Application Note - Milestone 1

Veronica Williams, Sam Toanone, and Alex Revolus
Rowan University

October 17, 2017

1 Abstract

In the past few weeks of Intro to Embedded Systems, the class has been learning how to interface with the many functionalities that the MSP40 has to offer. The functionalities range from timers, interrupts,and pwms. These tools are now being put to the test in the Milestone 1 lab: Stranger Things Light Wall. This milestone presents the opportunity for the MSP40 to communicate with other microprocessors of its kind, using the peripheral called UART. The UART(Universal asynchronous receiver-transmitter) is a computer hardware device for asynchronous serial communication in which the data format and transmission speeds are configurable. Using this new tool and the tools from the past labs, it is possible to configure RGB LEDs via the MSP40 to communicate and generate patterns to another set of RGB LEDs.

2 Background

In “Stranger Things”, Will Byers communicates with his mom through lights that have a certain letter of the alphabet associated with them. If he wants to say ”hi”, then the ’h’ and the ’i’ lights must be lit (Figure 1).
This lab will try to communicate the same way but with hex bytes and MSP430 boards. To do this the MSP430 must be able to receive hex bytes, then send the extra hex bytes to another MSP430 board. The only hex bytes that the current board will use to drive the LED are the 2nd, 3rd, and 4th bytes. These hex bytes control the PWM for each part of the RGB LED (Figure 2).

2.1 Timer/PWM

For the RGB LEDs to have the correct PWM, the timers in the MSP430 had to be utilized. In this case the timer used was Timer B. The hex values that are being sent over UART to control the LED have a max value of 255. This will be the max

Figure 1: Stranger Things Light Wall

Byte Number	Content	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

Figure 2: Byte String Structure via Russell Trafford

value for TB0CCR0. TB0CCTL1-3 will each be set to OUTMOD 3 which is set/reset mode. The clock used is SMCLK and it will be in up mode. To set TB0CCR1-3, the hex byte received over UART will be used. For the first color byte, this value is stored into TB0CCR1 by (255-UCA3RXBUF). This takes into consideration that you want your pulse width to be the difference between CCR0 (255) and the CCRx registers. UCA3RXBUF contains the current hex byte. This process is repeated for each TB0CCR2,3. If the value sent is 255, this means 100 percent duty cycle. When the value is stored into TB0CCR1 for example, it will be stored at 255-255=0. This results in the output being high until it resets at CCR0. Then at CCR1 again the output is high. This results in a 100 percent duty cycle. This is also the case for 0 percent duty cycle. The value sent will be 0 which will be stored as 255 after the subtraction. This output will only be high for this CCR register at 255 which then it is immediately reset. This will result in a 0 percent duty cycle. To output directly to the RGB LED without using an interrupt vector, the pins which have TB0CCR1, CCR2, and CCR3, will be driving the RGB LED. On the MSP430FR5549, these pins are 1.4,1.5 and 3.4. This allows the hardware to control the PWM.

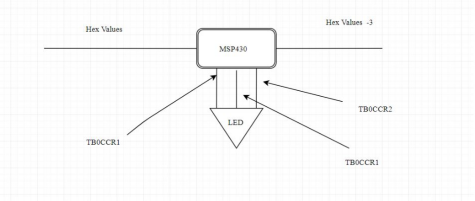

Figure 3: LED Layout

2.2 UART

To communicate with each processor, the UART function had to be used. The UART function allows data to be sent and received between processors and the computer. At a Baud Rate of 9600, the data sent will be 8 bytes of data. These bytes are hex values between 0 and 255. When the first byte is received by the processor, the number is saved. This byte tells the processor how many bytes in total will be sent. The next 3 bytes control the RGB values of the RGB LED. The final 4 bytes are sent to the next processor. The final byte is an end byte with the hex number '0x0D'. This number tells the processor the message is done. When the message is being sent to the next processor, the first value has to be subtracted by 3. This is because 3 bytes were used by the current processor. This process is repeated until there are no bytes between the first and last byte.

To store the data being sent, the UCA3RXBUF register is used. When the first byte is sent, this register stores the hex value. To transmit the correct number of bytes to the next processor, this number must be subtracted by 3, as stated previously. UCA3TXBUF is the register that transmits the data. UCA3TXBUF will be equal to the new value in UCA3RXBUF. Then for the next 3 bytes the transmit register will not transmit the bytes. Once byte count is greater than 3, or past the current LED duty cycle bytes, the transmit register transmits all the bytes it receives.

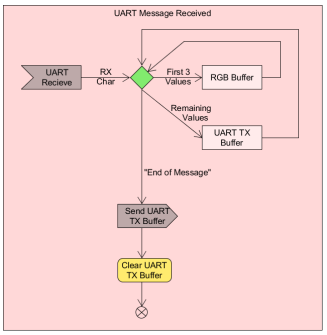

Figure 4: UART Process via Russell Trafford

3 Evaluation and Results

The MSP430 FR5994 was able to control the outputs of the RGB Led given a hex string. The hex string given to the MSP 430 was 0x08 0xFF 0XFF 0XFF 0X00 0X00 0X00 0X0D, and the Hex number sent back was 0x05 0x00 0x00 0x00 0x0D. The MSP 430 was supposed to take the first byte of information and subtract 3, then echo it back. The next 3 bytes control the RGB led. These bytes are not echoed back. The final 4 bytes are used to control the RGB led for the next node and to tell the MSP 430 when the message is done (0x0D). According to the color the RGB LED was and the echo that was sent, the code worked. The correct echo was sent back and the RGB LED changed to the correct color.

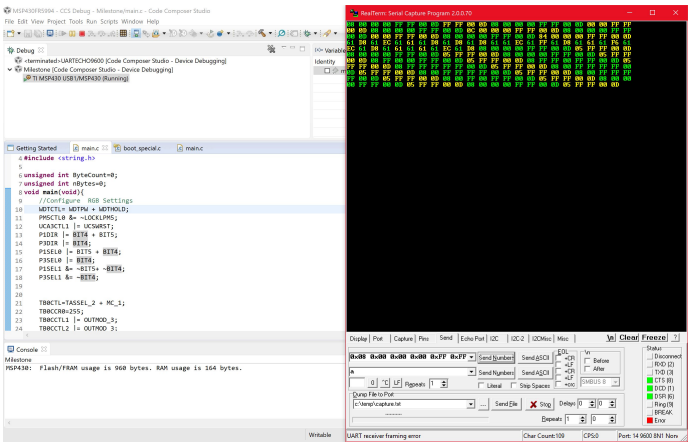

Figure 5: UART Transmit and Receive

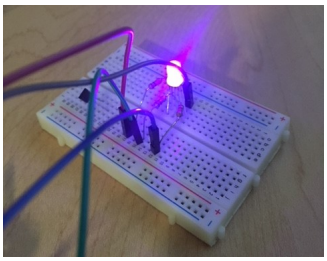

Figure 6: LED Displaying Received Code