CPE 301 – Embedded Systems Design Lab

Lab # 07 – Debouncing, 7-Segements, EEPROM

Fall 2018

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
7. 8-Bit Buffer (74LS373 or 74LS240 or 74LS244)
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 be active low using the internal pullup resistor.
   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.
   c. Include in your report a table showing the 16 hex characters and the 16 byte values necessary to display the hex characters on the display as you wired it.
   d. Include in your report a circuit diagram showing how you wired 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 in the lab report following the code submission guidelines listed below.

f. In your lab report, detail how your program works and what steps you took to build the circuit, write the program, and test each step along the way. This description should be written like a tutorial for someone else who wishes to build a similar system.

Code Submission Guidelines

Code should be submitted inline within the text of your report, such that it retains syntax highlighting similar to that of the Arduino IDE. There are several ways to perform this, so feel free to find a method that works for your system. On Microsoft Word, follow the directions below:

1. Install the Code Format Addin.
2. Copy the code from the Arduino IDE, and paste into Word.
3. Select the code, and click the convert button to convert the text into formatted text.

This will result in the code appearing as below. Submissions without proper code formatting will not receive full credit.

```
// CPE 301 - REGISTER-LEVEL Blink Example

// Define Port B Register Pointers
volatile unsigned char *port_b = (unsigned char*) 0x25;
volatile unsigned char *ddr_b = (unsigned char*) 0x24;
volatile unsigned char *pin_b = (unsigned char*) 0x23;

void setup()
{
    // Initialize the Serial Port
    Serial.begin(9600);
    //set PB4 to INPUT
    *ddr_b &= 0xEF;
    // enable the pullup resistor on PB4
    *port_b |= 0x10;
}
void loop()
{
    // if the pin is high
    if (*pin_b & 0x10)
    {
        Serial.println("PIN IS HIGH");
    }
    // if the pin is low
    else
    {
        Serial.println("PIN IS LOW");
    }
    delay(250);
}
```