Lab1: The Beginning

This lab is an introduction to the STM32 F3 Discovery board and the tools we will be using throughout the course.

Assigned: September 21, 2015

Due: Week of September 28, 2015

Part 1

Hello World!

In this section, youll set up the tools and library code and build the existing demo project.

Tools

For building our projects we will use an embedded ARM toolchain paired with tools specific for interfacing with the Discovery board. The main programs we will be using are:

1. *arm-none-eabi-gcc*, the GNU C cross-compiler for embedded ARM processors.
2. *STLink*, tool to communicate with a ST microcontroller and program it.
3. *OpenOCD*, tool to remotely interface with microcontroller and allow remote debugging.

These tools are used by the provided Makefiles for each project. You should not ever have to use these tools directly, instead interfacing with them with `make program` or `make debug`.

The toolchain has been designed to be run on Ubuntu 14.04. We suggest installing the tools in a virtual machine if you don't run Ubuntu natively. Ask the TAs for a prebuilt VM. If you wish to install the tools yourself, clone the git repository here and run the setup script. The detailed instructions for installing the toolchain can be found here.

Get the Code

You should have already cloned the Github repository to your local drive (if you used the prebuilt image this is in `/green-electronics`). If not use the following command to clone the repository `git clone https://github.com/ndanyliw/green-electronics.git`. Then enter the directory and you should find the Lab 1 starter code. You are encouraged to look around the directories especially the Lab 1 code and the provided libraries.

Part 1 continued on next page...

Add the Tools

Before being able to use the EE155 toolchain, you will need to make sure that they are installed on your PATH. To do this run `source ~/ee155_tools/env.sh`. You will need to run this command every time you start a new development session. If you want it to automatically be run when you log in you can add it to your `.bashrc` file.

Connect the Hardware

At your station you should have a Discovery board as well as a breakout board. Plug the Discovery board into the breakout board and then connect the board to your computer. Use the USB connector labeled 'STLink'. The board should power on and will begin running the last program it had loaded.

For debugging you can start a GDB session with `make debug` and step through your code.

The breakout board has several pin headers for selecting functionality. Make sure that the 5V level shifter is populated and the instrumentation amp reference is set to GND.

Program the Hardware

To program the hardware go to the project directory and make sure the development board is plugged in and connected to the VM. Then in a command prompt type `make program` to compile the code and upload it to the board.

The LCD should display "Hello World!" and the LEDs on the board should flash.

If you want to step through the code in GDB you can instead type `make debug`. Try stepping through the code while it runs on the microcontroller.

Part 2

Stopwatch

Write a new program that works as a stopwatch:

1. Elapsed time is shown on the LCD in the format minutes:seconds.millseconds (initially 00:00.000)
2. One button starts the stopwatch
3. One button stops the stopwatch
4. One button resets the stopwatch

To make a new project copy the project_template folder and rename it. Then you can place your code in the src and inc folders. You may have to update the relative path to the EE155 libraries in the Makefile if you copy the directory in a different location than the `green-electronics` folder.

Part 2 continued on next page...

Implementation Hints: Take a look at the ge_timer.h to see what functions are available to you when creating a timer for your stopwatch. If youre curious, take a look at ge_timer.c to see what the functions do.

Use timer_register and timer_start to register a function to be called every millisecond. Update the current time in this function and display it on the LCD.

For buttons, take a look at ge_gpio.c. Look at the lab1 Hello World program for an example on reading in a digital input.

Poll the buttons periodically to check if each button has been pressed. See the Hello World demo and library_snippets.c for examples of how to use the user interface and timer parts of the library.

Part 3

Pulse-Width Modulation

Write a new program that produces a PWM signal on one of the timer output compare pins which is accessible on the header at the top of the controller board:

1. One button increases the duty cycle
2. One button decreases the duty cycle
3. Display the current PWM level on the LCD

Choose an output pin on the header. Connect an LED from the output pin to ground via a 330 resistor to limit the current to less than 10mA. Choose a PWM frequency which will cause the LED to appear to dim without flickering. Use one of the timer PWM modes rather than turning the output on and off in software.

Observe the PWM signal with an oscilloscope. Demonstrate that you can make the output constantly low, constantly high, and a wide range of duty cycles in between. Measure the PWM frequency and verify that it is the same as what you calculated for your chosen timer configuration.

Implementation suggestions: Before writing any timer code, just turn the LED on. This will verify that your wiring and output configuration is correct.

The PWM signal is generated by built-in timers in the microcontroller. First pick a timer and output pin, then determine what registers you need to set and what values you will set the registers to.

Look at the included PWM library for helper functions for setting up the appropriate timer and pin registers. Look at the library documentation and included datasheets to see what pins are available. Note that the library only implements a small subset of the PWM capabilities of the STM32F303. For more complicated operations take a look at the user manual to see what is possible.

Part 4

Extras

These are not required for this lab, but will give you a head start on functionality that well use in the future.

Voltmeter

Use the analog-to-digital converter to measure a voltage. Be careful about the range of safe input voltages on the microcontrollers pins: 0-3V only! Well look at ADCs in more detail in lab 2.

Signoffs

Stopwatch

1. Start, stop, and reset buttons work properly. Resetting while running has reasonable and predicted behavior.

Pulse-Width Modulation

1. Adjust duty cycle with buttons
2. Show PWM signal on the oscilloscope. The duty cycle matches the displayed value.
3. The PWM frequency is as predicted.
4. Demonstrate 0-100% duty cycle.