– ECE 09.342: Milestone 1 –

The Greek Freaks: Stranger Things LEDs

Colin Craig and Kevin Beenders
Rowan University

October 22, 2018

1 Design Overview

The first milestone project focuses on everything that has been covered in the first six weeks of an introduction to Embedded Systems course. This includes how to use GPIO pins, implementing timers and interrupts, utilizing pulse width modulation (PWM), and communicating with microcontrollers using UART. By using the skills learned during the semester so far, the group was able to deliver a software package that can be used to send a packet of ASCII characters through another computer with UART to change the color scheme on an RGB LED circuit.

1.1 Design Features

- 9600 BAUD Rate UART Communication
- Addressable RGB Node
- 256 steps of PWM brightness
- 3.3 V source to High-Side Switch

1.2 Featured Applications

- Stranger Things LED communication with Will Byers
- High-to-Low Powered RGB LED driver

1.3 Design Resources

https://github.com/RU09342-F18/milestone-1-greek-freaks

1.4 Block Diagram

Figure 1 is included in this section.

1.5 Board Image

Figures 2 and 3 are included in this section.

2 Key System Specifications

For this project, some specifications included to have a range of message length from 0 to 100 bytes. This is to account for the maximum number of nodes that will be in the chain. Regarding the values of each RGB LED, the maximum hexadecimal value that each LED can hold is 0xFF which is 255 in decimal. This is the maximum brightness of the LED and is limited by the CCR0 value. This is also the maximum value of a 1 byte number.

3 System Description

3.1 Highlighted Devices

There are two main devices used in this project. One, the microcontroller itself which does all of the initialization, interrupts, and drivers for the PWM. Second, the driving circuitry for the LEDS on an external breadboard.

3.2 MSP430G2553

The MSP430G2553 is where the bulk of the project is performed. All of the buffers, initialization, and drivers are performed through this device.

3.3 High-Side Switch

Once the MSP430G2553 drives the PWM to each port, the high-side switches take care of driving the proper current to the load or LED. Current limiting must be done to ensure the proper amount of power is being dissipated in the LED. If too much goes through, the LED can ”burn out” and fail to work. Proper current limiting should be done for any load because we do not want too much current to be supplied. Too much current causes bad things to happen in a load and that load isn’t always as simple and cheap as a RGB LED.

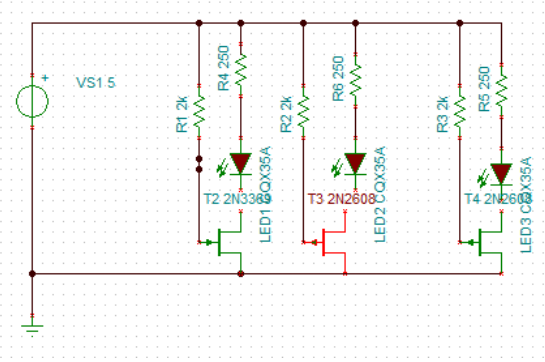

Figure 1: Flow diagram of project code (W/O RGB Color Check)

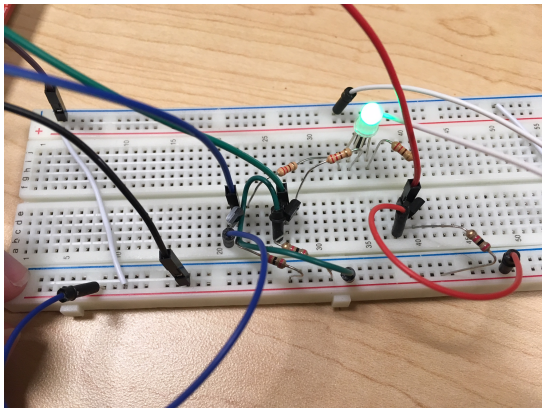

Figure 2: High side switching circuit

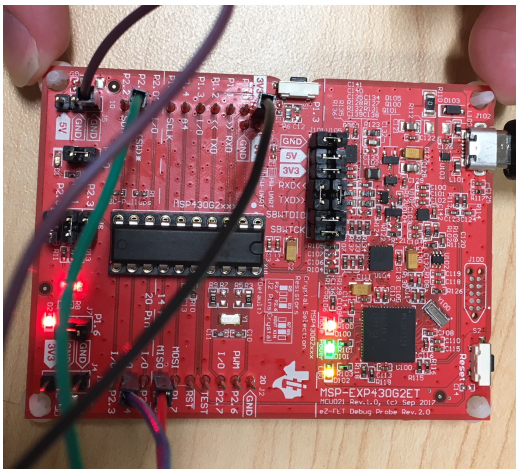

Figure 3: Top of G2553 with connected pins.

4 System Design Theory

For this design, multiple components came together to give the proper implementation of the design, which was to address an RGB node in a sequence of ASCII characters sent through a message in UART. To give a general overview, multiple components have to be initialized first before getting the message. Once the components become initialized, the node must be able to receive the serial data and store it into a RGB buffer. The only needed data is the first four bytes. Everything after that must be sent in a separate buffer and transmitted to the next node. Once the data is in the RGB buffer, it must drive three PWM drivers for each respected LED color. These drivers are then sent out through specific ports to an external breadboard. Each port is connected to it's own high-side switch connected to a 3.3V source via the microcontroller. The output of the switch drives each pin for the RGB LED.

4.1 Initialization

Initialization is needed for the processor to understand which parts of the microcontroller need to be enabled and for what purpose at startup. The first step is to initialize the pins that are to be used on the board. In the case of this Milestone, three pins (P1.6, P2.1, and P2.4) will output each of the three PWM drivers. The second step is to initialize the UART transfer and receive buffers on the microcontroller as well as setting the baud rate. For this lab, the group used a baud rate of 9600, which is rather slow in the grand scheme of things but fast enough for the purpose of this lab. The final part of the initialization process is setting up the timers which will be needed for the PWM to drive the LEDs. Once this is all complete, Low Power mode is enabled until an interrupt, in this case, when a UART message is received, an interrupt is sent to begin the RGB buffer.

4.2 RGB Buffer

Once the UART interrupt is enabled, a case statement breaks up the first four bytes received in the message or packet. The first case or byte is to set the receiving buffer equal to the size of the packet. The second byte is then set equal to the capture/compare register that drives the PWM of the Red LED. The third and fourth bytes are set to other capture/compare registers which drive the PWM for the Green and Blue LEDs.

4.3 Transmit Buffer

Once the first four bytes are processed, the remaining bytes get sent into a transmit buffer. This is the data that must be sent to the next node in series. The difference between the receiving and transmitting buffer is 3 bytes. So, this transmit buffer can be done in one line by subtracting 3 from the receiving buffer. This will then give the new size of the packet to the next node. Once this new packet is sent in the transmit buffer, the buffers are then cleared to receive the next packet.

4.4 PWM Driver

Three PWM drivers are needed for each corresponding LED. For this project, SMCLK was used in up-mode. This means the timer counts up to a specific CCR value and sends an interrupt on its falling edge. Though, by having a second CCR, we can send two interrupts on different edges. This creates the PWM. By setting the second CCR to the byte in the receiving buffer, the PWM of that LED can vary based on the serial data sent through UART. The hardware built in the timer modules is used to directly control the PWM. This is done by setting the capture/compare control register to be in set/reset mode or OUTMOD 3. Each capture/control register is built into specific ports on the board. This is why we chose the GPIO ports chosen. The G2553 is ideal for these circumstances because the receiving and transmitting ports are no the same as the timer ports. The FR2311 was looked at as a candidate for this project until it became known one of the timer ports was the same as the receiving port. This would have required a software PWM implementation to move the driver to another port. This would have required too much work for such a simple process that can be completed by the G2553.

4.5 High-Side Switch

Each port driven by a PWM is connected to a breadboard via a jumper cable. To accommodate for a higher current output on the LED, we used three high-side switches. Each switch was connected to a 3.3 V source located on the microcontroller. On the output of each switch was the specific input on the RGB LED. For this task, a high-side switch was not needed. Though, it can be used to drive a load requiring a larger voltage/current from a secondary source such as a car or 5 V battery.

5 Getting Started/How to use the device

To use this project on your device, a couple things are needed. One, a laptop with Code Composer Studio installed is needed. This allows your personal device to communicate to the microcontroller. Also, a breadboard is needed with three high-side switches with the LED as the load. If you wanted the device to transmit needed data, multiple devices would need to be connected. To just get this specific device working, Realterm is a recommended program to test the individual node. Once the program is debugged and running, open the com port connected to the microcontroller and send hexadecimal values in the format of 0x04 0xFF 0xFF 0xFF. The first byte indicates the size of the packet and the other three indicate the brightness of the three LEDs. For that example, FF is the highest value which would create a corresponding white light.

6 Getting Started Software/Firmware

6.1 Hierarchy Chart

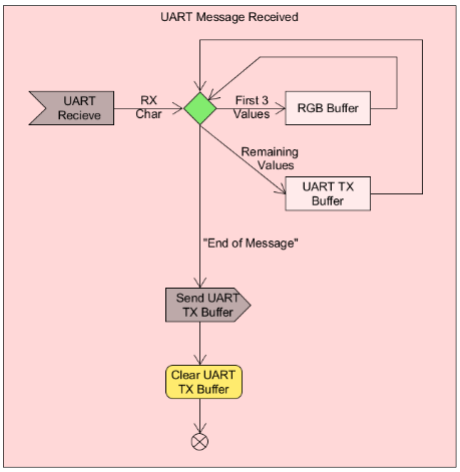

Figure 4: UART Message Protocol

6.2 Communication with the Device

The device communicates with a computer using a USB to UART cable. The cable allows the computer to use serial communication and send a string a bytes to the device. The device then processes that string of bytes, and then will send an updated string back to the computer that can be seen using a program like Realterm. The device is programmed to take the first three instructions bytes, so that is sent back is

a byte that details how many bits are left in the string. So if the string 0x06 0x00 0xFF 0xFF 0xFF 0x00 0x00, what will be sent back is 0x03 0xFF 0x00 0x00.

6.3 Device Specific Information

Specs for MSP430G2553:

- 3.3 VCC source on microcontroller
- 6 mA max source/sink current through a single pin
- 4 Capture/Compare register
- Low power mode current of .5 micro Amps
- 1 MHz internal clock

7 Test Setup

To test the design, Realterm, CCS, and the G2553 plus the high side switching circuit are used together run the circuit. The computer and the embedded circuit communicate via a USB to UART cable. To test the device, the microcontroller plus circuitry was connected to the computer by a USB to UART cable. As well as having the connections between the used GPIO pins on the board to the Gates of the PMOS transistors on the circuit.

7.1 Test Data

The test worked as it should, when sending different packets of information, all the tests when performed gave the correct output. The system firstly processes the first three instructions bytes properly to the switches, but also sends back only the next bytes and the byte that describes the number of bytes left in the instruction.

7.2 Bill of Materials

- 3 550 Ohm resistors
- 3 2000 Ohm resistors
- 3 2N2000 PMOS transistors
- breadboard
- USB-UART cable
- Computer with CCS and Realterm
- MSP430G2553 microcontroller