```
/*===================================================================
Title:      Digital Room Thermostate - Range -6.00 / 46.00°C
Author:     BITOULIS SPYRIDON
Date:       29/12/2003
Revision:   V1.38
-------------------------------------------------------------------
Digital Room Thermostate - Range -6.00 / 46.00°C
Microcontroller Avr 8-bit RISC (AT90S2313)
-------------------------------------------------------------------
    The microcontroller increments a counter every 2sec using
the Watchdog timer, after the RESET the microcontroller
makes all the jobs pending, like communication, temperature
measurement, manual operations, relay control, after that
is goes to power down mode using the sleep command
(0.040mA in sleep mode).
    Using the icNE555 & a NTC Thermistor 10K (4300) a
temperature measurement is been made every 1min. In series
with the NTC there is a Resistor 4Ê54 that reduces the current
during the temperature measurement.
    There is serial link with the PC for Downloading and Uploading
Parameters every 2sec an attempt is been made to link with the
PC. Only a READ or WRITE command can be made during this period.
===================================================================*/

#include <avr/io.h>
#include <avr/signal.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <avr/wdt.h>
#include <avr/eeprom.h>
#include <avr/pgmspace.h>

typedef unsigned char  u08;
typedef unsigned int  u16;
typedef unsigned long  u32;
/* DEFINITIONS ================================================*/
#define FALSE               0
#define TRUE                1

#define DISABLE             0
#define RUNNING             1
#define ACTIVATED           2
#define FINISHED            3

#define AUTO_RUNS           0
#define AUTO_STOPPED        1
#define MAN_DELAY_TIME      2
#define MAN_CHARGE_R        3
#define MAN_OPEN_EV_R       4
#define MAN_WAIT            5
#define MAN_CHARGE_F        6
#define MAN_OPEN_EV_F       7

#define PORTB_TRIGGER       0
#define PORTB_ENABLE        1
#define PORTB_RS232         2
#define PORTB_RUNNING       3
#define PORTB_MANUAL        4

#define PORTD_RELAY         4

#define AVG_SAMPLES         2  /* 2^AVG_SAMPLES = 4 */
#define PARAMSIZE           40 /* Number of Bytes */
#define UART_38400          12

#define UART_QUERY          'Q'
#define UART_READ           'R'
```

```
#define UART_WRITE          'W'
#define UART_R_ACK          'B'
#define UART_W_ACK          'S'

#define MAXIMUM_TH          4600
#define MINIMUM_TH          -600

#define E2PROM_DATA_LEN     10
#define NTC_DATA_LEN        53

/* VARIABLES ================================================*/
u16 Start_ICP;
u08 *pParam, NTC_Status;
int AvgSum              __attribute__ ((section (".noinit")));
u08 AvgCount            __attribute__ ((section (".noinit")));
/*Parameters Section */
u08 Cnt_seconds         __attribute__ ((section (".noinit")));  /* 0 Low */
u08 Relay_Status        __attribute__ ((section (".noinit")));  /* 0 High */
int Act_Therm           __attribute__ ((section (".noinit")));  /* 1 */
u08 AutoMode            __attribute__ ((section (".noinit")));  /* 2 Low */
u08 ManMode             __attribute__ ((section (".noinit")));  /* 2 High */
u16 Sp1                 __attribute__ ((section (".noinit")));  /* 3 */
u08 ForcedReset         __attribute__ ((section (".noinit")));  /* 4 Low */
u08 Sp3                 __attribute__ ((section (".noinit")));  /* 4 High */
u16 TimeNE555           __attribute__ ((section (".noinit")));  /* 5 */
u16 Time_On             __attribute__ ((section (".noinit")));  /* 6 */
u16 Time_Off            __attribute__ ((section (".noinit")));  /* 7 */
int AvgAct_Therm        __attribute__ ((section (".noinit")));  /* 8 */
u16 Sp4                 __attribute__ ((section (".noinit")));  /* 9 */
u08 CycleNE555          __attribute__ ((section (".noinit")));  /* 10 Low */
u08 Hysterisis_a        __attribute__ ((section (".noinit")));  /* 10 High */
int Setpoint_aut        __attribute__ ((section (".noinit")));  /* 11 */
u16 Time_aut_on         __attribute__ ((section (".noinit")));  /* 12 */
u16 Time_aut_off        __attribute__ ((section (".noinit")));  /* 13 */
int Setpoint_man        __attribute__ ((section (".noinit")));  /* 16 */
u16 Time_man_on         __attribute__ ((section (".noinit")));  /* 14 */
u16 Time_man_off        __attribute__ ((section (".noinit")));  /* 15 */
int Setpoint_manP       __attribute__ ((section (".noinit")));  /* 17 */
u16 Time_manP_off       __attribute__ ((section (".noinit")));  /* 18 */
u08 OperationModes      __attribute__ ((section (".noinit")));  /* 19 Low */
u08 Sp19                __attribute__ ((section (".noinit")));  /* 19 High */

/* E2prom memory */
u08 E2pos __attribute__((section(".eeprom"))) = FALSE;
u16 EEpromMem[E2PROM_DATA_LEN] __attribute__((section(".eeprom"))) = {
2590, 2285, 158, 237, 4600, 79, 158, 4600, 526, 7 };
/* NTC Thermistor Curve: -6.50 / 55.00°C */
u16 NTC_Curve[NTC_DATA_LEN] __attribute__((section(".eeprom"))) = {
62286, 58938, 55808, 52880, 50140, 47576, 45173, 42922,
40812, 38833, 36976, 35233, 33596, 32058, 30613, 29254,
27975, 26772, 25639, 24573, 23567, 22620, 21727, 20884,
20089, 19339, 18630, 17961, 17328, 16730, 16164, 15629,
15122, 14643, 14189, 13759, 13351, 12965, 12599, 12251,
11921, 11608, 11311, 11029, 10760, 10506, 10263, 10033,
9814, 9605, 9406, 9217, 9037 };
/* FUNCTIONS ================================================*/
/* Hardware */
void Init_AT90S2313( void );
/* NTC Resistor measurement */
void Start_NE555( void );
void Calculation( void );
void usDelay( u16 tmdelay );
/* Uart */
u08 Uart_get( void );
void Uart_put( u08 Uart_data );
void Communication( void );
/* Timing & Setpoints */
```

```
void Load_Man_Settings( void );
void Load_Auto_Settings( void );
void TimingController( void );
/*=================================================================*/
int main( void )
{
    /* Initialeze AT90S2313 hardware */
    Init_AT90S2313( );
    /* Temperature measurement Status */
    NTC_Status = DISABLE;
    /* Counts Watchdog RESET increment every 2 seconds */
    ++Cnt_seconds;
    /* LED System is running 0V (on) */
    PORTB ^= _BV(PORTB_RUNNING);
    usDelay(400);
    PORTB ^= _BV(PORTB_RUNNING);
    /* LED Relay /Manual Off when Relay is ON (Power Down) */
    PORTB |= _BV(PORTB_MANUAL);
    /* Job function every 3min */
    if( Cnt_seconds >= CycleNE555 ){
        Cnt_seconds = 0;
        /* Start a NTC Resistor measurement every 3 minutes */
        NTC_Status = ACTIVATED;
        Start_NE555( );
    }
    /* Serial Link with PC every 2 seconds */
    if( bit_is_clear( PIND, PIND5 ) )
        Communication( );
    /* Check NTC Resistor measured FINISHED */
    /* If NTC measurement Started */
    if( NTC_Status != DISABLE ){
        /* Wait for NTC measurement to finished */
        while( NTC_Status != FINISHED );
    }
    /* Timing Controller */
    TimingController( );
    /* Enable Power Save Mode */
    MCUCR |= _BV(SE);
    sei( );
    for( ;; )
        /* Sleep 0,038mA total supply current */
        asm( "sleep" );
}
/*=================================================================*/
void Init_AT90S2313( void )
{
    u08 i, *E2Addr;

    /* Watchdog Enable timeout after 2 second */
    wdt_enable( WDTO_2S );
    /* Analog Comparator DISABLE */
    ACSR = 0x80;
    /* PORTB as outputs & +5V pull-up */
    DDRB = 0x1F;
    PORTB = 0x1F;
    /* PORTD as inputs & without pull-up */
    DDRD = 0x13;
    /* Pointer BYTES to Parameters Array of INTEGERS */
    pParam = (u08 *)&Cnt_seconds;
    /* Relay Operation Modes */
    PORTD = 0x13;
    Relay_Status = FALSE;
    /* Pulses */
    if( Time_On != DISABLE ){
        Relay_Status = TRUE;
        PORTD = 0x03;
        PORTB ^= _BV(PORTB_MANUAL);
```

```
    }
    /* Power Save Sleep mode */
    MCUCR |= _BV(SM);
    /* Enable INT0 */
    GIMSK = 0x40;
    /* Enable Interrupts AT90S2313 */
    sei( );
    /* If the capacitor is not yet charged then
       initialeze the system (1st Time Power Up)*/
    /* 100K 5% (Resistor) - 10uF/16V (Capasitor)*/
    E2Addr = (u08 *)&EEpromMem[0];
    i = *(pParam+20);
    if( eeprom_read_byte(E2Addr) != i || ForcedReset == 1 ) {
        AvgCount = FALSE;
        ManMode = 0;
        ForcedReset = 0;
        for( i = 0; i < PARAMSIZE; i++ ){
            *(pParam+i) = 0;
            if( i >= 20){
                *(pParam+i) = eeprom_read_byte(E2Addr);
                ++E2Addr;
            }
        }
    }
}
/*=================================================================*/
void Communication( void )
{
    u08 Data, Address, E2Address;
    u08 Uart_Ch, i, *E2a;

    /* Enable UART system 38400,n,8,1 */
    /* Power ON ic MAX232 */
    PORTB ^= _BV(PORTB_RS232);
    /* UART Enable, Baudrate (12, 8MHz) : 38400,n,8,1 */
    UBRR = UART_38400;
    UCR |= (_BV(RXEN) | _BV(TXEN));
    /* Small Delay 10us */
    usDelay(10);
    /* Send 'Q' QUERY to see is a PC is connected */
    Uart_put(UART_QUERY);
    i = 0;
    /* Delay 1000 * 10us = 10.000us = 10ms */
    while( bit_is_clear( USR, RXC ) && i < 50 ){
        usDelay(10);
        ++i;
    }
    if( bit_is_set( USR, RXC ) ){
        /* if 'R' SEND_ALL command */
        Uart_Ch = Uart_get( );
        if( Uart_Ch == UART_READ ){
            Uart_put(UART_R_ACK);
            /* Send all Parameters to PC */
            for( i = 0; i < PARAMSIZE; i++ )
                Uart_put( *(pParam+i) );
        }
        /* if 'W' SEND_ALL command */
        else if( Uart_Ch == UART_WRITE ){
            /* Get Address, Low & High Bytes and Write to
             E2prom and to Param Array */
            Uart_put(UART_W_ACK);
            Address = Uart_get( );
            E2Address = Uart_get( );
            E2a = (u08 *)&EEpromMem[E2Address];
            for( i =0; i < 2; i++ ){
                Data = Uart_get( );
                if( Address >= 20 ){
```

```
                    eeprom_write_byte( E2a, Data );
                    ++E2a;
                }
                *(pParam+Address) = Data;
                ++Address;
            }
        }
    }
    /* Wait for all bytes to be SEND */
    loop_until_bit_is_set( USR, UDRE );
    loop_until_bit_is_set( USR, TXC );
    /* Small Delay 10us */
    usDelay(2000);
    /* Power OFF ic MAX232 */
    PORTB ^= _BV(PORTB_RS232);
    /* Disable UART */
    UCR = 0x00;
    /* Disable Tx Line else power consumption 14mA */
    DDRD = 0x13;
    PORTD |= 0x03;
}
/*=====================================================================*/
void Start_NE555( void )
{
    /* NE555 Trigger +5V */
    PORTB |= _BV(PORTB_TRIGGER);
    /* NE555 Power activation (ON) */
    PORTB &= (~_BV(PORTB_ENABLE));
    /*Setup Timer 1 Input Capture */
    TCCR1A = 0x00;
    TCCR1B = 0xC2;
    /* Enable interrupt Input Capture */
    TIMSK |= _BV(TICIE);
    /* NE555 Trigger +0V */
    PORTB &= (~_BV(PORTB_TRIGGER));
    /* Small delay 2us */
    usDelay( 2 );
    /* NE555 Trigger +5V */
    PORTB |= _BV(PORTB_TRIGGER);
}
/*=====================================================================*/
void usDelay( u16 tmdelay )
{
    /* Delay for us*/
    u16 tdly;
    tdly = tmdelay << 3;
    while( tdly != 0 )
        --tdly ;
}
/*=====================================================================*/
SIGNAL(SIG_INTERRUPT0 )
{
    if( AutoMode == DISABLE && (Time_On | Time_Off) == DISABLE ){
        ManMode = MAN_CHARGE_F;
        Load_Man_Settings( );
    }
}
/*=====================================================================*/
SIGNAL(SIG_INPUT_CAPTURE1)
{
    u16 Stop_ICP;

    if( bit_is_set( TCCR1B, ICES1 ) ){
        Start_ICP = (u16)ICR1;
        TCCR1B = 0x82;
    }
    else {
```

```
        Stop_ICP = (u16)ICR1;
        /* Stop Timer 1 */
        TCCR1B = 0x00;
        /* Disable interrupt Input Capture */
        TIMSK &= (~_BV(TICIE));
        if( Stop_ICP > Start_ICP )
            TimeNE555 = Stop_ICP - Start_ICP;
        else
            TimeNE555 = 0xFFFF - Start_ICP + Stop_ICP;
        /* Calculation Temperature */
        Calculation( );
        /* Check automatic setpoint */
        if( AutoMode == DISABLE && Setpoint_aut > AvgAct_Therm ){
            AutoMode = RUNNING;
            if( (OperationModes & 0x01) == 0x01 )
                Load_Man_Settings( );
            else
                Load_Auto_Settings( );
        }
        /* Delay for Capasitor Dischanging */
        usDelay(200);
        /* POWER OFF NE555 */
        PORTB |= _BV(PORTB_ENABLE);
        NTC_Status = FINISHED;
    }
}
/*=================================================================*/
void Calculation( void )
{
    u16 Up, Upper, Div, Scale;
    u08 i, Low, High, *E2Addr;

    Act_Therm = MAXIMUM_TH;
    E2Addr = (u08 *)&NTC_Curve[0];
    for( i = 0; i < NTC_DATA_LEN; i++ ){
        Low = eeprom_read_byte(E2Addr);
        ++E2Addr;
        High = eeprom_read_byte(E2Addr);
        ++E2Addr;
        Up = (High<<8) + Low;
        if( TimeNE555 > Up ){
            if( i == 0 )
                Act_Therm = MINIMUM_TH;
            else {
                E2Addr -= 4; /*(u08 *)&NTC_Curve[i-1]; */
                Low = eeprom_read_byte(E2Addr);
                ++E2Addr;
                High = eeprom_read_byte(E2Addr);
                Upper = (High<<8) + Low;
                Div = TimeNE555 - Up;
                Scale = Upper - Up;
                Act_Therm = ((i-6)*100) - ((Div*100)/Scale);
            }
            i = 70; /* Exit For */
        }
    }
    /* Average calculation */
    AvgSum += Act_Therm;
    if( AvgCount != TRUE ){
        AvgCount = TRUE;
        AvgSum = Act_Therm << AVG_SAMPLES;
    }
    AvgAct_Therm = AvgSum >> AVG_SAMPLES;
    AvgSum -= AvgAct_Therm;
}
/*=================================================================*/
u08 Uart_get( void )
```

```
{
    loop_until_bit_is_set( USR, RXC );
    return( UDR );
}
/*================================================================*/
void Uart_put( u08 Uart_data )
{
    loop_until_bit_is_set( USR, UDRE );
    UDR = Uart_data;
}
/*================================================================*/
void TimingController( void )
{
    /* Times for counting, room cold start relay */
    if( (Time_On | Time_Off) != DISABLE ){
        if( Time_On != DISABLE )
            --Time_On;
        else {
            if( Time_Off != DISABLE )
                --Time_Off;
        }
    }
    if( (Time_On | Time_Off) == DISABLE ){
        if( ManMode == MAN_DELAY_TIME || ManMode == MAN_CHARGE_R || ManMode ==
 MAN_CHARGE_F){
            ++ManMode;
            if( ManMode == MAN_CHARGE_R && (OperationModes & 0x02) == 0x02 )
                ++ManMode;
            Load_Man_Settings( );
        }
        else if( ManMode == MAN_OPEN_EV_R )
            ++ManMode;
        else if( ManMode == MAN_OPEN_EV_F )
            ManMode = DISABLE;
    }
    /* If Hysterisis reached stop relay */
    if( AutoMode == RUNNING ){
        ManMode = DISABLE;
        if( Setpoint_aut + Hysterisis_a <= AvgAct_Therm ){
            AutoMode = DISABLE;
            ++ManMode;
            Time_On = DISABLE;
            Time_Off = DISABLE;
        }
        else {
            if( (Time_On | Time_Off) == DISABLE )
                Load_Auto_Settings( );
        }
    }
    else {
        if( ManMode == MAN_WAIT ){
            /* Falling  */
            if( Setpoint_man < MAXIMUM_TH ){
                if( Setpoint_man >= AvgAct_Therm  ){

                    ++ManMode;
                    if( (OperationModes & 0x04) == 0x04 )
                        ++ManMode;
                    Load_Man_Settings( );
                }
            }
            else
                ManMode = MAN_OPEN_EV_F;

        }
        if( ManMode == AUTO_STOPPED ){
            /* Raising */
```

```
            if( Setpoint_manP < MAXIMUM_TH ){
                if( Setpoint_manP <= AvgAct_Therm ){

                    ++ManMode;
                    Time_On = 0;
                    Time_Off = Time_manP_off;
                }
            }
            else
                ManMode = MAN_OPEN_EV_R;
        }
    }
}
/*=================================================================*/
void Load_Auto_Settings( void )
{
    AutoMode = RUNNING;
    Time_On = Time_aut_on;
    Time_Off = Time_aut_off;
}
/*=================================================================*/
void Load_Man_Settings( void )
{
    Time_On = Time_man_on;
    Time_Off = Time_man_off;
}
/*=================================================================*/
```