-ECE: 09.342-3-

Milestone 1: Communicating with Will Byers

Marc Giordano and Kevin Miskovich
Rowan University

October 16, 2017

Abstract

An MSP430F5529 microprocessor is implemented to display an LED output dependant on the string of hexadecimal values received through UART ports. Many of these implementations will be strung together in order to replicate the communications light wall from the show “Stranger Things.” This is possible due to pulse width modulation, or PWM. This allows us to send messages over waves and change wavelengths that can alter different aspects of a circuit, such as changing the color of an LED. UART, which is used in conjunction with PWM for this lab, allows us to communicate between boards. This project is highly dependent, not only on our code, but on the code of others, as it is to function as a single circuit in the end. If even one code is incorrect or circuit is built improperly, then the whole unit will fail. This lab, as well as teaching us how to code and work with UART, also teaches us to work together, and communicate with other groups to achieve a common goal.

Introduction/Overview

Do you want to make a difference in the world? A wise man once taught us that if you can blink an LED, then you can control the world. We, as ECE students, firmly believe in this notion and use it as a drive. This lab is the perfect metaphor of the literal power at your fingertips.
The objective of this lab was to create a circuit, using both hardware parts and software coding, that allows for a RGB LED (Red, green, blue) to light up according to how we program it to, as well as being able to light up in a string of other circuits similar to this one. This required several parts including a breadboard, Mosfets, RGB LED, resistors, a microprocessor of your choice, and a power source. These parts, in conjunction with the correct code on Code Composer Studio, caused our LED to light up whatever color or sequence we program it to.

The Code

The first step in creating a working circuit is to have working code. When we created our code, several crucial parts were needed for the circuit to properly function. Defining the LEDS and pins, as well as using interrupts were standard. Other functions, including PWM and switch statements containing the cases in which the LEDs would light up, were needed as well.

PWM

Immediately after defining our pins and LEDs, we set up the PWM, or pulse width modulation. This section is crucial for the rest of the code because it sets the clock for use with the LEDs, as well as determining which type of clock is used in our code. We chose to use the sub-main clock(SMCLK) and put it in up mode, which causes the clock to count up to a predetermined number (CCR0) which we decided, in this case it is 255. The initial values of the three LEDs are also set to 0 in this section.

```
//PWM TIMER SETUP
TA0CCTL1 = OUTMOD_7;    //SET/RESET MODE RED
TA0CCTL2 = OUTMOD_7;    //SET/RESET MODE GREEN
TA0CCTL3 = OUTMOD_7;    //SET/RESET MODE BLUE
TA0CCR0 = 255;          // Full Cycle
TA0CCR1 = 0;            //Red set at 0%
TA0CCR2 = 0;            //Green set at 0%
TA0CCR3 = 0;            //Blue set at 0%
TA0CTL = TBSSEL_2 + MC_1 + ID_2; //SMCLK/4, up mode
```

Figure 1: Snippet of PWM from code

UART

When it came to the UART section, we relied heavily on the example code used in lab 1. We were instructed to do so because we had not learned about UART extensively at the time. Once the UART section was implemented, we went back and tried to study and understand it line by line, as to be more prepared for future assignments involving UART. We were able to understand it enough to implement what we need for this function to communicate with programs such as Putty and RealTerm.

State Machine

Within the interrupt protocol, we implemented a switch statement in order to control the behavior of the program for each individual case. For safety precautions, we set a default case to break in case faulty data slips in. Our first case is that of no data being received when the interrupt flag triggers; in this case the program will break out as is the case with the default.

The next case handles the instance when a UART input flag is triggered (USCI_UCRXIFG). As soon as this case is met, the following switch statement executes:

```
switch(bitcounter)
{
    case 0:
        while(!(UCA0IFG & UCTXIFG));
        UCA0TXBUF = UCA0RXBUF - 3;
        __no_operation();
        break;
    case 1:
        TA0CCR1 = (UCA0RXBUF); //SETS RED SECTION
        break;
    case 2:
        TA0CCR3 = (UCA0RXBUF); //SETS GREEN SECTION
        break;
    case 3:
        TA0CCR3 = (UCA0RXBUF); //SETS BLUE SECTION
        break;
    default:
        while(!(UCA0IFG & UCTXIFG));
        UCA0TXBUF = UCA0RXBUF;
}
```

Figure 2: State Machine for UART Input Values

The switch statement displayed in figure 2 controls the behavior of the program depending on the state of the machine. The machine uses a variable ‘bitcounter’ that begins at 0 and increments using the if statement after the switch statement. When the variable is 0, as long as we are not transmitting data, then the transmit data will be 3 bytes less than the received data. At case 1, we set the first capture compare register to the received data (Red LED). At cases 2 and 3, it is the same with the green and blue LEDs respectively. The default of this switch statement is to transmit the incoming data to the next board. From the if statement after this state machine, the variable will only reset to 0 once the final byte is received.

Implementation

Processor Selection

The processor selected for this implementation is the MSP430F5529. This board was narrowed down by eliminating the other four boards at our disposal:
MSP430FR2311: Memory count is too low.
MSP430G2553: Programming issues. Personal preference, harder to program.
MSP430FR5994: Do not need the super capacitor. Waste of resources.
MSP430FR6989: Too many costly features, such as LED screen. Another waste of resources.
From these decisions, it was decided to use the MSP430F5529.

RGB LED Circuitry

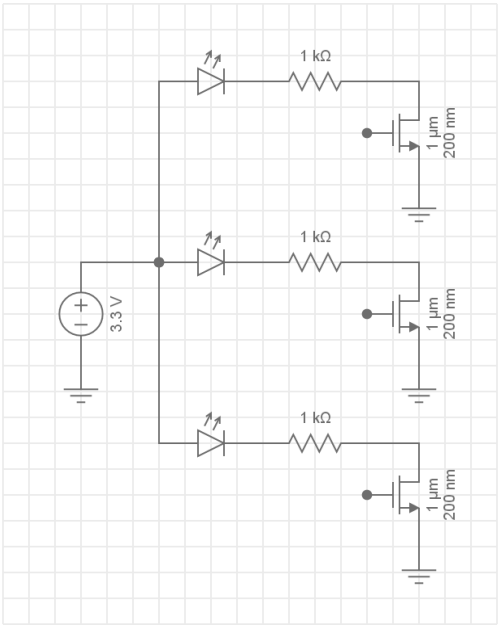

Figure 3: RGB LED Circuit Design

The circuit used to power the LED is displayed in figure 3. The 3.3Vdc power supply is symbolic of the MSP430F5529, since that will be the source of power for the circuit. Since the RGB LED supplied has a common anode configuration, that means the anodes of the three LEDs are tied together while the cathodes each have an output pin. The output of each of the cathodes were paired with 1kΩ resistors in order to limit the current through them. The mosfets are there in order to control which LED is on because the gate of the mosfet is tied to each of the PWMs from the code (no model in the SPICE program for that). This will only allow current to flow through the LEDs when the code allows it.

Usage

In terms of using the program, the user controls it through a UART program. Since we are using programs such as Putty and RealTerm to send data, that is how the data is to be sent. The interrupt is what controls the behavior of the program, which is explained in-depth above. The switch statement within the interrupt loop controls the LEDs individually, depending on the bytes that are input into the system. Say there are 80 bytes in the string; the first processor will take the 80 bytes, subtract 3 to use for its outputs, then send the rest of the 77 onto the next processor through its UART ports. As stated, this does rely heavily on the other teams' boards.

Issues/Errors

While we were able to complete the lab, it was not without our fair share of issues throughout. The biggest issue we had was making our code compatible with RealTerm, to the point where the LED would light up. One issue we still have is that we need the UART cable to have our circuit properly function, as it would not work with just the standard USB.

References

We worked very closely with another group consisting of Mike Guiliano and Tyler Brady. We helped each other with both the code and hardware parts of the lab.

MSP430F25529 Datasheet:
http://www.ti.com/lit/ug/slau533d/slau533d.pdf
Milestone 1 Repository:
https://github.com/RU09342/milestone-1-communicating-with-will-byers-i-miss-phil-mease
Baud Rate Calculator:
http://processors.wiki.ti.com/index.php/USCI_UART_Baud_Rate_Gen_Mode_Selection

Appendix

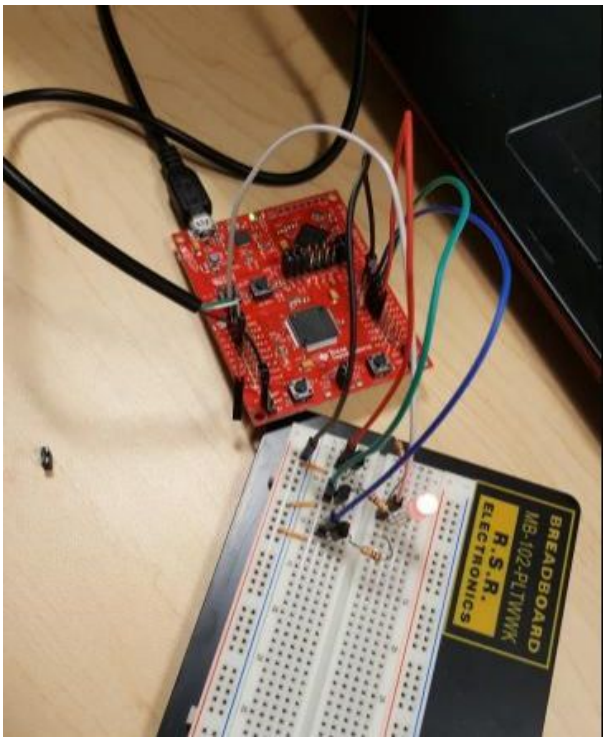

Figure 4: Hardware Circuit with LED