;
                                  else output_bts6208_62 = 0x0;
                                } break;

        case 2:         {
                                send_text(AUTO,0,0);
                                if( out_temperature > driver_temperature )  output_bts6208_62     =
0x1;
                                if( out_temperature < driver_temperature )  output_bts6208_62     =
0x2;
                                if( out_temperature == driver_temperature )output_bts6208_62     =
0x0;

                                } break;


        case 3:         send_text(VENT,0,0);  break;
        }
        if(set_passangers_zone==1)
        {

                switch(passangers_zone)
                {
                case 0:   {send_znak(0xFF,1,11);  break;}
                case 1:   {send_znak(0xFF,1,11);  send_znak(0xFF,1,12);  break;}
                case 2:    {send_znak(0xFF,1,11);  send_znak(0xFF,1,12);  send_znak(0xFF,1,13);
break;}
                case 3:   {send_text(AUTO,1,11);  break;}
                }
        }
        else
        {

          switch(driver_zone)
                {
                case 0:   {send_znak(0xFF,1,11);  break;}
                case 1:   {send_znak(0xFF,1,11);  send_znak(0xFF,1,12);  break;}
```

```
                case 2:     {send_znak(0xFF,1,11);  send_znak(0xFF,1,12);  send_znak(0xFF,1,13);
break;}
                case 3:    {send_text(AUTO,1,11);  break;}
                }
        }
        if(set_passangers_zone==1)
        {

          switch(fresh_air_passangers_zone)
                {
                case 0:           send_text(OFF,1,16);  break;
                case 1:     send_text(P50,1,16);  break;
                case 2:           send_text(P100,1,16);  break;
                case 3:           send_text(AUTO,1,16);  break;
                }
        }
        else
        {

         switch(fresh_air_driver_zone)
                {
                case 0:           send_text(OFF,1,16);  break;
                case 1:     send_text(P50,1,16);  break;
                case 2:           send_text(P100,1,16);  break;
                case 3:           send_text(AUTO,1,16);  break;
                }
        }
      }// konec menu
       else
       {
        DP8  = 0x0000;      // set port direction register
        menu();

       if (set_value==1)
       {send_znak(0xFF,1,19);}
       else {send_znak(' ',1,19);}

       } // konec servise menu
     }   // konec OFF
       send_disp();         // prepise vsechny znaky na displ.

     //i++;
     //if (i>100) {display_init();  i=0;}

     }
     }
```

1.2 Menu

```
#include <REG164CR.H>
#include "menu.h"
#include "eeprom.h"

                                                    //12345678901234567890
#define SERVICE_MENU                    "SERVICE MENU"
#define ambient_switch_point "Ambient switch point"
#define ambient_floor_heat        "Ambient floor heat"
#define REHEAT_mode                     "REHEAT mode"
#define DELTA_T                                 "DELTA T"
#define Floor_heating_mode   "Floor heating mode"
#define Floor_heating_offset "Floor heating offset"
#define Floor_heating_in     "Floor heating in"
#define reheat_mode                "reheat mode"
#define floor_heating_fixed        "floor heating fixed"
#define set_point                  "set point"
#define Winter_start_of      "Winter start of"
#define compressor_clutch          "compressor clutch"
#define low_set_point              "low set point"
#define high_set_point             "high set point"
#define evaporator_freeze          "evaporator freeze"
#define heating_water              "heating water"
#define switch_point               "switch point"
#define low_speed_evaporator "low speed evaporator"
#define blower                          "blower"
#define med_speed_evaporator "med speed evaporator"
#define high_speed_evaporator "high speed evaporator"
#define stupne "°C"


void menu(void)
{
if (read==0)
{

P_ambient_switch_point = read_EE_int(0x1110,1); // 0°C to 15°C
P_ambient_floor_heat = read_EE_int(0x1020,1);        // 0°C to 30°C
P_REHEAT_mode = read_EE_int(0x1030,1);                      // 00 01 02 03
P_DELTA_T = read_EE_int(0x1040,1);                                  // 1°C to 15°C
P_Floor_heating_mode = read_EE_int(0x1050,1);                       // 0 1
P_Floor_heating_offset = read_EE_int(0x1060,1);      // 10°C to 15°C
P_Floor_heating_in_reheat_mode = read_EE_int(0x1070,1);   //0 1
P_floor_heating_fixed_set_point = read_EE_int(0x1080,1);    // 10°C to 30°C
```

```
P_Winter_start = read_EE_int(0x1090,1);    //0 1

P_low_set_point = read_EE_int(0x10A0,1);                // 15°C to 30°C
P_high_set_point = read_EE_int(0x10B0,1);     // 15°C to 30°C

P_evaporator_freeze = read_EE_int(0x10C0,1); // -5°C to 5°C
P_heating_water = read_EE_int(0x10D0,1);                // 15°C to 75°C
P_low_speed_evaporator = read_EE_int(0x10E0,1);  //0-100
P_med_speed_evaporator = read_EE_int(0x10F0,1);  //0-100
P_high_speed_evaporator = read_EE_int(0x1100,1);  //0-100
read=1;
}
switch(servise_menu)
            {
            case 0:    {
                                    send_text(SERVICE_MENU,0,0);
                                    break;
                                }
            case 1:    {send_text(ambient_switch_point,0,0);
                                    send_cislo(P_ambient_switch_point,1,0,2,0,4);
                                    send_znak(0xDF,1,2);
                                    send_znak('C',1,3);
                                    break;}


            case 2:    {send_text(ambient_floor_heat,0,0);
                                    send_cislo(P_ambient_floor_heat,1,0,2,0,4);
                                    send_znak(0xDF,1,2);
                                    send_znak('C',1,3);
                                break;}

            case 3:    {send_text(REHEAT_mode,0,0);
                                    send_cislo(P_REHEAT_mode,1,0,2,0,4);
                                    break;}

            case 4:    {send_text(DELTA_T,0,0);
                                    send_cislo(P_DELTA_T,1,0,2,0,4);
                                    send_znak(0xDF,1,2);
                                    send_znak('C',1,3);
                                    break;}

            case 5:    {send_text(Floor_heating_mode,0,0);
                                    send_cislo(P_Floor_heating_mode,1,0,1,0,4);
                                    break;}

            case 6:    {send_text(Floor_heating_offset,0,0);
                                    send_cislo(P_Floor_heating_offset,1,0,2,0,4);
                                    send_znak(0xDF,1,2);
                                    send_znak('C',1,3);
```

```
                    break;}

case 7:   {send_text(Floor_heating_in,0,0);
                    send_text(reheat_mode,1,0);
                    send_cislo(P_Floor_heating_in_reheat_mode,1,11,2,0,4);
                    break;}

case 8:   {send_text(floor_heating_fixed,0,0);
                    send_text(set_point,1,0);
                    send_cislo(P_floor_heating_fixed_set_point,1,10,2,0,4);
                    send_znak(0xDF,1,12);
                    send_znak('C',1,13);
                    break;}

case 9:   {send_text(Winter_start_of,0,0);
                    send_text(compressor_clutch,1,0);
                    send_cislo(P_Winter_start,1,18,1,0,4);
                    break;}

case 10:   {send_text(low_set_point,0,0);
                     send_cislo(P_low_set_point,1,0,2,0,4);
                     send_znak(0xDF,1,2);
                     send_znak('C',1,3);
                     break;}

case 11:   {send_text(high_set_point,0,0);
                     send_cislo(P_high_set_point,1,0,2,0,4);
                     send_znak(0xDF,1,2);
                     send_znak('C',1,3);

                     break;}

case 12:   {send_text(evaporator_freeze,0,0);
                     send_text(switch_point,1,0);
                     send_cislo(P_evaporator_freeze,1,13,3,0,4);
                     send_znak(0xDF,1,16);
                     send_znak('C',1,17);
                     break;}

case 13:  {send_text(heating_water,0,0);
                    send_text(switch_point,1,0);
                    send_cislo(P_heating_water,1,13,2,0,4);
                    send_znak(0xDF,1,15);
                    send_znak('C',1,16);
                    break;}

case 14:  {send_text(low_speed_evaporator,0,0);
                    send_text(blower,1,0);
                    send_cislo(P_low_speed_evaporator,1,8,3,0,4);
```

```
                                send_znak('%',1,11);

                                break;}


        case 15:  {send_text(med_speed_evaporator,0,0);
                                send_text(blower,1,0);
                                send_cislo(P_med_speed_evaporator,1,8,3,0,4);
                                send_znak('%',1,11);

                                break;}

        case 16:  {send_text(high_speed_evaporator,0,0);
                                send_text(blower,1,0);
                                send_cislo(P_high_speed_evaporator,1,8,3,0,4);
                                send_znak('%',1,11);
                                break;}

        }
}
```

1.3 Button

```
#include <REG164CR.H>
#include "button.h"

void nastav_hodnoty (void)
{
write_EE_intK(0x1010,22,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
write_EE_intK(0x1110,P_ambient_switch_point,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
write_EE_intK(0x1020,P_ambient_floor_heat,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
write_EE_intK(0x1030,P_REHEAT_mode,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
write_EE_intK(0x1040,P_DELTA_T,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
write_EE_intK(0x1050,P_Floor_heating_mode,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
write_EE_intK(0x1060,P_Floor_heating_offset,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
write_EE_intK(0x1070,P_Floor_heating_in_reheat_mode,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
write_EE_intK(0x1080,P_floor_heating_fixed_set_point,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
write_EE_intK(0x1090,P_Winter_start,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
```

```
    write_EE_intK(0x10A0,P_low_set_point,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
    write_EE_intK(0x10B0,P_high_set_point,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
    write_EE_intK(0x10C0,P_evaporator_freeze,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
    write_EE_intK(0x10D0,P_heating_water,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
    write_EE_intK(0x10E0,P_low_speed_evaporator,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
    write_EE_intK(0x10F0,P_med_speed_evaporator,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
    write_EE_intK(0x1100,P_high_speed_evaporator,1); // adesa INT, co se ma ulozit SIGNED LONG, pocet bytu
    }
    void button_timer(void)
    {
      i++;
                 if((tlac0==1)&&(setup_menu==1))
       {
       set_value=1;
                 }
      if((tlac1==1)&&(setup_menu==1))
       {
       set_value=0;
        }
     //---------in servise menu
        if((P1H0==0)&&(i==1)&&(setup_menu==0))
          {
          setup_menu=1;
          i=0;
                read=0;
                }
        //---------out servise menu

        if((P1H1==0)&&(i==1)&&(setup_menu==1))
          {
          setup_menu=0;
          i=0;
                nastav_hodnoty();
                servise_menu=0;
                driver_temperature = read_EE_int(0x1120,1);
                passangers_temperature = read_EE_int(0x1130,1);
                out_temperature = read_EE_int(0x1140,1);
                }
     if((tlac3==1)&&(set_driver_zone==1)&&(setup_menu==0))
      {
       if(driver_temperature < 40)
         {driver_temperature++;}
```

```
    else
            {driver_temperature=(-20);}
  }
  if((tlac3==1)&&(set_passangers_zone==1)&&(setup_menu==0))
  {
   if(passangers_temperature < 40)
     {passangers_temperature++;}
    else
            {passangers_temperature=(-20);}
  }
 if((tlac2==1)&&(set_driver_zone==1)&&(setup_menu==0))
  {
   if(driver_temperature >(-20))
     {driver_temperature--;}
    else
            {driver_temperature=40;}
  }
   if((tlac2==1)&&(set_passangers_zone==1)&&(setup_menu==0))
  {
   if(passangers_temperature >(-20))
     {passangers_temperature--;}
    else
            {passangers_temperature=40;}
  }
// ---------------------- service menu
  if((tlac3==1)&&(setup_menu==1)&&(set_value==0))
  {
   if(servise_menu < 16)
     {servise_menu++;}
    else
            {servise_menu=0;}
  }
   if((tlac2==1)&&(setup_menu==1)&&(set_value==0))
  {
   if(servise_menu >0)
     {servise_menu--;}
    else
            {servise_menu=16;}
  }
//---------------------------------------
  if((tlac3==1)&&(setup_menu==1)&&(set_value==1))
  {
switch(servise_menu)
            {
            case 0:    {
                                    break;}

            case 1:    {if(P_ambient_switch_point < 15)
                                      {P_ambient_switch_point++;}
```

```
                        else
                          {P_ambient_switch_point=0;}
                       break;}

case 2:    {if(P_ambient_floor_heat < 30)
                          {P_ambient_floor_heat++;}
                        else
                          {P_ambient_floor_heat=0;}
                  break;}

case 3:    {if(P_REHEAT_mode < 3)
                          {P_REHEAT_mode++;}
                        else
                          {P_REHEAT_mode=0;}
                 break;}

case 4:    {if(P_DELTA_T < 15)
                          {P_DELTA_T++;}
                        else
                          {P_DELTA_T=1;}
                       break;}

case 5:    {P_Floor_heating_mode=~P_Floor_heating_mode;
                       break;}

case 6:    {if(P_Floor_heating_offset < 15)
                          {P_Floor_heating_offset++;}
                        else
                          {P_Floor_heating_offset=10;}
                       break;}

case 7:    {P_Floor_heating_in_reheat_mode=~P_Floor_heating_in_reheat_mode;
                       break;}

case 8:    {if(P_floor_heating_fixed_set_point < 30)
                          {P_floor_heating_fixed_set_point++;}
                        else
                          {P_floor_heating_fixed_set_point=10;}
                       break;}

case 9:    {P_Winter_start=~P_Winter_start;
                       break;}

case 10:   {if(P_low_set_point < 30)
                          {P_low_set_point++;}
                        else
                          {P_low_set_point=10;}
                        break;}
```

```
                case 11:    {if(P_high_set_point < 30)
                                    {P_high_set_point++;}
                                   else
                                    {P_high_set_point=10;}
                                  break;}

                case 12:    {if(P_evaporator_freeze < 5)
                                    {P_evaporator_freeze++;}
                                   else
                                    {P_evaporator_freeze=(-5);}
                                  break;}

                case 13:   {if(P_heating_water < 75)
                                    {P_heating_water++;}
                                   else
                                    {P_heating_water=15;}
                                 break;}

                case 14:   {if(P_low_speed_evaporator < 100)
                                     {P_low_speed_evaporator=P_low_speed_evaporator+10;}
                                    else
                                     {P_low_speed_evaporator=0;}
                                  break;}


                case 15:   {if(P_med_speed_evaporator < 100)
                                     {P_med_speed_evaporator=P_med_speed_evaporator+10;}
                                    else
                                     {P_med_speed_evaporator=0;}
                                  break;}

                case 16:   {if(P_high_speed_evaporator < 100)

{P_high_speed_evaporator=P_high_speed_evaporator+10;}
                                    else
                                     {P_high_speed_evaporator=0;}
                                  break;}


          }
            } //end set valu ==1
        if((tlac2==1)&&(setup_menu==1)&&(set_value==1))
        {
      switch(servise_menu)
                {
                case 0:    {
                                  break;}
                case 1:    {if(P_ambient_switch_point >0)
                                     {P_ambient_switch_point--;}
                                    else
```

```
                    {P_ambient_switch_point=15;}
                  break;}

case 2:   {if(P_ambient_floor_heat >0)
                    {P_ambient_floor_heat--;}
                  else
                    {P_ambient_floor_heat=30;}
                  break;}

case 3:   {if(P_REHEAT_mode >0)
                    {P_REHEAT_mode--;}
                  else
                    {P_REHEAT_mode=3;}
                  break;}

case 4:   {if(P_DELTA_T >1)
                    {P_DELTA_T--;}
                  else
                    {P_DELTA_T=15;}
          break;}

case 5:   {P_Floor_heating_mode=~P_Floor_heating_mode;
                  break;}

case 6:   {if(P_Floor_heating_offset >10)
                    {P_Floor_heating_offset--;}
                  else
                    {P_Floor_heating_offset=15;}
                  break;}

case 7:   {P_Floor_heating_in_reheat_mode=~P_Floor_heating_in_reheat_mode;
                  break;}

case 8:   {if(P_floor_heating_fixed_set_point >10)
                    {P_floor_heating_fixed_set_point--;}
                  else
                    {P_floor_heating_fixed_set_point=30;}
                  break;}

case 9:   {P_Winter_start=~P_Winter_start;
                  break;}

case 10:   {if(P_low_set_point >10)
                    {P_low_set_point--;}
                  else
                    {P_low_set_point=30;}
                  break;}

case 11:   {if(P_high_set_point >10)
```

```
                                    {P_high_set_point--;}
                                  else
                                    {P_high_set_point=30;}
                                  break;}

            case 12:    {if(P_evaporator_freeze >(-5))
                                    {P_evaporator_freeze--;}
                                  else
                                    {P_evaporator_freeze=5;}
                                  break;}

            case 13:   {if(P_heating_water >10)
                                    {P_heating_water--;}
                                  else
                                    {P_heating_water=75;}
                                  break;}

            case 14:   {if(P_low_speed_evaporator >0)
                                    {P_low_speed_evaporator=P_low_speed_evaporator-10;}
                                  else
                                    {P_low_speed_evaporator=100;}
                                 break;}


            case 15:   {if(P_med_speed_evaporator >0)
                                    {P_med_speed_evaporator=P_med_speed_evaporator-10;}
                                  else
                                    {P_med_speed_evaporator=100;}
                                 break;}

            case 16:   {if(P_high_speed_evaporator >0)
                                    {P_high_speed_evaporator=P_high_speed_evaporator-10;}
                                  else
                                    {P_high_speed_evaporator=100;}
                                 break;}

    }
    } //end set valu ==1
 tlac0=0;
 tlac1=0;
 tlac2=0;
 tlac3=0;
}  //end casovac
 void button(void)
 {
  if((P51==0)&&(PO_P51==0))
             {PO_P51=1;
              if(mode < 3)
                     {mode++;}
```

```
            else
                    {mode=0;}
            }

    if((P51==1)&&(PO_P51==1))
            {PO_P51=0;}

//-----------------------------------------------------
    if((P1H4==0)&&(PO_P1H4==0))
            {PO_P1H4=1;
            set_passangers_zone=0;
            set_driver_zone=1;
      }
    if((P1H4==1)&&(PO_P1H4==1))
            {PO_P1H4=0;}
//-----------------------------------------------------

    if((P1H5==0)&&(PO_P1H5==0))
            {PO_P1H5=1;
            set_passangers_zone=1;
            set_driver_zone=0;
            }
    if((P1H5==1)&&(PO_P1H5==1))
            {PO_P1H5=0;}


//-----------------------------------------------------
   if((P53==0)&&(PO_P53==0)&&(set_driver_zone==1))
            {PO_P53=1;
             if(driver_zone < 3)
              {driver_zone++;}
            else
              {driver_zone=0;}
            }
  if((P53==0)&&(PO_P53==0)&&(set_passangers_zone==1))
            {PO_P53=1;
             if(passangers_zone < 3)
                    {passangers_zone++;}
             else
                    {passangers_zone=0;}
            }
    if((P53==1)&&(PO_P53==1))
            {PO_P53=0;}
//-----------------------------------------------------
   if((P52==0)&&(PO_P52==0)&&(set_driver_zone==1))
            {PO_P52=1;
             if(driver_zone > 0)
               {driver_zone--;}
             else
```

```
                    {driver_zone=3;}
                }
    if((P52==0)&&(PO_P52==0)&&(set_passangers_zone==1))
                {PO_P52=1;
                 if(passangers_zone > 0)
                        {passangers_zone--;}
                 else
                    {passangers_zone=3;}
                 }
     if((P52==1)&&(PO_P52==1))
                {PO_P52=0;}
//-----------------------------------------------------
    if((P54==0)&&(PO_P54==0)&&(set_driver_zone==1))
                {PO_P54=1;
                 if(fresh_air_driver_zone >0)
                        {fresh_air_driver_zone--;}
                 else
                        {fresh_air_driver_zone=3;}
                }
    if((P54==0)&&(PO_P54==0)&&(set_passangers_zone==1))
                {PO_P54=1;
                 if(fresh_air_passangers_zone > 0 )
                   {fresh_air_passangers_zone--;}
                 else
                   {fresh_air_passangers_zone=3;}
                 }
     if((P54==1)&&(PO_P54==1))
                {PO_P54=0;}
  //-----------------------------------------------------
     if((P55==0)&&(PO_P55==0)&&(set_driver_zone==1))
                {PO_P55=1;
                 if(fresh_air_driver_zone < 3)
                        {fresh_air_driver_zone++;}
                 else
                        {fresh_air_driver_zone=0;}
                }
    if((P55==0)&&(PO_P55==0)&&(set_passangers_zone==1))
                {PO_P55=1;
                 if(fresh_air_passangers_zone < 3)
                   {fresh_air_passangers_zone++;}
                 else
                   {fresh_air_passangers_zone=0;}
                 }

     if((P55==1)&&(PO_P55==1))
                {PO_P55=0;}
//-----------------------------------------------------
 }
```

1.4 Ovladač pro displej

```
// sbernice je port1H V MAIN program napsat DP1L=0xFF  !!!!

#include <REG164CR.H>
#include <math.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include <stdlib.h>
#include "display.h"


 sfrbit  rs _atbit(P1L, 0);
 sfrbit  rw _atbit(P1L, 1);
 sfrbit  en _atbit(P1L, 2);

 sfrbit  P1L7 _atbit(P1L, 7);
 // PORT P1L.3 nevyuzit

unsigned char _iram       s_disp_r1[21];     //radek displeje cislo 1
unsigned char _iram s_disp_r2[21];     //radek displeje cislo 2

void cekej(void)
{
unsigned long a;
for (a=0; a<5000; a++)
{
_nop();
}
}

// Z 8bit cisla da na port jen vyssi 4bity cisla
void H4bit(unsigned char vyssi)
{
unsigned char PORT;

vyssi &= 0xF0;
PORT = P1L & 0x0F;
P1L = PORT ^ vyssi;
}

// Z 8bit cisla da na port jen nizsi 4bity cisla
void L4bit(unsigned char nizsi)
{
unsigned char PORT;

nizsi &= 0x0F;
nizsi = nizsi << 4;  // posun vlevo
```

```
PORT = P1L & 0x0F;
P1L = PORT ^ nizsi;
}

//.....................CEKANI NA DISPLAY, JESTLI JE PRIPRAVEN
void wait_lcd(void)
{

//_srvwdt();

DP1L_7 =0;
rs=0;
rw=1;
en=1;
while(P1L7==1);
en=0;
rw=0;          // pred RW nesmi byt zadna instrukce, pak displ nezobrazuje
rs=0;
DP1L_7 =1;
}
//.....................Napise jeden znak na disp. Je nutne predem zadat pozici funkci lcd_post(x,y);
void send_char(unsigned char znak)
{
rs=1;
rw=0;
en=1;
H4bit(znak);
en=0;

rs=1;
rw=0;
en=1;
L4bit(znak);
en=0;
wait_lcd();
}
//.....................Napise jeden znak
void send_znak(unsigned char znak,unsigned char radek,unsigned char sloupec)
{
if ( radek==0 )  s_disp_r1[sloupec]=znak;
if ( radek==1 )  s_disp_r2[sloupec]=znak;
}
//.....................
void send_text (unsigned char *ukaz,unsigned char radek,unsigned char sloupec  )
{
unsigned char i, delka;

delka = strlen(ukaz);
```

```
for (i=sloupec; i < (sloupec+delka) ; i++)
{
  if ( radek==0 )  s_disp_r1[i]=*ukaz++;
  if ( radek==1 )  s_disp_r2[i]=*ukaz++;
}

}


void send_st (unsigned char *ukaz)
{
unsigned char poc=0;
while (poc++<20)
    {
    rs=1;
    rw=0;
   en=1;
    H4bit(*ukaz);
    en=0;

    rs=1;
    rw=0;
   en=1;
    L4bit(*ukaz++);
    en=0;
    wait_lcd();
    }
ukaz=0;
}

//.....................................
void display_init_c(unsigned char inicial)
{
en=1;
rw=0;
rs=0;
H4bit(inicial);
en=0;

rw=0;
rs=0;
en=1;
L4bit(inicial);
en=0;
wait_lcd();
}

void nastav_disp(void)
{
```

```
    DP1L &=0x8;
    DP1L ^=0xF7;  //vystup celeho portu krome bit 4

    display_init_c(0x28);   // 101 000    sbernice 4bit, 2radky, font znaku 5*7
    display_init_c(0x28);
    display_init_c(0x28);
    display_init_c(0x1C);   // 111 00   posun displ bez zapisu, smer posuvu vpravo
    display_init_c(0x1C);   // 111 00   posun displ bez zapisu, smer posuvu vpravo
    display_init_c(0x06);    //   11  0   po zapisu do DD RAM je ukazatel adresy inkrementovan, posun kurzoru po zapsani
    display_init_c(0x06);    //   11  0   po zapisu do DD RAM je ukazatel adresy inkrementovan, posun kurzoru po zapsani

    }


    //.............................................
    // Inicializace displ.
    void display_init(void)
    {
    nastav_disp();
    display_init_c(0x0C);   // 11 00  zapnuti displ, vypnuti kurzoru, vypnuti blikani
    clear_lcd();
    }

    //.........................
    // Vymazani displ.
    void clear_lcd(void)
    {
    display_init_c(0x01);
    }
    //.........................

    // nastavi pozici
    void lcd_post (unsigned char radek,unsigned char sloupec)
    {
    display_init_c(0x80+(radek*0x40)+sloupec);
    }
    //.................................


    //.............................Predani retezcu s_disp_rX na display a pak se vynuluji
    void send_disp(void)
    {
    unsigned char i;

    lcd_post(0,0);

    if ((on==1)&&(setup_menu==0))
```

```
   {
   for (i=0; i<20; i++)
     {
     switch(i)
      {
     case 15: {display_init_c(0x80);cekej();
                        lcd_post(0,14); send_char(0x4);}
     default:{rs=1;
                              rw=0;
                    en=1;
                              H4bit(s_disp_r1[i]);
                              en=0;
                               rs=1;
                              rw=0;
                    en=1;
                              L4bit(s_disp_r1[i]);
                              en=0;
                              wait_lcd();

                               }
      }
     }

lcd_post(1,0);
   for (i=0; i<20; i++)
     {
  switch(i)
      {
     case 0: {display_init_c(0x80);cekej();
                    lcd_post(1,0); send_char(0x1);
                      }break;
    case 3: {display_init_c(0x80);cekej();
                     lcd_post(1,3); send_char(0x3);
                      } break;
    case 5: {display_init_c(0x80);cekej();
                    lcd_post(1,5); send_char(0x2);
                      } break;
     default:{rs=1;
                              rw=0;
                    en=1;
                              H4bit(s_disp_r2[i]);
                              en=0;
                               rs=1;
                              rw=0;
                    en=1;
                              L4bit(s_disp_r2[i]);
                              en=0;
                              wait_lcd();
```

```
                                } break;
            }
          }
    }
    else
      {
       for (i=0; i<20; i++)
           {
                                rs=1;
                                rw=0;
                        en=1;
                                H4bit(s_disp_r1[i]);
                                en=0;
                                 rs=1;
                                rw=0;
                        en=1;
                                L4bit(s_disp_r1[i]);
                                en=0;
                                wait_lcd();

                                }

     lcd_post(1,0);

        for (i=0; i<20; i++)
        {
           rs=1;
                rw=0;
           en=1;
                H4bit(s_disp_r2[i]);
                en=0;

                rs=1;
                rw=0;
           en=1;
                L4bit(s_disp_r2[i]);
                en=0;
                wait_lcd();
      }
      }
    for (i=0; i<20; i++) {s_disp_r1[i]=' '; s_disp_r2[i]=' ';} //vynulovani retezce displeje
    }
    //.....................................

    //.......ZACATEK send_cislo.....................................................................
    void send_cislo(double VSTUP,unsigned char radek,unsigned char sloupec,unsigned char
poc_m,signed char poc_d,unsigned char poc_z)
    {
    // VSTUP = vstupni cislo;  radek = radek disp.{0,1}   sloupec = sloupec disp.<0,19>
```

```
    // poc_m-pocet mist zobrazeni       poc_d-pocet desetinnych mist    poc_z = od ktereho cisla se
mi zobrazi pomoci xEx
    //                                                  maximum poc_z=poc_m , 3 od 1000 => cislo
1E+3
    //        _ _ - 45 , 00235 E - 15
    //          a b  c d   e   f g  h
    //  a = pocet volnych mist pred cislem         b = znamenko cisla, + se nezobrazuje {0=+ ,1=-}
    //  c = pocet celych cisel                     d = desetinna carka
    //  e = pocet zobrazovanych desetinnych cisel  f = exponent { 0-nezobrazuje se, 1-zobrazeni }
    //  g = znamenko,=1 pokud bude zobrazeno       h = pocet mist exponentu 15 => h=2
    // ch = ciselne exponentu                      gg = 1-znamenko (-)  2-znamenko (+)

    double IN;
    float kk;
    unsigned char k;        //pocet celych cisel
    unsigned char a,b,c,d,dd,e,f,g,gg,h,ch,znam,z,z1;        //
    unsigned char i,j;      //pocitadla
    double celeR;        //cele cislo
    double desetinyR;  //desetiny cisla
    signed long int cele,desetiny;
    char szobraz[21];     //retezec,ve kterem bude cislo
    signed char  si;              //pocitadlo retezce
    unsigned char pom;     //pomocna promenna
    unsigned long pom1;

    a=b=c=d=dd=e=f=g=gg=h=ch=z=0;

    for (i=0; i<21; i++) szobraz[i]=' '-0x30; //vynulovani retezce

    IN = VSTUP;

    if (VSTUP<0)  IN*=-1;    //absolutni hodnota zaporneho cisla

    if (VSTUP<0)  znam=1; else znam=0;

    if (VSTUP==0)
        {
        szobraz[poc_m-1]='0'-0x30;
        }
        else{
        if (IN>=1) { kk = (log10 (IN))+1; k=kk;            //vypocet celych cisel
                     z1 = poc_m-poc_z;    }                //vypocet od ktereho radu se cislo zobrazuje
xxE+x
             else
             { kk = (log10 (IN))-1; k=abs(kk);  z=1;   //IN<1
                      z1 = poc_m-poc_z-1;                 //vypocet od ktereho radu se cislo zobrazuje
xxE+x
                      if ( (poc_m > (k+znam+z1)) )      { k=1;}
                    }
```

```
        if (z1<0) z1=0;                         //jenkontrola, aby nebylo z1<0
        if (poc_d>0) dd=1;                              //pro zobrazovani desetinne tecky

                if ( (poc_m < (k+znam+z+z1))&(poc_m>4) )
                          { if (IN>=1) { j=0;
                          pom1=1;
                          while (j<k-1) {pom1=pom1*10;  j++;}
                          IN=IN/pom1;                   //vydeleni na cislo x.xxx
                          if ((k-1)>=10) h=2; else h=1;   //h=2 e+11  h=1 e+9
                                                            poc_d = 20; gg=2; ch=k-1;
                                                          }
                                  else
                                                          { j=0;
                          pom1=1;
                          while (j<k) {pom1=pom1*10;  j++;}
                          IN=IN*pom1;
                                                            if (k>=10) h=2; else h=1;          //h=2
e+11  h=1 e+9
                          gg=1; ch=k; }

                    f=1; g=1;  k=1;
                  }

                        if (poc_m>=poc_d+k+znam+dd+f+g+h)
                      {b=znam; c=k; d=dd; e=poc_d; a=poc_m-c-e-znam-dd-f-g-h;}

                if (poc_m==k+znam+dd+f+g+h)
                          { b=znam; c=k; d=0; e=0; a=1;}                        //neni misto na
desetinnou tecku

                          if (poc_m==k+znam+f+g+h)
                  { b=znam; c=k; d=0; e=0;  a=0;}                       //zobrazi cele cislo se
znamenkem

                          if ( ((k+poc_d+1+znam+f+g+h)>poc_m) & (poc_m>(k+1+znam+f+g+h))
)     //snizeni poctu desetinnych mist
                  { b=znam; c=k; d=1;  e=poc_m-k-1-znam-f-g-h; a=0; }

        desetinyR = modf(IN,&celeR);        //cele=cela cast cisla, desetiny= desetiny cisla
        cele = celeR;

        si=a;                           //pocet mezer pred cislem
        if (b==1)
           szobraz[si++]='-'-0x30;                  //pokud je cislo zaporne,da znamenko minus, kladne
se nezobrazuje

        for (i=c ; i>0; i--) {  j=0;
                        pom1=1;
                              while (j<i-1) {pom1=pom1*10;  j++;}
```

```
                        pom = cele / pom1;
                        cele= cele % pom1;        //zbytek
                        szobraz[si++]= pom ;}     //zobrazi celou cast cisla

        if (d==1) szobraz[si++]=','-0x30;               //zobrazi desetinnou carku

        j=0;
        pom1=1;
        while (j<e) {pom1=pom1*10;  j++;}
        desetiny = (desetinyR * pom1);                  //vynasobi desetiny poctem desetinnych mist a
zaokrouhli
        for (i=e; i>0; i--) {    j=0;
                                 pom1=1;
                                 while (j<i-1) {pom1=pom1*10;  j++;}
                          pom = desetiny / pom1;
                    desetiny = desetiny % pom1;   //zbytek
                        szobraz[si++]= pom ;}         //zobrazi desetinnou cast cisla

        if (f==1)
          {szobraz[si++]='e'-0x30;}                     //zobrazi "e"
        if (gg==1)
          {szobraz[si++]='-'-0x30;}
        if (gg==2)
          {szobraz[si++]='+'-0x30;}
        if (h==1)
          {szobraz[si++]= ch;}
        if (h==2)
          {szobraz[si++]= ch/10; szobraz[si++]=ch%10;}
      }//konec pokud neni 0

     for (i=0; i<poc_m; i++)                                    //zapsani cisla do retezce radku
displ.
     {   if (radek==0)
         { s_disp_r1[sloupec++] = szobraz[i]+0x30 ; }
         if (radek==1)
         { s_disp_r2[sloupec++] = szobraz[i]+0x30 ; }
     }
     }
     void set_special_char(void)
     {

       display_init_c(0x40);
       display_init_c(0x40);
       send_char(0x1A);
       cekej();
       send_char(0xA);
       cekej();
       send_char(0xA);
       cekej();
```

```
send_char(0xA);
cekej();
send_char(0xF);
cekej();
send_char(0xD);
cekej();
send_char(0x1A);
cekej();
send_char(0x0);
cekej();

display_init_c(0x48);
display_init_c(0x48);
send_char(0x5);
cekej();
send_char(0xD);
cekej();
send_char(0x1D);
cekej();
send_char(0xD);
cekej();
send_char(0x5);
cekej();
send_char(0x1);
cekej();
send_char(0x1F);
cekej();
send_char(0x0);
cekej();

display_init_c(0x50);
display_init_c(0x50);
send_char(0x5);
cekej();
send_char(0x5);
cekej();
send_char(0x5);
cekej();
send_char(0x1D);
cekej();
send_char(0x9);
cekej();
send_char(0x1F);
cekej();
send_char(0x4);
cekej();
send_char(0x0);
cekej();
```

```
  display_init_c(0x58);
  display_init_c(0x58);
  send_char(0x1F);
  cekej();
  send_char(0x15);
  cekej();
  send_char(0x1C);
  cekej();
  send_char(0x4);
  cekej();
  send_char(0x4);
  cekej();
  send_char(0x5);
  cekej();
  send_char(0x7);
  cekej();
  send_char(0x0);
  cekej();

  display_init_c(0x60);
  display_init_c(0x60);
  send_char(0x1A);
  cekej();
  send_char(0xA);
  cekej();
  send_char(0xA);
  cekej();
  send_char(0xA);
  cekej();
  send_char(0xF);
  cekej();
  send_char(0xD);
  cekej();
  send_char(0x1A);
  cekej();
  send_char(0x0);
  cekej();
}

void send_special_char(void)
{
 display_init_c(0x80);
 cekej();
 lcd_post(0,15);
 send_char(0x0);
 lcd_post(1,0);
 send_char(0x1);
 lcd_post(1,6);
 send_char(0x3);
```

```
}
```

1.5 Ovladač pro displej - čas

```
#include <reg164ci.h>
#include <math.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>
#include "cas.h"

// Napise na disp. datum od umisteni
void send_datum(unsigned char radek,unsigned char sloupec,unsigned char DEN,unsigned char MESIC,unsigned char ROK)
{
                send_cislo(DEN,radek,sloupec,2,0,3);
                send_znak('.',radek,sloupec+2);
                if (MESIC<10) { send_znak('0',radek,sloupec+3); send_cislo(MESIC,radek,sloupec+4,1,0,3); }
                        else
                         send_cislo(MESIC,radek,sloupec+3,2,0,3);
                send_znak('.',radek,sloupec+5); // send_znak('2',radek,sloupec+6); send_znak('0',radek,sloupec+7);
                if (ROK<10) { send_znak('0',radek,sloupec+6); send_cislo(ROK,radek,sloupec+7,1,0,3); }
                        else
                        send_cislo(ROK,radek,sloupec+6,2,0,3);
}

// Napise na disp. hodiny od umisteni
void send_hodinys(unsigned char radek,unsigned char sloupec,unsigned char HOD,unsigned char MIN,unsigned char SEC)
{
   send_cislo(HOD,radek,sloupec,2,0,3);
   send_znak(':',radek,sloupec+2);
   if (MIN<10) {send_znak('0',radek,sloupec+3); send_cislo(MIN,radek,sloupec+4,1,0,3); }
            else
          {send_cislo(MIN,radek,sloupec+3,2,0,3);}
   send_znak(':',radek,sloupec+5);
   if (SEC<10) {send_znak('0',radek,sloupec+6); send_cislo(SEC,radek,sloupec+7,1,0,3); }
            else
                    send_cislo(SEC,radek,sloupec+6,2,0,3);
}

// Napise na disp. hodiny od umisteni
void send_hodiny(unsigned char radek,unsigned char sloupec,unsigned char HOD,unsigned char MIN)
```

```
    {
       send_cislo(HOD,radek,sloupec,2,0,3);
       send_znak(':',radek,sloupec+2);
       if (MIN<10) {send_znak('0',radek,sloupec+3); send_cislo(MIN,radek,sloupec+4,1,0,3); }
                    else
                  {send_cislo(MIN,radek,sloupec+3,2,0,3);}

    }

    //..Provede soucet hodin
    void soucet_hodin(unsigned char *DEN1,unsigned char *HOD1,unsigned char
*MIN1,unsigned char *SEC1, //vstup hodin1
                  unsigned char *DEN2,unsigned char *HOD2,unsigned char *MIN2,unsigned char
*SEC2, //vstup hodin2
                                  unsigned char *DENS,unsigned char *HODS,unsigned char
*MINS,unsigned char *SECS) //vystup hodin
    {
    unsigned char a;
    *SECS=0;
    *MINS=0;
    *HODS=0;
    *DENS=0;

    a= *SEC1 + *SEC2;
     if (a>=60) { *SECS = a-60; *MINS=*MINS+1;}
              else
                        *SECS = a;

    a=*MIN1 + *MIN2 + *MINS;
     if (a>=60) { *MINS = a-60; *HODS=*HODS+1;}
              else
                        *MINS = a;

    a=*HOD1 + *HOD2 + *HODS;
     if (a>=24) { *HODS = a-24; *DENS=*DENS+1;   }
              else
                        *HODS = a;

    *DENS=*DEN1 + *DEN2 + *DENS;

    }
    // Pricita sekundy a meni cas: dny,hodiny,minuty, sekundy
    void pricti_hodiny(unsigned char *DEN1,unsigned char *HOD1,unsigned char
*MIN1,unsigned char *SEK1)
    {
           *SEK1 = *SEK1 + 1;
           if (*SEK1 >= 60) { *SEK1=0;
                        *MIN1 = *MIN1 + 1;
                        if (*MIN1>=60) { *MIN1 = 0;
```

```
                                        *HOD1 = *HOD1 + 1;
                                        if (*HOD1>=24) { *HOD1 = 0;
                                                         *DEN1 = *DEN1 + 1;

                                                       }

                                      }
                           }
}
//.....................................................................
```

1.6 Ovladač pro Dallas DS1307 Real Time Clock

```
#include <reg164ci.h>
#include "ds1307.h"

 sfrbit  scl _atbit(P3, 8);
 sfrbit  sda _atbit(P3, 9);

//sda_dir  =  DP3_9;


//unsigned char const *dny[] = {"Pondeli   ","Utery     ","Streda    ","Ctvrtek   ",
//                                                        "Patek     ","Sobota    ","Nedele    "};

//unsigned char const *days[] = {"Monday    ","Tuesday   ","Wednesday ","Thursday  ",
//                                                         "Friday    ","Saturday  ","Sunday    "};

unsigned char language;
//.....................................................................

void txt_copy (unsigned char *dest, unsigned char *source)
{
 unsigned char i;

 i=0;
 do
 dest[i] = source[i];
 while (source[i++] != '\0');
}
//.....................................................................

void delay (unsigned int loop)
{
 unsigned int i;
 for(i=0;i<=loop;i++);
```

```
}
/*************************************************************************
/
/*                    INTERFACE ROUTINES                                 */
/*************************************************************************
/
/* 1. Issue a start condition */
/*****************************/

void start(void)
{
  DP3_9=1;
  sda=1;
  scl=1;
  delay(2);
  sda=0;
  delay(2);
  scl=0;

}

/*****************************/
/* 2. Issue a stop condition */
/*****************************/

void stop(void)
{
   DP3_9=1;
   sda=0;
    delay(2);
   scl=1;
    delay(2);
   sda=1;
    delay(2);
   scl=0;
}

/*****************************************/
/* 3. Issue a clock pulse and read SDA line */
/*****************************************/

unsigned char clock(void)
{
 unsigned char sda_val;

 delay(2);
 scl=1;             /* send SCL high */
 delay(1);
 sda_val=sda;       /* determine SDA status */
```

```
 scl=0;            /* send SCL low */

 return (sda_val);   /* return SDA status */
}

/*************************************************************************
***/
/* 4. Master issues an acknowledge by sending SDA low to signal data received */
/*************************************************************************
***/

void ack(void)
{
 DP3_9=1;
 sda=0;
 clock();
}

/*************************************************************************
***/
/* 5. DS1307 issues an acknowledge by pulling SDA low to signal data received */
/*************************************************************************
***/
void nack(void)
{
 DP3_9=0;
 clock();
}

/*****************************************/
/* 6. Sends 8 bit 'byte' to the DS1307 */
/*****************************************/
void wr_byte(unsigned char byte)
{
 unsigned char count;

 DP3_9=1;
 for (count = 0; count <= 7; count++)
 {
  if ((byte & 0x80) == 0x00) sda=0; else sda=1;
  clock();
  byte = byte << 1;
 }
 nack();
}
/*************************************************/
/* 7. Reads and returns 8 bits from the DS1307 */
/*************************************************/
unsigned char rd_byte(void)
```

```
{
 unsigned char count,data_byte;

 DP3_9=0;
 for (count = 0; count <= 7; count++)
 {
  data_byte = data_byte << 1;
  if ((clock()) == 0x01) data_byte = data_byte | 0x01;
 }
 return(data_byte);
}
//.....

void init_1307(unsigned char hour_mode, unsigned char clk_halt,
                        unsigned char ctrl_reg)
{
 unsigned char hs_val;

 //init Port 3
 ODP3=(ODP3|0x0200); // SDA open drain, SCL push pull
 DP3=(DP3|0x0100);   //SCL output
 DP3_9 = 1;        //SDA output

 start();
 wr_byte(0xD0);
 wr_byte(0x00);
 start();
 wr_byte(0xD1);
 hs_val = rd_byte(); nack();
 start();
 wr_byte(0xD0);
 wr_byte(0x00);
 if(clk_halt) wr_byte(hs_val|0x80); else wr_byte(hs_val&0x7F);

 start();
 wr_byte(0xD0);
 wr_byte(0x02);
 start();
 wr_byte(0xD1);
 hs_val = rd_byte(); nack();
 start();
 wr_byte(0xD0);
 wr_byte(0x02);
 if(hour_mode==12) wr_byte(hs_val|0x40); else wr_byte(hs_val&0x3F);

 start();
 wr_byte(0xD0);
 wr_byte(0x07);
 wr_byte(ctrl_reg);
```

```
 stop();
}
//.............................................

void read_time(sTIME *rd_time)
{
 unsigned char se,mi,ho;

 start();
 wr_byte(0xD0);
 wr_byte(0x00);

 start();
 wr_byte(0xD1);
 se = rd_byte(); ack();
 mi = rd_byte(); ack();
 ho = rd_byte(); nack();
 stop();

 rd_time->sec  = (10*((se&0x70)>>4)) + (se&0x0F);
 rd_time->min  = (10*((mi&0x70)>>4)) + (mi&0x0F);
 if(ho&0x40)
    //12-hr. mode
 {
  rd_time->hour = (10*((ho&0x10)>>4)) + (ho&0x0F);
  if(ho&0x20) rd_time->am_pm = 'P'; else rd_time->am_pm = 'A';
 }
 else
    //24-hr. mode
 {
  rd_time->hour = (10*((ho&0x30)>>4)) + (ho&0x0F);
  rd_time->am_pm = 0;
 }
}
//.............................................

void set_time(sTIME *wr_time)
{
 unsigned char se,mi,ho;

 start();
 wr_byte(0xD0);
 wr_byte(0x00);
 start();
 wr_byte(0xD1);
 se = rd_byte(); ack();
 mi = rd_byte(); ack();
 ho = rd_byte(); nack();
```

```
 se = (se&0x80) | ((wr_time->sec/10)<<4) + (wr_time->sec - (10*(wr_time->sec/10)));
 mi = ((wr_time->min/10)<<4) + (wr_time->min - (10*(wr_time->min/10)));
 ho = (ho&0x40) | ((wr_time->hour/10)<<4) + (wr_time->hour -(10*(wr_time->hour/10)));

 if(ho&0x40)  //12-hr. mode
 {
  if(wr_time->am_pm == 'P') ho |= 0x20; else ho &= 0xDF;
 }

 start();
 wr_byte(0xD0);
 wr_byte(0x00);
 wr_byte(se);
 wr_byte(mi);
 wr_byte(ho);
 stop();
}
//.....

void read_date(sDATE *rd_date)
{
 unsigned char day,date,month,year;

 start();
 wr_byte(0xD0);
 wr_byte(0x03);

 start();
 wr_byte(0xD1);
 day = rd_byte(); ack();
 date  = rd_byte(); ack();
 month = rd_byte(); ack();
 year  = rd_byte(); nack();
 stop();

 /*switch(language)
 {
  case 1:  {txt_copy(rd_date->day, dny[day-1]);break;}
  case 2:
  default: {txt_copy(rd_date->day, days[day-1]);break;}
 }*/

 rd_date->date  = (10*((date&0x30)>>4)) + (date&0x0F);
 rd_date->month = (10*((month&0x10)>>4)) + (month&0x0F);
 rd_date->year  = (10*(year>>4)) + (year&0x0F);
}
//.....
```

```
void set_date(sDATE *wr_date)
{
 unsigned char date,month,year;

 date = ((wr_date->date/10)<<4) + (wr_date->date - (10*(wr_date->date/10)));
 month = ((wr_date->month/10)<<4) + (wr_date->month - (10*(wr_date->month/10)));
 year = ((wr_date->year/10)<<4) + (wr_date->year -(10*(wr_date->year/10)));

 start();
 wr_byte(0xD0);
 wr_byte(0x03);
 wr_byte(wr_date->day[10]);
 wr_byte(date);
 wr_byte(month);
 wr_byte(year);
 stop();
}


//.....................................................

void write_1307_RAM(unsigned char address, char *wr_buf, unsigned char len)
{
 unsigned char i;

 address += 0x08;

 start();
 wr_byte(0xD0);
 wr_byte(address);

 for(i=0;i<len;i++) wr_byte(wr_buf[i]);

 stop();
}


//.....................................................

void read_1307_RAM(unsigned char address, char *rd_buf, unsigned char len)
{
 unsigned char i;

 address += 0x08;

 start();
 wr_byte(0xD0);
 wr_byte(address);
 start();
```

```
 wr_byte(0xD1);

 for(i=0;i<len;i++)
 {
  rd_buf[i] = rd_byte();
  if(i==(len-1)) nack(); else ack();
 }
 stop();
}
//.....................................................
```

1.7 Ovladač pro EEPROM 24c64

```
#include <REG164CR.H>

#include <MATH.H>

//#include <intrins.h>

#include "eeprom.h"


 sfrbit  scl _atbit(P3, 8);

 sfrbit  sda _atbit(P3, 9);


 //sbit sda_dir  = DP3^9;


//.....................................................

// Casova Proodleva


void cas_prodleva (unsigned int loop)

{

 unsigned int i;

 for(i=0;i<=loop;i++);

}
```

```
//*************************************************************************
*/
//*                    INTERFACE ROUTINES                    */
//*************************************************************************
*/
//* 1. Issue a start condition */
//*****************************/

void estart()
{
  DP3_9=1;
  sda=1;
  scl=1;
  cas_prodleva(2);     //2
  sda=0;
  cas_prodleva(2);     //2
  scl=0;
}

//*****************************/
//* 2. Issue a stop condition */
//*****************************/

void estop()
{
   DP3_9=1;
   sda=0;
    cas_prodleva(2);       //2
   scl=1;
```

```
    cas_prodleva(2);        //2
    sda=1;
    cas_prodleva(2);        //2
    scl=0;
}

//*****************************************/
//* 3. Issue a clock pulse and read SDA line */
//*****************************************/

unsigned char eclock()
{
 unsigned char sda_val;

 cas_prodleva(2);          //2
 scl=1;             /* send SCL high */
 cas_prodleva(1);          //1
 sda_val=sda;       /* determine SDA status */
 scl=0;             /* send SCL low */

 return (sda_val);   /* return SDA status */    // navraci status SDA
}

//*****************************************************************************
****/
//* 4. Master issues an acknowledge by sending SDA low to signal data received */
//*****************************************************************************
****/
```

```
void eack()
{
 DP3_9=1;  // nastaveni vystupu
 sda=0;
 eclock();
}

//*************************************************************************
****/
//* 5. DS1307 issues an acknowledge by pulling SDA low to signal data received */
//*************************************************************************
****/

void enack()
{
 DP3_9=0;  // nastaveni vstupu
 eclock();
}

//*************************************/
//* 6. Sends 8 bit 'byte' to the DS1307 */
//*************************************/
void ewr_byte(unsigned char byte)
{
 unsigned char count;

 DP3_9=1;
 for (count = 0; count <= 7; count++)
 {
```

```
 if ((byte & 0x80) == 0x00) sda=0; else sda=1;
 eclock();
 byte = byte << 1;
 }
}
//*********************************************/
//* 7. Reads and returns 8 bits from the DS1307 */
//*********************************************/
unsigned char erd_byte()
{
 unsigned char count, data_byte;

 DP3_9=0;
 for (count = 0; count <= 7; count++)
 {
  data_byte = data_byte << 1;
  if ((eclock()) == 0x01) data_byte = data_byte | 0x01;
 }
 return(data_byte);            // navraci nacteny byte
}

//.....................................................
//.....................................................
//        ZAPIS DAT DO EEPROM
//                adresa zapisu    , zapsana data, pocet zapsanych bytu
void write_EEPROM(unsigned int address, char *wr_buf, unsigned char len)
{
 unsigned char i;
 unsigned char addressL,addressH;
```

```
  addressL=address;
  addressH=address>>8;

 ODP3=(ODP3|0x0200); // SDA open drain, SCL push pull       10 0000 0000
 DP3=(DP3|0x0100);   //SCL output                          100 0000  P3.6  40h
                     //                                   1 0000 0000  P3.8  100h

 estart();
 ewr_byte(0xA0);          // Control BYTE EEPROM 1010 0000, nasleduje NACK
 enack();
 ewr_byte(addressH);          // Zapis vyssi adresy odkud se ma cist, nasleduje NACK
 enack();
 ewr_byte(addressL);       // Zapis nizsi adresy odkud se ma cist, nasleduje NACK
 enack();

 for(i=0;i<len;i++) { ewr_byte(wr_buf[i]); enack(); } // zapis jednotlivych bytu

 estop();
}

//////////////////////////////////////////////////////////
//.....................................................
//    NACTENI DAT Z EEPROM
//                  adresa cteni            , pocet ctenych bytu
signed long  read_EE_int(unsigned int address,  unsigned char len)
{
 unsigned char i;
 signed long cislo;
```

```
unsigned char addressL,addressH;

addressL=address;
addressH=address>>8;



cislo=0;

ODP3=(ODP3|0x0200); // SDA open drain, SCL push pull       10 0000 0000          ODP3.9
DP3=(DP3|0x040);   //SCL output                              100 0000   40h     DP3.6
                                                  //                1 0000 0000  100h     DP3.8
estart();
ewr_byte(0xA0);          // Control BYTE EEPROM 1010 0000, zapis adresy
enack();
ewr_byte(addressH);           // Zapis vyssi adresy odkud se ma cist, nasleduje NACK
enack();
ewr_byte(addressL);       // Zapis nizsi adresy odkud se ma cist, nasleduje NACK
enack();
estop();                 // Stop bit, kvuli moznemu preslechu
estart();                // START bit
ewr_byte(0xA1);          // Control BYTE EEPROM 1010 0001, cteni z adresy
enack();

for(i=0;i<(len);i++)
{
 if (i==0) cislo=erd_byte();  // else cislo=cislo+(_lrol_(erd_byte(),(i*8)));
  else {cislo=cislo+((erd_byte())*(i*8));}
```

```
 if(i==(len-1)) enack();
                else
                                eack();
 }
 estop();
 return cislo;
}

//.....................................................
////////////////////////////////////////////////////////////////
// Ulozeni do EEPROM integer
void write_EE_int(unsigned int address, signed long cislo, unsigned char velikost)
{
  unsigned char pole[4],i;

  for(i=0;i<velikost;i++)
  { pole[i]=cislo;
    cislo=cislo>>8;}

  write_EEPROM(address,&pole[0],velikost);

}

////////////////////////////////////////////////////////////////
// ZAPIS S KONTROLOU ULOZENI
void write_EE_intK(unsigned int address, signed long cislo, unsigned char len)
{
 unsigned char i,j,K;
 signed long cti_cislo;
```

```
i=0; K=0;

while(i<2)                          // 2 pokusy o zapsani do EEPROM
{
i++;
write_EE_int( address, cislo,  len );
 j=0;
 while(j<20)
 {
  cti_cislo = read_EE_int(address,len);              // nacteni cisla z EEPROM
 if (cislo==cti_cislo) {j=51; i=20; K=1;
                                WDTCON = 0x0001;
                    _srvwdt();
                                }
                     else
                                j++;

  }
 }
}


//////////////////////////////////////////////////////////////////////////////
//////////////////////////////////////////////////////////////////////////////
// Nacti pole z EE pro typ LONG
void read_EE_pole_long( unsigned long *p_pole,unsigned long adresa, unsigned int delka)
{
 unsigned int i;
```

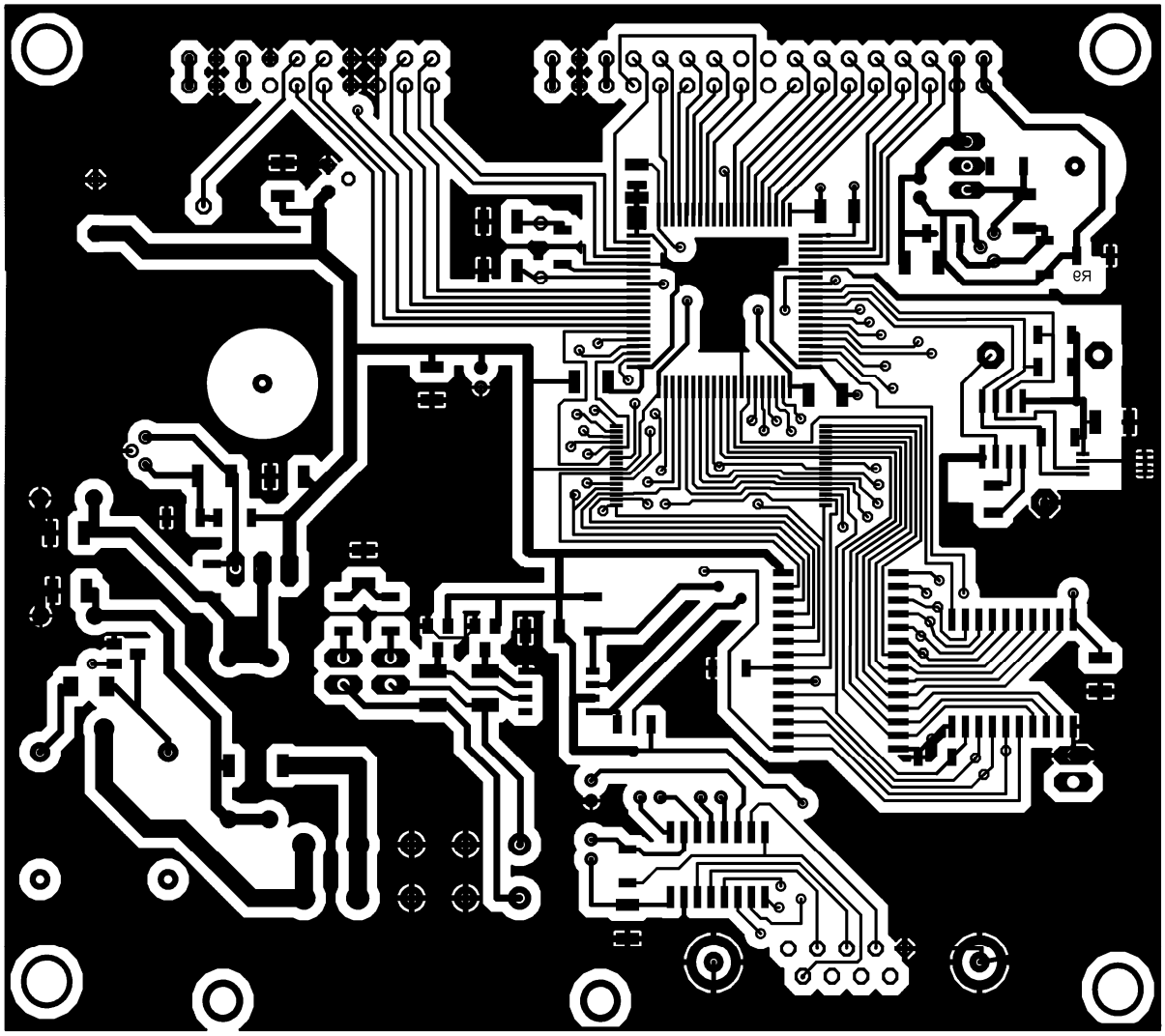
```
 signed long  pr;

 for (i=0; i<delka; i++)
    {
     WDTCON = 0x0001;
     _srvwdt();

    pr = read_EE_int( (adresa+i*4) , 4);
    *p_pole = pr;
     *p_pole++;
    }
}

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
// Nacti pole z EE pro typ INT
void read_EE_pole_int( unsigned int *p_pole,unsigned long adresa, unsigned int delka)
{
 unsigned int i;
 unsigned int  pr;

 for (i=0; i<delka; i++)
    {
     WDTCON = 0x0001;
     _srvwdt();

    pr = read_EE_int( (adresa+i*2) , 2);
    *p_pole = pr;
     *p_pole++;
```

```
    }
}
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
// Nacti pole z EE pro typ CHAR
void read_EE_pole_char( unsigned char *p_pole,unsigned long adresa, unsigned int delka)
{
 unsigned int i;
 unsigned char pr;
 unsigned long add;
 add = adresa;

 for (i=0; i<delka; i++)
    {
     WDTCON = 0x0001;
     _srvwdt();

    pr = read_EE_int( add , 1);
     add = add + 1;
    *p_pole = pr;
     *p_pole++;
    }
}
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
// Zapis pole do EE pro typ LONG
void write_EE_pole_long(unsigned long *p_pole,unsigned long adresa, unsigned int delka)
{
 unsigned int i;
```

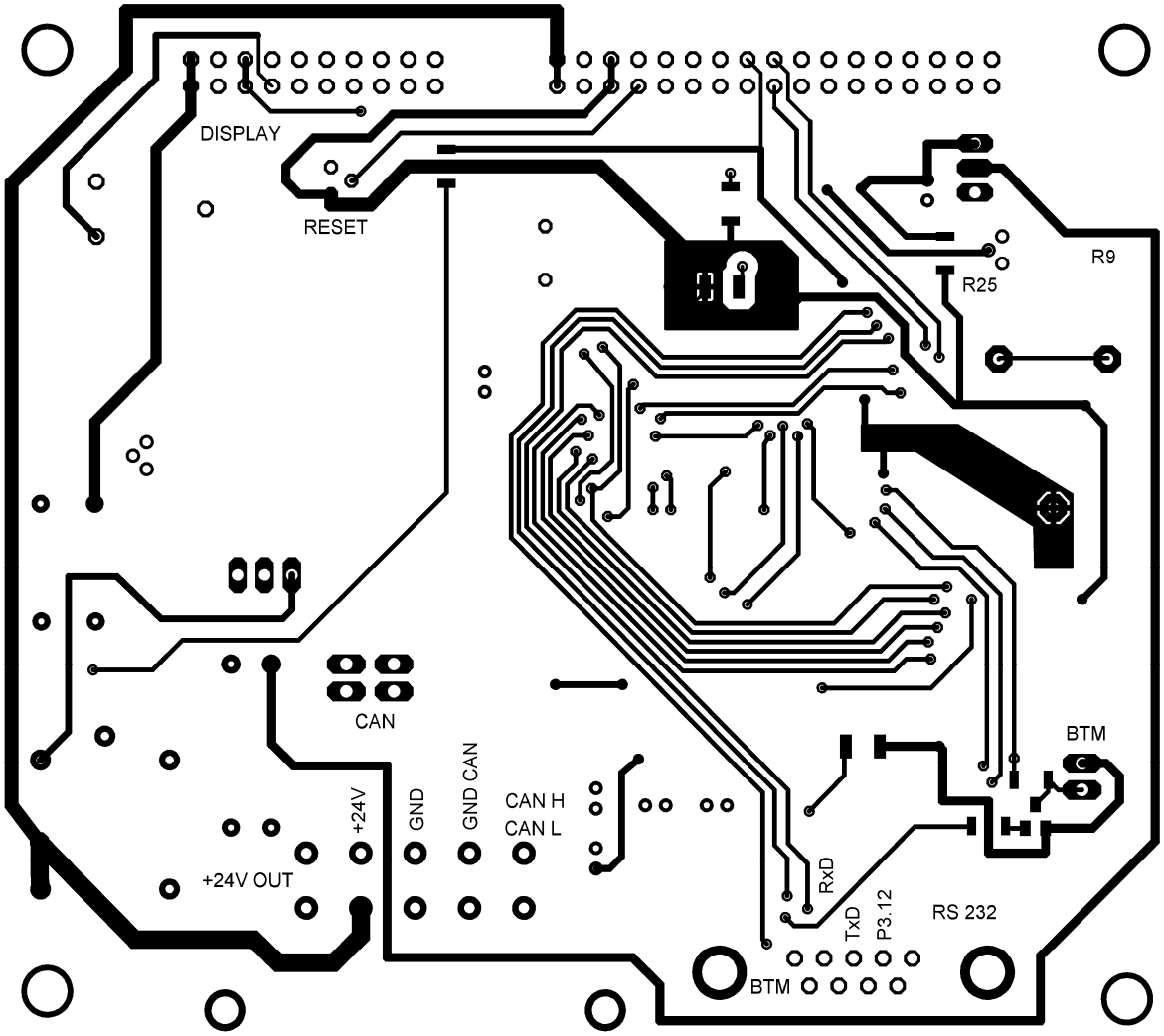
```
 unsigned long pr;

 for (i=0; i<delka; i++)
    {
     WDTCON = 0x0001;
     _srvwdt();

     pr = *p_pole++;

    write_EE_intK( (adresa+i*4) , pr, 4 );

    }
}
////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
// Zapis pole do EE pro typ INT
void write_EE_pole_int(unsigned int *p_pole,unsigned long adresa, unsigned int delka)
{
 unsigned int i;
 unsigned long pr;

 for (i=0; i<delka; i++)
    {
     WDTCON = 0x0001;
     _srvwdt();

     pr = *p_pole++;

    write_EE_intK( (adresa+i*2) ,pr, 2 );
```

```
    }
}

////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////
// Zapis pole do EE pro typ CHAR
void write_EE_pole_char(unsigned char *p_pole,unsigned long adresa, unsigned int delka)
{
 unsigned int i;
 unsigned int pr;
 unsigned long add;

 add = adresa;

 for (i=0; i<delka; i++)
    {
     WDTCON = 0x0001;
     _srvwdt();

     pr = *p_pole++;

    write_EE_intK( add , pr, 1 );
    add = add + 1;
    }
}
```

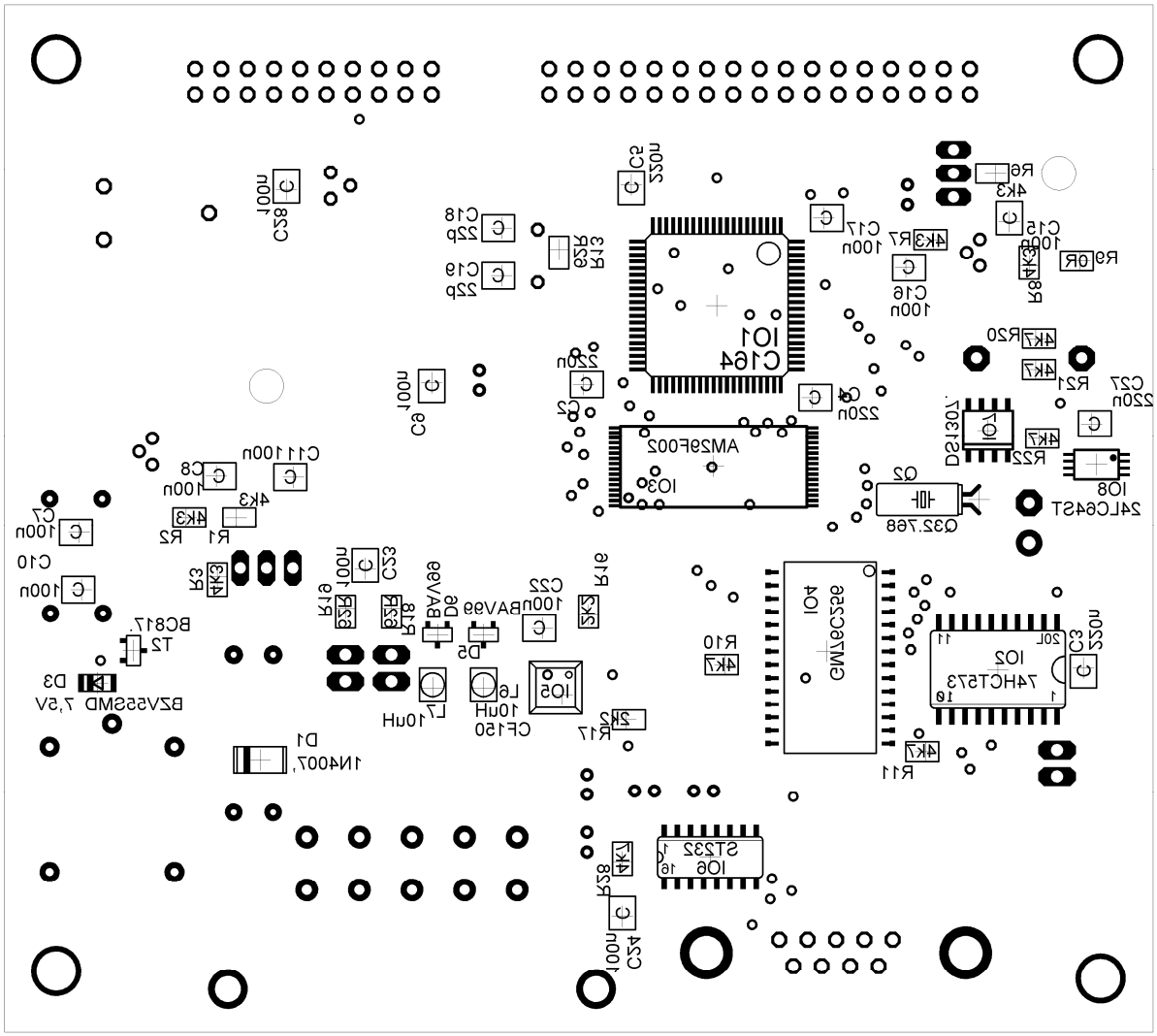
2 Podklady pro výrobu DPS



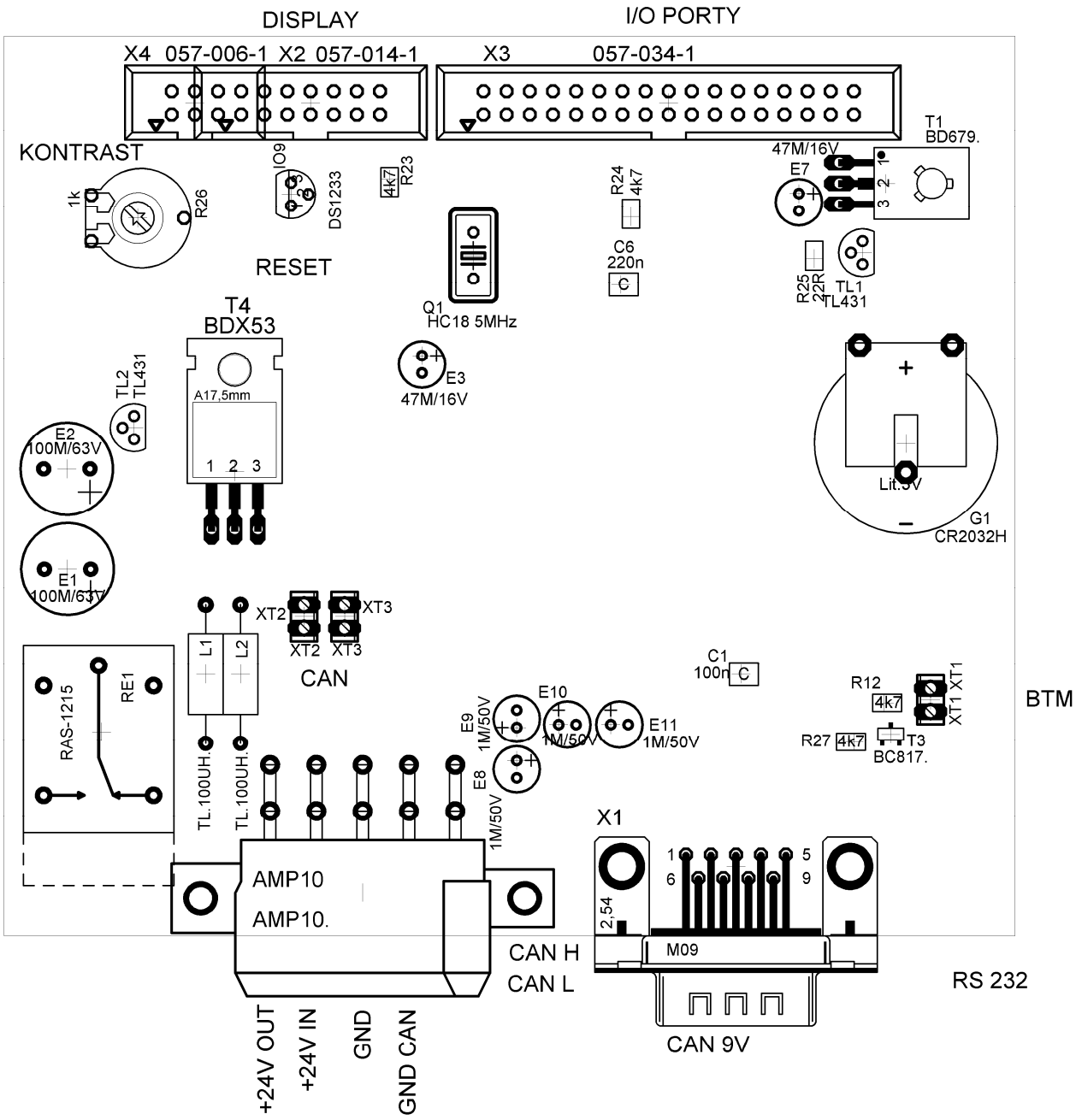
Obr.1 Deska plošného spoje - BOTTOM



Obr.2 Deska plošného spoje - TOP



Obr.3 Deska součástky - BOTTOM



Obr.4 Deska součástky - TOP

SCHEMA MIKROPROCESOROVE CASTI RIDICI JEDNOTKY RJ-SBH20