COMPUTER ORGANIZATION (ELE408) LAB 6

TWR_LCD usage in TWR-K70F120M tower system

1. Pre-lab Report

- Read all documents in directory C:\Freescale\TWR_LCD\Documentation\.
- Read this lab handout carefully. Take notes and prepare the programs required in this handout.

2. Objectives of this Lab

This lab is intended to familiarize you with design and implementation of a touch screen LCD display on an embedded system.

You will use the Freescale TWR-LCD board in your tower system learn

- how to use bootloader flash a per-compiler image to TWR-LCD board,
- how to use Freescale Embedded GUI(D4D) to add text, icon, and button in a given project,
- and how to flash a bootloader image to TWR-LCD board.

3. Basics

The TFT-LCD module populated on the TWR-LCD board is integrated with Solomon Systech TFT-LCD controller driver SSD1289. This TFT-LCD controller driver integrates the graphic display data RAM, power circuits, gate driver, and source driver into a single chip. On this board, the graphic display data RAM is interfaced with the common MCU/MPU through a 16-bit 6800-series / 8080-series compatible parallel interface or SPI.

The TWR-LCD features a 3.2" QVGA TFT LCD Display with touch sensitive overlay, 5-way navigation control, MicroSD Card slot, dedicated MCF51JM microcontroller, and a Piezo Buzzer for audible feedback. The LCD Display Controller is accessible to the dedicated MCF51JM microcontroller through the SPI. The LCD Display Controller is also accessible to any capable Tower MCU module utilizing either the SPI or the EBI.

Many embedded applications require a human interface for control and display purposes, but until recently these have been restricted to simple led or segment LCD displays for most MCU applications due to the limited RAM and ROM available on the chip. When the required complexity has led to the need for a graphics LCD panel, then the hardware would normally be changed to add in an additional MPU with a dedicated LCD drive or with the addition of an external graphics LCD controller. The additional cost associated with this has restricted the introduction of graphics LCDs to most MCU applications. However, with the recent introduction of low-priced graphics LCD panels with simple serial or parallel interfaces and integrated display RAM, graphic LCD applications can now easily be implemented with MCUs if the graphics driver software is well designed and takes into consideration the limited MCU resources. The D4D has been specifically written with the constraints of an MCU (low FLASH and RAM) and the assumption that the graphics display RAM is write-only, as is the case of many “Smart LCD” panels. As a result, D4D can produce stunning layered graphics displays using only limited RAM and FLASH from MCU and has a very small library footprint.

A solution with the LCD is needed in case of control, measure, and diagnostics of some system or device with human-machine interface, when the operator can actively change conditions and requirements. The D4D is capable of generating user menu, graphics, pictures, text, and display them on the LCD module. It allows interacting with all objects, dynamically changing, adding, or removing them, and also read/write their status or current value. The D4D also fully supports touch screen capabilities of the LCD displays.

- Supports graphical color LCD displays of various sizes
- Very small RAM (volatile) memory footprint
- Multiple platform support
- Object style of driver
- Very smart support-screen-oriented structure of user code
- Custom screen sizes, position, and header like a window
- Objects:
    - — Button
    - — Check Box / User handled Radio Button
    - — Gauge

—Icon

—Label

—Menu

— Picture

— Slider

—Graph

- Touch screen support
- Multiple font support
- Buffer for input keys

4. Experiment Requirements and Procedures

In this lab, you are required to finish several tasks including the usage of MCF51JM bootloader, Programming a required window.

In the Lab:

- Read first chapter of C:\Freescale\TWR_LCD\Documentation\TWR_LCD_DP_Walk Through.pdf very carefully. Finish bootloader experiment in chapter 1.3. Play with the demo and see how it works.
- Set workspace as "C:\Freescale\TWR_LCD\" imported projects Demo_MCF51JM_SPI and Hello_World_MCF51JM_SPI in directory "CW MCU 10"

1. Compile and download Demo_MCF51JM_SPI to TWR-LCD board.
2. Read main.c in \Sources very carefully. Draw a flowchart of main function, put it in your lab report.
3. Read screen_icon.c in \Sources very carefully. Draw a flowchart how this traffic light demo works. How many functions need to be called for a complete demo application? List them in your lab report. How many D4D functions were called? What's the usage of each function? List in your lab report.
4. Compile and download project Hello_World_MCF51JM_SPI. Play with the demo project
5. Read screen_hello_world.c very carefully, modify it to finish following tasks:

```
1) Add a text box in the window, the content of the text box is
   your name.
2) Change the background color to blue.
3) When touching the text box you added, the content of the text
   box should change to "ELE408_Lab6"
```

- Import project JM128_Bootloader, flash it to the board. Try flash a pre-compiled image. See if it works

5. Lab Report Requirements

In your lab report, you should discuss your designs, trade-offs between performance and power, Explanation and interpretation of your results are very important. The lab report will be graded based on your report and discussions. Total mark for the report is 100 points.

- Prelab report: 20 points
- Successful experiment: 50 points
- Results analysis, interpretation, and discussions on your design and engineering constraints: 30 points

In the following items, numbers inside each bracket indicates the point you will earn on a satisfactory report and discussions.

In your discussion and explanations of your results, you should consider the following constraints:

- What knowledge of mathematics, science and engineering have you applied in this lab and what tools have you used in this lab?[5]
- Economic Constraint (performance/cost ratios) [5]
- Manufacturability, Modularity and Expandability Constraints, Environmental Constraint (power consumption) [5]
- Sustainability: Is your design and implementation sustainable? [5]
- What is the potential impact of your design on real time applications? [5]
- How would you utilize the memory hierarchy available to you to design real time applications? [5]

For each of the above programs hand in the debugged source code with comments; the machine code is not necessary. Be very specific with your comments that explain what you are doing and why you are doing it.