Lab 6: LCD Display

COEN-4720 Embedded Systems

Cris Ababei
Dept. of Electrical and Computer Engineering, Marquette University

1. Objective

The objective of this lab is to use for the first time an LCD display. We will use SPI communication protocol to talk to the display.

2. Description

Prerequisites for this Lab
Read Chapter 10 and 15 from the textbook.

LCD TFT Display
In this lab, we will use an LCD display. Specifically, we will use the **ST7789 EYESPI Display** from Adafruit, which is a 2.0" 320x240 Color IPS TFT Display with microSD Card Breakout. The display module is available here:
https://www.adafruit.com/product/4311
You can download PCB layout files for this display module here:
https://learn.adafruit.com/2-0-inch-320-x-240-color-ips-tft-display/downloads
The display module uses a display driver, the ST7789 driver from Sitronix.
https://www.sitronix.com.tw/en/products/aiot-device-ddi/

In working with this display, we will need to connect it to our Nucleo board, and we will make use of an existing library to be able to draw stuff or write text on the display's screen.

Connection of LCD Display to Nucleo Board
The communication to the LCD display is done via the SPI communication protocol or standard. In addition to the SPI connections to the LCD display module, we need to create a few more connections as indicated below.

Display -> Nucleo Board
GND -> GND
VCC -> 5V or 3.3V
SCK -> PB3 (SCK of SPI1 peripheral of MCU) – *this PB3 is default pin for SCK*
MISO -> PA6 (MISO of SPI1 peripheral of MCU) – *default pin for MISO*
MOSI -> PA7 (MOSI of SPI1 peripheral of MCU) – *default pin for MOSI*
CS -> PB6
RST -> PC7
Data or Command (D/C) -> PA9
SDSC -> *left disconnected*
BL -> *left disconnected*

NOTE: Pins PB3, PA6, PA7 are the default pins that the peripheral SPI1 uses as connections for SCK, MISO, and MOSI. So, we do not need to do anything about them when we will configure/enable the SPI1 peripheral using the STM32CubeMX tool. However, pins PB6, PC7, PA9 will have to be configured separately to be GPIO outputs (can be done using the STM32CubeMX tool, from System

Core > GPIO and add those three pins as Outputs; or, we can skip using the STM32CubeMX tool and configure those pins directly on our own inside the code after project is created – we will do that in the example 1 of this lab).

For convenience, see in Figure 1 below which are those connections on the Nucleo board. In addition, in Figure 2, you can see my own wiring for this lab.

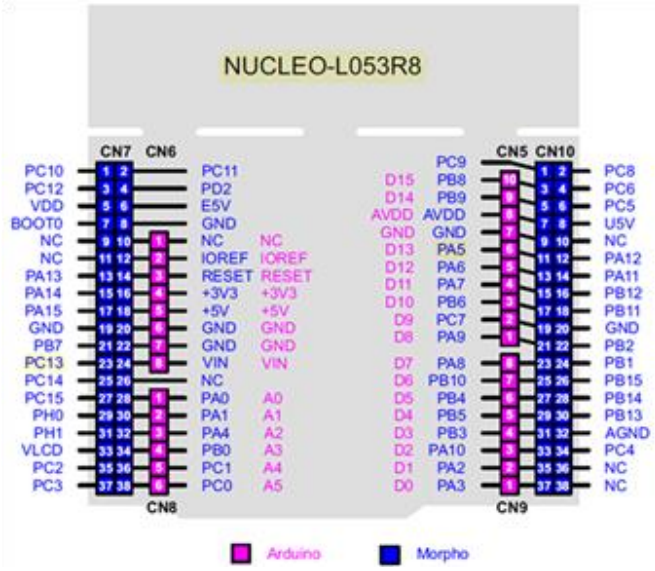

Figure 1: Pins of Nucleo-L035R8 board.

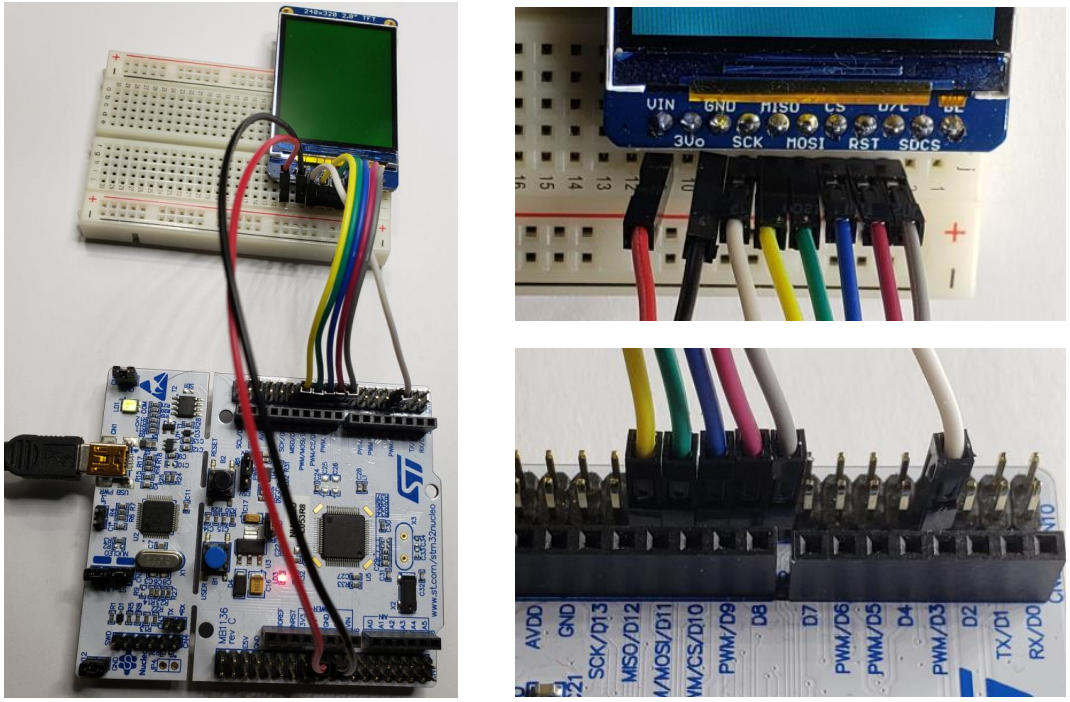

Figure 2: Connections of LCD Display to Nucleo board.

LCD Graphics Library

In working with an LCD display, one needs some graphics library that provides basic functions to draw and write on the display (of course, one could always white his/her own such library – but, that is outside the scope of this class).

We will use a slightly modified/cleaned version of the following library: "Driver for ST7789 displays using STM32 and uGUI library", available on github:
https://github.com/deividAlfa/ST7789-STM32-uGUI

The actual modified version is included in the STME32CubeIDE project for example 1, provided with the files for this lab.

Example 1

In this example, we create a simple project that uses the LCD display to run several tests of the LCD library. While the complete project folder is included in the files for this lab, you should create your own project from scratch and then copy the necessary files and code from the project provided.

When creating the new project, inside STM32CubeMX Tool right after you created the new project, select to use the SPI1 peripheral and configure it as shown in Figure 3 below. Copy the LCD/ and UGUI/ folders into your own project folder inside Drivers/ folder. Also, copy from the provided main.c the code portions that you are missing inside your newly created main.c.

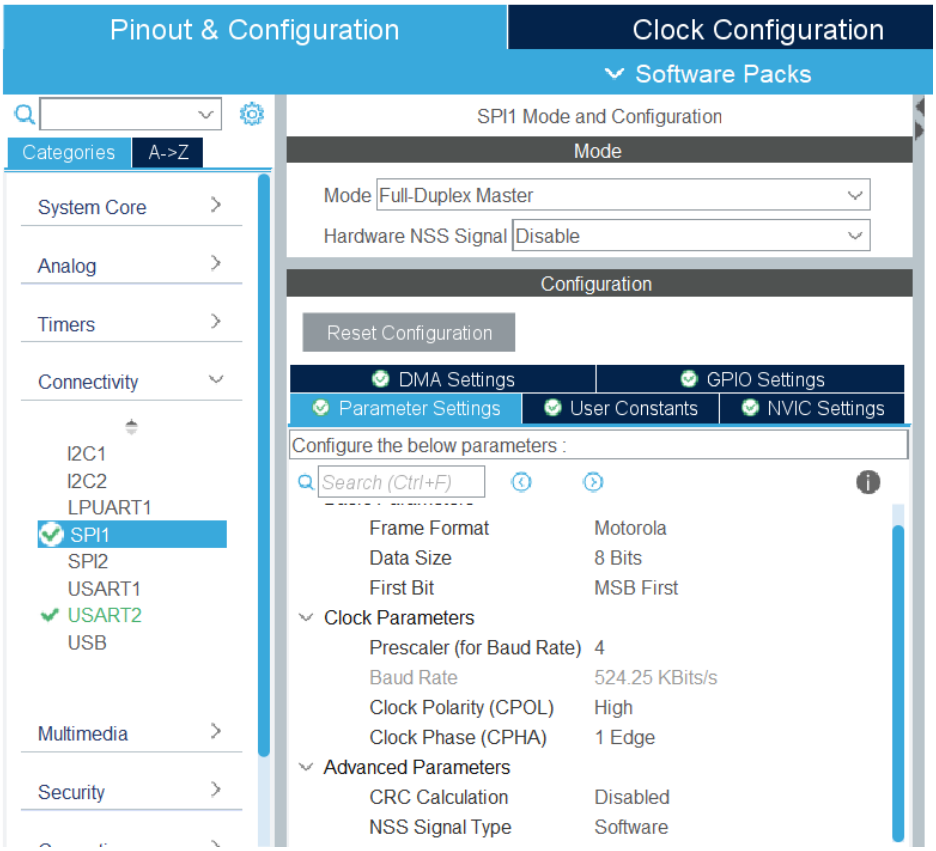

Figure 3: Configuration of SPI1 device in STM32CubeMX/CubeIDE.

Pay attention to the code inside the function:
`static void MX_GPIO_Init(void)`
where several pins are configured, including those connected to the RST, CS (chip-select), and D/C (data or command) pins of the display module. For example, the code that takes care of the configuration of the RST pin is:

```
  /*Configure GPIO pin Output Level - RST */
  HAL_GPIO_WritePin(GPIOC, GPIO_PIN_7, GPIO_PIN_RESET);
  /*Configure GPIO pin : ST7789_RST_Pin */
  GPIO_InitStruct.Pin = GPIO_PIN_7;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);
```

Please take some time to look at source code inside main.c as well as inside LCD/lcd.c and LCD/lcd.h files. As needed, look into source files from UGUI/. Try to understand as much as you can.

Create the necessary connections to the LCD display module as discussed above.

Before building, go to Project -> Properties. Go to C/C++ Build and then Settings. Then, expand MCU GCC Compiler and select Include Paths. Add to your paths Drivers/LCD and Drivers/UGUI.

Also, inside the Project Explorer of the CubeIDE, expand Drivers/UGUI and right-click on **ugui_sim_x11.c** select Resource Configurations -> Exclude from build… Do the same for **ugui_sim.c.**

Now, finally, build the project and program the board.

Observe operation and comment. You should see several tests running on the LCD display.

3. Lab Assignment

Create an application that uses the LCD display in the following way. Simple circles (not filled) are drawn at **random** selected locations on the LCD display. The radius of each circle should also be **randomly** selected from the interval [10, 50] pixels. The color of each circle should also be **randomly** selected. The interval of time between two consecutive circles drawn on the screen should be precisely 1 second, “measured” either with SysTick or a Timer interrupt.

4. References and Credits

[1] Textbook:
Carmine Noviello, Mastering STM32 - Second Edition, 2022, Available to purchase online:
https://leanpub.com/mastering-stm32-2nd <--- Buy from
https://github.com/cnoviello/mastering-stm32-2nd <--- Code examples