CPE 301 – Embedded Systems Design Lab

Lab # 07 – Debouncing, 7-Segments, EEPROM

Fall 2020

Objectives:

1. Demonstrate methods for debouncing GPIO input.
2. Demonstrate methods for driving parallel output.
3. Demonstrate methods for storing non-volatile data in EEPROM memory.

Required Equipment:

1. Arduino Mega 2560
2. USB programming cable
3. Laptop or Lab PC with Arduino IDE installed
4. 330 Ohm DIP Resistor
5. 7-Segment LED
6. Push-button
8. Breadboard
9. Jumper Kit

BEFORE THE LAB: Read the section on AVR memory in the 2560 datasheet starting on page 20.

Procedure:

1. Connect a push-button and a 7-segment LED to you Arduino Mega 2560.
    a. Wire the push button to use as an input to the Arduino Board
    b. Wire the 7-segment to a single GPIO port such that writing an 8 bit value to the port will result in a particular character being displayed on the 7-segment. Don't forget to use a buffer chip to protect the GPIO port of your Arduino.
    d. Include in your report a picture of your circuit.
2. Write a program to increment your 7-segment display by one character each time the button is pressed.
    a. Write the program such that it rolls-over from F back to 0.
    b. Make sure your program properly debounces the signal such that the display only increments once for each button press.
    c. The code should not contain 16 if statements or switch statements. Write your code with efficiency and maintainability in mind, refer to the timer lab for hints.
    d. Write your program to use the 2560's onboard EEPROM memory to retain the previous display position after a power cycle.
        i. For example, if the 7-segment is displaying '4' before a power cycle, it should display '4' when you power it back up.
        ii. The program only needs to store one byte at a chosen address, and then read that byte back when the program starts.
        iii. Write a function which stores a byte at an address, and a function which returns a byte from an address.
    e. Include your program code for submission

Step-by-Step Procedure

1. **Correctly set up your input**
    a. Use DDR/PORT/PIN registers
    b. You can either enable/disable internal pullup resistors
        i. If you enable internal pullup resistors - One end of switch goes into the GROUND and the diagonal end goes to the INPUT pin of the arduino (that you set using the ddr ports)
        ii. If you disable internal pullup resistors - Wire up the switch like in Lab 1 (using pullup resistors)
    c. Debug if input works using the Serial print statements / Oscilloscope (you should observe changes in the output with pressing/un-pressing buttons)
2. **Set up the debouncing code**
    a. See demo
3. **Ensure that the EEPROM save works.**
    a. Integrate the debouncing code with the EPROM sample code.
    b. Test it -
        i. Save counter value in EPROM
        ii. Reset the Arduino (the button on the board)
        iii. Read back the counter and show on Serial print
        iv. If it shows the previous value, it works.
4. **Set up the 7-Segment Display**
    a. Start by setting pin # 3 and 8 to resistor. Then resistor -> GND.
    b. Test out each LED by putting Vcc (power) through every pin. If it lights up, then you know which pin is what.
    c. Fill up the LED port bit array in the code.
    d. Hook up the 7-Segment Display to the Arduino OUTPUT ports.
    e. Based on counter value, change output on 7-Segment display
6. **Test end-to-end**
    a. Press button
    b. Check if increments correctly
    c. Check if Saves to EEPROM correctly (reset arduino)
    d. Check if incrementing values result in the proper 7-segment output.
7. **Submit Code and picture of circuit.**