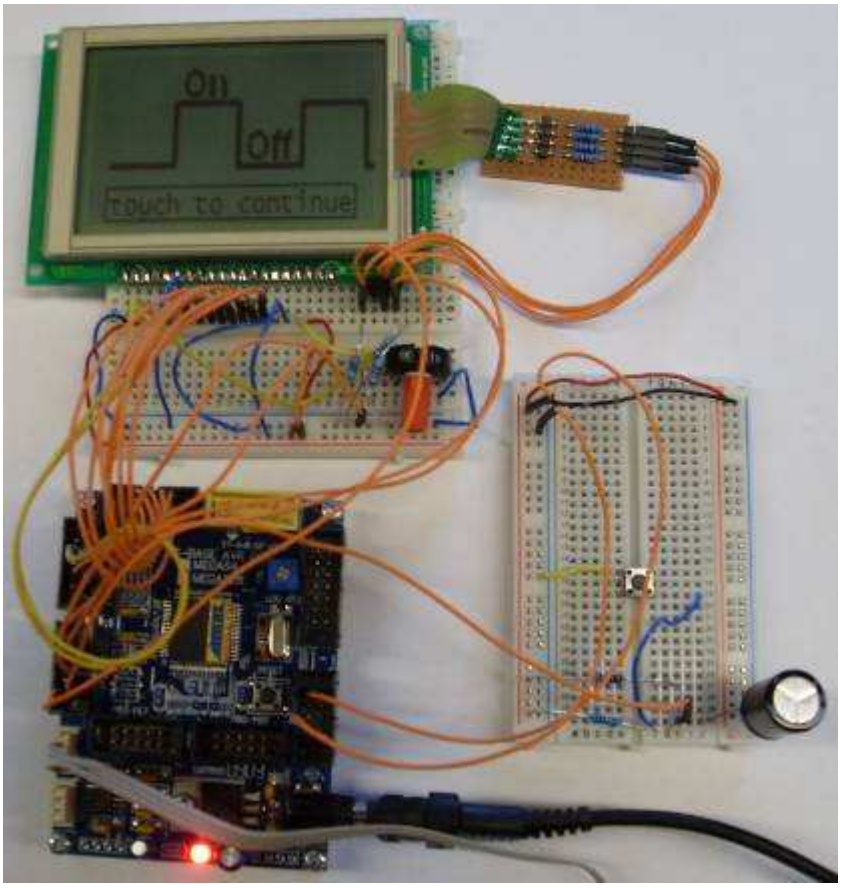
Names: Alexander Wegner, Yvonne Wöstefeld

Date: 19.01.2010

Class: 08_2_M_A

Microcontroller Technology

# Task: Programming an on/off controller



# Table of content:

# Task description

## Program requirements:

- Program an on/off controller with your AVR Evaboard
- On the start of the program let the user enter the on/off values and the time of checking using the serial port or an external connected keypad
- The on/off values must be entered in degrees Celsius (0 = 0V-99 = 5V) and the time of checking in ms(1-1000)
- Make it possible to correct typing mistakes
- Display the on/off state and the recent measured value on your LCD display
- Use the 10-bit AD result to calculate with, do not cut it to an 8-bit value
- Use inline documentation and the good style programming rules
- Do not use float/double or other fractured number variables

## Simulation/Measurements:

- Connect an 1st order RC network of your choice to your controller output pin and check, if your controller is working
- Draw/record a diagram with a program of your choice for a measurement where your system temperature goes up and is cycling between on and off state, then is disturbed and after this goes back to the normal cycling states

## Documentation:

- Make a small users manual for using your on/off controller
- Document your basic program operation by using one or more Nassi - Shneiderman or flow chart diagram(s)
- Document also, what ports/pins you are using in your program
- Include your measurements, the diagram and the used RC network description. Explain the parts of the measured diagram
- Include the source code in your documentation in a readable monospace font

## General principle of the on/off controller

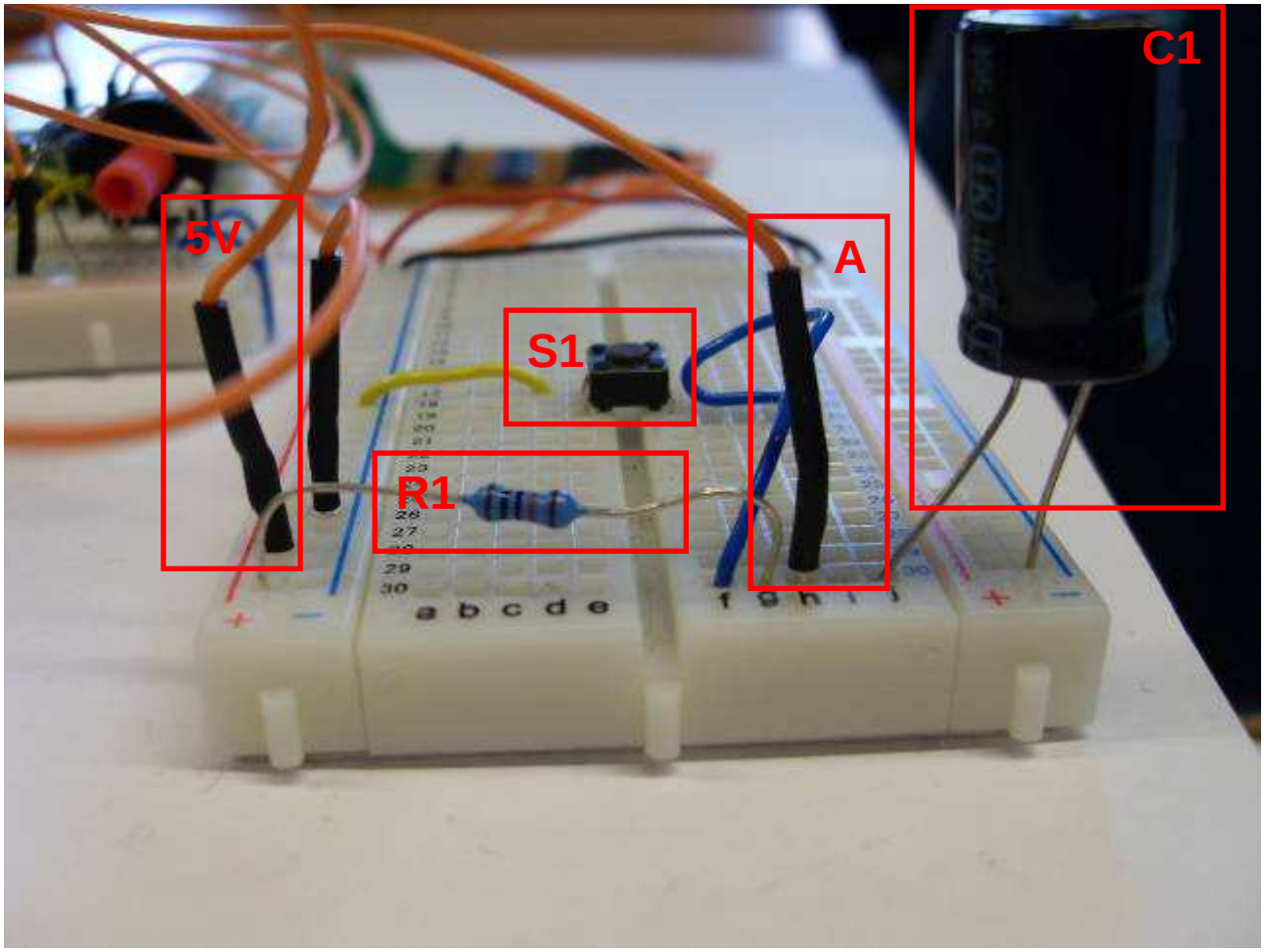
An on-off controller is the simplest form of a temperature control device. The output from the device is either on or off, with no middle state.

An on-off controller will switch the output only when the temperature crosses the set-point. For heating control, the output is on when the temperature is below the set-point, and off above set-point.

Since the temperature crosses the set-point to change the output state, the process temperature will be cycling continually, going from below set-point to above, and back below. In cases where this cycling occurs rapidly an on-off differential, or “hysteresis,” is added to the controller operations. This differential requires that the temperature exceed set-point by a certain amount before the output will turn off or on again. On-off differential prevents the output from “chattering” or making fast, continual switches if the cycling above and below the set-point occurs very rapidly.

On-off control is usually used where a precise control is not necessary, in systems which cannot handle having the energy turned on and off frequently, where the mass of the system is so great that temperatures change extremely slowly, or for a temperature alarm.

## Our principle of the on/off controller




5V are connected to the capacitor (on status).

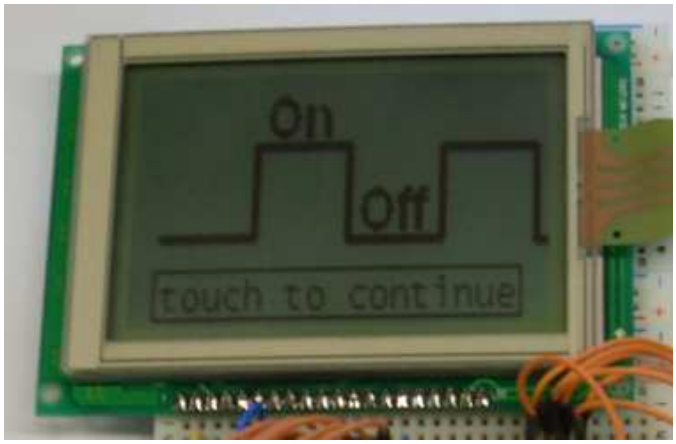
This capacitor loads up until it reached the maximum value, which you can decide (for example 4V).

If the capacitor reached this value, the 5V will turn off (0V = off status).

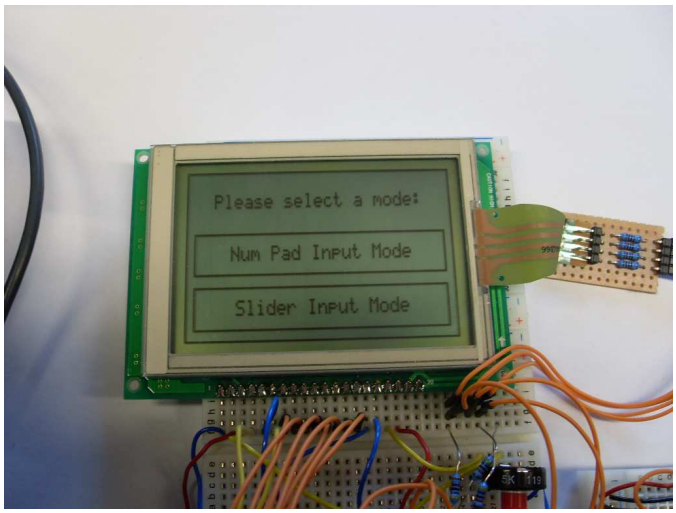
The capacitor will discharge until it reached the minimum value, which you can also decide (for example 2V).

# User Manual Controller

If you start the program, the start screen is shown. It looks like in the picture below. To touch on the display you need a pointed object (for example a pencil).




To go on you must click anywhere on the screen. After you did this a menu will turn up, where you can choose the way you want to type in your values for the on/off values and the time of checking.

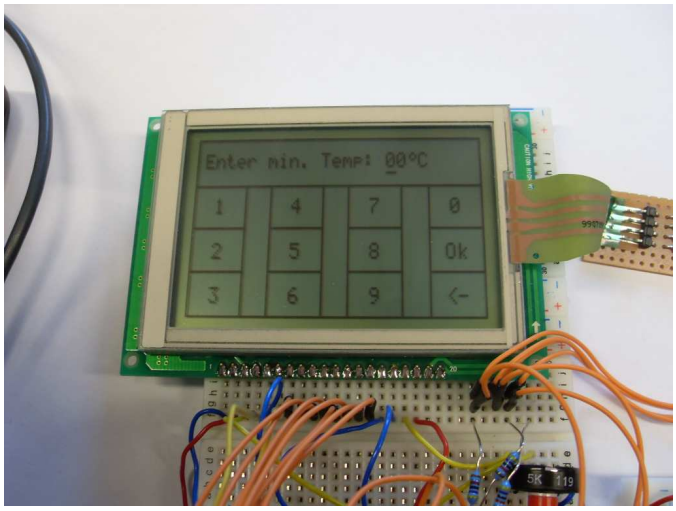



You can decide between two variations. You can type in the values via a “Num Pad Input Mode” or via a “Slider Input Mode”.

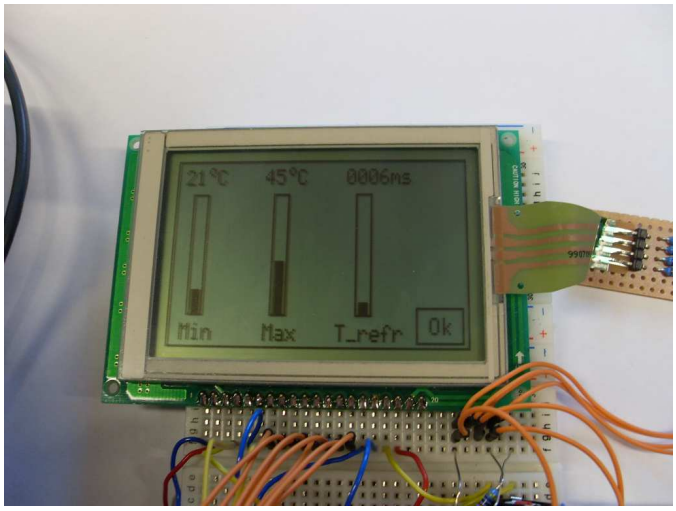
| | | |
|---|---|---|
| “Num Pad Input Mode” | → Min/Max value: | 0-99℃ in steps of 2℃ |
| | → Refresh Time: | 0-50ms in steps of 1ms |
| “Slider Input Mode”. | → Min/Max value: | 0-99℃ in steps of 1℃ |
| | → Refresh Time: | 0-1000ms in steps of 1ms |

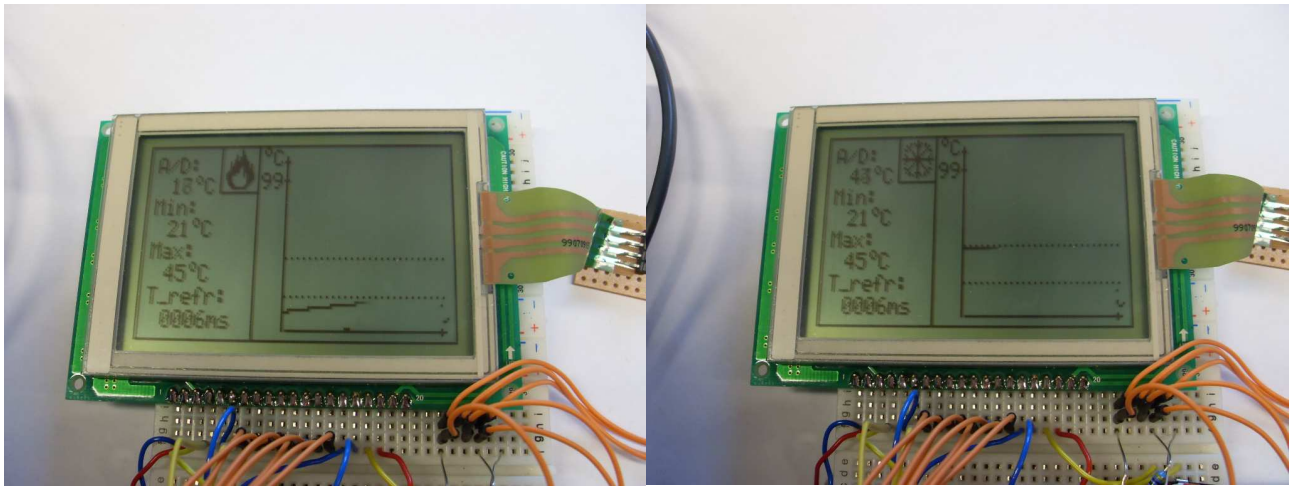
If you decide to type in the values via the “Num Pad Input Mode” the numeric keypad arise. There you can type in step by step your values for the minimum and maximum temperature and the checking time. If you click on “Ok” the transmission begins. If you click on the arrow you can correct your previous entry. An error comes up if the minimum temperature is higher than the maximum value or if the checking time is higher than 50ms.

Use the “Ok” button to go on to the next value to enter or the “<-“button to return to the previous value. Mistyping can be corrected. If all digits are field, the cursor returns to the first position to reenter the value.




If you decide to use the “Slider Input Mode” you can also choose the values for the on/off state and the checking time. You must slide over the LCD-display to change these values. The values are shown above the timber. If you click on “Ok” the transmission begins.



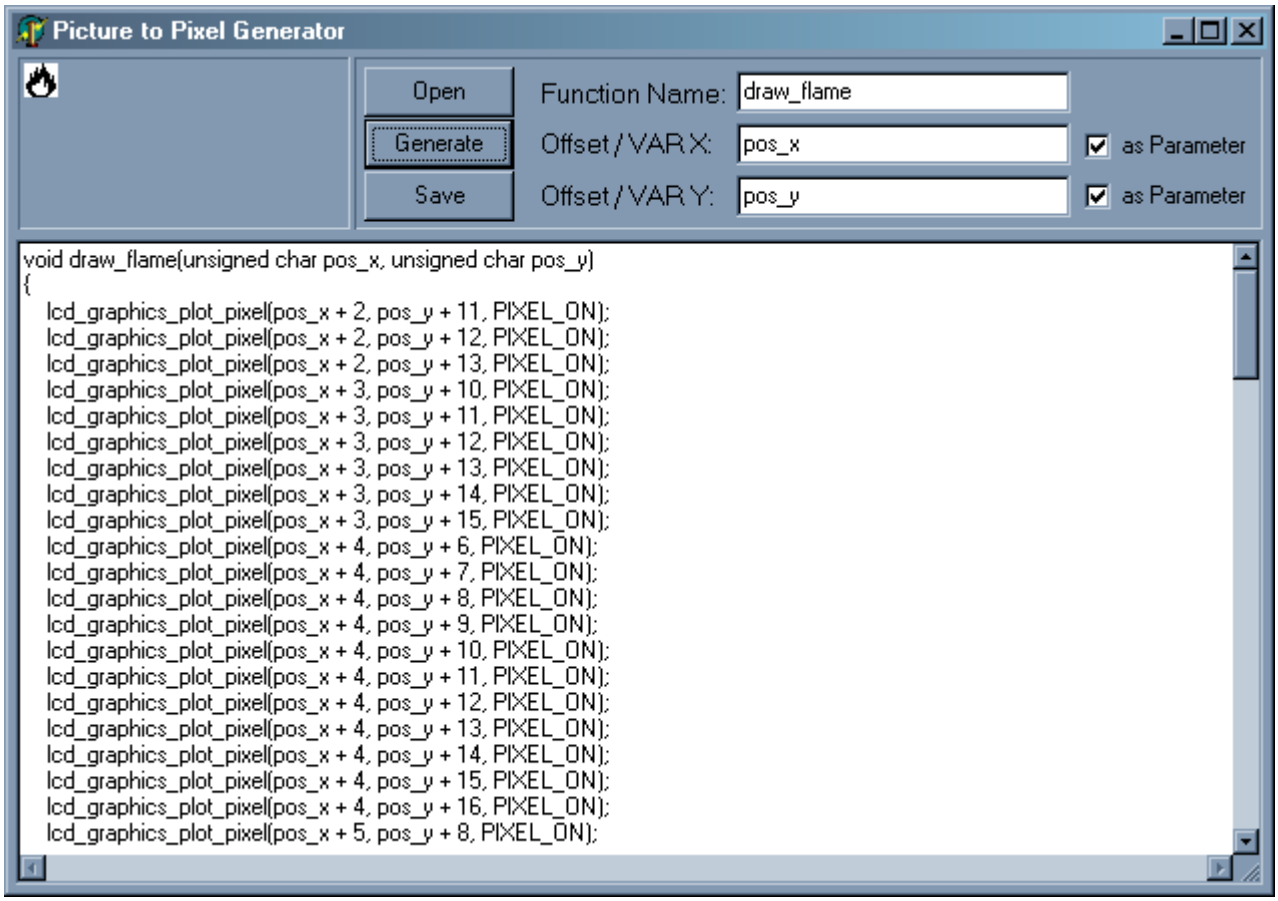



After your settings the transmission starts. If the controller heats up a flame is shown on the display. It means that the capacitor loads up (actual temperature is increasing) until it reached your maximum voltage (maximal temperature).

If the controller cools down a snowflake is shown on the display. It means that the capacitor discharge (actual temperature is decreasing) until it reached your minimum voltage (minimal temperature).

# User Manual Picture Converter

To draw a picture on the screen or to customize the cover screen you can use our Picture to Pixel converter tool.




First you need a picture with maximum size of 160x80 pixels and of the picture file format bmp. The pixels, which should be drawn on the screen, have to be colored black.

Start the program and open the picture with the “Open” button.

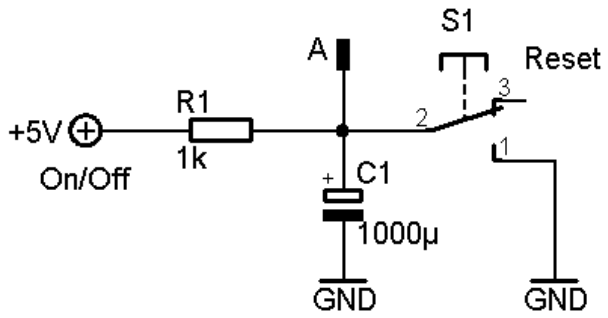
Choose a function name (c naming rules) and an offset/variable name (c naming rules) for the picture position on the screen.

You can use the variable as defines (for example #define pos_x 13) or as parameters to call the function.

Click on “Generate” to create your function. You can copy the code into your main.c or use the “Save” button to save the function into an individual c file and call it from your main.c.

## Simulation & Measurements

### R-C Network 1st Order




- R1 = 10000Ω
- C1 = 1000µF
- S1 Reset button for rapid discharge of C1
- Measurement point A for µC analog input
- +5V digital output of µC
- GND of µC

The R-C network is simulating a thermal system. We can simulate for example room temperature control or a water heater for a bathroom. R1 limits the current which is used to charge C1. The time constant tau represents the time needed to reach 63.2% of the supplied voltage across C1.

$$\tau = R \cdot C$$
$$\tau = 10000\Omega \cdot 1000\mu F$$
$$\tau = 10s$$
$$5 \cdot \tau \rightarrow 99.9\% charged$$

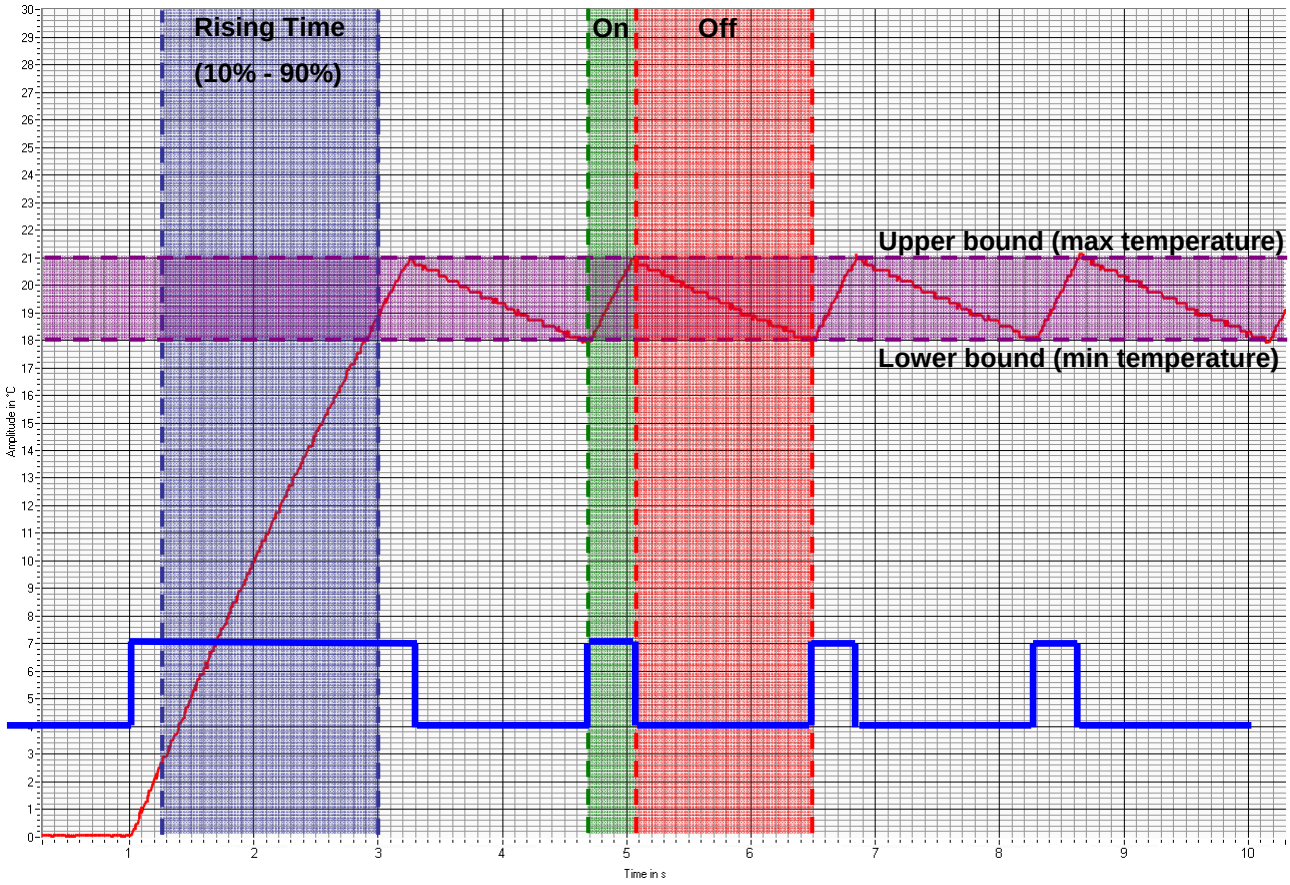
The charging and discharging process is observed by a microcontroller (analog measurement of the voltage across C1). If the measured value (actual temperature) is smaller or equal to the minimum voltage level (minimum temperature) the microcontroller supplies the system with 5V and the capacitor starts charging (temperature is increasing). If the measured value (actual temperature) is bigger or equal to the maximal voltage level (maximal temperature) the microcontroller is setting the supply pin to 0V and the capacitor starts discharging (temperature is decreasing).

## LABView Measurements

The charging and discharging of C1 was captured with LABView.

***Setup for room temperature control:***

- R1 = 10000Ω, C1 = 1000µF
- Minimal temperature: 18℃
- Maximal temperature: 21℃
- Refresh time: ~0ms




Picture 1: Controlling behavior

Rising time $T_r$: 3.00s - 1.25s = 1.75s

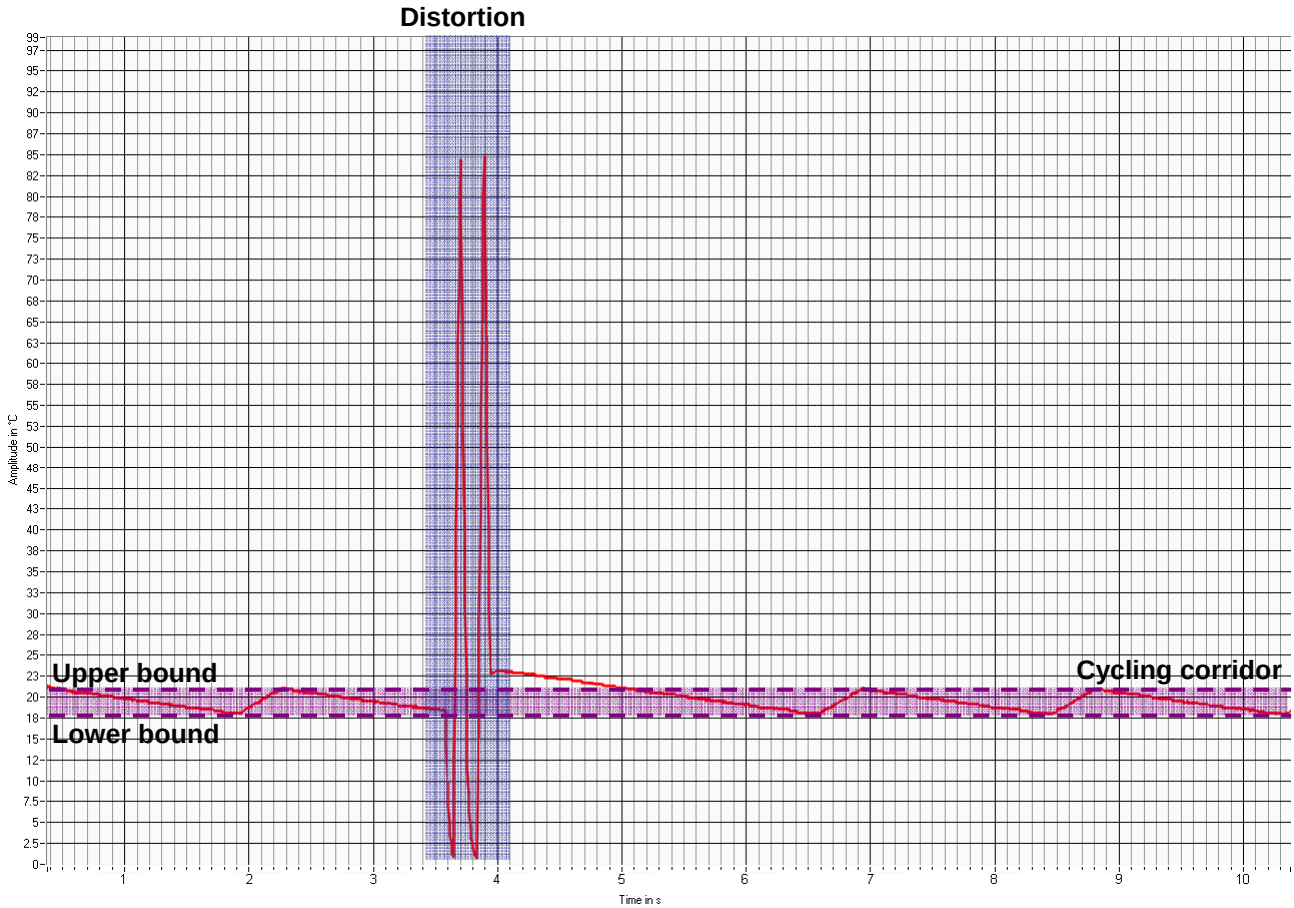
On time $T_{on}$: 5.06s - 4.68s = 0.38s

Off time $T_{off}$: 6.50s - 5.06s = 1.44s

Duty Cycle: $g = T_{on} / T_{on} + T_{off}$ = 0.38s / 0.38s + 1.44s = 0.21 = 21%

***Distortion of the system:***

The same values and system were used to generate with LABView the distortion behavior of the system.




Picture 2: Distortion of System

After the system was disturbed, the microcontroller adjusts the system back to the given parameters.

## Ports & Pins

| Port F (a/d converter, touch control) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Pin 7 | Pin 6 | Pin 5 | Pin 4 | Pin 3 | Pin 2 | Pin 1 | Pin 0 |
| x | x | x | x | x | A/D (x) + Digital Output (y) | A/D (y) + Digital output (x) | A/D (analogue input) |

| Port C (LCD data bus) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Pin 7 | Pin 6 | Pin 5 | Pin 4 | Pin 3 | Pin 2 | Pin 1 | Pin 0 |
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

| Port A (LCD control bus, touch control) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Pin 7 | Pin 6 | Pin 5 | Pin 4 | Pin 3 | Pin 2 | Pin 1 | Pin 0 |
| RS | Enable | R/W | x | x | x | x | Touch |

| Port D (system supply) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Pin 7 | Pin 6 | Pin 5 | Pin 4 | Pin 3 | Pin 2 | Pin 1 | Pin 0 |
| x | x | x | x | x | x | x | 5V/0V |

GND → connected to GND of microcontroller

5V → connected to 5V output of microcontroller

## Define Settings

***Show flame and flake for heating/cooling status***

```
#define FLAME_AND_FLAKE_ON
```

***Show grad sign in front of Celsius***

```
#define GRAD_ON
```

***Show cover screen***

```
#define COVER_ON
```

***Show current temperature, refresh automatically***

```
#define SHOW_AD_VALUE
```

***Live update of the graph***

```
#define LIVE_DIAGRAM_ON
```

# Customize to Other Microcontroller Boards

Using the define statements on top of the main.c it's easily possible to customize the used ports to other microcontroller boards using the same chip (like the EVA board).
Change the define statements to your specific controller setup.

## Hardware Setup

The display data port is initially set to port C. Change it for example to port B: DDR**B**, PORT**B**, PIN**B**.

```
// Display Data Port
          #define LCD_DATA_DDR                DDRC
          #define LCD_DATA_PORT               PORTC
          #define LCD_DATA_PIN                PINC
```

The display control port is initially set to port A. Change it for example to port B: DDR**B**, PORT**B**.

```
// Display Control Port
          #define LCD_CTRL_DDR                DDRA
          #define LCD_CTRL_PORT               PORTA
```

Change the display control pins to your setup.

```
// Display Control Pins
          #define LCD_CTRL_RS                 7
          #define LCD_CTRL_RW                 5
          #define LCD_CTRL_E                  6
```

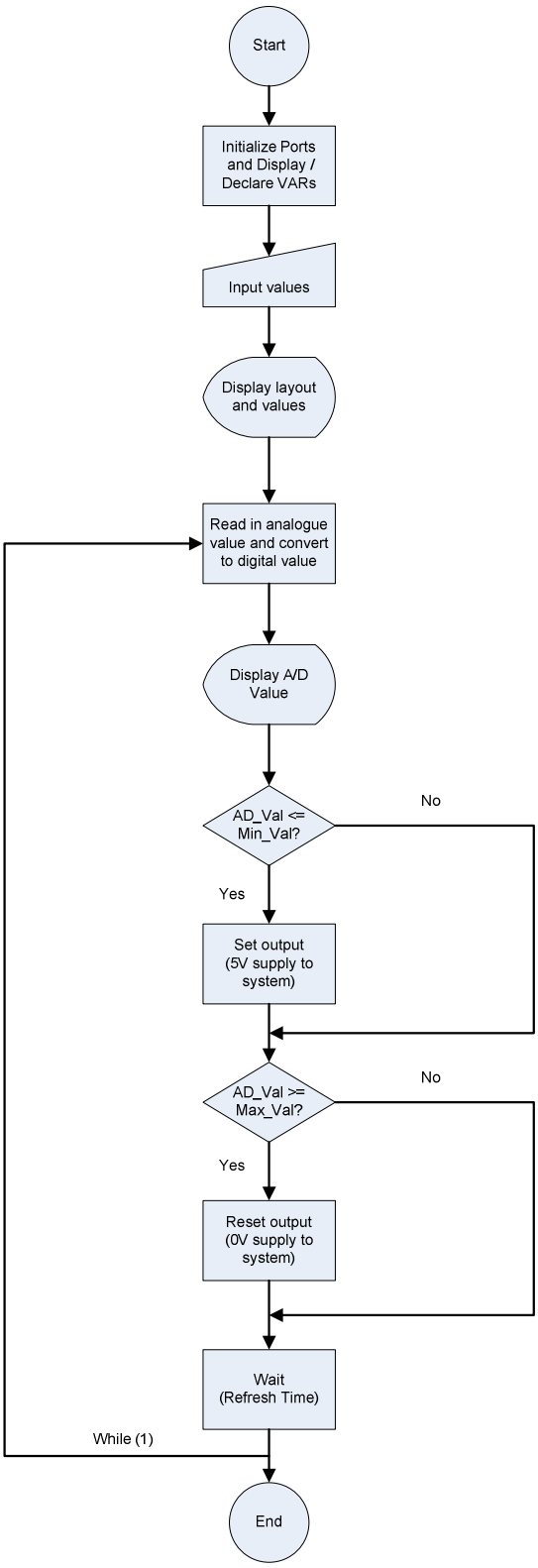
Change the digital output pin for the touch panel to your setup. For example **'b'** (lower case letter with '')
Change the analogue port to your chosen setup.

```
// Touch Control
          #define TOUCH_DIGIT_OUT_PORT        'a'
          #define TOUCH_DIGIT_OUT_PIN         0
          #define TOUCH_ANA_PORT              'f'
```

The r-c network is supplied with port d, pin 0. Change it to your setup.

```
// System Control
          #define SYS_SUPPLY_PORT             'd'
          #define SYS_SUPPLY_PIN              0
```

## Flowchart



## Source Code - main.c

```
//=====================================================
//=============   Hardware Setup        ================
//=====================================================
// Display Data Port
#define LCD_DATA_DDR DDRC
#define LCD_DATA_PORT PORTC
#define LCD_DATA_PIN PINC
// Display Control Port
#define LCD_CTRL_DDR DDRA
#define LCD_CTRL_PORT PORTA
// Display Control Pins
#define LCD_CTRL_RS 7
#define LCD_CTRL_RW 5
#define LCD_CTRL_E 6
// Display Size
#define LCD_WIDTH 160
#define LCD_HEIGHT 80
// Touch Control
#define TOUCH_DIGIT_OUT_PORT 'a'
#define TOUCH_DIGIT_OUT_PIN 0
#define TOUCH_ANA_PORT 'f'
// System Control
#define SYS_SUPPLY_PORT 'd'
#define SYS_SUPPLY_PIN 0
//=====================================================
//=============   Program Setup         ================
//=====================================================
#define PIC_POS_FLAME_FLAKE_X 40  //origin pos x for flame/flake picture
#define PIC_POS_FLAME_FLAKE_Y 1   //origin pos y for flame/flake picture
#define AD_PIN 0                  //connect to ad0

#define FLAME_AND_FLAKE_ON        //show flame/flake for heating/cooling process
#define GRAD_ON                   //show ° sign in front of Celsius
#define COVER_ON                  //show cover screen
#define SHOW_AD_VALUE             //automatically update of current temperature
#define LIVE_DIAGRAM_ON         //constantly refreshing the graph in the diagram

#include "lib/io_fct.c"           //library for input/output functions
#include "lib/LC7981.c"           //library for gfx LCD functions
#include "lib/delay.c"            //library for delay function
#include <avr/signal.h>           //library for interrupt functions
#include <avr/interrupt.h>        //library for interrupt functions
#include <math.h>                 //library for mathematic functions
#include "lib/touch_pictures.c"   //library for cover picture function

void clr_diagram(void);           //clear diagram array
void draw_diagram(void);          //draw the graph on the display
void draw_line(unsigned char x_start, unsigned char y_start, unsigned char
x_end, unsigned char y_end, unsigned char draw); //connect to points with a
line
unsigned char get_touch_y(void);  //returns y coordinate of touch point in pixel
unsigned int get_touch_x(unsigned int y); //returns x coordinate of touch point
in pixel
void call_input_mode_screen(void); //calls the input mode selection screen
void call_num_pad_mode(void);      //calls the num pad mode for input
void call_slider_mode(void);       //calls the slider mode for input
void draw_layout(void);            //draws the layout of the controller screen

unsigned char diagram[80];         //array for measured points
```

```
unsigned char val_ad_bcd[3] = "00";    //ad value string
unsigned char val_min_bcd[3] = "00";    //min value string
unsigned char val_max_bcd[3] = "00";    //max value string
unsigned char val_refr_bcd[7] = "0000ms";  //refresh time string
unsigned int val_ad = 0;        //ad value decimal
unsigned int val_min = 0;       //min value decimal
unsigned int val_max = 0;       //max value decimal
unsigned int val_refr = 0;       //refresh time decimal
//=============  VAR generell  ================
unsigned int buffer_touch_ad_x = 0;    //buffer for ad values x
unsigned char buffer_touch_ad_y = 0;    //buffer for ad values y
unsigned char selected_mode = 0;    //input mode - 2 slider, 1 num pad
//=============  VAR Slider Input ================
unsigned char bar_1_height = 0;    //height bar 1 - min value
unsigned char val_min_bar = 0;     //min value
unsigned char bar_2_height = 0;    //height bar 2 - max value
unsigned char val_max_bar = 0;     //max value
unsigned char bar_3_height = 0;    //height bar 3 - refresh time
unsigned char val_refr_bar = 0;    //refresh time
unsigned char button_pressed = 0;    //button pressed - 1 yes, 0 no - also used
for num pad
//=============  VAR Num Pad Input ================
unsigned char num_pad_pressed[12] = {0,0,0,0,0,0,0,0,0,0,0,0};//num pad buttons
control
unsigned char num_input_min_temp[3] = "00";//input string min value
unsigned char num_input_max_temp[3] = "00";//input string max value
unsigned char num_input_refr_time[7] = "0000ms"; //input string refresh time
unsigned char num_pad_value_control = 0;  //input for diff. values - 0 min, 1
max, 2, refr time, 3 error check
unsigned char num_pad_input_recognised = 0;//help var for num pad input
detection
unsigned char num_pad_cursor_pos = 0;   //cursor position for input
unsigned char i_count = 0;     //help var for num pad input detection

SIGNAL (SIG_ADC)         //a/d converter in interrupt mode
{
 val_ad = ADCL;
 val_ad |= (ADCH<<8);
}

int main(void)          //main program
{
//=============  VAR generell  ================
 unsigned char x_pix, y_pix, i = 0;  //x & y position of measured point,
counter var
 unsigned char buffer_x_pix = 0;    //buffer for x_pix
 unsigned char buffer_y_pix = 0;   //buffer for y_pix
//=============  Flake and Flame  ================
#ifdef FLAME_AND_FLAKE_ON
 unsigned char flake_drawn = 0;    //flake was drawn = 1, not = 0
 unsigned char flame_drawn = 0;    //flame was drawn = 1, not = 0
#endif
//=============   Init LCD  ================
 lcd_graphics_init();       //init LCD
 lcd_graphics_clear();       //clear LCD
 init_out_pin(TOUCH_DIGIT_OUT_PORT, TOUCH_DIGIT_OUT_PIN);       //porta - pin0
as output for touchscreen
 init_out_pin(SYS_SUPPLY_PORT, SYS_SUPPLY_PIN); //portd - pin0 as output for
system supply (r-c)
```

```
 reset_pin(SYS_SUPPLY_PORT, SYS_SUPPLY_PIN); //reset power supply of system (r-
c)
#ifdef COVER_ON
//=============   Cover Picture================
 draw_pic();           //draw cover picture and wait for touch input
 while (!((buffer_touch_ad_x >= 1) && (buffer_touch_ad_x <= 160)))
 {
  buffer_touch_ad_y = get_touch_y();
  buffer_touch_ad_x = get_touch_x(buffer_touch_ad_y);
 }
 lcd_graphics_clear();       //clear display
#endif
//============= Input Mode Selection ================
 call_input_mode_screen();     //input selection screens appears: 1 num pad
mode, 2 slider mode
 if (selected_mode == 1)
  call_num_pad_mode();     //num pad screen appears
 if (selected_mode == 2)
  call_slider_mode();      //slider screen appears
//=============  Controller Screen ================
 draw_layout();          //draw controller layout
//=============  Init a/d converter ================
 sei();             //global interrupt enable on
 ADMUX = 0x40|AD_PIN;       //connect to AD_PIN
 ADCSRA = 0x9e;          //enable ad converter
 ADCSRA |= 0x60;           //start converter
//=============   Program   ================
 x_pix = 0;           //pixel x position
 y_pix = 0;           //pixel y position
 clr_diagram();          //clear data array
 init_out_pin(TOUCH_DIGIT_OUT_PORT, TOUCH_DIGIT_OUT_PIN);   //touch control pin

 while (1)
 {
  if (x_pix <= 79)
  {
   diagram[x_pix] = y_pix;
#ifdef LIVE_DIAGRAM_ON
   draw_line(buffer_x_pix + 75, (-1*(signed int)buffer_y_pix) + 74, x_pix + 75,
(-1*(signed int)y_pix) + 74, 1);
#endif
   y_pix = round(((long int)val_ad*118*489) / 1000000);
   buffer_x_pix = x_pix;
   buffer_y_pix = y_pix;
   x_pix++;
  }
  if (x_pix == 80)
  {
   //draw_diagram();
#ifdef LIVE_DIAGRAM_ON
   clr_diagram();
  //=============   Min/Max Level================
   for (i = 73; i <= 154; i++)
   {
    if (!(i%3))
    {
     lcd_graphics_plot_pixel(i, 14+round(60 - bar_2_height * 118 / 100),
PIXEL_ON);
     lcd_graphics_plot_pixel(i, 14+round(60 - bar_1_height * 118 / 100),
```

```
PIXEL_ON);
      }
    }
    x_pix = 0;
    buffer_x_pix = 0;
#endif
#ifndef LIVE_DIAGRAM_ON
    draw_diagram();
#endif
   }
   if (val_ad <= val_min)          //current temperature <= min level
   {
    set_pin(SYS_SUPPLY_PORT, SYS_SUPPLY_PIN);  //feeding system with heat
(current on)
#ifdef FLAME_AND_FLAKE_ON
    if (!(flame_drawn))        //draw flame, if not already drawn
    {
     draw_flame();
     flame_drawn = 1;
     flake_drawn = 0;
    }
#endif
   }
   if ((val_ad >= val_max))         //current temperature >= max level
   {
    reset_pin(SYS_SUPPLY_PORT, SYS_SUPPLY_PIN);//turn off heating, cooling
(current of)
#ifdef FLAME_AND_FLAKE_ON
    if (!(flake_drawn))        //draw flake, if not already drawn
    {
     draw_flake();
     flake_drawn = 1;
     flame_drawn = 0;
    }
#endif
   }
#ifdef SHOW_AD_VALUE
   val_ad_bcd[0] = ((round(((long int)val_ad * 968) / 10000)) / 10) + 48;
 //integer to bcd
   val_ad_bcd[1] = ((int)(round(((long int)val_ad * 968) / 10000))%10) + 48;
 //integer to bcd
   g_draw_string(13, 14, val_ad_bcd);  //draw current temperature on screen
#endif
   delay_ms(val_refr);       //wait for the given time
  }
}

void call_input_mode_screen(void)
{
//=======================================================
//=============   Input Mode            ================
//=======================================================
 g_draw_rectangle(0, 0, 160, 80);    //frame
 g_draw_string(17, 11, "Please select a mode:");
//=============   num pad mode ================
 g_draw_rectangle(5, 30, 150, 20);    //frame
 g_draw_string(27, 36, "Num Pad Input Mode");
//=============   slider mode  ================
 g_draw_rectangle(5, 55, 150, 20);    //frame
 g_draw_string(30, 61, "Slider Input Mode");
```

```
 delay_ms(1000);          //wait for touch refresh
 buffer_touch_ad_x = 0;
 buffer_touch_ad_y = 0;
 button_pressed = 0;

 while (button_pressed <= 3)     //wait for user choice (long press on touch)
 {
  buffer_touch_ad_y = get_touch_y();
  buffer_touch_ad_x = get_touch_x(buffer_touch_ad_y);
  if (buffer_touch_ad_x > 160)
   buffer_touch_ad_x = 160;
  if (buffer_touch_ad_x == 0)
   buffer_touch_ad_y = 0;
//=============   num pad mode ================
  if ((buffer_touch_ad_y >= 30) && (buffer_touch_ad_y <= 50) &&
(buffer_touch_ad_x >= 2) && (buffer_touch_ad_x <= 152))
  {
   selected_mode = 1;
   button_pressed++;
  }
  else
  {
//=============   slider mode  ================
   if ((buffer_touch_ad_y >= 55) && (buffer_touch_ad_y <= 80) &&
(buffer_touch_ad_x >= 2) && (buffer_touch_ad_x <= 152))
   {
    selected_mode = 2;
    button_pressed++;
   }
   else
    button_pressed = 0;
  }
 }
}

void call_num_pad_mode(void)
{
//=========================================================
//=============   Num Pad mode          ================
//=========================================================
 unsigned char i;         //counting var
//=============   Layout    ================
 lcd_graphics_clear();
 g_draw_rectangle(0, 0, 160, 80);
 g_draw_rectangle(0, 0, 160, 23);

 g_draw_rectangle(0, 22, 30, 20);
 g_draw_string(12, 28, "1");
 g_draw_rectangle(0, 41, 30, 20);
 g_draw_string(12, 48, "2");
 g_draw_rectangle(0, 60, 30, 20);
 g_draw_string(12, 67, "3");

 g_draw_rectangle(43, 22, 30, 20);
 g_draw_string(55, 28, "4");
 g_draw_rectangle(43, 41, 30, 20);
 g_draw_string(55, 48, "5");
 g_draw_rectangle(43, 60, 30, 20);
```

```
 g_draw_string(55, 67, "6");

 g_draw_rectangle(86, 22, 30, 20);
 g_draw_string(98, 28, "7");
 g_draw_rectangle(86, 41, 30, 20);
 g_draw_string(98, 48, "8");
 g_draw_rectangle(86, 60, 30, 20);
 g_draw_string(98, 67, "9");

 g_draw_rectangle(130, 22, 30, 20);
 g_draw_string(142, 28, "0");
 g_draw_rectangle(130, 41, 30, 20);
 g_draw_string(139, 48, "Ok");
 g_draw_rectangle(130, 60, 30, 20);
 g_draw_string(139, 67, "<-");
//=============   /layout   ================
 num_pad_cursor_pos = 0;         //position of cursor at start

 while(num_pad_value_control != 4)    //wait for complete input
 {
  buffer_touch_ad_y = get_touch_y();
  buffer_touch_ad_x = get_touch_x(buffer_touch_ad_y);
  if (buffer_touch_ad_x > 160)
   buffer_touch_ad_x = 160;

  if (buffer_touch_ad_x == 0)
   buffer_touch_ad_y = 0;

//============= Buttons pressed check================
  if ((buffer_touch_ad_y >= 20) && (buffer_touch_ad_y <= 40) &&
(buffer_touch_ad_x >= 30) && (buffer_touch_ad_x <= 50))
   num_pad_pressed[1]++;
  else
   num_pad_pressed[1] = 0;
   //g_draw_string(100, 2, "1");
  if ((buffer_touch_ad_y >= 41) && (buffer_touch_ad_y <= 61) &&
(buffer_touch_ad_x >= 30) && (buffer_touch_ad_x <= 50))
   num_pad_pressed[2]++;
  else
   num_pad_pressed[2] = 0;
   //g_draw_string(100, 2, "2");
  if ((buffer_touch_ad_y >= 62) && (buffer_touch_ad_y <= 80) &&
(buffer_touch_ad_x >= 30) && (buffer_touch_ad_x <= 60))
   num_pad_pressed[3]++;
  else
   num_pad_pressed[3] = 0;
   //g_draw_string(100, 2, "3");
  if ((buffer_touch_ad_y >= 20) && (buffer_touch_ad_y <= 40) &&
(buffer_touch_ad_x >= 60) && (buffer_touch_ad_x <= 80))
   num_pad_pressed[4]++;
  else
   num_pad_pressed[4] = 0;
   //g_draw_string(100, 2, "4");
  if ((buffer_touch_ad_y >= 41) && (buffer_touch_ad_y <= 61) &&
(buffer_touch_ad_x >= 60) && (buffer_touch_ad_x <= 80))
   num_pad_pressed[5]++;
  else
   num_pad_pressed[5] = 0;
```

```
    //g_draw_string(100, 2, "5");
   if ((buffer_touch_ad_y >= 62) && (buffer_touch_ad_y <= 80) &&
(buffer_touch_ad_x >= 80) && (buffer_touch_ad_x <= 100))
    num_pad_pressed[6]++;
   else
    num_pad_pressed[6] = 0;
    //g_draw_string(100, 2, "6");

   if ((buffer_touch_ad_y >= 20) && (buffer_touch_ad_y <= 40) &&
(buffer_touch_ad_x >= 90) && (buffer_touch_ad_x <= 130))
    num_pad_pressed[7]++;
   else
    num_pad_pressed[7] = 0;
    //g_draw_string(100, 2, "7");
   if ((buffer_touch_ad_y >= 41) && (buffer_touch_ad_y <= 61) &&
(buffer_touch_ad_x >= 90) && (buffer_touch_ad_x <= 130))
    num_pad_pressed[8]++;
   else
    num_pad_pressed[8] = 0;
    //g_draw_string(100, 2, "8");
   if ((buffer_touch_ad_y >= 62) && (buffer_touch_ad_y <= 80) &&
(buffer_touch_ad_x >= 110) && (buffer_touch_ad_x <= 130))
    num_pad_pressed[9]++;
   else
    num_pad_pressed[9] = 0;
    //g_draw_string(100, 2, "9");

   if ((buffer_touch_ad_y >= 20) && (buffer_touch_ad_y <= 40) &&
(buffer_touch_ad_x >= 140) && (buffer_touch_ad_x <= 160))
    num_pad_pressed[0]++;
   else
    num_pad_pressed[0] = 0;
    //g_draw_string(100, 2, "0");
   if ((buffer_touch_ad_y >= 41) && (buffer_touch_ad_y <= 61) &&
(buffer_touch_ad_x >= 140) && (buffer_touch_ad_x <= 160))
    num_pad_pressed[10]++;
   else
    num_pad_pressed[10] = 0;
    //g_draw_string(100, 2, "r");
   if ((buffer_touch_ad_y >= 62) && (buffer_touch_ad_y <= 80) &&
(buffer_touch_ad_x >= 140) && (buffer_touch_ad_x <= 160))
    num_pad_pressed[11]++;
   else
    num_pad_pressed[11] = 0;
    //g_draw_string(100, 2, "b");
//============== /Buttons pressed check================
//============== Display diff msgs     ================
   switch (num_pad_value_control)
   {
    case 0://min temp input
     g_draw_string(5, 7, "Enter min. Temp: ");
     g_draw_string(107, 7, num_input_min_temp);
     g_draw_string(125, 7, "C");
#ifdef GRAD_ON
     draw_grad(119, 6);             //draw ° sign in front of C
#endif
     break;
    case 1://max temp input
     g_draw_string(5, 7, "Enter max. Temp: ");
```

```
      g_draw_string(107, 7, num_input_max_temp);
      g_draw_string(125, 7, "C");
#ifdef GRAD_ON
      draw_grad(119, 6);              //draw ° sign in front of C
#endif
      break;
     case 2://refresh time input
      g_draw_string(5, 7, "Enter ref. Time: ");
      g_draw_string(107, 7, num_input_refr_time);
      break;
     case 3:
      val_min = (num_input_min_temp[0] - 48)*10;      //bcd to integer
      val_min += (num_input_min_temp[1] - 48)*1;      //bcd to integer

      val_max = (num_input_max_temp[0] - 48)*10;      //bcd to integer
      val_max += (num_input_max_temp[1] - 48)*1;      //bcd to integer

      if (val_max < val_min)            //error check, max temp should be >= min
temp
      {
       g_draw_string(5, 7, "Error: Min > Max Temp!  ");  //show error msg
       delay_ms(3000);              //short wait for reading error msg
       num_pad_value_control = 0;         //reset input and start with min temp

      }
      else
      {
       val_refr = (num_input_refr_time[0] - 48)*1000;   //bcd to integer
       val_refr += (num_input_refr_time[1] - 48)*100;   //bcd to integer
       val_refr += (num_input_refr_time[2] - 48)*10;   //bcd to integer
       val_refr += (num_input_refr_time[3] - 48)*1;   //bcd to integer
       if (val_refr > 1000)          //error check, max refresh time = 1000ms
       {
        g_draw_string(5, 7, "Error: Ref Time > 1000! "); //show error msg
        delay_ms(3000);              //short wait for reading error msg
        num_pad_value_control = 2;         //reset input and start with refresh
time
       }
       else
       {
        bar_1_height = (val_min + 1) / 2;      //later used for level line in
diagram
        bar_2_height = (val_max + 1) / 2;      //later used for level line in
diagram
        num_pad_value_control = 4;         //leaving while loop
        val_min = round(((long int)val_min*10000)/968); //converts grad Celsius
(0-99) to 10bit
        val_max = round(((long int)val_max*10000)/968); //converts grad Celsius
(0-99) to 10bit
        for (i = 0; i <= 3; i++)
        {
         val_refr_bcd[i] = num_input_refr_time[i];  //new string for displaying
in main screen
         val_min_bcd[i] = num_input_min_temp[i];  //new string for displaying in
main screen
         val_max_bcd[i] = num_input_max_temp[i];  //new string for displaying in
main screen
        }
       }
      }
```

```
      break;
    }
//============= debouncing & press detection ================
    i_count = 0;
    for (i = 0; i <= 11; i++)
    {
     if (num_pad_pressed[i] >= 2)
     {
      i_count++;
     }
     if ((num_pad_pressed[i] >= 2) && !(num_pad_input_recognised))
     {
      num_pad_input_recognised = 1;
      if (i == 10)   //ok button
      {
       num_pad_value_control++;
       num_pad_cursor_pos = 0;
      }
      else if (i == 11) //<- button
      {
       if (num_pad_value_control != 0)
       {
        num_pad_value_control--;
        g_draw_string(107, 7, "      ");
       }
       num_pad_cursor_pos = 0;
      }
      else
      {
       switch (num_pad_value_control)
       {
        case 0:
         num_input_min_temp[num_pad_cursor_pos] = i+48;
         num_pad_cursor_pos++;
         if (num_pad_cursor_pos == 2)
          num_pad_cursor_pos = 0;
         break;
        case 1:
         num_input_max_temp[num_pad_cursor_pos] = i+48;
         num_pad_cursor_pos++;
         if (num_pad_cursor_pos == 2)
          num_pad_cursor_pos = 0;
         break;
        case 2:
         num_input_refr_time[num_pad_cursor_pos] = i+48;
         num_pad_cursor_pos++;
         if (num_pad_cursor_pos == 4)
          num_pad_cursor_pos = 0;
         break;
       }
      }
     }
    }

    if (i_count == 0)
     num_pad_input_recognised = 0;

    for (i = 0; i <= 40; i++)            //clear cursor line
    {
     lcd_graphics_plot_pixel(105+i, 16, PIXEL_OFF);
```

```
   }
   g_draw_horizontal_line(107 + (num_pad_cursor_pos*6), 16, 5);  //drawing
cursor at cursor position
  }
}

void call_slider_mode(void)
{
//=========================================================
//=============   Slider mode           ================
//=========================================================
 unsigned char i;
//=============   Layout    ================
 lcd_graphics_clear();
 g_draw_rectangle(0, 0, 160, 80);
 g_draw_string(7, 70, "Min");
 g_draw_rectangle(10, 15, 9, 52);
 g_draw_string(52, 70, "Max");
 g_draw_rectangle(55, 15, 9, 52);
 g_draw_string(90, 70, "T_refr");
 g_draw_rectangle(100, 15, 9, 52);
 g_draw_rectangle(132, 63, 25, 15);
 g_draw_string(139, 67, "Ok");

 button_pressed = 0;

 while((button_pressed <= 2))          //wait for complete input
 {
  buffer_touch_ad_y = get_touch_y();
  buffer_touch_ad_x = get_touch_x(buffer_touch_ad_y);
  if (buffer_touch_ad_x > 160)
   buffer_touch_ad_x = 160;
  if (buffer_touch_ad_x == 0)
   buffer_touch_ad_y = 0;
//=============   Bar 1     ================
  if ((buffer_touch_ad_x >= 20) && (buffer_touch_ad_x <= 50))
  {
   if ((buffer_touch_ad_y >= 17) && (buffer_touch_ad_y <= 67))
   {
    if ((buffer_touch_ad_y < 17) && (buffer_touch_ad_y > 10))
     buffer_touch_ad_y = 17;
    if ((buffer_touch_ad_y > 67) && (buffer_touch_ad_y < 74))
     buffer_touch_ad_y = 67;
    bar_1_height = 50-(buffer_touch_ad_y-17);   //getting height of bar (touch
input)
    for (i = 17; i < buffer_touch_ad_y-1; i++)   //clear bar
    {
     lcd_graphics_plot_pixel(12, i, PIXEL_OFF);
     lcd_graphics_plot_pixel(13, i, PIXEL_OFF);
     lcd_graphics_plot_pixel(14, i, PIXEL_OFF);
     lcd_graphics_plot_pixel(15, i, PIXEL_OFF);
     lcd_graphics_plot_pixel(16, i, PIXEL_OFF);
    }
    for (i = 12; i <= 16; i++)       //draw bar
    {
     g_draw_vertical_line(i, buffer_touch_ad_y, bar_1_height);
    }
    val_min = round(((long int)bar_1_height * 10000)/489);
    if (bar_1_height != 0)
```

```
      val_min_bar = (bar_1_height*2) - 1;
     else
      val_min_bar = 0;
     val_min_bcd[0] = (val_min_bar/10)+48;    //integer to bcd
     val_min_bar %= 10;           //integer to bcd
     val_min_bcd[1] = val_min_bar+48;     //integer to bcd
    }
   }
//=============   Bar 2     ================
   if ((buffer_touch_ad_x >= 73) && (buffer_touch_ad_x <= 87))
   {
    if ((buffer_touch_ad_y >= 17) && (buffer_touch_ad_y <= 67))
    {
     if ((buffer_touch_ad_y < 17) && (buffer_touch_ad_y > 10))
      buffer_touch_ad_y = 17;
     if ((buffer_touch_ad_y > 67) && (buffer_touch_ad_y < 74))
      buffer_touch_ad_y = 67;
     bar_2_height = 50-(buffer_touch_ad_y-17);
     for (i = 17; i < buffer_touch_ad_y-1; i++)
     {
      lcd_graphics_plot_pixel(57, i, PIXEL_OFF);
      lcd_graphics_plot_pixel(58, i, PIXEL_OFF);
      lcd_graphics_plot_pixel(59, i, PIXEL_OFF);
      lcd_graphics_plot_pixel(60, i, PIXEL_OFF);
      lcd_graphics_plot_pixel(61, i, PIXEL_OFF);
     }
     for (i = 57; i <= 61; i++)
     {
      g_draw_vertical_line(i, buffer_touch_ad_y, bar_2_height);
     }
     val_max = round(((long int)bar_2_height * 10000)/489);
     if (bar_2_height != 0)
      val_max_bar = (bar_2_height*2) - 1;
     else
      val_max_bar = 0;
     val_max_bcd[0] = (val_max_bar/10)+48;
     val_max_bar %= 10;
     val_max_bcd[1] = val_max_bar+48;
    }
   }
//=============   Bar 3     ================
   if ((buffer_touch_ad_x >= 118) && (buffer_touch_ad_x <= 132))
   {
    if ((buffer_touch_ad_y >= 17) && (buffer_touch_ad_y <= 67))
    {
     if ((buffer_touch_ad_y < 17) && (buffer_touch_ad_y > 10))
      buffer_touch_ad_y = 17;
     if ((buffer_touch_ad_y > 67) && (buffer_touch_ad_y < 74))
      buffer_touch_ad_y = 67;
     bar_3_height = 50-(buffer_touch_ad_y-17);
     for (i = 17; i < buffer_touch_ad_y-1; i++)
     {
      lcd_graphics_plot_pixel(102, i, PIXEL_OFF);
      lcd_graphics_plot_pixel(103, i, PIXEL_OFF);
      lcd_graphics_plot_pixel(104, i, PIXEL_OFF);
      lcd_graphics_plot_pixel(105, i, PIXEL_OFF);
      lcd_graphics_plot_pixel(106, i, PIXEL_OFF);
     }
```

```
      for (i = 102; i <= 106; i++)
      {
       g_draw_vertical_line(i, buffer_touch_ad_y, bar_3_height);
      }
      val_refr = bar_3_height;
      val_refr_bar = bar_3_height;
      val_refr_bcd[2] = (val_refr_bar/10)+48;
      val_refr_bar %= 10;
      val_refr_bcd[3] = val_refr_bar+48;
     }
    }
//=============   Button    ================
    if ((buffer_touch_ad_x >= 152) && (buffer_touch_ad_x <= 160) &&
(buffer_touch_ad_y >= 63) && (buffer_touch_ad_y <= 80))
    {
     button_pressed++;
    }
    else
    {
     button_pressed = 0;
    }
    if ((button_pressed == 2) && (val_min > val_max)) //error check and display
    {
     g_draw_string(112, 20, "Error:   ");
     g_draw_string(112, 30, "Val_min  ");
     g_draw_string(112, 40, ">        ");
     g_draw_string(112, 50, "Val_max! ");
     delay_ms(2000);
     g_draw_string(112, 20, "         ");
     g_draw_string(112, 30, "         ");
     g_draw_string(112, 40, "         ");
     g_draw_string(112, 50, "         ");
     button_pressed = 0;
    }
//=============  Display values  ================
    g_draw_string(5, 4, val_min_bcd);
    g_draw_string(23, 4, "C");
    g_draw_string(50, 4, val_max_bcd);
    g_draw_string(68, 4, "C");
    g_draw_string(95, 4, val_refr_bcd);
#ifdef GRAD_ON
    draw_grad(17, 2);
    draw_grad(62, 2);
#endif
  }
}

void draw_layout(void)
{
//=============   Layout    ================
  unsigned char i;

  lcd_graphics_clear();
  g_draw_rectangle(0, 0, 160, 80);
  g_draw_rectangle(0, 0, 60, 80);
  g_draw_string(5, 5, "A/D:");
  g_draw_string(13, 14, val_ad_bcd);
  g_draw_string(31, 14, "C");
  g_draw_string(5, 23, "Min:");
```

```
 g_draw_string(13, 32, val_min_bcd);
 g_draw_string(31, 32, "C");
 g_draw_string(5, 41, "Max:");
 g_draw_string(13, 50, val_max_bcd);
 g_draw_string(31, 50, "C");
 g_draw_string(5, 59, "T_refr:");
 g_draw_string(13, 68, val_refr_bcd);
 g_draw_horizontal_line(74, 75, 80);
 g_draw_vertical_line(74, 5, 70);
 g_draw_string(151, 65, "t");
 g_draw_string(67, 2, "C");
#ifdef GRAD_ON
 draw_grad(61, 0);
 draw_grad(25, 12);
 draw_grad(25, 30);
 draw_grad(25, 48);
#endif
#ifdef FLAME_AND_FLAKE_ON
 g_draw_rectangle(PIC_POS_FLAME_FLAKE_X-1, PIC_POS_FLAME_FLAKE_Y-1, 21, 21);
#endif
//=============   Arrow X   ================
 lcd_graphics_plot_pixel(152, 74, PIXEL_ON);
 lcd_graphics_plot_pixel(152, 76, PIXEL_ON);
//=============   Arrow Y   ================
 lcd_graphics_plot_pixel(73, 7, PIXEL_ON);
 lcd_graphics_plot_pixel(75, 7, PIXEL_ON);
//=============   99°C Level  ================
 g_draw_horizontal_line(73, 15, 2);
 g_draw_string(61, 12, "99");
//=============   Min/Max Level================
 for (i = 73; i <= 154; i++)
 {
  if (!(i%3))
  {
   lcd_graphics_plot_pixel(i, 14+round(60 - bar_2_height * 118 / 100),
PIXEL_ON);
   lcd_graphics_plot_pixel(i, 14+round(60 - bar_1_height * 118 / 100),
PIXEL_ON);
  }
 }
//=============   /Layout   ================
}

unsigned int get_touch_x(unsigned int y)
{
 unsigned int x;
 unsigned int x_left;
 unsigned int x_right;
//======= reconfig pins for x detection ================
 init_in_pin(TOUCH_ANA_PORT, 2);
 init_out_pin(TOUCH_ANA_PORT, 1);
 set_pin(TOUCH_ANA_PORT, 1);
 reset_pin(TOUCH_DIGIT_OUT_PORT, TOUCH_DIGIT_OUT_PIN);
 delay_ms(10);             //wait for energy to vanish

 ADMUX = 0x42;             //connect to AD_2
 ADCSRA = 0x97;             //enable ad converter
 ADCSRA |= 0x40;             //start converter
```

```
 while((ADCSRA&0x10)==0);          //waiting for end of calculation
 ADCSRA |= 0x10;
 x = ADCL|ADCH<<8;
 if (x == 0)
  return 0;
 x_left = round(427-((y*(427-186))/80));    //calculating x limits related to
y_pos
 x_right = round(300-((y*(300-110))/80));    //calculating x limits related to
y_pos

 return round(160-((x-x_right)*160/(x_left-x_right))); //calculating x pos and
return
}

unsigned char get_touch_y(void)
{
//======= reconfig pins for y detection ================
 init_in_pin(TOUCH_ANA_PORT, 1);
 init_out_pin(TOUCH_ANA_PORT, 2);
 reset_pin(TOUCH_ANA_PORT, 2);
 set_pin(TOUCH_DIGIT_OUT_PORT, TOUCH_DIGIT_OUT_PIN);
 delay_ms(10);             //wait for energy to vanish

 ADMUX = 0x41;             //connect to AD_1
 ADCSRA = 0x97;             //enable ad converter
 ADCSRA |= 0x40;             //start converter
 while((ADCSRA&0x10)==0);          //waiting for end of calculation
 ADCSRA |= 0x10;

 return round((((ADCL|ADCH<<8)-262)*10)/65);   //return y pos
}

void clr_diagram(void)
{
 unsigned char x_buffer = 75;
 unsigned char y_buffer = 75;
 unsigned char x = 0;

 for (x = 0; x <= 79; x++) //clear old graph on display
 {
  if (diagram[x] != 0)
  {
   draw_line(x_buffer, y_buffer, x + 75, (-1*((signed int)diagram[x]))+75, 0);
   draw_line(x_buffer, y_buffer, x + 75, (-1*((signed int)diagram[x]))+74, 0);
   draw_line(x_buffer, y_buffer, x + 75, (-1*((signed int)diagram[x]))+73, 0);
   //lcd_graphics_plot_pixel((x*8)+75+bit, (-1*(y))+74, PIXEL_OFF);
   x_buffer = x + 75;
   y_buffer = (-1*((signed int)diagram[x]))+74;
  }
  diagram[x] = 0;    //clear data array
 }
}

void draw_diagram(void)
{
 unsigned char x = 0;
 unsigned char x_buffer = 75;
 unsigned char y_buffer = 75;
```

```
 for (x = 0; x <= 79; x++) //draw graph on display
 {
  if (diagram[x] != 0)
  {
   draw_line(x_buffer, y_buffer, x + 75, (-1*((signed int)diagram[x]))+74, 1);
   x_buffer = x + 75;
   y_buffer = (-1*((signed int)diagram[x]))+74;
  }
 }
 g_draw_string(151, 65, "t"); //redraw t for time axis
}

void draw_line(unsigned char x_start, unsigned char y_start, unsigned char
x_end, unsigned char y_end, unsigned char draw)
{
 signed char dx;
 signed char dy;
 float m, n;
 unsigned char style = 0;

 if (draw)
  style = 0xff;          //pixel on
 else
  style = 0x00;          //pixel off

 dx = (signed)(x_end - x_start);     //calculating delta x
 dy = (signed)(y_end - y_start);     //calculating delta y
 lcd_graphics_plot_pixel(x_start, y_start, style);//draw start pos
 if (fabs(dx) > fabs(dy))
 {
  m = (float) dy / (float) dx;      //calculating slope of graph
  n = y_start - (m*x_start);       //calculating offset
  if (dx < 0)            //direction of graph
   dx = -1;
  else
   dx = 1;
  while (x_start != x_end)       //draw graph on screen
  {
   x_start += dx;
   lcd_graphics_plot_pixel(x_start, (int)(m*x_start + n), style);
  }
 }
 else
  if (dy != 0)
  {
   m = (float) dx / (float) dy;     //calculating slope of graph
   n = x_start - (m*y_start);      //calculating offset
   if (dy < 0)           //direction of graph
    dy = -1;
   else
    dy = 1;
   while (y_start != y_end)      //draw graph on screen
   {
    y_start += dy;
    lcd_graphics_plot_pixel((int)(m*y_start + n), y_start, style);
   }
  }
}
```