ECE 3710 Lab 2 – Fall 2017

Due Date: Week of September 25 at the beginning of your lab section (20 points)

Objectives

The purpose of this lab is for the student to gain experience interfacing the microcontroller with external peripherals using GPIO. In addition, the student will learn how to use delay loops and verify their accuracy using a logic analyzer.

Overview

For this lab, you will be writing a ten-bit binary counter in assembly that increments at a frequency of 2 Hz (0.5 seconds). The value of the counter will be displayed on an LED bar graph. Three buttons will be used to start, stop and reset the counter. The program will be run on your microcontroller and it will interface with an LED display and switches using GPIO ports. All timing requirements will be verified using a logic analyzer.

Preparation

1. Prior to lab, please read chapters 2.2 and 4.2.2 in volume one of your textbook

2. Read this document in entirety

3. Come with the following:
   a. Breadboard (1)
   b. Tiva C Series LaunchPad Evaluation Kit
   c. 10 LED bar graph (1) – Available for purchase at the ECE store
   d. Push Button (1) – Available for purchase at the ECE store
   e. Bar resistor 220 Ohm (1) – Available for purchase at the ECE store
   f. Standard 220 Ohm Resistor (1) – Available for purchase at the ECE store
   g. Leads for the voltage source
   h. Jumper wire – Free in a box on a counter by the ECE store (Note that ribbon cable of various sizes is available for purchase at the ECE store – it can make connecting to the uController much easier and it is reusable.)

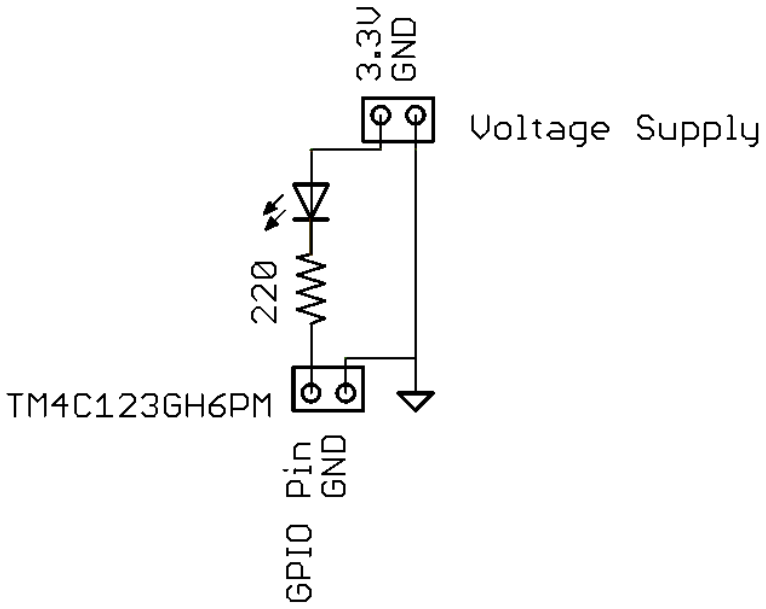
Procedure

1. LED Display

    a. The LED display will be connected to your microcontroller via a breadboard; power for the LEDs will be supplied by an external voltage source.

    b. Connect one ground pin from your microcontroller and the voltage source to the breadboard (this allows the voltage source and the microcontroller to share a common ground).

    c. Read over the datasheets for the ten LED bar graph and bar resistor to determine how to connect the LEDs to the microcontroller (a single LED from the display is depicted below); you will need to decide which ports/pins to use. Note that this is a 'sinking configuration' (i.e. the microcontroller is sinking current instead of supplying it). When connecting the LED display, use an active-low configuration (0 [low] means active [LED on], 1 [high] means inactive [LED off]).



    d. ***Caution!*** *In general, powering external hardware through the microcontroller runs the risk of drawing too much current and then damaging the internal components. Please give careful consideration to power and current requirements of external hardware when connecting it to a microcontroller.*

2. Buttons

    a. The Tiva C Launchpad that you are using only has two internal buttons available so a third button will need to be mounted on the breadboard. The button labeled "reset" on the Tiva board is not usable for your project – it serves as a power reset for the board. The two internal buttons and the one external button will serve as inputs for the counter program.

    b. Connect the external button to a port/pin of your choosing. Look over the first schematic in Appendix A of the board's user guide to understand how the on-board buttons work so that you'll know how to configure your port/pin. A datasheet for the external button can be found on the wiki.

3. Software Requirements
    a. The counter will count at 2Hz +/- 5% as verified by the logic analyzer.

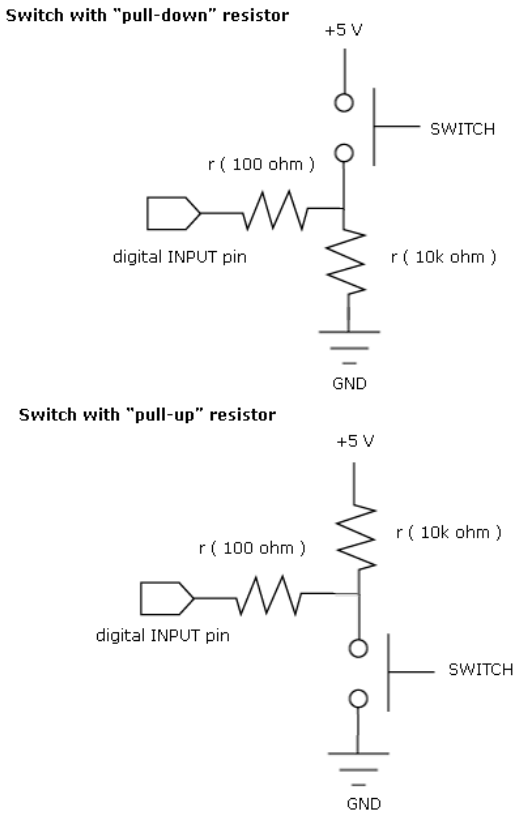
    b. When the stop button is pressed, the counter will pause with the current value still visible.

    c. When the start button is pressed, the counter will resume from the current count.

    d. When the reset button is pressed, the counter will start counting from zero.

4. Pass-off – Show the functioning system to the TA

Definitions



- Tri-state. When the GPIO pin is disabled, it enters a high-impedance state, also known as tri-state.

- Push-pull. When the pin is write-enabled without the open drain or other options enabled, it is in a push-pull configuration. The pin will source current or sink current depending on its value.

- Open drain. A configuration that works well as part of a bus because it allows multiple devices to communicate on the same line. The configuration will cause the device to sink current when active or remain in a high-impedance state when inactive.

- Pull-up vs pull-down resistor. The difference between the two configurations is the placement of the switching device; see the image. The pull-up resistor is more efficient at supplying current, while the pull-down resistor is more efficient at draining current. Read Introduction to Arm Section 9.5 for more details.

Documentation

The report must include the following:

1. Logic analyzer screen shot that shows the delay using the markers.

2. Description of your software design. Use the pseudocode program header and comment style discussed in class and practiced in the homework assignment.

3. The code of the program will form the final addendum to the report.

4. Schematic of your physical design, noting port/pin configuration.

5. Be sure that your reports meet all other criteria in the "Guidelines for ECE 3710 Lab Reports" section of the course wiki – you can find it under lab supplementary materials.

Hints and Tips

1. Sections 10.3—4 and Table 10.3 of the microcontroller datasheet will be useful for help in configuring the ports/pins (NOTE: the Advanced Peripheral Bus is used in the course notes).

2. Use Sections 2.1.5 and Appendix A of the user guide to figure out how uC ports/pins are mapped to the board's headers.

3. The recommended configuration for this lab is open-drain w/pull-up resistor while input pins should use a pull-up resistor. You must be able to explain/justify the port/pin configuration in your lab report (that is, if you use the recommended setting then you need to be able to tell us why it works the way it does).

4. Pins C0-C3 are JTAG connectors, and therefore unavailable for GPIO.

5. Certain pins may be locked, which will cause errors during configuration. Two of these pins are PD7 and PF0. You can tell if your pin is locked if the corresponding GPIOCR register value is 0. Unlocking a GPIO allows the GPIOCR to be written to, which will allow configuration of the pin.

6. Unlocking is performed by writing a specific 32-bit value to the GPIOLOCK register. Look in the datasheet and Introduction to ARM Section 4.2.1 (I/O ports) for more information.

7. Writing to a GPIO when it's locked causes a hard fault. To verify a GPIO pin is locked, view the GPIOCR and GPIOLOCK registers in the debug mode System Viewer.