Časopis priemyselnej elektrotechniky (JIEE)
Journal of Industrial Electrical Engineering
ISSN 2454-0900
www.jiee.eu

Development tool based on touch display for educational and industrial applications

[1]*Slavomír ŠIMKO,* [2]*Dobroslav KOVÁČ*

[1]BSH Drives and Pumps, s.r.o., Michalovce, Slovak Republic, [2]Department of Theoretical and Industrial Electrical Engineering, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovak Republic

[1]slavomir.simko@bshg.com, [2]dobroslav.kovac@tuke.sk

***Abstract* — The paper describes how to choose, program and design a touch display, which is becoming one the best solutions for controlling electronic systems and devices. Mentioned facts, criteria and the design of own solution tries to present the possibilities, which can touch display offer and guide the reader to create some application.**

***Keywords* — touch display, control panel, temperature measurement, programming**

I. INTRODUCTION

The user interface for any programmable electronic device is very important. It allows the user to change the behavior of electronic system. The more complex the system is, the more difficult is to design an interactive control panel for all system features.

Touch display is the most modern way of complete "user friendly" interface. Hardware signalization and control elements like LEDs, buttons, switches, potentiometers and others are all substituted by objects on display. These objects can dynamically change their shape, color, text, etc. based on system inputs and its running program. All features can be easily and quickly modified by programming software, which gives a ton of possibilities how to greatly fit control panel functionalities for a concrete application.

II. DISPLAY REQUIREMENTS

The display choice is based on criteria, which makes it a suitable as an educational tool for students and allows them to develop a custom touch panel for some industrial application. To meet the requirements, display is being judged by its:

- properties,
- programming environment,
- price,
- availability.

A sufficient screen resolution, size and touch sensitivity of display is the main parameter, which are responsible for how smooth and precise the panel would be.

The most of products use TFT screens in many variations of resolution. Customer can then choose the touch sensing technology. Currently, there are two most common types of panels- resistive and capacitive. Comparing these types, resistive sensing is an older and simpler method, which uses the principle, where point of touch creates a pressure on screen and the deformation is sensed as a change of resistance. Lower sensitivity, lower color contrast and the fact that display for its principle cannot be behind a protective glass are some disadvantages for a consideration. On the other hand, this type of panel has significantly lower price and is sufficient for standard applications.

Basically, the biggest points in ratings of the display are for its computing power and its programming environment. The computing power is based on processor, which runs the program. Fast processor with enough program and data memory will enable designing applications with lot of graphics and complex programs. How easy it will be, depends on software and resources. Current software distributors are trying to make the programming environment in way, that customer/user needs minimum knowledge in programming.

Price and availability of the display are related. Many e-shops are selling cheap displays, but not available in our country.

All mentioned facts have to be considered, when choosing the suitable display.

III. CHOICE OF THE DISPLAY

The best match for selected criteria becomes the intelligent display uLCD-43DT from 4D systems. The uLCD-43DT model has:

- 4.3" LCD-TFT display with 480 x 272 resolution, RGB 65k true to life colors
- 4-wire Resistive Touch Panel
- Computing power by DIABLO16 Processor
- 6 Banks of 32kB Flash
- 32kB of User RAM
- 16 General Purpose I/O pins for user interfacing, including:
  - 3x I2C channels
  - 1x SPI dedicated for SD Card and 3x configurable SPI channels
  - 1x dedicated and 3x configurable TTL Serial Com ports
  - 4x GPIO available as 12bit analog inputs
  - 6x GPIO available as Pin Counters
  - 6x GPIO available as PWM
  - 10x GPIO available as Pulse Output
  - 14x GPIO available as Quadrature Encoder Inputs (2 channels)
- on-board micro-SD memory card connector for multimedia storage and data logging purposes
- audio amplifier with small speaker to play sounds

All models from 4D systems are being programmed in free development software called Workshop4 IDE. This software consists of multiple environments, making it suitable for every user with different programming skills and for specific requirements of application. All programming and configuring is made here and final application can be then easily downloaded to the display.

With all mentioned features seems uLCD-43DT like a great option for purposes of this work. The next section of this document will try to present the steps of how to create the program and configuration of display in order to better understanding its possibilities.

IV. APPLICATION

To present the functionalities of the display uLCD-43DT will be in steps created a simple application, describing all software and hardware configuration. The display will be used as a control panel with two switchable digital outputs, sensing two digital inputs and one analog input to measure outer temperature with thermistor.

Hardware

The following table Tab.1 shows possible configurations of all 16 GPIO pins of the display.

These pins are available on the back side of the touch display on header J1, which is shown on the figure Fig.1.

TAB. 1 GPIO CONFIGURATION

	Digital input	Digital output	Analog read	Pulse output	PWM output	Pin counter	Quadrature input
PA0	x	x	x	x			x
PA1	x	x	x	x			x
PA2	x	x	x	x			x
PA3	x	x	x	x			x
PA4	x	x		x	x	x	x
PA5	x	x		x	x	x	x
PA6	x	x		x	x	x	x
PA7	x	x		x	x	x	x
PA8	x	x		x	x	x	x
PA9	x	x		x	x	x	x
PA1	x	x					x
PA1	x	x					x
PA1	x	x					x
PA1	x	x					x
PA1	x						
PA1	x						

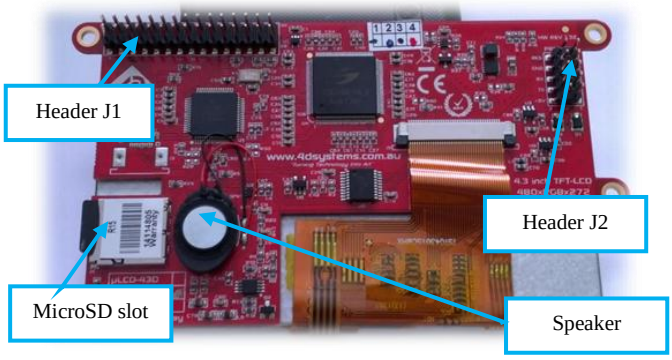

Fig. 1 Back view of touch display uLCD-43DT

Pin PA0 will be used as analog input, pins PA1, PA2 as digital inputs and PA3, PA4 as digital output. Schematic view on the Fig. 2 shows how to make all the circuitry and connections to a J1. To enable the display programming and communication with a PC we will connect USB to UART adaptor to a header J2. Description of each pin on J1 and J2 headers is on the Fig. 3.

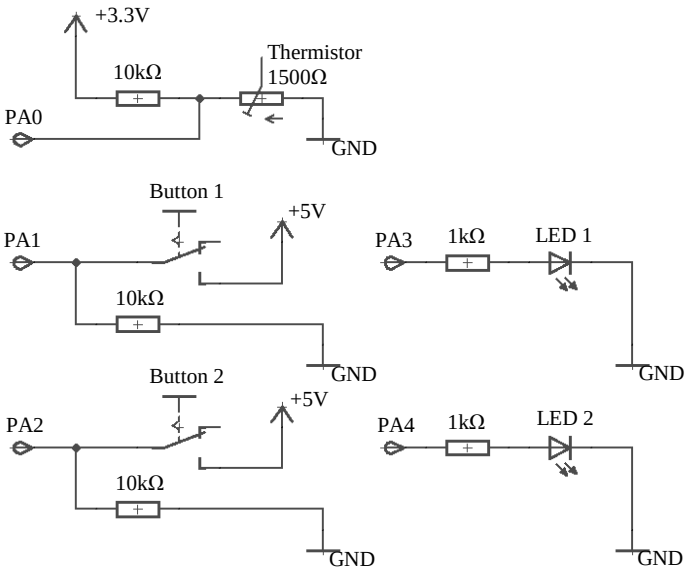

Fig. 2 Schematic for the application with uLCD-43DT

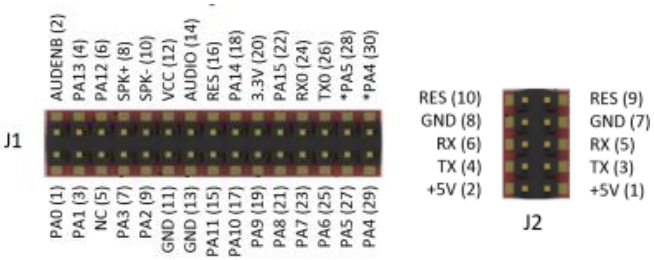

Fig. 3 Description of the uLCD-43DT connectors

Software

First of all we have to download the 4D workshop programming software from web page: http://www.4dsystems.com.au/product/4D_Workshop_4_IDE.

Installation is standard and intuitive. When we run the software and click on the "Create a new Project", we have to select the right type of the display (in our case it is uLCD-43DT). Now we will select the programming environment:

- The available options are:
  - "Designer" that enables the user to write the code in its natural form to program the display,
  - "ViSi" that enables the drag and drop placements of objects, generates their code and visualizes how the display will look while developing,
  - "ViSi Genie" that enables to create the display program without writing the code. The code is automatically generated and the user just chooses the display objects and their parameters,
  - "Serial" that transforms the display into a slave serial module and allows the user to control the display with any other serial device.

The "ViSi" environment appears to be the best for purposes of this work, because the user can learn all the functions of the display objects, but also is able to learn the programming functions and coding.

Writing the code

After opening the new ViSi project there appears the environment shown on the Fig. 4. Toolbox is on top of the screen, code editor is on the left side with information about compilation process under it. On the right side is display visualization with object editor under it. Every part can be resized or its position can be changed.

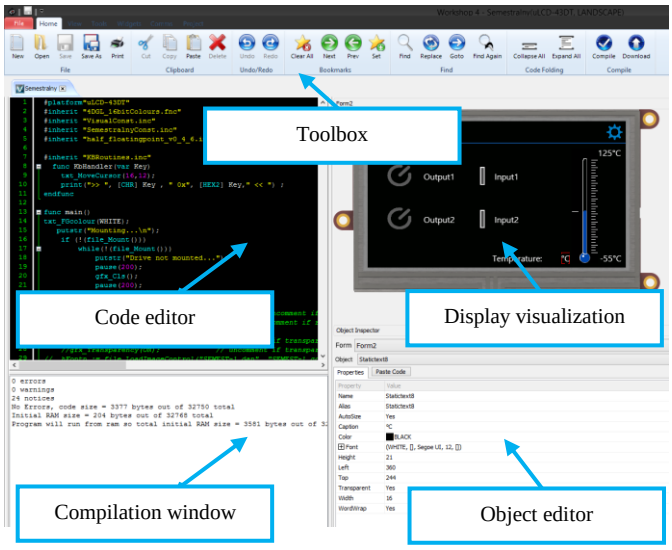

Fig. 4 ViSi programming environment

Program structure is typical. At the first lines are included libraries and the definition of our platform. The code is typically written in the main function *func main* and is repeated by processor forever. The new program already consists of some commands and functions in the main, but they are commented by default. We will uncomment this code to enable mounting the images, fonts and data from the SD card.

For our project are needed two buttons to change the output states, two LEDs that will display states of the input and thermometer image that will display the temperature sensed from the analog input. We will also add some labels to describe all the objects. These objects can be found by clicking on the "Widgets" from the "Toolbox". User can select the multiple types of predefined buttons or create his own from "Buttons" tab. In the tab "Digits" can be found LED objects. In the tab "Gauges" can be found thermometer object and other. We will click on the concrete object and then on the panel representation on the right, which adds the object to our project. The objects parameters like size, name, and color and other properties can be modified in the object editor. Mouse click on the "Paste Code" or "Paste all Code" from the object editor will generates the code of the object on the position of the current line of our program.

Generated commands:

- *img_ClearAttributes(hndl, iOutput1, I_TOUCH_DISABLE);* makes the object named "Output1" touchable,
- *img_SetWord(hndl, iOutput1, IMAGE_INDEX, frame);*
  sets the object named "Output1", where "frame" attribute represents the number of image displayed (when displaying the output- 0 is for switched off and 1is for switched on; when displaying thermometer or any video object the number represents a video frame, etc.)
- *img_Show(hndl,iOutput1) ;*
  will show the object named "Output1" on the display.

As mentioned ViSi environment needs programming knowledge and we need to add another commands and functions to finish our application.

Defining the pin operation:

- *pin_Set(function,pin);*
  sets the pin, where "function" is a number specifying the pin operation (0 for digital input, 3 for digital output, 5 for analog input) and "pin" represents GPIO pin number PA0-PA15.
- *pin_Val(pin, value);*
  writes a value to the digital pin, where "pin" represents GPIO pin number PA0-PA15 and "value" represents state 0 or 1.
- *pin_Read(pin);*
  reads a pin value, where "pin" represents GPIO pin number PA0-PA15. Returns 0 or 1 when pin is set as digital and returns 12bit value, when set as analog;

To make the objects react on a touch press we will use a condition *if (touch_Get(TOUCH_STATUS)== TOUCH_PRESSED.*

Condition *if (img_Touched(hndl,-1)== iOutput1)) ;* will be true when we touch the object on display named "Output1". In this case we will change its state with *pin_Val(pin,value);* and with *img_SetWord(hndl, iOutput1, IMAGE_INDEX, frame);* we change the image as a feedback to user. The same change of image will be programmed when *pin_Read(pin);* changes its state.

The programming features have one big disadvantage. There is available only one type of a variable- *var*. Strings, chars, integers are all stored only in this one type. In the most of cases it can considered as a plus, because user is exempt from learning more data types and their operations. One the other side, this wastes the processor memory- missing in more complex programs and also complicates the math operations e.g. conversions from analog reading. To get the accuracy from conversions, we need to work with floating point numbers. The programmers of the ViSi environment made special functions *flt*, which can transform the integers to floating point numbers and perform math operations. Requirement is, that every variable must be declared as a field with minimal length

of 2, e.g. *var Resistance[2];*. Note, that every number in the calculations has to be converted, even number 1 has to be converted to variable like *var One[2] a*nd then *flt_ITOF(One,1);*.

To get and display the temperature value from analog reading we will need to program the following steps:

1. Calculate the actual thermistor resistance from ADC value.
   We can use the formula:

$$R = \frac{R_{Series}}{\frac{\mathrm{Re}\,solution}{ADCvalue} - 1} \tag{1}$$

   where R_{Series} is 10kΩ from schematic on Fig. 2, *Resolution* is the maximal value from ADC according to its number of bits (2^{12}-1= 4095), *ADC value* is the actual value from analog reading (0 to 4095).

   To make the results more precise we will make the average from the 20 samples of analog read every 1ms.

2. Calculate the resistance to temperature value with a good approximation.
   We can use the formula:

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B}\ln\left(\frac{R}{R_0}\right) \tag{2}$$

   known as *B* or *β* parameter equation, where T_0 is room temperature (25 °C = 298.15 K), *B* is thermistor coefficient (in our case 3950), R_0 is resistance at room temperature (in our case 1500Ω). The result *T* is the temperature in Kelvin, which can be easily converted to °C by subtraction with 273.15.

3. Print the results.

The following code, resp. function calculates and prints the temperature on display based on our previous configuration.

```
func getTemperature()
//*** Defining variables ****************************************
   var anVal;
   var anValSamples[20];
   var coefB:=3950;
   var temperature[2];
   var Samples[2];
   var Rnom[2];
   var RT0[2];
   var Steps[2];
   var Result[2];
   var Anval[2];
   var One[2];
   var B[2];
   var T0[2];
   var Kelvin[2];
   var numberOfSamples;
//***Geting analog reading samples ********************************
   for(numberOfSamples:=0;numberOfSamples<20;numberOfSamples++)
      anValSamples[numberOfSamples] := pin_Read(PA0);
      pause(1);
   next
```

```
    anVal:=0;
    for(numberOfSamples:=0;numberOfSamples<20;numberOfSamples++)
      anVal+=anValSamples[numberOfSamples];
    next
  //***Conversion to floats ****************************************
    flt_VAL(T0, "298.15");
    flt_VAL(Kelvin, "273.15");
    flt_ITOF(Rnom,10000);
    flt_ITOF(RT0,1500);
    flt_ITOF(Steps,4095);
    flt_ITOF(Anval,anVal);
    flt_ITOF(One,1);
    flt_ITOF(Samples,20);
    flt_ITOF(B,coefB);
    flt_DIV(T0,One,T0);
  //***Resistance calculation ***************************************
    flt_DIV(Anval,Anval,Samples);       // get average ADC value
    flt_DIV(Result,Steps,Anval);        // Resolution/ADC value
    flt_SUB(Result,Result,One);         // (Resolution/ADC value)-1
    flt_DIV(Result,Rnom,Result);        // RSeries/((Resolution/ADC value)-1)
  //***Calculation using B parameter equation ***********************
    flt_DIV(Result,Result,RT0);         // (R/Ro)
    flt_LOG(Result,Result);             // ln(R/Ro)
    flt_DIV(Result,Result,B);           // 1/B * ln(R/Ro)
    flt_ADD(Result,Result,T0);          // + (1/To)
    flt_DIV(Result,One,Result);         // invert
    flt_SUB(Result,Result,Kelvin);      // convert to °C
    to(temperature);
    flt_PRINT(Result, "%.1f");
    gfx_MoveTo(328, 272-21) ;
    putstr(temperature);                // print on display
  endfunc
```

Downloading the code to display

After final compilation of our code we have to click on a "Download" button from "Home" toolbox. The Workshop4 automatically asks to select the micro SD card, formated as FAT, where will be saved our application images and files. Card with succesfully downloaded data can be now put from PC reader to microSD slot on the display and application should run automatically. Final application enabling the user to control the states of 2 digital outputs and informs him about states of 2 digital inputs and about temperature is shown on Fig. 5.

Fig. 5 The display and its application

V. CONCLUSION

Touch display uLCD-43DT opens the great possibilities to design and implement the complete interactive control panel to any industrial application. Before making of complex programs and circuits was necessary to create the application, which would demonstrates functionalities of display and helps to understand the programming techniques and principles for the further development. After getting these experiences can user implement by program more display interactions and extend program for more IO, math or communication functions.

REFERENCES

[1] http://www.sos.sk/optoelektronicke-a-zobrazovacie-prvky/e33-moduly-tft
[2] http://www.4dsystems.com.au/product/uLCD_43D/
[3] http://www.4dsystems.com.au/productpages/uLCD-43D/downloads/uLCD-43D_datasheet_R_1_1.pdf
[4] http://www.4dsystems.com.au/productpages/DIABLO16/downloads/DIABLO16_internalfunctions_R_1_13.pd
[5] http://www.4dsystems.com.au/product/4D_Workshop_4_IDE
[6] http://www.4dsystems.com.au/productpages/DIABLO16/downloads/DIABLO16_datasheet_R_1_5.pdf
[7] http://www.4dsystems.com.au/appnotes
[8] http://en.wikipedia.org/wiki/Thermistor
[9] https://learn.adafruit.com/thermistor/using-a-thermistor
[10] Kováčová, I.: *Applied electronics*. 1st. ed. TU Košice, 2015, p. 141, ISBN 978-80-553-1943-8
[11] Kováč, D., Kováčová, I.: *Industrial electrical engineering*. 1st. ed. TU Košice, 2015, p. 145, ISBN 978-80-553-1939-1
[12] Ďaďovský, M., Kováčová, I.: *Distribution of electromobiles*. In: SSIEE 2015 : Proceeding of scientific and student's works in the field of Industrial Electrical Engineering, Volume 4, TU Košice, 2015, pp. 14-16, ISBN 978-80-553-2151-6