Milestone 1: Addressable LED

Tanner Smith, Gray Thurston, Phil Quinn
Rowan University

Dr. Schmalzel
Russell Trafford
Section 4
October 16, 2017

1 Overview

Addressable LEDs are an innovative way of controlling individual LEDs using minimal wires. These LEDs are most often utilized in decorative lighting applications such as computer lighting, household lighting, alcove lighting, etc. In order to implement addressability, each LED must capture bytes of color information one at a time, and pass on any bytes of information not used. RGB values are passed on from LED to LED allowing for the manipulation of the LED display as a whole. This application note describes an implementation of this design strategy using an MSP430G2553 microprocessor manufactured by Texas Instruments™.

2 Introduction

To create and design an addressable LED it is important to understand the theory and application of two major concepts, Pulse Width Modulation and UART. Within an addressable LED, Pulse Width Modulation is used to control the combination of red, green and blue while the UART communication protocol is used to receive and transmit color messages.

2.1 Pulse Width Modulation

Pulse Width Modulation is a technique used to control the power supplied to a device [1]. The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast rate. The longer the switch is on compared to off, the higher the total power supplied to the load. A quantitative

term to describe PWM is duty cycle, which refers to the portion of on time compared to the regular interval period.

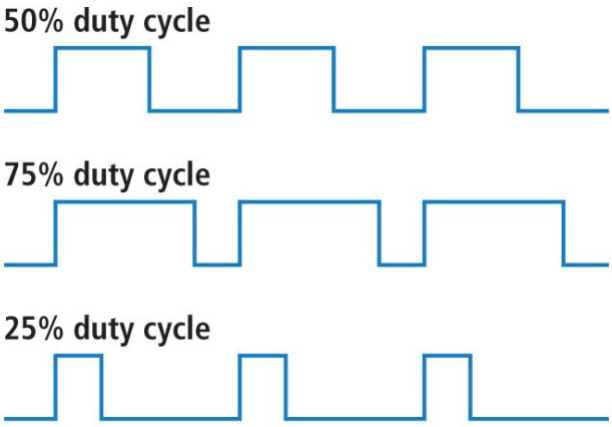

Figure 1: Duty Cycle Example using PWM

An RGB LED has the ability to output different shades of red, green and blue depending on the voltage supplied to each pin. By using Pulse Width Modulation, the led can turn on and off faster than the human eye can see. The longer the LED is turned on the brighter it looks. However, if the LED is only turned on for 10% duty cycle, the LED looks very dim. This allows a vast amount of colors to be seen through an RGB LED. By changing the brightness of red, green and blue, millions of different color combinations can be made. The addressable LED created will use 256 levels of brightness which creates over 16 million possible colors.

2.2 Universal Asynchronous Receiver Transmitter (UART) Communication Protocol

UART is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable [2]. The universal asynchronous receiver-transmitter (UART) takes bytes of data and transmits the individual bits sequentially. At the destination, a second UART re-assembles the bits into complete bytes. Each UART contains a shift register, which is the fundamental method of conversion between serial and parallel forms. Serial transmission of digital information (bits) through a single wire or other medium is less costly than parallel transmission through multiple wires.

Figure 2: UART Communication Protocol Timing Diagram

The start bit signals the receiver that a new character is coming. The next five to nine bits, depending on the code set employed, represent the character. If a parity bit is used, it would be placed after all of the data bits. The next one or two bits are always in the mark (logic high, i.e., '1') condition and called the stop bit(s). They signal the receiver that the character is completed. Since the start bit is logic low (0) and the stop bit is logic high (1) there are always at least two guaranteed signal changes between characters. If the line is held in the logic low condition for longer than a character time, this is a break condition that can be detected by the UART.

3 Application

An RGB led has three dies in a single package. One that is red, one that is blue, and one that is green. By turning on these dies with varying intensities we can create the entire color spectrum. This particular system receives 4 bytes of UART data. 1 byte per each die in the LED and then a byte at the beginning of the message containing the number of bytes in the message. With the exception of the first byte, each byte contains the information needed to create a PWM signal with the correct duty cycle to create the desired color.

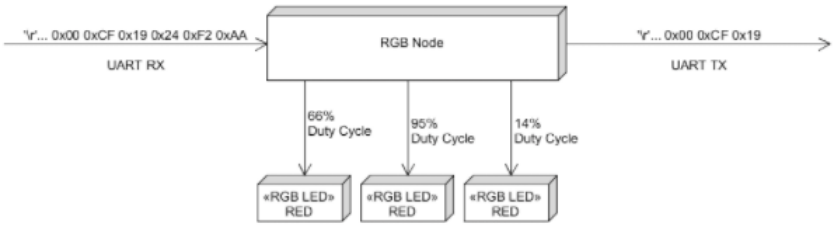

Figure 3: Diagram of Communication Protocol with Single RGB Node

By abstracting an LED into a node that receives three bytes, each individually containing portions of RGB values, it can be understood that duplication of this would allow us to string together multiple RGB values into a single signal that can be chopped away at to find RGB values to store. Following the storing of these values, they are then passed along to any succeeding nodes, or LEDs in the strip. The creation of a single node is enough to see that this design strategy can be duplicated and strung together to create an Addressable LED strip.

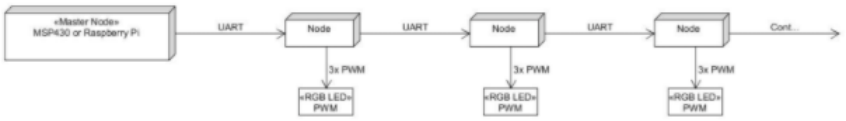

Figure 4: Diagram of Communication Protocol with Multiple RGB Nodes

As stated above each node will receive 3 bytes of data that correspond to the duty cycle of each LED in HEX. So the system as a whole would have a message that is 3bytes* 26 nodes = 78 bytes long. Additionally for debugging purposes there will be a byte at the end of the message so that our head node can determine whether or not each node removed the first 3 bytes from their received message. With the addition of the first byte containing message length information, our system wide message is 80 bytes long.

Byte Number	**Hex Value**	**Byte Meaning**
Byte 0	0x08	8 total bytes in the package
Byte 1	0x7E	Red (current node): 50% duty cycle
Byte 2	0x00	Green (current node): 0% duty cycle
Byte 3	0xFF	Blue (current node): 100% duty cycle
Byte 4	0x40	Red (next node): 25% duty cycle
Byte 5	0xFF	Green (next node): 100% duty cycle
Byte 6	0x00	Blue (next node): 0% duty cycle
Byte 7	0x0D	End of Message Check byte

Table 1: Example Communication Between Two Nodes: Message Received

Byte Number	**Hex Value**	**Byte Meaning**
Byte 0	0x05	5 total bytes in the package
Byte 1	0x40	Red (current node): 25% duty cycle
Byte 2	0xFF	Green (current node): 100% duty cycle
Byte 3	0x00	Blue (current node): 0% duty cycle
Byte 4	0x0D	End of Message Check Byte

Table 2: Example Communication Between Two Nodes: Message Transmitted

4 Functionality

4.1 Hardware

The hardware requirements for this system were fairly simple, control an RGB LED with 3 pins of a MSP430 microcontroller. First, a specific MSP430 was selected. The MSP403G2553 was selected because of the team's familiarity with it. The G2553 was the first processor used in previous experiences and labs, so it was chosen to save time and effort with configuring software. The G2553 is also the only processor that can be detached from the board so that it can be forced into a smaller package. This made the G2553 the most feasible board to use for the most practical implementation possible. The other aspect of hardware was the LED circuit. Since the forward voltage and forward current of the LED were 5v and 20mA respectively, 3 high side switches (1 per die) were built to not draw more than 6mA (G2553 max current rating) from the processor pins. Each section of the circuit contained a 2N3604 transistor and a current limiting resistor.

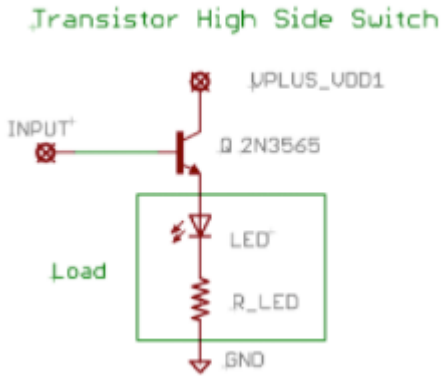

Figure 5: Example of NPN Transistor High Side Switch

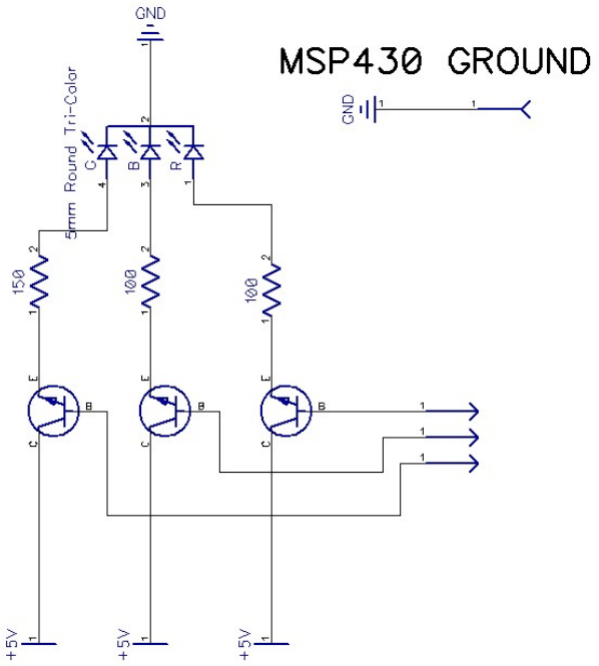

Figure 6: RGB LED Circuit Schematic

Absolute Maximum Rating (Ta=25°C)

Parameter	Symbol	Absolute Maximum Rating	Unit
Forward Pulse Current	IFPM	R :60 G: 100 B: 100	mA
Forward Current	IFM	20	mA
Reverse Voltage	VR	5	V
Power Dissipation	PD	R :60 G: 130 B: 130	mW
Operating Temperature	Topr	-40~+80	°C
Storage Temperature	Tstg	-40~+100	°C
Soldering Temperature	Tsol	Reflow Soldering : 260 °C for 10 sec. Hand Soldering : 350 °C for 3 sec.	°C

Figure 7: Device Rating of Circuit Built

4.2 Software

4.2.1 PWM

A Pulse Width Modulation module was created using C programming and Code Composer Studio. The MSP430G2553's built in hardware was used to its full potential by using its PWM features. Using a timer and a compare register, a PWM signal was directly output to a pin without using any interrupts. This was relatively simple to implement because of the well documented datasheet. Unfortunately, the MSP403G2553 only has three compare registers per timer. To properly implement PWM for an addressable LED application, four compare registers were needed, three for each color's duty cycle and one to force the timer back to zero. A quick workaround was using two timers which did not add too much complication. Timer0 and Timer1 were both set to 250kHz (SMCLK divided by 4) and UP mode with a CCR0 value of 256 (the color resolution). The value 256 was selected to reset the timer because it allowed the passed in color brightness to be set directly to the compare registers with no scaling necessary. The clock was set to be 250kHz because with a timer cap of 256, the overflow frequency of each timer became around 1kHz. Using the datasheet, the correct values for PxDIR, PxSEL and PxSEL2 were found. Pins P1.6, P2.1 and P2.4 were used for red, green and blue pulse width modulation respectively.

Pin	CCRx	PxDIR	PxSEL	PxSEL2
P1.6	TA0CCR1	1	1	0
P2.1	TA1CCR1	1	1	0
P2.4	TA1CCR2	1	1	0

Table 3: Table for PWM Mode Pin Settings

Using these pin values created a working PWM module for RGB LEDs. However, when adjusting the PWM, it was noticed that the brightness did not increase linearly. At very low duty cycle values, the brightness was much higher than expected. To correct for this, an array of values was used to set the correct linear value for duty cycle. The array was accessed using the passed in brightness value, which returned the linearized value for the PWM duty cycle.

```
static unsigned char linear_brightness_curve[256] = {
    0,   0,   0,   0,   0,   0,   0,   0,
    0,   0,   0,   0,   0,   0,   0,   0,
    0,   0,   1,   1,   1,   1,   1,   1,
    1,   1,   1,   1,   1,   1,   1,   1,
    1,   1,   1,   1,   1,   1,   1,   1,
    1,   1,   2,   2,   2,   2,   2,   2,
    2,   2,   2,   2,   2,   2,   2,   2,
    2,   3,   3,   3,   3,   3,   3,   3,
    3,   3,   3,   3,   3,   4,   4,   4,
    4,   4,   4,   4,   4,   4,   5,   5,
    5,   5,   5,   5,   5,   5,   6,   6,
    6,   6,   6,   6,   6,   7,   7,   7,
    7,   7,   8,   8,   8,   8,   8,   9,
    9,   9,   9,   9,  10,  10,  10,  10,
   11,  11,  11,  11,  12,  12,  12,  12,
   13,  13,  13,  14,  14,  14,  15,  15,
   15,  16,  16,  16,  17,  17,  18,  18,
   18,  19,  19,  20,  20,  21,  21,  22,
   22,  23,  23,  24,  24,  25,  25,  26,
   26,  27,  28,  28,  29,  30,  30,  31,
   32,  32,  33,  34,  35,  35,  36,  37,
   38,  39,  40,  40,  41,  42,  43,  44,
   45,  46,  47,  48,  49,  51,  52,  53,
   54,  55,  56,  58,  59,  60,  62,  63,
   64,  66,  67,  69,  70,  72,  73,  75,
   77,  78,  80,  82,  84,  86,  88,  90,
   91,  94,  96,  98, 100, 102, 104, 107,
  109, 111, 114, 116, 119, 122, 124, 127,
  130, 133, 136, 139, 142, 145, 148, 151,
  155, 158, 161, 165, 169, 172, 176, 180,
  184, 188, 192, 196, 201, 205, 210, 214,
  219, 224, 229, 234, 239, 244, 250, 255
};
```

Figure 8: Linearized Duty Cycle Values

4.2.2 UART

The on board UART hardware was used along with pins P1.1 (RXD) and P1.2 (TXD). The UART receive interrupt was implemented to filter the incoming messages into their proper function. The first byte was stored into a local variable, numBytes, and was used to determine how many future bytes to expect. The next three bytes were stored in red, green and blue local variables to set the PWM duty cycle at a later time. Finally, the rest of the incoming bytes were transmitted to the next node. A few design characteristics were taken into consideration when developing the filtering algorithm. First, local variables were used to store the incoming red, green and blue bytes instead of immediately setting the duty cycle. The purpose of this was to latch all colors to the LED at the same time. If all duty cycles were set immediately when received, a slight delay between the individual colors latching could be seen between receiving messages. Latching all colors at the same time minimizes the delay. The second consideration was transmitting time. All received values could have been

stored into an array and transmitted after all messages were received. Instead, while the blue byte is being received, the new packet size is already being transmitted to the next node. Then as each new byte is received, they are immediately transmitted to the next node, minimizing the message delay between nodes. All of these features were tested using the RealTerm software which allows UART packets to be sent and received over USB.

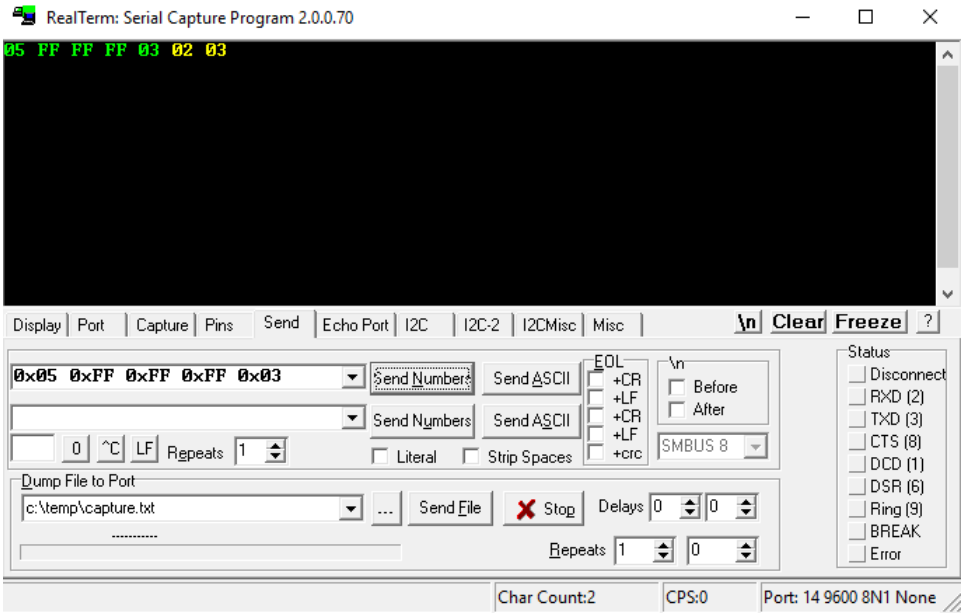

Figure 9: RealTerm Test Proving Proper UART Functionality

5 Conclusion

The developed RGB addressable LED functioned properly and is able to be strung in series with other addressable LEDs to create cool patterns and light shows. To create the best addressable LED, many design considerations were taken into consideration. On the hardware side, the MSP430G2553 was chosen due to its familiarity and ability to be taken off of its development board. Also, three high side switches were used to not draw too much current from the max 6mA pins. On the software side, the G2553 hardware was utilized to minimize interrupts and power consumption. Steps were taken to linearize the brightness curve for each color dye. Also, the time delays between latching colors and transmitting messages were minimized. All of these considerations makes the created addressable LED both functional and practical.

References

[1] JORDANDEE. Pulse Width Modulation . Pulse Width Modulation, Sparkfun, learn.sparkfun.com/tutorials/pulse-width-modulation.

[2] Adam Osborne, An Introduction to Microcomputers Volume 1: Basic Concepts, Osborne-McGraw Hill Berkeley California USA, 1980 ISBN 0-931988-34-9 pp. 116-126