Milestone 1: Addressable LED

Skylar Adams
Rowan University

October 22, 2018

1 Design Overview

For this milestone, students were required to design code for any of the MSP430 architecture to create an addressable RGB LED. The code must be able to take a large byte string, detect and set the three RGB values for the LED, and transmit the rest down the chain. In this case, the MSP430F5529 was the chosen board, and UART was used for communication between boards. Along with the code to set the proper PWM values of the LED, a low side switch was also designed using N-channel MOSFETs to ensure the LED stayed off while the LED was not receiving power.

1.1 Design Features

The key design features include:

- MSP430F5529 for PWM and UART
- Low side switch to prevent wasted power via flickering LED
- Utilization of the MSP430 timer peripheral

1.2 Featured Applications

A few applications of an addressable LED include:

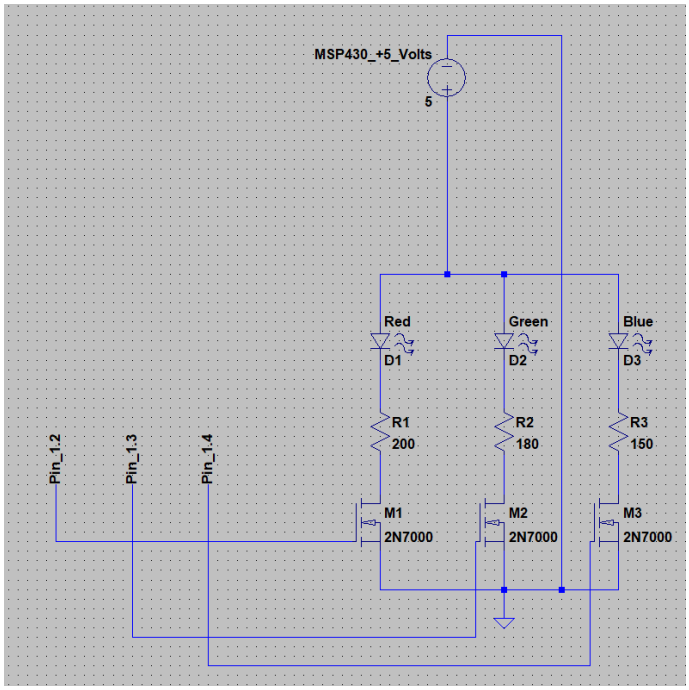
- RGB LED strips
- PWM control of devices
- UART communication

1.3 Design Resources

The github link for the design folders and code can be found here:
https://github.com/RU09342-F18/milestone-1-the-board-with-no-name

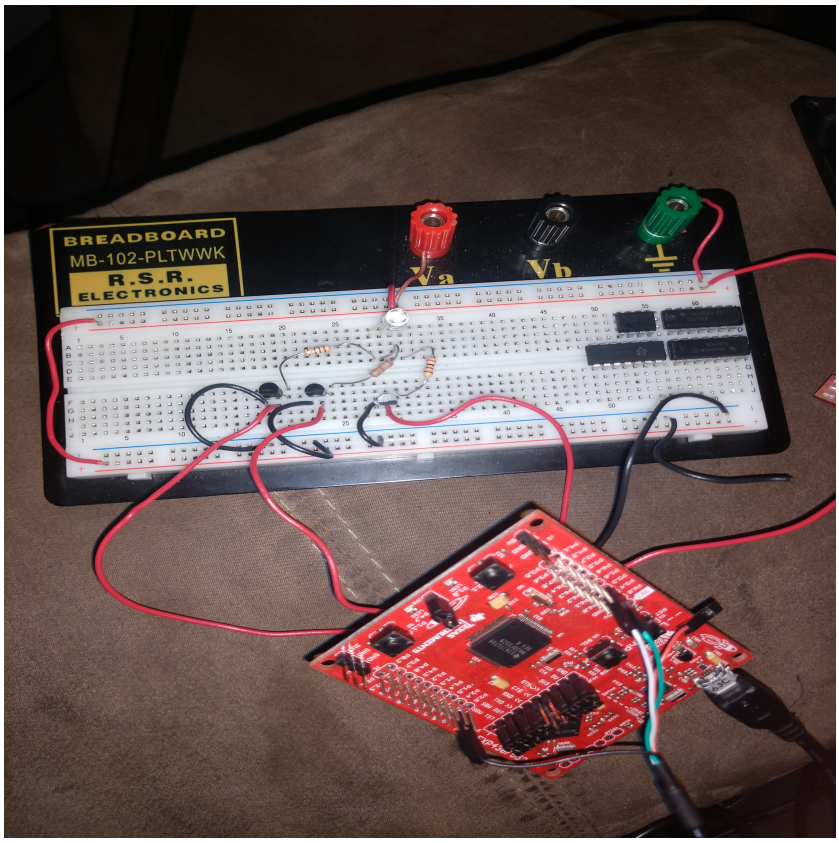
1.4 Block Diagram

Figure 1: Simple Block Diagram



1.5 Board Image

Figure 2: Image of Board



2 System Description

The main problem to be solved in this milestone is to create an addressable RGB LED, which can change RGB values via data transmitted through UART, and connect to a chain of LED circuits to be able to read a long data string which will set the PWM values on all chained LEDs. This was solved by using the MSP430F5529 microcontroller. Another problem that was kept in mind was power consumption, and the system itself uses both a low side switch and low power mode code to solve these problems.

2.1 Detailed Block Diagram

The main difference here between the previous block diagram is the level of detail. This should show specific pins on the device and voltage levels. Don't go so far as putting in pictures of how the peripherals are connected to each other inside of the controller, but enough that you could basically put this thing together again.

2.2 Highlighted Devices

The Devices used for this project was:

- MSP430F5529 for PWM and UART
- LED driver circuitry

2.3 MSP430F5529

The MSP430F5529 was used for both generating PWM signals for the RGB LED, and for UART communication. On the board, the Timer A peripheral was used to create hardware pwm signals for each of the LEDS, when the data was sent via UART. Timer A was initialized using the SMCLK on the board, which oscillates at a frequency of 1MHz. This device was chosen, mainly due to the fact that it is one of the cheaper boards, so the project could be manufactured at a lower cost. The PWM signal sent to the LED was an approximately 3.9kHz waveform, 3.3Vpp. For UART communication, the pints 3.3 and 3.4 were set to Tx and Rx respectively, and the baud rate was set to 9600. The PWM output pins of the board were pins 1.2, 1.3, and 1.4 for red, green, and blue respectively.

2.4 LED Driver Circuit

The driver circuit of the LED consisted of 3 N channel MOSFETs in a low side switch configuration to prevent the LED from flickering while no code is running. For each of the MOSFETS, a drain current of 16mA was used, due to it being the recommended current running through each color of the RGB LED for optimal use. Along with the MOSFETs were resistors to ensure the proper drain current values for each of the LEDs, due to each one having a different drop across the diode. The low side switch configuration was used, due to the LED being common anode. If a high side switch were used, the LED would be constantly on while no code was running. The LEDs anode was connected to 5V instead of 3.3V due to the large voltage drop across the blue LED. If 3.3V was used, the MOSFET connected to the blue LED would never turn on.

3 SYSTEM DESIGN THEORY

Within this project, the micro-controller was used to control an LED, and the driver circuitry was to ensure that the LED remained without power until a PWM signal was

sent to the LED. The theory with the micro-controller is the fact that they are extremely good at creating a PWM signal. Along with the fact that the students were required to use the MSP430 micro-controllers, the fact that they perform extremely well at creating PWM signals was the reasoning behind thier use. The theory behind a low side switch is that the input is off by default, hence the name low side. Due to the LED being a common anode, or the ”positive side” of an LED, and due to there being slight voltages on the output pins, even when code isn't running, the LED will tend to flicker. The low side switch's MOSFETS remain off, because the voltage isn't large enough to turn them on, so the LED remains unlit, as if it were in an open circuit.

3.1 Design Requirements

To complete the requirement of creating a PWM signal with the MSP430F5529, a hardware PWM method, using Timer A and OUTMOD settings was used. Timer A was set to up mode, with SMCLK as its source, and OUTMOD 7, or reset/set was used for the registers. Reset/set will toggle the output register on the first interrupt, CCR0, and set the value on the second, CCR1, which creates the PWM signal. The closer the value that CCR1 is to CCR0, the larger the duty cycle, and vice-versa. For the low side switch, some light circuit analysis had to be used to determine the proper values for the load resistors, and to see if the LED could be powered by 3.3 or 5V. The factors which determined these values were the current needed on the drain, and the voltage dropped by each color LED. After analysis, it was determined that the voltage drop on the blue LED was too large to be able to use 3.3V, due to the MOSFET being unable to turn on, so 5V needed to be used. While each color had the same current requirements, the voltage drop across each color was different, so 3 different resistors were needed to exactly match each colors requirements.

4 Getting Started/How to use the device

Subsection 4.1 will go over how to connect hardware components, and subsection 4.2 will go over how to set up the required software.

4.1 Hardware Setup

First, view the detailed block diagram in order to see how to properly set up the low side switch on a breadboard. The resistor values and MOSFET models are included in the schematic, as well as any extra voltages needed. Be sure when setting up to avoid accidentally plugging pins 1.2, 1.3, or 1.4 into the 5V rails, at risk of one damaging or destroying their board. Plug both the MSP430 power cable and the UART cable into USB ports, connect the white wire of the UART cable to pin 3.3, connect the green cable to pin 3.4, and connect the black cable to any ground pin on the board. DO NOT connect the red cable, for it is a 5V cable and can damage or destroy the board.

4.2 Software Setup

First, be sure both code composer studio and Realterm are installed on the computer. In code composer studio, once the board is connected via the USB, debug the project, proceed on the low power mode message, and once complete, click the resume button on the top bar to start debugging. In Realterm, under the ports tab, be sure the baud rate is set to 9600, and make sure the COM port of the UART cable is selected. To find the COM port, open up device manager (if on windows) and expand the ports tab. The COM port of the UART cable should be labelled. Once set up, navigate to the display tab and check off Hex[space] to see what's being sent to, and returned by the board. Finally, navigate to the send tab, and prepare a data string in the provided text box.

5 Getting Started Software/Firmware

This section will go into more detail on how to setup the software, mainly Realterm setup.

5.1 Communicating with the Device

To communicate with the device, the program Realterm, and a UART cable are required. To start, start up Realterm as shown in the Software Set up subsection. Once navigated to the text box, one can start sending data strings. Send all strings using either hexadecimal or binary values, with a minimum of 0, and a maximum of 255. When entering a data string, space out each part of the message, and add an end message bit, 0x0D. The hexadecimal input in Realterm is not case sensitive. For example, a single package string should look like ”0x05 0xFF 0x00 0XFF 0x0D.” Once typed in, click the send button, which will send the code to the board, and light up the LED. On the screen, the message sent, and the return from the board, ”0x02 0x0D” for a single package should display.

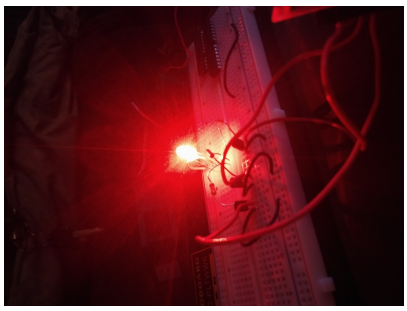
6 Test Setup

To setup the system for test, follow all guidelines in the hardware and software getting started sections. Once Realterm is open and ready to send data, use the following test strings, resetting the processor if flashed, and restarting the code in the debugger if in debug mode:

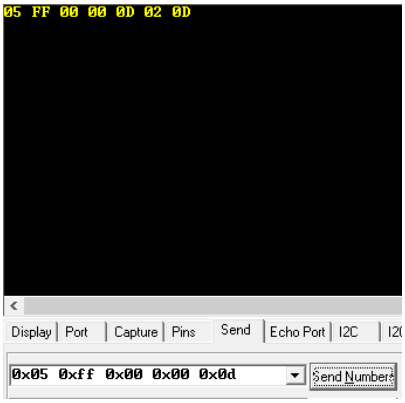
- Red: 0x05 0xff 0x00 0x00 0x0d
- Green: 0x05 0x00 0xff 0x00 0x0d
- Blue: 0x05 0x00 0x00 0xff 0x0d
- Full Bright White: 0x05 0xff 0xff 0xff 0x0d
- Half Bright White: 0x05 0x7f 0x7f 0xf7 0x0d
- Double String: 0x08 0xff 0x00 0xff 0xff 0xff 0x00 0x0d

6.1 Test Data

The following subsection includes pictures of both the LED and the Realterm return of each test data package.

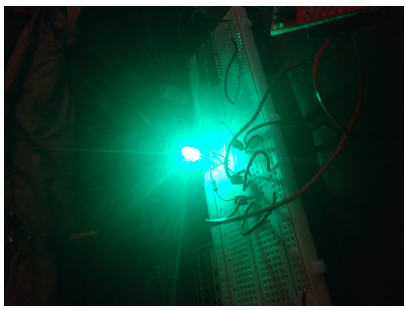


(a) Red Data Package

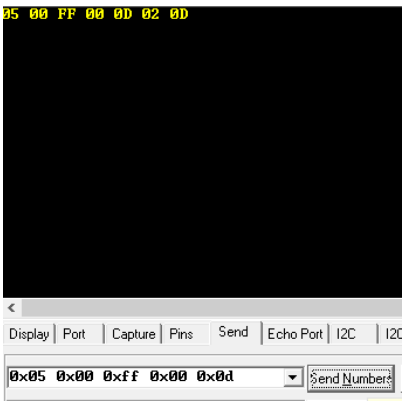


(b) Red Data Package Return

Figure 3: Red LED and Realterm Terminal

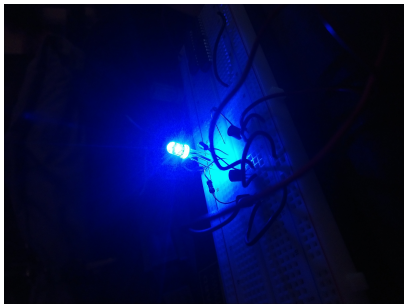


(a) Green Data Package

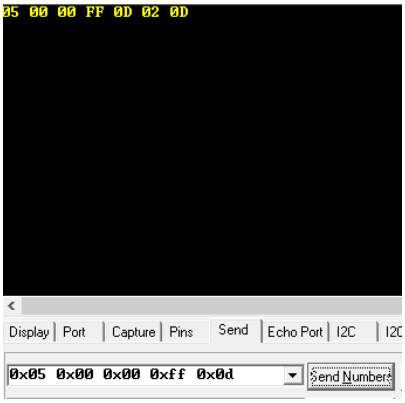


(b) Green Data Package Return

Figure 4: Green LED and Realterm Terminal

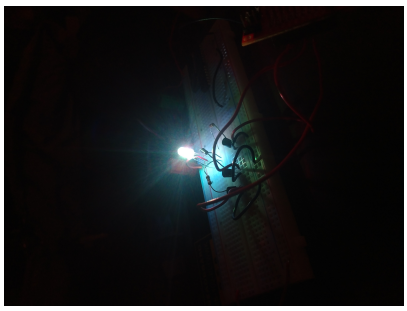


(a) Blue Data Package

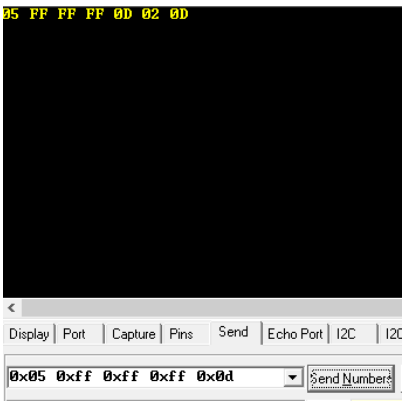


(b) Blue Data Package Return

Figure 5: Blue LED and Realterm Terminal

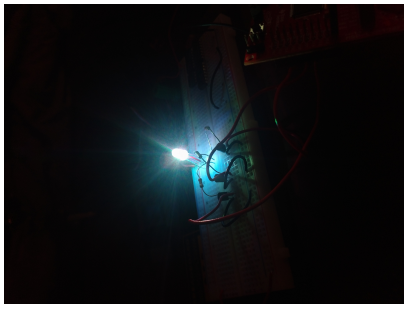


(a) Full White Data Package

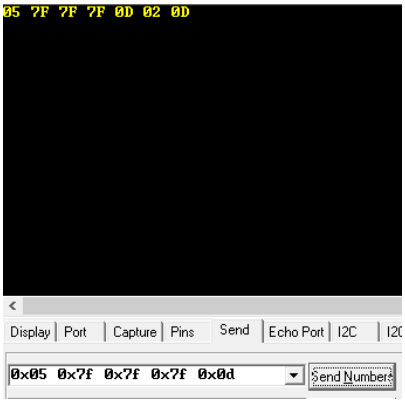


(b) Full White Data Package Return

Figure 6: Full White LED and Realterm Terminal



(a) Half White Data Package



(b) Half White Data Package Return

Figure 7: Half White LED and Realterm Terminal

(a) Long Data Package

(b) Long Data Package Return

Figure 8: Long String LED and Realterm Terminal

As shown, all of the test data was recieved and transmitted properly. For each of the single package tests, the LED properly lit up to the correct color and returned 0x02 0x0d, meaning there are no more messages. For the long string, the LED correctly lit up to the first value, and transmitted the second, as shown by the return 0x05 0xff 0xff 0x00 0x0d.

7 Design Files

Design Files can be found in the github repository located at:
https://github.com/RU09342-F18/milestone-1-the-board-with-no-name

7.1 Schematics

The Schematic of the LED driver circuit used can be found in the detailed block diagram section of this Application Note.

7.2 Bill of Materials

Used materials include 1 RGB LED, the MSP430F5529, 3 2N7000 n channel small signal MOSFETs, 1 200 ohm 5 percent resistor, 1 180 ohm 5 percent resistor, 1 150 ohm 5 percent resistor, a UART cable, and a breadboard.