Quick starter for Visual TFT of Mikroe

Henk Kiela Opteq Mechatronics BV

Introduction

Mikroelektronika supplies a nice software tool to develop embedded tft touch screen applications. But starting a first project with it is somewhat a challenge as their documentation is nothing more than the help file in Visual TFT and the examples. But none of this tells how to make a simple application and what the code generated actually does

Therefore I documented my first steps to help others.

What does Vtft and what not

Vtft has a palette with a simple but practical set of GUI building elements with boxes, buttons, images, checkboxes and a progress bar. For not too complicated TFT touch applications this is OK.

It generates a complete frame work of code that looks impressive at the first start. But is is not immediately clear what to do with all the modules. The answer is: 'Very little'. And that is the good news.

In fact the only useful part of the generated code shown in the tab User code of the Vtft tool. The rest is useful or documentary purposes or later for understanding how the code Works.

In the compiler environment several modules are generated. We just describe them in brief for the example generated in C:

- xxx_main.c holds the maim program of the applications. Here initialization and execution are called. Note that there is an endless loop in the mainprogram calling Check_Tp every time to check for events on the touch screen
- xxx_driver.c holds almost all programming code to build the screen graphics, set initial properties of buttons, labels, implement clicking behaviour of buttons pressed, initialize touch panel, set up I/O for TFT and touch screen and implements Check_Tp.
- The file xxx_events_code.c implements the event handler when for example a button is clicked on the screen.

For C two header files are generated: ccc_objects.h and xxx_resource.h.

The xxx stands for the project name you entered for the Vtft project.

What to do with all these files?

For most application only the xxx_events_code.c and the xxx_main.c need to be adapted to build a full application. The rest is for documentation

How to load the project into a device.

Use the method you would use for any compiled project to load the generated code into your device. In the case of the Mikromedia devices the USB boot loader method is the quickest.

Use the boot loader software to load the software:

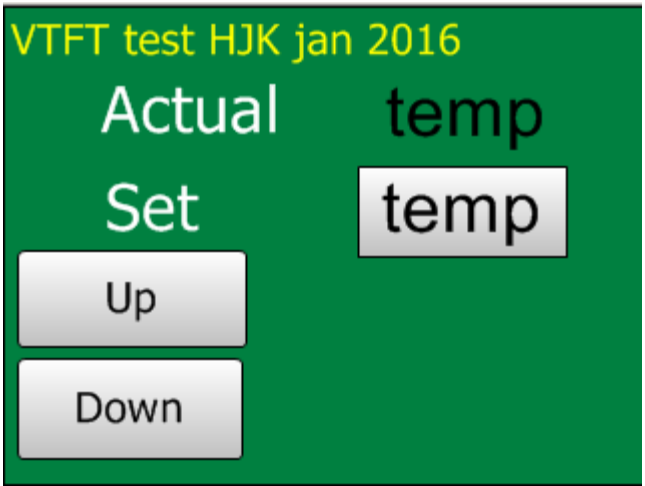
- Press reset on the device.
- Press the connect button as soon as the device is visible (for a few seconds only) in the tool
- Press select Browse for HEX to load the generated hex file (Do this every time to be sure to load the last version!)
- Press begin uploading.

After a few seconds the device reboots and starts the application and you will probably see a black screen requesting calibration.

How to skip the touch screen calibration

In the standard Project settings the TFT touch screen will always start with the annoying calibration screen. The idea of the calibration is to match the corners for the touch screen with the measured ADC values. In practice the standard values are very well usable. So you can skip the calibration in most cases.

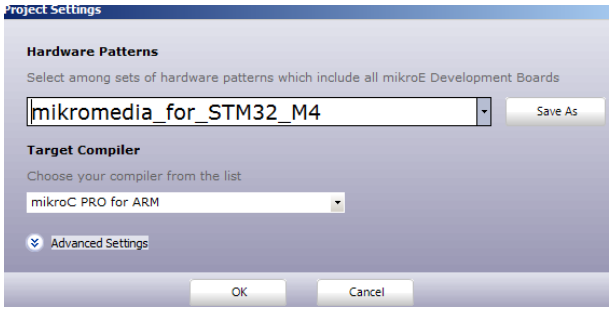
To do that open the menu item Project settings and click advanced and the Touch panel tab. At the bottom select Preset in stead of the standard Manual calibration.



Where to start a project

After dragging components onto the start screen and properties are set for the components, pressing 

Generates a complete frame work in the chosen language (Basic, Pascal or C) for the selected hardware (and processor) as specified in the Project settings (menu item). The code does not do much more than showing a screen when loaded in the device as the event code has not been written.



Always good to start with a simple application, in our case for the ARM C compiler for the Micromedia M4. We assume that the way of working is similar for all other hardware and software platforms.

Note on displaying changing values:

Here is a note to take in account when designing the GUI:

Use buttons for displaying values. The simplest application will need fixed labels, buttons and display fields. It took me some time to understand that updating a label on the screen is cumbersome as previous text needs to be rewritten in the background color first to be erased. I do not understand why MikroE did not add an entry field for displaying an entering values. Anyhow, for displaying a button works well. Set for the backgrounds and the pencolor for the button to the same color as the screen background or any other background and display the new value in the button caption.

Generated code in C:

As you can see in the code of the added project it is pretty simple to build your application in the User code section (xxx_driver.c) and xxx_main.c

xxx_main.c only needs modification to add your own initialization. Xxx_driver.c holds all the program logic when running.

NB: It seems that Visual TFT and the compiler IDE stay in sync, but not always… So you are able to make changes in both environments while files stay synced